Active Learning for Large Multi-class Problems

Prateek Jain[*]
University of Texas at Austin
Austin, TX 78759 USA
pjain@cs.utexas.edu

Ashish Kapoor
Microsoft Research
Redmond, WA 98052 USA
akapoor@microsoft.com

Abstract

Scarcity and infeasibility of human supervision for large scale multi-class classification problems necessitates active learning. Unfortunately, existing active learning methods for multi-class problems are inherently binary methods and do not scale up to a large number of classes. In this paper, we introduce a probabilistic variant of the K-Nearest Neighbor method for classification that can be seamlessly used for active learning in multi-class scenarios. Given some labeled training data, our method learns an accurate metric/kernel function over the input space that can be used for classification and similarity search. Unlike existing metric/kernel learning methods, our scheme is highly scalable for classification problems and provides a natural notion of uncertainty over class labels. Further, we use this measure of uncertainty to actively sample training examples that maximize discriminating capabilities of the model. Experiments on benchmark datasets show that the proposed method learns appropriate distance metrics that lead to state-of-the-art performance for object categorization problems. Furthermore, our active learning method effectively samples training examples, resulting in significant accuracy gains over random sampling for multi-class problems involving a large number of classes.

1. Introduction

Collecting data for training a large scale classification system is a potentially expensive process, as most of the existing systems rely on human supervision to obtain accurate labels. Furthermore, most of the real world applications, such as image search and object recognition, involve hundreds of labels. Thus, the user needs to select from amongst a large class of labels in addition to handling a huge data set. Hence, to make the most of scarce human labeling resources, it is imperative to carefully select the data points that a user labels.

Recent research in the area of active learning have been reasonably successful in handling the problem of active selection of the examples to be labelled. Typically, such schemes build upon notions of uncertainty in classification. For example, data points that are most likely to be misclassified can be considered to be the most informative and will be selected for supervision. However, most of the existing research in active learning has focused on binary classification tasks. Relatively few approaches [15, 21, 12, 18, 13] have been proposed for multi-class active learning and are typically based on extensions of predominantly binary active learning methods to the multi-class scenario. A crucial drawback of such methods is that the underlying classification models consist of a collection of *independent* binary classification subproblems, e.g., 1-vs-all GP. Consequently, the model cannot compare the uncertainty in prediction of an example for two different binary subproblems, implying that it cannot identify the classes that require more training data. As a result, these methods do not scale-up well with the number of classes. Figure 1 shows that for large number of classes, our proposed method (pKNN+AL) consistently outperforms the existing Gaussian Process (GP) [13] and SVM [15] based active learning methods in terms of gain over random sampling[1].

Ideally, an active learning scheme should be designed on top of a principled multi-class system that can provide a sound way of comparing classification uncertainty across classes. The K-nearest neighbor classifier (KNN) is a suitable candidate as a principled multi-class classifier. It is one of the simplest classification schemes that classifies a data point based on the labels of its neighbors, and can naturally handle multi-class problems. But, KNN does not admit a natural notion of uncertainty in classification, and hence, it is unclear how to estimate the probability of misclassification for a given data point.

In this paper, we introduce a probabilistic variant of the K-nearest neighbor (KNN) method for classification that can be further used for active learning in multi-class scenarios. Our method defines a probability measure based on the pairwise distances between the data points, leading to a

[*] Work done while at Microsoft Research, Redmond.

[1] Throughout this paper, *random sampling* will refer to the strategy of choosing a random selection of examples to label.

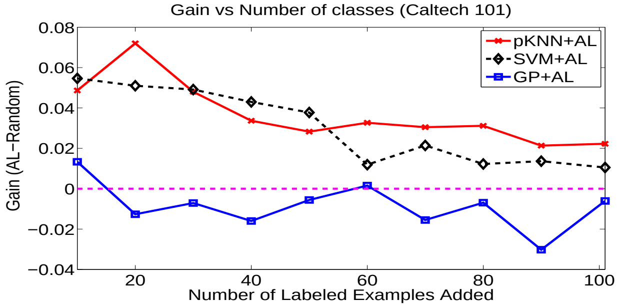

Figure 1. Accuracy gained by various active learning methods over random sampling after 10 rounds of active learning for different subsets of the Caltech101 dataset. Proposed method (pKNN+AL) has significant gains even for a large number of classes, and consistently outperforms both GP and SVM based active learning.

natural notion of uncertainty over the class labels, and can be used for active sampling of examples to be labelled so as to maximize the discriminating capabilities of the model. The performance of a nearest neighbor method is critically dependent on the distance/similarity measure that it operates on. While general purpose measures like L_p norms or standard image kernels are reasonably accurate for simple problems, they typically fail to reflect the correct distance measure for complex large scale problems involving a large number of classes and data points in high dimensions. Recent advances in metric/kernel learning make it possible to learn more effective data driven distance (or similarity) functions for a given problem using partially labeled data or pairwise distance constraints. Typically, these methods learn a Mahalanobis metric (or a kernel matrix) that accurately reflects the pairwise distances. Consequently, most of the metric learning methods involve optimizing over quadratic (either in the dimensionality of the feature space or the number of data points) number of parameters, thus making them prohibitively expensive for large scale classification problems. However, classification problems should require smaller number of parameters as they require accurate distances to "classes" only, rather than accurate pairwise distances. In this paper, we present a method that provides a scheme that learns an accurate distance to each class and is parameterized by just linear number of parameters, leading to a highly scalable and robust distance learning method for classification and active learning.

Our contributions are twofold: 1) an efficient distance function learner along with a probabilistic variant of the K-nearest neighbor classifier for accurate classification in the multi-class scenario, and 2) an active learning paradigm based on the proposed classifier that can handle a large number of classes. We empirically demonstrate the effectiveness of the proposed classification framework and the active learning method on a variety of problems including an object recognition task.

