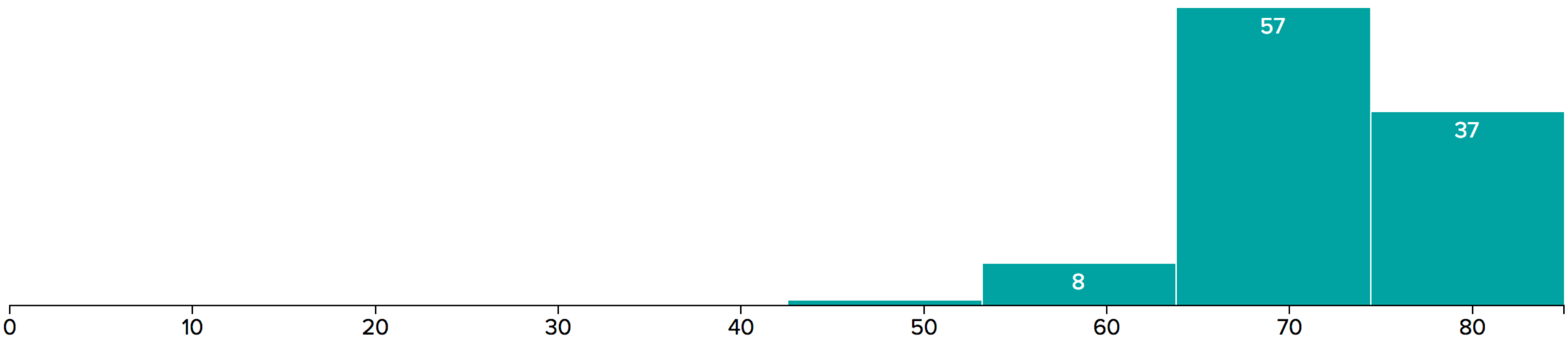
CS 4803 / 7643: Deep Learning

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

Dhruv Batra
Georgia Tech

Administrativia

- HW3 Grades Released
  - Max regular points: 62 (4803), 66 (7643)
  - Regrade requests close: 12/04, 11:55pm



MINIMUM	MEDIAN	MAXIMUM	MEAN	STD DEV
52.0	70.0	85.0	71.56	6.33

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

$$\boxed{\theta^{(t)} \leftarrow \theta^{(t)} - \nabla_\theta}$$

Basic Problem

- Goal

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

- Need to compute:

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

$$\nabla_\theta \int f_\theta(z)\, p(z)\, dz$$

$$\int \nabla_\theta f_\theta(z) \cdot p(z)\, dz$$

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

$$\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta f_\theta(z_i)$$

$$\nabla_\theta \int f(z)\, p_\theta(z)\, dz$$

$$\int \nabla_\theta \, f(z)\, p_\theta(z)\, dz$$

$$\int \cancel{\nabla_\theta} f(z)\, \nabla_\theta p_\theta(z)\, dz$$

Does this happen in supervised learning?

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

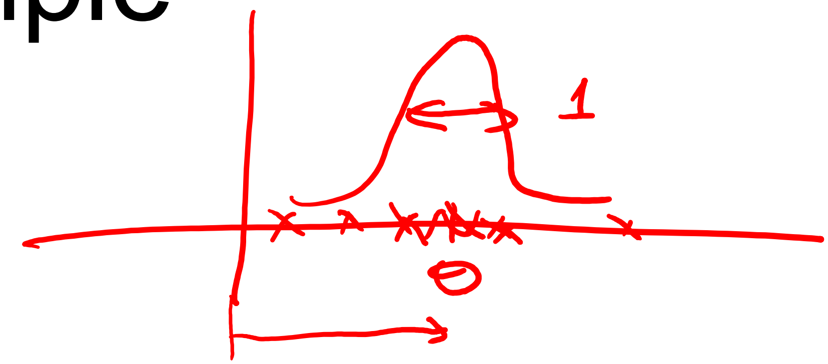
$$\min_{\theta} \mathbb{E}_{x,y \sim p_{data}} \left[\ell(y, \hat{y}(x,\theta)) \right]$$

$$= \mathbb{E}_{x,y \sim p_{data}} \left[\nabla_\theta \ell(\ldots \theta) \right]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \left[\nabla_\theta \ell(y_i, \hat{y}(x_i, \theta)) \right]$$

Example

$$z \sim N(\theta, 1)$$



$$f(z) = z^2$$

$$\min_{\theta} \; E_z[f(z)]$$

$$\min_{\theta} \nabla_{\theta} \int z^2 \, p(z) \, dz$$

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

$$Var(z) = E[(z-\theta)^2]$$

$$= E[z^2] - \theta^2$$

$$E[z^2] = \theta^2 + \boxed{Var(z)}$$

Two Options

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

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

- 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 1

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

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

$$\nabla_\theta \int f(z)\, p_\theta(z)\, dz$$

$$= \int f(z) \frac{\nabla_\theta p_\theta(z)}{p_\theta(z)} \cdot p_\theta(z)\, dz$$

$$= \int f(z) \left[\nabla_\theta \log p_\theta(z)\right] \cdot \boxed{p_\theta(z)} \cdot dz$$

$$= E_{z \sim p_\theta(z)}\left[f(z)\, \nabla_\theta \log p_\theta(z)\right] \approx \frac{1}{N}\sum$$

Example

$$P_\theta(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(z-\theta)^2}{2}}$$

$$\frac{\partial}{\partial\theta} \log P_\theta(z) = -\frac{(z-\theta)^2}{2} - \cancel{\frac{1}{2}\log 2\pi}$$

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

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

$$\approx \frac{1}{N} \sum_{i=1}^{N} (z_i^2)(z_i-\theta)$$

Two Options

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

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

- 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"

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

$z \sim p_\theta(z)$

$\boxed{z = g(\theta, \epsilon)}$

$\boxed{\epsilon \sim U(\cancel{0}, 1)}$

$\sim N(0,1)$

$z \sim N(\mu, \sigma^2)$

$\epsilon \sim N(0,1)$

$\rightarrow z = \boxed{\mu} + \sigma \epsilon \leftarrow$ simple RV

constants

Option 2

- Path Derivative Gradient Estimator aka "reparameterization trick"

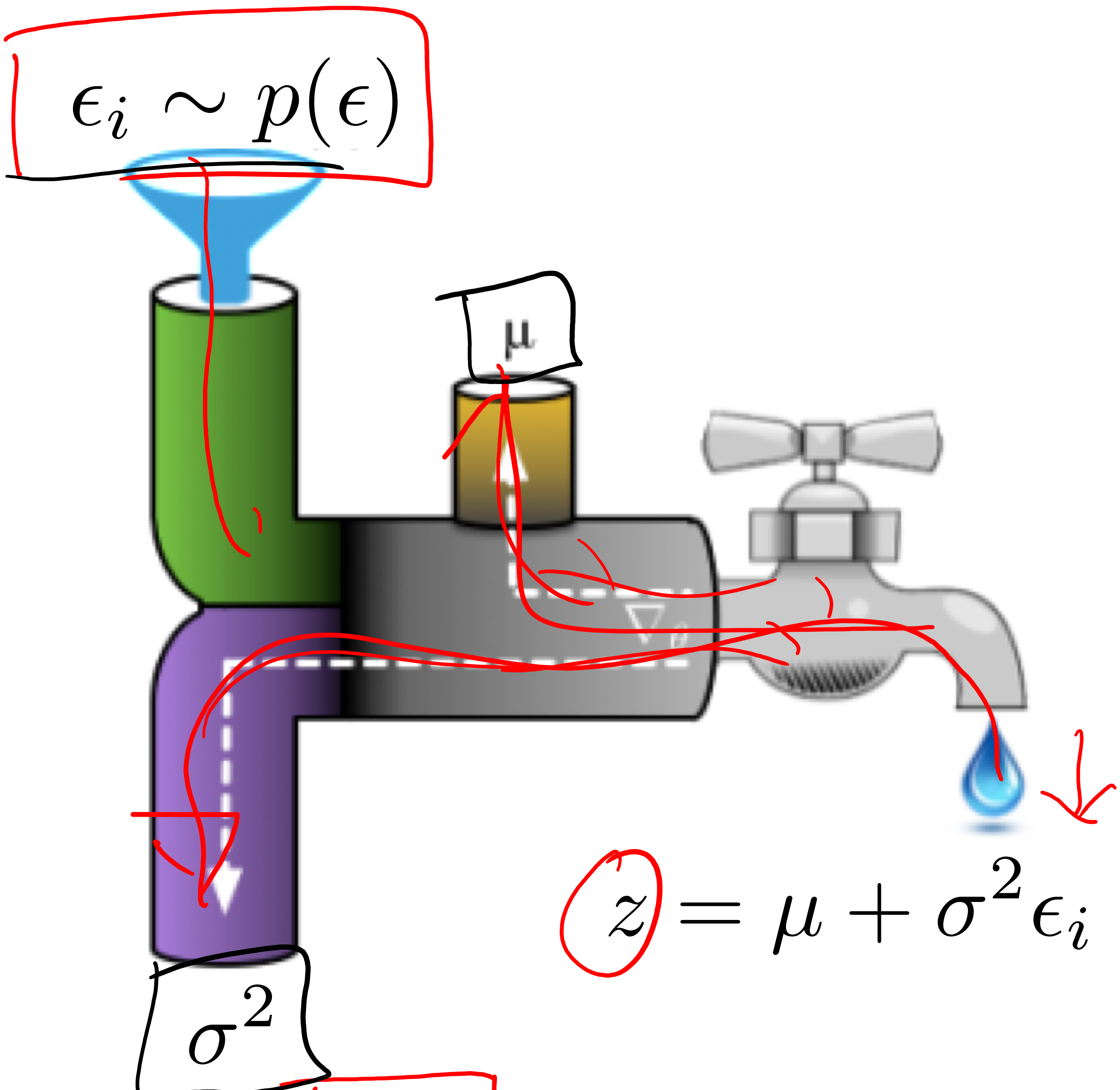
$$\frac{\partial}{\partial\theta}\mathbb{E}_{z\sim p_\theta}\left[f(z)\right] = \frac{\partial}{\partial\theta}\mathbb{E}_{\epsilon}\left[f(g(\theta,\epsilon))\right] = \boxed{\mathbb{E}_{\epsilon\sim p_\epsilon}}\left[\frac{\partial f}{\partial g}\frac{\partial g}{\partial\theta}\right] \approx \frac{1}{N}\sum(\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

