MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Coverage Optimized Active Learning for k - NN Classifiers

Joshi, A.J.; Porikli, F.; Papanikolopoulos, N.

TR2012-036 May 2012

Abstract

Fast image recognition and classification is extremely important in various robotics applications such as exploration, rescue, localization, etc. k-nearest neighbor (kNN) classifiers are popular tools used in classification since they involve no explicit training phase, and are simple to implement. However, they often require large amounts of training data to work well in practice. In this paper, we propose a batchmode active learning algorithm for efficient training of kNN classifiers, that substantially reduces the amount of training required. As opposed to much previous work on iterative single sample active selection, the proposed system selects samples in batches. We propose a coverage formulation that enforces selected samples to be distributed such that all data points have labeled samples at a bounded maximum distance, given the training budget, so that there are labeled neighbors in a small neighborhood of each point. Using submodular function optimization, the proposed algorithm presents a nearoptimal selection strategy for an otherwise intractable problem. Further we employ uncertainty sampling along with coverage to incorporate model information and improve classification. Finally, we employ locality sensitive hashing for fast retrieval of nearest neighbors during classification, which provides 1-2 orders of magnitude speedups thus allowing real-time classification with large datasets.

IEEE International Conference on Robotics and Automation (ICRA)

Coverage Optimized Active Learning for $k - NN$ Classifiers

Ajay J. Joshi*, Fatih Porikli†, and Nikolaos Papanikolopoulos*
*University of Minnesota, Twin Cities †Mitsubishi Electric Research Labs

***Abstract*— Fast image recognition and classification is extremely important in various robotics applications such as exploration, rescue, localization, etc. k-nearest neighbor (kNN) classifiers are popular tools used in classification since they involve no explicit training phase, and are simple to implement. However, they often require large amounts of training data to work well in practice. In this paper, we propose a batch-mode active learning algorithm for efficient training of kNN classifiers, that substantially reduces the amount of training required. As opposed to much previous work on iterative *single-sample* active selection, the proposed system selects samples in *batches*. We propose a *coverage formulation* that enforces selected samples to be distributed such that all data points have labeled samples at a *bounded maximum distance*, given the training budget, so that there are labeled neighbors in a small neighborhood of each point. Using submodular function optimization, the proposed algorithm presents a near-optimal selection strategy for an otherwise intractable problem. Further we employ uncertainty sampling along with coverage to incorporate model information and improve classification. Finally, we employ locality sensitive hashing for fast retrieval of nearest neighbors during classification, which provides 1-2 orders of magnitude speedups thus allowing real-time classification with large datasets.**

I. Introduction

In many robotics applications using cameras, lasers or other sensors, it is important to make quick decisions based on sensor inputs. For example, determining whether an obstacle is present in the scene, whether an object of interest is approachable, finding whether a path exists to the goal, etc. The primary technique that often lies at the heart of such intelligent behavior is classification – to choose one out of multiple actions / categories for a certain sensor input.

Due to the natural requirements of fast real-time processing, along with the limited hardware resources available, simple and fast classification techniques such as k-nearest neighbor classifiers are extremely popular. One appealing aspect of these classifiers is that they require no explicit training phase – a complicated statistical model is not required to be learned. Instead, as a part of classification, the method chooses labeled training data that is close to the test input, and using these to perform classification.

However, collecting such training data so as to perform accurate classification is often a challenging task. Due to the large variability and high dimensionality of classification problems (with images and many other forms of data), a large number of training samples are needed. Recently, active learning has received a lot of attention towards achieving good classification with limited training. The main idea in active learning is to choose the "most informative" samples and avoid the redundant ones so that training effort is allocated effectively. However, challenges remain since notions of informativeness are difficult to capture, and depend on data types and distributions, statistical model characteristics, and computational limitations.

Typically, active learning is performed with discriminative classifiers such as Support Vector Machines / logistic regression where the notion of classification boundary is explicit. However, training such classifiers can be time-consuming and impractical for applications involving large amounts of data. On the other hand, even though classifiers such as kNN are extremely popular for real applications, we are not aware of work on active learning for the same. This paper proposes algorithms for active selection of informative training samples specifically for kNN classifiers. Furthermore, as opposed to iterative single sample selection, the algorithms we propose can be seamlessly used for *batch-mode* selection, which refers to querying for labels on batches of data samples and receiving feedback in one shot. This form of interaction can be advantageous since it can be easily performed in parallel, and inherently incorporates look-ahead which avoids myopic sample selection.

II. Related work

There has been a substantial recent interest in active learning, especially due to the limited quantities of training data that can be provided, which by current computing standards is much lesser than what can be used for training. As such, it is important to be able to effectively utilize whatever little annotated data that is available. The main goal of active learning is to select "informative samples" for training, thus focusing effort on the most important data to improve predictive performance. At the same time, this reduces redundancy in the chosen data.

