Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition Letters 25 (2004) 1647–1656

Pattern Recognition Letters

www.elsevier.com/locate/patrec

Training algorithms for fuzzy support vector machines with noisy data

Chun-fu Lin, Sheng-de Wang [*]

Department of Electrical Engineering, National Taiwan University, EE Building, Taipei 106, Taiwan

Received 12 September 2003; received in revised form 9 December 2003
Available online 2 July 2004

Abstract

The previous study of fuzzy support vector machines (FSVMs) provides a method to classify data with noises or outliers by manually associating each data point with a fuzzy membership that can reflect their relative degrees as meaningful data. In this paper, we introduce two factors in training data points, the confident factor and the trashy factor, and automatically generate fuzzy memberships of training data points from a heuristic strategy by using these two factors and a mapping function. We investigate and compare two strategies in the experiments and the results show that the generalization error of FSVMs is comparable to other methods on benchmark datasets. The proposed approach for automatic setting of fuzzy memberships makes the FSVMs more applicable in reducing the effects of noises or outliers.

Keywords: Support vector machines; Fuzzy membership; Noise; Optimization and classification

1. Introduction

The theory of support vector machines (SVMs), which is based on the idea of structural risk minimization (SRM), is a new classification technique and has drawn much attention on this topic in recent years (Burges, 1998; Cortes and Vapnik, 1995; Vapnik, 1995, 1998). The good generalization ability of SVMs is achieved by finding a large margin between two classes (Bartlett and Shawe-Taylor, 1998; Shawe-Taylor and Bartlett, 1998). In many applications, the theory of SVMs has been shown to provide higher performance than traditional learning machines (Burges, 1998) and has been introduced as powerful tools for solving classification problems.

Since the optimal hyperplane obtained by the SVM depends on only a small part of the data points, it may become sensitive to noises or outliers in the training set (Boser et al., 1992; Zhang, 1999). To solve this problem, one approach is to do some preprocessing on training data to remove noises or outliers, and then use the remaining set

[*] Corresponding author. Tel.: +886-22-3635251; fax: +886-22-3671909.
E-mail addresses: genelin@hpc.ee.ntu.edu.tw (C.-f. Lin), sdwang@cc.ee.ntu.edu.tw (S.-d. Wang).

to learn the decision function (Cao et al., 2003). This method is hard to implement if we do not have enough knowledge about noises or outliers. In many real world applications, we are given a set of training data without knowledge about noises or outliers. There are some risks to remove the meaningful data points as noises or outliers.

There are many discussions in this topic and some of them show good performance. The theory of leave-one-out SVMs (Weston, 1999) (LOO-SVMs) is a modified version of SVMs. This approach differs from classical SVMs in that it is based on the maximization of the margin, but minimizes the expression given by the bound in an attempt to minimize the leave-one-out error. No free parameter makes this algorithm easy to use, but it lacks the flexibility of tuning the relative degree of outliers as meaningful data points. Its generalization, the theory of adaptive margin SVMs (AM-SVMs) (Weston and Herbrich, 2000), uses a parameter λ to adjust the margin for a given learning problem. It improves the flexibility of LOO-SVMs and shows better performance. The experiments in both of them show the robustness against outliers.

FSVMs solve this kind of problems by introducing the fuzzy memberships of data points. The main advantage of FSVMs is that we can associate a fuzzy membership to each data point such that different data points can have different effects in the learning of the separating hyperplane. We can treat the noises or outliers as less importance and let these points have lower fuzzy membership. It is also based on the maximization of the margin like the classical SVMs, but uses fuzzy memberships to prevent noisy data points from making narrower margin. This equips FSVMs with the ability to train data with noises or outliers by setting lower fuzzy memberships to the data points that are considered as noises or outliers with higher probability.

The previous work of FSVMs (Lin and Wang, 2002) did not address the issue of automatic setting of the fuzzy membership from the data points. We need to assume a noise model of the training data points, and then try and tune the fuzzy membership of each data point in the training. Without any knowledge of the distribution of data points, it is hard to associate the fuzzy membership to the data point.

In this paper, we design a noise model that introduces two factors in training data points, the confident factor and the trashy factor, and automatically generates fuzzy memberships of training data points from a heuristic strategy by using these two factors and a mapping function. This model is used to estimate the probability that the data point is considered as noisy information and use this probability to tune the fuzzy membership in FSVMs. This simplifies the use of FSVMs in the training of data points with noises or outliers. The experiments show that the generalization error of FSVMs is comparable to other methods on benchmark datasets.

The rest of this paper is organized as follows. A brief review of the architectures of SVMs and FSVMs will be described in Section 2. In Section 3, the detail training procedures and the noisy distribution model are explained. Two heuristic strategies are also introduced in this section. The experiments setup and results are presented in Section 4. Some concluding remarks are given in Section 5.

2. Fuzzy support vector machines

In this section, we describe SVMs and FSVMs for the classification problems, and show the difference between SVMs and FSVMs.

