Künstliche Intelligenz 2 – SS 2018

Michael Kohlhase
Informatik, FAU Erlangen-Nürnberg
For Course Purposes Only

June 26, 2018

Contents

1 Assignment 1 (Conditional Probabilities) – Given April 22., Due April 29.

Problem 1.1 (AFT Tests)

Trisomy 21 (*Down syndrome*) is a genetic anomaly that can be diagnosed during pregnancy using an amniotic fluid test. 25pt

The probability of a fetus having Down syndrome is strongly correllated with the age of the mother during pregnancy. For 25 year old mothers the probability is one in 1250, for 43 year old mothers it increases to one in fifty (we only consider those two age groups).

However, diagnostic tests are never perfect. We distinguish two kinds of errors:

- **Type I Error (False Positive)**: The test result is positive even though the child is healthy.
- **Type II Error (False Negative)**: The test result is negative even though the child has trisomy 21.

The probabilities of Type I and Type II Errors are both merely 1% for amniotic fluid tests for Down syndrome.

10 pt

1. Express all of the above in the form of conditional probabilities. Use the random variable F with Domain $\{Age_{25}, Age_{43}\}$ for the age of a mother and the boolean random variables Pos and $Down$ for the propositions "*The amniotic fluid test is positive*" and "*The child has Down syndrome*" respectively.

15 pt

2. Express and compute the probability that a child has Down syndrome, given that the mother is 25 years old and the amniotic fluid test is positive. What can we conclude from the result?

Solution:

1. $P(Down \mid F = Age_{25}) = 0.0008$, $P(Down \mid F = Age_{43}) = 0.02$, $P(Pos \mid \neg Down) = 0.01$, $P(\neg Pos \mid Down) = 0.01$.
2. We normalize to $F = Age_{25}$ and compute:

$$\begin{aligned} P(Down \mid Pos) &= \frac{P(Pos \mid Down) \cdot P(Down)}{P(Pos)} = \frac{P(Pos \mid Down) \cdot P(Down)}{P(Pos \wedge Down) + P(Pos \wedge \neg Down)} \\ &= \frac{P(Pos \mid Down) \cdot P(Down)}{P(Pos \mid Down) \cdot P(Down) + P(Pos \mid \neg Down \cdot P(\neg Down))} \\ &= \frac{(1 - P(\neg Pos \mid Down)) \cdot P(Down)}{(1 - P(\neg Pos \mid Down) \cdot P(Down)) + P(Pos \mid \neg Down) \cdot (1 - P(Down))} \\ &= \frac{0.99 \cdot 0.0008}{0.99 \cdot 0.0008 + 0.01\ 0.9992} \approx 0.07 \end{aligned}$$

So, even with a positive test result, the probability of the child actually having Down syndrome is still only 7%, simply due to Down syndrome being relatively rare in young mothers. Consequently, there is little point in applying this particular test without exceptional cause for concern.

Problem 1.2 (Chained Production Elements)

An apparatus consists of six element A, B, C, D, E, F. The apparatus works if and only if either A and B are operational, C and D are operational, or E and F are operational. 30pt

1. Assume the probabilities $P(X)$, that element X breaks down, are all stochastically independent, with $P(A) = 5\%$, $P(B) = 10\%$, $P(C) = 15\%$, etc. What is the probability the apparatus works? 10 pt
2. Unfortunately, the elements A and C, D and F and B and E are pairwise linked; such that if either of them breaks, then the linked element is not operational either. What is the probability the apparatus works now? 20 pt

(Note that we deliberately differentiate between *not being operational* and *being broken*!)

Solution: Let W be the random variable stating that the apparatus works.

1.

$$\begin{aligned}
P(W) &= P((A \wedge B) \vee (C \wedge D) \vee (E \wedge F)) \\
&= 1 - P(\underbrace{\neg(A \wedge B) \wedge \neg(C \wedge D) \wedge \neg(E \wedge F)}_{\text{all events are independent}}) \\
&= 1 - P(\neg(A \wedge B)) \cdot P(\neg(C \wedge D)) \cdot P(\neg(E \wedge F)) \\
&= 1 - P(\neg A \vee \neg B) \cdot P(\neg C \vee \neg D) \cdot P(\neg E \vee \neg F) \\
&= 1 - (P(\neg A) + P(\neg B) - P(\neg A) \cdot P(\neg B)) \cdot (P(\neg C) + P(\neg D) - P(\neg C) \cdot P(\neg D)) \\
&\cdot (P(\neg E) + P(\neg F) - P(\neg E) \cdot P(\neg F)) \\
&= 1 - (0.05 + 0.1 - (0.05 \cdot 0.1)) \cdot (0.15 + 0.2 - (0.15 \cdot 0.2)) \cdot (0.25 + 0.3 - (0.25 \cdot 0.3))
\end{aligned}$$

2. Using the exclusion principle:

$$\begin{aligned}
P(W) &= P((A \wedge B) \vee (C \wedge D) \vee (E \wedge F)) \\
&= P(A, B) + P(C, D) + P(E, F) - P(A, B, C, D) - P(A, B, E, F) - P(C, D, E, F) \\
&+ P(A, B, C, D, E, F)
\end{aligned}$$

Due to the links, this is equivalent to:

$$\begin{aligned}
P(W) &= P(A, C, B, E) + P(A, C, D, F) + P(B, E, D, F) - P(A, B, C, D, E, F) - P(A, B, C, D, E, F) \\
&- P(A, B, C, D, E, F) + P(A, B, C, D, E, F) \\
&= P(A, C, B, E) + P(A, C, D, F) + P(B, E, D, F) - 2P(A, B, C, D, E, F) \\
&= P(A)P(B)P(C)P(E) + P(A)P(C)P(D)P(F) + P(B)P(D)P(E)P(F) \\
&- 2P(A)P(B)P(C)P(D)P(E)P(F) \\
&= 0.5450625 + 0.4522 + 0.378 - 2 \cdot 0.305235 \approx 76\%
\end{aligned}$$

Problem 1.3 (Bayesian Epistemology)

Consider the following sayings. How can we express them in the form of conditional probabilities? Give actual mathematical formulas and explain how they relate to the sayings. 45pt

Are they (as statements about probabilities) actually true or under which assumptions can they be?

- *Occam's Razor:* The simplest explanation is always the best.
- Extraordinary claims require extraordinary evidence.
- Absence of evidence is not evidence of absence.

Solution: Open question, but here are some ways to think about these things:

- Obviously, we have a problem to quantify what *simple* exactly means. So let's take one step back and think about what it means for one hypothesis A to be simpler than some hypothesis B.

  One way to answer that question would be to say: A is *simpler* than B if the set of propositions entailed by A is a proper subset of those entailed by B. In this case we can say $A \equiv P_1 \wedge \ldots \wedge P_n$ and $B \equiv P_1 \wedge \ldots \wedge P_n \wedge P_{n+1} \wedge \ldots \wedge P_m$.

