A Weighting Scheme Based on Emerging Patterns for Weighted Support Vector Machines

Hongjian Fan
Department of Computer Science and Software Engineering
The University of Melbourne, VIC 3010, Australia
Email: hfan@cs.mu.oz.au

Kotagiri Ramamohanarao
Department of Computer Science and Software Engineering
The University of Melbourne, VIC 3010, Australia
Email: rao@cs.mu.oz.au

***Abstract*— Support Vector Machines (SVMs) are powerful tools for solving classification problems and have been applied to many application fields, such as pattern recognition and data mining, in the past decade. Weighted Support Vector Machines (weighted SVMs) extend SVMs by considering that different input vectors make different contributions to the learning of decision surface. An important issue in training weighted SVMs is how to develop a reliable weighting model to reflect the true noise distribution in the training data, i.e., noise and outliers should have low weights. In this paper we propose to use Emerging Patterns (EPs) to construct such a model. EPs are those itemsets whose supports in one class are significantly higher than their supports in the other class. Since EPs of a given class represent the discriminating knowledge unique to their home class, noise and outliers should contain no EPs or EPs of the both contradicting classes, while a representative instance of the class should contain strong EPs of the same class. We calculate numeric scores for each instance based on EPs, and then assign weights to the training data using those scores. An extensive experiments carried out on a large number of benchmark datasets show that our weighting scheme often improves the performance of weighted SVMs over SVMs. We argue that the improvement is due to the ability of our model to approximate the true distribution of data points.**

I. Introduction

Support Vector Machines (SVMs) were introduced in the early 1990s and the topic on the entire family of kernel-based learning methods has developed into a very active field of Machine Learning research [1], [2]. SVMs are based on statistical learning theory developed by Vapnik and their formulations embody the structural risk minimization (SRM) principle. Due to their good generalization ability for a wide range of applications, SVMs have been powerful tools for solving classification problems, delivering state of the art performance in applications from analysis of DNA microarray data to text categorization, from handwritten digits recognition to protein homology detection.

SVMs combine two key ideas. The first is the concept of an *optimum margin classifier*. An optimum margin classifier is a linear classifier; it constructs a separating hyperplane which maximizes the distance to the training points. The important point is maximization of the margin, which turns the under-specified learning problem into an optimization problem (without local optima) and has been shown to give very good generalization performance. In general, the optimum margin hyperplane will be a linear combination of the input vectors; *support vectors* are those training instances which obtain a non-zero coefficient, i.e., the ones that lie closest to the separating hyperplane. The second key concept underlying SVMs is a *kernel*. In its simplest form, a kernel is a function which calculates the dot product of two training vectors. Intuitively, this dot product expresses the similarity of the two training instances in terms of the given attributes. If we use feature transformation techniques to reformulate the input vectors in terms of new features and find a way to calculate dot products in this feature space, we can leave the linear classifier unaffected. Generally, the kernel can be thought of as a non-linear similarity measure and kernel functions are inner products in some feature space (potentially very complex).

Because the optimal hyperplane obtained by a SVM depends on only a small part of the data points (support vectors), it may become sensitive to noise or outliers in the training set, as shown in previous works [3]. To address this problem, Weighted Support Vector Machines (weighted SVMs)[1] are proposed in [4], which associate a weight with each data point such that different points can have different impacts on the learning of the optimal separating hyperplane. Weighted SVMs try to maximize the margin like the classical SVMs, but use weights to prevent some points (i.e., noise or outliers) from making narrower margin. If the data points are already associated with the weights, it is straightforward to use this information to train weighted SVMs. If a noise distribution model of the data is given, we can set the weight as the probability of the point that is not a noise, or as a function of it. However, almost all real world applications lack the weight information and it is very hard to know the true noise distribution. It is an open issue how to generate a good and reliable weighting model from data without any domain knowledge. In this paper we propose to use Emerging Patterns as a basis to construct such a model.

Patterns are conjunctions of simple conditions, where each conjunct is a test of the value of one of the attributes. Emerging Patterns (EPs) [5] are defined as multivariate features (i.e., patterns or itemsets) whose supports (or frequencies) change *significantly* from one class to another. Since EPs capture the knowledge of sharp differences between data classes, effective classification systems based on EPs have been built

[1]Although it is called Fuzzy Support Vector Machine in the work [4], the fuzzy membership for each training instance is effectively weight. In order to avoid confusion, we use the term "weighted SVMs" in this paper.

