Assignment 4: Normalization, k-NN and Perceptron

Due: Wed February 12, at 11:59 pm

Objectives

The goals for this assignment are as follows:

1. Understand the mechanics and impact of z-score normalization.
2. Become more familiar with the perceptron and how it differs from other learning algorithms, namely the k-Nearest Neighbor algorithm.

Z-Score Normalization

Write a function/method that implements z-score normalization based on a set of examples S. Recall that the z-score for an example is defined as: $z_{i,j} = \frac{x_{i,j} - \mu_j}{\sigma_j}$, where $x_{i,j}$ is the value of feature j of example i, μ_j is the sample mean of feature j, and σ_j is the sample standard deviation of feature j.

Remember the following when implementing the function/method: i) calculate μ_j and σ_j from the **TRAINING SET ONLY** then apply it to the **TRAINING AND TEST SET**, and ii) to divide by $N - 1$ when calculating the σ values, where N is the number of examples in S (dividing by N produces a biased estimate of the standard deviation).

Perceptron vs. k-NN

In this assignment, you will implement the perceptron using stochastic gradient descent (see page 93 of the perceptron handout) and compare the performance with the the k-NN classifier you implemented in Assignment 1. For the perceptron, you will need to z-score normalize the data first before you run it. For k-NN, you will run it with unnormalized and normalized data. We have provided you with two datasets named *vertebrate* and *sonar*. If you are interested in the datasets, you can read about them here:

http://archive.ics.uci.edu/ml/datasets/Vertebral+Column
http://archive.ics.uci.edu/ml/datasets/Connectionist+Bench+%28Sonar%2C+Mines+v\
s.+Rocks%29

For each dataset, we have provided you with two versions. The first version is the original data split into a training and testing set. In the second version, we have added noise features (the values were randomly chosen) to both the training and the testing set. Note that we have mapped the original class values onto 1 and -1 for your convenience.

You will perform the following experiments:

Experiment 1: Run the perceptron with an η (i.e., learning rate) of 0.01 and 0.001 on the normalized data sets. You should initialize your starting weights to 0 and stop training after 500 cycles through all the training data. There should be a total of 8 result files once you finish – there are two versions of each of the two datasets and there are two settings of the parameter η.

Experiment 2: Run the k-NN algorithm on both datasets for each of the original and noisy versions of the data at $k = 1, 3, 5$. Also, you will be testing the effect of running on normalized vs. unnormalized data. There should be a total of 24 result files once you finish.

For your write up, you will fill in the following table using the output of Experiment 1. The metric to calculate is percent accuracy on the test test.

	Vertebrate		Sonar	
	Original	Noisy	Original	Noisy
$\eta = 0.01$				
$\eta = 0.001$				

And you will fill in this table using the output of Experiment 2.

	Vertebrate				Sonar			
	Normalized		Unnormalized		Normalized		Unnormalized	
	Original	Noisy	Original	Noisy	Original	Noisy	Original	Noisy
$k = 1$								
$k = 3$								
$k = 5$								

In your discussion of the results please answer the following questions:

1. For k-NN, does normalization help more on one dataset than the other? What about when noisy attributes are added? Do you have an explanation for why? (Hint: Which features are influencing the distance calculation when running on un-normalized data?)

2. Why did we have you look at the performance only for odd values of k for k-NN? (Hint: What additional code would you need to write when moving from $k = 1$ to $k = 2$?)

3. For the perceptron, is there a difference in performance for one or both of the datasets when the learning rate (i.e., η) is varied? If so, which learning rate resulted in better performance? Why?

4. Questions examining the performance differences between k-NN and the perceptron

   (a) On the *vertebrate* dataset without the added noise features, which method performed better? Please explain why you think there is a difference or lack of difference in performance?

   (b) When we add noise features to the *vertebrate* data, which method performed better? Why do you think it did?

   (c) For the *sonar* dataset without added noise features which method performed better? Please explain why you think there is a difference or lack of difference in performance.

   (d) For the *sonar* dataset, which method performs better when we add noisy features? Please explain why you think it does.

   (e) Is the performance drop from the noisy features as a percentage of the accuracy on the original data better or worse for the *sonar* or *vertebrate* datasets? Please explain why.

Implementation Requirements

1. In addition to the command line format specified in Assignment 1 for the k-NN classifier, your program must accept two additional command line arguments: i) a "-Z <0 or 1>" argument to specify whether normalization should be performed on the data "(-Z 1") or not ("-Z 0"); ii) a "-k <integer>" argument to specify the k-level to run the classifier at. An example run of the k-NN classifier may be:

   ```
   java kNN -Z 1 -k 5 train.arff test.arff output.arff
   ```

   Here, a Java implementation of the k-NN classifier will normalize the data in the input files, perform 5-NN classification with the normalized data, and output the classification results on the test data to the file specified.

2. The perceptron implementation must accept 4 command line arguments. The first argument is the learning rate of the perceptron given as "-eta <float>". The last three arguments are the input training data, input test data, and output file, just as in the case for the k-NN classifier. An example run of the perceptron may be:

   ```
   java perceptron -eta 0.1 train.arff test.arff output.arff
   ```

   Here, a Java implementation of the perceptron will perform classification at a learning rate of 0.1 and output the classification results to the file specified.

3. To generate the output file for each run, copy all contents from the test file. However, **replace the label of each instance with the results from your classification**. Please save the output as regular text files. For the perceptron, name the text files as follows:

   ```
   <dataset>_perceptron_<version>_<option>
   ```

   For example, an output file created with an η of 0.01 on the original (no noise) *vertebrate* data will be named

   ```
   vertebrate_perceptron_nonoise_0.01
   ```

   Similarly, a perceptron output created with an η of 0.001 on noisy *sonar* data will be named

   ```
   sonar_perceptron_noise_0.001
   ```

For k-NN, please name the text files as follows:

`<dataset>_knn_<version>_<normalization>_<option>`

For example, an output file created with $k = 3$ on original (no noise) un-normalized *vertebrate* data will be named

`vertebrate_knn_nonoise_unnormalized_3`

Similarly, an output file created with $k = 5$ on noisy normalized *sonar* data will be named

`sonar_knn_noise_normalized_5`

Scoring

We will be grading your submissions on the machines in Halligan 116/118 or *homework.cs.tufts.edu*.

This assignment will be graded as follows:

1. 40 points for programming portion and correct classification results.
2. 30 points for written portion
3. **30 points for following submission instructions**

If your program does not compile or does not work, **you will receive a zero on the programming portion**.

Submission and Due Date

1. Programming Portion - Please submit, via *provide* the following items:
   (a) A $README$ containing the **correct** instructions on how to compile and run your code.
   (b) A Bash Script that contains the commands to compile and run your program. (For those programming in Python we will waive this provided you document the commands needed in the README)

(c) All source code

2. Written Portion (provide this as **ONE** pdf file)

   (a) Create the table shown above and fill in the numbers (please round to whole numbers, e.g., 71.578 becomes 72).

   (b) Answer the questions above.

The command for provide for this assignment is:

```
provide comp135 hw4 file1 file2 .. fileN
```

This assignment is due on **Wed., Feb. 12 at 11:59 pm**.