Pre-Processing With Restricted Boltzmann Machines

Kevin Yan
Department of Statistics
University of Toronto
kevin@utstat.utoronto.edu

Abstract

In this paper, we use a binary Restricted Boltzmann Machines as well as PCA in order to reduce the dimensionality of data for the use in the email(spam) classification task. The focus will be on the use of RBMs for this task, and so the model and some of its uses for this task will be described in depth.

1 Introduction

Spam filtering is something that is, for the most part, automatically done for us these days. People have designed algorithms and systems in order to detect e-mails and messages that are likely to be spam, and separate them from those that are important or desired by the user. In many cases, these algorithms will analyze a particular email looking for the presence of particular keywords that may indicate spam. Instead of looking at each email as a bag of words (i.e.: as a collection of words), it may be useful to try to find some hidden structure/representation of the emails that will allow us to see better performance in the classification task. For this project, we will simply do this by using algorithms for dimensionality reduction, which include PCA as well as the Restricted Boltzmann Machine model. Standard PCA is a non-probabilistic algorithm, which extracts the directions of highest variability in the original input space, and projects the original data points onto a lower dimensional subspace of these high variance directions. Alternatively the Restricted Boltzmann Machine is a undirected probabilistic model, with latent variables which we will assume are the hidden structure/latent representation of the input data. For the most part, we will focus on the use of RBMs for this task. We will perform the data reduction on personal emails collected by Sam Roweis. We will then evaluate the effectiveness of the data reduction by using these features in the spam-filtering task using a logistic regression classifier. The hope is that the reduced representations will perform just as well, if not better than the original data.

2 Restricted Boltzmann Machines

The Restricted Boltzmann Machine is a 2-layer network with latent features (h), visible features (input data, v), and symmetrically weighted connections between each pair of hidden and visible units (W). These weights will indicate how likely a certain hidden unit will be active when a particular visible unit is active (i.e.: a word is present in the email). Notably, in the RBM model, there are no connections between hidden units, and no connections between the visible units as well (See Figure 1). Therefore, because of d-separation, we see that conditioning on the visible units, the conditional distribution over the hidden units factorizes, making it easy to make inferences on the hidden units. In particular, the type of RBM that we will focus on in this paper is binary, meaning that both the hidden (h) and visible units (v), only have binary outcomes (e.g.: 0

or 1).

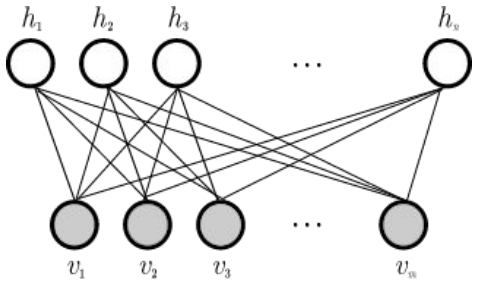

Figure 1: Restricted Boltzmann Machine, grey circles indicate observed variables

The algorithm which we will use to learn the parameters of the RBM is going to Persistent Contrastive Divergence (PCD) [2]. In summary, this is a variant of approximate gradient descent. The likelihood for the model is given in (1), and represents the probability of a particular value of v, the likelihood of an entire data set can be derived by evaluating (1) for each v in the data-set, and summing them up; however it suffices to show the equations for a single input case. It then follows that the log-likelihood can be separated into two parts, which are named the positive, and negative part (2). The derivatives of each of these two parts with respect to a particular parameter w_{ij} can be seen in equation (3). The probability in the positive gradient can easily be calculated as the sigmoid of a linear combination of the visible units, however, the probability in the negative gradient needs to be calculated by marginalizing out all other v *and* h. Thus the calculation for the negative gradient is often intractable if our visible and hidden units are of high dimension. Therefore, in many cases this justifies the use of approximate methods like PCD.

