Pattern Recognition 48 (2015) 2072–2083

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Improved support vector machine algorithm for heterogeneous data

Shili Peng [a,b], Qinghua Hu [a,c,*], Yinli Chen [b], Jianwu Dang [a,c]

[a] School of Computer Science and Technology, Tianjin University, Tianjin, China
[b] Department of Computer Science and Technology, Guangdong University of Finance, Guangzhou, China
[c] Tianjin key Laboratory of Cognitive Computing, Tianjin, China

ARTICLE INFO

Article history:
Received 17 October 2013
Received in revised form
2 November 2014
Accepted 17 December 2014
Available online 24 December 2014

Keywords:
Support vector machine
Heterogeneous data
Nominal attribute
Numerical attribute
Classification learning

ABSTRACT

A support vector machine (SVM) is a popular algorithm for classification learning. The classical SVM effectively manages classification tasks defined by means of numerical attributes. However, both numerical and nominal attributes are used in practical tasks and the classical SVM does not fully consider the difference between them. Nominal attributes are usually regarded as numerical after coding. This may deteriorate the performance of learning algorithms. In this study, we propose a novel SVM algorithm for learning with heterogeneous data, known as a heterogeneous SVM (HSVM). The proposed algorithm learns an mapping to embed nominal attributes into a real space by minimizing an estimated generalization error, instead of by direct coding. Extensive experiments are conducted, and some interesting results are obtained. The experiments show that HSVM improves classification performance for both nominal and heterogeneous data.

1. Introduction

In the last decade, the support vector machine (SVM) classifier [1] has proven to be an effective method in the field of machine learning. SVM possesses advantages with respect to the management of high dimensional data and reveals effective generalization capability. It has been widely used in various applications, including handwritten digits recognition [2,3], time series classification [4,5], gene selection [6,7], and image retrieval [8–10].

However, SVM assumes that samples are represented with vectors of real numbers [11]. If nominal attributes exist, they are usually converted into numerical attributes before learning occurs. Integer and one-of-n coding are popular methods used in managing nominal attributes. If the number of values in a nominal attribute is not large, one-of-n coding might be more stable than integer coding [11]. In fact, both methods possess disadvantages. Regarding integer coding, performance is easily affected by the coding mechanism because different coding methods lead to different distances between samples. With respect to one-of-n coding, a nominal attribute is mapped into multiple binary attributes. After one-of-n coding is completed, the number of attributes is equal to the number of values of the original nominal attribute. This method can effectively prevent instability problems in integer coding. However, it may dramatically increase the dimensions of samples if a lot of different values exist in the nominal attributes. Furthermore, both integer coding and one-of-n coding do not take full advantage of the implicit classification information of samples.

Three different methods exist for managing heterogeneous data. The first is to convert nominal attributes to integers through coding, and then consider them as numerical attributes. Its major problem is instability as the performance is easily affected by the use of a coding mechanism. The second method is to discretize numerical attributes, and then treat them as nominal attributes, as done in C4.5 [12], classification and regression tree (CART) [13] and other methods. In general, discretization causes information loss. The third method is to learn a distance, such as the value difference metric (VDM), heterogeneous value difference metric (HVDM) and other methods [14–16]. This type of method can be combined with classifiers based on distance (e.g. K-nearest neighbor) [17,18]. In distance learning algorithms, we usually adopt an overlap method or a Bayesian approach to deal with nominal attributes. The overlap is a simple and effective method. However, it only determines whether nominal attributes are equal to one another, and does not fully exploit classification information. The Bayesian approach is very effective for handling nominal attributes. However, the use of this approach implies that all attributes are independent. Therefore, its performance will degenerate if relation among attributes is very high. Such as XOR data, the probability of each attribute is the same, VDM then results in zero distance between attributes [19]. Moreover, the performance of these algorithms may deteriorate when decisions depend on multiple attributes [19].

Essential differences exist between nominal and numerical attributes. In general, a numerical attribute describes a particular feature of a sample. If the value of a numerical attribute is changed, the entire sample is changed such that the new sample is no longer the

[*] Corresponding author.

http://dx.doi.org/10.1016/j.patcog.2014.12.015
0031-3203/© 2014 Elsevier Ltd. All rights reserved.

previous. However, the value of a nominal attribute simply indicates a certain nominal value and does not describe the specific character of the sample. Regardless of the value of nominal attribute, as long as the same nominal attribute is assigned the same value, no problems will occur. Thus, the nominal attribute is not limited to a fixed value which makes it possible for a nominal attribute to be mapped into a real number according to the classification information. Based on this observation, we develop a new approach to manage nominal attributes. In order to deal effectively with heterogeneous data, we use classification information by mapping nominal attributes into a real space based on generalization error estimation. The values of nominal attributes are obtained from an optimization task rather than from integer or one-of-n coding. After mapping is completed, nominal attributes are treated numerically in the subsequent learning procedure.

SVM has been successfully applied to various classification tasks that use numerical data. However, the topic of training SVM with heterogeneous data has not been fully examined. In this study, we design a novel heterogeneous support vector machine (HSVM) algorithm to classify heterogeneous data. Our HSVM maps nominal attributes into a real space by minimizing generalization error. The main advantages of HSVM are listed as follows: (1) HSVM can effectively improve the performance of SVM in dealing with nominal data or heterogeneous data, (2) HSVM can improve the interpretability of decisions, and (3) HSVM is effective in learning with imbalanced data.

The remainder of this paper is organized as follows. Section 2 reviews related studies. Section 3 presents a novel mapping algorithm for nominal attributes and HSVM. Sections 4 and 5 analyze the experiments using standard datasets. Section 6 concludes our study.

Notations used in the paper are described as follows. The variable n represents the number of training samples and x_i represents a sample with an index i. For nominal attributes, we use a^k to refer to the kth nominal attribute of the samples. Its values are expressed as $\{a^k_1, a^k_2, \ldots, a^k_m\}$.

2. Related works

We map nominal attributes into a real space by minimizing the generalization error, and then use SVM to manage heterogeneous data. Thus, in this study, we employ the SVM algorithm, generalization error, and heterogeneous data. In this section, we review relevant terms and algorithms.

2.1. SVM and kernel functions

SVM is an effective method for binary classification tasks. It constructs an optimal separating hyperplane in a feature space. By a function Φ, we map an input vector x into a high dimensional feature space [20]. Given n samples $\{(x_i, y_i)\}_{i=1}^n$, SVM searches for a linear decision function with a maximum margin between different classes in the feature space, where x_i is an input vector with d dimensions, and y_i is a class label of x_i. The decision function $f(x) = \langle w, \Phi(x)\rangle + b$ defines a linear hyperplane in the feature space. The parameters w and b are obtained by solving the following convex quadratic problem:

$$\min_{w,b} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\xi_i,$$
$$\text{s.t.} \quad y_i(\langle w, \Phi(x)\rangle + b) \ge 1 - \xi_i, \quad \xi_i \ge 0, \ \forall i, \tag{1}$$

where C is a constant that penalizes for the training errors and ξ_i is a slack variable. $w \in R^d$ and $b \in R$ are the parameters of the hyperplane [1]. Instead of solving this optimization problem, we use the Lagrangian dual function to obtain a dual formula:

$$\max_{\alpha} \quad \sum_{i=1}^{n}\alpha - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j\langle\Phi(x_i), \Phi(x_j)\rangle,$$
$$\text{s.t.} \quad \sum_{i=1}^{n}\alpha_i y_i = 0, \quad 0 \le \alpha_i \le C, \ \forall i, \tag{2}$$

where α_i is a dual variable and contains the upper bound C. The inner product $\langle\Phi(x_i), \Phi(x_j)\rangle$ in the feature space is computed by a kernel function: $\langle\Phi(x_i), \Phi(x_j)\rangle = K(x_i, x_j)$. By means of the kernel function, the inner product in a high dimensional feature space can be efficiently computed without an explicit nonlinear mapping. The dual formula shown in (2) is a convex quadratic optimization problem and possesses a global optimal solution [21]. The linear kernel function ($K_{LIN}(x_i, x_j)$), polynomial kernel function ($K_{POL}(x_i, x_j)$) and Gaussian kernel function ($K_{GAU}(x_i, x_j)$) are widely used in the following:

$$K_{LIN}(x_i, x_j) = \langle x_i, x_j\rangle;$$
$$K_{POL}(x_i, x_j) = (\langle x_i, x_j\rangle + 1)^q, \quad q \in N;$$
$$K_{GAU}(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right), \quad \sigma \in R_+. \tag{3}$$

2.2. Generalization error estimation

Some techniques used to estimate generalization errors, such as *leave one out* (LOO) as well as *span* and *radius margin* estimations, are common. LOO estimation consists of three steps: (1) remove one element from the training data, (2) construct a decision function over the remained data, and (3) test the model with the removed element [20]. LOO is nearly unbiased as an estimator of the expected generalization error [1], where the estimation is given as

$$E(p^{n-1}_{err}) = \frac{1}{n}E(L(x_1, y_1, \ldots, x_n, y_n)), \tag{4}$$

where p^{n-1}_{err} is a probability of a classification error tested on the samples of size $n-1$, and $L(x_1, y_1, \ldots, x_n, y_n)$ is the number of misclassified samples. LOO is an important statistical estimator of learning algorithms and it is frequently used in model selection. Unfortunately, it is time-consuming, as testing of each element in the training samples is required. Some generalization error estimations are derived from LOO, such as the *span* and *radius margin* estimations [1].

