Assignment 4

Decision Making under Uncertainty and Learning

Deadline: May 1, 11:59pm
Available points: 110 - Perfect score: 100 points.

Assignment Instructions:

Teams: Assignments should be completed by pairs of students. No additional credit will be given for students working individually. You are strongly encouraged to form a team of two students. If you have not done so already, please inform the TAs as soon as possible about the members of your team so they can update the scoring spreadsheet (find the TAs' contact info under the course's website: `http://www.pracsyslab.org/cs440`).

Submission Rules: Submit your reports electronically as a PDF document through Sakai (`sakai.rutgers.edu`). For programming questions, you need to also submit a compressed file via Sakai, which contains your code. Do not submit Word documents, raw text, or hardcopies etc. Make sure to generate and submit a PDF instead. Each team of students should submit only a single copy of their solutions and indicate all team members on their submission. Failure to follow these rules will result in lower grade in the assignment.

Late Submissions: No late submissions are allowed. You will be awarded 0 points for late assignments!

Extra Credit for LaTeX: You will receive 10% extra credit points if you submit your answers as a typeset PDF (using LaTeX, in which case you should also submit electronically your source code). Resources on how to use LaTeX are available on the course's website (`http://www.pracsyslab.org/cs440`). There will be a 5% bonus for electronically prepared answers (e.g., on MS Word, etc.) that are not typeset. If you want to submit a handwritten report, scan it and submit a PDF via Sakai. We will not accept hardcopies. If you choose to submit handwritten answers and we are not able to read them, you will not be awarded any points for the part of the solution that is unreadable.

Precision: Try to be precise. Have in mind that you are trying to convince a very skeptical reader (and computer scientists are the worst kind...) that your answers are correct.

Collusion, Plagiarism, etc.: Each team must prepare its solutions independently from other teams, i.e., without using common code, notes or worksheets with other students or trying to solve problems in collaboration with other teams. You must indicate any external sources you have used in the preparation of your solution. Do not plagiarize online sources and in general make sure you do not violate any of the academic standards of the course, the department or the university (the standards are available through the course's website: `http://www.pracsyslab.org/cs440`). Failure to follow these rules may result in failure in the course.

Question 1: Consider the specification of a Markov Decision Process according to the following figure. Code your own implementation of Value Iteration and compute the optimal policy as well as the optimum utilities for this challenge.

Indicate the original utilities you used in order to start the process. Provide at least 5 intermediate results (in terms of optimum utilities and policies) depending on the number of iterations needed for convergence as well as the final results. Describe your implementation and your convergence criterion. Report computation time and number of iterations.

s	$r(s)$
s_1	0
s_2	0
s_3	1
s_4	0

s_i	a	s_j	$T(s_i, a, s_j)$
s_1	a_1	s_1	0.2
s_1	a_1	s_2	0.8
s_1	a_2	s_1	0.2
s_1	a_2	s_4	0.8
s_2	a_2	s_2	0.2
s_2	a_2	s_3	0.8
s_2	a_3	s_2	0.2
s_2	a_3	s_1	0.8
s_3	a_4	s_2	1
s_3	a_3	s_4	1
s_4	a_1	s_4	0.1
s_4	a_1	s_3	0.9
s_4	a_4	s_4	0.2
s_4	a_4	s_1	0.8

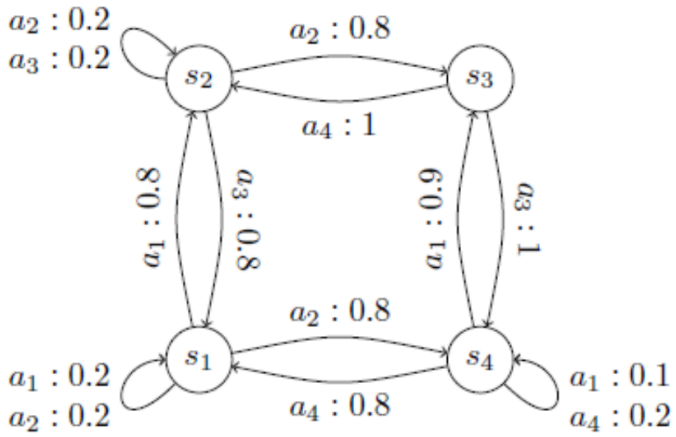

Figure 1: The specification of the MDP problem.

