Clustering and Latent Variable Models

Mengye Ren

(Slides credit to David Rosenberg, He He, et al.)

NYU

Dec 3, 2024

Lecture Slides

K-means Clustering

Unsupervised learning

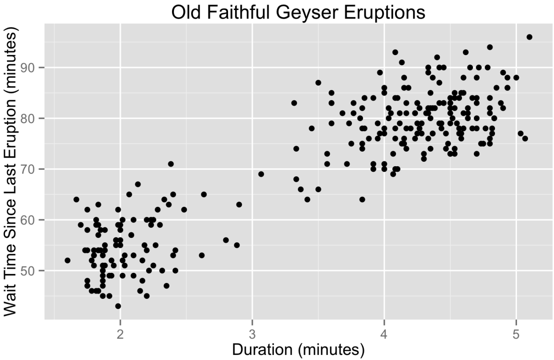
Goal Discover interesting *structure* in the data.

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.

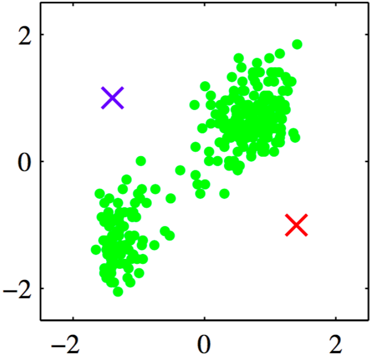
Example: Old Faithful Geyser



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

k-Means: By Example

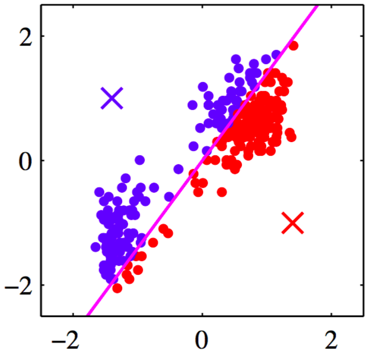
- Standardize the data.
- Choose two cluster centers.



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

k-means: by example

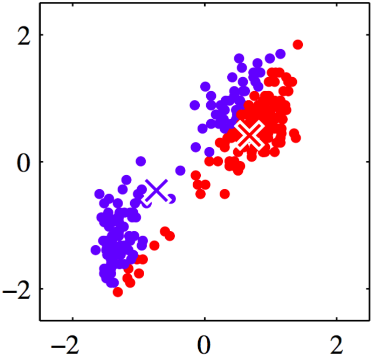
- Assign each point to closest center.



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

k-means: by example

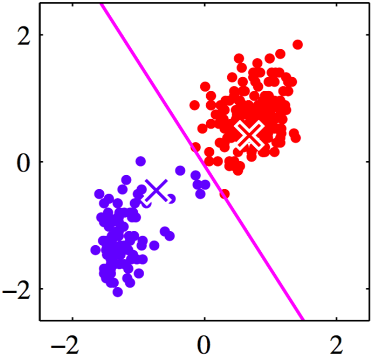
- Compute new cluster centers.



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

k-means: by example

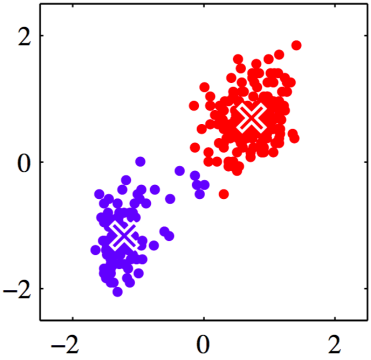
- Assign points to closest center.



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

k-means: by example

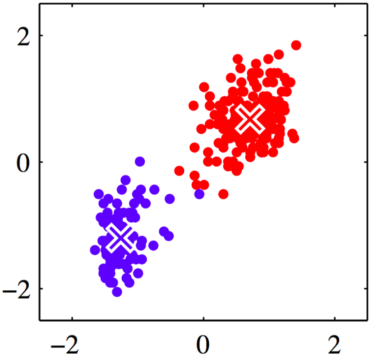
- Compute cluster centers.



