AdaBoost and Support Vector Machines for Unbalanced Data Sets

Chi Zhang
Department of Electrical Engineering and Computer Science,
University of Tennessee, Knoxville, TN 37996, USA
Email: czhang24@utk.edu

Abstract—Boost is a kind of method for improving the accuracy of a given learning algorithm by combining multiple weak learners to “boost” into a strong learner. The gist of AdaBoost is based on the assumption that even though a weak learner cannot do good for all classifications, each of them is good at some subsets of the given data with certain bias, so that by assembling many weak learner together, the overall accuracy is expected to be higher.

Support Vector Machine (SVM) is a popular machine learning technique for solving classification and regression problems. In this project, LIBSVM tools of SVMs was used to solve classification problems. The AdaBoost.M1 algorithm utilized SVMs as component learners and the new algorithm was proved to boost the accuracy of unbalanced datasets sharply. In the best case, AdaBoost.M1 with SVM algorithm achieved accuracy improvement of 10%. However, AdaBoost was not always useful for performance boosting. In the worst case of the vowel dataset, the performance of AdaBoost.M1 with SVM was slightly worse than the grid search method. By exploring various aspects of AdaBoost.M1 with SVM algorithm, I found that the gamma update settings had an important impact on the accuracy. It effected the number of component learners as well as the generalization of each learner. Ideally, proper number of weak learners would fit in unbalanced training data very well.

I. INTRODUCTION

AdaBoost algorithm was introduced to solved many practically difficult problems in the earlier boosting algorithms. By calling a given weak learner repeatedly and changing the weight over all the training set, the difficult examples will gradually have higher weights so that the weak learner will focus on those hard examples in training[2]. In this project, SVMs were used as the weak learners by the Adaboost.M1 algorithm.

To compare the performance of SVMs with AdaBoost.M1 approach, the AdaBoost.M1 with SVM algorithm was implemented. During the experiments, I found that AdaBoost.M1 can boost the performance of the unbalanced datasets greatly but cannot help improve performance on the balanced datasets. Two kinds of SVM approaches offered by the LIBSVM were explored to get different results. The grid search integrated cross-validation to avoid overfitting and can automatically find the best C and γ values for SVM. The default SVMs had lowest classification accuracy among all the three methods.

Algorithm 1 Pseudo Code of AdaBoost.M1 with SVM algorithm

\begin{align*}
\text{Input: sequence of } m \text{ examples } (x_1, y_1), \ldots, (x_m, y_m), \text{ with labels } y_i \in \{1, \ldots, k\}, \text{ weak learning algorithm SVMs} \\
\text{Initialize } D_1(i) = 1/m \text{ for all } i \\
\text{Do while } (\sigma > \sigma_{\text{min}}) \\
&\quad (1) \text{ Train a SVM component classifier } h_t \text{ on the weighted training set} \\
&\quad (2) \text{ Compute the training error of } h_t : \epsilon_t = \sum_{i=1}^{N} w_i^t, \ y_i \neq h_t(x_i). \\
&\quad (3) \text{ If } \epsilon_t > 0.5, \text{ decrease } \sigma \text{ by } \sigma_{\text{step}}, \text{ and goto } (1). \\
&\quad (4) \text{ Set } \beta_t = \frac{1}{\epsilon_t}. \\
&\quad (5) \text{ Update } D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \\
&\quad \beta_t \text{ if } h_t(x_i) = y_i \\
&\quad 1 \text{ otherwise} \\
\end{cases} \text{ where } Z_t \text{ is a normalization constant} \\
\text{Output}: h_{\text{fin}}(x) = \arg\max_{y \in \{1, \ldots, k\}} \sum_{t} h_t(x) = y \log\frac{1}{\epsilon_t}. \\
\end{align*}

II. PROBLEM FORMULATION

A. Introduction of LIBSVM

LIBSVM is a popular tool for SVM classification and regression. It provides rich tools to address different aspects of SVM problem. grid.py is such a parameter selection tool in Libsvm for C-SVM classification using the RBF (radial basis function) kernel. It utilizes cross validation (CV) technique to eliminate overfitting and estimate the accuracy of each parameter combination in the specified range, so that helps user to decide the best parameters for given problem[1].

B. Design AdaBoost.M1 algorithm with SVM

Based on the existing SVM tools, the AdaBoost.M1 algorithm called SVMs as component learners at each training iteration, as Algorithm 1 showed.

III. EXPERIMENTS

A. Using LIBSVM grid search to find ideal C and γ values

Using the tool grid.py in LIBSVM, I tried various pairs of (C, γ) values. This tool enables exponentially growing sequences of C and γ to find good parameters with the best cross validation accuracy. In this section, I used the grid search tool on each of the three datasets to find the the best value of C and γ. A coarse grid was firstly applied to search for the roughly good parameter ranges. Then the fine grid restricted
searching within a smaller range around the good area to find the ideal C and γ for given training data.

