

---

# Large-Scale Kernel Discriminant Analysis with Application to Quasar Discovery

Alexander Gray and Ryan Riegel\*

Georgia Institute of Technology, College of Computing  
{agray, rriegel}@cc.gatech.edu

**Summary.** Kernel discriminant analysis offers qualities desirable to many applications, including its potential for accurate class probability estimates, its straightforward interpretation, and its natural way of incorporating prior knowledge. Unfortunately, like other highly accurate classifiers, naïve implementations of this technique are computationally infeasible for massive datasets, which are becoming common in many areas. We present a fast algorithm for performing classification with the kernel discriminant exactly (i.e. without any approximation error). We demonstrate its use for quasar discovery, a problem central to cosmology and astrophysics, tractably using 500K training data and 800K testing data from the Sloan Digital Sky Survey. The resulting catalog of 100K quasars significantly exceeds existing quasar catalogs in both size and quality, opening a number of new scientific possibilities, including the recent empirical confirmation of cosmic magnification described in *Nature*.

**Key words:** kernel discriminant analysis, quasar discovery, *kd*-trees

## 1 Kernel Discriminant Analysis

The optimal Bayes rule [4] for a 2-class classification (or discriminant analysis) problem assigns a  $D$ -dimensional observation  $x$  to either class  $C_1$  or  $C_2$  according to the larger posterior class probability

$$P(C_1|x) = \frac{f(x|C_1)P(C_1)}{f(x|C_1)P(C_1) + f(x|C_2)P(C_2)},$$

where  $f(x|C)$  is the density of data from class  $C$  and  $P(C)$  is the prior for class  $C$ . We will consider the form of this rule which uses kernel density estimates [8] of the class-conditional densities of the form

---

\* In collaboration with Gordon T. Richards, Johns Hopkins University, Dept. of Physics & Astronomy; Robert C. Nichol, Univ. of Portsmouth, Inst. of Cosmology & Gravitation; and Robert J. Brunner, Univ. of Illinois, Dept. of Astronomy.

$$\hat{f}(x|C) = \frac{1}{W} \sum_i^N w_i K_h(x, x_i),$$

where the kernel is, for example, the Gaussian  $K_h(x, x_i) = \frac{1}{C(h)} \exp \|x - x_i\|^2/h^2$  where  $C(h)$  is a normalizing constant and  $W = \sum_i^N w_i$ , a slight generalization of the usual kernel density estimator allowing for measurement uncertainty or other knowledge via weighted points. Priors may be constant or given by a data-dependent function  $\pi(x)$ . Training the kernel discriminant consists of finding the bandwidths  $\{h_1, h_2\}$  that minimize the expected error on future test data, which we estimate using leave-one-out cross-validation.

Kernel discriminant analysis (KDA) can be very useful in practical applications such as the astrophysics problem we describe later in this paper. In our experience, scientists find it to be a natural formulation of classification which is easy to interpret and allows clear ways of encoding prior knowledge. Our application is one in which accurate class probability estimates are important, not just class label predictions, making KDA additionally compelling.

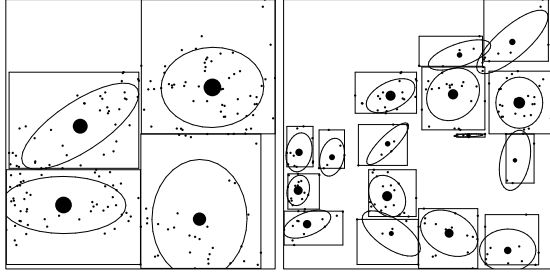
Unfortunately, the obvious approach, which sums over all pairs of training and test points, is not computationally tractable for the very large datasets becoming more common throughout science. In this paper we describe and demonstrate an algorithm that performs KDA classification in less time than the naïve algorithm while still computing the same class label predictions (*i.e.* exactly, without any approximation error). Note that using a method such as the FFT [7] to compute each kernel density estimate approximately does not guarantee correct classifications in general. Because we can also perform this fast computation in a leave-one-out cross-validation setting, we have a way of evaluating bandwidth parameters quickly (training) as well as labeling new points quickly (testing). The algorithm achieves its speed by computing only bounds on the class probability estimates, not the estimates themselves.

