2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology

An Effective Evidence Theory based K-nearest Neighbor (KNN) classification

Lei Wang, Latifur Khan and Bhavani Thuraisingham
Department of Computer Science
University of Texas at Dallas
leiwang, lkhan, bhavani.thuraisingham@utdallas.edu

Abstract

In this paper, we study various K nearest neighbor (KNN) algorithms and present a new KNN algorithm based on evidence theory. We introduce global frequency estimation of prior probability (GE) and local frequency estimation of prior probability (LE). A GE for a class is the prior probability of the class across the whole training data space based on frequency estimation; on the other hand, a LE for a class in a particular neighborhood is the prior probability of the class in this neighborhood space based on frequency estimation. By considering the difference between the GE and the LE of each class, we present a solution to the imbalanced data problem in some degree without doing re-sampling. We compare our algorithm with other KNN algorithms using two benchmark datasets. Results show that our KNN algorithm outperforms other KNN algorithms, including basic evidence based KNN.

1. Introduction

Classification is a broad ranging research field which includes many decision-theoretic approaches for identifying data. A datum is typically described numerically via a vector (x_1, x_2, ...x_n) where *n* is the number of attributes/features. Therefore, each piece of data can be treated as one point in an *n* dimensional space, and belongs to one or more classes. Classification algorithms normally employ two steps, training and testing. Characteristic properties of data (or the partition of *n* dimensional space) calculated through the analysis of labeled training data will be applied to classify unlabeled testing data. For example, for image data, classifiers trained through the use of training images will be used for the prediction of unseen images. Obviously, there is a hidden assumption behind classification, i.e. that training data and testing data share the same distribution in the n dimensional space.

Many classification algorithms are available, such as the K nearest neighbor (KNN) algorithm, neural network, decision tree, Bayesian network, and support vector machine (SVM). In general, it is hard to say which classification algorithm is better. We can only say one classification algorithm is better than others for a specific problem. In this paper, we study various KNN algorithms. KNN is a very popular classification algorithm demonstrating good performance characteristics and a short period of training time. However, the shortcomings of KNN are also obvious. First, each neighbor is equally important in the standard KNN. Second, KNN is prone to be affected by the imbalanced data problem. Large classes always have a better chance to win. To solve the first problem, many modified KNN algorithms have been published [2], [3], [4], [6]. Here, we are presenting a new KNN algorithm based on evidence theory. The novelties of our algorithm include two parts. First, according to the distribution of K nearest neighbors, we define a set of neighborhoods which will favor the close neighbors. Second, in order to address the imbalanced data problem, we introduce frequency estimation of *prior probability (GE)* and *local* frequency estimation of *prior probability (LE)*. We define a GE for each class, which is the prior probability of the class across the whole training data space based on frequency estimation. We also define an LE for each class in each neighborhood, which is the prior probability of the class in this neighborhood space based on frequency estimation. By considering the difference between the GE and the LE of each class, we solve imbalanced data problem to some degree without doing re-sampling. We then compare our algorithm with other KNN algorithms for classification accuracy based on two benchmark datasets. Results show that our KNN algorithm outperforms other KNN algorithms.

The contributions of this work can be summarized as follows: first, we extend evidence KNN based on GE and LE. Second, we demonstrate how our proposed method can to a certain extent address the imbalanced data problem without considering re-sampling. Finally, we test our proposed method using various benchmark datasets from various domains, and constantly show that our method outperforms all classical KNNs, including basic evidence based KNN.

978-0-7695-3496-1/08 $25.00 © 2008 IEEE
DOI 10.1109/WIIAT.2008.411

The paper is organized as follows: In section 2, we discuss related work on KNN algorithm and its extensions. In section 3, we introduce evidence-theory-based KNN which is the basis for our algorithm. In Section 4, we present the modified evidence-theory-based KNN. Finally, in section 5, we present experimental results.

2. Related Work

A main drawback of KNN algorithm is that each of the K nearest neighbors is equally important. Intuitively, the closer the neighbor, the more possible that the unknown vector *f* will be in the class of this neighbor. Hence, assigning neighbors with different voting weights based on their distances to the vector *f* is intuitively appealing. Dudani [3] proposes a distance weighted k-nearest neighbor rule. Given the k nearest neighbor v_1, v_2, ..., v_k of the vector f, the d_1, d_2, …, d_k are corresponding distances which are sorted in increasing order. The label of the neighbor v_i will be assigned more voting weight than the label of the neighbor v_j if $d_i < d_j$.

