AdaBoost with SVM-based component classifiers

Xuchun II
Nanyang Technological University, Singapore

Lei Wang
Nanyang Technological University, Singapore, leiw@uow.edu.au

Eric Sung
Nanyang Technological University, Singapore

Publication Details
AdaBoost with SVM-based component classifiers

Abstract
The use of SVM (Support Vector Machine) as component classifier in AdaBoost may seem like going against the grain of the Boosting principle since SVM is not an easy classifier to train. Moreover, Wickramaratna et al. [2001. Performance degradation in boosting. In: Proceedings of the Second International Workshop on Multiple Classifier Systems, pp. 11-21] show that AdaBoost with strong component classifiers is not viable. In this paper, we shall show that AdaBoost incorporating properly designed RBFSVM (SVM with the RBF kernel) component classifiers, which we call AdaBoostSVM, can perform as well as SVM. Furthermore, the proposed AdaBoostSVM demonstrates better generalization performance than SVM on imbalanced classification problems. The key idea of AdaBoostSVM is that for the sequence of trained RBFSVM component classifiers, starting with large s values (implying weak learning), the s values are reduced progressively as the Boosting iteration proceeds. This effectively produces a set of RBFSVM component classifiers whose model parameters are adaptively different manifesting in better generalization as compared to AdaBoost approach with SVM component classifiers using a fixed (optimal) s value. From benchmark data sets, we show that our AdaBoostSVM approach outperforms other AdaBoost approaches using component classifiers such as Decision Trees and Neural Networks. AdaBoostSVM can be seen as a proof of concept of the idea proposed in Valentini and Dietterich [2004. Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods. Journal of Machine Learning Research 5, 725-775] that Adaboost with heterogeneous SVMs could work well. Moreover, we extend AdaBoostSVM to the Diverse AdaBoostSVM to address the reported accuracy/diversity dilemma of the original Adaboost. By designing parameter adjusting strategies, the distributions of accuracy and diversity over RBFSVM component classifiers are tuned to maintain a good balance between them and promising results have been obtained on benchmark data sets.

Keywords
classifiers, adaboost, svm, component

Disciplines
Engineering | Science and Technology Studies

Publication Details

This journal article is available at Research Online: http://ro.uow.edu.au/eispapers/602
AdaBoost with SVM-based component classifiers

Xuchun Li*, Lei Wang, Eric Sung

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore

Received 28 May 2006; received in revised form 22 May 2007; accepted 13 July 2007
Available online 14 September 2007

Abstract

The use of SVM (Support Vector Machine) as component classifier in AdaBoost may seem like going against the grain of the Boosting principle since SVM is not an easy classifier to train. Moreover, Wickramaratna et al. [2001. Performance degradation in boosting. In: Proceedings of the Second International Workshop on Multiple Classifier Systems, pp. 11–21] show that AdaBoost with strong component classifiers is not viable. In this paper, we shall show that AdaBoost incorporating properly designed RBFSVM (SVM with the RBF kernel) component classifiers, which we call AdaBoostSVM, can perform as well as SVM.

Furthermore, the proposed AdaBoostSVM demonstrates better generalization performance than SVM on imbalanced classification problems. The key idea of AdaBoostSVM is that for the sequence of trained RBFSVM component classifiers, starting with large $s$ values (implying weak learning), the $s$ values are reduced progressively as the Boosting iteration proceeds. This effectively produces a set of RBFSVM component classifiers whose model parameters are adaptively different manifesting in better generalization as compared to AdaBoost approach with SVM component classifiers using a fixed (optimal) $s$ value. From benchmark data sets, we show that our AdaBoostSVM approach outperforms other AdaBoost approaches using component classifiers such as Decision Trees and Neural Networks. AdaBoostSVM can be seen as a proof of concept of the idea proposed in Valentini and Dietterich [2004. Bias-variance analysis of support vector machines for the development of SVM-based ensemble methods. Journal of Machine Learning Research 5, 725–775] that Adaboost with heterogeneous SVMs could work well. Moreover, we extend AdaBoostSVM to the Diverse AdaBoostSVM to address the reported accuracy/diversity dilemma of the original Adaboost. By designing parameter adjusting strategies, the distributions of accuracy and diversity over RBFSVM component classifiers are tuned to maintain a good balance between them and promising results have been obtained on benchmark data sets.

© 2007 Published by Elsevier Ltd.

Keywords: AdaBoost; Support Vector Machine; Component classifier; Diversity

1. Introduction

One of the major developments in machine learning in the past decade is the Ensemble method, which finds a highly accurate classifier by combining many moderately accurate component classifiers. Two of the commonly used techniques for constructing Ensemble classifiers are Boosting (Schapire, 2002) and Bagging (Breiman, 1996). Compared with Bagging, Boosting performs better when the data do not have much noise (Opitz and Maclin, 1999; Bauer and Kohavi, 1999). As the most popular Boosting method, AdaBoost (Freund and Schapire, 1997) creates a collection of component classifiers by maintaining a set of weights over training samples and adaptively adjusting these weights after each Boosting iteration: the weights of the training samples which are misclassified by current component classifier will be increased while the weights of the training samples which are correctly classified will be decreased. Several ways have been proposed to implement the weight update in AdaBoost (Kuncheva and Whitaker, 2002).

The success of AdaBoost can be attributed to its ability to enlarge the margin (Schapire et al., 1998), which could enhance the generalization capability of AdaBoost. Many studies that use Decision Trees (Dietterich, 2000) or Neural Networks (Schwenk and Bengio, 2000; Ratsch, 2001) as component classifiers in AdaBoost have been reported. These studies show good generalization performance of
these AdaBoost. Still, some difficulties remain. When Decision Trees are used as component classifiers, what should be the suitable tree size? When Radial Basis Function (RBF) Neural Networks are used as component classifiers, how could the complexity be controlled to avoid overfitting? Moreover, we have to decide on the optimum number of centers and the width of the RBFs? All of these have to be carefully tuned for AdaBoost to achieve better performance. Furthermore, diversity is known to be an important factor which affects the generalization performance of Ensemble classifiers (Melville and Mooney, 2005; Kuncheva and Whitaker, 2003). Some methods are proposed to quantify the diversity (Kuncheva and Whitaker, 2003; Windeatt, 2005). It is also known that there is an accuracy/diversity dilemma in AdaBoost (Dietterich, 2000), which means that the more accurate the two component classifiers become, the less they can disagree with each other. Only when the accuracy and diversity are well balanced, can the AdaBoost demonstrate excellent generalization performance. However, the existing AdaBoost algorithms do not explicitly take sufficient measures to deal with this problem.

