Introduction to Data Mining

CS 145

Lecture 8:

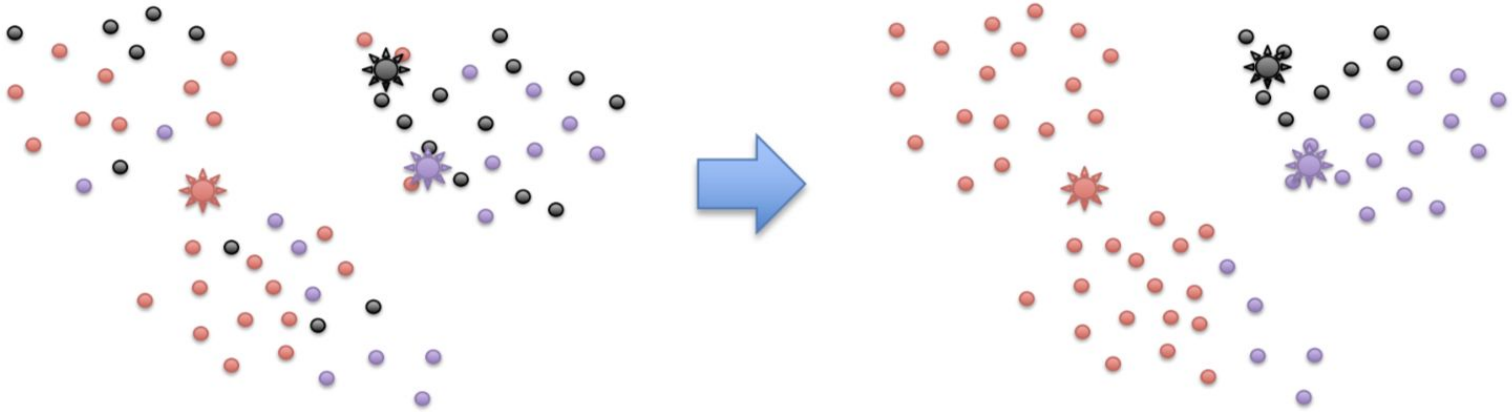
Unsupervised Learning: Gaussian Mixture Model & EM Algorithm

Recall K-Means

- Objective function
  - $J = \sum_{j=1}^{k} \sum_{C(i)=j} ||x_i - c_j||^2$
- Re-arrange the objective function
  - $J = \sum_{j=1}^{k} \sum_{i} w_{ij} ||x_i - c_j||^2$
    - $w_{ij} \in \{0,1\}$
    - $w_{ij} = 1, if\ x_i\ belongs\ to\ cluster\ j; w_{ij} = 0, otherwise$
  - Looking for:
    - The best assignment w_{ij}
    - The best center c_j

Step 1: Update Assignment

- For each x: $w_{ij} = 1, if\ ||x_i - c_j||^2$ is the smallest
  - Assign to cluster C_k with smallest distance to c_k



Step 1: Update Assignment

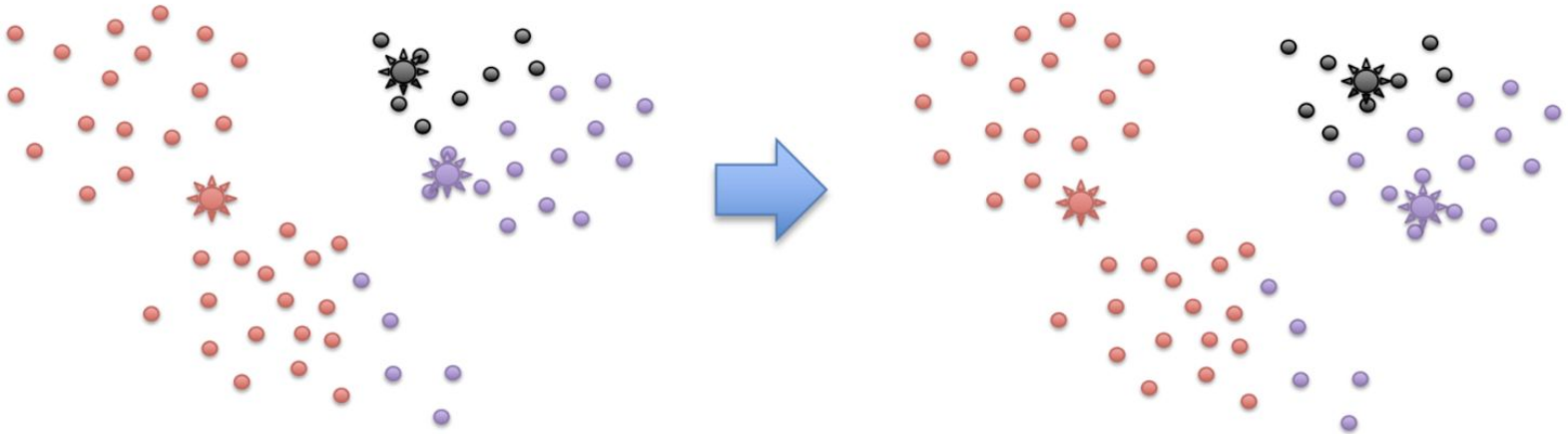
- For each x: $w_{ij} = 1, if\ ||x_i - c_j||^2$ is the smallest

- Such K-Nearest Neighbor Assignment **minimize** the cost function given cluster center (c1, c2, …) **fixed**

$$J = \sum_{j=1}^{k} \sum_{i} w_{ij} ||x_i - c_j||^2$$

Step 2: Update Cluster Center

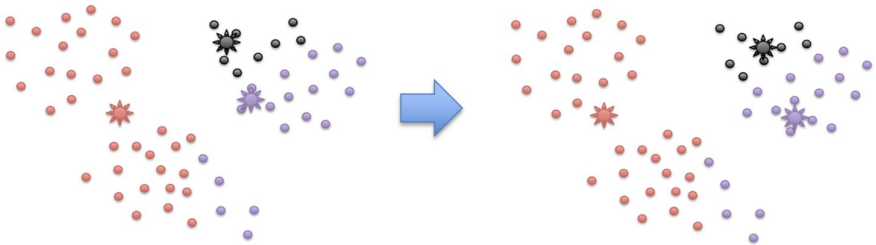
- For each c_k:
  - Compute $c_k = \text{mean}(C_k)$



Step 2: Fix assignment w_{ij}, find centers that minimize J

$$J = \sum_{j=1}^{k} \sum_{i} w_{ij} ||x_i - c_j||^2$$

- => first derivative of J = 0
- => $\frac{\partial J}{\partial c_j} = -2 \sum_i w_{ij}(x_i - c_j) = 0$
- =>$c_j = \frac{\sum_i w_{ij} x_i}{\sum_i w_{ij}}$
  - Note $\sum_i w_{ij}$ is the total number of objects in cluster j



K-Means Algorithm

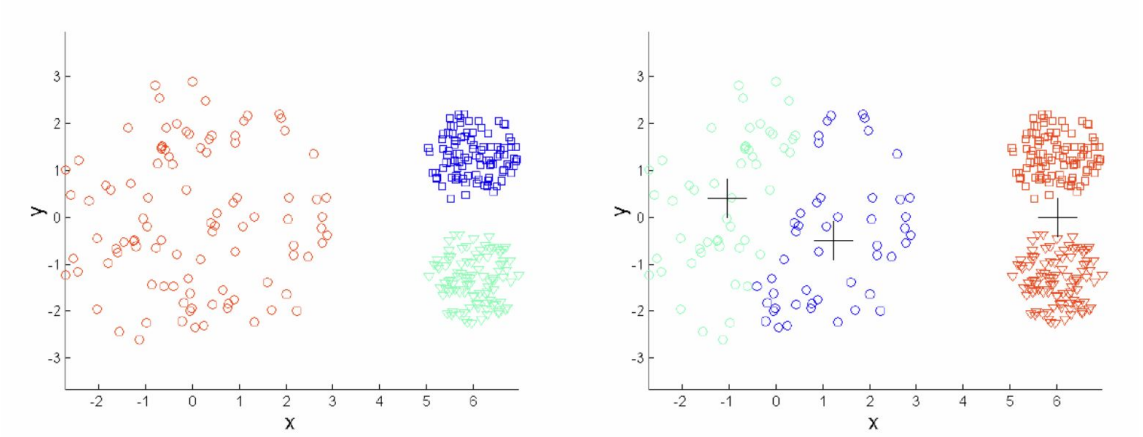
- Iterations

$$J = \sum_{j=1}^{k} \sum_{i} w_{ij} ||x_i - c_j||^2$$

  - Step 1: Fix centers c_j, find assignment w_{ij} that minimizes J
    - => $w_{ij} = 1, if\ ||x_i - c_j||^2$ is the smallest
  - Step 2: Fix assignment w_{ij}, find centers that minimize J
    - => first derivative of J = 0
    - => $\frac{\partial J}{\partial c_j} = -2 \sum_i w_{ij}(x_i - c_j) = 0$
    - => $c_j = \frac{\sum_i w_{ij} x_i}{\sum_i w_{ij}}$
      - Note $\sum_i w_{ij}$ is the total number of objects in cluster j

Limitation of K-Means

- Size: number of data points
- Variance: how scattered a cluster is

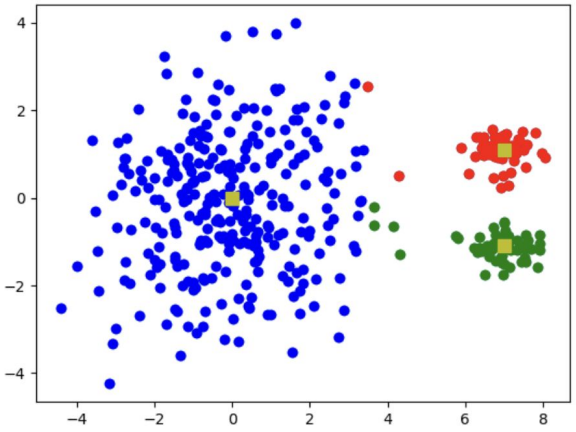


Original Points

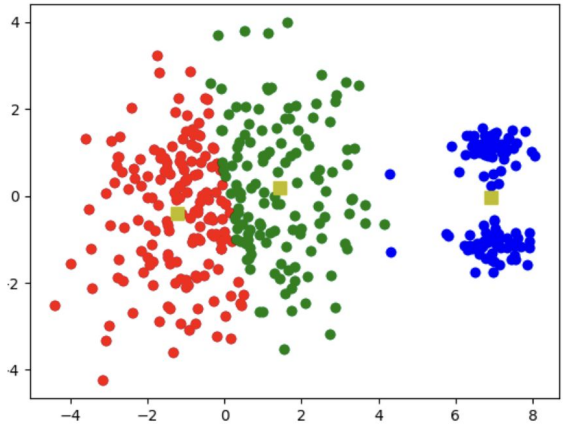
K-means (3 Clusters)

Limitation of K-Means

- Consider the cost of K-means in two cases



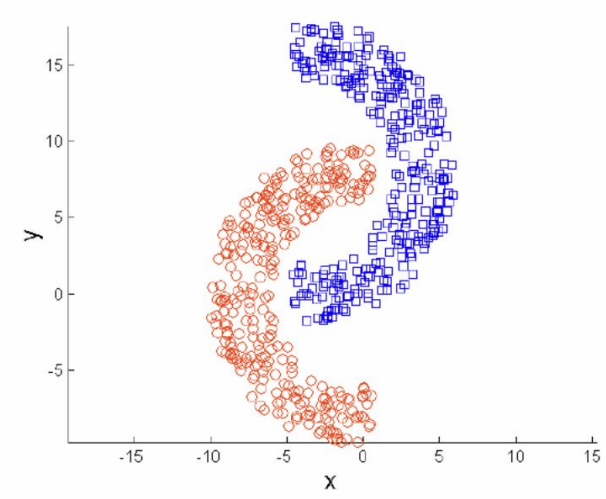
Cost: J = 1560.86



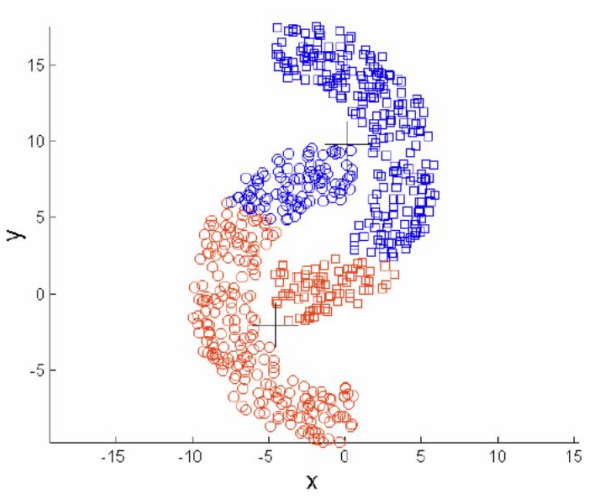
Cost: J = 1147.42

Recall: $J = \sum_{j=1}^{k} \sum_{C(i)=j} ||x_i - c_j||^2$

Non-Spherical Shapes (L2 fails)



Original Points



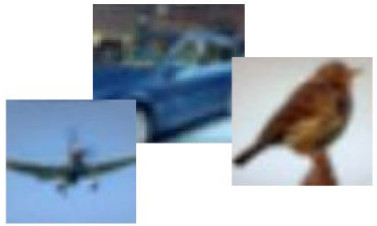
K-means (2 Clusters)

A generative View of Clustering

- This lecture: probabilistic formulation of clustering
- We need a sensible measure of what it means to cluster the data well
  - This makes it possible to judge different methods
  - It may help us decide on the number of clusters
- An obvious approach is to imagine that the data was produced by a generative model
  - Then we adjust the model parameters using maximum likelihood i.e. to maximize the probability that it would produce exactly the data we observed

Generative Models

Given training data, generate new samples from same distribution



Training data ~ $p_{data}(x)$

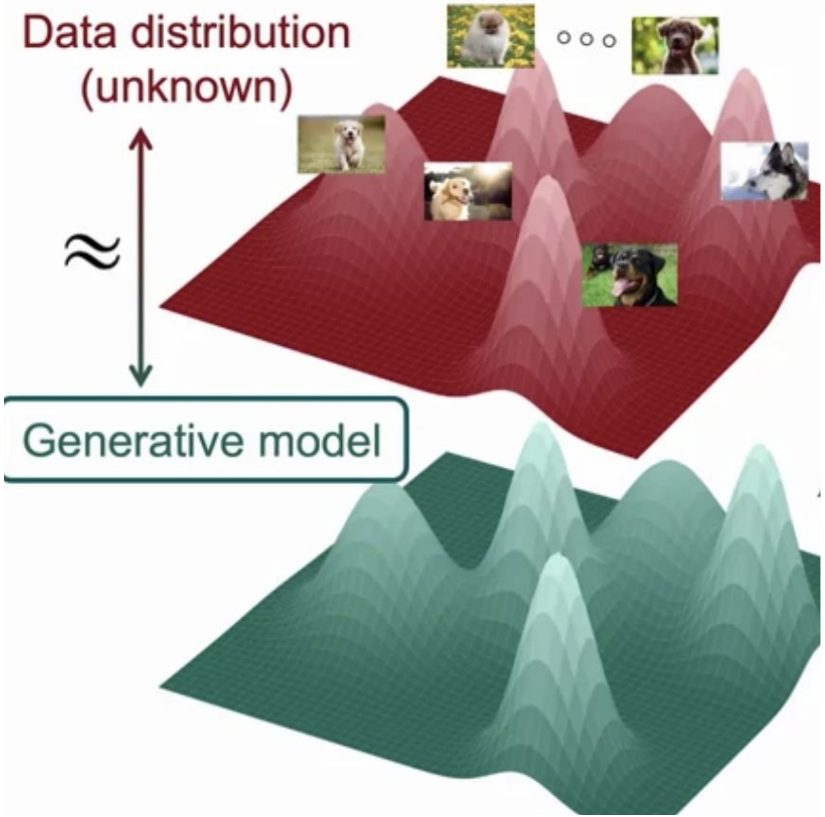
Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

