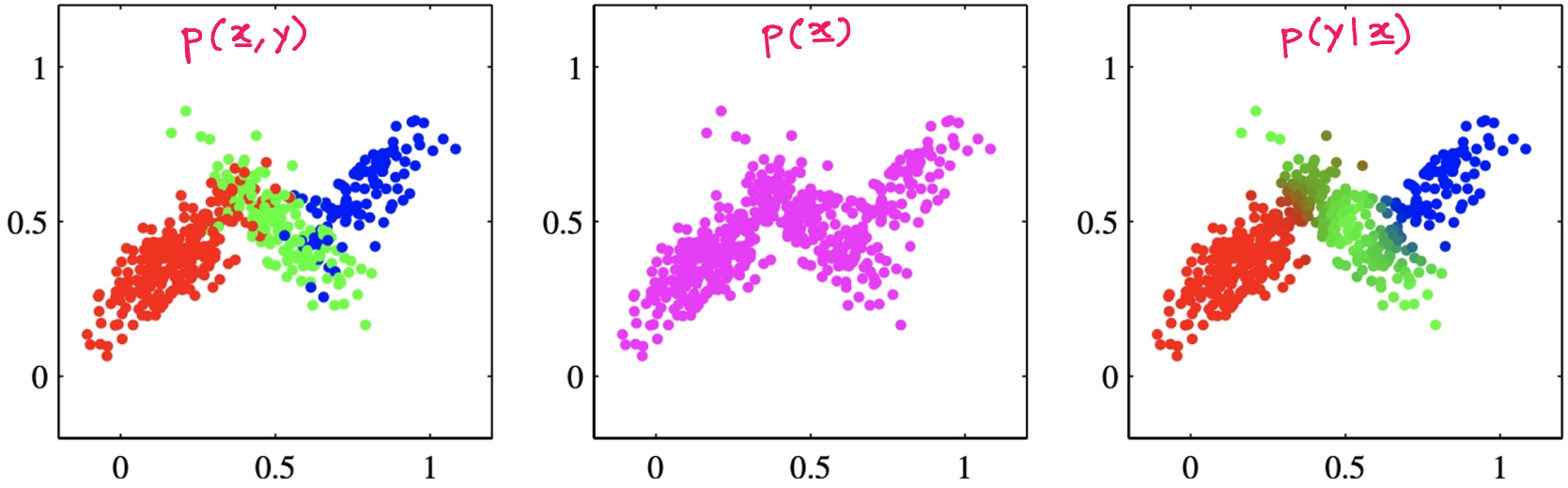
# Recap of Generative Models

- 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

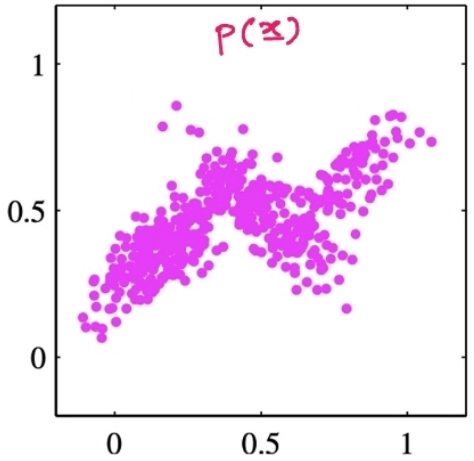



E.g. Mixture of 3 Gaussians

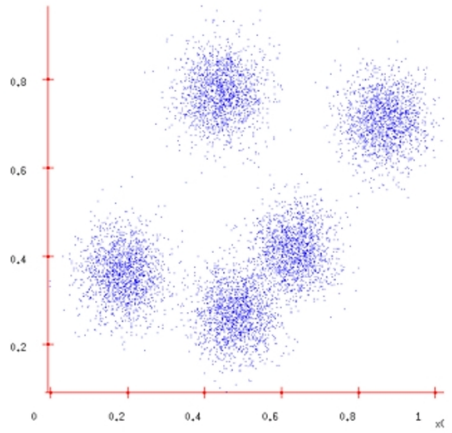
# Unsupervised Learning: Cluster Analysis

- 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



# 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

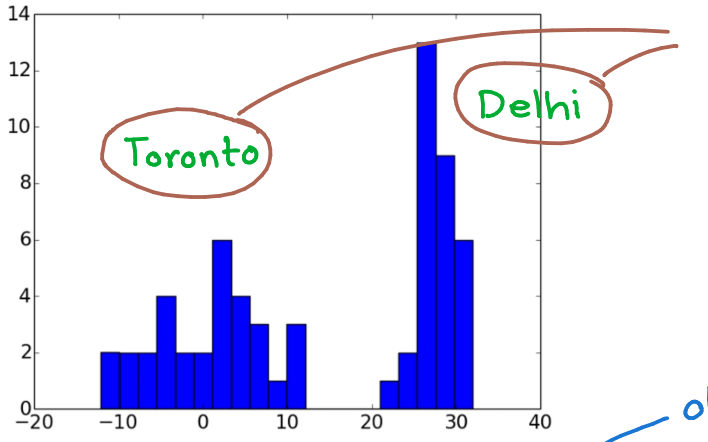
$$p(\underline{x}) = \sum_y p(\underline{x}, y)$$

summation because $y$ is a discrete-valued class

- 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:

