

# Clustering and Latent Variable Models

Mengye Ren

NYU

Dec 5, 2023

# K-means Clustering

# Unsupervised learning

**Goal** Discover interesting *structure* in the data.

# Unsupervised learning

**Goal** Discover interesting *structure* in the data.

**Formulation** Density estimation:  $p(x; \theta)$  (often with *latent* variables).

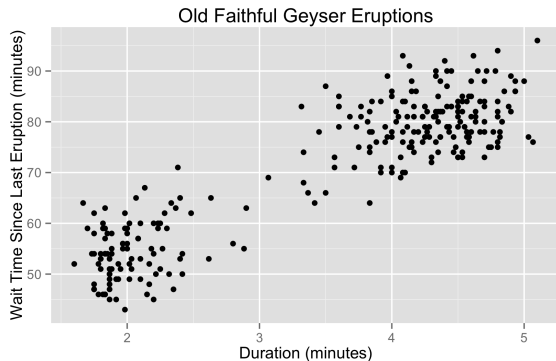
# Unsupervised learning

**Goal** Discover interesting *structure* in the data.

**Formulation** Density estimation:  $p(x; \theta)$  (often with *latent* variables).

- Examples**
- Discover *clusters*: cluster data into groups.
  - Discover *factors*: project high-dimensional data to a small number of “meaningful” dimensions, i.e. dimensionality reduction.
  - Discover *graph structures*: learn joint distribution of correlated variables, i.e. graphical models.

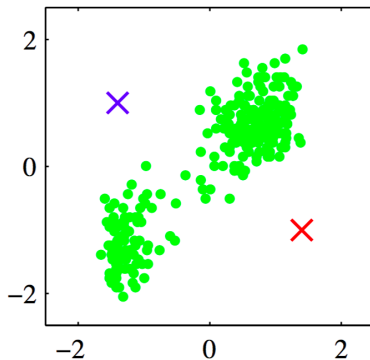
## Example: Old Faithful Geyser



- Looks like two clusters.
- How to find these clusters algorithmically?

## k-Means: By Example

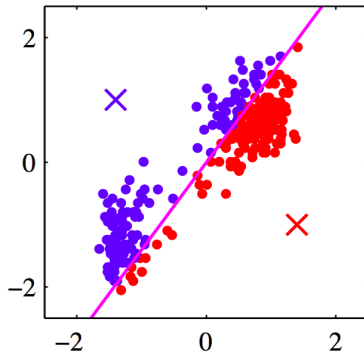
- Standardize the data.
- Choose two cluster centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(a).

## k-means: by example

- Assign each point to closest center.

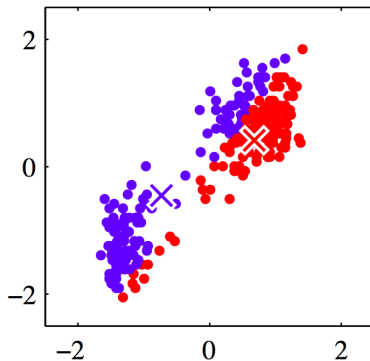


From Bishop's *Pattern recognition and machine learning*, Figure 9.1(b).



## k-means: by example

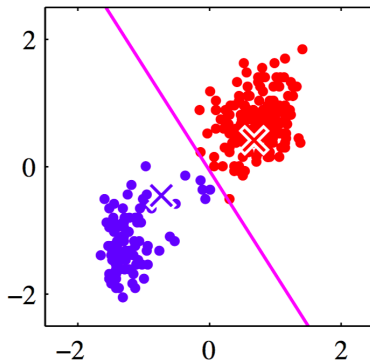
- Compute new cluster centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(c).

## k-means: by example

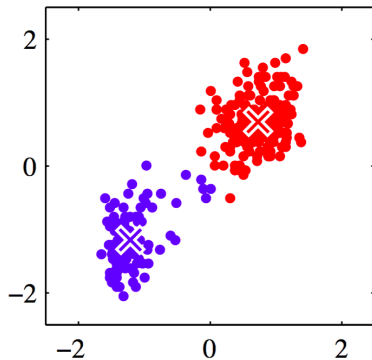
- Assign points to closest center.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(d).

## $k$ -means: by example

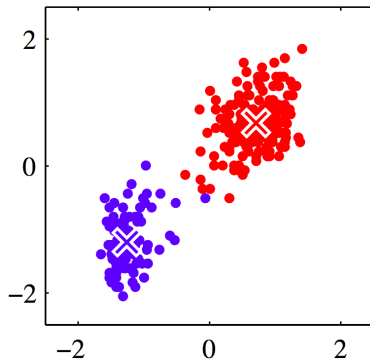
- Compute cluster centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(e).

## $k$ -means: by example

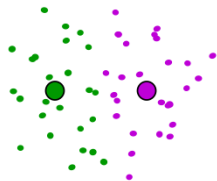
- Iterate until convergence.



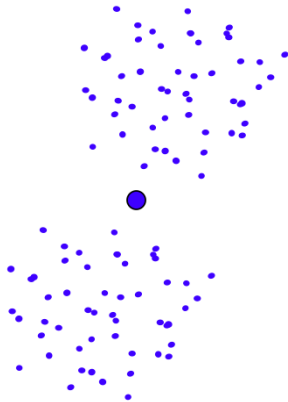
From Bishop's *Pattern recognition and machine learning*, Figure 9.1(i).

# Suboptimal Local Minimum

- The clustering for  $k = 3$  below is a local minimum, but suboptimal:



Would be better to have  
one cluster here



... and two clusters here

## Formalize $k$ -Means

- Dataset  $\mathcal{D} = \{x_1, \dots, x_n\} \subset \mathcal{X}$  where  $\mathcal{X} = \mathbb{R}^d$ .

## Formalize $k$ -Means

- Dataset  $\mathcal{D} = \{x_1, \dots, x_n\} \subset \mathcal{X}$  where  $\mathcal{X} = \mathbb{R}^d$ .
- Goal: Partition data  $\mathcal{D}$  into  $k$  disjoint sets  $C_1, \dots, C_k$ .

## Formalize $k$ -Means

- Dataset  $\mathcal{D} = \{x_1, \dots, x_n\} \subset \mathcal{X}$  where  $\mathcal{X} = \mathbb{R}^d$ .
- Goal: Partition data  $\mathcal{D}$  into  $k$  disjoint sets  $C_1, \dots, C_k$ .
- Let  $c_i \in \{1, \dots, k\}$  be the cluster assignment of  $x_i$ .



## Formalize $k$ -Means

- Dataset  $\mathcal{D} = \{x_1, \dots, x_n\} \subset \mathcal{X}$  where  $\mathcal{X} = \mathbb{R}^d$ .
- Goal: Partition data  $\mathcal{D}$  into  $k$  disjoint sets  $C_1, \dots, C_k$ .
- Let  $c_i \in \{1, \dots, k\}$  be the cluster assignment of  $x_i$ .
- The **centroid** of  $C_i$  is defined to be

$$\mu_i = \arg \min_{\mu \in \mathcal{X}} \sum_{x \in C_i} \|x - \mu\|^2. \quad \text{mean of } C_i \quad (1)$$

# Formalize $k$ -Means

- Dataset  $\mathcal{D} = \{x_1, \dots, x_n\} \subset \mathcal{X}$  where  $\mathcal{X} = \mathbb{R}^d$ .
- Goal: Partition data  $\mathcal{D}$  into  $k$  disjoint sets  $C_1, \dots, C_k$ .
- Let  $c_i \in \{1, \dots, k\}$  be the cluster assignment of  $x_i$ .
- The **centroid** of  $C_i$  is defined to be

$$\mu_i = \arg \min_{\mu \in \mathcal{X}} \sum_{x \in C_i} \|x - \mu\|^2. \quad \text{mean of } C_i \quad (1)$$

- The  $k$ -means objective is to minimize the distance between each example and its cluster centroid:

$$J(c, \mu) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2. \quad (2)$$

# $k$ -Means: Algorithm

- 1 Initialize: Randomly choose initial centroids  $\mu_1, \dots, \mu_k \in \mathbb{R}^d$ .

# $k$ -Means: Algorithm

- 1 Initialize: Randomly choose initial centroids  $\mu_1, \dots, \mu_k \in \mathbb{R}^d$ .
- 2 Repeat until convergence (i.e.  $c_i$  doesn't change anymore):

# $k$ -Means: Algorithm

- 1 Initialize: Randomly choose initial centroids  $\mu_1, \dots, \mu_k \in \mathbb{R}^d$ .
- 2 Repeat until convergence (i.e.  $c_i$  doesn't change anymore):
  - 1 For all  $i$ , set

$$c_i \leftarrow \arg \min_j \|x_i - \mu_j\|^2. \quad (3)$$

# k-Means: Algorithm

- ➊ Initialize: Randomly choose initial centroids  $\mu_1, \dots, \mu_k \in \mathbb{R}^d$ .
- ➋ Repeat until convergence (i.e.  $c_i$  doesn't change anymore):
  - ➊ For all  $i$ , set

$$c_i \leftarrow \arg \min_j \|x_i - \mu_j\|^2. \quad (3)$$

- ➋ For all  $j$ , set

$$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \quad (4)$$

# k-Means: Algorithm

- ➊ Initialize: Randomly choose initial centroids  $\mu_1, \dots, \mu_k \in \mathbb{R}^d$ .
- ➋ Repeat until convergence (i.e.  $c_i$  doesn't change anymore):
  - ➊ For all  $i$ , set

$$c_i \leftarrow \arg \min_j \|x_i - \mu_j\|^2. \quad (3)$$

- ➋ For all  $j$ , set

$$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \quad (4)$$

- Recall the objective:  $J(c, \mu) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2$ .

# k-Means: Algorithm

- 1 Initialize: Randomly choose initial centroids  $\mu_1, \dots, \mu_k \in \mathbb{R}^d$ .
- 2 Repeat until convergence (i.e.  $c_i$  doesn't change anymore):

- 1 For all  $i$ , set

$$c_i \leftarrow \arg \min_j \|x_i - \mu_j\|^2. \quad \text{Minimize } J \text{ w.r.t. } c \text{ while fixing } \mu \quad (3)$$

- 2 For all  $j$ , set

$$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \quad (4)$$

- Recall the objective:  $J(c, \mu) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2$ .



# k-Means: Algorithm

- 1 Initialize: Randomly choose initial centroids  $\mu_1, \dots, \mu_k \in \mathbb{R}^d$ .
- 2 Repeat until convergence (i.e.  $c_i$  doesn't change anymore):

- 1 For all  $i$ , set

$$c_i \leftarrow \arg \min_j \|x_i - \mu_j\|^2. \quad \text{Minimize } J \text{ w.r.t. } c \text{ while fixing } \mu \quad (3)$$

- 2 For all  $j$ , set

$$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \quad \text{Minimize } J \text{ w.r.t. } \mu \text{ while fixing } c. \quad (4)$$

- Recall the objective:  $J(c, \mu) = \sum_{i=1}^n \|x_i - \mu_{c_i}\|^2$ .

