Deep Belief Networks for Spam Filtering

Grigorios Tzortzis and Aristidis Likas
Department of Computer Science, University of Ioannina
GR 45110, Ioannina, Greece
E-mail: {gtzortzi, arly}@cs.uoi.gr

Abstract

This paper proposes a novel approach for spam filtering based on the use of Deep Belief Networks (DBNs). In contrast to conventional feedfoward neural networks having one or two hidden layers, DBNs are feedforward neural networks with many hidden layers. Until recently it was not clear how to initialize the weights of deep neural networks, which resulted in poor solutions with low generalization capabilities. A greedy layer-wise unsupervised algorithm was recently proposed to tackle this problem with successful results. In this work we present a methodology for spam detection based on DBNs and evaluate its performance on three widely used datasets. We also compare our method to Support Vector Machines (SVMs) which is the state-of-the-art method for spam filtering in terms of classification performance. Our experiments indicate that using DBNs to filter spam e-mails is a viable methodology, since they achieve similar or even better performance than SVMs on all three datasets.

1. Introduction

A characteristic of the last decade is the huge growth of the spam phenomenon. According to recent surveys, 60% of all e-mail traffic is spam. In this way a great amount of bandwidth is wasted and the e-mail systems are overloaded. Due to the above serious problems, measures must be taken to deal with the spam phenomenon. The best such measure has proven to be spam filtering.

There are two general approaches to mail filtering: *knowledge engineering* (*KE*) and *machine learning* (*ML*). Spam filters based on the first approach usually exploit a set of predefined and user-defined rules. Rules of this kind try to identify in a message typical characteristics of spam. However, it has been proved in practice that this technique suffers from poor generalization.

The machine learning approach consists of the automatic construction of a classifier based on a training set (usually the e-mails of the user). Experiments with machine learning classifiers have shown that they achieve higher generalization compared to rule based filters. A variety of classification methods have been proposed for the spam detection task. The most widely acceptable methods are the Naïve Bayes classifier and the Support Vector Machines (SVMs) (e.g. [1], [2], [3], [4], [5]).

In this paper we propose the use of a *Deep Belief Network* (*DBN*) to tackle the spam problem. DBNs are feedforward neural networks which have a *deep architecture* i.e. they consist of many hidden layers. Until recently the main obstacle in using DBNs was the difficulty in training such deep networks. Usually gradient based optimization gets stuck in poor local minima due to the random initialization of the network weights. Hinton et al. [6] introduced a new greedy layer-wise unsupervised algorithm for initializing DBNs based on the use of *Restricted Boltzmann Machines* (*RBMs*). This algorithm provides a sensible initialization of the network weights which are subsequently fine tuned using a gradient based algorithm such as gradient descent. In a study reported in [7], the effectiveness of the DBN initialization method [6] is justified on several datasets.

In this work, we propose a machine learning approach for identifying spam mails based on DBNs and evaluate its performance on three well-studied spam detection datasets. We also compare our method with the SVM classifier [9] which constitutes the state-of-the-art method in spam detection. The experimental results indicate that our method based on DBNs provides similar or even better performance than SVMs in spam filtering and should be seriously considered as an effective alternative to SVMs.

The rest of this paper is organized as follows: in Section 2 we present the details of training a DBN. Our experimental results and performance measures are presented in Section 3 together with an analysis of the datasets used. Finally Section 4 concludes this work.

2. Deep belief networks

A DBN is a feedforward neural network with a deep architecture, i.e. with many hidden layers. An example of a DBN for classification is shown in Figure 1. It consists of an input layer which contains the input units (called

visible units), a number L of hidden layers and finally an output layer which has one unit for each class considered. The parameters of a DBN are the weights $\mathbf{W}^{(j)}$ between the units of layers j-1 and j and the biases $\mathbf{b}^{(j)}$ of layer j (note that there are no biases in the input layer).

One of the main problems for training deep neural network architectures is how to initialize these parameters. Random initialization causes optimization algorithms to find poor local minima of the error function resulting in low generalization. Hinton et al. [6] introduced a new algorithm to solve the above problem based on the training of a *sequence of RBMs*. An RBM is a two layer recurrent neural network in which stochastic binary inputs are connected to stochastic binary outputs using symmetrically weighted connections. The first layer corresponds to inputs (visible units **v**) and the second layer to the hidden units **h** of the RBM. After RBM training, hidden units can be considered to act as feature detectors, i.e. they form a compact representation of the input vector. An example of an RBM is given in Figure 2.