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

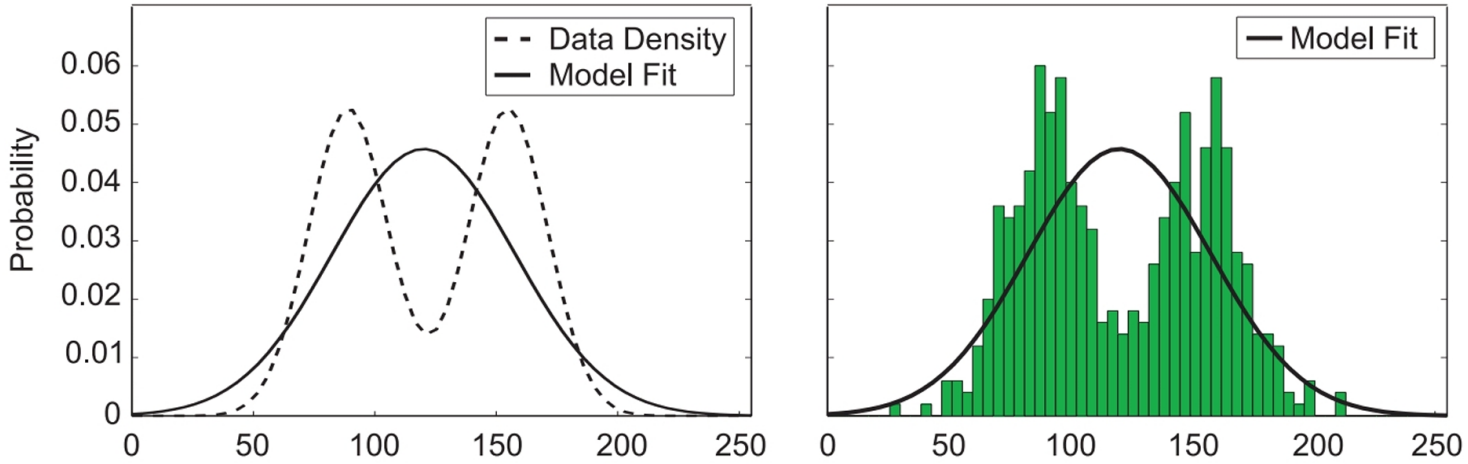
$$= \arg\max_{\underline{\theta}} \sum_{i=1}^{N} \ln p( \underline{x}_i \mid \underline{\theta} ) \qquad \text{[ Assuming independence of inputs ]}$$

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

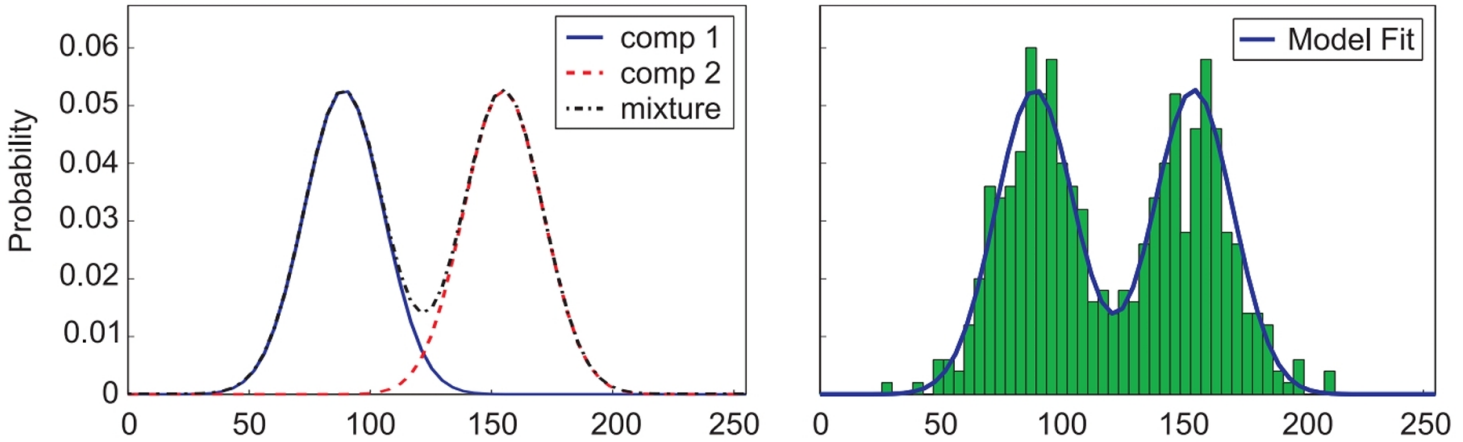
$$p(\underline{x}) = \sum_{m=1}^{M} p(\underline{x}, y=m) = \sum_{m=1}^{M} p(y=m)\, p(\underline{x} \mid y=m)$$

