Kernel-based Transductive Learning with Nearest Neighbors

Liangcai Shu, Jinhui Wu, Lei Yu, and Weiyi Meng

Dept. of Computer Science, SUNY at Binghamton
Binghamton, New York 13902, U. S. A.
{lshu,jwu6,lyu,meng}@cs.binghamton.edu

Abstract. In the k-nearest neighbor (KNN) classifier, nearest neighbors involve only labeled data. That makes it inappropriate for the data set that includes very few labeled data. In this paper, we aim to solve the classification problem by applying transduction to the KNN algorithm. We consider two groups of nearest neighbors for each data point — one from labeled data, and the other from unlabeled data. A kernel function is used to assign weights to neighbors. We derive the recurrence relation of neighboring data points, and then present two solutions to the classification problem. One solution is to solve it by matrix computation for small or medium-size data sets. The other is an iterative algorithm for large data sets, and in the iterative process an energy function is minimized. Experiments show that our solutions achieve high performance and our iterative algorithm converges quickly.

Key words: KNN, transductive learning, semi-supervised learning, kernel function

1 Introduction

The k-nearest neighbor (KNN) algorithm [1] is a simple and effective supervised learning algorithm. One of the disadvantages of supervised learning is that it requires significant amount of labeled data for training in order to achieve high performance. But in applications like object identification, text categorization, a large amount of labeled data need a lot of human efforts, whereas unlabeled data is quite cheap and easy to obtain. This is the reason that transductive learning or semi-supervised learning has been developed in recent years [2][3][4][5][6][7][8][9][10].

Transduction or transductive learning was first introduced by Vladimir Vapnik who thinks it is preferable to induction because induction requires solving a more general problem before solving a specific problem, e.g. decision trees, SVMs, and transduction solves the specific problem directly. This is so-called Vapnik's principle [11]. Transductive learning takes advantage of unlabeled data to capture the global structure of the data that is assumed to be helpful to predict class labels. But this approach requires assumptions on the structure of data — first, cluster assumption that data in the same cluster are supposed to be in

the same class, and second, manifold assumption that the high-dimensional data lie on a low-dimensional manifold [11]. The second assumption is for algorithms using pairwise distance to avoid the curse of dimensionality.

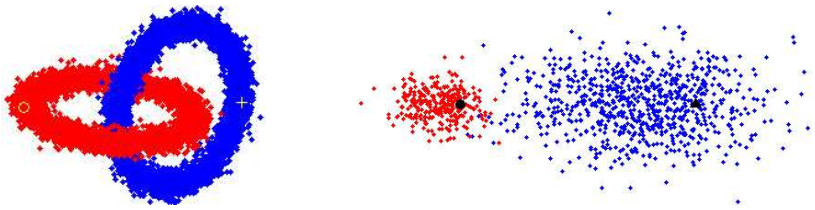

Fig. 1. (left) The two interlock rings problem in 3D space. Only two points, one for each class and marked by o and +, are labeled. **(right)** An example in which simple local propagation fails. • and ▲ represent labeled points.

Consider the two interlock rings problem in Fig. 1 (left). We aim to find the class label for each point, given only two labeled points. Mixture models cannot solve it since distribution of points in a cluster is unknown instead of standard distributions like Gaussian. Radial basis functions (RBFs) and KNN cannot solve it because of too few labeled points. The difficulty of applying support vector machines (SVMs) on such data is that it is hard to find the right kernel function to map the data from the original space to a higher dimensional space where the data becomes linearly separable. Also, kernel learning is time-consuming. So we do not consider SVMs in this paper.

Instead, we propose an approach called TKNN, which applies transduction to the KNN algorithm to solve this problem. Traditionally, the nearest neighbors in the KNN algorithm are defined on the labeled data only. In this paper, we re-examine the method and extend KNN by redefining nearest neighbors as from both labeled data and unlabeled data. Two groups of nearest neighbors are simultaneously considered for each data point — one group from labeled data, and the other from unlabeled data. A kernel function or radial basis function is used to increase weights for nearer neighbors. Then global structure of data is captured in a graph. Taking advantage of global structure, labels propagate through the graph, and the two interlock rings problem can be solved completely.

Note that TKNN is more sophisticated than simple local propagation staring from the labeled points. In Fig. 1 (right), simple local propagation fails because, without global structure of data, the label propagates across the sparse area between two classes. Otherwise in TKNN, due to the kernel function, label propagation in sparse areas gives way to that in dense areas and finally the sparsest area is identified as the boundary.