The concept of *span* for support vectors was first proposed by Chapelle and Vapnik [22]. The *span* is derived from an LOO error estimation [1,20] and the upper bound of the *span* is computed by means of

$$T = \frac{1}{n}\sum_{i=1}^{n}\Psi(\alpha^*_i s^2_p - 1), \tag{5}$$

where Ψ is a step function (i.e. $\Psi(x) = 1$ if $x \ge 0$, and $\Psi(x) = 0$ otherwise) [20] and α^*_i is the optimal solution for dual formation, as shown in (2). The variable s^2_p is the distance between the point $\Phi(x_p)$ and set Λ_p in the feature space where

$$\Lambda_p = \left\{\sum_{i \ne p, \alpha^*_i \ge 0}\lambda_i\Phi(x_i), \sum_{i \ne p}\lambda_i = 1\right\}. \tag{6}$$

The *span* is an upper bound of LOO and is not continuous. A small change in kernel functions causes a considerable change in the support vector set Λ_p. This change is discontinuous and results in discontinuous changes to s^2_p and error bound T [20].

The *radius margin* estimate is another generalization error estimation and can be considered as a rough upper bound of the *span* estimation. Suppose that the maximal distance between different classes is γ, and R is the minimum radius of a sphere that includes all samples in the feature space. The upper bound of radius margin error estimation is computed as in [1] as the following:

$$T = \frac{R^2}{\gamma^2}. \tag{7}$$

The margin γ is computed according to the following formula:

$$\gamma^2 = \left(\sum_{i=1}^{n} \alpha_i^*\right)^{-1}, \tag{8}$$

where α_i^* is the optimal solution of the dual formula in (2). The minimum radius R is then computed based on the following optimization task:

$$\min_{R,\mathbf{c}} \quad R^2, \text{s.t.} \quad \|\mathbf{c} - \Phi(x_i)\|^2 \le R^2, \quad i = 1..n. \tag{9}$$

where $\mathbf{c}$ is the center of the minimal sphere with radius R in the feature space constructed by a mapping function $\Phi(\cdot)$. The corresponding dual formula is given as

$$\max_{\beta_i} \quad \sum_{i=1}^{n} \beta_i K(x_i, x_i) - \sum_{i,j=1}^{n} \beta_i \beta_j K(x_i, x_j),$$
$$\text{s.t.} \quad \sum_{i=1}^{n} \beta_i = 1, \quad \beta_i \ge 0, \; i = 1..n. \tag{10}$$

This dual formula is a convex quadratic programming problem, and the radius R in primal formula (9) can be then obtained from the following formula:

$$R^2 = \beta^T \operatorname{diag}(K) - \beta^T K \beta, \tag{11}$$

where β is the optimal solution of the dual formula in (10), and $K_{n\times n} = [k(x_i, x_j)]$ is a kernel matrix. In regularization kernel functions, $K(x_i, x_i)$ is a constant [23]. We can then further simplify the programming problem given in (11). For example, in a Gaussian kernel function, $\operatorname{diag}(K)$ is a constant of 1. In order to simplify the calculation, a variance of kernel matrix K can be used to estimate the radius of sphere R [20].

It should be noted that R^2/γ^2 reflects the upper bound of a generalization error estimation and does not reflect a real test error. In a Gaussian kernel function, if the parameter σ takes an extremely large value, samples will be mapped into a flat ellipsoid with a large radius, and the radius will consider only those points showing the greatest deviation. In order to avoid this problem, we must rescale samples in the feature space. We can then obtain a much tighter error bound [22].

A tight error bound can yield improved prediction performance. Although generalization error estimation using the *span* bound is only slightly improved than when using the *radius margin*, the *radius margin* bound yields a quality result similar to that of *span* bound. In fact, optimization with a *span* bound is more difficult to implement than a *radius margin* bound [20]. In this study, we analyze optimization in relation only to the *radius margin* bound, although in our experiments, similar results have been obtained with the *span* bound.

2.3. Managing heterogeneous data

Decision trees are a class of prediction models that describe the relationship between attributes and class labels in the field of machine learning [24]. C4.5 is a popular decision tree algorithm. It has inherited the advantage of the ID3 algorithm and can be extended to manage heterogeneous data [12]. CART is another classic decision tree algorithm that can also handle heterogeneous data [25].

Distance learning is another kind of technique to classify heterogeneous data. Examples include the heterogeneous Euclidean overlap metric (HEOM), and the previously mentioned VDM and HVDM. These methods can be combined with distance-based classification algorithms, such as KNN [16].

HEOM is a simple distance measure for heterogeneous data [16]. The HEOM distance between x_i and x_j is given as

$$\text{HEOM}(x_i, x_j) = \sqrt{\sum_{k=1}^{m} d(x_i^k, x_j^k)^2}, \tag{12}$$

where m is the number of the attributes, and $d(x_i^k, x_j^k)$ is the distance to the kth attribute of x_i and x_j, defined as

$$d(x_i^k, x_j^k) = \begin{cases} 1, & \text{if } i\text{th attribute of } x \text{ or } y \text{ is unknown;} \\ \text{overlap}(x_i^k, x_j^k), & \text{if } i\text{th attribute of } x \text{ or } y \text{ is nominal;} \\ \text{distance}(x_i^k, x_j^k), & \text{if } i\text{th attribute of } x \text{ and } y \text{ is numerical;} \end{cases} \tag{13}$$

where $\text{overlap}(x_i^k, x_j^k)$ is the overlap distance between nominal attributes that is defined as $\text{overlap}(x_i^k, x_j^k) = 1$ if $x_i^k = x_j^k$; and $\text{overlap}(x_i^k, x_j^k) = 0$ otherwise. $\text{overlap}(x_i^k, x_j^k)$ is a simple and fast matching function which ignores information of classification. The $\text{distance}(x_i^k, x_j^k)$ is a normalized Euclidean distance between x_i^k and x_j^k.

VDM calculates the distance between nominal attributes by utilizing the label information of samples [16]. The more similar the frequencies of nominal attributes that appear in one class, the shorter is the distance between attributes. The VDM distance with respect to two nominal attribute values x_i^k and x_j^k is defined as

$$\text{vdm}(x_i^k, x_j^k) = \sum_{l=1}^{c} \left(P(l|x_i^k) - P(l|x_j^k)\right)^2, \tag{14}$$

where c is the number of output classes, and $P(l|x_i^k)$ is a conditional probability that the output class is l given that the kth attribute has the value x_i^k [16].

In the VDM and HVDM methods, we use (14) to compute the distance with respect to the ith nominal attribute. Regarding the distance between numerical and unknown attributes, we use the same method as in HEOM in which the distance between numerical attributes is a normalized Euclidean distance, whereas the distance between unknown attributes is the maximum distance. However, VDM utilizes only the conditional probability to manage nominal values.

Cascading is an interesting architecture for designing classification algorithms and uses multiple classifiers to increase learning accuracy [26]. Cascading usually includes two levels. The first level uses the original dataset. The outputs from this original dataset are then used as inputs for the second level. In other word, the second level is trained through the use of the original dataset as well as outputs from the previous level [27]. In addition, cascading can manage heterogeneous data. For example, one level is trained through the use of nominal attributes and another is trained through the use of numerical attributes. Yet another algorithm for managing nominal attributes is SVM [28]. This method translates a nominal attribute (with M states) to M points in a $M-1$ dimensional space, and the final position is determined by minimizing the LOO error.

In generally, the classical SVM constructs a decision function for numerical samples. If nominal attributes exist, they must be converted in advance to numerical attributes by means of coding. A simple and fast method is positive integer coding [11]. However, different coding techniques lead to different kernel matrices. Another coding technique for nominal attributes is one-of-n coding [11] previously mentioned. A nominal attribute with m possible values is decomposed into m attributes. For example, a color attribute with values of {red, green, blue} is decomposed to three binary attributes. The inner product of two nominal attributes is 0 or 1 accordingly. If two samples have the same nominal value, the inner product is 1; otherwise, it is 0. In essence, this kernel represents a matching function. The inner product using one-of-n coding indicates the number of nominal attributes that are equal.

A previous study has reported that one-of-n coding is more stable than integer encoding when the range of nominal attributes is not extensive [11]. However, the dimension increases considerably if many values exist within an attribute is very large.

