CME 241 Notes

December 2019

1 Lecture 1

intro and housekeeping

2 Lecture 2

2.1 Markov Process

2.1.1 Markov Property

A given state S, is markovian iff $\mathbb{P}(S_{t+1}|S_t) = \mathbb{P}(S_{t+1}|S_1, S_2, ...S_t)$

2.1.2 State Transitions

$$\mathcal{P}_{ss'} = \mathbb{P}(S_{t+1} = s'|S_t = s)$$

$$\mathcal{P} = \begin{bmatrix} \mathcal{P}_{11} & \mathcal{P}_{12} & \dots & \mathcal{P}_{1n} \\ \vdots & \ddots & & \vdots \\ \mathcal{P}_{n1} & \mathcal{P}_{n2} & \dots & \mathcal{P}_{nn} \end{bmatrix}$$

A Markov Process (or Markov Chain) is a tuple $M = (\mathcal{S}, \mathcal{P})$

1. $\mathcal{S}$ is a finite set of states
2. $\mathcal{P}$ is a state transition probability matrix $\mathcal{P}_{ss'} = \mathbb{P}(S_{t+1} = s'|S_t = s)$.

2.2 Markov Reward Process

A Markov Reward Process is a tuple $M = (\mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma)$

1. $\mathcal{S}$ is a finite set of states
2. $\mathcal{P}$ is a state transition probability matrix $\mathcal{P}_{ss'} = \mathbb{P}(S_{t+1} = s'|S_t = s)$.
3. $\mathcal{R}$ is a reward function, $\mathcal{R}_s = \mathbb{E}[R_{t+1}|S_t = s]$
4. $\gamma \in [0, 1]$ is the discount factor

2.2.1 Return

The return at a timestamp t $G_t = R_{t+1} + \gamma T_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$
A closely related concept is the value function for a given state $v(s) = \mathbb{E}[G_t|S_t = s]$

2.2.2 Bellman Equations for MRPs

$$\begin{aligned}
v(s) &= \mathbb{E}[G_t|S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ...|S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma (R_{t+2} + \gamma R_{t+3} + ...)|S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma G_{t+1}|S_t = s] \\
&= \mathbb{E}[R_{t+1} + \gamma v(S_{t+1})|S_t = s]
\end{aligned}$$

$$v(s) = \mathcal{R}s + \gamma \sum_{s_0 \in S} \mathcal{P}_{ss_0} v(s_0)$$

or equivalently using matrices

$$v = \mathcal{R} + \gamma \mathcal{P} v$$

$$v = (I - \gamma \mathcal{P})^{-1} \mathcal{R}$$

2.3 Markov Decision Process

A Markov Decision Process is a tuple $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$

1. $\mathcal{S}$ is a finite set of states
2. $\mathcal{A}$ is a finite set of actions
3. $\mathcal{P}$ is a state transition probability matrix $\mathcal{P}^a_{ss'} = \mathbb{P}(S_{t+1} = s'|S_t = s, A_t = a)$.
4. $\mathcal{R}$ is a reward function, $\mathcal{R}^a_s = \mathbb{E}[R_{t+1}|S_t = s, A_t = a]$
5. $\gamma \in [0, 1]$ is the discount factor

2.3.1 Policies

A policy π is a distribution over actions given states

$$\pi(a|s) = \mathbb{P}[A_t = a|S_t = s]$$

Note that MDP policies depend only on the current state (again history independent / memory-less).

We also observe that an MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ and a policy π together define a Markov Process $(\mathcal{S}, \mathcal{P}^\pi$, and a Markov Reward Process $(\mathcal{S}, \mathcal{P}^\pi, \mathcal{R}^\pi, \gamma$ where

$$\mathcal{P}^\pi = \sum_{a \in \mathscr{A}} \pi(a|s), \mathcal{P}^a_{ss'}$$

$$\mathcal{R}^\pi = \sum_{a \in \mathscr{A}} \pi(a|s), \mathcal{R}^a_s$$

2.3.2 Value Function

The 'state value function' $v_\pi(s)$ is the expected return starting from state s and following policy π

$$v_\pi(s) = \mathbb{E}[G_t|S_t = s]$$

The 'action value function' $q_\pi(s, a)$ is the expected return starting from state s, taking action a then following policy p.

$$q_\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a]$$

2.3.3 Bellman Equations for MDPs

We start by decomposing the equations for the state and action value functions

$$v_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s]$$
$$q_\pi(s,a) = \mathbb{E}_\pi[R_{t+1} + \gamma q_\pi(S_{t+1}, A_{t+1})|S_t = s, A_t = a]$$

next we note that

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a)$$

and

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s')$$

Substitution one in the other both ways yields

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s)(\mathcal{R} + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(s'))$$
$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a (\sum_{a' \in \mathcal{A}} \pi(a'|s') q_\pi(s', a'))$$

We can once again express these functions concisely in matrix form:

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi$$

2.3.4 Optimal things

We define the optimal state value function $v_*(s)$ as the maximum value function over all possible policies.

$$v_*(s) = \max_\pi v_\pi(s)$$

Similarly, we define the optimal action-value function $q_*(s,a)$ is the maximum action-value function over all policies

$$q_*(s,a) = \max_\pi q_\pi(s,a)$$

The optimal policy is generally what we are searching for in an MDP, we consider an MDP 'solved' when we know the optimal value function.
We define a partial ordering on all policies π s.t.

$$\pi_1 \geq \pi_2 \ if \ v_{\pi_1}(s) \geq v_{\pi_2}(s) \forall s \in S$$

Theorem: For any markov decision process

- There exists an optimal policy π_* s.t. $\pi_* \geq \pi$ for any valid policy π
- all optimal policies achieve the same value function $v_{\pi_{*1}}(s) = v_{\pi_{*2}}(s) \forall s \in S$
- all optimal policies achieve the same action-value function $q_{\pi_{*1}}(s|a) = q_{\pi_{*2}}(s|a) \forall s \in S$

Given an optimal action value function or value function, we can easily find an optimal policy:

$$\pi_*(a|s) = \begin{cases} 1 & a = \underset{a \in A}{argmax} \ q_*(s,a) \\ 0 & otherwise \end{cases}$$

Lecture 3, Planning By Dynamic Programming

2.4 Iterative Policy Evaluation

Goal: evaluate a given plicy π for a MDP $M =< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$. We iterative apply the Bellman expectation backup $v_1 \rightarrow v_2 \rightarrow ... \rightarrow v_pi$.

$$v_{k+1}(s) = \sum_{a \in \mathcal{A}} \pi(a|s)(\mathcal{R}_s^a + \gamma \sum_{s \ \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s'))$$

$$\boldsymbol{v}^{k+1} = \boldsymbol{\mathcal{R}}^{\pi} + \gamma \boldsymbol{\mathcal{P}}^{\pi} \boldsymbol{v}^{v}$$

The convergence of this iterative algorithm is based on a fixed point argument. Nameley that the point $\boldsymbol{v}_{pi}$ is a fixed point up this update equation, (which should be clear from the bellman equations), and moreover that this update function is contraction mapping. Therefore, convergence to $\boldsymbol{v}_{\pi}$ is guaranteed for *any* starting point v_0.

2.5 Iterative Policy Improvement

Now that we have a way to evaluate a given policy, we would like to be able to improve it as well. we do this in a two step process
Given a policy π

1. evaluate the policy π_k via iterative policy evaluation
   $v_1 \rightarrow v_2 \rightarrow ... \rightarrow v_pi$.
2. improve the policy by acting greedily with respect to v_{π}

   $$\pi_{k+1}(a|s) = arg \max_{a \in \mathcal{A}} q_{\pi}(s, a)$$

3. iterate

Again, this procedure will always converge to the optimal policy π_*, with the proof for this again being based on fixed point / contraction mapping arguments.

2.6 Value Iteration

It turns out we can actually shortcut the policy iteration portion of our iterative improvement algorithm. This should intuitively make sense as given any value function, the optimal policy is determined using a greedy strategy.

$$v_{k+1}(s) = \max_{a \in \mathcal{A}} (\mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_k(s'))$$

$$\boldsymbol{v}_{k+1}(s) = \max_{a \in \mathcal{A}} \boldsymbol{\mathcal{R}}^{\boldsymbol{a}} + \gamma \boldsymbol{\mathcal{P}}^{\boldsymbol{a}} \boldsymbol{v}_k$$

3 Why This Works: Contraction Mappings and Fixed Points

Define the *Bellman Expectation Backup* operator T^{π} as :

$$T^{\pi}(v) = \mathcal{R} + \gamma \mathcal{P}^{\pi} v$$

This function is a γ contraction i.e.

$$\begin{aligned}||T^\pi(u) - T^\pi(u)||_\infty &= ||\mathcal{R} + \gamma\mathcal{P}^\pi u - (\mathcal{R} + \gamma\mathcal{P}^\pi v)||_\infty \\ &= ||\gamma\mathcal{P}^\pi(u-v)||_\infty \\ &\leq \gamma||\mathcal{P}^\pi||_\infty||(u-v)||_\infty \\ &\leq \gamma||(u-v)||_\infty\end{aligned}$$

Similarly, if we define

$$T^*(v) = \max_{a\in\mathcal{A}} \mathcal{R}^a + \gamma\mathcal{P}^a v$$

we have

$$\begin{aligned}||T^*(u) - T^*(u)||_\infty &= ||\max_{a\in\mathcal{A}}(\mathcal{R}^a + \gamma\mathcal{P}^a v) - (\max_{a\in\mathcal{A}}(\mathcal{R}^a + \gamma\mathcal{P}^a v))||_\infty \\ &\leq \gamma||(u-v)||_\infty\end{aligned}$$

Thus convergence is shown.

4 Risk Aversion through Utility Theory

4.1 Definitions

- Utility of consumption $U(x)$
  - x represents the uncertain outcome being consumed
  - $U(\cdot)$ is a concave function, therefore $\mathbb{E}[U(x)] \leq U(\mathbb{E}[x])$
- **Certainty-Equivalent Value** $x_{CE} = U^{-1}(\mathbb{E}(U(x))$
- **Absolute Risk-Premium** $\pi_A = \mathbb{E}[x] - x_{CE}$
- **Relative Risk-Premium** $\pi_R = \frac{\pi_A}{\mathbb{E}[x]} = \frac{\mathbb{E}[x]-x_{CE}}{\mathbb{E}[x]} = 1 - \frac{x_{CE}}{\mathbb{E}[x]}$

4.2 Calculating Risk-Premium

From here on we will call $\mathbb{E}[x] = \bar{x}$ and $Var(x) = \sigma_x^2$
First we take the second order taylor expansion of $U(x)$ around $\bar{x}$:

$$U(x) \approx U(\bar{x}) + U'(\bar{x})(x-\bar{x}) + \frac{1}{2}U''(\bar{x})(x-\bar{x})^2$$

next we take the first order taylor expansion of $U(x_{CE})$ around $\bar{x}$:

$$U(x_{CE}) \approx U(\bar{x}) + U'(\bar{x})(x_{CE} - \bar{x})$$

Taking the expectation of $U(x)$ we get

$$\mathbb{E}[U(x)] \approx U(\bar{x}) + \frac{1}{2}U''(\bar{x})\sigma_x^2$$

Noting that $\mathbb{E}[U(x)] = U(x_{CE})$ we have

$$U'(\bar{x}(x_{CE} - x) \approx \frac{1}{2}U''(\bar{x}) \cdot \sigma_x^2$$

From this, we can get expressions for the Absolute and Relative Risk Aversion

$$\pi_a = \bar{x} - x_{CE} \approx \frac{1}{2} \cdot \frac{U''(\bar{x})}{U'(\bar{x}} \cdot \sigma_x^2$$

$$\pi_r = \frac{\pi_a}{\bar{x}} \approx \frac{1}{2} \cdot \frac{U''(\bar{x}) \cdot \bar{x}}{U'(\bar{x}} \cdot \frac{\sigma_x^2}{\bar{x}^2} = 1\frac{1}{2} \cdot \frac{U''(\bar{x}) \cdot \bar{x}}{U'(\bar{x}} \cdot \sigma_{\frac{x}{\bar{x}}}^2$$