  Naturally, we have $P(A) \geq P(A \wedge B)$ for any propositions A, B, hence under this interpretation the claim is true.

- Let's assume "extraordinary" means that the prior probability (in the absence of any evidence) is rather small, i.e. $P(A) \approx 0$[1]. If we want the claim A to be *likely*, we need to find some evidence e such that $P(A \mid e) \approx 1$. By Bayes' Theorem:

$$1 \approx P(A \mid e) = \frac{P(e \mid A) \cdot \overbrace{P(A)}^{\approx 0}}{P(e)}$$

  It is immediately obvious that for this to hold, $P(e)$ needs to be highly unlikely, i.e. extraordinary.

- Let's assume that e is evidence *for* A iff $P(A \mid e) > P(A)$, i.e. observing e actually makes the proposition A more likely. I claim: *If e is evidence for A, then $\neg e$ is evidence for $\neg A$*, making the claim false:

[1] I'll write $P(x) \approx 0$ resp. ≈ 1 simply for "is very unlikely" and "is very likely"

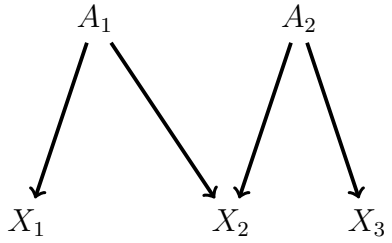
$$
\begin{aligned}
&P(A \mid e) = \frac{P(e \mid A)P(A)}{P(e)} > P(A) \\
\Rightarrow &P(e \mid A)P(A) > P(e)P(A) \\
\Rightarrow &(1 - P(\neg e \mid A))P(A) > (1 - P(\neg e))P(A) \\
\Rightarrow &\underbrace{P(\neg e \mid A)P(A)}_{=P(\neg e, A)} < P(\neg e)P(A) \\
\Rightarrow &\overbrace{P(A \mid \neg e)P(\neg e)} < P(\neg e)P(A) \\
\Rightarrow &P(A \mid \neg e) < P(A) \\
\Rightarrow &1 - P(\neg A \mid \neg e) < 1 - P(\neg A) \\
\Rightarrow &P(\neg A \mid \neg e) > P(\neg A)
\end{aligned}
$$

□

---

2 Assignment 2 (Bayesian Networks) – Given April 29., Due May 06.

Problem 2.1 (Bayesian Networks)
Consider the following Bayesian network with boolean variables: 20pt



1. Which nodes in the network are
   - Stochastically independent
   - Conditionally independent and under which conditions?
2. What exactly (formal criterion) does an arrow between two nodes in a bayesian network mean for the associated events?

Solution:

1. - A_1 and A_2 are stochastically independent.
   - X_1 and X_3 are stoachastically independent.
   - A_1 and X_3 (respectively A_2 and X_1) are stochastically independent.
   - X_1 and X_2 are conditionally independent given A_1.
   - X_2 and X_3 are conditionally independent given A_2.
2. We draw an arrow from X_j to X_i if X_j is in the smallest set $\texttt{Parents}(X_i)$ with the property $P(X_i|X_{i-1},...,X_1) = P(X_i|\texttt{Parents}(X_i))$

Problem 2.2 (Nuclear Test)
Assume it is your responsibility to monitor the Nuclear Test Ban treaty. You receive data from two different stations (seismometers), S_1 and S_2. Each S_i is modeled as a Boolean variable where "true" stands for "I detected a Nuclear test" and "false" stands for "I did not detect a Nuclear test". The seismometers are not fully reliable, however; they may not detect a Nuclear test even though there was one, and they may mistake an earthquake for a Nuclear test. We model this situation with two additional Boolean variables: N for Nuclear test, and E for Earthquake. 30pt

Use the algorithm from the lecture (Slide 667) to construct a Bayesian network for these 4 variables. Do so for the following two variable orders:

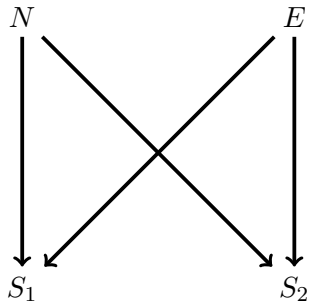
(a) $X_1 = N$, $X_2 = E$, $X_3 = S_1$, $X_4 = S_2$.

(b) $X_1 = S_1$, $X_2 = S_2$, $X_3 = E$, $X_4 = N$.

For each of these orders, draw the resulting Bayesian network. Justify your design, i.e., for each variable X_i added to the network explain why the set of parents you give X_i are needed, and why they are sufficient.

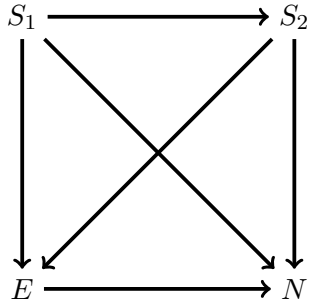
Solution:

(a) With this variable order, we get the following network:



$X_2 = E$ does not need $X_1 = N$ as a parent because Earthquakes are independent from Nuclear tests. $x_3 = S_1$ needs both $X_1 = N$ and $X_2 = E$ as parents because each of these may influence the measurement; same for $X_4 = S_2$, i.e., here we also need the parents $X_1 = N$ and $X_2 = E$. However, given the values of N and E, the measurements of $X_3 = S_1$ and $x_4 = S_2$ are independent. So $X_4 = S_2$ does not require the parent $X_3 = S_1$.

(b) With this variable order, we get the following network:



$X_2 = S_2$ needs $X_1 = S_1$ as a parent because, if S_1 detects a seismic phenomenon, then chances are higher S_2 will detect one as well. $X_3 = E$ needs each of $X_1 = S_1$ and $X_2 = S_2$ as parents because, if a station detects a seismic phenomenon, then chances are higher there was an earthquake; same for $X_4 = N$, i.e., here we also need the parents $X_1 = S_1$ and $X_2 = S_2$ because measurements indicate Nuclear tests as well. Finally, say we already know that S_1 and S_2 are true; then the value of E still has an influence on the value of N: If there was an earthquake, then there is a chance that the seismic measurements were caused by the earthquake rather than a Nuclear test. Thus N is *not* conditionally independent of E given S_1 and S_2, and we need $X_3 = E$ as a parent of $X_4 = N$ as well.

