An Adaptive *k*-Nearest Neighbor Algorithm

Shiliang Sun Rongqing Huang
Department of Computer Science and Technology,
East China Normal University
500 Dongchuan Road, Shanghai 200241, P.R. China
slsun@cs.ecnu.edu.cn, rqhuang09@gmail.com

***Abstract*—An adaptive *k*-nearest neighbor algorithm (AdaNN) is brought forward in this paper to overcome the limitation of the traditional *k*-nearest neighbor algorithm (*k*NN) which usually identifies the same number of nearest neighbors for each test example. It is known that the value of *k* has crucial influence on the performance of the *k*NN algorithm, and our improved *k*NN algorithm focuses on finding out the suitable *k* for each test example. The proposed algorithm finds out the optimal *k*, the number of the fewest nearest neighbors that every training example can use to get its correct class label. For classifying each test example using the *k*NN algorithm, we set *k* to be the same as the optimal *k* of its nearest neighbor in the training set. The performance of the proposed algorithm is tested on several data sets. Experimental results indicate that our algorithm performs better than the traditional *k*NN algorithm.**

***Keywords*—pattern classification, *k*-nearest neighbor algorithm (*k*NN), adaptive *k*-nearest neighbor algorithm (AdaNN), nearest neighbors**

I. Introduction

Classification aims to automatically place the pre-defined labels on previously unlabeled examples. It is an active research area in information retrieval, machine learning and natural language processing. A number of machine learning algorithms have been introduced to deal with pattern classification, such as *k*-nearest neighbor (*k*NN) [1], [3]–[7] and support vector machine (SVM) [2]. In this paper, we only introduce the traditional *k*-nearest neighbor algorithm (*k*NN) and the proposed adaptive *k*-nearest neighbor algorithm (AdaNN).

The traditional *k*NN usually assumes that the training samples are evenly distributed among different classes. However, unbalanced data sets appear in many practical applications [3]. In an unbalanced data set, the majority class is represented by a large portion of all the examples, while the other, the minority class has only a small percentage of all examples [4]. In fact, *k* is the most important parameter in a classification system based on *k*NN. In the classification process, *k* nearest neighbors of a test example in the training set are identified first. Then, the prediction can be made according to the class labels of these *k* nearest neighbors. Generally speaking, the distribution of examples in every class in the training set is uneven. Some classes may have more examples than others. Therefore, the classification performance is very sensitive to the choice of the parameter *k*. It is very likely that a fixed *k* value would result in a bias on large classes [9].

In order to improve the performance of the traditional *k*NN in practical applications, we propose an improved algorithm the AdaNN in this paper. Different from the traditional *k*NN algorithms, the proposed algorithm identifies different numbers of nearest neighbors for every test example rather than a same number for all. We test the proposed algorithm on 15 datasets. Experimental results show that the proposed algorithm gets better performance than the traditional *k*NN in classification.

The rest of this paper is organized as follows. Section 2 introduces the traditional *k*NN algorithm. Section 3 describes the proposed AdaNN algorithm. Section 4 reports experimental results of 9 traditional *k*NN algorithms from 1NN to 9NN and the proposed algorithm. Finally, Section 5 concludes this paper and gives future research directions.

II. The traditional *k*NN Algorithm

The traditional *k*NN algorithm [5] is one of the oldest and simplest methods for pattern classification. Nevertheless, it often yields competitive results, and in certain domains, when cleverly combined with prior knowledge, it has significantly advanced the state-of-the-art [10], [11]. The *k*NN rule classifies each unlabeled example by the majority label among its *k*-nearest neighbors in the training set. Its performance thus depends crucially on the distance metric used to identify nearest neighbors. In the absence of prior knowledge, most *k*NN classifiers use simple Euclidean metric to measure the dissimilarities between examples represented as vector inputs [15]. Euclidean distance is defined as the following formula.

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} w_r(a_r(x_i) - a_r(x_j))^2} \quad , \tag{1}$$

where we define an example as a vector $x = (a_1, a_2, a_3, \ldots, a_n)$, n is the dimensionality of the vector input, namely, the number of an example's attributes. a_r is the example's *r*th attribute, w_r is the weight of the *r*th attribute, *r* is from *1* to *n*, the smaller $d(x_i, x_j)$ is the two examples are more similar [6].