Support Vector Machine (SVM) (Vapnik, 1998) is developed from the theory of Structural Risk Minimization. By using a kernel trick to map the training samples from an input space to a high-dimensional feature space, SVM finds an optimal separating hyperplane in the feature space and uses a regularization parameter, $C$, to control its model complexity and training error. One of the popular kernels used in SVM is the RBF kernel, which has a parameter known as Gaussian width, $\sigma$. In contrast to the RBF networks, SVM with the RBF kernel (RBFSVM in short) can automatically determine the number and location of the centers and the weight values (Scholkopf et al., 1997). Also, it can effectively avoid overfitting by selecting proper values of $C$ and $\sigma$. From the performance analysis of RBFSVM (Valentini and Dietterich, 2004), we know that $\sigma$ is a more important parameter compared to $C$: although RBFSVM cannot learn well when a very low value of $C$ is used, its performance largely depends on the $\sigma$ value if a roughly suitable $C$ is given. This means that, over a range of suitable $C$, the performance of RBFSVM can be changed by simply adjusting the value of $\sigma$.

Therefore, in this paper, we try to answer the following questions: Can the SVM be used as an effective component classifier in AdaBoost? If yes, what will be the generalization performance of this AdaBoost? Will this AdaBoost show some advantages over the existing ones, especially on the aforementioned problems? Furthermore, compared with the individual SVM, what is the benefit of using an AdaBoost as a combination of multiple SVMs? In this paper, RBFSVM is adopted as component classifier for AdaBoost. As mentioned above, there is a parameter $\sigma$ in RBFSVM which has to be set beforehand. An intuitive way is to simply apply a single $\sigma$ to all RBFSVM component classifiers. However, we observed that this way cannot lead to successful AdaBoost due to the over-weak or over-strong RBFSVM component classifiers encountered in Boosting process. Although there may exist a single best $\sigma$, we find that AdaBoost with this single best $\sigma$ obtained by cross-validation cannot lead to the best generalization performance and also doing cross-validation for it will increase the computational load. Therefore, using a single $\sigma$ in all RBFSVM component classifiers should be avoided if possible.

The following fact opens the door for us to avoid searching the single best $\sigma$ and help AdaBoost achieve even better generalization performance. It is known that the classification performance of RBFSVM can be conveniently changed by adjusting the kernel parameter, $\sigma$. Enlightened by this, the proposed AdaBoostSVM approach adaptively adjusts the $\sigma$ values in RBFSVM component classifiers to obtain a set of moderately accurate RBFSVMs for AdaBoost. As will be shown later, this gives rise to a better SVM-based AdaBoost. Compared with the existing AdaBoost approaches with Neural Networks or Decision Tree component classifiers, our proposed AdaBoostSVM can achieve better generalization performance and it can be seen as a proof of concept of the idea suggested by Valentini and Dietterich (2004) that AdaBoost with heterogeneous SVMs could work well. Furthermore, compared with individual SVM, AdaBoostSVM can achieve much better generalization performance on imbalanced data sets. We argue that in AdaBoostSVM, the Boosting mechanism forces some RBFSVM component classifiers to focus on the misclassified samples from the minority class, and this can prevent the minority class from being considered as noise in the dominant class and be wrongly classified. This also justifies, from another perspective, the significance of exploring AdaBoost with SVM component classifiers.

Furthermore, since AdaBoostSVM provides a convenient way to control the classification accuracy of each RBFSVM component classifier by simply adjusting the $\sigma$ value, it also provides an opportunity to deal with the well-known accuracy/diversity dilemma in Boosting methods. This is a happy “discovery” found during the investigation of AdaBoost with RBFSVM-based component classifiers. Through some parameter adjusting strategies, we can tune the distributions of accuracy and diversity over these component classifiers to achieve a good balance. We also propose an improved version of AdaBoostSVM called Diverse AdaBoostSVM in this paper. It is observed that, benefiting from the balance between accuracy and diversity, it can give better generalization performance than AdaBoostSVM.

2. Background

2.1. AdaBoost

Given a set of training samples, AdaBoost (Schapire and Singer, 1999) maintains a weight distribution, $W$, over these samples. This distribution is initially set uniform.
Table 1

Algorithm: AdaBoost (Schapire and Singer, 1999)
1. Input: a set of training samples with labels \( ((x_1, y_1), \ldots, (x_N, y_N)) \), a ComponentLearn algorithm, the number of cycles \( T \).
2. Initialize: the weights of training samples: \( w_i^0 = 1/N \), for all \( i = 1, \ldots, N \).
3. Do for \( t = 1, \ldots, T \):
   (1) Use the ComponentLearn algorithm to train a component classifier, \( h_t \), on the weighted training samples.
   (2) Calculate the training error of \( h_t \): \( \varepsilon_t = \sum_{i=1}^{N} w_i^t \cdot y_i \cdot h_t(x_i) \).
   (3) Set weight for the component classifier \( h_t \): \( w_t = \frac{1}{2} \ln \left( \frac{1-\varepsilon_t}{\varepsilon_t} \right) \).
   (4) Update the weights of training samples: \( w_i^{t+1} = \frac{w_i^t \exp (-y_i h_t(x_i))}{C_t} \), for \( i = 1, \ldots, N \), where \( C_t \) is a normalization constant, and \( \sum_{i=1}^{N} w_i^{t+1} = 1 \).
4. Output: \( f(x) = \text{sign} \left( \sum_{t=1}^{T} w_t h_t(x) \right) \).