[6], [7], which are competitive to other existing state-of-the-art classifiers. From our experiences in using EPs for classification, we find that (1) most instances contain only the EPs of the same class; (2) some contain a small number of EPs from both classes; (3) a few do not contain any EPs. Intuitively, instances in case (1) are good representatives of their class, since they contain only the "signatures" of the home class; while instances in case (2) and (3) are very possibly noise or outliers, since they contain the "signatures" of both classes or no "signatures" at all. To implement the above idea, we can calculate a score for each instance, depending on the EPs contained in it. For a positive instance, it should contain as many positive EPs as possible, and as few negative EPs as possible. Its score is the aggregation of positive EPs minus the aggregation of negative EPs. We can turn the scores into weights: we define the weights as a function of the scores.

In this paper, we propose to use Emerging Patterns to assign a weight to each data point, which can reflect relative degrees of data points as meaningful data. Our method does not need any domain knowledge of noises or outliers – it is automatic and completely data-driven. Our method does not assume any class centers. In high dimensional space, the notion of center is subtle. What is worse, noise and outliers may skew the mean (center) significantly. While previous distance based method for assigning weights can only cope with convex shape classes, our method has the potential to deal with arbitrary shapes. The experiments show that weighted SVMs using our EP-based weighting model often achieve generalization error superior to other methods on benchmark datasets, which confirms the advantages of our model.

Organization: The remaining of the paper is organized as follows. In Section II, we give the background of Weighted SVMs. Section III introduces our weighting model based on Emerging Patterns. In Section IV, we discuss related work. In Section V, we provide experimental results using a number of benchmark databases and show that our weighting model is very reliable for weighted SVMs. Finally, we summarize our work in Section VI.

II. Weighted Support Vector Machines

Before we analyze Weighted Support Vector Machines (weighted SVMs), we first briefly introduce the theory of Support Vector and Kernel Methods for Learning using a two-class classification problem.

The theory of SVM is very powerful for solving classification problems, but it has some limitations. It treats all training points of a given class uniformly. However, in many real world application domains, not all training data points are equally important for classification purpose. Weighted Support Vector Machines (weighted SVM) [4] is proposed to address this issue: it treats each individual point differently, according to its weight. Weighted SVMs makes more efforts to correctly classify more important points (with larger weights) while not caring much about less important points (with lower weights, probably noise or outliers) whether or not they are misclassified.

Suppose we are given a set of labelled training data associated with *weights*

$$(x_1, y_1, s_1), \cdots, (x_m, y_m, s_m) \in \mathcal{R}^N \times \{\pm 1\}, \quad (1)$$

where $\epsilon \leq s_i \leq 1$ is a weight for $(x_i, y_i)(i = 1, \cdots, m)$ and $\epsilon > 0$ is sufficiently small. We will discuss in detail how to obtain reliable weights for training data in Section III.

Like SVM, the basic idea of weighted SVM is to maximize the margin of separation and minimize the classification error to achieve good generalization ability. Unlike SVM, weighted SVM uses a function of weights to reduce the effect of less important data points (i.e., increase the effect of more important points).

The optimal hyperplane problem in the case of weighted training data is then

$$\text{minimize } \tau(\mathbf{w}, \xi, \mathbf{s}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} s_i \xi_i \quad (2)$$

$$\text{subject to } y_i \cdot ((\mathbf{w} \cdot \mathbf{x}) + b) \geq 1 - \xi_i,\ i = 1, 2, \cdots, m \quad (3)$$

$$\xi_i \geq 0,\ i = 1, 2, \cdots, m \quad (4)$$

where C is a constant. Note that a small s_i reduces the effect of the parameter ξ_i in the optimization problem, thus the corresponding point (x_i, y_i) is regarded as less important for classification.

The above optimization problem can be transformed into

$$\text{maximize } W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j) \quad (5)$$

$$\text{subject to } \sum_{i=1}^{m} \alpha_i y_i = 0,\ 0 \leq \alpha_i \leq s_i C,\ i = 1, 2, \cdots, m \quad (6)$$

where the kernel function $K(\cdot, \cdot)$ measures the similarity between two points.

