FPT Education
FPT UNIVERSITY

Estimating Value Functions as Supervised Learning

Objectives

- Understand how we can use parameterized functions to approximate values
- Understand what is meant by generalization and discrimination
- Understand how supervised learning methods can be useful for handling parts of the reinforcement learning problem

Parameterized Functions

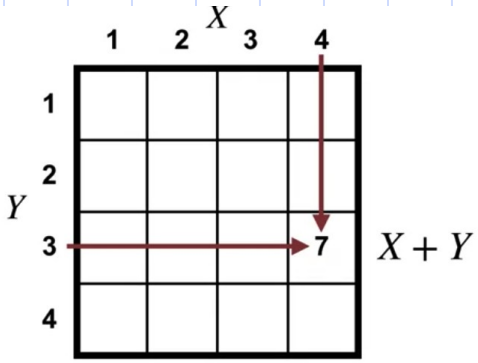
- A value function can be represented in many different ways
  - For each state, we store separate value in a table.
  - We look at values, and modify them in the table as learning progress

State	Value
s_1	-4
s_2	6
s_3	12
s_4	5
s_5	53
...	
s_{16}	-9

FPT Education
FPT UNIVERSITY

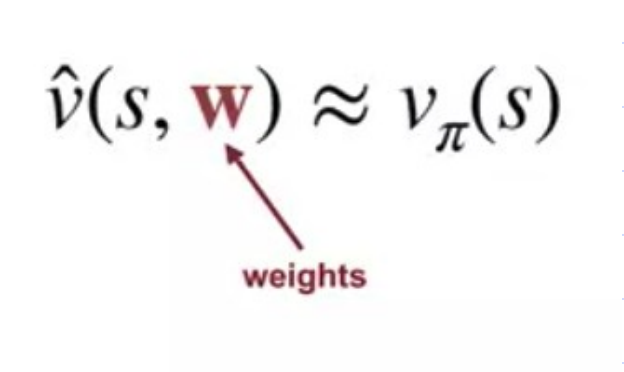
Parameterized Functions

- A value function can be represented in many different ways
  - we can use any function that takes a state, and produces a real number.
  - For example in a grid like this, our value function approximation could take the X and Y position, and add them together to produ



Parameterized Functions

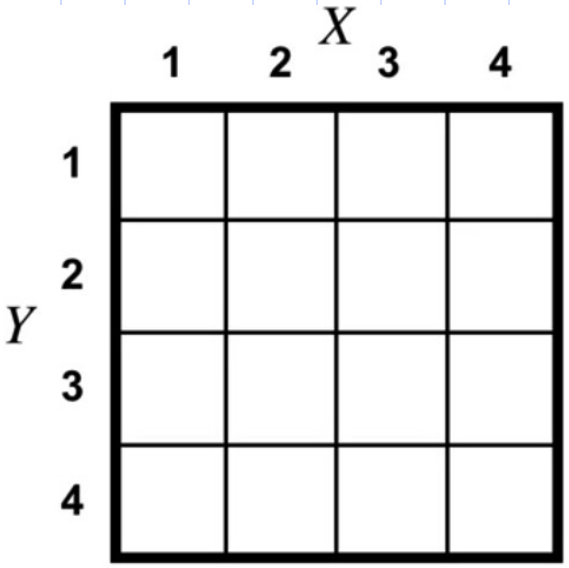
- In both ways, we can't modify the approximation.
- → We incorporate a set of real valued weights, which we can adjust to change the function.



Parameterized Functions

- Example
  - In this grid, instead of using a fixed sum of X and Y, we could use a function of the form w-1 times X, plus w-2 times Y.
  - T

$$\hat{v}(s, \mathbf{w}) \doteq w_1 X + w_2 Y$$

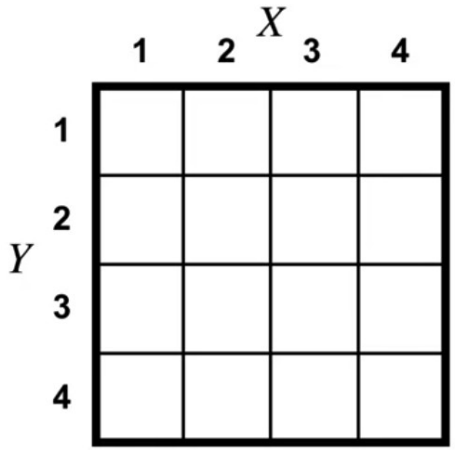


Parameterized Functions

- Example
  - They allow us to change the output the function generates. v hat is a function approximate the true value function
  - W is a vector containing all the weights that parameterize the approximation.
  - We do not have to store a whole table of values

