PHY 835: Machine Learning in Physics Lecture 22: Reinforcement Learning Part 2 April 11, 2024

Gary Shiu

Fitted Q-learning

- Instead of tabulating the action-values (table size grows as $|S|^2 |A|$ $|S|, |A|$ are sizes of state & action spaces), we can learn with a NN. action space is small. Units is represented that is represented to the constraint in the constraint in the constrained to constrain the constraint in $\mathcal{O}(\epsilon,|\Lambda|)$ are sizes virstate α activitispaces), we can iediti $T_{\rm c}$, $T_{\rm c}$ and $T_{\rm c}$ and $T_{\rm c}$ are peated above repeated above repeated above $T_{\rm c}$ ad of tabulating the action-values (table size grows as $|D|/|A|$ $\vert A\vert$ are sizes of state & action spaces), we can learn with a NN. environment of a chessboard, there are more than 1040 possible legal states. The more than 1040 possible legal
The more than 1040 possible legal states are more than 1040 possible legal states. The more than 1040 possible
- Replace the action values $q[\mathbf{s}_t, a_t]$ by a ML model $q[\mathbf{s}_t, a_t, \phi]$. $\frac{1}{2}$ c piace life action values $q[\mathbf{s}_t, a_t]$ by a ivic model $q[\mathbf{s}_t, a_t, \psi]$. e the action values $a[s, a]$ by a ML model $a[s, a, b]$. \mathbf{r} and \mathbf{r} are not the state is represented by a vector is represented by a
- Loss function which measures consistency of adjacent action values: s*^t* rather than just an index. We then defne a least squares loss based on the consistency oss function which measures consistency of adjacent $\it i$ function which measures consistency of adjacent action values of adjacent action values (similarly to in Q-learning, see equation 19.15):

$$
L[\boldsymbol{\phi}] = \bigg(r[\mathbf{s}_t, a_t] + \gamma \cdot \max_a \Big[q[\mathbf{s}_{t+1}, a, \boldsymbol{\phi}]\Big] - q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]\bigg)^2,
$$

which in turn leads to an update: hich in turn leads to an upda

$$
\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha \bigg(r[\mathbf{s}_t, a_t] + \gamma \cdot \max_a \Big[q[\mathbf{s}_{t+1}, a, \boldsymbol{\phi}] \Big] - q[\mathbf{s}_t, a_t, \boldsymbol{\phi}] \bigg) \frac{\partial q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]}{\partial \boldsymbol{\phi}}.
$$

• Convergence is not guaranteed. A change to the parameters modifies both the target $r[\mathbf{s}_t, a_t] - \gamma \cdot \max_{a} q[\mathbf{s_{t+1}}, \mathbf{a}, \phi]$ & the prediction $q[\mathbf{s}_t, a_t, \phi]$. Fitted Q-learning difers from Q-Learning in that convergence is no longer guar**anticeded. A change to the parameters modified. A change to the parameters modified both the target of the target of** \overline{a} **and** \overline{b} **and** \overline{a} **are** \overline{b} $T = \frac{1}{2}$ both theoretically and empirically theoretically to damage convergence. $\mathbf{F} = \mathbf{F} \cdot \mathbf{F} = \mathbf{F} \cdot \mathbf{F} = \mathbf{F} \cdot \mathbf{F} = \mathbf{F} \cdot \mathbf{F} = \mathbf{F} \cdot \mathbf{F}$ anteed. A change to the parameters potentially modifes both the target *r*[s*t, at*] + γ *·* $\frac{1}{\sqrt{1-\frac{1$

Deep Q-networks

- Use deep NN for fitted Q-learning. Q stands for action-value $q[s_t, a_t, \phi]$.
- **Deep Q-network** was a RL architecture that exploited deep NN to learn to play ATARI 2600 games.

