Kernel K-Nearest Neighbor Classifier with P-Value Computation

Darrin P. Lewis William Stafford Noble

17 July, 2001

Abstract

1 Introduction

2 Kernel Methods

"Kernel methods" broadly refers to the use of kernel functions for the purposes of machine learning. A kernel is a function K, such that for all $x, z \in X$

$$K(x, z) = \langle \phi(x) \cdot \phi(z) \rangle$$

where ϕ is a mapping from the input space, X, to an inner product feature space.[5]

In machine learning applications, a kernel is used in place of the standard dot product to evaluate a distance between two vectors in a feature space. The feature map ϕ is often chosen to be non-linear, thus providing the advantage of the kernel over a simple dot product. Using a kernel, input data which are not linearly separable can be compared in a non-linear, often higher-dimensional feature space in which they are linearly separable. The kernel function can incorporate prior knowledge of the input space, to make the data maximally separable in the feature space, with respect to the desired classifications.

2.1 Kernel Functions

Kernel functions have been known to mathematicians for nearly one hundred years. They originated in the study of integral operator theory. [] In the 1950s reproducing kernels were discovered and reproducing kernel Hilbert space (RKHS) [] was defined. These important advances in theory found practical application when, in the 1990s, Boser, Guyon, and Vapnik introduced kernel functions to the machine learning community in the context of their Support Vector Machines (SVM) [3]. Since that introduction, researchers have found benefit in applying kernel functions in other algorithms such as clustering [], and principle components analysis [] as well.

The simplest kernel function is the standard dot product. In that case, the feature map is just the identity,

$$\phi(x) = x$$

and the feature space is equal to the input space. Another simple but non-linear kernel is the squared dot product. It can be written as follows:

$$\begin{aligned}
\langle x \cdot z \rangle^2 &= \Big(\sum_{i=1}^{n} x_i z_i \Big)^2 = \Big(\sum_{i=1}^{n} x_i z_i \Big) \Big(\sum_{j=1}^{n} x_j z_j \Big) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j z_i z_j = \sum_{(i,j)=(1,1)}^{(n,n)} (x_i x_j)(z_i z_j) \\
&= \langle \phi(x) \cdot \phi(z) \rangle,
\end{aligned}$$

where

$$\phi(x) = \big((x_1 x_1), (x_1 x_2), \ldots, (x_i x_j), \ldots, (x_n x_n) \big).$$

In this case, the feature space is all monomials of degree 2.

3 Class Significance by Cross-Validation

We propose to rank the significance of a given class labeling according to how well it can be used to discriminate the data. The premise is that if the class generalizes well, as measured by cross validation, then it exposes some characteristic of the data which may be worth further investigation.

We chose to perform hold-one-out cross validation experiments using an implementation of a K-nearest neighbor (KNN) classifier. The same three data sets and class labels are used as for the other experiments. In this case, each example is in turn regarded as unlabeled and KNN is used to determine a predicted label. When all examples have been classified, the predictions are compared to the known labels and the rates of true positive (TP), false negative (FN), true negative (TN), and false positive (FP) are computed. In addition, a p-value is computed as a statistical measure of significance.

3.1 K-Nearest Neighbor

The K-nearest neighbor classifier defines the label of an unclassified example as the label belonging to the majority of the K nearest "neighboring" examples. It is unique as a supervised learning algorithm in that there is no training step required. The interpoint distances must simply be computed for all pairs of examples. The resulting savings in computational time, along with the simplicity of the algorithm make it attractive.

One potential weakness of K-nearest neighbor is its strong dependence on a distance metric. With a simple Euclidean distance metric, KNN defines a hypershpere centered in the input space at the example point of interest, and of smallest radius, which contains K neighboring points. The class labels of these points are tallied to arrive at the predicted class for the unlabeled example. We address this shortcoming by replacing the distance metric with a Mercer kernel function. The kernel acts as a dot product in some possibly nonlinear feature space which is implicitly defined by the kernel function and into which the input is projected. Thus, KNN still employs a hypersphere to select nearby neighbors, but this hypersphere exists in the feature space. The geometry of the sphere in the input space may be difficult or impossible to determine.

For the remainder of this paper, and through the experiments, we use a simple dot product kernel function for which the feature space and input space are identical. The dot product between two examples, represented as vectors, yeilds a measure of similarity. The dot product is easily converted to a Euclidean distance by the relation

$$D(x,y) = < x,x > -2 < x,y > + < y,y >$$

. It is this Euclidean distance that will be used by KNN.

