Computational Statistics and Data Analysis 54 (2010) 1247–1265

Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

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

Conversion of categorical variables into numerical variables via Bayesian network classifiers for binary classifications

Namgil Lee [a], Jong-Min Kim [b,*]

[a] Department of Mathematical Sciences, Korea Advanced Institute of Science and Technology, Daejeon 305-701, Republic of Korea

[b] Statistics Discipline, Division of Science and Mathematics, University of Minnesota-Morris, Morris, MN 56267, USA

ARTICLE INFO

Article history:
Received 10 September 2008
Received in revised form 4 November 2009
Accepted 5 November 2009
Available online 18 November 2009

ABSTRACT

Many pattern classification algorithms such as Support Vector Machines (SVMs), Multi-Layer Perceptrons (MLPs), and K-Nearest Neighbors (KNNs) require data to consist of purely numerical variables. However many real world data consist of both categorical and numerical variables. In this paper we suggest an effective method of converting the mixed data of categorical and numerical variables into data of purely numerical variables for binary classifications. Since the suggested method is based on the theory of learning Bayesian Network Classifiers (BNCs), it is computationally efficient and robust to noises and data losses. Also the suggested method is expected to extract sufficient information for estimating a minimum-error-rate (MER) classifier. Simulations on artificial data sets and real world data sets are conducted to demonstrate the competitiveness of the suggested method when the number of values in each categorical variable is large and BNCs accurately model the data.

1. Introduction

The primary goal of pattern classification is to estimate a classification function, i.e., a classifier, using labeled training patterns so that the estimated classifier will correctly assign class labels to novel test patterns. Some examples of the most widely used classification algorithms are Support Vector Machines (SVMs), Multi-Layer Perceptrons (MLPs), and K-Nearest Neighbors (KNNs). SVMs (Vapnik, 1995; Burges, 1998; Cristianini and Shawe-Taylor, 2000) build a sparsely formulated hyperplane classifier through the maximization of a margin criterion. MLPs (Bishop, 1995; Haykin, 1999) construct networks with two or more layers of nonlinear computation units, and the synaptic weights of each unit are estimated by a maximum likelihood estimation. KNNs (Duda et al., 2001) make a rule which classify a pattern by assigning it the label most common among its k nearest samples.

Many classification algorithms including the above examples assume that a pattern is represented as a vector of numerical values. For example, the common basic operations of those algorithms are the computations of dot products and Euclidean distances between patterns and other vectors. However, in many real world data sets a pattern is represented as a collection of discrete or structured objects. For example, a text is represented as a string of letters, gene as a sequence of nucleotides, image as a set of pixels, and so on.

In this paper we concentrate on the case that a pattern is represented as a collection of only two types of values: categorical values and numerical values. That is, each of the variables in a pattern is of either categorical type or numerical type, and a categorical variable takes its values in some finite set of categories. In this case the classification algorithms

* Corresponding author.
E-mail addresses: namgil@kaist.ac.kr (N. Lee), jongmink@morris.umn.edu (J.-M. Kim).

0167-9473/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2009.11.003

such as SVMs, MLPs, and KNNs are not directly applicable, and one might have to either discard the categorical values or convert the categorical values into numerical values. One typical conversion method is to use a single number to represent a categorical value. But this method depends on an arbitrary ordering of values in a categorical variable. Alternatively, Hsu et al. (2003) suggest to use m binary numbers to represent a m-category variable. Hsu et al. (2003) remark that if there are not too many values in a categorical variable, the method is more stable than using a single number to represent a categorical variable.

On the other hand, there have been many researches on designing kernel functions for various structured data (Gärtner, 2003; Shawe-Taylor and Cristianini, 2004). A kernel function is a measure of meaningful similarities between a pair of patterns (Schölkopf and Smola, 2002, Ch.2), and appropriately selected kernel functions have led to improvements in classification performances (Vapnik, 1995; Joachims, 1998; Chapelle et al., 1999; Pavlidis et al., 2002). The Fisher kernel (Jaakkola and Haussler, 1999) and the marginalized kernel (Tsuda et al., 2002) are typical kernels defined from probabilistic models such as Hidden Markov Models (HMMs), and both of them have achieved remarkable improvements in biological sequence classifications. This implies that defining a kernel on a probabilistic model is a useful way of incorporating prior knowledge and manipulating structured data.

In this paper we propose a new method of converting mixed data of categorical and numerical values into data of numerical values by defining a kernel function from a probabilistic model. First we define an ideal kernel function for binary classification problems based on the definition of a minimum-error-rate (MER) classifier. Second we propose to use Bayesian Network Classifiers (BNCs) (Friedman et al., 1997) to accurately estimate the ideal kernel function. The estimation using BNCs allows an effective modeling of the categorical variables, and it is computationally efficient and robust to noises and data losses. Third we show that the ideal kernel function is decomposed into products of simpler kernel functions. This decomposition enables us to explicitly present the conversion of original mixed data into numerical data. Since the suggested method uses a small number of real numbers to represent a categorical value, there is not much increases in dimensions of patterns regardless of the number of values in a categorical variable. Moreover a simple linear classifier can approximate the MER classifier using the converted numerical values as far as the estimation by BNCs is accurate.

This paper is organized as follows. In Section 2 we describe the mixed data of categorical and numerical variables, and we introduce basic properties of a kernel function. In Section 3 we define the ideal kernel and the MER classifier. In Section 4 the estimation of probabilities for the mixed data using BNCs is explained. In Section 5 the decomposition of the ideal kernel is described, and the explicit conversion of the mixed data into numerical data is proposed. In Section 6 we present simulation results on artificial data sets and real world data sets comparing the suggested method with the other typical methods. We discuss about the results and future researches in Section 7.

2. Backgrounds

2.1. Mixed data of categorical and numerical variables

In this paper we suppose that an input pattern is a collection of categorical and numerical values. We denote the jth categorical variable of the ith input pattern by $x_j^{(i)}$, and the jth numerical variable of the ith input pattern by $r_j^{(i)}$. Then the ith input pattern $x^{(i)}$ is represented as a vector by

$$x^{(i)} = \left(x_1^{(i)}, \ldots, x_n^{(i)}, r_1^{(i)}, \ldots, r_m^{(i)}\right), \quad i = 1, \ldots, N. \tag{1}$$

Each categorical variable $x_j^{(i)}$ takes its value from a finite set of discrete categories, which is denoted by

$$x_j^{(i)} \in V_j = \left\{v_{j,1}, \ldots, v_{j,n_j}\right\} \tag{2}$$

where $v_{j,q}$, $q = 1, \ldots, n_j$, represents the qth categorical value that the jth categorical variable can take.

Categorical variables are abundant in real world data. For example, variables that take values from {true, false} or {yes, no} are called binary attributes, and variables that take values from any collection of words which describe characteristics of an instance are called nominal attributes. For example, {Bisexual, Homosexual, Heterosexual} is a set of words characterizing sexual orientation of a person, and {United States, Cambodia, England, …} is a set of words identifying the country of residence of a person.

However, when we want to apply the classification algorithms such as SVMs, MLPs, and KNNs to data with categorical variables, one has to either discard the categorical variables or convert them into numerical variables. The following are the two typical methods for converting categorical variables into numerical variables:

1. Single number representation: The discrete values of a categorical variable are converted into discrete real numbers. For instance, the discrete values {Bisexual, Homosexual, Heterosexual} are converted into {1, 2, 3} in order. But this conversion depends on an arbitrary ordering of values in a categorical variable, so it can result in unreliable performances.
2. Binary number representation: The n_j discrete values of a categorical variable are encoded into binary numbers of length n_j. For instance, {Bisexual, Homosexual, Heterosexual} is converted into {001, 010, 100}. Hsu et al. (2003) make a remark that this representation is more stable than the single number representation. However, if the number of categories for

each categorical variable is too large, the dimension of input patterns must be greatly increased, which will result in poor classification performances and high computational costs.

However, rather than to rely on human experts in converting categorical values into numerical values, it is more reasonable to estimate the conversions through data. Moreover, it is a good idea to deal with the conversion within the framework of kernels, because a kernel also implicitly conducts a transformation of input patterns into a feature space before processing the data.

2.2. Properties of a Kernel function

In the two class pattern classification tasks, a classifier is a function f from a set of input patterns $\mathcal{X}$ into the set of binary labels $\{-1, +1\}$. A classification algorithm takes a set of labeled training patterns

$$(x^{(1)}, y^{(1)}), \ldots, (x^{(N)}, y^{(N)}) \in \mathcal{X} \times \{-1, +1\} \tag{3}$$

and infer a classifier. In order to avoid overfitting and reduce computational costs, classification algorithms often restrict the class of functions from which f is chosen to a subset of all classifiers, which is called a hypothesis space. The hypothesis space that SVMs consider is, in the case that $\mathcal{X}$ is a dot product space, the class of linear decision functions:

$$f(x) = \Theta(\langle w, x\rangle_{\mathcal{X}} + b) \tag{4}$$

where Θ is a step function defined as $\{\Theta(t) = +1\ \forall t \geq 0;\ \Theta(t) = -1\ \forall t < 0\}$, $\langle\cdot,\cdot\rangle_{\mathcal{X}}$ is the dot product in $\mathcal{X}$, and $w \in \mathcal{X}$ and $b \in \mathrm{R}$ are parameters.

However, the relations between class labels and input patterns are often nonlinear, and the space $\mathcal{X}$ may not be a dot product space. To deal with these problems, kernel based classification algorithms such as SVMs use a map $\phi : \mathcal{X} \to \mathcal{H}$ to transform all the patterns in $\mathcal{X}$ into vectors in a (possibly infinite dimensional) dot product space $\mathcal{H}$. As a result the classifiers selected by SVMs by using the training data (3) are written in the following form:

$$f(x) = \Theta\left(\sum_{i=1}^{N} w^{(i)} \langle \phi(x^{(i)}), \phi(x)\rangle_{\mathcal{H}} + b\right) \tag{5}$$

where $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ is the dot product in the space $\mathcal{H}$, and $w^{(1)}, \ldots, w^{(N)} \in \mathrm{R}$ and $b \in \mathrm{R}$ are the selected parameters.

To classify patterns with the classifier (5), we need to compute dot products in $\mathcal{H}$. Since $\mathcal{H}$ can be infinite dimensional, it is not easy to compute each dot product with ϕ in explicit form. However, if there exists a function k such that $k(x^{(i)}, x) = \langle \phi(x^{(i)}), \phi(x)\rangle_{\mathcal{H}}$, the dot products can be computed more efficiently as a direct function of the input patterns. We say a function k is a *kernel* if for a dot product space $\mathcal{H}$ and a map $\phi : \mathcal{X} \to \mathcal{H}$ it satisfies

$$k(x, x') = \langle \phi(x), \phi(x')\rangle_{\mathcal{H}} \tag{6}$$

for all $x, x' \in \mathcal{X}$. Consequently the classifier (5) with a kernel (6) becomes

$$f(x) = \Theta\left(\sum_{i=1}^{N} w^{(i)} k(x^{(i)}, x) + b\right). \tag{7}$$

We may want to create a kernel without explicitly constructing the map ϕ and the space $\mathcal{H}$. The following theorem explains the characteristics of a valid kernel. See Shawe-Taylor and Cristianini (2004, Ch.3) for the proof.