$$\hat{v}(s, \mathbf{w}) \doteq w_1 X + w_2 Y$$

We only have to store the two weights



FPT Education
FPT UNIVERSITY

Linear Value Function Approximation

- Linear value function approximation is a technique used in reinforcement learning (RL) to approximate the value function using a linear combination of features.
- It is commonly employed in situations where the state space is large or continuous, making it impractical to represent the value function explicitly for every state.

FPT Education
FPT UNIVERSITY

Linear Value Function Approximation

- The value of each state, is computed as the sum of the weights multiplied by some fixed attributes of the state called features.
- We write that the approximate value is given by the inner product of this feature vector, and the weight vector. We will use bold X of S to denote the feature ve

Features

$$\hat{v}(s, \mathbf{w}) \doteq \sum w_i x_i(s)$$

$$= < \mathbf{w}, \mathbf{x}(s) >$$

Linear Value Function Approximation

- Limitations:
  - **Linearity Assumption**: Linear value function approximation assumes that the relationship between the features and the value function is linear.
  - **Limited Expressiveness**: Linear value function approximation may not be expressive enough to capture complex relationships between states and values.

Linear Value Function Approximation

- Limitations:
  - **Feature Engineering Dependency**: The performance of linear value function approximation heavily depends on the quality of the features chosen for representation.
  - **Curse of Dimensionality:** Linear value function approximation may suffer from the curse of dimensionality when dealing with high-dimensional state spaces.

Linear Value Function Approximation

- Limitations:
  - **Difficulty with Function Approximation**: Linear value function approximation may struggle to approximate highly nonlinear value functions accurately.
  - **Sensitivity to Noise**: Linear value function approximation may be sensitive to noisy or irrelevant features, leading to overfitting or poor generalization performance.

Generalization vs Discrimination

- While generalization refers to the ability of a system to perform well on new, unseen data, discrimination involves unjust or prejudiced distinctions between individuals or groups based on certain characteristics.
- While generalization is a desirable property in many contexts, discrimination is harmful and should be avoided.

Generalization vs Discrimination

- Definition:
  - Generalization refers to the ability of a system or model to perform accurately on new, unseen data that is similar to the data it was trained on.
  - Discrimination refers to the act of making unjust or prejudiced distinctions between individuals or groups based on certain characteristics such as race, gender, or ethnicity.

Generalization vs Discrimination

- Purpose:
  - The goal of generalization is to develop models or systems that can effectively generalize from training data to new, unseen data.
  - Discrimination has no legitimate purpose. It involves treating individuals unfairly based on arbitrary or irrelevant characteristics rather than their individual merits or actions.

Generalization vs Discrimination

- Outcome:
  - Successful generalization results in models that can accurately predict outcomes or make decisions on new, unseen data.
  - Discrimination, particularly in the negative sense, leads to unjust treatment and inequality.

Generalization vs Discrimination

- Evaluation:
  - Generalization is typically evaluated using metrics such as accuracy, precision, recall, or mean squared error on held-out test data.
  - Discrimination is evaluated based on whether individuals or groups are treated differently based on irrelevant characteristics such as race, gender, or ethnicity.

Supervised Learning vs RL

- While RL focuses on learning optimal policies through interaction with an environment
- Supervised learning methods can be valuable tools for handling various aspects of the RL problem, including modeling complex environments, approximating value functions, learning policies, and leveraging prior knowledge.

Supervised Learning vs RL

- Supervised learning can complement RL in several ways:
  - **Modeling**: Supervised learning can be used to approximate complex or unknown aspects of the environment. For example, if the dynamics of the environment are unknown, a supervised learning model can be trained to predict state transitions or rewards based on observed data, allowing the RL agent to plan and make decisions more effectively.

Supervised Learning vs RL

- Supervised learning can complement RL in several ways:
  - **Value Function Approximation**: Supervised learning methods can be used to approximate the value function in RL. By treating the value function as a target variable and using observed state-action pairs and their associated returns, supervised learning algorithms can learn to estimate the value function more efficiently than traditional RL methods.