2. Related Work

Recent work in the area of active learning has yielded a variety of heuristics based on the variance in prediction [13], version space of SVMs [19, 4], disagreement among classifiers [7] and expected informativeness [14, 17]. Most of these heuristics are designed for binary classification and are applicable primarily to the classifiers such as SVMs and GPs. Extensions to the multi-class scenario are typically based on extensions of binary classification using pairwise comparisons or 1-vs-all strategy. Representative research includes methods based on functions of margin loss [15, 21] or entropy [13, 12]. However, all of these methods have been demonstrated on problems that have fewer than 15 classes, and as shown in Figure 1, typically do not scale well with increasing number of classes. This is because, these multi-class extensions consist of *independent* binary subproblems and it is not clear that how uncertainty of a data point w.r.t. different binary subproblems can be compared. Specifically, let point $\boldsymbol{x}_i$ be the most uncertain point according to the binary subproblem "class p-vs-all" for class p and let $\boldsymbol{x}_j$ be the most uncertain point according to the binary subproblem "class q-vs-all" for class q, then it is not clear how the uncertainty in the points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ can be compared. In contrast, our method is designed specifically for multi-class problems and provides a natural notion of comparing uncertainty in classification over all the classes.

The K-Nearest Neighbor (KNN) method and its variants provide a natural way of handling multi-class problems [6]. But, these methods depend heavily on the provided distance or similarity measure to compare two data points. Recent methods in metric learning learns a distance metric parameterized by a Mahalanobis metric that is consistent with the training data [1, 5, 20, 8]. Although these methods provide a principled way for handling multi-class problems, for large scale problems these methods are prohibitively expensive. Furthermore, there have been only a few attempts to perform active learning with nearest neighbor methods [16, 10, 22, 11] and mostly, have been limited to a small number of classes. Closely related to our work are the active distance learning approaches proposed by Yang et. al. [22], and Hoi & Jin [11]. Both of these methods aims to improve pairwise distances using pairwise constraints, but either do not have a kernelized version [22] or are designed for the transductive setting [11]. In contrast, our method is designed for classification problems and can be used to actively learn kernel matrices in the inductive setting.

3. Problem Formulation

Given a set of n points $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ in $\mathbb{R}^d$ and their corresponding class labels $\{y_1, y_2, \ldots, y_n\}$ with $1 \leq y_i \leq C, \forall i$, we define the class label y of a given data point $\boldsymbol{x}$ as:

$$y = \operatorname*{argmax}_c \frac{1}{n_c} \sum_{i \text{ s.t. } y_i = c} K(\boldsymbol{x}, \boldsymbol{x}_i), \qquad (1)$$

where n_c is the number of data points with class label c, C is the total number of classes, and $K(\boldsymbol{a}, \boldsymbol{b})$ denotes some similarity measure between data points $\boldsymbol{a}$ and $\boldsymbol{b}$. Thus, $\frac{1}{n_c}\sum_{i \text{ s.t. } y_i=c} K(\boldsymbol{x}, \boldsymbol{x}_i)$ denotes average similarity of $\boldsymbol{x}$ to the class c. If K is a non-negative similarity measure then (1) can be equivalently written as:

$$y = \operatorname*{argmax}_{c} p_c(\boldsymbol{x}), \tag{2}$$

where $p_c(x) = \frac{\frac{1}{n_c}\sum_{i \text{ s.t. } y_i=c} K(\boldsymbol{x},\boldsymbol{x}_i)}{\sum_{t=1}^{C}\frac{1}{n_t}\sum_{i \text{ s.t. } y_i=t} K(\boldsymbol{x},\boldsymbol{x}_i)}$, and can be interpreted as the probability of point $\boldsymbol{x}$ belonging to the class c. Also, note that this formulation is closely related to kernel density estimation using Parzen windows and can be derived by considering a uniform prior probability over the class labels. Furthermore, this formulation is better suited for learning more accurate kernels, and extends naturally to the problem of active learning for large number of classes.

Success of the above defined classifier (Equation (2)) is critically dependent on the similarity measure (K) that is used to compare data points. A good distance/similarity measure should accurately reflect the true underlying relationships, i.e., data points in the same category should have higher similarity measure than data points in different categories. Unfortunately, for most of the practical applications, the data points are high dimensional vectors (as in the case of images) and it might be non-trivial to know the distance/similarity function beforehand. An alternate approach is to learn a similarity measure from the data directly. However, a single similarity measure might not be powerful enough for multi-class problems involving a large number of classes. Instead, we devise a method that learns a similarity measure for each of the C classes using the provided training data and the associated label information.

Given an initial $n \times n$ positive definite similarity matrix K_0 that reflects the prior domain knowledge about the similarity measure, we learn a new positive definite similarity measure per class K^c that is "close" to K_0 but is also consistent with the labeled training data. We pose the problem of learning the kernels $\{K^1, K^2, \ldots, K^C\}$ as an optimization problem that minimizes the *LogDet* divergence between the learned matrices $\{K^1, K^2, \ldots, K^C\}$ and K_0 such that the learned kernels $\{K^1, K^2, \ldots, K^C\}$ satisfy specified class-label constraints. Formally, the optimization problem is:

$$\begin{aligned}
\min_{K^1,\ldots,K^C} \quad & \sum_{l=1}^{C} D_{\ell d}(K^l \| K_0), && \\
\text{s.}t. \quad & p_c(\boldsymbol{x}_i) \ge \alpha, && c = y_i, \forall i, \\
& p_c(\boldsymbol{x}_i) \le \beta, && c \ne y_i, 1 \le c \le C, \forall i, \\
& K^c \succeq 0, && \forall c,
\end{aligned} \tag{3}$$

where α and β are positive constants, and $p_c(\boldsymbol{x})$ is as defined earlier. Thus, the constraint $p_c(\boldsymbol{x}_i) \ge \alpha$ requires that the probability of classifying a point $\boldsymbol{x}_i$ as its true class label $c = y_i$ is at least α. Similarly, the constraint $p_c(\boldsymbol{x}_j) \le \beta$ requires that the probability of assigning a point $\boldsymbol{x}_j$ to an incorrect class $c \ne y_j$ is at most β. Note that, if the similarity measures K^c are all not positive then $p_c(\boldsymbol{x})$ does not define a valid probability distribution. But, the problem (3) still is well defined because then the constraint $p_c(\boldsymbol{x}_i) \ge \alpha$ requires the average similarity of a point $\boldsymbol{x}_i$ to its true class y_i to be greater than α times average similarity of $\boldsymbol{x}_i$ to all the classes. Similarly, the constraint $p_c(\boldsymbol{x}_j) \le \beta$ can be interpreted appropriately.

