Haoming Liang
1430396

To apply reinforcement learning methods in large problems, computing the value function and optimal policy of each state could be computationally expensive, or sometimes impossible even with infinite time and data. Another issue is that for the first visit of some states in a large problem, the agent may behave abnormally due to the lack of information. We should generalize an approximation of a subset of states instead of computing values of each state, according to the results from similar states. In other words, we replace the computing of state value $v_\pi(s)$ with the addition of a weight vector $\mathbf{w}$, into $v(s, \mathbf{w})$. Each weight is applied to a subset of states, so changing one weight will change multiple state value estimates, and updating one state will affect the values of other states.

Such updates will have state values differ from the true values, and also estimating one state towards accurate measure will decrease the accuracy of other states. A state weighting is assigned to define the significance of error in each state. The error sums up square of differences between the approximation and true value, called Mean Squared Value Error: $\text{MSVE}(\mathbf{w}) = \sum_{s \in S} \mu(s)[v_\pi(s) - v(s, \mathbf{w})]^2$ The goal is to find a convergence to global optimum in which a weight vector $\mathbf{w}^*$ such that $\text{MSVE}(\mathbf{w}^*)$ is the least among all weights. This is only possible under the condition that function is simple (i.e. linear), more complex functions should alternatively find the convergence to local optimum, a weight vector $\mathbf{w}^*$ such that $\text{MSVE}(\mathbf{w}^*)$ is least among all $\mathbf{w}$ in some neighborhood of $\mathbf{w}^*$.

Stochastic-gradient methods minimizes MSVE by adjusting the weight vector after each example slightly towards reducing the error of the example. By the term "gradient" we should have the value function differentiable. A simple Monte-Carlo algorithm is as follow:

Initialize value-function weight $\mathbf{w}$
Repeat forever:
Generate an episode using π
$\mathbf{w} \leftarrow \mathbf{w} + \alpha[G_t - v(S_t, \mathbf{w})] \nabla v(S_t, \mathbf{w})$

G_t is initially U_t, which denotes the estimation update target. If the estimation is unbiased, the weight vector will converge to local optimum. Since Monte Carlo computes state values according to the return at the end of episode, Monte Carlo is indeed unbiased estimate and guaranteed to converge to local optimum, in which case we could use G_t to replace U_t.

If the value function is linear, we can have a feature vector $\mathbf{x}(s)$ with the same size as the weight vector $\mathbf{w}$. In this case the gradient is simply the feature vector, instead of computing derivative. Linear methods are simpler, and more favorable for mathematical analysis.