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

k-means: by example

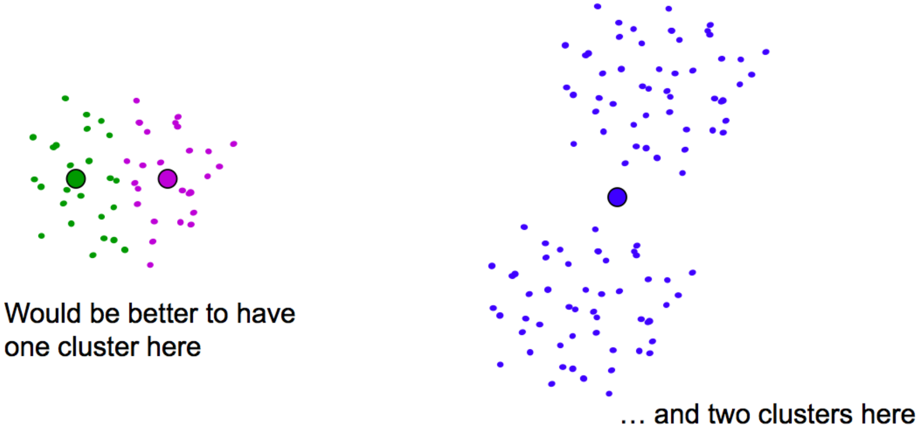
- Iterate until convergence.



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

From Sontag's DS-GA 1003, 2014, Lecture 8.

Formalize k-Means

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

$$\mu_i = \underset{\mu \in \mathcal{X}}{\arg\min} \sum_{x \in C_i} \|x - \mu\|^2. \qquad \text{mean of } C_i \tag{1}$$

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

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

k-Means: Algorithm

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

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

    2. For all j, set

$$\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x \in C_j} x. \qquad \text{Minimze } J \text{ w.r.t. } \mu \text{ while fixing } c. \tag{4}$$

- 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.
- 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: minmizing 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

Gaussian mixture model (GMM)

Generative story of GMM with k mixture components:

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

Probability density of x:

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

$$\begin{aligned} p(x) &= \sum_z p(x, z) && (5) \\ &= \sum_z p(x \mid z) p(z) && (6) \\ &= \sum_k \pi_k \mathcal{N}(\mu_k, \Sigma_k) && (7) \end{aligned}$$

Identifiability Issues for GMM

- Suppose we have found parameters

$$\begin{aligned} \text{Cluster probabilities}: \quad & \pi = (\pi_1, \ldots, \pi_k) \\ \text{Cluster means}: \quad & \mu = (\mu_1, \ldots, \mu_k) \\ \text{Cluster covariance matrices:} \quad & \Sigma = (\Sigma_1, \ldots \Sigma_k) \end{aligned}$$

  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.

Learning GMMs

How to learn the parameters π_k, μ_k, Σ_k?

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

$$
\begin{aligned}
L(\theta) &= \sum_{i=1}^{n} \log p(x_i; \theta) && (8) \\
&= \sum_{i=1}^{n} \log \sum_{z} p(x, z; \theta) && (9)
\end{aligned}
$$

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

Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

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

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

Learning GMMs: observable case

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

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

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

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

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

Learning GMMs: inference

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

$$\begin{aligned} p(z=j \mid x_i) &= p(x, z=j)/p(x) && (14) \\ &= \frac{p(x \mid z=j)p(z=j)}{\sum_k p(x \mid z=k)p(z=k)} && (15) \\ &= \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_k \pi_k \mathcal{N}(x_i \mid \mu_k, \Sigma_k)} && (16) \end{aligned}$$

- $p(z \mid x)$ is a *soft assignment*.
- If we know the parameters μ, Σ, π, this would be easy to compute.

