CSC 2515 Lecture 9: Expectation-Maximization

Roger Grosse

University of Toronto

Motivating Examples

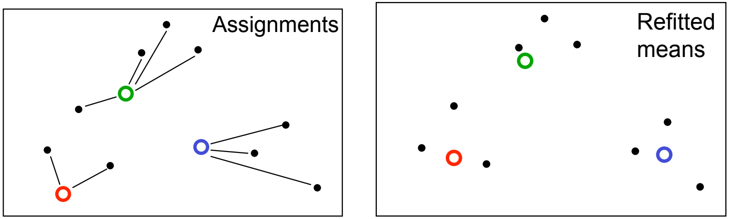
- Some examples of situations where you'd use unupservised learning
  - You want to understand how a scientific field has changed over time. You take a large database of papers and model how the distribution of topics changes from year to year. But what are the topics?
  - You're a biologist studying animal behavior, so you want to infer a high-level description of their behavior from video. You don't know the set of behaviors ahead of time.
  - You want to reduce your energy consumption, so you take a time series of your energy consumption over time, and try to break it down into separate components (refrigerator, washing machine, etc.).
- Common theme: you have some data, and you want to infer the causal structure underlying the data.
- This structure is latent, which means it's never observed.

Overview

- In last lecture, we looked at density modeling where all the random variables were fully observed.
- The more interesting case is when some of the variables are latent, or never observed. These are called latent variable models.
- Today, we'll see how to cluster data by fitting a latent variable model. This will require a new algorithm called Expectation-Maximization (E-M).

Recall: K-means

- Initialization: randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - Assignment step: Assign each data point to the closest cluster
  - Refitting step: Move each cluster center to the center of gravity of the data assigned to it



Recall: K-Means

K-means Objective:

Find cluster centers $\mathbf{m}$ and assignments $\mathbf{r}$ to minimize the sum of squared distances of data points $\{\mathbf{x}^{(i)}\}$ to their assigned cluster centers

$$\min_{\{\mathbf{m}\},\{\mathbf{r}\}} J(\{\mathbf{m}\},\{\mathbf{r}\}) = \min_{\{\mathbf{m}\},\{\mathbf{r}\}} \sum_{i=1}^{N}\sum_{k=1}^{K} r_k^{(i)} \|\mathbf{m}_k - \mathbf{x}^{(i)}\|^2$$

$$\text{s.t.} \sum_k r_k^{(i)} = 1, \forall i, \quad \text{where} \qquad r_k^{(i)} \in \{0,1\}, \forall k, i$$

where $r_k^{(i)} = 1$ means that $\mathbf{x}^{(i)}$ is assigned to cluster k (with center $\mathbf{m}_k$)

- The assignment and refitting steps were each doing coordinate descent on this objective.
- This means the objective improves in each iteration, so the algorithm can't diverge, get stuck in a cycle, etc.

Recall: K-Means

- **Initialization**: Set K means $\{\mathbf{m}_k\}$ to random values
- Repeat until convergence (until assignments do not change):
  - **Assignment**:

$$\hat{k}^i = \arg\min_k d(\mathbf{m}_k, \mathbf{x}^{(i)})$$
$$r_k^{(i)} = 1 \longleftrightarrow \hat{k}^{(i)} = k$$

(hard assignments)

$$r_k^{(i)} = \frac{\exp[-\beta d(\mathbf{m}_k, \mathbf{x}^{(i)})]}{\sum_j \exp[-\beta d(\mathbf{m}_j, \mathbf{x}^{(i)})]}$$

(soft assignments)

  - **Refitting:**

$$\mathbf{m}_k = \frac{\sum_i r_k^{(i)} \mathbf{x}^{(i)}}{\sum_i r_k^{(i)}}$$

A Generative View of Clustering

- What if the data don't look like spherical blobs?
  - elongated clusters
  - discrete data
- This lecture: formulating clustering as a probabilistic model
  - specify assumptions about how the observations relate to latent variables
  - use an algorithm called E-M to (approximtely) maximize the likelihood of the observations
- This lets us generalize clustering to non-spherical ceters or to non-Gaussian observation models (as you do in Homework 4).

Generative Models Recap

- Recall generative classifiers:

$$p(\mathbf{x}, t) = p(\mathbf{x} \mid t)\, p(t)$$

- We fit $p(t)$ and $p(\mathbf{x} \mid t)$ using labeled data.
- If t is never observed, we call it a latent variable, or hidden variable, and generally denote it with z instead.
  - The things we *can* observe (i.e. $\mathbf{x}$) are called observables.
- By marginalizing out z, we get a density over the observables:

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

- This is called a latent variable model.
- If $p(z)$ is a categorial distribution, this is a mixture model, and different values of z correspond to different components.

Gaussian Mixture Model (GMM)

Most common mixture model: Gaussian mixture model (GMM)

- A GMM represents a distribution as

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \, \mathcal{N}(\mathbf{x} \,|\, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

  with π_k the mixing coefficients, where:

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

- This defines a density over $\mathbf{x}$, so we can fit the parameters using maximum likelihood. We're try to match the data density of $\mathbf{x}$ as closely as possible.
  - This is a hard optimization problem (and the focus of this lecture).
- GMMs are **universal approximators of densities** (if you have enough components). Even diagonal GMMs are universal approximators.