$$\frac{\partial}{\partial\theta}\int f(g(\theta,\varepsilon))\cdot p(\varepsilon)\cdot d\varepsilon$$

$$= \int \underbrace{\frac{\partial}{\partial\theta} f(g(\theta,\varepsilon))}\cdot p(\varepsilon)\cdot d\varepsilon$$

$$= \int \frac{\partial f}{\partial g}\frac{\partial g}{\partial\theta}\cdot p(\varepsilon)\, d\varepsilon$$

Reparameterization Intuition



Example

$z \sim N(\theta, 1)$ $\quad$ $z = \theta + \varepsilon$ $\quad$ $\boxed{\varepsilon \sim N(0,1)}$

$f(z) = (\theta + \varepsilon)^2$

$$\underset{\theta}{\cancel{\min}}\ \nabla E_{\varepsilon \sim p(\varepsilon)}\left[(\theta+\varepsilon)^2\right]$$

$$= E_\varepsilon\left[\nabla_\theta (\theta+\varepsilon)^2\right]$$

$$= E_\varepsilon\left[2(\theta+\varepsilon)\right] = \underline{2\theta} + \boxed{E_\varepsilon[\varepsilon]}\ 0$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} 2(\theta+\varepsilon_i)$$

Two Options

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

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

- Path Derivative Gradient Estimator aka "reparameterization trick"

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

Example

```
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

```
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
```

```
[ 3.8409546   3.97298803  4.03007634  3.98531095  3.99579423]
[ 3.97775271  4.00232825  3.99894536  4.00353734  3.99995899]

