Proceedings of the

Twentieth Annual Symposium of the Pattern Recognition Association of South Africa

30 November – 1 December 2009
Stellenbosch, South Africa

Proceedings of the

Twentieth Annual Symposium of the Pattern Recognition Association of South Africa

30 November – 1 December 2009
Stellenbosch, South Africa

http://www.prasa.org

Edited by: F. Nicolls
ISBN 978-0-7992-2356-9

Member of the International Association of Pattern Recognition (IAPR)

Table of Contents

Full papers

Density Estimation from Local Structure

Christiaan van der Walt and Etienne Barnard

Human Language Technologies Research Group
CSIR, Meraka Institute, Pretoria, South Africa
Department of Electrical, Electronic and Computer Engineering,
University of Pretoria, Pretoria, South Africa
{cvdwalt,ebarnard}@csir.co.za

Abstract

We propose a hyper-ellipsoid clustering algorithm that grows clusters from local structures in a dataset and estimates the underlying geometrical structure of data with a set of hyper-ellipsoids. The clusters are used to estimate a Gaussian Mixture Model (GMM) density function of the data and the log-likelihood scores are compared to the scores of a GMM trained with the expectation maximization (EM) algorithm on 5 real-world classification datasets (from the UCI collection). We show that our approach gives better generalization performance on unseen test sets for 4 of the 5 datasets considered.

1. Introduction

Statistical pattern recognition and data analysis are powerful tools that have gained significant importance in the digital information era. With the dawn of the internet and increase in computing power numerous databases are constructed with increasing size and dimensionality. In order to analyse and extract information from these large high-dimensional databases a better understanding of the intrinsic properties of data is required. The task of data analysis is further complicated by the increasingly wide range of applications to which data analysis is applied; these applications range from technical applications such as automatic speech recognition, computer vision and bioinformatics to general applications such as marketing, politics and even sport.

Intensive research in supervised and unsupervised learning have led to the development of a wide range of techniques for analysing and modelling data; these techniques include methods for tasks such as density estimation, dimensionality reduction, clustering, bi-clustering, topological modelling, Bayesian networks and various others. The wide range of applications to which these techniques are applied requires the analysis of datasets with highly variable data properties which requires a significant amount of expert knowledge in both the problem domain and in data analysis and modelling. More general data analysis and modelling techniques are thus required that are not dependant on the problem domain and that do not require a significant amount of expertise of all available methods in data analysis.

In this paper we propose a general purpose density estimation technique based on hyper-ellipsoid clustering that can be used to model arbitrary datasets. Our density estimation technique is based on general properties of data in high-dimensional spaces; we discuss some of these properties in Section 2 and show how we derived a hyper-ellipsoid clustering algorithm based on these insights in Section 3.

We design and perform a set of experiments in Section 4 to compare the density estimation performance of our clustering approach to the expectation-maximization (EM) algorithm and we conclude on our findings in Section 5.

2. Background

Some insight into the properties of data as dimensionality increases is given in [1], Landgrebe demonstrates that (1) as the dimensionality of a feature space increases, the majority of the volume of the hyper-cube containing the feature space moves to the edges and corners and (2) the majority of the volume of a hyper-sphere moves to the outer shell with increasing dimensionality. The most of the volume of high-dimensional spaces thus tends to move to regions of the feature space far removed from the centroid; nevertheless, data tends to lie in manifolds (which make up small parts of the feature space) with high densities while the remaining part of the feature space is relatively empty. This phenomenon thus suggests that data, specifically in higher dimensional feature spaces, are generated from underlying manifolds with high density.

Another argument along the same lines is given in [2]. If we consider an example of a body suit with N sensors capturing motion in 3 dimensions, we have a feature space of dimensionality 3N. The exact position of a body can actually be specified by k angles between the joints of the body. The intrinsic dimensionality of the problem (k) is significantly smaller than the dimensionality of the feature space (3N). Similarly, Mumford [3] illustrated that high-dimensional natural images can, to a good approximation, be reduced to points on a 7-sphere. These examples suggest that high-dimensional data can be described by underlying manifolds with intrinsic dimensionalities (k) much lower than the dimensionality of the feature space (d).