The objective function $D_{\ell d}(K \| K_0) = \operatorname{tr}(KK_0^{-1}) - \log\det(KK_0^{-1}) - n$ is the LogDet divergence between the matrix K and K_0, and has previously been shown to be useful in the context of distance learning [5]. It is a generalized Bregman matrix divergence defined over positive definite matrices and is defined only when the range space of matrix K is the same as that of K_0. Thus the constraint $K_0 \succeq 0$ is trivially satisfied and computationally expensive procedure of semi-definite programming is not required.

Using the definition of p_c, the constraint $p_c(\boldsymbol{x}_i) \ge \alpha$ can be written as a linear constraint in all $\{K^1, K^2, \ldots, K^C\}$: $\operatorname{tr}(\sum_{l=1}^{C} K^l \boldsymbol{e}_i (\boldsymbol{v}^l)^T) \le \frac{1}{\alpha}\operatorname{tr}(K^c \boldsymbol{e}_i (\boldsymbol{v}^c)^T)$, where $\boldsymbol{v}^l$ denotes the indicator vector for class l, i.e. $\boldsymbol{v}^l(i) = 1$ if $y_i = l$. Vector $\boldsymbol{e}_i$ denotes the i-th standard basis vector. Similarly, constraint $p_c(\boldsymbol{x}_i) \le \beta$ can be written as $\operatorname{tr}(\sum_{l=1}^{C} K^l \boldsymbol{e}_i (\boldsymbol{v}^l)^T \ge \frac{1}{\beta}\operatorname{tr}(K^c \boldsymbol{e}_i (\boldsymbol{v}^c)^T)$. Thus, (3) can be equivalently written as:

$$\begin{aligned}
&\min_{K^1..K^C} \sum_{l=1}^{C} D_{\ell d}(K^l \| K_0), \\
&\quad \text{s.}t. \ \operatorname{tr}\left(\frac{1}{\alpha} K^c \boldsymbol{e}_i (\boldsymbol{v}^c)^T - \sum_{l=1}^{C} K^l \boldsymbol{e}_i (\boldsymbol{v}^l)^T\right) \ge 0, c = y_i, \forall i, \\
&\qquad \operatorname{tr}\left(\frac{1}{\beta} K^c \boldsymbol{e}_i (\boldsymbol{v}^c)^T - \sum_{l=1}^{C} K^l \boldsymbol{e}_i (\boldsymbol{v}^l)^T\right) \le 0, \\
&\qquad\qquad \forall c \ s.t. \ c \ne y_i, \forall i.
\end{aligned} \tag{4}$$

Note that the problem (4) is a strictly convex, constrained optimization program, and any standard convex optimization toolbox can be used to obtain the optimal solution. But, a naïve method would require optimizing over n^2C variables which is prohibitively large. Instead, we show below that the dual formulation of the problem can be solved by optimizing over nC parameters only. That is, the number of parameters is linear in both n and C, and is of the same order as those of 1-vs-all SVM or 1-vs-all GP. We further propose an optimization algorithm for this problem that requires only $O(nC)$ computational operations per iteration.

By setting the gradient of the Lagrangian of (4) w.r.t K^l to be zero and using the Sherman-Morrison-Woodbury formula, we get:

$$K^l = K_0 - K_0[\boldsymbol{w}^l \ \boldsymbol{v}^l] S^l \begin{bmatrix} (\boldsymbol{v}^l)^T \\ (\boldsymbol{w}^l)^T \end{bmatrix} K_0, \tag{5}$$

where, $\lambda_i^l \geq 0, 1 \leq i \leq n, 1 \leq l \leq C$ are the dual variables,

$$\boldsymbol{w}^l = \sum_{i=1}^{n} \left(\lambda_i^{y_i} \left(1 - \frac{\delta_l(y_i)}{\alpha} \right) - \sum_{c \neq y_i} \lambda_i^c \left(1 - \frac{\delta_l(c)}{\beta} \right) \right) \boldsymbol{e}_i, \tag{6}$$

$$S^l = \left(I_{2\times 2} + \begin{bmatrix} (\boldsymbol{v}^l)^T K_0 \boldsymbol{w}^l & (\boldsymbol{v}^l)^T K_0 \boldsymbol{v}^l \\ (\boldsymbol{w}^l)^T K_0 \boldsymbol{w}^l & (\boldsymbol{w}^l)^T K_0 \boldsymbol{v}^l \end{bmatrix} \right)^{-1}, \tag{7}$$

and $\delta_a(b) = 1$ iff $a = b$, and 0 elsewhere. Thus, each of the learned kernel K^l is a two rank update to the original kernel K_0. Furthermore, given the optimal dual variable values, each of the kernel matrix K^l can be computed in the closed form using just $O(n^2)$ operations.