# Visualizing a mixture of Gaussians (1D - Gaussians)

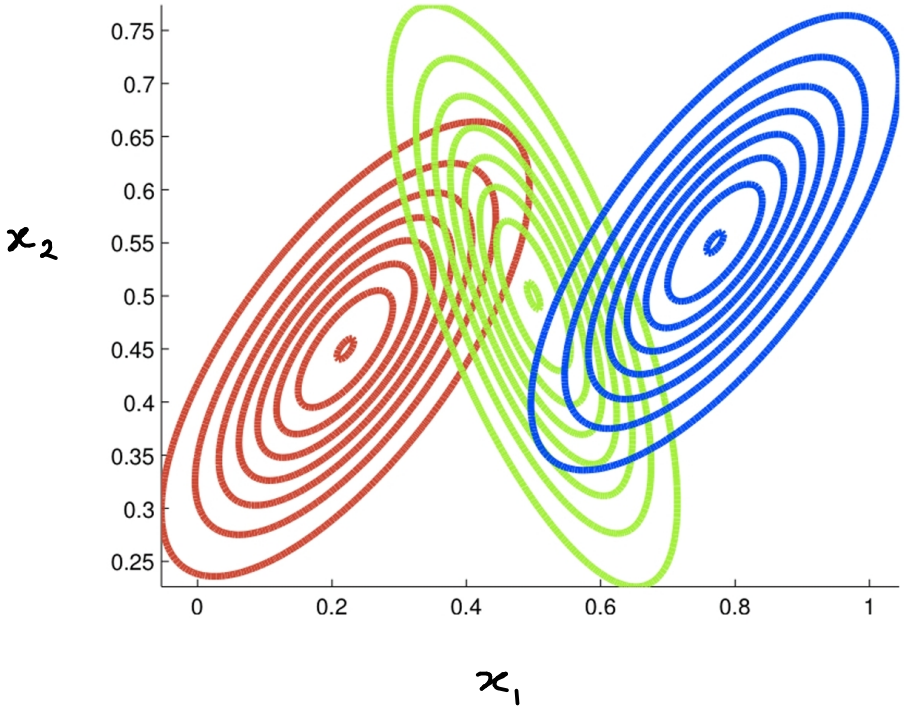
- If you fit a single Gaussian to bimodal data:

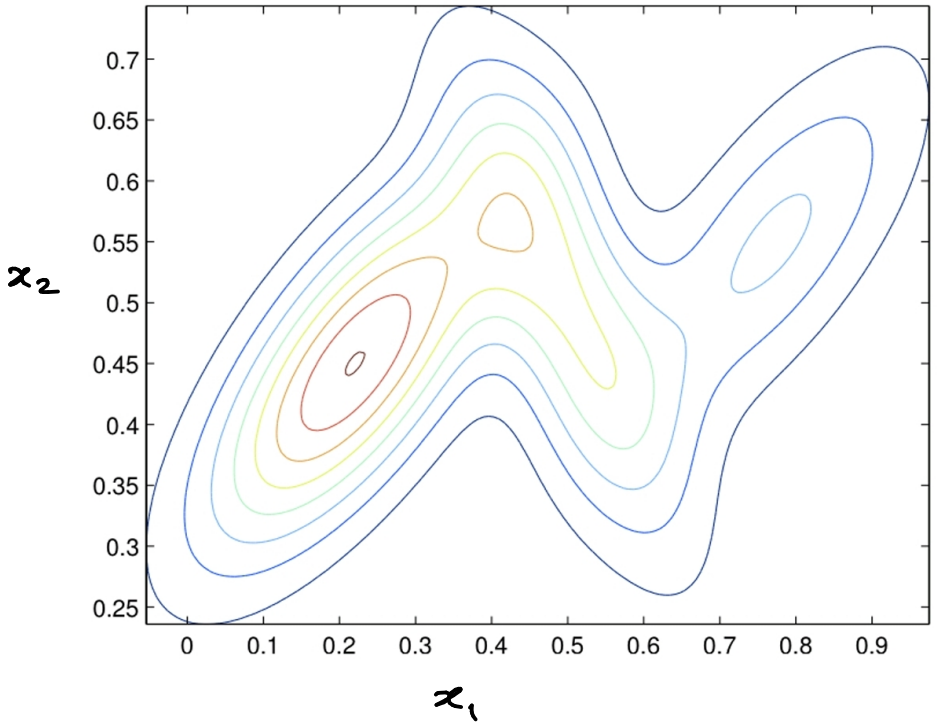


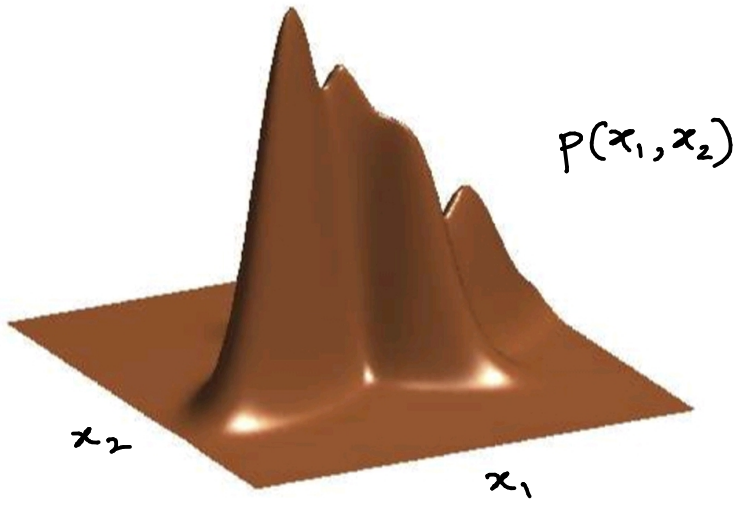

- If you fit a GMM with M=2 (or 2 Gaussians)



# 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:

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

- 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 $\pi_m$ 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:

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

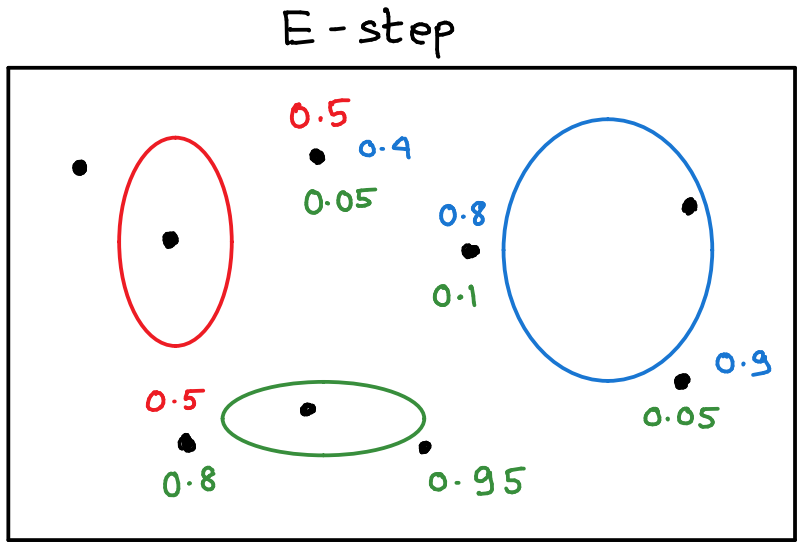
- Optimized parameters (from closed-form):

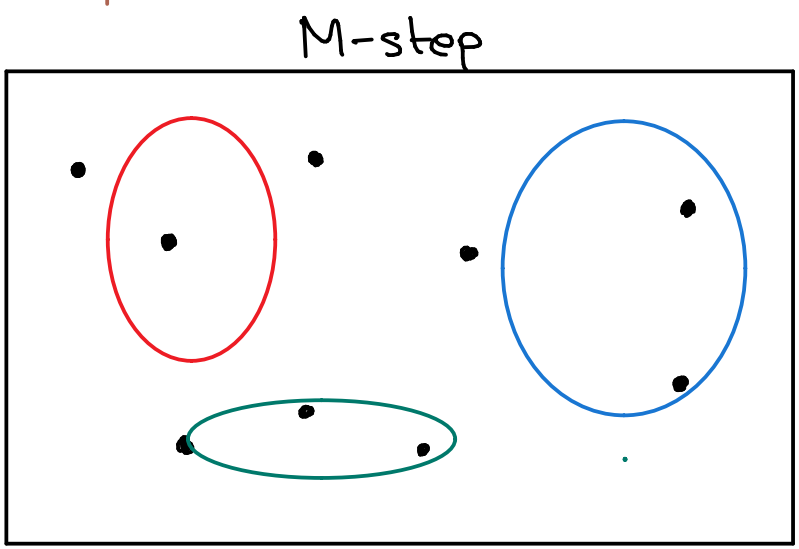
$$\hat{\pi}_m = \frac{n_m}{N}, \quad \hat{\underline{\mu}}_m = \frac{1}{n_m} \sum_{i:\, y_i = m} \underline{x}_i, \quad \hat{\underline{\underline{\Sigma}}}_m = \frac{1}{n_m} \sum_{i:\, y_i = m} (\underline{x}_i - \hat{\underline{\mu}}_m)(\underline{x}_i - \hat{\underline{\mu}}_m)^T$$