[  6.45307927e+00   6.80227241e-01   8.69226368e-02   1.00489791e-02
   8.62396526e-04]
[  4.59767676e-01   4.26567475e-02   3.33699503e-03   5.17148975e-04
   4.65338152e-05]
```

REINFORCE

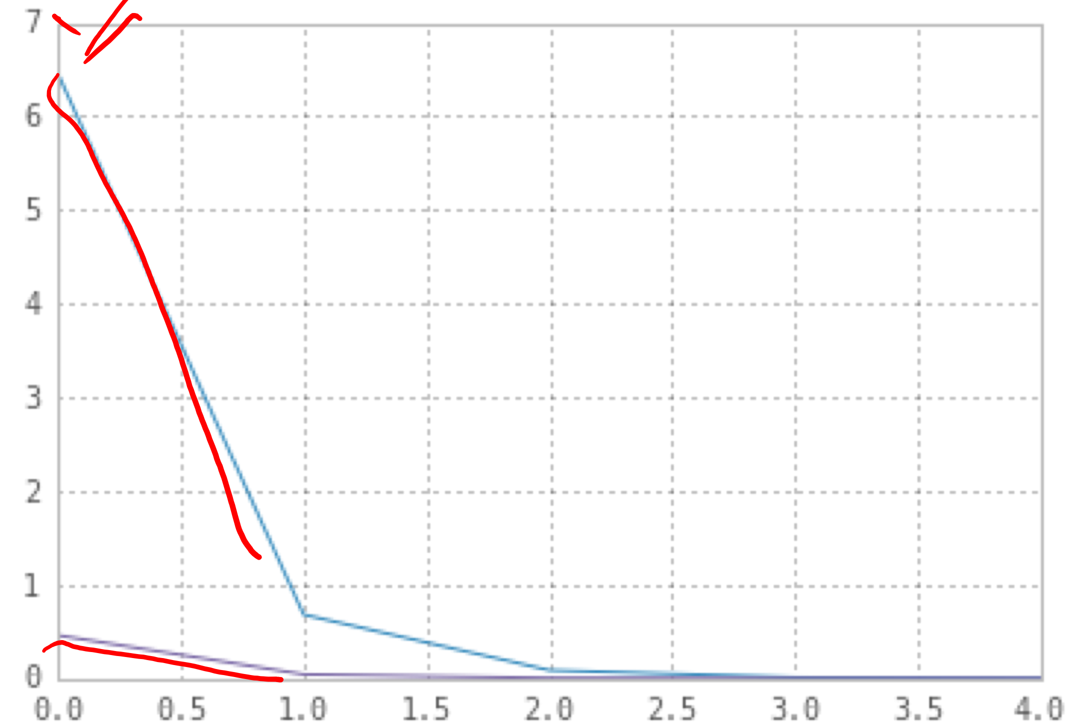


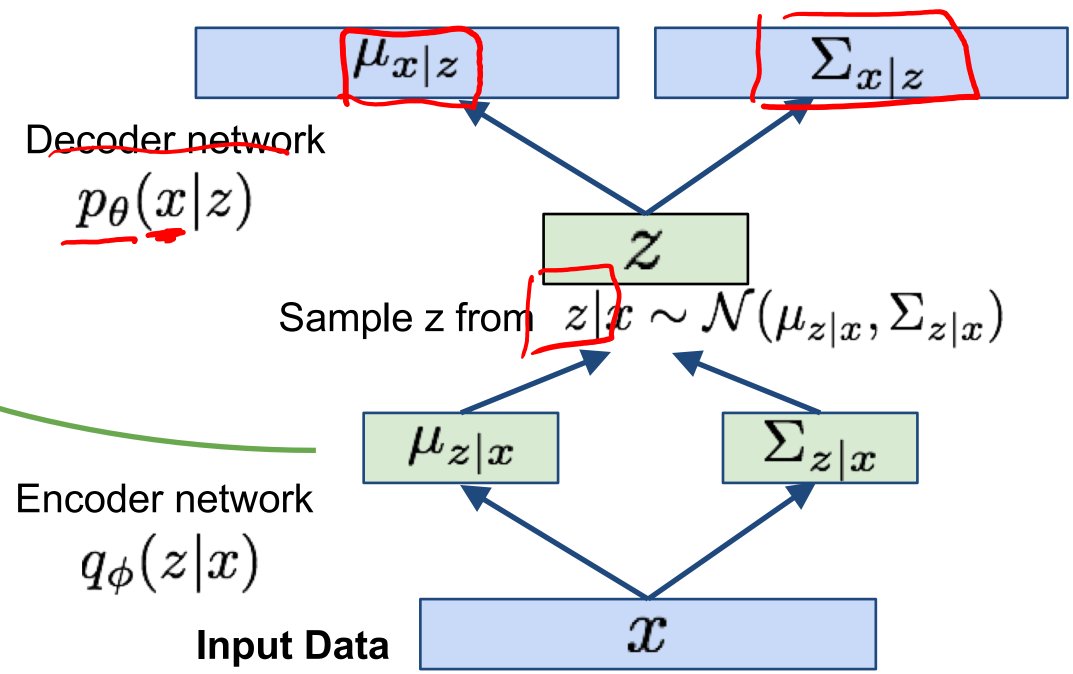
Figure Credit: http://gokererdogan.github.io/2016/07/01/reparameterization-trick/

Variational Auto Encoders

Putting it all together: maximizing the likelihood lower bound

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

Make approximate posterior distribution close to prior



Generative Adversarial Networks (GAN)

So far...

$p_\theta(\vec{x})$

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

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

VAEs define intractable density function with latent **z**:

$p(\vec{x}, z)$

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

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

So far...

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

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

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!

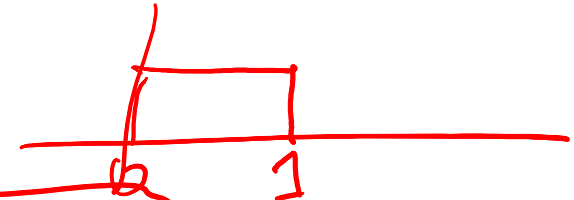
Generative Adversarial Networks (GANs)

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

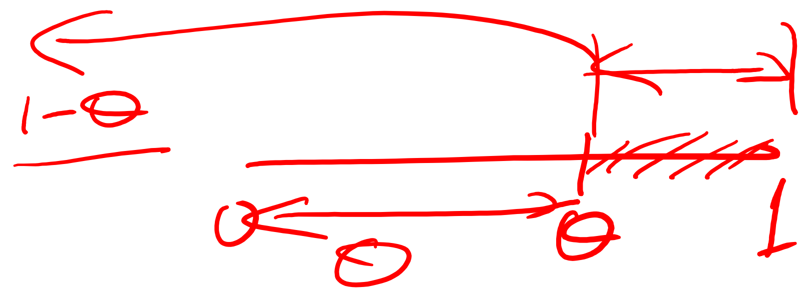
- I give you $u \sim U(0,1)$
- Produce a sample from Bern(θ)



$p(1) = \theta$

if $\boxed{u > \theta}$ output $\boxed{0}$

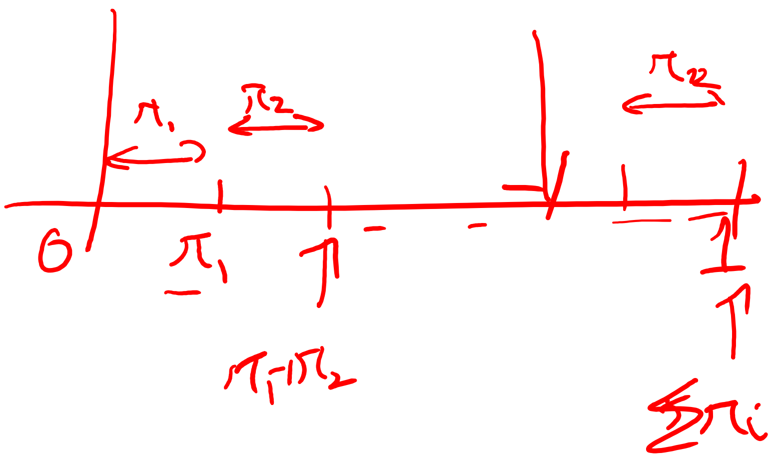
$u < \boxed{\theta}$ output $\boxed{1}$



Slightly Harder Interview Question

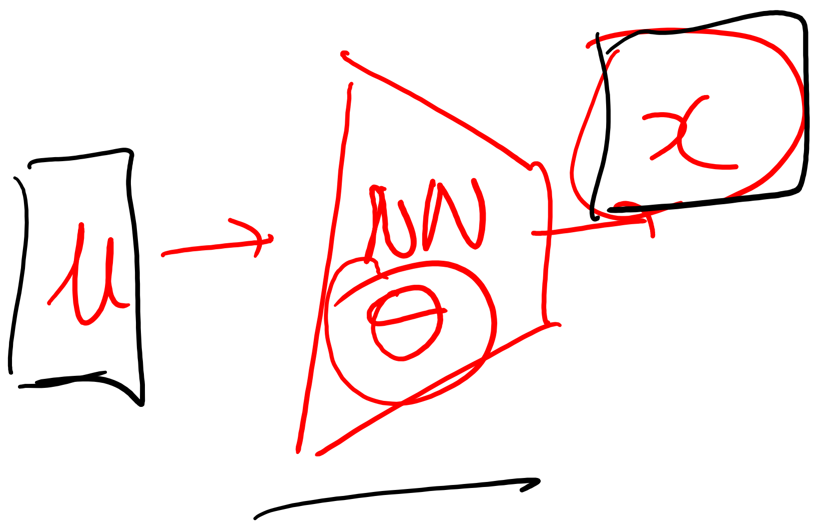
- I give you $\boxed{u \sim U(0,1)}$
- Produce a sample from $\boxed{Cat(\pi)}$

$$\begin{bmatrix}\pi_1 \\ \vdots \\ \pi_K\end{bmatrix}$$



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?

Generative Adversarial Networks

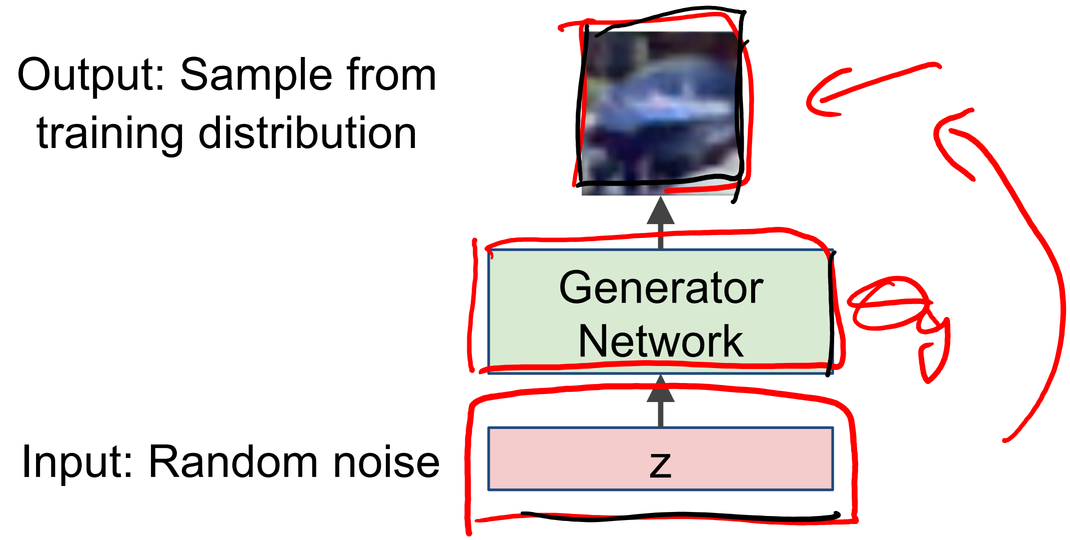
Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

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?

A: A neural network!



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

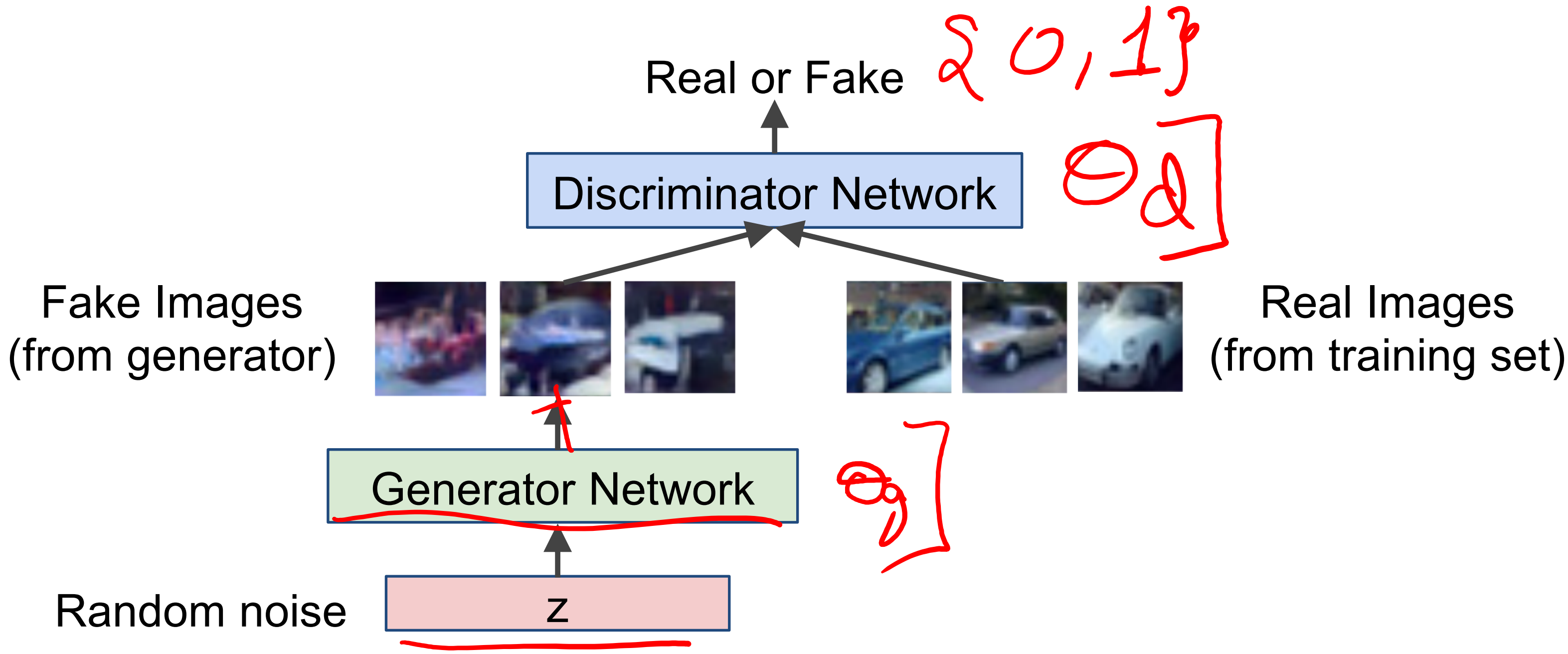
