Topics:

- Reinforcement Learning Part 1
  - **Policy Gradients**

CS 4803-DL / 7643-A
ZSOLT KIRA

- **Assignment 4 out**
  - Due date **extended** to **April 8th 11:59pm EST.**
  - **Last HW!**

- **Projects**
  - Will try to get **feedback** back to you before project period starts

- Outline of rest of course:
  - Reinforcement Learning
  - Guest lectures/other topics (e.g. self-supervised learning, audio)
    - **April 7th:** Wav2Vec !!
    - **April 9th:** Ishan Misra (FB) on Self-Supervised Learning
    - **April 14th:** Automatic Speech Recognition Systems
  - Generative models (VAEs / GANs)

Nirbhay Modhe

Nirbhay Modhe is a PhD Student in the School of Interactive Computing at Georgia Tech advised by Prof. Dhruv Batra. His research interests within Reinforcement Learning (RL) include model based RL, generalization guarantees in RL and unsupervised or reward-free RL for exploration. Prior to starting his PhD program in 2017, he received his Bachelor's degree in Computer Science at the Indian Institute of Technology (IIT), Kanpur where he worked with Prof. Piyush Rai on Bayesian ML applied to multi-label learning.

Slides Brought to You By…

Georgia Tech

Recap: MDPs, Policy

- Markov Decision Processes (MDPs)
  - States, Actions, Reward dist., Transition dist., Discount factor (gamma)

MDP
$$(\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathbb{T}, \gamma)$$

- Policy:
  - Mapping from states to actions (deterministic)
  - Distribution of actions given states (stochastic)

State	Action
A ⟶	2
B ⟶	1

- What is a good policy?
  - Maximize **discounted sum of future rewards**
    - Discount factor: γ

First Lecture

Value Iteration

- Bellman update to state value estimates

Q-Value Iteration

- Bellman update to (state, action) value estimates

Policy Iteration

- Policy evaluation + refinement

Sample-Based Policy Evaluation?

- We want to improve our estimate of V by computing these averages:

$$V^{\pi}_{k+1}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}_{k}(s')]$$

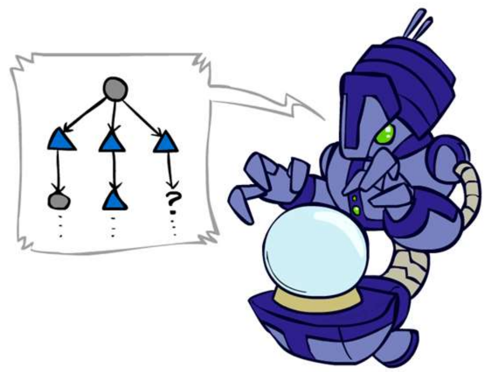
- Idea: Take samples of outcomes s' (by doing th

$$sample_1 = R(s, \pi(s), s'_1) + \gamma V^{\pi}_{k}(s'_1)$$

$$sample_2 = R(s, \pi(s), s'_2) + \gamma V^{\pi}_{k}(s'_2)$$

$$\dots$$

$$sample_n = R(s, \pi(s), s'_n) + \gamma V^{\pi}_{k}(s'_n)$$



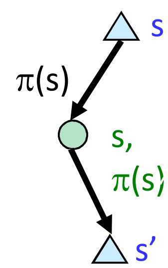
$$V^{\pi}_{k+1}(s) \leftarrow \frac{1}{n} \sum_{i} sample_i$$

What's the difficulty of this algorithm?

state s.

Temporal Difference Learning

- Big idea: learn from every experience!
  - Update V(s) each time we experience a transition (s, a, s', r)
  - Likely outcomes s' will contribute updates more often
- Temporal difference learning of values
  - Policy still fixed, still doing evaluation!
  - Move values toward value of whatever successor occurs: running average



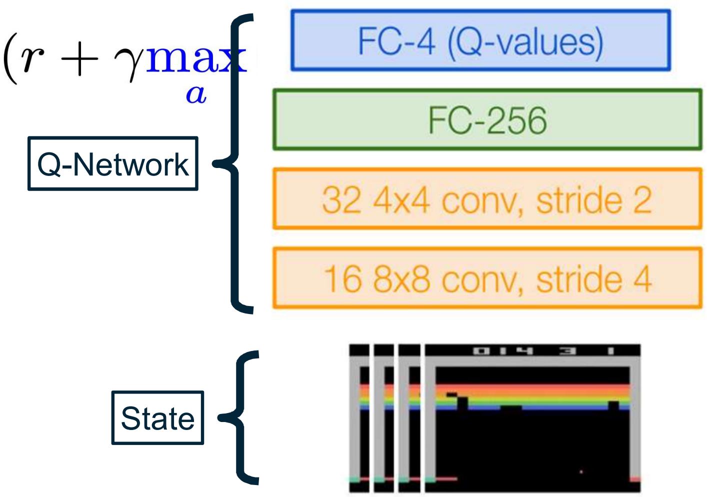
Sample of V(s): $sample = R(s, \pi(s), s') + \gamma V^{\pi}(s')$