## 2 Algorithm

We developed an algorithm which computes for each query point  $x_q$  its class label  $c(x_q)$  as if the probabilities  $P(C_1|x)$  and  $P(C_2|x)$  had been computed *exactly*, though in many cases they need not be. The algorithm uses unnormalized sums  $\Phi(x|C) = \sum_{c(x_r)=C} w_i K_h(x, x_r)$ , with  $f(x|C) = \frac{1}{W} \Phi(x|C)$ . We compress the notation by referring to  $P(C_j|x)$  as  $p_j(x)$  and  $\Phi(x|C_j)$  as  $\Phi_j(x)$ .

**kd-trees.** First, a space-partitioning tree data structure such as a *kd-tree* [3] is constructed on the query (testing) data, and another is created on the reference (training) data. Each node represents a subset of the data by its bounding box. Non-leaf nodes have two children, obtained by splitting the widest dimension of the parent's bounding box. Nodes are split until they contain fewer than a critical number of points, and then become leaves.

The idea is to maintain bounds on  $p_1(x)$  and  $p_2(x)$  and successively tighten them in a multi-resolution manner, as nodes at increasingly finer levels of the



**Fig. 1.** A *kd*-tree.  
 (left) Nodes at level 3.  
 (right) Nodes at level 5.  
 Shown are the points, centroids, covariances, and bounding boxes for each node.

trees are considered, until we can show that the bounds determine that one of these class probabilities must dominate the other. This is true when one’s lower bound is greater than the other’s upper bound, or equivalently, when one’s lower bound is greater than 0.5. Initially the class label for each query point  $c(x_q)$  is recorded as “?” (unknown), and is updated to  $C_1$  or  $C_2$  when the bounds determine it. An efficiency over the naïve algorithm is obtained when we determine the label for large chunks of the query points simultaneously.

**Bounds.** We maintain various bounds during the run of the algorithm, including bounds on  $\Phi_1(x)$  and  $\Phi_2(x)$ , e.g.  $\Phi_1^L(x) \leq \Phi_1(x)$  and  $\Phi_1^U(x) \geq \Phi_1(x)$ . We also maintain bounds for subsets  $X$  of the data corresponding to tree nodes, e.g.  $\forall x \in X: \Phi_1^L(X) \leq \Phi_1(x)$  and  $\Phi_1^U(X) \geq \Phi_1(x)$ . We use these bounds to obtain bounds on the posterior class probability as follows:

$$p_1^L(x) := (\Phi_1^L(x)\pi_1^L(x)) / (\Phi_1^L(x)\pi_1^L(x) + \Phi_2^U(x)\pi_2^U(x))$$

$$p_1^U(x) := (\Phi_1^U(x)\pi_1^U(x)) / (\Phi_1^U(x)\pi_1^U(x) + \Phi_2^L(x)\pi_2^L(x)).$$

Similar equations are used to find bounds on class probabilities for tree nodes.

Within each reference node  $R$ , in an efficient bottom-up (dynamic programming) fashion, we compute and store certain properties of the points of either class residing within the node: bounding boxes for the points in either class and the sums of the weights of the points in either class,  $W_1(R)$  and  $W_2(R)$ .<sup>2</sup> We can use these bounding boxes to compute simple lower and upper bounds on the distance between any point in a node  $Q$  and any point (of a certain class) in a node  $R$ , e.g.:  $\forall x_q \in Q, \forall x_r \in R$  such that  $c(x_r) = C_1$ :  $\delta_1^L(Q, R) \leq \delta_{qr}$  and  $\delta_1^U(Q, R) \geq \delta_{qr}$ , where  $\delta_{qr} = \|x_q - x_r\|$ .

