Document Classification with Deep Rectifier Neural Networks and Probabilistic Sampling

Tamás Grósz and István Nagy T.

Department of Informatics, University of Szeged, Hungary
{groszt,nistvan}@inf.u-szeged.hu

Abstract. Deep learning is regarded by some as one of the most important technological breakthroughs of this decade. In recent years it has been shown that using rectified neurons, one can match or surpass the performance achieved using hyperbolic tangent or sigmoid neurons, especially in deep networks. With rectified neurons we can readily create sparse representations, which seems especially suitable for naturally sparse data like the bag of words representation of documents. To test this, here we study the performance of deep rectifier networks in the document classification task. Like most machine learning algorithms, deep rectifier nets are sensitive to class imbalances, which is quite common in document classification. To remedy this situation we will examine the training scheme called probabilistic sampling, and show that it can improve the performance of deep rectifier networks. Our results demonstrate that deep rectifier networks generally outperform other typical learning algorithms in the task of document classification.

Keywords: deep rectifier neural networks, document classification, probabilistic sampling.

1 Introduction

Ever since the invention of deep neural nets (DNN), there has been a renewed interest in applying neural networks (ANNs) to various tasks. The application of a deep structure has been shown to provide significant improvements in speech [5], image [7], and other [11] recognition tasks. As the name suggests, deep neural networks differ from conventional ones in that they consist of several hidden layers, while conventional shallow ANN classifiers work with only one hidden layer. To properly train these multi-layered feedforward networks, the training algorithm requires modifications as the conventional backpropagation algorithm encounters difficulties ("vanishing gradient" and "explaining away" effects). In this case the "vanishing gradient" effect means that the error might vanish as it gets propagated back through the hidden layers [1]. In this way some hidden layers, in particular those that are close to the input layer, may fail to learn during training. At the same time, in fully connected deep networks, the "explaining away" effects make inference extremely difficult in practice [6].

As a solution, Hinton et al. presented an unsupervised pre-training algorithm [6] and evaluated it for an image recognition task. After the pre-training of the DNN,

P. Sojka et al. (Eds.): TSD 2014, LNAI 8655, pp. 108–115, 2014.

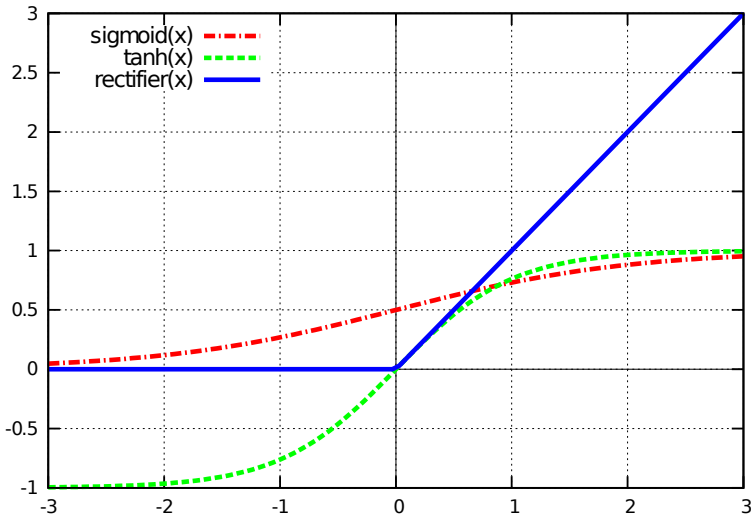

Fig. 1. The rectifier activation function and the commonly used activation functions in the neural networks, namely the logistic sigmoid and hyperbolic tangent (tanh)

the backpropagation algorithm can find a much better local optimum of the weights. Based on their new technique, a lot of effort has gone into trying to scale up deep networks in order to train them with much larger datasets. The main problem with Hinton's pre-training algorithm is the high computational cost. This is the case even when the implementation utilizes graphic processors (GPUs). Several solutions [4,10,2] have since been proposed to alleviate or circumvent the computational burden and complexity of pre-training, one of them being deep rectifier neural networks [2].

Deep Rectifier Neural Networks (DRNs) modify the neurons in the network and not the training algorithm. Owing to the properties of rectified linear units, the DRNs do not require any pre-training to achieve good results [2]. These rectified neurons differ from standard neurons only in their activation function, as they apply the rectifier function $(max(0, x))$ instead of the sigmoid or hyperbolic tangent activation. With rectified neurons we can readily create sparse representations with true zeros, which seem well suited for naturally sparse data [2]. This suggests that they can be used in document classification, say, where the bag of words representation of documents might be extremely sparse [2]. Here, we will see how well DRNs perform in the document classification task and compare their effectiveness with previously used successful methods. To address the problem of unevenly distributed data, we combine the training of DRNs and ANNs with a probabilistic sampling method, in order to improve their overall results.