Ian Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Training GANs: Two-player game

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

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.

Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

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:

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

Training GANs: Two-player game

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

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:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

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

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:

Discriminator outputs likelihood in (0,1) of real image

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

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

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

θ_g const

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

2. **Gradient descent** on generator

θ_d const

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

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

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

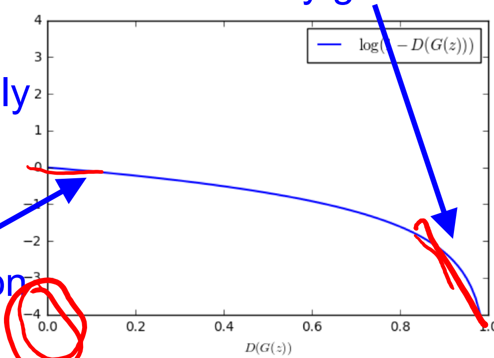
2. **Gradient descent** on generator

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

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!



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:

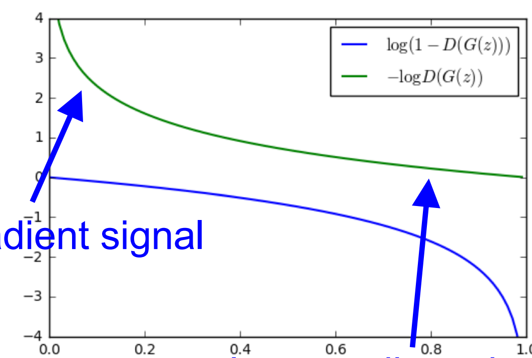
1. **Gradient ascent** on discriminator

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

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.



Training GANs: Two-player game

Ian Goodfellow et al., "Generative Adversarial Nets", NIPS 2014

Minimax objective function:

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

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

Alternate between:

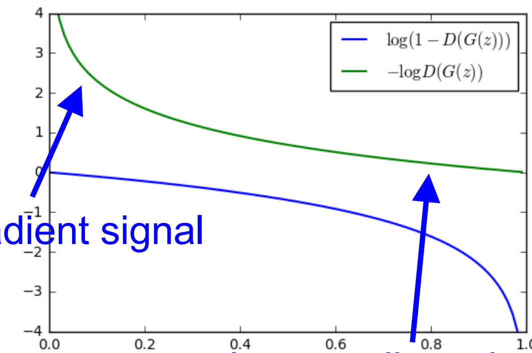
1. **Gradient ascent** on discriminator

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

2. **Instead: Gradient ascent** on generator, different objective

