Error Minimizing Algorithms for Nearest Neighbor Classifiers

Reid B. Porter[a] and Don Hush[a] and G. Beate Zimmer[b]

[a]Space and Remote Sensing Sciences Group,
Los Alamos National Lab, Los Alamos, NM 87544, USA.
[b]Dept. of Mathematics and Statistics, Texas A&M University–Corpus Christi,
6300 Ocean Drive, Corpus Christi, TX 78412-5825, USA.

ABSTRACT

Stack Filters define a large class of discrete nonlinear filter first introduced in image and signal processing for noise removal. In recent years we have suggested their application to classification problems, and investigated their relationship to other types of discrete classifiers such as Decision Trees. In this paper we focus on a continuous domain version of Stack Filter Classifiers which we call Ordered Hypothesis Machines (OHM), and investigate their relationship to Nearest Neighbor classifiers. We show that OHM classifiers provide a novel framework in which to *train* Nearest Neighbor type classifiers by minimizing empirical error based loss functions. We use the framework to investigate a new cost sensitive loss function that allows us to train a Nearest Neighbor type classifier for low false alarm rate applications. We report results on both synthetic data and real-world image data.

Keywords: Stack Filters, Classification, Nearest Neighbor

1. INTRODUCTION

Nearest Neighbor classifiers define a family of pattern classifiers that are both simple to understand and implement. Much of the early work in Nearest Neighbor methods was inspired by the proof that with infinite samples the performance of the Nearest Neighbor method is no worse than twice the Bayes error.[1] At the same time, it was also observed that the Nearest Neighbor approach assumes the class conditional densities are constant within local neighborhoods of training samples and that this can lead to high bias (and poor performance) in problems with high dimensions. A large number of extensions and modifications to Nearest Neighbor classifiers are motivated by this observation.[2] We point out that this assumption is also key to a number of the advantages associated with Nearest Neighbor approaches. Specifically, the extension to multi-class classification becomes straightforward, and the implementation and application of the classifier to future data can be highly optimized.

Another unique characteristic of Nearest Neighbor classifiers is that training does not involve fitting a model to the training data. While this has advantages (such as simple implementation), it also has disadvantages which limits the applicability of Nearest Neighbor methods. Specifically, there is no direct link to empirical error minimization which means there is no consistent way to control classifier complexity, compress the training set and/or design Nearest Neighbor classifiers for cost sensitive problems. In this paper we present a new classification framework based on Stack Filters that directly minimizes empirical error based loss functions under the Nearest Neighbor assumption that class conditional densities are locally constant. This leads to a new Nearest Neighbor variant that inherits many of characteristics of traditional Nearest Neighbor classifiers, and also many of the advantages of empirical error minimizing classifiers.

In the next section, we provide a more formal summary of the main results. In Section 3 we summarize Stack Filter Classifier framework and in Section 4 describe how it relates to Nearest Neighbor classifiers with the help of synthetic experiments. In Section 5 we make use of error minimization to design Nearest Neighbor classifiers specifically for the low False Alarm rate regime in several practical applications.

E-mail: rporter@lanl.gov, Telephone: (505) 665 6117

2. SUMMARY

In two-class classification we are given a training set of N points, $x \in \mathbb{R}^D$, with labels, $y \in \{-1, 1\}$, drawn at random according to a probability distribution $P_{X,Y}$. The task is to find a decision function $F : \mathbb{R}^D \rightarrow \mathbb{R}$ that has small error: $e(F) = E_{X,Y}(\mathbf{I}(sgn(F(x)) \neq y))$. $P_{X,Y}$ is unknown and we therefore seek a decision function that minimizes an empirical estimate, known as the misclassification risk:

$$\hat{e}(F) = \sum_{n=1}^{N} (\mathbf{I}(sgn(F(x(n))) \neq y(n)). \tag{1}$$

Consider the memory based function class defined by the Nearest Neighbor classifier:

$$n^* = \underset{n}{\text{argmin}}\{d(x(n), x)\} \tag{2}$$

$$\hat{F}(x) = y(n^*) \tag{3}$$

where $d(x(n), x)$ is the distance between a test point x and a training sample $x(n)$. We observe that $\hat{F}(x)$ minimizes Equation 1, and assuming there are no class conflicts in the training set, it obtains zero training error. Of course the classifiers real performance $(e(F))$ in the finite sample case is unknown. Part of the problem is that Nearest Neighbor classifiers do not belong to any obvious function class, and therefore it is difficult to characterize class complexity. In this paper we present a new function class which provides good control on class complexity and leads to Nearest Neighbor like classifiers. Specifically, our approach finds solutions of the form:

$$n^* = \underset{n}{\text{argmin}}\{d(x(n), x) - c(n)\} \tag{4}$$

$$\hat{F}_\gamma(x) = y(n^*) \tag{5}$$

where $c(n)$ is a sample dependent constant. The constants are found by solving one of a number of optimization problems which select empirical error minimizers from a nestled set of function classes indexed by γ:

$$\hat{F}_\gamma(x) = \arg\min_{F \in \mathcal{F}_\gamma} \hat{e}(F) \tag{6}$$

where $\mathcal{F}_\gamma \subseteq \ldots \subseteq \mathcal{F}_1 \subseteq \mathcal{F}$. At $\gamma = 0$ the classifier obtains zero error on any (non-conflicted) training set, $c(n)$ are equal for all samples, and the classifier is the standard Nearest Neighbor classifier. At $\gamma = \infty$ the size of the function class is two and $c(n)$ will be assigned in such a way that samples from one class will always be closer to any given test point. That is, we can throw away all samples and simply keep the label of the majority class. For intermediate values of γ, the approach provides a systematic way to trade empirical error with class complexity where increasing numbers of training samples, which do not contribute to the final Nearest Neighbor classifier, can be thrown away.

3. APPROACH

3.1 Stack Filter Classifiers

Stack Filters include the Median, Order Statistics, and Weighted Order Statistics as sub-classes.[3] Stack Filters are uniquely defined by a positive (or monotone) Boolean function and therefore represent a discrete, or finite function class. Given a real valued input vector $x = (x_1, x_2, \ldots, x_D)$ the Stack Filter will select one of the input components as its output value. In two dimensions, the only nontrivial Stack Filters are the minimum and maximum functions. To increase the expressive power of the function class Generalized Stack Filters expand the input space by adding T monotonically increasing and evenly spaced constant offsets (t_i) to each component:

$$xt = [\{x_1 - t_1, x_1 - t2, \ldots, x_1 - t_T\}, \ldots, \{x_D - t_1, x_D - t2, \ldots, x_D - t_T\}] \tag{7}$$

In addition, since Stack Filters are constrained to a monotonic function class it is also common to mirror the expanded input vector: $xm = [xt, -xt]$. The monotonicity constraints imply that a Stack Filter, $F(xm)$, commutes with thresholding:

$$F(xm) \geq t \Longleftrightarrow f(xm \succeq t) = 1 \tag{8}$$

where $\succeq$ is a thresholding function parameterized by a scalar t that produces a binary vector with components $xb_i = \mathbf{1}_{\{xm_i \geq t\}}$. Equation 8 tells us that a thresholded Stack Filter, $F(xm) \geq 0$, reduces to positive Boolean function $f(xb)$ applied to thresholded inputs $xb = xm \geq 0$. The fact that Stack Filters commute with thresholding is also key to efficiently finding a Stack Filter minimizer for Mean Absolute Error. The property can also be exploited to optimize classification risk functions. To find the Stack Filter which minimizes zero-one or misclassification risk (Equation 1), we define a partially specified Boolean function where we assign class labels to the rows of a look-up table that appear in the (input expanded) training set thresholded at zero. Since the xb associated with each training sample has Hamming weight DT, there can be no violation of monotonicity constraints, and the look-up table is guaranteed to be a positive Boolean function.

Note, that if we choose a sufficient number of thresholds during input expansion (and assuming training samples are unique), training samples fall into unique partitions, and the positive Boolean function obtains zero error on the training set. However, the corresponding look-up table is extremely sparse, and a large fraction of the input space remains undecided.

In previous work we suggested various ways to infer undecided entries in the Stack Filter look-up table by choosing different risk functions.[4] We found that large margin risk functions, which require the Stack Filter output to be further away from zero ($yF(xm) > t$), have both practical and theoretical interest. It led to the notion of rank-order margin where we require the Stack Filter to select an input component that is γ samples greater than the median for class 1, and γ samples less than the median for class -1. This also means the positive Boolean function is applied to different inputs ($xb = xm \geq t$ for class 1). Each training vector has less 1's for class 1 and more 1's for class 0, which increases the number of monotonicity constraints that must be satisfied as margin is increased. Geometrically, training samples correspond to tiles of increasing size in the input space and the monotonicity constraints require that tiles from different classes do not overlap.

More formally, to find a Stack Filter that minimizes misclassification risk we use the same integer linear program that is used to optimize Stack Filters under Mean Absolute Error. We associate a binary variable $z(n)$ to each training sample. This variable determines whether the sample is kept or not: 1 = keep, 0 = discard. Our objective is to maximize the sum (keep as many samples/tiles as possible) subject to the constraint that tiles associated with samples from different classes are not allowed to overlap.