Problem 2.3 (Medical Bayesian Network 2)

Both Malaria and Meningitis can cause a measured high body temperature; the latter is an indicator of fever. We consider the following random variables for a given patient: 50pt

- Mal: The patient has malaria (probability 5%).
- Men: The patient has meningitis (probability 20%).
- HBT: The patient has a high body temperature:

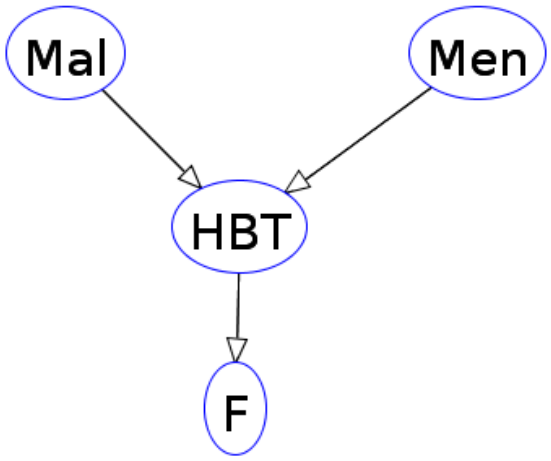
Mal	Men	HBT
$\top$	$\top$	95%
$\top$	$\bot$	90%
$\bot$	$\top$	90%
$\bot$	$\bot$	0.5%

- F: The patient has a fever:

HBT	F
$\top$	95%
$\bot$	0.2%

1. Draw the corresponding Bayesian network for the above data.
2. What is the probability some patient has malaria and not meningitis, given that he has a fever?

Solution:



We need to compute all four cases $\{\pm Mal, \pm Men\}$:

$$\begin{aligned}
\langle P(v_{Mal}, v_{Men}|F)\rangle &= \alpha \left\langle \sum_{v_{HBT}} P(v_{HBT}, F, v_{Men}, v_{Mal}) \right\rangle \\
&= \alpha \left\langle \sum_{v_{HBT}} (P(F|v_{HBT}) \cdot P(v_{HBT}|v_{Men}, v_{Mal}) \cdot P(v_{Men}) \cdot P(v_{Mal})) \right\rangle \\
&= \alpha \left\langle P(v_{Men}) \cdot P(v_{Mal}) \cdot \sum_{v_{HBT}} \underbrace{(P(F|v_{HBT}) \cdot P(v_{HBT}|v_{Men}, v_{Mal}))}_{:=\sigma_{v_{HBT}, v_{Men}, v_{Mal}}} \right\rangle
\end{aligned}$$

, hence:

$$\begin{aligned}
P(Mal, Men|F) &= \alpha \cdot P(Men) \cdot P(Mal) \cdot (\sigma_{HBT,Men,Mal} + \sigma_{\neg HBT,Men,Mal}) \\
&= \alpha \cdot 0.2 \cdot 0.05 \cdot (0.95 \cdot 0.95 + 0.002 \cdot 0.05) = \alpha \cdot 0.009026 \\
P(Mal, \neg Men|F) &= \alpha \cdot P(\neg Men) \cdot P(Mal) \cdot (\sigma_{HBT,\neg Men,Mal} + \sigma_{\neg HBT,\neg Men,Mal}) \\
&= \alpha \cdot 0.8 \cdot 0.05 \cdot (0.95 \cdot 0.9 + 0.002 \cdot 0.1) = \alpha \cdot 0.034208 \\
P(\neg Mal, Men|F) &= \alpha \cdot P(Men) \cdot P(\neg Mal) \cdot (\sigma_{HBT,Men,\neg Mal} + \sigma_{\neg HBT,Men,\neg Mal}) \\
&= \alpha \cdot 0.2 \cdot 0.95 \cdot (0.95 \cdot 0.9 + 0.002 \cdot 0.1) = \alpha \cdot 0.162488 \\
P(\neg Mal, \neg Men|F) &= \alpha \cdot P(\neg Men) \cdot P(\neg Mal) \cdot (\sigma_{HBT,\neg Men,\neg Mal} + \sigma_{\neg HBT,\neg Men,\neg Mal}) \\
&= \alpha \cdot 0.8 \cdot 0.95 \cdot (0.95 \cdot 0.005 + 0.002 \cdot 0.995) = \alpha \cdot 0.0051224
\end{aligned}$$

Therefore:

$$\alpha\langle 0.009026, 0.034208, 0.162488, 0.0051224\rangle \approx \langle 4.3\%, 16.2\%, 77\%, 2.4\%\rangle$$

Hence, $P(Mal, \neg Men|F) \approx 16.2\%$

3 Assignment 3 (Bayesian Networks) – Given May 06., Due May 13.

Problem 3.1 (Bayesian Networks in Java)

100pt

The goal of this exercise is to implement inference by enumeration (or aonther inference algorithm of your choice) in bayesian networks in java. Use the attached java classes. Those are:

- Node: This class has two fields: A string id for the name, and a field of type Node[] called parents (for the parents of the node). To initialize the probabilities of a node, you can use the method void setProb(Boolean[] given, double value). To access the probability of a node (given values for the parents), use public double getProb(Boolean[] given).
- Nodes are collected in an object of the class Network, which has the two methods public ArrayList<Node> getNodes() and public void addNode(Node n).

The final class is Query. Your taks is to implement an extension of this class, which requires you to implement its method

public Double query(Network network, Node node, ArrayList<Pair<Node, Boolean>>$\dot{>}$ evidence);

,which is supposed to compute the probability $P(\mathsf{node}|\mathsf{evidence})$ (where evidence is a list of pairs of the form $\mathsf{node} = \top/\bot$).

You can test your network with the Test class, which contains an implementation of the Network from the following exercise, which you are not required to solve.

Hand in your solution as a java file containing a (uniquely named!) class extending Query.

Solution:

Problem 3.2 (Medical Bayesian Network)

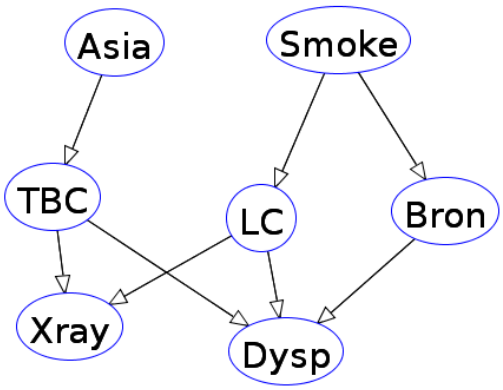
Dyspnea (shortness of breath) can be caused by several medical conditions; among them lung cancer, tuberculosis and bronchitis. Tuberculosis and cancer lead to abnormal x-ray results. Lung cancer and bronchitis can be caused by smoking, tuberculosis occurs more often in asia. We use the following random variables for some given patient: 0pt

- *Asia*: The patient recently visited asia.
- *Smoke*: The patient is a smoker.
- TBC: The patient has tuberculosis.
- LC: The patient has lung cancer.
- *Bron*: The patient has bronchitis.

- $Xray$: The patient's X-ray result is abnormal.
- $Dysp$: The patient is short of breath.

Model the dependencies stated above as a bayesian network.

Solution:



4 Assignment 4 (Rational Preferences) – Given May 13., Due May 20.

Problem 4.1 (Ramsey's Theorem)
According to Ramsey's Theorem, for every set of preferences satisfying the constraints in Definition 20.2.3, there exists a utility function that respects these preferences. 100pt

Choose one of the following:

1. For 20 bonus points: Implement an algorithm in Scala that computes a utility function from a set of precedences using the Scala classes attached.