Single frame does not specify velocity \Rightarrow 4 adjacent frames to represent a state

> 18 possible actions (9 directions, on/off)

• Issue of convergence was alleviated by fixing the target parameters to ϕ [−] and only updating them periodically. Only update the prediction: and only updating them periodically. Only update the l between the samples in the batch that arise due to the similarity of adjacent frames. Finally, the issue of convergence in ftted Q-Networks was tackled by fxing the target parameters to value them periodically. This gives the update: Update:: This gives the update: Update: Update:
This gives the update: This gives the update: Update: Update: Update: Update: Update: Update: Update: Update:

$$
\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} + \alpha \bigg(r[\mathbf{s}_t, a_t] + \gamma \cdot \max_a \Big[q[\mathbf{s}_{t+1}, a, \boldsymbol{\phi}^-] \Big] - q[\mathbf{s}_t, a_t, \boldsymbol{\phi}] \bigg) \frac{\partial q[\mathbf{s}_t, a_t, \boldsymbol{\phi}]}{\partial \boldsymbol{\phi}}.
$$

• Not chasing a moving target; less prone to oscillations. $\overline{}$ *r*_{*m*} *moving* ta " *q*[s*t*+1*, a,* φ[−]] # $prone$ *τ* \$∂*q*[s*t, at,* φ] $\,$ s $\,$ cillations. Now the network no longer chases a moving target and is less prone to oscillation. Using these and other heuristics and with an \$-greedy policy, Deep Q-Networks per-

Policy gradient methods

- Recall the notions of value estimation vs policy estimation. Q-learning is an example of value estimation: estimate $q[s_t, a_t, \phi]$ and update π .
- **Policy-based methods** directly learn a stochastic policy $\pi[a_t | s_t, \theta]$.
- For MDP, there is always an optimal deterministic policy.
- There are reasons to use instead a stochastic policy:
	- **Exploration of the action-state space:** not obliged to take the best action at each step.
	- **• Loss function changes smoothly:** can use gradient descent.
	- **• Knowledge of the state is often incomplete:** two locations may look locally the same but nearby reward structure is different. Stochastic policy: taking different actions until ambiguity resolved.

Gradient update **19.5.1** Consider a tradient update this trajectory *P r*(τ *|*θ) depends on both the state evolution function *P r*(s*t*+1*|*s*t, at*) and

- Consider a trajectory $\tau = [\mathbf{s}_1, a_1, \mathbf{s}_2, a_2, ..., \mathbf{s}_T, a_T]$ through an MDP. Consider a trajectory τ = [s1*, a*1*,* s2*, a*2*,...,* s*^T , a^T*] through an MDP. The probability of the current stochastic policy π[*at|*s*t,* θ]: \mathbf{r} **e** Consider a trajectory $\tau =$ $\left[$ s, a , s₀, a ₂, σ ₂, a ¹ throu \bullet Current a trajectury
	- The probability of this trajectory depends on the current policy: lep *rajectory depends on the current policy:*

$$
Pr(\boldsymbol{\tau}|\boldsymbol{\theta}) = Pr(\mathbf{s}_1) \prod_{t=1}^T \pi[a_t|\mathbf{s}_t, \boldsymbol{\theta}] Pr(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t).
$$

• Policy gradient algorithms aim to maximize the expected return *r*[*τ*] over many such trajectories: Policy gradient algorithms aim to maximize the expected return *r*[τ] over many such **• Policy gradient algorithms aim to maximize the expected** *r* trajectories:

$$
\boldsymbol{\theta} = \operatorname*{argmax}_{\boldsymbol{\theta}} \bigg[\mathbb{E}_{\boldsymbol{\tau}} \big[r[\boldsymbol{\tau}] \big] \bigg] = \operatorname*{argmax}_{\boldsymbol{\theta}} \bigg[\int P r(\boldsymbol{\tau} | \boldsymbol{\theta}) r[\boldsymbol{\tau}] d \boldsymbol{\tau} \bigg],
$$

- The return is the sum of all the rewards received along the trajectory. θ = argmax θ E^τ *r*
[τ] θ *P r*(τ *|*θ)*r*[τ]*d*τ *,* (19.23) • The return is the sum of all the rewards received along the traj
	- To maximize the return, we use the gradient ascent update: where the return is the sum of all the rewards received along the trajectory. maximize the return, we use the gradient ascent To maximize this quantity, we use this quantity, we use the gradient ascent update: \mathbf{u}_i

$$
\theta \leftarrow \theta + \alpha \cdot \frac{\partial}{\partial \theta} \int Pr(\tau | \theta) r[\tau] d\tau
$$
\n
$$
= \theta + \alpha \cdot \int \frac{\partial Pr(\tau | \theta)}{\partial \theta} r[\tau] d\tau.
$$
\n\alpha = learning rate

• \approx this integral with a sum over en and divide the integrand by this distribution: policy should be modifed to avoid prothis integral with a sum over receives high rewards *and* is unusual. his integral with a sum over rno milograf willt a oarn ovor

where α is the learning rate.

