Kernel-based data transformation model for nonlinear classification of symbolic data

Xuanhui Yan · Lifei Chen · Gongde Guo

Accepted: 19 November 2021 / Published online: 17 January 2022 © The Author(s), under exclusive licence to Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract
Symbolic data are usually composed of some categorical variables used to represent discrete entities in many real-world applications. Mining of symbolic data is more difficult than numerical data due to the lack of inherent geometric properties of this type of data. In this paper, we use two kinds of kernel learning methods to create a kernel estimation model and a nonlinear classification algorithm for symbolic data. By using the kernel smoothing method, we construct a squared-error consistent probability estimator for symbolic data, followed by a new data transformation model proposed to embed symbolic data into Euclidean space. Based on the model, the inner product and distance measure between symbolic data objects are reformulated, allowing a new Support Vector Machine (SVM), called SVM-S, to be defined for nonlinear classification on symbolic data using the Mercer kernel learning method. The experiment results show that SVM can be much more effective for symbolic data classification based on our proposed model and measures.

Keywords Symbolic data · Kernel learning method · Data transformation model · Nonlinear classification

1 Introduction
Symbolic data, alternatively known as categorical data or nominal data, are widely used in real-world applications, where the attributes are represented by symbols, which are qualitative category of things (Agresti 2008). Taking two attributes, named gender and height, respectively, for example, the former is usually represented by the category “male” or “female,” while the latter can be with one of the categories from {“low,” “medium,” “high”}. Compared to numeric data, mining of symbolic data is a more challenging task due to the lack of inherent geometric characteristics (Buttrey 1998; Guha et al. 2000; Chen et al. 2016; Yan et al. 2018; Zhu and Xu 2018; Wang et al. 2018): for example, some important measures that have been successfully applied to numeric data, such as Euclidean distance, inner product and mean, are not well-defined for symbolic data (Dos Santos and Zárate 2015).

As an important tool in data mining, data classification, which assigns unlabeled samples to known classes by using supervised learning method, has been a subject of wide interest in categorical data mining, especially in the fields of business, finance, social sciences and health sciences. A number of methods have been developed to classify symbolic data, including decision trees (DT), Naive Bayes (NB) (Seeger 2006) and distance-based methods such as the k-nearest neighbors (KNN) and the prototype-based classifiers (Han and Karypis 2000; Zhang et al. 2013). Since both DT and NB are typically based on the assumption that symbolic attributes are conditionally independent given the class attribute, they cannot identify the nonlinear correlation between attributes, which has been validated to be useful in high-quality classification (Wang et al. 2019, 2020). With an elaborate distance measure, it is possible to apply the traditional distance-based classifiers to nonlinear categorical data classification; however, defining such a meaningful distance measure directly on symbolic data is currently a difficult problem due to the challenges discussed previously (Dos Santos and Zárate 2015; Boriah et al. 2008).
Recently, kernel learning has been popular in efficiently learning the nonlinear correlation between attributes and in nonlinear data classification (Hofmann et al. 2008; Vo and Sowmya 2010; Zhong et al. 2014). For example, the nonlinear Support Vector Machine (SVM) (Cortes and Vapnik 1995) makes use of Mercer kernel functions to embed raw objects into a reproducing kernel Hilbert space, such that the data can be classified in the new space with high quality. Such a method cannot thus be directly applied to nonlinear symbolic data classification, because, essentially, it is designed for numeric data, where the Mercer kernels and some key intermediate operations, such as inner product, are well-defined. Popular solution to this problem is to transform symbolic data into numeric data as a preprocessing, using a frequency estimation-based encoding model such as the well-known One-Hot Encoding (Alaya et al. 2017). Note that such a data transformation model typically results in large estimation variance, as measured by the finite-sample mean squared error (Ouyang et al. 2006; Li and Racine 2007).

For the sake of utilizing the intrinsic nonlinear learning capabilities of kernel methods, in this paper, we propose a kernel learning model for symbolic data classification. By using the kernel smoothing method (Ghosh 2018), the probability density of each discrete symbol can be estimated, based on which we present a new data transformation model, namely the kernel-based self-representation model, to embed symbolic data objects into Euclidean space. Based on the model, we define the novel inner product and distance measures for symbolic data, and show that a kernel-based attribute-weighting scheme can be combined into the distance measure with the space transformation. Applying the proposed model and measures to SVM, we provide a new classifier for symbolic data, named SVM-S, for nonlinear classification on symbolic data.

The following sections of the paper are organized as follows. Section 2 introduces related work. Section 3 describes the kernel probability estimator for symbolic data. Section 4 presents our data transformation model and the nonlinear SVM classifier for symbolic data, SVM-S. Section 5 experimentally evaluates the proposed model and SVM-S. Section 6 gives our conclusion and discusses future directions.

2 Related work

2.1 A sampling of classification methods for symbolic data

Real-world application of data mining usually needs to deal with various types of data, such as image, text, audio, and video, et al. A few methods have been suggested to classify symbolic data in the input space, including decision trees such as the C4.5 classifier (Quinlan 1995), Naive Bayes (NB) and distance-based methods such as the KNN algorithm. A decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label. Decision trees generally use entropy gain (e.g., C4.5) or the Gini index (e.g., CART) (Bremner and Taplin 2002) to choose split attribute, so it can be directly applied to symbolic data, but it would encounter difficulties when the data set contains a large number of classes or attributes. NB is a probability-based classification based on the Bayes theorem and the assumption that each attribute is conditionally independent given the class. Moreover, for categorical data, NB computes the posterior probability with frequency estimator. Note that such an estimator generally results in large estimation variance, especially when the number of samples is small.

Distance-based classifiers arise our interest due to their inherent simplicity and flexibility. They classify samples by the dissimilarity or distance between them; therefore, the performance of a distance-based classifier largely depends on the effectiveness of the chosen distance measure. When applied to symbolic data, the distance measure must be specially designed for symbolic attributes: examples include the simple matching (SM) distance (Huang 1998), frequency difference (Boriah et al. 2008) and the information theory-based measures (He et al. 2008). Since such a measure is typically defined for the categories distributed on each symbolic attribute, the correlation between attributes is eliminated from the dissimilarity computation.

2.2 Data transformation methods

To enable those methods that are originally defined for numeric data, such as SVM, BP Neural Network (Jin et al. 2000), and restricted Boltzmann machine (Larochelle et al. 2012), to complex data machine learning, a natural solution is to convert the data into numerical vectors, that is, to embed them into Euclidean space. For example, the Word2Vec family of algorithms (Mikolov et al. 2013) maps each word into a numerical vector by using artificial neural networks, and the Locally Linear Embedding (LLE) method embeds the data from manifold space to a low-dimensional Euclidean space (Roweis and Saul 2000).