3.1. Optimization Algorithm

Next, we describe our proposed algorithm for optimizing (4). Similar to [5], we use the Bregman's cyclic projection method where the current solution is cyclically projected onto all the constraints till convergence. But rather than projecting the current kernels $\{K_t^1, K_t^2, \ldots, K_t^C\}$ onto a single constraint, our approach projects the current kernels onto all of the C constraints associated with the current data point $\boldsymbol{x}_i$. That is, given current kernels $\{K_t^1, K_t^2, \ldots, K_t^C\}$, the following sub-problem is solved at each step to obtain the next set of kernels $\{K_{t+1}^1, K_{t+1}^2, \ldots, K_{t+1}^C\}$:

$$\begin{aligned} &\min_{K^1 \ldots K^C} \sum_{l=1}^{C} D_{\ell d}(K^l \| K_0), \\ &\text{s.}t.\ \operatorname{tr}\left(\frac{1}{\alpha} K^c \boldsymbol{e}_i (\boldsymbol{v}^c)^T - \sum_{l=1}^{C} K^l \boldsymbol{e}_i (\boldsymbol{v}^l)^T \right) \geq 0, c = y_i, \forall i, \\ &\operatorname{tr}\left(\frac{1}{\beta} K^c \boldsymbol{e}_i (\boldsymbol{v}^c)^T - \sum_{l=1}^{C} K^l \boldsymbol{e}_i (\boldsymbol{v}^l)^T \right) \leq 0, \\ &\qquad \forall c\ s.t.\ c \neq y_i, \forall i. \end{aligned} \tag{8}$$

Note that the above given sub-problem involves constraints only for the i-th data point, i.e. solving this sub-problem implies projecting the current kernel matrices $\{K_t^1, K_t^2, \ldots, K_t^C\}$ onto the constraints for the i-th data point only. Updating the C kernels $\{K_{t+1}^1, K_{t+1}^2, \ldots, K_{t+1}^C\}$ at each step is prohibitively expensive, as it requires O(n^2C) operations. However, as shown in the previous section, by solving for the dual problem of (8) we can implicitly update the kernel matrices in O(nC) operations only. The dual problem of (8) is given by:

$$\begin{aligned} &\min_{\lambda_i^1, \ldots, \lambda_i^C} \sum_{l=1}^{C} -\log\det(S_{t+1}^l) \\ &\text{s.}t.\ \lambda_i^c \geq 0, \forall c, \end{aligned} \tag{9}$$

Algorithm 1 Probabilistic K-Nearest Neighbor (pKNN)

```
Input:  $K_0$: input $n \times n$ kernel matrix,
        $\{y_1, y_2, \ldots, y_n\}$: labels for each training point
Output: Parameters $\mathcal{S} = \{S^l, 1 \leq l \leq C\}$,
        $\mathcal{W} = \{\boldsymbol{w}^l, 1 \leq l \leq C\}$ (See (6), (7))
1: Intialize $S^l = 0, \boldsymbol{w}^l = 0, \forall l$
2: repeat
3:     Pick a data point $\boldsymbol{x}_i$
4:     Solve the dual problem (9) for $\lambda_i^c, \forall c$, and apply the
       correction term (See supplementary material)
5:     Update $\boldsymbol{w}^l, \forall l$ using (6)
6:     Update $S^l, \forall l$ using (10)
7:     Update $(\boldsymbol{v}^l)^T K_{t+1}^l \boldsymbol{w}^l$,$(\boldsymbol{w}^l)^T K_{t+1}^l \boldsymbol{w}^l$, $(\boldsymbol{v}^l)^T K_{t+1}^l \boldsymbol{v}^l$
       using $K_{t+1}^l = K_t - K_t[\boldsymbol{w}^l\ \boldsymbol{v}^l] S^l \begin{bmatrix} (\boldsymbol{v}^l)^T \\ (\boldsymbol{w}^l)^T \end{bmatrix} K_{t+1}^l$
8: until convergence
```

where,

$$S_{t+1}^l = \left(I_{2\times 2} + \begin{bmatrix} (\boldsymbol{v}^l)^T K_t^l \boldsymbol{w}^l & (\boldsymbol{v}^l)^T K_t^l \boldsymbol{v}^l \\ (\boldsymbol{w}^l)^T K_t^l \boldsymbol{w}^l & (\boldsymbol{w}^l)^T K_t^l \boldsymbol{v}^l \end{bmatrix} \right)^{-1}. \tag{10}$$

The dual problem (9) is a non-negative, non-linear convex program and can be solved using standard optimization tools. In our implementation, we use Matlab's projected gradient procedure, but for faster implementation more sophisticated tools can be used.

By optimizing (9) for the dual variables $\{\lambda_i^1, \lambda_i^2, \ldots, \lambda_i^C\}$, we can implicitly update the kernel matrices by updating the inner products $(\boldsymbol{v}^l)^T K_t^l \boldsymbol{w}^l$, $(\boldsymbol{w}^l)^T K_t^l \boldsymbol{w}^l$, and $(\boldsymbol{v}^l)^T K_t^l \boldsymbol{v}^l$ that are required for solving the dual problem (9) for the next set of constraints. We iterate through all the constraints cyclically till convergence. Since, problem (4) is a convex optimization problem, the cyclic projection algorithm is guaranteed to converge to the global optima. Algorithm 1 provides a pseudo code for our optimization algorithm. Note that, in step 7 of Algorithm 1, rather than computing each kernel K_t^l explicitly, we just update the inner products $(\boldsymbol{v}^l)^T K_{t+1}^l \boldsymbol{w}^l$, $(\boldsymbol{w}^l)^T K_{t+1}^l \boldsymbol{w}^l$, and $(\boldsymbol{v}^l)^T K_{t+1}^l \boldsymbol{v}^l$ efficiently. We can show that this step can be performed in just O(nC) computational operations. Please refer to the longer version for further details about the optimization procedure.

