Incremental Learning With Sample Queries

Joel Ratsaby

Abstract—The classical theory of pattern recognition assumes labeled examples appear according to unknown underlying class conditional probability distributions where the pattern classes are picked randomly in a passive manner according to their a priori probabilities. This paper presents experimental results for an incremental nearest-neighbor learning algorithm which actively selects samples from different pattern classes according to a querying rule as opposed to the a priori probabilities. The amount of improvement of this query-based approach over the passive batch approach depends on the complexity of the Bayes rule.

Index Terms—Incremental learning, sample querying, nearest-neighbor algorithm, active learning, model selection.

1 Introduction

We consider the general problem of learning multicategory classification from labeled examples. In many practical learning settings, the time or sample size available for training are limited which may have adverse effects on the accuracy of the resulting classifier, for instance, learning to recognize handwritten characters. In case of a limited sample size, it is of primary importance to make the learning more efficient by obtaining specific training samples which contain high amount of information about the separability of the pattern classes. Researchers in the area known as *active learning* have considered various such ways which usually require some form of extra interaction between the learner and teacher which permits the learner to actively select or query only for interesting training samples. The on-line mode of learning is typically more useful in such cases than the batch mode since the choice of which examples to query for is often based on the performance of the learner on the examples obtained so far. Some previous work in this direction include querying for correct classification labels of points in the feature space (cf. Angluin [1], Rivest and Eisenberg [13]), the notion of selective sampling (cf. Cohn, et al. [4], Cohn [3]) where regions of high classification-uncertainty are identified and from which labeled samples are randomly drawn.

In pattern classification problems, it is often not useful or possible to query for the correct classification of vectors in the feature space. For instance, in handwritten recognition problem the computer (the learner) could ask the user for labels of patterns generated by the computer, however, such information is not useful since the labels provided by the user are not necessarily representative of his handwriting style. His handwriting is characterized by pattern-class conditional probability distributions which may be weakly related to his ability to recognize letter patterns generated by the computer. It is, however, possible to let the learner select particular pattern classes, not necessarily according to their a priori probabilities, and then obtain randomly drawn patterns according to the underlying unknown class-conditional probability distribution. We refer to such selective sampling as *sample querying*. It is not immediate though, whether the freedom to select different classes at any time during the training stage is beneficial to the accuracy of the classifier learned. Recent theory (cf. Ratsaby [12]) indicates that such sample querying can be an effective means of optimizing the learning accuracy for problems of zero Bayes error and where the Bayes classifier is assumed to be contained in the family of learnable classifiers. It is conjectured that even in the case of nonzero Bayes loss, sample querying can improve the learning rates.

In the current paper, we report on experimental results of an incremental algorithm which utilizes the sample-querying procedure based on such theory, parts of which we review in the next section.

• *The author is with Manna Network Technologies, Harachev St. #4, Tel Aviv 61574, ISRAEL. E-mail: jer@mannanetwork.com.*

Manuscript received 20 June 1997; revised 29 Apr. 1998. Recommended for acceptance by A. Webb.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 106855.

2 Theoretical Background

We use the following setting: Given M distinct pattern classes each with a class conditional probability density $f_i(x)$, $1 \le i \le M$, $x \in \mathbb{R}^d$, and a priori probabilities p_i, $1 \le i \le M$. The functions $f_i(x)$, $1 \le i \le M$, are assumed to be unknown while the p_i are assumed to be known or easily estimable as is the case of learning character recognition. For a sample-size vector $m = [m_1, \ldots, m_M]$ where $\sum_{i=1}^{M} m_i = \overline{m}$ denote by $\zeta^m = \left\{\left(x_j, y_j\right)\right\}_{j=1}^{\overline{m}}$ a sample of labeled examples consisting of m_i examples from pattern class i, where y_j, $1 \le j \le \overline{m}$, are chosen *not* necessarily at random from $\{1, 2, \ldots, M\}$, and the corresponding x_j are drawn at random i.i.d. according to the class conditional probability density $f_{y_j}(x)$. The expected misclassification error of a classifier c is referred to as the *loss* of c and is denoted by $L(c)$. It is defined as the probability of misclassification of a randomly drawn x with respect to the underlying mixture probability density function

$$f(x) = \sum_{i=1}^{M} p_i f_i(x).$$

Let $1_{\{x:x\in A\}}$ denote the indicator function of a set A. We use the same notation to denote the Kronecker delta function $1_{\{B(x)\}}$ for x being a discrete variable and where $B(x)$ is a condition on x which can either be true or false, for instance, an equality condition $g(x) = i$. The loss is commonly represented as

$$\mathrm{L}(c) = E1_{\{(x,y):c(x)\neq y\}}$$

where expectation is taken with respect to the joint probability distribution $f_y(x)p(y)$ where $p(y)$ is a discrete probability distribution taking values p_i over $1 \le i \le M$, while y denotes the label of the class whose distribution $f_y(x)$ was used to draw x. The loss $L(c)$ may also be written as

$$L(c) = \sum_{i=1}^{M} p_i E_i 1_{\{x:c(x)\neq i\}}$$

where E_i denotes expectation with respect to $f_i(x)$. The pattern recognition problem is to learn based on ζ^m the optimal classifier, also known as the *Bayes classifier*, which by definition has minimum loss which we denote by L^*.

