CS 4803 / 7643: Deep Learning

Topics:
- Variational Auto-Encoders (VAEs)
  - Reparameterization trick

Dhruv Batra
Georgia Tech
Administrativia

• HW4 Grades Released
  – Regrade requests close: 12/03, 11:55pm
  – Please check solutions first!

• Grade histogram: 7643
  – Max possible: 100 (regular credit) + 40 (extra credit)
Administrativia

• HW4 Grades Released
  – Regrade requests close: 12/03, 11:55pm
  – Please check solutions first!

• Grade histogram: 4803
  – Max possible: 100 (regular credit) + 40 (extra credit)
Recap from last time
Variational Autoencoders (VAE)
So far...

PixelCNNs define tractable density function, optimize likelihood of training data:

$$p_\theta(x) = \prod_{i=1}^{n} p_\theta(x_i|x_1, \ldots, x_{i-1})$$

VAEs define intractable density function with latent $z$:

$$\underbrace{p_\theta(x)} = \int p_\theta(z)p_\theta(x|z)dz$$

$z$ continuous

$$\sum_z p_\theta(z)p_\theta(x|z)$$

$z$ discrete

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Variational Auto Encoders

VAEs are a combination of the following ideas:

1. Auto Encoders

2. Variational Approximation
   • Variational Lower Bound / ELBO

3. Amortized Inference Neural Networks

4. “Reparameterization” Trick
Train such that features can be used to reconstruct original data

L2 Loss function:

$$\| x - \hat{x} \|^2$$

Doesn’t use labels!

**Encoder**:
4-layer conv

**Decoder**:
4-layer upconv

Autoencoders

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Autoencoders can reconstruct data, and can learn features to initialize a supervised model.

Features capture factors of variation in training data. Can we generate new images from an autoencoder?
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

\[ p(z|x) \]

Encoder: \( q_\phi(z|x) \)

Data: \( x \)

Decoder: \( p_\theta(x|z) \)

Reconstruction: \( \tilde{x} \)

Image Credit: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/
Variational Auto Encoders

VAEs are a combination of the following ideas:

1. Auto Encoders

2. Variational Approximation
   - Variational Lower Bound / ELBO

3. Amortized Inference Neural Networks

4. “Reparameterization” Trick
Key problem

\[ P(z|x) = \frac{P(z, x)}{P(x)} = \frac{p(x|z) p(z)}{\sum_{z} p(x|z) p(z)} \]
What is Variational Inference?

• Key idea
  – Reality is complex
  – Can we approximate it with something “simple”?
  – Just make sure simple thing is “close” to the complex thing.
Intuition

\[ \min_{q \in \mathcal{Y}} ~ d(p, q) \]

\[ KL(p(2) || q(2)) \]

\[ KL(q(2) || p(2)) \]

\[ KL(p(2) || q(2)) = \sum_{z} p(2) \log \frac{p(2)}{q(2)} \]

\[ = \sum_{z} q(2) \log \frac{q(2)}{p(2)} \]
The general learning problem with missing data

- Marginal likelihood – $\mathbf{x}$ is observed, $\mathbf{z}$ is missing:

$$ll(\theta : D) = \log \prod_{i=1}^{N} P(x_i \mid \theta)$$

$$= \sum_{i=1}^{N} \log P(x_i \mid \theta)$$

$$= \sum_{i=1}^{N} \log \sum_{z} P(x_i, z \mid \theta)$$

$$= \sum_{i=1}^{N} \log \sum_{z} P(x_i \mid \theta) P(z_i \mid x_i, \theta)$$

$$\log \sum_{2} \log \left\{ P(z_i \mid x_i, \theta) P(x_i \mid \theta) \right\}$$
Jensen’s inequality

- Use: $\log \sum_z P(z) \ g(z) \geq \sum_z P(z) \ \log \ g(z)$
Applying Jensen’s inequality

- Use: \( \log \sum_z P(z) g(z) \geq \sum_z P(z) \log g(z) \)

\[ l(\theta) = \log P(\tilde{x}_i \mid \theta) = \log \mathbb{E}_z \left[ \frac{P(\tilde{x}_i, z \mid \theta)}{Q_i(z)} \right] \]

\[ l(\theta) \geq F(\theta, Q_i) \]

\[ \max_{\theta, Q_i} F \]