Theorem 1. *Let $\mathcal{X}$ be a nonempty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathrm{R}$ is a kernel if and only if it is*

(i) *symmetric, i.e., for any x and $x' \in \mathcal{X}$, $k(x, x') = k(x', x)$, and,*
(ii) *finitely positive semi-definite, i.e., for any $N \geq 1$ and $x^{(1)}, \ldots, x^{(N)} \in \mathcal{X}$ the matrix K with elements $K_{ij} = k(x^{(i)}, x^{(j)})$ is positive semi-definite, that is, $\sum_{i=1}^{N}\sum_{j=1}^{N} a^{(i)}a^{(j)}K_{ij} \geq 0$ for all $a^{(1)}, \ldots, a^{(N)} \in \mathrm{R}$.*

Moreover, many kernels are built by combining a number of other kernels, e.g., the convolution kernel (Haussler, 1999) and the string subsequence kernel (Lodhi et al., 2002). This is based on the following closure properties of the class of kernels. See Shawe-Taylor and Cristianini (2004, Ch.3) for the proof.

Proposition 2 (*Closure Properties*). *Let k_1 and k_2 be kernels over $\mathcal{X} \times \mathcal{X}$, $a > 0$, $g(\cdot)$ a real valued function on $\mathcal{X}$, and $h(\cdot)$ a polynomial with positive coefficients. Then the following functions are kernels:*

(i) $k(x, x') = k_1(x, x') + k_2(x, x')$,
(ii) $k(x, x') = ak_1(x, x')$,
(iii) $k(x, x') = k_1(x, x')k_2(x, x')$,
(iv) $k(x, x') = g(x)g(x')$,
(v) $k(x, x') = h(k_1(x, x'))$,
(vi) $k(x, x') = \exp(k_1(x, x'))$.

3. Ideal Kernel for classification

For evaluating the quality of a classifier, a proper criterion is required. In statistical learning theory framework, we assume that all the pairs $(x, y) \in \mathcal{X} \times \{-1, +1\}$ are independently generated from some unknown probability distribution $p(x, y)$. Given a pair (x, y), suppose $\hat{y}$ is the decision made by a classifier f, that is, $\hat{y} = f(x)$. Then the decision is correct if $\hat{y} = y$ and incorrect if $\hat{y} \neq y$. The indicator function of the occurrence of an error is called the zero–one loss function, which is defined by

$$l(y, \hat{y}) = 0, \quad \hat{y} = y; \qquad l(y, \hat{y}) = 1, \quad \hat{y} \neq y. \tag{8}$$

Then, given a test pattern x, the probability of error for the decision made by a classifier f is the conditional expectation of the zero–one loss function, which is calculated as

$$\begin{aligned} \mathrm{E}(l(y, \hat{y})|x) &= \sum_{y \in \{-1,+1\}} l(y, \hat{y}) p(y|x) && (9) \\ &= 1 - p(y = \hat{y}|x). && (10) \end{aligned}$$

An optimal classifier makes a decision that minimizes the probability of error (10), that is, $\hat{y} = +1$ if $p(y = +1|x) \geq p(y = -1|x)$, and $\hat{y} = -1$ if $p(y = +1|x) < p(y = -1|x)$. So we define the *minimum-error-rate (MER) classifier* as follows.

Definition 1 (*MER Classifier*). The minimum-error-rate (MER) classifier f_{I} is defined by

$$\begin{aligned} f_{\mathrm{I}}(x) &= \Theta\left(\log p(y = +1|x) - \log p(y = -1|x)\right) \\ &= \Theta\left(\log \frac{p(y = +1|x)}{p(y = -1|x)}\right) \end{aligned} \tag{11}$$

where the log represents the natural logarithm.

The logarithms in the definition make the analysis easier later on.

Generally a kernel determines similarities between two input patterns. If the similarity measure is perfect, then it would contain the information on which classes the given input patterns belong to. That is, a larger value is taken between the patterns in the same class, and a smaller value is taken between the patterns in the different classes. Therefore, if the class labels $y, y' \in \{+1, -1\}$ of the two patterns $x, x' \in \mathcal{X}$ are known beforehand, it would always be possible to construct a perfect kernel as follows:

$$k^*(x, x') = yy' \tag{12}$$

so that $k^*(x, x') = 1$ if $y = y'$ and $k^*(x, x') = -1$ if $y \neq y'$. Then it becomes a trivial task to construct a kernel classifier such as (7) with perfect classification performances using this kernel.

However the true class labels are unknown for test data. Instead, if an effective method for estimating the posterior probability $p(y|x)$ is given, we can define the *ideal kernel* based on the definition of the MER classifier in a similar way to (12).

Definition 2 (*Ideal kernel*). The *ideal kernel* k_{I} for binary classifications is defined by

$$k_{\mathrm{I}}(x, x') = \frac{p(y = +1|x)}{p(y = -1|x)} \cdot \frac{p(y' = +1|x')}{p(y' = -1|x')}. \tag{13}$$

The function k_{I} defined above is shown to be a kernel by Proposition 2(iv). And a kernel classifier such as (7) using the ideal kernel can easily approximate the MER classifier f_{I} since

$$\begin{aligned} f(x) &= \Theta\left(\sum_{i=1}^{N} w^{(i)} k_{\mathrm{I}}(x^{(i)}, x) + b\right) \\ &= \Theta\left(C \cdot \frac{p(y = +1|x)}{p(y = -1|x)} + b\right) \end{aligned} \tag{14}$$

where $C = \sum_{i=1}^{N} w^{(i)} \cdot \frac{p(y^{(i)}=+1|x^{(i)})}{p(y^{(i)}=-1|x^{(i)})}$, and $f(x)$ becomes exactly same to $f_{\mathrm{I}}(x)$ whenever $C > 0$ and $b = -C$.

4. Bayesian network classifiers for mixed data

To effectively determine the posterior probability $p(y|x)$ from the given training data set, we suggest to use the so-called *Bayesian network classifiers* (BNCs) (Friedman et al., 1997). The BNCs are special cases of Bayesian networks.

4.1. Bayesian networks

A *Bayesian network* (BN) (Pearl, 1988) is a directed acyclic graph that encodes a joint probability distribution over a set of random variables. We denote the set of random variables by $Z = \{z_1, \ldots, z_{d+1}\} = \{x_1, \ldots, x_d, y\}$, and we suppose each random variable is a categorical variable. Formally a BN for Z is defined by a pair $B = \langle G, \Theta \rangle$, where G is a directed acyclic graph and Θ is a set of parameters. Each node in the graph represents each random variable. The edge connections represent independence properties of the random variables, i.e., each variable z_j is independent of its non-descendants given its parents in G. We denote the joint probability distribution encoded by a BN B by $p_B(z_1, \ldots, z_{d+1})$, and the set of random variables corresponding to the parent nodes of z_j by $pa(z_j)$. Then each parameter $\theta_{z_j|pa(z_j)} \in \Theta$ represents the conditional probability $p_B(z_j|pa(z_j))$. Then $p_B(z_1, \ldots, z_{d+1})$ is determined in the factorized form by

$$p_B(z_1, \ldots, z_{d+1}) = \prod_{j=1}^{d+1} p_B(z_j|pa(z_j)) = \prod_{j=1}^{d+1} \theta_{z_j|pa(z_j)}. \tag{15}$$

4.2. Bayesian network classifiers

Given a set of training data $D = \{Z^{(1)}, \ldots, Z^{(N)}\}$ the goal of learning a BN B is to find a $\langle G, \Theta \rangle$ that accurately models the true joint distribution $p(z_1, \ldots, z_{d+1})$. The commonly used scoring function to evaluate BNs is the minimum description length (MDL) score (Lam and Bacchus, 1994), which is defined by

$$MDL(B|D) = \frac{\log N}{2}|B| - LL(B|D) \tag{16}$$

where $|B|$ is the number of parameters in the BN, and $LL(B|D)$ is the log-likelihood of B given D defined by $LL(B|D) = \sum_{i=1}^{N} \log(p_B(Z^{(i)}))$.

However the goal of classifications is to correctly predict the class label y for a test pattern $(x_1, \ldots, x_d)$. Since the decision of a class label using a BN B depends on the posterior distribution $p_B(y|x_1, \ldots, x_d)$, accurately modeling the true posterior distribution $p(y|x_1, \ldots, x_d)$ is more important than modeling the joint distribution $p(x_1, \ldots, x_d, y)$. Friedman et al. (1997) indicated that the nonspecialized scoring functions such as the MDL score may result in a poor classifier when they are used in learning unrestricted Bayesian networks. Therefore Friedman et al. (1997) suggested restricting the Bayesian network structures so that the node corresponding to the class variable y is a parent node of every other nodes. This restriction ensures that all the variables in a pattern are taken into account in the decision of the label.

In this paper we call this kind of restricted Bayesian networks as the Bayesian network classifiers (BNCs). Some examples of the BNCs are naive Bayesian classifiers (NBs) and tree-augmented naive Bayesian classifiers (TANs), and they are highly competitive in various classification tasks (Friedman et al., 1997; Leray and Francois, 2004). First, the NBs assume that all the variables in a pattern are conditionally independent given the value of the class label. The performance of the NBs is surprisingly good despite of the above unrealistic assumption (Friedman et al., 1997). The structure of the NBs is illustrated in Fig. 1 (a). The joint probability distribution of NBs is simplified as

$$p_B(x_1, \ldots, x_d, y) = p_B(y) \prod_{j=1}^{d} p_B(x_j|y) = \theta_y \prod_{j=1}^{d} \theta_{x_j|y}. \tag{17}$$

Second, the TANs are a structural extension of the NBs, where additional edges are allowed in order to relax the strong assumption of the conditional independence. The additional edges are constructed from a maximal weighted spanning tree with variables in an input pattern as vertices. So each variable x_j in an input pattern has the class variable y as parents and at most one other variable denoted by $x_{\pi(j)}$. The structure of a TAN is illustrated in Fig. 1 (b). The joint probability distribution of TANs is simplified as

$$p_B(x_1, \ldots, x_d, y) = p_B(y) \prod_{j=1}^{d} p_B(x_j|x_{\pi(j)}, y) = \theta_y \prod_{j=1}^{d} \theta_{x_j|x_{\pi(j)},y}. \tag{18}$$

4.3. Smoothed parameter estimation for categorical variables

In this paper we use the Bayesian approach (Heckerman, 1995) to the estimation of parameters of a BNC. This technique can be understood as a smoothing operation on the parameters. While the maximum likelihood estimates tend to be unreliable when the size of training data is not enough, the smoothing operation works well in such situations (Friedman et al., 1997).