$$P(v) = \frac{1}{Z}\sum_{h} exp(-E(v,h)), \qquad E(v,h) = \sum_{i,j} v_i h_j w_{i,j} + \sum_{i} v_i b_i + \sum_{j} h_j b_j \qquad (1)$$

$$Z = \textit{partition function} = \sum_{v,h} exp(-E(v,h))$$

v are the visible units, *h are hidden units*, w_{ij}*'s are weights of connections between* v_i, h_j

$$logP(V) = \varphi^{+} - \varphi^{-}, \quad \varphi^{+} = log(\sum_{h} exp(-E(v,h))), \quad \varphi^{-} = log(Z) \qquad (2)$$

$$\frac{\delta(\varphi^{+})}{\delta(w_{ij})} = v_i \cdot P(h_j = 1|v) \quad , \quad \frac{\delta(\varphi^{-})}{\delta(w_{ij})} = P(v_i = 1, h_j = 1) \qquad (3)$$

Another very popular algorithm, very similar to PCD is Contrastive Divergence (CD-K) (Hinton 2002). In CD-1, the negative gradient is approximated by one run through of the Gibbs Sampler initiated at a data point v. This often works well, however there is an issue that one iteration will not be enough to properly sample from the true joint distribution. Also, in CD, we initialize the markov chains at data points, which may bias the samples towards sample data. A correction would to increase K, so we take a sample from further into the chain via CD-K. Alternatively, we can use PCD, which has been shown to often outperform CD [3]. In PCD, we initialize S chains, from a collection of our data points. For each update of the parameters, we sample from the same chains starting from where we left off. To make this point more concrete, some rough pseudo-code will be given for a batch learning approach of this process below:

1) *Initialize values of the parameters W*

2) *Initialize S markov chains*, *using Gibbs Sampling*(*in order to sample from the joint*)

Repeat over batches of training data:

3) *Compute the positive gradient using data from the batch*

4) *Compute the negative gradient from the recent values of h,an v sampled from the chains*

5) *Update the parameters W using the difference between the positive and negative gradients*

6) *Update the MCMC via a few iterations through the Gibbs Sampler*

Note also, we do not restart/re-initialize the markov chains for each repetition, hence the name persistent. To justify this process, PCD allows us to better explore the distribution of the model, because we do not have to initialize the chains near the data points, and we keep the chains running throughout the entire algorithm. A concern may be that we may not be correctly sampling from the updated model each time, because we do not update the distributions in the Gibbs sampler (ie: not re-initializing the markov chains). However because we will be moving in the direction of the gradient slowly (by keeping the learning rate small, which is important in PCD), the overall distribution will not deviate too much after each iteration, therefore, our samples will be from a distribution very similar to that of the model's, where step 6 will be used to "catch-up" to the updated distribution [2]. We will now see how all this is used for the main purpose of the project.

3 Experiment

The plan of action for this project is to see the effect of dimensionality reduction of inputs in a spam detection/classification task. The main focus will be on the use of RBMs for this task, in addition however, we will compare the results from the RBM to the results of a non-probabilistic technique, Principal Component Analysis.

3.1 Data Overview

Some information about the data set is that there are 4000 emails in the training set, each with 185 features. Each particular feature corresponds to the existence of a word, or the presence of some other notable feature that the author has deemed useful. For example, some features that he decided to keep were: 'subject_empty','subject_CAPS', 'friday', 'stocks' and etc. Thus each email was represented as a 185 dimensional binary vector. A test set including another 1000 emails was also included with the same 185 features. In addition there were labels assigned to each email, deeming it as spam or not. This was an ideal data set because all the units in the the binary RBM are binary, thus the visible units/our input data, must be a binary vector.

3.2 Process

In the experiment I will reduce the data to 50, and 100 dimensional representations. This will be done by training an RBM with 185 visible units, and only 50 or 100 hidden units. After training the RBM, we can find the most probable hidden representation for each email by computing the MAP estimate of $p(h|v)$, and because of factorization we can just compute the MAP estimate of each h_i, *via* $p(h_i|v)$ individually. Instead of using the associated hidden units however, we can instead use the probabilities $p(h_i|v)$ instead, and so our latent representation will be a 50 or 100 length vector of probabilities (ie: element $k = p(h_k|v)$). This is a better alternative because it would

make the representations more comparable with those given by PCA, which will return vectors with continuous elements.

