Mixture Models

Week	TUES. Week 9
Assignment Due	
Assignment Done	☐
Due Date	
Notes Done	☑

Class Notes

Mixture Models

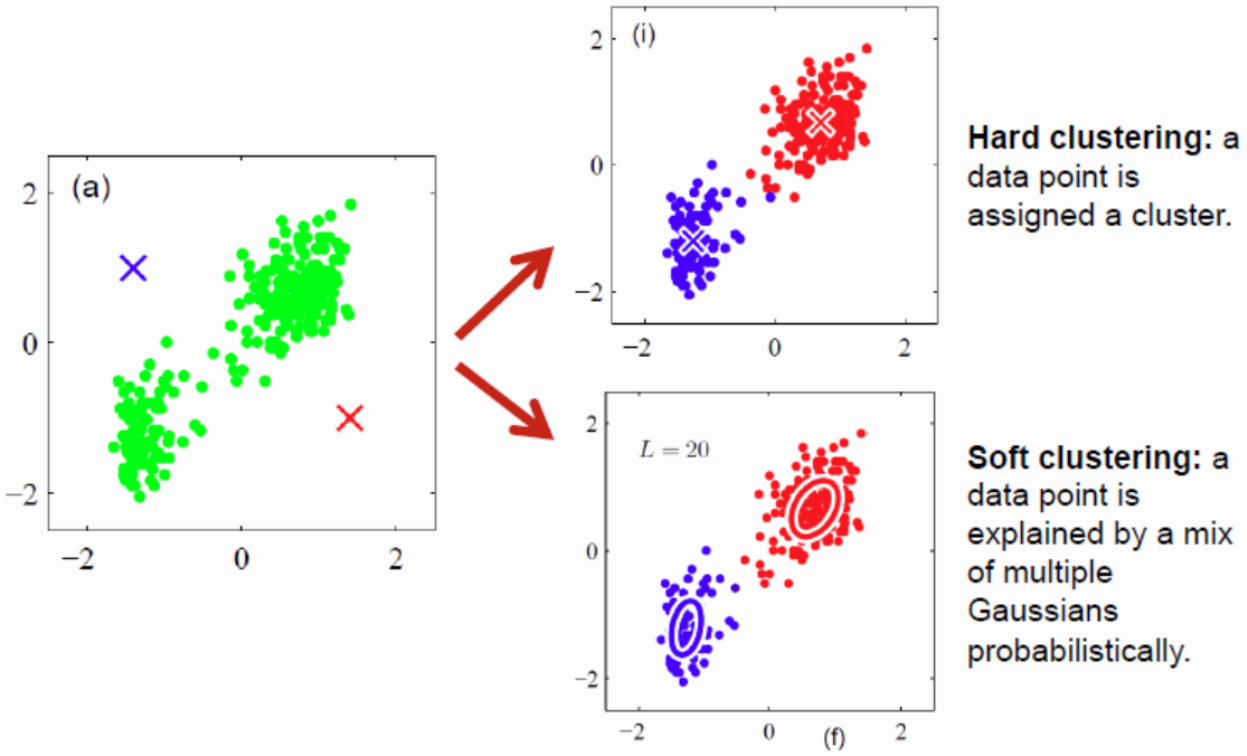
In some ways, **mixture models** are a general way of doing flat-clustering than K-means:

- ***K-means*** is in many ways a specific version of **mixture models**.
- The next lecture will consider a hierarchical clustering model approach, termed **agglomerative clustering**.
- ***K-means*** is an example of an **expectation maximization problem**:
    - This type of algorithm iterates through 2 steps:
        - **Expectation step** → take model and predict outcome
            - For k-means, this is assigning each instance to a cluster.
        - After expectation, we do **maximization,** updating our models to minimize/maximize something.
            - For k-means, we were updating our reference vectors to the mean of the cluster to minimize the distance.
            - Conversely, at times, we maximize the log-likelihood.
        - This expectation/maximization process occurs again and again until we run out of time or there's convergence (ex: we don't see any maximization).