In the Bayesian approach, we suppose that each conditional probability $p_B(x_j|pa(x_j))$ has a parametric form with a set of parameters $\vec{\theta}_{j,pa(x_j)}$. Among the possible choices for the parametric probability models, the *multinomial* distribution is

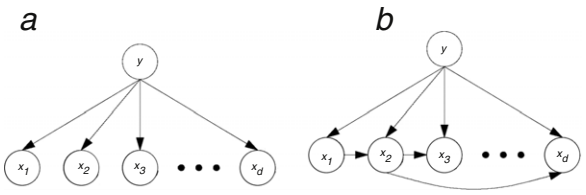

Fig. 1. Structures of the Bayesian network classifiers. (a) naive Bayesian classifiers, and (b) tree-augmented naive Bayesian classifiers.

most appropriate in modeling a categorical variable (Johnson et al., 1997, p. 69). In this case, each parameter represents the conditional probability that the variable x_j will take a categorical value, that is,

$$\vec{\theta}_{j,pa(x_j)} = \left\{\theta_{v_{j,q}|pa(x_j)}\right\}_{q=2}^{n_j}, \qquad \theta_{v_{j,1}|pa(x_j)} = 1 - \sum_{q=2}^{n_j} \theta_{v_{j,q}|pa(x_j)} \tag{19}$$

and

$$p_B(x_j = v_{j,q}|pa(x_j)) = \theta_{v_{j,q}|pa(x_j)} > 0, \quad q = 1, \ldots, n_j. \tag{20}$$

Note that the above parameters coincide with the parameters of BNs. Our uncertainties about the optimal parameter values can be expressed using a probability density function over the parameters, $p(\vec{\theta}_{j,pa(x_j)}|pa(x_j))$. We assume that the set of parameters $\vec{\theta}_{j,pa(x_j)}$ follows the Dirichlet distribution, which is a *conjugate prior* of the multinomial distribution. That is,

$$p(\vec{\theta}_{j,pa(x_j)}|pa(x_j)) = \mathrm{Dir}\left(\vec{\theta}_{j,pa(x_j)}\middle|\alpha_{pa(x_j),1}, \ldots, \alpha_{pa(x_j),n_j}\right) \tag{21}$$

where $\alpha_{pa(x_j),q}$, $q = 1, \ldots, n_j$, are hyperparameters. We usually initialize $\alpha_{pa(x_j),q} = 1, q = 1, \ldots, n_j$, which implies the uniform distribution. Then, assuming that all the instances in the training data are statistically independent, the *posterior* probability of $\vec{\theta}_{j,pa(x_j)}$ given training data D is computed using Bayes' rule, which is again the Dirichlet distribution with updated parameters as

$$p(\vec{\theta}_{j,pa(x_j)}|D, pa(x_j)) = \mathrm{Dir}\left(\vec{\theta}_{j,pa(x_j)}\middle|\tilde{\alpha}_{pa(x_j),1}, \ldots, \tilde{\alpha}_{pa(x_j),n_j}\right) \tag{22}$$

where

$$\tilde{\alpha}_{pa(x_j),q} = \alpha_{pa(x_j),q} + N_{pa(x_j),q}, \qquad q = 1, \ldots, n_j, \tag{23}$$

and $N_{pa(x_j),q}$ is the number of instances in the training data which contains the values $v_{j,q}$ and $pa(x_j)$.

Finally, the parameters are determined as expected values with respect to the posterior probability, that is,

$$\begin{aligned}\hat{\theta}_{v_{j,q}|pa(x_j)} &= \int \theta_{v_{j,q}|pa(x_j)} p(\vec{\theta}_{j,pa(x_j)}|D, pa(x_j))\mathrm{d}\vec{\theta}_{j,pa(x_j)} \\ &= \frac{\alpha_{pa(x_j),q} + N_{pa(x_j),q}}{\alpha_{pa(x_j),*} + N_{pa(x_j),*}}, \quad q = 1, \ldots, n_j.\end{aligned} \tag{24}$$

where $\alpha_{pa(x_j),*} = \sum_q \alpha_{pa(x_j),q}$ and $N_{pa(x_j),*} = \sum_q N_{pa(x_j),q}$. Especially if $\alpha_{pa(x_j),q} = 1, q = 1, \ldots, n_j$, then $\alpha_{pa(x_j),*} = n_j$.

4.4. Discretization of numerical variables

In order to apply BNCs, all the variables in the data have to be categorical. Since we assume the original data is represented as a mixture of categorical and numerical values, we need to convert the numerical values into categorical values. We use the method suggested by O Colot et al. (1994) to construct an optimal *discretization*, that is, the whole range of a numerical variable is split into intervals and then each interval is converted into a categorical value. Consequently, the original data representation for the mixed data (1) is replaced with the following representation:

$$r_j^{(i)} \mapsto x_{n+j}^{(i)}, \quad j = 1, \ldots, m, \tag{25}$$

and

$$x^{(i)} = \left(x_1^{(i)}, \ldots, x_n^{(i)}, r_1^{(i)}, \ldots, r_m^{(i)}\right) \mapsto \tilde{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right), \quad i = 1, \ldots, N \tag{26}$$

where $d = n + m$ and

$$x_j^{(i)} \in V_j = \left\{v_{j,1}, \ldots, v_{j,n_j}\right\}, \quad j = 1, \ldots, d. \tag{27}$$

5. Conversion of categorical variables into numerical variables

Both of the MER classifiers (11) and the ideal kernels (13) are defined with respect to the posterior probability $p(y|x)$. So we can approximate good classifiers by accurately modeling the posterior probability. Especially we will apply the BNCs introduced in the previous section. From now on we use the following notations: for NBs with the notations in (17),

$$\psi_0^{NB} = \log \frac{\theta_{+1}}{\theta_{-1}}, \qquad \psi_j^{NB}(x_j) = \log \frac{\theta_{x_j|+1}}{\theta_{x_j|-1}},\ j = 1, \ldots, d, \tag{28}$$

and for TANs with the notations in (18),

$$\psi_0^{TAN} = \log \frac{\theta_{+1}}{\theta_{-1}}, \qquad \psi_j^{TAN}(\tilde{x}) = \log \frac{\theta_{x_j|x_{\pi(j)},+1}}{\theta_{x_j|x_{\pi(j)},-1}}, \quad j = 1, \ldots, d \tag{29}$$

where $\tilde{x} = (x_1, \ldots, x_d)$ represents the pattern obtained by discretizing the numerical variables in an original pattern x.

5.1. Decompositions

The MER classifier is decomposed as follows.

Proposition 3. *Assuming that the posterior probability $p(y|x)$ is modeled by Bayesian network classifiers, the MER classifier defined by (11) is decomposed into the following forms:*

(i) *by NBs,*

$$f_I(x) = \Theta\left(\psi_0^{NB} + \sum_{j=1}^{d} \psi_j^{NB}(x_j)\right), \tag{30}$$

and

(ii) *by TANs,*

$$f_I(x) = \Theta\left(\psi_0^{TAN} + \sum_{j=1}^{d} \psi_j^{TAN}(\tilde{x})\right). \tag{31}$$

Proof. Suppose $p_B(x_1, \ldots, x_d, y)$ denotes the joint probability distribution estimated by any BNC B for modeling the true joint probability $p(x, y)$. Then, since $p_B(x_1, \ldots, x_d, y) = p_B(x_1, \ldots, x_d)p_B(y|x_1, \ldots, x_d)$, we get

$$\begin{aligned} \log \frac{p(y=+1|x)}{p(y=-1|x)} &\approx \log \frac{p_B(y=+1|x_1, \ldots, x_d)}{p_B(y=-1|x_1, \ldots, x_d)} \\ &= \log \frac{p_B(x_1, \ldots, x_d, y=+1)}{p_B(x_1, \ldots, x_d, y=-1)}. \end{aligned}$$

Using the factorization properties of the BNCs, i.e., (17) and (18), we get the results.

In the same way, the ideal kernel is decomposed as follows.

Proposition 4. *Assuming that the posterior probability $p(y|x)$ is modeled by Bayesian network classifiers, the ideal kernel defined by (13) is decomposed into products of simpler kernels as follows:*

(i) *by NBs,*

$$k_I(x, x') = k_{I,0}^{NB} \prod_{j=1}^{d} k_{I,j}^{NB}(x_j, x'_j) \tag{32}$$

where

$$k_{I,0}^{NB} = \exp\left(\psi_0^{NB}\right) \exp\left(\psi_0^{NB}\right) \tag{33}$$

and

$$k_{I,j}^{NB}(x_j, x'_j) = \exp\left(\psi_j^{NB}(x_j)\right) \exp\left(\psi_j^{NB}(x'_j)\right), \quad j = 1, \ldots, d, \tag{34}$$

and

(ii) *by TANs,*

$$k_{\mathrm{I}}(x, x') = k_{\mathrm{I},0}^{TAN} \prod_{j=1}^{d} k_{\mathrm{I},j}^{TAN}(\tilde{x}, \tilde{x}') \tag{35}$$

where

$$k_{\mathrm{I},0}^{TAN} = \exp\left(\psi_0^{TAN}\right) \exp\left(\psi_0^{TAN}\right) \tag{36}$$

and

$$k_{\mathrm{I},j}^{TAN}(\tilde{x}, \tilde{x}') = \exp\left(\psi_j^{TAN}(\tilde{x})\right) \exp\left(\psi_j^{TAN}(\tilde{x}')\right), \quad j = 1, \ldots, d. \tag{37}$$

5.2. Conversions

From Proposition 3, we conclude that the set of component functions $\{\psi_j\}$, which denotes either of $\{\psi_j^{NB}\}$ or $\{\psi_j^{TAN}\}$, can be used to approximate the MER classifier through a simple linear function.

