Machine Learning Module

Week 8

Lecture Notes 15 & 16

Cluster Analysis

Mark Girolami
girolami@dcs.gla.ac.uk
Department of Computing Science
University of Glasgow

February 27, 2006

1 Cluster Analysis

Given some data composed of feature descriptions of a number of objects a standard question to ask is whether there are natural groupings of the objects such that the objects within each group share to some extent common attributes. In other words do a number of the examples when grouped together share an internal cohesiveness which discriminates them from other groupings?

As a very simple illustration consider objects described by two attributes, when a random sample of these objects are made available then we can visualise the objects in the 2-D attribute or feature space and Figure (1) illustrates the scatter of the objects in attribute space.

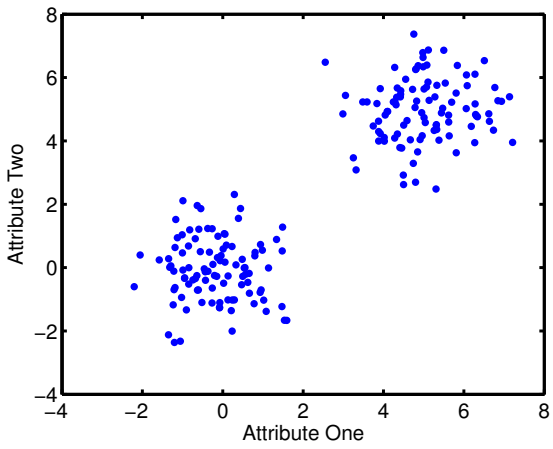

Figure 1: A sample of 200 examples of objects described by two attributes. Each dot represents a sample as defined by attribute 1 & 2, it should be obvious that there appears to be two groupings of objects which each share and internal cohesiveness and are somewhat separated from each of the other groups.

Clearly there are two groupings which, within each group, data points are 'similar' to each other in some sense, and are more similar to each other than they are to objects within the other group or cluster. These groupings may well be meaningful and capture some natural similarity of the objects within each grouping or cluster. For this toy example it is a trivial matter to see that there are clearly two natural groupings within this data and the

clustered nature of the data indicates that there are two self similar clusters of objects. In more complex data representations segmentation of a sample of data into coherent clusters is not such a trivial matter and it is this subject that we focus on this week.

There are a multitude of applications of cluster analysis and a small number of them will be considered here.

1.0.1 Bioinformatics

Applications of cluster analysis within Bioinformatics are rather extensive and ont of the more familiar applications is the production of a phylogenetic tree based on the similarities of gene or protein sequences. Other applications which have become prominent are the clustering of genes or samples within microarray studies.

1.0.2 Information Retrieval

Cluster analysis within Information Retrieval has a very long history and the notion of grouping documents within a collection which share the same topical content has been the subject of many research papers and applications.

2 Algorithms for Cluster Analysis

There are a number (a large number actually) of methods available to perform cluster analysis but the one which we will be introduced to this week is the K-means algorithm.

2.1 K-Means Clustering Algorithm

Let us define our data points $\mathbf{x}_n \in \mathbb{R}^D$ where in a given sample n will range from $1 \cdots N$. We make our first assumption that there are at most K possible groupings or clusters within the data. Of course what leads us to make this assertion will be application specific but clearly by now you should be able to see that this value K is one over which we would wish to do inference. However for the simplest of algorithms it is considered that K - the number of clusters - is given *a priori*.

We will also assume that there are a number of binary indicator variables associated with each data point and cluster z_{kn} which takes on the value

1 if $\mathbf{x}_n$ has been assigned to cluster k and zero otherwise. You should see similarities here between the problem of estimating the probability density of data using a mixture model and cluster analysis. Cluster analysis is in many senses a less complex problem to solve as we are simply assigning data points to clusters rather than trying to learn the actual form of probability density.

2.1.1 Measure of Cluster Quality

Now we need to define a measure which will indicate the quality of the clustering. Once this is defined it can be employed in guiding the selection of the optimal clustering of our data.

We would be more convinced that a cluster which had been identified was not some spurious random artefact if the internal cohesiveness of the points allocated to the cluster was high. A simple way of defining this is to measure how close each of the cluster points is to the cluster average. If all the points are tightly surrounding the mean then the cluster is highly cohesive. On the other hand if the data points are diffusely spread around the cluster centre then our confidence in the cluster actually representing a compact grouping of self-similar points will be low.

