Neural Process Lett (2009) 29:7–27
DOI 10.1007/s11063-008-9092-y

Stray Example Sheltering by Loss Regularized SVM and *k*NN Preprocessor

Chan-Yun Yang · Che-Chang Hsu · Jr-Syu Yang

Published online: 30 November 2008

Abstract This paper presents a new model developed by merging a non-parametric *k*-nearest-neighbor (*k*NN) preprocessor into an underlying support vector machine (SVM) to provide shelters for meaningful training examples, especially for stray examples scattered around their counterpart examples with different class labels. Motivated by the method of adding heavier penalty to the stray example to attain a stricter loss function for optimization, the model acts to shelter stray examples. The model consists of a filtering *k*NN emphasizer stage and a classical classification stage. First, the filtering *k*NN emphasizer stage was employed to collect information from the training examples and to produce arbitrary weights for stray examples. Then, an underlying SVM with parameterized real-valued class labels was employed to carry those weights, representing various emphasized levels of the examples, in the classification. The emphasized weights given as heavier penalties changed the regularization in the quadratic programming of the SVM, and brought the resultant decision function into a higher training accuracy. The novel idea of real-valued class labels for conveying the emphasized weights provides an effective way to pursue the solution of the classification inspired by the additional information. The adoption of the *k*NN preprocessor as a filtering stage is effective since it is independent of SVM in the classification stage. Due to its property of estimating density locally, the *k*NN method has the advantage of distinguishing stray examples from regular examples by merely considering their circumstances in the input space. In this paper, detailed experimental results and a simulated application are

C.-Y. Yang (✉)
Department of Mechanical Engineering, Technology and Science Institute of Northern Taiwan,
No. 2 Xue-Yuan Rd., Beitou, Taipei, 112 Taiwan, ROC
e-mail: cyyang.research@gmail.com

C.-C. Hsu · J.-S. Yang
Department of Mechanical and Electro-Mechanical Engineering, Tamkang University,
No. 151 Ying-Chuan Rd., Tamsui, Taipei County, 25137 Taiwan, ROC

C.-C. Hsu
e-mail: 692342792@s92.tku.edu.tw

J.-S. Yang
e-mail: 096034@mail.tku.edu.tw

given to address the corresponding properties. The results show that the model is promising in terms of its original expectations.

Keywords k-nearest-neighbor preprocessor · Stray training examples · Support vector machines · Classification · Pattern recognition

1 Introduction

As one of the categories of powerful learning machines, support vector machines (hereafter SVMs) are gaining popularity due to their superior performance. Based on statistical learning theory, the mathematics of SVM was firmly grounded by Vapnik [1,2]. The basic concept of this theory for SVMs seeks to design a learning hypothesis for an optimal function that is obtained through the minimization of generalization risk. Considering a classification problem, a set of statistical hypotheses regularized by the relevant parameters is generated to minimize the expected risk over all available training examples. But in general, the expected risk cannot easily be found with unknown probability densities. An approximation thus is usually adopted by replacing the expected risk with an empirical risk [1–3].

$$R_{\text{emp}} = \frac{1}{n} \sum_{\mathbf{x}_i \in S} L(y_i, f(\mathbf{x}_i)), \quad i = 1, 2, \ldots, n \tag{1}$$

where $L(.)$ denotes a loss function designed to evaluate the associated errors in the training examples. The empirical risk, empirically measured from the loss function, has the advantage that it can be easily and readily computed using only the available training examples. Since the problems with regard to the SVMs can actually refer to the related convex optimization learning problems, the governing loss function measuring the associated errors of training examples plays a key role in the learning.

In a basic two-class classification problem, a set of training examples $S = \{(\mathbf{x}_i, y_i)\}$, $i = 1, 2, \ldots, n$, is given. In this expression, $\mathbf{x}_i$ is described as an input pattern in the d dimensional input space, $\mathbf{x}_i \in R^d$. A class label y_i is confirmed as a response of $\mathbf{x}_i$ from either of the two classes, and is hence assigned with a value in the set of $\{-1, +1\}$. In real world applications, only a small part of training examples might drift far from the normal region of their familiar majority in the input space R^d. These few examples usually do not evidence less confidence than the crowds of familiar majority, despite the fact that they often stray beyond the normal region. Hence, we call them stray examples. Considering the two partly overlapping classes, the stray examples might be scattered in the area of their counterpart examples with different class labels. The examples mixes in with the adversaries might occasionally be misclassified by entanglement with their neighbors; however they are crucial instances in the training set S. From the perspective of the loss function [2,3], these stray examples that may be liable to misclassification may be saved by increasing their corresponding losses in the optimization procedure. This paper proposes a model merging a non-parametric kNN estimation [4–8] into an underlying SVM to produce an instance-dependent loss function. This model breaks the equivalent attitude of examples in the training set S, and tries to give the examples various levels of weight by their significance. The kNN method, mining locally for useful information among the training examples, provides a special independent aspect to evaluate the significance of the examples. According to this evaluation, penalties from the instance-dependent loss function are determined for the optimization procedure. Various penalties in the optimization procedure will produce a set of new Lagrangian multipliers and form a separating hyperplane different from the one generated by the set of original multipliers.

With the present method, modification of SVMs with parameterized class labels are used to convey the emphasized weights are employed [9]. The use of such parameterized class labels provides a solution to connect both the *k*NN rule and the underlying SVM. From the perspective of the loss function, the parameterized SVM produces a particular surrogate of the loss function to transfer those various penalties to the stray examples. The surrogate still fulfills the criterion of increasing penalization for those examples tending towards misclassification, but the degree of penalization depends on the degree that the stray examples have immersed in the adversary class.

2 Merging *k*NN with SVM

2.1 Loss Functions in SVMs

According to the fundamentals of SVMs, examples with a positive margin are known as those classified correctly and examples with a negative margin are those misclassified. According to this definition, the goal of the learning is to produce a positive margin as frequently as possible. Under this criterion, a formal definition of the loss function is incurred by the triplet consisting of an example $\mathbf{x}_i$, the class label y_i, and the predicted value coming from the resultant decision function $f(\mathbf{x}_i)$. Here, the soft margin loss function, which is popularly used in the classical SVM, is defined as [3,10]:

$$c(\mathbf{x}_i, y_i, f(\mathbf{x}_i)) = \max(0, 1 - y_i f(\mathbf{x}_i)) = \begin{cases} 0, & \text{if } y_i f(\mathbf{x}_i) \geq 1, \\ 1 - y_i \cdot f(\mathbf{x}_i), & \text{otherwise.} \end{cases} \tag{2}$$

where y_i is a binary target,$y_i \in \{+1, -1\}$, and $f(\mathbf{x}_i)$ is a real-valued prediction from the decision function. In the expression, the scale of loss depends on the product $y_i f(\mathbf{x}_i)$ if the product has a value less than one. The loss function will be minimized in the process of fitting the model to meet the goal of being “classified correctly as frequently as possible.” In other words, the loss function represents a selected measure of the discrepancy between the target y_i and the predicted value, which is the response by the fitted function $f(\mathbf{x}_i)$.