The basic idea for understanding Proposition 4 is that a kernel is represented as a dot product between two feature vectors in the corresponding dot product space as (6). Let $\phi_{\mathrm{I},j}$ and $\mathcal{H}_{\mathrm{I},j}$ be the feature map and the dot product space respectively corresponding to each component kernel $k_{\mathrm{I},j}$, that is, $k_{\mathrm{I},j}(\tilde{x}, \tilde{x}') = \langle \phi_{\mathrm{I},j}(\tilde{x}), \phi_{\mathrm{I},j}(\tilde{x}') \rangle_{\mathcal{H}_{\mathrm{I},j}}$, $j = 1, \ldots, d$. And let ϕ_{I} and $\mathcal{H}_{\mathrm{I}}$ be the feature map and the dot product space respectively corresponding to the ideal kernel k_{I}, that is, $k_{\mathrm{I}}(x, x') = \langle \phi_{\mathrm{I}}(x), \phi_{\mathrm{I}}(x') \rangle_{\mathcal{H}_{\mathrm{I}}}$. Then the decompositions of the ideal kernel in Proposition 4 are rewritten as

$$\langle \phi_{\mathrm{I}}(x), \phi_{\mathrm{I}}(x') \rangle_{\mathcal{H}_{\mathrm{I}}} = k_{\mathrm{I},0} \prod_{j=1}^{d} \langle \phi_{\mathrm{I},j}(\tilde{x}), \phi_{\mathrm{I},j}(\tilde{x}') \rangle_{\mathcal{H}_{\mathrm{I},j}} \tag{38}$$

where $k_{\mathrm{I},0} > 0$. This representation implies that the dot product in the space $\mathcal{H}_{\mathrm{I}}$ is defined by the products of the dot products in the spaces $\mathcal{H}_{\mathrm{I},j}$, $j = 1, \ldots, d$. This implies that the space $\mathcal{H}_{\mathrm{I}}$ is defined by Cartesian products of the component spaces $\mathcal{H}_{\mathrm{I},j}$, $j = 1, \ldots, d$, that is,

$$\mathcal{H}_{\mathrm{I}} = \mathcal{H}_{\mathrm{I},1} \times \cdots \times \mathcal{H}_{\mathrm{I},d}, \tag{39}$$

and the corresponding feature map $\phi_{\mathrm{I}} : \mathcal{X} \to \mathcal{H}_{\mathrm{I}}$ is represented by

$$\phi_{\mathrm{I}}(x) = \left(\phi_{\mathrm{I},1}(\tilde{x}), \ldots, \phi_{\mathrm{I},d}(\tilde{x})\right). \tag{40}$$

In functional analysis, the above process of defining a new *Hilbert space* is called *tensor products* of Hilbert spaces.

Moreover, Proposition 4 explicitly presents the component feature map $\phi_{\mathrm{I},j}(\tilde{x})$. Let

$$\phi_{\mathrm{I},j}(\tilde{x}) = \exp\left(\psi_j(\tilde{x})\right), \tag{41}$$

and let the dot product $\langle \phi_{\mathrm{I},j}(\tilde{x}), \phi_{\mathrm{I},j}(\tilde{x}') \rangle_{\mathcal{H}_{\mathrm{I},j}}$ be defined by the product $\phi_{\mathrm{I},j}(\tilde{x})\phi_{\mathrm{I},j}(\tilde{x}')$. Then (34) and (37) are satisfied. Therefore each component function $\psi_j(\tilde{x})$ corresponds to the natural logarithm of the component feature map $\phi_{\mathrm{I},j}(\tilde{x})$.

In conclusion, we propose the following methods of converting the mixed data of categorical and numerical variables into data of purely numerical variables. Given a training sample of observations $(x^{(i)}, y^{(i)})$, $i = 1, \ldots, N$, and any number of test patterns x, we first perform discretization of each numerical variable to get $\tilde{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_{n+m}^{(i)}\right)$ and $\tilde{x} = (x_1, \ldots, x_{n+m})$. Then we learn a BNC using the training sample to estimate $\psi_j^{NB}(x_j)$ or $\psi_j^{TAN}(\tilde{x})$, $j = 1, \ldots, n+m$. Then any input pattern $x = (x_1, \ldots, x_n, r_1, \ldots, r_m)$ can be converted into the following representations:

1. By using a NB,

$$\left(\psi_1^{NB}(x_1), \ldots, \psi_n^{NB}(x_n), r_1, \ldots, r_m\right) \in \mathrm{R}^{n+m}. \tag{42}$$

2. By using a NB, since $\psi_j^{NB}(x_j) = \log \theta_{x_j|+1} - \log \theta_{x_j|-1}$,

$$\left(\theta_{x_1|+1}, \theta_{x_1|-1}, \ldots, \theta_{x_n|+1}, \theta_{x_n|-1}, r_1, \ldots, r_m\right) \in \mathrm{R}^{2n+m}. \tag{43}$$

3. By using a TAN,

$$\left(\psi_1^{TAN}(\tilde{x}), \ldots, \psi_{n+m}^{TAN}(\tilde{x}), r_1, \ldots, r_m\right) \in \mathrm{R}^{n+2m}. \tag{44}$$

4. By using a TAN, since $\psi_j^{TAN}(\tilde{x}) = \log \theta_{x_j|x_{\pi(j)},+1} - \log \theta_{x_j|x_{\pi(j)},-1}$,

$$\left(\theta_{x_1|x_{\pi(1)},+1}, \theta_{x_1|x_{\pi(1)},-1}, \ldots, \theta_{x_{n+m}|x_{\pi(n+m)},+1}, \theta_{x_{n+m}|x_{\pi(n+m)},-1}, r_1, \ldots, r_m\right) \in \mathrm{R}^{2n+3m}. \tag{45}$$

6. Simulations

We conducted a series of simulations on artificial data sets and real world data sets. The main purpose of the simulations is to compare the different methods of converting the categorical variables into numerical representations:

1. One can simply remove all the categorical variables from the data and use only the remaining numerical variables. We denote this representation by REM.
2. The *single number representation* (SNR) means the direct conversion from a categorical variable into ordered discrete numerical values. For the jth categorical variable x_j, the qth categorical value $v_{j,q}$ is converted as
$$v_{j,q} \mapsto q, \quad q = 1, \ldots, n_j. \tag{46}$$
The order of the categorical values can be decided by human experts with knowledge of the data.
3. The *binary number representation* (BNR) means an encoding of n_j values of a categorical variable into binary numbers of length n_j. For the jth categorical variable x_j, the qth categorical value $v_{j,q}$ is converted as
$$v_{j,q} \mapsto (0, \ldots, 0, \underbrace{1, 0, \ldots, 0}_{q}) \in \{0, 1\}^{n_j}, \quad q = 1, \ldots, n_j. \tag{47}$$
Therefore the increase in the dimension of input patterns depends on the number of values n_j in each categorical variable.
4. The *parametric number representation* (PNR) means the suggested methods based on learning of BNCs.
   (a) By using a NB, for the jth categorical variable x_j, the qth categorical value $v_{j,q}$ is converted as
   $$v_{j,q} \mapsto \psi_j^{NB}(v_{j,q}), \quad q = 1, \ldots, n_j. \tag{48}$$
   We denote this representation by PNR.NB1.
   (b) Alternatively, by using a NB,
   $$v_{j,q} \mapsto \left(\theta_{v_{j,q}|+1}, \theta_{v_{j,q}|-1}\right), \quad q = 1, \ldots, n_j. \tag{49}$$
   We denote this representation by PNR.NB2.
   (c) By using a TAN, any input pattern $x = (x_1, \ldots, x_n, r_1, \ldots, r_m)$ is converted into
   $$\left(\psi_1^{TAN}(\tilde{x}), \ldots, \psi_{n+m}^{TAN}(\tilde{x}), r_1, \ldots, r_m\right) \tag{50}$$
   where $\tilde{x} = (x_1, \ldots, x_{n+m})$ represents the pattern obtained by discretizing the numerical variables in x. We denote this representation as PNR.TAN1.
   (d) Alternatively, by using a TAN, an input pattern x is converted into
   $$\left(\theta_{x_1|x_{\pi(1)},+1}, \theta_{x_1|x_{\pi(1)},-1}, \ldots, \theta_{x_{n+m}|x_{\pi(n+m)},+1}, \theta_{x_{n+m}|x_{\pi(n+m)},-1}, r_1, \ldots, r_m\right). \tag{51}$$
   We denote this representation by PNR.TAN2.

To train BNCs such as NBs and TANs, we used the Bayes Net Toolbox (BNT) (Murphy, 2001) and the Structure Learning Package (SLP) (Leray and Francois, 2004) for Matlab. For discretization of numerical variables, we use hist_ic function from the SLP package, which is the method suggested by O Colot et al. (1994).

To compare classification accuracies of various classifiers, SVMs, MLPs, and KNNs were used as classification algorithms. For SVMs, we used LibSVM software (Chang and Lin, 2001), which implements a sequential minimal optimization (SMO) algorithm. We used C-SVC algorithm with either the linear kernel or the Gaussian radial basis function kernel, and the parameter C was fixed to 100. For MLPs, we used Netlab neural network toolbox (Nabney and Bishop, 2001). We used scaled conjugate gradient descent algorithm with 100 iterations to train each network. For KNNs, we used Spider machine learning package (Weston et al., 2003).

6.1. Artificial data sets

We constructed Bayesian networks for generating artificial data sets. The structure of the Bayesian network is described in Fig. 2. The two input variables x_1 and x_2 are categorical and dependent on each other. We set the probabilities $p(y)$ and $p(x_1)$ as

$$p(y = -1) = p(y = +1) = 0.5$$

and

$$p(x_1 = v_1^1) = p(x_1 = v_2^1) = 0.5$$

respectively. The second variable x_2 takes its value in the set of n_2 categorical values, i.e., $\{v_1^2, v_2^2, \ldots, v_{n_2}^2\}$. We generated two types of data by providing different conditional probability tables for $p(x_2|x_1, y)$.

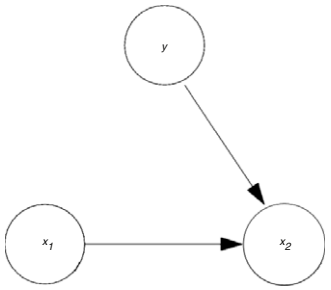

Fig. 2. Structure of the Bayesian network for generating the artificial data sets.

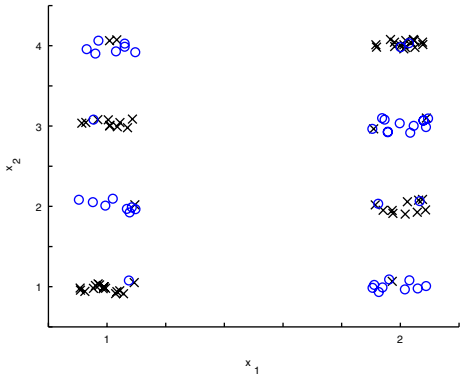

Fig. 3. Scatter plot of a Bumpy XOR data set of size $N = 100$ with $n_2 = 4$ and $\rho = 0.1$. The data set is in the SNR representation and some small uniform random values are added. The black x represents $y = -1$ and the blue o represents $y = +1$.

Table 1
The conditional probability table representing $p(x_2|x_1, y)$ for the Bumpy XOR data with $0 < \rho < 0.5$. Each row should be scaled to have a sum of 1.