A multicategory classifier c is represented as a vector $c(x) = [c_1(x), \ldots, c_M(x)]$ of *Boolean classifiers*, where $c_i(x) = 1$, if $c(x) = i$ and $c_i(x) = 0$ otherwise, $1 \le i \le M$. The loss $L(c)$ of a multicategory classifier c may then be expressed as the average of the losses of its component classifiers, i.e.,

$$L(c) = \sum_{i=1}^{M} p_i L(c_i)$$

where for a Boolean classifier c_i the loss is defined as

$$L(c_i) = E_i 1_{\{x: c_i(x) \neq 1\}}.$$

As an estimate of $L(c)$ we define the *empirical loss*

$$L_m(c) = \sum_{i=1}^{M} p_i L_{m_i}(c),$$

where

$$L_{m_i}(c) = \frac{1}{m_i} \sum_{j: y_j = i} 1_{\{c(x_j) \neq i\}}$$

which may also can be expressed as

$$L_{m_i}(c_i) = \frac{1}{m_i} \sum_{j: y_j = i} 1_{\{c_i(x_j) \neq 1\}}.$$

Vapnik and Chervonenkis introduced a measure of complexity of a class of Boolean classifiers which is known as the VC-dimension, cf. [14]. It is defined as follows:

DEFINITION 1. (*Vapnik-Chervonenkis Dimension*) *Given a class $\mathcal{B}$ of Boolean classifiers $b(x)$ over X, the Vapnik-Chervonenkis dimension of $\mathcal{B}$, is defined as the largest integer l such that there exists a sample $x^l = x_1, \ldots, x_l$ of points in X such that the cardinality of the set of Boolean vectors*

$$S_{x^l}(\mathcal{B}) = \left\{ \left[b(x_1), \ldots, b(x_1) \right] : b \in \mathcal{B} \right\}$$

satisfies $\left| S_{x^l}(\mathcal{B}) \right| = 2^l$. If l is arbitrarily large then the VC-dimension of $\mathcal{B}$ is infinite.

The main interest in the VC-dimension quantity is due to the following result on a uniform strong law of large numbers which is a variant of Theorem 6.7 in Vapnik [14]

LEMMA 1. (*Uniform SLLN for Boolean Classifier Class*) *Let $\mathcal{B}$ be a class of Boolean classifiers over X with $VC(\mathcal{B}) = k < \infty$. Let*

$$z^l = \left\{ \left(x_j, y_j \right) \right\}_{j=1}^{l}, \; x_j \in X, \; y_j \in \{0, 1\}, \; 1 \leq j \leq l,$$

be a sample of size $l > k$ consisting of randomly drawn examples according to any fixed probability distribution P on $X \times \{0, 1\}$. Let

$$L(b) = E 1_{\{(x,y): b(x) \neq y\}}$$

with expectation with respect to P and

$$L_l(b) = \frac{1}{l} \sum_{j=1}^{l} 1_{\{b(x_j) \neq y_j\}}$$

denote the loss and empirical loss for b, respectively. Then for arbitrary confidence parameter $0 < \delta < 1$, the deviation between the empirical loss and the true loss uniformly over $b \in \mathcal{B}$ is bounded as

$$\sup_{b \in \mathcal{B}} \left| L(b) - L_l(b) \right| \leq 4 \sqrt{\frac{k\left(\ln \frac{2l}{k} + 1 \right) + \ln\left(\frac{9}{\delta} \right)}{l}}$$

with probability $1 - \delta$.

We consider a class of multicategory classifiers to be decomposed into a multistructure $S = S_1 \times S_2 \times \cdots \times S_M$, where S_i is a nested structure (cf. Vapnik [14]) of Boolean families $\mathcal{B}_{k_{j_i}}$, $j_i = 1, 2, \ldots$, for $1 \leq i \leq M$, i.e., $S_1 = \mathcal{B}_{k_1}, \mathcal{B}_{k_2}, \ldots, \mathcal{B}_{k_{j_1}}, \ldots$, $S_2 = \mathcal{B}_{k_1}, \mathcal{B}_{k_2}, \ldots, \mathcal{B}_{k_{j_2}}, \ldots$, up to $S_M = \mathcal{B}_{k_1}, \mathcal{B}_{k_2}, \ldots, \mathcal{B}_{k_{j_M}}, \ldots$, where $k_{j_i} \in \mathbb{Z}_+$ denotes the VC-dimension of $\mathcal{B}_{k_{j_i}}$ and $\mathcal{B}_{k_{j_i}} \subseteq \mathcal{B}_{k_{j_{i+1}}}$, $1 \leq i \leq M$. For any fixed positive integer vector $j \in \mathbb{Z}_+^M$, consider the class of vector classifiers $\mathcal{H}_{k(j)} = \mathcal{B}_{k_{j_1}} \times \mathcal{B}_{k_{j_2}} \times \cdots \times \mathcal{B}_{k_{j_M}} \equiv \mathcal{H}_k$, where we take the liberty in dropping the multiindex j and write k instead of $k(j)$. Define by $\mathcal{G}_k$ the subfamily of $\mathcal{H}_k$ consisting of classifiers c that are well-defined, i.e., ones whose components c_i, $1 \leq i \leq M$ satisfy

