Policy Gradients for Reinforcement Learning

Keith L. Downing

The Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
keithd@idi.ntnu.no

October 29, 2018
A comparison between Error Gradients and Policy Gradients in the context of reinforcement learning is shown. The goal is to minimize error by changing weights in the Error Gradients approach, while in the Policy Gradients approach, the goal is to maximize performance by changing weights.

- **Error Gradients**
  - Targets from MCTS
  - Action Probabilities
  - Act Probs
  - Error
  - Goal: minimize error by changing weights

- **Policy Gradients**
  - Action Probabilities
  - Actor
  - Performance
  - Goal: maximize performance measure by changing weights

Keith L. Downing - Policy Gradients for Reinforcement Learning
Why Bother?

- Sometimes it’s easier to train the actor directly than to train a value function. The policy may be a simpler function to approximate than a state-value or action-value function.
- MCTS (for training actors) is computationally demanding.
- Easily modified to handle continuous action spaces.
- User bias is often easier to achieve in a policy network than a value network...for domains where you want to bias behavior.

Design Problems to Solve

- What is the performance measure, $J(\theta)$? $\forall i : w_i \in \theta$
- How to compute the policy gradient $\equiv \nabla_\theta J(\theta)$
Policy Gradient Theorem

- Performance measure = value of start state:
  \[ J(\theta) \equiv V_\pi(s_0) \]
  = Total expected reward from start state to final state.
- The policy gradient theorem shows that:
  \[ \nabla_\theta J(\theta) \propto \sum_s \mu(s) \sum_a Q_\pi(s, a) \nabla_\theta \Pi(a \mid s, \theta) \]

where:
- \( \mu(s) = \) frequency of state \( s \) visits under policy \( \Pi \)
- \( Q_\pi(s, a) = \) action-value function when following policy \( \Pi \)
- \( \nabla_\theta \Pi(a \mid s, \theta) = \) deriv of the policy function w.r.t. its parameters \( (\theta) \), evaluated at \( (s,a) \).
Intuitive view of the Policy Gradient Theorem

$$\nabla_{\theta} J(\theta) \propto \sum_{s} \mu(s) \sum_{a} Q_\pi(s,a) \nabla_{\theta} \Pi(a | s, \theta)$$

- The effect of the function parameters (e.g. NN weights) on performance is proportional to
- their effect on the output action probabilities
- weighted by the values of those actions, and
- averaged over all state-action combinations, weighted by the probabilities of being in each state.

An influential parameter, $w \in \theta$, is one that, for input state $s$, creates high (magnitude) gradients for the output corresponding to action $a$ when $Q(s,a)$ also has a high magnitude and state $s$ occurs frequently.
Using the Policy Gradient Theorem (PGT)

Stochastic Gradient Ascent (goal: performance $\uparrow$)

$$\forall w \in \theta : \Delta w = \alpha \nabla_w J(\theta)$$

where $\alpha =$ learning rate.

- Need to insure that our samples (of states and actions) during RL search yield expected values for the gradients that are proportional to those given by PGT.
- As long as sampling is governed by the policy, $\Pi$, then since PGT is also based on $\Pi$, our gradient estimates (based on samples) should be true to the gradients of PGT.
- Thus, we should be able to define quantities that can be calculated on each timestep, based on the most recent $(s,a)$, that give gradients proportional to those of PGT $\rightarrow$ we can gradually optimize performance, $J(\theta)$. 

Keith L. Downing
Policy Gradients for Reinforcement Learning
Sources of Policy-Gradient Data

- $\Pi(a \mid s, \theta) = $ outputs of the actor network
- $\nabla_{\theta} \Pi(a \mid s, \theta) - $ gradients of outputs w.r.t. weights
- $Q(s,a) - $ action values approximated by another network and/or using the latest return, $G_t$
Assuming policy $\Pi$ is used for sampling during RL search, the distribution of state visits should match the $\mu(s)$ from PGT. Thus, we can rewrite PGT in terms of an expected value across our samples:

$$\nabla_\theta J(\theta) \propto \sum_s \mu(s) \sum_a Q_\pi(s, a) \nabla_\theta \Pi(a \mid s, \theta)$$

$$= E_\pi [\sum_a Q_\pi(S_t, a) \nabla_\theta \Pi(a \mid S_t, \theta)]$$

where $s$ is replaced with $S_t$, a variable representing a sampled state.