EM for GMM

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

The expectation-minimization (EM) algorithm:

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

M-step for GMM

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

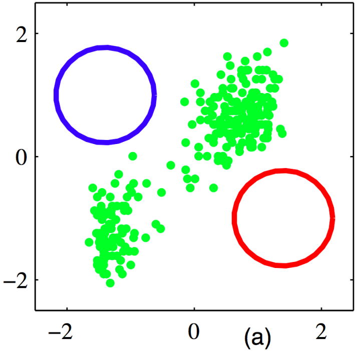
$$\gamma_i^j = \frac{\pi_j^{\text{old}} \mathcal{N}\left(x_i \mid \mu_j^{\text{old}}, \Sigma_j^{\text{old}}\right)}{\sum_{c=1}^{k} \pi_c^{\text{old}} \mathcal{N}\left(x_i \mid \mu_c^{\text{old}}, \Sigma_c^{\text{old}}\right)}.$$

- Exercise: show that

$$\begin{aligned}
n_z &= \sum_{i=1}^{n} \gamma_i^z \\
\mu_z^{\text{new}} &= \frac{1}{n_z} \sum_{i=1}^{n} \gamma_i^z x_i \\
\Sigma_z^{\text{new}} &= \frac{1}{n_z} \sum_{i=1}^{n} \gamma_i^z \left(x_i - \mu_z^{\text{new}}\right)\left(x_i - \mu_z^{\text{new}}\right)^T \\
\pi_z^{\text{new}} &= \frac{n_z}{n}.
\end{aligned}$$

EM for GMM

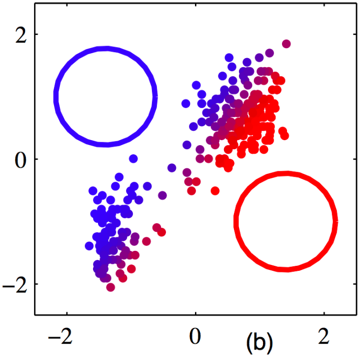
- Initialization



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

EM for GMM

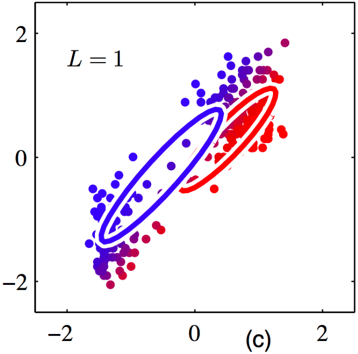
- First soft assignment:



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

EM for GMM

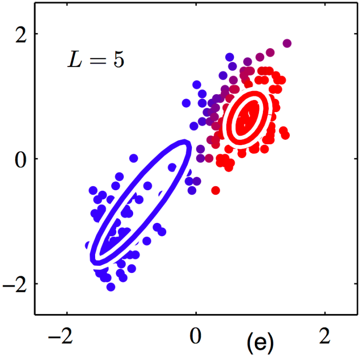
- First soft assignment:



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

EM for GMM

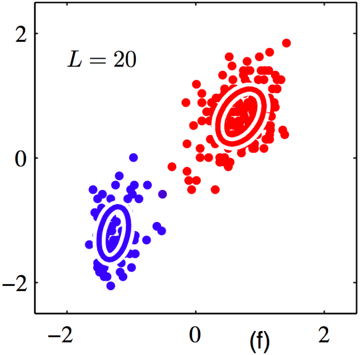
- After 5 rounds of EM:



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**.
- 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 $(x_1, \ldots, x_n)$.
- To simplify notation, take x to represent the entire dataset

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

  and z to represent the corresponding unobserved variables

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

- 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
$$\hat{\theta} = \underset{\theta}{\arg\max}\, p(x \mid \theta).$$
- **Inference problem**: Given x, find conditional distribution over z:
$$p\left(z \mid x, \theta\right).$$
- For Gaussian mixture model, learning is hard, inference is easy.
- For more complicated models, inference can also be hard.

Log-Likelihood and Terminology

- Note that

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

- 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

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 $f : \mathrm{R} \to \mathrm{R}$ is a ***convex*** *function, and x is a random variable, then*

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

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 $\mathbb{E}x^2 \geqslant (\mathbb{E}x)^2$. Thus

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

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

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

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

- Can also write this as

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

Gibbs Inequality ($\mathbf{KL}(p\|q) \geqslant 0$ and $\mathbf{KL}(p\|p) = 0$)

Theorem (Gibbs Inequality)

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

$$KL(p\|q) \geqslant 0,$$

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**.