3.2. Classifying a Test Point

The optimization procedure described above learns a set of kernel matrices $K^1, K^2, \ldots, K^C$ over the input data points, that takes into account the provided label information. Interestingly, our method not only learns a set of kernel matrices, but also learns a set of kernel functions that can be used to compute the kernel function value between two unseen data points. Recall that the learned kernels are given by (5), or equivalently:

$$K^l(\boldsymbol{a},\boldsymbol{b}) = K_0(\boldsymbol{a},\boldsymbol{b}) - K_0(\boldsymbol{a},\cdot)[\boldsymbol{w}^l\boldsymbol{v}^l]S^l \begin{bmatrix}(\boldsymbol{v}^l)^T\\(\boldsymbol{w}^l)^T\end{bmatrix} K_0(\boldsymbol{b},\cdot)', \tag{11}$$

where S^l is given by (10), $K_0(\boldsymbol{a},\cdot) = [K_0(\boldsymbol{a},\boldsymbol{x}_1)\ \ K_0(\boldsymbol{a},\boldsymbol{x}_2)\ \ \ldots\ \ K_0(\boldsymbol{a},\boldsymbol{x}_n)]$. Thus given the initial kernel function K_0, the input points, and the learned parameter matrix S^l, kernel function K^l can be evaluated at any two points $\boldsymbol{a}$ and $\boldsymbol{b}$ in O(n) operations. Consequently, given a new data point $\boldsymbol{x}$, its similarity value $K^l(\boldsymbol{x},\cdot)$ can be computed with respect to all the training data points. Label y of the test point $\boldsymbol{x}$ can be computed using three different schemes:

1. y can be computed using our classifier defined by (2), i.e., $y = \mathrm{argmax}_c \frac{1}{n_c}\sum_{i \text{ s.t. } y_i=c} K^c(\boldsymbol{x},\boldsymbol{x}_i)$. We refer to this method as pKNN.

2. Since we compute more accurate similarity measures K^l for each class l, we can use the learned similarity measure for KNN classification. To compare an unseen point $\boldsymbol{x}$ to a point $\boldsymbol{x}_i$ in the training set, we use the learned kernel matrix K^{y_i} along with (11) to compute the distance $d(\boldsymbol{x},\boldsymbol{x}_i) = K^{y_i}(\boldsymbol{x},\boldsymbol{x}) + K^{y_i}(\boldsymbol{x}_i,\boldsymbol{x}_i) - 2K^{y_i}(\boldsymbol{x},\boldsymbol{x}_i)$. We refer to this method as pKNN-n.

3. Since existing classification methods like SVM and GP also depend on the underlying similarity measure, we can use our learned kernels with SVM (or GP) to classify new data points. Specifically, we use the learned kernel K^l for training the l-vs-all SVM classifier and use (11) to compute the decision value for a new data point to be classified. We refer to this classification method as pKNN-SVM.

Above given schemes are suitable for different problem scenarios. Scheme 2 is useful for robust similarity search, while scheme 3 is useful for fast classification, as the sparsity of the SVM classifier can be exploited. Scheme 1 is useful for robust classification.

4. Active Learning

Let $X_U = \{\boldsymbol{x}_{n+1}, \boldsymbol{x}_{n+2}, \ldots, \boldsymbol{x}_{n+m}\}$ be the pool of unlabeled data points that can be queried to find out the true label. Then at each step in active learning, an unlabeled data point is selected from the set X_U to label and is incorporated into the training set to update the classification model. Thus, the goal is to select the data point that strengthens the existing classification model the most in terms of its discriminative capabilities.

As described in the previous section, our method learns a set of kernels $K^l, \forall l$, which are then used to predict the label of a new data point according to following criteria: $y = \mathrm{argmax}_c \frac{1}{n_c}\sum_{i \text{ s.t. } y_i=c} K^c(\boldsymbol{x},\boldsymbol{x}_i)$. Now, if the learned kernels K^c are all non-negative then $p_c(\boldsymbol{x}) = \frac{\frac{1}{n_c}\sum_{i \text{ s.t. } y_i=c} K^c(\boldsymbol{x},\boldsymbol{x}_i)}{\sum_{t=1}^{C}\frac{1}{n_t}\sum_{i \text{ s.t. } y_i=t} K^t(\boldsymbol{x},\boldsymbol{x}_i)}$ is a valid probability distribution over the class labels for the given data point $\boldsymbol{x}$.

As in many existing active learning methods [13, 18], we assume that an uncertain data point (in terms of classification) should strengthen the discriminative capability of the existing classifier. A standard active learning approach is to use the Shannon entropy—$H(\boldsymbol{x}) = -\sum_c p_c \log p_c$—as the measure of uncertainty in classification. Note that the learned kernels K^c are guaranteed to be positive semidefinite, but there is no guarantee that all individual entries will be positive. In such cases it is non-trivial to use Shannon entropy as an uncertainty measure. However, we can use mode of the probability density for $\boldsymbol{x}$, i.e. $\max_c p_c(\boldsymbol{x})$, as the measure of certainty. Note that a higher mode of probability density $p_c(x)$ corresponds to more certainty in classification. Consequently we can select the points according to the criterion:

$$\underset{\boldsymbol{x}}{\mathrm{argmin}}\, \max_c p_c(\boldsymbol{x})$$

This results in selecting data points with minimum amount of certainty over class labels. Note that this scheme can be easily implemented even if K^c is negative for some point.

After selecting a set of highly uncertain data points using the above given active learning method, we re-train our classifier to include the selected data point and its label. This involves solving the optimization problem (4) again with a new set of constraints. By using the current optimal dual variables to initialize the dual variables for the new optimization problem, the new optimization problem can be solved efficiently with a small number of iterations. Furthermore, online updates can be derived for the kernels K^c, so that the addition of a new data point to the training set can be performed using a small number of operations.

5. Results

In this section, we present empirical results to demonstrate 1) the effectiveness of the pKNN framework on large multi-class problems, 2) how active learning can guide the learning procedure to select critical examples to be labeled, and 3) the ability of the proposed framework to learn a good kernel function.

The empirical evaluation is performed on two standard benchmarks: Caltech101 and UCI datasets. We compare our active learning method with the baseline random sampling method, and two state-of-the-art methods: 1) an active learning method based on GP (GP+AL) that uses the posterior distribution of the prediction to select the most uncertain data point [13], 2) an SVM based scheme (SVM+AL) that tries to minimize approximate version space [15]. Both of these heuristics are based on the one-vs-all formulation of binary classifiers to handle multi-class problems.

