Due Date: March 30th, 2016 at 11:59pm.

Submission Instructions: All assignments are to be submitted through LEARN, in the Dropbox in the Assignment 4 folder. Late assignments will be accepted up until April 1st at 11:59pm. Please read the course policy on assignments submitted after the official due date. *No assignments will be accepted, for any reason, after 11:59pm on April 1st.*

Lead TA: Luyu Wang (l366wang@uwaterloo.ca). Office hours and location will be announced on Piazza.

Announcements: The following exercises are to be done individually.

Written Exercises

1. Value Iteration for Markov Decision Processes [40 points]

While we covered MDPs earlier in the semester, we have not yet had an assignment covering them. Since a number of people struggled with MDPs on the midterm exam, I thought it was important that you get a little more experience working with them.

r	-1	**+10**
-1	-1	-1
-1	-1	-1

Consider the 3x3 grid world shown above. An agent in the grid world can move up, down, left, or right at each time step, but cannot move off the edge of the grid.

At each timestep, the agent receives a reward equal to the value shown in the square it is currently in, and may then attempt to move to an adjacent square. **Moving to the top right square with reward 10 ends the game**. When the agent tries to move in a given cardinal direction, it has an 80% chance of successfully moving there, and a 10% of moving in either perpendicular direction. For example, if the agent tries to move left from the centre square, it moves up with probability 0.1, down with probability 0.1, and left with probability 0.8. If the agent attempts to move off the grid, it remains in its current position until the next timestep.

Use value iteration to find an estimate of the expected utility of starting at different points in this grid, and executing an optimal policy. You may either implement value iteration in the language of your choice, or perform the calculations by hand.

There is no penalty for doing the calculations by hand, but doing so is likely to be more time consuming than writing the program. It is permissible to use a spreadsheet if you want to perform the calculations that way, but be sure to include your spreadsheet file (exactly as if you had written a program).

You will need to perform no more than 25 iterations to uncover the optimal policy in any of the problem instances.

Assume a discount factor of 0.9, and show the policy obtained in each of the following cases (10 points each):

a. r = +100

b. r = -3

c. r = 0

d. r= +3

Include in your submission:

- A copy of your code (if applicable).
- A copy of your calculations or spreadsheet file (if applicable).
- A 3x3 grid showing the optimal policy for each case (e.g. arrows to show the direction to move in).
- A 3x3 grid showing the estimated expected utility of starting in each square.

2. Games [10 points]

Solve the normal form game in Figure 1 by eliminating dominated strategies. In each step, show the reduced game created by eliminating a dominated strategy. Verify the resulting outcome is a Nash equilibrium of the game.

	N	C	J
N	73, 25	57, 42	66, 32
C	80, 26	35, 12	32, 54
J	28, 27	63, 31	54, 29

Figure 1: Normal Form Game for Question 2.

Programming Exercises

4. Decision Tree for Equine Colic Diagnosis [80 points]

Equine colic is one of the leading causes of death in adult horses (https://www.liverpool.ac.uk/equine/common-conditions/colic/). However, if diagnosed early enough, it is often surgically curable. Using the language of your choice, write a decision tree algorithm that will learn to diagnose whether a patient is healthy or has colic. Use the **horseTrain** file, (found in the Assignment 4 folder) to train the decision tree. Each training instance has 16 numeric attributes (features) and a classification, all separated by commas. The attributes correspond to the following measurements made from each patient at admission to the clinic.

1. K
2. Na
3. CL
4. HCO_3
5. Endotoxin
6. Aniongap
7. PLA2
8. SDH
9. GLDH
10. TPP
11. Breath rate
12. PCV
13. Pulse rate
14. Fibrinogen
15. Dimer
16. FibPerDim

In the decision tree, use only binary tests, *i.e.* each node should test whether a particular attribute has a value greater or smaller than a threshold. In deciding which attribute to test at any point, use the information gain metric. Set the node test threshold for each potential attribute using this same metric *i.e.* at each point, see all the values that exist for a particular attribute in the remaining instances, order those values, and try threshold values that are (half way) between successive attribute values. Use the threshold value that gives the highest information gain. Allow the same attribute to be tested again later in the tree (with a different threshold). This means that along a path from the root to a leaf, the same attribute might be tested multiple times. After learning the decision tree, use the **horseTest** file to test the generalization accuracy of the tree.

Submit the following

1. Your code.
2. Output of the algorithm – annotated if it is not in an easy to understand form.
3. Picture of your decision tree. Hand drawn and then scanned is fine.
4. (15 points) Answer the question "How many of the training instances does the tree classify correctly?"
5. (15 points) Answer to the question "How many of the test instances does the tree classify correctly?"
6. (10 points) Description of how you used the information metric.
7. Note that a total of 40 points are awarded for the correct decision tree. That is, the decision tree produced when following the instructions in the assignment.

Format of data files

We are providing you with data from an actual vet clinic. Each line in a file is one instance. The first 16 numbers are the values for the 16 attributes listed above. The last entry on a line is whether the horse was healthy or not. You are allowed to reformat the data files in what ever way you want so as to make reading them easier for your program.