Work in active learning initially focused on binary classification [3], [19]–[21], where it provided substantial reduction in label complexity (amount of annotated training data required) to achieve a certain classifier performance. There has been a lot of recent work on multi-class active learning, examples include [10], [11], [13], which has shown promising classification results with up to hundreds of categories. However, most of the proposed active learning algorithms are still iterative – they start with a random training set, and add the most informative samples at each round, after which the model is retrained. This iterative process can be extremely expensive in practice if computationally intensive models are to be learned, e.g., Support vector machines, Gaussian Process models, etc.

To overcome the problem due to iterative retraining, there has been some work on batch-mode active learning, examples include [6]–[8], [12]. These works perform batch-mode selection through attempting to improve discrimination

Preprint submitted to 2012 IEEE International Conference on Robotics and Automation. Received September 16, 2011.

performance [6], by reducing model uncertainty via Fisher information [7], or by maximizing a joint objective that measures informativeness and redundancy of the selected samples [12]. On the other hand our paper models batch-mode selection as a coverage problem, and then chooses training samples to maximize coverage. In order to incorporate current model information, sample similarity measures obtained from the current model could be used. The method proposed here is especially suitable for kNN classifiers, since they classify data using majority voting on nearest labeled data – as such, maximizing coverage to thereby minimize the distance to the labeled point for a given set of points is intuitively appealing.

Although typical batch-mode selection approaches overcome computational intractability of subset selection via certain approximations, note that the methods are still quite expensive, and hard to scale to very large data sizes. We employ kNN classifiers here since they are extremely useful for very large scale problems, primarily because no explicit training phase is involved, which can otherwise be time consuming. We also use locality sensitive hashing based approximations that allow the methods to perform extremely fast classification, since only the nearest neighbors are required.

III. Selection framework

In a $1-NN$ classifier, for each data point, the label is obtained by finding the label of the nearest labeled point. As such, our goal is to maximize the coverage of the training points. To this end, we want to ensure that all points have their nearest labeled point at a bounded distance. Consider that the distance between two points is given by the function $\mathbf{d} \in \mathbb{R}^d \mathrm{x} \mathbb{R}^d \rightarrow \mathbb{R}$. Denote the training set (to be chosen by active selection) as $\mathcal{S}$. Say the budget allows for only a certain number of labeled samples to be collected, such that $|\mathcal{S}| \leq n$. Also, denote the active pool from which samples are to be chosen by $\mathcal{A}$.

The minimum distance for any point x from the training set $\mathcal{S}$ is given by:

$$\min_{i:x_i \in \mathcal{S}} \quad \mathbf{d}(x_i, x), \tag{1}$$

The maximum distance for any point in the dataset to its nearest neighbor is then:

$$\max_{j:x_j \in \mathcal{A}} \quad \min_{i:x_i \in \mathcal{S}} \quad \mathbf{d}(x_i, x_j). \tag{2}$$

Our objective, as mentioned previously is to minimize the above distance, so that for each data point, there exist at least one training point at a bounded distance. Hence the goal is to solve the following

$$\mathcal{S} = \quad \operatorname*{argmin}_{|\mathcal{S}| \leq n} \quad \max_{j:x_j \in \mathcal{A}} \quad \min_{i:x_i \in \mathcal{S}} \quad \mathbf{d}(x_i, x_j). \tag{3}$$

For easier analysis, we instead define the objective function to be *maximized* as:

$$\begin{aligned} \mathbf{Q}(\mathcal{S}) &= \mathbf{d}_m - \max_{j:x_j \in \mathcal{A}} \quad \min_{i:x_i \in \mathcal{S}} \quad \mathbf{d}(x_i, x_j), \text{ for } \mathcal{S} \neq \phi \\ &= 0, \text{ otherwise.} \end{aligned} \tag{4}$$

where $\mathbf{d}_m \geq \mathbf{d}_{max}$, the maximum possible pairwise distance in the dataset. Thus, $\mathbf{d}_m$ is at least as large as the diameter of the data. As we will see later, the exact value of $\mathbf{d}_m$ is inconsequential for the actual implementation, since its value does not depend on the optimization variable.

Note that solving Equation (3) is equivalent to maximizing $\mathbf{Q}$ given in Equation (4), subject to the budget constraint. Thus, we can define our problem $\mathbf{P}$ as:

$$\mathbf{P}: \quad \max \quad \mathbf{Q}(\mathcal{S}) \quad \text{s.t. } |\mathcal{S}| \leq n. \tag{5}$$

The above problem in Equation (3) is similar to the known k-center problem is combinatorial optimization, where the goal is to find k-cluster centers along with point assignments so as to minimize the maximum cluster radius. It can be shown that the problem is intractable by a reduction to SET-COVER, for instance.