An RBM is characterized by the *energy function* defined as:

$$E(\mathbf{v},\mathbf{h}) = -\mathbf{h}^{\mathrm{T}}\mathbf{W}\mathbf{v} - \mathbf{b}^{\mathrm{T}}\mathbf{v} - \mathbf{c}^{\mathrm{T}}\mathbf{h} \quad (1)$$

where $\mathbf{W}$ is the weight matrix and $\mathbf{b}$, $\mathbf{c}$ are the bias vectors for visible and hidden layers respectively. The layer to layer conditional distributions are:

$$P(\mathrm{v_i} = 1 \mid \mathbf{h}) = \sigma(\mathrm{b_i} + \sum_{\mathrm{j}} \mathrm{W_{ji}} \mathrm{h_j}) \quad (2)$$

$$P(\mathrm{h_j} = 1 \mid \mathbf{v}) = \sigma(\mathrm{c_j} + \sum_{\mathrm{i}} \mathrm{W_{ji}}\, \mathrm{v_i}) \quad (3)$$

where $\sigma(a) = 1/(1 + e^{-a})$ is the logistic function providing outputs in the (0, 1) range. By sampling using the above probabilities the output (0 or 1) of an RBM unit is determined.

Next we describe how to train an RBM and how it is used in the construction of a DBN. First of all we must emphasize that RBM training is unsupervised. Given a training example, we ignore its class label and we propagate it stochastically through the RBM. The outputs of the hidden units follow the conditional distribution specified in equation (3). We then sample from this distribution thus producing a binary vector. This vector is propagated in the opposite direction through the RBM (from hidden units to visible units using (2)) which results in a "confabulation" (reconstruction) of the original input data. Finally the state of the hidden units is updated by propagating this confabulation through the RBM.

The above procedure is performed repeatedly for all the examples of the training set and then the update of the parameters takes place as follows [8]:

$$\Delta \mathrm{W_{ji}} = \eta(\langle \mathrm{v_i}\, \mathrm{h_j} \rangle_{\mathrm{data}} - \langle \mathrm{v_i}\, \mathrm{h_j} \rangle_{\mathrm{confabulation}}) \quad (4)$$

$$\Delta \mathrm{b_i} = \eta(\langle \mathrm{v_i} \rangle_{\mathrm{data}} - \langle \mathrm{v_i} \rangle_{\mathrm{confabulation}}) \quad (5)$$

$$\Delta \mathrm{c_j} = \eta(\langle \mathrm{h_j} \rangle_{\mathrm{data}} - \langle \mathrm{h_j} \rangle_{\mathrm{confabulation}}) \quad (6)$$

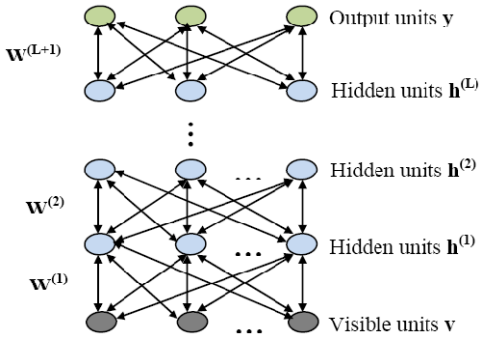

Figure 1. DBN architecture with L hidden layers.

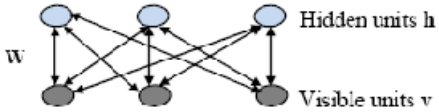

Figure 2. RBM architecture.

where $\eta > 0$ is the learning rate and $\langle \mathrm{v_i}\, \mathrm{h_j} \rangle_{\mathrm{data}}$ denotes the fraction of times that visible unit *i* and hidden unit *j* are on together when the original data is propagated through the RBM. Similar is the meaning of the notation for the rest of the parameters' update formulas. We can repeat this method for a defined number of epochs or until the reconstruction error of the original data becomes small, i.e. the confabulation is very similar to the data.

To construct a DBN we train sequentially as many RBMs as the number of hidden layers in the DBN, i.e. for a DBN with L hidden layers we have to train L RBMs. These RBMs are placed one on top of the other resulting in a DBN without the output layer. A natural question that arises is which the inputs of each RBM are. For the first RBM, which consists of the DBN's input layer and the first hidden layer, the answer is simply the training set. For the second RBM, which consists of the DBN's first and second hidden layers, the answer is the output of the previous RBM i.e. the activations of its hidden units when they are driven by data, not confabulations. The same holds for the remaining RBMs.

The above procedure is justified by a variational bound [6]. It can be proved that under some general assumptions, the addition of an extra hidden layer improves a lower bound on the log probability that the network assigns to the training set.

After performing this layer-wise algorithm we have obtained a good initialization for the hidden weights and biases of the DBN. It still remains to determine the weights from the last hidden layer to the outputs (as well as the biases of the outputs) and also to fine tune the parameters of all layers together. In order to perform classification, as in spam filtering, we add an output layer with two units, whose weights are randomly initialized, and the DBN is fine tuned with respect to a typical supervised criterion (such as mean square error or cross-entropy).