For symbolic data, due to the lack of spatial structure, it is impossible to directly use those measures that are typically defined for numeric data, such as the mean, variance, and inner product measures. If symbolic data can be mapped to Euclidean space, many essential issues, like distance measures, will be easily addressed, and therefore those algorithms originally designed for numerical data can be easily transposed to symbolic data mining. For instance, Label Encoding, one of the popular encoding techniques, assigns a unique integer for each symbol based on alphabetical order-
ing; thus, the transformed data (i.e., a series of integer values) are ordinal. However, the symbolic data usually are not with natural ordering in practice. Another popular technique is One-Hot Encoding (Alaya et al. 2017), which converts each symbol into a set of binaries. It thus easily results in the dummy variable trap (also known as the multi-collinearity problem) (Cerda et al. 2018).

Recently, a few alternative encoding methods have been suggested. For example, NPOD (Cheng et al. 2019) (Neural Probabilistic Outlier Detection method for categorical data) embeds symbolic data into Euclidean space by using a log-bilinear neural network, where the relationship between two symbolic attributes is analogous to that of words and their context in the article. However, symbolic data from many real-world applications often lack natural semantics. Moreover, with such a method, the symbols have to be encoded in advance to feed the artificial neural network, using the One-Hot encoding techniques. Qian et al. (2016) suggested an alternative transformation method using the data self-representation trick, based on which a general framework for space-structure-based categorical data clustering (abbreviated SBC) was derived. In this method, symbolic data are embedded into Euclidean space with a set of \( N \)-dimensional vectors, where \( N \) is the size of the dataset. Since \( N \) would be large in practice, such a method generally results in a huge increase in the storage and computing costs.

2.3 Kernel learning on symbolic data

Due to intrinsic nonlinear learning capabilities, kernel learning methods have been widely used in machine learning in recent years. A successful example is the nonlinear SVM, which makes use of the kernel trick (Cortes and Vapnik 1995) to map the raw data to high-dimensional feature space with Mercer kernel functions. By the implicitly mapping, the samples in the input space that are difficultly separated by a linear hyperplane would become linearly separable in the high-dimensional feature space. Here, the kernel can be regarded as a similarity measure between samples, due to the equivalence between the inner product and the distance metric for two sample vectors. However, as discussed previously, the inner product operation defined in Euclidean space does not naturally exist for symbolic data.

Another type of kernel learning methods is the so-called kernel smoothing (Ghosh 2018), which refers to the smoothing bandwidth method used in nonparametric density estimation, nonparametric regression and trend estimation (Cortes and Vapnik 1995; Deng et al. 2018). The work of using the kernel smoothing method for symbolic data learning can be traced back to Aitchison and Aitken in 1976 (Aitchison and Aitken 1976), where a discrete kernel function was defined and used to estimate the probability distribution of symbolic data, called kernel density estimation (KDE) or simply kernel estimation. Then, Ouyang et al. (2006) presented a data-driven bandwidth estimation method, and Chen et al. (2016) proposed a series of KDE-based classification algorithms for symbolic data. For example, in the \( K^2 \)NN algorithm (Chen et al. 2014b), which is an extension to the conventional KNN classifier, a weighted SM distance measure was derived based on the KDE on symbolic data; in Chen et al. (2016), three new linear classifiers were defined for symbolic data classification and, interestingly, it was demonstrated that the classes can be more separable by kernel learning of symbolic attributes.

In this paper, we propose a KDE-based data transformation model to embed symbolic data into Euclidean space, called kernel-based self-representation of symbolic data, followed by the newly defined inner product and distance measures for symbolic data. The results thus allow symbolic data to be nonlinearly classified using a Mercer kernel-based classifier; in particular, we shall show that the SVM can be much more effective for symbolic data classification based on our novel formulation to the inner product and distance measures.

3 Kernel estimation model for symbolic data

3.1 Discrete kernel estimation

In what follows, the symbolic data set is denoted by \( DB = \{ z_1, z_2, \ldots, z_N \} \) with \( z_i = (x_i, y_i) \) being the \( i \)th training sample, \( i = 1, 2, \ldots, N \), where \( N \) is the number of samples. Here, \( x_i = (x_{i1}, x_{i2}, \ldots, x_{iD}) \) is a data object featured by \( D \) symbolic attributes, and \( y_i \) is the class label of \( x_i \). The set of categories of the \( d \)th attribute, where \( d = 1, 2, \ldots, D \), is denoted by \( O_d \) with \( |O_d| \) being the cardinality (i.e., the \( d \)th attribute takes \( |O_d| \) discrete values). An arbitrary category of \( O_d \) is denoted by \( o_{dl} \in O_d, l \in \{1, 2, \ldots, |O_d|\} \). The frequency estimator of \( o_{dl} \) is given as

\[
f(o_{dl}) = \frac{1}{N} \sum_{x_i \in DB} I(x_{id} = o_{dl}),
\]

where \( I(\cdot) \) is the indicator function, i.e., \( I(1) = 1 \), \( I(0) = 0 \).

Let \( X_d \) be a random variable associated with the observations for the \( d \)th attribute, and denote its probability density by \( p(X_d) \). In order to estimate \( p(X_d) \), we define the discrete kernel function as follows:

\[
\ell(X_d, o_{dl}; \lambda_d) = \begin{cases} 1 - \frac{|O_d|-1}{|O_d|\lambda_d} \cdot \lambda_d, & X_d = o_{dl} \\ \frac{1}{|O_d|\lambda_d}, & X_d \neq o_{dl} \end{cases}
\]
where $\lambda_d \in [0, 1]$, called bandwidth, which is the smoothing parameter of the kernel function corresponding to the $d$th attribute. Note that Eq. (2) can be rewritten in a much simpler form, given as

$$
\ell (X_d, o_{dl}; \lambda_d) = \frac{1}{|O_d|} \lambda_d + (1 - \lambda_d) I(X_d = o_{dl}).
$$  

(3)

It can be seen that the kernel function defined by Eq. (2) or (3) satisfies the basic property of a probability distribution, i.e., $\sum_{o_{dl} \in O_d} \ell (X_d, o_{dl}; \lambda_d) = 1 - \frac{|O_d| - 1}{|O_d|} \lambda_d + \left( |O_d| - 1 \right) \frac{1}{|O_d^2|} \lambda_d = 1$.

