1 Introduction

With the rise of technology over the last several decades, spam detection has become an important machine learning problem. Nowadays abundance of labeled email data allows to build automatic systems effectively detecting spam. The majority of these systems are based on supervised machine learning classifiers. Even though some unsupervised systems exist as well, they are much less popular due to the lower accuracy in comparison to the supervised systems. However, unsupervised systems have the advantage over the supervised systems that they do not need any labeled data for training. This may be a very important factor in the cases when no labeled data is available. In my project, I examined the performance of both supervised and unsupervised neural networks for classifying spam data. Specifically, I implemented two layer back-propagation network and competitive network and compared their performance on the same spam dataset. As will be shown in this report, unsurprisingly two layer back-propagation network significantly outperformed competitive network in this particular spam classification task.

2 Dataset

To compare the performance between back-propagation and competitive networks I used spam dataset that I acquired through my machine learning course, which I took last winter. The dataset contains 4601 data instances, each of which corresponds to a unique email message. Each data instance consists of 48 binary features. Each feature refers to a specific word from the vocabulary, which was constructed using the entire set of emails. Every binary feature represents whether a given word occurs in a particular email (value equal to 1) or not (value equal to 0). Each data instance also contains a binary label, which has a value of 1 in the case when given email is a spam, and 0 otherwise.

For this particular task I used 400 data instances for training, and the rest for testing. I divided the dataset in this way to ensure optimal balance between algorithm's ability to generalize after the training stage and the computational cost of the training.
3 Methods

3.1 Back-Propagation Network

For my supervised method I implemented 2 layer back-propagation network with 96 hidden units (twice the size of dimensionality). In the training procedure I used the learning rate of 0.1. As typical in multi-layer network, my performance rule was defined as a sigmoid function

\[ s(x) = \frac{1}{1 + e^{-\sum_i w_{ij}x_i}} \]

I used the following formula as my learning rule for the output layer:

\[ \Delta w_{ij} = k(t_j - y_j)s'(x)x_i \]

where \( k \) refers to the learning rate, \( t_j \) and \( y_j \) to the given and predicted labels respectively, \( s'(x) \) to the derivative of a sigmoid function and \( x_i \) depicts the input.

For the hidden unit learning rule I utilized the following expression:

\[ \Delta w_{ij} = k(\sum_k \delta_k w_{jk})s'(x)x_i \]

where the variable \( \delta \) refers to the errors coming from the second layer (not from the target outputs).

3.2 Competitive Network

To implement competitive network I used hidden layer of size 48 \( \times \) 2 (number of dimensions \( \times \) number of desired clusters). Similarly to my back-propagation network I set learning rate to be 0.1 and ran the training procedure for 200 iterations.

For the learning formula I utilized the following expression (for the 'winning' neurons only):

\[ \Delta w_{ij} = k(x_i - w_{ij}) \]

whereas for my performance rule I employed:

\[ y_j(x) = \begin{cases} 1 & \text{if } \sum_i w_{ij}x_i \text{ is max} \\ 0 & \text{otherwise} \end{cases} \]

4 Results

After running both algorithms on the same dataset, the results clearly suggest that supervised back-propagation network significantly outperforms unsupervised competitive network. These results are illustrated in Figure 1.
Figure 1: Error rates when using back-propagation network and competitive network respectively.

Figure 2 illustrates how training error decreases throughout back-propagation learning procedure. Just as expected at each iteration we are advancing towards some local minimum until we reach it. As illustrated in the figure, in this particular case we have reached the local minimum after $\approx 90$ iterations.

Figure 2: Training error behavior for back-propagation network learning procedure.

5 Conclusions

As the results suggest, by using back-propagation network we were able to achieve pretty impressive accuracy rate for detecting spam emails. Competitive network, on the other hand, had barely outperformed random guessing strategy.
However, such a behavior is not surprising because competitive network does not use any labeled data in its learning mechanism but instead tries to discover general structure of the data, which is obviously a much more difficult task.

To improve current results even further it would be possible to design better features for our dataset. One way to do that would be to expand the dictionary to include more words as our features. Another way would be to use more complex features to represent the text (maybe tf-idf weights, or features produced by Google ‘word2vec’ deep-learning toolbox).

In addition, to enhance the performance of our classifiers we would need to tune all of the parameters such as size of the network, learning rate, etc.

However, overall even the current results for back-propagation network are really good so the suggested changes may only have a slight effect on a classification accuracy.