Then, AdaBoost calls ComponentLearn algorithm repeatedly in a series of cycles (Table 1). At cycle \( t \), AdaBoost provides training samples with a distribution \( W_t \) to ComponentLearn. In response, the ComponentLearn trains a classifier \( h_t \). The distribution \( W_t \) is updated after each cycle according to the prediction results on the training samples. “Easy” samples that are correctly classified \( h_t \) get lower weights, and “hard” samples that are misclassified get higher weights. Thus, AdaBoost focuses on the samples with higher weights, which seem to be harder for ComponentLearn. This process continues for \( T \) cycles, and finally, AdaBoost linearly combines all the component classifiers into a single final hypothesis \( f \). Greater weights are given to component classifiers with lower training errors. The important theoretical property of AdaBoost is that if the component classifiers consistently have accuracy only slightly better than half, then the training error of the final hypothesis drops to zero exponentially fast. This means that the component classifiers need to be only slightly better than random.

2.2. Support Vector Machine

SVM was developed from the theory of Structural Risk Minimization. In a binary classification problem, the decision function of SVM is

\[
f(x) = \langle w, \phi(x) \rangle + b,
\]

where \( \phi(x) \) is a mapping of sample \( x \) from the input space to a high-dimensional feature space, \( \langle \cdot, \cdot \rangle \) denotes the dot product in the feature space. The optimal values of \( w \) and \( b \) can be obtained by solving the following optimization problem:

\[
\text{minimize: } g(w, \xi) = \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} \xi_i
\]

subject to: \( y_i (\langle w, \phi(x_i) \rangle + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \),

where \( \xi_i \) is the \( i \)th slack variable and \( C \) is the regularization parameter. According to the Wolfe dual form, the above minimization problem can be written as

\[
\text{minimize: } W(z) = -\sum_{i=1}^{N} z_i + \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} y_i y_j z_i z_j k(x_i, x_j)
\]

subject to: \( \sum_{i=1}^{N} y_i z_i = 0, \quad \forall i : 0 \leq z_i \leq C \),

where \( z_i \) is a Lagrange multiplier which corresponds to the sample \( x_i \), \( k(\cdot, \cdot) \) is a kernel function that implicitly maps the input vectors into a suitable feature space

\[
k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle.
\]

Compared with RBF networks (Scholkopf et al., 1997), SVM automatically calculates the number and location of centers, weights, and thresholds in the following way: by the use of a suitable kernel function (in this paper, the RBF kernel, \( k(x_i, x_j) = \exp(-\|x_i - x_j\|^2/2\sigma^2) \), is used), the samples are mapped nonlinearly into a high-dimensional feature space. In this space, an optimal separating hyperplane is constructed by the support vectors. Support vectors correspond to the centers of RBF kernels in the input space. The generalization performance of SVM is mainly affected by the kernel parameters, for example, \( \sigma \), and the regularization parameter, \( C \). They have to be set beforehand.

3. Proposed algorithm: AdaBoostSVM

This work aims to employ RBFSVM as component classifier in AdaBoost. But how should we set the \( \sigma \) value for these RBFSVM component classifiers during the AdaBoost iterations? Problems are encountered when applying a single \( \sigma \) to all RBFSVM component classifiers. In detail, having too large a value of \( \sigma \) often results in too weak a RBFSVM component classifier. Its classification accuracy is often less than 50% and cannot meet the requirement on a component classifier given in AdaBoost. On the other hand, a smaller \( \sigma \) often makes the RBFSVM component classifier stronger and boosting them may become inefficient because the errors of these component classifiers are highly correlated. Furthermore, too small a value of \( \sigma \) can even make RBFSVM overfit the training samples. Hence, finding a suitable \( \sigma \) for these SVM component classifiers in AdaBoost becomes a problem. By using model selection techniques such as \( k \)-fold or leave-one-out cross-validation, a single best \( \sigma \) may be found for these component classifiers. However, the process of model selection is time consuming and should be avoided if possible. Hence, it seems that SVM component classifiers do not perform optimally if only one single value of \( \sigma \) is used.
3.1. Influence of parameters on SVM performance

The classification performance of SVM is affected by its model parameters. For RBFSVM, the model parameters include the Gaussian width, $\sigma$, and the regularization parameter, $C$. The variation of either of them leads to the change of classification performance. However, as reported in Valentini and Dietterich (2004), although RBFSVM cannot learn well when a very low value of $C$ is used, its performance largely depends on the $\sigma$ value if a roughly suitable $C$ is given. An example is shown in Fig. 1, which plots the test error of SVM against the values of $C$ and $\sigma$ on a non-separable data set used in Baudat and Anouar (2000). Clearly, changing $\sigma$ leads to larger variation on test error than changing $C$. This means that, over a large range of $C$, the performance of RBFSVM can be adjusted by simply changing the value of $\sigma$. It is known that, in a certain range, a larger $\sigma$ often leads to a reduction in classifier complexity but at the same lowers the classification performance. Also, a smaller $\sigma$ often increases the learning complexity and leads to higher classification performance largely depends on the $C$ parameter, $C$.

3.2. Proposed algorithm: AdaBoostSVM

When applying Boosting method to strong component classifiers, these component classifiers must be appropriately weakened in order to benefit from Boosting (Dietterich, 2000). Hence, if RBFSVM is used as component classifier in AdaBoost, a relatively large $\sigma$ value, which corresponds to a RBFSVM with relatively weak learning ability, is preferred. In the proposed AdaBoostSVM, without loss of generality, the re-weighting technique is used to update the weights of training samples. AdaBoostSVM can be described as follows (Table 2): Initially, a large value is set to $\sigma$, corresponding to a RBFSVM classifier with very weak learning ability. Then, RBFSVM with this $\sigma$ is trained as many cycles as possible as long as more than half accuracy can be obtained. Otherwise, this $\sigma$ value is decreased slightly to increase the learning capability of RBFSVM to help it achieve more than half accuracy. By decreasing the $\sigma$ value slightly, this prevents the new RBFSVM from being too strong for the current weighted training samples, and thus moderately accurate RBFSVM component classifiers are obtained. The reason why moderately accurate RBFSVM component classifiers are favored lies in the fact that these classifiers often have larger diversity than those component classifiers which are very accurate. These larger diversities may lead to a better generalization performance of AdaBoost. This process