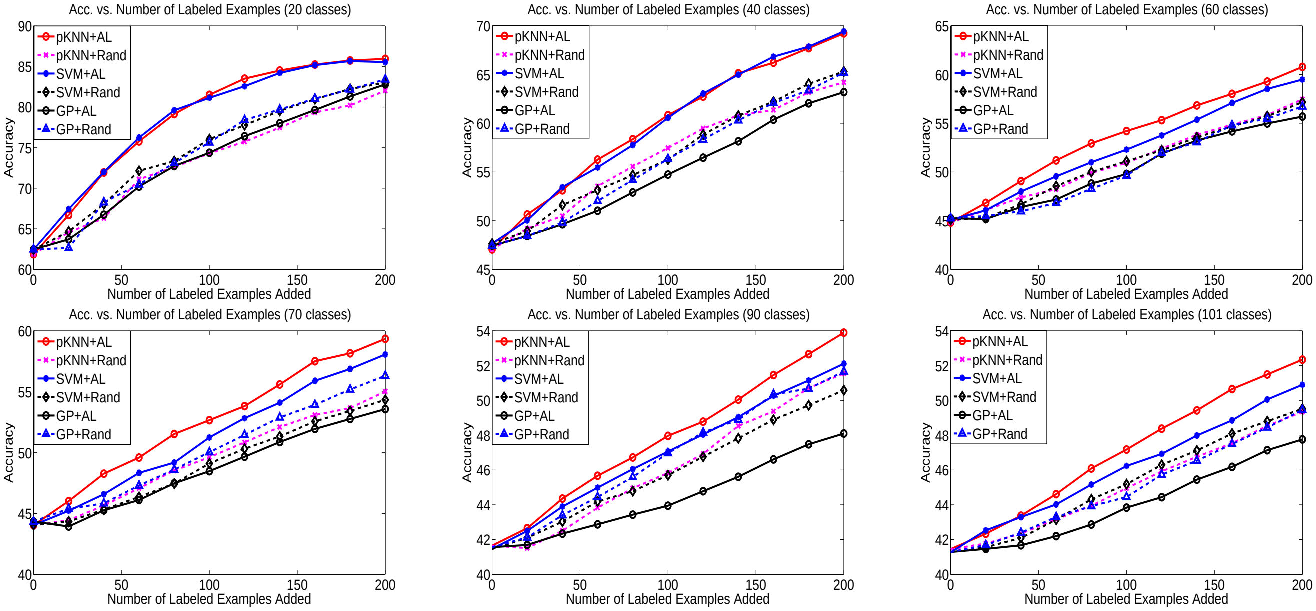

Figure 2. Comparison of classification accuracy obtained by various methods on three different subsets of Caltech101 containing 20, 40, 60, 70, 90 and 101 classes. Each subplot shows variation in accuracy (averaged over 10 runs) as the actively learned examples are added. Note that pKNN+AL is significantly more accurate than GP+AL and SVM+AL, especially for subsets containing a large number of classes.

For the experiments, we consider a randomly initialized training set, a test set and a query set containing unlabeled examples (X_U). The active learning scheme queries for the labels of examples contained in the query set X_U only, while the recognition performance is reported on the test set. For every method, we report the mean classification accuracy per class (averaged over 10 runs). Similar to [13], we use $\sigma = 10^{-5}$ for noise model variance for the Gaussian Process models and fix constraint penalty $C = 10^5$ for the SVMs. For the pKNN method, we choose the parameters α and β to be $\frac{2}{C}$ and $\frac{0.99}{C}$, where C is the number of classes. Slightly better accuracies can be expected with cross-validation.

Caltech101: Caltech101 is a benchmark dataset for object recognition, where given a test image, the goal is to predict the correct category out of the 101 classes. Our experiments use a pool of 30 images per class of which 15 randomly chosen images per class forms the test set. The initial training set is seeded randomly by selecting 2 training examples per class and the remaining images form the query set X_U. For each round of active learning, 20 images selected from the query set X_U are labeled and added to the training set.

For the proposed pKNN method (Algorithm 1), the initial kernel K_0 is set to be the average of 4 well-known kernels. Specifically, we use two kernels based on the geometric blur descriptors [2] and two kernels—the Pyramid Match Kernel (PMK) and the Spatial PMK—based on the SIFT features [9, 3][2]. We use the same kernel K_0 to train both SVM and GP.

First, we evaluate accuracy of various active learning schemes with the number of classes. We randomly select a subset of classes from the Caltech101 dataset and consider all the images from those classes to generate a classification subproblem. Figure 2 shows results on randomly chosen subproblems with 20, 40, 60, 70, 90, and 101 classes. It is clear from the Figure 2 that the proposed pKNN+AL method outperforms all the other schemes including GP+AL and SVM+AL. Also, pKNN+AL is significantly better than the random sampling (pKNN+Rand) for all the different subproblems including the full 101 class case. However, that is not the case with GP+AL, for which the active learning versions provide little advantage for large number of classes. Figure 1 summarizes the gain achieved by various active learning methods over random selection. As the number of classes increases, pKNN+AL gains up to 2.9% of accuracy and is significantly more accurate than other methods.

Next, to demonstrate the effectiveness of the pKNN classification algorithm, we compare the accuracy obtained by pKNN+AL for the Caltech101 dataset with that of the state-of-the-art methods. Figure 3 (left plot) shows that except for GP based Multi-kernel method [13], our method outperforms the existing methods by a significant margin. Also, the accuracies for pKNN+AL are generated using the active learning scheme, that is at each step we add 101 data points. In contrast, the GP-Multi-Kernel method assumes knowledge of all the labels to form a balanced training set consisting of an equal number of examples per class.

