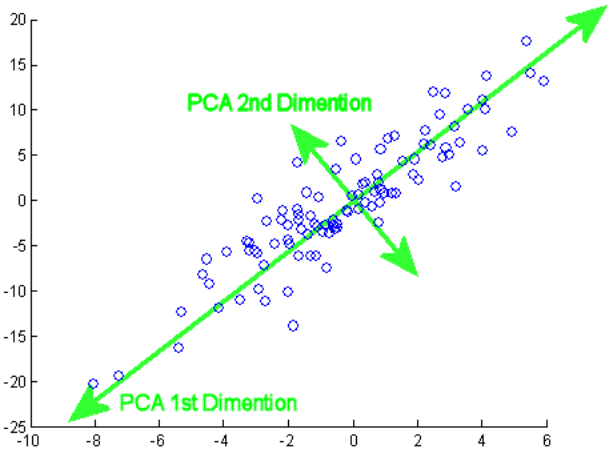
CSCI 5150 Midterm Questions

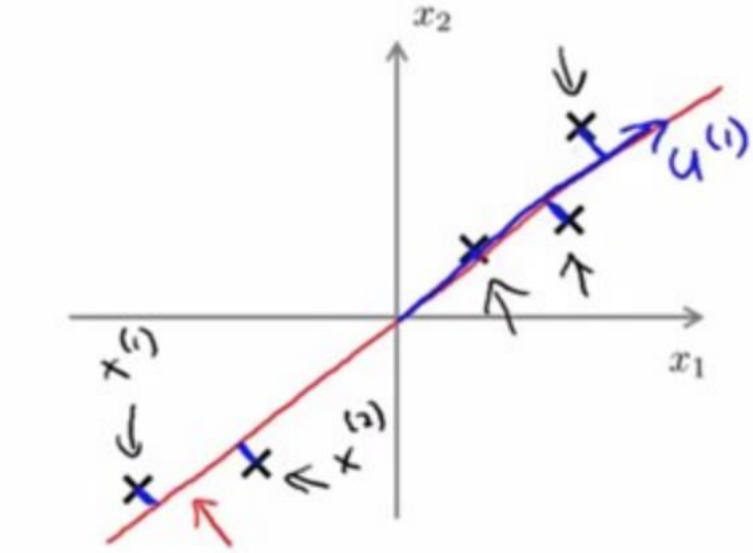
Zhang,Tuxin
1155101156

2-1.

If we want to use lines to cluster data points instead of “centroids” in K-means. We need to find a way to represent a cluster. Recall in K-means we use centroid to represent a cluster.

