



Dynamic Programming and Reinforcement Learning

Week 6b: Policy Gradients

Rainer Schlosser und Alexander Kastius
Enterprise Platform and Integration Concepts

20.05.21

Continuous Control



Already solved:

- State Space Complexity
 - Many Dimensions
 - Continuous Values
- Current methods require discretization and become intractable at some point

■ Continuous Control

- Action Space might consist of continuous values as well
- Can be discretized sometimes, which prevents us from finding the actual optimal policy



Effects Of Continuous Action Spaces

- Maximization operation becomes impossible, can only be approximated by discretization.
- Discretization does introduce a penalty in the achievable reward, unless the decision problem is very well formulated in that regard
- Discretization does have to relearn that actions are related, something that might be encoded in the action space
- Introduces more hyper-parameters, which have to be tuned manually
- **Is there anything we can do about this?**

Policy Gradients

- **All operations performed previously, aimed at finding the optimal policy with regard to the expected discounted reward.**
- Most of those used estimated the value of performing something to do so and defined the assignment of a to s based on rules (greedy, ϵ -greedy).
- To change the policy, the value estimation had to change first.

- The optimization goal was usually related to a measure of performance.
- One example for this, would be the expected discounted reward of π from the starting state s_0 :

$$E(V^\pi(s_0))$$

Policy Gradients

- One example for this, would be the expected discounted reward of π from the starting state s_0 :

$$E(V^\pi(s_0))$$

- This depends on the policy as well as the process itself (transitions, reward function etc.).
- A policy can be parametric, e.g. depend on a set of parameters.
- A non-deterministic policy with parameters ϕ will be denoted

$$\pi_\phi(a|s) = p(a_t = a | s_t = s, \phi_t = \phi)$$

(The probability of choosing a in s given our parameters are set to ϕ at that point in time)

- The known greedy or ϵ -greedy policy could already be formulated in that way.

Policy Formulation

Discrete

The following methods can be used for discrete action spaces as well.

There could be a deterministic mechanism $h(s, a; \phi)$ that outputs a measure of preference for the given action a in state s .

$$\pi_\phi(a|s) = \frac{e^{h(s,a;\phi)}}{\sum_{a' \in A} e^{h(s,a';\phi)}}$$

Is there any advantage of doing something like this if we have our fine-tuned DQN ready to use?

Continuous

In the continuous case, things get more complicated.

Instead of assigning each action a probability, we can provide a probability density function for our continuous action space variables.

Usually, we use an existing probability distribution, a parameter estimator $h(s; \phi)$ and then do something like this:

$$\pi_\phi(a|s) = \frac{1}{h(s; \phi)_\sigma \sqrt{2\pi}} \exp\left(-\frac{(a - h(s; \phi)_\mu)^2}{2h(s; \phi)_\sigma^2}\right)$$

Our parameter estimator outputs the mean (μ) and standard deviation (σ) of a normal dist.!

Policy Gradients

- Given a parametric policy, we can define:

$$J(\phi) = V^{\pi_{\phi}}(s_0)$$

- This has the shape of a loss function. Given some magic optimization logic, we could find well performing policies by finding the parameters ϕ that maximize this value!

- First idea: Reuse previous logic and do SGD or something related to it:

$$\phi_{t+1} \leftarrow \mu \nabla_{\phi} J(\phi_t) + \phi_t$$

- Nice, problem solved. Well, not really.
- The actual problem in this case: **Finding $\nabla_{\phi} J(\phi_t)$.**

Finding $\nabla_{\phi} J(\phi_t)$

The actual problem in this case: Finding $\nabla_{\phi} J(\phi)$.

