EP-based Robust Weighting Scheme for Fuzzy SVMs

Shaoyi Zhang[1], Kotagiri Ramamohanarao[1] and James C. Bezdek[2]
[1]Department of Computer Science and Software Engineering, University of Melbourne, VIC 3010, Australia
[2]Department of Computer Science, University of West Florida, Pensacola, FL 32514, USA

{shaoyi, rao}@csse.unimelb.edu.au; jbezdek@uwf.edu

Abstract

Support vector machine (SVM) classifiers represent one of the most powerful and promising tools for solving classification problems. In the past decade SVMs have been shown to have excellent performance in the field of data mining. The standard SVM classifier treats all instances equally. However, in many applications we have different levels of confidence in different instances that belong to a particular class. Fuzzy SVMs have been used to recognize the importance of each training instance. Although these schemes are called fuzzy SVMs, they are basically trained by weighted training instances. In this paper we propose a new robust weighting scheme for the class memberships for fuzzy SVM classifier. The weighting scheme is a sophisticated and effective method for weighting the training instances which makes use of highly discriminating patterns called emerging patterns (EPs). Our experiments show that this new weighting method has excellent performance and noise tolerance compared to the weighting scheme previously proposed.

Keywords: Classification, data mining, support vector machines, weighting schemes.

1 Introduction

The concept of *support vector machines* (SVMs) was developed by Vapnik in the early 1990s, based on *statistical learning theory* (SLT) and the principle of *structural risk minimization* (SRM) [1]. SVMs have gained wide acceptance due to their high generalization ability and better performance than many traditional learning methods over a wide range of applications. In many applications the SVM provides better generalization performance and less overfitting than other learning techniques such as *artificial neural networks* (ANNs) [1]. For example, SVMs have been effectively applied in many classification and recognition fields, such as isolated handwritten digit recognition, object recognition, speech recognition, and spatial data analysis [2].

In principal, the SVM uses a mapping ϕ that transforms vectors from the original *labeled* 2-class input (object) data into vectors in a high dimensional *feature space*. In the new space it may be possible to construct a separating hyperplane between the two (imaged) classes of labeled training data. The hyperplane is then pulled back to the input space via inverse image algebra, where it becomes a (usually) non-linear decision that separates the labeled input data. Subsequent input data are classified by their location in one of the two decision regions defined by the non-linear boundary. In practice, the input data are not mapped anywhere. Instead, inner products involving hypothetical pairs of data in the feature space are replaced by the value of a kernel function K on their "preimages", so that $(\phi(\boldsymbol{x}_i),\phi(\boldsymbol{x}_j)) = K(\boldsymbol{x}_i,\boldsymbol{x}_j)$. This device, universally known as the "kernel trick", is based on a very old theorem due to Mercer [1], and renders the SVM idea feasible in practice.

The standard SVM classifier assumes that each training instance belongs unequivocally to only one class, and further, that all instances are equally important for classification. In addition, we usually assume that the class labels are accurate. But real-world data do not always belong unequivocally to one class (e.g., hybrids almost always deserve partial membership in two or more progenitor classes). Moreover, class labels are not always correct because of noise or a lack of expert knowledge. These two problems affect the optimal hyperplane obtained by an SVM. This classifier depends on only a small fraction of the instances (i.e., on the *support vectors*, SVs). So, the SVM classifier can be unduly sensitive to noise and mislabeled instances in the data [23].

In this paper, we propose a new weighting scheme for fuzzy SVM classifier that allows each training data to possess a different level of importance. We define importance of an instance by how strongly it contributes in decision making. For example, an instance that has features that strongly determine a class is generally considered more important than an instance that has weak correlation with any of the classes. We find sub-weights (class-memberships) for each training instance indicating its relationship to the two input classes. The sub-weights are then merged, becoming a single weight associated with the instance. The training data are assigned different levels of contribution towards the fuzzy SVM based on these memberships. This modification is accomplished by reformulating the constrained optimization problem upon which the fuzzy SVM classifier is based. The usual construction follows the Lagrangian to a solution for the optimal hyperplane in the primal form, found in the dual form.

We employ the recently introduced idea of *emerging patterns* (EPs) to compute sub-weights for class memberships. EPs are defined as itemsets whose supports (probabilities) increase significantly from one class to another [7, 8]. The discriminating power of EPs is in most cases proportional to their growth rates [11]. The growth rate of an EP is the ratio of its support in one class to that in the other class. EPs have had a great impact in many applications such as rare-class classification, and expansion of training data [9, 13]. Although mining EPs (like mining association rules) is a time consuming task in data mining, they can capture significant changes between datasets [10, 12]. Hence, EPs can be used to build robust, accurate classifiers. We make use of EPs to determine the importance of each instance before building a SVM classifier. This type of SVM is as computationally efficient as the standard SVM because EPs do not play a role during decision-making.

The remainder of the paper is organized as follows. Section 2 reviews the basic theory of the standard SVM and fuzzy SVM classifiers. In section 3, we briefly review EPs. Section 4 introduces our quality weighting model based on EPs. Section 5 reports our experiments. In section 6, we compare our method with other weighting methods. Finally, we summarize our work and list some ideas for future research in section 7.

2 Standard & Fuzzy SVM Classifiers

In this section, we briefly introduce the theory of standard and fuzzy SVM classifiers.

2.1 Standard SVM Classifier

Suppose we are given a set of labeled training data

$$\{(\boldsymbol{x}_i, y_i): i = 1, ..., n\} \subset \Re^N \times \{1, -1\} \qquad (1)$$

Each input vector $\boldsymbol{x}_i$ is considered to be a full member of either of two classes (called, simply, + and –), its membership indicated by the class label $y_i \in \{-1, 1\}$ for i=1, …, n. We wish to find a hyperplane

$$\mathbf{w}^T \boldsymbol{x}_i + b = 0 \qquad (2)$$

defined by the pair (w, b), such that we can separate the points $\boldsymbol{x}_i$ by the function

$$f(\boldsymbol{x}_i) = \mathrm{sign}(\mathbf{w}^T \boldsymbol{x}_i + b) = \begin{cases} 1, & if\ y_i = 1; \\ -1, & if\ y_i = -1. \end{cases} \qquad (3)$$

The set X={$\boldsymbol{x}_1$, …, $\boldsymbol{x}_n$}is said to be *linearly separable* if there exists ($\mathbf{w}$, b) such that the inequality

$$y_i(\mathbf{w}^T \boldsymbol{x}_i + b) \geq 1 \qquad (4)$$

is valid for all elements of X. When X is linearly separable, we can find a unique optimal hyperplane (called the SVM) for which the margin between the projections of the training points from the two classes onto the hyperplane is maximized. If the set is not linearly separable, classification violations occur in the SVM formulation. To deal with data that are not linearly separable, the previous model is generalized by introducing n nonnegative (slack) variables ξ_i such that

$$y_i(\mathbf{w}^T \boldsymbol{x}_i + b) \geq 1 - \xi_i \qquad (5)$$

where $\xi_i \geq 0$ for those points which do not satisfy (4).

