Integrating Adaptive Boosting and Support Vector Machines with Varying Kernels

Kuo-Wei Hsu
Department of Computer Science, National Chengchi University
No. 64, Sec. 2, Zhi Nan Rd., Wen Shan District
Taipei City 11605, Taiwan (R.O.C.)
+886-2-29393091 ext. 62297
kwhsu@nccu.edu.tw

ABSTRACT
Adaptive Boosting, or AdaBoost , is a meta-learning algorithm that employs a classification algorithm as a base learner to train classification models and uses these models to perform collective classification. One of its main features is that iteratively it forces the base learner to work more on difficult samples. Usually it can achieve better overall classification performance, when compared to a single classification model trained by the classification algorithm used as the base learner. SVM, short for Support Vector Machine, is a learning algorithm that employs a kernel to project the original data space to a data space where a hyperplane that can linearly separate as many samples of classes as possible can be found. Because both are top algorithms, researchers have been exploring the use of AdaBoost with SVM. Unlike others simply using SVM with a single kernel as the base learner in AdaBoost, we propose an approach that uses SVM with multiple kernels as the base learners in a variant of AdaBoost. Its main feature is that it not only considers difficulties of samples but also classification performance of kernels, and accordingly it selects as well as switches between kernels in the boosting process. The experiment results show that we can obtain better classification performance by using the proposed approach.

CCS Concepts
• **Information systems→Data mining** • **Computing methodologies→Machine learning** • *Applied computing →Physical sciences and engineering*

Keywords
AdaBoost; Classification; Multi-class; SVM

1. INTRODUCTION
Adaptive Boosting, commonly abbreviated to AdaBoost, is more a framework than an algorithm for classification problems because it can be used with many other classification algorithms [16]. AdaBoost forms a group of classification models generated by a classification algorithm, which is used as a base learner, and AdaBoost collects from them their classifications for a sample that needs to be classified and then combines these classifications into one by weighted voting. The weight of a classification model depends on its classification performance, and the better performance corresponds to the higher weight. Furthermore, AdaBoost maintains the distribution of weights of samples. Iteratively, AdaBoost updates the distribution by adjusting the weight of each sample according to its degree of being misclassified, and the more misclassified samples (which are made by the group of classification models formed so far) correspond to the higher weight. AdaBoost emphasizes more on hard-to-classify samples.

AdaBoost has been applied in various applications. For example, character recognition [40], text categorization [15, 41], gender recognition [49], bankruptcy prediction [1], and financial distress prediction [43].

Support Vector Machine, commonly abbreviated to SVM, is a learning algorithm that constructs a hyperplane in a high-dimensional data space to separate samples of classes, and its objective is to achieve an optimal separation by having the hyperplane with the largest distance to the nearest samples of any class [5, 9]. Such a distance is called margin, and SVM achieves low generalization error or high accuracy by having a large margin on a set of samples used in training. Usually it is difficult to draw a line or a hyperplane to separate samples in the original data space of a given data set, because usually samples collected from real-world applications are not linearly separable. Using an appropriate kernel (which in fact is a function), SVM maps or projects the samples from the original data space to a higher-dimensional data space in which possibly it is less difficult to linearly separate samples [37, 39]; even so, it is not always the case that all samples are linearly separable. Nevertheless, selecting an appropriate kernel is the first and most important step in applying SVM to real-world data sets. After the projection, SVM is looking for a line or a hyperplane that can separate as many samples as possible. This means that some misclassified samples or errors are allowed. For a given data set, SVM constructs the hyperplane with the largest distance to the nearest samples of any class by learning the parameters of the hyperplane or, more specifically, solving an optimization problem to find the best combination of the parameters of the hyperplane. Sequential Minimal Optimization, commonly abbreviated to SMO, solves such an optimization problem by dividing it into lower-dimensional sub-problems and solving without greatly depending on a numerical optimization technique [19, 22, 36].

SVM is finding applications in various domains. For example, spam categorization [12], text classification [21], facial expression recognition [30], Web taxonomy integration [50], intrusion detection system [23], software defect prediction [13], network traffic classification [14], clinical entity recognition [45],

IMCOM'17, January 05–07, 2017, Beppu, Japan.

DOI: http://dx.doi.org/10.1145/3022227.3022314

computer worm activity detection [33], video event detection [17], fault diagnosis [31], and customer retention [29].

AdaBoost and SVM are top algorithms [48], so researchers have been exploring the use of AdaBoost with SVM in the hope that, on the basis of high classification performance often achieved by SVM, AdaBoost can boost SVM and achieve higher classification performance. As others, we surely want to correct as many errors as possible and at the same time make no errors on samples that are correctly classified in the first place, and therefore we focus on errors that are associated with hard-to-classify samples. AdaBoost pays more attention to hard-to-classify samples, and accordingly we propose an approach to integrate AdaBoost and SVM. Like others, we use SVM as the base learner to build classification models. However, unlike others, we propose not to simply use SVM as a general building block in AdaBoost but to integrate SVM into the boosting process.

We propose an approach that uses SVM with different kernels in different iterations of AdaBoost. The intuition behind our design is as follows: Once we perform weighted sampling on samples according to their degrees of being hard-to-classify, as we do in each iteration in the boosting process, we change the distribution of samples input into SVM to train a classification model, or in other words, we change the original data space even before the used kernel projects it into another data space. In this situation, the used kernel may not work as well as it did at the beginning in the boosting process, and the kernel used earlier may not be suitable for being used later.