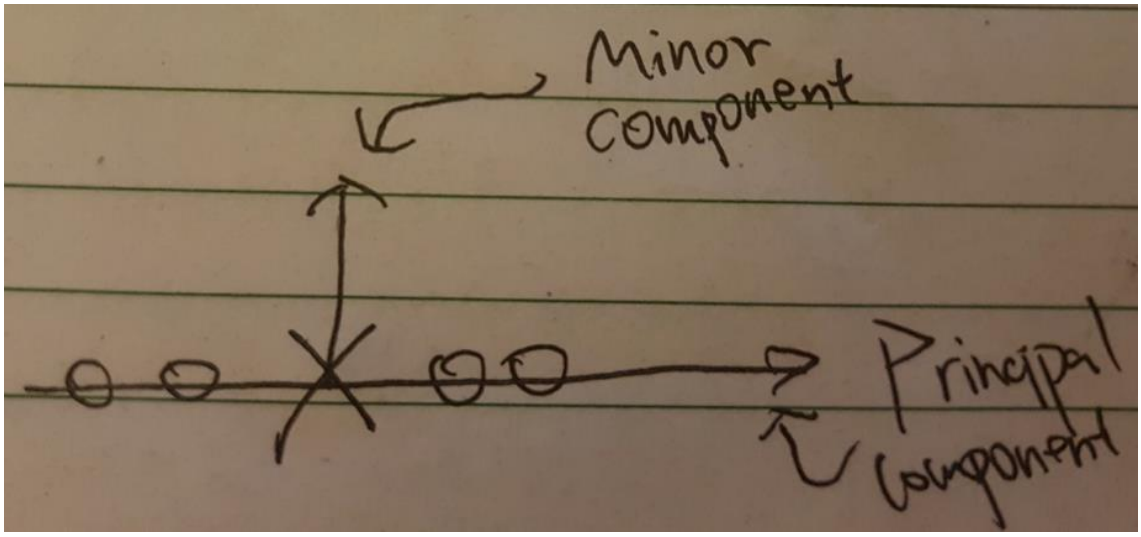


In PCA (or MCA) , let p be a point and v be a unit vector. Then p.v gives the distance from the origin to the projection of p on v. We can think of the vector v is the line, and the distance between a point



p and v is that the euclidean distance of position of x and after projection the position of x’. But we don’t actually want the principal component to represent my data since after projection the data points are far away. We actually want Minor component (the unit eigenvector corrsponding to the smallest eigenvalue in PCA) to represent our data. We can see below, we only use 4 points to illustrate. If we use Minor component, the X in the image is actually 1 point, all 4 data points are project to one single point X in the image, this is very similar to K-means we use 1 point to represent

the cluster. After projection the data points are very close, which is what a cluster need.



K-Lines Algorithm:

Notice: Same as K-means algorithm, this algorithm also highly depends on initialization. For simplicity we will just assume the initialization will be relatively fine.

1. Randomly initialized k lines j. (For j in 1…k)
2. For all points p, find the euclidean distance(vertical distance) from p the line j. Assign p to cluster j where p is closest among all k lines.
3. Computer MCA inside each cluster, make the minor component the vector representing the line.
4. Re-assign the all points p base on the distance(p,p') where p' is the position of p after projection onto minor component.
5. See if the minor component converge(stable). If not, go back to step 2.

2-2.

(a)

For convenience, Let's first take a look at the formula in PPT-CSCI5150.18b-P10.

1. $$p(j \mid x_t) = \frac{\alpha_j G(x_t \mid m_j, \Sigma_j)}{\sum_r \alpha_r G(x_t \mid m_r, \Sigma_r)}$$

2. $$N_\ell = \sum_t p_{\ell,t}, \quad \alpha_\ell^{new} = \frac{N_\ell}{N},$$
$$\mu_\ell^{new} = \frac{1}{N_\ell} \sum_t p_{\ell,t} x_t,$$
$$\Sigma_\ell^{new} = \frac{1}{N_\ell} \sum_t p_{\ell,t} (x_t - \mu_\ell^{old})(x_t - \mu_\ell^{old})^T$$

Notice that K-means uses hard classification, Gaussian mixture model uses soft classification (We can view as each point is divide into some parts and they belongs to different clusters) .

K-Gaussian Algorithm.

Introduction:

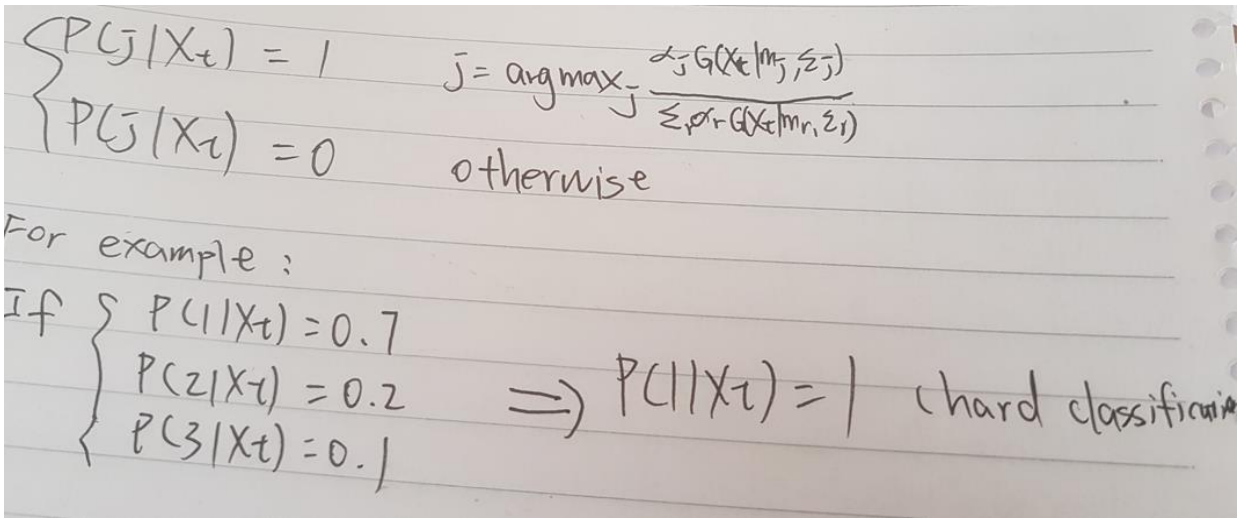
1.Use K Gaussian to represent data points (In K-means we use K centroids to represent data points).