The class label assigned to a test example is determined by the majority vote of its *k* nearest neighbors.

$$y(d_i) = arg\max_{k} \sum_{x_j \in kNN} y(x_j, c_k) \quad , \tag{2}$$

where d_i is a test example, x_j is one of its *k* nearest neighbors in the training set, $y(x_j, c_k)$ indicates that whether x_j belongs to class c_k. Equation (2) means that the prediction will be the class having most members in the *k* nearest neighbors. For

978-1-4244-5934-6/10/$26.00 ©2010 IEEE

example, if making *5*-nearest neighbor algorithm the classifier, three of an example's 5 nearest neighbors belong to class One and the other two belong to class Two, then we can conclude that the test example belongs to class One. When an example's class label is got only by identifying its nearest neighbor, the algorithm is called the nearest neighbor algorithm (NN) [7].

The traditional *k*NN is well-known and widely used for its simplicity and its easy implementation [8]. *k*NN expects the class conditional probabilities to be locally constant, and suffers from bias in high dimensions [16]. *k*NN is an extremely flexible classification scheme, and does not involve any preprocessing of the training data. This can offer both space and speed advantages in very large problems. However, in many practical applications, it fails to get good results on most of data sets because of the unevenly distribution of the examples among classes. It is unwise to decide all the test examples' class labels by identifying the same number of nearest neighbors, that is, using the same *k*NN algorithm. So an improved *k*NN algorithm should focus on finding out the suitable *k*, the number of its nearest neighbors, for every test example to get its possible class label. To this end, we propose an adaptive *k*NN algorithm (AdaNN) in this paper and we will describe it on the next section in detail.

TABLE I
THE ADAPTIVE *k*-NEAREST NEIGHBOR ALGORITHM

1) 9NN algorithm:
Inputs: the whole training examples
Output: the optimal *k* of each training example
the optimal *k* is the number of the fewest nearest neighbors a training example has to identify to get its correct class label. The value of *k* may be from 1 to 9. If a training example can not get its correct class label by using *1*NN algorithm to *9*NN algorithm, we make 9 its optimal *k*.
Procedure:
a) for each training example, use the Euclidean distance metric to compute the Euclidean distances of it and the rest training examples.
b) sort the Euclidean distances to get the training example's 9 nearest neighbors.
c) get the training example's optimal *k* by checking from the nearest neighbor to 9 nearest neighbors.

2) AdaNN algorithm:
Inputs: the whole training examples and their optimal *k*, the whole test examples
Output: the classification accuracy rate of the AdaNN algorithm
Procedure:
a) for each test example, use the Euclidean distance metric to find out its nearest neighbor in the training set.
b) get the optimal *k* of its nearest neighbor and adopt the corresponding *k*NN algorithm to get its class label for each test example.
c) calculate the number of test examples getting their correct class labels by using the AdaNN algorithm, *num*.
d) compute the classification accuracy rate of the AdaNN algorithm: num/N. *N* is the number of the whole test examples.

III. THE PROPOSED ALGORITHM

A. The Adaptive k-Nearest Neighbor Algorithm

In our experiments, the examples are represented using the vector space model (VSM) [10]. In this model, each example *x* is expressed as a vector. The similarity of two examples is evaluated by the Euclidean distance of the two vectors they represent respectively.

The proposed algorithm is an improved *k*NN algorithm deriving from the traditional *k*NN. Base on the principle that nearest neighbors have similar attributes, we can assume that the test example has the most similar attributes with its nearest neighbor in the training set. The probability is high that a test example adopts the same *k*NN algorithm as its nearest neighbor in the training set to get its correct class label. The optimal *k* is the number of the fewest nearest neighbors a training example has to identify to get its correct class label when assuming it is a test example to the other training examples. Therefore, if we want to get a test example's label, we just need to get the optimal *k* of its nearest neighbor in the training set.

According to the above analysis, we propose in this paper the idea of adaptive *k*-nearest neighbor algorithm (AdaNN) which is shown in TABLE I.

B. The Error Rate of Adaptive k-Nearest Neighbor Algorithm

The error rate of the traditional *k*NN algorithm is proved to be between Bayes and double Bayes. Its accurate expression is shown below:

$$P^* \le P \le P^*(2 - \frac{c}{c-1}P^*) \quad , \tag{3}$$

where P^* is the Bayes error rate and *c* is the number of the classes of the whole data sets. For the number of nearest neighbors, *k*, in the case that the number of the data set N approaches infinity, the larger *k* is, the *k*NN classifier performs better. When $k \to \infty$, the performance of *k*NN classifier is the optimal and the error rate infinitely approaches the Bayes error rate [8]. Therefore, in most cases, the performance of the nearest neighbor algorithm is the worst.

