Weighting Mesopotamia

A Teknowledge Activity by Christopher George

1.0 The Story

You are a government leader of Mesopotamia, and Mesopotamia is facing an influx of immigrants, many of whom die, and you want to determine what factors contribute to people dying so that immigrants can make an informed decision.

1. The initial settlement is conducting the first 5 year experiment. At the end of the five years, they have their data telling them,the DNA sequence of each individual and whether they survived for five years or not. The data may look something like the following (training set). **Note: Mesopotamians had different symbols to represent a DNA strand.**

[“MYDNASTRANDS”, “live”]

[“ANOTHEREXMPL”, “die”]

[“XYZSDUDNOOPS”, “live”]

[“ASDFQWERZXCV”, “die”]

1. After the first five years, a new group of settlers come in (test set). Their DNA sequence is written down, and the settlers predict whether they will live or die based on the DNA sequences and results of the previous experiment. (explained below)
2. The results are looked at, after five years, and weights are given to each expert depending on whether it predicted correctly.

2.0 Write Up

There are three experts, which we call “Speed”, “Height”, and “Weight”.

The first four letters in each DNA strand is used as the value for speed of a person.

The next four letters in each DNA strand is used as the value for the height of a person.

The last four letters in each DNA strand is used as the value for the weight of a person.

For example, in the first data element above, [“MYDNASTRANDS”, “live”].

“MYDN” is correlated with the “Speed” expert.

“ASTR” is correlated with the “Height” expert.

“ANDS” is correlated with the “Weight” expert.

Each of these experts have their own associated KNN that is used to test whether or not a person will live based on points closest to their own in the training set. (Check out the distance formula used to see how this number is correlated)

Using the KNN for each expert, we predict the label and test it against the actual label, downweighting the current expert when the predicted label and actual label are different.

3.0 Challenge

Complete the following functions:

|  |
| --- |
| # Challenge 0: For the first step, complete the getNeighbors function which  # will iterate over the trainingSet call the distance function, and then  # add the distance and traingingInstance to a list similar to last time.  # Be careful, and make sure to send the correct information in the distance  # function. Then similarly to our original KNN, retrieve the top k instances  # from the created 2-D list of distances and points.  # Challenge 1: For the next step, we will be completing the getLabel function.  # We are trying to predict the label ‘live’ or ‘die’, so similarly to the  # previous KNN, we will iterate through the neighbors, and add the labels to  # our dictionary accordingly.  # Challenge 2: For the final step, you will want to follow a similar format  # to our previous iteration of KNN. Except in this case (a) you will want to  # iterate through each expert over each test instance and (b) after you have  # successfully found the label for the given expert:instance, you want to check  # if it is correct.  # If it is correct, then leave it, if it is not correct, then you will  # have to downweight your current expert. |

4.0 Bonus Challenges

* **Change the functions to instead of an expert being weighted, each element in the training set has a weight that is changing based on if they correctly predicted a neighbor.**
* This will involve changing the dataCollection.py file to include a weight for each point, then putting the new outcome into the data.txt file, and finally modifying your mainKNN file.
* Modify KNN (or create a new file) to test whether you will live or die with these new experts!
* Modify KNN to allow different amount of letters per expert!
* Add more experts!

5.0 Comprehension Questions

|  |
| --- |
| Challenge 4: COMPREHENSION QUESTIONS -- think about them and/or discuss them  with a friend. We will discuss them at the end of class.   * How could you change the experts in this situation to enhance this algorithm? What happens if you wait each individual point in the training set? * When are the experts downweighted? * As we had learned in the past weighted majority is an online learner, which means that it’s constantly changing with each data point. Additionally KNN, is an offline learner which means that it should just take all of the data and modify nothing until it’s completely run through it’s test set. Is it okay for us to be mixing an online and offline learner together like this? What downsides does it bring? What upsides? * We provided one way to measure the distance between two 4-character strings. How else could you measure the distance between strings? Keep in mind that different distance metrics will impact how the algorithm works. For example, with our current metric the order of the DNA strange matters (i.e. “ABCD” is different from “ACDB”). Can you think up a metric where the order does not matter? If you have time, feel free to code up different metrics and see how that affects the algorithm. |

