Assignment 2

Ziwen Sun Netid: zs197

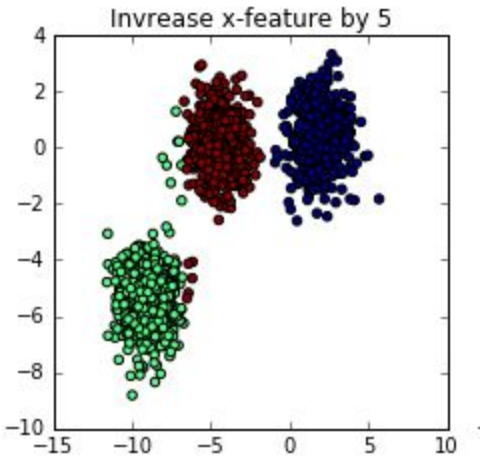
Q1

1.

- The number of clusters:
    - The value of k is very crucial to the performance of k-means. When k is greater than the actual number of clusters of the data, the cost will decrease very slow if we increase k, (as the cost the cost always decrease with k), while if k is less than the actual number of clusters, the cost will decrease very fast as we increase k. This is because at the initialization of random k points, if k is less than the actual number of clusters, there will always be actual clusters do not have good clusters center, at the assignment step, there will always be points assigned to further centers, no matter how many iterations, the cost will still be high, since the sum of squares is the squared Euclidean distance.
- Anisotropy in data:
    - Because in the assignment step of k-means, we use Euclidean distance as the measurement, so the algorithm cannot correctly reflect the anisotropy in data.
- Unequal variance:
    - k-means works best with clusters that the variance of the distribution of each attribute is spherical and that have the same variance. If the clusters have varying variances, at the assignment step yields the least within-cluster sum of squares , k-means will still treat the clusters as they have same variances, thus split them incorrectly.

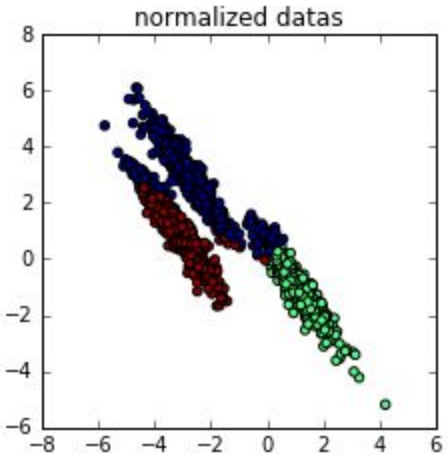
- Unequal cluster sizes:
    - When the clusters have different sizes especially when they differ greatly, k-means will prefer giving more weight to larger clusters at the update step, while leaving the small clusters far away from any center.

2. Few points are incorrectly clustered, because after increasing *x*-feature, some points have closer Euclidean distance to other cluster centers. So k-means clustered these points to wrong clusters.

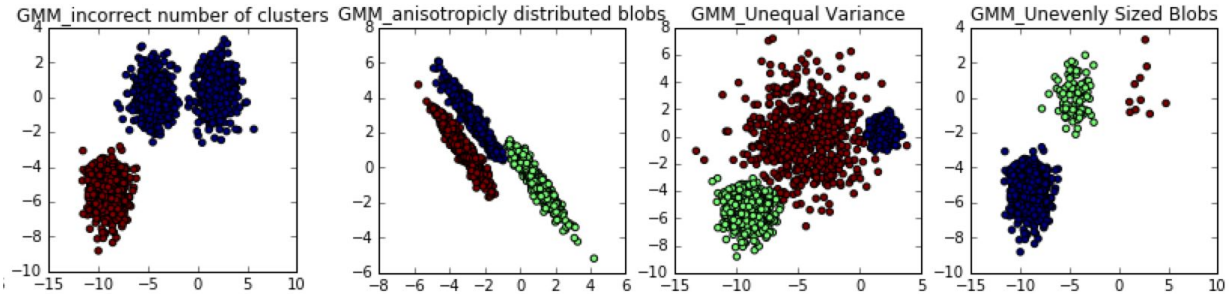


3. The affine transformation of data should be applied to all features. In this case, we should also do same scale affine transformation to y-feature.

