Information Sciences 367–368 (2016) 673–688

Contents lists available at ScienceDirect

Information Sciences

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

A graph-based semi-supervised *k* nearest-neighbor method for nonlinear manifold distributed data classification

Enmei Tu[a,*], Yaqian Zhang[b], Lin Zhu[c], Jie Yang[d], Nikola Kasabov[e]

[a] *Rolls-Royce@NTU Corporate Lab, Nanyang Technological University, Singapore*
[b] *School of Computer Science and Engineering, Nanyang Technological University, Singapore*
[c] *School of Computer Science and Technology, Shanghai University of Electric Power, China*
[d] *Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, China*
[e] *The Knowledge Engineering and Discovery Research Institute, Auckland University of Technology, New Zealand*

ARTICLE INFO

Article history:
Received 27 September 2015
Revised 28 June 2016
Accepted 6 July 2016
Available online 6 July 2016

Keywords:
k Nearest neighbors
Manifold classification
Constrained tired random walk
Semi-Supervised learning

ABSTRACT

k nearest neighbors (*k*NN) is one of the most widely used supervised learning algorithms to classify Gaussian distributed data, but it does not achieve good results when it is applied to nonlinear manifold distributed data, especially when a very limited amount of labeled samples are available. In this paper, we propose a new graph-based *k*NN algorithm which can effectively handle both Gaussian distributed data and nonlinear manifold distributed data. To achieve this goal, we first propose a constrained Tired Random Walk (TRW) by constructing an *R*-level nearest-neighbor strengthened tree over the graph, and then compute a TRW matrix for similarity measurement purposes. After this, the nearest neighbors are identified according to the TRW matrix and the class label of a query point is determined by the sum of all the TRW weights of its nearest neighbors. To deal with online situations, we also propose a new algorithm to handle sequential samples based a local neighborhood reconstruction. Comparison experiments are conducted on both synthetic data sets and real-world data sets to demonstrate the validity of the proposed new *k*NN algorithm and its improvements to other version of *k*NN algorithms. Given the widespread appearance of manifold structures in real-world problems and the popularity of the traditional *k*NN algorithm, the proposed manifold version *k*NN shows promising potential for classifying manifold-distributed data.

1. Introduction

k Nearest Neighbors (*k*NN) [8,42,43] is one of the most popular classification algorithms and has been widely used in many fields, such as anomaly detection [3], face super-resolution [19], early prediction of diabetes mellitus [47] and very large database manipulation [21], because it is simple but effective, and can generally obtain good results in many tasks. One main drawback of the traditional *k*NN is that it does not take the manifold distribution information into account and this can cause bias which results in bad performance. It becomes even worse when there are only a very small amount of labeled samples available. To address this, an example is shown in Fig. 1(a), in which there are two one-dimensional manifolds (the outer arch and the interior reflected *S*) which correspond to two classes, respectively. Each class has only

* Corresponding author.
E-mail address: hellotem@hotmail.com (E. Tu).

http://dx.doi.org/10.1016/j.ins.2016.07.016
0020-0255/

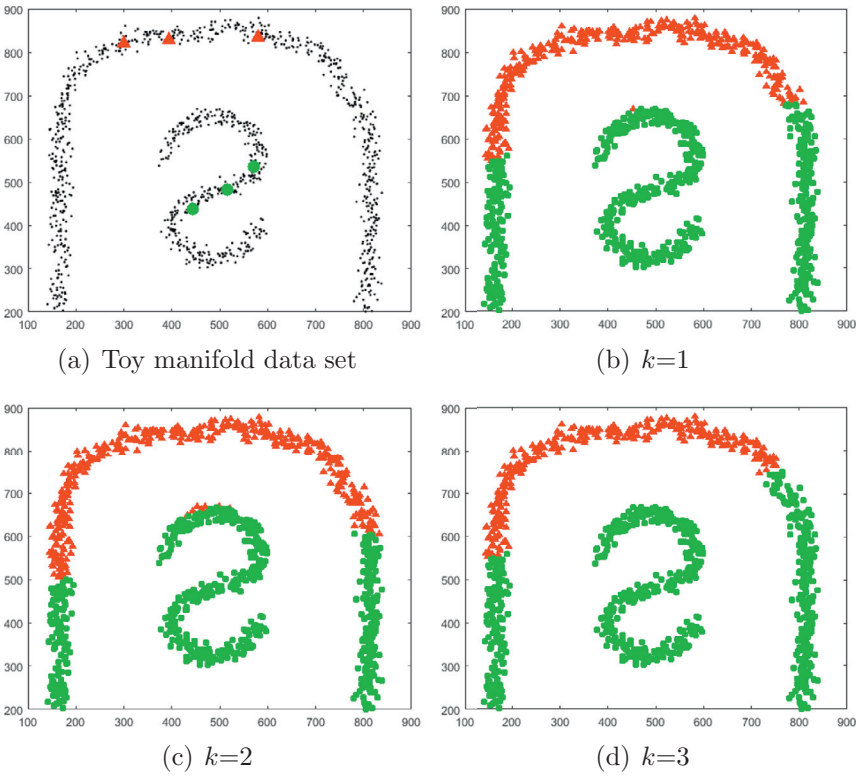


Fig. 1. Results of traditional *k*NN classification with $k = 1, 2, 3$ on manifold distributed data, in which the colored shapes (red triangles and green dots) are labeled samples and the black dots are unlabeled samples.

3 labeled samples, indicated by the colored triangles and circles. Black dots are unlabeled samples. Fig. 1(b)–(d) show the 1NN, 2NN and 3NN classification results produced by the traditional *k*NN, respectively. We can see that although the data have apparent manifold distribution, the traditional *k*NN incorrectly classifies many samples due to ignoring the manifold information.

To improve the performance of the traditional *k*NN, some new *k*NN algorithms have been proposed. Hastie et. al. [17] proposed an adaptive *k*NN algorithm which computes a local metric for each sample and uses Mahalanobis distance to find the nearest neighbors of a query point. Hechenbichler and Schliep [18] introduced a weight scheme to attach different importance to the nearest neighbors with respect to their distances to the query point. To reduce the effect of unbalanced training set sizes of different classes, Tan [30] used different weights for different classes regarding the number of labeled samples in each class. There are also some other improvements and weight schemes for different tasks [9,11,16,41].

However, none of these new *k*NN algorithms takes manifold structure into consideration explicitly. For high-dimensional data, such as face images, documents and video sequences, the nearest neighbors of a point found by traditional *k*NN algorithms can be very far in terms of the geodesic distance between them, because the dimension of the underlying manifold is usually much lower than that of the data space [2,29,32]. There have also been attempts to make *k*NN adaptive to manifold data. In Turaga and Chellappa's paper [37], geodesic distance is used to directly replace standard Euclidean distance in traditional *k*NN, but geodesic distance can be computed with good accuracy only if the manifold is sampled with sufficient points. Furthermore, geodesic distance tends to be very sensitive to short-circuit phenomenon. Li [24] proposed a weighted manifold *k*NN using Local Linear Embedding (LLE) techniques, but LLE tends to be unstable due to local changes on the manifold. Percus and Olivier [26] studied the general k^{th} nearest neighbor distance metric on close manifold, but their method needs to know exactly the analytical form of the manifold and thus is unsuitable for most real-world applications.

In this paper, we propose a novel graph-based *k*NN algorithm which can effectively handle both traditional Gaussian distributed data and nonlinear manifold distributed data. To do so, we first present a manifold similarity measure method, the constrained tired random walk, and then we modify the traditional *k*NN algorithm to adopt the new measuring method. To deal with online situations, we also propose a new algorithm to handle sequential samples based on a local neighborhood reconstruction method. Experimental results on both synthetic and real-world data sets are presented to demonstrate the validity of the proposed method.

The remainder of this paper is organized as follows: Section 2 reviews the tired random walk model. Section 3 presents a new constrained tired walk random walk model and Section 4 describes the graph-based *k*NN algorithm. Section 5 proposes

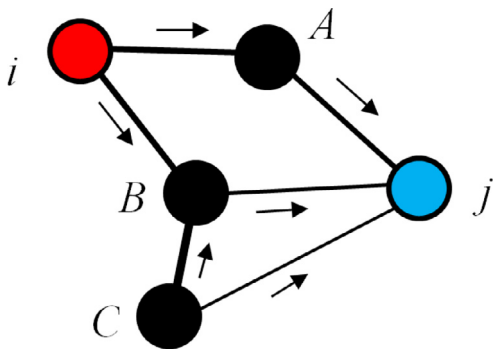


Fig. 2. Tired random walk on weighted undirected graph.

a sequential algorithm for online samples. The simulation and comparison results are presented in Section 6, followed by conclusions in Section 7.

2. Review of tired random walk

Assume a training set $\mathcal{X}_T = \{x_1, x_2, \ldots, x_{l-1}, x_l\} \subset \mathbb{R}^d$ contains l labeled samples and the class label of x_i is y_i , $y_i \in \{1, 2, \ldots, C\}; i = 1 \ldots l$, where d is feature length and C is class number. There are also $n - l$ unlabeled samples to be classified, $\mathcal{X}_U = \{x_1, x_2, \ldots, x_{n-l}\}$. Denote $\mathcal{X} = \mathcal{X}_T \cup \mathcal{X}_U = \{x_1, x_2, \ldots, x_n\}$ and $y = (y_1, y_2, \ldots, y_l)$. We also use X for the sample matrix, whose columns are the samples in $\mathcal{X}$. We use $\| \cdot \|$ to denote the Frobenius norm.