However, exploiting the properties of the objective function, strong approximation guarantees can be provided. We explore two such approaches in this paper:

- In the following, we present a simple proof that the above problem given in Equation (5) confirms to the paradigm of submodular optimization [17], and thus greedy near-optimal algorithms are applicable. A $(1 - 1/e)$ approximation guarantee can be provided for the corresponding maximization problem.
- It can be shown that a farthest-first heuristic algorithm for greedy selection of k centers leads to a near-optimal solution for Equation (3), with a 2-approximation for the minimization problem.

In this paper, we perform thorough empirical analysis of batch-mode active learning with both of the above approximation algorithms. Results show that the proposed methods substantially improve the classifiers trained by active learning, and at the same time are efficient for application to large-scale data.

Claim 1: $\mathbf{Q}$ as defined in Equation (4) is a monotonically non-decreasing function.

Proof: First note that for any $\mathcal{S} \neq \phi$, $\mathbf{Q} \geq 0$, since $\mathbf{d}_m$ is the at least as large as the largest pairwise distance in the point set. Further, it is easy to see that for any two sets $\mathcal{S}_1 \subseteq \mathcal{S}_2$, and $x_j \in \mathcal{A}$, $\min_{i:x_i \in \mathcal{S}_1} \mathbf{d}(x_i, x_j) \geq \min_{i:x_i \in \mathcal{S}_2} \mathbf{d}(x_i, x_j)$. Thus, the maximum such distance in the set $\mathcal{A}$ is also bounded, giving $\mathbf{Q}(\mathcal{S}_1) \leq \mathbf{Q}(\mathcal{S}_2)$. ■

Claim 2: $\mathbf{Q}$ is a submodular set function.

Proof: Assume $\mathcal{S}_1 \subseteq \mathcal{S}_2$ such that $\mathcal{S}_2 = \mathcal{S}_1 \cup \mathcal{S}$. Also, say x is a sample data point such that $x \in \mathcal{A}, x \notin \mathcal{S}_2$. Now

$$\begin{aligned} \mathbf{Q}(\mathcal{S}_2 \cup \{x\}) - \mathbf{Q}(\mathcal{S}_2) = \max_{j \in \mathcal{A}} \quad \min_{i \in \mathcal{S}_2} \mathbf{d}(x_i, x_j) \\ - \max_{j \in \mathcal{A}} \quad \min_{i \in \mathcal{S}_2 \cup \{x\}} \mathbf{d}(x_i, x_j). \end{aligned} \tag{6}$$

$$\begin{aligned} \mathbf{Q}(\mathcal{S}_1) - \mathbf{Q}(\mathcal{S}_1 \cup \{x\}) = \max_{j \in \mathcal{A}} \quad \min_{i \in \mathcal{S}_1 \cup \{x\}} \mathbf{d}(x_i, x_j) \\ - \max_{j \in \mathcal{A}} \quad \min_{i \in \mathcal{S}_1} \mathbf{d}(x_i, x_j). \end{aligned} \tag{7}$$

Denote

$$\begin{aligned} M_1(j) &= \min_{i \in \mathcal{S}_1} \mathbf{d}(x_i, x_j), \\ M_2(j) &= \min_{i \in \mathcal{S}_1 \cup \{x\}} \mathbf{d}(x_i, x_j), \\ \mathbf{Q}_d &= \mathbf{Q}(\mathcal{S}_2 \cup \{x\}) - \mathbf{Q}(\mathcal{S}_2) + \mathbf{Q}(\mathcal{S}_1) - \mathbf{Q}(\mathcal{S}_1 \cup \{x\}). \end{aligned} \tag{8}$$

Preprint submitted to 2012 IEEE International Conference on Robotics and Automation. Received September 16, 2011.

Thus,

$$\begin{aligned}\mathbf{Q}_d &= \max_{jin\mathcal{A}} \min_{i\in\mathcal{S}_1\cup\mathcal{S}} \mathbf{d}(x_i,x_j) - \max_{j\in\mathcal{A}} M_1(j) \\ &+ \max_{j\in\mathcal{A}} M_2(j) - \max_{j\in\mathcal{A}} \min_{i\in\mathcal{S}_1\cup\mathcal{S}\cup\{x\}} \mathbf{d}(x_i,x_j). \\ &= \max_{j\in\mathcal{A}} \min(M_1(j), \min_{i\in\mathcal{S}} \mathbf{d}(x_i,x_j)) - \max_{j\in\mathcal{A}} M_1(j) \\ &+ \max_{j\in\mathcal{A}} M_2(j) - \max_{j\in\mathcal{A}} \min(M_2(j), \min_{i\in\mathcal{S}} \mathbf{d}(x_i,x_j)).\end{aligned} \tag{9}$$