$$\bigcup_{i=1}^{M} \{x: c_i(x) = 1\} = \mathbb{R}^d$$

and $\{x: c_i(x) = 1\} \cap \{x: c_j(x) = 1\} = \varnothing$, for $1 \leq i \neq j \leq M$.

From Lemma 1, it follows that the loss of any Boolean classifier $c_i \in \mathcal{B}_{k_{j_i}}$ is, with high confidence, related to its empirical loss as

$$L(c_i) \leq L_{m_i}(c_i) + \epsilon\left(m_i, k_{j_i} \right)$$

where

$$\epsilon\left(m_i, k_{j_i} \right) = const \sqrt{k_{j_i} \ln m_i / m_i},$$

$1 \leq i \leq M$, where, henceforth, we denote by *const* any constant which does not depend on the relevant variables in the expression. Let the vectors $m = [m_1, \ldots, m_M]$ and

$$k \equiv k(j) = \left[k_{j_1}, \ldots, k_{j_M} \right]$$

in $\mathbb{Z}_+^M$. Define

$$\epsilon(m, k) = \sum_{i=1}^{M} p_i \epsilon\left(m_i k_{j_i} \right).$$

It follows that the deviation between the empirical loss and the loss is bounded uniformly over all multicategory classifiers in a class $\mathcal{G}_k$ by $\epsilon(m, k)$. We henceforth denote by c_k^* the optimal classifier in $\mathcal{G}_k$, i.e., $c_k^* = \arg\min_{c \in \mathcal{G}_k} L(c)$ and $\hat{c}_k = \arg\min_{c \in \mathcal{G}_k} L_m(c)$ is the empirical loss minimizer over the class $\mathcal{G}_k$.

For any $k \in \mathbb{Z}_+^M$, by the triangle inequality we have with high probability $1 - \delta$,

$$L(\hat{c}_k) \leq L_m(\hat{c}_k) + \epsilon(m, k) \leq L_m(c_k^*) + \epsilon(m, k) \leq L(c_k^*) + 2\epsilon(m, k)$$

which, after passing the factor of two inside the constant in the definition of $\epsilon(m, k)$, says that $\hat{c}_k$ has a loss which is no more than $\epsilon(m, k)$ from the best possible loss in $\mathcal{G}_k$. Denote by k^* the minimal complexity of a class $\mathcal{G}_k$ which contains the Bayes classifier. We refer to it as the *Bayes complexity* and, henceforth, assume $k_i^* < \infty$, $1 \leq i \leq M$. Theoretically, if k^* was known then based on a sample of size $\overline{m}$ with a sample size vector $m = [m_1, \ldots, m_M]$ a classifier $\hat{c}_{k^*}$ whose loss is bounded from above by $L^* + \epsilon(m, k^*)$ may be determined just by doing empirical loss minimization over $\mathcal{G}_{k^*}$, where $L^* = L(c_{k^*}^*)$ is the Bayes loss. This bound is minimal with respect to k by definition of k^* and we refer to it as the *minimal criterion*. It can be further minimized by selecting a sample of size vector

$$m^* = \arg\min_{\left\{m \in \mathbf{Z}_+^M : \sum_{i=1}^M m_i = \overline{m}\right\}} \epsilon(m, k).$$

The latter implies that a larger number of examples should be queried from pattern classes which require more complex discriminating rules within the Bayes classifier.

Thus, sample-querying via minimization of the minimal criterion makes learning more efficient through tuning the subsample sizes to the complexity of the Bayes classifier. However, the Bayes classifier depends on the underlying probability distributions which in most interesting scenarios are unknown so k^* should be assumed unknown. In Ratsaby [12], an incremental learning algorithm, based on Vapnik's structural risk minimization (cf. Vapnik [5], Devroye et. al. [5]), generates a random complexity sequence $\hat{k}(n)$, corresponding to a sequence of empirical loss minimizers $\hat{c}_{\hat{k}(n)}$ over $\mathcal{G}_{\hat{k}(n)}$, which converges to k^* with increasing time n. Based on this, a sample-query rule which achieves the same minimization is defined without needing to know k^*. The theory in [12] holds for learning problems with a zero Bayes loss and where the Bayes classifier is assumed to be contained in the structure of families of classifiers. We now briefly describe the main ideas of this incremental learning algorithm.