In differential geometry studies, a manifold can be defined from an intrinsic or extrinsic point of view [4]. But in data processing studies, such as dimension reduction [25,27] and manifold learning [1,38], it is helpful to consider a manifold as a distribution which is embedded in a higher Euclidean space, i.e. adopting an extrinsic view in its ambient space. Borrowing the concept of *intrinsic dimension* from [5], we give a formal definition of a manifold data set as follows.

Definition: *A data set is considered to be manifold distributed if its intrinsic dimension is less than its data space dimension.*

For more information on the intrinsic dimensions of a data set and how it can be estimated from data samples, we refer readers to [5,27]. To determine whether a data set has manifold distribution, one can simply estimate its intrinsic dimension and compare it with the data space dimension, i.e. the length of a sample vector.

Similarity measure is an important factor while processing manifold distributed data, because on manifolds traditional distance metrics (such as Euclidean distance) are not a proper measure [45,46]. Recent studies have also proved that classical random walk is not a useful measure for large sample cases or high-dimensional data because it does not take any global properties of the data into account [23]. The tired random walk (TRW) model was proposed in Tu's paper [33] and has been demonstrated to be an effective measure of nonlinear manifold [39,44], because it takes global geometrical structure information into consideration.

Recall that on a weighted undirected graph, the classical random walk transition matrix is $P = D^{-1}W$, where W is the graph adjacency matrix and D is a diagonal matrix with entries $D_{ii} = \sum_{j=1}^{n} W_{ij}$. Now imagine that a tired random walker walks continuously through edges in a graph, but it becomes more tired after each walk and finally stops after all energy is exhausted, i.e. the transition probability of the random walk reduces with a fixed ratio (e.g. 0.01) after each walk and finally approaches 0. After t steps the tired random walk transition probability matrix becomes $(0.01P)^t$. Now considering Fig. 2, the tired random walker starts from vertex i and its destination is vertex j on the graph, walking with a strength reduction rate $\alpha \in (0, 1)$. Then it may walk through any path that connects vertices i and j, with an arbitrary number of steps before its strength is used up. For example, the tired random walker can walk through path $i \to A \to j$, or $i \to B \to j$. It can also walk through $i \to B \to C \to j$, or even $i \to B \to C \to B \to j$, because while at vertex C, the probability of walking to B is much larger than that to j. But all these walking paths reflect the underlying geometrical structure of the graph, hence the distribution of the data. Therefore, a good similarity measure between vertex i to j should take (globally) all possible paths and an arbitrary number of steps (can potentially be infinite) into consideration, rather than only considering (locally) a single path or a single step as the classical random walk [23] does. This makes a fundamental difference between the *tired random walk* and *classical random walk* and entitles the tired random walk to be more robust and effective, especially for manifold distributed data, as will be demonstrated in the experimental section. Mathematically, the accumulated transition probability of the tired random walk between vertex i and j is $(P_{TRW})_{ij} = \left[\sum_{t=0}^{\infty} (\alpha P)^t\right]_{ij}$. For all vertices, the accumulated transition probability matrix becomes $P_{TRW} = \sum_{t=0}^{\infty} (\alpha P)^t$. As the eigenvalue of P is in $[-1, 1]$ and $\alpha \in (0, 1)$, the series converges and the tired random walk matrix is

$$P_{TRW} = \sum_{t=0}^{\infty} (\alpha P)^t = (I - \alpha P)^{-1} \tag{1}$$

Because P_{TRW} takes all the possible paths into account, it captures the global geometrical structure information of the underlying manifold and thus is demonstrated to be more effective and robust to describe manifold similarity.

Table 1
Constrained graph construction.

Steps	
1	$W_{ij} = 1$ if $x_i, x_j \in \mathcal{X}_T$ and have same class label.
2	$W_{ij} = 0$ if $x_i, x_j \in \mathcal{X}_T$ and have different class labels.
3	$W_{ij} = \exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma^2\right)$ if at least one of x_i, x_j is unlabeled.
4	$W_{ij} = (1 + \theta_{ij}^r) W_{ij}$, $1 \le r \le R$, if x_i is a node in level $r-1$ and x_j is a child of x_i in level r in the strengthened tree.

3. A constrained tired random walk model

In this section, we further extend the model into a constrained situation. For classification purposes, we find that labeled samples provide not only class distribution information, but also constraint information, i.e., samples which have the same class labels are must-link pairs and samples which have different class labels are cannot-link pairs. In most of the existing supervised learning algorithms, only class information is utilized but constraint information is discarded. Here we include constraint information into the TRW model by modifying the weights of graph edges between the labeled samples, because constraint information has been demonstrated to be useful for performance improvement [10,14,36]. Class information will be utilized in the next section for the proposed new *k*NN algorithm.

We first construct an R-level nearest-neighbor strengthened tree for each labeled sample $x_i \in \mathcal{X}_T$ as follows:

(i) Set x_i as the first level node (tree root node, $r = 0$) and its k nearest neighbors as the second level nodes.
(ii) For each node in level $r-1$, set its nearest neighbors as its level r descendants. If any node in level r appears in its ancestor level, remove it from level r.
(iii) If $r < R$, go to (ii).

where R is a user-specified parameter to define the depth of the tree. Then for each pair of samples (x_i, x_j), the corresponding graph edge weight is set according to the rules in Table 1.

σ is the Gaussian kernel width parameter and θ is the strengthening parameter. Note that for the Gaussian kernel, step 1 is equivalent to merging the two same-label samples together (0 distance between them) and step 2 is equivalent to separating the two different-label samples from each other to infinitely far (infinite distance between them). The connections from a labeled sample to its nearest neighbors are strengthened by θ_{ij} in step 4. This can spread the hard constraints in steps 1 and 2 to farther neighborhoods on the graph in a form of *soft* constraints and thus causes these constraints to have a wider influence. The motivation of constructing the strengthened tree is inspired by the neural network reservoir structure analysis techniques, in which information has been shown to spread out from input neurons to interior neurons in the reservoir following a tree-structure path [34].

The selection of parameter θ is based the following conditions

- the strengthened weight should be positive and less than the weight of the must-link constraint.
- the strengthening effect should be positive and decays along the strengthened tree level.

Mathematically, the conditions are

$$\begin{cases} 0 < W_{ij} + \theta_{ij} W_{ij} < 1 \\ 0 < \theta_{ij}^{r+1} < \theta_{ij}^{r} \end{cases} \tag{2}$$

As a result, θ should be

$$0 < \theta_{ij} < \min\left(\frac{1 - W_{ij}}{W_{ij}}, 1\right) = \bar{\theta} \tag{3}$$

In all our experiments, we used a single value of $\theta = 0.1\bar{\theta}$, which gives good results for both synthetic and real-world data.

4. A new graph-based *k*NN classification algorithm on nonlinear manifold

Here we present a graph-based *k*NN algorithm for nonlinear manifold data classification. The procedure of the algorithm is summarized in Table 2.

Specifically, given P_{TRW} matrix, the TRW weight between sample x_i and x_j is defined as

$$\bar{w}_{ij} = w(x_i, x_j) = \frac{(P_{TRW})_{ij} + (P_{TRW})_{ji}}{2} \tag{4}$$

Note that while the similarity measure defined by matrix P_{TRW} between two samples is not necessarily symmetric (P is not symmetric and thus its matrix series P_{TRW} is also not symmetric), the weight defined in Eq. (4) is indeed a symmetric

Table 2
A graph-based *k*NN algorithm.

Steps	
1	Input $\mathcal{X} = \mathcal{X}_T \cup \mathcal{X}_U$, y and k.
2	Construct a constrained graph according to Table 1
3	Compute the P_{TRW} using Eq. (1)
4	Evaluate samples' similarity according to Eq. (4)
5	Find nearest neighbors of an unlabeled sample using Eq. (5)
6	Determine the class label according to Eq. (6)

measure. For each unlabeled sample $x \in \mathcal{X}_U$, we could find its $k \leq l$ nearest neighbors from $\mathcal{X}_T$ by

$$x_i = \arg\max_{x_j \in \mathcal{X}_T} w(x, x_j) \tag{5}$$

Instead of counting the number of labeled samples from each class in the classical *k*NN, we sum the TRW weights of the labeled samples of each class and the class label of the unlabeled sample x is determined by

$$y = \arg\max_{c=1,2,\ldots,C} \sum\nolimits_{i=1}^{k} w(x, x_i) I(y_i = c) \tag{6}$$

It is worth mentioning that because the proposed semi-supervised m*k*NN utilizes class label information only in the classifying stage and it uses the same k value equally for all classes, m*k*NN naturally lacks the so-called class bias problem in many semi-supervised algorithms (such as [15,28,35]) that is due to the influence of unbalanced labeled samples of each class in $\mathcal{X}_T$ and needs to be re-balanced by various weighting schemes [40,48]. Furthermore, with the algorithm described in the next section, m*k*NN immediately becomes a supervised classifier and enjoys the advantages of both semi-supervised learning (classifying abundant unlabeled samples with only a tiny number of labeled samples) and supervised learning (classifying new samples immediately without repeating the whole learning process).

