Rage Against the Machine: Avoiding Overfitting with Nearly Many Variables as Training Cases

Abstract

In many classification problems we are required to work with ill formed training data. In some cases, the data is highly dimensional or only a small data set of training data is available. In this paper I compare different machine learning techniques for binary classification using data set that has nearly as many variables as training samples and a lot of unlabeled data. I also perform a feature selection technique based on SVM improving the classification performance. Overall, Logistic regression is the technique that obtains the best classification performance with and without performing feature selection.

1. Introduction

One of the main objectives creating a predictive model, is to construct one that will give accurate predictions on unseen data. A necessary step in order to construct such model is to ensure that the model does not overfit the training data, that may cause predictions on the new data that are not the optimal.

There are many areas where often the number of cases is severely restricted. In order to build an accurate model not only the appropriate techniques should be used, but some techniques and strategies need to be used in order avoid overfitting and improve the classification performance.

In this paper we compare 4 algorithms, while using a feature selection technique to improve the overall performance. The data set used was obtained from a Kaggle competition [1] Don't Overfit! containing a data set of 20,000 cases, but only 250 labeled cases.

The remainder of this document is organized as follows: In Section 2 I describe the data set that is being used as well as discuss its characteristics. In section 3 I briefly describe the techniques used and their implementation. Section 4 discusses the technique of dimensionality reduction issues. Section 5 and 6 describe the experiments performed and their results. Finally, in section 7 we discuss the results obtained.

2. Know Your Enemy: The Data Set

The data used comes from the Kaggle Challenge *Don't Overfit*[2]. The organizers of the challenge created a simulated data set with 200 variables and 20,000 cases. An equation based on this data was created in order to generate a Target to be predicted.

The data set file consists of 20,000 rows of data, each with 200 variables of which only 'Target' for the first 250 is given. The 'Target' is either 1 or 0, so this is a binary classification problem.

In addition this basic setup, additional 'target' columns are provided coming from different models, of these one is worth mention, the **'Target Practice'** column that contains the target for all the 20,000 cases. According to the creators of the challenge, These extra columns are based on the same underlying data, but the generated equation is different. That is, of similar form but with different values for the underlying parameters.

There are two characteristics of these data set that are worth to mention. First, there is a lot of unlabeled data. This is obviously a opportunity to try semi-supervised learning techniques in order to gain insight from the existing data. Second, there are 200 variables in the model, maybe many of these variables are not important for the model, we could then try to reduce model complexity and reduce the dimension of the data set.

These two characteristics of the data will provide the strategies for the prepossessing and the application of techniques.

[1] http://www.kaggle.com/

[2] http://www.kaggle.com/c/overfitting

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

3. Bulls on the Parade: Techniques and Software

The main software used in this paper is R version 2.16[3] . R is a free software environment for statistical computing and graphics. A lot of libraries are available containing implementation and interfaces of popular machine learning algorithms. I have to confess that even if I knew of its existence I had never used R before, but given that is popular in Kaggle competitions and that many ML algorithms are available for the platform, I decided to try learn the basic and implement most of the experiments using it.

Furthermore, R contains a excessively useful package called Caret that provided an easy to use, unified framework to build classifiers (Kuhn, 2008).

The selection of the techniques what somewhat arbitrary. Basically there was two main strategies to select the techniques. One was known techniques that worked with unlabeled data and the other is to choose whatever it worked.

The fact that I did not have any prior knowledge of haw the data is distributed makes the selection of the techniques just a process try and error.

After exploring the Kaggle forums many techniques where mentioned as working somewhat well with this data set. Being this a contest, many of the teams were secretive about the techniques they where using.

In order to experiment and to try to beat the classification of the other team the following algorithm were selected.

- Transductive Support Vector Machine
- Linear SVM
- RBF SVM
- Logistic Regression

Deep Neural Networks were also considered but not having a working implementation out of the box made difficult to use them.

In the following subsections are give short descriptions of the techniques and their software implementations.

3.1. Transductive Support Vector Machines

Transductive Support Vector Machines (TSVM) is an extension made to SVM use unlabeled data to build the classifier. In order word, allows us to use SVM in a semi-supervised learning setting. The algorithm proceeds by solving a sequence of optimization problems lower-bounding the solution using a form of local search. A detailed description of the algorithm can be found in (Joachims, 1999).