These properties of data imply that data points are not uniformly distributed throughout the feature space and that data points are concentrated in small parts of the feature space that contribute to most of the density of the data. These sub-feature spaces with concentrated density can be described as underlying manifolds from which data points originate and in order to characterise data, we need to learn and describe the geometrical structures and intrinsic dimensionality of these underlying manifolds [3].

In the next section we propose a density estimation technique based on a hyper-ellipsoidal clustering that approximates the geometrical structure of data points in feature space by starting from local structures (dense regions) in the data and then expanding from these structures to more

general structures (hyper-ellipsoids). This method groups data points into hyper-ellipsoidal sub-feature spaces, such that the properties of the data points in each hyper-ellipsoid are similar.

3. Hyper-ellipsoid clustering

3.1. Preliminary experiments

We first report on a set of investigative experiments on an artificial dataset with known properties, used in the development of our hyper-ellipsoid clustering algorithm . We generate an artificial dataset by sampling data points from 5 bi-variate Gaussian densities; this dataset is illustrated in Figure 1; 50 points are sampled from each Gaussian density. We effectively simulate a dataset sampled from two underlying manifolds, groups 1-3 are part of a manifold and groups 4 -5 are part of another manifold. Note that groups 4 and 5 have identical covariance matrices, and have means relatively close to each other which will make them difficult to distinguish for most clustering algorithms.

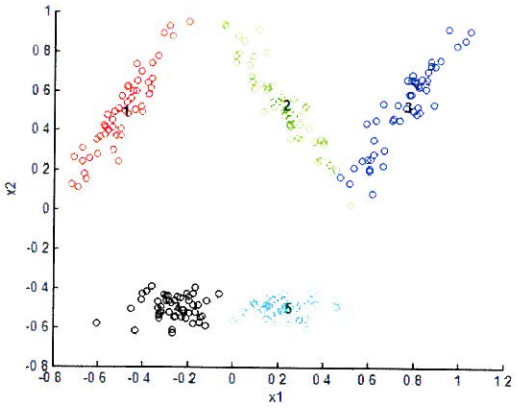

Figure 1: Data sampled from 2-dimensional Gaussian distributions.

We initialise our clustering algorithm by finding the densest region in features space; the motivation for this is that (as explained earlier) the most of the density of datasets tend to lie in small regions of the feature space. We thus try to predict where the clusters in the feature space are by starting clusters from these densest regions (this is a significant difference from the EM algorithm which is usually initialised with a k-means algorithm that is dependant on random initialisation). We calculate the density of each point in the feature space by calculating bivariate histograms between all feature pairs. The density of any point in the dataset is calculated as the sum of the counts of all the bins to which the feature vector belongs.

A hyper-spherical cluster is then grown from the densest region by calculating the Euclidean distances between the densest point and all other points in the dataset. Data points are then added to the cluster by ordering the data point according to Euclidean distance and adding the nearest points. We make use of a geometrical motivation to decide when to stop adding nearest neighbours to the initial group. In Figure 2 we show an elliptical region *E1*, with three circles centred on the middle point of *E1*. If we select any radius smaller than *R*, for example *r1*, the area of the circle created by this radius will always be equal to the intersecting area between the *E1* and the circle (the circle lies completely within *E1*). If any radius larger than *R* is selected, for example *r2*, then the area of the circle created by this radius will always be more than the interesting area between *E1* and this new circle.

