6.867 Machine Learning

Problem Set 5 Solutions

Due date: Monday November 22

Problem 1: Model Selection

1.

$$P(y_1,\ldots,y_n|\mathbf{x}_1,\ldots,\mathbf{x}_n,\mathrm{PST}) = \int_{[0,1]^K}\left[\prod_{i=1}^{K} p_i^{n_i^+}(1-p_i)^{n_i^-}\right] dp_1 dp_2 \ldots dp_K =$$
$$= \prod_{i=1}^{K}\left[\int_0^1 p_i^{n_i^+}(1-p_i)^{n_i^-}\, dp_i\right] = \prod_{i=1}^{K}\frac{n_i^+!\, n_i^-!}{(n_i+1)!}$$

2.

$$\sum_{i=1}^{K}\log \hat{p}_i^{n_i^+}(1-\hat{p}_i)^{n_i^-} - \frac{d}{2}\log n = \sum_{i=1}^{K}\left(n_i^+\log n_i^+ + n_i^-\log n_i^- - n_i\log n_i\right) - \frac{d}{2}\log n$$

3. Possible implementation of `log_bayesian_model_score`:

```
% You can assume n_plus and n_minus are never 0 together
%
function score = log_bayesian_model_score(n_plus, n_minus)

% total counts for each node
n = n_plus + n_minus;

%
% SCORE COMPUTATION
%
score = sum(log_fact(n_plus) + log_fact(n_minus) - log_fact(n+1));

% ------------------------------
function [logfac] = log_fact(n)

logfac = gammaln(n+1);
```

Possible implementation of `bic_model_score`:

```
% You can assume n_plus and n_minus are never 0 together
%
function score = bic_model_score(n_plus, n_minus)

K = length(n_plus);
n = n_plus + n_minus;

% maximum likelihood probabilities
Pml = n_plus./n;
Pml(n_plus == 0) = eps;
Pml(n_minus == 0) = 1 - eps;

%
% SCORE COMPUTATION
%
score = sum(n_plus .* log(Pml) + n_minus .* log(1 - Pml)) - 0.5*K*log(sum(n));
```

The plots of the resulting PST's are shown in Figure 1.

4. The generated plot is shown in Figure 2. We can interpret the properties of the plot in the following manner:

   – The curves are negative for small sample sizes because the simpler models will always be preferred if data is not sufficient.

   – The curve of the Bayesian score becames positive for smaller sample size because in this task the BIC score tends to prefer simpler models than the BIC score. In general this may not always be the case, because Bayesian selection depends on the prior, while BIC does not.

   – At large sample sizes the two curves converge to each other because the BIC score is a large-sample approximation of the log-Bayesian score (up to an additive constant that cancels out when taking the difference of scores).

   – At large sample size the log-likelihood of the data at the ML parameters becomes linear in the number of samples, because the ML parameters converge to their asymptotic value. Since the BIC complexity penalty is sublinear, it follows that at large sample size the BIC score becomes linear. Thus the difference in BIC scores for two models will also be asymptotically linear. The slope of the difference is positive or negative depending on which of the two models is more complex (in this case positive).

