BOOSTING CLASSIFICATION ACCURACY WITH SAMPLES CHOSEN FROM A VALIDATION SET

TZU-CHENG CHUANG
School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana 47907

OKAN K. ERSOY
School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana 47907

SAUL B. GELFAND
School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana 47907

ABSTRACT

It is common to train a classifier with a training set, and to test it with a testing set to study the classification accuracy. In this paper, we show how to effectively use a number of validation sets obtained from the original training data to improve the performance of a classifier. The proposed validation boosting algorithm is illustrated with a support vector machine (SVM) in the application of lymphography classification. A number of runs with the algorithm is generated to show its robustness as well as to generate consensus results. At each run, a number of validation datasets are generated by randomly picking a portion of the original training dataset. At each iteration during a run, the trained classifier is used to classify the current validation dataset. The misclassified validation vectors are added to the training set for the next iteration. Every time the training set is changed, new classification borders are generated with the classifier used. Experimental results on a lymphography dataset shows that the proposed method with validation boosting can achieve much better generalization performance with a testing set than the case without validation boosting.

INTRODUCTION

Machine learning has been used in cancer prediction and prognosis for nearly 20 years (Cruz and Wishart 2006). There are several methods widely used for this purpose, such as decision tree, Naïve Bayes, k-nearest neighbor, neural network and support vector machine algorithms. New algorithms are still developed to improve the classification accuracy. One approach is to utilize feature extraction to select fewer features to train the classifier. Bagging and boosting techniques to generate different training samples are also utilized for this purpose. In this way, a number of different classifiers can be generated, and consensus techniques such as majority voting and least squares estimation-based weighting (Kim 2003) can be used to achieve better and more stable classification accuracy.

In Bagging (Breiman 1996), several classifiers are trained independently via a bootstrap method, and their results are combined together to obtain the final decision. In this procedure, a single training set $TR=\{(\boldsymbol{x}_i;y_i)|i=1,2,\ldots,n\}$ is used to generate K different classifiers. In order to get K different training sets and make them indepent of each other, the original training set is resampled. The

new *K* training sets have the same size as the original dataset, but some instances may appear more than once, and some instances may not be in a new resampled training set.

The adaboost algorithm by Freund and Schapire (1994, 1996, 1997) is generally considered as a first step towards more practical boosting algorithms. A boosting algorithm defines different distributions over the training samples, and uses a weak learner to generate hypotheses with respect to the generated distributions. From the different distributions of training samples, different classifiers are generated, and they are next combined with different weights to get the final results.

Although the resampling technique of our proposed method is similar to bagging and boosting, our appoach is different since we utilize a number of validation sets obtained from the training set, and they are utilized to modify the training sets. Initially, we divide the original training data into 2 groups, one for training, and the other for validation. We use the training portion to train the classifier, and then we validate it with the validation set. The misclassified validation samples are added to the current training set to generate the next training set. At each iteration, the current validation set is regenerated as a randomly chosen part of the original training dataset with a fixed percentage. At each run, the procedure is repeated in several iterations until the validation accuracy reaches its maximum. At this point, a classifier is generated. Due to the random initialization of the training set and the validation set at each run, different independent classifiers are obtained with a number of runs. The results from different runs can be combined by a consensus rule such as majority voting to get the final results.

DATASET

Lymphography data is obtained from the UCI machine learning repository (Kononenko and Cestnik 1988). The examples in this data set use 18 attributes, with four possible final diagnostic classes. The attributes include lympho node dimension, number of nodes, types of lymphatics, etc. For convenient represantation, the attributes are transformed to integer type. There are a total of 148 samples, 2 are normal, 81 are metastases, 61 are malignant lymph and 4 are fibrosis. Because normal and fibrosis cases are scarce compared to the other two cases, we used 142 samples to classify whether the sample is metastases or malignant lymph.

SUPPORT VECTOR MACHINES

Vapnik invented SVM's with a kernel function in the 1990s (Vapnik 1992). This algorithm is initially designed for the two-class classification problem. One class output is marked as 1, and the other class output is marked as -1.

The algorithm tries to find the best separating hyperplane with the largest margin width. By getting a better hyperplane from the training samples, it is expected to get better testing accuracy. In SVM, the hyperplane of the non-separable case is determined by solving the following equation:

$$\begin{aligned} &\min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i \\ &\textit{subject to} \\ &y_i(x_i^T w + b) \geq 1 - \xi_i \\ &\xi_i \geq 0 \end{aligned} \tag{1}$$

where x_i is the ith data vector, y_i is the binary (-1 or 1) class label of the ith data vector, ξ_i is the slack variable, w is the weight vector normal to the hyperplane, C is the regularization parameter, and b is the bias. It can be shown that the margin width is equal to $2/|w|$.

Usually the original data is mapped by using a kernel function to a higher dimensional representation before classification. Some common kernel functions are linear, polynomial, radial basis and sigmoid functions. In our case, we used the radial basis function given by

$$K(x, x_i) = C * \exp(-\gamma * | x - x_i |^2) \tag{2}$$

In the experiments conducted, the SVM-Light (Joachims, 2004) software was utilized. We picked γ equal to 1 and C equal to 1 in these experiments.

