

Artificial Intelligence 2

Summer Semester 2025

– Lecture Notes –

Prof. Dr. Michael Kohlhase
Professur für Wissensrepräsentation und -verarbeitung
Informatik, FAU Erlangen-Nürnberg
Michael.Kohlhase@FAU.de

2025-05-14

Chapter 20

Preliminaries

What you should learn here...

► What you should learn in AI-2:

- In the broadest sense: *A bunch of tools for your toolbox* (i.e. various (quasi-mathematical) models, first and foremost)

What you should learn here...

► What you should learn in AI-2:

- In the broadest sense: *A bunch of tools for your toolbox* (i.e. various (quasi-mathematical) models, first and foremost)
- the underlying *principles* of these models (assumptions, limitations, the math behind them ...)

What you should learn here...

► What you should learn in AI-2:

- In the broadest sense: *A bunch of tools for your toolchest* (i.e. various (quasi-mathematical) models, first and foremost)
- the underlying *principles* of these models (assumptions, limitations, the math behind them ...)
- the ability to describe real-world problems in terms of these models, **where adequate** (...and knowing **when they are adequate!**),

What you should learn here...

► What you should learn in AI-2:

- In the broadest sense: *A bunch of tools for your toolchest* (i.e. various (quasi-mathematical) models, first and foremost)
- the underlying *principles* of these models (assumptions, limitations, the math behind them ...)
- the ability to describe real-world problems in terms of these models, **where adequate** (...and knowing **when they are adequate!**), and
- the ideas behind effective *algorithms* that solve these problems (and to understand them well enough to implement them)
- **Note:** You will likely never get paid to implement an algorithm that e.g. solves Bayesian networks. (They already exist)
- *But* you might get paid to *recognize* that some given problem *can be* represented as a Bayesian network!
- **Or:** you can recognize that it is *similar to* a Bayesian network, and reuse the underlying principles to develop new specialized tools.

- ▶ **“We have the following problem and we need a solution: ...”**

- ▶ “We have the following problem and we need a solution: ...”
- ▶ **Employee 1 – Deep Learning can do everything:** “I just need ≈ 1.5 million labeled examples of potentially sensitive data, a GPU cluster for training, and a few weeks to train, tweak and finetune the model.
But *then* I can solve the problem... with a confidence of 95%, within 40 seconds of inference per input. Oh, as long as the input isn't longer than 15unit, or I will need to retrain on a bigger input layer...”

- ▶ “We have the following problem and we need a solution: ...”
- ▶ **Employee 1 – Deep Learning can do everything:** “I just need ≈ 1.5 million labeled examples of potentially sensitive data, a GPU cluster for training, and a few weeks to train, tweak and finetune the model.
But *then* I can solve the problem... with a confidence of 95%, within 40 seconds of inference per input. Oh, as long as the input isn't longer than 15unit, or I will need to retrain on a bigger input layer...”
- ▶ **Employee 2 – AI-2 Alumna:** “...while you were talking, I quickly built a custom UI for an off-the-shelf <problem> solver that runs on a medium-sized potato and returns a *provably correct* result in a few milliseconds. For inputs longer than 1000unit, you might need a slightly bigger potato though...”

Compare two employees

- ▶ “We have the following problem and we need a solution: ...”
- ▶ **Employee 1 – Deep Learning can do everything:** “I just need ≈ 1.5 million labeled examples of potentially sensitive data, a GPU cluster for training, and a few weeks to train, tweak and finetune the model.
But *then* I can solve the problem... with a confidence of 95%, within 40 seconds of inference per input. Oh, as long as the input isn't longer than 15unit, or I will need to retrain on a bigger input layer...”
- ▶ **Employee 2 – AI-2 Alumna:** “...while you were talking, I quickly built a custom UI for an off-the-shelf <problem> solver that runs on a medium-sized potato and returns a *provably correct* result in a few milliseconds. For inputs longer than 1000unit, you might need a slightly bigger potato though...”
- ▶ **Moral of the story:** Know your *tools* well enough to select the right one for the job.

20.1 Administrative Ground Rules

- ▶ **Remember:** **AI-1** dealt with situations with “complete information” and strictly computable, “perfect” solutions to problems. (i.e. tree search, logical inference, planning, etc.)
- ▶ **AI-2** will focus on *probabilistic* scenarios by introducing uncertain situations, and *approximate* solutions to problems. (Bayesian networks, Markov models, machine learning, etc.)

- ▶ **Remember:** **AI-1** dealt with situations with “complete information” and strictly computable, “perfect” solutions to problems. (i.e. tree search, logical inference, planning, etc.)
- ▶ **AI-2** will focus on *probabilistic* scenarios by introducing uncertain situations, and *approximate* solutions to problems. (Bayesian networks, Markov models, machine learning, etc.)
- ▶ **Weak Prerequisites for AI-2:** (if you do not have them, study up as needed)
 - ▶ AI-1 (in particular: PEAS, propositional logic/first-order logic (mostly the syntax), some logic programming)
 - ▶ (very) elementary complexity theory. (big Oh and friends)

- ▶ **Remember:** **AI-1** dealt with situations with “complete information” and strictly computable, “perfect” solutions to problems. (i.e. tree search, logical inference, planning, etc.)
- ▶ **AI-2** will focus on *probabilistic* scenarios by introducing uncertain situations, and *approximate* solutions to problems. (Bayesian networks, Markov models, machine learning, etc.)
- ▶ **Weak Prerequisites for AI-2:** (if you do not have them, study up as needed)
 - ▶ AI-1 (in particular: PEAS, propositional logic/first-order logic (mostly the syntax), some logic programming)
 - ▶ (very) elementary complexity theory. (big Oh and friends)
 - ▶ rudimentary probability theory (e.g. from stochastics)

- ▶ **Remember:** **AI-1** dealt with situations with “complete information” and strictly computable, “perfect” solutions to problems. (i.e. tree search, logical inference, planning, etc.)
- ▶ **AI-2** will focus on *probabilistic* scenarios by introducing uncertain situations, and *approximate* solutions to problems. (Bayesian networks, Markov models, machine learning, etc.)
- ▶ **Weak Prerequisites for AI-2:** (if you do not have them, study up as needed)
 - ▶ AI-1 (in particular: PEAS, propositional logic/first-order logic (mostly the syntax), some logic programming)
 - ▶ (very) elementary complexity theory. (big Oh and friends)
 - ▶ rudimentary probability theory (e.g. from stochastics)
 - ▶ basic linear algebra (vectors, matrices,...)

- ▶ **Remember:** **AI-1** dealt with situations with “complete information” and strictly computable, “perfect” solutions to problems. (i.e. tree search, logical inference, planning, etc.)
- ▶ **AI-2** will focus on *probabilistic* scenarios by introducing uncertain situations, and *approximate* solutions to problems. (Bayesian networks, Markov models, machine learning, etc.)
- ▶ **Weak Prerequisites for AI-2:** (if you do not have them, study up as needed)
 - ▶ AI-1 (in particular: PEAS, propositional logic/first-order logic (mostly the syntax), some logic programming)
 - ▶ (very) elementary complexity theory. (big Oh and friends)
 - ▶ rudimentary probability theory (e.g. from stochastics)
 - ▶ basic linear algebra (vectors, matrices,...)
 - ▶ basic real analysis (aka. calculus) (primarily: (partial) derivatives)

- ▶ **Remember:** **AI-1** dealt with situations with “complete information” and strictly computable, “perfect” solutions to problems. (i.e. tree search, logical inference, planning, etc.)
- ▶ **AI-2** will focus on *probabilistic* scenarios by introducing uncertain situations, and *approximate* solutions to problems. (Bayesian networks, Markov models, machine learning, etc.)
- ▶ **Weak Prerequisites for AI-2:** (if you do not have them, study up as needed)
 - ▶ AI-1 (in particular: PEAS, propositional logic/first-order logic (mostly the syntax), some logic programming)
 - ▶ (very) elementary complexity theory. (big Oh and friends)
 - ▶ rudimentary probability theory (e.g. from stochastics)
 - ▶ basic linear algebra (vectors, matrices,...)
 - ▶ basic real analysis (aka. calculus) (primarily: (partial) derivatives)
- ▶ **Meaning:** I will *assume* you know these things, but some of them we will recap, and what you don't know will make things slightly harder for you, but by no means prohibitively difficult.

“Strict” Prerequisites

- ▶ **Most crucially – Mathematical Literacy:** Mathematics is the language that computer scientists express their ideas in!
 (“*A search problem is a tuple (N, S, G, \dots) such that...*”)
- ▶ **Note:** This is a skill that can be *learned*, and more importantly, *practiced*! Not having/honing this skill *will* make things more difficult for you. Be aware of this and, if necessary, work on it – it will pay off, not only in this [course](#).

“Strict” Prerequisites

- ▶ **Most crucially – Mathematical Literacy:** Mathematics is the language that computer scientists express their ideas in! (“*A search problem is a tuple (N, S, G, \dots) such that...*”)
- ▶ **Note:** This is a skill that can be *learned*, and more importantly, *practiced*! Not having/honing this skill *will* make things more difficult for you. Be aware of this and, if necessary, work on it – it will pay off, not only in this [course](#).
- ▶ **But also:** Motivation, interest, curiosity, hard work. (AI-2 is non-trivial)

“Strict” Prerequisites

- ▶ **Most crucially – Mathematical Literacy:** Mathematics is the language that computer scientists express their ideas in!
(“*A search problem is a tuple (N, S, G, \dots) such that...*”)
- ▶ **Note:** This is a skill that can be *learned*, and more importantly, *practiced*! Not having/honing this skill *will* make things more difficult for you. Be aware of this and, if necessary, work on it – it will pay off, not only in this *course*.
- ▶ **But also:** Motivation, interest, curiosity, hard work. (AI-2 is non-trivial)
- ▶ **Note:** Grades correlate significantly with invested effort; including, but not limited to:
 - ▶ time spent on exercises, (learning is 80% perspiration, only 20% inspiration)
 - ▶ being here in presence, (humans are social animals ↔ mirror neurons)
 - ▶ asking questions, (Q/A dialogues activate brains)

“Strict” Prerequisites

- ▶ **Most crucially – Mathematical Literacy:** Mathematics is the language that computer scientists express their ideas in! (“*A search problem is a tuple (N, S, G, \dots) such that...*”)
- ▶ **Note:** This is a skill that can be *learned*, and more importantly, *practiced*! Not having/honing this skill *will* make things more difficult for you. Be aware of this and, if necessary, work on it – it will pay off, not only in this *course*.
- ▶ **But also:** Motivation, interest, curiosity, hard work. (AI-2 is non-trivial)
- ▶ **Note:** Grades correlate significantly with invested effort; including, but not limited to:
 - ▶ time spent on exercises, (learning is 80% perspiration, only 20% inspiration)
 - ▶ being here in presence, (humans are social animals ↔ mirror neurons)
 - ▶ asking questions, (Q/A dialogues activate brains)
 - ▶ talking to your peers, (pool your insights, share your triumphs/frustrations)...All of these we try to support with the ALEA system. (which also gives us the data to prove this)

► Overall (Module) Grade:

- Grade via the exam (Klausur) \leadsto 100% of the grade.
- Up to 10% bonus on-top for an exam with $\geq 50\%$ points.
- Bonus points $\hat{=}$ percentage sum of the best 10 prepquizzes divided by 100.


($< 50\% \leadsto$ no bonus)

► Overall (Module) Grade:

- Grade via the exam (Klausur) \leadsto 100% of the grade.
- Up to 10% bonus on-top for an exam with $\geq 50\%$ points. ($< 50\% \leadsto$ no bonus)
- Bonus points $\hat{=}$ percentage sum of the best 10 prepquizzes divided by 100.

► **Exam:** exam conducted in presence on paper! (\sim Oct. 10. 2025)

► **Retake Exam:** 90 minutes exam six months later. (\sim April 10. 2026)

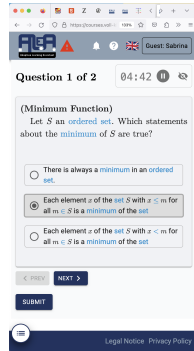
►  You have to register for exams in <https://campo.fau.de> in the first month of classes.

► **Note:** You can de-register from an exam on <https://campo.fau.de> up to three working days before exam. (do not miss that if you are not prepared)

Preparedness Quizzes

- ▶ **PrepQuizzes:** Before every lecture we offer a 10 min online quiz – the **PrepQuiz** – about the material from the previous week. (16:15-16:25; starts in week 2)
- ▶ **Motivations:** We do this to
 - ▶ keep you prepared and working continuously.
 - ▶ bonus points if the exam has $\geq 50\%$ points
 - ▶ update the **ALEA learner model**. (primary)(potential part of your grade)
(fringe benefit)
- ▶ The **prepquizzes** will be given in the **ALEA** system

- ▶ <https://courses.voll-ki.fau.de/quiz-dash/ai-2>
- ▶ You have to be **logged into ALEA!** (via FAU IDM)
- ▶ You can take the **prepquiz** on your laptop or phone, ...
- ▶ ...in the **lecture** or at home ...
- ▶ ...via WLAN or 4G Network. (do not overload)
- ▶ **Prepquizzes** will only be available 16:15-16:25!



- ▶ Some **degree programs** do not “import” the **course** Artificial Intelligence 1, and thus you may not be able to register for the **exam** via <https://campo.fau.de>.
 - ▶ Just send me an e-mail and come to the **exam**, (we do the necessary admin)
 - ▶ Tell your **program** coordinator about AI-1/2 so that they remedy this situation
- ▶ In “Wirtschafts-Informatik” you can only take AI-1 and AI-2 together in the “Wahlpflichtbereich”.
 - ▶ **ECTS credits** need to be divisible by five $\Leftarrow 7.5 + 7.5 = 15$.

20.2 Getting Most out of AI-2

AI-2 Homework Assignments

- ▶ **Goal:** Homework assignments reinforce what was taught in lectures.
- ▶ **Homework Assignments:** Small individual problem/programming/proof task
 - ▶ but take time to solve (at least read them directly \leadsto questions)
- ▶ **Didactic Intuition:** Homework assignments give you material to test your understanding and show you how to apply it.
- ▶ **⚠ Homeworks** give no points, but without trying you are unlikely to pass the exam.
- ▶ **Our Experience:** Doing your homework is probably even *more* important (and predictive of exam success) than attending the lecture in person!

AI-2 Homework Assignments

- ▶ **Goal:** Homework assignments reinforce what was taught in lectures.
- ▶ **Homework Assignments:** Small individual problem/programming/proof task
 - ▶ but take time to solve (at least read them directly \leadsto questions)
- ▶ **Didactic Intuition:** Homework assignments give you material to test your understanding and show you how to apply it.
- ▶ **⚠ Homeworks** give no points, but without trying you are unlikely to pass the exam.
- ▶ **Our Experience:** Doing your homework is probably even *more* important (and predictive of exam success) than attending the lecture in person!
- ▶ Homeworks will be mainly peer-graded in the ALEA system.
- ▶ **Didactic Motivation:** Through peer grading students are able to see mistakes in their thinking and can correct any problems in future assignments. By grading assignments, students may learn how to complete assignments more accurately and how to improve their future results. (not just us being lazy)

AI-2 Homework Assignments – Howto

- ▶ **Homework Workflow:** in **ALEA** (see below)
 - ▶ Homework assignments will be published on thursdays: see <https://courses.voll-ki.fau.de/hw/ai-1>
 - ▶ Submission of solutions via the **ALEA** system in the week after
 - ▶ Peer grading/feedback (and master solutions) via answer classes.
- ▶ **Quality Control:** TAs and instructors will monitor and supervise peer grading.

AI-2 Homework Assignments – Howto

- ▶ **Homework Workflow:** in **ALEA** (see below)
 - ▶ Homework assignments will be published on thursdays: see <https://courses.voll-ki.fau.de/hw/ai-1>
 - ▶ Submission of solutions via the **ALEA** system in the week after
 - ▶ Peer grading/feedback (and master solutions) via answer classes.
- ▶ **Quality Control:** TAs and instructors will monitor and supervise peer grading.
- ▶ **Experiment:** Can we motivate enough of you to make peer assessment self-sustaining?
 - ▶ I am appealing to your sense of community responsibility here ...
 - ▶ You should only expect other's to grade your submission if you grade their's (cf. Kant's "Moral Imperative")
- ▶ **Make no mistake:** The grader usually learns at least as much as the grader.

AI-2 Homework Assignments – Howto

- ▶ **Homework Workflow:** in [ALEA](#) (see below)
 - ▶ [Homework assignments](#) will be published on thursdays: see <https://courses.voll-ki.fau.de/hw/ai-1>
 - ▶ Submission of solutions via the [ALEA](#) system in the week after
 - ▶ [Peer grading/feedback](#) (and master solutions) via answer classes.
- ▶ **Quality Control:** [TAs](#) and [instructors](#) will monitor and supervise [peer grading](#).
- ▶ **Experiment:** Can we motivate enough of you to make [peer assessment](#) self-sustaining?
 - ▶ I am appealing to your sense of community responsibility here ...
 - ▶ You should only expect other's to [grade](#) your submission if you [grade](#) their's (cf. Kant's "Moral Imperative")
- ▶ **Make no mistake:** The [grader](#) usually [learns](#) at least as much as the [gradee](#).
- ▶ **Homework/Tutorial Discipline:**
 - ▶ [Start early!](#) (many assignments need more than one evening's work)
 - ▶ Don't start by sitting at a blank screen (talking & study groups help)
 - ▶ Humans will be trying to understand the text/code/math when [grading](#) it.
 - ▶ [Go to the tutorials, discuss with your TA!](#) (they are there for you!)

Tutorials for Artificial Intelligence 1

- ▶ **Approach:** Weekly **tutorials** and **homework assignments** (first one in week two)
- ▶ **Goal 1:** Reinforce what was taught in the **lectures**. (you need practice)
- ▶ **Goal 2:** Allow you to ask any question you have in a protected environment.

Tutorials for Artificial Intelligence 1

- ▶ **Approach:** Weekly **tutorials** and **homework assignments** (first one in week two)
- ▶ **Goal 1:** Reinforce what was taught in the **lectures**. (you need practice)
- ▶ **Goal 2:** Allow you to ask any question you have in a protected environment.
- ▶ **Instructor/Lead TA:** Florian Rabe (**KWARC** Postdoc, Privatdozent)
 - ▶ Room: 11.137 @ Händler building, florian.rabe@fau.de
- ▶ **Tutorials:** One each taught by Florian Rabe (lead); Primula Mukherjee, Ilhaam Shaikh, Praveen Kumar Vadlamani, and Shreya Rajesh More.
 - ▶ Tutorials will start in week 3. (before there is nothing to do)
 - ▶ Details (rooms, times, etc) will be announced in time (i.e. not now) on the forum and matrix channel.
- ▶ **Life-saving Advice:** Go to your **tutorial**, and prepare for it by having looked at the slides and the **homework assignments**!

- ▶ **Definition 2.1.** **Collaboration** (or **cooperation**) is the process of groups of **agents** **acting** together for common, mutual benefit, as opposed to **acting** in **competition** for selfish benefit. In a **collaboration**, every **agent** contributes to the common goal and benefits from the contributions of others.
- ▶ In **learning** situations, the benefit is “better **learning**”.
- ▶ **Observation:** In **collaborative learning**, the overall result can be significantly better than in **competitive learning**.
- ▶ **Good Practice:** Form **study groups**. (long- or short-term)
 1. ⚠ Those **learners** who work/help most, **learn** most!
 2. ⚠ Freeloaders – individuals who only watch – **learn** very little!
- ▶ It is OK to **collaborate** on **homework assignments** in AI-2! (no bonus points)
- ▶ Choose your **study group** well! (ALeA helps via the study buddy feature)

Do I need to attend the AI-2 Lectures

- ▶ Attendance is not mandatory for the AI-2 course. (official version)
 - ▶ **Note:** There are two ways of learning: (both are OK, your mileage may vary)
 - ▶ Approach **B**: Read a book/papers (here: lecture notes)
 - ▶ Approach **I**: come to the lectures, be involved, interrupt the instructor whenever you have a question.
- The only advantage of **I** over **B** is that books/papers do not answer questions
- ▶ Approach **S**: come to the lectures and sleep does not work!
 - ▶ The closer you get to research, the more we need to discuss!

20.3 Learning Resources for AI-2

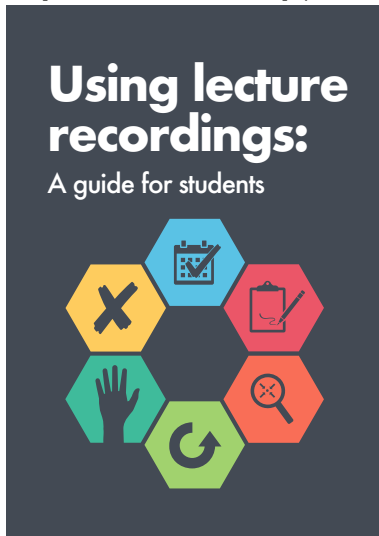
► **Textbook:** *Russel/Norvig: Artificial Intelligence, A modern Approach* [RusNor:AIMA09].

- basically “broad but somewhat shallow”
- great to get intuitions on the basics of AI

Make sure that you read the **edition ≥ 3** \leftarrow vastly improved over ≤ 2 .

- ▶ **Lecture notes** will be posted at <https://kwarc.info/teaching/AI>
 - ▶ We mostly prepare/update them as we go along (semantically preloaded \leadsto research resource)
 - ▶ Please report any errors/shortcomings you notice. (improve for the group/successors)
- ▶ **StudOn Forum:** For announcements –
https://www.studon.fau.de/studon/goto.php?target=lcode_70Bjcaxg
- ▶ **Matrix Channel:** <https://matrix.to/#/#ai-12:fau.de> for questions, discussion with instructors and among your fellow students. (your channel, use it!)
Login via **FAU IDM** \leadsto instructions
- ▶ **Course Videos** are at <https://fau.tv/course/id/4225>.
- ▶ **Do not let the videos mislead you:** Coming to **class** is highly correlated with passing the **exam**!

- **Excellent Guide:** [NorKueRob:lcprs18] (German version at [NorKueRob:vnas18])



Attend lectures.



Take notes.



Be specific.



Catch up.



Ask for help.



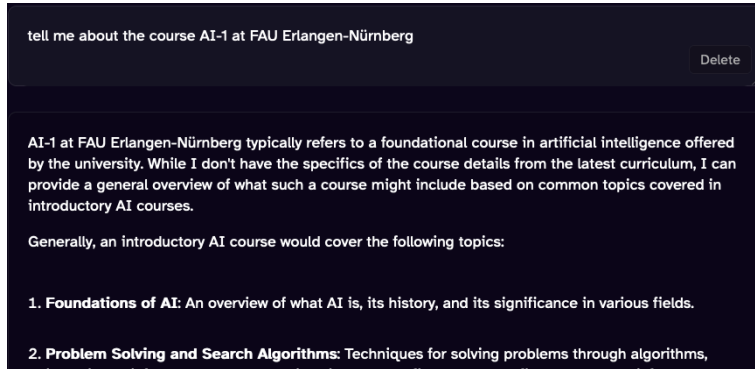
Don't cut corners.

NOT a Resource for : LLMs – AI-based tools like ChatGPT

- ▶ **Definition 3.1.** A **large language model (LLM)** is a computational model capable of language generation or other natural language processing tasks.
- ▶ **Example 3.2.** OpenAI's GPT, Google's Bard, and Meta's Llama.

NOT a Resource for : LLMs – AI-based tools like ChatGPT

- ▶ **Definition 3.6.** A **large language model (LLM)** is a computational model capable of language generation or other natural language processing tasks.
- ▶ **Example 3.7.** OpenAI's GPT, Google's Bard, and Meta's Llama.
- ▶ **Definition 3.8.** A **chatbot** is a software application or web interface that is designed to mimic human conversation through text or voice interactions. Modern **chatbots** are usually based on **LLMs**.
- ▶ **Example 3.9 (ChatGPT talks about AI-1).** (but remains vague)



NOT a Resource for : LLMs – AI-based tools like ChatGPT

- ▶ **Definition 3.11.** A **large language model (LLM)** is a computational model capable of language generation or other natural language processing tasks.
 - ▶ **Example 3.12.** OpenAI's GPT, Google's Bard, and Meta's Llama.
 - ▶ **Definition 3.13.** A **chatbot** is a software application or web interface that is designed to mimic human conversation through text or voice interactions. Modern **chatbots** are usually based on **LLMs**.
 - ▶ **Example 3.14 (ChatGPT talks about AI-1).** (but remains vague)
 - ▶ **Note:** LLM-based **chatbots** invent *every word!* (surprisingly often correct)
 - ▶ **Example 3.15 (In the AI-1 exam).** ChatGPT scores ca. 50% of the points.
 - ▶ ChatGPT can almost pass the exam ... (We could award it a Master's degree)
 - ▶ But can you? (the AI-1 exams will be in person on paper)
- You will only pass the exam, if you can do AI-1 yourself!

NOT a Resource for : LLMs – AI-based tools like ChatGPT

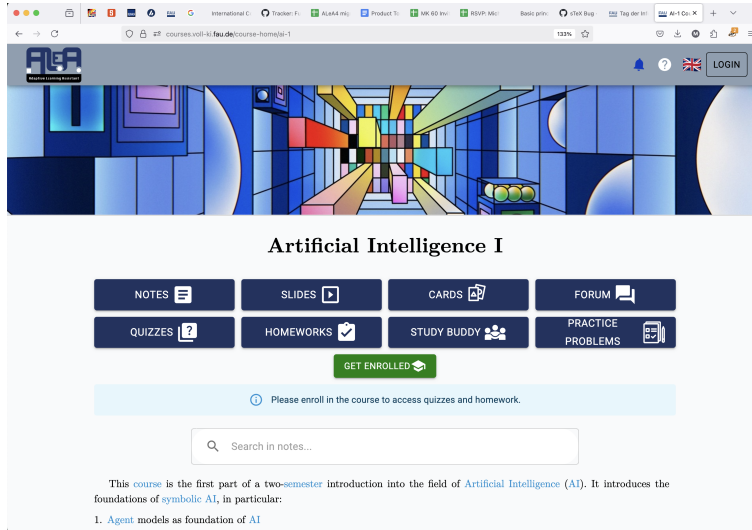
- ▶ **Definition 3.16.** A **large language model (LLM)** is a computational model capable of language generation or other natural language processing tasks.
 - ▶ **Example 3.17.** OpenAI's GPT, Google's Bard, and Meta's Llama.
 - ▶ **Definition 3.18.** A **chatbot** is a software application or web interface that is designed to mimic human conversation through text or voice interactions. Modern **chatbots** are usually based on **LLMs**.
 - ▶ **Example 3.19 (ChatGPT talks about AI-1).** (but remains vague)
 - ▶ **Note:** LLM-based **chatbots** invent *every word!* (surprisingly often correct)
 - ▶ **Example 3.20 (In the AI-1 exam).** ChatGPT scores ca. 50% of the points.
 - ▶ ChatGPT can almost pass the exam ... (We could award it a Master's degree)
 - ▶ But can you? (the AI-1 exams will be in person on paper)
- You will only pass the exam, if you can do AI-1 yourself!
- ▶ **Intuition:** AI tools like ChatGPT, CoPilot, etc. (see also [Shein:iacse24])
 - ▶ can help you solve problems, (valuable tools in production situations)
 - ▶ hinders **learning** if used for homeworks/quizzes, etc. (like driving instead of jogging)

NOT a Resource for : LLMs – AI-based tools like ChatGPT

- ▶ **Definition 3.21.** A **large language model (LLM)** is a computational model capable of language generation or other natural language processing tasks.
- ▶ **Example 3.22.** OpenAI's GPT, Google's Bard, and Meta's Llama.
- ▶ **Definition 3.23.** A **chatbot** is a software application or web interface that is designed to mimic human conversation through text or voice interactions. Modern **chatbots** are usually based on **LLMs**.
- ▶ **Example 3.24 (ChatGPT talks about AI-1).** (but remains vague)
- ▶ **Note:** LLM-based **chatbots** invent *every word!* (surprisingly often correct)
- ▶ **Example 3.25 (In the AI-1 exam).** ChatGPT scores ca. 50% of the points.
 - ▶ ChatGPT can almost pass the exam ... (We could award it a Master's degree)
 - ▶ But can you? (the AI-1 exams will be in person on paper)

You will only pass the exam, if you can do AI-1 yourself!
- ▶ **Intuition:** AI tools like GhatGPT, CoPilot, etc. (see also [Shein:iacse24])
 - ▶ can help you solve problems, (valuable tools in production situations)
 - ▶ hinders **learning** if used for homeworks/quizzes, etc. (like driving instead of jogging)
- ▶ **What (not) to do:** (to get most of the brave new AI-supported world)
 - ▶ try out these tools to get a first-hand intuition what they can/cannot do
 - ▶ challenge yourself while learning so that you can also do it (mind over matter!)

- We assume that you already know the ALEA system from last semester



The screenshot shows the ALEA (Adaptive Learning Environment) interface for the course 'Artificial Intelligence I'. The browser address bar shows 'courses.voll-ki.fau.de/course-home/ai-1'. The page features a colorful, abstract header image with geometric shapes and a blue and white color scheme. Below the header, the course title 'Artificial Intelligence I' is displayed. A grid of buttons provides access to various course materials: NOTES, SLIDES, CARDS, FORUM, QUIZZES, HOMEWORKS, STUDY BUDDY, and PRACTICE PROBLEMS. A green 'GET ENROLLED' button is prominently displayed. A light blue banner below the buttons contains a message: 'Please enroll in the course to access quizzes and homework.' Below this banner is a search bar labeled 'Search in notes...'. At the bottom, a paragraph of text describes the course as the first part of a two-semester introduction into Artificial Intelligence (AI), focusing on the foundations of symbolic AI. A numbered list begins with '1. Agent models as foundation of AI'.

Artificial Intelligence I

NOTES SLIDES CARDS FORUM

QUIZZES HOMEWORKS STUDY BUDDY PRACTICE PROBLEMS

GET ENROLLED

Please enroll in the course to access quizzes and homework.

Search in notes...

This course is the first part of a two-semester introduction into the field of Artificial Intelligence (AI). It introduces the foundations of symbolic AI, in particular:

1. Agent models as foundation of AI

▶ We assume that you already know the ALEA system from last semester

▶ Use it for

▶ lecture notes

(notes- vs slides-oriented)

▶ flashcards

(drill yourself on the AI-2 jargon/concepts)

▶ course forum

(questions, discussions and error reporting)

▶ solving and peer-grading homework assignments

▶ finding study groups

(you need not endure AI-2 alone)

▶ practicing with targeted problems

(e.g. from old exams)

▶ doing the prepquizzes

(before each lecture)

Chapter 21

Overview over AI and Topics of AI-II

21.1 What is Artificial Intelligence?

What is Artificial Intelligence? Definition

- ▶ **Definition 1.1 (According to Wikipedia).** Artificial Intelligence (AI) is intelligence exhibited by machines
- ▶ **Definition 1.2 (also).** Artificial Intelligence (AI) is a sub-field of CS that is concerned with the automation of intelligent behavior.
- ▶ **BUT:** it is already difficult to define intelligence precisely.
- ▶ **Definition 1.3 (Elaine Rich).** artificial intelligence (AI) studies how we can make the computer do things that humans can still do better at the moment.



What is Artificial Intelligence? Components

- ▶ **Elaine Rich:** AI studies how we can make the computer do things that humans can still do better at the moment.
- ▶ This needs a combination of

the ability to learn



What is Artificial Intelligence? Components

- ▶ **Elaine Rich:** AI studies how we can make the computer do things that humans can still do better at the moment.
- ▶ This needs a combination of

Inference



What is Artificial Intelligence? Components

- ▶ **Elaine Rich:** AI studies how we can make the computer do things that humans can still do better at the moment.
- ▶ This needs a combination of

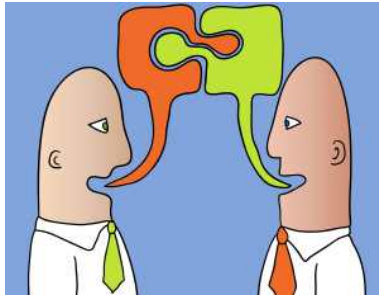
Perception



What is Artificial Intelligence? Components

- ▶ **Elaine Rich:** AI studies how we can make the computer do things that humans can still do better at the moment.
- ▶ This needs a combination of

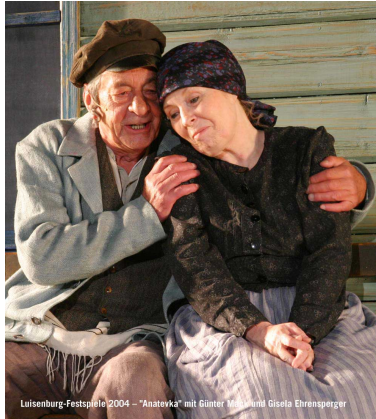
Language understanding



What is Artificial Intelligence? Components

- ▶ **Elaine Rich:** AI studies how we can make the computer do things that humans can still do better at the moment.
- ▶ This needs a combination of

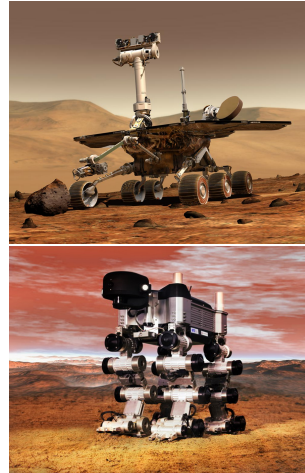
Emotion



21.2 Artificial Intelligence is here today!

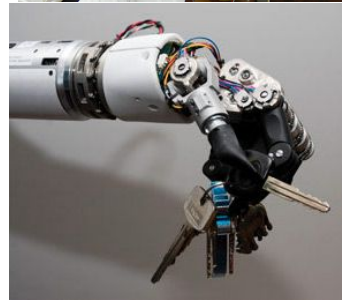
Artificial Intelligence is here today!

- ▶ in outer space
 - ▶ in outer space systems need autonomous control:
 - ▶ remote control impossible due to time lag
- ▶ in artificial limbs
- ▶ in household appliances
- ▶ in hospitals
- ▶ for safety/security



Artificial Intelligence is here today!

- ▶ in outer space
- ▶ in artificial limbs
 - ▶ the **user** controls the prosthesis via existing nerves, can e.g. grip a sheet of paper.
- ▶ in household appliances
- ▶ in hospitals
- ▶ for safety/security



Artificial Intelligence is here today!

- ▶ in outer space
- ▶ in artificial limbs
- ▶ in household appliances
 - ▶ The iRobot Roomba vacuums, mops, and sweeps in corners, . . . , parks, charges, and discharges.
 - ▶ general robotic household help is on the horizon.
- ▶ in hospitals
- ▶ for safety/security



Artificial Intelligence is here today!

- ▶ in outer space
- ▶ in artificial limbs
- ▶ in household appliances
- ▶ in hospitals
 - ▶ in the USA 90% of the prostate operations are carried out by RoboDoc
 - ▶ Paro is a cuddly robot that eases solitude in nursing homes.
- ▶ for safety/security



Artificial Intelligence is here today!

- ▶ in outer space
- ▶ in artificial limbs
- ▶ in household appliances
- ▶ in hospitals
- ▶ for safety/security
 - ▶ e.g. Intel verifies **correctness** of all chips after the "Pentium 5 disaster"



© 1999 Randy Glasbergen. www.glasbergen.com



"It's the latest innovation in office safety.
When your computer crashes, an air bag is activated."

2025-05-14

- ▶ **Observation:** Reserving the term “artificial intelligence” has been quite a land grab!
- ▶ **But:** researchers at the Dartmouth Conference (1956) really thought they would solve/reach AI in two/three decades.
- ▶ **Consequence:** AI still asks the big questions. (and still promises answers soon)
- ▶ **Another Consequence:** AI as a field is an incubator for many innovative technologies.
- ▶ **AI Conundrum:** Once AI solves a subfield it is called “CS”. (becomes a separate subfield of CS)
- ▶ **Example 2.1.** Functional/Logic Programming, automated theorem proving, Planning, machine learning, Knowledge Representation, ...
- ▶ **Still Consequence:** AI research was alternatingly flooded with money and cut off brutally.

The current AI Hype — Part of a longer Story

- ▶ The history of AI as a discipline has been very much tied to the amount of funding – that allows us to do research and development.

The current AI Hype — Part of a longer Story

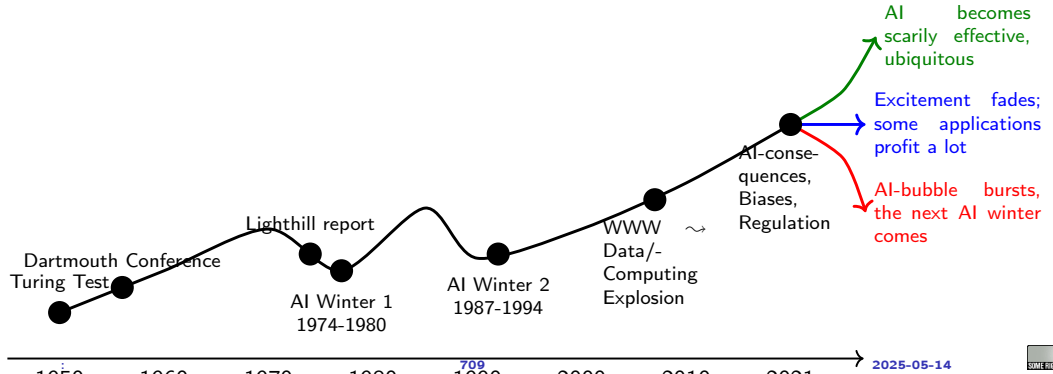
- ▶ The history of AI as a discipline has been very much tied to the amount of funding – that allows us to do research and development.
- ▶ Funding levels are tied to public perception of success (especially for AI)

The current AI Hype — Part of a longer Story

- ▶ The history of AI as a discipline has been very much tied to the amount of funding – that allows us to do research and development.
- ▶ Funding levels are tied to public perception of success (especially for AI)
- ▶ **Definition 2.4.** An AI winter is a time period of low public perception and funding for AI, mostly because AI has failed to deliver on its – sometimes overblown – promises
An AI summer is a time period of high public perception and funding for AI

The current AI Hype — Part of a longer Story

- ▶ The history of AI as a discipline has been very much tied to the amount of funding – that allows us to do research and development.
- ▶ Funding levels are tied to public perception of success (especially for AI)
- ▶ **Definition 2.5.** An AI winter is a time period of low public perception and funding for AI, mostly because AI has failed to deliver on its – sometimes overblown – promises
An AI summer is a time period of high public perception and funding for AI
- ▶ A potted history of AI (AI summers and winters)



21.3 Ways to Attack the AI Problem

Four Main Approaches to Artificial Intelligence

- **Definition 3.1.** **Symbolic AI** is a subfield of **AI** based on the assumption that many aspects of **intelligence** can be achieved by the manipulation of **symbols**, combining them into **meaning**-carrying structures (**expressions**) and manipulating them (using processes) to produce new **expressions**.

Four Main Approaches to Artificial Intelligence

- ▶ **Definition 3.5.** **Symbolic AI** is a subfield of **AI** based on the assumption that many aspects of **intelligence** can be achieved by the manipulation of **symbols**, combining them into **meaning**-carrying structures (**expressions**) and manipulating them (using processes) to produce new **expressions**.
- ▶ **Definition 3.6.** **Statistical AI** remedies the two shortcomings of **symbolic AI** approaches: that all concepts represented by **symbols** are crisply defined, and that all aspects of the world are knowable/representable in principle. **Statistical AI** adopts sophisticated **mathematical models** of **uncertainty** and uses them to create more accurate world models and reason about them.

Four Main Approaches to Artificial Intelligence

- ▶ **Definition 3.9.** **Symbolic AI** is a subfield of **AI** based on the assumption that many aspects of **intelligence** can be achieved by the manipulation of **symbols**, combining them into **meaning**-carrying structures (**expressions**) and manipulating them (using processes) to produce new **expressions**.
- ▶ **Definition 3.10.** **Statistical AI** remedies the two shortcomings of **symbolic AI** approaches: that all concepts represented by **symbols** are crisply defined, and that all aspects of the world are knowable/representable in principle. **Statistical AI** adopts sophisticated **mathematical models** of **uncertainty** and uses them to create more accurate world models and reason about them.
- ▶ **Definition 3.11.** **Subsymbolic AI** (also called **connectionism** or **neural AI**) is a subfield of **AI** that posits that **intelligence** is inherently tied to brains, where information is represented by a simple sequence pulses that are processed in parallel via simple calculations realized by neurons, and thus concentrates on neural computing.

Four Main Approaches to Artificial Intelligence

- ▶ **Definition 3.13.** **Symbolic AI** is a subfield of **AI** based on the assumption that many aspects of **intelligence** can be achieved by the manipulation of **symbols**, combining them into **meaning**-carrying structures (**expressions**) and manipulating them (using processes) to produce new **expressions**.
- ▶ **Definition 3.14.** **Statistical AI** remedies the two shortcomings of **symbolic AI** approaches: that all concepts represented by **symbols** are crisply defined, and that all aspects of the world are knowable/representable in principle. **Statistical AI** adopts sophisticated **mathematical models** of **uncertainty** and uses them to create more accurate world models and reason about them.
- ▶ **Definition 3.15.** **Subsymbolic AI** (also called **connectionism** or **neural AI**) is a subfield of **AI** that posits that **intelligence** is inherently tied to brains, where information is represented by a simple sequence pulses that are processed in parallel via simple calculations realized by neurons, and thus concentrates on neural computing.
- ▶ **Definition 3.16.** **Embodied AI** posits that **intelligence** cannot be achieved by **reasoning** about the state of the world (**symbolically**, **statistically**, or **connectivist**), but must be **embodied** i.e. situated in the world, equipped with a “body” that can interact with it via **sensors** and **actuators**. Here, the main method for realizing **intelligent behavior** is by **learning** from the world.

Two ways of reaching Artificial Intelligence?

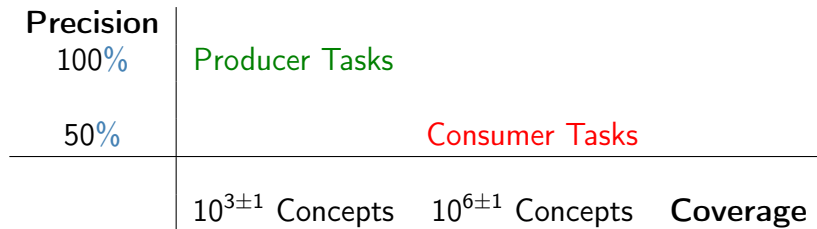
- ▶ We can classify the AI approaches by their coverage and the analysis depth (they are complementary)

Deep	symbolic AI-1	not there yet cooperation?
Shallow	no-one wants this	statistical/sub symbolic AI-2
Analysis ↑ vs. Coverage →	Narrow	Wide

- ▶ **This semester** we will cover foundational aspects of symbolic AI (deep/narrow processing)
- ▶ **next semester** concentrate on statistical/subsymbolic AI. (shallow/wide-coverage)

Environmental Niches for both Approaches to AI

- **Observation:** There are two kinds of applications/tasks in AI
 - **Consumer tasks:** consumer grade applications have tasks that must be fully generic and wide coverage. (e.g. machine translation like Google Translate)
 - **Producer tasks:** producer grade applications must be high-precision, but can be domain-specific (e.g. multilingual documentation, machinery-control, program verification, medical technology)



after Aarne Ranta [Ranta:atcp17].

- **General Rule:** Subsymbolic AI is well suited for consumer tasks, while symbolic AI is better suited for producer tasks.
- A domain of producer tasks I am interested in: mathematical/technical documents.

21.4 AI in the KWARC Group

- ▶ **Observation:** The ability to **represent knowledge** about the world and to **draw logical inferences** is one of the central components of **intelligent behavior**.
- ▶ **Thus:** reasoning components of some form are at the heart of many AI systems.
- ▶ **KWARC Angle:** Scaling up (web-coverage) without dumbing down (too much)
 - ▶ **Content markup** instead of full formalization (too tedious)
 - ▶ **User support** and **quality control** instead of “The Truth” (elusive anyway)
 - ▶ use **Mathematics** as a test tube ($\triangle \text{ Mathematics } \triangleq \text{ Anything Formal } \triangle$)
 - ▶ care more about applications than about philosophy (we cannot help getting this right anyway as logicians)
- ▶ The **KWARC** group was established at Jacobs Univ. in 2004, moved to FAU Erlangen in 2016
- ▶ See <http://kwarc.info> for projects, publications, and links

Applications: eMath 3.0, Active Documents, Active Learning, Semantic Spreadsheets/CAD/CAM, Change Management, Global Digital Math Library, Math Search Systems, [SMGloM](#): Semantic Multilingual Math Glossary, Serious Games, ...

Foundations of Math:

- ▶ [MathML](#), *OpenMath*
- ▶ advanced Type Theories
- ▶ [MMT](#): Meta Meta Theory
- ▶ Logic Morphisms/Atlas
- ▶ Theorem Prover/CAS Interoperability
- ▶ Mathematical Models/Simulation

KM & Interaction:

- ▶ Semantic Interpretation (aka. Framing)
- ▶ math-literate interaction
- ▶ [MathHub](#): math archives & active docs
- ▶ Active documents: embedded semantic services
- ▶ Model-based Education

Semantization:

- ▶ [L^AT_EXML](#): $\text{L^AT_EX} \leadsto \text{XML}$
- ▶ [S_TE_X](#): Semantic [L^AT_EX](#)
- ▶ invasive editors
- ▶ Context-Aware IDEs
- ▶ Mathematical Corpora
- ▶ Linguistics of Math
- ▶ ML for Math Semantics Extraction

Foundations: Computational Logic, Web Technologies, [OMDoc](#)/[MMT](#)

- ▶ We are always looking for bright, motivated KWARCies.
- ▶ We have topics in for all levels! (Enthusiast, Bachelor, Master, Ph.D.)
- ▶ List of current topics: <https://gl.kwarc.info/kwarc/thesis-projects/>
 - ▶ Automated Reasoning: Maths Representation in the Large
 - ▶ Logics development, (Meta)ⁿ-Frameworks
 - ▶ Math Corpus Linguistics: Semantics Extraction
 - ▶ Serious Games, Cognitive Engineering, Math Information Retrieval, Legal Reasoning, ...
 - ▶ ...last but not least: KWARC is the home of [ALeA](#)!
- ▶ We always try to find a topic at the intersection of your and our interests.
- ▶ We also sometimes have positions! (HiWi, Ph.D.: $\frac{1}{2}$ E-13, PostDoc: full E-13)

21.5 Agents and Environments in AI2

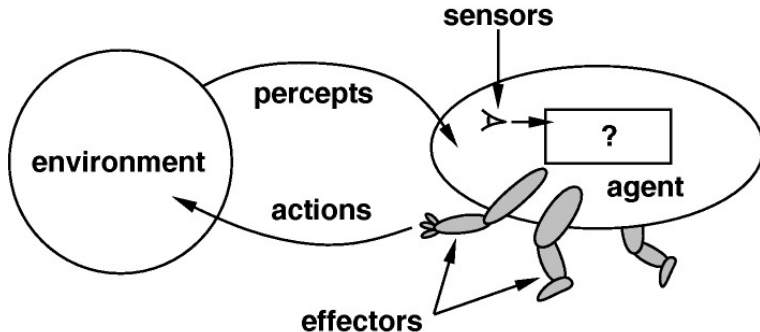
21.5.1 Recap: Rational Agents as a Conceptual Framework

Agents and Environments

► **Definition 5.1.** An **agent** is anything that

- **perceives** its **environment** via **sensors** (a means of sensing the **environment**)
- **acts** on it with **actuators** (means of changing the **environment**).

Any recognizable, coherent employment of the **actuators** of an **agent** is called an **action**.

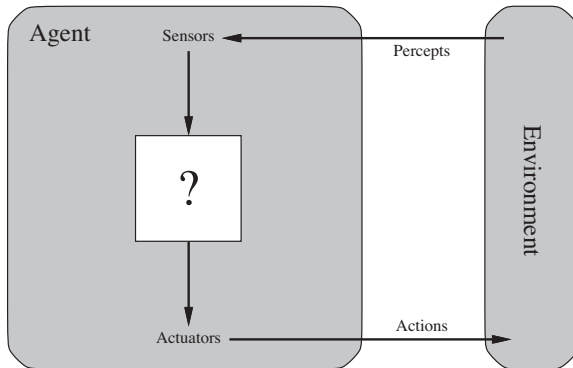


► **Example 5.2.** **Agents** include humans, robots, softbots, thermostats, etc.

► **Remark:** The notion of an **agent** and its **environment** is intentionally designed to be inclusive. We will classify and discuss subclasses of both later.

Agent Schema: Visualizing the Internal Agent Structure

- **Agent Schema:** We will use the following kind of **agent schema** to visualize the internal structure of an **agent**:



Different **agents** differ on the contents of the white box in the center.

- ▶ **Idea:** Try to design **agents** that are successful! (aka. “do the right thing”)
- ▶ **Problem:** What do we mean by “successful”, how do we measure “success”?
- ▶ **Definition 5.3.** A **performance measure** is a **function** that evaluates a sequence of **environments**.
- ▶ **Example 5.4.** A **performance measure** for a vacuum cleaner could
 - ▶ award one point per “square” cleaned up in time T ?
 - ▶ award one point per clean “square” per time step, minus one per move?
 - ▶ penalize for $> k$ dirty squares?
- ▶ **Definition 5.5.** An **agent** is called **rational**, if it chooses whichever **action** maximizes the **expected value** of the **performance measure** given the **percept** sequence to date.
- ▶ **Critical Observation:** We only need to **maximize** the **expected value**, not the actual **value** of the **performance measure**!
- ▶ **Question:** Why is **rationality** a good quality to aim for?

Consequences of Rationality: Exploration, Learning, Autonomy

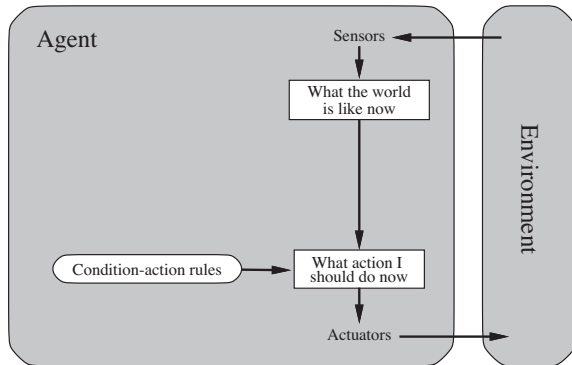
- ▶ **Note:** A rational agent need not be perfect:
 - ▶ It only needs to maximize expected value (rational \neq omniscient)
 - ▶ need not predict e.g. very unlikely but catastrophic events in the future
 - ▶ Percepts may not supply all relevant information (rational \neq clairvoyant)
 - ▶ if we cannot perceive things we do not need to react to them.
 - ▶ but we may need to try to find out about hidden dangers (exploration)
 - ▶ Action outcomes may not be as expected (rational \neq successful)
 - ▶ but we may need to take action to ensure that they do (more often) (learning)
- ▶ **Note:** Rationality may entail exploration, learning, autonomy (depending on the environment / task)
- ▶ **Definition 5.6.** An agent is called autonomous, if it does not rely on the prior knowledge about the environment of the designer.
- ▶ Autonomy avoids fixed behaviors that can become unsuccessful in a changing environment. (anything else would be irrational)
- ▶ The agent may have to learn all relevant traits, invariants, properties of the environment and actions.

- ▶ **Observation:** To design a rational agent, we must specify the task environment in terms of performance measure, environment, actuators, and sensors, together called the PEAS components.
- ▶ **Example 5.7.** When designing an automated taxi:
 - ▶ **Performance measure:** safety, destination, profits, legality, comfort, ...
 - ▶ **Environment:** US streets/freeways, traffic, pedestrians, weather, ...
 - ▶ **Actuators:** steering, accelerator, brake, horn, speaker/display, ...
 - ▶ **Sensors:** video, accelerometers, gauges, engine sensors, keyboard, GPS, ...
- ▶ **Example 5.8 (Internet Shopping Agent).** The task environment:
 - ▶ **Performance measure:** price, quality, appropriateness, efficiency
 - ▶ **Environment:** current and future WWW sites, vendors, shippers
 - ▶ **Actuators:** display to user, follow URL, fill in form
 - ▶ **Sensors:** HTML pages (text, graphics, scripts)

- ▶ **Observation 5.9.** *Agent design is largely determined by the type of environment it is intended for.*
- ▶ **Problem:** There is a vast number of possible kinds of environments in AI.
- ▶ **Solution:** Classify along a few “dimensions”. (independent characteristics)
- ▶ **Definition 5.10.** For an agent a we classify the environment e of a by its type, which is one of the following. We call e
 1. **fully observable**, iff the a 's sensors give it access to the complete state of the environment at any point in time, else **partially observable**.
 2. **deterministic**, iff the next state of the environment is completely determined by the current state and a 's action, else **stochastic**.
 3. **episodic**, iff a 's experience is divided into atomic episodes, where it perceives and then performs a single action. Crucially, the next episode does not depend on previous ones. Non-episodic environments are called **sequential**.
 4. **dynamic**, iff the environment can change without an action performed by a , else **static**. If the environment does not change but a 's performance measure does, we call e **semidynamic**.
 5. **discrete**, iff the sets of e 's state and a 's actions are countable, else **continuous**.
 6. **single-agent**, iff only a acts on e ; else **multi-agent** (when must we count parts of e as agents?)

Reflex Agents

- **Definition 5.11.** An agent $\langle \mathcal{P}, \mathcal{A}, f \rangle$ is called a **reflex agent**, iff it only takes the last **percept** into account when choosing an **action**, i.e. $f(p_1, \dots, p_k) = f(p_k)$ for all $p_1, \dots, p_k \in \mathcal{P}$.
- **Agent Schema:**

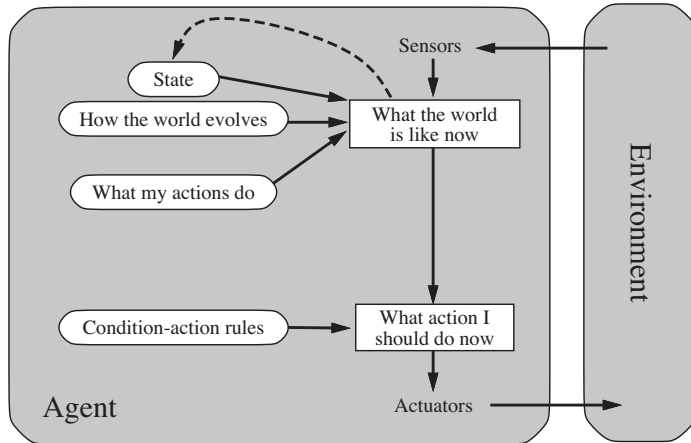


- **Example 5.12 (Agent Program).**

```
procedure Reflex-Vacuum-Agent [location,status] returns an action  
  if status = Dirty then ...
```

Model-based Reflex Agents: Idea

- **Idea:** Keep track of the state of the world we cannot see in an internal model.
- **Agent Schema:**



Model-based Reflex Agents: Definition

► **Definition 5.13.** A **model-based agent** $\langle \mathcal{P}, \mathcal{A}, \mathcal{S}, \mathcal{T}, s_0, S, a \rangle$ is an agent $\langle \mathcal{P}, \mathcal{A}, f \rangle$ whose actions depend on

1. a **world model**: a set \mathcal{S} of possible **states**, and a **start state** $s_0 \in \mathcal{S}$.
2. a **transition model** \mathcal{T} , that predicts a new **state** $\mathcal{T}(s, a)$ from a **state** s and an **action** a .
3. a **sensor model** S that given a **state** s and a **percept** p determine a new **state** $S(s, p)$.
4. an **action function** $a: \mathcal{S} \rightarrow \mathcal{A}$ that given a **state** selects the next **action**.

If the **world model** of a **model-based agent** A is in **state** s and A has last taken **action** a , and now perceives p , then A will transition to **state** $s' = S(p, \mathcal{T}(s, a))$ and take **action** $a' = a(s')$.

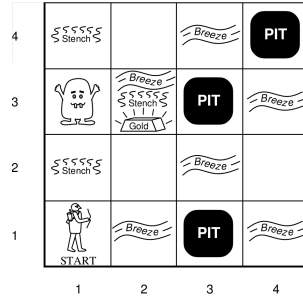
So, given a **sequence** p_1, \dots, p_n of **percepts**, we recursively define **states** $s_n = S(\mathcal{T}(s_{n-1}, a(s_{n-1})), p_n)$ with $s_1 = S(s_0, p_1)$. Then $f(p_1, \dots, p_n) = a(s_n)$.

- **Note:** As different **percept** sequences lead to different **states**, so the **agent function** $f(): \mathcal{P}^* \rightarrow \mathcal{A}$ no longer depends only on the last **percept**.
- **Example 5.14 (Tail Lights Again).** **Model-based agents** can do the ??? if the **states** include a concept of tail light brightness.

21.5.2 Sources of Uncertainty

Sources of Uncertainty in Decision-Making

Where's that d... Wumpus?
And where am I, anyway??

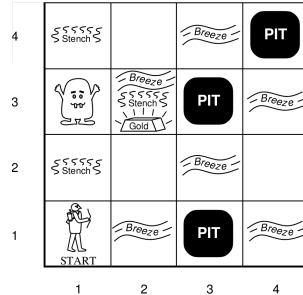


► Non-deterministic actions:

- “When I try to go forward in this dark cave, I might actually go forward-left or forward-right.”

Sources of Uncertainty in Decision-Making

Where's that d... Wumpus?
And where am I, anyway??



► Non-deterministic actions:

► “When I try to go forward in this dark cave, I might actually go forward-left or forward-right.”

► Partial observability with unreliable sensors:

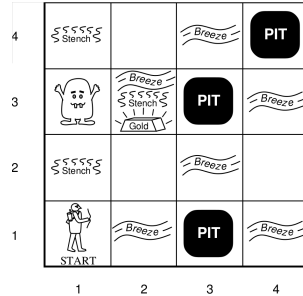
► “Did I feel a breeze right now?”;

► “I think I might smell a Wumpus here, but I got a cold and my nose is blocked.”

► “According to the heat scanner, the Wumpus is probably in cell [2,3].”

Sources of Uncertainty in Decision-Making

Where's that d... Wumpus?
And where am I, anyway??



► Non-deterministic actions:

► “When I try to go forward in this dark cave, I might actually go forward-left or forward-right.”

► Partial observability with unreliable sensors:

► “Did I feel a breeze right now?”;

► “I think I might smell a Wumpus here, but I got a cold and my nose is blocked.”

► “According to the heat scanner, the Wumpus is probably in cell [2,3].”

► Uncertainty about the domain behavior:

► “Are you *sure* the Wumpus never moves?”

- ▶ **Robot Localization:** Suppose we want to support localization using landmarks to narrow down the area.
- ▶ **Example 5.15.** *“If you see the Eiffel tower, then you’re in Paris.”*

- ▶ **Robot Localization:** Suppose we want to support localization using landmarks to narrow down the area.
- ▶ **Example 5.16.** *"If you see the Eiffel tower, then you're in Paris."*
- ▶ **Difficulty:** Sensors can be imprecise.
 - ▶ Even if a landmark is perceived, we cannot conclude with certainty that the robot is at that location.
 - ▶ *"This is the half-scale Las Vegas copy, you dummy."*
 - ▶ Even if a landmark is *not* perceived, we cannot conclude with certainty that the robot is *not* at that location.
 - ▶ *"Top of Eiffel tower hidden in the clouds."*
- ▶ Only the probability of being at a location increases or decreases.

21.5.3 Agent Architectures based on Belief States

- **Problem:** We do not know with certainty what state the world is in!

- ▶ **Problem:** We do not know with certainty what state the world is in!
- ▶ **Idea:** Just keep track of all the possible **states** it could be in.
- ▶ **Definition 5.18.** A **model-based agent** has a **world model** consisting of
 - ▶ a **belief state** that has information about the possible **states** the world may be in,
 - ▶ a **sensor model** that updates the **belief state** based on **sensor** information, and
 - ▶ a **transition model** that updates the **belief state** based on **actions**.

- ▶ **Problem:** We do not know with certainty what state the world is in!
- ▶ **Idea:** Just keep track of all the possible **states** it could be in.
- ▶ **Definition 5.19.** A **model-based agent** has a **world model** consisting of
 - ▶ a **belief state** that has information about the possible **states** the world may be in,
 - ▶ a **sensor model** that updates the **belief state** based on **sensor** information, and
 - ▶ a **transition model** that updates the **belief state** based on **actions**.
- ▶ **Idea:** The **agent environment** determines what the **world model** can be.

- ▶ **Problem:** We do not know with certainty what state the world is in!
- ▶ **Idea:** Just keep track of all the possible **states** it could be in.
- ▶ **Definition 5.20.** A **model-based agent** has a **world model** consisting of
 - ▶ a **belief state** that has information about the possible **states** the world may be in,
 - ▶ a **sensor model** that updates the **belief state** based on **sensor** information, and
 - ▶ a **transition model** that updates the **belief state** based on **actions**.
- ▶ **Idea:** The **agent environment** determines what the **world model** can be.
- ▶ In a **fully observable, deterministic environment**,
 - ▶ we can observe the initial **state** and subsequent **states** are given by the **actions** alone.
 - ▶ Thus the **belief state** is a **singleton** (we call its sole member the **world state**) and the **transition model** is a function from **states** and **actions** to **states**: a **transition function**.

World Models by Agent Type in AI-1

- ▶ **Search-based Agents:** In a fully observable, deterministic environment
 - ▶ goal-based agent with world state $\hat{=}$ "current state"
 - ▶ no inference. (goal $\hat{=}$ goal state from search problem)
- ▶ **CSP-based Agents:** In a fully observable, deterministic environment
 - ▶ goal-based agent with world state $\hat{=}$ constraint network,
 - ▶ inference $\hat{=}$ constraint propagation. (goal $\hat{=}$ satisfying assignment)
- ▶ **Logic-based Agents:** In a fully observable, deterministic environment
 - ▶ model-based agent with world state $\hat{=}$ logical formula
 - ▶ inference $\hat{=}$ e.g. DPLL or resolution.
- ▶ **Planning Agents:** In a fully observable, deterministic, environment
 - ▶ goal-based agent with world state $\hat{=}$ PL0, transition model $\hat{=}$ STRIPS,
 - ▶ inference $\hat{=}$ state/plan space search. (goal: complete plan/execution)

World Models for Complex Environments

- ▶ In a fully observable, but stochastic environment,
 - ▶ the belief state must deal with a set of possible states.
 - ▶ \leadsto generalize the transition function to a transition relation.

World Models for Complex Environments

- ▶ In a fully observable, but stochastic environment,
 - ▶ the belief state must deal with a set of possible states.
 - ▶ \leadsto generalize the transition function to a transition relation.
- ▶ **Note:** This even applies to online problem solving, where we can just perceive the state. (e.g. when we want to optimize utility)

World Models for Complex Environments

- ▶ In a fully observable, but stochastic environment,
 - ▶ the belief state must deal with a set of possible states.
 - ▶ \leadsto generalize the transition function to a transition relation.
- ▶ **Note:** This even applies to online problem solving, where we can just perceive the state. (e.g. when we want to optimize utility)
- ▶ In a deterministic, but partially observable environment,
 - ▶ the belief state must deal with a set of possible states.
 - ▶ we can use transition functions.
 - ▶ We need a sensor model, which predicts the influence of percepts on the belief state – during update.

World Models for Complex Environments

- ▶ In a **fully observable**, but **stochastic environment**,
 - ▶ the **belief state** must deal with a set of possible **states**.
 - ▶ \leadsto generalize the **transition function** to a **transition relation**.
- ▶ **Note:** This even applies to **online problem solving**, where we can just perceive the **state**. (e.g. **when we want to optimize utility**)
- ▶ In a **deterministic**, but **partially observable environment**,
 - ▶ the **belief state** must deal with a set of possible **states**.
 - ▶ we can use **transition functions**.
 - ▶ We need a **sensor model**, which predicts the influence of **percepts** on the **belief state** – during update.
- ▶ In a **stochastic, partially observable environment**,
 - ▶ mix the ideas from the last two. (sensor model + transition relation)

Preview: New World Models (Belief) \leadsto new Agent Types

- ▶ **Probabilistic Agents:** In a partially observable environment
 - ▶ belief state $\hat{=}$ Bayesian networks,
 - ▶ inference $\hat{=}$ probabilistic inference.

Preview: New World Models (Belief) \leadsto new Agent Types

- ▶ **Probabilistic Agents:** In a partially observable environment
 - ▶ belief state $\hat{=}$ Bayesian networks,
 - ▶ inference $\hat{=}$ probabilistic inference.
- ▶ **Decision-Theoretic Agents:** In a partially observable, stochastic environment
 - ▶ belief state + transition model $\hat{=}$ decision networks,
 - ▶ inference $\hat{=}$ maximizing expected utility.
- ▶ We will study them in detail this semester.

- Basics of probability theory (probability spaces, random variables, conditional probabilities, independence,...)

- ▶ Basics of probability theory (probability spaces, random variables, conditional probabilities, independence,...)
- ▶ Probabilistic reasoning: Computing the *a posteriori* probabilities of events given evidence, causal reasoning (Representing distributions efficiently, Bayesian networks,...)

- ▶ Basics of probability theory (probability spaces, random variables, conditional probabilities, independence,...)
 - ▶ Probabilistic reasoning: Computing the *a posteriori* probabilities of events given evidence, causal reasoning (Representing distributions efficiently, Bayesian networks,...)
 - ▶ Probabilistic Reasoning over time (Markov chains, Hidden Markov models,...)
- ⇒ We can update our world model episodically based on observations (i.e. sensor data)

- ▶ Basics of probability theory (probability spaces, random variables, conditional probabilities, independence,...)
- ▶ Probabilistic reasoning: Computing the *a posteriori* probabilities of events given evidence, causal reasoning (Representing distributions efficiently, Bayesian networks,...)
- ▶ Probabilistic Reasoning over time (Markov chains, Hidden Markov models,...)
- ⇒ We can update our world model episodically based on observations (i.e. sensor data)
- ▶ Decision theory: Making decisions under uncertainty (Preferences, Utilities, Decision networks, Markov Decision Procedures,...)
- ⇒ We can choose the right action based on our world model and the likely outcomes of our actions

- ▶ Basics of probability theory (probability spaces, random variables, conditional probabilities, independence,...)
- ▶ Probabilistic reasoning: Computing the *a posteriori* probabilities of events given evidence, causal reasoning (Representing distributions efficiently, Bayesian networks,...)
- ▶ Probabilistic Reasoning over time (Markov chains, Hidden Markov models,...)
- ⇒ We can update our world model episodically based on observations (i.e. sensor data)
- ▶ Decision theory: Making decisions under uncertainty (Preferences, Utilities, Decision networks, Markov Decision Procedures,...)
- ⇒ We can choose the right action based on our world model and the likely outcomes of our actions
- ▶ Machine learning: Learning from data (Decision Trees, Classifiers, Neural Networks,...)

Part 1

Reasoning with Uncertain Knowledge

Chapter 22

Quantifying Uncertainty

22.1 Probability Theory

22.1.1 Prior and Posterior Probabilities

- ▶ **Definition 1.1 (Mathematically (slightly simplified)).** A **probability space** or (**probability model**) is a pair $\langle \Omega, P \rangle$ such that:
 - ▶ Ω is a **set** of **outcomes** (called the **sample space**),
 - ▶ P is a **function** $\mathcal{P}(\Omega) \rightarrow [0,1]$, such that:
 - ▶ $P(\Omega) = 1$ and
 - ▶ $P(\bigcup_i A_i) = \sum_i P(A_i)$ for all **pairwise disjoint** $A_i \in \mathcal{P}(\Omega)$. P is called a **probability measure**.

These properties are called the **Kolmogorov axioms**.

- ▶ **Intuition:** We run some experiment, the outcome of which is any $\omega \in \Omega$.
 - ▶ For $X \subseteq \Omega$, $P(X)$ is the **probability** that the result of the experiment is *any one* of the **outcomes** in X .
 - ▶ Naturally, the **probability** that *any outcome* occurs is 1 (hence $P(\Omega) = 1$).
 - ▶ The probability of **pairwise disjoint** sets of **outcomes** should just be the sum of their probabilities.
- ▶ **Example 1.2 (Dice throws).** Assume we throw a (fair) die two times. Then the **sample space** Ω is $\{(i,j) \mid 1 \leq i,j \leq 6\}$. We define P by letting $P(\{A\}) = \frac{1}{36}$ for every $A \in \Omega$. Since the probability of any **outcome** is the same, we say P is **uniformly distributed**.

- ▶ In practice, we are rarely interested in the *specific outcome* of an experiment, but rather in some *property* of the *outcome*. This is especially true in the very common situation where we don't even *know* the precise *probabilities* of the individual *outcomes*.
- ▶ **Example 1.3.** The probability that the *sum* of our two dice throws is 7 is
$$P(\{(i, j) \in \Omega \mid i + j = 7\}) = P(\{(6, 1), (1, 6), (5, 2), (2, 5), (4, 3), (3, 4)\}) = \frac{6}{36} = \frac{1}{6}.$$
- ▶ **Definition 1.4 (Again, slightly simplified).** Let D be a *set*. A *random variable* is a *function* $X: \Omega \rightarrow D$. We call D (somewhat confusingly) the *domain* of X , denoted $\text{dom}(X)$. For $x \in D$, we define the *probability* of x as $P(X = x) := P(\{\omega \in \Omega \mid X(\omega) = x\})$.
- ▶ **Definition 1.5.** We say that a *random variable* X is *finite domain*, iff its *domain* $\text{dom}(X)$ is *finite* and *Boolean*, iff $\text{dom}(X) = \{T, F\}$. For a *Boolean random variable*, we will simply write $P(X)$ for $P(X = T)$ and $P(\neg X)$ for $P(X = F)$.

Some Examples

- ▶ **Example 1.6.** Summing up our two dice throws is a **random variable** $S: \Omega \rightarrow [2,12]$ with $S((i,j)) = i + j$. The probability that they sum up to 7 is written as $P(S = 7) = \frac{1}{6}$.
 - ▶ **Example 1.7.** The first and second of our two dice throws are **random variables** First, Second: $\Omega \rightarrow [1,6]$ with $\text{First}((i,j)) = i$ and $\text{Second}((i,j)) = j$.
 - ▶ **Remark 1.8.** Note, that the *identity* $\Omega \rightarrow \Omega$ is a **random variable** as well.
 - ▶ **Example 1.9.** We can model **toothache**, **cavity** and **gingivitis** as **Boolean random variables**, with the underlying **probability space** being...?? $\neg_(_)_/_$
 - ▶ **Example 1.10.** We can model tomorrow's weather as a **random variable** with **domain** $\{\text{sunny, rainy, foggy, warm, cloudy, humid, ...}\}$, with the underlying **probability space** being...?? $\neg_(_)_/_$
- ⇒ This is why *probabilistic reasoning* is necessary: We can rarely reduce probabilistic scenarios down to clearly defined, fully known **probability spaces** and derive all the interesting things from there.
- But:** The definitions here allow us to *reason* about **probabilities** and **random variables** in a *mathematically rigorous* way, e.g. to make our intuitions and assumptions precise, and prove our methods to be *sound*.

