데이터 분포에 기반한 유사 군집 선택법

김계현° 최승진

포항공과대학교 컴퓨터공학과

fenrir@postech.ac.kr, seungjin@postech.ac.kr

Neighborhood Selection with Intrinsic Partitions

Kye-Hyeon Kim ° Seungjin Choi

Department of Computer Science, POSTECH, Korea

Abstract

We present a novel method for determining k nearest neighbors, which accurately recognizes the underlying clusters in a data set. To this end, we introduce the "tiling neighborhood" which is constructed by tiling a number of small local circles rather than a single circle, as existing neighborhood schemes do. Then we formulate the problem of determining the tiling neighborhood as a minimax optimization, leading to an efficient message passing algorithm. For several real data sets, our method outperformed the k-nearest neighbor method. The results suggest that our method can be an alternative to existing methods for general classification tasks, especially for data sets which have many missing values.

1. Introduction

Finding similar elements of an item plays a key role in most of the pattern recognition tasks such as classification, clustering and information retrieval, since a group of similar items generally forms a meaningful pattern. The k-nearest neighbor method (k-NN) [1] was proposed in this perspective and has been widely used. Given an unlabeled test data point x_*, the method first finds the k nearest points of x_* in the training set of labeled data points using a distance metric. It then chooses the most frequent label occurring in the k nearest neighbors for the class label of x_* (majority voting). The success of k-NN depends on how many neighbors are "actually similar" to the test data point.

Various methods for global or local metric learning have been developed [2-6] to discover a proper distance measure for a given data set. However, they have failed to achieve success in practical applications because (1) a real data set generally consists of several arbitrary-shaped partitions but those methods cannot reflect such nonlinearity and discontinuity; (2) those methods take too much time compared to k-NN. In this paper, we develop a new way of assigning points to a neighborhood.

We present a new way for neighborhood construction which adequately captures and reflects the intrinsic partitions of a data set. To this end, we introduce the "tiling neighborhood". In contrast to k-nearest neighborhood where a single circle determines a neighborhood of a data, the tiling neighborhood is constructed by tiling a number of small local circles whose centers are referred to as tiling neighbors.

We illustrate the tiling neighborhood in detail, including how to determine such neighborhoods, and associated useful properties of it. Then, for classification, we present an efficient message passing algorithm which allows us to predict a class label in the framework of the tiling neighborhood using local computations. Numerical experiments with several data sets, including a collaborative filtering application, confirm the useful behavior of the proposed method, compared to k-NN.

2. Theory and Algorithm

The main motivation of our method is to estimate appropriate intrinsic clusters as neighborhoods, for successful nearest neighborhood classification (Fig. 1). However, k-NN may not accurately detect an intrinsic partition of a data set. We present a new neighboring method which preserves the intrinsic partition of a data set. The intuitive idea is shown in Fig. 1. The desirable neighborhood in Fig. 1 is constructed by tiling a number of circles with diameter δ, following the tiling rule: *A*

circle whose center corresponds to a data point must contact or overlap one or more other circles.

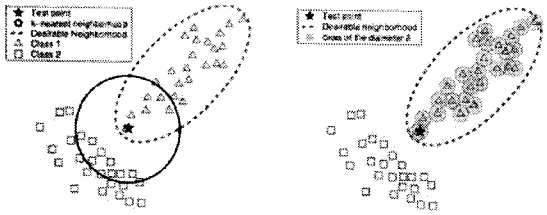

Figure 1. (Left) The difference between a k-NN and our tiling neighborhood. The solid circle represents a k-NN of a test data point (green star). The intrinsic partition is not detected in this case, leading to the misclassification. (Right) In contrast, the tiling neighborhood exactly reconstructs the desirable result by joining a number of small circles. In such a case, the test data point is correctly classified.

Let us start from a circle initially attached to the test point, and attach each circle by the rule until there is no point that cannot be attached. Then, the set of data points covered by circles can be seen as a cluster whose size and shape are determined by δ. By choosing the proper value of δ, we can obtain the correct neighborhood, denoted by the dotted ellipse in both sides of Fig. 1.