The optimal hyperplane is found as the solution to the optimization problem:

$$\text{minimize } \tau(\mathbf{w}, \xi) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \qquad (6)$$

$$\text{subject to } y_i(\mathbf{w}^T \boldsymbol{x}_i + b) \geq 1 - \xi_i,\ i=1, \dots, n \qquad (7)$$

where constant C is regarded as a regularization parameter. C is the only free parameter in the SVM classifier formulation. Tuning this parameter balances margin maximization against classifier error. We construct the Lagrangian of τ:

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}^T \boldsymbol{x}_i + b) - 1 + \xi_i) - \sum_{i=1}^{n}\beta_i\xi_i \qquad (8)$$

Differentiating L with respect to $\mathbf{w}$ and b, and setting the results equal to zero yields the first order necessary conditions that solutions must satisfy:

$$\frac{\partial L}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n}\alpha_i y_i \boldsymbol{x}_i = 0 \qquad (9)$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n}\alpha_i y_i = 0 \qquad (10)$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \beta_i = 0 \qquad (11)$$

Solving these equations and back substituting into the original optimization problem converts it into the dual problem for data in the input space:

$$\text{maximize } Q(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T \boldsymbol{x}_j \qquad (12)$$

$$\text{subject to } \sum_{i=1}^{n}\alpha_i y_i = 0,\ 0 \leq \alpha_i \leq C,\ i = 1, \dots, n \qquad (13)$$

where α = (α_1, …, α_n) is the vector of nonnegative Lagrange multipliers associated with the constraints. The solution for this problem satisfies

$$\alpha_i(y_i(\mathbf{w}^T \boldsymbol{x}_i + b) - 1 + \xi_i) = 0 \qquad (14)$$

$$\beta_i\xi_i = 0 \qquad (15)$$

If we cannot find a hyperplane that (linearly) separates the X in the input space into its two labeled classes with high classification accuracy, we consider the possibility of transforming X to a higher dimensional feature space, say $Z = \phi(X)$. It is our hope that the extracted data Z are linearly separable in $\phi(\Re^N)$. The property of the SVM classifier that renders it feasible is that it is *not* necessary to find the nonlinear mapping ϕ. Instead, we need only choose a *kernel function* $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that satisfies Mercer's theorem [1], for then dot products in feature space take values $(\phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j)) = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. When using a kernel function, the dual problem (for vectors in feature space) becomes

$$\text{maximize } Q(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i,\boldsymbol{x}_j) \quad (16)$$

$$\text{subject to } \sum_{i=1}^{n}\alpha_i y_i=0,\ 0\le\alpha_i\le C,\ i=1,\ldots,n \quad (17)$$

The solution also satisfies (14) and (15), and is available via quadratic programming. Two kernels are used to construct SVM and fuzzy SVM classifiers in this paper.

Polynomial kernel:

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=(\boldsymbol{x}_i^T\boldsymbol{x}_j+\theta)^d \quad (18)$$

Radial-basis function (RBF) *kernel*:

$$K(\boldsymbol{x}_i,\boldsymbol{x}_j)=\exp\left(-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}{2\sigma^2}\right) \quad (19)$$

2.2 Fuzzy SVM Classifier

The fuzzy SVM has proposed to extend SVM by G.C Cawley et al [26], C. Lin et al. [4] and H. Huang et al. [5]. Their schemes are identical. These authors call their models *fuzzy* SVMs because they determine the importance of each training instance from fuzzy membership values. However, their schemes still produce crisp labels, i.e., they are still crisp classifiers. In this paper we use the term "soft SVM" to avoid confusion with earlier designs. Our classifier is also crisp, but we build the fuzzy SVM classifier by replacing the fuzzy memberships of each instance with a single weight based on EPs. There are other classifiers called "fuzzy SVMs" by their authors [3, 6, 20, 21, 22], but they are not based on instance weights or fuzzy memberships. We do not discuss these alternate designs in this article, but we do compare our design to the fuzzy SVM of Lin et al. [4].

Similar to (1), suppose we are given a set of labeled training data associated with weights

$$\{(\boldsymbol{x}_i,y_i,\mu_i):i=1,\ldots,n\}\subset\Re^N\times\{1,-1\}\times[0,1] \quad (20)$$

where $\mu_i\in[0,1]$ is a weight which indicates the *importance* of $(\boldsymbol{x}_i, y_i)$ in the determination of a SVM classifier. Normally μ_i is not less than ε, which is a sufficiently small positive number. We discuss in detail how to obtain reliable weights for training data in section 4.

Analogous to the standard SVM classifier, the basic idea of the fuzzy SVM classifier is to maximize the margin of separation whilst minimizing the training error, in order to achieve good generalization ability. On the other hand, unlike the SVM classifier, a fuzzy SVM classifier uses a function of the weights to reduce the effect of less important data points (i.e., increase the effect of more important points). The optimal hyperplane problem for this case, using weighted training data is the solution of the primal problem

$$\text{minimize } \tau(\mathbf{w},\xi,\mu)=\frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{n}\mu_i\xi_i \quad (21)$$

$$\text{subject to } y_i(\mathbf{w}^T\boldsymbol{x}_i+b)\ge 1-\xi_i,\ i=1,\ldots,n \quad (22)$$

Note that a small μ_i reduces the effect of the parameter ξ_i in the optimization problem. This means that the corresponding point $(\boldsymbol{x}_i, y_i)$ is regarded as less important for building the optimal classifer than points with higher weight values. The Lagrangian function becomes

$$L=\frac{1}{2}\|\mathbf{w}\|^2+C\sum_{i=1}^{n}\mu_i\xi_i-\sum_{i=1}^{n}\alpha_i(y_i(\mathbf{w}^T\boldsymbol{x}_i+b)-1+\xi_i)-\sum_{i=1}^{n}\beta_i\xi_i \quad (23)$$

By differentiating L with respect to $\mathbf{w}$, b and ξ_i, and setting the results equal to zero, we obtain the first order necessary conditions for a solution:

$$\frac{\partial L}{\partial\mathbf{w}}=\mathbf{w}-\sum_{i=1}^{n}\alpha_i y_i\boldsymbol{x}_i=0 \quad (24)$$

$$\frac{\partial L}{\partial b}=-\sum_{i=1}^{n}\alpha_i y_i=0 \quad (25)$$

$$\frac{\partial L}{\partial\xi_i}=\mu_i C-\alpha_i-\beta_i=0 \quad (26)$$

Solving these and back substituting in the usual way transform the primal optimization problem into its dual

$$\text{maximize } Q(\alpha)=\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j\boldsymbol{x}_i^T\boldsymbol{x}_j \quad (27)$$

$$\text{subject to } \sum_{i=1}^{n}\alpha_i y_i=0,\ 0\le\alpha_i\le\mu_i C,\ i=1,\ldots,n \quad (28)$$

and the solution satisfies (14) and (15).

