Efficient Learning on Point Sets

Liang Xiong, Barnabás Póczos, and Jeff Schneider
School of Computer Science
Carnegie Mellon University
Email: {lxiong,bapoczos,jeff.schneider}@cs.cmu.edu

***Abstract*—Recently several methods have been proposed to learn from data that are represented as sets of multidimensional vectors. Such algorithms usually suffer from the high demand of computational resources, making them impractical on large-scale problems. We propose to solve this problem by *condensing i.e.* reducing the sizes of the sets while maintaining the learning performance. Three methods are examined and evaluated with a wide spectrum of set learning algorithms on several large-scale image data sets. We discover that k-Means can successfully achieve the goal of condensing. In many cases, k-Means condensing can improve the algorithms' speed, space requirements, and surprisingly, learning performances simultaneously.**

I. Introduction

In many problems the object of interest can be represented by a set of multidimensional vectors. For instance, in computer vision an image is often treated as a set of patches [1], each being described by a feature vector such as SIFT [2]. A video sequence is an ordered set of images. In a social network, a community is a set of people. In text processing and retrieval, though it is prevalent to consider a document as a bag of words, we can also think of it as a set of sections/paragraphs to cope with its structure. It is important to devise algorithms that can effectively learn from these data.

A convenient, and indeed traditionally used, way to deal with point sets is to extract useful signals and construct a feature vector for each set. After reducing the sets to vectors, standard learning techniques can be applied. Nevertheless, the conversion process is often problem-specific, sometimes complicated, and involves much human effort. More importantly, this set-to-vector reduction can cause loss of information.

On the other hand, the development of algorithms that handle sets directly has been largely left behind. One major disadvantage of such algorithms is their high computational cost compared to those that operate on vectors. For example, in computer vision, by quantizing the local features and aggregating them by the "bag of visual words" method [1], typically a 256×256 image can be characterized by one $1,000$-dimensional vector, amounting to just 1KB of data. Yet, to represent it as a set of SIFT features, we typically need about $1,500$ 128-dimensional vectors, amounting to about 190KB of data. The further computation needed to process such sets is likely to be also orders of magnitudes larger.

Despite the difficulties, recently several methods have been proposed to deal with point sets directly. They learn from the sets without the set-to-vector reduction, so that the researchers do not have to design the feature vector for a set, and the loss of information caused by the reduction can be avoided. [3] design a novel kernel between point sets based on consistent estimators of divergences between distributions, and achieved the state-of-the-art classification performance on a couple of datasets. [4] proposed an extremely simple classifier for point sets based on image-to-class matching, and showed that it could compete with classifiers based on very sophisticated set features. These successes demonstrate the advantage of learning directly from point sets over the reduction approach.

Early set learning algorithms (specifically the set similarities), such as the *Hausdorff* distance and the *mean map kernels* by [5], rely on the similarities between every pair of points and are thus computationally expensive. Recent improvements such as [3], [4], [6] gained efficiency by designing algorithms based on the points' local neighborhoods, which can be obtained via efficient search algorithms. [4], [6] proposed a new classification paradigm by comparing images to classes and significantly accelerated the prediction. Details of these methods are described in Section II. Nevertheless, they still demand much time and storage space, and are not suitable for large-scale problems and less likely to be adopted by practitioners.

In this paper, we aim to further improve the computational efficiency of set learning algorithms. In most algorithms, the cost to train, store, and apply the model is determined by the sizes/cardinalities of the sets. Therefore, our approach is to directly attack the crux of the problem by reducing the size of sets while maintaining the learning performance, in an unsupervised way. We call such an operation *condensing*.

To achieve this goal, we analyze and evaluate three possible ways of decrease the size of a set: random sampling, uniform covering, and distribution approximation using k-Means. These three methods are chosen because they are easy to implement and run in large data scenarios. Our discovery is that distribution approximation via k-Means is the only method that can successfully achieve the goal of condensing.

In our experiments, we apply the k-Means condensing as a pre-processing step to various point-set learning methods on image classification tasks, and find that the performance is surprisingly good and consistent. In most problems, we do not have to make a speed-accuracy tradeoff; condensing can actually *improve both speed and accuracy simultaneously*. In addition, this condensing step can be easily implemented and parallelized for large-scale problems. We believe this discovery is useful to practitioners that have large-scale point-set data.

The rest of this paper is organized as follows. In Section II we introduce the notation and common learning algorithms on point sets to provide a context to this study. Section II-E briefly reviews related work. Section III describes in detail the condensing methods we are examining. In Section IV, we thoroughly evaluate the performance of different methods

on different data sets and discuss our findings. Finally, we conclude the paper in Section VI.

II. Learning on Point Sets

Most learning tasks can easily be accomplished if we know the similarities between the point sets. Many set learning algorithms assume that a set has an unknown underlying distribution, and the points in the set are *i.i.d.* samples from that distribution. Then the similarity measures between point sets can be designed based on the divergences between their underlying distributions. For example, [7], [5] proposed the *mean map* set kernel to test if two point sets have the same underlying distribution. The same technique has been known and used by the computer vision community as the *summation kernel* [8]. [4] uses a simplified *kernel density estimator* to estimate the divergence between a set and the classes, and assign the set to the class with the most similar distribution. [9], [3] use a consistent nonparametric estimator to get the divergences between the point sets and use these dissimilarities to construct Gaussian kernels so that SVM can be used for classification.

From the computational perspective, many of the set similarity measures can be considered as aggregations of the similarities between the individual points of the sets. We will discuss in more details how these similarities are measured and aggregated in Section II-B. The key point is that these point-level pairwise comparisons make the speed of the algorithms crucially dependent on the sizes of the sets. This is why reducing the sets' sizes by condensing would greatly improve their computational efficiency. Generative methods such as [10], [11] have also been developed to model point sets. Condensing the sets will also benefit these methods. They, however, are mainly designed for anomaly detection and it is hard to evaluate them quantitatively. In this paper we will focus on the similarity based approaches and their applications in set classification problems.

Multiple-instance learning (MIL) [12] is a closely related topic; indeed some set kernels have been used in MIL *e.g.* the mean map kernels [13]. In MIL, the training data are also sets of points, and the label of a set is determined by the labels of its *points*; a set is positive if it contains at least one positive point, otherwise it is negative. After training, the model is then usually used to classify *points*. We can see that MIL, even though trained on point sets, focuses on the behaviors of a few points. On the contrary, in this paper we shall concentrate on sets that are characterized by the holistic behavior of all the points in them.