Our Contributions are as follows:

– We apply transductive learning to KNN, with the kernel function adjusting weights for nearest neighbors.

- We derive the recurrence relation between labeled data and unlabeled data, and propose solutions to the classification problem.
- We propose an efficient and flexible iterative algorithm for large data sets. Techniques are applied to achieve fast convergence. And an outlier detection in labeled data is presented too.
- Experimental results show the performance and efficiency of our algorithm.

The remaining part of this paper is organized as follows. In Section 2 we discuss the model formulation, including deriving recurrence relation between labeled data and unlabeled data. In Section 3 we propose solutions and derive the iterative algorithm and its ability to converge, and also discuss outlier detection. In Section 4 we report our experimental results and analysis. Section 5 is related work. Section 6 is the conclusion.

2 Model formulation

2.1 Problem description and notation

In this paper, we aim to solve the problem of classification by applying transductive learning to the KNN algorithm, i.e., taking advantage of unlabeled data as well as labeled data.

Assume we are given the set of points $\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_U$ in a p-dimensional Euclidean space $\mathbf{R}^p$, where $\mathcal{D}_L = \{\mathbf{x}_i\}_{1 \le i \le l}$ is the set of labeled points and $\mathcal{D}_U = \{\mathbf{x}_i\}_{l+1 \le i \le l+u}$ is the set of unlabeled points. One class label should be assigned to each member of $\mathcal{D}$. The set of classes is denoted by $\mathcal{C} = \{c_r\}_{1 \le r \le |C|}$ and correspondingly, the set of class labels is denoted by $C = \{r\}_{1 \le r \le |C|}$. Here, $|C|$ is the cardinality of C and r is an integer.

The class label of point $\mathbf{x}_i \in \mathcal{D}$ is denoted by $y(\mathbf{x}_i)$ or y_i, where $y_i \in C$. Each point in $\mathcal{D}_L$ has been labeled or assigned to one class in C, while the class label of each point in $\mathcal{D}_U$ is unknown and needs to be determined. In this paper, k_l denotes the number of nearest neighbors in labeled data, and k_u denotes the number of nearest neighbors in unlabeled data. The k_l nearest neighbors of a given point $\mathbf{x}_i \in \mathcal{D}$ are denoted by $\{\mathbf{n}_{iq}\}_{1 \le q \le k_l}$, where $\mathbf{n}_{iq}$ is $\mathbf{x}_i$'s qth nearest neighbor in $\mathcal{D}_L$. Similarly, the k_u nearest neighbors of $\mathbf{x}_i$ are denoted by $\{\mathbf{m}_{iq}\}_{1 \le q \le k_u}$, where $\mathbf{m}_{iq}$ is $\mathbf{x}_i$'s qth nearest neighbor in $\mathcal{D}_U$.

2.2 Extending KNN

In supervised KNN learning [12][1], the nearest neighbors included only labeled points. Therefore, it needs a lot of labeled points for good performance. In this paper, we propose TKNN after extending KNN by including both labeled and unlabeled points in the nearest neighbors of a point.

After applying the KNN to density estimation [1], for a point $\mathbf{x}$ the posterior probability $p(c_r|\mathbf{x})$ can be estimated as

$$\hat{p}(c_r|\mathbf{x}) = \frac{k(r)}{k}, \tag{1}$$

where $k(r)$ is the number of points with class label r among the k nearest neighbors of $\mathbf{x}$. Obviously, $\sum_{r=1}^{|C|} k(r) = k$.

In supervised KNN learning [12][1], $k(r)$ in Eq. (1) is computed based on labeled points and $\hat{p}(c_r|\mathbf{x})$ can be directly obtained. Differently, in our model TKNN, there are two groups of nearest neighbors for each point. One group includes k_l nearest neighbors from labeled data, and the other includes k_u nearest neighbors from unlabeled points. With the help of unlabeled data, the amount of labeled data can be small.

Let $k_l(r)$ be the number of labeled points in class c_r, and $k_u(r)$ be the number of unlabeled points in class c_r. $k_l(r)$ can be easily obtained, but $k_u(r)$ is not available. Then $\hat{p}(c_r|\mathbf{x})$ in Eq. (1) cannot be directly obtained anymore. But we can learn $\hat{p}(c_r|\mathbf{x})$ by deriving recurrence relation among neighboring data points. For convenience of description, we define the class matrix.