Denote $\delta = \max_j M_1(j) - \max_j M_2(j)$. Since $M_1(j) \geq M_2(j), \forall j$, we have $\delta \geq 0$, and

$$\begin{aligned}\mathbf{Q}_d &= \max_{j\in\mathcal{A}} \min(M_1(j), \min_{i\in\mathcal{S}} \mathbf{d}(x_i,x_j)) \\ &- \max_{j\in\mathcal{A}} \min(M_2(j), \min_{i\in\mathcal{S}} \mathbf{d}(x_i,x_j)) - \delta.\end{aligned} \tag{10}$$

We divide the analysis in 3 cases below.
Case 1:

$$M = \min_{i\in\mathcal{S}} \mathbf{d}(x_i,x_j) \leq M_2(j).$$

Then,

$$\begin{aligned}\mathbf{Q}_d &= \max_{j\in\mathcal{A}} M - \max_{j\in\mathcal{A}} M - \delta \\ &= -\delta \leq 0.\end{aligned} \tag{11}$$

Case 2:

$$M = \min_{i\in\mathcal{S}} \mathbf{d}(x_i,x_j) \geq M_1(j).$$

Then,

$$\begin{aligned}\mathbf{Q}_d &= \max_{j\in\mathcal{A}} M_1(j) - \max_{j\in\mathcal{A}} M_2(j) - \delta \\ &= 0.\end{aligned} \tag{12}$$

Case 3:

$$M_2(j) < M = \min_{i\in\mathcal{S}} \mathbf{d}(x_i,x_j) < M_1(j).$$

Then,

$$\mathbf{Q}_d = M - \max_{j\in\mathcal{A}} M_2(j) - \delta \leq 0. \tag{13}$$

From Equations (11), (12), and (13), we have $\mathbf{Q}_d \leq 0$. Thus from Eqn (8),

$$\mathbf{Q}(\mathcal{S}_2 \cup \{x\}) - \mathbf{Q}(\mathcal{S}_2) \leq \mathbf{Q}(\mathcal{S}_1 \cup \{x\}) - \mathbf{Q}(\mathcal{S}_1), \tag{14}$$

where $\mathcal{S}_1 \subseteq \mathcal{S}_2$, thus showing that $\mathbf{Q}$ is submodular. ■

Intuitively, this means that adding an element to a smaller set presents more value than adding it to a larger set – the property of diminishing returns.

Given a submodular non-decreasing set function $\mathbf{Q}$ such that $\mathbf{Q}(\phi) = 0$, Nemhauser et al. [17] show that a greedy algorithm gives an objective value no worse than a $(1-1/e)$ factor of the optimal. Note that even though the globally optimal solutions to both Equations (3) and (5) are the same, the approximation guarantees are not. Specifically, the greedy algorithm $(1-1/e)$-approximation bound does not hold for Equation (3), and the solution can be arbitrarily worse. To see this, assume that the optimal value $\mathbf{Q}^* = \mathbf{d}_m - d^*$, where d^* is the optimal distance of interest. Also, denote by $\hat{\mathbf{Q}}$ the achieved value of $\mathbf{Q}$, and $\hat{d}$ the corresponding distance. Let c be the approximation factor. Then

$$\mathbf{d}_m - \hat{d} = c * (\mathbf{d}_m - d^*), \tag{15}$$

which implies that $\hat{d}$ can be far away from d^*, and the actual approximation obtained depends on $\mathbf{d}_m$, which is independent of our desired optimization variable. However, in our experiments, we often see approximations that are much closer to the optimal giving good results in practice.

Given near-optimality guarantees, tools in submodular optimization have been used extensively for problems in experiment design, sensor placements, network outbreak detections, etc. For example, in [14]–[16], submodular optimization techniques are used for effective sensor placement in Gaussian Processes and other graphical models.

A. kNN and submodularity

Now we turn our attention to the kNN classifier – unfortunately, the previous analysis does not apply here. It is straightforward to create a counter example showing that the analogous problem for k nearest neighbors does not give a submodular objective function. For example, consider a point x and a set $\mathcal{S}_2 = \{x_1, x_2, x_3, x_4\}, \mathcal{S}_1 = \{x_3, x_4\}$, so that $\mathcal{S}_1 \subset \mathcal{S}_2$. Further assume that the samples are at distances such that x_1 is the closest to x and x_4 is the farthest in $\mathcal{S}_2$. Also, let $\mathbf{d}(x,x_2) - \mathbf{d}(x,x_1) > \mathbf{d}(x,x_4) - \mathbf{d}(x,x_3)$. If we consider $k = 2$, and add a point $\hat{x}$ which is closest to x amongst all points, we can see that the property of diminishing returns is violated – i.e., set $\mathcal{S}_2$ encounters larger increase in objective value than set $\mathcal{S}_1$ according to Equation (4). Thus the corresponding objective function is not submodular.