2 Deep Rectifier Neural Networks

Rectified neural units were recently applied with success in standard neural networks, and they were also found to improve the performance of Deep Neural Networks on tasks like image recognition and speech recognition. These rectified neurons apply the rectifier function $(max(0, x))$ as the activation function instead of the sigmoid or hyperbolic tangent activation. As Figure 1 shows, the rectifier function is one-sided,

hence it does not enforce a sign symmetry or antisymmetry. Here, we will examine the two key properties of this one-sided function, namely its hard saturation at 0 and its linear behaviour for positive input.

The hard saturation for negative input means that only a subset of neurons will be active in each hidden layer. For example, when we initialize the weights uniformly, around half of the hidden units output are real zeros. This allows rectified neurons to achieve truly sparse representations of the data. In theory, this hard saturation at 0 could harm optimization by blocking gradient back-propagation. Fortunately, experimental results do not support this opinion, suggesting that hard zeros can actually help supervised training. These results show that the hard non-linearities do no harm as long as the gradient can propagate along some path [2].

For a given input, the computation is linear on the subset of active neurons. Once the active neurons have been selected, the output is a linear combination of their input. This is why we can treat the model as an exponential number of linear models that share parameters. Based on this linearity, there is no vanishing gradient effect [2], and the gradient can propagate through the active neurons. Another advantage of this linear behaviour is the smaller computational cost: there is no need to compute the exponential function during the activation, and the sparsity can also be exploited. A disadvantage of the linearity property is the "exploding gradient" effect, when the gradients can grow without limit. To prevent this, we applied L1 normalization by scaling the weights such that the L1 norm of each layer's weights remained the same as it was after initialization. What makes this possible is that for a given input the subset of active neurons behaves linearly, so a scaling of the weights is equivalent to a scaling of the activations.

Overall, we see that Deep Rectifier Neural Networks use rectified neurons as hidden neurons. Owing to this, they can outperform pre-trained sigmoid deep neural networks without the need for any pre-training.

3 Probabilistic Sampling

Most machine learning algorithms – including deep rectifier nets – are sensitive to class imbalances in the training data. DRNs tend to behave inaccurately on classes represented by only a few examples, which is sometimes the case in document classification. To remedy this problem, we will examine the training scheme called probabilistic sampling [12].

When one of the classes is over-represented during training, it might cause that the network will favour that output and label everything as the most frequent class. To avoid this, it is necessary to balance the class distribution by presenting more examples taken from the rarer classes to the learner. If we have no way of generating additional samples from any class, then resampling is simulated by repeating some of the samples of the rarer classes.

Probabilistic sampling is a simple two-step sampling scheme: first we select a class, and then randomly pick a training sample from the samples belonging to this class. Selecting a class can be viewed as sampling from a multinomial distribution after we assign a probability to each class. That is,

$$P(c_k) = \lambda \frac{1}{K} + (1 - \lambda) Prior(c_k), \tag{1}$$

where $Prior(c_k)$ is the prior possibility of class c_k, K is the number of classes and $\lambda \in 0, 1$ is a parameter. If λ is 1, then we get a uniform distribution over the classes; and with $\lambda = 0$ we get the original class distribution.

4 Experimential Setup

In our experiments, the Reuters-21,578 dataset was used as our training and testing sample set. This corpus contains 21,578 documents collected from the Reuters newswire, but here just the 10 most frequent categories were taken from the 135. For each category, 30% of the documents were randomly selected as test documents and the rest were employed as the training sets. In the evaluation phase, one category was employed as the positive class, and the other nine categories were lumped together and treated as the negative class; and each category played the role of the positive class just once. The documents were represented in a tf-idf weighted vector space model, where the stopwords and numeric characters were ignored.

4.1 Baseline Methods

In order to compare the performance of our method with that for the other machine learning algorithms, we also evaluated some well-known machine learning methods on our test sets.

First, we applied C4.5, which is based on the well-known ID3 decision tree learning algorithm [9]. This machine learning method was a fast learner as it applied axis-parallel hyperplanes during the classification. We trained the J48 classifier of the WEKA package [3], which implements the decision tree algorithm C4.5. Decision trees were built that had at least two instances per leaf, and used pruning with subtree raising and a confidence factor of 0.25.