$p(x_2\|x_1, y)$				x_2			
				v_1^2	v_2^2	$\cdots$	$v_{n_2}^2$
x_1	v_1^1	y	-1	$1-\rho$	ρ	$\cdots$	ρ
			$+1$	ρ	$1-\rho$	$\cdots$	$1-\rho$
	v_2^1	y	-1	ρ	$1-\rho$	$\cdots$	$1-\rho$
			$+1$	$1-\rho$	ρ	$\cdots$	ρ

In each simulation, 10 data sets of N input–output pairs are generated from a Bayesian network model with fixed parameter values. Then we transform the data sets into numerical representations by applying the different methods of conversion. Next, SVM algorithms with either linear kernels or Gaussian kernels are trained on each of the data sets. The parameters for the Gaussian kernels are decided by the 5-fold cross validation. The trained SVMs are tested on an independently generated test set of size 10000. Finally we obtain the 10 classification accuracies on the test set.

6.1.1. Bumpy XOR data

The first type of the data is generated by using the conditional probability table given in Table 1. The parameter $0 < \rho < 0.5$ represents noises in the data. From the Bayesian networks provided with the conditional probabilities, a number of data sets of different sizes are generated. Fig. 3 illustrates the example of the data set of size $N = 100$ and $\rho = 0.1$.

Figs. 4 and 5 illustrate the simulation results on the Bumpy XOR data by SVMs with Gaussian kernels. Most of the results with various sample sizes and number of values of x_2 show that the PNR.TAN1, PNR.TAN2, and TAN methods are superior to the other methods. But the PNR.NB1, PNR.NB2, and NB methods are worse than the other methods. This is because TAN accurately models the data and the underlying distribution. Also for most of the cases PNR.TAN1 and PNR.TAN2 are better than TAN, and PNR.NB1 and PNR.NB2 are better than NB. This means that using the PNR methods is better than using only the NB or TAN for classifications.

Fig. 6 illustrate the simulation results on the Bumpy XOR data by SVMs with linear kernels. Only the PNR.TAN1, PNR.TAN2, and TAN methods are applicable to linear classifiers on a data with highly dependent variables.

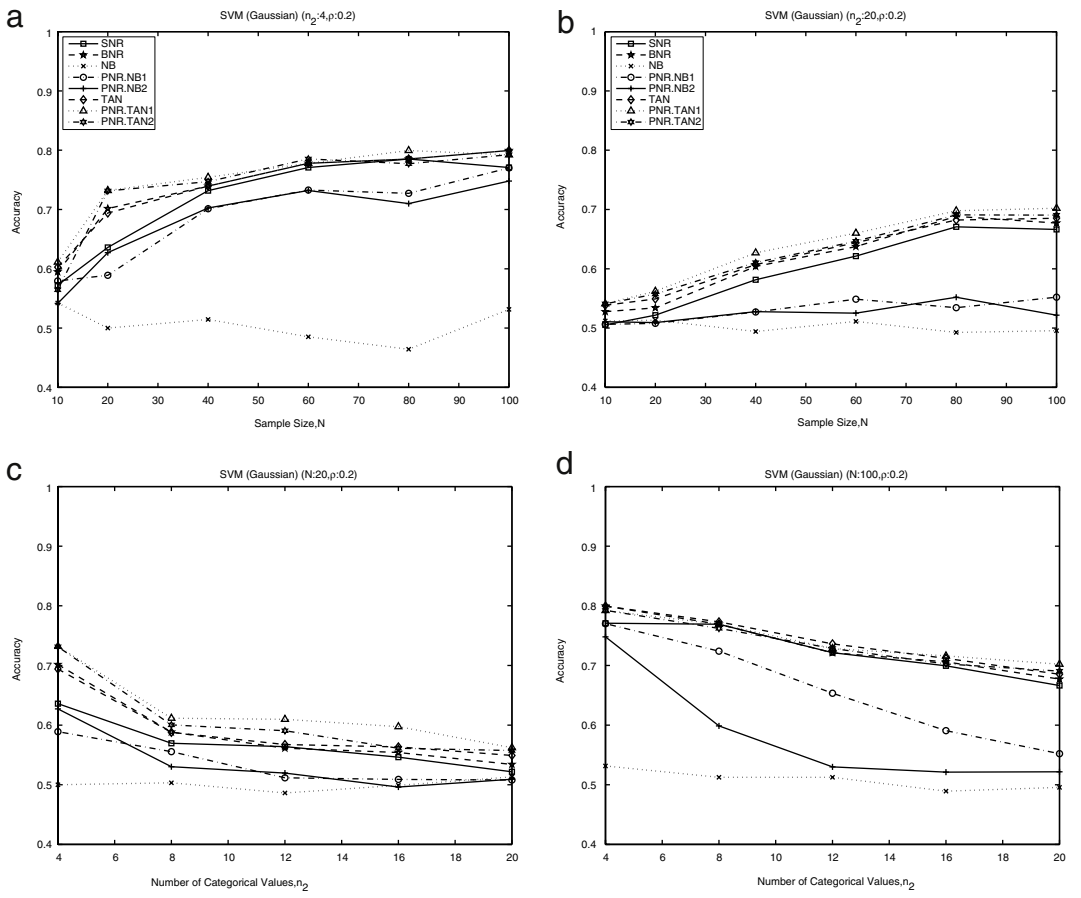

Fig. 4. Average classification accuracies of SVMs with Gaussian kernels on the Bumpy XOR data. (a) and (b) are the results for various sample sizes with fixed $n_2 = 4$ and $n_2 = 20$ respectively. (c) and (d) are the results for various number of categorical values n_2 with fixed $N = 20$ and $N = 100$ respectively.

6.1.2. Flat XOR data

The second type of the data is generated by using the conditional probability table given in Table 2. Fig. 7 illustrates the example of the data set of size $N = 100$ and $\rho = 0.1$.

Figs. 8 and 9 illustrate the simulation results on the Flat XOR data by SVMs with Gaussian kernels. In most of the cases the SNR method is superior to the other methods. Since the accuracies of the SNR method were as good as the other methods for the Bumpy XOR data, we can conclude that the order of the categorical values of a categorical variable is very important for the SNR representation. For all cases the PNR.TAN1, PNR.TAN2, TAN methods are as good as the BNR method. Also in Fig. 8 the PNR.TAN1 shows robustness to small sample situations.

Fig. 10 illustrate the simulation results on the Flat XOR data by SVMs with linear kernels. Only the PNR.TAN1, PNR.TAN2, and TAN methods give meaningful classification accuracies. Even if the Flat XOR data is simpler than the Bumpy XOR data, its variables are still correlated, so linear classifiers are not easily applicable.

6.2. Real world data sets

We selected seven real world data sets for experiments. The descriptions of the data sets are presented in Table 3. All of the data sets were selected from the UCI repository of machine learning databases (Newman et al., 1998). All of the data sets are binary classification problems, and contain categorical variables.

The procedure for a classification experiment on real world data sets is as follows:

Procedure for Experiments

Step 1. Given a data set of N input–output pairs, each numerical variable in the data set is standardized. That is, we first calculate the sample mean and the sample standard deviation of each variable. Then we subtract the sample mean from the corresponding variable and divide the variable by the sample standard deviation.

Step 2. We randomly divide the data set into 10 disjoint sets of equal size, $D^{(1)}, \ldots, D^{(10)}$. Of the 10 subsets, a single subset $D^{(i)}$ is retained as test data, and the remaining 9 subsets are used as training data.

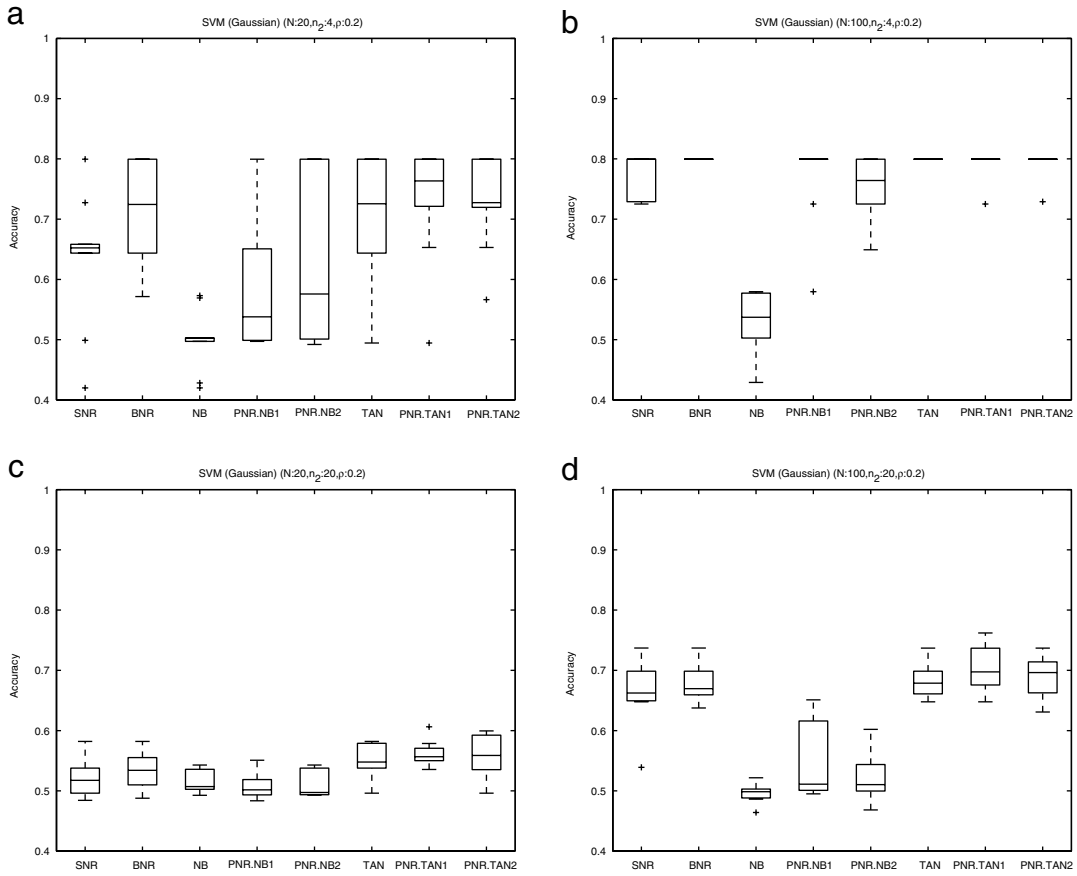

Fig. 5. Box plots for the average classification accuracies of SVMs with Gaussian kernels on the Bumpy XOR data. (a) and (b) are the results for $n_2 = 4$ with $N = 20$ and $N = 100$ respectively. (c) and (d) are the results for $n_2 = 20$ with $N = 20$ and $N = 100$ respectively.