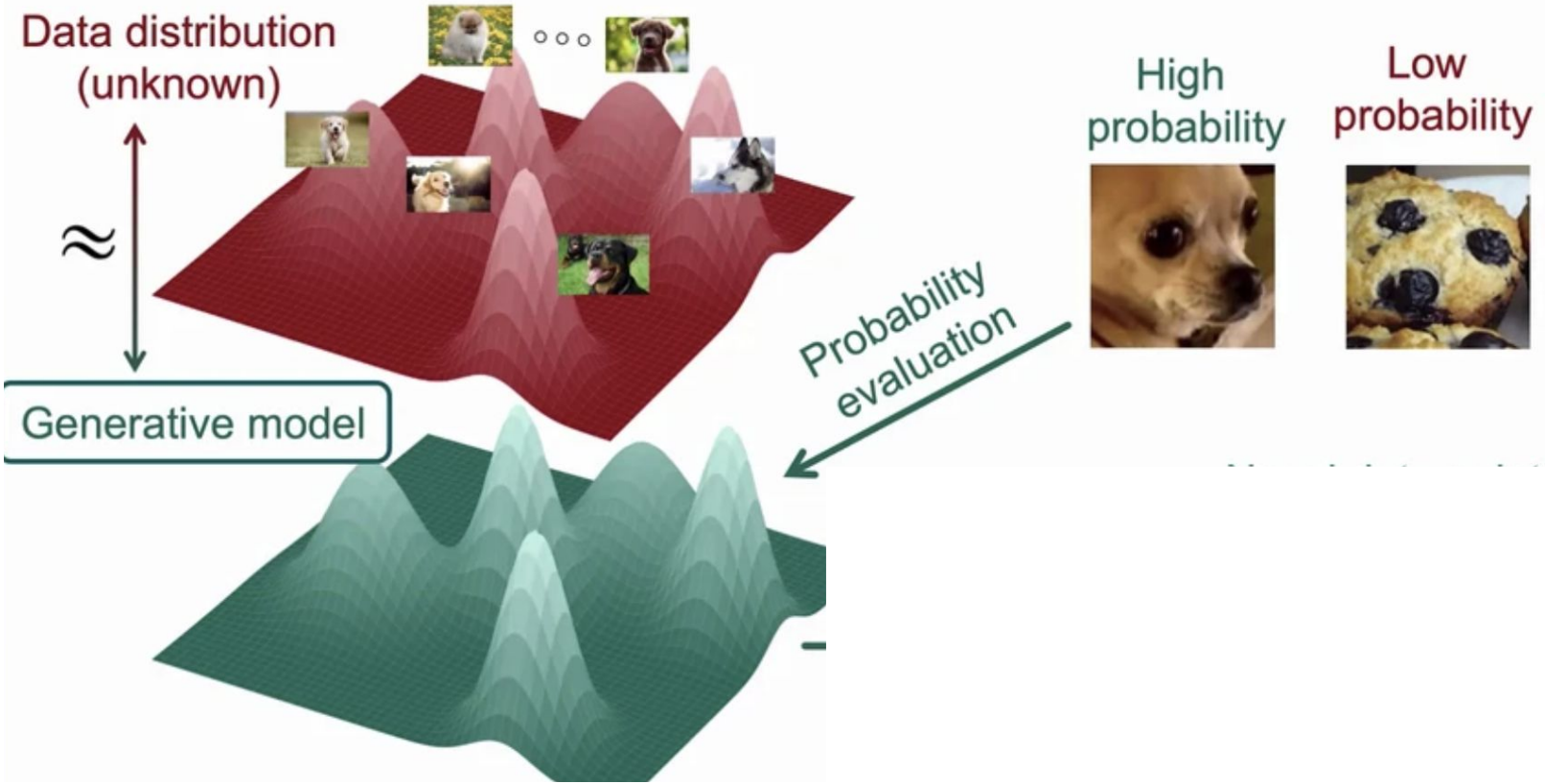
Addresses density estimation, a core problem in unsupervised learning

2,121 × 820

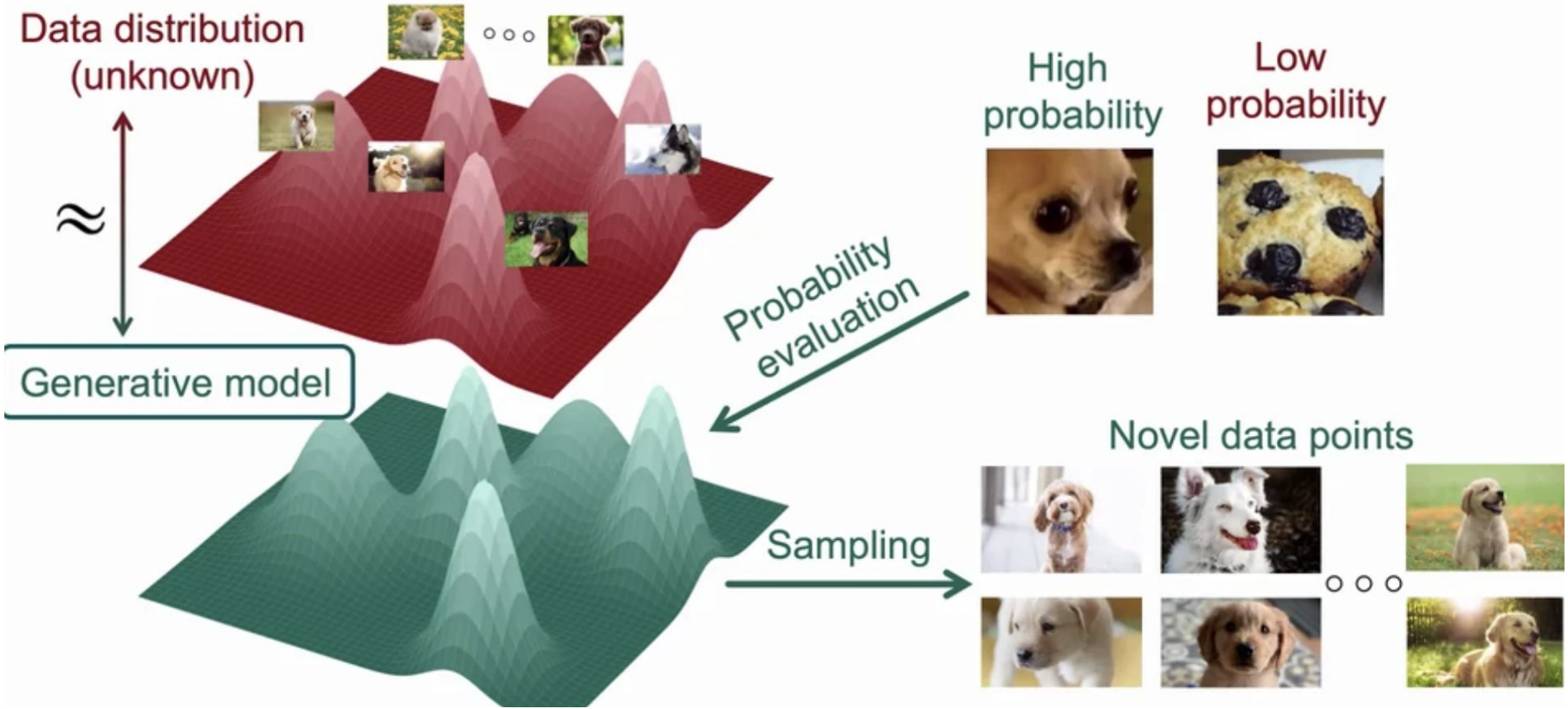
Estimating the probability distribution of data



Estimating the probability distribution of data



Estimating the probability distribution of data



Generative Probabilistic Models

- Models joint distribution of x and y: $P(x, y)$
- Can make predictions via Bayes Rule:

$$P(y \mid x) = \frac{P(x, y)}{P(x)} = \frac{P(x \mid y)P(y)}{P(x)}$$

Prediction = choose y with maximal P(y|x)

- Can infer marginal distributions:

$$P(y) = \sum_x P(y, x) \qquad P(x) = \sum_y P(y, x)$$

Example

- P(x,y) sums to 1
  - Joint distribution

- P(x=Homework) ??
  - **Answer: 0.5**
  - "Marginalize out the y"

y	x	P(x,y)
Y= SPAM	Help!	0.15
y= NOT	Help!	0.1
y= SPAM	Homework	0.05
y= NOT	Homework	0.45
Y= SPAM	Winner!	0.2
Y= NOT	Winner!	0.05

Margin distribution of P(x)

$$P(x) = \sum_{y} P(y,x)$$

Example #2

- P(x,y) sums to 1
  - Joint distribution

y	x	P(x,y)
Y= SPAM	Help!	0.15
y= NOT	Help!	0.1
y= SPAM	Homework	0.05
y= NOT	Homework	0.45
Y= SPAM	Winner!	0.2
Y= NOT	Winner!	0.05

- P(y=SPAM|x=Help!) ??
  - **Answer: 0.6**
  - P(x,y) = 0.15
  - P(x) = 0.25

$$P(y \mid x) = \frac{P(x,y)}{P(x)} \qquad P(x) = \sum_{y} P(y,x)$$

Example #3

- P(x,y) sums to 1
  - Joint distribution

- P(x=Help!|y=NOT) ??
  - **Answer: 0.17**
  - P(x,y) = 0.1
  - P(y) = 0.6

y	x	P(x,y)
Y= SPAM	Help!	0.15
y= NOT	Help!	0.1
y= SPAM	Homework	0.05
y= NOT	Homework	0.45
Y= SPAM	Winner!	0.2
Y= NOT	Winner!	0.05

$$P(x \mid y) = \frac{P(x,y)}{P(y)} \qquad P(y) = \sum_{x} P(y,x)$$

Training

- Goal is to learn P(x,y)
  - What is objective function?

y	x	P(x,y)
Y= SPAM	Help!	0.15
y= NOT	Help!	0.1
y= SPAM	Homework	0.05
y= NOT	Homework	0.45
Y= SPAM	Winner!	0.2
Y= NOT	Winner!	0.05

- **Maximum Likelihood!**

$$\text{argmax}\, P(S) = \text{argmax} \prod_i P(x_i, y_i)$$

$$= \text{argmin} \sum_i -\log P(x_i, y_i)$$

Interpretation: Given model structure, find that parameterization that best explains data

N
$i=1$

How to Learn p(x)? Maximum Likelihood

- Motivating example: estimating the parameter of a biased coin
  - You flip a coin 100 times. It lands heads $N_H = 55$ times and tails $N_T = 45$ times.
  - What is the probability it will come up heads if we flip again?
- Model: flips are independent Bernoulli random variables with parameter θ.
  - Assume the observations are **independent and identically distributed (i.i.d.)**

How to Learn p(x)? Maximum Likelihood

- The **likelihood function** is the probability of the observed data, as a function of θ.
- In our case, it's the probability of a *particular* sequence of H's and T's.
- Under the Bernoulli model with i.i.d. observations,

$$L(\theta) = p(\mathcal{D}) = \theta^{N_H}(1-\theta)^{N_T}$$

How to Learn p(x)? Maximum Likelihood

- The **likelihood function** is the probability of the observed data, as a function of θ.
- In our case, it's the probability of a *particular* sequence of H's and T's.
- Under the Bernoulli model with i.i.d. observations,

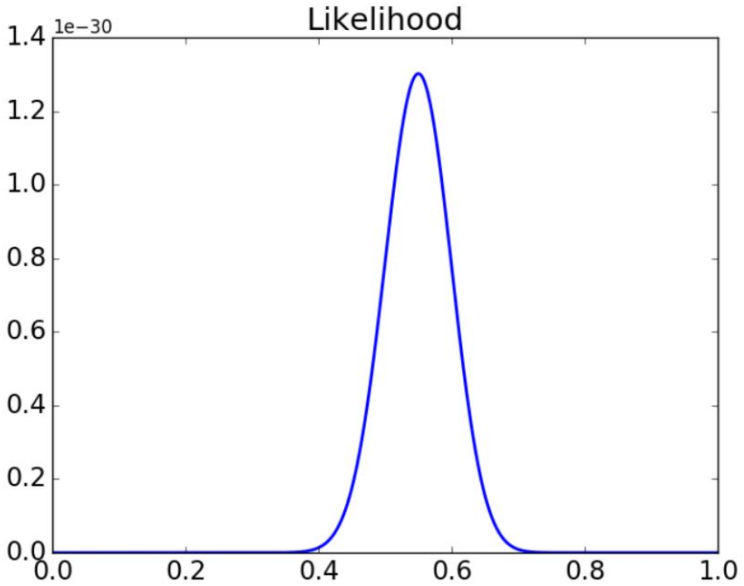
$$L(\theta) = p(\mathcal{D}) = \theta^{N_H}(1-\theta)^{N_T}$$

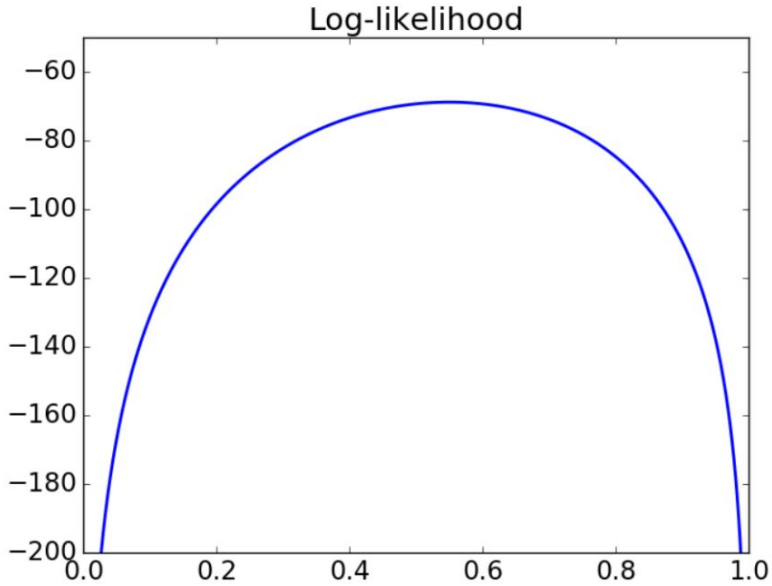
- This takes very small values. In this case, $L(0.5) = 0.5^{100} \approx 7.9 \times 10^{-31}$
- Therefore, we usually work with log-likelihoods:

$$\ell(\theta) = \log L(\theta) = N_H \log \theta + N_T \log(1-\theta)$$

- Here, $\ell(0.5) = \log 0.5^{100} = 100 \log 0.5 = -69.31$

$N_H = 55,\ N_T = 45$





$$L(\theta) = p(\mathcal{D}) = \theta^{N_H}(1-\theta)^{N_T}$$

$$\log L(\theta) = N_H \log \theta + N_T \log(1-\theta)$$

Maximum Likelihood Estimation

- Good values of θ should assign high probability to the observed data. This motivates the **maximum likelihood criterion**.
- Remember how we found the optimal solution to linear regression by setting derivatives to zero? We can do that again for the coin example.

