COMP6247: Reinforcement and Online Learning

Mahesan Niranjan

School of Electronics and Computer Science
University of Southampton

Sample of Case Studies

Spring Semester 2020/21

Review of Recent Papers

- **Li, K. and Malik, J. Learning to Optimize Neural Networks,** `arXiv preprint arXiv:1703.00441`
- Mnih, V. *et al.* Human-level control... [Atari games], *Nature* (2015)
- Yuan, X. *et al.* Reinforcement Learning for Elevator Control, *IFAC Conference* (2008)
- Popova, M. *et al.* Deep reinforcement learning for *de novo* drug design, *Science Advances* (2018)
- Mao, H. *et al.* Resource management with deep reinforcement learning, 15th *ACM Workshop on Hot Topics in Networks* (2016).

Review of Recent Papers

When reviewing for the assignment... look for:

- State and action spaces
- Reward, cost function
- Representations
- Algorithm
- Empirical work to illustrate the point

- Ignore some technical / algorithmic details outside the scope of what we have studied

I. Li and Malik: Learning to Optimize Neural Networks

- Training algorithm:
  - Objective function f
  - Initial conditions $\boldsymbol{x}^{(0)}$
  - Learning curve: $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(T)}$
- We do a lot of this:
  - $f_1, f_2, \ldots, f_n$
- We tune hyper-parameters, choose step size, taper down learning rate...
- Can we learn from the experience of training networks, to efficiently train new networks?
- Reinforcement Learning formulation:
  - Step size selection as actions
  - Past updates as observation / state: o_t, features Ψ_t
  - Cost to minimize, rather than reward to maximize

Li and Malik: Learning to Optimize Neural Networks

Formulation

- Goal: Policy that minimizes *expected total* cost over time (epochs)

> algorithm is to learn a policy π^* that minimizes the total expected cost over time. More precisely,
>
> $$\pi^* = \arg\min_{\pi} \mathbb{E}_{s_0, a_0, s_1, \ldots, s_T} \left[\sum_{t=0}^{T} c(s_t) \right],$$
>
> where the expectation is taken with respect to the joint dis-

- Probability distribution over trajectory (state sequence)

$$q(s_0, a_0, s_1, \ldots, s_T) = \int_{o_0, \ldots, o_T} p_i(s_0)\, p_o(o_0 | s_0) \prod_{t=0}^{T-1} \pi(a_t | o_t, t)\, p(s_{t+1} | s_t, a_t)\, p_o(o_{t+1} | s_{t+1}).$$

- Policy: $\pi(a_t | o_t, t)$, conditional probability over actions
- $\pi(a_t | o_t, t) = \mathcal{N}(\mu^{\pi}(o_t), \Sigma^{\pi}(o_t))$
- $\mu^{\pi}(o_t)$ and $\Sigma^{\pi}(o_t)$ come from function approximators (neural networks)

Guided Policy Search

- Searching over large class of policies: π, probability distribution over actions (discrete / continuous)
- Idea is to maintain two policies:
  - ψ Linear, learned in closed form
  - π More expressive, desired
  - In each iteration, solve ψ and use it to train π.
- Objective function as constrained optimization problem:

> More precisely, GPS solves the following constrained optimization problem:
>
> $$\min_{\theta, \eta} \mathbb{E}_{\psi} \left[\sum_{t=0}^{T} c(s_t) \right] \text{ s.t. } \psi(a_t | s_t, t; \eta) = \pi(a_t | s_t; \theta)\ \forall a_t, s_t, t$$
>
> where η and θ denote the parameters of ψ and π respectively, $\mathbb{E}_{\rho}[\cdot]$ denotes the expectation taken with respect to the trajectory induced by a policy ρ and $\pi(a_t | s_t; \theta) := \int_{o_t} \pi(a_t | o_t; \theta)\, p_o(o_t | s_t)^2$.

Guided Policy Search (cont'd)

- Constrained optimization

$$\min_{\boldsymbol{\theta},\boldsymbol{\eta}} E_\psi \left[\sum_{t=0}^{T} c(o_t) \right]$$
$$\text{subject to} \ \ \psi(a_t|s_t, t; \boldsymbol{\eta}) = \pi(a_t|s_t; \boldsymbol{\theta})$$

- Relaxed version:

$$\min_{\boldsymbol{\theta},\boldsymbol{\eta}} E_\psi \left[\sum_{t=0}^{T} c(o_t) \right]$$
$$\text{subject to} \ \ E_\psi[a_t] = E_\psi[E_\pi[a_t|s_t]]$$

- The simpler model and the desired model to be the same (select the same action) in expectation (on average).