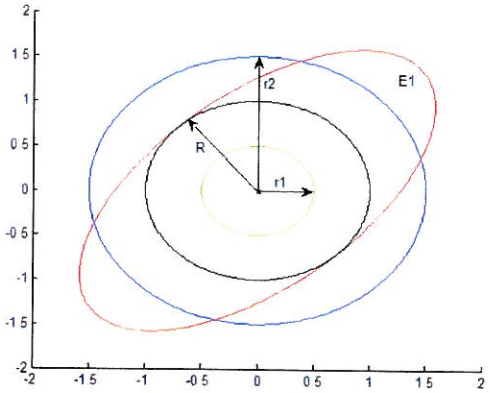

Figure 2: Geometrical motivation for determining Euclidean cut-off point

If data points are uniformly distributed on the ellipse *E1* and if the nearest points to the middle point of *E1* are added in terms of Euclidean distance, then the area of the minimum enclosing ellipse of all the points added up to the k^{th} nearest neighbour will be approximately equal to the area of the circle with radius from the middle point to the k^{th} nearest neighbour when the k^{th} nearest neighbour is closer than *R*. The area of the circle will become larger than the minimum-enclosing ellipse of the k^{th} nearest points once the k^{th} nearest neighbour lies further than *R* from the middle point. The ratio (V_r) between the area of the minimum-enclosing ellipse of the k-nearest-neighbours and the enclosing circle of the k^{th} nearest neighbour will thus drastically decrease if the k^{th} nearest neighbour is further than *R*. We can measure this ratio for every nearest neighbour added to the group and plot a graph that will show a significant decline once the data points in the group are no longer distributed spherically.. The point where the V_r drastically deceases indicates when we should start considering correlation between features, by employing elliptical data structures.

We calculate the volume of a hyper-sphere by first calculating the volume of a unit hyper-sphere as follows:

$$C_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)}, \tag{1}$$

where Γ is the gamma function and d is the dimensionality of the unit hyper-sphere. The volume of a hyper-sphere with radius r is then calculated with

$$V_d = C_d r^d. \tag{2}$$

The minimum volume enclosing ellipse (MVEE) of a d x N dimensional set of points $\mathbf{X}$ can be obtained by solving the following optimization problem:

$$\begin{aligned} &\textit{minimize} && \det(\mathbf{E}^{-1}) \\ &\textit{subject to} && (\mathbf{x}_i - \mathbf{c})^T \mathbf{E} (\mathbf{x}_i - \mathbf{c}) \leq 1, \end{aligned} \tag{3}$$

where $\mathbf{x}_i$ is the i^{th} data point in set $\mathbf{X}$.

The Khachinayan algorithm can be used to solve $\mathbf{E}$ iteratively (a Matlab implementation is available from [5]), given a set of data points $\mathbf{X}$. The volume of the MVEE (ε) can be obtained by using the following equation:

$$V_\varepsilon = v_0 \det(\mathbf{E}^{-1})^{1/2} \tag{4}$$

To illustrate how the volume ratio (V_r) changes as points are added to a group, we generated two types of data. First, we sampled data points from a Gaussian distribution with a spherical covariance matrix. We calculated the densest point in the dataset by making use of a combination of bivariate histograms and then calculated the Euclidean distances between all other points in the group and the densest point. We grew a cluster by adding the nearest point to the densest point, one point at a time, and calculated the volume of the sphere and volume of the minimum-enclosing ellipse for every point added. The change in volume ratio as points are added to the spherical group is shown in Figure 3. We also generated a group of samples from a Gaussian distribution with correlated features and followed the same procedure as in the case of the spherical Gaussian and the volume ratios for the points in this group are shown in Figure 4. (Note that we used a moving average method to smooth the V_r values in Figures 3 and 4.)

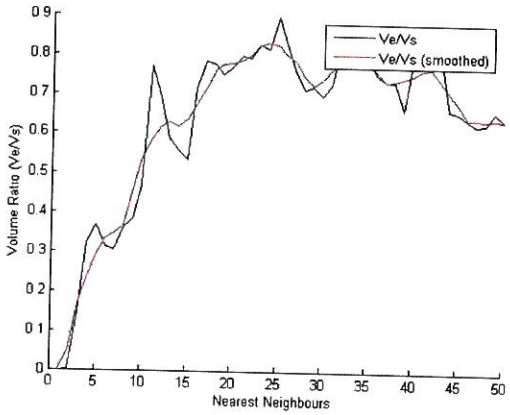

Figure 3: V_r for uncorrelated spherical Gaussian data

We note that the spherical Gaussian group obtains a maximum V_r value at around 25 points and maintains a relatively high V_r value as the remaining data points are added. From 40 points onwards, this value decreases since these points will typically be outliers and will not suit the spherical shape of the data very well. We see in Figure 4 that the decline in V_r values happens significantly earlier (around 25 points) for the correlated Gaussian groups. This point indicates where the data points start deviating from the spherical structure and where correlation should start to be considered.