2.Use hard classification in K-Gaussian (Like K-means).

We can still using EM algorithm to estimate the parameters in K-Gaussian but with come changes.

By referring to the formulas we can get K Gaussian algorithm:

1.Since K-Gaussian use hard classification, $p(j \mid x_t) = \frac{\alpha_j G(x_t \mid m_j, \Sigma_j)}{\sum_r \alpha_r G(x_t \mid m_r, \Sigma_r)}$ need to be change to



In this way, we are using hard classification like K-means.

2. Next we need use K Gaussian to represent data instead of centroids. Actually

$$N_\ell = \sum_t p_{\ell,t},\ \alpha_\ell^{new} = \frac{N_\ell}{N},$$
$$\mu_\ell^{new} = \frac{1}{N_\ell}\sum_t p_{\ell,t} x_t,$$
$$\Sigma_\ell^{new} = \frac{1}{N_\ell}\sum_t p_{\ell,t}(x_t - \mu_\ell^{old})(x_t - \mu_\ell^{old})^T$$

we can use the same formula , notice that the $P_{l,t}$ is changed as in step 1(Hard classification). N_l is really how many points in cluster l (no probability involved, hard classification). α_l is the percentage of points in this cluster with respect to the total points. μ_l is "true mean" in that cluster (without any probability since $P_{l,t}$ is just 1). Same for the co-variance matrix **Σ.** Overall, the formulas in this step are not changing but the meaning of some of the elements inside the formula changed.

3. Just like EM algorithm, repeat step 1 and step 2 until it is converge(parameters are all stable). Then stop the algorithm.

(b) When all α_j are equal, what is the algorithm now?

First α_j are all equal means the probability that an observation comes from any population k are all equal.

Case 1:

If α_j are just set to be equal at the initialization stage of EM algorithm, then as EM algorithm runs, α_j will be changed until it is converge for all clusters.

Case 2:

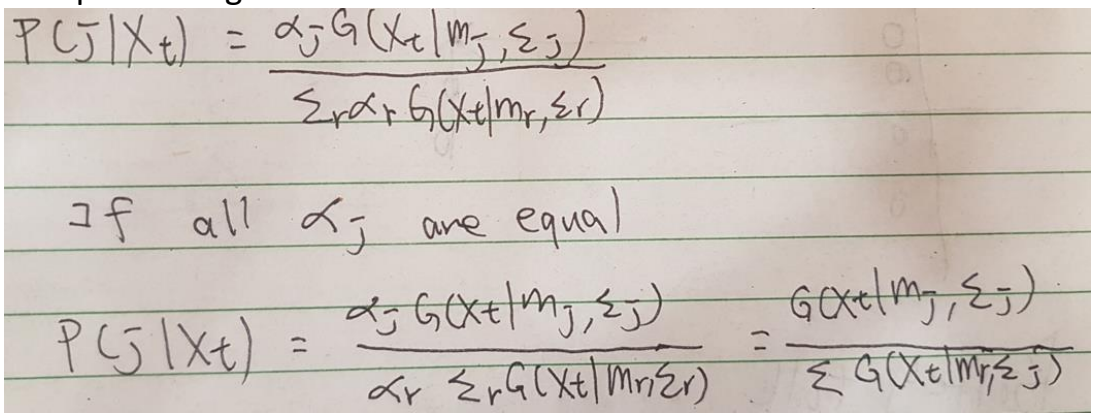
If α_j are truly equal from the beginning to the end of the algorithm. Then there are some modification can be done to the algorithm.