In addition, Keller et al. propose a fuzzy KNN algorithm [4], [8]. Denoeux et al. generate an evidence theoretic KNN [2]. Wang et al. present an extended KNN based on evidence theory [6], which we will discuss in the next section. All these various KNN algorithms modify standard KNN in different ways, but the basic idea is common. They try to improve the performance of KNN algorithm by treating the neighbors of the unknown pattern differently. However, imbalanced data is still a problem. Large classes are always favored in these algorithms. Favoring large classes is not always bad. If training data and testing data share the same distribution, the unknown pattern is more likely to go to large classes than to small classes. The problem is determining when we should favor large classes and when we should not. In this paper, we present an evidence–theory-based KNN algorithm to solve this problem by considering the difference between GE and LE of classes.

3. Background

In this section first, we present Dempster-Shafer evidence theory and next, we present evidence-theory-based KNN. Note that our approach relies on these two.

3.1 Dempster-Shafer evidence theory

Evidence theory was proposed by Shafer in 1976 [5]. Evidence theory represents the degree of belief that may be attributed to a given hypotheses on the basis of given evidence, and combines evidences from different sources using Dempster's rule. Evidence theory is applied to combine outputs of multiple classifiers to generate a more accurate classification procedure. Classifier combination has received more and more attention.

We define Ω as frame of discernment [5], which is a finite set of mutually exclusive and exhaustive hypotheses in a problem domain. The size of power set of Ω is 2^{Ω} which includes the empty set $\emptyset$ and the entire set Ω. In evidence theory, the contribution of evidence to the credibility of different hypotheses is described by a basic probability assignment (BPA) function *m*, the belief function *Bel*, and the plausibility function *Pl*. The BPA function *m* assigns a number between 0 and 1 to each non-empty subset of Ω, and 0 to the empty set $\emptyset$. The sum of BPAs for all subsets *A* of Ω is equal to 1.

The mass *m(A)* measures the quotient of belief that is contributed exactly to *A*. The subsets *A* of Ω are called the focal elements of the belief function, if *m(A)>0*. It is obvious that the degree of belief committed to a hypothesis *A* must be committed to all hypotheses it implies. For example, an animal is a subset of creatures, if the evidence shows that *X* is an animal, this evidence also shows that *X* is a creature. Therefore, to obtain the total belief in hypotheses *A*, we must add BPAs for all subsets *B* of *A*. It is very easy to prove that the summation of the belief of hypotheses *A* and its contradiction $\overline{A}$ is not necessarily equal to 1. So, *Bel(A)* cannot show the summation of our belief in $\overline{A}$. The plausibility of A $Pl(A) = 1 - Bel(\overline{A}) = \sum_{B \cap A \neq empty} m(B)$ defines the degree to which we find *A* plausible.

3.2 Evidence-theory-based KNN (EKNN)

The first evidence theoretic KNN algorithm was published in [2]. In this approach, each neighbor of a pattern is considered as evidence supporting some hypotheses about the class membership of that pattern. The BPAs are calculated for each of the *k* nearest neighbors. The belief of each hypothesis is obtained by aggregating BPAs using Dempster's rule of combination. However, the Dempster Shafer rule is highly complex.

Wang et al. generate an extended KNN based on evidence theory [6]. Instead of combining *k* BPAs in [2], they construct a mass function based on neighborhoods. We will present this work and show how our work differs from them.

As we discussed before, each image is represented by *d* visual features with a vector of *d* attributes/dimensions $<x_1, x_2, ...x_d>$. Each image will belong to one and only one class in the finite set $C = \{c_1, c_2, ...c_M\}$ where *M* is the number of classes.

V is the data space which has *d* dimensions. The labeled training dataset will be specified as:

$$D = \{< s_i, c_j >: s_i \in V, c_j \in C, \quad where \quad i = 1,2,...,N; \quad j = 1,2,...M; \quad V = dom(x_1) \times ... \times dom(x_d)\}$$

Definition 1: Neighborhood is a region in *V* which covers a set of neighbors of an unknown pattern/data point *s*. *V* could have different shapes according to the definition of space and the metric used to calculate the distance for nearest neighbor methods.

