Survey of Improving K-Nearest-Neighbor for Classification

Liangxiao Jiang, Zhihua Cai, Dianhong Wang, Siwei Jiang
Faculty of Computer Science, China University of Geosciences, Wuhan 430074, China
ljiang@cug.edu.cn

Abstract

KNN (k-nearest-neighbor) has been widely used as an effective classification model. In this paper, we summarize three main shortcomings confronting KNN and single out three main methods for overcoming its three shortcomings. Keeping to these methods, we try our best to survey some improved algorithms and experimentally tested their effectiveness. Besides, we discuss some directions for future study on KNN.

1 Introduction

Learning classifiers to address the classification problems is a fundamental issue in data mining. Typically, a set of training instances with corresponding class labels is given, and a classifier is learned and used to predict the class of an unseen instance. An instance is represented by an attribute vector. In this paper, an instance x is described by an attribute vector $< a_1(x), a_2(x), \ldots, a_n(x) >$, where $a_i(x)$ denotes the value of the ith attribute A_i of x, and we use C and c to denote the class variable and its value respectively. The class of the instance x is denoted by $c(x)$.

KNN (k-nearest-neighbor)[2] has been widely used in classification problems. KNN is based on a distance function that measures the difference or similarity between two instances. The standard Euclidean distance $d(x, y)$ between two instance x and y is often used as the distance function, defined as follows.

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(a_i(x) - a_i(y))^2} \tag{1}$$

Given an instance x, KNN assigns the most common class of x's k nearest neighbors to x, as shown in Equation 2. KNN is an example of lazy learning[1]. Lazy learning simply stores training data at training time and delays its learning until classification time. In contrast, eager learning generates an explicit model at training time.

$$c(x) = \arg\max_{c \in C} \sum_{i=1}^{k} \delta(c, c(y_i)) \tag{2}$$

where y_1, y_2, $\cdots$, y_k are the k nearest neighbors of x, k is the number of the neighbors, and $\delta(c, c(y_i)) = 1$ if $c = c(y_i)$ and $\delta(c, c(y_i)) = 0$ otherwise.

Although KNN has been widely used for decades due to its simplicity, effectiveness and robustness. there exist three main shortcomings confronting KNN according to our observation: 1) The distance function for measuring the difference or similarity between two instances is the standard Euclidean distance; 2) The neighborhood size is artificially assigned as an input parameter; 3) The class probability estimation is based on a simple voting.

Responding to these shortcomings, we single out three main approaches for improving KNN's classification accuracy in this paper: 1) Use some more accurate distance functions to replace the standard Euclidean distance; 2) Search a best neighborhood size to replace the artificial input parameter k; 3) Find some more accurate class probability estimation methods to replace the simple voting method.

Keeping to these methods, we try our best to survey some improved algorithms and experimentally tested their effectiveness. The rest of the paper is organized as follows. In section 2, we survey some improved algorithms of KNN. In section 3, we describe the experimental setup and results in detail. In section 4, we make conclusions and discuss some directions for future study on KNN.

2 Improved K-Nearest-Neighbor Classifiers

2.1 Improving by Distance Function

KNN is based on a distance function that measures the difference or similarity between two instances. Its inductive bias corresponds to an assumption that the classification of a test instance will be most similar to the classification of other instances that are nearby in Euclidean space. Thus, how to define the distance function is crucial.

In KNN, the standard Euclidean distance is used. Thus, the distance between instances is calculated based on all attributes of the instance. However, when the problem we research include irrelevant attributes, this standard Euclidean distance will be dominated by the large number of irrelevant attributes and become inaccurate. This difficulty, which arises when many irrelevant attributes are present, is sometimes referred to as the *curse of dimensionality*. KNN is especially sensitive to this problem.

A drastic approach[16] to deal with the *curse of dimensionality* is to completely eliminate the least relevant attributes from the attribute space when calculating the distance between two instances. This approach is widely known as *feature selection*. To address this problem, a large amount of algorithms are presented. For example, Kohavi et al.[13] proposed an approach called *wrapper* to search for an optimal feature subset. Besides, the greedy search[15] and genetic search[10] methods are often alternative.

Another effective approach to overcoming the *curse of dimensionality* problem is to weight each attribute differently when calculating the distance between two instances. This approach is widely known as *attribute weighting*. Thus, the distance function should be defined as:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} w_i^2 (a_i(x) - a_i(y))^2} \quad (3)$$

where $w_i (i = 1, \ldots, n)$ is the weight of attribute A_i.

