Machine Learning, 54, 125–152, 2004

Selective Sampling for Nearest Neighbor Classifiers

MICHAEL LINDENBAUM mic@cs.technion.ac.il
SHAUL MARKOVITCH shaulm@cs.technion.ac.il
DMITRY RUSAKOV rusakov@cs.technion.ac.il
Computer Science Department, Technion, Israel Institute of Technology 32000, Haifa, Israel

Editor: David W. Aha

Abstract. Most existing inductive learning algorithms work under the assumption that their training examples are already tagged. There are domains, however, where the tagging procedure requires significant computation resources or manual labor. In such cases, it may be beneficial for the learner to be *active*, intelligently selecting the examples for labeling with the goal of reducing the labeling cost. In this paper we present LSS—a lookahead algorithm for selective sampling of examples for nearest neighbor classifiers. The algorithm is looking for the example with the highest utility, taking its effect on the resulting classifier into account. Computing the expected utility of an example requires estimating the probability of its possible labels. We propose to use the *random field* model for this estimation. The LSS algorithm was evaluated empirically on seven real and artificial data sets, and its performance was compared to other selective sampling algorithms. The experiments show that the proposed algorithm outperforms other methods in terms of average error rate and stability.

Keywords: active learning, selective sampling, nearest neighbor, random field

1. Introduction

Most existing inductive learning algorithms assume the availability of a training set of labeled examples. In many domains, however, labeling the examples is a costly process. Consider, for example, training a classifier for a character recognition problem. In this case, the character images are easily available, while their classification is a costly process requiring a human operator. Another example is the financial expense incurred for a medical test required to find out whether a patient suffers from a certain disease. Yet another example may be a chess playing program in which one wants to learn a position-appraising function. Usually, the true score for each position is ascertained by doing a deep minimax search in the game tree. Such a search is computationally expensive, while any number of legal positions can be easily generated.

In such domains it is desirable to reduce the number of training examples while maintaining the quality of the resulting classifier. A possible solution to this problem is to provide the learning algorithm with some control over the inputs on which it trains. This paradigm is called *active learning*.

One approach of active learning is using *membership queries* (Angluin, 1988), in which the learner builds an untagged example and requests for it a label from the teacher. One potential practical problem with this approach is that the constructed example may make

no sense in the particular domain, such as an illegal chess board or an arbitrary image in a character recognition domain.

An alternative approach assumes that a (potentially large) set of unlabeled examples is available. The learner selects an unlabeled example from the given set and asks the teacher to label it. This approach is called *selective sampling* (Lewis & Catlett, 1994; Freund et al., 1997; Dagan & Engelson, 1995). This paper concentrates on the selective sampling approach, which is more common in practice.

The problem of selective sampling resembles the problem of *cost-sensitive learning* (Turney, 1995; Tan & Schlimmer, 1990). In cost-sensitive learning, however, the costs are associated with obtaining additional information about each case (i.e., obtaining the missing attributes) rather than with obtaining the labels of unclassified points, as in selective sampling. Also related to selective sampling is the *example filtering* paradigm (Wilson & Martinez, 2000; Smyth & McKenna, 1999; Zhang, Yim, & Yang, 1997; Aha, Kibler, & Albert, 1991), where redundant labeled examples (with respect to previously stored examples) are removed in order to reduce the complexity of the resulting classifier and improve generalization.

One recent approach to selective sampling in a general setting, which also has a strong theoretical justification, is *Query By Committee*. In this approach, a committee of hypotheses consistent with the labeled data is specified and an unlabeled example on which the committee members most disagree is chosen (Seung, Opper, & Sompolinsky, 1992; Freund et al., 1997; Hasenjager & Ritter, 1996; RayChaudhuri & Hamey, 1995). Techniques from the field of optimal experiment design (Fedorov, 1972) have been applied to active learning of real-valued functions (MacKay, 1992; Cohn, Ghahramani, & Jordan, 1995). Other approaches use a sampling process to find an optimum of some unknown function (Moore et al., 1998).

Various selective sampling methods have been developed for specific classification learning algorithms: for neural networks (Davis & Hwang, 1992; Cohn, Atlas, & Lander, 1994), for the $C4.5$ rule-induction algorithm (Lewis & Catlett, 1994), and for Hidden Markov Models (Dagan & Engelson, 1995). The problem of active learning for nearest neighbor classifiers in the context of the membership queries paradigm was considered by Hasenjager and Ritter (1998). However, no selective sampling algorithms were developed for nearest neighbor classifiers.

The goal of this paper is to fill the gap and to develop a selective sampling methodology for nearest-neighbor (NN) classification learning algorithms, for domains with uniform labeling costs in the instance space. The nearest neighbor algorithm (Cover & Hart, 1967; Aha, Kibler, & Albert, 1991) is a non-parametric classification method, useful especially when little information is available about the structure of the class distribution, a fact which would imply that parametric classifiers are harder to construct. The NN classifier is widely used because it does not rely on a parametric model, is simple to implement, and asymptotically errs with a bounded rate that is at most twice the Bayesian error.

This paper introduces a lookahead algorithm for selective sampling that is suitable for nearest neighbor classification. The lookahead principle has already been applied in learning algorithms. See, for example, Murthy and Salzberg (1995) for its use in decision tree induction. Our algorithm, named LSS (*Lookahead Selective Sampling*), chooses the next

example (or sequence of examples) in order to maximize the expected utility of the resulting classifier. The major components of this framework are a utility function for appraising classifiers and a posteriori class probability estimate for unlabeled points in the instance space. We derive a Bayesian utility function and propose a *random field* model for the feature space classification structure. This model serves as the basis for class probability estimation.

The merit of our approach is empirically demonstrated on real and artificial problems. We compared LSS with a number of known selective sampling methods adopted for use with nearest neighbor classifiers. The lookahead selective sampling method outperformed the other methods in terms of average error rate and performed with much more stability on a wide range of real and artificial problems. The proposed algorithm has a rather large computational complexity which is a square of the number of training examples. Such complexity, however, is well-justified for many real-world problems, where labeling costs are high in comparison to the cost of computational resources.

This article is organized as follows. We start by formalizing the problem of selective sampling and presenting intuitive considerations for selective sampling methods (Section 2). In Section 3 we describe the lookahead-based framework for selective sampling. In Section 4 we describe methods for estimating the utility of classifiers. In Section 5 we develop a random field model for the feature space classification structure, and in Section 6 we show how this methodology can be implemented. Section 7 describes the experimental evaluation of our algorithm and Section 8 concludes.

2. Selective sampling for nearest neighbor classifiers

In order to address the problem of selective sampling for nearest neighbor classifiers, we first explain and formalize the concept of selective sampling.

2.1. *The selective sampling process*

We consider the following selective sampling paradigm. Let $\mathcal{X}$ be an *instance space*, a set of objects described by a finite collection of attributes (*features*). Let $f : \mathcal{X} \to \{0, 1\}$ be a *teacher* (also called an expert) that labels instances as 0 or 1. A *(supervised) learning algorithm* takes a set of *labeled examples*, $\{\langle x_1, f(x_1)\rangle, \ldots, \langle x_n, f(x_n)\rangle\}$, and returns a *hypothesis* $h : \mathcal{X} \to \{0, 1\}$. Throughout this paper we assume that $\mathcal{X} = \mathbb{R}^{\mathrm{d}}$.

Let X be an *unlabeled training set*, a set of objects drawn randomly from $\mathcal{X}$ according to distribution $\mathbf{p}$. Let $D = \{\langle x_i, f(x_i)\rangle : x_i \in X, i = 1, \ldots, n\}$ be the *training data*—a set of labeled examples from X. A selective sampling algorithm S_L, specified relative to a learning algorithm L, receives X and D as input and returns an unlabeled element of X. This is in contrast to the common, non-incremental, learning framework, where the algorithms receive only D as an input.

The process of learning with selective sampling can be described as an iterative procedure where at each iteration the selective sampling procedure is called to obtain an unlabeled example and the teacher is called to label that example. The labeled example is added to the set of currently available labeled examples and the updated set is given to the learning

Active Learner$(X, f(\,))$:

1. $D \leftarrow \emptyset$.
2. $h \leftarrow L(\emptyset)$.
3. While stopping-criterion is not satisfied do:
   a) $x \leftarrow S_L(X, D)$; Apply S_L and get the next example.
   b) $\omega \leftarrow f(x)$; Ask the teacher to label x.
   c) $D \leftarrow D \bigcup \{\langle x, \omega \rangle\}$; Update the labeled examples set.
   d) $h \leftarrow L(D)$; Update the classifier.
4. Return classifier h.

Figure 1. Active learning with selective sampling. Active Learner is defined by specifying stopping-criterion, learning algorithm L and selective sampling algorithm S_L. It then works with unlabeled data X and teacher $f(\,)$ as input.

procedure, which induces a new classifier. This sequence repeats until some stopping criterion is satisfied. This criterion may be a resource bound, M, on the number of examples that the teacher is willing to label, or a lower bound on the desired class accuracy. Adopting the first stopping criterion, the goal of the selective sampling algorithm is to produce a sequence of length M which leads to a best classifier according to some given measure.

The pseudo-code for an active learning system that uses selective sampling is shown in figure 1.

2.2. *Nearest neighbor classification*

In this paper we are particularly interested in selective sampling for the nearest neighbor classifier (Cover & Hart, 1967). The nearest neighbor classifier accumulates the labeled examples received as input. An unlabeled instance is then classified according to the label of its nearest labeled neighbor. Variations of this scheme include k-nearest neighbor classifiers (Duda & Hart, 1973), which use the vote of the k-nearest labeled neighbors, and selective classifiers, which store and utilize the labeled examples selectively (Aha, Kibler, & Albert, 1991).

Given a set of labeled examples $\{\langle x_1, \omega_1 \rangle, \ldots, \langle x_n, \omega_n \rangle\} \subseteq \mathcal{X} \times \{0, 1\}$ and a similarity function d over $\mathcal{X}$, the nearest neighbor rule decides that $x \in \mathcal{X}$ belongs to the category ω_j of its nearest neighbor x_j, where $j = \arg\min_{i=1,\ldots,n} d(x_i, x)$. In this paper we assume that d is Euclidean metric.

