Mixtures of Rectangles: Interpretable Soft Clustering

Dan Pelleg DPELLEG@CS.CMU.EDU
Andrew Moore AWM@CS.CMU.EDU
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract

To be effective, data-mining has to conclude with a succinct description of the data. To this end, we explore a clustering technique that finds dense regions in data. By constraining our model in a specific way, we are able to represent the interesting regions as an intersection of intervals. This has the advantage of being easily read and understood by humans.

Specifically, we fit the data to a mixture model in which each component is a hyper-rectangle in M-dimensional space. Hyper-rectangles may overlap, meaning some points can have soft membership of several components. Each component is simply described by, for each attribute, lower and upper bounds of points in the cluster. The computational problem of finding a locally maximum-likelihood collection of k rectangles is made practical by allowing the rectangles to have soft "tails" in the early stages of an EM-like optimization scheme. Our method requires no user-supplied parameters except for the desired number of clusters. These advantages make it highly attractive for "turn-key" data-mining application. We demonstrate the usefulness of the method in subspace clustering for synthetic data, and in real-life datasets. We also show its effectiveness in a classification setting.

1. Introduction

Technological advances have made collection and storage of massive amounts of data commodity technologies. Unfortunately, much of this data is doomed to accumulate dust in the warehouse because deriving useful information from it is still a human-controlled and poorly understood task. We focus our view on clustering. A typical example for this problem is the mixtures-of-Gaussians model. This is a powerful model which allows for a wide variety of cluster shapes. It also conveniently accommodates membership of any point in multiple clusters (a property known as "soft" membership). It can be fitted accurately with the EM method, and scaled to massive datasets (Moore, 1998). But, the produced clusters are inherently hard to understand, and so the strength of the model becomes its weakness. When interpreted back in the original data domain, a Gaussian cluster involves computing weighted distances from datapoints to cluster centroids. In arithmetic terms, this boils down to subtractions and multiplications of record attributes, most probably represented in different, often incomparable, units. The reason people turn to data-mining is to seek help with important decisions such as marketing strategies or credit allocation. From the cognitive aspect, it is highly unlikely that any decision will be taken when its only justification is such cumbersome manipulation. To make our technique useful, we must choose an aesthetically pleasing model, even if the price is reduced expressiveness.

Our model is a mixture of uniform density M-dimensional hyper-rectangles, supplemented with Gaussian "tails". The tails mean that (as is the case with conventional Gaussians), the probability of a datapoint decreases with the distance from the "centroid". But, in contrast to conventional Gaussians, there is a difference in the way we measure distances. We consider the distance from the point to the closest point to it that is included in the rectangular kernel. Note that under this definition all points contained in the kernel are equally (and maximally) likely. We fit the model by means of an EM procedure. Finally, we report the rectangular kernels. These are just lists of intervals (one interval per dimension, per cluster) the intersection of which defines the dense regions. Note that this model only has $O(M)$ parameters, as opposed to $O(M^2)$ for mixtures of Gaussian (however, it is likely to require more components).

Much of the related work in the area of clustering is concerned with scaling of the algorithms to support huge datasets. CLARANS (Ng & Han, 1994) performs a search over the space of centroids for a k-means model (Duda & Hart, 1973; Bishop, 1995). BIRCH (Zhang et al., 1995) aims to scale this to massive datasets by concentrating on heuristic clustering of the data into spherical clusters while minimizing running time and memory usage. In (Pelleg & Moore, 1999)[1] we show how to use a kd-tree to make this calculation exact, and extend this result to high-dimensionality data in (Moore, 2000), and to automatic estimation of the number of clusters in (Pelleg & Moore, 2000).

Liu et al. (2000) show how to use decision-trees, traditionally used in supervised learning, in clustering. While the generated cluster description is human-readable for simple problems, one can easily construct an example where the cutoff points chosen by the decision tree are not very meaningful. This approach also assumes hard membership. CLIQUE (Agrawal et al., 1998) is specifically designed to generate interpretable output in the form of a DNF formula. The creation of these formulae, however, is done as a post-processing step and may miss the goal of presenting the clusters succinctly. Another problem is that they support only a single notion of membership (whether the datapoint is in a dense region or not). This precludes multiple class memberships. It also requires two user-supplied parameters (the resolution of the grid and a density threshold) which are unlikely to be specified correctly for all but expert users and simple densities. Nagesh et al. (1999) try to fix this, but the hard-membership assumption still holds in their work.