Instead of looking at the maximum distance to the k^{th} nearest neighbor, one way might be to instead minimize the average distance. Note that this problem (also intractable in general) is slightly different from the k-median problem heavily studied in the literature. In the k-median problem, the goal is to minimize the sum of distances of points to their cluster centers, which is different from our problem of minimizing the average distance[1] from each data point.

Due to the computational intractability of the problem, we still use the $1NN$ formulation even when using a kNN classifier with k different from 1. This approach performs well empirically.

B. Greedy algorithm

Figure 1 describes the greedy batch-mode active selection algorithm. As mentioned before, the algorithm achieves an objective function value that is within a $(1-1/e)$ factor of the optimal value.

C. Computational requirements

Even though the above greedy algorithm is polynomial in the unlabeled data size and the batch size, it can still be slow in practice. Specifically, the algorithm has a time bound $\mathcal{O}(Nk^2)$, with N points, and a batch size of k, and performs worse with larger batches.

IV. Greedy algorithm for k-center

The optimization formulation given in Equation (3) is the k-center problem studied heavily in the literature. In

[1]Minimizing the average is equivalent to minimizing the sum since all points have k near neighbors considered.

Input: Unlabeled data pool $\mathcal{A}$, batch-size k

1. $\mathcal{S} := \{\phi\}$, the current batch of examples;
2. **for** $i := 1$ **to** k, **do**
3. **foreach element** $x \in \mathcal{A} \backslash \mathcal{S}$, **do**
4. Compute $\mathbf{Q}(\mathcal{S} \cup \{x\})$ using Equation (5);
5. Select the example $x^* = \text{argmax}_x \, \mathbf{Q}(\mathcal{S} \cup \{x\})$;
6. $\mathcal{S} := \mathcal{S} \cup \{x^*\}$;
7. **end**
8. return $\mathcal{S}$.

Output: The actively selected batch $\mathcal{S}, |\mathcal{S}| = k$.

Fig. 1. A greedy batch-mode active selection algorithm.

Input: Unlabeled data pool $\mathcal{A}$, batch-size k

1. $\mathcal{S} := x_r$, $x_R \in \mathcal{A}$, a randomly chosen sample;
2. **for** $i := 2$ **to** k, **do**
3. Find $x^* \in \mathcal{A} \backslash \mathcal{S}$ such that $x^* = \text{argmax}_x \, d(x, \mathcal{S})$;
6. $\mathcal{S} := \mathcal{S} \cup \{x^*\}$;
7. **end**
8. return $\mathcal{S}$.

Output: The actively selected batch $\mathcal{S}, |\mathcal{S}| = k$.

Fig. 2. Greedy farthest-first active selection algorithm.

this setting, greedy algorithms such as farthest first point selection have been explored for actively seeding clustering [2]. The algorithm begins with a point chosen randomly from the unlabeled pool, and at each iteration a new point is picked to be farthest from the current chosen set. The distance of a point to a set implies the distance from the point to its closest element in the set. It can be shown that this simple greedy algorithm gives a factor of 2 approximation [22] for the minimization problem in Equation (3). We describe the algorithm in Figure 2.

Even though the simplicity of the algorithm is appealing, it tends to pick samples at the boundary of the data. As such, the chosen samples are not representatives of the data, and often lead to poor generalization. Since labeled samples influence the accuracy of classification of nearby points, choosing samples on the boundary (or outliers in a sense) does not improve classification. As we show in the next section, this problem can be alleviated by incorporating model information in the formulation.

V. Incorporating classifier information

The methods described so far confirm well to the coverage formulation, so that samples selected for training are distributed well across the training set. However, no information pertinent to classification is used – thus even though these methods successfully solve the optimization problem to near-optimality they do not work well in the classification setting. For instance, consider the case where the data is highly imbalanced, such that one class has many data samples, while there are other classes with very sparse populations. In this case, it is not useful to cover the data set with well-distributed training samples, since most such samples would be from the same class – instead, it would be more beneficial to actively seek sparser classes. Label information is thus extremely important in addition to coverage. In this section, we improve the models to account for current information from the training data, so that only data which is "informative" for the *current model* is chosen for training.

The main idea is to use model uncertainty to bias the method towards the selection of points for which there is classification uncertainty. Similar approaches have been used for other classifiers like SVM and are known as uncertainty sampling [11], [13]. Here we combine uncertainty sampling with coverage objectives in the previous sections to incorporate both pieces of information. The experiments demonstrate that this method significantly improves classification accuracy, and also requires lesser computation.