The $\sum_a$ is a problem, since we won’t consider all $a$’s during sampling. We will sample $a$’s based on the policy: $\Pi(a \mid S_t, \theta)$. We can introduce that term by multiplying and dividing by it:

$$= E_\pi \left[ \sum_a \Pi(a \mid S_t, \theta) Q_\pi(S_t, a) \frac{\nabla_\theta \Pi(a \mid S_t, \theta)}{\Pi(a \mid S_t, \theta)} \right]$$
Since our action sampling will be true to $\Pi$, our distribution of $(s,a)$ visits will match those of $\Pi(a | S_t, \theta)$.

Hence, we can remove the $\sum_a \Pi(a | S_t, \theta)$, and replace $a$ with the sampled action, $A_t$:

$$= E_\pi \left[ Q_\pi(S_t, A_t) \frac{\nabla_\theta \Pi(A_t | S_t, \theta)}{\Pi(A_t | S_t, \theta)} \right]$$

Since, from calculus, $\frac{\nabla_\theta f(\theta)}{f(\theta)} = \nabla_\theta ln(f(\theta))$, we simplify to:

$$= E_\pi [Q_\pi(S_t, A_t) \times \nabla_\theta ln(\Pi(A_t | S_t, \theta))]$$

Finally, during sampling: $E[G_t | S_t, A_t] = Q_\pi(S_t, A_t)$, where $G_t$ = our normal return:

$$\nabla_\theta J(\theta) \propto E_\pi [G_t \times \nabla_\theta ln(\Pi(A_t | S_t, \theta))]$$
So our final sampling-based estimate of performance, based on PGT, is:

$$\nabla_\theta J(\theta) \propto E_\pi [G_t \nabla_\theta \ln(\Pi(A_t | S_t, \theta))]$$

And our parameter (e.g. weight) updates at each step of RL are then:

$$\Delta \theta = \alpha \nabla_\theta J(\theta) = \alpha G_t \nabla_\theta \ln(\Pi(A_t | S_t, \theta))$$

Intuitively...

$$\Delta \theta = \alpha G_t \frac{\nabla_\theta \Pi(A_t | S_t, \theta)}{\Pi(A_t | S_t, \theta)}$$

The updates of each parameter $w \in \theta$ are:

- directly proportional to the return, and
- directly proportional to the effects of $w$ upon the output probability corresponding to the action actually taken, $A_t$, but
- inversely proportional to that output probability
The REINFORCE Algorithm

An Episodic, Monte-Carlo Policy-Gradient RL method.

- Do one episode: $S_0, A_0, R_1, S_1, A_1..., R_T, S_T$ using $\Pi_\theta$.
- For each step $t \in 0, 1, ..., T - 1$:
  - $G_t$ is the return from step $t$ (to $T$)
  - $\theta \leftarrow \theta + \alpha G_t \nabla_\theta \ln(\Pi(A_t | S_t, \theta))$
Policy Gradient Theorem with a Baseline

- Policy Gradient Theorem (PGT) can be generalized to include an arbitrary baseline function of s, b(s), as long as b(s) is independent of the action a.
- This does not affect the average of $\triangle \theta$ but normally reduces its variance $\rightarrow$ faster convergence.

PGT with baseline

$$\nabla_\theta J(\theta) \propto \sum_s \mu(s) \sum_a (Q_\pi(s,a) - b(s)) \nabla_\theta \Pi(a \mid s, \theta)$$

- This has no effect on $\nabla_\theta J(\theta)$, since:
  $$\sum_a b(s) \nabla_\theta \Pi(a \mid s, \theta) = b(s) \nabla_\theta \sum_a \Pi(a \mid s, \theta) = b(s) \nabla_\theta 1 = 0$$

- A natural choice for b(s) is V(s). Then Q(s,a) - V(s) indicates how much better (or worse) a is than the other possible actions from s.
Episodic Monte-Carlo with policy $\Pi(a \mid s, \theta)$ and state-value function, $V(s \mid \Phi)$. Both typically NNs.

Parameters $\theta$ and $\Phi$ learned, using rates $\alpha_\theta$ and $\alpha_\Phi$, respectively.

Do one episode: $S_0, A_0, R_1, S_1, A_1, \ldots, R_T, S_T$ using $\Pi_\theta$.

