Chapter 6, 7, 12 RL Book [?] UPDATED BY: Dhawal Gupta at October 13, 2019

1 Temporal Difference Learning

TD learning is a combination of Monte Carlo ideas and Dynamic Programming (DP) ideas. Like Monte Carlo they learn from raw experience, without the model of world, and like DP they bootstrap and the estimated values. Used a policy evaluation methodology. It also uses a variation of Generalized Policy Iteration (GPI).

1.1 TD Prediction

Monte Carlo (MC) methods wait until the return of the state and can be written as $V(S_t) \leftarrow V(S_t)+\alpha[G_t - V(S_t)]$ for non stationary case. Whereas on one hand MC have to wait for the episode to end, TD(0) only need one step , TD(0) update is written as

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

TD error is defined as the error in estimation and can be written as follows

$$\delta_t \doteq R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$$

, note that δ_t is available at time $t+1$, and if the state value function are not updated during the episode we can write Monte Carlo Error in terms of TD.

$$G_t - V(S_t) = \sum_{k=t}^{T-1} \gamma^{k-t}\delta_k$$

We try to estimate the value function v_π , from experience generated by π. We try to use approximation methods with a weight/adjustable parameter such as $\mathbf{w} \in R^d$ where $d << |\mathcal{S}|$. Change in a single value of weight can change value of multiple states, and change value of one states can change the value of other state which may help in generalizing. Function approximation can't augment the state representation with memories of past observations.

1.2 Value Function Approximation

We define the updates to estimated value function that shift a state value to a given target value i.e. given state s, it shifts it value to a target value say g, represented as $s \mapsto g$. This can be interpreted as a similar problem to the **supervised learning (SL) problem** , where given an input we have to give an output (in case of real numbers, regression). So we can consider out case where out input is the state s and the output is the target value g. But one underlying issue with this is most of the SL methods assume data to be coming from a stationary distribution , whereas in the case of RL , due to bootstrapping the the underlying distribution for the target value keeps on changing.

1.3 Prediction Objective ($\overline{VE}$)

There was not requirement for an objective function for the case of tabular approximation, as the values eventually converged to the optimal values, without effecting each other independently. But in the case of function approximation, improving the accuracy of one states, will make the estimate of other states less accurate, so in this case we need to place an importance over the possible states. Specifying a state distribution $\mu(s) \geq 0, \sum_s \mu(s) = 1$, tells us how much we care about the error of a given state. In the case of on policy training in continuing tasks the distribution $\mu(s)$ is the stationary distribution under the policy π.

In case of episodic tasks this function starts to depend on the distribution of the starting states as well. We can write the probability of starting in state as $h(s)$, and we define $\eta(s)$ as the number of time steps spent on average in state s in an episode. We can write it as

$$\eta(s) = h(s) + \gamma \sum_{s'} \eta(s') \sum_a \pi(a|s')p(s|a,s') \forall s \in \mathcal{S}$$

and we can normalize the values of $\eta(s)$ to get $\mu(s)$. i.e. $\mu(s) = \frac{\eta(s)}{\sum_{s'} \eta(s')} \forall s \in \mathcal{S}$. The idea goal is to find $\mathbf{w}^*$ which satisfies the following property $\overline{VE}(\mathbf{w}^*) \leq \overline{VE}(\mathbf{w}) \forall\, \mathbf{w}$,but usually we are only able to obtain the a local optimum where $\overline{VE}(\mathbf{w}^*) \leq \overline{VE}(\mathbf{w}) \forall\, \mathbf{w}$ in neighbourhood of $\mathbf{w}^*$

1.4 Stochastic and Semi Gradient Methods

We taken an approximation $\hat{v}(s,\mathbf{w})$ which is differential wrt to $\mathbf{w}$ for all values of s. We update $\mathbf{w}$ at all discrete time steps and we denote $\mathbf{w}_t$ as the weight vector at each time step. In the ideal case we will be observing the following $s \mapsto \mathrm{v}_\pi(s)$ at each step and we will taking a gradient for :

$$\begin{aligned}\mathbf{w}_{t+1} &\doteq \mathbf{w}_t - \tfrac{1}{2}\alpha \nabla_w \left[\mathrm{v}_\pi(s) - \hat{v}(S_t, \mathbf{w}_t)\right]^2 \\ &= \mathbf{w}_t - \alpha \nabla_{w_t} \left[\mathrm{v}_\pi(s) - \hat{v}(S_t, \mathbf{w}_t)\right] \nabla_{w_t} \hat{v}(S_t, \mathbf{w}_t)\end{aligned}$$

