**Abstract**

K is the most important parameter in a text categorization system based on k-Nearest Neighbor algorithm (kNN).In the classification process, k nearest documents to the test one in the training set are determined firstly. Then, the predication can be made according to the category distribution among these k nearest neighbors. Generally speaking, the class distribution in the training set is uneven. Some classes may have more samples than others. Therefore, the system performance is very sensitive to the choice of the parameter k. And it is very likely that a fixed k value will result in a bias on large categories. To deal with these problems, we propose an improved kNN algorithm, which uses different numbers of nearest neighbors for different categories, rather than a fixed number across all categories. More samples (nearest neighbors) will be used for deciding whether a test document should be classified to a category, which has more samples in the training set. Preliminary experiments on Chinese text categorization show that our method is less sensitive to the parameter k than the traditional one, and it can properly classify documents belonging to smaller classes with a large k. The method is promising for some cases, where estimating the parameter k via cross-validation is not allowed

**1.Introduction**

K Nearest Neighbour (KNN from now on) is one of those algorithms that are very simple to understand but works incredibly well in practice. Also, it is surprisingly versatile and its applications range from vision to proteins to computational geometry to graphs and so on. Most people learn the algorithm and do not use it much which is a pity as a clever use of KNN can make things very simple. It also might surprise many to know that KNN is one of the [**top 10 data mining algorithms**](http://www.cs.umd.edu/~samir/498/10Algorithms-08.pdf). Let’s see why this is the case!

In this post, I will talk about KNN and how to apply it in various scenarios. I will focus primarily on classification even though it can also be used in regression). I also will not discuss much about [**Voronoi diagram**](http://en.wikipedia.org/wiki/Voronoi_diagram)or[**tessellation**](http://en.wikipedia.org/wiki/Tessellation)**.**

KNN is an non parametric lazy learning algorithm. That is a pretty concise statement. When you say a technique is non parametric, it means that it does not make any assumptions on the underlying data distribution. This is pretty useful, as in the real world, most of the practical data does not obey the typical theoretical assumptions made (e.g. gaussian mixtures, linearly separable etc). Non parametric algorithms like KNN come to the rescue here.

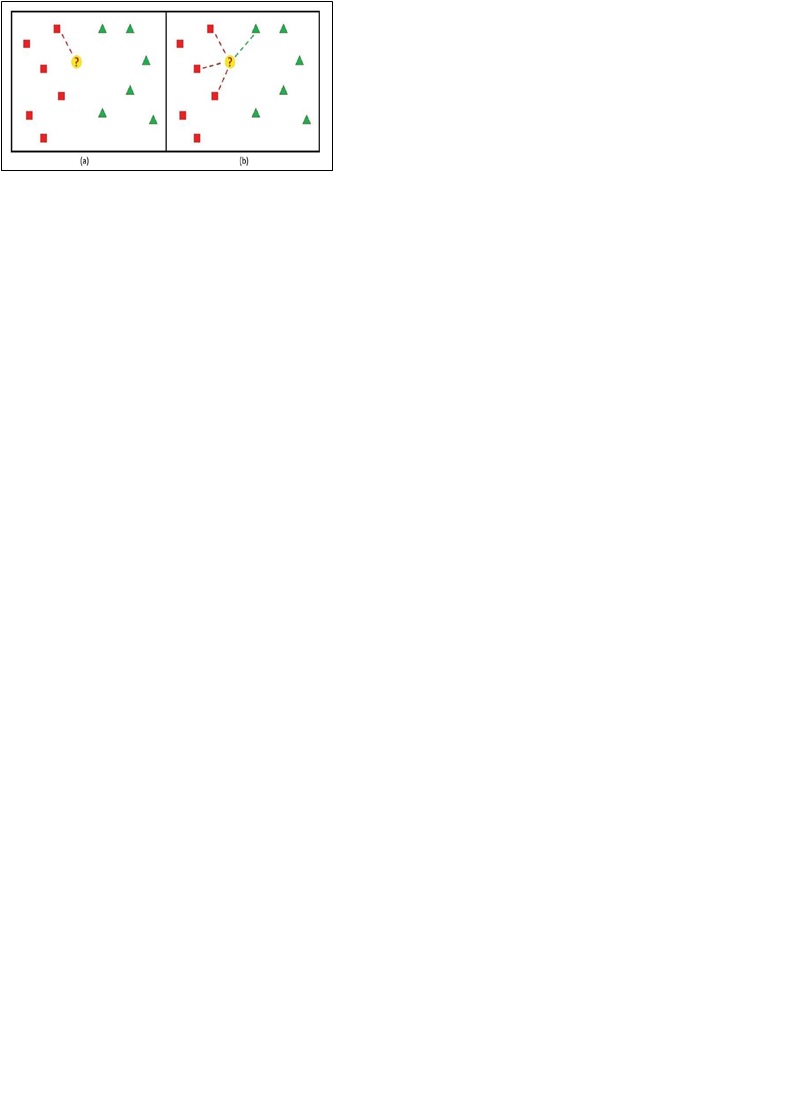
It is also a lazy algorithm. What this means is that it does not use the training data points to do any generalization*.* In other words, there is no explicit training phase or it is very minimal. This means the training phase is pretty fast. Lack of generalization means that KNN keeps all the training data. More exactly, all the training data is needed during the testing phase. (Well this is an exaggeration, but not far from truth). This is in contrast to other techniques like SVM where you can discard all non-support vectors without any problem.  Most of the lazy algorithms – especially KNN – makes decision based on the entire training data set (in the best case a subset of them).