Supervised Learning vs RL

- Supervised learning can complement RL in several ways:
  - **Policy Improvement**: Supervised learning can be used to learn a policy directly from expert demonstrations or human feedback. This can be particularly useful in settings where expert knowledge is available but collecting reward signals from the environment is difficult or expensive.

Supervised Learning vs RL

- Supervised learning can complement RL in several ways:
  - **Function Approximation**: Supervised learning techniques such as neural networks can be used to approximate complex functions involved in RL algorithms, such as Q-functions or policy functions. This allows for more flexible and scalable representations, especially in high-dimensional state and action spaces.

Supervised Learning vs RL

- Supervised learning can complement RL in several ways:
  - **Transfer Learning**: Supervised learning methods can leverage pre-trained models or features from related tasks to accelerate learning in RL. By transferring knowledge from one domain to another, RL agents can benefit from previously learned representations or policies, reducing the need for extensive exploration and training.

Supervised Learning vs RL

- Supervised learning can complement RL in several ways:
  - **Data Augmentation**: Supervised learning techniques can be used to augment the RL agent's experience by generating synthetic data or perturbing existing data. This can help improve the agent's robustness to variations in the environment and enhance its ability to generalize across different scenarios.

Supervised Learning in RL Example

- The example of using supervised learning to learn a policy for a simple grid world environment.

```python
# 3.1 Supervised Learning  in RL- HoaDNt@fe.edu.vn
import numpy as np

class GridWorld:
    def __init__(self):
        self.grid_size = (3, 3)
        self.num_actions = 4  # Up, Down, Left, Right
        self.start_state = (0, 0)
        self.goal_state = (2, 2)

    def step(self, state, action):
        # Define the dynamics of the environment
        row, col = state
        if action == 0:  # Up
            row = max(0, row - 1)
        elif action == 1:  # Down
            row = min(self.grid_size[0] - 1, row + 1)
        elif action == 2:  # Left
            col = max(0, col - 1)
        elif action == 3:  # Right
            col = min(self.grid_size[1] - 1, col + 1)
        next_state = (row, col)
        reward = 0
        if next_state == self.goal_state:
            reward = 1  # Reward of +1 upon reaching the goal state
        return next_state, reward

```

FPT Education
FPT UNIVERSITY

Supervised Learning in RL Example

```
27
28 def generate_training_data(grid_world, num_samples):
29     X = np.zeros((num_samples, 2))  # State features
30     y = np.zeros((num_samples,))     # Actions
31     for i in range(num_samples):
32         state = (np.random.randint(grid_world.grid_size[0]), np.random.randint(grid_world.grid_size[1]))
33         action = np.random.randint(grid_world.num_actions)
34         next_state, _ = grid_world.step(state, action)
35         X[i] = state
36         y[i] = action
37     return X, y
38
39 # Create a grid world environment
40 grid_world = GridWorld()
41
42 # Generate training data
43 num_samples = 1000
44 X_train, y_train = generate_training_data(grid_world, num_samples)
45
46 # Train a supervised learning model (e.g., a decision tree classifier)
47 from sklearn.tree import DecisionTreeClassifier
48 model = DecisionTreeClassifier()
49 model.fit(X_train, y_train)
50
```

Supervised Learning in RL Example

```
50
51 # Evaluate the learned policy
52 def evaluate_policy(grid_world, model):
53     total_reward = 0
54     state = grid_world.start_state
55     while state != grid_world.goal_state:
56         # Predict action based on current state
57         action = model.predict([state])[0]
58         # Take action and observe next state and reward
59         next_state, reward = grid_world.step(state, action)
60         total_reward += reward
61         state = next_state
62     return total_reward
63
64 # Evaluate the learned policy
65 total_reward = evaluate_policy(grid_world, model)
66 print("Total reward obtained by learned policy:", total_reward)
67
```

Supervised Learning in RL Example

- Show your result after run that code

Summary

- Understand how we can use parameterized functions to approximate values
- Understand what is meant by generalization and discrimination
- Understand how supervised learning methods can be useful for handling parts of the reinforcement learning problem

FPT Education
FPT UNIVERSITY

Q & A