The loss function is commonly employed as a penalization to penalize an example with negative margin more heavily than one with positive margin in the SVMs. Following this statement, any penalty that is incurred by the loss function is not necessary for the examples which are correctly classified with large enough positive margins. For instance, the classical SVM takes the penalties focused on the examples with margins less than one. In a sense, it seems feasible to slightly change the scale of penalties for the examples with small or negative margins under the penalization rule in Eq. 2. Hence, several surrogates of loss function, such as the misclassification, exponential, binomial deviance, squared error, and support vector loss functions have been proposed for selected topics in the theory of statistical learning [11,12] (Fig. 1). Except for the misclassification loss, all the surrogates are strict convex functions. However, their common essential property is to continuously penalize examples that have a small or negative margin. The differences among the surrogates are the degrees of the penalization exerted on the examples with negative margins. For example, the penalties for large and increasingly negative margins associated with the binomial deviance loss function increase linearly, and those with the exponential loss function increase exponentially. Having a role in the regularization of the hypothesis, the loss function is very important and has received much attention in the related literature. Many researchers have stressed that the

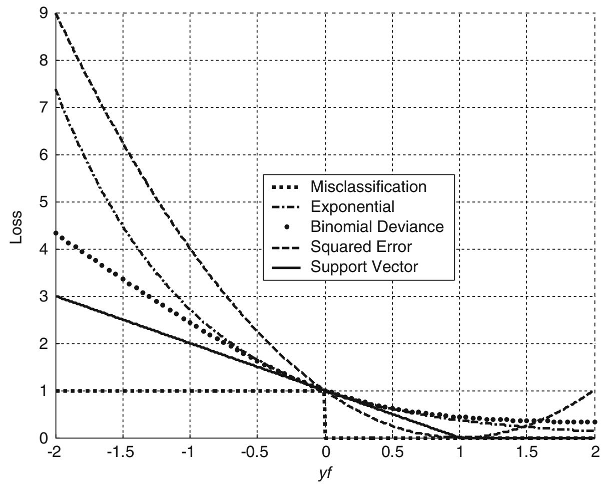


Fig. 1 Surrogates of loss function in statistical learning [11]

performance assessment of a hypothesis can be related to a minimization problem regarding the loss function [12–15].

2.2 A Preprocessor to Emphasize Local Heterogeneities

A preprocessor based on the *k*NN method is employed to identify and re-weight the stray examples. The *k*NN methods have been widely used in the field of data mining for applications such as density estimation and classification [4–8]. Based on the non-parametric assumption, these methods can be described as a class of instance-based learning techniques that learn directly from a set of available examples and interpret results statistically. Instead of trying to create rules, the *k*NN approaches work directly from the examples themselves. Suppose an unlabeled example $\mathbf{x}$ is placed among the n training examples in a hyperspace of volume V. From the Bayes rule, the posteriori probability of $P(\omega_j|\mathbf{x})$ can be expressed as:

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x},\ \omega_j)}{p(\mathbf{x})}, \tag{3}$$

where ω_j denotes the jth class. With the prior probability $p(\mathbf{x}) = \sum_h p(\mathbf{x}, \omega_h)$ and a locally approximated joint density function $p(\mathbf{x}, \omega_j) = k_j/nV$, the posteriori probability of $P(\omega_j|\mathbf{x})$ can be obtained by:

$$P(\omega_j|\mathbf{x}) = \frac{k_j}{k}, \tag{4}$$

where k denotes the number prototypes captured in the volume V, and k_j denotes the portion of k prototypes from the class ω_j. From Eq. 4, one can estimate $P(\omega_j|\mathbf{x})$ by the fraction of neighboring prototypes labeled as ω_j that are captured in a finite local region. The equation describes the class belonging to an unlabeled query example can be determined locally by the k nearest neighboring prototypes based on the basis of the similarity in the input space around the query example. The determination of k usually depends on the locality that the query examples should refer to. In general, the coverage of volume V is decreased with a decreasing value of k. Low locality is referred to estimate the probability if a small value of k is used.

Objectively speaking, we should be mindful of the opportunity of some examples if they are critical in the prototypes, especially if they are the ethnic minorities. In general, the ethnic minorities in the prototypes have less opportunity to receive attention in the decision making process. To rectify this, a modification to increase the hiring cost of the minorities in a fair way has been proposed [16]. This modification uses heavier weights for the minorities by considering the class size in an unbalanced dataset. An equal-valued compensation is given for all the examples in a certain minority-class, regardless of the difficult circumstances of individual examples. These examples, referring to stray examples, are used to indicate the examples which are difficult to correctly classify, whatever the asserted distribution is. In general, stray examples may be distant from the gathering area of the examples with the same class label, or may be close to the border of an adjacent overlapping region in which examples from different classes reside together. However, the non-parametric method, which identifies the uneven circumstances of the prototypes locally, is quite suitable for use as a preprocessor for filtering the individual stray examples. Instead of treating the training set from the global perspective by most learning algorithms , the *k*NN approaches locally mine the information in the training data. With the *k*NN, it does not matter fundamentally what particular distribution form or global probability density of the training data is assumed. Therefore, the *k*NN is for certain a universal candidate to improve a classification involving stray examples.

2.3 Support Vector Machines Carrying Emphasized Weights

According to the principle of structural risk minimization in statistical learning theory [1,2, 10], the SVMs were devised to find a decision boundary with maximal margin. Based on the decision boundary, good generalization ability can be achieved. By learning from the set of training examples $S = \{(x_i, y_i)\}$, the decision function, given as $f(\mathbf{x}) = \text{sign}\,(\langle \mathbf{w}, \mathbf{x}\rangle + b)$, can be determined by the orientation vector w and bias b. With the notation y_i in the input training set, altered parameterized labels $\tilde{y}_i$ tied to the training set have been introduced to carry the emphasized weights [17]. Because of this alteration, the training set is changed as:

$$\tilde{S} = \{(\mathbf{x}_i,\ \tilde{y}_i\ )\},\quad i = 1,\ 2,\ \ldots,\ n. \tag{5}$$

The real-valued class label $\tilde{y}_i$ having its sign identical to y_i, carries potential weights of example i. With the weight, the set $\tilde{S}$ tries to carry more information for training, even if $\tilde{S}$ contains a set of the same patterns $\mathbf{x}_i$ in the set S. The change in $\tilde{S}$ provides the chance to assign different weights for the stray examples. In Eq. 5, labels $\tilde{y}_i$'s are no longer a discrete value; instead it would be a real, $|\tilde{y}_i| \geq 1$, to carry the weight that we want to suggest for the stray examples. Therefore, the value of $\tilde{y}_i$ can be obtained by incorporating the idea of *k*NN

$$\tilde{y}_i = \eta \frac{y_i}{P(\omega_i|\mathbf{x}_i)}, \tag{6}$$

where $P(\omega_i|\mathbf{x}_i)$ is the posteriori probability denoted in Eq. 4.

The method of Eq. 6, called the *k*NN emphasizer, adopts an inverted scheme to scale up the value of $\tilde{y}_i$. The key point of the expression in Eq. 6 can be revealed from the ratio of magnification $1/P(\omega_i|\mathbf{x}_i)$. To fulfill the intention of carrying heavier weights in the stray examples, the value of the ratio should definitely be greater than 1. The acceleration parameter η as a scaling factor should also be a positive real number greater than 1 to ensure that $|\tilde{y}_i| \geq 1$. Hence, for different classes, $\tilde{y}_i$ should be:

$$\tilde{y}_i \geq 1, \quad \text{for } y_i = +1, \text{ and} \tag{7}$$

$$\tilde{y}_i \leq -1, \quad \text{for } y_i = -1. \tag{8}$$

As stressed above, the class labels $\tilde{y}_i$'s, carrying heavier weights, provide corresponding stray examples stricter penalties in the optimization, and are able to conduct the training more accurately. Following the steps in the classical SVM [10], a set of canonical constraints is set up with $\tilde{y}_i$ for optimization:

$$\langle \tilde{\mathbf{w}}, \mathbf{x}_i \rangle + \tilde{b} \geq +1 - \tilde{\xi}_i, \quad \text{for } \tilde{y}_i \geq +1, \text{ and} \tag{9}$$

$$\langle \tilde{\mathbf{w}}, \mathbf{x}_i \rangle + \tilde{b} \leq -1 + \tilde{\xi}_i, \quad \text{for } \tilde{y}_i \leq -1, \tag{10}$$

where $\tilde{\xi}$ denotes the slack variables equivalent to that in the classical model of soft margin SVM [10] for solving a linear non-separable problem. The inequalities in Eqs. 9 and 10 can be merged as

$$\tilde{y}_i \left(\langle \tilde{\mathbf{w}}, \mathbf{x}_i \rangle + \tilde{b} \right) \geq 1 - \tilde{\xi}_i, i = 1, 2, \ldots, n. \tag{11}$$

