Latent Variable Models

Stefano Ermon, Aditya Grover

Stanford University

Lecture 5
Recap of last lecture

1. Autoregressive models:
   - Chain rule based factorization is fully general
   - Compact representation via *conditional independence* and/or *neural parameterizations*

2. Autoregressive models Pros:
   - Easy to evaluate likelihoods
   - Easy to train

3. Autoregressive models Cons:
   - Requires an ordering
   - Generation is sequential
   - Cannot learn features in an unsupervised way
Plan for today

1. Latent Variable Models
   - Variational EM
Latent Variable Models: Motivation

1. Lots of variability in images $\mathbf{x}$ due to gender, eye color, hair color, pose, etc. However, unless images are annotated, these factors of variation are not explicitly available (latent).

2. **Idea**: explicitly model these factors using latent variables $\mathbf{z}$
1. Only shaded variables $x$ are observed in the data (pixel values).
2. Latent variables $z$ correspond to high level features.
   - If $z$ chosen properly, $p(x|z)$ could be much simpler than $p(x)$.
   - If we had trained this model, then we could identify features via $p(z|x)$, e.g., $p(\text{EyeColor} = \text{Blue}|x)$.
3. **Challenge:** Very difficult to specify these conditionals by hand.
1. $z \sim \mathcal{N}(0, I)$

2. $p(x \mid z) = \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z))$ where $\mu_\theta, \Sigma_\theta$ are neural networks

3. Hope that after training, $z$ will correspond to meaningful latent factors of variation (features). Unsupervised representation learning.

4. As before, features can be computed via $p(z \mid x)$
Mixture of Gaussians. Bayes net: $z \rightarrow x$.

1. $z \sim \text{Categorical}(1, \cdots, K)$
2. $p(x \mid z = k) = \mathcal{N}(\mu_k, \Sigma_k)$

Generative process

1. Pick a mixture component $k$ by sampling $z$
2. Generate a data point by sampling from that Gaussian
Mixture of Gaussians:

1. \( z \sim \text{Categorical}(1, \cdots, K) \)
2. \( p(x \mid z = k) = \mathcal{N}(\mu_k, \Sigma_k) \)

3. **Clustering:** The posterior \( p(z \mid x) \) identifies the mixture component
4. **Unsupervised learning:** We are hoping to learn from unlabeled data (ill-posed problem)
Unsupervised learning
Unsupervised learning

Shown is the posterior probability that a data point was generated by the $i$-th mixture component, $P(z = i|x)$.
Unsupervised learning

Unsupervised clustering of handwritten digits.
Unsupervised learning

Combine simple models into a more complex and expressive one

\[ p(x) = \sum_{z} p(x, z) = \sum_{z} p(z)p(x \mid z) = \sum_{k=1}^{K} p(z = k) \mathcal{N}(x; \mu_k, \Sigma_k) \]
A mixture of an infinite number of Gaussians:

1. \( z \sim \mathcal{N}(0, I) \)

2. \( p(x \mid z) = \mathcal{N}(\mu_\theta(z), \Sigma_\theta(z)) \) where \( \mu_\theta, \Sigma_\theta \) are neural networks
   - \( \mu_\theta(z) = \sigma(Az + c) = (\sigma(a_1z + c_1), \sigma(a_2z + c_2)) = (\mu_1(z), \mu_2(z)) \)
   - \( \Sigma_\theta(z) = \text{diag}(\exp(\sigma(Bz + d))) = \begin{pmatrix} \exp(\sigma(b_1z + d_1)) & 0 \\ 0 & \exp(\sigma(b_2z + d_2)) \end{pmatrix} \)
   - \( \theta = (A, B, c, d) \)

3. Even though \( p(x \mid z) \) is simple, the marginal \( p(x) \) is very complex/flexible
Latent Variable Models

- Allow us to define complex models $p(\mathbf{x})$ in terms of simple building blocks $p(\mathbf{x} \mid \mathbf{z})$
- Natural for unsupervised learning tasks (clustering, unsupervised representation learning, etc.)
- No free lunch: much more difficult to learn compared to fully observed, autoregressive models
Marginal Likelihood

- Suppose some pixel values are missing at train time (e.g., top half)
- Let $X$ denote observed random variables, and $Z$ the unobserved ones (also called hidden or latent)
- Suppose we have a model for the joint distribution (e.g., PixelCNN)

$$p(X, Z; \theta)$$

What is the probability $p(X = \bar{x}; \theta)$ of observing a training data point $\bar{x}$?

$$\sum_z p(X = \bar{x}, Z = z; \theta) = \sum_z p(\bar{x}, z; \theta)$$

- Need to consider all possible ways to complete the image (fill green part)
Suppose that our joint distribution is

\[ p(X, Z; \theta) \]

We have a dataset \( D \), where for each datapoint the \( X \) variables are observed (e.g., pixel values) and the variables \( Z \) are never observed (e.g., cluster or class id.). \( D = \{x^{(1)}, \ldots, x^{(M)}\} \).

