1. (a)
   - Setting the number of points apriori: K-Means is sensitive to this factor because it assumes there are K clusters in the data set where K is determined apriori. K-Means does not attempt to find the inherent organization of the data but simply finding the most probable clusters according to K.
   - Anisotropy: K-Means is sensitive to anisotropy in data because it assumes the data is spherical/convex. In the K-Means model, it tries to fit data around K centers which minimizes the within cluster sum of squared criterion. If the data in an intuitive cluster is distributed anisotropicly, K-Means is likely to group a part of the data with data from other intuitive cluster to minimize within cluster sum of square instead of preserving the intuitive non-sperical shape.
   - Unequal variance: K-Means assumes the data is separated in K groups of equal variance. In each iteration, K-Means reduces the difference in variance between K clusters and converges until the variance between groups are the same. Thus if two intuitive clusters have different variance, K-Means will compensate by assigning points from other intuitive clusters to each until the variance converges, producing an unsatisfactory cluster assignment.
   - Unequal cluster sizes: K-Means is affected because it tries to minimized the within-cluster sum of squares, which natually put more weight to clusters with more data. Consider a dataset which contains two intuitive clusters K, J, where the number of data points in each cluster $k << j$ and the average squared distance to cluster center is the same d_{avg}. Then the within-cluster sum of squares $k * d_{avg} << j * d_{avg}$. K-Means will assign points from intuitive cluster J to K in order to balance the within-cluster sum of squares between them, thus producing an undesirable cluster assignment. In the example given by the code, the within-cluster sum of squares are incidentally equal between three intuitive clusters although they have different number of data points, thus K-Means is able to produce a desirable result under such specific condition.

   (b) As we can see in Figure 1, some points that belongs to the green cluster in the original cluster assignment is marked as red after feature scaling and vice versa. Feature scaling affects the K-Means because it affects the variance and the isotropy of each cluster. In the lower graph of Figure 1, the clusters are less spherical compare to the upper graph, thus K-Means moves points between the red and green clusters to balance the variance between them. Please refer to *simple_feature_scaling.py* for code.

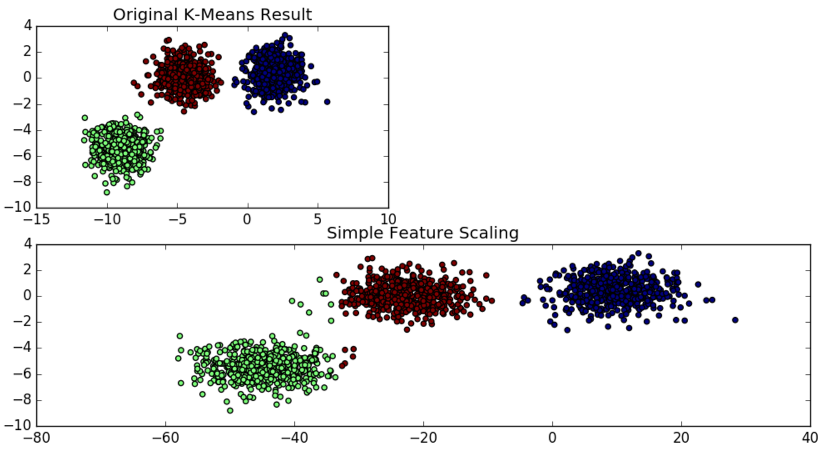

Figure 1: Before and after increasing the x-feature by a factor of 5.

   (c) The transformation must be orthogonal, that is, $AA^T = kI$. In this case, the mean and variance of each cluster are preserved thus K-Means will produce the same result. b only affects the relative position of all data points to the origin to the same extent, thus won't affect clustering result.

(d) We can preprocess the data by normalization. Normalize the mean and scale each feature to unit variance.

(e) As shown in the upper right graph in Figure 2, GMM mitigates the effect of anistropicy by allowing custom covariance matrices for each Gaussian distribution. As shown in the bottom-left graph in Figure 2, GMM mitigates the effect of unequal variance because it allows each cluster(Gaussian distribution) to have different variances. Although cannot be shown through this dataset, GMM also mitigates the effect of unevenly sized blobs because it allows different covariance matrices for each cluster. Since GMM requires a input of K mixtures apriori, it does not mitigate the effect of setting the number of clusters apriori in K-Means.