Learning axis-parallel boxes and their unions has been discussed in Maass and Warmuth (1995). Note, however, that our algorithm is unsupervised whereas the learning-theory work is mainly concerned with supervised learning. Another example of a machine learning approach that searches rectangles is Friedman (1998), which addresses the supervised learning problem of finding a hyper-rectangle in input space that contains points with relatively high mean output value.

2. The probabilistic model and derivation of the EM step

2.1 Tailed Rectangular Distributions

We begin by defining M to be the number of dimensions. A hyper-rectangle R will be represented by a pair of M-length vectors, which define the upper (R^h) and lower (R^l) boundaries for each dimension. Let x_d denote the d-th element of the vector x. Define the function $\text{closest}(x_d, l, h)$ as the closest point to x_d on the interval $[l, h]$:

$$\text{closest}(x_d, l, h) = \begin{cases} l & \text{if } x_d < l \\ x_d & \text{if } l \le x_d \le h \\ h & \text{if } h < x_d \end{cases}$$

with the natural multi-dimensional extension of $\text{closest}(x, R)$ being the point in R which is closest to x. Consider the following single-rectangle PDF:

$$P(x) = K \exp -\frac{1}{2} \sum_{d=1}^{M} \left(\frac{x_d - \text{closest}(x_d, R^l_d, R^h_d)}{\sigma_d} \right)^2 \tag{1}$$

$P(x)$ can be thought of as a generalization of the Gaussian distribution where the covariance matrix is zero except for the squares of the elements of σ on the diagonal. What makes it different from Gaussian is the way it measures distances to the distribution mean; instead of being the distance between two fixed points, we measure how far away x is from the boundary of the rectangle R [2].

Another important property of this distribution is that it can be factored into independent components:

$$\begin{aligned} & K \exp -\frac{1}{2} \left[\sum_{d=1}^{M} \left(\frac{x_d - \text{closest}(x_d, R^l_d, R^h_d)}{\sigma_d} \right)^2 \right] \\ = & \prod_{d=1}^{M} K_d \exp -\frac{1}{2} \left(\frac{x_d - \text{closest}(x_d, R^l_d, R^h_d)}{\sigma_d} \right)^2 . \end{aligned}$$

One of these components is shown in Figure 1.

And so when we proceed to integrate it we only need to consider the single-dimension case:

$$\int_{-\infty}^{\infty} \exp -\frac{1}{2} \left(\frac{x - \text{closest}(x, R)}{\sigma} \right)^2 dx$$

where x and σ are now real numbers and R is an interval $[l, h]$. We get an integral similar to the one derived from the univariate Normal distribution, except for the fact that the mean point is stretched into an interval:

$$\int_{-\infty}^{l} \exp -\frac{1}{2} \left(\frac{x - l}{\sigma} \right)^2 dx$$

[1] Independently published by AlSabti et al. (1999).

[2] Strictly speaking, the definition is how far x is from any point in R. This is implied since $\text{closest}(x, R) = x$ for $x \in R$.

$$
\begin{aligned}
&+ \int_l^h \exp(0)dx \\
&+ \int_h^\infty \exp -\frac{1}{2}\left(\frac{x-h}{\sigma}\right)^2 dx \\
&= \sqrt{2\pi}\sigma + (h-l) .
\end{aligned}
$$

Thus the normalizing constant in Equation 1 is simply the product of the following per-dimension constants:

$$K^{-1} = \prod_{d=1}^{M} \left[\sqrt{2\pi}\sigma_d + (R_d^h - R_d^l)\right] .$$

Note the penalty imposed on high-volume rectangles. This will later prevent them from expanding infinitely to try and capture all possible points.

