Topics:
- Variational Auto-Encoders (VAEs)
  - Reparameterization trick
- Generative Adversarial Networks (GANs)
Administrativia

• Project submission instructions released
  – Due: 12/04, 11:55pm
  – Last deliverable in the class
  – Can’t use late days
  – [https://piazza.com/class/jkujs03pgu75cd?cid=225](https://piazza.com/class/jkujs03pgu75cd?cid=225)
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 \( z \):

\[ p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz \]

Cannot optimize directly, derive and optimize lower bound on likelihood instead.

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
Autoencoders

Train such that features can be used to reconstruct original data

L2 Loss function:

\[ \| x - \hat{x} \|^2 \]

Encoder: 4-layer conv
Decoder: 4-layer upconv

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

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

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 need to make sure the simple thing is “close” to the complex thing.
Intuition

\[ \text{Model} \]

\[ \log p(z) \]

\[ \log q(z) \]

\[ \text{min}_q \text{KL}(p(z) \| q(z)) \]

\[ \text{KL}(p(z) \| q(z)) = \sum_z p(z) \log \frac{p(z)}{q(z)} \]

\[ - \frac{1}{\text{Sample}} \sum \log q(z) \]
The general learning problem with missing data

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

$$ll(\theta : \mathcal{D}) = \log \prod_{i=1}^{N} P(\mathbf{x}_i | \theta)$$

$$= \sum_{i=1}^{N} \log P(\mathbf{x}_i | \theta)$$

$$= \sum_{i=1}^{N} \log \sum_{\mathbf{z}} P(\mathbf{x}_i, \mathbf{z} | \theta)$$

$$= \log \left( \sum_{\mathbf{z}} P(\mathbf{z} | \mathbf{x}_i, \theta) \right)$$

$$= \log \left( \frac{1}{N} \sum_{i=1}^{N} P(\mathbf{x}_i | \theta) \right)$$
Applying Jensen’s inequality

• Use: \[ \log \sum_z P(z) g(z) \geq \sum_z P(z) \log g(z) \]
Applying Jensen’s inequality

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

- Use: \[ \log \sum_z P(z) g(z) \geq \sum_z P(z) \log g(z) \]
ELBO: Factorization #1

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

\[ \sum_{z} Q_i(z) \log \left( \frac{P(\bar{x}_i | \theta) P(z_1 \bar{x}_i, \theta)}{Q_i(z)} \right) \]

\[ = \sum_{z} Q_i(z) \log P(\bar{x}_i | \theta) + \left[ \sum_{z} Q_i(z) \log \frac{P(z_1 \bar{x}_i, \theta)}{Q_i(z)} \right] \]

\[ f(\theta | x_i) = \log P(\bar{x}_i | \theta) - \left[ \text{KL} \left( Q_i(z) \| P(z_1 \bar{x}_i, \theta) \right) \right] \]

\[ \max_{\theta, Q_i} E = l_l - \text{KL} \]
\[
\begin{align*}
\max_{\theta} F(\theta, Q_i) & : D)
\geq F(\theta, Q_i) = \sum_{i=1}^{N} \sum_{z} Q_i(z) \log \frac{P(x_i, z | \theta)}{Q_i(z)} \\
& = \sum_{i} Q_i(z) \log \frac{P(21 \theta) P(z_i | \theta)}{Q_i(z)} \\
& = \sum_{i} Q_i(z) \log P(z_i | \theta)
\end{align*}
\]
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

$\mathbf{X}_i 
\xrightarrow{\phi} \mathbf{Z}_i 
\xrightarrow{\text{Softmax}} \nu_{\phi}(\mathbf{Z}_i | \mathbf{X}_i)$

$\mathbf{Z}_i \sim \text{Cat}(\cdot)$

$\mathbf{Z}_i \in \mathbb{R}^k$

$\mathbf{Z}_i | \mathbf{X}_i \sim \mathcal{N}(\mu_{\phi}(\mathbf{X}_i), \Sigma_{\phi}(\mathbf{X}_i))$

$\mathbf{x}_i | \mathbf{Z}_i \sim \mathcal{G}(\mathbf{z}^{(i)}, \Sigma_{\phi}(\mathbf{X}_i))$
VAEs

![Diagram of Variational Autoencoder](image-url)
Variational Autoencoders

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

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

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

Variational Auto Encoders

Encoder network

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

Input Data

\[ x \]

\[ \mu_z | x \]

\[ \sum z | x \]

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

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

Make approximate posterior distribution close to prior

Encoder network

Input Data
Putting it all together: maximizing the likelihood lower bound

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

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

Make approximate posterior distribution close to prior

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

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

Input Data \( x \)
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
\[ z | x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x}) \]

\[ \mu_{x|z} \]

\[ \sum_{x|z} \]

Input Data

\[ \mathcal{X} \]

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

Make approximate posterior distribution close to prior

Maximize likelihood of original input being reconstructed

Variational Auto Encoders

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Use decoder network. Now sample z from prior!