It is clear that the only difference between the standard SVM classifier and the fuzzy SVM classifier is the upper bounds of Lagrange multipliers $\{\alpha_i\}$ in the dual problem. In the standard SVM classifier, the $\{\alpha_i\}$ are bounded by a single constant C, while in the weighted formulation they are bounded by dynamical values that are functions of the corresponding membership values in the fuzzy SVM classifier. The lower the membership value of a data point $\boldsymbol{x}_i$ is to its own class, the narrower the feasible region is along the α_i axis.

A fuzzy SVM classifier may maximize the margin like a standard SVM classifier, and correctly classify more important points (with higher weights) while preventing less important points (with lower weights, probably noise or outliers) from making the margin narrower, whether or not they are misclassified. So, different data points can have different impacts during learning of the optimal separating hyperplane. If every instance has the weight μ_i=1, the fuzzy SVM classifier reduces to the standard SVM classifier. With different values of μ_i, we can control the tradeoff of the corresponding training point $\boldsymbol{x}_i$. Consequently, the effectiveness of the fuzzy SVM classifier depends on the choice of the weights $\{\mu_i\}$. In section 4 we show how to compute the $\{\mu_i\}$ based on *emerging patterns* (EPs).

3 Emerging Patterns (EPs)

For a dataset D, an instance I in D is $I = \{(A_1=a_1), (A_2=a_2), (A_3=a_3), \ldots, (A_n=a_n)\}$, where $a_1, a_2, a_3, \ldots, a_n$ are values related to the attributes$\{A_1, A_2, A_3, \ldots, A_n\}$. We call each *pair* (A, a) an *item* [8]. Let Z denote the set of all items in an encoding dataset D. Itemsets are subsets of Z. We say an instance Y contains an itemset X, if $X \subset Y$.

Definition 1. Given a dataset D and an itemset X, the support of X in D, $Supp_D(X)$, is defined as

$$Supp_D(X) = \frac{count_D(X)}{|D|} \qquad (29)$$

where $count_D(X)$ is the number of instances in D containing X.

EPs are itemsets whose supports change *significantly* from one class to another [7, 13]. EPs capture sharp differences between data classes, thus affording a competitive alternative to other existing state-of-the-art classifiers [24].

Definition 2. Given two different datasets D_1 and D_2, where instances in D_i belong to class C_i, let $Supp_{D_i}(X)$ denote the support of the itemset X in D_i. The growth rate of X from D_1 to D_2, $GR_{D_1 \to D_2}(X)$, is defined as

$$GR_{D_1 \to D_2}(X) = \begin{cases} 0, & \text{if } Supp_{D_1}(X) = 0 \text{ and } Supp_{D_2}(X) = 0; \\ \infty, & \text{if } Supp_{D_1}(X) = 0 \text{ and } Supp_{D_2}(X) \neq 0; \\ \dfrac{Supp_{D_2}(X)}{Supp_{D_1}(X)}, & \text{otherwise.} \end{cases} \qquad (30)$$

When D_1 is clear from the context, an EP e from D_1 to D_2 is called an EP of D_2; the support of e in D_2, $Supp_{D_2}(e)$, is simply denoted as the support of e, $Supp(e)$; and its growth rate from D_1 to D_2, $GR_{D_1 \to D_2}(e)$, is denoted as the growth rate of e, $GR(e)$.

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. Thus, EPs can be used to help find the class memberships for training instances and then to build the fuzzy SVM classifier. Section 4 presents our method for constructing the fuzzy SVM.

4 A Weighting Method Based on EPs

We can train the fuzzy SVM classifier directly if the training data already have associated weights $\{\mu_i \in [0, 1]\}$. And in this case, the weights are sometimes regarded as probabilities of the instances that represent their importance or meaning confidence. However, data collected in almost all real-world applications lacks information about weights and noise. Without any, or with little, prior information, it is very hard to generate a reliable weighting model from data and to find the true noise distribution. Therefore finding a good function to calculate the weights *from the data* is a primary concern when building a fuzzy SVM classifier. In this paper we propose a model based on using EPs.

First, we calculate a sub-weight for each class and instance, depending on the EPs contained, and then map the sub-weights of each instance into a single weight representing the importance of each point. Then we normalize these values to determine a final weight for each instance which reflects its relative importance for determining the decision surface.

4.1 Using EPs to Calculate Sub-weights

We discretize continuous attributes in the training instances so that we can extract EPs. (The fuzzy SVM classifier will be built using the original training instances.) Assume that after dicretization we have a set of training instances, $D = \{I_1, I_2, I_3, \ldots, I_n\}$, and a set of classes, $C = \{C_1, C_2, C_3, \ldots, C_m\}$. We partition D into m datasets, $D_1, D_2, \ldots, D_m$, where $D = D_1 \cup D_2 \cup \ldots \cup D_m$. E_k is a set of EPs extracted from the dataset related to class C_k such that the EPs in E_k have significantly higher support in D_k than in $\overline{D_k}$, which is the complementary set of D_k. The support of an EP $e \in E_k$ is $Supp_{D_k}(e)$ and the growth rate of it is $GR_{\overline{D_k} \to D_k}(e)$. The strength of e in class C_k, $Strength_k(e)$, is defined as follows [8]:

$$Strength_k(e) = \frac{GR_{\overline{D_k} \to D_k}(e)}{GR_{\overline{D_k} \to D_k}(e) + 1} \times Supp_{D_k}(e) \qquad (29)$$

$Strength_k(e)$ represents the contribution of $e \in E_k$ in class C_k. This contribution is proportional to both the growth rate (discriminating power) of e, $GR_{\overline{D_k} \to D_k}(e)$, and its support in the home class (C_k), $Supp_{D_k}(e)$. An EP might have a high growth rate and low support in its home class and hence, its strength will be low. Alternatively, an EP might have a low growth rate and high support in its home class, again resulting in low strength. That is, in order for an EP to be strong, it has to have both a high growth rate and high support.