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

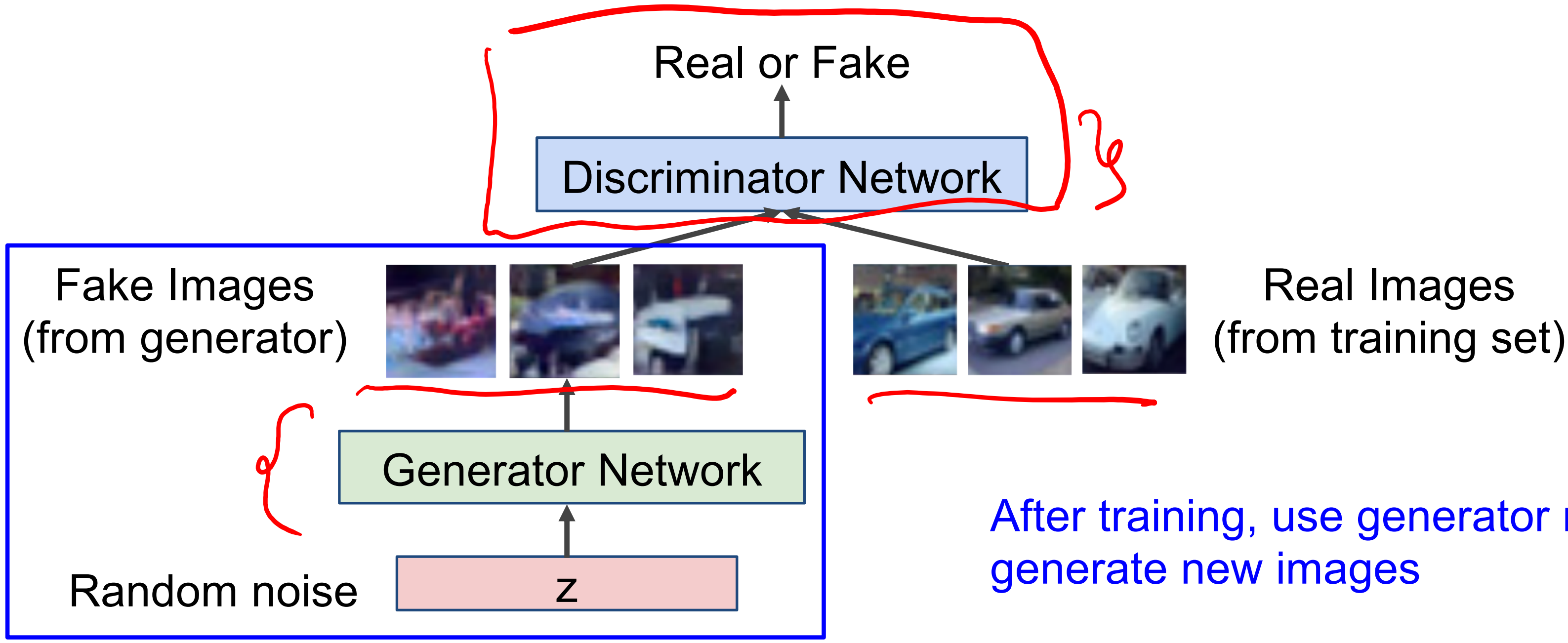
Instead of minimizing likelihood of discriminator being correct, now maximize likelihood of discriminator being wrong.
Same objective of fooling discriminator, but now higher gradient signal for bad samples => works much better! Standard in practice.



Training GANs: Two-player game

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

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images



After training, use generator network to generate new images

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

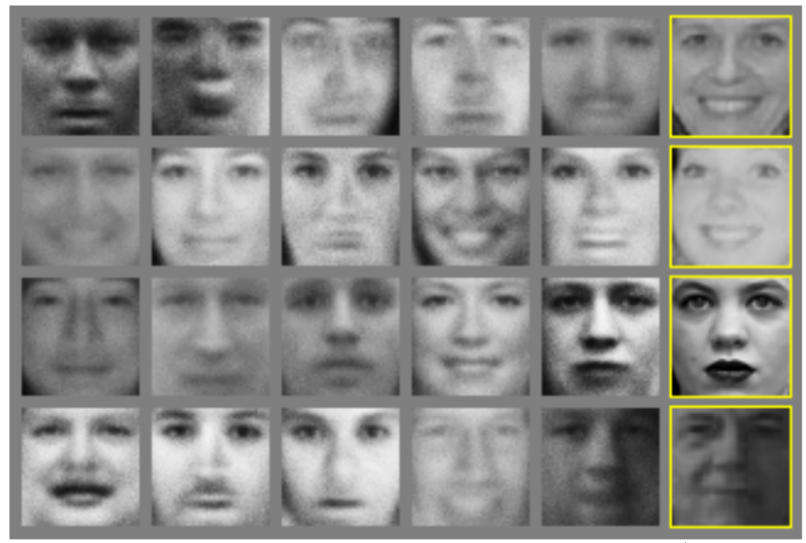
GANs

- Demo
  - https://poloclub.github.io/ganlab/

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

Generative Adversarial Nets

Generated samples



Nearest neighbor from training set

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

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

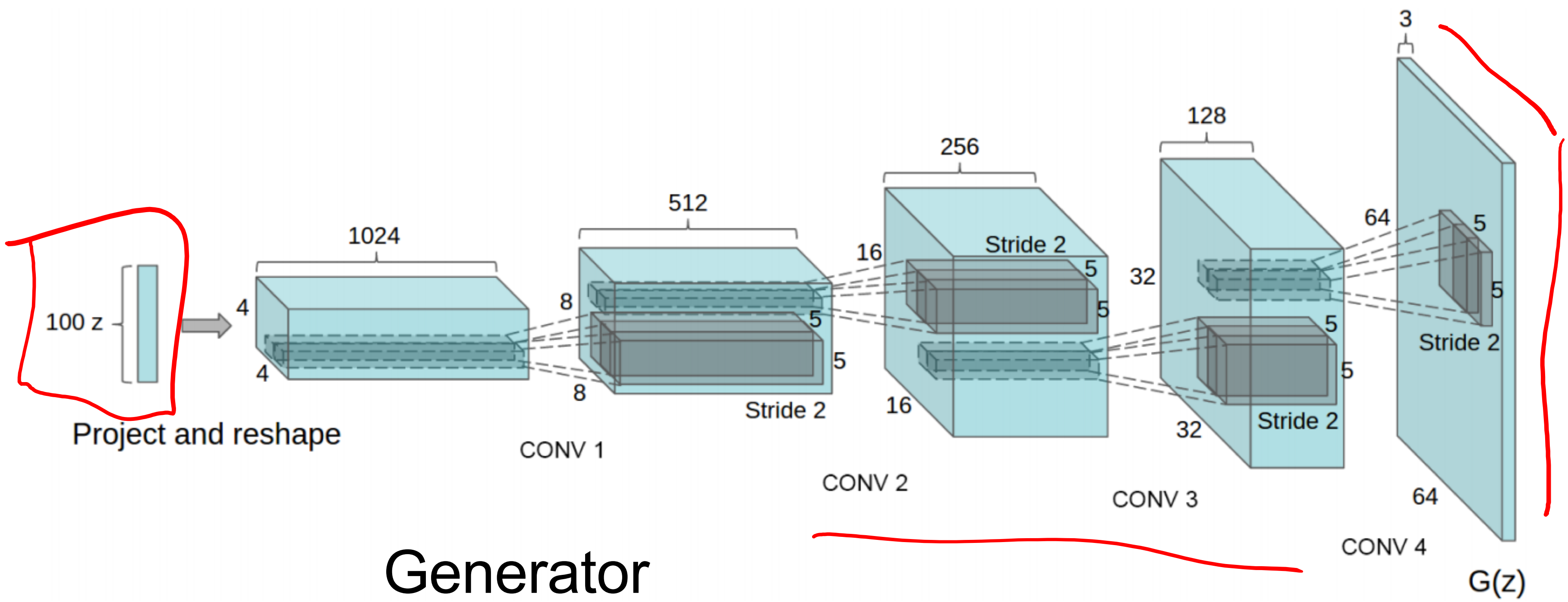
Generative Adversarial Nets

Generated samples (CIFAR-10)

Nearest neighbor from training set

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

Generative Adversarial Nets: Convolutional Architectures



Generator

Radford et al, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

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

BigGAN

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

BigGAN


(a) 128×128 (b) 256×256 (c) 512×512 (d)

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).

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

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 Categorial 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
- GAWWN - 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

2017: Explosion of GANs

See also: https://github.com/soumith/ganhacks for tips and tricks for trainings 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 Categorial 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
- GAWWN - 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

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