At any time n, the criterion function is $\epsilon\left(\cdot, \hat{k}(n)\right)$ and is defined over the m-domain $\mathbf{Z}_+^M$. A gradient descent step of a fixed size is taken to minimize the current criterion. After a step is taken, a new sample-size vector $m(n + 1)$ is obtained and the difference $m(n + 1) - m(n)$ dictates the sample-query at time n, namely, the increment in subsample size for each of the M pattern classes. With increasing n the vector sequence $m(n)$ gets closer to an *optimal path* defined as the set which is comprised of the solutions to the minimization of $\epsilon(m, k^*)$ under all different constraints of $\sum_{i=1}^M m_i = \overline{m}$, where $\overline{m}$ runs over the positive integers. Thus for all large n the sample-size vector $m(n)$ is optimal in that it minimizes the minimal criterion $\epsilon(\cdot, k^*)$ for the current total sample size $\overline{m}(n)$. This constitutes the sample-querying procedure of the learning algorithm. The remaining part simply does empirical loss minimization over the current class $\mathcal{G}_{\hat{k}(n)}$ and outputs $\hat{c}_{\hat{k}(n)}$. By assumption, since the Bayes classifier is contained in $\mathcal{G}_{k^*}$, it follows that for all large n, the loss

$$L\left(\hat{c}_{\hat{k}(n)}\right) \le L^* + \min_{\left\{m \in \mathbf{Z}_+^M : \sum_{i=1}^M m_i = \overline{m}(n)\right\}} \epsilon\left(m, k^*\right),$$

which is basically the minimal criterion mentioned above. Thus, the algorithm produces a classifier $\hat{c}_{\hat{k}(n)}$ with a minimal loss even when the Bayes complexity k^* is unknown.

In the next section we consider specific model classes consisting of nearest-neighbor classifiers on which we implement this incremental learning approach.

3 Incremental Nearest-Neighbor Algorithm

Fix and Hodges (cf. Silverman, et al. [2]) introduced the simple but powerful nearest-neighbor classifier which based on a labeled training sample $\left\{\left(x_i, y_i\right)\right\}_{i=1}^{\overline{m}}$, $x_i \in \mathbb{R}^d$, $y_i \in \{1, 2, \ldots, M\}$, when given a pattern x, it outputs the label y_j corresponding to the example whose x_j is closest to x. Every example in the training sample is used for this decision (we denote such an example as a *prototype*), thus, the empirical loss is zero. The condensed nearest-neighbor algorithm (Hart [9]) and the reduced nearest neighbor algorithm (Gates [8]) are procedures which aim at reducing the number of prototypes while maintaining a zero empirical loss. Both procedures are convergent and, when given a training sample of size $\overline{m}$, they output a nearest neighbor classifier which has a zero empirical loss and is based on $\bar{l} \le \overline{m}$ prototypes. Learning in this manner may be viewed as a form of empirical loss minimization with a complexity regularization component that puts a penalty proportional to the number of prototypes.

A cell boundary $e_{i,j}$ of the Voronoi diagram (cf. Preparata and Shamos [11]) corresponding to a multicategory nearest-neighbor classifier c is defined as the $(d - 1)$-dimensional perpendicular-bisector hyperplane of the line connecting the x-component of two prototypes x_i and x_j. For a fixed $l \in \{1, \ldots, M\}$, the collection of Voronoi cell boundaries based on pairs of prototypes of the form (x_i, l), (x_j, q) where $q \ne l$, forms the boundary which separates the decision region labeled l from its complement and which represents the boolean nearest-neighbor classifier c_l. Denote by k_l the number of such cell boundaries and denote by s_l the number of prototypes from a total of m_l examples of pattern class l. Using Delaunay triangulation (cf. Preparata and Shamos [11]) the value of k_l may be calculated directly from the knowledge of the s_l prototypes, $1 \le l \le M$.

The Boolean classifier c_l may be taken as an element of an infinite class of Boolean classifiers based on partitions of $\mathbb{R}^d$ by arrangements of k_l hyperplanes of dimensionality $d - 1$, where each of the cells of a partition is labeled either 0 or 1. From Devroye et al. [p. 512, 5], the number of cells in a partition is 2^{k_l} if $k_l \le d$ and is no more than $(ek_l/d)^d$ when $d < k_l$. Combined with Theorem 19.6, p. 311, which upper bounds the loss of a Boolean nearest-neighbor classifier, it follows that the loss of a multicategory nearest-neighbor classifier c which consists of s_l prototypes from m_l examples, $1 \le l \le M$, is bounded as $L(c) \le L_m(c) + \epsilon(m, k)$, where the a priori probabilities are taken as known, $m = [m_1, \ldots, m_M]$, $k = [k_1, \ldots, k_M]$ and

$$\epsilon(m, k) = \sum_{l=1}^M p_l \epsilon\left(m_l, k_l\right),$$

where

$$\epsilon\left(m_l, k_l\right) = const \sqrt{\left((d+1)k_l \ln m_l + \left(ek_l/d\right)^d\right)/m_l}.$$

Letting k^* denote the Bayes complexity then $\epsilon(\cdot, k^*)$ represents the minimal criterion.