TRAINING AND VALIDATION RESAMPLING TECHNIQUE

In the training phase, we initially decide the percentage of the training set as p_{train} and the percentage of the validation set as p_{vali}. We divide the original training set into 2 groups according to p_{train} and p_{vali}. These 2 initial training and validation sets do not overlap with each other. The training set is used to train the classifier, which is next validated with the validation set (Figure 1). Then, the misclassified validation samples are included in the training set to generate the next iteration training set. In the next iteration, the validation set is randomly picked from the original complete training set with percentage p_{vali}. With the new training set and the validation set, other misclassified validation samples are generated and are included in the training set to generate the next iteration training set. After several iterations, the performance of the classifier trained in this way becomes better than that of the classifier trained with all the original training set without any validation set. The iterations are stopped after reaching nearly 100% validation accuracy.

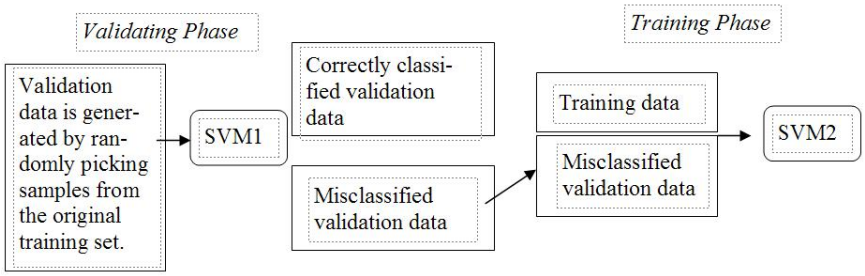

Figure 1. The misclassified validation samples are added to the training samples of the previous stage.

The proposed method emphasizes misclassified validation samples. If a misclassified sample is still misclassified the next time, it would be reemphasized, resulting in the following weighting:

$$2 \times 1_{x_i, misclassified}(x_i) + 1_{x_i, correclyclassified}(x_i) \quad (3)$$

where

$$1_{x_i, type}(x_i) = \begin{cases} 1, x_i -> type \\ 0, x_i -> other\ type \end{cases} \quad (4)$$

Due to the random initializations of the training and the validation set, we get different classifiers at each run. In order to get better results, we can use consensus such as majority voting between these classifiers.

EXPERIMENTS

We initially picked 50% of all the data for training and the other 50% for testing. In the training phase, we chose p_{train} equal to 0.5 and p_{vali} equal to 0.5. The results with four runs with different training and testing data are shown in Table 1. From Table 1, we can see that when the validation accuracy nearly reaches 100%, the testing accuracy also reaches its maximum. Because 100% validation accuracy means there are no misclassified validation samples, the iteration process is stopped after nearly reaching this value.

Table 1. Comparison of the testing classification accuracy between the classifier trained by all training data and the classifier trained by the proposed method. TestByAll means testing accuracy of the classifier trained by all training data. Valid means validation accuracy of that iteration. Test. means testing accuracy of that iteration.

TestByAll		0.5634		0.57746		0.59155		0.5493
iteration	Valid.	Test	Valid	Test	Valid	Test	Valid	Test
1	0.5714	0.5634	0.6286	0.5775	0.5143	0.5775	0.6000	0.5493
2	0.7143	0.4366	0.7714	0.4507	0.8286	0.4225	0.6571	0.4789
3	0.9714	0.7183	0.8571	0.4789	0.8286	0.5493	1.0000	0.5916
4	1.0000	0.6620	1.0000	0.6620	1.0000	0.6197	1.0000	0.5916
5	1.0000	0.6620	1.0000	0.6620	1.0000	0.6197	1.0000	0.5916

To test whether our proposed method is significant to improve the accuracy, we could convert the decimal value to the percentile value and then calculate Kai Square. By picking α =0.05, if the Kai square is larger than 0.3841, we can say that our proposed method is significantly different.

$$\chi^2 = \sum_{i=1}^{k} \frac{(x_i - E_i)^2}{E_i} \quad (5)$$

where x_i is the percentile value of testing accuracy from our proposed method, and E_i is the percentile value of testing accuracy from the classifier which is trained by using all training data. Due to short of space, we only show 4 cases. After we take more runs, we can see that it is significantly different.

The following figure (Fig. 2) shows that the testing accuracy drops down first, and then boost to higher than initial one.

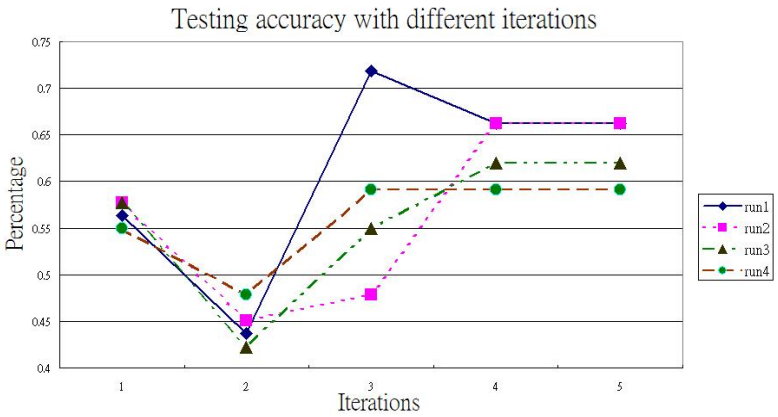

Figure 2. The testing classification accuracy varies with iterations.