Given this we can define a measure of cluster compactness as the total distance from the cluster mean in other words

$$\sum_{\mathbf{x}_n \in \mathcal{C}_k} ||\mathbf{x}_n - \mathbf{m}_k||^2 = \sum_{n=1}^{N} z_{kn} ||\mathbf{x}_n - \mathbf{m}_k||^2$$

where the cluster mean is defined as

$$\mathbf{m}_k = \frac{1}{N_k} \sum_{\mathbf{x}_n \in \mathcal{C}_k} \mathbf{x}_n$$

and $N_k = \sum_{n=1}^{N} z_{kn}$ is the total number of points allocated to cluster K. The total goodness of the clustering will then be based on the sum of the cluster compactness measures for each of the K clusters. Using the indicator variables z_{kn} then we can define the overall cluster goodness as

$$\mathcal{E}_K = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{kn} ||\mathbf{x}_n - \mathbf{m}_k||^2$$

So we have our overall measure of cluster quality the next step is to devise an algorithm which will allow us to optimise this.

2.1.2 Iterative Criterion Optimisation Algorithm

The criterion which we have to optimise is

$$\mathcal{E}_K = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{kn} ||\mathbf{x}_n - \mathbf{m}_k||^2$$

there are two sets of parameters - the cluster mean values $\mathbf{m}_k$ and the cluster allocation indicator variables z_{kn}.

In a similar spirit to the EM algorithm (Week 6 lectures on density estimation) we will optimise our criterion over each set of variables by holding one set fixed whilst performing optimisation over the other set.

So given our current indicator variables z_{kn} the optimal value of the mean vectors $\mathbf{m}_k$ simply boils down to the estimates based on the data points allocated to each cluster. Convince yourself that this is the case by differentiating the cluster criterion with respect to each $\mathbf{m}_k$ and solving.

Therefore given each z_{kn} we obtain our K-means by

$$\mathbf{m}_k = \frac{\sum_{n=1}^{N} z_{kn} \mathbf{x}_{kn}}{\sum_{n'=1}^{N} z_{kn'}}$$

Now give each of our new $\mathbf{m}_k$ we need to update the values of our indicator values z_{kn}.

From the expression for $\mathcal{E}_K$ we can see that each $\mathbf{x}_n$ should be assigned to the cluster k for which it has the shortest distance to the cluster centre i.e. $||\mathbf{x}_n - \mathbf{m}_k||^2$ is the smallest. This is then just a simple matter of finding the centre with the minimum distance to the data point under question and setting the associated indicator variable to this cluster value i.e. $z_{kn} = 1$ for k which yields the minimum of $||\mathbf{x}_n - \mathbf{m}_k||^2$.

Once these values have been redefined then we can go back and revise our estimates of each $\mathbf{m}_k$ and continue this iteration until $\mathcal{E}_K$ converges to some steady value.

This is very simple algorithm and is the K-Means Clustering algorithm for which a simple Matlab implementation is included here. Note that a fully commented version is available for download form the class website.

```
function [M,j,e] = kmeans(X,K,Max_Its)

[N,D]=size(X);
I=randperm(N);

M=X(I(1:K),:);
Mo = M;

for n=1:Max_Its

    for k=1:K
        Dist(:,k) = sum((X - repmat(M(k,:),N,1)).^2,2)';
    end

    [i,j]=min(Dist,[],2);

    for k=1:K
        if size(find(j==k))>0
            M(k,:) = mean(X(find(j==k),:));
        end
    end

    Z = zeros(N,K);
    for m=1:N
        Z(m,j(m)) = 1;
    end

    e = sum(sum(Z.*Dist)./N);
    fprintf('%d Error = %f\n', n, e);
    Mo = M;
end
```

2.2 Illustrative Example of K-Means Clustering

The image of a 'wee dog' looking out to sea is given in Figure (2). The image can be downloaded from the class website as usual. The image is a small 100 $\times$ 100 colour thumbnail and we can represent each pixel in the image as a three-dimensional vector corresponding to the Red, Green & Blue channels of the JPEG image. If we were to attempt to segment the image into self consistent regions corresponding to the background or foreground (i.e. the dog) then we need to cluster the pixels together based on their Red, Green & Blue representations.