3. HSVM for heterogeneous data

Classification tasks are usually described by nominal and numerical attributes. In this study, we design a novel algorithm (HSVM) to manage heterogeneous data. HSVM maps nominal attributes into a real space by minimizing the generalization error. The mapping is learned by a gradient descent scheme. Starting from initialization, HSVM iteratively updates the assignments of nominal attributes in the learning procedure.

As shown in Fig. 1, we can iteratively adjust the assignments of the nominal attributes in the normal direction of a classification plane in order to increase the margin between different classes. The margin between classes increases if the assignments of red-circled support vectors increase while blue-starred support vectors decreased within each dimension. In other words, we can increase the classification margin by moving the red-circled points in an upper-right direction or by moving the blue-starred points in a lower-left direction. If we simply consider the *margin* bound as a generalization error estimation to update the assignments of nominal attributes, the margin may increase constantly such that the algorithm will not converge. The *radius margin* generalization error bound avoids this problem because the *radius* parameter prevents values of nominal attributes from infinitely increasing. In fact, the error bound of a classical SVM depends not only on the margin, but also on the radius of the smallest sphere containing all samples in the feature space. Nevertheless, we usually ignore the *radius* in the optimization process because the radius of a sphere is a fixed constant in a given feature space [29].

Fig. 2 presents 400 samples in a 2-D space, i.e., each sample is described by two nominal attributes. Twenty possible values for each nominal attribute exist, thus yielding 400 different combinations.

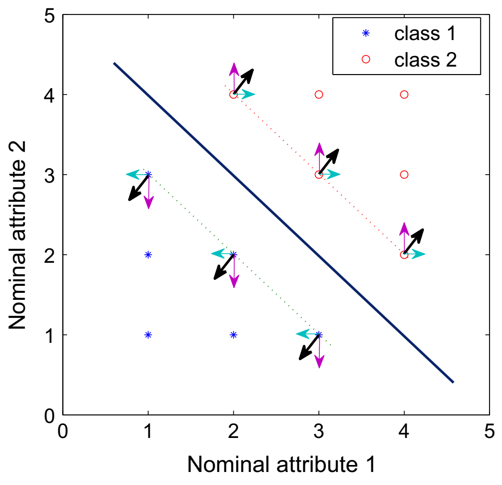

Fig. 1. Schematic diagram for mapping nominal attributes. The vertical and horizontal axes indicate different nominal attributes that are coded by integer. There is a classification plane in the middle of the figure. We can expand the margin between classes by increasing the values of red-circle points in the direction of vertical and horizontal axes simultaneously, or decreasing the values of blue-star-like points in the same way. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Red circles and blue stars represent samples from different classes. Fig. 2(a) and (b) shows the samples before and after mapping respectively, by means of HSVM. Curves in the figures represent the decision hyperplane learned with the Gaussian kernel. The *radius margin* error estimation is used to learn the mapping function. We observe that the 400 training samples are collected in nine positions after mapping nominal attributes into the real space. We can explain this interesting phenomenon as follows. The role of certain feature values in the decision is the same and they are assigned the same value by means of HSVM. This means that we can use these nine new samples to replace the original 400 training samples. The number of feature values is reduced to three. In addition, the computational cost reduces considerably. Moreover, the classification margin increases significantly after mapping. The corresponding generalization error estimation falls from 0.2224 to 0.0217, and the classification accuracy rises from 98.5% to 100%.

3.1. Mapping function for nominal attributes

HSVM maps the nominal attributes into a real space by iteratively updating the values of nominal attributes such that the generalization error estimation is minimized. HSVM alternates the classical SVM optimization with a gradient descent procedure. We next provide the gradient (regarding values of nominal attributes) based on the *radius margin* error bound. The objective function of HSVM is defined as

$$T=\frac{R^2}{\gamma^2}=R^2 * \|w\|^2, \tag{15}$$

where R^2 is the radius of a minimum sphere enclosing all samples and γ^2 is the maximum margin between classes in the feature space. The margin γ is equal to $1/\|w\|$. Therefore, we can describe the *radius margin* error bound as R^2/γ^2 as $R^2 * \|w\|^2$. We suppose that parameters α^* solve the quadratic dual formula given in (2). The weight vector $\|w\|$ then realizes the maximal margin hyperplane in the feature space with a geometric margin [30] such that

$$\gamma^2=\frac{1}{\|w\|^2}=\left(\sum_{i=1}^{n}\alpha_i^*\right)^{-1}. \tag{16}$$

The partial derivative of the generalization error estimation T with respect to a nominal attribute a_i^k is computed as

$$\frac{\partial T}{\partial a_i^k}=R^2 * \frac{\partial \|w\|^2}{\partial a_i^k}+\frac{\partial R^2}{\partial a_i^k} * \|w\|^2. \tag{17}$$

Starting from the initial values of nominal attributes, we iteratively update all values using (17) until the generalization error bound T converges to an optimal point. In order to calculate the partial derivative of the error bound, we must compute the derivative of R^2 and $\|w\|^2$ with respect to the nominal attributes.

The optimal objective value $\frac{1}{2}\|w\|^2$ of primal formula in (1) is equal to the optimal objective value of the dual formula in (2) because no duality gap exists in the optimal solution. In mathematical terms, this can be expressed as

$$\|w\|^2=2 * \sum_{i=1}^{n}\alpha_i^* - \sum_{i,j=1}^{n}\alpha_i^*\alpha_j^* y_i y_j K(x_i^k,x_j)=2\alpha^T\mathbf{1}-\alpha^T K\alpha, \tag{18}$$

where α^T is the optimal solution $(\alpha_1^*,\alpha_2^*\ldots\alpha_n^*)^T$. Note that the value of α depends implicitly on nominal attribute values, but the partial derivative of $\|w\|^2$ with respect to nominal values is irrelevant to α when α is the optimal solution for $\|w\|^2$ [20]. We can then calculate the derivative of $\|w\|^2$ with respect to nominal attributes

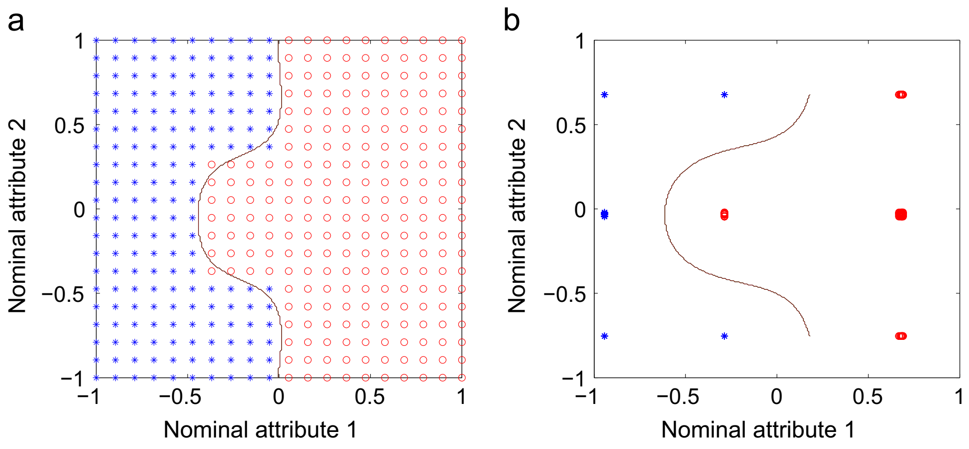

Fig. 2. Demonstration of mapping nominal attributes. A total of 400 samples are shown in the figure, and each sample has two nominal attributes containing 20 different values. The nominal attributes are coded by integer coding and then are normalized. A classification boundary appears in the middle of each figure. (a) Initialize and rescale nominal attributes by integer coding. (b) Map nominal attributes with HSVM. Some samples (indicated by blue stars and empty red circles) gather together. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

a_i^k according to the following:

$$\frac{\partial \|w\|^2}{\partial a_i^k} = -\sum_{i,j=1}^{n} \alpha_i^* \alpha_j^* y_i y_j \frac{\partial K(x_i,x_j)}{\partial a_i^k} = -\alpha^T \left(\frac{\partial K}{\partial a_i^k}\right)\alpha. \tag{19}$$

In the same manner, we can compute the derivative of radius R^2 with respect to nominal attribute value a_i^k from (10) and (11) such that

$$\frac{\partial R^2}{\partial a_i^k} = \sum_{i=1}^{n} \beta_i^* \frac{\partial K(x_i,x_i)}{\partial a_i^k} - \sum_{i,j=1}^{n} \beta_i^* \beta_j^* \frac{\partial K(x_i,x_j)}{\partial a_i^k}$$
$$= \beta^T \operatorname{diag}\left(\frac{\partial K}{\partial a_i^k}\right) - \beta^T \left(\frac{\partial K}{\partial a_i^k}\right)\beta, \tag{20}$$

where β^* is the optimal solution for the dual problem given in (10).

