(364-1-1441) Foundations of Artificial Intelligence

Problem Set 3: *Half a league, half a league, half a league onward*

Due: 6/1/2022

Title from the poem The Charge of the Light Brigade *by Alfred, Lord Tennyson*

You need to submit both code and written answers. Problem 1 is a programmatic one. Problems 2 and 3 only require written answers and not programming. You will submit one `q1.py` file containing your code, and one `answers.pdf` file containing your written, typewritten (not a scan of handwriting) answers, in English or Hebrew.

Make sure your code compiles, and the output matches what is requested. Your grader will not debug your code, and if it does not compile or output correctly, they will not be in charge of fixing your errors, even if its cause was a very minor mistake.

Also, to simplify your work process, as well as the grader's, please name your variables and methods in a meaningful way (e.g., name a function `entropyCalculation` and not `myAwesomeFunction`).

1 Finding If it's Going to be Busy in Seoul Bicycle Lanes

This exercise includes data from Seoul's bike sharing scheme, and in it you will use a decision tree to predict if it is going to be a busy hour on Seoul's bicycle lane (if there are more than 650 bicycle rentals that hour). In order to do this, you will use the data file `SeoulBikeData.csv`, which you can download from the course Moodle. The description of the data file is found in the `SeoulBikeData.pdf` file you can also download from the course Moodle.

Broadly, the second variable indicates the number of bicycles rented in a particular hour. The other variables show various other parameters of the day and time. Since some of these variables are continuous, you will need to set thresholds yourself to "bucket" them – which means each of your trees might look different than someone else's (which is fine!). Notice you can parse the date range to get a day and month information from the file as well, to use as variables for your tree.

When you build your decision trees, you are expected to use entropy to calculate the more meaningful attributes, and to use the χ^2 test to prune vertices.

Your Python code will include the following functions. You can assume the file `SeoulBikeData.csv` is found in the same directory as your code.

`build_tree(<float> ratio)` for `ratio`$\in (0, 1]$, so function definition:

```
def build_tree(ratio):
```

You need to build a decision tree, using *ratio* ratio of the data (so if $ratio = 0.6$, you arbitrarily choose 60% of the data), and validate it on the remainder. The outcome is printing out the decision tree, and reporting the error.

`tree_error(<int> k)` so function definition:

```
def tree_error(k):
```

You need to report the quality of the decision tree by building k-fold cross validation (k being an integer number received by the function for the number of folds), and reporting the error.

`is_busy(<array> row_input)` so function definition:

```
def is_busy(row_input):
```

You receive an an input for a particular day. It will be an array of values, in the same order as the data file (but without the column on the number of rented bikes, of course). You return 1 if you think the day will be a busy one (more than 650 rentals), and 0 if not.

Obviously, you cannot use any library, package, module, or existing code that implements decision trees. You can use mathematical libraries for simple calculations, but everything, including entropy and χ^2 tests needs to implemented by you.

2 Hebbian Learning

In class, we saw a symmetric, one-neuron-at-a-time updating neural net converges to a value.

(a) Show an example of a non-symmetric (yet still one-neuron-at-a-time updating) neural net that does not converge.

(b) Show an example of a symmetric neural net that updates simultaneously (i.e., not one-neuron-at-a-time updating) that does not converge.

3 Learning

Could you use both neural nets and decision trees together in a boosting algorithm? How would you use the boosting weights in the training of a deep neural net?