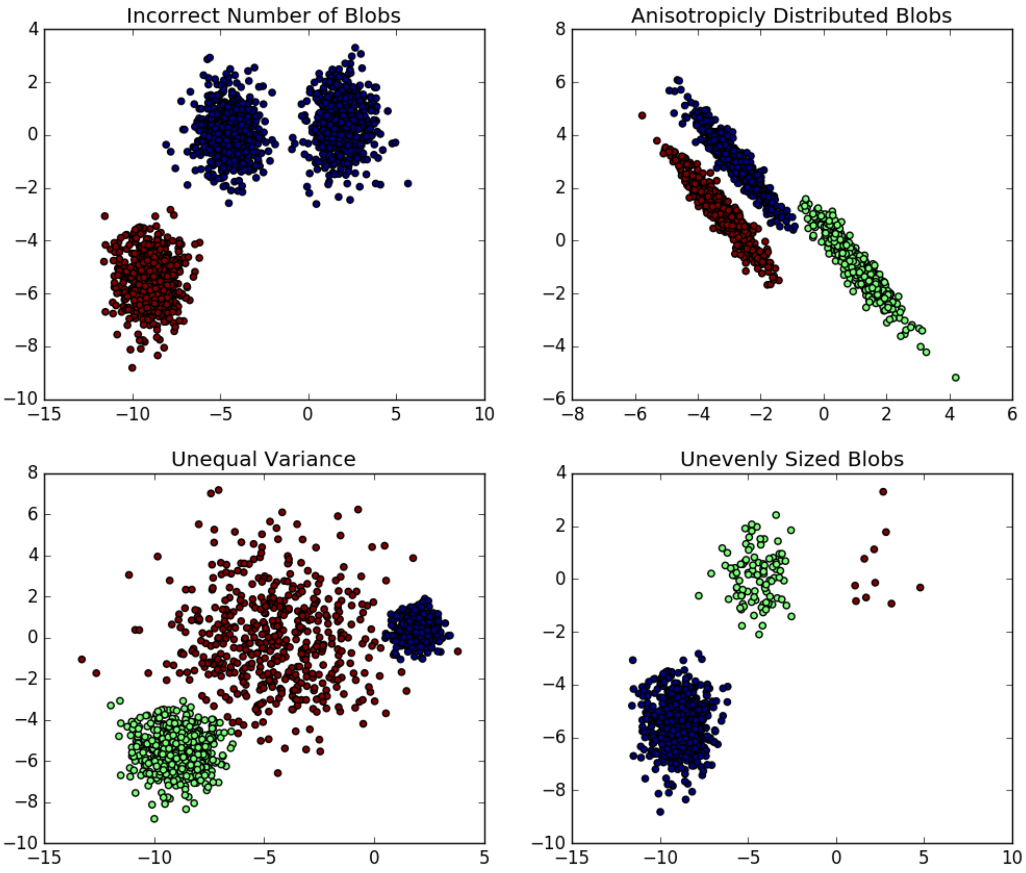

Figure 2: Clustering using GMM.

2. (a) For Gaussian distribution in d dimensions, we have density function:

$$p(x_d) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)) \tag{1}$$

where Σ is the covariance matrix and $|\Sigma|$ is the determinant of Σ. Since we have $\mu = 0$ and $\Sigma = \sigma^2 I$, substitute them in (1) we get:

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

Since the integral of the probability density over a thin shell of radius r and thickness ϵ is equivalent of the volume of the d dimentional geometric space, we can find the integral of $p(x_d)$ by calculating the volume of the shell $S_d r^{d-1}\epsilon$:

$$p(r|d)\epsilon = \int p(x_d)dx_d \approx p(r)S_d r^{d-1}\epsilon \tag{3}$$

Since $||x||^2 = r^2$, substitute r into (2) we have:

$$p(r|d)\epsilon = \frac{S_d r^{d-1}}{(2\pi\sigma^2)^{d/2}} \exp(-\frac{r^2}{2\sigma^2})\epsilon \tag{4}$$

which is what we wanted to show.