Gaussian Mixture Model (GMM)

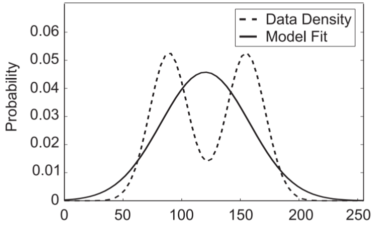
- Can also write the model as a generative process:

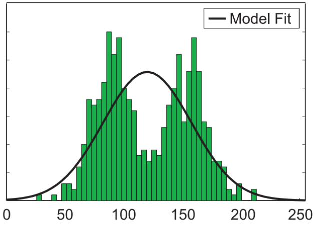
For $i = 1, \dots, N$:

$$\begin{aligned} z^{(i)} &\sim \mathrm{Categorical}(\boldsymbol{\pi}) \\ \mathbf{x}^{(i)} \mid z^{(i)} &\sim \mathcal{N}(\boldsymbol{\mu}_{z^{(i)}}, \boldsymbol{\Sigma}_{z^{(i)}}) \end{aligned}$$

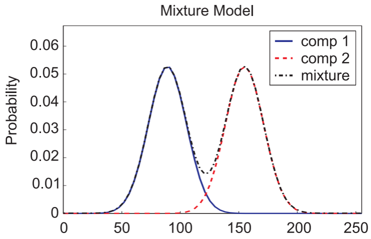
Visualizing a Mixture of Gaussians – 1D Gaussians

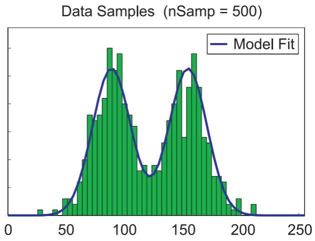
- If you fit a Gaussian to data:





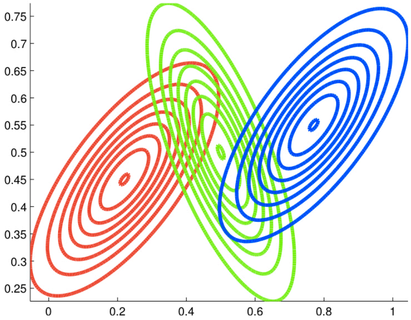
- Now, we are trying to fit a GMM (with $K = 2$ in this example):

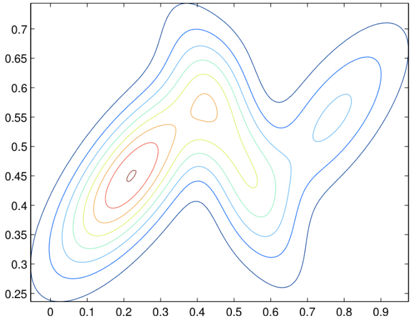


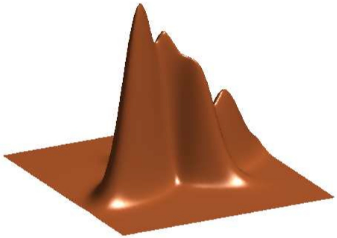


[Slide credit: K. Kutulakos]

Visualizing a Mixture of Gaussians – 2D Gaussians







Fitting GMMs: Maximum Likelihood

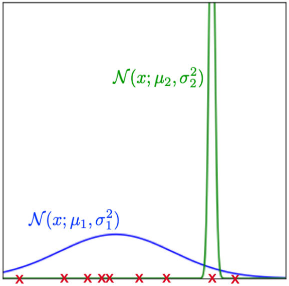
- Some shorthand notation: let $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$ denote the full set of model parameters. Let $\mathbf{X} = \{\mathbf{x}^{(i)}\}$ and $\mathbf{Z} = \{z^{(i)}\}$.
- Maximum likelihood objective:

$$\log p(\mathbf{X}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left(\sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$

- In general, no closed-form solution
- Not identifiable: solution is invariant to permutations
- Challenges in optimizing this using gradient descent?
  - Non-convex (due to permutation symmetry, just like neural nets)
  - Need to enforce non-negativity constraint on π_k and PSD constraint on $\boldsymbol{\Sigma}_k$
  - Derivatives w.r.t. $\boldsymbol{\Sigma}_k$ are expensive/complicated.
- We need a different approach!

Fitting GMMs: Maximum Likelihood

- **Warning:** you don't want the global maximum. You can achieve arbitrarily high training likelihood by placing a small-variance Gaussian component on a training example.
- This is known as a singularity.



Latent Variable Models: Inference

- If we knew the parameters $\boldsymbol{\theta} = \{\pi_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}$, we could infer which component a data point $\mathbf{x}^{(i)}$ probably belongs to by inferring its latent variable $z^{(i)}$.
- This is just posterior inference, which we do using Bayes' Rule:

$$\Pr(z^{(i)} = k \mid \mathbf{x}^{(i)}) = \frac{\Pr(z = k)\, p(\mathbf{x} \mid z = k)}{\sum_{\ell} \Pr(z = \ell)\, p(\mathbf{x} \mid z = \ell)}$$

- Just like Naïve Bayes, GDA, etc. at test time.