3.2 p-value Computation

Due to the electoral nature of KNN, the relative sizes of the classes has a large bearing on the outcome, such that even in a randomly labeled data set, the number of misclassifications are correlated with the size of the class. This is clearly undesirable. However, due to the combinatorial nature of the KNN election process, it is possible to compute the probability of an individual misclassification error, given K and the size of the class and data set, assuming randomly labeled data. Because the geometry of the space becomes random, the problem becomes a straightforward selection problem.

Denote the entire data set as $\mathcal{D}$, the positive class as $\mathcal{P}$, and the negative class as $\mathcal{N}$. Their respective sizes are D, P, and N. $\mathcal{P} \cap \mathcal{N} = \oslash$ and $\mathcal{P} \cup \mathcal{N} = \mathcal{D}$. Also denote an example, X classified positive by KNN, as $X_{\mathcal{P}}$ and likewise for $\mathcal{N}$. Assume that an example, $X \in \mathcal{P}$ is chosen at random. To encounter an error, we must classify X as $\mathcal{N}$, which can only occur if fewer than $\lfloor \frac{K}{2} \rfloor$ of the next K points, also chosen at random, have label $\mathcal{P}$. This is expressed in the following conditional probability,

$$\mathbf{pr}(X_{\mathcal{N}} | X \in \mathcal{P}) = \frac{\sum_{i=0}^{\lfloor \frac{K}{2} \rfloor} \binom{P-1}{i} \binom{D-P}{K-i}}{\binom{D-1}{K}}.$$

Thus,

$$\mathbf{pr}(FN) = \mathbf{pr}(X_{\mathcal{N}} | X \in \mathcal{P}) \mathbf{pr}(X \in \mathcal{P})$$

and

$$\mathbf{pr}(FN + FP) = \mathbf{pr}(X_{\mathcal{N}} | X \in \mathcal{P}) \mathbf{pr}(X \in \mathcal{P}) + \mathbf{pr}(X_{\mathcal{P}} | X \in \mathcal{N}) \mathbf{pr}(X \in \mathcal{N}) \tag{1}$$

Based upon the probability in (1), a binomial cummulative distribution can be computed over $[0, D]$ for each class size. Using these distributions, a p-value can be obtained for any KNN cross validation score based on the positive class size.

(Can illustrate that p-values address the problem by including the normalized rank vs. p-value plot for random classes.)

4 Experiments

(DBF kernel vs. dot product kernel using KKNN)

5 Results

(Show table and graph of comparison.)

References

[1] B. Alberts, D. Bray, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter. *Essential Cell Biology.* Garland Publishing, 1998.

[2] D. Anastassiou. Lecture 10. *Genomic Information Science and Technology.* Columbia University, 2000.

[3] B. Boser, I. Guyon, V. Vapnik. A training algorithm for optimal margin classifiers. *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory,* p144-152. ACM Press, 1992.

[4] M. Brown, W. Grundy, D. Lin, N. Christianini, C. Sugnet, T. Furey, M. Ares, Jr., D. Haussler. Knowledge-based analysis of microarray gene expression data by using support vector machines. *Proc. Natl. Acad. Sci. USA* 97(1):262-267, 2000.

[5] N. Christianini, J. Shawe-Taylor. *Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[6] M. Eisen, P. Spellman, P. Brown, D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA* 95(25):14863-14868, 1998.

[7] L. Hunter, R. Taylor, S. Leach, R. Simon. GEST: A Gene Expression Search Tool Based on a Novel Bayesian Similarity Metric. *Unpublished Manuscript.* National Cancer Institute.

[8] A. Jain, R. Dubes. *Algorithms for Clustering Data.* Prentice Hall, 1988.

[9] M. Kerr, G. Churchill. Experimental Design for Gene Expression Microarrays. *Unpublished Manuscript.* The Jackson Laboratory.

[10] M. Kerr, M. Martin, G. Churchill. Analysis of Variance for Gene Expression Microarray Data. *Unpublished Manuscript.* The Jackson Laboratory.

[11] M. Schena, D. Shalon, R. Davis, P. Brown. Quantitative Monitoring of Gene Expression Patterns with a Complementary DNA Microarray *Science* 270(5235):467-470, 1995.

[12] B. Schölkopf, C. Burges, A. Smola. *Advances in Kernel Methods.* The MIT Press, 1999.

[13] P. Sneath, R. Sokal. *Numerical Taxonomy.* W.H. Freeman and Company, 1973.

[14] D. Wilson, T. Martinez. Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research.* 6:1-34, 1997.