4. We can use normalization to scale individual samples to have unit norm to mitigate the anisotropy in data. There are still some data points are incorrectly clustered.



5. **Incorrect number of clusters**: GMM still can be affected by incorrect number of clusters, actually the number depends on the data itself, if we use less clusters to cluster it, we can only get incorrect results.



Anisotropy in data: GMM can mitigate the anisotropy in data quite well. Unlike k-means using Euclidean distance as measurement, we use r_{ic} - the probability that data points belong to clusters, so the anisotropy in data doesn't affect GMM.

Unequal variance: GMM consists of k components, each component is a Gaussian distribution, they may have different covariances $\sum_c$, so the unequal variances of data won't affect the performance of GMM.

Unevenly sized blobs: The data which has unevenly sized blobs can be clustered well by GMM. In k-means, we using Euclidean distance as measurement, so larger clusters get more weight when updating new cluster centers, but GMM use r_{ic} - the probability that data points belong to clusters, so the different sized blobs won't affect the performance of GMM.

(see the code in attachment for implementation and experiment)

Q2

1. A Gaussian distribution in *d* dimensions is:

$$p(x) = \frac{1}{(2\pi\sigma^2)^{d/2}} exp(-\frac{\|x\|^2}{2\sigma^2})$$

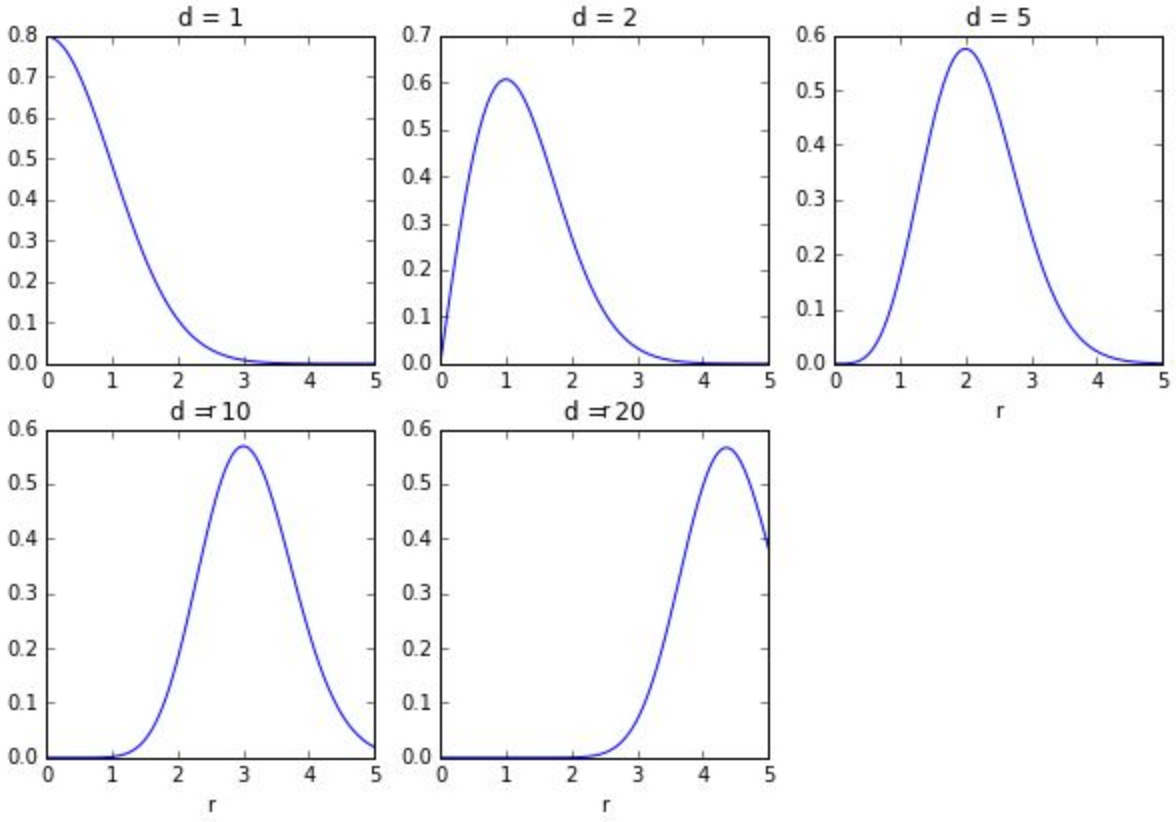
Since *p(x)* is radially symmetric it will be roughly constant over the shell of radius *r* and thickness *ε*. This shell has volume $S_d r^{d-1}\varepsilon$ and since $||x||^2 = r^2$ we have