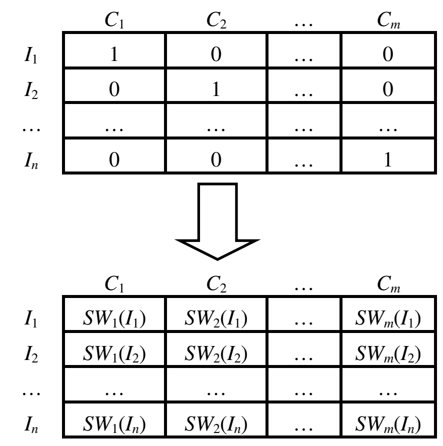
Getting all the EPs contained in an instance $I \in D$ for class C_k, we calculate the sub-weight $SW_k(I)$ of I for C_k, which is found by aggregating the contributions of these EPs.

$$SW_k(I) = \sum_{e \subseteq I, e \in E_k} Strength_k(e) \qquad (32)$$

Now we get the sub-weights of each instance for every class, no matter whether it is the instance's home class or not. The result of this, depicted schematically in Figure 1, is that the crisp instances are converted to weighted ones.

4.2 Merging Sub-weights of Each Instance to A Total-weight

For an m-class problem, we have m sub-weights for each instance. We need a reliable way to combine them into a



	C_1	C_2	…	C_m
I_1	1	0	…	0
I_2	0	1	…	0
…	…	…	…	…
I_n	0	0	…	1

	C_1	C_2	…	C_m
I_1	$SW_1(I_1)$	$SW_2(I_1)$	…	$SW_m(I_1)$
I_2	$SW_1(I_2)$	$SW_2(I_2)$	…	$SW_m(I_2)$
…	…	…	…	…
I_n	$SW_1(I_n)$	$SW_2(I_n)$	…	$SW_m(I_n)$

Fig. 1: Conversion of crisp instances to weighted ones.

single total-weight for each instance to build the fuzzy SVM classifier. We can do this for each instance *I* in class C_k by computing a total-weight using the sub-weights as follows:

$$TW''(I)=\frac{\sum_{p=1}^{m-1}\sum_{q=p+1}^{m}\left|SW_p(I)-SW_q(I)\right|}{\binom{m}{2}} \tag{33}$$

The function $TW''(I)$ in (33) uses absolute values of differences of sub-weight pairs. Although $TW''(I)$ indicates the difference between sub-weights very well, it still has a problem, viz., the value of the total-weight for an instance will be the same, no matter which home class it belongs to. For example, two instances in class C_1 with sub-weight sets (9, 1) and (1, 9) will have the same total-weight.

In real-world data there are usually noise and outliers, and some data may be misclassified. The performance of any classifier will improve if these problems are taken into account. Our method of calculating the sub-weights of classes usually results in the labeled class of a training instance having the highest sub-weight value. However, something on the order of 5%-15% of the instances in any training data seem to have a higher sub-weight in a class other than the marked class. We should not arbitrarily move such instances into the class with the highest sub-weight, but we can reduce weights of the instances whose labeled class sub-weight is much lower than the highest values. Consequently, for an instance *I* in home class C_k, we modify (33) by introducing the sub-weight $SW_k(I)$ into it:

$$TW(I)=\sqrt{\frac{SW_k(I)\times\sum_{p=1}^{m-1}\sum_{q=p+1}^{m}\left|SW_p(I)-SW_q(I)\right|}{\binom{m}{2}}} \tag{34}$$

The square root is used to avoid "polarization" of the values of *TW* (some are too large and others are too close to 0).

Using this function, $TW=\sqrt{9\times|9-1|/1}=8.5$ for sub-weight set (9, 1) and $TW=\sqrt{1\times|9-1|/1}=2.8$ for sub-weight set (1, 9). Hence, *TW* distinguishes the two cases from each other. Moreover, if an instance is labeled as belonging to a class, and is associated with a low sub-weight for the class, it will have a low total-weight.

Alternatively, we can introduce a method similar to standard deviation (note it is *not* a standard deviation) to compute the total-weight as

$$TW(I)=\sqrt{SW_k(I)\times\frac{1}{m}\sum_{p=1}^{m}(SW_p(I)-\overline{SW(I)})^2} \tag{35}$$

4.3 Weight Normalization

Now we have a single total-weight for each instance. However, these weights cannot be directly used for training instances to build a fuzzy SVM classifier, because the number of EPs may differ significantly from one class to another. As a result, the class with the largest number of EPs will have the highest aggregated value.

To overcome this problem, the total-weights of instances in a class are normalized by the value range of the class. Having a total-weight $TW(I)$ for training instance *i*, we need a normalization function that maps $TW(I)$ from $(-\infty, +\infty)$ to [0, 1]. Let $TW_{\max}$ and $TW_{\min}$ be the maximum and the minimum total-weights for a given class. We use the following mapping to get the normalized weights:

$$W(I)=\frac{TW(I)-TW_{\min}}{TW_{\max}-TW_{\min}} \tag{36}$$

where $TW(I)$ is the final weight for training instance *I*. We perform this mapping class by class, thereby obtaining a normalized final weight for each instance. Figure 2 depicts the architecture of our weight assignment scheme underlying the fuzzy SVM.

5 Experimental Evaluation

In order to evaluate the effectiveness of our EP-based weighting model, we carry out a number of experiments. We used 25 benchmark datasets from the UCI Machine Learning Repository [19]. Both polynomial and RBF kernels were used. We used the WEKA [17] discretization filter "weka.filters.supervised.attribute.Discretize -R first -last" for continuous attributes [25], and modified the WEKA SVM classifier to build our fuzzy SVM classifier.

Error estimates are obtained using stratified 10-fold Cross-Validation (CV-10). Results reported are the mean classification performance over the 10 folds. Here we use a range of values for hyper-parameters *C*, *d* and σ, and report the best classification accuracy for each classifier.