Note that if we set all s_i as 1, the weighted SVM will be the same as the traditional SVM. With different values of s_i, we can control the tradeoff of the corresponding training point x_i in the system. A smaller value of s_i makes x_i less important in determining the optimal separating hyperplane; and vice versa. There is only one free parameter (i.e., C) in SVMs while the number of free parameters in weighted SVMs is equal to the number of training points.

III. A Weighting Scheme Based on Emerging Patterns

Before introducing our weighting model, we give background of EPs first.

A. Emerging Patterns

Attributes can be categorical or continuous. For a continuous attribute, we assume that its value range is discretized into intervals. We call each (attribute, categorical-value) or (attribute, continuous-interval) pair an *item*. $(sex, male)$ and $(age, [18, 60])$ are two examples of items. Let I denote the

set of all items. A set X of items is also called an itemset, which is defined as a subset of I. We say any instance S contains an itemset X, if $X \subseteq S$. The support of an itemset X in a dataset D, $supp_D(X)$, is $count_D(X)/|D|$, where $count_D(X)$ is the number of instances in D containing X. Given two different classes of datasets D_1 and D_2, the growth rate of an itemset X from D_1 to D_2 is defined as GrowthRate(X) = $supp_2(X)/supp_1(X)$ if $supp_1(X) \neq 0$; otherwise, GrowthRate(X) = ∞.

Definition 1 Given a growth rate threshold $\rho > 1$, an itemset X is said to be an *ρ-Emerging Pattern* (ρEP or simply EP) from a background dataset D_1 to a target dataset D_2 if GrowthRate(X)$\geq \rho$.

When D_1 is clear from context, an EP X from D_1 to D_2 is simply called an EP of D_2. The support of X in D_2, $supp_2(X)$, denoted as $supp(X)$, is called the support of the EP.

There are several algorithms for efficiently discovering EPs [5]. In this work, we use the recent tree-based methods [8] (please refer to the paper for more details).

B. Using EPs to Calculate Importance Scores

Due to their high support in the home class and low support in the contrasting class, EPs can be regarded as strong signals that distinguish classes of data. Intuitively, a good instance should provide strong EPs of the same class - it should contain strong signals that are unique to this class. A bad instance (i.e., noise or outlier), however, may contain no EPs, or EPs of both contrasting classes with approximately equal strength.

The input is a training dataset D without weights associated with each instance. Suppose there are $k(k \geq 2)$ classes in the training dataset. We partition D into k datasets, $D_1, D_2, \cdots, D_k$, where $D = D_1 \cup D_2 \cup \cdots \cup D_k$. We discover EPs from the training dataset and obtain a set of EPs for each class. Let E_i denote the set of EPs from D_i, which have significantly higher support in D_i than in $(D - D_i)$. To assign weights, we use EPs to compute a score for each instance, reflecting the relative importance for determining the decision surface.

Definition 2 Given an instance $T \in D_v (1 \leq v \leq k)$ and k sets of EPs, one set per class, $E_1, E_2, \cdots, E_k$, the **importance score (or score)** of T

$$score(T) = \sum_{X \subseteq T, X \in E_v} \frac{GR(X)}{GR(X)+1} * supp(X)$$

$$- \sum_{i=1, i \neq v}^{k} \sum_{Y \subseteq T, Y \in E_i} \frac{GR(Y)}{GR(Y)+1} * supp(Y).$$

As shown in Definition 2, when calculating the importance score for T, we first aggregate the strength of EPs that are contained in T and from the same class, then we deduct the strength of EPs from the other classes if they are also contained in T.

C. Mapping Importance Scores to Weights

Having an importance score for each training instance, we need a mapping from these scores (range $[-\infty, +\infty]$) to weights (range $[0, 1]$). We perform this mapping class by class, that is, we look at scores of one class at a time.

Let `maxS` and `minS` be the maximum and the minimum score for a given class. We could use the following mapping:

$$x \mapsto \frac{x - \texttt{minS}}{\texttt{maxS} - \texttt{minS}}, \tag{7}$$

where x is an importance score of a given instance. Instead of linear mapping, logarithm can be used:

$$x \mapsto \log(1 + (e-1)\frac{x - \texttt{minS}}{\texttt{maxS} - \texttt{minS}}). \tag{8}$$

There are several problems with this simple mapping.