Aiming at building text classifiers using KNN, Han[7] presented a method for attribute weighting. He propose to weight the importance of discriminating words using *mutual information* between each word and the class variable. In his paper, he called his improved algorithm weight adjusted k-nearest-neighbor, simply WAKNN. Friedman[5] gives the complete definition of *mutual information* between between each pair of variables.

Thus, when all the attributes are nominal, the attribute-weighted distance function can be defined as:

$$d(x,y) = \sum_{i=1}^{n} I_P(A_i; C)\delta(a_i(x), a_i(y)) \quad (4)$$

Where $I_P(A_i; C)$ is the mutual information between the attribute variable A_i and the class variable C, and $\delta(a_i(x), a_i(y)) = 0$ if $a_i(x) = a_i(y)$ and $\delta(a_i(x), a_i(y)) = 1$ otherwise. To test its effect, We implement WAKNN in Weka[20]. Our experimental results in section 3 show that WAKNN outperforms KNN significantly.

Besides, Huang[8] also presented a frequency-based distance function, he called it *dissimilarity measure*, to address the clustering problem in categorical domains. Now, we rewrite its definition as follow.

$$d(x,y) = \sum_{i=1}^{n} \frac{F(a_i(x)) + F(a_i(y))}{F(a_i(x))F(a_i(y))}\delta(a_i(x), a_i(y)) \quad (5)$$

where $F(a_i(x))$ and $F(a_i(y))$ respectively is a count of the number of the training instances having attribute-value $a_i(x)$ and $a_i(y)$, and $\delta(a_i(x), a_i(y)) = 0$ if $a_i(x) = a_i(y)$ and $\delta(a_i(x), a_i(y)) = 1$ otherwise. This distance function is similar to the *chi-square distance*[6].

The value difference metric (VDM) presented by Stanfill and Waltz[17] is another appropriate distance function for nominal attributes. A simplified version of the VDM (without the weighting schemes) can be defined as:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}\sum_{c=1}^{C} (P(c|a_i(x)) - P(c|a_i(y)))^2} \quad (6)$$

where C is the number of output classes; $P(c|a_i(x))$ is the conditional probability that the output class is c given that the attribute A_i of x has the value $a_i(x)$; $P(c|a_i(y))$ is the conditional probability that the output class is c given that the attribute A_i of y has the value $a_i(y)$.

The VDM was designed to calculate distance values between nominal attribute values, but it largely ignores continuous attributes, requiring discretization to map continuous values into nominal values. Thus, three new heterogeneous distance functions (HVDM, IVDM, and WVDM) are designed by Wilson and Martinez[19] to handle applications with nominal attributes, continuous attributes, or both.

2.2 Improving by Neighborhood Size

KNN is a k-related algorithm. Its classification accuracy is sensitive to the value of k. Thus, a natural thought is designing a learning algorithm to automatically search a best k value for KNN. To achieve this goal, the most direct method is try various k values and choose the best one. Based on this idea, Xie et al.[21] propose a model called selective neighborhood naive Bayes, simply SNNB. In SNNB, given a test instance, various k values are tested. For each k value, a local naive Bayes[14] is learned from the k nearest neighbors and is evaluated. After this, the most accurate naive Bayesian classifier is used to classify the test instance. Although SNNB demonstrates good classification performance, it has very high time complexity.

Known from the basic idea of SNNB, the process searching for the best k value is very time-consuming. This limits its use in many real-world data mining applications. In fact, a more efficient approach used to learn the best value of k for KNN is described in the file of *weka.classifiers.lazy.IBk.java* in the Weka-3-4 version. The algorithm implemented by function of *crossValidate()*. It select the best value for k by executing a brute-force search based on leave-one-out cross-validation. Compared to SNNB, it select the best value of k for the whole training instances instead of the single test instance. Once the best value of k is learned at training time, it can be used by KNN

for all test instances at classification time. In this paper, we call the KNN using this method to search the best k value dynamic k-nearest-neighbor, simply DKNN. In DKNN, the meaning of *dynamic* is the value of k in KNN varies dynamically with training instances. We implement DKNN in Weka[20]. Our experimental results in section 3 show that DKNN outperforms KNN significantly.

Motivated by DKNN, SNNB and the method of attribute weighting, we proposed another improved algorithm called DKNAW[9]. In DKNAW, we firstly weight attributes using mutual information between each attribute and the class variable when we calculate the distance between two instances. Then, a k value best fitting the training data is learned at training time. At classification, this best k value is used. This is a strategy of combining lazy learning with eager learning. Finally, a local naive Bayesian classifier is learned within the neighborhood of the test instance, with which the instance is classified.