Solution

Iterative algorithm (ADMM)

- Update groups of parameters

$$\eta \leftarrow \arg\min_{\eta} \sum_{t=0}^{T} \mathbb{E}_\psi \left[c(s_t) - \lambda_t^T a_t \right] + \nu_t D_t(\eta, \theta)$$
$$\theta \leftarrow \arg\min_{\theta} \sum_{t=0}^{T} \lambda_t^T \mathbb{E}_\psi \left[\mathbb{E}_\pi \left[a_t | s_t \right] \right] + \nu_t D_t(\theta, \eta)$$
$$\lambda_t \leftarrow \lambda_t + \alpha \nu_t \left(\mathbb{E}_\psi \left[\mathbb{E}_\pi \left[a_t | s_t \right] \right] - \mathbb{E}_\psi \left[a_t \right] \right) \ \forall t,$$

where $D_t(\theta, \eta) := \mathbb{E}_\psi \left[D_{KL} \left(\pi(a_t|s_t; \theta) \| \psi(a_t|s_t, t; \eta) \right) \right]$
and $D_t(\eta, \theta) := \mathbb{E}_\psi \left[D_{KL} \left(\psi(a_t|s_t, t; \eta) \| \pi(a_t|s_t; \theta) \right) \right]$.

- $D_t(\boldsymbol{\theta}, \boldsymbol{\eta}) = E_\psi \left[D_{\mathrm{KL}}(\pi(a_t|s_t; \boldsymbol{\theta}) \, || \, \psi(a_t|s_t, t; \boldsymbol{\eta}) \right]$
- Recall: $D_{\mathrm{KL}}(p||q) = E_p \left[\log(p/q) \right] = \int_{-\infty}^{\infty} p(x) \log(p(x)/q(x)) dx$

- Setting the densities:

  The algorithm assumes that $\psi\left(a_t \middle| s_t, t; \eta\right) = \mathcal{N}\left(K_t s_t + k_t, G_t\right)$, where $\eta := \left(K_t, k_t, G_t\right)_{t=1}^{T}$ and $\pi\left(a_t \middle| o_t; \theta\right) = \mathcal{N}\left(\mu_\omega^\pi(o_t), \Sigma^\pi\right)$, where $\theta := \left(\omega, \Sigma^\pi\right)$ and $\mu_\omega^\pi(\cdot)$ can be an arbitrary function that is typically modelled using a nonlinear function approximator like a neural net.

- Recall: linear transform; Multi-variate Gaussian!

$$\widetilde{p}(s_{t+1}|s_t, a_t, t; \xi) = \mathcal{N}\left(A_t s_t + B_t a_t + c_t,\ F_t\right)$$

  Draw samples of trajectory (using ψ) and fit linear model above.

- Quadratic approximation to cost function

  samples of s_t drawn from the trajectory induced by ψ so that $c(s_t) \approx \tilde{c}(s_t) := \frac{1}{2} s_t^T C_t s_t + d_t^T s_t + h_t$ for s_t's that are near the samples.

  With these assumptions, the subproblem that needs to be solved to update $\eta = (K_t, k_t, G_t)_{t=1}^{T}$ becomes:

$$\min_{\eta} \sum_{t=0}^{T} \mathbb{E}_{\tilde{\psi}}\left[\tilde{c}(s_t) - \lambda_t^T a_t\right] + \nu_t D_t(\eta, \theta)$$

$$\text{s.t.} \sum_{t=0}^{T} \mathbb{E}_{\tilde{\psi}}\left[D_{KL}\left(\psi\left(a_t \middle| s_t, t; \eta\right) \middle\| \psi\left(a_t \middle| s_t, t; \eta'\right)\right)\right] \leq \epsilon,$$

- Heavy in detail **skip**; But, But, But... Have we seen this before?
- Recall Extended Kalman Filter:
  - We had a nonlinear function
  - We had quadratic cost
  - We updated using local gradient
  - Got a Gaussian approximation going from $n-1|n-1$ to $n|n-1$