5.1 EP-based Fuzzy SVM vs. Standard SVM

In Table 1 we compare the accuracy of the standard SVM classifier to our fuzzy SVM classifier with EPs-based weights calculated by (34). The comparison is effected by calculating the percent improvement (*%impr*) in accuracy (*acc*) as

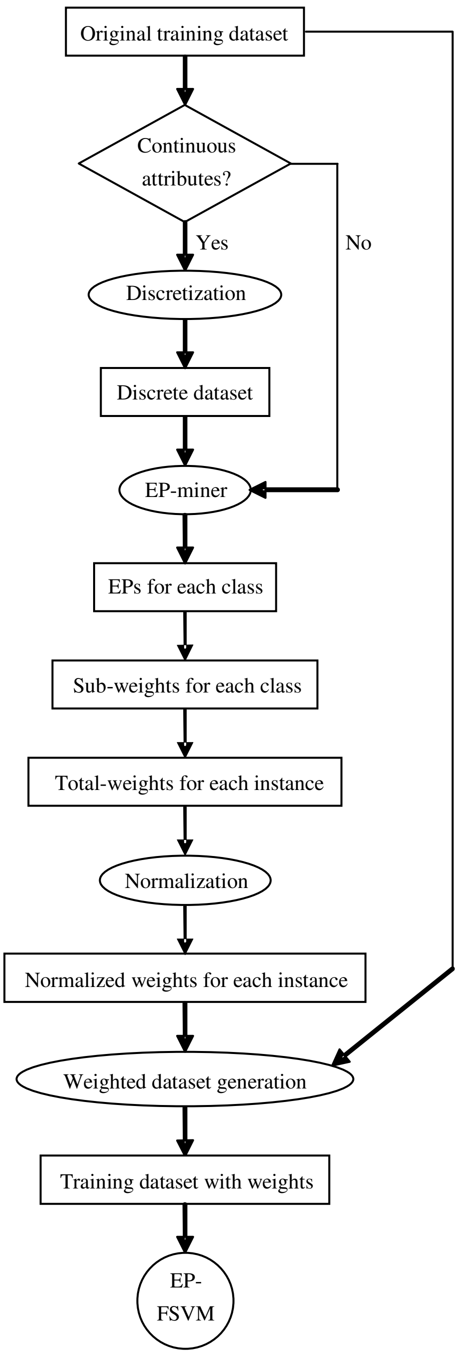

Original training dataset
Continuous attributes?
Yes
No
Discretization
Discrete dataset
EP-miner
EPs for each class
Sub-weights for each class
Total-weights for each instance
Normalization
Normalized weights for each instance
Weighted dataset generation
Training dataset with weights
EP-FSVM

Fig. 2: Procedure of building the EP-based fuzzy SVM.

$$\%impr = \frac{acc\_w\text{–}SVM - acc\_SVM}{acc\_SVM} \times 100 \qquad (37)$$

Equation (37) will yield a negative % when the SVM performs better than the fuzzy SVM; conversely, a positive % indicates superior performance by the fuzzy SVM. Table 1 shows that our EP-based method to assign weights results in a fuzzy SVM classifier which almost always outperforms the standard SVM classifier using either kernel. The maximum improvement, 22.03%, is realized for the dataset Breast-c, while the maximum -*%impr* is -4.17% for colic-o. Put another way, SVM is better than fuzzy SVM in only 9 out of 50 tries (18% of the tries). The parameter sets are shown in Table 2.

Dataset	**Polynomial Kernel**			**RBF Kernel**		
	SVM	FSVM	*%impr*	SVM	FSVM	*%impr*
Anneal-o	90.20	**98.89**	9.63	90.87	**98.23**	8.10
Autos	78.05	**81.51**	4.43	78.54	**81.85**	4.21
Balance-s	**99.36**	98.76	–0.60	98.72	**100**	1.30
Breast-c	69.63	**84.97**	22.03	**75.91**	75.52	–0.51
Breast-w	97.00	**99.42**	2.49	97.14	**99.08**	2.00
Colic	82.61	**90.07**	9.03	84.51	**85.05**	0.64
Colic-o	**78.26**	75.00	–4.17	78.53	**79.35**	1.04
Credit-a	85.51	**95.46**	11.64	86.96	**93.17**	7.14
Credit-g	75.70	**91.50**	20.87	78.30	**81.60**	4.21
Diabetes	77.86	**78.85**	1.27	77.73	**77.87**	0.18
Glass	74.77	**78.52**	5.02	72.90	**76.07**	4.35
Heart-c	84.49	**95.18**	12.65	86.14	**94.59**	9.81
Heart-h	84.75	**94.35**	11.33	84.05	**91.67**	9.07
Heart-s	84.44	**95.56**	13.17	84.44	**85.93**	1.76
Hepatitis	85.17	**93.96**	10.32	86.46	**94.43**	9.22
Ionosphere	**91.75**	91.45	–0.33	**94.89**	94.59	–0.32
Iris	**96.67**	**96.67**	0	**97.33**	96.00	–1.37
Kr-vs-kp	**99.66**	**99.66**	0	**99.72**	99.56	–0.16
Labor	**89.47**	87.72	–1.96	**92.98**	87.72	–5.66
Lymph	86.49	**87.84**	1.56	86.49	**87.84**	1.56
Mushroom	**100**	**100**	0	**100**	**100**	0
Sick	**96.66**	**96.66**	0	**97.03**	**97.03**	0
Sonar	84.62	**89.90**	6.24	**88.46**	87.50	–1.09
Vote	96.55	**97.24**	0.71	**96.55**	96.09	–0.48
Weather-n	**71.43**	**71.43**	0	71.43	**78.57**	10.00
Average	86.44	**90.82**	5.70	87.44	**89.57**	2.98
Best	4	**16**		7	**16**	

Wilcoxon Test (Win/Draw/Loss)		
	Polynomial Kernel	**RBF Kernel**
FSVM vs. SVM	13/11/1	10/14/1

Table 1: %Accuracy and %improvement; SVM vs. FSVM (eq.34)

We performed CV-10 Wilcoxon signed-rank tests (at the 0.05 level) and found that our fuzzy SVM classifier with polynomial kernels got 13 wins, 11 draws and 1 loss. And with RBF kernels, fuzzy SVM got 10 wins, 14 draws and 1 loss. By changing the constant *C* and CV foldnumber, we get similar results with all of the datasets.

When we use the weighting function in (35) to build the fuzzy SVM machine, the results (in Table 3) are also much better than those attained by the standard SVM. Comparing Tables 1 and 3, we see that the fuzzy SVM with weights calculated by (34) is somewhat more accurate than the fuzzy SVM machine based on (35). This is because (35) does not indicate the differences of sub-weights as well as (34). However, for some datasets with a large number of classes, (35) may be slightly more efficient than (34). Henceforth, we use only weighting scheme (34).

