CS 4803 / 7643: Deep Learning

Topics:
  – Generative Adversarial Networks (GANs)
  – Reinforcement Learning (RL)

Dhruv Batra
Georgia Tech
• HW3 Grades Released
  – Max regular points: 62 (4803), 66 (7643)
  – Regrade requests close: 12/04, 11:55pm
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
Recap from last time
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:

\[
\nabla_\theta \mathbb{E}_{z \sim p_\theta(z)} [f(z)]
\]
Does this happen in supervised learning?

\[
\mathbb{E}_{\mathcal{X}, y \sim \mathcal{P}_{\text{data}}} \left[ \min_{\theta} \mathbb{E}_{z \sim p_\theta(z)}[f(z)] \right] \\
= \mathbb{E}_{\mathcal{X}, y \sim \mathcal{P}_{\text{data}}} \left[ \sum_{i=1}^{N} \sum_{\epsilon=1}^{n} \mathbb{E}_{\epsilon} l(y_i, \hat{y}(x_i, \theta)) \right]
\]
Example

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

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

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

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

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

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

\[ = \mathbb{E}[z^2] - \theta^2 \]

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

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

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

\[ = \int f(z) \nabla_\theta \log p_\theta(z) \sqrt{p_\theta(z)} \cdot \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_z[p_\theta(z)] \]
Example

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

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

\[ \frac{\partial}{\partial \Theta} \log P_0(z) = -\frac{2(z-\Theta)}{2} \]

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

\[ \nabla_\Theta = E \left[ z^2 (z-\Theta) \right] \]

\[ \underbrace{\frac{1}{N} \sum_{i=1}^{N} (z_i^2)}_{\text{}} (z_i - \Theta) \]
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”

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

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

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

\[Z \sim N(\mu, \sigma^2) \quad \text{and} \quad E \sim N(0,1)\]
\[\Rightarrow Z = \mu + \sigma E\]

simple RV
constants

(C) Dhruv Batra
Option 2

• Path Derivative Gradient Estimator aka “reparameterization trick”

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

\[
\frac{\partial}{\partial \theta} \int f(g(\theta, \epsilon)) \cdot \rho(\epsilon) \, d\epsilon
\]

\[
= \int \frac{\partial}{\partial \theta} f(g(\theta, \epsilon)) \cdot \rho(\epsilon) \, d\epsilon
\]

\[
= \int \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial \theta} \cdot \rho(\epsilon) \, d\epsilon
\]
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

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

\[ z = \theta + \epsilon \]

\[ f(z) = (\theta + \epsilon)^2 \]

\[ E_{\epsilon \sim N(0, 1)}[ (\theta + \epsilon)^2 ] \]

\[ \theta = E_{\epsilon} \left[ \sum_{i=1}^{N} (\theta + \epsilon_i)^2 \right] \]

\[ = \sum_{i=1}^{N} 2(\theta + \epsilon_i) \]