Using the K-means algorithm as previously described and setting the number of clusters to $K = 4$ which would correspond to the dog, the road, the grass and the water the segmentation of the image obtained is given in Figure (2). Four colours have been used to represent the clustered regions and it seems that a sensible segmentation of the image has been achieved. See the Laboratory sheet regarding this example and the code `wk8_demo_1.m`.

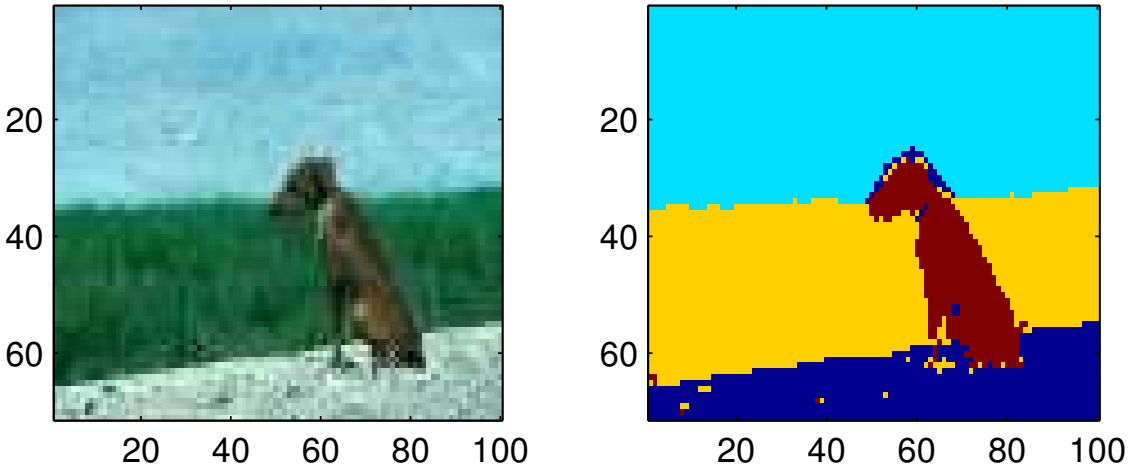

Figure 2: The image of a dog looking out to sea, the right hand image shows the areas of the original image which have been allocated to one of four possible clusters. We have managed to segment the image based on the regions corresponding to water, grass, road, dog

This simple example illustrates how a very simple algorithm such as the K-Means Clustering method can be employed at segmenting a data set. Of

course there are two major shortcomings of this method, the first being that the algorithm will converge to different optima of $\mathcal{E}_K$ depending on the intialisation of the clusters. Secondly the number of clusters has to be defined beforehand and it would be desirable to have an algorithm which would select the value of K based on the data.

There is one other problem associated with the K-Means algorithm which can be demonstrated with an example using the code `wk8_demo_2.m`. Assume that our data has cluster structure which is highly nonlinear such that there are strong nonlinear dependencies within our features, see Figure (3).