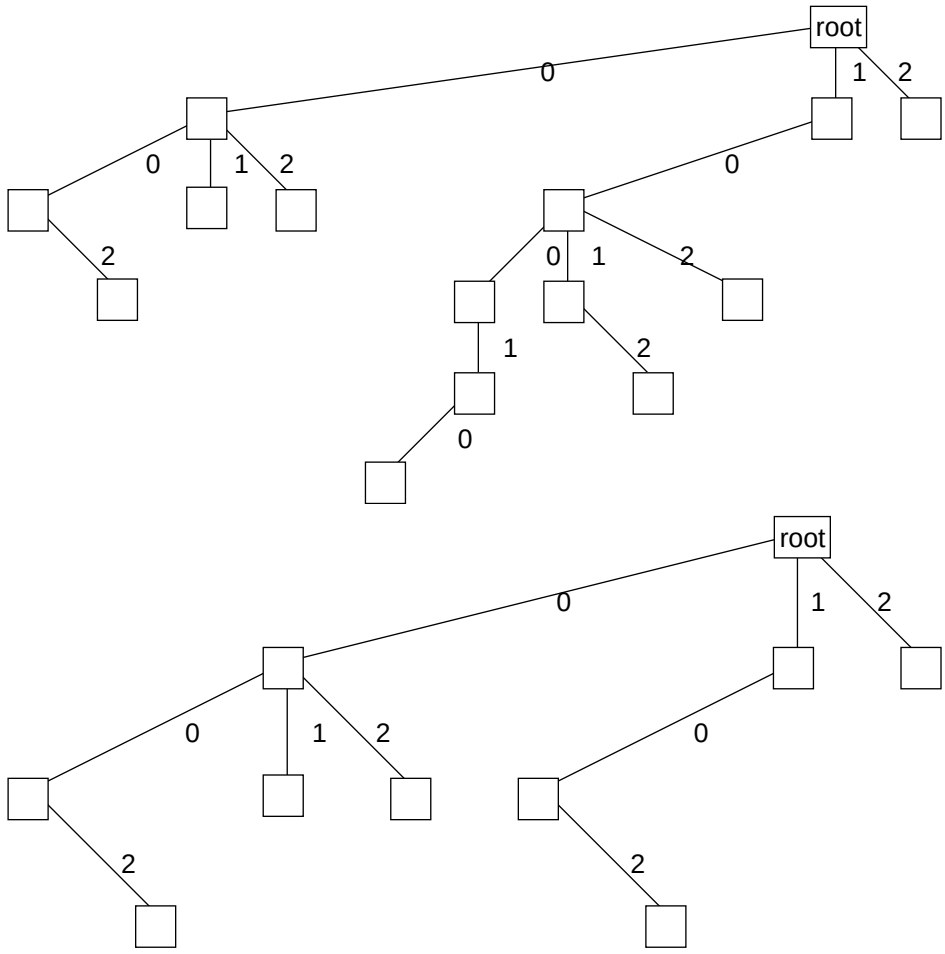


Figure 1: Probelm 1 Part 3: PST's trained with Bayesian model selection (above) and the BIC score (below). Note that the second tree is a subset of the first tree.

Problem 2

1. **Solution:** The hidden variables for a training point x_i are j_i and k_i.

$$p(j_i, k_i | x_i, \theta^c) = \frac{p(x_i | j_i, k_i, \theta^c) p(j_i, k_i | \theta^c)}{p(x_i, \theta^c)} \quad (1)$$

$$= \frac{p^c_{j_i} q^c_{k_i} N(x_i; \mu^c_{j_i}, {\sigma^c_{k_i}}^2)}{\sum_{j=1}^{m} \sum_{k=1}^{l} p^c_j q^c_k N(x_i; \mu^c_j, \sigma^{c\,2}_k)} \quad (2)$$