**Bound tightening.** Let  $K^L$  and  $K^U$  be constants such that  $\forall x, y: K^L \leq K(x, y)$  and  $K^U \geq K(x, y)$ . Most kernels of interest, such as the Gaussian, are probability density functions and can be scaled by  $C(h)$  such that these are 0 and 1, respectively. At the beginning of the algorithm, the bounds are initialized using these values, e.g.:  $\forall x_q: \Phi_1^L(x_q) = W_1K^L$  and  $\Phi_1^U(x_q) = W_1K^U$ . For each query  $x_q$ , the bounds  $\Phi_1^L(x)$  and  $\Phi_1^U(x)$  account for each reference point’s potential contribution in a worst-case manner.

Nodes are examined in pairs  $\{Q, R\}$ , with node  $Q$  from the query tree and node  $R$  from the reference tree. The idea is that when we see a new

<sup>2</sup> Expressions like  $W_1(R)$  are generally implemented as  $X.W_1$  in C-like notation.

reference node  $R$ , we can tighten our bounds on the contribution of the reference points in  $R$  to the sum for each query point. When doing so, we must also undo any previous contributions of the points in  $R$  already expressed in the bounds. For example, the new contribution of  $R$  to  $\Phi_1^U(Q)$  is  $W_1(R)K(h_1, \delta_1^L(Q, R))$  whereas the old contribution, made by the immediate parent of  $\{Q, R\}$ ,  $\{\text{pa}(Q), \text{pa}(R)\}$ , was  $W_1(\text{pa}(R))K(h_1, \delta_1^L(\text{pa}(Q), \text{pa}(R)))$ , and the initial contribution was implicitly  $W_1(R)K^U$ . We update  $\Phi_1^U(Q)$  by adding to it  $\Delta\Phi_1^U(Q, R) := W_1(R)K_{h_1}(\delta_1^L(Q, R)) - W_1(R)K^U$  and by subtracting  $\Delta\Phi_1^U(\text{pa}(Q), \text{pa}(R))$  exactly once for each child of  $\text{pa}(Q)$ , when processing the first pair that includes it and a child of  $\text{pa}(R)$ . As we move down the tree, processing various pairs of nodes' children, updates improve the bounds or at worst leave them unchanged.

**Control flow.** The order in which nodes are examined is determined by a min-priority queue which stores node-pair objects  $\{Q, R\}$ .<sup>3</sup> A node-pair object stores the change values that are computed for it, the *previous* such values (denoted by apostrophes) computed for its parents, its priority, and an “undo” flag which indicates whether its parent’s contribution should be subtracted.

Node-pair  $\{Q, R\}$  is assigned its priority based on the heuristic  $\rho(Q, R) := (\Delta\Phi_1^U - \Delta\Phi_1^{U'}) + (\Delta\Phi_2^U - \Delta\Phi_2^{U'}) - (\Delta\Phi_1^L - \Delta\Phi_1^{L'}) - (\Delta\Phi_2^L - \Delta\Phi_2^{L'})$ , or the overall improvement of the upper and lower bounds. Note that the changes to the upper bounds are non-positive and the changes to the lower bounds are non-negative, and that the current values are at least as extreme as the previous values, so this result is always non-positive. This heuristic favors pairs that provide the most additional bounds tightening, and thus aims to increase the frequency of proving one class probability to dominate the other.

A procedure **makePair** $(Q, R, \dots)$  creates the node-pair structure  $\{Q, R\}$  and stores the other arguments in its slots for  $\Delta\Phi_1^{L'}$ ,  $\Delta\Phi_1^{U'}$ ,  $\Delta\Phi_2^{L'}$ , and  $\Delta\Phi_2^{U'}$ . **computeBounds** $(\{Q, R\})$  computes the  $\Delta$  values and priority for the input pair. Node-pairs are expanded further by placing all pairwise combinations of their children on the queue. Whenever improvements are made to the bounds of a query node, they are updated in all the children of the query node with a simple recursive routine **passDown** $(Q, \dots)$ . For each node  $Q$  in the query tree we store  $M(Q)$ , the number of points in the tree which have proven classifications. If we encounter a node where  $M(Q) = N(Q)$ , its total number of points, we can stop recursing on it. When both  $Q$  and  $R$  are leaf nodes, we perform the base case of the recursion, computing the contribution of each point in  $R$  to each point in  $Q$  exhaustively. Because this direct type of contribution is exact and thus unchangeable, it is useful to record it separately from the bounds. We denote it by  $\phi(x_q)$  for query points and  $\phi(Q)$  for query nodes. The two main functions are shown in pseudocode.<sup>4</sup>