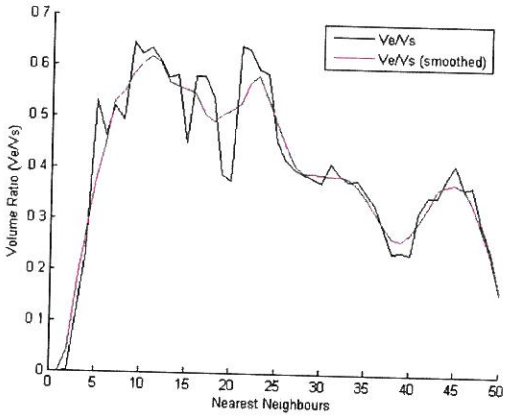

Figure 4: V_r for correlated Gaussian data

To test the effect that other groups would have on the V_r values of a group, we calculated the V_r values for group 5 of the artificial data set in Figure 1 and obtained the results shown in Figure 5.

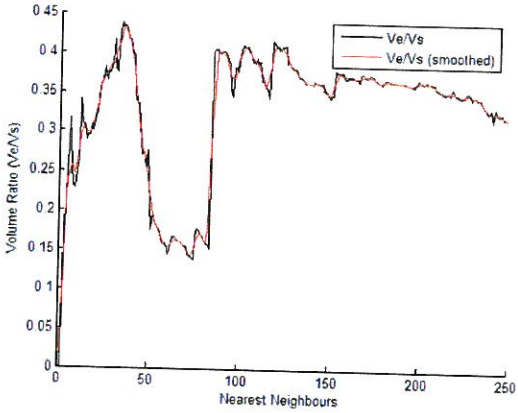

Figure 5: V_r for group 5 in Figure 1

We see that V_r increases up to 40 points and then starts to decrease slightly from 40 to 50 points as outliers of the original group are added, a significant decrease in V_r ratio then takes place from 50-100 points as samples from group 4 are added (note that the decline is gradual since groups 4 and 5 have identical covariance matrices). A drastic increase in V_r takes place between 100 and 110 points. This is when points from group 3 are added – the drastic change in V_r can be attributed to the fact that the covariance matrix and mean of group 3 are very different from those of groups 4 and 5. The increase in V_r is due to the fact that the size of the MVEE increases significantly as data points from the top manifold (groups 1-3) are added to data points from the bottom manifold (groups 4-5). The new structure of the point cloud stabilizes at 110 points and V_r decreases gradually until all points are added to the group. Note that the V_r generally becomes less sensitive as the size of the group increases and that there are no more drastic changes since all the remaining

points (from 110-250) are from the top manifold and do thus not cause any major changes in V_r.

When trying to estimate the cut-off point where Mahalanobis distance will be used instead of Euclidean distance one should be cautious not to use the Mahalanobis distance prematurely. Theoretically the Mahalanobis distance can be used after d (where d is the dimensionality of the feature space) points have been selected with the Euclidean distance. If the Mahalanobis distance is, however, used too early the distance measure will force the group to grow in the direction of the initial covariance estimate. If the covariance estimate of the d points is not representative of the covariance of the entire group, the Mahalanobis distance will continue to grow in the direction of the original d points (thus imposing an artificial structure on the group covariance matrix). We must thus ensure that we have obtained enough data points that represent the covariance of the entire group sufficiently before using the Mahalanobis distance measure.

In order to obtain this cut-off point where enough samples have been selected to represent the covariance matrix of the group sufficiently we take the maximum V_r value (V_{max}) – in the case of Figure 5, the cut-off point will be at around 35 samples. At this point the covariance matrix (obtained from the nearest 35 data points) will be representative of the covariance matrix of the entire group (50 data points).

After the V_{max} cut-off point has been reached, the Mahalanobis distances between the initial group of points and all the remaining points are calculated. Points are then ordered according to Mahalanobis distance and the nearest points are added. We measure the change in covariance of the growing group of data points as samples are added one at a time in order to detect when to stop growing the cluster. If a cluster contains all the points of a certain group and starts adding points of another group, the covariance matrix will change significantly. We make use of the Bhattacharyya distance measure to measures the change in covariance and mean every time a new point is added to the group. The Bhattacharyya distance between two multivariate normal distributions can be expressed as:

$$D_B = \frac{1}{8}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) + \frac{1}{2}\ln\left(\frac{\det(\boldsymbol{\Sigma})}{\sqrt{\det(\boldsymbol{\Sigma}_1)\det(\boldsymbol{\Sigma}_2)}}\right) \tag{5}$$

where $\boldsymbol{\mu}_i$ are the means and $\boldsymbol{\Sigma}_i$ the covariances of the Gaussian distributions and

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2}{2}. \tag{6}$$

The Bhattacharyya distance thus takes the distances between the means and covariance matrices of the two density functions into account.

