Università degli Studi di Milano, Data Science for Economics Master programme

Value Function Approximation

Prof. Alfio Ferrara

Reinforcement Learning

Features engineering

Before introducing how to adapt Policy Evaluation to VFA, let's discuss the choice of the features that compose the state vector $\mathbf{x}(s)$.

One obvious choice is to take some of the raw features that describe the state of the agent. However, since we are interested in exploring **linear function approximation** methods, we cannot take into account all the interactions between features. To overcome this problem, we can **select a subset of the features** or **introduce some new feature as a function or a combination of other features** in the state vector.

Example

Suppose to have an agent that is described by two feaures s_1 and s_2. We can create the feature vector according to different approaches:

Raw features

$$\mathbf{x}(s) = (s_1, s_2)^T \tag{1}$$

Polynomial feature construction

$$\mathbf{x} = (1, s_1, s_2, s_1 s_2) \tag{2}$$

Note that higher order polynomials support more complex functions approximation. However, the numer of features we obtain grows exponentially with the dimension of the raw feature vector. So, prior beliefs about the function we exptect to approximate or some automated feature selection method should be improved to build the feature vector.

Other possible transformations include

- **Fourier basis**: linear combination of sine and cosine functions of s
- **Coarse coding**: we split the state space in regions (by circles or squared subspaces) and we represent a state as a sequence of this regions where $s_1 = 1$ if the raw value of s_1 is within the region and $s_1 = 0$ otherwise.

Back to Linear VFA

Let's firt define the gradient of the VFA with respect to $\mathbf{w}$ that is:

$$\nabla_{\mathbf{w}} \hat{V}(s;\mathbf{w}) = \nabla_{\mathbf{w}} \left(\mathbf{x}(s)^T \mathbf{w}\right) = \mathbf{x}(s) \tag{3}$$

So, the update will be

$$\Delta \mathbf{w} = -\frac{1}{2}\alpha \left[2\left(V_\pi(s) - \hat{V}(s;\mathbf{w})\right)\right]\mathbf{x}(s) \tag{4}$$

which corresponds to the combination of *step-size*, *prediction error*, and *feature value*.

Monte Carlo VFA

Now we get back to the situation where we do not have a *true* value of V_π. However, with Monte Carlo methods we can compute the return G_t that is a sample of the true expected value $V_\pi(s_t)$.

Then, we can produce a set of state-return pairs of the form $\langle s_1, G_1\rangle, \langle s_2, G_2\rangle, \ldots, \langle s_t, G_t\rangle$ to use for supervised learning by substituting G_t to $V_\pi(s_t)$.

$$\Delta \mathbf{w} = \alpha \left[\left(G_t - \hat{V}(s_t;\mathbf{w})\right)\right]\mathbf{x}(s_t) = \alpha \left[\left(G_t - \mathbf{x}(s_t)^T \mathbf{w}\right)\right]\mathbf{x}(s_t) \tag{5}$$

Monte Carlo first visit VFA

```
**Initialize** $\mathbf{w} = \mathbf{0}$

**For** episode counter $e$ **in** $E$:

    **Sample** episode $e = \langle s_{e,1}, \pi(s_{e,1}), r_{e,1}, s_{e,2}, \ldots, s_{e,m}\rangle$

    **For** $t \in [1, \ldots, m]$:

        **If** first visit of $s_t \in e$:

            $G_t(s_t) = \sum_{i=t}^{m} \gamma^i r_{e,i}$

            **Update**: $\mathbf{w} = \mathbf{w} + \alpha\left(G_t(s_t) - \mathbf{x}(s_t)^T\mathbf{w}\right)\mathbf{x}(s_t)$
```

TD Learning VFA

Review of tabular TD learning

- We use bootstrapping and sampling to approximate V_π

- $V_\pi(s)$ is updated after each transition (s, a, r, s') as: $V_\pi(s) = V_\pi(s) + \alpha\left(r + \gamma V_\pi(s') - V_\pi(s)\right)$

Now, our approximation of the next state value is not more $r + \gamma V_\pi(s')$ but $r + \gamma V_\pi(s'; \mathbf{w})$.

As for MC, we can see this as a supervised task, where data are pairs of the form $\langle s_1, r_1 + \gamma \hat{V}_\pi(s_2, \mathbf{w}), \rangle, \langle s_2, r_2 + \gamma \hat{V}_\pi(s_3, \mathbf{w}) \rangle, \ldots$. Here, we want to find $\mathbf{w}$ that minimize

$$\mathbb{E}_\pi \left[\left(r_i + \gamma \hat{V}_\pi(s_{i+1}; \mathbf{w}) - \hat{V}_\pi(s_i; \mathbf{w}) \right)^2 \right] \tag{6}$$

Remember that in TD learning we want to find a state value such that the value of the future steps taken from a state is the same than the value of the state itself.

Now, rememeber also that in linear VFA

- $\hat{V}_\pi(s; \mathbf{w}) \to \mathbf{x}(s)^T \mathbf{w}$
- $\nabla_{\mathbf{w}} \hat{V}_\pi(s; \mathbf{w}) \to \mathbf{x}(s)$

Thus we have:

$$\Delta \mathbf{w} = \alpha \left(r + \gamma\, \mathbf{x}(s')^T \mathbf{w} - \mathbf{x}(s)^T \mathbf{w} \right) \mathbf{x}(s) \tag{7}$$

TD(0) linear VFA for policy evaluation

```
**Initialize** $\mathbf{w} = \mathbf{0}$

**For** $i$ **in** Iterations:

    **Sample** $(s_i, \pi(s_i), r_i, s_{i+1})$ given policy $\pi$

    **Update**: $\mathbf{w} = \mathbf{w} + \alpha \left( r_i + \gamma\, \mathbf{x}(s_{i+1})^T \mathbf{w} - \mathbf{x}(s_i)^T \mathbf{w} \right) \mathbf{x}(s_i)$
```

Convergence for linear VFA in policy evaluation

MDP, given a particular policy π will converge to a stationary probability distiribution $d(s)$ over states

- $\sum_s d(s) = 1$
- $d(s)$ satisfies the following property: $d(s') = \sum_s \sum_a \pi(a \mid s) p(s' \mid a, s) d(s)$