Latent Variable Models: Learning

- If we somehow knew the latent variables for every data point, we could simply maximize the joint log-likelihood.

$$\begin{aligned}\log p(\mathbf{X}, \mathbf{Z}; \boldsymbol{\theta}) &= \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}) \\ &= \sum_{i=1}^{N} \log p(z^{(i)}) + \log p(\mathbf{x}^{(i)} \mid z^{(i)}).\end{aligned}$$

- This is just like GDA at training time. Our formulas from last week, written in a suggestive notation:

$$\begin{aligned}\pi_k &= \frac{1}{N} \sum_{i=1}^{N} r_k^{(i)} \\ \boldsymbol{\mu}_k &= \frac{\sum_{i=1}^{N} r_k^{(i)} \cdot \mathbf{x}^{(i)}}{\sum_{i=1}^{N} r_k^{(i)}} \\ \boldsymbol{\Sigma}_k &= \frac{1}{\sum_{i=1}^{N} r_k^{(i)}} \sum_{i=1}^{N} r_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\top \\ r_k^{(i)} &= \mathbb{1}[z^{(i)} = k]\end{aligned}$$

Back to GMM

- But we *don't* know the $z^{(i)}$, so we need to marginalize them out. Now the log-likelihood is more awkward.

$$\begin{aligned}\log p(\mathbf{X}; \boldsymbol{\theta}) &= \sum_{i=1}^{N} \log p(\mathbf{x}^{(i)} \mid \boldsymbol{\theta}) \\ &= \sum_{i=1}^{N} \log \sum_{z^{(i)}=1}^{K} p(\mathbf{x}^{(i)} \mid z^{(i)}; \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\Sigma}_k\})\, p(z^{(i)} \mid \boldsymbol{\pi})\end{aligned}$$

- Problem: the log is outside the sum, so things don't simplify.
- We have a chicken-and-egg problem, just like with K-Means!
    - Given $\boldsymbol{\theta}$, inferring the $z^{(i)}$ is easy.
    - Given the $z^{(i)}$, learning $\boldsymbol{\theta}$ (with maximum likelihood) is easy.
    - Doing both simultaneously is hard.

GMM: Maximum Likelihood

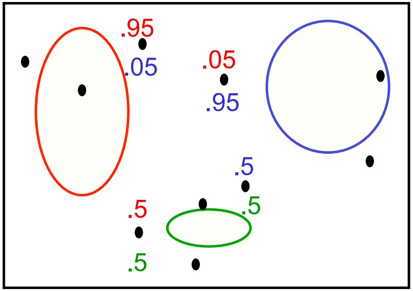
- Here are the maximum likelihood equations for $(\mathbf{x}, z)$ jointly again:

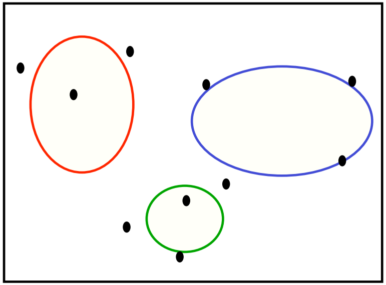
$$\begin{aligned}
\pi_k &= \frac{1}{N} \sum_{i=1}^{N} r_k^{(i)} \\
\boldsymbol{\mu}_k &= \frac{\sum_{i=1}^{N} r_k^{(i)} \cdot \mathbf{x}^{(i)}}{\sum_{i=1}^{N} r_k^{(i)}} \\
\boldsymbol{\Sigma}_k &= \frac{1}{\sum_{i=1}^{N} r_k^{(i)}} \sum_{i=1}^{N} r_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\top \\
r_k^{(i)} &= \mathbb{1}[z^{(i)} = k]
\end{aligned}$$

- Can you guess the algorithm?

Intuitively, How Can We Fit a Mixture of Gaussians?

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





Expectation Maximization for GMM Overview

1. E-step:
   - Assign the responsibility $r_k^{(i)}$ of component k for data point i using the posterior probability:

$$r_k^{(i)} = \Pr(z^{(i)} = k \mid \mathbf{x}^{(i)}; \boldsymbol{\theta})$$

2. M-step:
   - Apply the maximum likelihood updates, where each component is fit with a weighted dataset. The weights are proportional to the responsibilities.

$$\begin{aligned}
\pi_k &= \frac{1}{N} \sum_{i=1}^{N} r_k^{(i)} \\
\boldsymbol{\mu}_k &= \frac{\sum_{i=1}^{N} r_k^{(i)} \cdot \mathbf{x}^{(i)}}{\sum_{i=1}^{N} r_k^{(i)}} \\
\boldsymbol{\Sigma}_k &= \frac{1}{\sum_{i=1}^{N} r_k^{(i)}} \sum_{i=1}^{N} r_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\top
\end{aligned}$$

So why does this work?

Jensen's Inequality

- Recall: if a function f is convex, then

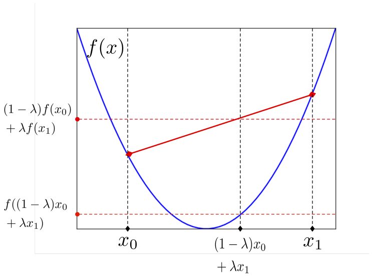
$$f\left(\sum_i \lambda_i f(\mathbf{x}_i)\right) \leq \sum_i \lambda_i f(\mathbf{x}_i),$$

  where $\{\lambda_i\}$ are such that each $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$.

- If we treat the λ_i as the parameters of a categorical distribution, $\lambda_i = \Pr(\mathbf{X} = \mathbf{x}_i)$, this can be rewritten as:

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

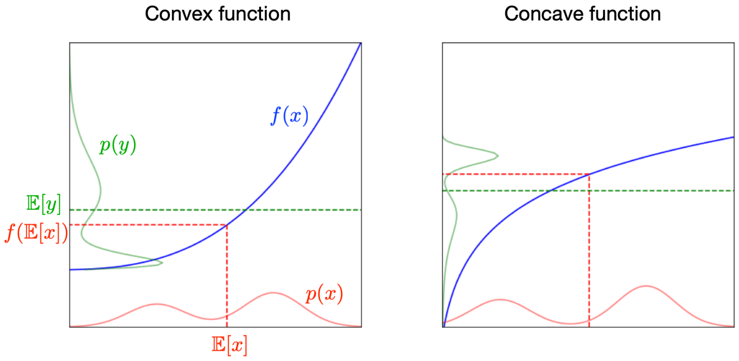
- This is known as Jensen's Inequality. It holds for continuous distributions as well.