The dichotomy is pretty obvious here – There is a non-existent or minimal training phase but a costly testing phase. The cost is in terms of both time and memory. More time might be needed as in the worst case, all data points might take point in decision. More memory is needed as we need to store all training data.

**2. Theoretical Background**

**2.1 KNN for classification**

In pattern recognition, the KNN algorithm is a method for classifying objects based on closest training examples in the feature space. KNN is a type of instance-based learning, or lazy learning where the function is only approximated locally and all computation is deferred until classification [9-12]. The KNN is the fundamental and simplest classification technique when there is little or no prior knowledge about the distribution of the data [12-15]. This rule simply retains the entire training set during learning and assigns to each query a class represented by the majority label of its k-nearest neighbours in the training set. The Nearest Neighbour rule (NN) is the simplest form of KNN when K = 1. In this method each sample should be classified similarly to its surrounding samples. Therefore, if the classification of a sample is unknown, then it could be predicted by considering the classification of its nearest neighbour samples. Given an unknown sample and a training set, all the distances between the unknown sample and all the samples in the training set can be computed. The distance with the smallest value corresponds to the sample in the training set closest to the unknown sample. Therefore, the unknown sample may be classified based on the classification of this nearest neighbour [15-20]. Figure 1 shows the KNN decision rule for K= 1 and K= 4 for a set of samples divided into 2 classes. In Figure 1(a), an unknown sample is classified by using only one known sample; in Figure 1(b) more than one known sample is used. In the last case, the parameter K is set to 4, so that the closest four samples are considered for classifying the unknown one. Three of them belong to the same class, whereas only one belongs to the other class. In both cases, the unknown sample is classified as belonging to the class on the left. Figure 2 provides a sketch of the KNN algorithm.



**Figure 1.** (a) The 1-NN decision rule: the point?is assigned to the class on the left; (b) the KNN decision rule, with K= 4: the point?is assigned to the class on the left as well .

The performance of a KNN classifier is primarily determined by the choice of K as well as the distance metric applied [20-25]. The estimate is affected by the sensitivity of the selection of the neighbourhood size K, because the radius of the local region is determined by the distance of the Kth nearest neighbour to the query and different K yields different conditional class probabilities. If K is very small, the local estimate tends to be very poor owing to the data sparseness and the noisy, ambiguous or mislabeled points. In order to further smooth the estimate, we can increase K and take into account a large region around the query. Unfortunately, a large value of K easily makes the estimate over smoothing and the classification performance degrades with the introduction of the outliers from other classes. To deal with the problem, the related research works have been done to improve the classification performance ofKNN.

How to select a suitable neighbourhood size K is a key issue that largely affects the classification performance of KNN. As for KNN, the small training sample size can greatly affect the selection of the optimal neighbourhood size K and the degradation of the classification performance of KNN is easily produced by the sensitivity of the selection of K. Generally speaking, the classification results are very sensitive to two aspects: the data sparseness and the noisy, ambiguous or mislabeled points if K is too small, and many outliers within the neighbourhood from other classes if K is too large. From a theoretical point of view, the classification performance of KNN is determined by the estimate of the conditional class probabilities of the query in a local region of the data space, which is determined by the distance of the Kth nearest neighbour to the query. So, the classification performance is very sensitive to the selected value of K. Furthermore, the simplest majority voting of combining the class labels for KNN can be a problem if the nearest neighbours vary widely over their distances and the closer ones more reliably indicate the class of the query object. With the goal of addressing the sensitivity issue of different choices of the neighbourhood size K,some weighted voting methods have been developed for KNN.