"Free Energy" \( F(\theta, Q_i) \)

Variational Lower Bound

Evidence Lower Bound (ELBO)
Evidence Lower Bound

- Define potential function $F(\theta, Q)$:

$$l(\theta : D) \geq F(\theta, Q_i) = \sum_{i=1}^{N} \sum_{z} Q_i(z) \log \frac{P(x_i, z | \theta)}{Q_i(z)}$$
ELBO: Factorization #1 (GMMs)

\[ ll(\theta : D) \geq F(\theta, Q_i) = \sum_{i=1}^{N} \sum_{z} Q_i(z) \log \frac{P(x_i; \theta) P(z_i|x_i, \theta)}{Q_i(z)} \]

\[ = \left[ \sum Q_i(z) \log P(x_i; \theta) \right] + \left[ \sum Q_i(z) \log \frac{P(z_i|x_i, \theta)}{Q_i(z)} \right] \]

\[ F(\theta, Q_i) = \log P(x_i; \theta) - KL \left( \sum Q_i(z) \right) \]

\[ ll(\theta) = F(\theta, Q_i) + KL \left( \sum Q_i(z) \right) \]
ELBO: Factorization #2 (VAEs)

$$l_{\text{ELBO}}(\theta : D) \geq \max_{\theta, Q_i} F(\theta, Q_i) = \sum_{i=1}^{N} \sum_{z} Q_i(z) \log \frac{P(x_i, z | \theta)}{Q_i(z)}$$

$$= \sum_{z} Q_i(z) \log p(z_i | z, \theta) + \sum_{z} Q_i(z) \log \frac{p(z_i | \theta)}{Q_i(z)}$$

"Regulariser" Be simple

"Explain the data"
Variational Auto Encoders

VAEs are a combination of the following ideas:

1. Auto Encoders

2. Variational Approximation
   • Variational Lower Bound / ELBO

3. Amortized Inference Neural Networks

4. “Reparameterization” Trick
Amortized Inference Neural Networks

\[ Q_i(z) = \begin{pmatrix} 0.3 \\ 0.3 \\ 0.3 \end{pmatrix} \]

\[ z \sim \mathcal{N}(\mu_{2|x_i}, \Sigma_{2|x_i}) \]

\[ \mathcal{N}(\cdot, \cdot) \]

\[ x_i \rightarrow \text{NN} \rightarrow q_{\phi}(z|x_i) \]

\[ \text{softmax} \]

(C) Dhruv Batra
VAEs

(C) Dhruv Batra

Image Credit: https://www.kaggle.com/rvislaywade/visualizing-mnist-using-a-variational-autoencoder
Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Encoder $q_{\phi}(z|x)$

Decoder $p_{\theta}(x|z)$

Data: $x$

Reconstruction: $\tilde{x}$

Image Credit: https://jaan.io/what-is-variational-autoencoder-vae-tutorial/
Putting it all together: maximizing the likelihood lower bound

$$F(\theta, \phi) = \mathbb{E}_z \left[ \log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) \| p_{\theta}(z))$$
Variational Auto Encoders

Putting it all together: maximizing the likelihood lower bound

\[
\mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z))
\]

Make approximate posterior distribution close to prior

Encoder network

\[ q_\phi(z | x) \]

\[ \mu_{z|x} \]

\[ \Sigma_{z|x} \]

Input Data

\[ x \]
Putting it all together: maximizing the likelihood lower bound

\[
L(x^{(i)}, \theta, \phi) = \mathbb{E}_z \left[ \log p_\theta(x^{(i)} \mid z) \right] - D_{KL}(q_\phi(z \mid x^{(i)}) \parallel p_\theta(z))
\]

Make approximate posterior distribution close to prior

Encoder network:

\[ q_\phi(z \mid x) \]

Sample z from:

\[ z \mid x \sim \mathcal{N}(\mu_z \mid x, \Sigma_z \mid x) \]
Variational Auto Encoders

Putting it all together: maximizing the likelihood lower bound

\[
E_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) || p_\theta(z))
\]

\[\mathcal{L}(x^{(i)}, \theta, \phi)\]

Encoder network
\[q_\phi(z | x)\]

Decoder network
\[p_\theta(x | z)\]