Define:

1. **Absolute Risk-Aversion** $A(\bar{x}) = -\frac{U''(\bar{x})}{U'(\bar{x})}$
2. **Relative Risk-Aversion** $R(\bar{x}) = -\frac{U''(\bar{x}) \cdot \bar{x}}{U'(\bar{x})}$

4.3 CARA and Applications

4.3.1 CARA definition

Allow $U(x) = \frac{-e^{-ax}}{a}$ for $a \neq 0$

$$A(x) = \frac{-U''(x)}{U'(x)} = a$$

a is called the coefficient of Constant Absolute Risk Aversion (CARA) if we allow the randome outcome $x \sim \mathcal{N}(mu, \sigma^2)$, then

$$\mathbb{E}[U(x)] = \begin{cases} \frac{-e^{-a\mu + \frac{a^2\sigma^2}{2}}}{a} & a \neq 0 \\ \mu & a = 0 \end{cases}$$

$$x_{CE} = \mu - \frac{a\sigma^2}{2}$$

Therefore, $\pi_a = \frac{a\sigma^2}{2}$

4.3.2 CARA applied to portfolio allocation

consider the following scenario

- We are given \$1 to invest and hold for a horizon of 1 year
- Investment choices are 1 risky asset and 1 riskless asset
  1. riskless asset annual return $\sim r$
  2. risky asset annual return $\sim \mathcal{N}(\mu, \sigma^2)$
- we want to determine the optimal (unconstrained) π to allocate to risky asset $(1 - \pi)$ is allocated to riskless asset to maximize utility of wealth in 1 year

We note that portfolio wealth $\mathcal{N}(1 + r + \pi(\mu - r), \pi^2\sigma^2$ so we optimize this, i.e. differentiate and set equal to zero yielding

$$\pi^* = \frac{\mu - r}{a\sigma^2}$$

4.4 CRRA and Applications

4.4.1 CRRA definition

Allow $U(x) = \frac{x^{1-\gamma}}{1-\gamma}$ for $\gamma \neq 1$

Relative Risk Aversion: $R(x) = \gamma$, γ is the *Coefficient of Constant Relative Risk Aversion*, note for $\gamma = 1,\ U(x) = log(x)$

If the randome outcome x is lognormal, $log(x) \sim \mathcal{N}(\mu, \sigma^2)$

$$\mathbb{E}[U(x)] = \begin{cases} \frac{e^{\mu(1-\gamma)} + \frac{\sigma^2}{2}(1-\gamma)^2}{1-\gamma} & gamma \neq 1 \\ \mu & \gamma = 1 \end{cases}$$

Relative Risk Premium, $\pi_R = s - \frac{x_{CE}}{\bar{x}} = 1 - e^{-\frac{\sigma^2 \gamma}{2}}$

4.4.2 CRRA application, Portfolio Construction (Merton 1969)

Problem Definition, Merton's 1969 Portfolio Problem

- 1 risky asset, 1 riskless asset
- Riskless asset $dR_y = r \cdot R_t \cdot dt$
- Risky asset $dR_y = \mu S_t \cdot dt + \sigma \cdot S_t \cdot dz_t$, (Geometric Brownian, more on this later)
- Given \$1 to invest, continuous rebalancing
- Determine π, fraction of W_t to allocate to risky asset to maximize expected utility of wealth $W = W_1$

The process for wealth with this construction is:

$$dW_t = (r + \pi(\mu - r)) \cdot W_t \cdot dt + \pi \cdot \sigma \cdot W_t \cdot dz_t$$

Solve with CRRA Utility $U(W) = \frac{W^{1-\gamma}}{1-\gamma}$, $\gamma \in (0, 1)$

Applying Ito's Lemma on $log(W_t)$ gives:

$$log(W_t) = \int_0^t (r + \pi(\mu - r) - \frac{\pi^2 \sigma^2}{2}) \cdot du + \int_0^t \pi \cdot \sigma \cdot dz_u$$
$$\rightarrow log(W) \sim \mathcal{N}(r + \pi(\mu - r) - \frac{\pi^2 \sigma^2}{2}, \pi^2 \sigma^2)$$

From the previous section, we need to maximize :

$$r + \pi(\mu - r) - \frac{\pi^2 \sigma^2 \gamma}{2}$$

therefore

$$\pi^* = \frac{\mu - r}{\gamma \sigma^2}$$

5 Stochastic Calc Primer

Taking a step back to do some math to understand what's happening...

5.1 Intro

Let's begin by looking at a basic random process. Consider a series of coin tosses. At each toss, a heads means a win of \$1, a tails means a loss of \$1. Let R_i be the outcome of the ith toss.
Clearly, R_i is a random variable:

$$\mathbb{E}[R_i] = 0,\ \mathbb{E}[R_i^2] = 1,\ \mathbb{E}[R_iR_j] = 0$$

Now, let $S_i = \sum_{j=1}^{i} R_j$, S_j is an example of a random walk.

$\mathbb{E}[S_i] = 0$, $\mathbb{E}[S_i^2] = \mathbb{E}[R_1^2 + 2R_iR_j + ...] = i$ importantly, however, $\mathbb{E}[S_i|S_1...S_j] = \mathbb{E}[S_i|S_j] = S_j$.

This process exhibits both the markov property and the martingale property.

5.2 Brownian Motion Properties

- Continuity: the paths are continuous
- Markov: $X(t|X(1)....X(\tau) = X(t|X(\tau)$ for $\tau \leq t$
- Martingale: Given information up until $\tau < t$ $\mathbb{E}[X(t)] = X(\tau)$
- Sample paths have "Quadratic Variation" i.e.

$$\lim_{h \longrightarrow 0} \sum_{i=m}^{n-1} (z_{(i+1)h} - z_{ih})^2 = h(n-m)$$

or equivalently

$$\int_S^T (dz_t)^2 = T - S$$

5.3 Stochastic calculus stuff

Define the *stochastic integral of f with respect to the brownian motion X* by:

$$W(t) = \int_0^t f(\tau) dX(\tau) := \lim_{n \longrightarrow \infty} \sum_{j=1}^{n} f(t_{j-1}(X(t_j) - X(t_{j-w})$$

where $t_j = \frac{jt}{n}$
This leads to *Stochastic differential euqations*

$$dW = f(t)dX$$

5.3.1 Mean Square Limit

Looking at

$$E[(\sum_{j=1}^{n}(X(t_j) - X(t_i))^2 - t)^2]$$

where $t_j = \frac{jt}{n}$
we note that $X(t_j) - X(t_i) \sim \mathcal{N}(0, t/n)$, therefore the above expectation is $O(\frac{1}{n}$, therefore we say

$$\sum_{j=1}^{n}(X(t_j) - X(y_i))^2 = t$$

in the *mean square limit*, we often write this as

$$\int_0^t (dX)^2 = t$$

5.3.2 Functions of stochastic variables

Stochastic functions do not behave as they do in normal calculus i.e. if $F = X^2$, it is not generally true that $dF = 2XdX$, for this we need Itô's Lemma
For the setup, allow $F(X)$ to be an arbitrary function, where $X(t)$ is a Brownian motion. If we consider a very small time scale $h = \frac{\delta t}{n}$ s.t. $F(X(t+h))$ can be approximated with a taylor series:

$$\begin{aligned}&F(X(t+h)) - F(X(t)) = \\ &\quad ((X(t+h) - C(t))\frac{dF}{dX}(X(t)) + \frac{1}{2}(X(t+h) - X(t))^2\frac{d^2F}{dX^2}(X(t)) + ... =\end{aligned}$$

from this we have that

$$\begin{aligned}((F(X(t+h)) - F(X(t))) + ((F(X(t+2h)) - F(X(t+h))) + ...\\ + ((F(X(t+nh)) - F(X(t+(n-1)h))) =\\ \sum_{j=1}^{n}(X(t+jh) - X(t+(j-1)h))\frac{dF}{dX}(X(t+(j-i)h)) + \\ \frac{1}{2}\frac{d^2F}{dX^2}(X(t))\sum_{j=1}^{n}(X(t+jh) - X(t+(j-1)h))^2 + ...\end{aligned}$$

We note that the rhs is simply $F(X(t+nh)) - F(X(t)) = F(X(t+\delta t)) - F(X(t))$ and that the lhs is just $\int_t^{t+\delta t}\frac{dF}{dX}dX + \int_t^{t+\delta t}\frac{1}{2}\frac{d^2F}{dX^2}(X(t))\delta t$ This leaves us :

$$F(X(t+\delta t)) - F(X(t)) = \int_t^{t+\delta t}\frac{dF}{dX}(X(\tau))dX(\tau) + \int_t^{t+\delta t}\frac{1}{2}\frac{d^2F}{dX^2}(X(\tau))d\tau$$

Extending this over arbitrary time scales, we get the integral version of Itô's Lemma

$$F(X(t)) = F(X(0)) + \int_t^{t+\delta t}\frac{dF}{dX}(X(\tau))dX(\tau) + \int_t^{t+\delta t}\frac{1}{2}\frac{d^2F}{dX^2}(X(\tau))d\tau$$

the differental form of this is:

$$dF = \frac{dF}{dX}dX + \frac{1}{2}\frac{d^2F}{dX^2}dt$$

Now looking at $F = X^2$ we see that if we apply Itô's Lemma we get

$$dF = 2XdX + dt$$

5.3.3 Black-Scholes

Some baseline definitions:

A stock S is usually modelled as $dS = \mu Sdt + \sigma SdX$, where μ is the drift and σ is the volatility

Allowing $F(S) - \log(S)$, we use Itô's Lemma to get

$$dF = \frac{dF}{dS}dS + \frac{1}{2}\sigma^2 S^2 \frac{d^2F}{dS^2}df = (\mu - \frac{1}{2}\sigma^2)dt + \sigma dX$$

so

$$S(t) = S(0)e^{(\mu - \frac{1}{2}\sigma^2)t + \sigma(X(t) - X(0))}$$

So $S(t)$ is not exactly a random walk, but rather it is a lognormal random walk

We would like to value a call option that gives us the right to buy a specific asset for an agreed amount at a specified time in the future. At expiration, the value of this option is clearly $\max(S - E, 0)$, to uncover how much we should pay for it now,we consider a portfolio of a position of ΔS assets

$$\Pi = V(S,t) - \Delta S$$

We note that the change in the portfolio from t to $t + dt$ is

$$d\Pi = dV - \Delta dS$$

Ito's lemma tells us that V must satisfy

$$dV = \frac{\partial V}{\partial t}dt + \frac{\partial V}{\partial S}dS + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2}dt$$

hence, the change in value of our portfolio is

$$d\Pi = (\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2})dt + (\frac{\partial V}{\partial S} - \Delta)dS$$

If we were to choose $\Delta = \frac{\partial V}{\partial S}$, we would remove all randomness from our portfolio, which is called *delta hedging*, a dynamic hedging strategy. After choosing Δ to be this, our portfolio changes by the amount

$$d\Pi = (\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2})dt$$

This is a "riskless" change. The no arbitrage principle states then this change must be the same as the growth we would get putting an equivalent amount of cash into a risk free interest bearing asset:

$$d\Pi = r\Pi dt$$

therefore

$$(\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2})dt = r\Pi dt$$

Therefore, as $\Pi = V - \Delta S = V - \frac{\partial V}{\partial S}S$

$$\frac{\partial V}{\partial t} = \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

6 HJB Equation and Merton's Portfolio Problem

Problem Statement:

- You will live for T more years

- $W_0 > 0$
- n risky assets, 1 riskless asset
- long and short positions allowed
- each asset as a normal distribution of treturns
- trading in continuous time, with no transaction costs
- continuous consumption of any fraction of wealth at any time

