Topics:

- Variational Autoencoders
• **A4 grades slated for this weekend**

• **Projects!**
  • Due May 1\textsuperscript{rd} (May 3\textsuperscript{th} with grace period)
  • Cannot extend due to grade deadlines!

• **CIOS**
  • Please make sure to fill out! Let us know about things you liked and didn’t like in comments so that we can keep or improve!
  • [http://b.gatech.edu/cios](http://b.gatech.edu/cios)
Introduction
Spectrum of Low-Labeled Learning

Supervised Learning
- Train Input: \( \{X, Y\} \)
- Learning output: \( f : X \rightarrow Y, P(y|x) \)
- e.g. classification

Unsupervised Learning
- Input: \( \{X\} \)
- Learning output: \( P(x) \)
- Example: Clustering, density estimation, etc.
Generative Models

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks
$$p(x) = p(x_1)p(x_2|x_1)p(x_3|x_1) \prod_{i=1}^{n^2} p(x_i|x_1, \ldots, x_{i-1})$$

- **Training:**
  - We can train similar to language models: Teacher/student forcing
  - Maximum likelihood approach

- **Downsides:**
  - Slow sequential generation process
  - Only considers few context pixels

*Oord et al., Pixel Recurrent Neural Networks*
- Input can be a vector with (independent) Gaussian random numbers
- We can use a CNN to generate images!
Generative Adversarial Networks (GANs)

Vector of Random Numbers

Generator:
- Update weights to improve realism of generated images

Discriminator:
- Update weights to better discriminate

Mini-batch of real & fake data

Cross-entropy (Real or Fake?)
We know the answer (self-supervised)

Question: What loss functions can we use (for each network)?
Generative Adversarial Networks (GANs)

Vector of Random Numbers

Generator Loss
\[ \nabla \theta_g \frac{1}{m} \sum_{i=1}^{m} \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right). \]

Discriminator Loss
\[ \nabla \theta_d \frac{1}{m} \sum_{i=1}^{m} \left[ \log D \left( x^{(i)} \right) + \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right) \right]. \]

Mini-batch of real & fake data

Cross-entropy (Real or Fake?)
We know the answer (self-supervised)
Early Results

- Low-resolution images but look decent!
- Last column are nearest neighbor matches in dataset
- GANs are very difficult to train due to the mini-max objective

- Advancements include:
  - More stable architectures
  - Regularization methods to improve optimization
  - Progressive growing/training and scaling
Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.
Training GANs is difficult due to:
- Minimax objective – For example, what if generator learns to memorize training data (no variety) or only generates part of the distribution?
- Mode collapse – Capturing only some modes of distribution

Several theoretically-motivated regularization methods
- Simple example: Add noise to real samples!

\[
\lambda \cdot \mathbb{E}_{x \sim P_{real}, \delta \sim N_d(0, cI)} \left[ \left\| \nabla_x D_\theta (x + \delta) \right\| - k \right]^2
\]
Generative Adversarial Nets: Convolutional Architectures

Samples from the model look much better!

Radford et al, ICLR 2016
Generative Adversarial Nets: Convolutional Architectures

Interpolating between random points in latent space

Radford et al, ICLR 2016
Example Generated Images - BigGAN

Brock et al., Large Scale GAN Training for High Fidelity Natural Image Synthesis
Figure 4: Samples from our model with truncation threshold 0.5 (a-c) and an example of class leakage in a partially trained model (d).
https://www.youtube.com/watch?v=PCBTZh41Ris

Video Generation
A few other examples:

- Deep nostalgia: [https://www.myheritage.com/deep-nostalgia](https://www.myheritage.com/deep-nostalgia)
- High-resolution outputs: [https://compvis.github.io/taming-transformers/](https://compvis.github.io/taming-transformers/)
GANs

Don't work with an explicit density function
Take game-theoretic approach: learn to generate from training distribution through 2-player game

Pros:
- Beautiful, state-of-the-art samples!

Cons:
- Trickier / more unstable to train
- Can't solve inference queries such as p(x), p(z|x)

Active areas of research:
- Better loss functions, more stable training (Wasserstein GAN, LSGAN, many others)
- Conditional GANs, GANs for all kinds of applications
Mode Collapse

• Optimization of GANs is tricky
  – Not guaranteed to find Nash equilibrium

• Large number of methods to combat:
  – Use history of discriminators
  – Regularization
  – Different divergence measures
Application: Data Augmentation
Application: Domain Adaptation

- **Idea:** Train a model on *source* data and adapt to *target* data using unlabeled examples from target.
Approach

Table 2: Experimental results on unsupervised adaptation among MNIST, USPS, and SVHN.
Aside: Other ways to Align

[Image of two cups: digital SLR camera and low-cost camera, flash]

[Diagram showing a deep learning model with features as input, a feature extractor $G_f(\cdot; \theta_f)$, a gradient reversal layer, and a classifier $G_d(\cdot; \theta_d)$ with loss $L_d$ and class label $y$ with loss $L_y$.]