3. Experimental evaluation

3.1. Testing corpora

The DBN performance is evaluated on three widely used datasets namely LingSpam, EnronSpam (both available at http://www.iit.demokritos.gr/skel/i-config) and SpamAssassin (available at http://www.spamassassin.org/publiccorpus). LingSpam contains 2893 e-mails out of which 2412 are legitimate messages and 481 are spam messages resulting in a 16.6% spam ratio. Legitimate messages come from the archives of the Linguist list and spam messages from the inboxes of its creators. Each e-mail contains the subject and the body only. In our experiments we use the version where stop words are included and no stemming has been performed.

The SpamAssassin corpus contains 4150 legitimate messages and 1897 spam messages which make a total of 6047 messages with a 31.3% spam ratio. These e-mails were collected from public fora or were donated by users and come in row format.

The EnronSpam dataset was recently introduced in [2]. The preprocessed version, which is used in our experiments, contains only the subject and the body of the messages. Here we focus on Enron1 which includes the legitimate messages of one of the six Enron employees contained on the EnronSpam corpus. It has a total of 5172 messages out of which 3672 are legitimate and 1500 are spam resulting in a 29% spam ratio. The spam messages were collected from the inbox of one of its creators.

3.2. Performance measures

To evaluate the DBN and SVM classifiers the following performance measures were considered: *Accuracy* (*Acc*) which measures the percentage of correctly classified messages, *Ham Recall* (*HR*) (resp. *Spam Recall* (*SR*)) which is the percentage of legitimate (resp. spam) messages assigned to the correct category and finally *Ham Precision* (*HP*) (resp. *Spam Precision* (*SP*)) which is the percentage of messages classified as legitimate (resp. spam) that are indeed legitimate (resp. spam). By denoting X→Y the messages of class X that are classified to class Y we have the following definitions where S, L denote the spam and the legitimate class respectively:

$$Acc = \frac{|S\rightarrow S| + |L\rightarrow L|}{|S\rightarrow S| + |L\rightarrow L| + |S\rightarrow L| + |L\rightarrow S|} \quad (7)$$

$$HR = \frac{|L\rightarrow L|}{|L\rightarrow L| + |L\rightarrow S|} \quad (8) \qquad SR = \frac{|S\rightarrow S|}{|S\rightarrow S| + |S\rightarrow L|} \quad (9)$$

$$HP = \frac{|L\rightarrow L|}{|L\rightarrow L| + |S\rightarrow L|} \quad (10) \qquad SP = \frac{|S\rightarrow S|}{|S\rightarrow S| + |L\rightarrow S|} \quad (11)$$

3.3. Experimental results

Next we describe the details for DBN and SVM training as well as the performance results using the three datasets. In what concerns message preprocessing, each message is represented as a vector with length equal to the number of distinct words of the corpus. This set of words is known as the *vocabulary*. The value of a vector's component is the frequency of the corresponding word (as in many other works on spam filtering e.g. [1], [2], [4]). We removed stop words and words appearing in less than two documents in each corpus. We further reduced the vocabulary size by retaining the top k words with respect to information gain score. For LingSpam we set k = 1500, while for SpamAssassin and Enron1 we set k = 1000. Finally on SpamAssassin we removed HTML tags.

All our experiments were performed using 10-fold cross validation. LingSpam is already partitioned into 10 parts by its creators. For the other two datasets, we randomly split the corpus to form 10 partitions in such a way that the spam ratio of the original corpus was retained in each partition.

As in many other SVM approaches to text classification, we used the *cosine* kernel and so there is only one parameter to determine a priori, which is the cost parameter C. We tried several different values for C and kept the one resulting in higher performance. We implemented the SVM classifier using LIBSVM (available at http://www.csie.ntu.edu.tw/~cjlin/libsvm).

In order to apply the DBN for spam detection, one must determine appropriate values for the number of hidden layers and units for each such layer. As suggested in previous work [6], [7], [8], we experimented with 3 and 4 hidden layers in order to get a network with deep architecture. Results indicate similar performance on each dataset so we selected the simpler architecture with 3 hidden layers for the rest of our experiments. By trying some configurations with different number of hidden units, we found that 50-50-200 neurons for the first, second and third layer respectively give the best results. We observed that moderate variations on the number of neurons in each layer did not significantly affect the results. The same was true when we varied the ratio of units between different layers. To emphasize the robustness of the DBN method with respect to architectural variations, we used the same architecture in all experiments reported in this paper, which is a DBN with 3 hidden layers and consists of |vocabulary| input units, 50-50-200 hidden units and 2 output units, one for each class.