(b) Please refer to Figure 3. The code can be found in $/src/plot_density.py$.

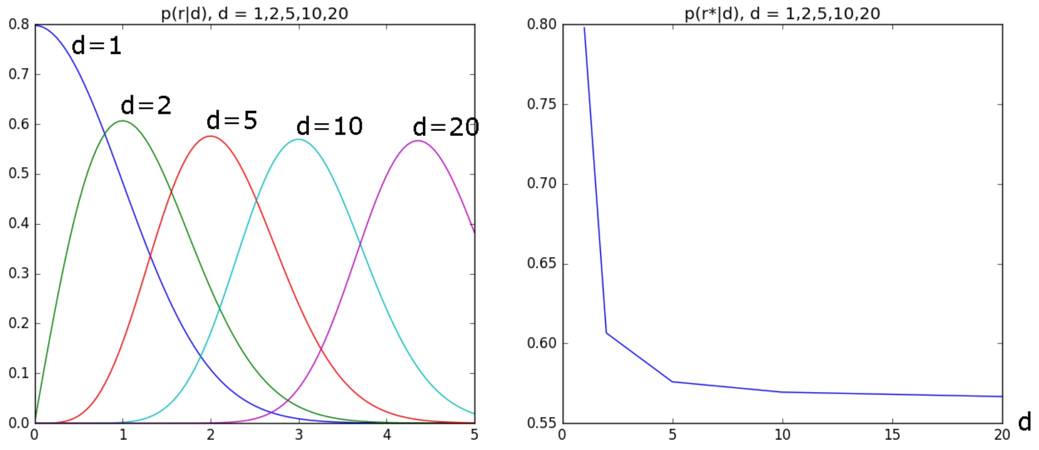

Figure 3: $p(r|d)andp(r*d), d = 1, 2, 5, 10, 20$

(c) Stationary points can be found by differentiating $p(r|d)$ and solve for r when the derivative is 0::

$$\frac{d}{dx}p(r|d) \propto [(d-1)r^{d-2} + r^{d-1}(-\frac{r}{\sigma^2})]\exp(-\frac{r^2}{2\sigma^2}) = 0 \tag{5}$$

Since $\exp(-\frac{r^2}{2\sigma^2}) \neq 0$, we solve:

$$\begin{aligned} (d-1)r^{d-2} + r^{d-1}(-\frac{r}{\sigma^2}) &= 0 \\ (d-1)r^{d-2} &= \frac{r^d}{\sigma^2} \\ \sigma^2(d-1) &= \frac{r^d}{r^{d-2}} \\ r^2 &= \sigma^2(d-1) \\ r &= \sqrt{d-1}\sigma \end{aligned} \tag{6}$$

As $d >> 1$, $r^* \approx \sqrt{d}\sigma$.

(d) Please refer to Figure 3. The code can be found in $/src/plot_density.py$.

(e) To find the density at the origin and r^*, substitute $||x||^2 = 0$ and $r^* = \sqrt{d}\sigma$ into equation (2) and we have:

$$\begin{aligned} p(x=0) &= \frac{1}{(2\pi\sigma^2)^{d/2}} \\ p(x=r^*) &= \frac{1}{(2\pi\sigma^2)^{d/2}} exp(-\frac{(\sqrt{d}\sigma)^2}{2\sigma^2}) \\ &= \frac{1}{(2\pi\sigma^2)^{d/2}} exp(-\frac{d}{2}) \end{aligned} \tag{7}$$

Thus it is obvious that $p(x = r^*) = exp(-\frac{d}{2})p(x = 0)$.

(f) Although the value of the Gaussian is maximum at $x = 0$, there is very little probability mass at the origin. In fact, not until $r^* = \sqrt{d}\sigma$ is there a significant volume and the probability mass is concentrated in a thin shell near the surface. As the means of the Gaussians are very close, given data points from the mixture it is very hard to tell which Gaussian each data point comes from. Thus the analysis implies that it is hard to determine how much separation is needed between the means to tell which Gaussian generated which data point when modeling clusters in high dimensions as Gaussian densities.