Class matrix The class matrix for a data set describes probabilities that data are assigned to classes, data as rows and classes as columns. Then the sum of values in each row is 1.

Definition 1. *The* class matrix *of data set* $\mathcal{D} = \{\mathbf{x}_i\}_{1 \le i \le l+u}$ *is defined as* $\mathbf{P} = (p_{ij})_{(l+u)\times|C|}$, *where* $p_{ij} = p(c_j|\mathbf{x}_i)$.

The class matrix $\mathbf{P}$ can be written in two blocks:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_L \\ \mathbf{P}_U \end{bmatrix}, \tag{2}$$

where $\mathbf{P}_L$'s rows correspond to labeled data and $\mathbf{P}_U$'s rows correspond to unlabeled data.

$\mathbf{P}_L$ is defined according to prior knowledge of labeled data. For any point $\mathbf{x}_i \in \mathcal{D}_L$, assume its class label is $y_i \in C$. Then the element at the ith row and y_ith column is set to 1. Formally, $\mathbf{P}_L = (p_{ij})_{l\times|C|}$, where

$$p_{ij} = p(c_j|\mathbf{x}_i) = \begin{cases} 1, & j = y_i\,, \\ 0, & j \neq y_i\,. \end{cases}$$

2.3 Kernel-based weights for neighbors

Every neighbor of $\mathbf{x}_i$ has a weight according to its distance to $\mathbf{x}_i$. Nearer neighbors are given greater weights. We use a kernel function, also called radial basis function, to compute weights.

Kernel function In this paper, we use Gaussian function as kernel

$$K(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{\sqrt{2\pi}h} exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2h^2}\right), \tag{3}$$

where $\|\cdot\|$ is Euclidean distance and h is the bandwidth.

The bandwidth h in Eq. (3) is also called smooth parameter. Its selection impacts the performance of the kernel-based algorithm.

Weight matrix Based on the kernel function, we define the weight matrix. The weight matrix of a data set describes impacts of neighbors to each data point. Its rows and columns are all the data points. For each row, only columns corresponding to two groups of nearest neighbors are given weights, and other column are given 0. Then the matrix is not necessarily symmetric. Based on the weight matrix, a KNN graph is constructed.

Definition 2. *Let λ be a real number in $[0,1]$. The* weight matrix *of data set $\mathcal{D} = \{\mathbf{x}_i\}_{1\le i\le l+u}$ is defined as $\mathbf{W} = (w_{ij})_{(l+u)\times(l+u)}$, where*

$$w_{ij} = \begin{cases} K(\mathbf{x}_i, \mathbf{x}_j), & 1 \le j \le l \ \text{ and } \ \mathbf{x}_j \in \{\mathbf{n}_{iq}\}_{1\le q\le k_l}, \\ \lambda\, K(\mathbf{x}_i, \mathbf{x}_j), & l+1 \le j \le l+u \ \text{ and } \ \mathbf{x}_j \in \{\mathbf{m}_{iq}\}_{1\le q\le k_u}, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

where $\{\mathbf{n}_{iq}\}_{1\le q\le k_l}$ and $\{\mathbf{m}_{iq}\}_{1\le q\le k_u}$ are $\mathbf{x}_i$'s neighbors from labeled data and unlabeled data.

The parameter λ, called influence factor of unlabeled data, is used to adjust unlabeled data's influence in the graph. If $\lambda = 0$, unlabeled data have no influence. If $\lambda = 1$, unlabeled data have the same influence as labeled data.

We row-normalize $\mathbf{W}$, i.e., adding row elements together and dividing each element by the corresponding row totals. Then we get row-normalized matrix $\mathbf{V}$. Obviously, the sum of each row's elements in $\mathbf{V}$ equals 1. We call $\mathbf{V}$ the *normalized weight matrix*. Considering $\mathcal{D} = \mathcal{D}_L \cup \mathcal{D}_U$, $\mathbf{V}$ consists of four blocks:

$$\mathbf{V} = (v_{ij})_{(l+u)\times(l+u)} = \begin{bmatrix} \mathbf{V}_{LL} & \mathbf{V}_{LU} \\ \mathbf{V}_{UL} & \mathbf{V}_{UU} \end{bmatrix}. \tag{5}$$

where $\mathbf{V}_{LL}, \mathbf{V}_{LU}, \mathbf{V}_{UL}$ and $\mathbf{V}_{UU}$ are l-by-l, l-by-u, u-by-l and u-by-u matrices, respectively. Superscripts L and U in $\mathbf{V}_{LU}$ mean rows from labeled data and columns from unlabeled data, respectively.