2.2 Maximum likelihood estimation of a single Tailed Rectangular Distribution

Suppose we have a dataset $X = \{x^1, x^2, \ldots x^N\}$ and given σ we wish to find the Maximum Likelihood (MLE) set of $2M$ parameters defining the rectangle. The log-likelihood function is

$$
\begin{aligned}
LL(X) \quad &= \qquad\qquad (2) \\
\sum_d \Bigg(-N\log(\sqrt{2\pi}\sigma_d + R_d^h - R_d^l) \\
+ \quad \sum_i -\frac{1}{2}\left[\frac{x_d^i - \text{closest}(x_d^i, R_d^l, R_d^h))}{\sigma_d}\right]^2\Bigg) .
\end{aligned}
$$

From this equation it is immediately clear that we can perform the MLE for each dimension independently. For each dimension d in turn find the values l and h that maximize

$$
\begin{aligned}
&(-N\log(\sqrt{2\pi}\sigma + h - l)) \\
+ \quad &\textstyle\sum_i -\frac{1}{2}\left[\frac{x_d^i - \text{closest}(x_d^i, l, h)}{\sigma_d}\right]^2 .
\end{aligned}
$$

We begin by guessing an initial value of (l, h). First we fix the low-point at l, and find a good candidate for the new high boundary h'. Then we find a good candidate for the low boundary l' based on the existing h. For each of the boundaries, we want to maximize the likelihood which is a function of the new value. To achieve this we use the golden-ratio one-dimensional optimizer (Press et al., 1992). Under the assumption that the function is unimodal in the given range, this optimizer will find a maximum point of it, up to the specified tolerance value. The number of function evaluations is logarithmic in the width of the range. Although we have not yet proved that the likelihood function is indeed unimodal, empirical evidence suggests this is indeed the case.

2.3 EM search for a mixture of Tailed Rectangles

EM for mixture model clustering general takes the following form:

1. Begin with a guess of the the mixture parameters $\{p_1, \theta_1, p_2, \theta_2, \ldots p_k, \theta_k\}$ where p_j is the probability of the mixture generating a datapoint from component j and θ_j are the parameters for the j-th component. In our example θ_j consists of $2M$ parameters: the upper and lower bounds of the j-th rectangle. Note that we are holding σ fixed during the iterations—we are not estimating it by EM.

2. For each datapoint x^i and each mixture component j let w_{ij} be:

   $$P(\text{generated by } j\text{-th component} \,|\, \text{located at } x_i)$$

   or, more succinctly,

   $$w_{ij} = P(\text{class} = j \,|\, x_i) .$$

   This, by Bayes' rule, is

   $$w_{ij} = \frac{P(x_i \,|\, \text{class} = j) \cdot p_j}{\sum_l P(x_i \,|\, \text{class} = l) \cdot p_l} .$$

3. For each component j, re-fit the parameters θ_j to a "weighted" version of the dataset in which the i-th datapoint is given weight w_{ij}. Each datapoint thus only counts as a fraction of a full datapoint in its contribution to the likelihood.

   In our case this means we need to reestimate the coordinates of rectangle $R = R_j$ to maximize

   $$
   \begin{aligned}
   &LL(X) = \\
   &\sum_d \Bigg(-(\sum_i w_{ij})\log(\sqrt{2\pi}\sigma_d + R_d^h - R_d^l) \\
   + \quad &\sum_i -\frac{w_{ij}}{2}\left[\frac{x_d^i - \text{closest}(x_d^i, R_d^l, R_d^h)}{\sigma_d}\right]^2\Bigg) .
   \end{aligned}
   $$

   This can again be achieved one dimension at a time with two golden-ratio searches per dimension: one for the lower and one for the upper bound.

2.4 The full algorithm

The full algorithm follows. Initialize the mixture components by taking, e.g. initial points as used in the anchors hierarchy (Moore, 2000). Initialize σ (we currently use some constant fraction of the range of the data in each dimension). For each dimension and component, compute a new upper boundary based on the existing lower one, and a new lower boundary, based on the current upper one. Change the boundaries and iterate. Once stability is obtained, decrease σ (we multiply it by a constant factor) and continue iterating. Terminate if the components are all stable immediately after decreasing σ.

2.5 Example

For illustration, Figure 2 shows a run on a synthetic two-dimensional dataset made by drawing points from a triple-rectangular distribution. Initially, the boundaries are quite arbitrary and σ is large. This allows the rectangles to freely move to better locations. In following iterations, as σ decreases, the rectangles try to reposition the boundaries to capture as many points as possible (since now the penalty for excluding them is high). After a few more iterations, the distribution is modeled accurately.