A. Notation

We consider a data set that consists of M point sets $\{G_m\}_{m=1,\dots,M}$, and each point set G_m is a set of d-dimensional vectors, $G_m = \{x_{mn}\}_{n=1,\dots,N_m}$, $x_{mn} \in \mathbb{R}^d$. Note that the numbers of points in each set can be different. We also assume that each G_m has an unknown underlying distribution f_m, and the points $\{x_{mn}\}$ are *i.i.d.* samples from f_m. For instance, in the context of image classification, each G_m is an image, and vector x_{mn} is the feature of the nth patch in this image. In text processing, G_m is a document and x_{mn} could be the bag-of-words (BoW) feature of the nth section.

Nearest neighbors (NN) are frequently used in set learning algorithms. We use $\mathrm{NN}_G(x)$ to denote the NN of x in point set G. If x is in G then it excludes itself during the search. Ties, if any, are broken arbitrarily.

B. Set Similarities

a) Set Kernels: [7], [5] proposed the following kernel (similarity) for two sets of points G_1 and G_2:

$$K(G_1, G_2) = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} k(x_{1i}, x_{2j}) \tag{1}$$

where $k(x, y)$ is a kernel between points x and y. One particularly popular example is the Gaussian kernel $k(x, y) = \exp(-\left\|x - y\right\|_2^2 / \sigma^2)$. The underlying principle for this kernel is that if the point-level kernel $k(x, \cdot)$ induces a feature map $\phi(x)$ for x in a *reproducing kernel Hilbert space* (RKHS), then this corresponding set-level kernel $K(G, \cdot)$ will induce the feature map $\Phi(G) = \frac{1}{N} \sum_{n=1}^{N} \phi(x_n)$ for G in the same RKHS. Since $\Phi(G)$ is the empirical mean of the mapped features of the points, K is called the *mean map kernel* (MMK).

We can see that MMK essentially is a way of aggregating the point-level similarities between two sets. It possesses many nice theoretical properties such as positive definiteness [5], [14]. The same idea has also been used by computer vision researchers [8]. Yet since it averages the similarities between every pair of points, MMK will slow down quadratically w.r.t. the sets' sizes, and become infeasible for even moderately sized problems.

[15] used a variation of MMK that only considers the most similar pairs of points:

$$K(G_1, G_2) = \frac{1}{N_1} \sum_{i=1}^{N_1} k(x_{1i}, \mathrm{NN}_{G_2}(x_{1i})) + \frac{1}{N_2} \sum_{j=1}^{N_2} k(x_{2j}, \mathrm{NN}_{G_1}(x_{2j})). \tag{2}$$

Computationally it improves MMK by only using the points' NNs instead of dealing with all pairs. Unfortunately, this kernel is no longer a proper *Mercer kernel*, but it still serves well as a similarity measure. Other related methods include [16], which uses multi-resolution histograms to approximates MMK, and [17], which constructs explicit approximate feature maps for the MMK so that linear classifiers can be used to achieve faster computation.

b) Set Divergences: Another class of dissimilarity measures is defined based on the statistical divergences between two distributions. In [18], [9], [3], the authors provided consistent NN based estimators for various divergences including the *Kullback-Leibler* (KL), Rényi, and the L_2 divergences. They have been successfully applied to image classification problems when used in SVMs [3].

For example, the KL divergence between two point sets can be estimated by [18]

$$\hat{\mathrm{KL}}(G_1 || G_2) = \frac{d}{N_1} \sum_{i=1}^{N_1} \ln \frac{\left\|x_{1i} - \mathrm{NN}_{G_2}(x_{1i})\right\|_2}{\left\|x_{1i} - \mathrm{NN}_{G_1}(x_{1i})\right\|_2} + \ln \frac{N_2}{N_1 - 1} \tag{3}$$

where d is the dimensionality of x_{1i}. It was proved that these estimators are consistent, *i.e.* under regularity conditions, (3) converges to $\mathrm{KL}(f_1||f_2)$, when the sample sizes N_1 and N_2 approach infinity.

[4] proposed an alternative estimate of the KL divergences. Consider *kernel density estimation* at point x_{1i} given the points in G_2 with a Gaussian kernel of width σ:

$$\hat{f}(x_{1i}; G_2) \propto \frac{1}{\sigma N_2} \sum_{j=1}^{N_2} \exp\left(- \left\| \frac{x_{1i} - x_{2j}}{\sigma} \right\|_2^2 \right). \quad (4)$$

This estimator is computationally demanding since we have to consider every pair of points. In [4], the authors let the width σ be small enough, so that the summation will be dominated by its largest term, and the estimator becomes

$$\ln \hat{f}(x_{1i}; G_2) \approx - \left\| x_{1i} - \mathrm{NN}_{G_2}(x_{1i}) \right\|_2^2 / \sigma^2 - \ln N_2 \sigma + \text{const}, \quad (5)$$

which again is based on NNs. The corresponding estimated KL divergence is

$$\hat{\mathrm{KL}}(G_1||G_2) \propto \sum_{i=1}^{N_1} \left\| x_{1i} - \mathrm{NN}_{G_2}(x_{1i}) \right\|_2^2 - \sum_{i=1}^{N_1} \left\| x_{1i} - \mathrm{NN}_{G_1}(x_{1i}) \right\|_2^2 \quad (6)$$

up to a constant. Note the resemblance between (6) and (3). Unlike (3), this estimator is not consistent even with infinite points, but in practice it can still produce good results.

These set similarities follow a similar pattern. They generally use nearest neighbor statistics and can be efficient in low dimensions where various search trees work well. However, in high dimensions (as few as 10) the computational speed of the estimators will deteriorate rapidly.

C. Set Classification Schemes

Having the similarities between the sets, we can easily apply SVM, KNN, or other techniques to accomplish tasks like classification, ranking, and clustering as in *e.g.* [9], [3], [14]. For example, the KL divergences estimated using (3) can be used construct Gaussian kernels *e.g.*

$$K_{\mathrm{KL}}(G_1||G_2) = \exp\left(-\hat{\mathrm{KL}}(G_1||G_2)/\sigma^2 \right), \quad (7)$$

where σ is the width of the kernel. Due to the properties of the KL divergence, this kernel is neither symmetric nor positive definite. Therefore [3] proposed to approximate this "pseudo" kernel matrix by the closest positive semi-definite matrix, and then use this approximation as the input to SVMs.

The drawback of this set-vs-set approach is that the training cost grows quadratically and the prediction cost grows linearly with the number of sets for training. Considering that comparing a pair of sets already requires significant work, this scheme quickly becomes infeasible in larger problems. To solve this problem, [4] proposed a set-vs-class paradigm for set classification. Assume that there are C classes indexed by c, and class c is represented by the merged set

$$H_c = \bigcup_{G_m \in c} G_m,$$

which contains all the points in the sets that belong to class c. The classification rule is to assign a test set G to the class with the smallest divergence

$$c(G) = \arg\min_c \hat{\mathrm{KL}}(G||H_c), \quad (8)$$

where the KL-divergence is estimated by (6). The assumption under this scheme is that all the sets in the same class c have the same distribution f_c, from which H_c are the *i.i.d.* samples. Though it is debatable if this assumption is valid in real-world problems, the resulting algorithm called the *Naive Bayes nearest neighbor* (NBNN) [4] is extremely simple and performs well empirically.

