JMLR: Workshop and Conference Proceedings 25:221–235, 2012 Asian Conference on Machine Learning

Active Learning with Hinted Support Vector Machine

Chun-Liang Li B97018@CSIE.NTU.EDU.TW
Chun-Sung Ferng R99922054@CSIE.NTU.EDU.TW
Hsuan-Tien Lin HTLIN@CSIE.NTU.EDU.TW
Department of Computer Science and Information Engineering, National Taiwan University

Editor: Steven C. H. Hoi and Wray Buntine

Abstract

The abundance of real-world data and limited labeling budget calls for active learning, which is an important learning paradigm for reducing human labeling efforts. Many recently developed active learning algorithms consider both uncertainty and representativeness when making querying decisions. However, exploiting representativeness with uncertainty concurrently usually requires tackling sophisticated and challenging learning tasks, such as clustering. In this paper, we propose a new active learning framework, called *hinted sampling*, which takes both uncertainty and representativeness into account in a simpler way. We design a novel active learning algorithm within the hinted sampling framework with an extended support vector machine. Experimental results validate that the novel active learning algorithm can result in a better and more stable performance than that achieved by state-of-the-art algorithms.

Keywords: Active Learning, Support Vector Machine

1. Introduction

Labeled data are the basic ingredients in training a good model in machine learning. It is common in real-world applications that one needs to cope with a large amount of data, and labeling such data can be costly. For example, in the medical domain, a doctor may be required to distinguish (label) cancer patients from non-cancer patients according to their clinical records (data). In such applications, an important issue is to achieve high accuracy within a limited budget. This issue demands active learning (Settles, 2009), which is a machine learning setup that allows iteratively querying the labeling oracle (doctor) in a strategic manner to label some selected instances (clinic records). By using a suitable query strategy, an active learning approach can achieve high accuracy while performing only a few iterations i.e., only a few calls to the (expensive) querying oracle (Settles, 2009).

One intuitive approaches in active learning is called uncertainty sampling (Lewis and Gale, 1994). This approach maintains a classifier on hand, and queries the most uncertain instances, whose uncertainty is measured by the closeness to the decision boundary of the classifier, to fine-tune the boundary. However, the performance of uncertainty sampling becomes restricted owing to the limited view of the classifier. In other words, uncertainty sampling can be hair-splitting on the local instances that confuse the classifier, but not considering the global distribution of instances. Therefore, queries may not represent the underlying data distribution well, leading to unsatisfactory performance of uncertainty sampling (Settles, 2009).

© 2012 C.-L. Li, C.-S. Ferng & H.-T. Lin.

As suggested in Cohn et al. (1996); Xu et al. (2003), active learning can be improved by considering the unlabeled instances in order to query the instance that is not only uncertain to the available classifier but also "representative" to the global data distribution. There are many existing algorithms that use unlabeled information to improve the performance of active learning, such as representative sampling (Xu et al., 2003).

Representative sampling makes querying decisions use not only the uncertainty of each instance, but also the representativeness, which is measured by determining whether the instances reside in a dense area. Typical representative sampling algorithms (Xu et al., 2003; Nguyen and Smeulders, 2004; Dasgupta and Hsu, 2008) estimate the underlying data distribution via clustering methods. However, the performance of the algorithms depends on the result of clustering, which is a sophisticated and non-trivial task, especially when the instances are within a high dimensional space. Another state-of-the-art algorithm (Huang et al., 2010) models the representativeness by estimating the potential label assignment of the unlabeled instances on the basis of the min-max view of active learning (Hoi et al., 2008). The performance of this algorithm depends on the results of estimating the label assignments, which is also a complicated task.

In this work, we propose a novel framework of active learning, hinted sampling, which considers the unlabeled instances as hints (Abu-Mostafa, 1995) of the global data distribution, instead of directly clustering them or estimating their label assignments. This leads to a simpler active learning algorithm. Similar to representative sampling, hinted sampling also considers both uncertainty and representativeness. Hinted sampling enjoys the advantage of simplicity by avoiding the clustering or label-assignment estimation steps. We demonstrate the effectiveness of hinted sampling by designing a novel algorithm with support vector machine (SVM; Vapnik, 1998). In the algorithm, we extend the usual SVM to a novel formulation, HintSVM, which is easier to solve than either clustering or label-assignment estimation. We then study the hint selection strategy to improve the efficiency and effectiveness of the proposed algorithm. Experimental results demonstrate that the simple HintSVM with a proper hint selection strategy is comparable to the best of both uncertainty sampling and representative sampling algorithms, and results in better and more stable performance than other state-of-the-art active learning algorithms.

The rest of the paper is organized as follows. Section 2 introduces the formal problem definition and reviews the related works. Section 3 describes our proposed hinted sampling framework as well as the HintSVM algorithms. Section 4 elucidates the hint selection strategy. Section 5 reports experiment results and comparisons. Finally, Section 6 concludes this work.

2. Problem Definition and Related Works

In this work, we focus on pool-based active learning for binary classification, which is one of the most common setup in active learning (Lewis and Gale, 1994). At the initial stage of the setup, the learning algorithm is presented with a labeled data pool and an unlabeled data pool. We denote the labeled data pool by $\mathcal{D}_l = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$ and the unlabeled data pool by $\mathcal{D}_u = \{\widetilde{\mathbf{x}}_1, \widetilde{\mathbf{x}}_2, ..., \widetilde{\mathbf{x}}_M\}$, where the input vectors $\mathbf{x}_i, \widetilde{\mathbf{x}}_j \in \mathbf{R}^d$ and the labels $y_i \in \{-1, 1\}$. Usually, the labeled data pool $\mathcal{D}_l$ is relatively small or even empty, whereas the unlabeled data pool $\mathcal{D}_u$ is assumed to be large. Active learning is an

iterative process that contains R iterations of querying and learning. That is, an active learning algorithm can be split into two parts: the querying algorithm $\mathcal{Q}$ and the learning algorithm $\mathcal{L}$.

Using the initial $\mathcal{D}_l \cup \mathcal{D}_u$, the learning algorithm $\mathcal{L}$ is first called to learn a decision function $f^{(0)}\colon \mathbf{R}^d \to \mathbf{R}$, where the function $\text{sign}(f^{(0)}(\mathbf{x}))$ is taken for predicting the label of any input vector $\mathbf{x}$. Then, in each r-th iteration, where $r = 1, 2, ..., R$, the querying algorithm $\mathcal{Q}$ is allowed to select an instance $\widetilde{\mathbf{x}}_s \in \mathcal{D}_u$ and query its label y_s from a labeling oracle. After querying, $(\widetilde{\mathbf{x}}_s, y_s)$ is added to the labeled pool $\mathcal{D}_l$ and $\widetilde{\mathbf{x}}_s$ is removed from the unlabeled pool $\mathcal{D}_u$. The learning algorithm $\mathcal{L}$ then learns a decision function $f^{(r)}$ from the updated $\mathcal{D}_l \cup \mathcal{D}_u$. The goal of active learning is to use the limited querying and learning opportunities properly to obtain a decent list of decision functions $[f^{(1)}, f^{(2)}, ..., f^{(R)}]$ that can achieve low out-of-sample (test) error rates.

