Machine Learning for Data Science (CS 4786)

Lecture 6,7 & 8: Ellipsoidal Clustering, Gaussian Mixture Models and General Mixture Models

The text in black outlines high level ideas. The text in blue provides simple mathematical details to "derive" or get to the algorithm or method. The text in red are mathematical details for those who are interested.

1 Motivation

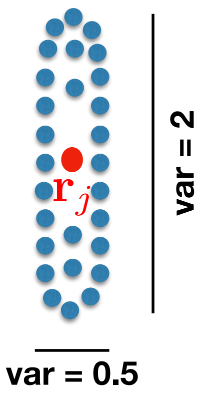
K-means algorithm looks for round clusters and cannot explicitly model clusters where one cluster has fewer number of points than another. We would like to address this issue. We will do this by changing the dissimilarity functions to first allow ellipsoidal clusters and further by explicitly by maintaining parameter π called mixture distribution that tells us the proportion of points within each cluster.

2 Ellipsoidal Clustering

The basic idea is going to be that each of our clusters will be explicitly modeled by an ellipsoid.

2.1 Prelude: Axis aligned case

To this end, say data-points within a cluster are spread as in the figure below.



For the example below, intuitively we would like the ellipse to be a vertically standing one. How so we obtain such an ellipse?

Well what we require is that in terms of dissimilarity measure, all the blue points on the outer ellipse have the same value of dissimilarity. That is to say that, we want to squish the ellipse vertically and elongate it horizontally so that it is circular (ie. all blue dots are same distance from center). Now say we scale each coordinate by $1/\sqrt{\text{Variance} - \text{of} - \text{the} - \text{coordinate}}$. In this case, we will find that the points have a variance of 1 on each coordinate. To see this, say

$$\tilde{\mathbf{x}}_t = \left[\mathbf{x}_t[1]/\sqrt{\text{Var}(\mathbf{x}_1[1], \ldots, \mathbf{x}_n[1])}, \mathbf{x}_t[2]/\sqrt{\text{Var}(\mathbf{x}_1[2], \ldots, \mathbf{x}_n[2])}\right]^\top$$

That is for each $\mathbf{x}_t$each $\tilde{\mathbf{x}}_t$ is the new variable whose coordinates are scaled inversely by standard deviation. We will notice that when the set of points are axis aligned (ie. are standing vertically or laying down horizontally), then $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n$ will have a varinace of 1 on each coordinate and 0 covariance amongst coordinates. Hence under $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n$ we see that all the blue points on the ellipse under the original set are now on a circle. To mathematically see this note that

$$\begin{aligned}
\text{Var}(\tilde{\mathbf{x}}_1[1], \ldots, \tilde{\mathbf{x}}_n[1]) &= \frac{1}{n}\sum_{t=1}^{n}\left(\tilde{\mathbf{x}}_t[1] - \frac{1}{n}\sum_{s=1}^{n}\tilde{\mathbf{x}}_s[1]\right)^2 \\
&= \frac{1}{n}\sum_{t=1}^{n}\left(\frac{\mathbf{x}_t[1]}{\sqrt{\text{Var}(\mathbf{x}_1[1], \ldots, \mathbf{x}_n[1])}} - \frac{1}{n}\sum_{s=1}^{n}\frac{\mathbf{x}_s[1]}{\sqrt{\text{Var}(\mathbf{x}_1[1], \ldots, \mathbf{x}_n[1])}}\right)^2 \\
&= \frac{1}{n}\sum_{t=1}^{n}\left(\mathbf{x}_t[1] - \frac{1}{n}\sum_{s=1}^{n}\mathbf{x}_s[1]\right)^2 \times \frac{1}{\text{Var}(\mathbf{x}_1[1], \ldots, \mathbf{x}_n[1])} \\
&= \text{Var}(\mathbf{x}_1[1], \ldots, \mathbf{x}_n[1]) \times \frac{1}{\text{Var}(\mathbf{x}_1[1], \ldots, \mathbf{x}_n[1])} \\
&= 1
\end{aligned}$$