2.1. Architecture of SVMs

Suppose we are given a set S of labeled training points

$$(y_1, \boldsymbol{x}_1), \ldots, (y_l, \boldsymbol{x}_l). \tag{1}$$

Each training point $\boldsymbol{x}_i \in \mathscr{R}^N$ belongs to either of two classes and is given a label $y_i \in \{-1, 1\}$ for $i = 1, \ldots, l$. In most cases, the searching of a suitable hyperplane in an input space is too restrictive to be of practical use. A solution to this situation is mapping the input space into a higher dimension feature space and searching the optimal hyperplane in this feature space. Let $\boldsymbol{z} = \varphi(\boldsymbol{x})$ denote the corresponding feature space vector with a mapping φ from $\mathscr{R}^N$ to a feature space $\mathscr{Z}$. We wish to find the hyperplane

$$\boldsymbol{w} \cdot \boldsymbol{z} + b = 0, \tag{2}$$

defined by the pair $(\boldsymbol{w}, b)$, such that the inequality

$$y_i(\boldsymbol{w} \cdot \boldsymbol{z}_i + b) \geqslant 1 - \xi_i, \quad i = 1, \ldots, l \tag{3}$$

holds, where $\xi_i \geqslant 0$ is the slack variable for the data point that cannot be fitted in the optimal hyperplane.

The optimal hyperplane problem is then regraded as the solution to the problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{w} \cdot \boldsymbol{w} + C\sum_{i=1}^{l} \xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w} \cdot \boldsymbol{z}_i + b) \geqslant 1 - \xi_i, \quad i = 1, \ldots, l, \\ & \xi_i \geqslant 0, \quad i = 1, \ldots, l, \end{aligned} \tag{4}$$

where C is a constant. The parameter C can be regarded as a regularization parameter. This is the only free parameter in the SVM formulation. Tuning this parameter can make balance between margin maximization and classification violation. Detail discussions can be found in (Vapnik, 1998; Pontil and Verri, 1997).

Searching the optimal hyperplane in (4) is a quadratic programming (QP) problem, that can be solved by constructing a Lagrangian and transformed into the dual

$$\begin{aligned} \text{maximize} \quad & W(\alpha) = \sum_{i=1}^{l} \alpha_i \\ & \qquad - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} \alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^{l} y_i\alpha_i = 0, \\ & 0 \leqslant \alpha_i \leqslant C, \quad i = 1, \ldots, l, \end{aligned} \tag{5}$$

where $\alpha = (\alpha_1, \ldots, \alpha_l)$ is the vector of non-negative Lagrange multipliers associated with the constraints (3), and $K(\cdot,\cdot)$ is called *kernel* that can compute the dot product of the data points in feature space $\mathscr{Z}$, that is

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j) = \boldsymbol{z}_i \cdot \boldsymbol{z}_j. \tag{6}$$

The Kuhn–Tucker theorem plays an important role in the theory of SVM. According to this theorem, the solution $\bar{\alpha}_i$ of problem (5) satisfies the equality

$$\bar{\alpha}_i(y_i(\bar{\boldsymbol{w}} \cdot \boldsymbol{z}_i + \bar{b}) - 1 + \bar{\xi}_i) = 0, \quad i = 1, \ldots, l, \tag{7}$$

$$(C - \bar{\alpha}_i)\bar{\xi}_i = 0, \quad i = 1, \ldots, l. \tag{8}$$

From this equality it comes that the only non-zero values $\bar{\alpha}_i$ in Eq. (7) are those for that the constraints (3) are satisfied with the equality sign. The point $\boldsymbol{x}_i$ corresponding with $\bar{\alpha}_i > 0$ is called *support vector*.

To construct the optimal hyperplane $\bar{\boldsymbol{w}} \cdot \boldsymbol{z} + \bar{b}$, it follows that

$$\bar{\boldsymbol{w}} = \sum_{i=1}^{l} \bar{\alpha}_i y_i \boldsymbol{z}_i \tag{9}$$

and the scalar $\bar{b}$ can be determined from the Kuhn–Tucker conditions (7). Thus the optimal hyperplane can be reformulated as

$$f_{\mathrm{H}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{z} + b = \sum_{i=1}^{l} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b \tag{10}$$

and the decision function is

$$\begin{aligned} f_{\mathrm{D}}(\boldsymbol{x}) &= \operatorname{sign}(f_{\mathrm{H}}(\boldsymbol{x})) \\ &= \operatorname{sign}\left(\sum_{i=1}^{l} \alpha_i y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b\right). \end{aligned} \tag{11}$$

2.2. Architecture of FSVMs

In the previous study, we can easily check that each training data point is considered as the same before the optimizing process in SVMs. For the purpose of tuning the importance of training data points, we can assign a fuzzy membership to each data point. Suppose we are given a set S of labeled training points with associated fuzzy memberships

$$(y_1, \boldsymbol{x}_1, s_1), \ldots, (y_l, \boldsymbol{x}_l, s_l). \tag{12}$$

Each training point $\boldsymbol{x}_i \in \mathscr{R}^N$ is given a label $y_i \in \{-1, 1\}$ and a fuzzy membership $\sigma \leqslant s_i \leqslant 1$ with $i = 1, \ldots, l$, and sufficient small $\sigma > 0$, since the data point with $s_i = 0$ means nothing and can be just removed from training set without affecting the result of optimization.