I decided to use TSVM given that it can use unlabeled data to train the SVM and that has been used successfully in high dimensional problems like automated text categorization.

The implementation used is SVMLight v0.6[4] developed by the same author of the referenced paper.

3.2. Linear SVM and RBF SVM

A linear SVM y and RBF SVM were selected, with the objective of comparing the performance with the Transductive SVM. However, the performance of these two methods in the problem was not expected.

The implementation used was based on the R package Kernlab (Karatzoglou et al., 2004) which is turn based on **libsvm** (Chang & Lin, 2001).

3.3. Logistic Regression

The last but not the least. Logistic regression is used, however via the implementation of (Friedman et al., 2010) that also exists for R. This specific implementation is popular in R community and was mentioned several times. Therefore, I wanted to compare the results with SVM based approach.

4. Down on the Street: Dimensionality Reduction

As explained int section 2, the data from Kaggle contains 20.000 rows of data but only 250 of them are labeled. Even if we performed normal overfitting prevention techniques like cross validation, the risk of overfittin is still high. Data overfitting arises when the number of features n is large and the number i of training is small, like in our case (Guyon et al., 2002). In other words, the higher the dimension of the data the more data we will need in order to create a model that will not overfit.

Although some of the techniques we use in this work include some sort of regularization, which is helpful to avoid overfitting, we can see from the experiments performed that nevertheless, they can benefit from space dimensionality reduction.

Not having more training samples, we could try to

[3] http://www.r-project.org/

[4] http://svmlight.joachims.org/

reduce dimension of the data. For this, There are two main ways to have basically two alternatives: feature extraction or feature selection.

Originally, I tried to apply principal components analysis (PCA) for feature extraction. But after the first experiments we noticed that the number of principal components varied of data used. Given that we only have 200 examples it I was afraid that the principal components detected were not the adequate. It is worth to mention that feature selection can also be done using principal components.

We then I turned to another method, SVM for recursive feature elimination SVM RFE, devised initially for gene selection for cancer classification (Guyon et al., 2002), in which as in our data, they only have few training examples and high dimensional data.

SVM for RFE uses the magnitude of each of the weights multiplying the inputs of a linear classifier as a ranking criteria. In order to eliminate features, they train a linear classifier (in this particular case a linear SVM) and recursively eliminate the features with the smaller weight magnitude. Then I train the selected classifier (which can also be a SVM) and based on a defined metric the best model (e.g. AUC).

The feature ranking algorithm as described (Guyon et al., 2002) is described in algorithm 1.

We used SVM RFE using SVM Linear performing k-Fold cross validation with a value of k equal to 10 using the 250 training data available. For this I used the RFE function found in Caret R package mentioned in section 2. After the selection, 150 features remained. Figure 1 shows the behaviour of the training against the number of features.

5. Ashes in the Fall: Experiments

In order to look for the best mode we performed two groups of experiments. One with the original variables and one after performing the dimensionality reduction procedure.

The performance metric used for all the training and the classification is the area under the curve (AUC) of a Receptor Operated Characteristic (ROC) curve, metric that is popular in binary classification problem in the machine learning community.

As mentioned in section 3, we trained Linear SVM, RBF SVM and the Logistic Regression methods using 250 labeled examples available. In order to minimize overfitting the standard 10-Fold cross-validation was used as hyper-parameter tuning technique.

Algorithm 1 SVM Feature Ranking

Input: Training samples $X = \{x_1, x_2, .., x_i\}$, Class labels=$y = \{y_0, y_1, ..., y_j\}$
Initialize Surviving features $S = \{1, ...n\}$
Initialize Feature ranked list $R =$
repeat
  $X = X\{S\}$; only use features in the surviving feature set.
  $\alpha =$ svmtrain(X,y)
  $w = \sum_{k=0}^{n} \alpha_k y_k x_k.$; calculate w
  $c_i = (w_i)^2$
  $f = argmin(c)$; find the feature with the smallest ranking criteria.
  update the feature ranking list R and eliminate f from the surviving list R.
until S is empty

The TSVM was trained differently. Given that it needs the unlabeled data to train no cross-validation, but a sort of greedy search for the best hyper-parameter based on the classification performance on the unlabeled data was used.

