Category Detection Using Hierarchical Mean Shift

Pavan Vatturi
School of EECS
1148 Kelley Engineering Center
Oregon State University
Corvallis, OR 97331
vatturi@eecs.oregonstate.edu

Weng-Keen Wong
School of EECS
1148 Kelley Engineering Center
Oregon State University
Corvallis, OR 97331
wong@eecs.oregonstate.edu

ABSTRACT

Many applications in surveillance, monitoring, scientific discovery, and data cleaning require the identification of anomalies. Although many methods have been developed to identify statistically significant anomalies, a more difficult task is to identify anomalies that are both interesting and statistically significant. Category detection is an emerging area of machine learning that can help address this issue using a "human-in-the-loop" approach. In this interactive setting, the algorithm asks the user to label a query data point under an existing category or declare the query data point to belong to a previously undiscovered category. The goal of category detection is to bring to the user's attention a representative data point from each category in the data in as few queries as possible. In a data set with imbalanced categories, the main challenge is in identifying the rare categories or anomalies; hence, the task is often referred to as *rare* category detection. We present a new approach to rare category detection based on hierarchical mean shift. In our approach, a hierarchy is created by repeatedly applying mean shift with an increasing bandwidth on the data. This hierarchy allows us to identify anomalies in the data set at different scales, which are then posed as queries to the user. The main advantage of this methodology over existing approaches is that it does not require any knowledge of the dataset properties such as the total number of categories or the prior probabilities of the categories. Results on real-world data sets show that our hierarchical mean shift approach performs consistently better than previous techniques.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering*; G.3 [**Probability and Statistics**]: [Nonparametric statistics]

General Terms

Algorithms, Experimentation, Performance

KDD'09 June 28–July 1, 2009, Paris, France.

Keywords

clustering, anomaly detection, category detection, mean shift

1. INTRODUCTION

Many applications in surveillance, monitoring, scientific discovery, and data cleaning require the identification of anomalies. Ideally, these anomalies correspond to data points or events of interest, such as a disease outbreak in biosurveillance or a network attack in intrusion detection. Although many methods have been developed to characterize anomalies as statistically unusual events, not all statistically significant anomalies are necessarily useful. In fact, many anomalies are simply uninteresting, corresponding to known sources of noise or known combinations of features that are irrelevant to actual events of interest.

Consider the following two examples. The first example, taken from [15], involves the task of scientific discovery from the Sloan Digital Sky Survey (SDSS) [17], which is a 5 year survey of the northern skies by ground-based telescopes. Most of the images in the SDSS capture known phenomena such as stars, comets, nebulae, etc. which have already been discovered. Anomalies in the data correspond to unusual or unknown objects which could potentially lead to new scientific discoveries. However, the majority of anomalies in the SDSS are of no interest to astronomers. These anomalies include diffraction spikes, which are artifacts of the telescope, and satellite trails. Such anomalies clearly do not lead to new discoveries in astronomy. The anomalies of interest, such as unusual galaxies, are extremely rare and constitute a miniscule 0.001% of the entire data set. A similar task exists in the analysis of network log files. The IT infrastructure of a company can contain thousands of computers and devices networked together. These components are often equipped with monitoring agents that generate log files that capture characteristics of the network traffic. Statistical anomalies in these log files often correspond to uninteresting events such as those arising from events that are already known or expected, such as events arising from maintenance upgrades. A very small fraction of the log file anomalies correspond to actual network failures or attacks of interest. Identifying these meaningful anomalies would be beneficial for the diagnosis of faults and the prevention of malicious attacks.

Thus, a challenging new task has emerged for the field of anomaly detection – identifying anomalies that are not only statistically significant but also interesting. Since the "interestingness" of an anomaly is subjectively defined, a human-in-the-loop approach is needed. Category detection [15] is an emerging area of machine learning that can help address

this issue. Category detection operates on a set of unlabeled examples $S = \{x_1, x_2, ..., x_n\}$ where $x_i \in \Re^d$ are from m distinct categories[1] labeled $y_i = \{1, 2..., m\}$. The category detection algorithm asks the user to label a query data point under an existing category or declare the query data point to belong to a previously undiscovered category. The goal of category detection is to bring to the user's attention at least a single instance from each category in as few queries to the user as possible. Figure 1 illustrates the interactive category detection process. The main challenge in this task is discovering the rare categories, which appear as small, dense clusters or isolated outliers in the data set. This task becomes especially challenging if the data set is dominated by a handful of disproportionately large categories, which makes the rare categories become extremely difficult to discover through manual inspection. As a result, category detection is often referred to as *rare* category detection.

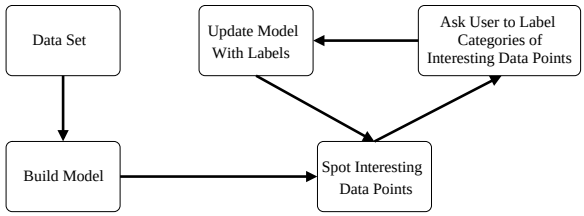

Figure 1: The interactive category detection loop

In this paper we present a new approach to rare category detection using a Hierarchical Mean Shift procedure. Mean Shift is a nonparametric clustering technique widely used in computer vision for segmentation. In our approach, we use Mean Shift to discover cluster modes. By repeatedly applying Mean Shift with increasing bandwidths, we can create a hierarchy of clusters at different scales and thus use this information to score each cluster by how "anomalous" it is. This score can then be used to rank the representative data points of each cluster for labeling by a user. The main advantage of the proposed method over previous related work [15, 8] is twofold. First, our approach dramatically reduces the number of queries to the user needed to discover all the categories in the data. Secondly, our approach does not require any prior knowledge regarding the properties of the data set, such as the total number of clusters present in the data or the prior probability of a data point belonging to a cluster.