- ▶ This is nice and all, but in practice we are interested in “compound” probabilities like:
“What is the *probability* that the sum of our two dice throws is 7, but neither of the two dice is a 3?”
 - ▶ **Idea:** Reuse the *syntax* of *propositional logic* and define the *logical connectives* for *random variables*!
 - ▶ **Example 1.11.** We can express the above as: $P(\neg(\text{First} = 3) \wedge \neg(\text{Second} = 3) \wedge (S = 7))$
 - ▶ **Definition 1.12.** Let X_1, X_2 be *random variables*, $x_1 \in \text{dom}(X_1)$ and $x_2 \in \text{dom}(X_2)$. We define:
 1. $P(X_1 \neq x_1) := P(\neg(X_1 = x_1)) := P(\{\omega \in \Omega \mid X_1(\omega) \neq x_1\}) = 1 - P(X_1 = x_1)$.
 2. $P((X_1 = x_1) \wedge (X_2 = x_2)) := P(\{\omega \in \Omega \mid (X_1(\omega) = x_1) \wedge (X_2(\omega) = x_2)\})$
 $= P(\{\omega \in \Omega \mid X_1(\omega) = x_1\} \cap \{\omega \in \Omega \mid X_2(\omega) = x_2\})$.
 3. $P((X_1 = x_1) \vee (X_2 = x_2)) := P(\{\omega \in \Omega \mid (X_1(\omega) = x_1) \vee (X_2(\omega) = x_2)\})$
 $= P(\{\omega \in \Omega \mid X_1(\omega) = x_1\} \cup \{\omega \in \Omega \mid X_2(\omega) = x_2\})$.
- It is also common to write $P(A, B)$ for $P(A \wedge B)$
- ▶ **Example 1.13.** $P((\text{First} \neq 3) \wedge (\text{Second} \neq 3) \wedge (S = 7)) = P(\{(1, 6), (6, 1), (2, 5), (5, 2)\}) = \frac{1}{9}$

- ▶ **Definition 1.14 (Again slightly simplified).** Let $\langle \Omega, P \rangle$ be a probability space. An event is a subset of Ω .
- ▶ **Definition 1.15 (Convention).** We call an event (by extension) anything that represents a subset of Ω : any statement formed from the logical connectives and values of random variables, on which $P(\cdot)$ is defined.
- ▶ **Problem 1.1**
Remember: We can define $A \vee B := \neg(\neg A \wedge \neg B)$, $T := A \vee \neg A$ and $F := \neg T$ – is this compatible with the definition of probabilities on propositional formulae? And why is $P(X_1 \neq x_1) = 1 - P(X_1 = x_1)$?
- ▶ **Problem 1.2 (Inclusion-Exclusion-Principle)**
Show that $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$.
- ▶ **Problem 1.3**
Show that $P(A) = P(A \wedge B) + P(A \wedge \neg B)$

Conditional Probabilities

- ▶ **Observation:** As we gather new information, our beliefs (*should*) change, and thus our **probabilities**!
- ▶ **Example 1.16.** Your “probability of missing the connection train” increases when you are informed that your current train has 30 minutes delay.
- ▶ **Example 1.17.** The “probability of **cavity**” increases when the doctor is informed that the patient has a toothache.
- ▶ **Example 1.18.** The probability that $S = 3$ is clearly higher if I know that $\text{First} = 1$ than otherwise – or if I know that $\text{First} = 6$!
- ▶ **Definition 1.19.** Let A and B be **events** where $P(B) \neq 0$. The **conditional probability** of A **given** B is defined as:

$$P(A \mid B) := \frac{P(A \wedge B)}{P(B)}$$

We also call $P(A)$ the **prior probability** of A , and $P(A \mid B)$ the **posterior probability**.

- ▶ **Intuition:** If we *assume* B to hold, then we are only interested in the “part” of Ω where A is true *relative to* B .
- ▶ **Alternatively:** We restrict our **sample space** Ω to the subset of **outcomes** where B holds. We then define a new **probability space** on this subset by scaling the **probability measure** so that it sums to 1 – which we do by dividing by $P(B)$.
(We “**update our beliefs based on new evidence**”)

Examples

- **Example 1.20.** If we assume $\text{First} = 1$, then $P(S = 3 \mid (\text{First} = 1))$ should be precisely $P(\text{Second} = 2) = \frac{1}{6}$. We check:

$$P(S = 3 \mid (\text{First} = 1)) = \frac{P((S = 3) \wedge (\text{First} = 1))}{P(\text{First} = 1)} = \frac{1/36}{1/6} = \frac{1}{6}$$

- **Example 1.21.** Assume the **prior probability** $P(\text{cavity})$ is 0.122. The **probability** that a patient has both a **cavity** and a **toothache** is $P(\text{cavity} \wedge \text{toothache}) = 0.067$. The **probability** that a patient has a **toothache** is $P(\text{toothache}) = 0.15$.

If the patient complains about a **toothache**, we can update our estimation by computing the **posterior probability**:

$$P(\text{cavity} \mid \text{toothache}) = \frac{P(\text{cavity} \wedge \text{toothache})}{P(\text{toothache})} = \frac{0.067}{0.15} = 0.45.$$

- **Note:** We just computed the probability of some underlying *disease* based on the presence of a *symptom*!
- **More Generally:** We computed the probability of a *cause* from observing its *effect*.

- ▶ Equations on **unconditional probabilities** have direct analogues for **conditional probabilities**.

- ▶ **Problem 1.4**

Convince yourself of the following:

- ▶ $P(A \mid C) = 1 - P(\neg A \mid C)$.
- ▶ $P(A \mid C) = P(A \wedge B \mid C) + P(A \wedge \neg B \mid C)$.
- ▶ $P(A \vee B \mid C) = P(A \mid C) + P(B \mid C) - P(A \wedge B \mid C)$.

- ▶ But **not on the right hand side!**

- ▶ **Problem 1.5**

Find *counterexamples* for the following (**false**) claims:

- ▶ $P(A \mid C) = 1 - P(A \mid \neg C)$
- ▶ $P(A \mid C) = P(A \mid (B \wedge C)) + P(A \mid (B \wedge \neg C))$.
- ▶ $P(A \mid (B \vee C)) = P(A \mid B) + P(A \mid C) - P(A \mid (B \wedge C))$.

Bayes' Rule

- **Note:** By definition, $P(A | B) = \frac{P(A \wedge B)}{P(B)}$. In practice, we often know the conditional probability already, and use it to compute the probability of the conjunction instead:
- $$P(A \wedge B) = P(A | B) \cdot P(B) = P(B | A) \cdot P(A).$$

Bayes' Rule

- **Note:** By definition, $P(A | B) = \frac{P(A \wedge B)}{P(B)}$. In practice, we often know the conditional probability already, and use it to compute the probability of the conjunction instead:

$$P(A \wedge B) = P(A | B) \cdot P(B) = P(B | A) \cdot P(A).$$

- **Theorem 1.23 (Bayes' Theorem).** Given propositions A and B where $P(A) \neq 0$ and $P(B) \neq 0$, we have:

$$P(A | B) = \frac{P(B | A) \cdot P(A)}{P(B)}$$

- *Proof:*

$$1. \ P(A | B) = \frac{P(A \wedge B)}{P(B)} = \frac{P(B | A) \cdot P(A)}{P(B)}$$



...okay, that was straightforward... what's the big deal?

Bayes' Rule

- **Note:** By definition, $P(A | B) = \frac{P(A \wedge B)}{P(B)}$. In practice, we often know the conditional probability already, and use it to compute the probability of the conjunction instead:

$$P(A \wedge B) = P(A | B) \cdot P(B) = P(B | A) \cdot P(A).$$

- **Theorem 1.24 (Bayes' Theorem).** Given propositions A and B where $P(A) \neq 0$ and $P(B) \neq 0$, we have:

$$P(A | B) = \frac{P(B | A) \cdot P(A)}{P(B)}$$

- *Proof:*

$$1. \ P(A | B) = \frac{P(A \wedge B)}{P(B)} = \frac{P(B | A) \cdot P(A)}{P(B)}$$

□

...okay, that was straightforward... what's the big deal?

- **(Somewhat Dubious) Claim:** Bayes' Rule is the entire scientific method condensed into a single equation!
- This is an extreme overstatement, but there is a grain of truth in it.

Bayes' Theorem - Why the Hype?

- ▶ Say we have a *hypothesis* H about the world. (e.g. “The universe had a beginning”)
- ▶ We have *some prior belief* $P(H)$.
- ▶ We gather *evidence* E . (e.g. “We observe a cosmic microwave background at 2.7K everywhere”)
- ▶ Bayes' Rule tells us how to *update our belief* in H based on H 's ability to *predict* E (the *likelihood* $P(E | H)$) – and, importantly, the ability of *competing hypotheses* to predict the *same* evidence. (This is actually how scientific hypotheses should be evaluated)

$$\underbrace{P(H | E)}_{\text{posterior}} = \frac{P(E | H) \cdot P(H)}{P(E)} = \frac{\overbrace{P(E | H)}^{\text{likelihood}} \cdot \overbrace{P(H)}^{\text{prior}}}{\underbrace{P(E | H)}_{\text{likelihood}} \underbrace{P(H)}_{\text{prior}} + \underbrace{P(E | \neg H)P(\neg H)}_{\text{competition}}}$$

... if I keep gathering evidence and update, ultimately the impact of the prior belief will diminish.
“You’re entitled to your own priors, but not your own likelihoods”

22.1.2 Independence

Independence

- ▶ **Question:** What is the probability that $S = 7$ and the patient has a toothache?
Or less contrived: What is the probability that the patient has a gingivitis and a cavity?
- ▶ **Definition 1.25.** Two events A and B are called independent, iff $P(A \wedge B) = P(A) \cdot P(B)$.
Two random variables X_1, X_2 are called independent, iff for all $x_1 \in \text{dom}(X_1)$ and $x_2 \in \text{dom}(X_2)$, the events $X_1 = x_1$ and $X_2 = x_2$ are independent. We write $A \perp B$ or $X_1 \perp X_2$, respectively.
- ▶ **Theorem 1.26.** Equivalently: Given events A and B with $P(B) \neq 0$, then A and B are independent iff $P(A \mid B) = P(A)$ (equivalently: $P(B \mid A) = P(B)$).
- ▶ *Proof:*
 - \Rightarrow
By definition, $P(A \mid B) = \frac{P(A \wedge B)}{P(B)} = \frac{P(A) \cdot P(B)}{P(B)} = P(A)$,
 - \Leftarrow
Assume $P(A \mid B) = P(A)$.
Then $P(A \wedge B) = P(A \mid B) \cdot P(B) = P(A) \cdot P(B)$.

□

- ▶ **Note:** Independence asserts that two events are “not related” – the probability of one does not depend on the other.

Mathematically, we can determine independence by checking whether $P(A \wedge B) = P(A) \cdot P(B)$.

In practice, this is impossible to check. Instead, we assume independence based on domain knowledge,

Independence (Examples)

► Example 1.27.

- First = 2 and Second = 3 are **independent** – more generally, First and Second are **independent** (The outcome of the first die does not affect the outcome of the second die)

Quick check: $P((\text{First} = a) \wedge (\text{Second} = b)) = \frac{1}{36} = P(\text{First} = a) \cdot P(\text{Second} = b) \quad \checkmark$

- First and S are **not independent**. (The outcome of the first die affects the sum of the two dice.)

Counterexample: $P((\text{First} = 1) \wedge (S = 4)) = \frac{1}{36} \neq P(\text{First} = 1) \cdot P(S = 4) = \frac{1}{6} \cdot \frac{1}{2} = \frac{1}{12}$

Independence (Examples)

► Example 1.29.

- First = 2 and Second = 3 are **independent** – more generally, First and Second are **independent** (The outcome of the first die does not affect the outcome of the second die)
Quick check: $P((\text{First} = a) \wedge (\text{Second} = b)) = \frac{1}{36} = P(\text{First} = a) \cdot P(\text{Second} = b) \quad \checkmark$
- First and S are **not independent**. (The outcome of the first die affects the sum of the two dice.)
Counterexample: $P((\text{First} = 1) \wedge (S = 4)) = \frac{1}{36} \neq P(\text{First} = 1) \cdot P(S = 4) = \frac{1}{6} \cdot \frac{1}{2} = \frac{1}{12}$
- **But:** $P((\text{First} = a) \wedge (S = 7)) = \frac{1}{36} = \frac{1}{6} \cdot \frac{1}{6} = P(\text{First} = a) \cdot P(S = 7)$ – so the **events** First = a and $S = 7$ are **independent**. (Why?)

Independence (Examples)

► Example 1.31.

- First = 2 and Second = 3 are **independent** – more generally, First and Second are **independent** (The outcome of the first die does not affect the outcome of the second die)

Quick check: $P((\text{First} = a) \wedge (\text{Second} = b)) = \frac{1}{36} = P(\text{First} = a) \cdot P(\text{Second} = b) \quad \checkmark$

- First and S are **not independent**. (The outcome of the first die affects the sum of the two dice.)

Counterexample: $P((\text{First} = 1) \wedge (S = 4)) = \frac{1}{36} \neq P(\text{First} = 1) \cdot P(S = 4) = \frac{1}{6} \cdot \frac{1}{2} = \frac{1}{12}$

- **But:** $P((\text{First} = a) \wedge (S = 7)) = \frac{1}{36} = \frac{1}{6} \cdot \frac{1}{6} = P(\text{First} = a) \cdot P(S = 7)$ – so the **events** First = a and $S = 7$ are **independent**. (Why?)

► Example 1.32.

- Are **cavity** and **toothache** independent?

Independence (Examples)

► Example 1.33.

- First = 2 and Second = 3 are **independent** – more generally, First and Second are **independent** (The outcome of the first die does not affect the outcome of the second die)

Quick check: $P((\text{First} = a) \wedge (\text{Second} = b)) = \frac{1}{36} = P(\text{First} = a) \cdot P(\text{Second} = b)$ ✓

- First and S are **not independent**. (The outcome of the first die affects the sum of the two dice.)

Counterexample: $P((\text{First} = 1) \wedge (S = 4)) = \frac{1}{36} \neq P(\text{First} = 1) \cdot P(S = 4) = \frac{1}{6} \cdot \frac{1}{2} = \frac{1}{12}$

- **But:** $P((\text{First} = a) \wedge (S = 7)) = \frac{1}{36} = \frac{1}{6} \cdot \frac{1}{6} = P(\text{First} = a) \cdot P(S = 7)$ – so the **events** First = a and $S = 7$ are **independent**. (Why?)

► Example 1.34.

- Are **cavity** and **toothache independent**?

...since cavities can cause a toothache, that would probably be a bad design decision ...

Independence (Examples)

► Example 1.35.

- First = 2 and Second = 3 are **independent** – more generally, First and Second are **independent** (The outcome of the first die does not affect the outcome of the second die)

Quick check: $P((\text{First} = a) \wedge (\text{Second} = b)) = \frac{1}{36} = P(\text{First} = a) \cdot P(\text{Second} = b)$ ✓

- First and S are **not independent**. (The outcome of the first die affects the sum of the two dice.)

Counterexample: $P((\text{First} = 1) \wedge (S = 4)) = \frac{1}{36} \neq P(\text{First} = 1) \cdot P(S = 4) = \frac{1}{6} \cdot \frac{1}{2} = \frac{1}{12}$

- **But:** $P((\text{First} = a) \wedge (S = 7)) = \frac{1}{36} = \frac{1}{6} \cdot \frac{1}{6} = P(\text{First} = a) \cdot P(S = 7)$ – so the **events** First = a and $S = 7$ are **independent**. (Why?)

► Example 1.36.

- Are **cavity** and **toothache independent**?

...since cavities can cause a toothache, that would probably be a bad design decision ...

- Are **cavity** and **gingivitis independent**? Cavities do not cause gingivitis, and gingivitis does not cause cavities, so... yes... right? (...as far as I know. I'm not a dentist.)

Independence (Examples)

► Example 1.37.

- First = 2 and Second = 3 are **independent** – more generally, First and Second are **independent** (The outcome of the first die does not affect the outcome of the second die)
Quick check: $P((\text{First} = a) \wedge (\text{Second} = b)) = \frac{1}{36} = P(\text{First} = a) \cdot P(\text{Second} = b) \quad \checkmark$
- First and S are **not independent**. (The outcome of the first die affects the sum of the two dice.)
Counterexample: $P((\text{First} = 1) \wedge (S = 4)) = \frac{1}{36} \neq P(\text{First} = 1) \cdot P(S = 4) = \frac{1}{6} \cdot \frac{1}{2} = \frac{1}{12}$
- **But:** $P((\text{First} = a) \wedge (S = 7)) = \frac{1}{36} = \frac{1}{6} \cdot \frac{1}{6} = P(\text{First} = a) \cdot P(S = 7)$ – so the **events** First = a and S = 7 are **independent**. (Why?)

► Example 1.38.

- Are **cavity** and **toothache independent**?
...since cavities can cause a toothache, that would probably be a bad design decision ...
- Are **cavity** and **gingivitis independent**? Cavities do not cause gingivitis, and gingivitis does not cause cavities, so... yes... right? (...as far as I know. I'm not a dentist.)
- **Probably not!** A patient who has cavities has probably worse dental hygiene than those who don't, and is thus more likely to have gingivitis as well.
- \leadsto **cavity** may be *evidence* that raises the probability of **gingivitis**, even if they are not directly causally related.

Conditional Independence – Motivation

- ▶ A dentist can diagnose a cavity by using a *probe*, which may (or may not) *catch* in a cavity.
- ▶ Say we know from clinical studies that $P(\text{cavity}) = 0.2$, $P(\text{toothache} \mid \text{cavity}) = 0.6$, $P(\text{toothache} \mid \neg\text{cavity}) = 0.1$, $P(\text{catch} \mid \text{cavity}) = 0.9$, and $P(\text{catch} \mid \neg\text{cavity}) = 0.2$.
- ▶ Assume the patient complains about a toothache, and our probe indeed catches in the aching tooth. What is the likelihood of having a cavity $P(\text{cavity} \mid (\text{toothache} \wedge \text{catch}))$?

Conditional Independence – Motivation

- ▶ A dentist can diagnose a cavity by using a *probe*, which may (or may not) *catch* in a cavity.
- ▶ Say we know from clinical studies that $P(\text{cavity}) = 0.2$, $P(\text{toothache} \mid \text{cavity}) = 0.6$, $P(\text{toothache} \mid \neg \text{cavity}) = 0.1$, $P(\text{catch} \mid \text{cavity}) = 0.9$, and $P(\text{catch} \mid \neg \text{cavity}) = 0.2$.
- ▶ Assume the patient complains about a toothache, and our probe indeed catches in the aching tooth. What is the likelihood of having a cavity $P(\text{cavity} \mid (\text{toothache} \wedge \text{catch}))$?
- ▶ **Idea:** Use Bayes' rule:

$$P(\text{cavity} \mid (\text{toothache} \wedge \text{catch})) = \frac{P(\text{toothache} \wedge \text{catch} \mid \text{cavity}) \cdot P(\text{cavity})}{P(\text{toothache} \wedge \text{catch})}$$

- ▶ **Note:** $P(\text{toothache} \wedge \text{catch}) = P(\text{toothache} \wedge \text{catch} \mid \text{cavity}) \cdot P(\text{cavity}) + P(\text{toothache} \wedge \text{catch} \mid \neg \text{cavity}) \cdot P(\neg \text{cavity})$
- ▶ **Problem:** Now we're only missing $P(\text{toothache} \wedge \text{catch} \mid (\text{cavity} = b))$ for $b \in \{T, F\}$ Now what?

Conditional Independence – Motivation

- ▶ A dentist can diagnose a cavity by using a *probe*, which may (or may not) *catch* in a cavity.
- ▶ Say we know from clinical studies that $P(\text{cavity}) = 0.2$, $P(\text{toothache} \mid \text{cavity}) = 0.6$, $P(\text{toothache} \mid \neg\text{cavity}) = 0.1$, $P(\text{catch} \mid \text{cavity}) = 0.9$, and $P(\text{catch} \mid \neg\text{cavity}) = 0.2$.
- ▶ Assume the patient complains about a toothache, and our probe indeed catches in the aching tooth. What is the likelihood of having a cavity $P(\text{cavity} \mid (\text{toothache} \wedge \text{catch}))$?
- ▶ **Idea:** Use Bayes' rule:

$$P(\text{cavity} \mid (\text{toothache} \wedge \text{catch})) = \frac{P(\text{toothache} \wedge \text{catch} \mid \text{cavity}) \cdot P(\text{cavity})}{P(\text{toothache} \wedge \text{catch})}$$

- ▶ **Note:** $P(\text{toothache} \wedge \text{catch}) = P(\text{toothache} \wedge \text{catch} \mid \text{cavity}) \cdot P(\text{cavity}) + P(\text{toothache} \wedge \text{catch} \mid \neg\text{cavity}) \cdot P(\neg\text{cavity})$
- ▶ **Problem:** Now we're only missing $P(\text{toothache} \wedge \text{catch} \mid (\text{cavity} = b))$ for $b \in \{T, F\}$ Now what?
- ▶ Are *toothache* and *catch* independent, maybe?

Conditional Independence – Motivation

- ▶ A dentist can diagnose a cavity by using a *probe*, which may (or may not) *catch* in a cavity.
- ▶ Say we know from clinical studies that $P(\text{cavity}) = 0.2$, $P(\text{toothache} \mid \text{cavity}) = 0.6$, $P(\text{toothache} \mid \neg\text{cavity}) = 0.1$, $P(\text{catch} \mid \text{cavity}) = 0.9$, and $P(\text{catch} \mid \neg\text{cavity}) = 0.2$.
- ▶ Assume the patient complains about a toothache, and our probe indeed catches in the aching tooth. What is the likelihood of having a cavity $P(\text{cavity} \mid (\text{toothache} \wedge \text{catch}))$?
- ▶ **Idea:** Use Bayes' rule:

$$P(\text{cavity} \mid (\text{toothache} \wedge \text{catch})) = \frac{P(\text{toothache} \wedge \text{catch} \mid \text{cavity}) \cdot P(\text{cavity})}{P(\text{toothache} \wedge \text{catch})}$$

- ▶ **Note:** $P(\text{toothache} \wedge \text{catch}) = P(\text{toothache} \wedge \text{catch} \mid \text{cavity}) \cdot P(\text{cavity}) + P(\text{toothache} \wedge \text{catch} \mid \neg\text{cavity}) \cdot P(\neg\text{cavity})$
- ▶ **Problem:** Now we're only missing $P(\text{toothache} \wedge \text{catch} \mid (\text{cavity} = b))$ for $b \in \{\text{T}, \text{F}\}$ Now what?
- ▶ Are *toothache* and *catch* independent, maybe? **No:** Both have a common (possible) cause, *cavity*. Also, there's this pesky $P(\cdot \mid \text{cavity})$ in the way.wait a minute...

Conditional Independence – Definition

- ▶ *Assuming* the patient has (or does not have) a cavity, the events *toothache* and *catch* are *independent*: Both are caused by a cavity, but they don't influence each other otherwise.
i.e. *cavity* “contains all the information” that links *toothache* and *catch* in the first place.

Conditional Independence – Definition

- ▶ Assuming the patient has (or does not have) a cavity, the events *toothache* and *catch* are independent: Both are caused by a cavity, but they don't influence each other otherwise. i.e. *cavity* “contains all the information” that links *toothache* and *catch* in the first place.
- ▶ **Definition 1.41.** Given events A, B, C with $P(C) \neq 0$, then A and B are called **conditionally independent given C** , iff $P(A \wedge B \mid C) = P(A \mid C) \cdot P(B \mid C)$.
Equivalently: iff $P(A \mid (B \wedge C)) = P(A \mid C)$, or $P(B \mid (A \wedge C)) = P(B \mid C)$.

Let Y be a random variable. We call two random variables X_1, X_2 **conditionally independent given Y** , iff for all $x_1 \in \text{dom}(X_1)$, $x_2 \in \text{dom}(X_2)$ and $y \in \text{dom}(Y)$, the events $X_1 = x_1$ and $X_2 = x_2$ are conditionally independent given $Y = y$.

Conditional Independence – Definition

- Assuming the patient has (or does not have) a cavity, the events **toothache** and **catch** are **independent**: Both are caused by a cavity, but they don't influence each other otherwise. i.e. **cavity** “contains all the information” that links **toothache** and **catch** in the first place.

- Definition 1.43.** Given events A, B, C with $P(C) \neq 0$, then A and B are called **conditionally independent given C** , iff $P(A \wedge B \mid C) = P(A \mid C) \cdot P(B \mid C)$.
Equivalently: iff $P(A \mid (B \wedge C)) = P(A \mid C)$, or $P(B \mid (A \wedge C)) = P(B \mid C)$.

Let Y be a random variable. We call two random variables X_1, X_2 **conditionally independent given Y** , iff for all $x_1 \in \text{dom}(X_1)$, $x_2 \in \text{dom}(X_2)$ and $y \in \text{dom}(Y)$, the events $X_1 = x_1$ and $X_2 = x_2$ are **conditionally independent given $Y = y$** .

- Example 1.44.** Let's assume **toothache** and **catch** are **conditionally independent given cavity/ \neg cavity**. Then we can finally compute:

$$\begin{aligned} P(\text{cavity} \mid (\text{toothache} \wedge \text{catch})) &= \frac{P(\text{toothache} \wedge \text{catch} \mid \text{cavity}) \cdot P(\text{cavity})}{P(\text{toothache} \wedge \text{catch})} \\ &= \frac{P(\text{toothache} \mid \text{cavity}) \cdot P(\text{catch} \mid \text{cavity}) \cdot P(\text{cavity})}{P(\text{toothache} \mid \text{cavity}) \cdot P(\text{catch} \mid \text{cavity}) \cdot P(\text{cavity}) + P(\text{toothache} \mid \neg \text{cavity}) \cdot P(\text{catch} \mid \neg \text{cavity}) \cdot P(\neg \text{cavity})} \\ &= \frac{0.6 \cdot 0.9 \cdot 0.2}{0.6 \cdot 0.9 \cdot 0.2 + 0.1 \cdot 0.2 \cdot 0.8} = 0.87 \end{aligned}$$

Conditional Independence

- **Lemma 1.45.** If A and B are *conditionally independent* given C , then $P(A \mid (B \wedge C)) = P(A \mid C)$

Proof:

$$P(A \mid (B \wedge C)) = \frac{P(A \wedge B \wedge C)}{P(B \wedge C)} = \frac{P(A \wedge B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \wedge C)}{P(B \wedge C)} = P(A \mid C)$$

□

- **Question:** If A and B are *conditionally independent* given C , does this imply that A and B are independent?

Conditional Independence

- **Lemma 1.46.** If A and B are *conditionally independent* given C , then $P(A \mid (B \wedge C)) = P(A \mid C)$

Proof:

$$P(A \mid (B \wedge C)) = \frac{P(A \wedge B \wedge C)}{P(B \wedge C)} = \frac{P(A \wedge B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \wedge C)}{P(B \wedge C)} = P(A \mid C)$$

□

- **Question:** If A and B are *conditionally independent* given C , does this imply that A and B are *independent*? **No.** See previous slides for a counterexample.
- **Question:** If A and B are *independent*, does this imply that A and B are also *conditionally independent* given C ?

Conditional Independence

- **Lemma 1.47.** If A and B are *conditionally independent* given C , then $P(A \mid (B \wedge C)) = P(A \mid C)$

Proof:

$$P(A \mid (B \wedge C)) = \frac{P(A \wedge B \wedge C)}{P(B \wedge C)} = \frac{P(A \wedge B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \wedge C)}{P(B \wedge C)} = P(A \mid C)$$

□

- **Question:** If A and B are *conditionally independent* given C , does this imply that A and B are *independent*? **No.** See previous slides for a counterexample.
- **Question:** If A and B are *independent*, does this imply that A and B are also *conditionally independent* given C ? **No.** For example: *First* and *Second* are *independent*, but not *conditionally independent* given $S = 4$.

Conditional Independence

- **Lemma 1.48.** If A and B are *conditionally independent* given C , then $P(A \mid (B \wedge C)) = P(A \mid C)$

Proof:

$$P(A \mid (B \wedge C)) = \frac{P(A \wedge B \wedge C)}{P(B \wedge C)} = \frac{P(A \wedge B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \wedge C)}{P(B \wedge C)} = P(A \mid C)$$

□

- **Question:** If A and B are *conditionally independent* given C , does this imply that A and B are *independent*? **No.** See previous slides for a counterexample.
- **Question:** If A and B are *independent*, does this imply that A and B are also *conditionally independent* given C ? **No.** For example: *First* and *Second* are *independent*, but not *conditionally independent* given $S = 4$.
- **Question:** Okay, so what if A , B and C are *all* pairwise *independent*? Are A and B *conditionally independent* given C *now*?

Conditional Independence

- **Lemma 1.49.** If A and B are *conditionally independent* given C , then $P(A \mid (B \wedge C)) = P(A \mid C)$

Proof:

$$P(A \mid (B \wedge C)) = \frac{P(A \wedge B \wedge C)}{P(B \wedge C)} = \frac{P(A \wedge B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \wedge C)}{P(B \wedge C)} = P(A \mid C)$$

□

- **Question:** If A and B are *conditionally independent* given C , does this imply that A and B are *independent*? **No.** See previous slides for a counterexample.
- **Question:** If A and B are *independent*, does this imply that A and B are also *conditionally independent* given C ? **No.** For example: *First* and *Second* are *independent*, but not *conditionally independent* given $S = 4$.
- **Question:** Okay, so what if A , B and C are *all* pairwise *independent*? Are A and B *conditionally independent* given C *now*? **Still no.** Remember: *First* = a , *Second* = b and $S = 7$ are all independent, but *First* and *Second* are not *conditionally independent* given $S = 7$.
- **Question:** When can we infer *conditional independence* from a “more general” notion of independence?

Conditional Independence

- **Lemma 1.50.** If A and B are *conditionally independent* given C , then $P(A \mid (B \wedge C)) = P(A \mid C)$

Proof:

$$P(A \mid (B \wedge C)) = \frac{P(A \wedge B \wedge C)}{P(B \wedge C)} = \frac{P(A \wedge B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \mid C) \cdot P(C)}{P(B \wedge C)} = \frac{P(A \mid C) \cdot P(B \wedge C)}{P(B \wedge C)} = P(A \mid C)$$

□

- **Question:** If A and B are *conditionally independent* given C , does this imply that A and B are *independent*? **No.** See previous slides for a counterexample.
- **Question:** If A and B are *independent*, does this imply that A and B are also *conditionally independent* given C ? **No.** For example: *First* and *Second* are *independent*, but not *conditionally independent* given $S = 4$.
- **Question:** Okay, so what if A , B and C are *all* pairwise *independent*? Are A and B *conditionally independent* given C *now*? **Still no.** Remember: *First* = a , *Second* = b and $S = 7$ are all independent, but *First* and *Second* are not *conditionally independent* given $S = 7$.
- **Question:** When can we infer *conditional independence* from a “more general” notion of independence?
- We need *mutual independence*. Roughly: A set of *events* is called *mutually independent*, if every *event* is *independent* from any *conjunction* of the others. (Not really relevant for this course though)

22.1.3 Conclusion

Summary

- ▶ Probability spaces serve as a mathematical model (and hence justification) for everything related to probabilities.
- ▶ The “atoms” of any statement of probability are the random variables. (Important special cases: Boolean and finite domain)
- ▶ We can define probabilities on compound (propositional logical) statements, with (outcomes of) random variables as “propositional variables”.
- ▶ Conditional probabilities represent *posterior probabilities* given some observed outcomes.
- ▶ Independence and conditional independence are strong assumptions that allow us to simplify computations of probabilities
- ▶ Bayes' Theorem (can be used between “causal” and “diagnostic” conditional probabilities)

So much about the math...

- ▶ We now have a mathematical setup for **probabilities**.
- ▶ **But:** The math does not tell us what probabilities *are*:
- ▶ Assume we can mathematically derive this to be the case: *the probability of rain tomorrow is 0.3*. What does this even *mean*?

So much about the math...

- ▶ We now have a mathematical setup for **probabilities**.
- ▶ **But:** The math does not tell us what probabilities *are*:
- ▶ Assume we can mathematically derive this to be the case: *the probability of rain tomorrow is 0.3*. What does this even *mean*?
- ▶ **Frequentist Answer:** The **probability** of an **event** is the limit of its relative frequency in a large number of trials.
In other words: “In 30% of the cases where we have similar weather conditions, it rained the next day.”

So much about the math...

- ▶ We now have a mathematical setup for **probabilities**.
- ▶ **But:** The math does not tell us what probabilities *are*:
- ▶ Assume we can mathematically derive this to be the case: *the probability of rain tomorrow is 0.3*. What does this even *mean*?
- ▶ **Frequentist Answer:** The **probability** of an **event** is the limit of its relative frequency in a large number of trials.
In other words: “In 30% of the cases where we have similar weather conditions, it rained the next day.”
- ▶ **Objection:** Okay, but what about *unique* events? “The probability of me passing the **exam** is 80%” – does this mean anything, if I only take the **exam** once? Am I comparable to “similar students”? What counts as sufficiently “similar”?

So much about the math...

- ▶ We now have a mathematical setup for **probabilities**.
- ▶ **But:** The math does not tell us what probabilities *are*:
- ▶ Assume we can mathematically derive this to be the case: *the probability of rain tomorrow is 0.3*. What does this even *mean*?
- ▶ **Frequentist Answer:** The **probability** of an **event** is the limit of its relative frequency in a large number of trials.
In other words: “In 30% of the cases where we have similar weather conditions, it rained the next day.”
- ▶ **Objection:** Okay, but what about *unique* events? “The probability of me passing the **exam** is 80%” – does this mean anything, if I only take the **exam** once? Am I comparable to “similar students”? What counts as sufficiently “similar”?
- ▶ **Bayesian Answer:** **Probabilities** are *degrees of belief*. It means you **should** be 30% confident that it will rain tomorrow.

So much about the math...

- ▶ We now have a mathematical setup for **probabilities**.
- ▶ **But:** The math does not tell us what probabilities *are*:
- ▶ Assume we can mathematically derive this to be the case: *the probability of rain tomorrow is 0.3*. What does this even *mean*?
- ▶ **Frequentist Answer:** The **probability** of an **event** is the limit of its relative frequency in a large number of trials.
In other words: “In 30% of the cases where we have similar weather conditions, it rained the next day.”
- ▶ **Objection:** Okay, but what about *unique* events? “The probability of me passing the **exam** is 80%” – does this mean anything, if I only take the **exam** once? Am I comparable to “similar students”? What counts as sufficiently “similar”?
- ▶ **Bayesian Answer:** **Probabilities** are *degrees of belief*. It means you **should** be 30% confident that it will rain tomorrow.
- ▶ **Objection:** And why *should* I? Is this not purely *subjective* then?

- ▶ **Pragmatically** both interpretations amount to the same thing: I should *act as if* I'm 30% confident that it will rain tomorrow. (Whether by fiat, or because in 30% of comparable cases, it rained.)
- ▶ **Objection:** Still: why should I? And why should my beliefs follow the seemingly arbitrary Kolmogorov axioms?

- ▶ **Pragmatically** both interpretations amount to the same thing: I should *act as if* I'm 30% confident that it will rain tomorrow. (Whether by fiat, or because in 30% of comparable cases, it rained.)
- ▶ **Objection:** Still: why should I? And why should my beliefs follow the seemingly arbitrary Kolmogorov axioms?
- ▶ **[deFinetti:sssdp31]:** If an agent has a belief that violates the Kolmogorov axioms, then there exists a combination of “bets” on propositions so that the agent *always* loses money.
- ▶ **In other words:** If your beliefs are not consistent with the mathematics, and you *act in accordance with your beliefs*, there is a way to exploit this inconsistency to your disadvantage.

- ▶ **Pragmatically** both interpretations amount to the same thing: I should *act as if* I'm 30% confident that it will rain tomorrow. (Whether by fiat, or because in 30% of comparable cases, it rained.)
- ▶ **Objection:** Still: why should I? And why should my beliefs follow the seemingly arbitrary Kolmogorov axioms?
- ▶ **[deFinetti:ssdp31]:** If an agent has a belief that violates the Kolmogorov axioms, then there exists a combination of "bets" on propositions so that the agent *always* loses money.
- ▶ **In other words:** If your beliefs are not consistent with the mathematics, and you *act in accordance with your beliefs*, there is a way to exploit this inconsistency to your disadvantage.
- ▶ ...and, more importantly, the AI agents you design! 😊
- ▶ **I (and my agents) do not bet:** That is not true, in a partially observable or non-deterministic world, every action choice is a necessarily bet: The outcome is not sure.

22.2 Probabilistic Reasoning Techniques

Okay, now how do I implement this?

- ▶ This is a **CS course**. We need to implement this stuff.
- ▶ Do we... implement **random variables** as functions? Is a **probability space** a... class maybe?

Okay, now how do I implement this?

- ▶ This is a **CS course**. We need to implement this stuff.
- ▶ Do we... implement **random variables** as functions? Is a **probability space** a... class maybe?
- ▶ **No:** As mentioned, we rarely know the **probability space** entirely. Instead we will use **probability distributions**, which are just **arrays** (of **arrays** of...) of **probabilities**.
- ▶ And then we represent *those* as sparsely as possible, by exploiting **independence**, **conditional independence**, ...

22.2.1 Probability Distributions

Probability Distributions

- ▶ **Definition 2.1.** The **probability distribution** for a **random variable** X , written $\mathbb{P}(X)$, is the **vector of probabilities** for the (ordered) **domain** of X .
- ▶ **Note:** The values in a **probability distribution** are all positive and sum to 1. (Why?)
- ▶ **Example 2.2.** $\mathbb{P}(\text{First}) = \mathbb{P}(\text{Second}) = \langle \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6}, \frac{1}{6} \rangle$. (Both First and Second are **uniformly distributed**)
- ▶ **Example 2.3.** The **probability distribution** $\mathbb{P}(S)$ is $\langle \frac{1}{36}, \frac{1}{18}, \frac{1}{12}, \frac{1}{9}, \frac{5}{36}, \frac{1}{6}, \frac{5}{36}, \frac{1}{9}, \frac{1}{12}, \frac{1}{18}, \frac{1}{36} \rangle$. Note the symmetry, with a “peak” at 7 – the **random variable** is (*approximately*, because our domain is discrete rather than continuous) **normally distributed** (or **gaussian distributed**, or **follows a bell-curve**,...).
- ▶ **Example 2.4.** **Probability distributions** for **Boolean random variables** are naturally *pairs* (probabilities for T and F), e.g.:

$$\mathbb{P}(\text{toothache}) = \langle 0.15, 0.85 \rangle$$

$$\mathbb{P}(\text{cavity}) = \langle 0.122, 0.878 \rangle$$

- ▶ More generally:
- ▶ **Definition 2.5.** A **probability distribution** is a **vector** v of values $v_i \in [0,1]$ such that $\sum_i v_i = 1$.

The Full Joint Probability Distribution

► **Definition 2.6.** Given random variables X_1, \dots, X_n with domains D_1, \dots, D_n , the **full joint probability distribution**, denoted $\mathbb{P}(X_1, \dots, X_n)$, is the n -dimensional array of size $|D_1 \times \dots \times D_n|$ that lists the probabilities of all conjunctions of values of the random variables.

► **Example 2.7.** $\mathbb{P}(\text{cavity}, \text{toothache}, \text{gingivitis})$ could look something like this:

	toothache		\neg toothache	
	gingivitis	\neg gingivitis	gingivitis	\neg gingivitis
cavity	0.007	0.06	0.005	0.05
\neg cavity	0.08	0.003	0.045	0.75

► **Example 2.8.** $\mathbb{P}(\text{First}, S)$

First \ S	2	3	4	5	6	7	8	9	10	11	12
1	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	0	0	0	0	0
2	0	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	0	0	0	0
3	0	0	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	0	0	0
4	0	0	0	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	0	0
5	0	0	0	0	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	0
6	0	0	0	0	0	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$	$\frac{1}{36}$

Note that if we know the value of **First**, the value of **S** is completely determined by the value of **Second**.

Conditional Probability Distributions

- **Definition 2.9.** Given random variables X and Y , the conditional probability distribution of X given Y , written $\mathbb{P}(X|Y)$ is the table of all conditional probabilities of values of X given values of Y .
- For sets of variables analogously: $\mathbb{P}(X_1, \dots, X_n | Y_1, \dots, Y_m)$.
- **Example 2.10.** $\mathbb{P}(\text{cavity} | \text{toothache})$:

	toothache	\neg toothache
cavity	$P(\text{cavity} \text{toothache}) = 0.45$	$P(\text{cavity} \neg\text{toothache}) = 0.065$
\neg cavity	$P(\neg\text{cavity} \text{toothache}) = 0.55$	$P(\neg\text{cavity} \neg\text{toothache}) = 0.935$

- **Example 2.11.** $\mathbb{P}(\text{First} | S)$

First \ S	2	3	4	5	6	7	8	9	10	11	12
1	1	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{6}$	0	0	0	0	0
2	0	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{5}$	0	0	0	0
3	0	0	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{5}$	$\frac{1}{4}$	0	0	0
4	0	0	0	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{5}$	$\frac{1}{4}$	$\frac{1}{3}$	0	0
5	0	0	0	0	$\frac{1}{5}$	$\frac{1}{6}$	$\frac{1}{5}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	0
6	0	0	0	0	0	$\frac{1}{6}$	$\frac{1}{5}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	1

- **Note:** Every “column” of a conditional probability distribution is itself a probability distribution. (Why?)

- ▶ We now “lift” multiplication and division to the level of whole **probability distributions**:
- ▶ **Definition 2.12.** Whenever we use \mathbb{P} in an equation, we take this to mean a *system of equations*, for each value in the **domains** of the **random variables** involved.

Example 2.13.

- ▶ $\mathbb{P}(X, Y) = \mathbb{P}(X|Y) \cdot \mathbb{P}(Y)$ represents the system of equations
 $P(X = x \wedge Y = y) = P(X = x | Y = y) \cdot P(Y = y)$ for all x, y in the respective domains.
- ▶ $\mathbb{P}(X|Y) := \frac{\mathbb{P}(X, Y)}{\mathbb{P}(Y)}$ represents the system of equations $P(X = x | (Y = y)) := \frac{P((X=x) \wedge (Y=y))}{P(Y=y)}$
- ▶ Bayes' Theorem: $\mathbb{P}(X|Y) = \frac{\mathbb{P}(Y|X) \cdot \mathbb{P}(X)}{\mathbb{P}(Y)}$ represents the system of equations
 $P(X = x | (Y = y)) = \frac{P(Y=y | (X=x)) \cdot P(X=x)}{P(Y=y)}$

So, what's the point?

- ▶ Obviously, the **probability distribution** contains all the information about a specific **random variable** we need.
- ▶ **Observation:** The **full joint probability distribution** of variables X_1, \dots, X_n contains *all* the information about the **random variables** *and their conjunctions* we need.
- ▶ **Example 2.14.** We can read off the **probability** $P(\text{toothache})$ from the **full joint probability distribution** as $0.007 + 0.06 + 0.08 + 0.003 = 0.15$, and the **probability** $P(\text{toothache} \wedge \text{cavity})$ as $0.007 + 0.06 = 0.067$
- ▶ We can actually implement this! (They're just (nested) arrays)
- ▶ **But:** just as we often don't have a fully specified **probability space** to work in, we often don't have a **full joint probability distribution** for our **random variables** either.
- ▶ **Also:** Given **random variables** X_1, \dots, X_n , the **full joint probability distribution** has $\prod_{i=1}^n |\text{dom}(X_i)|$ entries! ($\mathbb{P}(\text{First}, S)$ already has 60 entries!)
- ▶ **So:** The rest of this section deals with keeping things small, by *computing probabilities* instead of *storing* them all.

- ▶ **Probabilistic reasoning** refers to inferring **probabilities** of **events** from the **probabilities** of other **events**
as opposed to determining the **probabilities** e.g. *empirically*, by gathering (sufficient amounts of *representative*) data and counting.
- ▶ **Note:** In practice, we are *primarily* interested in, and have access to, **conditional probabilities** rather than the **unconditional probabilities** of **conjunctions** of **events**:
 - ▶ We don't reason in a vacuum: Usually, we have some **evidence** and want to infer the posterior **probability** of some related **event**.
(e.g. *infer a plausible cause given some symptom*)
 \leadsto we are interested in the **conditional probability** $P(\text{hypothesis} \mid \text{observation})$.
 - ▶ “80% of patients with a cavity complain about a toothache” (i.e. $P(\text{toothache} \mid \text{cavity})$) is more the kind of data people actually collect and publish than “1.2% of the general population have both a cavity and a toothache” (i.e. $P(\text{cavity} \wedge \text{toothache})$).
 - ▶ Consider the probe catching in a cavity. The probe is a diagnostic tool, which is usually evaluated in terms of its *sensitivity* $P(\text{catch} \mid \text{cavity})$ and *specificity* $P(\neg \text{catch} \mid \neg \text{cavity})$. (You have probably heard these words a lot since 2020...)

22.2.2 Naive Bayes

Naive Bayes Models

- Consider again the dentistry example with random variables *cavity*, *toothache*, and *catch*. We assume *cavity* **causes** both *toothache* and *catch*, and that *toothache* and *catch* are **conditionally independent** given *cavity*:



- We likely know the *sensitivity* $P(\text{catch} \mid \text{cavity})$ and *specificity* $P(\neg \text{catch} \mid \neg \text{cavity})$, which jointly give us $\mathbb{P}(\text{catch} \mid \text{cavity})$, and from medical studies, we should be able to determine $P(\text{cavity})$ (the *prevalence* of cavities in the population) and $\mathbb{P}(\text{toothache} \mid \text{cavity})$.
- This kind of situation is surprisingly common, and therefore deserves a name.



► **Definition 2.15.** A **naive Bayes model** (or, less accurately, **Bayesian classifier**, or, derogatorily, **idiot Bayes model**) consists of:

1. random variables C, E_1, \dots, E_n such that all the E_1, \dots, E_n are **conditionally independent** given C ,
2. the probability distribution $\mathbb{P}(C)$, and
3. the conditional probability distributions $\mathbb{P}(E_i|C)$.

We call C the **cause** and the E_1, \dots, E_n the **effects** of the model.

- **Convention:** Whenever we draw a graph of **random variables**, we take the arrows to connect *causes* to their direct *effects*, and assert that unconnected nodes are **conditionally independent** given all their ancestors. We will make this more precise later.
- Can we compute the **full joint probability distribution** $\mathbb{P}(\text{cavity}, \text{toothache}, \text{catch})$ from this information?

Recovering the Full Joint Probability Distribution

- ▶ **Lemma 2.16 (Product rule).** $\mathbb{P}(X, Y) = \mathbb{P}(X|Y) \cdot \mathbb{P}(Y)$.
- ▶ We can generalize this to more than two variables, by repeatedly applying the **product rule**:
- ▶ **Lemma 2.17 (Chain rule).** For any sequence of **random variables** X_1, \dots, X_n :

$$\mathbb{P}(X_1, \dots, X_n) = \mathbb{P}(X_1|X_2, \dots, X_n) \cdot \mathbb{P}(X_2|X_3, \dots, X_n) \cdot \dots \cdot \mathbb{P}(X_{n-1}|X_n) \cdot \mathbb{P}(X_n)$$

Hence:

- ▶ **Theorem 2.18.** Given a **naive Bayes model** with **effects** E_1, \dots, E_n and **cause** C , we have

$$\mathbb{P}(C, E_1, \dots, E_n) = \mathbb{P}(C) \cdot \left(\prod_{i=1}^n \mathbb{P}(E_i|C) \right).$$

- ▶ *Proof:* Using the chain rule:

1. $\mathbb{P}(E_1, \dots, E_n, C) = \mathbb{P}(E_1|E_2, \dots, E_n, C) \cdot \dots \cdot \mathbb{P}(E_n|C) \cdot \mathbb{P}(C)$
2. Since all the E_i are **conditionally independent**, we can drop them on the right hand sides of the $\mathbb{P}(E_j|\dots, C)$



Marginalization

- ▶ Great, so now we can compute $\mathbb{P}(C|E_1, \dots, E_n) = \frac{\mathbb{P}(C, E_1, \dots, E_n)}{\mathbb{P}(E_1, \dots, E_n)} \dots$
...except that we don't know $\mathbb{P}(E_1, \dots, E_n) :-/$
...except that we can compute the **full joint probability distribution**, so we can recover it:
- ▶ **Lemma 2.19 (Marginalization)**. Given *random variables* X_1, \dots, X_n and Y_1, \dots, Y_m , we have $\mathbb{P}(X_1, \dots, X_n) = \sum_{y_1 \in \text{dom}(Y_1), \dots, y_m \in \text{dom}(Y_m)} \mathbb{P}(X_1, \dots, X_n, Y_1 = y_1, \dots, Y_m = y_m)$.
(This is just a fancy way of saying “we can add the relevant entries of the full joint probability distribution”)
- ▶ **Example 2.20**. Say we observed **toothache** = T and **catch** = T. Using **marginalization**, we can compute

$$\begin{aligned} P(\text{cavity} \mid (\text{toothache} \wedge \text{catch})) &= \frac{P(\text{cavity} \wedge \text{toothache} \wedge \text{catch})}{P(\text{toothache} \wedge \text{catch})} \\ &= \frac{P(\text{cavity} \wedge \text{toothache} \wedge \text{catch})}{\sum_{c \in \{\text{cavity}, \neg \text{cavity}\}} P(c \wedge \text{toothache} \wedge \text{catch})} \\ &= \frac{P(\text{cavity}) \cdot P(\text{toothache} \mid \text{cavity}) \cdot P(\text{catch} \mid \text{cavity})}{\sum_{c \in \{\text{cavity}, \neg \text{cavity}\}} P(c) \cdot P(\text{toothache} \mid c) \cdot P(\text{catch} \mid c)} \end{aligned}$$

Unknowns

- ▶ What if we don't know *catch*? (I'm not a dentist, I don't have a probe...)
- ▶ We split our *effects* into $\{E_1, \dots, E_n\} = \{O_1, \dots, O_{n_o}\} \cup \{U_1, \dots, U_{n_u}\}$ – the *observed* and *unknown* random variables.
- ▶ Let $D_U := \text{dom}(U_1) \times \dots \times \text{dom}(U_{n_u})$. Then

$$\begin{aligned}\mathbb{P}(C | O_1, \dots, O_{n_o}) &= \frac{\mathbb{P}(C, O_1, \dots, O_{n_o})}{\mathbb{P}(O_1, \dots, O_{n_o})} \\&= \frac{\sum_{u \in D_U} \mathbb{P}(C, O_1, \dots, O_{n_o}, U_1 = u_1, \dots, U_{n_u} = u_{n_u})}{\sum_{c \in \text{dom}(C)} \sum_{u \in D_U} \mathbb{P}(O_1, \dots, O_{n_o}, C = c, U_1 = u_1, \dots, U_{n_u} = u_{n_u})} \\&= \frac{\sum_{u \in D_U} \mathbb{P}(C) \cdot (\prod_{i=1}^{n_o} \mathbb{P}(O_i | C)) \cdot (\prod_{j=1}^{n_u} \mathbb{P}(U_j = u_j | C))}{\sum_{c \in \text{dom}(C)} \sum_{u \in D_U} \mathbb{P}(C = c) \cdot (\prod_{i=1}^{n_o} \mathbb{P}(O_i | C = c)) \cdot (\prod_{j=1}^{n_u} \mathbb{P}(U_j = u_j | (C = c)))} \\&= \frac{\mathbb{P}(C) \cdot (\prod_{i=1}^{n_o} \mathbb{P}(O_i | C)) \cdot (\sum_{u \in D_U} \prod_{j=1}^{n_u} \mathbb{P}(U_j = u_j | C))}{\sum_{c \in \text{dom}(C)} \mathbb{P}(C = c) \cdot (\prod_{i=1}^{n_o} \mathbb{P}(O_i | C = c)) \cdot (\sum_{u \in D_U} \prod_{j=1}^{n_u} \mathbb{P}(U_j = u_j | (C = c)))}\end{aligned}$$

...oof...

- ▶ Continuing from above:

$$\mathbb{P}(C|O_1, \dots, O_{n_o}) = \frac{\mathbb{P}(C) \cdot (\prod_{i=1}^{n_o} \mathbb{P}(O_i|C)) \cdot (\sum_{u \in D_U} \prod_{j=1}^{n_U} \mathbb{P}(U_j = u_j|C))}{\sum_{c \in \text{dom}(C)} \mathbb{P}(C = c) \cdot (\prod_{i=1}^{n_o} \mathbb{P}(O_i|C = c)) \cdot (\sum_{u \in D_U} \prod_{j=1}^{n_U} \mathbb{P}(U_j = u_j | (C = c)))}$$

- ▶ First, note that $\sum_{u \in D_U} \prod_{j=1}^{n_U} \mathbb{P}(U_j = u_j | (C = c)) = 1$ (We're summing over all possible events on the (conditionally independent) U_1, \dots, U_{n_U} given $C = c$)



$$\mathbb{P}(C|O_1, \dots, O_{n_o}) = \frac{\mathbb{P}(C) \cdot (\prod_{i=1}^{n_o} \mathbb{P}(O_i|C))}{\sum_{c \in \text{dom}(C)} \mathbb{P}(C = c) \cdot (\prod_{i=1}^{n_o} \mathbb{P}(O_i|C = c))}$$

- ▶ Secondly, note that the *denominator* is

1. the same for any given observations O_1, \dots, O_{n_o} , independent of the value of C , and
2. the *sum* over all the *numerators* in the full distribution.

That is: The denominator only serves to *scale* what is *almost* already the distribution $\mathbb{P}(C|O_1, \dots, O_{n_o})$ to sum up to 1.

- **Definition 2.21 (Normalization).** Given a vector $w := \langle w_1, \dots, w_k \rangle$ of numbers in $[0,1]$ where $\sum_{i=1}^k w_i \leq 1$.

Then the **normalized vector** $\alpha(w)$ is defined (component-wise) as

$$(\alpha(w))_i := \frac{w_i}{\sum_{j=1}^k w_j}.$$

Note that $\sum_{i=1}^k \alpha(w)_i = 1$, i.e. $\alpha(w)$ is a **probability distribution**.

- This finally gives us:

Theorem 2.22 (Inference in a Naive Bayes model). Let C, E_1, \dots, E_n a *naive Bayes model* and $E_1, \dots, E_n = O_1, \dots, O_{n_O}, U_1, \dots, U_{n_U}$.

Then

$$\mathbb{P}(C | O_1 = o_1, \dots, O_{n_O} = o_{n_O}) = \alpha(\mathbb{P}(C)) \cdot \left(\prod_{i=1}^{n_O} \mathbb{P}(O_i = o_i | C) \right)$$

- Note, that this is entirely independent of the *unknown random variables* U_1, \dots, U_{n_U} !
- Also, note that this is just a fancy way of saying “first, compute all the numerators, then divide all of them by their sums”.

- ▶ Putting things together, we get:

$$\begin{aligned}\mathbb{P}(\text{cavity}|\text{toothache} = \text{T}) &= \alpha(\mathbb{P}(\text{cavity}) \cdot \mathbb{P}(\text{toothache} = \text{T}|\text{cavity})) \\ &= \alpha(\langle P(\text{cavity}) \cdot P(\text{toothache} \mid \text{cavity}), P(\neg\text{cavity}) \cdot P(\text{toothache} \mid \neg\text{cavity}) \rangle)\end{aligned}$$

- ▶ Say we have $P(\text{cavity}) = 0.1$, $P(\text{toothache} \mid \text{cavity}) = 0.8$, and $P(\text{toothache} \mid \neg\text{cavity}) = 0.05$.
Then

$$\mathbb{P}(\text{cavity}|\text{toothache} = \text{T}) = \alpha(\langle 0.1 \cdot 0.8, 0.9 \cdot 0.05 \rangle) = \alpha(\langle 0.08, 0.045 \rangle)$$

$0.08 + 0.045 = 0.125$, hence

$$\mathbb{P}(\text{cavity}|\text{toothache} = \text{T}) = \left\langle \frac{0.08}{0.125}, \frac{0.045}{0.125} \right\rangle = \langle 0.64, 0.36 \rangle$$

Naive Bayes Classification

We can use a **naive Bayes model** as a very simple *classifier*.

- ▶ Assume we want to classify newspaper articles as one of the categories *politics*, *sports*, *business*, *fluff*, etc. based on the words they contain.
- ▶ Given a large set of articles, we can determine the relevant **probabilities** by counting the occurrences of the categories $\mathbb{P}(\text{category})$, and of words per category – i.e. $\mathbb{P}(\text{word}_i | \text{category})$ for some (huge) list of words $(\text{word}_i)_{i=1}^n$.
- ▶ We assume that the occurrence of each word is **conditionally independent** of the occurrence of any other word given the category of the document. (This assumption is clearly wrong, but it makes the model simple and often works well in practice.) (\leadsto “Idiot Bayes model”)
- ▶ Given a new article, we just count the occurrences k_i of the words in it and compute

$$\mathbb{P}(\text{category} | \text{word}_1 = k_1, \dots, \text{word}_n = k_n) = \alpha(\mathbb{P}(\text{category}) \cdot \left(\prod_{i=1}^n \mathbb{P}(\text{word}_i = k_i | \text{category}) \right))$$

- ▶ We then choose the category with the highest probability.

22.2.3 Inference by Enumeration

- ▶ The rules we established for **naive Bayes models**, i.e. **Bayes's theorem**, the **product rule** and **chain rule**, **marginalization** and **normalization**, are *general* techniques for probabilistic reasoning, and their usefulness is not limited to the **naive Bayes models**.
- ▶ More generally:
- ▶ **Theorem 2.23.** Let $Q, E_1, \dots, E_{n_E}, U_1, \dots, U_{n_U}$ be *random variables* and $D := \text{dom}(U_1) \times \dots \times \text{dom}(U_{n_U})$. Then

$$\mathbb{P}(Q|E_1 = e_1, \dots, E_{n_E} = e_{n_E}) = \alpha \left(\sum_{u \in D} \mathbb{P}(Q, E_1 = e_1, \dots, E_{n_E} = e_{n_E}, U_1 = u_1, \dots, U_{n_U} = u_{n_U}) \right)$$

We call Q the **query variable**, E_1, \dots, E_{n_E} the **evidence**, and U_1, \dots, U_{n_U} the **unknown** (or **hidden**) **variables**, and computing a **conditional probability** this way **enumeration**.

- ▶ Note that this is just a “mathy” way of saying we
 1. sum over all relevant entries of the **full joint probability distribution** of the variables, and
 2. normalize the result to yield a **probability distribution**.

22.2.4 Example – The Wumpus is Back

Example: The Wumpus is Back

- ▶ We have a maze where
 - ▶ Every cell except $[1, 1]$ possibly contains a *pit*, with 20% probability.
 - ▶ pits cause a *breeze* in neighboring cells (we forget the wumpus and the gold for now)
- ▶ Where should the agent go, if there is a breeze at $[1, 2]$ and $[2, 1]$?
- ▶ Pure logical inference can conclude nothing about which square is *most likely* to be safe!

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 B OK	2,2	3,2	4,2
1,1 OK	2,1 B OK	3,1	4,1

We can model this using the Boolean random variables:

- ▶ $P_{i,j}$ for $i, j \in \{1, 2, 3, 4\}$, stating there is a pit at square $[i, j]$, and
 - ▶ $B_{i,j}$ for $(i, j) \in \{(1, 1), (1, 2), (2, 1)\}$, stating there is a breeze at square $[i, j]$
- ⇒ let's apply our machinery!

- **First:** Let's try to compute the full joint probability distribution

$\mathbb{P}(P_{1,1}, \dots, P_{4,4}, B_{1,1}, B_{1,2}, B_{2,1})$.

1. By the product rule, this is equal to

$$\mathbb{P}(B_{1,1}, B_{1,2}, B_{2,1} | P_{1,1}, \dots, P_{4,4}) \cdot \mathbb{P}(P_{1,1}, \dots, P_{4,4}).$$

2. Note that $\mathbb{P}(B_{1,1}, B_{1,2}, B_{2,1} | P_{1,1}, \dots, P_{4,4})$ is either 1 (if all the $B_{i,j}$ are consistent with the positions of the pits $P_{k,l}$) or 0 (otherwise).

3. Since the pits are spread independently, we have

$$\mathbb{P}(P_{1,1}, \dots, P_{4,4}) = \prod_{i,j=1,1}^{4,4} \mathbb{P}(P_{i,j})$$

- \leadsto We know all of these probabilities.

- \leadsto We can now use enumeration to compute

$$\mathbb{P}(P_{i,j} | \langle \text{known} \rangle) = \alpha(\sum_{\langle \text{unknowns} \rangle} \mathbb{P}(P_{i,j}, \langle \text{known} \rangle, \langle \text{unknowns} \rangle))$$

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 B OK	2,2	3,2	4,2
1,1 OK	2,1 B OK	3,1	4,1

Wumpus Continued

► **Problem:** We only know $P_{i,j}$ for three fields. If we want to compute e.g. $P_{1,3}$ via enumeration, that leaves $2^{4^2-4} = 4096$ terms to sum over!

► **Let's do better.**

► Let $b := \neg B_{1,1} \wedge B_{1,2} \wedge B_{2,1}$ (All the breezes we know about)

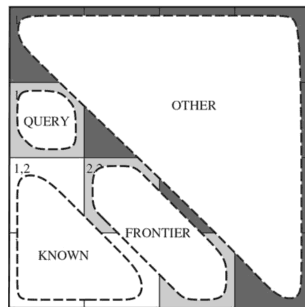
► Let $p := \neg P_{1,1} \wedge \neg P_{1,2} \wedge \neg P_{2,1}$. (All the pits we know about)

► Let $F := \{P_{3,1} \wedge P_{2,2}, \neg P_{3,1} \wedge P_{2,2}, P_{3,1} \wedge \neg P_{2,2}, \neg P_{3,1} \wedge \neg P_{2,2}\}$ (the current "frontier")

► Let O be (the set of assignments for) all the other variables $P_{i,j}$. (i.e. except p , F and our query $P_{1,3}$)

Then the observed breezes b are **conditionally independent** of O given p and F . (Whether there is a pit anywhere else does not influence the breezes we observe.)

► $\Rightarrow P(b \mid P_{1,3}, p, O, F) = P(b \mid P_{1,3}, p, F)$. Let's exploit this!



Optimized Wumpus

► In particular:

$$\begin{aligned}\mathbb{P}(P_{1,3}|p, b) &= \alpha \left(\sum_{o \in O, f \in F} \mathbb{P}(P_{1,3}, b, p, f, o) \right) = \alpha \left(\sum_{o \in O, f \in F} P(b \mid P_{1,3}, p, o, f) \cdot \mathbb{P}(P_{1,3}, p, f, o) \right) \\ &= \alpha \left(\sum_{f \in F} \sum_{o \in O} P(b \mid P_{1,3}, p, f) \cdot \mathbb{P}(P_{1,3}, p, f, o) \right) = \alpha \left(\sum_{f \in F} P(b \mid P_{1,3}, p, f) \cdot \left(\sum_{o \in O} \mathbb{P}(P_{1,3}, p, f, o) \right) \right) \\ &= \alpha \left(\sum_{f \in F} P(b \mid P_{1,3}, p, f) \cdot \left(\sum_{o \in O} \mathbb{P}(P_{1,3}) \cdot P(p) \cdot P(f) \cdot P(o) \right) \right) \\ &= \alpha \left(\mathbb{P}(P_{1,3}) \cdot P(p) \cdot \underbrace{\left(\sum_{f \in F} P(b \mid P_{1,3}, p, f) \cdot P(f) \right)}_{\in \{0,1\}} \cdot \underbrace{\left(\sum_{o \in O} P(o) \right)}_{=1} \right)\end{aligned}$$

↪ this is just a sum over the frontier, i.e. 4 terms ☺

- So: $\mathbb{P}(P_{1,3}|p, b) = \alpha(\langle 0.2 \cdot (0.8)^3 \cdot (1 \cdot 0.04 + 1 \cdot 0.16 + 1 \cdot 0.16 + 0), 0.8 \cdot (0.8)^3 \cdot (1 \cdot 0.04 + 1 \cdot 0.16 + 0 + 0) \rangle) \approx \langle 0.31, 0.69 \rangle$
- Analogously: $\mathbb{P}(P_{3,1}|p, b) = \langle 0.31, 0.69 \rangle$ and $\mathbb{P}(P_{2,2}|p, b) = \langle 0.86, 0.14 \rangle$ (\Rightarrow avoid [2,2]!)

- ▶ In general, when you want to reason probabilistically, a good heuristic is:
 1. Try to frame the **full joint probability distribution** in terms of the probabilities you know. Exploit **product rule/chain rule**, **independence**, **conditional independence**, **marginalization** and **domain knowledge** (as e.g. $\mathbb{P}(b|p, f) \in \{0, 1\}$)
 \leadsto the problem can be solved at all!
 2. **Simplify**: Start with the equation for enumeration:

$$\mathbb{P}(Q|E_1, \dots) = \alpha \left(\sum_{u \in U} \mathbb{P}(Q, E_1, \dots, U_1 = u_1, \dots) \right)$$

3. Substitute by the result of 1., and again, exploit all of our machinery
4. Implement the resulting (system of) equation(s)
5. ???
6. Profit

- ▶ Probability distributions and conditional probability distributions allow us to represent random variables as convenient datastructures in an implementation (Assuming they are finite domain...)
- ▶ The full joint probability distribution allows us to compute all probabilities of statements about the random variables contained (But possibly inefficient)
- ▶ Marginalization and normalization are the specific techniques for extracting the specific probabilities we are interested in from the full joint probability distribution.
- ▶ The product and chain rule, exploiting (conditional) independence, Bayes' Theorem, and of course domain specific knowledge allow us to do so much more efficiently.
- ▶ Naive Bayes models are one example where all these techniques come together.

Chapter 23

Probabilistic Reasoning: Bayesian Networks

23.1 Introduction

► Example 1.1 (From Russell/Norvig).

- I got very valuable stuff at home. So I bought an alarm. Unfortunately, the alarm just rings at home, doesn't call me on my mobile.
- I've got two neighbors, Mary and John, who'll call me if they hear the alarm.
- The problem is that, sometimes, the alarm is caused by an earthquake.
- Also, John might confuse the alarm with his telephone, and Mary might miss the alarm altogether because she typically listens to loud music.

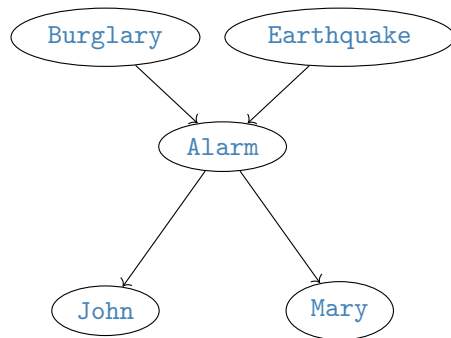
↪ Random variables: Burglary, Earthquake, Alarm, John, Mary. Given that both John and Mary call me, what is the probability of a burglary?

- ↪ This is *almost* a naive Bayes model, but with multiple causes (Burglary and Earthquake) for the Alarm, which in turn may cause John and/or Mary.

John, Mary, and My Alarm: Assumptions

We assume:

- ▶ We (should) know $\mathbb{P}(\text{Alarm}|\text{Burglary}, \text{Earthquake})$, $\mathbb{P}(\text{John}|\text{Alarm})$, and $\mathbb{P}(\text{Mary}|\text{Alarm})$.
- ▶ Burglary and Earthquake are independent.
- ▶ John and Mary are conditionally independent given Alarm.
- ▶ Moreover: Both John and Mary are conditionally independent of any other random variables in the graph given Alarm. (Only Alarm causes them, and everything else only causes them indirectly through Alarm)
- ▶ **First Step:** Construct the full joint probability distribution,
- ▶ **Second Step:** Use enumeration to compute $\mathbb{P}(\text{Burglary}|\text{John} = \text{T}, \text{Mary} = \text{T})$.



John, Mary, and My Alarm: The Distribution



$$\begin{aligned} & \mathbb{P}(\text{John}, \text{Mary}, \text{Alarm}, \text{Burglary}, \text{Earthquake}) \\ &= \mathbb{P}(\text{John} | \text{Mary}, \text{Alarm}, \text{Burglary}, \text{Earthquake}) \cdot \mathbb{P}(\text{Mary} | \text{Alarm}, \text{Burglary}, \text{Earthquake}) \\ & \quad \cdot \mathbb{P}(\text{Alarm} | \text{Burglary}, \text{Earthquake}) \cdot \mathbb{P}(\text{Burglary} | \text{Earthquake}) \cdot \mathbb{P}(\text{Earthquake}) \\ &= \mathbb{P}(\text{John} | \text{Alarm}) \cdot \mathbb{P}(\text{Mary} | \text{Alarm}) \cdot \mathbb{P}(\text{Alarm} | \text{Burglary}, \text{Earthquake}) \cdot \mathbb{P}(\text{Burglary}) \cdot \mathbb{P}(\text{Earthquake}) \end{aligned}$$

- We plug into the [equation](#) for enumeration:

$$\begin{aligned} & \mathbb{P}(\text{Burglary} | \text{John} = \text{T}, \text{Mary} = \text{T}) = \alpha \left(\mathbb{P}(\text{Burglary}) \sum_{a \in \{\text{T}, \text{F}\}} P(\text{John} | \text{Alarm} = a) \cdot P(\text{Mary} | \text{Alarm} = a) \right. \\ & \quad \cdot \left. \sum_{q \in \{\text{T}, \text{F}\}} \mathbb{P}(\text{Alarm} = a | \text{Burglary}, \text{Earthquake} = q) P(\text{Earthquake} = q) \right) \end{aligned}$$

- \rightsquigarrow Now let's scale things up to arbitrarily many [variables](#)!

Bayesian Networks: Definition

► **Definition 1.2.** A **Bayesian network** consists of

1. a directed acyclic graph $\langle \mathcal{X}, E \rangle$ of random variables $\mathcal{X} = \{X_1, \dots, X_n\}$, and
2. a conditional probability distribution $\mathbb{P}(X_i | \text{Parents}(X_i))$ for every $X_i \in \mathcal{X}$ (also called the **CPT** for **conditional probability table**)

such that every X_i is conditionally independent of any conjunctions of non-descendants of X_i given $\text{Parents}(X_i)$.

► **Definition 1.3.** Let $\langle \mathcal{X}, E \rangle$ be a directed acyclic graph, $X \in \mathcal{X}$, and E^* the reflexive transitive closure of E . The **non-descendants** of X are the elements of the set $\text{NonDesc}(X) := \{Y \mid (X, Y) \notin E^*\} \setminus \text{Parents}(X)$.

► Note that the roots of the graph are conditionally independent given the empty set; i.e. they are independent.

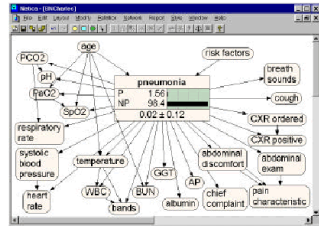
► **Theorem 1.4.** The full joint probability distribution of a Bayesian network $\langle \mathcal{X}, E \rangle$ is given by

$$\mathbb{P}(X_1, \dots, X_n) = \prod_{X_i \in \mathcal{X}} \mathbb{P}(X_i | \text{Parents}(X_i))$$

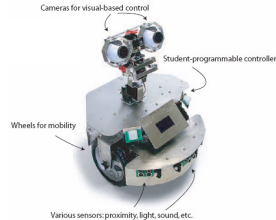
Some Applications

- ▶ A ubiquitous problem: Observe “symptoms”, need to infer “causes”.

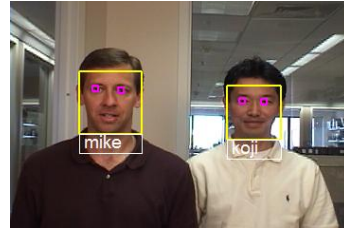
Medical Diagnosis



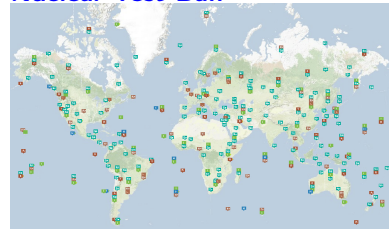
Self-Localization



Face Recognition



Nuclear Test Ban



23.2 Constructing Bayesian Networks

Compactness of Bayesian Networks

- **Definition 2.1.** Given random variables X_1, \dots, X_n with finite domains D_1, \dots, D_n , the size of $\mathcal{B} := \langle \{X_1, \dots, X_n\}, E \rangle$ is defined as

$$\text{size}(\mathcal{B}) := \sum_{i=1}^n |D_i| \cdot \left(\prod_{X_j \in \text{Parents}(X_i)} |D_j| \right)$$

- **Note:** $\text{size}(\mathcal{B}) \hat{=}$ The total number of entries in the conditional probability distributions.

Compactness of Bayesian Networks

- **Definition 2.5.** Given random variables X_1, \dots, X_n with finite domains D_1, \dots, D_n , the size of $\mathcal{B} := \langle \{X_1, \dots, X_n\}, E \rangle$ is defined as

$$\text{size}(\mathcal{B}) := \sum_{i=1}^n |D_i| \cdot \left(\prod_{X_j \in \text{Parents}(X_i)} |D_j| \right)$$

- **Note:** $\text{size}(\mathcal{B}) \hat{=}$ The total number of entries in the conditional probability distributions.
- **Note:** Smaller BN \leadsto need to assess less probabilities, more efficient inference.
- **Observation 2.6.** Explicit full joint probability distribution has size $\prod_{i=1}^n |D_i|$.
- **Observation 2.7.** If $|\text{Parents}(X_i)| \leq k$ for every X_i , and D_{\max} is the largest random variable domain, then $\text{size}(\mathcal{B}) \leq n |D_{\max}|^{k+1}$.

Compactness of Bayesian Networks

- **Definition 2.9.** Given random variables X_1, \dots, X_n with finite domains D_1, \dots, D_n , the size of $\mathcal{B} := \langle \{X_1, \dots, X_n\}, E \rangle$ is defined as

$$\text{size}(\mathcal{B}) := \sum_{i=1}^n |D_i| \cdot \left(\prod_{X_j \in \text{Parents}(X_i)} |D_j| \right)$$

- **Note:** $\text{size}(\mathcal{B}) \hat{=}$ The total number of entries in the conditional probability distributions.
- **Note:** Smaller BN \leadsto need to assess less probabilities, more efficient inference.
- **Observation 2.10.** Explicit full joint probability distribution has size $\prod_{i=1}^n |D_i|$.
- **Observation 2.11.** If $|\text{Parents}(X_i)| \leq k$ for every X_i , and D_{\max} is the largest random variable domain, then $\text{size}(\mathcal{B}) \leq n |D_{\max}|^{k+1}$.
- **Example 2.12.** For $|D_{\max}| = 2$, $n = 20$, $k = 4$ we have $2^{20} = 1048576$ probabilities, but a Bayesian network of size $\leq 20 \cdot 2^5 = 640 \dots!$
- In the worst case, $\text{size}(\mathcal{B}) = n \cdot (\prod_{i=1}^1 n) |D_i|$, namely if every variable depends on all its predecessors in the chosen variable ordering.
- **Intuition:** BNs are compact – i.e. of small size – if each variable is directly influenced only by few of its predecessor variables.

