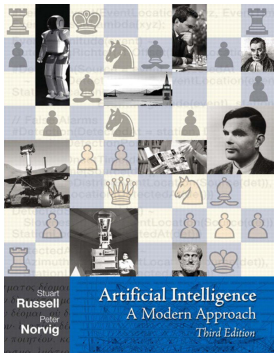
# TDT4171 Artificial Intelligence Methods

Exercise 4

March 3, 2020

- **Delivery deadline: March 11, 2020** by 23:59 sharp. No late delivery will be graded! Deadline extensions will only be considered for extraordinary situations such as family or health-related circumstances. However, these circumstances **must be documented**, e.g., with a doctor's note("legeerklæring"). Having a lot of work in other classes is not a legitimate excuse for late delivery. Also, happy events such as birthdays, weddings, and vacations are not valid excuses as they are typically known well in advance, so you will need to plan your work (and delivery) schedule around them.
- Required reading for this assignment: Chapter 16 and 17 (the parts in the curriculum).
- Deliver your solution on *Blackboard*. Please upload your report as a **PDF file**, and please **do not** put the pdf into an archive. For the programming part, deliver the source code alongside the pdf file. Again, please **do not** put it into an archive. Just upload the code alongside the pdf file. In other words, upload, **PDF file + source code**.
- Students can NOT work in groups. Each student can only submit solution individually.
- This homework counts for 4% of the final grade.
- Copying("koking") from other students is not accepted, and if detected, will lead to immediate failure of the course. The consequence will apply to both the source and the one cribbing.
- For help and questions related to the assignment, **ask the student assistants during the assignment hours**. The assignment hours can be found on the assignment page on Blackboard. If the student assistants are not able to help you or you have other types of inquiries, an email can be sent to Yan(yanzhe.bekkemoen@ntnu.no).
- All references in this text refer to this version of the book, *Artificial Intelligence: A Modern Approach, Third Edition*, "Blue version".



# 1 Problem 1: Utility

Risk-seeker is a person who has a preference for risk. Risk-averse is just the opposite of "seeking". Specifically, it means "having a strong dislike for something", in this case, risk. Assume Gabriel, Gustav, Maria, and Sonja are four people with the utility functions for apples as described in Figure 1. The x-axis is the number of apples, and the y-axis is the utility.

a) Classify these four persons as either risk-seeking, risk-neutral, or risk-averse. Explain your reasoning for the classification.