Now we define the tiling neighborhood. Let V be a finite set and d: V×V→R be a metric on V. Given v∈V and δ ≥ 0, an element u∈V is called a tiling neighbor of v, denoted by u~v, if the following condition is satisfied: *There exists a series of elements w between u and v such that*

$$d(u,w) \le \delta \quad \text{for} \quad w \sim v.$$

The inequality d(u,w) ≤ δ represents the tiling rule such that two circles must contact or overlap each other. The notation ~ is a relation (a subset of the cartesian product V×V) on V and u~v means "u is a tiling neighbor of v". A tiling neighborhood of v, denoted by T(v), is defined by a set of all eligible tiling neighbors of v, i.e.,

$$T(v) = \{u \in V \mid u \sim v\}.$$

In general, finding a tiling neighborhood requires the computational complexity $O(|V|^2)$, hence it is much slower than finding a k-NN. Now we formulate the problem of selecting a tiling neighborhood as a minimax optimization, and then develop an efficient message passing algorithm. We denote by S(u,v) a sequence from u to v, i.e., S(u,v) = (u,⋯,v). It follows from Sec. 2.1.3 that we can determine whether a sequence S(u,v) is δ-bounded or not, by checking whether there is no pair of adjacent elements (a,b)∈S(u,v) such that d(a,b) > δ. Clearly, if max d(a,b) ≤ δ for any two adjacent elements a and b in the sequence S(u,v), then the path S(u,v) is δ-bounded. For a pair of elements u and v, there may be several possible sequences from u to v. Among those sequences, if the minimum value of max d(a,b) ≤ δ, there exists one or more δ-bounded sequences so that u~v is satisfied. Consequently, if we compute the minimax distance between u and v as

$$d_{mm}(u,v) = \min S(u,v) \{ \max (a,b) \in S(u,v)\, d(a,b) \}$$

for all u∈V, then we can find a tiling neighborhood of v as {u | $d_{mm}(u,v) \le \delta$}. The corresponding sequence(s) is called a minimax path and a tree T is called a minimax spanning tree if the paths between all pairs u,v∈V are minimax paths.

Since there are enormously many possible sequences between a pair, computing the minimax distance seems to be a very complicated task. However, a minimum spanning tree (MST) is also a minimax spanning tree [7]. That is, every path between two nodes u and v in an MST, denoted by M(u,v), is also a minimax path between them. Hence, we can obtain the minimax distance d_{mm} between any pair u,v∈V by simply computing

$$d_{mm}(u,v) = \max (a,b) \in M(u,v)\, d(a,b).$$

Let X be a set of training points $\{x_1,\cdots,x_N\}$ and x_* be a test point. To find a tiling neighborhood of x_*, the minimax distance $d_{mm}(x_i,x_*)$ should be computed for all $x_i \in X$. Clearly, we can easily obtain them by constructing an MST of the set $X \cup \{x_*\}$ and then computing them. However, constructing the MST takes at least $O(N^2)$ time, so this approach is too inefficient to be practically used. Now we derive a formula for computing $d_{mm}(x_i,x_*)$ without constructing an MST every time. There are N types of paths between x_i and x_* such that $(x_i,\cdots,x_1,x_*)$, $(x_i,\cdots,x_2,x_*)$, ⋯, $(x_i,\cdots,x_N,x_*)$. All paths $S(x_i,x_*)$ belong to one of the types. Hence, there are N candidates for the minimax path between x_i and x_* such that $(M(x_i,x_1),x_*)$, $(M(x_i,x_2),x_*)$, ⋯, $(M(x_i,x_N),x_*)$, and the minimax distance $d_{mm}(x_i,x_*)$ is

$$\begin{aligned} d_{mm}(\boldsymbol{x}_i,\boldsymbol{x}_*) &= \min_j \max \left[d_{mm}(\boldsymbol{x}_i,\boldsymbol{x}_j), d(\boldsymbol{x}_j,\boldsymbol{x}_*) \right] \\ &= \min \left[d(\boldsymbol{x}_i,\boldsymbol{x}_*),\ \min_{j \ne i} \max \left[d_{mm}(\boldsymbol{x}_i,\boldsymbol{x}_j), d(\boldsymbol{x}_j,\boldsymbol{x}_*) \right] \right], \end{aligned}$$

where the second inequality is derived from the fact that max $[d_{mm}(x_i,x_i), d(x_i,x*)]$ is simply $d(x_i,x*)$ for j = i.

Now we develop a message passing algorithm that determines a tiling neighborhood, when a minimum spanning tree of $v=\{x_1,\cdots,x_N\}$ is given. Let ε be a set of edges on the MST and $\mathcal{N}_i$ be an index set

such that $\{j \mid (i,j)\in\varepsilon\}$. Then, minimax distances between x_* and $\{x_1,\cdots,x_N\}$ can be efficiently computed on the MST by the following message passing algorithm:

$$m_{ij} = \max\left[d(x_i,x_j),\ \min\left[d(x_i,x_*),\ \min_{k\in N_i-\{j\}} m_{ki}\right]\right],\quad \forall(i,j)\in\mathcal{E},$$

$$d_{mm}(x_i,x_*) = \min\left[d(x_i,x_*),\ \min_{k\in N_i} m_{ki}\right],\quad \forall x_i\in V.$$

The algorithm can obtain all minimax distances and the tiling neighborhood of x_* in O(N) time. It is equal to the time complexity of the k-NN.