# Expectation - Maximization (EM) algorithm

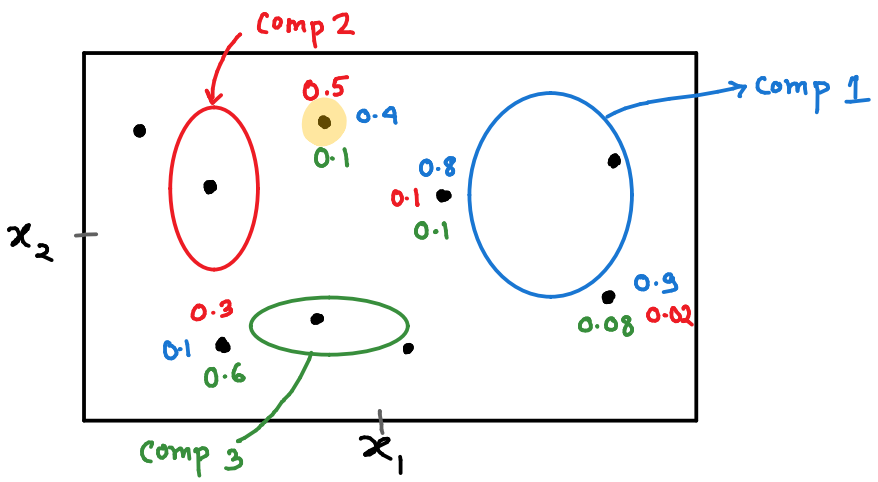
- We will use the EM algorithm that alternates between the two steps:
  - Expectation step (E-step): Compute the probability over $\underline{y}$ given we know our current model parameters $\underline{\theta}$

    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

$$w_m^{(i)} = p(y_i = m \mid \underline{x}_i, \underline{\theta})$$

Example: Suppose we observe $\underline{x}_i = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -0.1 \\ 0.5 \end{bmatrix}$

$$p(y=1) = 0.4 \quad (\pi_1)$$

$$p(\underline{x}_i \mid y=1) = \mathcal{N}\left(\begin{bmatrix} -0.1 \\ 0.5 \end{bmatrix}; \underbrace{\begin{bmatrix} 0.8 \\ 0.2 \end{bmatrix}}_{\underline{\mu}_1}, \underbrace{\begin{bmatrix} 0.5 & 0 \\ 0 & 0.6 \end{bmatrix}}_{\underline{\underline{\Sigma}}_1}\right)$$

$$p(y=2) = 0.35 \quad (\pi_2)$$

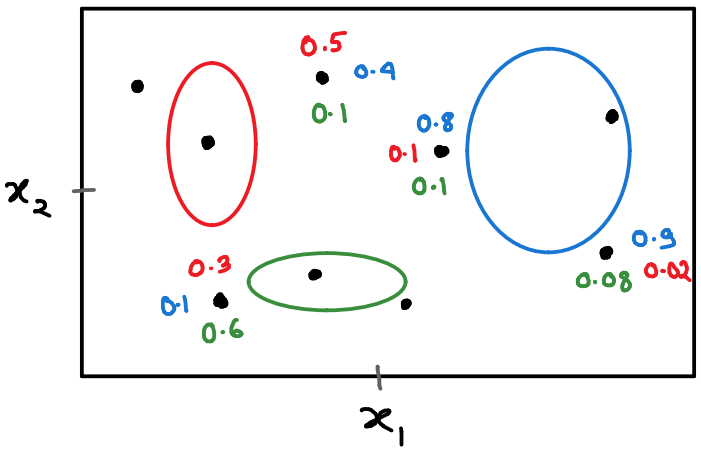
$$p(\underline{x}_i \mid y=2) = \mathcal{N}\left(\begin{bmatrix} -0.1 \\ 0.5 \end{bmatrix}; \underbrace{\begin{bmatrix} -0.5 \\ 0.2 \end{bmatrix}}_{\underline{\mu}_2}, \underbrace{\begin{bmatrix} 0.3 & 0 \\ 0 & 0.4 \end{bmatrix}}_{\underline{\underline{\Sigma}}_2}\right)$$

$$p(y=3) = 0.25 \quad (\pi_3)$$

$$p(\underline{x}_i \mid y=3) = \mathcal{N}\left(\begin{bmatrix} -0.1 \\ 0.5 \end{bmatrix}; \underbrace{\begin{bmatrix} -0.1 \\ -0.3 \end{bmatrix}}_{\underline{\mu}_3}, \underbrace{\begin{bmatrix} 0.7 & 0 \\ 0 & 0.2 \end{bmatrix}}_{\underline{\underline{\Sigma}}_3}\right)$$

$$w_1^{(i)} = p(y=1 \mid \underline{x}_i) = \frac{p(\underline{x}_i \mid y=1)\, p(y=1)}{\sum_{m=1}^{3} p(\underline{x}_i \mid y=m)\, p(y=m)} = 0.4$$