In a word, all these algorithms need to determine (artificially assigning or automatically search) the size of neighborhood for finding k nearest neighbors for each test instance. Thus, an algorithm without determining the size of neighborhood is desirable. To achieve this, KDTree[3] and NBTree[12] are presented. In KDTree and NBTree, instances are stored at the leaves of the tree, with nearby instances stored at the same or nearby leaf nodes. The internal nodes of the tree sort the test instance to the relevant leaf by testing selected attributes of it. Besides, Zheng et al.[18] presented another approach to find the neighborhood of each test instance by producing decision rules.

2.3 Improving by Class Probability Estimation

KNN uses a simple voting to produce the class probability estimation. That is say that the class labels of instances in the neighborhood are treated equally. So, an obvious improved method is to weight the vote of k nearest neighbors differently according to their distance to the test instance x. The resulting classifier is called k-nearest-neighbor with distance weighted[16] (KNNDW). We implement KNNDW in Weka[20]. Our experimental results in section 3 show that KNNDW outperforms KNN significantly.

$$c(x) = \arg\max_{c \in C} \sum_{i=1}^{k} \frac{\delta(c, c(y_i))}{d(y_i, x)^2} \tag{7}$$

Another most efficient approach is deploying a local probability-based classification model within the neighborhood of the test instance. Talking of the local probability-based classification models, naive Bayes is absolutely necessary. The idea of combining KNN with naive Bayes is quite straightforward. Like all lazy learning methods, the training data is simply stored, and learning is deferred until classification time. Whenever a test instance is classified, a local naive Bayes is trained using the k nearest neighbors of the test instance, with which the test instance is classified. In fact, the conclusion that naive Bayes performs well when the training data is small drew by Kohavi[12] underlies the success of combining KNN with naive Bayes.

In recently years, researchers have done considerable work on combining KNN with naive Bayes. Besides SNNB[21] and DKNAW[9] discussed before, locally weighted naive Bayes, simply LWNB, presented by Frank et al.[4] is a state-of-the-art example. In LWNB, k nearest neighbors of the test instance are firstly found and each of them is weighted in terms of its distance to the test instance. Then a local naive Bayes is built from the locally weighted training instances. Although it is a k-related algorithm, its classification performance is not particularly sensitive to the size of k as long as it is not too small.

Last year, Jiang et al.[11] observed that keeping the size of the neighborhood (the values of k) small will reduce the chance of encountering strong dependencies confronting naive Bayes. When k is small, however, the class probability estimation of naive Bayes isn't reliable. Thus, the resulting classification would be inaccurate. To deal with this contradiction. Another new algorithm called instance cloning local naive Bayes (ICLNB) is presented. In ICLNB, for each neighbor, a number of clones are firstly generated and added into the local training data in terms of its similarity to the test instance. Then, a naive Bayes is trained from the expanded local training data.

In fact, A more straightforward approach to combining KNN with naive Bayes is that learning a local naive Bayes directly using the k nearest neighbors of the test instance to produce class probability estimation and classify this test instance. The resulting classifier is called k-nearest-neighbors naive Bayes (KNNNB). We implement KNNNB in Weka[20]. Our experimental results in section 3 show that KNNNB outperforms KNN significantly.

3 Experimental Methodology and Results

We run our experiments under the framework of Weka to study the effectiveness of above improved methods on KNN. We run our experiments on 36 UCI data sets selected by Weka[20], which represent a wide range of domains and data characteristics. In our experiments, we adopted three preprocessing steps.

1. Missing values: We used the unsupervised filter *ReplaceMissingValues* in Weka to replace the missing values in each data set.

2. Discretization of numeric attributes: We used the unsupervised 10-bin filter *Discretize* in Weka to discretize

all the numeric attributes.

3. Removal of useless attributes: Apparently, if the number of values of an attribute is almost equal to the number of instances in a data set, it does not contribute useful information to classification. Thus, we removed this type of attributes using the unsupervised filter *Remove* in Weka.

We conduct extensive empirical comparison for KNN and its related improved algorithms WAKNN, DKNN, KNDW, and KNNNB in terms of classification accuracy. The value of k in all k-related algorithms is 10. We use the implementation of KNN in Weka system. We implement WAKNN, DKNN, KNNDW, and KNNNB in Weka.