2. RELATED WORK

Category detection incorporates ideas from active learning [11] and semi-supervised learning [19, 1, 21] but work that investigates category detection itself is relatively sparse. We briefly review the main contributions in the area of category detection.

Category detection was first proposed by Pelleg et al. [15] through their Interleave algorithm, which assumes that the data is generated by a mixture model. Interleave starts with an entirely unlabeled data set and clusters it using EM for mixture models. Using the results of EM, the Interleave algorithm maintains, for every mixture component, a list of data points that are "most owned" by that mixture component. This list is sorted in increasing order of the "degree of ownership" by the mixture component. Intuitively, the algorithm queries data points that are "least owned" by the components. It cycles through all the lists in a round robin fashion and selects the data point in the first position of each component's list for user-labeling. Once the user has provided the category labels, the Interleave algorithm clamps the labeled data points to their user-supplied labels before EM is applied again. This process continues until the user terminates the loop.

Despite some nice properties such as being model-independent and robust to noise in the data, Interleave has several shortcomings. First, the algorithm is sensitive to initial conditions as the EM clustering converges to a local optimum. Second, the algorithm requires an initial estimate of the number of classes in the data. Although the number of classes changes as the user provides feedback to the Interleave algorithm, this initial estimate of the number of classes is often not known in advance and an incorrect value at the outset can adversely affect the algorithm. Finally, as pointed out by [8], Interleave requires the classes to be separable in feature space.

He et al. [8] propose a nearest-neighbor based active learning for rare category detection (NNDM) to address the separability assumption in Interleave. NNDM uses an unsupervised local density differential sampling strategy. It makes use of nearest neighbors to measure the local density around each data point. The algorithm starts with an entirely unlabeled data set. In each iteration, the algorithm selects for labeling the data point with the largest change in local density. Like Interleave, NNDM needs to know the number of classes in advance. NNDM also requires the prior probability of a data point belonging to each class. Although NNDM is effective at discovering categories that overlap each other, we have noticed some undesirable behavior by NNDM in which the algorithm repeatedly queries data points (with large local density differential values) from the same cluster which has already been discovered.

Fine et al. [5] abstract the rare category detection problem as an output identification task in a learning model. The learning model has an unknown target function f which maps every input in χ to one of m output values. The output identification task is to find m inputs, one for each output value. The algorithm knows that the target function f is in a given function class F. The work assumes an unknown distribution over the inputs and describes algorithms for many classes of functions. The algorithms are shown to have an expected sample bound of the order of m and $log(1/\epsilon)$ where ϵ is the lower bound on the probability of each output class. The work similarly reports sample bounds in special cases like binary outputs i.e. $m = 2$, specific distributions (the uniform distribution) and a concept class defined over the Boolean hypercube $\{0, 1\}^n$.

3. MEAN SHIFT

Mean Shift is a non-parametric clustering algorithm proposed by Fukunaga and Hostetler [6]. It is based on the concept of nonparametric estimation of probability density functions in which the value of a density function at a point can be estimated using the sample observations that fall within a small region around that point. The popular Parzen window technique generalized this concept for density estimation. The Mean Shift algorithm is commonly used in vision for image segmentation [2, 3] but it has received rel-

[1]We will use the word category and class interchangeably.

atively little attention in the machine learning community compared to other clustering techniques. We now describe Mean Shift based on the description and notation in Comaniciu and Meer [3].

3.1 Kernel Density Estimation

Let $(x_1, \ldots, x_n)$ be n independent and identically distributed d-dimensional data points. Furthermore, let $\boldsymbol{H}$ be a symmetric positive definite d-by-d bandwidth matrix. We define the kernel function with bandwidth $\boldsymbol{H}$ to be

$$K_{\boldsymbol{H}}(x) = |\boldsymbol{H}|^{-1/2} K(\boldsymbol{H}^{-1/2} x) \tag{1}$$

The term $K(x)$ is a d-dimensional kernel function that is non-negative and integrates to one. It satisfies the conditions for asymptotic unbiasedness, consistency, and uniform consistency of the gradient of the density estimate [6]. The multivariate kernel density estimator with bandwidth matrix $\boldsymbol{H}$ computed at point x is given by

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} K_{\boldsymbol{H}}(x - x_i) \tag{2}$$

In order to reduce the complexity of estimation, we assume $\boldsymbol{H}$ is proportional to the identity matrix $\boldsymbol{H} = h^2 \boldsymbol{I}$ where $h > 0$. Assuming $\boldsymbol{H}$ to be proportional to the identity matrix, the multivariate kernel density estimator in Equation 2 becomes