Figure 6 shows the change in covariance matrix (measured as the difference in Bhattacharyya distance) as the nearest samples are added to the existing group with the Mahalanobis distance.

We see that the Bhattacharyya distance is relatively stable up to 50 points, as points of another group are added the change increases and at around 65 points groups 5 and 4 have merged and the structure converges, at 75 points outliers of these groups are added which gives an increase in change up to 100 points where a drastic increase in Bhattacharyya distance occurs due to data points of group 3 that are being added. After the data points of group 3 have been added, the distance stabilises as the cluster includes all the remaining points in groups 1-3.

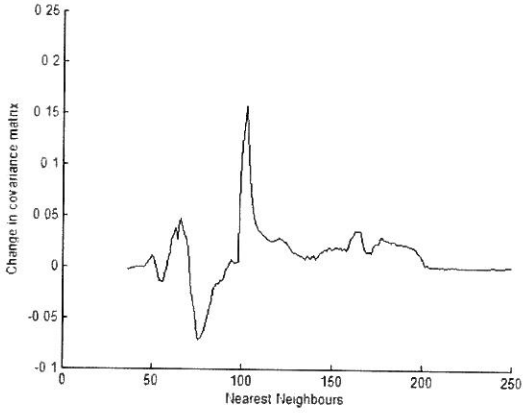

Figure 6: Change in group covariance

In order to find the cut-off point where a drastic change in Bhattacharyya distance takes place, we take the derivative of the change in Bhattacharyya graph. This graph is illustrated in Figure 7.

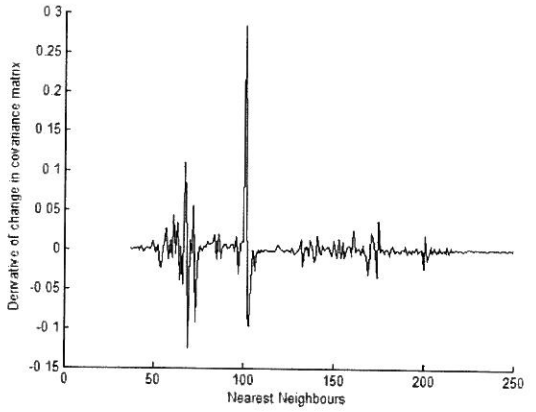

Figure 7: Derivative of change in group covariance

We see that significant changes occur around 70 points (as was seen in Figure 6), with even more significant changes at 100 points. This is where groups 4 and 5 have merged and started to include points from group 3. We also calculate the second derivative of the change in Bhattacharyya distance to amplify the drastic changes even more; the second derivative graph is shown in Figure 8.

We obtain the cut-off point where a cluster should stop growing by calculating the relative change in standard deviation of every point in the second derivative. Thus, for every new point the mean and standard deviation of the previous second derivative values are calculated and the deviation of the new point in standard deviations is calculated. After the entire graph is completed, the point with the highest deviation relative to the previous points is taken as

the cut-off point. The cut-off point in this case is indicated by the red circle in Figure 8. Note that the cut-off point should have been at around 50 points, but since groups 4 and 5 are so similar and close to each other the clusters were merged and the cut-off was found at around 100 points.

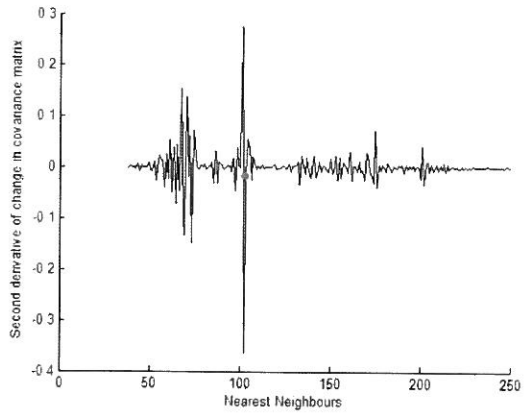

Figure 8: Second derivative of change in group covariance

3.2. Hyper-ellipsoid clustering algorithm

From the insights obtained in Section 3.1, we propose the following hyper-ellipsoid clustering algorithm that finds hard and soft clusters as explained in the previous section:

1. Find the densest point in the dataset by using a sum of bivariate histogram bin counts.
2. Use Euclidean distance measure to add nearest samples until Euclidean cut-off point is reached.
3. Use Mahalanobis distance measure to add the nearest points to existing group until the Mahalanobis cut-off point is reached.
4. Recalculate the densities of the remaining points (not assigned to a group), and start growing a new cluster from the densest point; any point may be assigned to a new group even though that point might have been assigned to a previous group.
5. Continue finding new groups (repeat steps 1-3) until the number of points not assigned to a group yet are less than *d*.
6. Assign the remaining points (<d points) to their nearest group in terms of Mahalanobis distance.
7. Find the overlapping points (points belonging to more than one group) between all the groups and assign each overlapping point to its nearest group in terms of Mahalanobis distance.
8. Perform an iterative k-means procedure with Mahalanobis distance for 10 iterations.

This algorithm contains one free parameter (α) that is used to control the Mahalanobis cut-off point in step 3. The final cut-off point for a group is calculated as follows:

$$N_G = N_E + \alpha(N_M - N_E), \qquad (7)$$

where N_g is the number of points in the final group, N_E is the number of points selected with the Euclidean distance measure and N_M is the number of points (including N_E) included until the Mahalanobis cut-off point was reached. If α=0, all the groups will have N_E samples (thus no points added with the Mahalanobis distance).

The algorithm described above can be used both for hard and soft clustering - the final two steps are additional steps that are required to convert soft clusters to hard clusters.

In order to obtain a density function of the training set, the maximum likelihood (ML) mean and covariance of each group can be calculated and the resulting Gaussian density functions can be summed to give a GMM density estimate of the data.

In the next section we perform a range of experiments to benchmark the accuracy of our GMM density estimation against the EM algorithm.

4. Experiments

4.1. Experimental design

We compare the log-likelihoods of the density estimates obtained with the clustering approach to the log-likelihood scores obtained with the EM approach on 5 real-world UCI classification datasets [6]. These datasets are: Iris, Diabetes, German, Heart, Australian and Balance-Scale.

Each data set is divided into 10 equal folds; the training set is constructed from 8 folds whereas the test and validation sets each consist of a single fold. In order to obtain the optimal number of mixtures for density estimation with the EM algorithm, we train GMM models with mixtures ranging from 1-20 on the training set and calculate their log-likelihood scores on the validation set. The optimal GMM model is then selected (according to the highest log-likelihood of the 20 different models) and the log-likelihood of the unseen test set is then calculated. Each class is treated independently – a separate GMM density function is thus estimated for each class and the final log-likelihoods of the classes are combined (proportional to the number of samples in each class).

In order to find the optimal model for our clustering approach we perform clustering and determine the density estimate of the clusters on the training set for α values ranging from 0 to 1 with increments of 0.1, the log-likelihood scores of the validation set is then calculated for each of these models and the model is selected with the highest log-likelihood. Finally, the log-likelihood of the unseen test set is calculated on the optimal model selected with the validation set.

4.2. Experimental results

Table 1 shows the log-likelihoods obtained with EM density estimation, hyper-ellipsoid hard-clustering (hard) and hyper-ellipsoid soft-clustering (soft) on the unseen test set for each UCI dataset (the total number of groups created per dataset are indicated in brackets).

The results in Table 1 show that both clustering approaches give better GMM density estimates on all the test sets except for balance-scale, where the test-set log-likelihood is approximately equal to the EM algorithm.

The average difference in log-likelihood scores between the validation set (on which parameters were optimised) and unseen test sets of the EM algorithm is 1.6647, whereas the average difference for the hard clustering method is 0.2459 and 0.3342 for soft clustering.

Dataset	EM	Hard	Soft
Iris	-2.205 (27)	-1.868 (7)	-1.855 (7)
Diabetes	-2.974 (34)	-2.965 (10)	-2.902 (8)
Heart	-4.038 (27)	-2.678 (6)	-2.615 (6)
Australian	-4.368 (16)	-3.041 (7)	-2.978 (8)
Balance-s	-2.071 (8)	-2.122 (15)	-2.073 (19)

Table 1: Comparison of log-likelihood scores (test set)

This implies that the clustering approaches have better generalization performance, since the EM algorithm has the lowest log-likelihood scores on all the validation sets, but is outperformed on the test sets by the clustering methods. Overall the soft clustering approach gives the best density estimates on the test sets; the hard clustering approach has a slightly better generalization performance than the soft clustering and the EM seems to overtrain models on the validation sets and has the poorest generalization performance on the unseen test sets.