![Fig. 1. The test error of SVM vs. $\sigma$ and $C$ values.](image)

Table 2

<table>
<thead>
<tr>
<th>Algorithm: AdaBoostSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Input: a set of training samples with labels ${(x_1, y_1), \ldots, (x_N, y_N)}$; the initial $\sigma, \sigma_{\text{ini}}$; the minimal $\sigma, \sigma_{\text{min}}$; the step of $\sigma, \sigma_{\text{step}}$.</td>
</tr>
<tr>
<td>2. Initialize: the weights of training samples: $w_t^i = 1/N$, for all $i = 1, \ldots, N$.</td>
</tr>
<tr>
<td>3. Do While ($\sigma &gt; \sigma_{\text{min}}$)</td>
</tr>
<tr>
<td>(1) Train a RBFSVM component classifier, $h_t$, on the weighted training set.</td>
</tr>
<tr>
<td>(2) Calculate the training error of $h_t$: $e_t = \sum_{i=1}^N w_t^i \cdot y_i \neq h_t(x_i)$.</td>
</tr>
<tr>
<td>(3) If $e_t &gt; 0.5$, decrease $\sigma$ value by $\sigma_{\text{step}}$ and goto (1).</td>
</tr>
<tr>
<td>(4) Set the weight of component classifier $h_t$: $z_t = \frac{1}{\sigma_t} \ln \left( \frac{1-e_t}{e_t} \right)$.</td>
</tr>
<tr>
<td>(5) Update the weights of training samples: $w_t^{i+1} = \frac{w_t^i \exp(-y_i h_t(x_i))}{\sum_{i=1}^N w_t^i \exp(-y_i h_t(x_i))}$,</td>
</tr>
<tr>
<td>where $C_t$ is a normalization constant, and $\sum_{i=1}^N w_t^{i+1} = 1$.</td>
</tr>
<tr>
<td>4. Output: $f(x) = \text{sign}(\sum_{t=1}^T z_t h_t(x))$.</td>
</tr>
</tbody>
</table>
continues until the $\sigma$ is decreased to the given minimal value.

4. Improvement: diverse AdaBoostSVM

4.1. Accuracy/diversity dilemma of AdaBoost

Diversity is known to be an important factor affecting the generalization performance of Ensemble methods (Melville and Mooney, 2005; Kuncheva and Whitaker, 2003), which means that the errors made by different component classifiers should be uncorrelated. If each component classifier is moderately accurate and these component classifiers largely disagree with each other, the uncorrelated errors of these component classifiers will be removed by the voting process so as to achieve good generalization performance (Shin and Sohn, 2005). This also applies to AdaBoost. As mentioned before, there exists a dilemma in AdaBoost between the classification accuracy of component classifiers and the diversity among them (Dietterich, 2000), which means that the more accurate the two component classifiers become, the less they can disagree with each other. The Accuracy–Diversity diagram in Fig. 2 is used to explain this dilemma. Following Margineantu and Dietterich (1997) and Domeniconi and Yan (2004), the Kappa statistic is used to measure the diversity.

This diagram is a scatter-plot where each point corresponds to a component classifier. The $x$ coordinate value of a point is the diversity value of the corresponding component classifier while the $y$ coordinate value is the accuracy value of the corresponding component classifier. From this figure, it can be observed that, if the component classifiers are too accurate, it is difficult to find very diverse ones, and combining these accurate but non-diverse classifiers often leads to very limited improvement (Wind-icatt, 2005). On the other hand, if the component classifiers are too inaccurate, although we can find diverse ones, the combination result may be worse than that of combining both more accurate and diverse component classifiers. This is because if the combination result is dominated by too many inaccurate component classifiers, it will be wrong most of the time, leading to poor classification result. Hence, how could we maximize the diversity under the condition of obtaining a fairly good component classifier accuracy in AdaBoost?

In AdaBoostSVM, the obtained RBFSVM component classifiers are mostly moderately accurate, which gives chances to select more diverse component classifiers. As aforementioned, through adjustment of the $\sigma$ value, a set of RBFSVM component classifiers with different learning abilities is obtained. This provides an opportunity of selecting more diverse component classifiers from this set to deal with the accuracy/diversity dilemma. Hence, we proposed a Diverse AdaBoostSVM approach (Table 3), and it is hoped to further improve the generalization performance of AdaBoostSVM.

4.2. Improvement of AdaBoostSVM: diverse AdaBoostSVM

Although the matter of how diversity is measured and used in Ensemble methods is still an open problem (Kuncheva and Whitaker, 2003), some promising results (Melville and Mooney, 2005; Dasgupta and Long, 2003) have been reported recently. By increasing the diversity of component classifiers, these methods can achieve higher generalization accuracy. In the proposed Diverse AdaBoostSVM approach, we use the definition of diversity in Melville and Mooney (2005), which measures the disagreement between one component classifier and all the existing component classifiers. In the Diverse AdaBoostSVM, the diversity is calculated as follows: If $h_t(x_i)$ is the prediction label of the $t$th component classifier on the sample $x_i$, and $c_t$ is a normalization constant, and $\sum_{i=1}^{N} w_i^{t+1} = 1$.

![Fig. 2. Accuracy and diversity dilemma of AdaBoost.](image)