$$\int_{shell} p(x)dx \simeq p(r)S_d r^{d-1}\varepsilon$$

Substitute to Gaussian distribution, we obtain:

$$p(r|d)\varepsilon = \frac{S_d r^{d-1}}{(2\pi\sigma^2)^{d/2}} e^{-\frac{r^2}{2\sigma^2}}\varepsilon$$

2. The function graph is plotted as below. `(see the code in attachment for implementation)`



3. Do differentiation on *p*(*r*) and let it equals 0:

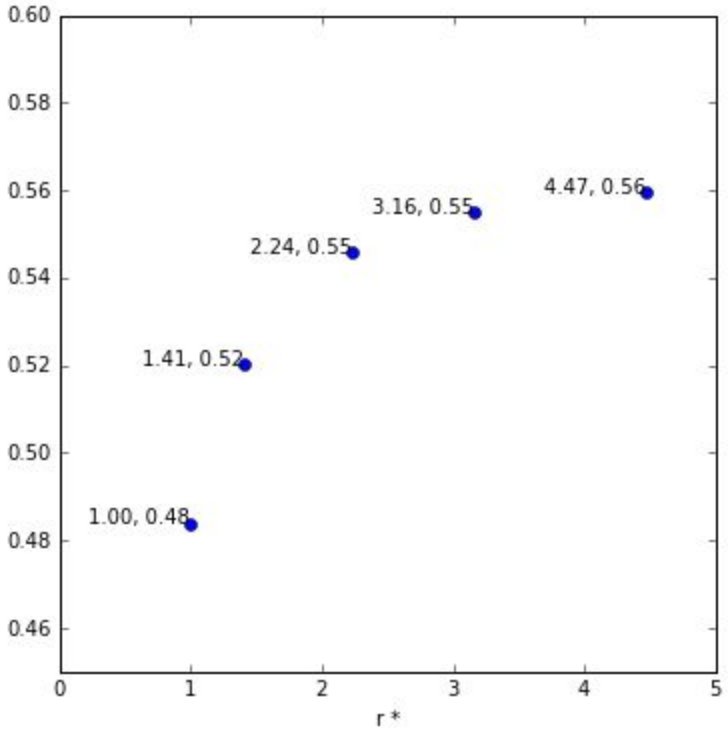
$$\frac{d}{dr}p(x) \propto \left[(d-1)r^{d-2} + r^{d-1}\left(-\frac{r}{\sigma^2}\right)\right] exp\left(-\frac{r^2}{2\sigma^2}\right) = 0$$

Solving for *r*, and because d is large, so we get :

$$r^* \approx \sqrt{d}\sigma$$

So the that the function $p(r|d)$ has a stationary point located at for $r^* \approx \sqrt{d}\sigma$ for large *d*.