2.3. *Some intuitive considerations*

What should be considered a good example? In this subsection we discuss some of the intuitive considerations for selective sampling. The underlying principle is that the uncertainty of classification should be reduced. One possible strategy for reducing this uncertainty is to assign a classification uncertainty to each unlabeled example and select the unlabeled instance associated with the highest uncertainty. For the nearest neighbor classifier, such points have two nearest neighbors with similar distances but conflicting labeling. This idea can be used to develop a selective sampling algorithm for the nearest neighbor classifier.

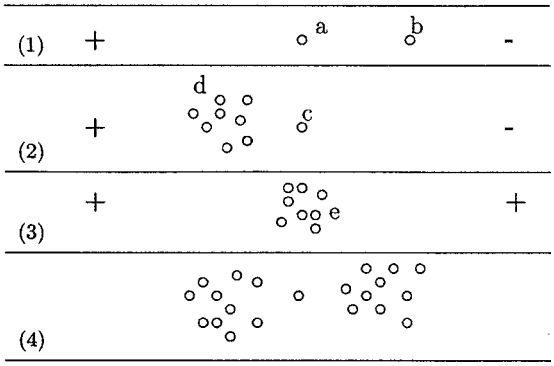

Figure 2. Various point configurations that may arise in the training set. Unlabeled points are marked by circles. (1) Single point near the border. (2) Group of points near the border. (3) Compact group of points far from labeled examples. (4) Two adjacent clusters, very far from labeled examples.

However, a careful look at the sampling style of this method exposes a number of intrinsic problems.

Consider the various point configurations shown in figure 2. These configurations are much simpler than the typical ones that may arise in practice, but they provide test cases for analyzing the behavior of selective sampling algorithms.

The boundary sampling will obviously prefer to sample point *a* over point *b* in configuration (1). This decision fits our intuition, since point *b* seems to be less important because of its proximity to the labeled point. Configuration (2), however, shows a weakness of the naive uncertainty sampling. In this configuration, the boundary method will sample in the border (point *c*). If point *c* will be labeled as '-', the uncertainty associated with the classification of cluster *d* will remain high. On the other hand, sampling at any point in cluster *d* will yield a classification with a high certainty for all its points. Thus, in configuration (2), sampling in cluster *d* is preferable over sampling point *c*.

These two examples demonstrate that selective sampling algorithms should consider not only the uncertainty of the candidate sample point, but also the effect of its classification on the remaining unlabeled points. Thus, sampling in dense regions may be preferred over sampling at an isolated point. Even when a compact group of unlabeled points is surrounded by instances of the same classification, as illustrated in configuration (3), it makes sense to sample in it, because the resulting label influences many points.

What if we can sample two or more instances in a row? Although we will select these points one by one (according to the framework described in Section 2.1), the very knowledge of the fact that we can sample more than one point can change our priorities in example selection. For example, in configuration (4), the two clusters may be of different classes. If we are allowed to sample only one instance, then selecting a point between the two clusters may be the best strategy. On the other hand, if we know that we will be able to request the label of another instance, we may be better off sampling in the center of one cluster (and sampling in the center of the other cluster afterwards).

The above examples show that the main consideration is not the uncertainty of the particular unlabeled point in the training set, but rather the effect that the labeling of this

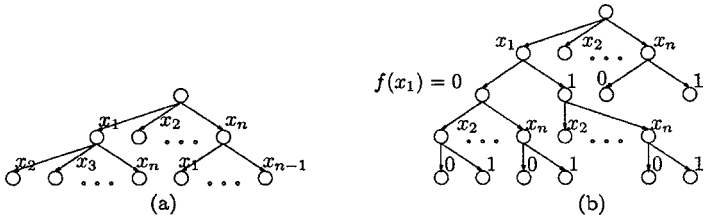

Figure 3. Part of the lookahead tree of depth 2, (a) without considering the teacher response and (b) considering the teacher response.

point may have on its neighborhood. The *lookahead* selective sampling algorithm described in the next section takes these considerations into account.

3. The lookahead algorithm for selective sampling

Using the intuition developed in the previous section, we now introduce the *lookahead* algorithm for selective sampling, which considers sampling sequences of length k and selects an example that leads to the best sequence (as illustrated in figure 3(a)). A simple way to evaluate the merit of the sequence is by estimating the utility of the selected points as training data for the classifier, regardless of their labels. For example, one may prefer sequences that uniformly sample the training set (according to its distribution).

One problem with this approach is that it does not take into account the possible responses of the teacher. An alternative approach views the selective sampling process as an interaction between the learner and the teacher. At each stage the learner must select an object from the set of unclassified instances and the teacher assigns one of the possible labels to the selected object. This interaction can be represented by a "game tree" such as the one shown in figure 3(b).

We use this tree representation to develop a lookahead algorithm for selective sampling. Let $U_L(D', D)$, where $D \subseteq D'$, be a *utility function* that estimates the merit of adding labeled instances $D' \setminus D$ to the set D as training examples for learning algorithm L. Let $P(f(x) = \omega \mid D)$ denote the conditional class probabilities of x for a given labeled set D. The k-deep lookahead algorithm for selective sampling with respect to learning algorithm L selects the example that leads to the learning sequence with the highest expected utility. This algorithm is presented in figure 4.

This algorithm is a specific case of a decision theoretic agent (Russell & Norvig, 1995). While it is specified for maximizing the expected utility, one can also be pessimistic and consider a *minimax* approach. A simplified version, associated with lower computational cost, is a one-step lookahead algorithm (see figure 5). In our implementation we use one-step and two-step lookahead algorithms, which maximize the expected utility.

The actual lookahead example selection scheme relies on two choices:

- The utility function $U_L(D', D)$;
- The method for estimating $P(f(x) = 0 \mid D)$ (and $P(f(x) = 1 \mid D)$).

These choices are considered in the next sections.

$\mathbf{S_L^k(X, D)}$:
Select $x' \in X$ with maximal expected utility:

$$x' = \arg\max_{x \in X} E[U_L^*(X \setminus \{x\}, D, D \cup \{\langle x, \omega \rangle\}, k-1)]$$

where $U_L^*(X, D, D', k)$ is a recursive utility propagation function:

$$U_L^*(X, D, D', k) = \begin{cases} U_L(D', D) & k = 0 \\ \max_x E[U_L^*(X \setminus \{x\}, D, D' \cup \{\langle x, \omega \rangle\}, k-1)] & \text{otherwise} \end{cases}$$

and the expected value $E[\cdot]$ is taken according to conditional probabilities for classifying x for a given D, $P(f(x) = \omega \mid D)$.

Figure 4. The k-deep lookahead algorithm.

$\mathbf{S_L^1(X, D)}$:
Select $x \in X$ with maximal expected utility, $E[U_L(D \cup \{\langle x, \omega \rangle\}, D)]$, which is equal to:

$$\sum_{w=0,1} P(f(x) = w \mid D) \cdot U_L(D \cup \{\langle x, w \rangle\}, D)$$

Figure 5. One-step lookahead algorithm.

4. Accuracy-based utility functions

There are two major approaches for evaluating the expected performance of the hypothesis resulting from adding the new example(s) to the dataset. One approach estimates the absolute expected accuracy of the hypothesis regardless of the current hypothesis. The other approach estimates the accuracy gain of a new hypothesis relative to the current one. The second method is different from the first one since it estimates the accuracy of the current hypothesis according to the updated data. The following subsections present two utility functions based on these approaches.

4.1. Absolute-accuracy utility function

Taking a Bayesian approach, we specify the utility of the classifier as its expected accuracy. Estimating this expected accuracy is a subtle issue: if we had known the true classifier (denoted "target"), then we could have used X to estimate the accuracy of any hypothesis. Here, however, the target is not known. Unlike the PAC approach, which assumes the worst case regarding the target concept, in the Bayesian framework we assume that the target function is drawn from a set of possible target functions (concept class) according to some fixed distribution. Given the labeled data, the posterior distribution over the target functions can be calculated from the prior distribution using Bayes' rule, and then the expected accuracy can be defined based on this posterior distribution. Explicit modeling of the prior distribution (of target functions) could be very complex. Fortunately, as we shall see later, we can calculate the expected accuracy without the explicit knowledge of the actual prior

distribution of target functions, based only on the (simpler) expressions of the posterior label probabilities at every point. A similar but more direct approach, which uses the distribution of targets, is taken by Freund et al. (1997).

First, consider a specific target f and a hypothesis h. Let $I_{h,f}$ be a binary indicator function, where $I_{h,f}(x) = 1$ if and only if $h(x) = f(x)$, and let $\alpha_h(f)$ be the accuracy of hypothesis h relative to f,

$$\alpha_h(f) = E_x[I_{h,f}(x)] = \int_{\mathbb{R}^d} I_{h,f}(x)\mathfrak{p}(x)\,dx. \tag{1}$$

Recall that $\mathfrak{p}(x)$ is the probability density function specifying the instance distribution over $\mathbb{R}^d$ (Section 2.1). Thus $\alpha_h(f)$ is simply the probability that $h(x)$ is equal to $f(x)$ on some random point x drawn from the distribution on $\mathbb{R}^d$. Note that $\alpha_h(f) = 1 - \mathrm{err}(h) = P[h(x) = f(x)]$.

Consider now the set $\mathfrak{F}_D$ of target functions f that are consistent with the sampled data D, i.e., $\mathfrak{F}_D = \{f \in \mathfrak{F} : \forall\langle x, \omega\rangle \in D,\ f(x) = \omega\}$ where $\mathfrak{F}$ denotes the a priori defined class of target functions (concept class). Let $\mathcal{A}_L(D)$ denote the expected accuracy of hypothesis $h = L(D)$, produced by a learning algorithm L on labeled data D. This expected accuracy is taken relative to the posterior distribution on $\mathfrak{F}_D$ given D, which is defined by Bayes' rule from the prior distribution on $\mathfrak{F}$. We shall assume that such a prior distribution exists even though we do not specify it. We have

$$\mathcal{A}_L(D) = E_f[\alpha_h(f)] = E_f[E_x[I_{h,f}]] = E_x[E_f[I_{h,f}]], \tag{2}$$

where $E_f[I_{h,f}]$ is the probability $P(f(x) = h(x) \mid D)$ that a random target function f consistent with D is equal to $h = L(D)$ at specific point x. Combining Eqs. (1) and (2) we get

$$\mathcal{A}_L(D) = \int_{\mathbb{R}^d} P(f(x) = h(x) \mid D)\mathfrak{p}(x)\,dx. \tag{3}$$

Suppose now that the class probabilities $P(f(x) = 0 \mid D)$ and $P(f(x) = 1 \mid D)$ are given. (The evaluation of such class probabilities is considered in Section 5.) Then the expected accuracy of hypothesis h with respect to instance distribution $\mathfrak{p}(x)$ can be calculated using Eq. (3). With a finite set of instances X drawn according $\mathfrak{p}$, this probability can be estimated by

$$\mathcal{A}_L(D) \approx \tilde{\mathcal{A}}_L(D) = \frac{1}{|X|}\sum_{x\in X} P(f(x) = h(x) \mid D). \tag{4}$$

The accuracy function, whose definition is based on the target distribution, was calculated without knowing the explicit form of this distribution. This could be done because it is not the specific distribution of targets that matters, but only the expected value of the target function at every point.