2.4 Recurrence relation

Applying Eq. (1) to $\mathbf{x}_i$, we get estimation for $\hat{p}(c_r|\mathbf{x}_i)$ that is p_{ir}:

$$p_{ir} = \hat{p}(c_r|\mathbf{x}_i) = \frac{k_l(r) + k_u(r)}{k_l + k_u}. \tag{6}$$

Due to introduction of the kernel function, $k_l(r)$ and $k_u(r)$ are real numbers instead of integers. The sum of them can be estimated based on influence to $\mathbf{x}_i$ from other data points:

$$k_l(r) + k_u(r) = (k_l + k_u) \sum_{j=1}^{l+u} v_{ij}\, \hat{p}(c_r|\mathbf{x}_j) = (k_l + k_u) \sum_{j=1}^{l+u} v_{ij}\, p_{jr}. \tag{7}$$

Eq. (6) and (7) combine to give

$$p_{ir} = \sum_{j=1}^{l+u} v_{ij}\, p_{jr}. \tag{8}$$

Eq. (8) can be written in matrix: $\mathbf{P} = \mathbf{VP}$. Considering Eq. (2) and (5), Eq. (8) can be further written as:

$$\begin{bmatrix} \mathbf{P}_L \\ \mathbf{P}_U \end{bmatrix} = \begin{bmatrix} \mathbf{V}_{LL} & \mathbf{V}_{LU} \\ \mathbf{V}_{UL} & \mathbf{V}_{UU} \end{bmatrix} \begin{bmatrix} \mathbf{P}_L \\ \mathbf{P}_U \end{bmatrix}. \tag{9}$$

Since we do not consider the influence to a labeled data point from other points, we substitute the identity matrix $\mathbf{I}$ for $\mathbf{V}_{LL}$, and a zero matrix $\mathbf{0}$ for $\mathbf{V}_{LU}$. Eq. (9) becomes

$$\begin{bmatrix} \mathbf{P}_L \\ \mathbf{P}_U \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{V}_{UL} & \mathbf{V}_{UU} \end{bmatrix} \begin{bmatrix} \mathbf{P}_L \\ \mathbf{P}_U \end{bmatrix}. \tag{10}$$

From Eq. (10), we obtained the recurrence relation in the data set:

$$\mathbf{P}_U = \mathbf{V}_{UL}\mathbf{P}_L + \mathbf{V}_{UU}\mathbf{P}_U. \tag{11}$$

3 Algorithm derivation

3.1 Solution 1: matrix solution

From the recurrence relation in Eq. (11), we have the matrix solution

$$\mathbf{P}_U = (\mathbf{I} - \mathbf{V}_{UU})^{-1}\mathbf{V}_{UL}\mathbf{P}_L, \tag{12}$$

assuming that the inverse of matrix $(\mathbf{I}-\mathbf{V}_{UU})$ exists. This is the matrix solution.

After $\mathbf{P}_L$ and $\mathbf{V}$ are known, by Eq. (12), we obtain $\mathbf{P}_U$, the probability distributions of all unlabeled points over classes. The class label with the largest probability for each row of $\mathbf{P}_U$ is selected for classification decision. Assume $l \ll u$, the time complexity of this solution is $O(u^{2.376} + u^2 l)$ which is the time for matrix inversion and multiplication, where u is the number of unlabeled data. This solution is appropriate for small and medium-size data sets.

Special cases: If $\lambda = 0$, then $\mathbf{V}_{UU} = \mathbf{0}$ and Eq. (12) becomes $\mathbf{P}_U = \mathbf{V}_{UL}\mathbf{P}_L$, and the solution is downgraded to supervised learning — kernel-based weighted KNN. If $\lambda = 1$, the unlabeled data have the same influence as labeled data in the graph.

3.2 Solution 2: an iterative algorithm

In some cases the matrix $(\mathbf{I} - \mathbf{V}_{UU})$ in Eq. (12) is singular or approximately singular, causing that its inverse $(\mathbf{I} - \mathbf{V}_{UU})^{-1}$ does not exist (when $det(\mathbf{I} - \mathbf{V}_{UU}) = 0$) or approaches infinity. Then the matrix resolution fails or become imprecise. In addition, matrix solution is not efficient enough for large data sets.