# Avoid bad local minima

$k$ -means converges to a local minimum.

- $J$  is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.

## Avoid bad local minima

$k$ -means converges to a local minimum.

- $J$  is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.
- **$k$ -means++**: choose initial centroids that spread over all data points.

## Avoid bad local minima

$k$ -means converges to a local minimum.

- $J$  is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.
- **$k$ -means++**: choose initial centroids that spread over all data points.
  - Randomly choose the first centroid from the data points  $\mathcal{D}$ .

## Avoid bad local minima

$k$ -means converges to a local minimum.

- $J$  is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.
- **$k$ -means++**: choose initial centroids that spread over all data points.
  - Randomly choose the first centroid from the data points  $\mathcal{D}$ .
  - Sequentially choose subsequent centroids from points that are farther away from current centroids:

## Avoid bad local minima

$k$ -means converges to a local minimum.

- $J$  is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.
- **$k$ -means++**: choose initial centroids that spread over all data points.
  - Randomly choose the first centroid from the data points  $\mathcal{D}$ .
  - Sequentially choose subsequent centroids from points that are farther away from current centroids:
    - Compute distance between each  $x_i$  and the closest already chosen centroids.

## Avoid bad local minima

$k$ -means converges to a local minimum.

- $J$  is non-convex, thus no guarantee to converging to the global minimum.

Avoid getting stuck with bad local minima:

- Re-run with random initial centroids.
- **$k$ -means++**: choose initial centroids that spread over all data points.
  - Randomly choose the first centroid from the data points  $\mathcal{D}$ .
  - Sequentially choose subsequent centroids from points that are farther away from current centroids:
    - Compute distance between each  $x_i$  and the closest already chosen centroids.
    - Randomly choose next centroid with probability proportional to the computed distance squared.

# Summary

We've seen

- Clustering—an unsupervised learning problem that aims to discover group assignments.
- $k$ -means:
  - Algorithm: alternating between assigning points to clusters and computing cluster centroids.
  - Objective: minimizing some loss function by coordinate descent.
  - Converge to a local minimum.



# Summary

We've seen

- Clustering—an unsupervised learning problem that aims to discover group assignments.
- $k$ -means:
  - Algorithm: alternating between assigning points to clusters and computing cluster centroids.
  - Objective: minimizing some loss function by coordinate descent.
  - Converge to a local minimum.

Next, probabilistic model of clustering.

- A generative model of  $x$ .
- Maximum likelihood estimation.

# Gaussian Mixture Models

# Probabilistic Model for Clustering

- Problem setup:
  - There are  $k$  clusters (or **mixture components**).
  - We have a probability distribution for each cluster.

# Probabilistic Model for Clustering

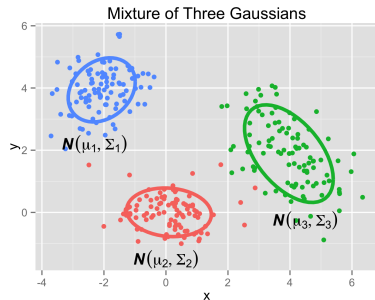
- Problem setup:
  - There are  $k$  clusters (or **mixture components**).
  - We have a probability distribution for each cluster.
- Generative story of a **mixture distribution**:
  - 1 Choose a random cluster  $z \in \{1, 2, \dots, k\}$ .
  - 2 Choose a point from the distribution for cluster  $z$ .

# Probabilistic Model for Clustering

- Problem setup:
  - There are  $k$  clusters (or **mixture components**).
  - We have a probability distribution for each cluster.
- Generative story of a **mixture distribution**:
  - 1 Choose a random cluster  $z \in \{1, 2, \dots, k\}$ .
  - 2 Choose a point from the distribution for cluster  $z$ .

Example:

- 1 Choose  $z \in \{1, 2, 3\}$  with  $p(1) = p(2) = p(3) = \frac{1}{3}$ .
- 2 Choose  $x \mid z \sim \mathcal{N}(X \mid \mu_z, \Sigma_z)$ .



# Gaussian mixture model (GMM)

Generative story of GMM with  $k$  mixture components:

- 1 Choose cluster  $z \sim \text{Categorical}(\pi_1, \dots, \pi_k)$ .
- 2 Choose  $x \mid z \sim \mathcal{N}(\mu_z, \Sigma_z)$ .

# Gaussian mixture model (GMM)

Generative story of GMM with  $k$  mixture components:

- 1 Choose cluster  $z \sim \text{Categorical}(\pi_1, \dots, \pi_k)$ .
- 2 Choose  $x \mid z \sim \mathcal{N}(\mu_z, \Sigma_z)$ .

Probability density of  $x$ :

- Sum over (marginalize) the **latent variable**  $z$ .

$$p(x) = \sum_z p(x, z) \tag{5}$$

$$= \sum_z p(x \mid z) p(z) \tag{6}$$

$$= \sum_k \pi_k \mathcal{N}(\mu_k, \Sigma_k) \tag{7}$$

# Identifiability Issues for GMM

- Suppose we have found parameters

Cluster probabilities:  $\pi = (\pi_1, \dots, \pi_k)$

Cluster means:  $\mu = (\mu_1, \dots, \mu_k)$

Cluster covariance matrices:  $\Sigma = (\Sigma_1, \dots, \Sigma_k)$

that are at a local minimum.



# Identifiability Issues for GMM

- Suppose we have found parameters

Cluster probabilities:  $\pi = (\pi_1, \dots, \pi_k)$

Cluster means:  $\mu = (\mu_1, \dots, \mu_k)$

Cluster covariance matrices:  $\Sigma = (\Sigma_1, \dots, \Sigma_k)$

that are at a local minimum.

- What happens if we shuffle the clusters? e.g. Switch the labels for clusters 1 and 2.

# Identifiability Issues for GMM

- Suppose we have found parameters

Cluster probabilities:  $\pi = (\pi_1, \dots, \pi_k)$

Cluster means:  $\mu = (\mu_1, \dots, \mu_k)$

Cluster covariance matrices:  $\Sigma = (\Sigma_1, \dots, \Sigma_k)$

that are at a local minimum.

- What happens if we shuffle the clusters? e.g. Switch the labels for clusters 1 and 2.
- We'll get the same likelihood. How many such equivalent settings are there?

# Identifiability Issues for GMM

- Suppose we have found parameters

Cluster probabilities:  $\pi = (\pi_1, \dots, \pi_k)$

Cluster means:  $\mu = (\mu_1, \dots, \mu_k)$

Cluster covariance matrices:  $\Sigma = (\Sigma_1, \dots, \Sigma_k)$

that are at a local minimum.

- What happens if we shuffle the clusters? e.g. Switch the labels for clusters 1 and 2.
- We'll get the same likelihood. How many such equivalent settings are there?
- Assuming all clusters are distinct, there are  $k!$  equivalent solutions.

# Identifiability Issues for GMM

- Suppose we have found parameters

Cluster probabilities:  $\pi = (\pi_1, \dots, \pi_k)$

Cluster means:  $\mu = (\mu_1, \dots, \mu_k)$

Cluster covariance matrices:  $\Sigma = (\Sigma_1, \dots, \Sigma_k)$

that are at a local minimum.

- What happens if we shuffle the clusters? e.g. Switch the labels for clusters 1 and 2.
- We'll get the same likelihood. How many such equivalent settings are there?
- Assuming all clusters are distinct, there are  $k!$  equivalent solutions.
- Not a problem *per se*, but something to be aware of.

# Learning GMMs

How to learn the parameters  $\pi_k, \mu_k, \Sigma_k$ ?

# Learning GMMs

How to learn the parameters  $\pi_k, \mu_k, \Sigma_k$ ?

- MLE (also called maximize marginal likelihood).
- Log likelihood of data:

$$L(\theta) = \sum_{i=1}^n \log p(x_i; \theta) \quad (8)$$

$$= \sum_{i=1}^n \log \sum_z p(x, z; \theta) \quad (9)$$

# Learning GMMs

How to learn the parameters  $\pi_k, \mu_k, \Sigma_k$ ?

- MLE (also called maximize marginal likelihood).
- Log likelihood of data:

$$L(\theta) = \sum_{i=1}^n \log p(x_i; \theta) \quad (8)$$

$$= \sum_{i=1}^n \log \sum_z p(x, z; \theta) \quad (9)$$

- Cannot push log into the sum...  $z$  and  $x$  are coupled.
- No closed-form solution for GMM—try to compute the gradient yourself!

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = - \sum_{i=1}^n \log \left\{ \sum_{z=1}^k \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\}?$$



- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = - \sum_{i=1}^n \log \left\{ \sum_{z=1}^k \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\}?$$

- Can be done, in principle – but need to be clever about it.

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = - \sum_{i=1}^n \log \left\{ \sum_{z=1}^k \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\}?$$

- Can be done, in principle – but need to be clever about it.
- For example, each covariance matrix  $\Sigma_1, \dots, \Sigma_k$  has to be positive semidefinite.

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = - \sum_{i=1}^n \log \left\{ \sum_{z=1}^k \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\}?$$

- Can be done, in principle – but need to be clever about it.
- For example, each covariance matrix  $\Sigma_1, \dots, \Sigma_k$  has to be positive semidefinite.
- How to maintain that constraint?

# Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = - \sum_{i=1}^n \log \left\{ \sum_{z=1}^k \pi_z \mathcal{N}(x_i | \mu_z, \Sigma_z) \right\}?$$

- Can be done, in principle – but need to be clever about it.
- For example, each covariance matrix  $\Sigma_1, \dots, \Sigma_k$  has to be positive semidefinite.
- How to maintain that constraint?
  - Rewrite  $\Sigma_i = M_i M_i^T$ , where  $M_i$  is an unconstrained matrix.

# Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = - \sum_{i=1}^n \log \left\{ \sum_{z=1}^k \pi_z \mathcal{N}(x_i | \mu_z, \Sigma_z) \right\}?$$

- Can be done, in principle – but need to be clever about it.
- For example, each covariance matrix  $\Sigma_1, \dots, \Sigma_k$  has to be positive semidefinite.
- How to maintain that constraint?
  - Rewrite  $\Sigma_i = M_i M_i^T$ , where  $M_i$  is an unconstrained matrix.
  - Then  $\Sigma_i$  is positive semidefinite.

# Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = - \sum_{i=1}^n \log \left\{ \sum_{z=1}^k \pi_z \mathcal{N}(x_i | \mu_z, \Sigma_z) \right\}?$$

- Can be done, in principle – but need to be clever about it.
- For example, each covariance matrix  $\Sigma_1, \dots, \Sigma_k$  has to be positive semidefinite.
- How to maintain that constraint?
  - Rewrite  $\Sigma_i = M_i M_i^T$ , where  $M_i$  is an unconstrained matrix.
  - Then  $\Sigma_i$  is positive semidefinite.

## Learning GMMs: observable case

Suppose we observe cluster assignments  $z$ . Then MLE is easy:

$$n_z = \sum_{i=1}^n \mathbb{1}[z_i = z] \quad \# \text{ examples in each cluster} \quad (10)$$

(13)

## Learning GMMs: observable case

Suppose we observe cluster assignments  $z$ . Then MLE is easy:

$$n_z = \sum_{i=1}^n \mathbb{1}[z_i = z] \quad \# \text{ examples in each cluster} \quad (10)$$

$$\hat{\pi}(z) = \frac{n_z}{n} \quad \text{fraction of examples in each cluster} \quad (11)$$

(13)



# Learning GMMs: observable case

Suppose we observe cluster assignments  $z$ . Then MLE is easy:

$$n_z = \sum_{i=1}^n \mathbb{1}[z_i = z] \quad \# \text{ examples in each cluster} \quad (10)$$

$$\hat{\pi}(z) = \frac{n_z}{n} \quad \text{fraction of examples in each cluster} \quad (11)$$

$$\hat{\mu}_z = \frac{1}{n_z} \sum_{i: z_i = z} x_i \quad \text{empirical cluster mean} \quad (12)$$

(13)

## Learning GMMs: observable case

Suppose we observe cluster assignments  $z$ . Then MLE is easy:

$$n_z = \sum_{i=1}^n \mathbb{1}[z_i = z] \quad \# \text{ examples in each cluster} \quad (10)$$

$$\hat{\pi}(z) = \frac{n_z}{n} \quad \text{fraction of examples in each cluster} \quad (11)$$

$$\hat{\mu}_z = \frac{1}{n_z} \sum_{i: z_i = z} x_i \quad \text{empirical cluster mean} \quad (12)$$

$$\hat{\Sigma}_z = \frac{1}{n_z} \sum_{i: z_i = z} (x_i - \hat{\mu}_z)(x_i - \hat{\mu}_z)^T. \quad \text{empirical cluster covariance} \quad (13)$$

The inference problem: observe  $x$ , want to know  $z$ .

(16)

The inference problem: observe  $x$ , want to know  $z$ .

$$p(z = j | x_i) = p(x, z = j) / p(x) \quad (14)$$

(16)

The inference problem: observe  $x$ , want to know  $z$ .

$$p(z = j | x_i) = p(x, z = j) / p(x) \quad (14)$$

$$= \frac{p(x | z = j)p(z = j)}{\sum_k p(x | z = k)p(z = k)} \quad (15)$$

$$(16)$$

The inference problem: observe  $x$ , want to know  $z$ .

$$p(z = j | x_i) = p(x, z = j) / p(x) \quad (14)$$

$$= \frac{p(x | z = j)p(z = j)}{\sum_k p(x | z = k)p(z = k)} \quad (15)$$

$$= \frac{\pi_j \mathcal{N}(x_i | \mu_j, \Sigma_j)}{\sum_k \pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k)} \quad (16)$$

The inference problem: observe  $x$ , want to know  $z$ .

$$p(z = j | x_i) = p(x, z = j) / p(x) \quad (14)$$

$$= \frac{p(x | z = j)p(z = j)}{\sum_k p(x | z = k)p(z = k)} \quad (15)$$

$$= \frac{\pi_j \mathcal{N}(x_i | \mu_j, \Sigma_j)}{\sum_k \pi_k \mathcal{N}(x_i | \mu_k, \Sigma_k)} \quad (16)$$

- $p(z | x)$  is a *soft assignment*.
- If we know the parameters  $\mu, \Sigma, \pi$ , this would be easy to compute.

Let's compute the cluster assignments and the parameters iteratively.



Let's compute the cluster assignments and the parameters iteratively.

The expectation-minimization (EM) algorithm:

- 1 Initialize parameters  $\mu, \Sigma, \pi$  randomly.
- 2 Run until convergence:

Let's compute the cluster assignments and the parameters iteratively.

The expectation-minimization (EM) algorithm:

- ① Initialize parameters  $\mu, \Sigma, \pi$  randomly.
- ② Run until convergence:
  - ① E-step: fill in latent variables by inference.
    - compute soft assignments  $p(z | x_i)$  for all  $i$ .

Let's compute the cluster assignments and the parameters iteratively.

The expectation-minimization (EM) algorithm:

- ① Initialize parameters  $\mu, \Sigma, \pi$  randomly.
- ② Run until convergence:
  - ① E-step: fill in latent variables by inference.
    - compute soft assignments  $p(z | x_i)$  for all  $i$ .
  - ② M-step: standard MLE for  $\mu, \Sigma, \pi$  given “observed” variables.
    - Equivalent to MLE in the observable case on data weighted by  $p(z | x_i)$ .

## M-step for GMM

- Let  $p(z | x)$  be the soft assignments:

$$\gamma_i^j = \frac{\pi_j^{\text{old}} \mathcal{N}(x_i | \mu_j^{\text{old}}, \Sigma_j^{\text{old}})}{\sum_{c=1}^k \pi_c^{\text{old}} \mathcal{N}(x_i | \mu_c^{\text{old}}, \Sigma_c^{\text{old}})}.$$

- Exercise: show that

$$n_z = \sum_{i=1}^n \gamma_i^z$$

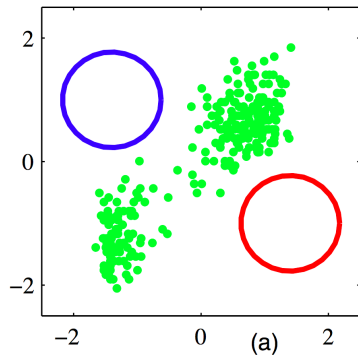
$$\mu_z^{\text{new}} = \frac{1}{n_z} \sum_{i=1}^n \gamma_i^z x_i$$

$$\Sigma_z^{\text{new}} = \frac{1}{n_z} \sum_{i=1}^n \gamma_i^z (x_i - \mu_z^{\text{new}}) (x_i - \mu_z^{\text{new}})^T$$

$$\pi_z^{\text{new}} = \frac{n_z}{n}.$$

# EM for GMM

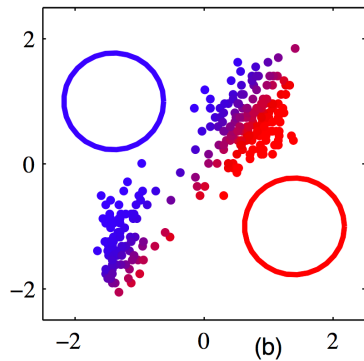
## • Initialization



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

# EM for GMM

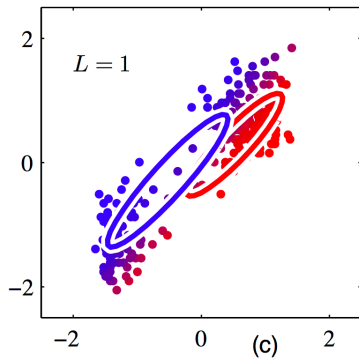
- First soft assignment:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

# EM for GMM

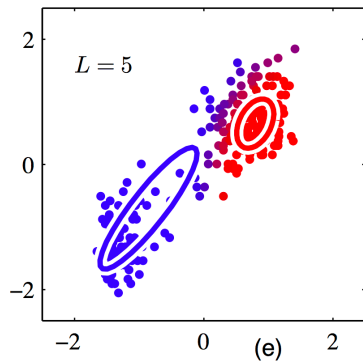
- First soft assignment:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

# EM for GMM

- After 5 rounds of EM:

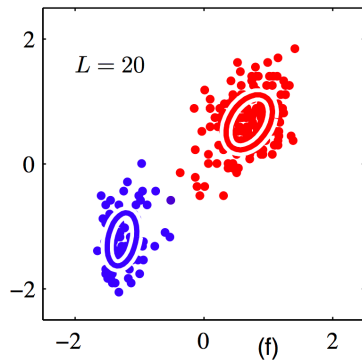


From Bishop's *Pattern recognition and machine learning*, Figure 9.8.



# EM for GMM

- After 20 rounds of EM:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

# EM for GMM: Summary

- EM is a general algorithm for learning latent variable models.
- *Key idea*: if data was fully observed, then MLE is easy.
  - E-step: fill in latent variables by computing  $p(z \mid x, \theta)$ .
  - M-step: standard MLE given fully observed data.
- Simpler and more efficient than gradient methods.
- Can prove that EM monotonically improves the likelihood and converges to a local minimum.
- $k$ -means is a special case of EM for GMM with *hard assignments*, also called hard-EM.

# Latent Variable Models

# General Latent Variable Model

- Two sets of random variables:  $z$  and  $x$ .
- $z$  consists of unobserved **hidden variables**.
- $x$  consists of **observed variables**.

# General Latent Variable Model

- Two sets of random variables:  $z$  and  $x$ .
- $z$  consists of unobserved **hidden variables**.
- $x$  consists of **observed variables**.
- Joint probability model parameterized by  $\theta \in \Theta$ :

$$p(x, z \mid \theta)$$

# General Latent Variable Model

- Two sets of random variables:  $z$  and  $x$ .
- $z$  consists of unobserved **hidden variables**.
- $x$  consists of **observed variables**.
- Joint probability model parameterized by  $\theta \in \Theta$ :

$$p(x, z \mid \theta)$$

## Definition

A **latent variable model** is a probability model for which certain variables are never observed.

# General Latent Variable Model

- Two sets of random variables:  $z$  and  $x$ .
- $z$  consists of unobserved **hidden variables**.
- $x$  consists of **observed variables**.
- Joint probability model parameterized by  $\theta \in \Theta$ :

$$p(x, z \mid \theta)$$

## Definition

A **latent variable model** is a probability model for which certain variables are never observed.

e.g. The Gaussian mixture model is a latent variable model.

# Complete and Incomplete Data

- Suppose we observe some data  $(x_1, \dots, x_n)$ .



# Complete and Incomplete Data

- Suppose we observe some data  $(x_1, \dots, x_n)$ .
- To simplify notation, take  $x$  to represent the entire dataset

$$x = (x_1, \dots, x_n),$$

and  $z$  to represent the corresponding unobserved variables

$$z = (z_1, \dots, z_n).$$

- An observation of  $x$  is called an **incomplete data set**.
- An observation  $(x, z)$  is called a **complete data set**.