5.2 Robustness of the EP-based Fuzzy SVM Classifier

In the real world we must expect errors or noise in datasets. Therefore, robustness (or noise tolerance) is an important

Dataset	Polynomial Kernel				RBF Kernel			
	SVM		FSVM		SVM		FSVM	
	C	*d*	*C*	*d*	*C*	$2\sigma^2$	*C*	$2\sigma^2$
Anneal-o	5	2	10	3	1000	1	500	10
Autos	1	2	1	2	100	10	10	10
Balance-s	100	2	1000	2	1000	10	150	1
Breast-c	1	1	1	1	5	10	5	10
Breast-w	1	1	1	2	1	1	500	100
Colic	1	1	1	1	10	1000	500	1000
Colic-o	1	3	1	1	10	100	1000	1000
Credit-a	1	2	5	1	10	10	100	10
Credit-g	50	1	5	1	100	100	150	1000
Diabetes	100	1	10	1	1	1	10	1
Glass	500	3	50	2	500	1	1000	10
Heart-c	50	1	100	1	1000	100	1000	1000
Heart-h	10	1	10	1	50	10	1000	100
Heart-s	50	1	1	1	50	100	50	100
Hepatitis	1	1	1	1	100	100	150	1000
Ionosphere	10	2	1	2	10	10	10	1
Iris	5	1	500	3	100	100	5	1
Kr-vs-kp	1	3	1	3	100	10	100	10
Labor	1	2	10	1	5	10	500	10
Lymph	1	1	10	1	5	10	500	100
Mushroom	1	1	1	1	1	0.1	5	1
Sick	100	3	1000	1	150	1	1000	10
Sonar	1	3	5	1	5	1	10	1
Vote	5	1	10	1	5	10	500	100
Weather-n	10	1	1	2	5	1	5	10

Table 2: Parameter sets for models in table 1

feature of a classifier [16, 18]. Next, we investigate how well these EP-based classifiers respond to increasing noise in data, and compare the robustness of our fuzzy SVM classifier to that of the standard SVM classifier. To simulate the effect of noise, we replace the attribute values of all training instances as follows:

$$av' = av \times (1 + r \times \lambda) \tag{38}$$

where *av* and *av*' are the original and new attribute values; *r* is a random value in the range [–1, 1] and *λ* is the percentage of noise. We leave the testing data intact. Using these training datasets with noise, we build the standard SVM and fuzzy SVM classifiers and compare their performance.

Here, we use model difference to evaluate the noise tolerance of classifiers. The model difference between two models M_1 and M_2, $MD(M_1, M_2)$, is:

$$MD(M_1, M_2) = \frac{Ins\_d(M_1, M_2)}{Ins\_all} \times 100\% \tag{39}$$

where $Ins\_d(M_1, M_2)$ is the number of instances models M_1 and M_2 label differently, and *Ins_all* is the number of instances in the test set. The model difference $MD(M_1, M_2)$ between models trained by noisy data and noise-free data is one way to measure the robustness of a classifier. The results on four datasets are reported in Table 4. (We performed similar experiments on a large number of datasets and observed similar behavior.)

Dataset	Polynomial Kernel			RBF Kernel		
	SVM	FSVM	*%impr*	SVM	FSVM	*%impr*
Anneal-o	90.20	**93.32**	3.46	90.87	**92.65**	1.96
Autos	78.05	**79.02**	1.24	78.54	**80.00**	1.86
Balance-s	99.36	**99.68**	0.32	98.72	**100**	1.30
Breast-c	69.63	**81.12**	16.50	**75.91**	72.73	–4.19
Breast-w	97.00	**99.42**	2.49	97.14	**99.08**	2.00
Colic	82.61	**90.07**	9.03	84.51	**86.96**	2.90
Colic-o	**78.26**	75.00	–4.16	78.53	**79.35**	1.04
Credit-a	85.51	**96.52**	12.88	86.96	**94.35**	8.50
Credit-g	75.70	**87.20**	15.19	78.30	**79.80**	1.92
Diabetes	77.86	**78.85**	1.27	77.73	**77.87**	0.18
Glass	74.77	**76.07**	1.74	72.90	**76.07**	4.35
Heart-c	84.49	**95.18**	12.65	86.14	**94.59**	9.81
Heart-h	84.75	**95.24**	12.38	84.05	**92.86**	10.48
Heart-s	84.44	**93.33**	10.53	84.44	**85.19**	0.89
Hepatitis	85.17	**89.68**	5.30	86.46	**90.96**	5.20
Ionosphere	**91.75**	**91.75**	0	**94.89**	**94.89**	0
Iris	**96.67**	**96.67**	0	**97.33**	96.00	–1.37
Kr-vs-kp	**99.66**	**99.66**	0	**99.72**	99.56	–0.16
Labor	**89.47**	**89.47**	0	**92.98**	91.23	–1.88
Lymph	86.49	**87.16**	0.77	86.49	**87.16**	0.77
Mushroom	**100**	**100**	0	**100**	**100**	0
Sick	**96.66**	**96.66**	0	**97.03**	**97.03**	0
Sonar	84.62	**86.54**	2.27	88.46	**88.94**	0.54
Vote	96.55	**97.24**	0.71	**96.55**	96.09	–0.48
Weather-n	**71.43**	**71.43**	0	**71.43**	**71.43**	0
Average	86.44	**89.85**	3.94	87.44	**88.99**	1.77
Best	1	**17**		5	**16**	
Wilcoxon Test (Win/Draw/Loss)						
	Polynomial Kernel			**RBF Kernel**		
FSVM vs. SVM	10/14/1			7/17/1		

Table 3: %Accuracy and %improvement; SVM vs. FSVM (eq.35)

Noise Percentage (%)		0	10	20	30	40
Breast-w	SVM	0	11.59	14.16	18.03	20.03
	FSVM	0	8.44	13.02	15.74	17.02
Glass	SVM	0	9.35	17.29	21.50	26.64
	FSVM	0	4.67	9.81	13.55	16.36
Ionosphere	SVM	0	8.26	17.09	21.08	24.22
	FSVM	0	5.13	14.25	17.38	19.09
Iris	SVM	0	25.33	26.67	33.33	36.00
	FSVM	0	21.33	22.00	26.00	28.67

Table 4: %Model difference; SVM vs. FSVM with increasing noise

The graphs in Figure 3 show that the model differences of the fuzzy SVM classifier change less than those of the standard SVM classifier in the presence of noise. For example, on the "Glass" dataset, the model difference of our fuzzy SVM classifier increases from 0% to 16.36% when the noise level increases from 0% to 40%. Over the same range, the standard SVM classifier increases to

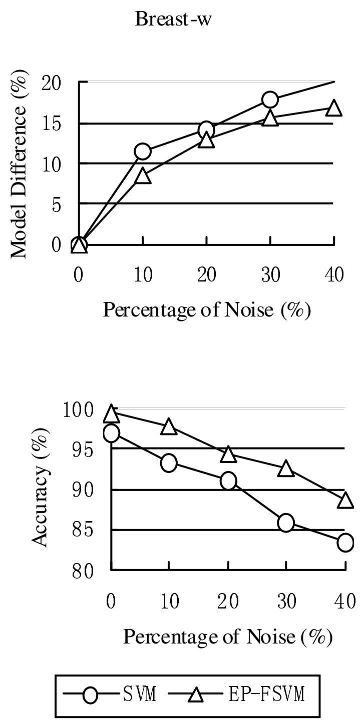


Fig. 3: Effect of increasing noise on model differences and accuracy for dataset "Breast-w".