 $f(x) = \int_{0}^{\infty} x^{x} dx$

there is no need to change. Converse to change the converse \sim

there is no need to change. The converse $\mathbf S$

wards, but similar trajectories already already already already already already already already already already

$$
\theta \leftarrow \theta + \alpha \cdot \int \frac{\partial Pr(\tau)}{\partial \theta} \n= \theta + \alpha \cdot \int Pr(\tau | \theta) \frac{1}{Pr(\tau | \theta)} \frac{\partial Pr(\tau | \theta)}{\partial \theta} r[\tau] d\tau \n\approx \theta + \alpha \cdot \frac{1}{I} \sum_{i=1}^{I} \frac{1}{Pr(\tau_i | \theta)} \frac{\partial Pr(\tau_i | \theta)}{\partial \theta} r[\tau_i].
$$

• Using identity involving log, we can simplify the update on θ : rameters θ to increase the likelihood *P r*(τ *ⁱ|*θ) of an observed trajectory τ *ⁱ* in proportion which yields the update update.

$$
\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I}\sum_{i=1}^I \frac{\partial \log \bigl[\mathit{Pr}(\boldsymbol{\tau}_i|\boldsymbol{\theta})\bigr]}{\partial \boldsymbol{\theta}} \mathit{r}[\boldsymbol{\tau}_i].
$$

• The log probability is given by the sum of logs: yields high rewards, then we don't need to change much. The biggest updates will come he log probability is given by the sum of logs: \overline{P} and probability is given by the sum of legs

$$
\log[Pr(\boldsymbol{\tau}|\boldsymbol{\theta})] = \log [Pr(\mathbf{s}_1) \prod_{t=1}^T \pi[a_t|\mathbf{s}_t, \boldsymbol{\theta}] Pr(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)]
$$

=
$$
\log[Pr(\mathbf{s}_1)] + \sum_{t=1}^T \log[\pi[a_t|\mathbf{s}_t, \boldsymbol{\theta}]] + \sum_{t=1}^T \log[Pr(\mathbf{s}_{t+1}|\mathbf{s}_t, a_t)],
$$

Gradient update *P r*(s1) +! *t*=1 log" $\frac{1}{2}$ and noting that only the center term depends on G radiant update from G

• Only the policy $\pi[a_t | s_t, \theta]$ term depends on θ : ϵ as ϵ as: $\overline{1}$ *III* αepenas on σ:

$$
\boldsymbol{\theta} \;\leftarrow\; \boldsymbol{\theta} + \alpha \cdot \frac{1}{I} \sum_{i=1}^{I} \sum_{t=1}^{T} \frac{\partial \log[\pi[a_{it}|\mathbf{s}_{it},\boldsymbol{\theta}]]}{\partial \boldsymbol{\theta}} r[\boldsymbol{\tau}_i],
$$

- Since the state evolution $Pr(\mathbf{s}_{t+1} | \mathbf{s}_t, a_t)$, parameter update does not assume Markov time evolution process. $(9.8⁺)$ • Since the state evolution $Pr(s_{i+1} | s_i, a_i)$, parameter update does not disappear, this parameter update does not assume a Markov time evolution process.
- The total reward can be expressed as a sum of two contributions: in episode *i*. Note that since the terms relating to the state evolution *P r*(s*t*+1*|*s*t, at*) The total reward can be expressed as a sum of two contributions: The total reward can he expressed as • The total reward can be expressed as a sum of two contributions:

$$
r[\boldsymbol{\tau}_i] = \sum_{t=1}^T r_{it} = \sum_{k=1}^{t-1} r_{ik} + \sum_{k=t}^T r_{ik},
$$