2.6 Intuition

We are really interested in obtaining a final mixture of hard rectangles (i.e., with infinitely steeply declining tails). But the key to the whole algorithm is the use of relatively wider tails in the early stages. Without them, it is impossible for rectangles to move because they would pay an infinite penalty for missing a single datapoint with weight $w_{ij} > 0$. Thus, without the tails, the initial EM iterations see all rectangles jump to the bounding box of the data, where they remain. It is only with the tails that the rectangles are guided in directions to grow or shrink, and are able to negotiate ownership of points with the other rectangles.

3. Experimental Results

Our first test is as follows. Fix two parameters r and M. As usual, M is the dimensionality of the data. Additionally, choose a set $R \subseteq \{1 \ldots M\}$ of r relevant dimensions. Now generate points, choosing one of two classes for each point x, and then setting the i-th coordinate to be:

$$\begin{cases} \text{uniform}(0,1) \text{ if } i \notin R \\ \text{uniform}(0,0.5) \text{ if } i \in R \text{ and } x \text{ is in class 1} \\ \text{uniform}(0.5,1) \text{ if } i \in R \text{ and } x \text{ is in class 2} \end{cases}$$

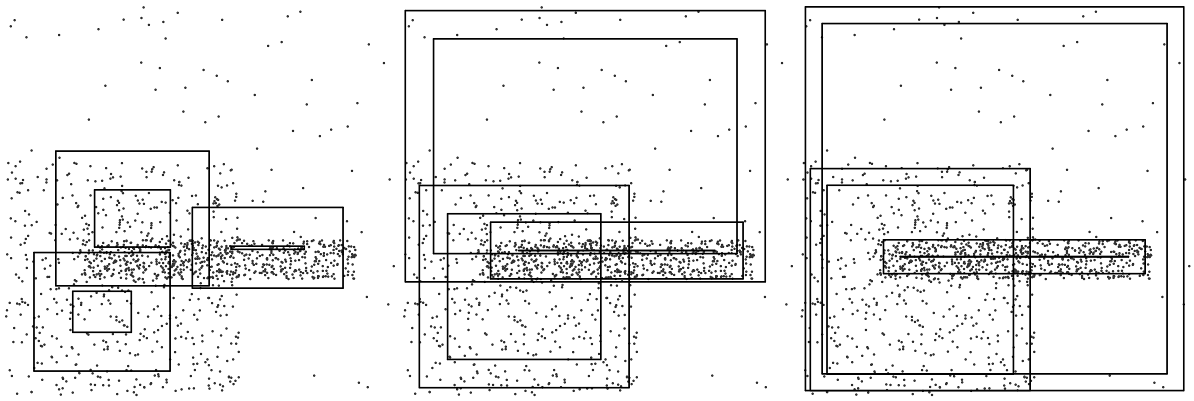
For $M = r = 2$, this distribution looks like a 2×2 checkerboard. This set is interesting because it clearly contains clusters, yet when the data is projected onto any single dimension, the distribution is indistinguishable from uniform.

After estimating the rectangles, we evaluated the result by rounding the boundaries to $0, 0.5$, or 1 if they lie within less than 2σ away from these values, and keeping them unchanged otherwise (σ was never more than 0.075 in any dimension). We then declare the run a "success" if the rounded boundaries match the generating rectangles exactly — that is, they always have 0 and 1 in the irrelevant dimensions, and $0, 0.5$, or 1, as the case may be, in the relevant ones. Results are shown in Figure 3. In general, it was possible to identify the relevant $r = 5$ dimensions from data with dimensionality up to $M = 30$.

Another experiment involves a similar setup, this time in three dimensions. Consider the a 3^3 grid placed on the unit cube. We will use nine of the resulting grid cells to generate datapoints from. See Figure 4. Now, projection along any *two* dimensions is indistinguishable from the joint two-dimensional uniform distribution. Again, the estimated mixture is very close to the original one, as seen in Figure 5.