Jensen's Inequality

- A function $f(\mathbf{x})$ is concave if $-f(\mathbf{x})$ is convex. In this case, we flip Jensen's Inequality:

$$f(\mathbb{E}[\mathbf{X}]) \geq \mathbb{E}[f(\mathbf{X})].$$



- When would you expect the inequality to be tight?

Where does EM come from?

- Recall: the log-likelihood function is awkward because it has a summation inside the log:

$$\log p(\mathbf{X}; \boldsymbol{\theta}) = \sum_i \log(p(\mathbf{x}^{(i)}; \boldsymbol{\theta})) = \sum_i \log \left(\sum_{z^{(i)}} p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}) \right)$$

- Introduce a new distribution $q(z^{(i)})$ (we'll see what this is shortly):

$$\begin{aligned} \log p(\mathbf{X}; \boldsymbol{\theta}) &= \sum_i \log \left(\sum_{z^{(i)}} q(z^{(i)}) \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})} \right) \\ &= \sum_i \log \mathbb{E}_{q(z^{(i)})} \left[\frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})} \right] \end{aligned}$$

- Notice that log is a concave function. So we can use Jensen's Inequality to push the log inwards, obtaining the variational lower bound:

$$\log p(\mathbf{X}; \boldsymbol{\theta}) \geq \sum_i \mathbb{E}_{q(z^{(i)})} \left[\log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})} \right] \triangleq \mathcal{L}(q, \boldsymbol{\theta})$$

Where does EM come from?

- Just derived a lower bound on the log-likelihood:

$$\log p(\mathbf{X}; \boldsymbol{\theta}) \geq \sum_i \mathbb{E}_{q(z^{(i)})}\left[\log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})}{q(z^{(i)})}\right] \triangleq \mathcal{L}(q, \boldsymbol{\theta})$$

- Simplifying the right-hand-side:

$$\mathcal{L}(q, \boldsymbol{\theta}) = \sum_i \mathbb{E}_{q(z^{(i)})}[\log p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})] - \underbrace{\mathbb{E}_{q(z^{(i)})}[\log q(z^{(i)})]}_{\text{constant w.r.t. } \boldsymbol{\theta}}$$

- The expected log-probability will turn out to be nice.

Where does EM come from?

- Everything so far holds for any choice of q. But what should we actually pick?
- Jensen's inequality gives a lower bound on the log-likelihood, so the best we can achieve is to make the bound tight (i.e. equality).
- Denote the current parameters as $\boldsymbol{\theta}^{\text{old}}$.
- It turns out the posterior probability $p(z^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}})$ is a very good choice for q. Plugging it in to the lower bound:

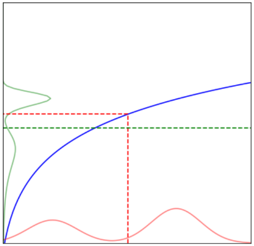
$$\begin{aligned}\sum_i \mathbb{E}_{q(z^{(i)})}\left[\log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}^{\text{old}})}{q(z^{(i)})}\right] &= \sum_i \mathbb{E}_{q(z^{(i)})}\left[\log \frac{p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta}^{\text{old}})}{p(z^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}})}\right] \\ &= \sum_i \mathbb{E}_{q(z^{(i)})}\left[\log p(\mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}})\right] \\ &= \sum_i \log p(\mathbf{x}^{(i)}; \boldsymbol{\theta}^{\text{old}}) \\ &= \log p(\mathbf{X}; \boldsymbol{\theta}^{\text{old}})\end{aligned}$$

- Equality achieved!

Where does EM come from?

An aside:

- How could you pick $q(z^{(i)}) = p(z^{(i)} \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}^{\mathrm{old}})$ if you didn't already know the answer?
- Observe: if f is strictly concave, then Jensen's inequality becomes an equality exactly when the random variable X is determinisic.



- Hence, to solve

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

we should set $q(z^{(i)}) \propto p(\mathbf{x}^{(i)}, z^{(i)}; \boldsymbol{\theta})$.

Where does EM come from?

- **E-step:** compute the responsibilities using Bayes' Rule:

$$r_k^{(i)} \triangleq q(z^{(i)} = k) = \Pr(z^{(i)} = k \mid \mathbf{x}^{(i)}; \boldsymbol{\theta}^{\mathrm{old}})$$

- Rewriting the variational lower bound in terms of the responsibilities:

$$\begin{aligned}\mathcal{L}(q, \boldsymbol{\theta}) = & \sum_i \sum_k r_k^{(i)} \log \Pr(z^{(i)} = k; \boldsymbol{\pi}) \\ & + \sum_i \sum_k r_k^{(i)} \log p(\mathbf{x}^{(i)} \mid z^{(i)} = k; \{\boldsymbol{\mu}_k\}, \{\boldsymbol{\Sigma}_k\}) \\ & + \mathrm{const}\end{aligned}$$

- **M-step:** maximize $\mathcal{L}(\mathbf{X}, \boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, giving $\boldsymbol{\theta}^{\mathrm{new}}$. This can be done analytically, and gives the parameter updates we saw previously.
- The two steps are guaranteed to improve the log-likelihood:

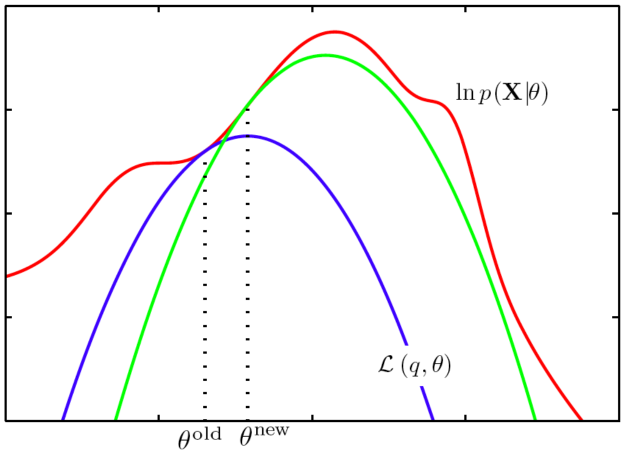
$$\ell(\mathbf{X}; \boldsymbol{\theta}^{\mathrm{new}}) \geq \mathcal{L}(q, \boldsymbol{\theta}^{\mathrm{new}}) \geq \mathcal{L}(q, \boldsymbol{\theta}^{\mathrm{old}}) = \ell(\mathbf{X}; \boldsymbol{\theta}^{\mathrm{old}}).$$

EM: Recap

Recap of EM derivation:

- We're trying to maximize the log-likelihood $\ell(\mathbf{X};\boldsymbol{\theta})$.
- The exact log-likelihood is awkward, but we can use Jensen's Inequality to lower bound it with a nicer function $\mathcal{L}(q,\boldsymbol{\theta})$, the variatonal lower bound, which depends on a choice of q.
- The **E-step** chooses q to make the bound tight at the current parameters $\boldsymbol{\theta}^{\mathrm{old}}$. Mechanistically, this means computing the responsibilities $r_k^{(i)} = \mathrm{Pr}(z^{(i)} = k \,|\, \mathbf{x}^{(i)}; \boldsymbol{\theta}^{\mathrm{old}})$.
- The **M-step** maximizes $\mathcal{L}(q,\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$, giving $\boldsymbol{\theta}^{\mathrm{new}}$. For GMMs, this can be done analytically.
- The combination of the E-step and M-step is guaranteed to improve the true log-likelihood.

Visualization of the EM Algorithm



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

GMM E-Step: Responsibilities

Lets see how it works on GMM:

- Conditional probability (using Bayes' rule) of **z** given **x**

$$\begin{aligned} r_k = \Pr(z = k \mid \mathbf{x}) &= \frac{\Pr(z = k)\, p(\mathbf{x} \mid z = k)}{p(\mathbf{x})} \\ &= \frac{p(z = k)\, p(\mathbf{x} \mid z = k)}{\sum_{j=1}^{K} p(z = j)\, p(\mathbf{x} \mid z = j)} \\ &= \frac{\pi_k\, \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j\, \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)} \end{aligned}$$

GMM E-Step

- Once we computed $r_k^{(i)} = \Pr(z^{(i)} = k \mid \mathbf{x}^{(i)})$ we can compute the expected likelihood

$$
\begin{aligned}
& \mathbb{E}_{p(z^{(i)} \mid \mathbf{x}^{(i)})} \left[\sum_i \log(p(\mathbf{x}^{(i)}, z^{(i)} \mid \boldsymbol{\theta})) \right] \\
= & \sum_i \sum_k r_k^{(i)} \left(\log(\Pr(z^{(i)} = k \mid \boldsymbol{\theta})) + \log(p(\mathbf{x}^{(i)} \mid z^{(i)} = k, \boldsymbol{\theta})) \right) \\
= & \sum_i \sum_k r_k^{(i)} \left(\log(\pi_k) + \log(\mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)) \right) \\
= & \sum_k \sum_i r_k^{(i)} \log(\pi_k) + \sum_k \sum_i r_k^{(i)} \log(\mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))
\end{aligned}
$$

- We need to fit k Gaussians, just need to weight examples by r_k

GMM M-Step

- Need to optimize

$$\sum_k \sum_i r_k^{(i)} \log(\pi_k) + \sum_k \sum_i r_k^{(i)} \log(\mathcal{N}(\mathbf{x}^{(i)}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k))$$

- Solving for $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ is like fitting k separate Gaussians but with weights $r_k^{(i)}$.
- Solution is similar to what we have already seen:

$$\begin{aligned}
\boldsymbol{\mu}_k &= \frac{1}{N_k} \sum_{i=1}^{N} r_k^{(i)} \mathbf{x}^{(i)} \\
\boldsymbol{\Sigma}_k &= \frac{1}{N_k} \sum_{i=1}^{N} r_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^T \\
\pi_k &= \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{i=1}^{N} r_k^{(N)}
\end{aligned}$$

EM Algorithm for GMM

- Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k
- Iterate until convergence:
  - E-step: Evaluate the responsibilities given current parameters