$$\begin{aligned}\frac{\mathrm{d}\ell}{\mathrm{d}\theta} &= \frac{\mathrm{d}}{\mathrm{d}\theta}\left(N_H \log\theta + N_T \log(1-\theta)\right) \\ &= \frac{N_H}{\theta} - \frac{N_T}{1-\theta}\end{aligned}$$

This is binary cross entropy

- Setting this to zero gives the maximum likelihood estimate:

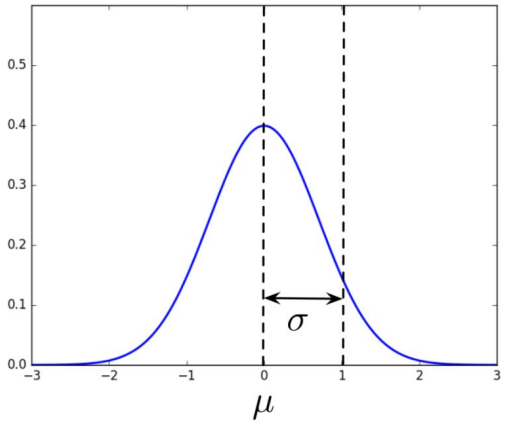
$$\hat{\theta}_{\mathrm{ML}} = \frac{N_H}{N_H + N_T},$$

Likelihood for Gaussian

- Recall the **Gaussian**, or **normal**, distribution:

$$\mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

- The Central Limit Theorem says that sums of lots of independent random variables are approximately Gaussian.
- In machine learning, we use Gaussians a lot because they make the calculations easy.



Likelihood for Gaussian

$$\mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

-2.5 -9.9 -12.1 -8.9 -6.0 -4.8 2.4

- Assume they're drawn from a Gaussian distribution with known standard deviation $\sigma = 5$, and we want to find the mean μ.
- Log-likelihood function:

$$\ell(\mu) = \log \prod_{i=1}^{N} \left[\frac{1}{\sqrt{2\pi} \cdot \sigma} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right)\right]$$

$$= \sum_{i=1}^{N} \log \left[\frac{1}{\sqrt{2\pi} \cdot \sigma} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right)\right]$$

Likelihood for Gaussian

$$\mathcal{N}(x; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

-2.5 -9.9 -12.1 -8.9 -6.0 -4.8 2.4

- Assume they're drawn from a Gaussian distribution with known standard deviation $\sigma = 5$, and we want to find the mean μ.
- Log-likelihood function:

$$\begin{aligned}
\ell(\mu) &= \log \prod_{i=1}^{N} \left[\frac{1}{\sqrt{2\pi} \cdot \sigma} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right) \right] \\
&= \sum_{i=1}^{N} \log \left[\frac{1}{\sqrt{2\pi} \cdot \sigma} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right) \right] \\
&= \sum_{i=1}^{N} \underbrace{-\frac{1}{2} \log 2\pi - \log \sigma}_{\text{constant}} - \frac{(x^{(i)} - \mu)^2}{2\sigma^2}
\end{aligned}$$

Maximize Likelihood for Mean

$$\ell(\mu) = \sum_{i=1}^{N} \underbrace{-\frac{1}{2}\log 2\pi - \log\sigma}_{\text{constant}} - \frac{(x^{(i)} - \mu)^2}{2\sigma^2}$$

- Maximize the log-likelihood by setting the derivative to zero:

$$0 = \frac{\mathrm{d}\ell}{\mathrm{d}\mu} = -\frac{1}{2\sigma^2}\sum_{i=1}^{N}\frac{\mathrm{d}}{\mathrm{d}\mu}(x^{(i)} - \mu)^2$$

$$= \frac{1}{\sigma^2}\sum_{i=1}^{N} x^{(i)} - \mu$$

- Solving we get $\mu = \frac{1}{N}\sum_{i=1}^{N} x^{(i)}$
- This is just the mean of the observed values, or the **empirical mean**.

Maximize Likelihood for Variance

$$0 = \frac{\partial \ell}{\partial \mu} = -\frac{1}{\sigma^2} \sum_{i=1}^{N} x^{(i)} - \mu$$

$$0 = \frac{\partial \ell}{\partial \sigma} = \frac{\partial}{\partial \sigma} \left[\sum_{i=1}^{N} -\frac{1}{2} \log 2\pi - \log \sigma - \frac{1}{2\sigma^2} (x^{(i)} - \mu)^2 \right]$$

$$= \sum_{i=1}^{N} -\frac{1}{2} \frac{\partial}{\partial \sigma} \log 2\pi - \frac{\partial}{\partial \sigma} \log \sigma - \frac{\partial}{\partial \sigma} \frac{1}{2\sigma} (x^{(i)} - \mu)^2$$

$$= \sum_{i=1}^{N} 0 - \frac{1}{\sigma} + \frac{1}{\sigma^3} (x^{(i)} - \mu)^2$$

$$= -\frac{N}{\sigma} + \frac{1}{\sigma^3} \sum_{i=1}^{N} (x^{(i)} - \mu)^2$$

$$\hat{\mu}_{\mathrm{ML}} = \frac{1}{N} \sum_{i=1}^{N} x^{(i)}$$

$$\hat{\sigma}_{\mathrm{ML}} = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (x^{(i)} - \mu)^2}$$

Recall MLE

- So far, maximum likelihood has told us to use empirical counts or statistics:
  - **Bernoulli:** $\theta = \frac{N_H}{N_H + N_T}$
  - **Gaussian:** $\mu = \frac{1}{N} \sum x^{(i)}$, $\sigma^2 = \frac{1}{N} \sum (x^{(i)} - \mu)^2$

Recall MLE used before

We've been doing maximum likelihood estimation all along!

- Squared error loss (e.g. linear regression)
  - **Gaussian:** $\mu = \frac{1}{N}\sum x^{(i)}$, $\sigma^2 = \frac{1}{N}\sum (x^{(i)} - \mu)^2$

$$p(t|y) = \mathcal{N}(t; y, \sigma^2)$$

$$-\log p(t|y) = \frac{1}{2\sigma^2}(y - t)^2 + \text{const}$$

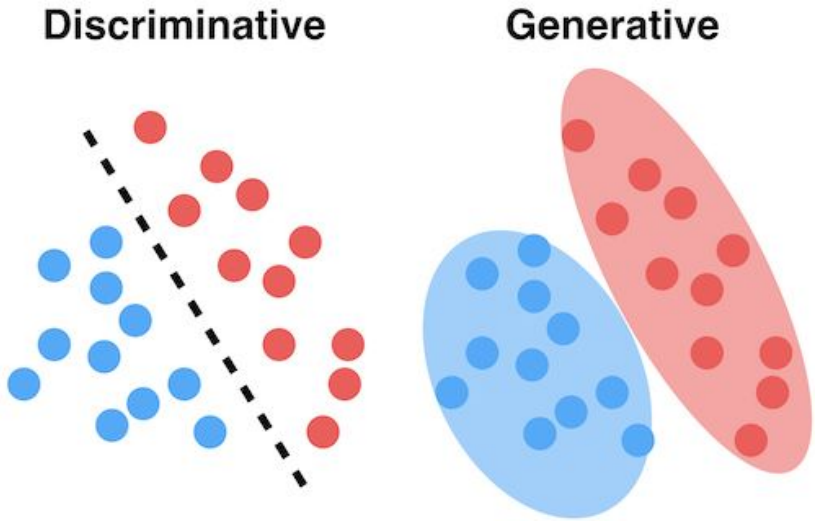
- Cross-entropy loss (e.g. logistic regression)
  - **Bernoulli:** $\theta = \frac{N_H}{N_H + N_T}$

$$p(t = 1|y) = \text{Bernoulli}(t; y)$$

$$-\log p(t|y) = -t \log y - (1 - t) \log(1 - y)$$

Generative vs Discriminative

- Generative models
  - Models both y AND x
  - P(x,y)
- Discriminative models
  - Models y GIVEN x
  - P(y|x)
  - E.g., Logistic Regression



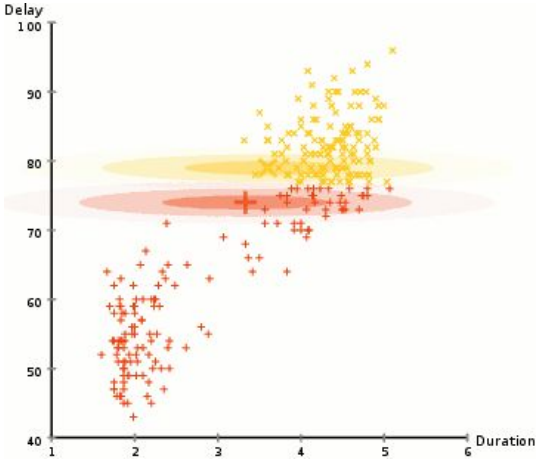
Generative vs Discriminative

- Generative:
  - Models all of P(x,y)
  - Prediction via Bayes Rule
    - Tolerates missing data
- Discriminative:
  - Only models P(y|x)
  - Directly models prediction task
    - Cannot naturally tolerate missing data

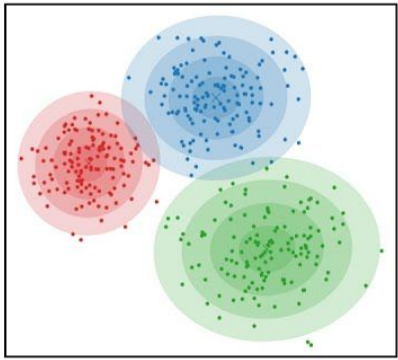
y	x	P(x,y)
Y= SPAM	Help!	0.15
y= NOT	Help!	0.1
y= SPAM	Homework	0.05
y= NOT	Homework	0.45
Y= SPAM	Winner!	0.2
Y= NOT	Winner!	0.05

Gaussian Mixture Model

A Generative Model for Clustering



Generative Clustering Setup



- We'll be working with the following generative model for data $\mathbf{x} \in \mathbb{R}^D$
- Assume a datapoint $\mathbf{x}$ is generated as follows:
  - Choose a cluster z from $\{1, \ldots, K\}$ such that $p(z = k) = \pi_k$
  - Given z, sample $\mathbf{x}$ from a Gaussian distribution $\mathcal{N}(\mu_z, I)$
- Can also be written:

$$p(z = k) = \pi_k$$

$$p(\mathbf{x}|z = k) = \mathcal{N}(\mathbf{x}|\mu_k, I)$$

with π_k the **mixing coefficients**, where:

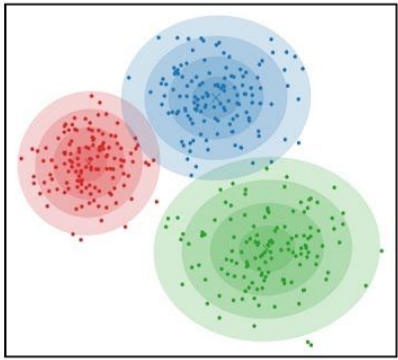
$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{and} \quad \pi_k \geq 0 \quad \forall k$$

Generative Clustering Setup

$$p(z = k, \mathbf{x}) = p(z = k)p(\mathbf{x}|z = k)$$

Prior of cluster z

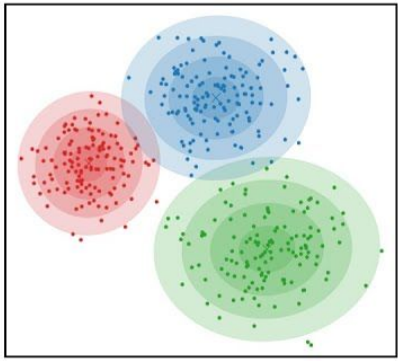
Given cluster z, sample data x



$$p(z = k, \mathbf{x}) = p(z = k)p(\mathbf{x}|z = k)$$

Prior of cluster z

Given cluster z, sample data x



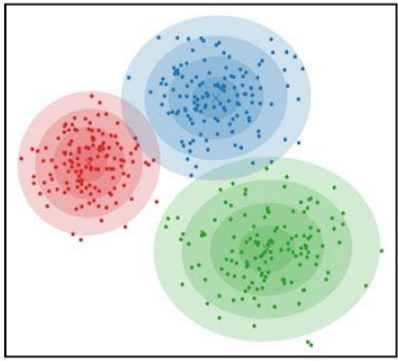
Question:

Q1: What is $p(z = k|\mathbf{x})$?

$$p(z = k, \mathbf{x}) = p(z = k)p(\mathbf{x}|z = k)$$

Prior of cluster z

Given cluster z, sample data x



Question:

Q1: What is $p(z = k|\mathbf{x})$?

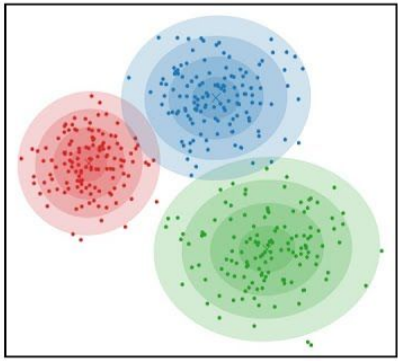
A1: Given a data point x, the probability x belongs to cluster (gaussian) k

Exactly the Wij we use in K-means, except that Wij is 0/1, while p(z|x) is a soft probability value between 0 to 1.

$$p(z = k, \mathbf{x}) = p(z = k)p(\mathbf{x}|z = k)$$

Prior of cluster z

Given cluster z, sample data x



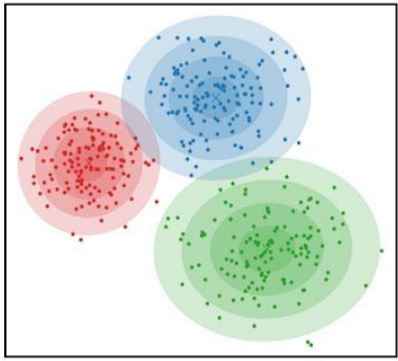
Question:

Q2: What is $p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$?

$$p(z = k, \mathbf{x}) = p(z = k)p(\mathbf{x}|z = k)$$

Prior of cluster z

Given cluster z, sample data x



Question:

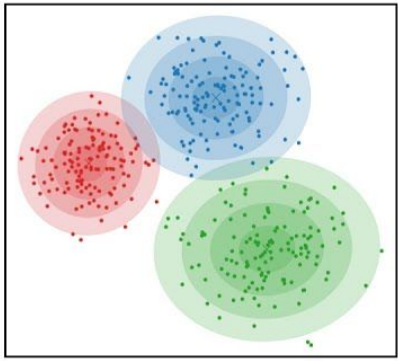
Q2: What is $p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$?