$$\nabla_{\phi} J(\phi) = \nabla_{\phi} V(s_0)$$

Notice: We consider V generalized to any s for the moment:

$$\begin{aligned} \nabla_{\phi} V(s) &= \nabla_{\phi} \left(\sum_{a \in A} \pi_{\phi}(a | s) Q^{\pi}(s, a) \right) \\ &= \sum_{a \in A} (\nabla_{\phi} \pi_{\phi}(a | s) Q^{\pi}(s, a) + \pi_{\phi}(a | s) \nabla_{\phi} Q^{\pi}(s, a)) \\ &= \sum_{a \in A} \left(\nabla_{\phi} \pi_{\phi}(a | s) Q^{\pi}(s, a) + \pi_{\phi}(a | s) \nabla_{\phi} \left(\sum_{s' \in S, r \in R} p(s', r | s, a) (r + V(s')) \right) \right) \end{aligned}$$

We just found a recursive formulation of this gradient.

Finding $\nabla_{\phi} J(\phi_t)$ (2)

$$= \sum_{a \in A} \left(\nabla_{\phi} \pi_{\phi}(a | s) Q^{\pi}(s, a) + \pi_{\phi}(a | s) \nabla_{\phi} \left(\sum_{s' \in S, r \in R} p(s', r | s, a) (r + V(s')) \right) \right) \nabla_{\phi} V(s)$$

We just found a recursive formulation of this gradient.

With some “smaller” tricks (rolling this out, then noticing that we can replace the “chance of reaching something” part with the probability of going from state s to state x in k steps given we follow π) this leads us to:

$$= \sum_{x \in S} \sum_{k=0}^{\infty} \Pr(s \rightarrow x, k, \pi) \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a | x) Q^{\pi}(x, a)$$

Finding $\nabla_{\phi} J(\phi_t)$ (3)

$$\begin{aligned} & \nabla_{\phi} V(s) \\ = & \sum_{x \in S} \sum_{k=0}^{\infty} \Pr(s \rightarrow x, k, \pi) \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a|x) Q^{\pi}(x, a) \end{aligned}$$

Now lets back to the original problem:

$$\begin{aligned} & \nabla_{\phi} V(s_0) \\ = & \sum_{x \in S} \sum_{k=0}^{\infty} \Pr(s_0 \rightarrow x, k, \pi) \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a|x) Q^{\pi}(x, a) \\ = & \sum_{x \in S} \eta(x) \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a|x) Q^{\pi}(x, a) \end{aligned}$$

The last trick uses the detail that the sum of going to state x from the start state in k steps over all k is equal to the number of times to be in state x during an episode ($\eta(x)$, don't confuse with LR).


Finding $\nabla_{\phi} J(\phi_t)$ (3)

$$\begin{aligned} & \nabla_{\phi} V(s_0) \\ &= \sum_{x \in S} \eta(x) \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a|x) Q^{\pi}(x, a) \end{aligned}$$

The last trick uses the info that the sum of going to state x from the start state in k steps over all k is equal to the number of times to be in state x during an episode ($\eta(x)$).

The relative time spent in an episode is then:

$$\mu(x) = \frac{\eta(x)}{\sum_{s' \in S} \eta(s')}$$



$$= \sum_{s' \in S} \eta(s') \sum_{x \in S} \frac{\eta(x)}{\sum_{s' \in S} \eta(s')} \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a|x) Q^{\pi}(x, a)$$

This is multiplication by one.

Finding $\nabla_{\phi} J(\phi_t)$ (3)

$$\begin{aligned} \mu(x) &= \frac{\eta(x)}{\sum_{s' \in S} \eta(s')} \\ \nabla_{\phi} V(s_0) &= \sum_{s' \in S} \eta(s') \sum_{x \in S} \frac{\eta(x)}{\sum_{s' \in S} \eta(s')} \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a|x) Q^{\pi}(x, a) \\ &= \sum_{s' \in S} \eta(s') \sum_{x \in S} \mu(x) \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a|x) Q^{\pi}(x, a) \\ &\propto \sum_{x \in S} \mu(x) \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a|x) Q^{\pi}(x, a) \end{aligned}$$

Okay now why had we go through this, you could have given us this one on chart 3?