In Gaussian or other kernel functions, $\partial K(x_i,x_i)/\partial a_i^k$ is zero. Thus, the corresponding derivative of the radius is simplified to

$$\frac{\partial R^2}{\partial a_i^k} = -\sum_{i,j=1}^{n} \beta_i \beta_j \frac{\partial K(x_i,x_j)}{\partial a_i^k} = -\beta^T \left(\frac{\partial K}{\partial a_i^k}\right)\beta. \tag{21}$$

We next combine (20) and (19) into (17) to obtain the derivative of the error bound T with respect to the nominal attribute value a_i^k so that

$$\frac{\partial T}{\partial a_i^k} = -R^2\left(\alpha^T \frac{\partial K}{\partial a_i^k}\alpha\right) + \|w\|^2 \left(\beta^T \operatorname{diag}\left(\frac{\partial K}{\partial a_i^k}\right) - \beta^T \frac{\partial K}{\partial a_i^k}\beta\right), \tag{22}$$

where R^2 and β are learned from dual optimization (10), and $\|w\|^2$ and α are trained from (2). We use this formula to update values of nominal attributes to decrease the estimated generalization error. In regularization kernel functions, we may reformulate (21) as (22), and then the derivative of the error bound is simplified to

$$\frac{\partial T}{\partial a_i^k} = -R^2\left(\alpha^T \frac{\partial K}{\partial a_i^k}\alpha\right) - \|w\|^2 \left(\beta^T \frac{\partial K}{\partial a_i^k}\beta\right), \tag{23}$$

where $R^2 = \beta^T \mathbf{1} - \beta^T K \beta$ and $|w\|^2 = 2\alpha^T \mathbf{1} - \alpha^T K \alpha$. The derivative of the kernel matrix K with respect to a_i^k is a major part of (22) and (23), and we can calculate the derivative of the kernel matrix according to a given kernel function.

For example, the Gaussian kernel is $k(x,x') = \exp(-(\|x-x'\|^2)/(2\sigma^2))$. The partial derivative of the kernel matrix K with respect to nominal attribute value a_i^k is

$$\frac{\partial K(x,x')}{\partial a_i^k} = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)\left(-\frac{1}{2\sigma^2} \times \frac{\partial\left(\|x-x'\|^2\right)}{\partial a_i^k}\right)$$
$$= K(x,x')\left(-\frac{x^k - x'^k}{\sigma^2}\right) \times \begin{cases} +1 & \text{if } x^k = a_i^k \\ -1 & \text{if } x'^k = a_i^k \\ 0 & \text{otherwise,} \end{cases} \tag{24}$$

where a_i^k is the ith nominal attribute value of the kth dimension of Sample x, and x^k and x'^k are real values of the kth dimension of Samples x and x'.

For a polynomial kernel function, the kernel function is $k(x,x') = (\langle x,x'\rangle + c)^d$, and its partial derivative with respect to a_i^k is

$$\frac{\partial K(x,x')}{\partial a_i^k} = d(\langle x,x'\rangle + c)^{d-1} \times \frac{\partial(\langle x,x'\rangle)}{\partial a_i^k}$$
$$= d(\langle x,x'\rangle + c)^{d-1} \times \begin{cases} x'^k & \text{if } x^k = a_i^k \text{ and } x'^k \neq a_i^k \\ x^k & \text{if } x^k \neq a_i^k \text{ and } x'^k = a_i^k \\ 2a_i^k & \text{if } x^k = a_i^k \text{ and } x'^k = a_i^k \\ 0 & \text{otherwise,} \end{cases} \tag{25}$$

In the same manner, we can obtain the derivatives of the other kernels.

3.2. HSVM

We next present the details of the HSVM. After mapping nominal attributes into a real space, the nominal and numerical attributes are combined. A notable fact is that numerical attributes of samples participate in the entire learning process, including the mapping of nominal values and computing of the generalization error. The procedure is formulated in Algorithm 1.

Algorithm 1. Heterogeneous support vector machine algorithm.

```
Input:
    Heterogeneous dataset $X = \{x_1, x_2, \ldots, x_n\}$.
Output:
    A SVM classifier and a mapping table for each nominal
    attribute.
Iteration:
1:  $t \leftarrow 0$
2:  Initialize each nominal value $a_i^k$ (using $P(k|x_i)$ or an integer).
3:  while stop criteria not satisfied do
```

```
4:   Calculate margin and kernel matrix $K$ by solving
     optimization problem (2).
5:   Compute radius $R$ by solving (10), or approximate $R$ with
     variance of the matrix $K$.
6:   Calculate $\frac{\partial T}{\partial a_i^k}$ for each nominal value according to (22).
7:   Update mapping value $(a_i^k)^{t+1} \leftarrow (a_i^k)^t + \gamma \frac{\partial T}{\partial a_i^k}$ and
     compute the error bound $T$ with a step $\gamma$.
8:   $t \leftarrow t+1$
9: end while
```

In Step 2, we initialize each nominal value a_i^k with its conditional probability $P(k|x_i)$ in the same manner as VDM:

$$P(k|x_i) = \frac{N_{a_i,k,c}}{N_{a_i,k}}, \tag{26}$$

where $N_{a_i,k,c}$ is the total number of times of a_i of the kth attribute that occurs in Class c, and $N_{a_i,k}$ is the total number of times of a_i of the kth attribute in all samples, that is $N_{a_i,k} = \sum_{c=1}^{C} N_{a_i,c}$. C is the total number of classes [19]. In (26), we fix Class c as either the positive or the negative class. In addition, we can certainly initialize the nominal attribute values by means of integer or other coding.

4. Experiments

We test the proposed algorithm HSVM. Datasets are downloaded from the UCI machine learning repository [31]. We compare the performance of HSVM with other popular classification algorithms, including CART, J4.8, SVM, and IBk. The detailed information of datasets is given in Table 1. The samples containing missing values are removed before experiments are conducted. The first 10 classification tasks possess only nominal attributes, and the other 10 tasks are described with both nominal and numerical attributes.

Three multi-class tasks are shown in Table 1. The *one-against-one* and *one-against-all* methods are two popular strategies for extending SVM [32]. In the *one-against-one* approach, we must train C_2^N binary classifiers that separate each pair of classes. The *one-against-all* approach trains only N different binary classifiers that distinguish the samples in a single class from those in all remaining classes. Assuming that each binary classifier is well tuned, the *one-against-all* strategy is as accurate as other approaches using SVM [32]. In this study, we utilize *one-against-all* for multi-class tasks and we evaluate the algorithms based on 10-fold cross validation. Both SVM and HSVM use a Gaussian kernel function in the following experiments.

4.1. Comparing HSVM with standard SVM

In this section, we compare HSVM with SVM using a *tic-tac-toe* dataset. Both HSVM and SVM estimate the generalization error using the *radius margin* error bound.

In Fig. 3, the blue curve represents the generalization error bound, and the red curve represents the actual prediction error. The errors shown in Fig. 3(a) and (b) are computed using SVM and HSVM, respectively. As shown in Fig. 3(a), the actual prediction error of SVM does not decrease efficiently, although the SVM utilizes the *radius margin* error bound to optimize parameters. In Fig. 3(b), HSVM uses the same error bound *radius margin* to optimize parameters and map nominal values into a real space. Both the error bound and the actual prediction error decrease significantly, as shown in Fig. 3(b).

Table 1
Datasets information.

No.	Dataset	Instance	Nominal	Numerical	Class
1	tic-tac-toe	958	9	0	2
2	monks-1	432	6	0	2
3	monks-2	432	6	0	2
4	monks-3	432	6	0	2
5	breast cancer	286	9	0	2
6	solar flare1X	323	10	0	2
7	kr-vs-kp	3196	36	0	2
8	spect	267	23	0	2
9	vote	435	16	0	2
10	mushroom	8124	22	0	2
11	crx	690	9	6	2
12	heart	270	7	6	2
13	hepatitis	155	13	6	2
14	sick	2643	16	6	2
15	adult	48 842	8	6	2
16	credit	690	8	6	2
17	tae	151	2	3	3
18	cmc	1473	7	2	3
19	ann	3428	15	6	3
20	allhypo	2800	21	8	2

In order to explain the impact of different initial assignments of nominal attributes, we use *monk*-1 data as an example. A comparative analysis is given in Fig. 4, and both SVM and HSVM use the same error bound *radius margin*. In Fig. 4, the curve containing blue circles indicates errors (test and estimate errors) using an initial assignment method, whereas the red rhombus indicates errors using another initial assignment method.

As shown in Fig. 4, the estimated and real test errors of SVM are affected by initial assignment methods for nominal attributes. SVM will converge at the fifth iteration. In addition, as shown in Fig. 4(b), the real prediction error may increase slightly as the error bound decreases.

