Deep Neural Network in classification of Twitter Buzz

Ning LI
Department of Staitstics, University of Washington
Seattle, WA, USA, 98105
ningli30@uw.edu

Abstract

The prediction of bursty events on the Internet is a challenging task. [**?**] generated artificial data by defining different features to represent the Buzz case. Considering less 1% Buzz datapoints in original data, this kind data shows extremely inbalanced property. In this study, we investigate the performances of neural network, especially deep neural network classifiers for this inbalanced data. Experiments demonstrates the effectiveness and robustness of deep neural network classification model with 98.21% areas under the ROC curve (AUC) for validation data.

1 Motivation

In this study, we are going to classify binary case(Buzz/NonBuzz) from Online Social Media benchmark data by neural network. This data set has serious imbalance property,which is that the number of Buzz instances are much smaller than non-Buzz instances. A number of solutions to the class-imbalance problem were previously proposed both at the data and algorithmic levels. At the data level, these solutions include many different forms of resampling such as random oversampling with replacement, random undersampling and so on. At the algorithmic level, solutions include adjusting the costs of the various classes, adjusting the probabilistic estimate at the tree leaf, adjusting the decision threshold an so on.
In this paper, we use neural network, which is very sensitive to imbalanced data, to perform this classification task. Because of inefficient algorithm to solve neural network, people tend not to use neural network on imbalanced data. However,we implement deep neural network with introducing dropout, which is a efficient strategy to solve slow update in gradients, and momentum, which is used to avoid over-fitting issue in neural network.

2 Data description

2.1 Feature description

The data we are going to study is a subset of the Buzz Prediction in Online Social Media benchmark data In this paper, we are only talking about binary classification (buzz/no buzz) and the domain is Twitter. The total data we collected contains 10,000 instances.
Each instance covers seven days of observation for a specific topic (eg. overclocking...). Considering the couple day following this initial observation; If there is at least 500 additional active discussions by day (on average, with respect to the initial observation) then, the predicted attribute Buzz is True. Observations are Independent and identically distributed. There are 11 primary features in each instance, which are listed below:

- **NCD** Number of Created Discussions measures the number of discussions created at time step t and involving the instance's topic;

- **AI** Author Increase measures the number of new authors interacting on the instance's topic at time t (i.e. its popularity);
- **AS(NA)** Attention Level measures the attention payed to a instance's topic on a social media;
- **BL** Burstiness Level is the burstiness level for a topic z at a time t is defined as the ratio of ncd and nad
- **NAC**Number of Atomic Containers measures the total number of atomic containers generated through the whole social media on the instance's topic until time t;
- **CS** Contribution Sparseness measures the spreading of contributions over discussion for the instance's topic at time t.
- **AT** Author Interaction measures the average number of authors interacting on the instance's topic within a discussion.
- **NA** Number of Authors measures the number of authors interacting on the instance's topic at time t.
- **ADL**Average Discussions Length directly measures the average length of a discussion belonging to the instance's topic.
- **NAD**Average Discussions Length measures the number of discussions involving the instance's topic until time t.

2.2 Data property

Among these 10,000 instances in this dataset, only 856 Buzz data points (0.856%), which can be considered as extremely inbalanced dataset. The commonly classification measure L_{01} is not appropriate for this dataset. Since even if we classify all data points as non-Buzz instance, the $L_{01} = 0.856\%$, which is still very small. However, this error is not reliable, since we can't get any reasonable classification. What we really care is which kind of instance can be Buzz. Therefore, in this paper, we use True Positive rate (TP), which is also called "recall", and False Positive rate(FP) to measure the performance of a classifier.

		observation	
		positive	negative
predict	positive	TP	FP
	negative	FN	TN

Table 1: Confusion Table

3 Shallow Neural Network

Shallowed neural network is one of the shallowed-structured architectures in machine learning and signal processing techniques. In general,these techniques contain a single layer of nonlinear feature transformation.For example, Gaussian mixture models(GMMs), Support vector machines(SVMs) and so on. Shallow architectures have been shown effective in solving many simple or well-constrained problems, but their limited modeling and representational power can cause difficulties when dealing with more complicated real-world applications involving natural signals such as human speech, natural sound and language, and natural image and visual scenes. In our paper, shallowed neural network means a neural network containing only one hidden layer with sigmoid activation fucntion.