The next is the error rate analysis of the AdaNN algorithm. According to the meaning of the optimal *k*, for most of the examples in the training set, they successfully get their correct class labels by identifying their optimal *k* nearest neighbors. However, as to certain training examples, they can't get their correct class for some reason. As a result, the larger the value of *k* assigned to them as their optimal *k*, the higher the classification accuracy rate is in the training set. Thus, when both the number of the training examples, *M*, and the number of the fewest nearest neighbors, *k*, approach infinity in this manner that $M \to \infty$, $k \to \infty$, the error rate of the *k*NN algorithm for the training set approaches the optimal Bayes error rate. In the best case, the examples of the data set distribute evenly and densely in a small range. Each test example is the same as its nearest neighbor in the training set, the test example uses the same *k*NN algorithm as its nearest neighbor to get its class label. The AdaNN algorithm would perform best in this case and its error rate infinitely approaches the optimal Bayes error rate. In the worst case, all the test examples use the nearest neighbor algorithm to get their class labels, then the AdaNN algorithm degenerates into the nearest neighbor algorithm. In fact, it is impossible that all the test examples use the same kNN algorithm to get their class labels, because the values of the optimal k of each training example are different from each other. Therefore, we can conclude that the AdaNN algorithm can perform better than the traditional kNN algorithm but its error rate is still between the Bayes and double Bayes. The error rate of the AdaNN algorithm can be described as the same as Equation (3).

IV. Experiments

A. Datasets used from UCI

We have tested 9 traditional kNN algorithms, from *1*NN to *9*NN, and the AdaNN algorithm on a number of real world pattern recognition problems. In our experiments, 15 data sets are used, available in the UCI repository website ($http://archive.ics.uci.edu/ml/$). For each data set, 90% of all examples were randomly selected as training examples and the rest 10% as testing ones. The detailed information of the 15 data sets is shown in TABLE II, where the data set name listed in the table is the first word of its full name.

B. The Procedures of Experiments

1) Select Examples: input a data set whose number of examples is N, then randomly select 90% of examples as training examples and the rest 10% as testing ones. For each data set, get 10 training sets and 10 test sets.
2) 9 traditional kNN algorithms: test the 9 kNN algorithms on the 10 test sets got in step 1), k is from *1* to *9*.
3) Get the Optimal k: use the traditional *9*NN for every training sets got in step 1) and output the optimal k for each training example.
4) AdaNN: test the AdaNN on the 10 test sets got in step 1) using the results got in step 3).
5) Compute the average accuracy rate of the 10 test sets using different classification methods, from *1*NN to *9*NN and the AdaNN.
6) Compare the results of different classification methods got in step 5).

C. Experimental Results

The experimental results for 13 data sets are summarized in TABLE III and TABLE IV. Due to lack of space, we only present the average accuracy rates of 10 algorithms on different data sets and ignore the corresponding standard deviations.

We have tested the proposed algorithm on 15 data sets. Comparing to the other 9 traditional kNN algorithms from *1*NN to *9*NN, the AdaNN performs the best on the data sets of Iris, protein, Haberman and Blood. It gets the second best performance on six data sets, the third best performance on one data set and the fourth best performance on two data sets. Experimental results also reveal that the AdaNN gets the worst performance on the last two data sets. It ranks the sixth in the ten. The first column of TABLE III represents 10 different classification methods. The ranks of the proposed algorithm in the ten algorithms are placed in the bottom row of the table. TABLE IV has the same meaning of TABLE III, but it shows the experimental results on the other 7 data sets.