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{3}$$

Various multivariate kernel functions can be used for density estimation using Equation 3. In our work, we use a radially symmetric kernel obtained by rotating in R^d a symmetric univariate Gaussian kernel function with mean 0 and variance 1. The kernel is of the form

$$K_N(x) = (2\pi)^{-d/2} \exp(-\frac{1}{2}\|x\|^2) \tag{4}$$

We can write this kernel more abstractly as:

$$K(x) = c_{k,d} k(\|x\|^2) \tag{5}$$

where $c_{k,d}$ is a normalization constant that makes $K(x)$ integrate to one and $k(x)$ is called the profile of kernel $K(x)$. As defined by [2], the profile $k(x)$ is a function $k : [0, \infty] \rightarrow \mathcal{R}$ such that $K(x) = k(\|x\|^2)$. For the Gaussian kernel function, the profile is given by

$$k_N(x) = \exp(-\frac{1}{2}x) \qquad (x \geq 0). \tag{6}$$

The density estimator expression for the multivariate normal kernel is:

$$\hat{f}(x) = \frac{c_{k,d}}{nh^d} \sum_{i=1}^{n} k(\|\frac{x - x_i}{h}\|^2) \tag{7}$$

3.2 Density Gradient Estimation

The density gradient estimator can be obtained by the gradient of the density estimator in Equation 7 [3]. If we define $k'(x)$ as the derivative of the profile, the gradient of the density estimator can be calculated as:

$$\begin{aligned} &\bigtriangledown \hat{f}_{h,K}(x) \\ &= \frac{2c_{k,d}}{nh^{d+2}} \sum_{i=1}^{n} (x - x_i) k'(\|\frac{x - x_i}{h}\|^2) \\ &= \frac{2c_{k,d}}{nh^{d+2}} \left[\sum_{i=1}^{n} k'\left(\left\|\frac{x - x_i}{h}\right\|^2\right)\right] \left[\frac{\sum_{i=1}^{n} x_i k'\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} k'\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x\right] \end{aligned} \tag{8}$$

The second term, shown in Equation 9 is the Mean Shift which is the difference between the weighted mean, using k' for the weighting, and x.

$$m_{h,K}(x) = \frac{\sum_{i=1}^{n} x_i k'\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} k'\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x \tag{9}$$

Using the normal kernel, the Mean Shift vector becomes

$$m_{h,K}(x) = \frac{\sum_{i=1}^{n} x_i \exp\left(\left\|\frac{x - x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} \exp\left(\left\|\frac{x - x_i}{h}\right\|^2\right)} - x \tag{10}$$

$m_{h,K}(x)$ is proportional to the normalized density gradient and always points toward the steepest ascent direction of the density function.

3.3 Mean Shift Clustering

Mean Shift can then be used for clustering or mode detection by iteratively shifting each data point towards the direction of its Mean Shift vector [6]. The amount of the shift is proportional to the gradient at the data point. Each data point x^i, at the ith iteration, is shifted according to Equation 11. Eventually, the points converge to a stationary point which corresponds to a mode of the density.

$$x^{i+1} = x^i + m_{h,K}(x^i) \tag{11}$$

We distinguish between two data sets required for the Mean Shift algorithm, which we call the *reference* and the *query* data sets. The reference data set consists of the data points that are used for the density estimation. The points in the reference data set do not move. The query data set consists of data points that are shifted by every iteration of Mean Shift. When the Mean Shift algorithm converges, the query data set contains the cluster modes. The query data set is not necessarily the same as the reference data set. When the query data set is the reference data set, the process is known as blurring [2] since it progressively blurs the data set on each iteration.

3.4 Bandwidth Selection

The bandwidth parameter h plays a critical role in the accuracy of the density estimation during Mean Shift. If the bandwidth is too small, it produces an undersmoothed density estimate, resulting in many small clusters. If the bandwidth is too large, the density estimate will be oversmoothed, resulting in a small number of very large clusters. A variety of techniques have been developed for bandwidth selection, such as the Maximal Smoothing Principle, Likelihood Cross Validation, Least Squares Cross Validation and Plug-in methods [9].

4. METHODOLOGY

Our approach to category detection is based on Hierarchical Mean Shift (HMS) [12, 4], which produces a hierarchy of clusters by repeatedly blurring the data. The repeated blurring of the data is accomplished by re-running Mean Shift with increasingly larger bandwidths, using at each iteration the cluster centers from the previous iteration. HMS is closely related to scale-space theory [12], which is a multiscale data representation framework designed to model how the human visual system sees details in an image as the image is progressively smoothed.

One of the main reasons we chose HMS is because Mean Shift is a non-parametric clustering algorithm, which permits clusters to take on arbitrary shapes, unlike a Gaussian Mixture Model. Initially, we created a non-parametric version of the Interleave algorithm by replacing EM with Mean Shift and measuring cluster ownership by the total distance shifted to reach the cluster mode. However, this Interleave variant (called MS-Interleave) requires estimating the bandwidth parameter h. After trying many different bandwidth estimation techniques [9] on a variety of data sets, we found that in many cases, the bandwidth estimates tend to undersmooth the data, producing a large number of modes when Mean Shift is applied[2]. With a large number of modes, the MS-Interleave algorithm produces many redundant queries for the same cluster as the algorithm believes that the data points belonging to that cluster belong to different clusters.

In order to avoid selecting a single optimal bandwidth value, we used HMS, which repeatedly blurs the data with increasing bandwidths. HMS results in a cluster hierarchy similar to a dendrogram, except the clustering algorithm used is Mean Shift and not single-linkage clustering. This hierarchy contains extremely useful information regarding the properties of the clusters such as their outlierness, compactness, and isolation, which we will define in more detail in Section 4.3. We use these properties to determine which data points, that are representative of these clusters, should be brought to the attention of the user. One other benefit to our HMS-based category detection is the fact that it does not require the user to input parameters requiring knowledge of the data set properties, such as an initial guess as to the number of clusters in the data (in the case of Interleave) or the category priors (in the case of NNDM).