More importantly, NBNN discards the training phase and makes the prediction cost of (8) only proportional to the number of classes as $O(NC\zeta_{H_c})$, where N is the size of the test set and ζ_{H_c} denotes the complexity of one NN search in H_c. *Local NBNN* (LNBNN) [6] further improves NBNN by merging all classes into one large set $H = \bigcup_c H_c$ and decreases the complexity to $O(N\zeta_H)$. As a result, LNBNN can classify many classes very efficiently. Interested readers are encouraged to see [6] for more details. Yet in large, high-dimensional problems, $\{H_c\}$ and H can easily become huge, making ζ_{H_c} and ζ_H unaffordable. Another problem is that H might become so large that it cannot be held in memory, making even the approximate NN search methods infeasible.

D. Computational Issues

Looking at the above algorithms, we realize that one would face severe challenges due to both computational time and space demand if one were to apply these algorithms to large-scale problems. These computational requirements are determined by the sizes of the sets. If we could reduce them by half, the space complexity would drop by half, and ideally the running time would drop to a quarter. Therefore, our approach to make the "learning on sets" problem more efficient is to directly reduce the size of the sets, *condensing* the information into a much smaller amount of data while preserving the learning performance.

Even though NBNN and LNBNN have successfully made the complexity linear w.r.t. the number of sets, they create huge point sets $\{H_c\}$ for each class that are difficult to store and use. To put it in context, suppose we have $1,000$ images for training. Typically each image is characterized by around $2,000$ densely sampled 128-dimensional single precision SIFT vectors. Using Local NBNN, this relatively small training set would result in a model consisting of 2×10^6 points and 1GB of data. Additionally, in high dimensions searching for nearest neighbors in such a large set can be very slow.

In practice one could also consider using the approximate NN search algorithms. One popular method, for example, is the randomized KDTree [19] algorithm implemented in the *FLANN*[1] [20] package. It checks in multiple KDTrees for a fixed number of leaf nodes, which is the budget set by the user, and then returns the best results it can find. Its exact approximation accuracy and time complexity is unclear and dependent on the data. When using such approximate methods, it is rare to achieve quadratic improvement of speed

[1] http://www.cs.ubc.ca/~mariusm/index.php/FLANN

by reducing the size, yet condensing can still greatly help the construction, use, and storage of models. When the data is too large to fit into memory, then even approximate search is infeasible, but condensing can make it possible.

E. Related Work

As far as we know, there is little previous work that thoroughly studies how to reduce the sets' sizes in set-based learning. One reason might be that set-based learning itself is rather new. Random sampling is a common practice. In [21], the authors used an *asymmetric approach* for image classification. The reasoning is that we can find good matching patches between two similar images as long as one of them is densely sampled. This approach is actually subsampling the set on one side of the similarity/divergence computation. It can speed up the computation but will also deteriorate the classification performance. In the following sections we show that by using more carefully chosen condensing methods, we can improve both the speed and the accuracy of their algorithm.

In point-based learning, condensing point sets is embodied in the *prototype selection* (PS) problem [22]. In prototype selection, we are given one training set of labeled points, and the goal is to reduce the size of the training set while maximizing the performance of the resulting classifier. We can see that PS is very different from condensing in the context of set-based learning. PS handles *one* set and focuses on the behavior of individual points. In contrast, set-based learning handles multiple sets and focuses on the set as a whole. PS methods usually focus on discriminating the points instead of preserving the statistical properties of the sets that are needed in set-based learning. Several underlying techniques are shared between PS and condensing in set-based learning. Below we will comprehensively evaluate the techniques that are suitable for set-based learning. Our contribution is the discovery of a good condensing method for set-based learning algorithms that can *improve their speed, space requirement, and accuracy all at the same time*.

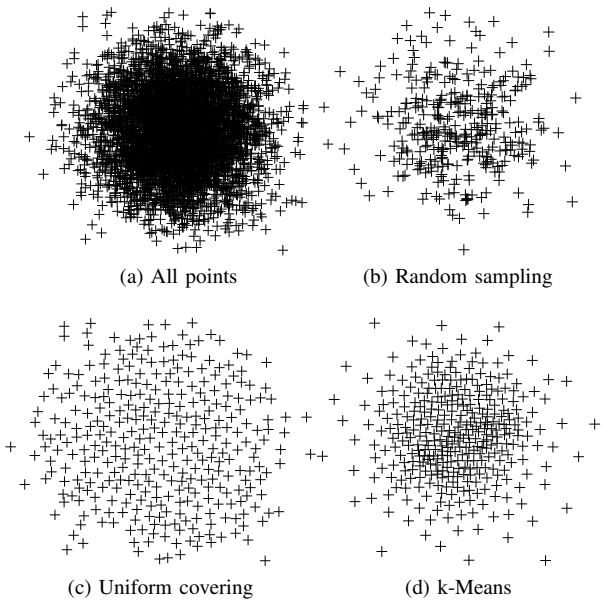
III. Condensing Methods

We examine three potential ways of condensing a point set G of size N to a smaller point set $\tilde{G}$ of size K, so that the properties of G that are useful to set learning are preserved in $\tilde{G}$. These three methods are selected for the following reasons:

- They have sound theoretical bases.
- They are easy to use in large-scale problems.
- They are universal *i.e.* not coupled with the subsequent learning algorithms, and thus widely applicable.

A. Random Sampling

Statistically, random subsampling can create smaller point sets $\tilde{G}$ with the same statistical properties as the original set G. This is the common method used to make trade-offs between speed and accuracy [23], and it is essentially the *asymmetric approach* used in [21]. It is easy to implement with virtually no computational cost. However, when given only one chance to subsample, the result is random with possibly high variance, and might lead to poor results. Figure 1b shows an example in which the subsampled set is far from an ideal representation of the original set. We will use this method as our baseline.



(a) All points (b) Random sampling (c) Uniform covering (d) k-Means

Figure 1: Condensing results of a 2-D standard Gaussian point set using different condensers.

B. Uniform Covering

We can also control the condensing quality at the point level. Since many learning algorithms are based on NNs, we require for any point x, the change of distance to its NN in G is bounded individually after condensing. Formally, we are looking for a $\tilde{G}$ such that

$$\left| \|x - \mathrm{NN}_{\tilde{G}}(x)\|_2 - \|x - \mathrm{NN}_G(x)\|_2 \right| \leq r, \forall x,$$

where r is the error threshold.