We present an iterative algorithm that is more efficient and more flexible. Based on Eq. (8), we propose an iterative process to solve $\mathbf{P}_U$.

Iterative process

- Initialization:
  - Normalized weight matrix $\mathbf{V}$ is initialized as in Section 2.3.
  - In class matrix $\mathbf{P}$, $\mathbf{P}_L$ is initialized as in Section 2.2. And $\mathbf{P}_U$ is initialized such that each row in it is an uniform distribution over all classes, i.e., each element is set to $\frac{1}{|C|}$.
- Repeat until convergence:
  - Update $\mathbf{P}$'s rows for unlabeled data according to Eq. (8). For all $l+1 \leq i \leq n$ $(n = l + u)$ and $1 \leq r \leq |C|$,

$$p_{ir}^{[t+1]} \leftarrow \sum_{j=1}^{n} v_{ij} p_{jr}^{[t]}. \tag{13}$$

    where $p_{ir}^{[t+1]}$ is the new state and $p_{jr}^{[t]}$ is the current state.

Please note that when updating p_{ir}, we use previously computed p_{jr}'s as soon as they are available, instead of using old estimated p_{jr}'s.

Another reason that our algorithm converges quickly is that for each unlabeled point, there are always nearest neighbors from labeled data to influence its label. Then the rate of label propagation is fast. For most points, their labels can be stable quickly. That makes our algorithm faster than some previous graph-based iterative algorithms. In addition, considering $\mathbf{V}$ is a sparse matrix, each iteration's time complexity is actually $O(ku|C|)$ where $k = k_l + k_u$, instead of $O(nu|C|)$.

Convergence: We can prove that this iterative process is convergent, considering the nonnegative energy function $E = \frac{1}{2} \sum_{i=l+1}^{l+u} \sum_{r=1}^{|C|} \sum_{j=1}^{n} v_{ij}(p_{ir} - p_{jr})^2$ decreases monotonically in the process.

Flexibility: This algorithm is also flexible and can be combined with online learning. During online learning, the learner receives feedback and some unlabel data are labeled and added to labeled data. To address this problem, the algorithm can be slightly modified to allow $\mathbf{P}_L$ to change according to feedback.

3.3 Outlier detection

In labeled data $\mathcal{D}_L$, there are possibly outliers, which could be labeled by mistake. Within the update process, we propose a method to detect outliers in $\mathcal{D}_L$. Different from Section 2.4, we consider the influence all over the data set, including both labeled data and unlabeled data. Assume the initial $\mathbf{P}_L$ is $\mathbf{P}_L^{[0]}$ and its estimation for outlier detection is $\hat{\mathbf{P}}_L$. From Eq. (9), we have

$$\hat{\mathbf{P}}_L = \mathbf{V}_{LL}\mathbf{P}_L^{[0]} + \mathbf{V}_{LU}\hat{\mathbf{P}}_U. \tag{14}$$

From Eq. (12), we have

$$\hat{\mathbf{P}}_U = (\mathbf{I} - \mathbf{V}_{UU})^{-1}\mathbf{V}_{UL}\mathbf{P}_L^{[0]}. \tag{15}$$

Algorithm 1 TKNN – transductive learning with nearest neighbors

Input:
λ: influence factor of unlabeled data.
C: the set of class labels $\{1, 2, \ldots, |C|\}$.
$\mathcal{D}_L$: set of labeled points, $\{\mathbf{x}_i\}_{1 \le i \le l}$.
Y_L: set of class labels for labeled points, $\{y_i\}_{1 \le i \le l}$ $(y_i \in C)$.
$\mathcal{D}_U$: set of unlabeled points, $\{\mathbf{x}_i\}_{l+1 \le i \le n}$ $(n = l + u)$.
k: number of nearest neighbors.
h: bandwidth for kernel method.
p_{thresh}: threshold of $\hat{p}(c_{y_i}|\mathbf{x}_i)$ for outlier detection.
Output:
Y_U: set of class labels for unlabeled points, $\{y_i\}_{l+1 \le i \le l+u}$.
Method:

```
Detects outliers D_O.
D_L ← D_L − D_O and l ← |D_L|.
D_U ← D_L ∪ D_O and u ← |D_U|.
Using λ, obtain row-normalized weight matrix V = (v_{i,j})_{n×n} with four blocks
    denoted by V_LL, V_LU, V_UL and V_UU.
Initialize class matrix P = (p_{i,r})_{n×|C|} with two blocks denoted by P_L and P_U.
if D_U is not too large and det(I − V_UU) is not too small then
   P_U ← (I − V_UU)^{-1} V_UL P_L.
else
   {Iteratively update elements of P_U until convergence.}
   repeat
      for all x_i ∈ D_U do
         for r = 1 to |C| do
            p_ir ← Σ_{j=1}^{l+u} v_ij p_jr
         end for
      end for
   until convergence
end if
Retrieve class labels {y_i}_{l+1≤i≤l+u} from class matrix P_U.
```