The proposed approach varies the use of kernels in SVM according to weights of kernels. Similar to how we update weights of samples in AdaBoost, we update the weight of each kernel by referring to its classification performance, and the better performance corresponds to the higher weight, which further corresponds to the higher probability that the kernel will be used again in the next iteration. This feature distinguishes the proposed approach from those proposed by others. Inspired by the exponential function used to update weights of samples in AdaBoost, we use an exponential function to update weights of kernels. These two exponential functions are related to classification performance achieved by the classification model used in an iteration.

The reminder of this paper is organized as follows: Section 2 briefly reviews the work related to the integration of boosting and SVM. Section 3 presents the approach proposed to integrate AdaBoost and SVM with varying kernels. Section 4 describes the experiment settings and reports the experiment results. Section 5 concludes this paper and discusses the possible directions for future work.

2. RELATED WORK

Some researchers use SVM as the base learner in AdaBoost (or in a variant of AdaBoost), and most of their designs are made for particular applications. Examples are as follows: For pedestrian detection, Nishida and Kurita propose an algorithm that integrates feature selection in AdaBoost and uses SVM as the base learner [32]. Liu et al. propose to combine an integrated sampling technique with a group of SVM models in order to obtain better classification performance on imbalanced data sets, and they conclude from their experiments that a group of SVM models outperforms individual SVM models [28]. Wang and Japkowicz use SVM as the base learner in boosting to form a group of SVM models for classification on imbalanced data sets [47]. Cheng et al. first use an optimization-based feature selection method and then SVM-based AdaBoost to build classification models of ligands of a serotonin receptor subtype [7], and Cheng and Zhang use similar methods to build classification models of Estrogen Receptor-β ligands [8]. In [34], the authors use this combination in facial expression recognition. More examples of applications of this combination of AdaBoost and SVM include sentiment analysis [42] and facial expression recognition [34].

Dong and Han empirically compare several types of combinations of SVM models for text classification problems and conclude that boosting does not provide performance improvement for SVM, and they explain it by that SVM is strong learner while boosting is often used for weak (or less accurate) base learners [11]. We argue that their findings are only for their experiment settings, including the data that they use. First of all, there is no theoretical evidence indicating that boosting cannot be used with a strong base learner but can be used only with weak base learners. Second of all, there is empirical evidence showing that boosting can improve the classification performance of SVM.

Some researchers use AdaBoost as a special pre-processing method to manipulate the data set that is then input to SVM for model training. Examples are as follows: Pavlov and Mao use boosting to determine the weights of samples and accordingly create subsets of the original data set that are smaller and will be used to train SVM models, and they use AdaBoost and SMO [35]. Littlewort et al. study AdaBoost, SVM, and an approach using both of them for expression recognizer in order to assist in human-robot interaction, and in their paper Adaboost is used to select features and SVM is used to train models with data sets of reduced features [27]. In [2], Bartlett et al. compare various approaches designed to recognize facial actions, and they conclude from their experiments that "*best results were obtained by selecting a subset of Gabor filters using AdaBoost and then training Support Vector Machines on the outputs of the filters selected by AdaBoost.*" Bartlett et al. first use AdaBoost to select features and then use SVM to build classification models for facial expression recognition [3]. Working on eye detection, Tang et al. propose an approach that first uses AdaBoost to train a model for face location and a model for eye detection and then uses SVM to train a model for precise eye position [44].

Some researchers propose approaches that manipulate the settings of SVM for the case where AdaBoost is used as a pre-processing method for SVM or for the case where SVM is used as the base learner in AdaBoost. The manipulation includes kernel selection, kernel construction, and parameter tuning. Examples are as follows: To address the issue of kernel selection for SVM, Bennett et al. focus on the regression problem and propose an algorithm named Multiple Additive Regression Kernels in which a boosting-type procedure is used to select among kernels formed by different kernel functions and parameters [4]. Crammer et al. use a boosting-type procedure to learn a proper combination of weights of base kernels that are less accurate and then construct a kernel that is more accurate, and they use the classification error to adjust the weight of a base kernel (but not weights of samples) [10]. Li et al. propose an algorithm that uses SVM with RBF kernel as the base learner in AdaBoost and adaptively adjusts a kernel parameter; in [24], they conclude from their experiments that their algorithm is advantageous in "*easier model selection and better generalization performance*", and in [25], they conclude that their algorithm "*demonstrates better generalization performance than SVM on imbalanced classification problems.*"

Considering the situation that only a small number of samples is available for training a classification model, Hertz et al. indicate

that classification performance of SVM can be improved by using a kernel learned or computed from the training data, and they present a boosting algorithm named KernelBoost that uses boosting to combine "weak space partitions" and further to compute a kernel that will be used in SVM [20].

To detect Alzheimer's disease on brain Magnetic Resonance Imaging (MRI), Savio et al. discuss an algorithm named "*Diverse AdaBoost SVM*" that uses SVM as the base learner in AdaBoost and adjusts a kernel parameter according to the error rate and the diversity value in each iteration of AdaBoost [38].

3. APPROACH

Because AdaBoost and SVM are well-developed and well-known, we directly introduce the proposed approach in this section.