1) If `maxS` is significantly higher than other values, we will end up with many small weights (closer to 0);
2) If `minS` is significantly lower than other values, we will end up with many big weights (closer to 1);
3) Usually, half training instances will be assigned a weight less than 0.5 and half more than 0.5.

Obviously, (1) and (2) are not good. (3) is not good because it regards too many examples as bad for classification by assigning small weights.

An instance is characterized by the values of attributes that measure different aspects of the instance. Data is made up of a set of instances, where each instance is an individual that can be used for concept learning. People collect data in order to discover useful information from data. Although mistakes occur in data collection due to human and machine errors, most of the data is good enough to represent the underlying concept. Therefore, we make the following assumptions:

- most examples are generally good for classification purpose;
- some can provide partially correct information;
- a few are outliers or noisy data which provide wrong information and are harmful to the classification task.

We design a more intelligent mapping function based on the above analysis. Suppose we have a set of scores from a given class, $s_1, s_2, \cdots, s_n$, where n is the number of instances in the class. We sort these scores by ascending order. Let $s_1 < s_2 < \cdots < s_n$. We partition all instances of the class into three groups according to their scores (shown in Figure 1) and perform the logarithmic mapping within each group.

- Group 1 is made up of 80% instances whose scores are among the top 80% highest. In order to deal with the issue (1) discussed above, we remove those extremely big scores. We then map these scores into (0.8, 1] using the above logarithm function E.q. 8, i.e., an instance with the minimum score within this group will have a weight slightly more than 0.8.
- Group 2 consists of 15% instances whose scores are in the middle range. We map these scores into (0.5, 0.8] using E.q. 8, i.e., an instance with the maximum score within this group will exactly have a weight of 0.8.

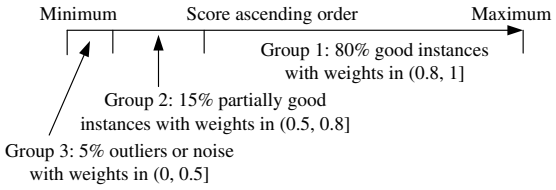

Fig. 1. Mapping importance scores into weights by forming three groups. Instances are sorted in the score-ascending order

- Group 3 includes the remaining 5% instances whose scores are among the lowest 5%. Instances within this group can be safely regarded as outliers or noise. We map these scores into (0, 0.5] using E.q. 8, i.e., an instance with the maximum score within this group will have a weight of 0.5. Note that the minimum score is zero when an instances contains no EPs or contains contradicting EPs with equal strength. We assign these zero-score instances with a very small weight (e.g., 0.001).

Note that the percentage for Group 1 does not have to be 80%. Other percentage (such as 90%, 70% and 60%) can be used. Our study indicates that 80% for Group 1, 5% for Group 2 and 15% for Group 3 are reasonably good choices.

D. Overall Procedure

The general approach of our model is shown in Figure 2. Starting with the original training dataset without weights, we try to assign a weight for each training example. Because EPs are defined on discrete attribute values, before discovering EPs, we discretize continuous attribute values using the entropy-based method [9]. For datasets containing more than two classes, we use the one-against-all class binarization technique to handle them: we take each class in turn and discover EPs that discriminate this class from all other classes. For example, given a three-class dataset ($D = D_1 + D_2 + D_3$), we will generate three sets of EPs, one for each class: (1) EPs from $D_2 \cup D_3$ to D_1; (2) EPs from $D_1 \cup D_3$ to D_2 and (3) EPs from $D_1 \cup D_2$ to D_3. The discovered EPs will be feeded into the scoring function to compute importance scores for each training instances. We then map the score into weights [0, 1]. The final result is the original training dataset with the generated weights, which is ready for weighted Support Vector Machines. Note that all optimization techniques for SVMs can be applied to weighted SVMs.