As discussed in a detailed survey (Settles, 2009), many active learning algorithms for binary classification exist. In this paper, we shall review some well-recognized and relevant ones. One of the most intuitive families of algorithms is called uncertainty sampling (Lewis and Gale, 1994). As the name suggests, the querying algorithm $\mathcal{Q}$ of uncertainty sampling queries the most uncertain $\widetilde{\mathbf{x}}_s \in \mathcal{D}_u$, where the uncertainty for each input vector $\widetilde{\mathbf{x}}_j \in \mathcal{D}_u$ is usually computed by re-using the decision function $f^{(r-1)}$ returned from the learning algorithm $\mathcal{L}$. For instance, Tong and Koller (2000) take the support vector machine (SVM; Vapnik, 1998) as $\mathcal{L}$ and measure the uncertainty of $\widetilde{\mathbf{x}}_j$ by the distance between $\widetilde{\mathbf{x}}_j$ and the boundary $f^{(r-1)} = 0$. In other words, the algorithm in Tong and Koller (2000) queries the $\widetilde{\mathbf{x}}_s$ that is closest to the boundary.

Uncertainty sampling can be viewed as a greedy approach that queries instances from the viewpoint only of the decision function $f^{(r-1)}$. When the decision function is not close enough to the ideal one, however, this limited viewpoint can hinder the performance of the active learning algorithm. Thus, Cohn et al. (1996) suggest that the viewpoint of the unlabeled pool $\mathcal{D}_u$ should also be included. Their idea leads to another family of active learning algorithms, called representative sampling (Xu et al., 2003), or density-weighted sampling (Settles, 2009). Representative sampling takes both the uncertainty and the representativeness of each $\widetilde{\mathbf{x}}_j \in \mathcal{D}_u$ into account concurrently in the querying algorithm $\mathcal{Q}$, where the representativeness of $\widetilde{\mathbf{x}}_j$ with respect to $\mathcal{D}_u$ is measured by the density of its neighborhood area. For instance, Xu et al. (2003) employ the SVM as the learning algorithm $\mathcal{L}$ as do Tong and Koller (2000). They use a querying algorithm $\mathcal{Q}$ that first clusters the unlabeled instances near the boundary of $f^{(r-1)}$ by a K-means algorithm, and then queries one of the centers of those clusters. In other words, the queried instance is not only uncertain for $f^{(r-1)}$ but also representative for $\mathcal{D}_u$. Some other works estimate the representativeness with a generative model. For instance, Nguyen and Smeulders (2004) propose a querying algorithm $\mathcal{Q}$ that uses multiple Gaussian distributions to cluster all input vector $\mathbf{x}_i \in \mathcal{D}_l$, $\widetilde{\mathbf{x}}_j \in \mathcal{D}_u$ and estimate the prior probability $p(\mathbf{x})$; $\mathcal{Q}$ then makes querying decisions using the product of the prior probability and some uncertainty measurement. The idea of estimating the representativeness via clustering is a core element of many representative sampling algorithms (Xu et al., 2003; Nguyen and Smeulders, 2004; Dasgupta and Hsu, 2008). Nevertheless, clustering is a challenging task and it is not always easy to achieve satisfactory clustering performance. When the clustering performance is not satisfactory, it has been observed (Donmez et al., 2007; Huang et al., 2010) that repre-

sentative sampling algorithms also fail to achieve decent performance. In other words, the clustering step is usually the bottleneck of representative sampling.

Huang et al. (2010) propose an improved algorithm that models representativeness without clustering. In the algorithm, the usefulness of each $\widetilde{\mathbf{x}}_j$, which implicitly contains both uncertainty and representativeness, is estimated by using a technique in semi-supervised learning (Hoi et al., 2008) that checks approximately all possible label assignments for each unlabeled $\widetilde{\mathbf{x}}_j \in \mathcal{D}_u$. The querying algorithm $\mathcal{Q}$ proposed (Huang et al., 2010) is based on the usefulness of each $\widetilde{\mathbf{x}}_j$; the learning algorithm $\mathcal{L}$ is simply a stand-alone SVM. While the active learning algorithm (Huang et al., 2010) often achieves promising empirical results, its bottleneck is the label-estimation step, which is rather sophisticated and thus not always easy to achieve a satisfactory performance.

Another improvement of representative sampling is presented by Donmez et al. (2007), who report that representative sampling is less efficient than uncertainty sampling for later iterations, in which the decision function is closer to the ideal one. To combine the best properties of uncertainty sampling and representative sampling, Donmez et al. (2007) propose a mixed algorithm by extending representative sampling (Nguyen and Smeulders, 2004). The proposed query algorithm $\mathcal{Q}$ (Donmez et al., 2007) is split into two stages. The first stage performs representative sampling (Nguyen and Smeulders, 2004) while estimating the expected error reduction. When the expected reduction is less than a given threshold, the querying algorithm $\mathcal{Q}$ switches to uncertainty sampling for fine-tuning the decision boundary. The bottleneck of the algorithm (Donmez et al., 2007) is still the clustering step in the first stage.

Instead of facing the challenges of either clustering or label-estimation, we propose to view the information in $\mathcal{D}_u$ differently. In particular, the unlabeled instances $\widetilde{\mathbf{x}}_j \in \mathcal{D}_u$ are taken as hints (Abu-Mostafa, 1995) that guide the querying algorithm $\mathcal{Q}$. The idea of using hints leads to a simpler active learning algorithm with better empirical performance, as introduced in the next sections.

3. Hinted Sampling Framework

First, we illustrate the potential drawback of uncertainty sampling with a linear SVM classifier (Vapnik, 1998), which is applied to a two-dimensional artificial dataset. Figure 1 shows the artificial dataset, which consists of three clusters, each of which contains instances of a particular class. We denote one class by a red cross and the other by a filled green circle. The labeled instances in $\mathcal{D}_l$ are marked with a blue square while other instances are in $\mathcal{D}_u$. In Figure 1(a), the initial two labeled instances reside in two of the clusters with different labels. The initial decision function $f^{(0)}$ trained on the labeled instances (from the two clusters) is not aware of the third cluster. The decision function $f^{(0)}$ then mis-classifies the instances in the third cluster, and causes the querying algorithm $\mathcal{Q}$ (which is based on $f^{(0)}$) to query only from the instances near the "wrong" boundary rather than exploring the third cluster. After several iterations, as shown in Figure 1(b), the uncertainty sampling algorithm still outputs an unsatisfactory decision function that mis-classifies the entire unqueried (third) cluster.