Another **expectation-maximization (EM)** problem is trying to discover the parameters for Gaussians, called **Gaussian mixture models**:

- In our GMM, we're trying to define our model as a mixture of Gaussians.
    - We want to find the set of Gasussians combined that best describe our data → our final "model" is a set of Gaussians (Gaussian mixture model or GMM)
- Instead of identifying clusters by nearest centroids, we fit a set of k Gaussians to our data.
    - Since each gaussian has a mean and variance, the **maximization** step will involve updating the mean and variances values for each Gaussian.
        - Remember in maximiation of k-means, we only moved the location of the reference vectors. The updating in GMMs don't only move the center of the Gaussians but also their variance.
        - The maximization step in this GMM wil not only move the center of the gaussians but also the variance of them.

Visualization of difference b/w k-means and GMM:

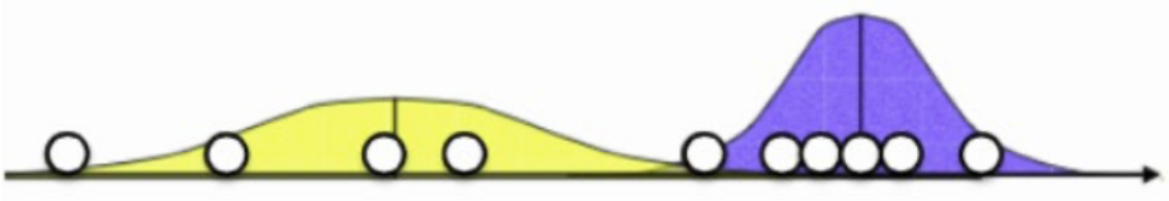


- With k-means , each observation belongs to one and *only one* cluster.
- With GMM, we can say an observation can belong to several (even all) Gaussians, but can belong to each with different probabilities.
    - In the visualization in the bottom right, assignments are made to the cluster with the highest probability.
    - Also, our model does not only have locations, but ALSO have variance/co-variance.

Mixture Models in 1D

To work through this and figure out how to use current Gaussians to make estimations and then use them to update the params of the Gaussians, we can begin with a 1D example to determine how we can use the parameters of the Gaussians in this mixture model:

- Say we want to model our data using two Gaussians, where they look like this:



- Given:
  - Observations $\{X_i\}_{i=1}^{N}$
  - Say we want to model with using $k = 2$ Gaussians
- Assume n observations with 1 feature
  - Goal: Find params of two Gaussians that belong to these two clusters, C_1 and C_2, that maximize the log likelihood of the data (or minimize some distance)

$$\mathcal{N}_1\left(\mu^{(1)}, \sigma^{(1)}\right) \text{ and } \mathcal{N}_2\left(\mu^{(2)}, \sigma^{(2)}\right)$$

The process for finding these Gaussians is similar to K-means:

- Expectation-maximization steps:
  1. Start with Gaussians placed at two random initial locations $\mu^{(1)}$ and $\mu^{(2)}$, as with k-means, but they also need to be initialized with standard deviations from the samples $\sigma^{(1)}$and $\sigma^{(2)}$.
     - Used to initialize the Gaussians:

$$\mathcal{N}_1\left(\mu^{(1)}, \sigma^{(1)}\right) \text{ and } \mathcal{N}_2\left(\mu^{(2)}, \sigma^{(2)}\right)$$

  2. Compute probability of each observation belonging to each Gausssian.
     - To do this, we need to use Bayes' Rule.
     - Initially, we may say that the Gaussian priors are uniform unless there's prior info:

$$P(\mathcal{N}_i) = \frac{1}{k}$$

1. Where k is the number of Gaussians

3. ***Expectation step:***
    a. <u>For each observation:</u>
        - Compute the posterior probabaility (expectation) of each Gaussian, given the observation:
            - We'll compute $P(N_1|x)$ and $P(N_2|x)$
            - We'll see how to do this in the next slide
4. ***Maximization step:***
    a. Upon iterating through all of our observations, we'll use these newly calculated values to adjust the parameters of our Gaussians.

$$\mu^{(1)}, \sigma^{(1)}, \mu^2, \sigma^{(2)}, P(\mathcal{N}_1) \text{ and } P(\mathcal{N}_2)$$

- We continue iterating, repeating steps 3 and 4 until convergence.