(25 points)

Question 2: Consider the criteria for accepting graduate students at the hypothetical Univ. of Excellence. Each candidate is evaluated according to four attributes:

1. the grade point average (GPA)
2. the quality of the undergraduate degree
3. the publication record
4. the strength of the recommendation letters

To simplify our example, let us discretize and limit the possible values of each attribute: Possible GPA scores are 4.0, 3.6, and 3.3; universities are categorized as rank_1, rank_2, and rank_3; publication record is a binary attribute - either the applicant has published previously or not; and recommendation letters are similarly binary, they are either good or normal. Finally, the candidates are classified into two classes: accepted, or P (for 'positive') and rejected, or N (for 'negative'). Figure 2 provides an example of one possible decision tree determining acceptance.

An applicant doesn't know this decision tree, but does have the data regarding twelve of last year's applicants as in Table 1.

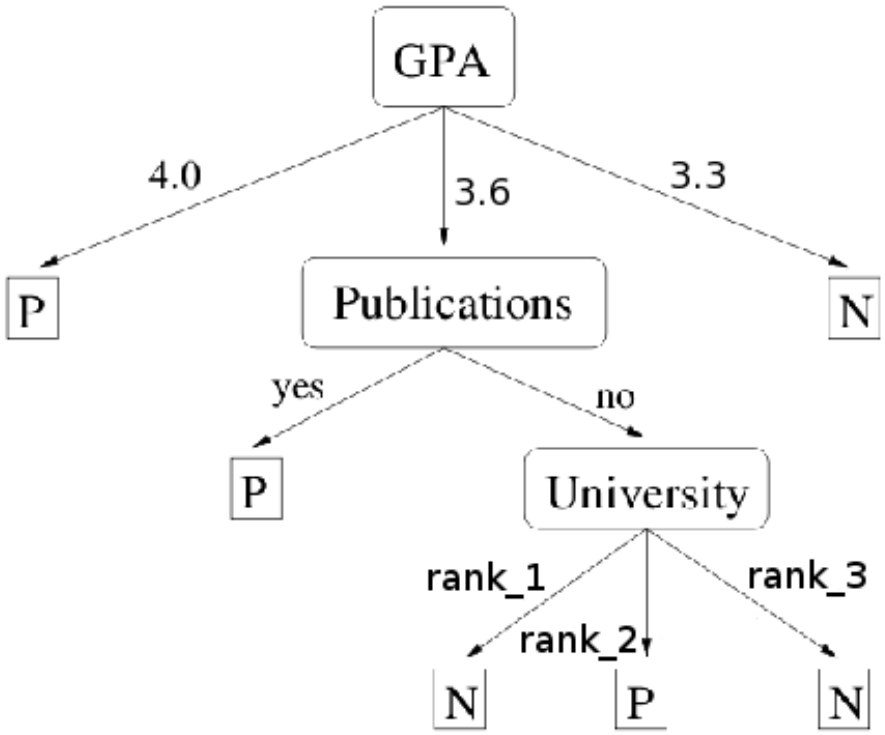

Figure 2: The decision tree that University of Excellence is using to determine acceptance.

a) Does the provided tree correctly categorize the provided examples?

b) The applicant uses the decision tree algorithm shown in class (with the information gain computations for selecting split variables) to induce the decision tree employed by U. of E. officials. What tree will the algorithm come up with? Show the computations involved, in addition to the decision tree itself.
[Hint: The information content of the examples before choosing any split variable is:

$$I(\frac{6}{12}, \frac{6}{12}) = -\frac{6}{12} log_2 \frac{6}{12} - \frac{6}{12} log_2 \frac{6}{12} = 1 \text{ bit of information}$$

You have to find the attribute that has the highest information gain:

$$Gain(A) = I(\frac{p}{p+n}, \frac{n}{p+n}) - \sum_{i=1}^{n} \frac{p_i + n_i}{p+n} I(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i})$$

where attribute A divides the examples into i subsets, and p_i and n_i represent the number of positive and negative examples in subset i.]

c) Is the tree that you got in part b) equivalent to the tree provided here (i.e., do the two trees classify every application in the same way)? If the answer is yes, explain whether this is a coincidence or not. If the answer is no, give an example of a data case that will be classified differently by the two trees.