The unsatisfactory performance of uncertainty sampling originates in its lack of awareness of candidate unlabeled instances that should be queried. When trained on only a few

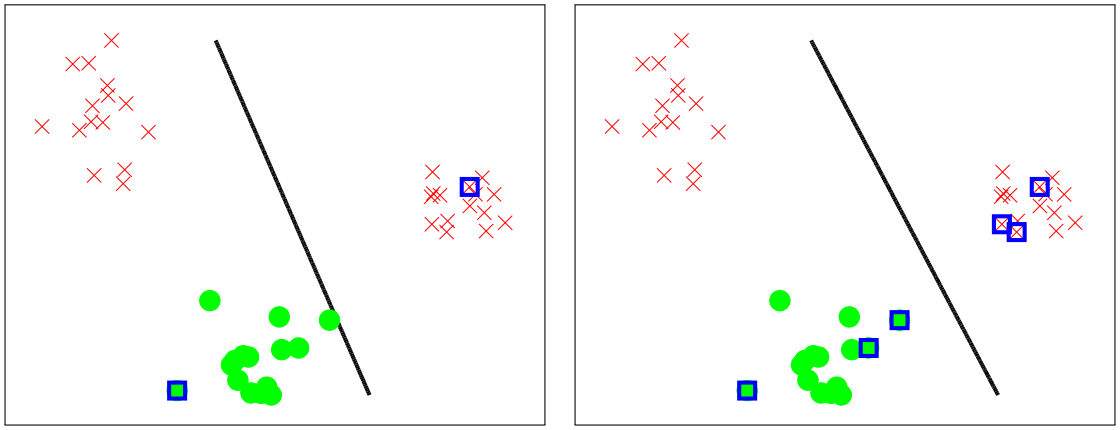


Figure 1: (a) The decision function (black) obtained from two labeled (blue) instances; (b) when using the decision function in (a) for uncertainty sampling, the upper-left cluster keeps being ignored

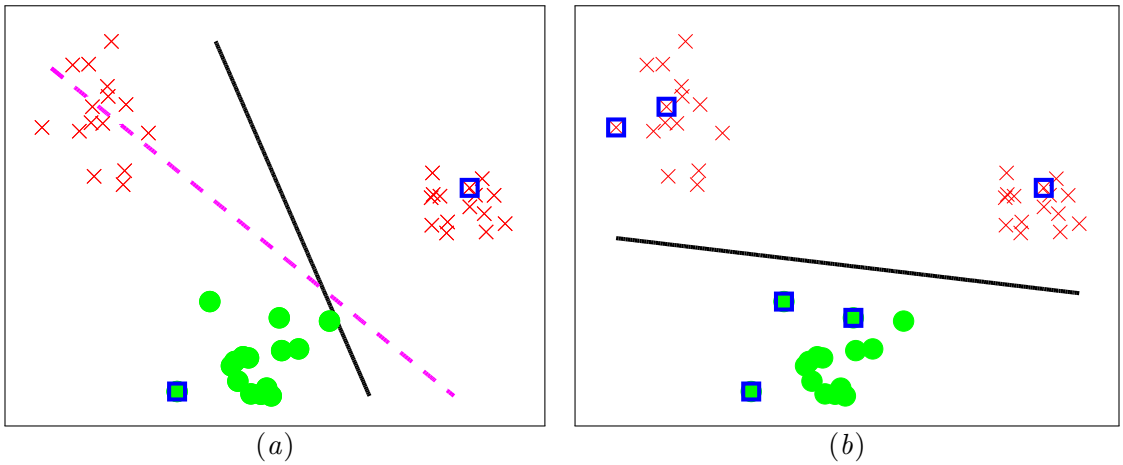


Figure 2: (a) The hinted query function (dashed magenta line) that is aware of the upper-left cluster; (b) when using the hinted decision function in (a) for uncertainty sampling, all three clusters are explored

labeled instances, the resulting (linear) decision function is overly confident about the unlabeled instances that are far from the boundary. Intuitively, uncertainty sampling could be improved if the querying algorithm $\mathcal{Q}$ were aware of and less confident about the unqueried regions. Both clustering (Nguyen and Smeulders, 2004) and label-estimation (Huang et al., 2010) are based on this intuition, but they explore the unlabeled regions in a rather sophisticated way.

We propose a simpler alternative as follows. Note that the uncertainty sampling algorithm measures the uncertainty by the distance between instances and the boundary. In order to make $\mathcal{Q}$ less confident about the unlabeled instances, we seek a "query boundary" that not only classifies the labeled instances correctly but also passes through the unqueried regions, denoted by the dashed magenta line in Figure 2(*a*). Then, in the later iterations, the query algorithm $\mathcal{Q}$, using the query boundary, would be less confident about the unqueried regions, and thus be able to explore them. The instances in the unqueried regions give hints as to where the query boundary should pass. Using these hints about the unqueried regions, the uncertainty sampling algorithm can take both uncertainty and the underlying distribution into account concurrently, and achieve better performance, as shown in Figure 2(*b*).

Based on this idea, we propose a novel active learning framework, *hinted sampling*. The learning algorithm $\mathcal{L}$ in hinted sampling is similar to that in uncertainty sampling, but the querying algorithm is different. In particular, the querying algorithm $\mathcal{Q}$ is provided with some unlabeled instances, called the hint pool $\mathcal{D}_h \subseteq \mathcal{D}_u$. When suitably using the information in the hint pool $\mathcal{D}_h$, both uncertainty and representativeness can be considered concurrently to obtain a query boundary that assists $\mathcal{Q}$ in making query decisions. Next, we design a concrete active learning algorithm based on SVM, which is also used as the core of many state-of-the-art algorithms (Tong and Koller, 2000; Xu et al., 2003; Huang et al., 2010), as both $\mathcal{L}$ and $\mathcal{Q}$. Before illustrating the complete algorithm, we show how SVM can be appropriately extended to use the information in $\mathcal{D}_h$ for $\mathcal{Q}$.

3.1. HintSVM

The extended SVM is called HintSVM, which takes hints into account. The goal of HintSVM is to locate a query boundary which does well on two objectives: (1) classifying labeled instances in $\mathcal{D}_l$, and (2) being close to the unlabeled instances in hint pool $\mathcal{D}_h$. Note that the two objectives are different from the usual semi-supervised SVM (Bennett and Demiriz, 1998), which pushes the unlabeled instances away from the decision boundary.

