CME241 Notes

Saakaar Bhatnagar

Value Function Approximation

1 Necessity and Methods

- When the state space is very large or continuous
- Approximate the value function using a surrogate model
- Is useful in doing tractable calculations for value function estimation
- These estimates can be then applied to prediction or control problems
- Depending on how the target is defined, is coupled with existing value function calculation algorithms like Monte Carlo, TD(0) and TD(λ)
- Usually train a linear or non-linear system's parameters to estimate the V.F (V or Q)

2 Incremental Prediction and Control

If we start out by assuming we have an oracle that tells us the value of v_π, we want to build a function approximator for the value function. We do this by minimizing:

$$J(w) = E_\pi(v(s)_\pi - \hat{v}(s,w))^2$$

For Stochastic gradient descent, taking only one example, and obtaining the gradient of the above:

$$\triangle \boldsymbol{w} = \alpha(v_\pi(s) - \hat{v}(s,w)) \bigtriangledown_{\boldsymbol{w}} \boldsymbol{\hat{v}(s,w)}$$

For a linear approximation method, the above equation is simplified to

$$\triangle \boldsymbol{w} = \alpha(v_\pi(s) - \boldsymbol{x^T w})\boldsymbol{x}$$

Now in the absence of the oracle, we have to approximate the true value function using M.C, TD(0) or TD-Lambda methods

$$\text{M.C target}=G_t$$

$$\text{TD(0) target}=R_{t+1} + \gamma \hat{V}(s_{t+1})$$

$$\text{TD}(\lambda)\text{ target}=G_t^\lambda$$

The above targets can be calculated either online, or based on stored data offline.

- Doing the value function inference online, we would actively gather data from the surroundings and constantly keep updating the weights, and consequently the value function approximator.
- Or we could have some training data beforehand and just train the approximator.
- The choice of the target depends on us.

When we are doing prediction problems (or solving for the state value function), the features $\boldsymbol{x}$ are functions of state only. When we are doing control problems, we prefer to learn the state-action value function, because we can map the policy directly from it, and now the features are a function of state and action.
Most of the mathematics remains the same as in the prediction problem. For eg, linear SARSA update equation can be written as:

$$\triangle \boldsymbol{w} = \alpha(R_{t+1} + \gamma \hat{Q}(s_{t+1}, A_{t+1}, \boldsymbol{w}) - \hat{Q}(s_t, A_t, \boldsymbol{w})) \bigtriangledown_{\boldsymbol{w}} \boldsymbol{\hat{Q}(s_t, A_t, w)}$$

where $\hat{Q}(s,a) = \boldsymbol{x(s,a)^T w}$

In the above equation, selection of A_{t+1}plays a big role. If we have a stack of sequential data of the form ¡$S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}$¿, we could use the action defined in that(hence learning some policy). Otherwise we would select it according to some other policy.

3 Batch Methods

- Because the calculated gradients in the above section are semi approximations, the above algorithms tend to diverge in certain use cases
- Moreover S.G.D is inefficient
- We use Batch gradient descent by modifying the definition of the loss function
- If we have the training data stored beforehand, this is very useful as an offline method
- It also allows for behavioural/off policy learning

The new loss is defined as, for a dataset with T time steps of data:

$$LS(w) = \sum_{t=1}^{T} (v_t^{\pi} - \hat{v}(s_t, w))^2$$

- If we sample from D and update our weights, it becomes equivalent to S.G.D, but we would be using the same experience multiple times until it converges.
- A very efficient way to obtain converged weights for a linear approximation, is by matrix inversion(least squares method)

We know that at the optimal weights,

$$E(\triangle \boldsymbol{w}) = 0$$

For a linear approximation,

$$\triangle \boldsymbol{w} = \sum_{t=1}^{T} \alpha(v_t^{\pi} - \boldsymbol{x^T w})\boldsymbol{x}$$

Therefore,

$$\sum_{t=1}^{T} \boldsymbol{x} v_t^{\pi} = \sum_{t=1}^{T} \boldsymbol{x x^T w}$$

or

$$\boldsymbol{w} = (\sum_{t=1}^{T} \boldsymbol{x x^T})^{-1} \sum_{t=1}^{T} \boldsymbol{x} v_t^{pi}$$

Now we still need to replace v_{π}^{t} by a target. We can derive some as follows:

1) The dervation of LSMC is trivial, replace v_t^{π} by G_t

2) LSTD(0): Start with

$$\sum_{t=1}^{T} \boldsymbol{x x^T w} = \sum_{t=1}^{T} \boldsymbol{x}(R_{t+1} + \gamma \boldsymbol{x_{t+1}^T w})$$

$$\sum_{t=1}^{T} \boldsymbol{(x_t(x_t^T - \gamma x_{t+1}^T)w)} = \sum_{t=1}^{T} \boldsymbol{x_t R_{t+1}}$$

get $\boldsymbol{w}$ from above by inverting LHS

3) LSTD(Lambda): at optima,

$$\sum_{t=1}^{T} \alpha \delta_t E_t = 0$$

where δ_t is the one step error. Therefore,

E would have to be calculated separately for every time step. Put the value of delta in above,

$$\sum R_{t+1} \boldsymbol{E_t} + \boldsymbol{(\gamma x_{t+1}^T w - x_t^T w) E_t} = 0$$

Rearranging, we get the optimal weights given in the slides for TD(lambda). One drawback here is that if we are given raw sequential data, we would need to calculate the elibility traces for the same.

3.1 Least Squares Control

- For the prediction step, use Least Squares Q Learning
- I feel here off policy works very well since we can just gather some data based on a bunch of policies, and use a greedy policy as our update policy to converge to the optimal value function

Again, for a linear approximation, for a bunch of data:

$$\triangle \boldsymbol{w} = \sum_{t+1}^{T} \alpha \delta_t \boldsymbol{x_t}$$

where

$$\delta_t = R_{t+1} + \gamma \hat{Q}(s_{t+1}, \pi(a|s_{t+1}), \boldsymbol{w}) - \hat{Q}(s_t, A_t, \boldsymbol{w})$$

Substitute in the above, we get the final expression given in the slides:

$$\boldsymbol{w} = (\sum \boldsymbol{x}(\boldsymbol{x_t} - \gamma \boldsymbol{x}(\boldsymbol{s_{t+1}}, \boldsymbol{\pi_{s_{t+1}}}))^{\boldsymbol{T}}) \sum R_{t+1}\boldsymbol{x_t}$$

LSPI is powerful because it will converge for a linear approximation of value function, while allowing quick evaluation of weights using inversion. Even though Q Learning by itself does not guarantee convergence for linear approximation, a Least squares batch method version of Q learning followed by greedy policy improvement works.