[2] We obtained each kernel from the respective authors.

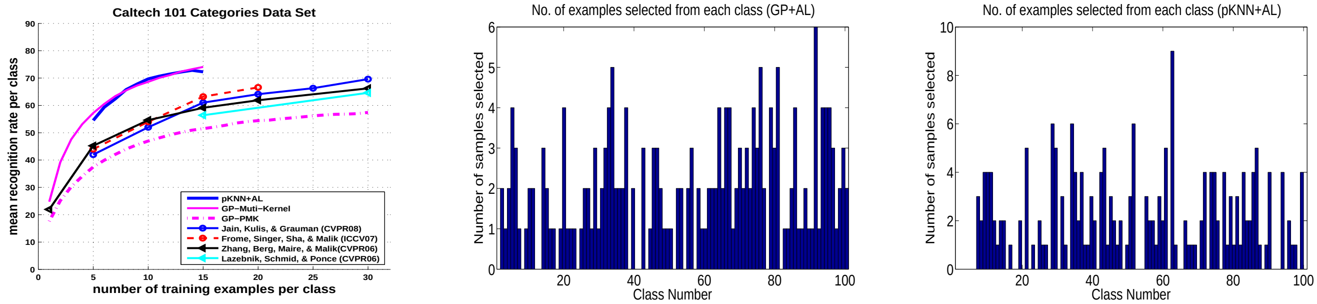

Figure 3. **Left**: Caltech 101 results. Comparison of classification accuracy with the state-of-the-art methods. Our classification method is competitive with the existing methods. **Middle**: Number of examples added per class by the GP+AL method. Clearly, the GP+AL samples all the classes fairly uniformly. **Right**:Number of examples added per class by the pKNN+AL method. Clearly, the proposed pKNN method samples classes unevenly, depending on their difficulty.

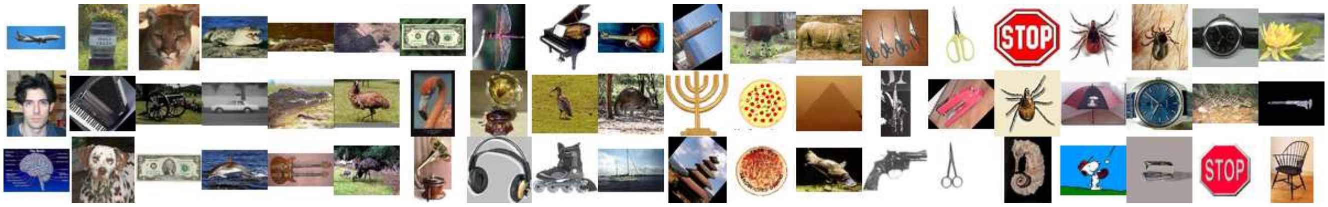

Figure 4. Images from the Caltech101 dataset that are actively selected by the GP+AL method. Each row of the plot shows 20 images selected in a single active learning round. Due to the 1-vs-all formulation, GP+AL is restricted to selecting one image from each binary subproblem and thus selects images from many easy to predict categories, e.g. *Face*, *dollar bill*, *car side* etc.

Discussion: Next, we analyze and highlight the reasons behind significantly higher accuracy achieved by the proposed active learning method compared to the other methods. Figure 3 (middle and rightmost plot) compares the frequency distribution of examples selected per class by pKNN+AL to that of GP+AL. Note that the distribution of samples selected by pKNN+AL is much more skewed than the ones selected by GP+AL. This suggests that sampling uniformly across all the classes is not helpful as there might be *easy* classes and sampling from those classes amounts to wastage of supervision. By design one-vs-all extensions of the binary methods to multi-class problems consist of *independent* binary subproblems; consequently, it is not clear how such designs can compare the *difficulty* of the sub-problems across all the classes and thus, might end up selecting samples that correspond to easy classes as well. This observation is illustrated by Figures 4 and 5, that show the different images selected by pKNN+AL and GP+AL during 3 different rounds of active learning. We note that pKNN+AL tends to select more images from a hard class for which the current classifier requires more training data (for example wild cat). But, GP+AL ends up selecting a number of images from classes that might be easy, e.g., *Face*, *dollar bill* etc.

Next, we analyze the classes that were maximally sampled by the pKNN+AL method. Figure 6 shows the top five classes sampled by pKNN+AL. Note that these classes are particularly hard as they contain images that are diverse in terms of background, or size and shape of the object–e.g., *starfish*, *mayfly*, *dolphin* category –or are very similar to some other category–e.g., the *ibis* category which is similar to the *flamingo* and *flamingo head* category, the *lotus* category which is similar to the *sunflower* and *water lily*, and the *crocodile* category which is similar to the *crocodile body* category.

Table 1. Accuracy obtained by standard metric learning methods (LMNN [20], ITML [5]) against the accuracy obtained by the proposed pKNN, and pKNN-n. Note that both pKNN and pKNN-n performs comparable to the state-of-the-art methods.

Dataset\ Method	LMNN	ITML	pKNN	pKNN-n
Ionosphere	**86.1**	85.5	85.7	81.1
Scale	88.4	90.4	88.9	**92.0**
Iris	95.1	95.7	94.3	**97.1**
Soybean	**92.3**	92.1	91.9	89.3

UCI Datasets: A key feature of our pKNN classification framework is that along with training a classifier, it also learns an improved similarity measure. Thus, our method can also be used for similarity search in the context of classification. We demonstrate this aspect of the algorithm by applying it for the nearest neighbor search problem on a variety of standard UCI datasets. We compare our method with existing distance learning methods. We evaluate the results for the nearest neighbor search problem using 1-nearest neighbor classification with 2-fold cross validation. Table 1 compares accuracies obtained by various metric learning methods on 4 standard UCI datasets. Note that