1) Glass dataset: For glass dataset, I set the coarse search as \( \log_2 C \in [4, 15] \) and \( \log_2 γ \in [-6, 2] \), with step size 1. Figure 1 showed the results of accuracy was 75.56% when \( C = 512 \), \( γ = 0.5 \). Since the high accuracy area located around the center of Figure 1, the searching range was decreased to have a closer view of this area. Figure 2 showed the result of \( \log_2 C \in [8, 10] \) \( \log_2 γ \in [-2, 1] \) when step sizes were 1. At last, to find the best parameter, several find grid searches were executed with smaller step size and smaller range. The best testing accuracy achieved \( 76.4706\% \) with \( \log_2 C \in [9, 10] \) \( \log_2 γ \in [1, 3] \) and step size 0.05. The ideal parameters for glass dataset were \( C = 515.561241629 \), \( \gamma = 0.5 \), as Figure 3 showed. In the experiment, there may exist several high accuracy area, which had the same training and testing accuracy values. The one I picked here is just one of those best parameters.

2) Liver-disorders dataset: The coarse search for liver-disorders data was set to a wide rang of \( \log_2 C \in [0, 30] \) \( \log_2 γ \in [-20, 0] \) at the beginning. After trying for several times, I found that the speed of grid search for liver-disordered dataset was quite slow, so the step size was set to be 5 to speed up the coarse search while still in a large search space. Figure 4 was the snapshots of coarse grid search. Based on the rough area with high accuracy, I did fine grid searching in the rang of \( \log_2 C \in [0, 30] \) \( \log_2 γ \in [-20, 0] \), with step size 0.1. The result of the fine search was showed in Figure 5. After applying the optimal C value of 147.0338944 and gamma value of 0.054409410206 to the svm-train and svm-predict programs, the accuracy of liver-disordered testing dataset was 75.5556\% (34/45).

3) Vowel dataset: For vowel dataset, I set the initial coarse search range of \( \log_2 C \in [3, 20] \) \( \log_2 γ \in [-6, 5] \), with step size of 1. Since the high accuracy in Figure 6 located at around \( C = 4 \) and \( γ = 2 \), the fine grid searching focused on this area.

Figure 7 showed the fine search in area \( \log_2 C \in [1, 3] \) \( \log_2 γ \in [0, 3] \), with step size of 0.1.

The comparison of training accuracy of grid search and default LIBSVM was displayed in Table 1. From the results, we can observe that the grid search had higher training accuracy.
accuracy than the default LIBSVM. This is because the grid search applied cross-validation to eliminate overfitting and tried various C and γ combination to find the ideal parameters of SVMs for the given datasets. Note that the cross-validation here was set to be 5 fold.

The optimal parameters found by grid search for each dataset was displayed in Table 2. Those values were applied to AdaBoost.M1 with SVM algorithm in the next section.

B. Implement AdaBoost.M1 with SVMs as Component Learners

AdaBoost.M1 algorithm with SVMs as component learners was implemented based on the LIBSVM, as Algorithm 1 showed. A maximum iteration number of 1000 was set to avoid the growing number of component learners. Experiments indicated that too many component learners didn’t help to improve the performance and would result in long training time. 1000 was discovered to be a reasonable max learner size.

Table 3 showed the testing accuracy of all three methods. In this experiment, I used the ideal C values obtained from the grid search in Table 2, and varied the value of γ. One can see that the AdaBoost.M1 with SVMs method had better overall performance for glass dataset. It boosted the accuracy from 76% to 85%, which was a considerable improvement. However, for liver-disorders dataset, the increasement was not that obvious. Results showed that only it only had one more point correctly classified for the testing dataset. The worst case happened in the vowel dataset, which degraded the classification performance by 6%. However, the AdaBoost.M1 with SVMs still worked better than the default LIBSVM in all of the three datasets.

The confusion matrix for each class in each dataset was
also displayed to have a better view of what happened in the classifiers. For the liver-disordered dataset, Table 4 showed all examples were classified as class 2. It may be because the liver-disordered training set had low degree of unbalance so that AdaBoost.M1 didn’t help too much.

For the glass dataset, AdaBoost.M1 had a good performance in classifying most of the classes except class 5. The glass data was a typical unbalanced data because Class 3, 6 and 7 had much less training instances than Class 1 and 2. Consequently, the AdaBoost.M1 algorithm helped boost the overall classification accuracy greatly. From Table 5, it was obvious that all the three Classes of 3, 6 and 7 were correctly classified.