Sample \(z\) from
\[z | x \sim \mathcal{N}(\mu_z|x, \Sigma_z|x)\]

Make approximate posterior distribution close to prior

Input Data
\[x\]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Putting it all together: maximizing the likelihood lower bound

\[
\mathbb{E}_z \left[ \log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) \| p_{\theta}(z))
\]

Make approximate posterior distribution close to prior

Maximize likelihood of original input being reconstructed

Sample \( z \) from \( z | x \sim \mathcal{N}(\mu_{z | x}, \Sigma_{z | x}) \)

Sample \( x | z \) from \( x | z \sim \mathcal{N}(\mu_{x | z}, \Sigma_{x | z}) \)

Input Data

Encoder network
\( q_{\phi}(z | x) \)

Decoder network
\( p_{\theta}(x | z) \)

Putting it all together: maximizing the likelihood lower bound

Variational Auto Encoders

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Variational Auto Encoders: Generating Data

Use decoder network. Now sample \( z \) from prior!

\[
\hat{x} \\
\text{Sample } x|z \text{ from } x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})
\]

Decoder network
\[ p_\theta(x|z) \]

Sample \( z \) from \( z \sim \mathcal{N}(0, I) \)

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Data manifold for 2-d \( z \)

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Variational Auto Encoders: Generating Data

Diagonal prior on $z$
=> independent latent variables

Different dimensions of $z$
encode interpretable factors of variation

Kingma and Welling, “Auto-Encoding Variational Bayes”, ICLR 2014

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Plan for Today

• VAEs
  – Reparameterization trick
Variational Auto Encoders

VAEs are a combination of the following ideas:

1. Auto Encoders

2. Variational Approximation
   - Variational Lower Bound / ELBO

3. Amortized Inference Neural Networks

4. “Reparameterization” Trick
Variational Auto Encoders

Putting it all together: maximizing the likelihood lower bound

\[
E_z \left[ \log p_\theta(x^{(i)} \mid z) \right] - D_{KL}(q_\phi(z \mid x^{(i)}) \mid \mid p_\theta(z))
\]

\( \mathcal{L}(x^{(i)}, \theta, \phi) \)

Maximize likelihood of original input being reconstructed

Sample \( x \mid z \sim \mathcal{N}(\mu_{x \mid z}, \Sigma_{x \mid z}) \)

Sample \( z \mid x \sim \mathcal{N}(\mu_{z \mid x}, \Sigma_{z \mid x}) \)

Make approximate posterior distribution close to prior

Encoder network

\( q_\phi(z \mid x) \)

Decoder network

\( p_\theta(x \mid z) \)

Input Data

\( \mathcal{X} \)
Putting it all together: maximizing the likelihood lower bound

$$\max_{\theta, \phi} \mathcal{L}(x^{(i)}, \theta, \phi)$$

$$\mathbb{E}_{z \sim q_{\phi}(z | x^{(i)})} \left[ \log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) \| p_{\theta}(z))$$
Basic Problem

\[ z \sim \text{Cat}(\pi) \]

\[ E_{z \sim p_\theta(z)} [f(z)] \]
Basic Problem

- Goal

$$\min_{\theta} \mathbb{E}_{z \sim p_{\theta}(z)}[f(z)]$$
Basic Problem

- Goal

\[ \theta(t+1) = \theta(t) - h \nabla_\theta \mathbb{E}_{z \sim p_\theta(z)} [f(z)] \]

- Need to compute:

\[ f_\theta(z) \quad p(z) \]

\[ \int \nabla_\theta f_\theta(z) \, p(z) \, dz \]

\[ = \mathbb{E} \left[ \nabla_\theta f_\theta(z) \right] \]

\[ \approx \frac{1}{N} \sum_{i=1}^{N} \nabla_\theta f(\mathbf{z}_i) \quad z_i \sim p(z) \]

\[ \nabla_\theta \mathbb{E}_{z \sim p_\theta(z)} [f(z)] \]

\[ \nabla_\theta \int f(z) \, p_\theta(z) \, dz \]

\[ \int \nabla_\theta f(z) \, p_\theta(z) \, dz \]

\[ \int f(z) \, \nabla_\theta p_\theta(z) \, dz \]

(C) Dhruv Batra
Basic Problem

- Need to compute: \[ \nabla_{\theta} \mathbb{E}_{z \sim p_{\theta}(z)} [f(z)] \]
Example

\[ z \sim N(\theta, 1) \]

\[ f(z) = \frac{z^2}{\theta^2} \int_{-\theta}^{\theta} e^{-\frac{z^2}{2\theta^2}} \, dz \]

\[ \min_{\theta} E[z^2] = \frac{\text{Var}(z)}{1} + \frac{\theta^2}{4} \]

\[ \text{Var}(z) = E[(z-\theta)^2] = E[z^2] - \theta^2 \]

"hard"

"easy"
Does this happen in supervised learning?