3. Numerical Experiments

We compared the classification performance between a k-NN and our tiling neighborhood using five *UCI* data sets and the *MovieLens* data set. To predict a class label of a test data point, we used the average rating and the majority voting. The performance is measured by the average absolute error and the average zero-one error rate.

We used five data sets taken from UCI Machine Learning Repository[1]: Iris, Sonar, Glass, Vowel, and Segmentation. Iris consists of 100 4-dimensional data points. Sonar consists of 208 60-dimensional data points. Glass consists of 214 9-dimensional data points. Vowel consists of 528 10-dimensional data points. Segmentation consists of 2310 19-dimensional data points. We also performed experiments with the MovieLens recommend system where the data set is collected by the GroupLens Research Project[2]. The data set consists of 100,000 ratings (scaled from 1 to 5) from 943 users on 1682 movies.

For Iris, Sonar, and Glass, we performed N experiments where N is the number of data points. For the i-th experiment, we select the i-th data point as a test point, and the remaining data points as a training set.

For Vowel, we perform 10 experiments by selecting 200 data points randomly as a training data set, and the remaining data points as a test set.

For Segmentation, we divide the whole data into 10 subsets and then perform 10 experiments such that one subset as a test set and the others as a training set.

For MovieLens, we selected 4 groups of uses as the test sets according to the number of ratings of each user: users who have rated (1) 150~500 movies, (2) 75~250, (3) 45~150, and (4) 20~50. For each group, we randomly selected 10, 30, 50, or 100 users respectively as the test cases. For each user in that group, we randomly selected 100, 50, 30, or 10 ratings respectively as the observed features, and the other ratings were used for prediction. To predict the rating, we define the distance between two users u and v as the average difference of ratings such that $d(u,v) = \sum_m |r(u,m)-r(v,m)|/N_{uv}$, where r(u,m) is the rating on the movie m given by the user u, and N_{uv} is the number of movies rated by both u and v.

For all data sets, we used majority voting, i.e., y_* is labeled as a majority of y_js in $T(x_*)$. For movie lens, we also used the average rating, i.e., $y_* = \sum y_j/|T(x_*)|$, where y_* is the predicted rating, y_j is the class label of x_j, and $T(x_*)$ is a tiling neighborhood of a test point x_*. The average rating is used only in the experiments on MovieLens, because it is appropriate for ordinal labeling (e.g. ratings) but not for nominal labeling.

For all data sets, we used the average zero-one error rate, i.e., (the number of misclassified test points)/M. For MovieLens, we also used the average absolute error which the average deviation of the prediction from the true target, i.e., $\sum |y_*^{(i)} - y^{(i)}|/M$ for a set of M test points. The average absolute error measure is used only in the experiments on MovieLens, since the measure is appropriate for user-preference prediction tasks but not for common classification tasks.

4. Results

Our method outperformed k-NN for all UCI data sets (Table 1): For Iris, the average error rate of our method was 0.2%P[3] lower than that of k-NN. For Sonar, it was 0.5%P lower. For Glass, it was 0.9%P lower. For Vowel, it was about 2%P lower. For Segmentation, it was 0.5%P lower.

Our method outperformed k-NN for MovieLens, for all groups and rating methods, with respect to both absolute error (Table 2) and zero-one error rate (Table 3). MovieLens is a very sparse data set. For 943 users and 1,682 movies, there are 1,586,126 possible combinations of ratings, but MovieLens provides only 100,000 ratings. That is, about 94% of values are unknown. For users who give many ratings (Group 1),

[1] UCI Machine Learning Repository: http://www.ics.uci.edu/~mlearn/MLRepository.html

[2] The GroupLens Research Project: http://www.grouplens.org

[3] Percentage point, the arithmetic difference of two percentages.

Table 1. Zero-one classification error rate (%) for UCI data sets. For Iris, Sonar, and Glass, n = 1; For Vowel and Segmentation, means and standard deviations (n = 10) of classification error rates are given.

Method	Data Set				
	Iris	Sonar	Glass	Vowel	Segment
Tiling	0.4	12.0	27.1	9.72 ± 0.89	3.1 ± 0.91
k-NN	0.6	12.5	28.0	11.8 ± 2.56	3.6 ± 1.27

Table 2. Absolute error for MovieLens. Means and standard deviations (n = # of users in each test group) are given. Users in Group 1 rate 150~500 movies, in Group 2 rate 75~250 movies, in Group 3 rate 45~150 movies, in Group 4 rate 20~50 movies.