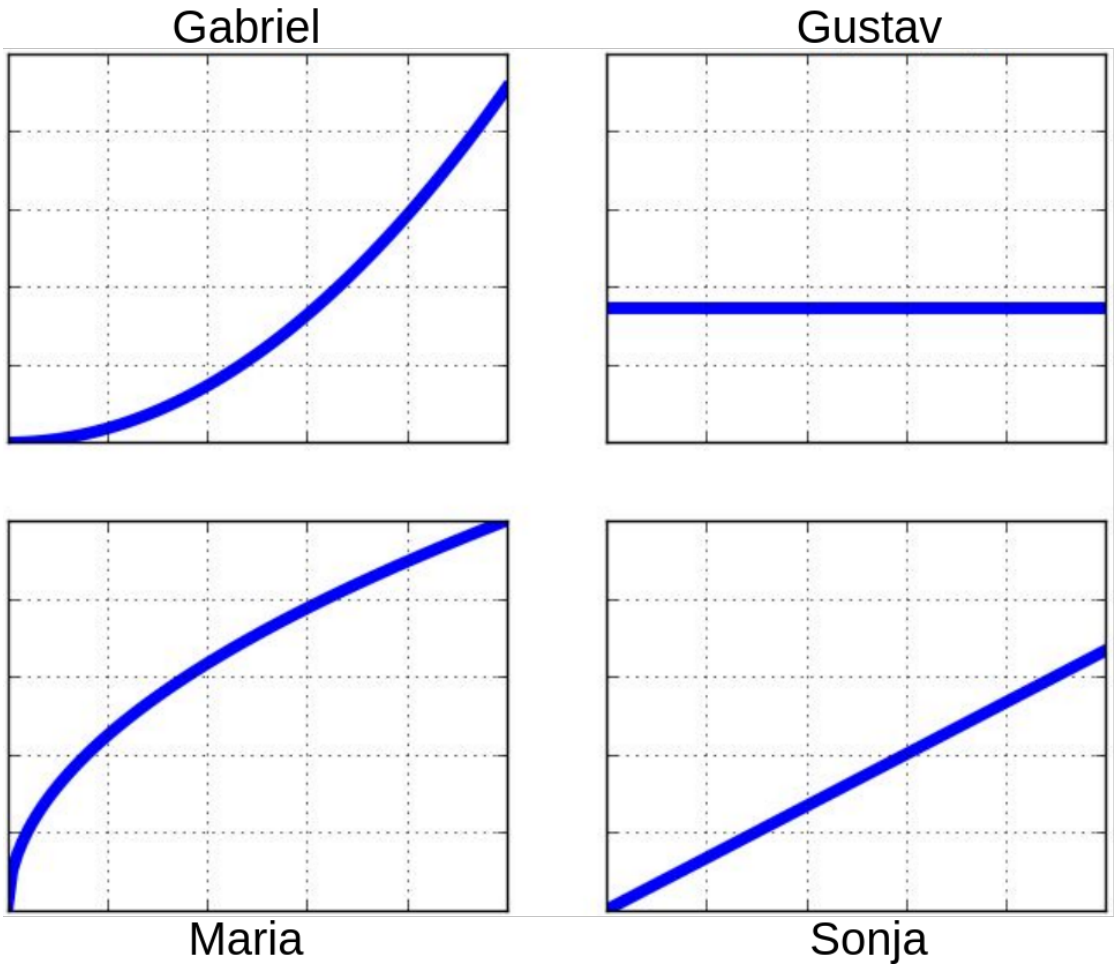



Figure 1: Utility functions.

Some utility functions are sometimes risk-seeking, and sometimes risk-averse.

b) Show that $U(x) = x^3$ is one such function, by providing two lotteries (recall that a lottery is a tuple like this $[(\text{probability}_A, \text{reward}_A), (\text{probability}_B, \text{reward}_B)]$): one for which it is risk-seeking, and one for which it is risk-averse. Provide justification by filling out Table 1 with calculation.

| | Risk-seeking | Risk-averse |
|---|---|---|
| Lottery | | |
| Expected utility of lottery | | |
| Utility of expected monetary value | | |

Table 1: Fill in the missing values.

## 2 Problem 2: Decision Network

Consider a student who has the choice to buy or not buy a textbook for a course. We'll model this as a decision problem with one Boolean decision node, B, indicating whether the agent chooses to buy the book or not. Furthermore, we want two Boolean chance nodes, M, indicating whether the student has mastered the material in the book, and P, indicating whether the student passes the course. You might think that P would be independent of B given M, but this course has an open-book final—so having the book helps. Of course, there is also a utility node, U. A certain student, Sam, has an additive utility function consisting of two parts: $U_1(\neg b) = 0$ for not buying the book and $U_1(b) = -\$100$ for buying it; and $U_2(p) = \$2000$ for passing the course and $U_2(\neg p) = 0$ for not passing. In other words, Sam wants to maximize $U = U_1 + U_2$. Sam's conditional probability estimates are as follows:

$$
\begin{aligned}
P(p|b,m) &= 0.9 \\
P(p|b,\neg m) &= 0.5 \\
P(p|\neg b,m) &= 0.8 \\
P(p|\neg b,\neg m) &= 0.3 \\
P(m|b) &= 0.9 \\
P(m|\neg b) &= 0.7
\end{aligned}
$$

a) Draw the decision network for this problem.

b) Compute the expected utility of the two decision alternatives $B = b$ (buying the book) and $B = \neg b$ (not buying it). Show the calculations.

c) What should Sam do? Explain the reasoning behind your answer.

# 3 Problem 3: Markov decision process

Assume that we have a simple 3 state Markov decision process (MDP) with two actions (L and R). The transition probabilities are given in Table 2, 3.

| | Moving to | | |
|---|---|---|---|
| | State 1 | State 2 | State 3 |
| Going from state 1 | 0 | 1/4 | 3/4 |
| Going from state 2 | 3/4 | 0 | 1/4 |
| Going from state 3 | 1/4 | 3/4 | 0 |

Table 2: Transition probabilities of Action L. Note that the y-axis denotes the states you are going from, and the x-axis denotes the states you are arriving at.

| | Moving to | | |
|---|---|---|---|
| | State 1 | State 2 | State 3 |
| Going from state 1 | 0 | 3/4 | 1/4 |
| Going from state 2 | 1/4 | 0 | 3/4 |
| Going from state 3 | 3/4 | 1/4 | 0 |

Table 3: Transition probabilities of Action R. Note that the y-axis denotes the states you are going from, and the x-axis denotes the states you are arriving at.

The reward in state 2 is 1 and 0 elsewhere. The discount factor is 0.5.

a) Show the utility (value) estimates for the first two iterations of the value iteration algorithm. To make things simpler, assume that you keep a copy of the utility estimates from the previous iteration and use those in the new iteration. Explain how you did it.

| | | | |
|---|---|---|---|
| Initial | U[1] = 0 | U[2] = 0 | U[3] = 0 |
| Iteration 1 | U[1] = | U[2] = | U[3] = |
| Iteration 2 | U[1] = | U[2] = | U[3] = |

b) The converged utility values of the states, with discount factor = 0.5, are **approximately** $U_1 = U_3 = 0.5$ and $U_2 = 1.25$. Use the approximately optimal values of the utilities, and compute the best action for state 1 from these values numerically. Explain how you did it.

