

Last Name:

First Name:

Matriculation Number:

**Exam  
Artificial Intelligence 2**

2024-10-29

	To be used for grading, do not write here											
prob.	1.1	1.2	2.1	2.2	3.1	3.2	4.1	4.2	4.3	5.1	Sum	grade
total	8	10	9	10	10	9	9	12	7	7	91	
reached												

In the course Artificial Intelligence I/II we award bonus points for the first student who reports a factual error in an old exam. (Please report spelling/formatting errors as well.)

# 1 Probabilities

## Problem 1.1 (Python)

1. Consider the Python program below.

4 pt

```
def foo(J, x):
    px = sum(J[x])
    res = []
    for y in range(len(J[x])):
        res.append(J[x][y]/px)
    return res
```

The input  $J$  represents the joint probability distribution of random variables  $X$  with domain  $\{0, \dots, m-1\}$  and  $Y$  with domain  $\{0, \dots, n-1\}$  in such a way that  $J[x][y] = P(X = x, Y = y)$ . The input  $x$  represents an element in the domain of  $X$ .

Which probability-related operation does the function `foo` compute?

---

*Solution:* The conditional probability distribution  $P(Y = y \mid X = x)$ .

---

2. Assume a random variable  $X$  with domain  $\{0, \dots, m-1\}$  and distribution  $D$ , i.e.,  $D[x] = P(X = x)$ . Assume that  $D[x]$  is non-zero for some even value of  $x$ .

4 pt

Complete the definition of  $p$  in the program below in such a way that  $p(D, x)$  returns the conditional probability  $P(X = x \mid X \text{ is even})$ , i.e., the probability of  $X = x$  under the condition that  $X$  takes an even value.

```
def even(x):
    return x % 2 == 0

def p(D, x):
```

---

*Solution:*

```
def p(D, x):
    if even(x):
        sumEven = 0
        for i in range(len(D)):
            if even(i):
                sumEven += D[i]
        return D[x]/sumEven
    else:
        return 0
```

---

## Problem 1.2 (Calculations)

Assume random variables  $X, Y$  both with domain  $\{0, 1, 2\}$ . For some outcomes  $A$ , the probabilities are known as follows:

$A$	$P(A)$
$X = 0 \wedge Y = 0$	$a$
$X = 0 \wedge Y \neq 0$	$b$
$X \neq 0 \wedge Y = 0$	$c$
$X \neq 0 \wedge Y \neq 0$	$d$

1. Give all subsets of the probabilities  $\{a, b, c, d\}$  that must sum to 1. 2 pt

---

*Solution:* Only  $\{a, b, c, d\}$

---

2. In terms of  $a, b, c, d$ , give  $P(X \neq 0)$  or argue why there is not enough information to compute the value. 2 pt

---

*Solution:*  $c + d$

---

3. In terms of  $a, b, c, d$ , give  $P(X > Y)$  or argue why there is not enough information to compute the value. 2 pt

---

*Solution:* Not enough information. The result is  $c + P(X = 2, Y = 1)$ , but we cannot compute the latter because there is no data that distinguishes between the values 1 and 2 for  $X$  or  $Y$ .

---

4. In terms of  $a, b, c, d$ , give  $P(X = 0 \mid Y = 0)$  or argue why there is not enough information to compute the value. 2 pt

---

*Solution:*  $a/(a + c)$

---

5. Now assume we additionally know  $e = P(X = Y)$ . In terms of  $a, b, c, d, e$ , give  $P(X + Y = 3)$ . 2 pt

---

*Solution:*  $d - e + a$

---

## 2 Bayesian Reasoning

### Problem 2.1 (Basic Rules)

Assume you are trying to relate economic development and your business results. You have collected the following data:

- The economy does well 40% of the time and badly otherwise.
- If your business does well, the economy does well 50% of the time.
- If the economy does well, your business does well 70% of the time.

You model the problem using two Boolean random variables  $E$  (economy does well) and  $B$  (business does well). You also abbreviate the events  $E = \text{true}$  and  $B = \text{true}$  as  $e$  and  $b$ .