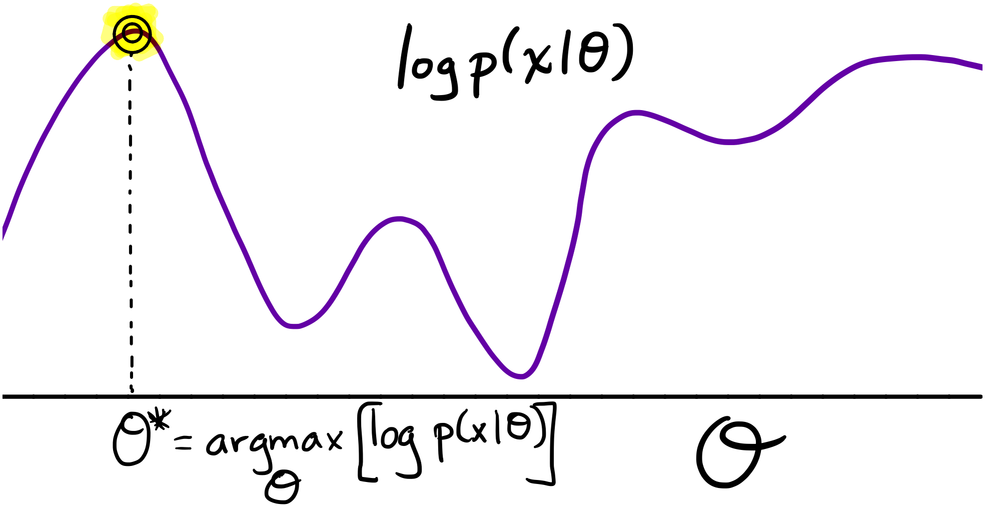
Gibbs Inequality: Proof

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

- 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 $\log p(x \mid \theta)$

The Maximum Likelihood Estimator



$$\theta^* = \arg\max_{\theta} \left[\log p(x|\theta)\right]$$

Lower bound of the marginal log-likelihood

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

- **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 $\log p(x;\theta)$

MLE, EM, and the ELBO

- The MLE is defined as a maximum over θ:

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

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

$$\log p(x \mid \theta) \geqslant \mathcal{L}(q, \theta).$$

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

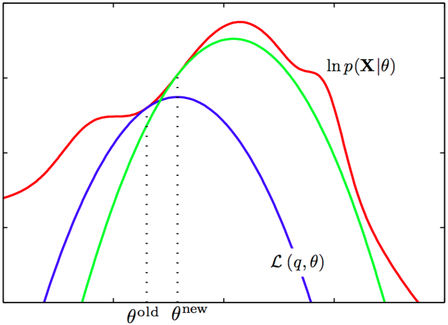
$$\hat{\theta}_{\text{EM}} \approx \arg\max_{\theta} \left[\max_{q} \mathcal{L}(q, \theta)\right]$$

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

EM: Coordinate Ascent on Lower Bound

- Choose sequence of q's and θ's by "**coordinate ascent**" on $\mathcal{L}(q,\theta)$.
- EM Algorithm (high level):
  1. Choose initial θ^{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: $p(x \mid \theta^{\textbf{new}}) \geqslant p(x \mid \theta^{\textbf{old}})$
- **Get sequence of θ's with monotonically increasing likelihood.**

EM: Coordinate Ascent on Lower Bound



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

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

- **KL divergence**: measures “distance” between two distributions (not symmetric!)
- $\mathrm{KL}\left(q\|p\right) \geqslant 0$ with equality iff $q(z) = p(z\mid x)$.
- ELBO $=$ evidence - KL $\leqslant$ evidence

Maximizing over q for fixed θ.