\[ \approx \frac{1}{N} \sum_{i=1}^{N} 2(\theta + \epsilon_i) \]
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
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
```

Figure Credit: http://gokererdogan.github.io/2016/07/01/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

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

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

$$\mathcal{X}$$

$$\mu_z | x$$

$$\Sigma_z | x$$

$$\mu x | z$$

$$\Sigma x | z$$

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Generative Adversarial Networks (GAN)
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.
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

What if we give up on explicitly modeling density, and just want ability to sample?

**GANs: don’t work with any explicit density function!**
Generative Adversarial Networks (GANs)

GANs are a combination of the following ideas:

1. Learning to Sample
   - Connection to Inverse Transform Sampling

2. Adversarial Training

3. “Reparameterization” Trick
Easy Interview Question

- I give you \( u \sim \text{U}(0,1) \)
- Produce a sample from \( \text{Bern}(\theta) \)

\[
\begin{align*}
& \text{if } u > \theta \\
& \quad \text{output } 0 \\
& \text{if } u < \theta \\
& \quad \text{output } 1
\end{align*}
\]
Slightly Harder Interview Question

- I give you $u \sim U(0,1)$
- Produce a sample from $\text{Cat} (\pi)$
Harder Interview Question

- I give you \( u \sim U(0,1) \)
- Produce a sample from \( F_X(x) \)
Generative Adversarial Networks

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g., random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?
Problem: Want to sample from complex, high-dimensional training distribution. No direct way to do this!

Solution: Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.

Q: What can we use to represent this complex transformation?

A: A neural network!

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

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

• (Finish) Generative Adversarial Networks (GANs)

• Reinforcement Learning
Generative Adversarial Networks (GANs)

GANs are a combination of the following ideas:

1. Learning to Sample
   - Connection to Inverse Transform Sampling

2. Adversarial Training

3. “Reparameterization” Trick
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission.

Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission.
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

\[
\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]
\]

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

Train jointly in **minimax game**

Minimax objective function:

\[
\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]
\]

- Discriminator ($\theta_d$) wants to **maximize objective** such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake)
- Generator ($\theta_g$) wants to **minimize objective** such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real)

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator
   $$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator
   $$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator

   $$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

   $$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

In practice, optimizing this generator objective does not work well!

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Training GANs: Two-player game

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Minimax objective function:
\[
\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]
\]

Alternate between:
1. **Gradient ascent** on discriminator
   \[
   \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]
   \]

2. **Instead**: **Gradient ascent** on generator, different objective
   \[
   \max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))
   \]

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Training GANs: Two-player game

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Alternate between:

1. **Gradient ascent** on discriminator
   $$\max_{\theta_d} \left[ \mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log (1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Instead: Gradient ascent** on generator, different objective
   $$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log (D_{\theta_d}(G_{\theta_g}(z)))$$

Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.

Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.

Aside: Jointly training two networks is challenging, can be unstable. Choosing objectives with better loss landscapes helps training, is an active area of research.

**Slide Credit:** Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Training GANs: Two-player game

**Generator network**: try to fool the discriminator by generating real-looking images

**Discriminator network**: try to distinguish between real and fake images

Fake images (from generator) → Generator Network → Random noise $z$ → Discriminator Network → Real or Fake

Real images (from training set) → Discriminator Network

After training, use generator network to generate new images