► To keep our **Bayesian networks** small, we can:

1. **Reduce the number of edges:** \Rightarrow Order the variables to allow for exploiting **conditional independence** (causes before effects), or

2. **represent the conditional probability distributions efficiently:**

2.1 For **Boolean random variables** X , we only need to store $P(X = T | \text{Parents}(X))$

$(P(X = F | \text{Parents}(X)) = 1 - P(X = T | \text{Parents}(X)))$ (Cuts the number of entries in half!)

2.2 Introduce different **kinds** of nodes exploiting domain knowledge; e.g. **deterministic** and **noisy disjunction nodes**.

Reducing Edges: Variable Order Matters

- ▶ Given a set of random variables X_1, \dots, X_n , consider the following (impractical, but illustrative) pseudo-algorithm for constructing a Bayesian network:

- ▶ **Definition 2.13 (BN construction algorithm).**

1. Initialize $BN := \langle \{X_1, \dots, X_n\}, E \rangle$ where $E = \emptyset$.
2. Fix any variable ordering, X_1, \dots, X_n .
3. **for** $i := 1, \dots, n$ **do**
 - a. Choose a minimal set $\text{Parents}(X_i) \subseteq \{X_1, \dots, X_{i-1}\}$ such that

$$\mathbb{P}(X_i | X_{i-1}, \dots, X_1) = \mathbb{P}(X_i | \text{Parents}(X_i))$$

- b. For each $X_j \in \text{Parents}(X_i)$, insert (X_j, X_i) into E .
- c. Associate X_i with $\mathbb{P}(X_i | \text{Parents}(X_i))$.

- ▶ **Attention:** Which variables we need to include into $\text{Parents}(X_i)$ depends on what “ $\{X_1, \dots, X_{i-1}\}$ ” is ... !

- ▶ **Thus:** The size of the resulting BN depends on the chosen variable ordering X_1, \dots, X_n .

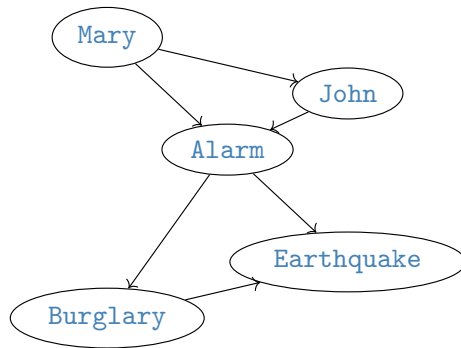
- ▶ **In Particular:** The size of a Bayesian network is *not* a fixed property of the domain. It depends on the skill of the designer.

John and Mary Depend on the Variable Order!

► **Example 2.14.** Mary, John, Alarm, Burglary, Earthquake.

John and Mary Depend on the Variable Order!

► **Example 2.15.** Mary, John, Alarm, Burglary, Earthquake.

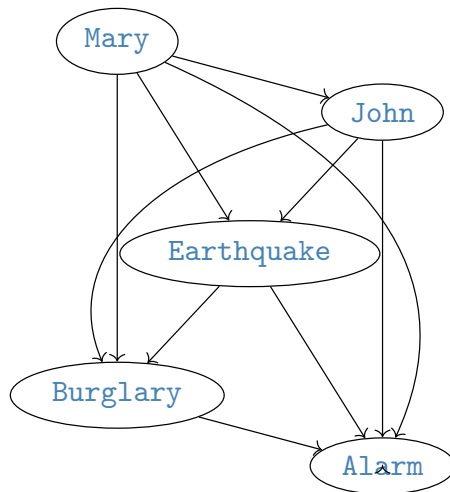


John and Mary Depend on the Variable Order! Ctd.

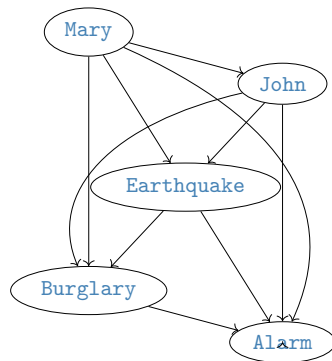
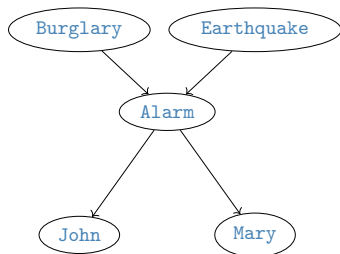
- **Example 2.16.** Mary, John, Earthquake, Burglary, Alarm.

John and Mary Depend on the Variable Order! Ctd.

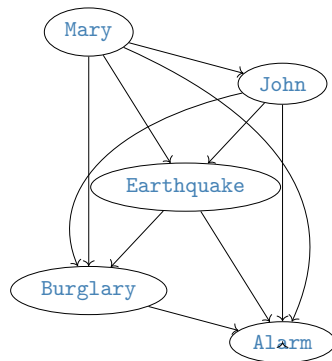
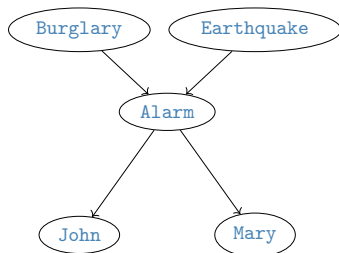
► **Example 2.17.** Mary, John, Earthquake, Burglary, Alarm.



John and Mary, What Went Wrong?



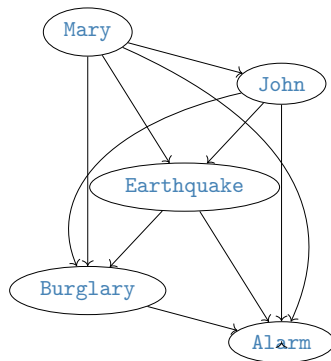
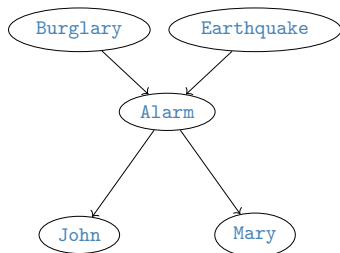
John and Mary, What Went Wrong?



► **Intuition:** These BNs link from *effects* to their *causes*!

⇒ Even though **Mary** and **John** are **conditionally independent** given **Alarm**, this is not exploited, since **Alarm** is not ordered before **Mary** and **John**!

John and Mary, What Went Wrong?



► **Intuition:** These BNs link from *effects* to their *causes*!

⇒ Even though **Mary** and **John** are **conditionally independent** given **Alarm**, this is not exploited, since **Alarm** is not ordered before **Mary** and **John**!

⇒ **Rule of Thumb:** We should **order** causes before symptoms.

- **Definition 2.18.** A node X in a Bayesian network is called **deterministic**, if its value is completely determined by the values of $\text{Parents}(X)$.

- ▶ **Definition 2.21.** A node X in a Bayesian network is called **deterministic**, if its value is completely determined by the values of $\text{Parents}(X)$.
- ▶ **Example 2.22.** The *sum of two dice throws* S is entirely determined by the values of the two dice *First* and *Second*.

- ▶ **Definition 2.24.** A node X in a Bayesian network is called **deterministic**, if its value is completely determined by the values of $\text{Parents}(X)$.
- ▶ **Example 2.25.** The *sum of two dice throws* S is entirely determined by the values of the two dice *First* and *Second*.
- ▶ **Example 2.26.** In the *Wumpus* example, the *breezes* are entirely determined by the *pits*

- ▶ **Definition 2.27.** A node X in a Bayesian network is called **deterministic**, if its value is completely determined by the values of $\text{Parents}(X)$.
- ▶ **Example 2.28.** The *sum of two dice throws* S is entirely determined by the values of the two dice *First* and *Second*.
- ▶ **Example 2.29.** In the *Wumpus* example, the *breezes* are entirely determined by the *pits*
- ▶ \leadsto *Deterministic* nodes model direct, *causal* relationships.
- ▶ \leadsto If X is **deterministic**, then $\mu(X \mid \text{Parents}(X)) \in \{0, 1\}$

- ▶ **Definition 2.30.** A node X in a Bayesian network is called **deterministic**, if its value is completely determined by the values of $\text{Parents}(X)$.
- ▶ **Example 2.31.** The *sum of two dice throws* S is entirely determined by the values of the two dice *First* and *Second*.
- ▶ **Example 2.32.** In the *Wumpus* example, the *breezes* are entirely determined by the *pits*
- ▶ \leadsto *Deterministic* nodes model direct, *causal* relationships.
- ▶ \leadsto If X is **deterministic**, then $\mu(X \mid \text{Parents}(X)) \in \{0, 1\}$
- ▶ \leadsto we can replace the **conditional probability distribution** $\mathbb{P}(X \mid \text{Parents}(X))$ by a boolean function.

Representing Conditional Distributions: Noisy Nodes

- Sometimes, values of nodes are “almost deterministic”:

- **Example 2.33 (Inhibited Causal Dependencies).**

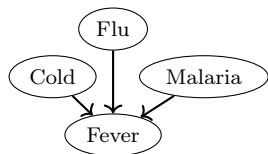
Assume the network on the right contains *all* possible causes of fever. (Or add a dummy-node for “other causes”)

If there is a fever, then *one* of them (at least) must be the cause, but none of them *necessarily* cause a fever: The causal relation between parent and child is **inhibited**.

↪ We can model the **inhibitions** by individual **inhibition factors** q_d .

- **Definition 2.34.** The **conditional probability distribution** of a **noisy disjunction node** X with $\text{Parents}(X) = X_1, \dots, X_n$ in a **Bayesian network** is given by $P(X \mid X_1, \dots, X_n) = 1 - (\prod_{\{j \mid X_j = \top\}} q_j)$, where the q_i are the **inhibition factors** of $X_i \in \text{Parents}(X)$, defined as $q_i := P(\neg X \mid \neg X_1, \dots, \neg X_{i-1}, X_i, \neg X_{i+1}, \dots, \neg X_n)$

- ↪ Instead of a distribution with 2^k parameters, we only need k parameters!



Representing Conditional Distributions: Noisy Nodes

► **Example 2.35.** Assume the following **inhibition factors** for **??**:

$$q_{\text{cold}} = P(\neg \text{fever} \mid \text{cold}, \neg \text{flu}, \neg \text{malaria}) = 0.6$$

$$q_{\text{flu}} = P(\neg \text{fever} \mid \neg \text{cold}, \text{flu}, \neg \text{malaria}) = 0.2$$

$$q_{\text{malaria}} = P(\neg \text{fever} \mid \neg \text{cold}, \neg \text{flu}, \text{malaria}) = 0.1$$

If we model Fever as a **noisy disjunction node**, then the general rule $P(X_i \mid \text{Parents}(X_i)) = \prod_{\{j \mid x_j = \tau\}} q_j$ for the **CPT** gives the following table:

Cold	Flu	Malaria	$P(\text{Fever})$	$P(\neg \text{Fever})$
F	F	F	0.0	1.0
F	F	T	0.9	0.1
F	T	F	0.8	0.2
F	T	T	0.98	$0.02 = 0.2 \cdot 0.1$
T	F	F	0.4	0.6
T	F	T	0.94	$0.06 = 0.6 \cdot 0.1$
T	T	F	0.88	$0.12 = 0.6 \cdot 0.2$
T	T	T	0.988	$0.012 = 0.6 \cdot 0.2 \cdot 0.1$

- ▶ Note that **deterministic** nodes and **noisy disjunction nodes** are just two examples of “specialized” kinds of nodes in a **Bayesian network**.
- ▶ In general, noisy logical relationships in which a variable depends on k **parents** can be described by $\mathcal{O}(k)$ parameters instead of $\mathcal{O}(2^k)$ for the full conditional probability table. This can make **assessment** (and learning) **tractable**.
- ▶ **Example 2.36.** The CPCS network [PraProMid:kelbn94] uses noisy-OR and noisy-MAX distributions to model relationships among diseases and symptoms in internal medicine. With 448 nodes and 906 links, it requires only 8,254 values instead of 133,931,430 for a network with full **conditional probability distributions**.

23.3 Inference in Bayesian Networks

Probabilistic Inference Tasks in Bayesian Networks

- ▶ Remember:

- ▶ **Definition 3.1 (Probabilistic Inference Task).** Let

$X_1, \dots, X_n = Q_1, \dots, Q_{n_Q}, E_1, \dots, E_{n_E}, U_1, \dots, U_{n_U}$ be a set of random variables, a probabilistic inference task.

We wish to compute the conditional probability distribution $\mathbb{P}(Q_1, \dots, Q_{n_Q} | E_1 = e_1, \dots, E_{n_E} = e_{n_E})$. We call

- ▶ a Q_1, \dots, Q_{n_Q} the query variables,
 - ▶ a E_1, \dots, E_{n_E} the evidence variables, and
 - ▶ U_1, \dots, U_{n_U} the hidden variables.
- ▶ We know the full joint probability distribution: $\mathbb{P}(X_1, \dots, X_n) = \prod_{i=1}^n \mathbb{P}(X_i | \text{Parents}(X_i))$
 - ▶ And we know about enumeration:

$$\mathbb{P}(Q_1, \dots, Q_{n_Q} | E_1 = e_1, \dots, E_{n_E} = e_{n_E}) = \\ \alpha \left(\sum_{u \in D_U} \mathbb{P}(Q_1, \dots, Q_{n_Q}, E_1 = e_1, \dots, E_{n_E} = e_{n_E}, U_1 = u_1, \dots, U_{n_U} = u_{n_U}) \right)$$

(where $D_U = \text{dom}(U_1) \times \dots \times \text{dom}(U_{n_U})$)

Enumeration: The Alarm-Example

- ▶ Remember our example: $\mathbb{P}(\text{Burglary} | \text{John}, \text{Mary})$
(hidden variables: Alarm, Earthquake)

$$= \alpha \left(\sum_{b_a, b_e \in \{T, F\}} P(\text{John}, \text{Mary}, \text{Alarm} = b_a, \text{Earthquake} = b_e, \text{Burglary}) \right)$$

$$= \alpha \left(\sum_{b_a, b_e \in \{T, F\}} P(\text{John} | \text{Alarm} = b_a) \cdot P(\text{Mary} | \text{Alarm} = b_a) \cdot \right. \\ \left. \cdot \mathbb{P}(\text{Alarm} = b_a | \text{Earthquake} = b_e, \text{Burglary}) \cdot P(\text{Earthquake} = b_e) \cdot \mathbb{P}(\text{Burglary}) \right)$$

- ▶ \leadsto These are 5 factors in 4 summands ($b_a, b_e \in \{T, F\}$) over two cases ($\text{Burglary} \in \{T, F\}$),
- ▶ \leadsto 38 arithmetic operations (+3 for α)
- ▶ **General worst case:** $\mathcal{O}(n2^n)$
- ▶ **Let's do better!**

Enumeration: First Improvement

► **Some abbreviations:** $j := \text{John}$, $m := \text{Mary}$, $a := \text{Alarm}$, $e := \text{Earthquake}$, $b := \text{Burglary}$,

►

$$\mathbb{P}(b|j, m) = \alpha \left(\sum_{b_a, b_e \in \{T, F\}} P(j \mid a = b_a) \cdot P(m \mid a = b_a) \cdot \mathbb{P}(a = b_a | e = b_e, b) \cdot P(e = b_e) \cdot \mathbb{P}(b) \right)$$

► Let's "optimize":

$$\mathbb{P}(b|j, m) = \alpha(\mathbb{P}(b) \cdot \left(\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot \left(\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a | e = b_e, b) \cdot P(j \mid a = b_a) \cdot P(m \mid a = b_a) \right) \right))$$

\leadsto 3 factors in 2 summand + 2 factors in 2 summands + two factors in the outer product, over two cases = 28 arithmetic operations (+3 for α)

Second Improvement: Variable Elimination 1

- ▶ Consider $\mathbb{P}(j|b = \text{T})$.
- ▶ Using enumeration:

$$= \alpha(P(b) \cdot \left(\sum_{b_e \in \{\text{T}, \text{F}\}} P(e = b_e) \cdot \left(\sum_{a_e \in \{\text{T}, \text{F}\}} P(a = a_e \mid e = b_e, b) \cdot \mathbb{P}(j|a = a_e) \cdot \underbrace{\left(\sum_{a_m \in \{\text{T}, \text{F}\}} P(m = a_m \mid a = a_e) \right)}_{=1} \right) \right)$$

$\leadsto \mathbb{P}(\text{John}|\text{Burglary} = \text{T})$ does not depend on Mary (duh...)

- ▶ **More generally:**
- ▶ **Lemma 3.2.** Given a query $\mathbb{P}(Q_1, \dots, Q_{n_Q} | E_1 = e_1, \dots, E_{n_E} = e_{n_E})$, we can ignore (and remove) all *hidden* leaves of the *Bayesian network*.
- ▶ ...doing so yields new leaves, which we can then ignore again, etc., until:
- ▶ **Lemma 3.3.** Given a query $\mathbb{P}(Q_1, \dots, Q_{n_Q} | E_1 = e_1, \dots, E_{n_E} = e_{n_E})$, we can ignore (and remove) all *hidden variables* that are not ancestors of any of the Q_1, \dots, Q_{n_Q} or E_1, \dots, E_{n_E} .

Enumeration: First Algorithm

- Assume the X_1, \dots, X_n are topologically sorted

(causes before effects)

```
function ENUMERATE-QUERY( $Q, \langle E_1 = e_1, \dots, E_{n_E} = e_{n_E} \rangle$ )  
     $P := \langle \rangle$  /* =  $\mathbb{P}(Q | E_i = e_i)$  */  
     $X_1, \dots, X_n :=$  variables filtered according to ??, topologically sorted  
    for all  $q \in \text{dom}(Q)$  do  
         $P_i := \text{ENUMALL}(\langle X_1, \dots, X_n \rangle, \langle E_1 = e_1, \dots, E_{n_E} = e_{n_E}, Q = q \rangle)$   
    return  $\alpha(P)$   
  
function ENUMALL( $\langle Y_1, \dots, Y_{n_Y} \rangle, \langle A_1 = a_1, \dots, A_{n_A} = a_{n_A} \rangle$ )  
    /* By construction,  $\text{Parents}(Y_1) \subset \{A_1, \dots, A_{n_A}\}$  */  
    if  $n_Y = 0$  then return 1.0  
    else if  $Y_1 = A_j$  then return  $P(A_j = a_j \mid \text{Parents}(A_j)) \cdot \text{ENUMALL}(\langle Y_2, \dots, Y_{n_Y} \rangle, \langle A_1 = a_1, \dots, A_{n_A} = a_{n_A} \rangle)$   
    else return  $\sum_{y \in \text{dom}(Y_1)} P(Y_1 = y \mid \text{Parents}(Y_1)) \cdot \text{ENUMALL}(\langle Y_2, \dots, Y_{n_Y} \rangle, \langle A_1 = a_1, \dots, A_{n_A} = a_{n_A}, Y_1 = y \rangle)$ 
```

- General worst case Complexity: $\mathcal{O}(2^n)$ – better, but still not great

Enumeration: Example

► Variable order: b, e, a, j, m

ENUMERATE-QUERY($b, \langle j = T, m = T \rangle$)

$$\mathbb{P}(b | j = T, m = T) =$$

Enumeration: Example

► Variable order: b, e, a, j, m

ENUMERATE-QUERY($b, \langle j = T, m = T \rangle$)

$$\mathbb{P}(b | j = T, m = T) =$$

Enumeration: Example

- ▶ Variable order: b, e, a, j, m
 - ▶ $P_0 := \text{ENUMALL}(\langle b, e, a, j, m \rangle, \langle j = \text{T}, m = \text{T}, b = \text{T} \rangle)$
 - ▶ $P_1 := \text{ENUMALL}(\langle b, e, a, j, m \rangle, \langle j = \text{T}, m = \text{T}, b = \text{F} \rangle)$
 - $\Leftarrow \alpha(\langle P_0, P_1 \rangle)$

$$\mathbb{P}(b | j = \text{T}, m = \text{T}) = \alpha()$$

Enumeration: Example

► Variable order: b, e, a, j, m

$$\text{► } P_0 := \underbrace{\text{ENUMALL}(\langle b, e, a, j, m \rangle, \langle j = T, m = T, b = T \rangle)}$$

$$= P(b) \cdot \text{ENUMALL}(\langle e, a, j, m \rangle, \langle j = T, m = T, b = T \rangle)$$

$$\text{► } P_1 := \underbrace{\text{ENUMALL}(\langle b, e, a, j, m \rangle, \langle j = T, m = T, b = F \rangle)}$$

$$= P(\neg b) \cdot \text{ENUMALL}(\langle e, a, j, m \rangle, \langle j = T, m = T, b = F \rangle)$$

$$\Leftarrow \alpha(\langle P_0, P_1 \rangle)$$

$$\mathbb{P}(b|j = T, m = T) = \alpha(\mathbb{P}(b) \cdot \cdot)$$

Enumeration: Example

- ▶ Variable order: b, e, a, j, m
 - ▶ $P_0 := P(b) \cdot \text{ENUMALL}(\langle e, a, j, m \rangle, \langle j = T, m = T, b = T \rangle)$
 - ▶ $P_1 := P(\neg b) \cdot \text{ENUMALL}(\langle e, a, j, m \rangle, \langle j = T, m = T, b = F \rangle)$
 - ▶ $\Leftarrow \alpha(\langle P_0, P_1 \rangle)$

$$\mathbb{P}(b|j = T, m = T) = \alpha(\mathbb{P}(b) \cdot \cdot)$$

Enumeration: Example

► Variable order: b, e, a, j, m

$$\begin{aligned} \text{► } P_0 &:= P(b) \cdot \underbrace{\text{ENUMALL}(\langle e, a, j, m \rangle, \langle j = \text{T}, m = \text{T}, b = \text{T} \rangle)} \\ &= (\sum_{b_e \in \{\text{T}, \text{F}\}} P(e = b_e) \cdot \text{ENUMALL}(\langle a, j, m \rangle, \langle j = \text{T}, m = \text{T}, b = \text{T}, e = b_e \rangle)) \end{aligned}$$

$$\begin{aligned} \text{► } P_1 &:= P(\neg b) \cdot \underbrace{\text{ENUMALL}(\langle e, a, j, m \rangle, \langle j = \text{T}, m = \text{T}, b = \text{F} \rangle)} \\ &= (\sum_{b_e \in \{\text{T}, \text{F}\}} P(e = b_e) \cdot \text{ENUMALL}(\langle a, j, m \rangle, \langle j = \text{T}, m = \text{T}, b = \text{F}, e = b_e \rangle)) \end{aligned}$$

$$\Leftarrow \alpha(\langle P_0, P_1 \rangle)$$

$$\mathbb{P}(b|j = \text{T}, m = \text{T}) = \alpha(\mathbb{P}(b) \cdot (\sum_{b_e \in \{\text{T}, \text{F}\}} P(e = b_e) \cdot))$$

Enumeration: Example

► Variable order: b, e, a, j, m

$$\begin{aligned} \text{► } P_0 &:= P(b) \cdot \left[\begin{aligned} &+ P(e) \cdot \text{ENUMALL}(\langle a, j, m \rangle, \langle j = T, m = T, b = T, e = T \rangle) \\ &+ P(\neg e) \cdot \text{ENUMALL}(\langle a, j, m \rangle, \langle j = T, m = T, b = T, e = F \rangle) \end{aligned} \right] \\ \text{► } P_1 &:= P(\neg b) \cdot \left[\begin{aligned} &+ P(e) \cdot \text{ENUMALL}(\langle a, j, m \rangle, \langle j = T, m = T, b = F, e = T \rangle) \\ &+ P(\neg e) \cdot \text{ENUMALL}(\langle a, j, m \rangle, \langle j = T, m = T, b = F, e = F \rangle) \end{aligned} \right] \\ &\Leftarrow \alpha(\langle P_0, P_1 \rangle) \end{aligned}$$

$$\mathbb{P}(b|j = T, m = T) = \alpha(\mathbb{P}(b) \cdot (\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot))$$

Enumeration: Example

► Variable order: b, e, a, j, m

$$\begin{aligned}
 \text{► } P_0 &:= P(b) \cdot \left[\begin{aligned} &P(e) \cdot \underbrace{\text{ENUMALL}(\langle a, j, m \rangle, \langle j = T, m = T, b = T, e = T \rangle)}_{=(\sum_{b_a \in \{T, F\}} P(a = b_a \mid b, e) \cdot \text{ENUMALL}(\langle j, m \rangle \cdot \langle j = T, m = T, b = T, e = T, a = b_a \rangle))} \\ &+ P(\neg e) \cdot \underbrace{\text{ENUMALL}(\langle a, j, m \rangle, \langle j = T, m = T, b = T, e = F \rangle)}_{=(\sum_{b_a \in \{T, F\}} P(a = b_a \mid b, \neg e) \cdot \text{ENUMALL}(\langle j, m \rangle \cdot \langle j = T, m = T, b = T, e = F, a = b_a \rangle))} \end{aligned} \right] \\
 \text{► } P_1 &:= P(\neg b) \cdot \left[\begin{aligned} &P(e) \cdot \underbrace{\text{ENUMALL}(\langle a, j, m \rangle, \langle j = T, m = T, b = F, e = T \rangle)}_{=(\sum_{b_a \in \{T, F\}} P(a = b_a \mid \neg b, e) \cdot \text{ENUMALL}(\langle j, m \rangle \cdot \langle j = T, m = T, b = F, e = T, a = b_a \rangle))} \\ &+ P(\neg e) \cdot \underbrace{\text{ENUMALL}(\langle a, j, m \rangle, \langle j = T, m = T, b = F, e = F \rangle)}_{=(\sum_{b_a \in \{T, F\}} P(a = b_a \mid \neg b, \neg e) \cdot \text{ENUMALL}(\langle j, m \rangle \cdot \langle j = T, m = T, b = F, e = F, a = b_a \rangle))} \end{aligned} \right] \\
 &\Leftarrow \alpha(\langle P_0, P_1 \rangle)
 \end{aligned}$$

$$\mathbb{P}(b \mid j = T, m = T) = \alpha(\mathbb{P}(b) \cdot (\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot (\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a \mid e = b_e, b) \cdot \dots)))$$

Enumeration: Example

► Variable order: b, e, a, j, m

$$\begin{aligned}
 \text{► } P_0 &:= P(b) \cdot \left[\begin{aligned} &P(e) \cdot \left[\begin{aligned} &P(a \mid b, e) \cdot \text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = T, e = T, a = T \rangle) \\ &P(\neg a \mid b, e) \cdot \text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = T, e = T, a = F \rangle) \end{aligned} \right. \\ &P(\neg e) \cdot \left[\begin{aligned} &P(a \mid b, \neg e) \cdot \text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = T, e = F, a = T \rangle) \\ &P(\neg a \mid b, \neg e) \cdot \text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = T, e = F, a = F \rangle) \end{aligned} \right] \end{aligned} \right] \\
 \text{► } P_1 &:= P(\neg b) \cdot \left[\begin{aligned} &P(e) \cdot \left[\begin{aligned} &P(a \mid \neg b, e) \cdot \text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = F, e = T, a = T \rangle) \\ &P(\neg a \mid \neg b, e) \cdot \text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = F, e = T, a = F \rangle) \end{aligned} \right. \\ &P(\neg e) \cdot \left[\begin{aligned} &P(a \mid \neg b, \neg e) \cdot \text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = F, e = F, a = T \rangle) \\ &P(\neg a \mid \neg b, \neg e) \cdot \text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = F, e = F, a = F \rangle) \end{aligned} \right] \end{aligned} \right] \\
 &\Leftarrow \alpha(\langle P_0, P_1 \rangle)
 \end{aligned}$$

$$\mathbb{P}(b \mid j = T, m = T) = \alpha(\mathbb{P}(b) \cdot \left(\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot \left(\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a \mid e = b_e, b) \cdot \dots \right) \right))$$

Enumeration: Example

► Variable order: b, e, a, j, m

$$\begin{aligned}
 \text{► } P_0 &:= P(b) \cdot \left[\begin{array}{l} P(e) \cdot \left[\begin{array}{l} P(a \mid b, e) \cdot \underbrace{\text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = T, e = T, a = T \rangle)}_{=P(j \mid a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = T, a = T \rangle)} \\ P(\neg a \mid b, e) \cdot \underbrace{\text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = T, e = T, a = F \rangle)}_{=P(j \mid \neg a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = T, a = F \rangle)} \end{array} \right] \\ + \\ P(\neg e) \cdot \left[\begin{array}{l} P(a \mid b, \neg e) \cdot \underbrace{\text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = T, e = F, a = T \rangle)}_{=P(j \mid a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = F, a = T \rangle)} \\ P(\neg a \mid b, \neg e) \cdot \underbrace{\text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = T, e = F, a = F \rangle)}_{=P(j \mid \neg a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = F, a = F \rangle)} \end{array} \right] \end{array} \right] \\
 \text{► } P_1 &:= P(\neg b) \cdot \left[\begin{array}{l} P(e) \cdot \left[\begin{array}{l} P(a \mid \neg b, e) \cdot \underbrace{\text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = F, e = T, a = T \rangle)}_{=P(j \mid a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = T, a = T \rangle)} \\ P(\neg a \mid \neg b, e) \cdot \underbrace{\text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = F, e = T, a = F \rangle)}_{=P(j \mid \neg a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = T, a = F \rangle)} \end{array} \right] \\ + \\ P(\neg e) \cdot \left[\begin{array}{l} P(a \mid \neg b, \neg e) \cdot \underbrace{\text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = F, e = F, a = T \rangle)}_{=P(j \mid a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = F, a = T \rangle)} \\ P(\neg a \mid \neg b, \neg e) \cdot \underbrace{\text{ENUMALL}(\langle j, m \rangle, \langle j = T, m = T, b = F, e = F, a = F \rangle)}_{=P(j \mid \neg a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = F, a = F \rangle)} \end{array} \right] \end{array} \right]
 \end{aligned}$$

Enumeration: Example

► Variable order: b, e, a, j, m

► $P_0 :=$

$$P(b) \cdot \left[\begin{array}{l} P(e) \cdot \left[\begin{array}{l} + P(a | b, e) \cdot P(j | a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = T, a = T \rangle) \\ + P(\neg a | b, e) \cdot P(j | \neg a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = T, a = F \rangle) \end{array} \right. \\ \left. + P(\neg e) \cdot \left[\begin{array}{l} + P(a | b, \neg e) \cdot P(j | a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = F, a = T \rangle) \\ + P(\neg a | b, \neg e) \cdot P(j | \neg a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = F, a = F \rangle) \end{array} \right] \right] \end{array} \right.$$

► $P_1 := P(\neg b) \cdot$

$$\left[\begin{array}{l} P(e) \cdot \left[\begin{array}{l} + P(a | \neg b, e) \cdot P(j | a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = T, a = T \rangle) \\ + P(\neg a | \neg b, e) \cdot P(j | \neg a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = T, a = F \rangle) \end{array} \right. \\ \left. + P(\neg e) \cdot \left[\begin{array}{l} + P(a | \neg b, \neg e) \cdot P(j | a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = F, a = T \rangle) \\ + P(\neg a | \neg b, \neg e) \cdot P(j | \neg a) \cdot \text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = F, a = F \rangle) \end{array} \right] \right] \end{array} \right.$$

$\Leftarrow \alpha(\langle P_0, P_1 \rangle)$

$$\mathbb{P}(b | j = T, m = T) = \alpha(\mathbb{P}(b) \cdot \left(\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot \left(\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a | e = b_e, b) \cdot P(j | a = b_a) \right) \right))$$

Enumeration: Example

► Variable order: b, e, a, j, m

► $P_0 :=$

$$\begin{aligned}
 P(b) \cdot & \left[\begin{aligned} & P(e) \cdot \left[\begin{aligned} & P(a \mid b, e) \cdot P(j \mid a) \cdot \underbrace{\text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = T, a = T \rangle)}_{=P(m \mid a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = T, a = T \rangle)} \\ & P(\neg a \mid b, e) \cdot P(j \mid \neg a) \cdot \underbrace{\text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = T, a = F \rangle)}_{=P(m \mid \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = T, a = F \rangle)} \end{aligned} \right] \\ & + \\ & P(\neg e) \cdot \left[\begin{aligned} & P(a \mid b, \neg e) \cdot P(j \mid a) \cdot \underbrace{\text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = F, a = T \rangle)}_{=P(m \mid a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = F, a = T \rangle)} \\ & P(\neg a \mid b, \neg e) \cdot P(j \mid \neg a) \cdot \underbrace{\text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = T, e = F, a = F \rangle)}_{=P(m \mid \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = F, a = F \rangle)} \end{aligned} \right] \end{aligned} \right]
 \end{aligned}$$

► $P_1 := P(\neg b) \cdot$

$$\begin{aligned}
 & \left[\begin{aligned} & P(e) \cdot \left[\begin{aligned} & P(a \mid \neg b, e) \cdot P(j \mid a) \cdot \underbrace{\text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = T, a = T \rangle)}_{=P(m \mid a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = T, a = T \rangle)} \\ & P(\neg a \mid \neg b, e) \cdot P(j \mid \neg a) \cdot \underbrace{\text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = T, a = F \rangle)}_{=P(m \mid \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = T, a = F \rangle)} \end{aligned} \right] \\ & + \\ & P(\neg e) \cdot \left[\begin{aligned} & P(a \mid \neg b, \neg e) \cdot P(j \mid a) \cdot \underbrace{\text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = F, a = T \rangle)}_{=P(m \mid a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = F, a = T \rangle)} \\ & P(\neg a \mid \neg b, \neg e) \cdot P(j \mid \neg a) \cdot \underbrace{\text{ENUMALL}(\langle m \rangle, \langle j = T, m = T, b = F, e = F, a = F \rangle)}_{=P(m \mid \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = F, a = F \rangle)} \end{aligned} \right] \end{aligned} \right]
 \end{aligned}$$

Enumeration: Example

► Variable order: b, e, a, j, m

► $P_0 := P(b) \cdot$

$$\left[\begin{array}{l} P(e) \cdot \left[\begin{array}{l} + P(a | b, e) \cdot P(j | a) \cdot P(m | a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = T, a = T \rangle) \\ + P(\neg a | b, e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = T, a = F \rangle) \end{array} \right. \\ \left. + P(\neg e) \cdot \left[\begin{array}{l} + P(a | b, \neg e) \cdot P(j | a) \cdot P(m | a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = F, a = T \rangle) \\ + P(\neg a | b, \neg e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = F, a = F \rangle) \end{array} \right] \right]$$

► $P_1 := P(\neg b) \cdot$

$$\left[\begin{array}{l} P(e) \cdot \left[\begin{array}{l} + P(a | \neg b, e) \cdot P(j | a) \cdot P(m | a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = T, a = T \rangle) \\ + P(\neg a | \neg b, e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = T, a = F \rangle) \end{array} \right. \\ \left. + P(\neg e) \cdot \left[\begin{array}{l} + P(a | \neg b, \neg e) \cdot P(j | a) \cdot P(m | a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = F, a = T \rangle) \\ + P(\neg a | \neg b, \neg e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = F, a = F \rangle) \end{array} \right] \right]$$

$\Leftarrow \alpha(\langle P_0, P_1 \rangle)$

$$\mathbb{P}(b | j = T, m = T) = \alpha(\mathbb{P}(b) \cdot \left(\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot \left(\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a | e = b_e, b) \cdot P(j | a = b_a) \cdot P(m | a = b_a) \right) \right))$$

Enumeration: Example

► Variable order: b, e, a, j, m

► $P_0 := P(b) \cdot$

$$\left[\begin{array}{l} P(e) \cdot \left[+ \begin{array}{l} P(a | b, e) \cdot P(j | a) \cdot P(m | a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = T, a = T \rangle) \\ P(\neg a | b, e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = T, a = F \rangle) \end{array} \right. \\ \left. + P(\neg e) \cdot \left[+ \begin{array}{l} P(a | b, \neg e) \cdot P(j | a) \cdot P(m | a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = F, a = T \rangle) \\ P(\neg a | b, \neg e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = T, e = F, a = F \rangle) \end{array} \right] \right] \end{array} \right.$$

► $P_1 := P(\neg b) \cdot$

$$\left[\begin{array}{l} P(e) \cdot \left[+ \begin{array}{l} P(a | \neg b, e) \cdot P(j | a) \cdot P(m | a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = T, a = T \rangle) \\ P(\neg a | \neg b, e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = T, a = F \rangle) \end{array} \right. \\ \left. + P(\neg e) \cdot \left[+ \begin{array}{l} P(a | \neg b, \neg e) \cdot P(j | a) \cdot P(m | a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = F, a = T \rangle) \\ P(\neg a | \neg b, \neg e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot \text{ENUMALL}(\langle \rangle, \langle j = T, m = T, b = F, e = F, a = F \rangle) \end{array} \right] \right] \end{array} \right.$$

$\Leftarrow \alpha(\langle P_0, P_1 \rangle)$

$$\mathbb{P}(b | j = T, m = T) = \alpha(\mathbb{P}(b) \cdot \left(\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot \left(\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a | e = b_e, b) \cdot P(j | a = b_a) \cdot P(m | a = b_a) \right) \right))$$

Enumeration: Example

► Variable order: b, e, a, j, m

$$\begin{aligned} \text{► } P_0 &:= P(b) \cdot \left[\begin{array}{l} P(e) \cdot \left[\begin{array}{l} P(a | b, e) \cdot P(j | a) \cdot P(m | a) \cdot 1.0 \\ P(\neg a | b, e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot 1.0 \end{array} \right] \\ + P(\neg e) \cdot \left[\begin{array}{l} P(a | b, \neg e) \cdot P(j | a) \cdot P(m | a) \cdot 1.0 \\ P(\neg a | b, \neg e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot 1.0 \end{array} \right] \end{array} \right] \\ \text{► } P_1 &:= P(\neg b) \cdot \left[\begin{array}{l} P(e) \cdot \left[\begin{array}{l} P(a | \neg b, e) \cdot P(j | a) \cdot P(m | a) \cdot 1.0 \\ P(\neg a | \neg b, e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot 1.0 \end{array} \right] \\ + P(\neg e) \cdot \left[\begin{array}{l} P(a | \neg b, \neg e) \cdot P(j | a) \cdot P(m | a) \cdot 1.0 \\ P(\neg a | \neg b, \neg e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot 1.0 \end{array} \right] \end{array} \right] \\ &\Leftarrow \alpha(\langle P_0, P_1 \rangle) \end{aligned}$$

$$\mathbb{P}(b|j = T, m = T) = \alpha(\mathbb{P}(b) \cdot (\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot (\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a | e = b_e, b) \cdot P(j | a = b_a) \cdot P(m | a = b_a))))$$

Enumeration: Example

► Variable order: b, e, a, j, m

$$\begin{aligned}
 \text{► } P_0 &:= P(b) \cdot \left[\begin{array}{l} P(e) \cdot \left[\begin{array}{l} P(a | b, e) \cdot P(j | a) \cdot P(m | a) \cdot 1.0 \\ P(\neg a | b, e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot 1.0 \end{array} \right] \\ + \\ P(\neg e) \cdot \left[\begin{array}{l} P(a | b, \neg e) \cdot P(j | a) \cdot P(m | a) \cdot 1.0 \\ P(\neg a | b, \neg e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot 1.0 \end{array} \right] \end{array} \right] \\
 \text{► } P_1 &:= P(\neg b) \cdot \left[\begin{array}{l} P(e) \cdot \left[\begin{array}{l} P(a | \neg b, e) \cdot P(j | a) \cdot P(m | a) \cdot 1.0 \\ P(\neg a | \neg b, e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot 1.0 \end{array} \right] \\ + \\ P(\neg e) \cdot \left[\begin{array}{l} P(a | \neg b, \neg e) \cdot P(j | a) \cdot P(m | a) \cdot 1.0 \\ P(\neg a | \neg b, \neg e) \cdot P(j | \neg a) \cdot P(m | \neg a) \cdot 1.0 \end{array} \right] \end{array} \right] \\
 \Leftarrow & \underbrace{\alpha(\langle P_0, P_1 \rangle)}_{= \langle \frac{P_0}{P_0 + P_1}, \frac{P_1}{P_0 + P_1} \rangle}
 \end{aligned}$$

$$\mathbb{P}(b|j = T, m = T) = \alpha(\mathbb{P}(b) \cdot (\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot (\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a | e = b_e, b) \cdot P(j | a = b_a) \cdot P(m | a = b_a))))$$

Enumeration: Example

► Variable order: b, e, a, j, m

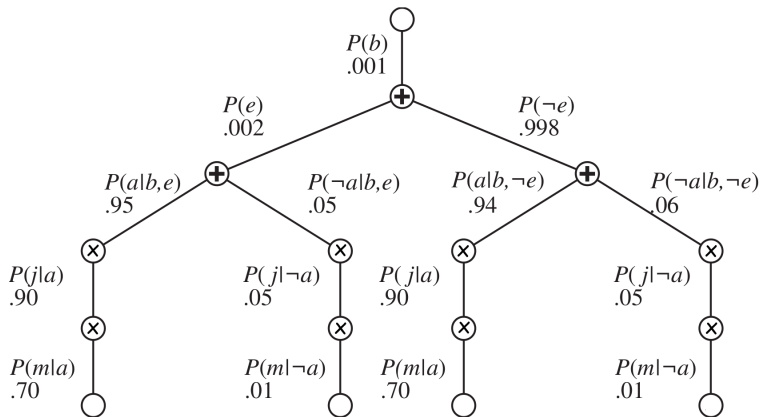
$$\begin{aligned}
 \text{► } P_0 &:= P(b) \cdot \left[\begin{aligned} &P(e) \cdot \left[\begin{aligned} &P(a \mid b, e) \cdot P(j \mid a) \cdot P(m \mid a) \cdot 1.0 \\ &P(\neg a \mid b, e) \cdot P(j \mid \neg a) \cdot P(m \mid \neg a) \cdot 1.0 \end{aligned} \right. \\ &+ P(\neg e) \cdot \left[\begin{aligned} &P(a \mid b, \neg e) \cdot P(j \mid a) \cdot P(m \mid a) \cdot 1.0 \\ &P(\neg a \mid b, \neg e) \cdot P(j \mid \neg a) \cdot P(m \mid \neg a) \cdot 1.0 \end{aligned} \right. \end{aligned} \right] \\
 \text{► } P_1 &:= P(\neg b) \cdot \left[\begin{aligned} &P(e) \cdot \left[\begin{aligned} &P(a \mid \neg b, e) \cdot P(j \mid a) \cdot P(m \mid a) \cdot 1.0 \\ &P(\neg a \mid \neg b, e) \cdot P(j \mid \neg a) \cdot P(m \mid \neg a) \cdot 1.0 \end{aligned} \right. \\ &+ P(\neg e) \cdot \left[\begin{aligned} &P(a \mid \neg b, \neg e) \cdot P(j \mid a) \cdot P(m \mid a) \cdot 1.0 \\ &P(\neg a \mid \neg b, \neg e) \cdot P(j \mid \neg a) \cdot P(m \mid \neg a) \cdot 1.0 \end{aligned} \right. \end{aligned} \right] \\
 \Leftarrow &\left\langle \frac{P_0}{P_0 + P_1}, \frac{P_1}{P_0 + P_1} \right\rangle
 \end{aligned}$$

$$\mathbb{P}(b \mid j = T, m = T) = \alpha(\mathbb{P}(b) \cdot \left(\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot \left(\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a \mid e = b_e, b) \cdot P(j \mid a = b_a) \cdot P(m \mid a = b_a) \right) \right))$$

The Evaluation of $P(b \mid j, m)$ as a “Search Tree”

$$\mathbb{P}(b \mid j, m) = \alpha(\mathbb{P}(b) \cdot (\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot (\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a \mid e = b_e, b) \cdot P(j \mid a = b_a) \cdot P(m \mid a = b_a))))$$

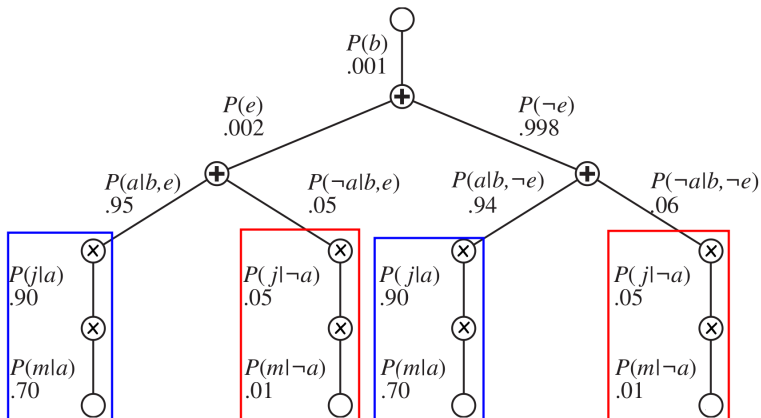
Note: ENUMERATE-QUERY corresponds to depth-first traversal of an arithmetic expression-tree:



The Evaluation of $P(b \mid j, m)$ as a “Search Tree”

$$\mathbb{P}(b \mid j, m) = \alpha(\mathbb{P}(b) \cdot (\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot (\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a \mid e = b_e, b) \cdot P(j \mid a = b_a) \cdot P(m \mid a = b_a))))$$

Note: ENUMERATE-QUERY corresponds to depth-first traversal of an arithmetic expression-tree:



Variable Elimination 2



$$\mathbb{P}(b|j, m) = \alpha(\mathbb{P}(b) \cdot \left(\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot \left(\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a | e = b_e, b) \cdot P(j | a = b_a) \cdot P(m | a = b_a) \right) \right))$$

The last two factors $P(j | a = b_a)$, $P(m | a = b_a)$ only depend on a , but are “trapped” behind the summation over e , hence computed twice in two distinct recursive calls to `ENUMALL`

- **Idea:** Instead of left-to-right (top-down DFS), operate right-to-left (bottom-up) and store intermediate “factors” along with their “dependencies”:

$$\underbrace{\underbrace{\underbrace{\underbrace{\underbrace{\underbrace{\underbrace{\underbrace{\alpha(\mathbb{P}(b))}_{f_7(b)}} \cdot \left(\sum_{b_e \in \{T, F\}} \underbrace{P(e = b_e)}_{f_5(e)} \cdot \left(\sum_{b_a \in \{T, F\}} \underbrace{\mathbb{P}(a = b_a | e = b_e, b)}_{f_3(a, b, e)} \cdot \underbrace{P(j | a = b_a)}_{f_2(a)} \cdot \underbrace{P(m | a = b_a)}_{f_1(a)} \right)}_{f_4(b, e)} \right)}_{f_6(b)}}}_{f_6(b)}$$

Variable Elimination: Example

- ▶ We only show variable elimination by example: (implementation details get tricky, but the idea is simple)

$$\mathbb{P}(b) \cdot \left(\sum_{b_e \in \{T, F\}} P(e = b_e) \cdot \left(\sum_{b_a \in \{T, F\}} \mathbb{P}(a = b_a | e = b_e, b) \cdot P(j | a = b_a) \cdot P(m | a = b_a) \right) \right)$$

- ▶ Assume reverse topological order of variables: m, j, a, e, b
 - ▶ m is an **evidence variable** with value T and dependency a , which is a **hidden variable**. We introduce a new “factor” $f(a) := f_1(a) := \langle P(m | a), P(m | \neg a) \rangle$.
 - ▶ j works analogously, $f_2(a) := \langle P(j | a), P(j | \neg a) \rangle$. We “multiply” with the existing factor, yielding $f(a) := \langle f_1(a) \cdot f_2(a), f_1(\neg a) \cdot f_2(\neg a) \rangle = \langle P(m | a) \cdot P(j | a), P(m | \neg a) \cdot P(j | \neg a) \rangle$
 - ▶ a is a **hidden variable** with dependencies e (**hidden**) and b (**query**).
 1. We introduce a new “factor” $f_3(a, e, b)$, a $2 \times 2 \times 2$ table with the relevant **conditional probabilities** $\mathbb{P}(a | e, b)$.
 2. We multiply each entry of f_3 with the relevant entries of the existing factor f , yielding $f(a, e, b)$.
 3. We “sum out” the resulting factor over a , yielding a new factor $f(e, b) = f(a, e, b) + f(\neg a, e, b)$.
 - ▶ ...

- ▶ \leadsto can speed things up by a factor of 1000! (or more, depending on the order of variables!)

The Complexity of Exact Inference

- ▶ **Definition 3.4.** A graph G is called **singly connected**, or a **polytree** (otherwise **multiply connected**), if there is at most one **undirected path** between any two **nodes** in G .
- ▶ **Theorem 3.5 (Good News).** *On **singly connected Bayesian networks**, **variable elimination** runs in **polynomial time**.*

The Complexity of Exact Inference

- ▶ **Definition 3.8.** A **graph** G is called **singly connected**, or a **polytree** (otherwise **multiply connected**), if there is at most one **undirected path** between any two **nodes** in G .
- ▶ **Theorem 3.9 (Good News).** *On **singly connected Bayesian networks**, **variable elimination** runs in **polynomial time**.*
- ▶ Is our **BN** for Mary & John a **polytree**? (Yes.)

The Complexity of Exact Inference

- ▶ **Definition 3.12.** A graph G is called **singly connected**, or a **polytree** (otherwise **multiply connected**), if there is at most one **undirected path** between any two **nodes** in G .
- ▶ **Theorem 3.13 (Good News).** On *singly connected Bayesian networks*, *variable elimination* runs in *polynomial time*.
- ▶ Is our BN for Mary & John a **polytree**? (Yes.)
- ▶ **Theorem 3.14 (Bad News).** For *multiply connected Bayesian networks*, *probabilistic inference* is $\#P$ -hard. ($\#P$ is harder than NP, i.e. $NP \subseteq \#P$)
- ▶ **So?:** Life goes on ... In the hard cases, if need be we can throw exactitude to the winds and approximate.
- ▶ **Example 3.15.** Sampling techniques as in **MCTS**.

23.4 Conclusion

- ▶ **Bayesian networks (BN)** are a wide-spread tool to model **uncertainty**, and to reason about it. A BN represents **conditional independence** relations between **random variables**. It consists of a graph encoding the variable dependencies, and of **conditional probability tables (CPTs)**.
- ▶ Given a **variable ordering**, the BN is small if every variable depends on only a few of its predecessors.
- ▶ **Probabilistic inference** requires to compute the **probability distribution** of a set of **query variables**, given a set of **evidence variables** whose values we know. The remaining variables are **hidden**.
- ▶ **Inference by enumeration** takes a BN as input, then applies **Normalization+Marginalization**, the **chain rule**, and exploits **conditional independence**. This can be viewed as a tree search that branches over all values of the hidden variables.
- ▶ **Variable elimination** avoids unnecessary computation. It runs in polynomial time for poly-tree BNs. In general, exact probabilistic inference is **#P-hard**. Approximate probabilistic inference methods exist.

Topics We Didn't Cover Here

- **Inference by sampling:** A whole zoo of methods for doing this exists.

Topics We Didn't Cover Here

- ▶ **Inference by sampling**: A whole zoo of methods for doing this exists.
- ▶ **Clustering**: Pre-combining subsets of variables to reduce the **running time** of inference.

Topics We Didn't Cover Here

- ▶ **Inference by sampling**: A whole zoo of methods for doing this exists.
- ▶ **Clustering**: Pre-combining subsets of variables to reduce the **running time** of inference.
- ▶ **Compilation to SAT**: More precisely, to “weighted model counting” in **CNF** formulas. Model counting extends DPLL with the ability to determine the number of satisfying interpretations. Weighted model counting allows to define a mass for each such interpretation (= the probability of an **atomic event**).

Topics We Didn't Cover Here

- ▶ **Inference by sampling**: A whole zoo of methods for doing this exists.
- ▶ **Clustering**: Pre-combining subsets of variables to reduce the **running time** of inference.
- ▶ **Compilation to SAT**: More precisely, to “weighted model counting” in **CNF** formulas. Model counting extends DPLL with the ability to determine the number of satisfying interpretations. Weighted model counting allows to define a mass for each such interpretation (= the probability of an **atomic event**).
- ▶ **Dynamic BN**: **BN** with one slice of variables at each “time step”, encoding probabilistic behavior over time.

Topics We Didn't Cover Here

- ▶ **Inference by sampling**: A whole zoo of methods for doing this exists.
- ▶ **Clustering**: Pre-combining subsets of variables to reduce the **running time** of inference.
- ▶ **Compilation to SAT**: More precisely, to “weighted model counting” in **CNF** formulas. Model counting extends DPLL with the ability to determine the number of satisfying interpretations. Weighted model counting allows to define a mass for each such interpretation (= the probability of an **atomic event**).
- ▶ **Dynamic BN**: **BN** with one slice of variables at each “time step”, encoding probabilistic behavior over time.
- ▶ **Relational BN**: **BN** with predicates and object variables.

Topics We Didn't Cover Here

- ▶ **Inference by sampling**: A whole zoo of methods for doing this exists.
- ▶ **Clustering**: Pre-combining subsets of variables to reduce the **running time** of inference.
- ▶ **Compilation to SAT**: More precisely, to “weighted model counting” in **CNF** formulas. Model counting extends DPLL with the ability to determine the number of satisfying interpretations. Weighted model counting allows to define a mass for each such interpretation (= the probability of an **atomic event**).
- ▶ **Dynamic BN**: **BN** with one slice of variables at each “time step”, encoding probabilistic behavior over time.
- ▶ **Relational BN**: **BN** with predicates and object variables.
- ▶ **First-order BN**: Relational **BN** with quantification, i.e. probabilistic logic. E.g., the BLOG language developed by Stuart Russel and co-workers.

Chapter 24

Making Simple Decisions Rationally

24.1 Introduction

Overview

- ▶ We now know how to update our **world model**, represented as (a set of) **random variables**, given observations. Now we need to *act*.

Overview

- ▶ We now know how to update our **world model**, represented as (a set of) **random variables**, given observations. Now we need to *act*.
- ▶ For that we need to answer two questions:
- ▶ **Questions:**
 - ▶ Given a **world model** and a set of *actions*, what will the likely consequences of each action be?
 - ▶ How “good” are these consequences?

- ▶ We now know how to update our **world model**, represented as (a set of) **random variables**, given observations. Now we need to *act*.
- ▶ For that we need to answer two questions:
- ▶ **Questions:**
 - ▶ Given a **world model** and a set of *actions*, what will the likely consequences of each action be?
 - ▶ How “good” are these consequences?
- ▶ **Idea:**
 - ▶ Represent actions as “special **random variables**”:
Given disjoint actions a_1, \dots, a_n , introduce a **random variable** A with **domain** $\{a_1, \dots, a_n\}$. Then we can model/query $\mathbb{P}(X|A = a_i)$.
 - ▶ Assign *numerical values* to the possible outcomes of actions (i.e. a function $u: \text{dom}(X) \rightarrow \mathbb{R}$) indicating their desirability.
 - ▶ Choose the action that maximizes the *expected value* of u

- ▶ We now know how to update our **world model**, represented as (a set of) **random variables**, given observations. Now we need to *act*.
- ▶ For that we need to answer two questions:
- ▶ **Questions:**
 - ▶ Given a **world model** and a set of *actions*, what will the likely consequences of each action be?
 - ▶ How “good” are these consequences?
- ▶ **Idea:**
 - ▶ Represent actions as “special **random variables**”:
Given disjoint actions a_1, \dots, a_n , introduce a **random variable** A with **domain** $\{a_1, \dots, a_n\}$. Then we can model/query $\mathbb{P}(X|A = a_i)$.
 - ▶ Assign *numerical values* to the possible outcomes of actions (i.e. a function $u: \text{dom}(X) \rightarrow \mathbb{R}$) indicating their desirability.
 - ▶ Choose the action that maximizes the *expected value* of u

Definition 1.4. **Decision theory** investigates **decision problems**, i.e. how a **utility-based agent** a deals with choosing among **actions** based on the desirability of their outcomes given by a real-valued **utility function** U on **states** $s \in S$: i.e. $U: S \rightarrow \mathbb{R}$.

- ▶ If our states are random variables, then we obtain a random variable for the utility function:
- ▶ **Observation:** Let $X_i: \Omega \rightarrow D_i$ random variables on a probability model $\langle \Omega, P \rangle$ and $f: D_1 \times \dots \times D_n \rightarrow E$. Then $F(x) := f(X_0(x), \dots, X_n(x))$ is a random variable $\Omega \rightarrow E$.

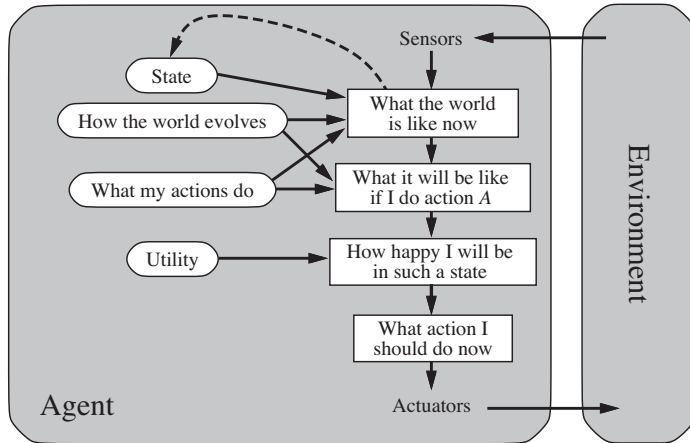
- ▶ If our states are random variables, then we obtain a random variable for the utility function:
- ▶ **Observation:** Let $X_i: \Omega \rightarrow D_i$ random variables on a probability model $\langle \Omega, P \rangle$ and $f: D_1 \times \dots \times D_n \rightarrow E$. Then $F(x) := f(X_0(x), \dots, X_n(x))$ is a random variable $\Omega \rightarrow E$.
- ▶ **Definition 1.7.** Given a probability model $\langle \Omega, P \rangle$ and a random variable $X: \Omega \rightarrow D$ with $D \subseteq \mathbb{R}$, then $E(X) := \sum_{x \in D} P(X = x) \cdot x$ is called the expected value (or expectation) of X . (Assuming the sum/series is actually defined!)
Analogously, let e_1, \dots, e_n a sequence of events. Then the expected value of X given e_1, \dots, e_n is defined as $E(X|e_1, \dots, e_n) := \sum_{x \in D} \mu(X = x | e_1, \dots, e_n) \cdot x$.

- ▶ If our **states** are **random variables**, then we obtain a **random variable** for the **utility function**:
- ▶ **Observation:** Let $X_i: \Omega \rightarrow D_i$ **random variables** on a **probability model** $\langle \Omega, P \rangle$ and $f: D_1 \times \dots \times D_n \rightarrow E$. Then $F(x) := f(X_0(x), \dots, X_n(x))$ is a **random variable** $\Omega \rightarrow E$.
- ▶ **Definition 1.9.** Given a **probability model** $\langle \Omega, P \rangle$ and a **random variable** $X: \Omega \rightarrow D$ with $D \subseteq \mathbb{R}$, then $E(X) := \sum_{x \in D} P(X = x) \cdot x$ is called the **expected value** (or **expectation**) of X . (Assuming the sum/series is actually defined!)
Analogously, let e_1, \dots, e_n a sequence of **events**. Then the **expected value** of X **given** e_1, \dots, e_n is defined as $E(X|e_1, \dots, e_n) := \sum_{x \in D} \mu(X = x | e_1, \dots, e_n) \cdot x$.
- ▶ Putting things together:
- ▶ **Definition 1.10.** Let $A: \Omega \rightarrow D$ a **random variable** (where D is a set of **actions**) $X_i: \Omega \rightarrow D_i$ **random variables** (the **state**), and $U: D_1 \times \dots \times D_n \rightarrow \mathbb{R}$ a **utility function**. Then the **expected utility** of the **action** $a \in D$ is the **expected value** of U (interpreted as a **random variable**) given $A = a$; i.e.

$$EU(a) := \sum_{\langle x_1, \dots, x_n \rangle \in D_1 \times \dots \times D_n} \mu(X_1 = x_1, \dots, X_n = x_n | A = a) \cdot U(x_1, \dots, x_n)$$

Utility-based Agents

- **Definition 1.11.** A **utility-based agent** uses a **world model** along with a **utility function** that models its preferences among the **states** of that world. It chooses the **action** that leads to the best **expected utility**.
- **Agent Schema:**



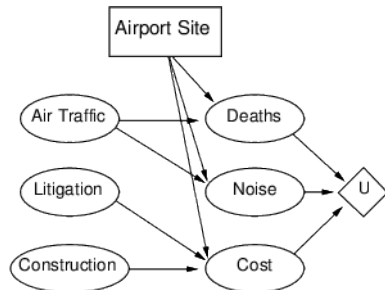
Maximizing Expected Utility (Ideas)

- ▶ **Definition 1.12 (MEU principle for Rationality).** We call an **action rational** if it **maximizes expected utility (MEU)**. An **utility-based agent** is called **rational**, iff it always chooses a **rational action**.
(in principle)
- ▶ **Hooray:** This solves all of AI.
- ▶ **Problem:** There is a long, long way towards an operationalization ;)
- ▶ **Note:** An **agent** can be entirely **rational** (consistent with **MEU**) without ever representing or manipulating **utilities** and probabilities.
- ▶ **Example 1.13.** A **reflex agent** for tic tac toe based on a perfect **lookup table** is **rational** if we take (the negative of) “winning/drawing in n steps” as the **utility function**.
- ▶ **Example 1.14 (AI1).** **Heuristics** in **tree search** (**greedy search**, A^*) and game-play (minimax, alpha-beta pruning) maximize “expected” utility.
⇒ In fully observable, deterministic environments, “expected utility” reduces to a specific determined utility value:
$$EU(a) = U(T(S(s, e), a))$$
, where e the most recent **percept**, s the current **state**, S the sensor function and T the transition function.
- ▶ Now let's figure out how to actually assign **utilities**!

24.2 Decision Networks

Definition 2.1. A **decision network** is a **Bayesian network** with two additional kinds of **nodes**:

- ▶ **action nodes**, representing a set of possible **actions**, and (**square nodes**)
- ▶ A single **utility node** (also called **value node**). (**diamond node**)



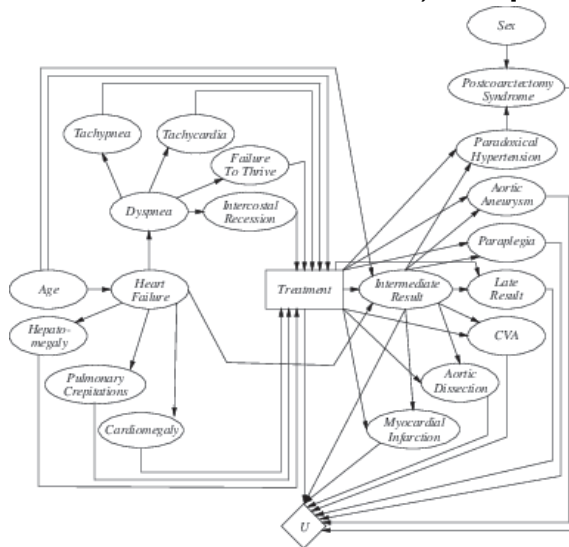
- ▶ **General Algorithm:** Given evidence $E_j = e_j$, and **action nodes** A_1, \dots, A_k , compute the expected utility of each action, given the evidence, i.e. return the sequence of actions

$$\operatorname{argmax}_{a_1, \dots, a_k} \underbrace{\sum_{\langle x_1, \dots, x_n \rangle} \underbrace{\mu(X_i = x_i \mid A_1 = a_1, \dots, A_k = a_k, E_j = e_j)}_{\text{usual Bayesian Network inference}} \cdot U(X_i = x_i)}_{\text{=expected utility of } a_1, \dots, a_k}$$

- ▶ **Note** the sheer amount of summands in the sum above in the general case! (\Rightarrow We will simplify where possible later)

Decision Networks: Example

► Example 2.2 (A Decision-Network for Aortic Coarctation). from [Lucas:kadtes96]



24.3 Preferences and Utilities

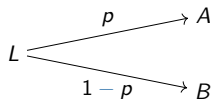
Preferences in Deterministic Environments

- ▶ **Problem:** How do we determine the **utility** of a **state**? (We cannot directly measure our satisfaction/happiness in a possibly future state...)
(What unit would we even use?)
 - ▶ **Example 3.1.** I have to decide whether to go to class today (or sleep in). What is the **utility** of this lecture? (obviously 42)
 - ▶ **Idea:** We can let people/**agents** choose between two **states** (**subjective preference**) and derive a **utility** from these choices.
 - ▶ **Example 3.2.** “*Give me your cell-phone or I will give you a bloody nose*”. \leadsto
To make a decision in a **deterministic environment**, the **agent** must determine whether it **prefers** a **state** without phone to one with a bloody nose?
 - ▶ **Definition 3.3.** Given **states** A and B (we call them **prizes**) an **agent** can express **preferences** of the form
 - ▶ $A \succ B$ A **preferred** over B
 - ▶ $A \sim B$ **indifference** between A and B
 - ▶ $A \succeq B$ B not **preferred** over A
- i.e. Given a **set** \mathcal{S} (of **states**), we define binary relations \succ and \sim on \mathcal{S} .

Preferences in Non-Deterministic Environments

- ▶ **Problem:** In **nondeterministic environments** we do not have full information about the **states** we choose between.
- ▶ **Example 3.4 (Airline Food).** “*Do you want chicken or pasta*” (but we cannot see through the tin foil)
- ▶ **Definition 3.5.**

Let \mathcal{S} a set of **states**. We call a **random variable** X with **domain** $\{A_1, \dots, A_n\} \subseteq \mathcal{S}$ a **lottery** and write $[p_1, A_1; \dots; p_n, A_n]$, where $p_i = P(X = A_i)$.



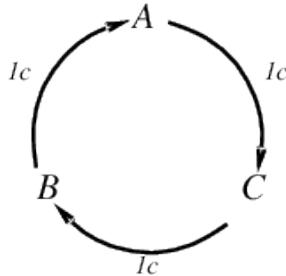
- ▶ **Idea:** A **lottery** represents the result of a **nondeterministic action** that can have **outcomes** A_i with **prior probability** p_i . For the binary case, we use $[p, A; 1-p, B]$. We can then extend **preferences** to include **lotteries**, as a measure of how *strongly* we **prefer** one **prize** over another.
- ▶ **Convention:** We assume \mathcal{S} to be *closed under lotteries*, i.e. **lotteries** themselves are also **states**. That allows us to consider **lotteries** such as $[p, A; 1-p, [q, B; 1-q, C]]$.

- **Note:** Preferences of a rational agent must obey certain constraints – An agent with rational preferences can be described as an MEU-agent.
- **Definition 3.6.** We call a set \succsim of preferences rational, iff the following constraints hold:

Orderability	$A \succsim B \vee B \succsim A \vee A \sim B$
Transitivity	$A \succsim B \wedge B \succsim C \Rightarrow A \succsim C$
Continuity	$A \succ B \succ C \Rightarrow (\exists p. [p, A; 1-p, C] \sim B)$
Substitutability	$A \sim B \Rightarrow [p, A; 1-p, C] \sim [p, B; 1-p, C]$
Monotonicity	$A \succ B \Rightarrow ((p > q) \Leftrightarrow [p, A; 1-p, B] \succ [q, A; 1-q, B])$
Decomposability	$[p, A; 1-p, [q, B; 1-q, C]] \sim [p, A; ((1-p)q), B; ((1-p)(1-q)), C]$
- From a set of rational preferences, we can obtain a meaningful utility function.

Rational preferences contd.

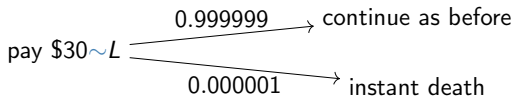
- ▶ Violating the rationality constraints from ??? leads to self-evident **irrationality**.
- ▶ **Example 3.7.** An **agent** with **intransitive preferences** can be induced to give away all its money:
 - ▶ If $B \succ C$, then an **agent** who has C would pay (say) 1 cent to get B
 - ▶ If $A \succ B$, then an **agent** who has B would pay (say) 1 cent to get A
 - ▶ If $C \succ A$, then an **agent** who has A would pay (say) 1 cent to get C



24.4 Utilities

- ▶ **Theorem 4.1.** (Ramsey, 1931; von Neumann and Morgenstern, 1944)
Given a *rational* set of *preferences* there exists a real valued *function* U such that $U(A) \geq U(B)$, iff $A \succeq B$ and $U([p_1, S_1; \dots; p_n, S_n]) = \sum_i p_i U(S_i)$
- ▶ This is an existence theorem, uniqueness not guaranteed.
- ▶ **Note:** Agent behavior is *invariant* w.r.t. *positive linear transformations*, i.e. an agent with utility function $U'(x) = k_1 U(x) + k_2$ where $k_1 > 0$ behaves exactly like one with U .
- ▶ **Observation:** With deterministic *prizes* only (no *lottery* choices), only a *total ordering* on *prizes* can be determined.
- ▶ **Definition 4.2.** We call a *total ordering* on *states* a *value function* or *ordinal utility function*. (If we don't need to care about *relative* utilities of states, e.g. to compute non-trivial expected utilities, that's all we need anyway!)

- ▶ **Intuition:** Utilities map states to real numbers.
- ▶ **Question:** Which numbers exactly?
- ▶ **Definition 4.3 (Standard approach to assessment of human utilities).** Compare a given state A to a standard lottery L_p that has
 - ▶ “best possible prize” u_{\top} with probability p
 - ▶ “worst possible catastrophe” u_{\perp} with probability $1 - p$adjust lottery probability p until $A \sim L_p$. Then $U(A) = p$.
- ▶ **Example 4.4.** Choose $u_{\top} \hat{=}$ current state, $u_{\perp} \hat{=}$ instant death



- **Definition 4.5.** Normalized utilities: $u_{\top} = 1$, $u_{\perp} = 0$.
(Not very meaningful, but at least it's independent of the specific problem...)

- ▶ **Definition 4.8.** Normalized utilities: $u_{\top} = 1$, $u_{\perp} = 0$.
(Not very meaningful, but at least it's independent of the specific problem...)
- ▶ **Obviously:** Money function (see later) (Very intuitive, often easy to determine, but actually not well-suited as a utility function)

- ▶ **Definition 4.11. Normalized utilities:** $u_{\top} = 1$, $u_{\perp} = 0$.
(Not very meaningful, but at least it's independent of the specific problem...)
- ▶ **Obviously:** *Money* (Very intuitive, often easy to determine, but actually not well-suited as a utility function (see later))
- ▶ **Definition 4.12. Micromorts:** one millionth chance of instant death.
(useful for Russian roulette, paying to reduce product risks, etc.)
- ▶ **But:** Not necessarily a good measure of risk, if the risk is “merely” severe injury or illness. . .

- ▶ **Definition 4.14. Normalized utilities:** $u_{\top} = 1$, $u_{\perp} = 0$.
(Not very meaningful, but at least it's independent of the specific problem...)
- ▶ **Obviously:** *Money* (Very intuitive, often easy to determine, but actually not well-suited as a utility function (see later))
- ▶ **Definition 4.15. Micromorts:** one millionth chance of instant death.
(useful for Russian roulette, paying to reduce product risks, etc.)
- ▶ **But:** Not necessarily a good measure of risk, if the risk is “merely” severe injury or illness. . .
- ▶ The following measure is better (more informative)
- ▶ **Definition 4.16. QALYs: quality adjusted life years**
QALYs are useful for medical decisions involving substantial risk.

- **Problem:** What is the monetary value of a micromort?

Comparing Utilities

- ▶ **Problem:** What is the monetary value of a **micromort**?
- ▶ **Just ask people:** What would you pay to avoid playing Russian roulette with a million-barrelled revolver?
(Usually: quite a lot!)

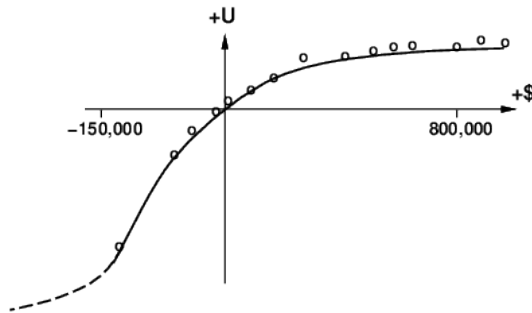
- ▶ **Problem:** What is the monetary value of a micromort?
- ▶ **Just ask people:** What would you pay to avoid playing Russian roulette with a million-barrelled revolver? (Usually: quite a lot!)
- ▶ **But their behavior suggests a lower price:**
 - ▶ Driving in a car for 370km incurs a risk of one micromort;
 - ▶ Over the life of your car – say, 150,000km that's 400 micromorts.
 - ▶ People appear to be willing to pay about 10,000€ more for a safer car that halves the risk of death. (≈ 25€ per micromort)

This figure has been confirmed across many individuals and risk types.