To find such a $\tilde{G}$ with minimal size is NP-hard [24]. [25] proposed *kernel vector quantization* which used the l_1 relaxation and provided an approximate solution using *linear programming*, but it can be too slow when condensing large sets. Instead, we adopt the following simple solution. Starting from an empty $\tilde{G}$, a) pick a point x from G and move x into $\tilde{G}$; b) Remove points in G whose distance to x is less than r; Repeat a) and b) until G is empty. The points in $\tilde{G}$ form the centers of radius-r spheres that uniformly cover the support of G's underlying distribution. Therefore, we call this *uniform covering* condensing. An illustration is given in Figure 1c.

To minimize the size $\tilde{G}$, a heuristic is to pick points in denser regions early so that more points can be removed. [26] implemented such a heuristic. It runs *Mean Shift* [27] to find a local peak of the density, and then picks a point from the peak. The complexity is $O(N^2)$ for a suitable r.

Though it seems well motivated, this approach is flawed. First, we cannot control the number of points to cover the support. To find a proper r we would need trial runs and tuning. Secondly, it only captures the support of the underlying distribution and ignores the actual density levels. The most

severe problem, however, is the curse of dimensionality. In high dimensions, the neighborhood of a point becomes so large that we need a very large r to effectively trim the size down. Sometimes r is not much smaller than the diameter of the support, making the bound useless. We shall demonstrate this effect in the experiments.

C. k-Means

As a clustering algorithm, k-Means can also serve our condensing purposes well. We can run k-Means on the set G, and then use the set of cluster centers as $\tilde{G}$. Recall that k-Means minimizes the following objective

$$\tilde{G} = \{\tilde{x}_k\}_{k=1,\dots,K} = \arg\min_{\tilde{G}} \sum_{i=1}^{N} \|x - \mathrm{NN}_{\tilde{G}}(x)\|_2^2 . \quad (9)$$

In other words, it is trying to find a point set $\tilde{G}$ to best reconstruct G using the nearest neighbor method given the budget of K points.

We can prove that the k-Means condensing is maximizing the performance of NBNN. Recall that NBNN assumes that sets from the same class c share the same distribution f_c that is characterized by the merged class representation H_c, as described in Section II-C. Then we assign a test set G to the class with the smallest KL-divergence according to (8). Hence, to find a condensed set $\tilde{H}_c$ for H_c that can maximize the classification performance of class c, we should minimize $\mathrm{KL}(G||\tilde{H}_c)$ for any G with the distribution f_c. Since f_c is characterized by H_c, the objective should be

$$\tilde{H}_c = \arg\min_{\tilde{H}_c} \mathrm{KL}(H_c||\tilde{H}_c). \quad (10)$$

We can see that $\tilde{H}_c$ should approximate the distribution of H_c. Further, by plugging (6) – the KL-divergence estimator used by NBNN – into (10), we see that (10) is indeed equivalent to the k-Means objective (9). In this sense, k-Means is the *ideal* condenser for NBNN. In the experiments we show that it is also generally good for other set learning algorithms.

Another advantage of k-Means is its efficiency. Being an extensively used and studied method, k-Means can be implemented with highly efficient algorithms for even massive data sets (*e.g.* [28], [29]). In practice, we also found that the exact solutions or even the local minima of k-Means is not required. Usually running k-Means for only tens of iterations is enough to achieve good condensing performance.

To sum up, k-Means provides a well-justified and efficient way to condense point sets. Figure 1d shows the visually appealing result of k-Means compared to other condensers. In the experiments, we will also show that it performs surprising well in classification tasks.

IV. Empirical Evaluation

In this section we evaluate the performances of the above condensing methods when applied to various set learning algorithms in different image classification tasks. The three condensing methods described in Section III are tested. They are denoted as **Rand:K**, **Unif:K**, and **KMeans:K** respectively, where **K** is the condensed size. **All** is used to denote the original set. We only evaluate the uniform covering condenser on a small problem to demonstrate its deficiencies. For the k-Means condenser, we run k-Means once using the random initialization for 20 iterations.

Five image data sets of different scales and natures are used for evaluation. For classification, as described in Section II-C, we consider both the set-to-set scheme using the **MMK** (1) and the KL divergence based Gaussian kernel (7) (**KLK**), and the set-to-class scheme using **NBNN** [4], **LNBNN** [6], and **NPKL** which is the NBNN classifier paired with the divergence estimator (3).

For MMK and KLK, we use the condenser to reduce the size of every training and testing point set separately. For NBNN, LNBNN, and NPKL, the condensers are used to reduce the class representations $\{H_c\}$. When using KLK with SVM, the resulting kernel matrices are projected to the closest positive definite matrix as in [3]. For SVMs, the kernel width σ and slack parameter C are tuned on the training data using cross validation.

MMK, KLK, and Uniform Covering is only applied to small problems because they are slow compared to other methods. Only LNBNN is used for very large scale problems, since it is the only one that is fast enough.

We extract multiscale *dense SIFT* features called *PHOW* [30] for images. Images are resized so that the longest side is no larger than 256 pixels. The step size 10 is used to sample patches and the patch sizes are $[24, 36, 48]$. This setting will produce about $1,500$ 128-dimensional points for a 256×256 image. We also append the patches' spatial position in the image to the feature vectors to enable spatial matching, making the points 130 dimensional. The weight of the coordinates in distance calculation is roughly tuned on a small subset and kept the same throughout all the algorithms and runs.

The *VlFeat* [31] software is used for k-Means and dense SIFT feature extraction. The *FLANN* [20] software is used for NN search. Exact NNs are used for KLK on small scale data sets. In large experiments of NBNN, LNBNN, and NPKL, we use approximate NNs with four randomized KDTrees. The number of *leaf node checks*, which controls the precision of the approximate NN search, is stated in each experiment and figures. Experiments are done on *Opteron K10 2.3 GHz* CPUs.

A. Scene-15

The first data set we consider is the *Scene-15* data set [32], which is a widely used benchmark for scene classification. This data set contains $4,485$ images from 15 classes. In general a scene image is characterized by the distribution of features *e.g.* the proportion of sky, water, flat surfaces, etc. This is quite different from images for object recognition that we will present later. For this data set, the weight of the spatial coordinates is set to 3. In each run, we randomly choose 100 images for training and 100 images for testing unless stated otherwise.

c) Set-to-Set Classification: To use MMK and KLK for classification, we first calculate the kernel values between every pair of sets by (1) and (7), and then given them to SVM and *k-nearest-neighbor* (KNN) for classification. For KNN the number of neighbors is tuned based on the training data. Because this approach is very computationally expensive, we

test it only on the first 8 classes, known as the *OT* data set [33]. Each image is a set of $1,542$ points, and the condensers Rand:100 and KMeans:100 are compared, meaning that we use subsampling and k-Means to reduce the sets' sizes from $1,542$ to 100. In each run, we randomly choose 50 images for training and 50 for testing.