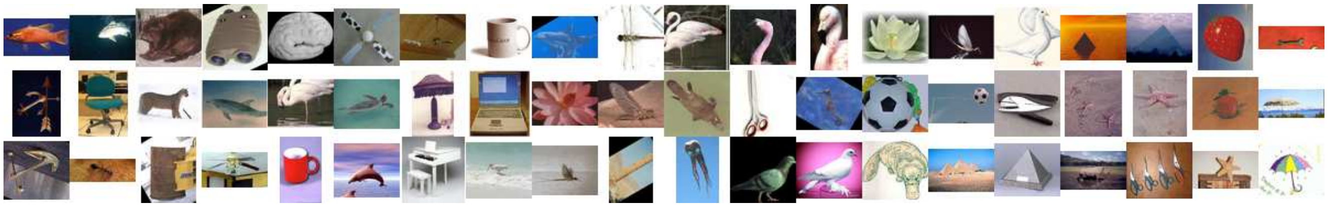

Figure 5. Images from the Caltech101 dataset that are actively selected by the pKNN+AL method. Each row of the plot shows 20 images selected in a single active learning round. Note that, for difficult categories like the *flamingo* category, multiple images are selected in a single round itself. In contrast, GP+AL typically selects 1 image per category only (See Figure 4).

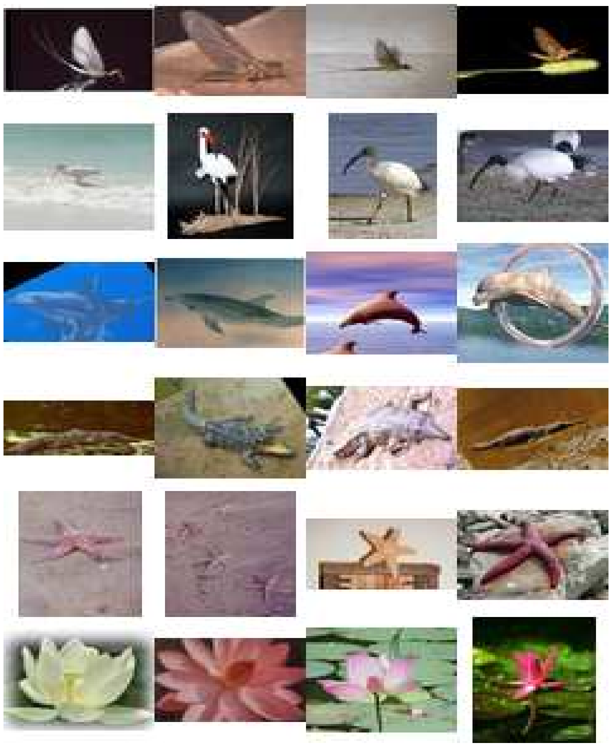

Figure 6. Figure shows the selected images from the six most sampled classes by the pKNN+AL method.

our method involves learning fewest number of parameters of all the existing metric learning algorithms; our method requires O(nC) parameters while most of the existing metric learning methods require O(n^2) parameters. Still, our method is able to achieve competitive accuracy for a variety of classification problems (Table 1).

6. Conclusions

We have presented a probabilistic classification framework that we further use for active learning in multi-class scenarios along with learning an accurate similarity measure. Based on the proposed classification framework, we formulated a kernel learning algorithm that learns accurate kernel functions and is efficient. The probabilistic nature of the formulation allowed us to seamlessly incorporate an active learning strategy into our framework. We demonstrated empirically that for large multi-class problems, our proposed active learning method (pKNN+AL) is significantly more accurate than the existing active learning methods. Future directions for this research include non-myopic active learning, exploration of other active learning heuristics and application of this method to even larger problems.

References

[1] A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis Metric from Equivalence Constraints. *JMLR*, 6:937–965, 2005.

[2] A. C. Berg and J. Malik. Geometric blur for template matching. In *CVPR*, 2001.

[3] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *CIVR*, 2007.

[4] E. Y. Chang, S. Tong, K. Goh, and C. Chang. Support vector machine concept-dependent active learning for image retrieval. *IEEE Transactions on Multimedia*, 2005.

[5] J. Davis, B. Kulis, P. Jain, S. Sra, and I. Dhillon. Information-Theoretic Metric Learning. In *ICML*, 2007.

[6] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. John Wiley and Sons, 2001.

[7] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3), 1997.

[8] A. Frome, Y. Singer, and J. Malik. Image retrieval and classification using local distance functions. In *NIPS*, 2007.

[9] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. In *ICCV*, 2005.

[10] J. He and J. Carbonell. Nearest neighbor based active learning for rare category detection. In *NIPS*, 2007.

[11] S. C. H. Hoi and R. Jin. Active kernel learning. In *ICML*, 2008.

[12] A. Holub, P. Perona, and M. Burl. Entropy-based active learning for object recognition. In *CVPR workshop on Online Learning for Classification*, 2008.

[13] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Gaussian processes for object categorization. In *IJCV (in submission)*, 2008.

[14] N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse Gaussian Process method: Informative vector machines. *NIPS*, 2002.

[15] X. Li, L. Wang, and E. Sung. Multilabel svm active learning for image classification. In *ICIP*, 2004.

[16] M. Lindenbaum, S. Markovitch, and D. Rusakov. Selective sampling for Nearest Neighbor classifiers. *Machine Learning*, 2004.

[17] D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4), 1992.

[18] G. Qi, X. Hua, Y. Rui, J. Tang, and H. Zhang. Two-dimensional active learning for image classification. In *CVPR*, 2004.

[19] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In *ICML*, 2000.

[20] K. Weinberger, J. Blitzer, and L. Saul. Distance Metric Learning for Large Margin Nearest Neighbor Classification. In *NIPS*, 2006.

[21] R. Yan, J. Yang, and A. Hauptmann. Automatically labeling video data using multi-class active learning. In *ICCV*, 2003.

[22] L. Yang, R. Jin, and R. Sukthankar. Bayesian active distance metric learning. In *UAI*, 2007.