The constraints in Eq. 11 are used to specify the searching space of the maximal solution of margin $\rho = 2/\|\tilde{\mathbf{w}}\|$. Instead of using the inner-product $< \cdot, \cdot >$ to produce a linear classification, a kernel trick is then employed to extend the SVM to the case of non-linear classification [18]. At this point, a mapping function, $\phi(\cdot) : R^d \rightarrow R^{d_h}$, is introduced first to map features from the lower dimensional input space R^d to a higher dimensional feature space R^{d_h} and the objective and subjective functions for a general non-linear classification problem can be established, equivalent to those in the classical SVM [10]:

$$\min_{\tilde{\mathbf{w}}}, \tilde{b}, \tilde{\mathbf{w}} L_P(\tilde{\mathbf{w}}) = \frac{1}{2} \|\tilde{\mathbf{w}}\|_2^2 + \tilde{C} \sum_{i=1}^{n} \tilde{\xi}_i, \tag{12}$$

subject to

$$\tilde{y}_i \left(\langle \varphi(\tilde{\mathbf{w}}), \varphi(\mathbf{x}_i) \rangle + \tilde{b} \right) \geq 1\text{-}\tilde{\xi}_i, \text{ and} \tag{13}$$

$$\tilde{\xi}_i \geq 0, i = 1, 2, \ldots, n, \tag{14}$$

where $\tilde{C}$ is the penalty factor, which controls the trade-off between the classification capability and the margin width that the learning machine can achieve. With the derivations according to the Wolfe dual form [3], the minimization problem (12–15) can be written as:

$$\max_{\tilde{\boldsymbol{\alpha}}} L_D(\tilde{\boldsymbol{\alpha}}) = -\frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \tilde{y}_i \tilde{y}_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \tilde{\alpha}_i \tilde{\alpha}_j + \sum_{i=1}^{n} \tilde{\alpha}_i, \tag{15}$$

subject to

$$\sum_{i=1}^{n} \tilde{\alpha}_i \tilde{y}_i = 0, \quad \text{and} \tag{16}$$

$$0 \leq \tilde{\alpha}_i \leq \tilde{C}, \quad i = 1, 2, \ldots, n, \tag{17}$$

where $\tilde{\alpha}_i$ denotes the Lagrange multiplier corresponding to example $\mathbf{x}_i$ in the dual space. The expression in Eq. 15 employs the kernel function $\kappa(\mathbf{x}_i, \mathbf{x}_j)$ [18] as an alternative to compute similarities in the pair-wise examples for non-linear classification. The kernel function, computing implicitly the dot-product without mapping into a high-dimensional feature space R^{d_h},

$$\kappa(\mathbf{x}_i,\ \mathbf{x}_j) =< \varphi(\mathbf{x}_i),\ \varphi(\mathbf{x}_j) >, \quad i, j = 1,\ 2,\ \ldots,\ n, \tag{18}$$

is actually an excellent solution to manipulate the high dimensional mapping $\phi(\cdot)$. Here, it is noted that the kernel trick is only applied in the second stage of SVM. Eventually, a set of optimized $\tilde{\alpha}_i$ can be obtained after the quadratic programming of Eqs. 15–17. Only those $\tilde{\alpha}_i$'s corresponding to support vectors (*SVs*) will have non-zero values, $\tilde{\alpha}_i \neq 0$. According to the Karush-Kuhn-Tucker (KKT) conditions, the sparse expansion of *SVs* is sufficient to compute $\tilde{\mathbf{w}}$:

$$\tilde{\mathbf{w}} = \sum_{i \in \mathrm{SV}s} \tilde{\alpha}_i \tilde{y}_i \varphi(\mathbf{x}_i). \tag{19}$$

Furthermore, those examples with unsaturated non-zero Largrange multipliers, $0 < \tilde{\alpha}_i < \tilde{C}$, are selected to determine the bias $\tilde{b}$ with the KKT's complementarily conditions [3]. Once the optimal value of parameters $\tilde{\mathbf{w}}$ and $\tilde{b}$ are determined, the task of learning from the training data is eventually finished, and the decision function can therefore be given as:

$$\tilde{f}(\mathbf{x}) = \mathrm{sign}\left(\sum_{i \in SVs} \tilde{\alpha}_i \tilde{y}_i \kappa(\mathbf{x}, \mathbf{x}_i) + \tilde{b} \right). \tag{20}$$

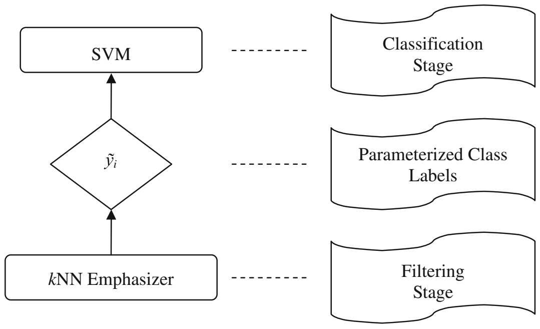
2.4 A Two-Stage Model

As in the illustrations, the loss function always focuses on the examples with small or negative margin, and tries to penalize these examples heavily. Moreover, we should note the weighting individual penalties by the parameterized class labels. Accordingly, a filtering stage of the *k*NN emphasizer will be inserted in front of the classification stage. A two-stage model of *k*NN-SVM (Fig. 2) is proposed in order to match the heavy penalization criterion. In this model, the *k*NN emphasizer sifts through the training set to choose all the examples that may be stray, and creates a set of various emphasized weights corresponding to the stray examples. In the second classification stage, the training examples with parameterized class label $\tilde{y}_i$, organize a temporal input set of $\tilde{S}$, including those refilled by the emphasized weights. Following the model in the previous section, a new set of Lagrangian multipliers $\tilde{\alpha}_i$ will be produced to form a new hyperplane. With the refilled emphasized weights for the stray examples, the induced hyperplane tends towards high training accuracy. Being a two-stage model, completely independent approaches adopted by *k*NN-SVM in both the stages does really make sense. The independence potentially reduces the possibility of over-penalization which may be caused by employing the identical approach twice, since the approach is often inclined to behave in a particular way if it has been employed again, and tends to focus on the same crew of training examples with over-weighted penalization.

2.5 Change of Loss Through Merging

In the present method, an SVM with parameterized class labels to convey the emphasized weights is employed. The use of such parameterized class labels provides a solution to connect both the *k*NN emphasizer and the underlying SVM. We should hereby point out the crucial effects on the loss function which accompany the parameterized class labels. Due to the change of $\tilde{y}_i$, all the examples are re-evaluated and are given arbitrary penalties individually.

Fig. 2 Model of *k*NN-SVM



Compared to the loss function (Eq. 2) of classical SVM, the penalties are no longer linearly increased with the margin $y_i f(\mathbf{x}_i)$ only. The circumstances of the local neighborhood should additionally be considered. The loss function here can be expressed as

$$\tilde{c}(\mathbf{x}_i, \tilde{y}_i, \tilde{f}(\mathbf{x}_i)) = \max(0, 1 - \tilde{y}_i \tilde{f}(\mathbf{x}_i)) = \begin{cases} 0, & \text{if } \tilde{y}_i \tilde{f}(\mathbf{x}_i) \geq 1, \\ 1 - \tilde{y}_i \tilde{f}(\mathbf{x}_i), & \text{otherwise.} \end{cases} \tag{21}$$

The change of loss function still satisfies the criterion of increasing the penalization for those examples tending towards misclassification in the training, but the penalties become higher if stray examples occur. In general, only a small portion of the training examples attained a higher penalty. The criterion alters the loss of individual stray examples, but it does not change the aggregative decision boundary very much, and instead it produces some standalone sheltered regions for the stray examples. This is because the potential of these stray examples is intensified locally by the *k*NN emphasizer and forces the convex optimizer to produce such standalone sheltered regions.