- ▶ **Problem:** What is the monetary value of a micromort?
 - ▶ **Just ask people:** What would you pay to avoid playing Russian roulette with a million-barrelled revolver? (Usually: quite a lot!)
 - ▶ **But their behavior suggests a lower price:**
 - ▶ Driving in a car for 370km incurs a risk of one micromort;
 - ▶ Over the life of your car – say, 150,000km that's 400 micromorts.
 - ▶ People appear to be willing to pay about 10,000€ more for a safer car that halves the risk of death. (≈ 25€ per micromort)
- This figure has been confirmed across many individuals and risk types.
- ▶ Of course, this argument holds only for small risks. Most people won't agree to kill themselves for 25M€. (Also: People are pretty bad at estimating and comparing risks, especially if they are small.) (Various cognitive biases and heuristics are at work here!)

Money vs. Utility

- ▶ Money does *not* behave as a **utility function** should.
- ▶ Given a **lottery** L with **expected monetary value** $EMV(L)$, usually $U(L) < U(EMV(L))$, i.e., people are **risk averse**.
- ▶ **Utility curve:** For what probability p am I indifferent between a prize x and a lottery $[p, M\$; 1-p, 0\$]$ for large numbers M ?
- ▶ Typical empirical data, extrapolated with **risk prone** behavior for debtors:



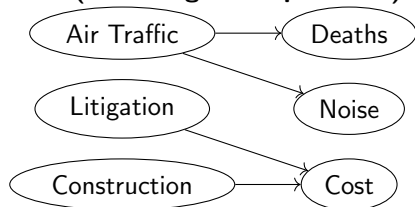
- ▶ **Empirically:** Comes close to the **logarithm** on the **natural numbers**.

24.5 Multi-Attribute Utility

Utility Functions on Attributes

- ▶ **Recap:** So far we understand how to obtain **utility functions** $u: S \rightarrow \mathbb{R}$ on **states** $s \in S$ from (**rational**) **preferences**.
- ▶ **But** in practice, our actions often impact *multiple* distinct “**attributes**” that need to be weighed against each other.
⇒ **Lotteries** become complex very quickly
- ▶ **Definition 5.1.** Let X_1, \dots, X_n be **random variables** with **domains** D_1, \dots, D_n . Then we call a function $u: D_1 \times \dots \times D_n \rightarrow \mathbb{R}$ a (**multi-attribute**) **utility function** on **attributes** X_1, \dots, X_n .
- ▶ **Note:** In the general (worst) case, a **multi-attribute utility function** on n **random variables** with **domain** sizes k each requires k^n parameters to represent.
- ▶ **But:** A **utility** function on multiple **attributes** often has “internal structure” that we can exploit to simplify things.
For example, the distinct **attributes** are often “independent” with respect to their utility (a higher-quality product is better than a lower-quality one that costs the same, and a cheaper product is better than an expensive one of the same quality)

► Example 5.2 (Assessing an Airport Site).

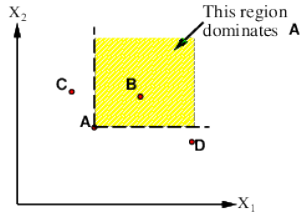


- **Attributes:** Deaths, Noise, Cost.
- **Question:** What is $U(\text{Deaths, Noise, Cost})$ for a projected airport?

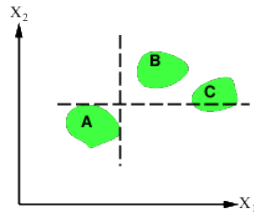
- How can complex **utility function** be assessed from **preference** behaviour?
- **Idea 1:** Identify conditions under which decisions can be made without complete identification of $U(X_1, \dots, X_n)$.
- **Idea 2:** Identify various types of *independence* in **preferences** and derive consequent canonical forms for $U(X_1, \dots, X_n)$.

Strict Dominance

- **First Assumption:** U is often *monotone* in each argument. (wlog. growing)
- **Definition 5.3.** (Informally) An action B *strictly dominates* an action A , iff every possible outcome of B is at least as good as every possible outcome of A ,



Deterministic attributes

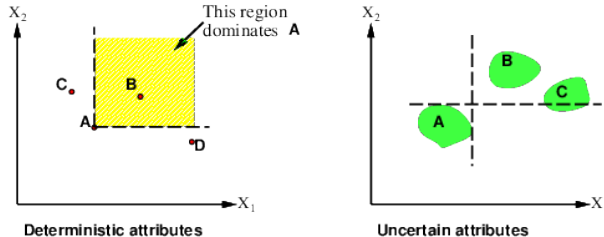


Uncertain attributes

- If A strictly dominates B , we can just ignore B entirely.

Strict Dominance

- **First Assumption:** U is often *monotone* in each argument. (wlog. growing)
- **Definition 5.4.** (Informally) An action B *strictly dominates* an action A , iff every possible outcome of B is at least as good as every possible outcome of A ,



- If A strictly dominates B , we can just ignore B entirely.
- **Observation:** *Strict dominance* seldom holds in practice (life is difficult) but is useful for narrowing down the field of contenders.

- ▶ **Definition 5.5.** Let X_1, X_2 distributions with domains $\subseteq \mathbb{R}$.
 X_1 **stochastically dominates** X_2 iff for all $t \in \mathbb{R}$, we have $P(X_1 \geq t) \geq P(X_2 \geq t)$, and for some t , we have $P(X_1 \geq t) > P(X_2 \geq t)$.
- ▶ **Observation 5.6.** If U is **monotone** in X_1 , and $\mathbb{P}(X_1|a)$ **stochastically dominates** $\mathbb{P}(X_1|b)$ for actions a, b , then a is always the better choice than b , with all other attributes X_i being equal.
 \Rightarrow If some action $\mathbb{P}(X_i|a)$ **stochastically dominates** $\mathbb{P}(X_i|b)$ for all **attributes** X_i , we can ignore b .

- ▶ **Definition 5.8.** Let X_1, X_2 distributions with domains $\subseteq \mathbb{R}$.
 X_1 **stochastically dominates** X_2 iff for all $t \in \mathbb{R}$, we have $P(X_1 \geq t) \geq P(X_2 \geq t)$, and for some t , we have $P(X_1 \geq t) > P(X_2 \geq t)$.
- ▶ **Observation 5.9.** If U is *monotone* in X_1 , and $\mathbb{P}(X_1|a)$ *stochastically dominates* $\mathbb{P}(X_1|b)$ for actions a, b , then a is always the better choice than b , with all other attributes X_i being equal.
 \Rightarrow If some action $\mathbb{P}(X_i|a)$ *stochastically dominates* $\mathbb{P}(X_i|b)$ for all *attributes* X_i , we can ignore b .
- ▶ **Observation:** Stochastic dominance can often be determined without exact distributions using *qualitative* reasoning.
- ▶ **Example 5.10 (Construction cost increases with distance).** If airport location S_1 is closer to the city than $S_2 \rightsquigarrow S_1$ stochastically dominates S_2 on cost. q

- ▶ **Recall:** In deterministic environments an agent has a value function.
- ▶ **Definition 5.11.** X_1 and X_2 **preferentially independent** of X_3 iff preference between $\langle x_1, x_2, z \rangle$ and $\langle x'_1, x'_2, z \rangle$ does not depend on z . (i.e. the tradeoff between x_1 and x_2 is independent of z)
- ▶ **Example 5.12.** E.g., $\langle \text{Noise, Cost, Safety} \rangle$: are **preferentially independent** $\langle 20,000 \text{ suffer, } 4.6 \text{ G\$, } 0.06 \text{ deaths/mpm} \rangle$ vs. $\langle 70,000 \text{ suffer, } 4.2 \text{ G\$, } 0.06 \text{ deaths/mpm} \rangle$
- ▶ **Theorem 5.13 (Leontief, 1947).** If every pair of attributes is **preferentially independent** of its complement, then every subset of attributes is **preferentially independent** of its complement: **mutual preferential independence**.
- ▶ **Theorem 5.14 (Debreu, 1960).** **Mutual preferential independence** implies that there is an **additive value function**: $V(S) = \sum_i V_i(X_i(S))$, where V_i is a **value function** referencing just one variable X_i .
- ▶ Hence assess n single-attribute functions. (often a good approximation)
- ▶ **Example 5.15.** The **value function** for the airport decision might be

$$V(\text{noise, cost, deaths}) = -\text{noise} \cdot 10^4 - \text{cost} - \text{deaths} \cdot 10^{12}$$

- ▶ **Definition 5.16.** X is **utility independent** of Y iff **preferences** over **lotteries** in X do not depend on particular values in Y
- ▶ **Definition 5.17.** A set X is **mutually utility independent (MUI)**, iff each subset is **utility independent** of its complement.

- ▶ **Definition 5.19.** X is **utility independent** of Y iff **preferences** over **lotteries** in X do not depend on particular values in Y
- ▶ **Definition 5.20.** A set X is **mutually utility independent (MUI)**, iff each subset is **utility independent** of its complement.
- ▶ **Theorem 5.21.** For a **MUI** set of **attributes** \mathcal{X} , there is a **multiplicative utility function** of the form:
[Keeney:muf74]

$$U = \sum_{(\{X_0, \dots, X_k\} \subseteq \mathcal{X})} \prod_{i=1}^k U_i(X_i = x_i)$$

$\Rightarrow U$ can be represented using n single-attribute utility functions.

- ▶ **System Support:** Routine procedures and software packages for generating **preference** tests to identify various canonical families of **utility functions**.

- ▶ There are multiple ways to improve inference in decision networks:
- ▶ Exploit “inner structure” of the utility function to simplify the computation,
- ▶ eliminate dominated actions,
- ▶ label pairs of nodes with *stochastic dominance*: If (the utility of) some attribute dominates (the utility of) another attribute, focus on the dominant one (e.g. if price is always more important than quality, ignore quality whenever the price between two choices differs)
- ▶ various techniques for variable elimination,
- ▶ policy iteration (more on that when we talk about Markov decision procedures)

24.6 The Value of Information

What if we do not have all information we need?

- ▶ We now know how to exploit the information we have to make decisions. But if we knew more, we might be able to make even better decisions in the long run - potentially at the cost of gaining utility. (exploration vs. exploitation)
- ▶ **Example 6.1 (Medical Diagnosis).**
 - ▶ We do not expect a doctor to already know the results of the diagnostic tests when the patient comes in.
 - ▶ Tests are often expensive, and sometimes hazardous. (directly or by delaying treatment)
 - ▶ **Therefore:** Only test, if
 - ▶ knowing the results lead to a significantly better treatment plan,
 - ▶ information from test results is not drowned out by a-priori likelihood.
- ▶ **Definition 6.2.** Information value theory is concerned with agent making decisions on information gathering rationally.

Value of Information by Example

- ▶ **Idea:** Compute the expected *gain in utility* from acquiring information.
- ▶ **Example 6.3 (Buying Oil Drilling Rights).** There are n blocks of drilling rights available, exactly one block actually has oil worth $k\text{€}$, in particular:
 - ▶ The prior probability of a block having oil is $\frac{1}{n}$ each (mutually exclusive).
 - ▶ The current price of each block is $\frac{k}{n}\text{€}$.
 - ▶ A “consultant” offers an accurate survey of block (say) 3. How much should we be willing to pay for the survey?

Value of Information by Example

- ▶ **Idea:** Compute the expected *gain in utility* from acquiring information.
- ▶ **Example 6.5 (Buying Oil Drilling Rights).** There are n blocks of drilling rights available, exactly one block actually has oil worth $k\text{€}$, in particular:
 - ▶ The prior probability of a block having oil is $\frac{1}{n}$ each (mutually exclusive).
 - ▶ The current price of each block is $\frac{k}{n}\text{€}$.
 - ▶ A “consultant” offers an accurate survey of block (say) 3. How much should we be willing to pay for the survey?
- ▶ **Solution:** Compute the expected value of the best action given the information, minus the expected value of the best action without information.
- ▶ **Example 6.6 (Oil Drilling Rights contd.).**
 - ▶ Survey may say “*oil in block 3 with probability $\frac{1}{n}$* ” \leadsto we buy block 3 for $\frac{k}{n}\text{€}$ and make a profit of $(k - \frac{k}{n})\text{€}$.
 - ▶ Survey may say “*no oil in block 3 with probability $\frac{n-1}{n}$* ” \leadsto we buy another block, and make an expected profit of $\frac{k}{n-1} - \frac{k}{n}\text{€}$.
 - ▶ Without the survey, the expected profit is 0
 - ▶ Expected profit is $\frac{1}{n} \cdot \frac{(n-1)k}{n} + \frac{n-1}{n} \cdot \frac{k}{n(n-1)} = \frac{k}{n}$.
 - ▶ So, we should pay up to $\frac{k}{n}\text{€}$ for the information. (as much as block 3 is worth!)

- **Definition 6.7.** Let A the set of available actions and F a **random variable**. Given evidence $E_i = e_i$, let α be the action that maximizes **expected utility** a priori, and α_f the action that maximizes **expected utility** given $F = f$, i.e.: $\alpha = \operatorname{argmax}_{a \in A} \text{EU}(a | E_i = e_i)$ and $\alpha_f = \operatorname{argmax}_{a \in A} \text{EU}(a | E_i = e_i, F = f)$
- The **value of perfect information (VPI)** on F given evidence $E_i = e_i$ is defined as

$$\text{VPI}_{E_i=e_i}(F) := \left(\sum_{f \in \text{dom}(F)} P(F = f \mid E_i = e_i) \cdot \text{EU}(\alpha_f | E_i = e_i, F = f) \right) - \text{EU}(\alpha | E_i = e_i)$$

- **Intuition:** The **VPI** is the expected gain from knowing the value of F relative to the current expected utility, and considering the relative probabilities of the possible outcomes of F .

- **Observation 6.8 (VPI is Non-negative).**

$VPI_E(F) \geq 0$ for all j and E

(in expectation, not post hoc)

- **Observation 6.9 (VPI is Non-additive).**

$VPI_E(F, G) \neq VPI_E(F) + VPI_E(G)$

(consider, e.g., obtaining F twice)

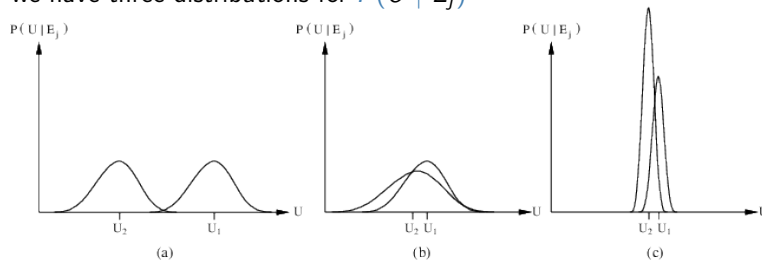
- **Observation 6.10 (VPI is Order-independent).**

$$VPI_E(F, G) = VPI_E(F) + VPI_{E,F}(G) = VPI_E(G) + VPI_{E,G}(F)$$

- **Note:** When more than one piece of evidence can be gathered, maximizing VPI for each to select one is not always optimal
 \leadsto evidence-gathering becomes a sequential decision problem.

Qualitative behavior of VPI

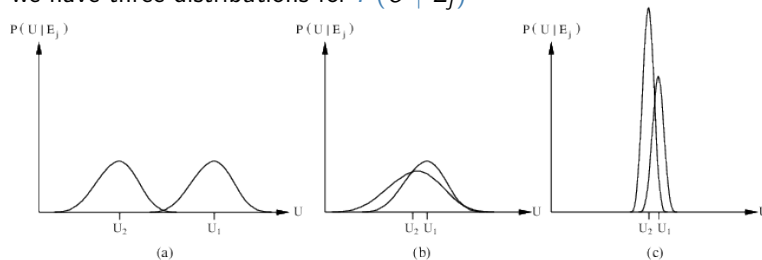
► **Question:** Say we have three distributions for $P(U | E_j)$



Qualitatively: What is the value of information (VPI) in these three cases?

Qualitative behavior of VPI

► **Question:** Say we have three distributions for $P(U | E_j)$

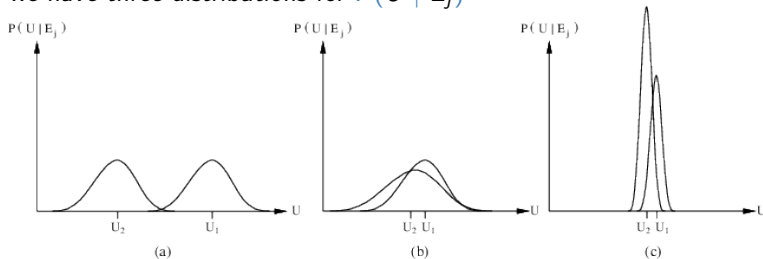


Qualitatively: What is the value of information (VPI) in these three cases?

► **Answers:** qualitatively:

a) Choice is obvious (a_1 almost certainly better) \leadsto information worth little

► **Question:** Say we have three distributions for $P(U | E_j)$



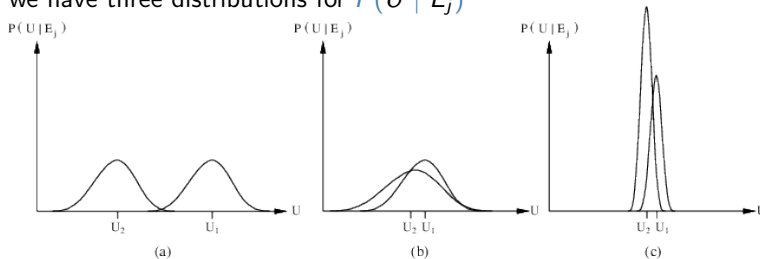
Qualitatively: What is the value of information (VPI) in these three cases?

► **Answers:** qualitatively:

- a) Choice is obvious (a_1 almost certainly better) \leadsto information worth little
- b) Choice is non-obvious (unclear) \leadsto information worth a lot

Qualitative behavior of VPI

► **Question:** Say we have three distributions for $P(U | E_j)$



Qualitatively: What is the value of information (VPI) in these three cases?

► **Answers:** qualitatively:

- a) Choice is obvious (a_1 almost certainly better) \leadsto information worth little
- b) Choice is non-obvious (unclear) \leadsto information worth a lot
- c) Choice is non-obvious (unclear) **but** makes little difference \leadsto information worth little

Note two things

- The difference between (b) and (c) is the width of the distribution, i.e. how close the possible outcomes are together
- The fact that U_2 has a high peak in (c) means that its expected value is known with higher certainty than U_1 .

(irrelevant to the argument)

A simple Information-Gathering Agent

- **Definition 6.11.** A simple **information gathering agent**. (gathers info before acting)

function Information—Gathering—Agent (percept) **returns** an action

persistent: D , a decision network

integrate percept into D

$j := \operatorname{argmax}_k \text{VPI}_E(E_k) / \text{Cost}(E_k)$

if $\text{VPI}_E(E_j) > \text{Cost}(E_j)$ **return** Request(E_j)

else return the best action from D

The next **percept** after Request(E_j) provides a value for E_j .

- **Problem:** The **information gathering implemented** here is **myopic**, i.e. only acquires a single **evidence variable**, or acts immediately. (cf. greedy search)
- But it works relatively well in practice. (e.g. outperforms humans for selecting diagnostic tests)
- Strategies for nonmyopic information gathering exist (Not discussed in this course)

Summary

- ▶ An MEU agent maximizes expected utility.
- ▶ Decision theory provides a framework for rational decision making.
- ▶ Decision networks augment Bayesian networks with action nodes and a utility node.
- ▶ rational preferences allow us to obtain a utility function (orderability, transitivity, continuity, substitutability, monotonicity, decomposability)
- ▶ multi-attribute utility functions can usually be “destructured” to allow for better inference and representation (can be monotone, attributes may dominate others, actions may dominate others, may be multiplicative,...)
- ▶ information value theory tells us when to explore rather than exploit, using
- ▶ VPI (value of perfect information) to determine how much to “pay” for information.

Chapter 25

Temporal Probability Models

25.1 Modeling Time and Uncertainty

The world changes in *stochastically predictable* ways.

Example 1.1.

- ▶ The weather changes, but the weather tomorrow is somewhat predictable *given* today's weather and other factors, (which in turn (somewhat) depends on yesterday's weather, which in turn...)
- ▶ the stock market changes, but the stock price tomorrow is probably related to today's price,
- ▶ A patient's blood sugar changes, but their blood sugar is related to their blood sugar 10 minutes ago (in particular if they didn't eat anything in between)

How do we model this?

Stochastic Processes

The world changes in *stochastically predictable ways*.

Example 1.4.

- ▶ The weather changes, but the weather tomorrow is somewhat predictable *given* today's weather and other factors, (which in turn (somewhat) depends on yesterday's weather, which in turn...)
- ▶ the stock market changes, but the stock price tomorrow is probably related to today's price,
- ▶ A patient's blood sugar changes, but their blood sugar is related to their blood sugar 10 minutes ago (in particular if they didn't eat anything in between)

How do we model this?

Definition 1.5. Let $\langle \Omega, P \rangle$ a probability space and $\langle S, \preceq \rangle$ a (not necessarily *totally*) ordered set. A sequence of random variables $(X_t)_{t \in S}$ with $\text{dom}(X_t) = D$ is called a **stochastic process** over the **time structure** S .

Intuition: X_t models the outcome of the random variable X at time step t . The **sample space** Ω corresponds to the set of all possible sequences of outcomes.

Note: We will almost exclusively use $\langle S, \preceq \rangle = \langle \mathbb{N}, \leq \rangle$.

Definition 1.6. Given a **stochastic process** X_t over S and $a, b \in S$ with $a \preceq b$, we write $X_{a:b}$ for the sequence $X_a, X_{a+1}, \dots, X_{b-1}, X_b$ and $E_{a:b}^{\bar{e}}$ for $E_a = e_a, \dots, E_b = e_b$.

Example 1.7 (Umbrellas). You are a security guard in a secret underground facility, want to know it if is raining outside. Your only source of information is whether the director comes in with an umbrella.

- ▶ We have a stochastic process $\text{Rain}_0, \text{Rain}_1, \text{Rain}_2, \dots$ of hidden variables, and
- ▶ a related stochastic process $\text{Umbrella}_0, \text{Umbrella}_1, \text{Umbrella}_2, \dots$ of evidence variables.

...and a combined stochastic process $\langle \text{Rain}_0, \text{Umbrella}_0 \rangle, \langle \text{Rain}_1, \text{Umbrella}_1 \rangle, \dots$

Note that Umbrella_t only depends on Rain_t , not on e.g. Umbrella_{t-1} (except indirectly through Rain_t / Rain_{t-1}).

Definition 1.8. We call a stochastic process of *hidden* variables a **state variable**.

Idea: Construct a Bayesian network from these variables
...without everything exploding in size...?

(parents?)

Markov Processes

Idea: Construct a Bayesian network from these variables (parents?)
...without everything exploding in size...?

Definition 1.11. Let $(X_t)_{t \in S}$ a stochastic process. X has the (n th order) Markov property iff X_t only depends on a bounded subset of $X_{0:t-1}$ – i.e. for all $t \in S$ we have

$$\mathbb{P}(X_t | X_0, \dots, X_{t-1}) = \mathbb{P}(X_t | X_{t-n}, \dots, X_{t-1}) \text{ for some } n \in S.$$

A stochastic process with the Markov property for some n is called a (n th order) Markov process.

Markov Processes

Idea: Construct a Bayesian network from these variables
...without everything exploding in size...?

(parents?)

Definition 1.13. Let $(X_t)_{t \in S}$ a stochastic process. X has the (n th order) Markov property iff X_t only depends on a bounded subset of $X_{0:t-1}$ – i.e. for all $t \in S$ we have

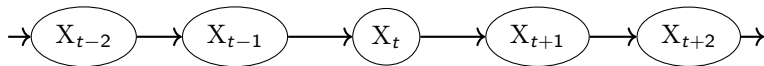
$$\mathbb{P}(X_t | X_0, \dots, X_{t-1}) = \mathbb{P}(X_t | X_{t-n}, \dots, X_{t-1}) \text{ for some } n \in S.$$

A stochastic process with the Markov property for some n is called a (n th order) Markov process.

Important special cases:

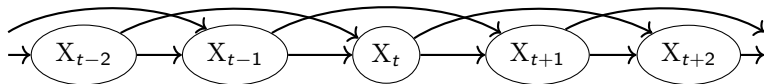
Definition 1.14.

► **First-order Markov property:** $\mathbb{P}(X_t | X_{0:t-1}) = \mathbb{P}(X_t | X_{t-1})$



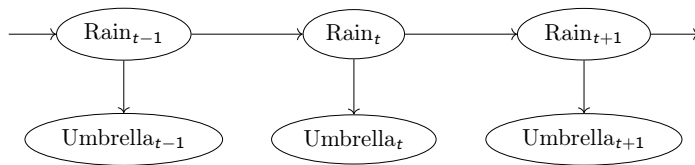
A first order Markov process is called a Markov chain.

► **Second-order Markov property:** $\mathbb{P}(X_t | X_{0:t-1}) = \mathbb{P}(X_t | X_{t-2}, X_{t-1})$



Markov Process Example: The Umbrella

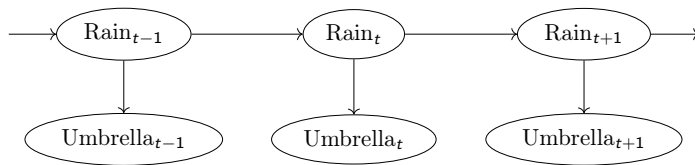
Example 1.15 (Umbrellas continued). We model the situation in a Bayesian network:



Problem: This network does not actually have the First-order Markov property...

Markov Process Example: The Umbrella

Example 1.16 (Umbrellas continued). We model the situation in a **Bayesian network**:



Problem: This network does not actually have the **First-order Markov property**...

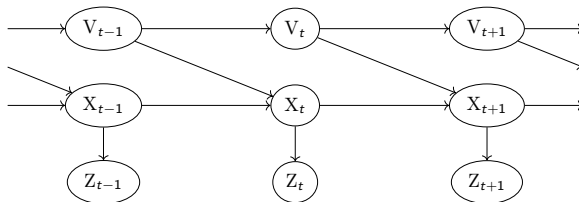
Possible fixes: We have two ways to fix this:

1. Increase the **order** of the **Markov process**. (more dependencies \Rightarrow more complex inference)
2. Add more **state variables**, e.g., Temp_t , Pressure_t . (more information sources)

Markov Process Example: Robot Motion

Example 1.17 (Random Robot Motion). Assume we want to track a robot wandering randomly on the X/Y plane, whose position we can only observe roughly (e.g. by approximate GPS coordinates:)

Markov chain



- ▶ the velocity V_i may change unpredictably.
- ▶ the exact position X_i depends on previous position X_{i-1} and velocity V_{i-1}
- ▶ the position X_i influences the observed position Z_i .

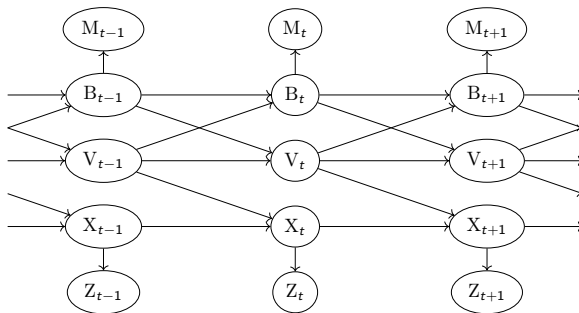
Example 1.18 (Battery Powered Robot). If the robot has a *battery*, the Markov property is violated!

- ▶ Battery exhaustion has a systematic effect on the change in velocity.
- ▶ This depends on how much power was used by all previous manoeuvres.

Markov Process Example: Robot Motion

Idea: We can restore the **Markov property** by including a **state variable** for the charge level B_t . (Better still: Battery level sensor)

Example 1.19 (Battery Powered Robot Motion).



- ▶ Battery level B_i is influenced by previous level B_{i-1} and velocity V_{i-1} .
- ▶ Velocity V_i is influenced by previous level B_{i-1} and velocity V_{i-1} .
- ▶ Battery meter M_i is only influenced by Battery level B_i .

Stationary Markov Processes as Transition Models

Remark 1.20. Given a stochastic process with state variables X_t and evidence variables E_t , then $\mathbb{P}(X_t|X_{0:t})$ is a transition model and $\mathbb{P}(E_t|X_{0:t}, E_{1:t-1})$ a sensor model in the sense of a model-based agent.

Note that we assume that the X_t do not depend on the E_t .

Also note that with the Markov property, the transition model simplifies to $\mathbb{P}(X_t|X_{t-n})$.

Problem: Even with the Markov property the transition model is infinite. ($t \in \mathbb{N}$)

Stationary Markov Processes as Transition Models

Remark 1.23. Given a stochastic process with state variables X_t and evidence variables E_t , then $\mathbb{P}(X_t|X_{0:t})$ is a transition model and $\mathbb{P}(E_t|X_{0:t}, E_{1:t-1})$ a sensor model in the sense of a model-based agent.

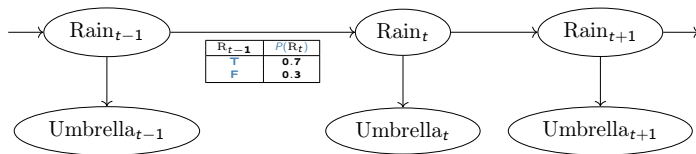
Note that we assume that the X_t do not depend on the E_t .

Also note that with the Markov property, the transition model simplifies to $\mathbb{P}(X_t|X_{t-n})$.

Problem: Even with the Markov property the transition model is infinite. ($t \in \mathbb{N}$)

Definition 1.24. A Markov chain is called stationary if the transition model is independent of time, i.e. $\mathbb{P}(X_t|X_{t-1})$ is the same for all t .

Example 1.25 (Umbrellas are stationary). $\mathbb{P}(\text{Rain}_t|\text{Rain}_{t-1})$ does not depend on t . (need only one table)



Stationary Markov Processes as Transition Models

Remark 1.26. Given a stochastic process with state variables X_t and evidence variables E_t , then $\mathbb{P}(X_t|X_{0:t})$ is a transition model and $\mathbb{P}(E_t|X_{0:t}, E_{1:t-1})$ a sensor model in the sense of a model-based agent.

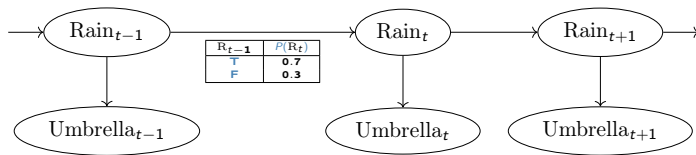
Note that we assume that the X_t do not depend on the E_t .

Also note that with the Markov property, the transition model simplifies to $\mathbb{P}(X_t|X_{t-n})$.

Problem: Even with the Markov property the transition model is infinite. ($t \in \mathbb{N}$)

Definition 1.27. A Markov chain is called stationary if the transition model is independent of time, i.e. $\mathbb{P}(X_t|X_{t-1})$ is the same for all t .

Example 1.28 (Umbrellas are stationary). $\mathbb{P}(\text{Rain}_t|\text{Rain}_{t-1})$ does not depend on t . (need only one table)



⚠ Don't confuse "stationary" (Markov processes) with "static" (environments).

We restrict ourselves to stationary Markov processes in AI-2.

Markov Sensor Models

Recap: The **sensor model** $\mathbb{P}(E_t | X_{0:t}, E_{1:t-1})$ allows us (using **Bayes rule** et al) to update our belief state about X_t given the observations $E_{0:t}$.

Problem: The **evidence variables** E_t could depend on any of the variables $X_{0:t}, E_{1:t-1} \dots$

Markov Sensor Models

Recap: The **sensor model** $\mathbb{P}(E_t | X_{0:t}, E_{1:t-1})$ allows us (using **Bayes rule** et al) to update our belief state about X_t given the observations $E_{0:t}$.

Problem: The **evidence variables** E_t could depend on any of the variables $X_{0:t}, E_{1:t-1} \dots$

Definition 1.31. We say that a **sensor model** has the **sensor Markov property**, iff $\mathbb{P}(E_t | X_{0:t}, E_{1:t-1}) = \mathbb{P}(E_t | X_t)$ – i.e., the sensor model depends only on the current state.

Assumptions on Sensor Models: We usually assume the **sensor Markov property** and make it **stationary** as well: $\mathbb{P}(E_t | X_t)$ is fixed for all t .

Markov Sensor Models

Recap: The **sensor model** $\mathbb{P}(E_t | X_{0:t}, E_{1:t-1})$ allows us (using **Bayes rule** et al) to update our belief state about X_t given the observations $E_{0:t}$.

Problem: The **evidence variables** E_t could depend on any of the variables $X_{0:t}, E_{1:t-1} \dots$

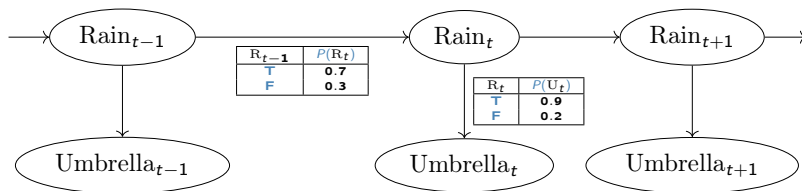
Definition 1.33. We say that a **sensor model** has the **sensor Markov property**, iff $\mathbb{P}(E_t | X_{0:t}, E_{1:t-1}) = \mathbb{P}(E_t | X_t)$ – i.e., the sensor model depends only on the current state.

Assumptions on Sensor Models: We usually assume the **sensor Markov property** and make it **stationary** as well: $\mathbb{P}(E_t | X_t)$ is fixed for all t .

Definition 1.34 (Note).

- ▶ If a **Markov chain** X is **stationary** and **discrete**, we can represent the **transition model** as a **matrix** $T_{ij} := P(X_t = j \mid X_{t-1} = i)$.
- ▶ If a **sensor model** has the **sensor Markov property**, we can represent each observation $E_t = e_t$ at time t as the **diagonal matrix** O_t with $O_{tii} := P(E_t = e_t \mid X_t = i)$.
- ▶ A pair $\langle X, E \rangle$ where X is a (**stationary**) **Markov chains**, E_i only depends on X_i , and E has the **sensor Markov property** is called a (**stationary**) **Hidden Markov Model (HMM)**. (X and E are **single variables**)

Example 1.35 (Umbrellas, Transition & Sensor Models).



This is a **hidden Markov model**

Observation 1.36. If we know the initial prior probabilities $\mathbb{P}(X_0)$ ($\hat{=}$ time $t = 0$), then we can compute the **full joint probability distribution** as

$$\mathbb{P}(X_{0:t}, E_{1:t}) = \mathbb{P}(X_0) \cdot \left(\prod_{i=1}^t \mathbb{P}(X_i | X_{i-1}) \cdot \mathbb{P}(E_i | X_i) \right)$$

25.2 Inference: Filtering, Prediction, and Smoothing

Inference tasks

Definition 2.1. Given a Markov process with state variables X_t and evidence variables E_t , we are interested in the following Markov inference tasks:

- ▶ **Filtering** (or **monitoring**) $\mathbb{P}(X_t | E_{1:t}^=)$: Given the sequence of observations up until time t , compute the likely state of the world at *current* time t .
- ▶ **Prediction** (or **state estimation**) $\mathbb{P}(X_{t+k} | E_{1:t}^=)$ for $k > 0$: Given the sequence of observations up until time t , compute the likely *future* state of the world at time $t + k$.
- ▶ **Smoothing** (or **hindsight**) $\mathbb{P}(X_{t-k} | E_{1:t}^=)$ for $0 < k < t$: Given the sequence of observations up until time t , compute the likely *past* state of the world at time $t - k$.
- ▶ **Most likely explanation** $\underset{x_{1:t}}{\operatorname{argmax}} (P(X_{1:t}^= | E_{1:t}^=))$: Given the sequence of observations up until time t , compute the most likely sequence of states that led to these observations.

Note: The most likely sequence of states is *not* (necessarily) the sequence of most likely states ;-)

In this section, we assume X and E to represent *multiple* variables, where X jointly forms a Markov chain and the E jointly have the sensor Markov property.

In the case where X and E are stationary *single* variables, we have a stationary hidden Markov model and can use the matrix forms.

Filtering (Computing the Belief State given Evidence)

Note:

- ▶ Using the **full joint probability distribution**, we can compute any **conditional probability** we want, but not necessarily efficiently.
- ▶ We want to use **filtering** to update our “world model” $\mathbb{P}(X_t)$ based on a new observation $E_t = e_t$ and our *previous* world model $\mathbb{P}(X_{t-1})$.

⇒ We want a function $\mathbb{P}(X_t | E_{1:t}^e) = F(e_t, \underbrace{\mathbb{P}(X_{t-1} | E_{1:t-1}^e)}_{F(e_{t-1}, \dots)})$

Filtering (Computing the Belief State given Evidence)

Note:

- ▶ Using the full joint probability distribution, we can compute any conditional probability we want, but not necessarily efficiently.
- ▶ We want to use filtering to update our “world model” $\mathbb{P}(X_t)$ based on a new observation $E_t = e_t$ and our *previous* world model $\mathbb{P}(X_{t-1})$.

⇒ We want a function $\mathbb{P}(X_t | E_{1:t}^e) = F(e_t, \underbrace{\mathbb{P}(X_{t-1} | E_{1:t-1}^e)}_{F(e_{t-1}, \dots)})$

Spoiler:

$$F(e_t, \mathbb{P}(X_{t-1} | E_{1:t-1}^e)) = \alpha(O_t \cdot T^T \cdot \mathbb{P}(X_{t-1} | E_{1:t-1}^e))$$

Filtering Derivation

$$\begin{aligned}\mathbb{P}(X_t | E_{1:t}^{\bar{e}}) &= \mathbb{P}(X_t | E_t = e_t, E_{1:t-1}^{\bar{e}}) && \text{(dividing up evidence)} \\ &= \alpha(\mathbb{P}(E_t = e_t | X_t, E_{1:t-1}^{\bar{e}}) \cdot \mathbb{P}(X_t | E_{1:t-1}^{\bar{e}})) && \text{(using Bayes' rule)} \\ &= \alpha(\mathbb{P}(E_t = e_t | X_t) \cdot \mathbb{P}(X_t | E_{1:t-1}^{\bar{e}})) && \text{(sensor Markov property)} \\ &= \alpha(\mathbb{P}(E_t = e_t | X_t) \cdot (\sum_{x \in \text{dom}(X)} \mathbb{P}(X_t | X_{t-1} = x, E_{1:t-1}^{\bar{e}}) \cdot P(X_{t-1} = x | E_{1:t-1}^{\bar{e}}))) && \text{(marginalization)} \\ &= \underbrace{\alpha(\mathbb{P}(E_t = e_t | X_t))}_{\text{sensor model}} \cdot (\sum_{x \in \text{dom}(X)} \underbrace{\mathbb{P}(X_t | X_{t-1} = x)}_{\text{transition model}} \cdot \underbrace{P(X_{t-1} = x | E_{1:t-1}^{\bar{e}})}_{\text{recursive call}})) && \text{(conditional independence)}\end{aligned}$$

$$\begin{aligned}
 \mathbb{P}(X_t | E_{1:t}^{\bar{e}}) &= \mathbb{P}(X_t | E_t = e_t, E_{1:t-1}^{\bar{e}}) && \text{(dividing up evidence)} \\
 &= \alpha(\mathbb{P}(E_t = e_t | X_t, E_{1:t-1}^{\bar{e}}) \cdot \mathbb{P}(X_t | E_{1:t-1}^{\bar{e}})) && \text{(using Bayes' rule)} \\
 &= \alpha(\mathbb{P}(E_t = e_t | X_t) \cdot \mathbb{P}(X_t | E_{1:t-1}^{\bar{e}})) && \text{(sensor Markov property)} \\
 &= \alpha(\mathbb{P}(E_t = e_t | X_t) \cdot (\sum_{x \in \text{dom}(X)} \mathbb{P}(X_t | X_{t-1} = x, E_{1:t-1}^{\bar{e}}) \cdot P(X_{t-1} = x | E_{1:t-1}^{\bar{e}}))) && \text{(marginalization)} \\
 &= \underbrace{\alpha(\mathbb{P}(E_t = e_t | X_t))}_{\text{sensor model}} \cdot \underbrace{(\sum_{x \in \text{dom}(X)} \underbrace{\mathbb{P}(X_t | X_{t-1} = x)}_{\text{transition model}} \cdot \underbrace{P(X_{t-1} = x | E_{1:t-1}^{\bar{e}})}_{\text{recursive call}}))}_{\text{(conditional independence)}}
 \end{aligned}$$

Reminder: In a stationary HMM, we have the matrices $T_{ij} = P(X_t = j | X_{t-1} = i)$ and

$O_{tij} = P(E_t = e_t | X_t = i)$.

Then interpreting $\mathbb{P}(X_{t-1} | E_{1:t-1}^{\bar{e}})$ as a **vector**, the above corresponds exactly to the **matrix multiplication** $\alpha(O_t \cdot T^T \cdot \mathbb{P}(X_{t-1} | E_{1:t-1}^{\bar{e}}))$

Definition 2.3. We call the inner part of the above expression the **forward** algorithm, i.e.

$$\mathbb{P}(X_t | E_{1:t}^{\bar{e}}) = \alpha(\text{FORWARD}(e_t, \mathbb{P}(X_{t-1} | E_{1:t-1}^{\bar{e}}))) =: \mathbf{f}_{1:t}.$$

Filtering the Umbrellas

Example 2.4. Let's assume:

► $\mathbb{P}(\mathbf{R}_0) = \langle 0.5, 0.5 \rangle$, (Note that with growing t (and evidence), the impact of the prior at $t = 0$ vanishes anyway)

► $P(\mathbf{R}_{t+1} \mid \mathbf{R}_t) = 0.6$, $P(\neg \mathbf{R}_{t+1} \mid \neg \mathbf{R}_t) = 0.8$, $P(\mathbf{U}_t \mid \mathbf{R}_t) = 0.9$ and $P(\neg \mathbf{U}_t \mid \neg \mathbf{R}_t) = 0.85$

$$\Rightarrow \mathbf{T} = \begin{pmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{pmatrix}$$

► The director carries an umbrella on days 1 and 2, and *not* on day 3.

$$\Rightarrow \mathbf{O}_1 = \mathbf{O}_2 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.15 \end{pmatrix} \text{ and } \mathbf{O}_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.85 \end{pmatrix}.$$

Then:

$$\text{► } f_{1:1} := \mathbb{P}(\mathbf{R}_1 \mid \mathbf{U}_1 = \mathbf{T}) = \alpha(\mathbb{P}(\mathbf{U}_1 = \mathbf{T} \mid \mathbf{R}_1) \cdot (\sum_{b \in \{\mathbf{T}, \mathbf{F}\}} \mathbb{P}(\mathbf{R}_1 \mid \mathbf{R}_0 = b) \cdot P(\mathbf{R}_0 = b)))$$

$$= \alpha(\langle 0.9, 0.15 \rangle \cdot (\langle 0.6, 0.4 \rangle \cdot 0.5 + \langle 0.2, 0.8 \rangle \cdot 0.5)) = \alpha(\langle 0.36, 0.09 \rangle) = \langle 0.8, 0.2 \rangle$$

$$\text{► Using matrices: } \alpha(\mathbf{O}_1 \cdot \mathbf{T}^T \cdot \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}) = \alpha(\begin{pmatrix} 0.9 & 0 \\ 0 & 0.15 \end{pmatrix} \cdot \begin{pmatrix} 0.6 & 0.2 \\ 0.4 & 0.8 \end{pmatrix} \cdot \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix})$$

$$= \alpha(\begin{pmatrix} 0.9 \cdot 0.6 & 0.9 \cdot 0.2 \\ 0.15 \cdot 0.4 & 0.15 \cdot 0.8 \end{pmatrix} \cdot \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}) = \alpha(\begin{pmatrix} 0.9 \cdot 0.6 \cdot 0.5 + 0.9 \cdot 0.2 \cdot 0.5 \\ 0.15 \cdot 0.4 \cdot 0.5 + 0.15 \cdot 0.8 \cdot 0.5 \end{pmatrix}) = \alpha(\begin{pmatrix} 0.36 \\ 0.09 \end{pmatrix})$$

Example 2.5. $f_{1:1} := \mathbb{P}(R_1 | U_1 = T) = \langle 0.8, 0.2 \rangle$

$$\begin{aligned} \blacktriangleright f_{1:2} &:= \mathbb{P}(R_2 | U_2 = T, U_1 = T) = \alpha(O_2 \cdot T^T \cdot f_{1:1}) = \alpha(\mathbb{P}(U_2 = T | R_2) \cdot (\sum_{b \in \{T, F\}} \mathbb{P}(R_2 | R_1 = b) \cdot f_{1:1}(b))) \\ &= \alpha(\langle 0.9, 0.15 \rangle \cdot (\langle 0.6, 0.4 \rangle \cdot 0.8 + \langle 0.2, 0.8 \rangle \cdot 0.2)) = \alpha(\langle 0.468, 0.072 \rangle) = \langle 0.87, 0.13 \rangle \end{aligned}$$

$$\begin{aligned} \blacktriangleright f_{1:3} &:= \mathbb{P}(R_3 | U_3 = F, U_2 = T, U_1 = T) = \alpha(O_3 \cdot T^T \cdot f_{1:2}) \\ &= \alpha(\mathbb{P}(U_3 = F | R_3) \cdot (\sum_{b \in \{T, F\}} \mathbb{P}(R_3 | R_2 = b) \cdot f_{1:2}(b))) \\ &= \alpha(\langle 0.1, 0.85 \rangle \cdot (\langle 0.6, 0.4 \rangle \cdot 0.87 + \langle 0.2, 0.8 \rangle \cdot 0.13)) = \alpha(\langle 0.0547, 0.3853 \rangle) = \langle 0.12, 0.88 \rangle \end{aligned}$$

Prediction in Markov Chains

Prediction: $\mathbb{P}(X_{t+k} | E_{1:t}^{\neg e})$ for $k > 0$.

Intuition: Prediction is filtering without new evidence – i.e. we can use filtering until t , and then continue as follows:

Lemma 2.6. By the same reasoning as filtering:

$$\mathbb{P}(X_{t+k+1} | E_{1:t}^{\neg e}) = \sum_{x \in \text{dom}(X)} \underbrace{\mathbb{P}(X_{t+k+1} | X_{t+k} = x)}_{\text{transition model}} \cdot \underbrace{P(X_{t+k} = x | E_{1:t}^{\neg e})}_{\text{recursive call}} = \mathbf{T}^T \cdot \underbrace{\mathbb{P}(X_{t+k} = x | E_{1:t}^{\neg e})}_{\text{HMM}}$$

Prediction in Markov Chains

Prediction: $\mathbb{P}(X_{t+k} | E_{1:t}^{\bar{e}})$ for $k > 0$.

Intuition: Prediction is filtering without new evidence – i.e. we can use filtering until t , and then continue as follows:

Lemma 2.8. By the same reasoning as filtering:

$$\mathbb{P}(X_{t+k+1} | E_{1:t}^{\bar{e}}) = \sum_{x \in \text{dom}(X)} \underbrace{\mathbb{P}(X_{t+k+1} | X_{t+k} = x)}_{\text{transition model}} \cdot \underbrace{P(X_{t+k} = x | E_{1:t}^{\bar{e}})}_{\text{recursive call}} = \mathbf{T}^T \cdot \underbrace{\mathbb{P}(X_{t+k} = x | E_{1:t}^{\bar{e}})}_{\text{HMM}}$$

Observation 2.9. As $k \rightarrow \infty$, $\mathbb{P}(X_{t+k} | E_{1:t}^{\bar{e}})$ converges towards a fixed point called the stationary distribution of the Markov chain. (which we can compute from the equation $S = \mathbf{T}^T \cdot S$)

↪ the impact of the evidence vanishes.

↪ The stationary distribution only depends on the transition model.

↪ There is a small window of time (depending on the transition model) where the evidence has enough impact to allow for prediction beyond the mere stationary distribution, called the mixing time of the Markov chain.

↪ Predicting the future is difficult, and the further into the future, the more difficult it is (Who knew...)

Smoothing: $\mathbb{P}(X_{t-k} | E_{1:t}^{\bar{e}})$ for $k > 0$.

Intuition: Use **filtering** to compute $\mathbb{P}(X_t | E_{1:t-k}^{\bar{e}})$, then recurse *backwards* from t until $t - k$.

$$\begin{aligned}\mathbb{P}(X_{t-k} | E_{1:t}^{\bar{e}}) &= \mathbb{P}(X_{t-k} | E_{t-(k-1):t}^{\bar{e}}, E_{1:t-k}^{\bar{e}}) && \text{(Divide the evidence)} \\ &= \alpha(\mathbb{P}(E_{t-(k-1):t}^{\bar{e}} | X_{t-k}, E_{1:t-k}^{\bar{e}}) \cdot \mathbb{P}(X_{t-k} | E_{1:t-k}^{\bar{e}})) && \text{(Bayes Rule)} \\ &= \underbrace{\alpha(\mathbb{P}(E_{t-(k-1):t}^{\bar{e}} | X_{t-k}))}_{=: \mathbf{b}_{t-(k-1):t}} \cdot \underbrace{\mathbb{P}(X_{t-k} | E_{1:t-k}^{\bar{e}})}_{=: \mathbf{f}_{1:t-k}} && \text{(cond. independence)} \\ &= \alpha(\mathbf{f}_{1:t-k} \times \mathbf{b}_{t-(k-1):t})\end{aligned}$$

(where \times denotes component-wise multiplication)

Smoothing (continued)

Definition 2.10 (Backward message). $\mathbf{b}_{t-k:t} = \mathbb{P}(E_{t-k:t}^{\bar{e}} | X_{t-(k+1)})$

$$\begin{aligned} &= \sum_{x \in \text{dom}(X)} \mathbb{P}(E_{t-k:t}^{\bar{e}} | X_{t-k} = x, X_{t-(k+1)}) \cdot \mathbb{P}(X_{t-k} = x | X_{t-(k+1)}) \\ &= \sum_{x \in \text{dom}(X)} P(E_{t-k:t}^{\bar{e}} | X_{t-k} = x) \cdot \mathbb{P}(X_{t-k} = x | X_{t-(k+1)}) \\ &= \sum_{x \in \text{dom}(X)} P(E_{t-k} = e_{t-k}, E_{t-(k-1):t}^{\bar{e}} | X_{t-k} = x) \cdot \mathbb{P}(X_{t-k} = x | X_{t-(k+1)}) \\ &= \sum_{x \in \text{dom}(X)} \underbrace{P(E_{t-k} = e_{t-k} | X_{t-k} = x)}_{\text{sensor model}} \cdot \underbrace{P(E_{t-(k-1):t}^{\bar{e}} | X_{t-k} = x)}_{=\mathbf{b}_{t-(k-1):t}} \cdot \underbrace{\mathbb{P}(X_{t-k} = x | X_{t-(k+1)})}_{\text{transition model}} \end{aligned}$$

Note: in a stationary hidden Markov model, we get the matrix formulation $\mathbf{b}_{t-k:t} = \mathbf{T} \cdot \mathbf{O}_{t-k} \cdot \mathbf{b}_{t-(k-1):t}$

Smoothing (continued)

Definition 2.12 (Backward message). $\mathbf{b}_{t-k:t} = \mathbb{P}(E_{t-k:t}^{\bar{e}} | X_{t-(k+1)})$

$$\begin{aligned} &= \sum_{x \in \text{dom}(X)} \mathbb{P}(E_{t-k:t}^{\bar{e}} | X_{t-k} = x, X_{t-(k+1)}) \cdot \mathbb{P}(X_{t-k} = x | X_{t-(k+1)}) \\ &= \sum_{x \in \text{dom}(X)} P(E_{t-k:t}^{\bar{e}} | X_{t-k} = x) \cdot \mathbb{P}(X_{t-k} = x | X_{t-(k+1)}) \\ &= \sum_{x \in \text{dom}(X)} P(E_{t-k} = e_{t-k}, E_{t-(k-1):t}^{\bar{e}} | X_{t-k} = x) \cdot \mathbb{P}(X_{t-k} = x | X_{t-(k+1)}) \\ &= \sum_{x \in \text{dom}(X)} \underbrace{P(E_{t-k} = e_{t-k} | X_{t-k} = x)}_{\text{sensor model}} \cdot \underbrace{P(E_{t-(k-1):t}^{\bar{e}} | X_{t-k} = x)}_{=\mathbf{b}_{t-(k-1):t}} \cdot \underbrace{\mathbb{P}(X_{t-k} = x | X_{t-(k+1)})}_{\text{transition model}} \end{aligned}$$

Note: in a stationary hidden Markov model, we get the matrix formulation $\mathbf{b}_{t-k:t} = \mathbf{T} \cdot \mathbf{O}_{t-k} \cdot \mathbf{b}_{t-(k-1):t}$

Definition 2.13. We call the associated algorithm the **backward** algorithm, i.e.

$$\mathbb{P}(X_{t-k} | E_{1:t}^{\bar{e}}) = \alpha \left(\underbrace{\text{FORWARD}(e_{t-k}, \mathbf{f}_{1:t-k})}_{\mathbf{f}_{1:t-k}} \times \underbrace{\text{BACKWARD}(e_{t-(k-1)}, \mathbf{b}_{t-(k-1):t})}_{\mathbf{b}_{t-(k-1):t}} \right).$$

As a starting point for the recursion, we let $\mathbf{b}_{t+1:t}$ the uniform vector with 1 in every component.

Smoothing example

Example 2.14 (Smoothing Umbrellas). **Reminder:** We assumed $\mathbb{P}(R_0) = \langle 0.5, 0.5 \rangle$,

$P(R_{t+1} | R_t) = 0.6$, $P(\neg R_{t+1} | \neg R_t) = 0.8$, $P(U_t | R_t) = 0.9$, $P(\neg U_t | \neg R_t) = 0.85$

$\Rightarrow T = \begin{pmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{pmatrix}$, $O_1 = O_2 = \begin{pmatrix} 0.9 & 0 \\ 0 & 0.15 \end{pmatrix}$ and $O_3 = \begin{pmatrix} 0.1 & 0 \\ 0 & 0.85 \end{pmatrix}$. (The director carries an umbrella on days 1 and 2, and *not* on day 3)

$f_{1:1} = \langle 0.8, 0.2 \rangle$, $f_{1:2} = \langle 0.87, 0.13 \rangle$ and $f_{1:3} = \langle 0.12, 0.88 \rangle$

Let's compute

$$\mathbb{P}(R_1 | U_1 = T, U_2 = T, U_3 = F) = \alpha(f_{1:1} \times b_{2:3})$$

► We need to compute $b_{2:3}$ and $b_{3:3}$:

$$\text{► } b_{3:3} = T \cdot O_3 \cdot b_{4:3} = \begin{pmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{pmatrix} \cdot \begin{pmatrix} 0.1 & 0 \\ 0 & 0.85 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0.7 \end{pmatrix}$$

$$\text{► } b_{2:3} = T \cdot O_2 \cdot b_{3:3} = \begin{pmatrix} 0.6 & 0.4 \\ 0.2 & 0.8 \end{pmatrix} \cdot \begin{pmatrix} 0.9 & 0 \\ 0 & 0.15 \end{pmatrix} \cdot \begin{pmatrix} 0.4 \\ 0.7 \end{pmatrix} = \begin{pmatrix} 0.258 \\ 0.156 \end{pmatrix}$$

$$\Rightarrow \alpha\left(\begin{pmatrix} 0.8 \\ 0.2 \end{pmatrix} \times \begin{pmatrix} 0.258 \\ 0.156 \end{pmatrix}\right) = \alpha\left(\begin{pmatrix} 0.2064 \\ 0.0312 \end{pmatrix}\right) = \begin{pmatrix} 0.87 \\ 0.13 \end{pmatrix}$$

\Rightarrow Given the evidence $U_2, \neg U_3$, the posterior probability for R_1 went up from 0.8 to 0.87!

Forward/Backward Algorithm for Smoothing

Definition 2.15. **Forward backward algorithm:** returns the sequence of posterior distributions $\mathbb{P}(X_1) \dots \mathbb{P}(X_t)$ given evidence e_1, \dots, e_t :

```
function FORWARD-BACKWARD( $\langle e_1, \dots, e_t \rangle, \mathbb{P}(X_0)$ )  
   $f := \langle \mathbb{P}(X_0) \rangle$   
   $b := \langle 1, 1, \dots \rangle$   
   $S := \langle \mathbb{P}(X_0) \rangle$   
  for  $i = 1, \dots, t$  do  
     $f_i := \text{FORWARD}(f_{i-1}, e_i)$  /* filtering */  
  for  $i = t, \dots, 1$  do  
     $S_i := \alpha(f_i \times b)$  /* smoothing */  
     $b := \text{BACKWARD}(b, e_i)$   
  return  $S$ 
```

Time complexity linear in t (polytree inference), Space complexity $\mathcal{O}(t \cdot |f|)$.

Country dance algorithm

Idea: If \mathbf{T} and \mathbf{O}_i are *invertible*, we can avoid storing all forward messages in the *smoothing* algorithm by running *filtering* backwards:

$$\begin{aligned} \mathbf{f}_{1:i+1} &= \alpha(\mathbf{O}_{i+1} \cdot \mathbf{T}^T \cdot \mathbf{f}_{1:i}) \\ \Rightarrow \mathbf{f}_{1:i} &= \alpha(\mathbf{T}^{T^{-1}} \cdot \mathbf{O}_{i+1}^{-1} \cdot \mathbf{f}_{1:i+1}) \end{aligned}$$

\Rightarrow we can trade *space complexity* for *time complexity*:

- ▶ In the first for-loop, we only compute the final $\mathbf{f}_{1:t}$ (No need to store the intermediate results)
 - ▶ In the second for-loop, we compute both $\mathbf{f}_{1:i}$ and $\mathbf{b}_{t-i:t}$ (Only one copy of $\mathbf{f}_{1:i}$, $\mathbf{b}_{t-i:t}$ is stored)
- \Rightarrow *constant space*.

But: Requires that both *matrices* are *invertible*, i.e. *every observation must be possible in every state*.
(Possible hack: increase the probabilities of 0 to “negligibly small”)

Most Likely Explanation

Smoothing allows us to compute the *sequence of most likely states* X_1, \dots, X_t given $E_{1:t}^e$. What if we want the *most likely sequence* of states? i.e. $\max_{x_1, \dots, x_t} (P(X_{1:t}^x | E_{1:t}^e))$?

Example 2.16. Given the sequence $U_1, U_2, \neg U_3, U_4, U_5$, the most likely state for R_3 is F , but the most likely sequence *might* be that it rained throughout...

Prominent Application: In speech recognition, we want to find the **most likely** word sequence, given what we have heard. (can be quite noisy)

Idea:

- ▶ For every $x_t \in \text{dom}(X)$ and $0 \leq i \leq t$, recursively compute the most likely path X_1, \dots, X_i ending in $X_i = x_i$ given the observed evidence.
- ▶ remember the x_{i-1} that most likely leads to x_i .
- ▶ Among the resulting paths, pick the one to *the* $X_t = x_t$ with the most likely path,
- ▶ and then recurse backwards.

\leadsto we want to know $\max_{x_1, \dots, x_{t-1}} \mathbb{P}(X_{1:t-1}^x, X_t | E_{1:t}^e)$, and then pick the x_t with the maximal value.

Most Likely Explanation (continued)

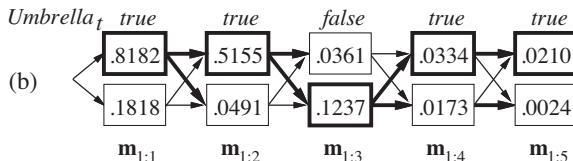
By the same reasoning as for **filtering**:

$$\begin{aligned}
 & \max_{x_1, \dots, x_{t-1}} \mathbb{P}(X_{1:t-1}^{\bar{x}}, X_t | E_{1:t}^{\bar{e}}) \\
 &= \underbrace{\alpha \mathbb{P}(E_t = e_t | X_t)}_{\text{sensor model}} \cdot \underbrace{\max_{x_{t-1}} \mathbb{P}(X_t | X_{t-1} = x_{t-1})}_{\text{transition model}} \cdot \underbrace{\max_{x_1, \dots, x_{t-2}} \mathbb{P}(X_{1:t-2}^{\bar{x}}, X_{t-1} = x_{t-1} | E_{1:t-1}^{\bar{e}})}_{=: \mathbf{m}_{1:t-1}(x_{t-1})}
 \end{aligned}$$

$\mathbf{m}_{1:t}(i)$ gives the maximal **probability** that the **most likely** path up to t leads to state $X_t = i$.

Note that we can leave out the α , since we're only interested in the maximum.

Example 2.17. For the sequence $[T, T, F, T, T]$:



bold arrows: best predecessor measured by “best preceding sequence probability \times transition probability”

The Viterbi Algorithm

Definition 2.18. The **Viterbi algorithm** now proceeds as follows:

```
function VITERBI( $\langle e_1, \dots, e_t \rangle, \mathbb{P}(X_0)$ )  
   $m := \mathbb{P}(X_0)$   
   $\text{prev} := \langle \rangle$  /*  $m_{1:i}$  */  
  for  $i = 1, \dots, t$  do /* the most likely predecessor of each possible  $x_i$  */  
     $m' := \max_{x_{i-1}} (\mathbb{P}(E_i = e_i | X_i) \cdot \mathbb{P}(X_i | X_{i-1} = x_{i-1}) \cdot m_{x_{i-1}})$   
     $\text{prev}_{i-1} := \operatorname{argmax}_{x_{i-1}} (\mathbb{P}(E_i = e_i | X_i) \cdot \mathbb{P}(X_i | X_{i-1} = x_{i-1}) \cdot m_{x_{i-1}})$   
     $m \leftarrow m'$   
   $P := \langle 0, 0, \dots, \operatorname{argmax}_{(x \in \text{dom}(X))} m_x \rangle$   
  for  $i = t - 1, \dots, 0$  do  
     $P_i := \text{prev}_{i, P_{i+1}}$   
  return  $P$ 
```

Observation 2.19. Viterbi has *linear time complexity* and *linear space complexity* (needs to keep the most likely sequence leading to each state).

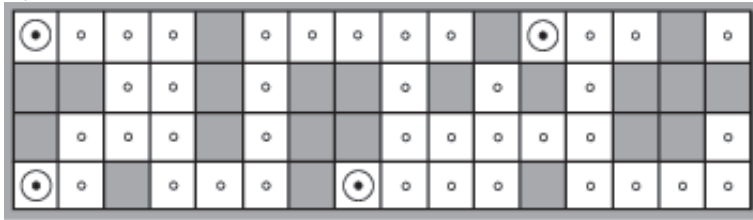
25.3 Hidden Markov Models – Extended Example

Example: Robot Localization using Common Sense

Example 3.1 (Robot Localization in a Maze). A robot has four sonar sensors that tell it about obstacles in four directions: N, S, W, E.

We write the result where the sensor that detects obstacles in the north, south, and east as N S E.

We filter out the impossible states:



a) Possible robot locations after $e_1 = \text{N S W}$

Remark 3.2. This only works for perfect sensors.
What if our sensors are imperfect?

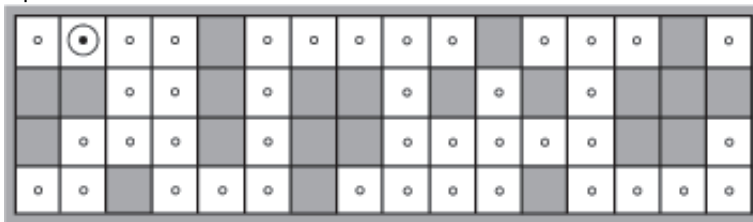
(else no impossible states)

Example: Robot Localization using Common Sense

Example 3.3 (Robot Localization in a Maze). A robot has four sonar sensors that tell it about obstacles in four directions: N, S, W, E.

We write the result where the sensor that detects obstacles in the north, south, and east as N S E.

We filter out the impossible states:



b) Possible robot locations after $e_1 = \text{N S W}$ and $e_2 = \text{N S}$

Remark 3.4. This only works for perfect sensors.

(else no impossible states)

What if our sensors are imperfect?

HMM Example: Robot Localization (Modeling)

Example 3.5 (HMM-based Robot Localization). We have the following setup:

- ▶ A hidden **Random variable** X_t for robot location (domain: 42 empty squares)
- ▶ Let $N(i)$ be the set of neighboring fields of the field $X_i = x_i$
- ▶ The **Transition matrix** for the **move** action (T has $42^2 = 1764$ entries)

$$P(X_{t+1} = j \mid X_t = i) = T_{ij} = \begin{cases} \frac{1}{|N(i)|} & \text{if } j \in N(i) \\ 0 & \text{else} \end{cases}$$

- ▶ We do not know where the robot starts: $P(X_0) = \frac{1}{n}$ (here $n = 42$)
- ▶ **Evidence variable** E_t : four bit presence/absence of obstacles in N, S, W, E. Let d_{it} be the number of wrong bits and ϵ the **error rate** of the sensor. Then

$$P(E_t = e_t \mid X_t = i) = O_{t,ii} = (1 - \epsilon)^{4-d_{it}} \cdot \epsilon^{d_{it}}$$

(We assume the sensors are independent)

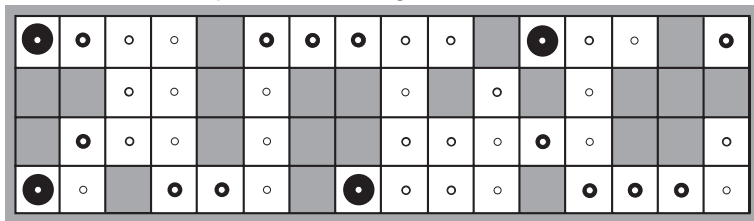
For example, the probability that the sensor on a square with obstacles in north and south would produce N S E is $(1 - \epsilon)^3 \cdot \epsilon^1$.

We can now use **filtering** for localization, **smoothing** to determine e.g. the starting location, and the **Viterbi algorithm** to find out how the robot got to where it is now.

HMM Example: Robot Localization

We use HMM filtering equation $f_{1:t+1} = \alpha \cdot O_{t+1} T^t f_{1:t}$ to compute posterior distribution over locations.
(i.e. robot localization)

Example 3.6. Redoing ???, with $\epsilon = 0.2$.



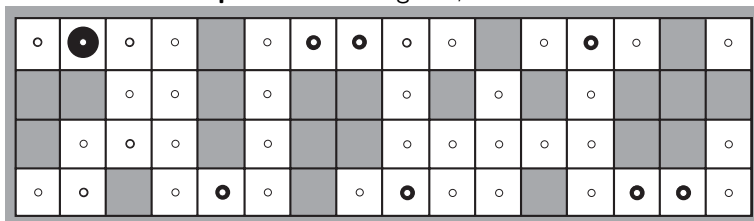
a) Posterior distribution over robot location after $E_1 = \text{NSW}$

Still the same locations as in the “perfect sensing” case, but now other locations have non-zero probability.

HMM Example: Robot Localization

We use HMM filtering equation $f_{1:t+1} = \alpha \cdot O_{t+1} T^t f_{1:t}$ to compute posterior distribution over locations.
(i.e. robot localization)

Example 3.7. Redoing ???, with $\epsilon = 0.2$.



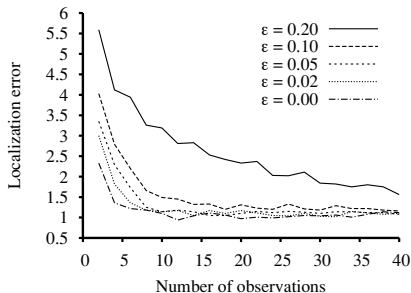
b) Posterior distribution over robot location after $E_1 = \text{N S W}$ and $E_2 = \text{N S}$

Still the same locations as in the “perfect sensing” case, but now other locations have non-zero probability.

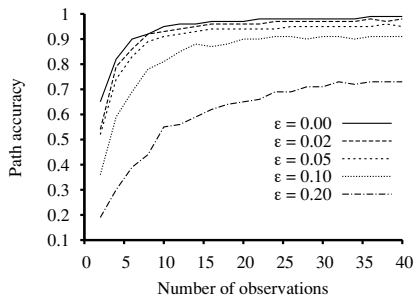
HMM Example: Further Inference Applications

Idea: We can use **smoothing**: $b_{k+1:t} = \text{TO}_{k+1} b_{k+2:t}$ to find out where it started and the **Viterbi algorithm** to find the **most likely path** it took.

Example 3.8. Performance of HMM localization vs. observation length (various error rates ϵ)



Localization error (Manhattan distance from true location)



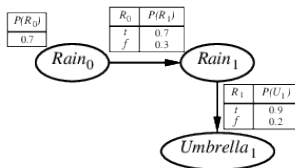
Viterbi path accuracy (fraction of correct states on Viterbi path)

25.4 Dynamic Bayesian Networks

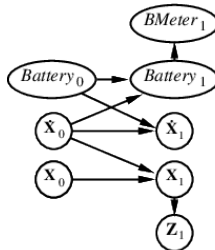
Dynamic Bayesian networks

- **Definition 4.1.** A Bayesian network \mathcal{D} is called **dynamic** (a **DBN**), iff its random variables are indexed by a time structure. We assume that \mathcal{D} is
 - **time sliced**, i.e. that the time slices \mathcal{D}_t – the subgraphs of t -indexed random variables and the edges between them – are **isomorphic**.
 - a stationary **Markov chain**, i.e. that variables X_t can only have parents in \mathcal{D}_t and \mathcal{D}_{t-1} .
- X_t, E_t contain arbitrarily many variables in a replicated Bayesian network.
- **Example 4.2.**

Umbrellas



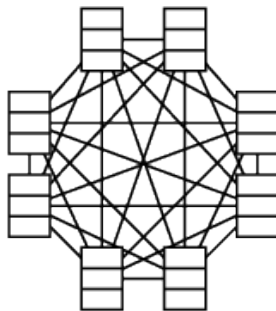
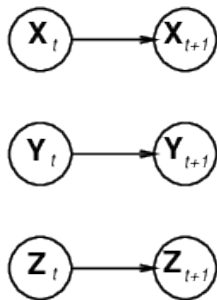
Robot Motion



► Observation 4.3.

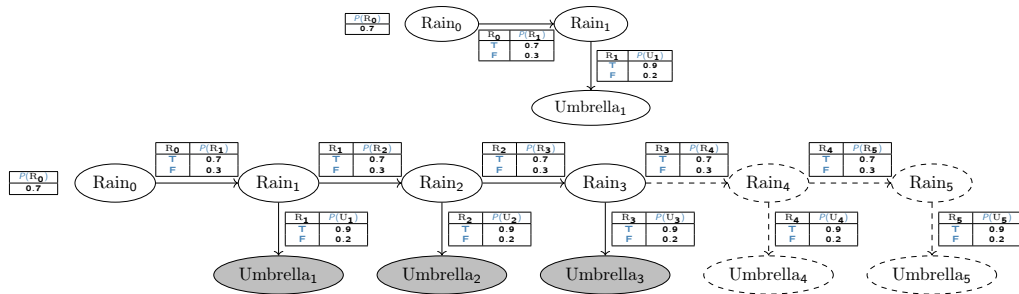
- Every *HMM* is a single-variable *DBN*.
- Every *DBN* can be turned into an *HMM*.
(combine variables into tuple \Rightarrow lose information about dependencies)
- *DBNs* have sparse dependencies \leadsto exponentially fewer parameters;

(trivially)
(combine variables into tuple \Rightarrow lose information about dependencies)



- **Example 4.4 (Sparse Dependencies).** With 20 Boolean *state variables*, three *parents* each, a *DBN* has $20 \cdot 2^3 = 160$ parameters, the corresponding *HMM* has $2^{20} \cdot 2^{20} \approx 10^{12}$.

- **Definition 4.5 (Naive method).** **Unroll** the network and run any exact **algorithm**.



- **Problem:** Inference cost for each update grows with t .
- **Definition 4.6.** **Rollup filtering:** add slice $t + 1$, “sum out” slice t using **variable elimination**.
- **Observation:** Largest factor is $\mathcal{O}(d^{n+1})$, update cost $\mathcal{O}(d^{n+2})$, where d is the maximal domain size.
- **Note:** Much better than the **HMM** update cost of $\mathcal{O}(d^{2n})$

Summary

- ▶ Temporal probability models use state and evidence variables replicated over time.
- ▶ Markov property and stationarity assumption, so we need both
 - ▶ a transition model and $P(X_t|X_{t-1})$
 - ▶ a sensor model $P(E_t|X_t)$.
- ▶ Tasks are filtering, prediction, smoothing, most likely sequence; (all done recursively with constant cost per time step)
- ▶ Hidden Markov models have a single discrete state variable; (used for speech recognition)
- ▶ DBNs subsume HMMs, exact update intractable.