Fake and real images copyright Emily Denton et al. 2015. Reproduced with permission.

---

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014
GANs

• Demo
  – https://poloclub.github.io/ganlab/
Generative Adversarial Nets

Generated samples

Nearest neighbor from training set

Figures copyright Ian Goodfellow et al., 2014. Reproduced with permission.
Generative Adversarial Nets

Generated samples (CIFAR-10)

Nearest neighbor from training set

Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Figures copyright Ian Goodfellow et al., 2014. Reproduced with permission.
Generative Adversarial Nets: Convolutional Architectures


Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Generative Adversarial Nets: Convolutional Architectures

Samples from the model look much better!

Radford et al, ICLR 2016

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
Generative Adversarial Nets: Convolutional Architectures

Interpolating between random points in latent space

Radford et al, ICLR 2016
BigGAN

Large Scale GAN Training for High Fidelity Natural Image Synthesis
Andrew Brock, Jeff Donahue, Karen Simonyan  https://arxiv.org/abs/1809.11096
BigGAN

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).
BigGAN

https://gist.github.com/phillipi/d2921f2d4726d7e3cdac7a4780c6050a
### 2017: Explosion of GANs

#### "The GAN Zoo"

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorical GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWNN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- iCGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

https://github.com/hindupuravinash/the-gan-zoo

**Slide Credit:** Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n
2017: Explosion of GANs

“The GAN Zoo”

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorical GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters With Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWNN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- iCGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-rainging Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

See also: https://github.com/soumith/ganhacks for tips and tricks for trainings GANs

https://github.com/hindupuravinash/the-gan-zoo

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

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

• (Finish) Generative Adversarial Networks (GANs)

• Reinforcement Learning
Supervised Learning

**Data**: (x, y)  
x is data, y is label

**Goal**: Learn a *function* to map x -> y

**Examples**: Classification, regression, object detection, semantic segmentation, image captioning, etc.
Unsupervised Learning

**Data:** $x$
Just data, no labels!

**Goal:** Learn some underlying hidden *structure* of the data

**Examples:** Clustering, dimensionality reduction, feature learning, density estimation, etc.

Figure copyright Ian Goodfellow, 2016. Reproduced with permission.

1-d density estimation

2-d density estimation

2-d density images left and right are CC0 public domain
Types of Learning

• **Supervised learning**
  – Learning from a “teacher”
  – Training data includes desired outputs

• **Unsupervised learning**
  – Discover structure in data
  – Training data does not include desired outputs

• **Reinforcement learning**
  – Learning to act under evaluative feedback (rewards)
What is Reinforcement Learning?

- **Agent-oriented learning**—learning by interacting with an environment to achieve a goal
  - more *realistic* and *ambitious* than other kinds of machine learning

- Learning by trial and error, with only delayed evaluative feedback (reward)
  - the kind of machine learning most like natural learning
  - learning that can tell for itself when it is right or wrong

*Slide Credit: Rich Sutton*
Example: Hajime Kimura’s RL Robots

Before

Backward

After

New Robot, Same algorithm

Slide Credit: Rich Sutton
● Environment may be unknown, nonlinear, stochastic and complex

● Agent learns a policy mapping states to actions
  ○ Seeking to maximize its cumulative reward in the long run

Slide Credit: Rich Sutton
RL API

- At each step $t$ the agent:
  - Executes action $a_t$
  - Receives observation $o_t$
  - Receives scalar reward $r_t$

- The environment:
  - Receives action $a_t$
  - Emits observation $o_{t+1}$
  - Emits scalar reward $r_{t+1}$

$$q_t = f(1 | s_t)$$

$$r_t(s_t, a_t)$$

$$RL: \frac{s_{t+1}}{a_t} = x_t \frac{y_t}{y_t}$$

$$(y^*, \theta_t)$$
Signature challenges of RL

- Evaluative feedback (reward)
- Sequentiality, delayed consequences
  - Need for trial and error, to explore as well as exploit
  - Non-stationarity
    - $x, y \in \mathcal{D}$
  - The fleeting nature of time and online data

Slide Credit: Rich Sutton
Robot Locomotion

Objective: Make the robot move forward

State: Angle and position of the joints

Action: Torques applied on joints

Reward: 1 at each time step if upright + forward movement

Figures copyright John Schulman et al., 2016. Reproduced with permission.
Atari Games

Objective: Complete the game with the highest score

State: Raw pixel inputs of the game state

Action: Game controls e.g. Left, Right, Up, Down

Reward: Score increase/decrease at each time step