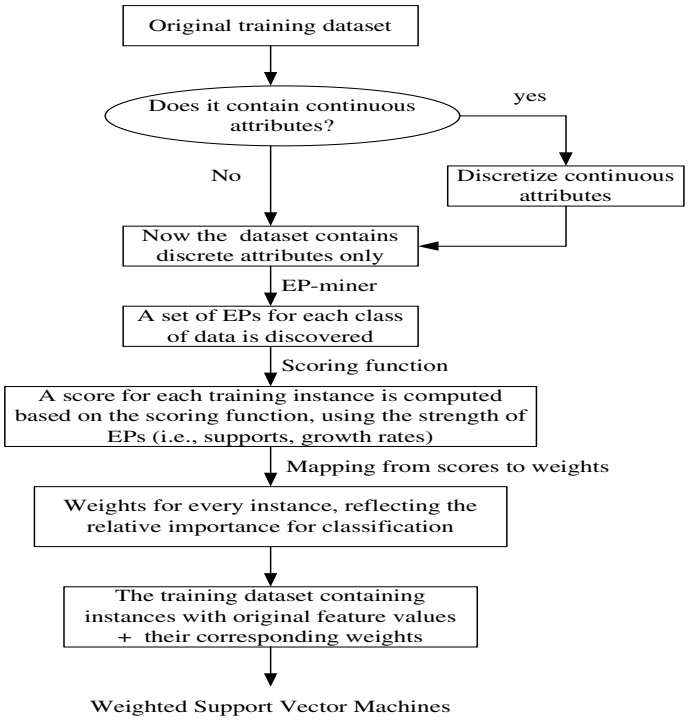

Fig. 2. Overall procedure of the EP-based weighting model

IV. Related Works

In [10], a method based on relative distances from the means of the classes is proposed to assign fuzzy membership values to the training data for the fuzzy perceptron. As an example gets closer to its mean and far away from the other mean, the membership values approach 1.0 exponentially. The fuzzy value is 0.5 when the example is equal to the mean of the other class and near 0.5 if it is equidistant from the two means. Although their fuzzy perceptron improves upon the crisp perceptron, their method for assigning fuzzy memberships has several weakness. It is applicable only when the mean is defined. What about categorical data? It makes the assumption that classes have convex shapes. What about classes with arbitrary shapes? In addition, noisy data and outliers may distort the mean significantly. Our EP-based weighting model is able to overcome these difficulties, which is supported by the experimental results.

In [4], fuzzy membership values (weights) are generated in the sequential learning context, where fuzzy membership (weight) s_i is a function of time t_i. They regard the last point (the latest arrived point) as the most important and the first point (the first arrived point) as the least important. Both linear function and quadric function are used, such as $s_i = f(t_i) = at_i + b$ and $s_i = f(t_i) = a(t_i - b)^2 + c$. However, no general method is given to address the typical classification task.

In [11], based on geometry properties of distribution of the training points in space, the guard vector method and the circle method are proposed to determine fuzzy memberships of the training points. But these methods may not work well in high dimensional space where data is sparse due to the curse of dimensionality.

In [12], [13], Fuzzy Support Vector Machines (FSVMs) and Fuzzy Least Squares Support Vector Machines (fuzzy LS-SVMs) are proposed to address the issues of unclassifiable regions in n-class classification problem. Their FSVMs are different from the weighted SVMs discussed in this work: the weighted SVMs aim to solve two-class problem well by weighting training instances in deciding the hyperplane; while their FSVMs treat all instances equally in training and generate fuzzy (instead of crisp) classification.

V. Performance Evaluation

In order to evaluate the effectiveness of our EP-based weighting model, we carry out a number of experiments. We select 29 benchmark datasets from the UCI Machine Learning Repository [14]. All experiments were conducted on a Dell PowerEdge 2850 (Dual Intel Xeon 3.0GHz CPU, 4G RAM) running Solaris 9/x86. The accuracy was obtained by using the methodology of *stratified* ten-fold cross-validation (CV-10). Results are reported as the mean classification performance over the 10 folds.

The complexity constant C is set to 5. Two types of kernels are used: one is the polynomial kernel; the other is the Radial-Basis Function (RBF) kernel. We use WEKA's Java implementation of SVM [15]. The commands are `java weka.classifiers.functions.SMO -C 5` and `java weka.classifiers.functions.SMO -C 5 -R`. Note that SVM is a special case of weighted SVM, where all training instances have the same weight 1.