A2: Joint Distribution over variable x (data) and z (model / clusters)

$$p(z = k, \mathbf{x}) = p(z = k)p(\mathbf{x}|z = k)$$

Prior of cluster z

Given cluster z, sample data x



Question:

Q2: What is $p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$?

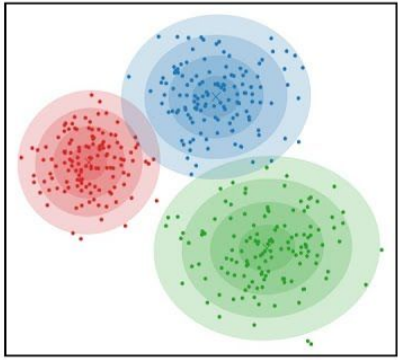
A2: Joint Distribution over variable x (data) and z (model / clusters)

Q3: What is $p(x) = \sum_z p(x, z) = \sum_k p(x, z = k) = \sum_k p(x|z = k) \cdot p(z = k)$

$$p(z = k, \mathbf{x}) = p(z = k)p(\mathbf{x}|z = k)$$

Prior of cluster z

Given cluster z, sample data x



Question:

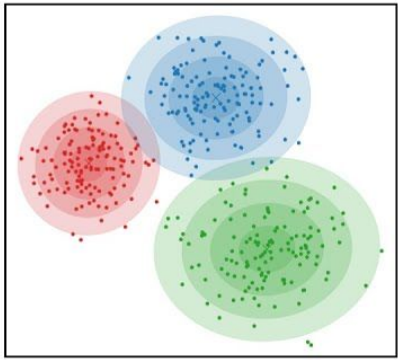
Q2: What is $p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$?

A2: Joint Distribution over variable x (data) and z (model / clusters)

Q3: What is $$p(x) = \sum_z p(x, z) = \sum_k p(x, z = k) = \sum_k p(x|z = k) \cdot p(z = k)$$

A3: Given all the clusters z, the probability we can sample / observe data x from the current system. We would like to maximize the likelihood over the whole dataset.

Generative Clustering Setup



- We model the joint distribution as,

$$p(\mathbf{x}, z) = p(\mathbf{x}|z)p(z)$$

Given cluster z, sample data x

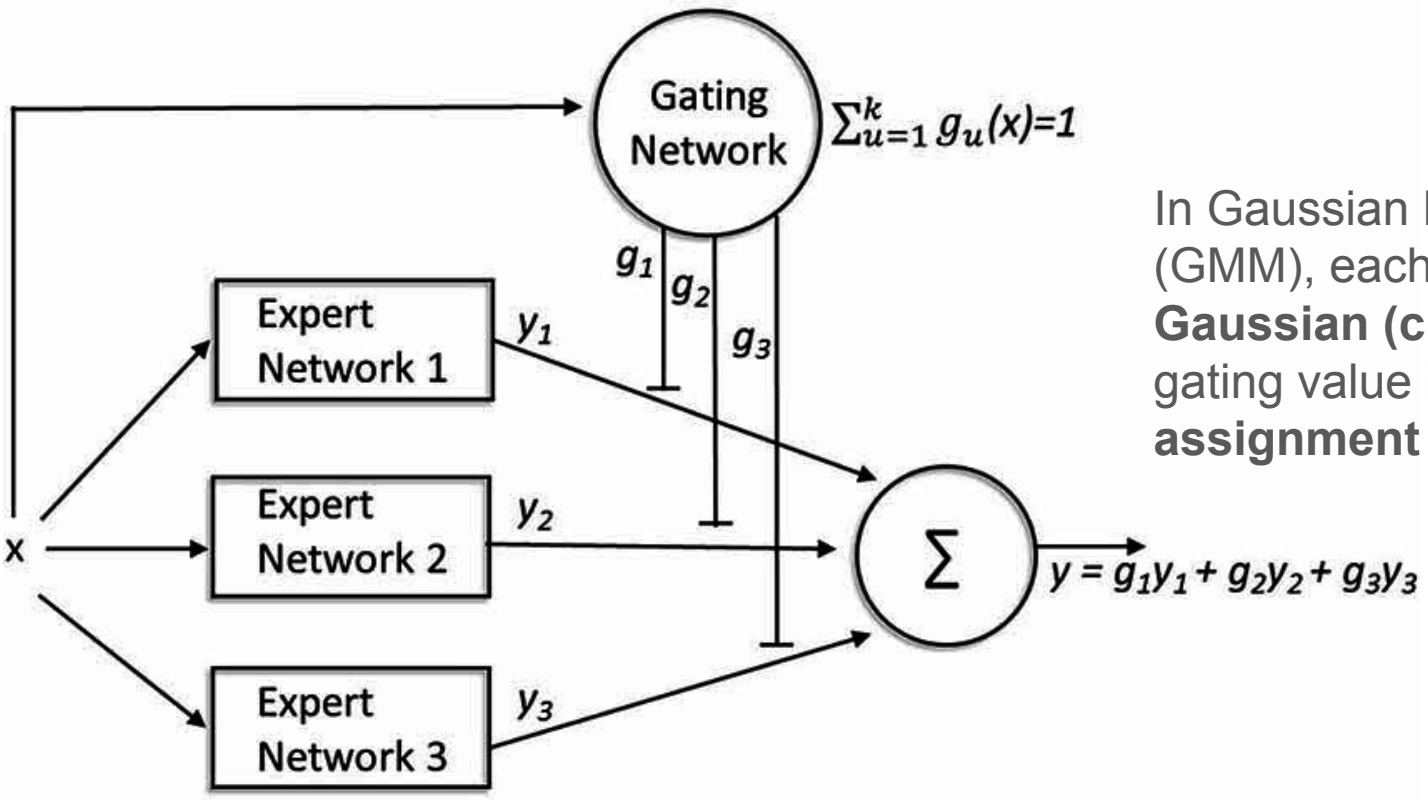
Prior of cluster z

- But in unsupervised clustering we do not have the class labels z.
- What can we do instead?

$$p(\mathbf{x}) = \sum_z p(\mathbf{x}, z) = \sum_z p(\mathbf{x}|z)p(z)$$

- This is a **mixture model**

Recall: Mixture of Expert



In Gaussian Mixture Model (GMM), each Expert is a **Gaussian (cluster)**, and gating value is **soft cluster assignment**

Gaussian Mixture Model (GMM)

Most common mixture model: **Gaussian mixture model** (GMM)

- A GMM represents a **distribution** as

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

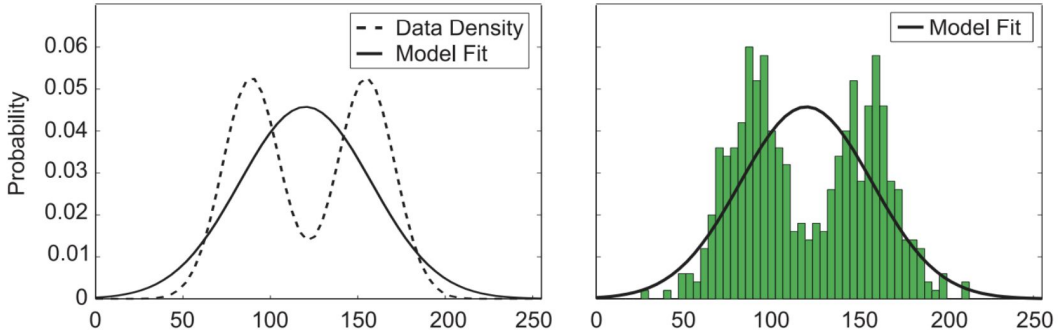
with π_k the **mixing coefficients**, where:

$$\sum_{k=1}^{K} \pi_k = 1 \quad \text{and} \quad \pi_k \geq 0 \quad \forall k$$

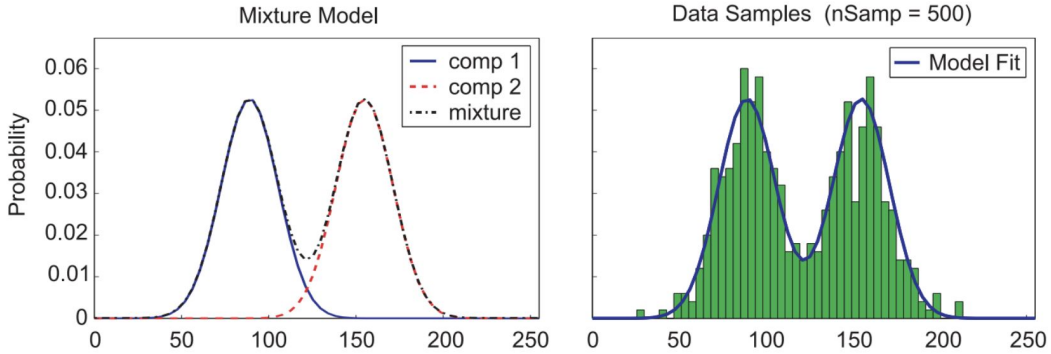
- GMM is a density estimator

Mixture of 1D Gaussians

- If you fit a Gaussian to data:



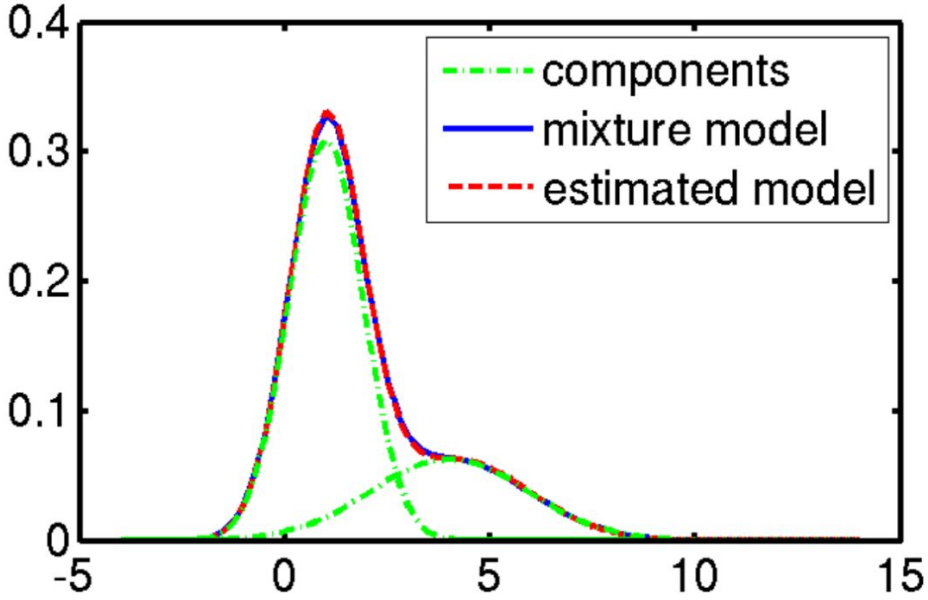
- Now, we are trying to fit a GMM (with $K = 2$ in this example):



[Slide credit: K. Kutulakos]

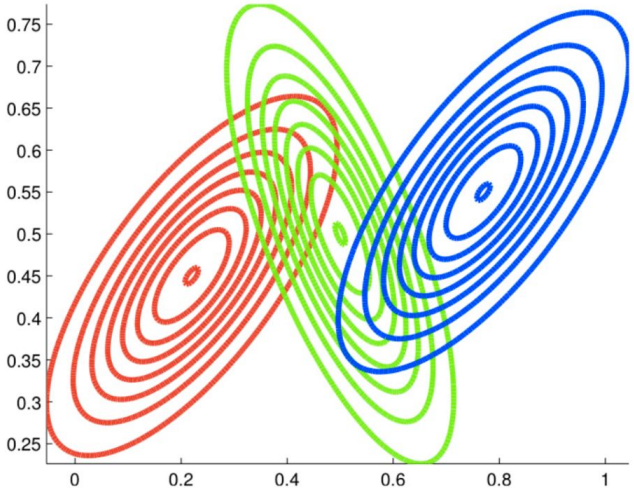
Example: 1-D GMM

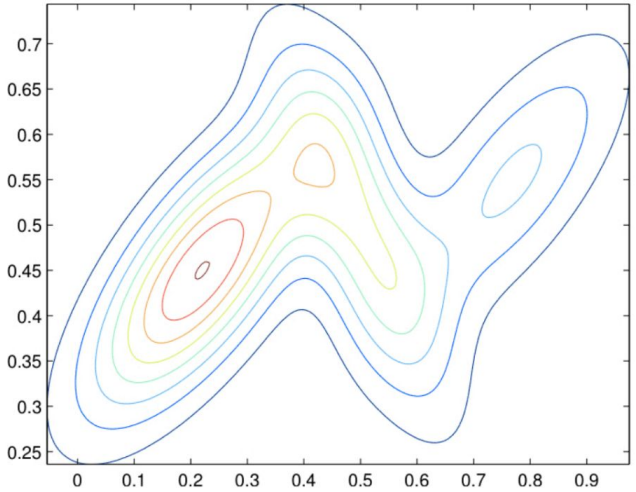
- Blue curve: ground truth distribution
- Sample data points from blue curve
- Red curve: estimated distribution

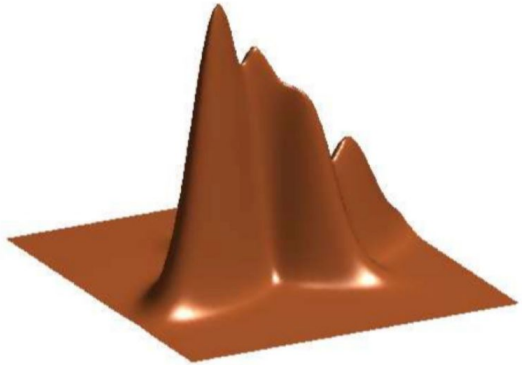


Assuming w_1 and w_2 are the same (0.5); Each component is plotted as $0.5 f_j(x)$

