Intelligent junk mail detection using neural networks

Michael Vinther mv@logicnet.dk

June 26, 2002

1 Introduction

In this paper I will describe an automatic method for detecting junk or spam e-mail using artificial neural networks. Other methods for detecting unwanted mail are typically based on lists of known IP-addresses, senders or e-mail subjects. These lists needs frequent updating, and a lot of junk mail will not be detected because the spammers frequently change sender address (often faked) and use different mail servers and hence different IP-addresses. The algorithm presented here looks at the actual words in the text, and, to some extend, the structure of the mail. Training and updating of the rules can be done automatically with very little help from the user.

2 The algorithm

2.1 Building a vocabulary

My theory was that by looking at which words are used in the message body and the “from:”, “to:” and “subject:” fields, it would be possibly to distinguish junk mail from “good” mail. A word is here defined as any sequence of letters, meaning that e.g. names, servers and HTML tags will be included. The order of the words, where and how many times they appear are not used in this study, and only words from 3 to 15 letters are included.

Example

This is part of an advertisement sent to me:

```
From: "gay Verrilli" <mqurheafbg@hotmail.com>
To: descartes@mailme.dk
Subject: forward from Cohasset
Date: Mon, 14 Jan 2002 09:26:35
Mime-Version: 1.0
Content-Type: text/plain; format=flowed
Message-ID: <F971moUS2c1FpDK2QTH00021b64@hotmail.com>

    Hi, do You want some fresh online porn OR fresh DVD videos?

        http://vip.adultdomains.com/?fp=9077129&fs=adultxxxx

                            Check it now!
```

The words found in this e-mail are: *cohasset dvd verrilli you adultdomains check com descartes forward fresh from gay hotmail http mailme mqurheafbg now online porn some verrilli videos vip want.*

The input to the neural network is a list of words present in a mail. To limit the size of the network, a vocabulary of a few hundreds words is build based on statistics from a training set of both good and junk mails. The first step is to build a list for each category containing all words found and the probability of finding that word in a mail from that category.

Example

This table contains the first 24 entries from the list of words in junk mails sorted by probability. It shows that 92.4% of the mails contain the word “com”, making it the most common word in junk mails. This list is made from examining 186 junk mails.

92.4%	com	62.9%	html	52.1%	size
83.3%	you	61.8%	http	50.5%	our
82.2%	the	60.2%	with	50.0%	font
77.4%	and	57.5%	please	49.4%	email
76.3%	your	55.9%	href	46.7%	here
69.8%	for	53.7%	body	45.6%	remove
68.2%	from	53.7%	are	45.1%	will
65.0%	this	53.2%	click	45.1%	that

The next step is to select the words from the two lists, which should make up the vocabulary of the network. I did that by creating a combined list with absolute differences in probability: For all words, *w*, in both lists, the difference

$$d_w = |p_{w,junk} - p_{w,good}| \qquad (1)$$

was computed. If a word is only found in one category, the probability of that word being in a mail from the other category was set to zero. The new list is sorted by d_w, and the e.g. 400 words with the largest values are selected as the vocabulary. The advantage of this approach is that the vocabulary will consist of words, which are both very common in at least one of the categories, and which imply what category the message belongs to. One could also just have selected a number of the most common words from both lists, but that would include a lot words which are just as common in good mails as in junk mails, and hence does not tell anything about what category the mail belongs to.

Junk mails are usually impersonal and does not contain the name of the receiver, whereas other mails often do. Therefore the first word in the vocabulary will often be the name of the receiver, and the training of the junk mail detector will be specific for this user.

2.2 Neural network training

The neural network used is a standard non-linear feed-forward network with the sigmoid activation function

$$a(x) = \frac{1}{1+e^{-x}} \tag{2}$$

at both the hidden and output neurons. This activation function will produce an output in the range]0 ; 1[. A network with one input for each word in the vocabulary, a single hidden layer and one output neuron is constructed (see Figure 1). The input and hidden layer contain a bias neuron with a constant activity of 1, which is not shown in the figure.

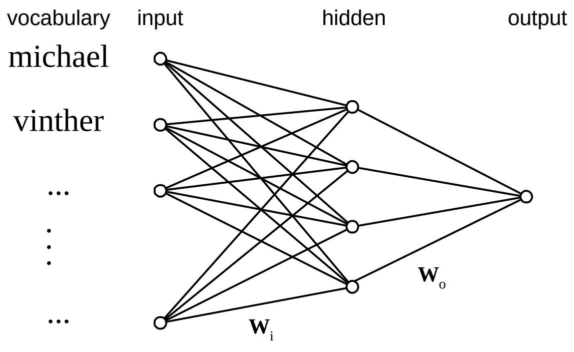

Figure 1: Neural network structure.