The performance of 10 runs are reported in Figure 2. We can see that random sampling significantly decreased the accuracies of the classifiers. However, using the KMeans:100 condensing, the performances of SVMs are very close to using the original sets. For MMK, the decrease of mean accuracies is less than 0.5% and the pair t-test shows a p-value of 0.03%. For KLK, the difference is slightly larger at about 2%. This is a very good performance considering that only $1/15$ of the data is kept. It is interesting to see that KNN's accuracy using the k-Means condensed sets is significantly better than the uncondensed result. We shall see later this is not a random effect. As a reference, we also provide the results based on the distance between bag-of-words representations (500 visual words) using the same features and classifiers.

The time to compute both MMK and KLK using all the points is about $6,700$ CPU$\times$minutes. After condensing it took about 26 CPU$\times$minutes to compute MMK and 33 CPU$\times$minutes to compute KLK, which are 200 – 250 times faster. k-Means condensing only took about 0.1 CPU$\times$second for each set.

d) The Uniform Covering Condenser: Here we test the performance of uniform covering condenser paired with LNBNN classification on the full Scene-15 data set. The original H in LNBNN contains about 2×10^6 points. Note that with this condenser we cannot specify the size but only the radius r.

As mentioned in Section III, this condenser is problematic in high dimensional spaces. Figure 3b shows the relationship between the condensed size and the radius r. To reduce the size to *e.g.* $3,000$, we need $r \approx 400$ which is very large. Either increasing or decreasing r would result in a dramatic change of size. This phenomenon is a natural result of the curse of dimensionality. In practice, it is very hard to get a good sense of how large the radius should be in high dimensions.

Figure 3a shows the relationship between the accuracy and the radius from 10 runs. Different number of checks in the approximate NN search is tried. We can see that the decreases of accuracy is unacceptable when the sets are condensed to a reasonable size. In fact, as we will show later, its performance is even worse than random sampling.

This experiment confirms that the uniform covering condenser has ill-formed behavior and bad performance. Moreover, it is also slower than other condensers. Therefore, we shall exclude it from the subsequent experiments.

e) The Random and k-Means Condenser: Now we evaluate the sampling and k-Means condensers with the NBNN, LNBNN, and NPKL classifiers. Again the original classifiers contain about 2×10^6 points. 512 checks are used in NN search. Figure 4 shows the accuracies from 5 random runs using different condensers and different classifiers. We can see that k-Means condensing is much better than random sampling. More surprisingly, classifiers using data condensed by k-Means consistently and significantly outperform those using the original uncondensed data. In other words, we *improved the speed and the accuracy simultaneously*, as opposed to make trade-offs between them. The explanation might be that the condenser removes some of the noisy and outlier points in the original sets. In Figure 4a we can observe that the accuracy decreases a little when the condensed size is very large. We shall see that this behavior is consistent throughout most of our experiments.

We also examine the impact of the number of checks in NN search in Figure 5. The approximate search algorithm performs very well. The impact of the number of checks is minimum, and the performance usually saturates with 512 checks.

B. UIUC-Sports

The UIUC-Sports data set [34] contains $1,030$ images from 8 sport events. In order to test the performance in higher dimensions, we use the color version of the PHOW feature which is 384 dimensional. The image sizes vary so that each one contains 295 to $1,542$ points. In each run 70 images are used for training and 60 for testing. As a result, the original classifiers contain about 6×10^5 points. The weight of spatial coordinates is 0.6. Other settings remain the same as the Scene-15 experiment.

Accuracies of 5 random runs are reported in Figure 6. We can observe again that the k-Means condensing is better than both sampling and no-condensing. This verifies again the benefit of removing noise brought by condensing. We also noticed that using all the points, the accuracy of NPKL is worse than NBNN and LNBNN. But after condensing, these three algorithms performs almost the same. This shows that k-Means condensing could make the data less sensitive to different algorithms.

C. CalTech-101

The CalTech-101 data set [35] is a standard benchmark for object recognition. This data set contains $9,144$ images of 102 different object classes. Unlike the Scene-15 and UIUC-Sports data set, the class of an object's image is more determined by the presence of a few distinctive local features (intuitively the object parts) than the distribution of features.

We follow the standard protocol and use $10, 15, 30$ images per class for training and the rest for testing. We only test the performance of LNBNN using different condensers as it is the only classifier that scales well with this problem. We compare the accuracies of Rand:4000 and KMeans:4000 condensers to the accuracy without condensing. Without condensing, the classifier contains 6×10^6 points taking about 3GB memory given 30 training images, while the condensed classifier only takes 202MB memory irrespective of the number of training images. Note that we use a larger condensed size of $4,000$ expecting that more points are needed to accurately capture the distinctive features of the objects. The weight of spatial coordinates is 1.5.

Figure 7 shows the performance of 10 random runs. We can see that k-Means is much better than the random condensing. k-Means slightly outperforms the uncondensed results again,

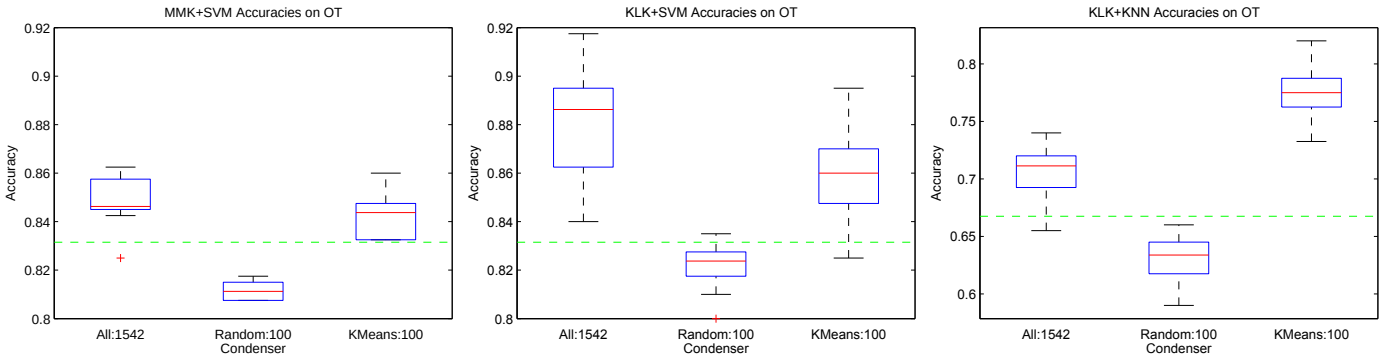
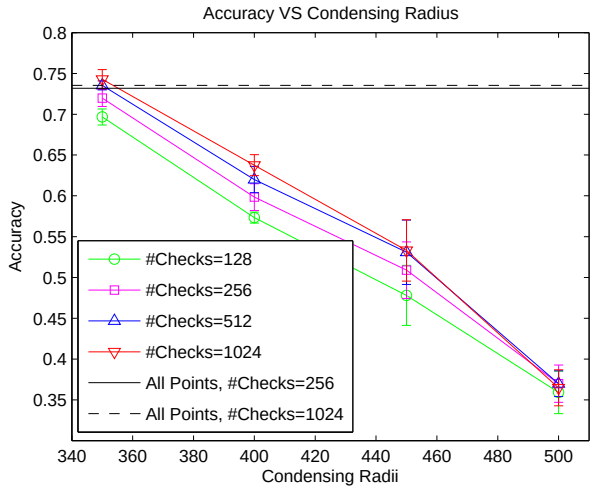
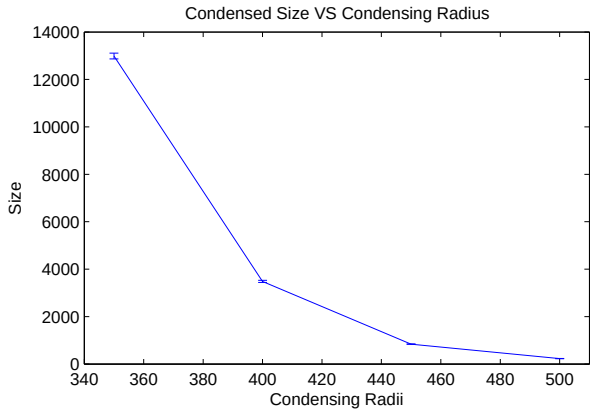