The first objective matches an ordinary support vector classification (SVC) problem. To deal with the second objective, we consider ϵ-support vector regression (ϵ-SVR) and set regression targets to 0 for all instances in $\mathcal{D}_h$, which means that instances in $\mathcal{D}_h$ should be close to the query boundary. By combining the objective functions of SVC and ϵ-SVR together, HintSVM solves the following convex optimization problem, which simultaneously achieves the two objectives.

$$
\begin{aligned}
\min_{\mathbf{w},b,\xi,\widetilde{\xi},\widetilde{\xi}^*} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C_l\sum_{i=1}^{|\mathcal{D}_l|}\xi_i + C_h\sum_{j=1}^{|\mathcal{D}_h|}\left(\widetilde{\xi}_j + \widetilde{\xi}_j^*\right) \\
\text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{for } (\mathbf{x}_i, y_i) \in \mathcal{D}_l, \\
& \mathbf{w}^T\widetilde{\mathbf{x}}_j + b \leq \epsilon + \widetilde{\xi}_j \quad \text{for } \mathbf{x}_j \in \mathcal{D}_h, \\
& -(\mathbf{w}^T\widetilde{\mathbf{x}}_j + b) \leq \epsilon + \widetilde{\xi}_j^* \quad \text{for } \mathbf{x}_j \in \mathcal{D}_h.
\end{aligned} \tag{1}
$$

Here ϵ is the margin of tolerance for being close to the boundary, and C_l, C_h are the weights of the classification errors (on $\mathcal{D}_l$) and hint errors (on $\mathcal{D}_h$), respectively. Similar to the usual SVC and ϵ-SVR, the convex optimization problem can be transformed to the dual form to

allow using the kernel trick. Define $\hat{\mathbf{x}}_i = \mathbf{x}_i$, $\hat{\mathbf{x}}_{|\mathcal{D}_l|+j} = \hat{\mathbf{x}}_{|\mathcal{D}_l|+|\mathcal{D}_h|+j} = \widetilde{\mathbf{x}}_j$, $\hat{y}_i = y_i$, $\hat{y}_{|\mathcal{D}_l|+j} = 1$, and $\hat{y}_{|\mathcal{D}_l|+|\mathcal{D}_h|+j} = -1$ for $1 \le i \le |\mathcal{D}_l|$ and $1 \le j \le |\mathcal{D}_h|$. The dual problem of (1) can be written as follows:

$$\begin{aligned} \min_{\alpha} \quad & \tfrac{1}{2}\alpha^T Q\alpha + \mathbf{p}^T\alpha \\ \text{subject to} \quad & \hat{\mathbf{y}}^T\alpha = 0, \\ & 0 \le \alpha_i \le C_l \quad \text{for } i = 1, 2, \cdots, |\mathcal{D}_l|, \\ & 0 \le \alpha_j \le C_h \quad \text{for } j = |\mathcal{D}_l|+1, \cdots, |\mathcal{D}_l|+2|\mathcal{D}_h|, \end{aligned}$$

where $p_i = -1$, $p_j = \epsilon$, and $Q_{ab} = \hat{y}_a\hat{y}_b\hat{\mathbf{x}}_a^T\hat{\mathbf{x}}_b$. The derived dual form can be easily solved by any state-of-the-art quadratic programming solver, such as the one implemented in LIBSVM (Chang and Lin, 2011).

3.2. Hinted Sampling with HintSVM

Next, we incorporate the proposed hinted sampling with the derived HintSVM formulation to make a novel active learning algorithm, *Active Learning with HintSVM* (ALHS).

The querying algorithm $\mathcal{Q}$ of ALHS selects unlabeled instances from the unlabeled pool $\mathcal{D}_u$ as the hint pool $\mathcal{D}_h$ and trains HintSVM from $\mathcal{D}_l$ and $\mathcal{D}_h$ to obtain the query boundary for uncertainty sampling. The use of both $\mathcal{D}_l$ and $\mathcal{D}_h$ combines uncertainty and representativeness. The learning algorithm $\mathcal{L}$ of ALHS, on the other hand, trains a stand-alone SVM from $\mathcal{D}_l$ to get a decision function $f^{(r)}$, just like $\mathcal{L}$ in uncertainty sampling (Tong and Koller, 2000). The full ALHS algorithm is listed in Algorithm 1.

Algorithm 1 The ALHS algorithm

input: the number of rounds R; a labeled pool $\mathcal{D}_l$; an unlabeled pool $\mathcal{D}_h$; parameters of HintSVM and SVM

for $r \leftarrow 1$ **to** R **do**
  Select $\mathcal{D}_h$ from $\mathcal{D}_u$
  $h \leftarrow Train\,\text{HintSVM}(C_l, C_h, \epsilon, \mathcal{D}_h \cup \mathcal{D}_l)$
  $(\widetilde{\mathbf{x}}_s, y_s) \leftarrow Query(h, \mathcal{D}_u)$
  $\mathcal{D}_u \leftarrow \mathcal{D}_u \setminus \widetilde{\mathbf{x}}_s$; $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup (\widetilde{\mathbf{x}}_s, y_s)$
  $f^{(r)} \leftarrow Train\,\text{SVM}(C, \mathcal{D}_l)$
end

Uncertainty sampling with SVM is a special case of ALHS when $\mathcal{D}_h$ is empty. In other words, ALHS can be viewed as a generalization of uncertainty sampling that considers representativeness through the hints. The simple use of hints avoids the challenges in clustering or label-estimation steps. With a proper selection of hints, ALHS can be aware of the key unqueried regions, therefore improving the performance. Next, we design and illustrate one promising selection strategy.

4. Hint Selection Strategy

A naïve strategy for selecting a proper hint pool $\mathcal{D}_h \subseteq \mathcal{D}_u$ is to directly let $\mathcal{D}_h = \mathcal{D}_u$, which retains all the information about the unlabeled data. However, given that the size of $\mathcal{D}_u$

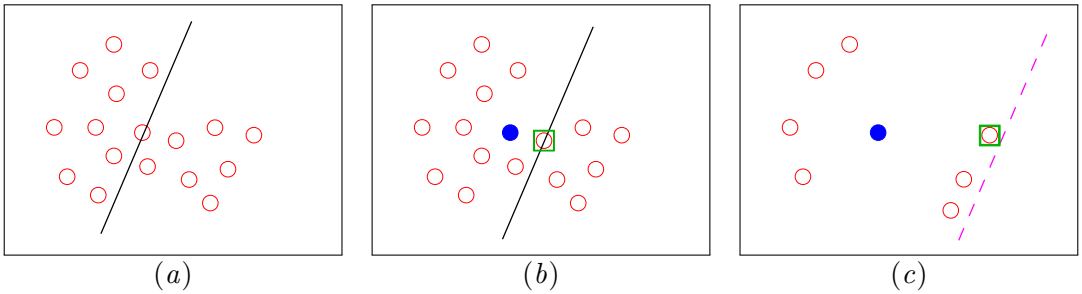


Figure 3: (a) The original decision boundary; (b) HintSVM boundary after querying; (c) HintSVM boundary (dashed magenta) after querying and dropping

is usually much larger than the size of $\mathcal{D}_l$, this strategy may cause the hints to overwhelm HintSVM, which leads to performance and computational concerns. Another naïve strategy is to uniformly select $\mathcal{D}_h$ from $\mathcal{D}_l$ by random sampling. However, the uniform-random strategy is not aware of the labeled instances and can hence result in less effective hints. In this section, we propose a strategy that resolves the problems in the naïve strategies above, and echoes some ideas from the design of modern active learning algorithms. Our strategy consists of three steps: hint influence balancing, hint sampling and hint termination.