We require notions of uncertainty that capture the value of obtaining the label for a given data point. Also, we require the uncertainty measure to be applicable to classification problems with many classes, and be amenable to fast computation. Here, we focus on the proportion of points coming from the different classes amongst the k nearest neighbors of a sample point. For instance if all k neighbors belong to the same class, the classification is lesser confusion, and choosing that sample to obtain the label might be redundant. On the other hand, if the k neighbors has each sample coming from a different class, then the classifier is uncertain about the membership of the sample point, and thus it is "informative" to query. More precisely, our uncertainty score is the difference between the number of points coming from the most populous class and the second most populous class amongst the k nearest neighbors. This difference is similar to the notion of multi-class margin used in other uncertainty sampling approaches [11], and results show that it gives better performance compared to considered other measures such as the entropy of the distributions.

In order to incorporate this measure into the coverage formulation, we make a small modification to Equation 3. Instead of searching over the entire active pool for samples, we restrict the search to samples that give a high uncertainty score. Specifically, if the batch of examples to be selected is of size b, we simply choose $b' = mb$ most uncertain samples over which the more extensive optimization is carried out, for a multiplier m (say 5). The underlying hypothesis is that given informative samples, coverage is an important criterion to ensure good distribution of training samples. Apart from improved classification, we also get the benefit of substantially faster computation since the farthest first algorithms and greedy submodular set selection only need to be performed on the reduced informative sample set. Note that the most informative samples can be chosen in $\mathcal{O}(Nk)$ time, which is a factor of k improvement on the previous algorithms. The actual subset selection then runs in time independent of N, only relying on the required batch size.

Preprint submitted to 2012 IEEE International Conference on Robotics and Automation. Received September 16, 2011.

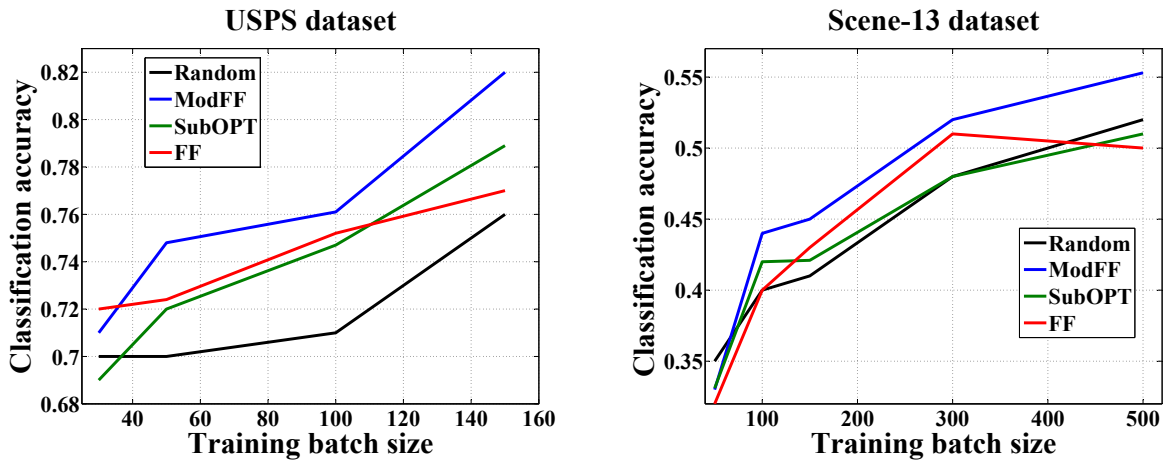

Fig. 3. Classification accuracy values with increasing batch sizes for USPS and Scene-13 datasets. FF – farthest first greedy selection, SubOPT – greedy algorithm using submodular optimization, ModFF – Farthest first modified using informative sample subsampling. Note that SubOPT performs as bad as random for the Scene-13, perhaps due to very high dimensional data. Also, the improvements due to ModFF are smaller for Scene-13 than USPS.

VI. Fast selection with LSH

Locality sensitive hashing (LSH) is a popular technique for fast approximate near neighbor search in high dimensions. It is particularly suited to our problem since the proposed algorithms require repeated computation of nearest neighbors, in order to update the chosen set, as well as perform classification. Also, since the descriptors we use are high dimensional, it can be expensive to do a linear scan to find nearest neighbors. In the following, we give a brief overview of one particular form of LSH for Euclidean spaces using p-stable distributions.

Consider a d-dimensional space $\mathbb{R}^d$, with the p-norm denoted by $||v||_p^d$ for vector v. Let the metric space be $\mathcal{M} = (X, d)$, in which the ball of radius r centered at q is defined as $B(q, r) = \{v \in X \mid d(v, q) \leq r\}$.