Substitute $\hat{\mathbf{P}}_U$ in Eq. (14) with Eq. (15) and we have

$$\hat{\mathbf{P}}_L = \left[\mathbf{V}_{LL} + \mathbf{V}_{LU}(\mathbf{I} - \mathbf{V}_{UU})^{-1}\mathbf{V}_{UL}\right] \mathbf{P}_L^{[0]}. \tag{16}$$

This is the estimation of $\mathbf{P}_L$ for outlier detection. In Eq. (16), $\mathbf{V}_{LL}$ reflects the influence between labeled data directly, and $\mathbf{V}_{LU}(\mathbf{I} - \mathbf{V}_{UU})^{-1}\mathbf{V}_{UL}$ reflects the influence between labeled data *through* unlabeled data, which is first introduced for outlier detection in this paper. Alternatively, in Eq. (14), $\hat{\mathbf{P}}_U$ can be obtained by the iterative process described in Section 3.2, which is more efficient for large data sets.

Looking for outliers, we compare each row of $\hat{\mathbf{P}}_L$ with the corresponding row of $\mathbf{P}_L^{[0]}$. Assume two corresponding rows are the row vector $\hat{\mathbf{p}}$ from $\hat{\mathbf{P}}_L$ and the row vector $\mathbf{p}_{[0]}$ from $\mathbf{P}_L^{[0]}$. Let a be the dot product of $\hat{\mathbf{p}}$ and $\mathbf{p}^{[0]}$. We know one component of $\mathbf{p}^{[0]}$ is 1 and others are 0. Then a is the component of $\hat{\mathbf{p}}$ for the

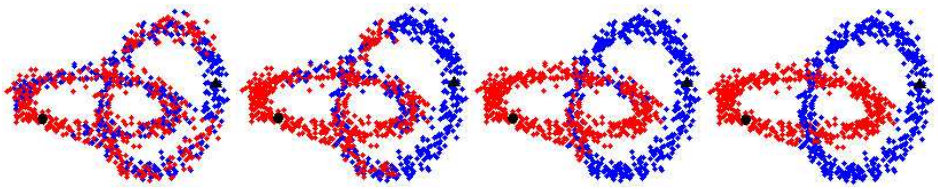

Fig. 2. The convergence of TKNN iterative algorithm on the 2-rings data set. Two labeled points are marked with • and ▲. Parameter configuration: $k_l = 1$, $k_u = 10$. From left to right: (1) after 1 iteration, accuracy 0.588; (2) after 10 iterations, accuracy 0.773; (3) after 30 iterations, accuracy 0.919; and (4) after 49 iterations, accuracy 1.0.

class labeled in $\mathbf{p}^{[0]}$. For example, if $\mathbf{p}^{[0]} = [0, 1, 0, 0]$ and $\hat{\mathbf{p}} = [0.1, 0.4, 0.3, 0.2]$, then $a = \hat{p}_2 = 0.4$. This is for one data point. For all labeled data, we have the column vector

$$\mathbf{a} = diag\left(\hat{\mathbf{P}}_L(\mathbf{P}_L^{[0]})^T\right), \tag{17}$$

where superscript T is matrix transpose, and $diag(\cdot)$ returns a vector that includes diagonal entries of a square matrix.

We decide if labeled data are outliers by vector $\mathbf{a}$. Assume a is a component of $\mathbf{a}$. If $a < p_{thresh}$, a's corresponding data point could have been mislabeled and is considered as an outlier. Generally we let the threshold p_{thresh} be $\frac{1}{|C|}$. The detected outliers are treated as unlabeled points; labeled and unlabeled data are re-partitioned before classification.