Our HMS procedure for category detection consists of 3 phases: 1) data standardization, 2) building the cluster hierarchy and 3) querying data points.

[2]We also tried a common post-processing step in Mean Shift where modes that are within some ϵ of each other are merged into the same cluster but the results for category detection were still poor.

4.1 Data Standardization

The data sets are first standardized to prevent one dimension dominating the distance calculations used in the analysis. Sphering is perfomed on the datasets using the following transformation [13].

$$Z_i = \Lambda^{-1/2} Q^T (x_i - \bar{x}) \qquad i = 1, ..., n \tag{12}$$

In the equation above, $\bar{x}$ is the sample mean, the columns of Q are the eigenvectors of the sample covariance matrix, Λ is a diagonal matrix of corresponding eigenvalues and x_i is the i^{th} data point in the dataset.

4.2 Building the cluster hierarchy

In the second step, HMS builds the cluster hierarchy. Before we can apply Mean Shift, we first compute the initial bandwidth value by finding the minimum non-zero distance h_{min} between any two points in the data set[3]. The bandwidth matrix is computed as $\boldsymbol{H} = h_{min}^2 * \boldsymbol{I}$. For convenience, we will refer to the bandwidth $\boldsymbol{H}$ as a scalar value h.

The lowest level of the hierarchy consists of the individual data points. The initial iteration of HMS applies Mean Shift with bandwidth h_{min} to these individual data points. Each data point either remains as its own cluster of size one or it is merged into a larger cluster. On the next iteration of HMS, the centers of these clusters become the data points at the next level of the hierarchy and the bandwidth is increased by multiplying the current bandwidth scalar value by k, which is the bandwidth increment. Mean Shift, with the increased bandwidth, is applied to these data points (which are the cluster centers one level down) to produce a new set of clusters. Mean Shift runs faster for each iteration of HMS as the number of data points at each level of the hierarchy decreases quickly. This entire process continues until there is only one cluster left. In effect, we are repeatedly blurring the data by summarizing the data points in terms of their cluster centers.

The algorithm to build the cluster hierarchy is described in Algorithm 1. In each iteration of Algorithm 1, Mean Shift is applied to a list of cluster centers stored in $Hierarchy[l].ClusterCenters$ to produce a clustering of these cluster centers. Initially, each individual data point is considered a cluster center. Hierarchical Mean Shift performs some bookkeeping that is not shown in Algorithm 1. For each cluster center at hierarchy level l, the algorithm stores the total distance it was moved by Mean Shift to a new cluster center at level $(l+1)$. Furthermore, the algorithm maintains two cluster membership lists for each cluster at level $(l+1)$. The first cluster membership list consists of the previous cluster centers at level l. The second cluster membership list consists of the original data points in the initial query data set.

4.3 Querying the user

One of the advantages of Hierarchical Mean Shift is its ability to preserve "the structure and integrity of the outliers in the clustering process" [12]. Leung et al. define different cluster validity metrics such as lifetime, compactness, isolation and outlierness to measure the "goodness" of

[3]As mentioned in Section 4.5, we are using kd-trees to speed up Mean Shift, which allows this minimum distance computation to be efficient.

Algorithm 1: Hierarchical Mean Shift - Building the clustering hierarchy

Let S be the data set and h be the bandwidth parameter;
Set h = minimum non-zero distance between any two points in S;
Set Hierarchy[0].ClusterCenters = S;
$l = 0$;
repeat
  newClusterCenters = MeanShift(Hierarchy[l].ClusterCenters, h);
  Set Hierarchy[l+1].ClusterCenters = newClusterCenters;
  $h = h * 1.1$;
  $l = l + 1$;
until *size(Hierarchy[l].ClusterCenters) == 1* ;

a cluster in the hierarchy. This section describes two types of criteria, each of which can be used with Hierarchical Mean Shift resulting in two different approaches to rare category detection. We refer to these two approaches as Hierarchical Mean Shift - Outlierness (HMS-Out) and Hierarchical Mean Shift - Compactness-Isolation (HMS-CI).

Outlierness.

The *Outlierness* criterion [12] is based on the concept of the lifetime of a cluster. The lifetime of a cluster is defined as the range of logarithmic scales over which the cluster survives. Similarly, we can define the lifetime in the Hierarchical Mean Shift scenario as the range of logarithmic bandwidths over which the cluster survives i.e. the logarithmic difference between the bandwidth when the cluster is formed and the bandwidth when the cluster is merged with other clusters. The Outlierness metric helps in identifying outliers in the data. The cluster which has a long lifetime and fewer points will have high Outlierness value. Intuitively, rare categories are more likely to have high Outlierness value as they tend to have fewer points and longer lifetime.

$$Outlierness_i = \frac{\text{lifetime of } C_i}{\text{number of data points in } C_i} \tag{13}$$

The cluster hierarchy built in the first phase is traversed and the list of unique clusters is formed. The same cluster can persist at more than one level of the hierarchy and all the different occurrences are treated as one unique cluster. The Outlierness value for each cluster is calculated and the list is sorted in the decreasing order of their Outlierness value. To query a cluster, we ask the user to label its representative data point, which is the data point which has moved the least in terms of its Mean Shift distance before being assigned to the cluster center.

Compactness-Isolation.