# Our Objectives

- **Learning problem:** Given incomplete dataset  $x$ , find MLE

$$\hat{\theta} = \arg \max_{\theta} p(x | \theta).$$

# Our Objectives

- **Learning problem:** Given incomplete dataset  $x$ , find MLE

$$\hat{\theta} = \arg \max_{\theta} p(x | \theta).$$

- **Inference problem:** Given  $x$ , find conditional distribution over  $z$ :

$$p(z | x, \theta).$$

# Our Objectives

- **Learning problem:** Given incomplete dataset  $x$ , find MLE

$$\hat{\theta} = \arg \max_{\theta} p(x | \theta).$$

- **Inference problem:** Given  $x$ , find conditional distribution over  $z$ :

$$p(z | x, \theta).$$

- For Gaussian mixture model, learning is hard, inference is easy.
- For more complicated models, inference can also be hard. (See DSGA-1005)

# Log-Likelihood and Terminology

- Note that

$$\arg \max_{\theta} p(x | \theta) = \arg \max_{\theta} [\log p(x | \theta)] .$$

# Log-Likelihood and Terminology

- Note that

$$\arg \max_{\theta} p(x | \theta) = \arg \max_{\theta} [\log p(x | \theta)] .$$

- Often easier to work with this “**log-likelihood**”.

# Log-Likelihood and Terminology

- Note that

$$\arg \max_{\theta} p(x | \theta) = \arg \max_{\theta} [\log p(x | \theta)].$$

- Often easier to work with this “**log-likelihood**”.
- We often call  $p(x)$  the **marginal likelihood**,
  - because it is  $p(x, z)$  with  $z$  “marginalized out”:

$$p(x) = \sum_z p(x, z)$$

# Log-Likelihood and Terminology

- Note that

$$\arg \max_{\theta} p(x | \theta) = \arg \max_{\theta} [\log p(x | \theta)].$$

- Often easier to work with this “**log-likelihood**”.
- We often call  $p(x)$  the **marginal likelihood**,
  - because it is  $p(x, z)$  with  $z$  “marginalized out”:

$$p(x) = \sum_z p(x, z)$$

- We often call  $p(x, z)$  the **joint**. (for “joint distribution”)



# Log-Likelihood and Terminology

- Note that

$$\arg \max_{\theta} p(x | \theta) = \arg \max_{\theta} [\log p(x | \theta)].$$

- Often easier to work with this “**log-likelihood**”.
- We often call  $p(x)$  the **marginal likelihood**,
  - because it is  $p(x, z)$  with  $z$  “marginalized out”:

$$p(x) = \sum_z p(x, z)$$

- We often call  $p(x, z)$  the **joint**. (for “joint distribution”)
- Similarly,  $\log p(x)$  is the **marginal log-likelihood**.

# EM Algorithm

**Problem:** marginal log-likelihood  $\log p(x; \theta)$  is hard to optimize (observing only  $x$ )

# Intuition

**Problem:** marginal log-likelihood  $\log p(x; \theta)$  is hard to optimize (observing only  $x$ )

**Observation:** complete data log-likelihood  $\log p(x, z; \theta)$  is easy to optimize (observing both  $x$  and  $z$ )

# Intuition

**Problem:** marginal log-likelihood  $\log p(x; \theta)$  is hard to optimize (observing only  $x$ )

**Observation:** complete data log-likelihood  $\log p(x, z; \theta)$  is easy to optimize (observing both  $x$  and  $z$ )

**Idea:** guess a distribution of the latent variables  $q(z)$  (soft assignments)

# Intuition

**Problem:** marginal log-likelihood  $\log p(x; \theta)$  is hard to optimize (observing only  $x$ )

**Observation:** complete data log-likelihood  $\log p(x, z; \theta)$  is easy to optimize (observing both  $x$  and  $z$ )

**Idea:** guess a distribution of the latent variables  $q(z)$  (soft assignments)

Maximize the **expected complete data log-likelihood**:

$$\max_{\theta} \sum_{z \in \mathcal{Z}} q(z) \log p(x, z; \theta)$$

# Intuition

**Problem:** marginal log-likelihood  $\log p(x; \theta)$  is hard to optimize (observing only  $x$ )

**Observation:** complete data log-likelihood  $\log p(x, z; \theta)$  is easy to optimize (observing both  $x$  and  $z$ )

**Idea:** guess a distribution of the latent variables  $q(z)$  (soft assignments)

Maximize the **expected complete data log-likelihood**:

$$\max_{\theta} \sum_{z \in \mathcal{Z}} q(z) \log p(x, z; \theta)$$

**EM assumption:** the expected complete data log-likelihood is easy to optimize

Why should this work?

## Math Prerequisites



# Jensen's Inequality

## Theorem (Jensen's Inequality)

If  $f : \mathbb{R} \rightarrow \mathbb{R}$  is a **convex** function, and  $x$  is a random variable, then

$$\mathbb{E}f(x) \geq f(\mathbb{E}x).$$

# Jensen's Inequality

## Theorem (Jensen's Inequality)

If  $f : \mathbb{R} \rightarrow \mathbb{R}$  is a **convex** function, and  $x$  is a random variable, then

$$\mathbb{E}f(x) \geq f(\mathbb{E}x).$$

Moreover, if  $f$  is **strictly convex**, then equality implies that  $x = \mathbb{E}x$  with probability 1 (i.e.  $x$  is a constant).

# Jensen's Inequality

## Theorem (Jensen's Inequality)

If  $f : \mathbb{R} \rightarrow \mathbb{R}$  is a **convex** function, and  $x$  is a random variable, then

$$\mathbb{E}f(x) \geq f(\mathbb{E}x).$$

Moreover, if  $f$  is **strictly convex**, then equality implies that  $x = \mathbb{E}x$  with probability 1 (i.e.  $x$  is a constant).

- e.g.  $f(x) = x^2$  is convex. So  $\mathbb{E}x^2 \geq (\mathbb{E}x)^2$ . Thus

$$\text{Var}(x) = \mathbb{E}x^2 - (\mathbb{E}x)^2 \geq 0.$$

# Kullback-Leibler Divergence

- Let  $p(x)$  and  $q(x)$  be probability mass functions (PMFs) on  $\mathcal{X}$ .
- How can we measure how “different”  $p$  and  $q$  are?

# Kullback-Leibler Divergence

- Let  $p(x)$  and  $q(x)$  be probability mass functions (PMFs) on  $\mathcal{X}$ .
- How can we measure how “different”  $p$  and  $q$  are?
- The **Kullback-Leibler** or “**KL**” **Divergence** is defined by

$$\text{KL}(p\|q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}.$$

(Assumes  $q(x) = 0$  implies  $p(x) = 0$ .)

# Kullback-Leibler Divergence

- Let  $p(x)$  and  $q(x)$  be probability mass functions (PMFs) on  $\mathcal{X}$ .
- How can we measure how “different”  $p$  and  $q$  are?
- The **Kullback-Leibler** or “**KL**” **Divergence** is defined by

$$\text{KL}(p\|q) = \sum_{x \in \mathcal{X}} p(x) \log \frac{p(x)}{q(x)}.$$

(Assumes  $q(x) = 0$  implies  $p(x) = 0$ .)

- Can also write this as

$$\text{KL}(p\|q) = \mathbb{E}_{x \sim p} \log \frac{p(x)}{q(x)}.$$

## Gibbs Inequality ( $\text{KL}(p\|q) \geq 0$ and $\text{KL}(p\|p) = 0$ )

### Theorem (Gibbs Inequality)

Let  $p(x)$  and  $q(x)$  be PMFs on  $\mathcal{X}$ . Then

$$\text{KL}(p\|q) \geq 0,$$

with equality iff  $p(x) = q(x)$  for all  $x \in \mathcal{X}$ .

## Gibbs Inequality ( $\text{KL}(p\|q) \geq 0$ and $\text{KL}(p\|p) = 0$ )

### Theorem (Gibbs Inequality)

Let  $p(x)$  and  $q(x)$  be PMFs on  $\mathcal{X}$ . Then

$$\text{KL}(p\|q) \geq 0,$$

with equality iff  $p(x) = q(x)$  for all  $x \in \mathcal{X}$ .

- KL divergence measures the “distance” between distributions.



# Gibbs Inequality ( $\text{KL}(p\|q) \geq 0$ and $\text{KL}(p\|p) = 0$ )

## Theorem (Gibbs Inequality)

Let  $p(x)$  and  $q(x)$  be PMFs on  $\mathcal{X}$ . Then

$$\text{KL}(p\|q) \geq 0,$$

with equality iff  $p(x) = q(x)$  for all  $x \in \mathcal{X}$ .

- KL divergence measures the “distance” between distributions.
- Note:
  - KL divergence **not a metric**.
  - KL divergence is **not symmetric**.

## Gibbs Inequality: Proof

$$\text{KL}(p\|q) = \mathbb{E}_p \left[ -\log \left( \frac{q(x)}{p(x)} \right) \right]$$

## Gibbs Inequality: Proof

$$\begin{aligned}\mathrm{KL}(p\|q) &= \mathbb{E}_p \left[ -\log \left( \frac{q(x)}{p(x)} \right) \right] \\ &\geq -\log \left[ \mathbb{E}_p \left( \frac{q(x)}{p(x)} \right) \right] \quad (\text{Jensen's})\end{aligned}$$

## Gibbs Inequality: Proof

$$\begin{aligned}\text{KL}(p\|q) &= \mathbb{E}_p \left[ -\log \left( \frac{q(x)}{p(x)} \right) \right] \\ &\geq -\log \left[ \mathbb{E}_p \left( \frac{q(x)}{p(x)} \right) \right] \quad (\text{Jensen's}) \\ &= -\log \left[ \sum_{\{x|p(x)>0\}} p(x) \frac{q(x)}{p(x)} \right]\end{aligned}$$

## Gibbs Inequality: Proof

$$\begin{aligned}\text{KL}(p\|q) &= \mathbb{E}_p \left[ -\log \left( \frac{q(x)}{p(x)} \right) \right] \\ &\geq -\log \left[ \mathbb{E}_p \left( \frac{q(x)}{p(x)} \right) \right] \quad (\text{Jensen's}) \\ &= -\log \left[ \sum_{\{x|p(x)>0\}} p(x) \frac{q(x)}{p(x)} \right] \\ &= -\log \left[ \sum_{x \in \mathcal{X}} q(x) \right]\end{aligned}$$

## Gibbs Inequality: Proof

$$\begin{aligned}\text{KL}(p\|q) &= \mathbb{E}_p \left[ -\log \left( \frac{q(x)}{p(x)} \right) \right] \\ &\geq -\log \left[ \mathbb{E}_p \left( \frac{q(x)}{p(x)} \right) \right] \quad (\text{Jensen's}) \\ &= -\log \left[ \sum_{\{x|p(x)>0\}} p(x) \frac{q(x)}{p(x)} \right] \\ &= -\log \left[ \sum_{x \in \mathcal{X}} q(x) \right] \\ &= -\log 1 = 0.\end{aligned}$$