5. A sequential method to handle online samples

For one single unlabeled sample, if we compute its TRW weights using Eqs. (1) and (4), we have to add it to $\mathcal{X}$ and re-compute the matrix P_{TRW}. As a result, the computational cost for *one* sample is too high. Actually this is the so-called transductive learning problem[1], a common drawback of many existing algorithms [7,12,20]. To attack this problem, we propose a new method based on rapid neighborhood reconstruction, in which a local neighborhood is first constructed in sample space and then the TRW weights can be reconstructed in the same local neighborhood with very trivial computational cost.

Given a new sample x, it has been shown that x can be well reconstructed by its nearest neighbors on the manifold if there are sufficient data points sampled from the manifold [6,29,31]. Thus, it is also reasonable to assume that the neighborhood relationships, hence the weights of sample x in Eq. (4), have the same geometrical distribution as the sample distribution. So, to compute the weights of x without explicitly recomputing matrix P_{TRW} in Eq. (4), we first find x's k nearest neighbors[2] in $\mathcal{X}$, written as X_k which contains these k nearest neighbors in its columns, and then minimize the local reconstruction error by solving the following constraint quadratic optimization problem

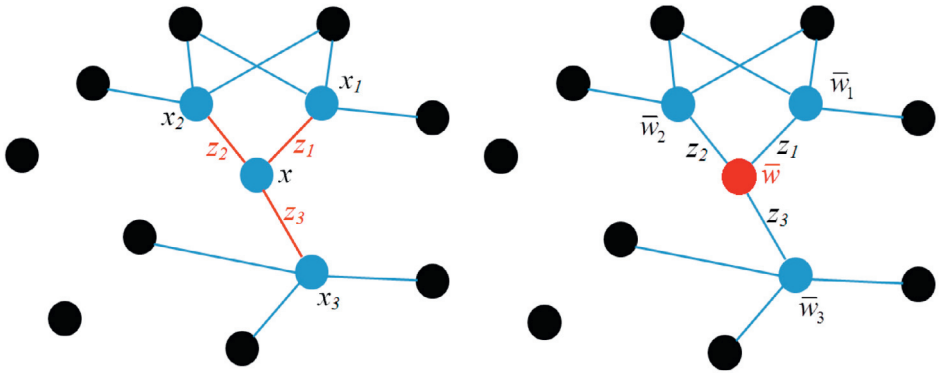
$$\begin{aligned} &\min \|x - X_k z\|^2 \\ &s.t.\ z \geq 0; z^T e = 1 \end{aligned} \tag{7}$$

where e is a vector with all entries being 1. Note that the entries of z are nonnegative and the sum of all entries must be 1, so z is expected to be sparse [22]. Problem (7) is a constrained quadratic optimization problem and can be solved very efficiently with many publicly available toolboxes. We use the OPTI optimization toolbox[3] to solve this. After obtaining the

[1] Transductive learning is an opposite concept to inductive learning. Inductive learning means the learning algorithm, such as SVM, learns a model explicitly in data space that partitions the data space into several different regions. Then the model can be applied directly to unseen samples to obtain the class labels. On the other side, transductive learning does not build any model. It performs one-time learning only on a fixed data set.Whenever the data set changes (for example, existing samples are changed or new samples are added.), the whole learning process has to be repeated again to assign new class labels.

[2] One should note that finding nearest neighbors in $\mathcal{X}$ is quite different from that in $\mathcal{X}_T$. The former is the basis of many nearest-neighbor operations (such as constructing nearest-neighbor graph in [29,32] and the R-level strengthened tree in Section 3 of this paper) and the latter is the basis of the classical *k*NN classifier. Because $\mathcal{X}$ contains many instances, which are sampled densely from the underlying data distribution, an instances's local neighborhood in $\mathcal{X}$ is usually very small and thus Euclidean distance is still valid in this small range for that any manifold can be locally well approximated by Euclidean space [4]. However, instances in $\mathcal{X}_T$ are very few and usually not densely sampled from the manifold and *nearest neighbors*in $\mathcal{X}_T$ can be very far. Thus Euclidean distance is no longer suitable for measuring the closeness of the points in $\mathcal{X}_T$. This is why we need other new similarity (or closeness) measure methods, which is one of the main contributions of this paper.

[3] OPTI TOOLBOX: http://www.i2c2.aut.ac.nz/Wiki/OPTI/



(a) Computing reconstruction weight z using equation (7) (b) Computing TRW weight $\bar{w}$ using equation (8)

Fig. 3. The process of computing sequential TRW weight.

Table 3
The procedure of sequential manifold *k*NN algorithm.

Steps	
1	Input $\mathcal{X} = \mathcal{X}_T \cup \mathcal{X}_U$, P_{TRW}, x and k.
2	Find x's k nearest neighbors in $\mathcal{X}$
3	Use Eqs. (7) and (8) to compute its TRW weights
4	Use Eq. (6) to classify it

optimal z, the TRW weights between x and its nearest labeled samples in X_k can be computed by solving the following optimization problem

$$\begin{aligned} &\min \left\| \bar{w} - \bar{W}_k^T z \right\|^2 \\ &s.t. \quad \bar{w} \geq 0 \end{aligned} \tag{8}$$

where $\bar{W}_k = (\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_k)$ and $\bar{w}_i$ contains the weights between sample x_i in X_k and its k nearest neighbors in X.

This process is illustrated in Fig. 3. Fig. 3(a) corresponds to Eq. (7) which computes z and Fig. 3(b) corresponds to Eq. (8) which computes $\bar{w}$.

It is easy to see that the optimal solution of problem (8) is simply the result of a nonnegative projection operation

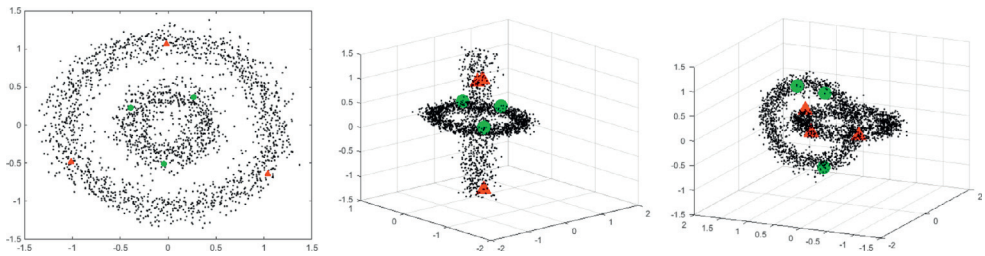
$$\bar{w} = \max(0, \bar{W}_k^T z) = \bar{W}_k^T z \tag{9}$$

The second equation holds because both $\bar{W}_k$ and z are nonnegative and therefore their multiplication result is also nonnegative.

One should note that with this sequential learning strategy, the proposed m*k*NN can be treated as an inductive classifier, whose model consists of both the P_{TRW} matrix and the data samples that have been classified so far. Whenever a new sample arrives, the model can quickly give its class label by the three steps in Table 3, without repeating the whole learning process in Table 2.

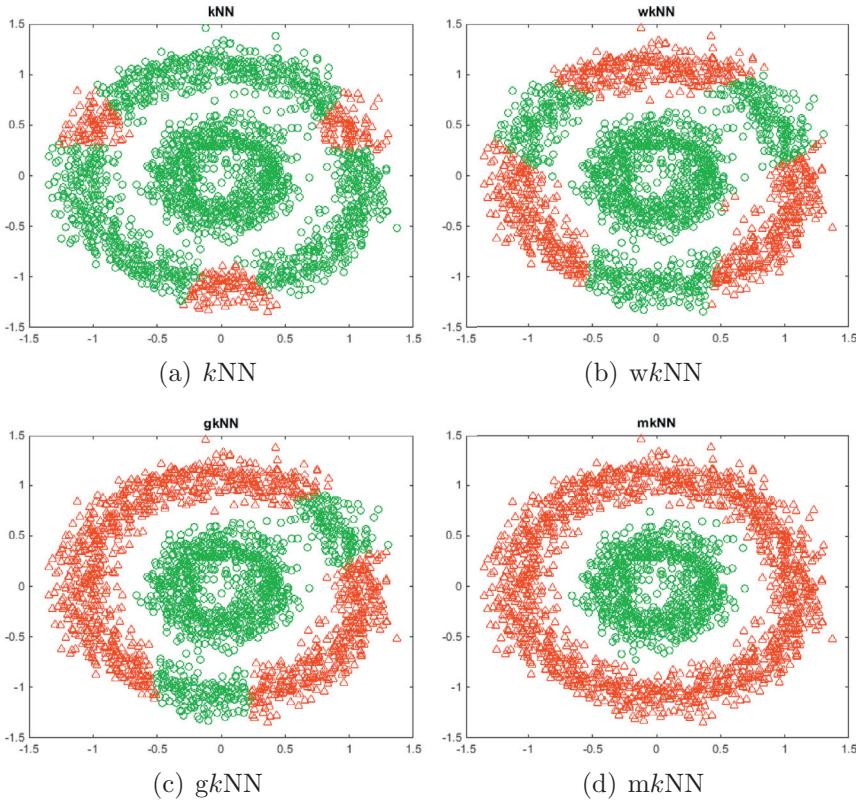
6. Experimental results

In this section, we report the experimental results on both the synthetic data sets and real-world data sets. The comparison algorithms include traditional *k*NN, the weighted k nearest neighbors (w*k*NN) and the geodesic *k*NN (g*k*NN) proposed by Pavan and Rama[37], as well as our manifold k nearest neighbors (m*k*NN). For *k*NN and w*k*NN, the only parameter is k. For g*k*NN and m*k*NN, there is one more parameter for each, i.e. the number of nearest neighbors for computing geodesic distance in g*k*NN and the kernel width σ in m*k*NN. We tune these two parameters by grid-search and choose their values to produce the minimal 2-fold cross validation error rate.