$$\begin{array}{cc} \text{maximize} & \sum_{n=1}^{N} z(n) \\ \text{subject to} & z(n) + z(m) \leq 1 \text{ if } \Delta^{\gamma}_{n,m} > 0 \text{ and } y(n) > y(m) \\ \text{and} & z(n) \in \{0,1\} \; \forall n \in \{1, \ldots N\} \end{array} \tag{9}$$

where $\Delta^{\gamma}_{n,m}$ is greater than zero when tiles associated with $x(n)$ and $x(m)$ overlap. The size (and shape) of tiles are determined by the value of margin γ and the thresholds used in the input expansion in Equation 7. For a more detailed example and further discussion of this topic see.[4] On the left of Figure 1 we show an example of this optimization problem. There are four training samples in total indicated by crosses. Class 1 tiles at margin γ are drawn with a solid line. Class -1 tiles at margin γ are drawn with a dotted line. Tiles from different classes that overlap (tiles that intersect the shaded area) introduce constraints into the linear program. In the middle panel of Figure 1 we show a hypothetical solution to Equation 9 where two tiles have been removed to satisfy the constraints.

In previous work we also suggested an alterative to the large margin misclassification risk function for Stack Filter Classifiers. We call this risk function, the large margin hinge risk, and it is analogous to the hinge risk used in SVMs:

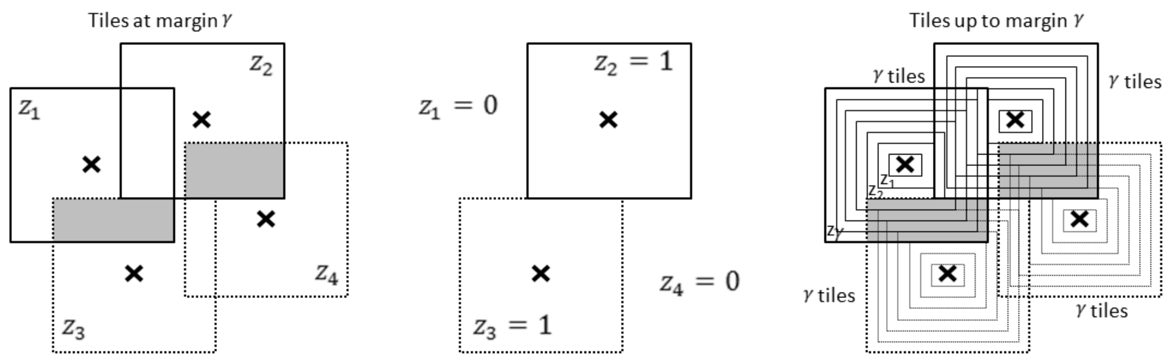


Figure 1. Geometric view of Stack Filter Classifier training. Left) The objective function aims to maximize the number of non-overlapping tiles kept. Middle) A hypothetical solution to zero-one risk where overlapping tiles have been removed. Right) The training problem that minimizes hinge risk.

$$\hat{e}(F) = \sum_{n=1}^{N} (\gamma - y(n)F(x(n)))_+ \tag{10}$$

In this case, we keep all the tiles associated with a training sample as the margin is increased up to γ. The optimization problem is identical to Equation 9, but we have $N\gamma$ variables instead of N. This new optimization problem is shown on the right in Figure 1. The hinge risk has a number of advantages over misclassification risk for Stack Filter classifier design, however the optimization problem depends on γ, which depends on the size of the input expansion. In order to make Stack Filter Classifiers as general purpose as possible, we would like the size of the input expansion to be very large, and this means that Equation 9 gets expensive quickly. In the next section we introduce a continuous domain version of Stack Filter Classifiers which we call Ordered Hypothesis Machines (OHM), which provides an efficient solution to this problem.

3.2 Ordered Hypothesis Machines

Due to the monotonicity constraints, the tile associated with training samples in Stack Filter Classifiers form an increasing set as margin is increased. A simple, and convenient way to parameterize these increasing sets of tiles is to to imagine letting the distance between thresholds to shrink to zero (or conversely the number of thresholds in Equation 7 to grow to infinity). The margin parameter becomes continuous and the overlap between tiles can be calculated directly:

$$\begin{gathered} \Delta^{\gamma}_{n,m} = \max(2\gamma - d(n,m), 0) \\ \text{where } d(n,m) = \|x(n) - x(m)\|_p \end{gathered} \tag{11}$$

When $p = \infty$, the distance function $\|\|_\infty$ defines a sequence of square tiles analogous to the expansion used in Equation 7. In this paper we will use spherical tiles, based on $p = 2$: $d(n,m) = \|x(n) - x(m)\|_2$.

Finding Ordered Hypothesis Machine (OHM) Classifiers that minimize zero-one risk is identical to finding a Stack Filter Classifier classifier that minimizes zero-one risk except that γ indexes tiles differently. The problem is illustrated on the left in Figure 2 and the optimization problem in Equation 9 is used to find the hypothetical solution shown in the middle panel where monotonicity constraints have been satisfied.