This is know as the **Stochastic Gradient Descent** as we are doing a GD step on a single sample. where this will be the gradient in the real direction of descent but sadly we don't have access to the $\mathrm{v}_\pi(s)$. We turn to methods which give us $s \mapsto U_t$, where U_t is itself an estimate for the $\mathrm{v}_\pi(S_t)$. An example of unbiased U_t can be the monte carlo target $U_t \doteq G_t$. Whereas if we get U_t by bootstrapping, the estimate will be inherently biased to the value of $\mathbf{w}_t$ at time step t. Also ideally we should be taking the gradient of U_t as well, but we only take the gradient of the second ter m and hence this technique is know as the **Semi Gradient Descent**. Their convergence is not that robust. But they can be used in a continual and on-line fashion where updates can be done in a singe step. $U_t \doteq R + \gamma \hat{v}(S', \mathbf{w})$ We can rewrite our SGD equation in the form of $\mathbf{w}_{t+1} = \mathbf{w}_t - \alpha \nabla_{w_t} \left[U_t - \hat{v}(S_t, \mathbf{w}_t)\right] \nabla_{w_t} \hat{v}(S_t, \mathbf{w}_t)$. Where if U_t is unbiased ($\mathbb{E}[U_t|S_t = s] = \mathrm{v}_\pi(s)$), then it is guaranteed to converge to the value. We need to use some other way, if we use a unbiased estimator of the $\mathrm{v}_\pi(s)$ we can guarantee that the expected gradient will be in the correct direction , like using Monte Carlo Methods to use G_t A sample trajectory looks like

1.5 Linear Methods

Approximation methods which are a linear function of weight vector $\mathbf{w}$ are called as linear methods. The state representation may be a real valued vector $\mathbf{x}(s) \doteq (x_1(s), x_2(s), \ldots, x_d(s))^T$, and written as

$$\hat{v}(s, \mathbf{w}) \doteq \mathbf{w}^T\,\mathbf{x}(s)$$

$$\nabla_{\mathbf{w}} \hat{v}(s, \mathbf{w}) = \mathbf{x}(s)$$

, where the SGD update for the weights is now given by

$$\begin{aligned}\mathbf{w}_{t+1} &\doteq \mathbf{w}_t + \alpha\left[U_t - \hat{v}(S_t, \mathbf{w}_t)\right]\mathbf{x}(S_t) \text{ (taking TD(0) we get)}\\ &= \mathbf{w}_t + \alpha(R_{t+1}\mathbf{x}_t - \mathbf{x}_t(\mathbf{x}_t - \gamma\,\mathbf{x}_{t+1})^T\mathbf{w}_t)\\ \mathbb{E}[\mathbf{w}_{t+1} \mid \mathbf{w}_t] &= w_t + \alpha(\mathbf{b} - \mathbf{A}\mathbf{w_t})\end{aligned}$$

where $\mathbf{b} \doteq \mathbb{E}[R_{t+1}\mathbf{x}_t] \in R^d$ and $\mathbf{A} \doteq \mathbb{E}[\mathbf{x}_t(\mathbf{x}_t - \gamma\,\mathbf{x}_{t+1})^T] \in R^d \times R^d$, on convergence the system converges to $\mathbf{w}_{TD} \doteq \mathbf{A}^{-1}\mathbf{b}$. The convergence of the above system is dependent on the existence of $\mathbf{A}^{-1}$. Proof of convergence is expanded. in the appendix **??**

2 Linear Value Function Geometry

We can now start to decouple the value function approximation from the learning procedure and build a more geometric picture of the same. We can start by representing our set of state as an ordered vector i.e. $\mathcal{S} = \{s_1, s_2, \ldots s_{|\mathcal{S}|}\}$ and the value function can be represent a vector in $|\mathcal{S}|$ dimensions. i.e,$\mathrm{v}_\pi = \left[v(s_1), v(s_2), \ldots, v(s_{|\mathcal{S}|})\right]^T$, and typically out vector of weight $\mathbf{w} \in R^d$ where $d \ll |\mathcal{S}|$. Hence we can say that v_w lies in a lower dimension when compared to v_π, as we can only vary $\mathbf{w}$ in a lower dimension resulting of v_w being restricted in a very small plane.