In order to train the RBM, we will use the BernoulliRBM package in the sklearn toolkit, which uses Persistent Contrastive Divergence, without momentum as the the learning algorithm. It is important to choose the correct learning rate, batch-size and number of iterations to use in training, and so we will use 5-fold cross validation over the training set in order to choose appropriate values for these hyperparameters. For each fold, we will effectively fit the RBM on 4/5ths of the training data, and use the hidden representation as described previously to train a logistic regression classifier; then we will test on on the last 1/5th. We will then choose the settings that perform the best on the validation sets for use in the later parts of the experiment.

Using these hyperparameters, we will then fit the 50 and 100 hidden unit RBMs and use their latent representations as training data for a logistic regression classifier. We will then test the logistic regression classifier on the 1000 emails in the test set. Before that is possible however, I will use the parameters learned by the RBM in order to transform each input vector into its latent representation.

We will repeat the overall procedure again, except with PCA instead of RBMs. We will then compare the best results of the two methods on the test data, as well as with the benchmark (the performance without any dimensionality reduction).

3.3 Layer-by-Layer RBM

Following this I will use the parameters for the best RBM model, including the optimal number of hidden units, and see if training 2 RBMs in succession to reduce the data dimension from 185 to the optimal latent dimension will be more effective than just using one single RBM. In more detail, I will use the latent representations from the first RBM as inputs to the second RBM. A similar idea has been presented in by Hinton and Salakhutdinov [1]. It is also mentioned that if we do not reduce the number of dimensions of the hidden units in this layer-by-layer, approach, adding an additional layer will always result in the improvement of log-likelihood of the data, however this is not necessarily the case for us.

3.4 Classification with RBM

Lastly, another experiment is to perform the classification task using only the RBM itself, as documented in [4] by Swersky et al. We would have to adjust the training data by appending the classifications to the original 185 features to create 186 length vectors as inputs. Then we would train an RBM using these features. Now to classify a new email from the test set, we would choose the class that satisfies the following: $class(v) = argmax_c \, log(\sum_h exp(-E(< v, c >, h)))$, for a particular input v. The complexity of each of these classifications is linear in the number of hidden units [4]. This is because given that we know v, the exponential term will factorize in h, leading to a complexity of $O(K \cdot 2)$. We will compute the free energy of a particular data point on both possible inputs $< v, 0 >$ *and* $< v, 1 >$, and choose the one with the smallest free energy, as the correct class. We will then compare the performance of this with technique with all the ones proposed before.

4 Results

4.1 Cross-Validation

I will begin by performing 5 fold cross validation on the training data in order to find the optimal parameters for the logistic regression learning rate, RBM learning rate, optimal number of hidden

units, and iterations of the training set for training, a batch size of 10 was arbitrarily chosen. In total I had 36 combinations of parameters, of which I performed cross validation on each. It took about 1 hour to complete this process on a dual-core 1.4ghz laptop. After cycling through a few different values of the RBM parameters, as well as the logistic regression learning rate, it appears the best results are:

Logistic Regression: Learning Rate = 1.0, with L2 Regularization
RBM: Learning Rate = 0.1, 100 hidden components with a batch size of 10, and 50 iterations

This combination scored a validation test score of 97.2% accuracy, and thus 2.8% error.

4.2 Comparisons

Now using the learning parameters decided upon via cross validation, I fitted an RBM with 100 hidden units, as well as one with 50 hidden units. I then used the extracted features from both these RBMs, and used them in a logistic regression classifier. I repeated the same procedure with features I extracted from PCA. The results are quite interesting, in fact, all reductions used almost had the same performance on the test set. The latent features from the 100 unit RBM actually outperformed all other representations, and in fact, it actually performed .2% better than the original data on the classification on the test set. The results can be seen in Table 1. It appears that the number of units of the reduced representation matter, as we see both the 100 unit RBM and the 100 representation from PCA performed better than their 50 unit counterparts. This seems to indicate that there are latent features/a latent structure that can be found in the bag of words representations containing information about the class of the email. Figure 1 shows the intensity of the weights between certain words and hidden units in the 100 unit RBM model. The darkest areas are those connections with large positive weights, leading to high energy, and low probability of occurring together. Where as the whitest areas are high probability pairs of hidden units and visible units.