[Equations: $\frac{\partial L_y}{\partial \theta_f}$ and $\lambda \frac{\partial L_d}{\partial \theta_f}$ for forwardprop and backprop (and produced derivatives).]

[Ganin et al., JMLR 2016]
Generative Adversarial Networks (GANs) can produce amazing images!

Several drawbacks
- High-fidelity generation heavy to train
- Training can be unstable
- No explicit model for distribution

Larger number of extensions:
- GANs conditioned on labels or other information
- Adversarial losses for other applications
Variational Autoencoders (VAEs)
Generative Models

Goodfellow, NeurIPS 2016 Tutorial: Generative Adversarial Networks
Autoencoders

Encoder

Decoder

Minimize the difference (with MSE)

Low dimensional embedding

Linear layers with reduced dimension or Conv-2d layers with stride

Linear layers with increasing dimension or Conv-2d layers with bilinear upsampling
What is this?
Hidden/Latent variables
Factors of variation that produce an image:
(digit, orientation, scale, etc.)

\[ P(X) = \int P(X|Z; \theta)P(Z)dZ \]

- We cannot maximize this likelihood due to the integral
- Instead we maximize a variational lower bound (VLB) that we can compute

Kingma & Welling, Auto-Encoding Variational Bayes
We can combine the probabilistic view, sampling, autoencoders, and approximate optimization.

Just as before, sample $Z$ from simpler distribution.

We can also output parameters of a probability distribution!

**Example**: $\mu, \sigma$ of Gaussian distribution

For multi-dimensional version output diagonal covariance.

How can we maximize

$$P(X) = \int P(X|Z; \theta)P(Z)dZ$$
We can combine the probabilistic view, sampling, autoencoders, and approximate optimization.

- Given an image, estimate $Z$
- Again, output parameters of a distribution

Variational Autoencoder: Encoder
We can tie the encoder and decoder together into a probabilistic autoencoder.

- Given data ($X$), estimate $\mu_z, \sigma_z$ and sample from $N(\mu_z, \sigma_z)$
- Given $Z$, estimate $\mu_x, \sigma_x$ and sample from $N(\mu_x, \sigma_x)$
How can we optimize the parameters of the two networks?

Now equipped with our encoder and decoder networks, let’s work out the (log) data likelihood:

$$\log p_\theta(x^{(i)}) = \mathbf{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)$$
\[ \log p_{\theta}(x^{(i)}) = \mathbb{E}_{z \sim q_{\phi}(z|x^{(i)})} \left[ \log p_{\theta}(x^{(i)}) \right] \quad (p_{\theta}(x^{(i)}) \text{ Does not depend on } z) \]

\[ = \mathbb{E}_{z} \left[ \log \frac{p_{\theta}(x^{(i)} | z)p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \right] \quad \text{(Bayes' Rule)} \]

\[ = \mathbb{E}_{z} \left[ \log \frac{p_{\theta}(x^{(i)} | z)p_{\theta}(z)}{p_{\theta}(z | x^{(i)})} \frac{q_{\phi}(z | x^{(i)})}{q_{\phi}(z | x^{(i)})} \right] \quad \text{(Multiply by constant)} \]

\[ = \mathbb{E}_{z} \left[ \log p_{\theta}(x^{(i)} | z) \right] - \mathbb{E}_{z} \left[ \log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z)} \right] + \mathbb{E}_{z} \left[ \log \frac{q_{\phi}(z | x^{(i)})}{p_{\theta}(z | x^{(i)})} \right] \quad \text{(Logarithms)} \]
Aside: KL Divergence (distance measure for distributions), always $\geq 0$

$$KL(p||q) = H_c(p, q) - H(p) = \sum p(x) \log p(x) - \sum p(x) \log q(x)$$

**Definition of Expectation**

$$\mathbb{E}[f] = \mathbb{E}_{x \sim q}[f(x)] = \sum_{x \in \Omega} q(x) f(x)$$

$$KL(a||b) = \mathbb{E}[\log a(x)] - \mathbb{E}[\log b(x)] = \mathbb{E}[\log \frac{a(x)}{b(x)}]$$
\[
\log p_\theta(x^{(i)}) = E_{z \sim q_\phi(z | x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)
\]

\[
= E_z \left[ \log \frac{p_\theta(x^{(i)} | z) p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \quad \text{(Bayes’ Rule)}
\]

\[
= E_z \left[ \log \frac{p_\theta(x^{(i)} | z) p_\theta(z)}{p_\theta(z | x^{(i)})} \right] \cdot \frac{q_\phi(z | x^{(i)})}{q_\phi(z | x^{(i)})} \quad \text{(Multiply by constant)}
\]

\[
= E_z \left[ \log p_\theta(x^{(i)} | z) \right] - E_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z)} \right] + E_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)}
\]