   As always, hand in your solution as a single .scala-file containing a *uniquely named* class extending Utility.

2. If you're unfamiliar with Scala, you can alternatively:

   (a) Describe an algorithm (in pseudocode) that – given a set of preferences Σ – returns a utility function as in Ramsey's Theorem.

   (b) Describe in detail how your algorithm operates on the following set of preferences:

   - $A \prec B$
   - $A \prec C$
   - $E \prec B$
   - $E \prec D$
   - $C \sim E$
   - $[0.2, A; 0.8, B] \prec [0.3, C; 0.7, D]$
   - $[0.3, C; 0.7, D] \prec [0.25, E; 0.75, D]$

Solution:

5 Assignment 5 (Decisions and Utilities) – Given May 20., Due May 27.

Problem 5.1 (Textbook Decisions)

Abby has to decide whether to buy Russell&Norveig for 100$. There are three boolean variables involved in this decision: B indicating whether Abby buys the book, M indicating whether Abby knows the material in the book perfectly anyway and P indicating that Abby passes the course. Additionally, we use a utility node U. 50pt

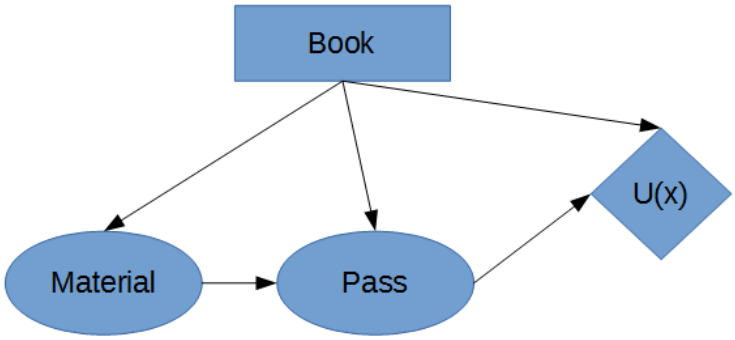
Abbys utility function is additive, so $U(B) = -100$. Furthermore, she evaluates passing the course with a utility of $U(P) = 2000$. The course has an open book final exam, so B and P are not independent given M. The conditional probabilities are as follows:

- $P(P|B, M) = 0.95$
- $P(P|B, \neg M) = 0.45$
- $P(P|\neg B, M) = 0.9$
- $P(P|\neg B, \neg M) = 0.2$
- $P(M|B) = 0.85$
- $P(M|\neg B) = 0.6$

1. Draw the decision network for this problem. 20 pt
2. Compute the expected utility of buying the book and of not buying it. 30 pt

---

Solution: (Note that the numbers are wrong/outdated)



P	B	Utility
F	F	0
F	T	-100
T	F	2000
T	T	1900

B	P(B)
F	0.7
T	0.9

B	M	P(P)
F	F	0.3
F	T	0.8
T	F	0.5
T	T	0.9

Assuming B:
P(B,P,M) = P(P | M,B) · P(B) · P(M | B) = 0.95 · 1 · 0.85 = 0.8075
P(B,P,¬M) = 0.45 · 1 · 0.15 = 0.0675
P(B,¬P,M) = 0.05 · 1 · 0.85 = 0.0425
P(B,¬P,¬M) = 0.55 · 1 · 0.15 = 0.0825

P(B,P) = 0.8075 + 0.0675 = 0.875
P(B,¬P) = 0.0425 + 0.0825 = 0.125
→ U(B) = 0.875 · 1900 + 0.125 · −100 = 1650

Assuming ¬B:
P(¬ B,P,M) = P(P | M,¬B) · P(¬B) · P(M |¬B) = 0.9 · 1 · 0.6 = 0.54
P(¬B,P,¬M) = 0.2 · 1 · 0.4 = 0.08
P(¬B,¬P,M) = 0.1 · 1 · 0.6 = 0.06
P(¬B,¬P,¬M) = 0.8 · 1 · 0.4 = 0.32

→ P(¬B,P) = 0.54 + 0.08 = 0.62
P(¬B,¬P) = 0.06 + 0.32 = 0.38
→ U(¬B) = 0.62 · 2000 + 0.38 · 0 = 1240

So buying the book has the higher utility.

Problem 5.2 (Decision Theory)
You are offered the following game: You pay x dollars to play. A fair coin is then tossed repeatedly until it comes up heads for the first time. Your payout is 2^n, where n is the number of tosses that occurred. 50pt

- Assume your utility function is exactly the monetary value. How much should you, as a rational agent, be willing to pay to play? Use the formal definition of "expected utility" from the lecture.
- Assume now, that your utility function for having k dollars is $U(k) = 5 \log_2 k$. How does this change the result?
- What is "wrong" with the result from the first exercise? Which "implicit assumption" leads to the apparently nonsensical result? Can you think of a way to "repair" our utility function in a more "realistic" way than taking logarithms?

Solution:

- We have

$$EU(\mathsf{Play}) = \sum_{s'}(P(\mathsf{Payout} = s'|\mathsf{Play}) \cdot U(s') = \sum_{k\in\mathbb{N}^+} P(\mathsf{Payout} = k) \cdot 2^k$$
$$= \sum_{k\in\mathbb{N}^+} \frac{1}{2^k} \cdot 2^k = \sum_{k\in\mathbb{N}^+} 1 \to \infty$$

  We should be willing to pay any amount for a chance to play.

-

$$EU(\mathsf{Play}) = \sum_{k\in\mathbb{N}^+} \frac{1}{2^k} \cdot 5\log_2(2^k) = 5\sum_{k\in\mathbb{N}^+} \frac{k}{2^k} = 5 \cdot 2 = 10$$

  Of course, this is wrong insofar as the utility, being logarithmic, is in particular not linear, i.e. the actual utility depends on our original capital K, but this just makes everything more complicated. The point is that using a logarithmic utility for money yields a finite result.

- This model assumes that the payout is potentially infinite (which is unrealistic), as well as that we have an unlimited amount of money at our disposal. If we add the (negative!) utility of the cost, we can set an "upper limit" of how much we can afford to pay in the first place; e.g. by setting $U(k) = -\infty$ for k being the amount in our bank account.

---

6 Assignment 6 (Markov Processes) – Given May 27., Due June 3.

Problem 6.1 (Markov Mood Detection)

On any given day d, your flatmate Moody is in one of two states – either he is happy (H_d) or he is not. Usually when he's in a bad mood, it's because he had a fight with his spouse and those tend to go on for a couple of days, so $P(\neg H_{d+1}|\neg H_d) = 0.7$, but aside from that he's a cheery guy, so $P(H_{d+1}|H_d) = 0.85$. 100pt

Notably, you can hear his music blasting all day which tends to shift depending on his mood. On a good day he often listens to Jazz (i.e. $P(J_d|H_d) = 0.6$), on a bad day he usually blasts Death Metal at full volume ($P(\neg J_d|\neg H_d) = 0.85$). He has a limited taste in music, so it's always one of the two.