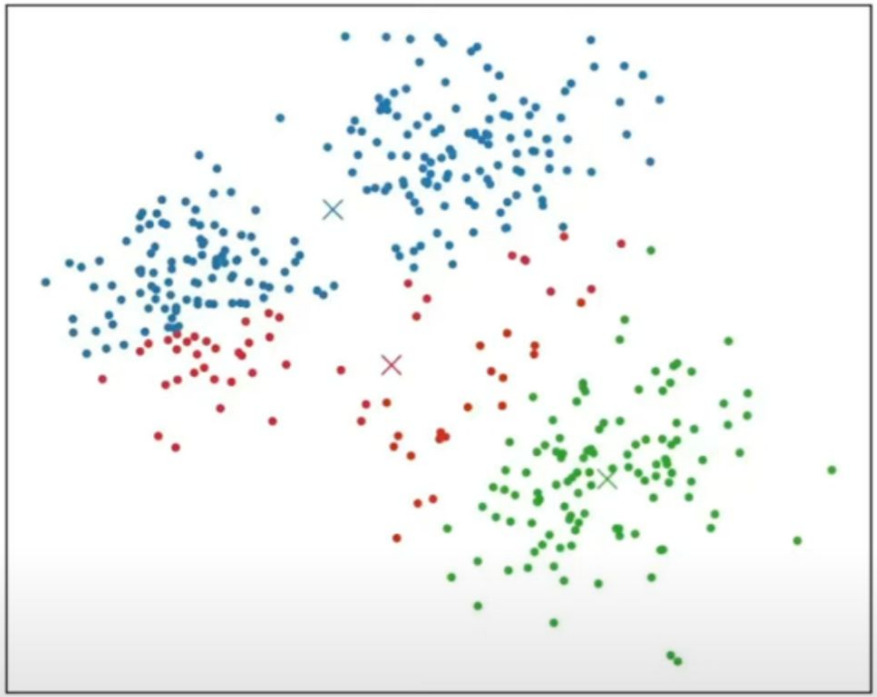
Mixture of 2D Gaussians



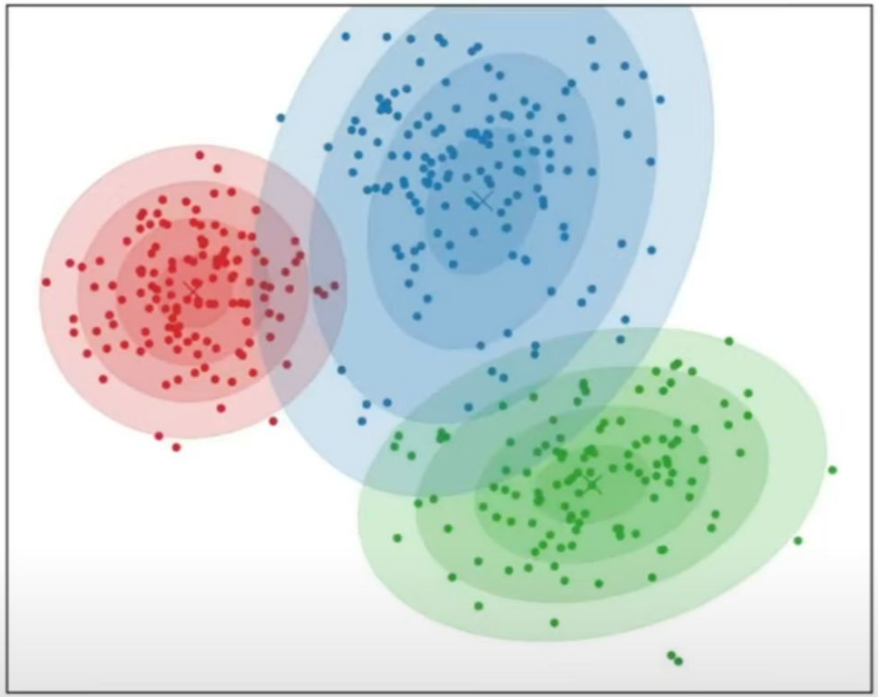




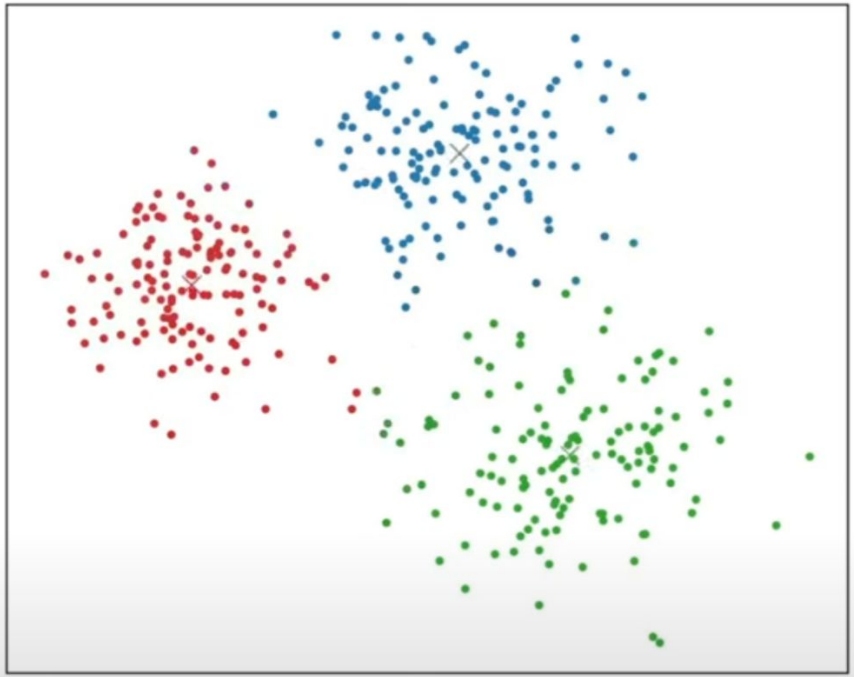
K-Means



GMM

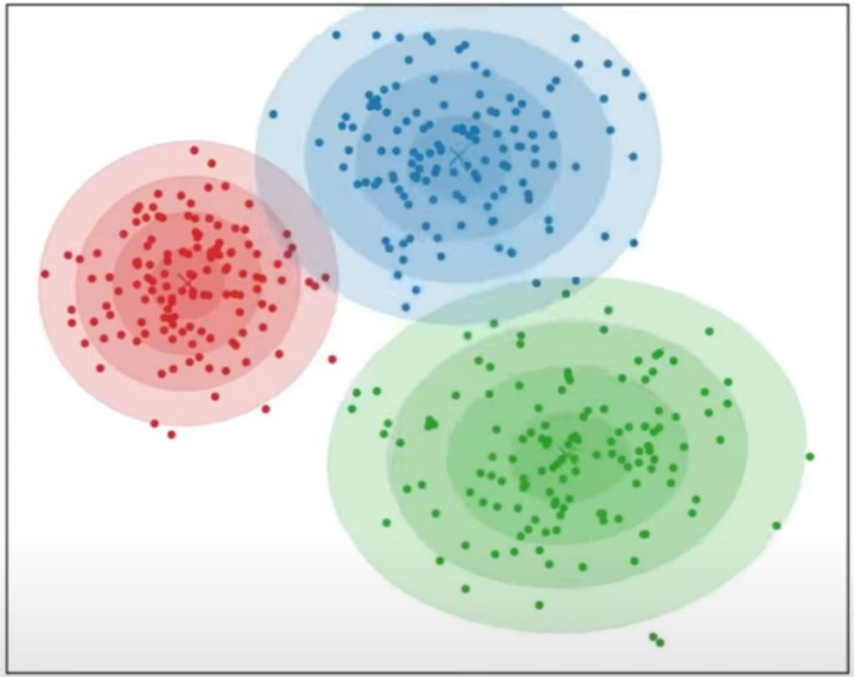


K-Means



red, blue or green

GMM



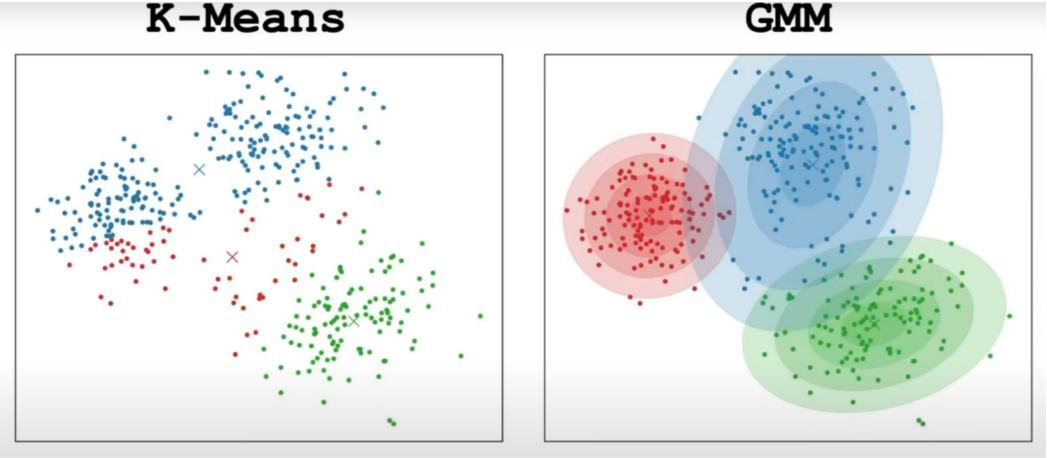
60% red
30 % blue
10% green

- Maximum likelihood principle: choose parameters to maximize likelihood of **observed data**
- We don't observe the cluster assignments z- we only see the data $\mathbf{x}$
- Given data $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$, choose parameters to maximize:

$$\log p(\mathcal{D}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)})$$

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

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(z = k, \mathbf{x}) = \sum_{k=1}^{K} p(z = k) p(\mathbf{x} | z = k)$$



- Our original representation had a hidden (latent) variable z which would represent which Gaussian generated our observation $\mathbf{x}$, with some probability
- Let $z \sim \text{Categorical}(\boldsymbol{\pi})$ (where $\pi_k \geq 0, \quad \sum_k \pi_k = 1$)
- Then:

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}, z = k) \qquad = \sum_{k=1}^{K} \underbrace{p(z = k)}_{\pi_k} \underbrace{p(\mathbf{x}|z = k)}_{\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}$$

- This breaks a complicated distribution into simple components - the price is the hidden variable.

- A Gaussian mixture distribution:

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

- We had: $z \sim \text{Categorical}(\boldsymbol{\pi})$ (where $\pi_k \geq 0, \quad \sum_k \pi_k = 1$)
- Joint distribution: $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z})$
- Log-likelihood:

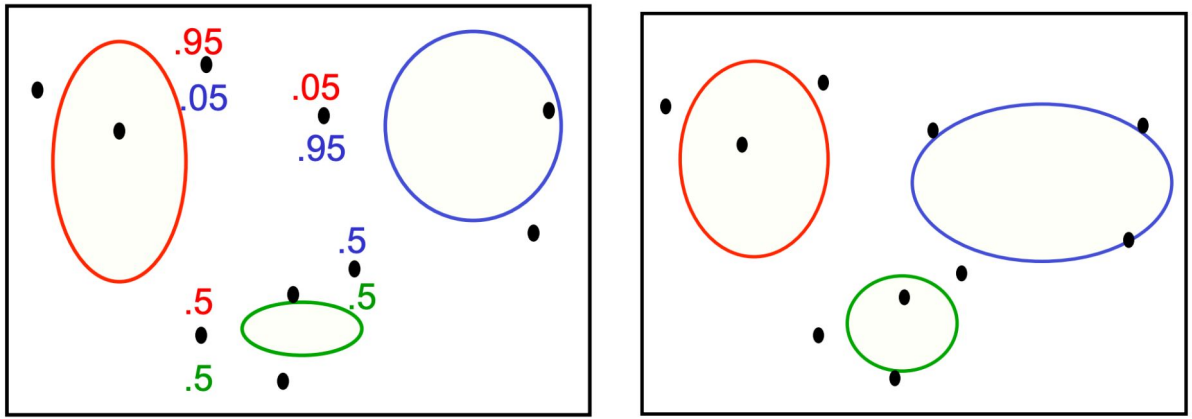
$$\ell(\boldsymbol{\pi}, \mu, \Sigma) = \ln p(\mathbf{X}|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln p(\mathbf{x}^{(n)}|\pi, \mu, \Sigma)$$

$$= \sum_{n=1}^{N} \ln \sum_{z^{(n)}=1}^{K} p(\mathbf{x}^{(n)}|\, z^{(n)}; \mu, \Sigma) p(z^{(n)}|\, \boldsymbol{\pi})$$

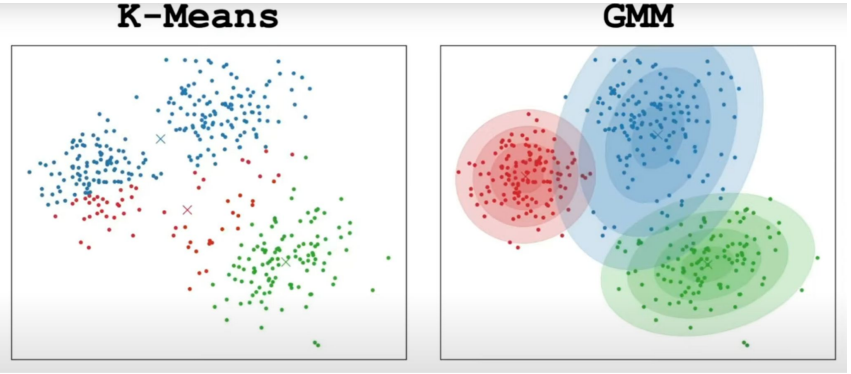
- Note: We have a hidden variable $z^{(n)}$ for every observation
- General problem: sum inside the log
- How can we optimize this?

How to fit GMM with Latent Variable?

- Optimization uses the **Expectation Maximization algorithm**, which alternates between two steps:
  1. **E-step**: Compute the posterior probability over z given our current model - i.e. how much do we think each Gaussian generates each datapoint.
  2. **M-step**: Assuming that the data really was generated this way, change the parameters of each Gaussian to maximize the probability that it would generate the data it is currently responsible for.