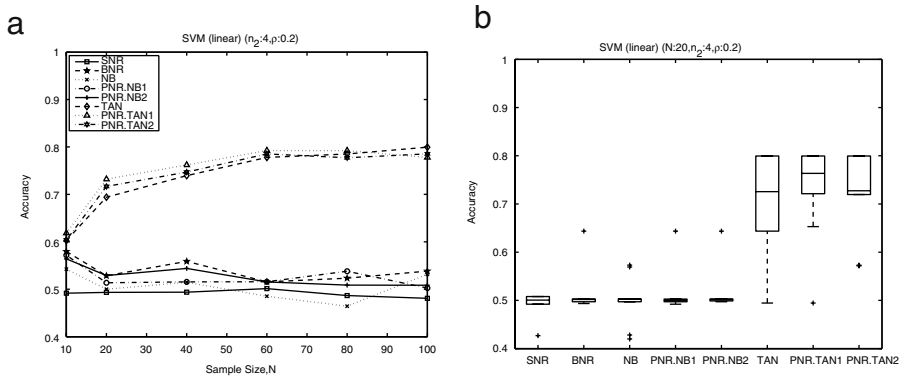

Fig. 6. Average classification accuracies (a) and box plots (b) of SVMs with linear kernels on the Bumpy XOR data.

Table 2
The conditional probability table representing $p(x_2|x_1, y)$ for the Flat XOR Data with $0 < \rho < 0.5$. Each row should be scaled to have a sum of 1.

$p(x_2\|x_1, y)$				x_2			
				v_1^2	v_2^2	$\cdots$	$v_{n_2}^2$
x_1	v_1^1	y	-1	$1-\rho$	$1-\rho$	$\cdots$	ρ
			$+1$	ρ	ρ	$\cdots$	$1-\rho$
	v_2^1	y	-1	ρ	ρ	$\cdots$	$1-\rho$
			$+1$	$1-\rho$	$1-\rho$	$\cdots$	ρ

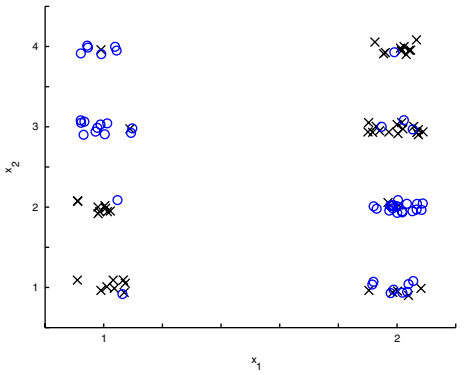

Fig. 7. Scatter plot of a Flat XOR data set of size $N = 100$ with $n_2 = 4$ and $\rho = 0.1$. The data set is in the SNR representation and some small uniform random values are added. The black x represents $y = -1$ and the blue o represents $y = +1$.

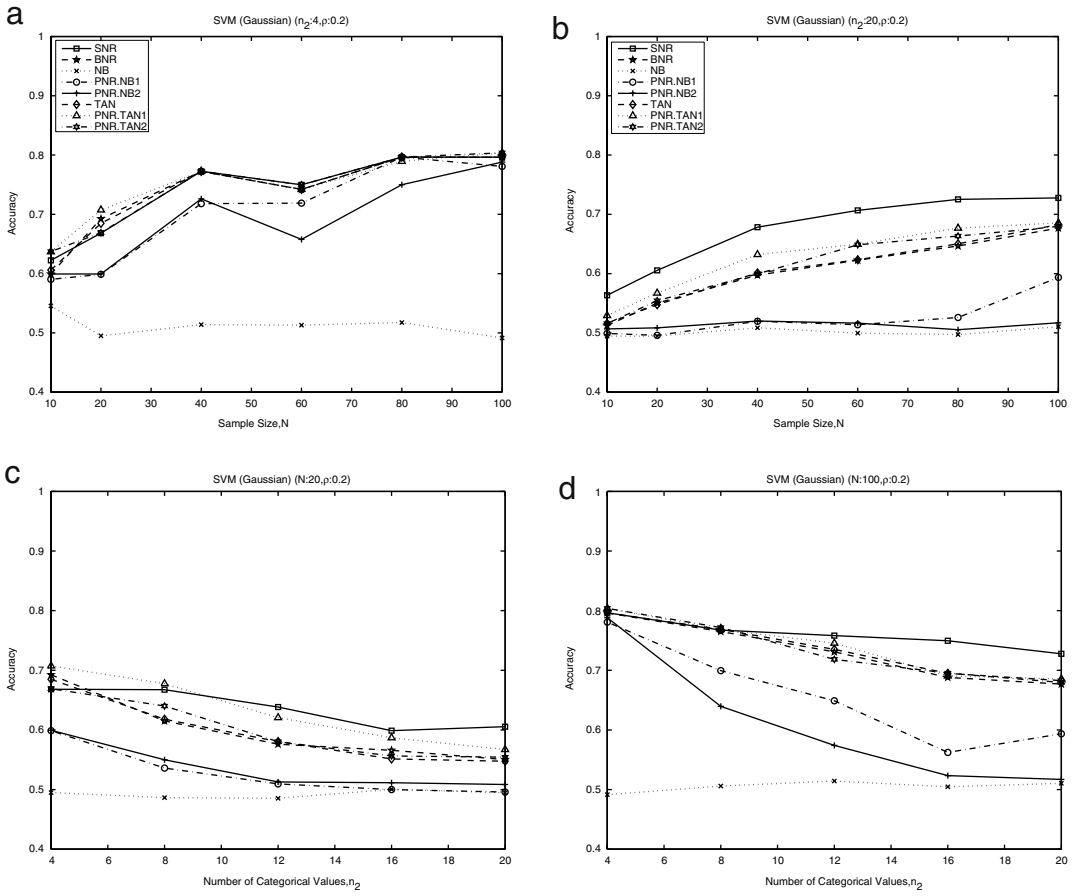

Fig. 8. Average classification accuracies of SVMs with Gaussian kernels on the Flat XOR data. (a) and (b) are the results for various sample sizes with fixed $n_2 = 4$ and $n_2 = 20$ respectively. (c) and (d) are the results for various number of categorical values n_2 with fixed $N = 20$ and $N = 100$ respectively.

Step 3. Seven different data sets are obtained by applying the different methods for converting the categorical variables into numerical representations:

(a) The REM data set is obtained by removing all the categorical variables.

(b) The SNR data set is obtained by applying the procedure (46) to each categorical variable and standardizing it.

(c) The BNR data set is obtained by applying the procedure (47) to each categorical variable.

(d) The PNR.NB1 and the PNR.NB2 data sets are obtained by training a NB on the discretized training data and applying the procedure (48) and (49).

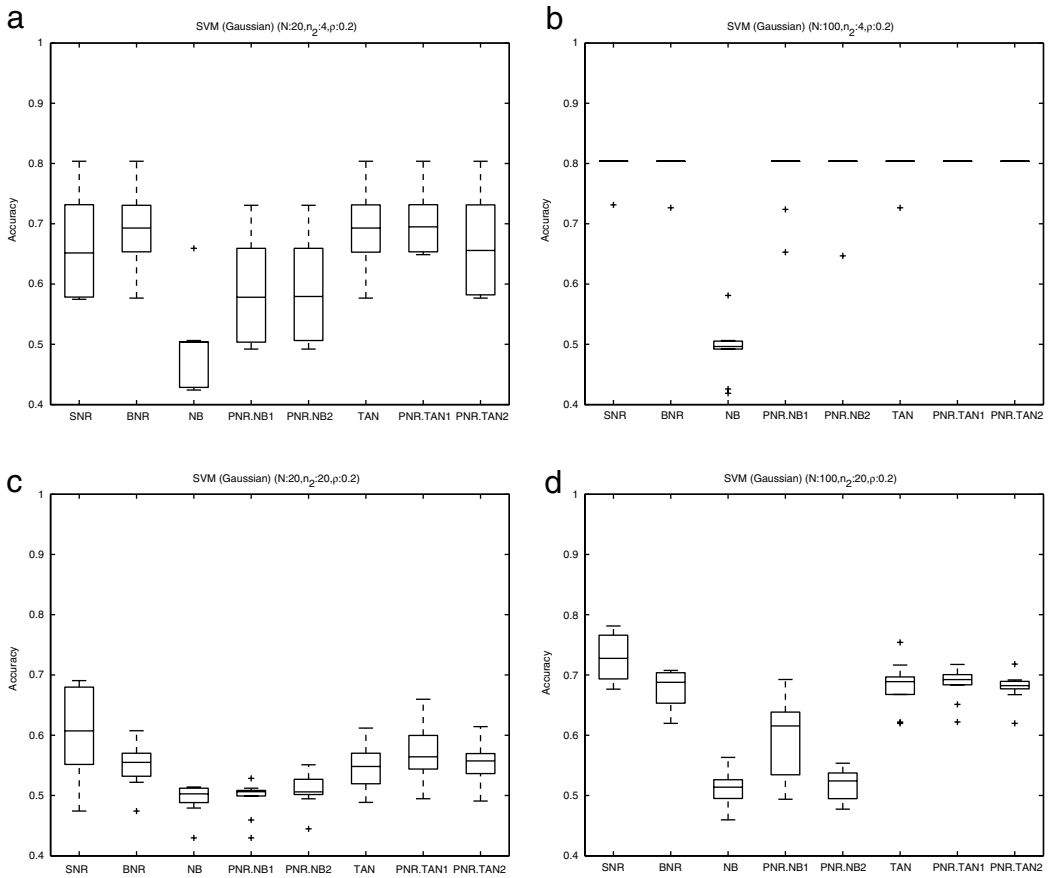

Fig. 9. Box plots for the average classification accuracies of SVMs with Gaussian kernels on the Flat XOR data. (a) and (b) are the results for $n_2 = 4$ with $N = 20$ and $N = 100$ respectively. (c) and (d) are the results for $n_2 = 20$ with $N = 20$ and $N = 100$ respectively.

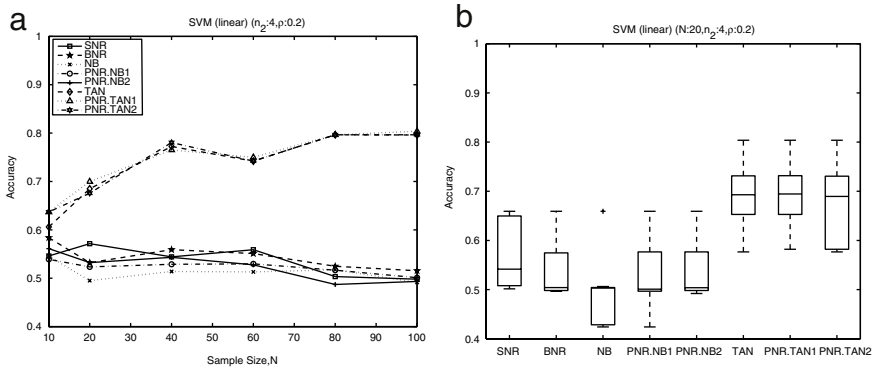

Fig. 10. Average classification accuracies (a) and box plots (b) of SVMs with linear kernels on the Flat XOR data.