- Goal

\[
\min_{\theta} \mathbb{E}_{z \sim p_\theta(z)} [f(z)]
\]

\[
\min_{\theta} \mathbb{E}_{x, y \sim P_{\text{data}}} \left[ l(y, g(x, \theta)) \right]
\]

\[
\frac{1}{N} \sum_{i=1}^{N} l(y_i, g_i(x_i, \theta))
\]
But what about other kinds of learning?

- Goal

\[ \min_{\theta} \mathbb{E}_{z \sim p_{\theta}(z)}[f(z)] \]

UL \quad \max_{\theta, \phi} \quad \min_{\theta} \quad \mathbb{E}_{z \sim q_{\phi}(z)}[\log p(z \mid \theta, \phi)] 

RL \quad \max_{\theta} \quad \mathbb{E}_{a_t \sim \pi_{\phi}(a_t \mid s_t)} \left[ \sum_{t=1}^{T} r_t(s_t, a_t) \right] 

(C) Dhruv Batra
Two Options

1. Score Function based Gradient Estimator aka REINFORCE (and variants)

\[
\nabla_\theta \mathbb{E}_z [f(z)] = \mathbb{E}_z [f(z) \nabla_\theta \log p_\theta(z)]
\]

2. Path Derivative Gradient Estimator aka “reparameterization trick”

\[
\frac{\partial}{\partial \theta} \mathbb{E}_{z \sim p_\theta} [f(z))] = \frac{\partial}{\partial \theta} \mathbb{E}_\epsilon [f(g(\theta, \epsilon))] = \mathbb{E}_{\epsilon \sim p_\epsilon} \left[ \frac{\partial f}{\partial g} \frac{\partial g}{\partial \theta} \right]
\]
Option 1

- Score Function based Gradient Estimator aka REINFORCE (and variants)

\[ \nabla_\theta \mathbb{E}_z [f(z)] = \mathbb{E}_z [f(z) \nabla_\theta \log p_\theta(z)] \]

\[
= \int f(z) P_\theta(z) \, dz = \int f(z) \underbrace{\nabla_\theta \log P_\theta(z)}_{\sim \frac{1}{N}} \cdot P_\theta(z) \, dz
\]

\[
= \mathbb{E} \left[ f(z) \nabla_\theta \log P_\theta(z) \right]
\]

(C) Dhruv Batra
Recall: Policy Gradients

\[ \nabla_{\theta} J(\theta) = \nabla_{\theta} \mathbb{E}_{\tau \sim p_{\theta}(\tau)}[R(\tau)] \]

\[ = \nabla_{\theta} \int \pi_{\theta}(\tau) R(\tau) d\tau \]

\[ = \int \nabla_{\theta} \pi_{\theta}(\tau) R(\tau) d\tau \]

\[ = \int \nabla_{\theta} \pi_{\theta}(\tau) \cdot \frac{\pi_{\theta}(\tau)}{\pi(\tau)} \cdot R(\tau) d\tau \]

\[ = \int \pi_{\theta}(\tau) \nabla_{\theta} \log \pi_{\theta}(\tau) R(\tau) d\tau \]

\[ = \mathbb{E}_{\tau \sim p_{\theta}(\tau)}[\nabla_{\theta} \log \pi_{\theta}(\tau) R(\tau)] \]

Expand expectation

Exchange integration and expectation

\[ \nabla_{\theta} \log \pi(\tau) = \frac{\nabla_{\theta} \pi(\tau)}{\pi(\tau)} \]
Example

\[ Z \sim N(\theta, 1) \]

\[ P_0(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(z-\theta)^2}{2}} \]

\[ \frac{\partial}{\partial \theta} \left[ \log P_0(z) \right] = \left[ -\frac{(z-\theta)^2}{2} - \frac{1}{2} \log 2\pi \right] \]

\[ \text{Gradient Estimate} \]

\[ E_z \left[ 2(z - \theta) \right] \]

\[ \gamma \frac{1}{N} \sum_{i=1}^{N} Z_i (z - \theta) \]

\[ Z_i \sim N(\theta, 1) \]
Mental Break!