Update to V(s): $V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + (\alpha)sample$

Same update: $V^{\pi}(s) \leftarrow V^{\pi}(s) + \alpha(sample - V^{\pi}(s))$

- Minibatch of $\{(s, a, s', r)_i\}_{i=1}^{B}$
- Forward pass: State → Q-Network → Q-Values per action

  $B \times D$ ... $B \times n_{actions}$
- Compute loss: $\Big(\underbrace{Q_{new}(s, a)}_{\theta_{new}} - (r + \gamma \max_{a}$



- Backward pass: $\dfrac{\partial Loss}{\partial \theta_{new}}$

Deep Q-Learning

Georgia Tech

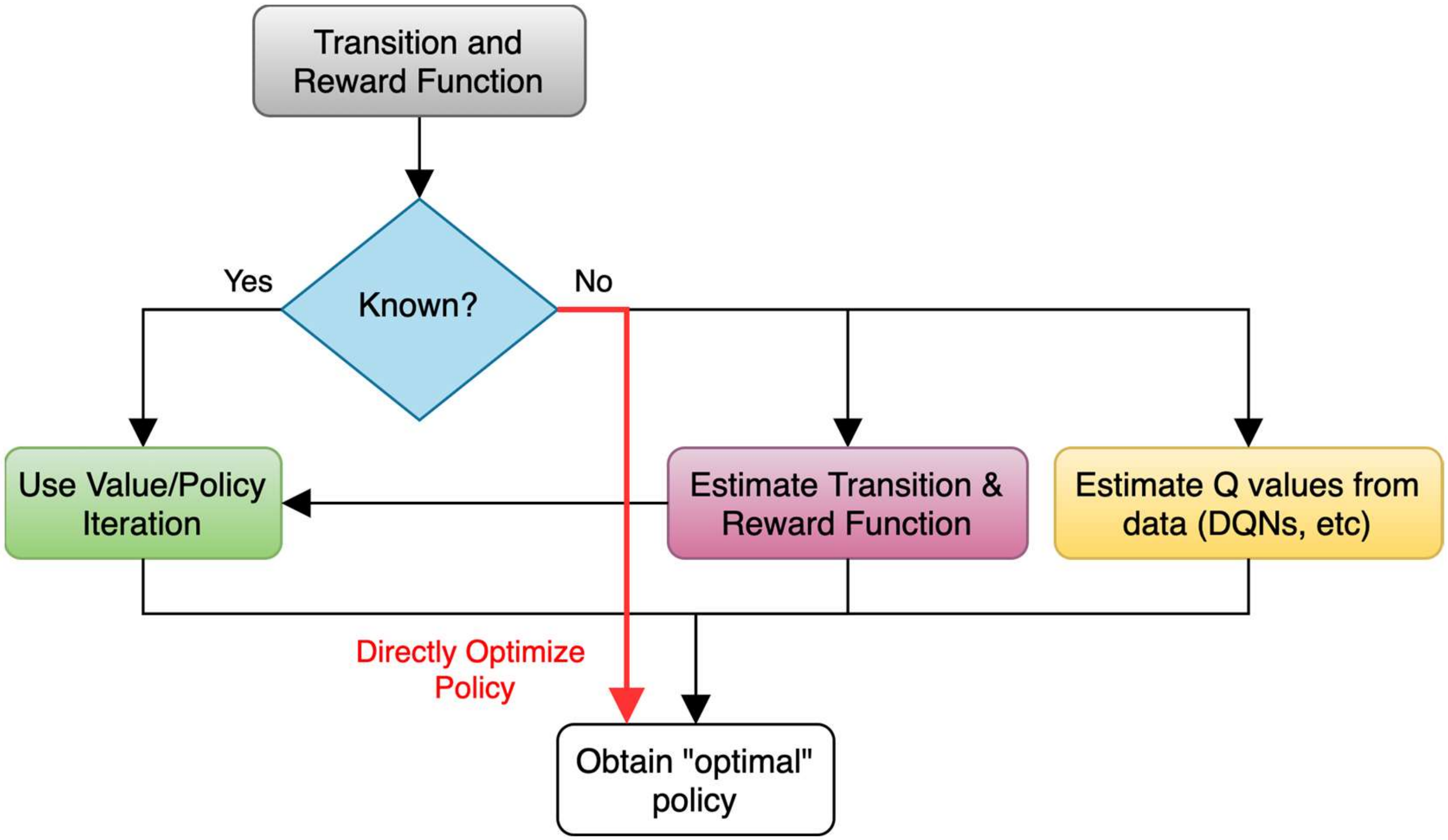
- **Dynamic Programming**
  - Value, Q-Value Iteration
  - Policy Iteration

- **Reinforcement Learning (RL)**
  - The challenges of (deep) learning based methods
  - Value-based RL algorithms
    - Deep Q-Learning

Today

- **Policy-based RL algorithms** (policy gradients)

Policy Gradients, Actor-Critic


Transition and Reward Function
Known?
Yes
No
Use Value/Policy Iteration
Estimate Transition & Reward Function
Estimate Q values from data (DQNs, etc)
Directly Optimize Policy
Obtain "optimal" policy

Overview

- Class of policies defined by parameters θ

$$\boxed{\pi_\theta(a|s) : \mathcal{S} \to \mathcal{A}}$$

  - Eg: θ can be parameters of linear transformation, deep network, etc.