TABLE II
THE TESTED DATA SETS FROM UCI

dataset	classes	attributes	training	test	total
Iris	3	4	140	10	150
Protein	8	7	297	39	336
Haberman	2	3	270	36	306
Blood	2	4	666	82	748
Zoo	7	16	90	11	101
glass	6	9	189	25	214
Pima	2	8	684	84	768
Heart	2	13	243	27	270
Teaching	3	5	135	16	151
Wine	3	13	153	25	178
Balance	3	4	558	67	625
Parkinsons	2	22	171	24	195
Ionosphere	2	34	315	36	351
Contraceptive	3	9	1323	150	1473
Wisconsin	2	30	504	65	569

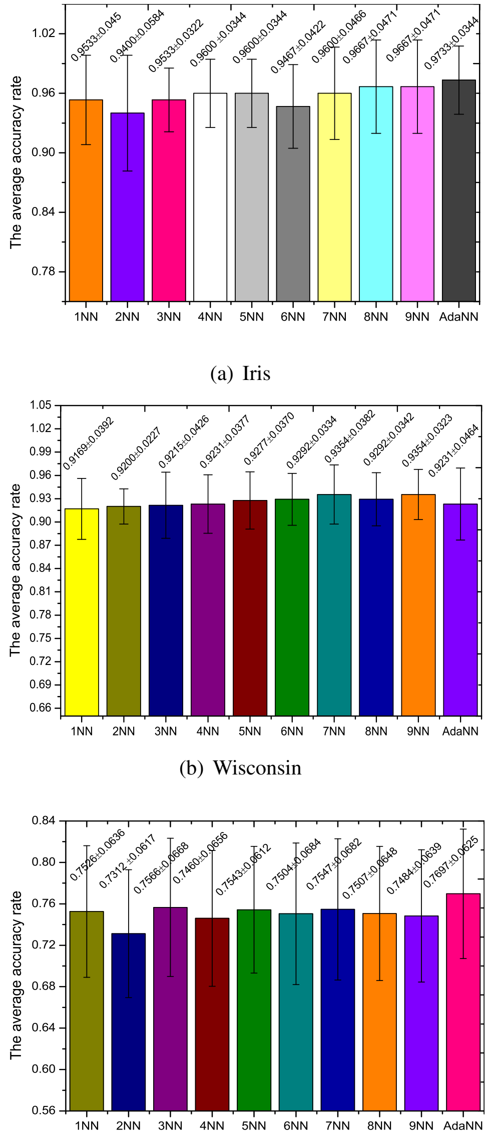
TABLE III
TEST AVERAGE ACCURACY RATES OF TEN ALGORITHMS ON 6 DATA SETS(%)

datasets	Protein	Haberman	Blood	Zoo	Glass	Pima
1NN	81.54	68.06	65.49	97.27	72.40	69.05
2NN	83.08	74.17	62.44	93.64	68.00	63.45
3NN	86.15	71.39	73.90	95.45	67.20	69.64
4NN	85.38	73.33	74.15	91.82	67.60	67.38
5NN	85.90	73.06	74.88	89.09	66.80	72.14
6NN	85.38	72.50	72.68	89.09	64.80	70.12
7NN	86.92	72.22	74.63	84.55	65.20	72.86
8NN	86.15	73.89	75.00	82.73	64.00	72.14
9NN	86.92	73.33	75.85	80.00	62.40	73.57
AdaNN	86.92	75.56	76.46	95.45	68.00	72.86
Rank	1	1	1	2	2	2

TABLE IV
TEST AVERAGE ACCURACY RATES OF TEN ALGORITHMS ON 7 DATA SETS(%)

Heart	Teach	Wine	Bal	Parkin	Iono	Contra
59.63	58.13	74.80	78.51	84.58	86.39	46.00
62.59	44.37	65.60	78.51	86.67	80.83	47.47
65.93	43.75	69.60	82.09	87.08	85.83	49.47
66.30	41.25	65.20	82.09	85.00	81.94	49.27
67.41	41.25	66.40	85.22	85.83	84.44	50.33
65.93	44.37	68.40	87.61	85.00	81.39	50.67
67.41	41.88	69.60	88.51	84.17	82.78	51.73
65.19	43.75	68.80	88.96	81.67	81.67	52.53
63.70	40.00	70.80	88.96	81.67	82.78	52.47
66.30	45.00	72.00	88.06	85.42	83.06	49.80
2	2	2	3	4	4	6

The AdaNN performs the best on Iris and the worst on Wisconsin, comparing to its performance on the 15 data sets. We also compare the average accuracy rate of the ten different algorithms on the total 15 data sets. Fig.1 demonstrates the detailed information of ten algorithms' performance on Iris, Wisconsin and the total 15 data sets. Experimental results presented in TABLE III, TABLE IV and Fig.1 show that



(a) Iris

(b) Wisconsin

(c) Total

Fig. 1. The performance of ten algorithms on Iris, Wisconsin and the total 15 data sets.

