Objective for On-policy Prediction

Objectives

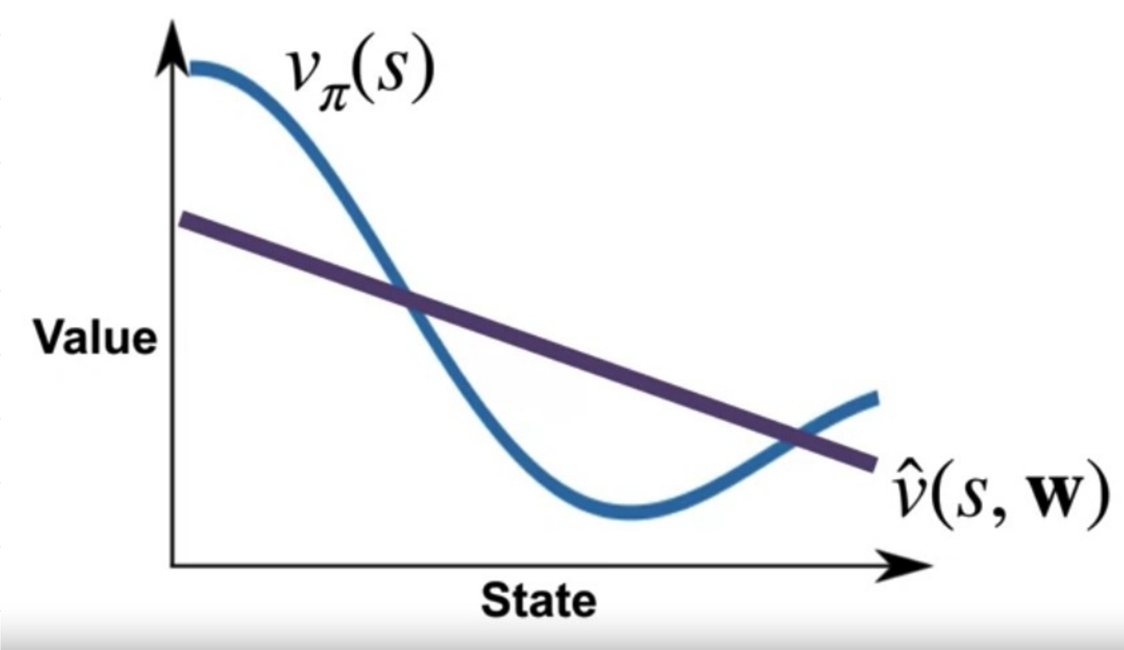
- Understand the mean-squared value error objective for policy evaluation
- Understand the gradient descent and how gradient descent apply in Reinforcement Learning
- Understand how state aggregation can be used to approximate the value function

The Value Error Objective

- In reinforcement learning, the mean-squared value error objective is a common method used for policy evaluation, particularly when estimating value functions.
- This objective is used to train models to predict the value function associated with a given policy.