Problem Notation

- Riskless asset: $dR_t = r \cdot R_t \cdot dt$
- Risky assets: $dS_t = \mu \cdot S_)t \cdot dt + \sigma \cdot S_t \cdot dz_t$ (geometric brownian)
- $\mu > r > 0$, $\sigma > 0$ (for n assets, you need a covariance matrix)
- $W_t > 0$ wealth at time t
- $\pi(t, W_t) \in \mathbb{R}$, the fraction of wealth to be allocated to risky asset, (pi_t)
- $((1 - \pi(t, W_t)) \in \mathbb{R}$, the fraction of wealth to be allocated to riskless asset
- $c(t, W_t) \in \mathbb{R} \cap [0, W_t]$, wealth consumption per unit time, (c_t)
- utility of consumption $U(x) = \frac{x^{1-\gamma}}{1-\gamma},\ \gamma \in (0, 1)$
- utility of consumption $U(x) = log(x),\ \gamma = 1$
- γ (constant) Relative Risk-Aversion

We have the following constraint for our wealth at any timestep

$$dW_t = ((\pi_t \cdot (\mu - r) + r)W_t - c_t) + \pi_t \cdot \sigma \cdot W_t \cdot dz_t$$

At each timestep, we would like to determine the optimal $[pi_t, c_t]$ to maximize the expectation of future utility

$$E[\int_t^T \frac{e^{\rho(s-t)} \cdot c_s^{1-\gamma}}{1-\gamma} ds + \frac{e^{\rho(T-t)} \cdot B(T) \cdot W_T^{1-\gamma}}{1-\gamma} | W_t]$$

This is essentially a continuous time stochastic control problem

1. State, (t, W_T)
2. Action, $[\pi_t, c_t]$
3. Reward, $U(c_t)$
4. Return, accumulated discounted reward
5. find optimal Policy: $(t, W_t) \rightarrow [pi_t, c_t]$ to maximize expected return
6. we note that π_t is unconstrained however $c_t \geq 0$

We solve for the *Optimal Discounted Value Function*, the value function discounted to time zero

$$V^*(t, W_t) = \max_{\pi_t, c_t} \mathbb{E}[\int_t^T \frac{e^{-\rho s} \cdot c^{1-\gamma}}{1-\gamma} ds + \frac{e^{-\rho T} e^{\gamma} W_T^{1-\gamma}}{1-\gamma}]$$

We note that $V^*(t, W_t)$ satisfies the following recursive relation for all $t \leq t_1 < T$:

$$V^*(t, W_t) = \max_{\pi_t, c_t} \mathbb{E}[\int_t^{t_1} \frac{e^{-\rho s} \cdot c^{1-\gamma}}{1-\gamma} ds + V^*(t_1, W_{t_1})]$$

We can rewrite this in SDE form, and it gives us the HJB formulation:

$$\max_{\pi_t, c_t} \mathbb{E}[dV^*(t, w_t) + \frac{e^{-\rho t} \cdot c^{1-\gamma}}{1-\gamma} \cdot dt]$$

We use Ito's lemma to expand dV^*

$$dV^* = FIXME$$

removing the dz_t term as it is a matringale and we are taking the expectation, we get:

$$\max_{\pi_t, c_t}[\frac{\partial V^*}{\partial t} + \frac{\partial V^*}{\partial W_t}((\pi_t(\mu - r) + r)W_t - c_t) + \frac{\partial^2 V^*}{\partial W_t^2} \frac{\pi_t^2 \sigma^2 W_t^2}{2} + \frac{e^{-\rho t} \cdot c^{1-\gamma}}{1-\gamma}] = 0$$

We can call the LHS of this $\Phi(t, w_t; \pi_t, c_t)$, so we have

$$\max_{\pi_t, c_t} \Phi(t, w_t; \pi_t, c_t) = 0$$

To find optimal π_t, c_t we can simply differentiate $\Phi(t, w_t; \pi_t, c_t)$ with respsect to π_t and c_t and sept equal to zero.

$$\begin{aligned} \frac{\partial}{\partial \pi_t} \Phi(t, w_t; \pi_t, c_t) &= (\mu - r) \cdot \frac{\partial V^*}{\partial W_t} + \frac{\partial^2 V^*}{\partial W_t^2} \cdot \pi_t \cdot \sigma^2 \cdot W_t \\ &= 0 \\ &\downarrow \\ \pi_t^* &= \frac{-\frac{\partial V^*}{\partial W_t} \cdot (\mu - r)}{\frac{\partial^2 V^*}{\partial W_t^2} \cdot \sigma^2 \cdot W_t} \end{aligned}$$

Similarly we have

$$\begin{aligned} \frac{\partial}{\partial c_t} \Phi(t, w_t; \pi_t, c_t) &= -\frac{\partial V^*}{\partial W_t} + e^{\rho t} \cdot (c_t^*)^{\gamma} = 0 \\ &\downarrow \\ c_t^* &= (\frac{\partial V^*}{\partial W_t} \cdot e^{\rho t})^{-\frac{1}{\gamma}} \end{aligned}$$

We can substitute these back into our equation for $\Phi(t, w_t; \pi_t, c_t)$ giving us

$$\frac{\partial V^*}{\partial t} - \frac{(\mu - r)^2}{2\sigma^2} \cdot \frac{(\frac{\partial V^*}{\partial W_t})^2}{\frac{\partial^2 V^*}{\partial W_t^2}} + \frac{\partial V^*}{\partial W_t} \cdot r \cdot W_t + \frac{\gamma}{1-\gamma} \cdot e^{\frac{\rho t}{\gamma}} \cdot (\frac{\partial V^*}{\partial W_t})^{\frac{\gamma - 1}{\gamma}} = 0$$

This is just a second order PDE, with a boundary condition of

$$V^*(T, W_T) = e^{-\rho T} \cdot \epsilon^{\gamma} \cdot \frac{W_T^{1-\gamma}}{1-\gamma}$$

whatever we have we consume right before we die
We guess that the solution has the form

$$V^*(t, W_t) = f(t)^\gamma \cdot e^{-\rho t} \cdot \frac{W_t^{1-\gamma}}{1-\gamma}$$

then we would have

$$\frac{\partial V^*}{\partial t} = (\gamma \cdot f(t)^{\gamma-1} f'(t) - \rho f(t)^\gamma) \cdot e^{-\rho t} \cdot \frac{W_t^{1-\gamma}}{1-\gamma}$$

$$\frac{\partial V^*}{\partial W_t} = f(t)^\gamma \cdot e^{-\rho t} \cdot W_t^{-\gamma}$$

$$\frac{\partial^2 V^*}{\partial W_t^2} = -f(t)^\gamma \cdot e^{-\rho t} \cdot \gamma \cdot W_t^{-\gamma-1}$$

We substitude this back into our PDE, yielding

$$f'(t) = (\frac{\rho - (1-\gamma) \cdot (\frac{(\mu-r)^2}{2\sigma^2\gamma} + r)}{\gamma}) f(t) - 1$$

$f(T) = \epsilon$ if we allow $v = \frac{\rho-(1-\gamma)\cdot(\frac{(\mu-r)^2}{2\sigma^2\gamma}+r)}{\gamma}$
the ODE has a solution of

$$f(t) = \begin{cases} \frac{1+(v\epsilon)\cdot e^{-v(T-t)}}{v} \text{ for } v \neq 0 \\ T - t + \epsilon \text{ for } v = 0 \end{cases}$$

this yields to solution:

$$\pi^*(t, W_t) = \frac{\mu - r}{\sigma^2 \gamma}$$

$$c^*(t, W_t) = \frac{W_t}{f(t)} = \begin{cases} \frac{v \cdot W_t}{1+(v\epsilon)\cdot e^{-v(T-t)}} \text{ for } v \neq 0 \\ \frac{W_t}{T-t+\epsilon} \text{ for } v = 0 \end{cases}$$

$$V^*(t, W_t) = \begin{cases} e^{\rho t} \cdot \frac{(1+(v\epsilon-1)\cdot e^{-v(T-t)}\gamma}{c^\gamma} \cdot \frac{W_t^{1-\gamma}}{1-\gamma} \text{ for } v \neq 0 \\ e^{\rho t} \cdot \frac{(T-t+\epsilon)^\gamma W_t^{1-\gamma}}{1-\gamma} \text{ for } v \neq 0 \end{cases}$$

7 Derivatives Hedging with DRL

7.1 Background

Classical derivatives pricing/hedging theory is based on several key assumptions

- Arbitrage-Free Market - you cannot make money from nothing
- Complete Market - a market where all derivative payoffs can be replicated
- Risk Neutral Measure - an altered probability measure for movements of underlying securities

Arbitrage-Free and Complete Market assumptions lead to (dynamic, exact, and unique) replication of derivatives with underlying assumptions. Generally, ”friction-less” trading must be assumed to provide these market conditions. Replication then gives us pricing and hedging solutions for any derivative and these assumptions form the base for the Black-Scholes formulas.

In practice, we need to interact with an ”incomplete market”, which gives us the existence of multiple risk-neutral measures. The trader must then choose a risk-neutral measure to use in pricing a derivative.

An alternative approach is to view this as a portfolio optimization problem. We want to maximize the ”risk adjusted return” of the derivative plus hedges. This formulation allows us to set this up as a stochastic control problem, where at each time step we must decide what trades to make in the hedges.

In practice, this problem is not generally suitable for a dynamic programming approach due to the size of the state space and the complexity of the environment, (curses of dimensionality and modeling). Instead, we turn to deep reinforcement learning. JPM Deep Hedging Paper

the problem formulation looks as such:

- time is discreet and finite, steps $t = 0, 1, ...T$
- We hold a position D in m derivatives
- each of these m derivatives d_i expires at time $e_i \in [0, T]$
- We receive portfolio aggregated *Contingent Cashflows* at time t denoted as X_t
- We denote our hedge positions at time t as $\alpha_t \in \mathbb{R}^n$
- We also receive cashflows per unit of hedges held at time t denoted as $Y_t \in \mathbb{R}^n$
- The price of each hedge at time t is denoted as $P_t \in \mathbb{R}^n$
- Our PnL position at time t is $\beta_t \in \mathbb{R}$
- we denote the (continuous) state space at time t as $\mathcal{S}_t$, state at time t is ${}_t \in \mathcal{S}_t$
- $(t, \alpha_t, P_t, \beta_t, D) \subseteq s_t$
- denote the (continuous) action space at time t as $\mathcal{A}_t$, action at time t as $a_t \in \mathcal{A}_t$
- $a_t \in \mathbb{R}^n$ denotes the units of each hedge traded +/- for buy/sell

We rely on a simulator to provide samples of transitions $P_{t+1}|P_t$, and at each time step t we:

1. observe state $s_t = (t, \alpha_t, P_t, \beta_t, D, ...)$
2. Realize cashflows $= X_t + \alpha_t \cdot Y_t$
3. perform actions a_t to produce PnL $= -a_t \cdot P_t$
4. account for trading costs $-\gamma P_t \cdot |a_t|$ $(\gamma > 0)$
5. update α_t as $\alpha_{t+1} = \alpha_t + a_t$
6. update PnL as $\beta_{t+1} = \beta_t + X_t + \alpha_t\ cdotT_y - a_t \cdot P_t - \gamma P_t \cdot |a_t|$
7. reward $r_t = 0 \forall t \in [0, T)$, $r_T = U(\beta_{T+1})$

8 Pricing American Options with Reinforcement Learning

9 Pricing American Options with Reinforcement Learning

10 Stochastic Control for Optimal Trade Order Execution

First some definitions of a trading order book (TOB)

- Buyers / sellers submit bids /asks (buy/sell orders at a given prive)
  these are limit orders LO with defined with price P and size N
  a buy LO states intent /willingness to buy N shares at price $\leq P$
  a sell LO states intent /willingness to sell N shares at price $\geq P$
- TOB aggregates order sizes for each unique price
  bids: $[(P_i^b, N_i^b)|1 \leq i \leq m], P_i^b > P_j^b \forall i < j$
  asks: $[(P_i^a, N_i^a)|1 \leq i \leq n], P_i^a < P_j^a \forall i < j$
- we call P_1^b simply *Bid*, P_1^a simply Ask, $\frac{P_1^b + P_1^a}{2}$ as *Mid*
- We call $P_1^a - P_1^b$ the *Spread*, $P_n^a - P_m^b$ the *Market Depth*
- A Market order states intent to buy/sell N shares ath the best possible prices available on the TOB at the time of the MO submission

Now let's examine what the rough dynamics of the TOB look like

- A new sell LO (P,N) will potentially remove the best bid prices on the TOB
  Removes: $[P_i(b), \min(N_i^b, \max(0, N - \sum_{j=1}^{i-1} N_j^B)))|(i : P_i^b \geq P)]$
- After these are removed, it adds the following to the asks side of the TOB
  $(P, \max(0, N - \sum_{i: P_i^b \geq P} N_i^b$
- A new but LO operates analogously on the other side of the TOB
- A Sell Market order N will remove the N best big prices on the TOB.
  Removes: $[P_i(b), \min(N_i^b, \max(0, N - \sum_{j=1}^{i-1} N_j^B)))|(1 \leq i \leq m)]$

For this problem, we will focus how a large MO alters to TOB.