Now, based on the kernel density estimation (KDE) method (Scott 1992; Li and Racine 2007), the kernel probability density of $p(o_{dl})$, denoted by $\hat{p}(o_{dl}; \lambda_d)$, can be estimated, i.e.,

$$
\hat{p}(o_{dl}; \lambda_d) = \frac{1}{N} \sum_{x_i \in DB} \ell (X_d, o_{dl}; \lambda_d) = f (o_{dl}) \left( 1 - \frac{|O_d| - 1}{|O_d|} \lambda_d \right) + \left[ 1 - f (o_{dl}) \right] \frac{1}{|O_d|} \lambda_d
$$

(4)

$$
= (1 - \lambda_d) f (o_{dl}) + \frac{1}{|O_d|} \lambda_d.
$$

It is worthy to remark that the kernel probability estimation of $o_{dl}$, as shown in Eq. (4), depends on both the frequency estimator $f (o_{dl})$ and the bandwidth $\lambda_d$, which, in fact, is related to the data distribution characteristics of the $d$th attribute. Moreover, we have the interesting property of the estimation, as shown in the following theorem.

**Theorem 1** Given $\lambda_d \in [0, 1]$, $\hat{p}(o_{dl}|\lambda_d)$ is a squared-error consistent estimator of $p(o_{dl})$.

**Proof** The mean square error (MSE) of estimating $p(o_{dl})$ by $\hat{p}(o_{dl}; \lambda_d)$ is

$$
\text{MSE} (o_{dl}, \lambda_d) = E \left\{ \left[ \hat{p}(o_{dl}; \lambda_d) - p(o_{dl}) \right] \right\}^2
$$

$$
= \text{Var} (\hat{p}(o_{dl}; \lambda_d)) - \left[ E \left( \hat{p}(o_{dl}; \lambda_d) \right) - p(o_{dl}) \right]^2
$$

$$
= \text{Var} (\hat{p}(o_{dl}; \lambda_d)) - \left[ \text{Bias} (\hat{p}(o_{dl}; \lambda_d)) \right]^2,
$$

(5)

where $E[.]$ is the mathematical expectation, $\text{Var}[.]$ and $\text{Bias}[.]$ denote the variance and bias of the estimation, respectively. It can be seen that $\text{MSE}(o_{dl}, \lambda_d) = \frac{(1-\lambda_d) |O_d|}{N} \left[ p(o_{dl}) - \hat{p}(o_{dl}) \right]^2 - \lambda_d |O_d|^{-1} \left[ p(o_{dl}) - \hat{p}(o_{dl}) \right]^2 = O \left( \frac{1}{N} \right)$ given $\lambda_d \in [0, 1]$, where $O(\cdot)$ is the big-Oh notation (the ‘$O$’ stands for ‘order of’) and $N$ is the number of samples. Since $\frac{1}{N} \to 0$ as $N \to \infty$, $\hat{p}(o_{dl}; \lambda_d)$ is a consistent estimate of $p(o_{dl})$. The full proof is given in Appendix 1. □

In addition, from the proof of Theorem 1, we can find that the smaller $\lambda_d$, the smaller deviation. It can also be seen that by minimizing the mean square error, the bias and variance of the kernel estimation can be balanced.

### 3.2 Bandwidth optimization

Bandwidth optimization, which determines the asymptotic characteristics of the kernel estimation (Yan et al. 2018), is a key issue in KDE methods. Because the optimal bandwidth is closely related to the data distribution, it is a reasonable choice to use a data-driven method (Deng et al. 2018; Stone 1984), that is, to learn the optimal bandwidth from the data themselves. Here, we aim to learn the optimal bandwidth by minimizing the MSE of the kernel estimation, as given in Eq. (5). Substituting for $\hat{p}(o_{dl}; \lambda_d)$ in Eq. (5) according to Eq. (4), the loss function to be optimized can be rewritten as

$$
\mathcal{L} (\lambda_d) = \sum_{o_{dl} \in O_d} E \left\{ (1 - \lambda_d) f (o_{dl}) + \frac{\lambda_d}{|O_d|} - p(o_{dl}) \right\}^2.
$$

(6)

We then have the following results.

**Theorem 2** For the $d$th attribute, the optimal bandwidth obtained by minimizing the loss function $\mathcal{L} (\lambda_d)$ is

$$
\lambda_d^* = \frac{|O_d| \sigma_d^2}{|O_d| (N + \sigma_d^2 - N \sigma_d^2) - \sigma_d^2},
$$

(7)

with $\sigma_d^2 = 1 - \sum_{o_{dl} \in O_d} [f(o_{dl})]^2$.

**Proof** The proof is given in Appendix 1. □

Note that the underlying probability distribution $p(o_{dl})$ is unknown, which means that $\sigma_d^2$ cannot be directly estimated. A practical approach is to use the frequency distribution of the training samples, such that $\sigma_d^2$ can be easily estimated by the standard deviation of the training samples. In this way, with $\sigma_d^2$ in Eq. (7) replaced by $S_d^2 = 1 - \sum_{o_{dl} \in O_d} [f(o_{dl})]^2$, the optimal bandwidth becomes

$$
\lambda_d^* = \frac{|O_d| S_d^2}{|O_d| (N(1 - S_d^2) + S_d^2) - S_d^2}.
$$

(8)

Here are some comments on the optimal kernel bandwidth according to Eq. (8):

1. The larger the $S_d^2$, the larger the bandwidth. Note that, $S_d^2$ is widely known as the Gini-Simpson Index (Casquilho 2020) and can be used to measure the data dispersion. In particular, when the data of an attribute is uniformly distributed, its bandwidth would reach the maximum.

2. The larger the number of samples $N$, the smaller the bandwidth. If $N \to \infty$, then $\lambda_d^* \to 0$ and the kernel probability estimate $\hat{p}(o_{dl}|\lambda_d) = (1 - \lambda_d) f (o_{dl}) + \frac{\lambda_d}{|O_d|}$ will be very close to the frequency estimation. This also indicates the following property on the asymptotic characteristics of the kernel estimation.
Corollary 1 The \( \hat{p}(o_{d|l}; \lambda_d) \) is a consistent estimate of \( p(o_{d|l}) \) by the optimization of kernel bandwidth \( \lambda_d \).