3.4 Algorithm

Based on our solution to the classification problem and outlier detection, we describe our algorithm called TKNN in Algorithm 1. We first detect outliers in labeled data and treat them as unlabeled. Then for a small or medium-size data set, we use matrix computation solution. And for a large data set, we use the iterative process solution.

4 Experiments

4.1 Data sets

Three data sets are used in our experiments. (1) 2-rings. The 2-rings data set is generated to simulate the two interlock rings problem. 1000 data points are distributed around two rings in 3D space with Gaussian noise. Only two points are labeled, one for each class. (2) Digit1. This is from benchmark of [11], two classes, 241 dimensions, 1500 points. *Digit1-10* means 10 points are labeled and *Digit-100* means 100 points are labeled. (3) USPS. This is also from [11], also two classes, 241 dimensions, 1500 points. Two classes are imbalanced with relative sizes of 1:4. *USPS-10* means 10 points are labeled and *USPS-100* means 100 points are labeled.

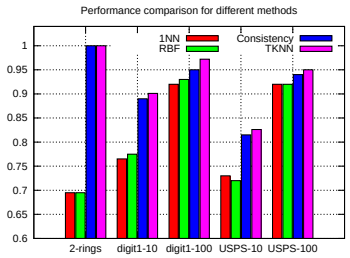

Fig. 3. Performance comparison of 1NN, RBF, Consistence method and TKNN.

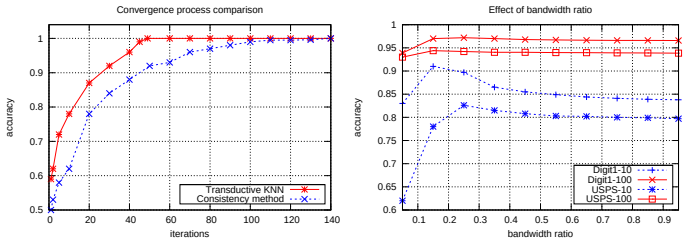

Fig. 4. (left) Convergence comparison of TKNN and consistence method on data set 2-rings. TKNN converges faster than the consistency method. **(right)** The effect of bandwidth to performance of TKNN. Here $k_l = 1$, k_u is 9 for Digit1 and 3 for USPS.

4.2 Experimental results

First, we have to mention the parameter b. In Section 2.3 we introduced bandwidth h of the kernel function. The selection of h has impact to performance. But h ranges largely for different data sets. Then we define the bandwidth ratio b as the ratio of h and average distance of a data set. b is relatively stable and after its selection, h for a specific data is obtained.

Fig. 2 shows the convergence process of our TKNN iterative algorithm on the 2-rings data set. At the beginning of the process, points are randomly classified. It can be observed that during the process, the number of randomly classified points decreases and finally reaches zero, and all points are classified correctly, with accuracy 1.0. Fig. 4 (left) shows our algorithm converges faster than the consistency method [13]. Two reasons contribute to the efficiency. First, we considers labeled neighbors and unlabeled points can receive impact from labeled data at each iteration. Second, the newest data are immediately used to compute neighbors' impact to a point at each iteration.

We compare our algorithm to 1NN (KNN with $k = 1$), Radial Basis Function (RBF) and the consistency method as shown in Fig. 3. The Gaussian function is used for RBF. The best performance is recorded for each method. We can observe that when the number of labeled points is less (2-rings, Digit1-10 and USPS-10), transductive learning improves more from supervised learning that is 1NN and RBF here. While both TKNN and the consistency method achieve

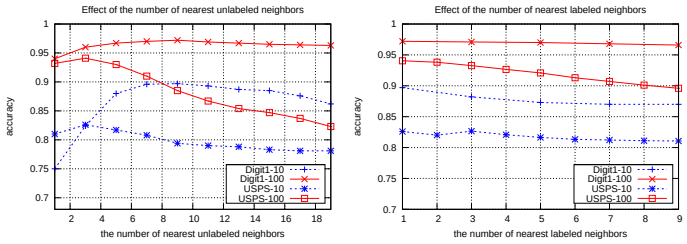

Fig. 5. (left) The effect of k_u of unlabeled data to performance of TKNN. Here $k_l = 1$ and b is 0.25 for Digit1 and 0.35 for USPS. **(right)** The effect of k_l of labeled data to performance of TKNN. Here $k_u = 9$, $b = 0.25$ for Digit1 and $k_u = 3$, $b = 0.35$ for USPS.

1.0 accuracy on the 2-rings data set, TKNN has the best performance on other data sets.