Plan for Today

- (Finish) Generative Adversarial Networks (GANs)
- Reinforcement Learning

Supervised Learning

$f(\cdot)$

$x \xrightarrow{f} y$

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.

Cat

Classification

This image is CC0 public domain

Unsupervised Learning

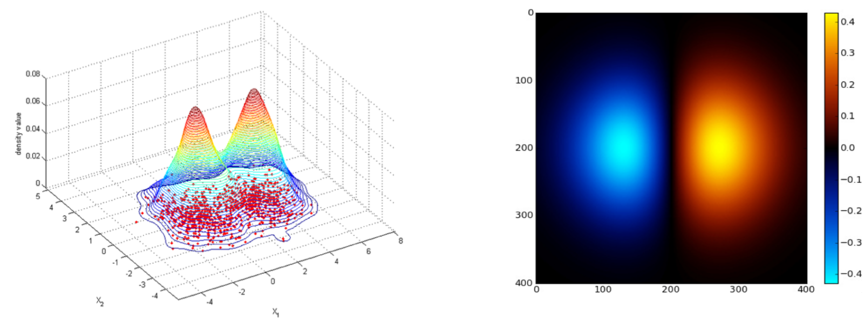
Data: x
Just data, no labels!

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

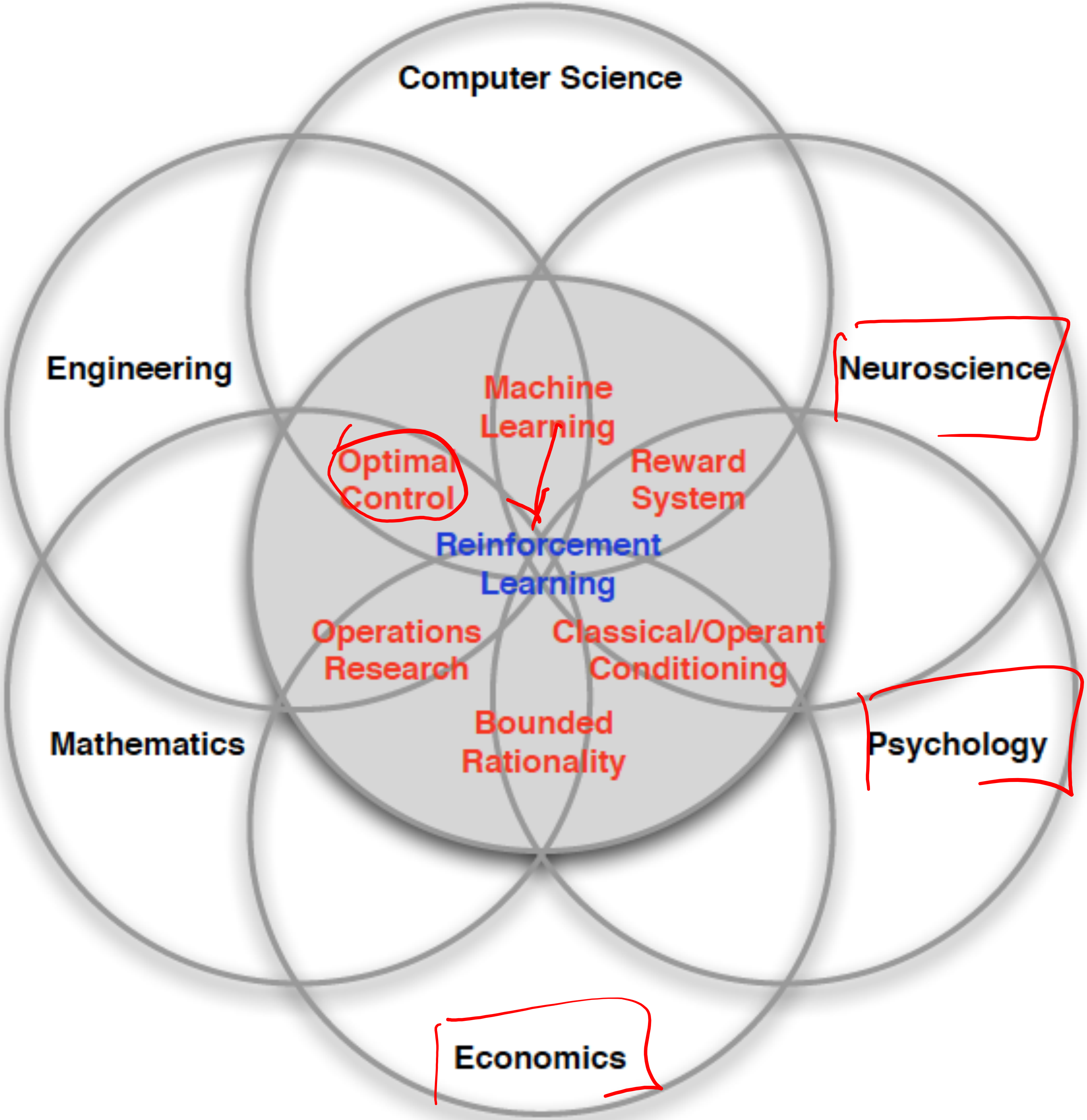
Types of Learning

- 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?

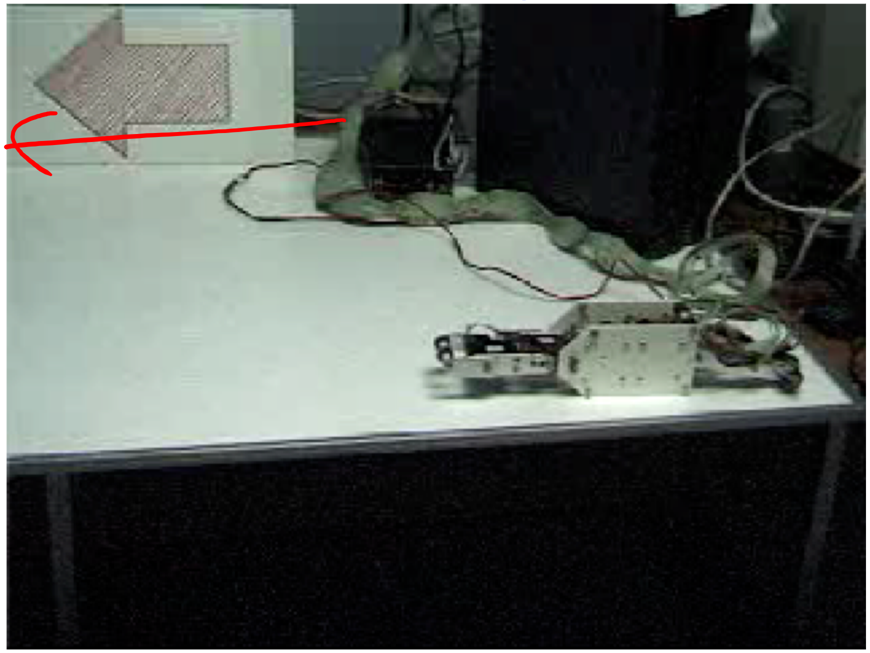
- Agent-oriented learning—learning by interacting with an environment to achieve a goal
  - more realistic and ambitious than other kinds of machine learning
- Learning by trial and error, with only delayed evaluative feedback (reward)
  - the kind of machine learning most like natural learning
  - learning that can tell for itself when it is right or wrong

Slide Credit: Rich Sutton

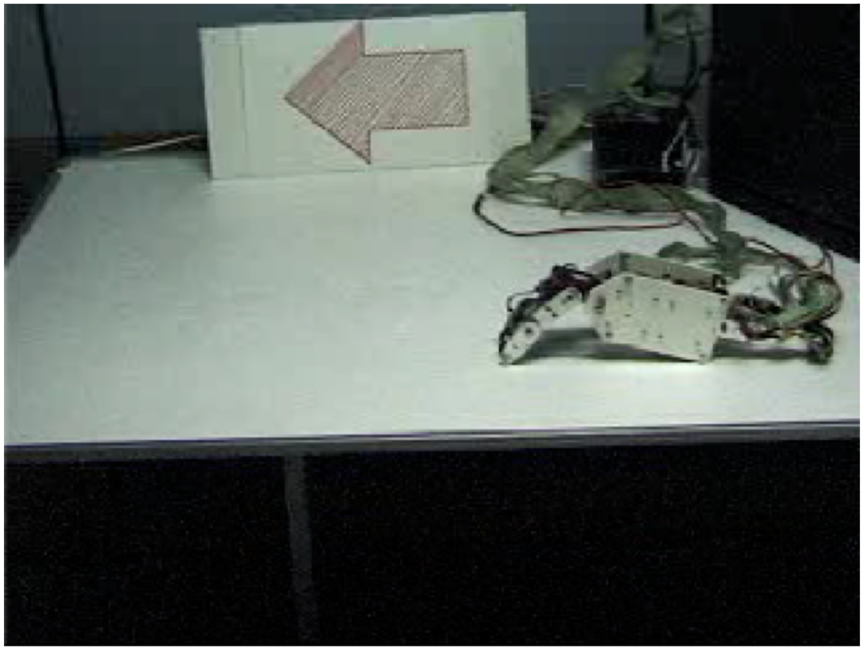

Computer Science
Engineering
Neuroscience
Machine Learning
Optimal Control
Reward System
Reinforcement Learning
Operations Research
Classical/Operant Conditioning
Bounded Rationality
Mathematics
Psychology
Economics

Silver 2015

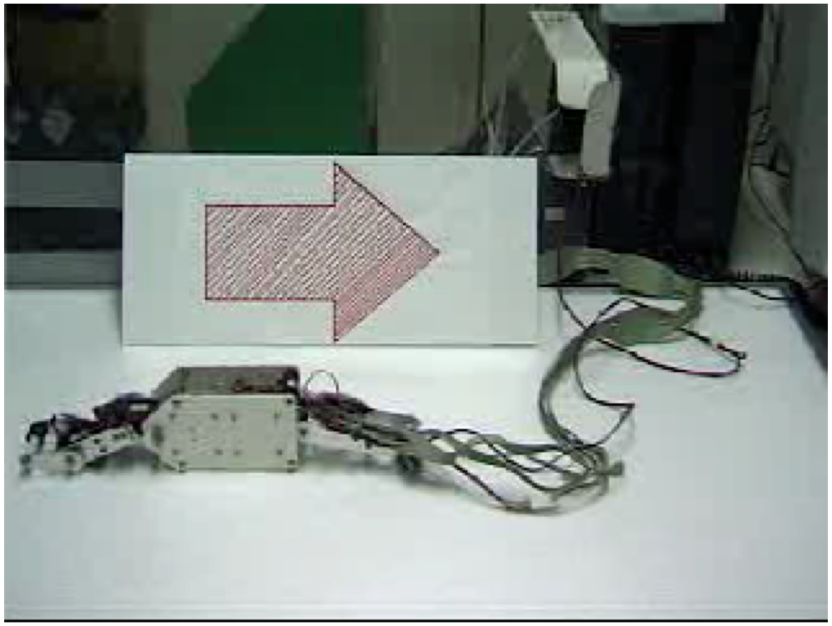
Example: Hajime Kimura's RL Robots



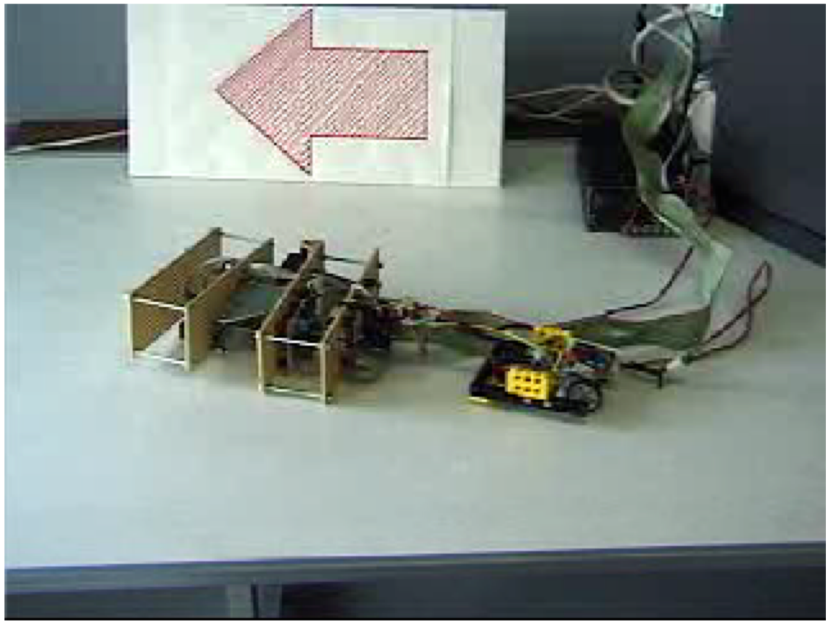
Before



After



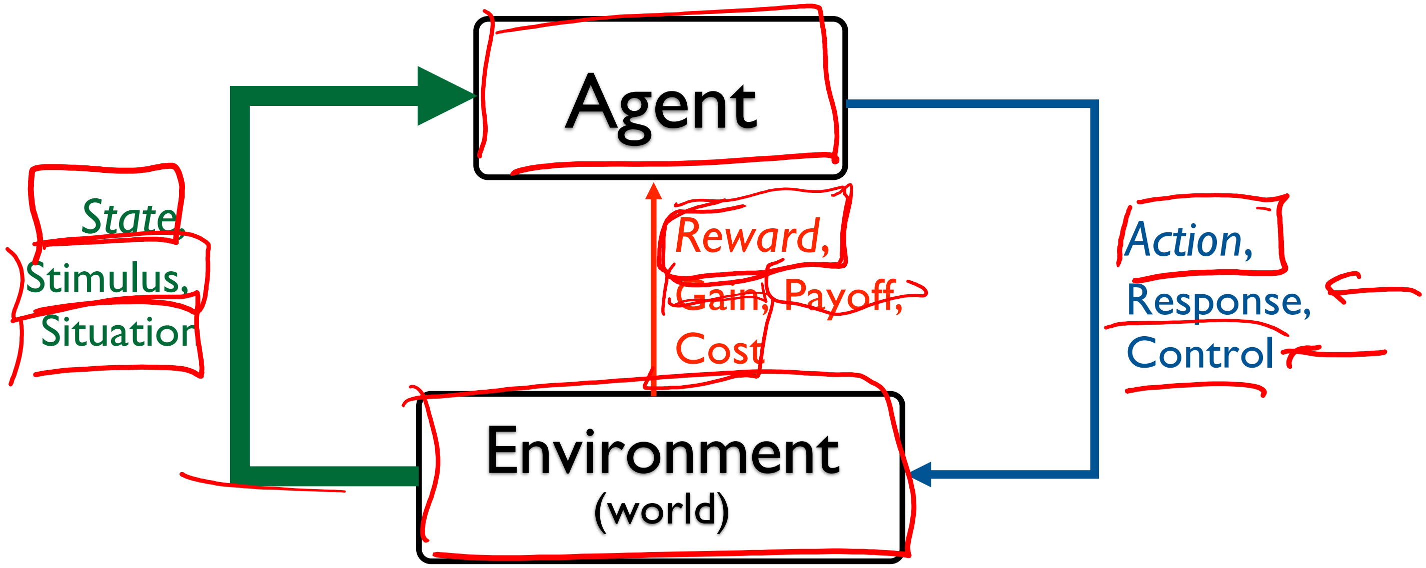
Backward



New Robot, Same algorithm

Slide Credit: Rich Sutton

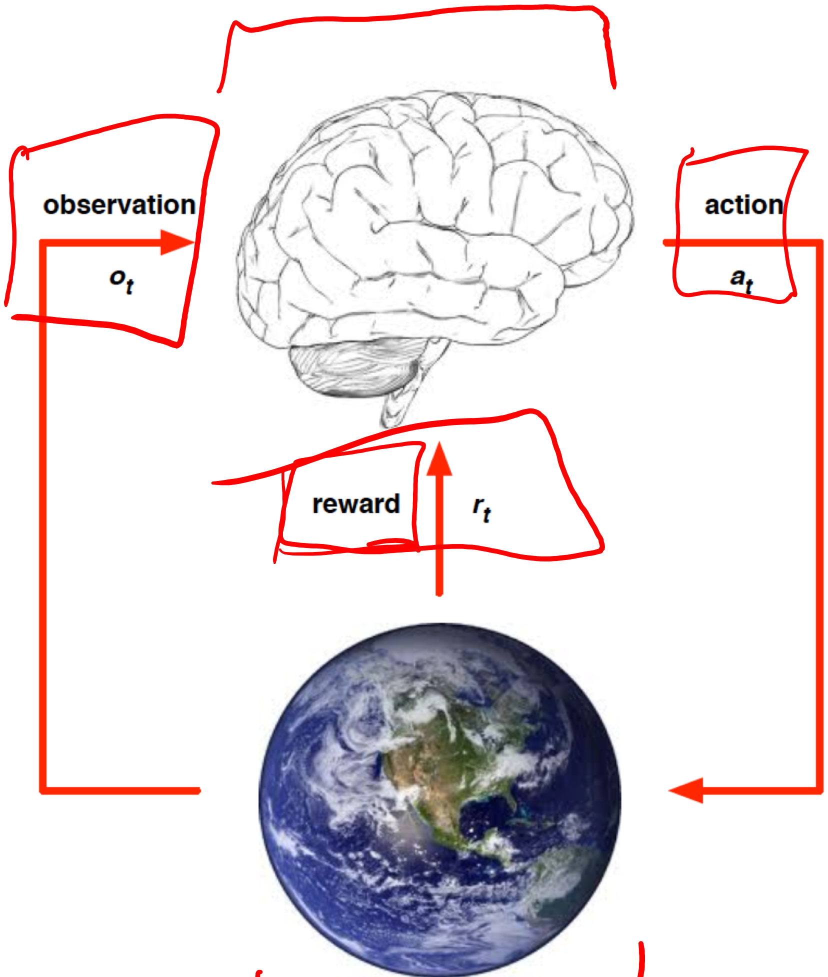
RL API



- 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}

$r_t(s_t, a_t)$

RL SL

$s_t \equiv x_t$

$a_t \equiv y_t$

$\ell(y^*, \hat{y}(x))$

$o_t = f(\boxed{s_t})$

Signature challenges of RL

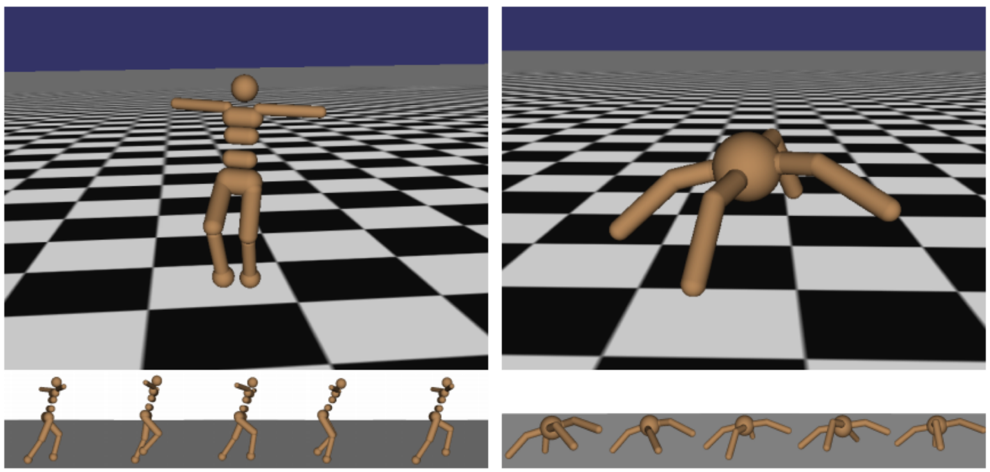
$-1, +1000$

$S_0, a_1, S_1, \ldots ;$

- $\boxed{\text{Evaluative feedback}}$ (reward)
- Sequentiality, delayed consequences
- Need for trial and error, to explore as well as exploit
- Non-stationarity $x, y \sim P_{data}(\ \ )$
- 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 upright + forward movement