• VAE Demo
Two Options

1. Score Function based Gradient Estimator aka REINFORCE (and variants)

\[ \nabla_\theta \mathbb{E}_z \left[ f(z) \right] = \mathbb{E}_z \left[ f(z) \nabla_\theta \log p_\theta(z) \right] \]

2. Path Derivative Gradient Estimator aka “reparameterization trick”

\[ \frac{\partial}{\partial \theta} \mathbb{E}_{z \sim p_\theta} \left[ f(z) \right] = \frac{\partial}{\partial \theta} \mathbb{E}_\epsilon \left[ f(g(\theta, \epsilon)) \right] = \mathbb{E}_{\epsilon \sim p_\epsilon} \left[ \frac{\partial f}{\partial g} \frac{\partial g}{\partial \theta} \right] \]
Option 2

• Path Derivative Gradient Estimator aka “reparameterization trick”

$$\frac{\partial}{\partial \theta} \mathbb{E}_{z \sim p_\theta} [f(z))] = \frac{\partial}{\partial \theta} \mathbb{E}_\epsilon [f(g(\theta, \epsilon))] = \mathbb{E}_{\epsilon \sim p_\epsilon} \left[ \frac{\partial f}{\partial g} \frac{\partial g}{\partial \theta} \right]$$

\[ Z \sim p_\theta(z) \implies Z = g(\Theta, \epsilon) \]

\[ Z \sim N(\mu, \sigma^2) \quad \epsilon \sim N(0,1) \]

\[ Z = \Theta \mu + \sigma \epsilon \]

\[ g(\Theta, \epsilon) \]
Option 2

- Path Derivative Gradient Estimator aka “reparameterization trick”

\[
\frac{\partial}{\partial \theta} \mathbb{E}_{z \sim p_\theta} [f(z))] = \frac{\partial}{\partial \theta} \mathbb{E}_\epsilon [f(g(\theta, \epsilon))] = \mathbb{E}_{\epsilon \sim p_\epsilon} \left[ \frac{\partial f}{\partial \epsilon} \frac{\partial g}{\partial \theta} \right]
\]

\[
\frac{\partial}{\partial \theta} \mathbb{E}_{z \sim p_\theta(z)} [f(z)] = \frac{\partial}{\partial \theta} \mathbb{E}_{\epsilon \sim p_\epsilon} \left[ f(g(\theta, \epsilon)) \right]
\]

\[
= \mathbb{E}_\epsilon \left[ \frac{\partial}{\partial \theta} f(g(\theta, \epsilon)) \right]
\]

\[
= \mathbb{E}_\epsilon \left[ \frac{\partial f}{\partial \theta} \cdot \frac{\partial g}{\partial \theta} \right]
\]
Reparameterization Intuition

\[ z = \mu + \sigma^2 \epsilon_i \]

\( \epsilon_i \sim p(\epsilon) \)

Figure Credit: http://blog.shakirm.com/2015/10/machine-learning-trick-of-the-day-4-reparameterisation-trick/
Reparameterization Intuition

Inference

\( \mathcal{N}(0,1) \)

Generative

Image Credit: https://www.kaggle.com/rvislaywade/visualizing-mnist-using-a-variational-autoencoder
Example

\[ Z \sim N(\theta, 1) \]

\[ f(z) = z^2 \]

\[ \min_{\theta} E_z [z^2] \]

\[ \frac{\partial}{\partial \theta} E_\varepsilon [(\theta + \varepsilon)^2] \]

\[ = E_\varepsilon \left[ \frac{\partial}{\partial \theta} (\theta + \varepsilon)^2 \right] \]

\[ = E_\varepsilon \left[ 2(\theta + \varepsilon) \cdot 1 \right] \]

\[ \approx \frac{1}{N} \sum_{i=1}^{N} (\theta + \varepsilon_i) \]

\[ E_\varepsilon \left[ 2\theta + \varepsilon^2 \right] = 2\theta + 2E_\varepsilon [\varepsilon] \]

\[ \min_{\theta} \theta^2 \]

\[ Z = \theta + E \]

\[ E \sim N(0, 1) \]

\[ 3 \sim N(0, 1) \]
Two Options

1. Score Function based Gradient Estimator aka REINFORCE (and variants)
   \[ \nabla_{\theta} \mathbb{E}_{z} [f(z)] = \mathbb{E}_{z} [f(z) \nabla_{\theta} \log p_{\theta}(z)] \nabla_{\theta} \]