Support Vector Machines (SVM) [13] were also applied. SVM is a linear function having the form $f(x) = w^t x + b$, where w is the weight vector, x is the input vector and $w^t x$ denotes the inner product. SVM is based on the idea of selecting the hyperplane that separates the space (between the positive and negative classes) while maximizing the smallest margin. In our experiments we utilized LibSVM[1] and the Weka **SMO** implementation.

4.2 Neural Network Parameters

For validation purposes, a random 10% of the training vectors were selected before training. Our deep networks consisted of three hidden layers and each hidden layer had 1,000 rectified neurons, as DRNs with this structure yielded the best results on the development sets. The shallow neural net was a sigmoid net with one hidden layer, with the same number of hidden neurons (3,000) as that for the deep one.

[1] `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

Table 1. The F-score results got from applying different machine learning algorithms (DRN: Deep Rectifier Network, ANN: Shallow Neural Network, SMO, LibSVM: Support Vector Machine, J48: Decision Tree) on the Reuters Top 10 classes

Task	DRN	ANN	SMO	LibSVM	J48
ship	88.20	87.12	87.65	**88.61**	83.15
grain	**96.40**	95.11	94.77	93.1	95
money-fx	93.52	**94.06**	88.56	78	86.13
corn	83.22	76.80	86.9	78.12	**91.78**
trade	**95.74**	93.38	94.41	91.04	85.52
crude	**94.62**	91.21	91.23	90.63	86.36
earn	**98.74**	98.31	98.46	98.52	96.43
wheat	87.12	81.97	**92.49**	86.42	91.86
acq	**97.54**	97.13	96.76	96.86	91.83
interest	94.46	**96.00**	89.96	77.25	82.71
micro-avg	**96.22**	95.42	92.18	87.64	87.86

The output layer for both the shallow and the deep rectifier nets was a softmax layer with 2 neurons – one for the positive class and one for the negative class. The softmax activation function we employed was

$$softmax(y_i) = \frac{e^{y_i}}{\sum_{j=1}^{K} e^{y_j}}, \tag{2}$$

where y_i is the i^{th} element of the unnormalised output vector y. After applying the *softmax* function on the output, we simply select the output neuron with the maximal output value, and this gives us the classification of the input vector. For the error function, we applied the cross entropy function.

Regularization is vital for good performance with neural networks, as their flexibility makes them prone to overfitting. Two regularization methods were used in our study, namely early stopping and weight decay. Early stopping regularization means that the training is halted when there is no improvement in two subsequent iterations on the validation set. The weight decay causes the weights to converge to smaller absolute values than they otherwise would.

The DRNs were trained using semi-batch backpropagation, the batch size being 10. The initial learn rate was set to 0.04 and held fixed while the error on the development set kept decreasing. Afterwards, if the error rate did not decrease in the given iteration, then the learn rate was subsequently halved. The λ parameter of the probabilistic sampling was set to 1, which means that we sampled from a uniform class distribution.

5 Results

Table 1 lists the overall performance we got from training the different machine learning methods on the Reuters dataset. Here, F-scores were used to measure the effectiveness of the various classifiers and we applied the micro-average method [8] to calculate an

Table 2. Neural networks results got with and without probabilistic sampling (P.S.), on the three most unbalanced tasks

	ship			corn			wheat		
Method	F-score	Prec.	Recall	F-score	Prec.	Recall	F-score	Prec.	Recall
DRN	88.20	94.67	82.56	83.22	80.52	86.11	87.12	93.42	81.61
DRN+ P.S.	**90.48**	92.68	88.37	**87.50**	87.50	87.50	**89.89**	87.91	91.95
ANN	87.12	92.21	82.56	76.80	90.57	66.67	81.97	78.13	86.21
ANN+ P.S.	90.36	93.75	87.21	85.29	90.63	80.56	85.56	80.00	91.95

overall F-score. Micro-averaging pools per-document decisions across classes, and then computes an effectiveness measure on the pooled contingency table.

As can be seen, the DRN method outperformed the other methods in general, but it performed poorly (F-score below 90) on three classes. From among the baseline algorithms, the best one was the SMO, with a micro-average score of 92.18. Compared to the other two baseline methods, which yielded approximately the same micro-average score, the SMO achieved a better overall score of 4.5. To make a sense of the relative effectiveness of the neural nets, we decided to compare their perfomance with that for the SMO – the best one of the baseline methods. The micro-average score of the DRN is 96.22, which is 4.04 higher than that for the SMO. The ANN achieved an average F-score of 95.42, which is 3.24 higher than that for the micro-average score of the SMO. This means that the average effectiveness of DRNs is competitive with classifiers like SVMs and decision trees. However, on small classes (‘ship’, ‘corn’ and ‘wheat’), which were represented with fewer than 200 positive examples in the training set, DRNs and ANNs performed much worse. Interestingly, on these rare classes the baseline algorithms performed quite differently. On the ‘ship’ class LibSVM yielded the best result, but on the ‘corn’ class J48 was the best and for the ‘wheat’ class the SMO achieved the best result.