### Table 3

<table>
<thead>
<tr>
<th>Algorithm: Diverse AdaBoostSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Input: a set of training samples with labels $({x_1, y_1}, \ldots, {x_N, y_N})$; the initial $\sigma$, $\sigma_{\text{min}}$; the minimal $\sigma$, $\sigma_{\text{init}}$; the step of $\sigma$, $\sigma_{\text{step}}$; the threshold on diversity $\text{DIV}$.</td>
</tr>
<tr>
<td>2. Initialize: the weights of training samples: $w_i^1 = 1/N$, for all $i = 1, \ldots, N$.</td>
</tr>
<tr>
<td>3. Do While ($\sigma &gt; \sigma_{\text{init}}$)</td>
</tr>
<tr>
<td>(1) Train a RBFSVM component classifier, $h_t$, on the weighted training set.</td>
</tr>
<tr>
<td>(2) Calculate the training error of $h_t$: $e_t = \sum_{i=1}^{N} w_i^t$, $y_i \neq h_t(x_i)$.</td>
</tr>
<tr>
<td>(3) Calculate the diversity of $h_t$: $d_t = \sum_{i=1}^{N}</td>
</tr>
<tr>
<td>(4) If $e_t &gt; 0.5$ or $d_t &lt; \text{DIV}$, decrease $\sigma$ by $\sigma_{\text{step}}$ and goto (1).</td>
</tr>
<tr>
<td>(5) Set the weight of component classifier $h_t$: $w_i^t = \frac{1}{C_t} \frac{1}{\exp(-\sigma t) \times d_t(x_i)}$.</td>
</tr>
<tr>
<td>(6) Update the weights of training samples: $w_i^{t+1} = \frac{w_i^t \exp(-\sigma t / d_t(x_i))}{C_t}$, where $C_t$ is a normalization constant, and $\sum_{i=1}^{N} w_i^{t+1} = 1$.</td>
</tr>
<tr>
<td>4. Output: $f(x) = \text{sign}(\sum_{i=1}^{T} x_t h_t(x))$.</td>
</tr>
</tbody>
</table>
\( f(x_i) \) is the combined prediction label of all the existing component classifiers, the diversity of the 4th component classifier on the sample \( x_i \) is calculated as:

\[
d_i(x_i) = \begin{cases} 
0 & \text{if } h_1(x_i) = f(x_i) \\
1 & \text{if } h_1(x_i) \neq f(x_i)
\end{cases}
\]

(7)

and the diversity of \( AdaBoostSVM \) with \( T \) component classifiers on \( N \) samples is calculated as

\[
D = \frac{1}{TN} \sum_{i=1}^{T} \sum_{j=1}^{N} d_i(x_i).
\]

(8)

At each cycle of \( Diverse AdaBoostSVM \), the diversity value, \( D \), is calculated first. If \( D \) is larger than the predefined threshold, \( DIV \), this new RBFSVM component classifier will be selected. Otherwise, this component classifier will be discarded. Through this mechanism, a set of moderately accurate and diverse RBFSVM component classifiers can be generated. This is different from the above \( AdaBoostSVM \) which simply takes all the available RBFSVM component classifiers. As seen from the following experimental results, the \( Diverse AdaBoostSVM \) gives better generalization performance. We think that the improvement is due to its explicit dealing with the accuracy/diversity dilemma.

5. Experimental results

In this section, our proposed \( AdaBoostSVM \) and \( Diverse AdaBoostSVM \) are compared with the commonly used AdaBoost, which takes Neural Networks or Decision Tree as component classifiers. Then, they are compared with several state-of-the-art imbalanced classification algorithms to show the significance of exploring SVM-based AdaBoost algorithms applied to some imbalanced data sets.

5.1. Comparison on benchmark data sets

5.1.1. Data set information and parameter setting

Thirteen benchmark data sets from UCI Repository, DELVE, and STATLOG are used to evaluate the generalization performance of two proposed algorithms. The dimensions of these data sets range from 2 to 60, the numbers of training samples range from 140 to 1300, and the numbers of test samples range from 75 to 7000. Detailed information about these data sets can be found in (http://mlg.anu.edu.au/~raetsch/data). Each data set is partitioned into training and test subsets, usually in the ratio of 60–40%. 100 such partitions are generated randomly for the experiments. On each partition, the compared algorithms are trained and tested, respectively. The final performance of each algorithm on a data set is the average of the results over the 100 partitions.

As the generalization performance of RBFSVM is mainly affected by the parameter, \( \sigma \), the regularization parameter, \( C \), is empirically set as a value within 10–100 for all experiments. The \( \sigma_{\text{min}} \) is set as the average minimal distance between any two training samples and the \( \sigma_{\text{maj}} \) is set as the scatter radius of the training samples in the input space. Although the value of \( \sigma_{\text{step}} \) affects the number of \( AdaBoostSVM \) learning cycles, it has less impact on the final generalization performance, as shown later. Therefore, \( \sigma_{\text{step}} \) is set to a value within 1–3. The threshold “\( DIV \)” in the \( Diverse AdaBoostSVM \) is set as \( \eta \cdot D_{\text{max}}' \), where \( \eta \in [0,1] \) and \( D_{\text{max}}' \) denotes the maximal diversity value obtained in past \( t \) cycles. In this experiment, \( \eta = 0.7 \) is used to handle the possible small variation on diversity.

5.1.2. Evaluation of the generalization performance

Firstly, we give the generalization errors with standard deviation of the six algorithms on the benchmark data sets in Table 4: AdaBoost with Decision Tree component classifier (\( AB_{DT} \)), AdaBoost with Neural Networks component classifiers (\( AB_{SN} \)), AdaBoost with SVM component classifiers using single best parameters for SVM (\( AB_{SVM-s} \)), proposed AdaBoost SVM (\( AB_{SVM} \)), proposed Diverse AdaBoost SVM (\( DAB_{SVM} \)) and SVM.

