Announcements

① This is the last lecture

(Woohoo! You made it thru)

I'll be here next week, same time if you want to chat

② Final projects due in 1 week!

Deep Reinforcement Learning

Today we'll talk about applications of deep learning to <u>sequential prediction</u>

First let's start with the classic setup

<u>Markov Decision Processes</u>

A common abstraction of discrete time stochastic control, applications robotics, games, and economics

Specified by:

1. A finite set of <u>states</u>, S

2. A finite set of <u>actions</u>, A

③ transition probabilities

$\mathbb{P}[s' | s, a]$ = Probability of transitioning to s' after playing a in s

④ rewards

$R(s', a, s)$ = reward of transit. to s' by playing a in s

⑤ a discount factor $\gamma < 1$

Goal: Maximize the expected reward

def: A policy π is a distribution $\pi(\cdot, s)$ on actions, for any state $s \in S$

With a policy in hand, we can take the expectation over trajectories τ where

$$\tau = (s_0, \dots s_T ; a_0, \dots a_T)$$

The key point is the choice of the next state is Markovian

$$\mathbb{P}[S_i = s' \mid s_0, \dots s_{i-1}, a_0 \dots a_{i-1}] = \mathbb{P}[s' \mid s_{i-1}, a_{i-1}]$$

and we can define the total reward

$$R(\tau) = \sum_{i=0}^{\infty} \gamma^i R(s_i, a_{i-1}, s_{i-1})$$

Main Question: Is there an eff. algorithm to find an optimal policy?

i.e. we want to maximize $\mathbb{E}_{\pi}[R(\tau)]$

Yes!

Note: This is called the tabular setting because we can afford to write down a table of values of states, or even state-action pairs

Later we will discuss what happens when the set of possible states becomes too large (could even be infinite)

Example: Modified Blackjack

1. all cards face up
2. dealer goes first
3. infinitely large deck

otherwise need state to model what cards are left

Actions: hit, stay, split, ...

Characterizing Optimal Policies

Let's introduce some helper valuations

def: The value function of π is

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[R(\tau) \mid s_0 = s\right]$$

i.e. the expected reward of playing policy π, starting from s

def: The Q-function of π is

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}\left[R(\tau) \mid s_0 = s, a_0 = a\right]$$

i.e. the expected reward of starting from s and playing a, and playing according to π there after

Abusing notation, let's write

$$Q^{\pi}(s, \pi') = \sum_{a \in A} \pi'(s, a)\, Q^{\pi}(s, a)$$

Now let's study how to improve a policy

<u>Theorem</u> [Policy Improvement] Let π and π be policies s.t.

$$\forall_{s \in S} \quad Q^{\pi}(s, \pi') \geq Q^{\pi}(s, \pi) \qquad (1)$$

Then we have

$$\forall_{s \in S} \quad V^{\pi'}(s) \geq V^{\pi}(s) \qquad (2)$$

Moreover (1) is strict for some s, then so is (2)

Proof: First note that by definition

$$Q^{\pi}(s,\pi) = V^{\pi}(s)$$

Now we can consider hybrid "policies" that

1. play according to π' up to but not including time i
2. then switch to π thereafter

Call this "policy" π_i

Now consider the difference in value functions

$$V^{\pi_{i+1}}(s) - V^{\pi_i}(s) =$$

$$\sum_{s' \in S} \gamma^{i} \left(Q^{\pi}(s',\pi') - Q^{\pi}(s',\pi)\right) \mathbb{P}[s_i = s' \mid s_0 = s]$$

From (i), we have

$$V^{\pi_{i+1}}(s) - V^{\pi_i}(s) \geq 0$$

Now using telescoping sums, we get:

$$\sum_i V^{\pi_{i+1}}(s) - V^{\pi_i}(s) = V^{\pi'}(s) - V^{\pi}(s) \geq 0$$

which completes the proof. ☒

Now we are ready to discuss optimality

<u>Theorem 2</u> [Bellman Optimality] A policy π is <u>optimal</u>* iff

$$\pi(s,a) > 0 \Rightarrow a \in \operatorname*{argmax}_{b \in A} Q^{\pi}(s,b)$$

Actually the notion of optimality is subtle

We say π is optimal if for any other policy π' we have

$$V^{\pi}(s) \geq V^{\pi'}(s) \quad \forall s \in S$$

Otherwise can have two policies that both maximize the reward but differ only on a state that can't be reached

We'll omit the proof of theorem 2

But note that this tells us that an optimal policy defines a <u>fixed point</u>

In particular, consider the <u>Bellman operator</u>:

$$\mathcal{B} : \begin{matrix}\text{function from}\\ S \times A \to \mathbb{R}\end{matrix} \to \begin{matrix}\text{function from}\\ S \times A \to \mathbb{R}\end{matrix}$$

which is defined as

$$\mathcal{B}Q(s,a) = \sum_{s' \in S} \mathbb{P}[s' \mid a, s]\left(R(s',a,s) + \gamma \max_{a' \in A} Q(s',a')\right)$$