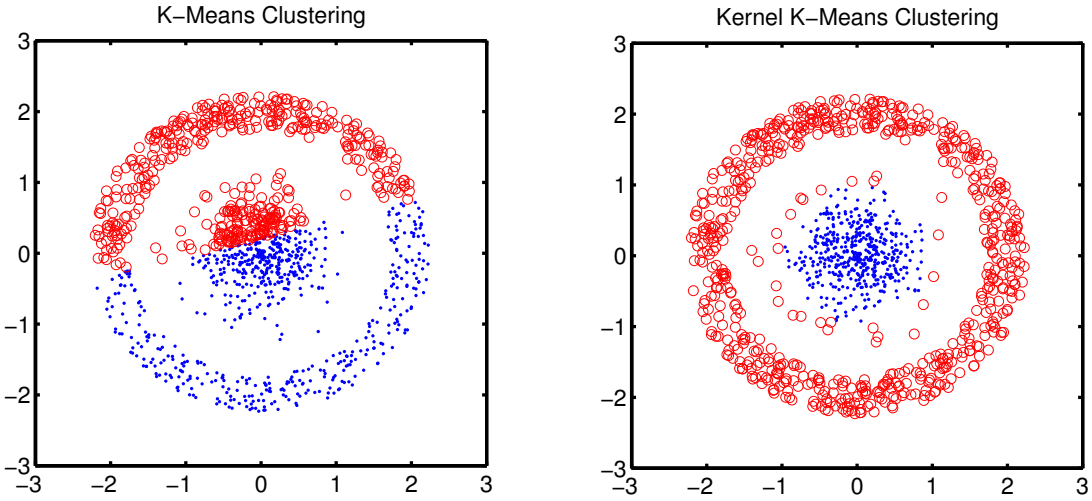

Figure 3: The data is generated such that two consistent clusters both share the same mean but are distributed as a Gaussian cloud and uniformly within a unit width annulus centered at the origin. The left hand plot shows the clustering using the standard K-Means algorithm. It fails to obtain a reasonable clustering. The right hand plot shows the clustering obtained by using Kernel K-Means clustering. A more sensible segmentation of the data is obtained.

The data is such that one grouping is centered at the origin and is clumped as a radially symmetric Gaussian. The second grouping is also centered at the origin but forms an annular ring of width one unit.

There clearly are two clusters which the data could sensibly be allocated to, however running K-Means on this data produces a clustering as shown in the left-hand side of Figure(3) which clearly is not appropriate.

What is the problem? K-Means is based on Euclidean distances and so will segment the data space employing linear hyper-planes and so will fail if nonlinear separation boundaries are required. We can get round this problem by employing the kernel trick introduced back in Week 5 and this is what we will now do.

3 Kernel K-Means Clustering

The clustering criterion upon which the K-Means algorithm is based is simply

$$\mathcal{E}_K = \sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn}||\mathbf{x}_n - \mathbf{m}_k||^2$$

Now this can be written as follows

$$\begin{aligned}
\mathcal{E}_K &= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn}||\mathbf{x}_n - \mathbf{m}_k||^2 \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn}(\mathbf{x}_n - \mathbf{m}_k)^\mathsf{T}(\mathbf{x}_n - \mathbf{m}_k) \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn}\left(\mathbf{x}_n^\mathsf{T}\mathbf{x}_n - 2\mathbf{m}_k^\mathsf{T}\mathbf{x}_n + \mathbf{m}_k^\mathsf{T}\mathbf{m}_k\right)
\end{aligned}$$

Note that

$$\mathbf{m}_k^\mathsf{T}\mathbf{x}_n = \frac{1}{N_k}\sum_{m=1}^{N} z_{km}\mathbf{x}_m^\mathsf{T}\mathbf{x}_n$$

and

$$\begin{aligned}
\mathbf{m}_k^\mathsf{T}\mathbf{m}_k &= \left(\frac{1}{N_k}\sum_{p=1}^{N} z_{kp}\mathbf{x}_p\right)^2 \\
&= \frac{1}{N_k^2}\sum_{p=1}^{N}\sum_{l=1}^{N} z_{kp}z_{kl}\mathbf{x}_p^\mathsf{T}\mathbf{x}_l
\end{aligned}$$

It should be obvious that each of the elements of $\mathcal{E}_k$ can be written in terms of an inner-product and as such if we were to map our data into another

feature space defined by a function ϕ (refer to the lecture notes for Week 5) then by taking the above terms and plugging them together we can write the clustering criterion in the feature space defined by ϕ as $\mathcal{E}_K^\phi$ where the mean vectors in this space are denoted by $\mathbf{m}_k^\phi$.

$$\begin{aligned}
\mathcal{E}_K^\phi &= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn}||\phi(\mathbf{x}_n) - \mathbf{m}_k^\phi||^2 \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn} \left(\begin{array}{c} \phi(\mathbf{x}_n)^\mathsf{T}\phi(\mathbf{x}_n) - \\ \frac{2}{N_k}\sum_{m=1}^{N} z_{km}\phi(\mathbf{x}_m)^\mathsf{T}\phi(\mathbf{x}_n) + \\ \frac{1}{N_k^2}\sum_{p=1}^{N}\sum_{l=1}^{N} z_{kp}z_{kl}\phi(\mathbf{x}_p)^\mathsf{T}\phi(\mathbf{x}_l) \end{array} \right) \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn} \left(\begin{array}{c} K(\mathbf{x}_n, \mathbf{x}_n) - \\ \frac{2}{N_k}\sum_{m=1}^{N} z_{km}K(\mathbf{x}_m, \mathbf{x}_n) \\ \frac{1}{N_k^2}\sum_{p=1}^{N}\sum_{l=1}^{N} z_{kp}z_{kl}K(\mathbf{x}_p, \mathbf{x}_l) \end{array} \right) \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{kn}\delta_{kn}
\end{aligned}$$