Given a dataset P and a query q, in the (R, c)–near neighbor (NN) problem [9], one has to retrieve points p such that $d(p, q) \leq cR$, if there exists a point in P within distance R from q. In other words, the approximate nearest neighbors retrieved must be bounded close to the true nearest neighbor.

Definition 1: [9] A LSH family $\mathcal{H} = \{h : S \rightarrow U\}$ is called (r_1, r_2, p_1, p_2)-sensitive for D if for any $u, v \in S$,

- if $u \in B(v, r_1)$, then $Pr_{\mathcal{H}}[h(u) = h(v)] \geq p_1$,
- if $u \notin B(v, r_2)$, then $Pr_{\mathcal{H}}[h(u) = h(v)] \leq p_2$.

If $p_1 > p_2$ and $r_1 < r_2$, the family $\mathcal{H}$ can be used for the (R,c)-NN problem. The basic idea is that the hash functions evaluate to the same values with high probability for points that are close to each other, whereas for distant points the probability of matching (collision) is low. The probability gap can be increased by concatenation of multiple hash functions chosen randomly from the family $\mathcal{H}$.

Using the notation in Datar et al. [4], define the function family $\mathcal{G} = \{g : S \rightarrow U^k\}$, where $g(v) = \{h_1(v), \ldots, h_k(v)\}$, where $h \in \mathcal{H}$. For a given L, choose $g_1, \ldots, g_L$ uniformly at random from $\mathcal{G}$. k and L can be chosen to satisfy the desired collision probability guarantees as described in the next section.

In the pre-precessing step, each data sample from the dataset is stored in buckets $g_i(x), i \in \{1, \ldots, L\}$. For a given query q, points from all the buckets $g_i(q), i \in \{1, \ldots, L\}$ are retrieved. The nearest neighbor from these retrieved points is then returned as the approximate nearest neighbor. It is shown in [4] that given a (R, cR, p_1, p_2)–sensitive family $\mathcal{H}$ for the distance measure d, then there exists an algorithm that solves the (R, c)-NN problem in query time $\mathcal{O}(N^\rho)$, where N is the dataset size and $\rho = \frac{ln1/p_1}{ln1/p_2}$. Thus for $p_1 > p_2$, the above results in sublinear time retrieval.

It is further shown in [4] that the hash functions $h_{\mathbf{a},b}(\mathbf{x}) = \lfloor \frac{\mathbf{a} \cdot \mathbf{x} + b}{r} \rfloor$, where each element of $\mathbf{a}$ is sampled from $\mathcal{N}(0, 1)$, and b chosen uniformly from $[0, r]$ represents a (R, cR, p_1, p_2)-sensitive LSH family for the Euclidean distance measure. The result crucially relies on the fact that the Gaussian distribution from which $\mathbf{a}$ is sampled is a 2-stable distribution. As such, we use this hash family for fast search of informative samples in our active learning algorithm.

VII. Experiments

In this section we perform experiments on 3 different real-world datasets, namely US Postal service handwritten digits dataset (USPS) [1], the Letter recognition dataset (Letter) [1] and the Scene-13 dataset [5] consisting of images from 13 natural scene categories. For the USPS and Letter datasets, vectorized pixel values were used as features, whereas for Scene-13, 384-dimensional Gist descriptors [18] that give a scene summary were used.

For USPS and Letter, we used 5000 samples in the unlabeled pool, and 1000 samples for testing. For Scene-13, 1000 samples were used in the unlabeled pool, along with another 1000 for testing. We experimented with kNN for varying k between 1 and 10. The figures show results obtained using the 1 nearest neighbor classifier, however, other results are very similar.

Figure 3 shows classification accuracy values as a function of batch size, for two different datasets. We compare

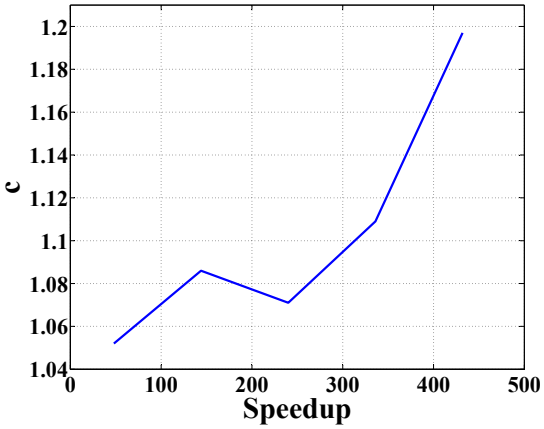

Fig. 4. Speedup for approximate near-neighbor search using Locality Sensitive Hashing. As expected, the speedup factor increases with a weaker approximation in terms of distance to nearest neighbors.