Fig. 4(c) and (d) shows that the estimated and real test errors decrease in HSVM with an increase in the number of iterations, and the algorithm converges quickly. The different initial assignments do not yield the same results, although the results are very similar after the convergence. The difference emerges from the fact that the generalization error estimation is a non-convex optimization function [33] and it may converge at a local optimal point. Either random search methods or global optimization methods may be used to select hyper-parameters [34,35].

4.2. Comparing HSVM with other algorithms

VDM, C4.5 and CART are commonly used in dealing with nominal data. We compare these algorithms with HSVM, and the results are shown in Table 2, where both SVM and HSVM algorithms use the *radius margin* error bound. HVDM, HMOM and HEOM are three KNN algorithms using the VDM distance, the *Manhattan overlap* distance and the *Euclidean overlap* distance, respectively [36]. The J4.8, IBk and CART algorithms are implemented in Weka 3.6 [37,38]. The results of the cascading method are those reported in [27]. However, the corresponding deviations of cascading are not reported in [27].

In Table 2, regarding each data and classification algorithm, the first row shows the average accuracy, whereas the second shows the corresponding deviation. The highest accuracy for each dataset is given in boldface. As shown in Table 2, the accuracy of HSVM is superior to that of other algorithms. Moreover, the average accuracy of HSVM is higher than that of SVM by 4.09%.

Many heterogeneous datasets exist that contain both nominal and numerical attributes in practice. We compare the performances of HSVM with other classifiers and the results are shown in Table 3.

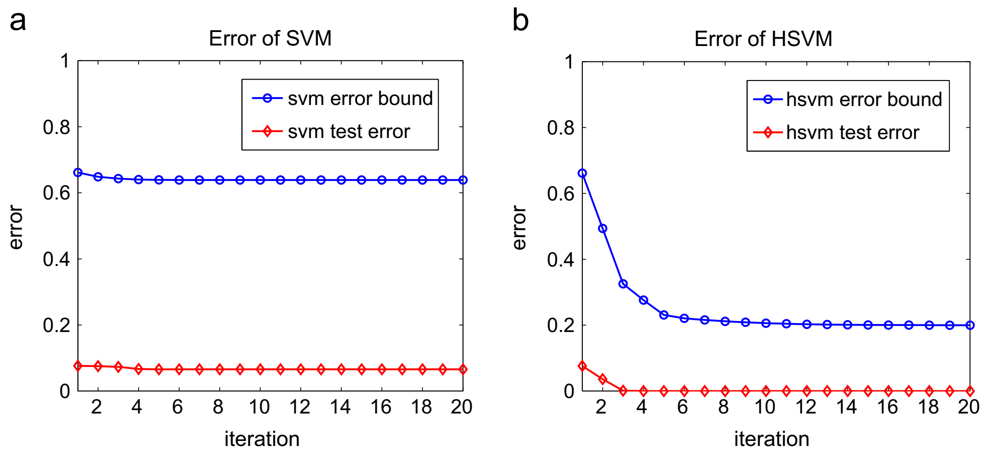

Fig. 3. Different initial assignment. Error bound and test error with (a) SVM and (b) HSVM. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

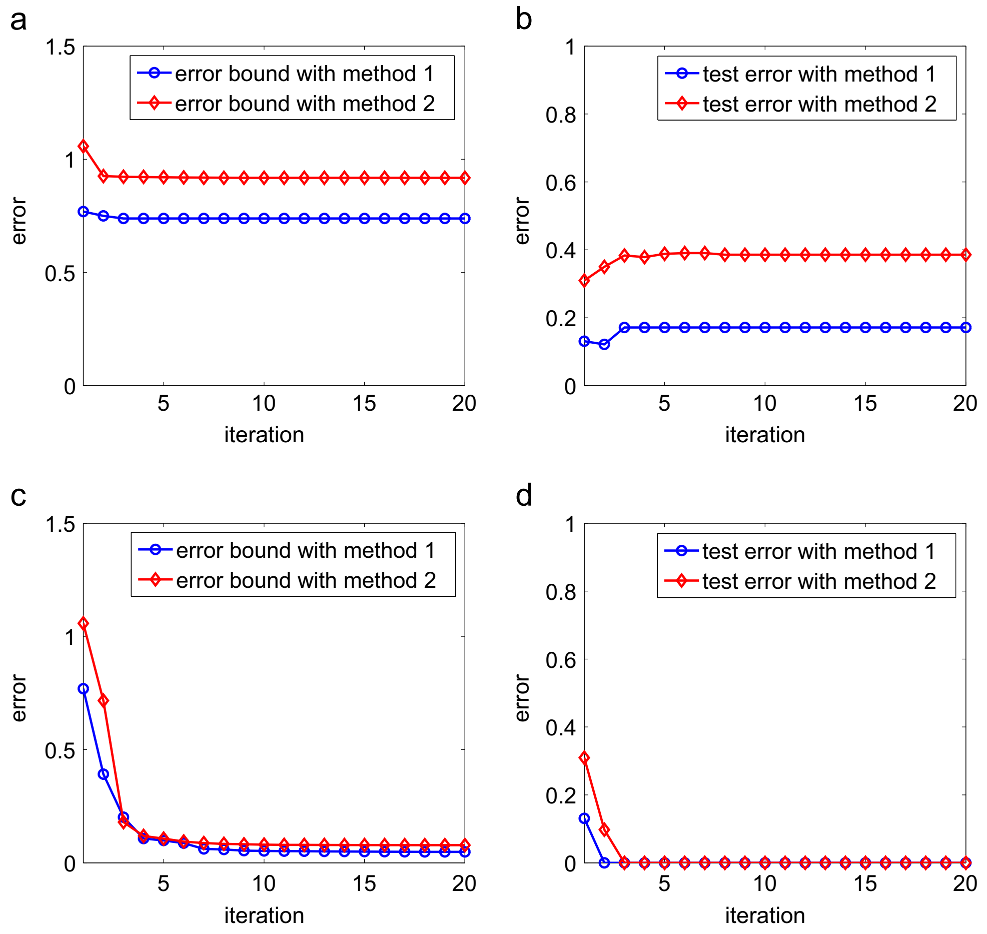

Fig. 4. Influence of different initial assignments using different algorithms. (a) Generalization error estimation using SVM. (b) Actual test error using SVM. (c) Generalization error estimation using HSVM. (d) Actual test error using HSVM. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Based on these results, we observe that HSVM can effectively improve the accuracy of classification. A paired t-test is a common way to determine whether the average difference in results between two classifiers is considerably different [39]. In order to compare these algorithms, we use paired t-test to compare the performances of different algorithms. We first obtain the average classification accuracies of the 10 nominal and heterogeneous datasets computed with different classification techniques, and then conduct t-test on the classification accuracies. According to our experiments, although HSVM does not produce the best performance on all the classification tasks, averagely speaking, HSVM is better than the other algorithms under confidence of 95%.

5. Discussion

In this section, we analyze the properties of HSVM beyond classification performance.

Table 2
Comparative analysis of accuracy and variance for nominal datasets.

Dataset	SVM	HSVM	HVDM	HMOM	HEOM	J4.8	IBk	CART	CASDE
tic-tac-toe	93.43 (0.05)	**100.00** (0.00)	89.98 (0.32)	98.64 (0.12)	98.64 (0.12)	84.24 (0.35)	98.85 (0.23)	92.90 (0.24)	98.33 –
monks-1	89.25 (0.32)	**100.00** (0.00)	80.09 (0.45)	99.77 (0.05)	99.77 (0.05)	96.53 (0.15)	99.54 (0.29)	90.51 (0.20)	99.34 –
monks-2	83.10 (0.31)	**100.00** (0.00)	94.91 (0.23)	56.71 (0.66)	94.91 (0.23)	67.13 (0.47)	59.26 (0.49)	91.90 (0.23)	67.14 –
monks-3	97.14 (0.09)	**100.00** (0.00)	**100.00** (0.00)	99.07 (0.10)	99.07 (0.10)	**100.00** (0.00)	98.84 (0.22)	**100.00** (0.00)	98.63 –
breast-ca	75.39 (0.19)	**76.15** (0.15)	67.48 (0.60)	74.83 (0.50)	74.83 (0.50)	74.13 (0.44	72.38 (0.50)	69.23 (0.46)	74.14 –
solar-1X	97.75 (0.05)	97.75 (0.05)	96.90 (0.18)	97.83 (0.15)	97.83 (0.15)	97.83 (0.15)	95.96 (0.19)	97.83 (0.15)	**97.84** –
kr-vs-kp	97.26 (0.02)	98.99 (0.01)	96.81 (0.18)	97.00 (0.17)	97.00 (0.17)	99.44 (0.07)	97.28 (0.19)	96.27 (0.19)	**99.44** –
spect	83.90 (0.46)	**84.29** (0.41)	81.65 (0.43)	76.40 (0.49)	76.40 (0.49)	78.27 (0.39)	75.28 (0.42)	80.90 (0.38)	83.68 –
vote	96.03 (0.15)	**96.92** (0.09)	95.63 (0.21)	93.10 (0.26)	93.10 (0.26)	96.32 (0.17)	92.41 (0.24)	95.40 (0.20)	96.69 –
mushroom	**100.00** (0.00)	**100.00** (0.00)	**100.00** (0.00)	**100.00** (0.00)	**100.00** (0.00)	**100.00** (0.00)	**100.00** (0.00)	99.94 (0.00)	**100.00** –
Average	91.32	**95.41**	90.34	89.34	93.15	89.39	88.98	91.49	91.52

Table 3
Comparative analysis of heterogeneous datasets.