Equation (4) translates the problem of evaluating the expected classifier accuracy, which we have chosen as our utility measure, into the problem of estimating the class probabilities.

Once the class probabilities are calculated, the selective sampling strategy uses the utility function $U_L^{\mathrm{acc}}(D', D) \stackrel{\Delta}{=} \tilde{\mathcal{A}}_L(D')$. (The superscript "acc" stands for accuracy.)

4.2. *Gain-based utility function*

We now introduce an alternative, more exploratory utility function. This function prefers examples that maximize the expected gain in classifier accuracy. That is, it maximizes the difference between the expected accuracy of the new hypothesis $L(D')$ and the previous hypothesis $L(D)$. This expected accuracy is calculated on the basis of data D' and the implied label probabilities for both hypotheses. This is in contrast to the previous method, which essentially chooses D' such that the difference between the accuracy of $L(D')$, estimated relative to D', and the accuracy of $L(D)$, estimated relative to D, is maximum.

Suppose that we have labeled data D and that we wish to evaluate the merit of sampling the points $D' \setminus D$. Let $h = L(D)$ and $h' = L(D')$. Given a target function f, the increase in accuracy is

$$\epsilon_{h',h}(f) \stackrel{\Delta}{=} \alpha_{h'}(f) - \alpha_h(f) \tag{5}$$

and the expected accuracy gain given D' is

$$\begin{aligned}
\mathcal{G}_L(D', D) &\stackrel{\Delta}{=} E_f[\epsilon_{h',h}(f) \mid D'] = E_f[\alpha_{h'}(f) - \alpha_h(f) \mid D'] \\
&= E_f[E_x[I_{h',f}] - E_x[I_{h,f}] \mid D'] \\
&= E_x[E_f[I_{h',f} \mid D'] - E_f[I_{h,f} \mid D']] \\
&= \int_{\mathbb{R}^d} (P(f(x) = h'(x) \mid D') - P(f(x) = h(x) \mid D'))\mathfrak{p}(x)\,dx,
\end{aligned} \tag{6}$$

where the notation $E_f[a_f \mid D]$ stands for the expected value of random variable a_f (depending on f) taken over the distribution of target functions f consistent with D. Since $h'(x)$ and $h(x)$ are binary, $h'(x) \neq h(x)$ implies that $P(f(x) = h(x)) = 1 - P(f(x) = h'(x))$. Thus the integrand for $h'(x) = h(x)$ is zero, and we have

$$\mathcal{G}_L(D', D) = \int_{x \in \mathbb{R}^d, h(x) \neq h'(x)} (2P(f(x) = h'(x) \mid D') - 1)\mathfrak{p}(x)\,dx. \tag{7}$$

The expected accuracy gain is non-negative if the learning algorithm L chooses the best hypothesis from a fixed class of hypotheses, because L can always choose $h' = h$ if no better hypothesis is available. Moreover, even if the hypothesis class is not fixed, the expected accuracy gain is non-negative if the learning algorithm produces the Bayesian hypothesis (for which, at every point, the most a posteriori probable label is chosen). As we shall see, in the context of our model (Section 6), the nearest neighbor algorithm satisfies this assumption.

Since X was drawn randomly from $\mathbb{R}^d$ according to $\mathfrak{p}$, we can approximate $\mathcal{G}_L(D', D)$ by

$$\mathcal{G}_L(D', D) \approx \tilde{\mathcal{G}}_L(D', D) = \frac{1}{|X|} \sum_{x \in X, h(x) \neq h'(x)} (2P(f(x) = h'(x) \mid D') - 1). \tag{8}$$

We shall use $U_L^{\text{gain}}(D', D) \stackrel{\Delta}{=} \tilde{\mathcal{G}}_L(D', D)$ as an alternative utility function. This utility function tends to sample points that have the potential to radically change the current hypothesis. This may happen, for example, when the new sampled point provides information (about the expected labels) that reduces the expected accuracy of the previous classifier. The first proposed utility function, U_L^{acc}, is more conservative in this sense. This difference may lead to faster learning in some domains (see Section 7.4).

In this section we have solved the problem of estimating the utility function by focusing on determining the correct class probabilities from the labeled data. Once these probabilities are estimated, one can readily construct the utility function for a lookahead selective sampler. In the next section we introduce a realistic feature space model for estimating the class probabilities.

5. Random field model for feature space classification

Feature vectors from the same class tend to cluster in the feature space (though sometimes the clusters are quite complex). Therefore, close feature vectors share the same label more often than not. This intuitive observation, which is the rationale for the nearest neighbor classification approach, is used here to estimate the classes of unlabeled instances and their uncertainties.

Mathematically, this observation is described by assuming that the label of every point is a random variable, and that these random variables are mutually dependent. Such dependencies are usually described (in a higher than 1-dimensional space) by *random field models*. In the probabilistic setting, estimating the classification of unlabeled vectors and their uncertainties is equivalent to calculating the conditional class probabilities from the labeled data, relying on the random field model.

Thus, we assume that the classification of a feature space is a *sample function* of a binary valued *homogeneous isotropic random field* (Wong & Hajek, 1985) characterized by a *covariance* function that decreases with distance. A similar method was used by Eldar et al. (1997) for progressive image sampling. We let x_0, x_1 be points in $\mathcal{X}$ and θ_0, θ_1 be their classifications, i.e., random variables that can have values of 0 or 1. The homogeneity and isotropy properties imply that the expected values of θ_0 and θ_1 are equal, i.e., $E[\theta_0] = E[\theta_1] = \bar{\theta}$. These properties also imply that the covariance between θ_0 and θ_1 is specified only by the distance between x_0 and x_1:

$$Cov[\theta_0, \theta_1] = E[(\theta_0 - \bar{\theta})(\theta_1 - \bar{\theta})] \stackrel{\Delta}{=} \gamma(d(x_0, x_1)), \tag{9}$$

where $\gamma : \mathbb{R}^+ \to (-1, 1)$ is a covariance function with $\gamma(0) = Var[\theta] = E[(\theta - \bar{\theta})^2] = P_0 P_1$, and P_0, $P_1 = 1 - P_0$ are the a priori class probabilities. Usually we will assume that γ decreases with the distance and that $\lim_{r\to\infty} \gamma(r) = 0$. While specifying the covariance does not uniquely determine the random field and the distribution of target functions, it limits them considerably, and also accords them a certain similarity. If, for example, the covariance is substantial for close points and decreases with distance (as we assume), then the labels of close points are expected to be identical.

We shall now describe three ways of calculating estimates of these conditional probabilities, using the underlying random field model. The first two methods complement each other, while the third provides theoretical justification for the first two.

5.1. *Calculating the probabilities from correlation data by mean square estimation of the conditional mean*

In estimation, one tries to find the value of some unobserved random variable, from the observed values of other, related, random variables, and from prior knowledge about their joint statistics. Having only two classes implies that the class probabilities associated with some feature vector are uniquely specified by the conditional mean of its associated random variable (r.v.). This conditional mean is also the best estimator for the r.v. value in the least squares sense (Papoulis, 1991). Therefore, common methods for *mean square error* (MSE) estimation can be used for estimating the class probabilities.

We choose a linear estimator, for which a closed-form solution described below is available. Let θ_0 be the binary r.v. associated with some unlabeled feature vector, x_0, and let $\theta_1, \ldots, \theta_n$ be the (known) r.v. associated with the feature vectors, $x_1, \ldots, x_n$, which were already labeled. A linear estimator of the unknown label θ_0 is

$$\hat{\theta}_0 = \alpha_0 + \sum_{i=1}^{n} \alpha_i \theta_i. \tag{10}$$

The estimate uses the known labels and relies on the coefficients $\alpha_i, i = 1, \ldots, n$. The construction of such estimators is a common statistical procedure (Papoulis, 1991). The optimal linear estimator that minimizes the MSE $\epsilon_{\text{mse}} = E[(\hat{\theta} - \theta_0)^2]$ is

$$\hat{\theta}_0 = E[\theta_0] + \vec{\mathbf{a}} \cdot (\vec{\theta} - E[\vec{\theta}]), \tag{11}$$

where $\vec{\mathbf{a}}$ is an n-dimensional coefficients vector specified by the covariance values:

$$\begin{aligned} \vec{\mathbf{a}} &= \mathbf{R}^{-1} \cdot \vec{\mathbf{r}}, \\ R_{ij} &= E[(\theta_i - E[\theta])(\theta_j - E[\theta])], \\ r_i &= E[(\theta_0 - E[\theta])(\theta_i - E[\theta])]. \end{aligned} \tag{12}$$

($\mathbf{R}$ is an $n \times n$ matrix, and $\vec{\mathbf{a}}, \vec{\mathbf{r}}$ are $n-$dimensional vectors.) The values of $\mathbf{R}$ and $\vec{\mathbf{r}}$ are specified by the random field model:

$$\begin{aligned} R_{ij} &= \gamma(d(x_i, x_j)), \\ r_i &= \gamma(d(x_0, x_i)). \end{aligned} \tag{13}$$

The procedure is straightforward and easy to implement. In practice only reasonably close labeled points are used to construct the estimate, because distant points are not correlated and therefore cannot contribute anything to it, since this lack of correlation implies that

their corresponding coefficients become zero. Therefore this estimation procedure is also fast.