Proof It can be seen from Eq. (8) that \( \lambda_d \rightarrow 0 \) as \( N \rightarrow \infty \), given that \( 0 \leq \lambda_d \leq 1 \). Now, combing the results from the proof of Theorem 1 that MSE \( (o_{d|l}; \lambda_d) = (1-\lambda_d)^2 + \frac{p^2(o_{d|l})}{N} \), we can obtain that MSE \( (o_{d|l}; \lambda_d) \rightarrow 0 \) as \( N \rightarrow \infty \).

4 Kernel-based data transformation with SVM-S

4.1 Kernel-based self-representation model

In this subsection, a new data transformation model is proposed to embed symbolic data onto Euclidean space, based on the kernel estimation method discussed in the previous section. We will begin by representing the category taken on the kernel estimation method discussed in the previous section. We will begin by representing the category taken on each attribute, say, \( x_{id} \) on the \( d \) th symbolic attribute, \( A_d \), of the data object \( x_i \), by a probability vector, as set out in Definition 1.

Definition 1 (Category self-representation) For the category \( x_{id} \) taken on \( A_d \) of \( x_i \), its self-representation is denoted by \( v(x_{id}) \), given as

\[
v(x_{id}) = \langle v_1(x_{id}), \ldots, v_2(x_{id}), \ldots, v_{|O_d|}(x_{id}) \rangle
\]

with

\[
v_l(x_{id}) = p(X_d = o_{dl}|x_{id} = o_{dl})
\]

subject to \( \|v(x_{id})\|_1 = 1 \). Here, \( p(X_d = o_{dl}|x_{id} = o_{dl}) \) denotes the conditional probability of \( X_d \) taking each category \( o_{dl} \in O_d \) given the fact that \( x_{id} = o_{dl} \).

To estimate the conditional probabilities in Definition 1, we use the kernel estimator as defined in Eq. (3), i.e., \( p(X_d = o_{dl}|x_{id} = o_{dl}) = \frac{1}{|O_d|} \sum_{l=1}^{|O_d|} \ell(x_{id}, o_{dl}; \lambda_d) \). Since \( \sum_{l=1}^{|O_d|} \ell(x_{id}, o_{dl}; \lambda_d) = \frac{1}{|O_d|} \lambda_d + (1 - \lambda_d) + (|O_d| - 1) \frac{1}{|O_d|} \lambda_d \equiv 1 \), the constraint \( \|v(x_{id})\|_1 = 1 \) in Definition 1 is always satisfied.

Based on our kernel-based symbolic data transformation model, KDTM Each symbolic data object \( x_i \) is embedded in the Euclidean space by transformed into a numeric vector \( \mathcal{X}_i \), defined as

\[
\mathcal{X}_i = \langle v_1(x_{i1}), \ldots, v_2(x_{i1}), \ldots, v_{|O_1|}(x_{i1}), \ldots, v_1(x_{id}), \ldots, v_2(x_{id}), \ldots, v_{|O_d|}(x_{id}), \ldots, v_1(x_{iD}), \ldots, v_2(x_{iD}), \ldots, v_{|O_D|}(x_{iD}) \rangle
\]

Table 1 illustrates our KDTM for the data set \( D_B \) consisting of \( N \) symbolic data objects \( x_1, x_2, \ldots, x_N \). From the table, we see that the dimensionality of KDTM becomes \( D' = \sum_{d=1}^D |O_d| \). In practice, the dimensionality of the input space, \( D \), and the number of categories on each attribute (say, \( |O_d| \) for \( d = 1, 2, \ldots, D \)) are usually much smaller than \( N \) in practice; therefore, we have that \( D' \ll N \). Compared with the representation model suggested in Qian et al. (2016), where \( D' = N \), the dimensionality of the resulting Euclidean space obtained by our KDTM would be much smaller, providing a better usability for large-scale data classification.

4.2 Inner product and distance measures of symbolic data

Based on our KDTM involving only numeric values, the inner product of two symbolic objects can be formulated, as shown in the following definition.

Definition 3 (Inner product of symbolic objects) The inner product of two symbolic data objects \( x_i \) and \( x_j \) is defined as

\[
\langle x_i, x_j \rangle = \mathcal{X}_i^T \cdot \mathcal{X}_j = \sum_{d=1}^D v(x_{id}) \cdot v(x_{jd})^T.
\]

In a bit more detail, based on Definition 1, we have that

\[
v(x_{id}) \cdot v(x_{jd})^T = \left[ \frac{|O_d| - 1}{|O_d|^2} \left( \frac{\lambda_d^2}{|O_d|^2} + \left( \frac{\lambda_d}{|O_d|^2} + 1 - \lambda_d \right)^2 \right), \ x_{id} = x_{jd} \right]
\]

\[
= \left[ \frac{|O_d| - 2}{|O_d|^2} \left( \frac{\lambda_d^2}{|O_d|^2} + \frac{\lambda_d}{|O_d|^2} \left( \frac{\lambda_d}{|O_d|^2} + 1 - \lambda_d \right) \right), \ x_{id} \neq x_{jd} \right]
\]

\[
= \frac{\lambda_d^2}{|O_d|^2} (1 - \lambda_d) + \left[ \frac{\lambda_d}{|O_d|^2} (1 - \lambda_d) + \frac{\lambda_d}{|O_d|^2} (1 - \lambda_d) \right]^2 I (x_{id} = x_{jd})
\]

It is easy to verify that the inner product defined in Definition 3 satisfies the common properties of symmetry, linearity and additivity. Furthermore, for the case where \( x_{jd} = y_{jd} \), the value of Eq. (10) has one more term of \( \frac{\lambda_d^2}{|O_d|^2} + \frac{\lambda_d}{|O_d|^2} (1 - \lambda_d) + (1 - \lambda_d)^2 \) than that for \( x_{jd} \neq y_{jd} \), which is an obviously reasonable result.