<table>
<thead>
<tr>
<th>Data set</th>
<th>( AB_{DT} )</th>
<th>( AB_{SN} )</th>
<th>( AB_{SVM-s} )</th>
<th>( AB_{SVM} )</th>
<th>( DAB_{SVM} )</th>
<th>SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Banana</td>
<td>13.2 ± 0.7</td>
<td>12.3 ± 0.7</td>
<td>14.2 ± 0.6</td>
<td>12.1 ± 1.7</td>
<td>11.3 ± 1.4</td>
<td>11.5 ± 0.7</td>
</tr>
<tr>
<td>B. cancer</td>
<td>32.3 ± 4.7</td>
<td>30.4 ± 4.7</td>
<td>30.4 ± 4.4</td>
<td>25.5 ± 5.0</td>
<td>24.8 ± 4.4</td>
<td>26.0 ± 4.7</td>
</tr>
<tr>
<td>Diabetes</td>
<td>27.8 ± 2.3</td>
<td>26.5 ± 2.3</td>
<td>24.8 ± 2.0</td>
<td>24.8 ± 2.3</td>
<td>24.3 ± 2.1</td>
<td>23.5 ± 1.7</td>
</tr>
<tr>
<td>German</td>
<td>29.3 ± 2.4</td>
<td>27.5 ± 2.5</td>
<td>25.8 ± 1.9</td>
<td>23.4 ± 2.1</td>
<td>22.3 ± 2.1</td>
<td>23.6 ± 2.1</td>
</tr>
<tr>
<td>Heart</td>
<td>21.5 ± 3.3</td>
<td>20.3 ± 3.4</td>
<td>19.2 ± 3.5</td>
<td>15.5 ± 3.4</td>
<td>14.9 ± 3.0</td>
<td>16.0 ± 3.3</td>
</tr>
<tr>
<td>Image</td>
<td>3.7 ± 0.8</td>
<td>2.7 ± 0.7</td>
<td>6.2 ± 0.7</td>
<td>2.7 ± 0.7</td>
<td>2.4 ± 0.5</td>
<td>3.0 ± 0.6</td>
</tr>
<tr>
<td>Ringnorm</td>
<td>2.5 ± 0.3</td>
<td>1.9 ± 0.3</td>
<td>5.1 ± 0.2</td>
<td>2.1 ± 1.1</td>
<td>2.0 ± 0.7</td>
<td>1.7 ± 0.1</td>
</tr>
<tr>
<td>F. solar</td>
<td>37.9 ± 1.5</td>
<td>35.7 ± 1.8</td>
<td>36.8 ± 1.5</td>
<td>33.8 ± 1.5</td>
<td>33.7 ± 1.4</td>
<td>32.4 ± 1.8</td>
</tr>
<tr>
<td>Splice</td>
<td>12.0 ± 0.3</td>
<td><strong>10.1 ± 0.5</strong></td>
<td>14.3 ± 0.5</td>
<td>11.1 ± 1.2</td>
<td>11.0 ± 1.0</td>
<td>10.9 ± 0.7</td>
</tr>
<tr>
<td>Thyroid</td>
<td>5.6 ± 2.0</td>
<td>4.4 ± 2.2</td>
<td>8.5 ± 2.1</td>
<td>4.4 ± 2.1</td>
<td>3.7 ± 2.1</td>
<td>4.8 ± 2.2</td>
</tr>
<tr>
<td>Titanic</td>
<td>23.8 ± 0.7</td>
<td>22.6 ± 1.2</td>
<td>25.6 ± 1.2</td>
<td>22.1 ± 1.9</td>
<td>21.8 ± 1.5</td>
<td>22.4 ± 1.0</td>
</tr>
<tr>
<td>Twnorm</td>
<td>3.5 ± 0.2</td>
<td>3.0 ± 0.3</td>
<td>5.7 ± 0.3</td>
<td>2.6 ± 0.6</td>
<td>2.5 ± 0.5</td>
<td>3.0 ± 0.2</td>
</tr>
<tr>
<td>Waveform</td>
<td>12.1 ± 0.6</td>
<td>10.8 ± 0.6</td>
<td>12.7 ± 0.4</td>
<td>10.3 ± 1.7</td>
<td>10.2 ± 1.2</td>
<td><strong>9.9 ± 0.4</strong></td>
</tr>
<tr>
<td>Average</td>
<td>17.4 ± 1.5</td>
<td>16.0 ± 1.6</td>
<td>17.7 ± 1.5</td>
<td>14.6 ± 1.9</td>
<td><strong>14.2 ± 1.7</strong></td>
<td>14.5 ± 1.5</td>
</tr>
</tbody>
</table>
classifier (ABNN), AdaBoost with SVM component classifier using cross-validated single best parameters for SVM (ABSVM), proposed AdaBoostSVM (ABSVM), proposed Diverse AdaBoostSVM (DABSVM) and standard SVM.

Next, the McNemar’s statistical test (Eveitt, 1977) is done to confirm whether proposed algorithms outperform others on these data sets. For a data set, the McNemar’s statistical test of algorithm a and algorithm b is based on the following values of these two algorithms:

\[ N_{00}: \text{number of test data misclassified by both algorithm a and algorithm b} \]
\[ N_{01}: \text{number of test data misclassified by algorithm a but not by algorithm b} \]
\[ N_{10}: \text{number of test data misclassified by algorithm b but not by algorithm a} \]
\[ N_{11}: \text{number of test data misclassified by neither algorithm a nor algorithm b} \]

then the following statistic is calculated:

\[
\frac{(|N_{01} - N_{10}| - 1)^2}{N_{01} + N_{10}}. \quad (9)
\]

If algorithm a and algorithm b perform significantly different, the McNemar’s statistic is greater than \( z_{0.05} = 3.841459 \) (Dietterich, 1998). Tables 5 and 6 show the McNemar’s statistical test results of AdaBoostSVM and Diverse AdaBoostSVM on the 13 benchmark data sets (“YES” or “NO” in the parentheses indicates whether these two algorithms perform significantly different on this data set).

In Table 5, the McNemar’s statistical test results illustrate that the performance of AdaBoostSVM significantly differs from that of AdaBoostDT on 10 data sets out of total 13 data sets. This is because on these 10 data sets,
each McNemar’s statistic (Eq. (9)) is larger than 3.841459. Furthermore, since the generalization errors of AdaBoostSVM are less than those of AdaBoostDT on these 10 data sets (see Table 6), AdaBoostSVM performs better than AdaBoostDT on these 10 data sets. Hence, we can conclude that proposed AdaBoostSVM algorithm performs better than ABDT in general on these data sets. A similar conclusion can also be drawn from Table 5 that AdaBoostSVM outperforms ABNN, and is comparable to the standard SVM. It can be observed that ABsvm−s performs worse than the standard SVM. We think that this is because ABsvm−s forces the strong SVM classifiers (SVM with its best parameters) to focus on very hard training samples or outliers more emphatically. This case is also observed in Wickramaratna et al. (2001). Generally speaking, for balanced classification problems, the contribution of the proposed AdaBoostSVM lies in the corroborative proof and realization of Valentini and Dietterich’s (2004) idea, that states that AdaBoost with heterogeneous SVMs could work well. Furthermore, proposed AdaBoostSVM outperforms other Boosting algorithms on these balanced data sets, which has been justified statistically verified by our extensive experiments based on McNemar’s statistical test.