4 Deep Neural Network

In this section, we begin to discuss deep neural network[**?**], meaning ones in which we have multiple hidden layers, this allows us to compute much more complex features of the input. The goal of using deep network is to help us generate significantly great representation of our input data, so that we can do better and robust classification.In specific, Deep Neural Network requires non-linear activation function.

4.1 Structure

The design of input and output layers in a network is often straightforward. The input is the one scaled vector, appropriately between 0 and 1, of the training data,and the output could be single value (e.g. 0 or 1), or multiple neurons which represent the probability of being classified to one group. However, it can be quite an art to the design of hidden layers. The size of hidden layers depends in a complex way on: the size of input and output units, the number of training cases, the complexity of the classification to be learned, learning algorithm, regularization and so on. In most situations, there is no way to determine the best number of hidden units without training several networks and estimating the generalization error of each. If you have too few hidden units, you will get high training error and high generalization error due to underfitting and high statistical bias. If you have too many hidden units, you may get low training error but still have high generalization error due to overfitting and high variance. The discussion about how the number of hidden units affects the bias/variance trade-off can be found in the paper from In practice, there are some empirically-derived rules-of-thumb, of these, the most commonly relied on is Jeff Heaton's advice: the optimal size of the hidden layer is usually between the size of the input and size of the output layers.

4.2 Activation Function

In neural network modeling, activation function [**?**] has a significant influence on the learning results. The traditional activation functions for neural network modeling are sigmoid functions or thanh functions, while nowadays there is a new trend for neural network learning using rectifier neural units.[**?**][**?**]
sigmoid function

$$f(x) \quad = \quad \frac{1}{1 + exp(-x)}$$

Rectified linear function(ReL)

$$f(x) \quad = \quad max(0, x)$$

From above definitions of these two activation functions, we find that sigmoid function has range [0,1], while ReL function [**?**] has the range [0,∞]. Hence, sigmoid function can be used to model probability, whereas ReL can be used to model positive real number. Besides, we know the gradient of a sigmoid function is smooth comparing with the gradient of a ReL function, but the gradient can vanish as x increases or decreases. However, this problem cannot occur for ReL function. Actually, for ReL, the gradient is defined as $\begin{Bmatrix} 0 & if\ x < 0 \\ 1 & if\ x > 0 \end{Bmatrix}$
Hence, the advantage for ReL in neural network is obvious.

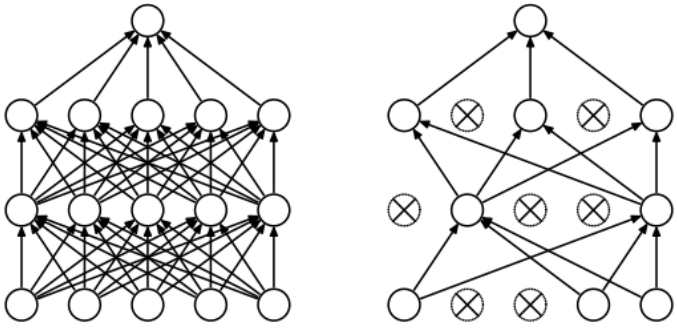
- ReL can induce the sparsity in hidden units.
- ReL doesn't face gradient vanishing problem as with sigmoid and tanh function. Also, It has been shown that deep networks can be trained efficiently using ReL even without pre-training.
- ReL can be used in Restricted Boltzmann Machine(RBM) to model real/integer valued inputs.

However, one potential problem for ReL is that the hard saturation at 0 for ReL may hurt optimization by blocking gradient back-propagation.

4.3 Dropout

Deep networks with broad hidden layers and full connectivity could not be trained to produce useful results, because of overfitting, slow convergence and other issues. Scientists have developed many techniques to solve this problem, such as auto-encoders, restricted Boltzman machine(RBM). The unsupervised, pre-training stage is a crucial component for achieving competitive overall performance on classication tasks. In this project, we adapted another recent new technique called dropout[**?**], which was shown to significantly improve the performance of deep neural networks on

various tasks, including vision problems. The scratch of dropout process is randomly dropping a unit out during training with a fixed probability p independent of other units. "Dropping" here refers to temporarily removing it from the network, along with all its incoming and outgoing connections. The value of p can be chosen using a validation set or can simply be set at 0.5, which seems to be close to optimal for a wide range of networks and tasks. For the input units, however, the optimal probability of retention is usually closer to 1 than to 0.5.