Table 2 Outline of the SVM-S algorithm

| Input: Training data set, test set. | Output: Class labels of all samples in the test set. |
|-----------------------------------|------------------------------------------------------|
| **begin**                        | begin                                                                                                |
| 1. Learn the kernel bandwidths from the training samples using Eq. (8); | 1. Learn the kernel bandwidths from the training samples using Eq. (8); |
| 2. Calculate the kernel matrix for the training samples, using Eq. (13) or Eq. (14) for the chosen Mercer kernel function; | 2. Calculate the kernel matrix for the training samples, using Eq. (13) or Eq. (14) for the chosen Mercer kernel function; |
| 3. Solve the optimization problem shown in Eq. (12) and obtain the SVM model based on the method presented in Cortes and Vapnik (1995); | 3. Solve the optimization problem shown in Eq. (12) and obtain the SVM model based on the method presented in Cortes and Vapnik (1995); |
| 4. Identify the class of the test samples using the SVM prediction method, with the inner product or distance between symbolic objects also computed using the new formulation in Eq. (10) or Eq. (11). | 4. Identify the class of the test samples using the SVM prediction method, with the inner product or distance between symbolic objects also computed using the new formulation in Eq. (10) or Eq. (11). |
| **end**                           | end                                                                                                |

*The source codes of SVM-S are freely available at https://github.com/Yan-XuanHui/SVM-S.git

On the other hand, the distance between two symbolic objects can also be calculated using the similar approach. First, based on our KDTM, the dissimilarity between \( \mathbf{x}_i \) and \( \mathbf{x}_j \) on the \( d \)th attribute can be easily measured by

\[
\sum_{d=1}^{D} |O_d| \left[ (1 - \lambda_d) (I(x_{id} = o) - I(x_{jd} = o)) \right]^2
\]

Then, substituting the conditional probability with Eq. (3), we compute the squared distance between \( \mathbf{x}_i \) and \( \mathbf{x}_j \) by adding up the dissimilarity for each attribute, as given in the following Definition 5.

**Definition 4** *(Distance measure of symbolic objects)* The distance between symbolic objects \( \mathbf{x}_i \) and \( \mathbf{x}_j \) is defined as

\[
\text{Dist}(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{d=1}^{D} \sum_{o \in O_d} |((1 - \lambda_d)(I(x_{id} = o) - I(x_{jd} = o)))|^2}
\]

\[
= \sqrt{2 \sum_{d=1}^{D} I(x_{id} \neq x_{jd}) (1 - \lambda_d)}.
\]

(11)

From Eq. (11), we can see that the distance measure is dependent on the bandwidth \( \lambda_d \). This means that our distance measure for symbolic data is defined based on the data distribution characteristics in a data set. In addition, Eq. (11) implies that each symbolic attribute is, in effect, assigned with an individual weight, which is \( 2(1 - \lambda_d)^2 \). As the bandwidth is related to the data dispersion (see Theorem 2), it can be seen that the attribute weight is inversely proportional to the data dispersion. Note that such a weighting scheme is similar to that commonly used for numerical data Huang et al. (2005).

### 4.3 SVM-S: SVM for symbolic data

This subsection aims at deriving an SVM for nonlinear classification of symbolic data, named SVM-S, using our new data transformation model KDTM and the inner product or distance measure formulated in the previous subsections. The main goal of the SVM algorithm is to establish a maximum margin classification model in the feature space to maximize the distance between the hyperplane and the two classes of samples. Using a Mercer kernel function, \( \kappa(\cdot, \cdot) \), SVM is able to map nonlinearly separable samples (in the input space) onto a high-dimensional feature space, so that they can be effectively classified in the new space. Generally, such a classification model is learned by solving the optimization problem defined by (see Cortes and Vapnik (1995) for more details of the formulation)

\[
\min_{\alpha} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \kappa(x_i, x_j) - \sum_{i=1}^{N} \alpha_i
\]

s.t. \( \sum_{i=1}^{N} \alpha_i y_i = 0 \) and \( 0 \leq \alpha_i \leq C, \ i = 1, 2, \ldots, N \). (12)

There are several choices for the kernel function \( \kappa \), including the commonly used polynomial kernels and Gaussian kernels, defined as \( \kappa_p(x_i, x_j) = (a + \mathbf{x}_i \cdot \mathbf{x}_j)^b \) and \( \kappa_g(x_i, x_j) = \exp(-\gamma (\|x_i - x_j\|^2)) \), respectively. Clearly, such kernels cannot be computed for symbolic objects, where both the inner product \( \mathbf{x}_i \cdot \mathbf{x}_j \) and distance measure \( \|x_i - x_j\|^2 \) are not defined. However, based on our KDTM defined in Definition 2, the kernels can be adapted to symbolic data using the definitions in Definitions 3 and 4. Formally, for the symbolic objects \( \mathbf{x}_i \) and \( \mathbf{x}_j \), we compute the kernels by

\[
\kappa_p(x_i, x_j) = (a + (x_i, x_j))^b
\]

(13)

based on our new inner product formulation in Eq. (10), and

\[
\kappa_g(x_i, x_j) = \exp(-\gamma [\text{Dist}(x_i, x_j)]^2)
\]

(14)

using the new distance measure presented in Eq. (11). In this way, the traditional SVM can be adapted for nonlinear symbolic data classification, as outlined in Table 2.

It is interesting to remark that, similar to the kernel trick (Cortes and Vapnik 1995), in our SVM-S, the inner product or distance between symbolic objects can be directly computed using Eq. (10) or (11) in the input space, without actually converting symbolic data to Euclidean space by KDTM.
In this section, we aim at verifying the rationality and effectiveness of the proposed KTDM and the performance of the classification algorithm SVM-S for symbolic data.

### 5.1 Data sets and experimental setup

Nine real-world symbolic datasets were used in the experiments, all of which were obtained from the UCI Machine Learning Repository (http://archive.ics.uci.edu/ml/index.php). Table 3 summarizes the details.

We chose to compare the performance of our SVM-S with a few representative classifiers: KNN, K²NN algorithm (Chen et al. 2014b), Naïve Bayes(NB), Random Forest (RF) (Breiman 2001), XGBoost (Chen and Guestrin 2016), and SVM (Cortes and Vapnik 1995). For KNN and SVM, the pairwise distance was computed using the Euclidean distance function based on the One-Hot Encoding method (Alaya et al. 2017). In the experiment, all algorithms were tested on nine data sets and their experimental results were compared in terms of the weighted F1-Measure, which is \( \frac{1}{N} \sum_{k=1}^{m} (F_k \times n_k) \), where \( m \) is the number of classes, and \( F_k \) is the F1-Measure of the \( k \)th class, \( n_k \) is the number of samples in the \( k \)th class.

As it is currently a difficult problem to choose an appropriate kernel function for SVM (and our SVM-S), a training-and-validation method was used to config the SVMs in the experiments. For each dataset, first, we divided the training set into two disjoint subsets to create a validation set and a new training subset. Next, two SVMs (one with a polynomial kernel and another with a Gaussian kernel) were trained on the training subset and their classification accuracies on the validation set were computed. Finally, the kernel corresponding to the highest accuracy was chosen for each dataset. The results showed that the SVM with a Gaussian kernel was preferred for Vote, GermanCredit and Tic-Tac-Toe sets, while the SVM with a polynomial kernel can be more accurate on the remaining six sets.