3 Experimental Results and Discussions

3.1 General Descriptions for Experiments

This section illustrates the basic characteristics of the *k*NN-SVM and compares its behaviors with its prototype stump of the classical SVM. In the experiments, an implementation according to Breiman's example [19] was adopted mainly to generate an artificial dataset. In order to clearly explain the effects on the *k*NN-SVM, only two classes of such normal distributed examples are taken on the two-dimensional input space, and form the simplified version of the dataset *Twonorm*. In the dataset, each class was drawn from a multivariate normal distribution with unit standard deviation. One class center is located at $\left(1/\sqrt{5}, 1/\sqrt{5}\right)$ and the other at $\left(-1/\sqrt{5}, -1/\sqrt{5}\right)$ [20]. In addition, several biomedical datasets were collected for performance tests, including *Bupa* [20], *Diabetes* [20], and *Biomed* [21]. According to the previous section, the *k*NN emphasizer Eq. 6 is used to evaluate locally the emphasizing weights of examples with their neighbors. One should be aware of a large value of η that will lead to fast saturation or even excess in the value of $\tilde{y}_i$ with a growing value of k, especially for stray examples in a complicated dataset. This may seriously suppress the

Table 1 Details of datasets

Dataset	Sample size			Dimension of features	K	Preprocess
	Class A	Class B	Total			
Twonorm	60	60	120	2	10	–
Bupa	145	200	345	6	5	Normalization
Diabetes	268	500	768	8	4	Normalization
Biomed	66	126	192	4	6	Normalization

sensitivity of the emphasizing weights that the examples in the neighboring region ought to have. Table 1 shows the details of all the datasets uses in this paper. In addition, an assessment of generalization performance with K-fold cross-validations is also described. A capitalized "K" is used here to make a difference with the special term of "k" for the kNN. The assessment of the generalization ability is an important issue for learning methods.

3.2 Changes of Decision Boundary

A separating hyperplane of dataset *Twonorm* depicted by the kNN-SVM is shown in Fig. 3b, and a comparison with that is made in Fig. 3a using the classical SVM. In this example, both the penalty factors, $\tilde{C}$ in the kNN-SVM and C in the classical SVM, are set to 1×10^2. The other relevant parameters setting the kNN-SVM are given as $k = 19$, $\eta = 4$ and $\tilde{\sigma} = \sqrt{d}$ in RBF kernel. Comparing to Fig. 3a, a more rugged hyperplane in Fig. 3b is obtained by the kNN-SVM. Instead of the smaller margin, the hyperplane obtained by the kNN-SVM creates sheltered sub-regions to make all the examples with same class label fall on the same side of the decision boundary. In Fig. 3b, the kNN-SVM urges the decision boundary to produce the sub-regions with various piecewise shapes, such as breaking into standalone islands or sticking out as peninsulas that are almost surrounded by the adversaries. This situation can be explained as follows. A crucial example settles, unfortunately, with its nearest k neighbors carrying class labels with an opposite value, and has been recognized as a stray example. As the essential goal of maintaining the membership of such an example in its native class, the kNN-SVM increases $\tilde{y}_i$ to set apart the stray example from the adversarial class. This means that protecting the stray example cannot be abandoned easily. In Fig. 3, the counts of the misclassified examples—those marked as solid symbols—are reduced from 19 to 16 due to the moderate change of the hyperplane. According to the rugged hyperplane, the related evidence of margin is reduced from 10.03×10^{-2} to 6.56×10^{-2}. As mentioned above, the margin reduction is a cost to trade off with the misclassification reduction in the training phase. The misclassifications, in our assumption, may consist of some stray examples which are immersed in the adversaries.

Similar tests were extended to the other datasets to validate the classifier's capability more widely (Table 2). The results show that the proposed kNN-SVM consistently has lower training error and higher generalization error than the underlying SVM does. Here, the higher generalization error is assessed directly by the narrower margin. Compared to an optimized prototype SVM with a set of equivalent parameter settings, the kNN-SVM with the additional emphasis of the stray examples may thus cause the system to be slightly over-fitted. The overfitting, as it happens, is related to the employment of the kNN-SVM, so users should consider this and carefully deal with it.

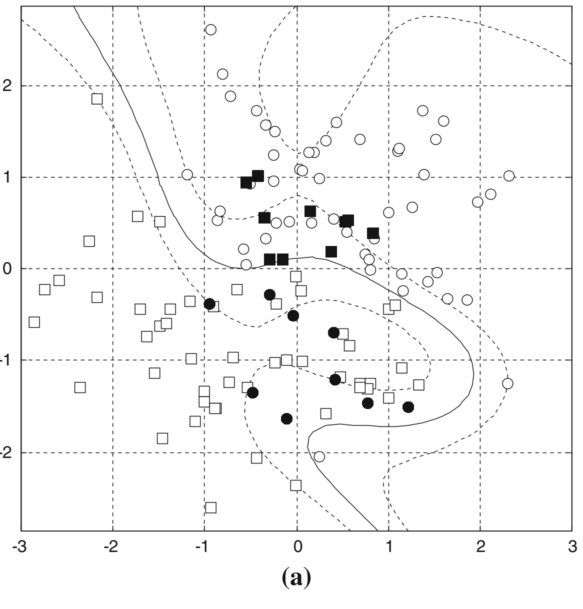
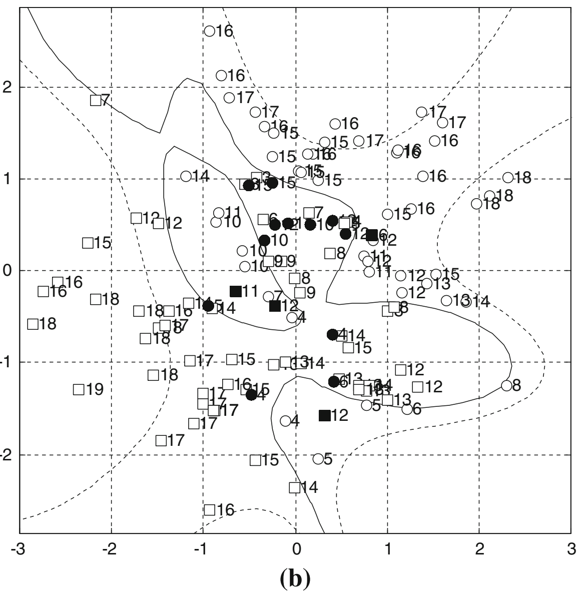



Fig. 3 A comparison of hyperplanes from both (**a**) classical SVM and (**b**) *k*NN-SVM with equivalent parameter settings, code words of k_j is placed beside the examples

3.3 Effects of Relevant Parameters

Upon examining the reduction along the increasing value of acceleration factor η, both the margin and training error reductions are found to be common effects. Diagrams of both reductions are given in Fig. 4. Even though both the equivalent settings of $\tilde{C} = C$ are changed from 10^2 to 10^1, the margin and classification errors of the *k*NN-SVM are generally lower than those of the classical SVM if a sufficient value of η is adopted. The general conditions can

Table 2 Extended validations on the datasets

Dataset	σ	Parameter setting			Misclassification counts		Margin ($\times 10^{-2}$)	
		$C = \tilde{C}$	k	η	*k*NN-SVM	Classical SVM	*k*NN-SVM	Classical SVM
Twonorm	$\sqrt{2}$	1×10^2	19	4	16	19	6.56	10.03
Bupa	$\sqrt{6}$	1×10^2	13	4	73	83	3.97	4.14
Diabetes	$\sqrt{8}$	1×10^3	23	6	112	144	0.68	1.22
Biomed	$\sqrt{4}$	1×10^1	25	4	16	18	17.06	21.71

be concluded as followings. Together with the lower classification error in Fig. 4b, a margin reduction can consistently be found in Fig. 4a as a sacrifice to trade off for classification accuracy in the training phase. The reduction of classification error is slightly increased with a higher value of η, which means that more stray examples are correctly classified in the training phase when the penalties from Eq. 6 are heavier. In terms of generalization performance, value of η shall not be too large to avoid an unexpected overfitting. In fact, a stray example with a huge value of η will dominate the other non-stray examples, and consume too much of the counter-balanced penalty in configuring the optimal separating hyperplane. This unexpected overfitting, as can be observed in the figure if η approaches a larger value, would degrade the generalization performance of the hypothesis (Fig. 4c). The details of this degradation will be discussed in the following section. However, a small but sufficient value for η to provide a smooth enough hypothesis is important in the model. Compared with the parameter-optimized classical SVM, a value slightly small than the equivalent C is suggested for $\tilde{C}$ in order to compensate for the increased penalty from the *k*NN emphasizer (Fig. 4c).