In Figure 1, Algorithm 1 describes the training part for integrating AdaBoost and SVM with varying kernels. We use upper-case letters to denote a group of items or elements, and we use lower-case letters to denote an item, an element, or a value. To describe the proposed approach in a concise yet complete manner, we use APIs (Application Programming Interfaces) to represent functional modules, in each of which computing operations are programmed for a specific purpose. These APIs are not directly related to the logic of the proposed approach. They are listed below in alphabetical order:

- *BuildSVM*: It is to build an operational SVM model for classification using the data set specified by the first input parameter as the training data set with the kernel specified by the second input parameter.
- *Exp*: The exponential function.
- *GetClasses*: It is to determine the number of class or category labels of the data set specified by the input parameter.
- *GetKernel*: It is to randomly select an index number of a kernel according to weights of kernels specified by the input parameter.
- *GetLabel*: It is to take a sample specified by the input parameter and return its (actual) class label.
- *GetMaxIndex*: It is to take a list of numbers specified by the input parameter and then determine the index (i.e. the position in the list) of the number with the largest value.
- *GetSamples*: It is to randomly draw samples with replacement from the data set specified by the first input parameter according to weights of samples specified by the second input parameter.
- *Ind*: The indicator function, which returns 1 if the condition specified by the input parameter turns out to be true and 0 otherwise.
- *Log*: The Logarithmic function.

Algorithm 1 takes two input parameters, namely the given data set for training and the number of iterations. It returns the resulting AdaBoost classification model, which is a group of SVM models for classification. We initialize variables at the beginning of the algorithm and iteratively update the values for these variables. In each iteration, it first randomly selects an index number of a kernel according to weights of kernels, and also it randomly draws samples with replacement from the data set according to weights of samples. These samples are then used with the selected kernel to train or build a classification model based on SVM. After the model is trained or built, the algorithm evaluates its classification performance and accordingly updates weights of samples and weights of kernels.

The goal of the first inner loop in Algorithm 1 is to calculate the weighted error rate of a model in an iteration of the outer loop. The goal of the second inner loop is to use the exponential function used in AdaBoost to update weights of samples. AdaBoost is originally designed for binary classification problems. There exist methods that allow AdaBoost to work on multi-class classification problems. Algorithm 1 designed by referring to an intrinsically multi-class AdaBoost algorithm named SAMME, short for Stagewise Additive Modeling using a Multi-class Exponential loss function, which is different from the original AdaBoost in that it considers the number of classes when computing the weight of a classification model [51]. This difference is reflected in the term $Log(GetClasses(D)\text{-}1)$ in Algorithm 1. Inspired by the exponential function used to update weights of samples in AdaBoost, we use an exponential function to update weights of kernels, as we can see in the third inner loop in Algorithm 1. Additionally, when updating weights of kernels, we consider the number of kernels in order to avoid that the algorithm is trapped in a kernel which is not actually the best. If there exists a kernel that is not actually suitable for the given data set but happens to perform well on some training samples in earlier iterations, and if the algorithm increases its weight by a large step, then the algorithm will need more iterations to realize its actual performance and decrease its weights such that the algorithm will have a chance to select other kernels. Because we would not like to run many iterations, we make smaller the size of a step in updating weights of kernels. The above two exponential functions are related to classification performance achieved by the classification model used in an iteration, so we say that the proposed approach literally integrates AdaBoost and SVM.

SVM is originally designed for binary classification problems. Most implementations of SVM deal with a multi-class classification problem by transforming it into multiple binary classification problems. In our current implementation of Algorithm 1, we use SMO.

The effectiveness of SVM is dependent on the kernel used, and the Gaussian kernel is commonly used. A byproduct of the propose approach is an automatic kernel selection approach. Considering that we run AdaBoost for t iterations for a kernel and there are m kernels from which we want to select as the best one for a given data set, we will build $t \times m$ SVM models and then evaluate classification performance of each kernel, while we will build only t SVM models and then know which kernel is the best if we use the proposed approach.

The proposed approach is comparable to or even better than the approach simply using SVM with a kernel in AdaBoost without integrating them. As for the using or testing part, we can simply use or test the trained AdaBoost classification model as a general classification model. We give the model a new sample without a pre-specified class or category label, and then the model will give us an assigned or classified label. In Figure 2, Algorithm 2 describes the testing part for integrating AdaBoost and SVM with varying kernels. We use the same notations as those used in Algorithm 1. Given a sample, Algorithm 2 calculates the likelihood that the sample belongs to a class, for all classes the algorithm finds the one with the largest likelihood, and finally the algorithm assigns the label of the found class to the sample.

Input: D is the given data set for training, and t is the number of iterations

Output: C is the resulting AdaBoost classification model, i.e. a group of SVM models for classification