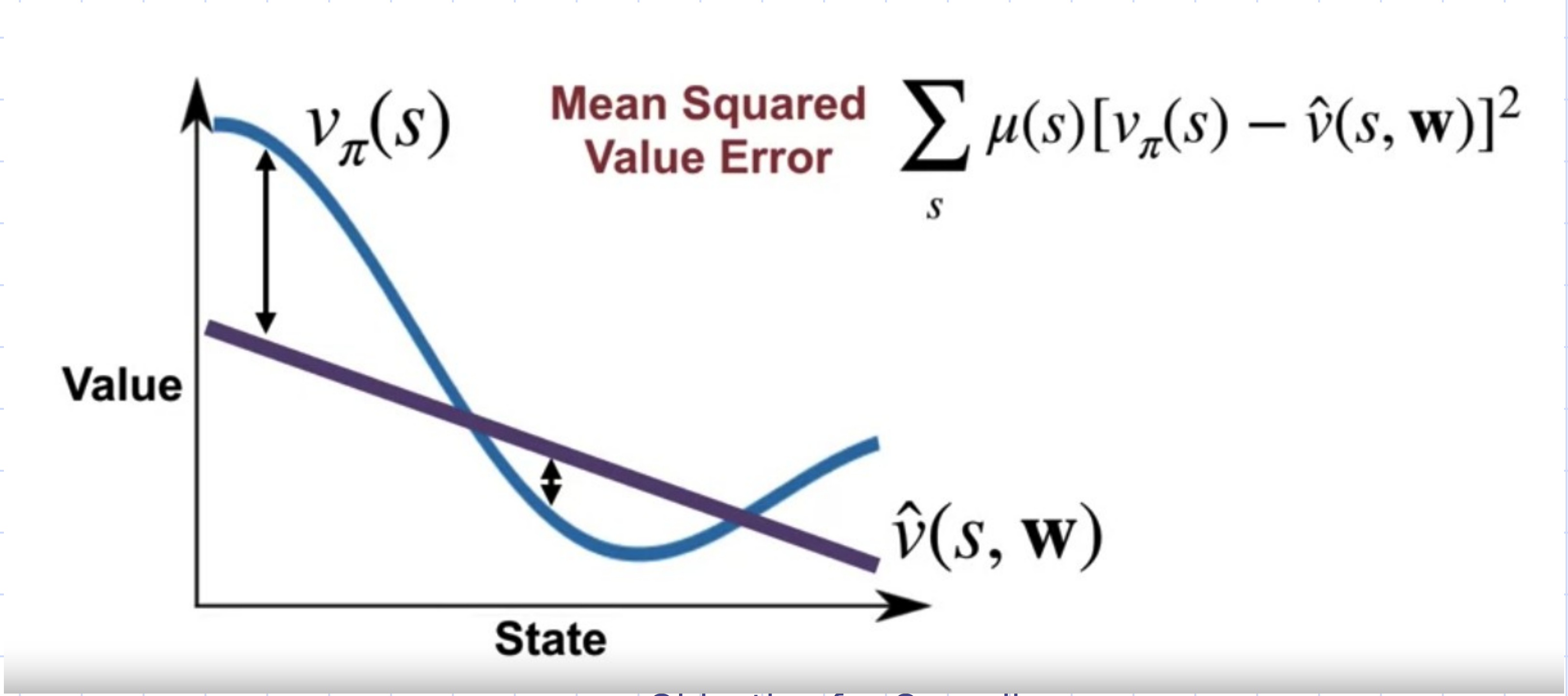
The Value Error Objective

- V is true values, V hat is estimate values
- our approximation of V Pi is not perfect, but how far off is it?



The Value Error Objective

- Define a measure of the error between the value of a state and the approximate value.



The Value Error Objective

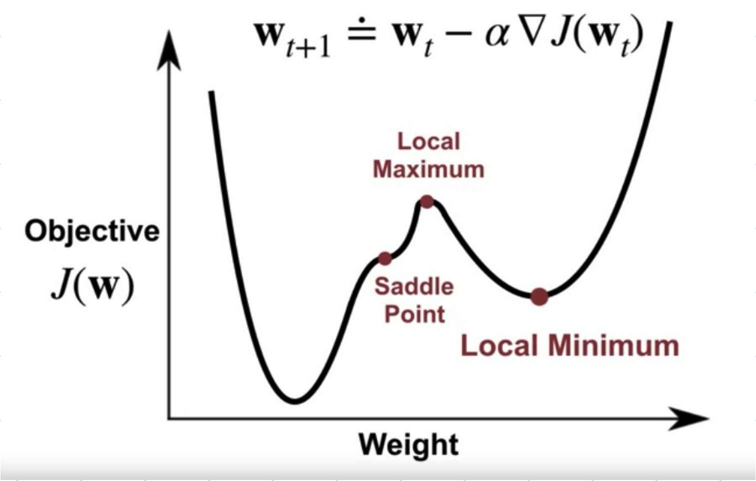
- Adapting the weights to minimize the mean squares value error objective
  - We want to adapt or weights to make the Mean Squared Value Error as low as possible.
  - We will call this objective VE bar.
  - Changing the weights in one way may increase the value error while changing them in a different way might decrease the value error.