Figure 2: Accuracies on the OT data set using the original sets and the condensed sets. Green dashed lines are the accuracies of bag-of-words classifiers.



(a)



(b)

Figure 3: Performance of LNBNN on Scene-15 using the uniform covering condenser.

even though the difference is insignificant. Observe that the improvement of k-Means condensing over using the all points is becoming smaller as more training images are used. This may be because as more images are added, $4,000$ points is becoming insufficient to capture all the information contained in the training set. The time used for k-Means condensing is 4 CPU*minutes per class with 30 training images. The prediction takes 100 CPU*minutes using the condensed classifier, while without condensing it takes 161 CPU*minutes. The acceleration is not much here because the highly efficient approximate NN search is used. Yet reducing the space requirement by 93% is still a significant benefit.

D. CalTech-256

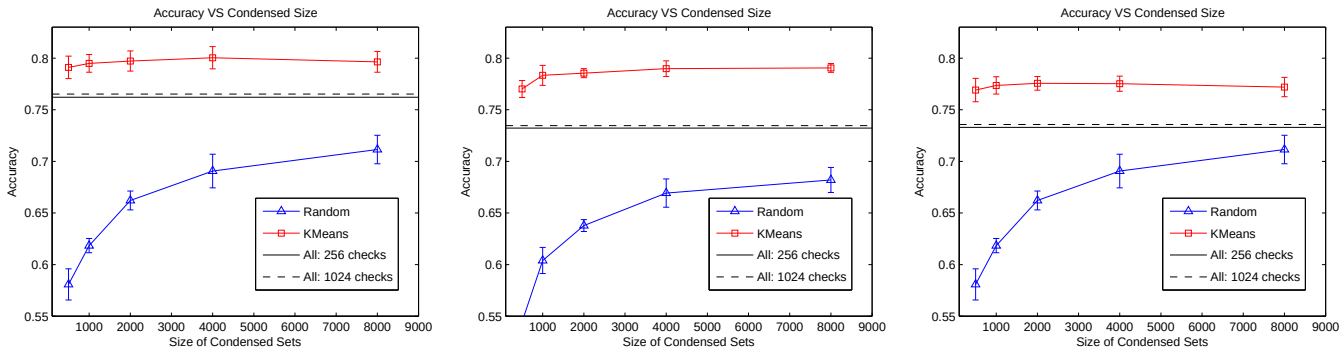
CalTech-256 is an enhanced version of the previous CalTech-101 data set, containing $30,607$ images from 257 object classes. The same settings are used as for CalTech-101 except that the weight of spatial coordinates is 0.6. Note that without condensing, the LNBNN classifier contains 1.4×10^7 points taking 7GB memory, which is approaching the limit of readily available machines. After the condensing, the classifier takes only 500MB memory, irrespective of the number of training images.

Figure 8 shows the performance of 5 random runs. The behaviors of the condensers are basically the same as in the CalTech-101 experiment, showing the consistency of the condensers. Notably, when 30 training images are used, the condenser seems to have reached the limit and causes a slight decrease of accuracy. It shows that the information carried by the training set is finally exceeding the capacity of the KMeans:4000 condenser and more points are needed to maintain performance. The time used for k-Means condensing is again 4 CPU$\times$minutes per class with 30 training images. The prediction takes 440 CPU$\times$minutes using the condensed classifier, while without condensing it takes 791 CPU$\times$minutes. In this larger problem the condenser's acceleration effect is becoming more prominent, even if the approximate NN searcher is used.

E. ImageNet Challenge 2012

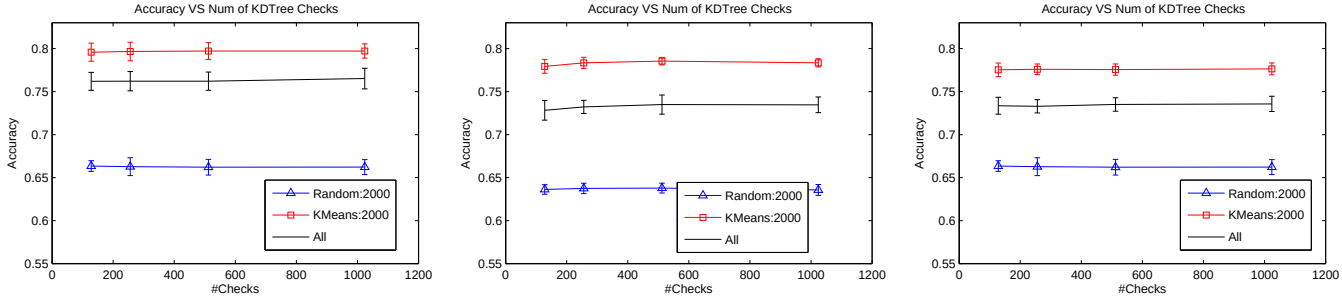
The ImageNet Challenge 2012[2] [36] provides a massive object image classification task, containing $1,261,406$ images from $1,000$ object classes retrieved by crawling the Internet. The large amount of variations in the perspective, object appearance, and background clutter make it a extremely challenging task. Because of the large number of classes and

[2]http://www.image-net.org/challenges/LSVRC/2012/index



(a) NBNN (b) LNBNN (c) NPKL

Figure 4: Scene-15 classification performances using different classifiers and condensers.



(a) NBNN (b) LNBNN (c) NPKL

Figure 5: The impact of the number of checks in the NN search to different methods on the Scene-15 data set.

possible ambiguities, 5 guesses are allowed when predicting an image's label.