Now let Q^* satisfy $\mathcal{B}Q^*(s,a) = Q^*(s,a)$ (3)

Fact: There is a unique soln. to (3)

Moreover we can read off an optimal policy from Q^* as follows

$$\pi^*(s) = \arg\max_{a} Q^*(s,a)$$

And we can prove that a natural alg. converges to Q^*

Q-Learning

For $t = 1$ to T

Update: $Q(s,a) \leftarrow (1-\eta)\, Q(s,a)$

$$+\eta\Big(\sum_{s'} \mathbb{P}[s' | a, s]\, R(s',a,s) + \gamma \max_{a' \in A} Q(s',a')\Big)$$

Alternatively, can use dynamic programming for a fixed time horizon H

Can also learn with the V-function

Value Iteration

For t=1 to T

Update: $V(s) \leftarrow (1-\eta) V(s)$

$$+ \eta \left(\max_a \sum_{s'} \mathbb{P}[s' | a, s] \left(R(s', a, s) + \gamma V(s') \right) \right)$$

In fact, we can even use linear prog.

$$\min \sum_s V(s)$$

$$\text{s.t. } V(s) \geq \sum_{s'} \mathbb{P}[s' | a, s] \left(R(s', a, s) + \gamma V(s') \right) \quad \forall s, a$$

There is an important distinction between these algorithms

Main Question: what if we don't know transitions/rewards but still want to learn an optimal policy?

e.g. by getting to run the MDP starting from s and playing a, called a <u>simulator</u>

or by getting to play a policy π and observing a trajectory, called <u>online RL</u>

Algorithms fall into one of two categories

<u>Model-based</u>: Algorithm explicitly learns the transitions/rewards, then can do value iteration, ...

<u>Model-free</u>: Algorithm does not explicitly do this, but if each (s,a) are observed enough, can still do Q-learning

Main Point: For huge games, can't afford to learn and store full MDP

i.e. $|S| \times |A| \times |S|$ table can be way bigger than an $|S| \times |A|$ table

Algorithms that run in $\tilde{O}(|S||A|)$ time are called sublinear

Parameterized Policies

Main Question: where does deep learning fit in?

In order to deal with huge MDPs, modern RL uses function approximation

... both for Q-functions

$$Q_\theta(s,a) = \underset{\substack{\uparrow \\ \text{deep net}}}{f_\theta}(\phi(s,\underset{\substack{\uparrow \\ \text{feature mapping}}}{a}))$$

and sometimes even for policies

$$\pi_\theta(s,\cdot) = f_\theta(\phi(s))_{\substack{\uparrow \\ \text{output a distr.}}}$$

<u>Use Case</u>: In Go, the number of possible actions is small but the state space is enormous (3^{361})

... a typical new game can reach a board configuration you've never seen before!

<u>The Hope</u>: There is a feature mapping that allows you to learn from "similar" configurations you have seen

With this generalization come a host of new issues

Main Question: How do we search among a restricted / parameterized set of policies?

First let $J(\theta) = \mathbb{E}_{\pi_\theta}[R(\tau)]$

Policy Gradient Method

```
For t = 1 to T
    update $\theta_{t+1} \leftarrow \theta_t + \eta \nabla J(\theta_t)$
```

Main Question: But how do we compute this gradient?

Amazingly, we can do this in a model-free way!

Theorem 3 [Policy Gradient]

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta}\left[R(\tau) \nabla \log \underset{\substack{\uparrow \\ \text{total prob. of trajectory} \\ \tau \text{ using } \pi_\theta}}{\pi_\theta(\tau)}\right]$$

Proof: Let's directly compute the gradient:

$$\nabla \mathbb{E}_{\pi_\theta}[R(\tau)] = \nabla \int \pi_\theta(\tau) R(\tau) \, d\tau$$

$$= \int R(\tau) \nabla \pi_\theta(\tau) \, d\tau$$

b/c the reward doesn't depend on θ

$$= \int R(\tau) \pi_\theta(\tau) \nabla \log \pi_\theta(\tau) \, d\tau$$

$$= \mathop{\mathbb{E}}_{\pi_\theta}\left[R(\tau)\nabla \log \pi_\theta(\tau)\right]$$

$\blacksquare$

But why doesn't this depend on knowing the model?