■

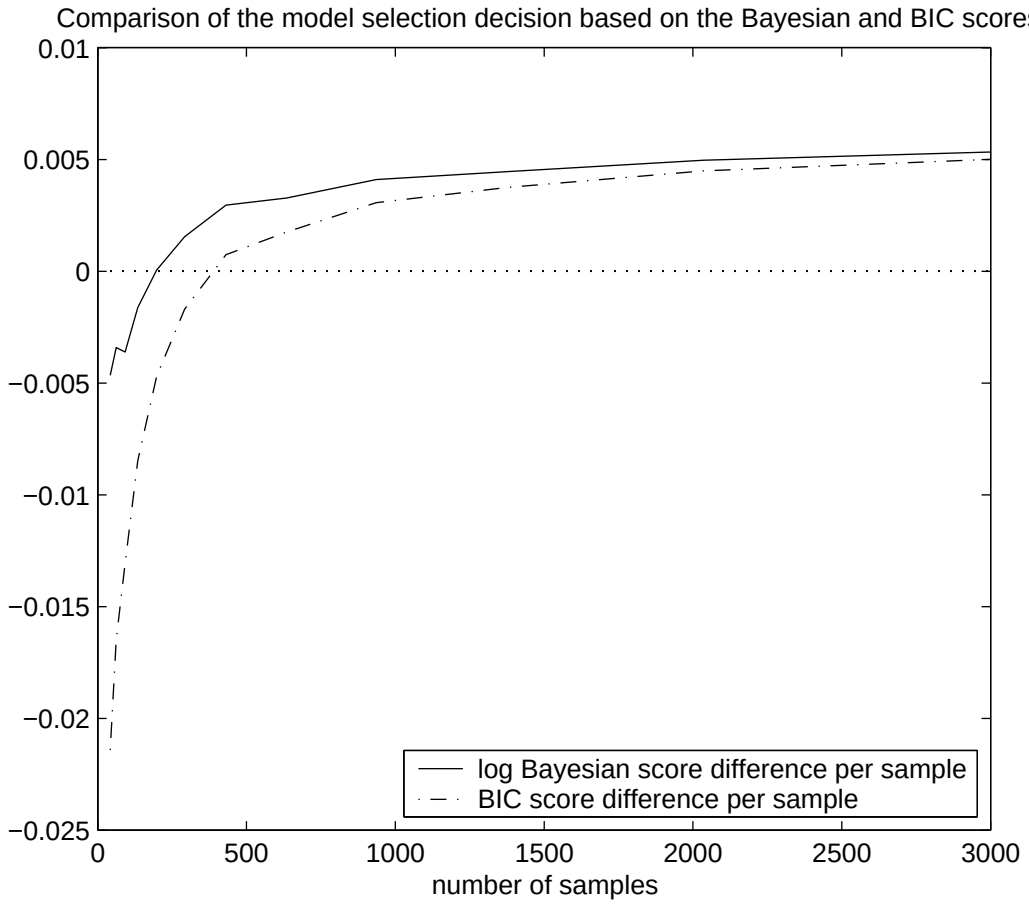


Figure 2: Problem 1 Part 4: Comparison of the log Bayesian and BIC scores on two models

2. **Solution:** The expression for the expected complete log-likelihood (ECLL) is:

$$\begin{aligned} \text{ECLL} &= \sum_{i=1}^{n} E\left[\log(p_{j_i} q_{k_i} N(x_i; \mu_{j_i}, {\sigma_{k_i}}^2))|x_i, \theta^c\right] &(3)\\ &= \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} p(j,k|x_i,\theta^c) log(p_j q_k N(x_i;\mu_j,\sigma_k^2)) &(4)\end{aligned}$$

The terms in the ECLL depending on the mean parameters are therefore:

$$\text{ECLL}(\mu) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} p(j,k|x_i,\theta^c)\left(\frac{1}{2\sigma_k^2}(x_i-\mu_j)^2\right) \quad (5)$$

■

3. **Solution:** Differentiating w.r.t. μ_j,

$$\sum_{i=1}^{n}\sum_{k=1}^{l} p(j,k|x_i,\theta^c)\frac{(x_i-\mu_j)^2}{2\sigma_k^2} = 0 \tag{6}$$

$$\Rightarrow \mu_j = \frac{\sum_{i=1}^{n}\sum_{k=1}^{l} p(j,k|x_i,\theta^c)(x_i/(\sigma_k^2))}{\sum_{i=1}^{n}\sum_{k=1}^{l} p(j,k|x_i,\theta^c)/(\sigma_k^2)} \tag{7}$$

■

4. **Solution:** We plotted the maximum likelihood solutions after 4 runs of the EM algorithm for each setting of m and l. See Figure 3 for the plots.

   A model where components can have two different variance parameters cannot be sufficiently well approximated by a model consisting of a small number of components where all components have a single variance parameter. In other words, the number of components with a single common variance required to approximate the underlying two-variance distribution accurately (using Parzen density estimation, for example) is much larger than 4. ■

Problem 3

1. **Solution:** For the data in `X1.mat`, the typical k-means solution is shown in Figure 4. This clearly does not correspond to the intuitive notion of clustering, where the clusters should be the inner and outer rings. The reason for this is that the clusters found by k-means are a partition (tessellation) of space into polygonal regions separated by straight line-segments. For $k = 2$, the two clusters must be separated by a straight line.

   For `X2.mat`, the typical k-means solution is shown in Figure 5. However this is not the only solution possible, since the initialisation of the means is random. Other solutions, such as those shown in Figure 6 occur about once in seven trials.

   The typical solution here does correspond to the intuitive notion of clusters: groups of closely spaced points, where distance between groups are typically larger than distances within the group. The two point in the top-left part of the space would be considered outliers, since there are many more points in the 'clusters' in the bottom-right part. The k-means algorithm assigns these two points to one or the other cluster seemingly randomly.

   ■

2. **Solution:** See Figures 7 and 8 for clustering results with data in `X1.mat`, and Figures 9 and 10 for corresponding results with `X2.mat`.