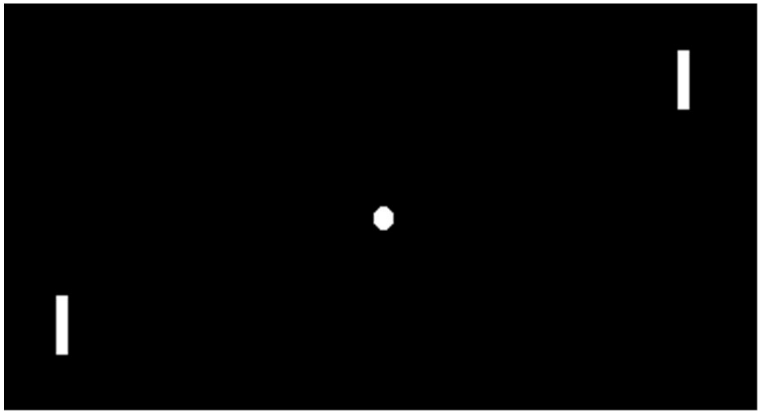
- Want to maximize:

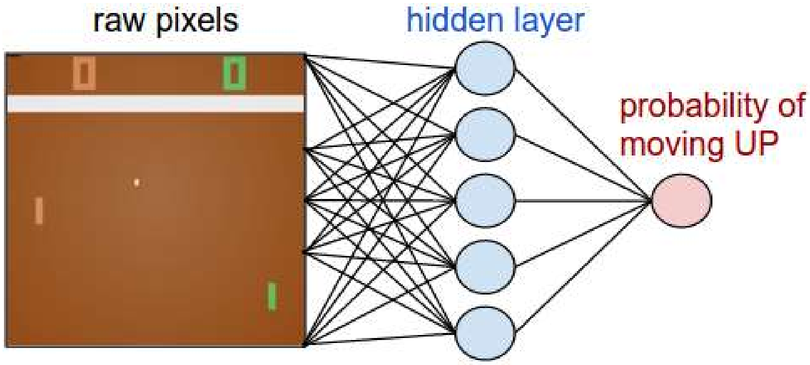
$$J(\pi) = \mathbb{E}\left[\sum_{t=1}^{T} \mathcal{R}(s_t, a_t)\right]$$

- In other words,

$$\pi^* = \arg\max_{\pi:\mathcal{S}\to\mathcal{A}} \mathbb{E}\left[\sum_{t=1}^{T} \mathcal{R}(s_t, a_t)\right] \Rightarrow \theta^* = \arg\max_{\theta} \mathbb{E}\left[\sum_{t=1}^{T} \mathcal{R}(s_t, a_t)\right]$$

