Nearest Neighbor Classification

Charles Elkan
elkan@cs.ucsd.edu

January 8, 2010

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 kNN 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:

$$d(x, y) = ||x - y|| = \sqrt{(x - y) \cdot (x - y)} = (\sum_{i=1}^{m} (x_i - y_i)^2)^{1/2}$$

where x and y are points in $\mathbb{R}^m$.

An obvious disadvantage of the kNN method is the time complexity of making predictions. Suppose that there are n training examples in $\mathbb{R}^m$. Then applying the method to one test example requires $O(nm)$ time, compared to just $O(m)$ 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 m is small. However, for dimensionality $m \geq 20$ about, no data structure is known that is useful in practice [Kibriya and Frank, 2007].

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, so $d(x,z) \leq d(x,y)+d(y,z)$ for all x, y, and z. Then for all u, v, and w $d(u,w) \geq |d(u,v) - d(v,w)|$.
Proof: Note that $d(x,z) - d(y,z) \leq d(x,y)$ and similarly, $d(y,z) - d(x,z) \leq d(x,y)$. Hence $|d(x,z) - d(y,z)| \leq d(x,y)$. ■

The LAESA algorithm applies the lemma above 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. For any test point x:

compute $d(b,x)$ for all $b \in B$
for all $t \in T$
    compute lower bound $g(t,x) = \max_b |d(t,b) - d(b,x)|$
// invariant: $d(t,x) \geq g(t,x)$ for $t \in T$
initialize $y = \text{argmin}_b\ d(b,x)$
for all r in T sorted by increasing $g(r,x)$
    if $g(r,x) \geq d(y,x)$ then return y and finish
    else compute $d(r,x)$
        if $d(r,x) < d(y,x)$ then set $y = r$

Distances are computed just once between all training points and all points in the basis set B, in a preprocessing stage. The cost of this preprocessing is amortized over many test points x. The algorithm avoids computing distances by doing bookkeeping calculations involving lower bounds that are scalars. The relative computational benefit is larger when distance calculations are more expensive. This is true for Euclidean distance in high dimensions, and for distance functions that are not Euclidean but satisfy the triangle inequality, such as edit distance between strings. For Euclidean distance in low dimensions, there may be little benefit.

A major advantage of the kNN 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 $X \to Y$. 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.

Tasks such as the last two above are called structured prediction problems. The word "structured" refers to the fact that there are internal patterns in each output label. For example, suppose that x is a sentence of length 2. The part-of-speech labels $y =$ NOUN VERB and $y =$ VERB NOUN both have high probability, while the labels $y =$ NOUN NOUN and $y =$ VERB VERB do not. One major theme of recent research in machine learning has been to develop methods that can learn and use structure in output labels.

The nearest neighbor method is the only algorithm that is directly usable for all the types of problem listed above. Of course, we still need a useful distance function $d : X \times X \to \mathbb{R}$, which may not be obvious to design for some sets X. Note that a kNN 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 subspace of low dimensionality. 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 [Aggarwal et al., 2001]. This distance function is

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

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 [Ripley, 1996, 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 $k \to \infty$ while $k/n \to 0$ then the error rate of kNN converges to the Bayes error rate as $n \to \infty$. The result is strongest with $k/\log n \to \infty$ also [Devroye et al., 1994]. Note that because these results are for $n \to \infty$ 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.

CSE 250B Quiz for Thursday January 7, 2010

Your name:

While taking this quiz you may use only your own handwritten notes, and printed copies of CSE 250B lecture notes.

The LAESA algorithm is as follows, given a test point x:

compute $d(b, x)$ for all $b \in B$
for all $t \in T$
    compute lower bound $g(t, x) = \max_b |d(t, b) - d(b, x)|$
initialize $y = \text{argmin}_b\ d(b, x)$
for all r in T sorted by increasing $g(r, x)$
    if $g(r, x) \geq d(y, x)$ then return y and finish
    else compute $d(r, x)$
        if $d(r, x) < d(y, x)$ then set $y = r$

The algorithm uses a fixed set of basis points B. Distances are computed just once between all training points and all points in the set B, in a preprocessing stage. We did not specify in class how to choose the points in B.

Question [4 points]: Is it better to select points in B that are (a) far from each other, or (b) close to each other? Justify your answer in one or two sentences.

Grading guide: Scores are integers between 0 and 4. Assuming that the answer is correct, namely (a), the score starts at 4 points. Points can be lost for the following reasons. This list is more or less in order of importance from most serious to least serious: statements that are incorrect, no good justification given, reasons that are valid but less important than other reasons that are not stated, statements that are true but not relevant, sentences that are so complicated they are hard to understand, and errors in English that make writing hard to understand.

Assignment 1

This assignment is due at the start of class on Thursday January 21. You should work in a group of three students for this project. (If enrollment declines, then we will have groups of two for later projects.)