4. The values are plotted as below. `(see the code in attachment for implementation)`



5. The probability density at the origin is:

$$p(x = 0) = \frac{1}{(2\pi\sigma^2)^{1/2}}$$

Now consider $p(r^* + \varepsilon)$ where $\varepsilon \ll r^*$, we have:

$$p(r^* + \epsilon) \propto (r^* + \epsilon)^{d-1} exp\Big[- \frac{(r^* + \epsilon)^2}{2\sigma^2} \Big]$$
$$= exp\Big[- \frac{(r^* + \epsilon)^2}{2\sigma^2} + (d-1) ln(r^* + \epsilon) \Big]$$

Now expand $p(r)$ around the point r^*, since this is a stationary point of $p(r)$, we can use the expansion $ln(1+x) = x - x^2/2 + O(x^3)$, together with $d \gg 1$, we can obtain:

$$p(r^* + \epsilon) = p(r^*)exp\Big(- \frac{3\epsilon^2}{2\sigma^2}\Big)$$

which shows that $p(r^*)$ is a maximum of radial probability density.

From question 1, the probability density at ||x|| = r^* where $r^* \approx \sqrt{d}\sigma$ is :

$$p(||x|| = r^*) = \frac{1}{(2\pi\sigma^2)^{1/2}}(-\frac{r^2}{2\sigma^2}) = \frac{1}{(2\pi\sigma^2)^{1/2}}(-\frac{d}{2})$$

So the ratio of density between the origin and the maximum probability mass in a thin shell is $exp(-\frac{1}{2})$.

6. From the analysis above we can know that most of the probability mass is concentrated in a thin shell at a large radius. And from question 3 and the graph in question 4, we can also see most of the probability mass in high-dimensional Gaussian distribution is located at a different radius from the region of high probability density. As we adding dimensionality, the probability mass of density will move away from the cluster center.

Q3

1. We have GMM constructed for dataset D, it consists of K component, each component is a Gaussian distribution. Together, we have the Gaussian mixture density distribution:

$$\begin{aligned} p(x) &= \sum_{k=1}^{K} p(k)p(x|k) \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k) \end{aligned} \tag{1}$$

For every data point x_i, the probability that it belongs to kth component is :

$$\gamma(i,k) = \frac{\pi_k \mathcal{N}(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(x_i|\mu_j, \Sigma_j)} \tag{2}$$

The assignment rule implies that each point in each corresponding cluster is mapped to the mean of the Gaussian modeling the cluster, $c_{post}(x) = \mu_k$, $k = arg\,max_i\, Pr\{x \in C_i | M_{GMM}\}$, in other words, we want to find the maximum $\gamma(i,k)$, note the denominator in (2) is constant for fixed x_i, so we only have to find the k that maximize :

$$k^* = arg\,max_i\, \pi_k N(x_i|\mu_k, \Sigma_k) \tag{3}$$

where $N_k = \Sigma_1^N \gamma(i,k)$, $\pi_k = N_k / N$

If we look at each component density is a D-variate Gaussian function of the form:

$$p(x|\mu_k, \Sigma_k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} exp\Big(-\frac{1}{2}(x-\mu_k)^T \Sigma_k^{-1}(x-\mu_k)\Big) \tag{4}$$

With mean vector μ_i and covariance matrix Σ_i

While we look at $c_{locmode}(x)$, we want to find the nearest Gaussian cluster center, denote as :