The advantage of Ordered Hypothesis Machines becomes more obvious when we consider minimizing the hinge risk in Equation 10. In this case we replace the binary variables $z(n)$ with real valued variables $v(n)$, that represent the number of tiles associated with each training sample. The generalization of the linear program is straightforward:

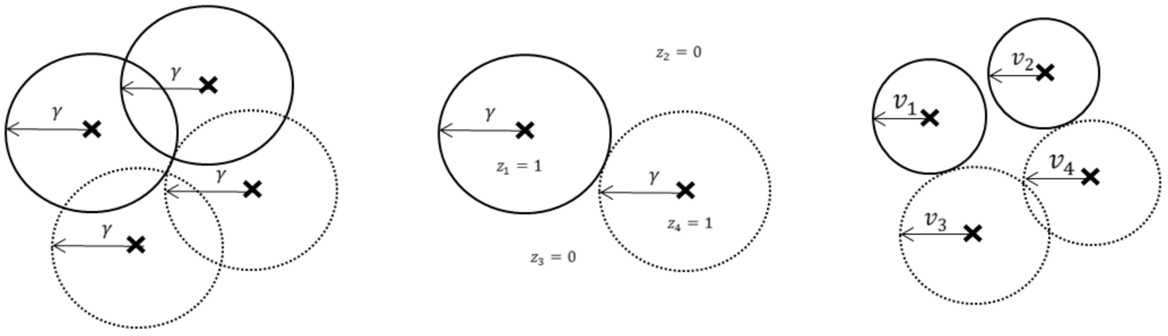


Figure 2. Geometric view of OHM Classifier training. Left) Tiles of size γ compete to cover training samples. Middle) A hypothetical minimizer of zero-one risk maximizes the number of non-overlapping tiles. Right) A hypothetical minimizer of hinge risk maximizes the size of tiles up to size γ.

$$\begin{array}{cc} \text{maximize} & \sum_{n=1}^{N} v(n) \\ \text{subject to} & v(n) + v(m) \leq 4\gamma - \Delta^{\gamma}_{n,m} \text{ if } y(n) > y(m) \\ \text{and} & 0 \leq v(n) \leq 2\gamma \ \forall n \in \{1, \dots N\} \end{array} \tag{12}$$

where $\Delta^{\gamma}_{n,m}$ is defined by Equation 11 and γ is a free parameter. Geometrically, we try to maximize the size of tiles, centered on training samples, up to a maximum radius of γ. In addition, tiles with similar labels can overlap, but tiles with different labels must not overlap. On the right in Figure 2 we show a notional result where the size of the tiles have been chosen to satisfy the monotonicity constraints.

Unlike the Stack Filter Classifier, the OHM formulation of hinge risk does not increase the number of variables in the problem. In the Stack Filter problem, the variables are integer $z \in 0, 1$ and we solve a linear program relaxation with a uni-modular constraint matrix. This means we can threshold the real-valued variables found with the linear program to obtain the exact integer solution. In the OHM formulation we solve the same linear program but the constraint matrix is no longer uni-modular (since we introduce non binary values into the right hand side of the constraint equations) and we use the real-valued variables found by the linear program directly as the tile radii.

4. RELATIONSHIP TO NEAREST NEIGHBOR CLASSIFIERS

After training OHM classifiers we have a set of tiles associated with training samples. To apply this classifier to a test point, we need to test if the point falls within a tile. Due to the monotonicity constraints, the point is guaranteed to fall in at most one tile. However, the point may also fall outside of all tiles in which case it's value is undetermined. Note that as the dimension of the problem increases, it is more and more likely that a test point will be undetermined. Various schemes for assigning labels to undetermined points are possible (e.g. assign label to the class which largest prior probability) that may or may not work better in different applications. We suggest Nearest Neighbor type classifiers are particularly appropriate for OHM classifiers since in our formulation, finding the nearest neighbor is equivalent to finding the nearest tile.

For example, in the case of zero-one risk, an OHM classifier is defined by a set of the tiles centered on a subset of the training samples and all tiles have the same size. To test if a point is within a tile, we implement a Nearest Neighbor type search and determine if the distance is within γ of the training sample. At $\gamma = 0$ all (nontrivial) test points are undecided and if we are using the Nearest Neighbor classifier to assign labels, then the OHM classifier and the Nearest Neighbor classifier are identical. As γ is increased, OHM classifiers will discard more and more training samples. In this case, applying the OHM classifier is identical to applying the Nearest Neighbor classifier to the reduced training set. A natural extension to this approach, is to use K-Nearest Neighbor rules to assign labels to undecided points. The relationship between the risk function and the final classifier is less clear in this case, however we investigate the relationship in our experiments.

In the case of the hinge risk OHM classifier also derives a set of tiles centered on a subset of the training samples. However in this case each tile can potentially have a different size. As defined by Equation 4, the classifier is implemented through a modified nearest neighbor rule where distances to the (remaining) training samples are translated by a sample specific constant. To apply this classifier to a test point, we simply find the Nearest Neighbor using the modified distance measure. Again, K-Nearest Neighbor rules could also be used with these modified distances, and we investigate the approach in our experiments.