Dataset	SVM	HSVM	HVDM	HMOM	HEOM	J4.8	IBk	CART
crx	85.84 (0.11)	**86.60** (0.11)	83.92 (0.40)	85.60 (0.38)	84.69 (0.39)	85.30 (0.35)	81.16 (0.43)	86.37 (0.34)
heart	83.12 (0.47)	**84.48** (0.64)	79.26 (0.46)	80.37 (0.44)	77.04 (0.48)	77.41 (0.45)	74.81 (0.50)	78.89 (0.42)
hepatitis	87.24 (0.34)	**90.76** (0.60)	84.52 (0.39)	84.52 (0.39)	85.81 (0.38)	89.68 (0.31)	82.58 (0.41)	90.32 (0.29)
sick	96.76 (0.04)	**98.89** (0.02)	96.19 (0.21)	96.93 (0.20)	96.20 (0.20)	97.60 (0.15)	96.50 (0.19)	97.10 (0.16)
adult	86.82 (0.11)	**86.87** (0.09)	78.40 (0.46)	78.98 (0.46)	77.82 (0.47)	81.86 (0.37)	77.91 (0.39)	82.77 (0.36)
Au-credit	86.33 (0.18)	**86.49** (0.24)	82.61 (0.42)	83.91 (0.40)	81.88 (0.43)	85.22 (0.35)	83.62 (0.36)	85.65 (0.34)
tae	73.61 (0.14)	**77.27** (0.14)	60.26 (0.63)	64.02 (0.60)	60.71 (0.63)	68.65 (0.47)	73.07 (0.51)	69.76 (0.47)
cmc	70.67 (0.05)	**72.78** (0.09)	63.48 (0.60)	59.81 (0.63)	59.81 (0.63)	68.97 (0.47)	59.67 (0.62)	71.49 (0.44)
ann-test	94.77 (0.02)	98.94 (0.01)	95.80 (0.21)	94.31 (0.24)	93.14 (0.26)	**99.30** (0.08)	92.21 (0.28)	**99.30** (0.08)
allhypo	94.51 (0.03)	96.33 (0.02)	94.5 (0.23)	93.73 (0.25)	92.45 (0.27)	**99.54** (0.07)	91.37 (0.29)	99.49 (0.07)
average	85.97	**87.94**	81.89	82.22	80.95	85.35	81.29	86.11

5.1. Interpretability of decisions

Because SVMs nonlinearly map samples into a high dimensional feature space when constructing the optimal separating hyperplane, the decision is difficult to understand. We can improve the interpretability of the decision by mapping nominal attributes into a real space using HSVM. Regarding the Gaussian kernel, calculating the coordinates of samples in the feature space is difficult, but we can analyze the distance between samples in the input space and feature space. Since the Gaussian kernel is a monotonic function of distance, the closer the samples are to each other in the input space, the smaller is the distance between samples in the feature space and vice versa. Therefore, the mapping value in the feature space indirectly reflects the distance in the input space. The mapping values of nominal attributes directly reflect the distance between samples when we use the linear kernel function.

Table 4
Mapping nominal attribute “odor” to real numbers.

Nominal value	Almond	Anise	Creosote	Fishy	Foul	Musty	None	Pungent
Initial value	1	2	3	4	5	6	7	8
Mapping value	1.076	1.138	6.89	null	6.883	null	1.657	8.355

We use *mushroom* data as an example to explain the advantages of HSVM. The mapping values of the nominal attribute “odor” are shown in Table 4, where the Gaussian kernel and the *radius margin* generalization error are considered.

Because the attributes “fishy” and “musty” do not appear in the *mushroom* data, the mapping values for these are *null* in Table 4. We can clearly observe that, based on Table 4, the mapping values of “almond”, “anis” and “none” attributes are quite similar, and samples containing these attributes corresponded to the innocuous mushroom. It is important to note that the mapping value of “none” attribute is changed from the initial value of 7 to 1.657. The mapping values of “creosote”, “foul” and “pungent” attributes are all relatively large and samples showing these types of odor corresponded to the noxious mushroom. This confirms our intuition and is identical to the analysis in [19].

5.2. Analysis on learning with imbalance data

A dataset is imbalanced if the categories are not equally represented or nearly so, that is, one class is at least under-represented relative to the others. Many imbalanced datasets appear in practical applications such as intrusion detection, medical diagnosis and text categorization [40,41]. The

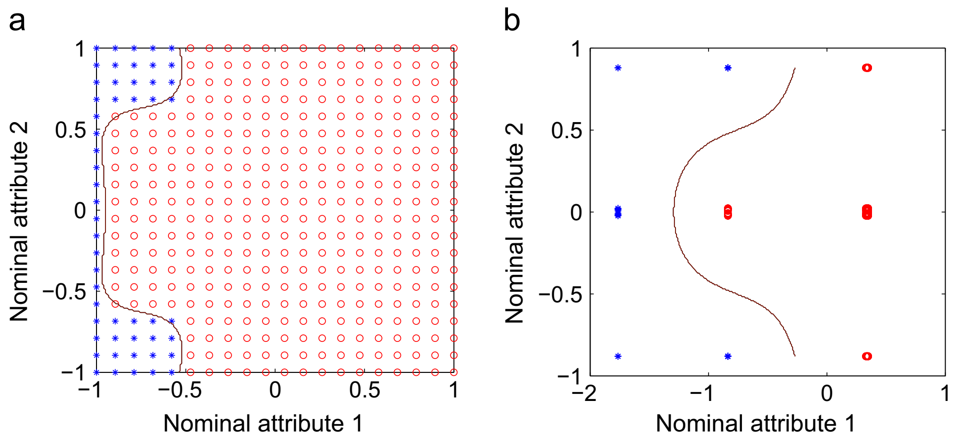

Fig. 5. Imbalanced data processing and analysis. Manage imbalanced data with (a) SVM and (b) HSVM. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

Table 5
Comparison of the performance of HSVM with that of other classifiers for handling imbalanced data.

Dataset	SVM	HSVM	HVDM	HMOM	HEOM	J4.8	IBk	CART
Correct in c1	16	44	10	4	4	0	17	8
Error in c1	(34)	(6)	(40)	(46)	(46)	(50)	(33)	(42)
Accuracy c1	32	**88**	20	16	16	0	34	16
Correct in c2	290	290	267	286	286	290	289	282
Error in c2	(0)	(0)	(23)	(4)	(4)	(0)	(1)	(8)
Accuracy c2	**100**	**100**	92.07	98.62	98.62	**100**	99.65	97.24
Total-acc	90	**98.24**	81.47	85.29	85.29	85.29	90	85.29

classification of imbalanced datasets is often associated with asymmetric costs of misclassification [42]. SVM has been well developed for classification and recognition tasks, but this has been generally based on the assumption that datasets are balanced. Performance is not satisfactory when samples are seriously imbalanced. Currently, methods for imbalanced data focus on preprocessing data or improving classifiers. HSVM combines these two methods in a single learning procedure. The attribute mapping can be interpreted as preprocessing data, but it is also a step in constructing classifiers. As shown in Fig. 5 and Table 5, the mapping methods can effectively deal with imbalanced datasets. We should note that the *one-against-all* method, which is used in multi-classification, may lead to imbalanced datasets.

In Fig. 5, the blue-starred points belong to one class and the red-circled points belong to another class. Fig. 5(a) uses a standard SVM method to construct a classifier, and the curve in the middle of the figure is a separating hyperplane. Fig. 5(b) uses HSVM to learn a separating hyperplane. As shown in Fig. 5(b), HSVM maps the red and blue points to four and five groups, respectively. It is important to note that there are a fewer number of blue points than red points in the original dataset, but there are a greater number of blue groups than red groups. If we utilize a clustering algorithm, the red and blue groups can be represented by four and five points respectively. Then the proportion of blue points and red points is altered from 52:348 to 5:4. HSVM can increase the margin from 0.0274 to 0.1139, and the real prediction accuracy increases from 95.75% to 100%. As shown in Fig. 5, HSVM changes the distribution of samples by mapping nominal attributes into a real space.