The examination is extended to the verification of parameter k in the filtering stage (Fig. 5). One should be convinced again that the diagram meets the assumption in the previous sections. The training error from the *k*NN-SVM is generally lower than the classical SVM, even if we set k to a large value. During the training stage, we can see that the scale of misclassification reduction depends not only on η but also on k. As illustrated in Fig. 5, the maximal reduction of training error will generally be satisfactory with a relatively small value of k in the beginning. The reduction is similar to the behavior of the class of *k*NN classifier. According to the behavior, the training error is approximated as an increasing function of k because of the inverse correlation between the neighboring number k and the model complexity [11,13]. Hence, a setting of k as large as possible is suggested if the model complexity is concerned. In fact, the setting of a larger value of k refers to more reference examples in the neighboring region, and urges the posteriori density $P(\omega_j|\mathbf{x})$ in Eq. 4 to behave more like a global estimation. This also avoids the condition of high model complexity caused by an overemphasis of the locality in the input space, since high model complexity in general leads to degradation in the generalization performance. In spite of the consideration of model complexity, it is noted that an excessively large value of k would unfortunately suppress the sheltering effect. The setting of k actually depends on the tradeoff between the model complexity and the sheltering effect.

3.4 Classification Improvement in the Neighborhood of Stray Examples

As mentioned above, the *k*NN emphasizer is able to resist the inattention to stray examples. This resistance protects the stray examples from neglect by an un-weighted hypothesis if they are highly important in the training set. If test examples appear close to the stray

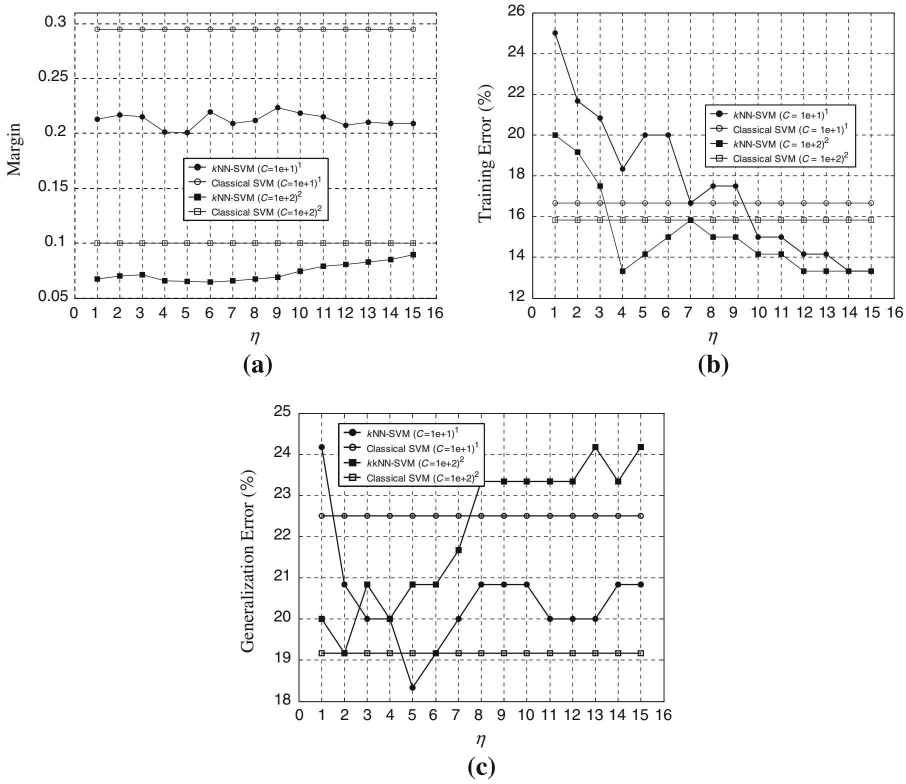


Fig. 4 (**a**) Margin, (**b**) training errors, and (**c**) generalization errors with increased settings of acceleration factor η. The superscript "1" dedicates to the equivalent condition of $C = \tilde{C} = 1 \times 10^1$, and the superscript "2" dedicates to $C = \tilde{C} = 1 \times 10^2$. Here, k is chosen as 19

examples with an identical class label, they will be protected by the sheltering effect. In order to examine this effect, an experiment was designed and tested on the dataset *Twonorm*. This experiment is designed to examine the classification improvement if some validation examples are aggregated close to the emphasized stray examples. If the emphasized stray examples have been given heavier weights, those validation examples may gain much confidence due to their closeness to the emphasized stray examples.

The procedures to generate the validation examples are described as follows. First, the emphasized stray examples were selected from the other prototypes with their $\tilde{y}_i$ greater than the average of $\sum \tilde{y}_i/n$. In contrast to the other prototypes, marked as small symbols "□" and "∘," for two classes, 31 of the scattered prototypes indicated by the large symbols are selected in Fig. 6a as the mother emphasized stray examples, and these are marked with the larger symbols "□" and "∘," according to the two classes. Second, the validation examples were then generated around those emphasized stray examples with an underlying parametric distribution. For the illustrative instance, we chose five examples randomly drawn from a multivariate distribution with their mean lay on the location of a certain mother emphasized stray example. We collected all the five examples of every mother emphasized stray example, and formed the full set of 155 validation examples. All the validation examples were assigned the same class labels as their mother emphasized stray examples. The full validation set is

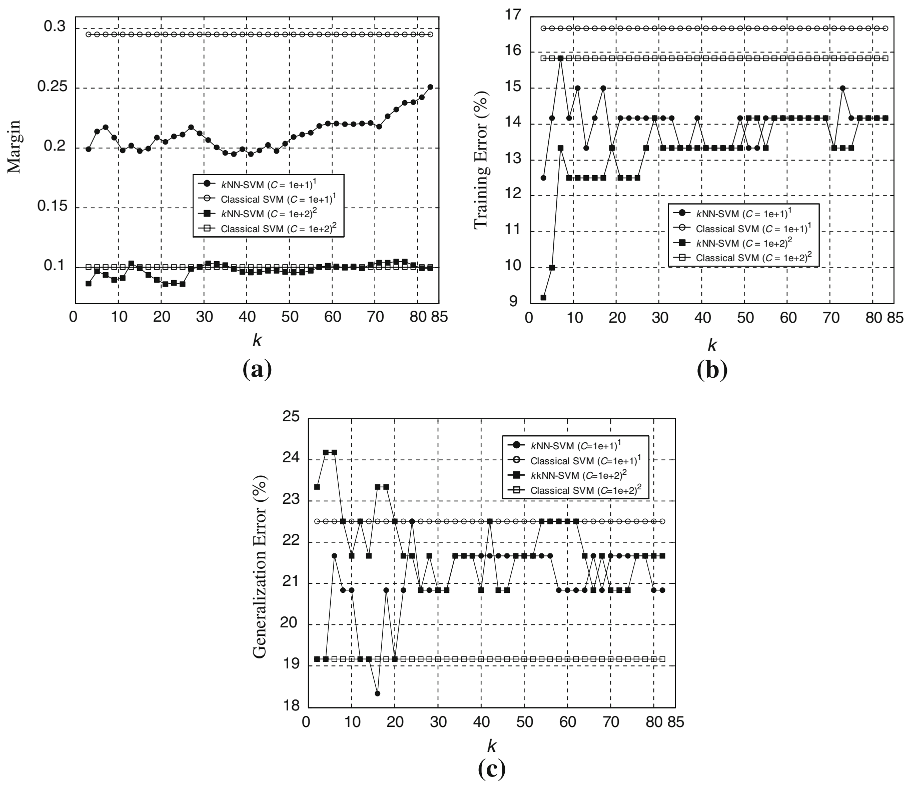


Fig. 5 (**a**) Margin, (**b**) training errors, and (**c**) generalization errors with increased setting of k. The superscript "1" dedicates to the equivalent condition of $C = \tilde{C} = 1 \times 10^1$, and the superscript "2" dedicates to $C = \tilde{C} = 1 \times 10^2$. Here, η is chosen as 14