Chapter 26

Making Complex Decisions

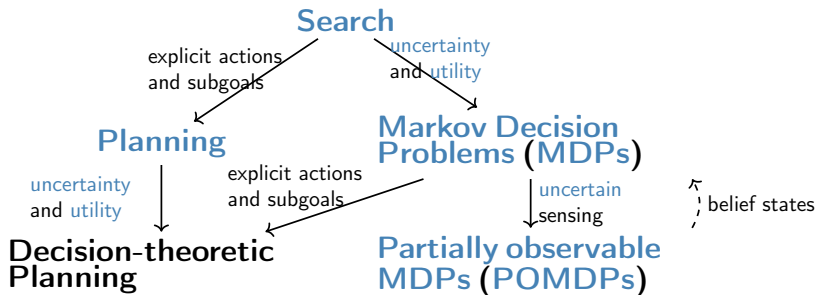
We will now combine the ideas of stochastic process with that of acting based on **maximizing expected utility**:

- ▶ Markov decision processes (MDPs) for **sequential environments**.
- ▶ Value/policy iteration for computing utilities in MDPs.
- ▶ Partially observable MDP (POMDPs).
- ▶ Decision theoretic agents for POMDPs.

26.1 Sequential Decision Problems

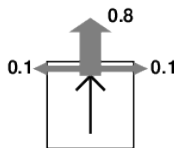
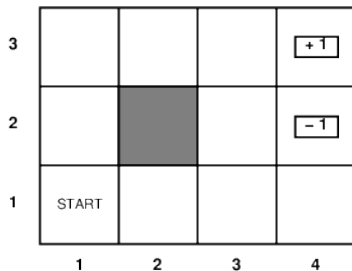
Sequential Decision Problems

- ▶ **Definition 1.1.** In **sequential decision problems**, the **agent's utility** depends on a sequence of **decisions** (or their result **states**).
- ▶ **Definition 1.2.** **Utility functions** on **action** sequences are often expressed in terms of **immediate rewards** that are incurred upon reaching a (single) **state**.
- ▶ **Methods:** depend on the **environment**:
 - ▶ If it is **fully observable** \leadsto **Markov decision process (MDPs)**
 - ▶ else \leadsto **partially observable MDP (POMDP)**.
- ▶ Sequential decision problems incorporate **utilities**, **uncertainty**, and sensing.
- ▶ **Preview:** Search problems and **planning tasks** are special cases.



Markov Decision Problem: Running Example

- **Example 1.3 (Running Example: The 4x3 World).** A (fully observable) 4×3 environment with non-deterministic actions:



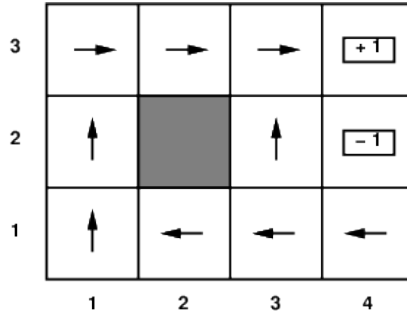
- States $s \in \mathcal{S}$, actions $a \in \mathcal{A}_s$.
- Transition model: $P(s' \mid s, a) \triangleq$ probability that a in s leads to s' .
- reward function:

$$R(s) := \begin{cases} -0.04 & \text{if (small penalty) for nonterminal states} \\ \pm 1 & \text{if for terminal states} \end{cases}$$

- ▶ **Motivation:** Let us (for now) consider sequential decision problems in a fully observable, stochastic environment with a Markovian transition model on a finite set of states and an additive reward function.
(We will switch to partially observable ones later)
- ▶ **Definition 1.4.** A Markov decision process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, s_0, R \rangle$ consists of
 - ▶ a set of \mathcal{S} of states (with initial state $s_0 \in \mathcal{S}$),
 - ▶ for every state s , a set of actions \mathcal{A}_s .
 - ▶ a transition model $\mathcal{T}(s, a) = \mathbb{P}(\mathcal{S}|s, a)$, and
 - ▶ a reward function $R: \mathcal{S} \rightarrow \mathbb{R}$; we call $R(s)$ a reward.
- ▶ **Idea:** We use the rewards as a utility function: The goal is to choose actions such that the expected cumulative rewards for the “foreseeable future” is maximized
 \Rightarrow need to take future actions and future states into account

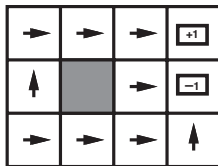
Solving MDPs

- ▶ In MDPs, the aim is to find an optimal policy $\pi(s)$, which tells us the best action for every possible state s . (because we can't predict where we might end up, we need to consider all states)
- ▶ **Definition 1.5.** A policy π for an MDP is a function mapping each state s to an action $a \in \mathcal{A}_s$. An optimal policy is a policy that maximizes the expected total rewards. (for some notion of "total"...)
- ▶ **Example 1.6.** Optimal policy when state penalty $R(s)$ is 0.04:

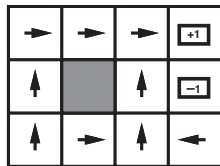


Note: When you run against a wall, you stay in your square.

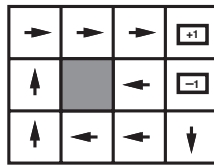
- **Example 1.7.** Optimal policy depends on the reward function $R(s)$.



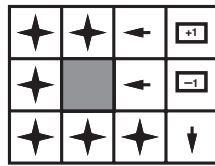
$$R(s) < -1.6284$$



$$-0.4278 < R(s) < -0.0850$$



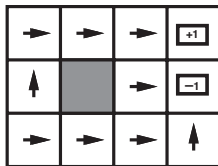
$$-0.0221 < R(s) < 0$$



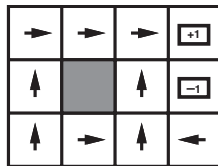
$$R(s) > 0$$

- **Question:** Explain what you see in a qualitative manner!

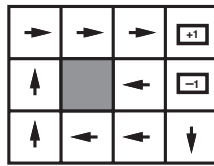
- **Example 1.8.** Optimal policy depends on the reward function $R(s)$.



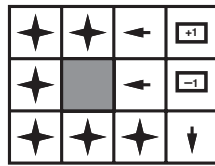
$$R(s) < -1.6284$$



$$-0.4278 < R(s) < -0.0850$$



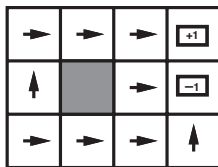
$$-0.0221 < R(s) < 0$$



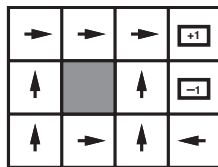
$$R(s) > 0$$

- **Question:** Explain what you see in a qualitative manner!
- **Answer:** Careful risk/reward balancing is characteristic of MDPs.
1. $-\infty \leq R(s) \leq -1.6284 \rightsquigarrow$ Life is so painful that agent heads for the next exit.

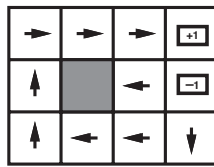
- **Example 1.9.** Optimal policy depends on the reward function $R(s)$.



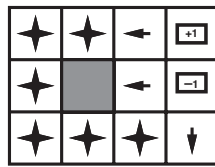
$$R(s) < -1.6284$$



$$-0.4278 < R(s) < -0.0850$$



$$-0.0221 < R(s) < 0$$



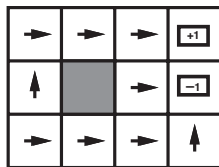
$$R(s) > 0$$

- **Question:** Explain what you see in a qualitative manner!

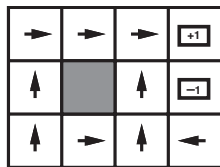
- **Answer:** Careful risk/reward balancing is characteristic of MDPs.

1. $-\infty \leq R(s) \leq -1.6284 \rightsquigarrow$ Life is so painful that agent heads for the next exit.
2. $-0.4278 \leq R(s) \leq -0.0850$, life is quite unpleasant; the agent takes the shortest route to the +1 state and is willing to risk falling into the -1 state by accident. In particular, the agent takes the shortcut from (3,1).

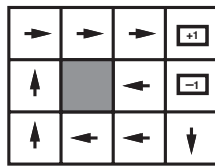
- **Example 1.10.** Optimal policy depends on the reward function $R(s)$.



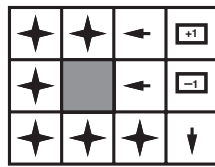
$$R(s) < -1.6284$$



$$-0.4278 < R(s) < -0.0850$$



$$-0.0221 < R(s) < 0$$



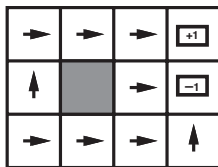
$$R(s) > 0$$

- **Question:** Explain what you see in a qualitative manner!

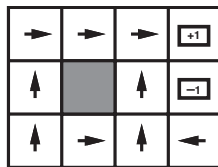
- **Answer:** Careful risk/reward balancing is characteristic of MDPs.

1. $-\infty \leq R(s) \leq -1.6284 \rightsquigarrow$ Life is so painful that agent heads for the next exit.
2. $-0.4278 \leq R(s) \leq -0.0850$, life is quite unpleasant; the agent takes the shortest route to the +1 state and is willing to risk falling into the -1 state by accident. In particular, the agent takes the shortcut from (3,1).
3. Life is slightly dreary ($-0.0221 < R(s) < 0$) \rightsquigarrow take no risks at all. In (4,1) and (3,2) head directly away from the -1 \rightsquigarrow cannot fall in by accident.

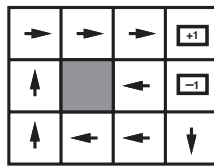
- **Example 1.11.** Optimal policy depends on the reward function $R(s)$.



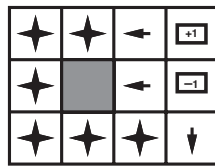
$$R(s) < -1.6284$$



$$-0.4278 < R(s) < -0.0850$$



$$-0.0221 < R(s) < 0$$



$$R(s) > 0$$

- **Question:** Explain what you see in a qualitative manner!

- **Answer:** Careful risk/reward balancing is characteristic of MDPs.

1. $-\infty \leq R(s) \leq -1.6284 \leadsto$ Life is so painful that agent heads for the next exit.
2. $-0.4278 \leq R(s) \leq -0.0850$, life is quite unpleasant; the agent takes the shortest route to the +1 state and is willing to risk falling into the -1 state by accident. In particular, the agent takes the shortcut from (3,1).
3. Life is slightly dreary ($-0.0221 < R(s) < 0$) \leadsto take no risks at all. In (4,1) and (3,2) head directly away from the -1 \leadsto cannot fall in by accident.
4. If $R(s) > 0$, then life is positively enjoyable \leadsto avoid both exits \leadsto reap infinite rewards.

26.2 Utilities over Time

Utility of state sequences

Why rewards?

- ▶ **Recall:** We cannot observe/assess utility functions, only preferences \leadsto induce utility functions from rational preferences
- ▶ **Problem:** In MDPs we need to understand preferences between sequences of states.
- ▶ **Definition 2.1.** We call preferences on reward sequences **stationary**, iff

$$[r, r_0, r_1, r_2, \dots] \succ [r, r'_0, r'_1, r'_2, \dots] \Leftrightarrow [r_0, r_1, r_2, \dots] \succ [r'_0, r'_1, r'_2, \dots]$$

(i.e. rewards over time are “independent” of each other)

▶ Good news:

Theorem 2.2. For stationary preferences, there are only two ways to combine rewards over time.

- ▶ **additive rewards:** $U([s_0, s_1, s_2, \dots]) = R(s_0) + R(s_1) + R(s_2) + \dots$
- ▶ **discounted rewards:** $U([s_0, s_1, s_2, \dots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots$ where $0 \leq \gamma \leq 1$ is called **discount factor**.

\Rightarrow we can reduce utilities over time to rewards on individual states

Utilities of State Sequences

Problem: Infinite lifetimes \leadsto additive rewards may become infinite.

Possible Solutions:

1. **Finite horizon:** terminate utility computation at a fixed time T

$$U([s_0, \dots, s_\infty]) = R(s_0) + \dots + R(s_T)$$

\leadsto nonstationary policy: $\pi(s)$ depends on time left.

2. If there are **absorbing states**: for any policy π agent eventually “dies” with probability 1 \leadsto expected utility of every state is finite.
3. **Discounting**: assuming $\gamma < 1$, $R(s) \leq R_{\max}$,

$$U([s_0, s_1, \dots]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \sum_{t=0}^{\infty} \gamma^t R_{\max} = R_{\max} / (1 - \gamma)$$

Smaller $\gamma \leadsto$ shorter horizon.

We will only consider discounted rewards in this course

Why discounted rewards?

Discounted rewards are both convenient and (often) realistic:

- ▶ stationary preferences imply (additive rewards or) discounted rewards anyway,
- ▶ discounted rewards lead to finite utilities for (potentially) infinite sequences of states (we can compute expected utilities for the entire future),
- ▶ discounted rewards lead to stationary policies, which are easier to compute and often more adequate (unless we know that remaining time matters),
- ▶ discounted rewards mean we value *short-term gains* over *long-term gains* (all else being equal), which is often realistic (e.g. the same amount of money gained *early* gives more opportunity to spend/invest \Rightarrow potentially more utility in the long run)
- ▶ we can interpret the discount factor as a measure of *uncertainty about future rewards* \Rightarrow more robust measure in uncertain environments.

Utility of States

Remember: Given a sequence of **states** $S = s_0, s_1, s_2, \dots$, and a **discount factor** $0 \leq \gamma < 1$, the **utility** of the sequence is

$$U(S) = \sum_{t=0}^{\infty} \gamma^t R(s_t)$$

Definition 2.3. Given a **policy** π and a starting **state** s_0 , let $S_{s_0}^{\pi}$ be the **random variable** giving the sequence of **states** resulting from executing π at every **state** starting at s_0 . (Since the environment is stochastic, we don't know the exact sequence.)

Then the **expected utility** obtained by executing π starting in s_0 is given by

$$U^{\pi}(s_0) := \mathbb{E}U(S_{s_0}^{\pi}).$$

We define the **optimal policy** $\pi_{s_0}^* := \operatorname{argmax}_{\pi} U^{\pi}(s_0)$.

Utility of States

Remember: Given a sequence of **states** $S = s_0, s_1, s_2, \dots$, and a **discount factor** $0 \leq \gamma < 1$, the **utility** of the sequence is

$$U(S) = \sum_{t=0}^{\infty} \gamma^t R(s_t)$$

Definition 2.4. Given a **policy** π and a starting **state** s_0 , let $S_{s_0}^{\pi}$ be the **random variable** giving the sequence of **states** resulting from executing π at every **state** starting at s_0 . (Since the environment is stochastic, we don't know the exact sequence.)

Then the **expected utility** obtained by executing π starting in s_0 is given by

$$U^{\pi}(s_0) := \mathbb{E}U(S_{s_0}^{\pi}).$$

We define the **optimal policy** $\pi_{s_0}^* := \operatorname{argmax}_{\pi} U^{\pi}(s_0)$.

Note: This is perfectly well-defined, but almost always computationally infeasible. (requires considering all possible (potentially infinite) sequences of states)

Observation 2.5. $\pi_{s_0}^*$ is independent of the *state* s_0 .

Proof sketch: If π_a^* and π_b^* reach point c , then there is no reason to disagree from that point on – or with π_c^* , and we expect *optimal policies* to “meet at some *state*” sooner or later.

⚠ ?? does not hold for *finite horizon policies*!

Observation 2.8. $\pi_{s_0}^*$ is independent of the *state* s_0 .

Proof sketch: If π_a^* and π_b^* reach point c , then there is no reason to disagree from that point on – or with π_c^* , and we expect *optimal policies* to “meet at some *state*” sooner or later.

⚠ ?? does not hold for *finite horizon policies*!

Definition 2.9. We call $\pi^* := \pi_s^*$ for some s the *optimal policy*.

Definition 2.10. The *utility* $U(s)$ of a *state* s is $U^{\pi^*}(s)$.

Utility of States (continued)

Observation 2.11. $\pi_{s_0}^*$ is independent of the state s_0 .

Proof sketch: If π_a^* and π_b^* reach point c , then there is no reason to disagree from that point on – or with π_c^* , and we expect optimal policies to “meet at some state” sooner or later.

⚠ ?? does not hold for finite horizon policies!

Definition 2.12. We call $\pi^* := \pi_s^*$ for some s the optimal policy.

Definition 2.13. The utility $U(s)$ of a state s is $U^{\pi^*}(s)$.

Remark: $R(s) \hat{=}$ “immediate reward”, whereas $U \hat{=}$ “long-term reward”.

Given the utilities of the states, choosing the best action is just MEU: maximize the expected utility of the immediate successor states

$$\pi^*(s) = \operatorname{argmax}_{a \in A(s)} \left(\sum_{s'} P(s' \mid s, a) \cdot U(s') \right)$$

\Rightarrow given the “true” utilities, we can compute the optimal policy and vice versa.

Utility of States (continued)

► Example 2.14 (Running Example Continued).

Expected Utility

3	0.812	0.868	0.912	<div style="border: 1px solid black; padding: 2px;">+1</div>
2	0.762		0.660	<div style="border: 1px solid black; padding: 2px;">-1</div>
1	0.705	0.655	0.611	0.388
	1	2	3	4

Optimal Policy

3	→	→	→	<div style="border: 1px solid black; padding: 2px;">+1</div>
2	↑		↑	<div style="border: 1px solid black; padding: 2px;">-1</div>
1	↑	←	←	←
	1	2	3	4

► **Question:** Why do we go left in (3, 1) and not up?

(follow the utility)

26.3 Value/Policy Iteration

Dynamic programming: the Bellman equation

- **Problem:** We have defined $U(s)$ via the optimal policy: $U(s) := U^{\pi^*}(s)$, but how to compute it without knowing π^* ?
- **Observation:** A simple relationship among utilities of neighboring states:

expected sum of rewards = current reward + $\gamma \cdot$ exp. reward sum after best action

- **Theorem 3.1 (Bellman equation (1957)).**

$$U(s) = R(s) + \gamma \cdot \max_{a \in A(s)} \sum_{s'} U(s') \cdot P(s' \mid s, a)$$

We call this equation the *Bellman equation*

- **Example 3.2.** $U(1,1) = -0.04$
+ $\gamma \max\{0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1),$
 $0.9U(1,1) + 0.1U(1,2)$
 $0.9U(1,1) + 0.1U(2,1)$
 $0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\}$

up
left
down
right

- **Problem:** One equation/state $\leadsto n$ nonlinear (\max isn't) equations in n unknowns.
 \leadsto cannot use linear algebra techniques for solving them.

Value Iteration Algorithm

► **Idea:** We use a simple **iteration** scheme to find a **fixpoint**:

1. start with arbitrary utility values,
2. update to make them locally consistent with the Bellman equation,
3. everywhere locally consistent \leadsto global optimality.

► **Definition 3.3.** The **value iteration algorithm** for **utility** utility function is given by

function VALUE-ITERATION (mdp, ϵ) **returns** a utility fn.

inputs: mdp, an MDP with states S , actions $A(s)$, transition model $P(s' \mid s, a)$,
rewards $R(s)$, and discount γ

ϵ , the maximum error allowed **in** the utility of any state

local variables: U , U' , vectors of utilities **for** states **in** S , initially zero

δ , the maximum change **in** the utility of any state **in** an iteration

repeat

$U := U'$; $\delta := 0$

for each state s **in** S **do**

$U'[s] := R(s) + \gamma \cdot \max_{a \in A(s)} (\sum_{s'} U[s'] \cdot P(s' \mid s, a))$

if $|U'[s] - U[s]| > \delta$ **then** $\delta := |U'[s] - U[s]|$

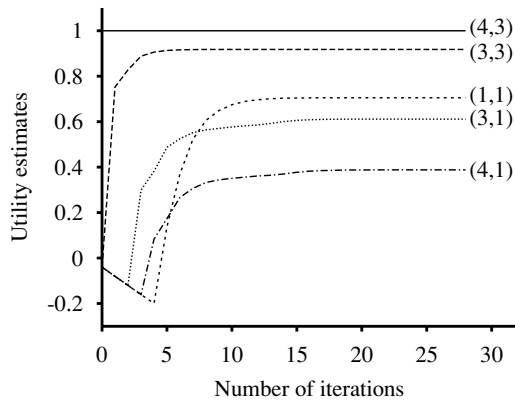
until $\delta < \epsilon(1 - \gamma)/\gamma$

return U

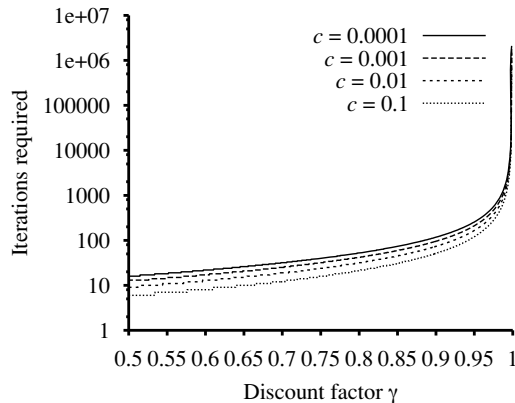
► **Remark:** Retrieve the optimal policy with $\pi[s] := \operatorname{argmax}_{a \in A(s)} (\sum_{s'} U[s'] \cdot P(s' \mid s, a))$

Value Iteration Algorithm (Example)

► Example 3.4 (Iteration on 4x3).



(where $\varepsilon = c \cdot R_{max}$)



- ▶ **Definition 3.5.** The **maximum norm** is defined as $\|U\| = \max_s |U(s)|$, so $\|U - V\| = \text{maximum difference between } U \text{ and } V$.
- ▶ Let U^t and U^{t+1} be successive approximations to the true utility U during **value iteration**.
- ▶ **Theorem 3.6.** For any two approximations U^t and V^t

$$\|U^{t+1} - V^{t+1}\| \leq \gamma \|U^t - V^t\|$$

*I.e., any distinct approximations get closer to each other over time
In particular, any approximation gets closer to the true U over time
 \Rightarrow **value iteration** converges to a unique, stable, optimal solution.*

- ▶ **Theorem 3.7.** If $\|U^{t+1} - U^t\| < \epsilon$, then $\|U^{t+1} - U\| < 2\epsilon\gamma / (1 - \gamma)$
(once the change in U^t becomes small, we are almost done.)
- ▶ **Remark:** The **policy** resulting from U^t may be optimal long before the utilities convergence!

- ▶ **Recap:** Value iteration computes utilities \leadsto optimal policy by MEU.
- ▶ This even works if the utility estimate is inaccurate. (\leadsto policy loss small)
- ▶ **Idea:** Search for optimal policy and utility values simultaneously [Howard:dpmp60]: Iterate
 - ▶ **policy evaluation:** given policy π_i , calculate $U_i = U^{\pi_i}$, the utility of each state were π_i to be executed.
 - ▶ **policy improvement:** calculate a new MEU policy π_{i+1} using 1 lookaheadTerminate if policy improvement yields no change in computed utilities.
- ▶ **Observation 3.8.** Upon termination U_i is a *fixpoint* of Bellman update \leadsto Solution to Bellman equation $\leadsto \pi_i$ is an *optimal policy*.
- ▶ **Observation 3.9.** Policy improvement improves policy and policy space is *finite* \leadsto termination.

Policy Iteration Algorithm

► **Definition 3.10.** The **policy iteration algorithm** is given by the following **pseudocode**:

```
function POLICY-ITERATION(mdp) returns a policy
inputs: mdp, and MDP with states  $S$ , actions  $A(s)$ , transition model  $P(s' \mid s, a)$ 
local variables:  $U$  a vector of utilities for states in  $S$ , initially zero
                 $\pi$  a policy indexed by state, initially random,
repeat
     $U := \text{POLICY-EVALUATION}(\pi, U, \text{mdp})$ 
    unchanged? := true
    foreach state  $s$  in  $X$  do
        if  $\max_{a \in A(s)} (\sum_{s'} P(s' \mid s, a) \cdot U(s')) > \sum_{s'} P(s' \mid s, \pi[s]) \cdot U(s')$  then do
             $\pi[s] := \operatorname{argmax}_{b \in A(s)} (\sum_{s'} P(s' \mid s, b) \cdot U(s'))$ 
        unchanged? := false
until unchanged?
return  $\pi$ 
```

Policy Evaluation

► **Problem:** How to **implement** the POLICY–EVALUATION **algorithm**?

► **Solution:** To compute utilities given a fixed π : For all s we have

$$U(s) = R(s) + \gamma \left(\sum_{s'} U(s') \cdot P(s' \mid s, \pi(s)) \right)$$

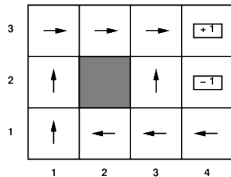
(i.e. Bellman equation with the maximum replaced by the current policy π)

► **Example 3.11 (Simplified Bellman Equations for π).**

$$U_i(1,1) = -0.04 + 0.8U_i(1,2) + 0.1U_i(1,1) + 0.1U_i(2,1)$$

$$U_i(1,2) = -0.04 + 0.8U_i(1,3) + 0.1U_i(1,2)$$

\vdots



► **Observation 3.12.** n simultaneous **linear equations** in n **unknowns**, solve in $\mathcal{O}(n^3)$ with standard **linear algebra** methods.

- ▶ **Value iteration** requires many iterations, but each one is cheap.
- ▶ **Policy iteration** often converges in few iterations, but each is expensive.
- ▶ **Idea:** Use a few steps of **value iteration** (but with π fixed), starting from the **value function** produced the last time to produce an approximate value determination step.
- ▶ Often converges much faster than pure VI or PI.
- ▶ Leads to much more general **algorithms** where Bellman value updates and Howard policy updates can be performed locally in any order.
- ▶ **Remark:** **Reinforcement learning algorithms** operate by performing such updates based on the observed transitions made in an initially unknown environment.

26.4 Partially Observable MDPs

Partial Observability

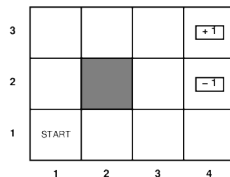
- **Definition 4.1.** A **partially observable MDP** (a **POMDP** for short) is a **MDP** together with an **observation model** O that has the **sensor Markov property** and is **stationary**: $O(s, e) = P(e \mid s)$.
- **Example 4.2 (Noisy 4x3 World).**

Add a partial and/or noisy sensor.

e.g. count number of adjacent walls
with 0.1 error

If sensor reports 1, we are in (3, ?)

$(1 \leq w \leq 2)$
(noise)
(probably)



Partial Observability

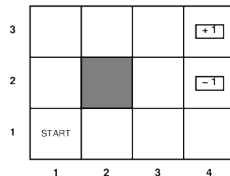
► **Definition 4.4.** A **partially observable MDP** (a **POMDP** for short) is a **MDP** together with an **observation model** O that has the **sensor Markov property** and is **stationary**: $O(s, e) = P(e \mid s)$.

► **Example 4.5 (Noisy 4x3 World).**

Add a partial and/or noisy sensor.

e.g. count number of adjacent walls $(1 \leq w \leq 2)$
with 0.1 error (noise)

If sensor reports 1, we are in $(3, ?)$ (probably)



► **Problem:** Agent does not know which state it is in \leadsto makes no sense to talk about **policy** $\pi(s)$!

Partial Observability

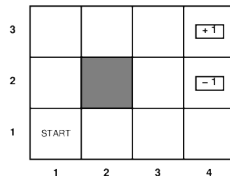
► **Definition 4.7.** A **partially observable MDP** (a **POMDP** for short) is a **MDP** together with an **observation model** O that has the **sensor Markov property** and is **stationary**: $O(s, e) = P(e \mid s)$.

► **Example 4.8 (Noisy 4x3 World).**

Add a partial and/or noisy sensor.

e.g. count number of adjacent walls $(1 \leq w \leq 2)$
with 0.1 error (noise)

If sensor reports 1, we are in $(3, ?)$ (probably)



► **Problem:** Agent does not know which state it is in \leadsto makes no sense to talk about **policy** $\pi(s)$!

► **Theorem 4.9 (Astrom 1965).** The **optimal policy** in a **POMDP** is a function $\pi(b)$ where b is the **belief state** (probability distribution over states).

► **Idea:** Convert a **POMDP** into an **MDP** in **belief state** space, where $\mathcal{T}(b, a, b')$ is the probability that the new **belief state** is b' given that the current **belief state** is b and the **agent** does a . I.e., essentially a filtering update step.

POMDP: Filtering at the Belief State Level

- ▶ **Recap:** Filtering updates the belief state for new evidence.
- ▶ For POMDPs, we also need to consider actions. (but the effect is the same)
- ▶ If b is the previous belief state and agent does action $A = a$ and then perceives $E = e$, then the new belief state is

$$b' = \alpha(\mathbb{P}(E = e|s') \cdot (\sum_s \mathbb{P}(s'|S = s, A = a) \cdot b(s)))$$

We write $b' = \text{FORWARD}(b, a, e)$ in analogy to recursive state estimation.

- ▶ **Fundamental Insight for POMDPs:** The optimal action only depends on the agent's current belief state. (good, it does not know the state!)
- ▶ **Consequence:** The optimal policy can be written as a function $\pi^*(b)$ from belief states to actions.
- ▶ **Definition 4.10.** The POMDP decision cycle is to iterate over
 1. Given the current belief state b , execute the action $a = \pi^*(b)$
 2. Receive percept e .
 3. Set the current belief state to $\text{FORWARD}(b, a, e)$ and repeat.
- ▶ **Intuition:** POMDP decision cycle is search in belief state space.

- ▶ **Recap:** The POMDP decision cycle is search in belief state space.
- ▶ **Observation 4.11.** *Actions change the belief state, not just the (physical) state.*
- ▶ **Thus** POMDP solutions automatically include information gathering behavior.
- ▶ **Problem:** The belief state is continuous: If there are n states, b is an n -dimensional real-valued vector.
- ▶ **Example 4.12.** The belief state of the 4x3 world is a 11 dimensional continuous space. (11 states)
- ▶ **Theorem 4.13.** *Solving POMDPs is very hard!* (actually, PSPACE hard)
- ▶ **In particular,** none of the algorithms we have learned applies. (discreteness assumption)
- ▶ The real world is a POMDP (with initially unknown transition model T and sensor model O)

Reducing POMDPs to Belief-State MDPs I

- ▶ **Idea:** Calculating the probability that an agent in belief state b reaches belief state b' after executing action a .
 - ▶ if we knew the action and the subsequent percept e , then $b' = \text{FORWARD}(b, a, e)$. (deterministic update to the belief state)
 - ▶ but we don't, since b' depends on e . (let's calculate $P(e \mid a, b)$)
- ▶ **Idea:** To compute $P(e \mid a, b)$ — the probability that e is perceived after executing a in belief state b — sum up over all actual states the agent might reach:

$$\begin{aligned}P(e \mid a, b) &= \sum_{s'} P(e \mid a, s', b) \cdot P(s' \mid a, b) \\&= \sum_{s'} P(e \mid s') \cdot P(s' \mid a, b) \\&= \sum_{s'} P(e \mid s') \cdot \left(\sum_s P(s' \mid s, a), b(s) \right)\end{aligned}$$

Reducing POMDPs to Belief-State MDPs II

Write the **probability** of reaching b' from b , given **action** a , as $P(b' \mid b, a)$, then

$$\begin{aligned} P(b' \mid b, a) &= P(b' \mid a, b) = \sum_e P(b' \mid e, a, b) \cdot P(e \mid a, b) \\ &= \sum_e P(b' \mid e, a, b) \cdot \left(\sum_{s'} P(e \mid s') \cdot \left(\sum_s P(s' \mid s, a), b(s) \right) \right) \end{aligned}$$

where $P(b' \mid e, a, b)$ is 1 if $b' = \text{FORWARD}(b, a, e)$ and 0 otherwise.

- **Observation:** This equation defines a **transition model** for **belief state** space!
- **Idea:** We can also define a **reward function** for **belief states**:

$$\rho(b) := \sum_s b(s) \cdot R(s)$$

i.e., the **expected reward** for the actual **states** the **agent** might be in.

- ▶ Together, $P(b' \mid b, a)$ and $\rho(b)$ define an (observable) MDP on the space of belief states.
- ▶ **Theorem 4.14.** An optimal policy $\pi^*(b)$ for this MDP, is also an optimal policy for the original POMDP.
- ▶ **Upshot:** Solving a POMDP on a physical state space can be reduced to solving an MDP on the corresponding belief state space.
- ▶ **Remember:** The belief state is always observable to the agent, by definition.

Ideas towards Value-Iteration on POMDPs

- ▶ **Recap:** The value iteration algorithm from ??? computes one utility value per state.
- ▶ **Problem:** We have infinitely many belief states \leadsto be more creative!
- ▶ **Observation:** Consider an optimal policy π^*
 - ▶ applied in a specific belief state b : π^* generates an action,
 - ▶ for each subsequent percept, the belief state is updated and a new action is generated ...

For this specific b : $\pi^* \hat{=}$ a conditional plan!

- ▶ **Idea:** Think about conditional plans and how the expected utility of executing a fixed conditional plan varies with the initial belief state. (instead of optimal policies)

Definition 4.15. Given a set of percepts E and a set of actions A , a conditional plan is either an action $a \in A$, or a tuple $\langle a, E', p_1, p_2 \rangle$ such that $a \in A$, $E' \subseteq E$, and p_1, p_2 are conditional plans. It represents the strategy “First execute a , If we subsequently perceive $e \in E'$, continue with p_1 , otherwise continue with p_2 .”

The depth of a conditional plan p is the maximum number of actions in any path from p before reaching a single action plan.

Expected Utilities of Conditional Plans on Belief States

- ▶ **Observation 1:** Let p be a conditional plan and $\alpha_p(s)$ the utility of executing p in state s .
 - ▶ the expected utility of p in belief state b is $\sum_s b(s) \cdot \alpha_p(s) \hat{=} b \cdot \alpha_p$ as vectors.
 - ▶ the expected utility of a fixed conditional plan varies linearly with b
 - ▶ \leadsto the “best conditional plan to execute” corresponds to a hyperplane in belief state space.

Expected Utilities of Conditional Plans on Belief States

- ▶ **Observation 1:** Let p be a conditional plan and $\alpha_p(s)$ the utility of executing p in state s .
 - ▶ the expected utility of p in belief state b is $\sum_s b(s) \cdot \alpha_p(s) \hat{=}$ $b \cdot \alpha_p$ as vectors.
 - ▶ the expected utility of a fixed conditional plan varies linearly with b
 - ▶ \leadsto the “best conditional plan to execute” corresponds to a hyperplane in belief state space.
- ▶ **Observation 2:** We can replace the *original* actions by conditional plans on those actions! Let π^* be the subsequent optimal policy. At any given belief state b ,
 - ▶ π^* will choose to execute the conditional plan with highest expected utility
 - ▶ the expected utility of b under the π^* is the utility of that plan:

$$U(b) = U^{\pi^*}(b) = \max_b (b \cdot \alpha_p)$$

- ▶ If the optimal policy π^* chooses to execute p starting at b , then it is reasonable to expect that it might choose to execute p in belief states that are very close to b ;
- ▶ if we bound the depth of the conditional plans, then there are only finitely many such plans
- ▶ the continuous space of belief states will generally be divided into regions, each corresponding to a particular conditional plan that is optimal in that region.

Expected Utilities of Conditional Plans on Belief States

- ▶ **Observation 1:** Let p be a conditional plan and $\alpha_p(s)$ the utility of executing p in state s .
 - ▶ the expected utility of p in belief state b is $\sum_s b(s) \cdot \alpha_p(s) \hat{=}$ $b \cdot \alpha_p$ as vectors.
 - ▶ the expected utility of a fixed conditional plan varies linearly with b
 - ▶ \leadsto the “best conditional plan to execute” corresponds to a hyperplane in belief state space.
- ▶ **Observation 2:** We can replace the *original* actions by conditional plans on those actions! Let π^* be the subsequent optimal policy. At any given belief state b ,
 - ▶ π^* will choose to execute the conditional plan with highest expected utility
 - ▶ the expected utility of b under the π^* is the utility of that plan:

$$U(b) = U^{\pi^*}(b) = \max_b (b \cdot \alpha_p)$$

- ▶ If the optimal policy π^* chooses to execute p starting at b , then it is reasonable to expect that it might choose to execute p in belief states that are very close to b ;
 - ▶ if we bound the depth of the conditional plans, then there are only finitely many such plans
 - ▶ the continuous space of belief states will generally be divided into regions, each corresponding to a particular conditional plan that is optimal in that region.
- ▶ **Observation 3 (combined):** The utility function $U(b)$ on belief states, being the maximum of a collection of hyperplanes, is piecewise linear and convex.

A simple Illustrating Example I

► **Example 4.16.** A world with states S_0 and S_1 , where $R(S_0) = 0$ and $R(S_1) = 1$ and two actions:

- “Stay” stays put with probability 0.9
- “Go” switches to the other state with probability 0.9.
- The sensor reports the correct state with probability 0.6.

Obviously, the agent should “Stay” when it thinks it’s in state S_1 and “Go” when it thinks it’s in state S_0 .

- The belief state has dimension 1. (the two probabilities sum up to 1)
- Consider the one-step plans $[Stay]$ and $[Go]$ and their direct utilities:

$$\alpha_{([Stay])}(S_0) = 0.9R(S_0) + 0.1R(S_1) = 0.1$$

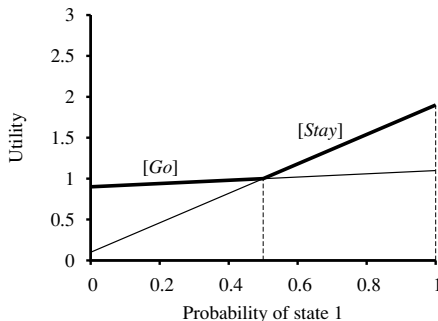
$$\alpha_{([stay])}(S_1) = 0.9R(S_1) + 0.1R(S_0) = 0.9$$

$$\alpha_{([go])}(S_0) = 0.9R(S_1) + 0.1R(S_0) = 0.9$$

$$\alpha_{([go])}(S_1) = 0.9R(S_0) + 0.1R(S_1) = 0.1$$

A simple Illustrating Example II

- ▶ Let us visualize the hyperplanes $b \cdot \alpha([Stay])$ and $b \cdot \alpha([Go])$.



- ▶ The maximum represents the utility function for the finite-horizon problem that allows just one action
- ▶ in each “piece” the optimal action is the first action of the corresponding plan.
- ▶ Here the optimal one-step policy is to “Stay” when $b(1) > 0.5$ and “Go” otherwise.

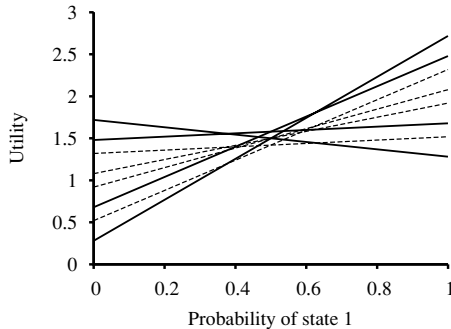
A simple Illustrating Example III

- ▶ compute the utilities for **conditional plans** of depth 2 by considering
 - ▶ each possible first action,
 - ▶ each possible subsequent **percept**, and then
 - ▶ each way of choosing a depth-1 plan to execute for each **percept**:

There are eight of depth 2:

[Stay, if $P = 0$ then Stay else Stay fi], [Stay, if $P = 0$ then Stay else Go fi], ...

A simple Illustrating Example IV

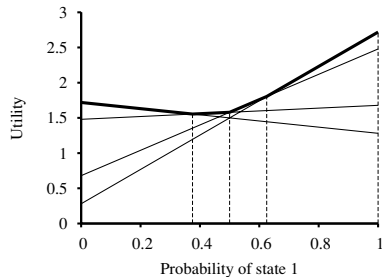


Four of them (dashed lines) are suboptimal for the whole belief space
We call them **dominated**

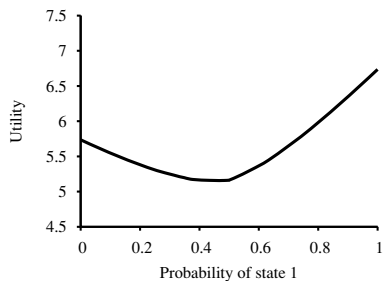
(they can be ignored)

A simple Illustrating Example V

- There are four **undominated** plans, each optimal in their region



A simple Illustrating Example VI



► **Idea:** Repeat for depth 3 and so on.

► **Theorem 4.17 (POMDP Plan Utility).** Let p be a depth- d *conditional plan* whose initial *action* is a and whose depth- $d - 1$ -subplan for *percept* e is $p.e$, then

$$\alpha_p(s) = R(s) + \gamma \left(\sum_{s'} P(s' \mid s, a) \left(\sum_e P(e \mid s') \cdot \alpha_{p.e}(s') \right) \right)$$

► This recursion naturally gives us a value iteration algorithm,

A Value Iteration Algorithm for POMDPs

Definition 4.18. The POMDP value iteration algorithm for POMDPs is given by recursively updating

$$\alpha_p(s) = R(s) + \gamma \left(\sum_{s'} P(s' \mid s, a) \left(\sum_e P(e \mid s') \cdot \alpha_{p.e}(s') \right) \right)$$

Observations: The complexity depends primarily on the generated plans:

- ▶ Given $|A|$ actions and $|E|$ possible observations, there are $|A|^{|E|^{d-1}}$ distinct depth- d plans.
- ▶ Even for the example with $d = 8$, we have 2255 (144 undominated)
- ▶ The elimination of dominated plans is essential for reducing this doubly exponential growth (but they are already constructed)

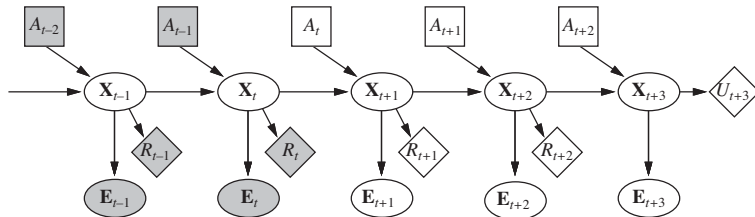
Hopelessly inefficient in practice – even the 3x4 POMDP is too hard!

26.5 Online Agents with POMDPs

- ▶ **Idea:** Let's try to use the computationally **efficient** representations (**dynamic Bayesian networks** and **decision networks**) for **POMDPs**.
- ▶ **Definition 5.1.** A **dynamic decision network (DDN)** is a **graph**-based representation of a **POMDP**, where
 - ▶ **Transition** and **sensor model** are represented as a **DBN**.
 - ▶ **Action nodes** and **utility nodes** are added as in **decision networks**.
- ▶ In a **DDN**, a filtering **algorithm** is used to incorporate each new **percept** and **action** and to update the **belief state** representation.
- ▶ Decisions are made in **DDN** by projecting forward possible action sequences and choosing the best one.
- ▶ **DDNs** – like the **DBNs** they are based on – are **factored** representations
~> typically **exponential complexity** advantages!

Structure of DDNs for POMDPs

- **DDN for POMDPs:** The generic structure of a **dynamic decision network** at time t is

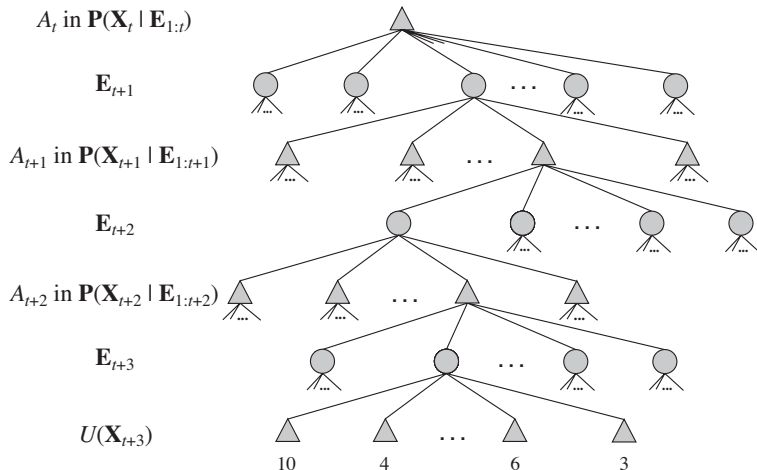


- POMDP state S_t becomes a set of **random variables** X_t
- there may be multiple **evidence variables** E_t
- **Action** at time t denoted by A_t . **agent** must choose a value for A_t .
- **Transition model**: $\mathbb{P}(X_{t+1}|X_t, A_t)$; **sensor model**: $\mathbb{P}(E_t|X_t)$.
- **Reward** functions R_t and utility U_t of **state** S_t .
- Variables with known values are gray, **rewards** for $t = 0, \dots, t+2$, but utility for $t+3$ ($\hat{=}$ **discounted sum of rest**)
- **Problem:** How do we compute with that?
- **Answer:** All **POMDP algorithms** can be adapted to **DDNs!** (only need CPTs)

Lookahead: Searching over the Possible Action Sequences

- **Idea:** Search over the tree of possible action sequences
- Part of the lookahead solution of the DDN above

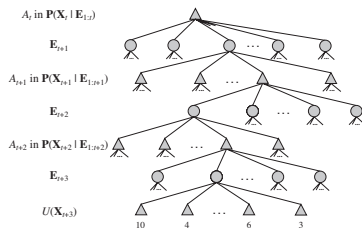
(like in game-play)
(three steps lookahead)



► circle $\hat{=}$ chance nodes
FAU triangle $\hat{=}$ belief state

(the environment decides)
(each action decision is taken)

Designing Online Agents for POMDPs



- ▶ **Belief state** at triangle computed by **filtering** with **actions/percepts** leading to it
 - ▶ for decision A_{t+i} will use **percepts** $E_{t+1:t+i}$ (even if values at time t unknown)
 - ▶ thus a **POMDP agent** automatically takes into account the **value of information** and executes **information gathering actions** where appropriate.
- ▶ **Observation:** Time complexity for exhaustive search up to depth d is $\mathcal{O}(|A|^d \cdot |E|^d)$ ($|A| \hat{=}$ number of actions, $|E| \hat{=}$ number of percepts)
- ▶ **Upshot:** Much better than POMDP value iteration with $\mathcal{O}(|A|^{|E|^{d-1}})$.
- ▶ **Empirically:** For problems in which the **discount factor** γ is not too close to 1, a shallow search is often good enough to give near-optimal decisions.

- ▶ Decision theoretic **agents** for **sequential environments**
- ▶ Building on temporal, probabilistic models/inference (dynamic Bayesian networks)
- ▶ **MDPs** for fully observable case.
- ▶ Value/Policy Iteration for **MDPs** \leadsto optimal policies.
- ▶ **POMDPs** for **partially observable** case.
- ▶ **POMDPs** $\hat{=}$ **MDP** on **belief state** space.
- ▶ The world is a **POMDP** with (initially) unknown **transition** and **sensor models**.

Part 2

Machine Learning

Chapter 27

Learning from Observations

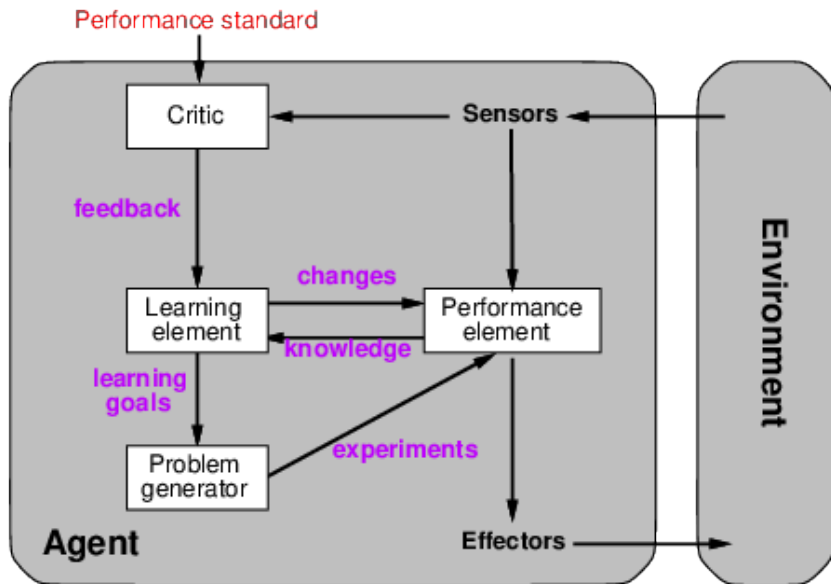
- ▶ Learning agents
- ▶ Inductive learning
- ▶ Decision tree learning
- ▶ Measuring learning performance
- ▶ Computational Learning Theory
- ▶ Linear regression and classification
- ▶ Neural Networks
- ▶ Support Vector Machines

27.1 Forms of Learning

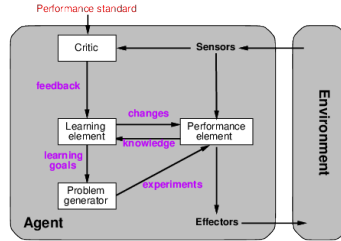
Learning (why is this a good idea)

- ▶ Learning is essential for unknown **environments**:
 - ▶ i.e., when designer lacks omniscience.
 - ▶ The world is a **POMDP** with (initially) unknown **transition** and **sensor models**.
- ▶ Learning is useful as a system construction method.
 - ▶ i.e., expose the agent to reality rather than trying to write it down
- ▶ Learning modifies the agent's decision mechanisms to improve performance.

Recap: Learning Agents



Recap: Learning Agents (continued)



- ▶ **Definition 1.1.** **Performance element** is what we called “agent” up to now.
- ▶ **Definition 1.2.** **Critic/learning element/problem generator** do the “improving”.
- ▶ **Definition 1.3.** **Performance standard** is fixed; (outside the environment)
 - ▶ We can't adjust **performance standard** to flatter own behaviour!
 - ▶ No standard *in the environment*: e.g. ordinary **chess** and suicide chess look identical.
 - ▶ Essentially, certain kinds of **percepts** are “hardwired” as good/bad (e.g., **pain, hunger**)
- ▶ **Definition 1.4.** **Learning element** may use knowledge already acquired in the **performance element**.
- ▶ **Definition 1.5.** Learning may require **experimentation** actions an agent might not normally consider such as dropping rocks from the Tower of Pisa.

- ▶ **Supervised learning:** There's an unknown **function** $f: A \rightarrow B$ called the **target function**. We do know a set of pairs $T := \{\langle a_i, f(a_i) \rangle\}$ of **examples**. The goal is to find a **hypothesis** $h \in \mathcal{H} \subseteq A \rightarrow B$ based on T , that is “approximately” equal to f .
(Most of the techniques we will consider)
- ▶ **Unsupervised learning:** Given a set of data A , find a *pattern* in the data; i.e. a function $f: A \rightarrow B$ for some predetermined B .
(Primarily *clustering/dimensionality reduction*)
- ▶ **Reinforcement learning:** The **agent** receives a reward for each action performed. The goal is to iteratively adapt the action function to maximize the total reward.
(Useful in e.g. *game play*)

27.2 Supervised Learning

Supervised learning a.k.a. inductive learning (a.k.a. Science)

Definition 2.1. A supervised (or inductive) learning problem consists of the following data:

- ▶ A set of hypotheses \mathcal{H} consisting of functions $A \rightarrow B$,
- ▶ a set of examples $T \subseteq A \times B$ called the training set, such that for every $a \in A$, there is at most one $b \in B$ with $\langle a, b \rangle \in T$,
($\Rightarrow T$ is a function on some subset of A)

We assume there is an unknown function $f: A \rightarrow B$ called the target function with $T \subseteq f$.

Definition 2.2. Inductive learning algorithms solve inductive learning problems by finding a hypothesis $h \in \mathcal{H}$ such that $h \sim f$ (for some notion of similarity).

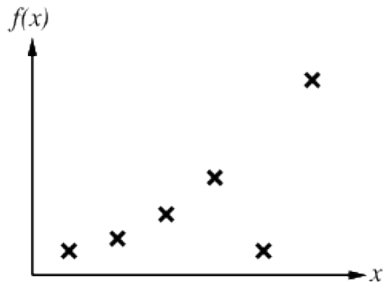
Definition 2.3. We call a supervised learning problem with target function $A \rightarrow B$ a classification problem if B is finite, and call the members of B classes.

We call it a regression problem if $B = \mathbb{R}$.

Inductive Learning Method

- ▶ **Idea:** Construct/adjust hypothesis $h \in \mathcal{H}$ to agree with a training set T .
- ▶ **Definition 2.4.** We call h consistent with f (on a set $T \subseteq \text{dom}(f)$), if it agrees with f (on all examples in T).
- ▶ **Example 2.5 (Curve Fitting).**

Training Set

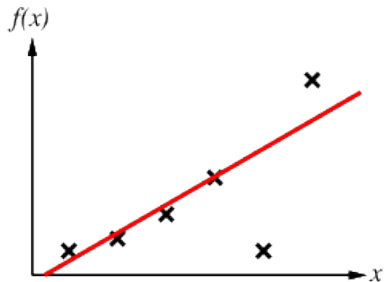


- ▶ **Ockham's-razor:** maximize a combination of consistency and simplicity.

Inductive Learning Method

- ▶ **Idea:** Construct/adjust hypothesis $h \in \mathcal{H}$ to agree with a training set T .
- ▶ **Definition 2.6.** We call h consistent with f (on a set $T \subseteq \text{dom}(f)$), if it agrees with f (on all examples in T).
- ▶ **Example 2.7 (Curve Fitting).**

Linear Hypothesis
partially, approximatively
consistent

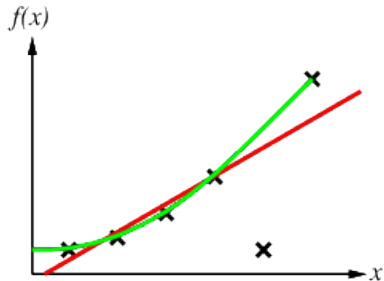


- ▶ **Ockham's-razor:** maximize a combination of consistency and simplicity.

Inductive Learning Method

- ▶ **Idea:** Construct/adjust hypothesis $h \in \mathcal{H}$ to agree with a training set T .
- ▶ **Definition 2.8.** We call h consistent with f (on a set $T \subseteq \text{dom}(f)$), if it agrees with f (on all examples in T).
- ▶ **Example 2.9 (Curve Fitting).**

Quadratic Hypothesis
partially consistent

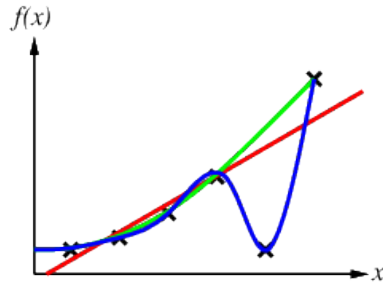


- ▶ **Ockham's-razor:** maximize a combination of consistency and simplicity.

Inductive Learning Method

- **Idea:** Construct/adjust hypothesis $h \in \mathcal{H}$ to agree with a training set T .
- **Definition 2.10.** We call h **consistent with f** (on a set $T \subseteq \text{dom}(f)$), if it agrees with f (on all examples in T).
- **Example 2.11 (Curve Fitting).**

Degree-4 Hypothesis
consistent

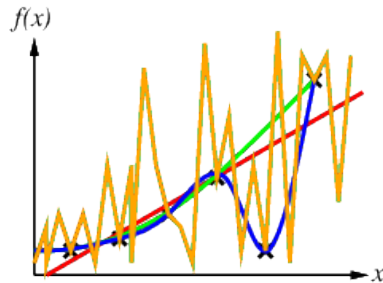


- **Ockham's-razor:** maximize a combination of consistency and simplicity.

Inductive Learning Method

- **Idea:** Construct/adjust hypothesis $h \in \mathcal{H}$ to agree with a training set T .
- **Definition 2.12.** We call h **consistent with f** (on a set $T \subseteq \text{dom}(f)$), if it agrees with f (on all examples in T).
- **Example 2.13 (Curve Fitting).**

High-degree Hypothesis
consistent



- **Ockham's-razor:** maximize a combination of consistency and simplicity.

Choosing the Hypothesis Space

- ▶ **Observation:** Whether we can find a consistent hypothesis for a given training set depends on the chosen hypothesis space.
- ▶ **Definition 2.14.** We say that an supervised learning problem is realizable, iff there is a hypothesis $h \in \mathcal{H}$ consistent with the training set T .
- ▶ **Problem:** We do not always know whether a given learning problem is realizable, unless we have prior knowledge. (depending on the hypothesis space)
- ▶ **Solution:** Make \mathcal{H} large, e.g. the class of all Turing machines.
- ▶ **Tradeoff:** The computational complexity of the supervised learning problem is tied to the size of the hypothesis space. E.g. consistency is not even decidable for general Turing machines.
- ▶ Much of the research in machine learning has concentrated on simple hypothesis spaces.
- ▶ **Preview:** We will concentrate on propositional logic and related languages first.

Independent and Identically Distributed

- ▶ **Problem:** We want to learn a *hypothesis* that fits the future data best.
- ▶ **Intuition:** This only works, if the *training set* is “representative” for the underlying process.
- ▶ **Idea:** We think of *examples* (seen and unseen) as a sequence, and express the “representativeness” as a *stationarity assumption* for the *probability distribution*.
- ▶ **Method:** Each *example* before we see it is a *random variable* E_j , the observed value $e_j = (x_j, y_j)$ samples its distribution.
- ▶ **Definition 2.15.** A sequence of E_1, \dots, E_n of random variables is **independent and identically distributed** (short **IID**), iff they are
 - ▶ **independent**, i.e. $\mathbb{P}(E_j | E_{j-1}, E_{j-2}, \dots) = \mathbb{P}(E_j)$ and
 - ▶ **identically distributed**, i.e. $\mathbb{P}(E_i) = \mathbb{P}(E_j)$ for all i and j .
- ▶ **Example 2.16.** A sequence of die tosses is **IID**. (fair or loaded does not matter)
- ▶ **Stationarity Assumption:** We assume that the set \mathcal{E} of *examples* is **IID** in the future.

27.3 Learning Decision Trees

Attribute-based Representations

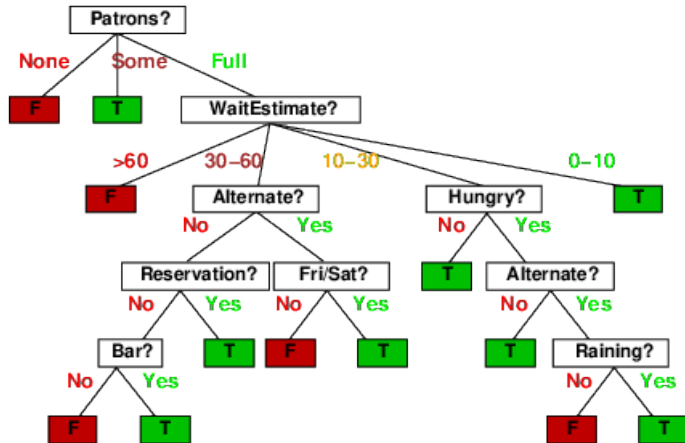
- ▶ **Definition 3.1.** In **attribute-based representations**, **examples** are described by
 - ▶ **attributes**: (simple) functions on **input samples**, (think pre classifiers on examples)
 - ▶ their **values**, and (classify by attributes)
 - ▶ **classifications**. (Boolean, discrete, continuous, etc.)
- ▶ **Example 3.2 (In a Restaurant)**. Situations where I will/won't wait for a table:

Example	Attributes										Target WillWait
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0-10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30-60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0-10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10-30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0-10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0-10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0-10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10-30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0-10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30-60	T

- ▶ **Definition 3.3.** For a boolean **classification** we say that an **example** is **positive** (T) or **negative** (F) depending on its class.

Decision Trees

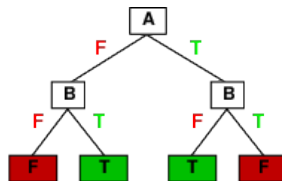
- ▶ Decision trees are one possible representation for hypotheses.
- ▶ **Example 3.4 (Restaurant continued).** Here is the “true” tree for deciding whether to wait:



- ▶ **Definition 3.5.** A **decision tree** for a given **attribute-based representation** is a **tree**, where the **non-leaf nodes** are labeled by **attributes**, their outgoing **edges** by disjoint **sets** of **attribute values**, and the **leaf nodes** are labeled by the **classifications**.
- ▶ **Definition 3.6.** We call an **attribute** together with a **set** of **attribute values** (an **inner node**) with outgoing **edge label** an **attribute test**.
- ▶ the **target function** is a function $A_1 \times \dots \times A_n \rightarrow C$, where A_i are the domains of the **attributes** and C is the set of **classifications**.

- ▶ **Decision trees** can express any function of the input attributes $\Rightarrow \mathcal{H} = A_1 \times \dots \times A_n$
- ▶ **Example 3.7.** For **Boolean functions**, a path from the root to a **leaf** corresponds to a row in a truth table:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



\Rightarrow a **decision tree** corresponds to a truth table

(Formula in DNF)

- ▶ Trivially, for any **training set** there is a **consistent hypothesis** with one **path** to a **leaf** for each **example**, but it probably won't generalize to new **examples**.
- ▶ **Solution:** Prefer to find more *compact* decision trees.

Decision Tree learning

- ▶ **Aim:** Find a small **decision tree consistent** with the training **examples**.
- ▶ **Idea:** (recursively) choose “most significant” attribute as root of (sub)tree.
- ▶ **Definition 3.8.** The following **algorithm** performs **decision tree learning (DTL)**

```
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
    best := Choose—Attribute(attributes, examples)
    tree := a new decision tree with root test best
    m := MODE(examples)
    for each value  $v_i$  of best do
      examplesi := {elements of examples with best =  $v_i$ }
      subtree := DTL(examplesi, attributes \ best, m)
      add a branch to tree with label  $v_i$  and subtree subtree
  return tree
```

MODE(*examples*)= most frequent value in *example*.

Choosing an Attribute

- ▶ **Idea:** A good **attribute** splits the **examples** into subsets that are (ideally) “all **positive**” or “all **negative**”.
- ▶ **Example 3.9.**



Attribute “Patrons?” is a better choice, it gives gives **information** about the **classification**.

- ▶ Can we make this more formal? \leadsto Use information theory!

(up next)

27.4 Using Information Theory

Information Entropy

Intuition: Information answers questions – the less I know initially, the more Information is contained in an answer.

Definition 4.1. Let $\langle p_1, \dots, p_n \rangle$ the distribution of a random variable P . The information (also called entropy) of P is

$$I(\langle p_1, \dots, p_n \rangle) := \sum_{i=1}^n -p_i \cdot \log_2(p_i)$$

Note: For $p_i = 0$, we consider $p_i \cdot \log_2(p_i) = 0$

($\log_2(0)$ is undefined)

The unit of information is a bit, where $1\text{b} := I(\langle \frac{1}{2}, \frac{1}{2} \rangle) = 1$

Example 4.2 (Information of a Coin Toss).

- ▶ For a fair coin toss we have $I(\langle \frac{1}{2}, \frac{1}{2} \rangle) = -\frac{1}{2} \log_2(\frac{1}{2}) - \frac{1}{2} \log_2(\frac{1}{2}) = 1\text{b}$.
- ▶ With a loaded coin (99% heads) we have $I(\langle \frac{1}{100}, \frac{99}{100} \rangle) = 0.08\text{b}$.

Rightarrow Information goes to 0 as head probability goes to 1.

“How likely is the outcome actually going to tell me something informative?”

Information Gain in Decision Trees

Idea: Suppose we have p examples classified as positive and n examples as negative. We can then estimate the probability distribution of the classification C with $\mathbb{P}(C) = \langle \frac{p}{p+n}, \frac{n}{p+n} \rangle$, and need $I(\mathbb{P}(C))$ bits to correctly classify a new example.

Example 4.3. For 12 restaurant examples and $p = n = 6$, we need $I(\mathbb{P}(\text{WillWait})) = I(\langle \frac{6}{12}, \frac{6}{12} \rangle) = 1\text{b}$ of information.
(i.e. exactly the information which of the two classes)

Treating attributes also as random variables, we can compute how much information is needed after knowing the value for one attribute:

Example 4.4. If we know $\text{Pat} = \text{Full}$, we only need $I(\mathbb{P}(\text{WillWait}|\text{Pat} = \text{Full})) = I(\langle \frac{4}{6}, \frac{2}{6} \rangle) \cong 0.9$ bits of information.

Note: The expected number of bits needed after an attribute test on A is

$$\sum_a P(A = a) \cdot I(\mathbb{P}(C|A = a))$$

Definition 4.5. The information gain from an attribute test A is

$$\text{Gain}(A) := I(\mathbb{P}(C)) - \sum_a P(A = a) \cdot I(\mathbb{P}(C|A = a))$$

Information Gain (continued)

- **Definition 4.6.** Assume we know the results of some **attribute tests** $b := B_1 = b_1 \wedge \dots \wedge B_n = b_n$. Then the **conditional information gain** from an attribute test A is

$$\text{Gain}(A|b) := I(\mathbb{P}(C|b)) - \sum_a P(A = a | b) \cdot I(\mathbb{P}(C|a, b))$$

- **Example 4.7.** If the **classification** C is Boolean and we have p **positive** and n **negative examples**, the **information gain** is

$$\text{Gain}(A) = I(\langle \frac{p}{p+n}, \frac{n}{p+n} \rangle) - \sum_a \frac{p_a + n_a}{p+n} I(\langle \frac{p_a}{p_a + n_a}, \frac{n_a}{p_a + n_a} \rangle)$$

where p_a and n_a are the **positive** and **negative examples** with $A = a$.

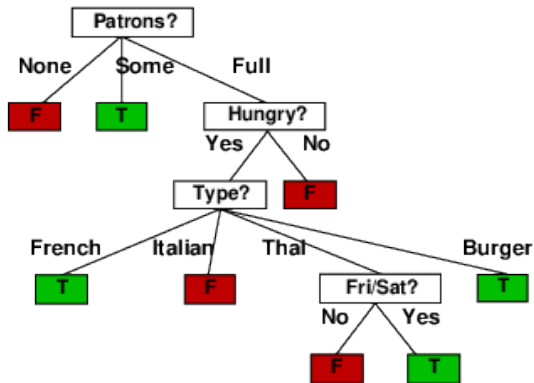
- **Example 4.8.**

$$\begin{aligned} \text{Gain}(\text{Patrons?}) &= 1 - \left(\frac{2}{12} I(\langle 0, 1 \rangle) + \frac{4}{12} I(\langle 1, 0 \rangle) + \frac{6}{12} I(\langle \frac{2}{6}, \frac{4}{6} \rangle) \right) \\ &\approx 0.541\text{b} \\ \text{Gain}(\text{Type}) &= 1 - \left(\frac{2}{12} I(\langle \frac{1}{2}, \frac{1}{2} \rangle) + \frac{2}{12} I(\langle \frac{1}{2}, \frac{1}{2} \rangle) + \frac{4}{12} I(\langle \frac{2}{4}, \frac{2}{4} \rangle) + \frac{4}{12} I(\langle \frac{2}{4}, \frac{2}{4} \rangle) \right) \\ &\approx 0\text{b} \end{aligned}$$

- **Idea:** Choose the **attribute** that **maximizes information gain**.

Restaurant Example contd.

- **Example 4.9.** Decision tree learned by DTL from the 12 examples using information gain maximization for Choose—Attribute:

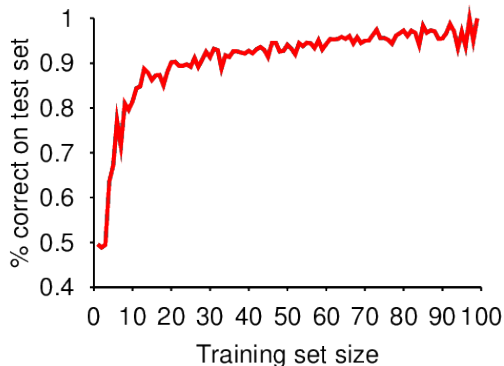


- **Result:** Substantially simpler than “true” tree – a more complex hypothesis isn’t justified by small amount of data.

27.5 Evaluating and Choosing the Best Hypothesis

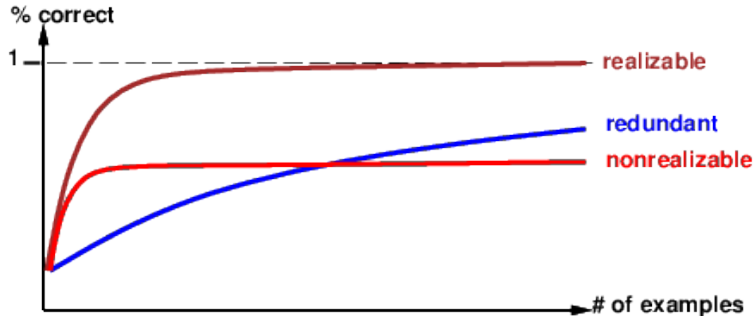
Performance measurement

- ▶ **Question:** How do we know that $h \cong f$? (Hume's *Problem of Induction*)
 1. Use theorems of computational/statistical learning theory.
 2. Try h on a new **test set** of **examples**. (use *same distribution over example space as training set*)
- ▶ **Definition 5.1.** The **learning curve** $\hat{=}$ **percentage** correct on **test set** as a **function** of **training set size**.
- ▶ **Example 5.2.** Restaurant data; graph averaged over 20 trials



► **Observation 5.3.** The *learning curve* depends on

- *realizable* (can express target function) vs. *non-realizable*
non-realizability can be due to missing attributes or restricted *hypothesis* class (e.g., thresholded linear function)
- *redundant expressiveness* (e.g., lots of irrelevant attributes)



- ▶ **Observation:** Sometimes a learned hypothesis is more specific than the experiments warrant.
- ▶ **Definition 5.4.** We speak of **overfitting**, if a hypothesis h describes random error in the (limited) training set rather than the underlying relationship. **Underfitting** occurs when h cannot capture the underlying trend of the data.
- ▶ **Qualitatively:** Overfitting increases with the size of hypothesis space and the number of attributes, but decreases with number of examples.
- ▶ **Idea:** Combat overfitting by “generalizing” decision trees computed by DTL.

- ▶ **Idea:** Combat **overfitting** by “generalizing” decision trees \leadsto **prune** “irrelevant” nodes.
- ▶ **Definition 5.5.** For **decision tree pruning** repeat the following on a learned decision tree:
 - ▶ Find a **terminal** test node n (only result leaves as **children**)
 - ▶ If test is **irrelevant**, i.e. has low **information gain**, **prune** it by replacing n by with a leaf node.
- ▶ **Question:** How big should the **information gain** be to split (\leadsto keep) a node?
- ▶ **Idea:** Use a **statistical significance** test.
- ▶ **Definition 5.6.** A result has **statistical significance**, if the probability they could arise from the **null hypothesis** (i.e. the assumption that there is no underlying pattern) is very low (usually 5%).

Determining Attribute Irrelevance

- ▶ For **decision tree pruning**, the **null hypothesis** is that the attribute is **irrelevant**.
- ▶ Compute the probability that the example distribution (p **positive**, n **negative**) for a terminal node deviates from the expected distribution under the **null hypothesis**.
- ▶ For an attribute A with d values, compare the actual numbers p_k and n_k in each subset s_k with the expected numbers
(expected if A is irrelevant)
 $\hat{p}_k = p \cdot \frac{p_k + n_k}{p + n}$ and $\hat{n}_k = n \cdot \frac{p_k + n_k}{p + n}$.
- ▶ A convenient measure of the total deviation is
(sum of squared errors)

$$\Delta = \sum_{k=1}^d \frac{(p_k - \hat{p}_k)^2}{\hat{p}_k} + \frac{(n_k - \hat{n}_k)^2}{\hat{n}_k}$$

- ▶ **Lemma 5.7 (Neyman-Pearson)**. Under the **null hypothesis**, the value of Δ is distributed according to the χ^2 distribution with $d - 1$ degrees of freedom. [NeyPea:pmtsh33]
- ▶ **Definition 5.8**. **Decision tree pruning** with Pearson's χ^2 with $d - 1$ degrees of freedom for Δ is called **χ^2 pruning**. (χ^2 values from stats library.)
- ▶ **Example 5.9**. The *type* attribute has four values, so three degrees of freedom, so $\Delta = 7.82$ would reject the **null hypothesis** at the 5% level.

- ▶ **Recall:** We want to learn a **hypothesis** that fits the future data best.
- ▶ **Definition 5.10.** Given an **inductive learning problem** with a set of **examples** $T \subseteq AB$, we define the **error rate** of a **hypothesis** $h \in \mathcal{H}$ as the fraction of errors:

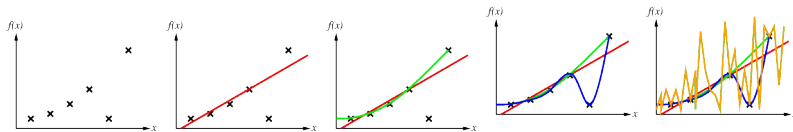
$$\frac{|\{\langle x, y \rangle \in T \mid h(x) \neq y\}|}{|T|}$$

- ▶ **Caveat:** A low **error rate** on the **training set** does not mean that a **hypothesis** generalizes well.
- ▶ **Idea:** Do not use homework questions in the exam.
- ▶ **Definition 5.11.** The practice of splitting the data available for learning into
 1. a **training set** from which the **learning algorithm** produces a **hypothesis** h and
 2. a **test set**, which is used for evaluating his called **holdout cross validation**. (no peeking at test set allowed)

- ▶ **Question:** What is a good ratio between training set and test set size?
 - ▶ small training set \leadsto poor hypothesis.
 - ▶ small test set \leadsto poor estimate of the accuracy.
- ▶ **Definition 5.12.** In k fold cross validation, we perform k rounds of learning, each with $1/k$ of the data as test set and average over the k error rates.
- ▶ **Intuition:** Each example does double duty: for training and testing.
- ▶ $k = 5$ and $k = 10$ are popular \leadsto good accuracy at k times computation time.
- ▶ **Definition 5.13.** If $k = |\text{dom}(f)|$, then k fold cross validation is called leave one out cross validation (LOOCV).