Relation to K-Means

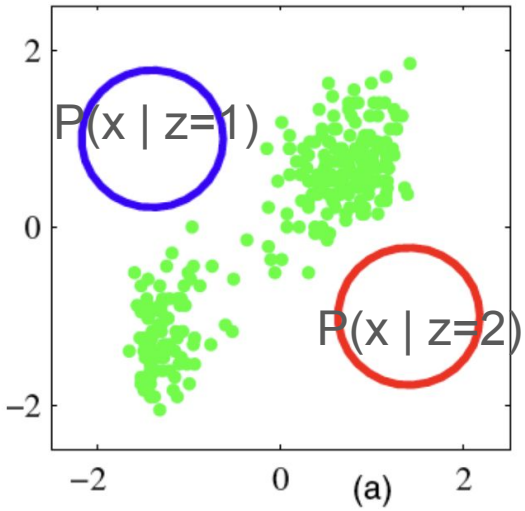


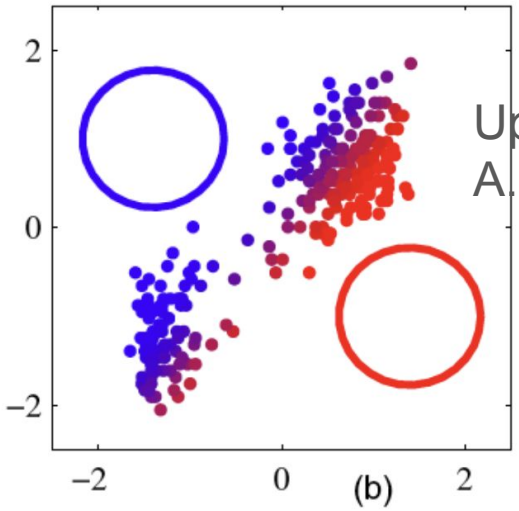
- The K-Means Algorithm:
  1. **Assignment step**: Assign each data point to the closest cluster
  2. **Refitting step**: Move each cluster center to the center of gravity of the data assigned to it
- The EM Algorithm:
  1. **E-step**: Compute the posterior probability over z given our current model
  2. **M-step**: Maximize the probability that it would generate the data it is currently responsible for.

$$p(\mathbf{x}) = \sum_{k=1}^{K} p(\mathbf{x}, z = k) \qquad = \sum_{k=1}^{K} \underbrace{p(z = k)}_{\pi_k} \underbrace{p(\mathbf{x}|z = k)}_{\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}$$

GMM E-Step: Cluster Assignment

- **Which Gaussian generated each datapoint?**





Update P(z=k|x)
A.k.a., “Classification”

$$\underbrace{p(z=k)}_{\pi_k}\underbrace{p(\mathbf{x}|z=k)}_{\mathcal{N}(\mathbf{x}|\mu_k,\Sigma_k)}$$

GMM E-Step: Cluster Assignment

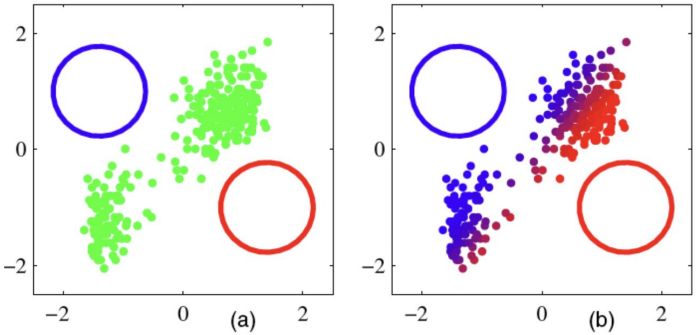
- **Which Gaussian generated each datapoint?**

Equivalent to **Wij** we used for K-means

Probability x belongs to cluster k

$$\gamma_k = p(z = k|\mathbf{x}) = \frac{p(z = k)p(\mathbf{x}|z = k)}{p(\mathbf{x})}$$

$$= \frac{p(z = k)p(\mathbf{x}|z = k)}{\sum_{j=1}^{K} p(z = j)p(\mathbf{x}|z = j)}$$



$$\underbrace{p(z = k)}_{\pi_k} \underbrace{p(\mathbf{x}|z = k)}_{\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}$$

GMM E-Step: Cluster Assignment

- **Which Gaussian generated each datapoint?**

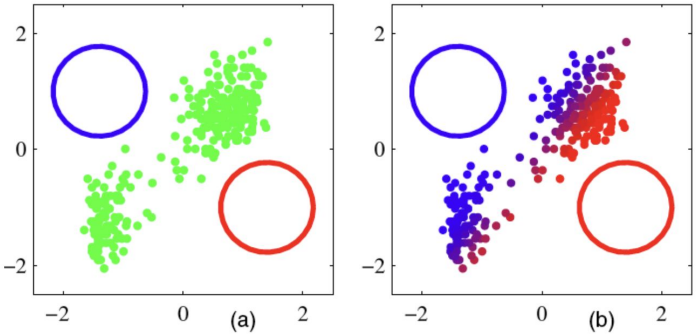
Equivalent to **Wij** we used for K-means

Probability x belongs to cluster k

$$\gamma_k = p(z = k|\mathbf{x}) = \frac{p(z = k)p(\mathbf{x}|z = k)}{p(\mathbf{x})}$$

$$= \frac{p(z = k)p(\mathbf{x}|z = k)}{\sum_{j=1}^{K} p(z = j)p(\mathbf{x}|z = j)}$$

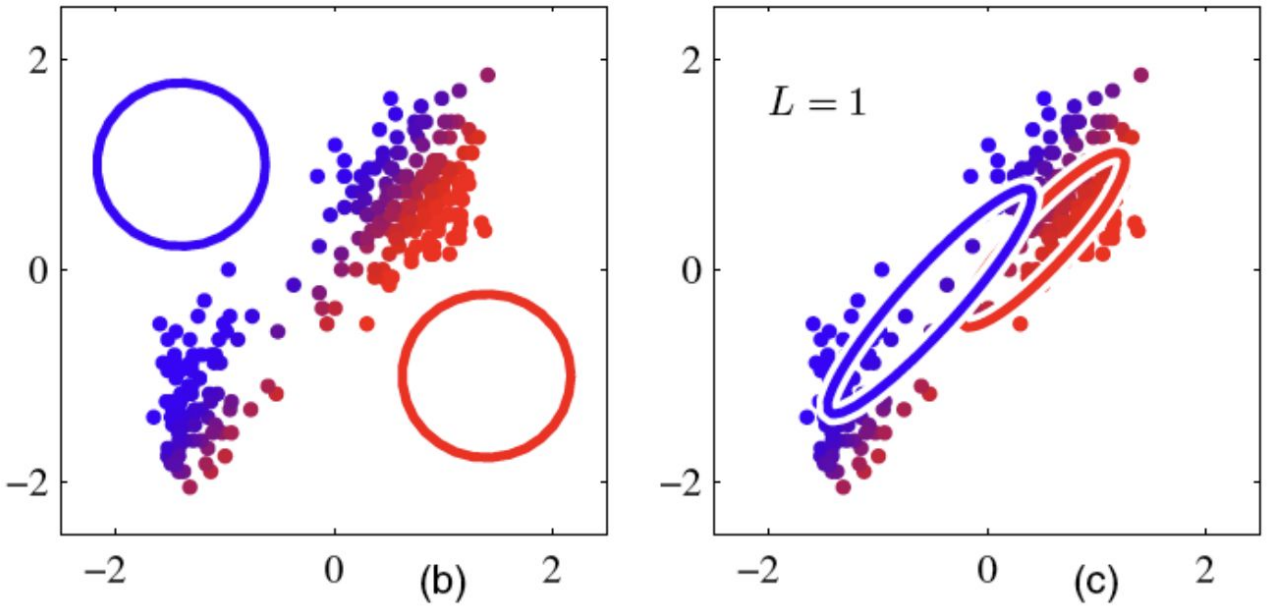
$$= \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, \Sigma_j)}$$



$\underbrace{p(z = k)}_{\pi_k}$ $\underbrace{p(\mathbf{x}|z = k)}_{\mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)}$

GMM M-Step: Cluster Update

Given assignment P(z=k|x), update each gaussian P(x|z)



GMM M-Step: Cluster Update

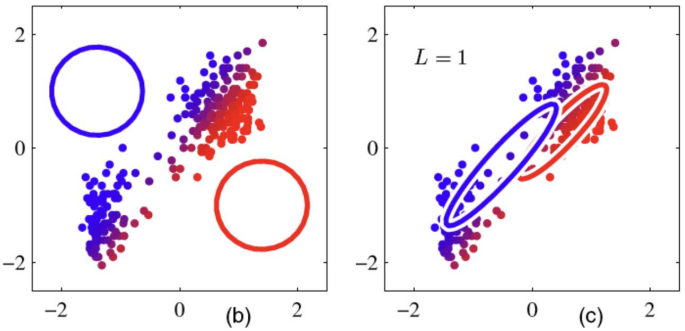
- Need to optimize

$$\sum_k \sum_i \gamma_k^{(i)} \log(\pi_k) + \sum_k \sum_i \gamma_k^{(i)} \log(\mathcal{N}(\mathbf{x}^{(i)}; \mu_k, \Sigma_k))$$

- Solving for μ_k and Σ_k is like fitting k separate Gaussians but with weights $\gamma_k^{(i)}$.
- Solution is similar to what we have already seen:

in single Gaussian MLE (slide 30)

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} \mathbf{x}^{(n)}$$



Exactly the same as K-means!
Except that cluster assignment now is soft (0-1), not hard (0/1)

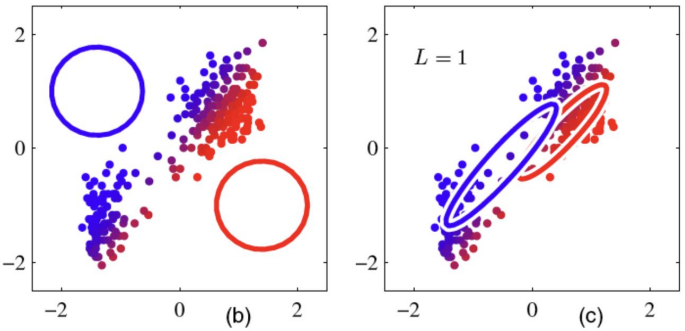
GMM M-Step: Cluster Update

- Need to optimize

$$\sum_k \sum_i \gamma_k^{(i)} \log(\pi_k) + \sum_k \sum_i \gamma_k^{(i)} \log(\mathcal{N}(\mathbf{x}^{(i)}; \mu_k, \Sigma_k))$$

- Solving for μ_k and Σ_k is like fitting k separate Gaussians but with weights $\gamma_k^{(i)}$.
- Solution is similar to what we have already seen:

in single Gaussian MLE (slide 30)



$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} \mathbf{x}^{(n)}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T$$

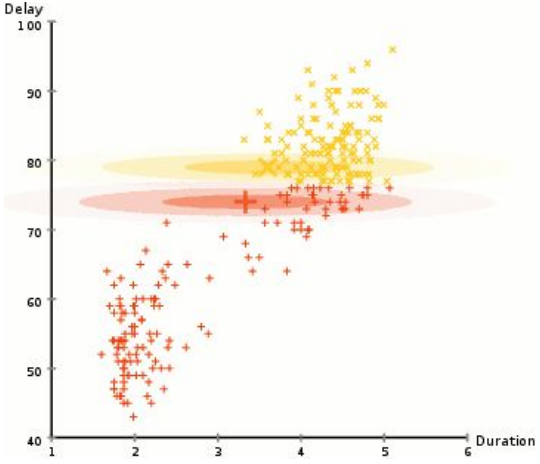
$$\pi_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^{N} \gamma_k^{(n)}$$

EM Algorithm for GMM

- **Initialize** the means μ_k, covariances Σ_k and mixing coefficients π_k
- Iterate until convergence:
  - **E-step**: Evaluate the responsibilities given current parameters

$$\gamma_k^{(n)} = p(z^{(n)}|\mathbf{x}) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}^{(n)}|\mu_j, \Sigma_j)}$$

  - **M-step**: Re-estimate the parameters given current responsibilities



$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} \mathbf{x}^{(n)}$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma_k^{(n)} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T$$

$$\pi_k = \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^{N} \gamma_k^{(n)}$$