Parametrized Policy





Pong from Pixels

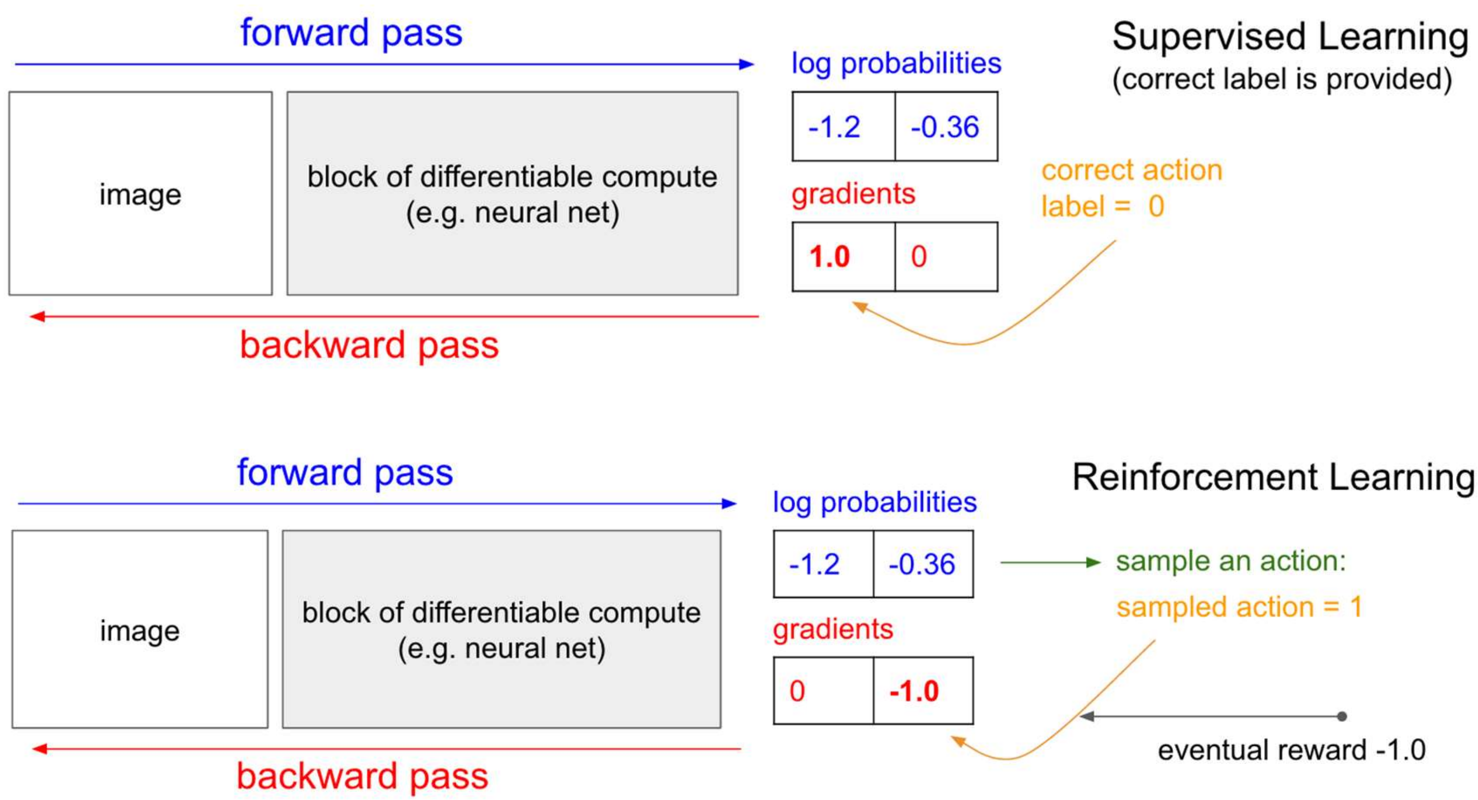


Image Source: http://karpathy.github.io/2016/05/31/rl/

Policy Gradient: Loss Function

Georgia Tech

- Slightly re-writing the notation

Let $\tau = (s_0, a_0, \ldots s_T, a_T)$ denote a trajectory

$$\pi_\theta(\tau) = p_\theta(\tau) = p_\theta\left(s_0, a_0, \ldots s_T, a_T\right)$$
$$= p(s_0) \prod_{t=0}^{T} p_\theta\left(a_t \mid s_t\right) \cdot p\left(s_{t+1} \mid s_t, a_t\right)$$

$$\arg\max_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\mathcal{R}(\tau)\right]$$

Gathering Data/Experience

Georgia Tech

$$J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)}\left[\mathcal{R}(\tau)\right]$$

$$= \mathbb{E}_{a_t \sim \pi(\cdot|s_t), s_{t+1} \sim p(\cdot|s_t, a_t)}\left[\sum_{t=0}^{T} \mathcal{R}(s_t, a_t)\right]$$

- How to gather data?
  - We already have a policy: π_θ
  - Sample N trajectories $\{\tau_i\}_{i=1}^{N}$ by acting according to π_θ

$$\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{T} r(s_t^i, a_t^i)$$

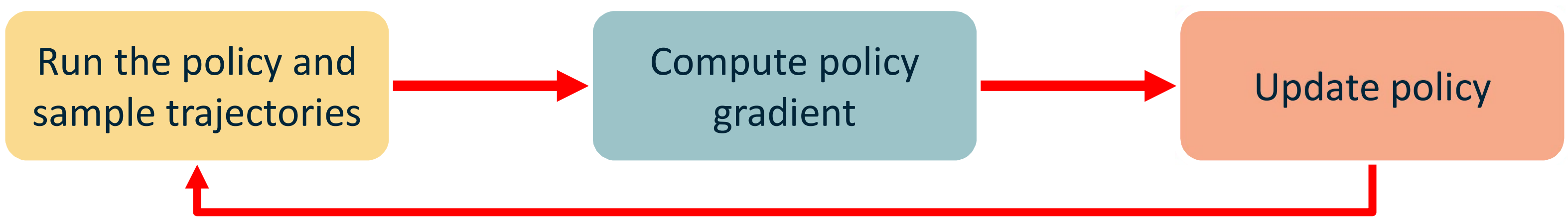
Gathering Data/Experience

The REINFORCE Algorithm

- Sample trajectories $\tau_i = \{s_1, a_1, \ldots s_T, a_T\}_i$ by acting according to π_θ
- Compute policy gradient as

$$\nabla_\theta J(\theta) \approx \ ?$$

- Update policy parameters: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$