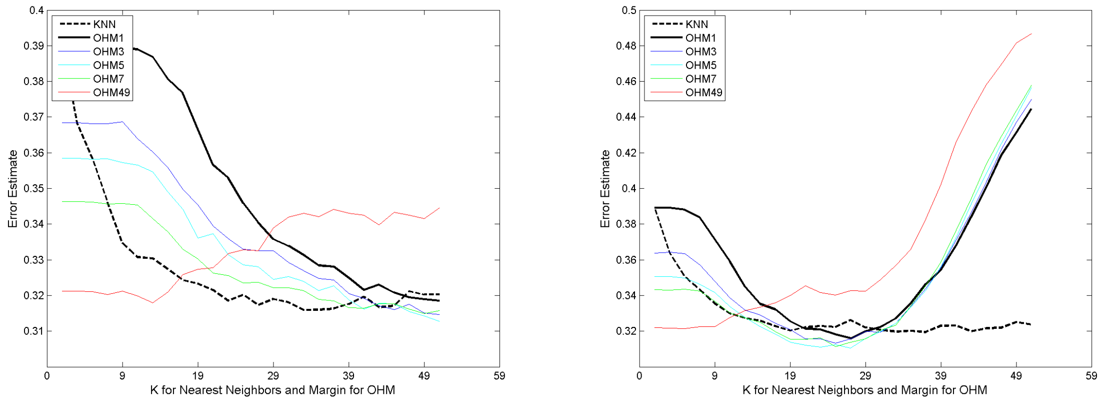


Figure 3. Comparison of Left) misclassification and Right) hinge risk compared to K-Nearest Neighbors on 4 dimensional Gaussian data.

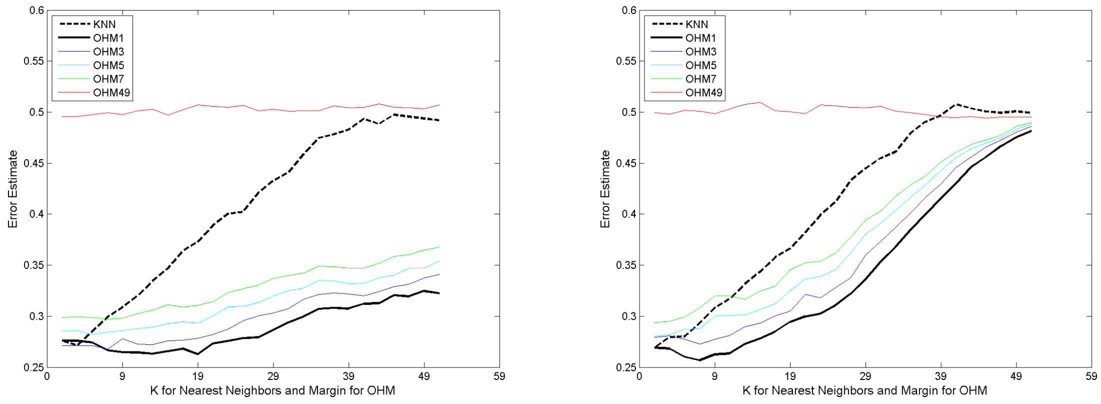


Figure 4. Comparison of Left) misclassification and Right) hinge risk compared to K-Nearest Neighbors on 4 dimensional chi-squared data.

We investigate the relationship between OHM classifiers and Nearest Neighbor classifiers with synthetic experiments. For the first dataset used in Figure 3, we chose overlapping independent normal random values in 4 dimensions: $\mu_1 = \vec{0}, \Sigma_1 = \mathbf{I}$ and $\mu_{-1} = \vec{0.5}, \Sigma_{-1} = 0.75\mathbf{I}$. For the dataset used in Figure 4, we used a Chi-square distribution with 3 degrees of freedom. In each of 4 dimensions the samples of class 1 are of the form $10 - X$ and for class -1 samples are of the form $5 + X$. We used 100 samples in training and 1000 samples to estimate performance, and we average the results over 100 trials.