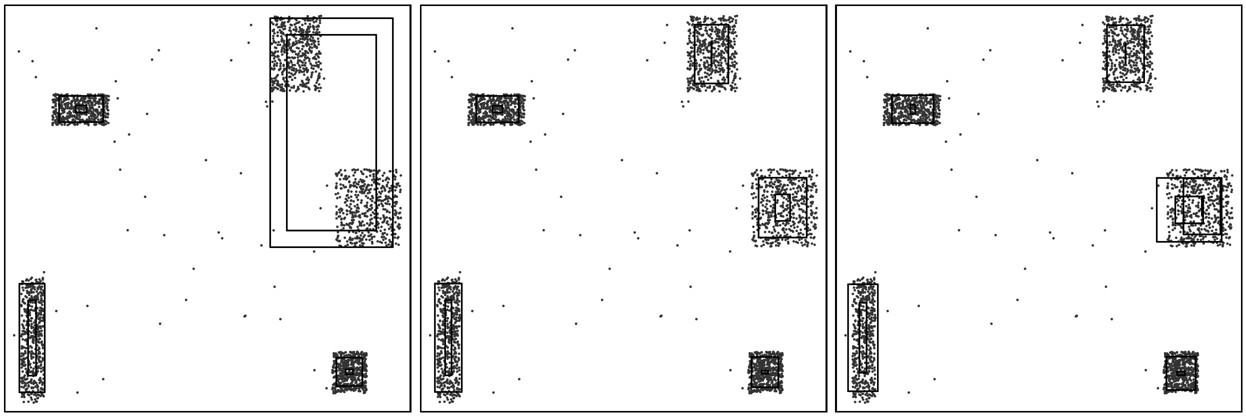
In another experiment, datapoints were generated from a mixture as follows. In each dimension, a random interval with expected width 0.4 was drawn from $[0, 1]$. The intersection of these intervals defines a hyper-rectangle, and datapoints were generated from the union of several such components. The estimated distribution was evaluated as follows. Fix a mapping from the true distribution to the estimated one. Each estimated rectangle is compared to the matching true rectangle by taking the M-th root of the ratio between the volumes of the intersection of the two rectangles to their union. The *similarity* of the two distributions is just the average of these values, taken over all rectangle pairs. The values reported here are the maximum similarity values taken over all possible mappings (ie, all permutations over $[1, \ldots k]$, where k is the number of components). See Figure 6.

We have also performed sporadic experiments to test the sensitivity of the algorithm to the different parameters. Figure 7 shows how supplying the wrong number of rectangles might affect the run. Another parameter is the initial value for σ. We currently use a rather simplistic estimate of $1/10$ of the range of the input. It does work for our datasets. A more sophisticated approach would be to first try and estimate σ (say, using a model with spherical Gaussians) and use the estimate to set the rectangle tails.



(a) (b) (c)

Figure 2. A three-component example. Shown from left to right, the components after the first, 20-th, and 40-th iteration. The inner rectangles mark the kernel boundary, while the outer ones are σ away from them.



(a) (b) (c)

Figure 7. Results when the reported number of rectangles k differs from the true number (which is 5). From left to right, the reported number was 4, 5, and 6. In (a), one of the estimated rectangles takes over two clusters. In (c), two estimated rectangles "fight" over the points in a single cluster.

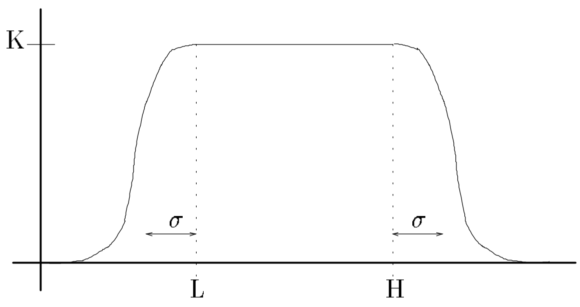


Figure 1. The 1-dimensional form of a rectangle (in this case a line-segment) with tails. An M-dimensional tailed rectangle is simply a product of these.

Experiments on real-life data were done on the "mpg" and "census" datasets from the UCI repository (Blake & Merz, 1998). The "mpg" data has about 400 records with 7 continuous[3] attributes. Running on this data with the number of components set to three, we get the results shown in Table 1. Interpreting this data, we see that cars with at most four cylinders are the most economical, and that eight-cylinder cars weighing more than 3000 pounds are the most environmentally-harmful. While these facts should come as no surprise, it is important to note that the "mpg" attribute was not flagged as special in the input. It was found useful in defining clusters because it has inherent correlation with other attributes. Had we been asked to

[3] We treat discrete attributes as continuous if the values can be linearly ordered (e.g., number of cylinders and model year).

