Policy Gradient Algorithms

• Why?
  – Value functions can be very complex for large problems, while policies have a simpler form.
  – Convergence of learning algorithms not guaranteed for approximate value functions whereas policy gradient methods are well-behaved with function approximation.
  – Value function methods run into a lot of problems in partially observable environments. Policy gradient methods are “better" behaved even in this scenario.
Policy Gradient Methods

• Policy depends on some parameters $\Theta$
  – Action preferences
  – Mean and variance
  – Weights of a neural network

• Modify policy parameters directly instead of estimating the action values

• Maximize:

$$\eta(\Theta) = E(r)$$
$$= \sum_a Q^*(a) \cdot \pi(\Theta, a)$$

$$\Theta \leftarrow \Theta + \alpha \cdot \nabla \eta(\Theta)$$
 Likelihood Ratio Method  

• Computing gradient of performance w.r.t. parameters:

\[
\eta(\Theta) = E(r) \\
= \sum_a Q^*(a)\pi(a; \Theta) \\
\nabla \eta(\Theta) = \sum_a Q^*(a)\nabla \pi(a; \Theta) \\
= \sum_a Q^*(a)\frac{\nabla \pi(a; \Theta)}{\pi(a; \Theta)}\pi(a; \Theta)
\]

• Estimate the gradient from N samples:

\[
\hat{\nabla}(\Theta) = \frac{1}{N} \sum_{i=1}^{N} r_i \cdot \frac{\nabla \pi(a_i; \Theta)}{\pi(a_i; \Theta)} \underbrace{\text{LikelihoodRatio}}
\]
REINFORCE (Williams ’92)

• Incremental version:

\[
\Delta \theta_t = \alpha_t \cdot r_t \cdot \frac{\nabla \pi(\Theta, a_t)}{\pi(\Theta, a_t)}
\]

\[
\Delta \theta_t = \alpha_t \cdot r_t \cdot \frac{\partial \ln \pi(\Theta, a_t)}{\partial \theta}
\]

- Reinforcement
- Baseline
- Characteristic
- Eligibility
Special case – Generalized $L_{R-I}$

- Consider binary bandit problems with arbitrary rewards

$$
\pi(\theta, a) = \begin{cases} 
\theta & \text{if } a = 1 \\
1 - \theta & \text{if } a = 0 
\end{cases} \\
\frac{\partial \ln \pi}{\partial \theta} = \frac{a - \theta}{\theta(1 - \theta)}
$$

$b = 0$ and $\alpha = \rho \cdot \theta(1 - \theta)$

$$
\Delta \theta = \rho \cdot r \cdot (a - \theta)
$$
Reinforcement Comparison

• Set baseline to average of observed rewards

\[ b_t = \bar{r}_t = \bar{r}_{t-1} + \beta \cdot (r_t - \bar{r}_{t-1}) \]

• Softmax action selection

\[ \Delta \theta_i = \alpha \cdot (r - \bar{r}) (1 - \pi(\Theta, a_i)) \]
Reinforcement Comparison contd.

\[ \pi(\Theta, a_i) = \frac{e^{\theta_i}}{\sum_{j=1}^{n} e^{\theta_j}} \]

\[ \frac{\partial \ln \pi(\Theta, a_i)}{\partial \theta_i} = \frac{\partial}{\partial \theta_i} \ln \left( \frac{e^{\theta_i}}{\sum_{j=1}^{n} e^{\theta_j}} \right) \]

\[ = \frac{\partial}{\partial \theta_i} \left( \theta_i - \ln \left( \sum_{j=1}^{n} e^{\theta_j} \right) \right) \]

\[ = 1 - \frac{e^{\theta_i}}{\sum_{j=1}^{n} e^{\theta_j}} \]

\[ = 1 - \pi(\Theta, a_i) \]
Continuous Actions

• Use a Gaussian distribution to select actions

\[ \pi(a, \mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(a-\mu)^2}{2\sigma^2}} \]

• For suitable choice of parameters:

\[ \Delta \mu = \alpha \cdot (r - \bar{r})(a - \mu) \]

\[ \Delta \sigma = (\alpha / \sigma) \cdot (r - \bar{r})((a - \mu)^2 - \sigma^2) \]
MC Policy Gradient