The goal of this project is to learn to recognize handwritten digits. Specifically, you will train nearest-neighbor and perceptron classifiers. The input to each classifier is a 16 by 16 array of pixels. The output of the classifier is one of the ten digits from 0 and 9. There are 7291 examples to be used for training, and 2007 examples for testing. Each pixel is a gray-scale intensity between -1 and +1. Note that although each digit is a two-dimensional array, for the project you should treat it as a one-dimensional vector of length $16^2 = 256$.

This dataset is called the USPS dataset, and results on it have been published in many papers. However, it is always important to check whether different papers use the exact same version of a dataset. It happens often that different papers use the same name for alternative versions of a dataset. The data are available at `http://www-stat-class.stanford.edu/~tibs/ElemStatLearn/data.html`. Do read the info file carefully to understand this particular version of the data.

Your task is to compare two supervised learning methods that are fundamentally different. Specifically, you must use Matlab or R to implement and do experiments with (i) the standard perceptron algorithm and (ii) k nearest-neighbor classification. For (i), you must first implement and test the two-class perceptron algorithm. Then, you must design and implement a multiclass classification strategy that uses two-class perceptrons as subroutines. Make sure your multiclass strategy is simple, sensible, and efficient. For (ii), you should use Euclidean distance, and select how many nearest neighbors k to use in a sensible way. You should concentrate on implementing and using the basic perceptron and nearest-neighbor methods correctly and efficiently; do not modify the basic methods. Be sure to design and describe clearly sensible tie-breaking heuristics. Also, design a simple sensible heuristic for terminating perceptron training.

You should do experiments with the multiclass task of distinguishing all ten digits. You must think rigorously about designing experiments that are conceptually sound. For final accuracy measurements, use the standard test set of 2007 examples. Compare your performance with published results using other methods from at least one published paper, for example [Shivaswamy and Jebara, 2008]. Do not overfit the test set!

For this project and for later ones, the only deliverable is a well-written joint report for each team. As is the case for a research paper, your job is to inform and convince the reader fully. Please bring a printed copy to the start of class on January 21.

To quote Sanjoy Dasgupta, "Discuss your results in precise and lucid prose. Content is king, but looks matter too!" Your report should be self-contained and complete, but concise. You should have separate sections for at least the following: (i) introduction,

(ii) design of algorithms, (iii) design of experiments, (iv) results of experiments, and (v) findings and lessons learned.

The report should be typeset with LaTeX and have appropriate equations, tables, figures, citations, and references. Explain in reproducible detail your algorithms, especially the heuristics that you invent, and the design of your experiments. These explanations should be separate from actual experimental results. Some issues to discuss briefly in the report include whether the training and test sets follow the same probability distribution, and whether overfitting is a problem. Analyze the time and space complexity of your algorithms, and provide brief timing results. Explain briefly the implementation choices that allow your code for the algorithms to be fast.

Matlab notes

You should implement the LAESA algorithm. Design your own simple heuristic for choosing the basis points. Do informal experiments to select how many basis points to use. An important question is whether LAESA, or any other method, can be implemented so that it actually is faster than "brute force" computation of all distances. Even if LAESA can be implemented to be faster in some languages, it may be slower in Matlab because Matlab performs large-scale matrix arithmetic very fast, but updating data structures and performing logical tests in Matlab tends to be slow.

The following is the fastest Matlab code that I know for calculating Euclidean distances between a test point and all points in a training set. It is fast because it uses the identity $(a-b)^2 = a^2 - 2ab + b^2$ to avoid copying data.

```
function distances = calcdist(data,point)
%  input: data matrix, single point (points are row vectors)
% output: column vector of distances
distances = sum(data.^2, 2) - 2*data*point' + point*point';
distances = sqrt(distances);
```

Use this code as a starting point for your work. If you can make it significantly faster, explain how in your report. If you can actually make LAESA faster, also explain how.

Generally, for fast Matlab subroutines use the Lightspeed toolbox published by Minka at http://research.microsoft.com/en-us/um/people/minka/software/lightspeed/.

References

[Aggarwal et al., 2001] Aggarwal, C. C., Hinneburg, A., and Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional spaces. In den Bussche, J. V. and Vianu, V., 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.

[Devroye et al., 1994] Devroye, L., Gyorfi, L., Krzyzak, A., and Lugosi, G. (1994). On the strong universal consistency of nearest neighbor regression function estimates. *Annals of Statistics*, 22:1371–1385.

[Kibriya and Frank, 2007] Kibriya, A. M. and Frank, E. (2007). 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.

[Ripley, 1996] Ripley, B. (1996). *Pattern Recognition and Neural Networks*. Cambridge University Press.

[Shivaswamy and Jebara, 2008] Shivaswamy, P. K. and Jebara, T. (2008). Relative margin machines. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *NIPS*, pages 1481–1488. MIT Press.