Similarly you will find that the variance of the second coordinate is 1 as well. Now since we began with points distributed in an axis aligned way, covariance between different coordinates will be 0. Thus the new set of points are best described by a circle. Thus, we find that to define the right ellipse for the original set, that is for axis-aligned points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, to measure the ellipsoidal distance of a point say $\mathbf{x}$ to center, we can instead measure the usual notion of distance (euclidean distance) of the new point $\tilde{\mathbf{x}} = \begin{bmatrix} \frac{1}{\sqrt{\text{Var}(\mathbf{x}_1[1],\ldots,\mathbf{x}_n[1])}} & 0 \\ 0 & \frac{1}{\sqrt{\text{Var}(\mathbf{x}_1[2],\ldots,\mathbf{x}_n[2])}} \end{bmatrix} \mathbf{x}$ to the modified center $\tilde{\mathbf{r}}_j = \begin{bmatrix} \frac{1}{\sqrt{\text{Var}(\mathbf{x}_1[1],\ldots,\mathbf{x}_n[1])}} & 0 \\ 0 & \frac{1}{\sqrt{\text{Var}(\mathbf{x}_1[2],\ldots,\mathbf{x}_n[2])}} \end{bmatrix} \mathbf{r}_j$.

That is, the ellipsoidal distance

$$\begin{aligned}
d(\mathbf{x}, C_j) &= \|\tilde{\mathbf{x}} - \tilde{\mathbf{r}}_j\|^2 \\
&= \left(\tilde{\mathbf{x}} - \tilde{\mathbf{r}}_j\right)^\top \left(\tilde{\mathbf{x}} - \tilde{\mathbf{r}}_j\right) \\
&= \left(\mathbf{x} - \mathbf{r}_j\right)^\top \begin{bmatrix} \frac{1}{\text{Var}(\mathbf{x}_1[1],\ldots,\mathbf{x}_n[1])} & 0 \\ 0 & \frac{1}{\text{Var}(\mathbf{x}_1[2],\ldots,\mathbf{x}_n[2])} \end{bmatrix} \left(\mathbf{x} - \mathbf{r}_j\right) \\
&= \left(\mathbf{x} - \mathbf{r}_j\right)^\top \Sigma^{-1} \left(\mathbf{x} - \mathbf{r}_j\right)
\end{aligned}$$

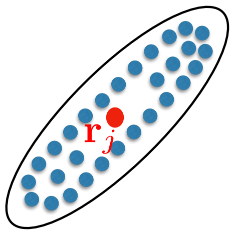
where Σ is the covariance matrix. Thus we have established that for the axis aligned case, the dissimilarity measure is

$$d(\mathbf{x}, C_j) = (\mathbf{x} - \mathbf{r}_j)^\top \Sigma^{-1} (\mathbf{x} - \mathbf{r}_j)$$

We will nest show that even for the general case, we have the same dissimilarity measure.

2.2 General Ellipsoids

Consider the general ellipsoid as shown below.



How do we get a slanted ellipsoidal dissimilarity for the above?

The high level picture is that, if we could somehow rotate the points so that they are axis aligned, then we can use the axis aligned version for dissimilarity. How do we rotate the points?

Basically what we want is to rotate the points such that the new set of rotated points are axis aligned. We can achieve this by considering the eigen decomposition

$$\Sigma = U \Lambda U$$

where U is a rotation matrix (and hence $U^\top = U$) and Λ is a diagonal matrix. Now for given $\mathbf{x}_1, \ldots, \mathbf{x}_n$ consider a new rotated bunch of points $\mathbf{x}'_1, \ldots, \mathbf{x}'_n$ where $\mathbf{x}'_t = U\mathbf{x}_t$. Note that, covariance matrix of $\mathbf{x}'_1, \ldots, \mathbf{x}'_n$ say Σ' is given by