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

Decoder network $p_\theta(x|z)$

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

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

• VAEs
  – Reparameterization trick

• Generative Adversarial Networks (GANs)
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
Basic Problem

- Goal

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

• Goal

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

• Need to compute:

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

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

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

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

\[
\int f(z) \nabla_{\theta} p_{\theta}(z) dz
\]
Does this happen in supervised learning?

\[
\mathbb{E}_{\mathbf{x}, y \sim P_{\text{data}}} \left[ \min_{\theta} \mathbb{E}_{z \sim p_{\theta}(z)} [f(z)] \right]
\]

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

\[
\approx \frac{1}{N} \sum_{i=1}^N \mathbb{E}_\theta l(y_i, \hat{y}(x_i, \theta))
\]
Example

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

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

\[ \min_{\theta} \mathbb{E}_z [f(z)] \]

\[ \min_{\theta} \int z^2 p(z) dz \]

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

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

\[ \mathbb{E}[z^2] = \theta^2 + \sqrt{\text{Var}(z)} \]
Example
Two Options

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

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

• 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)

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

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

\[ = \int f(z) \mathbb{E}_\theta [\nabla_\theta \log p_\theta(z) \cdot \frac{p_\theta(z)}{p_\theta(z)}] \, dz \]

\[ = \mathbb{E}_{z \sim p_\theta(z)} \left[ f(z) \nabla_\theta \log p_\theta(z) \right] \frac{1}{N} \sum_{i=1}^{N} z_i \]
Example

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

\[ \theta \]

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

\[ \frac{\partial}{\partial \theta} \]

\[ = \frac{2(z - \theta)}{2} \cdot (1) = (z - \theta) \]

\[ \nabla_0 = E \left[ z^2 (z - \theta) \right] \]

\[ \lambda = \frac{1}{N} \sum_{i=1}^{N} (z_i^2) (z_i - \theta) \]
Two Options

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

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

- 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 2

- Path Derivative Gradient Estimator aka “reparameterization trick”

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

\[Z \sim p_\theta(Z) \]
\[Z = g(\theta, \epsilon) \]
\[\epsilon \sim U(0,1) \]
\[\epsilon \sim N(0,1) \]
\[Z \sim N(\mu, \sigma^2) \]
\[Z = \mu + \sigma \epsilon \text{ (simple RV constant)} \]
Option 2

- Path Derivative Gradient Estimator aka “reparameterization trick”

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

\[
\frac{\partial}{\partial \theta} \int_0^1 f(g(\theta, \varepsilon)) \cdot p(\varepsilon) \cdot d\varepsilon \\
= \int_0^1 \frac{\partial}{\partial \theta} f(g(\theta, \varepsilon)) \cdot p(\varepsilon) \cdot d\varepsilon \\
= \int \frac{\partial f}{\partial g} \frac{\partial g}{\partial \theta} \cdot p(\varepsilon) \cdot d\varepsilon
\]
Reparameterization Intuition

\[ \epsilon_i \sim p(\epsilon) \]

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

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

\[ z \sim N(\theta, 1) \]
\[ z = \theta + \varepsilon \]
\[ f(z) = (\theta + \varepsilon)^2 \]

\[ E_\varepsilon \sim p(\varepsilon) \]
\[ \theta \]
\[ = E_\varepsilon \left[ \sum_0 (\theta + \varepsilon)^2 \right] \]
\[ = E_\varepsilon \left[ 2(\theta + \varepsilon) \right] = 2\theta + \bar{E_\varepsilon}[\varepsilon] = 0 \]
\[ \sum_{i=1}^n \frac{1}{N} \geq 2(\theta + \varepsilon) \]
Two Options

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

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

• Path Derivative Gradient Estimator aka “reparameterization trick”

\[ \frac{\partial}{\partial \theta} \mathbb{E}_{z \sim p_\theta} [f(z))] = \frac{\partial}{\partial \theta} \mathbb{E}_\epsilon [f(g(\theta, \epsilon))] = \mathbb{E}_{\epsilon \sim p_\epsilon} \left[ \frac{\partial f}{\partial g} \frac{\partial g}{\partial \theta} \right] \]
```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
4.1840532024 20
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
print vars1
print vars2
```

![Graph](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
Putting it all together: maximizing the likelihood lower bound

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

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

Input Data

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

\[ \mu_{z|x} \]

\[ \sum_{z|x} \]

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

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

Sample \( z \) from

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

Input Data

\[ \mathcal{X} \]

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

Putting it all together: maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior

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

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

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

Input Data
\[ \mathbf{x} \]

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