3. (a) The second assignment rule will not result in the same assignments as c_{post} applied to model M_{GMM}. The reason is that c_{post} does not care about the distance between the data point and the mean of the Gaussian. Consider a situation where point x is far away from the mean of Gaussian G_i but $Pr(x \in G_i)$ is high. In this case x will be assigned to G_i applying c_{post}, while using $c_{locmode}$ there is likely be another Gaussian which has mode closer to x than G_i. Thus the two assignment rules will produce different results. The condition under which the two rules will result in the same assignments is when the weight of each Gaussian in the mixture is equal. Thus the Gaussian with the highest posterior probability will always be the closest one to the data point, which is the same as the mode found by $c_{locmode}$

(b) According to Figure 4, the result of K-Means and mean-shift similarly produce undesirable results when handling anisotropic data and unequal variance. Since mean-shift aims to find the centroids that represent the mean of points in a given region, it is susceptible to the same pitfalls as K-Means. Mean-shift is different from K-Means in that it does not require the number of clusters be specified apriori, thus performs well compare to K-Means. Although can not be seen in the graph, mean-shift should perform better than K-Means when handling unevenly sized blobs because the mean of a cluster is not affected by the number of points in the cluster.

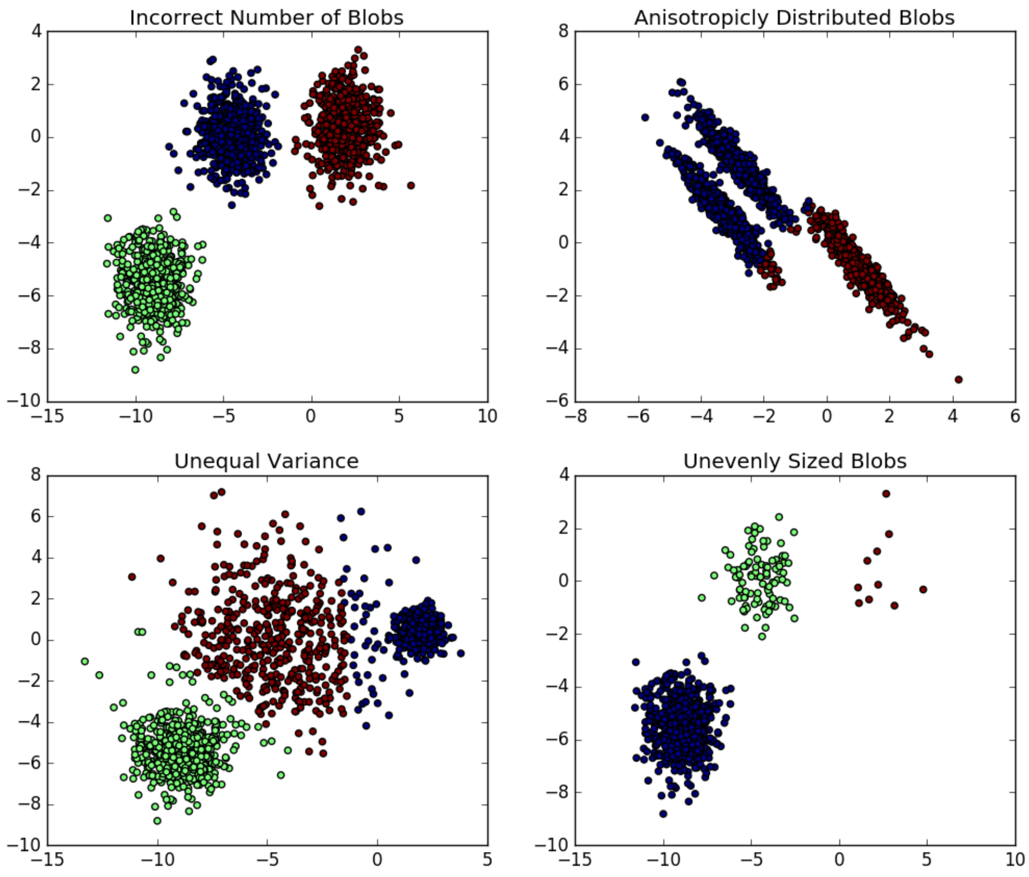

Figure 4: Applying Mean-Shift to the data from problem 1.