It's Friday now and and you can hear Jazz playing from his room, but Tuesday through Thursday you were constantly bombarded by Death Metal through your paper thin walls. You need to talk to him about his disregard for kitchen hygiene, but as a mentally healthy computer science student, you obviously try to avoid talking to people as much as possible – even more so when they're in a bad mood!

40 pt

1. Assume you happen to know he was in a good mood on Monday. Which algorithm is adequate for computing the probability that he's in a good mood today? What is that probability?

30 pt

2. On second thought, maybe you'd rather wait until Sunday. Which algorithm is adequate for computing the probability that he'll be in a good mood then? What is that probability?

30 pt

3. Maybe it would have been better to talk to him on Wednesday? Which algorithm is best suited and what is that probability?

Solution:

1. **Filtering:** We assume $P(H_0) = 1$. Let h_i our (hidden) variables and $e_1 = \neg J_1$, $e_2 =$

$\neg J_2 \wedge \neg J_1$, etc. Then:

$$\begin{aligned}
\langle P(h_1|e_1)\rangle &= \alpha \langle P(\neg J_1|h_1)P(h_1|H_0)\rangle \\
&= \alpha \left\langle \underbrace{P(\neg J_1|H_1)}_{0.4}\underbrace{P(H_1|H_0)}_{0.85}, \underbrace{P(\neg J_1|\neg H_1)}_{0.85}\underbrace{P(\neg H_1|H_0)}_{0.15} \right\rangle \\
&= \alpha \langle 0.34, 0.1275\rangle \approx \langle 0.73, 0.27\rangle \\
\langle P(h_2|e_2)\rangle &= \alpha \left\langle P(\neg J_2|h_2) \sum_{h_1} P(h_2|h_1)P(h_1|e_1) \right\rangle \\
&= \alpha \left\langle \underbrace{P(\neg J_2|H_2)}_{0.4} \underbrace{\sum_{h_1} P(H_2|h_1)P(h_1|e_1)}_{=0.85\cdot 0.73+0.3\cdot 0.27=0.7015}, \underbrace{P(\neg J_2|\neg H_2)}_{0.85} \underbrace{\sum_{h_1} P(\neg H_2|h_1)P(h_1|e_1)}_{=0.15\cdot 0.73+0.7\cdot 0.27=0.2985} \right\rangle \\
&= \alpha \langle 0.2806, 0.253725\rangle \approx \langle 0.53, 0.47\rangle \\
\langle P(h_3|e_3)\rangle &= \alpha \left\langle P(\neg J_3|h_3) \sum_{h_2} P(h_3|h_2)P(h_2|e_2) \right\rangle \\
&= \alpha \left\langle \underbrace{P(\neg J_3|H_3)}_{0.4} \underbrace{\sum_{h_2} P(H_3|h_2)P(h_2|e_2)}_{=0.85\cdot 0.53+0.3\cdot 0.47=0.5915}, \underbrace{P(\neg J_3|\neg H_3)}_{0.85} \underbrace{\sum_{h_2} P(\neg H_3|h_2)P(h_2|e_2)}_{=0.15\cdot 0.53+0.7\cdot 0.47=0.4085} \right\rangle \\
&= \alpha \langle 0.2366, 0.347225\rangle \approx \langle 0.41, 0.59\rangle \\
\langle P(h_4|e_4)\rangle &= \alpha \left\langle P(J_4|h_4) \sum_{h_3} P(h_4|h_3)P(h_3|e_3) \right\rangle \\
&= \alpha \left\langle \underbrace{P(J_4|H_4)}_{0.6} \underbrace{\sum_{h_3} P(H_4|h_3)P(h_3|e_3)}_{=0.85\cdot 0.41+0.3\cdot 0.59=0.5255}, \underbrace{P(J_4|\neg H_4)}_{0.15} \underbrace{\sum_{h_3} P(\neg H_4|h_3)P(h_3|e_3)}_{=0.15\cdot 0.41+0.7\cdot 0.59=0.4745} \right\rangle \\
&= \alpha \langle 0.3153, 0.071175\rangle \approx \langle 0.82, 0.18\rangle
\end{aligned}$$

Hence the probability he's happy today is 82%.

2. **Prediction:** Using the numbers from the previous exercise:

$$\begin{aligned}
\langle P(h_5|e_4)\rangle &= \left\langle \sum_{h_4} P(h_5|h_4)P(h_4|e_4) \right\rangle \\
&= \left\langle \underbrace{\sum_{h_4} P(H_5|h_4)P(h_4|e_4)}_{=0.85\cdot 0.82+0.3\cdot 0.18=0.751}, 1 - P(H_5|e_4) \right\rangle \\
&\approx \langle 0.75, 0.25\rangle \\
\langle P(h_6|e_4)\rangle &= \left\langle \sum_{h_5} P(h_6|h_5)P(h_5|e_4) \right\rangle \\
&= \left\langle \underbrace{\sum_{h_5} P(H_6|h_5)P(h_5|e_4)}_{=0.85\cdot 0.75+0.3\cdot 0.25=0.7125}, 1 - P(H_6|e_4) \right\rangle \\
&\approx \langle 0.71, 0.29\rangle
\end{aligned}$$

3. **Smoothing:** Using the numbers from Exercise 1:

$$
\begin{aligned}
\langle P(J_4|h_3)\rangle &= \left\langle \sum_{h_4} P(J_4|h_4)P(h_4|h_3) \right\rangle \\
&= \left\langle \underbrace{\sum_{h_4} P(J_4|h_4)P(h_4|H_3)}_{=0.6\cdot 0.85+0.15\cdot 0.15=0.5325}, \underbrace{\sum_{h_4} P(J_4|h_4)P(h_4|\neg H_3)}_{=0.6\cdot 0.3+0.15\cdot 0.7=0.285} \right\rangle \\
&\approx \langle 0.53, 0.29\rangle \\
\langle P(h_3|e_4)\rangle &= \alpha \langle P(h_3|e_3) \cdot P(J_4|h_3)\rangle \\
&= \alpha \left\langle \underbrace{P(H_3|e_3)}_{0.41} \cdot \underbrace{P(J_4|H_3)}_{0.53}, \underbrace{P(\neg H_3|e_3)}_{0.59} \cdot \underbrace{P(J_4|\neg H_3)}_{0.29} \right\rangle \\
&= \alpha\langle 0.2173, 0.1711\rangle \approx \langle 0.56, 0.44\rangle \\
\langle P(\neg J_3, J_4|h_2)\rangle &= \left\langle \sum_{h_3} P(\neg J_3|h_3)P(J_4|h_3)P(h_3|h_2) \right\rangle \\
&= \left\langle \underbrace{\sum_{h_3} P(\neg J_3|h_3)P(J_4|h_3)P(h_3|H_2)}_{=0.4\cdot 0.53\cdot 0.85+0.85\cdot 0.29\cdot 0.15=0.217175}, \underbrace{\sum_{h_3} P(\neg J_3|h_3)P(J_4|h_3)P(h_3|\neg H_2)}_{=0.4\cdot 0.53\cdot 0.3+0.85\cdot 0.29\cdot 0.7=0.23615} \right\rangle \\
&\approx \langle 0.22, 0.24\rangle \\
\langle P(h_2|e_4)\rangle &= \alpha \langle P(h_2|e_2) \cdot P(\neg J_3, J_4|h_2)\rangle \\
&= \alpha \left\langle \underbrace{P(H_2|e_2)}_{0.53} \underbrace{P(\neg J_3, J_4|H_2)}_{0.22}, \underbrace{P(\neg H_2|e_2)}_{0.47} \underbrace{P(\neg J_3, J_4|\neg H_2)}_{0.24} \right\rangle \\
&= \alpha\langle 0.1166, 0.1128\rangle \approx \langle 0.51, 0.49\rangle
\end{aligned}
$$