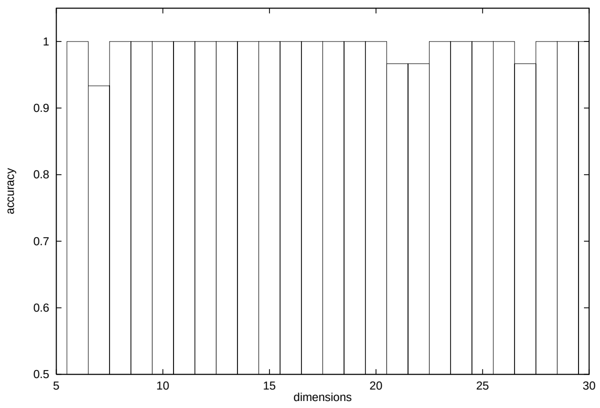


Figure 3. Estimated boundaries for the "checkerboard" data with 5 relevant dimensions. The y axis is the fraction of the experiments in which exactly the relevant dimensions were identified (out of 30 trials). The x axis is the dimensionality of the data (M). Similar experiments were held for 2 and 10 relevant dimensions (not shown).

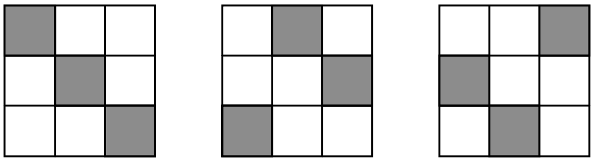


Figure 4. Grid cells used for generation of data. Shown, from left to right, "slices" along the third dimension, each of width 1/3.

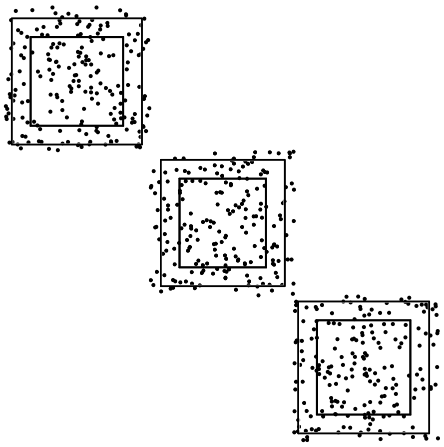


Figure 5. Estimated distribution along the first two dimensions for the "cube" dataset. Only datapoints in the first "slice" are shown. The inner rectangles mark the kernel boundary, while the outer ones are σ away from them.

classify the model year, for example, we would run the program in the exact same way. From the results we would conclude that this attribute is not informative (because all clusters have the same boundaries for it, approximately). Another interesting feature of this output is the fact that eight-cylinder cars are included in both the second and third cluster. So these clusters are overlapping, and further distinction can be made on other attributes (such as displacement, or weight). This effect is due to the "soft" membership that is inherent in the model.

The "census" set is significantly bigger (about $32,000$ records in the training set). It has a binary "income" attribute (indicating whether the annual income is over $\$50K$) which is to be predicted, based on other values. Again, we did not specify this to the algorithm directly, but rather let it cluster the whole data. On termination, we inspect to see if the resulting clusters have a very narrow range for this attribute, indicating that they represent records that are mostly in the same income class[4]. This was observed clearly when the number of clusters was set to four. Results are in Table 2. Three of the clusters predict the "income" attribute to be either zero or one. We built a simple classifier based on these clusters. If a datapoint is contained in one of the regions defined by the kernel boundaries, plus or minus the vector σ, it predicts the label zero or one according to the value in the corresponding estimated cluster. It also predicts one or zero if the point belongs in more than one cluster, but all said clusters have the same label. If we treat all other cases as wrong classification, we get accuracy of 78% on the test-set. But, if we classify as "zero" (the more frequent label in the training-set) the 3537 records that belong in no component, the accuracy increases to 97%. Note that the UCI repository contains a survey of the performance of classical techniques, such as C 4.5, Naive Bayes, and nearest-neighbor, on this data. The reported accuracies for about 16 algorithms range from 79% to 86%. This experiment shows the usefulness of clustering (and, in particular, of our technique) in a supervised-learning task, when the label attribute is informative.