No.	GPA	University	Published	Recommendation	Class
1	4.0	rank 1	yes	good	P
2	4.0	rank 1	no	good	P
3	4.0	rank 2	no	normal	P
4	3.6	rank 1	yes	good	P
5	3.6	rank 2	no	good	P
6	3.6	rank 3	yes	good	P
7	3.6	rank 3	no	good	N
8	3.6	rank 1	no	good	N
9	3.3	rank 2	yes	normal	N
10	3.3	rank 1	no	normal	N
11	3.3	rank 3	yes	normal	N
12	3.3	rank 3	no	good	N

Table 1: The available knowledge for the applicant.

(20 points)

Question 3: Consider building an SVM for the following two-class training data:
Positive class:

$$[-1\ 3]^T, [0\ 2]^T, [0\ 1]^T, [0\ 0]^T$$

Negative class:

$$[1\ 5]^T, [1\ 6]^T, [3\ 3]^T$$

(a) Plot the training points and, by inspection, draw a linear classifier that separates the data with maximum margin.

(b) This linear SVM is parameterized by $h(x) = w^t x + b$. Write the parameters w and b.

(c) Suppose you observe an additional set of points, all from the positive class:

More positive points:

$$[-2\ 0]^T, [-2\ 1]^T, [-2\ 3]^T, [-1\ 0]^T, [-1\ 1]^T, [0\ 0]^T$$

What is the linear SVM (in terms of w and b) now?

(15 points)

Question 4: Consider the chart below:

(a) Notice that for the above graph there is a perfect linear separator for the two input classes. Therefore, a single perceptron should be able to learn this classification task perfectly. Your task is to replicate the learning process, starting with a random perceptron with weights $w_0 = 0.2$, $w_1 = 1$, and $w_2 = -1$, where the weight w_0 corresponds to the constant offset $i_0 = 1$. For the inputs, just estimate their coordinates from the chart.

Start by adding the perceptron's initial line of separation to the chart. Compute then how many samples are misclassified? Then, select an arbitrary misclassified sample and describe the computation of the weight update (you can choose $\eta = 1$ or any other value; if you like you can experiment a bit to find a value that leads to efficient learning).

Illustrate the perceptron's new line of division in the same chart or a different one, and give the number of misclassified samples. Repeat this process four more times so that you have a total of six lines (or fewer if your perceptron achieves perfect classification earlier).

You can generate the computations and/or graphs either by hand or by writing a simple computer program. If you write a program, please attach a printout, and let the program run until the perceptron achieves perfect classification (after how many steps?).