We use the dense SIFT features from the organizer http://www.image-net.org/download-features, which provides about 800 SIFT vectors per image. We apply LNBNN to this classification task with 500 images per class for training and 500 images per class for testing, resulting in an experiment that involves 1 million images. This experiment is too large to be feasible on reasonable machines without condensing; the training set alone would take 140GB memory.

Rand:2000 and KMeans:2000 condensers are used in this task. The size of the classifier after condensing is about 1GB. Since we are not able to complete the task with all the training points, the condensed result by Rand:20000 is used as a surrogate, which is feasible but already runs very slow. For the k-Means condensing, we first use random sampling to reduce the input sets' sizes to 10^5 and then run k-Means. 256 checks are used in the NN search, and the weight of spatial coordinates is 0.9.

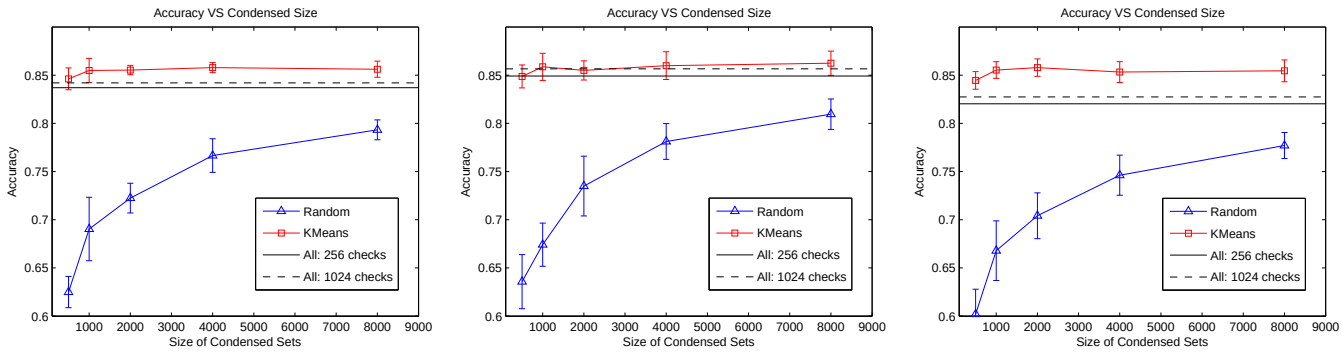
The results from 5 runs of KMeans:2000, Rand:2000 and 2 runs of Rand:20000 are shown in Table I. We can see that k-Means condenser performs around 70% better than the sampling condenser using the same amount of data, and also 11% better than the sampling condenser that uses 10 times more data, showing the effectiveness of k-Means condensing in optimizing the classification performance.

The running time for different condensers are also reported. Note that here "Training" is just the condensing step. We see that even if the approximate NN searcher is used, condensing can still make the prediction speed 6 times faster, and this improvement will become significantly larger if more accurate NN search is needed. The sampling condenser basically costs no time except for the disk IO. On the other hand, the k-Means condensing takes less than 4 minutes per class. In large-scale parallel computation, this extra cost is acceptable, and the improvement to the prediction speed and accuracy is significant. In all, again, condensing makes the classifier smaller, faster, and more accurate.

Note that our results here are mainly to show the effectiveness of the k-Means condenser and not comparable to the ImageNet Challenge's top performers. We used the provided features instead of doing feature engineering/learning, and the algorithm used here is extremely simple and efficient.

Condenser	Rand:2000	Rand:20000	KMeans:2000
Accuracy (%)	14.04 ± 0.05	21.18 ± 0.07	23.7 ± 0.55
Training Time	0.07	0.17	3.7
Testing Time	0.52	2.91	0.53

Table I: Accuracies and running time of LNBNN on ImageNet. The training time is measured by CPU*minute per class and the testing time is measured by CPU*second per test image.



(a) NBNN (b) LNBNN (c) NPKL

Figure 6: UIUC-Sports classification performances using different classifiers and condensers.

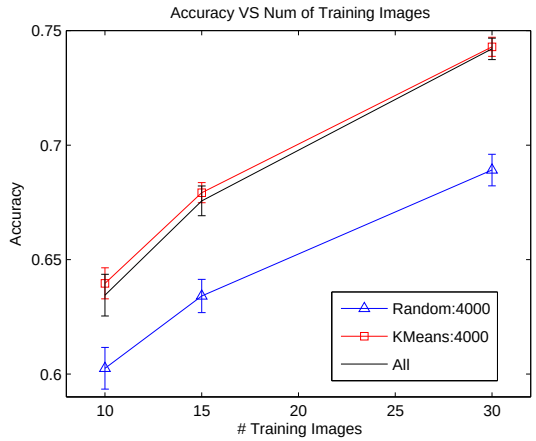


Figure 7: CalTech-101 classification accuracies using LNBNN with different condensers.

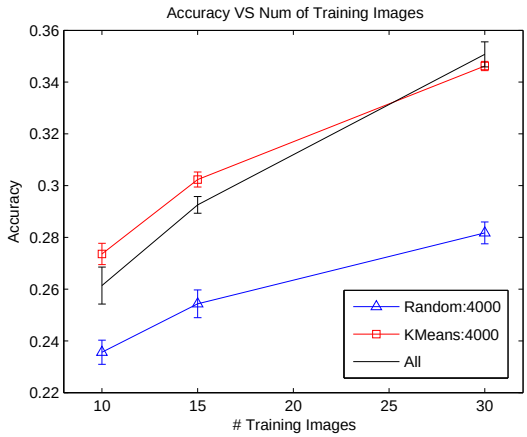


Figure 8: CalTech-256 classification accuracies using LNBNN with different condensers.

V. Discussion

Typically when facing large point sets, a popular approach is to subsample them to make a trade-off between accuracy and speed [23]. Our experiments show that this approach often compromises too much accuracy. However, when we use the k-Means condenser, we can often improve the speed, space requirement, and the accuracy all at the same time.

Depending on the data set, the k-Means condensing can have different impact on the performance. When the sets are mainly characterized by the holistic characteristics of its points, k-Means condensing can not only reduce the size significantly while retaining the information, but it can also possibly remove noise and outliers to enhance the accuracy. Examples of such data sets include the Scene-15 and the UIUC-Sports. If the sets are mainly characterized by a few distinctive points, like in the CalTech data sets, approximation error on the individual points plays a bigger role and condensing is usually less effective. Nonetheless, even in those data sets, we see that k-Means can still at least maintain the accuracy while greatly improve the time and space efficiency.

To use the condensing algorithms, we need to choose the size of the condensed set. A guideline is to set the budget of time and space and use the largest number of points allowed. Our experience shows that $1,000$ – $5,000$ points usually works well for set-vs-class classifiers, and 100 – 500 points should work for set-vs-set classifiers. If the purpose is to use condensing to remove the noise and improve the accuracy, then we can use cross-validation to determine the appropriate size.