4. Conclusion and Future Work

We have demonstrated a method that benefits from the usefulness of soft membership and the expressiveness and clean analysis of mixture models. At the same time, the produced clusters have a compact rep-

[4]The interval width is typically not zero, due to both numerical and statistical reasons, such as the inclusion of outliers.

Table 1. Clusters for the "mpg" data. "Prob." is the mixture probability for the cluster; the remaining attributes are from the input. Numbers are rounded to the nearest integer.

PROB.	MPG	CYLINDERS	DISPLACEMENT	HORSEPOWER	WEIGHT	ACCELERATION	MODELYEAR
52%	$[18, 46]$	$[3, 4]$	$[75, 150]$	$[49, 112]$	$[1700, 3205]$	$[12, 25]$	$[70, 82]$
23%	$[10, 18]$	$[8, 8]$	$[303, 453]$	$[130, 227]$	$[3134, 5092]$	$[8, 18]$	$[70, 79]$
25%	$[15, 37]$	$[5, 8]$	$[126, 347]$	$[70, 162]$	$[2539, 3993]$	$[11, 22]$	$[70, 82]$

Table 2. Clusters for the "census" data. "Prob." is the mixture probability for the cluster; the remaining attributes are from the input. Numbers are rounded to the nearest integer.

PROB.	AGE	TAXWEIGHT	EDUNUM	CAPITALGAIN	CAPITALLOSS	HOURS	INCOME
14%	$[28, 65]$	$[65863, 319616]$	$[7, 15]$	$[748, 767]$	$[3, 2391]$	$[28, 70]$	$[1, 1]$
11%	$[18, 88]$	$[44770, 998117]$	$[1, 16]$	$[45, 805]$	$[2, 33]$	$[4, 98]$	$[0, 1]$
8%	$[26, 70]$	$[63389, 395858]$	$[9, 16]$	$[73, 99452]$	$[32, 33]$	$[20, 71]$	$[1, 1]$
67%	$[19, 65]$	$[62643, 342083]$	$[4, 14]$	$[45, 787]$	$[4, 1898]$	$[15, 59]$	$[0, 0]$

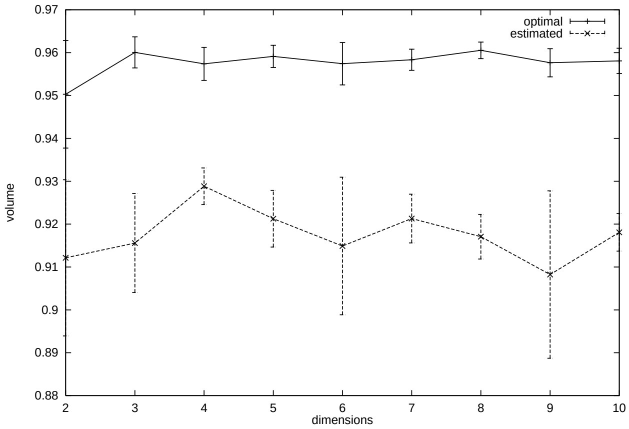


Figure 6. Similarity, by relative volume of intersection, of the estimated distribution to the true one. There were 4 rectangles in the distribution. The "estimated" line shows the performance of the proposed algorithm. The "optimal" line is for a version that initializes the distribution with the true one, therefore providing a theoretical upper bound for the similarity.

resentation which people can find comprehensible. As an added bonus, expanding the Gaussian means into a rectangular form seems to help avoid the inherent statistical problems that come with high-dimensional distance computations. The resulting clusters are highly readable and allow for the construction of very simple, yet very effective, classifiers. The proposed model was also shown to be effective in dimension-reduction tasks.

We started working on an accelerated version of this technique which stores sufficient statistics of subsets of the data in the nodes of a *kd*-tree. Future work can use the anchors hierarchy (Moore, 2000) for better performance in high-dimensional domains.

We believe it would be straightforward to extend this method to data involving discrete attributes. For a given mixture component, each discrete attribute would be modeled independently by a multinomial distribution. The parameters needed for an n-ary attribute would simply be the n multinomial probabilities (which must sum to one). In the case of all-discrete, no real attributes, this would degenerate to the well-known mixtures of products of multinomials distribution (e.g. (Meila & Hackerman, 1998)).