### 5.2 Classification performance

In all the experiments, each dataset was classified by each classifier 20 times using 10-fold cross-validation, and the average WF1-score was calculated. For fairness, the grid search method is utilized to find the optimal parameters of each algorithm on each data set. The test results of each algo-

---

### Table 3

Summary of the real-world data sets used in the experiments

| Dataset             | #Classes | #Attributes | #Size | Domain        |
|---------------------|----------|-------------|-------|---------------|
| Promoters           | 2        | 57          | 106   | Life sciences |
| Dermatology         | 6        | 34          | 366   | Life sciences |
| Vote                | 2        | 16          | 435   | Social science|
| Soybean             | 19       | 35          | 683   | Life sciences |
| BreastCancer        | 2        | 9           | 699   | Life sciences |
| Tic-Tac-Toe Endgame | 2        | 9           | 958   | Sports competition |
| GermanCredit        | 2        | 20          | 1000  | Business finance |
| Car                 | 4        | 6           | 1728  | Business finance |
| Chess (King-Rook vs. King-Pawn) | 2 | 36       | 3196  | Sports competition |

### Table 4

Comparison of WF1-score on various data sets

| Dataset             | KNN     | NB      | RF      | XGBoost  | SVM     | K²NN    | SVM-S   |
|---------------------|---------|---------|---------|----------|---------|---------|---------|
| Promoters           | 0.819   | 0.953   | 0.916   | 0.953    | 0.934   | 0.842   | 0.972   |
| Dermatology         | 0.965   | 0.973   | 0.972   | 0.956    | 0.947   | 0.976   | 0.990   |
| Vote                | 0.925   | 0.920   | 0.963   | 0.968    | 0.954   | 0.955   | 0.981   |
| Soybean             | 0.922   | 0.948   | 0.938   | 0.942    | 0.939   | 0.925   | 0.962   |
| BreastCancer        | 0.955   | 0.958   | 0.967   | 0.954    | 0.974   | 0.956   | 0.988   |
| Tic-Tac-Toe         | 0.987   | 0.686   | 0.974   | 0.983    | 0.987   | 0.969   | 0.997   |
| GermanCredit        | 0.721   | 0.740   | 0.748   | 0.739    | 0.623   | 0.736   | 0.892   |
| Car                 | 0.925   | 0.827   | 0.968   | 0.993    | 0.983   | 0.939   | 0.986   |
| Chess               | 0.965   | 0.862   | 0.991   | 0.994    | 0.991   | 0.978   | 0.993   |
| Avg. WF1-Score      | 0.910   | 0.874   | 0.937   | 0.942    | 0.926   | 0.920   | 0.973   |

Bold values indicate the best results
The highest WF1-score on each data set is highlighted in bold typeface.

From Table 4, we can see that our SVM-S achieves the highest classification score on seven data sets (BreastCancer, Promoters, Soybean, Dermatology, Vote, GermanCredit and Tic-Tac-Toe). On the Chess data set, SVM-S obtains comparable accuracy to XGBoost; in fact, the classification performance of Random Forest, XGBoost, SVM and SVM-S on this set are approximately equivalent, all reaching a high classification score of more than 99%. SVM-S is slightly worse than XGBoost on the Car set, due to the fact that the set is extremely imbalanced (the numbers of samples in the major class and the smallest class are 1210 and 65, respectively). Overall, our SVM-S significantly outperforms KNN, Random Forest, SVM and $K^2$NN algorithms for symbolic data classification. Moreover, it can be more accurate than XGBoost, the state-of-the-art classification algorithm, as evidenced by the average WF1-scores shown in the last line of Table 4, which are 0.973 and 0.942, respectively.

Our SVM-S can be viewed as an SVM variant that specially designed for symbolic data classification. Its excellent performance in symbolic data classification is mainly due to the use of our newly formulated inner product and distance measure for symbolic data. This shows that our proposed methods not only provide a new solution for applying the Mercer kernel learning method to symbolic data mining, but also obtain better performance than other commonly used algorithms.

5.3 Attribute-weight analysis

To gain insights into the good performance of our SVM-S, we now focus on experimentally analyzing the kernel-based data transformation model KDTM. As discussed in Section 4.2, converting symbolic data objects into numeric vectors using our KDTM is equivalent to weighting each symbolic
attribute according to its data dispersion. To demonstrate the effectiveness of the kernel-based weighting scheme, in this set of experiments, the weights learned by KDTM for each attribute of the nine datasets are used for further analysis.

As shown in Eq. (11), the weight assigned to the $d$th symbolic attribute equals to $2(1 - \lambda_d)^2$, with the bandwidth $\lambda_d$ computed by Eq. (8). To provide context, we chose the entropy-based attribute weighting method (Zhou et al. 2016) for comparisons, which is defined by $\frac{1 - \text{entropy}(A_d)}{N - \sum_{k=1}^{N} \text{entropy}(A_k)}$, where \text{entropy}(A_d) denotes the entropy of the $d$th attribute in terms of the category distribution. For convenience, we use $w_{\text{kernel}}$ and $w_{\text{entropy}}$ to denote the two kinds of attribute-weights, respectively. The weights learned by different methods on the nine datasets are shown in Fig. 1.

To examine the relationship between the two sets of the weights, the Pearson correlation coefficient was used, computed by $\rho_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$. Here, $X$, $Y$ denote $w_{\text{kernel}}$ and $w_{\text{entropy}}$, respectively. From the figure, an obvious positive correlation between $w_{\text{kernel}}$ and $w_{\text{entropy}}$ on the same dataset can be observed. The values of Pearson correlation coefficient between $w_{\text{kernel}}$ and $w_{\text{entropy}}$ are beyond 0.9 on the eight data sets except BreastCancer, on which the correlation coefficient is larger than 0.87. We also observe that the weights $w_{\text{kernel}}$ and $w_{\text{entropy}}$ are precisely equal on the Car data set; this is because the data for each attribute in this set is uniformly distribution (note that our KDTM is an unsupervised data transformation model). This means that the attribute weighting scheme implied in our KDTM behaves similarly to the entropy-based method; this, consequently, provides our model with more capacity to distinguish between symbolic attributes.