We consider V as the frame of discernment Ω. In [6], Wang et al. adopt the hypercube interpretation of neighborhood. Each neighborhood is a hypercube in V which contains s. The metric used to compute distance is important for nearest neighbor algorithms. In this paper, we choose to use Euclidean Distance and therefore a hyper sphere interpretation of a neighborhood rather than a hypercube interpretation. We define h neighborhoods of s: H_1, H_2, …, H_h. Each neighborhood is a hyper sphere in V covering a set of neighbors of s. When we consider k nearest neighbors, the largest neighborhood H_h is the hyper sphere which covers and only covers the k nearest neighbors. Then we divide the radius of the hyper sphere H_h into h equal intervals and define multiple hyper spheres with different radii. Each hyper sphere will be one neighborhood. If we say the radius of the neighborhood H_h is r, then the radius of the neighborhood H_i will be $i \times r/h$. Let us consider the definition of neighborhoods by projecting all neighborhoods (hyper spheres) onto a 2 dimensional space. The origin here represents the unknown pattern *s,* and the number of neighborhoods is 10. The neighborhood H_{10} in this example contains all k nearest neighbors of *s*. Each neighborhood is a source of evidence supporting hypotheses concerning the class membership of the pattern *s*.

Definition 2: Joint probability $P(H_i, c)$ is the probability that a random data point is in the neighborhood H_i ($H_i \in 2^\Omega$) and belongs to class *c* ($c \in C$).

Because data distribution information is not available, we assume data is uniformly distributed in *V* and give a prior estimation of joint probability $P(H_i, c)$ as below [11].

$$P(H_i, c) = \left|H_i^c\right| / |D| \quad (1)$$

$\left|H_i^c\right|$ is the number of data in H_i which belongs to class *c*. $|D|$ is the number of training data.

Definition 3: The mass function m_s induced for *s* from neighborhoods *H* will be defined as below:

$$m_s(A,c) = \begin{cases} \dfrac{P(A,c)}{\sum_{i=1}^{h} \sum_{c \in C} P(H_i, c)} & if \quad A = H_i \\ 0 & otherwise \end{cases} \quad (2)$$

where $A \in 2^\Omega$, and $c \in C$.

New patterns are classified by applying a conditional pignistic probability function .

$$\overline{BetP}(A,c) = \sum_{i=1}^{h} m_s(H_i, c) \times \frac{|A \cap H_i|}{|H_i|} \quad (3)$$

Wang et al. show that $\overline{BetP}$ is a probability function on Ω [11]. Because H_i is the neighborhood of *s*, so $s \in H_i$. If we consider the pattern *s* as a singleton set, we have

$$\overline{BetP}(s,c) = \sum_{i=1}^{h} m_s(H_i, c) / |H_i| \quad (4)$$

$$\overline{BetP}(s) = \sum_{c \in C} \overline{BetP}(s,c) \quad (5)$$

Now, based on Bayes rule, we can calculate conditional probability $\overline{BetP}(c \mid s)$ as below,

$$\overline{BetP}(c \mid s) = \overline{BetP}(s,c) / \overline{BetP}(s) \quad (6)$$

For the pattern *s*, we calculate $\overline{BetP}(c \mid s)$ for all $c \in C$, *s* will be classified as the class having the maximal $\overline{BetP}(c \mid s)$.

4. Density based EKNN (DEKNN)

In this section, we will present our KNN algorithm. As we discussed, the meaning of the joint probability $P(H_i, c)$ in equation (1) is the probability that a random data point is in the neighborhood H_i and belongs to class *c*. By expanding equation (1), we can see that the joint probability $P(H_i, c)$ is the multiplication of two parts (see equation (7)). $|H_i|$ is the number of data in the neighborhood H_i. The first part $\left|H_i^c\right| / |H_i|$ is the capability of the neighborhood H_i for discriminating the class *c*. The second part $|H_i| / |D|$ is the degree of support from the neighborhood H_i. Therefore, we can explain the joint probability $P(H_i, c)$ in another way. $P(H_i, c)$ can be thought as the support which the class *c* obtains from the neighborhood H_i.

$$P(H_i, c) = \left|H_i^c\right| / |D| = \frac{\left|H_i^c\right|}{|H_i|} \times \frac{|H_i|}{|D|} \quad (7)$$

$|H_i^c|/|H_i|$ is the percentage of the class c in the neighborhood H_i. If the neighborhood H_i contains more data points of the class c than data points of any other classes, the class c always gets more support from the neighborhood H_i than other classes. Therefore, large classes are always favored which is also a major problem of the KNN algorithm. To solve this problem, we modify EKNN by changing the probability function $P(H_i, c)$. To describe our algorithm, we need to define two concepts, GE and LE of classes.