4.1. Hint Influence Balancing

When the hint pool $\mathcal{D}_h$ is relatively larger than the labeled pool $\mathcal{D}_l$, one potential issue is that the objective function of HintSVM is dominated by the hint errors $\widetilde{\xi}_j + \widetilde{\xi}_j^*$ introduced by $\mathcal{D}_h$. In order to guide HintSVM to balance both the classification performance and the hint errors, we simply set $C_h = C$, $C_l = \max\left(\frac{|\mathcal{D}_h|}{|\mathcal{D}_l|}, 1\right) \times C$ to equalize the contribution of $\mathcal{D}_l$ and $\mathcal{D}_h$, where C is used in the learning algorithm $\mathcal{L}$ for ALHS to learn the SVM decision function $f^{(r)}$.

4.2. Hint Sampling

In ALHS, the hints are used for guiding the uncertainty-based querying algorithm to consider the less-queried regions. Thus, there is no need to allocate hints within regions that already contain many labeled examples. In fact, hints in those regions may even mislead the uncertainty-based querying algorithm. A strategic hint selection strategy that is aware of $\mathcal{D}_l$ can therefore improve not only the computational efficiency but also the learning performance of ALHS.

Instead of re-selecting a new $\mathcal{D}_h$ from $\mathcal{D}_u$ in each iteration, we propose a simple alternative: retaining a huge $\mathcal{D}_h = \mathcal{D}_u$ at the initial stage, and dropping some less-useful hints in each iteration. To design the hint sampling (dropping) strategy, we first review the characteristics of HintSVM. In Figure 3(a), the red empty circles represent the hints and the query boundary returned from HintSVM passes through their center. In Figure 3(b), after querying the filled blue circle $\mathbf{x}_i$, which is closest to the query boundary in Figure 3(a), the query boundary is pushed away from $\mathbf{x}_i$ because of the classification objective of HintSVM,

but still passes through the same region because of the many hints. The instance to be queried is then the green square, which is close to $\mathbf{x}_i$ and arguably not carrying much additional information. To drive the query boundary away from the known $\mathbf{x}_i$, the surrounding neighbors of $\mathbf{x}_i$ should be dropped from the hint pool $\mathcal{D}_h$, as shown in Figure 3(*c*). Then, the boundary could assist the querying algorithm $\mathcal{Q}$ in querying other potentially more valuable instances that are far from $\mathbf{x}_i$, such as the one marked by a square in Figure 3(*c*).

We implement the idea with a neighborhood function $\phi_i \colon \mathbf{R}^d \to [0,1]$ to measure the closeness of an unlabeled instance $\widetilde{\mathbf{x}}_j$ to each given labeled instance $\mathbf{x}_i$. Given the labeled pool $\mathcal{D}_l$ and the neighborhood functions ϕ_i of each $\mathbf{x}_i \in \mathcal{D}_l$, we propose dropping $\widetilde{\mathbf{x}}_j$ from the hint pool with the probability $\max_{\mathbf{x}_i \in \mathcal{D}_l} \phi_i(\widetilde{\mathbf{x}}_j)$. That is, if $\widetilde{\mathbf{x}}_j$ is close to some $\mathbf{x}_i$ (high $\phi_i(\widetilde{\mathbf{x}}_j)$), with a high probability that $\widetilde{\mathbf{x}}_j$ would be dropped from $\mathcal{D}_h$. The neighborhood function ϕ_i can be viewed as a "dropping recommendation" from $\mathbf{x}_i$. We design ϕ_i by requesting the function to satisfy three natural constraints: (1) $\phi_i(\mathbf{x}_i) = 1$, which means a duplicate example should be dropped (2) the ϕ_i for the closest neighbor to $\mathbf{x}_i$ is P (3) the ϕ_i for the farthest neighbor to $\mathbf{x}_i$ is p, where $p \leq P$.

We model ϕ_i by a radius basis function to satisfy the three constraints:

$$\phi_i(\widetilde{\mathbf{x}}_j) = P^{\left(r_j^{\alpha_i}\right)}. \tag{2}$$

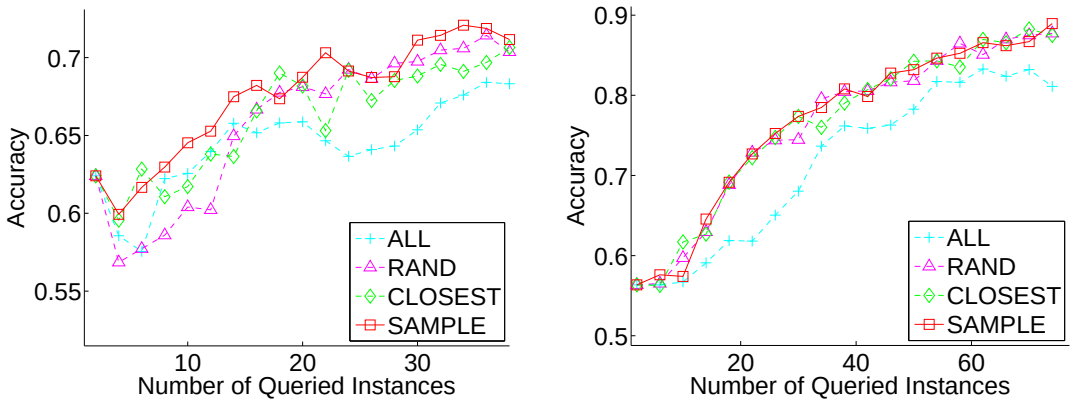
Here $r_j = \|\widetilde{\mathbf{x}}_j - \mathbf{x}_i\|/d_i$ is a normalized distance of $\widetilde{\mathbf{x}}_j$ given $\mathbf{x}_i$, and d_i is the distance to the closest neighbor of $\mathbf{x}_i$. Then, according to (2) and the constraints, we can easily solve $\alpha_i = \log\left(\frac{\log p}{\log P}\right) \Big/ \log R_i$, where R_i is the normalized distance of the farthest neighbor of $\mathbf{x}_i$.