## Gibbs Inequality: Proof

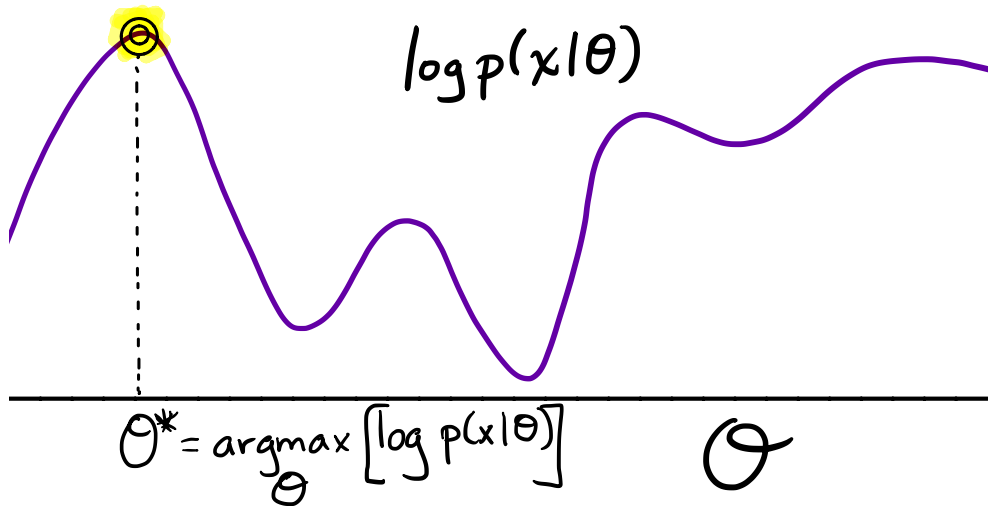
$$\begin{aligned}\mathrm{KL}(p\|q) &= \mathbb{E}_p \left[ -\log \left( \frac{q(x)}{p(x)} \right) \right] \\ &\geq -\log \left[ \mathbb{E}_p \left( \frac{q(x)}{p(x)} \right) \right] \quad (\text{Jensen's}) \\ &= -\log \left[ \sum_{\{x|p(x)>0\}} p(x) \frac{q(x)}{p(x)} \right] \\ &= -\log \left[ \sum_{x \in \mathcal{X}} q(x) \right] \\ &= -\log 1 = 0.\end{aligned}$$

- Since  $-\log$  is strictly convex, we have strict equality iff  $q(x)/p(x)$  is a constant, which implies  $q = p$ .

## The ELBO: Family of Lower Bounds on $\log p(x | \theta)$



# The Maximum Likelihood Estimator



## Lower bound of the marginal log-likelihood

$$\log p(x; \theta) = \log \sum_{z \in \mathcal{Z}} p(x, z; \theta)$$

## Lower bound of the marginal log-likelihood

$$\begin{aligned}\log p(x; \theta) &= \log \sum_{z \in \mathcal{Z}} p(x, z; \theta) \\ &= \log \sum_{z \in \mathcal{Z}} q(z) \frac{p(x, z; \theta)}{q(z)}\end{aligned}$$

## Lower bound of the marginal log-likelihood

$$\begin{aligned}\log p(x; \theta) &= \log \sum_{z \in \mathcal{Z}} p(x, z; \theta) \\ &= \log \sum_{z \in \mathcal{Z}} q(z) \frac{p(x, z; \theta)}{q(z)} \\ &\geq \sum_{z \in \mathcal{Z}} q(z) \log \frac{p(x, z; \theta)}{q(z)}\end{aligned}$$

## Lower bound of the marginal log-likelihood

$$\begin{aligned}\log p(x; \theta) &= \log \sum_{z \in \mathcal{Z}} p(x, z; \theta) \\ &= \log \sum_{z \in \mathcal{Z}} q(z) \frac{p(x, z; \theta)}{q(z)} \\ &\geq \sum_{z \in \mathcal{Z}} q(z) \log \frac{p(x, z; \theta)}{q(z)} \\ &\stackrel{\text{def}}{=} \mathcal{L}(q, \theta)\end{aligned}$$

- **Evidence:**  $\log p(x; \theta)$
- **Evidence lower bound (ELBO):**  $\mathcal{L}(q, \theta)$
- $q$ : chosen to be a family of tractable distributions
- Idea: *maximize the ELBO* instead of  $\log p(x; \theta)$

- The MLE is defined as a maximum over  $\theta$ :

$$\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} [\log p(x | \theta)].$$

# MLE, EM, and the ELBO

- The MLE is defined as a maximum over  $\theta$ :

$$\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} [\log p(x | \theta)].$$

- For any PMF  $q(z)$ , we have a lower bound on the marginal log-likelihood

$$\log p(x | \theta) \geq \mathcal{L}(q, \theta).$$

# MLE, EM, and the ELBO

- The MLE is defined as a maximum over  $\theta$ :

$$\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} [\log p(x | \theta)].$$

- For any PMF  $q(z)$ , we have a lower bound on the marginal log-likelihood

$$\log p(x | \theta) \geq \mathcal{L}(q, \theta).$$

- In EM algorithm, we maximize the lower bound (ELBO) over  $\theta$  and  $q$ :

$$\hat{\theta}_{\text{EM}} \approx \arg \max_{\theta} \left[ \max_q \mathcal{L}(q, \theta) \right]$$



# MLE, EM, and the ELBO

- The MLE is defined as a maximum over  $\theta$ :

$$\hat{\theta}_{\text{MLE}} = \arg \max_{\theta} [\log p(x | \theta)].$$

- For any PMF  $q(z)$ , we have a lower bound on the marginal log-likelihood

$$\log p(x | \theta) \geq \mathcal{L}(q, \theta).$$

- In EM algorithm, we maximize the lower bound (ELBO) over  $\theta$  and  $q$ :

$$\hat{\theta}_{\text{EM}} \approx \arg \max_{\theta} \left[ \max_q \mathcal{L}(q, \theta) \right]$$

- In EM algorithm,  $q$  ranges over all distributions on  $z$ .

## EM: Coordinate Ascent on Lower Bound

- Choose sequence of  $q$ 's and  $\theta$ 's by “**coordinate ascent**” on  $\mathcal{L}(q, \theta)$ .

## EM: Coordinate Ascent on Lower Bound

- Choose sequence of  $q$ 's and  $\theta$ 's by “**coordinate ascent**” on  $\mathcal{L}(q, \theta)$ .
- EM Algorithm (high level):
  - 1 Choose initial  $\theta^{\text{old}}$ .
  - 2 Let  $q^* = \arg \max_q \mathcal{L}(q, \theta^{\text{old}})$

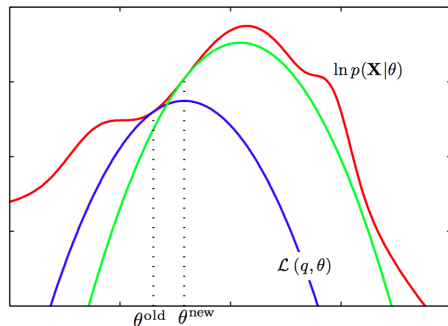
# EM: Coordinate Ascent on Lower Bound

- Choose sequence of  $q$ 's and  $\theta$ 's by “**coordinate ascent**” on  $\mathcal{L}(q, \theta)$ .
- EM Algorithm (high level):
  - ① Choose initial  $\theta^{\text{old}}$ .
  - ② Let  $q^* = \arg \max_q \mathcal{L}(q, \theta^{\text{old}})$
  - ③ Let  $\theta^{\text{new}} = \arg \max_{\theta} \mathcal{L}(q^*, \theta)$ .

## EM: Coordinate Ascent on Lower Bound

- Choose sequence of  $q$ 's and  $\theta$ 's by “**coordinate ascent**” on  $\mathcal{L}(q, \theta)$ .
- EM Algorithm (high level):
  - ① Choose initial  $\theta^{\text{old}}$ .
  - ② Let  $q^* = \arg \max_q \mathcal{L}(q, \theta^{\text{old}})$
  - ③ Let  $\theta^{\text{new}} = \arg \max_{\theta} \mathcal{L}(q^*, \theta)$ .
  - ④ Go to step 2, until converged.
- Will show:  $p(x | \theta^{\text{new}}) \geq p(x | \theta^{\text{old}})$
- **Get sequence of  $\theta$ 's with monotonically increasing likelihood.**

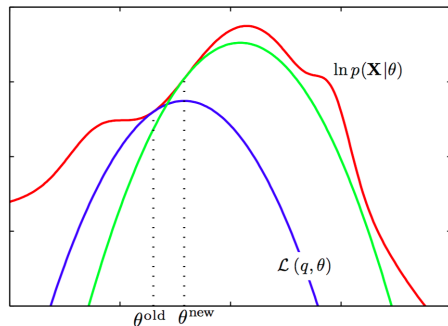
## EM: Coordinate Ascent on Lower Bound



- 1 Start at  $\theta^{\text{old}}$ .

From Bishop's *Pattern recognition and machine learning*, Figure 9.14.

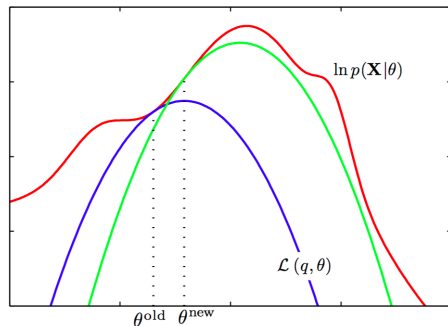
## EM: Coordinate Ascent on Lower Bound



- 1 Start at  $\theta^{\text{old}}$ .
- 2 Find  $q$  giving best lower bound at  $\theta^{\text{old}} \Rightarrow \mathcal{L}(q, \theta)$ .

From Bishop's *Pattern recognition and machine learning*, Figure 9.14.

## EM: Coordinate Ascent on Lower Bound



- 1 Start at  $\theta^{\text{old}}$ .
- 2 Find  $q$  giving best lower bound at  $\theta^{\text{old}} \implies \mathcal{L}(q, \theta)$ .
- 3  $\theta^{\text{new}} = \arg \max_{\theta} \mathcal{L}(q, \theta)$ .

From Bishop's *Pattern recognition and machine learning*, Figure 9.14.



