Machine Learning PS1: Perceptron

Template Write-Up

Xiangci Li xl1066

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.

The condition of terminating perceptron_train() is there is no update in a single pass, i.e. the error on training set is 0. Without validation set, there is no handy dataset to test the performance of the trained model (test if the model is overfitting the training set).

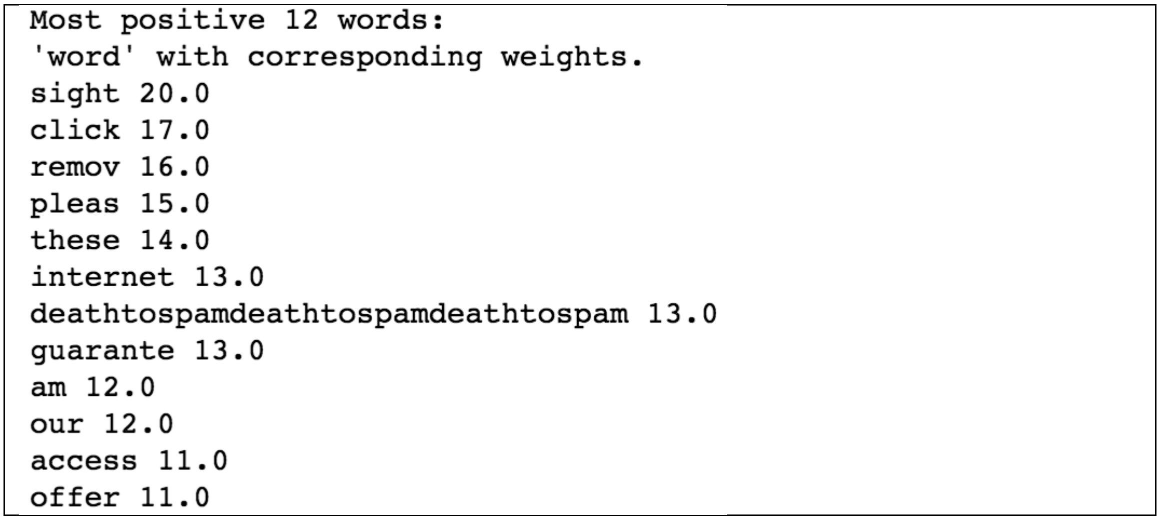
How many passes of the data did your implementation make before it converged? How many total mistakes did the algorithm make before convergence?

Iter = 9, k = 415

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

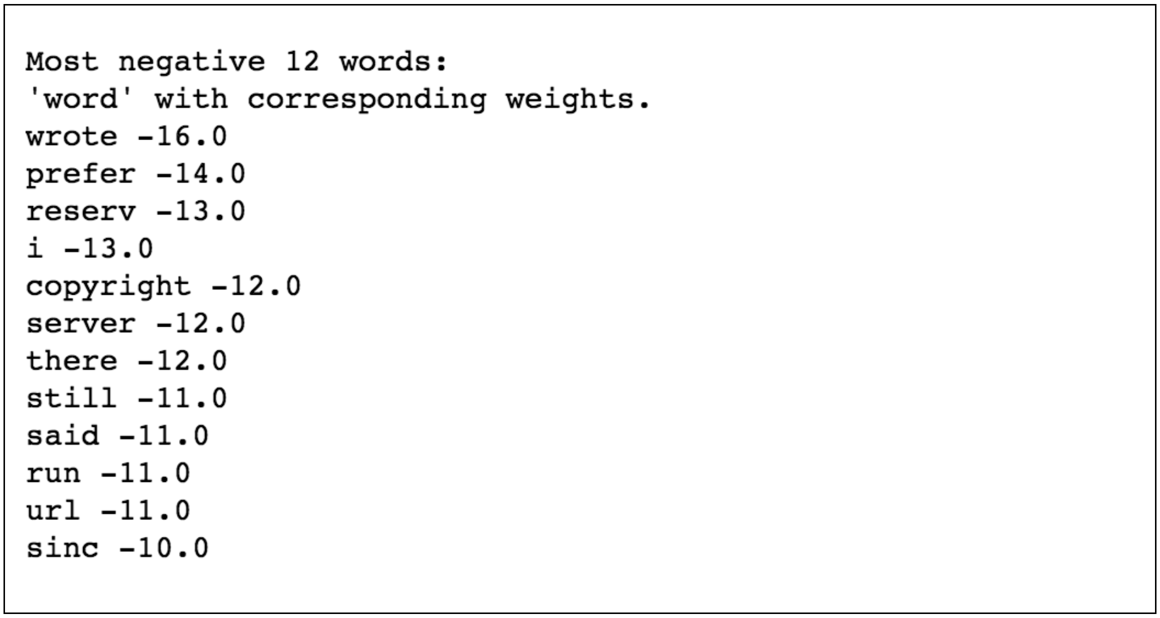
1.9% when there is no max_pass limit.
1.6% when the max_pass is 5.