- A Large MO generally results in a big spread
- This spread could be replenished by new LOs on either side, so a large MO typically moves the Mid/Ask/Mid (price impact)

10.1 Problem Formulation: Optimal Trade Order Execution

- The task is to sell a large number N of shares
- We are allowed to trade in T discreete time steps
- For simplicity, we assume we can only submit market orders
- we will model both temporary and permanent price impact

- For simplicity, we only model the bid price dynamics
- Goal: Maximize expected total utility of sales proceeds

Problem Notation:

- Time steps $t = 1..T$
- P_t denote the bid price at time step t
- N_t denotes the number of shares sold in time step t
- $R_t = N - \sum_{i=1}^{t-1} N_i$ = shares remaining to be sold at time step t
- Price dynamics given by $P_{t+1} = f_t(P_t, N_t, \epsilon_t)$
- Sales proceeds given by $N_t \cdot Q_t = N_t \cdot (P_t - g_t(P_t, N_t))$
- Utility of sales proceeds given by $U(\cdot)$ utility function

MDP Formulation:

- at each timestamp $1 \leq t \leq T$
  Observe state (t, P_t, R_t)
  perform action N_t
  receive reward $U(N_t \cdot Q_t) = U(N_t \cdot (P_t - g_t(P_t, N_t)))$
  transition to new state $(t+1, f_t(P_t, N_t, \epsilon_t), R_t - N_t)$

We search for $\pi^*(t, P_t, R_t) = N_t$ that maximizes

$$\mathbb{E}[\sum_{t=1}^{T} \gamma^T \cdot U(N_t \cdot Q_t)]$$

The key choice in modelling this problem is the choice of price impact model. Consider the very basic case of a *Linear Price Impact*

$$P_{t+1} = P_t - \alpha N_t + \epsilon,\ \alpha \geq 0$$

$$Q_t = P_t - \beta N_t,\ \beta \geq 0$$

We note that we can rewrite the value function as

$$V^\pi(t, P_t, R_t) = \mathbb{E}_\pi[\sum_{i=t}^{T} N_i(P_i\ betaN_i)|(t, P_t, R_t)]$$

The optimal value function should satisfy the Bellman equation for every $1 \leq t < T$

$$V^*(t, P_t, R_t) = \max_{N_t}(N_t(P_t - \beta N_t))\mathbb{E}_\pi[V^*(t, P_t, R_t)]$$

with

$$V^*(T, P_t, R_t) = R_T(P_t - \beta R_T)$$

We can recursively backout

$$V^*(T-1, P_{t-1}, R_{T-1} = \max_{N_{T-1}} N_{T-1}(P_{T-1} - \beta N_{T-1}) + \mathbb{E}[R_T(P_T - \beta R_T)]$$

$$V^*(T-1, P_{t-1}, R_{T-1} = \max_{N_{T-1}} N_{T-1}(P_{T-1} - \beta N_{T-1}) + (R_{T-1} - N_{T-1}(P_{T-1} - \alpha N_{T-1}\ beta(R_{T-1} - N_{T-1}))$$

$$V^*(T-1, P_{t-1}, R_{T-1} = \max_{N_{T-1}} R_{T-1}P_{T-1} + (\alpha - 2\beta)(N_{T-1}^2 - N_{T-1}R_{T-1})$$

Now we note that there are two potential cases

- $\alpha \geq 2\beta$

  then we must have that $N_{T-1}^* \in \{), R_{T-1}\}$

  We sell all N shares at any one of the time steps, and sell nothing at any other time

- $\alpha < 2\beta$ differentiating w.r.t. N_{T-1} gives that

  $(\alpha - 1\beta)(2N_{T-1}^* - R_{T-1}) = 0$

  $N_{T-1}^* = \frac{R_{T-1}}{2}$

  taking this backwards recursively, we see that

  $$N_t^* = \frac{R_t}{T-t+1}$$

  this means we chunk up the shares left to be sold in to $T-t$ evenly sized orders, and execute them once per day

A slightly more involved model would be to include dependence on a Serially correlated Variable X_t

$$P_{t+1} = P_t - (\alpha N_t + \theta X_t)$$

$$X_{t+1} = \rho X_t \eta_t$$

$$Q_t = P_t + (\beta N_t + \theta X_t)$$

A more realistic model would be the *Linear-Percentage Temporary Price* Impact model (LPT)

- Price follows a gemoetric random walk, avoids prices leq 0
- there is only a temporary price impact

  $P_{t+1} = P_t e^{Z_t}$, $X_{T+1} = \rho X_t + \eta_t$, $W_t = P_t(1 - \beta N_t - \theta X_t)$

With the same derivation, we get

$$N_t^* = c_t^{(1)} + c_t^{(2)} R_t + c_t^{(3)} X_t$$

11 Stochastic Control for Optimal Market-Making

12 Model Free Prediction

Model Free Predicicction refers to solving the value function for an *unknown* MDP, unknown as in we do note have access to the transitions / rewards probabilities.

12.1 Monte-Carlo Reinforcement Learning

A few properties of MC methods

- We learn directly from episodes of experience
- MC is *model-free*
- MC learns from *complete* episodes: no bootstrapping

12.2 Monte-Carlo Policy Evaluation

Given an unknown MDP, and policy π, we want to learn v_π from a episodes of experience:

$$S_1, A_1, R_2, S_2 ... S_k \sim \pi$$

Recalling that return is the total discounted reward:

$$G_t = R_{t+1} + \gamma R_{t+2} + ..\gamma^{T-t} R_T$$

and that the value function should be the expected return from any state

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s]$$

Monte Carlo control uses the empirical mean from episodes to estimate the expected return.

12.2.1 First Visit Monte-Carlo Policy Evaluation

To evaluate a given state s, keep a counter $N(s_i)$ for all states, reward tally for each state $S(s_i)$, the first time a state is visited in an episode,

- $N(s) \leftarrow 1 + N(s)$
- $S(s) \leftarrow G_t + S(s)$

We estimate $V(s) = S(s)/N(s)$, which, by lln $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$.

12.2.2 Every Visit Monte-Carlo Policy Evaluation

To evaluate a given state s, keep a counter $N(s_i)$ for all states, reward tally for each state $S(s_i)$, every time a state is visited in an episode,

- $N(s) \leftarrow 1 + N(s)$
- $S(s) \leftarrow G_t + S(s)$

We estimate $V(s) = S(s)/N(s)$, which, by lln $V(s) \rightarrow v_\pi(s)$ as $N(s) \rightarrow \infty$.
We can cut out the middleman, $S(s_i)$ by noting that the update is equivalent to saying

$$V(S_i) \leftarrow V(s_i) + \frac{1}{N(s_i)}(G_t - V(s_i))$$

if we are dealing with a nonstationary environment, we can adjust this to

$$V(S_i) \leftarrow V(s_i) + \alpha(G_t - V(s_i))$$

13 Temporal Difference Learning

Temporal Difference, or TD:

- Learns directly from episodes of experience, complete or not (bootstrapping)
- TD is *model free*

Whereas in MC, we used the return G_t to update the value function for the state, in TD we use the reward observed plus the value state of the succeeding state

$$V(s_t) \leftarrow V(s_t) + \alpha(R_{t+1} + \gamma V(s_{t+1}) - V(s_t))$$

- We call $R_{t+1} + \gamma V(s_{t+1})$ the TD target
- We call $\delta_y = R_{t+1} + \gamma V(s_{t+1}) - V(s_t)$ the TD error

13.1 TD vs MC

One way to view the difference between MC and TD is the through the lens of the bias variance tradeoff. We note that the return, $G_t = R_{t+1} + \gamma R_{t+2} + .. + \gamma^{T-t} R_T$ is an unbiased estimate of $v_\pi(S_t)$, and that the true TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ is also an unbiased estimate of $v_\pi(S_t)$. However, the estimated TD target $R_{t+1} + \gamma V(S_{t+1})$ is a *biased* estimate of $v_\pi(S_t)$. That being said, the TD target has lower variance than the return.

There is a way to trade off between the two approaches, and this becomes TD-lambda

13.2 TD(λ)

....

14 Model-Free Control

First define:

- *On-policy* learning, learn about policy π from episodes sampled from policy π
- *Off-policy* learning, learn about policy π from episodes sampled from policy μ

14.1 Exploration

We can perform a greedy policy improvement over $V(s)$ if we have a model for the MDP

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} \mathcal{R}_s^a + \mathcal{P}_{ss'}^a V(s')$$

However, in an RL scenario we do not have have access to the underlying model, in that case, the update looks like

$$\pi'(s) = \arg\max_{a \in \mathcal{A}} Q(s, a)$$

There is a fundamental difference in this, in that we don't have the ground truth for the where actions will transition us to. Therefore, there may be actions that unexplored if we iteratively proceed with a purely greedy approach.

This leads into the idea of an *epsilon-greedy* approach to policy iteration. Consider the following policy:

$$\pi(a|s) = \begin{cases} \frac{\epsilon}{m} + 1 - \epsilon & \text{if } a^* = \arg\max_{a \in \mathcal{A}} Q(s,a) \\ \frac{\epsilon}{m} & \text{otherwise} \end{cases}$$

This is a kind of exploration policy, we note that there is always a non-zero probability of taking any given action.

The first thing to check is that this policy provides a guarantee that we will converge to an optimal policy.

$$\begin{aligned} q_\pi(s, \pi'(s)) &= \sum_{a \in \mathcal{A}} \pi'(a|s) q_\pi(s,a) && (1) \\ &= \frac{\epsilon}{m} \sum_{a \in \mathcal{A}} q_\pi(s,a) + (1-\epsilon) \max_{a \in \mathcal{A}} q_\pi(s,a) && (2) \\ &\geq \frac{\epsilon}{m} \sum_{a \in \mathcal{A}} q_\pi(s,a) + (1-\epsilon) \max_{a \in \mathcal{A}} \frac{\pi(a|s) - \frac{\epsilon}{m}}{1-\epsilon} q_\pi(s,a) && (3) \\ &= \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s,a) = v_\pi(s) && (4) \end{aligned}$$

In order for this ϵ-greedy policy improvement strategy to converge, we need GLIE (greedy in the limit with infinite exploration).

- $\lim_{k \to \infty} N_k(s,a) = \infty$
- $\lim_{k \to \infty} \pi_k(s) = 1(a^* = \arg\max_{a \in \mathcal{A}} Q(s,a))$

Note that setting $\epsilon = \frac{1}{k}$ accomplishes this.

14.2 On Policy MC Control

To calculate $q^*(s,a)$ for an unknown MDP, at the kth iteration of our algorithm:

- Sample kth episode using π
- for each state and action pair in the episode: (S_t, A_t)
  - $N(S_t, A_t) \leftarrow N(S_t, A_t) + 1$
  - $Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \frac{1}{N(S_t, A_t)}(G_t - Q(S_t, A_t))$
- $\epsilon \leftarrow \frac{1}{k}$
- $\pi \leftarrow \epsilon - \text{greedy}(Q)$

With this setup, we have convergence to the optimal action-value function.