<sup>3</sup> Values like  $\Delta\Phi_1^L(\{Q, R\})$ , or  $\Delta\Phi_1^L$  for short, are  $\{Q, R\}.\Delta\Phi_1^L$  in C-like notation.

<sup>4</sup> For brevity, we use the convention that children of leaves point to themselves, and redundant node pairs are not placed on the priority queue. Also,  $a += b$  and  $a -= b$  denote  $a := a + b$  and  $a := a - b$ , respectively.

```

kda( $Q_{root}, R_{root}$ )
   $\{Q_{root}, R_{root}\} := \mathbf{makePair}(Q_{root}, R_{root}, 0, 0, 0, 0)$ .
  computeBounds( $\{Q_{root}, R_{root}\}$ ).
  insertHeap( $H, \{Q_{root}, R_{root}\}$ ).

while  $H$  is not empty,
   $\{Q, R\} := \mathbf{extractMin}(H)$ .
  if  $M(Q) = N(Q)$ , skip.
  if not leaf( $Q$ ),
     $\Phi_1^L(Q) := \min(\Phi_1^L(\text{ch}_1(Q)), \Phi_1^L(\text{ch}_2(Q)))$ .
     $\Phi_1^U(Q) := \max(\Phi_1^U(\text{ch}_1(Q)), \Phi_1^U(\text{ch}_2(Q)))$ .
    (similar for class 2 and  $\phi$ )
     $M(Q) := M(\text{ch}_1(Q)) + M(\text{ch}_2(Q))$ .

 $\Delta\Phi_1^L := \Delta\Phi_1^L(\{Q, R\}), \dots, \pi_1^L := \pi_1^L(Q), \dots$ 
if undo( $Q, R$ ),
  passDown( $Q, \Delta\Phi_1^L - \Delta\Phi_1^{L'}, \Delta\Phi_1^U - \Delta\Phi_1^{U'}, \Delta\Phi_2^L - \Delta\Phi_2^{L'}, \Delta\Phi_2^U - \Delta\Phi_2^{U'}$ ).
else, passDown( $Q, \Delta\Phi_1^L, \Delta\Phi_1^U, \Delta\Phi_2^L, \Delta\Phi_2^U$ ).

 $\phi_1^L := C(h_1)(\Phi_1^L(Q) + \phi_1^L(Q))$ .  $\phi_2^L := C(h_2)(\Phi_2^L(Q) + \phi_2^L(Q))$ .
 $\phi_1^U := C(h_1)(\Phi_1^U(Q) + \phi_1^U(Q))$ .  $\phi_2^U := C(h_2)(\Phi_2^U(Q) + \phi_2^U(Q))$ .
 $p_1^L(Q) = \Phi_1^L \pi_1^L / (\Phi_1^L \pi_1^L + \Phi_2^U \pi_2^U)$ .  $p_1^U(Q) = \Phi_1^U \pi_1^U / (\Phi_1^U \pi_1^U + \Phi_2^L \pi_2^L)$ .

if  $p_1^L(Q) \geq 0.5$ ,  $c(Q) := C_1$ . if  $p_1^U(Q) < 0.5$ ,  $c(Q) := C_2$ .