We now briefly compare four sampling strategies for ALHS: (1) ALL: include all unlabeled instances, (2) RAND: randomly drop instances from $\mathcal{D}_h$ with a fixed probability, (3) CLOSEST: drop a fixed number of neighbors closest to the queried instance, (4) SAMPLE: the proposed strategy. The results on two datasets are shown in Figure 4, and the detailed experimental settings are listed in Section 5. According to the experimental results, the ALL strategy is the worst because too many hints overwhelm HintSVM. RAND strategy can solve the weakness of ALL strategy, but its performance at the earlier stage may be unsatisfactory if current labeled instances are not considered. CLOSEST matches the characteristic of HintSVM, but it may be an overkill to drop all neighbors based on only one queried instance. Among the strategies, SAMPLE performs the best. It drops the neighbor instances with the probabilities computed from neighborhood functions, and has a chance to keep some neighbors as hints in dense regions.

Furthermore, based on the hint selection strategy, the hint pool $\mathcal{D}_h$ contains the most informative instances in $\mathcal{D}_u$. Therefore, when $\mathcal{D}_h$ is non-empty, we propose to let $\mathcal{Q}$ select queries from $\mathcal{D}_h$ instead of $\mathcal{D}_u$.

4.3. Hint Termination

After querying a sufficient number of instances, ALHS captures the underlying data distribution with a high probability and the classifier $f^{(r)}$ on hand shall be close to the ideal one. At that time, all hints carry little information to assist ALHS and thus are not important. The querying algorithm $\mathcal{Q}$ in ALHS can then drop all the hints to switch to uncertainty

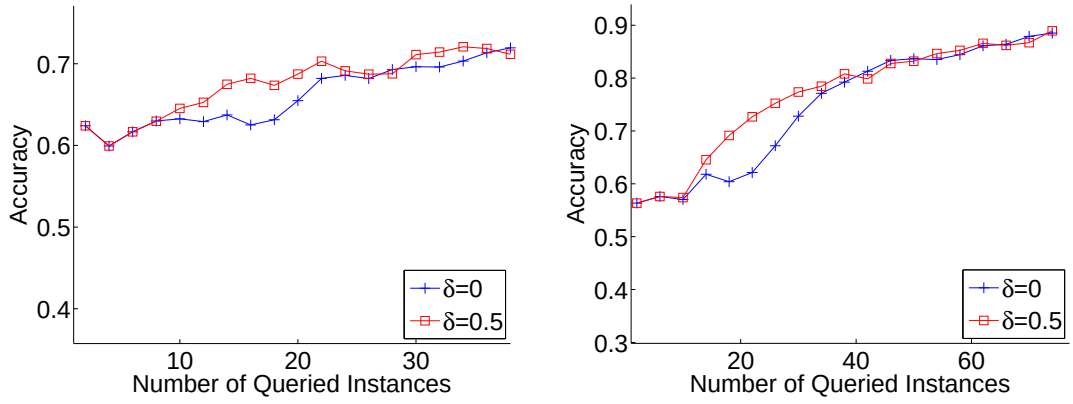


(a) *diabetes* (b) *letterVvsY*

Figure 4: Comparison of hint sampling methods for different datasets

sampling. This idea is similarly explored by Donmez et al. (2007), and we call it hint termination.

We set a termination rule based on the proportion of the remaining hint instances. After we drop many hints by querying enough instances, the remaining hints are not important. The termination rule is $\frac{|\mathcal{D}_h|}{|\mathcal{D}_l|+|\mathcal{D}_u|} \leq \delta$, where δ is a given threshold. We examine two thresholds at $\delta = 0$ (no termination) and $\delta = 0.5$. As shown in Figure 5, the experiment results show that $\delta = 0.5$ is comparable to $\delta = 0$ and could even outperform $\delta = 0$ in some cases. We observe the similar results in other datasets, and thus consider $\delta = 0.5$ in future experiments.



(a) *diabetes* (b) *letterVvsY*

Figure 5: Comparison of different δ values

Table 1: Comparison on accuracy (mean ± se) after querying 5% of unlabeled pool

	Algorithms (%), the highest accuracy for each dataset is in boldface				
data	UNCERTAIN	REPRESENT	QUIRE	DUAL	ALHS
australian	82.188 ± 1.571	83.739 ± 0.548	82.319 ± 1.126	81.304 ± 0.647	$\mathbf{84.072 \pm 0.454}$
breast	96.334 ± 0.278	95.264 ± 0.439	$\mathbf{96.657 \pm 0.187}$	96.408 ± 0.196	96.525 ± 0.219
diabetes	63.229 ± 2.767	66.758 ± 0.505	66.771 ± 0.960	65.143 ± 0.381	$\mathbf{66.862 \pm 1.632}$
german	69.060 ± 0.497	67.240 ± 1.099	68.750 ± 0.605	69.620 ± 0.323	$\mathbf{69.750 \pm 0.349}$
letterMvsN	89.632 ± 1.103	83.463 ± 1.348	81.372 ± 1.693	83.437 ± 1.211	$\mathbf{91.919 \pm 0.812}$
letterVvsY	79.245 ± 1.176	63.523 ± 2.335	68.516 ± 2.132	76.213 ± 1.549	$\mathbf{79.381 \pm 1.174}$
segment	95.437 ± 0.367	94.390 ± 0.482	96.074 ± 0.224	86.078 ± 2.834	$\mathbf{96.095 \pm 0.204}$
splice	74.430 ± 0.606	69.117 ± 1.452	70.340 ± 0.942	56.969 ± 0.576	$\mathbf{75.506 \pm 0.403}$
wdbc	93.842 ± 3.137	95.616 ± 0.711	96.613 ± 0.230	96.056 ± 0.250	$\mathbf{96.921 \pm 0.200}$

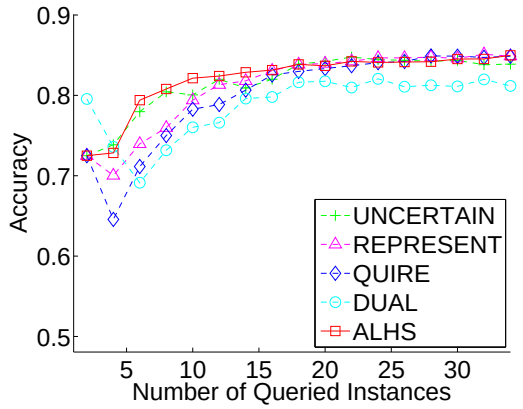
5. Experiment

We compared the proposed ALHS algorithm with the following active learning algorithms: (1) UNCERTAIN (Tong and Koller, 2000): uncertainty sampling with SVM, (2) REPRESENT (Xu et al., 2003): representative sampling with SVM and clustering, (3) DUAL (Donmez et al., 2007): mixture of uncertainty and representative sampling, (4) QUIRE (Huang et al., 2010): representative sampling with label estimation based on the min-max view.