One problem with the linear estimation of label probability is that the range of the estimated values is not limited and may lie outside the [0, 1] interval. This may happen because the arbitrary specified covariance function may not correspond to the true correlation function on binary variables. With the interpretation of $\hat{\theta}$ as probability such values are clearly not valid. Fortunately, this is rarely the case. When such invalid values occur, they are clipped to either 0 or 1, resulting in a legal value, which also gives a more accurate estimate in the MSE sense for binary variables.

Another problem with linear estimators is that they use only the information on second-order statistics (covariance matrix), neglecting the information in the higher order statistics.

5.2. Direct calculation of conditional probabilities

The conditional distribution of θ_0, given $\theta_1, \ldots, \theta_n$, can also be found from the joint distribution of $\theta_0, \ldots, \theta_n$, since

$$\begin{aligned} P(\theta_0 \mid \theta_1, \ldots, \theta_n) &= p(\theta_0, \theta_1, \ldots, \theta_n) / p(\theta_1, \ldots, \theta_n) \\ &= p(\theta_0, \theta_1, \ldots, \theta_n) \Big/ \sum_{\omega \in \{0,1\}} p(\theta_0 = \omega, \theta_1, \ldots, \theta_n). \end{aligned} \tag{14}$$

Let $\vec{\omega} \triangleq (\omega_0, \omega_1, \ldots, \omega_n)$, $\vec{\omega} \in \{0, 1\}^{n+1}$ denote the vector of values of $\theta_0, \theta_1, \ldots, \theta_n$ and $p_{\vec{\omega}} \triangleq P(\theta_0 = \omega_0, \ldots, \theta_n = \omega_n)$ denote the probability of $\theta_0, \ldots, \theta_n$ to obtain these values. We can find the joint distribution of $\theta_0, \ldots, \theta_n$ from the moment data:

$$\begin{aligned} &\sum_{\vec{\omega} \in \{0,1\}^{n+1}} p_{\vec{\omega}} = 1 && \text{sum of prob. is one;} \\ &\sum_{\vec{\omega} \in \{0,1\}^{n+1}} \omega_i \, p_{\vec{\omega}} = P_1 \text{ (for } i = 0, \ldots, n) && \text{a priori probabilities;} \\ &\sum_{\vec{\omega} \in \{0,1\}^{n+1}} (\omega_i - P_1)(\omega_j - P_1) p_{\vec{\omega}} = \gamma(d(x_i, x_j)) && \text{for } i \neq j. \end{aligned} \tag{15}$$

There are 2^{n+1} unknown variables and only $1 + (n + 1) + \frac{n(n+1)}{2}$ equations, so a unique solution cannot be obtained for n larger than 1. This could have been expected, since the second-order statistics do not uniquely specify an arbitrary joint probability function. Additional constraints must be introduced in order to obtain a unique solution.

Knowing a number of higher moments, denoted by $\zeta_3, \ldots, \zeta_{n+1}$, gives additional constraints:

$$\sum_{\vec{\omega} \in \{0,1\}^{n+1}} \left[\prod_{k \in I} (\omega_k - P_1) \right] p_{\vec{\omega}} = \zeta_{|I|}(\mathbf{x}_{i \in I}) \tag{16}$$

for $I \subseteq \{0, \ldots, n\}, 3 \leq |I| \leq n+1, (n \geq 2)$. The constraints specified in Eq. (15) are the lower $\zeta_0, \zeta_1, \zeta_2$ moments. This linear system (Eqs. (15) and (16)) is specified by all moments and consists of $1 + (n+1) + \frac{n(n+1)}{2} + \sum_{i=3}^{n+1} \binom{n+1}{i} = \sum_{i=0}^{n+1} \binom{n+1}{i} = 2^{n+1}$ equations, as well as 2^{n+1} unknown variables denoting the joint probabilities of $\theta_0, \ldots, \theta_n$. Thus it can be uniquely solved. However, the intuition on which the nearest neighbor classifier is based implies only that the second-order moments are positive and decrease with distance. Making further assumptions on higher order moments constrains the distribution beyond the basic nearest-neighbor rationale, and may be unjustified. Also, arbitrarily chosen moments may be inconsistent with any binary r.v. distribution.

In a certain context, it may be natural to assume that both class probabilities are equal, i.e., $P_0 = P_1 = \frac{1}{2}$, and the distribution is "symmetric". That is,

$$\forall \vec{\omega} \in \{0, 1\}^{n+1},\ p_{\vec{\omega}} = p_{\neg\vec{\omega}}. \tag{17}$$

This readily implies that all odd moments are zero, i.e., $\zeta_{2k+1} \equiv 0, k \in \mathbb{N}$. Interestingly, the symmetry property (17) holds if and only if $P_0 = P_1$ and $\zeta_{2k+1} \equiv 0, \forall k \in \mathbb{N}, k \leq \lceil \frac{n+1}{2} \rceil - 1$.

This assumption is much stronger than just assuming that $P_0 = P_1$, but if it holds, it provides the additional third order moments, which suffice, and allows computation of the joint distribution of three random variables from Eqs. (15) and (17) alone.

In contrast to the linear estimation approach, which relies only on the second-order statistics of the distribution and is therefore non-optimal for non-Gaussian distributions, the direct method described above tries to calculate the exact joint distribution. Therefore, it requires more information, in the form of higher order statistical moments, but it guarantees that the estimated probabilities lie in the legal [0, 1] interval (provided that all the moments that are used are indeed correct).

Both methods are associated with some theoretical deficiency, to be explained and solved below.

5.3. A modified random field: The excursion set model

As mentioned above, there are some deficiencies in the probability estimation methods that we described. The MSE approximation of the conditional mean can yield an invalid estimate of the mean, outside the [0, 1] range. The direct calculation avoids this problem but requires the use of a set of higher order moments that are not necessarily justified by the nearest-neighbor rationale.

The major theoretical deficiency, as we see it, is that an arbitrarily specified covariance function, $\gamma(d)$, may not correspond to a *binary* random field. That is, there is no guarantee that there indeed exists some random field with binary outcomes and with the chosen covariance function. One way to solve this problem is to specify the labels indirectly, using an additional, *real*-valued random field. Now, from the distribution of this "hidden" random field, we can infer the covariance of the binary field.

Define a *Gaussian* random field as a collection of random variables $\mathrm{Y}(x), x \in \mathbb{R}^{\mathrm{d}}$, such that all *finite dimensional* distributions of variables $\mathrm{Y}(x)$ are multivariate Gaussians (Adler, 1981). For simplicity, we also assume that this random field is homogeneous and isotropic,

meaning that $E[\mathrm{Y}(x)] \equiv \bar{\mathrm{Y}}$ and the covariance between $\mathrm{Y}(x_1)$ and $\mathrm{Y}(x_2)$ depends only on the distance between x_1 and x_2, i.e., $Cov[\mathrm{Y}(x_1), \mathrm{Y}(x_2)] \stackrel{\Delta}{=} \gamma^*(d(x_1, x_2))$. In this way all finite dimensional distributions of Y's are uniquely defined by γ^* and $\bar{\mathrm{Y}}$.

Now we can model the (random) *concept class* (the set of all points of class 1 in $\mathbb{R}^{\mathrm{d}}$) to be the (random) *excursion set* (Adler, 1981) above the level 0:

$$A_0(\mathrm{Y}, \mathbb{R}^{\mathrm{d}}) = \{x \in \mathbb{R}^{\mathrm{d}} : \mathrm{Y}(x) \geq 0\} \tag{18}$$

of the real-valued random field $\mathrm{Y}(x)$, $x \in \mathbb{R}^{\mathrm{d}}$. Then, the binary class label is

$$\theta(x) = \begin{cases} 0, & \mathrm{Y}(x) < 0 \\ 1, & \mathrm{Y}(x) \geq 0. \end{cases} \tag{19}$$

For any value assignment, the joint probability $p(\theta_0 = \omega_0, \theta_1 = \omega_1, \ldots, \theta_n = \omega_n)$ may be calculated by (numerically) integrating the multivariate Gaussian distribution of $\mathrm{Y}_0, \ldots, \mathrm{Y}_n$:

$$p(\theta_0 = \omega_0, \ldots, \theta_n = \omega_n) = \int_{a_0}^{b_0} \cdots \int_{a_n}^{b_n} \frac{1}{(2\pi)^{\frac{n+1}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(\vec{\mathrm{Y}} - \mu_{\vec{\mathrm{Y}}})\Sigma^{-1}(\vec{\mathrm{Y}} - \mu_{\vec{\mathrm{Y}}})^t} d\vec{\mathrm{Y}}, \tag{20}$$

where $a_i = \begin{cases} -\infty, & \omega_i = 0 \\ 0, & \omega_i = 1 \end{cases}$, $b_i = \begin{cases} 0, & \omega_i = 0 \\ \infty, & \omega_i = 1 \end{cases}$ and $\Sigma, \mu_{\vec{\mathrm{Y}}}$ are defined by the model, that is, $\Sigma[i, j] = \gamma^*(d(x_i, x_j))$ and $\mu_{\vec{\mathrm{Y}}}[i] = \bar{\mathrm{Y}}$. The "hidden" random field model and the corresponding observed concept are illustrated in figure 6.

The γ^* function must be *non-negative definite*, that is, for any finite collection of $x_1, x_2, \ldots, x_k \in \mathbb{R}^{\mathrm{d}}$, the covariance matrix C, where $C_{ij} = \gamma^*(d(x_i, x_j))$, should be non-negative definite. Knowledge of the prior probabilities makes it possible to specify the mean $\bar{\mathrm{Y}}$ of a Gaussian distribution that satisfies the following relation:

$$P_1 = P(\theta = 1) = \int_0^{\infty} (2\pi\gamma^*(0))^{-\frac{1}{2}} e^{-\frac{1}{2}\frac{(t-\bar{\mathrm{Y}})^2}{\gamma^*(0)}} dt. \tag{21}$$

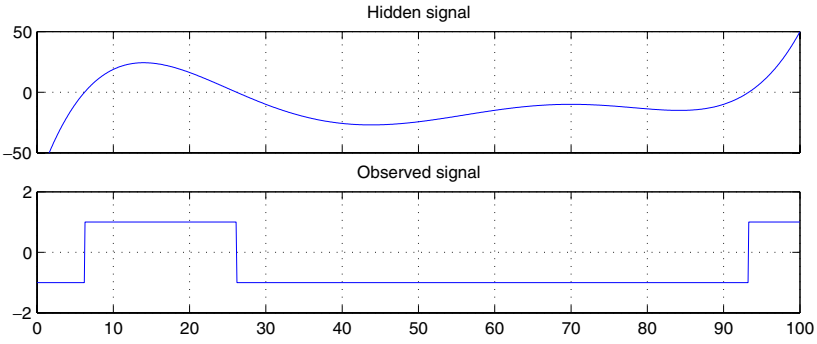

Figure 6. One-dimensional example of the relationship between the hidden field and the observed labels. Upper figure: the realization of the hidden field. Lower figure: the observed labeling function corresponding to the specific realization of the hidden field.

