(P25) Spam Detection with Machine Learning

Ryan Oglesby
Machine Learning 472/572
University of Oregon

Abstract

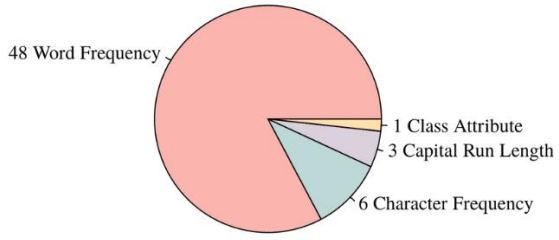
This document evaluates the effectiveness of K-Nearest Neighbor algorithm in the classification of spam emails. I will explore different distance metrics and use K-Fold cross validation procedures to determine the best hyperparameters to test against unseen data. I will compare the best performing hyper parameters against a single neuron neural network. I also attempt to improve accuracy of K-NN by reducing the number of dimensions using a heuristic.

1 The Problem

Spam is a broad subject, there are multiple sources of spam that range from physical to digital. Concerning digital spam, mostly in email, we can use Machine Learning classification algorithms to try and make a spam filter to block common spam. This would work especially well with personal emails such as work emails, such as the dataset used in this evaluation.

2 The Dataset

The data being used comes from UC Irvine Machine Learning Repository. There are 4601 total emails in the spam database, 39.4% (1813) of the emails are classified as spam.



For each email, there are 57 features which are distributed as: 48 features of word frequencies, such as “internet”, “money”, and “George”. “George” is common because this dataset is from the work emails of some person named George. 6 instances of character frequency, and 3 instances of capital run lengths (longest capital run length, average capital run length, number of capital run occurrences).

3 Transforming the Data

Since some features range from 0-1 to 0-4000+, I perform normalization on the data to prepare it for training and testing with our models. After normalizing the data, i split up the entire dataset by taking 20% of the data as random, and save it as unseen testing data, and with the remaining 80% data I perform 5-fold cross validation to try and find the best hyperparameters k for Nearest Neighbor. I also test with three different distance metrics: Manhattan (L1), Euclidean (L2) and Cosine Similarity.

4 K-Fold Validation Results

I perform 5-fold validation with each distance metric and different K-Values(1,3,5,10,15) on the same data per evaluation, to ensure that each distance metric and k value has the same data to work with and no bias/disadvantage.

The results between L1, L2 and Cosine Similarity during cross fold validation performed nearly the

same in all cases of K=1,3,5,10,15 with differences of less than <.01%.

Training	K=1	K=3	K=5	K=10	K=15
Manhattan	.8558	.8627	.8591	.8591	.8591
L2	.8528	.8555	.8557	.8557	.8557
Cosine	.8546	.8622	.8584	.8584	.8584

Since the folds were split up into 5 cross fold splits, the amount of data to pull from most likely affected the K-10 and K-15 not being able to get a different result than K-5.

5 Testing K-NN

After 5-Fold cross validation, the best performing distance metric was determined to be Manhattan, using a K-value of 3. After getting this information, I now take the entire training data set, and use this to train on with this new determined hyperparameter and attempt to classify the unseen 20% test data that I stored before. I also test the other hyperparameters and distance metrics with the same training and testing set to evaluate their performances in compared to the best predicted in cross validation.

Testing	K=1	K=3	K=5	K=10	K=15
Manhattan	.9045	.9058	.9064	.9061	.8956
L2	.9047	.9003	.9005	.8954	.8873
Cosine	.9055	.9030	.9026	.8940	.8952

As seen, the test data is evaluated much better, 5%+ over the training data, and the hyperparameters K-10 and K-15 now give different results as they have more examples to see and compare to.

Although the best performing distance metric is Manhattan L1, the best K value is 5-NN, not 1-NN. This would make sense since the amount of training data being used in the total training set is 5x what a single fold gets to see during cross-fold validation.

6 Improving K-NN (attempting)

After seeing the results, I try to implement a simple heuristic to improve the performance of K-NN on this spam data set. The idea is to go through each email, and for each feature we add or subtract the value to the feature weight, so we would have 57x feature weights – one for each attribute. The idea is a spam email would make feature weights go more negative and a good email would make feature weights more positive, after going through the entire dataset, we have a list of positive and negative "weights".

```
heuristics = [0.0 * len(features)]

for i in range(len(dataset)):
        for j in range(len(features)):
                If dataset[i][-1] == SPAM:
                        heuristics[j] -= dataset[i][j]
                else:
                        heuristics[j] += dataset[i][j]
return heuristics
```

I then take the values and remove them until the number of total attributes is 20, as I learned this is where K-NN starts to have problems. I take away the features in the middle with values closer to zero, as those values are less distinguishable vs other features when recognizing spam, or at least that is the idea.