### 6 Concluding remarks

In this work, we first use a kernel smoothing method to construct the kernel probability estimation model for symbolic data, and proved its convergence and consistency. By doing so, we then propose a kernel-based data transformation model, called KDTM, to embed symbolic data into Euclidean space. We also define new measures for the inner product and kernel-based weighted distance computation for symbolic objects. Finally, we extend the traditional SVM to SVM-S (i.e., SVM for symbolic data) by using the newly defined measures for nonlinear classification on symbolic data. The performance of the proposed methods is evaluated on nine real-world symbolic data sets, and the experimental results show their outstanding classification accuracy outperforming popular methods.

The important enlightenment of SVM is that some kind of nonlinear transformation can be achieved by the inner product based on the kernel learning method, for example, kernel principal component analysis (KPCA) (Wang and Tanaka 2016). Therefore, our work in this paper (e.g., the new kernel-based inner product measure) can help to extend more related methods to non-linearly mining on symbolic data. Another interesting extension would be to extend our kernel-based method to learning more complex data, such as mixed-type data and multi-variate time series.

### Acknowledgements

X. Yan, L. Chen and G. Guo’s work was supported by the National Natural Science Foundation of China under Grant Nos. U1105263, 61976053. X. Yan’s work was also supported by the National Natural Science Foundation of China under Grant No. 61772004 and the Guiding Foundation of Fujian Province of China No. 2020H0011.

### Declaration

**Conflict of interest** The authors declare that they have no conflict of interest.

### Appendices

#### A Proof of Theorem 1

Since $[I(\cdot)]^2 = I(\cdot)$ and $\sum_{o\in O_d} [p(o)] = 1$, the expectation of $\hat{p}(o_d|\lambda_d)$ can be obtained from Eq. (4):

\[
E(\hat{p}(o_d|\lambda_d)) = E[\ell(Y_{d_0}, o_d, \lambda_d)] = \sum_{o\in O_d} \left[\frac{1}{|O_d|} \lambda_d + (1 - \lambda_d) I(o = o_d) \right] p(o) = \frac{\lambda_d}{|O_d|} + (1 - \lambda_d) p(o) = p(o).
\]

So, the Bias($\hat{p}(o_d|\lambda_d)$) and the Var($\hat{p}(o_d|\lambda_d)$) can be computed as:

\[
\text{Bias}(\hat{p}(o_d|\lambda_d)) = \left[ \frac{\lambda_d}{|O_d|} - \lambda_d p(o) \right] \left[ \frac{1}{|O_d|} - p(o) \right],
\]

and

\[
\text{Var}(\hat{p}(o_d|\lambda_d)) = \frac{1}{N} \sum_{o\in O_d} \left[ \frac{\lambda_d}{|O_d|} + (1 - \lambda_d) I(o = o_d) \right]^2 p(o).
\]
\[ \frac{1}{N} [(1 - \lambda_d) p(\omega_{dI}) - (1 - \lambda_d) p(\omega_d)]^2 \]

By combining the above two equalities, the theorem is proven.

\section*{B Proof of Theorem 2}

For each \( \omega_{dI} \) in Eq. (6), we have that

\[ E \left[ ((1 - \lambda_d) f(\omega_{dI}) + \frac{\lambda_d}{|O_d|} - p(\omega_{dI}))^2 \right] \]
\[ = (1 - \lambda_d)^2 E \left[ f^2(\omega_{dI}) \right] \]
\[ + 2 \left[ \frac{\lambda_d - 1}{|O_d|} \lambda_d + (\lambda_d - 1) p(\omega_{dI}) \right] \]
\[ E \left[ f(\omega_{dI}) \right] + [p(\omega_{dI})]^2 - \frac{2\lambda_d}{|O_d|} p(\omega_{dI}) + \frac{\lambda_d}{|O_d|^2}. \]

Base on the facts that \( E \left[ f(\omega_{dI}) \right] = p(\omega_{dI}) \) and \( [I(\cdot)]^2 = I(\cdot) \), the above equality can be simplified as

\[ (1 - \lambda_d)^2 \left( E \left[ f^2(\omega_{dI}) \right] - (E \left[ f(\omega_{dI}) \right])^2 \right) \]
\[ + (1 - \lambda_d)^2 p^2(\omega_{dI}) + 2 \left[ \frac{\lambda_d - 1}{|O_d|} \lambda_d + (\lambda_d - 1) p(\omega_{dI}) \right] \]
\[ p(\omega_{dI}) + p^2(\omega_{dI}) - \frac{2\lambda_d}{|O_d|} p(\omega_{dI}) + \frac{\lambda_d}{|O_d|^2} \]
\[ = (1 - \lambda_d)^2 p^2(\omega_{dI}) + \lambda_d^2 p(\omega_{dI}) \]
\[ - \frac{2\lambda_d}{|O_d|} p(\omega_{dI}) + \frac{\lambda_d}{|O_d|^2} \]
\[ = \frac{\lambda_d}{|O_d|^2} - \left( \frac{1 - \lambda_d}{N} \right)^2 \]
\[ p^2(\omega_{dI}) + \left[ (1 - \lambda_d)^2 - \frac{2\lambda_d}{|O_d|} \right] p(\omega_{dI}) \]
\[ + \frac{\lambda_d}{|O_d|^2}. \]

Therefore, \( L(\lambda_d) \) can be computed as

\[ L(\lambda_d) = \left[ \frac{\lambda_d}{|O_d|^2} - \left( \frac{1 - \lambda_d}{N} \right)^2 \right] \sum_{\omega_{dI} \in \omega_{dI}} \left[ p(\omega_{dI}) \right]^2 \]
\[ + (1 - \lambda_d)^2 - \frac{\lambda_d}{|O_d|} \]
\[ = \left( 1 - \frac{1}{|O_d|} \right) \lambda_d^2 + \left( 1 - \lambda_d \right)^2 - \frac{\lambda_d}{|O_d|} \sigma^2. \]

Let \( \frac{\partial L(\lambda_d)}{\partial \lambda_d} = 0 \), we get the optimal estimate of \( \lambda_d \), and Eq. (7).