In both figures we compare standard Nearest Neighbor classifiers to OHM classifiers. On the x-axis we vary a free parameter and on the y- axis we show the estimated performance. For Nearest Neighbors the free parameter

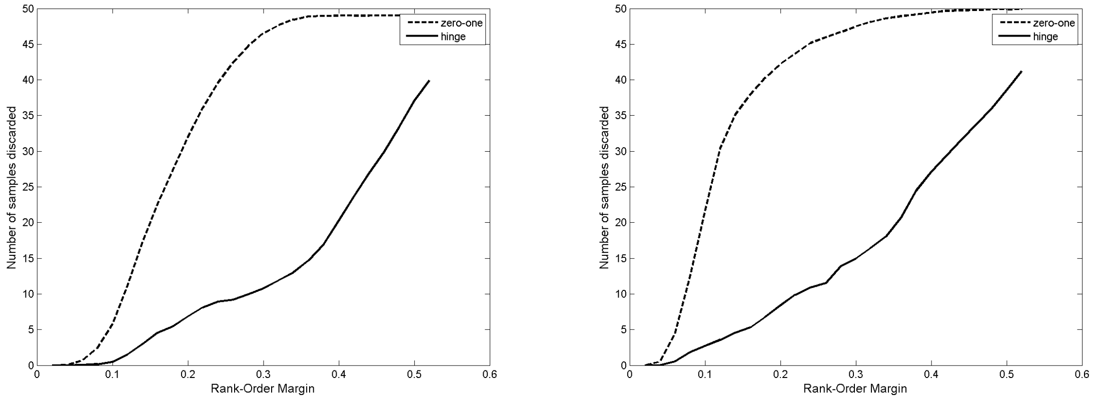


Figure 5. Number of samples discarded (on average) as γ is varied for Left) The Gaussian data used in Figure 3 and Right) The chisquared data used in Figure 4.

is K, and for OHM it is γ. We show multiple results for OHM, each corresponds to a different value of K used when applying the classifier. Note that K is not used in OHM training. Also note, that the performance of K Nearest Neighbor and OHM at $\gamma = 0$ for the various values of K, should be identical.

Several general observations can be made about Figures 3 and 4. For Gaussian data we observe that increasing K increases performance for Nearest Neighbor methods, and for Chi-squared data increasing K decreases performance.

When Nearest Neighbors is compared to OHM classifiers optimized under zero-one risk, we observe that the best performance obtained by OHM is a value of K for which $\gamma = 0$. For the Gaussian data this occurs at the maximum $K = 49$ and for Chi-square data this occurs at the minimum $K = 1$. In contrast when Nearest Neighbors is compared to OHM classifiers optimized under the hinge risk there is a non-zero value of γ for which the OHM classifier has slightly better performance than the best Nearest Neighbor classifier. In the Gaussian case, OHM with $K = 5$ and $\gamma = 13$ obtains similar performance to NN with $K = 49$. And in the Chi-square case, OHM with $K = 1$ and $\gamma = 5$ outperforms NN with $K = 1$.

Perhaps more important than the improvement in performance obtained by OHM classifiers optimized using hinge risk is the reduced cost associated with implementing OHM classifiers in comparison to the traditional Nearest Neighbor classifier. In Figure 5 we show the number of training samples discarded when minimizing zero-one and hinge risk for the Gaussian (left) and Chi-square (right) problems. By matching the margin of best hinge loss solutions we observe that close to 10 samples were discarded in the Gaussian problem, and approximately 5 samples in the Chi-square problem. In addition for the Gaussian problem, we observe that OHM can obtain similar performance to NN with a much smaller K. For example, the best OHM classifier for $K = 1$ does nearly as well as the $K = 49$ NN classifier, and at $K = 5$ it does better.

5. COST-SENSITIVE CLASSIFICATION

In many practical applications, we would like to treat the errors associated with different classes in different ways, e.g. many applications require a very low false alarm rate. In some cases this can be achieved by simply adjusting an output threshold associated with a pre-trained classifier. However, in Nearest Neighbor type classifiers, and in general, this is not possible. In addition, the class membership proportions represented in the training set are often different to the class membership proportions that are desired in the final application. What we would like is a training algorithm that can use all the training data available and also target a desired false alarm rate directly. Several general purpose methods have been proposed for designing cost insensitive classifiers (such as Nearest Neighbors) in a cost sensitive way.[5] In this paper we approach the problem by considering alternative risk functions for OHM classifiers.

Specifically, we investigate a one sided risk function that approximates a Neyman Pearson design criteria: maximize the detection rate subject to the false alarm rate being $\leq \alpha$, where α is a user specified value. The approximation arises from our use of hinge risk instead of zero-one risk. We associate N variables $v(n)$ with training samples from class 1 (the target class) and M variables $v(m)$ associated with training samples from class -1. The one sided linear program is:

$$\begin{array}{cc} \text{maximize} & \sum_{n=1}^{N} v(n) \\ \text{subject to} & \sum_{m=1}^{M} v(m) \geq 2\gamma * M * (1-\alpha) \\ \text{and} & v(n) + v(m) \leq 4\gamma - \Delta_{n,m}^{\gamma} \\ \text{and} & 0 \leq v(n), v(m) \leq 2\gamma \end{array} \tag{13}$$

5.1 Anomaly Detection

Anomaly detection is a general purpose application where there is often an upper limit on the amount of work, or the number of candidate anomalies, that a human user can inspect. On the left in Figure 6 we show a two dimensional synthetic experiment which captures some of the key ingredients of the anomaly detection application. The data is drawn from a Gaussian distribution. We label all samples that fall outside of the 0.15 probability level-set as *anomalies* and label the samples that fall inside as *normal*. The task is to build a classifier to approximate the level-set and identify the anomalous samples using only the data (i.e. sample labels are not provided to the learning algorithm).