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.



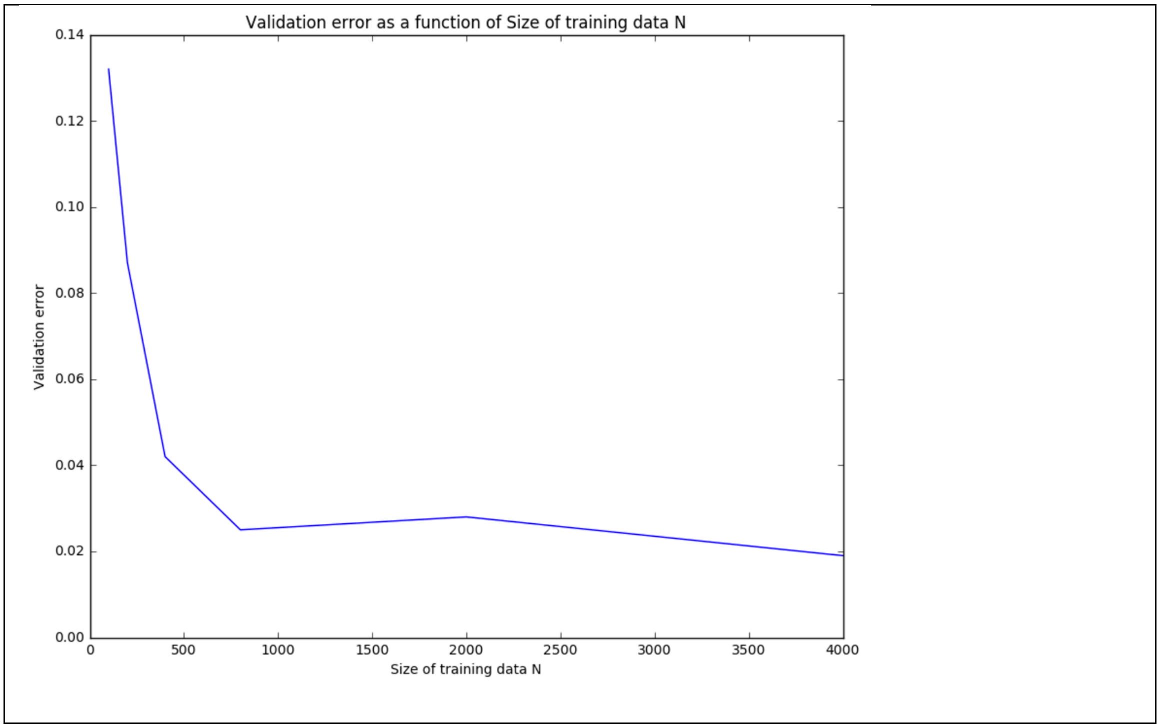
```
Most positive 12 words:
'word' with corresponding weights.
sight 20.0
click 17.0
remov 16.0
pleas 15.0
these 14.0
internet 13.0
deathtospamdeathtospamdeathtospam 13.0
guarante 13.0
am 12.0
our 12.0
access 11.0
offer 11.0
```

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.