- The difficulty comes in with how to compute the posterior probabilities and how do we use them in the maximization step to update our parameters.

<u>For reference:</u>

Let:

- x be the unknown cause
- y be the observed evidence

$$P(x|y) = \frac{P(x,y)}{P(y)} = \frac{P(y|x)P(x)}{\sum_{x \in X} P(x,y)}$$

We call

- $P(x|y)$ the posterior
- $P(y|x)$ the likelihood
- $P(x)$ the prior
- $\sum_{x \in X} P(x,y)$ the evidence

Computing the expectation $P(N_k|x)$:

- Given the current model for each Gaussian, the means and standard deviations, we can use our Gaussian's probability density functions to compute the $p(x|N_k)$, the probability given the model that we generate the observation x.
  - Then, we can use Bayes' rule to calculate $P(x|N_1)$!
    - Note that we initialized the Gaussian priors and have calculated something proportional to the evidence.
      - So, we can compute the numerator of our probabilities: $P(N_k|x) \propto P(N_k)p(x|N_k)$
      - Upon computing all of the numerators of these probabilities for each Gaussian, we can turn it into a true probabilty by dividing by the sum of those numerators across *all classes.*

Let $\rho = \sum_{\mathrm{m}=1}^{\mathrm{K}} P(\mathcal{N}_{\mathrm{m}}) p(x|\mathcal{N}_{\mathrm{m}})$

- The evidence, for normalization

We can then compute $P(\mathcal{N}_k|x)$ as:

$$P(\mathcal{N}_{\mathrm{k}}|x) = \frac{P(\mathcal{N}_{\mathrm{k}}) p(x|\mathcal{N}_{\mathrm{k}})}{\rho}$$

- For k-means, $P(N_k|X_i) \in \{0, 1\}$, where there is stricter clustering happening.
  - For GMM, each cluster has a probability of having an observation.
- We know the model has the current Gaussians' prior as uniform and use our initializations of the Gaussian values to calculate these probabilities.

To maximization (updating the model means and variances):

- As with k-means, we find the new centers (part of the maximization step) AND we also want to compute the average vector of those observations associated with the cluster (from the expectation step).
- After iterating through all of our observations and having the posterior probabilities, we now need to update our model to better align the data.
  - We do this by moving the Gaussians, changing their standard deviation values and priors.
- For each observation X_i, we know the probability that it came from Gaussian k, denoted as $P(N_k|X_i)$ from what we calculated above.
  - Let $P(N_k|X_i)$ be the probability that observation i belongs to cluster k, which we compute in the expectation step.
  - We also have a normalization factor β_k, which is the total sum of the probability of observations belonging to Gaussian N_k.

$$\beta_k = \sum_{i=1}^{N} P(\mathcal{N}_k | X_i),$$

- To then move our Gaussian, we can go over all the posteriors for ecah obsercavtions and do a sort of weighted average of them.
  - With k-means, all we had to do was compute the mean of the observations for a singular cluster.
  - Since this is a GMM, a soft-clustering algorithm, we need to compute a ***weighted mean*** where the weights are the probabilities of them belonging to a certain Gaussian.
- We can then re-write $\mu^{(k)}$ as:

$$\mu^{(k)} = \frac{\sum_{i=1}^{N} P(\mathcal{N}_k | X_i) X_i}{\beta_k}$$

Normalize by sum of probabilities

- Again $P(N_k | X_i)$ is the probability of observation X_i belonging to cluster k.
  - We use this to weight observation X_i, doing this over all observations, ensuring to normalize by the sum of the weighted probabilities, making sure all the values of $P(N_k | X_i)$ add to 1.
  - Again, this is effectively a ***weighted average***.
- We do this for each of our Gaussians, computing their mean as a weighted average of their observations, weighted by probabilities AND the standard deviations.
  - To update the standard deviations, we can also weigh the difference that we take in our regular standard deviation formula with the probability of the observation belonging to a Gaussian.
    - Like taking the regular standard deviation, but taking a weighted average here.