From Table 6, similar conclusions can also be drawn that our proposed Diverse AdaBoostSVM outperforms both ABDT and ABNN in general on these benchmark data sets. Moreover, since the Diverse AdaBoostSVM outperforms the standard SVM on 3 data sets while comparable on the other 10 data sets, we say that proposed Diverse AdaBoostSVM performs a little better than standard SVM in general.

5.1.3. Influence of C and σini

In order to show the influence of parameter C on AdaBoostSVM, we use the results on the UCI “Titanic” data set for illustration. We vary the value of C from 1 to 100, and perform experiments on 100 random partitions of this data set to obtain the average generalization performance. Fig. 3 shows the comparative results. Over a large range, the variation of C has little effect (less than 1%) on the final generalization performance. This is also consistent with the analysis of RBFSVM in Valentini and Dietterich (2004) that C value has less effect on the performance of RBFSVM. Note that the σ value decreases from σini to σmin as the number of SVM component classifiers increases (see the label of horizontal axis in Fig. 3). The small platform at the left top corner of this figure means that the test error does not decrease until the σ reduces to a certain value. Then, the test error decreases quickly to the lowest value and stabilizes there. This shows that the σini value does not have much impact on the final performance of AdaBoostSVM.

5.1.4. Influence of σstep

In order to show the influence of the step size of parameter σstep on AdaBoostSVM, we also use the results on the UCI “Titanic” data set for illustration. A set of experiments with different σstep values on the “Titanic” data set were performed. Fig. 4 gives the results. From this figure, we can find that, although the number of learning cycles in AdaBoostSVM changes with the value of σstep, the final test error is relatively stable. Similar conclusions can also be drawn on other benchmark data sets.

5.2. Comparison on imbalanced data sets

Although SVM has achieved great success in many area, such as handwriting recognition (Vapnik, 1998), text classification (Joachims, 1998) and image retrieval (Tong and Koller, 2001), when handling imbalanced classification problems, its performance drops significantly. In this section, we will show the performance of our proposed AdaBoostSVM on imbalanced classification problems and...
compare it with several state-of-the-art algorithms specifically designed to solve these problems.

5.2.1. Review of current algorithms dealing with imbalanced problems

In the case of binary classification, imbalanced classification means that the number of negative instances is much larger than that of positive ones, or vice versa, such as imbalanced document categorization (del Castillo and Serrano, 2004), imbalanced clustering for microarray data (Pearson et al., 2003), detecting credit card fraud (Fawcett and Provost, 1997) and many data mining tasks (Aaai’2000 Workshop on Learning from Imbalanced Data Sets, 2000; Icmi’2003 Workshop on Learning from Imbalanced Data Sets (ii), 2003; Editorial: Special issue on learning from imbalanced data sets, 2004. SIGKDD Explorations).

A common method to handle imbalanced problems is to rebalance them artificially by under-sampling (Kubat and Matwin, 1997) (ignoring instances from the majority class) or over-sampling (Chawla et al., 2002) (replicating instances from the minority class) or combination of both under-sampling and over-sampling (Ling and Li, 1998). The popular approach is the SMOTE algorithm (Chawla et al., 2002), which synthetically over-samples the minority class.

Another type of algorithms focuses on biasing the SVM to deal with the imbalanced problems. Several different ways are used. In Veropoulos et al. (1999), different penalty constants are used for different classes to control the balance between false positive instances and false negative instances. Cristianini et al. (2001) use kernel alignment to adjust the kernel matrix to fit the training samples. Wu and Chang (2005) realize this by kernel boundary alignment. Furthermore, in Akbani et al. (2004), SMOTE and different error costs are combined for SVM to better handle imbalanced problems. Decision tree (Drummond and Holte, 2000) and multiplayer perceptron (Nugroho et al., 2002) are also modified to improve the learning performance on imbalanced data sets. In Guo and Viktor (2004), Boosting combined with data generation is used to solve imbalanced problems. Another similar work (Yan et al., 2003) uses SVM ensemble to predict rare classes in scene classification.

5.2.2. Generalization performance on imbalanced data sets

Firstly, we compare our AdaBoostSVM with the standard SVM on the UCI “Splice” data set. The “Splice” data set has 483 positive training samples, 517 negative training samples and 2175 test samples. In the following experiments, the number of negative samples is fixed at 500 and the number of positive samples is reduced from 150 to 30 step-wise to realize different imbalance ratios. From Fig. 5, it can be found that along with the decreasing ratio of positive samples to that of negative ones, the improvement of AdaBoostSVM over SVM increases monotonically. When the ratio reaches 30:500, SVM almost cannot work and performs like random guess. On the other hand, AdaBoostSVM can still work, and the improvement reach about 15%.

In the following, AdaBoostSVM is compared with four algorithms, which are standard SVM (Vapnik, 1998), SVM with different penalty constants (SVM-DPC) (Veropoulos et al., 1999), Under-sampling (US) (Kubat and Matwin, 1997) and SMOTE (Chawla et al., 2002). Five UCI imbalanced data sets are used, namely Car(3), Glass(7), Letter(26), Segment(1) and Soybean(12). The class labels in the parentheses indicate the classes selected. For each data set, we randomly split it into training and test sets in the ratio of 70%;30%, and in these two sets, the ratios between the numbers of positive and negative instances are roughly same (Kubat and Matwin, 1997). In Table 7, the general characteristics of these data sets are given including the number of attributes, the number of positive instances and the number of negative instances. It also lists the amount of over-sampling of the minority class for SMOTE as suggested in Wu and Chang (2005). For SVM-DPC, the two penalty cost values are decided according to Akbani et al. (2004), where their ratio is set as the inverse of the ratio of the instance numbers in the two classes. For the under-sampling algorithm, the majority class is under-sampled by randomly removing samples from the majority class until it has the same number of instances as minority class.