Since the fuzzy membership s_i is the attitude of the corresponding point $\boldsymbol{x}_i$ toward one class and the parameter ξ_i can be viewed as a measure of error in the SVM, the term $s_i\xi_i$ is then a measure of error with different weighting. The optimal hyperplane problem is then regraded as the solution to

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w} + C\sum_{i=1}^{l} s_i\xi_i \\ \text{subject to} \quad & y_i(\boldsymbol{w}\cdot\boldsymbol{z}_i + b) \geqslant 1 - \xi_i, \quad i = 1,\dots,l, \\ & \xi_i \geqslant 0, \quad i = 1,\dots,l, \end{aligned} \tag{13}$$

where C is a constant. It is noted that a smaller s_i reduces the effect of the parameter ξ_i in problem (13) such that the corresponding point $\boldsymbol{x}_i$ is treated as less important.

The problem (13) can be transformed into

$$\begin{aligned} \text{maximize} \quad & W(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j K(\boldsymbol{x}_i,\boldsymbol{x}_j) \\ \text{subject to} \quad & \sum_{i=1}^{l} y_i\alpha_i = 0, \\ & 0 \leqslant \alpha_i \leqslant s_i C, \quad i = 1,\dots,l \end{aligned} \tag{14}$$

and the Kuhn–Tucker conditions are defined as

$$\bar{\alpha}_i(y_i(\bar{\boldsymbol{w}}\cdot\boldsymbol{z}_i + \bar{b}) - 1 + \bar{\xi}_i) = 0, \quad i = 1,\dots,l, \tag{15}$$

$$(s_i C - \bar{\alpha}_i)\bar{\xi}_i = 0, \quad i = 1,\dots,l. \tag{16}$$

The only free parameter C in SVMs controls the trade-off between the maximization of margin and the amount of misclassifications. A larger C makes the training of SVMs less misclassifications and narrower margin. The decrease of C makes SVMs ignore more training points and get a wider margin.

In FSVMs, we can set C to be a sufficient large value. It is the same as SVMs that the system will get narrower margin and allow less misclassifications if we set all $s_i = 1$. With different value of s_i, we can control the trade-off of the respective training point $\boldsymbol{x}_i$ in the system. A smaller value of s_i makes the corresponding point $\boldsymbol{x}_i$ less important in the training. There is only one free parameter in SVMs while the number of free parameters in FSVMs is equivalent to the number of training points.

3. Training algorithm for FSVMs

For efficient computation, the SVMs select the least absolute value to estimate the error function, that is $\sum_{i=1}^{l}\xi_i$, and use a regularization parameter C to balance between the minimization of the error function and the maximization of the margin of the optimal hyperplane. There are still some methods to estimate this error function. The LS-SVMs (Suykens and Vandewalle, 1999; Chua, 2003) select the least square value and show the differences in the constraints and optimization processes. In the situations that the underlying error probability distribution can be estimated, we can use the maximum likelihood method to estimate the error function. Let ξ_i be i.i.d. with probability density function $p_e(\xi)$, $p_e(\xi) = 0$ if $\xi < 0$. The optimal hyperplane problem (4) is then modified as the solution to the problem

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w} + C\sum_{i=1}^{l}\phi(\xi_i) \\ \text{subject to} \quad & y_i(\boldsymbol{w}\cdot\boldsymbol{z}_i + b) \geqslant 1 - \xi_i, \quad i = 1,\dots,l, \\ & \xi_i \geqslant 0, \quad i = 1,\dots,l, \end{aligned} \tag{17}$$

where $\phi(\xi) = -\ln p_e(\xi)$. Clearly, $\phi(\xi) \propto \xi$ when $p_e(\xi) \propto \mathrm{e}^{-\xi}$, that reduces the problem (17) to (4). Thus, with the different knowledge of the error probability model, a variety of error functions can be choosed and different optimal problems can be generated.

However, there are some critical issues in this kind of application. First, it is hard to implement the training program since solving the optimal hyperplane problem (17) is in general NP-complete (Cortes and Vapnik, 1995). Second, the error estimator ξ_i is related to the optimal hyperplane (10) by

$$\xi_i = \begin{cases} 0, & \text{if } f_{\mathrm{H}}(\boldsymbol{x}_i) \geqslant 1, \\ 1 - f_{\mathrm{H}}(\boldsymbol{x}_i), & \text{otherwise.} \end{cases} \tag{18}$$

Therefore, one needs to use the correct error model in the optimization process, but one needs to know the underlying function to estimate the error model. In practice it is impossible to estimate the error distribution reliably without a good estimation of the underlying function. This is so called a "catch 22" situation (Chen and Jain, 1994). Even though the probability density function $p_e(\xi)$ is unknown for almost all applications.