Definition 4: GE of a class c, GE^c, is the proportion of the class c in the training dataset.

$$GE^c = |c|/|D| \quad (8)$$

$|c|$ is the number of data in the class c.

Definition 5: LE of a class c in the neighborhood H_i, LE_i^c, is the proportion of the class c in the neighborhood H_i.

$$LE_i^c = |H_i^c|/|H_i| \quad (9)$$

In EKNN, the capability of the neighborhood H_i for discriminating the class c is totally determined by LE_i^c, the LE of the class c in the neighborhood H_i. Without data distribution information, large classes usually have larger LE than small classes, explaining why EKNN favors large classes. To address this problem, a highly dense neighborhood will get a higher weight as compared to lightly dense neighborhoods. In other words, intuitively, in a particular neighborhood, if the LE of a class is larger than its GE, this class will get more support from this neighborhood. If the LE of the class is less than its GE, the class will get less support from this neighborhood. We would like to exploit the notion of LE and GE. We formalize the idea and modify the equation (7) as below.

$$P(H_i, c) = (w1 \times LE_i^c + w2 \times \frac{LE_i^c - GE^c}{LE_i^c}) \times \frac{|H_i|}{|D|} \quad (10)$$

$$P(H_i, c) = LE_i^c \times \frac{|H_i|}{|D|} \quad (11)$$

In Equation (10), $w1$ and $w2$ are weights and $w1 + w2 = 1$. In our experiment, we assign equal weight 0.5 to $w1$ and $w2$. The construction of mass function in our algorithm is same as EKNN in equation (2).

In DEKNN Equation (10) is similar to Equation (11) in EKNN except for second additional terms. In both DEKNN and EKNN, the classes having larger LE will get more support. This is reasonable in most cases when we don't have information about data distribution. The difference is that the support in DEKNN is not totally determined by the LE of the classes. DEKNN depends on the difference between LE and GE (second term in equation (10)). When the LEis larger than the GE ($LE_i^c > GE^c$), the second part in equation (10) will be positive and the class c will get more support from the neighborhood H_i *(i.e., reward model)*. If the LE is less than GE ($LE_i^c < GE^c$), the second part in equation (10) will be negative and the class c will get less support from the neighborhood H_i *(punish model)*. If the LE is equal to GE ($LE_i^c = GE^c$), the second part in equation (10) will be zero and the equation (10) will be similar to the equation (7). Recall that EKNN only considers LE_i^c for $P(H_i, c)$ by ignoring other factors (GE). Therefore, we can notice that for $P(H_i, c)$ calculation DEKNN is aggressive as compared to EKNN in terms of punishing and rewarding of a class.

4.1 Imbalanced Dataset

We know that the imbalanced data problem is a serious problem for most classification algorithms. Normally, a re-sampling method will be applied to handle imbalanced data. Compared with the re-sampling method, the DEKNN algorithm will solve the data imbalance problem more effectively. This is because re-sampling is expensive to generate new data points. Furthermore, DEKNN does not always favor small classes like re-sampling methods do. If the LE of a small class in a specific neighborhood is smaller than its GE, the small class will get even less support. At the same time, DEKNN does not increase computational load because it does not require re-sampling. Experimental results show that DEKNN has better classification accuracy than EKNN.

In general, without knowing data distribution, it is more likely that larger classes have higher LE as compared to small classes for a certain neighborhood. Hence, in some cases according to Equation (11), EKNN may still favor large classes for prediction even when the unknown pattern belongs to the small class. With DEKNN, $P(H_i, c)$ calculation will be modified by the difference between LE and GE. Intuitively, if a small class's LE is greater than its GE in a certain neighborhood, this neighborhood should provide more support to this small class even if the value of this small class's LE is small. Considering this, DEKNN will modify $P(H_i, c)$ value for this small class accordingly by positive difference between LE and GE for this small class. In addition, a large class may have

a higher LE for a certain neighborhood than a small class. But if the GE of this large class is even higher than its LE, it will still be punished and get less support. However, if a large class's LE is greater than its GE for a certain neighborhood, this positive difference will award $P(H_i, c)$ value to give this large class more support. Hence, DEKNN adds some values for $P(H_i, c)$ in such a way that a small class will not be penalized and a large class will not be favored always.