AdaBoost.M1 with SVM had the worst accuracy in the vowel dataset. The overall performance was less than 60%, worse than the grid search method. This phenomenon was not consistent with the performance in glass datasets. The reason was the very balanced training instances of vowel. Each class in vowel training set had exactly 42 examples. As discussed before, AdaBoost.M1 can achieve better performance on unbalanced data sets but lose its power for low degree of unbalanced data. It was also observed that if the number of the component learner was more than 1, the performance for AdaBoost.M1 would be lower than 50%.

In Table 7, the best performance I can get came from one component learner situation, which was still lower than the grid search solution. Due to the “weak” prediction ability of the component learner and balanced characteristic of the vowel dataset, it was reasonable to have this result. The accuracy was expected to be lower than the single SVM in the grid search because this single SVM might be stronger.

The above experiments indicated that the performance boosting AdaBoost.M1 can provide depended heavily on the degree of unbalance of data. If the data has high degree of unbalance, the AdaBoost.M1 can help a lot in boosting the accuracy, but if the training data was quite balanced, the AdaBoost would not help and may even result in worse performance.

The number of weak learners generated by the algorithm can be effected by the \( \gamma \) value range. In the experiments, I found that if \( \gamma \) was too large, the error would be very low, which means the generated learner was quite strong and the \( \gamma \) had less chance to be updated. Consequently, more component learners would be generated and sometimes it reached the maximum learner size of 1000. If most learners were strong, it was better to have a large number of component learners to enlarge the generalization and the diversity of bias, due to the low generalization and strong bias in strong learners.

The range of \( \gamma \) in this experiment was found by experience. For different dataset, the fitting range was different, as Table 7 showed.

### IV. DISCUSSION

#### A. Effect of number of component learners and \( \gamma \) range

An interesting observation of the number of component learner was that it can be effected by the \( \gamma \) settings and they together effected the classification performance. In Table 7, the \( \gamma \) ranges of different datasets were quite different. For glass dataset, if the initial \( \gamma \) was large, e.g., more than 5, the component learners would have very small error which means they were not “weak learners” anymore and the accuracy of training set was close to 100%, which might indicate the risk of overfitting. Then the AdaBoost.M1 generated more learners to try to obtain better generalization from those non-weak learners, which was not supposed to be the correct way of using AdaBoost.M1. Therefore, too large \( \gamma \) value resulted in poor performance.

#### B. Effect of different C

To explore the effect of the C value, I did experiments with different C in AdaBoost.M1. Results were showed in Table 9. One can see that for the glass dataset, there was an obvious high accuracy area around the ideal C, ranging from 514 to 525. Then the accuracy dropped down as the C value was
far away from the ideal C. For the liver-disordered dataset, situation was almost the same except that there were several local maximum points around the idea C, which means it needs more careful search to identify the global optimal C. Varying C had no effect on the vowel dataset because it had balanced training examples and AdaBoost.M1 couldn’t do anything to improve.

C. Undersample training data

Since the glass dataset was the best sample with unbalanced characteristic, I artificially undersampled the training data to test if the regular SVM without AdaBoost can achieve better performance. Each class in the glass training set was randomly reduced to 9 ~ 11 examples. Then a grid search was applied to find the optimal C and γ, and finally the SVM trained those undersampled data. Table 10 showed that the performance didn’t change. Intuitively, performance after decreasing the degree of unbalance should be better, but the results showed they were almost the same. This may be because the number of training instances was too small compared with the number of features, so that it was difficult to get a good classification accuracy with such few training examples.

D. Effect of feature selection in LIBSVM

Using the feature select tool in LIBSVM, performance can easily approach the grid search and better than the default SVM. Different from the boosting approach which emphasizes in the hard to learn instances, the feature selection emphasize in the important features to represent the training data. They both improve the classification performance, but from different aspects of data characteristics.

V. Conclusion

In this project, the AdaBoost.M1 algorithm was implemented to compare with the grid search and default SVM methods. According to the results, we can draw the conclusion that AdaBoost.M1 was effective for high degree of unbalanced dataset, but couldn’t help to improve the performance for balanced data.

The number of component learners generated by AdaBoost.M1 was related with the γ value and had impact in the classification performance. As the γ value was very large, the component learners were not weak learners so generalization was lost, which resulted in a large number of non-weak learners. Consequently, the overall accuracy was not good. On the other hand, if the γ was too small, there would be insufficient weak learners so that the algorithm cannot fully represented the training data, which also leads to failure of performance boosting. So the γ value should be carefully selected for different training datasets.

To explore the effect of C, I also did experiment with different C values. Results showed that the accuracy was high in a quite large area of C. This was the reason to fix the C value while varying γ in the AdaBoost.M1 algorithm.

Feature selection was also investigated by applying the LIBSVM feature selection tool to the given datasets. Almost the same performance gain can be obtained as using grid search. It indicated that there were multiple ways to boost performance, as long as the classifiers can represent data more wisely.

### References