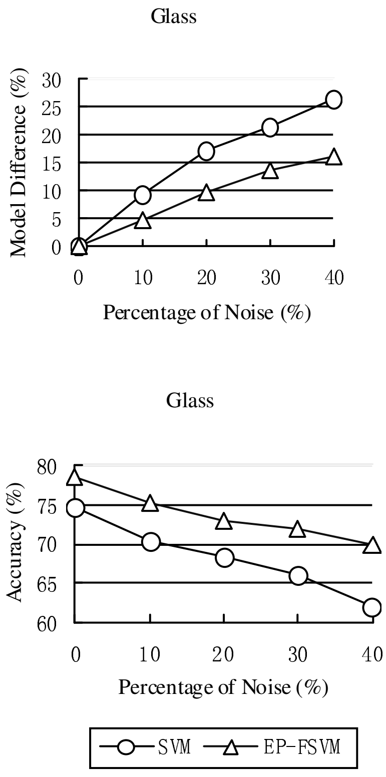


Fig. 4: Effect of increasing noise on model differences and accuracy for dataset "Glass".

26.64%, about 10% more than the fuzzy SVM design. Figures 3 and 4 show that our fuzzy SVM classifier is more tolerant to noise and more accurate than the standard SVM classifier for these data sets, and for many others that are not reported here.

Dataset	Polynomial Kernel			RBF Kernel		
	SVM	DC FSVM	EP FSVM	SVM	DC FSVM	EP FSVM
Balance-s	**99.36**	96.59	98.76	98.72	97.20	**100**
Breast-w	97.00	95.97	**99.42**	97.14	95.97	**99.08**
Colic-o	**78.26**	75.00	75.00	78.53	78.70	**79.35**
Diabetes	77.86	76.94	**78.85**	77.73	77.61	**77.87**
Glass	74.77	75.65	**78.52**	72.90	73.58	**76.07**
Heart-s	84.44	93.50	**95.56**	84.44	85.33	**85.93**
Ionosphere	**91.75**	**91.75**	91.45	**94.89**	94.40	94.59
Iris	96.67	**97.09**	96.67	**97.33**	96.00	96.00
Sonar	84.62	86.13	**89.90**	**88.46**	87.50	87.50
Average	87.19	87.62	**89.35**	87.79	87.37	**88.49**
Best	2	1	**5**	3	0	**6**
Wilcoxon Test (Win/Draw/Loss)						
	Polynomial Kernel			**RBF Kernel**		
EP FSVM vs.SVM	4/4/1			3/6/0		
EP FSVM vs. DC FSVM	5/4/0			4/5/0		

Table 5: %Accuracy; numeric datasets in table 1; SVM vs. (DC FSVM) vs. (EP FSVM);

6 Related Work

In this section we compare our EP based weighting scheme with the weighting methods in [4, 15].

6.1 Data-center-based Weighting Method

In [4], the following weighting function based on data centers (DC) is used.

$$W'(I) = \begin{cases} 1 - \dfrac{\|\boldsymbol{x}_+ - \boldsymbol{x}_I\|}{\max\|\boldsymbol{x}_+ - \boldsymbol{x}_I\| + \delta}, & if\ y_i = 1; \\ 1 - \dfrac{\|\boldsymbol{x}_- - \boldsymbol{x}_I\|}{\max\|\boldsymbol{x}_- - \boldsymbol{x}_I\| + \delta}, & if\ y_i = -1. \end{cases} \tag{40}$$

where $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ are the data centers of classes + and – respectively, and δ is a sufficiently small positive number used to avoid the case $W'(I)$=0. This weighting scheme is defined only for numeric feature vector data, whereas our weighting function (34) can handle datasets with either numeric or nominal attributes.

We calculate the weights based on data centers and use them to build a DC fuzzy SVM classifier. Table 5 compares this design to our fuzzy SVM classifier using EP-based weights (EP FSVM) for the numeric datasets in our study library.

Table 5 shows that our EP-based model is superior to the data-center-based scheme. The fuzzy SVM classifier combining EPs-based model got 5 wins, 4 draws, no losses (polynomial kernels), and 4 wins, 5 draws, no losses (RBF kernels). And it increases the accuracy significantly for most of the datasets. The best improvement is for dataset Glass, which enjoys improvements of 3.79% and 3.38%.

6.2 Other EP-based Weighting Methods

Alhammady and Ramamohanarao [15] introduced the following method for determining the sub-weights for decision trees:

$$SW_k'(I) = \frac{SW_k(I)}{MedianSW_k}; \tag{41}$$

$$SW_k''(I) = \frac{SW_k'(I)}{\sum_{p=1}^{m} SW_p'(I)}; \tag{42}$$

where $MedianSW_k$ is the median of the sub-weight values in class C_k. $SW_k(I)$ is the initial value of sub-weight of instance I for class C_k, and $SW_k''(I)$ is the final value. This function normalizes the sum of all sub-weights of each instance so that it is equal to 1. Using these sub-weights to build weighted decision trees has shown good results [15]. But if we use $SW_k''(I)$ as the final-weight of instance I to build the fuzzy SVM classifier, it poses a problem. For example, if there are two instances in class C_1 with sub-weights (1, 0) and (10, 0), then the total-weights of both instances are equal to 1. Consequently, these two instances will exert equal influence during training because they have the same total weight, even though the instances are quite different. Our method overcomes this problem.