Slide credit: Sergey Levine

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

$$= \nabla_\theta \int \pi_\theta(\tau)\mathcal{R}(\tau)d\tau$$

Expectation as integral

$$= \int \nabla_\theta \pi_\theta(\tau)\mathcal{R}(\tau)d\tau$$

Exchange integral and gradient

$$= \int \nabla_\theta \pi_\theta(\tau) \cdot \frac{\pi_\theta(\tau)}{\pi_\theta(\tau)} \cdot \mathcal{R}(\tau)d\tau$$

$$= \int \pi_\theta(\tau)\nabla_\theta \log \pi_\theta(\tau)\mathcal{R}(\tau)d\tau$$

$$\nabla_\theta \log \pi(\tau) = \frac{\nabla_\theta \pi(\tau)}{\pi(\tau)}$$

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

Deriving The Policy Gradient

Georgia Tech

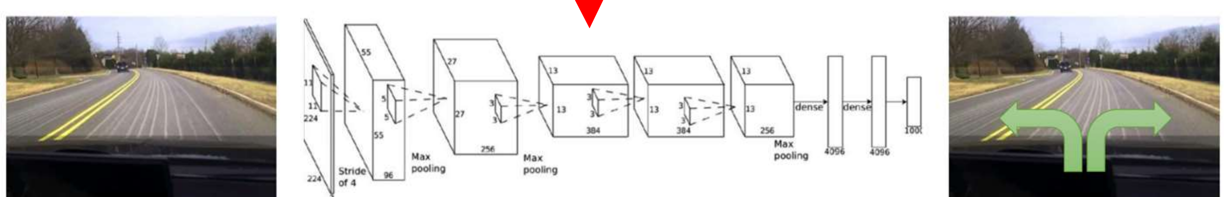
$$\pi_\theta(\tau) = p(s_0) \prod_{t=0}^{T} p_\theta(a_t \mid s_t) \cdot p(s_{t+1} \mid s_t, a_t)$$

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

$$\nabla_\theta \left[\log p(s_0) + \sum_{t=1}^{T} \log \pi_\theta(a_t|s_t) + \sum_{t=1}^{T} \log p(s_{t+1} \mid s_t, a_t) \right]$$

Doesn't depend on Transition probabilities!

$$= \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \sum_{t=1}^{T} \mathcal{R}(s_t, a_t) \right]$$



$\mathbf{s}_t$ $\qquad \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)$ $\qquad \mathbf{a}_t$

Continuous Action Space?

Deriving The Policy Gradient

Georgia Tech

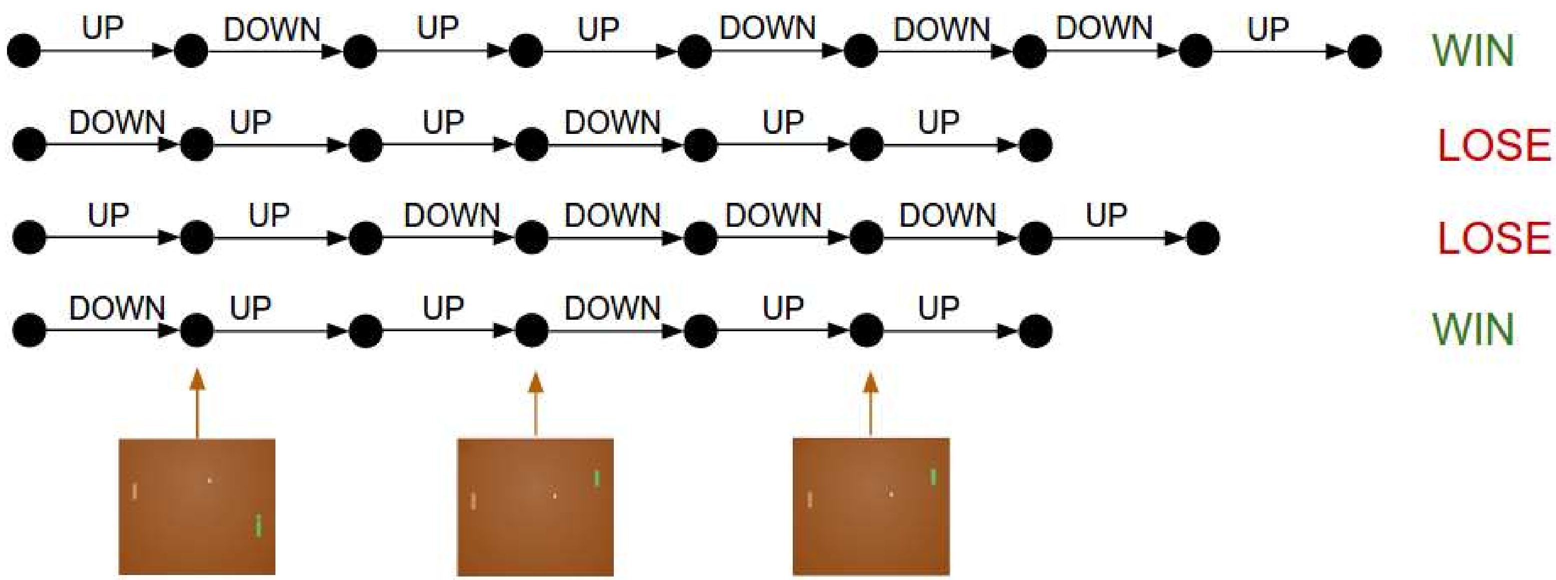
The REINFORCE Algorithm

- Sample trajectories $\tau_i = \{s_1, a_1, \ldots s_T, a_T\}_i$ by acting according to π_θ
- Compute policy gradient as