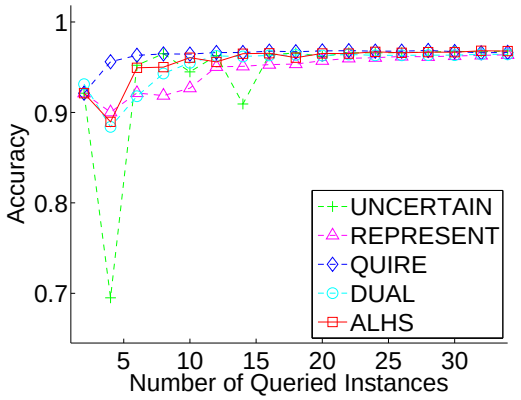
We conducted experiments on nine UCI benchmarks (Frank and Asuncion, 2010), which are *australian*, *breast*, *diabetes*, *german*, *splice*, *wdbc*, *letetrMvsN*, *letterVvsY* (Donmez et al., 2007; Huang et al., 2010) and *segment-binary* (Ratsch et al., 2001; Donmez et al., 2007) as chosen by other related works. For each dataset, we randomly divided it into two parts with equal size. One part was treated as the unlabeled pool $\mathcal{D}_u$ for active learning algorithms. The other part was reserved as the test set. Before querying, we randomly labeled one positive instance and one negative instance to form the labeled pool $\mathcal{D}_l$. For each dataset, we ran the algorithms 20 times with different random splits.

Due to the difficulty of locating the best parameters for each active learning algorithms in practice, we chose to compare all algorithms on fixed parameters. In the experiments, every SVM-based algorithm took LIBSVM (Chang and Lin, 2011) with the RBF kernel and the default parameters, except for $C = 5$. Correspondingly, the parameter λ in Donmez et al. (2007); Huang et al. (2010) was set to $\lambda = \frac{1}{C}$. These parameters ensure that all four algorithms behave in a stable manner. For ALHS, we fixed $\delta = 0.5$, $P = 0.5$ and $p = 0.01$ as discussed in the previous sections with no further tuning for each dataset. For other algorithms, we take the parameters in the original papers.

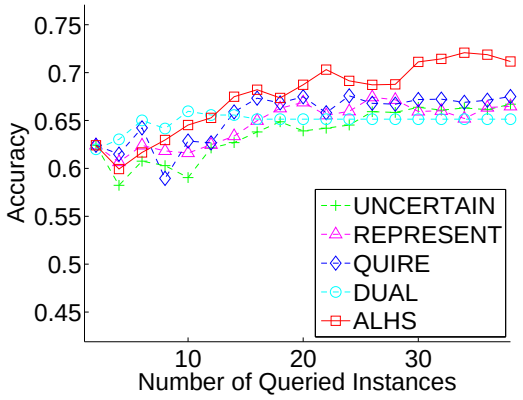
Figure 6 presents the accuracy of different active learning algorithms along with the number of rounds R, which equals the number of queried instances. Tables 1 and 2 list the mean and standard error of accuracy when $R = |\mathcal{D}_u| \times 5\%$ and $R = |\mathcal{D}_u| \times 10\%$, respectively. The highest mean accuracy is shown in boldface for each dataset. We also conducted the t-test at 95% significance level as described by Melville and Mooney (2004); Guo and Greiner (2007); Donmez et al. (2007). The t-test results are given in Table 3, which summarizes the number of datasets in which ALHS performs significantly better (or worse) than the other algorithms.