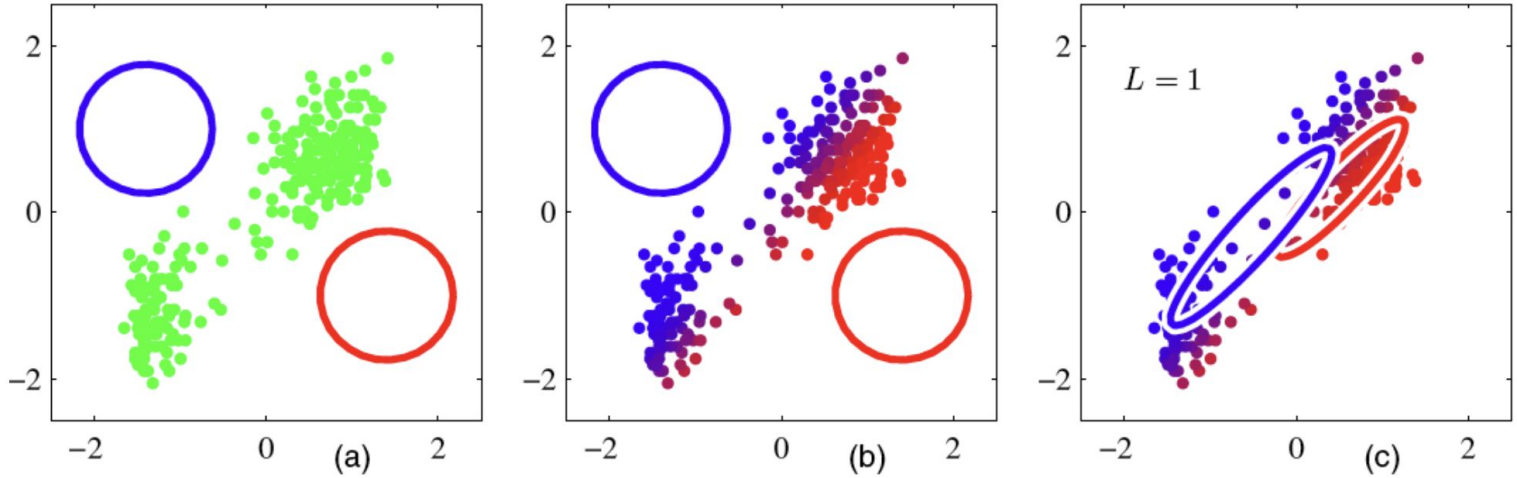
EM Algorithm for GMM

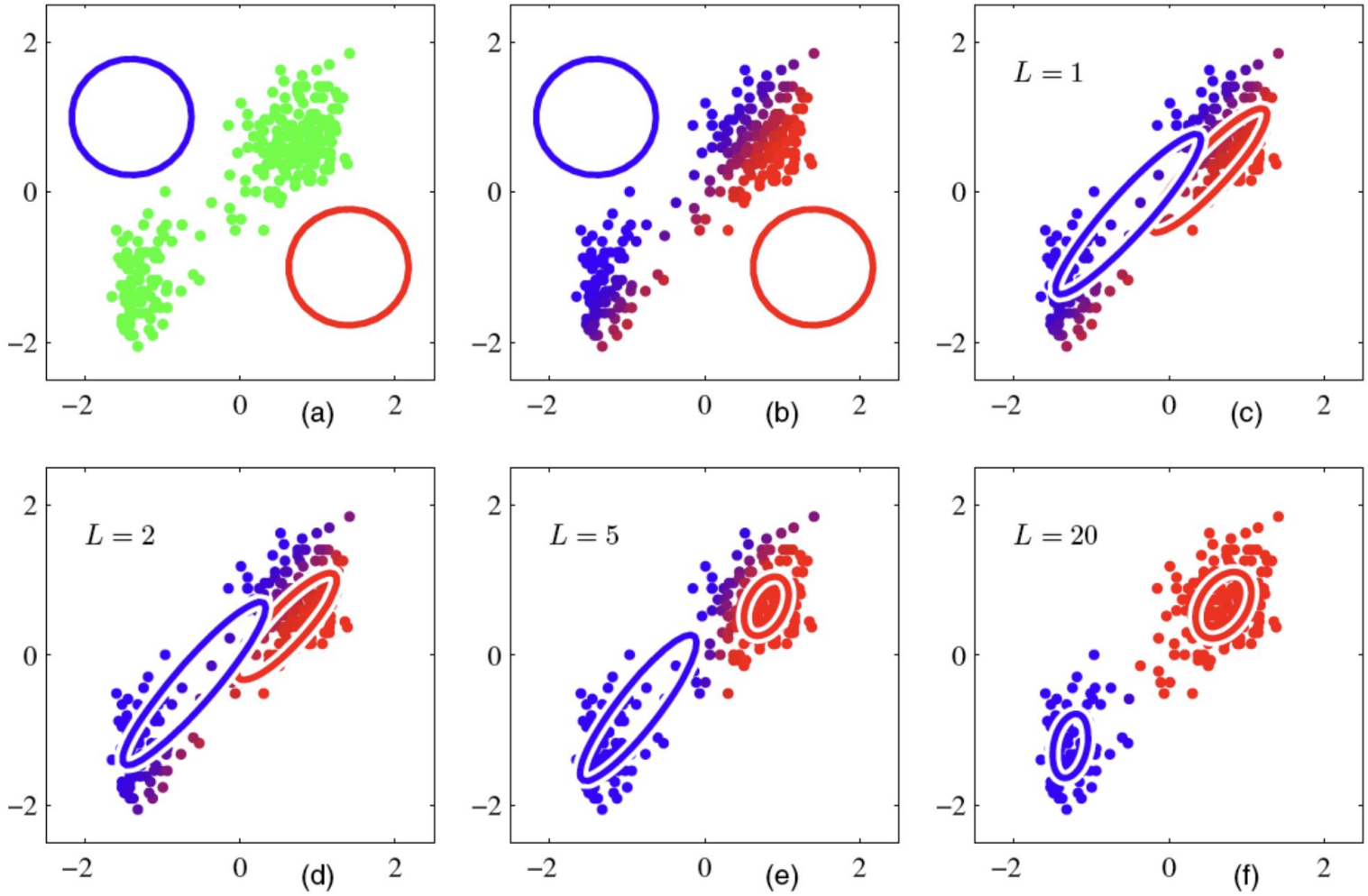
- **Initialize** the means μ_k, covariances Σ_k and mixing coefficients π_k
- Iterate until convergence:
  - **E-step**:

```
def e_step(self, X):
    self.weights = self.predict_proba(X)
    self.phi = self.weights.mean(axis=0)
```

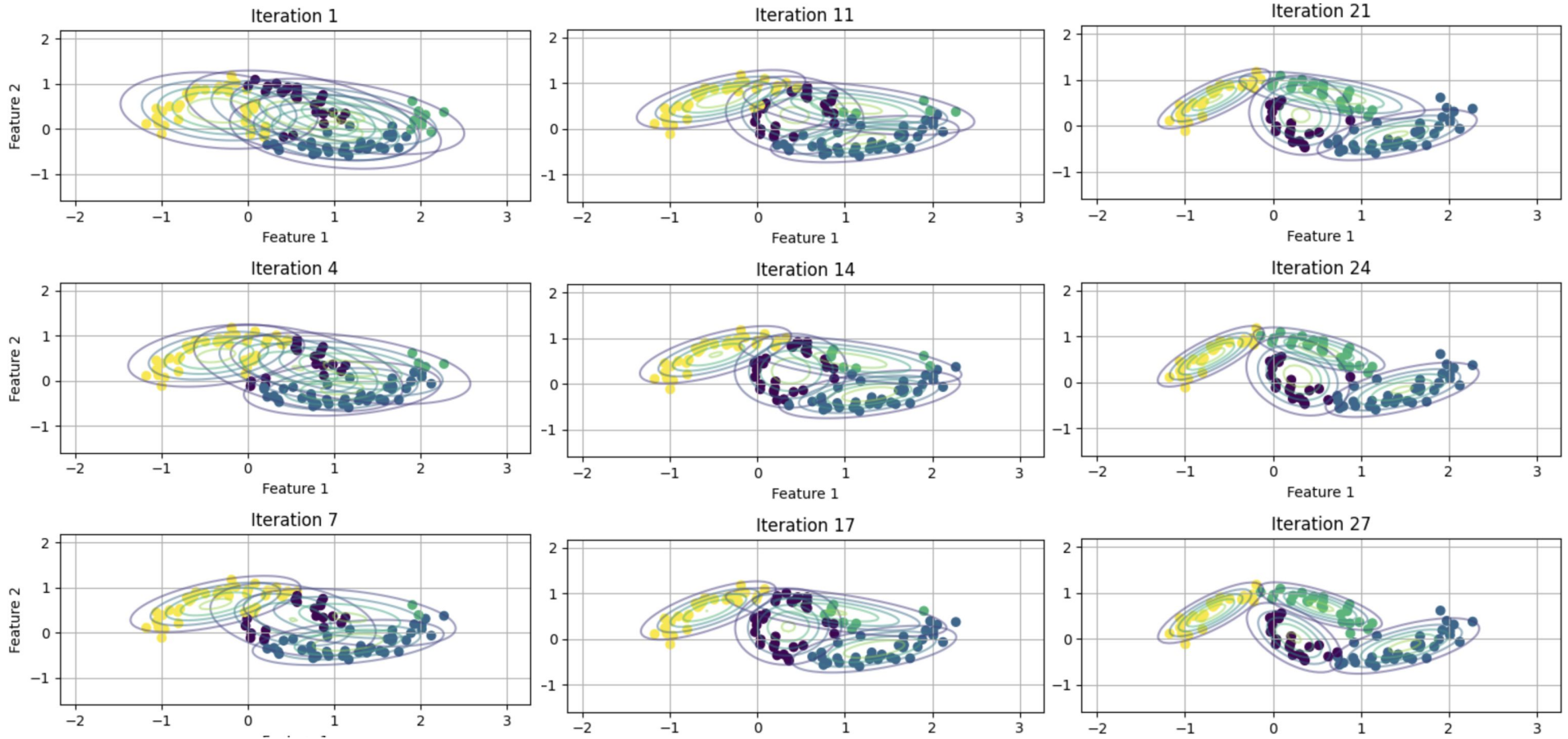
  - **M-step**: Re-estimate the parameters given current responsibilities

```
def m_step(self, X):
    for i in range(self.k):
        weight = self.weights[:, i]  # Simplified to use direct index
        total_weight = weight.sum()
        self.mu[i] = np.dot(weight, X) / total_weight  # Adjusted for
        X_mu = X - self.mu[i]
        self.sigma[i] = np.dot(weight * X_mu.T, X_mu) / total_weight
```


2
0
−2
−2
0
2
(a)
(b)
$L = 1$
(c)


2
0
−2
−2
0
2
(a)
(b)
$L = 1$
(c)
$L = 2$
(d)
$L = 5$
(e)
$L = 20$
(f)

Demo: https://colab.research.google.com/drive/1xcDd9SK0Toi8uQPsWocD--8jb-iankVA?usp=sharing

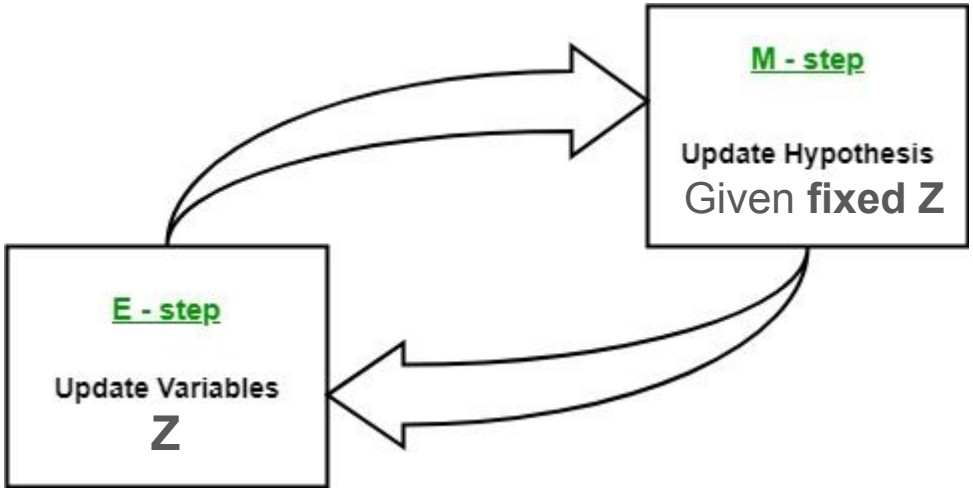


GMM vs K-Means

- EM for mixtures of Gaussians is just like a soft version of K-means, with **fixed priors and covariance**
- Instead of hard assignments in the E-step, we do **soft assignments** based on the softmax of the squared Mahalanobis distance from each point to each cluster.
- Each center moved by **weighted means** of the data, with weights given by soft assignments
- In K-means, weights are 0 or 1

EM Algorithm

Optimize Model with Latent Variable



Latent Variable Model

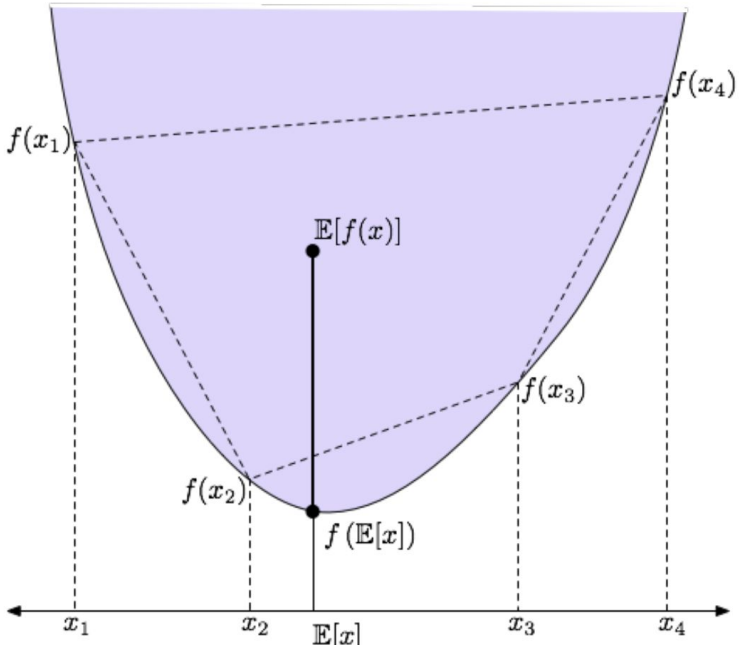
- Recall: variables which are always unobserved are called **latent variables** or sometimes hidden variables
- In a mixture model, the identity of the component that generated a given datapoint is a latent variable

Latent Variable Model

- Recall: variables which are always unobserved are called **latent variables** or sometimes hidden variables
- In a mixture model, the identity of the component that generated a given datapoint is a latent variable
- Why use latent variables if introducing them complicates learning?
  - We can build a complex model out of simple parts - this can simplify the description of the model
  - We can sometimes use the latent variables as a representation of the original data (e.g. cluster assignments in a GMM model)

Jensen's Inequality: For convex f:

$$f(\mathbb{E}[X]) \leq \mathbb{E}[f(X)]$$



- If g is **concave**, the inequality changes directions:

$$g(\mathbb{E}[X]) \geq \mathbb{E}[g(X)]$$

- In this lecture, we'll be using $\mathbf{x}$ to denote **observed data** and z to denote **the latent variables**
- We'll let $p(z, \mathbf{x}; \theta)$ denote the probabilistic model we've defined
  - Anything following a semicolon denotes a parameter of the distribution
  - We're not treating the parameters as random variables

- In this lecture, we'll be using $\mathbf{x}$ to denote **observed data** and z to denote **the latent variables**
- We'll let $p(z, \mathbf{x}; \boldsymbol{\theta})$ denote the probabilistic model we've defined
  - Anything following a semicolon denotes a parameter of the distribution
  - We're not treating the parameters as random variables
- We assume we have an observed dataset $\mathcal{D} = \{\mathbf{x}^{(n)}\}_{n=1}^{N}$ and would like to fit $\boldsymbol{\theta}$ using maximum likelihood:

$$\log p(\mathcal{D}; \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}; \boldsymbol{\theta})$$

- To compute $p(\mathbf{x}; \boldsymbol{\theta})$, we have to **marginalize** over z:

$$p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{z} p(z, \mathbf{x}; \boldsymbol{\theta})$$

- To compute $p(\mathbf{x}; \boldsymbol{\theta})$, we have to **marginalize** over z:

$$p(\mathbf{x}; \boldsymbol{\theta}) = \sum_{z} p(z, \mathbf{x}; \boldsymbol{\theta})$$

- Typically no closed form solution to the maximum likelihood problem

$$\log p(\mathcal{D}; \boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}; \boldsymbol{\theta}) = \sum_{n=1}^{N} \log \left(\sum_{z^{(n)}} p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta}) \right)$$

- Typically no closed form solution to the maximum likelihood problem

$$\log p(\mathcal{D};\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)};\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \left(\sum_{z^{(n)}} p(z^{(n)}, \mathbf{x}^{(n)};\boldsymbol{\theta}) \right)$$

- Key difficulty: once z is marginalized out, $p(\mathbf{x};\boldsymbol{\theta})$ could be complex (e.g. a mixture distribution)
- We'd like to write an objective in terms of $\log p(z, \mathbf{x};\boldsymbol{\theta})$, which should be simpler to solve
- To accomplish this, we need to move the summation outside the log
- We introduce auxilliary distributions $q_n(z^{(n)})$ over each of the latent variables

$$\log p(\mathcal{D};\boldsymbol{\theta}) = \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)};\boldsymbol{\theta}) = \sum_{n=1}^{N} \log \left(\sum_{z^{(n)}} p(z^{(n)}, \mathbf{x}^{(n)};\boldsymbol{\theta}) \right)$$

- We introduce auxilliary distributions $q_n(z^{(n)})$ over each of the latent variables

$$\begin{aligned} \sum_{n=1}^{N} \log \left(\sum_{z^{(n)}} p(z^{(n)}, \mathbf{x}^{(n)};\boldsymbol{\theta}) \right) &= \sum_{n=1}^{N} \log \left(\sum_{z^{(n)}} q_n(z^{(n)}) \frac{p(z^{(n)}, \mathbf{x}^{(n)};\boldsymbol{\theta})}{q_n(z^{(n)})} \right) \\ &= \sum_{n=1}^{N} \log \left(\mathbb{E}_{q_n(z^{(n)})} \left[\frac{p(z^{(n)}, \mathbf{x}^{(n)};\boldsymbol{\theta})}{q_n(z^{(n)})} \right] \right) \end{aligned}$$

- We introduce auxilliary distributions $q_n(z^{(n)})$ over each of the latent variables

$$\sum_{n=1}^{N} \log \left(\sum_{z^{(n)}} p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta}) \right) = \sum_{n=1}^{N} \log \left(\sum_{z^{(n)}} q_n(z^{(n)}) \frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})}{q_n(z^{(n)})} \right)$$

$$= \sum_{n=1}^{N} \log \left(\mathbb{E}_{q_n(z^{(n)})} \left[\frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})}{q_n(z^{(n)})} \right] \right)$$

$$\geq \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})} \left[\log \frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})}{q_n(z^{(n)})} \right]$$