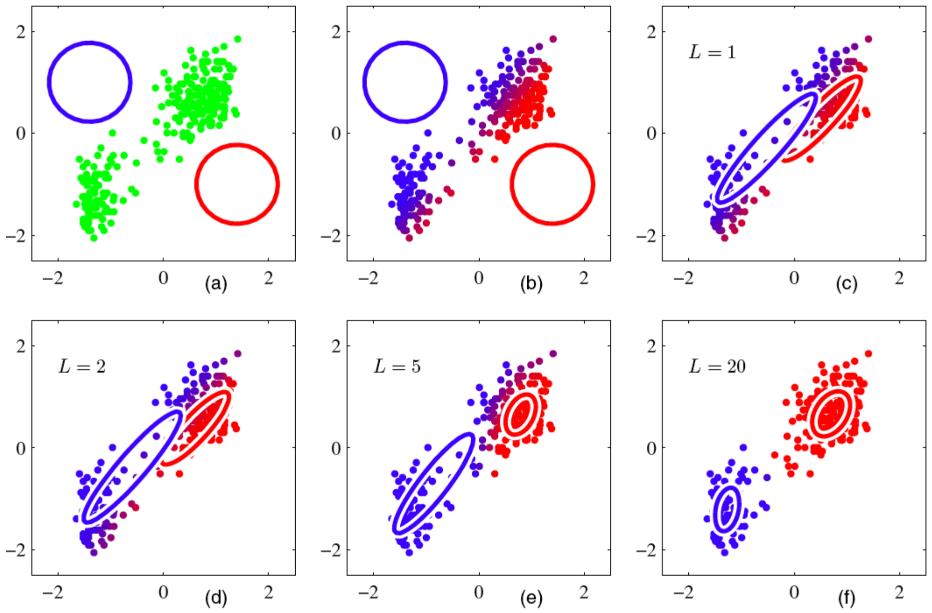
$$r_k^{(i)} = p(z^{(i)} \mid \mathbf{x}^{(i)}) = \frac{\pi_k \mathcal{N}(\mathbf{x}^{(i)} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(\mathbf{x}^{(i)} \mid \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}$$

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

$$\begin{aligned} \boldsymbol{\mu}_k &= \frac{1}{N_k} \sum_{i=1}^N r_k^{(i)} \mathbf{x}^{(i)} \\ \boldsymbol{\Sigma}_k &= \frac{1}{N_k} \sum_{i=1}^N r_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)(\mathbf{x}^{(i)} - \boldsymbol{\mu}_k)^\top \\ \pi_k &= \frac{N_k}{N} \quad \text{with} \quad N_k = \sum_{i=1}^N r_k^{(i)} \end{aligned}$$

  - Evaluate log likelihood and check for convergence

$$\log p(\mathbf{X} \mid \pi, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{i=1}^N \log \left(\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}^{(i)} \mid \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right)$$


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

Mixture of Gaussians vs. K-means

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

EM alternative approach (optional)

- Our goal is to maximize

$$p(\mathbf{X} \mid \boldsymbol{\theta}) = \sum_{\mathbf{z}} p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})$$

- Typically optimizing $p(\mathbf{X} \mid \boldsymbol{\theta})$ is difficult, but $p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})$ is easy
- Let $q(\mathbf{Z})$ be a distribution over the latent variables. For any distribution $q(\mathbf{Z})$ we have

$$\log p(\mathbf{X} \mid \boldsymbol{\theta}) = \mathcal{L}(q, \boldsymbol{\theta}) + \mathrm{D}_{\mathrm{KL}}(q \,\|\, p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta}))$$

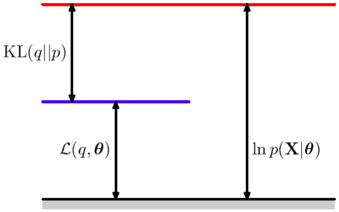
where

$$\begin{aligned} \mathcal{L}(q, \boldsymbol{\theta}) &= \sum_{\mathbf{Z}} q(\mathbf{Z}) \log \left\{ \frac{p(\mathbf{X}, \mathbf{Z} \mid \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} \\ \mathrm{D}_{\mathrm{KL}}(q \,\|\, p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})) &= -\sum_{\mathbf{Z}} q(\mathbf{Z}) \log \left\{ \frac{p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})}{q(\mathbf{Z})} \right\} \end{aligned}$$

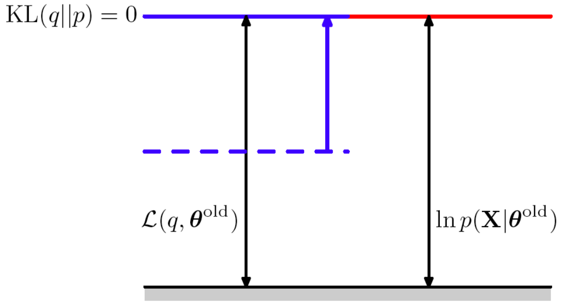
EM alternative approach (optional)

- The KL-divergence is always nonnegative and has value 0 only if $q(\mathbf{Z}) = p(\mathbf{Z} \mid \mathbf{X}, \boldsymbol{\theta})$
- Thus $\mathcal{L}(q, \boldsymbol{\theta})$ is a lower bound on the likelihood

$$\mathcal{L}(q, \boldsymbol{\theta}) \leq \log p(\mathbf{X} \mid \boldsymbol{\theta})$$

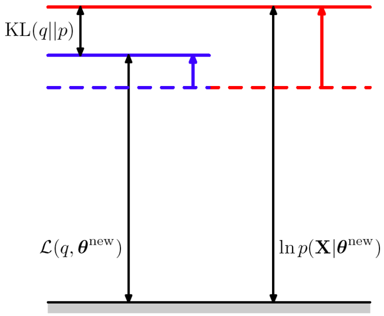

$\mathrm{KL}(q||p)$
$\mathcal{L}(q, \boldsymbol{\theta})$
$\ln p(\mathbf{X}|\boldsymbol{\theta})$

Visualization of E-step (optional)


$\mathrm{KL}(q||p) = 0$
$\mathcal{L}(q, \boldsymbol{\theta}^{\mathrm{old}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\mathrm{old}})$

- The q distribution equal to the posterior distribution for the current parameter values $\boldsymbol{\theta}^{old}$, causing the lower bound to move up to the same value as the log likelihood function, with the KL divergence vanishing.