$$\sigma^{(k)} = \sqrt{\frac{\sum_{i=1}^{N}\left(P(\mathcal{N}_k|X_i)\left(X_i - \mu^{(k)}\right)^2\right)}{\beta_k}}$$

acting as a weight

To maximization (updating the priors):

Finally, we need to update the Gaussian priors $P(N_k)$, which is just the average of the probabilities of observations belonging to Gaussian k.

$$P(\mathcal{N}_k) = \frac{\beta_k}{N}$$

Example: 1D GMM

Datapoints:

$$X = [.78, .72, .66, .51, .86, .83, .53, .32, .79, .97]^T$$

Let's assume two Gaussians $k = 2$

Choose two samples as initial means:

$$\mu^{(1)}=.78,\ \mu^{(2)}=.51$$

We can then use these values to compute the standard deviation:

$$\sigma^{(1)} = \sqrt{\frac{\sum_{i=1}^{N}\left(X_i - \mu^{(1)}\right)^2}{N}} = 0.2025., \sigma^{(2)} = 0.2628$$

To initialize our Gaussian priors, we'd simply assume a uniform distribution of classes (initially):

$$P(\mathcal{N}_1) = P(\mathcal{N}_2) = 0.5$$

We begin the ***expectation-maximization problem***!

- We go through each obseravtion and compute the probability of the *Gaussian* given that observation:
  - <u>For data point 0.78:</u>
    - We'd use our norm pdf function to calculate this value proportional to the evidence with both Gaussians (using their respective mean and standard deviation values):

$$p(0.78|\mathcal{N}_1) = \frac{1}{\sigma^{(1)}\sqrt{2\pi}} e^{-\frac{\left(0.78-\mu^{(1)}\right)^2}{2\sigma^{(1)^2}}}$$
$$p(0.78|\mathcal{N}_1) = 1.97$$
$$p(0.78|\mathcal{N}_2) = 0.8955$$

    - Now, we can compute the numerators of the posteriors by multiplying these normpdf values to the prior values, which are 0.5.

$$P(\mathcal{N}_1|0.78) \propto p(0.78|\mathcal{N}_1)P(\mathcal{N}_1) = 0.985$$
$$P(\mathcal{N}_2|0.78) \propto p(0.78|\mathcal{N}_2)P(\mathcal{N}_2) = 0.4478$$

- To arrive at a true probability value, we normalize these values so $P(N_1|0.78) + P(N_2|0.78) = 1$

$$P(\mathcal{N}_1|0.78) = 0.6875$$
$$P(\mathcal{N}_2|0.78) = 0.3125$$

Prob that the observation belongs to these Gaussians

- Based on how we initially placed our Gaussians and their initial standard deviations, this is the current association with each Gaussian.

- We continue this process for all other observations!

- As for the ***maximization step***, now we must update our models to better align with the expectation.
  - For this, we need our normalization factors here, where we sum over all of our observations and compute the sum of each Gaussian's posterior.
    - We begin here with that of the first Gaussian:

$$\beta_1 = \sum_{n=1}^{N} P(\mathcal{N}_1|X_n)$$

  - Then, we can compute the new mean as the weighted average of all locations weighed by their posterior with reference to the normalization factor:
    - Again, for the first Gaussian:

$$\mu^{(1)} = \frac{\sum_{n=1}^{N}(P(\mathcal{N}_1 \mid X_n)X_n)}{\beta_1}$$

        - For each observation, we know the observation's value and in the expectation step, we computed the posterior accordingly.
    - We do the same with the standard deviation update, weighing it by posterior probabilities:
        - For the first Gaussian:

$$\sigma^{(1)} = \sqrt{\frac{\sum_{n=1}^{N}(P(\mathcal{N}_1 \mid X_n)(X_n - \mu^{(1)})^2}{\beta_1}}$$

    - We also update our prior values:
        - For the first Gaussian:

$$P(\mathcal{N}_1) = \frac{\beta_1}{N}$$

    - We continue this process with the 2nd normal model.
        - Now we've updated μ and σ values, moving the Gaussians and their spread. We use these new values to go back and compute expectations again!
- Overall, we continue this process until convergence!