4.3 Analysis of 100 Unit RBM

Looking deeper into the inner workings of the 100 Unit RBM, I've done some reverse engineering to find the hidden units that resulted in the largest coefficients in the logistic regression model(indicating spam), and found the words which had the most negative weights associated with those hidden units. Again, the most negative weights were used because they would lead to the smallest energy functions, which result in higher probabilities (ie: there's a greater chance for the hidden unit and the word to appear together). This will hopefully give me an idea of which words are most likely to be in spam. The top words that I got using this approach were "send, company, thanks, quoted, transitional", however these words do not make much sense when thinking about spam. I think a reason for this, is the model is more complex and includes bias terms for each hidden unit and word, for which I haven't taken into consideration.
In comparison, the top words that indicated spam for the original data were: "unsubscribe, money, prices, rate, price." These words are much more likely, in my opinion, to indicate spam. However, the LR model trained on the latent representations from the RBM are very good, so there are definitely complexities in the model's representations that I'm not taking into account.

Table 1: Comparisons of classifier scores

Model Specifications	Training Score	Test Score
RBM 100 Hidden Units	97.675%	97%
RBM 50 Hidden Units	96.825%	96.5%

PCA 100 Features	97.475%	96.9%
PCA 50 Features	96.525%	96.1%
Original Data	97.85%	96.8%

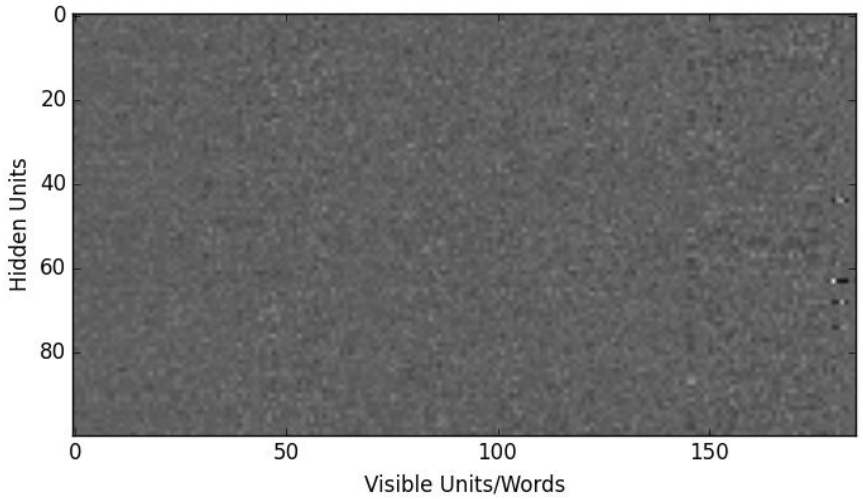

Figure 2: 100 Unit RBM weights.

4.4 Layer-by-layer approach

Using the best setting for the number of hidden units (100), I will train 2 RBM's in succession, to make the same dimensionality reduction. The first RBM will be trained on the training data (4000 data points, with 185 features), and will have 135 hidden units (chosen arbitrarily). The next RBM will take as input, the hidden units of the previous, ie: the first RBM. Therefore, the input to the second RBM is 4000 data points, with 135 dimensions each. Finally we will train a logistic regression classifier using the outputs of the second RBM, which will be 4000, 100 dimensional vectors. Using this approach I was able to get 93.974% accuracy on the training set, and 93.1% on the test data set. It appears that more information was lost while transferring information through another RBM than by using just a single RBM. The issue with this approach may be that the assumption that there's a latent representation of the latent representation may be incorrect, and too complex for the actual data.