- E-step

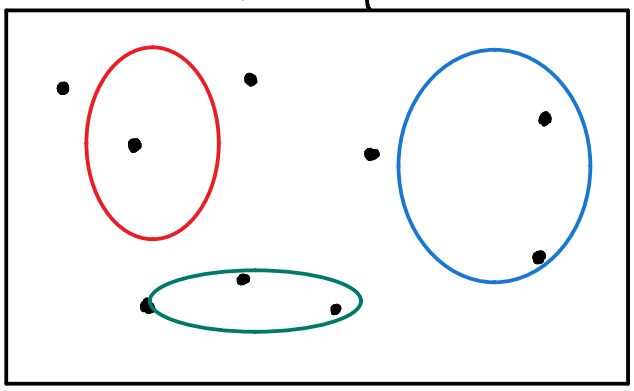



- Assign the responsibility $w_m^{(i)}$ of component $m$ for each data point $i$ using probability

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

---

- M-step



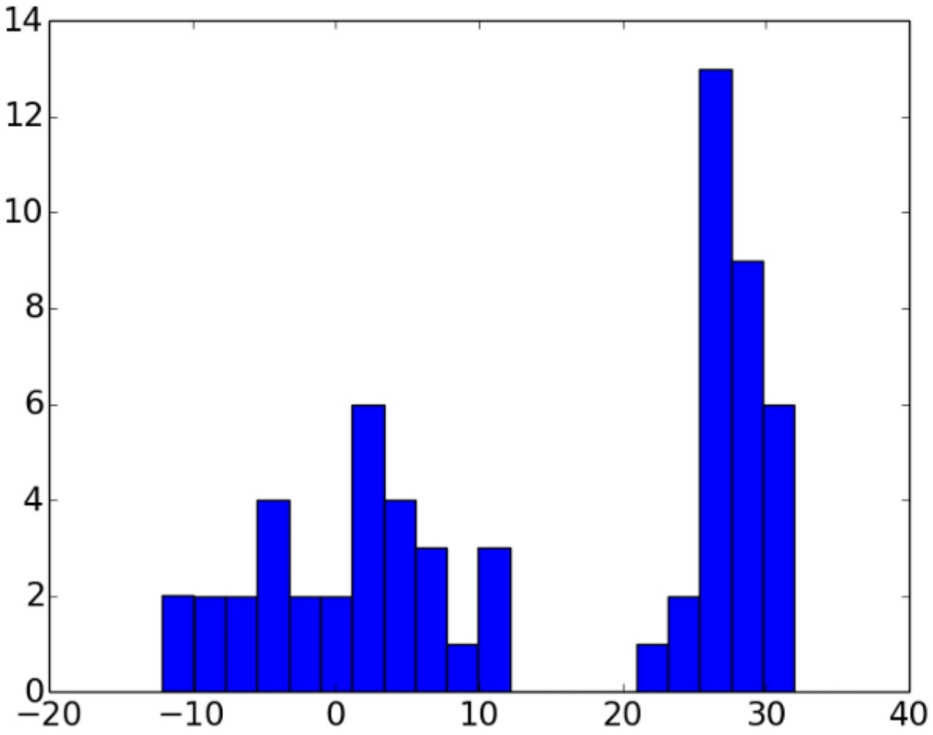

- Apply maximum likelihood updates, where the parameters of each Gaussian component is fit with a weighted dataset

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

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

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

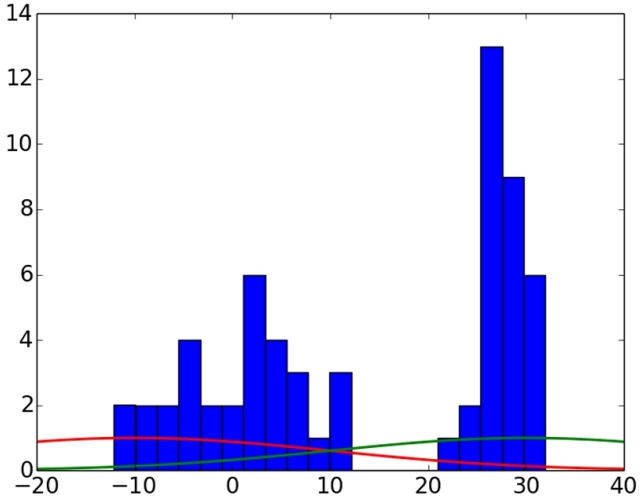
Example : Suppose you have recorded a bunch of temperatures in March for Toronto and Delhi , but forgot which was which, and they are all jumbled up together




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

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

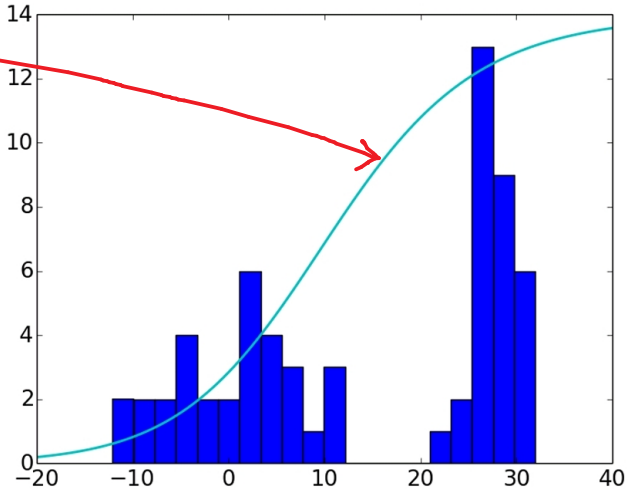
$$\underline{\theta} = \begin{bmatrix} \pi_1 \\ \pi_2 \\ \mu_1 \\ \mu_2 \\ \sigma_1 \\ \sigma_2 \end{bmatrix}$$