1. By filling in the gaps below, state for each number in the text above, which probability it describes. 2 pt

---

1.  $P(\quad) = 0.4$

2.  $P(\quad) = 0.5$

3.  $P(\quad) = 0.7$

---

*Solution:*

1.  $P(e) = 0.4$

2.  $P(e | b) = 0.5$

3.  $P(b | e) = 0.7$

---

2. Calculate the **probability** that your business does well. 2 pt

---

*Solution:* We have  $P(e | b)P(b) = P(b | e) \cdot P(e)$ . From that, we can calculate  $P(b) = 0.7 \cdot 0.4 / 0.5 = 0.56$

---

3. If your business does well, you estimate your earnings to be \$100/month, otherwise \$60/month. 2 pt  
Calculate your **expected utility** (in \$/month) if the economy does well.

---

*Solution:*  $0.7 \cdot 100 + 0.3 \cdot 60 = 88$

---

4. Calculate the **probability** that your business and the economy are doing badly at the same time. 3 pt

---

*Solution:* We need  $P(\neg b, \neg e)$ . We **know** that

- $P(b, e) = P(b | e) \cdot P(e) = 0.28$
- $P(e) = P(b, e) + P(\neg b, e) = 0.4$ , i.e.,  $P(\neg b, e) = 0.12$
- $P(b) = P(b, e) + P(b, \neg e) = 0.56$  (from previous question), i.e.  $P(b, \neg e) = 0.28$
- $P(b, e) + P(\neg b, e) + P(b, \neg e) + P(\neg b, \neg e) = 1$ , i.e.,  $P(\neg b, \neg e) = 0.32$

If the previous question was solved wrongly as  $w$ , the result should be  $0.88 - w$ .

---

### Problem 2.2 (Bayesian Networks)

Consider the following situation about a day out.

- If the sun shines, you are out long, and you did not use sunscreen, you may get sunburned.
- If the sun shines or you meet friends, you may stay out long.

You want to model this situation as a **Bayesian network** using **Boolean random variables**  $S$  (sunshine),  $O$  (staying out long),  $B$  (sunburn),  $U$  (sunscreen used), and  $F$  (meeting friends).

1. Using a good **variable ordering**, model this as a **Bayesian network**. 2 pt

---

*Solution:* Order: e.g.,  $FSOUB$ . Network: **edges** from  $F, S$  to  $O$ , and from  $O, S, U$  to  $B$ .

---

2. Assume you meet your friends and use sunscreen, and you want to determine the **probability** of sunburn. What are the **query/evidence/hidden variable**? 2 pt

---

*Solution:* **query:**  $B$ , **evidence:**  $F, U$ , **hidden:**  $S, O$

---

**For the remaining questions**, assume your **network** is  $F \leftarrow S \rightarrow O \leftarrow B \rightarrow U$  (which may or may not be a correct solution to the previous question).

3. Which **probabilities** are stored in the **conditional probability table** of the **node**  $O$ ? Which of those could be omitted and **computed** from the others? 3 pt

---

*Solution:*  $P(O = o \mid S = s, B = b)$  where  $o, s, b$  are **Booleans**, i.e., 8 entries in total. The entries for  $O = false$  can be **computed** from the ones for  $O = true$ .

---

4. Give the **probability distribution**  $P(S \mid O = o)$  (for fixed  $o$ ) in terms of the entries of the **probability tables** of the **network**. 3 pt

---

*Solution:*

$$P(S \mid O = o, B = b) = \alpha(P(S, O = o, B = true) + P(S, O = o, B = false))$$

where for each **Boolean**  $s$

$$P(S = s, O = o, B = b) = P(S = s) \cdot P(B = b) \cdot P(O = o \mid S = s, B = b)$$


---

### 3 Markovian Reasoning

#### Problem 3.1 (Hidden Markov Models)

Consider the following situation:

- Each year the groundwater level at your location is high or not.
- Each year your harvests are good or not.
- You want to **study** the groundwater level by **observing** the harvests.

You choose to model this situation as a **stationary first-order hidden Markov model** with a **stationary sensor model** with **Markov property**, using two families of year-indexed **Boolean random variables**.