- Find q maximizing

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

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

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

- Summary:

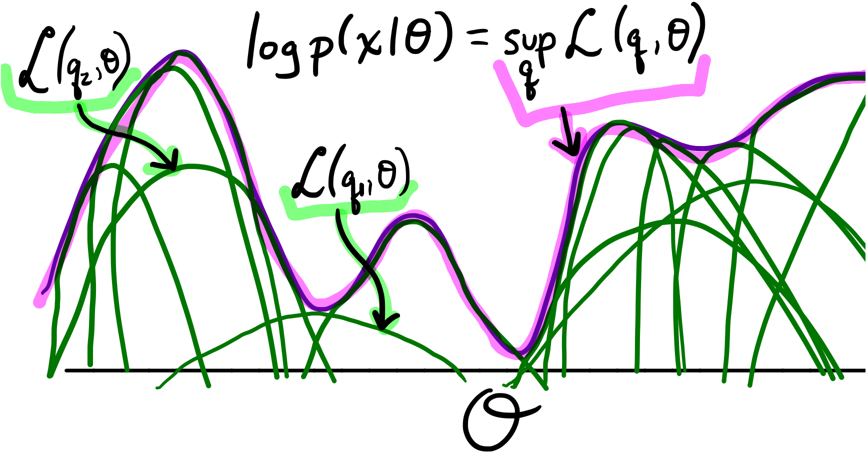
$$\log p(x \mid \theta) = \sup_q \mathcal{L}(q,\theta) \qquad \forall\theta$$

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

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

sup is over all distributions on z

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



Summary

Latent variable models: clustering, latent structure, missing lables etc.

Parameter estimation: maximum marginal log-likelihood

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

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

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

Why does it work?

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

EM algorithm

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

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

      Expectation (the E-step): $q^*(z) = p(z \mid x; \theta^{\text{old}})$

      $J(\theta) = \mathcal{L}(q^*, \theta)$

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

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

EM Algorithm

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

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

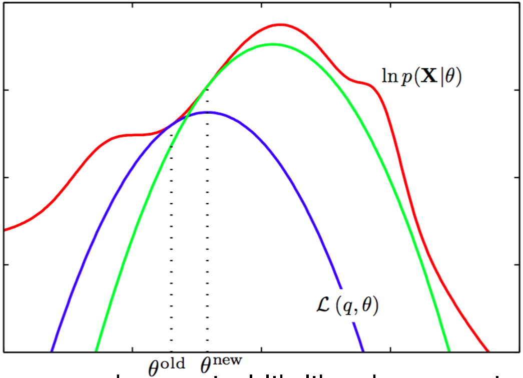
2. **Maximization Step**

$$\theta^{\text{new}} = \underset{\theta}{\arg\max}\, J(\theta).$$

   [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

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

Does EM converge to a global maximum?

Variations on EM

EM Gives Us Two New Problems

- The "E" Step: Computing

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

- 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.
- Rather than finding

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

  find **any** θ^{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.

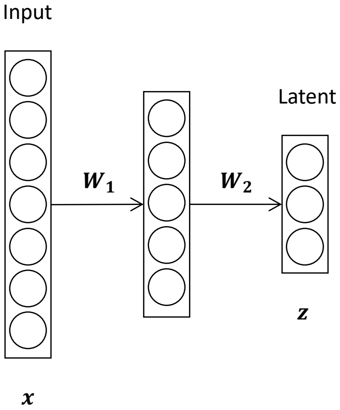
EM and More General Variational Methods

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

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

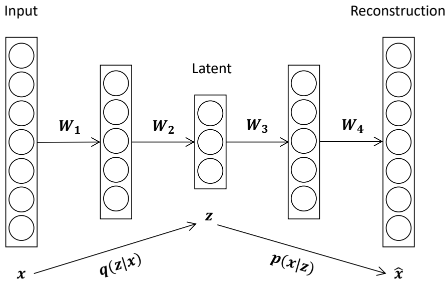
Deep Latent Variable Models

- Neural network is a flexible function class to represent transformation between random variables $e.g., q(z)$.
- 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) [1]

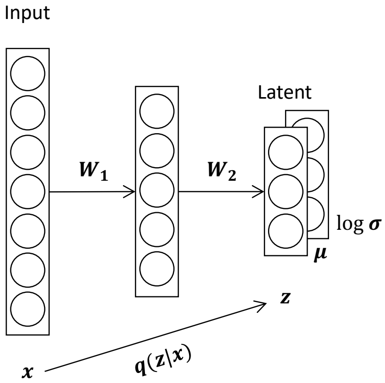
- An autoencoder (AE) is a neural network that reconstructs the same input.
- The first half is an encoder, from input to latent. The second half is a decoder.
- How to make q a probability distribution?