---

7 Assignment 7 (Complex Decisions and Value Iteration) – Given June 3., Due June 10.

Problem 7.1 (Markov Games)

We want to apply Markov Decision Procedures to two-player games as considered in KI1. We assume two players A, B and model everything from A's point of view - in particular, we have a reward function $R(s)$ for our states and our *states* are *only those game states, where it's A's move*. Hence, the transition function $T(s, a, s')$ models the probability that, if player A does action a in state s, player B will play such that we end up in state s'. 100pt

Let $N(s, a)$ be the set of possible successor states of s after picking move a.

10 pt

1. Let $U(s)$ be the utility of state s for player A. Write down the Bellman equation defining U, assuming player B always moves randomly (and uniformly distributed) and such that the equation does *not* contain the transition function T anymore (i.e. use $N(s, a)$ instead).

   20 pt

2. Now assume that your opponent picks a move with probability inversely proportional to our utility of the resulting state – e.g. if state s_1 has utility 20 and state s_2 has utility 10 (and $s_1, s_2 \in N(s, a)$), then after we pick move a in state s, player B will be twice as likely to pick s_2 than s_1. Give the corresponding Bellman equation. You may assume all utilities are positive.

   20 pt

3. Give the Bellman equation with a transition function, reward function and discount factor such that the resulting policy is equivalent to the one resulting from the Minimax algorithm from last semester.

   10 pt

4. Consider a game with four fields F_1, F_2, F_3, F_4. Player A starts in F_1, player B in F_4. Each turn, a player has to move to an adjacent field (i.e. if the player is in F_i, he can move to F_{i+1} or F_{i-1}). If an adjacent field contains the other player, one may jump over him. E.g. assume player A is in F_2 and player B in F_3, then player A may jump over B to end up in F_4.

   The game ends when one player reaches the opposite end of the board, i.e. A wins if he ends up in F_4, B wins if he ends up in F_1. player A starts. The reward for player A winning is 100, the reward for losing is 0.

   Draw the state space, showing the moves by A and B as arrows in different colors/styles.

   You will find it helpful to arrange the states (s_A, s_B) on a two-dimensional grid, using s_A and s_B as coordinates.

   40 pt

5. Apply value iteration (starting with utility 50 everywhere) assuming an unkown discount factor $0 < \gamma < 1$ to derive a strategy for player A, using the Bellman equation from the second exercise (you can disregard impossible states, of course). If you can not solve the second exercise, use the first Bellman equation instead.

Give the resulting strategy (as a sequence of actions) and how this derives from the computed utility function.

Note that in the lecture we alway assume the transition function $T(s, a, s')$ to be fixed, whereas for this exercise, the transition function itself depends on the utilities in the Bellman equations for Exercises 2 and 3, and hence will change during value iteration as well!

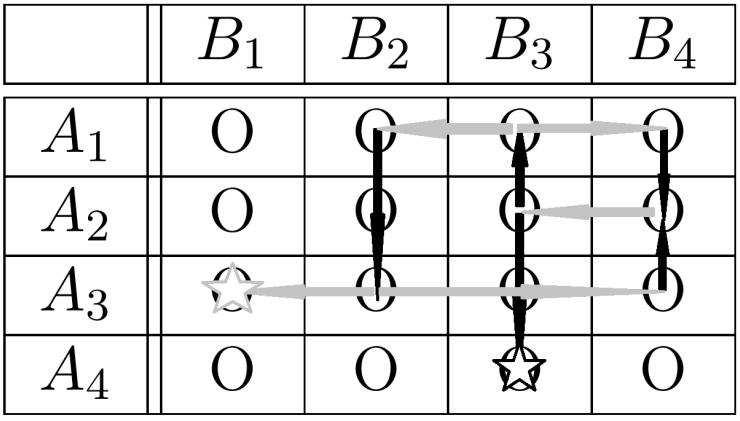
Solution:

1. $$U(s) = R(s) + \gamma \cdot \max_a \left(\sum_{s' \in N(s,a)} \frac{U(s')}{\mid N(s,a) \mid} \right)$$

2. $$U(s) = R(s) + \gamma \cdot \max_a \left(\sum_{s' \in N(s,a)} U(s') \left(1 - \frac{U(s')}{\sum_{s' \in N(s,a)} U(s')} \right) \right)$$

3. $$U(s) = R(s) + 1 \cdot \max_a \left(\min_{s' \in N(s,a)} U(s') \right)$$

   i.e $T(s, a, s') = 1$ iff $s' = \arg\min_{s'' \in N(s,a)} U(s'')$ and $= 0$ otherwise. The reward function is $= 0$ everywhere except at terminal nodes.

4. 

5. We initialize with $U(s) = 50$ for all states. We will call the actions u and d for up or down in the above table.

   - Initially, the only utilities that change are the ones for the terminal nodes A_4B_3 and A_3B_1, namely:
   $$U'(A_4B_3) = 100 \qquad U'(A_3B_1) = 0$$

- Now, only those utilities change that can lead to terminal nodes; namely:

$$U'(A_2B_3) = 0 + \gamma \max \left\{ \underbrace{\underbrace{U(A_1B_2)}_{=50}\frac{1}{2} + \underbrace{U(A_1B_4)}_{=50}\frac{1}{2}}_{u}, \underbrace{\underbrace{U(A_4B_3)}_{=100}}_{d} \right\} = 100\gamma$$

$$U'(A_1B_2) = 0 + \gamma \left(\underbrace{U(A_3B_1)}_{=0} \cdot 1 \right) = 0$$

- Next, all states that can directly lead to either of the previously changed states change. In particular, A_2B_3 can lead to A_1B_2, however, in the above update equation we can already see that the maximum of the two actions u, d does not change and remains maximal, so we do not need to update. Hence, the only utility that changes is $U(A_1B_4)$:

$$U'(A_1B_4) = 0 + \gamma \cdot \left(\underbrace{U(A_2B_3)}_{=100\gamma} \cdot 1 \right) = 100\gamma^2$$

- A_2B_3 can lead to A_1B_4 which we just changed, so we would need to update that - however, as before, the maximum of the two actions does not change and remains maximal, so we are finished.