The spectral clustering algorithm discussed in class has two parameters: the number of nearest neighbours to which each point is connected, r, and the exponential decay parameter, β.

First let us consider the data in `X1.mat`. When $r = 5$, the corresponding graph is not connected, but consists of two subgraphs (each of which is connected). In this case, the largest eigenvalue calculated in the spectral clustering solution is not unique, but has multiplicity two. Thus, the first and second eigenvectors are equivalent, and both of them appear in the expression for the asymptotic transition probability matrix, P^{∞}. Further, neither of these vectors has all elements equal, nor does either correspond to the first order correction term to P^{∞}. So looking at the sign of the elements of the second eigenvector is practically meaningless. In particular, the first two eigenvectors have zero elements corresponding to points in one of the two clusters whenever the graph is disconnected. This can be seen from Figure 11 in the present case. Thus, the resulting clusters depend on how Matlab chooses to interpret the zero elements, as a consequence of finite precision arithmetic. For instance, running the same code with $r = 4$ gives different results, with the inner ring containing both red and blue points (see Figure 12). With $r = 2$, all the points—both inner and outer rings—are considered to be belonging to the same cluster.

However, there is a second way of interpreting the eigenvectors of the normalised affinity matrix. When the affinities (or distances) within each cluster are all constant, and the distances between points belonging to different clusters are also (some other) constant, the two eigenvectors are piecewise constant (i.e. the elements of the eigenvectors are constant for all indices corresponding to a distinct cluster). Hence, clustering the elements of these two eigenvectors (using k-means clustering) provides us the indices of the points belonging to clusters in the original space.

Even if the affinity matrix is not exactly block constant, the eigenvector elements are still approximately piecewise constant as long as the cluster structure is clearly present in the data. This can be seen from both the first and second eigenvectors in the present case. For instance, the elements of the latter tend to cluster around 0 and -0.16, and can easily be separated by k-means clustering.

When $r = 20$, the graph becomes connected. Thus, the spectral clustering solution is a valid one. However, for the default value of the exponential decay parameter ($\beta = 1$), we find that the clustering does not separate the rings. This is because there are now many connections between the inner and outer rings, and the solution found is essentially the same as that using k-means clustering on the data.

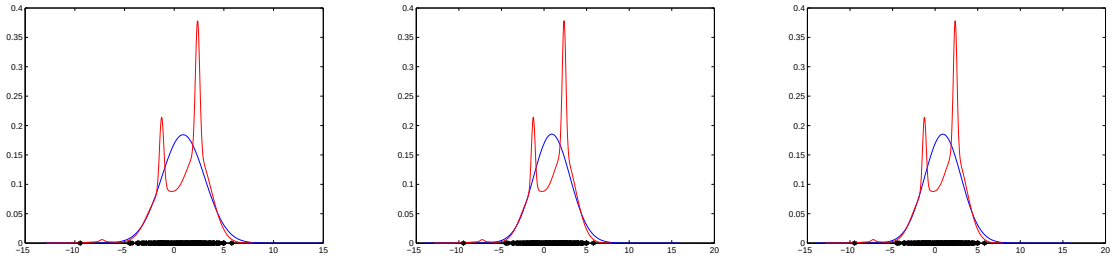
The 'correct' solution—the one separating the inner and outer rings—can be obtained by increasing β to 5, which makes longer distance transitions in the random walk more unlikely (see Figure 13).

Next, consider the data in `X2.mat`. Here, when $r = 5$, the corresponding graph is connected, by virtue of the two outlying points. However, even though connections within each cluster are strong, there are no direct connections between points in the

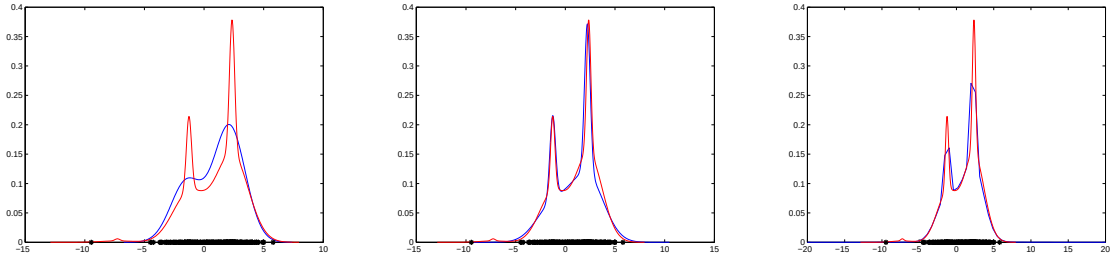
two clusters. Thus, the spectral clustering solution is the one we expect best explains the underlying data.