14.3 Sarsa(λ)

Sarsa refers to the order in which we observe datapoints while we learn: S(tate)a(ction)r(eward)s(tate)a(ction)

Updating our action value function with SARSA looks like:

$$Q(S,A) \leftarrow Q(S,A) + \alpha(R + \gamma Q(S', A') - Q(S,A))$$

Algorithm 1 SARSA

```
procedure Sarsa Iteration(MDP)
    Initialize $Q(s, a) \forall s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily, and $Q(\text{terminal state}, \cdot) = 0$
    for each episode do
        Initialize $S$
        Choose $A$ from $\mathcal{A}_S$ using policy derived from $Q$ (e.g. $\epsilon$-greedy)
        while $S$ is not terminal do
            Take action $A$, observe $R, S'$
            Choose $A'$ from $\mathcal{A}_{S'}$ using policy derived from $Q$ (e.g. $\epsilon$-greedy)
            $Q(S, A) \leftarrow Q(S, A) + \alpha(R + \gamma Q(S', A') - Q(S, A))$
            $S \leftarrow S'$, $A \leftarrow A'$
```

The algorithm implementing Sarsa is as follows If we follow a GLIE sequence of policies, and have a *Robbins-Monro* sequence of step-sizes α_t i.e. ($\sum_{t=1}^{\infty} \alpha_t = \infty$, $\sum_{t=1}^{\infty} \alpha_t^2 < \infty$) then we are gauranteed convergence of q to the optimal action-value function.

Consider if we extend the Sarsa algorithm out n steps, we can definte the n-step Q-return as

$$q_t^{(n)} = R_{t+1} + \gamma R_{t+2} + ... + \gamma^n Q(S_{t+n})$$

and the n-step Sarsa update as

$$Q(S_t, A_t) \xleftarrow{Q(S_t,A_t)+\alpha(q_t^{(n)}-Q(S_t,A_t))}$$

We could also sondier a q^λ return, combining all n-step Q return $q_t^{(n)}$ using weight $(1-\lambda)\lambda^{n-1}$ i.e. $q_t^\lambda = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} q_t^{(n-1)}$, then our Sarsa updates look like

$$Q(S_t, A_t) \xleftarrow{Q(S_t,A_t)+\alpha(q_t^\lambda-Q(S_t,A_t))}$$

Similarly to how we did with $TD(\lambda)$, we can maintain eligibility traces in an online algorithm to implement $Sarsa(\lambda)$, this gives us

Algorithm 2 SARSA

```
procedure Sarsa Iteration(MDP)
    Initialize $Q(s, a) \forall s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily, and $Q(\text{terminal state}, \cdot) = 0$
    for each episode do
        Initialize $S$
        Choose $A$ from $\mathcal{A}_S$ using policy derived from $Q$ (e.g. $\epsilon$-greedy)
        while $S$ is not terminal do
            Take action $A$, observe $R, S'$
            Choose $A'$ from $\mathcal{A}_{S'}$ using policy derived from $Q$ (e.g. $\epsilon$-greedy)
            $\delta \leftarrow R + \gamma Q(S', A') - Q(S, A)$
            $E(S, A) \leftarrow E(S, A) + 1$
            for all $s \in \mathcal{S}, a \in \mathcal{A}$ do
                $Q(s, a) \leftarrow Q(s, a) + \alpha \delta E(s, a)$
                $E(s, a) = \gamma \lambda E(s, a)$
            $S \leftarrow S'$, $A \leftarrow A'$
```

14.4 Off Policy Learning

Similar to how we did with prediction, we would like to be able to learn the optimal policy or even evaluate a target policy π based on experiences from some other policy μ.

14.4.1 Importance Sampling

We note that we are able to estimate the expectation of a difference distribution by

$$\begin{aligned}\mathbb{E}_{X\sim P}[f(X)] &= \sum P(X)f(x)\\ &= \sum Q(X)\frac{P(X)}{Q(X)}f(x)\\ &= \sum \mathbb{E}_{X\sim Q}[\frac{P(X)}{Q(X)}f(x)]\end{aligned}$$

We can use this for both Off-Policy Monte-Carlo and Off-Policy TD. For Monte-Carlo, we use weighted returns based on the similarity of μ and π

$$G_t^{\pi/mu} = \frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}\frac{\pi(A_{t+1}|S_{t+1})}{\mu(A_{t+1}|S_{t+1})}\cdots\frac{\pi(A_T|S_T)}{\mu(A_T|S_T)}$$

then we just update the value function towards to weighted return

$$V(S_t) \leftarrow V(S_t) + \alpha(G_t^{\pi/\mu} - V(S_t))$$

For TD we do a simliar weighting but we only look forward one step

$$V(S_t) \leftarrow V(s_t) + \alpha(\frac{\pi(A_t|S_t)}{\mu(A_t|S_t)}(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

14.4.2 Q-Learning

The central idea of Q-Learning is to consider an alternative successor action than the stream received from the episode. i.e. if the action stream is $S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$, we instead update our Q function with A'

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma Q(S_{t+1}, A') - Q(S_t, A_t))$$

We can not allow our policy to choose the next A' to be purely greedy

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha(R_{t+1} + \gamma \max_{A'\in\mathcal{A}} Q(S_{t+1}, A') - Q(S_t, A_t))$$

The algorithm for Q learning is elegantly simple

Algorithm 3 SARSA

```
procedure Sarsa Iteration(MDP)
    Initialize $Q(s,a)\forall s \in \mathcal{S}, a \in \mathcal{A}$ arbitrarily, and $Q(\text{terminal state}, \cdot) = 0$
    for each episode do
        Initialize $S$
        while $S$ is not terminal do
            Choose $A$ from $\mathcal{A}_S$ using policy derived from $Q$ (e.g. $\epsilon$-greedy)
            Take action $A$, observe $R, S'$
            $Q(s,a) \leftarrow Q(s,a) + \alpha[R + \gamma \max_{A'\in\mathcal{A}} Q(S', A') - Q(S, A)]$
            $S \leftarrow S'$
```

15 Value Function Approximation

Value function approximation appears when the tabular approach to RL we have been using so farm, in which we discretize every state and action pair, becomes computationally infeasible.
The idea behind value function approximation is as follows, instead of learning a discreet mapping from state to value, we learn a parameterized mapping from a set of features to value.

$$\hat{v}(s, \precapprox) \approx v_\pi(s)$$

or

$$\hat{q}(s, a, \precapprox) \approx q_\pi(s, a)$$

This also gives us the ability to (hopefully) generalize to unseen states (i.e. we don't need to have seen a state exactly before if we've seen something close to it).

There are many kind of function approximators, some popular ones being

1. affine functions
2. Neural Networks
3. Decision Trees
4. clustering techniques like knn
5. Fourier/wavelet bases
6. ...

This first requires some sort of featurization of our state space. i.e. we need a mapping : $\mathcal{S} \to \mathbb{R}^n$ that captures relevant information from the state. our value function then takes the form:

$$\hat{v}(S, w) = \phi(x(S), w)$$

where $\phi(\cdot, w)$ is the form of our function approximator parameterized by w (weights, biases, etc).

Fitting whatever function we choose to use follows the normal training routine (batch or iterative) except that our labels at any point are at best approximations. If we are using a linear form for our function approximator, then our updates look as such:

1. MC, the target is G_t

   $\mathbf{\Delta w = \alpha(G_t - \hat{v}(S_t, w)\nabla_w \hat{v}(S_t, w)}$

2. TD(0), the target is $R_t + \gamma\hat{v}(S_{t+1}, \mathbf{w})$

   $\mathbf{\Delta w = \alpha(R_t + \gamma\hat{v}(S_{t+1}, w) - \hat{v}(S_t, w)\nabla_w \hat{v}(S_t, w)}$

3. TD(λ), the target is G_t^λ

   $\mathbf{\Delta w = \alpha(G_t^\lambda - \hat{v}(S_t, w)\nabla_w \hat{v}(S_t, w)}$

15.1 Action-Value Function Approximation

We follow a similar stragegy to approximate $\hat{q}(S, A, w) \approx q(S, A)$

Again we define some sort of loss function to minimize i.e. SSE

$$J(w) = \mathbb{E}_\pi[(q_\pi(S, A) - \hat{q}(S, A, w))^2]$$

However, as we do not have access to the true value function, we again replace this metric with an approximation *target* based upon the current value of our value function.

1. MC, the target is G_t

   $\Delta \mathbf{w} = \alpha(\mathbf{G_t} - \hat{\mathbf{q}}(\mathbf{S_t}, \mathbf{A_t}, \mathbf{w})\nabla_{\mathbf{w}}\hat{\mathbf{q}}(\mathbf{S_t}, \mathbf{A_t}, \mathbf{w})$

2. TD(0), the target is $R_t + \gamma\hat{q}(S_{t+1}, A_{t+1}, \mathbf{w})$

   $\Delta \mathbf{w} = \alpha(\mathbf{R_t} + \gamma\hat{\mathbf{q}}(\mathbf{S_{t+1}}, \mathbf{A_{t+1}}, \mathbf{w}) - \hat{\mathbf{q}}(\mathbf{S_t}, \mathbf{A_t}, \mathbf{w})\nabla_{\mathbf{w}}\hat{\mathbf{q}}(\mathbf{S_t}, \mathbf{A_t}, \mathbf{w})$

3. For forward-view TD(λ), the target is the λ-return

   $\Delta \mathbf{w} = \alpha(\mathbf{q_t^\lambda} - \hat{\mathbf{q}}(\mathbf{S_t}, \mathbf{A_t}, \mathbf{w})\nabla_{\mathbf{w}}\hat{\mathbf{q}}(\mathbf{S_t}, \mathbf{A_t}, \mathbf{w})$

4. For backward-view TD(λ), the target is the λ-return

   $\delta_t = R_{t+1} + \gamma\hat{q}(S_t, A_t, \mathbf{w}))\nabla_{\mathbf{w}}\hat{\mathbf{q}}(\mathbf{S_t}, \mathbf{A_t}, \mathbf{w})$

   $E_t = \gamma\lambda E_{t-1} + \nabla_{\mathbf{w}}\hat{q}(S_t, A_t, w)$ $\Delta \mathbf{w} = \alpha\delta_{\mathbf{t}}\mathbf{E_t}$

15.2 Batch Methods

While incremental updates to our parameters (i.e. gradient descent) is simple and appealing, it is not necessarily *sample efficient*. As in traditional supervised methods, there are batch methods to fit the parameters of the model to the data (observed experiences).
Assume we have a set of experiences

$$\mathcal{D} = \{(s_t, a_t, r_{t+1}, s_{t+1})_i\}$$

We can define a loss function for our approximation function, i.e. least squares

$$\mathcal{L}(\mathbf{f}) = \sum_{\mathbf{e}\in\mathcal{D}}(\hat{\mathbf{v}}(\mathbf{e_s}) - \mathbf{v}_\pi(\mathbf{e_s}))^{\mathbf{2}}$$

where $v_\pi(s)$ is the value function we are looking to fit to, however as it is unknown, like with iterative methods, we need an approximate label. For linear approximation functions, the updates look like this:

- Least Squares Monte Carlo (LSMC)

  Target is G_e

  $\min_{\mathbf{w}} \sum_{e\in\mathcal{D}}(\hat{v}(e_s) - G_e)^2$

  $= 0 \sum_{e\in\mathcal{D}}(\hat{v}(e_s) - G_e)x(e_s)$

- Least Squares TD (LSTD)

  Target is $e_{r_{t+1}} + \gamma\hat{v}(e_{s_{t+1}}, \mathbf{w})$

  $\min_{\mathbf{w}} \sum_{e\in\mathcal{D}}(\hat{v}(e_s) - (e_{r_{t+1}} + \gamma\hat{v}(e_{s_{t+1}}, \mathbf{w})))^{\mathbf{2}}$