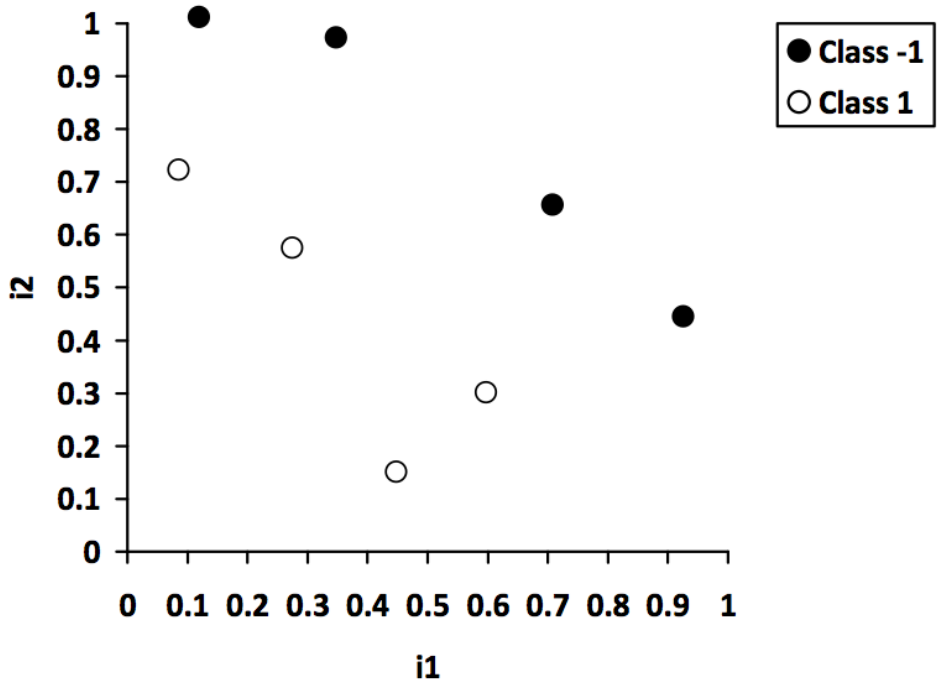

Figure 3: A set of two-dimensional input samples from two classes.

(b) If your perceptron did not reach perfect classification, determine a set of weights that would achieve perfect classification, and draw the separating line for those weights.

(c) Now assume that less information were available about the samples. For instance, consider we only know the value for i_1 for each sample, which means that our perceptron has only two weights to classify the input as best as possible, i.e., it has weights w_0 and w_1, where w_0 is once again the weight for the constant offset $i_0 = 1$. Draw a diagram that visualizes this one-dimensional classification task, and determine weights for a perceptron that does the task as best as possible (minimum error, i.e., minimum proportion of misclassified samples). Where does it separate the input space, and what is its error?

(25 points)

Question 5: Consider the more difficult classification problem shown in the chart below:

(a) As you certainly noticed, a single perceptron cannot do this classification task perfectly. Determine the minimum error that a single perceptron can reach, and show the dividing line in the input space for such a perceptron.

(b) Clearly, we need a multi-layer perceptron to do this task perfectly. Develop such a system of connected perceptrons and write it down, together with the required weights for each unit. Illustrate the dividing lines for these perceptrons in a copy of the chart above.

(25 points)

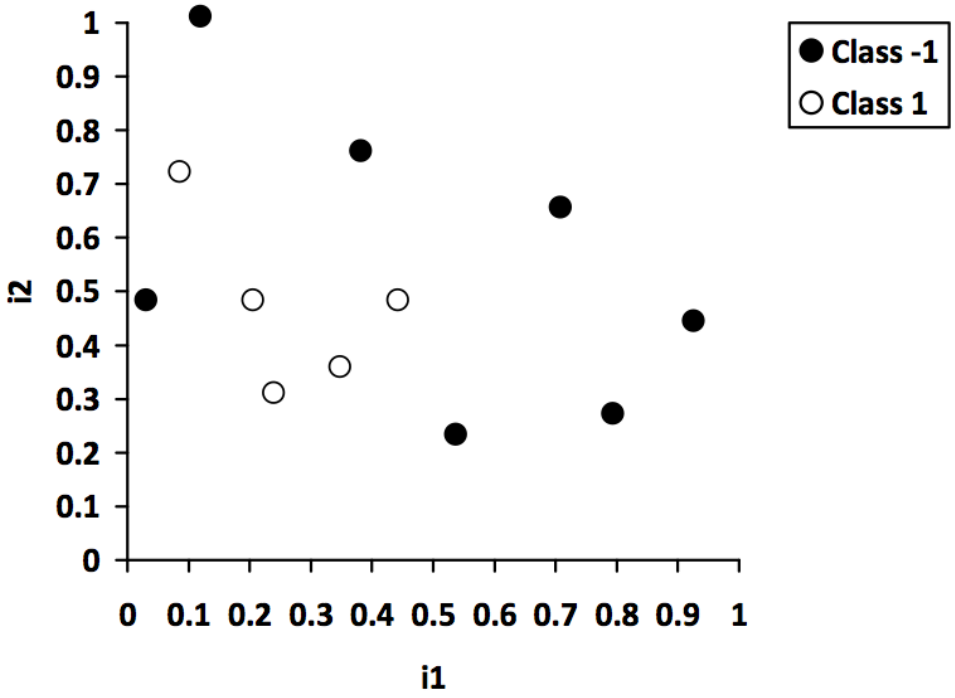

Figure 4: A set of two-dimensional input samples from two classes.