## Is ELBO a "good" lowerbound?

$$\begin{aligned}\mathcal{L}(q, \theta) &= \sum_{z \in \mathcal{Z}} q(z) \log \frac{p(x, z | \theta)}{q(z)} \\&= \sum_{z \in \mathcal{Z}} q(z) \log \frac{p(z | x, \theta) p(x | \theta)}{q(z)} \\&= - \sum_{z \in \mathcal{Z}} q(z) \log \frac{q(z)}{p(z | x, \theta)} + \sum_{z \in \mathcal{Z}} q(z) \log p(x | \theta) \\&= -\text{KL}(q(z) \| p(z | x, \theta)) + \underbrace{\log p(x | \theta)}_{\text{evidence}}\end{aligned}$$

- **KL divergence:** measures "distance" between two distributions (not symmetric!)
- $\text{KL}(q \| p) \geq 0$  with equality iff  $q(z) = p(z | x)$ .
- $\text{ELBO} = \text{evidence} - \text{KL} \leq \text{evidence}$

## Maximizing over $q$ for fixed $\theta$ .

- Find  $q$  maximizing

$$\mathcal{L}(q, \theta) = -\text{KL}[q(z), p(z | x, \theta)] + \log p(x | \theta)$$

## Maximizing over $q$ for fixed $\theta$ .

- Find  $q$  maximizing

$$\mathcal{L}(q, \theta) = -\text{KL}[q(z), p(z | x, \theta)] + \underbrace{\log p(x | \theta)}_{\text{no } q \text{ here}}$$

## Maximizing over $q$ for fixed $\theta$ .

- Find  $q$  maximizing

$$\mathcal{L}(q, \theta) = -\text{KL}[q(z), p(z | x, \theta)] + \underbrace{\log p(x | \theta)}_{\text{no } q \text{ here}}$$

- Recall  $\text{KL}(p \| q) \geq 0$ , and  $\text{KL}(p \| p) = 0$ .

## Maximizing over $q$ for fixed $\theta$ .

- Find  $q$  maximizing

$$\mathcal{L}(q, \theta) = -\text{KL}[q(z), p(z | x, \theta)] + \underbrace{\log p(x | \theta)}_{\text{no } q \text{ here}}$$

- Recall  $\text{KL}(p||q) \geq 0$ , and  $\text{KL}(p||p) = 0$ .
- Best  $q$  is  $q^*(z) = p(z | x, \theta)$  and

## Maximizing over $q$ for fixed $\theta$ .

- Find  $q$  maximizing

$$\mathcal{L}(q, \theta) = -\text{KL}[q(z), p(z | x, \theta)] + \underbrace{\log p(x | \theta)}_{\text{no } q \text{ here}}$$

- Recall  $\text{KL}(p \| q) \geq 0$ , and  $\text{KL}(p \| p) = 0$ .

- Best  $q$  is  $q^*(z) = p(z | x, \theta)$  and

$$\mathcal{L}(q^*, \theta) = -\underbrace{\text{KL}[p(z | x, \theta), p(z | x, \theta)]}_{=0} + \log p(x | \theta)$$

## Maximizing over $q$ for fixed $\theta$ .

- Find  $q$  maximizing

$$\mathcal{L}(q, \theta) = -\text{KL}[q(z), p(z | x, \theta)] + \underbrace{\log p(x | \theta)}_{\text{no } q \text{ here}}$$

- Recall  $\text{KL}(p \| q) \geq 0$ , and  $\text{KL}(p \| p) = 0$ .

- Best  $q$  is  $q^*(z) = p(z | x, \theta)$  and

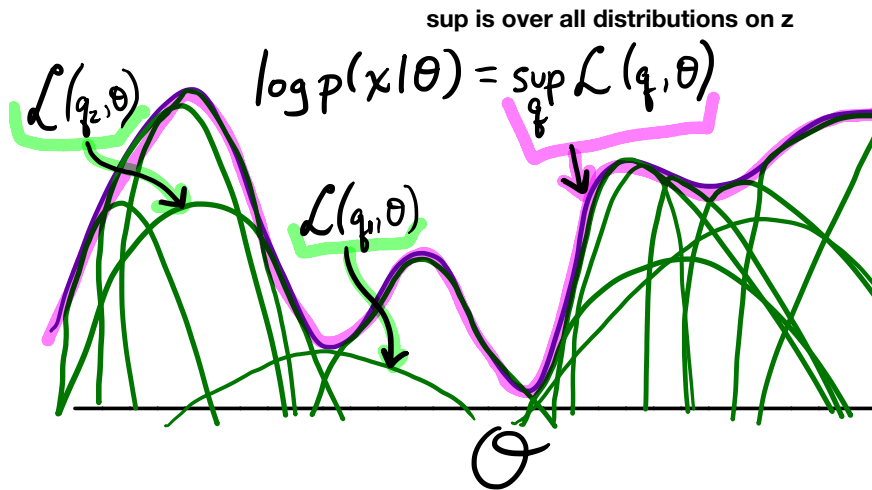
$$\mathcal{L}(q^*, \theta) = -\underbrace{\text{KL}[p(z | x, \theta), p(z | x, \theta)]}_{=0} + \log p(x | \theta)$$

- Summary:

$$\log p(x | \theta) = \sup_q \mathcal{L}(q, \theta) \quad \forall \theta$$

- For any  $\theta$ , **sup is attained** at  $q(z) = p(z | x, \theta)$ .

# Marginal Log-Likelihood **IS** the Supremum over Lower Bounds





# Summary

**Latent variable models:** clustering, latent structure, missing labels etc.

**Parameter estimation:** maximum marginal log-likelihood

**Challenge:** directly maximize the **evidence**  $\log p(x; \theta)$  is hard

**Solution:** maximize the **evidence lower bound**:

$$\text{ELBO} = \mathcal{L}(q, \theta) = -\text{KL}(q(z) \| p(z | x; \theta)) + \log p(x; \theta)$$

Why does it work?

$$q^*(z) = p(z | x; \theta) \quad \forall \theta \in \Theta$$
$$\mathcal{L}(q^*, \theta^*) = \max_{\theta} \log p(x; \theta)$$

# EM algorithm

*Coordinate ascent on  $\mathcal{L}(q, \theta)$*

- 1 Random initialization:  $\theta^{\text{old}} \leftarrow \theta_0$
- 2 Repeat until convergence
  - i  $q(z) \leftarrow \arg \max_q \mathcal{L}(q, \theta^{\text{old}})$

**Expectation** (the E-step):  $q^*(z) = p(z | x; \theta^{\text{old}})$   
 $J(\theta) = \mathcal{L}(q^*, \theta)$

ii  $\theta^{\text{new}} \leftarrow \arg \max_{\theta} \mathcal{L}(q^*, \theta)$

**Maximization** (the M-step):  $\theta^{\text{new}} \leftarrow \arg \max_{\theta} J(\theta)$

## ① Expectation Step

- Let  $q^*(z) = p(z \mid x, \theta^{\text{old}})$ . [ $q^*$  gives best lower bound at  $\theta^{\text{old}}$ ]

## ① Expectation Step

- Let  $q^*(z) = p(z | x, \theta^{\text{old}})$ . [ $q^*$  gives best lower bound at  $\theta^{\text{old}}$ ]
- Let

$$J(\theta) := \mathcal{L}(q^*, \theta) = \underbrace{\sum_z q^*(z) \log \left( \frac{p(x, z | \theta)}{q^*(z)} \right)}_{\text{expectation w.r.t. } z \sim q^*(z)}$$

## ① Expectation Step

- Let  $q^*(z) = p(z | x, \theta^{\text{old}})$ . [ $q^*$  gives best lower bound at  $\theta^{\text{old}}$ ]
- Let

$$J(\theta) := \mathcal{L}(q^*, \theta) = \underbrace{\sum_z q^*(z) \log \left( \frac{p(x, z | \theta)}{q^*(z)} \right)}_{\text{expectation w.r.t. } z \sim q^*(z)}$$

## ② Maximization Step

$$\theta^{\text{new}} = \arg \max_{\theta} J(\theta).$$

## 1 Expectation Step

- Let  $q^*(z) = p(z | x, \theta^{\text{old}})$ . [ $q^*$  gives best lower bound at  $\theta^{\text{old}}$ ]
- Let

$$J(\theta) := \mathcal{L}(q^*, \theta) = \underbrace{\sum_z q^*(z) \log \left( \frac{p(x, z | \theta)}{q^*(z)} \right)}_{\text{expectation w.r.t. } z \sim q^*(z)}$$

## 2 Maximization Step

$$\theta^{\text{new}} = \arg \max_{\theta} J(\theta).$$

[Equivalent to maximizing expected complete log-likelihood.]

## ① Expectation Step

- Let  $q^*(z) = p(z | x, \theta^{\text{old}})$ . [ $q^*$  gives best lower bound at  $\theta^{\text{old}}$ ]
- Let

$$J(\theta) := \mathcal{L}(q^*, \theta) = \underbrace{\sum_z q^*(z) \log \left( \frac{p(x, z | \theta)}{q^*(z)} \right)}_{\text{expectation w.r.t. } z \sim q^*(z)}$$

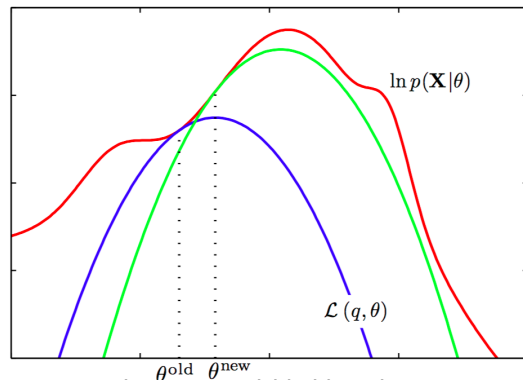
## ② Maximization Step

$$\theta^{\text{new}} = \arg \max_{\theta} J(\theta).$$

[Equivalent to maximizing expected complete log-likelihood.]

EM puts no constraint on  $q$  in the E-step and assumes the M-step is easy. In general, both steps can be hard.

## Monotonically increasing likelihood



Exercise: prove that EM increases the marginal likelihood monotonically

$$\log p(x; \theta^{\text{new}}) \geq \log p(x; \theta^{\text{old}}).$$

Does EM converge to a global maximum?



## Variations on EM

# EM Gives Us Two New Problems

- The “E” Step: Computing

$$J(\theta) := \mathcal{L}(q^*, \theta) = \sum_z q^*(z) \log \left( \frac{p(x, z | \theta)}{q^*(z)} \right)$$

# EM Gives Us Two New Problems

- The “E” Step: Computing

$$J(\theta) := \mathcal{L}(q^*, \theta) = \sum_z q^*(z) \log \left( \frac{p(x, z | \theta)}{q^*(z)} \right)$$

- The “M” Step: Computing

$$\theta^{\text{new}} = \arg \max_{\theta} J(\theta).$$

# EM Gives Us Two New Problems

- The “E” Step: Computing

$$J(\theta) := \mathcal{L}(q^*, \theta) = \sum_z q^*(z) \log \left( \frac{p(x, z | \theta)}{q^*(z)} \right)$$

- The “M” Step: Computing

$$\theta^{\text{new}} = \arg \max_{\theta} J(\theta).$$

# EM Gives Us Two New Problems

- The “E” Step: Computing

$$J(\theta) := \mathcal{L}(q^*, \theta) = \sum_z q^*(z) \log \left( \frac{p(x, z | \theta)}{q^*(z)} \right)$$

- The “M” Step: Computing

$$\theta^{\text{new}} = \arg \max_{\theta} J(\theta).$$

- Either of these can be too hard to do in practice.