A. EP-based Weighting Model Vs. Random Weighting Model

To validate the robustness of our EP-based weighting method, we perform the following control experiment: we assign random weights to the training instances and then feed them into weighted SVMs. Our hypothesis is that the performance of our EP-based model is better than SVM, and SVM better than random weighting model.

Table I compares SVMs, weighted SVMs with random weighting model and weighted SVMs with weights generated by our EP-based model. Both polynomial kernel and RBF kernel are used. We draw some interesting points as follows:

- When randomly assigning weights to the training instances, the weighted SVMs trained on these instances always perform worse than SVMs that treat all instances equally, no matter which kernel is used. This is as expected. Since weighted SVMs use weights to determine the importance of instances, wrong assignments of weights (either giving a bad instance a very high weight or giving a good instance a very low weight) will mislead weighted SVMs to generate non-optimal hyperplanes, either overfitting due to outliers and noise or missing reliable support vectors.
- When using our EP-based method to assign weights, weighted SVMs almost always outperform SVMs on both kernels. Paired-differences t tests show that (1) when using the polynomial kernel, SVM plus our EP-model is statistically significantly (at the 0.05 level) better than plain SVM on 12 datasets; (2) when using the RBF kernel, SVM plus our EP-model is statistically significantly better than plain SVM on 10 datasets; (3) on both kernels, SVM plus our EP-model is never statistically significantly worse. They tie on the Balance-scale and Mushroom datasets when using the polynomial kernel; they tie on Balance-scale, Breast-w, Glass, Lymph, Mushroom and Sick datasets when using the RBF kernel. In terms of the average accuracy, weighted SVM combining our EP model beats SVMs by around 2%. This shows that our EP based model is able to effectively distinguish good instances from bad ones: good ones are given high weights and hence play an important role in deciding hyperplanes; while outliers are given very low weights and hence do not affect the hyperplanes.
- Our proposed method weighted SVM (combining our EP model) increases the accuracy significantly (e.g., more than 5%) in some cases. For example, using the polynomial kernel, the increase is 7.32% on Auto, 5.6% on Breast-cancer and 8.88% on Labor. Using the RBF kernel, the increase is up to 8.27% on Vehicle. We found that EP-based classifiers also perform well on these datasets. This suggests that the improvement is not due to chance, but rather because of the usefulness of EPs.

TABLE I

Accuracy comparison. wSVM refers to Weighted SVMs. 'Rand' means assigning random weights to instances. 'EP' means our weighting method

	polynomial kernel			RBF kernel		
Dataset	SVM	wSVM Rand	wSVM EP	SVM	wSVM Rand	wSVM EP
anneal-ORIG	87.75	85.52	**89.20**	79.96	78.73	**80.51**
autos	71.22	65.85	**78.54**	56.98	52.20	**60.49**
balance-scale	**87.68**	87.04	**87.68**	**86.88**	84.72	**86.88**
breast-cancer	68.53	66.43	**74.13**	72.03	70.98	**74.13**
breast-w	97.00	96.71	**97.28**	**96.57**	95.42	**96.57**
cleve	81.52	81.52	**81.85**	82.18	81.52	**82.51**
colic	82.61	80.43	**82.88**	83.97	81.70	**84.24**
colic-ORIG	73.64	73.01	**77.99**	76.63	71.12	**77.99**
credit-a	84.78	84.49	**85.51**	85.51	84.93	**86.23**
credit-g	75.10	74.10	**76.10**	75.80	74.20	**76.20**
diabetes	77.34	75.52	**77.86**	66.02	65.10	**69.14**
glass	56.07	50.46	**57.48**	**45.33**	35.51	**45.33**
heart-c	83.83	80.53	**85.81**	83.17	82.18	**83.83**
heart-h	82.65	81.63	**82.99**	81.29	78.65	**81.97**
heart-statlog	83.70	81.48	**84.44**	82.59	81.48	**83.33**
hepatitis	83.87	81.29	**86.45**	82.58	79.35	**85.16**
hypothyroid	93.61	93.31	**94.01**	92.29	92.29	**92.44**
ionosphere	87.75	86.32	**89.17**	87.60	86.04	**88.32**
iris	96.00	94.67	**97.33**	92.67	86.67	**94.00**
labor	84.21	80.71	**92.98**	84.74	77.19	**87.72**
lymph	85.14	83.78	**87.16**	**80.41**	79.73	**80.41**
mushroom	**100.00**	**100.00**	**100.00**	**100.00**	99.90	**100.00**
pima	77.08	76.56	**77.34**	66.14	65.10	**66.93**
segment	93.07	91.60	**93.59**	86.88	85.50	**87.71**
sick	93.88	93.85	**95.79**	**93.88**	93.88	**93.88**
sonar	75.96	74.04	**80.77**	75.25	72.12	**79.33**
vehicle	74.35	72.22	**76.60**	52.25	48.10	**60.52**
vote	94.94	94.71	**95.63**	94.48	93.56	**95.63**
zoo	92.08	87.13	**94.06**	85.12	81.12	**87.13**
Average	83.63	81.89	**85.54**	80.32	77.90	**81.67**