The classification accuracy of each classifier on each data set was obtained via 10-fold cross validation. Run with the various algorithms were carried out on the same training sets and evaluated on the same test sets. Finally, we compare KNN and its improved algorithms via two-tailed t-test with a 95% confidence level.

Table 1 shows the classification accuracy and standard deviation of KNN and its improved algorithms on each data set. The symbols v and * in the tables respectively denotes statistically significant improvement or degradation over KNN with a 95% confidence level. The average values and the $w/t/l$ values of two-tailed t-test is summarized at the bottom of the table. The experimental results show that all of them significantly outperforms KNN.

4 Conclusions and Future Work

In this paper, we make a survey on improving KNN for accurate classification. Our main work include: 1) According to our observation, we summarize three main shortcomings confronting KNN; 2) Responding to these shortcomings, we single out three main methods for overcoming its shortcomings; 3) Keeping to these improving methods, we try our best to survey some improved algorithms and experimentally tested their effectiveness.

KNN has been widely used in various data mining applications as an effective classification model. Learning it for accurate classification is one of the main research directions, but not all, because an accurate class probability estimation and ranking also is desirable in many real-world data mining applications. Thus, investigating its performance of class probability estimation and ranking is one of the main direction for future work. Besides, How to set up efficient memory indexing so that the nearest neighbors can be identified more efficiently at some additional cost in memory is our another direction for future work.

References

[1] D. Aha. Lazy learning. Kluwer Academic Publishers, 1997.

[2] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18:509–517, 1975.

[4] H. M. P. B. Frank, E. Locally weighted naive bayes. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 249–256. Morgan Kaufmann, 2003.

[5] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[6] M. J. Greenacre. *Theory and Applications of Correspondence Analysis*. Academic Press, 1984.

[7] K. K. Han. text categorization using weight adjusted k-nearest neighbour classification. Technical report, Dept. of CS, University of Minnesota, 1999.

[8] Z. Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. In *In Proc. SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1997.

[9] L. Jiang, H. Zhang, and Z. Cai. Dynamic k-nearest-neighbor naive bayes with attribute weighted. In *Proceedings of the 3rd International Conference on Fuzzy Systems and Knowledge Discovery*, pages 365–368. Springer, 2006.

[10] L. Jiang, H. Zhang, Z. Cai, and J. Su. Evolutional naive bayes. In *Proceedings of the 1st International Symposium on Intelligent Computation and its Applications*, pages 344–350. China University of Geosciences Press, 2005.

[11] Z. H. S. J. Jiang, L. Instance cloning local naive bayes. In *Proceedings of the Eighteenth Canadian Conference on Artificial Intelligence*, pages 280–291. Springer, 2005.

[12] R. Kohavi. Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 202–207. AAAI Press, 1996.

[13] R. Kohavi and G. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97:273–324, 1997.

[14] P. Langley, W. Iba, and K. Thomas. An analysis of bayesian classifiers. In *Proceedings of the Tenth National Conference of Artificial Intelligence*, pages 223–228. AAAI Press, 1992.

[15] P. Langley and S. Sage. Induction of selective bayesian classifiers. In *Proceedings of Uncertainty in Artificial Intelligence*, pages 400–406. Morgan Kaufmann, 1994.

[16] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[17] C. Stanfill and D. Waltz. Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228, 1986.

[18] Z. Z. G. I. Webb. Lazy learning of bayesian rules. *Machine Learning*, 41:53–84, 2000.

[19] D. R. Wilson and T. R. Martinez. Improved heterogeneous distance functions. *Artificial Intelligence Research*, 6:1–34, 1997.

[20] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2005.

[21] Z. Xie, W. Hsu, Z. Liu, and M. Lee. SNNB: A selective neighborhood based naive bayes for lazy learning. In *Proceedings of the Sixth Pacific-Asia Conference on KDD*, pages 104–114. Springer, 2002.

Table 1. Experimental results for K-Nearest-Neighbor (KNN) versus Weight Adjusted K-Nearest-Neighbor (WAKNN), Dynamic K-Nearest-Neighbor (DKNN), K-Nearest-Neighbor with Distance Weighted (KNNDW) and K-Nearest-Neighbor Naive Bayes (KNNNB): classification accuracy and standard deviation. The value of k in KNN, WAKNN, KNNNB, and KNNDW is 10. v, * : statistically significant improvement or degradation over KNN with a 95% confidence level. The average values and the $w/t/l$ values of two-tailed t-test is summarized at the bottom of the table.