• The first term does not affect the update, thus:

$$
\boldsymbol{\theta} \;\gets\; \boldsymbol{\theta} + \alpha \cdot \frac{1}{I} \sum_{i=1}^I \sum_{t=1}^T \frac{\partial \log[\pi[a_{it}|\mathbf{s}_{it},\boldsymbol{\theta}]]}{\partial \boldsymbol{\theta}} \sum_{k=t}^T r_{ik}.
$$

REINFORCE Algorithm θ ← θ + α *· I i*=1 ∂θ *k*=*t* θ ← θ + α *·* **1** *I* **1**<u>RCE</u> Algori: ∂θ \cdot h *rik.* (19.31)

- A policy gradient algorithm that incorporates discounting.
- Use Monte Carlo to generate episodes $[s_{i1}, a_{i1}, r_{i2}, s_{i2}, a_{i2}, r_{i3}, ..., r_{iT}]$ based on the current policy $\pi[a \mid s, \theta]$. corporates discounting. It is a Monte Carlo method that generates episodes τ *ⁱ* = *REINFORCER* is a set of the policy $\mathcal{W}[W | \mathcal{Y}, \mathcal{Y}]$
	- $\pi[a|s,\theta]$ takes the current state & returns one output for each action. tions, this policy could be determined by a new angles that takes the current and takes the current and takes t
Policy could be determined by a new angles the current angles the current and takes the current and takes the [s*i*1*, ai*1*, ri*2*,* s*i*2*, ai*2*, ri*3*,...,riT*] based on the current policy π[*a|*s*,* θ]. For discrete ac-
	- The outputs ($|A|$ dim.) are passed through a softmax function to create a distribution over actions, which is sampled at each time step. through a softmax function to create a distribution over actions, which is sampled at electe d'une $r_{\rm H}$ re outputs ($\vert A \vert$ unii.) are passed through a solutiax function to reate a distribution over actions, which is sampled at each time ste
	- \bullet For each episode i , calculate the empirical discounted return for each τ itrajectory τ_{it} that starts at time t : return for the partial trajectory τ *it* that starts at time *t*: each episode *i*, calculate the empirical discounted return for each

$$
r[\boldsymbol{\tau}_{it}] = \sum_{k=t+1}^{T} \gamma^{k-t-1} r_{ik},
$$

and then we update the parameters for each step t in each trajectory:

$$
\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \gamma^t \frac{\partial \log[\pi_{a_{it}}[\mathbf{s}_{it}, \boldsymbol{\theta}]]}{\partial \boldsymbol{\theta}} r[\boldsymbol{\tau}_{it}] \qquad \forall i, t
$$

Baselines Notebook 19.5 and the expected value with increduces with increduces with increduces with irrelevant Control variates factors that add uncertainty, then subtracting it reduces that add uncertainty it reduces the variance (

- Drawback of policy gradient methods: high variance; many episodes may be needed to get stable updates of the derivatives. **19.5.3 Baselines** is a special case of the method of *control variates* (see problem 19.7). Problem 19.7 Jrawback of policy gradient methods: high variance; many episodes
- To reduce the variance, we subtract the returns from a baseline: o reduce the variance, we subtract the returns from a $\mathfrak k$ with respect to \overline{p} respect the result to \overline{p} and \overline{p} and \overline{p} is \overline{p} .

$$
\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I} \sum_{i=1}^{I} \sum_{t=1}^{T} \frac{\partial \log[\pi_{a_{it}}[\mathbf{s}_{it}, \boldsymbol{\theta}]]}{\partial \boldsymbol{\theta}} (r[\boldsymbol{\tau}_{it}] - b).
$$

• The baseline is often taken to be:

$$
b=\frac{1}{I}\sum_i r[\boldsymbol{\tau}_i].
$$

• Subtracting this baseline factors out variance that might occur when the trajectories happen to pass through states with higher than average returns. through states with higher than average returns *whatever* actions are taken.