Now, question will arise whether DEKNN outperforms EKNN in both large and small classes or only in large classes (i.e., not in small classes) or only in small classes (i.e., not in large class). We observe in experimental results that DEKNN outperforms EKNN for small classes with the same margin as for large classes. This demonstrates that DEKNN is less sensitive to small classes (i.e., solves imbalance dataset problem at some extent).

5. Experiment Results

The dataset we applied is from the forest covertype dataset [7]. There are 58,377 instances (observations) in the dataset which belong to 7 different cover types. 54 features of data include 4 binary wilderness areas, 40 binary soil type variables and 10 quantitative variable such as elevation, hill shade at different time of a day etc. We randomly select 58,377 (around 10%) instances as our dataset. Here, Spruce-Fir and Lodgepole Pine are two large classes and Cottonwood/Willow is a small class with only 54 instances.

As we discussed, DEKNN solves the data imbalance problem more effectively (see Section 4.1). In Table 1, we compare the average classification accuracy of EKNN and DEKNN for large classes, small classes in the Forest CoverType dataset. Results are shown in Table 1. Note that for small class, DEKNN outperforms EKNN with larger gap as compared to larger classes. For example, for Cottonwood/Willow class, DEKNN (accuracy 66.67%) outperforms EKNN (42.59%) with a large margin of 24.08. On the other hand, for Spruce-Fir large class, DEKNN (accuracy 91.30%) outperforms EKNN (83.85%) with a margin of 7.5. For Lodgepole Pine class, accuracy of DEKNN is slightly lower than that of EKNN. This demonstrates that DEKNN still gives better accuracy for small and large classes (some extent) —handles imbalance data problem. In other words, DEKNN is more sensitive to small classes positively as compared to EKNN.

References

[1] P.Bennett,S.Dumais and E.Horvitz,"Probabilistic combination of text classifiers using reliability indicators models and results", In *Proc. 25th Ann. Int. ACM Conf. on Research and Development in Information Retrieval(SIGIR02),* Tampere, Finland, pp. 207-214, ACM Press, New York, 2002.

[2] T.Denoeux, "A k-nearest neighbor classification rule based on Dempster-Shafer theory", *IEEE Transaction on Systems, Man and Cybernetics:* 25, pp. 804-813, 1995.

[3] S. A. Dudani, "The distance-weighted k nearest neighbor rule", *IEEE Trans. Syst Man Cyber.,*vol.6,pp325-327, 1976.

[4] J.M.Keller, M R.Gray, and J.A.Givens, "A fuzzy k-nearest neighbor algorithm", *IEEE Trans. Syst. Man Cyber*., vol. 15, no. 4, pp. 580-585, 1985.

[5] G.Shafer,"A Mathematical Theory of Evidence", Princeton, NJ: Princeton University Press, 1976.

[6] H. Wang and D. Bell, "Extended K-Nearest Neighbors based on Evidence Theory", *Computer Journal 47*(6): pp. 662-672, 2004.

[7] http://kdd.ics.uci.edu/databases/covertype/covertype.html

[8] H.B.Mitchell,P.A. Schaefer, "A soft K-nearest neighbor voting scheme,"*International Journal of Intelligent Systems,* Vol. 16, No. 4, 2001. Page 459-468, John Wiley & Sons Inc.

Table 1.Classification accuracies

	# of instance	Accuracies using EKNN(%)	Accuracies using DEKNN(%)
Spruce-Fir	4216	83.85	91.30
Lodgepole Pine	5602	91.68	89.18
Ponderosa Pine	686	86.30	91.25
Cottonwood/Willow	**54**	**42.59**	**66.67**
Aspen	**176**	**23.86**	**77.27**
Douglas-Fir	**352**	**54.26**	**80.97**
Krummholz	**396**	**61.87**	**90.91**