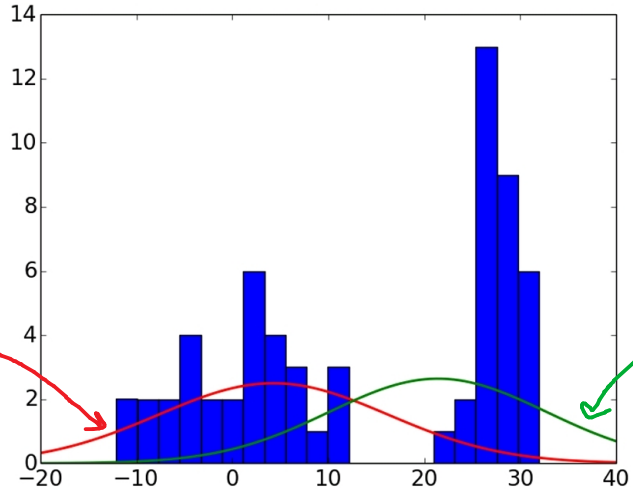

- Iteration 1: E-step and M-step

E-step

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




M-step

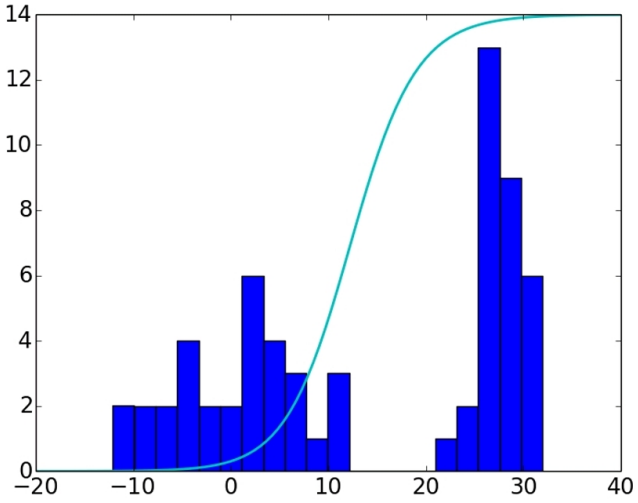



Gaussian component 1

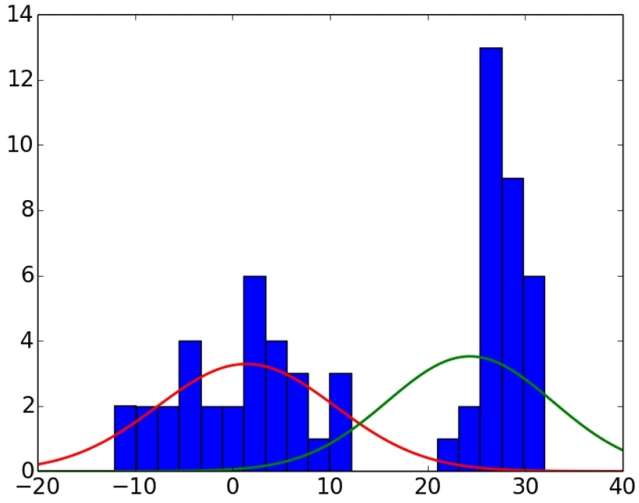