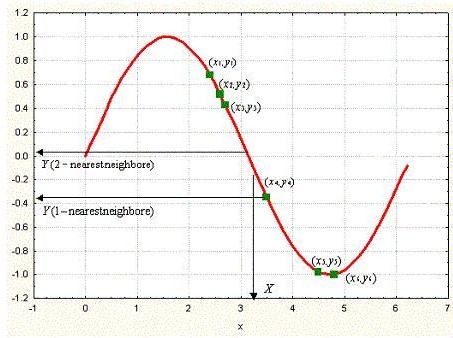
It has been shown in when the points are not uniformly distributed; predetermining the value of K becomes difficult. Generally, larger values of K are more immune to the noise presented, and make boundaries smoother between classes. As a result, choosing the same (optimal) K becomes almost impossible for different applications.

Regression problems are concerned with predicting the outcome of a dependent variable given a set of independent variables. To start with, we consider the schematic shown above in Figure 3, where a set of points (green squares) are drawn from the relationship between the independent variable x and the dependent variable y (red curve). Given the set of green objects (known as examples) we use the KNN method to predict the outcome of X (also known as query point) given the example set (green squares). To begin with, let's consider the 1-nearest neighbour method as an example. In this case we search the example set (green squares) and locate the one closest to the query point X. For this particular case, this happens to be x4. The outcome of x4 (i.e., y4) is thus then taken to be the answer for the outcome of X (i.e., Y). Thus for 1-nearest neighbour we can write: Y = y4 for the next step, let's consider the 2-nearest neighbour method. In this case, we locate the first two closest points to X, which happen to be y3 and y4. Taking the average of their outcome, the solution for Y is then given by: 𝑌 = 𝑦3 + 𝑦4 2 The above discussion can be extended to an arbitrary number of nearest neighbours K. To summarize, in a KNN method, the outcome Y of the query point X is taken to be the average of the outcomes of its K nearest neighbours.

**2.2 KNN for Regression**

**2.2.1 Theory**

The same method can be used for regression, by simply assigning the property value for the object to be the average of the values of its K nearest neighbours. It can be useful to weight the contributions of the neighbours, so that the nearer neighbours contribute more to the average than the more distant ones



**Figure 3.** the KNN decision rule for regression

**3. The KNN Algorithm**

1. Load the data
2. Initialize K to your chosen number of neighbours
3. For each example in the data
   1. Calculate the distance between the query example and the current example from the data.
   2. Add the distance and the index of the example to an ordered collection
4. Sort the ordered collection of distances and indices from smallest to largest (in ascending order) by the distances
5. Pick the first K entries from the sorted collection
6. Get the labels of the selected K entries
7. If regression, return the mean of the K labels
8. If classification, return the mode of the K labels

## **4.Choosing the right value for K**

To select the K that’s right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm’s ability to accurately make predictions when it’s given data it hasn’t seen before.

Here are some things to keep in mind:

1. As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (I’m thinking about the top left corner of the colored plot above), but the green is the single nearest neighbour. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.
2. Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.
3. In cases where we are taking a majority vote (e.g. picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

### 5.KNN for Classification

Let’s see how to use KNN for classification. In this case, we are given some data points for training and also a new unlabelled data for testing. Our aim is to find the class label for the new point. The algorithm has different behaviour based on k.

**5.1.1 Case 1: k = 1 or Nearest Neighbour Rule**

This is the simplest scenario. Let x be the point to be labelled. Find the point closest to x. Let it be y. Now nearest neighbour rule asks to assign the label of y to x. This seems too simplistic and sometimes even counter intuitive. If you feel that this procedure will result a huge error, you are right – but there is a catch. This reasoning holds only when the number of data points is not very large.

If the number of data points is very large, then there is a very high chance that label of x and y are same. An example might help – Let’s say you have a (potentially) biased coin. You toss it for 1 million time and you have got head 900,000 times. Then most likely your next call will be head. We can use a similar argument here.

Let me try an informal argument here - Assume all points are in a D dimensional plane. The number of points is reasonably large. This means that the density of the plane at any point is fairly high. In other words, within any subspace there is adequate number of points. Consider a point x in the subspace which also has a lot of neighbours. Now let y be the nearest neighbour. If x and y are sufficiently close, then we can assume that probability that x and y belong to same class is fairly same – Then by decision theory, x and y have the same class.

The book "Pattern Classification" by Duda and Hart has an excellent discussion about this Nearest Neighbour rule. One of their striking results is to obtain a fairly tight error bound to the Nearest Neighbour rule. The bound is

P^* \leq P \leq P^* ( 2 - \frac{c}{c-1} P^*)

Where P^* is the Bayes error rate, c is the number of classes and P is the error rate of Nearest Neighbour. The result is indeed very striking (atleast to me) because it says that if the number of points is fairly large then the error rate of Nearest Neighbour is less than twice the Bayes error rate. Pretty cool for a simple algorithm like KNN. Do read the book for all the juicy details.