The next algorithm uses the Condense and Reduce procedures in order to generate a sequence of classifiers $\hat{c}_{\hat{k}(n)}$ with a complexity vector $\hat{k}(n)$ which tends to k^* as $n \to \infty$. A sample-querying procedure referred to as Greedy Query (GQ) chooses at any time n to increment the single subsample of pattern class $j^*(n)$ where $m_{j^*(n)}$ is the direction of maximum descent of the criterion $\epsilon\left(\cdot, \hat{k}(n)\right)$ at the current sample-size vector $m(n)$. For the part of the algorithm which utilizes a Delaunay-Triangulation procedure, we use the fast Fortune's algorithm (cf. O'Rourke [10]) which can be used only for dimensionality $d = 2$. It turns out that since all we are interested is in counting Voronoi borders between all adjacent Voronoi cells, then an efficient computation is possible also for dimensions $d > 2$. Here, one may use algorithms that compute the adjacencies of facets of a polyhedron, cf.

Fukuda [7]. Specifically, for a set S of points $x_1, \ldots, x_n \in \mathbb{R}^d$ one first transforms x_i to a point z_i on a paraboloid and which is defined as $z_i = [x_{i,1}, \ldots, x_{i,d}, \|x\|^2]$ where $\|x\|$ denotes the l_2-norm in $\mathbb{R}^d$, $x_{i,j}$ is the jth component of the vector x_i, $1 \le i \le n$, $1 \le j \le d$. It is well known that the polyhedron in $\mathbb{R}^{d+1}$ with facets being the hyperplanes tangent to the paraboloid at the points z_i is a lifting of the Voronoi diagram of the set S. Adjacent Voronoi cells in the diagram map to adjacent facets of the polyhedron. Linear programming is then used for finding adjacent facets efficiently (cf. Fukuda [6]).

Incremental Nearest Neighbor (INN) Algorithm

Initialization: (Time $n = 0$)

Let increment-size Δ be a fixed small positive integer. Start with $m(0) = [c, \ldots, c]$, where c is a small positive integer. Draw

$$\zeta^{m(0)} = \left\{\zeta^{m_j(0)}\right\}_{j=1}^{M}$$

where $\zeta^{m_j(0)}$ consists of $m_j(0)$ randomly drawn i.i.d. examples from pattern class j.

While (number of available examples $\ge \Delta$) **Do:**

1) **Call Procedure CR**: $\hat{c}_{\hat{k}(n)} = CR\left(\zeta^{m(n)}\right)$.
2) **Call Procedure GQ**: $m(n+1) = GQ(n)$.
3) $n := n + 1$.

End While

`//Used up all examples`.

Output: NN-classifier $\hat{c}_{\hat{k}(n)}$.

Procedure Condense-Reduce (CR)

Input: Sample $\zeta^{m(n)}$ stored in an array $A[]$ of size $\overline{m}(n)$.

Initialize: Make only the first example $A[1]$ be a prototype.

`//Condense`

Do:

ChangeOccured := FALSE.

For $i = 1, \ldots, \overline{m}(n)$:

- **Classify** $A[i]$ based on available prototypes using the NN-Rule.
- **If** not correct **then**
  - Let $A[i]$ be a prototype.
  - *ChangeOccured* := TRUE.
- End If

End For

While (*ChangeOccured*).

`//Reduce`

Do:

ChangeOccured := FALSE.

For $i = 1, \ldots, \overline{m}(n)$:

- **If** $A[i]$ is a prototype **then** classify it using the remaining prototypes by the NN-Rule.
- **If** correct **then**
  - Make $A[i]$ be not a prototype.
  - *ChangeOccured* := TRUE.
- End If

End For

While (*ChangeOccured*).

Run Delaunay-Triangulation Let $\hat{k}(n) = \left[\hat{k}_1, \ldots, \hat{k}_M\right]$, $\hat{k}_i$ denotes the number of Voronoi-cell boundaries associated with the $\hat{s}_i$ prototypes.

Return (NN-classifier with complexity vector $\hat{k}(n)$).

Procedure Greedy-Query (GQ)

Input: Time n.

$$j^*(n) := \arg\max_{1 \le j \le M} \left| \frac{\partial}{\partial m_j} \epsilon\left(m, \hat{k}(n)\right) \right|_{|m(n)}.$$

Draw: Δ new i.i.d. examples from class $j^*(n)$. Denote them by ζ.