$$\overline{VE} = \sum_{s} \mu(s)[v_\pi(s) - \hat{v}(s, \mathbf{w})]^2$$

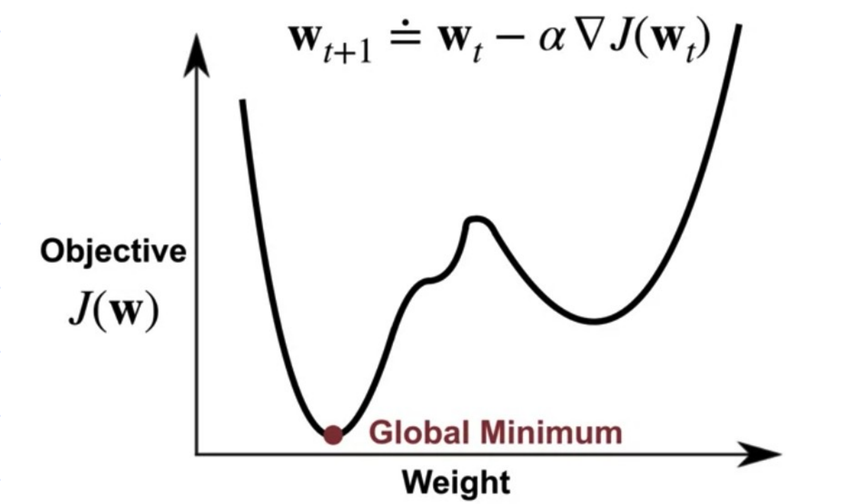
Introducing Gradient Descent

- Gradient descent is often used to optimize the policy or value function towards better performance in the environment.
- Gradient descent is a powerful optimization algorithm used in reinforcement learning to update the parameters of the learning algorithm towards better performance in the environment.
- It allows RL agents to learn from experience and adapt their behavior to achieve their goals more effectively

Introducing Gradient Descent



- ❑ Local minimum
- ❑ Local maximum
- ❑ Global minimum



Gradient Descent in RL

- Policy Gradient Methods:
  - In policy gradient methods, the agent directly learns a parameterized policy function $\pi\theta(a|s)$, where θ represents the parameters of the policy.
  - The objective is to maximize the expected return by adjusting the policy parameters.
  - The gradient of the expected return with respect to the policy parameters is computed using techniques like the policy gradient theorem

Gradient Descent in RL

- Policy Gradient Methods:
  - Gradient descent is then applied to update the policy parameters in the direction of the gradient, aiming to increase the likelihood of actions that lead to higher returns.
  - This process continues iteratively, with the agent exploring the environment, collecting experiences, computing gradients, and updating the policy parameters to improve its performance.

Introducing Gradient Descent

- Value Function Approximation:
  - The agent learns to estimate value functions, such as the state-value function V(s) or the action-value function Q(s,a), using parameterized functions.
  - The objective is to minimize the mean-squared error or the temporal difference error between predicted and observed returns.

Introducing Gradient Descent

- Value Function Approximation:
  - Gradient descent is used to update the parameters of the value function approximation towards minimizing the objective function.
  - Value function approximation can be used in combination with policy improvement techniques, such as Q-learning or SARSA, to derive optimal policies.

Introducing Gradient Descent

- Off-Policy Methods:
  - Off-policy methods: learn from experiences generated by a different behavior policy than the one being improved.
  - Gradient descent is used to update the parameters of the value function approximation towards minimizing the temporal difference error, even if the experiences were collected using a different policy.

Introducing Gradient Descent

- Actor-Critic Methods:
  - Actor-critic methods combine aspects of both policy gradient and value-based approaches.
  - The actor represents the policy function, while the critic evaluates the value function.
  - The actor's parameters are updated using policy gradients, while the critic's parameters are updated using value-based methods like temporal difference learning.
  - Gradient descent is used to update both the actor's and critic's parameters to improve the agent's policy and value estimates simultaneously.