Compactness and isolation [12] are another set of criteria that can measure the quality of a cluster. Intuitively, a cluster is well-defined if the distance between the data points inside the cluster is small (ie. it is compact) and those outside is large (ie. it is isolated). Given p_i is the cluster center of C_i and h is the scalar bandwidth parameter, the *isolation* and *compactness* of C_i can be calculated as

$$isolation = \frac{\sum_{x \in C_i} e^{-\|x-p_i\|^2/2h^2}}{\sum_x e^{-\|x-p_i\|^2/2h^2}} \tag{14}$$

$$compactness = \frac{\sum_{x \in C_i} e^{-\|x-p_i\|^2/2h^2}}{\sum_{x \in C_i} \sum_j e^{-\left\|x-p_j\right\|^2/2h^2}} \tag{15}$$

The *isolation* and *compactness* metric are close to one for a good cluster. These two criteria can be combined into one single $Compactness - Isolation$ criterion (CI) for the cluster which is simply the sum of *isolation* and *compactness* values of the cluster. This criterion is calculated for each cluster in the cluster hierarchy and stored in a list sorted in decreasing order. The CI criterion is more computationally expensive to compute than Outlierness. The worst case run time complexity for CI is $O(n^2)$ and occurs when CI needs to be computed for all clusters at the lowest level of the cluster hierarchy.

The list created by using one of the above two criteria is traversed in the third phase. Clusters with higher validity criterion are selected first. The representative point of the cluster is chosen as it was in the Outlierness criterion. If this representative point has already been selected for labeling then the cluster is skipped; otherwise, the representative point is presented to the user for labelling. The entire algorithm is described in Algorithm 2.

Algorithm 2: Hierarchical Mean Shift for rare category detection

Build cluster hierarchy using Hierarchichal Mean Shift.
Let ht be its height.
Let L = empty
for $i \leftarrow ht$ **to** *0* **do**
  For each cluster C at height i calculate its validity criteria value if it is not done previously and add it to list L;
end
Sort list L in decreasing order of the validity criteria values of the clusters;
while *not all classes have been discovered* **do**
  Let C be the next cluster to select in list L;
  Let p be the data point which has least Mean Shift distance from the cluster center of C;
  if *p has not been selected for labeling* **then**
    Present p to user for labeling;
  else
    continue;
  end
end

4.4 Tiebreaker

The sorted list may contain cluster entries with the same criterion values. These ties in criterion values can happen for clusters at the lower levels of the hierarchy as low bandwidth values lessen the effect of neighboring points for a given cluster resulting in high Compactness-Isolation values. Ties in Outlierness also occur when small compact clusters have low lifetimes due to being near much bigger clusters. In such cases a tiebreaker condition can be applied. For each cluster with the same criterion values, the distance between the cluster mode and each data point already labeled by

the expert is calculated. For our highest average distance (HAD) tiebreaker, the point with the highest average of the distances is picked first for labeling. This heuristic means that data points near already queried points are left to be queried later, thus decreasing the possibility of picking data points near the classes that have already been discovered. In this way, tiebreakers use the labels that have been provided by the user to improve performance. Computing the HAD is typically very quick when k is small as the number of clusters with the same criterion value and the number of points already labeled by the expert are far less than n.

4.5 Computational Considerations

Each iteration of Mean Shift requires computing the distance between an individual data point and all other data points. This distance computation is then performed over all the data points. If there are n data points of dimension d and if Mean Shift requires M iterations to converge, then the total time complexity of Mean Shift is $O(n^2dM)$. Mean Shift can be computationally expensive when n is large, hence several techniques, including locality-sensitive hashing [7] and an improved fast Gauss transform [20], have been developed to speed up the Mean Shift algorithm. Another technique for speeding up Mean Shift relies on the use of kd-trees [16], which is a multi-resolution space-partitioning data structure that can be used to dramatically reduce the $O(n^2)$ complexity of an all-pairs computation. The data points falling into a partition defined by a leaf node of the kd-tree are treated as a homogenous group, thereby allowing an approximation to be computed. Thus, rather than computing the distance between data point x_i and all other data points $x_j, j \neq i$, one can compare x_i against groups corresponding to the partitions of the kd-tree.

In our work, we used a single kd-tree approach which is similar to the optimization used for kernel regression by Deng et al. [10]. Like Mean Shift, kernel regression requires the calculation of the weighted average of points falling within a hypersphere centered at the query point, with radius specified by the bandwidth. This computation is shown in Equation 16, where $\hat{y}(x_q)$ is the weighted average being computed, x_q is the query point, and $w_i(x_q)$ is the weight of the ith datapoint with respect to the query point.

$$\hat{y}(x_q) = \frac{\sum_{i=1}^{n} w_i(x_q)x_i}{\sum_{i=1}^{n} w_i(x_q)} \tag{16}$$

Equation 16 for kernel regression is similar to Equation 9 for meanshift.

The use of kd-trees in kernel regression is based on the idea that if we have a group with k datapoints in which we know that all the weights in the group with respect to the query point x_q are close to the same value w then approximate values of $\sum w_i(x_q)x_i$ and $\sum w_i(x_q)$ can be used. This approximate value can be obtained in constant time without the need to sum the individual members of the group. Implementation of this approximation is straightforward by supplementing a kd-tree with extra information at each node. The nodes in the kd-tree act as hyper-rectangles enclosing all the points of the node and are treated as groups. To compute $\sum w_i(x_q)x_i$ and $\sum w_i(x_q)$ a top-down search of a kd-tree is performed where at each node a decision is made either to treat all the points in the node as a group (cutoff) or recursively continue the search of the children (recurse). Considering x_q and the hyper-rectangle of the node, the minimum and the maximum distance of x_q from the hyper-rectangle i.e. D_{min} and D_{max} can be easily computed in turn, providing the maximum and minimum possible weights w_{min} and w_{max} of any data points owned by the node. If the values of w_{min} and w_{max} are close enough then the cutoff option is taken. The algorithm makes a cutoff if

$$\frac{(w_{max} - w_{min})N_B}{\text{weight so far in search}} < \tau \tag{17}$$

where τ is a system constant.