Maximum likelihood learning:

\[
\ell(\theta; D) = \log \prod_{x \in D} p(x; \theta) = \sum_{x \in D} \log p(x; \theta)
\]

\[
= \sum_{x \in D} \log \sum_z p(x, z; \theta)
\]

Evaluating \( \sum_z p(x, z; \theta) \) can be intractable. Suppose we have 30 binary latent features, \( z \in \{0, 1\}^{30} \). Evaluating \( \sum_z p(x, z; \theta) \) involves a sum with \( 2^{30} \) terms. For continuous variables, \( \log \int_z p(x, z; \theta)dz \) is often intractable.
Why is parameter learning in presence of Partially Observed Data challenging?

Likelihood function for Fully Observed Data:

\[
p_{\theta}(x) = \prod_{i} \log p(x_i | x_{pa(i)})
\]

easy to compute

Compare with Likelihood function for Partially Observed Data:

\[
\sum_{z} p_{\theta}(x, z)
\]

hard to compute

Likelihood function for Partially Observed Data:

- is not decomposable (by variable and parent assignment) and not unimodal as a function of \( \theta \). Could still try gradient descent.
- Hard to compute and take gradients \( \nabla_{\theta} \) (too many completions)
First attempt: Naive Monte Carlo

Likelihood function for Partially Observed Data is hard to compute:

\[
\sum_{\text{All possible values of } z} p_{\theta}(x, z) = |Z| \sum_{z \in Z} \frac{1}{|Z|} p_{\theta}(x, z) = |Z| \mathbb{E}_{z \sim \text{Uniform}(Z)} [p_{\theta}(x, z)]
\]

We can think of it as an (intractable) expectation. Monte Carlo to the rescue:

1. Sample \( z^{(1)}, \ldots, z^{(k)} \) uniformly at random
2. Approximate expectation with sample average

\[
\sum_{z} p_{\theta}(x, z) \approx |Z| \frac{1}{k} \sum_{j=1}^{k} p_{\theta}(x, z^{(j)})
\]

Works in theory but not in practice. For most \( z \), \( p_{\theta}(x, z) \) is very low (most completions don’t make sense). Some are very large but will never ”hit” likely completions by uniform random sampling. Need a clever way to select \( z^{(j)} \) to reduce variance of the estimator.
The Expectation Maximization (EM) Algorithm

- Start with an initial guess (random) of the parameters $\theta^{(0)}$
- Repeat until convergence
  1. Complete (“hallucinate”) the incomplete data (the $z$ part) using current parameters (E-step)
  2. Train: Update the parameters based on the completed data (M-step)
The Expectation Maximization Algorithm: Example

\[ \theta_a = .3 \]
\[ \theta_b = .9 \]
\[ \theta_{c|\bar{a},\bar{b}} = .83 \]
\[ \theta_{c|\bar{a},b} = .09 \]
\[ \theta_{c|a,\bar{b}} = .6 \]
\[ \theta_{c|a,b} = .2 \]
\[ \theta_{d|\bar{c}} = .1 \]
\[ \theta_{d|c} = .8 \]

Data instance: \((a, ?, ?, \bar{d})\)

How to complete this example?

**For each possible completion**

- **STEP 1:** Compute how likely the completion is (given the observed part).
- Compute \(P(z|x)\). In this example
  \[ P(B, C \mid A = a, D = \bar{d})\].