For each step $t \in 0, 1, \ldots, T - 1$:

$G_t$ is the return from step $t$ (to $T$)

$\delta = G_t - V(S_t \mid \Phi)$

$\Phi \leftarrow \Phi + \alpha_\Phi \delta \nabla_\Phi V(S_t, \Phi)$

$\theta \leftarrow \theta + \alpha_\theta \delta \nabla_\theta \ln(\Pi(A_t \mid S_t, \theta))$

Note that $V$ is not used for bootstrapping, just as a baseline.

MC (episodic) nature makes this slower to converge.

Use TD actor-critic model to speed it up, and open up for continuous (non-episodic) tasks.
Now we update after every move in an episode.

Replace $G_t$ with $G_{t:t+1} = R_{t+1} + \gamma V(S_{t+1} \mid \Phi)$

$\delta$ is now our TD error, and since we already showed that the baseline $V(s)$ does not violate PGT, we are still in good shape w.r.t. the underlying theory.

Repeat Forever:

- $S \leftarrow$ episode start state
- $D \leftarrow 1$
- While $S$ is not a final state:
  - Choose $A$ based on policy $\Pi(a \mid S, \theta)$
  - Do action $A$ from state $S \mapsto$ state $S'$ and reward $R$.
  - $\delta \leftarrow R + \gamma V(S' \mid \Phi) - V(S \mid \Phi)$
  - $\Phi \leftarrow \Phi + \alpha_\Phi D \delta \nabla_\Phi V(S \mid \Phi)$
  - $\theta \leftarrow \theta + \alpha_\theta D \delta \nabla_\theta \ln(\Pi(A \mid S, \theta))$
  - $D \leftarrow \gamma D$
  - $S \leftarrow S'$
Now we can generalize to TD(\(\lambda\)), thus handling multi-step backups. Use a different backup factor for \(\theta\) and \(\Phi\): \(\lambda_\theta\) and \(\lambda_\Phi\).
Use eligibility traces (e) for each parameter set: \(e_\theta\) and \(e_\Phi\).

Repeat Forever:

- \(S \leftarrow\) episode start state
- \(D \leftarrow 1\); \(e_\theta \leftarrow 0\); \(e_\Phi \leftarrow 0\)
- While \(S\) is not a final state:
  - Choose \(A\) based on policy \(\Pi(a \mid S, \theta)\)
  - Do action \(A\) in \(S \rightarrow R, S'\)
  - \(\delta \leftarrow R + \gamma V(S' \mid \Phi) - V(S \mid \Phi)\)
  - \(e_\theta \leftarrow \gamma \lambda_\theta e_\theta + D \nabla_\theta \ln(\Pi(A \mid S, \theta))\)
  - \(e_\Phi \leftarrow \gamma \lambda_\Phi e_\Phi + D \nabla_\Phi V(S \mid \Phi)\)
  - \(\theta \leftarrow \theta + \alpha_\theta \delta e_\theta\)
  - \(\Phi \leftarrow \Phi + \alpha_\Phi \delta e_\Phi\)
  - \(D \leftarrow \gamma D\)
  - \(S \leftarrow S'\)
The Policy Gradient Theorem can also be proven for continuing cases, so we can extend REINFORCE as well.

- \( S \leftarrow \) a global start state.
- \( D \leftarrow 1 \); \( e_\theta \leftarrow 0 \); \( e_\Phi \leftarrow 0 \); \( R^* \leftarrow 0 \).

Repeat Forever:

- Choose \( A \) based on policy \( \Pi(a \mid S, \theta) \)
- Do action \( A \) in \( S \mapsto R, S' \)
- \( R^* \leftarrow f(R^*, R) \)
- \( \delta \leftarrow (R - R^*) + \gamma V(S' \mid \Phi) - V(S \mid \Phi) \)
- \( e_\theta \leftarrow \gamma \lambda_\theta e_\theta + D \nabla_\theta \ln(\Pi(A \mid S, \theta)) \)
- \( e_\Phi \leftarrow \gamma \lambda_\Phi e_\Phi + D \nabla_\Phi V(S \mid \Phi) \)
- \( \theta \leftarrow \theta + \alpha_\theta \delta e_\theta \)
- \( \Phi \leftarrow \Phi + \alpha_\Phi \delta e_\Phi \)
- \( D \leftarrow \gamma D \)
- \( S \leftarrow S' \)

\( R^* \propto \) running average of \( R \)