1. Give the **state** and **evidence variable** and their **domains**. 2 pt

---

*Solution:* **State:**  $G_a$  (groundwater level), **evidence:**  $H_a$  (harvests), all **domains** are the **Booleans**

---

2. Which **probabilities** do you need for this model? 3 pt

---

*Solution:* A **transition model**  $T_{ij} = P(G_{a+1} = j \mid G_a = i)$  and a **sensor model**  $S_{ij} = P(H_a = j \mid G_a = i)$ . (Because all **variables** are **Boolean**, half of these entries are redundant.)

---

3. What would change about the **answer** to the previous **question** if the **HMM** were not stationary? 2 pt

---

*Solution:* We would need a different  $T$  for every year.

---

4. Assume a **sequence**  $e_1, e_2, \dots$  of Booleans such that  $e_a$  gives the quality of the harvest in year  $a = 1, 2, \dots$ . Then the **filtering algorithm** can be **written** in **recursive matrix** form as 3 pt

$$\mathbf{f}_{1:a} = \alpha(O_a T^t \mathbf{f}_{1:a-1})$$

(where  $T^t$  is the **transposed transition matrix**).

Which **probability distributions** are **represented** by the  $\mathbf{f}_{1:a}$ , and how do we obtain the **values** of the  $O_a$ ?

---

*Solution:*  $\mathbf{f}_{1:a} = P(G_a \mid H_1 = e_1, \dots, H_{a-1} = e_{a-1})$

$O_a$  is the **diagonal matrix** whose **diagonal** contains the **values**  $P(H_a = e_a \mid G_a)$ .

---

### Problem 3.2 (Decision Processes and Utility)

Consider an **agent** moving along a rectangular grid of  $3 \times 3$  locations.

The **agent** can stand still, or move 1 location up, down, left, or right except when already at the **edge**.

A movement step succeeds with **probability** 90%. In the remaining cases, the **agent** does not move.

The **agent** is initially in the bottom-left corner location. Its **goal** is to move to the top-right location.

1. Model this situation as a **Markov Decision Process**  $\langle S, A, T, s_0, R \rangle$ . Use a **reward function** that uses a **constant reward** for **non-goal states**. 5 pt

---

*Solution:* One possible model is

- $S = \{0, 1, 2\} \times \{0, 1, 2\}$
  - Let  $M = \{(0,0), (0,1), (0,-1), (1,0), (-1,0)\}$ . Then  $A(s) = \{m \in M \mid s + m \in S\}$  where + is component-wise **addition**.
  - The **transition model** is given by
    - $a = (0,0)$ :  $T(s, a)(s) = 1$
    - otherwise:  $T(s, a)(s + a) = 0.9$ ,  $T(s, a)(s) = 0.1$
    - 0 for all other **probabilities**
  - $s_0 = (0, 0)$
  - A typical choice is any **function**  $R$  that is high for the **goal** and slightly **negative** for other **states**. E.g.,  $R((2,2)) = 1$  and  $R(s) = -0.1$  otherwise.
- 

2. Give an **optimal policy**  $\pi^*$ . 2 pt

---

*Solution:* Any **policy** is **optimal** that moves only up and right until at the goal and then stands

still. E.g.:  $\pi^*$  maps  $(2,2) \mapsto (0,0)$ ,  $(i,j) \mapsto (1,0)$  if  $i \neq 2$ , and  $(2,j) \mapsto (0,1)$  otherwise

3. Now assume the agent is unable to tell whether an action resulted in movement. Explain (in about 2 sentences) how we can still represent this situation as an Markov decision process. 2 pt

*Solution:* We define a new Markov decision process whose states are belief states, i.e., probability distribution over states  $s \in S$ . All actions are possible in every belief state, and cause transition between belief states. The reward of a belief state can be calculated as a probability-weighted of states.