(a) Standard Neural Net (b) After applying dropout.

Figure 1: Left: standard neural network; Right: neural network with dropout

4.4 Minibatch and Momentum Stochastic Gradient Descent

Stochastic gradient descent (SGD) is the common method we choose for solving neural network model. Compared to ordinary gradient descent, Stochastic gradient descent proceeds more quickly by estimating the gradient from just a few examples at a time instead of the entire training set, so it can update more quickly. In deep neural network model training, there are two variants on SGD: **minibatch** and **momentum**.

Minibatch SGD works identically to SGD, except that we use a small batch (minibatch) of B training examples (e.g. 10,20 or 100) to make each estimate of the gradient. This technique reduces variance in the estimate of the gradient, and often makes better use of the hierarchical memory organization in modern computers. However, as the minibatch size increases, the number of updates done per computation done decreases (eventually it becomes very inefficient, like batch gradient descent). There is an optimal trade-off (in terms of computational efficiency) that may vary depending on the data distribution and the particulars of the class of function considered, as well as how computations are implemented.

Momentum[**?**]variation that is similar in spirit to minibatch SGD. If the objective has the form of a long shallow ravine leading to the optimum and steep walls on the sides, standard SGD will tend to oscillate across the narrow ravine since the negative gradient will point down one of the steep sides rather than along the ravine towards the optimum. The objectives of deep architectures have this form near local optima and thus standard SGD can lead to very slow convergence particularly after the initial steep gains. The idea of momentum is to compute online a moving average of the past gradients, and use this moving average instead of the current example's gradient, in the update equation. The moving average is typically an exponentially decaying moving average, i.e.,

$$\Delta\theta^{k+1} = \alpha\Delta\theta^{k} + (1-\alpha)\frac{\partial L(\theta^{k},z)}{\partial\theta^{k}}$$

where α is a hyper-parameter that controls the how much weight is given in this average to older vs most recent gradients. Generally α is set to 0.5 until the initial learning stabilizes and then is increased to 0.9 or higher.

5 Application and Empirical Results

we fix the former 80% of twitter 10,0000 data as training data, and the left 20 % of the data as validation data. Besides, among 80,000 training data, there are 669 records being considered as buzz(approximately 8.36%), while among the left 20,000 validation data, there are 187 buzz records(approximately 9.35 %). Hence, the buzz data distributions for training and validation are almost as same as that for original data.
In addition, we standardize training data and validation data separately, which ensures that the range for each categories is $[0, 1]$, to avoid the huge difference in scale between different categories in original data.
Last, in order for later reasonable comparison between two classification models, we set the same random seed before model training.

5.1 Shallow Neural Network

Generally speaking, Neural Network with single hidden layer is relatively easy to implement. We choose sigmoid unit as activation function, and introduce minibatch in SGD to solve model. However, the choice in number of neurons contained in hidden layer is still a arbitrary problem, we fit two different shallow neural network models with 11 and 77 respectively as the size of hidden layers. In order to simply our later description of neural network structure, here we write the neural network model, which has single hidden layer of 77 hidden neurons, 77-dimension input and binary output as **77-77-2**. In all, the setting for other hyper parameters are listed here:

- Learning rate for SGD: 0.1, minibatch size: 100, number of epoch: 100;
- Activation function: sigmoid function;
- Lost function: cross-entropy.