B. EP-based Weighting Model Vs. Distance-based Model

As discussed before, weights can also be generated by distance-based method. For a two-class problem, suppose there exist two centers, one for each class. Weights are computed according to the distance to the centers: the closer to its own center, the higher the value; the farther away from its center (closer to the other center), the lower the value. The detailed procedure can be found in [10].

TABLE II

ACCURACY COMPARISON ON TWO-CLASS PROBLEMS. WSVM REFERS TO WEIGHTED SVMS. 'DIST' MEANS ASSIGNING WEIGHTS TO INSTANCES BASED ON DISTANCES. 'EP' MEANS OUR WEIGHTING METHOD

	polynomial kernel			RBF kernel		
Dataset	SVM	wSVM	wSVM	SVM	wSVM	wSVM
		Dist	EP		Dist	EP
breast-c	68.53	68.18	**74.13**	72.03	71.28	**74.13**
breast-w	97.00	96.71	**97.28**	**96.57**	96.42	**96.57**
cleve	81.52	**82.18**	81.85	82.18	82.18	**82.51**
colic	82.61	81.79	**82.88**	83.97	83.70	**84.24**
colic-orig	73.64	75.82	**77.99**	76.63	76.90	**77.99**
credit-a	84.78	85.22	**85.51**	85.51	85.51	**86.23**
credit-g	75.10	75.50	**76.10**	75.80	75.50	**76.20**
diabetes	77.34	76.62	**77.86**	66.02	65.23	**69.14**
heart-c	83.83	83.83	**85.81**	83.17	82.18	**83.83**
heart-h	82.65	81.63	**82.99**	81.29	80.95	**81.97**
heart-sta	83.70	**84.44**	**84.44**	82.59	**83.33**	**83.33**
hepatitis	83.87	**86.45**	**86.45**	82.58	82.65	**85.16**
iono	87.75	88.32	**89.17**	87.60	**89.17**	88.32
labor	84.21	91.23	**92.98**	84.74	84.21	**87.72**
mushroom	**100.00**	**100.00**	**100.00**	**100.00**	99.90	**100.00**
pima	77.08	76.49	**77.34**	66.14	66.02	**66.93**
sick	93.88	94.54	**95.79**	**93.88**	**93.88**	**93.88**
sonar	75.96	75.81	**80.77**	75.25	76.12	**79.33**
vote	94.94	95.40	**95.63**	94.48	95.40	**95.63**
Average	83.60	84.22	**85.53**	82.65	82.66	**83.85**

We compare our EP-based method with the distance based method. The results are shown in Table II. We performed paired-differences t tests and found that (1) when using the polynomial kernel, our EP-model is statistically significantly (at the 0.05 level) better than distanced based model on 6 datasets; (2) when using the RBF kernel, our EP-model is statistically significantly (at the 0.05 level) better than distanced based model on 7 datasets; (3) our EP-model is never statistically significantly worse, although it loses once and tie three times for polynomial kernels, and it loses once and draw twice for RBF kernels. Distance based method may yield worse results than SVMs while our method is cosistantly better than or equal to SVMs. In terms of average accuracy, our method is superior to the distance based method. Our EP model achieve much higher accuracy on some datasets. For example, our accuracy is 4.96% and 3.21% higher on Sonar dataset for the polynomial kernel and RBF kernel, respectively. The distance based method does not work well because of the following reasons.