1 Initialize weights of samples, $WS \leftarrow \{ws_i \mid ws_i = 1/n,\ 1 \le i \le n\}$, where ws_i is the weight of d_i, the i-th sample in D, and n is the number of samples
2 Initialize weights of kernels, $WK \leftarrow \{wk_i \mid wk_i = 1/m,\ 1 \le i \le m\}$, where wk_i is the weight of the i-th kernel and m is the number of kernels
3 Initialize $A \leftarrow \{a_i \mid a_i = 0,\ 1 \le i \le t\}$, where a_i is related to classification performance of c_i, the i-th model in C
4 For $i = 1$ to t
5 $k_i \leftarrow GetKernel(WS)$, where k is the index number of the kernel selected and used in the i-th iteration
6 $T \leftarrow GetSamples(D, WS)$, where T is the sampled data set for training in an iteration
7 $c_i \leftarrow BuildSVM(T,k_i)$, where c_i is the i-th model in C
8 $e \leftarrow 0$, where e is the weighted error rate of a model in an iteration
9 For $j = 1$ to n
10 $e \leftarrow e + Ind(c_i(d_j) \neq GetLabel(d_j)) \times ws_j$, where c_i takes a sample as an input parameter and returns a class label as the classification result
11 End For
12 $a_i \leftarrow 1/GetClasses(D) \times Log((1\text{-}e)/e) + Log(GetClasses(D)\text{-}1)$
13 For $j = 1$ to n
14 $ws_j \leftarrow ws_j \times Exp((2 \times Ind(c_i(d_j)) = GetLabel(d_j))\text{-}1) \times a_i)$
15 End For
16 For $j = 1$ to m
17 $wk_j \leftarrow wk_j \times Exp(Ind(e \le 1/GetClasses(D))) \times a_i/m)$
18 End For
19 End For
20 Return C

Figure 1. Algorithm 1: The training part for integrating AdaBoost and SVM with varying kernels

Input: s is the sample that is going to be tested (or labeled)

Output: l is the class label (or the classification)

1 Initialize $P \leftarrow \{p_i \mid p_i = 0,\ 1 \le i \le GetClasses(D)\}$, where p_i is related to the accumulated classification performance that models provide with respect to the i-th class label; D is the one input to Algorithm 1
2 For i =1 to t (the number of iterations in Algorithm 1)
3 $p_{ci(s)} \leftarrow p_{ci(s)} + a_i$
4 End For
5 $l \leftarrow GetMaxIndex(P)$
6 Return l

Figure 2. Algorithm 2: The testing part for integrating AdaBoost and SVM with varying kernels

4. EXPERIMENTS

All the data sets considered in experiments are public data sets, and most are derived from the data sets available on the UC Irvine Machine Learning Repository [26]. Table 1 summarizes the data sets used in experiments and their characteristics. In the table, the first and second columns are the number and name of the data set, respectively; the third, fourth, and fifth columns are the numbers of samples, attributes, and classes, respectively; the sixth column is the number of attributes with missing values, and the seventh column is the range of percentages of missing values on these attributes; the eighth column, the last column, is the percentage of samples that belong to the majority class (or classes). As we can see from the table that these data sets have different levels of quality or difficulty for classification.

We use AdaBoost and SMO provided by WEKA [18], and we use the functions built in WEKA to implement the proposed approach.

Table 2 summarizes the results in accuracy obtained by using 10-fold cross-validation. In the table, SVM means a single SVM model, AB+SVM means the 10-iteration AdaBoost with SVM (or, simply using SVM as the base learner in AdaBoost), ABSVM means the proposed approach with 4 kernels and 10 iterations; Poly means the polynomial kernel, N.Poly means the normalized polynomial kernel, Puk means the Pearson VII function based universal kernel [46], and RBF means the Radial Basis Function (Gaussian) kernel [6]. In the table, the last column is considering the number of kernels in updating weights of kernels, and the second last column is not doing so; the former implies a smaller

step in updating weights and is associated with Line 17 in Algorithm 1, while the latter implies a larger step and is associated with Line 17 with a minor change.

AB+SVM performs better, or at least equally well as, SVM in 9 data sets. AdaBoost can possibly increase the classification performance of SVM, but it decreases the classification performance in the following cases: 1) in the data set *analcatdata_authorship*, when the normalized polynomial kernel is used; 2) in the data set *breast-w*, when the polynomial kernel is used; and, 3) in the data sets *ionosphere* and *vowel*, when the RBF kernel is used. Moreover, in the data set *vehicle*, it increases the classification performance only when the normalized polynomial kernel is used; and, in the data set *vote*, it shows the same classification performance when the RBF kernel is used, while it decreases the classification performance when the other 3 kernels are used. These two data sets, *vehicle* and *vote*, possibly contain noisy samples such that noise is overemphasized (and, in some sense, amplified) in the boosting process and then misleads SVM in the search for the optimal hyperplane. In these two data sets, the proposed approach shows the highest accuracy. The results prove the concept of selecting and switching between kernels in the boosting process. For the proposed approach, when it is with a smaller step in updating weights of kernels, it shows the highest accuracy in 10 data sets; when it is with a larger step, it shows the highest accuracy in 6 data sets. Compared to using a larger step, using a smaller step shows higher accuracy in 13 data sets, and it shows lower accuracy in only 1 data set. Therefore, it is better to consider the number of kernels in updating weights of kernels.

Table 1. Data sets

No	Data set	Samples	Attributes	Classes	Attr. w/ missing	Missing	Majority
1	analcatdata_authorship	841	70	4	1		37.7%
2	anneal	898	38	6	0		76.2%
3	autos	205	25	7	7	1%-20%	32.7%
4	balance-scale	625	4	3	0		46.1%
5	breast-w	699	9	2	1	2%	65.5%
6	car	1728	6	4	0		70%
7	diabetes	768	8	2	0		65.1%
8	ecoli	336	7	8	0		42.6%
9	ionosphere	351	34	2	0		64.1%
10	irish	500	5	3	2	1%-5%	65%
11	liver-disorders	345	6	2	0		58%
12	sonar	208	60	2	0		53.4%
13	vehicle	846	18	4	0		25.8%
14	vote	435	16	2	16	2%-24%	61.4%
15	vowel	990	13	11	0		9.1%