In contrast, in cases that it is given the noise distribution model of the data set. Let $p_x(\boldsymbol{x})$ be the probability density function of the data point $\boldsymbol{x}$ that is not a noise. For the data point $\boldsymbol{x}_i$ with higher value of $p_x(\boldsymbol{x}_i)$, which means that this data point has higher probability to be a real data, such that we wish to get lower value of ξ_i in the training process. To achieve this purpose, we can modify the error function as

$$\sum_{i=1}^{l} p_x(\boldsymbol{x}_i)\xi_i. \tag{19}$$

Hence, the optimal hyperplane problem (4) is then modified as the solution to the problem

$$\begin{aligned} &\text{minimize} \quad \frac{1}{2}\boldsymbol{w}\cdot\boldsymbol{w} + C\sum_{i=1}^{l} p_x(\boldsymbol{x}_i)\xi_i \\ &\text{subject to} \quad y_i(\boldsymbol{w}\cdot \boldsymbol{z}_i + b) \geqslant 1 - \xi_i, \quad i = 1,\dots,l, \\ &\qquad\qquad \xi_i \geqslant 0, \quad i = 1,\dots,l. \end{aligned} \tag{20}$$

When the probability density function $p_x(\boldsymbol{x})$ in problem (20) can be viewed as some kind of fuzzy membership, we can simply replace $s_i = p_x(\boldsymbol{x}_i)$ in problem (13), such that we can solve problem (20) by using the algorithm of FSVMs.

3.1. The noisy distribution model

There are many methods to training data using FSVMs, depending on how much information contains in the data set. If the data points are already associated with the fuzzy memberships, we can just use this information in training FSVMs. If it is given a noise distribution model of the data set, we can set the fuzzy membership as the probability of the data point that is not a noise, or as a function of it. In other words, let p_i be the probability of the data point $\boldsymbol{x}_i$ that is not a noise. If there exists this kind of information in the training data, we can just assign the value $s_i = p_i$ or $s_i = f_p(p_i)$ as the fuzzy membership of each data point, and use these information to get the optimal hyperplane in the training of FSVMs. Since almost all applications lack this information, we need some other methods to predict this probability.

Suppose we are given a heuristic function $h(\boldsymbol{x})$ that is highly relevant to the probability density function $p_x(\boldsymbol{x})$. For this assumption, we can build a relationship between the probability density function $p_x(\boldsymbol{x})$ and the heuristic function $h(\boldsymbol{x})$, that is defined as

$$p_x(\boldsymbol{x}) = \begin{cases} 1, & \text{if } h(\boldsymbol{x}_i) > h_{\mathrm{C}}, \\ \sigma, & \text{if } h(\boldsymbol{x}_i) < h_{\mathrm{T}}, \\ \sigma + (1-\sigma)\left(\dfrac{h(\boldsymbol{x}) - h_{\mathrm{T}}}{h_{\mathrm{C}} - h_{\mathrm{T}}}\right)^{d}, & \text{otherwise,} \end{cases} \tag{21}$$

where h_{C} is the confident factor and h_{T} is the trashy factor. These two factors control the mapping region between $p_x(\boldsymbol{x})$ and $h(\boldsymbol{x})$, and d is the parameter that controls the degree of mapping function as shown in Fig. 1.

The training points are divided into three regions by the confident factor h_{C} and trashy factor

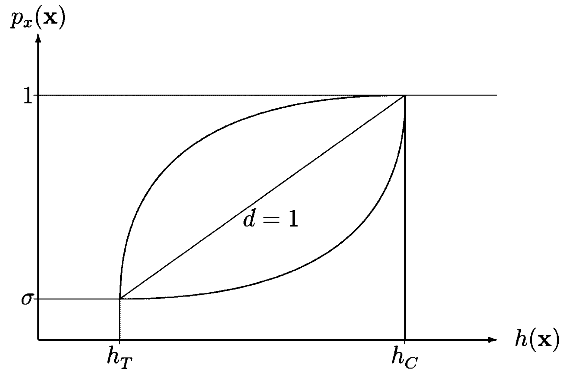

Fig. 1. The mapping between the probability density function $p_x(\boldsymbol{x})$ and the heuristic function $h(\boldsymbol{x})$.

h_T. If the data point, whose heuristic value $h(\boldsymbol{x})$ is bigger than the confident factor h_C, lies in the region of $h(\boldsymbol{x}) > h_C$, it can be viewed as valid examples with high confidence and the fuzzy membership is equal to 1. In contract, if the data point, whose heuristic value $h(\boldsymbol{x})$ is lower than the trashy factor h_T, lies in the region of $h(\boldsymbol{x}) < h_T$, it can be highly thought as noisy data and the fuzzy membership is assigned to lowest value σ. The data points in rest region are considered as noise with different probabilities and can make different distributions in the training process. There is no enough knowledge to choose proper function of this mapping. For simplicity, the polynomial function is selected in this mapping and the parameter d is used to control the degree of mapping.