This relation may be considered as a condition defining the mean when the prior probability is known and the variance is specified. This framework resembles Gaussian Process Modeling (MacKay, 1998; Williams & Barber, 1998), although we use a different transformation to get binary r.v.

The nearest-neighbor method rationale may also be satisfied by specifying the covariance (associated with the "hidden" Gaussian process) as a decreasing function of the distance between the points. Such a specification induces (another) covariance associated with the binary field, which is also a decreasing function of the distance. Therefore, the intuitive properties of the field are satisfied. We show below that these covariance functions are not necessarily very different. The main purpose of introducing the excursion set is to make the specification of the covariance legal, i.e., to guarantee that there is indeed a binary random field with specified covariance. A secondary benefit of choosing the Gaussian distribution is that the random field is now specified by its second-order statistics, namely by the covariance function associated with the hidden r.v., and no higher order statistics are required.

The excursion set model may be used directly to produce predictions on the conditional probabilities of the labels. It is, however, associated with high computational cost. Another way to use it would be to derive the covariance function of the binary random field from it, thus guaranteeing its validity.

Consider the situation when $P_0 = P_1 = \frac{1}{2}$ ($\bar{Y} = 0$) and the exponential covariance function of the hidden, real-valued, random field is $\gamma^*(d) = e^{-\alpha d}$. In this case, it is possible to (numerically) calculate the covariance function of the binary random field $\gamma(d)$. As we can see in figure 7, the resulting function is very close to exponential. Thus, if we choose an exponential covariance function for a binary valued random field in an experimental implementation of our method (see Section 6), it is at least a very good approximation of a valid covariance, which is justified theoretically. This is the method adopted in LSS algorithm.

The given labeled points, through the random field model, induce posterior label probabilities everywhere in the instance space. A Bayesian decision may be made simply by choosing the label with the higher probability, as in Williams and Barber (1998). In general, the nearest neighbor classification is not necessarily consistent with this optimal decision. For example, the prior probabilities influence the Bayesian decision (through the random

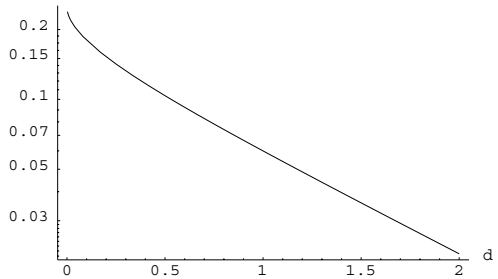

Figure 7. The log plot for the covariance function of binary r.v., $\gamma(d)$, computed numerically from the covariance function of the underlying Gaussian random field, with $\gamma^*(d) = e^{-d}$ ($\alpha = 1$). Except for very small values of d, the plot is almost linear indicating that $\gamma(d)$ can be approximated by an exponential function.

field model) but not the NN decision. Also, certain configurations of labeled points, may give the label "1" a probability larger than 0.5 to some point, even if its nearest neighbor is labeled "0" (and vice versa). Still, the NN classifier is often preferable because it is simple to implement, and asymptotically errs with a bounded rate that is at most twice the Bayesian error.

6. Lookahead selective sampling using a random field model

The previous sections describe a general framework for lookahead selective sampling as well as a spectrum of methods for probability estimation. In this section an instance of the proposed algorithm that is used in our experiments is described.

The motivation for selective sampling is to substantially reduce the number of labeled examples and still get a reasonable classifier. Therefore, we are mostly interested in the initial stage of the sampling, where the class probabilities cannot be estimated well, and the distance between the samples is large. Formally, every sampled point influences the estimated probability. However, such long-range influence is non-intuitive and is also computationally expensive. Therefore, in practice, we neglect the influence of all except the closest neighbors. Choosing only the two closest labeled points has the theoretical advantage that the Bayesian decisions are the same as the decision made by the NN classifier (see the end of Section 5).

The proposed simple one-step lookahead selective sampling algorithm uses the accuracy utility function and the random field model to estimate the conditional class probabilities. The random field model is designed in the context of the nearest neighbor algorithm and is thus biased towards selecting informative examples for nearest neighbor classifiers. The specific implementation of random field model for our selective sampling algorithm is based on the following two assumptions, which are reasonable in the context of this initial sampling stage and simplify the assessment of the class probabilities:

- The a priori class probabilities are equal.
- The influence of all except the two closest nearest neighbors on the conditional probability of the given point is negligible.

Following the derivations presented in Section 5.1 and using the above assumptions we get:

$$
\begin{aligned}
P(\theta_0 = 1 \mid \theta_1 = 0, \theta_2 = 0) &= \frac{1}{2} + \frac{-\gamma(d_{01}) - \gamma(d_{02})}{\frac{1}{2} + 2\gamma(d_{12})} \\
P(\theta_0 = 1 \mid \theta_1 = 0, \theta_2 = 1) &= \frac{1}{2} + \frac{-\gamma(d_{01}) + \gamma(d_{02})}{\frac{1}{2} - 2\gamma(d_{12})} \\
P(\theta_0 = 1 \mid \theta_1 = 1, \theta_2 = 0) &= \frac{1}{2} + \frac{\gamma(d_{01}) - \gamma(d_{02})}{\frac{1}{2} - 2\gamma(d_{12})} \\
P(\theta_0 = 1 \mid \theta_1 = 1, \theta_2 = 1) &= \frac{1}{2} + \frac{\gamma(d_{01}) + \gamma(d_{02})}{\frac{1}{2} + 2\gamma(d_{12})}.
\end{aligned}
\tag{22}
$$

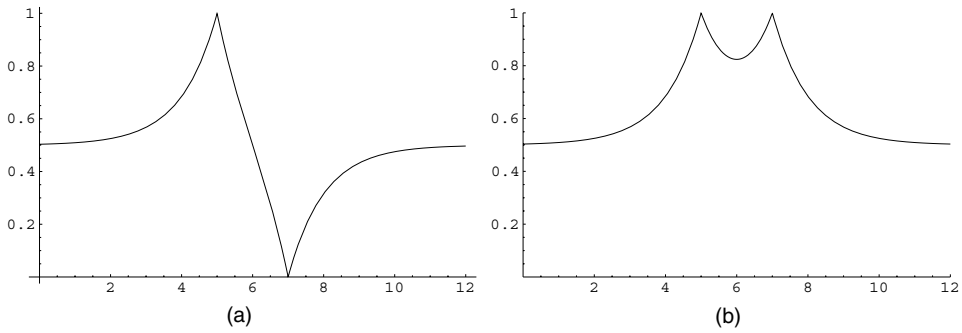

Figure 8. The conditional probabilities of class 1 as a function of location, in one dimension, $\gamma(d) = 0.25e^{-d}$ ($\sigma = 1$): (a) Point of class 1 at $x = 5$ and point of class 0 at $x = 7$. (b) Two points labeled as class 1 at $x = 5$ and $x = 7$.

These values are equal to the values we get by "direct" probability calculations (Section 5.2, Eqs. (15) and (16)) and by assuming $\zeta_3 \equiv 0$ or by inferring it from the "symmetry" property (Eq. (17)). In this way we rely only on second order statistics, which are used by the linear estimation approach.

We choose an exponentially decreasing covariance function $\gamma(d) = 0.25e^{-d/\sigma}$. Such a function is a good approximation for a covariance function, which can be computed numerically from the excursion set model (Section 5.3), thus providing theoretical justification for our approach. Given $P_0 = P_1 = \frac{1}{2}$ and $\gamma(d) = 0.25e^{-d/\sigma}$, the Bayesian classification of x_0 is defined by the label of x_1, i.e., $P(\theta_0 = 1 \mid \theta_1, \theta_2) > \frac{1}{2} \Leftrightarrow \theta_1 = 1, (d_{01} \neq d_{02})$, and is thus being consistent with the nearest neighbor classification. See figure 8 for an illustration.

Assuming $\gamma(d)$ to approximate the "correct" covariance function derived from an excursion set model does not guarantee that $P(\theta_0 \mid \theta_1, \theta_2)$ will be in the valid range of [0, 1]. To show the correctness of the choice of γ we prove the following:

Proposition 1. *Let* $P(\theta_0 \mid \theta_1, \theta_2)$ *be as defined by Eq.* (22), *then*

$$\forall x_0, x_1, x_2 \in \mathbb{R}^{\mathrm{d}} : \ 0 \leq P(\theta_0 \mid \theta_1, \theta_2) \leq 1. \tag{23}$$

Proof: The main step is to prove that $\frac{\gamma(d_{01}) - \gamma(d_{02})}{\frac{1}{2} - 2\gamma(d_{12})} \leq \frac{1}{2}$ (the rest is similar or trivial). After substituting $\gamma(d) = \frac{1}{4}e^{-d}$ (omitting σ as a scaling parameter) and simplifying, it must be shown that: $e^{-d_{01}} - e^{-d_{02}} + e^{-d_{12}} \leq 1$. Using the triangular inequality $d_{02} \leq d_{01} + d_{12}$ and the fact that $e^{-d_{ij}} \leq 1$ (for $d_{ij} \geq 0$), we get: $e^{-d_{01}} - e^{-d_{02}} + e^{-d_{12}} \leq e^{-d_{12}} + e^{-d_{01}} - e^{-d_{01}} e^{-d_{12}} \leq 1$. □

Putting everything together, we get the LSS (*Lookahead Selective Sampling*) algorithm, which is depicted in figure 9. The algorithm needs a specification of the characteristic distance σ, which we set depending on the average pair-distance in the unlabeled training

LSS(X,D):

1. If D is empty, return a random point from X.
2. Otherwise, set $U_{\max} \leftarrow 0$.
3. For each $x \in X$ do:
   (A) $D' \leftarrow D \cup \{\langle x, 0\rangle\}$.
   (B) Compute class probabilities for all points in X based on data D', using Eq. (22).
   (C) Compute utility by approximating the accuracy of the classifier based on data D' using computed class probabilities and Eq. (4), $U_0(x) \leftarrow \mathcal{A}(D')$.
   (D) Repeat the above steps for $D' \leftarrow D \cup \{\langle x, 1\rangle\}$ and get $U_1(x)$.
   (E) Compute class probabilities for x based on data D, using Eq. (22).
   (F) $U(x) \leftarrow P(f(x) = 0 \mid D) \cdot U_0(x) + P(f(x) = 1 \mid D) \cdot U_1(x)$.
   (G) If $U_{\max} < U(x)$ then $U_{\max} \leftarrow U(x)$, $x_{\text{best}} \leftarrow x$.
4. Return x_{best}.