2. Path Derivative Gradient Estimator aka “reparameterization trick”
   \[ \frac{\partial}{\partial \theta} \mathbb{E}_{z \sim p_{\theta}} [f(z))] = \frac{\partial}{\partial \theta} \mathbb{E}_{\epsilon} [f(g(\theta, \epsilon))] = \mathbb{E}_{\epsilon \sim p_{\epsilon}} \left[ \frac{\partial f}{\partial g} \frac{\partial g}{\partial \theta} \right] \]
\[
\theta'(t) = \theta(t) - \eta \nabla \theta(t)
\]

```python
import numpy as np
N = 1000
theta = 2.0
x = np.random.randn(N) + theta
eps = np.random.randn(N)

grad1 = lambda x: np.sum(np.square(x)*(x-theta)) / x.size
grad2 = lambda eps: np.sum(2*(theta + eps)) / x.size

print grad1(x)
print grad2(eps)
```

4.46239612174  ≈ 2\theta
4.1840532024
Example

```python
Ns = [10, 100, 1000, 10000, 100000]
reps = 100

means1 = np.zeros(len(Ns))
vars1 = np.zeros(len(Ns))
means2 = np.zeros(len(Ns))
vars2 = np.zeros(len(Ns))

est1 = np.zeros(reps)
est2 = np.zeros(reps)

for i, N in enumerate(Ns):
    for r in range(reps):
        x = np.random.randn(N) + theta
        est1[r] = grad1(x)
        eps = np.random.randn(N)
        est2[r] = grad2(eps)

        means1[i] = np.mean(est1)
        means2[i] = np.mean(est2)
        vars1[i] = np.var(est1)
        vars2[i] = np.var(est2)

print means1
print means2
print vars1
print vars2
```

Figure Credit: http://gokererdogan.github.io/2016/07/01/reparameterization-trick/
Variational Auto Encoders

VAEs are a combination of the following ideas:

1. Auto Encoders

2. Variational Approximation
   • Variational Lower Bound / ELBO

3. Amortized Inference Neural Networks

4. “Reparameterization” Trick
Variational Auto Encoders

Putting it all together: maximizing the likelihood lower bound

$$\mathbb{E}_z \left[ \log p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) \| p_{\theta}(z))$$

Encoder network

$$q_{\phi}(z | x)$$

Input Data

$$x$$
Putting it all together: maximizing the likelihood lower bound

\[ \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z)) \]

\[ \mathcal{L}(x^{(i)}, \theta, \phi) \]

Make approximate posterior distribution close to prior

Encoder network

\[ q_\phi(z | x) \]

Input Data

\[ x \]

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Putting it all together: maximizing the likelihood lower bound

\[
\mathbb{E}_z \left[ \ln p_{\theta}(x^{(i)} | z) \right] - D_{KL}(q_{\phi}(z | x^{(i)}) \| p_{\theta}(z))
\]

\[
\mathcal{L}(x^{(i)}, \theta, \phi)
\]

Make approximate posterior distribution close to prior

Encoder network

Sample \( z \) from \( z | x \sim \mathcal{N}(\mu_z | x, \Sigma_z | x) \)

Input Data

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Variational Auto Encoders

Putting it all together: maximizing the likelihood lower bound

\[
\mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - D_{KL}(q_\phi(z | x^{(i)}) \| p_\theta(z))
\]

\[\mathcal{L}(x^{(i)}, \theta, \phi)\]

Make approximate posterior distribution close to prior

Encoder network
\[q_\phi(z | x)\]

Decoder network
\[p_\theta(x | z)\]

Sample \(z\) from \(z | x \sim \mathcal{N}(\mu_z | x, \Sigma_z | x)\)

Input Data
\[x\]