(a) data set 1 (b) data set 2 (c) data set 3

Fig. 4. Three synthetic data sets, in which the colored shapes are labeled samples and the black dots are unlabeled samples.



(a) kNN (b) wkNN

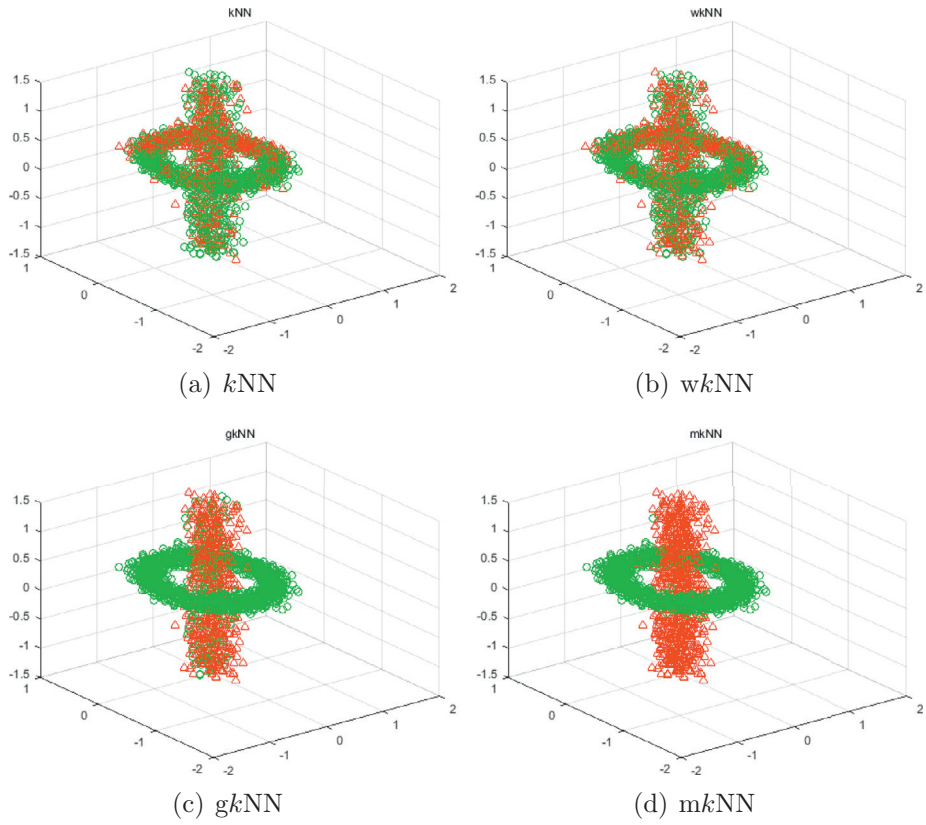
(c) gkNN (d) mkNN

Fig. 5. Experimental results on synthetic data sets. (a)–(d): results of kNN, wkNN, gkNN and mkNN on data set 1.

6.1. Experimental results on synthetic data sets

We first conduct experiments on three synthetic data sets shown in Fig. 4 to demonstrate the superiority of mkNN over other kNN algorithms. For each data set in Fig. 4, the three red triangles and the three green dots are the labeled samples of the two classes, respectively. Note that all three data sets contain some ambiguous points (or bridging points) in the gap between two classes, making the classification even more challenging. Experimental results on these data sets are shown in Figs. 5–7.

From these results, we can see that because kNN uses Euclidean distance to determine the class label and Euclidean distance is not a proper similarity measure on the manifold, the results given by traditional kNN are quite erroneous. By introducing a weight scheme, wkNN can perform better than kNN, but the improvement is still quite limited. gkNN has much better results because geodesic distance on the manifold is a valid similarity measure. However, as mentioned in Section 1, graph distance tends to be sensitive to short-circuit phenomenon (i.e. the noise points lying between the two classes), thus gkNN still misclassifies many samples due to the existence of noisy points. In contrast, mkNN achieves the best results, because TRW takes all the possible graph paths into consideration, thus it embodies the global geometrical structure information of the manifold and can be much more effective and robust to noisy points.



(a) kNN (b) wkNN (c) gkNN (d) mkNN

Fig. 6. Experimental results on synthetic data sets. (a)–(d): results of kNN, wkNN, gkNN and mkNN on data set 2.

Table 4
Reconstruction error of the synthetic data sets (%).

	data set 1	data set 2	data set 3
$RMSE(X, \hat{X})$	0.0724	0.2734	0.1606
$RMSE(P_{TRW}, \hat{P}_{TRW})$	0.5075	1.0524	1.9061

Fig. 8 plots the mean error rate of each algorithm over 10 runs on these data sets, as k changes from 1 to 10. In each run, k labeled samples are randomly selected from each class and the rest of the samples in the data set are treated as unlabeled samples to form the testing set.

From the results in Fig. 8 we can see that mkNN performs significantly better than other version of kNN algorithms. It is interesting to note that on these manifold distributed data sets, while the mean error rates of kNN and wkNN have no obvious reduction as k increases, the mean error rates of gkNN and mkNN decrease quickly. This indicates that gkNN and mkNN are able to exploit the information contained in the labeled samples in a more efficient way, because they take the manifold structure information into account. Again, mkNN achieves the lowest error rate.

In order to demonstrate the effectiveness of the weight reconstruction method, we run the algorithm on these data sets to reconstruct each sample and its TRW weight using its k nearest neighbours with equation (1.7) and (1.8), respectively. The relative mean square error (RMSE) of the reconstruction result is computed by

$$RMSE(T, R) = \frac{\|T - R\|^2}{\|T\|^2} \times 100\%$$

where T is the ground truth and R is the reconstructed result. The results are shown in Table 4. From these results we can see that the reconstruction method in Eqs. (7) and (8) can produce a very accurate approximation to the true samples and weights, respectively, on nonlinear manifold data sets.

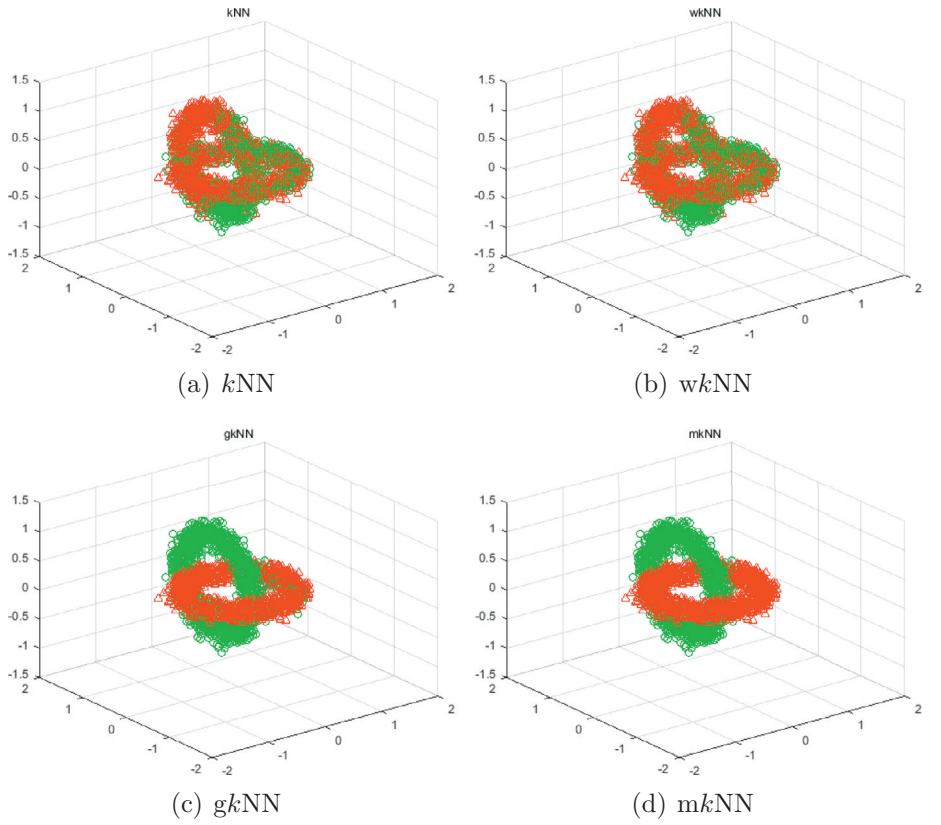


Fig. 7. Experimental results on synthetic data sets. (a)–(d): results of *k*NN, w*k*NN, g*k*NN and m*k*NN on data set 3.

