COMPSCI 383 – Fall 2020

Homework 5

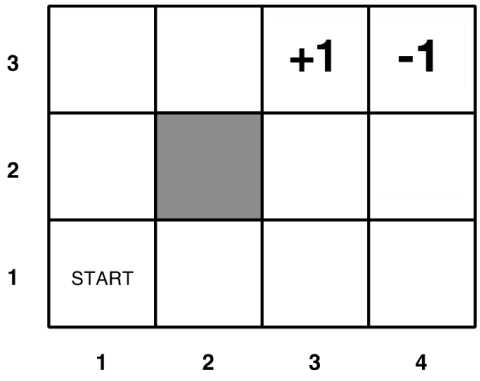
Due Thursday, November 19th at 11:59pm ET

You are encouraged to discuss the assignment in general with your classmates, and may optionally collaborate with one other student. If you choose to do so, you must indicate with whom you worked. Multiple teams (or non-partnered students) submitting the same code will be considered plagiarism.

Code must be written in a reasonably current version of Python (>3.0), and be executable from a Unix command line. You are free to use Python's standard modules for data structures and utilities, as well as the pandas, scipy, and numpy modules. Do not use any modules or libraries that implement score standardization or perceptron learning.

1. Value Iterations

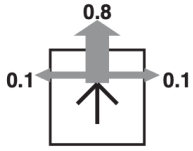
Consider the MDP shown on the right with eleven states, including two terminal states (3,3) and (4,3). The reward function, discount rate, and noise factor is indicated in the diagram.



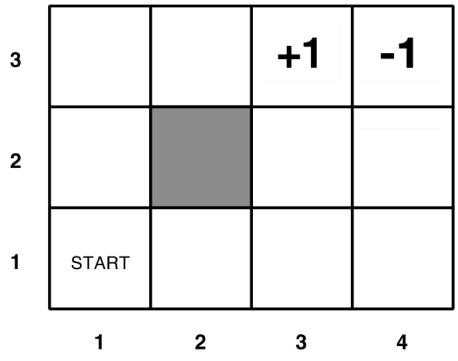
Your task is to simulate the Value Iteration algorithm to estimate the utility of each non-terminal state. Assume that at iteration 0, all utilities are initialized to 0. Fill in the utility values for the subsequent four iterations below, showing your work (you can exclude calculations for states that are unchanging or zero).

$$R(3,3) = +1$$
$$R(4,3) = -1$$
Otherwise,
$$R(s) = -0.05$$

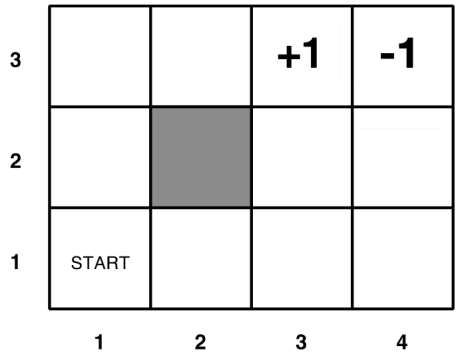
$$\gamma = 0.9$$



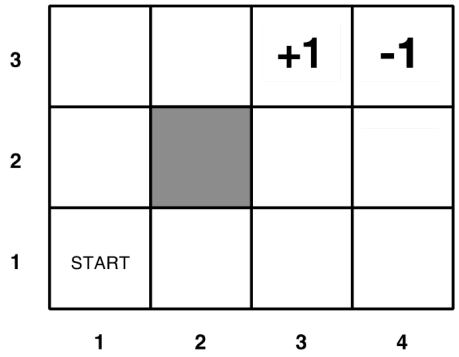
Iteration 1:



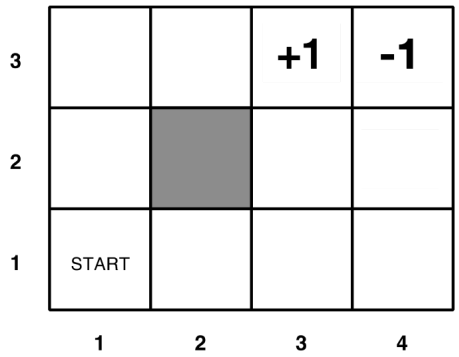
Iteration 2:



Iteration 3:



Iteration 4:



2. Perceptron Wining and Winning

Implement the algorithm for learning a set of attribute weights for a multiclass perceptron (see Lecture 21, slides 26-29) to classify the examples found in wine.csv[1]. Your code should repeatedly iterate through the examples to generate a set of attribute weights for each of the three classes (listed the not-so-strangely named "Class" column).

The file `wine_weights.py` contains a bit of code to get you started. The supplied code will read in the data, learn the weights, and print out the results. You are responsible for filling in a couple of function bodies as instructed below. While you are free to create additional functions, please do not change the API (method signature or return value) for `learn_weights()` or `standardize()` as specified in the comments.

Your first task is to fill in the function body for the `learn_weights()` function, which returns a dictionary containing the attribute weights for each class when given a list of training examples. This function should replicate the multiclass perceptron learning algorithm covered in Lecture 21. Note that your algorithm should halt after the weights misclassify none of the training examples or 1000 iterations, whichever comes first. Once you have your function working, answer the following questions:

(a) How well does your algorithm perform on the training instances? What is the minimum number of misclassifications you are able to achieve, and how many iterations did it take to reach that level?

(b) What does your answer above say about the separability of the data?

Many machine learning algorithms are sensitive to the scales of different features. One way to address this is to "standardize" attribute values by converting them to z-scores[2] on a column-by-column basis. Implement the `standardize()` function to transform your data into standardized form (note that you should not modify the class values!).

---

[1] Adapted from the UCI Wine data set: http://archive.ics.uci.edu/ml/datasets/Wine

[2] See: https://en.wikipedia.org/wiki/Standard_score

Uncomment lines 92-94 to run your code with standardized training data.

(c) How does the behavior of your learning algorithm differ when using standardized attribute values?

What to Submit

- A pdf with answers to the questions in 1 and 2.
- Your modified version of `wine_weights.py` containing your code.