if  $c(Q) \neq \text{"?"}$ ,
   $M(Q) := N(Q)$ . skip.

if leaf( $Q$ ) and leaf( $R$ ),
  passDown( $Q, -\Delta\Phi_1^L, -\Delta\Phi_1^U, -\Delta\Phi_2^L, -\Delta\Phi_2^U$ ).
  kdaBase( $Q, R$ ).
else,
   $\{\text{ch}_1(Q), \text{ch}_1(R)\} := \mathbf{makePair}(\text{ch}_1(Q), \text{ch}_1(R), \Delta\Phi_1^L, \Delta\Phi_1^U, \Delta\Phi_2^L, \Delta\Phi_2^U)$ .
   $\{\text{ch}_1(Q), \text{ch}_2(R)\} := \mathbf{makePair}(\text{ch}_1(Q), \text{ch}_2(R), \Delta\Phi_1^L, \Delta\Phi_1^U, \Delta\Phi_2^L, \Delta\Phi_2^U)$ .
  computeBounds( $\{\text{ch}_1(Q), \text{ch}_1(R)\}$ ). computeBounds( $\{\text{ch}_1(Q), \text{ch}_2(R)\}$ ).
  if  $\rho(\{\text{ch}_1(Q), \text{ch}_1(R)\}) < \rho(\{\text{ch}_1(Q), \text{ch}_2(R)\})$ ,
    undo( $\{\text{ch}_1(Q), \text{ch}_1(R)\}$ ) := true.
  else, undo( $\{\text{ch}_1(Q), \text{ch}_2(R)\}$ ) := true.
  insertHeap( $H, \{\text{ch}_1(Q), \text{ch}_1(R)\}$ ). insertHeap( $H, \{\text{ch}_1(Q), \text{ch}_2(R)\}$ ).

   $\{\text{ch}_2(Q), \text{ch}_1(R)\} := \mathbf{makePair}(\text{ch}_2(Q), \text{ch}_1(R), \Delta\Phi_1^L, \Delta\Phi_1^U, \Delta\Phi_2^L, \Delta\Phi_2^U)$ .
   $\{\text{ch}_2(Q), \text{ch}_2(R)\} := \mathbf{makePair}(\text{ch}_2(Q), \text{ch}_2(R), \Delta\Phi_1^L, \Delta\Phi_1^U, \Delta\Phi_2^L, \Delta\Phi_2^U)$ .
  computeBounds( $\{\text{ch}_2(Q), \text{ch}_1(R)\}$ ). computeBounds( $\{\text{ch}_2(Q), \text{ch}_2(R)\}$ ).
  if  $\rho(\{\text{ch}_2(Q), \text{ch}_1(R)\}) < \rho(\{\text{ch}_2(Q), \text{ch}_2(R)\})$ ,
    undo( $\{\text{ch}_2(Q), \text{ch}_1(R)\}$ ) := true.
  else, undo( $\{\text{ch}_2(Q), \text{ch}_2(R)\}$ ) := true.
  insertHeap( $H, \{\text{ch}_2(Q), \text{ch}_1(R)\}$ ). insertHeap( $H, \{\text{ch}_2(Q), \text{ch}_2(R)\}$ ).

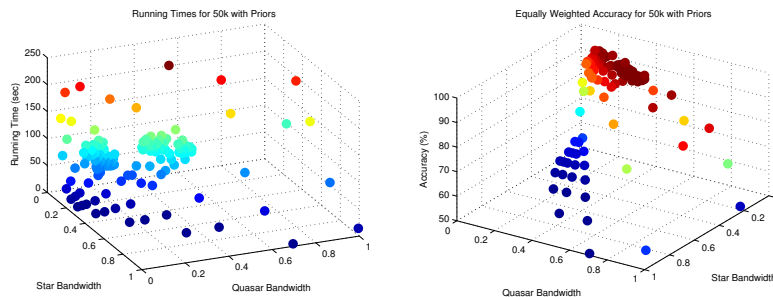
```