## 4 Learning

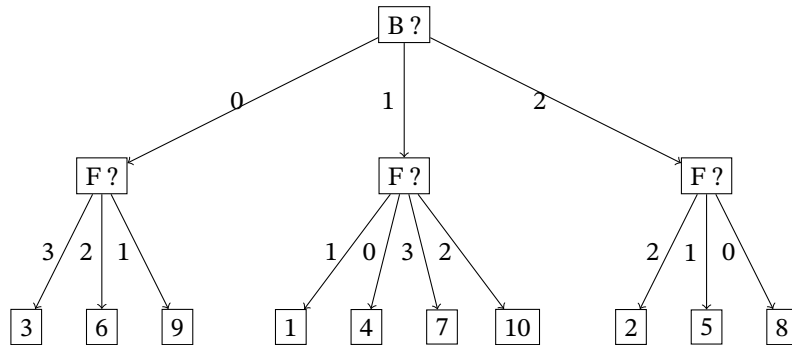
### Problem 4.1 (Decision Trees and Lists)

Consider an unknown natural number  $N \in \{1, \dots, 10\}$ . You are allowed to ask the following attributes/questions about  $N$ :

- A Is  $N$  prime? (possibly answers: yes, no)
- B What is  $N$  modulo 3? (possible answers: 0, 1, 2)
- C Is  $N > 5$ ? (possibly answers: yes, no)
- D Is  $N$  a root of  $X^2 + X - 7$ ? (possibly answers: yes, no)
- E What is the result of  $\sin((N + 0.5)\pi)$ ? (possibly answers:  $-1, 1$ )
- F What is  $N$  modulo 4? (possible answers: 0, 1, 2, 3)

1. Give the shortest (in terms of tree depth) decision tree for identifying  $N$  that uses at most the above questions. 3 pt

*Solution:* The tree must use B and F in either order. One possible tree is



Those have  $3 \times 4$  possible answer combinations giving us a chance (but no guarantee) to disambiguate 10 options. Giving a concrete tree of depth 2 shows that we can indeed do it. No single question (and no other pair of questions) has  $\geq 10$  answer combinations, showing 2 is a lower bound.

2. We can see each question as a **random variable**. Give the entropies  $I(A)$ ,  $I(C)$ , and  $I(D)$ , i.e., the number of **bits** obtained by asking the question. (Simplify as much as possible without introducing approximate values.) 3 pt

---

*Solution:*  $A: I(\langle 2/5, 3/5 \rangle) = -2/5 \log_2(2/5) - 3/5 \log_2(3/5)$

$C: 1$

$D: 0$

Counting 1 as a **prime number** is wrong but was accepted anyway. In that case  $I(A) = 1$ .

---

3. Now assume our goal is to learn the **function** that computes whether any **natural number** is a **square number**, given the answers to the questions A-F, using  $N = 1$  and  $N = 2$  for training data. Formally state this situation as an **inductive learning problem**  $\langle \mathcal{H}, T \rangle$ . 3 pt

---

*Solution:*  $\mathcal{H}$  is the **set of functions**  $\mathbb{B} \times \{0, 1, 2\} \times \mathbb{B} \times \mathbb{B} \times \{-1, 1\} \times \{0, 1, 2, 3\} \rightarrow \mathbb{B}$  where  $\mathbb{B} = \{\text{yes}, \text{no}\}$ .  $T$  is the **set** containing  $(\text{no}, 1, \text{no}, \text{no}, -1, 1, \text{yes})$  (for  $N = 1$ ) and  $(\text{yes}, 2, \text{no}, \text{no}, 1, 2, \text{no})$  (for  $N = 2$ ).

Alternatively, any **subset** of the hypothesis space can be used.

---

#### Problem 4.2 (Classifiers)

Consider a set  $E$  of examples of the form  $(x, y)$  where  $x \in \mathbb{R}^2$  and  $y \in \{0, 1\}$ .

We want to learn a linear classifier  $h_w$  with a hard threshold. For the hard threshold, we use  $\mathcal{T}(u)$  that returns 1 if  $u > 0$  and 0 otherwise.

1. Give the general form of such a classifier. 2 pt

---

*Solution:*  $h_w(x) = \mathcal{T}(w \cdot x)$  where  $w \in \mathbb{R}^3$  and we embed  $x$  into  $\mathbb{R}^3$  by putting  $x_0 = 1$ . Alternatively, we can **write** this as  $h_w(x) = \mathcal{T}(w_1 \cdot x_1 + w_2 \cdot x_2 + w_0)$ .