We can note here that this seems like a generalized version of k-means.

- Again, with k-means, the probabilities would be either 0 or 1, whereas for GMM, they can be any floating point value.

Mixture Models in D > 1

However, our data is often more than one dimensional.

- If we have more than one dimension of data , then we must learn the parameters of multi-variate normals, and therefore need:
  - Prior: $P(N_k)$
  - Mean vector: $\mu^{(k)}$ for length $1 \times D$
    - Mean
  - Covariance matrix: $\Sigma^{(k)}$ of size $D \times D$
  - Estimate $p(\mathbf{x}|N_k)$:

$$p(\boldsymbol{x}|\mathcal{N}_k) = \frac{1}{\sqrt{(2\pi)^D |\Sigma_k|}} \exp\left(-\frac{\left(\boldsymbol{x}-\boldsymbol{\mu}^{(k)}\right)^T \Sigma_k^{-1}\left(\boldsymbol{x}-\boldsymbol{\mu}^{(k)}\right)}{2}\right)$$

    - The key difference here is in the evidence we're computing.
      - Instead of having the sigma terms for univariate, we now have our Σ covariate matrix.
- This multidimensional problem will now be modeled by multidimensional Gaussians.
  - Each of our model's Gaussians will have a mean vector instead of a mean value.
    - As with k-means, we can initialize our mean vectors similarly in several ways.
  - Each of our model's Gaussians will have a covariance matrix rather than a standard deviation value.
- The ***maximization part*** of mixture models here ends up being relatively the same, where the only difference is that we'll update each feature.
  - We'll go through each feature and weigh it by its posterior for the Gaussian, normalizing by the normalization factor.

$$\mu_j^{(k)} = \frac{\sum_{i=1}^{N} P(\mathcal{N}_k | X_i) X_{i,j}}{\beta_k}$$

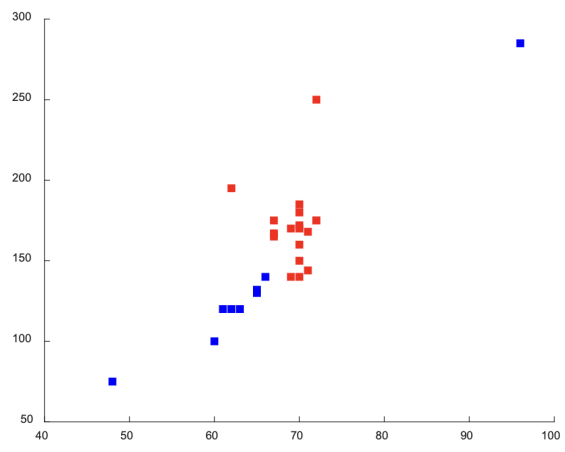
- We functionally do the same with the covariance matrix:

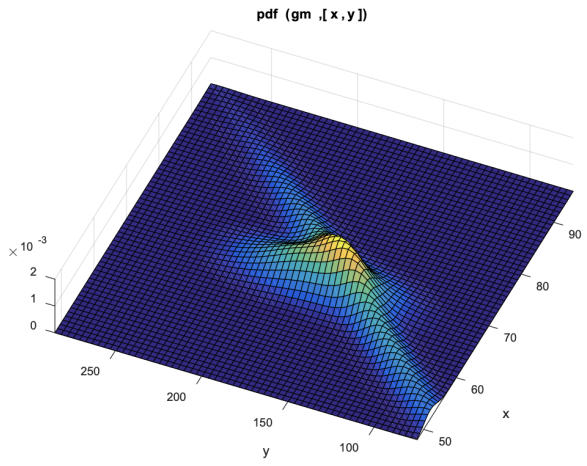
$$(\Sigma_k)_{j,t} = \frac{\sum_{i=1}^{N} P(\mathcal{N}_k | X_i) \left(X_{i,j} - \mu_j^{(k)}\right)\left(X_{i,t} - \mu_t^{(k)}\right)}{\beta_k}$$

