

Nearest Neighbor Classification

Charles Elkan

elkan@cs.ucsd.edu

September 29, 2008

The nearest-neighbor method is perhaps the simplest of all algorithms for predicting the class of a test example. The training phase is trivial: simply store every training example, with its label. To make a prediction for a test example, first compute its distance to every training example. Then, keep the k closest training examples, where $k \geq 1$ is a fixed integer. Look for the label that is most common among these examples. This label is the prediction for this test example.

This basic method is called the k NN algorithm. There are two design choices to make: the value of k , and the distance function to use. When there are two alternative classes, the most common choice for k is a small odd integer, for example $k = 3$. When each example is a fixed-length vector of real numbers, the most common distance function is Euclidean distance:

$$d(x, y) = \|x - y\| = \sqrt{(x - y) \cdot (x - y)} = \left(\sum_i (x_i - y_i)^2 \right)^{1/2}.$$

An obvious disadvantage of the k NN method is the time complexity of making predictions. Suppose there are n training examples in d dimensions. Then applying the method to one test example requires $O(nd)$ time, compared to just $O(d)$ time to apply a linear classifier such as a perceptron. If the training data are stored in a sophisticated data structure, for example a kd -tree, then finding nearest neighbors can be done much faster if the dimensionality d is small. However, for dimensionality $d \geq 20$ about, no data structure is known that is useful in practice [KF07].

If the distance function satisfies the triangle inequality, then this inequality can be used to reduce the number of distance calculations needed. The fundamental fact that is useful is the following.

Lemma: Suppose $d(\cdot, \cdot)$ is a distance metric that satisfies the triangle inequality $d(x, z) \leq d(x, y) + d(y, z)$. Then $d(u, w) \geq |d(u, v) - d(v, w)|$.

The LAESA algorithm uses this lemma as follows. Given a training set T and a test point x , the algorithm returns the nearest neighbor y in T of x . The algorithm uses a fixed set of basis points B whose distances to every point in T are computed in advance:

```

compute  $d(b, x)$  for all  $b \in B$ 
let  $y = \operatorname{argmin}_b d(b, x)$ 
for all  $t \in T$ 
    compute  $g(t, x) = \max_b |d(t, b) - d(b, x)|$ 
// invariant:  $d(t, x) \geq g(t, x)$  for  $t \in T$ 
for all  $r$  in  $T$  sorted by increasing  $g(r, x)$ 
    if  $g(r, x) \geq d$  then return  $y$ 
compute  $d(r, x)$ 
if  $d(r, x) < d(y, x)$  then let  $y = r$ 

```

A major advantage of the k NN method is that it can be used to predict labels of any type. Suppose training and test examples belong to some set X , and labels belong to some set Y . Formally, a classifier is a function $X \rightarrow Y$. Supervised learning problems are categorized as follows.

- The simplest case is $|Y| = 2$; this is a binary classification problem. The standard perceptron is only useful for this type of problem.
- If Y is finite and discrete, with $|Y| \geq 3$, this is a multiclass classification problem.
- If $Y = \mathbb{R}$ we have a regression task.
- If $Y = \mathbb{R}^q$ with $q \geq 1$ we have a multidimensional regression task.
- If Y is a powerset, that is $Y = 2^Z$ for some finite discrete set Z , we have a multilabel problem. For example Y might be the set of all newspaper articles and Z might be the set of labels {SPORTS, POLITICS, BUSINESS, ...}.
- If $X = A^*$ and $Y = B^*$ where A and B are sets of symbols, then we have a sequence labeling problem. For example, A^* might be the set of all English sentences, and B^* might be the set of part-of-speech sequences such as NOUN VERB ADJECTIVE.

The nearest neighbor method is the only algorithm that is usable for all these types of problem. Of course, we still need a useful distance function $d : X \times X \rightarrow \mathbb{R}$, which may not be obvious to design for some sets X . Note that a k NN method with $k \geq 2$ requires some sort of averaging or voting function for combining the labels of multiple training examples.

The nearest-neighbor method suffers severely from what is called the “curse of dimensionality.” This curse has multiple aspects. Computationally, as mentioned above, if $d \geq 20$ the method is very slow. More subtly, the accuracy of the method tends to deteriorate as d increases. The reason is that in a high-dimensional space all points tend to be far away from each other, so nearest neighbors are not meaningfully similar. Practically, if examples are represented using many features, then every pair of examples will likely disagree on many features, so it will be rather arbitrary which examples are closest to each other.

In many domains, data are represented as points in a high-dimensional space, but for domain-specific reasons almost all points are located close to a low-dimensional subspace. In this case the “effective dimensionality” of the data is said to be small, and nearest-neighbor methods may work well.