# Generalized EM (GEM)

- Addresses the problem of a difficult “M” step.

# Generalized EM (GEM)

- Addresses the problem of a difficult “M” step.
- Rather than finding

$$\theta^{\text{new}} = \arg \max_{\theta} J(\theta),$$

find **any**  $\theta^{\text{new}}$  for which

$$J(\theta^{\text{new}}) > J(\theta^{\text{old}}).$$

# Generalized EM (GEM)

- Addresses the problem of a difficult “M” step.
- Rather than finding

$$\theta^{\text{new}} = \arg \max_{\theta} J(\theta),$$

find **any**  $\theta^{\text{new}}$  for which

$$J(\theta^{\text{new}}) > J(\theta^{\text{old}}).$$

- Can use a standard nonlinear optimization strategy
  - e.g. take a gradient step on  $J$ .



# Generalized EM (GEM)

- Addresses the problem of a difficult “M” step.
- Rather than finding

$$\theta^{\text{new}} = \arg \max_{\theta} J(\theta),$$

find **any**  $\theta^{\text{new}}$  for which

$$J(\theta^{\text{new}}) > J(\theta^{\text{old}}).$$

- Can use a standard nonlinear optimization strategy
  - e.g. take a gradient step on  $J$ .
- We still get monotonically increasing likelihood.

# EM and More General Variational Methods

- Suppose “E” step is difficult:
  - Hard to take expectation w.r.t.  $q^*(z) = p(z \mid x, \theta^{\text{old}})$ .

# EM and More General Variational Methods

- Suppose “E” step is difficult:
  - Hard to take expectation w.r.t.  $q^*(z) = p(z \mid x, \theta^{\text{old}})$ .
- Solution: Restrict to distributions  $\mathcal{Q}$  that are easy to work with.

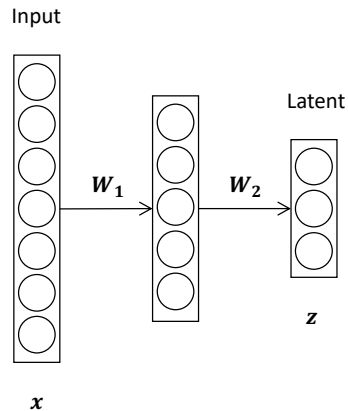
# EM and More General Variational Methods

- Suppose “E” step is difficult:
  - Hard to take expectation w.r.t.  $q^*(z) = p(z | x, \theta^{\text{old}})$ .
- Solution: Restrict to distributions  $\mathcal{Q}$  that are easy to work with.
- Lower bound now looser:

$$q^* = \arg \min_{q \in \mathcal{Q}} \text{KL}[q(z), p(z | x, \theta^{\text{old}})]$$

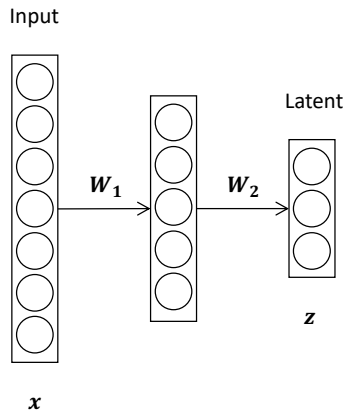
# Deep Latent Variable Models

- Neural network is a flexible function class to represent transformation between random variables e.g.,  $q(z)$ .



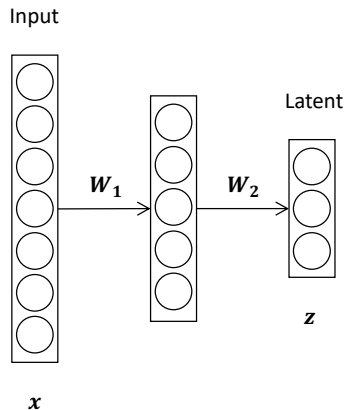
# Deep Latent Variable Models

- Neural network is a flexible function class to represent transformation between random variables e.g.,  $q(z)$ .
- In neural networks, the hidden activations do not have probabilistic interpretation as they are not random variables.



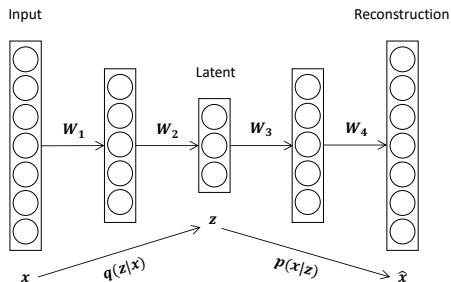
# Deep Latent Variable Models

- Neural network is a flexible function class to represent transformation between random variables e.g.,  $q(z)$ .
- In neural networks, the hidden activations do not have probabilistic interpretation as they are not random variables.
- What if we let the hidden represent some learned latent code?



# Variational Autoencoders (VAE) <sup>1</sup>

- An autoencoder (AE) is a neural network that reconstructs the same input.

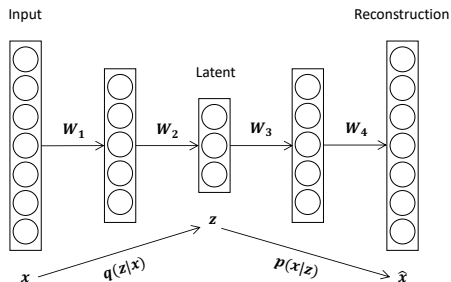


<sup>1</sup>Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014.



# Variational Autoencoders (VAE) <sup>1</sup>

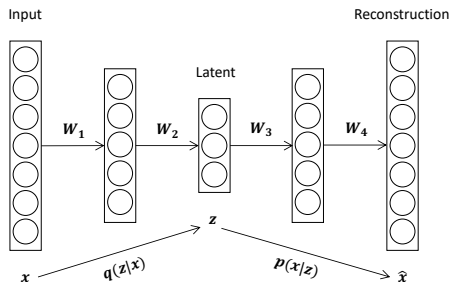
- An autoencoder (AE) is a neural network that reconstructs the same input.
- The first half is an encoder, from input to latent. The second half is a decoder.



<sup>1</sup>Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014.

# Variational Autoencoders (VAE) <sup>1</sup>

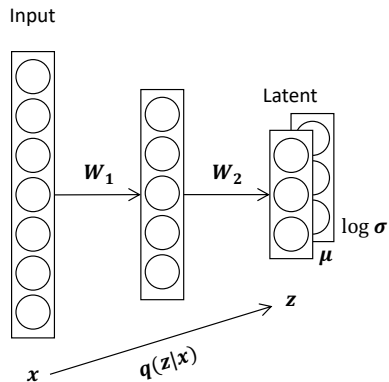
- An autoencoder (AE) is a neural network that reconstructs the same input.
- The first half is an encoder, from input to latent. The second half is a decoder.
- How to make  $q$  a probability distribution?



<sup>1</sup>Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014.

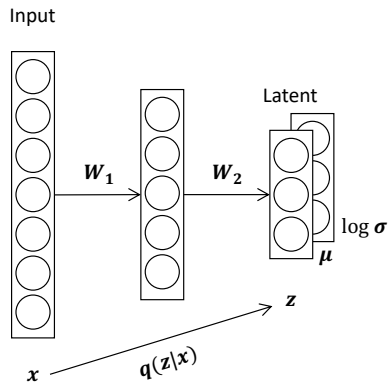
# Reparameterization Trick

- Let's assume that  $q(z|x)$  is a Gaussian distribution.



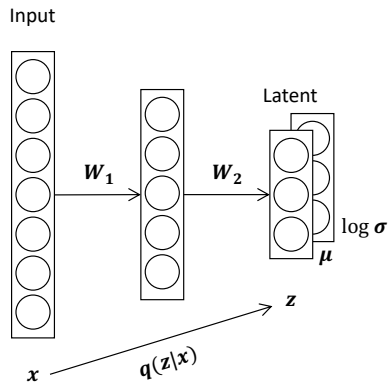
# Reparameterization Trick

- Let's assume that  $q(z|x)$  is a Gaussian distribution.
- Instead of letting the neural network to output a stochastic variable, we can let it predict deterministically the distribution parameters  $\mu$  and  $\sigma$ .



# Reparameterization Trick

- Let's assume that  $q(z|x)$  is a Gaussian distribution.
- Instead of letting the neural network to output a stochastic variable, we can let it predict deterministically the distribution parameters  $\mu$  and  $\sigma$ .
- A stochastic  $z$  can be sampled from  $\mathcal{N}(\mu, \sigma^2)$ :  $z = \mu + \sigma \cdot \epsilon$ , where  $\epsilon \sim \mathcal{N}(0, 1)$ .



# Variational Lower Bound

- Encoder  $q$  weights:  $\phi$ ; Decoder  $p$  weights:  $\theta$ .