Table 5 shows the test results of imbalanced data. In the original *monk-2* dataset, the numbers of samples in the first class and the second class are 140 and 290 respectively. We randomly select sample of 50 samples in the first class and construct a new imbalanced dataset that contains 50 samples in the first class and 290 samples in the second class. We then compare the performance of HSVM with that of other classifiers in Table 5.

In the dataset column in Table 5, *correct in c*1 and *correct in c*2 denote the number of samples that are correctly classified in class 1 and class 2 respectively. The *error in c*1 and *error in c*2 denote the number of samples (i.e. given in parentheses) that are misclassified by the classifier. As shown, all samples in class 1 are misclassified as class 2 by the J48 algorithm, and the IBk and SVM algorithms prove to have a slight advantage at classifying samples in class 1. HSVM has obvious advantages at classifying samples in classes 1 and 2.

5.3. *Analysis on running time*

We analyze the running time of our new algorithm by comparing it with SVM using the *ann-test* dataset. The hyper-parameters of SVM are learned by the gradient descent algorithm instead of by a grid search method because the former has lower computational costs [20]. The hyper-parameters of HSVM are learned together with the nominal mapping values. Both SVM and HSVM use the gradient norm as a stopping criterion in the experiments. In other words, if the gradient norm is less than the predefined tolerance, the iteration is halted.

As shown in Fig. 6(a), SVM is faster than HSVM when the stopping gradient norm is a very small value, i.e., 10e − 35. Two reasons explain this phenomenon. The first is that the gradient computational cost of HSVM is higher than that of SVM. HSVM must calculate the gradient of hyper-parameters and nominal values. SVM must only calculate the gradient of hyper-parameters.

The second reason is that the number of iterations varies considerably when the stopping criterion is very small. The number of iterations in HSVM is greater than that in SVM. SVM will converge to a large error bound with few iterations. However, HSVM will converge to a small error bound with many iterations. For example, the error bound of HSVM converges to 0.1813 with 20 iterations and the error bound of SVM converges to 0.205 with 5 iterations, as shown in Fig. 6(c).

If the stopping criterion (gradient norm) is not a very small number, HSVM and SVM require a similar running time. As shown in Fig. 6(b), HSVM and SVM use the gradient norm 10e − 5 as a stopping criterion and, because their iterations are similar, their running times are nearly identical. Specifically, the iteration of SVM is 5 and that of HSVM is 7 until convergence occurs, as shown in Fig. 6(d).

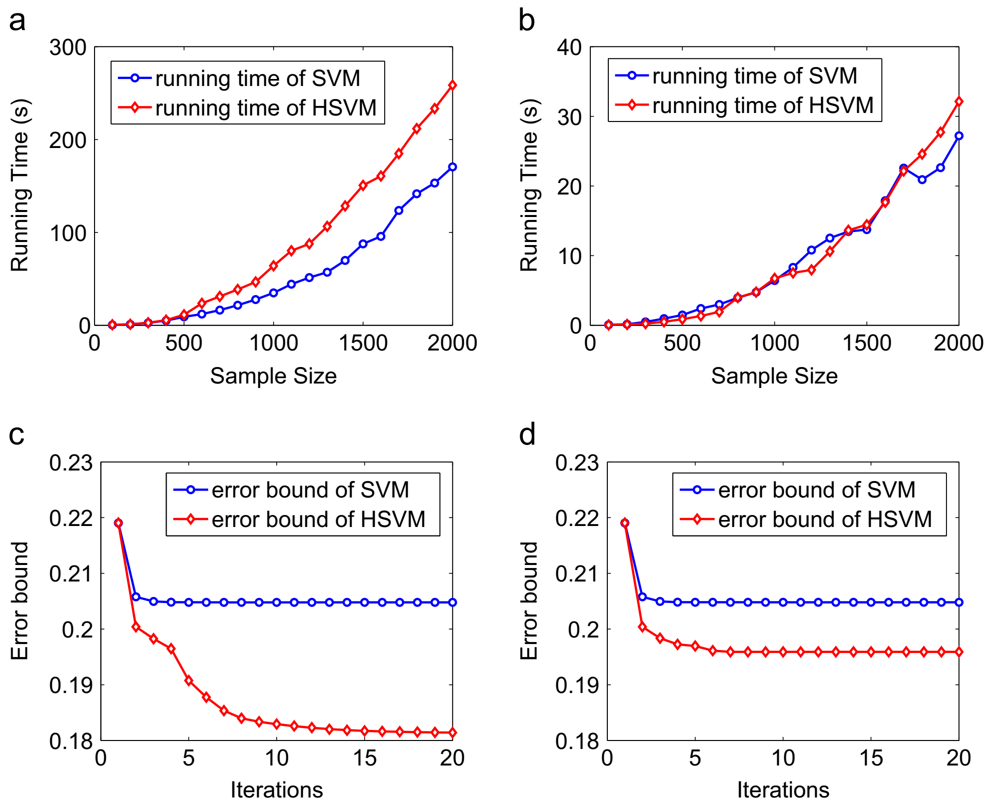

Fig. 6. Run-time analysis using the *ann−test* dataset. Running time with (a) 10e−35 tolerance and (b) 10e−5 tolerance. Error bound with (c) 10e−35 tolerance and (d) 10e−5 tolerance.

In point of fact, the convergent speed of HSVM is faster than that of SVM. However, the computational cost of HSVM is greater than that of SVM, primarily because HSVM can converge to a smaller value and requires more iterations. The computational cost chiefly depends on the number of iterations. We examine the number of iterations of both HSVM and SVM. We use gradient norm as the stopping criteria and a tolerance of 10e−3. The maximal number of iterations is 20. Both SVM and HSVM use the *radius margin* error bound in the learning procedure.

As shown in Table 6, HSVM requires more iterations than does SVM because we use the difference between two adjacent iterations as the stopping criterion. In fact, the convergence rate of HSVM is larger than that of SVM. If we assign a fixed value as the estimated error bound (i.e., use a fixed objective value as stopping criteria), HSVM has a faster convergence rate than does SVM. Although HSVM is more complex than SVM, the convergence speed of HSVM is not slower than that of the standard SVM. In this study, we use a simple gradient descent method to update nominal values. We can consider other fast methods such as the stochastic gradient method [43].

Table 6
Analysis of Iterations.

Dataset	Number of SVM iteration	Convergence value of SVM	Number of HSVM iteration	Convergence value of HSVM
tic-tac-toe	4	0.64	11	0.20
monk-1	5	0.79	7	0.05
monk-2	3	0.71	8	0.13
monk-3	2	0.45	14	0.04
kr-vs-kp	3	0.37	14	0.17
heart	4	0.61	6	0.54
hepatitis	3	0.49	13	0.35
sick	4	0.20	6	0.16
adult	2	0.60	11	0.47
credit	3	0.51	8	0.45
tae	5	0.84	6	0.82
average	**3.45**	**0.57**	**9.45**	**0.31**

6. Conclusion and future work

Classification and regression tasks are usually described by numerical and nominal attributes in many real world applications. The classical SVMs do not consider the difference between these two types of attributes. In this study, we designed a learning algorithm to deal with heterogeneous data. Our method learns a mapping from nominal attributes for use in a real space by minimizing the generalization error *radius margin*. Experiments were conducted to compare the performances of different learning algorithms and we arrived at the following conclusions:

1. Although it is common to code nominal attributes with integers and then consider them as numerical, this is not always the best solution. Nominal and numerical attributes should generally be handled separately.
2. After mapping nominal attributes into a real space, we can improve the performance of SVM by a reasonable computational cost.
3. HSVM produces quality performance in managing imbalanced tasks. This superiority arises from the fact that HSVM changes the distribution of samples in mapping nominal attributes. The different nominal attribute values may be converted to the same value and thus, the issue of imbalance will be reduced.

The proposed HSVM can also be applied to regression tasks, where the optimization objective and learning algorithm must nonetheless be reconsidered. We will examine this application in a future study.

Conflict of Interest

We declare that we have no conflict of interest.

Acknowledgments

This work is partly supported by National Program on Key Basic Research Project under Grant 2013CB329304, National Natural Science Foundation of China under Grants 61222210 and 61432011, and Qinghua Hu is supported by New Century Excellent Talents in University under Grant NCET-12-0399.