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

The expectation wrt. \( z \) (using encoder network) let us write nice KL terms

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung
\[
\log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z \mid x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z)
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{\lambda(x^{(i)} \mid z)} \right] \quad \text{(Bayes' Rule)}
\]

\[
= \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} \mid z)p_\theta(z)}{p_\theta(z \mid x^{(i)})} \frac{q_\phi(z \mid x^{(i)})}{q_\phi(z \mid x^{(i)})} \right] \quad \text{(Multiply by constant)}
\]

\[
= \mathbb{E}_z \left[ \log p_\theta(x^{(i)} \mid z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z)} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z \mid x^{(i)})}{p_\theta(z \mid x^{(i)})} \right] \quad \text{(Logarithms)}
\]

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

Decoder network gives \(p_\theta(x \mid z)\), can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick. see paper.)

This KL term (between Gaussians for encoder and \(z\) prior) has nice closed-form solution!

\(p_\theta(z \mid x)\) intractable (saw earlier), can’t compute this KL term :( But we know KL divergence always \(\geq 0\).
\[ \log p_\theta(x^{(i)}) = \mathbb{E}_{z \sim q_\phi(z|x^{(i)})} \left[ \log p_\theta(x^{(i)}) \right] \quad (p_\theta(x^{(i)}) \text{ Does not depend on } z) \]

\[ = \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} q_\phi(z | x^{(i)}) \right] \quad \text{(Bayes’ Rule)} \]

\[ = \mathbb{E}_z \left[ \log \frac{p_\theta(x^{(i)} | z)p_\theta(z)}{p_\theta(z | x^{(i)})} \right] q_\phi(z | x^{(i)}) \quad \text{(Multiply by constant)} \]

\[ = \mathbb{E}_z \left[ \log p_\theta(x^{(i)} | z) \right] - \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] + \mathbb{E}_z \left[ \log \frac{q_\phi(z | x^{(i)})}{p_\theta(z | x^{(i)})} \right] \quad \text{(Logarithms)} \]

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

\[ \mathcal{L}(x^{(i)}, \theta, \phi) \geq 0 \]

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

Variational lower bound (“ELBO”)

\[ \theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^{N} \mathcal{L}(x^{(i)}, \theta, \phi) \]

Training: Maximize lower bound

\[ \text{Maximizing Likelihood} \]

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung
Putting it all together: maximizing the likelihood lower bound

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

To make approximate posterior distribution close to prior
Putting it all together: maximizing the likelihood lower bound

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

Sample from $Q(Z|X) \sim N(\mu_z, \sigma_z)$

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung
Putting it all together: maximizing the likelihood lower bound

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

Maximize likelihood of original input being reconstructed

Sample from \( P(X|Z; \theta) \sim N(\mu_x, \sigma_x) \)

From CS231n, Fei-Fei Li, Justin Johnson, Serena Yeung

Forward and Backward Passes
Problem with respect to the VLB: updating $\phi$

$$L_{VAE} = \mathbb{E}_{q_\phi(z|x)} \left[ \log \frac{p_\theta(z, x)}{q_\phi(z|x)} \right]$$

$$= -D_{KL}(q_\phi(z|x)||p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$$

$Z \sim Q(Z|X; \phi)$: need to differentiate through the sampling process w.r.t $\phi$ (encoder is probabilistic)
Solution: make the randomness independent of encoder output, making the encoder deterministic

Gaussian distribution example:

- Previously: encoder output = random variable $z \sim N(\mu, \sigma)$
- Now encoder output = distribution parameter $[\mu, \sigma]$
- $z = \mu + \epsilon \ast \sigma, \epsilon \sim N(0,1)$

From: Tutorial on Variational Autoencoders
https://arxiv.org/abs/1606.05908

From: http://gokererdogan.github.io/2016/07/01/reparameterization-trick/
Interpretability of Latent Vector

Kingma & Welling, Auto-Encoding Variational Bayes
Variational Autoencoders (VAEs) provide a principled way to perform approximate maximum likelihood optimization

- Requires some assumptions (e.g. Gaussian distributions)

- Samples are often not as competitive as GANs

- Latent features (learned in an unsupervised way!) often good for downstream tasks:
  - Example: World models for reinforcement learning (Ha et al., 2018)

*Ha & Schmidhuber, World Models, 2018*
Several ways to learn *generative* models via deep learning

**PixelRNN/CNN:**
- Simple tractable densities we can model via a NN and optimize
- Slow generation – limited scaling to large complex images

**Generative Adversarial Networks (GANs):**
- Pro: Amazing results across many image modalities
- Con: Unstable/difficult training process, computationally heavy for good results
- Con: Limited success for discrete distributions (language)
- Con: Hard to evaluate (implicit model)

**Variational Autoencoders:**
- Pro: Principled mathematical formulation
- Pro: Results in disentangled latent representations
- Con: Approximation inference, results in somewhat lower quality reconstructions

*Ha & Schmidhuber, World Models, 2018*