Wang et al. [18] propose a dual-tree optimization for mean-shift in which two kd-trees are built separately, one for the query points and the other for the reference points. These trees are traversed simultaneously and each reference tree node's weight contribution to a query tree node is recursively updated by comparing these two nodes and their children. Once both trees have been traversed, the dual-tree algorithm produces a memory-efficient cache of the Mean Shift values of all the query points. However, for every iteration of Mean Shift, the query tree has to be rebuilt since the query points have been changed while the reference tree remains fixed. The dual-tree method is ideally suited for datasets which need a small number of Mean Shift iterations [18]. The data sets in our work involve many iterations of Mean Shift. As a result, the dual-tree Mean Shift method tends to be slower than the single kd-tree method due to the frequent rebuilding of the query tree.

5. EVALUATION

We evaluate our algorithms on the Abalone, Shuttle, Optical Digits, Optical Letters, Statlog and Yeast data sets taken from the UCI data repository [14]. We chose data sets that had a large number of class labels and had continuous feature values. All the datasets except for Abalone and Yeast are sub-sampled. This sub-sampling is done to create imbalanced data sets that suit the rare category detection problem where some classes dominate the data and the remaining classes have only a few records. Shuttle is randomly sub-sampled from the original dataset to produce a smaller data set with 4000 examples. The original Optical Digits, Optical Letters, and Image Segmentation (Statlog) datasets contain almost the same number of examples for each class. Following the evaluation in [15], we changed the class distributions for the Optical Digits and Image Segmentation datasets into a geometric series with the largest class owning half of the data and each subsequent class being half as small with the smallest class containing 8 examples. The Optical Letters data set has been sampled in such a way that the two largest classes own half the data and the subsequent pairs of classes being half as small with the smallest class containing 8 examples. Table 1 has a summary of their properties. The number of records listed in Table 1 is after subsampling if the data sets were subsampled. The data sets are first standardized before running the experiments.We use a bandwidth increment of $k = 1.1$ in all our experiments.

All the algorithms are evaluated based on the total number of queries presented to the user before the user sees at least one example from all the classes in the data set. In these queries, the user is assumed to provide the correct class label for the queried data point. This evaluation metric is the standard used in rare category detection [15, 8]. The assumption with this performance metric is that a sin-

Name	Dims	Records	Classes	Smallest class	Largest class
Abalone	7	4177	20	0.34%	16%
Shuttle	8	4000	7	0.02%	64.2%
Optical Digits	64	1040	10	0.77%	50%
Optical Letters	16	2128	26	0.37%	24%
Statlog	19	512	7	1.5%	50%
Yeast	8	1484	10	0.33%	31.68%

Table 1: Properties of the data sets.

Dataset	HMS-CI	HMS-CI +HAD	HMS-Out	HMS-Out +HAD	NNDM	Interleave
Abalone	1195	**93**	603	385	124	193
Shuttle	44	32	36	**28**	162	35
Optical Digits	**100**	**100**	160	118	576	117
Optical Letters	**133**	**133**	161	182	420	489
Statlog	**18**	20	34	124	228	54
Yeast	**73**	91	103	77	88	111

Table 2: Number of queries needed to identify all classes for the HMS methods using the Highest Average Distance (HAD) tiebreaker

gle example would help the expert to generalize about the class to which the example belongs. Intuitively, the metric measures the effort the expert expends in order to discover all the classes in the data set. These results are summarized in Table 2.

Table 4 illustrates the category detection curves, which show the number of classes discovered as a function of the number of queries to the user. The area under the category detection curve can be used as an alternative metric for evaluating the algorithms. The AUC metric favors algorithms that can quickly discover the majority of the classes but are slow to discover the last few remaining classes. Table 3 summarizes the AUC metric for the different algorithms, with the AUC being normalized by the total area in the graph.

We compare the performance of 'Hierarchical Mean Shift - Outlierness' (HMS-Out), 'Hierarchical Mean Shift - CompactnessIsolation' (HMS-CI), 'Hierarchical Mean Shift - Outlierness + HAD Tiebreaker (HMS-CI+Out), and 'Hierarchical Mean Shift - CompactnessIsolation + HAD Tiebreaker' (HMS-CI+HAD) against existing rare category detection techniques NNDM [8] and Interleave [15] on the 6 data sets. Since the performance of Interleave is sensitive to the starting conditions for EM, we ran Interleave 10 times on each data set with different random starting conditions and reported the best result. We set the category priors for NNDM to be their empirically observed values in the data sets.

6. DISCUSSION

The results in Table 2 and Table 4 show that the HMS approaches outperform both NNDM and Interleave. Without the tiebreaker heuristic, the HMS variants outperform Interleave and NNDM on 4 out of the 6 datasets and come within one hint of Interleave on the Shuttle dataset.

