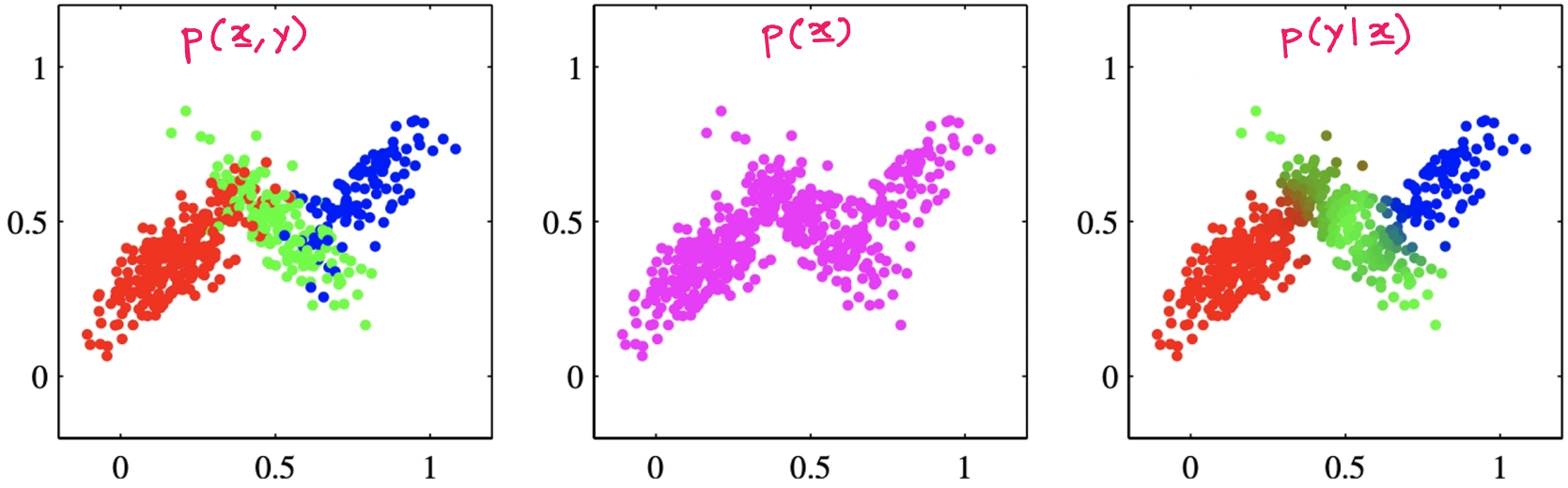
Recap of Generative Models

- In the last lecture, we introduced the concept of generative models that models the joint distribution $p(x, y)$
- We used GMM to model the joint distribution $p(x, y)$
- In supervised learning, the GMM was learned using both input-output data

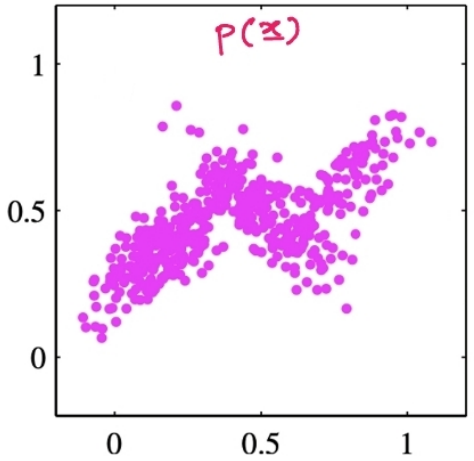


E.g. Mixture of 3 Gaussians

Unsupervised Learning: Cluster Analysis

- In unsupervised learning, all the datapoints are unlabelled
  - Training data, $\mathcal{T} = \{\underline{x}^{(i)}\}_{i=1}^{N}$
  - Objective: Build a model that can be used to find interesting patterns in data

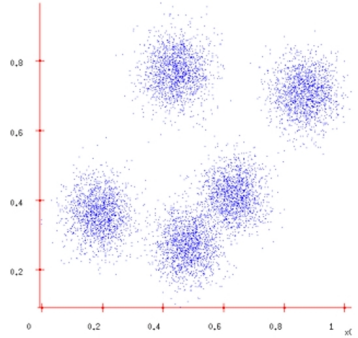
    In other words, build a model of the distribution $p(\underline{x})$



- Clustering → finding groups of similar $\underline{x}$ values in the data space

Ex: You are interested in studying the flow pattern of a viscous fluid from lots of videos of flow at different Re's

You want to group the videos into meaningful groups, so that you can look at each cluster and figure out what is the predominant pattern



GMM for clustering

- The GMM introduced in supervised learning is a joint model for $\underline{x}$ & y

$$p(\underline{x}, y) = p(\underline{x}|y)\, p(y) = \mathcal{N}(\underline{x} \mid \underline{\mu}_y, \underline{\underline{\Sigma}}_y)\, \pi_y$$