3.2. The heuristic function

As for steps, discriminating between noises and data, we propose two strategies: one is based on kernel-target alignment and the other is using k-NN.

3.2.1. Strategy of using kernel-target alignment

The idea of kernel-target alignment is introduced in (Cristianini et al., 2002). Let $f_K(\boldsymbol{x}_i, y_i) = \sum_{j=1}^{l} y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. The kernel-target alignment is defined as

$$A_{\mathrm{KT}} = \frac{\sum_{i=1}^{l} f_K(\boldsymbol{x}_i, y_i)}{l\sqrt{\sum_{i,j=1}^{l} K^2(\boldsymbol{x}_i, \boldsymbol{x}_j)}}. \tag{22}$$

This definition provides a method for selecting kernel parameters and the experimental results show that adapting the kernel to improve alignment on the training data enhances the alignment on the test data, thus improved classification accuracy.

In order to discover some relation between the noise distribution and the data point, we simply focus on the value $f_K(\boldsymbol{x}_i, y_i)$. Suppose $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is a kind of distance measure between data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in feature space $\mathscr{F}$. For example, by using the RBF kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathrm{e}^{-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}$, the data points live on the surface of a hypersphere in

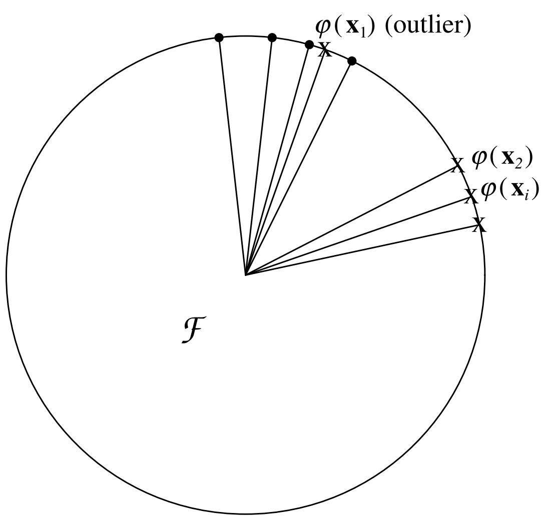

Fig. 2. The value $f_K(\boldsymbol{x}_1, y_1)$ is lower than $f_K(\boldsymbol{x}_2, y_2)$ in the RBF kernel.

feature space $\mathscr{F}$ as shown in Fig. 2. Then $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \varphi(\boldsymbol{x}_i) \cdot \varphi(\boldsymbol{x}_j)$ is the cosine of the angle between $\varphi(\boldsymbol{x}_i)$ and $\varphi(\boldsymbol{x}_j)$. For the outlier $\varphi(\boldsymbol{x}_1)$ and the representative $\varphi(\boldsymbol{x}_2)$, we have

$$\begin{aligned} f_K(\boldsymbol{x}_1, y_1) &= \sum_{y_i = y_1} K(\boldsymbol{x}_1, \boldsymbol{x}_i) - \sum_{y_i \neq y_1} K(\boldsymbol{x}_1, \boldsymbol{x}_i), \\ f_K(\boldsymbol{x}_2, y_2) &= \sum_{y_i = y_2} K(\boldsymbol{x}_2, \boldsymbol{x}_i) - \sum_{y_i \neq y_2} K(\boldsymbol{x}_2, \boldsymbol{x}_i). \end{aligned} \tag{23}$$

We can easily check the followings

$$\begin{aligned} \sum_{y_i = y_1} K(\boldsymbol{x}_1, \boldsymbol{x}_i) &< \sum_{y_i = y_2} K(\boldsymbol{x}_2, \boldsymbol{x}_i), \\ \sum_{y_i \neq y_1} K(\boldsymbol{x}_1, \boldsymbol{x}_i) &> \sum_{y_i \neq y_2} K(\boldsymbol{x}_2, \boldsymbol{x}_i), \end{aligned} \tag{24}$$

such that the value $f_K(\boldsymbol{x}_1, y_1)$ is lower than $f_K(\boldsymbol{x}_2, y_2)$.

We observe this situation and assume that the data point $\boldsymbol{x}_i$ with lower value of $f_K(\boldsymbol{x}_i, y_i)$ can be considered as outlier and should make less contribution of the classification accuracy. Hence, we can use the function $f_K(\boldsymbol{x}, y)$ as a heuristic function $h(\boldsymbol{x})$.