The Abalone data set was a difficult data set for category

Dataset	HMS-CI	HMS-CI +HAD	HMS-Out	NNDM	Interleave
Abalone	0.835	**0.873**	0.837	0.846	0.840
Shuttle	0.925	**0.929**	0.917	0.480	0.905
Optical Digits	**0.855**	**0.855**	0.840	0.199	0.808
Optical Letters	**0.936**	**0.936**	0.917	0.573	0.765
Statlog	0.956	**0.958**	0.944	0.472	0.934
Yeast	0.821	0.805	0.793	**0.838**	0.778

Table 3: Area under the category detection curves in Table 4, normalized by the total area in the graph.

detection by HMS. From the category detection curves for Abalone in Table 4, we see that the HMS-CI method finds 18 of the total 20 classes more quickly than NNDM and Interleave, but fails to find the last 2 classes. The HMS-Out method finds 17 of 20 classes more quickly than NNDM and Interleave and also fails to discover the same 2 classes as HMS-CI. Analyzing the Abalone dataset reveals that these classes are small compact clusters that are very close to more than one large cluster. As a result, their Outlierness lifetime is low as they are merged into one of the larger clusters early on in the cluster hierarchy building phase, resulting in a low Outlierness value. Hence, HMS-Out fails to find these clusters quickly. Similarly, in the case of HMS-CI, the points from nearby larger clusters contribute substantially to the denominators of the compactness and isolation criteria of the small cluster, resulting in low values for Compactness-Isolation. The end result is that for Abalone, there are several data points that are tied either in terms of Outlierness or Compactness-Isolation near the bottom of the cluster hierarchy. The NNDM algorithm performs well on the Abalone data set because it is successful at discovering changes in local density.

If we add the HAD tiebreaker heuristic, the HMS category detection methods outperform both NNDM and Interleave on all six data sets. The HAD heuristic allows the category detection algorithm to query the data points that are tied with the same criterion values using a more intelligent ordering. The improvement in category detection for the Abalone data set is due to the ability of the HAD heuristic to require the queries to be further away from all of the already queried data points. Without this tiebreaker, the HMS algorithms will query the data points with identical criterion values in some arbitrary order.

The HMS-CI methods, both with and without the HAD tiebreaker heuristic, outperform the HMS-Out methods on all data sets except for Shuttle. The CI criteria appears to be a better criterion for selecting data points for category detection but it is more computationally expensive to compute than Outlierness.

We also experimented with a range of values for the bandwidth increment k. Due to space limitations, we do not include the results here. Our empirical results show that for a range of $k = 1.1 - 1.9$, the HMS-CI+HAD algorithm has little variation in the total number of hints needed to discover all classes. The running times initially decrease as k increases. However, at a certain point when k becomes large, the running time starts to increase due to the cost of computing the HAD tiebreaker heuristic.

For future work, we would like to improve the use of the user feedback for category detection. Currently, the feedback is only used in the HAD tiebreaker heuristic. We would also like explore different presentation options, such as presenting the entire cluster to the user, rather than just the representative point from the cluster. In addition, we would like to make the HMS algorithm even more efficient. The bottleneck in the algorithm is building the cluster hierarchy. Although the data standardization and the cluster hierarchy construction can be completed offline while the query selection can be performed online, we would like to make the algorithm scale to extremely large data sets. Finally, we would like to investigate the theoretical issues surrounding the use of HMS for category detection.

7. CONCLUSION

We have proposed a rare category detection method using Hierarchical Mean Shift. A cluster hierarchy is built by successively running Mean Shift first on the dataset and then on the cluster centers using a series of increasing bandwidth values. Then, data points are chosen for querying using two different criteria: Outlierness and Compactness-Isolation. For data points with identical criterion values, we use a highest average distance heuristic as a tiebreaker. This HMS approach has a number of attractive properties. It does not require the user to provide information regarding the dataset properties such as the number of classes or the prior probabilities of the classes. Furthermore, the non-parametric nature of Mean Shift removes any restrictions on the shapes of the clusters. Finally, and most importantly, the HMS approach discovers all the categories in the data sets used in our experiments in much fewer queries than existing approaches such as Interleave and NNDM.

8. REFERENCES

[1] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Proceedings of the Twenty-First International Conference of Machine Learning*, pages 81–88, New York, NY, 2004. ACM Press.

[2] Yizong Cheng. Mean shift, mode seeking, and clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(8):790–799, 1995.

[3] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002.

[4] Daniel Dementhon. Spatio-temporal segmentation of video by hierarchical mean shift analysis. In *SMVP 2002 (Statistical Methods in Video Processing Workshop)*, 2002.

[5] Shai Fine and Yishay Mansour. Active sampling for multiple output identification. *Mach. Learn.*, 69(2-3):213–228, 2007.

[6] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on*, 21(1):32–40, 1975.

[7] Bogdan Georgescu, Ilan Shimshoni, and Peter Meer. Mean shift based clustering in high dimensions: A texture classification example. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 456, Washington, DC, USA, 2003. IEEE Computer Society.

[8] Jingrui He and Jaime Carbonell. Nearest-neighbor-based active learning for rare category detection. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 633–640. MIT Press, Cambridge, MA, 2008.

[9] M. Chris Jones, James S. Marron, and Simon J. Sheather. A brief survey of bandwidth selection for density estimation. *Journal of American Statistical Association*, 91(433):401–407, March 1996.

[10] Andrew Moore Kan Deng. Multiresolution instance-based learning. In *Proceedings of the Twelfth International Joint Conference on Artificial Intellingence*, pages 1233–1239, San Francisco, 1995. Morgan Kaufmann.

[11] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. Active learning with gaussian processes for object categorization. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 2007.

[12] Yee Leung, Jiang-She Zhang, and Zong-Ben Xu. Clustering by scale-space filtering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1396–1410, 2000.