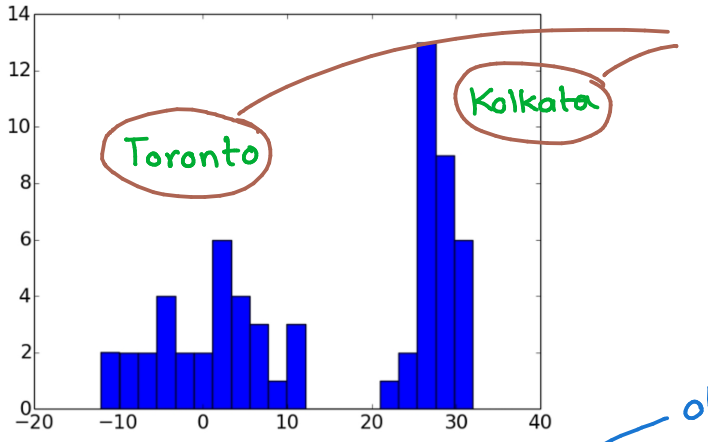
- To obtain a model only for input $\underline{x}$, use marginalization

$$p(\underline{x}) = \sum_y p(\underline{x}, y)$$

- Since the output y is not observed, we will treat it as a LATENT variable

latent r.v. $\leftrightarrow$ exists in the model but not observed in data

Example:



latent variables, if you are only given the temperature values and the city

Histogram of temperatures

observed variable

Maximum Likelihood with latent variables

- How should we choose the parameters $\underline{\theta} = \{\pi_m, \underline{\mu}_m, \underline{\underline{\Sigma}}_m\}_{m=1}^{M}$
- We will use Maximum likelihood principle $\longrightarrow$ Choose parameters that maximize the likelihood of observed data
- Note that, we don't observe the cluster labels y, we only get the input $\underline{x}$
- Given data $\mathcal{T} = \{\underline{x}^{(i)}\}_{i=1}^{N}$, choose parameters to maximize:

$$\hat{\underline{\theta}} = \arg\max_{\underline{\theta}} \ln p\left(\{\underline{x}^{(i)}\}_{i=1}^{N} \mid \underline{\theta}\right)$$

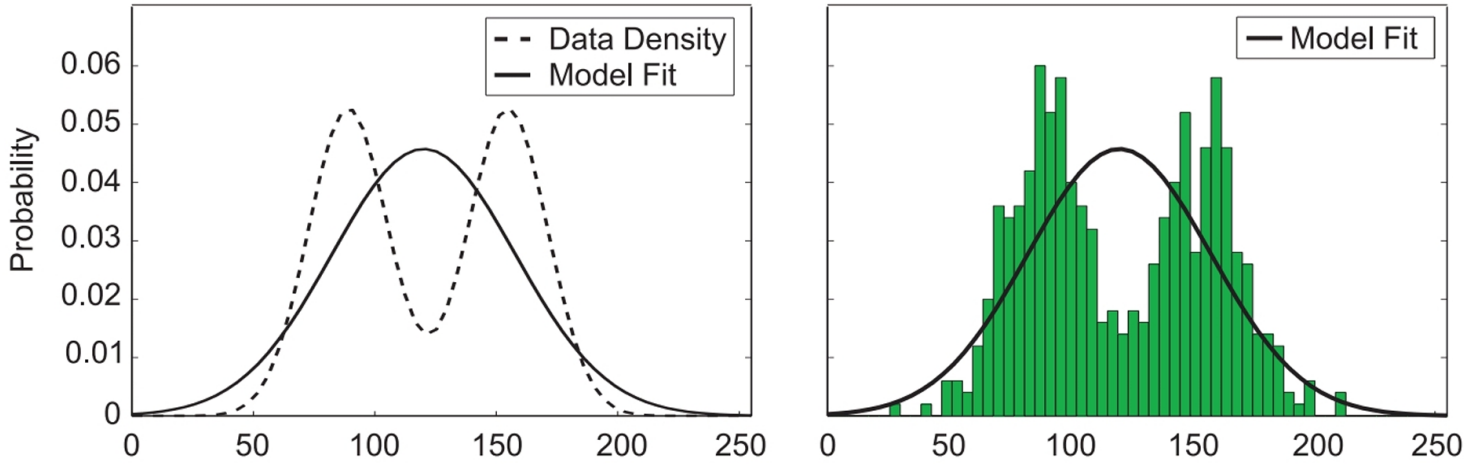
$$= \arg\max_{\underline{\theta}} \sum_{i=1}^{N} \ln p\left(\underline{x}^{(i)} \mid \underline{\theta}\right) \quad \text{[Assuming independence of inputs]}$$

- We can find $p(\underline{x})$ by marginalizing out y:

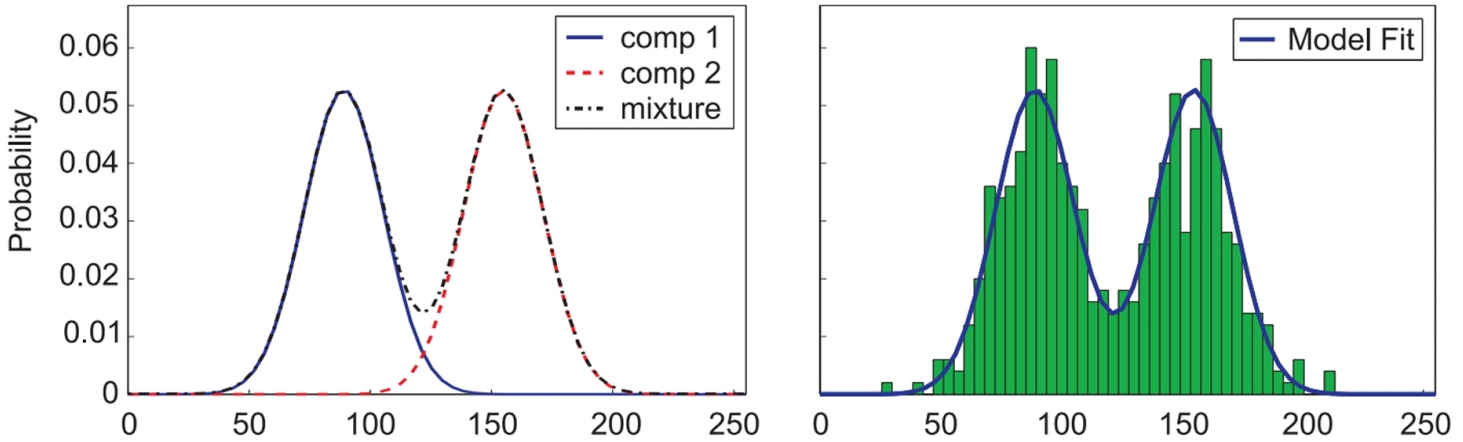
$$p(\underline{x}) = \sum_{m=1}^{M} p(\underline{x}, y=m) = \sum_{m=1}^{M} p(y=m)\, p(\underline{x} \mid y=m)$$

Visualizing a mixture of Gaussians (1D - Gaussians)

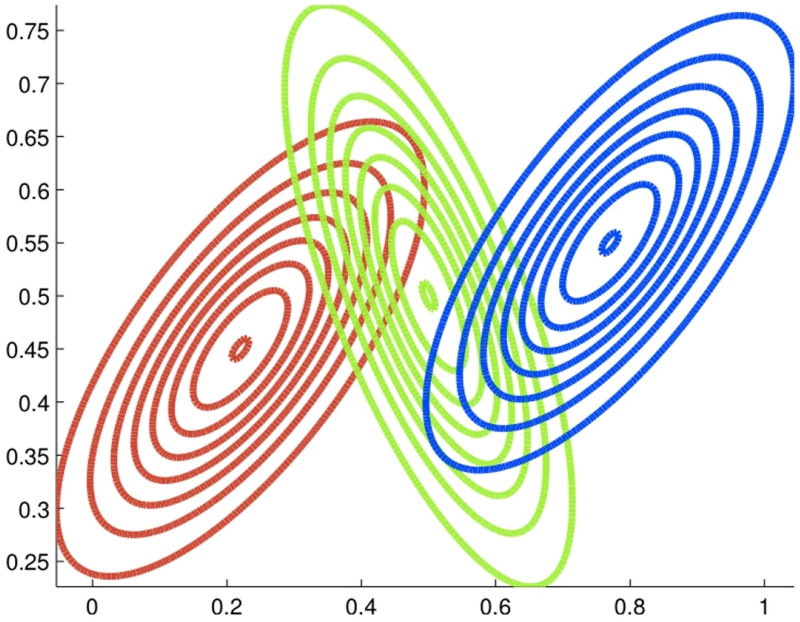
- If you fit a single Gaussian to bimodal data:

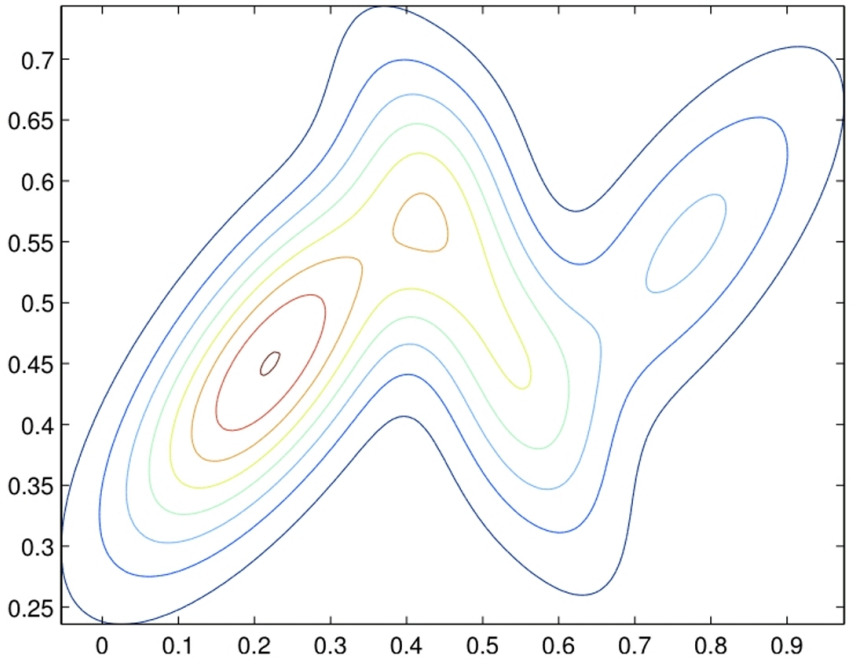


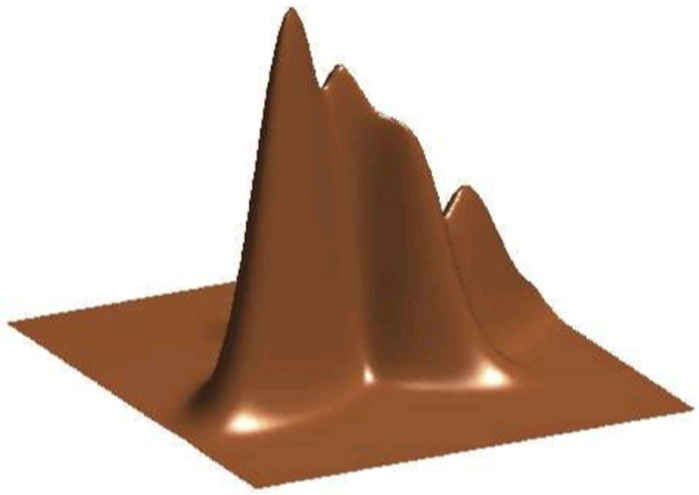
- If you fit a GMM with M=2 (or 2 Gaussians)



Visualizing a mixture of Gaussians (2D - Gaussians)







Fitting GMMs for unsupervised learning

- Notation: $\underline{\theta} = \{\pi_m, \underline{\mu}_m, \underline{\underline{\Sigma}}_m\}_{m=1}^{M}$, $\underset{N\times P}{\underline{\underline{X}}} = \{\underline{x}^{(i)}\}_{i=1}^{N}$, $\underset{N\times 1}{\underline{y}} = \{y^{(i)}\}_{i=1}^{N}$

- The maximum likelihood objective:

$$\underset{\underline{\theta}}{\operatorname{argmax}} \ln p(\underline{\underline{X}} \mid \underline{\theta}) = \underset{\underline{\theta}}{\operatorname{argmax}} \sum_{i=1}^{N} \ln \left(\sum_{m=1}^{M} \pi_m \mathcal{N}(\underline{x}^{(i)} \mid \underline{\mu}_m, \underline{\underline{\Sigma}}_m) \right)$$

- In general, there is no closed-form solution of the above optimization unlike the supervised learning case

- Use numerical iterative optimization
  - Gradient descent will face challenges:
    * Non-convex
    * Need to enforce non-negativity constraint on π_m and PSD constraint on $\underline{\underline{\Sigma}}_m$
    * Derivatives of $\underline{\underline{\Sigma}}_m$ are expensive

A Different Approach for optimization

- Estimating (or inferring) the latent variable $\underline{y}$ with known $\underline{\theta}$

Alternating steps

- Estimating the GMM parameters $\underline{\theta}$ with known latent variable $\underline{y}$

Estimating (or inferring) the latent variable $\underline{y}$ with known $\underline{\theta}$

- If we knew the parameters $\underline{\theta} = \{\pi_m, \underline{\mu}_m, \underline{\Sigma}_m\}_{m=1}^{M}$, we could infer which component a data point $\underline{x}^{(i)}$ probably belongs to by inferring $y^{(i)}$ (i.e Gaussian component)

- This is just finding: $p(y^{(i)} = m \mid \underline{x}^{(i)})$

$$p(y^{(i)} = m \mid \underline{x}^{(i)}) = \frac{p(y^{(i)} = m)\; p(\underline{x}^{(i)} \mid y^{(i)} = m)}{\sum_{j=1}^{M} p(y^{(i)} = j)\; p(\underline{x}^{(i)} \mid y^{(i)} = j)}$$

$p(y^{(i)} = m) \rightarrow \pi_m$

$p(\underline{x}^{(i)} \mid y^{(i)} = m) \rightarrow \mathcal{N}(\underline{x}^{(i)} \mid \underline{\mu}_m, \underline{\Sigma}_m)$

A Different Approach for optimization

- Estimating (or inferring) the latent variable $\underline{y}$ with known $\underline{\theta}$

Alternating steps

- Estimating the GMM parameters $\underline{\theta}$ with known latent variable $\underline{y}$

Estimating the GMM parameters $\underline{\theta}$ with known latent variable $\underline{y}$

- If somehow we knew the latent variable $\underline{y}$, we could simply maximize the likelihood of the joint distribution:

$$\ln p\left(\{\underline{x}^{(i)}, y^{(i)}\}_{i=1}^{N} \mid \underline{\theta}\right) = \sum_{i=1}^{N} \sum_{m=1}^{M} \mathbb{I}\{y^{(i)} = m\} \left\{\ln \mathcal{N}\left(\underline{x}^{(i)} \mid \underline{\mu}_m, \underline{\underline{\Sigma}}_m\right) + \ln p(y^{(i)} \mid \underline{\theta})\right\}$$

- Optimized parameters:

$$\hat{\pi}_m = \frac{n_m}{N}, \quad \hat{\underline{\mu}}_m = \frac{1}{n_m} \sum_{i: y^{(i)} = m} \underline{x}^{(i)}, \quad \hat{\underline{\underline{\Sigma}}}_m = \frac{1}{n_m} \sum_{i: y^{(i)} = m} (\underline{x}^{(i)} - \hat{\underline{\mu}}_m)(\underline{x}^{(i)} - \hat{\underline{\mu}}_m)^T$$