  $= 0 \sum_{e\in\mathcal{D}}(\hat{v}(e_s) - (e_{r_{t+1}} + \gamma\hat{v}(e_{s_{t+1}}, \mathbf{w})))\mathbf{x}(\mathbf{e_s})$

- Least Squares TD-lambda (LSTD(λ))

  Target is G_e^λ
  $\min_{\mathbf{w}} \sum_{e \in \mathcal{D}} (\hat{v}(e_s) - G_e^\lambda)^2$
  $= 0 \sum_{e \in \mathcal{D}} (\alpha delta_e E_e)$

We can follow a similar structure for action value function approximation, and this will lead us to LSTDQ, iteratively fitting a value function with approximate labels. However we need to introduce some π s.t. this π generates the next action from the next state, we can simply use a greedy policy from the current form of our q function
again with $\mathcal{D} = \{(s_t, a_t, r_{t+1}, s_{t+1})_i\}$

$$\min_{\mathbf{w}} \sum e \in \mathcal{D}(\hat{q}(S_t, A_t, \mathbf{w}) - (\mathbf{r_{t+1}}) + \gamma \mathbf{\hat{q}(s_{t+1}}, \pi(\mathbf{s_{t+1}}))^{\mathbf{2}}$$

Again, we would take the gradient w.r.t. w and set to zero. A complication arises however as w parameterizes both our labels and our predictions, so in practice we fix our labels before iterating a batch, then we fit our parameters, and then repeat. This gives us *LSTDQ*

Algorithm 4 LSTDQ

```
procedure LSTDQ($\mathcal{D}, \pi_0$)
    $\pi \leftarrow \pi_0$
    $\pi(s)' \leftarrow \arg\max_{a \in \mathcal{A}} Q(s,a) \forall s \in \mathcal{S}$
    while $\pi' \neq \pi$ do
        $\pi \leftarrow \pi'$
        Q $\leftarrow$ LSTD($\pi, \mathcal{D}$)
        $\pi(s)' \leftarrow \arg\max_{a \in \mathcal{A}} Q(s,a) \forall s \in \mathcal{S}$
    return $\pi$
```

16 Value Function Geometry and Gradient TD

For this section, let's restate some notation

- State space $\mathcal{S} = \{s_1, s_2..s_n\}$
- *finite* action space $\mathcal{A}$
- fixed, stochastic policy $\pi(a|s)$
- VF for a given policy π $v_\pi(s) : \mathcal{S} \to \mathbb{R}$
- featureization of state space: $\pi(s) : \mathcal{S} \to \mathbb{R}^m$
- linear value function approximation $\mathbf{v_w(s)} = \mathbf{w^T} \cdot \phi(\mathbf{s})$
- $\mu_\pi : \mathcal{S} \to [0,1]$ denotes the states probability distribution under π

We can think of value functions as vectors in an n dimensional space i.e. $v_\pi(S) = (..v_\pi(s_i)..)^T \in \mathbb{R}^n$. Now is we consider an nxm matrix $\mathbf{\Phi}$, we note that this is a linear transformation, we note that $v_w(S) = \mathbf{\Phi w}$ so we can consider $v_w(S)$ as the image of w under the transformation $\mathbf{\Phi}$

Some more notation:

- let $r(s,a)$ be the expected reward of action a in state s

- denote $p(s, s', a)$ as the probability of transitioning from state s to s' upon action a
- define

$$\mathbf{R_\pi(s)} = \sum_{\mathbf{a}\in\mathcal{A}} \pi(\mathbf{a}|\mathbf{s}) \cdot \mathbf{r(s, a)}$$

$$\mathbf{p_\pi(s, s')} = \sum_{\mathbf{a}\in\mathcal{A}} \pi(\mathbf{a}|\mathbf{s}) \cdot \mathbf{p(s, s', a)}$$

- define

$$\mathbf{R_\pi} = [\ldots \mathbf{R_\pi(s_i)} \ldots]$$

$$\mathbf{P_\pi} = \begin{bmatrix} \ddots & & \iddots \\ & \mathbf{R_\pi(s_i, s_j)} & \\ \iddots & & \ddots \end{bmatrix}$$

- γ the discount factor

We can define the bellman operator $\mathbf{B_\pi}$ as

$$\mathbf{B_\pi v} = \mathbf{R_\pi} + \mathbf{P_\pi} \cdot \mathbf{v}$$

We can also define the distance between two value function (vectors), we weight it by μ_π, where $\mathbf{D} = \mathbf{diag}(\mu_\pi)$

$$d(\mathbf{v_1}, \mathbf{v_2}) = (\mathbf{v_1} - \mathbf{v_2})^\mathbf{T} \cdot \mathbf{D} \cdot (\mathbf{v_1} - \mathbf{v_2})$$

Now remembering $\mathbf{\Phi}$, we introduce $\mathbf{\Pi}_\phi$ as performing the orthogonal projection onto $\mathbf{\Phi}$. We note that $\mathbf{\Pi}_\phi\mathbf{v}$ is the value function projected into the subspace $\mathbf{\Phi}$ defined as

$$arg \min_w d(\mathbf{v}, \mathbf{v_w})$$

i.e.

$$\mathbf{w} = (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi})^{\mathbf{-1}}\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{v}$$

so we can write

$$\mathbf{\Pi_\Phi} = \mathbf{\Phi} \cdot (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi})^{\mathbf{-1}}\mathbf{\Phi^T} \cdot \mathbf{D}$$

There are 4 quantities we are interested in

1. The projection of v_π in the $\mathbf{\Phi}$ defined subspace, $\mathbf{w}_\pi$

   this is what we would ideally like to converge to during function approximation, and monte carlo with linear function approximation will slowly converge to $\mathbf{w}_\pi$

2. The Bellman Error minimizing $\mathbf{w_{BE}} = \mathbf{arg}\min_\mathbf{w} \mathbf{d}(\mathbf{B_\pi v}, \mathbf{v_w})$

   It turns out that if we can only access features, not the underlying states, we cannot learn this

3. TDE minimizing $\mathbf{w_{TDE}} = \mathbf{arg}\min_\mathbf{w} \mathbb{E}_\pi[\delta^\mathbf{2}]$

4. Projected Bellman Error minimizing

$$\mathbf{w_{PBE}} = \arg\min_\mathbf{w} \mathbf{d}(\mathbf{\Phi} \cdot \mathbf{B_\pi v}, \mathbf{v_w})$$

   This is the fixed point of the projected bellman operator $\mathbf{\Phi} \cdot \mathbf{B_\pi}$

   This is generally what we will shoot for, as it performs well in practice and is usually attainable

16.0.1 Solution to WBE with Linear system

$$\begin{aligned}\mathbf{w_{BE}} &= \arg\min_{w} d(\mathbf{v_w}, \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \cdot \mathbf{v}_\pi) \\ &= \arg\min_{w} d(\mathbf{\Phi} \cdot \mathbf{w}, \mathbf{R}_\pi + \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi} \cdot \mathbf{w}) \\ &= \arg\min_{w} d(\mathbf{\Phi} \cdot \mathbf{w} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi} \cdot \mathbf{w}, \mathbf{R}_\pi) \\ &= \arg\min_{w} d((\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi}) \cdot \mathbf{w}, \mathbf{R}_\pi)\end{aligned}$$

This is just the weighted LS regression of $\mathbf{R}_\pi$ onto $\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi}$. the solution is

$$\mathbf{w_{BE}} = ((\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi})^{\mathbf{T}} \cdot \mathbf{D} \cdot (\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi}))^{\mathbf{-1}} \cdot (\mathbf{\Phi} - \gamma \mathbf{P}_\pi \cdot \mathbf{\Phi})^{\mathbf{T}} \cdot \mathbf{D} \cdot \mathbf{R}_\pi$$

16.0.2 Gradient TD

Recalling that $\mathbf{w_{PBE}} = \mathbf{arg}\min_{\mathbf{w}} \mathbf{d}(\mathbf{\Pi_\Phi B v_w}, \mathbf{v_w}) = \mathbf{arg}\min_{\mathbf{w}} \mathbf{d}((\mathbf{\Pi_\Phi B v_w}, \mathbf{\Pi_\Phi v_w})$
With $\delta_{\mathbf{w}} = \mathbf{B v_w} - \mathbf{v_w}$ We define our loss function as

$$\begin{aligned}\mathcal{L}(\mathbf{w}) = (\mathbf{\Pi_\Phi} \delta_{\mathbf{w}})^{\mathbf{T}} \cdot \mathbf{D} \cdot (\mathbf{\Pi_\Phi} \delta_{\mathbf{w}}) &= \delta_{\mathbf{w}}^{\mathbf{T}} \cdot \mathbf{\Pi_\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Pi_\Phi} \cdot \delta_{\mathbf{w}} \\ = \delta_{\mathbf{w}}^{T} \cdot (\mathbf{\Phi} \cdot ((\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi^T} \cdot \mathbf{D})^{\mathbf{T}} \cdot \mathbf{D} \cdot (\mathbf{\Phi} \cdot ((\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi^T} \cdot \mathbf{D}) \cdot \delta_{\mathbf{w}} \\ = \delta_{\mathbf{w}}^{T} \cdot (\mathbf{D} \cdot \mathbf{\Phi} \cdot (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi^T}) \cdot \mathbf{D} \cdot (\mathbf{\Phi} \cdot ((\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot \mathbf{\Phi^T} \cdot \mathbf{D}) \cdot \delta_{\mathbf{w}} \\ = (\delta_{\mathbf{w}}^{T} \cdot \mathbf{D} \cdot \mathbf{\Phi}) \cdot (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi}) \cdot (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \delta_{\mathbf{w}}) \\ = (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})^{\mathbf{T}} \cdot (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})\end{aligned}$$

Thus we have that

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \mathbf{2} \cdot (\nabla_{\mathbf{w}} (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})^{\mathbf{T}}) \cdot (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \mathbf{\Phi})^{-1} \cdot (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})$$

Now, allowing

- $\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \delta_{\mathbf{w}} = \mathbb{E}[\delta \cdot \phi(\mathbf{s})]$
- $\nabla_{\mathbf{w}} (\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \delta_{\mathbf{w}})^{\mathbf{T}} = \nabla_{\mathbf{w}} \mathbb{E}[\delta \cdot \phi(\mathbf{s})]^{\mathbf{T}} = \mathbb{E}[(\gamma \cdot \phi(\mathbf{s'}) - \phi(\mathbf{s})) \cdot \phi(\mathbf{s})^{\mathbf{T}}]$
- $\mathbf{\Phi^T} \cdot \mathbf{D} \cdot \delta_{\mathbf{w}} = \mathbb{E}[\phi(\mathbf{s}) \cdot \phi(\mathbf{s})^{\mathbf{T}}]$

We then have

$$\nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) = \mathbf{2} \cdot \mathbb{E}[(\gamma \cdot \phi(\mathbf{s'}) - \phi(\mathbf{s})) \cdot \phi(\mathbf{s})^{\mathbf{T}}] \cdot \mathbb{E}[\phi(\mathbf{s}) \cdot \phi(\mathbf{s})^{\mathbf{T}}]^{\mathbf{-1}} \cdot \mathbb{E}[\delta \cdot \phi(\mathbf{s})]$$