2.1 Distance Metric

We also define a distance metric for different value function approximations $||v||_\mu^2 \doteq \sum_{s\in\mathcal{S}} \mu(s)v(s)^2$. By which we c an define the $\overline{VE}(\mathbf{w}) = \parallel v_\mathbf{w} - \mathrm{v}_\pi \parallel_\mu^2$. Defining a projection operator which projects the value function from the higher dimension on the space of $\mathrm{v}_\mathbf{w}$, written as

$$\prod v \doteq \mathrm{v}_\mathbf{w}$$
$$\text{where } \mathbf{w} = \underset{\mathbf{w}\in R^d}{\operatorname{argmin}} \parallel v - \mathrm{v}_\mathbf{w} \parallel_\mu^2$$

2.2 Bellman Error $\overline{BE}(w)$

Using the bellman error to get an estimate how far our approximation $\mathrm{v}_\mathbf{w}$ is from the actual value function by doing a bootstrap on $\mathrm{v}_\mathbf{w}$. defined as

$$\begin{aligned}\bar{\delta}_\mathbf{w}(s) &\doteq \left(\sum_a \pi(a|s) \sum_{s',r} p(s',t|s,a)[r + \gamma\,\mathrm{v}_\mathbf{w}(s')]\right) - \mathrm{v}_\mathbf{w}(s)\\ &= \mathbb{E}_\pi\left[R_{t+1} + \gamma\,\mathrm{v}_\mathbf{w}(S_{t+1} - \mathrm{v}_\mathbf{w}(S_t)|S_t = s, A_t \sim \pi\right]\\ \overline{BE}(\mathbf{w}) &= \parallel \bar{\delta}_\mathbf{w} \parallel_\mu^2\end{aligned}$$

We can call this as the expected value of the TD(0) error. i.e. $\overline{BE} = \mathbb{E}[TDE]$ and

2.3 Bellman Operator B_π

Bellman Operator defined as $\mathrm{B}_\pi : R^{|\mathcal{S}|} \to R^{|\mathcal{S}|}$, to the appropriate value function defined as

$$(\mathrm{B}_\pi\, v)(s) \doteq \sum_a \pi(a|s) \sum_{s',r} p(s',t|s,a)[r + \gamma v(s')]$$
$$\bar{\delta}_\mathbf{w} = \mathrm{B}_\pi\,\mathrm{v}_\mathbf{w} - \mathrm{v}_\mathbf{w} \text{ (can be used to define)}$$

having a stationary point at v_π i.e. $\mathrm{v}_\pi = \mathrm{B}_\pi\,\mathrm{v}_\pi$.

2.4 Projected Bellman Error $\overline{\mathrm{PBE}}(\mathbf{w})$

Normally the operatorB_π project the vector v, outside the span of the $\mathrm{v}_\mathbf{w}$. Hence we use a projection back to the space of $\mathrm{v}_\mathbf{w}$ to be able to make a possible update to new values of $\mathbf{w}$. THe projected bellman error vector is given as $\Pi\bar{\delta}_{\mathrm{v}_\mathbf{w}}$. The **Projected Bellman Error (PBE)** where

$$\overline{PBE}(\mathbf{w}) = \parallel \Pi\bar{\delta}_\mathbf{w} \parallel_\mu^2$$

All these error normally don't converge to the same value $min\,\overline{\mathrm{VE}}$ gives us $\Pi\,\mathrm{v}_\pi$, B_π operator without projection back will converge to v_π, whereas $\overline{\mathrm{BE}}$ and $\overline{\mathrm{PBE}}$ are often different, where the projection operator is defined as $\Pi \doteq \mathbf{X}(\mathbf{X}^T\,\mathbf{D}\,\mathbf{X})^{-1}\,\mathbf{X}^T\,\mathbf{D}$.

3 Policy Gradients Methods

In these set of approached we directly try to optimize the policy with/without the use of value functions.

$$\theta_{t+1} = \theta_t + \alpha\widehat{\nabla_{\theta_t} J(\theta)}$$

Is the general update rule of this where $J(\theta)$ is a performance measure.