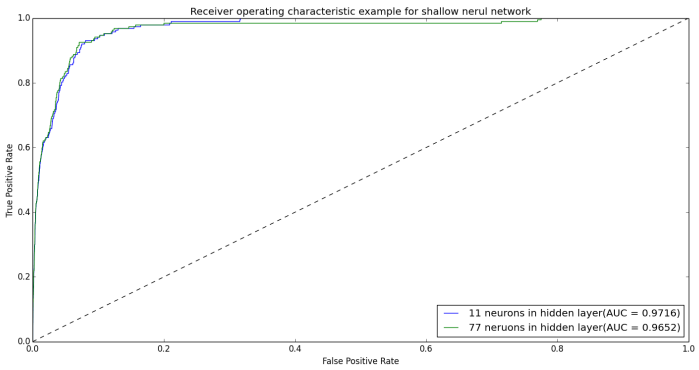


Figure 2: ROC curves for shallow neural networks

Figure[2] shows that neural model 77-11-2 (the blue line) has value of AUC 0.9716, which is a little bit higher than the one 0.9652 from model 77-77-2(the green line). Besides, the ROC curves for these two shallow neural networks are very close.

5.2 Deep Neural Network

Comparing with shallow neural network model, deep neural network fitting is much more complex in model parameters fitting, besides, we have more hyper parameters to be selected ahead. Below are a list of hyper parameters that we change in fitting deep neural network:

- Change layer structure, either with different layer size or number of layers;
- Change stochastic gradient descent algorithm parameters, e.g. batch size, number of epoch, learning rate and momentum;

- Change dropout rate

In specific, Dropout rate 0.5 means at each hidden layer, each neurons will be "dropout" at probability of 0.5 in each epoch. Generally, we only apply dropout technique on hidden layers, but in fourth experiment (which can be referred in Table[2]), we also apply it on input layer, the final AUC value does not improve very much, even lower than the deep neural network model without dropout input layer.
Another variant method we need to clarify is momentum. At first, the momentum rate is 0.5, which means the gradient for next step is the average of current gradient and the gradient calculated from last step. As the iteration times moves, we can reset the momentum rate as 0.99 or 1.
Figure[3] shows us two deep neural network ROC curved. The two ROC curves are very close, the deep neural network 77-77-77-2 model (blue one) is a little bit higher than 77-500-77-77-2 model. Many factors may affect this result that deeper model does not perform well, the choice of hyperparameters is very important in deep neural model learning.

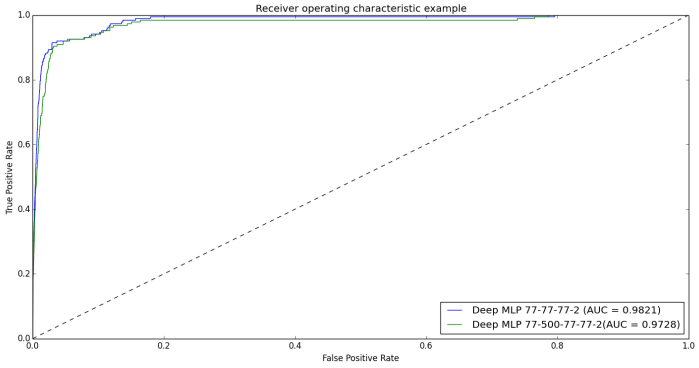


Figure 3: ROC curves for deep neural networks

5.3 Comparison between shallow and deep neural network

In this project, we conducted 11 experiments (9 deep models and 2 shallow models), all of the results are listed in Table[2].

Layer structure (input-hidden-output)	Batch size	number of epoch	Learning rate	Drop-out	Momentum*	Activation function	AUC
77-77-77-2	100	100	0.1	0.5	0.5-0.99	rectifier	0.9821
77-77-77-2	10	100	0.1	0.5	0.5-0.99	rectifier	0.9735
77-77-77-2	100	100	0.1	0.5	0.9-0.99	rectifier	0.9766
77-77-77-2	100	100	0.1	0.5(drop out input)	0.5-0.99	rectifier	0.9799
77-77-77-2	100	100	1	0.5	0.5-0.99	rectifier	0.9789
77-500-77-2	100	100	0.1	0.5	0.5-0.99	rectifier	0.9774
77-77-11-2	100	100	0.1	0.5	0.5-0.99	rectifier	0.9806
77-500-77-77-2	100	100	0.1	0.5	0.5-0.99	rectifier	0.9728
77-77-77-77-2	100	100	0.1	0.5	0.5-0.99	rectifier	0.9816
77-11-2	100	100	0.1	-	-	sigmoid	0.9716
77-77-2	100	100	0.1	-	-	sigmoid	0.9652