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{i}^{N} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_t^i \mid s_t^i \right) \cdot \sum_{t=1}^{T} \mathcal{R} \left(s_t^i \mid a_t^i \right) \right]$$

- Update policy parameters: $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Slide credit: Sergey Levine



Slide credit: Dhruv Batra

Drawbacks of Policy Gradients

Georgia Tech

Issues with Policy Gradients

- Credit assignment is hard!
  - Which specific action led to increase in reward
  - Suffers from high variance → leading to unstable training

Variance reduction

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} r(\tau)\nabla_\theta \log \pi_\theta(a_t|s_t)$$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t\geq 0} \left(\sum_{t'\geq t} r_{t'}\right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Variance reduction

Gradient estimator: $$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} r(\tau) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

First idea: Push up probabilities of an action seen, only by the cumulative future reward from that state

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

Second idea: Use discount factor γ to ignore delayed effects

$$\nabla_\theta J(\theta) \approx \sum_{t \geq 0} \left(\sum_{t' \geq t} \gamma^{t'-t} r_{t'} \right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

- Credit assignment is hard!
  - Which specific action led to increase in reward
  - Suffers from **high variance**, leading to unstable training

- How to reduce the variance?
  - Subtract an action independent baseline from the reward

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau \sim p_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right) \cdot \sum_{t=1}^{T} \left(\mathcal{R} \left(s_t, a_t \right) - b(s_t) \right) \right]$$

  - Why does it work?
  - What is the best choice of b?

Drawbacks of Policy Gradients

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function!

Actor-Critic

- Learn both policy and Q function
  - Use the "actor" to sample trajectories
  - Use the Q function to "evaluate" or "critic" the policy

Actor-Critic

- Learn both policy and Q function
  - Use the "actor" to sample trajectories
  - Use the Q function to "evaluate" or "critic" the policy

- REINFORCE: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) \mathcal{R}(s,a) \right]$

- Actor-critic: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a) \right]$

Actor-Critic

- Learn both policy and Q function
  - Use the "actor" to sample trajectories
  - Use the Q function to "evaluate" or "critic" the policy
- REINFORCE: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) \mathcal{R}(s, a) \right]$
- Actor-critic: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s, a) \right]$
- Q function is unknown too! Update using $\mathcal{R}(s, a)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  – Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$
  - evaluate "actor" using "critic" $Q_\beta(s,a)$

Actor-Critic

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$
  - evaluate "actor" using "critic" $Q_\beta(s,a)$ and update policy:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a \mid s) Q_\beta(s,a)$$

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$
  - evaluate "actor" using "critic" $Q_\beta(s,a)$ and update policy:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a \mid s) Q_\beta(s,a)$$

  - Update "critic": $\mathrm{MSE\ Loss} := \Big(Q_{new}(s,a) - (r + \max_a Q_{old}(s',a)) \Big)^2$
    - Recall Q-learning

Actor-Critic

Georgia Tech

- Initialize s, θ (policy network) and β (Q network)
- sample action $a \sim \pi_\theta(\cdot|s)$
- For each step:
  - Sample reward $\mathcal{R}(s,a)$ and next state $s' \sim p(s'|s,a)$
  - evaluate "actor" using "critic" $Q_\beta(s,a)$ and update policy:

$$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(a \mid s) Q_\beta(s,a)$$

  - Update "critic":
    - Recall Q-learning $\text{MSE Loss} := \left(Q_{new}(s,a) - (r + \max_a Q_{old}(s',a))\right)^2$

$$a \leftarrow a', s \leftarrow s'$$

    - Update β Accordingly

Actor-Critic

Georgia Tech

How to choose the baseline?

A better baseline: Want to push up the probability of an action from a state, if this action was better than the **expected value of what we should get from that state**.

Q: What does this remind you of?

A: Q-function and value function!

Intuitively, we are happy with an action a_t in a state s_t if $Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$ is large. On the contrary, we are unhappy with an action if it's small.

Using this, we get the estimator: $\nabla_\theta J(\theta) \approx \sum_{t \geq 0} (Q^{\pi_\theta}(s_t, a_t) - V^{\pi_\theta}(s_t)) \nabla_\theta \log \pi_\theta(a_t|s_t)$

Actor-critic

- In general, replacing the policy evaluation or the "critic" leads to different flavors of the actor-critic
  - REINFORCE: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\mathcal{R}(s,a)\right]$
  - Q – Actor Critic $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)Q^{\pi_\theta}(s,a)\right]$
  - Advantage Actor Critic: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)A^{\pi_\theta}(s,a)\right]$

$$= Q^{\pi_\theta}(s,a) - V^{\pi_\theta}(s)$$

"how much better is an action than expected?