Model Selection

- ▶ **Definition 5.14.** The **model selection** problem is to determine – given data – a good **hypothesis space**.
- ▶ **Example 5.15.** What is the best polynomial degree to fit the data



- ▶ **Observation 5.16.** We can solve the problem of “learning from observations f ” in a two-part process:
 1. **model selection** determines a **hypothesis space \mathcal{H}** ,
 2. **optimization** solves the induced **inductive learning problem**.
- ▶ **Idea:** Solve the two parts together by iteration over “size”. (they inform each other)
- ▶ **Problem:** Need a notion of “size” \Leftarrow e.g. number of nodes in a **decision tree**.
- ▶ **Concrete Problem:** Find the “size” that best balances **overfitting** and **underfitting** to optimize test set accuracy.

Model Selection Algorithm (Wrapper)

- **Definition 5.17.** The **model selection algorithm (MSA)** jointly optimizes **model selection** and **optimization** by partitioning and cross-validation:

function CROSS-VALIDATION-WRAPPER(*Learner*, *k*, *examples*) **returns** a hypothesis

local variables: *errT*, an array, indexed by size, storing training-set error rates

errV, an array, indexed by size, storing validation-set error rates

for size = 1 **to** ∞ **do**

errT[size], *errV*[size] := CROSS-VALIDATION(*Learner*, size, *k*, *examples*)

if *errT* has converged **then do**

best_size := the value of size with minimum *errV*[size]

return Learner(*best_size*, *examples*)

function CROSS-VALIDATION(*Learner*, size, *k*, *examples*) **returns** two values:

 average training set error rate, average validation set error rate

fold_errT := 0; *fold_errV* := 0

for fold = 1 **to** *k* **do**

training_set, *validation_set* := PARTITION(*examples*, fold, *k*)

h := Learner(size, *training_set*)

fold_errT := *fold_errT* + ERROR-RATE(*h*, *training_set*)

fold_errV := *fold_errV* + ERROR-RATE(*h*, *validation_set*)

return *fold_errT* / *k*, *fold_errV* / *k*

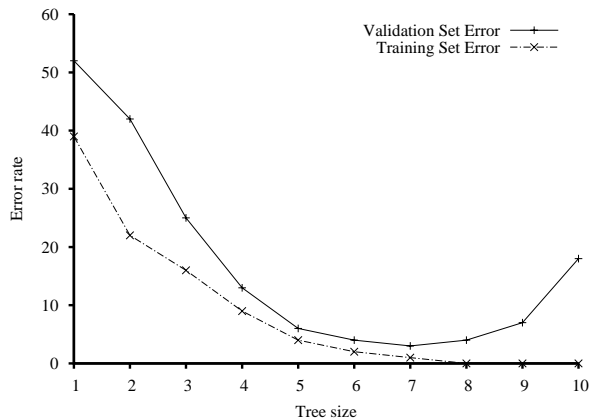
function PARTITION(*examples*, fold, *k*) **returns** two sets:

 a validation set of size $|examples|/k$ and the rest; the split is different **for** each *fold* value

Error Rates on Training/Validation Data

► Example 5.18 (An Error Curve for Restaurant Decision Trees).

Modify DTL to be breadth-first, information gain sorted, stop after k nodes.



Stops when training set error rate converges, choose optimal tree for validation curve. (here a tree with 7 nodes)

From Error Rates to Loss Functions

- ▶ **So far** we have been **minimizing error rates**. (better than maximizing 😊)
- ▶ **Example 5.19 (Classifying Spam)**. It is much worse to classify ham (legitimate mails) as spam than vice versa. (message loss)
- ▶ **Recall Rationality:** Decision-makers should **maximize expected utility (MEU)**.
- ▶ **So:** Machine learning should **maximize “utility”**. (not only minimize error rates)
- ▶ machine learning traditionally deals with utilities in form of “**loss functions**”.
- ▶ **Definition 5.20.** The **loss function L** is defined by setting $L(x, y, \hat{y})$ to be the amount of utility lost by prediction $h(x) = \hat{y}$ instead of $f(x) = y$. If L is independent of x , we often use $L(y, \hat{y})$.
- ▶ **Example 5.21.** $L(\text{spam}, \text{ham}) = 1$, while $L(\text{ham}, \text{spam}) = 10$.

► **Note:** $L(y, y) = 0$. (no loss if you are exactly correct)

► **Definition 5.22 (Popular general loss functions).**

absolute value loss $L_1(y, \hat{y}) := |y - \hat{y}|$ small errors are good

squared error loss $L_2(y, \hat{y}) := (y - \hat{y})^2$ ditto, but differentiable

0/1 loss $L_{0/1}(y, \hat{y}) := 0, \text{ if } y = \hat{y}, \text{ else } 1$ error rate

► **Idea:** Maximize expected utility by choosing hypothesis h that minimizes expected loss over all $(x, y) \in f$.

► **Definition 5.23.** Let \mathcal{E} be the set of all possible examples and $\mathbb{P}(X, Y)$ the prior probability distribution over its components, then the expected generalization loss for a hypothesis h with respect to a loss function L is

$$\text{GenLoss}_L(h) := \sum_{(x, y) \in \mathcal{E}} L(y, h(x)) \cdot P(x, y)$$

and the best hypothesis $h^* := \underset{h \in \mathcal{H}}{\text{argmin}} \text{GenLoss}_L(h)$.

- ▶ **Problem:** $P(X, Y)$ is unknown \leadsto learner can only estimate **generalization loss**:
- ▶ **Definition 5.24.** Let L be a **loss function** and E a set of **examples** with $\#(E) = N$, then we call

$$\text{EmpLoss}_{L,E}(h) := \frac{1}{N} \left(\sum_{(x,y) \in E} L(y, h(x)) \right)$$

the **empirical loss** and $\hat{h}^* := \underset{h \in \mathcal{H}}{\text{argmin}} \text{EmpLoss}_{L,E}(h)$ the **estimated best hypothesis**.

- ▶ There are four reasons why \hat{h}^* may differ from f :
 1. **Realizability**: then we have to settle for an approximation \hat{h}^* of f .
 2. **Variance**: different subsets of f give different $\hat{h}^* \leadsto$ more **examples**.
 3. **Noise**: if f is non deterministic, then we cannot expect perfect results.
 4. **Computational complexity**: if \mathcal{H} is too large to systematically explore, we make due with subset and get an approximation.

Regularization

- **Idea:** Directly use empirical loss to solve model selection.
Minimize the weighted sum of empirical loss and hypothesis complexity.
- **Definition 5.25.** Let $\lambda \in \mathbb{R}$, $h \in \mathcal{H}$, and E a set of examples, then we call

(finding a good \mathcal{H})
(to avoid overfitting).

$$\text{Cost}_{L,E}(h) := \text{EmpLoss}_{L,E}(h) + \lambda \text{Complexity}(h)$$

the total cost of h on E .

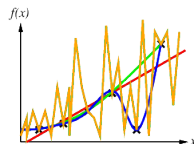
- **Definition 5.26.** The process of finding a total cost minimizing hypothesis

$$\hat{h}^* := \underset{h \in \mathcal{H}}{\operatorname{argmin}} \text{Cost}_{L,E}(h)$$

is called regularization; Complexity is called the regularization function or hypothesis complexity.

- **Example 5.27 (Regularization for Polynomials).**

A good regularization function for polynomials is the sum of squares of exponents. \leadsto keep away from wiggly curves!



- ▶ **Remark:** In regularization, empirical loss and hypothesis complexity are not measured in the same scale $\leadsto \lambda$ mediates between scales.
 - ▶ **Idea:** Measure both in the same scale \leadsto use information content, i.e. in bits.
 - ▶ **Definition 5.28.** Let $h \in \mathcal{H}$ be a hypothesis and E a set of examples, then the description length of (h, E) is computed as follows:
 1. encode the hypothesis as a Turing machine program, count bits.
 2. count data bits:
 - ▶ correctly predicted example $\leadsto 0b$
 - ▶ incorrectly predicted example \leadsto according to size of error.
- The minimum description length or MDL hypothesis minimizes the total number of bits required.
- ▶ This works well in the limit, but for smaller problems there is a difficulty in that the choice of encoding for the program affects the outcome.
 - ▶ e.g., how best to encode a decision tree as a bit string?

The Scale of Machine Learning

- ▶ Traditional methods in statistics and early machine learning concentrated on small-scale learning (50-5000 examples)
 - ▶ Generalization error mostly comes from
 - ▶ approximation error of not having the true f in the hypothesis space
 - ▶ estimation error of too few training examples to limit variance.
- ▶ In recent years there has been more emphasis on large-scale learning. (millions of examples)
 - ▶ Generalization error is dominated by limits of computation
 - ▶ there is enough data and a rich enough model that we could find an h that is very close to the true f ,
 - ▶ but the computation to find it is too complex, so we settle for a sub-optimal approximation.
 - ▶ Hardware advances (GPU farms, Amazon EC2, Google Data Centers, ...) help.

27.6 Computational Learning Theory

A (General) Theory of Learning?

- ▶ **Main Question:** How can we be sure that our learning algorithm has produced a hypothesis that will predict the correct value for previously unseen inputs?
- ▶ **Formally:** How do we know that the hypothesis h is close to the target function f if we don't know what f is?
- ▶ **Other - more recent - Questions:**
 - ▶ How many examples do we need to get a good h ?
 - ▶ What hypothesis space \mathcal{H} should we use?
 - ▶ If the \mathcal{H} is very complex, can we even find the best h , or do we have to settle for a local maximum in \mathcal{H} .
 - ▶ How complex should h be?
 - ▶ How do we avoid overfitting?
- ▶ “Computational Learning Theory” tries to answer these using concepts from AI, statistics, and theoretical CS.

► Basic idea of Computational Learning Theory:

- Any hypothesis h that is seriously wrong will almost certainly be “found out” with high probability after a small number of examples, because it will make an incorrect prediction.
- Thus, if h is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong.
- $\leadsto h$ is probably approximately correct.

► **Definition 6.1.** Any learning algorithm that returns hypotheses that are probably approximately correct is called a PAC learning algorithm.

► Derive performance bounds for PAC learning algorithms in general, using the

► **Stationarity Assumption (again):** We assume that the set \mathcal{E} of possible examples is IID \leadsto we have a fixed distribution $P(E) = P(X, Y)$ on examples.

► **Simplifying Assumptions:** f is a function (deterministic) and $f \in \mathcal{H}$.

- ▶ Start with PAC theorems for Boolean functions, for which $L_{0/1}$ is appropriate.
- ▶ **Definition 6.2.** The error rate $\text{error}(h)$ of a hypothesis h is the probability that h misclassifies a new example.

$$\text{error}(h) := \text{GenLoss}_{L_{0/1}}(h) = \sum_{(x,y) \in \mathcal{E}} L_{0/1}(y, h(x)) \cdot P(x, y)$$

- ▶ **Intuition:** $\text{error}(h)$ is the probability that h misclassifies a new example.
- ▶ This is the same quantity as measured in the learning curves above.
- ▶ **Definition 6.3.** A hypothesis h is called **approximatively correct**, iff $\text{error}(h) \leq \epsilon$ for some small $\epsilon > 0$. We write $\mathcal{H}_b := \{h \in \mathcal{H} \mid \text{error}(h) > \epsilon\}$ for the “seriously bad” hypotheses.

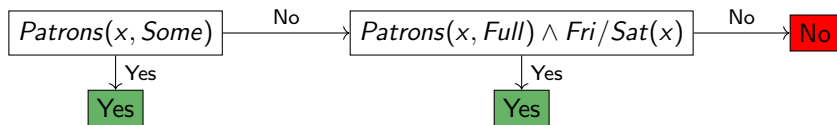
- ▶ Let's compute the probability that $h_b \in \mathcal{H}_b$ is consistent with the first N examples.
- ▶ We know $\text{error}(h_b) > \epsilon$
 - $\leadsto P(h_b \text{ agrees with } N \text{ examples}) \leq (1 - \epsilon)^N$. (independence)
 - $\leadsto P(\mathcal{H}_b \text{ contains consistent hyp.}) \leq |\mathcal{H}_b| \cdot (1 - \epsilon)^N \leq |\mathcal{H}| \cdot (1 - \epsilon)^N$. ($\mathcal{H}_b \subseteq \mathcal{H}$)
 - \leadsto to bound this by a small δ , show the algorithm $N \geq \frac{1}{\epsilon} \cdot (\log_2(\frac{1}{\delta}) + \log_2(|\mathcal{H}|))$ examples.
- ▶ **Definition 6.4.** The number of required examples as a function of ϵ and δ is called the sample complexity of \mathcal{H} .
- ▶ **Example 6.5.** If \mathcal{H} is the set of n -ary Boolean functions, then $|\mathcal{H}| = 2^{2^n}$.
 - \leadsto sample complexity grows with $\mathcal{O}(\log_2(2^{2^n})) = \mathcal{O}(2^n)$.
 - There are 2^n possible examples,
 - \leadsto PAC learning for Boolean functions needs to see (nearly) all examples.

Escaping Sample Complexity

- ▶ **Problem:** PAC learning for Boolean functions needs to see (nearly) all examples.
 - ▶ \mathcal{H} contains enough hypotheses to classify any given set of examples in all possible ways.
 - ▶ In particular, for any set of N examples, the set of hypotheses consistent with those examples contains equal numbers of hypotheses that predict x_{N+1} to be positive and hypotheses that predict x_{N+1} to be negative.
- ▶ **Idea/Problem:** restrict the \mathcal{H} in some way (but we may lose realizability)
- ▶ **Three Ways out of this Dilemma:**
 1. bring prior knowledge into the problem. (???)
 2. prefer simple hypotheses. (e.g. decision tree pruning)
 3. focus on “learnable subsets” of \mathcal{H} . (next)

PAC Learning: Decision Lists

- ▶ **Idea:** Apply PAC learning to a “learnable hypothesis space”.
- ▶ **Definition 6.6.** A **decision list** consists of a sequence of tests, each of which is a **conjunction** of **literals**.
 - ▶ If a test succeeds when applied to an **example** description, the decision list specifies the value to be returned.
 - ▶ If the test fails, processing continues with the next test in the list.
- ▶ **Remark:** Like **decision trees**, but restricted branching, but more complex tests.
- ▶ **Example 6.7 (A decision list for the Restaurant Problem).**



- ▶ **Lemma 6.8.** Given arbitrary size conditions, **decision lists** can represent arbitrary **Boolean functions**.
- ▶ This directly defeats our purpose of finding a “learnable subset” of \mathcal{H} .

- ▶ **Definition 6.9.** The set of **decision lists** where tests are of conjunctions of at most k **literals** is denoted by $k\text{-DL}$.
- ▶ **Example 6.10.** The **decision list** from ?? is in 2-DL .
- ▶ **Observation 6.11.** $k\text{-DL}$ contains $k\text{-DT}$, the set of **decision trees** of depth at most k .
- ▶ **Definition 6.12.** We denote the set of $k\text{-DL}$ **decision lists** with at most n Boolean attributes with $k\text{-DL}(n)$. The set of conjunctions of at most k **literals** over n attributes is written as $\text{Conj}(k, n)$.
- ▶ **Decision lists** are constructed of optional yes/no tests, so there are at most $3^{|\text{Conj}(k, n)|}$ distinct sets of component tests. Each of these sets of tests can be in any order, so
$$|k\text{-DL}(n)| \leq 3^{|\text{Conj}(k, n)|} \cdot |\text{Conj}(k, n)|!$$

Decision Lists: Learnable Subsets (Sample Complexity)

- ▶ The number of conjunctions of k literals from n attributes is given by

$$|\text{Conj}(k, n)| = \sum_{i=1}^k \binom{2n}{i}$$

thus $|\text{Conj}(k, n)| = \mathcal{O}(n^k)$. Hence, we obtain (after some work)

$$|k\text{-DL}(n)| = 2^{\mathcal{O}(n^k \log_2(n^k))}$$

- ▶ Plug this into the equation for the sample complexity: $N \geq \frac{1}{\epsilon} \cdot (\log_2(\frac{1}{\delta}) + \log_2(|\mathcal{H}|))$ to obtain

$$N \geq \frac{1}{\epsilon} \cdot (\log_2(\frac{1}{\delta}) + \log_2(\mathcal{O}(n^k \log_2(n^k))))$$

- ▶ **Intuitively:** Any algorithm that returns a consistent decision list will PAC learn a k -DL function in a reasonable number of examples, for small k .

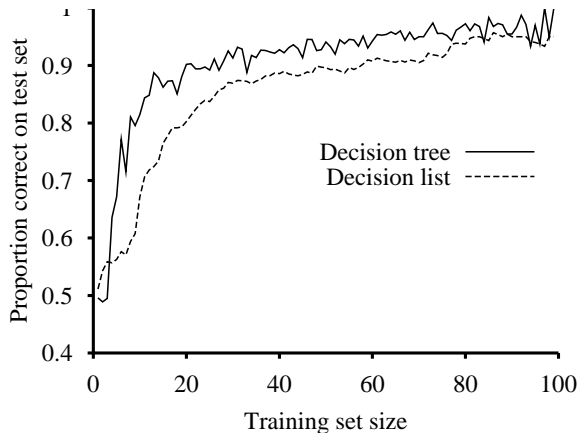
Decision Lists Learning

- **Idea:** Use a **greedy search algorithm** that repeats
 1. **find** test that agrees exactly with some subset E of the training set,
 2. **add** it to the decision list under construction and removes E ,
 3. **construct** the remainder of the DL using just the remaining **examples**,until there are no **examples** left.
- **Definition 6.13.** The following **algorithm** performs **decision list learning**

```
function DLL( $E$ ) returns a decision list, or failure
  if  $E$  is empty then return (the trivial decision list) No
   $t :=$  a test that matches a nonempty subset  $E_t$  of  $E$ 
    such that the members of  $E_t$  are all positive or all negative
  if there is no such  $t$  then return failure
  if the examples in  $E_t$  are positive then  $o :=$  Yes else  $o :=$  No
  return a decision list with initial test  $t$  and outcome  $o$  and remaining tests given by
    DLL( $E \setminus E_t$ )
```

Decision Lists Learning in Comparison

- **Learning curves:** for DLL (and DTL for comparison)



- **Upshot:** The simpler DLL works quite well!

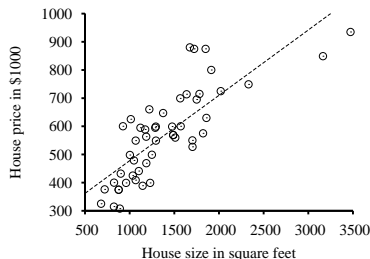
27.7 Regression and Classification with Linear Models

Univariate Linear Regression

- **Definition 7.1.** A **univariate** or **unary** function is a function with one argument.
- **Recall:** A mapping f between vector spaces is called **linear**, iff it preserves **rmodule/plus** and **rmodule/scalar multiplication**, i.e. $f(\alpha \cdot v_1 + v_2) = \alpha \cdot f(v_1) + f(v_2)$.
- **Observation 7.2.** A **univariate, linear function** $f: \mathbb{R} \rightarrow \mathbb{R}$ is of the form $f(x) = w_1x + w_0$ for some $w_i \in \mathbb{R}$.
- **Definition 7.3.** Given a vector $w := (w_0, w_1)$, we define $h_w(x) := w_1x + w_0$.
- **Definition 7.4.** Given a set of **examples** $E \subseteq \mathbb{R} \times \mathbb{R}$, the task of finding h_w that best fits E is called **linear regression**.
- **Example 7.5.**

Examples of house price vs. square feet in houses sold in Berkeley in July 2009.

Also: linear function hypothesis that minimizes squared error loss $y = 0.232x + 246$.



Univariate Linear Regression by Loss Minimization

- **Idea:** Minimize squared error loss over $\{(x_i, y_i) \mid i \leq N\}$ (used already by Gauss)

$$\text{Loss}(h_w) = \sum_{j=1}^N L_2(y_j, h_w(x_j)) = \sum_{j=1}^N (y_j - h_w(x_j))^2 = \sum_{j=1}^N (y_j - (w_1 x_j + w_0))^2$$

Task: find $w^* := \underset{w}{\operatorname{argmin}} \text{Loss}(h_w)$.

- **Recall:** $\sum_{j=1}^N (y_j - (w_1 x_j + w_0))^2$ is **minimized**, when the partial derivatives wrt. the w_i are zero, i.e. when

$$\frac{\partial}{\partial w_0} \left(\sum_{j=1}^N (y_j - (w_1 x_j + w_0))^2 \right) = 0 \quad \text{and} \quad \frac{\partial}{\partial w_1} \left(\sum_{j=1}^N (y_j - (w_1 x_j + w_0))^2 \right) = 0$$

- **Observation:** These equations have a unique solution:

$$w_1 = \frac{N(\sum_j x_j y_j) - (\sum_j x_j)(\sum_j y_j)}{N(\sum_j x_j^2) - (\sum_j x_j)^2} \quad w_0 = \frac{(\sum_j y_j) - w_1(\sum_j x_j)}{N}$$

- **Remark:** Closed-form solutions only exist for **linear regression**, for other (differentiable) **hypothesis spaces** use **gradient descent** methods for adjusting/learning weights.

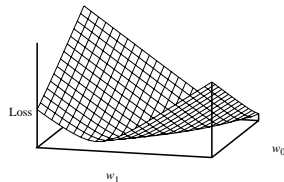
A Picture of the Weight Space

- **Remark:** Many forms of learning involve adjusting weights to **minimize** loss.
- **Definition 7.6.** The **weight space** of a parametric model is the space of all possible combinations of parameters (called the **weights**). Loss minimization in a **weight space** is called **weight fitting**.

The **weight space** of univariate **linear regression** is \mathbb{R}^2 .

→ graph the loss function over \mathbb{R}^2 .

Note: it is **convex**.



- **Observation 7.7.** The **squared error loss** function is **convex** for any **linear regression** problem → there are no **local minima**.

- ▶ If we do not have closed form solutions for minimizing loss, we need to search.
- ▶ **Idea:** Use local search (hill climbing) methods.
- ▶ **Definition 7.8.** The gradient descent algorithm for finding a minimum of a continuous function F is hill climbing in the direction of the steepest descent, which can be computed by the partial derivatives of F .

function gradient-descent(F, w, α) **returns** a **local** minimum of F

inputs: a differentiable **function** F and initial weights w .

loop until w converges **do**

for each w_i **do**

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} F(w)$$

end for

end loop

The parameter α is called the **learning rate**. It can be a fixed constant or it can decay as learning proceeds.

Gradient-Descent for Loss

- ▶ Let's try **gradient descent** for **Loss**.
- ▶ Work out the partial derivatives for one **example** (x,y) :

$$\frac{\partial \text{Loss}(w)}{\partial w_i} = \frac{\partial (y - h_w(x))^2}{\partial w_i} = 2(y - h_w(x)) \frac{\partial (y - (w_1 x + w_0))}{\partial w_i}$$

and thus

$$\frac{\partial \text{Loss}(w)}{\partial w_0} = -2(y - h_w(x)) \quad \frac{\partial \text{Loss}(w)}{\partial w_1} = -2(y - h_w(x))x$$

Plug this into the **gradient descent** updates:

$$w_0 \leftarrow w_0 - \alpha \cdot (-2(y - h_w(x))) \quad w_1 \leftarrow w_1 - \alpha \cdot -2((y - h_w(x))) \cdot x$$

Gradient-Descent for Loss (continued)

- ▶ Analogously for n training examples (x_j, y_j) :

- ▶ **Definition 7.9.**

$$w_0 \leftarrow w_0 - \alpha \left(\sum_j -2(y_j - h_w(x_j)) \right) \quad w_1 \leftarrow w_1 - \alpha \left(\sum_j -2(y_j - h_w(x_j))x_j \right)$$

These updates constitute the **batch gradient descent learning rule** for univariate **linear regression**.

- ▶ Convergence to the unique global loss minimum is guaranteed (as long as we pick α small enough) but may be very slow.
- ▶ Doing **batch gradient descent** on random subsets of the **examples** of fixed batch size n is called **stochastic gradient descent (SGD)**. (More computationally efficient than updating for every example)

Multivariate Linear Regression

- ▶ **Definition 7.10.** A **multivariate** or **n -ary** function is a function with one or more arguments.
- ▶ We can use it for **multivariate linear regression**.
- ▶ **Idea:** Every **example** \vec{x}_j is an n element vector and the **hypothesis space** is the set of functions

$$h_{sw}(\vec{x}_j) = w_0 + w_1 x_{j,1} + \dots + w_n x_{j,n} = w_0 + \sum_i w_i x_{j,i}$$

- ▶ **Trick:** Invent $x_{j,0} := 1$ and use matrix notation:

$$h_{sw}(\vec{x}_j) = \vec{w} \cdot \vec{x}_j = \vec{w}^t \vec{x}_j = \sum_i w_i x_{j,i}$$

- ▶ **Definition 7.11.** The best vector of weights, w^* , **minimizes** squared-error loss over the **examples**:
 $w^* := \underset{w}{\operatorname{argmin}} (\sum_j L_2(y_j)(w \cdot \vec{x}_j))$.
- ▶ **Gradient descent** will reach the (unique) minimum of the loss function; the update equation for each weight w_i is

$$w_i \longleftarrow w_i - \alpha \left(\sum_j x_{j,i} (y_j - h_w(\vec{x}_j)) \right)$$

Multivariate Linear Regression (Analytic Solutions)

- ▶ We can also solve analytically for the w^* that minimizes loss.
- ▶ Let \vec{y} be the vector of outputs for the training examples, and X be the data matrix, i.e., the matrix of inputs with one n -dimensional example per row.
Then the solution $w^* = (X^T X)^{-1} X^T \vec{y}$ minimizes the squared error.

Multivariate Linear Regression (Regularization)

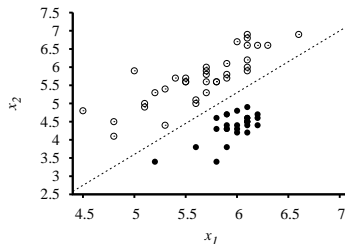
- ▶ **Remark:** Univariate linear regression does not overfit, but in the multivariate case there might be “redundant dimensions” that result in overfitting.
- ▶ **Idea:** Use regularization with a complexity function based on weights.
- ▶ **Definition 7.12.** $\text{Complexity}(h_w) = L_q(w) = \sum_i |w_i|^q$
- ▶ **Caveat:** Do not confuse this with the loss functions L_1 and L_2 .
- ▶ **Problem:** Which q should we pick? (L_1 and L_2 minimize sum of absolute values/squares)
- ▶ **Answer:** It depends on the application.
- ▶ **Remark:** L_1 -regularization tends to produce a sparse model, i.e. it sets many weights to 0, effectively declaring the corresponding attributes to be irrelevant. Hypotheses that discard attributes can be easier for a human to understand, and may be less likely to overfit. (see [RusNor:AIMA03])

Linear Classifiers with a hard Threshold

- **Idea:** The result of linear regression can be used for classification.
- **Example 7.13 (Nuclear Test Ban Verification).**

Plots of seismic data parameters:
body wave magnitude x_1 vs.
surface wave magnitude x_2 .
White: earthquakes, black:
underground explosions

Also: h_{w^*} as a decision boundary
 $x_2 = 17x_1 - 4.9$.

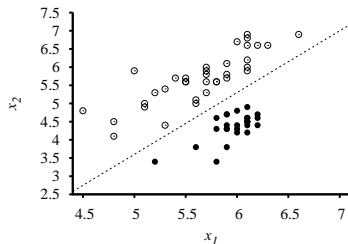


Linear Classifiers with a hard Threshold

- **Idea:** The result of linear regression can be used for classification.
- **Example 7.16 (Nuclear Test Ban Verification).**

Plots of seismic data parameters:
body wave magnitude x_1 vs.
surface wave magnitude x_2 .
White: earthquakes, black:
underground explosions

Also: h_{w^*} as a decision boundary
 $x_2 = 17x_1 - 4.9$.



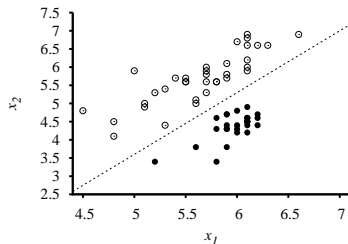
- **Definition 7.17.** A **decision boundary** is a line (or a surface, in higher dimensions) that separates two classes of points. A linear **decision boundary** is called a **linear separator** and data that admits one are called **linearly separable**.
- **Example 7.18 (Nuclear Tests continued).** The **linear separator** for ?? is defined by $-4.9 + 1.7x_1 - x_2 = 0$, explosions are characterized by $-4.9 + 1.7x_1 - x_2 > 0$, earthquakes by $-4.9 + 1.7x_1 - x_2 < 0$.

Linear Classifiers with a hard Threshold

- **Idea:** The result of linear regression can be used for classification.
- **Example 7.19 (Nuclear Test Ban Verification).**

Plots of seismic data parameters:
body wave magnitude x_1 vs.
surface wave magnitude x_2 .
White: earthquakes, black:
underground explosions

Also: h_{w^*} as a decision boundary
 $x_2 = 17x_1 - 4.9$.



- **Definition 7.20.** A **decision boundary** is a line (or a surface, in higher dimensions) that separates two classes of points. A linear **decision boundary** is called a **linear separator** and data that admits one are called **linearly separable**.
- **Example 7.21 (Nuclear Tests continued).** The **linear separator** for ?? is defined by $-4.9 + 1.7x_1 - x_2 = 0$, explosions are characterized by $-4.9 + 1.7x_1 - x_2 > 0$, earthquakes by $-4.9 + 1.7x_1 - x_2 < 0$.
- **Useful Trick:** If we introduce dummy coordinate $x_0 = 1$, then we can write the **classification hypothesis** as $h_w(x) = 1$ if $w \cdot x > 0$ and 0 otherwise.

Linear Classifiers with a hard Threshold (Perceptron Rule)

- ▶ So $h_w(x) = 1$ if $w \cdot x > 0$ and 0 otherwise is well-defined, how to choose w ?
- ▶ Think of $h_w(x) = \mathcal{T}(w \cdot x)$, where $\mathcal{T}(z) = 1$, if $z > 0$ and $\mathcal{T}(z) = 0$ otherwise. We call \mathcal{T} a **threshold function**.
- ▶ **Problem:** \mathcal{T} is not differentiable and $\frac{\partial \mathcal{T}}{\partial z} = 0$ where defined \leadsto
 - ▶ No closed-form solutions by setting $\frac{\partial \mathcal{T}}{\partial z} = 0$ and solving.
 - ▶ Gradient-descent methods in weight-space do not work either.
- ▶ We can learn weights by iterating over the following rule:
- ▶ **Definition 7.22.** Given an **example** (x, y) , the **perceptron learning rule** is

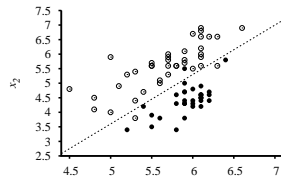
$$w_i \longleftarrow w_i + \alpha \cdot (y - h_w(x)) \cdot x_i$$

- ▶ as we are considering 0/1 **classification**, there are three possibilities:
 1. If $y = h_w(x)$, then w_i remains unchanged.
 2. If $y = 1$ and $h_w(x) = 0$, then w_i is in/decreased if x_i is **positive/negative**. (we want to make $w \cdot x$ bigger so that $\mathcal{T}(w \cdot x) = 1$)
 3. If $y = 0$ and $h_w(x) = 1$, then w_i is de/increased if x_i is **positive/negative**. (we want to make $w \cdot x$ smaller so that $\mathcal{T}(w \cdot x) = 0$)

Learning Curves for Linear Classifiers (Perceptron Rule)

► Example 7.23.

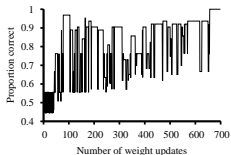
Learning curves (plots of total **training set** accuracy vs. number of iterations) for the perceptron rule on the earthquake/explosions data.



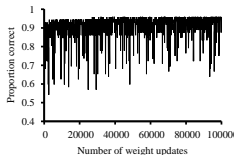
original data

noisy, non-separable data

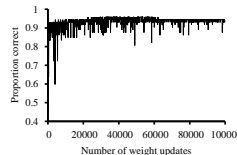
learning rate decay
 $\alpha(t) = 1000/(1000 + t)$



messy convergence
700 iterations



convergence failure
100,000 iterations



slow convergence
100,000 iterations

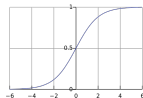
► **Theorem 7.24.** Finding the minimal-error **hypothesis** is **NP-hard**, but possible with **learning rate decay**.

Linear Classification with Logistic Regression

- ▶ **So far:** Passing the output of a linear function through a **threshold function** \mathcal{T} yields a linear classifier.
- ▶ **Problem:** The hard nature of \mathcal{T} brings problems:
 - ▶ \mathcal{T} is not differentiable nor continuous \leadsto learning via perceptron rule becomes unpredictable.
 - ▶ \mathcal{T} is “overly precise” near the boundary \Leftarrow need more graded judgments.
- ▶ **Idea:** Soften the threshold, approximate it with a differentiable function.

We use the **standard logistic function** $l(x) = \frac{1}{1+e^{-x}}$

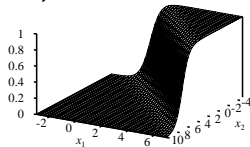
So we have $h_w(x) = l(w \cdot x) = \frac{1}{1+e^{-(w \cdot x)}}$



▶ Example 7.25 (Logistic Regression Hypothesis in Weight Space).

Plot of a **logistic regression hypothesis** for the earthquake/explosion data.

The value at (w_0, w_1) is the probability of belonging to the class labeled 1.



We speak of the **cliff** in the classifier intuitively.

- ▶ **Definition 7.26.** The process of **weight fitting** in $h_w(x) = \frac{1}{1+e^{-(w \cdot x)}}$ is called **logistic regression**.
- ▶ There is no easy closed form solution, but **gradient descent** is straightforward,
- ▶ As our **hypotheses** have continuous output, use the **squared error loss** function L_2 .
- ▶ For an **example** (x,y) we compute the **partial derivatives**: (via chain rule)

$$\begin{aligned}\frac{\partial}{\partial w_i} L_2(w) &= \frac{\partial}{\partial w_i} ((y - h_w(x))^2) \\ &= 2 \cdot h_w(x) \cdot \frac{\partial}{\partial w_i} (y - h_w(x)) \\ &= -2 \cdot h_w(x) \cdot l'(w \cdot x) \cdot \frac{\partial}{\partial w_i} (w \cdot x) \\ &= -2 \cdot h_w(x) \cdot l'(w \cdot x) \cdot x_i\end{aligned}$$

Logistic Regression (continued)

- ▶ The derivative of the **logistic function** satisfies $l'(z) = l(z)(1 - l(z))$, thus

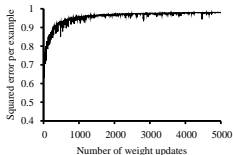
$$l'(w \cdot x) = l(w \cdot x)(1 - l(w \cdot x)) = h_w(x)(1 - h_w(x))$$

- ▶ **Definition 7.27.** The rule for **logistic update** (weight update for **minimizing** the loss) is

$$w_i \leftarrow w_i + \alpha \cdot (y - h_w(x)) \cdot h_w(x) \cdot (1 - h_w(x)) \cdot x_i$$

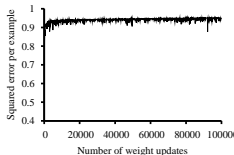
- ▶ **Example 7.28 (Redoing the Learning Curves).**

original data



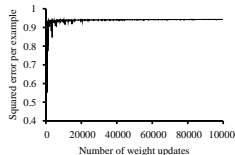
messy convergence
5000 iterations

noisy, non-separable data



convergence failure
100,000 iterations

learning rate decay
 $\alpha(t) = 1000/(1000 + t)$



slow convergence
100,000 iterations

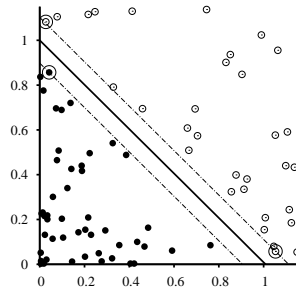
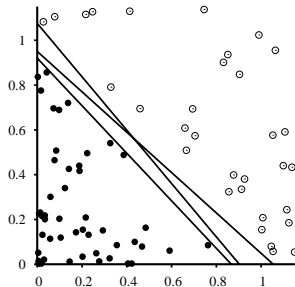
- ▶ **Upshot:** **Logistic update** seems to perform better than **perceptron update**.

27.8 Support Vector Machines

Support Vector Machines

Definition 8.1. Given a linearly separable data set E the maximum margin separator is the linear separator s that maximizes the margin, i.e. the distance of the E from s .

Example 8.2. All lines on the left are valid linear separators:



We expect the maximum margin separator on the right to generalize best

Note: To find the maximum margin separator, we only need to consider the innermost points (circled above).

Support Vector Machines (contd.)

Definition 8.3. Support-vector machines (SVMs; also support-vector networks) are supervised learning models for classification and regression.

SVMs construct a maximum margin separator by prioritizing critical examples (support vectors).

SVMs are still one of the most popular approaches for “off-the-shelf” supervised learning.

Setting:

- ▶ We have a training set $E = \{\langle x_1, y_1 \rangle, \dots, \langle x_n, y_n \rangle\}$ where $x_i \in \mathbb{R}^p$ and $y_i \in \{-1, 1\}$ (instead of $\{1, 0\}$)
- ▶ The goal is to find a *hyperplane* in \mathbb{R}^p that maximally separates the two classes (i.e. $y_i = -1$ from $y_i = 1$)

Remember A *hyperplane* can be represented as the set $\{x \mid (w \cdot x) + b = 0\}$ for some vector w and scalar b .
(w is orthogonal to the plane, b determines the offset from the origin)

Finding the Maximum Margin Separator (Separable Case)

Idea: The **margin** is bounded by the two hyperplanes described by $\{x \mid (w \cdot x) + b + 1 = 0\}$ (lower boundary) and $\{x \mid (w \cdot x) + b - 1 = 0\}$ (upper boundary).

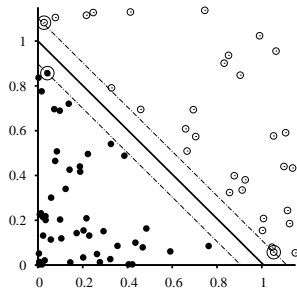
\Rightarrow The distance between them is $\frac{2}{\|w\|_2}$.

Constraints: To maximize the **margin**, minimize $\|w\|_2$ while keeping x_i out of the **margin**:

$(w \cdot x_i) + b \geq 1$ for $y_i = 1$ and $(w \cdot x_i) + b \leq -1$ for $y_i = -1$

$\leadsto y_i((w \cdot x_i) - b) \geq 1$ for $1 \leq i \leq n$.

\leadsto This is an optimization problem.



Theorem 8.4 (SVM equation). Let $\alpha = \operatorname{argmax}_{\alpha} \left(\sum_j \alpha_j - \frac{1}{2} \left(\sum_{j,k} \alpha_j \alpha_k y_j y_k (x_j \cdot x_k) \right) \right)$ under the

constraints $\alpha_j \geq 0$ and $\sum_j \alpha_j y_j = 0$.

The **maximum margin separator** is given by $w = \sum_j \alpha_j x_j$ and $b = w \cdot x_i - y_i$ for any x_i where $\alpha_i \neq 0$.

Proof sketch: By the duality principle for optimization problems

Finding the Maximum Margin Separator (Separable Case)

$$\alpha = \underset{\alpha}{\operatorname{argmax}} \left(\sum_j \alpha_j - \frac{1}{2} \left(\sum_{j,k} \alpha_j \alpha_k y_j y_k (x_j \cdot x_k) \right) \right), \text{ where } \alpha_j \geq 0, \quad \sum_j \alpha_j y_j = 0$$

Important Properties:

- ▶ The weights α_j associated with each data point are zero except at the **support vectors** (the points closest to the separator),
- ▶ The expression is **convex** \leadsto the single global maximum can be found **efficiently**,
- ▶ Data enter the expression only in the form of dot products of point pairs \leadsto once the optimal α_i have been calculated, we have $h(x) = \text{sign}(\sum_j \alpha_j y_j (x \cdot x_j) - b)$
- ▶ There are good software packages for solving such **quadratic programming** optimizations

Support Vector Machines (Kernel Trick)

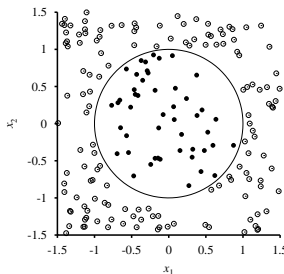
What if the data is not **linearly separable**?

Idea: Transform the data into a *feature space* where they are.

Definition 8.5. A **feature** for data in \mathbb{R}^p is a function $\mathbb{R}^p \rightarrow \mathbb{R}^q$.

Example 8.6 (Projecting Up a Non-Separable Data Set).

The true decision boundary is $x_1^2 + x_2^2 \leq 1$.



~> use the feature “distance from center”

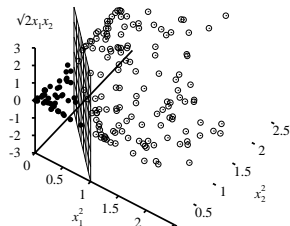
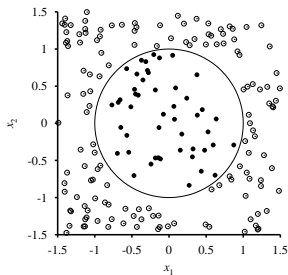
Support Vector Machines (Kernel Trick continued)

Idea: Replace $x_i \cdot x_j$ by some other product on the **feature space** in the **SVM equation**

Definition 8.7. A **kernel function** is a function $K: \mathbb{R}^p \times \mathbb{R}^p \rightarrow \mathbb{R}$ of the form $K(x_1, x_2) = \langle F(x_1), F(x_2) \rangle$ for some **feature** F and **inner product** $\langle \cdot, \cdot \rangle$ on the **codomain** of F .

Smart choices for a kernel function often allow us to compute $K(x_i, x_j)$ without needing to compute F at all.

Example 8.8. If we encode the distance from the center as the feature $F(x) = \langle x_1^2, x_2^2, \sqrt{2}x_1x_2 \rangle$ and define the kernel function as $K(x_i, x_j) = F(x_i) \cdot F(x_j)$, then this simplifies to $K(x_i, x_j) = (x_i \cdot x_j)^2$



Generally: We can learn non-linear separators by solving

$$\operatorname{argmax}_{\alpha} \left(\sum_j \alpha_j - \frac{1}{2} \left(\sum_{j,k} \alpha_j \alpha_k y_j y_k K(x_j, x_k) \right) \right)$$

where K is a **kernel function**

Definition 8.9. Let $X = \{x_1, \dots, x_n\}$. A **symmetric function** $K: X \times X \rightarrow \mathbb{R}$ is called **positive definite** iff the matrix $K_{i,j} = K(x_i, x_j)$ is a positive definite matrix.

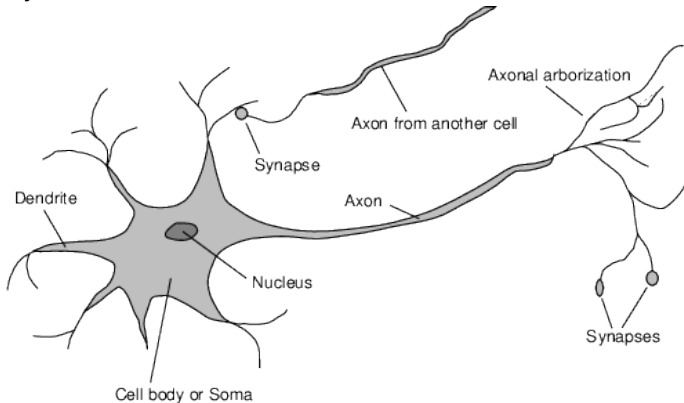
Theorem 8.10 (Mercer's Theorem). Every positive definite function K on X is a **kernel function** on X for some **feature** F .

Definition 8.11. The function $K(x_j, x_k) = (1 + (x_j \cdot x_k))^d$ is a **kernel function** corresponding to a feature space whose dimension is exponential in d . It is called the **polynomial kernel**.

27.9 Artificial Neural Networks

- ▶ Brains
- ▶ Neural networks
- ▶ Perceptrons
- ▶ Multilayer perceptrons
- ▶ Applications of neural networks

- ▶ **Axiom 9.1 (Neuroscience Hypothesis).** *Mental activity consists primarily of electrochemical activity in networks of brain cells called **neurons**.*



- ▶ **Definition 9.2.** The animal **brain** is a **biological neural network**
 - ▶ with 10^{11} **neurons** of > 20 types, 10^{14} synapses, $(1\text{ms}) - (10\text{ms})$ cycle time.
 - ▶ Signals are noisy “spike trains” of electrical potential.

Neural Networks as an approach to Artificial Intelligence

- ▶ One approach to artificial intelligence is to model and simulate brains. (and hope that AI comes along naturally)
- ▶ **Definition 9.3.** The AI subfield of neural networks (also called connectionism, parallel distributed processing, and neural computation) studies computing systems inspired by the biological neural networks that constitute brains.
- ▶ Neural networks are attractive computational devices, since they perform important AI tasks – most importantly learning and distributed, noise-tolerant computation – naturally and efficiently.

Neural Networks – McCulloch-Pitts “unit”

Definition 9.4. An artificial **neural network** is a **directed graph** such that every **edge** $a_i \rightarrow a_j$ is associated with a **weight** $w_{i,j} \in \mathbb{R}$, and each **node** a_j with parents a_1, \dots, a_n is associated with a function $f(w_{1,j}, \dots, w_{n,j}, x_1, \dots, x_n) \in \mathbb{R}$.

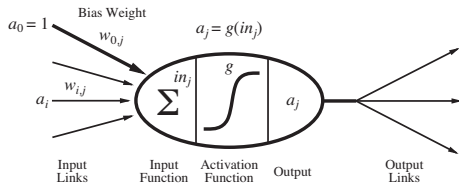
We call the output of a **node**’s function its **activation**, the **matrix** $w_{i,j}$ the **weight matrix**, the **nodes** **units** and the **edges** **links**.

In 1943 McCulloch and Pitts proposed a simple model for a **neuron**/brain:

Definition 9.5. A **McCulloch-Pitts unit** first computes a weighted sum of all inputs and then applies an **activation function** g to it.

$$\text{in}_i = \sum_j w_{j,i} a_j$$

$$a_i \leftarrow g(\text{in}_i) = g\left(+ \sum_j w_{j,i} a_j\right)$$

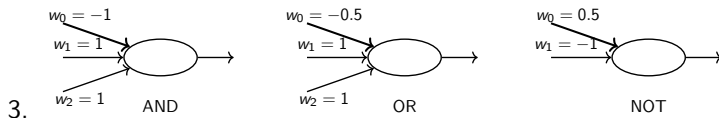


If g is a **threshold function**, we call the **unit** a **perceptron unit**, if g is a **logistic function** a **sigmoid perceptron unit**.

A **McCulloch-Pitts network** is a **neural network** with **McCulloch-Pitts units**.

Implementing Logical Functions as Units

- ▶ **McCulloch-Pitts units** are a gross oversimplification of real **neurons**, but its purpose is to develop understanding of what **neural networks** of simple **units** can do.
- ▶ **Theorem 9.6 (McCulloch and Pitts).** Every **Boolean function** can be **implemented** as **McCulloch-Pitts networks**.
- ▶ *Proof:* by construction
 1. Recall that $a_i \leftarrow g(\sum_j w_{j,i} a_j)$. Let $g(r) = 1$ iff $r > 0$, else 0.
 2. As for **linear regression** we use $a_0 = 1 \rightsquigarrow w_{0,i}$ as a **bias weight** (or **intercept**) (determines the threshold)



4. Any **Boolean function** can be **implemented** as a **DAG** of **McCulloch-Pitts units**.



- ▶ We have models for **neurons** \leadsto connect them to **neural networks**.
- ▶ **Definition 9.7.** A **neural network** is called a **feed-forward network**, if it is **acyclic**.
- ▶ **Intuition:** **Feed-forward networks implement functions**, they have no internal **state**.
- ▶ **Definition 9.8.** **Feed-forward networks** are usually organized in **layers**: a **n layer network** has a **partition** $\{L_0, \dots, L_n\}$ of the **nodes**, such that **edges** only connect **nodes** from subsequent **layer**. L_0 is called the **input layer** and its members **input units**, and L_n the **output layer** and its members **output units**. Any **unit** that is not in the **input layer** or the **output layer** is called **hidden**.

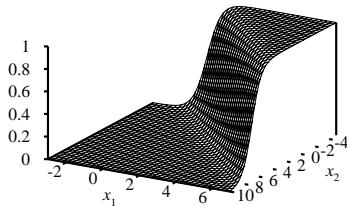
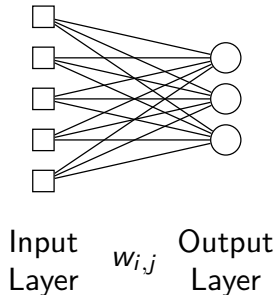
- ▶ **Definition 9.9.** A neural network is called recurrent (a RNNs), iff it has cycles.
 - ▶ Hopfield networks have symmetric weights ($w_{i,j} = w_{j,i}$) $g(x) = \text{sign}(x)$, $a_i = \pm 1$; (holographic associative memory)
 - ▶ Boltzmann machines use stochastic activation functions.
- ▶ Recurrent neural networks have cycles with delay \leadsto have internal state (like flip-flops), can oscillate etc.

Recurrent neural networks follow largely the same principles as feed-forward networks, so we will not go into details here.

Single-layer Perceptrons

► **Definition 9.10.** A **perceptron network** is a **feed-forward network** of **perceptron units**. A single layer perceptron network is called a **perceptron**.

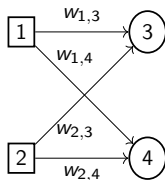
► **Example 9.11.**



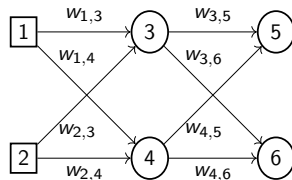
- All **input units** are directly connected to **output units**.
- **Output units** all operate separately, no shared weights \leadsto treat as the combination of n **perceptron units**.
- Adjusting weights moves the location, orientation, and steepness of **cliff**.

Feed-forward Neural Networks (Example)

- ▶ **Feed-forward network** $\hat{=}$ a parameterized family of nonlinear functions:
- ▶ **Example 9.12.** We show two **feed-forward networks**:



a) single layer (perceptron network)



b) 2 layer feed-forward network

$$\begin{aligned} a_5 &= g(w_{3,5} \cdot a_3 + w_{4,5} \cdot a_4) \\ &= g(w_{3,5} \cdot g(w_{1,3} \cdot a_1 + w_{2,3} a_2) + w_{4,5} \cdot g(w_{1,4} \cdot a_1 + w_{2,4} a_2)) \end{aligned}$$

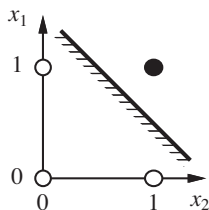
- ▶ **Idea:** Adjusting weights changes the function: do learning this way!

Expressiveness of Perceptrons

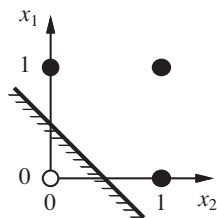
- ▶ Consider a **perceptron** with $g = \text{step function}$ (Rosenblatt, 1957, 1960)
- ▶ Can represent AND, OR, NOT, majority, etc., but not XOR
- ▶ Represents a **linear separator** in input space:

(and thus no adders)

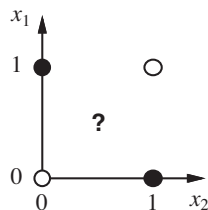
$$\sum_j w_j x_j > 0 \quad \text{or} \quad W, x \cdot > 0$$



(a) x_1 **and** x_2



(b) x_1 **or** x_2



(c) x_1 **xor** x_2

- ▶ Minsky & Papert (1969) pricked the first **neural network** balloon!

Perceptron Learning

For learning, we update the **weights** using **gradient descent** based on the **generalization loss** function.
Let e.g. $L(\mathbf{w}) = (y - h_{\mathbf{w}}(x))^2$ (the squared error loss).
We compute the gradient:

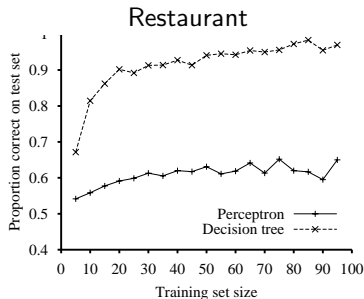
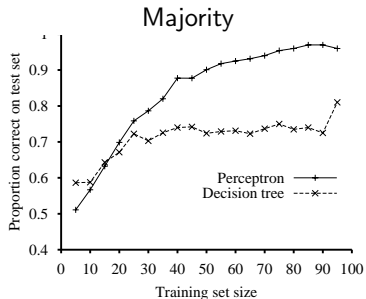
$$\begin{aligned}\frac{\partial L(\mathbf{w})}{\partial w_{j,k}} &= 2 \cdot (y_k - h_{\mathbf{w}}(x)_k) \cdot \frac{\partial (y - h_{\mathbf{w}}(x))}{\partial w_{j,k}} = 2 \cdot (y_k - h_{\mathbf{w}}(x)_k) \cdot \frac{\partial}{\partial w_{j,k}} \left(y - g\left(\sum_{j=0}^n w_{j,k} x_j\right) \right) \\ &= -2 \cdot (y_k - h_{\mathbf{w}}(x)_k) \cdot g'(\text{in}_k) \cdot x_j\end{aligned}$$

↪ Replacing the constant factor -2 by a learning rate parameter α we get the update rule:

$$w_{j,k} \leftarrow w_{j,k} + \alpha \cdot (y_k - h_{\mathbf{w}}(x)_k) \cdot g'(\text{in}_k) \cdot x_j$$

Perceptron learning contd.

The perceptron learning rule converges to a consistent function – *for any linearly separable data set*

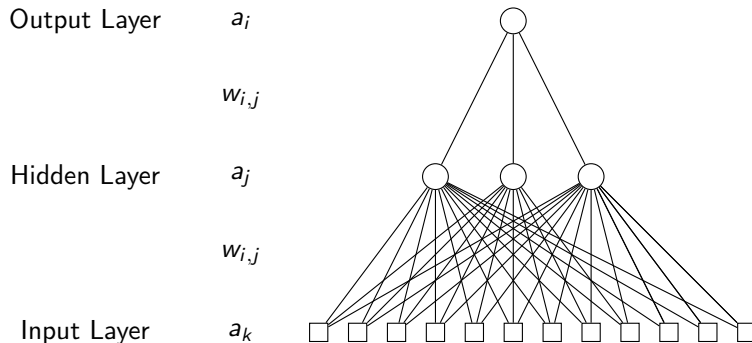


Perceptron learns the majority function easily, where DTL is hopeless.

Conversely, DTL learns the restaurant function easily, where a perceptron is hopeless. (not representable)

Multilayer perceptrons

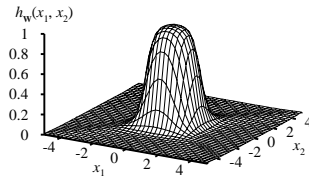
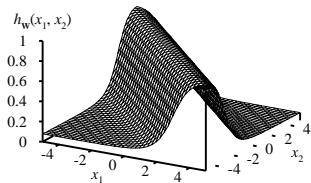
- **Definition 9.13.** In **multi layer perceptrons (MLPs)**, **layers** are usually fully connected; numbers of **hidden units** typically chosen by hand.



- **Definition 9.14.** Some **MLPs** have **residual connections**, i.e. connections that skip **layers**.

Expressiveness of MLPs

- ▶ All continuous functions w/ 2 layers, all functions w/ 3 layers.



- ▶ Combine two opposite-facing threshold functions to make a ridge.
- ▶ Combine two perpendicular ridges to make a bump.
- ▶ Add bumps of various sizes and locations to fit any surface.
- ▶ Proof requires exponentially many hidden units.

(cf. DTL proof)

Learning in Multilayer Networks

Note: The *output layer* of a multilayer **neural network** is a single-layer **perceptron** whose input is the output of the last hidden layer.

→ We can use the **perceptron** learning rule to update the weights of the output layer; e.g. for a squared error loss function: $w_{j,k} \leftarrow w_{j,k} + \alpha \cdot (y_k - h_w(x)_k) \cdot g'(\text{in}_k) \cdot a_j$

What about the hidden layers?

Idea: The hidden node j is “responsible” for some fraction of the error proportional to the weight $w_{j,k}$.

→ Back-propagate the error $\Delta_k = (y_k - h_w(x)_k) \cdot g'(\text{in}_k)$ from node k in the output layer to the hidden node j .

Let’s justify this:

$$\begin{aligned} \frac{\partial L(w)_k}{\partial w_{i,j}} &= -2 \cdot \underbrace{(y_k - h_w(x)_k) \cdot g'(\text{in}_k)}_{=:\Delta_k} \cdot \frac{\partial \text{in}_k}{\partial w_{i,j}} \quad (\text{as before}) \\ &= -2 \cdot \Delta_k \cdot \frac{\partial (\sum_{\ell} w_{\ell,k} a_{\ell})}{\partial w_{i,j}} = -2 \cdot \Delta_k \cdot w_{j,k} \cdot \frac{\partial a_j}{\partial w_{i,j}} = -2 \cdot \Delta_k \cdot w_{j,k} \cdot \frac{\partial g(\text{in}_j)}{\partial w_{i,j}} \\ &= -2 \cdot \underbrace{\Delta_k \cdot w_{j,k} \cdot g'(\text{in}_j)}_{=:\Delta_{j,k}} \cdot a_i \end{aligned}$$

Learning in Multilayer Networks (Hidden Layers)

$$\frac{\partial L(\mathbf{w})_k}{\partial \mathbf{w}_{i,j}} = -2 \cdot \underbrace{\Delta_k \cdot \mathbf{w}_{j,k} \cdot g'(\text{in}_j)}_{=:\Delta_{j,k}} \cdot a_i$$

Idea: The total “error” of the hidden node j is the sum of all the connected nodes k in the next layer

Definition 9.15. The **back-propagation rule** for **hidden** nodes of a **multilayer perceptron** is

$\Delta_j \leftarrow g'(\text{in}_j) \cdot \left(\sum_i \mathbf{w}_{j,i} \Delta_i \right)$ And the update rule for weights in a hidden layer is $\mathbf{w}_{k,j} \leftarrow \mathbf{w}_{k,j} + \alpha \cdot a_k \cdot \Delta_j$

Remark: Most neuroscientists deny that back-propagation occurs in the brain.

The back-propagation process can be summarized as follows:

1. Compute the Δ values for the output units, using the observed error.
2. Starting with output layer, repeat the following for each layer in the network, until the earliest hidden layer is reached:
 - 2.1 Propagate the Δ values back to the previous (hidden) layer.
 - 2.2 Update the weights between the two layers.

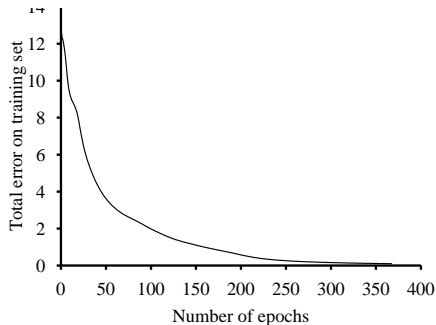
Backpropagation Learning Algorithm

► **Definition 9.16.** The **back-propagation learning algorithm** is given the following **pseudocode**

```
function BACK-PROP-LEARNING(examples, network) returns a neural network
  inputs: examples, a set of examples, each with input vector  $x$  and output vector  $y$ 
           network, a multilayer network with  $L$  layers, weights  $w_{i,j}$ , activation function  $g$ 
  local variables:  $\Delta$ , a vector of errors, indexed by network node
  foreach weight  $w_{i,j}$  in network do
     $w_{i,j} :=$  a small random number
  repeat
    foreach example  $(x, y)$  in examples do
      /* Propagate the inputs forward to compute the outputs */
      foreach node  $i$  in the input layer do  $a_i := x_i$ 
      for  $l = 2$  to  $L$  do
        foreach node  $j$  in layer  $l$  do
           $in_j := \sum_i w_{i,j} a_i$ 
           $a_j := g(in_j)$ 
      /* Propagate deltas backward from output layer to input layer */
      foreach node  $j$  in the output layer do  $\Delta[j] := g'(in_j) \cdot (y_j - a_j)$ 
      for  $l = L - 1$  to  $1$  do
        foreach node  $i$  in layer  $l$  do  $\Delta[i] := g'(in_i) \cdot (\sum_j w_{i,j} \Delta[j])$ 
      /* Update every weight in network using deltas */
      foreach weight  $w_{i,j}$  in network do  $w_{i,j} := w_{i,j} + \alpha \cdot a_i \cdot \Delta[j]$ 
  until some stopping criterion is satisfied
  return network
```

Back-Propagation – Properties

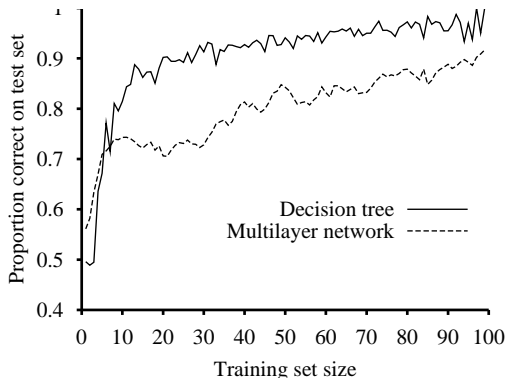
- ▶ Sum gradient updates for all **examples** in some “batch” and apply gradient descent.
- ▶ **Learning curve** for 100 restaurant **examples**: finds exact fit.



- ▶ **Typical problems:** slow convergence, **local minima**.

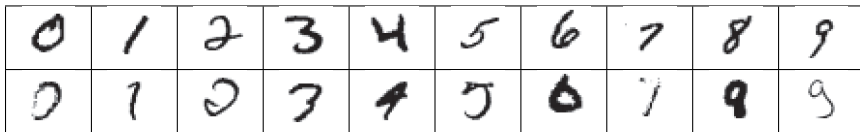
Back-Propagation – Properties (contd.)

- **Example 9.17.** Learning curve for MLPs with 4 hidden units:



- **Experience shows:** MLPs are quite good for complex pattern recognition tasks, but resulting hypotheses cannot be understood easily.
- This makes MLPs ineligible for some tasks, such as credit card and loan approvals, where law requires clear unbiased criteria.

Handwritten digit recognition



- ▶ 400–300–10 unit MLP = 1.6% error
- ▶ LeNet: 768–192–30–10 unit MLP = 0.9% error
- ▶ Current best (kernel machines, vision algorithms) \approx 0.6% error

- ▶ **neural networks** can be extremely powerful (hypothesis space intractably complex)
- ▶ **Perceptrons** (one-layer **networks**) insufficiently expressive for most applications
- ▶ **Multi-layer networks** are sufficiently expressive; can be trained by **gradient descent**, i.e., error back-propagation
- ▶ Many applications: speech, driving, handwriting, fraud detection, etc.
- ▶ Engineering, cognitive modelling, and neural system modelling subfields have largely diverged
- ▶ Drawbacks: take long to converge, require large amounts of data, and are difficult to *interpret* (Why is the output what it is?)

XKCD on Machine Learning

- **A Scepticists View:** see <https://xkcd.com/1838/>



Summary of Inductive Learning

- ▶ Learning needed for unknown environments, lazy designers.
- ▶ Learning agent = performance element + learning element.
- ▶ Learning method depends on type of performance element, available feedback, type of component to be improved, and its representation.
- ▶ For supervised learning, the aim is to find a simple hypothesis that is approximately consistent with training examples
- ▶ Decision tree learning using information gain.
- ▶ Learning performance = prediction accuracy measured on test set
- ▶ PAC learning as a general theory of learning boundaries.
- ▶ Linear regression (hypothesis space of univariate linear functions).
- ▶ Linear classification by linear regression with hard and soft thresholds.

Chapter 28

Statistical Learning

- ▶ **Definition 0.1.** **Statistical learning** has the goal to learn the correct **probability distribution** of a **random variable**.
- ▶ **Example 0.2.**
 - ▶ **Bayesian learning**, i.e. learning probabilistic models (e.g. the **CPTs** in **Bayesian networks**) from observations.
 - ▶ **Maximum *a posteriori*** and **maximum likelihood learning**
 - ▶ **Bayesian network learning**
 - ▶ ML Parameter Learning with Complete Data
 - ▶ Naive Bayes Models/Learning

28.1 Full Bayesian Learning

The Candy Flavors Example

► **Example 1.1.** Suppose there are five kinds of bags of candies:

1. 10% are h_1 : 100% cherry candies
2. 20% are h_2 : 75% cherry candies + 25% lime candies
3. 40% are h_3 : 50% cherry candies + 50% lime candies
4. 20% are h_4 : 25% cherry candies + 75% lime candies
5. 10% are h_5 : 100% lime candies

Then we observe candies drawn from some bag:

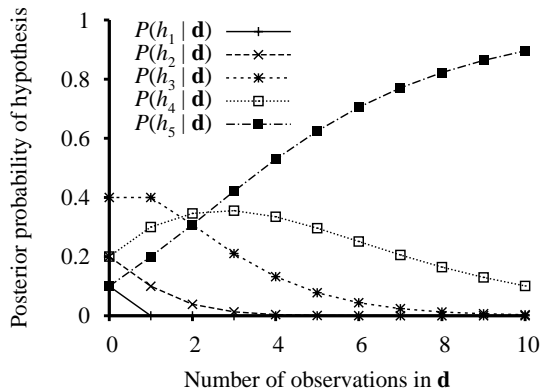


What kind of bag is it? What flavour will the next candy be?

Note: Every hypothesis is itself a probability distribution over the random variable “flavour”.

Candy Flavors: Posterior probability of hypotheses

- **Example 1.2.** Let d_i be the event that the i th drawn candy is green.
The probability of hypothesis h_i after n times are observed ($\hat{=} d_{1:n} =: d$) is



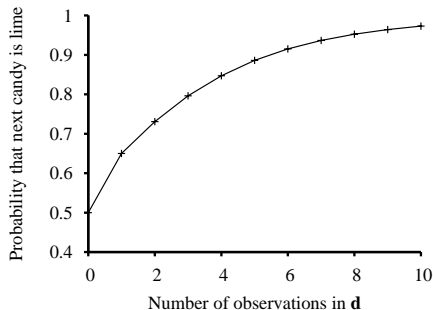
if the observations are IID, i.e. $P(d | h_i) = \prod_j P(d_j | h_i)$ and the hypothesis prior is as advertised. (e.g. $P(d | h_3) = 0.5^{10} = 0.1\%$)

The posterior probabilities start with the hypothesis priors, change with data.

Candy Flavors: Prediction Probability

- We calculate that the $n + 1$ -th candy is lime:

$$P(d_{n+1} = \text{lime} \mid \mathbf{d}) = \sum_i P(d_{n+1} = \text{lime} \mid h_i) \cdot P(h_i \mid \mathbf{d})$$



↪ we compute the expected value of *the probability of the next candy being lime* over all hypotheses (i.e. distributions).

↪ “meta-distribution”

- ▶ **Idea:** View **learning** as Bayesian updating of a probability distribution over the **hypothesis space**:
 - ▶ H is the **hypothesis** variable with values h_1, h_2, \dots and **prior** $\mathbb{P}(H)$.
 - ▶ j th observation d_j gives the **outcome** of **random variable** D_j .
 - ▶ $d := d_1, \dots, d_N$ constitutes the **training set** of a **inductive learning** problem.
- ▶ **Definition 1.3.** **Bayesian learning** calculates the probability of each **hypothesis** and makes predictions based on this:
 - ▶ Given the data so far, each **hypothesis** has a posterior probability:

$$P(h_i \mid d) = \alpha(P(d \mid h_i) \cdot P(h_i))$$

where $P(d \mid h_i)$ is called the **likelihood** (of the data under each **hypothesis**) and $P(h_i)$ the **hypothesis prior**.

- ▶ **Bayesian predictions** use a **likelihood-weighted** average over the **hypotheses**:

$$\mathbb{P}(X|d) = \sum_i \mathbb{P}(X|d, h_i) \cdot P(h_i \mid d) = \sum_i \mathbb{P}(X|h_i) \cdot P(h_i \mid d)$$

- ▶ **Observation:** No need to pick one best-guess **hypothesis** for **Bayesian predictions**! (and that is all an agent cares about)

- ▶ **Observation:** The Bayesian prediction eventually agrees with the true hypothesis.
 - ▶ The probability of generating “uncharacteristic” data indefinitely is vanishingly small.
 - ▶ *Proof sketch:* Argument analogous to PAC learning.
- ▶ **Problem:** Summing over the hypothesis space is often intractable.
- ▶ **Example 1.4.** There are $2^{2^6} = 18,446,744,073,709,551,616$ Boolean functions of 6 arguments.
- ▶ **Solution:** Approximate the learning methods to simplify.

28.2 Approximations of Bayesian Learning

Maximum A Posteriori (MAP) Approximation

- ▶ **Goal:** Get rid of summation over the space of all hypotheses in predictions.
- ▶ **Idea:** Make predictions wrt. the “most probable hypothesis”!
- ▶ **Definition 2.1.** For maximum a posteriori learning (MAP learning) choose the MAP hypothesis h_{MAP} that maximizes $P(h_i | d)$.
i.e., maximize $P(d | h_i) \cdot P(h_i)$ or (even better) $\log_2(P(d | h_i)) + \log_2(P(h_i))$.
- ▶ Predictions made according to a MAP hypothesis h_{MAP} are approximately Bayesian to the extent that $P(X|d) \approx P(X|h_{\text{MAP}})$.
- ▶ **Example 2.2.** In our candy example, $h_{\text{MAP}} = h_5$ after three limes in a row
 - ▶ a MAP learner then predicts that candy 4 is lime with probability 1.
 - ▶ compare with Bayesian prediction of 0.8. (see prediction curves above)
- ▶ As more data arrive, the MAP and Bayesian predictions become closer, because the competitors to the MAP hypothesis become less and less probable.
- ▶ For deterministic hypotheses, $P(d | h_i)$ is 1 if consistent, 0 otherwise
 \leadsto MAP = simplest consistent hypothesis. (cf. science)
- ▶ **Remark:** Finding MAP hypotheses is often much easier than Bayesian learning, because it requires solving an optimization problem instead of a large summation (or integration) problem.

Digression From MAP-learning to MDL-learning

- ▶ **Idea:** Reinterpret the log terms $\log_2(P(d | h_i)) + \log_2(P(h_i))$ in MAP learning:
 - ▶ Maximizing $P(d | h_i) \cdot P(h_i) \hat{=}$ minimizing $-\log_2(P(d | h_i)) - \log_2(P(h_i))$.
 - ▶ $-\log_2(P(d | h_i)) \hat{=}$ number of bits to encode data given hypothesis.
 - ▶ $-\log_2(P(h_i)) \hat{=}$ additional bits to encode hypothesis. (???)
- ▶ **Indeed** if hypothesis predicts the data exactly – e.g. h_5 in candy example – then $\log_2(1) = 0 \leadsto$ preferred hypothesis.
- ▶ This is more directly modeled by the following approximation to Bayesian learning:
- ▶ **Definition 2.3.** In minimum description length learning (MDL learning) the MDL hypothesis h_{MDL} minimizes the information entropy of the hypothesis likelihood.

- ▶ **Observation:** For large data sets, the prior becomes irrelevant. (we might not trust it anyways)
- ▶ **Idea:** Use this to simplify learning.
- ▶ **Definition 2.4.** Maximum likelihood learning (ML learning): choose the ML hypothesis h_{ML} maximizing $P(d \mid h_i)$. (simply get the best fit to the data)
- ▶ **Remark:** ML learning $\hat{=}$ MAP learning for a uniform prior. (reasonable if all hypotheses are of the same complexity)
- ▶ ML learning is the “standard” (non Bayesian) statistical learning method.

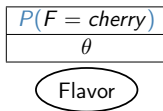
28.3 Parameter Learning for Bayesian Networks

ML Parameter Learning in Bayesian Nets

Bayesian networks (with continuous random variables) often feature nodes with a particular *parametric* distribution $D(\theta)$ (e.g. normal, binomial, Poisson, etc.).

How do we learn the parameters of these distributions from data?