Table 2: summary of all experiments results

Table[2] shows all the AUC values calculated from fitting validation data in fitted models. Among 9 deep neural network (number of hidden layers are greater than 2), neural network 77-77-77-2 model, with batch size 100, 100 epoches, learning rate 0.1, drop-out rate 0.5, momentum 0.5 and rectifier activation function, the value of AUC is the highest, which is 0.9821. While among the

two shallow neural network models, model 77-11-2 has the largest AUC value on validation data, which is 0.9716. Comparing the two best neural network models, we find that after introducing more hidden layer into model, our model complexity increases, especially the number of fitting parameters, the AUC value seems increase not too much. Does that mean we do not need to switch for deep neural network? The answer is no.
Figure[4] shows us that the two deep neural network models (green line and blue line) increase faster than shallow neural network (red line) when the false positive rate is small, which means deep model tends to give us higher recall(which can be measured by true positive rate) when false positive rate is fixed. In other words, deep model can return extreme probabilities for classification. Figure[5] shows the last 500 largest points with probability of being classified as 1. The darker color is, the higher probability will be. As we can see, deep neural network tends to give us extreme probabilities for each point, which convince us to get clear boundaries for Twitter Buzz data from deep neural model.

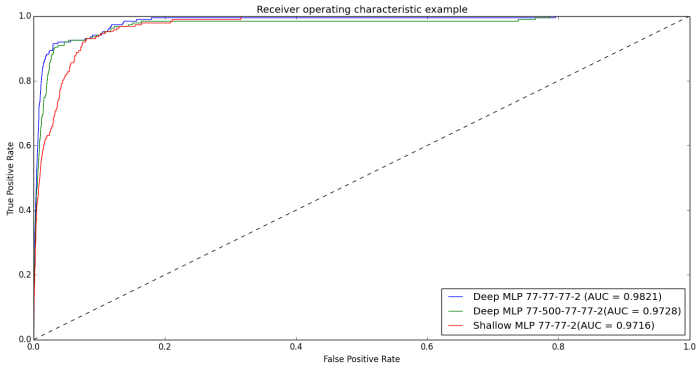


Figure 4: ROC curves for comparing shallow and deep neural networks

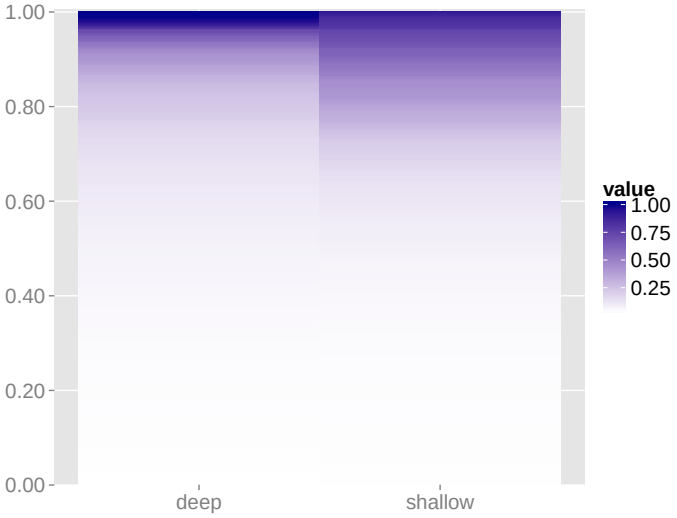


Figure 5: ROC curves for comparing shallow and deep neural networks

6 Conclusion

In all, based on the value of AUC, we find that deep neural network performs well than neural network. The problem of nerual network to classifying imbalanced data is largely caused by backpropagation algorithm in solving model. Dropout strategy and improvement in stochastic gradient descent solve this problem without resampling original data. However, as we know, deep neural network performs significantly well in image recognition field, which data is high dimensions. Comparing with image pixel data, our study data contains only 77 dimensions, which can be considered as ”low” dimension data. Therefore, the advantage of deep neural network cannot be seen clearly. Based on the comparison of shallow neural network and deep neural network, we can still tell the deep neural network can give us more robust classification results.

References