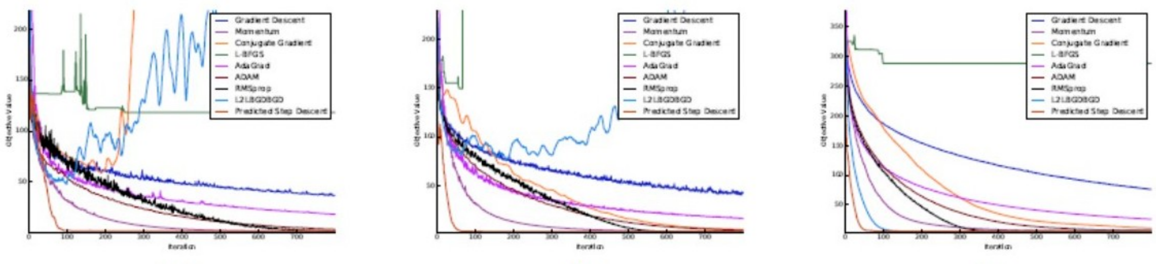
Experimental Work

- Features (to define state)

Because of the stochasticity of gradients and objective values, the state features $\Phi(\cdot)$ are defined in terms of summary statistics of the history of iterates $\left\{x^{(i)}\right\}_{i=0}^{t}$, gradients $\left\{\nabla \hat{f}(x^{(i)})\right\}_{i=0}^{t}$ and objective values $\left\{\hat{f}(x^{(i)})\right\}_{i=0}^{t}$. We define the following statistics, which we will refer to as the average recent iterate, gradient and objective value respectively:

- $\overline{x^{(i)}} := \frac{1}{\min(i+1,3)} \sum_{j=\max(i-2,0)}^{i} x^{(j)}$
- $\overline{\nabla \hat{f}(x^{(i)})} := \frac{1}{\min(i+1,3)} \sum_{j=\max(i-2,0)}^{i} \nabla \hat{f}(x^{(j)})$
- $\overline{\hat{f}(x^{(i)})} := \frac{1}{\min(i+1,3)} \sum_{j=\max(i-2,0)}^{i} \hat{f}(x^{(j)})$

- Results



(a) (b) (c)

- Review:
  - Show that you understand the framework
  - Disuss if the results are persuasive
  - Enjoy learning about the algorithm

II. Mnih, V. *et al.* Human-level control... [Atari games]

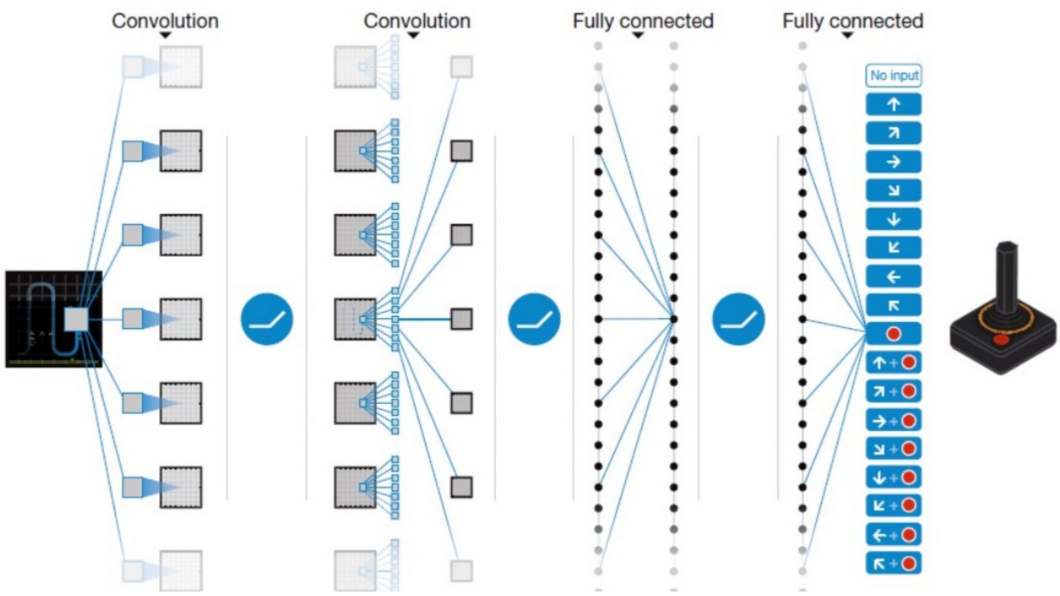
- Long history of computer games and machine learning
- Driven more by artificial intelligence than by statistical pattern recognition
- Sutton and Barto:
  - Samuel's checker player
  - Tesauro's Backgammon player
- Recent headline-grabbing developments in Atari Games (this paper), Go, Chess etc.
- What we are familiar with:

$$Q^*(s, a) = \max_{\pi} E\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots | s_t = s, a_t = 1, \pi\right]$$

- $Q(s, a)$ represented by a function approximator (*e.g.* RBF in the mountain car problem)
- Temporal difference error between $Q(s, a)$ and $r + \gamma \max_{a'} Q(s', a')$
- Issues with scaling up; two solutions in this paper (**experience replay** and **updating frequency**)

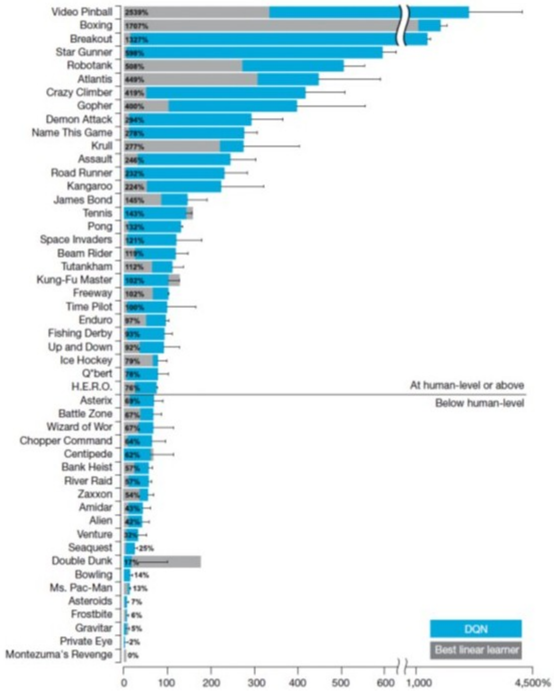
II. Mnih, V. *et al.* Human-level control... [Atari games] (cont'd)

- Architecture



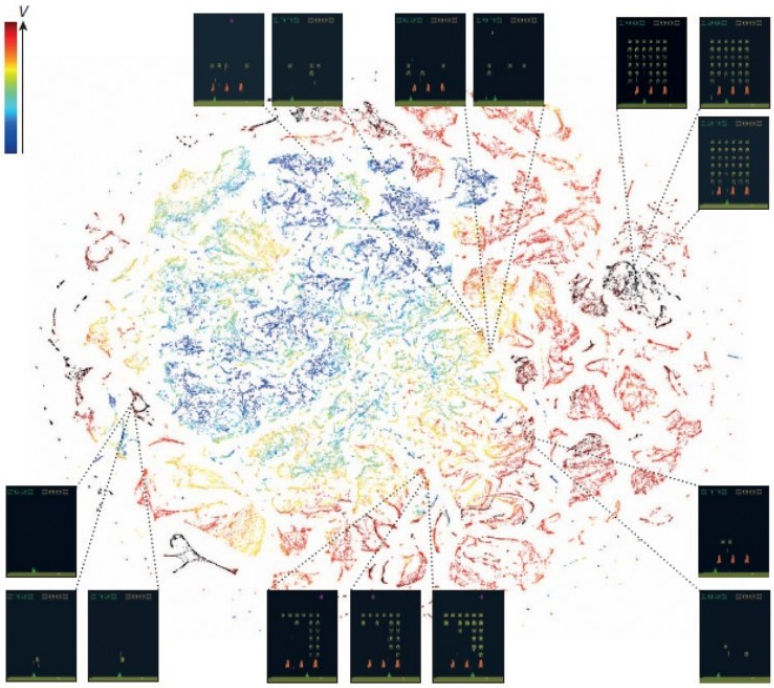
II. Mnih, V. *et al.* Human-level control... [Atari games] (cont'd)