To check this result, without having to arbitrarily choose the number of hidden units for the first RBM, I repeated this approach by first training 100 hidden unit RBM, and for the second, a 50 hidden unit RBM. The accuracy rate on the test set I got using this approach was 96.2%, which is very close to that of the 50 unit RBM from the previous section, which had an accuracy of 96.5%. Again the 2 RBM approach gave a slightly poorer result, however this time the performance gap was much smaller. This was quite an interesting result, signifying the choice of the size of the first RBM is extremely important. It may be the case that we do not get good representations from the first RBM, that things ultimately will not work out well. One more test was then repeated, going from a 135 unit RBM to a 50 unit RBM, which surprisingly gave a test accuracy of 95%. This further signifies the importance of getting good representations from the first RBM, but also that the choice of the reduction afterwards is dependent on this choice. As we see, going from 135->50 had about a 1% better performance than going from 135->100. Overall this provides some evidence, however not comprehensive, that a layer-by-layer approach does not improve on the quality of representations used in the email classification task.

4.5 Classification with the 100 hidden unit RBM

By concatenating the labels to the training data to create a training set of 4000 data points, and 186 dimensions, each new data point is of the form $v' = < v, label >$. Using the same settings as before, I trained an RBM with 100 units on this new training data. In order to make classifications on the test set, I chose the class c that satisfied this equation:

$$class(v) = argmax_c \, log(\sum_h exp(-E(< v, c >, h))) \quad (4)$$

The BernoulliRBM package that was used was able to calculate the free energy for inputs, and I used this available function in order to evaluate (4), and classify the emails.

Initially this gave a training accuracy of 64.025% on the training data, and 62.9% on the test data. However, I was convinced that the model should be doing better, so I ran cross-validation to pick a better learning rate. The new learning rate I came up with was 0.001, 10 times smaller than the one used in previous sections. Using this new learning rate, I was able to get the RBM to get 87.55% accuracy on the training set, and 87.5% on the test set. Firstly, it appears that the learning rate is very important in training RBMs using PCD as mentioned earlier. Secondly, although 87.5% accuracy is quite good, it falls way behind the logistic regression classifier. Lastly, although this test was not exhaustive, it seems to give some indication that using the RBM as a pre-processor for another classifier appears to be better than doing both tasks within the RBM itself (those tasks being, learning latent representations, and classifying the emails); this at least holds true for a binary classification task with binary inputs.

5 Conclusion

The purpose of the project was to see whether or not reducing the dimensions of the input data had any impact on the performance of a classifier on the email classification task; In particular, this was done with a special emphasis on the use of binary Restricted Boltzmann Machines. What the experiments have shown is that the use of RBMs for this task are indeed justified when the data itself is binary in nature. However, whether not it's worth it to use an RBM for this preprocessing task may be up for debate, considering preprocessing with PCA resulted in very similar performances. In addition, PCA is much less computationally expensive to train, and does not require the need to fine tune certain parameters like the learning rate. The idea of using RBMs to classify emails was also explored, although it did not perform as well as the logistic regression classifier (both with and without preprocessing), it was still a neat use of the RBM model. Finally, all results were based on the email data with binary inputs, in general, the conclusions may not hold for other data sets, or using other binary classifiers other than logistic regression. Overall there were too many variations and possibilities for experiments for me to explore and exhaust them all in this project, however I hope you enjoyed the read!

References

[1] G.E. Hinton and R. R Salakhutdinov. Reducing the dimensionality of data with neural networks. Science, 313(5786):504–507, 2006.

[2] T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. International conference on Machine Learning, 1064–1071, 2008.

[3] Tieleman and Hinton. Using Fast Weights to Improve Persistent Contrastive Divergence. ACM, 1033-1040, 2009.

[4] B.Chen, K.Swersky, N.Freitas. A Tutorial on Stochastic Approximation Algorithms for Training

Restricted Boltzmann Machines and Deep Belief Nets. 2010.

[5] G.E. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. 2010.

[6] G. E. Hinton. Training products of experts by minimizing contrastive divergence. Neural Computation, vol. 14, 2002.