A useful way to judge the impact of the curse of dimensionality is to plot the distribution of interpair distances in the training set. If n is large it is enough to take a random sample of all $n(n-1)/2$ pairs. It is desirable for the distribution to have large spread compared to its mean. If it does, then the effective dimensionality of the data is small.

For examples that are real-valued vectors using a p -norm distance with $p < 2$ instead of Euclidean distance can be beneficial [AHK01]. This distance function is

$$d_p(x, y) = \|x - y\|_p = \left(\sum_i |x_i - y_i|^p \right)^{1/p}.$$

Note that the definition includes taking the absolute value of $x_i - y_i$ before raising it to the power p . The case $p = 1$ is called Manhattan distance: $d_1(x, y) = \sum_i |x_i - y_i|$. The case $p = 0$ is called Hamming distance: $d_0(x, y) = \sum_i I(x_i \neq y_i)$ where $I(\alpha)$ is an indicator function: $I(\alpha) = 1$ if α is true and $I(\alpha) = 0$ otherwise.

The nearest-neighbor method has an important theoretical property. First we need a definition. The Bayes error rate for a classification problem is the minimum achievable error rate, i.e. the error rate of the best possible classifier. This error rate will be nonzero if the classes overlap. For example, suppose that $x \in \mathbb{R}$ and x given y follows a Gaussian distribution with mean μ_y and fixed variance. The two Gaussians overlap so no classifier can predict y correctly for all x , and the

Bayes error rate is nonzero. (Notice that the Bayes error rate has no connection with Bayes' rule.)

The Bayes error rate is the average over the space of all examples of the minimum error probability for each example. The optimal prediction for any example x is the label that has highest probability given x . The error probability for this example is then one minus the probability of this label. Formally, the Bayes error rate is

$$E^* = \int_x p(x)[1 - \max_i p(i|x)]$$

where the maximum is over the m possible labels $i = 1$ to $i = m$.

The theoretical property of the 1NN method is that in the limit where the number of training examples tends to infinity, the error rate of a 1NN classifier is at worst twice the Bayes error rate. This result was proved by Cover and Hart in 1967. For a sketch of the proof see [Rip96, page 192].

Theorem: For sufficiently large training set size n , the error rate of the 1NN classifier is less than twice the Bayes error rate.

Proof: The critical fact is that if the number n of training examples is large enough, then the label probability distributions for any point and its nearest neighbor will be essentially the same. In this case, for any point x , the expected error rate of the 1NN classifier is

$$\sum_{i=1}^m p(i|x)[1 - p(i|x)].$$

To prove the theorem we need to show that

$$\sum_{i=1}^m p(i|x)[1 - p(i|x)] \leq 2[1 - \max_i p(i|x)].$$

Let $\max_i p(i|x) = r$ and let this maximum be attained with $i = j$. Then the lefthand side is

$$r(1 - r) + \sum_{i \neq j} p(i|x)[1 - p(i|x)]$$

and the righthand side is $2(1 - r)$. The summation above is maximized when all values $p(i|x)$ are equal for $i \neq j$. The value of the lefthand side is then

$$\begin{aligned} A &= r(1 - r) + (m - 1) \frac{1 - r}{m - 1} \frac{m - 1 - (1 - r)}{m - 1} \\ &= r(1 - r) + (1 - r) \frac{m + r - 2}{m - 1}. \end{aligned}$$

Now $r \leq 1$ and $m - 2 + r < m - 1$ so $A < 2(1 - r)$ which is what we wanted to prove. ■

An alternative version of this result is the statement $E^* \geq E/2$. This says that with a large enough training set, no classifier can do better than half the error rate of a 1NN classifier. Conversely, we can obtain an estimated lower bound for the Bayes error rate by measuring the error rate of a 1NN classifier, then dividing by two.

Can a nearest-neighbor method actually come close to the Bayes error rate? The answer is yes. If $k \rightarrow \infty$ while $k/n \rightarrow 0$ then the error rate of k NN converges to the Bayes error rate as $n \rightarrow \infty$. The result is strongest with $k/\log n \rightarrow \infty$ also. Note that because these results are for $n \rightarrow \infty$ they do not say which k is best for any particular finite n .

The three results above are true regardless of which distance metric is used. If the sample size is large enough, then any distance metric is adequate. Of course, for reasonable finite sample sizes, different distance metrics typically give very different error rates.

References

- [AHK01] Charu C. Aggarwal, Alexander Hinneburg, and Daniel A. Keim. On the surprising behavior of distance metrics in high dimensional spaces. In Jan Van den Bussche and Victor Vianu, editors, *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*, volume 1973 of *Lecture Notes in Computer Science*, pages 420–434. Springer, 2001.
- [KF07] Ashraf M. Kibriya and Eibe Frank. An empirical comparison of exact nearest neighbour algorithms. In *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'07)*, volume 4702 of *Lecture Notes in Computer Science*, pages 140–151. Springer, 2007.
- [Rip96] Brian Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.