Summary

- Policy Learning:
  - Policy gradients
  - REINFORCE
  - Reducing Variance (Homework!)
- Actor-Critic:
  - Other ways of performing "policy evaluation"
  - Variants of Actor-critic

Summary

- **Policy gradients**: very general but suffer from high variance so requires a lot of samples. **Challenge**: sample-efficiency
- **Q-learning**: does not always work but when it works, usually more sample-efficient. **Challenge**: exploration

- Guarantees:
  - **Policy Gradients**: Converges to a local minima of J(θ), often good enough!
  - **Q-learning**: Zero guarantees since you are approximating Bellman equation with a complicated function approximator

- Sparse long-horizon tasks (Montezuma's revenge)
- Imitation Learning
- Sim2Real – Simulation to real, domain randomization
- Lifelong Learning
- Safety
- World Models

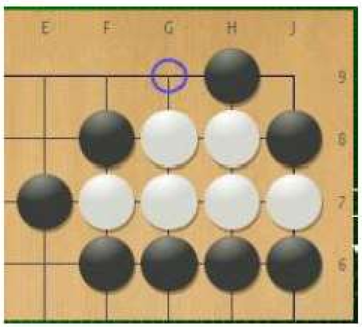
Open Problems / Challenges

Georgia Tech

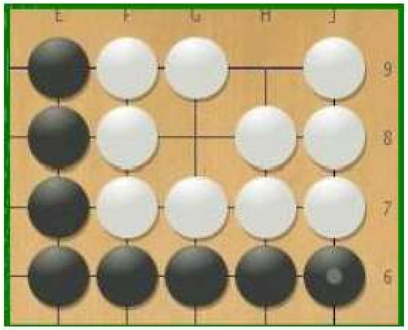
Playing Go

Rules

- Each player puts a stone on the goban, black first
- Each stone remains on the goban, except:



group w/o degree freedom is killed



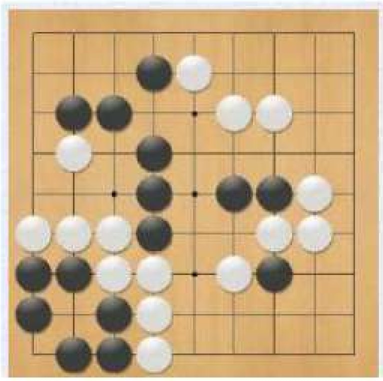
a group with two eyes can't be killed

- The goal is to control the max. territory

Go is a Difficult Game

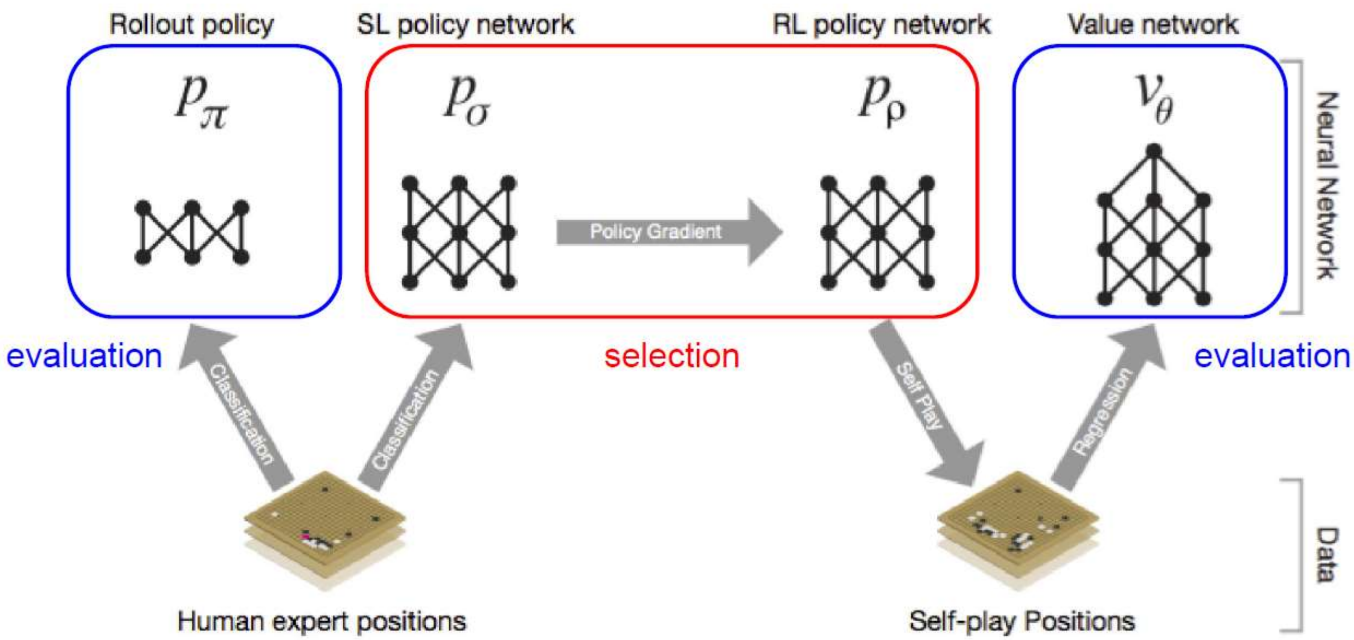
Features

- Size of the state space 2.10^{170}
- Size of the action space 200
- No good evaluation function
- Local and global features (symmetries, freedom, ...)
- A move might make a difference some dozen plies later