	Rating Method			
	Average Rating		Majority Voting	
	Tiling	k-NN	Tiling	k-NN
Group 1	0.73 ± 0.85	0.78 ± 0.89	0.76 ± 0.81	0.81 ± 0.83
Group 2	0.73 ± 0.77	0.77 ± 0.77	0.79 ± 0.79	0.84 ± 0.81
Group 3	0.74 ± 0.74	0.80 ± 0.79	0.78 ± 0.79	0.84 ± 0.84
Group 4	**0.86** ± 0.89	**0.96** ± 0.95	**0.93** ± 0.87	**1.01** ± 0.95
Mean	**0.77±0.81**	**0.83±0.85**	**0.82±0.82**	**0.88±0.88**

Table 3. Zero-one error rate (%) for MovieLens. Means and standard deviations (n = # of users in each test group) are given.

	Rating Method			
	Average Rating		Majority Voting	
	Tiling	k-NN	Tiling	k-NN
Group 1	55.5 ± 4.7	58.3 ± 5.1	57.8 ± 5.9	60.1 ± 5.0
Group 2	58.2 ± 3.9	61.1 ± 4.8	60.6 ± 3.1	62.5 ± 3.7
Group 3	59.5 ± 7.3	61.4 ± 8.1	59.6 ± 6.5	61.1 ± 6.2
Group 4	**62.8** ± 12.1	**66.5** ± 13.8	65.9 ± 10.1	67.3 ± 11.2
Mean	**59.0±7.0**	**61.8±8.0**	**61.0±6.4**	**62.8±6.5**

distances between the users and the other users are generally reliable, because there are plenty of observed features used to compute the distances. On the other hand, for users who give few ratings (Group 4), distances between the users and the other users are generally not reliable. However, our method achieved more improvement (the results of k-NN as baselines) in Group 4 than in Group 1, except for zero-one error rate with majority voting: For absolute error with average rating, the average error of our method was 0.10 smaller than that of k-NN in Group 4, while 0.05 smaller in Group 1. For absolute error with majority voting, it was 0.08 smaller in Group 4, while it was 0.05 smaller in Group 1. For zero-one error rate with average rating, the average error rate was 3.7%P smaller in Group 4, and 2.8%P smaller in Group 1.

5. Discussion

k-NN, the most popular neighboring method up to now, is incapable of detecting intrinsic partitions of a data set. Such incapability makes k-NN choose many irrelevant data points as neighbors of a test data point, and thus brings the poor classification performance. To remedy this problem, we proposed the tiling neighborhood method which accurately detects the intrinsic partitions. In this section, we discuss the results in Section 4 to show that our method can effectively solve the problem.

For all UCI data sets, our method obtained better results than k-NN. In general, most of the misclassifications occur at the boundary between two classes because nearest neighbors of a query include many irrelevant data points. On the other hand, our method is more robust than the existing classification methods because it retrieves the appropriate intrinsic cluster by (tiling) neighbors whether the query is near the boundary or not. Hence, our method is better at general classification tasks than the existing methods including k-NN.

For MovieLens, our method improved grouping more for users who give fewer ratings than for groups which gave more ratings. Since we define the distance between two users as the average difference of their ratings, such a group tends to yield many unreliable test-training distances $d(x_*,x_i)$ and thus many misclassifications. However, our method uses minimax distances $d_{mm}(x_i,x_*)$ which consider not only test-training distances $d(x_*,x_i)$, but also training-training distances $d(x_i,x_j)$ to compute them. Hence, our method is especially useful for data sets having many missing values because minimax distances can be reliable even though most of the test-training distances are unreliable.

6. Conclusion

We have proposed a novel approach to constructing a neighborhood that consists of a number of circles that overlap or contact each other. This approach identified

a tiling neighborhood that preserved the intrinsic partition of a data set by satisfying the equivalence relation. For implementation, we derived a message passing algorithm whose computational complexity is compared to k-NN. Experiments with real-world data sets showed the usefulness of our method for classification tasks, especially with a data set having many missing values.

Acknowledgments

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement) (IITA-2007-C1090-0701-0045).

References

[1] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, vol. IT-13, pp. 21-27, 1967.

[2] C. Domeniconi, J. Peng, and D. Gunipulos. Locally adaptive metric nearest-neighbor classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, pp. 1281-1285, 2002.

[3] A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Advances in Neural Information Processing Systems (NIPS 18)*, pp. 451-458, 2006.

[4] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighborhood component analysis. In *Advances in Neural Information Processing Systems (NIPS 17)*, pp. 513-520, 2005.

[5] K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Advances in Neural Information Processing Systems (NIPS 18)*, pp. 1473-1480, 2006.

[6] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems (NIPS 15)*, pp. 505-512, 2003.

[7] H. S. Ng, K. P. Lam, and W. K. Tai. Analog and VLSI implementation of connectionist network for minimum spanning tree problems. In *Proceedings of IEEE Region 10 International Conference on Microelectronics and VLSI (TENCON '95)*. pp. 137-140, 1995.