This heuristic function assumes that a data point will be considered as noise with high probability if this data point is more closer to the other class than its class. For a more theoretic discussion, let $D_{\pm}(\boldsymbol{x})$ be the mean distance between the data point $\boldsymbol{x}$ and data points $\boldsymbol{x}_i$ with $y_i = \pm 1$, which is defined as

$$D_\pm(\boldsymbol{x}) = \frac{1}{l_\pm} \sum_{y_i=\pm 1} \|\boldsymbol{x} - \boldsymbol{x}_i\|^2, \tag{25}$$

where $l_\pm$ is the number of data points with $y_i = \pm 1$, respectively. Then the value $y_k(D_+(\boldsymbol{x}_k) - D_-(\boldsymbol{x}_k))$ can be considered as an indication of noise. For the same case in feature space, $D_\pm(\boldsymbol{x})$ is reformulated as

$$\begin{aligned} D_\pm(\boldsymbol{x}) &= \frac{1}{l_\pm} \sum_{y_i=\pm 1} \|\varphi(\boldsymbol{x}) - \varphi(\boldsymbol{x}_i)\|^2 \\ &= \frac{1}{l_\pm} \sum_{y_i=\pm 1} (K(\boldsymbol{x},\boldsymbol{x}) - 2K(\boldsymbol{x},\boldsymbol{x}_i) + K(\boldsymbol{x}_i,\boldsymbol{x}_i)) \\ &= K(\boldsymbol{x},\boldsymbol{x}) + \frac{1}{l_\pm} \sum_{y_i=\pm 1} (K(\boldsymbol{x}_i,\boldsymbol{x}_i) - 2K(\boldsymbol{x},\boldsymbol{x}_i)). \end{aligned} \tag{26}$$

Assume that $l_+ \simeq l_-$, we can replace $l_\pm$ by $l/2$, and the value of $K(\boldsymbol{x},\boldsymbol{x})$ is 1 for the RBF kernel. Then

$$\begin{aligned} y_k(D_+(\boldsymbol{x}_k) - D_-(\boldsymbol{x}_k)) &= y_k\left(\frac{2}{l}\sum_{y_i=1}(1 - 2K(\boldsymbol{x}_k,\boldsymbol{x}_i)) \right. \\ &\qquad \left. - \frac{2}{l}\sum_{y_i=-1}(1 - 2K(\boldsymbol{x}_k,\boldsymbol{x}_i))\right) \\ &= \frac{4y_k}{l}\sum_i -y_i K(\boldsymbol{x}_k,\boldsymbol{x}_i) \\ &= -\frac{4}{l} f_K(\boldsymbol{x}_k, y_k), \end{aligned} \tag{27}$$

which is reduced to the heuristic function $f_K(\boldsymbol{x}_k, y_k)$.

3.2.2. Strategy of using k-NN

For each data point $\boldsymbol{x}_i$, we can find a set S_i^k that consists of k nearest neighbors of $\boldsymbol{x}_i$. Let n_i be the number of data points in the set S_i^k that the class label is the same as the class label of data point $\boldsymbol{x}_i$. It is reasonable to assume that the data point with lower value of n_i is more probable as noisy data. It is trivial to select the heuristic function $h(\boldsymbol{x}_i) = n_i$. But for the data points that are near the margin of two classes, the value n_i of these points may be lower. It will get poor performance if we set these data points with lower fuzzy memberships. In order to avoid this situation, the confident factor h_C, which controls the threshold of which data point needs to reduce its fuzzy membership, will be carefully chosen.

3.3. The overall procedure

There is no explicit way to solve the problem of choosing parameters for SVMs. The use of a gradient descent algorithm over the set of parameters by minimizing some estimates of the generalization error of SVMs is discussed in (Chapelle et al., 2002). On the other way, the exhaustive search is the popular method to choose the parameters, but it becomes intractable in this application as number of parameters is growing.

In order to select parameters in this kind of problem, we divide the training procedure into two main parts and propose the following procedures:

(1) Use the original algorithm of SVMs to get the optimal kernel parameters and the regularization parameter C.
(2) Fix the kernel parameters and the regularization parameter C, that are got in the previous procedure, and find the other parameters in FSVMs.
    (a) Define the heuristic function $h(\boldsymbol{x})$.
    (b) Use exhaustive search to choose the confident factor h_C, the trashy factor h_T, the mapping degree d, and the fuzzy membership lower bound σ.

4. Experiments

In these simulations, we use the RBF kernel as

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathrm{e}^{-\gamma\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}. \tag{28}$$

We conducted computer simulations of SVMs and FSVMs using the same data sets as in (Rätsch et al., 2001). Each data set is split into 100 sample sets of training and test sets. [1] For each sample set,

[1] These are available at http://ida.first.gmd.de/~raetsch/data/benchmarks.htm.

the test set is independent of training set. For each data set, we train and test the first 5 sample sets iteratively to find the parameters of the best average test error. Then we use these parameters to train and test the whole sample sets iteratively and get the average test error. Since there are more parameters than the original algorithm of SVMs, we use two procedures to find the parameters as described in the previous section. In the first procedure, we search the kernel parameters and C using the original algorithm of SVMs. In the second procedure, we fix the kernel parameters and C that are found in the first stage, and search the parameters of the fuzzy membership mapping function.