In [14] the following weighting function is used by Fan and Ramamohanarao:

$$TW'(I) = SW_k(I) - \sum_{q \neq k} SW_q(I) \tag{43}$$

This function, the sub-weight of the instance's home class minus the sum of all other sub-weights, can handle some two-class problems. But for multi-class cases (43) also poses a problem. For example, if there are two instances in class C_1 with sub-weights (5, 2, 2, 1) and (5, 5, 0, 0), then the total-weights of both are equal to 0 ($5 - 2 - 2 - 1 = 0$ and $5 - 5 - 0 - 0 = 0$). Again, these two instances will exert equal influence during training because they have the same total weight, even though the instances are quite different. Indeed, they may be ignored as noise with such low total-weights. However, these two instances are quite different and should not be treated as the same, or as noise.

Using (34) with sub-weights (5, 2, 2, 1) and (5, 5, 0, 0), the total-weights become 3.16 and 4.06 respectively. Thus indicating the second set is more important than the first. Further experiments have convinced us that our weighting method is usually better than these EP-based weighting methods.

7 Conclusions

In this paper we have defined and developed a new, robust weighting method to build fuzzy SVM classifiers by means of emerging patterns, or EPs. We use EPs to discover class memberships for the training instances, and combine the class membership weights of each instance into a single weight which represents the instance's importance in building the classifier. Then we use the weighted training instances rather than the original crisp ones to construct the fuzzy SVM classifier.

Our EP-based fuzzy SVM classifier performs consistently better than the standard SVM classifier over a wide variety of datasets and data types. The accuracy of our weighting method is usually better than some other recently studied weighted designs. Typical improvements are in the range of 2%-10%. Finally, our experiments suggest that our method is more tolerant to noise and outliers than the standard SVM. Of course, no amount of empirical evidence supports sweeping generalizations about classifier performance, but we think our experiments are sufficiently broad that we can recommend this design with some confidence.

In the future we will explore our weighting method with other classifiers (e.g., C4.5 and decision trees), and furthermore, in the fields other than classification (e.g., regression and clustering).

8 Acknowledgment

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

9 References

[1] Vapnik, V.N. (1998): *Statistical Learning Theory*. New York, John Wiley.

[2] Burges, C.J.C. (1998): A tutorial on support vector machines for pattern Recognition. *Data Mining and Knowledge Discovery* **2**(2):121-167.

[3] Inoue, T., & Abe, S. (2001): Fuzzy support vector machines for pattern classification. *Proc. Int. Joint Conf. on Neural Networks*, Washington, **2**:1449-1454.

[4] Lin, C., & Wang, S. (2002): Fuzzy support vector machines. *IEEE Trans. Neural Networks*, **13**(2):464-471.

[5] Huang, H., & Liu, Y. (2002): Fuzzy support vector machines for pattern recognition and data mining. *Int. Journal of Fuzzy Systems*, **4**(3):826-835.

[6] Abe, S., & Inoue, T. (2002): Fuzzy support vector machines for multiclass problems. *10th European Symposium on Artificial Neural Networks*, Bruges, 113-118.

[7] Dong, G., & Li, J. (1999): Efficient mining of emerging patterns: discovering trends and differences. *Proc. 5th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, San Diego, 43-52.

[8] Dong, G., Zhang, X., Wong, L., & Li, J. (1999): CAEP: Classification by aggregating emerging patterns. *Proc. 2nd Int. Conf. Discovery Science*, Tokyo. 30-42.

[9] Li, J., Dong, G., & Ramamohanarao, K. (2001): Making use of the most expressive jumping emerging patterns for classification. *Knowledge Information Systems*, **3**(2):131-145.

[10] Zhang, X., Dong, G., & Ramamohanarao, K. (2000): Exploring constraints to efficiently mine emerging patterns from large high-dimensional data sets. *Proc. 6th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Boston, 310-314.

[11] Li, J., Ramamohanarao, K., & Dong, G. (2000): The space of jumping emerging patterns and its incremental maintenance algorithms, *Proc. 17th Int. Conf. on Machine Learning*, Standord, 551-558.

[12] Fan, H., & Ramamohanarao, K. (2003): Efficiently mining interesting emerging patterns. *Proc. 4th Int. Conf. on Web-Age Information Management*, Chengdu, 189-201.

[13] Ramamohanarao, K., & Bailey, J. (2003): Discovery of emerging patterns and their use in classification. *Australian Conf. on Artificial Intelligence*, Perth, 1-12.

[14] Fan, H., & Ramamohanarao, K. (2005): A weighting scheme based on emerging patterns for weighted support vector machines. *IEEE Int. Conf. on Granular Computing*, Beijing, **2**:435-440.

[15] Alhammady, H., & Ramamohanarao, K. (2006): Using emerging patterns to construct weighted decision trees. *IEEE Trans. Knowledge and Data Engineering*, **18**(7):865-876.

[16] Sun, Q., Zhang, X., & Ramamohanarao, K. (2003): Noise tolerance of EP-based classifiers, *Australian Conf. on Artificial Intelligence*, Perth, 796-806.

[17] Witten, I.H., & Frank, E. (1999): *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco, Morgan Kaufmann.

[18] Han, J., & Kamber, M. (2000): *Data Mining, Concepts and Techniques*. Burlington, Morgan Kaufmann Publishers.

[19] Newman, D.J., Hettich, S.C., Blake, L., & Merz, C.J. (1998): *UCI repository of machine learning databases*. http://www.ics.uci.edu/~mlearn/MLRepository.html

[20] Tsang, E.C.C., Yeung, D.S., & Chan, P.K. (2003): Fuzzy support vector machines for solving two-class problems. *Proc. 2nd Int. Conf. on Machine Learning and Cybernetics*, Xi'an, 1080-1083.

[21] Tsujinishi, D., & Abe, S. (2003): Fuzzy least squares support vector machines for multiclass problems. *Neural Networks*, **16**(1):785-792.

[22] Abe, S. (2004): Fuzzy LP-SVMs for multiclass problems. *12th European Symposium on Artificial Neural Networks*, Bruges, 429-434.

[23] Zhang, X. (1999): Using class-center vectors to build support vector machines. *Proc. IEEE Workshop on Neural Networks for Signal Processing IX*, Madison, 3-11.

[24] Cong, D., Wang, J., Yang W., & Zhang, S. (2005): Pattern Decomposition Algorithm Based on FP-tree. *Computer Engineering*, **31**(16):77-79, 88.

[25] Fayyad, U.M., & Irani, K.B. (1993): Multi-interval discretization of continuous-valued attributes for classification learning. *Proc. 13th Int. Joint Conf. on Artificial Intelligence*, Chambéry, 1022-1029.

[26] Cawley, G.C., & Talbot, N.L.C. (2001): Manipulation of prior probabilities in support vector classification. *Proc. Int. Joint Conf. on Neural Networks*, Washington, **4**:2433-2438.