When presenting a mail to the network, all inputs corresponding to words found in the message are set to 1, and all other inputs are set to 0. The mail is classified as junk if the output value is above 0.5.

When training the network, the desired output was set to 0.1 for the good mails and 0.9 for the junk mails. The number of good mails and the number junk mails presented to the network should not differ too much. If the number of mails from one of the categories is much larger than that of the other, the network might learn to always answer yes or always answer no, simply because that would be true most of the time for a bad training set.

A gradient descent training algorithm using back propagation of errors was applied for optimizing the weights in the network. Optimization was done by minimizing output mean square error:

$$\min_{\mathbf{w}} \frac{1}{N_{train}} \sum_{i=1}^{N_{train}} (O_i - O_{desired,i})^2 \qquad (3)$$

2.3 Interpretation of the processing in the network

A network with this structure can make rules like “it is a junk mail if *word1* and *word2* is present but not *word3*” and similar logical rules. Because it uses “floating point logic”, it can also make other kind of rules, which are harder to interpret.

The number of rules that can be created is connected to the number of hidden neurons. If there are to few hidden neurons, the set of rules will be to small to cover many different mails, and the training set cannot be learned. If, however, the number of hidden neurons is too big, the ability to generalize will be lost. In the extreme case a rule will be made for each mail in the training set, which will result in perfect answers for these mails but poor answers for any new mails. Automatic methods to optimize the structure of neural networks exists, e.g. optimal brain damage, but have not been used in my experiments. A simple rule could be that there should be just enough hidden neurons to get good results with the training data set.

3 Experiments

3.1 Test data

The good mails used were my personal mail (both sent to and received) most in Danish, but also some in English. The junk mails were in English, some of them sent to me, and some of them downloaded from The Great Spam Archive[1].

I used a training set of 168 good mails and 186 junk mails. The validation set contained 204 good mails and 337 junk mails. The mails included in the two sets were selected randomly from all the available test data.

3.2 Results

From the training data set I selected the 400 words with largest d_w value as vocabulary, and the neural network should therefore have 400 input neurons. The words had d_w values in the range from 8% to 81%. Assuming that about 20 rules would be sufficient, I set the number of hidden neurons to 20. That is much less that the total number of mails in the training set (168+186 = 354) so there should be no risk of over-training.

400 input, 20 hidden and 1 output neurons give a network with a total of $401 \cdot 20+21 = 8041$ parameters including biases. The training required about 200 gradient descent training iterations, which was done in a few minutes on a standard PC.

[1] The Great Spam Archive: http://www.annexia.org/spam/

After the training, 100% of the training set and 94% of the validation set was classified correct as either junk or good mail. In the validation set, 6.7% of the good mails and 5.6% of the junk mails were misclassified. The good mails, which was misclassified, were not personal mails, but standard mails in both English and Danish from robots as response to registration at web sites and similar.

4 Conclusion

I have demonstrated that neural networks can indeed be used to separate wanted mails from junk mail. The method used was very simple with lot of room for improvement, and in spite of that the network answered correct for 94% of the mails in my validation set, which was not included in the training in any way. It is important to note that training is done for a specific user, so that the network recognizes the personal mail typically received. If all personal mails were in the same language as the junk mail, the result might not be as good, but I have not been able to test that because of lack of test data.

Both the vocabulary size of 400 words and the number of hidden neurons used were my first guesses at some good values, and the gradient descent is the simplest training algorithm for neural networks. Selection of words from the mails, vocabulary, network structure and training algorithm could all be improved in several ways to give a better result.

A good practical implementation would probably combine automatic neural network training with a set of user-defined rules like “never reject mail from girlfriend@somewhere.dk” to make sure that important personal mail gets through.

5 Literature

Neural Networks for Pattern Recognition
Christopher M. Bishop
Oxford University Press, 1995

Linjedeling med et neuralt netværk
Søren Brunak, DTU, and Benny Laustrup, The Niels Bohr Institute
1989

Introduction to Artificial Neural Networks
Jan Larsen
IMM, DTU, November 1999