K Nearest Neighbors for Regression `knn-cv-1`

Carl Edward Rasmussen

May 3, 1996

Nearest neighbor models are popular non-parametric memory based models. I will consider a simple k nearest neighbor model, which in the context of DELVE will be called `knn-cv-1`. I will attempt to define a nearest neighbor method that has reproducible results, that does not depend heavily on details of implementation, has few "free" parameters and behaves reasonably under a broad variety of conditions. Also, I wish the method to be applicable to our three standard loss functions (absolute error, squared error and negative log density loss).

Simple nearest neighbor models do not require any training. The strategy for making predictions involves searching through the training cases and finding those whose inputs are closest to the inputs of the test case, and using some kind of weighted average of the targets of these neighbors as a prediction. Neighborhoods can either be defined in terms of a kernel (eg. Gaussian) which supplies weighting factors, or in terms of the number of neighbors k to use (or hybrids of these). Although methods using kernels have the intuitively appealing ability of weighting neighbors according to distance, they will not be used here. Firstly, they involve the somewhat arbitrary choice of kernel shape and secondly, the choice of width for the kernel (a continuous equivalent of the discrete choice of k) may be plagued by local minima; this makes the procedure unsuitable as a reproducible base-line nearest neighbor approach. Instead, I will use k nearest neighbor approach with uniform weighting of neighbors for predictions.

We also need to define the metric in which to measure closeness; I will use the simple and common choice of Euclidean distance. Many extensions of nearest neighbor algorithms exist which attempt to adapt the distance metric (Lowe 1995; Hastie and Tibshirani 1996), but these more complicated algorithms will not be pursued here.

We need to resolve what to do if two or more examples have the same distance as the k'th nearest neighbor; in an attempt to make the algorithm less sensitive to round off effects on different floating point arithmetic implementations, I will further set some small distance tolerance, below which cases are deemed to have the *tied* distances. Two cases are defined to be ties if their squared distances to the test point differ by less than $tol = 1.0 \times 10^{-6}$. I propose the following scheme:

1. Sort the examples by distances, placing ties in arbitrary order.
2. Find all the neighbors which have the same distance as the k'th example in the ordering (including both examples earlier and later in the list) and average their targets.
3. Include this average in the final prediction with a weight equal to the number of ties that were at or earlier than k in the list.

This procedure together with the small tolerance on deciding whether cases are ties, should help in avoiding differences due to finite precision arithmetic, and also produce predictions that are not heavily dependent on distant neighbors in situations where many examples tie by weighting down their contributions accordingly.

We need to find an appropriate neighborhood size k. In general, we expect the optimal choice of k to depend on the particular dataset and on the number of available training points. As the number of training cases increases the nearest neighbor method remains consistent only as k grows. An appealing idea is to find the best k by leave one out cross validation. Using this procedure, we leave out each of the training examples in turn, find it's k' nearest neighbors (handling ties appropriately) and compute the loss associated with predictions using this neighborhood. This is repeated for all $k' = 1 \ldots n-1$, and k is selected to be the k' with the smallest average loss. The leave one out procedure gives an unbiased estimate for k for the situation where we had $n-1$ training cases, which in the case of fairly large n should be close to optimal for the current training set size.

The neighborhood size k is estimated using the desired loss function. For the squared error loss function, the average of the targets of the nearest neighbors as used as predictions. If the outputs are multidimensional then averages are computed for each dimension. For absolute error loss we use the median along each dimension as a prediction.

Some special problems present themselves for negative log density loss. We have to form a predictive distribution based on the k nearest neighbors. The simplest idea is to fit an axis-aligned Gaussian to the targets of the k neighbors, again weighting the neighbors uniformly. The mean for the Gaussian is chosen simply to be the empirical mean of the k nearest neighbors. Two special cases arise when estimating the variance. Firstly, if the leave one out procedure estimates $k = 1$, then we cannot estimate both the mean and variance for the predictive distribution based on the single nearest neighbor. Secondly, if all k neighbors have exactly the same targets (or targets that differ only very slightly), the naive empirical variance estimate is undesirable since it leads to unbounded losses. Both of these problems can be addressed by regularising the variance estimate

$$v = \frac{1}{k}\Big(\sum_{k'=1}^{k}(y^{(k')} - \bar{y})^2 + \varsigma^2\Big), \tag{1}$$

where ς^2 is the regulariser, $y^{(k')}$ is the targets of the k''th nearest neighbor and $\bar{y}$ is the average of the targets for the k nearest neighbors. A convenient value to choose for ς^2 is the average squared difference between targets corresponding to first nearest neighbors, the average extending over all training examples. This can be interpreted as a global estimate of the variance which, whenever available, is modified by local information. Thus the regulariser will act much like a canonical prior. In the case of $k = 1$ no local information about variance is available, and the sum over k' will vanish from eq. (1) leaving just the global term. As k grows, the amount of local information increases and the importance of the regulariser decreases. The problems of vanishing variance estimates is also addressed by this approach, since normalisation of the targets before learning guarantees that $\varsigma^2 > 0$ (except for the pathological case where all available training cases have equivalent targets).

The computational demands of the leave one out procedure for training the `knn-cv-1` model depends on how it is implemented. We need to leave out each of the n examples in turn; for each of these we compute the distance to all other neighbors, which takes $O(nm)$. Then we need to sort these distances, requiring $O(n \log n)$. Now for each of the $n-1$ possible values of k we need to compute the mean, variance and median of the neighbors. For simple implementations computation of these values take $O(n)$ for each of these estimates. However, it should be noted that it is possible to re-use partial results obtained with other values of k; means and variances can be updated in constant time and the median can (through use of a heap data structure) be updated in $O(\log n)$. The total computational effort involved in finding k is thus $O\big(n^2(m+n+\log n)\big)$ for a simple implementation or $O\big(n^2(m+\log n)\big)$. The current version of `knn-cv-1` uses the simple implementation.

Once the appropriate value of k is found, we can make predictions in time $O\big(mn+k+n\log n\big)$ for each test case, if the prediction algorithm sorts all training cases according to distance (the current implementation of `knn-cv-1` uses this approach). If instead of sorting we attempt to locate the k closest neighbors, we can replace $\log n$ in the previous expression by k, but it is not clear which is best in general since we expect the optimal k to grow as n grows.

References

Hastie, T. and Tibshirani, R. 1996. Discriminant adaptive nearest neihgbor classification and regression. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 290–297. MIT Press. `ftp://playfair.stanford.edu/pub/hastie/nips95.ps.Z`.

Lowe, D. G. 1995. Similarity metric learning for a varable–kernel classifier. *Neural Computation*, 7(1):72–85.