<p><b>kdaBase</b>(<math>Q, R</math>)</p> $\Phi_1^L(Q) := W_1(R)K^L. \Phi_2^L(Q) := W_2(R)K^L.$ $\Phi_1^U(Q) := W_1(R)K^U. \Phi_2^U(Q) := W_2(R)K^U.$ <p>forall <math>x_q \in Q</math> such that <math>c(x_q) = \text{"?"}</math>,  forall <math>x_r \in R</math>,  if <math>c(x_r) = C_1</math>, <math>\phi_1(x_q) += w_r K_{h_1}(\delta_{qr})</math>. if <math>c(x_r) = C_2</math>, <math>\phi_2(x_q) += w_r K_{h_1}(\delta_{qr})</math>.</p> $\Phi_1^L(x_q) := C(h_1)(\Phi_1^L(Q) + \phi_1(x_q)). \Phi_2^L(x_q) := C(h_2)(\Phi_2^L(Q) + \phi_2(x_q)).$ $\Phi_1^U(x_q) := C(h_1)(\Phi_1^U(Q) + \phi_1(x_q)). \Phi_2^U(x_q) := C(h_2)(\Phi_2^U(Q) + \phi_2(x_q)).$ $p_1^L(x_q) := \Phi_1^L(x_q)\pi_1(x_q) / (\Phi_1^L(x_q)\pi_1(x_q) + \Phi_2^U(x_q)\pi_2(x_q)).$ $p_1^U(x_q) := \Phi_1^U(x_q)\pi_1(x_q) / (\Phi_1^U(x_q)\pi_1(x_q) + \Phi_2^L(x_q)\pi_2(x_q)).$ <p>if <math>p_1^L(x_q) \geq 0.5</math>, <math>c(x_q) := C_1</math>. if <math>p_1^U(x_q) &lt; 0.5</math>, <math>c(x_q) := C_2</math>.  if <math>c(x_q) \neq \text{"?"}</math>, <math>M(Q) += 1</math>.</p> $\phi_1^L(Q) := \min_{x_q \in Q} \phi_1(x_q). \phi_2^L(Q) := \min_{x_q \in Q} \phi_2(x_q).$ $\phi_1^U(Q) := \max_{x_q \in Q} \phi_1(x_q). \phi_2^U(Q) := \max_{x_q \in Q} \phi_2(x_q).$
--

### 3 Quasar Discovery

Quasars are star-like objects that are not very well understood, yet they play a critical role in cosmology. As the most luminous objects in the universe, they can be used as markers of the mass in the very distant (and thus very early) universe. With the very recent advent of massive sky surveys such as the Sloan Digital Sky Survey (SDSS), it is now conceivable in principle to obtain a catalog of the locations of quasars which is more comprehensive than ever before, both in sky coverage and depth (distance). Such a catalog would open the door to numerous powerful analyses of the early/distant universe which have so far been impossible. A central challenge of this activity is the question of how to use the limited information we have in hand (a tiny set of known, nearby quasars) to extract a massive amount of more subtle information from the SDSS dataset (a large set of faint quasar candidates). In recent work with our physicist collaborators [5] which we only summarize here, we demonstrated the use of a classification approach using KDA to obtain a catalog of predicted quasars. To achieve its results, that work used an earlier version of the algorithm described here that did not allow data-dependent priors, but the algorithm itself has not been previously described in a refereed publication until now.

**Computational Experiments.** To give an idea of the computational behavior of the algorithm, for constant priors, we measured the CPU time for 143 leave-one-out training runs for a 50K subset of the training data, and the CPU time for 38 runs for the entire 500K dataset, representing searches for the optimal bandwidths. The data consists of 4-D color measurements.



**Fig. 2.** CPU times (left) and leave-one-out accuracy scores (weighting both classes equally) (right) as a function of the bandwidths, for the 50K subset. Note that accuracy and running time rise very steeply in about the same place, though running time can also be high in regions where accuracy drops off.