(e) The PNR.TAN1 and the PNR.TAN2 data set are obtained by training a TAN on the discretized training data and constructing the representation (50) and (51).

Step 4. SVM, MLP, and KNN algorithms are trained and tested for the five data sets. The model parameter of each algorithm is selected by 5-fold cross validation using the training data set: the kernel parameter for SVMs, the number of hidden units for MLPs, and the number of neighbors for KNNs. The test set accuracies (TSAs) of the trained classifiers are calculated.

Step 5. The procedures Step 6.2 and Step 6.2 are repeated for each pair of training data and test data, and 10 TSA values, $p_{R,A}^{(i)}$, $i = 1, \ldots, 10$, are obtained for each data representation R and each classification algorithm A.

Table 3
Descriptions of data sets.

Data Set	#Instances	#Variables		#Classes
		Numerical	Categorical	
Breast cancer	277	0	9	2
Voting records	435	0	16	2
Credit approval	653	6	9	2
Cylinder bands	349	20	13	2
MONK	556	0	6	2
SPECT heart	267	0	22	2
Statlog heart	270	7	6	2

Table 4
Descriptions of variables in Breast cancer data set.

Number	Variable	Possible values
1	Age	10–19, 20–29, *c* . . . , 90–99
2	Menopause	lt40, ge40, premeno
3	Tumor-size	0–4, 5–9, . . . , 55–59
4	Inv-nodes	0–2, 3–5, . . . , 36–39
5	Node-caps	Yes, no
6	Deg-malig	1, 2, 3
7	Breast	Left, right
8	Breast-quad	Left-up, left-low, right-up, right-low, central
9	Irradiat	Yes, no

The procedure described above enables us to perform the 10-fold cross validated paired t test (Dietterich, 1998). Suppose the null hypothesis is that two methods (R_1, A_1) and (R_2, A_2) will have the same TSAs for a randomly drawn training data. Through the 10-fold cross validation, we obtain 10 pairs of TSAs, $(p^{(i)}_{R_1,A_1}, p^{(i)}_{R_2,A_2})$, $i = 1, \ldots, 10$, and then we compute the statistic

$$T = \frac{\bar{p}\sqrt{10}}{\sqrt{\sum_{i=1}^{10}(p^{(i)} - \bar{p})^2/9}} \tag{52}$$

where $p^{(i)} = p^{(i)}_{R_1,A_1} - p^{(i)}_{R_2,A_2}$ and $\bar{p} = \sum_{i=1}^{10} p^{(i)}/10$. Under the null hypothesis the statistic T has a t distribution with $10 - 1 = 9$ degrees of freedom. The p-value is calculated as the probability $p(X > T)$ where X is a random variable having a t distribution with 9 degrees of freedom. Therefore when we want to compare a method (R_1, A_1) with respect to the other method (R_2, A_2), then small p-values, e.g., < 0.1, imply that the method (R_1, A_1) has higher TSAs, while large p-values, e.g., > 0.9, imply that (R_1, A_1) has smaller TSAs.

6.2.1. Breast cancer data set

The original Breast Cancer data set contains 286 instances, but we removed 9 instances which have missing values. The instances are described by 9 attribute variables, all of which are categorical variables. Table 4 contains information of the attribute variables. Note that the categorical values of some variables are subintervals of larger intervals, i.e., discretizations of numerical variables.

Table 5 illustrates the experimental results on the Breast Cancer data set. From the table, we can see that PNR.NB2 method is significantly better than or equivalent to SNR and BNR methods. This implies that modeling with NBs is appropriate for this data set. The TSAs of NBs and TANs are same to 74.03%. On the other hand, KNNs with SNR, PNR.TAN1, or PNR.TAN2 are significantly worse than KNNs with BNR method.

6.2.2. Voting records data set

This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes. This data set contains 435 instances with no missing value. The class label to be predicted represents the democrat or republican. Each of the 16 attributes has three types of votes as its values: yea, unknown disposition, and nay.

Table 6 illustrates the experimental results on the Voting Records data set. While most of the TSAs are similar, the KNN with BNR shows slightly bad results. This implies some of the variables are irrelevant in the classification and they may add noises to the distances calculated by KNNs.

Table 5
Results on breast cancer data set.

		REM	SNR	BNR	PNR.NB1	PNR.NB2	PNR.TAN1	PNR.TAN2
TSA(%)	SVM	–	68.75	70.46	74.35	**77.30**	66.12	70.81
	MLP	–	70.78	72.91	73.31	**75.85**	75.11	75.09
	KNN	–	69.68	73.68	72.61	**73.69**	70.45	70.81
p-value about SNR	SVM	–	–	0.1539	**0.0215**	**0.0105**	0.7466	**0.087**
	MLP	–	–	0.2696	0.1649	**0.0100**	0.1063	**0.0325**
	KNN	–	–	**0.0128**	**0.0666**	**0.0417**	0.3212	0.2402
p-value about BNR	SVM	–	0.8461	–	**0.0801**	**0.0137**	0.9449	0.3688
	MLP	–	0.7304	–	0.4472	0.2144	0.2934	0.2509
	KNN	–	0.9872	–	0.8014	0.4973	0.9574	0.9070

Table 6
Results on voting records data set.

		REM	SNR	BNR	PNR.NB1	PNR.NB2	PNR.TAN1	PNR.TAN2
TSA(%)	SVM	–	94.50	94.95	94.26	94.96	**95.42**	94.73
	MLP	–	94.95	**95.88**	95.42	95.42	95.41	95.41
	KNN	–	93.59	92.89	95.19	95.66	**96.33**	94.96
p-value about SNR	SVM	–	–	0.2556	0.6247	0.2934	0.1109	0.3649
	MLP	–	–	0.1338	0.2506	0.2199	0.3409	0.1747
	KNN	–	–	0.7850	**0.0970**	**0.0339**	**0.0187**	0.1305
p-value about BNR	SVM	–	0.7444	–	0.8309	0.4940	0.2181	0.6594
	MLP	–	0.8662	–	0.7048	0.7463	0.7819	0.7260
	KNN	–	0.215	–	**0.0212**	**0.0092**	**0.0004**	**0.0142**

Table 7
Descriptions of variables in credit approval data set.

Number	Variable	Possible values
1	*A*1	*b*, *a*
2	*A*2	(continuous)
3	*A*3	(continuous)
4	*A*4	*u*, *y*, *l*, *t*
5	*A*5	*g*, *p*, *gg*
6	*A*6	*c*, *d*, *cc*, *i*, *j*, *k*, *m*, *r*, *q*, *w*, *x*, *e*, *aa*, *ff*
7	*A*7	*v*, *h*, *bb*, *j*, *n*, *z*, *dd*, *ff* , *o*
8	*A*8	(continuous)
9	*A*9	*t*, *f*
10	*A*10	*t*, *f*
11	*A*11	(continuous)
12	*A*12	*t*, *f*
13	*A*13	*g*, *p*, *s*
14	*A*14	(continuous)
15	*A*15	(continuous)

6.2.3. Credit approval data set

This data set concerns credit card applications. But all attribute names and values have been changed to meaningless symbols. This data set originally contains 690 instances, but we removed 37 instances that have missing values. Table 7 describes the list of attribute variables and the possible values.

Table 8 illustrates the experimental results on the Credit Approval data set. Note that there are large number of values for attributes 6 and 7 in Table 7. So the TSEs of BNR methods could not be superior with any classification algorithms. PNR.TAN2 shows significantly good result with SVM algorithm. And SNR also shows good results with MLP algorithm.

6.2.4. Cylinder bands data set

This data set originally consists of 39 attribute variables and one class label. But we removed one variable that have too many missing values, and removed six non-informative categorical variables. Also we further removed instances with missing values.

Table 9 illustrates the experimental results. The TSA results shows that PNR.TAN1 is good for SVM and KNN, and BNR is good for MLP. Moreover, the *p*-value results demonstrates that BNR is superior to the other methods except PNR.TAN1 with SVM, and PNR.NB1, PNR.TAN1, and PNR.TAN2 methods are superior to the SNR method. These results may partly explained by the appropriateness of BNCs in the modeling of this data set. Actually the TSAs of NBs and TANs are 72.48% and 73.63% respectively.

Table 8
Results on credit approval data set.

		REM	SNR	BNR	PNR.NB1	PNR.NB2	PNR.TAN1	PNR.TAN2
TSA(%)	SVM	77.03	84.54	83.15	85.61	86.52	82.24	**87.75**
	MLP	74.90	**87.14**	84.84	85.92	86.38	83.15	86.68
	KNN	73.52	79.93	83.78	83.93	84.39	**85.32**	83.62
p-value about SNR	SVM	0.9992	–	0.7614	0.3146	0.1852	0.9136	**0.0685**
	MLP	1.0000	–	0.9592	0.9231	0.8079	0.9986	0.8291
	KNN	0.9930	–	**0.0494**	**0.0269**	**0.0332**	**0.0185**	**0.0135**
p-value about BNR	SVM	0.9981	0.2386	–	**0.0516**	**0.0212**	0.7634	**0.0017**
	MLP	1.0000	**0.0408**	–	0.1987	0.1607	0.8403	**0.0940**
	KNN	1.0000	0.9506	–	0.4557	0.2868	0.2194	0.5463

Table 9
Results on cylinder bands data set.

		REM	SNR	BNR	PNR.NB1	PNR.NB2	PNR.TAN1	PNR.TAN2
TSA(%)	SVM	69.88	73.31	80.22	79.36	73.61	**81.08**	76.46
	MLP	63.01	69.34	**82.24**	75.34	72.19	73.62	75.32
	KNN	72.48	68.17	78.51	75.34	73.62	**78.78**	73.34
p-value about SNR	SVM	0.8434	–	**0.0143**	**0.0465**	0.4573	**0.0083**	0.1064
	MLP	0.9896	–	**0.0001**	**0.0258**	0.1691	**0.0598**	**0.0362**
	KNN	**0.0684**	–	**0.0009**	**0.0092**	**0.0257**	**0.0004**	**0.0384**
p-value about BNR	SVM	0.9976	0.9857	–	0.7201	0.9914	0.1969	0.9675
	MLP	1.0000	0.9999	–	0.9984	0.9993	0.9996	0.9839
	KNN	0.9987	0.9991	–	0.9160	0.9903	0.4497	0.9763

Table 10
Results on MONK's problems data set.