Update Sample:

$$\zeta^{m_{j^*(n)}(n+1)} := \zeta^{m_{j^*(n)}(n)} \cup \zeta,$$

while $\zeta^{m_i(n+1)} := \zeta^{m_i(n)}$, for $1 \le i \ne j^*(n) \le M$.

Return: $\left(m(n) + \Delta e_{j^*(n)}\right)$, where e_j is an all-zero vector except 1 at jth element.

3.1 Experimental Results

We ran algorithm INN on four two-dimensional ($d = 2$) multicategory classification problems and compared its generalization error versus total sample size $\overline{m}$ with that of batch learning, the latter uses Procedure CR (but not Procedure GQ) with uniform subsample proportions, i.e., $m_i = \frac{\overline{m}}{M}$, $1 \le i \le M$. Each problem consists of four equiprobable pattern classes with a zero Bayes loss. The generalization curves were averaged over 15 independent learning runs (both for INN and Batch learning) where each run reaches a total of 800 examples. The results are displayed in Fig. 1, Fig. 2, Fig. 3, and Fig. 4 as a series of pairs, the first picture showing the pattern classes of the specific problem while the second shows the learning curves for the two learning algorithms plotted in a semilog form. In all problems, from the difference between the Batch and INN learning curves, we see that Algorithm INN outperformed the simple Batch approach, the latter ignoring the inherent Bayes complexity and using an equal subsample size for each of the pattern classes. In contrast, the INN algorithm learns incrementally over time which of the classes are harder to separate and queries more from these pattern classes. Asymptotically the learning rates of INN and batch appear to have a constant factor difference. Denote by $q(k)$ the proportion-vector corresponding to the vector k, i.e., $q(k) = \left[\frac{k_1}{k}, \ldots, \frac{k_4}{k}\right]$, where

$$\overline{k} = \sum_{j=1}^{4} k_j.$$

Let $u = \left[\frac{1}{4}, \ldots, \frac{1}{4}\right]$ and let

$$D(q(k)\|u) = \sum_{j=1}^{4} q_j(k) \ln \frac{q_j(k)}{u_j}$$

denote the Kullback-Leibler distance between $q(k)$ and u. For each of the four classification problems, we calculated $D\left(q\left(\hat{k}\right)\|u\right)$ for $\hat{k}$ being the complexity vector when the total number of examples $\overline{m} = 800$ which we assume is large enough so that $q\left(\hat{k}\right) \simeq q\left(k^*\right)$. Both $D\left(q\left(\hat{k}\right)\|u\right)$ and the improvement in the generalization error exhibited by the use of sample querying, increase as we go in Fig. 1, Fig. 2, Fig. 3, and Fig. 4. Thus, the amount of improvement is proportional to the degree to which the Bayes complexity vector k^* differs from being uniform.

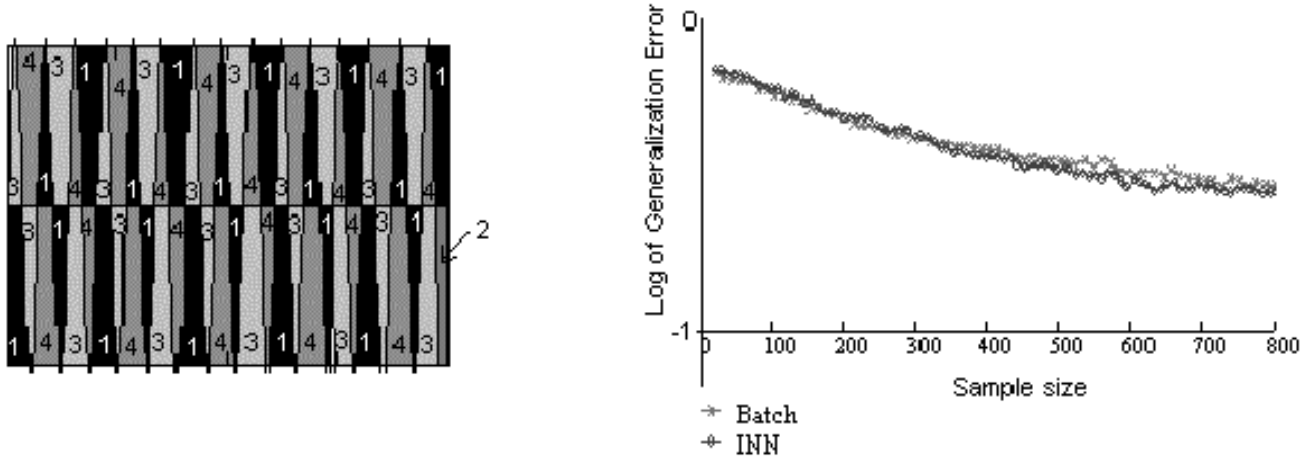

Fig. 1. Classification problem #1. at $m = 800$, the value of $D(q(k)||u) = 0.242$.