After the initial training was performed, the variable

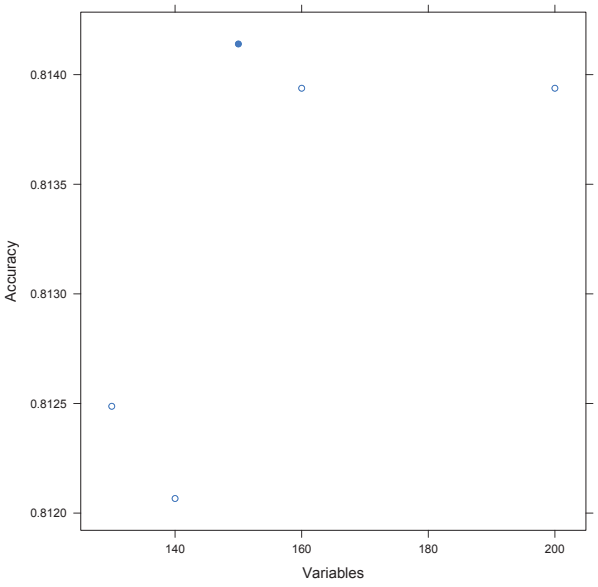

Figure 1. Behaviour of the training Accuracy (measured with ROC AUC) against the number the variables selected. The highest AUC is giving when using 150 variables.

selection method described in section 4 was performed. After this selection, the training procedure was performed one more time.

Figure 2 and 3 shows the training performance while changing the C hyper-parameter of the transductive SVM. The first graph correspond to the case when no feature selection had been performed, the second one correspond to the training after performing the feature selection.

With each of the trained model, the labels of the 19750 *unlabeled* samples was predicted. As the *Target_Practice* column of data set was used, I had the labels I could compare the predicted labels with the actual ones without relaying on the Kaggle system.

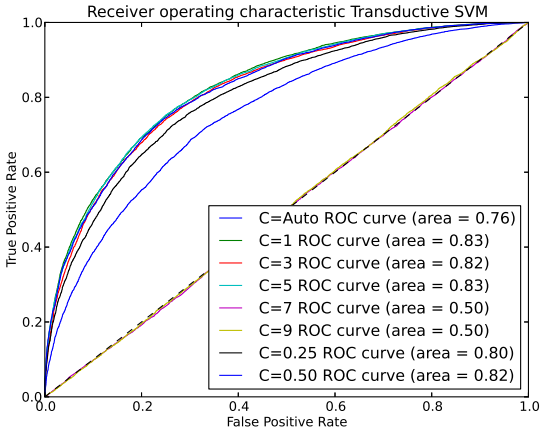

Figure 2. Training with original variables: different ROC curves for the C parameters of the Transductive SVM.

Table 1. Classification performance with original variables.

Technique	Best Parameters	AUC
Log Reg (glmnet)	α=0, λ=0.01	0.8678
Linear SVM	C =0.25	0.8568
RBF SVM	C =4, σ=0.00382	0.8595
TSVM	C=1	0.8302

6. Roll Right: Results

Tables 1 and 2 show the results of the classification on the testing data the *unlabeled* examples. Note that in this context, labels are known, and they used for measuring the prediction performance.

On the tables, the techniques are ordered in descendant way by best performance, using the AUC metric. We can see clearly that two-class logistic regression (LG) outperform all the other techniques in both training setting. However after variable selection, the performance of LG is basically the same that LSVM.

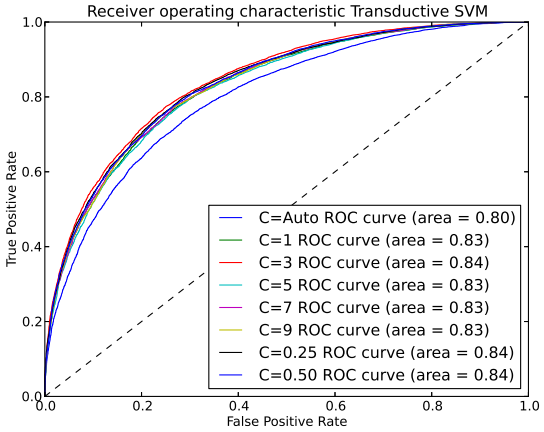

Figure 3. Training after the variable selection: A slight improvement in the AUC measures.