where the distance between the n'th data point and the k'th cluster mean, within the feature space, is denoted as δ_{kn} where

$$\begin{aligned}
\delta_{kn} &= ||\phi(\mathbf{x}_n) - \mathbf{m}_k^\phi||^2 \\
&= \left(\begin{array}{c} K(\mathbf{x}_n, \mathbf{x}_n) - \\ \frac{2}{N_k}\sum_{m=1}^{N} z_{km}K(\mathbf{x}_m, \mathbf{x}_n) \\ \frac{1}{N_k^2}\sum_{p=1}^{N}\sum_{l=1}^{N} z_{kp}z_{kl}K(\mathbf{x}_p, \mathbf{x}_l) \end{array} \right)
\end{aligned}$$

The first point to notice is that the clustering criterion can be written solely in terms of the kernel functions computed at each of the data point pairs and so an algorithm can be developed which only requires the step of updating the indicator variables z_{kn} as no explicit updating of cluster mean values of required.

An implementation of Kernel K-means clustering is given over the page and a fully commented version is available at the course website.

```
function [z,e] = kernel_kmeans(X,K,Max_Its,kwidth)
%This is a simple implementation of Kernel K-means clustering - an
%interesting paper which proposed kernel based Kmeans clustering is [1]
%Girolami, M, Mercer Kernel-Based Clustering in Feature Space,
%IEEE Trans Neural Networks, 13(3),780 - 784, 2002.

[N,D]=size(X);
C = kernel_func(X,X,'gauss',kwidth,1);

Z = zeros(N,K); for n = 1:N
  Z(n,rand_int(K)) = 1;
end

for its = 1:Max_Its

    for k=1:K
        Nk = sum(Z(:,k));
        Y(:,k) = diag(C) - 2*C*Z(:,k)./Nk + Z(:,k)'*C*Z(:,k)./(Nk^2);
    end

    [i,j]=min(Y,[],2);

    Z = zeros(N,K);
    for n=1:N
        Z(n,j(n)) = 1;
    end
    e = sum(sum(Z.*Y))./N;
    fprintf('%d Error = %f\n', its, e);
end

for n=1:N
    z(n) = find(Z(n,:));
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%this is a little utility function which returns
%a random integer between 1 & Max_Int.
function u = rand_int(Max_Int)
u=ceil(Max_Int*rand);
```

Now we have a kernel-based clustering method which will allow us to segment our data in a nonlinear manner as can now be seen from the right hand plot of Figure(3). Well this is good news, we have generalised our K-means algorithm to a more flexible representation which takes account of nonlinear relationships within our data. There is, of course, a small price to pay for this flexibility. The kernel function used may well have a parameter (or a number of parameters) of its own - which will need to be chosen in some way - so we have added an additional layer of parameters into our representation.

This weeks laboratory session will explore these two forms of clustering in some detail.

4 EM & K-Means Clustering

In Week 6 we developed an EM algorithm for a Gaussian Mixture model. Lets have another look at this algorithm and make some simplifying assumptions. Firstly lets assume that the covariance for each mixture component is simply an identity matrix which is fixed. There is no need to estimate the covariance matrices as these are set. Now lets think of the posterior probabilities of data points being allocated to a Gaussian component. In this case (where each $\mathbf{\Sigma_k}$ is an identity then in the E-step each

$$E\{z_{kn}\} = P(k|\mathbf{x}_n) \propto \exp\left(-\frac{1}{2}||\mathbf{x}_n - \mathbf{m}_k||^2\right)$$

and the M-step boils down to

$$\mathbf{m}_k = \frac{\sum_{n=1}^{N} P(k|\mathbf{x}_n)\mathbf{x}_n}{\sum_{m=1}^{N} P(k|\mathbf{x}_n)}$$

If we make a hard decision about the expected value of z_{kn} based on the maximum of posterior we should be able to see that the maximum posterior corresponds to the minimum of $||\mathbf{x}_n - \mathbf{m}_k||^2$ which is exactly what we are doing in K-means. So if we choose z_{kn} based on the maximum posterior our M-step is precisely the cluster centre updates for K-means clustering.

K-Means clustering can be obtained directly from the EM algorithm from a mixture of unit radius spherical Gaussians where at the E-step a hard decision about component membership is made.