[13] Wendy L. Martinez. *Exploratory Data Analysis with MATLAB (Computer Science and Data Analysis)*. Chapman & Hall/CRC, November 2004.

[14] C.L. Blake D.J. Newman and C.J. Merz. UCI repository of machine learning databases, 1998.

[15] Dan Pelleg and Andrew Moore. Active learning for anomaly and rare-category detection. In *Advances in Neural Information Processing Systems 18*, December 2004.

[16] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry - An Introduction*. Springer, 1985.

[17] Alexander S. Szalay. The sloan digital sky survey. *Comput. Sci. Eng.*, 1(2):54–62, 1999.

[18] Ping Wang, Dongryeol Lee, Alexander Gray, and James Rehg. Fast mean shift with accurate and stable convergence. In *In Proceedings of AISTATS 2007*, 2007.

[19] Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. In *Advances in Neural Information Processing Systems 15*, pages 505–512. MIT Press, 2003.

[20] Changjiang Yang, Ramani Duraiswami, Nail A. Gumerov, and Larry Davis. Improved fast gauss transform and efficient kernel density estimation. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, page 464, Washington, DC, USA, 2003. IEEE Computer Society.

[21] Liu Yang and Rong Jin. An efficient algorithm for local distance metric learning. In *in Proceedings of AAAI*, 2006.

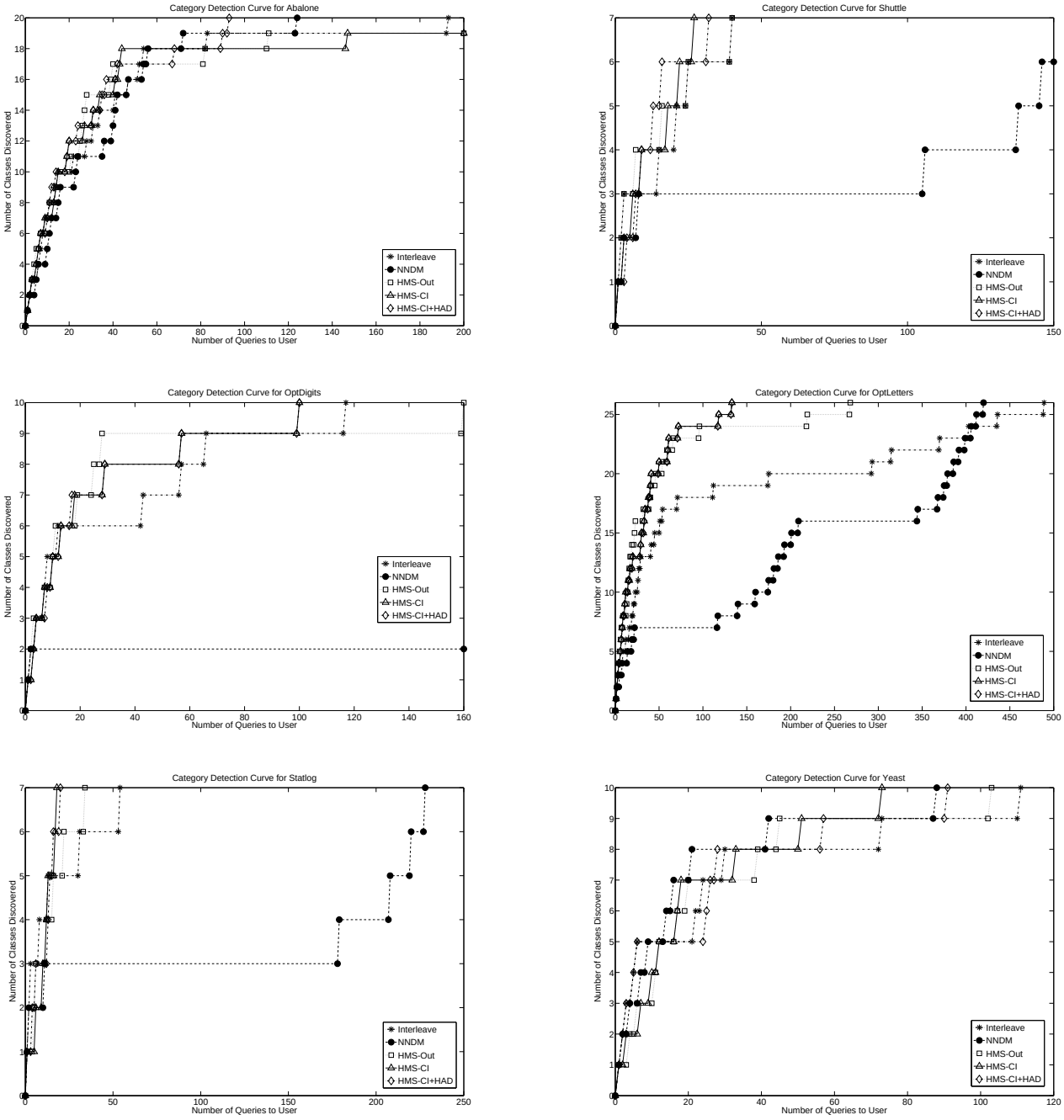

Table 4: Category Detection Curves for the Abalone, Shuttle, Optical Digits, Optical Letters, Statlog and Yeast data sets. We show the results of applying Interleave, NNDM, HMS-Out, HMS-CI, and HMS-CI+HAD to these data sets. To avoid excessive clutter, we do not show HMS-Out+HAD since its results are similar to that of HMS-CI+HAD.