AlphaGo

- Go is a perfect information game
  - See entire board at all times
  - Has an optimal value function!
- Key idea: We cannot unroll search tree to learn a policy/value for a large number of states, instead:
  - Reduce depth of search via **position evaluation**: Replace subtrees with estimated value function *v(s)*
  - Reduce breadth of search via **action sampling**: Don't perform unlikely actions
    - Start by predicting expert actions, gives you a probability distribution
- Use Monte Carlo rollouts, with a policy, selecting children with higher values
  - As policy improves this search improves too


Rollout policy
SL policy network
RL policy network
Value network
p_π
p_σ
p_ρ
v_θ
Policy Gradient
Neural Network
evaluation
selection
evaluation
Classification
Classification
Self Play
Regression
Data
Human expert positions
Self-play Positions

AlphaGo Zero

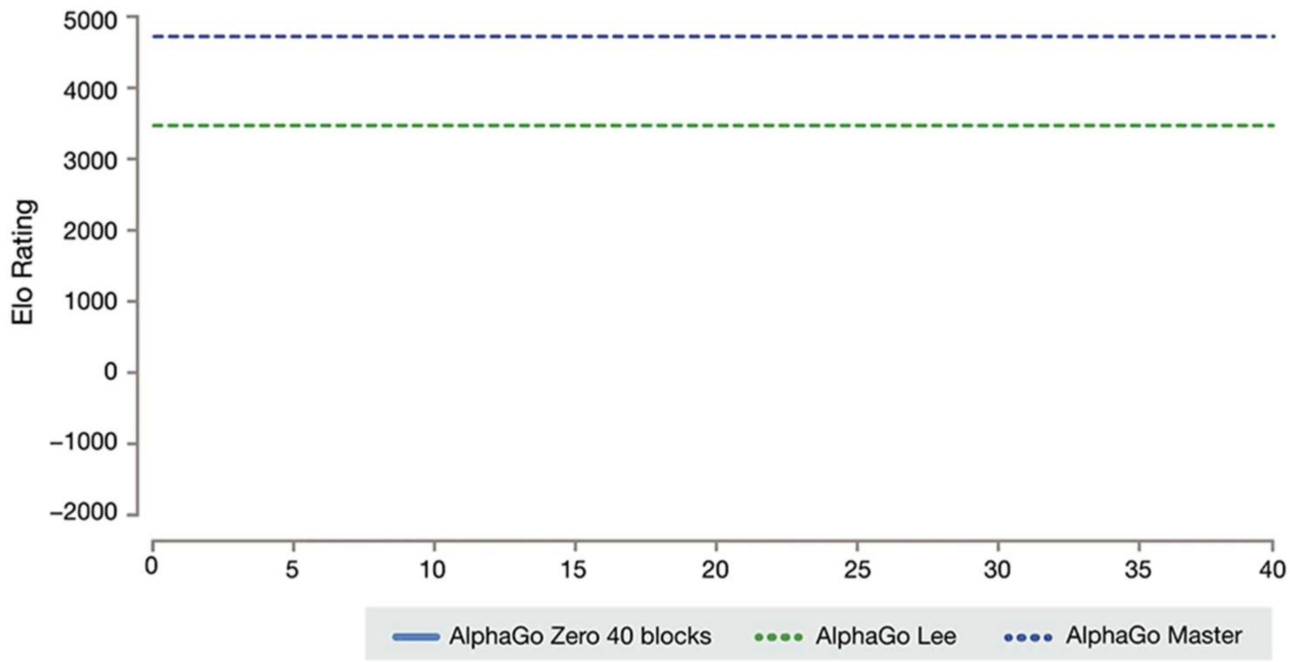
- MCTS with *Self-Play*
  - Don't have to guess what opponent might do, so…
  - If no exploration, a big-branching game tree becomes *one path*
  - You get an automatically improving, evenly-matched opponent who *is accurately learning your strategy*
  - "We have met the enemy, and he is us" (famous variant of Pogo, 1954)
  - No need for human expert scoring rules for boards from unfinished games

- Treat board as an image: use residual convolutional neural network

- AlphaGo Zero: One deep neural network learns both the value function and policy in parallel

- Alpha Zero: Removed rollout altogether from MCTS and just used current neural net estimates instead

AlphaGo Zero



D. Silver et al., Mastering the Game of Go without Human Knowledge, Nature 550, October 2017

https://deepmind.com/blog/alphago-zero-learning-scratch/

World Models