The RBMs were trained using binary vectors (indicating the presence or absence of a word in a message) instead of normalized frequency vectors. This is done because RBMs naturally operate on binary input data, and this decision led to improved performance in our experiments. Each RBM was trained for 20 epochs. The fine

tuning of the whole network was performed using conjugate gradients. Note that during fine tuning frequency vectors are used as inputs.

As our aim is to compare DBNs with SVMs on the spam filtering task here we report only the results obtained with the best configuration for each classifier on each dataset. Tables 1, 2 and 3 summarize the average of those results over 10 folds on LingSpam, SpamAssassin and Enron1 respectively.

These results make it clear that DBNs can achieve similar performance to SVMs in spam filtering tasks. Moreover in these experiments DBNs exhibit a slight advantage over SVMs by showing better accuracy on all three datasets. Also the DBN outperforms the SVM against all performance measures on SpamAssassin and for the majority of measures on the other two collections. This is a satisfactory result provided that SVMs are considered state-of-the-art and that DBNs' results may be improved by considering more epochs during training and by addressing the problem of architectural specification in a more systematic way. The above results justify that the algorithm proposed for training DBNs does work in practice and we show that it can be used to filter spam e-mails.

Table 1. DBNs vs. SVMs on LingSpam.

Performance Measure	LingSpam	
	DBN 1500-50-50-200-2	**SVM** C = 1
Accuracy	**99.45%**	99.24%
Spam Recall	**98.54%**	96.67%
Spam Precision	98.2%	**98.74%**
Ham Recall	99.63%	**99.75%**
Ham Precision	**99.71%**	99.35%

Table 2. DBNs vs. SVMs on SpamAssassin.

Performance Measure	SpamAssassin	
	DBN 1000-50-50-200-2	**SVM** C = 10
Accuracy	**97.5%**	97.32%
Spam Recall	**95.51%**	95.24%
Spam Precision	**96.4%**	96.14%
Ham Recall	**98.39%**	98.24%
Ham Precision	**98.02%**	97.89%

Table 3. DBNs vs. SVMs on Enron1.

Performance Measure	Enron1	
	DBN 1000-50-50-200-2	**SVM** C = 1
Accuracy	**97.43%**	96.92%
Spam Recall	96.47%	**97.27%**
Spam Precision	**94.94%**	92.74%
Ham Recall	**97.83%**	96.78%
Ham Precision	98.53%	**98.84%**

4. Conclusions

In this paper we focused on the use of Deep Belief Networks for spam filtering. DBNs have recently gained significant interest for solving practical problems, because an effective methodology for weight initialization has been proposed. We studied the details of DBN training and evaluated the performance of our approach on three well-known e-mail collections. Our results were compared to those achieved by Support Vector Machines. The comparative results indicate that DBNs constitute a viable solution for filtering e-mails.

Although the above results are encouraging provided that it is, to our knowledge, the first attempt to employ DBNs for spam detection, a number of issues could be examined in future work, such as techniques for selecting the number of hidden layers and units for each such layer.

References

[1] I. Androutsopoulos, G. Paliouras, and E. Michelakis, *Learning to Filter Unsolicited Commercial E-Mail*, tech. report 2004/2, NCSR "Demokritos", 2004.

[2] V. Metsis, I. Androutsopoulos, and G. Paliouras, "Spam Filtering with Naive Bayes – Which Naive Bayes?", *3rd Conf. Email and Anti-Spam* (CEAS 2006), Mountain View, California, 2006.

[3] L. Zhang, J. Zhu, and T. Yao, "An Evaluation of Statistical Spam Filtering Techniques", *ACM Trans. Asian Language Information Processing*, vol. 3, no. 4, 2004, pp. 243-269.

[4] J. Hovold, "Naive Bayes Spam Filtering Using Word-Position-Based Attributes", *2nd Conf. Email and Anti-Spam* (CEAS 2005), Stanford, California, 2005.

[5] G. Fumera, I. Pillai, and F. Roli, "Spam Filtering Based on the Analysis of Text Information Embedded into Images", *J. Machine Learning Research*, vol. 7, 2006, pp. 2699-2720.

[6] G.E. Hinton, S. Osindero, and Y. Teh, "A Fast Learning Algorithm for Deep Belief Nets", *Neural Computation*, vol. 18, no. 7, July 2006, pp. 1527-1554.

[7] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy Layer-Wise Training of Deep Networks", *Neural Information Processing Systems* (NIPS 2006), 2006.

[8] G.E. Hinton, and R.R. Salakhutdinov, "Reducing the Dimensionality of Data with Neural Networks", *Science*, vol. 313, July 2006, pp.504-507.

[9] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition", *Data Mining and Knowledge Discovery*, vol. 2, no.2, 1998, pp.121-167.