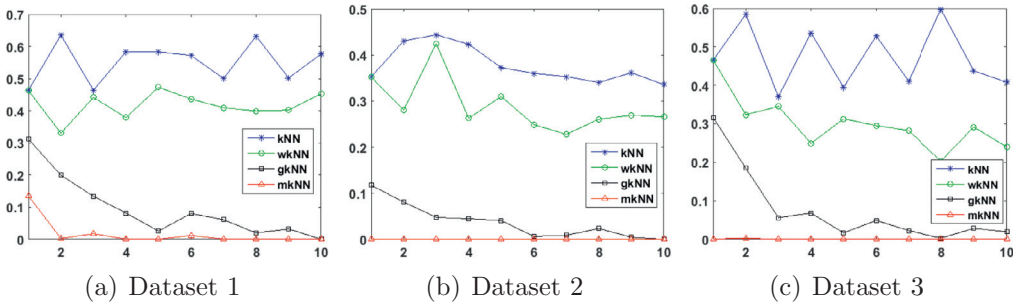


Fig. 8. Experimental results on synthetic data sets. Mean error rate on the ordinate and *k* on the abscissa.

6.2. Experimental results on real-world data sets

We also conduct experiments on six real-world data sets from the UCI data repository, which contains real application data collected in various fields and is widely used to test the performance of different machine learning algorithms.[4]. The information on these six data sets is listed in Table 5 (*n*: number of samples; *d*: feature dimension; *C*: class number).

On each data set, we let *k* change from 1 to 10. For each *k*, we run each algorithm 10 times, with a training set containing *k* labeled samples randomly selected from each class and a testing set containing all the rest samples. The final error rate is the mean of the 10 error rates and is shown in Fig. 9.

From Fig. 9, we can see that the proposed m*k*NN outperforms the other algorithms in terms of both the accuracy and stability of performance. The error curves of m*k*NN decrease much more quickly than that the other algorithms, which indicates that m*k*NN is able to utilize fewer labeled samples to achieve better accuracy. This is very important in applications where there is a very small amount of labeled samples available, because labeled samples are usually more expensive to

[4] UC Irvine Machine Learning Repository: http://archive.ics.uci.edu/ml/.

Table 5
Information on the experimental data sets.

	usps	segmentation	banknote	pendigits	multifeature	statlog
n	9298	2086	1348	10992	2000	6435
d	256	19	4	16	649	36
C	10	6	2	10	10	6

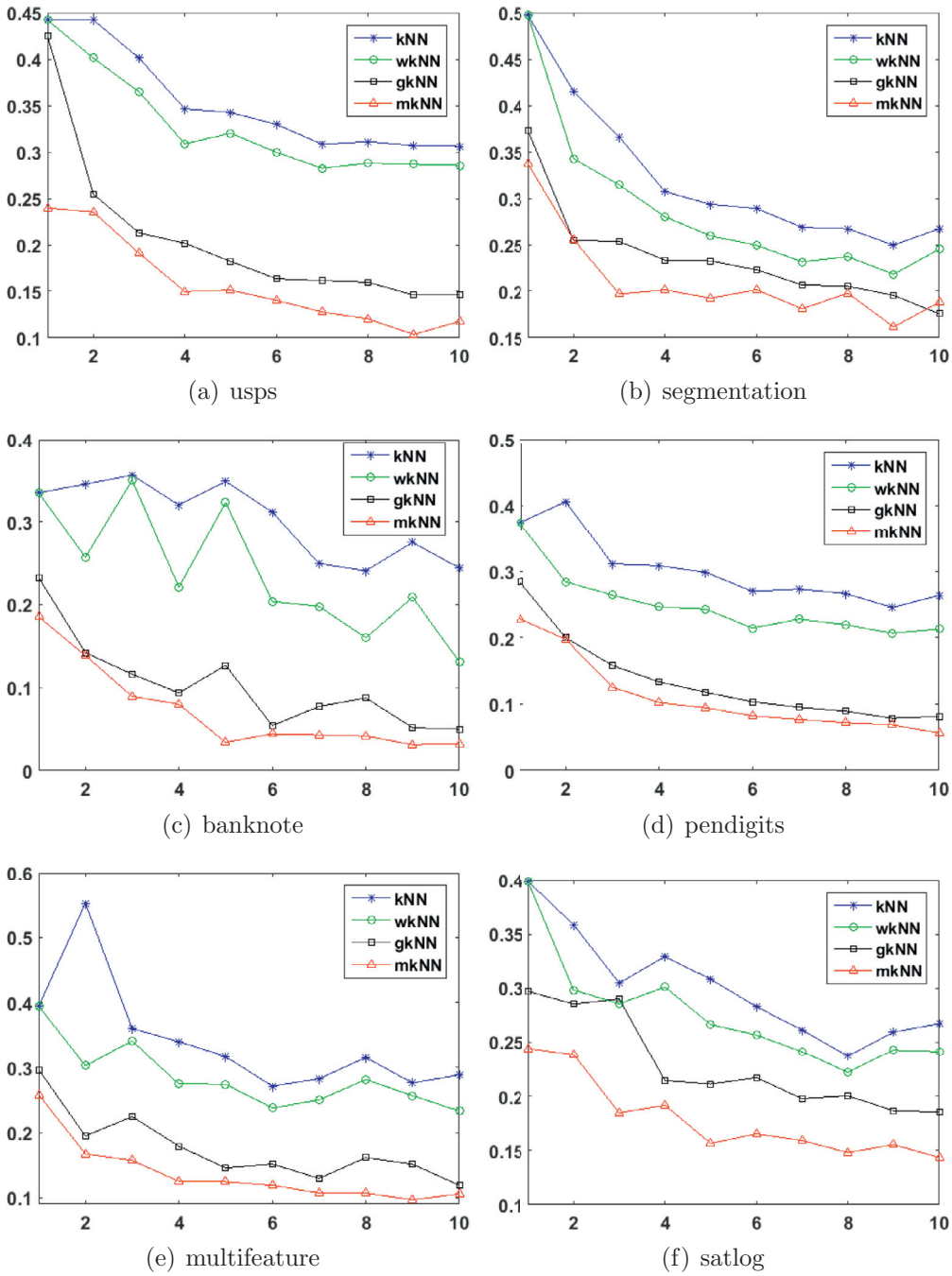


Fig. 9. Experimental results on six real-world data sets. Mean error rate on the ordinate and k on the abscissa.

Table 6
Experimental results on real-world data sets (error rate: %).

	SVM	ANN	NB	DT	m*k*NN
usps	77.35 ± 4.6	40.96 ± 4.3	34.73 ± 4.9	73.96 ± 8.4	**19.37 ± 3.9**
segmentation	66.54 ± 8.7	42.54 ± 3.8	50.83 ± 4.3	49.55 ± 2.6	**24.18 ± 4.5**
banknote	70.01 ± 14.7	23.03 ± 6.2	28.37 ± 7.8	44.43 ± 0.4	**9.73 ± 6.9**
pendigits	49.41 ± 12.3	44.02 ± 27.4	42.36 ± 6.2	80.83 ± 11.3	**12.51 ± 2.3**
multifeature	82.89 ± 6.4	50.53 ± 24.2	26.89 ± 2.9	76.14 ± 2.9	**15.36 ± 2.3**
satlog	57.26 ± 17.1	30.15 ± 4.9	28.81 ± 5.9	72.55 ± 13.5	**21.07 ± 8.0**

obtain, i.e. they need to be annotated by human with expert knowledge and/or special equipment and take a lot of time. Therefore, it is of great practical value that a classifier is capable of accurately classifying abundant unlabeled samples, given only a small number of labeled samples.

From Fig. 9, we can also conclude that: (1) by introducing a weight scheme in the traditional *k*NN algorithm, weighted *k*NN (w*k*NN) can generally outperform traditional *k*NN; (2) the performance of geodesic *k*NN (g*k*NN) has a relative large improvement to both *k*NN and weighted *k*NN for most cases, because geodesic distance is a valid measure on manifold; (3) the manifold *k*NN (m*k*NN) almost always achieves the smallest error rate, because the constrained TRW is a more effective and robust measure of the global geometrical structure on manifold and, meanwhile, m*k*NN can take both the class information and constraint information into account.

6.3. Experimental results of the comparison with other traditional supervised classifiers

We conduct experiments to compare the performance of the proposed manifold *k*NN algorithm with other popular supervised classifiers on the six real-world data sets. The baseline algorithms are: Support Vector Machine (SVM),[5] Artificial Neural Networks (ANN), Naive Bayes (NB) and Decision Tree (DT). The configurations of the baseline algorithms are: radial basis kernel for SVM; three-layer networks trained with back-propagation for ANN; kernel smoothing density estimation for NB; binary classification tree and merging-pruning strategy according to validation error for DT. We adopt a grid-search strategy to tune the parameters of each algorithm and the parameters are set to produce the lowest 2-fold cross validation error rate.

To examine the capability of classifying plenty of unlabeled samples with only a few labeled samples, we randomly choose three labeled samples from each class to form the training set and the remaining samples are treated as unlabeled samples to form the testing set. Each algorithm runs 10 times and the final result is the average of these 10 error rates. The experimental results (mean error ± standard deviation) are shown in Table 6.

From these results, we can see that m*k*NN significantly outperform these traditional supervised classifiers, given only three labeled samples per class. Similar to traditional *k*NN, traditional supervised classifiers are incapable of exploiting the manifold structure information of the underlying data distribution, thus their accuracies are very low while the labeled sample number is very small. Furthermore, when the labeled samples are randomly selected, their positions vary greatly in data space. Sometime they are not uniformly distributed and thus cannot well cover the whole data distribution. As a result, the performance of traditional classifiers also varies greatly. In contrast, because m*k*NN adopts tired random walk to measure manifold similarity which reflects the global geometrical information of the underlying manifold structure and is robust to local changes [33], it can achieve much better results.