Visualization of M-step (optional)



$\mathrm{KL}(q||p)$
$\mathcal{L}(q, \boldsymbol{\theta}^{\mathrm{new}})$
$\ln p(\mathbf{X}|\boldsymbol{\theta}^{\mathrm{new}})$

- The distribution $q(\mathbf{Z})$ is held fixed and the lower bound $\mathcal{L}(q, \boldsymbol{\theta})$ is maximized with respect to the parameter vector $\boldsymbol{\theta}$ to give a revised value $\boldsymbol{\theta}^{new}$. Because the KL divergence is nonnegative, this causes the log likelihood $\log p(\mathbf{X} \mid \boldsymbol{\theta})$ to increase by at least as much as the lower bound does.
- Hence, EM is basically a coordinate ascent procedure on a particular objective function, analogously to K-Means!

GMM Recap

- A probabilistic view of clustering - Each cluster corresponds to a different Gaussian.
- Model using latent variables.
- General approach, can replace Gaussian with other distributions (continuous or discrete)
- More generally, mixture model are very powerful models, universal approximator
- Optimization is done using the EM algorithm.

Hidden Markov Models (optional)

- The general EM framework probably seems very overpowered if all you want to do is clustering. But it's much more general.
- I'd like to very quickly give a more interesting example of the EM algorithm, namely the Baum-Welch algorithm for learning hidden Markov models.
- We don't have nearly enough time to cover this properly. So the rest of this lecture is optional as far as exams are concerned. I just want to give you a taste.
- This is covered in detail in CSC2506.

Hidden Markov Models (optional)

- Suppose we want a distribution over sequences of states $x_{1:T} = (x_1, \ldots, x_T)$. By the Chain Rule of Probability, this distribution factorizes as:

$$p(x_{1:T}) = p(x_1)\, p(x_2 \,|\, x_1)\, p(x_3 \,|\, x_1, x_2) \cdots p(x_T \,|\, x_1, \ldots, x_{T-1}).$$

- The Markov property is the assumption that the sequence is memoryless, in the sense that each state depends only on the previous state.
  - More formally, for each time t, x_t is conditionally independent of $x_t, \ldots, x_{t-2}$ given x_{t-1}.
  - This corresponds to a factorization of the joint distribution as:

$$p(x_{1:T}) = p(x_1)\, p(x_2 \,|\, x_1)\, p(x_3 \,|\, x_2) \cdots p(x_T \,|\, x_{T-1}).$$

- Markov assumptions are very common, and we'll use one next week for reinforcement learning (stay tuned...)

Hidden Markov Models (optional)

- Now suppose we don't get to observe the states directly. Instead, we get observations that tell us information about the states.
- Now the states are latent (or hidden) variables, so we'll denote them $z_1, \ldots, z_T$, and denote the observations $x_1, \ldots, x_T$.
- A hidden Markov model (HMM) makes the following assumptions:
  - The latent states are discrete
  - The latent states are Markov, i.e.

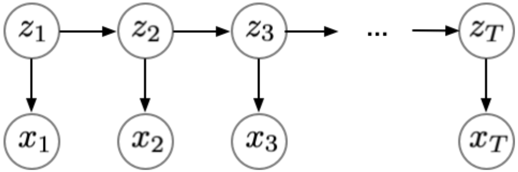
$$p(z_{1:T}) = p(z_1)\, p(z_2 \mid z_1)\, p(z_3 \mid z_2) \cdots p(z_T \mid z_{T-1}).$$

  - Each observation x_t depends only on the current state z_t. More precisely, each x_t is conditionally independent of all the other variables in the network given z_t.
- This corresponds to a factorization of the joint distribution:

$$p(z_{1:T}, x_{1:T}) = p(z_1) \prod_{t=2}^{T} p(z_t \mid z_{t-1}) \prod_{t=1}^{T} p(x_t \mid z_t).$$

Hidden Markov Models (optional)

- Representation of an HMM as a probabilistic graphical model:



- Some examples of HMMs:
  - In speech recognition, the state z_t can correspond to the phoneme being spoken, and the state x_t to a set of acoustic features. This is how speech recognition was done before deep learning took over in 2010 or so.
  - In part-of-speech tagging, z_t corresponds to the part of speech, and x_t to the English word that's generated.
- If we don't have any labels for the states (or even know what the categories should be), how can we learn this automatically from data?

Hidden Markov Models (optional)

- The HMM is another example of a latent variable model, and we can (approximately) maximize the likelihood as a special case of the more general EM framework we've developed.
- The difference is that the latent variables are more structured, and therefore the E- and M-steps are also more structured.
- Recall that we need to derive:
  - **E-step:** Compute the posterior distribution $q(z_{1:T}) = p(z_{1:T} \mid x_{1:T})$. (But what does it mean to "compute" it?)
  - **M-step:** Maximize the expected log-likelihood $\sum_i \mathbb{E}_{q(z^{(i)}_{1:T})}[\log p(z^{(i)}_{1:T}, x^{(i)}_{1:T})]$.
- Applying the EM algorithm to HMMs is the Baum-Welch Algorithm (and actually predated the general EM framework!).