Figure 9. The LSS algorithm.

set and on a scaling parameter $\mathfrak{D}$.

$$\sigma = \frac{1}{\mathfrak{D}} \cdot \frac{1}{|X|^2} \sum_{x_1 \in X} \sum_{x_2 \in X} d(x_1, x_2). \tag{24}$$

The $\mathfrak{D}$ parameter is set to 4 in all the experimental domains. Moreover, we found that its exact value does not effect the algorithm substantially (see Section 7.3).

The time complexity of the described algorithm is $O(|X|^2)$ for the straightforward implementation. This time complexity, however, may be reduced by selecting a random subset $X' \subseteq X$ and working with it instead of X, thus managing the time/performance trade-off of this algorithm.

7. Experimental evaluation

We have implemented LSS and tested it on several problems, comparing its performance with a number of other selective sampling methods in the context of nearest neighbor classification. Specifically, under the assumption of limited resources (i.e., the teacher is not able to label more than M examples, see Section 2.1), the algorithms were compared by the quality (accuracy) of the nearest neighbor classifiers they produce after sampling 10% of the available (unlabeled) training set. We have hypothesized that our algorithm would outperform other algorithm in terms of classification accuracy under the above settings due to its sophisticated lookahead approach. Additional experiments were conducted to test the effect of various parameters on the algorithm's performance.

7.1. Experimental methodology

The LSS algorithm (with $\mathfrak{D} = 4$) was compared with the following three selective sampling algorithms. These algorithms represent the most common choices found in the literature (see Section 1).

- *Random sampling*: The algorithm randomly selects the next example. While this method seems unsophisticated, it has the advantage of yielding a uniform exploration of the instance space. This method actually corresponds to a *passive* learning model.
- *Uncertainty sampling*: The method selects the example about which label the current classifier has the highest uncertainty. This is one of the most common choices for selective sampling. No specific uncertainty sampling method, however, was proposed for the nearest neighbor classifiers. We defined the uncertainty for each example as a probability of its misclassification, which is estimated by the ratio between the distances to the closest labeled neighbors of different classes

$$\begin{aligned} P_0(x) &= \frac{d_1}{d_0 + d_1},\ P_1(x) = 1 - P_0(x), \\ d_0 &= \min_{\langle s,0\rangle \in D} d(x, s), \\ d_1 &= \min_{\langle s,1\rangle \in D} d(x, s). \end{aligned} \tag{25}$$

- *Maximal distance sampling*: An adaptation of the membership query method described by Hasenjager and Ritter (1998) for nearest neighbor classification. This method selects the example from the set of such unlabeled points that for each point in the set its three nearest classified neighbors do not all have the same label. The example selected is the one which is most distant from its closest labeled neighbor.

The experiments were conducted on seven datasets. A short description of the datasets is presented below:

(a, b, c) *The Pima Indians*, *Ionosphere* and *Image Segmentation datasets*: These natural datasets were taken from UCI Machine Learning Repository (Blake, Keogh, & Merz, 1998). The classes in the Image Segmentation dataset were joined to create two class labels, one corresponding to BRICKFACE, SKY and FOLIAGE and the other corresponding to CEMENT, WINDOW, PATH and GRASS. The data in these datasets has already been split into training and test sets, which we used in the experiments.

(d) *The letters dataset*: This dataset is also available from the UCI Machine Learning Repository. As in the Image Segmentation dataset, the 26 classes of letters were joined to create two class labels, one corresponding to 'a'-'m' and the other corresponding to 'n'-'z'. This dataset, consisting of 20,000 instances, was randomly divided into 10 disjoint pairs of training and test sets (1000 examples each), and 10 runs were conducted on each pair, resulting in total of 100 runs.

(e) *The two spirals problem*: This artificial problem was included for a comparison, since it was used by Hasenjager and Ritter (1998). The task is to distinguish between two spirals in the XY plane. We used code available from Lang and Witbrock (1988) as a basis for our data generation program. In this and other artificial datasets the training and test sets were generated independently for each run.

(f, g) *The two Gaussians* and *Multi-Gaussian problems*: These artificial datasets were constructed to demonstrate the advantage of the LSS algorithm in domains consisting of

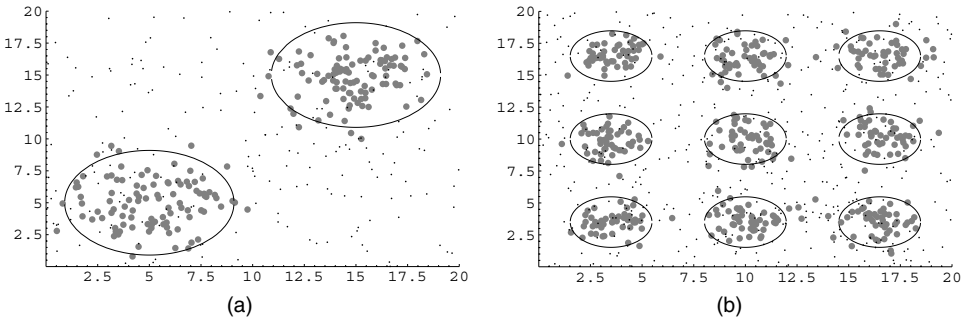

Figure 10. The feature spaces of the Two Gaussians (a) and the Multi-Gaussian (b) problems. Bayesian decision boundaries are shown.

more than one region with the same classification. The feature spaces of the Two Gaussians and the Multi-Gaussian problems consist of two-dimensional vectors belonging to two classes with equal a priori probability (of 0.5). The distribution of class 1 is uniform over the whole region and the distribution of class 0 consists of two symmetric Gaussians in the Two Gaussian problem and of nine symmetric Gaussians in the Multi-Gaussian problem. The feature spaces are illustrated in figure 10.

The dataset information is summarized in Table 1. It shows the number of attributes for each dataset, the sizes of (disjoint) training and test sets, the error rate of the best possible (Bayesian) classifier, and the error rate of the nearest neighbor classifier based on all the labeled points of the training set. The Bayesian error is only available for the artificial datasets, since the true probabilistic models that have generated the natural datasets are not known.

While all the examples in the datasets are already labeled, the training sets are treated as unlabeled data sets with uniform labeling cost. Also, if we would have to use these data sets in real settings, the labeling cost of each could be quite significant. For example, in the

Table 1. Dataset information includes the number of attributes, sizes of training and test sets, the least possible error (Bayesian error) and NN error (the test set error of nearest neighbor classifier given all the points in the training set).

Dataset	Number of attributes	Training set size	Test set size	Bayesian error	NN error (%)
Pima Indians	7	200	332	NA	31.0
Ionosphere	34	200	151	NA	7.9
Image segmentation	18	210	2100	NA	4.7
Letters	16	1000	1000	NA	11.6
Two spirals	2	1000	1000	0.0%	9.1
Two Gaussians	2	1000	1000	18.2%	27.3
Multi-Gauss	2	1000	1000	20.8%	29.9

Pima Indians domain, labeling an example would involve a non-trivial medical procedure for testing diabetics. Another example is the Image Segmentation domain, where labeling would require a human expert to determine the texture type.

The basic quantity measured in the experiments was the average error rate of the classifier based on the training points selected by the selective sampling algorithm. The following procedure was applied:

1. The training set and the test sets were obtained from the data. All the natural datasets which we used were already divided into training and test sets according to past usage. In the artificial datasets, training and testing sets of 1000 elements each were generated independently. The training set was used as an *unlabeled training set*, X, for a selective sampling algorithm. The test set was used *only* for the evaluation of error rates of the resulting classifiers.
2. The selective sampling algorithms were applied to the training set, X. After selecting each example, the error rate of the current hypothesis, h (which is the nearest neighbor classifier), was calculated using the test set of examples put aside.
3. Steps 1 and 2 were performed 100 times and the average error rate and standard deviation of each sampling method was approximated by maximum likelihood estimation, assuming that the accuracy of the resulting classifiers is normally distributed for each domain (DeGroot, 1986).

While the training and test sets for natural datasets remain the same, we are still interested in the average performance, since all non-trivial selective sampling methods are naturally randomized by the selection of the first example.

7.2. The performance of the lookahead selective sampling

The performance of the various selective sampling methods on the tested datasets in terms of average error rate and standard deviation is summarized in Tables 2 and 3. The learning curves for the Ionosphere dataset and for the Two Gaussians problem are presented in figure 11. Based on these results the following observations can be made:

- *The natural datasets (Pima Indians, Ionosphere and image segmentation)*: The LSS algorithm is superior by both performance measures. It has a better average performance and a more stable one than other selective sampling methods (Tables 2 and 3). Interestingly, in the Pima Indians Diabetes dataset, learning only 10% of the training set with the LSS algorithm yields a better classifier than the nearest neighbor classifier trained on all available points (compare with Table 1). Also, in the Ionosphere dataset the LSS algorithm requires only 8% of the training set in order to achieve the same average accuracy as the nearest neighbor classifier based on all the training points (figure 11(a)).
- *Letters dataset*: The LSS algorithm slightly outperforms other selective sampling methods in the particularly hard Letters dataset, where every class consists of many different subclasses (associated with the different letters).

Table 2. Estimated average error rates with 95% confidence interval for nearest neighbor classifiers based on sampling 10% of the training set.