To further investigate the performance of these algorithms under the condition of providing different number of training samples, we carry out experiments with different training set sizes. For each data set, we let k varies from 1 to 40 ($k = 1, 3, \ldots, 39$) and randomly choose k labeled samples from each class to form the training set. The rest of the data set are treated as unlabeled samples to test each algorithm's performance. For each k, every algorithm runs 10 times on each data set. The mean error rate and standard deviation of the 10-run results are shown in Figs. 10 and 11, respectively.

From Fig. 10 we can see that while the labeled sample number is small, the error rates of the traditional algorithms are very large (as also indicated in Table 6 which contains the results of the three labeled samples per class). As the number of labeled sample increases, the error rates decrease. This trend becomes slower after 15 labeled samples per class. The proposed m*k*NN achieves the lowest error rate for both situations of small and large number of labeled samples. The improvement is especially obvious and significant when the labeled samples number is small. From Fig. 11, we can also see that the performance of m*k*NN is also relatively steady for most cases. One should note that on the left of each plot in Fig. 11, the small standard deviation values of decision tree (DT) and SVM are due to the fact that their error rates are always very high in each run (e.g. the error rates of SVM on data set *usps* concentrate closely around 95).

[5] For SVM we use the libsvm toolbox: https://www.csie.ntu.edu.tw/~cjlin/libsvm/. Other algorithms we use MATAB toolbox.

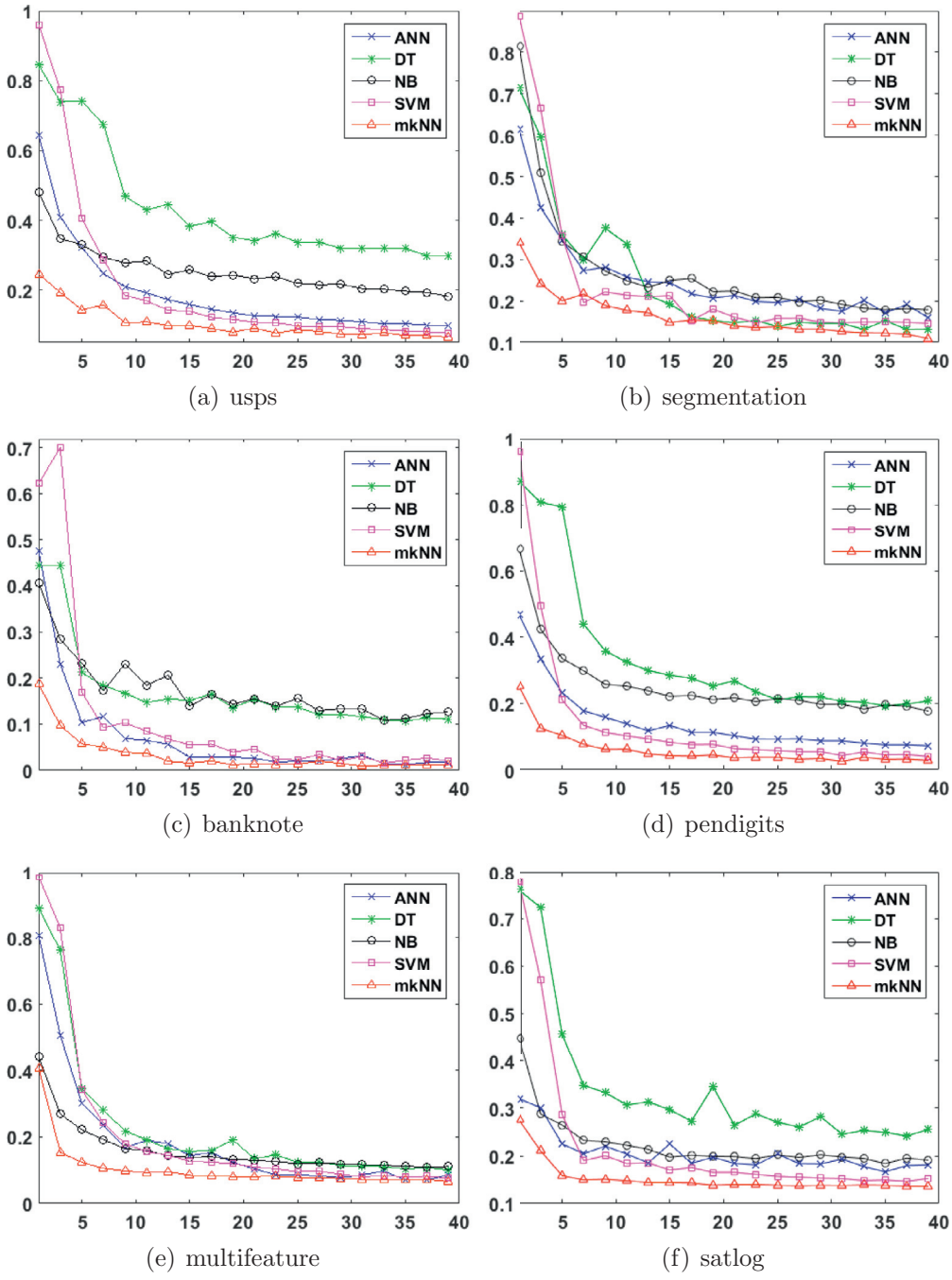


Fig. 10. Experimental results on six real-world data sets. Mean error rate on the ordinate and labeled samples number per class on the abscissa.

6.4. Experimental results of time complexity

To examine the effectiveness and efficiency of the proposed sequential learning strategy in Section 5, we conduct experiments on three real-world data sets *banknote*,*satlog* and *pendigits* to show the difference of m*k*NN's performance with and without sequential learning algorithm. Each data set is divided into three subsets: training set, validation set and online set. The training set is fixed to contain 10 labeled samples per class. We conduct 10 experiments, with the online set size changes from 100 to 1000 with step size 100. The validation set contains the rest of the samples. First, m*k*NN runs on training set and validation set to learn the class distribution. Thereafter, at each time a sample is drawn from the online set as a new coming sample. For sequential m*k*NN, the previous learning result is used to classify the new sample according to

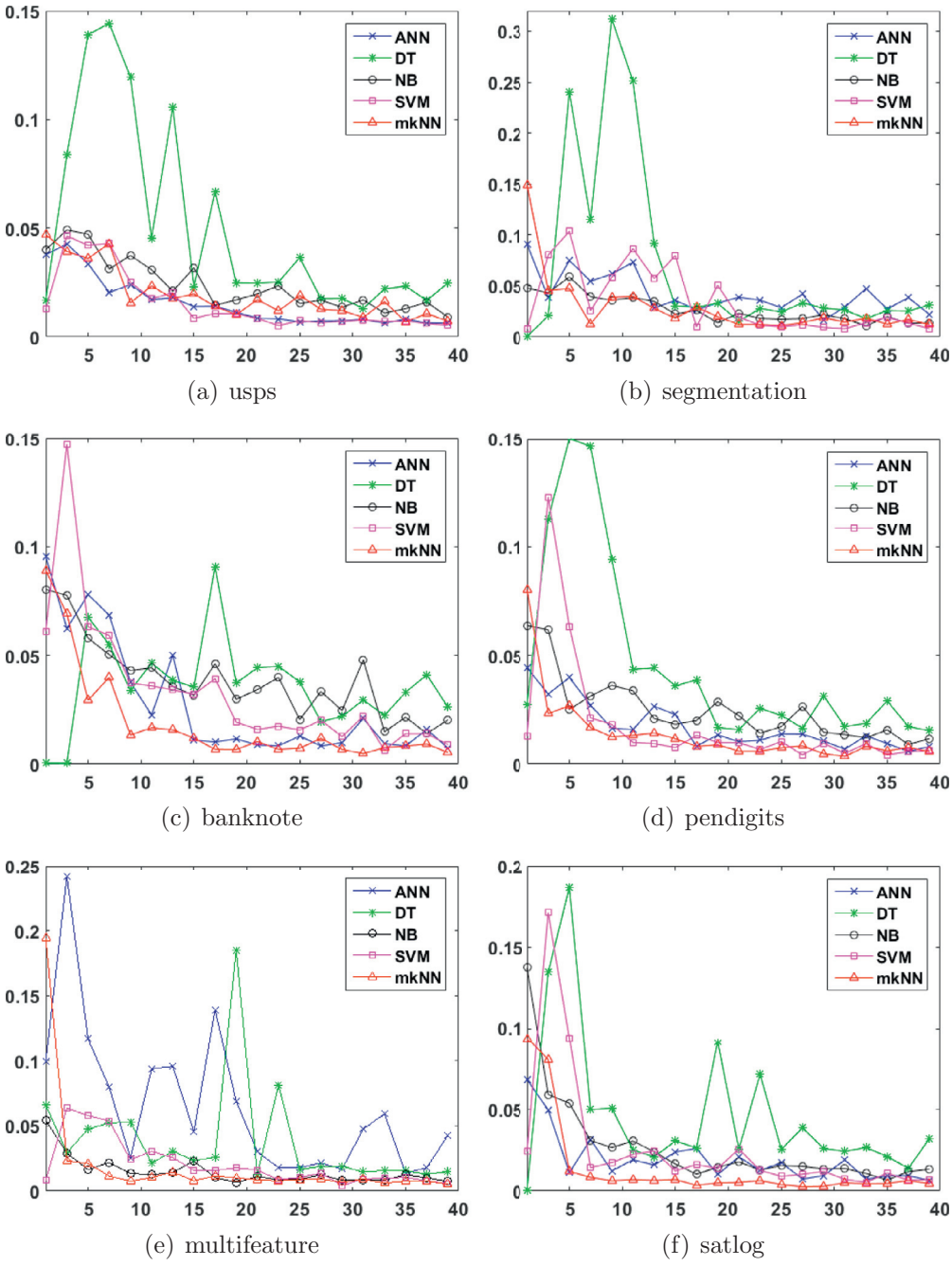
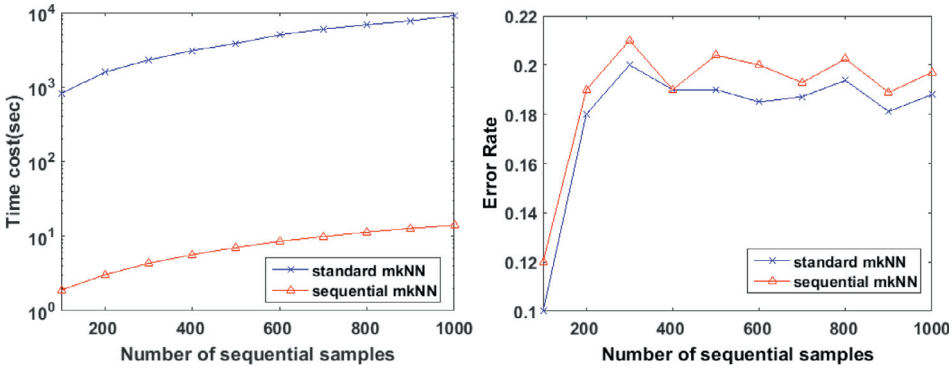