Table 2. Results in accuracy

	SVM				AB+SVM				ABSVM	
No	Poly	N.Poly	Puk	RBF	Poly	N.Poly	Puk	RBF	w/o kernel num.	w/ kernel num.
1	99.4	99.8	99.2	98.7	99.4	99.2	99.2	98.9	99.3	99.5
2	96.5	94.9	96.3	89.9	97.6	94.9	97.2	92	98.1	99.1
3	68.3	61.5	61	43.9	69.8	65.4	63.9	44.4	68.3	68.8
4	87	90.9	89.6	87.2	87.5	90.9	90.1	89	90.1	90.4
5	96.4	93.7	96.4	96	96.3	93.8	96.7	96	96.3	96.6
6	93.3	96.3	91.4	83.9	93.5	96.4	91.6	84.6	87.9	99.6
7	76.2	65.9	75.7	65.1	77.2	65.9	77.2	65.1	73.8	77.9
8	82.1	70.2	86.6	42.6	83.3	70.8	87.5	42.6	86	86.3
9	87.2	83.9	94.3	75.8	87.7	85.2	94.6	74.9	95.4	96
10	98.6	98.6	98.4	90.8	98.6	98.6	98.4	91.3	98.4	98.6
11	57.4	60.3	67.8	58	58.3	60.6	69.3	58	66.4	69.9
12	75	74	83.7	66.3	75.5	76.4	85.6	73.1	88	87.5
13	73.9	73.8	74.8	40	72.9	74.6	74.6	39.7	76.4	76.4
14	95.2	95.4	94.9	94.7	94.5	94.9	94.5	94.7	95.9	96.1
15	69.7	50.9	94.7	29	77.4	65.9	98.1	28.2	98.1	98.2

Although the proposed approach shows the highest accuracy in most of the data sets used in experiments, we are by no means going to conclude that it is the best algorithm combining AdaBoost and SVM. We need to run more experiments (and this would be part of future work). Nevertheless, the results demonstrate the proof of our concept for the integration of AdaBoost and SVM.

We consider AB+SVM with the best kernel by referring to Table 2 and ABSVM when the number of iterations is increased from 10 to 100 with a step of 10. Figures 3, 4, and 5 present the results for the data sets *liver-disorders*, *sonar*, and *vehicle*, respectively. In these figures, the x-axis is the number of iterations and the y-axis is accuracy.

In these three figures, we can observe periodic changes of accuracy (or accuracy periodically being going up and down). This is a characteristic of AdaBoost (or, more precisely, the boosting process). These changes come from that AdaBoost attempts to fix errors made in earlier iterations. As we can see from these figures, the accuracy achieved by running the proposed approach for 100 iterations is higher than that achieved by running it for 10 iterations, but this is not the case for AB+SVM, which possibly needs more iterations to fix errors. Therefore, these figures show us that the proposed approach can more efficiently fix errors. Furthermore, the data set associated with Figure 5 is a data set that possibly contains noisy samples. From the figure, we can see that noise has a negative effect on the classification performance of AB+SVM but not on the proposed approach. Again, The above findings are because the proposed approach selects and switches between kernels in the boosting

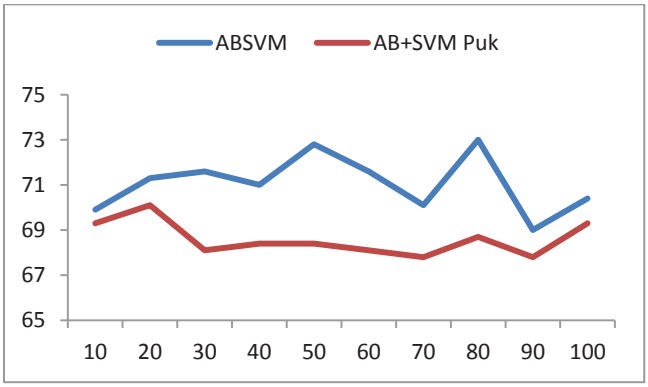

Figure 3. AB+SVM with Puk kernel and ABSVM for the data set *liver-disorders*

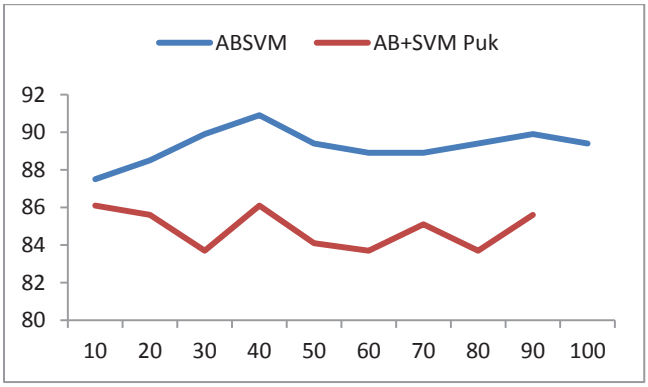

Figure 4. AB+SVM with Puk kernel and ABSVM for the data set *sonar*

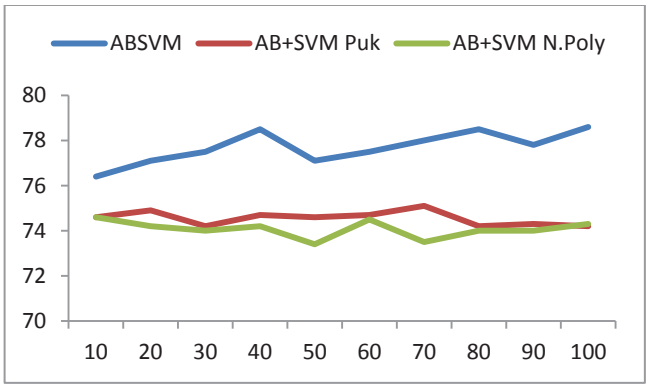

Figure 5. AB+SVM with normalized polynomial and Puk kernels and ABSVM for the data set *vehicle*