To find the parameters of strategy using kernel-target alignment, we first fix $h_C = \max_i f_K(\boldsymbol{x}_i, y_i)$ and $h_T = \min_i f_K(\boldsymbol{x}_i, y_i)$, and perform a two-dimensional search of parameters σ and d. The value of σ is chosen from 0.1 to 0.9 step by 0.1. For some case, we also compare the result of $\sigma = 0.01$. The value of d is chosen from 2^{-8} to 2^8 multiply by 2. Then, we fix σ and d, and perform a two-dimensional search of parameters h_C and h_T. The value of h_C is chosen such that 0%, 10%, 20%, 30%, 40%, and 50% of data points have the value of fuzzy membership as 1. The value of h_T is chosen such that 0%, 10%, 20%, 30%, 40%, and 50% of data points have the value of fuzzy membership as σ.

To find the parameters of strategy using k-NN, we just perform a two-dimensional search of parameters σ and k. We fix the value $h_C = k/2$, $h_T = 0$, and $d = 1$, since we do not find much gain or loss when we choose other values of these two parameters such that we skip searching for saving time. The value of σ is chosen from 0.1 to 0.9 stepped by 0.1. For some case, we also compare the result of $\sigma = 0.01$. The value of k is chosen from 2^1 to 2^8 multiplied by 2. Table 1 lists the parameters after our optimization in the simulations. For some data sets, we cannot find any parameters that can improve the performance of SVMs such that we left blank in this table.

Table 2 shows the results of our simulations. For comparison with SVMs, FSVMs with kernel-target alignment perform better in 9 data sets, and FSVMs with k-NN perform better in 5 data sets. By checking the average training error of SVMs in each data set, we find that FSVMs perform well in the data set when the average training error is high. These results show that our algorithm can improve the performance of SVMs when the data set contains noisy data.

We also list in Table 3 the other results for single RBF classifier (RBF), AdaBoost (AB), and regularized AdaBoost (AB_R), that are obtained from Rätsch et al. (2001), and the results for LOO-SVM, that are obtained from Weston and Herb-

Table 1
The parameters used in SVMs, FSVMs using strategy of kernel-target alignment (KT), and FSVMs using strategy of k-NN (k-NN) on 13 datasets

	SVMs		KT				k-NN	
	C	γ	σ	d	UB (%)	LB (%)	σ	k
Banana	316.2	1	0.01	64	10	0	0.1	32
B. Cancer	15.19	0.02	0.5	8	20	0	0.01	64
Diabetes	1	0.05	0.7	8	10	0	0.6	4
German	3.162	0.01818	0.6	8	20	30	0.8	4
Heart	3.162	0.00833	0.3	16	30	30	0.2	32
Image	500	0.03333	0.3	2^{-3}	10	0	–	–
Ringnorm	1e+9	0.1	–	–	–	–	–	–
F. Solar	1.023	0.03333	0.5	2^{-4}	20	0	0.3	256
Splice	1000	0.14286	–	–	–	–	–	–
Thyroid	10	0.33333	0.7	2^{-6}	0	0	–	–
Titanic	100,000	0.5	0.5	32	30	0	0.2	128
Twonorm	3.162	0.025	0.01	128	10	0	0.01	128
Waveform	1	0.05	0.01	2^{-8}	50	0	–	–

Table 2
The average training error of SVMs (TR), and the test error of SVMs, FSVMs using strategy of kernel-target alignment (KT), and FSVMs using strategy of *k*-NN (*k*-NN) on 13 datasets

	TR	SVMs	KT	*k*-NN
Banana	6.7	11.5 ± 0.7	**10.4 ± 0.5**	11.4 ± 0.6
B. Cancer	18.3	26.0 ± 4.7	25.3 ± 4.4	**25.2 ± 4.1**
Diabetes	19.4	23.5 ± 1.7	**23.3 ± 1.7**	23.5 ± 1.7
German	16.2	23.6 ± 2.1	**23.3 ± 2.3**	23.6 ± 2.1
Heart	12.8	16.0 ± 3.3	**15.2 ± 3.1**	15.5 ± 3.4
Image	1.3	3.0 ± 0.6	**2.9 ± 0.7**	–
Ringnorm	0.0	1.7 ± 0.1	–	–
F. Solar	32.6	**32.4 ± 1.8**	32.4 ± 1.8	32.4 ± 1.8
Splice	0.0	**10.9 ± 0.7**	–	–
Thyroid	0.4	4.8 ± 2.2	**4.7 ± 2.3**	–
Titanic	19.6	22.4 ± 1.0	**22.3 ± 0.9**	**22.3 ± 1.1**
Twonorm	0.4	3.0 ± 0.2	**2.4 ± 0.1**	2.9 ± 0.2
Waveform	3.5	**9.9 ± 0.4**	9.9 ± 0.4	–

Table 3
Comparison of test error of Single RBF classifier, AdaBoost (AB), regularized AdaBoost (AB_R), SVMs, LOO-SVMs (LOOS), FSVMs using strategy of kernel-target alignment (KT), and FSVMs using strategy of *k*-NN (*k*-NN) on 13 datasets