we can write out the probability of a trajectory

$$\pi_\theta(\tau) = \prod_i \pi_\theta(a_i|s_i)\,\mathbb{P}[s_{i+1}|s_0, a_i]$$

Now taking the log, we have

$$\log \pi_\theta(\tau) = \sum_i \log \pi_\theta(a_i|s_i) + \sum_i \log \mathbb{P}[s_{i+1}|s_i, a_i]$$

$$\Rightarrow \nabla \log \pi_\theta(\tau) = \sum_i \nabla \log \pi_\theta(a_i|s_i)$$

Thus the policy gradient method becomes

$$\theta_{t+1} \leftarrow \theta_t + \eta \underset{\pi_{\theta_t}}{\mathbb{E}}\left[R(\tau)\left(\sum_i \nabla \log \pi_{\theta_t}(a_i|s_i)\right)\right]$$

This is called an on-policy method b/c you need trajectories from θ_t at each step

Can also give an off-policy method that uses samples from θ, but has a correction term

$$\frac{\pi_{\theta_t}(\tau)}{\pi_\theta(\tau)}$$

Some issues to worry about

① Does policy gradient converge to an optimal policy?

Even in the tabular setting, this turns out to be tricky

Agarwal, Kakade, Lee, Mahajan showed:

Theorem [informal] Under certain regularity assumptions, any first-order stationary point is approx. optimal

In more detail, in the tabular setting they showed

Theorem: Projected gradient ascent with a softmax parameterization takes at most
$$O\left(\frac{D_\infty^2 |S||A|}{(1-\gamma)^6 \varepsilon^2}\right)$$
to find an ε-optimal policy

Here $\underset{\text{mismatch coefficient}}{D_\infty^2} = \max_s \frac{d_{s_0}^{\pi^*}(s)}{\mu(s)}$ ← fraction of time π^* spends in s

$\mu(s)$ ← initialize P.g.a with μ is starting distr. on s_0

The main idea is to give some scalar valued measure $G(\theta)$ of first-order stationarity that satisfies

$$J(\underset{\substack{\uparrow \\ \text{parameters of an optimal policy}}}{\theta^*}) - J(\theta) = O(G(\theta))$$

Polyak-like gradient domination condition

This implies there are no bad critical points, and the analysis then revolves around bounding how many steps you need to reach an approx. critical point

In particular, they show

$$J^*(\underset{\substack{\uparrow \\ \text{distribution on states}}}{\rho}) - J^\pi(\rho) \leq \left\| \frac{d\rho^{\pi^*}}{d_\mu^\pi} \right\|_\infty \max_{\pi'} (\pi' - \pi)^T \nabla_\pi V^\pi(\mu)$$

Earlier work of Fazel, Ge, Kakade,

Mesbahi studied convergence for the linear quadratic regulator

Bhandari and Russo also gave convergence guarantees in the tabular setting under various conditions

Moreover they gave an example that shows things can go wrong in general

Fact 2 There is a two state MDP with restricted set of policies param. by θ which contains an optimal policy, but $J(\theta)$ has spurrious local max.

Now let's talk about deep Q-learning

Q-Learning with Function Approx.

Repeat for $t = 1$ to T

Take some action a at S, and observe reward R and next state s'

Set $y = R + \gamma \max_{a'} Q_{\theta_t}(s', a')$

Update:

$$\theta_{t+1} \leftarrow \theta_t + \eta \left(Q_{\theta_t}(s,a) - y \right) \nabla_\theta Q_\theta(s,a) \Big|_{\theta_t}$$

<u>Intuition</u>: This is just SGD on the squared loss, measuring the fit btwn. our estimated Q-values and the corrected observed rewards

Also need to be careful about how we initialize / generate samples so that we <u>explore</u>

Epilogue

<u>Main Question</u>: Can we prove generalization guarantees for online RL?

In the tabular setting:

<u>Theorem</u> [Kearns, Singh] There is an algorithm that learns an ε-optimal policy with $\text{poly}(|S|, |A|, \underset{\substack{\uparrow \\ \text{horizon}}}{H}, 1/\varepsilon)$ samples

Since then, many refinements incl.

optimal <u>regret</u> bounds. See Azar, Osband, Munos who get

$$\tilde{O}(\sqrt{H|S||A|T})$$

Their algorithm is model-based, but by now the same bounds are known for model-free

While proving guarantees for deep Q-learning is tough, there are strong bounds in the case of <u>linear</u> function approx.

i.e. there is a choice of parameters θ^* s.t.

$$\underset{\substack{\uparrow \\ \text{optimal Q-function}}}{Q^*}(s,a) = \theta^{*T} \phi(s,a)$$

and there is a signed measure μ^* s.t.

$$\mathbb{P}[\cdot \mid s,a] = \mu^{*T} \phi(s,a)$$

and we <u>know the feature map ϕ</u>

<u>Goal:</u> Replace dependence on $|S|$ with d, where d is the dimension of $\phi(s,a)$

See the work of Jin, Yang, Wang, Jordan