$$\begin{aligned}
\Sigma' &= \frac{1}{n} \sum_{t=1}^{n} \left(\mathbf{x}'_t - \frac{1}{n} \sum_{s=1}^{n} \mathbf{x}'_s \right) \left(\mathbf{x}'_t - \frac{1}{n} \sum_{s=1}^{n} \mathbf{x}'_s \right)^\top \\
&= \frac{1}{n} \sum_{t=1}^{n} \left(U\mathbf{x}_t - \frac{1}{n} \sum_{s=1}^{n} U\mathbf{x}_s \right) \left(U\mathbf{x}_t - \frac{1}{n} \sum_{s=1}^{n} U\mathbf{x}_s \right)^\top \\
&= \frac{1}{n} \sum_{t=1}^{n} U \left(\mathbf{x}_t - \frac{1}{n} \sum_{s=1}^{n} \mathbf{x}_s \right) \left(\mathbf{x}_t - \frac{1}{n} \sum_{s=1}^{n} \mathbf{x}_s \right)^\top U^\top \\
&= U \left(\frac{1}{n} \sum_{t=1}^{n} \left(\mathbf{x}_t - \frac{1}{n} \sum_{s=1}^{n} \mathbf{x}_s \right) \left(\mathbf{x}_t - \frac{1}{n} \sum_{s=1}^{n} \mathbf{x}_s \right)^\top \right) U \\
&= U \Sigma U^\top = U U \Lambda U U = \Lambda
\end{aligned}$$

Thus we see that $\mathbf{x}'_t = U\mathbf{x}_t$'s the rotated points are now axis aligned with covariance Λ. Hence, the dissimilarity measure for the general case can be set as:

$$\begin{aligned} d(\mathbf{x}, C_j) &= \left(\mathbf{x}' - \mathbf{r}'_j\right)^\top \Lambda^{-1} \left(\mathbf{x}' - \mathbf{r}'_j\right) \\ &= \left(U\mathbf{x} - U\mathbf{r}_j\right)^\top \Lambda^{-1} \left(U\mathbf{x} - U\mathbf{r}_j\right) \\ &= \left(\mathbf{x} - \mathbf{r}_j\right)^\top U^\top \Lambda^{-1} U \left(\mathbf{x} - \mathbf{r}_j\right) \\ &= \left(\mathbf{x} - \mathbf{r}_j\right)^\top \Sigma^{-1} \left(\mathbf{x} - \mathbf{r}_j\right) \end{aligned}$$

Thus we can see that even for the general case, $d(\mathbf{x}, C_j) = \left(\mathbf{x} - \mathbf{r}_j\right)^\top \Sigma^{-1} \left(\mathbf{x} - \mathbf{r}_j\right)$ defines the right ellipsoidal dissimilarity measure.

Now as for the algorithm, it has the same flavor as the K-means algorithm in that at every step it first randomly initializes parameters $\mathbf{r}_1, \ldots, \mathbf{r}_K$ and $\Sigma_1, \ldots, \Sigma_K$ randomly. Next, each point is assigned to closest cluster under the new ellipsoidal dissimilarity measure

$$d(\mathbf{x}, C_j) = \left(\mathbf{x} - \mathbf{r}_j\right)^\top \Sigma_j^{-1} \left(\mathbf{x} - \mathbf{r}_j\right)$$

Next, in that iteration, for each cluster j, we recompute mean $\mathbf{r}_j$ and Σ_j. We repeat these two steps iteratively as shown in the pseudo code in the lecture slides.

2.3 Modeling Mixture Distribution

Say we had two clusters drawn from normal distribution of same covariance structure with means separated by some distance. Now say we have a point equidistant from both the means of the two clusters. Now if number of points drawn from both the gaussians were exactly the same, then we would of course have to conclude that this point equidistant to the mean could belong to either cluster with same probability. However, now say you were informed that one of the cluster has 10 times the number of points as the other cluster. Now you would expect that this point equidistant to the mean is 10 times more likely to be in the first cluster than the second. However, our cluster assignment step that only looks at dissimilarity does not capture this information. Hence to fix, this we maintain a mixture distribution parameter that maintains the proportion of points in each cluster at any iteration and aims to penalize more likely clusters lesser. This penalty to the dissimilarity function is given by $-\log(\pi_j)$ for the j'th cluster. The algorithm is given in the lecture slide.