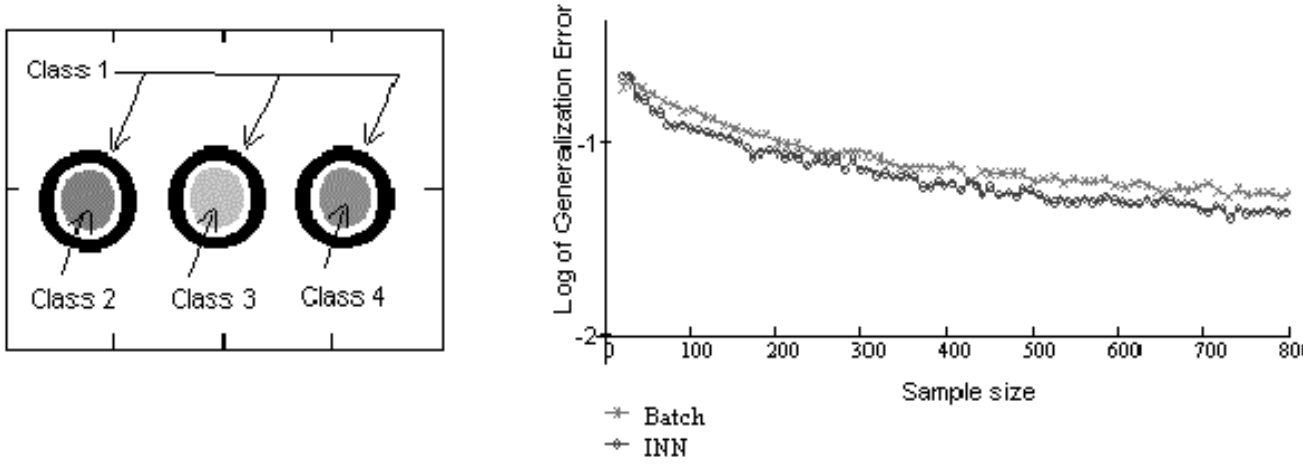

Fig. 2. Classification problem #2. at $m = 800$, the value of $D(q(k)||u) = 0.275$.

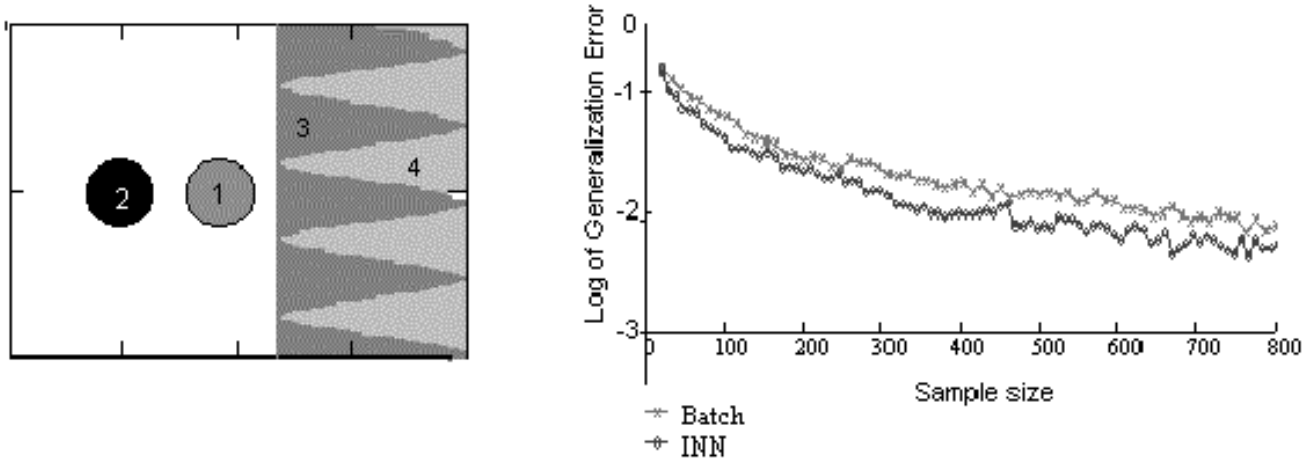

Fig. 3. Classification problem #3. at $m = 800$, the value of $D(q(k)||u) = 0.531$.