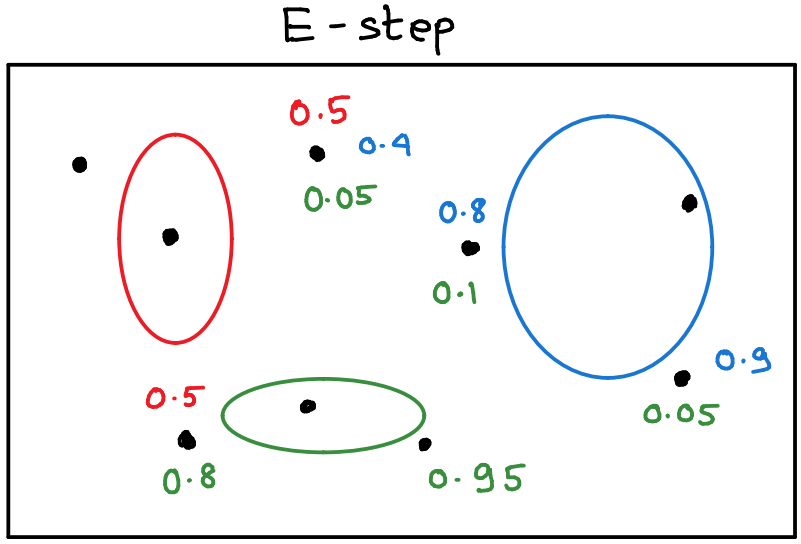
Expectation - Maximization (EM) algorithm

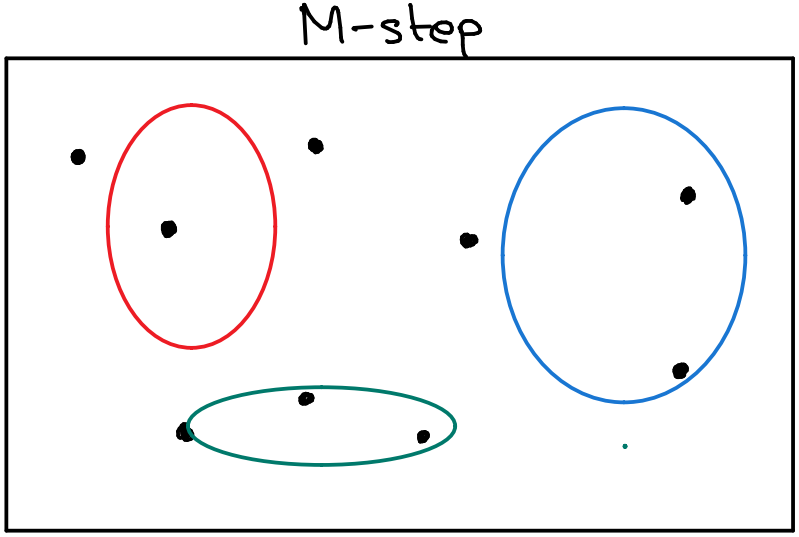
- We will use the EM algorithm that alternates between the two steps:

  Expectation step (E-step): Compute the probability over $\underline{y}$ given we know our current model parameters $\underline{\theta}$

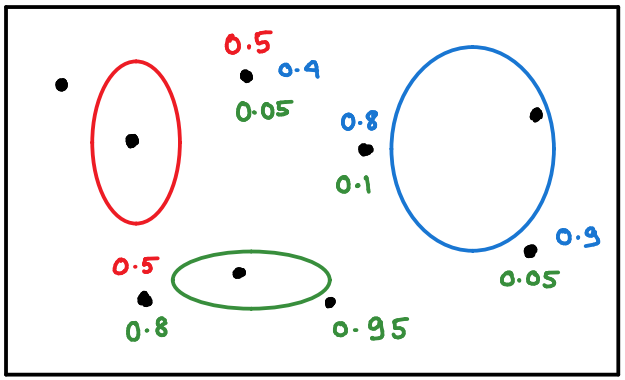
  i.e. how much do we think each Gaussian generates each datapoint

  Maximization step (M-step): Assuming that the complete data was generated this way, tune the parameters of each Gaussian to maximize the likelihood of the complete data





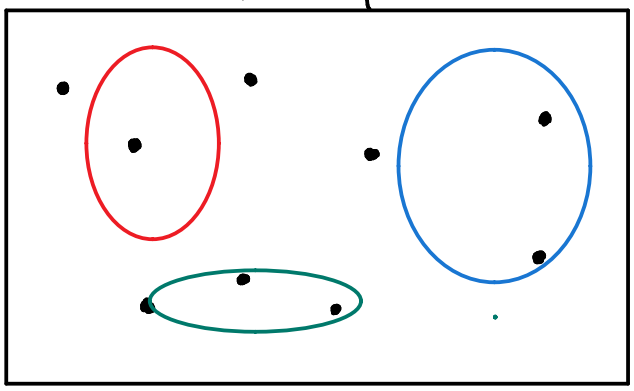
- E-step



- Assign the responsibility $w_m^{(i)}$ of component m for each data point i using probability

$$w_m^{(i)} = p\left(y^{(i)} = m \mid \underline{x}^{(i)}, \underline{\theta}\right)$$