For example,

\[
P(b, c \mid a, \bar{d}) = \frac{P(a, b, c, \bar{d})}{P(a, \bar{d})} = \frac{0.3 \cdot 0.9 \cdot 0.2 \cdot (1 - 0.8)}{0.3 \cdot 0.9 \cdot 0.2 \cdot (1 - 0.8) + \cdots + 0.3 \cdot 0.1 \cdot 0.4 \cdot 0.9} = .0492
\]

\[
P(\bar{b}, c \mid a, \bar{d}) = \frac{P(a, \bar{b}, c, \bar{d})}{P(a, \bar{d})} = \frac{0.3 \cdot 0.1 \cdot 0.6 \cdot (1 - 0.8)}{0.3 \cdot 0.9 \cdot 0.2 \cdot (1 - 0.8) + \cdots + 0.3 \cdot 0.1 \cdot 0.4 \cdot 0.9} = .0164
\]
The Expectation Maximization Algorithm

- Data set is now **bigger and weighted**
- If binary variables, \((a, ?, ?, \tilde{d})\) corresponds to four weighted examples
  - \((a, b, c, \tilde{d})\), weight = \(0.0492 = \Pr(b, c \mid a, \tilde{d})\)
  - \((a, b, \tilde{c}, \tilde{d})\), weight = \(0.8852 = \Pr(b, \tilde{c} \mid a, \tilde{d})\)
  - \((a, \tilde{b}, c, \tilde{d})\), weight = \(0.0164 = \Pr(b, c \mid a, d)\)
  - \((a, \tilde{b}, \tilde{c}, \tilde{d})\), weight = \(0.0492 = \Pr(b, \tilde{c} \mid a, \tilde{d})\)
- **weight = probability according to current parameter estimates**
- After completion, the dataset is fully observed, so we can train as usual via gradient descent or closed-form (M-Step)

\[
\theta_{t+1} = \arg \max_\theta \sum_{m=1}^{M} E_{p(z_m \mid x_m; \theta_t)} [\log p(x_m, z_m; \theta)]
\]

- Two problems:
  1. \(p(z_m \mid x_m; \theta_t) = p(x_m, z_m; \theta_t) / p(x_m; \theta_t)\) requires \(p(x_m; \theta_t) = \sum_z p(x_m, z; \theta_t)\) (what you’d need for gradient descent)
  2. Still requires looking at all possible completions
- Solution: replace \(p(z \mid x; \theta)\) with a tractable \(q(z)\) and do Monte Carlo
The EM Algorithm

Suppose $q(z)$ is any probability distribution over the hidden variables.

$$D_{KL}(q(z)||p(z|x; \theta)) = \sum_z q(z) \log \frac{q(z)}{p(z|x; \theta)}$$

$$= - \sum_z q(z) \log p(z|x; \theta) + \sum_z q(z) \log q(z)$$

$$= - \sum_z q(z) \log p(z|x; \theta) - H(q)$$

$$= - \sum_z q(z) \log \frac{p(z, x; \theta)}{p(x; \theta)} - H(q)$$

$$= - \sum_z q(z) \log p(z, x; \theta) + \sum_z q(z) \log p(x; \theta) - H(q)$$

$$= - \sum_z q(z) \log p(z, x; \theta) + \log p(x; \theta) - H(q) \geq 0$$
The EM Algorithm

- Suppose \( q(z) \) is any probability distribution over the hidden variables

\[
D_{KL}(q(z) \| p(z|x; \theta)) = - \sum_z q(z) \log p(z, x; \theta) + \log p(x; \theta) - H(q) \geq 0
\]

- **Evidence lower bound** (ELBO) holds for any \( q \)

\[
\log p(x; \theta) \geq \sum_z q(z) \log p(z, x; \theta) + H(q)
\]

- Equality holds if \( q = p(z|x; \theta) \)

\[
\log p(x; \theta) = \sum_z q(z) \log p(z, x; \theta) + H(q)
\]

- This is what we compute in the E-step of the EM algorithm
The EM Algorithm

- $q(z)$ is an arbitrary probability distribution over $z$

$$D_{KL}(q\|p(z|x; \theta)) = -\sum z q(z) \log p(z, x; \theta) + \log p(x; \theta) - H(q)$$

- Re-arranging can rewrite as

$$H(q) + \sum z q(z) \log p(z, x; \theta) = \log p(x; \theta) - D_{KL}(q\|p(z|x; \theta)) \triangleq F[q, \theta]$$

- Can interpret EM as coordinate ascent on $F[q, \theta]$

  1. Initialize $\theta^{(0)}$
  2. $q^{(0)} = \arg \max_q F[q, \theta^{(0)}] = p(z|x; \theta^{(0)})$
  3. $\theta^{(1)} = \arg \max_\theta F[q^{(0)}, \theta] = \arg \max_\theta \sum z q^{(0)}(z) \log p(z, x; \theta)$
  4. $q^{(1)} = \arg \max_q F[q, \theta^{(1)}]$
  5. ...