	RBF	AB	AB_R	SVMs	LOOS	KT	*k*-NN
Banana	10.8	12.3	10.9	11.5	10.6	**10.4**	11.4
B. Cancer	27.6	30.4	26.5	26.0	26.3	25.3	**25.2**
Diabetes	24.3	26.5	23.8	23.5	23.4	**23.3**	23.5
German	24.7	27.5	24.3	23.6	N/A	**23.3**	23.6
Heart	17.6	20.3	16.5	16.0	16.1	**15.2**	15.5
Image	3.3	**2.7**	**2.7**	3.0	N/A	2.9	–
Ringnorm	1.7	1.9	**1.6**	1.7	N/A	–	–
F. Solar	34.4	35.7	34.2	**32.4**	N/A	32.4	32.4
Splice	10.0	10.1	**9.5**	10.9	N/A	–	–
Thyroid	4.5	**4.4**	4.6	4.8	5.0	4.7	–
Titanic	23.3	22.6	22.6	22.4	22.7	**22.3**	**22.3**
Twonorm	2.9	3.0	2.7	3.0	N/A	**2.4**	2.9
Waveform	10.7	10.8	**9.8**	9.9	N/A	9.9	–

rich (2000). We can easily check that FSVMs perform better in the data set with noises.

5. Conclusions

In this paper, we propose training procedures for FSVMs, and describe two strategies for setting fuzzy membership in FSVMs. It makes FSVMs more feasible in the application of reducing the effects of noises or outliers. The experiments show that the performance is better in the applications with the noisy data.

We also compare the two strategies for setting the fuzzy membership in FSVMs. The usage of FSVMs with kernel-target alignment is more complicated since there exist many parameters. It costs more time to find the optimal parameters in the training process but the performance is better. FSVMs with *k*-NN is more simple to use and the results are close to the previous strategy. How to select a strategy for FSVMs is just like how to select a good kernel for SVMs. We still have not enough information to show that in which case a special strategy works.

The training time of FSVMs is more than the original SVMs due to the extra parameters. For the most cases, we do more effort to search the kernel parameters and C, which are needed for any SVMs, for the best result, but we spend less time to find the extra parameters of FSVMs in order to improve the previous result.

References

Bartlett, P., Shawe-Taylor, J., 1998. Generalization performance of support vector machines and other pattern classifiers. In: Schölkopf, B., Burges, C., Smola, A. (Eds.), Advances in Kernel Methods: Support Vector Learning. MIT Press, Cambridge, MA, pp. 43–54.

Boser, B.E., Guyon, I., Vapnik, V., 1992. A training algorithm for optimal margin classifiers. Comput. Learn. Theor., 144–152.

Burges, C.J.C., 1998. A tutorial on support vector machines for pattern recognition. Data Min. Knowl. Disc. 2 (2), 121–167.

Cao, L.J., Lee, H.P., Chong, W.K., 2003. Modified support vector novelty detector using training data with outliers. Pattern Recogn. Lett. 24, 2479–2487.

Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S., 2002. Choosing multiple parameters for support vector machines. Mach. Learn. 46 (1–3), 131–159.

Chen, D.S., Jain, R.C., 1994. A robust back propagation learning algorithm for function approximation. IEEE Trans. Neural Networks 5 (3), 467–479.

Chua, K.S., 2003. Efficient computations for large least square support vector machine classifiers. Pattern Recogn. Lett. 24, 75–80.

Cortes, C., Vapnik, V., 1995. Support vector networks. Mach. Learn. 20, 273–297.

Cristianini, N., Shawe-Taylor, J., Elisseeff, A., Kandola, J., 2002. On kernel-target alignment. In: Dietterich, T.G., Becker, S., Ghahramani, Z. (Eds.), Advances in Neural Information Processing Systems, vol. 14. MIT Press, pp. 367–373.

Lin, C.-F., Wang, S.-D., 2002. Fuzzy support vector machines. IEEE Trans. Neural Networks 13 (2), 464–471.

Pontil, M., Verri, A., 1997. Properties of support vector machines. Neural Comput. 10, 955–974.

Rätsch, G., Onoda, T., Müller, K.-R., 2001. Soft margins for AdaBoost. Mach. Learn. 42 (3), 287–320.

Shawe-Taylor, J., Bartlett, P.L., 1998. Structural risk minimization over data-dependent hierarchies. IEEE Trans. Inform. Theor. 44 (5), 1926–1940.

Suykens, J.A.K., Vandewalle, J., 1999. Least squares support vector machine classifiers. Neural Process. Lett. 9 (3), 293–300.

Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer, New York.

Vapnik, V., 1998. Statistical Learning Theory. Wiley, New York.

Weston, J., 1999. Leave-one-out support vector machines. In: Dean, T. (Ed.), Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99. Morgan Kaufmann, pp. 727–733.

Weston, J., Herbrich, R., 2000. Adaptive margin support vector machines. In: Smola, A., Bartlett, P., Scholkopf, B., Schuurmans, D. (Eds.), Advances in Large Margin Classifiers. MIT Press, Cambridge, MA, pp. 281–295.

Zhang, X., 1999. Using class-center vectors to build support vector machines. Neural Networks Signal Process., 3–11.