A natural extension would be to get rid of the single parameter that the user needs to supply — the desired number of clusters. Ideally the algorithm would estimate this by itself.

We argue that the representation of clusters as intersection of intervals is succinct. However, there is room for improvement, especially if the data dimensionality is very large. We believe that a post-processing step that identifies the irrelevant dimensions and removes them from the output can be easily implemented. Another useful operation would be to sort the dimensions by their "importance" (or contribution to the log-likelihood) before presenting the intervals.

References

Agrawal, R., Gehrke, J. E., Gunopulos, D., & Raghavan, P. (1998). Automatic subspace clustering of high dimensional data for data mining applications. *Proc. ACM SIGMOD Int. Conf. Management of Data*, *27*, 94–105.

AlSabti, K., Ranka, S., & Singh, V. (1999). An efficient space-partitioning based algorithm for the K-means clustering. *Proceedings of the 3rd Pacific-Asia Conference on Methodologies for Knowledge Discovery and Data Mining (PAKDD-99)* (pp. 355–359). Berlin: Springer.

Bishop, C. M. (1995). *Neural networks for pattern recognition*. Oxford: Clarendon Press.

Blake, C., & Merz, C. (1998). UCI repository of machine learning databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Duda, R. O., & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. John Wiley & Sons.

Fasulo, D. (1999). An analysis of recent work on clustering algorithms. `http://www.cs.washington.edu/homes/dfasulo/clustering.ps`.

Friedman, J. (1998). Bump Hunting in High-Dimensional Data. *NIPS-98*.

Guha, S., Rastogi, R., & Shim, K. (1998). CURE: An efficient clustering algorithm for large databases. *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD-98)* (pp. 73–84). New York: ACM Press.

Liu, B., Xia, Y., & Yu, P. (2000). *Clustering through decision tree construction* (Technical Report RC21695). IBM Research.

Maass, W., & Warmuth, M. K. (1995). Efficient learning with virtual threshold gates. *Proc. 12th International Conference on Machine Learning* (pp. 378–386). Morgan Kaufmann.

Mehta, M., Agrawal, R., & Rissanen, J. (1996). SLIQ: A fast scalable classifier for data mining. *5th Intl. Conf. on Extending Database Technology*.

Meila, M., & Hackerman, D. (1998). *An experimental comparison of several clustering and initilalization methods* (Technical Report 98-06). Microsoft Research, Redmond, WA.

Moore, A. (1998). Very fast EM-based mixture model clustering using multiresolution kd-trees. *Proceedings of Neural Information Processing Systems Conference*.

Moore, A. (2000). The anchors hierarchy: Using the triangle inequality to survive high dimensional data. *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence (UAI-00)* (pp. 397–405). San Francisco, CA: Morgan Kaufmann Publishers.

Nagesh, H., Goil, S., & Choudhary, A. (1999). *MAFIA: Efficient and scalable subspace clustering for very large data sets* (Technical Report 9906-010). Northwestern University.

Ng, R. T., & Han, J. (1994). Efficient and effective clustering methods for spatial data mining. *Proceedings of VLDB*.

Pelleg, D., & Moore, A. (1999). Accelerating exact k-means algorithms with geometric reasoning. *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 277–281). New York, NY: AAAI Press. An extended version is available as Technical Report CMU-CS-00-105.

Pelleg, D., & Moore, A. (2000). X-means: Extending K-means with efficient estimation of the number of clusters. *Proc. 17th International Conf. on Machine Learning* (pp. 727–734). Morgan Kaufmann, San Francisco, CA.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C, 2nd. edition*. Cambridge University Press.

Shafer, J. C., Agrawal, R., & Mehta, M. (1996). SPRINT: A scalable parallel classifier for data mining. *Proc. 22nd Int. Conf. Very Large Databases* (pp. 544–555). Mumbai (Bombay), India: Morgan Kaufmann.

Zhang, T., Ramakrishnan, R., & Livny, M. (1995). BIRCH: An efficient data clustering method for very large databases,. *Proceedings of ACM SIGMOD* (pp. 103–114).