The cost of k-Means is not trivial but very manageable. The condensing of different sets are independent. In our experiments, we used Elkan's algorithm [28] in VLFeat [31], which is not the fastest algorithm like [29] but can still condense 10^5 points to $2,000$ points in less than 4 minutes. In a large-scale parallel computation environment like *Map-Reduce*, this is very acceptable. We believe that, given the budget of time and space, it is almost always beneficial to apply k-Means condensing before a learning algorithm on point sets. Compared to random sampling, it will result in a much better accuracy within acceptable time.

For algorithms that need all point-wise similarities or exact NNs in high dimensions, condensing can easily provide quadratic improvement for speed and linear improvement for space requirement. In this latter case, condensing can turn impossible tasks into possibilities. When approximate NN

search is used and we have a small data set, the improvement for speed is not significant, such as in the Scene-15 and UIUC-Sports data set. However, when the data set becomes as big as the CalTech-256 or even the ImageNet datasets, then condensing can provide substantial improvement on top of the approximate NN search.

VI. Conclusions

Efficient algorithms for point sets are important and useful, yet existing methods suffer from high time and space demand. In this paper we tried to condense the point sets in order to make these methods faster and better. We discovered that the k-Means algorithm does this job very well.

On a wide range of algorithms and image data sets, we evaluated three different practical condensing strategies and found the k-Means is the only one that can successfully reduce the size of point sets without much loss of accuracy. In many cases it even improves the accuracy by removing the noise and outliers. This success seems to be universal despite the differences across various classifiers and data sets. We hope our discovery could help the adoption of the point-set based methods by the practitioners in large scale problems.

Acknowledgment

This work is funded in part by NSF grant IIS0911032, NSF grant 1250350, and DARPA grant FA87501220324.

References

[1] F.-F. Li and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2005.

[2] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, 2004.

[3] B. Póczos, L. Xiong, D. Sutherland, and J. Schneider, "Nonparametric kernel estimators for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[4] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest neighbor based image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[5] A. Gretton, K. M. Borgwardt, M. Rasch, B. Schölkopf, and A. J. Smola, "A kernel method for the two sample problem," in *Neural Information Processing Systems (NIPS)*, 2007.

[6] S. McCann and D. G. Lowe, "Local naive bayes nearest neighbor for image classification," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[7] A. Smola, A. Gretton, L. Song, and B. Schlkopf, "A hilbert space embedding for distributions," in *Algorithmic Learning Theory (ALT)*. Springer-Verlag, 2007, pp. 13–31.

[8] S. Lyu, "Mercer kernels for object recognition with local features," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[9] B. Póczos, L. Xiong, and J. Schneider, "Nonparametric divergence estimation with applications to machine learning on distributions," in *Uncertainty in Artificial Intelligence (UAI)*, 2011.

[10] L. Xiong, B. Póczos, and J. Schneider, "Hierarchical probabilistic models for group anomaly detection," in *AI and Statistics (AISTATS)*, 2011.

[11] ——, "Group anomaly detection using flexible genre models," in *Neural Information Processing Systems (NIPS)*, 2011.

[12] Z. Zhou, "Multi-instance learning: A survey," Department of Computer Science & Technology, Nanjing University, Tech. Rep., 2004.

[13] T. Gärtner, P. A. Flach, A. Kowalczyk, and A. J. Smola, "Multi-instance kernels," in *Internation Conference on Machine Learning (ICML)*, 2002.

[14] K. Muandet, K. Fukumizu, F. Dinuzzo, and B. Schölkopf, "Learning from distributions via support measure machines," in *Neural Information Processing Systems (NIPS)*, 2013.

[15] C. Wallraven, B. Caputo, and A. Graf, "Recognition with local features: the kernel recipe," in *International Conference on Computer Vision (ICCV)*, 2003.

[16] K. Grauman and T. Darrell, "The pyramid matching kernel: Discriminative classification with sets of image features," in *Internation Conference on Computer Vision (ICCV)*, 2005.

[17] L. Bo and C. Sminchisescu, "Efficient matching kernels between sets of features for visual recognition," in *Neural Information Processing Systems (NIPS)*, 2009.

[18] Q. Wang, S. R. Kulkarni, and S. Verdú, "Divergence estimation for multidimensional densities via k-nearest-neighbor distances," *IEEE Trans. on Information Theory*, vol. 55, 2009.

[19] C. SilpaAnan and R. Hartley, "Optimized kd-trees for fast image descriptor matching," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[20] M. Muja and D. G. Lowe, "Fast approximate nearest nneighbor with automatic algorithms configuration," in *International Conference on Computer Vision Theory and Applications (VISAPP)*, 2009.

[21] T. Tuytelssars, M. Fritz, K. Saenko, and T. Darrell, "The nbnn kernel," in *International Conference on Computer Vision (ICCV)*, 2011.

[22] S. García, J. Derrac, J. R. Cano, and F. Herrera, "Prototype selection for nearest neighbor classification: Taxonomy nd empirical study," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, 2012.

[23] F. Provost, D. Jensen, and T. Oates, "Efficient progressive sampling," in *International Conference on Knowledge Discovery and Data Mining (KDD)*, 2009.

[24] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *The Annals of Applied Statistics*, 2011.

[25] M. E. Tipping and B. Schölkopf, "A kernel approach for vector quantization with guaranteed distortion bounds," in *AI and Statistics (AISTATS)*, 2001.

[26] F. Jurie and B. Triggs, "Creating efficient codebooks for visual recognition," in *International Conference on Computer Vision (ICCV)*, 2005.

[27] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 24, 2002.

[28] C. Elkan, "Using the triangle inequality to accelerate k-means," in *International Conference on Machine Learning (ICML)*, 2003.

[29] M. Shindler, A. Wong, and A. Meyerson, "Fast and accurate k-means for large datasets," in *Neural Information Processing Systems (NIPS)*, 2011.

[30] A. Bosch and a. X. M. A. Zisserman, "Image classifcation using random forests and ferns," in *International Conference on Computer Vision (ICCV)*, 2007.

[31] A. Vedaldi and B. Fulkerson, "Vlfeat: An open and portable library of computer vision algorithms," http://www.vlfeat.org/, 2008.

[32] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[33] A.Oliva and A. Torralba, "Modmodel the shape of the scene: a holistic representation of spatial envelope," *International Journal of Computer Vision (IJCV)*, vol. 42, 2001.

[34] L.-J. Li and F.-F. Li, "What, where and who? classifying events by scene and object recognition," in *International Conference on Computer Vision (ICCV)*, 2007.

[35] F.-F. Li, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Workshop on Generative-Model based Vision*, 2004.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.