Datasets	KNN	WAKNN	DKNN	KNNDW	KNNNB
anneal	95.88±1.97	96.99±1.67 v	98.44±1.08 v	98.55±1.05 v	97.1±2.05
anneal.ORIG	84.41±3.3	87.64±2.04 v	90.76±2.58 v	89.75±3.10 v	86.64±3.48 v
audiology	58.79±8.3	70.28±7.86 v	75.59±8.69 v	74.72±6.74 v	68.10±6.28 v
autos	62.52±8.03	66.4±8.62	81.86±7.43 v	82.88±6.63 v	76.57±8.47 v
balance-scale	83.84±4.71	78.24±2.56 *	90.88±1.54 v	83.84±4.71	84.16±4.03
breast-cancer	73.09±4.25	70.95±6.06	72.41±7.74	75.22±5.58	74.85±4.09
breast-w	93.99±3.42	95.56±3.13 v	95.56±3.06	94.28±3.5	96.57±2.54 v
colic	83.13±6.29	82.86±6.45	80.98±6.62	84.23±5.09	83.13±5.75
colic.ORIG	69.82±3.4	74.45±4.31	73.90±4.56 v	73.09±5.23 v	73.09±5.54
credit-a	86.09±4.39	85.94±4.53	85.22±4.31	85.51±3.74	86.52±4.21
credit-g	71.9±3.28	75.5±3.66	70.9±3.87	73.7±3.13	74.40±3.50 v
diabetes	69.02±2.19	73.31±4.08	69.01±2.3	69.27±4.12	72.01±4.5
glass	57.06±8.03	58.48±11.1	60.22±9.54	60.82±7.99	64.55±8.76 v
heart-c	81.09±9.77	82.1±9.01	82.09±9.89	81.42±9.55	83.1±7.37
heart-h	82.02±6.06	81.69±6.98	81.69±6.69	81.68±5.66	81.67±4.73
heart-statlog	82.22±7.37	82.22±7.96	81.85±8.27	82.96±6.1	82.59±5.53
hepatitis	84.5±6.22	81.29±3.55	83.88±6.2	83.25±5.35	83.92±6.93
hypothyroid	93.08±0.64	93.21±0.64	92.87±0.87	93.16±0.58	93.21±0.62
ionosphere	89.74±2.78	89.46±2.7	91.44±5.05	89.74±3.37	90.31±3.62
iris	93.33±6.29	92.67±6.63	93.33±6.29	93.33±7.03	94±5.84
kr-vs-kp	95.06±1.34	95.65±0.83	96.50±1.16 v	96.84±0.95 v	96.46±0.78 v
labor	85.67±14.49	91.33±12.1	88.33±13.7	89.67±11.9	91.67±11.8
letter	86.56±0.67	87.07±0.78	89.79±0.57 v	89.68±0.64 v	89.73±0.55 v
lymph	80.86±12.02	82.29±8.93	79.57±11.4	82.86±13.1	85±9.18
mushroom	99.91±0.08	99.9±0.08	100±0	100±0	100±0
primary-tumor	42.47±5.67	44.81±3.59	43.65±2.95	43.03±5.86	46±4.85
segment	89.65±1.84	89.39±2.3	94.59±1.87 v	92.68±1.59 v	91.60±1.54 v
sick	97.03±0.73	97.51±0.57	97.93±0.61 v	97.45±0.60 v	97.56±0.53 v
sonar	81.33±8.42	76.95±11.2	77.86±9.93	80.38±8.44	81.83±9.39
soybean	89.01±2.12	91.5±3.25	92.08±2.72 v	91.94±3.29	91.94±3.06 v
splice	83.26±2.42	90.16±2.14 v	90.19±1.85 v	85.20±2.11 v	90.50±1.25 v
vehicle	68.68±2.74	68.21±4	69.86±3.34	70.8±4.48	70.69±3.01
vote	92.9±3.61	95.64±3.12	92.89±3.11	93.36±3.28	94.95±2.81
vowel	67.68±4.04	70.61±4	94.34±2.24 v	91.62±3.63 v	89.49±3.02 v
waveform-5000	76.36±1.15	78.72±1.37 v	79.06±1.47 v	76.34±1.15	77.2±1.35
zoo	89.18±9.78	89.18±8.57	95.18±6.65	96.09±5.05	95.09±5.18
Mean	81.14±4.77	82.45±4.73	84.30±4.73	84.15±4.56	84.34±4.34
w/t/l	-	6/29/1	14/22/0	11/25/0	13/23/0