5. CONCLUSION

AdaBoost is a meta-learning algorithm that employs a classification algorithm as a base learner to form a group of classification models and uses voting to combine individual classifications made by these models for a sample into an overall classification. It emphasizes more on hard-to-classify samples. SVM is a generalized linear classifier, and it employs a function called kernel to project the original data space to a data space where it can find a hyperplane that can linearly separate as many samples as possible based on their classes. It can minimize the error and maximize the margin at the same time. Because AdaBoost can improve the classification performance of its base learner, it is often used with a weak (or less accurate) but simple classification algorithm in order to perform faster classification with acceptable accuracy. Because SVM is considered as a strong (or more accurate) classification algorithm, to obtain accuracy that is more than acceptable, researchers have been exploring the use of AdaBoost with SVM in the hope that AdaBoost can further improve the classification performance of SVM. For instance, researchers use SVM with a single kernel as the base learner in AdaBoost. However, because samples are sampled based on their weights and their weights are updated based on their degrees of being hard-to-classify in each iteration in the boosting process, the data space is changed such that possibly the kernel used for data space projection needs to be changed too.

We propose an approach that employs SVM with multiple kernels as the base learners in a variant of the boosting process of AdaBoost designed for multi-class classification. It uses SVM with different kernels in different iterations. It varies the employment of kernels based on their weights. From iterations to iterations, it not only updates weights of samples but also updates weights of kernels. In each iteration, it updates the two types of weights based on the classification performance of the model trained with the selected samples and the selected kernel. This is the distinguishing feature of the proposed approach. Furthermore, it is important to select an appropriate kernel when using SVM (and when using AdaBoost with SVM), and often kernel selection is done empirically. The proposed approach can do automatic kernel selection, and it can let the kernels that do not perform well also contribute to overall classifications. We implement the propose approach by utilizing a popular machine learning toolkit. We conduct experiments on public data sets. According to the results, we can obtain better classification performance by using the proposed approach.

As for the possible directions for future work, we plan to study the use of other types of kernels and also investigate a more intelligent way to select or switch between kernels.

6. ACKNOWLEDGMENTS

The author would like to thank anonymous reviewers for their valuable time.

7. REFERENCES

[1] E Alfaro, N García, M Gámez, and D Elizondo. 2008. Bankruptcy forecasting: An empirical comparison of AdaBoost and neural networks. *Decis. Support Syst.* 45, 1 (April 2008), 110-122. DOI=http://dx.doi.org/10.1016/j.dss.2007.12.002

[2] MS Bartlett, G Littlewort, C Lainscsek, I Fasel, and J Movellan. 2004. Machine learning methods for fully automatic recognition of facial expressions and facial actions. In *Proc. of International Conference on Systems, Man and Cybernetics* (SMC '04), 592-597. DOI=http://dx.doi.org/10.1109/ICSMC.2004.1398364

[3] MS Bartlett, G Littlewort, M Frank, C Lainscsek, I Fasel, and J Movellan. 2005. Recognizing facial expression: Machine learning and application to spontaneous behavior. In *Proc. of Conference on Computer Vision and Pattern Recognition* (CVPR '05), 568-573. DOI=http://dx.doi.org/10.1109/CVPR.2005.297

[4] KP Bennett, M Momma, and MJ Embrechts. 2002. MARK: A boosting algorithm for heterogeneous kernel models. In *Proc. of the 8th International Conference on Knowledge Discovery and Data mining* (KDD '02), 24-31. DOI=http://dx.doi.org/10.1145/775047.775051

[5] CJC Burges. 1998. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Discov.* 2, 2 (June 1998), 121-167. DOI=http://dx.doi.org/10.1023/A:1009715923555

[6] Y-W Chang, C-J Hsieh, K-W Chang, M Ringgaard, and C-J Lin. 2010. Training and testing low-degree polynomial data mappings via linear SVM. *J. Mach. Learn. Res.* 11 (August 2010), 1471-1490.

[7] Z Cheng, Y Zhang, C Zhou, W Zhang and S Gao. 2009. Classification of 5-HT1A receptor ligands on the basis of their binding affinities by using PSO-Adaboost-SVM. *Int. J. Mol. Sci.* 10, 8 (July 2009), 3316-3337. DOI=http://dx.doi.org/10.3390/ijms10083316

[8] Z Cheng and Y Zhang. 2010. Classification models of estrogen receptor-β ligands based on PSO-Adaboost-SVM. *Journal of Convergence Information Technology* 5, 2 (April 2010), 67-83.

[9] C Cortes and V Vapnik. 1995. Support-vector networks. *Mach. Learn.* 20, 3 (September 1995), 273-297. DOI=http://dx.doi.org/10.1007/BF00994018

[10] K Crammer, J Keshet, and Y Singer. 2002. Kernel design using boosting. In *Proc. of Conference on Advances in Neural Information Processing Systems* 15 (NIPS '02), 553-560.