Fig. 4 (right) for b, Fig. 5(left) for k_u and Fig. 5(right) for k_l show effect of parameters. The performance is best when the bandwidth ratio $0.1 < b < 0.3$, the number of nearest unlabeled neighbors $3 \leq k_u \leq 10$ and the number of nearest labeled neighbors $k_l = 1$. The best value of k_u depends on data. The best value $k_l = 1$ agrees to human intuition that k_l/k_u should be proportional to ratio of number of l/u, the ratio of sizes of labeled data and unlabeled data.

5 Related work

Our work is related to semi-supervised learning, especially graph-based learning. [4] introduced a method that take advantage of unlabeled data for classification tasks. After that, semi-supervised learning or trandsductive learning have been extensively researched. And graphs are introduced in semi-supervised learning [14]. Work in [10][13][15] is most related to our work. [10] proposed a semi-supervised learning based on a Gaussian random field model. Labeled and unlabeled data are represented in a weighted graph. The mean of the field is efficiently obtained using matrix methods or belief propagation. [13] proposed the consistency method that is a smooth solution which captures the intrinsic structure revealed by labeled and unlabeled points. Different from [10] and [13], we present an efficient iterative algorithm for large data sets and for circumstances where matrix computation fails because of the nonexistent inverse matrix. [15] introduced a two-stage approach. In the first stage, a model is built based on training data. In the second stage, the model is used to transform unlabeled data into weighted pre-labeled data set which is used in the classifier. Our work is not two-stage and influence between labeled and unlabeled data are considered at the same time. [9] proposed a graph-based approach, mixed label propagation, that explores similarity and dissimilarity simultaneously.

6 Conclusion

In KNN algorithm, only labeled data can be the nearest neighbors. In this paper, we extend KNN to a transductive learning algorithm called TKNN. We consider two groups of nearest neighbors. One from labeled data and the other from unlabeled data. We then derive the recurrence relation in a data set, considering that each point receives influence from neighboring points, labeled or unlabeled. A matrix solution is then proposed and a more efficient iterative algorithm is presented. We compare our algorithm with baseline algorithms. Experimental results show our algorithm are effective and efficient. Future work will be focused on more precise estimation of parameters and application on data with more than two classes.

Acknowledgments. This work is supported in part by the following NSF grants: IIS-0414981 and CNS-0454298.

References

1. Webb, A.R.: Statistical Pattern Recognition. John Wiley and Sons Ltd. (2002)
2. Bennett, K.P., Demiriz, A.: Semi-supervised support vector machines. In: NIPS. (1998)
3. Kulis, B., Basu, S., Dhillon, I., Mooney, R.: Semi-supervised graph clustering: A kernel approach. In: ICML. (2005)
4. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Learning to classify text from labeled and unlabeled documents. In: AAAI. (1998)
5. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using em. Machine Learning **1**(34) (1999)
6. Seeger, M.: Learning with labeled and unlabeled data. Inst. for Adaptive and Neural Computation, technical report (2001)
7. Sindhwani, V., Niyogi, P., Belkin, M.: Beyond the point cloud: from transductive to semi-supervised learning. In: ICML. (2005)
8. Tang, W., Xiong, H., Zhong, S., Wu, J.: Enhancing semi-supervised clustering: A feature projection perspective. In: KDD. (2007)
9. Tong, W., Jin, R.: Semi-supervised learning by mixed label propagation. In: AAAI. (2007)
10. Zhu, X., Ghahramani, Z., Lafferty, J.: Semi-supervised learning using gaussian fields and harmonic functions. In: ICML. (2003)
11. Chapelle, O., Schölkopf, B., Zien, A., eds.: Semi-Supervised Learning. MIT Press, Cambridge, MA (2006)
12. Tran, T.N., Wehrensa, R., Buydens, L.M.: Knn-kernel density-based clustering for high-dimensional multivariate data. Computational Statistics & Data Analysis **51**(2) (November 2006) 513–525
13. Zhou, D., Bousquet, O., Lal, T., Weston, J., Schölkopf, B.: Learning with local and global consistency. In: NIPS. (2003)
14. Blum, A., Chawla, S.: Learning from labeled and unlabeled data using graph mincuts. In: ICML. (2001)
15. Driessens, K., Reutemann, P., Pfahringer, B., Leschi, C.: Using weighted nearest neighbor to benefit from unlabeled data. In: PAKDD. (2006) 60–69