References

[1] V. Vapnik, The Nature of Statistical Learning Theory, Springer, Berlin, Germany, 1999.
[2] X.-X. Niu, C.Y. Suen, A novel hybrid cnn-svm classifier for recognizing handwritten digits, Pattern Recognit. 45 (4) (2012) 1318–1325.
[3] C.-L. Liu, K. Nakashima, H. Sako, H. Fujisawa, Handwritten digit recognition: benchmarking of state-of-the-art techniques, Pattern Recognit. 36 (10) (2003) 2271–2285.
[4] R. Huerta, S. Vembu, M.K. Muezzinoglu, A. Vergara, Dynamical svm for time series classification, in: Pattern Recognition, Springer, Berlin, Germany, 2012, pp. 216–225.
[5] T. Warren Liao, Clustering of time series dataa survey, Pattern Recognit. 38 (11) (2005) 1857–1874.
[6] E. Tapia, P. Bulacio, L. Angelone, Sparse and stable gene selection with consensus svm-rfe, Pattern Recognit. Lett. 33 (2) (2012) 164–172.
[7] R. Ruiz, J.C. Riquelme, J.S. Aguilar-Ruiz, Incremental wrapper-based gene selection from microarray data for cancer classification, Pattern Recognit. 39 (12) (2006) 2383–2392.
[8] J. Cheng, K. Wang, Active learning for image retrieval with co-svm, Pattern Recognit. 40 (1) (2007) 330–334.
[9] Y. Liu, D. Zhang, G. Lu, W.-Y. Ma, A survey of content-based image retrieval with high-level semantics, Pattern Recognit. 40 (1) (2007) 262–282.
[10] R. Kachouri, K. Djemal, H. Maaref, Multi-model classification method in heterogeneous image databases, Pattern Recognit. 43 (12) (2010) 4077–4088.
[11] C.-W. Hsu, C.-C. Chang, C.-J. Lin, et al., A Practical Guide to Support Vector Classification, 2003.
[12] J.R. Quinlan, C4. 5: Programs for Machine Learning, vol. 1, Morgan Kaufmann, Massachusetts, America, 1993.
[13] L. Breiman, J. Friedman, C.J. Stone, R.A. Olshen, Classification and Regression Trees, Chapman & Hall/CRC, London, England, 1984.
[14] D. Lin, An information-theoretic definition of similarity, in: Proceedings of the 15th International Conference on Machine Learning, San Francisco, vol. 1, 1998, pp. 296–304.
[15] C. Stanfill, D. Waltz, Toward memory-based reasoning, Commun. ACM 29 (12) (1986) 1213–1228.
[16] D.R. Wilson, T.R. Martinez, Improved heterogeneous distance functions, J. Artif. Intell. Res. 6 (1997) 1–34.
[17] E. Fix, J.L. Hodges Jr., Discriminatory analysis. nonparametric discrimination: consistency properties, Int. Stat. Rev./Rev. Int. Stat. (1989) 238–247.
[18] P.-N. Tan, et al., Introduction to Data Mining, Pearson Education, India, 2007.
[19] V. Cheng, C.-H. Li, J.T. Kwok, C.-K. Li, Dissimilarity learning for nominal data, Pattern Recognit. 37 (7) (2004) 1471–1477.
[20] O. Chapelle, V. Vapnik, O. Bousquet, S. Mukherjee, Choosing multiple parameters for support vector machines, Mach. Learn. 46 (1) (2002) 131–159.
[21] C.J. Burges, A tutorial on support vector machines for pattern recognition, Data Min. Knowl. Discov. 2 (2) (1998) 121–167.
[22] O. Chapelle, V. Vapnik, Model selection for support vector machines, in: Advances in Neural Information Processing Systems, vol. 12, 1999, pp. 230–236.
[23] I.W. Tsang, J.T. Kwok, P.-M. Cheung, Core vector machines: fast svm training on very large data sets, J. Mach. Learn. Res. 6 (1) (2006) 363.
[24] R.C. Barros, M.P. Basgalupp, A. de Carvalho, A.A. Freitas, A survey of evolutionary algorithms for decision-tree induction, IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. 42 (3) (2012) 291–312.
[25] X. Wu, V. Kumar, J. Ross Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, P.S. Yu, et al., Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2008) 1–37.
[26] J. Gama, P. Brazdil, Cascade generalization, Mach. Learn. 41 (3) (2000) 315–343.
[27] J. Maudes, J. J. Rodríguez, C. García-Osorio, Cascading for nominal data, in: Multiple Classifier Systems, Springer, Berlin, Germany, 2007, pp. 231–240.
[28] Y. Tian, N. Deng, Support vector classification with nominal attributes, in: Computational Intelligence and Security, Springer, Berlin, Germany, 2005, pp. 586–591.
[29] H. Do, A. Kalousis, A. Woznica, M. Hilario, Margin and radius based multiple kernel learning, in: Machine Learning and Knowledge Discovery in Databases, Springer, Berlin, Germany, 2009, pp. 330–343.
[30] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, Cambridge, England, 2000.
[31] A. Asuncion, D.J. Newman, Uci Machine Learning Repository, 2007.
[32] R. Rifkin, A. Klautau, In defense of one-vs-all classification, J. Mach. Learn. Res. 5 (2004) 101–141.
[33] S. Li, M. Tan, Tuning svm parameters by using a hybrid clpso-bfgs algorithm, Neurocomputing 73 (10) (2010) 2089–2096.
[34] H. Frohlich, A. Zell, Efficient parameter selection for support vector machines in classification and regression via model-based global optimization, in: IJCNN'05. Proceedings of 2005 IEEE International Joint Conference on Neural Networks, vol. 3, IEEE, New Jersey, America, 2005, pp. 1431–1436.
[35] J. Bergstra, Y. Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2012) 281–305.
[36] G.E. Batista, D.F. Silva, How k-nearest neighbor parameters affect its performance, in: Argentine Symposium on Artificial Intelligence, 2009, pp. 1–12.
[37] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The weka data mining software: an update, ACM SIGKDD Explor. Newslett. 11 (1) (2009) 10–18.
[38] G. David, A. Averbuch, Spectralcat: categorical spectral clustering of numerical and nominal data, Pattern Recognit. 45 (1) (2012) 416–433.
[39] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[40] H. He, E.A. Garcia, Learning from imbalanced data, IEEE Trans. Knowl. Data Eng. 21 (9) (2009) 1263–1284.
[41] B.X. Wang, N. Japkowicz, Boosting support vector machines for imbalanced data sets, Knowl. Inf. Syst. 25 (1) (2010) 1–20.
[42] F. Provost, Machine learning from imbalanced data sets 101, in: Proceedings of the AAAI2000 Workshop on Imbalanced Data Sets, 2000.
[43] M. Takác, A. Bijral, P. Richtárik, N. Srebro, Mini-batch primal and dual methods for svms, in: 30th International Conference on Machine Learning, 2013.

Shili Peng got his M.E. from Central South University in 2005 and now he is a Ph.D. candidate with School of Computer Science and Technology, Tianjin University. He started working with Guangdong University of Finance from 2005. In 2011, he was a visiting scholar at Freiberg University of Mining and Technology, Germany. His research interests include machine learning, data mining, optimization theory, etc.

Qinghua Hu received his B.Sc., M.E. and Ph.D. degrees from Harbin Institute of Technology, Harbin, China in 1999, 2002 and 2008, respectively. He started working with Harbin Institute of Technology from 2006, and was a post-doctoral fellow with the Hong Kong Polytechnic University from 2009 to 2011. In 2012, he joined Tianjin University. Now he is a full Professor with School of Computer Science and Technology, Tianjin University and the Head of Department of Computer and Information Technology and the Leader of Lab of Pattern Analysis and Computational Intelligence (PANDIT). His research interests are focused on Pattern Analysis from data, machine learning for classification and regression. He is a PC co-chair of RSCTC 2010, CRSSC 2012 and ICMLC 2014. He also severs as referee for a great number of journals and conferences. He has published more than 100 journal and conference papers in the areas of pattern recognition and machine learning.

Yinli Chen received his B.Eng. degree from Nankai University, Tianjin, China, 1985 and M.E. degree from Institute of Computer Technology, Chinese Academy of Sciences, Beijing, China in 1988. He started working with Huazhong University of Science and Technology from 1988 to 1992. In 1992, he joined Guangzhou University of Finance. Now he is a full Professor with Department of Computer Science and Technology, GDUF and the Head of Department of Computer Science and Technology. His research interests are focused on Pattern Analysis from data, machine learning for classification and regression in financial market. He has published several journal and conference papers in the areas of pattern recognition.

Prof. Jianwu Dang graduated from Tsinghua Univ., China, in 1982, and got his M.S. degree at the same university in 1984. He worked for Tianjin University as a lecture from 1984 to 1988. He was awarded the Ph.D. degree from Shizuoka University Japan in 1992. He worked for ATR Human Information Processing Laboratories, Japan, as a senior researcher from 1992 to 2001. He joined the University of Waterloo, Canada, as a visiting scholar for one year from 1998. Since 2001, he has worked for Japan Advanced Institute of Science and Technology (JAIST) as a professor. He joined the Institute of Communication Parlee (ICP), Center of National Research Scientific, France, as a research scientist the first class from 2002 to 2003. Since 2009, he has joined Tianjin University, Tianjin, China. He was awarded the scholar of National One-Thousand Talents Plan, and also the principle scientist of the National Basic Research Program (973 Program). His research interests are in all the fields of speech production, speech synthesis, speech recognition, and speaker identification. He built MRI-based physiological models for speech and swallowing, and endeavors to apply these models on clinics.