Coming to our weight updates for the TDC algorithm, we wee that

$$\begin{aligned}\Delta w &= -\frac{1}{2} \alpha \cdot \nabla_{\mathbf{w}} \mathcal{L}(\mathbf{w}) \\ &= \alpha \cdot \mathbb{E}[(\gamma \cdot \phi(s') - \phi(s)) \cdot \phi(s)^T] \cdot \mathbb{E}[\phi(s) \cdot \phi(s)^T]^{-1} \cdot \mathbb{E}[\delta \cdot \phi(s)] \\ &= \alpha \cdot (\mathbb{E}[(\gamma \cdot \phi(s') \cdot \phi(s)^T] - \mathbb{E}[\phi(s) \cdot \phi(s)^T]) \cdot \mathbb{E}[\phi(s) \cdot \phi(s)^T]^{-1} \cdot \mathbb{E}[\delta \cdot \phi(s)] \\ &= \alpha \cdot (\gamma \mathbb{E}[\delta \cdot \phi(s)] - \mathbb{E}[\phi(s) \cdot \phi(s)^T] \cdot \mathbb{E}[\phi(s) \cdot \phi(s)^T]^{-1} \cdot \mathbb{E}[\delta \cdot \phi(s)]) \\ &= \alpha \cdot (\gamma \mathbb{E}[\delta \cdot \phi(s)] - \mathbb{E}[\phi(s) \cdot \phi(s)^T] \cdot \theta)\end{aligned}$$

where $\theta = \mathbb{E}[\phi(s) \cdot \phi(s)^T]^{-1} \cdot \mathbb{E}[\delta \cdot \phi(s)]$, the solution the the weighted ls regression of $\mathbf{B}_\pi \mathbf{v} - \mathbf{v}$ against $\mathbf{\Phi}$
We note that we can update both θ and w at the same time,

- $\Delta w = \alpha \cdot]detla \cdot \phi(s) - \alpha \cdot \gamma \cdot \phi(s') \cdot (\theta^T \cdot \phi(s))$
- $\Delta \theta = \beta \cdot (\delta - \theta^T \cdot \phi(s)) \cdot \phi(s)$

noting that $\theta^T \cdot \phi(s)$ is an approximation for the TD error δ for state s

17 Policy Gradient Algorithms

So far we have examined control from the point of view of learning some sort of value function for actions, and then choosing the action with the best expected return. When the size of the action space becomes untenable large, this breaks down. What we're searching for is the policy that fetches the "best expected returns." What we can do, is approximate our policy with a function

1. $\pi(s, a; \theta)$ function approximation for the Policy function: Actor
2. $Q(s, a; w)$ function approximation for the action value function: Critic

We can then perform policy improvement by **Gradient Ascent**
Here's how this breaks down vs what we have done previously:

1. Value Function Based

   Learn Value Function (function approx)

   Policy is implicit, (e.g. ϵ-greexy)

2. Policy Based

   Learn Policy (function approx)

   No need to learn VF

3. Actor-Critic

   Learn Policy (Actor)

   Learn VF (Critic)

Some advantages of the policy gradient approach are

1. We learn the optimal stochastic policy
2. exploration is natural due to the stochasticity of the policy
3. it can be effective in high dimensional or continuous action spaces
4. small changes in θ lead to small changes in π and in state distribution (greedy policy can change wildly with small changes in vf)

There is no free lunch, however, and some disadvantages are:

1. We don't have convergence gaurantees to global optimal values, and in fact it is typically local optimal convergence
2. policy evaluation is typically inefficient and has high variance
3. policy improvement happens in small steps and can lead to slow convergence

Notation!

1. discount factor γ

2. episodic → $\gamma \in [0,1]$
3. non-episodic → $\gamma \in [0,1)$
4. state $s_t \in \mathcal{S}$
5. actions $a_t \in \mathcal{A}$
6. rewards $r_t \in \mathbb{R}$
7. state transition probabilities $\mathcal{P}^a_{s,s'} = \mathbb{P}(s_{t+1} = s' | s_t = s, a_t = a)$
8. expected rewards: $\mathcal{R}^a_s = \mathbb{E}[r_t | s_t = s, a_t = a]$
9. initial state probability distribution $p_0 : \mathcal{S} \to [0, 11$
10. policy function approximation $\pi(s, a; \theta) = \mathbb{P}(a_t = a | s_t = s, \theta)$, $\theta \in \mathbb{R}^k$

Now let's get to what we are optimizing: Expected Returns: $J(\theta)$

$$J(\theta) = \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t]$$

We also note that we can define the value and action value functions as:

$$V^\pi(s) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} r_k | s_t = s]$$
$$Q^\pi(s,a) = \mathbb{E}_\pi[\sum_{k=t}^{\infty} \gamma^{k-t} r_k | s_t = s, a_t = a]$$

at this point let's also define the advantage function:

$$A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$$

Which gives us the idea of a "gain" of taking action a in state s over the expected distribution of actions. Now let's look at the loss again,

$$\begin{aligned}
J(\theta) &= \mathbb{E}_\pi[\sum_{t=0}^{\infty} \gamma^t r_t] \\
&= \sum_{t=0}^{\infty} \gamma^t \mathbb{E}_\pi[r_t] \\
&= \sum_{t=0}^{\infty} \gamma^t \int_{\mathcal{S}} (\int_{\mathcal{S}} p_0(s_0) \cdot p(s_0 \to s, t, \pi) \cdot ds_0) \int_{\mathcal{A}} \pi(s, a; \theta) \cdot \mathcal{R}^a_s \cdot da \cdot ds \\
&= \int_{\mathcal{S}} (\int_{\mathcal{S}} \sum_{t=0}^{\infty} \gamma^t p_0(s_0) \cdot p(s_0 \to s, t, \pi) \cdot ds_0) \int_{\mathcal{A}} \pi(s, a; \theta) \cdot \mathcal{R}^a_s \cdot da \cdot ds
\end{aligned}$$

At this point it makes sense to introduce the idea of the Discounted-Aggregate State-Visitation Measure

$$\rho^\pi(s) = \int_{\mathcal{S}} \sum_{t=0}^{\infty} \gamma^t p_0(s_0) \cdot p(s_0 \to s, t, \pi) \cdot ds_0$$

then our loss function befomes

$$J(\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s, a; \theta) \cdot \mathcal{R}^a_s \cdot da \cdot ds$$

17.1 Policy Gradient Theorem

what we would now like to show is that

$$\nabla_\theta J(\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi(s,a;\theta) \cdot Q^\pi(s,a) \cdot da \cdot ds$$

this gives us several advantages:

- $\rho^\pi(s)$ depends on θ but there is no theta gradient term in
- We can thus simply sample simulation paths, and at each time step calculate $\nabla_\theta log(\pi(s,a;\theta)) \cdot Q^\pi(s,a)$
- $\nabla_\theta log(\pi(s,a;\theta))$ is the score function, gradient of log likelihood
- We will estimate $Q^\pi(s,a)$ with a function approx $Q(s,a,;)$
- the numerical estimate $\nabla_\theta J(\theta)$ is what enables policy gradient ascent

To examine this further, let's look at specific forms for $\pi(s,a;\theta)$.
A canonical function for finite action spaces is the softmax function. (again let $\pi(s,a) \in \mathbb{R}^n$ be a feature vector for a given state and action.

$$\pi(s,a;\theta) = \frac{e^{\theta^T \phi(s,a)}}{\sum_b e^{\theta^T \phi(s,a)}}$$

with this form we have that

$$\nabla_\theta log(\pi(s,a;\theta)) = \phi(s,a) - \sum_b e^{\theta^T \phi(s,a)} = \phi(s,a) - \mathbb{E}[\phi(s,\cdot)]$$

For continuous action spaces, we often use a Gausian Policy, now $\phi(s) \in \mathbb{R}^n$ is the state feature vector, policy is:

$$a \sim \mathcal{N}(\theta^T \phi(s), \sigma^2)$$

and the score function is

$$\nabla_\theta log(\pi(s,a;\theta)) = \frac{(a - \theta^T \phi(s))\phi(s)}{\sigma^2}$$

17.1.1 Policy Gradient Theorem Proof

We first note that

$$\begin{aligned} J(\theta) &= \int_{\mathcal{S}} p_0(s_0) \cdot V^\pi(s_0) \cdot ds_0 \\ &= \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \pi(s_0,a_0;\theta) \cdot Q^\pi(s_0,a_0) \cdot da_0 \cdot ds_0 \end{aligned}$$

taking the gradient gives us

$$\begin{aligned} \nabla_\theta J(\theta) &= \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \nabla_\theta \pi(s_0,a_0;\theta) \cdot Q^\pi(s_0,a_0) \cdot da_0 \cdot ds_0 \\ &+ \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \pi(s_0,a_0;\theta) \cdot \nabla_\theta Q^\pi(s_0,a_0) \cdot da_0 \cdot ds_0 \end{aligned}$$

Now, noting that $Q^\pi(s_0,a_0) = \mathcal{R}_{s_0}^{a_0} + \int_{\mathcal{S}} \gamma \cdot \mathcal{P}_{s_0,s_1} \cdot V^\pi(s_1) \cdot ds_1$

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \nabla_\theta \pi(s_0, a_0; \theta) \cdot Q^\pi(s_0, a_0) \cdot da_0 \cdot ds_0 \\
&+ \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \pi(s_0, a_0; \theta) \cdot \nabla_\theta (\mathcal{R}_{s_0}^{a_0} + \int_{\mathcal{S}} \gamma \cdot \mathcal{P}_{s_0, s_1} \cdot V^\pi(s_1) \cdot ds_1) \cdot da_0 \cdot ds_0 \\
&= \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \nabla_\theta \pi(s_0, a_0; \theta) \cdot Q^\pi(s_0, a_0) \cdot da_0 \cdot ds_0 \\
&+ \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \pi(s_0, a_0; \theta) \cdot \nabla_\theta (\int_{\mathcal{S}} \gamma \cdot \mathcal{P}_{s_0, s_1} \cdot V^\pi(s_1) \cdot ds_1) \cdot da_0 \cdot ds_0 \\
&= \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \nabla_\theta \pi(s_0, a_0; \theta) \cdot Q^\pi(s_0, a_0) \cdot da_0 \cdot ds_0 \\
&+ \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \pi(s_0, a_0; \theta) \cdot (\int_{\mathcal{S}} \gamma \cdot \mathcal{P}_{s_0, s_1} \cdot \nabla_\theta V^\pi(s_1) \cdot ds_1) \cdot da_0 \cdot ds_0 \\
&= \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \nabla_\theta \pi(s_0, a_0; \theta) \cdot Q^\pi(s_0, a_0) \cdot da_0 \cdot ds_0 \\
&+ \int_{\mathcal{S}} \cdot (\int_{\mathcal{S}} \gamma p_0(s_0) \cdot \int_{\mathcal{A}} \pi(s_0, a_0; \theta) \cdot \mathcal{P}_{s_0, s_1} \cdot da_0 \cdot ds_0) \nabla_\theta V^\pi(s_1) \cdot \cdot ds_1
\end{aligned}$$

We note that $\int_{\mathcal{A}} \pi(s_0, a_0; \theta) \mathcal{P}_{s_0, s_1}^{a_0} \cdot da_0 = p(s_0 \to s_1, 1, \pi)$ so

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \nabla_\theta \pi(s_0, a_0; \theta) \cdot Q^\pi(s_0, a_0) \cdot da_0 \cdot ds_0 \\
&+ \int_{\mathcal{S}} p_0(s_0) \cdot (\int_{\mathcal{S}} \gamma \cdot p_0(s_0) \cdot p(s_0 \to s_1, 1, \pi) \cdot ds_0) \nabla_\theta V^\pi(s_1) \cdot \cdot ds_1 \\
&= \int_{\mathcal{S}} p_0(s_0) \cdot \int_{\mathcal{A}} \nabla_\theta \pi(s_0, a_0; \theta) \cdot Q^\pi(s_0, a_0) \cdot da_0 \cdot ds_0 \\
&+ \int_{\mathcal{S}} p_0(s_0) \cdot (\int_{\mathcal{S}} \gamma \cdot p_0(s_0) \cdot p(s_0 \to s_1, 1, \pi) \cdot ds_0) \nabla_\theta (\int_{\mathcal{A}} \pi(s_1, a_1; \theta) Q^\pi(s_1, a_1) da_1) \cdot ds_1
\end{aligned}$$