Figures copyright Volodymyr Mnih et al., 2013. Reproduced with permission.
**Objective**: Win the game!

**State**: Position of all pieces

**Action**: Where to put the next piece down

**Reward**: 1 if win at the end of the game, 0 otherwise
Demo

- [https://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html](https://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html)
Markov Decision Process

- Mathematical formulation of the RL problem

Defined by: \( (S, A, R, P, \gamma) \)

- \( S \): set of possible states
- \( A \): set of possible actions
- \( R \): distribution of reward given (state, action) pair
- \( P \): transition probability i.e. distribution over next state given (state, action) pair
- \( \gamma \): discount factor
Markov Decision Process

- Mathematical formulation of the RL problem

Defined by: \((S, A, R, P, \gamma)\)

- \(S\) : set of possible states
- \(A\) : set of possible actions
- \(R\) : distribution of reward given (state, action) pair
- \(P\) : transition probability i.e. distribution over next state given (state, action) pair
- \(\gamma\) : discount factor

- Life is trajectory: \(\ldots S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, \ldots\)
Markov Decision Process

- Mathematical formulation of the RL problem

Defined by: \((\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{P}, \gamma)\)

\[\begin{align*}
\mathcal{S} & : \text{set of possible states} \\
\mathcal{A} & : \text{set of possible actions} \\
\mathcal{R} & : \text{distribution of reward given (state, action) pair} \\
\mathbb{P} & : \text{transition probability i.e. distribution over next state given (state, action) pair} \\
\gamma & : \text{discount factor}
\end{align*}\]

- Life is trajectory: 

\[\ldots S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2}, \ldots \]

- **Markov property**: Current state completely characterizes the state of the world

\[p(r, s' | s, a) = \text{Prob} \left[ R_{t+1} = r, S_{t+1} = s' \mid S_t = s, A_t = a \right] \]
Components of an RL Agent

- **Policy**
  - How does an agent behave?

- **Value function**
  - How good is each state and/or state-action pair?

- **Model**
  - Agent’s representation of the environment
Policy

• A policy is how the agent acts

• Formally, map from states to actions

Deterministic policy: \( a = \pi(s) \)

Stochastic policy: \( \pi(a|s) = \mathbb{P}[A_t = a|S_t = s] \)
The optimal policy $\pi^*$

What’s a good policy?
The optimal policy $\pi^*$

What’s a good policy?

Maximizes current reward? Sum of all future reward?
The optimal policy $\pi^*$

What’s a good policy?

Maximizes current reward? Sum of all future reward?

Discounted future rewards!

$$\sum_{t} r_t + \sum_{t} \gamma^t r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3}.$$
What’s a good policy?

Maximizes current reward? Sum of all future reward?

Discounted future rewards!

Formally:

\[
\pi^* = \arg\max_{\pi} \mathbb{E}\left[ \sum_{t>0} \gamma^t r_t | \pi \right]
\]

with

\[
\begin{align*}
  s_0 &\sim p(s_0), & a_t &\sim \pi(\cdot | s_t), & s_{t+1} &\sim p(\cdot | s_t, a_t) \\
  a_t &= \pi(s_t) & S_{t+1} &= \mathbb{E}_{M(s_t, a_t)}
\end{align*}
\]
Value Function

- A value function is a prediction of future reward

- “State Value Function” or simply “Value Function”
  - How good is a state?
  - Am I screwed? Am I winning this game?

- “Action Value Function” or Q-function
  - How good is a state action-pair?
  - Should I do this now?
Definitions: Value function and Q-value function

Following a policy produces sample trajectories (or paths) \( s_0, a_0, r_0, s_1, a_1, r_1, \ldots \)
Definitions: Value function and Q-value function

Following a policy produces sample trajectories (or paths) $s_0, a_0, r_0, s_1, a_1, r_1, ...$

How good is a state?
The **value function** at state $s$, is the expected cumulative reward from state $s$ (and following the policy thereafter):

$$V^\pi(s) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t \mid s_0 = s, \pi \right]$$
Definitions: Value function and Q-value function

Following a policy produces sample trajectories (or paths) $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$

How good is a state?
The **value function** at state $s$, is the expected cumulative reward from state $s$ (and following the policy thereafter):

$$V^\pi(s) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t | s_0 = s, \pi \right]$$

How good is a state-action pair?
The **Q-value function** at state $s$ and action $a$, is the expected cumulative reward from taking action $a$ in state $s$ (and following the policy thereafter):

$$\max_a Q^\pi(s, a) = \mathbb{E} \left[ \sum_{t \geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi \right]$$

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

\[ a_t \quad \rightarrow \quad r_t \quad \rightarrow \quad a_t \]

observation

action

reward

(C) Dhruv Batra

Slide Credit: David Silver
Model

- Model predicts what the world will do next

Model is learnt from experience
Acts as proxy for environment
Planner interacts with model
e.g. using lookahead search

\[ s_t, a_t \rightarrow s_{t+1} \]

\[ P(s_{t+1} | s_t, a_t) \]