We also note that in general, the number of groups found by the clustering methods is much smaller than the number of mixtures selected with the EM algorithm. This explains why the EM algorithm has poor generalization performance on the test sets, since the EM algorithm tends to overfit the GMM density on the validation data by selecting too many mixtures. The only case where the EM algorithm has fewer mixtures per class than the number of groups per class obtained with the clustering algorithms is for the Balance-scale dataset. We see in Table 1 that Balance-scale is the only dataset where the EM algorithm outperforms the clustering methods; this good performance on the test set might be attributed to the small numbers of mixtures selected which led to better generalization performance.

5. Conclusions

We have shown that our clustering approach outperforms the EM algorithm with GMM density estimation on 4 of the 5 UCI datasets considered. Our clustering approach generally finds fewer clusters in the data than the number of mixtures found by the EM algorithm, which leads to better generalization performance on unseen test data.

The fact that our algorithm generalizes better than the EM algorithm on unseen test sets suggests that our approach is capturing underlying structure of the data in the estimated density, whereas the EM algorithm obtains an artificial structure with a density function that optimizes the log-likelihood of the data. The clusters obtained with our approach thus give us insight into the structure of the data, whereas the GMM obtained with the EM algorithm may optimize the log-likelihood of the validation set without mapping to a true structure in the data.

An advantage of our clustering approach is that it is not dependant on a random initialization as in the case of the EM algorithm. Our approach is thus not susceptible to local minima – the density function obtained with the EM algorithm, in contrast, depends on the initialization of data.

Another advantage of our approach is that the V_r values and change in covariance values can be plotted for each group in the training stage. These graphs can give an analyst insight into how the groups are constructed which is not possible with the EM algorithm.

Our approach can also be extended to extract the topology of the data by calculating the pair-wise Bhattacharyya distances between all groups obtained, and using multi-dimensional scaling (MDS) to visualize the relative distances between the hyper-elliptical groups in the data. Doing MDS for a GMM obtained with the EM algorithm will not be representative of the true topology of the data, since the mixtures obtained with the EM algorithm are less representative of the structures in the data, as argued above.

One drawback of our approach is that the Mahalanobis distance measure requires at least d points in a group to calculate the ML covariance matrix. In order to obtain stable groups with our approach, there must therefore be significantly more data points than features in a dataset, since the smallest possible group with our approach has at least d points. This shortcoming can, however, be addressed by making use of a Mahalanobis distance measure with diagonal covariance matrix. This ignores the pair-wise correlations between features; however, when the data is that sparse, correlations are in any case not reliably estimated [7].

6. References

[1] L. Jimenez and D. Landgrebe, "Supervised classification in high dimensional space: geometrical, statistical and asymptotical properties of multivariate data", *IEEE Trans. on Systems, Man and Cybernetics*, 28(1):39-54, 1998.

[2] J. Bruske and G. Sommer, "Intrinsic dimensionality estimation with optimally topology preserving maps", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(5):572-575, 1998.

[3] A.B. Lee, K.S Pedersen and D. Mumford, "The nonlinear statistics of high-contrast patches in natural images", *International Journal of Computer Vision*, 54(1):83-103, 2003.

[4] E. Barnard *et al*, "Learning structured representations of data", submitted to the Twentieth Annual Symposium of the Pattern Recognition Association of South Africa.

[5] N. Moshtagh, "Minimum volume enclosing ellipsoids", GRASP Laboratory, University of Pennsylvania. [Online] http://www.mathworks.com/matlabcentral/files/9542/MinVolEllipse.m

[6] C.L. Blake and C.J. Merz, "UCI repository of machine learning databases", 1998. [Online] Available: http://mlearn.ics.uci.edu/MLRepository

[7] R. Clarke *et al*, "The properties of high-dimensional data spaces: implications for exploring gene and protein expression data", *Nat Rev Cancer*, 8(1):37-49, 2008.