Finding $\nabla_{\phi} J(\phi_t)$ (3)

$$\begin{aligned} \nabla_{\phi} V(s_0) &= \sum_{s' \in S} \eta(s') \sum_{x \in S} \mu(x) \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a|x) Q^{\pi}(x, a) \\ &\propto \sum_{x \in S} \mu(x) \sum_{a \in A} \nabla_{\phi} \pi_{\phi}(a|x) Q^{\pi}(x, a) \end{aligned}$$

Okay now why had we go through this, you could have given us this one on chart 3?

To memorize that the recursiveness in the values allows us to compute something proportional to the actual gradient by only using 3 pieces of information:

- (i) The relative occurrence of all states, (ii) the gradient on the probability of an action and (iii) the Q-value of this action.

We can cancel out all transition probabilities at some point.

The Road to REINFORCE

Relative occurrence

If we compute the inner element of the gradient every time we observe state s the expected value of the resulting gradients will be the same as the sum just shown. (Everything else can be down-scaled by adjusting the learning rate).

Gradient on π

Can be simply computed with our backpropagation tools at hand. This requires the policy to be differentiable with regard to all parameters.

Q-value

We have already seen a simple method for the episodic case:
Follow the policy, wait until we reach the end of the episode and then simply compute it for the states and actions observed during that episode.

REINFORCE

Ha. You thought we're done with summing up sums of sums yet?

$$\begin{aligned}
 \nabla_{\phi} J(\phi) &\propto \sum_{s \in \mathcal{S}} \mu(s) \sum_{a \in \mathcal{A}} \nabla_{\phi} \pi_{\phi}(a|s) Q^{\pi}(s, a) \\
 &= E \left(\sum_{a \in \mathcal{A}} \frac{\pi_{\phi}(a|s)}{\pi_{\phi}(a|s)} \nabla_{\phi} \pi_{\phi}(a|s) Q^{\pi}(s, a) \right) \\
 &= E \left(\frac{\nabla_{\phi} \pi_{\phi}(a_t|s_t)}{\pi_{\phi}(a_t|s_t)} Q^{\pi}(s_t, a_t) \right) \\
 &= E \left(\frac{\nabla_{\phi} \pi_{\phi}(a_t|s_t)}{\pi_{\phi}(a_t|s_t)} G_t \right)
 \end{aligned}$$

In the last step, we replaced s and a with realizations of it. We are allowed to do so, because that's covered by our expected value.

REINFORCE

$$\nabla_{\phi} J(\phi) \propto E \left(\frac{\nabla_{\phi} \pi_{\phi}(a_t | s_t)}{\pi_{\phi}(a_t | s_t)} G_t \right)$$

Inserting this in our gradient ascent update, we get the REINFORCE update mechanism:

$$\phi_{t+1} = \phi_t + \eta_t \frac{\nabla_{\phi} \pi_{\phi_t}(a_t | s_t)}{\pi_{\phi_t}(a_t | s_t)} G_t$$

η_t = Learning rate

Note, the computation of G_t can only be performed in the episodic case with a guaranteed terminal state.

Schedule

- 3.5.: Problems with Q-Learning
 - 10.5.: Deep Q-Networks
 - 17.5.: Eligibility Traces
 -
 - 31.5.: Project Assignments
 - 7.6.: Project Work/Support
 - 14.6.: Project Work/Support
 - 21.6.: Project Work/Support
 - 28.6.: Project Work/Support
 - 5.7.: Project Work/Support
 - 12.7.: Final Presentations
 - 31.8.: Final Documentation
- 6.5.: Artificial Neural Networks
 - 20.5.: Policy Gradients (1)
 - 27.5.: Policy Gradients (2)**
 - 3.6.: Project Assignments
 - 10.6.: Project Work/Support
 - 17.6.: Project Work/Support
 - 24.6.: Project Work/Support
 - 31.6.: Project Work/Support
 - 8.7.: Project Work/Support
 - 15.7.: Final Presentations