Dataset	LSS (%)	Max. Dist. (%)	Uncertainty (%)	Random (%)
Pima Indians diabetes	26.9 ± 0.3	29.8 ± 1.1	30.3 ± 1.6	30.2 ± 0.9
Ionosphere	7.6 ± 0.4	19.8 ± 1.8	30.3 ± 4.3	18.5 ± 1.8
Image segmentation	10.9 ± 0.3	26.6 ± 1.9	14.8 ± 1.0	16.9 ± 1.1
Letters	28.2 ± 0.4	29.6 ± 0.8	28.6 ± 0.5	30.8 ± 0.3
Two spirals	26.3 ± 0.4	24.5 ± 1.0	26.1 ± 0.5	27.9 ± 0.4
Two Gaussians	24.9 ± 0.4	31.4 ± 1.3	32.7 ± 1.2	27.3 ± 0.5
Multi-Gaussian	33.6 ± 0.5	38.5 ± 0.9	39.4 ± 0.6	35.5 ± 0.6

The statistics are based on 100 runs. The results show that in most cases the LSS algorithm outperforms other selective sampling algorithms in terms of average error rates of the resulting classifiers.

Table 3. Estimated standard deviations with 95% confidence intervals for error rates of nearest neighbor classifiers based on sampling 10% of the training set.

Dataset	LSS (%)	Max. Dist. (%)	Uncertainty (%)	Random (%)
Pima Indians diabetes	1.7 ± 0.3	5.3 ± 0.8	7.8 ± 1.2	4.4 ± 0.7
Ionosphere	2.2 ± 0.3	9.2 ± 1.4	21.7 ± 3.3	8.9 ± 1.3
Image segmentation	1.4 ± 0.2	9.3 ± 1.4	5.0 ± 0.7	5.5 ± 0.8
Letters	2.0 ± 0.3	4.0 ± 0.6	2.3 ± 0.3	1.7 ± 0.3
Two spirals	2.0 ± 0.3	5.1 ± 0.8	2.5 ± 0.4	2.1 ± 0.3
Two Gaussians	2.1 ± 0.3	6.5 ± 1.0	6.0 ± 0.9	2.6 ± 0.4
Multi-Gaussian	2.6 ± 0.4	4.6 ± 0.7	3.1 ± 0.5	2.9 ± 0.4

The statistics are based on 100 runs. Results of LSS algorithm show that it is more stable than its competitors.

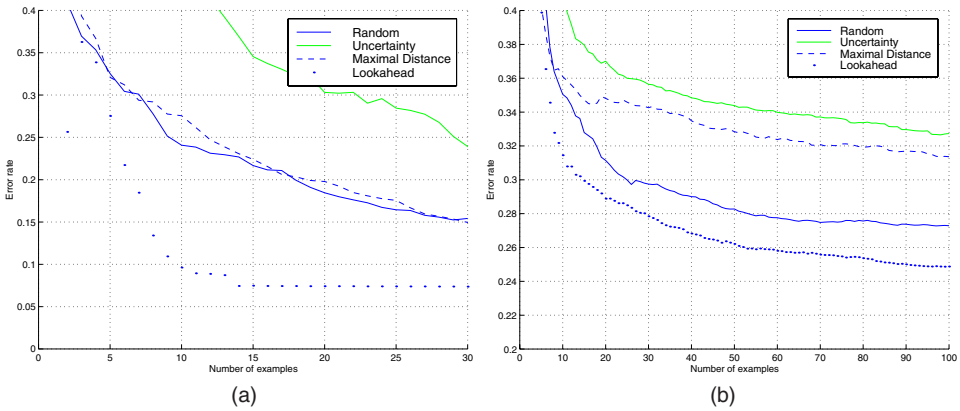

Figure 11. Learning curves of the selective sampling methods for (a) the Ionosphere dataset and (b) the Two Gaussian problem. Learning all points in the training set gives an error of 7.9% for the Ionosphere dataset and 27.3% for the Two Gaussians problem.

- *Two Spirals problem*: All three non-random methods performed comparably in terms of average error rate on the Two Spirals problem (Table 2), with the maximal distance method being slightly better than the others. On the other hand, the maximal distance method is the most unstable one (see Table 3).
- *Gaussian problems*: We can see from figure 11(b) that the uncertainty and maximal distance selective sampling methods apparently fail to detect one of the Gaussians, resulting in higher error rates. The variance of the LSS algorithm is much lower on both datasets, for the same reason: the LSS algorithm *always* detects both regions (or in the case of Multi-Gaussian problem, most of the regions) of the same classification, while the uncertainty and the maximal distance methods detect both of them only occasionally, thus creating greater variance in the quality of the resulting classifiers (Table 3). The uncertainty and the maximal distance selective sampling algorithms experience failure due to the fact that they consider sampling only at the existing boundary, while in these domains *exploration* is essential for good performance. The random sampling method is less affected by this problem.

The running times of the selective sampling algorithms were quite different from each other. Naturally, our LSS algorithm, being the most sophisticated, was also the slowest one. However, the typical query time for our algorithm was still less than a second, which is quite acceptable in the natural setup, with a human expert labeling the examples.

7.3. The effect of the covariance function on the performance of LSS

We carried out additional experiments to determine how much our algorithm depends on the choice of the covariance function, and in particular, how much it depends on the choice of the scaling parameter $\mathfrak{D}$. In the experiments reported in the previous section, $\mathfrak{D}$ was set to 4. In this section the scale parameter was set to be twice and four times larger and smaller than those in the original experiments, i.e., $\mathfrak{D} = 1, 2, 8$ and 16. This change increases and decreases the actual influence range of labeled points.

The dependence of the average error rate on $\mathfrak{D}$ is shown in Table 4, where the results for the Ionosphere and Image Segmentation datasets, which are typical for the rest of data, are summarized. These results demonstrate the stability of the lookahead algorithm over a wide range of values for the scale parameter. In addition, the variance of the error rates of the resulting classifiers decreases with $\mathfrak{D}$. This could have been expected, since increasing $\mathfrak{D}$ decreases the actual influence range of labeled points (see Eq. (24)). A higher $\mathfrak{D}$ will then result in a more local sampling strategy, which depends less on the initial random example.

Table 4. Estimated average error rates and their standard deviations of classifiers based on 10% of the training set produced by LSS algorithm with various $\mathfrak{D}$ parameter values.

Dataset scale	$\mathfrak{D} = 1$ (%)	$\mathfrak{D} = 2$ (%)	$\mathfrak{D} = 4$ (def.) (%)	$\mathfrak{D} = 8$ (%)	$\mathfrak{D} = 16$ (%)
Ionosphere	10.6, 6.4	7.4, 3.3	7.6, 2.2	7.2, 0.6	8.4, 0.6
Image segm.	15.5, 5.4	13.8, 5.1	10.9, 1.4	10.0, 0.7	11.6, 0.5

Statistics are based on 100 runs.

Table 5. Estimated average error rates for the lookahead selective sampling algorithm using the two alternative utility functions.

Utility	Pima Indians (%)	Ionosphere (%)	Image segm. (%)	Letters (%)	Two spirals (%)	Two Gauss. (%)	Multi Gauss (%)
U_L^{acc} (std.)	26.9	7.6	10.9	28.2	26.3	24.9	33.6
U_L^{gain}	30.0	6.7	9.5	27.5	25.0	24.2	34.0

7.4. The effect of the utility function on the performance of LSS

In Section 4.2 we proposed an alternative exploratory utility function, U_L^{gain}. We performed an experiment to compare this function to our standard utility function U_L^{acc}. The results are shown in Table 5. For most datasets the exploratory utility function (U_L^{gain}) yields a slight improvement over the accuracy-based utility function. A notable exception is the noisy Pima Indians Diabetes dataset, where the more conservative accuracy-based utility function has an advantage, because the more exploratory, gain-based utility function tends to select an example that changes the current hypothesis the most. In the noisy domains, such an example often turns to be mislabeled.

7.5. Experiments with the two-step lookahead sampling

The intuition developed in Section 2.3 (figure 2, configuration 4) shows that it may be beneficial to consider the lookahead sampling deeper than one example, as described by the lookahead algorithm in figure 4. We conducted such experiments for the two-step lookahead algorithm. Since the two-step lookahead algorithm has a large computational complexity, it was tested only on three real-world domains, with 10 runs on each (in the standard set of experiments the results were averaged over 100 runs).

The results are reported in Table 6. We can see that in these cases the two-step lookahead barely reduces the classification error of the resulting classifiers, but the results are achieved with much greater stability.

7.6. Summary of the experimental results

The experiments show that the lookahead sampling method performs better than or comparable to other selective sampling algorithms on both real and artificial domains. It is

Table 6. Experiments with two-step lookahead. Maximum likelihood estimates for average error rates and their standard deviations for classifiers based on selecting 10% of the training set.

Selective sampling method	Pima Indians (%)	Ionosphere (%)	Image segm. (%)
Lookahead, one-step deep	26.9, 1.7	7.6, 2.2	10.9, 1.4
Lookahead, two-steps deep	26.8, 1.6	7.2, 1.2	10.7, 0.2

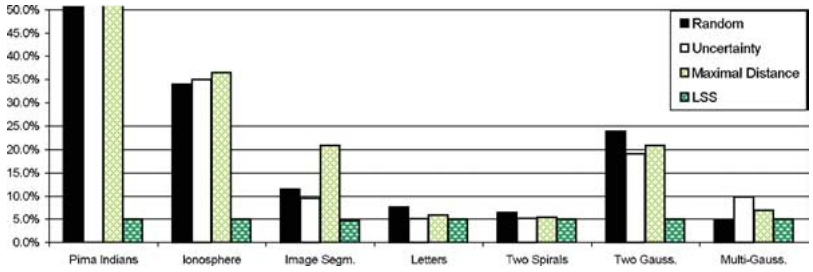

Figure 12. Comparing the number of examples (i.e., percent of the training set) needed for the average learning error of various methods to reach that attained by the LSS algorithm at 5% of the training sets. In the Pima Indians domain, the other selective sampling methods never achieve the accuracy obtained by the LSS algorithm.

especially strong when the instance space contains more than one region of the same class, as in the Two Gaussians and the Multi-Gauss datasets.[1] In this case, a selective sampling algorithm must consider not only the examples from the hypothesis boundary, but also explore large unsampled regions. The lack of an "exploration" element in the uncertainty and maximal-distance sampling methods often results in their failure.