		REM	SNR	BNR	PNR.NB1	PNR.NB2	PNR.TAN1	PNR.TAN2
TSA(%)	SVM	–	**100.00**	**100.00**	91.75	91.75	**100.00**	**100.00**
	MLP	–	**100.00**	**100.00**	74.81	74.65	**100.00**	**100.00**
	KNN	–	92.45	94.24	90.34	95.87	**100.00**	**100.00**
p-value about SNR	SVM	–	–	**0.0000**	0.9797	0.9987	**0.0000**	**0.0000**
	MLP	–	–	**0.0000**	1.0000	1.0000	**0.0000**	**0.0000**
	KNN	–	–	0.1189	0.7131	**0.0797**	**0.0001**	**0.0001**
p-value about BNR	SVM	–	**0.0000**	–	0.9797	0.9987	**0.0000**	**0.0000**
	MLP	–	**0.0000**	–	1.0000	1.0000	**0.0000**	**0.0000**
	KNN	–	0.8811	–	0.8234	0.2059	**0.0010**	**0.0010**

6.2.5. *MONK's problems data set*

There are three MONK's problems, and we selected the first data set among them. This MONKs-1 data set contains total 556 instances with 6 variables. Suppose $a_1, \ldots, a_6$ denote the variables. The number of values of each variable is 2, 3, or 4. All the instances in the MONKs-1 data set follows the rule: the label $y = +1$ if $a_1 = a_2$ or $a_5 = 1$, and $y = -1$ otherwise. There is no noise added. Therefore it is possible to attain 100% accuracies.

Table 10 illustrates the experimental results on the MONKs-1 data set. It is clear that the PNR.TAN1 and PNR.TAN2 is superior to the other methods. This is because the TANs accurately models the underlying target concept. On the other hand, the PNR.NB1 and PNR.NB2 methods are the worst methods. This is because the NBs could not model the underlying target concept of the data. Actually the TSAs of NBs and TANs are 74.65% and 100% respectively. Also SNR and BNR could achieve 100% TSAs except when they were applied to KNN algorithms. This is because KNNs could not remove irrelevant variables, i.e., a_3 and a_4.

6.2.6. *SPECT heart data set*

The purpose of this data set is to classify each of the patients into two categories: normal and abnormal. The data set consists of 22 binary attributes, which are the extracted features from original images.

Table 11 illustrates the experimental results on the SPECT heart data set. The SVMs with BNR or PNR.TAN2 achieve the maximum TSAs among the other methods. We also note that PNR.TAN1 and PNR.TAN2 are better than PNR.NB1 and PNR.NB2. This is because TANs more accurately model the data set than NBs. The TSAs of NBs and TANs are 78.25% and 82.07% respectively.

6.2.7. *Statlog heart data set*

The aim of the Statlog heart data set is to make diagnoses of heart diseases in a person. The data set contains 270 instances with no missing value. Each instance has 13 variables. 6 variables are real valued and one variable is ordered numeric, so

Table 11
Results on SPECT heart data set.

		REM	SNR	BNR	PNR.NB1	PNR.NB2	PNR.TAN1	PNR.TAN2
TSA(%)	SVM	–	83.93	**84.30**	83.19	83.93	84.29	**84.30**
	MLP	–	80.58	79.81	74.19	81.35	82.82	**83.56**
	KNN	–	79.09	80.21	81.70	80.60	79.77	**81.31**
p-value about SNR	SVM	–	–	0.1717	0.7223	0.5	0.3052	0.1717
	MLP	–	–	0.6961	0.9813	0.2920	**0.0829**	**0.035**
	KNN	–	–	**0.0955**	**0.0219**	0.1086	0.3657	**0.087**
p-value about BNR	SVM	–	0.8283	–	0.8283	0.8283	0.5098	0.5
	MLP	–	0.3039	–	0.9574	0.1814	**0.0820**	**0.0467**
	KNN	–	0.9045	–	0.1717	0.3864	0.5774	0.2651

Table 12
Results on Statlog heart data set.

		REM	SNR	BNR	PNR.NB1	PNR.NB2	PNR.TAN1	PNR.TAN2
TSA(%)	SVM	72.59	77.04	79.63	**81.48**	75.56	76.67	77.04
	MLP	69.63	74.07	74.07	76.30	76.30	**78.89**	73.33
	KNN	76.67	**82.96**	81.85	81.11	79.63	81.48	81.11
p-value about SNR	SVM	0.9263	–	**0.0863**	**0.0091**	0.6873	0.5681	0.5000
	MLP	0.8726	–	0.5000	0.1786	0.2217	**0.0352**	0.6032
	KNN	0.9757	–	0.7293	0.8185	0.8827	0.7690	0.7097
p-value about BNR	SVM	0.9695	0.9137	–	**0.0075**	0.9534	0.9736	0.9043
	MLP	0.9263	0.5000	–	0.1644	0.1394	**0.0874**	0.5839
	KNN	0.9876	0.2707	–	0.6537	0.8804	0.5839	0.5993

the 7 variables are treated as numerical variables. The other variables are either binary valued or nominal valued, so those variables are treated as categorical variables. The class label to be predicted represents the absence or presence of heart disease. In this data set, there is a moderate number of possible values for each categorical variable, i.e., at most 4 values.

Table 12 displays the experimental results on the Statlog heart data set. From the Table 12, each classification algorithm achieves its maximum TSA on different data representation. SVM takes its maximum TSA on PNR.NB1, MLP on PNR.TAN1, and KNN on SNR. This implies that even within a single data classification problem, different algorithms can have different optimal data preprocessing methods. And the maximum values obtained by PNR.NB1 and PNR.TAN1 have *p*-values less than 0.1. This implies that the BNCs are quite adequate modeling methods for this data set. Actually the averaged TSAs of NBs and TANs are 82.59% and 82.22% respectively, which are more accurate results than the other methods except KNNs with SNR method.

7. Conclusions

In this paper we suggested effective methods of converting categorical variables into numerical variables for the classification of mixed data of categorical and numerical variables. We suggested to use a probabilistic model based kernel function to model the mixed data, and defined the ideal kernel with respect to the minimum-error-rate classifier. Since the ideal kernel function is defined in terms of the posterior probability, the Bayesian network classifiers such as naive Bayesian classifiers and tree-augmented naive Bayesian classifiers were applied in modeling the posterior probability. The learning of the Bayesian network classifiers is computationally efficient and the smoothed parameter estimation method is robust to noises and data losses. Moreover, the suggested decomposition of the ideal kernel explicitly identifies the features extracted from the mixed data. Using the extracted numerical features we can easily reconstruct the estimated minimum-error-rate classifier, and the increase in dimensionality of input patterns is small and controllable.

From the simulations we can conclude that the suggested methods can be good alternatives to the typical methods. Especially, the SNR and BNR methods have degraded performances when the number of values in categorical variables is large and the order of the values are irrelevant to the class labels, or the classification algorithms are not properly selected. On the other hand, the suggested methods such as PNR.NB1 and PNR.TAN1 show good performances when NBs and TANs well model the data and the classification algorithms are properly selected.

Designing effective conversions for mixed data requires effective modeling of the data. Therefore, researches on more accurate generative models besides Bayesian network classifiers are important challenges. Also, preprocessing and modeling other various kinds of structured data such as strings and images are important problems.

References

Bishop, C.M., 1995. Neural Networks for Pattern Recognition. Oxford University Press.
Burges, C.J.C., 1998. A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2, 121–167.
Chang, C.-C., Lin, C.-J., 2001. LIBSVM : A library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

Chapelle, O., Haffner, P., Vapnik, V.N., 1999. Support vector machines for histogram-based image classification. IEEE Transactions on Neural Networks 10 (5), 1055–1064.
Cristianini, N., Shawe-Taylor, J., 2000. An Introduction to Support Vector Machines. Cambridge University Press.
Dietterich, T.G., 1998. Approximate statistical tests for comparing supervised classification learning algorithms. Neural Compuatation 10, 1895–1923.
Duda, R.O., Hart, P.E., Stork, D.G., 2001. Pattern Classification, 2nd edition. John Wiley & Sons, New York.
Friedman, N., Geiger, D., Goldszmidt, M., 1997. Bayesian network classifiers. Machine Learning 29, 131–163.
Gärtner, T., 2003. A survey of kernels for structured data. SIGKDD Explorations Newsletter 5 (1), 49–58.
Haussler, D., 1999. Convolutional kernels on discrete structures. Technical report UCSC-CRL-99-10, Computer Science Department, University of California at Santa Cruz.
Haykin, S., 1999. Neural Networks: A Comprehensive Foundation, 2nd edition. Prentice Hall.
Heckerman, D., 1995. A tutorial on learning Bayesian networks. Technical Report MSR-TR-95-06. Microsoft Research.
Hsu, C.-W., Chang, C.-C., Lin, C.-J., 2003. A practical guide to support vector classification. Technical report. Department of Computer Science, National Taiwan University. http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.
Jaakkola, T.S., Haussler, D., 1999. Exploiting generative models in discriminative classifiers. In: M.J., Kearns, S.A., Solla, D.A., Cohn (Eds.), Advances in Neural Information Processing Systems, vol. 11. The MIT Press, pp. 487–493.
Joachims, T., 1998. Text categorization with support vector machines: Learning with many relevent features. In: Nedellec, C., Rouveirol, C. (Eds.). Proceedings of the European Conference on Machine Learning, Berlin, pp. 137–142.
Johnson, N.L., Kotz, S., Balakrishnan, N., 1997. Discrete Multivariate Distributions. John Wiley & Sons, New York.
Lam, W., Bacchus, F., 1992. Learning Bayesian belief networks: An approach based on the MDL principle. Computational Intelligence 10, 269–294.
Leray, P., Francois, O., 2004. BNT, Structure learning package: Documentation and experiments. Technical report, Laboratoire PSI-INSA Rouen-FRE CNRS 2645.
Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C., 2002. Text classification using string kernels. Journal of Machine Learning Research 2, 419–444.
Murphy, K.P., 2001. The Bayes net toolbox for matlab. Computing Science and Statistics: Proceedings of Interface. 33. http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html.
Nabney, I., Bishop, C.M., 2001. Netlab neural network software. http://www.ncrg.aston.ac.uk/netlab/index.php.
Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J., 1998. UCI repository of machine learning databases. Irvine, CA: University of California, Department of Information and Computer Science. http://www.ics.uci.edu/~mlearn/MLRepository.html.
O Colot, C., Olivier, P.C., El-Matouat, A., 1994. Information criteria and abrupt changes in probability laws. In: Signal Processing VII: Théorie and Applications. pp. 1855–1858.
Pavlidis, P., Weston, J., Cai, J., Noble, W.S., 2002. Learning gene functional classifications from multiple data types. Journal of Computational Biology 9 (2), 401–411.
Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann.
Schölkopf, B., Smola, A.J., 2002. Learning with Kernels. The MIT Press.
Shawe-Taylor, J., Cristianini, N., 2004. Kernel Methods for Pattern Analysis. Cambridge University Press.
Tsuda, K., Kin, T., Asai, K., 2002. Marginalized kernels for biological sequences. Bioinformatics 18 (1), 268–275.
Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer-Verlag, New York.
Weston, J., Elisseeff, A., Baklr, G., Sinz, F., 2003. SPIDER: object oriented machine learning library. http://www.kyb.tuebingen.mpg.de/bs/people/spider/.