HMM: M-step (optional)

- For simplicity, assume all the x_t and z_t are binary, so we're trying to learn the parameters of Bernoulli distributions:

$$\begin{aligned} \Pr(z_1 = 1) &= \phi_{\text{init}} \\ \Pr(z_t = 1 \mid z_{t-1} = a) &= \phi_a \\ \Pr(x_t = 1 \mid z_t = a) &= \theta_a. \end{aligned}$$

- Joint log-probability of $x_{1:T}$ and $z_{1:T}$:

$$\log p(z_{1:T}, x_{1:T}) = \underbrace{\log p(z_1)}_{\text{only } \phi_{\text{init}}} + \underbrace{\sum_{t=2}^{T} \log p(z_t \mid z_{t-1})}_{\text{only } \phi_a} + \underbrace{\sum_{t=1}^{T} \log p(x_t \mid z_t)}_{\text{only } \theta_a}$$

- All three groups of parameters can be treated similarly, so let's focus on just the transition probabilities $\{\phi_a\}$.

HMM: M-step (optional)

- For estimating the $\{\phi_a\}$,

$$
\begin{aligned}
\log p(\mathbf{X}, \mathbf{Z}) &= \sum_{i=1}^{N} \log p(z_{1:T}^{(i)}, x_{1:T}^{(i)}) \\
&= \sum_{i=1}^{N} \sum_{t=2}^{T} \log p(z_t^{(i)} \mid z_{t-1}^{(i)}) + \mathrm{const} \\
&= \sum_{i=1}^{N} \sum_{t=2}^{T} z_t^{(i)} z_{t-1}^{(i)} \log \phi_1 + \sum_{i=1}^{N} \sum_{t=2}^{T} (1 - z_t^{(i)}) z_{t-1}^{(i)} \log(1 - \phi_1) \\
&\quad + \sum_{i=1}^{N} \sum_{t=2}^{T} z_t^{(i)} (1 - z_{t-1}^{(i)}) \log \phi_0 + \sum_{i=1}^{N} \sum_{t=2}^{T} (1 - z_t^{(i)})(1 - z_{t-1}^{(i)}) \log(1 - \phi_0)
\end{aligned}
$$

- Hence, the expected log-likelihood is given by:

$$
\begin{aligned}
\mathbb{E}_{q(\mathbf{Z})}[\log p(\mathbf{X}, \mathbf{Z})] &= \sum_{i=1}^{N} \sum_{t=2}^{T} \mathbb{E}[z_t^{(i)} z_{t-1}^{(i)}] \log \phi_1 + \sum_{i=1}^{N} \sum_{t=2}^{T} \mathbb{E}[(1 - z_t^{(i)}) z_{t-1}^{(i)}] \log(1 - \phi_1) \\
&\quad + \sum_{i=1}^{N} \sum_{t=2}^{T} \mathbb{E}[z_t^{(i)} (1 - z_{t-1}^{(i)})] \log \phi_0 + \sum_{i=1}^{N} \sum_{t=2}^{T} \mathbb{E}[(1 - z_t^{(i)})(1 - z_{t-1}^{(i)})] \log(1 - \phi_0) \\
&\quad + \mathrm{const}
\end{aligned}
$$

HMM: M-step (optional)

- Just showed:

$$\begin{aligned}\mathbb{E}_{q(\mathbf{Z})}[\log p(\mathbf{X}, \mathbf{Z})] &= \sum_{i=1}^{N}\sum_{t=2}^{T}\mathbb{E}[z_t^{(i)} z_{t-1}^{(i)}]\log\phi_1 + \sum_{i=1}^{N}\sum_{t=2}^{T}\mathbb{E}[(1-z_t^{(i)})z_{t-1}^{(i)}]\log(1-\phi_1) \\ &+ \sum_{i=1}^{N}\sum_{t=2}^{T}\mathbb{E}[z_t^{(i)}(1-z_{t-1}^{(i)})]\log\phi_0 + \sum_{i=1}^{N}\sum_{t=2}^{T}\mathbb{E}[(1-z_t^{(i)})(1-z_{t-1}^{(i)})]\log(1-\phi_0) \\ &+ \text{const}\end{aligned}$$

- Setting the partial derivatives to zero, we get the M-step update:

$$\phi_1 = \frac{\sum_i \sum_t \mathbb{E}_q[z_t^{(i)} z_{t-1}^{(i)}]}{\sum_i \sum_t \mathbb{E}_q[z_{t-1}^{(i)}]}$$

$$\phi_0 = \frac{\sum_i \sum_t \mathbb{E}_q[z_t^{(i)}(1-z_{t-1}^{(i)})]}{\sum_i \sum_t \mathbb{E}_q[1-z_{t-1}^{(i)}]}$$

- The M-step updates for the other parameters are analogous.

HMM: E-step (optional)

- That was the M-step. How about the E-step?
- In principle, we need to "find" a distribution $q(z_{1:T})$. But representing this distribution explicitly requires a table with 2^T entries!
- But notice: in the M-step, the only thing we needed from q was the expectations $\mathbb{E}_q[z_t z_{t-1}]$, etc. Hence, we only need to determine the marginal distributions $q(z_{t-1}, z_t)$ over pairs of states.
- There is a clever dynamic programming algorithm called the forward-backward algorithm which computes all these marginals in linear time. You can read about it in Bishop, and you'll learn about it (and a much broader class of related algorithms) in CSC2506.
- This is a good example where deriving the M-step tells us exactly what work we need to do in the E-step. Often, we can compute the necessary statistics using algorithms that exploit lots of problem structure.

EM Recap

- A general algorithm for optimizing many latent variable models.
- Iteratively computes a lower bound then optimizes it.
- Converges but maybe to a local minima.
- Can use multiple restarts.
- Can initialize from k-means
- Limitation - need to be able to compute $p(z \mid \mathbf{x}; \boldsymbol{\theta})$, not possible for more complicated models.
    - Solution: Variational inference (see CSC2506)