---

2. Give a **neural network** that can be used to represent such a classifier. 3 pt

---

*Solution:* One input cell for each  $x_1, x_2$  and constant bias  $x_0 = 1$ . One **perceptron** output cell with input weight  $w_i$  for  $x_i$ .

---

3. Give the formula for the **squared error loss** of such a classifier. 2 pt

---

*Solution:*  $\sum_{(x,y) \in E} (y - h_w(x))^2$

---

4. Why is **gradient descent** not applicable to **minimize** the loss in this case? 1 pt

---

*Solution:*  $\mathcal{T}$  and thus the **loss function** are not **differentiable**.

---

5. Instead, we can use the **perceptron learning rule** to update the weights. What is that rule, and how do we apply it using the examples from  $E$ ? 2 pt

*Solution:* We start by putting  $w_i = 0$  for all  $i$ . Then we apply for each example  $(x, y) \in E$ , the rule

$$w_i \leftarrow w_i + \alpha(y - h_w(x))x_i \quad \text{for each } i$$

where parameter  $\alpha$  is the **learning rate**.

6. Now assume our classifiers are multi-layer **neural networks** and that **gradient descent** is applicable. Explain informally (in about 2 sentences) the basic idea of **back-propagation**. 2 pt

*Solution:* Applying **gradient descent** at the **output layer** yields updates for the connections from the last hidden layer. Aggregating these updates for each cell in the last hidden layer yields errors for the last hidden layer, which are used to **recursively** apply **gradient descent** for the preceding layer.

### Problem 4.3 (Inductive Learning)

Consider the family tree given by the following relations:

couple	children
A, B	E, F
C, D	G
F, G	H, I

Assume we already know the **predicate**  $\text{par}(x, y)$  for  $x$  being a parent of  $y$ . Our goal is to learn the **predicate**  $\text{gp}(x, y)$  for  $x$  being a grandparent of  $y$ , i.e., to find a formula  $D$  such that  $\forall x, y. \text{gp}(x, y) \Leftrightarrow D(x, y)$ .

We do not know  $D$ , but we have the following (counter-)examples for  $\text{gp}$ :

person-pair	grandparent
A, H	yes
B, I	yes
A, E	no
A, F	no
A, B	no
A, C	no

1. Give the intended formula  $D_1$ , i.e., the correct definition of grandparent. 2 pt

*Solution:*  $D_1(x, y) = \exists u. \text{par}(x, u) \wedge \text{par}(u, y)$

2. Give a formula  $D_2$  that is true exactly for the **positive examples**. 2 pt

*Solution:*  $D_2(x, y) = x = A \wedge y = H \vee x = B \wedge y = I$

3. Explain (in about 3 sentences) the commonalities and pros and cons of learning the formula  $D$  as  $D_1$  vs.  $D_2$ . 3 pt

---

*Solution:* Both correctly learn the positive and the **negative examples**.  $D_2$  is easy to learn, but it is of size  $\mathcal{O}(n)$  and **overfits** to the examples.  $D_1$  is small and captures the intended formula, but is much harder to learn.

---

## 5 Natural Language Processing

### Problem 5.1 (Information Retrieval)

1. Explain (in about 2 sentences) the connection between **tfidf** and **cosine similarity** to relate texts. 2 pt

---

*Solution:* **tfidf** is a way to represent a text as a vector, relative to a **corpus**, by measuring the frequency of terms in some way. **Cosine similarity** uses the **angle** between two such vectors as a measure of how similar the texts represented by the vectors are.

---

2. Explain (in about 2 sentences) the difference between **information retrieval** and **information extraction**. 2 pt

---

*Solution:* **Information retrieval** identifies documents that are likely to contain information that answers the query. **Information extraction** obtains **structured representations** of parts of the content of a document, which can then be used to answer queries precisely.

---

3. Explain (in about 3 sentences) what kind of function is learned in the continuous **bag of words** algorithm, as used e.g. in **Word2Vec**, and how the training proceeds. 3 pt

---

*Solution:* It learns a **word embedding**, i.e., a function from **words** to vectors. Going through a **corpus**, each **word** is processed together with the context, i.e., the  $n$  **words** before and after. The neural network is trained to predict the **word** from its context.

---