# 4 Problem 4: Value Iteration

Imagine, there is a frozen lake from your home to the office, and you need to walk on the frozen lake to reach your office. But oops! There will be holes in the frozen lake in between, so you have to be careful while walking in the frozen lake to avoid getting trapped at holes. Look at Figure 2, where,

- S: the starting position (Home).
- F: Frozen lake which consists of tiles that you can walk on.
- H: Hole which you have to avoid.
- G: the Goal (office).

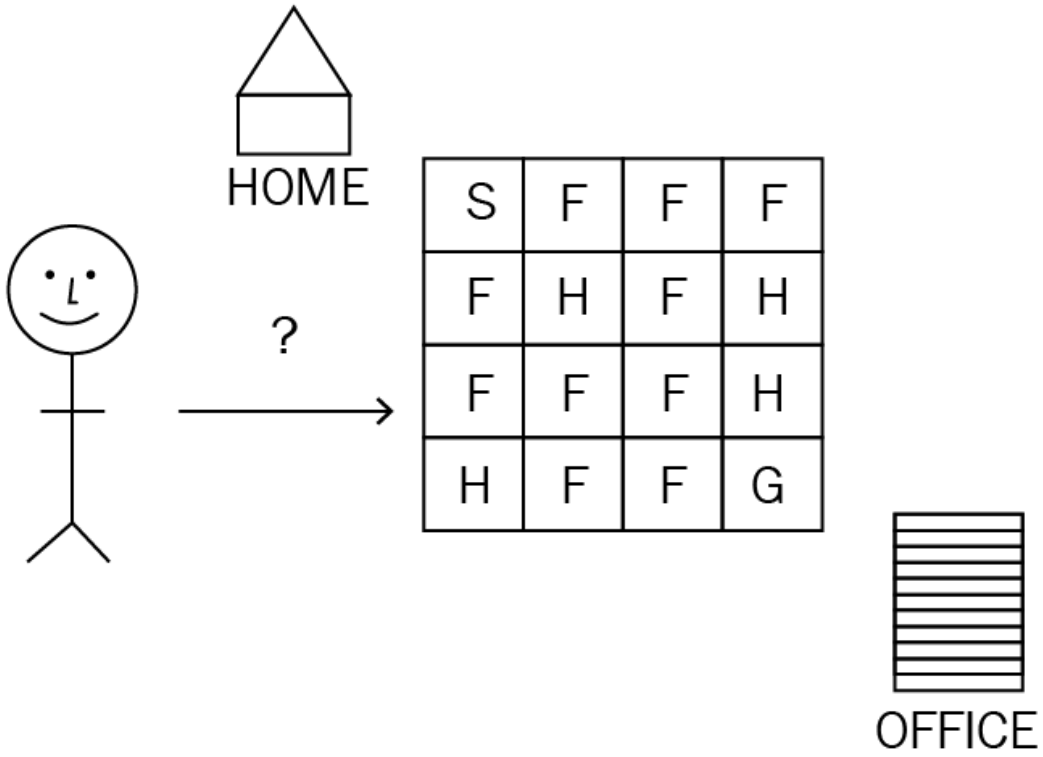



Figure 2: The environment we are going to use.

Our goal is to get from S to G without falling into the holes H in the shortest amount of time.

The states are represented as integers from 0 to 15. Actions are represented as strings left, down, right and up.

We have provided skeleton code for this problem along with the necessary functions in Python.

The information about the skeleton code is written as docstrings and comments in the provided file.

The skeleton code provides the following methods:

1. **get_outcome_states**: Fetches the possible outcome states given the current state and action taken.
2. **get_transition_probability**: Fetches the transition probability for the provided outcome states given the current state and action taken.
3. **get_reward**: Fetch the reward given the current state.

Furthermore, the skeleton code provides these constants stored in the variable **constants**:

1. **number_states**: Number of states in this Markov decision process.
2. **number_actions**: The size of the action space.
3. **gamma**: The discount factor.
4. **epsilon**: The maximum error allowed in the utility of any state, where value iteration is said to have converged.

These methods and the variable **constants** provided are sufficient to implement the value iteration algorithm.

a) Find the utility values for this described environment using the value iteration algorithm (described in Figure 17.4 in the book). You are going to do this by implementing the function "value_iteration" in the provided skeleton code.

b) Display the utility values similar to how it is done in Figure 17.3 in the book. You can do it either by printing the values as a table to the console, or visualize it with a library such as Matplotlib.

c) Based on the utility values find the optimal policy by implementing the method "extract_policy" and visualize it. You can do it either by printing the (state, action) pairs to the console, or visualize it with a library such as Matplotlib.

**Note:** Include all your visualizations and/or prints to the console in the PDF file.