- On one hand, it has a very high dimensionality (60 attributes). In such a high dimensional, it is unlikely that data points are closer to each other than the average distance between points. As a consequence, the difference between the distance to the nearest and the farthest neighbor of a data point goes to zero.
- On the other hand, there may exist several clusters inside a given class of data. So the computed class center may actually reside outside the class region, or even worse, it goes into the other class.

VI. CONCLUSIONS

We have proposed a reliable and robust weighting model based on the idea of Emerging Patterns. Incorporating our EP-based model into the weighted Support Vector Machine (weighted SVM), we are able to yield performance consistently better than SVM due to its ability to reduce the effects of noise and outliers. Experiments also show that our EP-based weighting scheme is often superior to the previous distance based weighting model. This supports the hypothesis that our EP model can deal with high dimensional data (where distance does not work) and cope with arbitrary class shape (no assumption about class centers).

ACKNOWLEDGMENT

The authors would like to thank Dr James Bailey for providing useful comments on a draft of this paper.

REFERENCES

[1] V. Vapnik, *Statistical Learning Theory*. NY: John Wiley, 1998.
[2] C. J. C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Knowledge Discovery and Data Mining*, vol. 2, no. 2, 1998.
[3] X.Zhang, "Using class-center vectors to build support vector machines," in *Proc. 1999 IEEE Workshop on Neural Networks for Signal Processing IX*, 1999.
[4] C.-F. Lin and S.-D. Wang, "Fuzzy support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, no. 2, pp. 464 – 471, March 2002.
[5] G. Dong and J. Li, "Efficient mining of emerging patterns: Discovering trends and differences," in *Proc. 5th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD'99)*, San Diego, CA, USA, Aug 1999, pp. 43–52.
[6] G. Dong, X. Zhang, L. Wong, and J. Li, "Caep: Classification by aggregating emerging patterns," in *Proc. 2nd Int'l Conf. on Discovery Science (DS'99)*, Tokyo, Japan, Dec 1999, pp. 30–42.
[7] J. Li, G. Dong, and K. Ramamohanarao, "Making use of the most expressive jumping emerging patterns for classification," *Knowl. Inf. Syst.*, vol. 3, no. 2, pp. 131–145, 2001.
[8] H. Fan and K. Ramamohanarao, "Efficiently mining interesting emerging patterns," in *Proc. 4th Int'l. Conf. on Web-Age Information Management (WAIM2003)*, Chengdu, China, Aug 2003, pp. 189–201.
[9] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger, "MLC++: a machine learning library in C++," *Tools with artificial intelligence*, pp. 740–743, 1994.
[10] J. M. Keller and D. J. Hunt, "Incorporating fuzzy membership functions into the perceptron algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 7, no. 6, pp. 693–699, 1985.
[11] L. Chu and C. Wu, "A fuzzy support vector machine based on geometric model," in *5th World Congress on Intelligent Control and Automation (WCICA2004)*, vol. 2, 2004, pp. 1843 – 1846.
[12] T. Inoue and S. Abe, "Fuzzy support vector machines for pattern classification," in *Proc. Int'l Joint Conf. on Neural Networks IJCNN'01*, vol. 2, 2001, pp. 1449 – 1454.
[13] D. Tsujinishi and S. Abe, "Fuzzy least squares support vector machines for multiclass problems," *Neural Networks*, vol. 16, no. 5-6, pp. 785–792, 2003.
[14] C. L. Blake and C. J. Merz, "UCI repository of machine learning databases," 1998. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html
[15] I. H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, CA, 1999.

Author/s:
FAN, HONGJIAN; Kotagiri, Ramamohanarao

Title:
A Weighting Scheme Based on Emerging Patterns for Weighted Support Vector Machines

Date:
2005

Citation:
Fan, Hongjian and Kotagiri, Ramamohanarao (2005) A Weighting Scheme Based on Emerging Patterns for Weighted Support Vector Machines, in Proceedings, IEEE International Conference on Granular Computing (GrC 2005).

Publication Status:
Published

Persistent Link:
http://hdl.handle.net/11343/33835