We have plotted the second eigenvector for this spectral clustering solution in Figure 14. As explained above, there are two clearly identifiable clusters of eigenvector elements (around 0.1 and -0.1), due to tight clustering evident from the data. However, note the two eigenvector elements with values near 0. These indicate the possible presence of a third cluster, and correspond to the two outlying points.

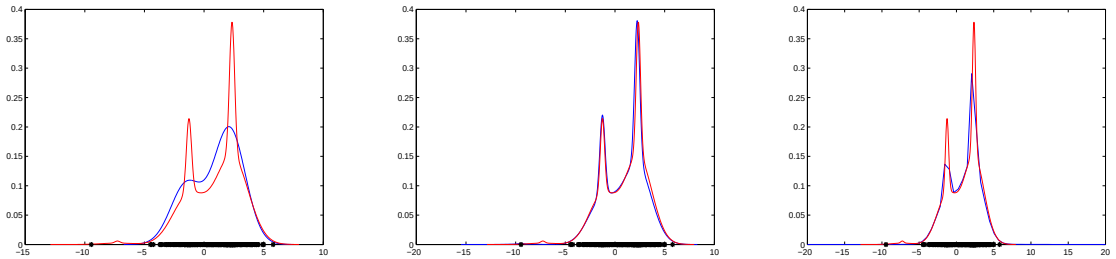
When $r = 20$, we find new connections between points in distinct clusters. Now, the random walk can start at a point in one cluster and end at a point in the other without having to transition through the outlying points. In fact, since the outlying points lie far away from the two clusters, they are connected to the latter by edges with very small weights. In contrast, the weights associated with the edges directly connecting the two underlying clusters are very large. Thus, spectral clustering effectively groups the two clusters into one, and labels the two outlying points as the other cluster. ■



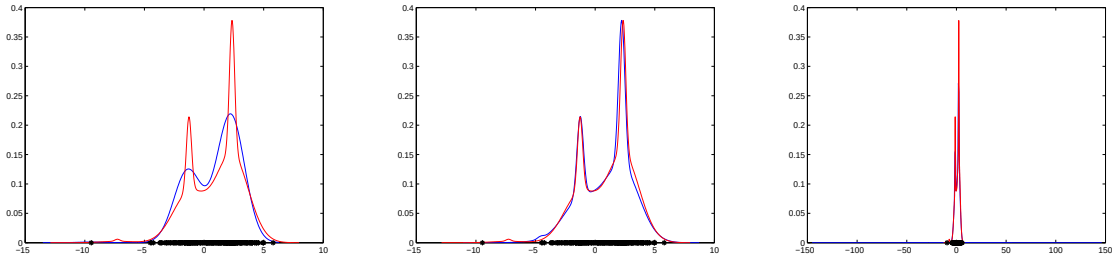
m=1, l=1 m=1, l=2 m=1, l=3



m=2, l=1 m=2, l=2 m=2, l=3



m=3, l=1 m=3, l=2 m=3, l=3



m=4, l=1 m=4, l=2 m=4, l=3

Figure 3: Samples, estimated densities and true densities, for different values of l and m