- Results



II. Mnih, V. *et al.* Human-level control... [Atari games] (cont'd)

- Visualization



Experience Replay

II. Mnih, V. *et al.* Human-level control... [Atari games] (cont'd)

- Agent's experience at time t: $e_t = \{s_t, a_t, r_t, s_{t+1}\}$
- Dataset $D_t = \{e_1, e_2, ..., e_t\}$
- Sample minibatches (subsets) from stored data: $U(D)$
- Minimize objective function:

$$L_i(\boldsymbol{\theta}_i) = E_{(s,a,r,s')\sim U(D)} \left[\left(r + \gamma \max_{a'} Q(s', a'; \boldsymbol{\theta}_i^-) - Q(s, a; \boldsymbol{\theta}_i) \right) \right]$$

- Note:
  - iteration i, time t
  - $\boldsymbol{\theta}_i^-$ Network parameters when we took the move
  - $\boldsymbol{\theta}_i$ variable with respect to which we minimize

II. Mnih, V. *et al.* Human-level control... [Atari games] (cont'd)

When summarising:

- Note details in Supplementary material (style of journal)
- Core algorithmic setting
- How persuasive are results claimed
- What do they find by visualizing what the models learned (tSNE – similar to PCA, NMF)

III: Yuan, X. *et al.* Reinforcement Learning for Elevator Control

- Sum of rewards: $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1}$
- Bellman: $Q^*(x,u) - \sum_{x' \in \mathcal{X}} f(x,u,x') \left[\rho(x,u,x') + \gamma \max_{u' \in \mathcal{U}} Q^*(x',u')\right]$
- Algorithms (good revision material):

Table 1. The Q-value iteration algorithm

```
Input f, ρ, γ, convergence threshold θ
Initialize Q_0 arbitrarily, e.g. Q_0(x,u) = 0, for all x ∈ X, u ∈ U
k = 0
Repeat
    For each x ∈ X, u ∈ U
        Q_{k+1}(x,u) =
            Σ_{x'∈X} f(x,u,x')[ρ(x,u,x') + γ max_{u'∈U} Q_k(x',u')]
    EndFor
    k = k + 1
Until max_{x,u} |Q_k(x,u) − Q_{k−1}(x,u)| < θ
Output π*(x) = arg max_{u∈U} Q_k(x,u)    ∀x ∈ X
```

```
Input γ, α_t, exploration parameters (e.g., ε, τ)
Initialize Q_0 arbitrarily, e.g. Q_0(x,u) = 0, for all x ∈ X, u ∈ U
Initialize x_0
Repeat at each time step t:
    Choose u_t in x_t using policy derived from Q_t
                                (e.g, Boltzmann)
    Apply u_t, observe r_{t+1}, x_{t+1}
    Q_{t+1}(x_t,u_t) = Q_t(x_t,u_t) +
            α_t[r_{t+1} + γ max_{u∈U} Q_t(x_{t+1},u) − Q_t(x_t,u_t)]
```

- Q Learning: $Q_{t+1}(x_t,u_t) = Q_t(x_t,u_t) + \alpha_t \left[r_{t+1} + \gamma \max_{u \in \mathcal{U}} Q_t(x_{t+1},u) - Q_t(x_t,u_t)\right]$
- Convergence; exploration

Elevator

- System Description

  - The number of elevators (positive integer). In general, there are several elevators in elevator systems. In order to simplify the problem, we assume here that the system consists of a single elevator.
  - The number of floors (positive integer). Set here to 5.
  - The height of a floor (positive real). Set here to 6 m.
  - The elevator speed (positive real). Here we set it to 3 m/s. This means the elevator takes 2 s to travel between two adjacent floors.
  - The elevator capacity (positive integer). Set here to 4 passengers.
  - The stop time, i.e., the sum of the intervals needed by the passengers to enter and exit the elevator on a floor. The stop time is set to 2 s. The fact that the

Elevator

What you should look for in summarising

- StateSpace

The state space of the elevator system is discrete and has 7 dimensions. The state signal x is composed of the following variables:

$$x = [c_1, c_2, c_3, c_4, p, v, o]^T. \qquad (8)$$

where:

- c_i, $i = 1, 2, 3, 4$: Binary values, representing call requests (call flags) on each floor i. There is no call request on the ground floor in the down-peak traffic scenario.
- p: Discrete elevator position, taking values in $\{0, 1, 2, 3, 4\}$.
- v: Discrete vertical velocity taking values in $\{-3, 0, 3\}$ m/s.
- o: Discrete elevator occupancy, taking values in $\{0, 1, 2, 3, 4\}$. The number 0 means no passengers are inside the elevator; the number 4 means that the elevator is at its maximum capacity.