- M-step



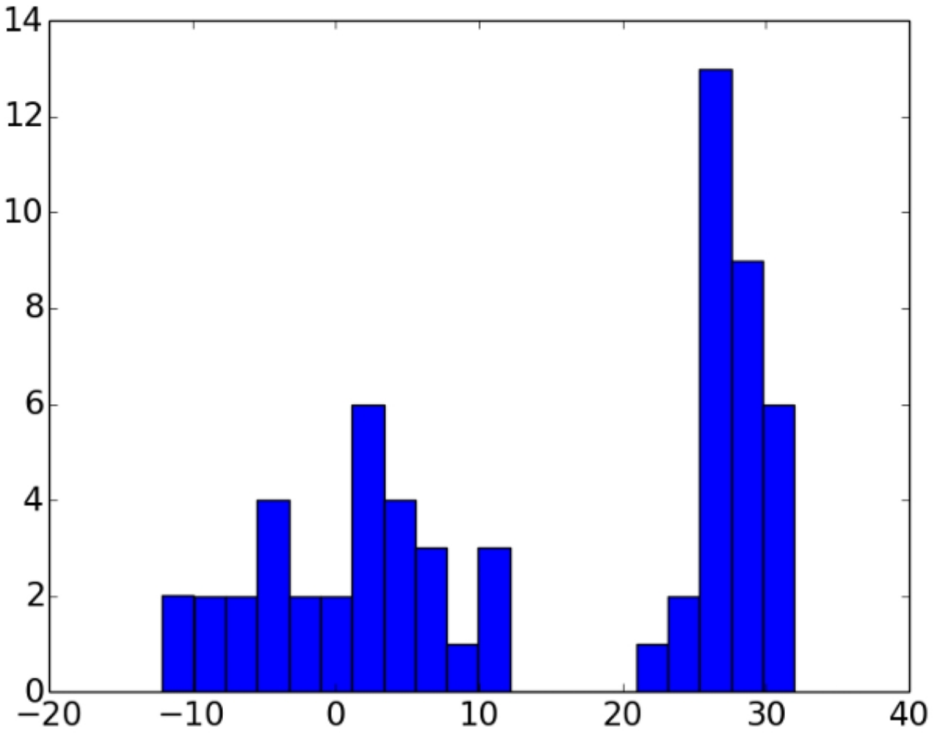
- Apply maximum likelihood updates, where the parameters of each Gaussian component is fit with a weighted dataset

$$\hat{\pi}_m = \frac{1}{N} \sum_{i=1}^{N} w_m^{(i)}$$

$$\hat{\underline{\mu}}_m = \frac{1}{\sum_{i=1}^{N} w_m^{(i)}} \sum_{i=1}^{N} w_m^{(i)} \underline{x}^{(i)}$$

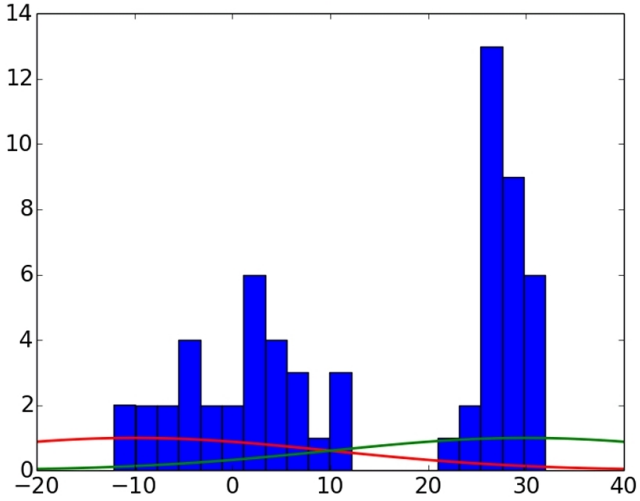
$$\hat{\underline{\underline{\Sigma}}}_m = \frac{1}{\sum_{i=1}^{N} w_m^{(i)}} \sum_{i=1}^{N} w_m^{(i)} \left(\underline{x}^{(i)} - \underline{\mu}_m\right)\left(\underline{x}^{(i)} - \underline{\mu}_m\right)^T$$

Example : Suppose you have recorded a bunch of temperatures in March for Toronto and Kolkata, but forgot which was which, and they are all jumbled up together



Lets try to separate them using a mixture of Gaussians with EM

- Choose the number of Gaussians, $M = 2$
- Randomly initialize the parameters $\underline{\theta}$