AdaNN algorithm can outperform the traditional *k*NN algorithm on most of data sets especially the small scale data sets. According to Fig.1, the proposed algorithm performs just a little worse than the other five *k*NN algorithms in the worst case. Importantly, the proposed algorithm gets a consistently better performance than most of *k*NN algorithms on every data set of the 15 data sets. Therefore, we can conclude that AdaNN algorithm performs better than the traditional *k*NN algorithm in general.

V. Conclusions

In this paper, an adaptive *k*NN algorithm is proposed for classification. Making use of the traditional *k*NN algorithm and the rule that the nearest neighbors have similar attributes, we show the feasibility of this algorithm. Experimental results also show that the proposed algorithm is superior to the traditional *k*NN algorithms in most cases. Possible directions for future work include three aspects. Firstly, we may go on testing the proposed algorithm on more data sets especially the large scale data sets with high dimensionality. Secondly, we should have a more theoretical study of the proposed algorithm's performance on small scale data sets. Thirdly, in this paper we set the optimal k to be the number of the fewest nearest neighbors that every training example can use to get its correct class label. It's well worth studying the performance of the AdaNN algorithm in the case that the optimal k is set to be other values such as the number of the most nearest neighbors that every training example can use to get its correct class label.

Acknowledgment

This work is supported in part by the National Natural Science Foundation of China under Project 60703005, and by Shanghai Educational Development Foundation under Project 2007CG30.

References

[1] Y. Yang, "An evaluation of statistical approaches to text categorization," Information Retrivel 1, 1999, pp. 69-90.

[2] T. Joachims, F. Informatik, and L. Viii, "Text categorization with support vector machines: learning with many relevant features," The 10th European Conference on Machine Learning, Springer, New York, 1998, pp. 137-142.

[3] N. Japkowicz, "Learning from imbalanced data sets: a comparison of various strategies," AAAI workshop on learning from imbalanced data sets, 2000, pp. 10-15.

[4] S. Tan, "Neighbor-weighted *k*-nearest neighbor for unbalanced text corpus," Expert Systems with Applications, vol. 28, 2005, pp. 667-671.

[5] R.O. Duda and P.E. Hart, "Pattern classification and scene analysis," John Wiley & Sons, New York, 1973.

[6] S.A. Dudani, "The distance-weighted *k*-nearest neighbor rule," IEEE Transactions on Systems, Man and Cybernetics, vol. 6, 1976, pp. 325-327.

[7] B. Li, S. Yu and Q. Lu, "An improved k-nearest neighbor algorithm for text categorization," Proceedings of the 20th International Conference on Computer Processing of Oriental Languages, China: Shenyang, 2003.

[8] S.L. Sun, "Ensembles of feature subspaces for object detection," Lecture Notes in Computer Science, vol. 5552, 2009, pp. 996-1004.

[9] Y. Zeng, Y. Yang and L. Zhao, "Pseudo nearest neighbor rule for pattern classification," Expert Systems with Applications, vol. 36, 2009, pp.3587-3595.

[10] S. Belongie, J. Malik and J. Puzicha, "Shape matching and object recognition using shape contexts," IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 24(4), 2002, pp. 509-522.

[11] P.Y. Simard, Y. LeCun and J. Decker, "Efficient pattern recognition using a new transformation distance," In Advances in Neural Information Processing Systems, vol. 6, 1993, pp. 50-58.

[12] N. Japkowicz, "Learning from imbalanced data sets: a comparison of various strategies," AAAI Press, 2000, pp. 10-15.

[13] E. Achtert, H. Kriegel, P. Kröger, M. Renz and A. Zfle, "Reverse k-nearest neighbor search in dynamic and general metric databases," Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology, vol. 360, 2009, pp. 886-897.

[14] T. Hastie and R. Tibshirani, "Discriminant adaptive nearest neighbor classification," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, 1996, pp. 607-616.

[15] K.Q. Weinberger and L.K. Saul, "Distance metric learning for large margin nearest neighbor classification," The Journal of Machine Learning Research, vol. 10, 2009, pp. 207-244.

[16] K. Mouratidis, D. Papadias and M. Hadjieleftheriou, "Conceptual partitioning: an efficient method for continuous nearest neighbor monitoring," Proceedings of the 2005 ACM SIGMOD international conference on Management of data, 2005, pp. 634-645.