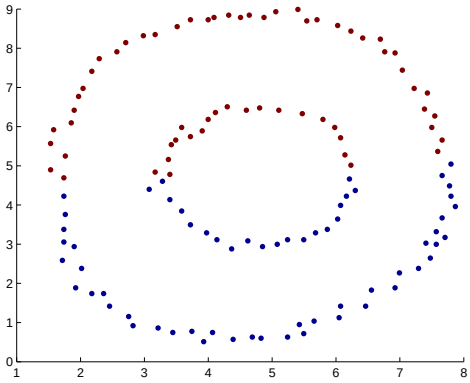


Figure 4: Typical clusters found by k-means ($k = 2$) for data in `X1.mat`

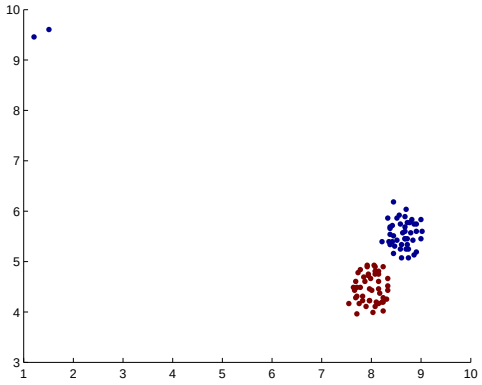


Figure 5: Typical clusters found by k-means ($k = 2$) for data in `X2.mat`

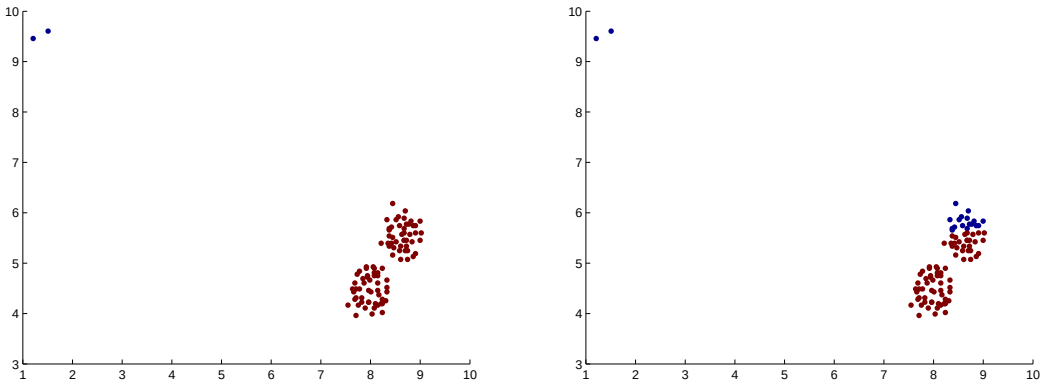


Figure 6: Other (less typical) clustering solutions found by k-means ($k = 2$) for `X2.mat`

