**Machine Learning PS1: Perceptron**

Template Write-Up

**NOTES:**

**X** = E-mail threshold for a word to be considered a feature (word has to show up in X e-mails)

**N** = The size of your training set (it’s how many e-mails in *spam\_train.txt* you’ll use to train)

You should start with **X = 20** and **N = 4000**. Some questions ask you to vary these numbers and then explain/show how your program’s results change.

Explain in a couple of sentences why measuring the performance of your perceptron classifier would be difficult if you do not create a validation set (size = **5000 – N**) from your training data.

|  |
| --- |
| Our model assumes that we can linearly separate the training data, which means for the training data we will always get an error rate of 0. We need the validation data, which is not included in the training, to emulate test conditions and to measure the error rate/performance. |

How many passes of the data did your implementation make before it converged? How many total mistakes did the algorithm make before convergence?

|  |
| --- |
| 9 passes 415 mistakes |

What is your error rate with the validation set? (You just need to enter a percentage)

|  |
| --- |
| 1.9 % (X = 20, N = 4000) |

Which 12 words are most correlated with spam? That is, which 12 “features” have the most *positive* weights? Provide a screenshot or copy/paste of your program output.

|  |
| --- |
| \*This printed 12+ features because it found all features with the same weight  positive weights: access ever market offer will am our deathtospamdeathtospamdeathtospam guarante internet these pleas remov click sight |

Which 12 words are least correlated with spam? That is, which 12 “features” have the most *negative* weights? Provide a screenshot or copy/paste of your program output.

|  |
| --- |
| \*This printed 12+ features because it found all features with the same weight  negative weights: wrote prefer i reserv copyright server there run said still url but date digit on re seem sinc which |

Vary N = 100, 200, 400, 800, 2000, 4000. Provide a plot of the validation error percentage as a function of N.

|  |
| --- |
|  |

Vary N = 100, 200, 400, 800, 2000, 4000. Provide a plot of the number of perceptron algorithm passes as a function of N.

|  |
| --- |
| For N=2000, it exceeded the maximum number of passes I allowed. |

Keep N=4000 constant, and vary the value of X. Try X = 30, X = 40, and any other values of X you’d like. What do you think is the optimal value of X for your perceptron configuration? (Optimal here means lowest validation set error percentage)

|  |
| --- |
| X = 24 is the optimal configuration when N = 4000, with a 1.2% error. |

Now use your best configuration on all of *spam\_train.txt* (N=5000, X=whatever you found to work well). Train with it, and report your error percentage on *spam\_test.txt*

|  |
| --- |
| 1.8% |

Try setting X=1200. How many features do you have in your model? What happens when the perceptron runs? Can you explain what’s going on, and why it happens?

|  |
| --- |
| 51 features that consist mostly of meaningless words such as 'an', 'at', 'on'. The perceptron algorithm does not terminate even after 10,000 iterations and the trained perceptron has high error rate of around 10~12%. Because the threshold (X) is high, the perceptron only selects features that are common to a lot of sentences, whether they be spammy or not. With such set of common words as the standard of classification, it becomes harder to correctly classify emails, and hence the high error rate. Also, because the number of features being considered is very low, there will probably exist emails that have the same feature vector but different labels. This means the algorithm will not be able to converge on a w such that y\*(w·x) > 0 for all x and y, leading to the infinitely many iterations. |

Why do we need a training set, validation set, and test set? One or two sentences on the purpose of each is fine.

|  |
| --- |
| The training set is for calculating the weight of each feature.  The validation set is there to make sure we don’t overfit for the training data and also make sure that we get the minimum error rate. It is like an emulation of the test set, but we can see how well our training worked out.  The test set is the actual data that we are classifying and producing expected labels for. |