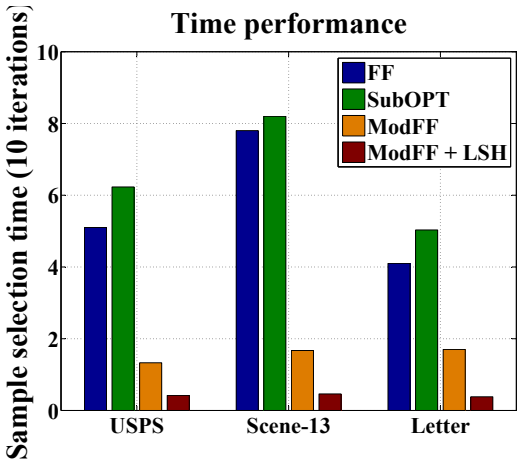

Fig. 5. Time required for selection of samples from the active pool for different datasets. Note that even though Farthest first and Submodular optimization are greedy algorithms, the quadratic scaling with respect to the batch size makes them slow. Modified Farthest first is much faster due to the sub-sampling of informative samples that essentially reduces pool size. Using locality sensitive hashing results is a further order of magnitude speedup.

the random selection, greedy farthest first algorithm (FF), greedy submodular optimization algorithm (SubOPT), and the modified farthest first incorporating model information (ModFF). Note that the modified algorithm performs significantly better than the two greedy algorithms, which themselves beat random selection. This shows that active learning can be used to chose informative samples to achieve higher classification accuracy with the same training effort.

In Figure 4, we show the speedup achieved for near neighbor queries on the USPS dataset using LSH. The y-axis shows the approximation factor c, whereas the x-axis indicates the speedup factor. Note that for reasonably close approximations, a 500-fold speedup is achievable. Furthermore, the approximation negligibly affected accuracy in our experiments.

Finally, we show time results of the 4 different methods (3 methods shown above) along with the addition of the LSH approximation in Figure 5. As mentioned before, incorporating model information has the desirable side-effect of reducing the search space for the greedy algorithm, thus resulting in improved performance. The LSH method further speeds this up by at least 1 order of magnitude. Also note that this approximation is even more useful for larger datasets due to the asymptotic improvements in running time.

VIII. Conclusion and Future Work

In this paper, we proposed efficient active learning algorithms for kNN classifiers. The proposed methods use greedy algorithms that provide near-optimal solutions to otherwise intractable problems in a coverage based selection formulation. We also proposed a simple way to incorporate model information along with coverage to result in improved classification performance. Finally, LSH based approximation allow the methods to scale to very large scale data, while still retaining classification accuracy. Future work will focus on extending the work to ensemble classifiers, which perform very well on many real-world applications.

References

[1] A. Asuncion and D. J. Newman. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2007. Available at `http://archive.ics.uci.edu/ml/datasets.html`.
[2] S. Basu, A. Banerjee, and R. Mooney. Semi-supervised clustering by seeding. In *ICML*, 2002.
[3] C. Campbell, N. Cristianini, and A. J. Smola. Query learning with large margin classifiers. In *ICML*, 2000.
[4] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on computational geometry*, 2004.
[5] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *CVPR*, 2005.
[6] Y. Guo and D. Schuurmans. Discriminative batch mode active learning. In *NIPS 07*, 2007.
[7] S. C. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Semi-supervised SVM batch mode active learning for image retrieval. In *CVPR*, 2008.
[8] S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, 2006.
[9] P. Indyk and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. In *Proceedings of the Symposium on Theory of Computing*, 1998.
[10] P. Jain and A. Kapoor. Active learning for large multi-class problems. In *CVPR*, 2009.
[11] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *CVPR*, 2009.
[12] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class batch-mode active learning for image classification. In *ICRA*, 2010.
[13] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with Gaussian Processes for object categorization. In *ICCV*, 2007.
[14] A. Krause and C. Guestrin. Near-optimal nonmyopic value of information in graphical models. In *UAI*, 2005.
[15] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Robust sensor placements at informative and cost-effective locations. *ACM Transactions on Sensor Networks*, 7(4), 2011.
[16] A. Krause, H. B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. Technical Report CMU-ML-08-100, Carnegie Mellon University, 2008.
[17] G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978.
[18] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.
[19] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Conf. Computational Learning Theory*, pages 287–294, 1992.
[20] S. Tong and E. Chang. Support vector machine active learning for image retrieval. In *MULTIMEDIA '01: Proceedings of the ninth ACM international conference on Multimedia*, 2001.
[21] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *JMLR*, 2:45–66, 2001.
[22] D. Williamson and D. Shmoys. *The design of approximation algorithms*. Cambridge University Press, 2010.