shown in Fig. 6b with the points marked with "□" and "◯" symbols. Here, the standard deviation of the multivariate normal distribution is 0.5 for the illustrative instance. More validation sets with different settings of standard deviation are presented below.

As described previously, the model a *k*NN emphasizer, referring to a local density estimator, provides protection for the stray examples. If those stray examples are substantial in the learning set of a real application, we believe that a large number of examples with an identical class label will replicate anywhere with feature values that are near to the stray examples. Based on the assumption, the sub-regions created by the emphasized stray examples will make sense according to the type of classifications. The degree of improvement can be determined by counting the misclassifications that are marked as solid symbols in Fig. 7. Compared to the 102 misclassifications produced by the classical SVM (Fig. 7a), the rugged separating hyperplane of the *k*NN-SVM (Fig. 7b) produces only 81 misclassifications. It is obvious that the hyperplane with local sub-regions is capable of improving the classification with examples that behave like the emphasized stray examples.

Following the illustrative instance above, we changed the standard deviation to different values, and repeated the experiment several times. For each standard deviation setting, 5 batches of the 155 validation examples are generated repeatedly to test the sheltering effect. The results were then averaged, and they are listed on Table 3. In the test, various standard

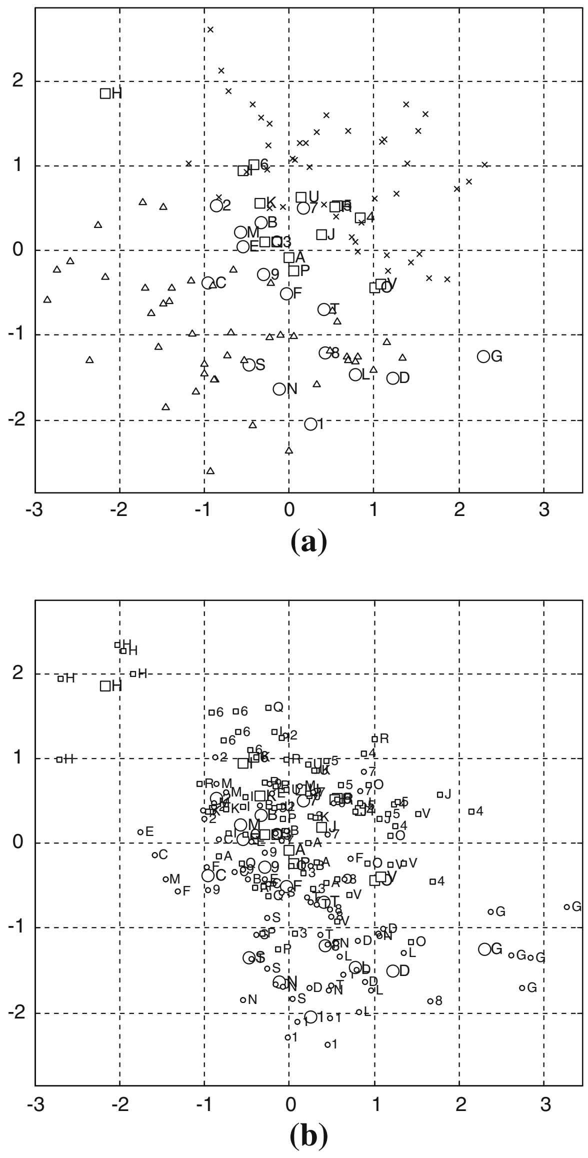


Fig. 6 The randomly generated validation examples: (**a**) Emphasized stray examples selected from the prototypes, and (**b**) Generated validation examples with their mother emphasized stray examples

deviations were used to control the scattering range of the validation examples. The smaller the standard deviation, the narrower is the scattering range. The table shows that the misclassification counts from the validation sets with small values of standard deviation are consistently lower than those with large values. All the sets with various standard deviations generally render an improved classification.

To show the effect to general real-world applications, some experiments have been carried out with the datasets listed in Table 1. By following the same procedure as described previously, the experiments are aimed to seek a set of parameter settings which achieves the

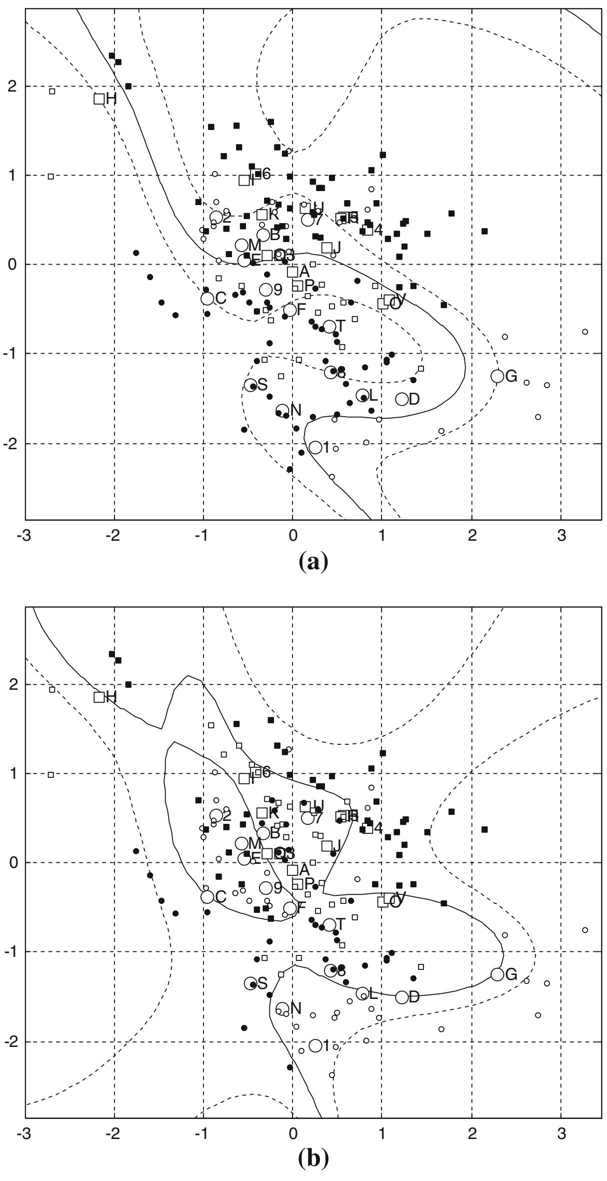


Fig. 7 Comparison of the classification of full validation set: (**a**) Classified by the classical SVM, and (**b**) Classified by the *k*NN-SVM. The misclassifications are represented by the solid symbols

lowest classification error rates via a grid-search scheme. With the best parameter settings, a comparison of the lowest classification error rates is concluded in Table 4. In addition to the comparison between the results of the *k*NN-SVM and those of the classical SVM, the results of cascaded SVM–SVM approach are also included in the table. The cascaded SVM–SVM approach adopts an underlying SVM as the preprocessor, instead of the *k*NN preprocessor, and presumes an underfitted setting $\sigma_p = \tilde{\sigma}$ and $C_p = 0.1\tilde{C}$ to collect sufficient stray examples for the preprocessor. In SVM–SVM, the class labels of the stray examples are then emphasized by:

Table 3 Classification improvement in the validation sets with various standard deviation

Misclassification counts	Repetition	Standard deviation				
		.1	.3	.5	.7	.9
Classical SVM	1	93	110	113	112	114
	2	96	106	102	106	109
	3	96	97	106	115	105
	4	97	105	103	110	103
	5	97	103	116	96	110
	Average	95.8	104.2	108	107.8	108.2
*k*NN-SVM	1	49	73	97	101	97
	2	49	73	87	98	100
	3	48	74	91	86	92
	4	45	78	93	96	100
	5	53	89	115	96	102
	Average	48.8	77.4	96.6	95.4	98.2

$$\tilde{y}_i = \begin{cases} y_i, & y_i f(\mathbf{x}_i) > 0 \\ y_i(1 - \eta' y_i f(\mathbf{x}_i)), & y_i f(\mathbf{x}_i) \leq 0, \end{cases} \tag{22}$$

where η' is also an acceleration parameter. Although the SVM–SVM may form a similar hybrid model, its sheltering effect is different. The *k*NN is more advantageous to the preprocessor to capture the stray examples locally. It is really different from the global perspective achieved by the SVM preprocessor. From Table 4, a better sheltering effect of the *k*NN-SVM is obtained with rough grids searching, compared with those of the best configured SVM-SVM and classical SVM. The results confirm that the sheltering effect on the stray examples is something to which particular attention should be paid.

3.5 Generalization Performance via K-Fold Cross Validation

The generalization performance is an important issue to qualify a learning hypothesis. In general, the expected prediction error over a sufficient independent validation sets is not easy to obtain. A K-fold or leave-one-out cross validation using only one dataset is often taken to assess the generalization performance. In this study, the method of K-fold cross-validation is adopted as an evaluation facility for the generalization performance.

The assessment of generalization performance with varying exponential grids of C or $\tilde{C}$ by the K-fold validation is depicted in Fig. 8a, where K is given as 10. In the diagram, the generalization errors of the *k*NN-SVM with different settings of $\tilde{C}$ are larger than those of the classical SVM with equivalent settings of C. This consequence indicates the overfitting tendency of the *k*NN-SVM. The justification is fair when we connect the consequence to the reductions of training error in Fig. 4b and 5b. However, this consequence is not significant for the current study of the *k*NN-SVM because we not only are concerned with the generalization performance, but also are concerned with the effects which may take place with the model.

An investigation of cross-validation error with respect to the major parameter k is given in Fig. 8b. As mentioned in Sect. 3.3, there are inverse correlations between model complexity and k. The figure shows consistent results with the inverse correlation. With larger values of

Table 4 Extended validations of the lowest classification errors on the datasets with corresponding parameter settings

Mother Dataset	Examples in test set	Searching grids	Lowest classification error (corresponding parameter settings)		
			*k*NN-SVM $(\tilde{C}, \tilde{\sigma}, k, \eta)$	Classical SVM (C, σ)	SVM–SVM $(\tilde{C}, \tilde{\sigma}, \eta')$
Twonorm	340	$\tilde{C}$: {.1, 1, 10, 1e2} $\tilde{\sigma}$: {.5, 1, 2, 4, 8} k: odds from 3 to 23 η and η': integers form 1 to 14	4.12% (1, 2, 7, 1)	4.76% (10, 1)	4.35% (1, 1, 8)
Bupa	570	$\tilde{C}$: {1, 10, 1e2, 1e3} $\tilde{\sigma}$:{1, 2, 4, 8} k: odds from 3 to 15 η and η': integers form 1 to 14	21.56% (10, 2, 5, 7)	24.67% (10, 1)	23.11% (1e3, 2, 9)
Diabetes	864	$\tilde{C}$: {0.1, 1, 10, 1e2} $\tilde{\sigma}$: {1, 2, 4} k: odds from 3 to 23 η and η': integers form 1 to 14	6.64% (10, 1, 17, 3)	7.91% (1, 1)	9.57% (10, 1, 5)
Biomed	450	$\tilde{C}$: {0.1, 1, 10, 1e2} $\tilde{\sigma}$: {1, 2, 4, 8} k: odds from 3 to 23 η and η': integers form 1 to 14	3.16% (1, 8, 7, 3)	3.68% (1, 4)	3.40% (10, 4, 9)

The test sets are generated following the same procedures illustrated in Fig. 6, and standard deviation 0.5 is adopted. Each cell of classification error is averaged from five random repeats

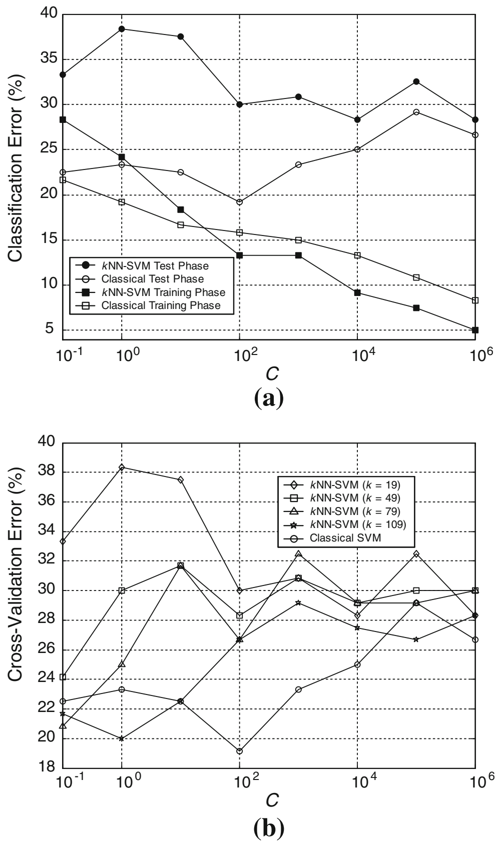


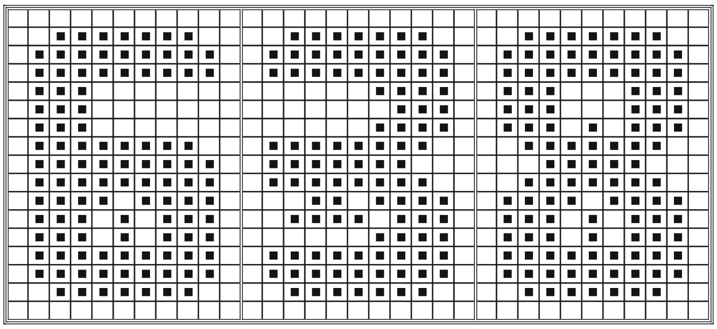
Fig. 8 Assessment of generalization performance by the K-fold cross-validation: (**a**) Cross-validation errors in both training phases and test phases. The conditions of $k = 19$, $\eta = 4$, and $C = \tilde{C}$ are used. (**b**) Cross-validation errors with increasing value of k. Setting $\eta = 4$ and $C = \tilde{C}$

k, there will generally be less the cross-validation error. In fact, a high value of k brings the local density estimation of Eq. 4 into a state of globalization. This, of course, conducts the model simplified.

Eventually, a set of optimal parameter settings should be obtained to sustain the kNN-SVM's generality. The basic principle of parameter settings is to seek its optimal values and thus to preserve the expected sheltering effect as much as possible and a low generalization error simultaneously. The selection of these values is indeed an optimization problem and will be investigated in future study.