The priors consisted of estimated class proportions based on astrophysical knowledge. The CPU time depends heavily on the bandwidth parameters, as shown in Figure 2. The sum over all the 143 runs for the 50K dataset was estimated at 16,100 seconds for the naïve algorithm and measured at 7,067 seconds for the proposed algorithm. The sum over the 38 runs of the 500K dataset was estimated at 1,030,000 seconds for the naïve algorithm and measured at 38,760 seconds for the proposed algorithm. For the 50K dataset, the average run took the naïve method 175 seconds and the proposed method 77 seconds. For the 500K dataset, the average was 27,000 seconds for naïve and 1,020 seconds for the new method. This gives some indication of the sub-quadratic scaling of the algorithm.

**Results.** Using the algorithm described, we were able to tractably estimate (find optimal parameters for) a classifier based on a large training set consisting of 500K star-labeled objects and 16K quasar-labeled objects, and predict the label for 800K faint (up to  $g = 21$ ) query objects from 2099  $\text{deg}^2$  of the SDSS DR1 imaging dataset. Of these, 100K were predicted to be quasars, forming our catalog of quasar candidates. This significantly exceeds the size, faintness, and sky coverage of the largest quasar catalogs to date. Based on spectroscopic hand-validation of a subset of the candidates, we estimate that 95.0% are truly quasars, and that we have identified 94.7% of the actual quasars. These efficiency and completeness numbers far exceed those any previous catalog, making our catalog both the most comprehensive and most accurate to date. The recent empirical confirmation of cosmic magnification [6], a prediction of relativity theory, using our catalog is an example of the scientific possibilities opened up by this work. In ongoing efforts we are exploring ways to make the method both more computationally and statistically efficient, with the goal of obtaining all 1.6M quasars we estimate are detectable in principle from the entire SDSS dataset. To accomplish this we

will need to be able to train and test with tens or hundreds of millions of points.

## 4 Conclusion and Discussion

We have presented an exact computational method for training a kernel discriminant and predicting with it which can allow KDA to scale to datasets which might not otherwise have been tractable, the first such method to our knowledge. The method is also applicable in principle to other classifiers which have similar forms in the prediction stage, such as support vector machines with Gaussian kernels. Though we considered only the 2-class problem, the methodology extends in principle to any number of classes. A recently suggested alternative to leave-one-out cross-validation [1] may also be amenable to this algorithmic approach.

One question not answered by the algorithm presented here is how to compute the full class probability estimates efficiently (only bounds on them are provided). If desired after the optimal bandwidths are found, they can be computed using the fast approximate methods described in [2], which were demonstrated to be faster in general than alternatives such as the FFT [7].

A disadvantage of the approach in this paper is its reliance on structures like *kd*-trees, whose efficiency decreases as the dimension of the data grows. Another is that such data structures are very difficult to analyze, making it impossible for us to state the asymptotic runtime growth for this algorithm at this point.

## References

1. Ghosh, A. K. and Chaudhuri, P., Optimal Smoothing in Kernel Discriminant Analysis, *Statistica Sinica* 14, 457–483, 2004.
2. Lee, D., Gray, A. G., and Moore, A. W., Dual-tree Fast Gauss Transforms, *Neural Information Processing Systems*, 2005.
3. Preparata, F. P. and Shamos, M., *Computational Geometry*, Springer-Verlag, 1985.
4. Rao, C. R., *Linear Statistical Inference*, Wiley, 1973.
5. Richards, G., Nichol, R., Gray, A. G., *et. al*, Efficient Photometric Selection of Quasars from the Sloan Digital Sky Survey: 100,000  $z > 3$  Quasars from Data Release One, *Astrophysical Journal Supplement Series* 155, 2, 257–269, 2004.
6. Scranton, R., *et. al*, Detection of Cosmic Magnification with the Sloan Digital Sky Survey, *Astrophysical Journal* 633, 2, 589–602, 2005. Described in *Nature*, April 27, 2005.
7. Silverman, B. W., Kernel Density Estimation using the Fast Fourier Transform, *Applied Statistics* 33, Royal Statistical Society, 1982.
8. Silverman, B. W., *Density Estimation for Statistics and Data Analysis*, Chapman and Hall/CRC, 1986.