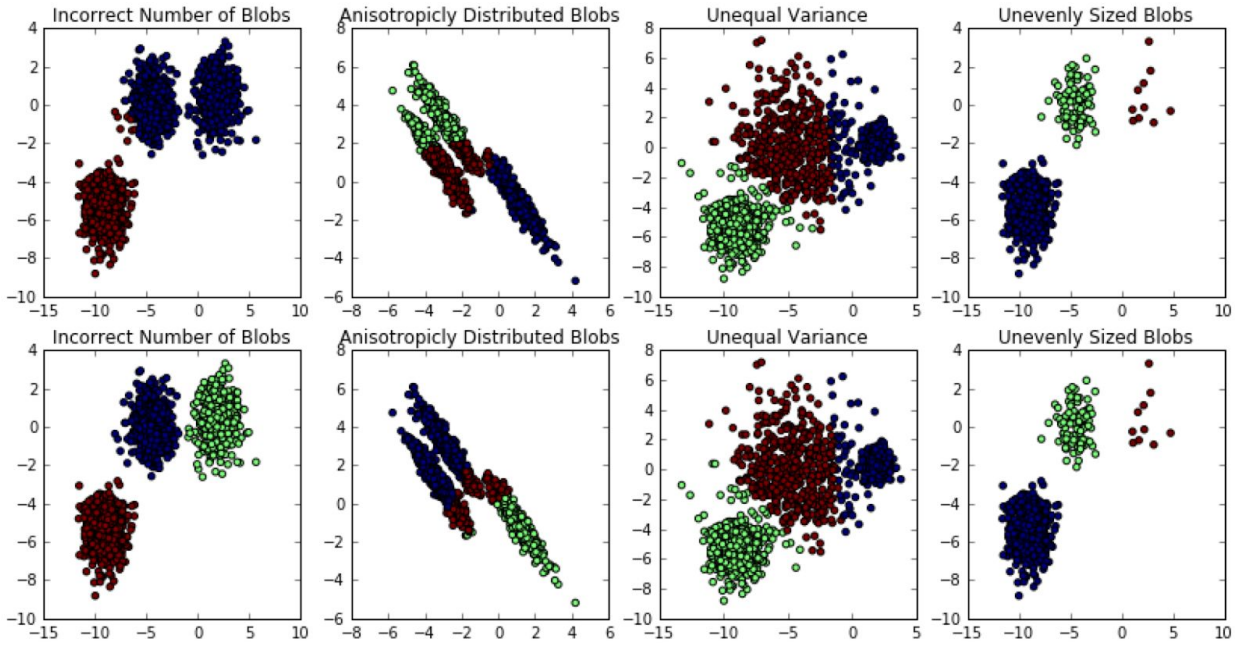
$$k^* = arg\,min_i ||x - \mu_k||^2 \tag{5}$$

From (3)(4)(5), we can know the **result of two assignment rules are not guaranteed to give same outputs.** First, the first rule has coefficient which the second rule does not have, the covariance matrix in the equation might affect the assignment of cluster center. Second, the first rule has treat every cluster as same weight, because only Euclidean distance affect the assignment center, while the first one has π_k as weight.

The conditions when two rules would result in the same assignment:

1) No anisotropy in data. All the coefficient in all Gaussian component should be equal.
2) The transpose of covariance matrix should be ***I***, or be in the form of $\sigma^2 I$ which will result in same results.
3) All component of GMM should be equal size..

2. The plotted results are below:



Incorrect Number of Blobs: k-means relies on the assumed number of clusters k, but mean-shift does not assume any predicted number of clusters, so incorrect number of blobs does not affect the performance of mean-shift.

Anisotropy in data: Both k-means and mean-shift performed badly when there is anisotropy in data. Mean shift clustering using a flat kernel, it cluster data points by updating candidates for centroids to be the mean of the points within a given region. So when there is anisotropy in data, mean-shift cannot correctly shift to the cluster center, just like k-means.

Unequal Variance: k-means and mean-shift has very similar clustering results. Both of two algorithm incorrectly clustered some data points. Mean shift computes the gradient of the density estimate $f(x)$ and takes an uphill step in that direction. When the clusters have different variance, mean-shift tend to give more weight to the clusters have higher density.

Unevenly sized blobs: Both k-means and mean-shift clustered unevenly sized blobs well, but k-means sometimes can be affect by unevenly sized blobs, for example when there is a extreme large cluster and a relatively very small cluster, k-means might ignore the small one. Mean-shift is based on density so the size has micro effect on performance.