Fig. 11. Experimental results on six real-world data sets. Standard deviation of error rates on the ordinate and labeled samples number per class on the abscissa.

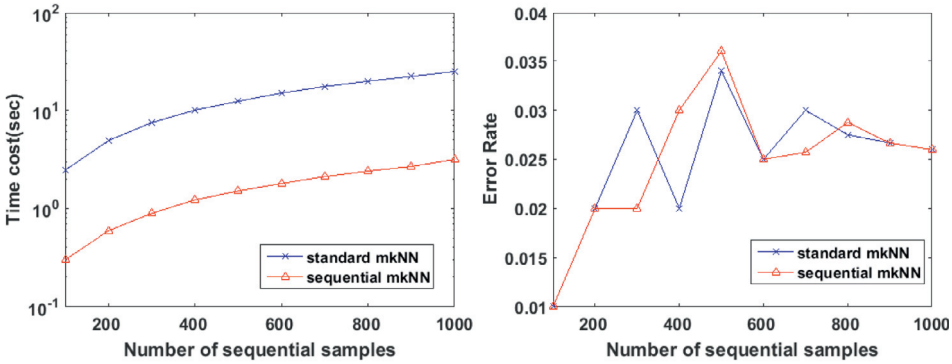
Eq. (7) and (8) For standard m*k*NN, the new sample is added to the validation set and the whole learning process is repeated to classify the new sample. The experimental results[6] are shown in Fig. 12.

From these results, we can see that while the classification accuracy of standard m*k*NN and sequential m*k*NN are comparable, the time cost of sequential m*k*NN reduces dramatically for online classification (e.g., for *satlog* data set, to classify 1000 sequentially coming samples, standard m*k*NN takes about 9100 sec but sequential m*k*NN uses only about 14 sec to achieve a similar result). Therefore, the sequential algorithm has great merit in solving the online classification problem and can be potentially applied to a wide range of transductive learning algorithms to make them inductive.

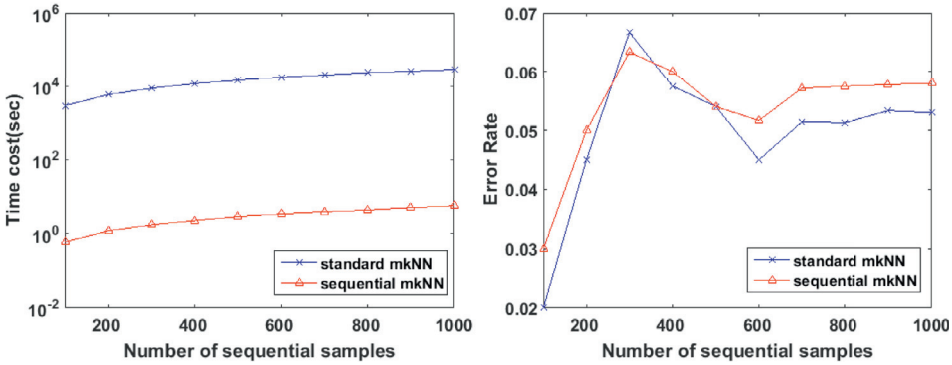
[6] The configuration of our computer: 16GB RAM, double-core 3.7 GHz Intel Xeon CPU and MATLAB 2015 academic version.



(a) Satlog data set



(b) Banknote data set



(c) Pendigits data set

Fig. 12. Experimental results of the comparison between standard m*k*NN and sequential m*k*NN to classify online samples real-world data sets.

We also conduct experiments to compare the time complexity of all the baseline algorithms with the proposed m*k*NN on these three data sets. Each data set is split into *training* and *testing* parts three times independently, with splitting ratios 10%, 25% and 50%, respectively. For each split, every algorithm runs 10 times and the mean time cost is recorded. Table 7 shows the experimental results (the second column shows the splitting ratio).

From this table we can see that the algorithms fall into two categories according to their time costs: one category contains SVM, DT, ANN, *k*NN and w*k*NN, whose time costs are closely related with the training data set size. Another category includes NB, g*k*NN and m*k*NN, whose time costs depend more upon the overall data set size. SVM, *k*NN and w*k*NN are generally faster than others. Although m*k*NN is the second slowest one, it is still much faster than g*k*NN and its overall time does not prolong as the training set size increases. It should be mentioned that although the classification accuracy of m*k*NN is much better than *k*NN and other traditional supervised classifiers, the computational complexity of m*k*NN is also higher ($O(n^3)$) than traditional *k*NN ($O(n)$). Directly computing P_{TRW} matrix is time cost, since it is not symmetrical, positive definite matrix and thus a LU decomposition has to be used. One way to speed-up matrix inverse is to convert P_{TRW}

Table 7
Comparison of overall time for training and testing on three data sets (sec).

data	(%)	SVM	DT	ANN	NB	*k*NN	w*k*NN	g*k*NN	m*k*NN
banknote	10	0.01	0.14	0.23	0.09	0.03	0.02	1.23	0.19
	25	0.01	0.19	0.24	0.09	0.04	0.03	1.23	0.20
	50	0.02	0.29	0.27	0.08	0.05	0.05	1.26	0.20
satlog	10	0.47	0.43	0.59	7.14	0.13	0.14	31.99	6.01
	25	1.78	1.16	1.58	7.56	0.24	0.29	32.55	6.08
	50	5.12	3.11	4.01	7.61	0.31	0.40	32.82	6.30
pendigits	10	0.44	0.78	0.70	8.70	0.22	0.28	149.14	25.27
	25	0.86	2.28	2.65	8.46	0.41	0.59	146.89	26.11
	50	1.55	6.09	5.06	6.93	0.54	0.80	145.71	25.35

to a symmetrical, positive definite matrix and then adopt the Cholesky decomposition, whose computational cost is just a half of the LU decomposition [13]. Noting that $P_{TRW} = \left(I - \alpha D^{-1}W\right)^{-1} = D^{1/2}\left(I - \alpha D^{-1/2}WD^{-1/2}\right)^{-1}D^{-1/2} = D^{1/2}R^{-1}D^{-1/2}$, where $R = I - \alpha D^{-1/2}WD^{-1/2}$ is a symmetrical, positive definite matrix[7], we can first inverse matrix R using Cholesky decomposition and then compute P_{TRW} with very small computational cost. Our following work will be focused on the further reduction of computational complexity.

7. Conclusions

In this paper we proposed a new k nearest-neighbor algorithm, mkNN, to classify nonlinear manifold distributed data as well as traditional Gaussian distributed data, given a very small amount of labeled samples. We also presented an algorithm to attack the problem of high computational cost for classifying online data with mkNN and other transductive algorithms. The superiority of the mkNN has been demonstrated by substantial experiments on both synthetic data sets and real-world data sets. Given the widespread appearance of manifold structures in real-world problems and the popularity of the traditional kNN algorithm, the proposed manifold version kNN shows promising potential for classifying manifold-distributed data.

Acknowledgments

The authors wish to thank the anonymous reviewers for reading the entire manuscript and offering many useful suggestions. This research is partly supported by NSFC, China (No: 61572315) and 973 Plan China (No. 2015CB856004). The research is also partly supported by YangFan Project (Grant No. 14YF1411000) of Shanghai Municipal Science and Technology Commission, the Innovation Program (Grant No. 14YZ131) and the Excellent Youth Scholars (Grant No. sdl15101) of Shanghai Municipal Education Commission, the Science Research Foundation of Shanghai University of Electric Power (Grant No. K2014-032).