- Samples are entire trajectories $s_0, a_0, r_1, s_1, a_1, \ldots, s_T$
- Evaluation criterion is the return along the path, instead of immediate rewards
- The gradient estimation equation becomes:

$$\hat{\nabla}(\Theta) = \frac{1}{N} \sum_{i=1}^{N} R_i(s_0) \cdot \frac{\nabla p_i(s_0; \Theta)}{p_i(s_0; \Theta)}$$

where, $R_i(s_0)$ is the return starting from state $s_0$ and $p_i(s_0; \Theta)$ is the probability of $i^{th}$ trajectory, starting from $s_0$ and using policy given by $\Theta$. 
MC Policy Gradient contd.

• The “likelihood ratio" in this case evaluates to:

\[
\frac{\nabla p_i(s_0; \Theta)}{p_i(s_0; \Theta)} = \frac{T-1}{\sum_{j=0}^{T-1} \frac{\nabla \pi(s_j, a_j; \Theta)}{\pi(s_j, a_j; \Theta)}} \tag{1}
\]

• Estimate depends on starting state \(s_0\). One way to address this problem is to assume a fixed initial state.

• More common assumption is to use the average reward formulation.
MC Policy Gradient contd.

• Recall:
  – Maximize average reward per time step:

    \[
    \rho^\pi(s) = \lim_{N \to \infty} \frac{1}{N} E \left( \sum_{t=0}^{N-1} r_t \mid s_0 = s \right)
    \]

  – Unichain assumption: One set of “recurrent" class of states
  – \( \rho^\pi \) is then state independent
  – Recurrent class: Starting from any state in the class, the probability of visiting all the states in the class is 1.
MC Policy Gradient contd.

- Assumption 1: For every policy under consideration, the Unichain assumption is satisfied, with the same set of recurrent states.
- Pick one recurrent state $i^*$. Trajectories are defined as starting and ending at this recurrent state.
- Assumption 2: Bounded rewards.
Incremental Update

• We can incrementally compute the summation in Equation 1, over one trajectory as follows:

\[
\begin{align*}
Z_{t+1} &= Z_t + \frac{\nabla \pi(s_t, a_t; \Theta)}{\pi(s_t, a_t; \Theta)} \\
R_{t+1} &= R_t + \frac{1}{t+1} [r_t - R_t]
\end{align*}
\]

• \( Z_T \) is known as an eligibility trace. Recall the characteristic eligibility term from REINFORCE:

\[
\frac{\partial \ln \pi(a_t; \Theta)}{\partial \Theta}
\]

• \( Z_T \) keeps track of this eligibility over time, hence is called a trace.
Simple MC Policy Gradient Algorithm

**Algorithm 1 Simple MC Policy Gradient Algorithm**

1: Set \( j = 0, R_0 = 0, z_0 = 0, \Delta_0 = 0 \)
2: for each episode do
3:   for each transition \( s_t, a_t, r_t, s_{t+1} \) do
4:     \[ z_{t+1} = z_t + \frac{\nabla \pi(s_t, a_t; \Theta)}{\pi(s_t, a_t; \Theta)} \]
5:     \[ R_{t+1} = R_t + \frac{1}{t+1} [r_t - R_t] \]
6:   end for
7:   \[ \Delta_{j+1} = \Delta_j + R_T z_T \]
8:   \[ j = j + 1 \]
9: end for
10: Return \( \Delta_N / N \), where \( N \) is the number of episodes

Adjust \( \Theta \) using a simple stochastic gradient ascent rule:

\[
\Theta \leftarrow \Theta + \alpha \frac{\Delta_N}{N}
\]

where \( \alpha \) is a positive step size parameter.
The algorithm computes an unbiased estimate of the gradient.

Can be very slow due to high variance in the estimates.

Variance is related to the “recurrence time” or the episode length.

For problems with large state spaces, the variance becomes unacceptably high.
Variance reduction techniques

• Truncate summation (eligibility traces)
• Decay eligibility traces. In this case, the decay rate controls the bias-variance trade off.
• Actor-Critic methods. These methods use value function estimates to reduce variance.
• Employ a set of recurrent states to define episodes, instead of just one $i^*$. 