3.6 Simulated Application on Pattern Recognition

To understand the practical applications of kNN-SVM, we present a scenario of image restoration to describe its effects on digital image processing. This scenario uses a blurred image, which might be the photograph of a moving vehicle escaping from the scene of an accident. In general, the image would be segmented into piecewise patterns and be recognized carefully, since the patterns on the image might be too fuzzy to interpret. Often, a thinning technique to extract the skeleton of the unclassified patterns is an intuitive idea for processing before recognition. In order to obtain a more distinct boundary from the hazy outline of those patterns, an additional process of utilizing SVM to enclose the patterns by their


(**a**)

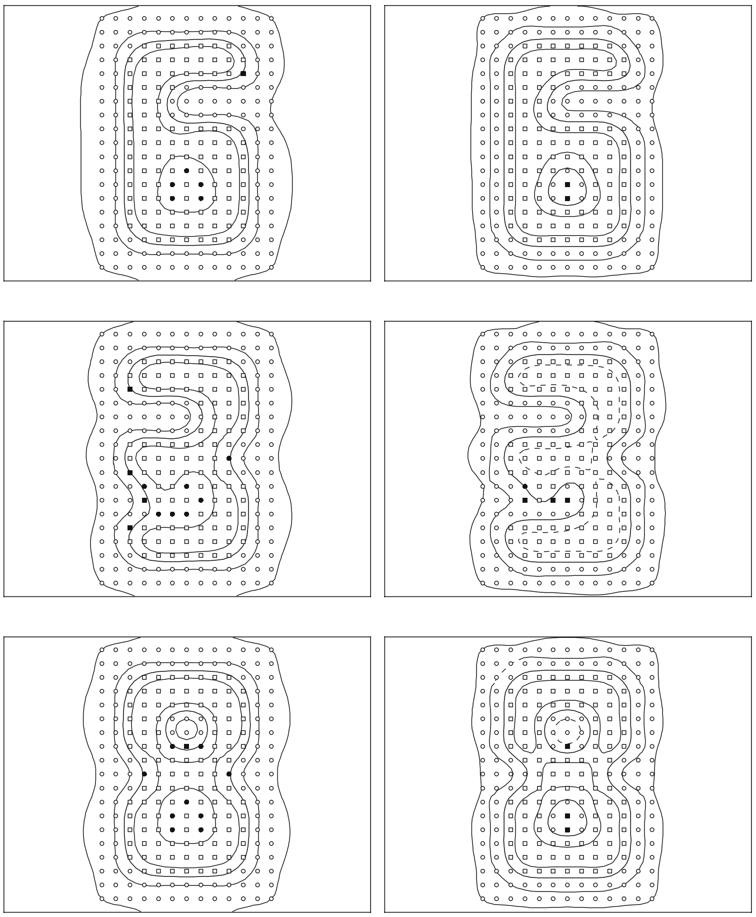

(**b**) (**c**)

Fig. 9 Simulated application on image processing: (**a**) Original patterns, (**b**) Restored by classical SVM, and (**c**) Restored by *k*NN-SVM

exact borders—not only the outer border, but also the inner border—before the thinning may be helpful for the recognition task. For instance, Fig. 9a shows three simulated characters with some inner corrupted noise. The case of such characters assumes that the inner noise is not easily rejected, and is harmful to the thinning algorithms. Serious noise may change the connected border and possibly results in an incorrect skeleton. The SVM model is therefore introduced to improve the restoration of the inner borders. The decision boundary by SVM, separating the black pixels from the surrounded blank pixels in the two dimensional spatial space, can be employed to sketch out the borders. Under equivalent parameter settings of $C = \tilde{C} = 1$, $\sigma = \sqrt{d}$, $k = 11$, and $\eta = 1$, the resultant borders restored by the classical SVM and the kNN-SVM are shown in Fig. 9b, c, respectively. In comparison, the kNN-SVM has better results than the classical model. The results show that the kNN emphasizer with its sheltering effect is really beneficial for reconstructing the corrupted sub-regions.

4 Conclusions

The paper presents an embedded classification model, kNN-SVM, to deal with the stray examples which deviate from their majorities. The objective of the model is to spotlight those heterogeneous stray examples, and provide a sheltering effect to protect them in the classification. With the preprocessor of kNN emphasizer searching locally for stray example candidates, the objective is achieved by the consecutive SVM classifier. The main results show significant improvement with the use of the paradigm. Detailed corresponding resident properties in the model are also examined to accomplish the study. In this paper, the key of parameterized class labels is used not only as a relaxation to the penalization policy in the loss function, but also as a remedy to connect both the kNN and SVM subsystems. In particular scenarios, the model demonstrates excellent accuracy and robustness for mining heterogeneous examples residing in the neighborhood of the stray training examples.

Acknowledgements The authors would like to express their appreciation to Timothy Baker, PhD and Liou-Chun Chang, PhD for their helpful comments on this paper. The first author dedicates this paper to the memory of his father (Shiou-Ching Yang, 1925-2007), who had given him rigid discipline and education.

References

1. Vapnik VN (1995) The nature of statistical learning theory. Springer-Verlag, New York
2. Vapnik VN (1999) An overview of statistical learning theory. IEEE Trans Neural Netw 10:988–999. doi:10.1109/72.788640
3. Schölkopf B, Smola AJ (2002) Learning with kernels. MIT Press, Cambridge, MA
4. Cover TM, Hart PE (1967) Nearest neighbor pattern classification. IEEE Trans Inf Theory 13:21–27. doi:10.1109/TIT.1967.1053964
5. Duda RO, Hart PE (1973) Pattern classification and scene analysis. John Wiley and sons, New York
6. Fukunaga K (1990) Statistical pattern recognition. 2. Academic Press, San Diego
7. Duda RO, Hart PE, Stork DG (2000) Pattern classification. John Wiley and sons, New York
8. Webb A (2002) Statistical pattern recognition,. 2. John Wiley and sons, New York
9. Hsu C-C, Yang C-Y, Yang J-S (2005) Associating kNN and SVM for higher classification accuracy. In: International conference on computational intelligence and security. Xi'an, China, pp. 550–555
10. Cortes C, Vapnik VN (1995) Support vector networks. Mach Learn 20:273–297
11. Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning. Springer-Verlag, New York
12. Bartlett PL, Jordan MI, McAuliffe JD (2003) Convexity, classification, and risk bounds. Technical report 638, Department of Statistics, UC Berkeley, CA
13. Vapnik VN (1998) Statistical learning theory. John Wiley and sons, New York

14. Shawe-Taylor J, Bartlett PL, Williamson RC, Anthony M (1998) Structural risk minimization over data-dependent hierarchies. IEEE Trans Inf Theory 44(5):1926–1940.doi:10.1109/18.705570
15. Zhang T (2004) Statistical behavior and consistency of classification methods based on convex risk minimization. Ann Stat 32:56–85. doi:10.1214/aos/1079120130
16. Yang C-Y, Chou J-J, Yang J-S (2003) A method to improve classification performance of ethnic minority classes in k-nearest-neighbor rule. In: IEEE international conference on computational cybernetics. Siófok, Hungary
17. Yang C-Y, Hsu C-C, Yang J-S (2006) A novel SVM to improve classification for heterogeneous learning samples. In: International conference on computational intelligence and security. Guangzhou, China, pp. 172–175
18. Schölkopf B, Smola AJ, Müller KR (1998) Nonlinear component analysis as a kernel Eigen value problem. Neural Comput 10:1299–1319. doi:10.1162/089976698300017467
19. Breiman L (1996) Bias, variance and arcing classifiers. Technical report 460, Department of Statistics, UC Berkeley, CA
20. Murphy PM (1995) UCI-benchmark repository of artificial and real data sets. University of California Irvine. http://www.ics.uci.edu/~mlearn
21. Vlachos P, Meyer M (1989) StatLib. Department of Statistics, Carnegie Mellon University. http://lib.stat.cmu.edu/