The cardinality of the state space is:

$$2^4 \cdot 5 \cdot 3 \cdot 5 = 1200 \qquad (9)$$

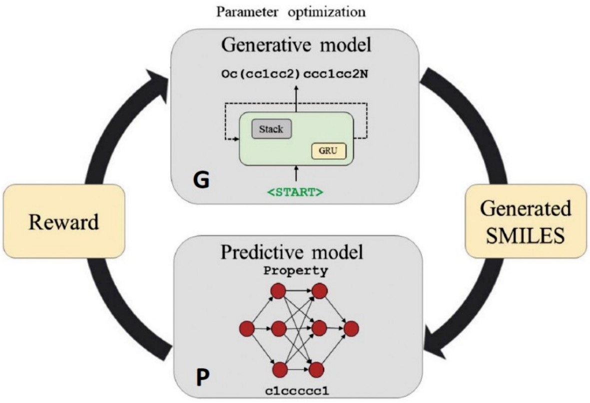
- Discrete action space; $\{-1, 0, 1\}$
- Constraints on actions
- Reward: $-\sum_{i=1}^{4} c_i - 0$; (call requests and occupancy)
- Passenger arrival/departure models to simulate;
- Evaluation

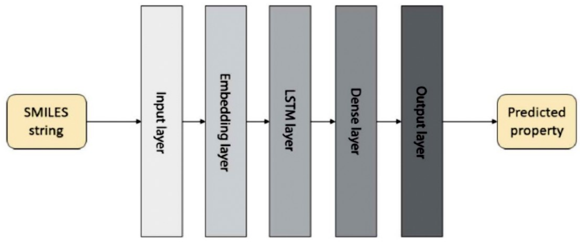
IV: Popova, M. *et al.* Deep reinforcement learning for *de novo* drug design

- Biology, chemistry offer challenging machine learning problems
- Easy to construct molecules (synthetic chemistry)
- Difficult to experimentally vaildate their properties
- Learn from molecules whose properties are known; test new ones.
- Drug design: Molecular structure, representation, function prediction

Drug design (cont'd)

- A generative model and a predictive model!





Drug Design (cont'd)

What to look for in the paper

- States as representation of molecules as strings
- Terminal states when length T is reached
- Reward from the predictive model for a string of lenth T
- Policy $p(a_t|s_{t-1})$ enables the string to grow (the generative model)
- Policy network: $J(\Theta) = \sum_{s_T \in S^*} p_\Theta(s_T) r(s_T)$
- Solved by Policy Gradient methods (Section 13.1, Sutton and Barto)
  - $\pi(a|s, \boldsymbol{\theta})$, diffrentiable function of $\boldsymbol{\theta}$
  - REINFORCE class of algorithms

REINFORCE Algorithm

Monte Carlo Policy Gradient

- Cost function $J(\boldsymbol{\theta})$ and its gradient $\nabla J(\boldsymbol{\theta})$
- Stochastic gradient descent $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha, \widehat{\nabla J(\boldsymbol{\theta})}$
- Parameterize the policy (softmax):

$$\pi(a|s, \boldsymbol{\theta}) = \frac{\exp(h(s, a, \boldsymbol{\theta}))}{\sum_b \exp(h(s, b, \boldsymbol{\theta}))}$$

- Policy gradient theorem ** skip **
- Algorithm: Eqn 13.7 & 13.8 of S & B.
- Monte Carlo: run multiple episodes and average (as seen before)

IV: Mao, H. *et al.* Resource management

- Captures ideas we discussed around the elevator and drug design problems.
- Optimize allocation of jobs to clusters in computing
  - Optimization problems
  - Gradient descent on (parameterized) policy
- See how the REINFORCE algorithm is specified in Eqn 2

$$\theta \leftarrow \theta + \alpha \sum_t \nabla_\theta \log \pi_\theta(s_t, a_t) v_t$$

- Sample multiple trajectories and estimate reward v_t
- Look for simulation details and performance in review.