3.1 Policy Gradient Theorem

We have to treat the case of policy gradients differently for the episodic task and the continual task. Considering the episodic task, we define the performance metric to be the value function under policy π_θ in the starting state i.e.

$$J(\theta) \doteq \mathrm{v}_{\pi_\theta}(s_0)$$

. **Challenges** : with function approximation it may be challenging to ensure policy improvement with the change in policy parameters, because change in parameters often results in the change in distribution of action in given state, and the change in policy also causes a change in the distributions of different states, hence we have to account for two gradients according to our instincts , which should balance and give us a net improvement in our performance metric. The effect of policy change on the distribution of the states is particularly unknown without the environment dynamics, that is where the Policy Gradient Theorem comes to our aid, which proves that the

$$\nabla J(\theta) \propto \sum_s \mu(s) \sum_a \mathrm{q}_\pi(s,a)\nabla_\pi(a|s,\theta)$$

Proof of this can be submitted separately. which gives us that the improvement in performance metric is not actually dependent on the gradient of the state distribution on the parameters.

3.2 REINFORCE

REINFORCE is a Monte Carlo Policy gradient approach , where we sample a complete episode and then take our gradients on that , hence REINFORCE is limited to episodic tasks then. The SGD update for gradient descent is given by

$$\theta_{t+1} \doteq \theta_t + \alpha \sum_a \hat{q}(S_t, a, \mathbf{w})\nabla\pi(a|S_t, \theta)$$

where $\hat{q}$ is an approximation of q_π. Going to the REINFORCE update we get

$$\begin{aligned}\nabla J(\theta) &= \mathbb{E}\left[\sum_a \pi(a|s,\theta)\,\mathrm{q}_\pi(S_t,a)\frac{\nabla\pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}\right]\\ &= \mathbb{E}\left[\mathrm{q}_\pi(S_t,A_t)\frac{\nabla\pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}\right]\\ &= \mathbb{E}\left[G_t\frac{\nabla\pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}\right]\end{aligned}$$

hence we get the following update rule for theta

$$\theta_{t+1} \doteq \theta_t + \alpha G_t \frac{\nabla\pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}$$

This update is great because the only parameter dependent term we have over here is the policy. Also intuitively what this update equation tells us that we change the parameters to increase the probability of action give, and that is scaled by the expected reward after that action as well , scaled down by the current probability of that action to smoothen the bias towards highly probable actions. **REINFORCE** with baseline, we can add a baseline term dependent on the current state which can help improve with the variance of the monte carlo estimation, subtracting a baseline doesnt change the expected gradient direction as $\sum_a b(s)\nabla\pi(a|s,\theta) = b(s)\nabla\sum_a \pi(a|s,\theta) = b(s)\nabla 1 = 0$ and we can write the modified update rule as

$$\theta_{t+1} \doteq \theta_t + \alpha(G_t - b(S_t))\frac{\nabla\pi(a|S_t,\theta)}{\pi(a|S_t,\theta)}$$

. The basic idea is that the baseline will be high to when all actions in a state have a high value to differentiate between good and bad actions similarly, for a state with all actions having a low value the baseline will also have a low value.

3.3 Actor - Critic Methods

When we do bootstrapping on our approximated value function (unlike baseline REINFORCE,which is unbiased) this is biased , it is called as actor critic. The update is given by

$$\theta_{t+1} \doteq \theta_t + \alpha\delta_t \frac{\nabla\pi(A_t|S_t,\theta_t)}{\pi(A_t|S_t,\theta_t)}$$

where $\delta = R_{t+1} + \gamma\hat{v}(S_{t+1},\mathbf{w}) - \hat{v}(S_t,\mathbf{w})$ and we use semi gradient methods

3.4 PG for Continual Task

The PG theorem holds for the case of continual task , we just have to modify performance metric which we put as the expected value of reward i.e. $J(\theta) = r(\pi) = \sum_s \mu(s)\sum_a \pi(a|s)\sum_{s',r} p(s',r|s,a)r$, and we define values of $\mathrm{v}_\pi(s) \doteq \mathbb{E}_\pi[G_t|S_t = s]$ wrt to differential return $G_t \doteq R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \ldots,$

3.5 Policy Parametrization for Continuous Actions

Using continues probability distribution like Gaussian distributions to define the probability over the action space which can we parametrized by say θ. example of using a Gaussian

$$\pi(a|s,\theta) \doteq \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} exp(-\frac{(a-\mu(s,\theta))^2}{2\sigma(s,\theta)^2})$$

where the mean and sigma can be parametrized as linear function over $x(s)$ as $\mu(s,\theta) = \theta_\mu^T\mathbf{x}_\mu(s)$ and $\sigma(a,\theta) = exp(\theta_\sigma^T\mathbf{x}_\sigma(s))$

I have provided a kinda of expanded version for policy gradient theorem in the second document.

4 Appendix

This section will contain proof and some extra results that might be helpful