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

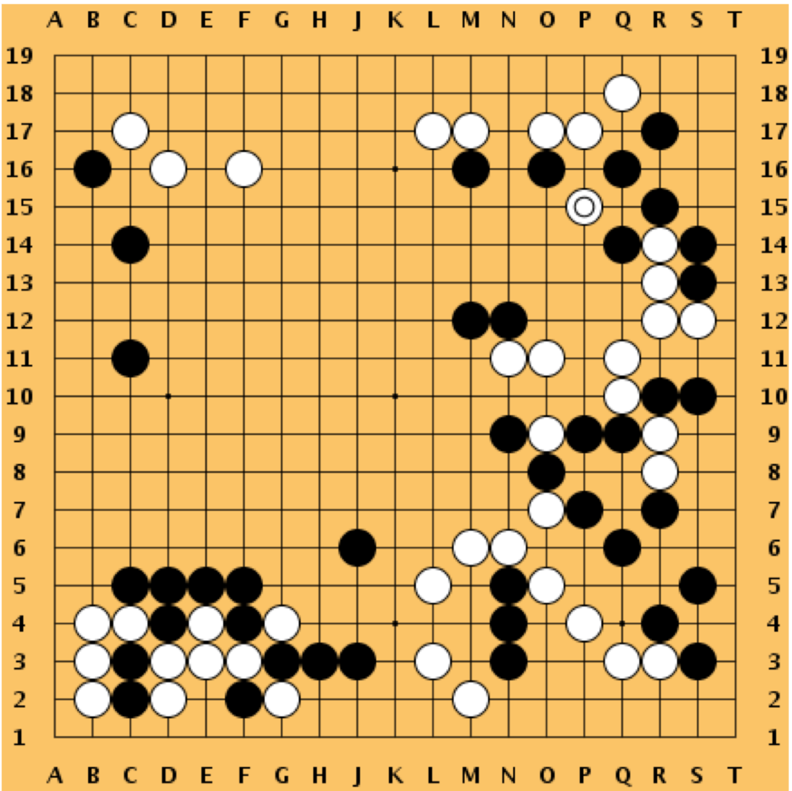
Atari Games

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.

Go



Objective: Win the game!

x y

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

This image is CC0 public domain

Demo

- http://projects.rajivshah.com/rldemo/
- https://cs.stanford.edu/people/karpathy/convnetjs/demo/rldemo.html

Markov Decision Process

- Mathematical formulation of the RL problem

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

π γ

$\mathcal{S}$: set of possible states
$\mathcal{A}$: set of possible actions
$\mathcal{R}$: distribution of reward given (state, action) pair
$\mathbb{P}$: transition probability i.e. distribution over next state given (state, action) pair
γ : discount factor

$P(S_{t+1} \mid S_t, a_t)$

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

Markov Decision Process

- Mathematical formulation of the RL problem

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

$\mathcal{S}$: set of possible states
$\mathcal{A}$: set of possible actions
$\mathcal{R}$: distribution of reward given (state, action) pair
$\mathbb{P}$: transition probability i.e. distribution over next state given (state, action) pair
γ : discount factor

- Life is trajectory: $\ldots S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, \boxed{R_{t+2}}, S_{t+2}, \ldots$

Markov Decision Process

- Mathematical formulation of the RL problem

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

$\mathcal{S}$: set of possible states
$\mathcal{A}$: set of possible actions
$\mathcal{R}$: distribution of reward given (state, action) pair
$\mathbb{P}$: transition probability i.e. distribution over next state given (state, action) pair
γ : discount factor

- Life is trajectory: $\ldots \boxed{S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}, S_{t+2},} \ldots$

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

$$p(r, s' | s, a) = Prob\left[R_{t+1} = r, S_{t+1} = s' \;\middle|\; S_t = s, A_t = a\right]$$

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