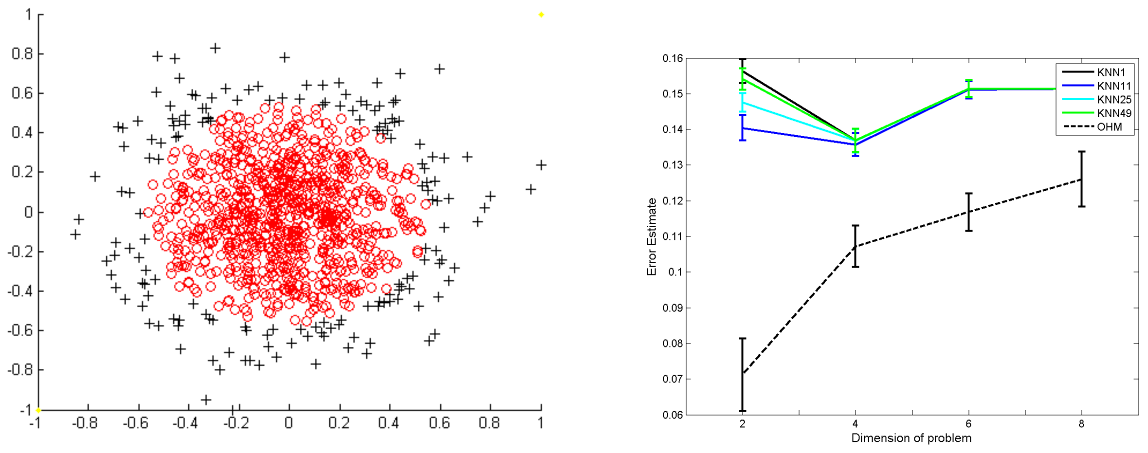


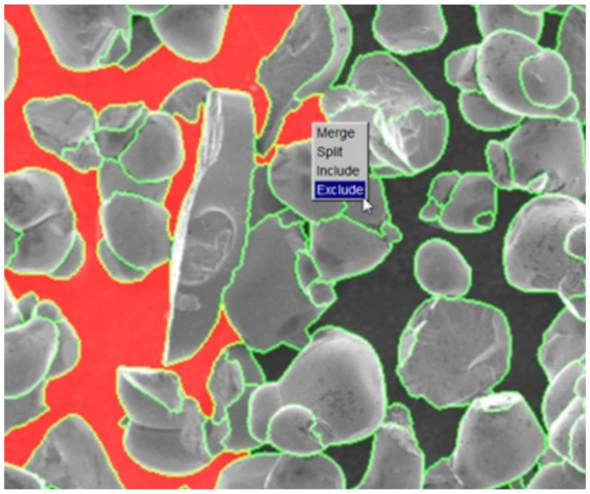
Figure 6. Left) The 0.15 probability level set (crosses) of a Gaussian distribution and Right) Performance as dimension increases.

A solution method for anomaly detection which employed a surrogate classification problem was presented in.[6] The unlabeled data is assigned a label of -1, and samples for class +1 are artificially generated based on our prior knowledge of anomalies. Lacking prior knowledge, a natural choice is to draw samples from a uniform distribution. In our experiments we draw the same number of samples from the uniform distribution as we have in the original data. We then train two different types of classifier.

For K-Nearest Neighbors we apply the standard algorithm on a sub-sampled training set. That is, we reduce the number of anomalous samples to 0.15, our expected alarm rate, and we use an artificially generated validation set to select the best classifier for various values of K. For OHM we use Equation 13 and use an artificially generated validation set we choose the values of γ and α. On the right in Figure 6 we show the average performance over 20 runs as the dimension of the problem is increased. OHM using just one Nearest Neighbor is able to clearly outperform the naive KNN methods in this problem.

5.2 Hemi-Supervised Learning

Another type of application where cost-sensitive classification becomes important is hemi-supervised learning. In hemi-supervised learning we are given examples and class labels for one class (often the target class) as well as a number (often much larger) of unlabeled samples. The unlabeled samples are assumed to contain both examples of the target class as well as a background class. In previous work we showed how this problem could also be cast as a surrogate classification problem[7] and developed a solution method with Support Vector Machines (SVMs). In recent work[8] (also reported at this conference) we describe an image processing application where hemi-supervised learning proved useful. In summary, we have a segmented image and a user begins to identify segmented regions which belong to the background. We would like to use machine learning to predict the user's future selections, thereby reducing the amount of work required by the user. An example image and segmentation used in the experiment is shown on the left in Figure 7.