\section*{References}

Agresti A (2008) An introduction to categorical data analysis. Wiley, New York

Aitchison J, Aitken CGG (1976) Multivariate binary discrimination by the kernel method. Biometrika 63(3):413–420

Alayya MZ, Bussy S, Gaifffas S, Guillox A (2017) Binarsity: a penalization for one-hot encoded features. J Machine Learn Res 20:1–34

Boriah S, Chandola V, Kumar V (2008) Similarity measures for categorical data: a comparative evaluation. In: Proceedings of the 8th SIAM international conference on data mining, pp 243–254

Breiman L (2001) Random forests. Machine Learn 45(1):5–32

Bremner AP, Taplin RH (2002) Theory & methods: modified classification and regression tree splitting criteria for data with interactions. Aust & N Z J Stat 44(2):169–176

Buttrey SE (1998) Nearest-neighbor classification with categorical variables. Comput Stat Data Anal 28(2):157–169

Casquillo JP (2020) On the weighted gini-simpson index: estimating feasible weights using the optimal point and discussing a link with possibility theory. Soft Comput 24(22):17187–17194

Cerda P, Varoquaux G, Kégl B (2018) Similarity encoding for learning with dirty categorical variables. Machine Learn 107(8–10):1477–1494

Chen L, Guo G (2015) Nearest neighbor classification of categorical data by attributes weighting. Expert Syst Appl 42(6):3142–3149

Chen L, Ye Y, Guo G, Zhu J (2016) Kernel-based linear classification on categorical data. Soft Comput 20(8):2981–2993

Chen L, Wang S, Wang K, Zhu J (2016) Soft subspace clustering of categorical data with probabilistic distance. Pattern Recognit 51:322–332

Cheng L, Wang Y, Ma X (2019) A neural probabilistic outlier detection method for categorical data. Neurocomputing 365:325–335

Chen T, Guestrin C (2016). XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining (KDD’16), pp 785–794

Chen L, Guo G, Wang S, Kong X (2014b). Kernel learning method for distance-based classification of categorical data. In: Proceedings of the 14th UK workshop on computational intelligence (UKCI’14), pp 58–63

Chen L, Wang S (2013). Central clustering of categorical data with automated feature weighting. In: Proceedings of the 23th international joint conference on artificial intelligence (IJCAI’13), pp 1260–1266

Cortes C, Vapnik V (1995) Support-vector networks. Machine Learn 20:273–297

Deng G, Manton JH, Wang S (2018) Fast kernel smoothing by a low-rank approximation of the kernel toepitz matrix. J Math Imaging Vis 60(8):1181–1195

Dos Santos TRL, Zárate LE (2015) Categorical data clustering: What similarity measure to recommend? Expert Syst Appl 42(3):1247–1260

Ghosh S (2018) Kernel smoothing principles. Wiley, Hoboken

Guha S, Rastogi R, Shim K (2000) ROCK: a robust clustering algorithm for categorical attributes. Inform Syst 25(5):345–366

Han E, Karypis G (2000). Centroid-based document classification: analysis & experimental results. In: Proceedings of the 4th European conference on principles and practice of knowledge discovery in databases (PKDD’00), pp 424–431

He Z, Xu X, Deng S (2018) K-ANMI: a mutual information based clustering algorithm for categorical data. Inform Fusion 36(3):1171–1220

Hofmann T, Schölkopf B, Smola AJ (2008) Kernel methods in machine learning. Ann Stat 36(3):1171–1220
Huang Z (1998) Extensions to the K-means algorithm for clustering large data sets with categorical values. Data Mining Knowl Discovery 2(3):283–304
Huang JZ, Ng MK, Rong H, Li Z (2005) Automated variable weighting in K-means type clustering. IEEE Trans Pattern Anal Machine Intell 27(5):657–668
Jin W, Li ZJ, Wei LS, Zhen H (2000). The improvements of BP neural network learning algorithm. In: Proceedings of the 5th international conference on signal processing, pp 1647–1649
Larochelle H, Mandel M, Pascanu R, Bengio Y (2012) Learning algorithms for the classification restricted boltzmann machine. J Machine Learn Res 13(1):643–669
Li Q, Racine JS (2007) Nonparametric econometrics: theory and practice. Princeton University Press, Princeton
Mikolov T, Chen K, Corrado G, Dean J (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781
Ouyang D, Li Q, Racine JS (2006) Cross-validation and the estimation of probability distributions with categorical data. J Nonparametric Stat 18(1):69–100
Qian Y, Li F, Liang J, Liu B, Dang C (2016) Space structure and clustering of categorical data. IEEE Trans Neural Netw Learn Syst 27(10):2047–2059
Quinlan J (1995). C4.5: Programs for machine learning. Morgan Kaufmann Publishers Inc
Roweis S, Saul L (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290:2323
Scott DW (1992) Multivariate density estimation: theory, practice, and visualization. Wiley, New York
Seeger M (2006). Bayesian modeling in machine learning: a tutorial review. Tutorial, Saarland University. http://lapmal.epfl.ch/papers/bayes-review
Stone CJ (1984) An asymptotically optimal window selection rule for kernel density estimates. Ann Stat 12(4):1285-1297
Vo KT, Sowmya A (2010). Multiple kernel learning for classification of diffuse lung disease using HRCT lung images. In: Proceedings of the 2010 annual international conference of the IEEE engineering in medicine and biology, pp 3085–3088
Wang MQ, Yue XD, Gao C, Chen Y (2018). Feature selection ensemble for symbolic data classification with AHP. In: Proceedings of the 24th international conference on pattern recognition (ICPR’08), pp 868–873
Wang Z, Zhu Z, Li D (2020) Collaborative and geometric multi-kernel learning for multi-class classification. Pattern Recognit 99:107050
Wang R, Li Z, Cao J, Chen T, Wang L (2019). Convolutional recurrent neural networks for text classification. In: Proceedings of the 2019 international joint conference on neural networks (IJCNN), pp 1–6
Wang D, Tanaka T (2016). Sparse kernel principal component analysis based on elastic net regularization. In: Proceedings of the 2016 international joint conference on neural networks (IJCNN), pp 3703–3708
Yan X, Chen L, Guo G (2018) Center-based clustering of categorical data using kernel smoothing methods. Front Computer Sci 12(5):1032–1034
Zhang J, Chen L, Guo G (2013) Projected-prototype-based classifier for text categorization. Knowl Based Syst 49:179–189
Zhong S, Chen T, He F, Niu Y (2014) Fast gaussian kernel learning for classification tasks based on specially structured global optimization. Neural Netw 57:51–62
Zhou J, Chen L, Chen CLP, Zhang Y, Li HX (2016) Fuzzy clustering with the entropy of attribute weights. Neurocomputing 198(19):125–134
Zhu S, Xu L (2018) Many-objective fuzzy centroids clustering algorithm for categorical data. Expert Syst Appl 96:230–248

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.