$\pi(\cdot)$

- A policy is how the agent acts

- Formally, map from states to actions

e.g.

State	Action
A	2
B	1

Deterministic policy: $\boxed{a = \pi(s)}$ $\pi_\theta(s)$

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

The optimal policy **π***

What's a good policy?

The optimal policy **π***

What's a good policy?

Maximizes current reward? Sum of all future reward?

The optimal policy **π***

What's a good policy?

Maximizes current reward? Sum of all future reward?

Discounted future rewards!

γ

$$\sum_t \underline{r_t} + \underline{\boxed{\gamma}\, r_{t+1}} + \underline{\gamma^2 r_{t+2}} + \underline{\gamma^3 r_{t+3}} \;\ldots$$

The optimal policy $\boldsymbol{\pi}$*

What's a good policy?

Maximizes current reward? Sum of all future reward?

Discounted future rewards!

Formally: $$\boxed{\pi^*} = \arg\max_{\pi} \mathbb{E}\left[\sum_{t \geq 0}^{\infty} \boxed{\gamma^t r_t} | \pi\right]$$

with $$s_0 \sim p(s_0), a_t \sim \pi(\cdot|s_t), s_{t+1} \sim \boxed{p}(\cdot|s_t, a_t)$$

$a_t = \pi(s_t)$ $s_{t+1} = EM(s_t, a_t)$

Value Function

- A value function is a prediction of future reward

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

- "Action Value Function" or Q-function Q
  - 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) $s_0, a_0, r_0, s_1, a_1, r_1, \ldots$

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

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

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):

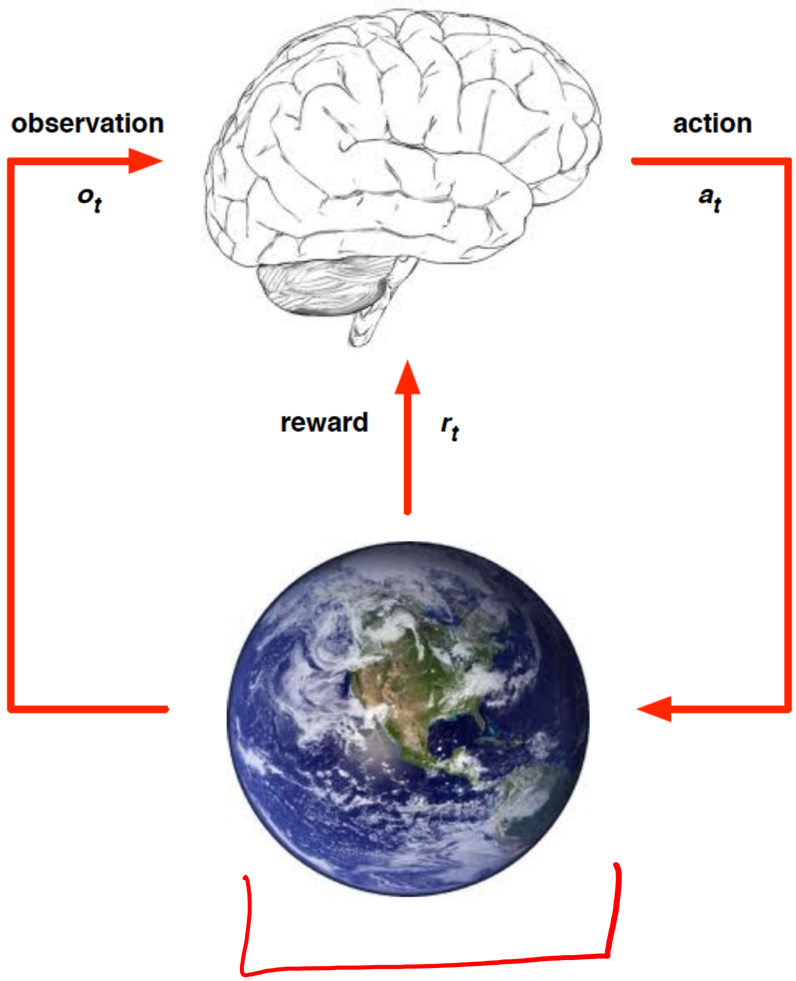
$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, \pi\right]$$

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

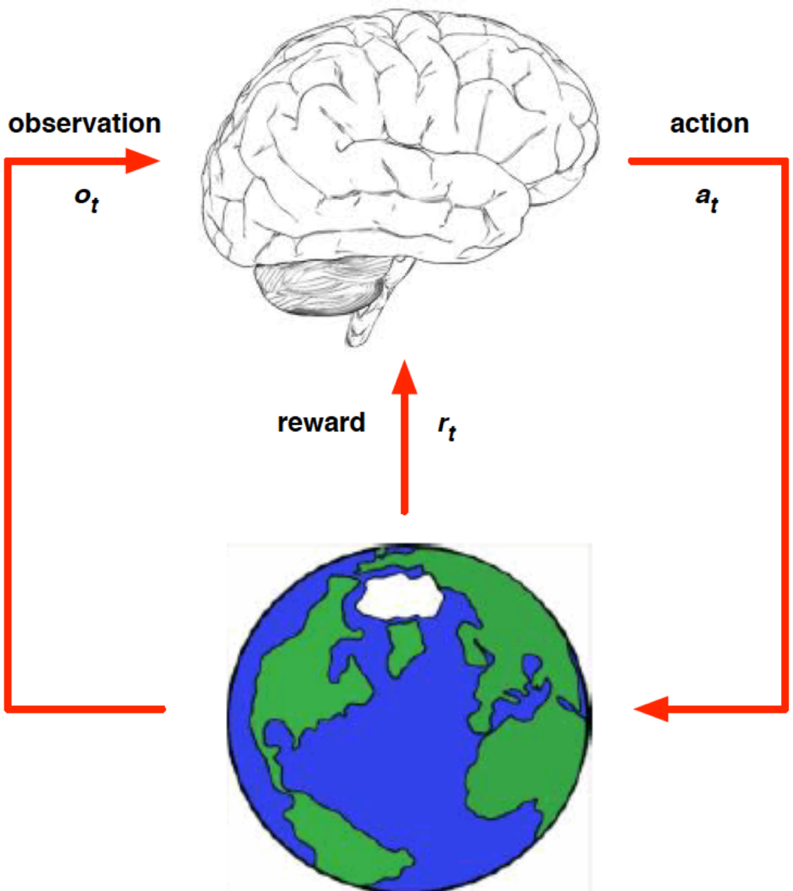
$$\boxed{Q^{\pi}(s,a)} = \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

Model



Model

- Model predicts what the world will do next

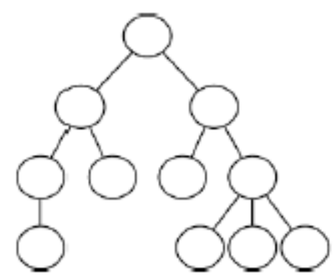


Model is learnt from experience

Acts as proxy for environment

Planner interacts with model

e.g. using lookahead search



$s_t, a_t \longrightarrow s_{t+1}$

$\boxed{P(s_{t+1} \mid s_t, a_t)}$