The resulting policy is: (d, d) - the first d being the only possible move, and the latter d having expected utility $1 \cdot 100$, whereas u has expected utility $1 \cdot U(A_1B_2) + 0 \cdot U(A_1B_4) = 0$.

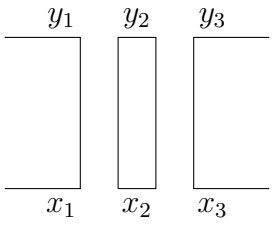
---

8 Assignment 8 (Partially Observable Markov Decision Problems) – Given June 10., Due June 17.

Problem 8.1 (POMDPs)

100pt

You're controlling a video game character trapped in a storage facility with an enemy agent. Both of you are armed, but you only have one bullet left. You're covering behind a thick concrete wall (at x_1), while your enemy is hiding at y_1, y_2 or y_3.



At every (realistically reachable) position x_1, x_2, x_3, you have three kinds of actions available:

- You can peek (**P**) out from behind your cover (to the left or right of your position), risking getting shot at but possibly seeing your enemy.
- You can (more or less blindly) fire (**F**) your gun (to the left or right of your position), losing your bullet but possibly killing your enemy.
- You can run (**R**) to the next wall (to the left or right of your position) for cover, risking getting shot at but changing your location.

That gives you six actions: $P_\ell, P_r, F_\ell, F_r, R_\ell$ and R_r.

At location x_1, we assume $P_\ell = P_r, F_\ell = F_r$ and $R_\ell = R_r$; analogously at x_3.

Your enemy has the same actions available, but luckily is a much worse shot than you. If you shoot in his direction while he peeks or runs past your line of sight, you'll kill him (reward: $+100$). If he shoots in your direction while you peek or run past his line of sight, he'll only kill you (reward: -100) with 50% probability. If you have no bullet left but your enemy is still alive, he'll surely kill you (reward: -100). If you both shoot, there's a 75% chance you kill him first, else you get killed.

Example: Say, you're at x_2 and fire to the left. If your enemy in the same "round" runs from y_1 to y_2 or from y_2 to y_1, he gets shot. If he's at y_1 and peeks to the right, he gets shot. If he's at y_2 and peeks to the left, he gets shot. Similarly, if you run from x_1 to x_2 while your enemy sits at y_1 and shoots to the right, you risk getting killed with 50% probability.

In general, your enemy's behavior is as follows:

P_ℓ	20%	P_r	20%
F_ℓ	10%	F_r	10%
R_ℓ	20%	R_r	20%

Every non-terminal state has reward -1. 20 pt

1. Give the full state space and write down the Bellman equation specifically for the initial state, assuming your enemy is in position y_1.

Let's now add uncertainty. We start at x_1, but the enemy can be at any of the y_i. When peeking, you can sometimes see a shadow, indicating that the enemy is behind either of the two walls adjacent to the corridor you're observing. To be precise: if you are at x_2 and peek to the left, then, if the enemy is either at y_1 or y_2, you will see a shadow with probability 80%. 10 pt

2. Give the initial belief state for this situation. 20 pt
3. Your first action is to run to the center, i.e. you execute R_r. Give the new belief state after doing so, and after observing that you're not dead. 30 pt
4. Your next action is to peek to the right. You observe a shadow (and that you're not dead). Give the new belief state before and after this observation. 20 pt
5. What's the expected utility of F_r in your current belief state? (Keep in mind that you'll get killed if you waste your only bullet!)

Solution:

1. $\{(x_1, y_1), (x_1, y_2), (x_1, y_3), (x_2, y_1), (x_2, y_2), (x_2, y_3), (x_3, y_1), (x_3, y_2), (x_3, y_3), \texttt{win}, \texttt{lose}\}$
2. $$U((x_1,y_1)) = -1 + \max \left\{ \underbrace{0.4 \cdot \underbrace{U((x_1,y_1))}_{P_r} + 0.2 \cdot \underbrace{((0.5 \cdot -100) + (0.5 \cdot U((x_1,y_1))))}_{F_r} + 0.4 \cdot \underbrace{U((x_1,y_2))}_{R_r}}_{P_r}, \right.$$
$$\underbrace{0.4 \cdot \underbrace{100}_{P_r} + 0.2 \cdot \underbrace{(0.75 \cdot 100 + 0.25 \cdot -100)}_{F_r} + 0.4 \cdot \underbrace{100}_{R_r}}_{F_r},$$
$$\left. \underbrace{0.4 \cdot \underbrace{U((x_2,y_1))}_{P_r} + 0.2 \cdot \underbrace{((0.5 \cdot -100) + (0.5 \cdot U((x_2,y_1))))}_{F_r} + 0.4 \cdot \underbrace{U((x_2,y_2))}_{R_r}}_{R_r} \right\}$$

$$= -1 + \max\{0.5U((x_1, y_1)) - 10 + 0.4U((x_1, x_2)), 90, 0.5U((x_2, y_1)) - 10 + 0.4((x_2, y_2))\}$$

3. $\langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, 0, 0, 0, 0, 0, 0, 0 \rangle$

4.

$$\begin{aligned}
P((x_2, y_1)) &= \frac{1}{3}T((x_1, y_1), R_r, (x_2, y_1)) + \frac{1}{3}T((x_1, y_2), R_r, (x_2, y_1)) + \frac{1}{3}T((x_1, y_3), R_r, (x_2, y_1)) \\
&= \frac{1}{3} \cdot (0.4 + \frac{1}{2} \cdot 0.2) + \frac{1}{3} \cdot 0.2 + \frac{1}{3} \cdot 0 = 0.23 \\
P((x_2, y_2)) &= \frac{1}{3}T((x_1, y_1), R_r, (x_2, y_2)) + \frac{1}{3}T((x_1, y_2), R_r, (x_2, y_2)) + \frac{1}{3}T((x_1, y_3), R_r, (x_2, y_2)) \\
&= \frac{1}{3} \cdot 0.4 + \frac{1}{3} \cdot (0.4 + \frac{1}{2} \cdot 0.1 + 0.1) + \frac{1}{3} \cdot 0.4 = 0.45 \\
P((x_2, y_3)) &= \frac{1}{3}T((x_1, y_1), R_r, (x_2, y_3)) + \frac{1}{3}T((x_1, y_2), R_r, (x_2, y_3)) + \frac{1}{3}T((x_1, y_3), R_r, (x_2, y_3)) \\
&= \frac{1}{3} \cdot 0 + \frac{1}{3} \cdot 0.2 + \frac{1}{3} \cdot (0.4 + 0.2) = 0.26 \\
P(\texttt{lose}) &= 0.05
\end{aligned}$$