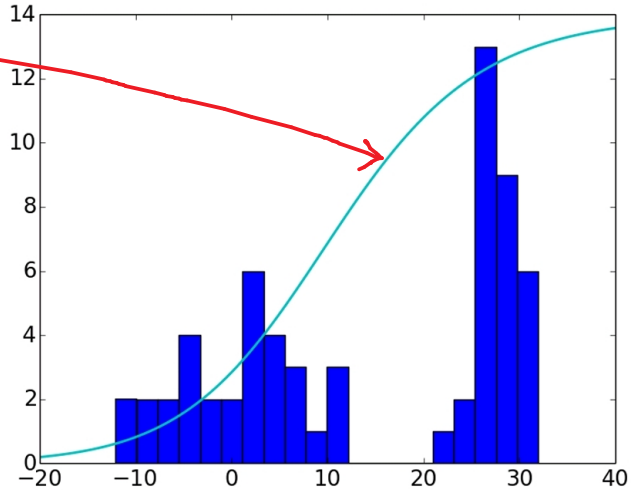


- Iteration 1: E-step and M-step

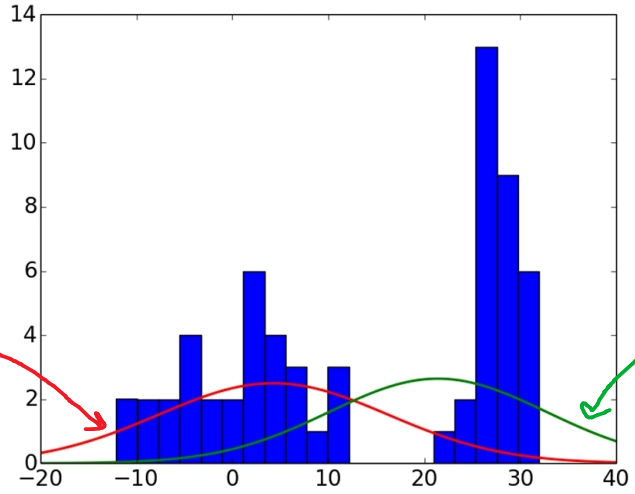
E-step

M-step

$p(y=2|\underline{x})$



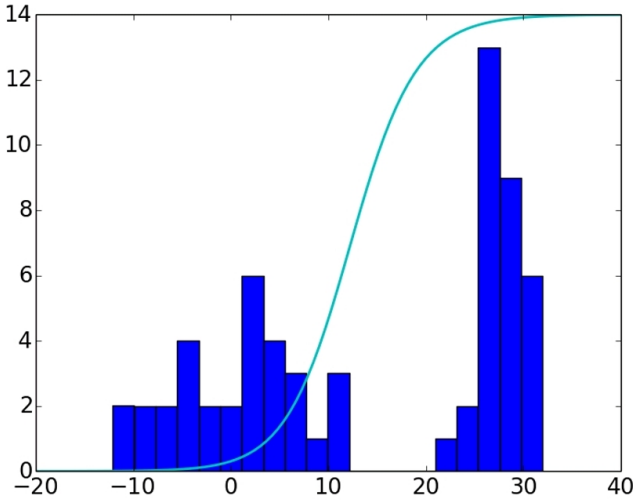
Gaussian component 1



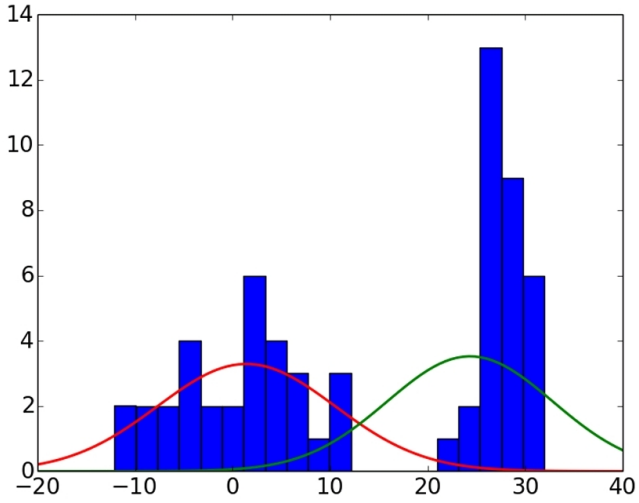
Gaussian component 2

- Iteration 2

E-step



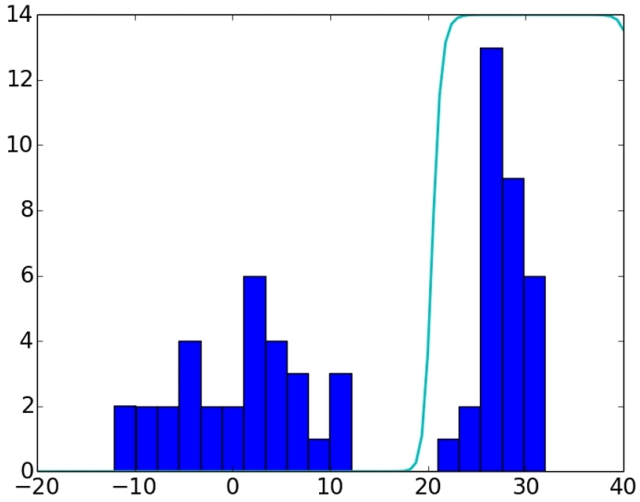
M-step



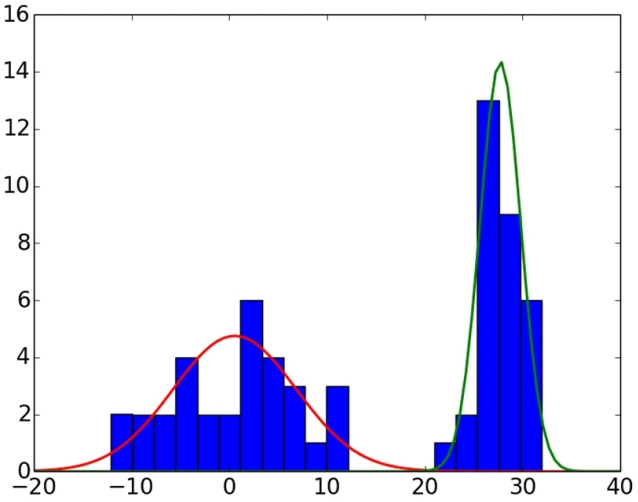
⋮ ⋮

- Iteration 10

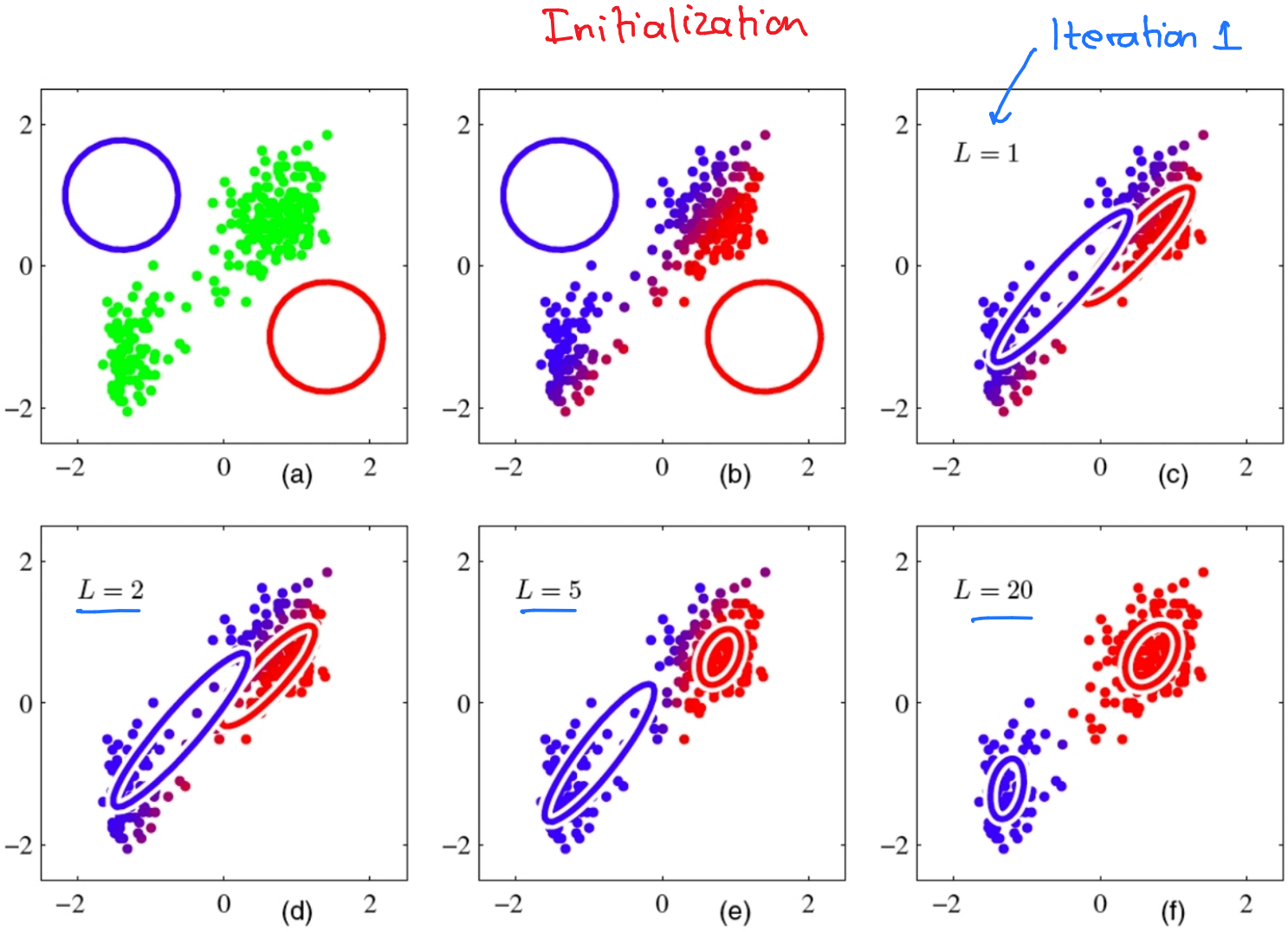
E-step



M-step



EM for multivariate Gaussians (2D)



Few points to be remembered

- The number of clusters (i.e. the # of Gaussian components M) has to be specified before running the algorithm
  - M is a hyperparameter in GMM for clustering

- EM is a local optimizer and hence is only guaranteed to converge to a local optima / saddle point
  - Poor initialization can result in convergence to poor local optimum
  - Use random initializations