Example 3.1. We get a candy bag from a new manufacturer; what is the fraction θ of cherry candies? (Note: We use the probability itself as the parameter. This is somewhat boring, but simple.)



New Facet: Any θ is possible: continuum of hypotheses h_θ

θ is a parameter for this simple (binomial) family of models; We call h_θ a **MLP hypothesis** and the process of learning θ **MLP learning**.

Example 3.2. Suppose we unwrap N candies, c cherries and $\ell = N - c$ limes. These are **IID**

observations, so the **likelihood** is $P(d \mid h_\theta) = \prod_{j=1}^N P(d_j \mid h_\theta) = \theta^c \cdot (1 - \theta)^\ell$

ML Parameter Learning in Bayes Nets

Trick: When optimizing a product, optimize the **logarithm** instead! ($\log_2(!)$ is monotone and turns products into sums)

Definition 3.3. The **log likelihood** is the **binary logarithm** of the **likelihood**. $L(d|h) := \log_2(P(d|h))$

Example 3.4. Compute the **log likelihood** as (using ??)

$$L(d|h_\theta) = \log_2(P(d|h_\theta)) = \sum_{j=1}^N \log_2(P(d_j|h_\theta)) = c \log_2(\theta) + \ell \log_2(1 - \theta)$$

Maximize this w.r.t. θ

$$\frac{\partial}{\partial \theta}(L(d|h_\theta)) = \frac{c}{\theta} - \frac{\ell}{1 - \theta} = 0 \leadsto \theta = \frac{c}{c + \ell} = \frac{c}{N}$$

In English: h_θ asserts that the actual proportion of cherries in the bag is equal to the observed proportion in the candies unwrapped so far! (...exactly what we should expect!) (\Rightarrow Generalize to more interesting parametric models later)

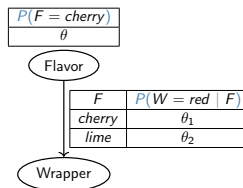
Warning: This causes problems with 0 counts!

► Cooking Recipe:

1. Write down an expression for the **likelihood** of the data as a function of the parameter(s).
2. Write down the derivative of the **log likelihood** with respect to each parameter.
3. Find the parameter values such that the derivatives are zero

Multiple Parameters Example

- **Example 3.5.** Red/green wrapper depends probabilistically on flavour:



- Likelihood for, e.g., cherry candy in green wrapper:

$$\begin{aligned} P(F = \text{cherry}, W = \text{green} \mid h_{\theta, \theta_1, \theta_2}) \\ &= P(F = \text{cherry} \mid h_{\theta, \theta_1, \theta_2}) \cdot P(W = \text{green} \mid F = \text{cherry}, h_{\theta, \theta_1, \theta_2}) \\ &= \theta \cdot (1 - \theta_1) \end{aligned}$$

- **Observation:** For N candies, r_c red-wrapped cherry candies, etc. we have

$$P(d \mid h_{\theta, \theta_1, \theta_2}) = \theta^c \cdot (1 - \theta)^\ell \cdot \theta_1^{r_c} \cdot (1 - \theta_1)^{g_c} \cdot \theta_2^{r_\ell} \cdot (1 - \theta_2)^{g_\ell}$$

Multiple Parameters Example (contd.)

- ▶ Minimize the log likelihood:

$$\begin{aligned} L &= c \log_2(\theta) + \ell \log_2(1 - \theta) \\ &+ r_c \log_2(\theta_1) + g_c \log_2(1 - \theta_1) \\ &+ r_\ell \log_2(\theta_2) + g_\ell \log_2(1 - \theta_2) \end{aligned}$$

- ▶ Derivatives of L contain only the relevant parameter:

$$\begin{aligned} \frac{\partial L}{\partial \theta} &= \frac{c}{\theta} - \frac{\ell}{1-\theta} = 0 \quad \leadsto \quad \theta = \frac{c}{c+\ell} \\ \frac{\partial L}{\partial \theta_1} &= \frac{r_c}{\theta_1} - \frac{g_c}{1-\theta_1} = 0 \quad \leadsto \quad \theta_1 = \frac{r_c}{r_c+g_c} \\ \frac{\partial L}{\partial \theta_2} &= \frac{r_\ell}{\theta_2} - \frac{g_\ell}{1-\theta_2} = 0 \quad \leadsto \quad \theta_2 = \frac{r_\ell}{r_\ell+g_\ell} \end{aligned}$$

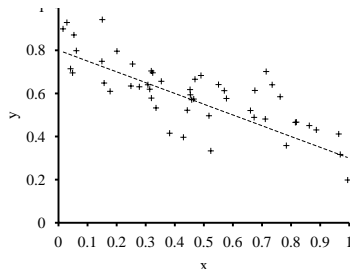
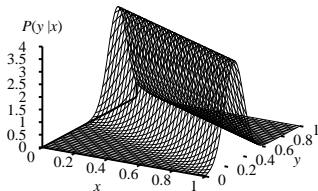
- ▶ **Upshot:** With complete data, *parameters can be learned separately* in Bayesian networks.
- ▶ **Remaining Problem:** Have to be careful with zero values! (division by zero)

Example: Linear Gaussian Model

A continuous **random variable** Y has the *linear-Gaussian distribution* with respect to a continuous **random variable** X , if the outcome of Y is determined by a linear function of the outcome of X *plus gaussian noise with a fixed variance* σ , i.e.

$$P(y_1 \leq Y \leq y_2 \mid X = x) = \int_{y_1}^{y_2} N(y; \theta_1 x + \theta_2, \sigma^2) dy = \int_{y_1}^{y_2} \frac{1}{\sigma \sqrt{2\pi}} \cdot e^{-\frac{1}{2} \cdot \left(\frac{y - (\theta_1 x + \theta_2)}{\sigma} \right)^2} dy$$

\leadsto assuming σ given, we have two parameter θ_1 and $\theta_2 \leadsto$ Hypothesis space is $\mathbb{R} \times \mathbb{R}$



Example: Linear Gaussian Model

$$P(y_1 \leq Y \leq y_2 \mid X = x) = \int_{y_1}^{y_2} \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2} \cdot \left(\frac{y - (\theta_1 x + \theta_2)}{\sigma}\right)^2} dy$$

↪ Given observations $X = X, Y = y$, maximize $\prod_{i=1}^N \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(y_i - (\theta_1 x_i + \theta_2))^2}{2\sigma^2}}$ w.r.t. θ_1, θ_2 . (we can ignore the integral for this)

Using the **log likelihood**, this is equivalent to **minimizing** $\sum_{i=1}^N (y_i - (\theta_1 x_i + \theta_2))^2$

↪ **minimizing** the sum of squared errors gives the ML solution

- ▶ Full Bayesian learning gives best possible predictions but is intractable.
- ▶ MAP learning balances complexity with accuracy on training data.
- ▶ Maximum likelihood learning assumes uniform prior, OK for large data sets:
 1. Choose a parameterized family of models to describe the data.
 ↪ requires substantial insight and sometimes new models.
 2. Write down the likelihood of the data as a function of the parameters.
 ↪ may require summing over hidden variables, i.e., inference.
 3. Write down the derivative of the log likelihood w.r.t. each parameter.
 4. Find the parameter values such that the derivatives are zero.
 ↪ may be hard/impossible; modern optimization techniques help.
- ▶ Naive Bayes models as a fall-back solution for machine learning:
 - ▶ conditional independence of all attributes as simplifying assumption.

Chapter 29

Reinforcement Learning

29.1 Reinforcement Learning: Introduction & Motivation

- ▶ **So far:** We have studied “learning from examples”. (functions, logical theories, probability models)
- ▶ **Now:** How can agents learn “what to do” in the absence of labeled examples of “what to do”. We call this problem unsupervised learning.
- ▶ **Example 1.1 (Playing Chess).** Learn transition models for own moves and maybe predict opponent’s moves.
- ▶ **Problem:** The agent needs to have some feedback about what is good/bad
↪ cannot decide “what to do” otherwise. (recall: external performance standard for learning agents)
- ▶ **Example 1.2.** The ultimate feedback in chess is whether you win, lose, or draw.
- ▶ **Definition 1.3.** We call a learning situation where there are no labeled examples unsupervised learning and the feedback involved a reward or reinforcement.
- ▶ **Example 1.4.** In soccer, there are intermediate reinforcements in the shape of goals, penalties, ...

Reinforcement Learning as Policy Learning

- ▶ **Definition 1.5.** Reinforcement learning is a type of unsupervised learning where an agent learns how to behave in an environment by performing actions and seeing the results.
- ▶ **Recap:** In ??? we introduced rewards as parts of MDPs (Markov decision processes) to define optimal policies.
 - ▶ an optimal policy maximizes the expected total reward.
- ▶ **Idea:** The task of reinforcement learning is to use observed rewards to come up with an optimal policy.
- ▶ In MDPs, the agent has total knowledge about the environment *and the reward function*, in reinforcement learning we do not assume this. (↪ POMDPs+reward-learning)
- ▶ **Example 1.6.** You play a game without knowing the rules, and at some time the opponent shouts “you lose!”

- ▶ **Reinforcement Learning solves all of AI:** An **agent** is placed in an **environment** and must learn to behave successfully therein.
- ▶ **KISS:** We will only look at simple **environments** and simple **agent** designs:
 - ▶ A **utility-based agent** learns a **utility function** on **states** and uses it to select **actions** that **maximize** the expected outcome **utility**. (passive learning)
 - ▶ A **Q-learning agent** learns an **action-utility function**, or **Q-function**, giving the **expected utility** of taking a given action in a given state. (active learning)
 - ▶ A **reflex agent** learns a **policy** that maps directly from **states** to **actions**.

29.2 Passive Learning

- ▶ **Definition 2.1 (To keep things simple).** Agent uses a state-based representation in a **fully observable environment**:
 - ▶ In **passive learning**, the agent's **policy** π is fixed: in state s , it always executes the action $\pi(s)$.
 - ▶ Its goal is simply to learn how good the **policy** is – that is, to learn the **utility function** $U^\pi(s)$.
- ▶ The **passive learning** task is similar to the **policy evaluation** task (part of the **policy iteration algorithm**) but the agent does not know
 - ▶ the **transition model** $P(s' \mid s, a)$, which specifies the probability of reaching state s' from state s after doing action a ,
 - ▶ the reward function $R(s)$, which specifies the reward for each state.

Passive Learning by Example

- **Example 2.2 (Passive Learning).** We use the 4×3 world introduced above

3	→	→	→	+1
2	↑		↑	-1
1	↑	←	←	←
	1	2	3	4

Optimal Policy π

3	0.812	0.868	0.918	+1
2	0.762		0.660	-1
1	0.705	0.655	0.611	0.388
	1	2	3	4

Utilities, given π

- The agent executes a set of **trials** in the **environment** using its **policy** π .
- In each **trial**, the agent starts in **state** (1,1) and experiences a sequence of state transitions until it reaches one of the terminal states, (4,2) or (4,3).
- Its percepts supply both the current state and the **reward** received in that **state**.

► **Example 2.3.** Typical trials might look like this:

1. $(1, 1)_{-0.4} \rightsquigarrow (1, 2)_{-0.4} \rightsquigarrow (1, 3)_{-0.4} \rightsquigarrow (1, 2)_{-0.4} \rightsquigarrow (1, 3)_{-0.4} \rightsquigarrow (2, 3)_{-0.4} \rightsquigarrow (3, 3)_{-0.4} \rightsquigarrow (4, 3)_{+1}$
2. $(1, 1)_{-0.4} \rightsquigarrow (1, 2)_{-0.4} \rightsquigarrow (1, 3)_{-0.4} \rightsquigarrow (2, 3)_{-0.4} \rightsquigarrow (3, 3)_{-0.4} \rightsquigarrow (3, 2)_{-0.4} \rightsquigarrow (3, 3)_{-0.4} \rightsquigarrow (4, 3)_{+1}$
3. $(1, 1)_{-0.4} \rightsquigarrow (2, 1)_{-0.4} \rightsquigarrow (3, 1)_{-0.4} \rightsquigarrow (3, 2)_{-0.4} \rightsquigarrow (4, 2)_{-1}$.

► **Definition 2.4.** The utility is defined to be the expected sum of (discounted) rewards obtained if policy π is followed.

$$U^\pi(s) := E \left[\sum_{t=0}^{\infty} \gamma^t R(S_t) \right]$$

where $R(s)$ is the reward for a state, S_t (a random variable) is the state reached at time t when executing policy π , and $S_0 = s$.
(for 4×3 we take the discount factor $\gamma = 1$)

Direct Utility Estimation

- ▶ A simple method for direct utility estimation was invented in the late 1950s in the area of adaptive control theory.
- ▶ **Definition 2.5.** The utility of a state is the expected total reward from that state onward (called the expected **reward to go**).
- ▶ **Idea:** Each trial provides a sample of the **reward to go** for each state visited.
- ▶ **Example 2.6.** The first trial in ??? provides a sample total reward of 0.72 for state (1,1), two samples of 0.76 and 0.84 for (1,2), two samples of 0.80 and 0.88 for (1,3), ...
- ▶ **Definition 2.7.** The **direct utility estimation algorithm** cycles over **trials**, calculates the **reward to go** for each state, and updates the estimated utility for that state by keeping the running average for that for each state in a table.
- ▶ **Observation 2.8.** *In the limit, the sample average will converge to the true expectation (**utility**) from ???.*
- ▶ **Remark 2.9.** **Direct utility estimation** is just **supervised learning**, where each **example** has the state as input and the observed reward to go as output.
- ▶ **Upshot:** We have reduced reinforcement learning to an **inductive learning problem**.

Adaptive Dynamic Programming

- ▶ **Problem:** The utilities of states are not independent in **direct utility estimation**!
- ▶ The utility of each state equals its own reward plus the **expected utility** of its **successor states**.
- ▶ **So:** The utility values obey a **Bellman equation** for a fixed **policy** π .

$$U^\pi(s) = R(s) + \gamma \cdot \left(\sum_{s'} P(s' \mid s, \pi(s)) \cdot U^\pi(s') \right)$$

- ▶ **Observation 2.10.** *By ignoring the connections between states, **direct utility estimation** misses opportunities for learning.*
- ▶ **Example 2.11.** Recall **trial 2** in ???; state $(3,3)$ is new.
2 $(1,1)_{-0.4} \rightsquigarrow (1,2)_{-0.4} \rightsquigarrow (1,3)_{-0.4} \rightsquigarrow (2,3)_{-0.4} \rightsquigarrow (3,3)_{-0.4} \rightsquigarrow (3,2)_{-0.4} \rightsquigarrow (3,3)_{-0.4} \rightsquigarrow (4,3)_{+1}$
 - ▶ The next transition reaches $(3,3)$, (known high utility from trial 1)
 - ▶ Bellman equation: \rightsquigarrow high $U^\pi(3,2)$ because $(3,2)_{-0.4} \rightsquigarrow (3,3)$
 - ▶ But **direct utility estimation** learns nothing until the end of the **trial**.
- ▶ **Intuition:** **Direct utility estimation** searches for U in a **hypothesis space** that too large \Leftarrow many functions that violate the Bellman equations.
- ▶ Thus the **algorithm** often converges very slowly.

- ▶ **Idea:** Take advantage of the constraints among the utilities of states by
 - ▶ learning the **transition model** that connects them,
 - ▶ solving the corresponding **Markov decision process** using a dynamic programming method.

This means plugging the learned **transition model** $P(s'|s, \pi(s))$ and the observed rewards $R(s)$ into the **Bellman equations** to calculate the utilities of the states.

- ▶ **As above:** These equations are linear (no maximization involved) (**solve with any any linear algebra package**).
- ▶ **Observation 2.12.** *Learning the model itself is easy, because the environment is fully observable.*
- ▶ **Corollary 2.13.** *We have a **supervised learning** task where the input is a state–action pair and the output is the resulting state.*
 - ▶ *In the simplest case, we can represent the **transition model** as a table of probabilities.*
 - ▶ *Count how often each action outcome occurs and estimate the transition probability $P(s' | s, a)$ from the frequency with which s' is reached by action a in s .*
- ▶ **Example 2.14.** In the 3 **trials** from ???, *Right* is executed 3 times in (1, 3) and 2 times the result is (2, 3), so $P((2, 3) | (1, 3), \text{Right})$ is estimated to be $2/3$.

Passive ADP Learning Algorithm

► **Definition 2.15.** The **passive ADP algorithm** is given by

function PASSIVE-ADP-AGENT(percept) **returns** an action

inputs: percept, a percept indicating the current state s' and reward signal r'

persistent: π a fixed policy

mdp , an MDP with model P , rewards R , discount γ

U , a table of utilities, initially empty

N_{sa} , a table of frequencies **for** state-action pairs, initially zero

$N_{s'|sa}$, a table of outcome frequencies given state-action pairs, initially zero

s, a , the previous state and action, initially null

if s' is new **then** $U[s'] := r'$; $R[s'] := r'$

if s is not null **then**

increment $N_{sa}[s, a]$ and $N_{s'|sa}[s', s, a]$

for each t such that $N_{s|sa}[t, s, a]$ is nonzero **do**

$P(t|s, a) := N_{s'|sa}[t, s, a] / N_{sa}[s, a]$

$U := \text{POLICY-EVALUATION}(\pi, mdp)$

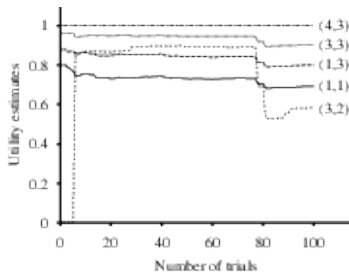
if $s'.\text{TERMINAL?}$ **then** $s, a := \text{null}$ **else** $s, a := s', \pi[s']$

return a

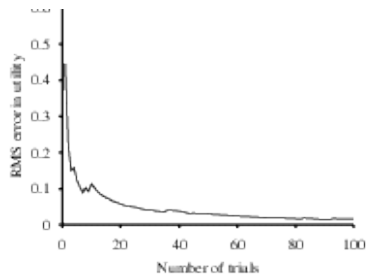
POLICY-EVALUATION computes $U^\pi(s) := E[\sum_{t=0}^{\infty} \gamma^t R(s_t)]$ in a MDP.

Passive ADP Convergence

- **Example 2.16 (Passive ADP learning curves for the 4x3 world).** Given the [optimal policy](#) from ???



utility estimates/trials



error for $U(1,1)$: 20 runs of 100 [trials](#)

Note the large changes occurring around the 78th [trial](#) – this is the first time that the agent falls into the -1 terminal state at (4,2).

- **Observation 2.17.** *The ADP agent is limited only by its ability to learn the transition model. ([intractable for large state spaces](#))*
- **Example 2.18.** In [backgammon](#), roughly 10^{50} equations in 10^{50} unknowns.
- **Idea:** Use this as a baseline to compare [passive learning algorithms](#)

29.3 Active Reinforcement Learning

Active Reinforcement Learning

- ▶ **Recap:** A passive learning agent has a fixed policy that determines its behavior.
- ▶ An active agent must also decide what actions to take.
- ▶ **Idea:** Adapt the passive ADP algorithm to handle this new freedom.
 - ▶ learn a complete model with outcome probabilities for all actions, rather than just the model for the fixed policy. (use PASSIVE-ADP-AGENT)
 - ▶ choose actions; the utilities to learn are defined by the optimal policy, they obey the Bellman equation:

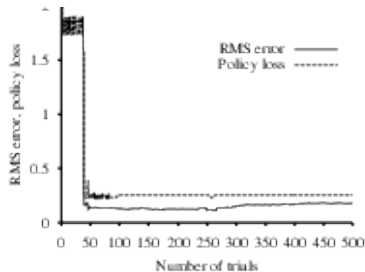
$$U(s) = R(s) + \gamma \cdot \max_{a \in A(s)} \left(\sum_{s'} U(s') \cdot P(s' \mid s, a) \right)$$

- ▶ solve with value/policy iteration techniques from ???.
- ▶ choose a good action, e.g.
 - ▶ by one-step lookahead to maximize expected utility, or
 - ▶ if agent uses policy iteration and has optimal policy, execute that.

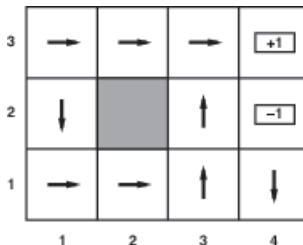
This agent/algorithm is greedy, since it only optimizes the next step!

Greedy ADP Learning (Evaluation)

► Example 3.1 (Greedy ADP learning curves for the 4x3 world).



average error/loss



suboptimal policy involved

The agent follows the **optimal policy** for the learned model at each step.

- It does not learn the true utilities or the true **optimal policy**!
- instead, in the 39th **trial**, it finds a **policy** that reaches the +1 reward along the lower route via (2,1), (3,1), (3,2), and (3,3).
- After experimenting with minor variations, from the 276th **trial** onward it sticks to that **policy**, never learning the **utilities** of the other states and never finding the optimal route via (1,2), (1,3), and (2,3).

Exploration in Active Reinforcement Learning

► **Observation 3.2.** *Greedy active ADP **learning agents** very seldom converge against the optimal solution*

- *The learned model is not the same as the true environment,*
- *What is optimal in the learned model need not be in the true environment.*

► What can be done? The agent does not know the true environment.

► **Idea:** **Actions** do more than provide **rewards** according to the learned model

- they also contribute to learning the true model by affecting the **percepts** received.
- By improving the model, the agent may reap greater **rewards** in the future.

► **Observation 3.3.** *An agent must make a tradeoff between*

- **exploitation** to **maximize** its **reward** as reflected in its current utility estimates and
- **exploration** to **maximize** its long term well-being.

*Pure **exploitation** risks getting stuck in a rut. Pure **exploration** to improve one's knowledge is of no use if one never puts that knowledge into practice.*

► Compare with the **information gathering agent** from ???.

Chapter 30

Knowledge in Learning

30.1 Logical Formulations of Learning

- ▶ **Recap:** Learning from *examples*. (last chapter)
- ▶ **Idea:** Construct a function with the input/output behavior observed in data.
- ▶ **Method:** Search for suitable functions in the *hypothesis space*. (e.g. decision trees)
- ▶ **Observation 1.1.** *Every learning task begins from zero. (except for the choice of hypothesis space)*
- ▶ **Problem:** We have to forget everything before we can learn something new.
- ▶ **Idea:** Utilize prior knowledge about the world! (represented e.g. in logic)

A logical Formulation of Learning

- ▶ **Recall:** Examples are composed of descriptions (of the input sample) and classifications.
- ▶ **Idea:** Represent examples and hypotheses as logical formulae.
- ▶ **Example 1.2.** For attribute-based representations, we can use PL^1 : we use predicate constants for Boolean attributes and classification and function constants for the other attributes.
- ▶ **Definition 1.3.** Logic based inductive learning tries to learn an hypothesis h that explains the classifications of the examples given their description, i.e. $h, \mathcal{D} \models \mathcal{C}$ (the explanation constraint), where
 - ▶ \mathcal{D} is the conjunction of the descriptions, and
 - ▶ \mathcal{C} the conjunction of their classifications.
- ▶ **Idea:** We solve the explanation constraint $h, \mathcal{D} \models \mathcal{C}$ for h where h ranges over some hypothesis space.
- ▶ **Refinement:** Use Occam's razor or additional constraints to avoid $h = \mathcal{C}$. (too easy otherwise/boring; see below)

A logical Formulation of Learning (Restaurant Examples)

- ▶ **Example 1.4 (Restaurant Example again).** Descriptions are **conjunctions** of **literals** built up from
 - ▶ predicates **Alt**, **Bar**, **Fri/Sat**, **Hun**, **Rain**, and **res**
 - ▶ equations about the functions **Pat**, **Price**, **Type**, and **Est**.

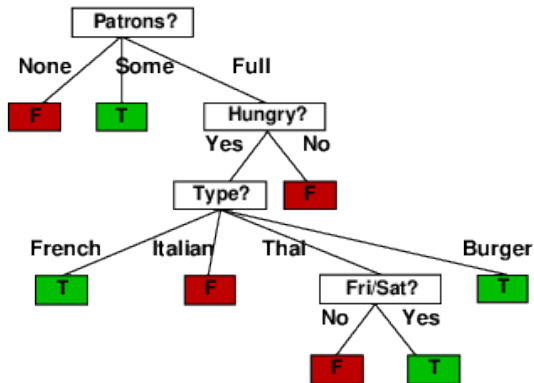
For instance the first example X_1 from ???, can be described as

$$\text{Alt}(X_1) \wedge \neg \text{Bar}(X_1) \wedge \text{Fri/Sat}(X_1) \wedge \text{Hun}(X_1) \wedge \dots$$

The **classification** is given by the goal predicate **WillWait**, in this case **WillWait**(X_1) or $\neg \text{WillWait}(X_1)$.

A logical Formulation of Learning (Restaurant Tree)

- **Example 1.5 (Restaurant Example again; Tree).** The induced decision tree from ???



A logical Formulation of Learning (Restaurant Tree)

- **Example 1.6 (Restaurant Example again; Tree).** The induced decision tree from ??? can be represented as

$$\begin{aligned}\forall r. \text{WillWait}(r) &\Leftrightarrow \text{Pat}(r, \text{Some}) \\ &\vee \text{Pat}(r, \text{Full}) \wedge \text{Hun}(r) \wedge \text{Type}(r, \text{French}) \\ &\vee \text{Pat}(r, \text{Full}) \wedge \text{Hun}(r) \wedge \text{Type}(r, \text{Thai}) \wedge \text{Fri/Sat}(r) \\ &\vee \text{Pat}(r, \text{Full}) \wedge \text{Hun}(r) \wedge \text{Type}(r, \text{Burger})\end{aligned}$$

Method: Construct a **disjunction** of all the **paths** from the **root** to the positive **leaves** interpreted as **conjunctions** of the **attributes** on the **path**.

Note: The **equivalence** takes care of **positive** and **negative examples**.

Cumulative Development

► **Example 1.7.** Learning from very few **examples** using **background knowledge**:

1. Caveman Zog and the fish on a stick:



"Hey! Look what Zog do!"

► **Example 1.9.** Learning from very few **examples** using **background knowledge**:

1. Caveman Zog and the fish on a stick:

2. Generalizing from one Brazilian:

Upon meeting her first Brazilian – Fernando – who speaks Portuguese, Sarah

► learns/generalizes that all Brazilians speak Portuguese,

► but not that all Brazilians are called Fernando.

► **Example 1.11.** Learning from very few examples using background knowledge:

1. Caveman Zog and the fish on a stick:
2. Generalizing from one Brazilian:
3. General rules about effectiveness of antibiotics:

When Sarah – gifted in diagnostics, but clueless in pharmacology – observes a doctor prescribing the antibiotic Proxadone for an inflamed foot, she learns/infers that Proxadone is effective against this ailment.

► **Observation:** The methods/algorithms from ??? cannot replicate this. (why?)

► **Example 1.13.** Learning from very few **examples** using **background knowledge**:

1. Caveman Zog and the fish on a stick:
2. Generalizing from one Brazilian:
3. General rules about **effectiveness** of antibiotics:

► **Observation:** The methods/algorithms from ??? cannot replicate this.

(why?)

► **Missing Piece:** The background knowledge!

► **Problem:** To use **background knowledge**, need a method to obtain it.

(use learning)

► **Question:** How to use knowledge to learn more **efficiently**?

Cumulative Development

► **Example 1.15.** Learning from very few examples using background knowledge:

1. Caveman Zog and the fish on a stick:
2. Generalizing from one Brazilian:
3. General rules about effectiveness of antibiotics:

► **Observation:** The methods/algorithms from ??? cannot replicate this.

(why?)

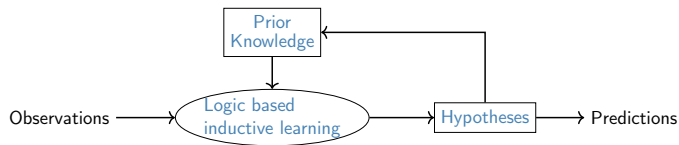
► **Missing Piece:** The background knowledge!

► **Problem:** To use background knowledge, need a method to obtain it.

(use learning)

► **Question:** How to use knowledge to learn more efficiently?

► **Answer:** Cumulative development: collect knowledge and use it in learning!



Cumulative Development

► **Example 1.17.** Learning from very few examples using background knowledge:

1. Caveman Zog and the fish on a stick:
2. Generalizing from one Brazilian:
3. General rules about effectiveness of antibiotics:

► **Observation:** The methods/algorithms from ??? cannot replicate this.

(why?)

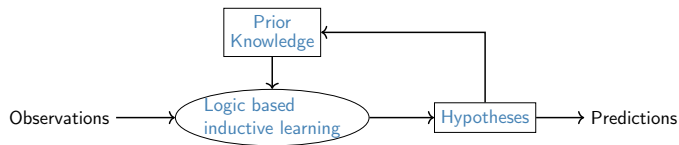
► **Missing Piece:** The background knowledge!

► **Problem:** To use background knowledge, need a method to obtain it.

(use learning)

► **Question:** How to use knowledge to learn more efficiently?

► **Answer:** Cumulative development: collect knowledge and use it in learning!



► **Definition 1.18.** We call the body of knowledge accumulated by (a group of) agents their background knowledge. It acts as prior knowledge in logic based learning processes.

- ▶ Explanation based learning (EBL)
- ▶ Relevance based learning (RBL)
- ▶ Knowledge based inductive learning (KBIL)

Three Principal Modes of Inference

► **Definition 1.19.** **Deduction** $\hat{=}$ knowledge extension

► **Example 1.20.**
$$\frac{\text{rains} \Rightarrow \text{wet_street} \quad \text{rains}}{\text{wet_street}} D$$

Three Principal Modes of Inference

► **Definition 1.25.** **Deduction** $\hat{=}$ knowledge extension

► **Example 1.26.**
$$\frac{\text{rains} \Rightarrow \text{wet_street} \quad \text{rains}}{\text{wet_street}} D$$

► **Definition 1.27.** **Abduction** $\hat{=}$ explanation

► **Example 1.28.**
$$\frac{\text{rains} \Rightarrow \text{wet_street} \quad \text{wet_street}}{\text{rains}} A$$

Three Principal Modes of Inference

- ▶ **Definition 1.31.** **Deduction** $\hat{=}$ knowledge extension
- ▶ **Example 1.32.**
$$\frac{\text{rains} \Rightarrow \text{wet_street} \quad \text{rains}}{\text{wet_street}} D$$
- ▶ **Definition 1.33.** **Abduction** $\hat{=}$ explanation
- ▶ **Example 1.34.**
$$\frac{\text{rains} \Rightarrow \text{wet_street} \quad \text{wet_street}}{\text{rains}} A$$
- ▶ **Definition 1.35.** **Induction** $\hat{=}$ learning general rules from examples
- ▶ **Example 1.36.**
$$\frac{\text{wet_street} \quad \text{rains}}{\text{rains} \Rightarrow \text{wet_street}} I$$

30.2 Inductive Logic Programming

- ▶ **Idea:** Background knowledge and new hypothesis combine to explain the examples.
- ▶ **Example 2.1.** Inferring disease D from the symptoms is not enough to explain the prescription of medicine M .
Need a new general rule: “ M is effective against D ” (induction from example)
- ▶ **Definition 2.2.** Knowledge based inductive learning (KBIL) replaces the explanation constraint by the KBIL constraint:

$$\text{Background} \wedge \text{Hypothesis} \wedge \text{Descriptions} \models \text{Classifications}$$

- ▶ **Definition 2.3.** Inductive logic programming (ILP) is logic based inductive learning method that uses logic programming as a uniform representation for examples, background knowledge and hypotheses. Given an encoding of the known background knowledge and a set of examples represented as a logical knowledge base of facts, an ILP system will derive a hypothesised logic program which entails all the positive and none of the negative examples.
- ▶ Main field of study for KBIL algorithms.
- ▶ Prior knowledge plays two key roles:
 1. The effective hypothesis space is reduced to include only those theories that are consistent with what is already known.
 2. Prior knowledge can be used to reduce the size of the hypothesis explaining the observations.
 - ▶ Smaller hypotheses are easier to find.
- ▶ **Observation:** ILP systems can formulate hypotheses in first-order logic.
 - ↪ Can learn in environments not understood by simpler systems.

- ▶ Combines **inductive methods** with the power of first-order representations.
- ▶ Offers a rigorous approach to the general KBIL problem.
- ▶ Offers complete **algorithms** for inducing general, first-order theories from **examples**.

30.2.1 An Example

- ▶ General knowledge-based induction problem

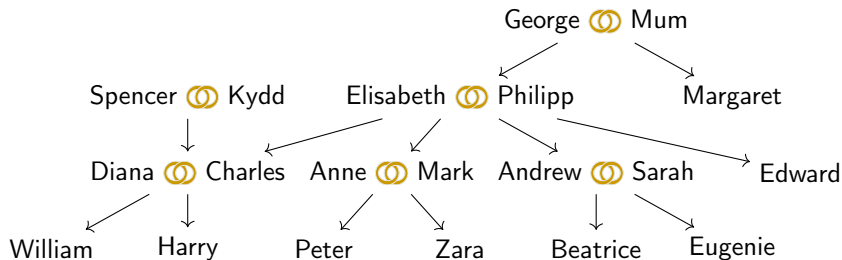
$$\text{Background} \wedge \text{Hypothesis} \wedge \text{Descriptions} \models \text{Classifications}$$

- ▶ **Example 2.4 (Learning family relations from examples).**

- ▶ Observations are an extended family tree
 - ▶ *mother*, *father* and *married* relations
 - ▶ *male* and *female* properties
- ▶ Target predicates: *grandparent*, *BrotherInLaw*, *Ancestor*
 - ↪ The goal is to find a logical formula that serves as a *definition* of the target predicates
 - ▶ equivalently: A *Prolog* program that *computes* the value of the target predicate
 - ↪ We obtain a perfectly comprehensible hypothesis

British Royalty Family Tree (not quite not up to date)

- The facts about kinship and relations can be visualized as a family tree:



Example

- ▶ Descriptions include facts like
 - ▶ `father(Philip, Charles)`
 - ▶ `mother(Mum, Margaret)`
 - ▶ `married(Diana, Charles)`
 - ▶ `male(Philip)`
 - ▶ `female(Beatrice)`
- ▶ Sentences in classifications depend on the target concept being learned (in the **example**: 12 **positive**, 388 **negative**)
 - ▶ `grandparent(Mum, Charles)`
 - ▶ `¬grandparent(Mum, Harry)`
- ▶ **Goal:** Find a set of sentences for **hypothesis** such that the **entailment** constraint is satisfied.
- ▶ **Example 2.5.** Without background knowledge, define **grandparent** in terms of **mother** and **father**.
$$\text{grandparent}(x, y) \Leftrightarrow (\exists z. \text{mother}(x, z) \wedge \text{mother}(z, y)) \vee (\exists z. \text{mother}(x, z) \wedge \text{father}(z, y)) \vee \dots \vee (\exists z. \text{father}(x, z) \wedge \text{father}(z, y))$$

Why Attribute-based Learning Fails

- ▶ **Observation:** Decision tree learning will get nowhere!
 - ▶ To express Grandparent as a (Boolean) attribute, pairs of people need to be objects *Grandparent*(*⟨Mum, Charles⟩*).
 - ▶ But then the *example* descriptions can not be represented

FirstElementIsMotherOfElizabeth(*⟨Mum, Charles⟩*)

- ▶ A large disjunction of specific cases without any hope of generalization to new *examples*.
- ▶ **Generally:** Attribute-based learning *algorithms* are incapable of learning relational predicates.

- **Observation:** A little bit of background knowledge helps a lot.
- **Example 2.6.** If the background knowledge contains

$$\text{parent}(x, y) \Leftrightarrow \text{mother}(x, y) \vee \text{father}(x, y)$$

then Grandparent can be reduced to

$$\text{grandparent}(x, y) \Leftrightarrow (\exists z. \text{parent}(x, z) \wedge \text{parent}(z, y))$$

- **Definition 2.7.** A **constructive induction algorithm** creates new predicates to facilitate the expression of explanatory **hypotheses**.
- **Example 2.8.** Use **constructive induction** to introduce a predicate **parent** to simplify the definitions of the target predicates.

30.2.2 Top-Down Inductive Learning: FOIL

- ▶ Bottom-up learning; e.g. Decision-tree learning: start from the observations and work backwards.
 - ▶ Decision tree is gradually grown until it is **consistent with** the observations.
- ▶ Top-down learning method
 - ▶ start from a general rule and specialize it on every example.

- ▶ Split **positive** and **negative examples**
 - ▶ **Positive:** $\langle \text{George}, \text{Anne} \rangle$, $\langle \text{Philip}, \text{Peter} \rangle$, $\langle \text{Spencer}, \text{Harry} \rangle$
 - ▶ **Negative:** $\langle \text{George}, \text{Elizabeth} \rangle$, $\langle \text{Harry}, \text{Zara} \rangle$, $\langle \text{Charles}, \text{Philip} \rangle$
- ▶ Construct a set of **Horn clauses** with **head** $\text{grandfather}(x, y)$ such that the **positive examples** are instances of the **grandfather** relationship.
 - ▶ Start with a **clause** with an empty body $\Rightarrow \text{grandfather}(x, y)$.
 - ▶ All **examples** are now classified as **positive**, so specialize to rule out the **negative examples**: Here are 3 potential additions:
 1. $\text{father}(x, y) \Rightarrow \text{grandfather}(x, y)$
 2. $\text{parent}(x, z) \Rightarrow \text{grandfather}(x, y)$
 3. $\text{father}(x, z) \Rightarrow \text{grandfather}(x, y)$
 - ▶ The first one incorrectly classifies the 12 **positive examples**.
 - ▶ The second one is incorrect on a larger part of the **negative examples**.
 - ▶ Prefer the third **clause** and specialize to $\text{father}(x, z) \wedge \text{parent}(z, y) \Rightarrow \text{grandfather}(x, y)$.

function Foil(*examples*, *target*) **returns** a set of Horn clauses

inputs: *examples*, set of examples

target, a literal **for** the goal predicate

local variables: *clauses*, set of clauses, initially empty

while *examples* contains positive examples **do**

clause := New-Clause(*examples*, *target*)

remove examples covered by clause from *examples*

add *clause* **to** *clauses*

return *clauses*

function New—Clause(*examples*,*target*) **returns** a Horn clause
local variables: *clause*, a clause with *target* as head and an empty body
I, a literal **to** be added **to** the clause
extendedExamples, a set of examples with values **for** new variables
extendedExamples := *examples*
while *extendedExamples* contains negative examples **do**
I := Choose—Literal(New—Literals(*clause*),*extendedExamples*)
append *I* **to** the body of *clause*
extendedExamples := map Extend—Example over *extendedExamples*
return *clause*

function Extend—Example(*example*,*literal*) **returns** a new example
if *example* satisfies *literal*
then return the set of examples created by extending *example* with each
possible constant value **for** each new variable **in** *literal*
else return the empty set

function New—Literals(*clause*) **returns** a set of possibly “useful” literals

function Choose—Literal(*literals*) **returns** the “best” literal from *literals*

FOIL: Choosing Literals

- ▶ New-Literals: Takes a **clause** and constructs all possibly “useful” **literals**
- ▶ $\text{father}(x, z) \Rightarrow \text{grandfather}(x, y)$
- ▶ Add **literals** using predicates
 - ▶ Negated or unnegated
 - ▶ Use any existing predicate (including the goal)
 - ▶ Arguments must be variables
 - ▶ Each **literal** must include at least one **variable** from an earlier **literal** or from the **head** of the **clause**
 - ▶ Valid: $\text{Mother}(z, u)$, $\text{Married}(z, z)$, $\text{grandfather}(v, x)$
 - ▶ Invalid: $\text{Married}(u, v)$
- ▶ Equality and inequality **literals**
 - ▶ E.g. $z \neq x$, empty list
- ▶ **Arithmetic** comparisons
 - ▶ E.g. $x > y$, threshold values

- ▶ The way New-Literal changes the **clauses** leads to a very large **branching factor**.
- ▶ Improve performance by using type information:
 - ▶ E.g., **parent**(x, n) where x is a person and n is a number
- ▶ Choose-Literal uses a **heuristic** similar to **information gain**.
- ▶ Ockham's razor to eliminate **hypotheses**.
 - ▶ If the **clause** becomes longer than the total length of the **positive examples** that the **clause** explains, this **clause** is not a valid **hypothesis**.
- ▶ Most impressive demonstration
 - ▶ Learn the correct definition of list-processing functions in Prolog from a small set of **examples**, using previously learned functions as background knowledge.

30.2.3 Inverse Resolution

- ▶ **Definition 2.9.** **Inverse resolution** in a nutshell
 - ▶ Classifications follows from $Background \wedge Hypothesis \wedge Descriptions$.
 - ▶ This can be proven by **resolution**.
 - ▶ Run the **proof** backwards to find **hypothesis**.

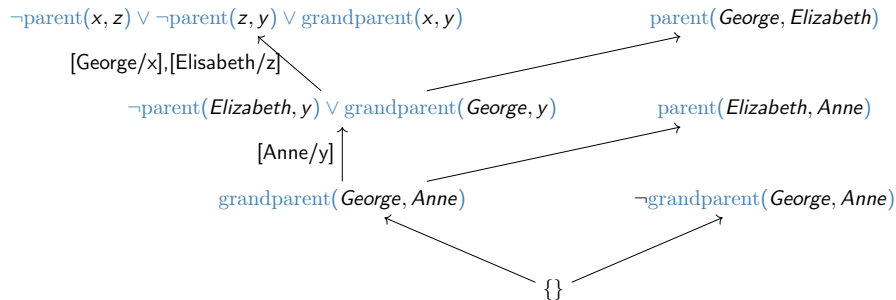
- ▶ **Definition 2.10.** *Inverse resolution* in a nutshell
 - ▶ Classifications follows from $Background \wedge Hypothesis \wedge Descriptions$.
 - ▶ This can be proven by *resolution*.
 - ▶ Run the *proof* backwards to find *hypothesis*.
- ▶ **Problem:** How to run the *resolution proof* backwards?

- ▶ **Definition 2.11.** **Inverse resolution** in a nutshell
 - ▶ Classifications follows from $Background \wedge Hypothesis \wedge Descriptions$.
 - ▶ This can be proven by **resolution**.
 - ▶ Run the **proof** backwards to find **hypothesis**.
- ▶ **Problem:** How to run the **resolution proof** backwards?
- ▶ **Recap:** In ordinary **resolution** we take two **clauses** $C_1 = L \vee R_1$ and $C_2 = \neg L \vee R_2$ and **resolve** them to produce the **resolvent** $C = R_1 \vee R_2$.
- ▶ **Idea:** Two possible variants of **inverse resolution**:
 - ▶ Take **resolvent** C and produce two **clauses** C_1 and C_2 .
 - ▶ Take C and C_1 and produce C_2 .

Generating Inverse Proofs (Example)

1. Start with an example classified as both positive and negative
2. Invent clauses that resolve with a fact in our knowledge base

(Need a contradiction)



$\neg\text{parent}(x, z) \vee \neg\text{parent}(z, y) \vee \text{grandparent}(x, y)$ is equivalent to
 $\text{parent}(x, z) \wedge \text{parent}(z, y) \Rightarrow \text{grandparent}(x, y)$

- ▶ **Inverse resolution** is a **search algorithm**: For any C and C_1 there can be several or even an **infinite** number of **clauses** C_2 .
- ▶ **Example 2.12.** Instead of **parent**(*George, Elizabeth*) there were numerous alternatives we could have picked!
- ▶ The **clauses** C_1 that participate in each step can be chosen from Background, Descriptions, Classifications or from hypothesized **clauses** already generated.
- ▶ **ILP** needs restrictions to make the search manageable
 - ▶ Eliminate function symbols
 - ▶ Generate only the most specific **hypotheses**
 - ▶ Use **Horn clauses**
 - ▶ All hypothesized **clauses** must be consistent with each other
 - ▶ Each hypothesized **clause** must agree with the observations

- ▶ An **inverse resolution** procedure is a complete **algorithm** for learning **first-order** theories:
 - ▶ If some unknown **hypothesis** generates a set of **examples**, then an **inverse resolution** procedure can generate **hypothesis** from the **examples**.
- ▶ Can **inverse resolution** infer the law of gravity from **examples** of falling bodies?
 - ▶ Yes, given suitable background **mathematics**!
- ▶ Monkey and typewriter problem: How to overcome the large **branching factor** and the lack of structure in the search space?

- ▶ Inverse resolution is capable of generating new predicates:
 - ▶ Resolution of C_1 and C_2 into C eliminates a **literal** that C_1 and C_2 share.
 - ▶ This **literal** might contain a predicate that does not appear in C .
 - ▶ When working backwards, one possibility is to generate a new predicate from which to construct the missing **literal**.

► **Example 2.13.**

$$\begin{array}{ccc} \text{Father}(\text{George}; y) \Rightarrow P(x, y) & & P(\text{George}; y) \Rightarrow \text{Ancestor}(\text{George}, y) \\ \text{[George/x]} \swarrow & & \nearrow \\ \text{Father}(\text{George}; y) \Rightarrow \text{Ancestor}(\text{George}, y) \end{array}$$

P can be used in later **inverse resolution** steps.

- **Example 2.14.** $\text{mother}(x, y) \Rightarrow P(x, y)$ or $\text{father}(x, y) \Rightarrow P(x, y)$ leading to the “Parent” relationship.
- Inventing new predicates is important to reduce the size of the definition of the goal predicate.
- Some of the deepest revolutions in science come from the invention of new predicates. (e.g. Galileo's invention of acceleration)

- ▶ ILP systems have outperformed knowledge free methods in a number of domains.
- ▶ Molecular biology: the GOLEM system has been able to generate high-quality predictions of protein structures and the therapeutic efficacy of various drugs.
- ▶ GOLEM is a completely general-purpose program that is able to make use of background knowledge about any domain.

Part 3

Natural Language

- ▶ **Definition 2.15.** A **natural language** is any form of **spoken** or signed means of **communication** that has evolved naturally in humans through use and repetition without conscious planning or premeditation.
- ▶ **In other words:** the language you use all day long, e.g. English, German, ...
- ▶ **Why Should we care about natural language?:**
 - ▶ Even more so than thinking, **language** is a skill that only humans have.
 - ▶ It is a miracle that we can express complex thoughts in a **sentence** in a matter of seconds.
 - ▶ It is no less miraculous that a child can learn tens of thousands of **words** and complex **syntax** in a matter of a few years.

- ▶ Without **natural language** capabilities (understanding and generation) no **AI**!
- ▶ Ca. 100.000 years ago, humans learned to speak, ca. 7.000 years ago, to write.
- ▶ Alan Turing based his **test** on **natural language**: (for good reason)
 - ▶ We want **AI agents** to be able to communicate with humans.
 - ▶ We want **AI agents** to be able to acquire knowledge from written **documents**.
- ▶ In this part, we analyze the problem with specific information-seeking tasks:
 - ▶ Language models (Which strings are English/Spanish/etc.)
 - ▶ Text classification (E.g. spam detection)
 - ▶ Information retrieval (aka. Search Engines)
 - ▶ Information extraction (finding objects and their relations in texts)

Chapter 31

Natural Language Processing

31.1 Introduction to NLP

What is Natural Language Processing?

- ▶ **Generally:** Studying of **natural languages** and development of systems that can use/generate these.
- ▶ **Definition 1.1.** **Natural language processing (NLP)** is an engineering field at the intersection of **computer science**, **AI**, and **linguistics** which is concerned with the **interactions** between **computers** and human (natural) languages. Most challenges in **NLP** involve:
 - ▶ **Natural language understanding (NLU)** that is, enabling **computers** to derive **meaning** (representations) from human or natural language input.
 - ▶ **Natural language generation (NLG)** which aims at generating **natural language** or **speech** from **meaning** representation.
- ▶ For communication with/among humans we need both **NLU** and **NLG**.

- ▶ Language Assistance:

- ▶ written language: Spell/grammar/style-checking,
- ▶ spoken language: dictation systems and screen readers,
- ▶ multilingual text: machine-supported text and dialog translation, eLearning.

- ▶ Language Assistance:

- ▶ written language: Spell/grammar/style-checking,
- ▶ spoken language: dictation systems and screen readers,
- ▶ multilingual text: machine-supported text and dialog translation, eLearning.

- ▶ Information management:

- ▶ search and classification of documents,
- ▶ information extraction, question answering.

(e.g. Google/Bing)
(e.g. <http://ask.com>)

► Language Assistance:

- written language: Spell/grammar/style-checking,
- spoken language: dictation systems and screen readers,
- multilingual text: machine-supported text and dialog translation, eLearning.

► Information management:

- search and classification of documents,
- information extraction, question answering.

(e.g. Google/Bing)
(e.g. <http://ask.com>)

► Dialog Systems/Interfaces:

- **information systems**: at airport, tele-banking, e-commerce, call centers,
- dialog interfaces for **computers**, robots, cars.

(e.g. Siri/Alexa)

- ▶ Language Assistance:
 - ▶ written language: Spell/grammar/style-checking,
 - ▶ spoken language: dictation systems and screen readers,
 - ▶ multilingual text: machine-supported text and dialog translation, eLearning.
- ▶ Information management:
 - ▶ search and classification of documents, (e.g. Google/Bing)
 - ▶ information extraction, question answering. (e.g. <http://ask.com>)
- ▶ Dialog Systems/Interfaces:
 - ▶ **information systems**: at airport, tele-banking, e-commerce, call centers,
 - ▶ dialog interfaces for **computers**, robots, cars. (e.g. Siri/Alexa)
- ▶ **Observation**: The earlier technologies largely rely on pattern matching, the latter ones need to compute the **meaning** of the input **utterances**, e.g. for **database** lookups in **information systems**.

31.2 Natural Language and its Meaning

What is (NL) Semantics? Answers from various Disciplines!

- **Observation:** Different (academic) disciplines specialize the notion of semantics (of natural language) in different ways.

What is (NL) Semantics? Answers from various Disciplines!

- ▶ **Observation:** Different (academic) disciplines specialize the notion of semantics (of natural language) in different ways.
 - ▶ **Philosophy:** has a long history of trying to answer it, e.g.
 - ▶ Platon \leadsto cave allegory, Aristotle \leadsto syllogisms.
 - ▶ Frege/Russell \leadsto sense vs. referent.
- (*"Michael Kohlhasé"* vs. *"Odysseus"*)

What is (NL) Semantics? Answers from various Disciplines!

- ▶ **Observation:** Different (academic) disciplines specialize the notion of semantics (of natural language) in different ways.
- ▶ **Philosophy:** has a long history of trying to answer it, e.g.
 - ▶ Platon \leadsto cave allegory, Aristotle \leadsto syllogisms.
 - ▶ Frege/Russell \leadsto sense vs. referent. (*"Michael Kohlhasé" vs. "Odysseus"*)
- ▶ **Linguistics/Language Philosophy:** We need semantics e.g. in translation
 - "*Der Geist ist willig aber das Fleisch ist schwach!*" vs.
 - "*Der Schnaps ist gut, aber der Braten ist verkocht!*" (*meaning counts*)

What is (NL) Semantics? Answers from various Disciplines!



- ▶ **Observation:** Different (academic) disciplines specialize the notion of semantics (of natural language) in different ways.
- ▶ **Philosophy:** has a long history of trying to answer it, e.g.
 - ▶ Platon \leadsto cave allegory, Aristotle \leadsto syllogisms.
 - ▶ Frege/Russell \leadsto sense vs. referent. (*"Michael Kohlhasé" vs. "Odysseus"*)
- ▶ **Linguistics/Language Philosophy:** We need semantics e.g. in translation
 - "*Der Geist ist willig aber das Fleisch ist schwach!*" vs.
 - "*Der Schnaps ist gut, aber der Braten ist verkocht!*" (*meaning counts*)
- ▶ **Psychology/Cognition:** Semantics $\hat{=}$ "what is in our brains" (\leadsto *mental models*)

What is (NL) Semantics? Answers from various Disciplines!

- ▶ **Observation:** Different (academic) disciplines specialize the notion of **semantics** (of **natural language**) in different ways.
- ▶ **Philosophy:** has a long history of trying to answer it, e.g.
 - ▶ Platon \leadsto cave allegory, Aristotle \leadsto **sylogisms**.
 - ▶ Frege/Russell \leadsto **sense** vs. **referent**. (*"Michael Kohlhasé" vs. "Odysseus"*)
- ▶ **Linguistics/Language Philosophy:** We need semantics e.g. in translation
 - "*Der Geist ist willig aber das Fleisch ist schwach!*" vs.
 - "*Der Schnaps ist gut, aber der Braten ist verkocht!*" (**meaning counts**)
- ▶ **Psychology/Cognition:** Semantics $\hat{=}$ "what is in our brains" (\leadsto **mental models**)
- ▶ **Mathematics** has driven much of modern logic in the quest for foundations.
 - ▶ Logic as "foundation of mathematics" solved as far as possible
 - ▶ In daily practice **syntax** and **semantics** are not differentiated (much).

What is (NL) Semantics? Answers from various Disciplines!

- ▶ **Observation:** Different (academic) disciplines specialize the notion of **semantics** (of **natural language**) in different ways.
- ▶ **Philosophy:** has a long history of trying to answer it, e.g.
 - ▶ Platon \leadsto cave allegory, Aristotle \leadsto **sylogisms**.
 - ▶ Frege/Russell \leadsto **sense** vs. **referent**. (*"Michael Kohlhasé" vs. "Odysseus"*)
- ▶ **Linguistics/Language Philosophy:** We need semantics e.g. in translation
 - "*Der Geist ist willig aber das Fleisch ist schwach!*" vs.
 - "*Der Schnaps ist gut, aber der Braten ist verkocht!*" (**meaning counts**)
- ▶ **Psychology/Cognition:** Semantics $\hat{=}$ "what is in our brains" (\leadsto **mental models**)
- ▶ **Mathematics** has driven much of modern logic in the quest for foundations.
 - ▶ Logic as "foundation of mathematics" solved as far as possible
 - ▶ In daily practice **syntax** and **semantics** are not differentiated (much).
- ▶ **Logic@AI/CS** tries to define **meaning** and **compute** with them. (**applied semantics**)
 - ▶ makes **syntax** explicit in a **formal language** (**formulae, sentences**)
 - ▶ defines **truth**/validity by mapping **sentences** into "world" (**interpretation**)
 - ▶ gives rules of **truth**-preserving **reasoning** (**inference**)

- ▶ **Idea:** Machine translation is very simple! (we have good lexica)
- ▶ **Example 2.1.** “*Peter liebt Maria.*” \leadsto “*Peter loves Mary.*”
- ▶  this only works for simple examples!
- ▶ **Example 2.2.** “*Wirf der Kuh das Heu über den Zaun.*” $\not\leadsto$ “*Throw the cow the hay over the fence.*”
(differing grammar; Google Translate)
- ▶ **Example 2.3.**  Grammar is not the only problem
 - ▶ “*Der Geist ist willig, aber das Fleisch ist schwach!*”
 - ▶ “*Der Schnaps ist gut, aber der Braten ist verkocht!*”
- ▶ **Observation 2.4.** We have to understand the *meaning* for high-quality translation!

- ▶ **Observation:** Humans use words (sentences, texts) in natural languages to represent and communicate information.
- ▶ **But:** What really counts is not the words themselves, but the meaning information they carry.

Language and Information

- **Observation:** Humans use **words** (**sentences**, **texts**) in **natural languages** to represent and communicate **information**.
- **But:** What really counts is not the **words** themselves, but the **meaning information** they carry.
- **Example 2.7 (Word Meaning).**

“*Newspaper*” ~→



- For questions/answers, it would be very useful to find out what **words** (**sentences/texts**) **mean**.

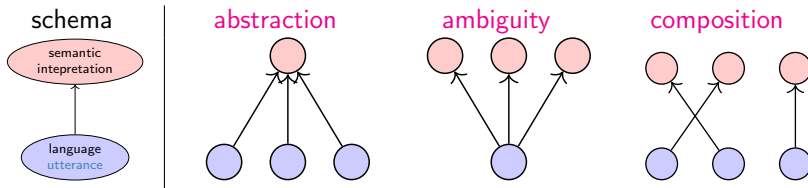
Language and Information

- **Observation:** Humans use **words** (**sentences**, **texts**) in **natural languages** to represent and communicate **information**.
- **But:** What really counts is not the **words** themselves, but the **meaning information** they carry.
- **Example 2.9 (Word Meaning).**

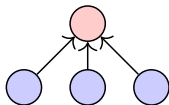
“*Newspaper*” \leadsto



- For questions/answers, it would be very useful to find out what **words** (**sentences/texts**) **mean**.
- **Definition 2.10.** Interpretation of **natural language utterances**: three problems

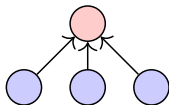


► **Example 2.11 (Abstraction).**



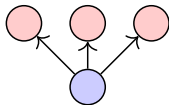
“*Car*” and “*automobile*” have the same meaning.

► **Example 2.14 (Abstraction).**



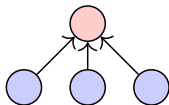
“*Car*” and “*automobile*” have the same meaning.

► **Example 2.15 (Ambiguity).**



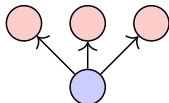
A “*bank*” can be a financial institution or a geographical feature.

► **Example 2.17 (Abstraction).**



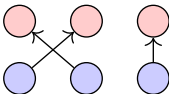
“*Car*” and “*automobile*” have the same meaning.

► **Example 2.18 (Ambiguity).**



A “*bank*” can be a financial institution or a geographical feature.

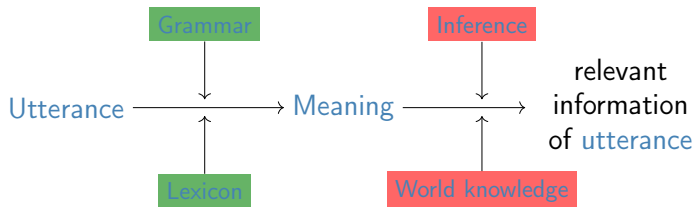
► **Example 2.19 (Composition).**



“*Every student sleeps*” $\leadsto \forall x. \textit{student}(x) \Rightarrow \textit{sleep}(x)$

Context Contributes to the Meaning of NL Utterances

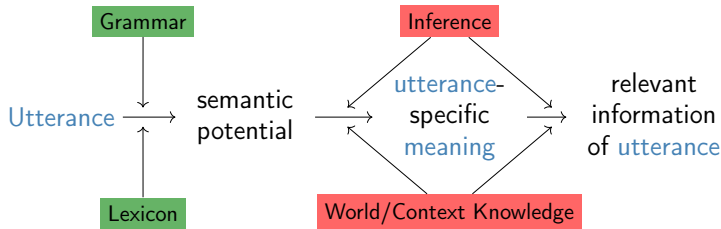
- **Observation:** Not all information conveyed is linguistically realized in an utterance.
- **Example 2.20.** “*The lecture begins at 11:00 am.*” What lecture? Today?
- **Definition 2.21.** We call a piece i of information linguistically realized in an utterance U , iff, we can trace i to a fragment of U .
- **Definition 2.22 (Possible Mechanism).** Inferring the missing pieces from the context and world knowledge:



We call this process semantic/pragmatic analysis.

Context Contributes to the Meaning of NL Utterances

- ▶ **Example 2.23.** “*It starts at eleven.*” What starts?
- ▶ Before we can resolve the time, we need to resolve the **anaphor** “*it*”.
- ▶ **Possible Mechanism:** More Inference!



~> Semantic/pragmatic analysis is quite complex!

(prime topic of AI-2)

31.3 Looking at Natural Language

► **Example 3.1.** We study the truth conditions of adjectival complexes:

► “*This is a diamond.*”

(\models *diamond*)

► **Example 3.2.** We study the truth conditions of adjectival complexes:

- “*This is a diamond.*”
- “*This is a blue diamond.*”

(\models *diamond*)

(\models *diamond*, \models *blue*)

► **Example 3.3.** We study the truth conditions of adjectival complexes:

- “*This is a diamond.*”
- “*This is a blue diamond.*”
- “*This is a big diamond.*”

(\models diamond)
(\models diamond, \models blue)
(\models diamond, $\not\models$ big)

► **Example 3.4.** We study the truth conditions of adjectival complexes:

- “*This is a diamond.*”
- “*This is a blue diamond.*”
- “*This is a big diamond.*”
- “*This is a fake diamond.*”

$(\models \text{diamond})$
 $(\models \text{diamond}, \models \text{blue})$
 $(\models \text{diamond}, \not\models \text{big})$
 $(\models \neg \text{diamond})$

► **Example 3.5.** We study the truth conditions of adjectival complexes:

- “*This is a diamond.*”
- “*This is a blue diamond.*”
- “*This is a big diamond.*”
- “*This is a fake diamond.*”
- “*This is a fake blue diamond.*”

(\models diamond)

(\models diamond, \models blue)

(\models diamond, $\not\models$ big)

($\models \neg$ diamond)

(\models blue?, \models diamond?)

► **Example 3.6.** We study the truth conditions of adjectival complexes:

► “*This is a diamond.*”

(\models diamond)

► “*This is a blue diamond.*”

(\models diamond, \models blue)

► “*This is a big diamond.*”

(\models diamond, $\not\models$ big)

► “*This is a fake diamond.*”

($\models \neg$ diamond)

► “*This is a fake blue diamond.*”

(\models blue?, \models diamond?)

► “*Mary knows that this is a diamond.*”

(\models diamond)

► **Example 3.7.** We study the truth conditions of adjectival complexes:

► “This is a diamond.”

(\models diamond)

► “This is a blue diamond.”

(\models diamond, \models blue)

► “This is a big diamond.”

(\models diamond, $\not\models$ big)

► “This is a fake diamond.”

($\models \neg$ diamond)

► “This is a fake blue diamond.”

(\models blue?, \models diamond?)

► “Mary knows that this is a diamond.”

(\models diamond)

► “Mary believes that this is a diamond.”

($\not\models$ diamond)

Ambiguity: The dark side of Meaning

- ▶ **Definition 3.8.** We call an utterance **ambiguous**, iff it has multiple meanings, which we call **readings**.
- ▶ **Example 3.9.** All of the following sentences are ambiguous:
 - ▶ “*John went to the bank.*” (river or financial?)

Ambiguity: The dark side of Meaning

- ▶ **Definition 3.10.** We call an utterance **ambiguous**, iff it has multiple meanings, which we call **readings**.
- ▶ **Example 3.11.** All of the following sentences are ambiguous:
 - ▶ “*John went to the bank.*” (river or financial?)
 - ▶ “*You should have seen the bull we got from the pope.*” (three readings!)

Ambiguity: The dark side of Meaning

- ▶ **Definition 3.12.** We call an utterance **ambiguous**, iff it has multiple meanings, which we call **readings**.
- ▶ **Example 3.13.** All of the following sentences are ambiguous:
 - ▶ “John went to the **bank**.” (river or financial?)
 - ▶ “You should have seen **the bull** we got from the pope.” (three readings!)
 - ▶ “I saw her duck.” (animal or action?)

Ambiguity: The dark side of Meaning

- ▶ **Definition 3.14.** We call an utterance **ambiguous**, iff it has multiple meanings, which we call **readings**.
- ▶ **Example 3.15.** All of the following sentences are ambiguous:
 - ▶ “John went to the **bank**.” (river or financial?)
 - ▶ “You should have seen **the bull** we got from the pope.” (three readings!)
 - ▶ “I saw her **duck**.” (animal or action?)
 - ▶ “John chased the gangster **in the red sports car**.” (three-way too!)

► **Example 3.16.** “*Every man loves a woman.*”

(Keira Knightley or his mother!)

► **Example 3.21.** “*Every man loves a woman.*”

(Keira Knightley or his mother!)

► **Example 3.22.** “*Every car has a radio.*”

(only one reading!)

- ▶ **Example 3.26.** “*Every man loves a woman.*” (Keira Knightley or his mother!)
- ▶ **Example 3.27.** “*Every car has a radio.*” (only one reading!)
- ▶ **Example 3.28.** “*Some student in every course sleeps in every class at least some of the time.*” (how many readings?)

- ▶ **Example 3.31.** “*Every man loves a woman.*” (Keira Knightley or his mother!)
- ▶ **Example 3.32.** “*Every car has a radio.*” (only one reading!)
- ▶ **Example 3.33.** “*Some student in every course sleeps in every class at least some of the time.*” (how many readings?)
- ▶ **Example 3.34.** “*The president of the US is having an affair with an intern.*” (2002 or 2000?)

- ▶ **Example 3.36.** “*Every man loves a woman.*” (Keira Knightley or his mother!)
- ▶ **Example 3.37.** “*Every car has a radio.*” (only one reading!)
- ▶ **Example 3.38.** “*Some student in every course sleeps in every class at least some of the time.*” (how many readings?)
- ▶ **Example 3.39.** “*The president of the US is having an affair with an intern.*” (2002 or 2000?)
- ▶ **Example 3.40.** “*Everyone is here.*” (who is everyone?)

More Context: Anaphora – Challenge for Pragmatic Analysis

► Example 3.41 (**Anaphoric** References).

- “*John is a bachelor. His wife is very nice.*”

(Uh, what?, who?)

More Context: Anaphora – Challenge for Pragmatic Analysis

► Example 3.45 (**Anaphoric** References).

- “*John is a bachelor. His wife is very nice.*”
- “*John likes his dog Spiff even though he bites him sometimes.*”

(Uh, what?, who?)
(who bites?)

More Context: Anaphora – Challenge for Pragmatic Analysis

► Example 3.49 (Anaphoric References).

- “*John is a bachelor. His wife is very nice.*”
- “*John likes his dog Spiff even though he bites him sometimes.*”
- “*John likes Spiff. Peter does too.*”

(Uh, what?, who?)

(who bites?)

(what to does Peter do?)

More Context: Anaphora – Challenge for Pragmatic Analysis

► Example 3.53 (Anaphoric References).

- “John is a bachelor. *His wife* is very nice.”
- “John likes his dog Spiff even though *he bites him* sometimes.”
- “John likes Spiff. Peter *does too*.”
- “John loves *his wife*. Peter does too.”

(Uh, what?, who?)

(who bites?)

(what to does Peter do?)

(whom does Peter love?)

More Context: Anaphora – Challenge for Pragmatic Analysis

► Example 3.57 (Anaphoric References).

- “*John is a bachelor. His wife is very nice.*”
- “*John likes his dog Spiff even though he bites him sometimes.*”
- “*John likes Spiff. Peter does too.*”
- “*John loves his wife. Peter does too.*”
- “*John loves golf, and Mary too.*”

(Uh, what?, who?)

(who bites?)

(what to does Peter do?)

(whom does Peter love?)

(who does what?)

More Context: Anaphora – Challenge for Pragmatic Analysis

► Example 3.61 (Anaphoric References).

- “John is a bachelor. *His wife* is very nice.” (Uh, what?, who?)
- “John likes his dog Spiff even though *he bites him* sometimes.” (who bites?)
- “John likes Spiff. *Peter does too*.” (what to does Peter do?)
- “John loves *his wife*. Peter does too.” (whom does Peter love?)
- “John loves golf, and *Mary too*.” (who does what?)

- **Definition 3.62.** A word or phrase is called **anaphoric** (or an **anaphor**), if its interpretation depends upon another phrase in context. In a narrower sense, an **anaphor** refers to an earlier phrase (its **antecedent**), while a **cataphor** to a later one (its **postcedent**).

Definition 3.63. The process of determining the antecedent or postcedent of an anaphoric phrase is called **anaphor resolution**.

Definition 3.64. An **anaphoric** connection between **anaphor** and its **antecedent** or **postcedent** is called **direct**, iff it can be understood purely **syntactically**. An **anaphoric** connection is called **indirect** or a **bridging reference** if additional **knowledge** is needed.

More Context: Anaphora – Challenge for Pragmatic Analysis

► Example 3.65 (Anaphoric References).

- “John is a bachelor. *His wife is very nice.*” (Uh, what?, who?)
- “John likes his dog Spiff even though *he bites him sometimes.*” (who bites?)
- “John likes Spiff. *Peter does too.*” (what to does Peter do?)
- “John loves *his wife.* Peter does too.” (whom does Peter love?)
- “John loves golf, and *Mary too.*” (who does what?)

- **Definition 3.66.** A word or phrase is called **anaphoric** (or an **anaphor**), if its interpretation depends upon another phrase in context. In a narrower sense, an **anaphor** refers to an earlier phrase (its **antecedent**), while a **cataphor** to a later one (its **postcedent**).

Definition 3.67. The process of determining the antecedent or postcedent of an anaphoric phrase is called **anaphor resolution**.

Definition 3.68. An anaphoric connection between anaphor and its antecedent or postcedent is called **direct**, iff it can be understood purely syntactically. An anaphoric connection is called **indirect** or a **bridging reference** if additional knowledge is needed.

- Anaphora are another example, where natural languages use the inferential capabilities of the hearer/reader to “shorten” utterances.
- Anaphora challenge pragmatic analysis, since they can only be resolved from the context using world knowledge.

► **Example 3.69.** Consider the following sentences involving **definite description**:

1. “*The king of America is rich.*”

(true or false?)

How do the interact with your context and **world knowledge**?

► **Example 3.70.** Consider the following sentences involving definite description:

1. “*The king of America is rich.*”
2. “*The king of America isn't rich.*”

(true or false?)

(false or true?)

How do the interact with your context and world knowledge?

► **Example 3.71.** Consider the following sentences involving definite description:

1. “*The king of America is rich.*” (true or false?)
2. “*The king of America isn't rich.*” (false or true?)
3. “*If America had a king, the king of America would be rich.*” (true or false!)

How do the interact with your context and world knowledge?

► **Example 3.72.** Consider the following sentences involving definite description:

1. “*The king of America is rich.*” (true or false?)
2. “*The king of America isn't rich.*” (false or true?)
3. “*If America had a king, the king of America would be rich.*” (true or false!)
4. “*The king of Buganda is rich.*” (Where is Buganda?)

How do the interact with your context and world knowledge?

Context is Personal and Keeps Changing

► **Example 3.73.** Consider the following sentences involving definite description:

1. “*The king of America is rich.*” (true or false?)
2. “*The king of America isn't rich.*” (false or true?)
3. “*If America had a king, the king of America would be rich.*” (true or false?)
4. “*The king of Buganda is rich.*” (Where is Buganda?)
5. “*... Joe Smith... The CEO of Westinghouse announced budget cuts.*” (CEO=J.S.!)

How do the interact with your context and world knowledge?

Context is Personal and Keeps Changing

► **Example 3.74.** Consider the following sentences involving **definite description**:

1. “*The king of America is rich.*” (true or false?)
2. “*The king of America isn't rich.*” (false or true?)
3. “*If America had a king, the king of America would be rich.*” (true or false?)
4. “*The king of Buganda is rich.*” (Where is Buganda?)
5. “*... Joe Smith... The CEO of Westinghouse announced budget cuts.*” (CEO=J.S.!)

How do they interact with your context and **world knowledge**?

► The interpretation or whether they make sense at all depends

► **Note:** Last two examples feed back into the context or even **world knowledge**:

- If 4. is uttered by an Africa expert, we add “*Buganda exists and is a monarchy*” to our world knowledge
- We add “*Joe Smith is the CEO of Westinghouse to the context/world knowledge*” (happens all the time in newspaper articles)

31.4 Language Models

Natural Languages vs. Formal Language

- ▶ **Recap:** A formal language is a set of strings.
- ▶ **Example 4.1.** Programming languages like Java or C++ are formal languages.
- ▶ **Remark 4.2.** Natural languages like English, German, or Spanish are not.
- ▶ **Example 4.3.** Let us look at concrete examples
 - ▶ “Not to be invited is sad!” (definitely English)
 - ▶ “To not be invited is sad!” (controversial)
- ▶ **Idea:** Let's be lenient, instead of a hard set, use a probability distribution.
- ▶ **Definition 4.4.** A (statistical) language model is a probability distribution over sequences of characters or words.
- ▶ **Idea:** Try to learn/derive language models from text corpora.
- ▶ **Definition 4.5.** A text corpus (or simply corpus; plural corpora) is a large and structured collection of natural language texts called documents.
- ▶ **Definition 4.6.** In corpus linguistics, corpora are used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistic rules within a specific natural language.

N-gram Character Models

- ▶ Written text is composed of **characters** letters, digits, punctuation, and spaces.
- ▶ **Idea:** Let's study **language models** for sequences of **characters**.
- ▶ As for **Markov processes**, we write $P(c_{1:N})$ for the **probability** of a **character** sequence $c_1 \dots c_n$ of length N .
- ▶ **Definition 4.7.** We call an **character** sequence of length n an **n gram** (**unigram**, **bigram**, **trigram** for $n = 1, 2, 3$).
- ▶ **Definition 4.8.** An **n gram model** is a **Markov process** of **order** $n - 1$.
- ▶ *Remark 4.9.* For a **trigram** model, $P(c_i \mid c_{1:i-1}) = P(c_i \mid c_{(i-2)}, c_{(i-1)})$. Factoring with the **chain rule** and then using the **Markov property**, we obtain

$$P(c_{1:N}) = \prod_{i=1}^N P(c_i \mid c_{1:i-1}) = \prod_{i=1}^N P(c_i \mid c_{(i-2)}, c_{(i-1)})$$

- ▶ **Thus**, a **trigram** model for a **language** with 100 **characters**, $P(c_i \mid c_{i-2:i-1})$ has 1.000.000 entries. It can be estimated from a **corpus** with 10^7 **characters**.