Next, we investigated the three tasks on which the neural networks approach was outperformed by the other methods. These tasks were the most under-represented classes, so to improve the results we applied probabilistic sampling. In Table 2, we see the improvements got for the deep and the shallow networks after applying it. For the DRNs, the improvement was 3.11 on average, while for the ANNs it was 5.1; but the DRNs yielded better results for all three classes.

With probabilistic sampling, DRNs outperformed LibSVM on all three tasks, and the SMO was better only on the ’wheat’ class. The J48 results were still better on the ’corn’ and the ’wheat’ classes, but the DRNs performed much better on the other eight classes.

6 Discussion

Deep Rectifier Neural Networks outperformed our baseline algorithms, which probably tells us that they are suitable for document classification tasks. However, they face difficulties when some of the classes are underrepresented.

The results of our experiment show that probabilistic sampling greatly improves the F-scores for the DRNs and the ANNs on the underrepresented classes. To understand precisely how probabilistic sampling helps the training of neural networks on these classes, we investigated the effects it produced. The most important one is that after probabilistic sampling balanced the distribution of positive and negative examples, the recall values increased here. The reason behind this is quite simple: the neural networks get more positive examples during training. As the neural nets get more positive samples, the proportion of negative samples decrease. This sometimes caused a drop in the precision score. However this reduction was much smaller than the increase in the recall score, as the negative samples were still well represented.

Comparing the results of the DRNs with those got using ANNs, we can say that the DRNs are not only better but their training and evaluation phases are faster too. To support this opinion, we should mention that the shallow sigmoid network had approximately 1.5 times more parameters. The ANN had $2{,}000\times3{,}000$ connections between input units and hidden units and $3{,}000\times2$ weights for the output layer, while the DRN had only $2{,}000\times1{,}000$ input-hidden, $1{,}000\times2$ hidden-output, and $2\times1{,}000\times1{,}000$ hidden-hidden connections. Thanks to the greater number of parameters, ANNs were able to learn a better model for the 'money-fx' and 'interest' classes. On the other eight classes, the DRNs yielded better results, and this suggests that deep structures are better than shallow ones, for the tasks described earlier.

7 Conclusions

In this paper, we applied deep sparse rectifier neural nets to the Reuters document classification task. Overall, our results tell us that these DRNs can easily outperform SVMs and decision trees if the class distribution is reasonably balanced. With extremely unbalanced data, we showed that probabilistic sampling generally improves the performance of neural networks.

In the future, we would like to investigate a semi-supervised training method for DRNs, so they could be applied on such tasks where we have only a small number of labelled examples and a large amount of unlabelled data.

Acknowledgment. Tamás Grósz were funded in part by the European Union and the European Social Fund through the project FuturICT.hu (TÁMOP-4.2.2.C-11/1/KONV-2012-0013).

References

1. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proc. AISTATS, pp. 249–256 (2010)
2. Glorot, X., Bordes, A., Bengio, Y.: Deep sparse rectifier networks. In: Proc. AISTATS, pp. 315–323 (2011)
3. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explorations 11(1), 10–18 (2009)
4. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. CoRR. 1207.0580 (2012)

5. Hinton, G.E., Deng, L., Yu, D., Dahl, G.E., Rahman Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., Kingsbury, B.: Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal Process. Mag. 29(6), 82–97 (2012)
6. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation 18(7), 1527–1554 (2006)
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Proc. NIPS, pp. 1106–1114 (2012)
8. Manning, C.D., Raghavan, P., Schütze, H.: Introduction to Information Retrieval. Cambridge University Press (2008)
9. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco (1993)
10. Seide, F., Li, G., Chen, X., Yu, D.: Feature engineering in context-dependent deep neural networks for conversational speech transcription. In: Proc. ASRU, pp. 24–29 (2011)
11. Srivastava, N., Salakhutdinov, R.R., Hinton, G.E.: Modeling documents with a deep Boltzmann machine. In: Proc. UAI, pp. 616–625 (2013)
12. Tóth, L., Kocsor, A.: Training HMM/ANN hybrid speech recognizers by probabilistic sampling. In: Duch, W., Kacprzyk, J., Oja, E., Zadrożny, S. (eds.) ICANN 2005. LNCS, vol. 3696, pp. 597–603. Springer, Heidelberg (2005)
13. Vapnik, V.N.: Statistical learning theory. Wiley (September 1998)