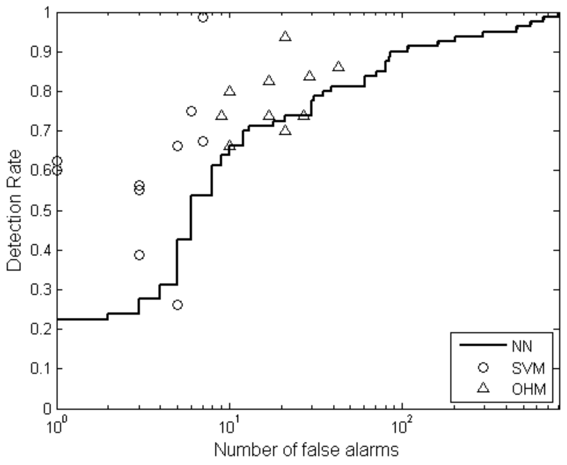


Figure 7. Left) Example imagery showing the regions that must be classified as either background or foreground. Right) The performance of the various machine learning approaches to this problem.

We compare three different classifiers for this problem. *1) NN:* We use a one-sided variant of Nearest Neighbor where we measure the minimum distance to a training example. We threshold this distance at various values to produce a Receiver-Operator Characteristics Curve (ROC-Curve). *2) SVM:* We train a SVM using the hemi-supervised method[7] and we obtain a classifier whose performance can be represented as a point in the ROC space - circles in Figure 7. Details of the SVM parameters are provided in.[8] *3) OHM:* We train a classifier using Equation 13. For the purposes of comparison we choose γ and α such that the alarm rate of the classifier is approximately equal to the target class probability. In practice this quantity may not be known and this is a topic of future work. Results are represented as triangles in Figure 7.

The experiment involved 5 training segments, and we observed a large variance in the performance of all algorithms. By including the unlabelled data in the optimization OHM was able to outperform the Nearest Neighbor approach in some parts of the ROC curve. In general, the parameter settings for the SVM favored lower false alarm rates compared to OHM which made comparison difficult, although we expect the SVM would outperform OHM in general since it does not enforce piecewise constant class conditional densities.

6. SUMMARY

We have presented a novel framework to produce Nearest-Neighbor like classifiers that optimize empirical error based risk functions. This approach can be used to optimize alternatives to misclassification error that may be better suited for particular applications, e.g. anomaly detection and hemi-supervised learning. We suggest future work in this area should investigate the relationship between OHM and K-Nearest Neighbors where K is greater than 1. As shown in experiments, the smoothing effect of K Nearest Neighbors improves performance in some problems, but the relationship to risk minimization is unclear. Formalizing this relationship may shed light on where and when K-Nearest Neighbors is successful and also suggest new ways to relax the constant class conditional density assumption.

ACKNOWLEDGMENTS

We gratefully acknowledge the support of the U.S. Department of Energy through the LANL/LDRD Program for this work

REFERENCES

[1] Cover, T. and Hart, P., "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on* **13**, 21 – 27 (Jan. 1967).

[2] Hastie, T. and Tibshirani, R., "Discriminant adaptive nearest neighbor classification," *IEEE Trans. Pattern Anal. Mach. Intell.* **18**, 607–616 (June 1996).

[3] Wendt, P., Coyle, E., and Gallagher, N., "Stack filters," *IEEE Trans. on Acoustics, Speech, and Signal Processing* **34**, 898–910 (1986).

[4] Porter, R. B., Zimmer, G. B., and Hush, D., "Stack filter classifiers," in [*Mathematical Morphology amd Its Applications to Signal and Image Porcessing, 9th International Symposium, ISMM 2009, Groningen, The Netherlands, August 24-27, 2009 Proceedings*], Wilkinson, M. F. and Roerdink, J., eds., *Lecture Notes in Computer Science* **5720**, 282–294, Springer (2009).

[5] Domingos, P., "Metacost: A general method for making classifiers cost-sensitive," in [*In Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*], 155–164, ACM Press (1999).

[6] Steinwart, I., Hush, D., and Scovel, C., "A classification framework for anomaly detection," *J. Mach. Learn. Res.* **6**, 211–232 (December 2005).

[7] Porter, R., Ruggiero, C., and Hush, D., "Density-based similarity measures for content based search," in [*Proceedings of the 43rd Asilomar conference on Signals, systems and computers*], *Asilomar'09*, 390–394, IEEE Press, Piscataway, NJ, USA (2009).

[8] Porter, R., Ruggiero, C., Hush, D., Harvey, N., Kelly, P., Scoggins, W., and Tandon, L., "Interactive image quantification tools in nuclear material forensics," in [*Proceedings of IS&T/SPIE Electronic Imaging*], SPIE (2011).