References

[1] Y. Bengio, H. Larochelle, P. Vincent, Non-local manifold parzen windows, in: Advances in Neural Information Processing Systems, 2005, pp. 115–122.
[2] K. Beyer, J. Goldstein, R. Ramakrishnan, U. Shaft, When is nearest neighbor meaningful? in: International Conference on Database Theory, Springer, 1999, pp. 217–235.
[3] M.H. Bhuyan, D. Bhattacharyya, J. Kalita, A multi-step outlier-based anomaly detection approach to network-wide traffic, Inf. Sci. 348 (2016) 243–271.
[4] W.M. Boothby, An Introduction to Differentiable Manifolds and Riemannian Geometry, vol. 120, Gulf Professional Publishing, 2003.
[5] F. Camastra, A. Staiano, Intrinsic dimension estimation: Advances and open problems, Inf. Sci. 328 (2016) 26–41.
[6] Y. Chen, J. Zhang, D. Cai, W. Liu, X. He, Nonnegative local coordinate factorization for image representation, Image Process. IEEE Trans. 22 (3) (2013) 969–979.
[7] R. Collobert, F. Sinz, J. Weston, L. Bottou, Large scale transductive svms, J. Machine Learn. Res. 7 (2006) 1687–1712.
[8] T.M. Cover, P.E. Hart, Nearest neighbor pattern classification, Inf. Theory IEEE Trans. 13 (1) (1967) 21–27.
[9] J. Derrac, F. Chiclana, S. Garcia, F. Herrera, Evolutionary fuzzy k-nearest neighbors algorithm using interval-valued fuzzy sets, Inf. Sci. 329 (2016) 144–163.
[10] Z. Fu, Z. Lu, H.H. Ip, H. Lu, Y. Wang, Local similarity learning for pairwise constraint propagation, Multimedia Tools Appl. 74 (11) (2015) 3739–3758.
[11] Y. Gao, Q. Liu, X. Miao, J. Yang, Reverse k-nearest neighbor search in the presence of obstacles, Inf. Sci. 330 (2016) 274–292.
[12] A. Goldberg, B. Recht, J. Xu, R. Nowak, X. Zhu, Transduction with matrix completion: three birds with one stone, in: Advances in Neural Information Processing Systems, 2010, pp. 757–765.
[13] G.H. Golub, C.F. Van Loan, Matrix Computations, vol.3, Johns Hopkins University Press, 2012.
[14] C. Gong, K. Fu, Q. Wu, E. Tu, J. Yang, Semi-supervised classification with pairwise constraints, Neurocomputing 139 (2014) 130–137.
[15] C. Gong, T. Liu, D. Tao, K. Fu, E. Tu, J. Yang, Deformed graph laplacian for semisupervised learning, IEEE Transa. Neural Networks Learn. Syst. 26 (10) (2015) 2261–2274.
[16] E.-H. S. Han, G. Karypis, V. Kumar, Text Categorization Using Weight Adjusted k-nearest neighbor Classification, Springer, 2001.

[7] R is apparently symmetrical because W is symmetrical (Let $W = (W + W^T)/2$ if W is not symmetrical). We prove its positive definiteness. Because the spectral radius of P_{TRW} is in (0, 1) and P_{TRW} is similar to R^{-1}, so spectral radius of R^{-1} is in (0, 1). So all eigenvalues of R are positive. Therefore R is a positive definite, symmetrical matrix.

[17] T. Hastie, R. Tibshirani, Discriminant adaptive nearest neighbor classification, Pattern Anal. Machine Intell. IEEE Trans. 18 (6) (1996) 607–616.
[18] K. Hechenbichler, K. Schliep, Weighted k-nearest-neighbor techniques and ordinal classification, Collaborative Research Center 386, Discussion Paper 399 (2004) 1–16.
[19] J. Jiang, C. Chen, K. Huang, Z. Cai, R. Hu, Noise robust position-patch based face super-resolution via tikhonov regularized neighbor representation, Inf. Sci. 367–368 (2016) 354–372.
[20] T. Joachims, Transductive learning via spectral graph partitioning, in: Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 290–297.
[21] M. Kolahdouzan, C. Shahabi, Voronoi-based k nearest neighbor search for spatial network databases, in: Proceedings of the Thirtieth International Conference on Very Large Data Bases-Volume 30, VLDB Endowment, 2004, pp. 840–851.
[22] W. Liu, J. He, S.-F. Chang, Large graph construction for scalable semi-supervised learning, in: Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 679–686.
[23] U.V. Luxburg, A. Radl, M. Hein, Getting lost in space: large sample analysis of the resistance distance, in: Advances in Neural Information Processing Systems, 2010, pp. 2622–2630.
[24] L. Ma, M.M. Crawford, J. Tian, Local manifold learning-based-nearest-neighbor for hyperspectral image classification, Geosci. Remote Sensing IEEE Trans. 48 (11) (2010) 4099–4109.
[25] E. Nowakowska, J. Koronacki, S. Lipovetsky, Dimensionality reduction for data of unknown cluster structure, Inf. Sci. 330 (2016) 74–87.
[26] A.G. Percus, O.C. Martin, Scaling universalities ofkth-nearest neighbor distances on closed manifolds, Adv. Appl. Math. 21 (3) (1998) 424–436.
[27] A. Petraglia, et al., Dimensional reduction in constrained global optimization on smooth manifolds, Inf. Sci. 299 (2015) 243–261.
[28] T. Reitmaier, A. Calma, B. Sick, Transductive active learning–a new semi-supervised learning approach based on iteratively refined generative models to capture structure in data, Inf. Sci. 293 (2015) 275–298.
[29] S.T. Roweis, L.K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (5500) (2000) 2323–2326.
[30] S. Tan, Neighbor-weighted k-nearest neighbor for unbalanced text corpus, Expert Syst. Appl. 28 (4) (2005) 667–671.
[31] D. Tao, J. Cheng, X. Lin, J. Yu, Local structure preserving discriminative projections for rgb-d sensor-based scene classification, Inf. Sci. 320 (2015) 383–394.
[32] J.B. Tenenbaum, V. De Silva, J.C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (5500) (2000) 2319–2323.
[33] E. Tu, L. Cao, J. Yang, N. Kasabov, A novel graph-based k-means for nonlinear manifold clustering and representative selection, Neurocomputing 143 (2014) 109–122.
[34] E. Tu, N. Kasabov, J. Yang, Mapping temporal variables into the neucube for improved pattern recognition, predictive modeling, and understanding of stream data, IEEE Trans. Neural Networks Learn. Syst. (99) (2016) 1–13, doi:10.1109/TNNLS.2016.2536742.
[35] E. Tu, J. Yang, J. Fang, Z. Jia, N. Kasabov, An experimental comparison of semi-supervised learning algorithms for multispectral image classification, Photogrammetric Eng. Remote Sensing 79 (4) (2013) 347–357.
[36] E. Tu, J. Yang, N. Kasabov, Y. Zhang, Posterior distribution learning (pdl): a novel supervised learning framework using unlabeled samples to improve classification performance, Neurocomputing 157 (2015) 173–186.
[37] P. Turaga, R. Chellappa, Nearest-neighbor search algorithms on non-euclidean manifolds for computer vision applications, in: Proceedings of the Seventh Indian Conference on Computer Vision, Graphics and Image Processing, ACM, 2010, pp. 282–289.
[38] R. Vemulapalli, J.K. Pillai, R. Chellappa, Kernel learning for extrinsic classification of manifold features, in: Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE, 2013, pp. 1782–1789.
[39] H. Wang, J. Wu, S. Yuan, J. Chen, On characterizing scale effect of chinese mutual funds via text mining, Signal Process. 124 (2016) 266–278.
[40] J. Wang, T. Jebara, S.-F. Chang, Semi-supervised learning using greedy max-cut, J. Machine Learn. Res. 14 (1) (2013) 771–800.
[41] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, J. Machine Learn. Res. 10 (2009) 207–244.
[42] X. Wu, V. Kumar, J.R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G.J. McLachlan, A. Ng, B. Liu, S.Y. Philip, et al., Top 10 algorithms in data mining, Knowl. Inf. Syst. 14 (1) (2008) 1–37.
[43] J. Xie, H. Gao, W. Xie, X. Liu, P.W. Grant, Robust clustering by detecting density peaks and assigning points based on fuzzy weighted k-nearest neighbors, Inf. Sci. 354 (2016) 19–40.
[44] H. Yin, S.M. Zaki, A self-organising multi-manifold learning algorithm, in: Bioinspired Computation in Artificial Systems, Springer, 2015, pp. 389–398.
[45] J. Yu, S.B. Kim, Density-based geodesic distance for identifying the noisy and nonlinear clusters, Inf. Sci. 360 (2016) 231–243.
[46] J. Yu, D. Tao, J. Li, J. Cheng, Semantic preserving distance metric learning and applications, Inf. Sci. 281 (2014) 674–686.
[47] J. Zhu, Q. Xie, K. Zheng, An improved early detection method of type-2 diabetes mellitus using multiple classifier system, Inf. Sci. 292 (2015) 1–14.
[48] X. Zhu, Z. Ghahramani, J.D. Lafferty, Semi-supervised learning using gaussian fields and harmonic functions, in: Proceedings of the 20th International Conference on Machine Learning (ICML-03), 2003, pp. 912–919.