Actor-critic methods *i*=1 *t*=1 *v*[sit*i*, φ] − **.** *j*=*y rij*

- Actor-critic algorithms: temporal difference policy gradient algorithms. Actor-critic algorithms are temporal diference (TD) policy gradient algorithms. They
- Parameters of the policy network are updated at each time step, in contrast with Monte Carlo REINFORCE algorithms. the Monte Carlo REINFORCE algorithment with $\frac{1}{2}$ complete with provide before the parameters. **19.6 Actor-critic methods** rast with Monte Carlo REINFORCE algorithms. The monodernight are the control of the T can update the parameters of the policy network at each step. This contrasts with
- We do not have access to the future rewards along the trajectory. along the thermal disclose to the future terres algorithms appears, \mathbf{y} . the Monte Carlo and Monte Series algorithm and the *inalectory*.
- Approximate the sum over all the future rewards with:

$$
\sum_{k=1}^{T} r[\boldsymbol{\tau}_{ik}] \approx r_{it} + \gamma \cdot v[\mathbf{s}_{i,t+1}, \boldsymbol{\phi}].
$$

- The value $v[s_{i,t+1}, \phi]$ is estimated by a second NN with parameter ϕ . The value $v[x_1, t_1]$ is estimated by a second NN with parameter ϕ $\sum_{i=1}^n \sum_{j=1}^n (t+1)^j$ is commuted by a cool. k $\mathcal{L} = \mathcal{L}$, $\mathcal{L} \mathcal{L}$ $\mathcal{L} = \mathcal{L}$
- This gives the update:

$$
\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \cdot \frac{1}{I}\sum_{i=1}^I\sum_{t=1}^T \frac{\partial \log \bigl[Pr(a_{it}|\mathbf{s}_{it},\boldsymbol{\theta})]\bigr]}{\partial \boldsymbol{\theta}} \Big(r_{it} + \gamma \cdot v[\mathbf{s}_{i,t+1},\boldsymbol{\phi}] - v[\mathbf{s}_{i,t},\boldsymbol{\phi}]\Big).
$$

Actor-critic methods *T* ∂ log" *Pr*(*ait|*s*it,* ^θ)]#

• Concurrently, we update the parameter ϕ using the loss function:

$$
L[\boldsymbol{\phi}] = \sum_{i=1}^{I} \sum_{t=1}^{T} (r_{it} + \gamma \cdot v[\mathbf{s}_{i,t+1}, \boldsymbol{\phi}] - v[\mathbf{s}_{i,t}, \boldsymbol{\phi}])^{2}.
$$

- The policy network $\pi[\mathbf{s}_t, \theta]$ that predicts $Pr(a | \mathbf{s}_t)$ is term the **actor**. $\mathbf{P}(\mathbf{p}|\mathbf{p})$
- The value network $v[s_t, \phi]$ is termed the **critic**.
- Actor-critic methods can update the policy parameter at each step.
- In practice, the agent typically collects a batch of experience over many time steps before the policy is updated.

Nature or Nurture?

Genetic Algorithm: The basic Idea

- **mutation rate**
- **further attributes**

The String Landscape Inverse Problem The String Landscape Inverse Pro

 (topologies of compactification, number of branes and wrapping numbers, quantized fluxes, ….) \dot{N} \equiv (topologies of compactification, number of branes and wrapping numbers, =

 $=\int_{0}^{t_{0}}$

 \overrightarrow{N} =

Understanding the structure of the Landscape Understanding the cardiocentry of the structure of the structure of the string landscape

<https://arxiv.org/abs/1907.10072>

<https://arxiv.org/abs/2111.11466>

Genetic Algorithms + RL

• Correlations:

- GAs and RL have also been used to optimize the search for realistic particle physics models: <https://arxiv.org/abs/2112.08391> \sim Some correlations are obvious as pairs of fluxes contribute as products to the tasks to the tadpoles. extigations under the there is no task of the task o
- Using dynamic programming, we can even count the exact number of solutions: <https://arxiv.org/abs/2206.03506> • Comparing individual correlation maps with the combined one can unpack how GA & RL Using dyn

Learning from Topology:

Topological **Aata Analysis**: Topological aata shalysis:

From String Theory to Cosmology to Phases of Matter have anomalously large numbers (like S Fr above), and Fr Mottor

String Landscape <https://arxiv.org/abs/1812.06960> <https://arxiv.org/abs/1907.10072>

Cosmology <https://arxiv.org/abs/1710.04737> <https://arxiv.org/abs/2009.04819> <https://arxiv.org/abs/2308.02636> <https://arxiv.org/abs/2403.13985>

Phases of Matter <https://arxiv.org/abs/2009.14231>

Learning from Topology