If g is **concave**, the inequality changes directions:

$$g(\mathbb{E}[X]) \geq \mathbb{E}[g(X)]$$

- In the last step, we use Jensen's Inequality. Since log is concave:

$$\log \left(\mathbb{E}_{q_n(z^{(n)})} \left[\frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})}{q_n(z^{(n)})} \right] \right) \geq \mathbb{E}_{q_n(z^{(n)})} \left[\log \frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})}{q_n(z^{(n)})} \right]$$

$$\sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}; \boldsymbol{\theta}) \geq \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})} \left[\log \frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})}{q_n(z^{(n)})} \right]$$
$$\equiv \mathcal{L}(q, \boldsymbol{\theta}) \text{ where } q = \{q_1, \ldots, q_N\}$$

- We expect $\mathcal{L}(q, \boldsymbol{\theta})$ might be easier to optimize w.r.t. $\boldsymbol{\theta}$, since it only appears in $\log p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})$, so we'll use this as our new objective
- For **any** auxilliary distributions q_n, we obtain a lower bound on the log likelihood
- Which q_n should we choose? Want to make the bound as tight as possible

- We know this bound is tight (i.e. the inequality becomes an equality) if there are constants c_n such that:

$$\frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})}{q_n(z^{(n)})} = \text{constant} \implies q_n(z^{(n)}) = c_n p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})$$

- Using $\sum_{z^{(n)}} q_n(z^{(n)}) = 1$, we have:

$$1 = \sum_{z^{(n)}} q_n(z^{(n)}) = c_n \sum_{z^{(n)}} p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta}) = c_n p(\mathbf{x}^{(n)}; \boldsymbol{\theta})$$

$$\implies c_n = \frac{1}{p(\mathbf{x}^{(n)}; \boldsymbol{\theta})}$$

- Hence:

$$q_n(z^{(n)}) = \frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})}{p(\mathbf{x}^{(n)}; \boldsymbol{\theta})} = p(z^{(n)} | \mathbf{x}^{(n)}; \boldsymbol{\theta})$$

- For fixed $\boldsymbol{\theta}_0$, if we set $q_n(z^{(n)}) = p(z^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}_0)$ the bound is tight:

$$\sum_{n=1}^{N} \log p(\mathbf{x}^{(n)}; \boldsymbol{\theta}_0) = \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})} \left[\log \frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta}_0)}{q_n(z^{(n)})} \right]$$

- Written another way:

$$\log p(\mathcal{D}; \boldsymbol{\theta}_0) = \mathcal{L}(q; \boldsymbol{\theta}_0) \text{ if } \forall\ n\,, q_n(z^{(n)}) = p(z^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}_0)$$

- The EM algorithm alternates between making the bound tight at the current parameter values and then optimizing the lower bound
- If the current parameter value is $\boldsymbol{\theta}^{\text{old}}$:
  - **E-step**: For all n, set $q_n(z^{(n)}) = p(z^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}})$ and form the lower bound $\mathcal{L}(q; \boldsymbol{\theta})$
    - Remember: $\log p(\mathcal{D}; \boldsymbol{\theta}^{\text{old}}) = \mathcal{L}(q; \boldsymbol{\theta}^{\text{old}})$ after this step
  - **M-step**: Optimize the lower bound:

$$\begin{aligned}\boldsymbol{\theta}^{\text{new}} &= \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}(q, \boldsymbol{\theta}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})} \left[\log \frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})}{q_n(z^{(n)})} \right]\end{aligned}$$

- **M-step**: Optimize the lower bound:

$$\sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})} \left[\log \frac{p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta})}{q_n(z^{(n)})} \right] =$$

$$\sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})} \left[\log p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta}) \right] - \underbrace{\mathbb{E}_{q_n(z^{(n)})} \left[\log q_n(z^{(n)}) \right]}_{\text{constant w.r.t.} \boldsymbol{\theta}}$$

- Substitute in $q_n(z^{(n)}) = p(z^{(n)} | \mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}})$:

$$\boldsymbol{\theta}^{\text{new}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{p(z^{(n)} | \mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}})} \left[\log p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta}) \right]$$

- This is the expected complete data log-likelihood.

- **E-step**: For all n, set $q_n(z^{(n)}) = p(z^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}})$ and form the lower bound $\mathcal{L}(q; \boldsymbol{\theta})$
- **M-step**: Optimize the lower bound:

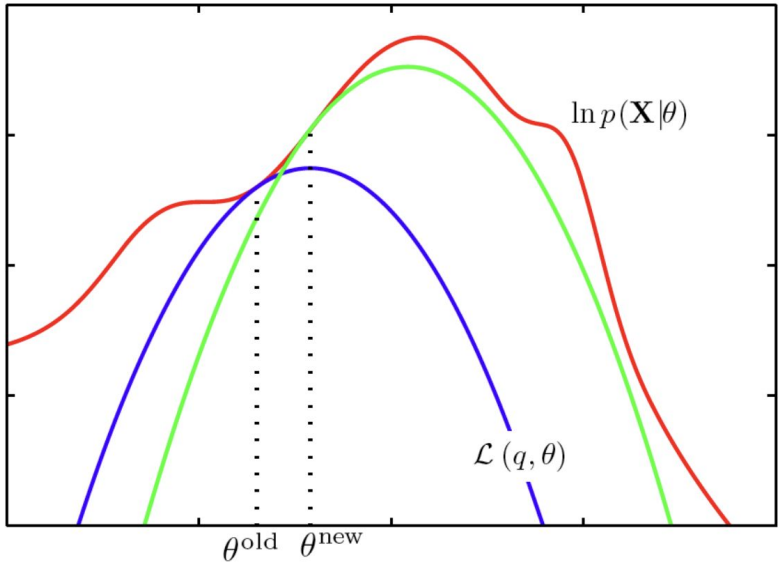
$$\begin{aligned}\boldsymbol{\theta}^{\text{new}} &= \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}(q, \boldsymbol{\theta}) \\ &= \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_{n=1}^{N} \mathbb{E}_{p(z^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}})} \left[\log p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta}) \right]\end{aligned}$$

EM Convergence

- We can deduce that an iteration of EM will improve the log-likelihood by using the fact that the bound is tight at $\boldsymbol{\theta}^{\text{old}}$ after the E-step
- Let q denote the q_n's after the E-step i.e. $q_n(z^{(n)}) = p(z^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}})$

$$\begin{aligned}
\log p(\mathcal{D}; \boldsymbol{\theta}^{\text{new}}) &\geq \mathcal{L}(q, \boldsymbol{\theta}^{\text{new}}) && \text{since } \log p(\mathcal{D}; \boldsymbol{\theta}) \geq \mathcal{L}(q, \boldsymbol{\theta}) \text{ always} \\
&\geq \mathcal{L}(q, \boldsymbol{\theta}^{\text{old}}) && \text{since } \boldsymbol{\theta}^{\text{new}} = \operatorname*{argmax}_{\boldsymbol{\theta}} \mathcal{L}(q, \boldsymbol{\theta}) \\
&= \log p(\mathcal{D}; \boldsymbol{\theta}^{\text{old}}) && \text{since } \log p(\mathcal{D}; \boldsymbol{\theta}^{\text{old}}) = \mathcal{L}(q; \boldsymbol{\theta}^{\text{old}})
\end{aligned}$$

EM Visualization



- The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values

Revisit GMM

- Let's revisit the mixture of Gaussians example from last lecture and derive the updates using our general EM algorithm
- Recall our model was:

$$p(z = k; \boldsymbol{\theta}) = \pi_k$$

$$p(\mathbf{x}|z = k; \boldsymbol{\theta}) = \mathcal{N}(\mathbf{x}; \mu_k, \Sigma_k)$$

- In this scenario, we have $\boldsymbol{\theta} = \{\mu_k, \pi_k, \Sigma_k\}_{k=1}^K$

E-Step for GMM

- Let the current parameters be $\boldsymbol{\theta}^{\text{old}} = \{\mu_k^{\text{old}}, \pi_k^{\text{old}}, \Sigma_k^{\text{old}}\}_{k=1}^K$
- **E-step**: For all n, set $q_n(z^{(n)}) = p(z^{(n)}|\mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}})$

$$r_k^{(n)} := q_n(z^{(n)} = k) = p(z^{(n)} = k|\mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}}) = \frac{\pi_k^{\text{old}} \mathcal{N}(\mathbf{x}^{(n)}|\mu_k^{\text{old}}, \Sigma_k^{\text{old}})}{\sum_{j=1}^K \pi_j^{\text{old}} \mathcal{N}(\mathbf{x}^{(n)}|\mu_j^{\text{old}}, \Sigma_j^{\text{old}})}$$

M-Step for GMM

M-step:

$$\boldsymbol{\theta}^{\text{new}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})} \left[\log p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta}) \right]$$

- Substitute in:
  - $\log p(z^{(n)}, \mathbf{x}^{(n)}; \boldsymbol{\theta}) = \sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k] \left(\log \pi_k + \log \mathcal{N}(\mathbf{x}^{(n)}; \mu_k, \Sigma_k) \right)$
  - $q_n(z^{(n)}) = p(z^{(n)} | \mathbf{x}^{(n)}; \boldsymbol{\theta}^{\text{old}})$:

$$\begin{aligned}
\boldsymbol{\theta}^{\text{new}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \mathbb{E}_{q_n(z^{(n)})} \left[\sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k] \left(\log \pi_k + \log \mathcal{N}(\mathbf{x}^{(n)}; \mu_k, \Sigma_k) \right) \right] \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} \left(\log \pi_k + \log \mathcal{N}(\mathbf{x}^{(n)}; \mu_k, \Sigma_k) \right)
\end{aligned}$$

M-Step for GMM

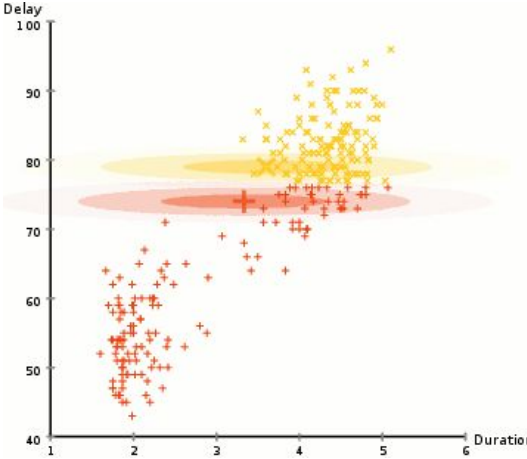
$$\boldsymbol{\theta}^{\text{new}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)} \left(\log \pi_k + \mathcal{N}(\mathbf{x}^{(n)}; \mu_k, \Sigma_k) \right)$$

- Taking derivatives and setting to zero, we get the updates from last lecture:

$$\begin{aligned}
\mu_k &= \frac{1}{N_k} \sum_{n=1}^{N} r_k^{(n)} \mathbf{x}^{(n)} \\
\Sigma_k &= \frac{1}{N_k} \sum_{n=1}^{N} r_k^{(n)} (\mathbf{x}^{(n)} - \mu_k)(\mathbf{x}^{(n)} - \mu_k)^T \\
\pi_k &= \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{n=1}^{N} r_k^{(n)}
\end{aligned}$$

Summary of EM & GMM

- A general algorithm for optimizing many latent variable models.
- Iteratively computes a lower bound then optimizes it.
- Converges but maybe to a local minima.
- Can use multiple restarts.
- Can initialize from k-means for mixture models
- Limitation - need to be able to compute $p(z|\mathbf{x};\boldsymbol{\theta})$, not possible for more complicated models.



MLE given fixed z (diagonal case)

- **Observation: if we knew** $z^{(n)}$ for every $\mathbf{x}^{(n)}$, (i.e. our dataset was $\mathcal{D}_{\text{complete}} = \{(z^{(n)}, \mathbf{x}^{(n)})\}_{n=1}^{N}$) the maximum likelihood problem is easy:

$$\begin{aligned}
\log p(\mathcal{D}_{\text{complete}}) &= \sum_{n=1}^{N} \log p(z^{(n)}, \mathbf{x}^{(n)}) \\
&= \sum_{n=1}^{N} \log p(\mathbf{x}^{(n)} | z^{(n)}) + \log p(z^{(n)}) \\
&= \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k](\log \mathcal{N}(\mathbf{x}^{(n)} | \mu_k, I) + \log \pi_k)
\end{aligned}$$

MLE given fixed z (diagonal case)

$$\log p(\mathcal{D}_{\text{complete}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k](\log \mathcal{N}(x^{(n)}|\mu_k, I) + \log \pi_k)$$

- We have been optimizing something similar for Gaussian bayes classifiers
- We would get this:

$$\mu_k = \frac{\sum_{n=1}^{N} \mathbb{I}[z^{(n)} = k]\, \mathbf{x}^{(n)}}{\sum_{n=1}^{N} \mathbb{I}[z^{(n)} = k]}$$

$$\pi_k = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}[z^{(n)} = k]$$

How to Estimate z (diagonal case)?

- We don't know $z^{(n)}$ for every $\mathbf{x}^{(n)}$, but we can compute $p(z^{(n)}|\mathbf{x}^{(n)})$ using Bayes rule
- Conditional probability (using Bayes rule) of $\mathbf{z}$ given $\mathbf{x}$

$$
\begin{aligned}
p(z = k|\mathbf{x}) &= \frac{p(z = k)p(\mathbf{x}|z = k)}{p(\mathbf{x})} \\
&= \frac{p(z = k)p(\mathbf{x}|z = k)}{\sum_{j=1}^{K} p(z = j)p(\mathbf{x}|z = j)} \\
&= \frac{\pi_k \mathcal{N}(\mathbf{x}|\mu_k, I)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\mu_j, I)}
\end{aligned}
$$

$$\log p(\mathcal{D}_{\text{complete}}) = \sum_{n=1}^{N} \sum_{k=1}^{K} \mathbb{I}[z^{(n)} = k](\log \mathcal{N}(x^{(n)}|\mu_k, I) + \log \pi_k)$$

- If we plug in $r_k^{(n)} = p(z^{(n)} = k|\mathbf{x}^{(n)})$ for $\mathbb{I}[z^{(n)} = k]$, we get:

$$\sum_{n=1}^{N} \sum_{k=1}^{K} r_k^{(n)}(\log \mathcal{N}(\mathbf{x}^{(n)}|\mu_k, I) + \log \pi_k)$$

- This is still easy to optimize! Solution is similar to what we have seen:

$$\mu_k = \frac{\sum_{n=1}^{N} r_k^{(n)} \mathbf{x}^{(n)}}{\sum_{n=1}^{N} r_k^{(n)}}$$

$$\pi_k = \frac{\sum_{n=1}^{N} r_k^{(n)}}{N}$$

- Note: this only works if we treat $r_k^{(n)}$ as fixed – really, it depends on the model parameters as well