- Marginal likelihood never decreases

$$\log p(x; \theta^{(0)}) = F[q^{(0)}, \theta^{(0)}] \leq F[q^{(0)}, \theta^{(1)}] = \log p(x; \theta^{(1)}) - D_{KL}(q^{(0)}\|p(z|x; \theta^{(1)}))$$

$$\leq \log p(x; \theta^{(1)}) = F[q^{(1)}, \theta^{(1)}] = \ldots$$
Coordinate ascent

- Maximize along one direction at a time:
  - Initialize $x_0, y_0$
  - $y_1 = \arg \max_y f(x_0, y)$
  - $x_1 = \arg \max_x f(x, y_1)$
  - $y_2 = \arg \max_y f(x_1, y)$
  - ...

- Objective keeps improving.
  \[ f(x_0, y_0) \leq f(x_0, y_1) \leq f(x_1, y_1) \leq f(x_1, y_2) \leq \cdots \]
Derivation of EM algorithm

(Figure adapted from tutorial by Sean Borman)
What if the posterior $p(z|x; \theta)$ is intractable to compute in the E-step?

Suppose $q(z; \phi)$ is a (tractable) probability distribution over the hidden variables parameterized by $\phi$ (variational parameters)

- E.g., a Gaussian with mean and covariance specified by $\phi$, a fully factored probability distribution, a FVSBN, etc.

$$q(z; \phi) = \prod_{\text{unobserved variables } z_i} (\phi_i)^{z_i} (1 - \phi_i)^{(1 - z_i)}$$

Note: conditioned on the bottom part ($x$), choosing pixels independently in $z$ is not a terrible approximation.
The Evidence Lower bound

\[
\log p(x; \theta) \geq \sum_z q(z; \phi) \log p(z, x; \theta) + H(q(z; \phi)) = \mathcal{L}(x; \theta, \phi) + D_{KL}(q(z; \phi) \parallel p(z|x; \theta))
\]

The better \( q(z; \phi) \) can approximate the posterior \( p(z|x; \theta) \), the smaller \( D_{KL}(q(z; \phi) \parallel p(z|x; \theta)) \) we can achieve, the closer ELBO will be to \( \log p(x; \theta) \)
The Variational EM Algorithm

- \( q(z; \phi) \) is an arbitrary probability distribution over \( z \)

\[
D_{KL}(q(z; \phi) \| p(z|x; \theta)) = -\sum_z q(z; \phi) \log p(z, x; \theta) + \log p(x; \theta) - H(q(z; \phi))
\]

- Re-arranging can rewrite as

\[
H(q(z; \phi)) + \sum_z q(z; \phi) \log p(z, x; \theta) = \log p(x; \theta) - D_{KL}(q(z; \phi) \| p(z|x; \theta)) = F[\phi, \theta]
\]

- Variational EM as coordinate ascent on \( F[\phi, \theta] \)
  1. Initialize \( \theta^{(0)} \)
  2. \( \phi^{(1)} = \arg \max_\phi F[\phi, \theta^{(0)}] \neq p(z|x; \theta^{(0)}) \) in general
  3. \( \theta^{(1)} = \arg \max_\theta F[\phi^{(1)}, \theta] \) (can do Monte Carlo!)
  4. \( \phi^{(2)} = \arg \max_\phi F[\phi, \theta^{(1)}] \neq p(z|x; \theta^{(1)}) \) in general
  5. ...

- Unlike EM, variational EM not guaranteed to reach a local maximum. Marginal likelihood might decrease!

Stefano Ermon, Aditya Grover (AI Lab) Deep Generative Models Lecture 5
Potential issues with Variational EM algorithm

\[ L(\theta_n) = l(\theta_n | \theta_n) \]

(Figure adapted from tutorial by Sean Borman)
Latent Variable Models Pros:
- Easy to build flexible models
- Suitable for unsupervised learning

Latent Variable Models Cons:
- Hard to evaluate likelihoods
- Hard to train via maximum-likelihood
- Fundamentally, the challenge is that posterior inference $p(z | x)$ is hard. Typically requires variational approximations

Alternative: give up on KL-divergence and likelihood (GANs)