[11] Y-S Dong and K-S Han. 2005. Boosting SVM classifiers by ensemble. In *Special Interest Tracks and Posters of the 14th International Conference on World Wide Web* (WWW '05), 1072-1073. DOI=http://dx.doi.org/10.1145/1062745.1062874

[12] H Drucker, D Wu, and V Vapnik. 1999. Support vector machines for spam categorization. *Trans. Neur. Netw.* 10, 5 (September 1999), 1048-1054. DOI=http://dx.doi.org/10.1109/72.788645

[13] KO Elish and MO Elish. 2008. Predicting defect-prone software modules using support vector machines. *J. Syst. Softw.* 81, 5 (May 2008), 649-660. DOI=http://dx.doi.org/10.1016/j.jss.2007.07.040

[14] A Este, F Gringoli, and L Salgarelli. 2009. Support vector machines for TCP traffic classification. *Comput. Netw.* 53, 14 (September 2009), 2476-2490. DOI=http://dx.doi.org/10.1016/j.comnet.2009.05.003

[15] A Esuli, T Fagni, and F Sebastiani. 2006. MP-Boost: a multiple-pivot boosting algorithm and its application to text categorization. In *Proc. of the 13th International Conference on String Processing and Information Retrieval* (SPIRE '06), 1-12. DOI=http://dx.doi.org/10.1007/11880561_1

[16] Y Freund and RE Schapire. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In *Proc. of the 2nd European Conference on Computational Learning Theory* (EuroCOLT '95), 23-37. DOI=http://dx.doi.org/10.1007/3-540-59119-2_166

[17] N Gkalelis and V Mezaris. 2014. Video event detection using generalized subclass discriminant analysis and linear support vector machines. In *Proc. of International Conference on Multimedia Retrieval* (ICMR '14), 25-32. DOI=http://dx.doi.org/10.1145/2578726.2578745

[18] M Hall, E Frank, G Holmes, B Pfahringer, P Reutemann, and IH Witten. 2009. The WEKA data mining software: an update. *SIGKDD Explor. Newsl.* 11, 1 (November 2009), 10-18. DOI=http://dx.doi.org/10.1145/1656274.1656278

[19] T Hastie and R Tibshirani. 1998. Classification by pairwise coupling. In *Proc. of Conference on Advances in Neural Information Processing Systems* 10 (NIPS '97), 507-513.

[20] T Hertz, AB Hillel, and D Weinshall. 2006. Learning a kernel function for classification with small training samples. In *Proc. of the 23rd International Conference on Machine learning* (ICML '06), 401-408. DOI=http://dx.doi.org/10.1145/1143844.1143895

[21] T Joachims. 2001. A statistical learning model of text classification for support vector machines. In *Proc. of the 24th International Conference on Research and Development in Information Retrieval* (SIGIR '01), 128-136. DOI=http://dx.doi.org/10.1145/383952.383974

[22] SS Keerthi, SK Shevade, C Bhattacharyya, and KRK Murthy. 2001. Improvements to Platt's SMO algorithm for SVM classifier design. *Neural Comput.* 13, 3 (March 2001), 637-649. DOI=http://dx.doi.org/10.1162/089976601300014493

[23] L Khan, M Awad, and B Thuraisingham. 2007. A new intrusion detection system using support vector machines and hierarchical clustering. *VLDB J.* 16, 4 (October 2007), 507-521. DOI=http://dx.doi.org/10.1007/s00778-006-0002-5

[24] X Li, L Wang, and E Sung. 2005. A study of AdaBoost with SVM based weak learners. In *Proc. of International Joint Conference on Neural Networks* (IJCNN '05), 196-201. DOI=http://dx.doi.org/10.1109/IJCNN.2005.1555829

[25] X Li, L Wang, and E Sung. 2008. AdaBoost with SVM-based component classifiers. *Eng. Appl. Artif. Intell.* 21, 5

(August 2008), 785-795. DOI=http://dx.doi.org/10.1016/j.engappai.2007.07.001

[26] M Lichman. 2013. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

[27] GC Littlewort, MS Bartlett, J Chenu, I Fasel, T Kanda, H Ishiguro, and J Movellan. 2003. Towards social robots: Automatic evaluation of human-robot interaction by face detection and expression classification. In *Proc. of Conference on Advances in Neural Information Processing Systems* 16 (NIPS '03), 1563-1570.

[28] Y Liu, A An, and X Huang. 2006. Boosting prediction accuracy on imbalanced datasets with SVM ensembles. In *Proc. of the 10th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining* (PAKDD '06), 107-118. DOI=http://dx.doi.org/10.1007/11731139_15

[29] S Maldonado, Á Flores, T Verbraken, B Baesens, and R Weber. 2015. Profit-based feature selection using support vector machines - General framework and an application for customer retention. *Appl. Soft Comput.* 35, C (October 2015), 740-748. DOI=http://dx.doi.org/10.1016/j.asoc.2015.05.058

[30] P Michel and R El Kaliouby. 2003. Real time facial expression recognition in video using support vector machines. In *Proc. of the 5th International Conference on Multimodal Interfaces* (ICMI '03), 258-264. DOI=http://dx.doi.org/10.1145/958432.958479