We normalize (since we're not dead):

$$\langle 0, 0, 0, 0.25, 0.47, 0.28, 0, 0, 0, 0, 0 \rangle$$

5.

$$\begin{aligned}
P((x_2, y_1)) &= 0.25T((x_2, y_1), P_r, (x_2, y_1)) + 0.47T((x_2, y_2), P_r, (x_2, y_1)) + 0.28T((x_2, y_3), P_r, (x_2, y_1)) \\
&= 0.25 \cdot (0.4 + 0.2) + 0.47 \cdot 0.2 + 0.28 \cdot 0 = 0.244 \\
P((x_2, y_2)) &= 0.25T((x_2, y_1), P_r, (x_2, y_2)) + 0.47T((x_2, y_2), P_r, (x_2, y_2)) + 0.28T((x_2, y_3), P_r, (x_2, y_2)) \\
&= 0.25 \cdot 0.4 + 0.47 \cdot (0.4 + 0.1 + \frac{1}{2} \cdot 0.1) + 0.28 \cdot 0.4 = 0.4705 \\
P((x_2, y_3)) &= 0.25T((x_2, y_1), P_r, (x_2, y_3)) + 0.47T((x_2, y_2), P_r, (x_2, y_3)) + 0.28T((x_2, y_3), P_r, (x_2, y_3)) \\
&= 0.25 \cdot 0 + 0.47 \cdot 0.2 + 0.28 \cdot (0.4 + \frac{1}{2} \cdot 0.2) = 0.234 \\
P(\texttt{lose}) &= 0.0515
\end{aligned}$$

Now we filter given the new evidence:

$$\begin{aligned}
P(s|e) &= P(e|s) \cdot P(s) \\
P((x_2, y_1)|e) &= \alpha \cdot 0 \cdot 0.244 = 0 \\
P((x_2, y_2)|e) &= \alpha \cdot 0.8 \cdot 0.4705 = \alpha \cdot 0.3764 \\
P((x_2, y_3)|e) &= \alpha \cdot 0.8 \cdot 0.234 = \alpha \cdot 0.1872
\end{aligned}$$

Hence:

$$\langle 0, 0, 0, 0, 0.67, 0.33, 0, 0, 0, 0, 0 \rangle$$

6.

$$
\begin{aligned}
EU(F_r) &= \sum_{s,s'} P(s) \cdot T(s, F_r, s') \cdot U(s') \\
&= \sum_{s'} 0.67 \cdot T((x_2, y_2), F_r, s') U(s') + 0.33 \cdot T((x_2, y_3), F_r, s') U(s') \\
&= 0.67 \left(0.2 \cdot -100 + 0.2 \cdot -100 + 0.1 \cdot -100 + 0.1 \cdot (0.25 \cdot -100 + 0.75 \cdot 100) + 0.4 \cdot 100\right) \\
&+ 0.33 \left(0.8 \cdot 100 + 0.2 \cdot (0.25 \cdot -100 + 0.75 \cdot 100)\right)
\end{aligned}
$$

---

9 Assignment 9 (Decision Tree Learning) – Given June 17., Due June 24.

Problem 9.1 (Decision Tree Learning in Java)
Consider the following decisions on whether or not to go play tennis. The target is "PlayTennis". 100pt

Outlook	Temperature	Humidity	Wind	PlayTennis
Sunny	Hot	High	Weak	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Strong	No
Rain	Mild	Normal	Weak	Yes

1. Compute by hand (!) the information gain for each attribute, and consequently at which attribute the "correct" decision tree will split first. 20 pt

2. Implement decision tree learning in java or scala: 80 pt

Use the attached java (or scala) classes. Those are:

- DecisionTable: The abstract class you're supposed to extend/implement. Takes as constructor argument a list of TableRows and returns a Tree on calling computeTree(Attribute attr) – intended to be the decision tree for the attribute attr.

  Also has an abstract interface for the method entropyGain for a specific attribute, which you're supposed to implement and can use in your implementation of computeTree

- TableRow: A row in a decision table; consisting of a name : String and a list of AttVals, being attribute-value-pairs.

- Attribute: An attribute; consisting of a name : String and a list of allowed values (as Strings).

- Tree: A common superclass of Leaf, which wraps around a single value of the queried attribute, and SubTree, which is the subtree that splits at an attribute A along an Evaluation, being a pair of a value of A and the corresponding Tree for that value.

- Converter.scala contains an extension of DecisionTree which implements implicit conversions between java and scala classes, and a Test object using the scala interfaces. Use this if you want to use scala for this exercise. If you use java, you can completely ignore/delete this file.

You can test your network with the Test class, which contains an implementation of the above decision table.

Hand in your solution as a java or scala file containing a class extending DecisionTable (or ScalaDecisionTable). Please

1. name your class <lastname>_<firstname>_<mat.nr> to be guaranteedly unique :)
2. put your class in the package info.kwarc.teaching.AI.DecisionTrees.SS18

Solution:

10 Assignment 10 (Neural Networks) – Given June 24., Due July 01.

Problem 10.1 (Neural Networks)

In this exercise we implement a very basic, but easy to extend multi-layer neural network with several hidden layers, and train it on some simple datasets. 100pt

All methods you are required to implement are in java-Files in the `Your_Name_Here` subfolder/namespace - please substitute in your own name and hand in only that folder.

We start with the easy parts of neural networks. Note that many of the following exercises are more or less trivial and serve mainly to guide you through the relevant methods and "components" of a neural network architecture. We start with the simple methods:

1. The first step is to implement at least one activation function – we will use `tanh`. Implement `tanh` and $\texttt{tanh}'$ in the file `TanhActivation.java`. 5 pt
2. Next, we implement the two `BiasFiller`s and `WeightFiller`s that initialize a neural network, `ConstantBias` and `RandomWeight`. 10 pt
3. Next, we implement a loss function for the output layer; specifically a `EuclideanLoss` (also called mean-square-error loss function). 10 pt
4. It is often a good idea to use a linear activation function for the last layer in a neural network: Hence, implement `LinearActivation`. 5 pt

So much for the easy parts. A `Network` now takes a `LearningRate` and holds several `Layer`s that can be added via `Network.add`. The first layer being an `InputLayer`, followed by arbitrarily many `FullyConnected` layers and finally an `OutputLayer`.

5. Implement the `forward` function for each of those layer classes, that takes an input vector, processes it according to the activation function, weights and biases, and returns the result. 20 pt
6. Implement the `backward` function that calculates the gradient/delta per neuron and returns it. 30 pt
7. Finally, implement the `updateWeightsAndBias` methods that are used for training. 20 pt

Your neural network is now ready to do some fancy stuff. You can use the methods in `MyTestcases.java` to test the individual features of your neural net and train/test it on simple generated XOR- and AND-datasets with two inputs and one output.