Applications of N -Gram Models of Character Sequences

- ▶ What can we do with N gram models?
- ▶ **Definition 4.10.** The problem of **language identification** is given a text, determine the **natural language** it is written in.
- ▶ *Remark 4.11.* Current technology can classify even short texts like “*Hello, world*”, or “*Wie geht es Dir*” correctly with more than 99% accuracy.
- ▶ **One approach:** Build a **trigram language model** $P(c_i | c_{i-2:i-1}, \ell)$ for each candidate language ℓ by counting **trigrams** in a ℓ -**corpus**.
Apply **Bayes' rule** and the **Markov property** to get the most likely language:

$$\begin{aligned}\ell^* &= \operatorname{argmax}_{\ell} (P(\ell \mid c_{1:N})) \\ &= \operatorname{argmax}_{\ell} (P(\ell) \cdot P(c_{1:N} \mid \ell)) \\ &= \operatorname{argmax}_{\ell} (P(\ell) \cdot (\prod_{i=1}^N P(c_i \mid c_{i-2:i-1}, \ell)))\end{aligned}$$

The **prior probability** $P(\ell)$ can be estimated, it is not a critical factor, since the **trigram language models** are extremely sensitive.

Other Applications of Character N -Gram Models

- ▶ Spelling correction is a direct application of a single-language **language model**: Estimate the probability of a **word** and all off-by-one variants.
- ▶ **Definition 4.12.** **Genre classification** means deciding whether a text is a news story, a legal **document**, a scientific article, etc.
- ▶ *Remark 4.13.* While many features help make this classification, counts of punctuation and other character n -gram features go a long way [**KesNunSch:adtg97**].
- ▶ **Definition 4.14.** **Named entity recognition (NER)** is the task of finding names of things in a **document** and deciding what class they belong to.
- ▶ **Example 4.15.** In “*Mr. Sopersteen was prescribed aciphex.*” **NER** should recognize that “*Mr. Sopersteen*” is the name of a person and “*aciphex*” is the name of a drug.
- ▶ *Remark 4.16.* Character-level **language models** are good for this task because they can associate the character sequence “*ex*” with a drug name and “*steen*” with a person name, and thereby identify **words** that they have never seen before.

► **Idea:** n gram models apply to word sequences as well.

► **Problems:** The method works identically, but

1. There are many more words than characters. (100 vs. 10^5 in English)
2. And what is a word anyways? (space/punctuation-delimited substrings?)
3. **Data sparsity:** we do not have enough data! For a language model for 10^5 words in English, we have 10^{15} trigrams.
4. Most training corpora do not have all words.

- ▶ **Definition 4.17.** Out of vocabulary (OOV) words are unknown words that appear in the test corpus but not training corpus.
- ▶ *Remark 4.18.* OOV words are usually content words such as names and locations which contain information crucial to the success of NLP tasks.
- ▶ **Idea:** Model OOV words by
 1. adding a new word token, e.g. <UNK> to the vocabulary,
 2. in the training corpus, replacing the respective first occurrence of a previously unknown word by <UNK>,
 3. counting n grams as usual, treating <UNK> as a regular word.This trick can be refined if we have a word classifier, then use a new token per class, e.g. <EMAIL> or <NUM>.

What can Word N -Gram Models do?

- ▶ **Example 4.19 (Test n -grams).** Build **unigram**, **bigram**, and **trigram** language models over the words [RusNor:AIMA03], randomly sample sequences from the models.
 1. Unigram: *"logical are as are confusion a may right tries agent goal the was ..."*
 2. Bigram: *"systems are very similar computational approach would be represented ..."*
 3. Trigram: *"planning and scheduling are integrated the success of naive bayes model ..."*
- ▶ **Clearly** there are differences, how can we measure them to evaluate the models?

What can Word N -Gram Models do?

- ▶ **Example 4.23 (Test n -grams).** Build **unigram**, **bigram**, and **trigram** language models over the words [RusNor:AIMA03], randomly sample sequences from the models.

1. Unigram: "*logical are as are confusion a may right tries agent goal the was ...*"
2. Bigram: "*systems are very similar computational approach would be represented ...*"
3. Trigram: "*planning and scheduling are integrated the success of naive bayes model ...*"

- ▶ **Clearly** there are differences, how can we measure them to evaluate the models?

- ▶ **Definition 4.24.** The **perplexity** of a sequence $c_{1:N}$ is defined as

$$\text{Perplexity}(c_{1:N}) := P(c_{1:N})^{-\left(\frac{1}{N}\right)}$$

- ▶ **Intuition:** The reciprocal of probability, normalized by sequence length.

- ▶ **Example 4.25.** For a language with n **characters** or **words** and a **language model** that predicts that all are equally likely, the **perplexity** of any sequence is n .

If some **characters** or **words** are more likely than others, and the model reflects that, then the **perplexity** of correct sequences will be less than n .

- ▶ **Example 4.26.** In ???, the **perplexity** was 891 for the **unigram** model, 142 for the **bigram** model and 91 for the **trigram** model.

31.5 Part of Speech Tagging

- ▶ **Recall:** *n*-grams can predict that a word sequence like “*a black cat*” is more likely than “*cat black a*”.
(as trigram 1. appears 0.000014% in a corpus and 2. never)
- ▶ **Native Speakers However:** Will tell you that “*a black cat*” matches a familiar pattern: article-adjective-noun, while “*cat black a*” does not!
- ▶ **Example 5.1.** Consider “*the fulvous kitten*” a native speaker reasons that it
 - ▶ follows the determiner-adjective-noun pattern
 - ▶ “*fulvous*” ($\hat{=}$ brownish yellow) ends in “*ous*” \leadsto adjectiveSo by generalization this is (probably) correct English.
- ▶ **Observation:** The order of syntactical categories of words plays a role in English!
- ▶ **Problem:** How can we compute them? (up next)

- ▶ **Definition 5.2.** **Part-of-speech tagging** (also **POS tagging**, **POST**, or **grammatical tagging**) is the process of **marking up** a **word** in **corpus** with tags (called **POS tags**) as corresponding to a particular **part of speech** (a category of **words** with similar syntactic properties) based on both its definition and its context.
- ▶ **Example 5.3.** A **sentence** tagged with **POS tags** from the **Penn treebank**: (see below)
From the start , it took a person with great qualities to succeed
IN DT NN , PRP VBD DT NN IN JJ NNS TO VB
 1. “*From*” is tagged as a **preposition** (IN)
 2. “*the*” as a **determiner** (DT)
 3. ...
- ▶ **Observation:** Even though **POS tagging** is uninteresting in its own right, it is useful as a first step in many **NLP** tasks.
- ▶ **Example 5.4.** In text-to-speech synthesis, a **POS tag** of “**noun**” for “*record*” helps determine the correct pronunciation (as opposed to the tag “**verb**”)

The Penn Treebank POS tags

► **Example 5.5.** The following 45 POS tags are used by the Penn treebank:

Tag	Word	Description	Tag	Word	Description
CC	<i>and</i>	Coordinating conjunction	PRP\$	<i>your</i>	Possessive pronoun
CD	<i>three</i>	Cardinal number	RB	<i>quickly</i>	Adverb
DT	<i>the</i>	Determiner	RBR	<i>quicker</i>	Adverb, comparative
EX	<i>there</i>	Existential there	RBS	<i>quickest</i>	Adverb, superlative
FW	<i>per se</i>	Foreign word	RP	<i>off</i>	Particle
IN	<i>of</i>	Preposition	SYM	<i>+</i>	Symbol
JJ	<i>purple</i>	Adjective	TO	<i>to</i>	to
JJR	<i>better</i>	Adjective, comparative	UH	<i>eureka</i>	Interjection
JJS	<i>best</i>	Adjective, superlative	VB	<i>talk</i>	Verb, base form
LS	<i>I</i>	List item marker	VBD	<i>talked</i>	Verb, past tense
MD	<i>should</i>	Modal	VBG	<i>talking</i>	Verb, gerund
NN	<i>kitten</i>	Noun, singular or mass	VCN	<i>talked</i>	Verb, past participle
NNS	<i>kittens</i>	Noun, plural	VBP	<i>talk</i>	Verb, non-3rd-sing
NNP	<i>Ali</i>	Proper noun, singular	VBZ	<i>talks</i>	Verb, 3rd-sing
NNPS	<i>Fords</i>	Proper noun, plural	WDT	<i>which</i>	Wh-determiner
PDT	<i>all</i>	Predeterminer	WP	<i>who</i>	Wh-pronoun
POS	<i>'s</i>	Possessive ending	WP\$	<i>whose</i>	Possessive wh-pronoun
PRP	<i>you</i>	Personal pronoun	WRB	<i>where</i>	Wh-adverb
\$	<i>\$</i>	Dollar sign	#	<i>#</i>	Pound sign
"	<i>'</i>	Left quote	"	<i>'</i>	Right quote
(<i>[</i>	Left parenthesis)	<i>]</i>	Right parenthesis
,	<i>,</i>	Comma	.	<i>!</i>	Sentence end
:	<i>;</i>	Mid-sentence punctuation			

Computing Part of Speech Tags

- ▶ **Idea:** Treat the POS tags in a sentence as state variables $C_{1:n}$ in a HMM: the words are the evidence variables $W_{1:n}$, use prediction for POS tagging.
- ▶ The HMM is a generative model that
 - ▶ starts in the tag predicted by the prior probability (usually IN) (problematic!)
 - ▶ and then, for each step makes two choices:
 - ▶ what word – e.g. “From” – should be emitted
 - ▶ what state – e.g. DT – should come next
- ▶ **This works, but** there are problems
 - ▶ the HMM does not consider context other than the current state (Markov property)
 - ▶ it does not have any idea what the sentence is trying to convey
- ▶ **Idea:** Use the Viterbi algorithm to find the most probable sequence of hidden states (POS tags)
- ▶ POS taggers based on the Viterbi algorithm can reach an F_1 score of up to 97%.

The Viterbi algorithm for POS tagging – Details

- ▶ We need a **transition model** $P(C_t \mid C_{t-1})$: the **probability** of one **POS tag** following another.
- ▶ **Example 5.6.** $P(C_t = VB \mid C_{t-1} = MD) = 0.8$ means that given a modal **verb** (e.g. “*would*”) the following **word** is a **verb** (e.g. “*think*”) with **probability** 0.8.
- ▶ **Question:** Where does the number 0.8 come from?
- ▶ **Answer:** From counts in the **corpus** – with appropriate **smoothing**!
There are 13124 instances of MD in the **Penn treebank** and 10471 are followed by a VB.
- ▶ For the **sensor model** $P(W_t = \textit{would} \mid C_t = MD) = 0.1$ means that if we choose a modal **verb**, we will choose “*would*” 10% of the time.
- ▶ These numbers also come from the **corpus** with appropriate **smoothing**.
- ▶ **Limitations:** **HMM** models only know about the **transition** and **sensor models**
In particular, we cannot take into account that e.g. **words** ending in “*ous*” are likely **adjectives**.
- ▶ We will see methods based on **neural networks** later.

31.6 Text Classification

- ▶ **Problem:** Often we want to (ideally) automatically see who can best deal with a given **document** (e.g. e-mails in customer service)
- ▶ **Definition 6.1.** Given a set of **categories** the task of deciding which one a given **document** belongs to is called **text classification** or **categorization**.
- ▶ **Example 6.2.** Language identification and genre classification are examples of **text classification**.
- ▶ **Example 6.3.** **Sentiment analysis** – classifying a product review as positive or negative.
- ▶ **Example 6.4.** **Spam detection** – classifying an email message as **spam** or **ham** (i.e. **non-spam**).

Spam Detection

- ▶ **Definition 6.5.** **Spam detection** – classifying an email message as **spam** or **ham** (i.e. **non-spam**)
- ▶ **General Idea:** Use **NLP/machine learning** techniques to learn the **categories**.
- ▶ **Example 6.6.** We have lots of **examples** of **spam/ham**, e.g.

Spam (from my **spam** folder)

Wholesale Fashion Watches -57% today. Designer watches for cheap ...

You can buy ViagraFr\$1.85 All Medications at unbeatable prices! ...

WE CAN TREAT ANYTHING YOU SUFFER FROM JUST TRUST US ...

Sta.rt earn*ing the salary yo,u d-eserve by o'btaining the prope,r crede'ntials!

Ham (in my inbox)

The practical significance of hypertree width in identifying more ...

Abstract: We will motivate the problem of social identity clustering: ...

Good to see you my friend. Hey Peter, It was good to hear from you. ...

PDS implies convexity of the resulting optimization problem (Kernel Ridge ...

- ▶ **Specifically:** What are good features to classify e-mails by?

- ▶ n -grams like “*for cheap*” and “*You can buy*” indicate **spam**
- ▶ character-level features: capitalization, punctuation

(but also occur in **ham**)
(e.g. in “*yo,u d-eserve*”)

- ▶ **Note:** We have two complementary ways of talking about **classification**:

(up next)

- ▶ using **language models**
- ▶ using **machine learning**

► **Idea:** Define two n -gram language models:

1. one for $P(\text{Message}|\text{spam})$ by training on the spam folder
2. one for $P(\text{Message}|\text{ham})$ by training on the inbox

Then we can classify a new message m with an application of Bayes' rule:

$$\operatorname{argmax}_{c \in \{\text{spam}, \text{ham}\}} (P(c \mid m)) = \operatorname{argmax}_{c \in \{\text{spam}, \text{ham}\}} (P(m \mid c)P(c))$$

where $P(c)$ is estimated just by counting the total number of spam and ham messages.

► This approach works well for **spam detection**, just as it did for **language identification**.

Classifier Success Measures: Precision, Recall, and F_1 score

- ▶ We need a way to measure success in **classification** tasks.
- ▶ **Definition 6.7.** Let $f_C: S \rightarrow \mathbb{B}$ be a binary **classifier** for a class $C \subseteq S$, then we call $a \in S$ with $f_C(a) = \text{T}$ a **false positive**, iff $a \notin C$ and $f_C(a) = \text{F}$ a **false negative**, iff $a \in C$. **False positives** and **negatives** are errors of f_C . **True positives** and **negatives** occur when f_C correctly indicates actual membership in S .
- ▶ **Definition 6.8.** The **precision** of f_C is defined as $\frac{\#(TP)}{\#(TP) + \#(FP)}$ and the **recall** is $\frac{\#(TP)}{\#(TP) + \#(FN)}$, where TP is the set of **true positives** and FN/FP the sets of **false negatives** and **false positives** of f_C .
- ▶ **Intuitively** these measure the rates of:
 - ▶ **true positives** in class C . (precision high, iff few false positives)
 - ▶ **true positives** in $f_C^{-1}(\text{T})$. (recall high, iff few true positives forgotten, i.e. few false negatives)
- ▶ **Definition 6.9.** The **F_1 score** combines **precision** and **recall** into a single number: (**harmonic mean**)
$$2 \frac{\text{precision} \cdot \text{recall}}{(\text{precision} + \text{recall})}$$
- ▶ **Observation:** Classifiers try to reach **precision** and **recall** \leadsto **F_1 score** of 1.
 - ▶ if that is impossible, compromise on one \leadsto **F_β score**. (**application-dependent**)
 - ▶ The **F_β score** generalizes the **F_1 score** by weighing the **precision** β times as important as **recall**.

31.7 Information Retrieval

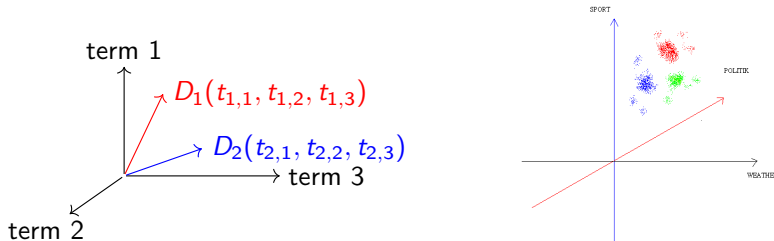


- ▶ **Definition 7.1.** An **information need** is an individual or group's desire to locate and obtain **information** to satisfy a conscious or unconscious need.
- ▶ **Definition 7.2.** An **information object** is **medium** that is mainly used for its **information content**.
- ▶ **Definition 7.3.** **Information retrieval (IR)** deals with the **representation**, organization, storage, and maintenance of **information objects** that provide **users** with easy access to the **relevant information** and satisfy their various **information needs**.
Observation (Hjørland 1997): **Information need** is closely related to **relevance**: If something is **relevant** for a person in relation to a given task, we might say that the person **needs** the **information** for that task.
- ▶ **Definition 7.4.** **Relevance** denotes how well an **information object** meets the **information need** of the **user**. **Relevance** may include concerns such as timeliness, authority or novelty of the **object**.
- ▶ **Observation:** We normally come in contact with **IR** in the form of **web search**.
- ▶ **Definition 7.5.** **Web search** is a fully automatic process that responds to a **user query** by returning a sorted **document** list **relevant** to the **user** requirements expressed in the **query**.
- ▶ **Example 7.6.** Google and Bing are **web search engines**, their **query** is a **bag** of **words** and **documents** are **web pages**, **PDFs**, **images**, **videos**, **shopping portals**.

- ▶ **Idea:** For web search, we usually represent documents and queries as bags of words over a fixed vocabulary V . Given a query Q , we return all documents that are “similar”.
- ▶ **Definition 7.7.** Given a vocabulary (a list) V of words, a word $w \in V$, and a document d , then we define the raw term frequency (often just called the term frequency) of w in d as the number of occurrences of w in d .
- ▶ **Definition 7.8.** A multiset of words in $V = \{t_1, \dots, t_n\}$ is called a bag of words (BOW), and can be represented as a word frequency vectors in $\mathbb{N}^{|V|}$: the vector of raw word frequencies.
- ▶ **Example 7.9.** If we have two documents: $d_1 = \text{“Have a good day!”}$ and $d_2 = \text{“Have a great day!”}$, then we can use $V = \text{“Have”, “a”, “good”, “great”, “day”}$ and can represent “good” as $\langle 0, 0, 1, 0, 0 \rangle$, “great” as $\langle 0, 0, 0, 1, 0 \rangle$, and d_1 as $\langle 1, 1, 1, 0, 1 \rangle$. Words outside the vocabulary are ignored in the BOW approach. So the document $d_3 = \text{“What a day, a good day”}$ is represented as $\langle 0, 2, 1, 0, 2 \rangle$.

Vector Space Models for IR

- **Idea:** Query and document are similar, iff the angle between their word frequency vectors is small.



- **Lemma 7.10 (Euclidean Dot Product Formula).** $A \cdot B = \|A\|_2 \|B\|_2 \cos \theta$, where θ is the angle between A and B .
- **Definition 7.11.** The **cosine similarity** of A and B is $\cos \theta = \frac{A \cdot B}{\|A\|_2 \|B\|_2}$.

TF-IDF: Term Frequency/Inverse Document Frequency

- ▶ **Problem:** Word frequency vectors treat all the words equally.
- ▶ **Example 7.12.** In an query “*the brown cow*”, the “*the*” is less important than “*brown cow*”. (because “*the*” is less specific)
- ▶ **Idea:** Introduce a weighting factor for the word frequency vector that de-emphasizes the dimension of the more (globally) frequent words.
- ▶ We need to normalize the word frequency vectors first:
- ▶ **Definition 7.13.** Given a document d and a vocabulary word $t \in V$, the **normalized term frequency** (confusingly often called just **term frequency**) $\text{tf}(t, d)$ is the raw term frequency divided by $|d|$.
- ▶ **Definition 7.14.** Given a document collection $D = \{d_1, \dots, d_N\}$ and a word t the **inverse document frequency** is given by $\text{idf}(t, D) := \log_{10}\left(\frac{N}{|\{d \in D \mid t \in d\}|}\right)$.
- ▶ **Definition 7.15.** We define $\text{tfidf}(t, d, D) := \text{tf}(t, d) \cdot \text{idf}(t, D)$.
- ▶ **Idea:** Use the **tfidf-vector** with cosine similarity for information retrieval instead.
- ▶ **Definition 7.16.** Let D be a document collection with vocabulary $V = \{t_1, \dots, t_{|V|}\}$, then the **tfidf-vector** $\overline{\text{tfidf}}(d, D) \in \mathbb{N}^{|V|}$ is defined by $\overline{\text{tfidf}}(d, D)_i := \text{tfidf}(t_i, d, D)$.

TF-IDF Example

- ▶ Let $D := \{d_1, d_2\}$ be a document corpus over the vocabulary

$$V = \{\text{"this"}, \text{"is"}, \text{"a"}, \text{"sample"}, \text{"another"}, \text{"example"}\}$$

with word frequency vectors $\langle 1, 1, 1, 2, 0, 0 \rangle$ and $\langle 1, 1, 0, 0, 2, 3 \rangle$.

- ▶ Then we compute for the word "this"

- ▶ $\text{tf}(\text{"this"}, d_1) = \frac{1}{5} = 0.2$ and $\text{tf}(\text{"this"}, d_2) = \frac{1}{7} \approx 0.14$,
- ▶ idf is constant over D , we have $\text{idf}(\text{"this"}, D) = \log_{10}(\frac{2}{1}) = 0$,
- ▶ thus $\text{tfidf}(\text{"this"}, d_1, D) = 0 = \text{tfidf}(\text{"this"}, d_2, D)$. ("this" occurs in both)

- ▶ The word "example" is more interesting, since it occurs only in d_2 (thrice)

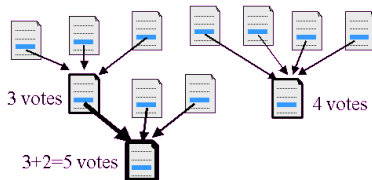
- ▶ $\text{tf}(\text{"example"}, d_1) = \frac{0}{5} = 0$ and $\text{tf}(\text{"example"}, d_2) = \frac{3}{7} \approx 0.429$.
- ▶ $\text{idf}(\text{"example"}, D) = \log_{10}(\frac{2}{1}) \approx 0.301$,
- ▶ thus $\text{tfidf}(\text{"example"}, d_1, D) = 0 \cdot 0.301 = 0$ and $\text{tfidf}(\text{"example"}, d_2, D) \approx 0.429 \cdot 0.301 = 0.129$.

Ranking Search Hits: e.g. Google's Page Rank

► **Problem:** There are many hits, need to sort them

(e.g. by importance)

► **Idea:** A web site is important, ... if many other hyperlink to it.



► **Refinement:** ..., if many important web pages hyperlink to it.

► **Definition 7.17.** Let A be a web page that is hyperlinked from web pages S_1, \dots, S_n , then the page rank PR of A is defined as

$$PR(A) = 1 - d + d \left(\frac{PR(S_1)}{C(S_1)} + \dots + \frac{PR(S_n)}{C(S_n)} \right)$$

where $C(W)$ is the number of links in a page W and $d = 0.85$.

► **Remark 7.18.** $PR(A)$ is the probability of reaching A by random browsing.

31.8 Information Extraction

- ▶ **Definition 8.1.** **Information extraction** is the process of acquiring **information** by skimming a text and looking for occurrences of a particular class of **object** and for relationships among **objects**.
- ▶ **Example 8.2.** Extracting instances of addresses from **web pages**, with **attributes** for street, city, state, and zip code;
- ▶ **Example 8.3.** Extracting instances of storms from weather reports, with **attributes** for temperature, wind speed, and precipitation.
- ▶ **Observation:** In a limited domain, this can be done with high accuracy.

Attribute-Based Information Extraction

- ▶ **Definition 8.4.** In **attribute-based information extraction** we assume that the text refers to a single **object** and the task is to extract a **factored** representation.
- ▶ **Example 8.5 (Computer Prices).** Extracting from the text “*IBM ThinkBook 970. Our price: \$399.00*” the **attribute-based representation** {Manufacturer=IBM, Model=ThinkBook970, Price=\$399.00}.
- ▶ **Idea:** Try a **template-based** approach for each **attribute**.
- ▶ **Definition 8.6.** A **template** is a **finite automaton** that recognizes the **information** to be extracted. The **template** often consists of three sub-automata per **attribute**: the **prefix pattern** followed by the **target pattern** (it matches the **attribute value**) and the **postfix pattern**.
- ▶ **Example 8.7 (Extracting Prices with Regular Expressions).** When we want to extract computer price **information**, we could use **regular expressions** for the **automata**, concretely, the
 - ▶ **prefix pattern:** .*price[:]?
 - ▶ **target pattern:** [\$][0–9]+([.][0–9][0–9])?
 - ▶ **postfix pattern:** + shipping|
- ▶ **Alternative:** take all the target matches and choose among them.
- ▶ **Example 8.8.** For “*List price \$99.00, special sale price \$78.00, shipping \$3.00.*” take the lowest price that is within 50% of the highest price. \leadsto “*\$78.00*”

- ▶ **Question:** Can we also do **structured** representations?
- ▶ **Answer:** That is the next step up from **attribute-based information extraction**.
- ▶ **Definition 8.9.** The task of a **relational extraction** system is to extract multiple **objects** and the relationships among them from a text.
- ▶ **Example 8.10.** When these systems see the text “**\$249.99**,” they need to determine not just that it is a price, but also which object has that price.
- ▶ **Example 8.11.** **FASTUS** is a typical **relational extraction** system, which handles news stories about corporate mergers and acquisitions. It can read the story
Bridgestone Sports Co. said Friday it has set up a joint venture in Taiwan with a local concern and a Japanese trading house to produce golf clubs to be shipped to Japan.
and extract the relations:

$$e \in \text{JointVentures} \wedge \text{Product}(e, \text{"golfclubs"}) \wedge \text{Date}(e, \text{"Friday"})$$

$$\text{Member}(e, \text{"BridgestoneSportsCo"}) \wedge \text{Member}(e, \text{"alocalconcern"})$$

$$\text{Member}(e, \text{"aJapanesetradinghouse"})$$

Advertisement: Logic-Based Natural Language Semantics

- ▶ **Advanced Course:** “Logic-Based Natural Language Semantics” (next semester)
 - ▶ Wed. 10:15-11:50 and Thu 12:15-13:50 (expected: ≤ 10 Students)
- ▶ **Contents:** (Alternating Lectures and hands-on Lab Sessions)
 - ▶ Foundations of Natural Language Semantics (NLS)
 - ▶ Montague’s Method of Fragments (Grammar, Semantics Constr., Logic)
 - ▶ Implementing Fragments in GLF (Grammatical Framework and MMT)
 - ▶ Inference Systems for Natural Language Pragmatics (tableau machine)
 - ▶ Advanced logical systems for NLS (modal, higher-order, dynamic Logics)
- ▶ **Grading:** Attendance & Wakefulness, Project/Homework, Oral Exam.
- ▶ **Course Intent:** Groom students for bachelor/master theses and as KWARC research assistants.

Chapter 32

Deep Learning for NLP

- ▶ **Observation:** Symbolic and statistical systems have demonstrated success on many NLP tasks, but their performance is limited by the endless complexity of natural language.
- ▶ **Idea:** Given the vast amount of text in machine-readable form, can data-driven machine-learning base approaches do better?
- ▶ In this chapter, we explore this idea, using – and extending – the methods from ???.
- ▶ Overview:
 1. Word embeddings
 2. Recurrent neural networks for NLP
 3. Sequence-to-sequence models
 4. Transformer Architecture
 5. Pretraining and transfer learning.

32.1 Word Embeddings

Word Embeddings

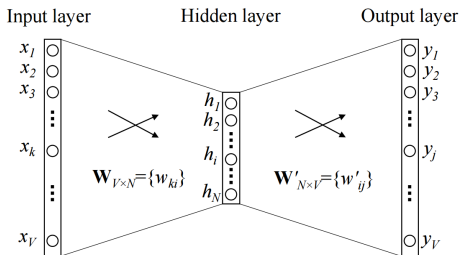
- ▶ **Problem:** For ML methods in NLP, we need numerical data. (not words)
- ▶ **Idea:** Embed words or word sequences into real vector spaces.
- ▶ **Definition 1.1.** A word embedding is a mapping from words in context into a real vector space \mathbb{R}^n used for natural language processing.
- ▶ **Definition 1.2.** A vector is called one hot, iff all components are 0 except for one 1. We call a word embedding one hot, iff all of its vectors are.
One hot word embeddings are rarely used for actual tasks, but often used as a *starting point* for better word embeddings.
- ▶ **Example 1.3 (Vector Space Methods in Information Retrieval).**
Word frequency vectors are induced by adding up one hot word embeddings.
- ▶ **Example 1.4.** Given a corpus D – the context – the tf idf word embedding is given by $\text{tfidf}(t, d, D) := \text{tf}(t, d) \cdot \log_{10}\left(\frac{|D|}{|\{d \in D \mid t \in d\}|}\right)$, where $\text{tf}(t, d)$ is the term frequency of word t in document d .
- ▶ **Intuition behind these two:** Words that occur in similar documents are similar.

Idea: Use *feature extraction* to map **words** to vectors in \mathbb{R}^N :

Train a **neural network** on a “dummy task”, throw away the output layer, use the previous layer’s output (of size N) as the **word embedding**

First Attempt: Dimensionality Reduction: Train to predict the original **one hot** vector:

- ▶ For a vocabulary size V , train a network with a single hidden layer; i.e. three layers of sizes (V, N, V) . The first two layers will compute our embeddings.
- ▶ Feed the **one hot** encoded input **word** into the network, and train it on the **one hot** vector itself, using a **softmax** activation function at the output layer. (**softmax normalizes a vector into a probability distribution**)

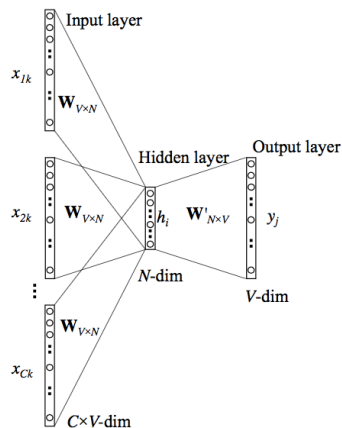


Word2Vec: The Continuous Bag Of Words (CBOW) Algorithm

Distributional Semantics: “a word is characterized by the company it keeps”.

Better Idea: Predict a **word** from its context:

- ▶ For a context window size n , take all sequences of $2n + 1$ **words** in our **corpus** (e.g. *the brown cow jumps over the moon* for $n = 3$) as training data. We call the **word** at the center (*jumps*) the *target word*, and the remaining **words** the *context words*.
- ▶ For every such sentence, pass all context **words** (one-hot encoded) through the first layer of the network, yielding $2n$ vectors.
- ▶ Pass their average into the output layer (*average pooling layer*) with a **softmax** activation function, and train the network to predict the target **word**. (sum pooling also works)



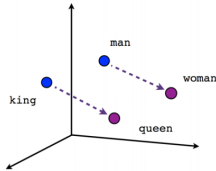
Properties

Vector embeddings like CBOW have interesting properties:

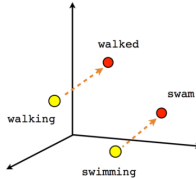
- ▶ *Similarity*: Using e.g. *cosine similarity* ($A \cdot B \cdot \cos(\theta)$) to compare vectors, we can find **words** with similar meanings.
- ▶ Semantic and syntactic relationships emerge as arithmetic relations:

$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$

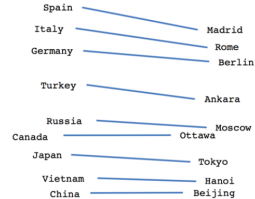
$$\text{germany} - \text{country} + \text{capitol} \approx \text{berlin}$$



Male-Female



Verb tense



Country-Capital

- ▶ **Observation:** Word embeddings are crucial as first steps in any NN-based NLP methods.
- ▶ In practice it is often sufficient to use generic, pretrained word embeddings
- ▶ **Definition 1.5.** Common pretrained – i.e. trained for generic NLP applications word embeddings include
 - ▶ Word2vec: the original system that established the concept (see above)
 - ▶ GloVe (Global Vectors)
 - ▶ FASTTEXT (embeddings for 157 languages)
- ▶ But we can also train our own word embedding (together with main task) (up next)

Learning POS tags and Word embeddings simultaneously

Specific **word embeddings** are trained on a carefully selected **corpus** and tend to emphasize the characteristics of the task.

Example 1.6. **POS tagging** – even though simple – is a good but non-trivial example.

Recall that many **words** can have multiple **POS tags**, e.g. “*cut*” can be

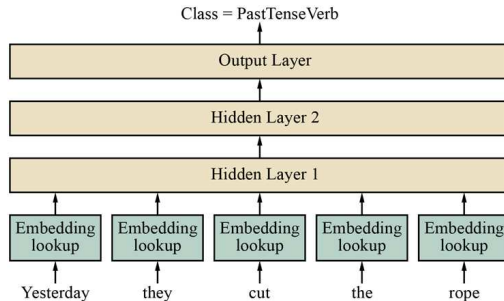
- ▶ a **present tense verb** (transitive or intransitive)
- ▶ a **past tense verb**
- ▶ a infinitive **verb**
- ▶ a **past** participle
- ▶ an **adjective**
- ▶ a **noun**.

If a nearby temporal **adverb** refers to the **past** \leadsto this occurrence may be a **past tense verb**.

Note: CBOW treats all context **words** identically regardless of *order*, but in **POS tagging** the exact *positions* of the **words** matter.

POS/Embedding Network

Idea: Start with a random (or pretrained) embedding of the **words** in the **corpus** and just concatenate them over some context window size



- ▶ Layer 1 has (in this case) $5 \cdot N$ inputs, Output layer is **one hot** over **POS** classes.
- ▶ The embedding layers treat all **words** the same, but the first hidden layer will treat them differently depending on the position.
- ▶ The embeddings will be finetuned for the **POS** task during training.

Note: Better *positional encoding* techniques exist (e.g. sinusoidal), but for fixed small context window sizes, this works well.

32.2 Recurrent Neural Networks

- ▶ word embeddings give a good representation of words in isolation.
- ▶ But natural language of word sequences \Leftarrow surrounding words provide context!
- ▶ For simple tasks like POS tagging, a fixed-size window of e.g. 5 words is sufficient.
- ▶ **Observation:** For advanced tasks like question answering we need more context!
- ▶ **Example 2.1.** In the sentence “*Eduardo told me that Miguel was very sick so I took him to the hospital*”, the pronouns “*him*” refers to “*Miguel*” and not “*Eduardo*”. (14 words of context)
- ▶ **Observation:** Language models with n -grams or n -word feed-forward networks have problems: Either the context is too small or the model has too many parameters! (or both)
- ▶ **Observation:** Feed-forward networks N also have the problem of asymmetry: whatever N learns about a word w at position n , it has to relearn about w at position $m \neq n$.
- ▶ **Idea:** What about recurrent neural networks – nets with cycles? (up next)

RNNs for Time Series

- **Idea:** RNNs – neural networks with cycles – have memory
→ use that for more context in neural NLP.

- **Example 2.2 (A simple RNN).**

It has an input layer x , a hidden layer z with recurrent connections and delay Δ , and an output layer y as shown on the right.

Defining Equations for time step t :

$$z_t = g_z(W_{z,z}z_{t-1} + W_{x,z}x_t)$$

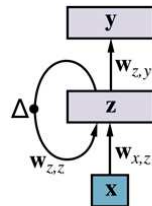
$$y_t = g_y(W_{z,y}z_t)$$

where g_z and g_y are the activation functions for the hidden and output layers.

- **Intuition:** RNNs are a bit like HMMs and dynamic Bayesian Networks:

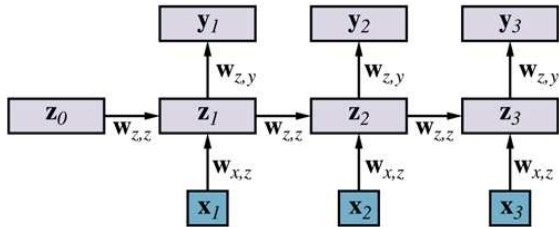
They make a Markov assumption: the hidden state z suffices to capture the input from all previous inputs.

- **Side Benefit:** RNNs solve the asymmetry problem \Leftarrow , the $W_{z,z}$ are the same at every step.



Training RNNs for NLP

- **Idea:** For training, **unroll** a **RNN** into a **feed-forward network** \leadsto **back-propagation**.
- **Example 2.3.** The **RNN** from ?? **unrolled** three times.

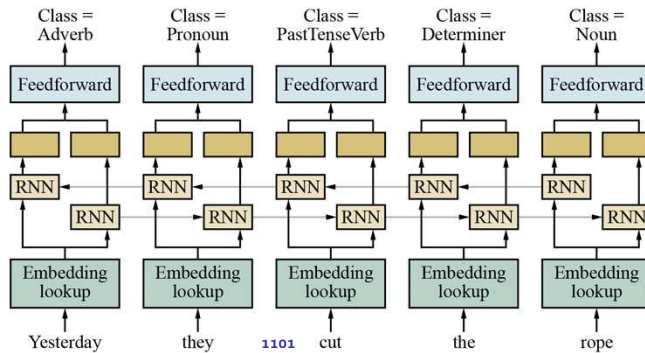


Problem: The **weight matrices** $W_{x,z}$, $W_{z,z}$, and $W_{z,y}$ are shared over all time slides.

- **Definition 2.4.** The **back-propagation through time algorithm** carefully maintains the identity of $W_{z,z}$ over all steps

Bidirectional RNN for more Context

- **Observation:** RNNs only take left context – i.e. words before – into account, but we may also need right context – the words after.
- **Example 2.5.** For “*Eduardo told me that Miguel was very sick so I took him to the hospital*” the pronoun “*him*” resolves to “*Miguel*” with high probability.
If the sentence ended with “*to see Miguel*”, then it should be “*Eduardo*”.
- **Definition 2.6.** A **bidirectional RNN** concatenates a separate right-to-left model onto a left-to-right model
- **Example 2.7.** Bidirectional RNNs can be used for POS tagging, extending the network from ???



- ▶ **Problem:** When training a vanilla RNN using back-propagation through time, the long-term gradients which are back-propagated can “vanish” – tend to zero – or “explode” – tend to infinity.
- ▶ **Definition 2.8.** LSTMs provide a short-term memory for RNN that can last thousands of time steps, thus the name “long short-term memory”. A LSTM can learn when to remember and when to forget pertinent information,
- ▶ **Example 2.9.** In NLP LSTMs can learn grammatical dependencies.
An LSTM might process the sentence “*Dave, as a result of his controversial claims, is now a pariah*” by
 - ▶ remembering the (statistically likely) grammatical gender and number of the subject “*Dave*”,
 - ▶ note that this information is pertinent for the pronoun “*his*” and
 - ▶ note that this information is no longer important after the verb “*is*”.

Introduce a *memory vector* c in addition to the recurrent (short-term memory) vector z

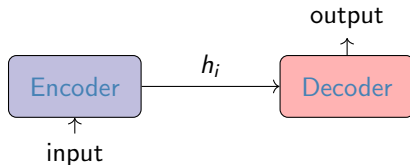
- ▶ c is essentially copied from the previous time step, but can be modified by the *forget gate* f , the *input gate* i , and the *output gate* o .
- ▶ the *forget gate* f decides which components of c to retain or discard
- ▶ the *input gate* i decides which components of the *current* input to *add* to c (additive, not multiplicative \leadsto no vanishing gradients)
- ▶ the *output gate* o decides which components of c to *output* as z

32.3 Sequence-to-Sequence Models

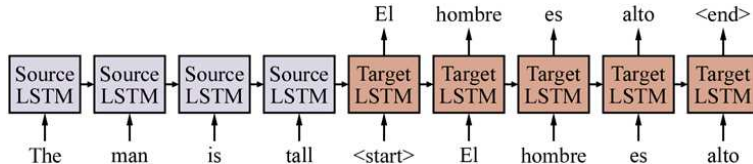
- ▶ **Question:** Machine translation (MT) is an important task in NLP, can we do it with neural networks?
 - ▶ **Observation:** If there were a one-to-one correspondence between source words and target words MT would be a simple tagging task. But
 - ▶ the three Spanish words “*caballo de mar*” translate to the English “*seahorse*” and
 - ▶ the two Spanish words “*perro grande*” translate to English as “*big dog*”.
 - ▶ in English, the subject is usually first and in Fijian last.
 - ▶ **Idea:** For MT, generate one word at a time, but keep track of the context, so that
 - ▶ we can remember parts of the source we have not translated yet
 - ▶ we remember what we already translated so we do not repeat ourselves.
- We may have to process the whole source sentence before generating the target!
- ▶ **Remark:** This smells like we need LSTMs.

Sequence-To-Sequence Models

- **Idea:** Use two coupled RNNs, one for the **source**, and one for the **target**. The input for the **target** is the output of the last hidden layer of the **source** RNN.
- **Definition 3.1.** A **sequence-to-sequence** (seq2seq) model is a **neural** model for translating an input sequence x into an output sequence y by an **encoder** followed by a **decoder** generates y .



- **Example 3.2.** A simple **seq2seq** model (without embedding and output layers)



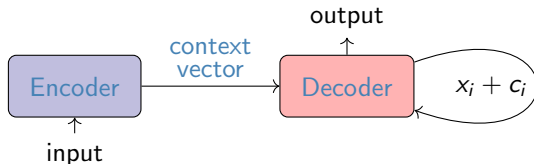
Each block represents one **LSTM** time step; inputs are fed successively followed by the token **<start>** to start the **decoder**.

- ▶ **Remark:** Seq2seq models were a major breakthrough in NLP and MT. But they have three major shortcomings:
 - ▶ **nearby context bias:** RNNs remember with their hidden state, which has more information about a word in – say – step 56 than in step 5. BUT long-distance context can also be important.
 - ▶ **fixed context size:** the entire information about the source sentence must be compressed into the fixed-dimensional – typically 1024 – vector. Larger vectors \leadsto slow training and overfitting.
- ▶ **Idea:** Concatenate all source RNN hidden vectors to use all of them to mitigate the nearby context bias.
- ▶ **Problem:** Huge increase of weights \leadsto slow training and overfitting.

Attention

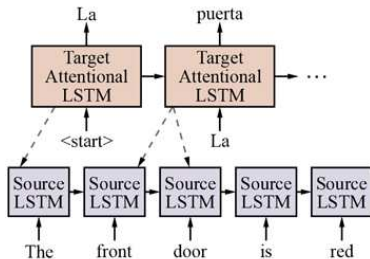
- ▶ **Bad Idea:** Concatenate all **source RNN hidden** vectors to use all of them to mitigate the nearby context bias.
- ▶ **Better Idea:** The **decoder** generates the **target** sequence one **word** at a time. \leadsto Only a small part of the **source** is actually relevant.
the **decoder** must focus on different parts of the **source** for every **word**.
- ▶ **Idea:** We need a **neural** component that does context-free summarization.
- ▶ **Definition 3.3.** An **attentional seq2seq** model is a **seq2seq** that passes along a **context vector** c_i in the **decoder**. If $h_i = \text{RNN}(h_{i-1}, x_i)$ is the standard **decoder**, then the **decoder** with **attention** is given by $h_i = \text{RNN}(h_{i-1}, x_i + c_i)$, where $x_i + c_i$ is the concatenation of the input x_i and **context vectors** c_i with

$$\begin{aligned} r_{ij} &= h_{i-1} \cdot s_j && \text{raw attention score} \\ a_{ij} &= e^{r_{ij}} / (\sum_k e^{r_{ik}}) && \text{attention probability matrix} \\ c_i &= \sum_j a_{ij} \cdot s_j && \text{context vector} \end{aligned}$$

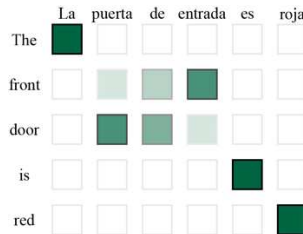


Attention: English to Spanish Translation

- **Example 3.4.** An **attentional seq2seq** model for English-to-Spish translation



dashed lines represent **attention**



attention probability matrix
darker colors \leadsto higher probabilities

- **Remarks:** The **attention**

- component learns no weights and supports variable-length sequences.
- is entirely latent – the developer does not influence it.

Attention: Greedy Decoding

- ▶ During training, a **seq2seq** model tries to maximize the probability of each **word** in the training sequence, conditioned on the **source** and the previous **target words**.
- ▶ **Definition 3.5.** The procedure that generates the **target** one **word** at a time and feeds it back at the next time step is called **decoding**.
- ▶ **Definition 3.6.** Always selecting the highest probability **word** is called **greedy decoding**.
- ▶ **Problem:** This may not always maximize the probability of the whole sequence
- ▶ **Example 3.7.** Let's use a **greedy decoder** on "*The front door is red*".
 - ▶ The correct translation is "*La puerta de entrada es roja*".
 - ▶ Suppose we have generated the first **word** "*La*" for "*The*".
 - ▶ A **greedy decoder** might propose "*entrada*" for "*front*".
- ▶ **Greedy decoding** is fast, but has no mechanism for correcting mistakes.
- ▶ **Solution:** Use an optimizing **search algorithm** (e.g. local beam search)

- ▶ **Recall:** Greedy decoding is not optimal!
- ▶ **Idea:** Search for an optimal decoding (or at least a good one) using one of the search algorithms from ???.
- ▶ **Local beam search** is a common choice in machine translation. Concretely:
 - ▶ keep the top k hypotheses at each stage,
 - ▶ extending each by one word using the top k choices of words,
 - ▶ then chooses the best k of the resulting k^2 new hypotheses.

When all hypotheses in the beam generate the special $\langle \text{end} \rangle$ token, the algorithm outputs the highest scoring hypothesis.

- ▶ **Observation:** The better the seq2seq models get, the smaller we can keep beam size
Today beams of $b = 4$ are sufficient after $b = 100$ a decade ago.

Decoding with Beam Search

- **Example 3.8.** A **local beam search** with beam size $b = 2$



- Word scores are log-probabilities generated by the **decoder softmax**
- hypothesis score is the sum of the **word** scores.

At time step 3, the highest scoring hypothesis “*La entrada*” can only generate low-probability continuations, so it “falls off the beam”. (as intended)

32.4 The Transformer Architecture

Self-Attention

- ▶ **Idea:** “Attention is all you need!” (see [VasShaPar:aiayn17])
- ▶ So far, attention was used from the encoder to the decoder.
- ▶ **Self-attention** extends this so that each hidden states sequence also attends to itself. (*coder to *coder)
- ▶ **Idea:** Just use the dot product of the input vectors
- ▶ **Problem:** Always high, so each hidden state will be biased towards attending to itself.
- ▶ **Self-attention** solves this by first projecting the input into three different representations using three different weight matrices:
 - ▶ the **query vector** $q_i = W_q x_i \hat{=}$ standard attention
 - ▶ **key vector** $k_i = W_k x_i \hat{=}$ the source in seq2seq
 - ▶ **value vector** $v_i = W_v x_i$ is the context being generated

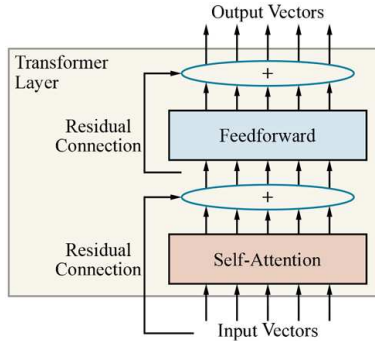
$$\begin{aligned} r_{ij} &= (q_i \cdot k_j) / \sqrt{d} \\ a_{ij} &= e^{r_{ij}} / (\sum_k e^{r_{ij}}) \\ c_i &= \sum_j a_{ij} \cdot v_j \end{aligned}$$

where d is the dimension of k and q .

The Transformer Architecture

- **Definition 4.1.** The **transformer architecture** uses **neural** blocks called **transformers**, which are built up from multiple **transformer layers**.
- **Remark:** The context modeled in **self-attention** is agnostic to word order \leadsto **transformers** use **positional embeddings** to cope with that.
- **Example 4.2.**

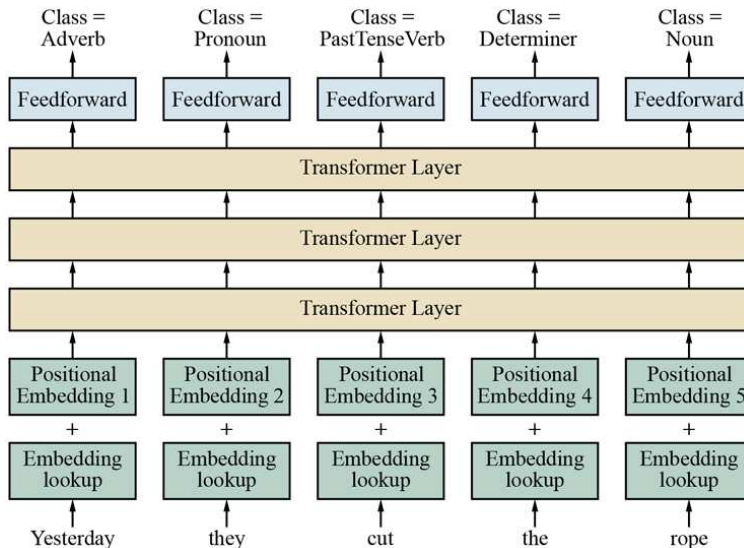
A single-layer **transformer** consists of **self-attention**, a **feed-forward network**, and **residual connections** to cope with the vanishing gradient problem.



- In practice **transformers** consist of 6-7 **transformer layers**.

A Transformer for POS tagging

► **Example 4.3.** A transformers for POS tagging:



32.5 Large Language Models

Pretraining and Transfer Learning

- ▶ Getting enough data to build a robust model can be a challenge.
- ▶ In **NLP** we often work with unlabeled data
 - ▶ syntactic/semantic labeling is much more difficult \leadsto costly than image labeling.
 - ▶ the Internet has lots of texts (adds $\sim 10^{11}$ words/day)
- ▶ **Idea:** Why not let other's do this work and re-use their training efforts.
- ▶ **Definition 5.1.** In **pretraining** we use
 - ▶ a large amount of shared general-domain language data to train an initial version of an **NLP** model.
 - ▶ a smaller amount of domain-specific data (perhaps labeled) to **finetune** it to the vocabulary, idioms, syntactic structures, and other linguistic phenomena that are specific to the new domain.
- ▶ **Pretraining** is a form of **transfer learning**:
- ▶ **Definition 5.2.** In **Transfer learning (TL)**, knowledge learned from a task is re-used in order to boost performance on a related task.
- ▶ **Idea:** Take a **pretrained neural network**, replace the **last layer(s)**, and then train those on your own corpus.
- ▶ **Observation:** Simple but surprisingly **efficient**!

Large Language Models

Definition 5.3. A **Large Language Model (LLM)** is a generic **pretrained neural network**, providing **embeddings** for **sentences** or entire **documents** for **NLP** tasks. In practice, they (usually) combine the following components:

- ▶ **Tokenization**: Splitting text into **tokens** (characters, words, punctuation,...)
- ▶ **embeddings** for these **tokens**, (e.g., **Word2vec** – or we let the **transformer** learn them)
- ▶ **positional embeddings** of **tokens** (encodes where in a sentence a **token** is)
- ▶ a **transformer architecture**, trained on
- ▶ a **masked token prediction** task.

LLMs can be used for a variety of tasks.

- ▶ *classification* (e.g., sentiment analysis, POS-tagging),
- ▶ *translation* (between languages, styles, etc.),
- ▶ *generation* (e.g., text completion, summarization, chatbots),
- ▶ ...

Tokenization - Byte Pair Encodings

So far: we have encoded text either as sequences of characters (non-semantic) or as sequences of **words** (semantic, but virtually unlimited vocabulary, OOV-problems).

Idea: Find a middle ground: Learn an optimal vocabulary of **tokens** from data and split text into a sequence of **tokens**.

Definition 5.4. The **Byte Pair Encoding (BPE)** algorithm learns a vocabulary of **tokens** of given size $N > 256$ from a **corpus** \mathcal{C} , by doing the following:

- ▶ Let $\ell = 256$ and set $\text{BPE}(\langle b \rangle) = b$ for every byte $0 \leq b \leq 255$.
- ▶ While $\ell < N$, find the most common pair of **tokens** (a, b) and let $\text{BPE}(\langle a, b \rangle) = \ell + 1$ (and increase ℓ by 1).
- ▶ Repeat until $\ell = N$.

~> we obtain a one-hot encoding of **tokens** of size N , where the most common sequences of bytes are represented by a single **token**. By retaining $\text{BPE}(\langle b \rangle) = b$, we avoid OOV problems.

~> We can then train a **word embedding** on the resulting **tokens**

Alternative techniques include *WordPiece* and *SentencePiece*.

Tokenization - Example

<https://huggingface.co/spaces/Xenova/the-tokenizer-playground>



Definition 5.5. Let $\langle w_1, \dots, w_n \rangle$ be a sequence of **tokens**. A **positional encoding** $\text{PE}_i(w_i)$ is a vector that retains the position of w_i in the sequence *alongside* the **word embedding** of w_i .

We want **positional encodings** to satisfy the following properties:

1. $\text{PE}_i(w) \neq \text{PE}_j(w)$ for $i \neq j$,
2. **PE** should retain *distances*: if $i_1 - i_2 = j_1 - j_2$, then given the embeddings for w_1, w_2 , we should be able to linearly transform $\langle \text{PE}_{i_1}(w_1), \text{PE}_{i_2}(w_2) \rangle$ into $\langle \text{PE}_{j_1}(w_1), \text{PE}_{j_2}(w_2) \rangle$.

↪ no entirely separate embeddings for w_1, w_2 depending on positions

↪ learning from short sentences generalizes (ideally) to longer ones

Sinusoidal positional encoding

Idea: Let $\text{PE}_t(w) = E(w) + p_t$, for some suitable p_t (where $E(w)$ is the word embedding for token w).
 $\leadsto p_t$ has the same dimensionality as our embedding E .

Idea: Use a combination of sine and cosine functions with different frequencies for each dimension of the embedding.

Attention is all you need: For a vocabulary size d , we define

$$p_{ti} := \begin{cases} \sin\left(\frac{t}{c^{2k/d}}\right) & \text{if } i = 2k \\ \cos\left(\frac{t}{c^{2k/d}}\right) & \text{if } i = 2k + 1 \end{cases}$$

for some constant c .

(10000 in the paper)

\leadsto works for arbitrary sequence lengths and vocabulary sizes.

Three strategies for training LLMs:

- ▶ *Masked Token Prediction*: Given a sentence (e.g. “The river rose five feet”), randomly replace **tokens** by a special *mask token* (e.g. “The river [MASK] five feet”). The LLM should predict the masked **tokens** (e.g. “rose”).
(BERT et al; well suited for *generic tasks*)
 - ▶ *Discrimination*: Train a small masked token prediction model M . Given a masked sentence, let M generate possible completions. Train the actual model to distinguish between tokens generated by M and the *original* tokens.
(Google Electra et al; well suited for *generic tasks*)
 - ▶ *Next Token Prediction*: Given the (beginning of) a sentence, predict the next **token** in the sequence.
(GPT et al; well suited for *generative tasks*)
- ↪ All techniques turn an unlabelled **corpus** into a *supervised learning* task.

- ▶ Deep learning methods are currently dominant in NLP!
 - ▶ Data-driven methods are easier to develop and maintain than symbolic ones
 - ▶ also perform better models crafted by humans
- ▶ But problems remain;
 - ▶ DL methods work best on immense amounts of data.
 - ▶ LLM contain knowledge, but integration with symbolic methods elusive.

(think ChatGPT)

(with reasonable effort)

(small languages?)

- ▶ Deep learning methods are currently dominant in NLP! (think ChatGPT)
 - ▶ Data-driven methods are easier to develop and maintain than symbolic ones
 - ▶ also perform better models crafted by humans (with reasonable effort)
- ▶ But problems remain;
 - ▶ DL methods work best on immense amounts of data. (small languages?)
 - ▶ LLM contain knowledge, but integration with symbolic methods elusive.
- ▶ DL4NLP methods do very well, but only after processing orders of magnitude more data than humans do for learning language.
- ▶ This suggests that there is of scope for new insights from all areas.

Chapter 33

What did we learn in AI 1/2?

Topics of AI-1 (Winter Semester)

▶ Getting Started

- ▶ What is artificial intelligence?
- ▶ Logic programming in Prolog
- ▶ Intelligent Agents

(situating ourselves)
(An influential paradigm)
(a unifying framework)

▶ Problem Solving

- ▶ Problem Solving and search
- ▶ Adversarial search (Game playing)
- ▶ constraint satisfaction problems

(Black Box World States and Actions)
(A nice application of search)
(Factored World States)

▶ Knowledge and Reasoning

- ▶ Formal Logic as the mathematics of Meaning
- ▶ Propositional logic and satisfiability
- ▶ First-order logic and theorem proving
- ▶ Logic programming
- ▶ Description logics and semantic web

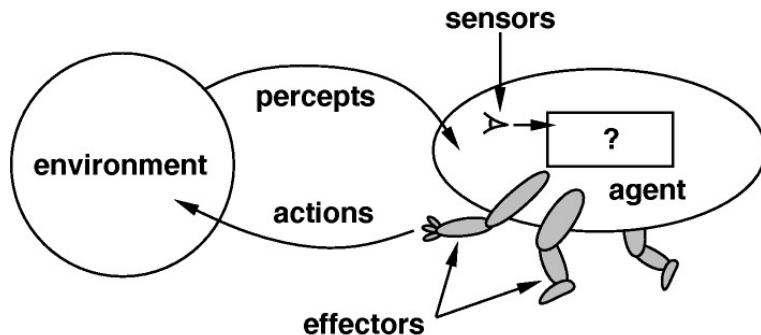
(Atomic Propositions)
(Quantification)
(Logic + Search \leadsto Programming)

▶ Planning

- ▶ Planning Frameworks
- ▶ Planning Algorithms
- ▶ Planning and Acting in the real world

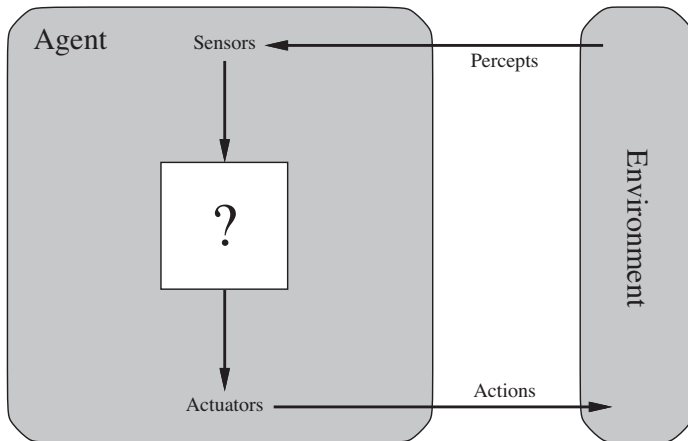
Rational Agents as an Evaluation Framework for AI

- Agents interact with the environment



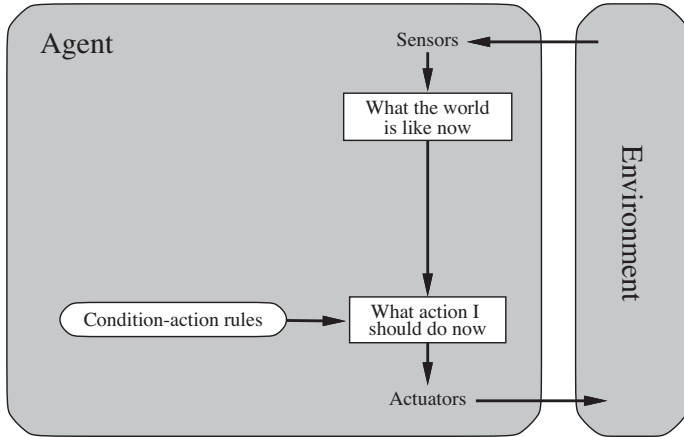
Rational Agents as an Evaluation Framework for AI

► General agent schema



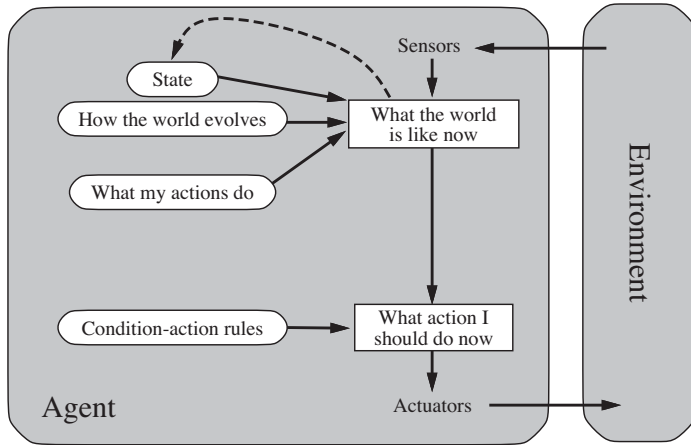
Rational Agents as an Evaluation Framework for AI

► Reflex Agents



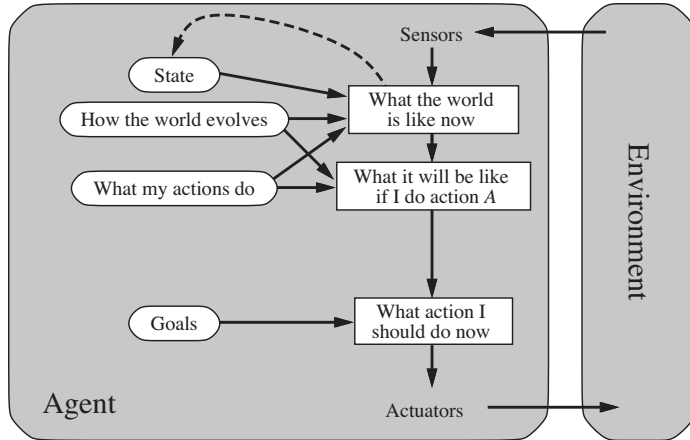
Rational Agents as an Evaluation Framework for AI

► Reflex Agents with State



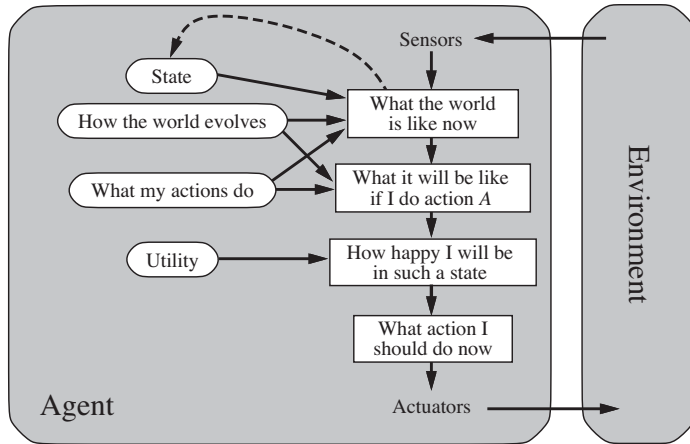
Rational Agents as an Evaluation Framework for AI

► Goal-Based Agents



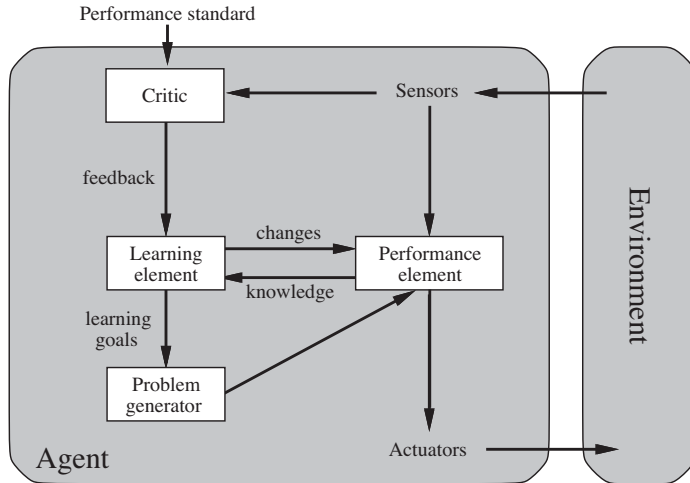
Rational Agents as an Evaluation Framework for AI

► Utility-Based Agent



Rational Agents as an Evaluation Framework for AI

► Learning Agents



- ▶ **Idea:** Try to design **agents** that are successful (do the right thing)
- ▶ **Definition 0.1.** An **agent** is called **rational**, if it chooses whichever **action** **maximizes** the expected value of the performance measure given the **percept** sequence to date. This is called the **MEU principle**.
- ▶ **Note:** A **rational agent** need not be perfect
 - ▶ only needs to **maximize expected value** (rational \neq omniscient)
 - ▶ need not predict e.g. very unlikely but catastrophic events in the future
 - ▶ **percepts** may not supply all relevant information (Rational \neq clairvoyant)
 - ▶ if we cannot perceive things we do not need to react to them.
 - ▶ but we may need to try to find out about hidden dangers (exploration)
 - ▶ **action** outcomes may not be as expected (rational \neq successful)
 - ▶ but we may need to take **action** to ensure that they do (more often) (learning)
- ▶ **Rational** \leadsto exploration, learning, autonomy

Symbolic AI: Adding Knowledge to Algorithms

- ▶ Problem Solving

- ▶ **Framework:** Problem Solving and Search
- ▶ **Variant:** Game playing (**Adversarial search**)

(**Black Box States**, **Transitions**, **Heuristics**)

(**basic tree/graph walking**)
(**minimax** + $\alpha\beta$ -Pruning)

Symbolic AI: Adding Knowledge to Algorithms

- ▶ Problem Solving (Black Box States, Transitions, Heuristics)
 - ▶ **Framework**: Problem Solving and Search (basic tree/graph walking)
 - ▶ **Variant**: Game playing (Adversarial search) (minimax + $\alpha\beta$ -Pruning)
- ▶ Constraint Satisfaction Problems (heuristic search over partial assignments)
 - ▶ States as partial variable assignments, transitions as assignment
 - ▶ **Heuristics** informed by current restrictions, constraint graph
 - ▶ Inference as constraint propagation (transferring possible values across arcs)

Symbolic AI: Adding Knowledge to Algorithms

- ▶ Problem Solving (Black Box States, Transitions, Heuristics)
 - ▶ **Framework**: Problem Solving and Search (basic tree/graph walking)
 - ▶ **Variant**: Game playing (Adversarial search) (minimax + $\alpha\beta$ -Pruning)
- ▶ Constraint Satisfaction Problems (heuristic search over partial assignments)
 - ▶ States as partial variable assignments, transitions as assignment
 - ▶ **Heuristics** informed by current restrictions, constraint graph
 - ▶ Inference as constraint propagation (transferring possible values across arcs)
- ▶ Describing world states by formal language (and drawing inferences)
 - ▶ Propositional logic and DPLL (deciding entailment efficiently)
 - ▶ First-order logic and ATP (reasoning about infinite domains)
 - ▶ **Digression**: Logic programming (logic + search)
 - ▶ Description logics as moderately expressive, but decidable logics

Symbolic AI: Adding Knowledge to Algorithms

- ▶ Problem Solving (Black Box States, Transitions, Heuristics)
 - ▶ **Framework:** Problem Solving and Search (basic tree/graph walking)
 - ▶ **Variant:** Game playing (Adversarial search) (minimax + $\alpha\beta$ -Pruning)
- ▶ Constraint Satisfaction Problems (heuristic search over partial assignments)
 - ▶ States as partial variable assignments, transitions as assignment
 - ▶ **Heuristics** informed by current restrictions, constraint graph
 - ▶ Inference as constraint propagation (transferring possible values across arcs)
- ▶ Describing world states by formal language (and drawing inferences)
 - ▶ Propositional logic and DPLL (deciding entailment efficiently)
 - ▶ First-order logic and ATP (reasoning about infinite domains)
 - ▶ **Digression:** Logic programming (logic + search)
 - ▶ Description logics as moderately expressive, but decidable logics
- ▶ **Planning:** Problem Solving using white-box world/action descriptions
 - ▶ **Framework:** describing world states in logic as sets of propositions and actions by preconditions and add/delete lists
 - ▶ **Algorithms:** e.g. heuristic search by problem relaxations

Topics of AI-2 (Summer Semester)

- ▶ Uncertain Knowledge and Reasoning
 - ▶ Uncertainty
 - ▶ Probabilistic reasoning
 - ▶ Making Decisions in Episodic Environments
 - ▶ Problem Solving in Sequential Environments
- ▶ Foundations of machine learning
 - ▶ Learning from Observations
 - ▶ Knowledge in Learning
 - ▶ Statistical Learning Methods
- ▶ Communication
 - ▶ Natural Language Processing
 - ▶ Natural Language for Communication

(If there is time)

Statistical AI: Adding uncertainty and Learning

- ▶ Problem Solving under **uncertainty** (non-observable environment, stochastic states)
 - ▶ **Framework**: Probabilistic Inference: Conditional Probabilities/Independence
 - ▶ **Intuition**: Reasoning in Belief Space instead of State Space!
 - ▶ **Implementation**: Bayesian Networks (exploit conditional independence)
 - ▶ **Extension**: Utilities and Decision Theory (for static/episodic environments)

Statistical AI: Adding uncertainty and Learning

- ▶ Problem Solving under **uncertainty** (non-observable environment, stochastic states)
 - ▶ **Framework**: Probabilistic Inference: Conditional Probabilities/Independence
 - ▶ **Intuition**: Reasoning in Belief Space instead of State Space!
 - ▶ **Implementation**: Bayesian Networks (exploit conditional independence)
 - ▶ **Extension**: Utilities and Decision Theory (for static/episodic environments)
- ▶ Problem Solving in Sequential Worlds:
 - ▶ **Framework**: Markov Processes, transition models
 - ▶ **Extension**: MDPs, POMDPs (+ utilities/decisions)
 - ▶ **Implementation**: Dynamic Bayesian Networks

Statistical AI: Adding uncertainty and Learning

- ▶ Problem Solving under **uncertainty** (non-observable environment, stochastic states)
 - ▶ **Framework**: Probabilistic Inference: Conditional Probabilities/Independence
 - ▶ **Intuition**: Reasoning in Belief Space instead of State Space!
 - ▶ **Implementation**: Bayesian Networks (exploit conditional independence)
 - ▶ **Extension**: Utilities and Decision Theory (for static/episodic environments)
- ▶ Problem Solving in Sequential Worlds:
 - ▶ **Framework**: Markov Processes, transition models
 - ▶ **Extension**: MDPs, POMDPs (+ utilities/decisions)
 - ▶ **Implementation**: Dynamic Bayesian Networks
- ▶ **Machine learning**: adding optimization in changing environments (unsupervised)
 - ▶ **Framework**: Learning from Observations (positive/negative examples)
 - ▶ **Intuitions**: finding consistent/optimal hypotheses in a hypothesis space
 - ▶ **Problems**: consistency, expressivity, under/overfitting, computational/data resources.
 - ▶ Extensions
 - ▶ knowledge in learning (based on logical methods)
 - ▶ statistical learning (optimizing the probability distribution over hypspace, learning BNs)

Statistical AI: Adding uncertainty and Learning

- ▶ Problem Solving under **uncertainty** (non-observable environment, stochastic states)
 - ▶ **Framework**: Probabilistic Inference: Conditional Probabilities/Independence
 - ▶ **Intuition**: Reasoning in Belief Space instead of State Space!
 - ▶ **Implementation**: Bayesian Networks (exploit conditional independence)
 - ▶ **Extension**: Utilities and Decision Theory (for static/episodic environments)
- ▶ Problem Solving in Sequential Worlds:
 - ▶ **Framework**: Markov Processes, transition models
 - ▶ **Extension**: MDPs, POMDPs (+ utilities/decisions)
 - ▶ **Implementation**: Dynamic Bayesian Networks
- ▶ **Machine learning**: adding optimization in changing environments (unsupervised)
 - ▶ **Framework**: Learning from Observations (positive/negative examples)
 - ▶ **Intuitions**: finding consistent/optimal hypotheses in a hypothesis space
 - ▶ **Problems**: consistency, expressivity, under/overfitting, computational/data resources.
 - ▶ Extensions
 - ▶ knowledge in learning (based on logical methods)
 - ▶ statistical learning (optimizing the probability distribution over hypspace, learning BNs)
 - ▶ Communication
 - ▶ Phenomena of natural language (NL is interesting/complex)
 - ▶ symbolic/statistical NLP (historic/as a backup)
 - ▶ Deep Learning for NLP (the current hype/solution)

Topics of AI-3 – A Course not taught at FAU ☹

- ▶ Machine Learning
 - ▶ Theory and Practice of Deep Learning
 - ▶ More Reinforcement Learning
- ▶ Communicating, Perceiving, and Acting
 - ▶ More NLP, dialogue, speech acts, ...
 - ▶ Natural Language Semantics/Pragmatics
 - ▶ Perception
 - ▶ Robotics
 - ▶ Emotions, Sentiment Analysis
- ▶ **The Good News:** All is not lost
 - ▶ There are tons of specialized courses at FAU
 - ▶ Russell/Norvig's AIMA [RusNor:AIMA09] cover some of them as well!

(more as we speak)