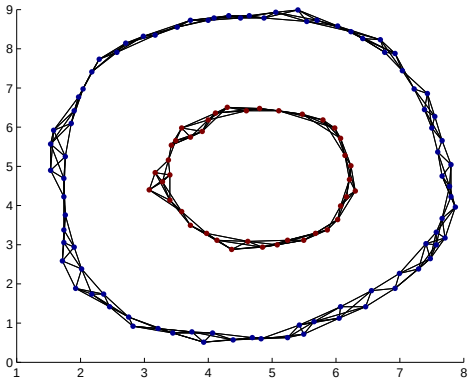


Figure 7: Neighbourhood graph and clusters for data in `X1.mat`, with $r = 5$

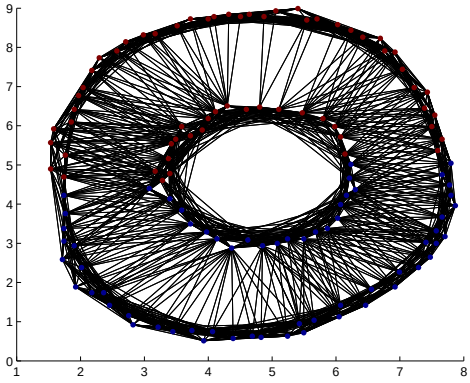


Figure 8: Neighbourhood graph and clusters for data in `X1.mat`, with $r = 20$

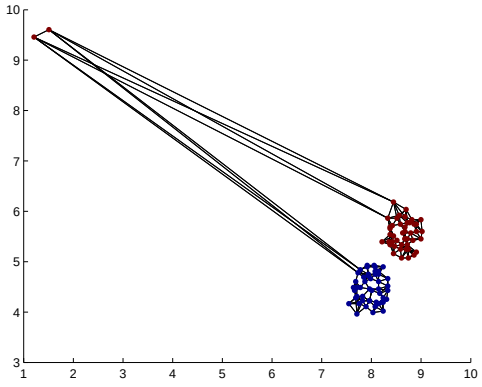


Figure 9: Neighbourhood graph and clusters for `X2.mat`, with $r = 5$

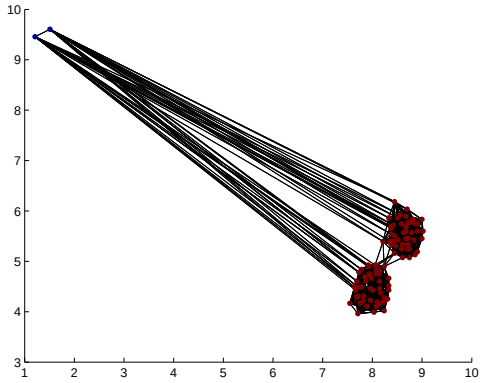


Figure 10: Neighbourhood graph and clusters for `X2.mat`, with $r = 20$

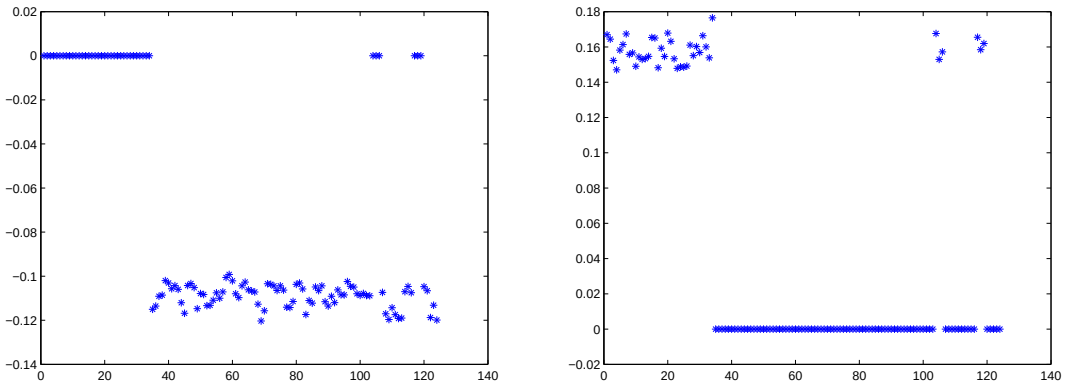


Figure 11: First two eigenvectors (plotted elementwise), with $r = 5$ for data in `X1.mat`

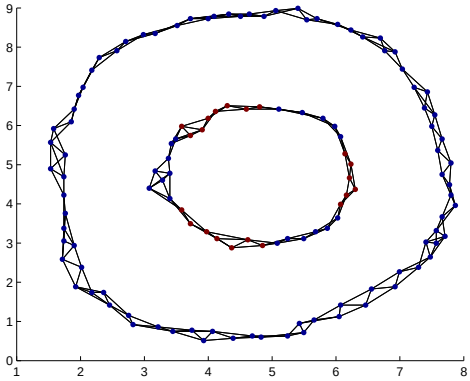


Figure 12: Neighbourhood graph and clusters for `X1.mat`, with $r = 4$

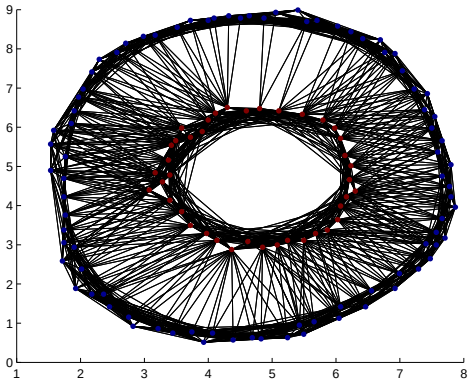


Figure 13: Neighbourhood graph and clusters for `X1.mat`, with $r = 20$ and $\beta = 5$. Notice that the 'correct' clustering is now obtained.

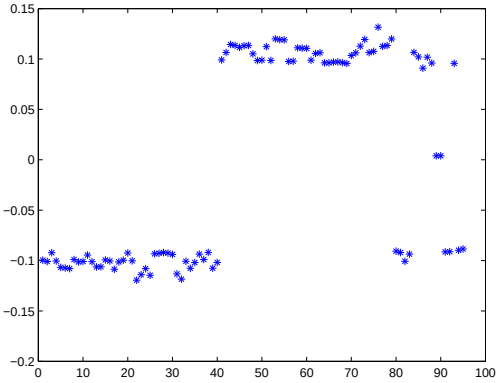


Figure 14: Second eigenvector (plotted elementwise), with $r = 5$ for data in `X2.mat`