Since all α_j are all equal, j = 1…K since total K clusters. And by definition $1 = \sum_{k=1}^{K} \alpha_k$. Thus for all $\alpha_j = 1/K$.

$$p(x) = \frac{1}{K}\sum_{k=1}^{K} \mathcal{N}(x|\mu_k, \Sigma_k)$$

Initialization step: All α_j are equal.

E step of EM algorithm:



The $P(j|X_t)$ will become only the Gaussian probability of X_t of Gaussian j divide by the sum all Gaussian probability of X_t. This in some sense means that the probability of X_t is belongs to cluster j depends on the position of X_t from μ_l with respect to $\mathbf{\Sigma}_l$. (Base on the $\mathbf{\Sigma}_l$ the closer you are from μ_l, the more likely you are belongs to cluster js)

M step of EM algorithm:

$\alpha_\ell^{new} = \frac{N_\ell}{N}$ since α_j are all equal, implies that all N_l are equal, which means the number of points in each cluster will be the same. Actually we don't even need to estimate this one, because since α_j are all equal, α_j = 1/K (K : Total number of clusters)

$\mu_l = 1/K \sum_t p_{\ell,t} x_t$

$\mathbf{\Sigma}_l = 1/K \sum_t p_{\ell,t} (x_t - \mu_\ell^{old})(x_t - \mu_\ell^{old})^T$

Finally, see if μ_l and $\mathbf{\Sigma}_l$ converge. If not run EM again with the newly computed parameters.

Overall, set all α_j to be equal means the probability that an observation comes from any population k are all equal (α_j = 1/K). Lead to number of points in each clusters are equal. Then the algorithm runs purely depends on the Gaussian distribution of data points, not related to mixing proportion α_j anymore since they are all equal. And we can still use EM algorithm to estimate μ_l and $\mathbf{\Sigma}_l$, (No need to estimate α_j since they are all equal).

(c)
First:

Gaussian mixture model soft assigns a point to clusters (so it give a probability of any point belonging to any centroid).

K means Hard assign a data point to one particular cluster on convergence. Since there are no probability involved, we need to delete all things related to probability.

So, $p_{\ell,t} = p(j \mid x_t) = \frac{\alpha_j G\left(x_t \mid m_j, \Sigma_j\right)}{\sum_r \alpha_r G\left(x_t \mid m_r, \Sigma_r\right)}$ needed to be change to $p(j \mid x_t)$ =1 if x_t has the least squared Euclidean distance to the cluster's centroid j which is (argmin_j ||j - x_t ||), otherwise it is equal to 0 , where the centroid is the mean $\mu_\ell^{new} = \frac{1}{N_\ell} \sum_t p_{\ell,t} x_t,$.

From $N_\ell = \sum_t p_{\ell,t}$, $\alpha_\ell^{new} = \frac{N_\ell}{N}$, , we can also delete $\alpha_\ell^{new} = \frac{N_\ell}{N}$, since we don't need probability. Now, $N_\ell = \sum_t p_{\ell,t}$ should simply means the number of points in cluster l. Now $\mu_\ell^{new} = \frac{1}{N_\ell} \sum_t p_{\ell,t} x_t,$ should means for all points x_t, $p_{\ell,t}$ = 1 if x_t belongs to cluster l, otherwise 0. And this cluster has N_ℓ points. $\sum_t p_{\ell,t} x_t$ add all points in this cluster together, $= \frac{1}{N_\ell}$ means divide by the numbers of points which is exactly the centroid. Compute it for all l (all clusters k). Again and again under it is stale (converge).

Finally, delete all variance and co-variance $\Sigma_\ell^{new} = \frac{1}{N_\ell} \sum_t p_{\ell,t} (x_t - \mu_\ell^{old})(x_t - \mu_\ell^{old})^T$ since it is K-means so we won't need it.