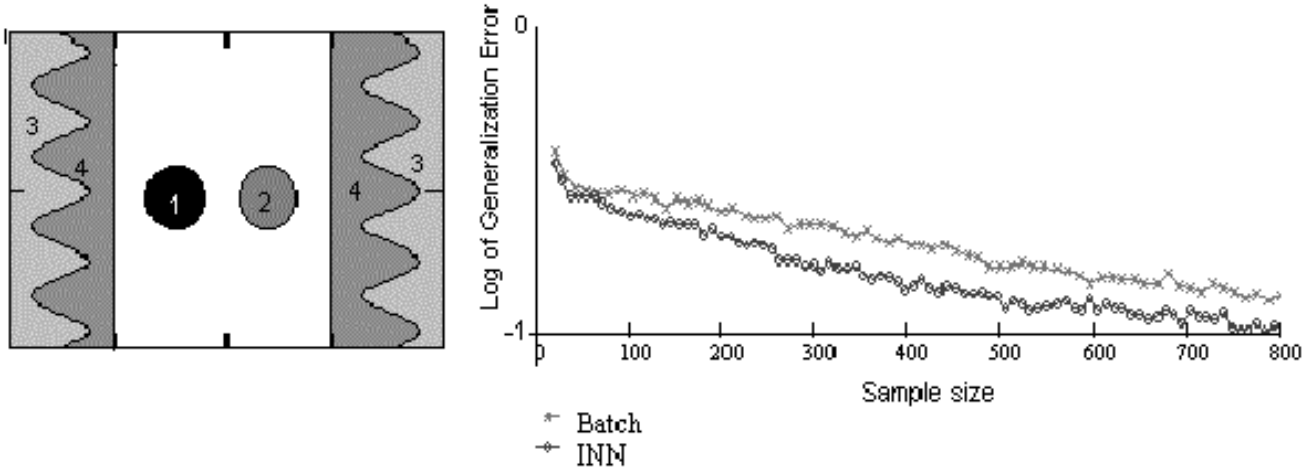

Fig. 4. Classification problem #4. at $m = 800$, the value of $D(q(k)||u) = 0.622$.

4 Conclusion

We considered the general problem of learning multicategory classification from labeled samples in a setting where the sample size or time available for training are limited, and where the class a priori probabilities are known or easily estimable. We introduced a novel notion of sample querying based on an incremental learning approach, which adaptively tunes the subsample sizes of each of the pattern classes according to the unknown complexity of the Bayes optimal classifier. The principle is general enough to be used in any learning algorithm that permits a model-selection criterion and for which the error rate for the classifier is calculable in terms of the complexity of the model. Experimental results for an incremental nearest-neighbor algorithm (INN) show that sample querying improves the expected misclassification error (the loss) over that of batch learning by an amount which depends on the underlying complexity of the Bayes classifier.

References

[1] D. Angluin, "Queries and Concept Learning," *Machine Learning*, vol. 2, pp. 319-342, 1988.

[2] B.W. Silverman, M.C. Jones, E. Fix, and J.l. Hodges, "An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation—Commentary on Fix and Hodges" (1951). *International Statistical Review*, vol. 57, no. 3, pp. 233-247, 1989.

[3] D. Cohn, "Neural Network Exploitation Using Optimal Experiment Design," *Neural Networks*, vol. 9, no. 6, pp. 1,071-1,083, 1996.

[4] D. Cohn, L. Atlas, and R. Ladner, "Improving Generalization With Active Learning," *Machine Learning*, vol 15, pp. 201-221, 1994.

[5] L. Devroye, L. Gyorfi, and G. Lugosi, "A Probabilistic Theory of Pattern Recognition," Springer Verlag, 1996.

[6] K. Fukuda, "Frequently Asked Questions in Geometric Computation," Technical Report, Swiss Federal Inst. of Technology, Lausanne, 1997. Available at ftp://ftp.ifor.ethz.ch/pub/fukuda/reports.

[7] K. Fukuda, *cdd+ Reference Manual*. Inst. for Operations Research, Swiss Federal Inst. of Technology, Zurich, Switzerland, 1995. Available at http://www.ifor.ethz.ch/staff/fukuda/fukuda.html.

[8] G.W. Gates, "The Reduced Nearest Neighbor Rule," *IEEE Trans. Information Theory*, pp. 431-433, 1972.

[9] P.E. Hart, "The Condensed Nearest Neighbor Rule," *IEEE Trans. on Information Theory*, vol. 14, no. 3, 1968.

[10] J. O'Rourke, *Computational Geometry in C*. Cambridge, Mass.: University Press, 1994.

[11] F.P. Preparata, M.I. Shamos. *Computational Geometry—An Introduction*. New York: Springer-Verlag, 1985.

[12] J. Ratsaby, "Learning Classification with Sample Queries," Submitted to *Information and Computation*, 1997. Available at http://www.ee.technion.ac.il/ jer/iandc.ps.

[13] R.L. Rivest, B. Eisenberg, "On the Sample Complexity of Pac-Learning Using Random and Chosen Examples," *Proc. 1990 Workshop on Computational Learning Theory*, pp. 154-162, San Maeto, CA, 1990.

[14] V.N. Vapnik, *Estimation of Dependences Based on Empirical Data*. Berlin: Springer-Verlag, 1982.