# Variational Lower Bound

- Encoder  $q$  weights:  $\phi$ ; Decoder  $p$  weights:  $\theta$ .
- Now maximize ELBO:

(20)

# Variational Lower Bound

- Encoder  $q$  weights:  $\phi$ ; Decoder  $p$  weights:  $\theta$ .
- Now maximize ELBO:

$$L(q; \phi, \theta) = \sum_z q(z) \frac{p_{\theta}(x, z)}{q_{\phi}(z|x)} \quad (17)$$

(20)



# Variational Lower Bound

- Encoder  $q$  weights:  $\phi$ ; Decoder  $p$  weights:  $\theta$ .
- Now maximize ELBO:

$$L(q; \phi, \theta) = \sum_z q(z) \frac{p_\theta(x, z)}{q_\phi(z|x)} \quad (17)$$

$$= \mathbb{E}_{z \sim q} [-\log q_\phi(z|x) + \log p_\theta(x, z)] \quad (18)$$

$$(20)$$

# Variational Lower Bound

- Encoder  $q$  weights:  $\phi$ ; Decoder  $p$  weights:  $\theta$ .
- Now maximize ELBO:

$$L(q; \phi, \theta) = \sum_z q(z) \frac{p_\theta(x, z)}{q_\phi(z|x)} \quad (17)$$

$$= \mathbb{E}_{z \sim q} [-\log q_\phi(z|x) + \log p_\theta(x, z)] \quad (18)$$

$$= \mathbb{E}_{z \sim q} [-\log q_\phi(z|x) + \log p_\theta(x|z) + \log p_\theta(z)] \quad (19)$$

$$(20)$$

# Variational Lower Bound

- Encoder  $q$  weights:  $\phi$ ; Decoder  $p$  weights:  $\theta$ .
- Now maximize ELBO:

$$L(q; \phi, \theta) = \sum_z q(z) \frac{p_\theta(x, z)}{q_\phi(z|x)} \quad (17)$$

$$= \mathbb{E}_{z \sim q} [-\log q_\phi(z|x) + \log p_\theta(x, z)] \quad (18)$$

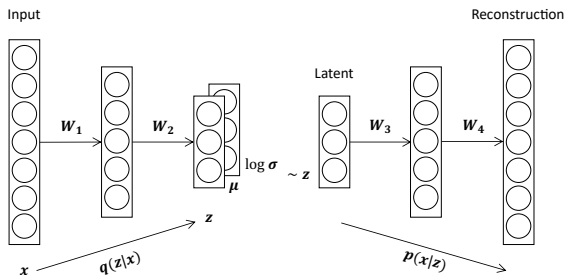
$$= \mathbb{E}_{z \sim q} [-\log q_\phi(z|x) + \log p_\theta(x|z) + \log p_\theta(z)] \quad (19)$$

$$= \underbrace{-KL(q_\phi(z|x) || p_\theta(z))}_{\text{Divergence between } q \text{ and the prior distribution}} + \underbrace{\mathbb{E}_{z \sim q}(\log p_\theta(x|z))}_{\text{Reconstruction based on } z} \quad (20)$$

# Stochastic Gradient

- The loss function needs to take expectation over  $q$ :

$$L(q; \phi, \theta) = -KL(q_\phi(z|x) || p_\theta(z)) + \mathbb{E}_{z \sim q}(\log p_\theta(x|z))$$

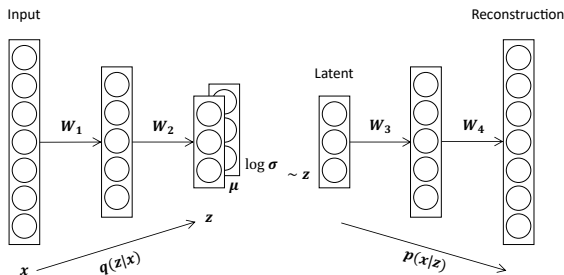


# Stochastic Gradient

- The loss function needs to take expectation over  $q$ :

$$L(q; \phi, \theta) = -KL(q_\phi(z|x) || p_\theta(z)) + \mathbb{E}_{z \sim q}(\log p_\theta(x|z))$$

- Turns out we just need to have a Monte Carlo sample size of 1:
  - For each  $x$ , sample one  $z$  from  $q(z|x)$ .

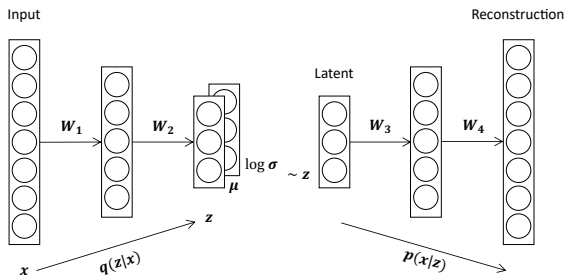


# Stochastic Gradient

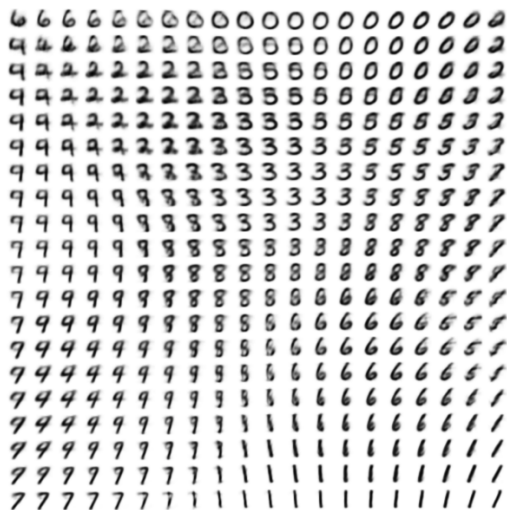
- The loss function needs to take expectation over  $q$ :

$$L(q; \phi, \theta) = -KL(q_\phi(z|x) || p_\theta(z)) + \mathbb{E}_{z \sim q}(\log p_\theta(x|z))$$

- Turns out we just need to have a Monte Carlo sample size of 1:
  - For each  $x$ , sample one  $z$  from  $q(z|x)$ .
- Backprop through reparameterization.



# Learned Manifold



# Today's Summary

- Motivation: Unsupervised learning



# Today's Summary

- Motivation: Unsupervised learning
- K-means: A simple algorithm for discovering clusters

# Today's Summary

- Motivation: Unsupervised learning
- K-means: A simple algorithm for discovering clusters
- Making k-means probabilistic: Gaussian mixture models

# Today's Summary

- Motivation: Unsupervised learning
- K-means: A simple algorithm for discovering clusters
- Making k-means probabilistic: Gaussian mixture models
- More generally: Latent variable models

# Today's Summary

- Motivation: Unsupervised learning
- K-means: A simple algorithm for discovering clusters
- Making k-means probabilistic: Gaussian mixture models
- More generally: Latent variable models
- Learning of latent variable models: EM

# Today's Summary

- Motivation: Unsupervised learning
- K-means: A simple algorithm for discovering clusters
- Making k-means probabilistic: Gaussian mixture models
- More generally: Latent variable models
- Learning of latent variable models: EM
- Underlying principle: Maximizing ELBO

# Today's Summary

- Motivation: Unsupervised learning
- K-means: A simple algorithm for discovering clusters
- Making k-means probabilistic: Gaussian mixture models
- More generally: Latent variable models
- Learning of latent variable models: EM
- Underlying principle: Maximizing ELBO
- VAE: Introducing variational inference to neural networks. A classic starting example for deep generative modeling.

## Conclusion and Outlook

# Acknowledgement

- Most content developed by David Rosenberg (now at Bloomberg).
- Later adapted by He He, Tal Linzan, and others.
- Added my own twists on the topics.



# Acknowledgement

- Most content developed by David Rosenberg (now at Bloomberg).
- Later adapted by He He, Tal Linzan, and others.
- Added my own twists on the topics.
- This is a very challenging grad-level course.
- Congrats, you are almost done.

# Acknowledgement

- Most content developed by David Rosenberg (now at Bloomberg).
- Later adapted by He He, Tal Linzan, and others.
- Added my own twists on the topics.
- This is a very challenging grad-level course.
- Congrats, you are almost done.
- A reminder to fill out course evaluation surveys.

## Next Lecture: Project Presentation

- Dec 12, in-person presentations.
- 24 groups, 120mins.
- Aim for **3 mins** per group, hard stop at 4 mins, and 1 min max for Q&A.
- Send me your slides in PDF with your group number by Dec 11 11:59pm.

Linear Perceptron, conditional probability models, SVMs

# Models

**Linear** Perceptron, conditional probability models, SVMs

**Non-linear** Kernelized models, trees, basis function models, neural nets

**Linear** Perceptron, conditional probability models, SVMs

**Non-linear** Kernelized models, trees, basis function models, neural nets

How to choose the model family?

- Trade-offs:
  - approximation error and estimation error (bias and variance),
  - accuracy and efficiency (during both training and inference).

**Linear** Perceptron, conditional probability models, SVMs

**Non-linear** Kernelized models, trees, basis function models, neural nets

How to choose the model family?

- Trade-offs:
  - approximation error and estimation error (bias and variance),
  - accuracy and efficiency (during both training and inference).
- Start from the task requirements, e.g. amount of data, computation resource
- The best lesson is to practice!

# Objectives

**Loss functions** How far off a prediction is from the target, e.g. 0-1 loss, margin-based loss, squared loss.



# Objectives

**Loss functions** How far off a prediction is from the target, e.g. 0-1 loss, margin-based loss, squared loss.

**Risk** Expected loss - but expectation over what?

# Objectives

**Loss functions** How far off a prediction is from the target, e.g. 0-1 loss, margin-based loss, squared loss.

**Risk** Expected loss - but expectation over what?

- Frequentist approach: expectation over data.
  - Empirical risk minimization, i.e. average loss on the training data.
  - Regularization: balance estimation error and generalization error.

# Objectives

**Loss functions** How far off a prediction is from the target, e.g. 0-1 loss, margin-based loss, squared loss.

**Risk** Expected loss - but expectation over what?

- Frequentist approach: expectation over data.
  - Empirical risk minimization, i.e. average loss on the training data.
  - Regularization: balance estimation error and generalization error.
- Bayesian approach: expectation over parameters.
  - Posterior: prior belief updated by observed data.
  - Bayes action minimizes the posterior risk.

**Learning** Find model parameters—often an optimization problem.

- (Stochastic) (sub)gradient descent
- Functional gradient descent (gradient boosting)
- Convex vs non-convex objectives

**Inference** Answer questions given a learned model.

- Bayesian inference: compute various quantities given the posterior.
- Dynamic programming: compute  $\arg \max$  in structured prediction.

# Do We Still Need ML?

- Deep Learning (DL) has been overwhelmingly popular in the past few years.

# Do We Still Need ML?

- Deep Learning (DL) has been overwhelmingly popular in the past few years.
- Many ML methods are considered out-dated.

# Do We Still Need ML?

- Deep Learning (DL) has been overwhelmingly popular in the past few years.
- Many ML methods are considered out-dated.
- However, DL is not necessarily good for all types of data (availability, data quality, data modality etc.). Classic methods may also have their sweet spots.

# Do We Still Need ML?

- Deep Learning (DL) has been overwhelmingly popular in the past few years.
- Many ML methods are considered out-dated.
- However, DL is not necessarily good for all types of data (availability, data quality, data modality etc.). Classic methods may also have their sweet spots.
- Classic ML sheds new insight into understand DL.



# Do We Still Need ML?

- Deep Learning (DL) has been overwhelmingly popular in the past few years.
- Many ML methods are considered out-dated.
- However, DL is not necessarily good for all types of data (availability, data quality, data modality etc.). Classic methods may also have their sweet spots.
- Classic ML sheds new insight into understand DL.
- Classic ML lays down foundation when we innovate DL algorithms.

## Other ML Related Advanced Courses in CS

- Computer Vision (Prof. Rob Fergus)
- Deep Learning (Prof. Yann LeCun)
- Deep Reinforcement Learning (Prof. Lerrel Pinto)
- Foundations of Deep Learning Theory (Prof. Matus Telgarsky)
- Inference and Representation (Prof. Joan Bruna)
- Learning with Large Language and Vision Models (Prof. Saining Xie)
- Mathematics of Deep Learning (Prof. Joan Bruna)
- Natural Language Processing (Prof. He He)