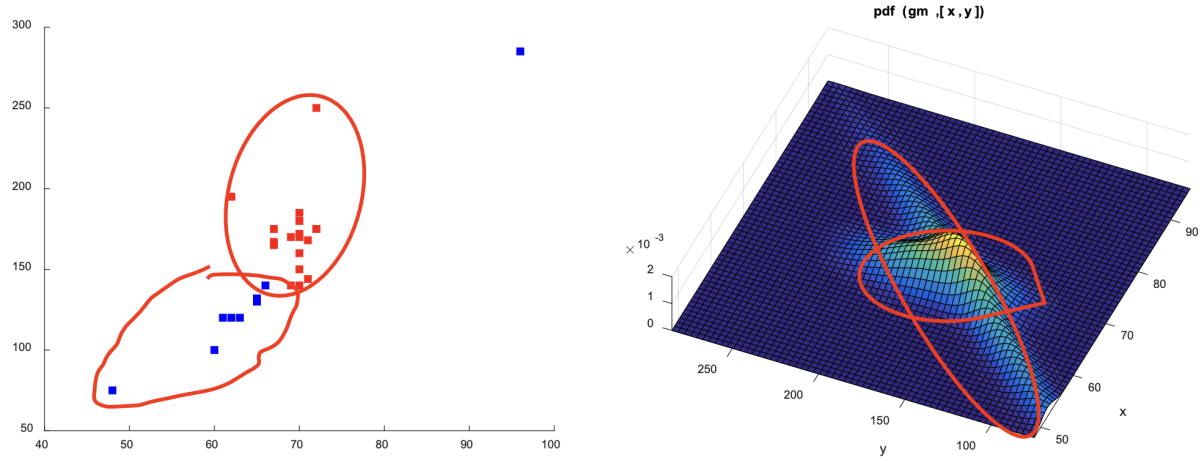
- And we do the same for the prior as well:

$$P(\mathcal{N}_k) = \frac{\beta_k}{N}$$

Example of $k = 2$ cluster multivariate GMM

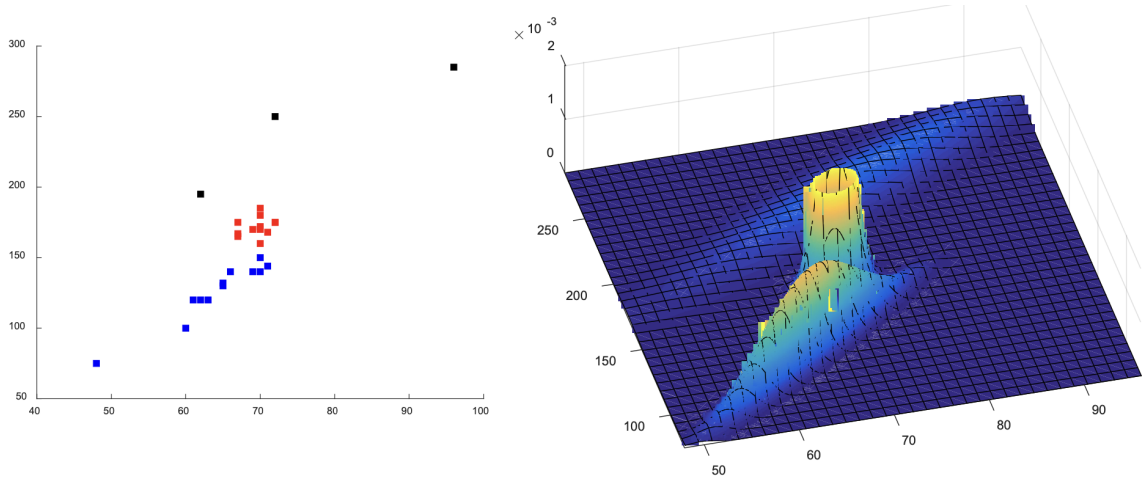


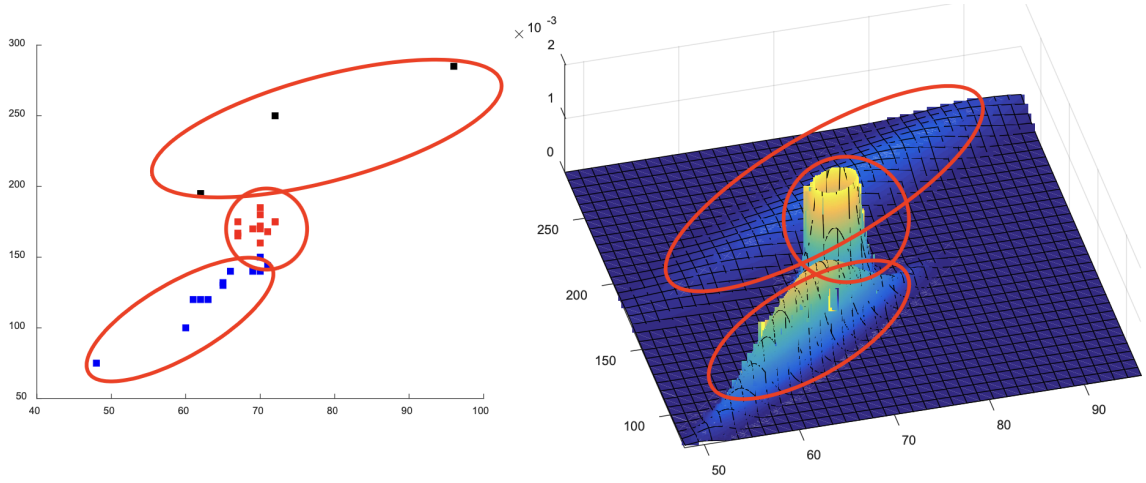




The final Gaussians learned can be visualized in the graphs, where we can see an elongated Gaussian, which is encoded on the left graph based on the maximum posterior.

Example of $k = 3$ clusters multivariate GMM





Again, GMMs is a more generalized version of k-means, where we have a soft clustering that each observation has a probability of belonging to a certain cluster.

- Clusters now not only have a mean, but also have covariance, allowing us to map it with a Gaussian.

Using Mixture Models

Once we've trained and learned the Gaussians that best fit our data, we can take in an incoming observation and compute the posterior of belonging to each Gaussian:

- If we want soft-to-hard clustering, we can associate the observation with the Gaussian which gave it the highest posterior value.

> Now that we have the parameters for a Gaussian for each class, given an observation $\boldsymbol{x}$ we can compute its probability of belonging to each class using the evidence:

$$\rho = \sum_{\mathrm{m}=1}^{\mathrm{K}} P(\mathcal{N}_{\mathrm{m}}) p(\boldsymbol{x}|\mathcal{N}_{\mathrm{m}})$$

$$P(\mathcal{N}_{\mathrm{k}}|\boldsymbol{x}) = \frac{P(\mathcal{N}_{\mathrm{k}}) p(\boldsymbol{x}|\mathcal{N}_{\mathrm{k}})}{\rho}$$

Expectation-Maximization Summary

The idea behind GMMs is that we decided to model our data as a set of Gaussians.

- This would work modeling the data using different sorts of models as well.
- Advantages:
  - Assuming the distirbution of the data works well with the GMM, it works very well.
- Disadvantages:
  - Have to choose the type of model
    - If chosen wrong, we may get distorted results
- In general, ***expectation-maximization*** problems are a general algorithm for parameter estimation.
  - We begin with some guesses, use them to make predictions, use predictions to update parameters to better fit our data.
    - In some ways, there are similarities to gradient descent, but they are NOT the same.
      - There are similarities in the way we use current weights to compute $\hat{y}$ as in linear and logistic regressions, almost acting as our expectations.
        - We then use these $\hat{y}$ and y values to then update our weights accordingly.