Gaussian component 2

- Iteration 2

E-step



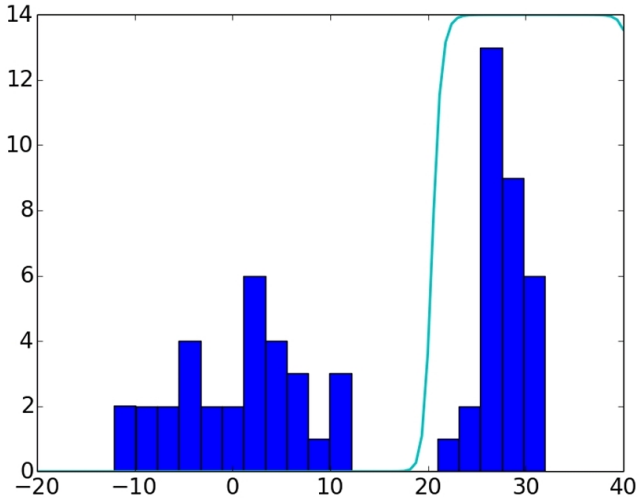

M-step



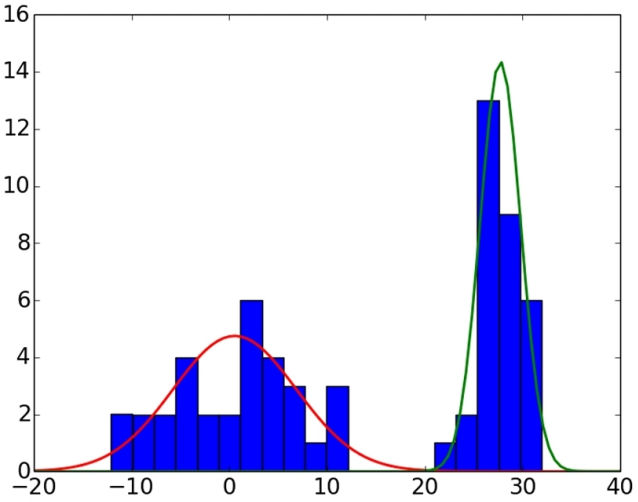

⋮ ⋮

- Iteration 10

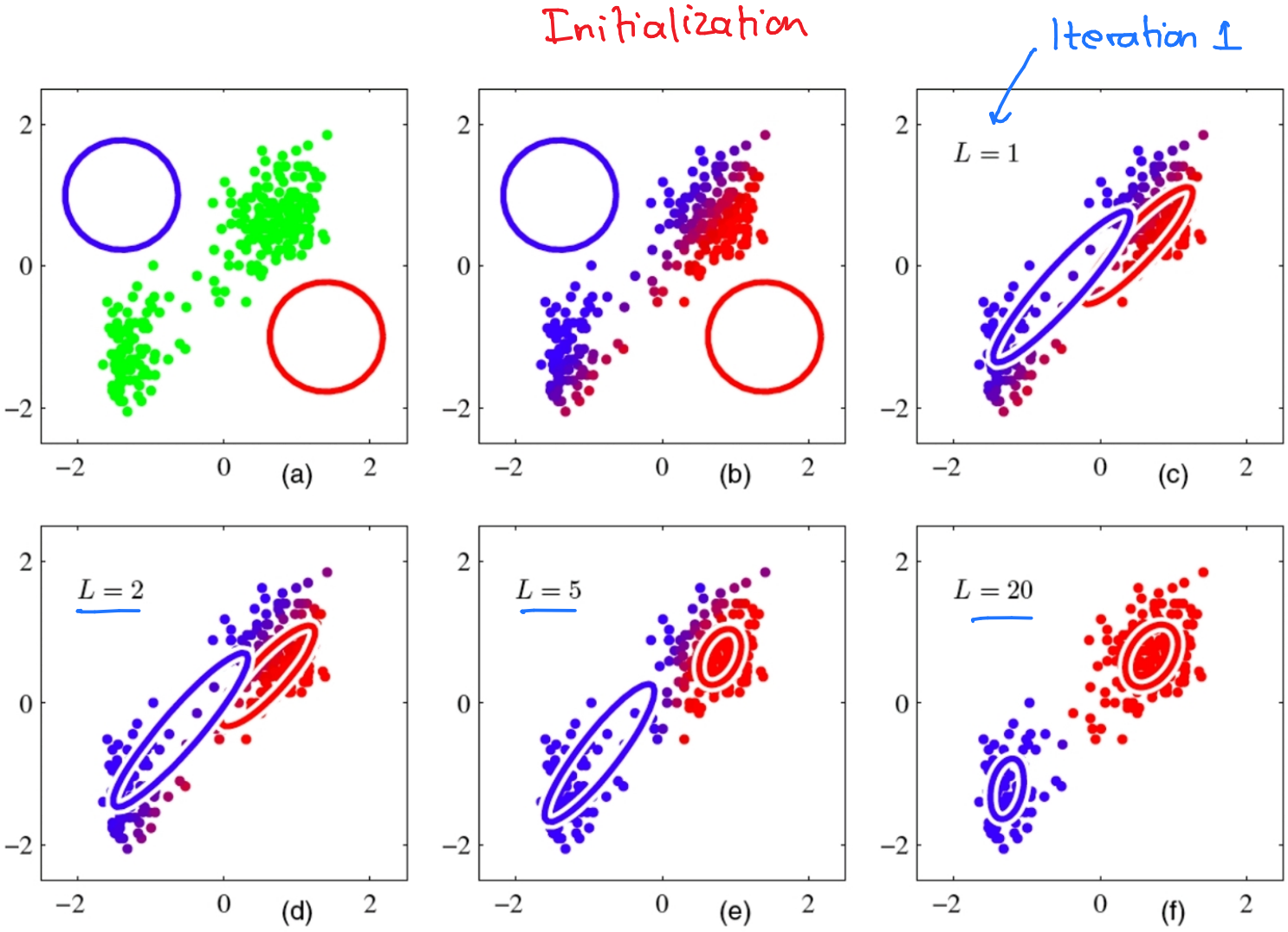
E-step




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