After retraining/testing with the new heuristic of only 20 features, the accuracy did not improve, but it did not fall(significantly) either. The average Manhattan ACC with 57 features was .9036% and the average Manhattan ACC with 20 features using the heuristic was .8899%. But the time it took to evaluate the dataset and test decreased by **%55** which I would say would almost be worth the consideration of losing about a 1% accuracy. So, this heuristic only improves test time, at the cost of slight accuracy lost.

I also attempted to remove/alter outliers from the data. The results were nothing different from the original test, with differences of <.0015%, which could be attributed to the randomness of the data when generating the splits and test data.

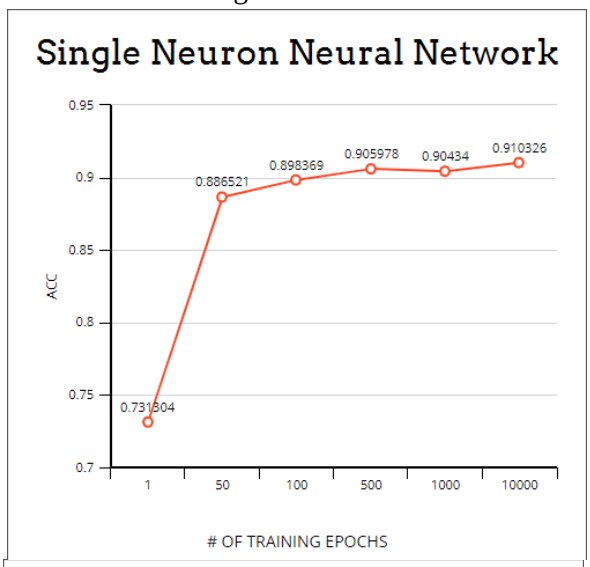
7 Neural Network vs Spam

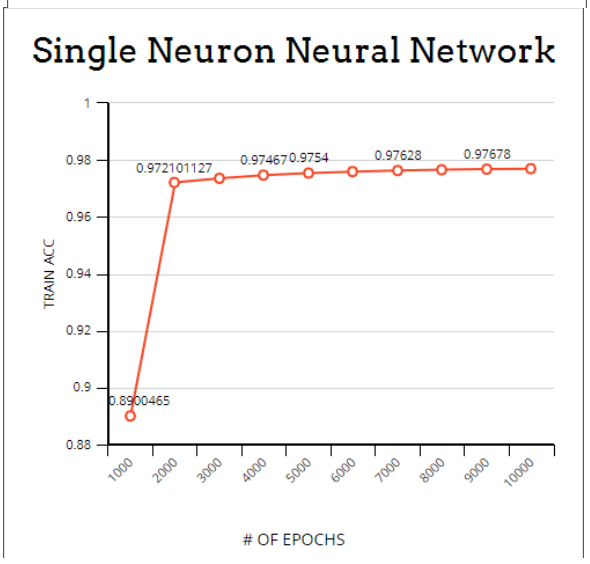
After testing K-NN towards the dataset, I then created a single neuron Neural Network, that performs with a learning rate of .1 and performs up to 10000 iterations.

Performing 5-Fold validation with the Neural Network gave similar results to K-NN. Roughly 85-86% average accuracy on the training folds.

The Training accuracy of the Neural Network surpasses the K-NN after about 1500 iterations and continues to rise to about 97% classification accuracy when classifying email with a >50% confidence rate.

On testing data, the accuracy after 1000 epochs is 90% and after 10,000 the accuracy improves to 91%. The time taken to train and test is much, much faster than using K-NN, which is to be noted as an advantage.





8 Comparison and Conclusion

The results showed that each model performs well enough at classifying spam email, although realistically, you would not want to have to run K-NN every time you get an email with as many features as this, as the time it takes to compare is suboptimal compared to having a neural network trained a single time that can then be used again without having to reuse the old data like K-NN has to.

If, however, you do decide to use a k nearest neighbors' algorithm for classifying spam with a larger number of features (50+), the most optimal hyperparameters would be to use the Manhattan distance with a K value of 5. An alpha of .1 with 10,000 iterations seemed to be a good input for the single neuron neural network for a dataset of this size.

9 References

CIS 472/572 Intro to Machine Learning (K-NN references) https://classes.cs.uoregon.edu/19S/cis472/notes/instLearn.pdf

Neural network reference to help build my neural network model

https://www.geeksforgeeks.org/single-neuron-neural-network-python/