The commonly used sensitivity and specificity are taken to measure the performance of each algorithm on the imbalanced data sets. They are defined as

\[
\text{Sensitivity} = \frac{\text{#true_positive}}{\text{#true_positive} + \text{#false_negative}},
\]

(10)

\[
\text{Specificity} = \frac{\text{#true_negative}}{\text{#true_negative} + \text{#false_positive}}.
\]

(11)

Several researchers (Kubat and Matwin, 1997; Wu and Chang, 2005; Akbani et al., 2004) have used the \(g\)-means
metric to evaluate the algorithm performance on imbalanced problems because $g$-means metric combines both the sensitivity and specificity by taking their geometric mean. Based on sensitivity and specificity, the $g$-means metric in Kubat and Matwin (1997) is calculated to evaluate these five algorithms on the imbalanced data sets. It is defined as follows:

$$g = \sqrt{\text{sensitivity} \times \text{specificity}}.$$  

(12)

The $g$-means metric value of the five algorithms on the UCI imbalanced data sets are shown in Table 8. From this table, it can be found that proposed AdaBoostSVM performs best among the five algorithms in general. It achieves the highest $g$-means metric value in four out of the five data sets and also obtain the highest average $g$-means metric value among them. The success of the proposed algorithm lies in its Boosting mechanism forcing part of RBFSVM component classifiers to focus on the misclassified instances in the minority class, which can prevent the minority class from being wrongly recognized as a noise of the majority class and classified into it. Hence, the AdaBoostSVM achieves better generalization performance on the imbalanced data sets. Note that the $g$-means metric value of SVM on “Car3” data set is 0. This is because SVM predict all the instances into the majority class.

Furthermore, the Receiver Operating Characteristic (ROC) analysis has been done. We use the Area Under the ROC curve (AUC) (Bradley, 1997) to compare these five algorithms on these imbalanced data sets. The AUC is defined as the area under an ROC curve. It is known that larger AUC values indicate generally better classifier performance (Hand, 1997). The AUC values listed in

### Table 7
General characteristics of UCI imbalanced data sets and the amount of over-sampling of the minority class for the SMOTE algorithm

<table>
<thead>
<tr>
<th>Data set</th>
<th># Attribute</th>
<th># Minority class</th>
<th># Majority class</th>
<th>Over-sampled (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment1</td>
<td>19</td>
<td>330</td>
<td>1980</td>
<td>200</td>
</tr>
<tr>
<td>Glass5</td>
<td>10</td>
<td>29</td>
<td>185</td>
<td>200</td>
</tr>
<tr>
<td>Soybean12</td>
<td>35</td>
<td>44</td>
<td>639</td>
<td>400</td>
</tr>
<tr>
<td>Car3</td>
<td>6</td>
<td>69</td>
<td>1659</td>
<td>400</td>
</tr>
<tr>
<td>Letter26</td>
<td>17</td>
<td>734</td>
<td>19266</td>
<td>400</td>
</tr>
</tbody>
</table>

### Table 8
$g$-means metric results on the five UCI imbalanced data sets

<table>
<thead>
<tr>
<th>Data set</th>
<th>SVM</th>
<th>SVM-DPC</th>
<th>US</th>
<th>SMOTE</th>
<th>AdaBoostSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment1</td>
<td>0.995</td>
<td>0.956</td>
<td>0.945</td>
<td>0.958</td>
<td>0.965</td>
</tr>
<tr>
<td>Glass5</td>
<td>0.867</td>
<td>0.874</td>
<td>0.835</td>
<td>0.863</td>
<td>0.885</td>
</tr>
<tr>
<td>Soybean12</td>
<td>0.926</td>
<td>0.993</td>
<td>0.921</td>
<td>0.975</td>
<td>0.997</td>
</tr>
<tr>
<td>Car3</td>
<td>0</td>
<td>0.382</td>
<td>0.925</td>
<td>0.966</td>
<td>0.975</td>
</tr>
<tr>
<td>Letter26</td>
<td>0.818</td>
<td>0.934</td>
<td>0.913</td>
<td>0.927</td>
<td>0.945</td>
</tr>
<tr>
<td>Average</td>
<td>0.721</td>
<td>0.827</td>
<td>0.908</td>
<td>0.938</td>
<td>0.953</td>
</tr>
</tbody>
</table>

### Table 9
AUC results on the five UCI imbalanced data sets

<table>
<thead>
<tr>
<th>Data set</th>
<th>SVM</th>
<th>SVM-DPC</th>
<th>US</th>
<th>SMOTE</th>
<th>AdaBoostSVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment1</td>
<td>0.998</td>
<td>0.982</td>
<td>0.978</td>
<td>0.985</td>
<td>0.991</td>
</tr>
<tr>
<td>Glass5</td>
<td>0.943</td>
<td>0.956</td>
<td>0.938</td>
<td>0.941</td>
<td>0.963</td>
</tr>
<tr>
<td>Soybean12</td>
<td>0.960</td>
<td>0.998</td>
<td>0.955</td>
<td>0.989</td>
<td>1.000</td>
</tr>
<tr>
<td>Car3</td>
<td>0.631</td>
<td>0.762</td>
<td>0.962</td>
<td>0.984</td>
<td>0.997</td>
</tr>
<tr>
<td>Letter26</td>
<td>0.933</td>
<td>0.978</td>
<td>0.930</td>
<td>0.964</td>
<td>0.985</td>
</tr>
<tr>
<td>Average</td>
<td>0.893</td>
<td>0.935</td>
<td>0.957</td>
<td>0.972</td>
<td>0.987</td>
</tr>
</tbody>
</table>

6. Conclusions

AdaBoost with properly designed SVM-based component classifiers is proposed in this paper, which is achieved by adaptively adjusting the kernel parameter to get a set of effective RBFSVM component classifiers. Experimental results on benchmark data sets demonstrate that proposed AdaBoostSVM performs better than other approaches of using component classifiers such as Decision Trees and Neural Networks. Besides these, it is found that AdaBoostSVM demonstrates good performance on imbalanced classification problems. An improved version is further developed to deal with the accuracy/diversity dilemma in Boosting algorithms, giving rising to better generalization performance.

### References