Gradient for Policy Evaluation

- The gradient of the value function approximation indicates how to change the weights to increase the value for that state.

$$\nabla \sum_{s \in \mathcal{S}} \mu(s)[v_\pi(s) - \hat{v}(s, \mathbf{w})]^2$$

$$= \sum_{s \in \mathcal{S}} \mu(s) \nabla [v_\pi(s) - \hat{v}(s, \mathbf{w})]^2$$

$$= -\sum_{s \in \mathcal{S}} \mu(s) 2 \underbrace{[v_\pi(s) - \hat{v}(s, \mathbf{w})]} \nabla \hat{v}(s, \mathbf{w})$$

$$\hat{v}(s, \mathbf{w}) \doteq < \mathbf{w}, \mathbf{x}(s) >$$

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

Gradient for Policy Evaluation

- If the difference is positive (the true value is higher than our estimate)→ we should change the weights in the direction that increases our estim

$$\nabla \sum_{s\in\mathcal{S}} \mu(s)[v_\pi(s) - \hat{v}(s,\mathbf{w})]^2$$

$$= \sum_{s\in\mathcal{S}} \mu(s)\,\nabla[v_\pi(s) - \hat{v}(s,\mathbf{w})]^2$$

$$= -\sum_{s\in\mathcal{S}} \mu(s)2\underbrace{[v_\pi(s) - \hat{v}(s,\mathbf{w})]}_{+}\nabla\hat{v}(s,\mathbf{w})$$

$$\hat{v}(s,\mathbf{w}) \doteq <\mathbf{w},\mathbf{x}(s)>$$

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

$$\Delta\mathbf{w} \propto \sum_{s\in\mathcal{S}} \mu(s)[v_\pi(s) - \hat{v}(s,\mathbf{w})]\nabla\hat{v}(s,\mathbf{w})$$

Gradient for Policy Evaluation

- If the current error is negative, we should change the weights in the opposite direction.

$$\nabla \sum_{s\in\mathcal{S}} \mu(s)[v_\pi(s) - \hat{v}(s,\mathbf{w})]^2$$

$$= \sum_{s\in\mathcal{S}} \mu(s)\,\nabla[v_\pi(s) - \hat{v}(s,\mathbf{w})]^2$$

$$= -\sum_{s\in\mathcal{S}} \mu(s) 2\underbrace{[v_\pi(s) - \hat{v}(s,\mathbf{w})]}_{-}\,\nabla \hat{v}(s,\mathbf{w})$$

$$\hat{v}(s,\mathbf{w}) \doteq <\mathbf{w}, \mathbf{x}(s)>$$

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

$$\Delta\mathbf{w} \propto \sum_{s\in\mathcal{S}} \mu(s)[v_\pi(s) - \hat{v}(s,\mathbf{w})]\,\nabla \hat{v}(s,\mathbf{w})$$

State Aggregation

- State aggregation is a useful technique in RL for approximating the value function in large state spaces, allowing agents to generalize their experience and learn more efficiently.
- By grouping similar states together, state aggregation enables the agent to focus on important state features while reducing computational complexity.

State Aggregation

- In this table of eight states, we might choose to aggregate states together in groups of four.
- So now instead of a table of eight entries for the value function, we just have two.
- When we update the value of any state in the first group, the values of all the other states in that group is updated.

State	Value
s_1	3
s_2	3
s_3	3
s_4	3
s_5	0
s_6	0
s_7	0
s_8	0

State Aggregation

- State aggregation is another example of linear function approximation.
- There is one feature for each group of states.
- Each feature will be 1 if the current state belongs to the associated group, and 0 otherwise.
- The approximate value of a state is the weight associated with the group that state belongs to.

FPT Education

FPT UNIVERSITY

State Aggregation

State	Value
s_1	3
s_2	3
s_3	3
s_4	3
s_5	0
s_6	0
s_7	0
s_8	0

$$\mathbf{x}(s) = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad \hat{v}(s, \mathbf{w}) = w_1$$

$$\mathbf{x}(s) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad \hat{v}(s, \mathbf{w}) = w_2$$

Summary

- Understand the mean-squared value error objective for policy evaluation
- Understand the gradient descent and how gradient descent apply in Reinforcement Learning
- Understand how state aggregation can be used to approximate the value function

Q & A