CMPT 310
Artificial Intelligence Survey

Simon Fraser University
Spring 2021

Instructor: Oliver Schulte

# Assignment 2: Chapters 4, 3, 5.

*Instructions*: The university policy on academic dishonesty and plagiarism (cheating) will be taken very seriously in this course. *Everything submitted should be your own writing or coding.* You must not let other students copy your work. On your assignment, put down your **name**, the number of the assignment and the number of the course. Spelling and grammar count. *For full credit on calculation questions, you should include a brief explanation of how you arrived at your numbers.*

Group Work: Discussions of the assignment is okay, for example to understand the concepts involved. If you work in a group, put down the name of all members of your group. There should be no group submissions. Each group member should write up their own solution to show their own understanding. This is different from the programming component where you can make a joint group submission.

For the due date please see our course management server https://courses.cs.sfu.ca. The time when you upload your assignment is the official time stamp. If your assignment is late because you did not figure this out soon enough, you will lose marks according to the syllabus policy.

Terminology: The questions are not self-explanatory. Even ordinary English words (e.g., "rationality") may not have their ordinary meaning in an AI context. Part of your task is to learn the AI terminology required to understand the questions.

*Handing in the Assignment*. Please use the submission system on courses.cs.sfu.ca. You should post a single pdf document that contains your written answers, as well as any screenshots required.

*Getting Help*. Check the syllabus for communication policy. You have the textbook, the lecture notes, the discussion forum, and you can ask us in office hours or class sessions. We do not provide individual email support.

## Chapter 3. 48 points.

See Programming Web page posted on the course site.

## Chapter 4. Local Search in Continuous Spaces. 26 points.

A common problem in machine learning is to find hypotheses that explain the data as well as possible. Let's consider a simple but important instance: modelling coin flips. Suppose you flip a coin 10 times and you observe 6 heads and 4 tails. You assume that the coin flips are mutually independent, and that the chance of getting heads on any given toss is some probability $p$ between 0 and 1 (inclusive). Which value of $p$ best explains the data?

For a fixed $p$, the probability of seeing 6 heads and 4 tails is given by

$f(p) = p^6 * (1-p)^4$.

Since our goal is to maximize this function, we minimize

$-f(p) = -(p^6 * (1-p)^4)$.

Since the logarithm is monotone increasing, we can also consider the natural logarithm $ln(-f(p))$, that is, minimizing the function

$l(p) = -(6ln(p) + 4ln(1-p))$.

1. Use gradient descent to try and find a minimizing value of $p$. You may do this using either the $-f$ or the $l$ function.
    *a.* (3) Write down the gradient (derivative) of the function you chose. (Hint: this is probably easier for $l$.)
    *b.* (10) Try to get close to the minimum in 5 gradient descent steps. Use as your initial guess $p=1/2$ (the coin is fair), and for the first 3 step sizes, use 0.04, 0.02, 0.01. The last 2 step sizes you can choose for yourself. For your answer fill in the table below.

| step | p | Step size |
|---|---|---|
| 0 | 1/2 | 0.04 |
| 1 | Fill in | 0.02 |
| 2 | Fill in | 0.01 |
| 3 | Fill in | Fill in |
| 4 | Fill in | Fill in |
| 5 | Fill in | |

2. Use the Newton-Raphson Method to try and find a minimizing value of *p*. The update formula is given in the lecture notes. You may do this using either the *-f* or the *l* function.

   *a.* (3) Write down the second-order gradient (derivative) of the function you chose. (Hint: this is probably easier for *l*.)

   *b.* (10) Show the results of 5 Newton-Raphson steps. Use as your initial guess *p=1/2* (the coin is fair).

| step | p |
|---|---|
| 0 | 1/2 |
| 1 | Fill in |
| 2 | Fill in |
| 3 | Fill in |
| 4 | Fill in |
| 5 | Fill in |

# Chapter 5. Adversarial Search. 26 points total.

*Summary*. The minimax algorithm finds perfect play in finite zero-sum game trees. This assignment gives you practice in finding the minimax values.

Consider the game of Figure 1. Player A moves first. The two players take turns moving, and each player must move his token to an open adjacent space in either direction. If the opponent occupies an adjacent space, then a player may jump over the opponent to the next open space if any. For example, if A is on 3 and B is on 2, then A may move back to 1. Repeated states are not allowed, so a player may not move to a space if this causes the play to enter a board position that was reached before. <u>The game ends</u> in the following conditions:

- Player A reaches space 3, then the value of the game to A is +1.
- Player B reaches space 2, then the value of the game to A is -1.
- Either player reaches the opposite side (i.e., A reaches 4 or B reaches 1). The value of the game to A is 0.
- Stalemate: The player to move has no legal move. Then the value of the game to A is 0.

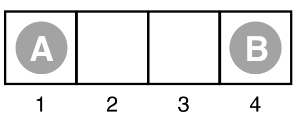


**Figure 1**

1. (10) Draw the complete game tree, using the following conventions.
    a. Write each state as ($s_A$, $s_B$), where $s_A$ and $s_b$ denote the token locations.
    b. Put each terminal state in a square box and write its game value in a circle.
2. (10) Now mark each node with its backed-up minimax value (also in a circle).
3. (6) Suppose that we allowed repeated states and otherwise kept the rules the same.
    a. Describe the optimal strategy for each player (informally but precisely).
    b. Does the standard minimax algorithm terminate with a strategy for each player? If yes, does it find the optimal strategy (from part a)? If not, how could you modify the standard minimax algorithm so that it terminates after finding the optimal strategy for each player in this game? The modification should be general in the sense that the modified algorithm can be applied to any game tree, not just this game.

## Optional Study Questions (not graded)

- Usually local search methods have problems with getting stuck in local minima. Is there an issue with local minima in the local search problem 1 (searching for an optimal coin probability)? Why or why not?