[31] M Namdari and H Jazayeri-Rad. 2014. Incipient fault diagnosis using support vector machines based on monitoring continuous decision functions. *Eng. Appl. Artif. Intell.* 28 (February 2014), 22-35. DOI=http://dx.doi.org/10.1016/j.engappai.2013.11.013

[32] K Nishida and T Kurita. 2005. Boosting soft-margin SVM with feature selection for pedestrian detection. In *Proc. of the 6th international conference on Multiple Classifier Systems* (MCS '05), 22-31. DOI=http://dx.doi.org/10.1007/11494683_3

[33] N Nissim, R Moskovitch, L Rokach, and Y Elovici. 2012. Detecting unknown computer worm activity via support vector machines and active learning. *Pattern Anal. Appl.* 15, 4 (November 2012), 459-475. DOI=http://dx.doi.org/10.1007/s10044-012-0296-4

[34] E Owusu, Y Zhan, and QR Mao. 2014. An SVM-AdaBoost facial expression recognition system. *Applied Intelligence* 40, 3 (April 2014), 536-545. DOI=http://dx.doi.org/10.1007/s10489-013-0478-9

[35] D Pavlov, J Mao, and B Dom. 2000. Scaling-up support vector machines using boosting algorithm. In *Proc. of the 15th International Conference on Pattern Recognition* (ICPR '00), 219-222. DOI=http://dx.doi.org/10.1109/ICPR.2000.906052

[36] JC Platt. 1999. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods*. MIT Press, 185-208.

[37] WH Press, SA Teukolsky, WT Vetterling, and BP Flannery. 2007. *Numerical Recipes: The Art of Scientific Computing* (3rd Ed.). Cambridge University Press.

[38] A Savio, M García-Sebastián, M Graña, and J Villanúa. 2009. Results of an Adaboost Approach on Alzheimer's Disease Detection on MRI. In Proc. *of the 3rd International Work-Conference on the Interplay Between Natural and Artificial Computation: Part II: Bioinspired Applications in Artificial and Natural Computation* (IWINAC '09), 114-123. DOI=http://dx.doi.org/10.1007/978-3-642-02267-8_13

[39] B Scholkopf and AJ Smola. 2001. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press.

[40] H Schwenk and Y Bengio. 1997. AdaBoosting Neural Networks: Application to on-line Character Recognition. In *Proc. of the 7th International Conference on Artificial Neural Networks* (ICANN '97), 967-972.

[41] F Sebastiani, A Sperduti, and N Valdambrini. 2000. An improved boosting algorithm and its application to text categorization. In *Proc. of the 9th International Conference on Information and Knowledge Management* (CIKM '00), 78-85. DOI=http://dx.doi.org/10.1145/354756.354804

[42] A Sharma and S Dey. 2013. A boosted SVM based sentiment analysis approach for online opinionated text. In *Proc. of Research in Adaptive and Convergent Systems* (RACS '13), 28-34. DOI=http://dx.doi.org/10.1145/2513228.2513311

[43] J Sun, M-Y Jia, and Hui Li. 2011. AdaBoost ensemble for financial distress prediction: An empirical comparison with data from Chinese listed companies. *Expert Syst. Appl.* 38, 8 (August 2011), 9305-9312. DOI=http://dx.doi.org/10.1016/j.eswa.2011.01.042

[44] X Tang, Z Ou, T Su, H Sun, and P Zhao. 2005. Robust precise eye location by adaboost and SVM techniques. In *Proc. of the 2nd International Conference on Advances in Neural Networks* (ISNN '05), 93-98.

[45] B Tang, H Cao, Y Wu, M Jiang, and H Xu. 2012. Clinical entity recognition using structural support vector machines with rich features. In *Proc. of the 6th International Workshop on Data and Text Mining in Biomedical Informatics* (DTMBIO '12), 13-20. DOI=http://dx.doi.org/10.1145/2390068.2390073

[46] B Üstün, WJ Melssen, and LMC Buydens. 2006. Facilitating the application of support vector regression by using a universal Pearson VII function based kernel. *Chemometr. Intell. Lab.* 81, 1 (March 2006), 29-40, 0169-7439, DOI=http://dx.doi.org/10.1016/j.chemolab.2005.09.003

[47] BX Wang and N Japkowicz. 2010. Boosting support vector machines for imbalanced data sets. *Knowl. Inf. Syst.* 25, 1 (October 2010), 1-20. DOI=http://dx.doi.org/10.1007/s10115-009-0198-y

[48] X Wu, V Kumar, JR Quinlan, J Ghosh, Q Yang, H Motoda, GJ McLachlan, A Ng, B Liu, PS Yu, Z-H Zhou, M Steinbach, DJ Hand, and D Steinberg. 2007. Top 10 algorithms in data mining. *Knowl. Inf. Syst.* 14, 1 (December 2007), 1-37. DOI=http://dx.doi.org/10.1007/s10115-007-0114-2

[49] X Xu and TS Huang. 2007. SODA-boosting and its application to gender recognition. In *Proc. of the 3rd International Conference on Analysis and Modeling of Faces and Gestures* (AMFG'07), 193-204.

[50] D Zhang and WS Lee. 2004. Web taxonomy integration using support vector machines. In *Proc. of the 13th International Conference on World Wide Web* (WWW '04), 472-481. DOI=http://dx.doi.org/10.1145/988672.988736

[51] J Zhu, H Zou, S Rosset, and T Hastie. 2009. Multi-class adaboost. *Statistics and Its Interface* 2 (2009), 349-360.