Table 2. Classification performance after feature selection.

Technique	Best Parameters	AUC
Log Reg (GLMNET)	α=0, λ=0.01	0.8710
Linear SVM	C =0.25	0.8705
RBF SVM	C =4, σ=0.004981	0.8611
TSVM	C=3	0.8427

However, what is really surprising is the performance of the TSVM. Of these techniques, the only that used actively the *unlabeled* data. However, it is technique is the one that performed the worst. It is surprising given that this techniques has shown to outperform many techniques in applications with high dimensional feature spaces such as text classification. The lesson here is that not necessarily a technique that work well in high dimensional space will do the same low dimensional space. However the reason behind the failure are beyond my current knowledge and the scope of this paper.

Another important aspect of the results is how much the performance was improved due to the selection of the variables. It is evident that all techniques improved the prediction performance after the selection of the variables. However, the improvement is very small. Based on the description of the challenge and

the data set is made clear that an appropriate selection of the variables will improve the performance of the classifier, however based on the results the technique used is not the adequate for this specific problem.

One last thing about the results is that after the selection of the variables, the technique with most improvement was the linear SVM. That might indicate that in fact, the function used to generate the data set is a linear function on the variables.

7. Testify: Conclusion

I performed binary classification task on an artificially generated data set with 200 features. To classify correctly 19750 samples I was given only 250 labeled exampled. Almost as many as features the data set. In other words, a problem where the risk of overfitting is high.

We used standard techniques in order to classify the data and also performed a feature selection proccess with the objective of improving the performance of the classifiers used.

The feature selection allowed the classifiers to perform better, however the improvement was very small. Probably the selection of the variables was not the adequate. Maybe with trying alternatives to the variable selection process the improvement on the classification performance can be higher.

Finally, something that I found really interesting is that the border between machine learning techniques and statistic is very diffuse. Most of the techniques I used are considered machine learning techniques, however in websites like Kaggle the most of the people use statistics software and many of the popular techniques (like GLMNET) are published in the journal of statistical software. So the the question is, when does statistics end and machine learning start?

8. Acknowledgments

I drew inspiration from the users in the forums of Kaggle[5], many of them posted code samples that helped to get me started in R. Many discussion led me to the techniques I end used. I have always believed that one only can learn thing by actually doing them and Kaggle is one of those websites that can help you to get started quickly even with only basic theoretical knowledge.

Rage Against the Machine is a two times Grammy award winner Rock band from Los Angeles - California, formed in 1991[6]. The titles of some of their song provided the co-titles of the sections of this paper and their music provided some inspiration for my work. I strongly believe they named their band after taking a 3 hours and half final exam on Machine Learning.

References

Chang, Chih-Chung and Lin, Chih-Jen. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Friedman, Jerome, Hastie, Trevor, and Tibshirani, Robert. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL http://www.jstatsoft.org/v33/i01/.

Guyon, Isabelle, Weston, Jason, Barnhill, Stephen, and Vapnik, Vladimir. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002. ISSN 0885-6125. URL http://dx.doi.org/10.1023/A:1012487302797. 10.1023/A:1012487302797.

Joachims, Thorsten. Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pp. 200–209, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc. ISBN 1-55860-612-2. URL http://dl.acm.org/citation.cfm?id=645528.657646.

Karatzoglou, Alexandros, Smola, Alex, Hornik, Kurt, and Zeileis, Achim. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004. URL http://www.jstatsoft.org/v11/i09/.

Kuhn, Max. Building predictive models in r using the caret package. *Journal of Statistical Software*, 28(5):1–26, 11 2008. ISSN 1548-7660. URL http://www.jstatsoft.org/v28/i05.

[5] http://www.kaggle.com/forums

[6] http://en.wikipedia.org/wiki/Rage_against_the_machine