3 Probabilistic Interpretation: Hard Gaussian Mixture Models

One can obtain a probabilistic interpretation of the algorithm as follows: the probability of a point belonging to a particular cluster j is proportional to probability of picking cluster j given by π_j times the likeloihood of the point belonging to cluster j. Notice that $\pi_j = \exp(-(-\log(\pi_j)))$ similarly, we can set likelihood $p(\mathbf{x}, C_j) \propto \exp(-\text{Dissimilarity}(\mathbf{x}, C_j))$. Notice that this ensures that density p is non-negative. To ensure that p is a valid density, it need to integrate to 1. Hence, $p(\mathbf{x}, C_j) = C_j \exp(-\text{Dissimilarity}(\mathbf{x}, C_j))$. Now once can calculate C_j the normalizing constant so that p integrates to 1. For the probabilisitic interpretation of the ellipsoidal dissimilarity, assume

$$\text{Dissimilarity}(\mathbf{x}, C_j) = \frac{1}{2} \left(\mathbf{x} - \mathbf{r}_j\right)^\top \Sigma_j^{-1} \left(\mathbf{x} - \mathbf{r}_j\right)$$

The factor 1/2 just makes calculations easier. Then likelihood is given by

$$p(\mathbf{x}; \mathbf{r}_j, \Sigma_j) \propto \exp(-\frac{1}{2} (\mathbf{x} - \mathbf{r}_j)^\top \Sigma_j^{-1} (\mathbf{x} - \mathbf{r}_j))$$

But note that p is basically proportional to the multivariate gaussian distribution and

$$\int \exp(-\frac{1}{2} (\mathbf{x} - \mathbf{r}_j)^\top \Sigma_j^{-1} (\mathbf{x} - \mathbf{r}_j)) = (2\pi)^{d/2} \sqrt{\det(\Sigma_j)}$$

and hence the density function for the probabilistic interpretation can be obtained by setting

$$p(\mathbf{x}; \mathbf{r}_j, \Sigma_j) = \frac{1}{(2\pi)^{d/2} \sqrt{\det(\Sigma_j)}} \exp(-\frac{1}{2} (\mathbf{x} - \mathbf{r}_j)^\top \Sigma_j^{-1} (\mathbf{x} - \mathbf{r}_j))$$

which is the multivariate gaussian distribution. Under the probabilistic interpretation the hard gaussian mixture model algorithm can be found in lecture slides. Specifically we can use for hard cluster assignment, assigning cluster to a point based on one that has maximum probability, that is a point is asinged to that cluster to which it has the maximum probability of belonging to. This probability is proportional to π_j (probability of picking cluster j) times the likelihood of point belonging to cluster j.

4 (Soft) Gaussian Mixture Models

One issue with hard clustering is that when we begin, we randomly guess parameters and recompute multiple times hoping to converge to the right ones. Say now there is a point that has probability 0.51 of belonging to cluster 1 and probability 0.49 of belonging to cluster 2 on iteration 1. So the point is close to being equally likely to belong to each of the two clusters, and this computation is based on randomly initialized parameters. Now based on this assigning point to only cluster 1 and not 2 seems too harsh. The soft assignment takes care of this issue by replacing the cluster assignment step on each iteration by a step that updates for each point the probability that it belongs to each of the K clusters. That is, every point belongs to every cluster with some probability given by the variable Q. Specifically, at any iteration m, $Q_t^{(m)}[j]$ specifies, based on parameters at iteration m, that is the probability that point $\mathbf{x}_t$ belongs to cluster j. Now when we compute mean and covariances at step 2 of iteration, for every cluster we compute the weighted covariance and means (and π) as shown in lecture slides.

When we do probabilistic models, we will come back to mixture models and see how this makes sense.