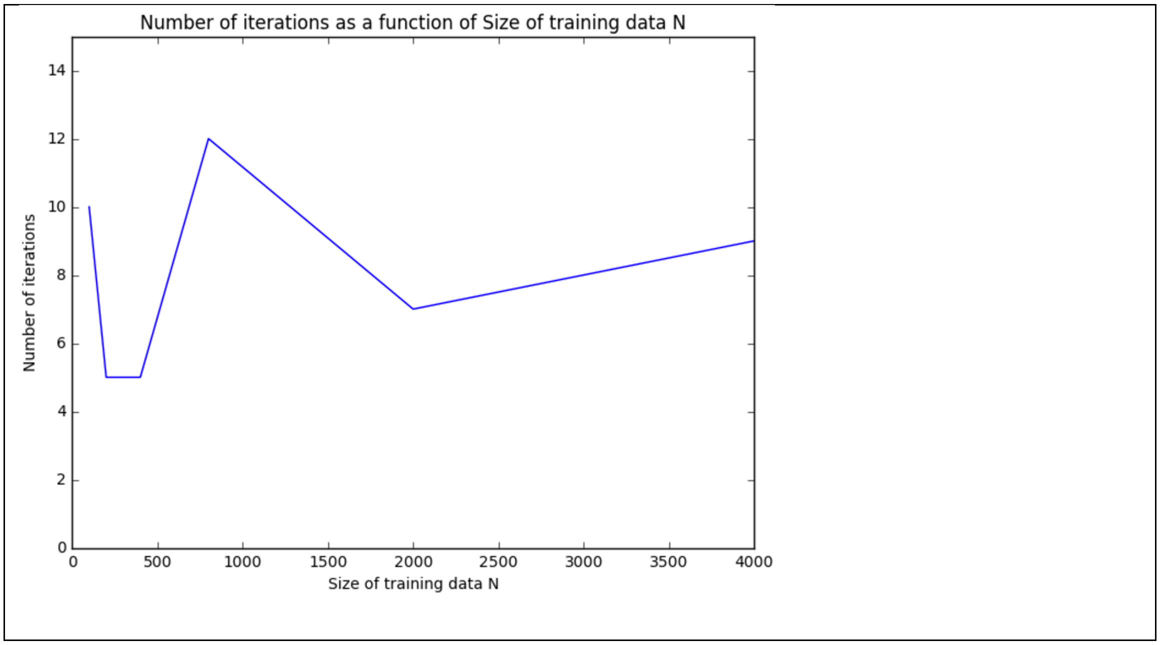


```
Most negative 12 words:
'word' with corresponding weights.
wrote -16.0
prefer -14.0
reserv -13.0
i -13.0
copyright -12.0
server -12.0
there -12.0
still -11.0
said -11.0
run -11.0
url -11.0
sinc -10.0
```

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.



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=15, validation error = 2.0%
X=20, validation error = 1.6%
X=25, validation error = 1.6%
X=30, validation error = 1.5%
X=35, validation error = 1.9%
X=40, validation error = 1.8%

So X=30 is the optimal.

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*

Error = 1.4%

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?

There are 70 features. The validation error is more than 10%. The algorithm does not converge even after 100 passes. The reason is there are too few features so that some emails with the same feature vectors can be either spam or non-spam, i.e. they are not linearly separable.

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

Validation set serves as a "test set" of the model trained by training set. Modification of hyper-parameters are made based on validation performance. In this validation set also contributes to the optimization of the model, i.e. the model is also fitting the validation set. However, test set must be new to the model which means completely unseen by the model to test the model's performance without any bias. Therefore, validation set is necessary.