In order to test for consensus results,we fixed the same training data and testing data for 3 classifiers, and then used majority voting to combine different classifier results. The results are shown in Tables 2 and 3.

Table 2. Combining 3 different classifiers by majority voting. In this training set and testing set, if we use all the training data to train the classifier, the testing accuracy is 0.5493. The three classifiers are generated from different initializations of the training set and the validation set.

Classifier	1		2		3		Consensus
Iteration	Valid.	Test.	Valid.	Test.	Valid.	Test.	Test.
1	0.54286	0.5493	0.57143	0.5493	0.6286	0.5493	0.5493
2	0.74286	0.46479	0.74286	0.46479	0.6857	0.4507	0.4507
3	1	0.60563	1	0.60563	1	0.60563	0.6056
4	0.97143	0.60563	1	0.60563	1	0.60563	0.6056

Table 3. Combining 3 different classifiers by majority voting. In this training set and testing set, if we use all the training data to train the classifier, the testing accuracy is 0.5634.

Classifier	1		2		3		Consensus
Iteration	Valid.	Test.	Valid.	Test.	Valid.	Test.	Test.
1	0.6000	0.5634	0.3143	0.4648	0.5429	0.5634	0.5634
2	0.6571	0.4507	0.7714	0.5634	0.7143	0.4507	0.4648
3	1	0.6479	0.9429	0.5775	1.0000	0.6620	0.6479
4	1	0.6479	0.9714	0.5775	1.0000	0.6620	0.6479

DISCUSSION AND CONCLUSIONS

From the results of the experiments, it is apparent that the resampling technique does generate a better training set to train the classifier, resulting in better classification accuracy.

Another approach would be to use all the training data to train the classifier, and then used the classifier to search for the misclassified vectors. However, then it is likely to get 100% training accuracy, meaning we do not know which samples to emphasize. Including validation sets works better due to this reason.

In some cases, we noticed that the testing accuracy was lower in iteration 2 or 3. However, the testing results always improved when we reached 100% validation accuracy in succeeding iterations.

In previous boosting methods, it is possible to overfit by running too many rounds. With our approach, we only add the misclassified validation samples to the training set. When 100% validation accuracy is reached, there are no more rounds to be used.

We also noticed that if the validation accuracy in the first iteration is not sufficiently high, such as better than 50%, it is a good idea to regenerate the initialization of the training and validation sets. This approach reduces the number of iterations to get the best training set.

We also considered rates of convergence. In all the experiments, the maximum validation accuracy is always reached within 5 iterations. This may take extra computation time compared to using all the training set to train one classifier, but the number of iterations to get better results is not excessive.

By using random initialization of the training set and the validation set, we can generate a number of different classifiers. The results from these classifiers can be combined, for example, by using majority voting to achieve better results. However, in our consensus experiments, the results did not improve further. This topic needs to be further investigated. We only generated 3 classifiers and then aggregated the results. It is possible that more classifiers would increase performance.

In the experiments, we chose p_{train} equal to 0.5 and p_{vali} eaul to 0.5. To estimate the optimal values of these parameters, we should do further research.

ACKNOWLEDGEMENT

This research was supported by NSF Grant MCB-9873139 and partly by NSF Grant #0325544.

REFERENCES

Joseph A. Cruz, and David S. Wishart, 2006, "Applications of Machine Learning in Cancer Prediction and Prognosis," *Cancer Informatics.*

Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim, Sung Yang Bang, 2003, "Constructing support vector machine ensemble," *Pattern Recognition* 36 pp. 2757 – 2767.

Leo Breiman, 1996, "Bagging Predictors" *Machine Learning* 24 (2) pp.123–140.

Y. Freund and R.E. Schapire, 1994, "A decision-theoretic generalization of on-line learning and an application to boosting," In Euro COLT: European Conference on Computational Learning Theory. LNCS.

Y. Freund and R.E. Schapire, 1996, "Experiments with a new boosting algorithm," In *Proceedings 13th International Conference on Machine Learning*, pp. 146-148. Morgan Kaufmann.

Y. Freund and R.E. Schapire,1997, "A decision-theoretic generalization of on-line learning and an application to boosting." *Journal of Computer and System Sciences*,55(1):119-139.

Igor Kononenko and Bojan Cestnik, 1988, Repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html , Irvine, CA: University of California, Department of Information and Computer Science.

B. E. Boser, I. M. Guyon, and V. N. Vapnik, 1992, "A training algorithm for optimal margin classifiers," D. Haussler, editor, 5th Annual ACM Workshop on COLT, pages 144-152, Pittsburgh, PA. ACM Press.

Thorsten Joachims, 2004, http://www.cs.cornell.edu/People/tj/svm_light/ .