Thus we can see this will telescope out to

$$\begin{aligned}
\nabla_\theta J(\theta) &= \sum_{t=0}^{\infty} \int_{\mathcal{S}} \int_{\mathcal{S}} \gamma^t \cdot p_0(s_0) \cdot p(s_0 \to s_t, t, \pi) \cdot ds_0 \int_{\mathcal{A}} \nabla_\theta \pi(s_t, a_t; \theta) \cdot Q^\pi(s_t, a_t) \cdot da_t \cdot ds_t \\
&= \int_{\mathcal{S}} \int_{\mathcal{S}} \sum_{t=0}^{\infty} \gamma^t \cdot p_0(s_0) \cdot p(s_0 \to s_t, t, \pi) \cdot ds_0 \int_{\mathcal{A}} \nabla_\theta \pi(s, a; \theta) \cdot Q^\pi(s, a) \cdot da \cdot ds \\
&= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi(s, a; \theta) \cdot Q^\pi(s, a) \cdot da \cdot ds
\end{aligned}$$

17.2 Monte Carlo Policy Gradient (REINFORCE)

We update θ by SGA using PGT, using $G_t = \sum_{k=t}^{T} \gamma^{k-1} \cdot r_k$ as an unbiased sample of $Q^\pi(s_t, a_t)$

$$\Delta\theta_t = \alpha \cdot \gamma^t \cdot \nabla_\theta \log(\pi(s_t, a_t; \theta) \cdot G_t$$

A drawback with this approach is that MCPC has high variance. Instead, we can use a critic $Q(s, a; w)$ to estimate the true value function. Actor-Critic algorithms maintain two sets of parameters

Algorithm 5 REINFORCE

```
procedure REINFORCE(MDP)
    initialize $\theta$ arbitrarily
    for each episode $\sim \pi(\cdot, \cdot; \theta)$ do
        for $t \in [0, T])$ do
            $G \leftarrow \sum_{k=t}^{T} \gamma^{k-t} \cdot r_k$
            $\theta \leftarrow \theta + \alpha \cdot \gamma^t \cdot \nabla_\theta \log(\pi(s_t, a_t; \theta) \cdot G_t$
```

- Critic updates parameters w w.r.t. to approximate Q-function for policy π
- Critic can use any of the usual algorithms

  MC policy eval, TD Learning, TD(λ), LSTD (if function is linear)
- Actor updates parameters θ in direction suggested by critic
- This is called Approximate Policy Gradient due to the Bias of the Critic

$$\nabla_\theta J(\theta) \approx \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi(s, a; \theta) \cdot Q(s, a; w) \cdot da \cdot ds$$

The general algorithm looks like

1. Generate a sufficient set of simulation paths $s_0, a_0, r_0, s_1, a_1, r_1, ...$

   s_0 sampled from $p_0(\cdot)$, a_t is sampled from $\pi(s_t, \cdot; \theta)$, s_{t+1} is sampled from transition probs and r_{t+1} from reward function
2. at each step t update w proportional to gradient of appropriate (MC or TD)-based loss function of $Q(s, a; w)$
3. sum $\gamma^t cdot \nabla_\theta \log(\pi(s_t, a_t; \theta) \cdot Q(s_t, a_t, ; w)$ over t and over paths
4. update θ using this biased estimation of $\nabla_\theta J(\theta)$
5. iterate with new simulation paths

We can reduce the variance of this by subtracting a baseline function $B(s)$ from $Q(s, a; w)$ in the Policy Gradient estimation

$$\gamma^t cdot \nabla_\theta \log(\pi(s_t, a_t; \theta) \cdot Q(s_t, a_t, ; w) \rightarrow \gamma^t cdot \nabla_\theta \log(\pi(s_t, a_t; \theta) \cdot (Q(s_t, a_t, ; w) - B(s_t))$$

Not that as B is only a function on s, it will not add bias

$$\int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi(s, a; \theta) \cdot B(s) \cdot da \cdot ds = \int_{\mathcal{S}} \rho^\pi(s) B(s) \int_{\mathcal{A}} \nabla_\theta \pi(s, a; \theta) \cdot da \cdot ds$$

A good start for $B(s)$ is a state value function $V(s; v)$, then we can rewrite the policy gradient algorithm using the advantage function

$$A(s, a; w, v) = W(s, a; w) - V(s; v)$$

now we have that

$$\nabla_\theta J(\theta) \approx \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi(s, a; \theta) \cdot A(s, a; w) \cdot da \cdot ds$$

at each iteration, we just need to make sure we update both sets of parameters w, v
We note that we could use the TD eror as an approximation of the advantage function

$$\delta^\pi = r + \gamma V^\pi(s') - V^\pi(s)$$

as this is an unbiased estimate of $A^\pi(s, a)$

$$\mathbb{E}_{pi}[\delta^\pi | s, a] = \mathbb{E}_{pi}[r + \gamma V^\pi(s') - V^\pi(s)] = Q^\pi(s, a) - V^\pi(s) = A^\pi(s, a)$$

so we can rewrite the Policy Gradient in terms of $\mathbb{E}_{pi}[\delta^\pi | s, a]$

$$\nabla_\theta J(\theta) \approx \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi(s, a; \theta) \cdot \mathbb{E}_{pi}[\delta^\pi | s, a] \cdot da \cdot ds$$

In practice, this allows us to use function approximation for the TD error

$$\delta(s, r, s'; v) = r + \gamma V(s'; v) - V(s; v)$$

which only requires that we maintain one set of critic parameters v.

Algorithm 6 ACTOR-CRITIC-TD-ERROR

```
procedure ACTOR-CRITIC-TD-ERROR(MDP)
    initialize policy params $\theta \in \mathbb{R}^m$ and state VF params $v \in \mathbb{R}^n$ arbitrarily
    for each episode $\sim \pi(\cdot, \cdot; \theta)$ do
        intialize $s$, first state in episode
        while $s$ is not terminal do
            $a \sim \pi(s, \cdot; \theta)$
            take action $a$, observe $r', s'$
            $\delta \leftarrow r + \gamma V(s'; v) - V(s; v)$
            $v \leftarrow v + \alpha_v \cdot \delta \cdot \nabla_v V(s; v)$
            $\theta \leftarrow \theta\alpha_\theta \cdot P \cdot \delta \cdot \nabla_\theta \log(\pi(s, a; \theta))$
            $P \leftarrow \gamma P$
            $s \leftarrow s'$
```

It turns out that we can use eligibility traces for both actor and critic, this gives us

Algorithm 7 ACTOR-CRITIC-ELIGIBILITY-TRACES

```
procedure ACTOR-CRITIC-ELIGIBILITY-TRACES(MDP)
    initialize policy params $\theta \in \mathbb{R}^m$ and state VF params $v \in \mathbb{R}^n$ arbitrarily
    for each episode $\sim \pi(\cdot, \cdot; \theta)$ do
        intialize $s$, first state in episode
        $z_\theta, z_v \leftarrow m, n$ component eligibility trace vectors
        while $s$ is not terminal do
            $a \sim \pi(s, \cdot; \theta)$
            take action $a$, observe $r', s'$
            $\delta \leftarrow r + \gamma V(s'; v) - V(s; v)$
            $z_v \leftarrow \gamma \cdot \lambda_v + \nabla_v V(s; v)$
            $z_\theta \leftarrow \gamma \cdot \lambda_\theta + P \cdot \nabla_\theta \log(\pi(s, a; \theta))$
            $v \leftarrow v + \alpha_v \cdot \delta \cdot z_v$
            $\theta \leftarrow \theta + \alpha_\theta \cdot P \cdot \delta z_\theta$
            $P \leftarrow \gamma P$
            $s \leftarrow s'$
```

17.2.1 Overcoming Bias

We have addressed ways to reduce variance, now we tackle bias, and the core theory underpinning our method of doing so is called the *Compatible Function Approximation Theorem*
Compatible Function Approximation Theorem
If the following two conditions are satisfied:

1. Critic gradient is compatible with actor score function
   $\nabla_w Q(s,a;w) = \nabla_\theta \log(\pi(s,a;\theta))$
2. Critic parameters w minimize the follow MSE
   $\epsilon = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s,a;\theta)(Q^\pi(s,a) - Q(s,a;w))^2 \cdot da \cdot ds$

The the Policy Gradient using the critic $Q(s,a;w)$ is exact:

$$\nabla_\theta J(\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi(s,a;\theta) \cdot Q(s,a;w) \cdot da \cdot ds$$

Compatible Function Approximation Theorem Proof
For w that minimizes

$$epsilon = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s,a;\theta)(Q^\pi(s,a) - Q(s,a;w))^2 \cdot da \cdot ds$$

we must have that

$$\int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s,a;\theta)(Q^\pi(s,a) - Q(s,a;w))^2 \cdot \nabla_w Q(s,a;w) \cdot da \cdot ds = 0$$

thus as $\nabla_w Q(s,a;w) = \nabla_\theta \log(\pi(s,a;\theta)$ we have

$$\int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s,a;\theta)(Q^\pi(s,a) - Q(s,a;w)) \cdot \nabla_\theta \log(\pi(s,a;\theta) \cdot da \cdot ds = 0$$

Therefore

$$\int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s,a;\theta) Q(s,a;w) \cdot \nabla_\theta \log(\pi(s,a;\theta) \cdot da \cdot ds = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s,a;\theta) Q^\pi(s,a) \cdot \nabla_\theta \log(\pi(s,a;\theta) \cdot da \cdot ds$$

However we have that

$$\nabla_\theta J(\theta) = \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s,a;\theta) Q^\pi(s,a) \cdot \nabla_\theta \log(\pi(s,a;\theta) \cdot da \cdot ds$$

so

$$\begin{aligned}\nabla_\theta J(\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s,a;\theta) Q(s,a;w) \cdot \nabla_\theta \log(\pi(s,a;\theta) \cdot da \cdot ds \\ &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \nabla_\theta \pi(s,a;\theta) Q(s,a;w) \cdot da \cdot ds\end{aligned}$$

Now that we have demonstrated that we *can* overcome bias, we examine *how* we can do so.

A simple method for doing so is to force $Q(s,a;w)$ to be linear in it's features

$$Q(s, a; w) = \phi(s)^T w$$

In this setting, $Q(s, a; w)$ serves as an approximation of the advantage function.

$$\begin{aligned}
\int_{\mathcal{A}} \pi(s, a; \theta) \cdot Q(s, a; w) \cdot da &= \int_{\mathcal{A}} \pi(s, a; \theta) \cdot (\sum_{i=1}^{n} \frac{\partial \log(\pi(s, a; \theta))}{\partial \theta_i} \cdot w_i) \cdot da \\
&= \int_{\mathcal{A}} (\sum_{i=1}^{n} \frac{\partial \pi(s, a; \theta)}{\partial \theta_i} \cdot w_i) \cdot da \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial \theta_i} \int_{\mathcal{A}} \pi(s, a; \theta) \cdot da \cdot w_i \\
&= 0
\end{aligned}$$

If we denote $[\frac{\partial \log(\pi(s,a;\theta))}{\partial \theta_i}] = SC(s, a; \theta)$ and assume a compatible linear-approximation critic

$$\begin{aligned}
\nabla_\theta J(\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \int_{\mathcal{A}} \pi(s, a; \theta) \cdot (SC(s, a; \theta) \cdot SC(s, a; \theta)^T \cdot w) \cdot da \cdot ds \\
&= \mathbb{E}_{s \sim \rho^{pi}, a \sim \pi} SC(s, a; \theta) \cdot SC(s, a; \theta)^T \cdot w \\
&= FIM_{s \sim \rho^{pi}}(\theta) \cdot w
\end{aligned}$$

17.2.2 Natural Policy Gradient

If we consider the natural gradient from numerical optimization, $\nabla_\theta^{nat} J(\theta)$ is the direction of optimal θ movement.

$$\nabla_\theta J(\theta) = FIM_{\rho_\pi, \pi}(\theta) \cdot \nabla_\theta^{nat} J(\theta)$$

thus $\nabla_\theta^{nat} J(\theta) = w$ Which gives us a compact resultfor updating actor/critic params

1. udpate critic params w wit the critic loss gradient as

$$\gamma^t \cdot (r_t + \gamma SC(s_{t+1}, a_{t+1}, \theta) \cdot w - SC(s_t, a_t, \theta) \cdot w) \cdot SC(s_t, a_t, \theta)$$

2. update actor params θ in the direction of w

References

- http://web.stanford.edu/class/cme241