### 5.1.2 Case 2 : k = K or k-Nearest Neighbour Rule

This is a straightforward extension of 1NN. Basically, what we do is that we try to find the k nearest neighbour and do a majority voting. Typically, k is odd when the number of classes is 2. Let’s say k = 5 and there are 3 instances of C1 and 2 instances of C2. In this case, KNN says that new point has to labelled as C1 as it forms the majority. We follow a similar argument when there are multiple classes.

One of the straight forward extension is not to give 1 vote to all the neighbours. A very common thing to do is weighted kNN where each point has a weight which is typically calculated using its distance. For eg under inverse distance weighting, each point has a weight equal to the inverse of its distance to the point to be classified. This means that neighbouring points have a higher vote than the farther points.

It is quite obvious that the accuracy \*might\* increase when you increase k but the computation cost also increases.

**6.Advantages and Disadvantages**

**6.1 Advantages**

KNN has several main advantages: simplicity, effectiveness, intuitiveness and competitive classification performance in many domains. It is Robust to noisy training data and is effective if the training data is large.

**6.2 Disadvantages**

Despite the advantages given above, KNN has a few limitations. KNN can have poor run-time performance when the training set is large. It is very sensitive to irrelevant or redundant features because all features contribute to the similarity and thus to the classification. By careful feature selection or feature weighting, this can be avoided. Two other disadvantages of the method are:

• Distance based learning is not clear which type of distance to use and which attribute to use to produce the best results.

• Computation cost is quite high because we need to compute distance of each query instance to all training samples.

**7.Applications of KNN**

1. **Nearest Neighbour based Content Retrieval**

This is one the fascinating applications of KNN. Basically, we can use it in Computer Vision for many cases- You can consider handwriting detection as a rudimentary nearest neighbour problem. The problem becomes more fascinating if the content is a video – given a video find the video closest to the query from the database Although this looks abstract, it has lot of practical applications. Eg: Consider [**ASL**](http://en.wikipedia.org/wiki/Asl) (American Sign Language). Here the communication is done using hand gestures.

1. **Gene Expression**

This is another cool area where many a time, KNN performs better than other state of the art techniques. In fact, a combination of KNN-SVM is one of the most popular techniques there. This is a huge topic on its own and hence I will refrain from talking much more about it.

1. **Text mining**

The KNN algorithm is one of the most popular algorithms for text categorization or text mining. Some of the most recent works on this topic are for instance. Different numbers of nearest neighbours are used for different classes in this approach, rather than a fixed number across all classes. In this way, the only parameter that needs to be chosen by the user when using KNN, the K value, becomes less sensible and hence it does not need to be carefully chosen as in the standard algorithm. Indeed, the probability that an unknown sample belongs to a class is computed by using only some top Kn nearest neighbours for that class. The Kn value is derived from K according to the size of the corresponding class in the training set. This modified KNN was efficient and less sensible to the K values when applied to text mining problems.

1. **Agriculture**

In general, KNN is applied less than other data mining techniques in agriculture related fields. It has been applied, for instance, for simulating daily precipitations and other weather variables. Another interesting application is the evaluation of forest inventories and for estimating forest variables. In these applications, satellite imagery is used, with the aim of mapping the land cover and land use with few discrete classes. The other applications of the k-NN method in agriculture include climate forecasting and estimating soil water parameters.

1. **Finance**

Data mining as a process of discovering useful patterns and correlations has its own niche in financial modelling. Similar to other computational methods almost every data mining method and technique has been used in financial modelling. An incomplete list includes a variety of linear and nonlinear models multi-layer neural networks, k-means and hierarchical clustering, k-nearest neighbours, decision tree analysis, regression (logistic regression, general multiple regression), ARIMA, principal component analysis, and Bayesian learning. Stock market forecasting is one of the most core financial tasks of KNN. Stock market forecasting includes uncovering market trends, planning investment strategies, identifying the best time to purchase the stocks, and what stocks to purchase

1. **Medicine**

Predict whether a patient, hospitalized due to a heart attack, will have a second heart attack. The prediction is to be based on demographic, diet and clinical measurements for that patient. • Estimate the amount of glucose in the blood of a diabetic person, from the infrared absorption spectrum of that person’s blood. • Identify the risk factors for prostate cancer, based on clinical and demographic variables

Some of other applications of KNN in finance are mentioned below:

* Forecasting stock market: Predict the price of a stock, on the basis of company performance measures and economic data.
* Currency exchange rate
* Bank bankruptcies
* Understanding and managing financial risk
* Trading futures
* Credit rating
* Loan management
* Bank customer Mr.iling
* Money laundering analyses