The advantage of a lookahead selective sampling method can be seen by comparing the number of examples needed to reach some pre-defined accuracy level. Figure 12 shows the number of examples needed for various selective sampling algorithms to reach the average error level attained by the lookahead selective sampling using only 5% of the training set. In the Pima Indians domain, the other selective sampling methods never achieve this accuracy due to noise in the training set.

The experimental results for the Pima Indians Diabetes dataset are especially interesting when compared with the results of Wilson and Martinez (2000) who explored filtering techniques for reducing the size of the stored set of examples used for instance based learning. The LSS algorithm, sampling only 10% of the training set, achieves accuracy of 73.1% (Table 2). The IB3 and DEL filtering algorithms achieve accuracy of 69.8% and 71.6% respectively (Table 1 in Wilson and Martinez (2000)) when retaining datasets of approximately the same size. This better performance of our algorithm was achieved despite that IB3 and DEL have used perfect information (i.e., all the labels were available to them).

The experiments indicate that the proposed selective sampling approach is more stable than the alternatives in the following two aspects:

Stability within a domain: As apparent from the variance of the error rates of the resulting classifiers, the lookahead selective sampling algorithm is the most stable one with regard to different training sets within a domain and with regard to the different choices of the first example (Table 3).

Stability across domains: The results, described in Table 2, show that while the second best selective sampling method changes, the lookahead selective sampling algorithm remains the best and the most stable among the compared methods in the majority of datasets.

8. Discussion

In many real-world domains unlabeled examples are available in abundance, but it is expensive to label a large number of them for training. A possible solution to this problem, investigated in this paper, is to enable the learning algorithm to automatically select the training examples to be labeled from an unlabeled training set. In the context of nearest neighbor classification, this paradigm can be viewed as one that uses example filtering in addition to the *selective utilization* filtering implemented in the IB3 algorithm (Aha, Kibler, & Albert, 1991).

This paper proposes a *lookahead* framework for selective sampling, which selects an optimal example in the Bayesian sense. This framework, along with a novel, *random field* model of the instance space labeling structure, are the major contributions of this research to the field of machine learning.

The random field model that we use was inspired by the rationale for the nearest neighbor classifier. Nearest neighbor classifiers are often used when little or no information is available about the feature space structure. The loose, minimalistic specification of the feature space labeling structure implied by the distance-based random field model seems appropriate in this case. We also observe that large changes in the covariance function had no significant effect on the classification performance.

Our algorithm, however, has a number of deficiencies, which can be addressed in future research. If we consider the classification of one point (including finding 1 or 2 labeled neighbors) to be a basic operation, the uncertainty and the maximal distance methods have time complexity of $O(|X|)$ while the straightforward implementation of the lookahead selective sampling has a time complexity of $O(|X|^2)$. (We need to compute class probabilities for all points in the unlabeled training set after each lookahead hypothesis.) This higher complexity, however, is well justified for a natural setup, where we are ready to invest computational resources to save time for a human expert whose role is to label examples. In practice, a maximal time for example selection observed in our experiments was a couple of seconds, and typically it was a fraction of a second, which is quite reasonable for real-world applications.

Another deficiency, and a possible direction for further research, lies in the application of the random field model as a probability estimation tool for a nearest neighbor classification. As mentioned in Section 6, this model is consistent with NN classifier only if we can disregard the influence of all labeled neighbors other than the two that are closest to the current point of interest. In some cases, however, taking only two neighbors into account is not an optimal strategy.

Consider, for example, n labeled points distributed uniformly within a unit ball of d dimensions, and suppose the point of interest lies in the center of the ball. The higher the dimensionality, the larger part of the ball weight tends to be on the outer sphere; hence the difference in distances from the point of interest to the closest labeled points in the sphere decreases with dimensionality. This example supports the claim that in order to define the classification of the point of interest in higher dimensions, one should consider a larger number of nearest neighbors.

A natural extension of this research is the application of lookahead sampling to other models of feature space classification structure and to other classification algorithms. Another extension of this work is the usage of techniques developed here for the reduction of

stored examples in instance-based learning algorithms. This direction looks very promising in light of comparison of the results presented here with the results of Wilson and Martinez (2000) (see Section 7.6).

The issue of reducing the time complexity of lookahead sampling should also be further investigated. One possibility is to define the cost model, i.e., the amount of resources/time one can allow in order to select the next example. The cost model can take into account the varying labeling costs, providing an important extension of this research for domains with labeling costs that are not uniform throughout the instance space.

We believe that the research presented in this paper is a significant contribution to the emerging field of selective sampling, whose relevance to modern data mining applications is growing.

Acknowledgments

The authors would like to thank Neri Merhav for helpful discussions. Portions of this work were presented at the Sixteenth National Conference on Artificial Intelligence (Lindenbaum, Markovitch, & Rusakov, 1999). Michael Lindenbaum was supported in part by the Israeli Ministry of Science, Grant no. 2104.

Note

1. The Letters dataset looks also as one with many disjuncts since each class contains 13 letters. We are not sure, however, what is exactly the shape of the instance space as represented by the particular features used for this dataset.

References

Adler, R. J. (1981). *The Geometry of Random Fields*. John Wiley & Sons.

Aha, D. W., Kibler, D., & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning*, *6:1*, 37–66.

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2:3*, 319–342.

Blake, C., Keogh, E., & Merz, C. (1998). UCI Repository of machine learning databases, University of California, Irvine. [http://www.ics.uci.edu/∼mlearn/MLRepository.html].

Cohn, D. A., Atlas, L., & Lander, R. (1994). Improving generalization with active learning. *Machine Learning*, *15:2*, 201–221.

Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1995). Active learning with statistical models. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in Neural Information Processing Systems*, vol. 7 (pp. 705–712). The MIT Press.

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13:1*, 21–27.

Dagan, I., & Engelson, S. P. (1995). Committee-based sampling for training probabilistic classifiers. In A. Prieditis, & S. Russell (Eds.), *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 150–157). Morgan Kaufmann.

Davis, D. T., & Hwang, J.-N. (1992). Attentional focus training by boundary region data selection. In *Proceedings of International Joint Conference on Neural Networks*, vol. 1 (pp. 676–681). IEEE Press.

DeGroot, M. H. (1986). *Probability and Statistics*, 2nd ed. Addison-Wesley.

Duda, R. O., & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley-Interscience.

Eldar, Y., Lindenbaum, M., Porat, M., & Zeevi, Y. Y. (1997). The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, *6:9*, 1305–1315.

Fedorov, V. V. (1972). *Theory of Optimal Experiments*. New York: Academic Press. Translation of Teoriia Optimalnogo Eksperimenta.

Freund, Y., Seung, H. S., Shamir, E., & Tishbi, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28:2/3*, 133–168.

Hasenjager, M., & Ritter, H. (1996). Active learning of the generalized high-low-game. In C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, & B. Sendhoff (Eds.), *Proceedings of the International Conference on Artificial Neural Networks*, vol. 1112 of *Lecture Notes in Computer Science* (pp. xxv+922, 501–506). Springer-Verlag.

Hasenjager, M., & Ritter, H. (1998). Active learning with local models. *Neural Processing Letters*, *7:2*, 107–117.

Lang, K. J., & Witbrock, M. J. (1988). Learning to tell two spirals apart. In D. Touretzky, G. Hinton, & T. Sejnowski (Eds.), *Proceedings of the Connectionist Models Summer School* (pp. 52–59). Morgan Kaufmann.

Lewis, D. D., & Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In W. W. Cohen, & H. Hirsh (Eds.), *Proceedings of the Eleventh International Conference on Machine Learning* (pp. 148–156). New Brunswick, NJ: Rutgers University.

Lindenbaum, M., Markovitch, S., & Rusakov, D. (1999). Selective sampling for nearest neighbor classifiers. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (pp. 366–371). Orlando, Florida: AAAI Press.

MacKay, D. J. (1998). Introduction to gaussian processes. *Neural Networks and Machine Learning. NATO ASI Series. Series F, Computer and System Sciences*, *168*, 133–166.

MacKay, D. J. C. (1992). Information-based objective functions for active data selection. *Neural Computation*, *4:4*, 590–604.

Moore, A. W., Schneider, J. G., Boyan, J. A., & Lee, M. S. (1998). Q2: Memory-based active learning for optimizing noisy continuous functions. In *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 386–394). Morgan Kaufmann.

Murthy, S., & Salzberg, S. (1995). Lookahead and pathology in decision tree induction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (pp. 1025–1031).

Papoulis, A. (1991). *Probability, Random Variables, and Stochastic Processes*, 3rd ed. McGraw-Hill.

RayChaudhuri, T., & Hamey, L. (1995). Minimisation of data collection by active learning. In *IEEE International Conference on Neural Networks Proceedings*, vol. 3 (pp. 6 vol. l+3219, 1338–1341). IEEE Press.

Russell, S. J., & Norvig, P. (1995). *Artificial Intelligence, A Modern Approach*. Prentice Hall.

Seung, H. S., Opper, M., & Sompolinsky, H. (1992). Query by committee. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory* (pp. v+452, 287–294). New York: ACM.

Smyth, B., & McKenna, E. (1999). Building compact competent case-bases. In *Lecture Notes in Artificial Intelligence*, number 1650 in Lecture Notes in Computer Science (pp. 329–342). Springer.

Tan, M., & Schlimmer, J. C. (1990). Two case studies in cost-sensitive concept acquisition. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 854–860). AAAI Press.

Turney, P. D. (1995). Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, *2*, 369–409.

Williams, C. K. I., & Barber, D. (1998). Bayesian classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20:12*, 1342.

Wilson, D. R., & Martinez, T. R. (2000). Reduction techniques for instance-based learning algorithms. *Machine Learning*, *38:3*, 257–286.

Wong, E., & Hajek, B. (1985). *Stochastic Processes in Engineering Systems*. Springer-Verlag.

Zhang, J., Yim, Y.-S., & Yang, J. (1997). Intelligent selection of instances for prediction functions in lazy learning algorithms. *Artificial Intelligence Review*, *11:1–5*, 175–191.

Received July 8, 1999
Revised April 1, 2001
Accepted September 19, 2002
Final manuscript September 19, 2002