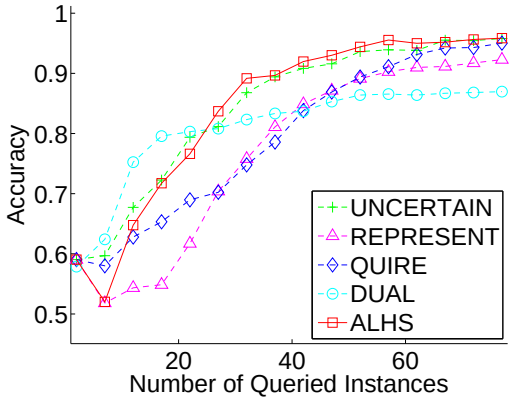
(a) *australian*



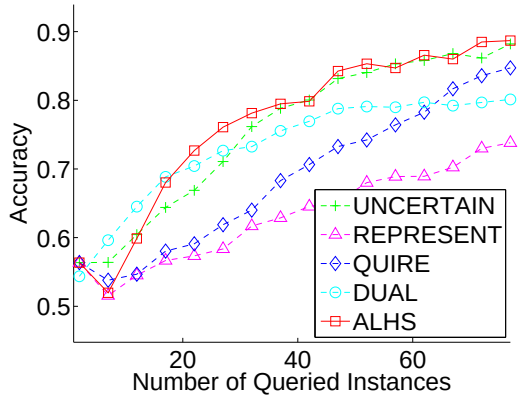
(b) *breast*



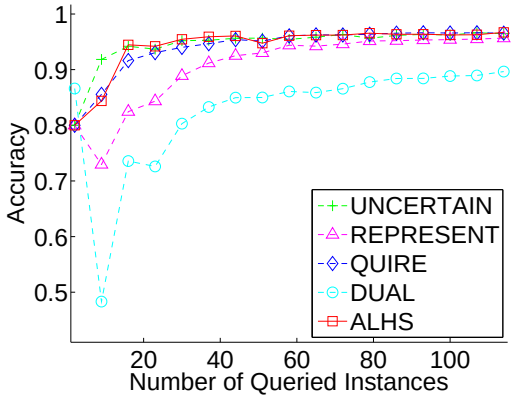
(c) *diabetes*



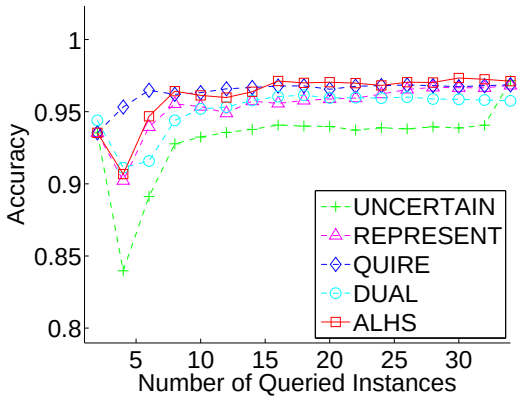
(d) *leterMvsN*



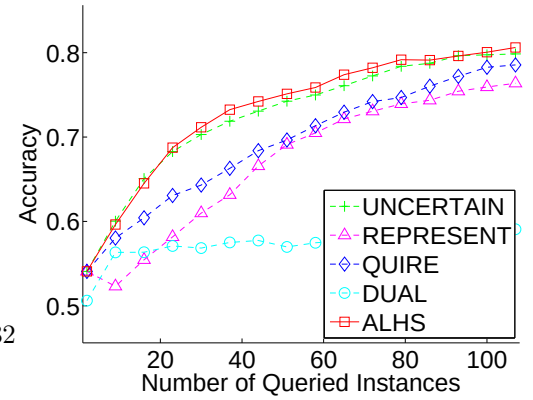
(e) *letterVvsY*



(f) *segment*



(g) *wdbc*



(h) *splice*

Table 2: Comparison on accuracy (mean ± se) after querying 10% of unlabeled pool

	Algorithms (%), the highest accuracy for each dataset is in boldface				
data	UNCERTAIN	REPRESENT	QUIRE	DUAL	ALHS
australian	83.884 ± 0.460	84.884 ± 0.367	84.870 ± 0.455	81.174 ± 0.798	$\mathbf{84.986 \pm 0.314}$
breast	$\mathbf{96.804 \pm 0.188}$	96.378 ± 0.212	96.642 ± 0.179	96.422 ± 0.235	96.789 ± 0.175
diabetes	66.706 ± 2.632	66.484 ± 1.223	67.500 ± 1.337	65.143 ± 0.381	$\mathbf{71.159 \pm 1.224}$
german	71.410 ± 0.488	67.150 ± 0.773	70.250 ± 0.560	69.760 ± 0.299	$\mathbf{71.690 \pm 0.333}$
letterMvsN	95.369 ± 0.315	92.433 ± 0.777	95.114 ± 0.486	86.893 ± 0.870	$\mathbf{95.648 \pm 0.264}$
letterVvsY	88.213 ± 0.635	73.806 ± 1.551	84.723 ± 0.891	80.123 ± 1.359	$\mathbf{88.697 \pm 0.607}$
segment	96.528 ± 0.143	95.684 ± 0.155	$\mathbf{96.658 \pm 0.110}$	89.519 ± 1.760	96.545 ± 0.100
splice	79.931 ± 0.274	76.274 ± 0.895	78.560 ± 0.648	58.947 ± 0.853	$\mathbf{80.635 \pm 0.309}$
wdbc	$\mathbf{97.155 \pm 0.141}$	96.818 ± 0.191	96.862 ± 0.206	95.748 ± 0.247	97.111 ± 0.157

Table 3: ALHS versus the other algorithm based on t-test at 95% significance level

	Algorithms (win/tie/loss)			
Percentage of queries	UNCERTAIN	REPRESENT	QUIRE	DUAL
5%	6/3/0	7/2/0	6/3/0	5/4/0
10%	5/4/0	7/2/0	6/3/0	5/4/0

For some datasets, such as *wdbc* and *breast* in Figure 6(g) and 6(b), representative sampling approaches (REPRESENT, DUAL and QUIRE) achieve a better performance, while the result for UNCERTAIN is unsatisfactory. This unsatisfactory performance is possibly caused by the lack of awareness of unlabeled instances, which echoes our illustration in Figure 1. ALHS improves on UNCERTAIN by using the hints, and is comparable to other representative sampling algorithms.[1] On the other hand, in Figure 6(h), since *splice* is a larger and higher dimensional dataset, representative sampling algorithms that perform clustering (REPRESENT, DUAL) or label estimation (QUIRE) fail to reach a decent performance, while ALHS keeps a stable performance and slightly outperforms UNCERTAIN by using the hints.

In Figure 6, we see that ALHS can achieve comparable results to those of the best representative sampling and uncertainty sampling algorithms. As shown in Tables 1 and 2, after querying 5% of the unlabeled instances (Table 1), ALHS achieves the highest mean accuracy in 8 out of 9 datasets; after querying 10% of unlabeled instances (Table 2), ALHS achieves the highest mean accuracy in 6 out of 9 datasets. Table 3 further confirms that ALHS usually outperforms each of the other algorithms at the 95% significance level.

1. There are some more aggressive querying criteria (Tong and Koller, 2000) than UNCERTAIN and we have compared with those as additional experiments. Our preliminary observation was that those criteria can be worse than UNCERTAIN with soft-margin SVM and hence we excluded them from the tables.

6. Conclusion

We propose a new framework of active learning, *hinted sampling*, which exploits the unlabeled instances as hints. Hinted sampling can take both uncertainty and representativeness into account concurrently in a more natural and simpler way. We design a novel active learning algorithm ALHS within the framework, and couple the algorithm with a promising hint selection strategy. Because ALHS models the representativeness by hints, it avoids the potential problems of other more sophisticated approaches that are employed by other representative sampling algorithms. Hence, ALHS results in a significantly better and more stable performance than other state-of-the-art algorithms.

Due to the simplicity and effectiveness of *hinted sampling*, it is worth studying more about this framework. An intensive research direction is to couple hinted sampling with other classification algorithms, and investigate deeper on the hint selection strategies. While we use SVM in ALHS, this framework could be generalized to other classification algorithms. In the future, we plan to investigate more general hint selection strategies and extend hinted sampling from binary classification to other classification problem.

Acknowledgments

We thank Dr. Chih-Han Yu, the anonymous reviewers and the members of the NTU Computational Learning Lab for valuable suggestions. This work is supported by the National Science Council of Taiwan via the grant NSC 101-2628-E-002-029-MY2.

References

Y. S. Abu-Mostafa. Hints. *Neural Computation*, 4:639–671, 1995.

K. P. Bennett and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing Systems 11*, pages 368–374, 1998.

C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, pages 27:1–27:27, 2011.

D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

S. Dasgupta and D. Hsu. Hierarchical sampling for active learning. In *Proceedings of the 25th International Conference on Machine learning*, pages 208–215, 2008.

P. Donmez, J. G. Carbonell, and P. N. Bennett. Dual strategy active learning. In *Proceedings of the 18th European Conference on Machine Learning*, pages 116–127, 2007.

A. Frank and A. Asuncion. UCI machine learning repository, 2010.

Y. Guo and R. Greiner. Optimistic active learning using mutual information. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 823–829, 2007.

S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Semi-supervised SVM batch mode active learning for image retrieval. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1–7, 2008.

S.-J. Huang, R. Jin, and Z.-H. Zhou. Active learning by querying informative and representative examples. In *Advances in Neural Information Processing Systems 23*, pages 892–900, 2010.

D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

P. Melville and R. J. Mooney. Diverse ensembles for active learning. In *Proceedings of the 21st International Conference on Machine Learning*, pages 584–591, 2004.

H. T. Nguyen and A. Smeulders. Active learning using pre-clustering. In *Proceedings of the 21st International Conference on Machine Learning*, pages 623–630, 2004.

G. Ratsch, T. Onoda, and K. R. Müller. Soft margins for AdaBoost. *Machine Learning*, 2: 27:1–27:27, 2001.

B. Settles. Active learning literature survey. Technical report, University of Wisconsin–Madison, 2009.

S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *Proceedings of the 17th International Conference on Machine Learning*, pages 999–1006, 2000.

V. Vapnik. *Statistical learning theory*. Wiley, 1998.

Z. Xu, K. Yu, V. Tresp, X. Xu, and J. Wang. Representative sampling for text classification using support vector machines. In *Proceedings of the 25th European Conference on Information Retrieval Research*, pages 393–407, 2003.