[1] Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014.

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 μ and σ.
- 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: ϕ; Decoder p weights: θ.
- Now maximize ELBO:

$$\begin{aligned}
L(q;\phi,\theta) &= \sum_z q(z) \log \frac{p_\theta(x,z)}{q_\phi(z|x)} && (17)\\
&= \mathbb{E}_{z\sim q}[-\log q_\phi(z|x) + \log p_\theta(x,z)] && (18)\\
&= \mathbb{E}_{z\sim q}[-\log q_\phi(z|x) + \log p_\theta(x|z) + \log p_\theta(z)] && (19)\\
&= \underbrace{-KL(q_\phi(z|x)||p_\theta(z))}_{\text{Divergence between } q \text{ and the prior distribution}} + \underbrace{\mathbb{E}_{z\sim q}(\log p_\theta(x|z))}_{\text{Reconstruction based on } z} && (20)
\end{aligned}$$

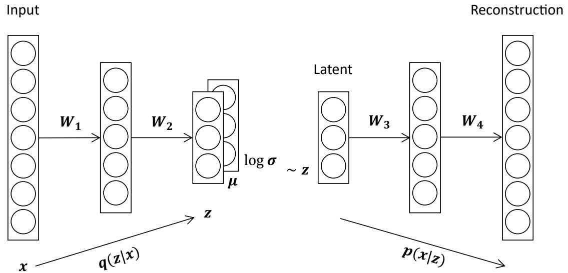
Stochastic Gradient

- The loss function needs to take expectation over q:

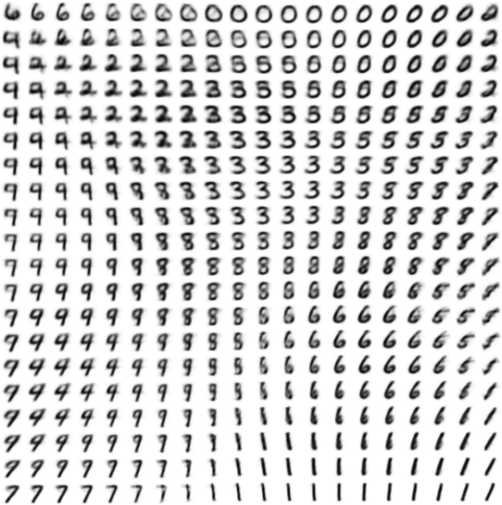
$$L(q;\phi,\theta) = -KL(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{z\sim q}(\log p_\theta(x|z))$$

- 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
- 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 Linzen, and others.
- This is a very challenging grad-level course.
- Congrats, you are almost done.

Next Lecture: Project Presentation

- Dec 10, in-person presentations.
- 22 groups, 120mins.
- Aim for **3 mins** per group, hard stop at 4 mins, and 1 min max for Q&A.
- Send your slides in PDF with your group number by Dec 9 11:59pm (via Google form).

Models

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.

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.

Algorithms

Learning Find model parameters—often an optimization problem.

- (Stocahstic) (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.
- Many ML methods are considered out-dated.
- However, DL is not necessarily good for all types of data (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 in DL algorithms.

Other ML Related Advanced Courses in CS/DS

- Bayesian Machine Learning(Andrew Wilson)
- Computer Vision (Saining Xie)
- Deep Learning (Yann LeCun)
- Deep Reinforcement Learning (Lerrel Pinto)
- Embodied Learning and Vision (Mengye Ren)
- Foundations of Deep Learning Theory (Matus Telgarsky)
- Inference and Representation (Joan Bruna)
- Learning with Large Language and Vision Models (Saining Xie)
- Mathematics of Deep Learning (Joan Bruna)
- Natural Language Processing (He He)