

Lab 11

Classic Platformer Game

Grounded State

$$V(\text{Walk Forward}) = R(\text{Walk Forward}) + \gamma * P(\text{Walk Forward}) * V(\text{next state}) = 1 + 0.9 * 0.8 * V(\text{next state})$$

$$V(\text{Stay Still}) = R(\text{Stay Still}) + \gamma * P(\text{Stay Still}) * V(\text{next state}) = 0 + 0.9 * 0.2 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Airborne State

$$V(\text{Land Safely}) = R(\text{Land Safely}) + \gamma * P(\text{Land Safely}) * V(\text{next state}) = 1 + 0.9 * 0.9 * V(\text{next state})$$

$$V(\text{Fall into Gap}) = R(\text{Fall into Gap}) + \gamma * P(\text{Fall into Gap}) * V(\text{next state}) = -5 + 0.9 * 0.1 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Enemy Nearby State

$$V(\text{Attack}) = R(\text{Attack}) + \gamma * P(\text{Attack}) * V(\text{next state}) = 3 + 0.9 * 0.6 * V(\text{next state})$$

$$V(\text{Evade}) = R(\text{Evade}) + \gamma * P(\text{Evade}) * V(\text{next state}) = 1 + 0.9 * 0.4 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Low Health State

$$V(\text{Find Health Item}) = R(\text{Find Health Item}) + \gamma * P(\text{Find Health Item}) * V(\text{next state}) = 4 + 0.9 * 0.5 * V(\text{next state})$$

$$V(\text{Avoid Danger}) = R(\text{Avoid Danger}) + \gamma * P(\text{Avoid Danger}) * V(\text{next state}) = 2 + 0.9 * 0.5 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Obstacle Ahead State

$$V(\text{Jump Over}) = R(\text{Jump Over}) + \gamma * P(\text{Jump Over}) * V(\text{next state}) = 2 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Destroy}) = R(\text{Destroy}) + \gamma * P(\text{Destroy}) * V(\text{next state}) = 3 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Power-Up Available State

$$V(\text{Collect Power-Up}) = R(\text{Collect Power-Up}) + \gamma * P(\text{Collect Power-Up}) * V(\text{next state}) = 5 + 0.9 * 0.9 * V(\text{next state})$$

$$V(\text{Ignore}) = R(\text{Ignore}) + \gamma * P(\text{Ignore}) * V(\text{next state}) = 0 + 0.9 * 0.1 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Collectible Near State

$$V(\text{Collect}) = R(\text{Collect}) + \gamma * P(\text{Collect}) * V(\text{next state}) = 2 + 0.9 * 0.8 * V(\text{next state})$$

$$V(\text{Ignore}) = R(\text{Ignore}) + \gamma * P(\text{Ignore}) * V(\text{next state}) = 0 + 0.9 * 0.2 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Boss Fight State

$$V(\text{Attack Boss}) = R(\text{Attack Boss}) + \gamma * P(\text{Attack Boss}) * V(\text{next state}) = 10 + 0.9 * 0.5 * V(\text{next state})$$

$$V(\text{Dodge Attacks}) = R(\text{Dodge Attacks}) + \gamma * P(\text{Dodge Attacks}) * V(\text{next state}) = 3 + 0.9 * 0.5 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Underwater State

$$V(\text{Swim Carefully}) = R(\text{Swim Carefully}) + \gamma * P(\text{Swim Carefully}) * V(\text{next state}) = 2 + 0.9 * 0.8 * V(\text{next state})$$

$$V(\text{Rush Through}) = R(\text{Rush Through}) + \gamma * P(\text{Rush Through}) * V(\text{next state}) = -3 + 0.9 * 0.2 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Level Completion State

$$V(\text{Reach Goal}) = R(\text{Reach Goal}) + \gamma * P(\text{Reach Goal}) * V(\text{next state}) = 10 + 0.9 * 1.0 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Detailed Step-by-Step Calculations

To derive the final Q-table and determine the agent's direction in the 'Classic Platformer Game' environment, we employ the Q-learning algorithm, which iteratively updates the Q-values using the Bellman equation. The Q-learning update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (R(s_{t+1}) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

In this formula:

- $Q(s, a)$ represents the Q-value for being in state s and taking action a .
- s_t is the current state.
- a_t is the current action.
- $R(s_{t+1})$ is the reward for transitioning to state s_{t+1} .
- $\max_a Q(s_{t+1}, a)$ is the highest Q-value for the next state s_{t+1} across all possible actions.
- α is the learning rate (e.g., 0.1).
- γ is the discount factor (e.g., 0.9).

As an example, if the agent in the 'Classic Platformer Game' is in the 'Grounded' state and chooses to 'Walk Forward' (reward of +1, probability of 80%), the Q-value update with $\alpha = 0.1$, $\gamma = 0.9$, and an assumed maximum Q-value of 2 for the next state would be:

$$Q(\text{Grounded}, \text{Walk Forward}) \leftarrow 0 + 0.1 (1 + 0.9 \times 2 - 0) \approx 0.28$$

This process iterates for each action in each state, refining the Q-table until it converges. The final Q-table represents the learned policy, guiding the agent to select actions that maximize cumulative rewards. The agent uses this Q-table to determine the optimal direction (action) in each state, selecting the action with the highest Q-value.

Autonomous Car Navigation

Green Light State

$$V(\text{Accelerate}) = R(\text{Accelerate}) + \gamma * P(\text{Accelerate}) * V(\text{next state}) = 1 + 0.9 * 0.9 * V(\text{next state})$$

$$V(\text{Maintain Speed}) = R(\text{Maintain Speed}) + \gamma * P(\text{Maintain Speed}) * V(\text{next state}) = 0.5 + 0.9 * 0.1 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Red Light State

$$V(\text{Stop}) = R(\text{Stop}) + \gamma * P(\text{Stop}) * V(\text{next state}) = 1 + 0.9 * 1.0 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Pedestrian Crossing State

$$V(\text{Stop}) = R(\text{Stop}) + \gamma * P(\text{Stop}) * V(\text{next state}) = 2 + 0.9 * 1.0 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Highway Driving State

$$V(\text{Maintain Speed Limit}) = R(\text{Maintain Speed Limit}) + \gamma * P(\text{Maintain Speed Limit}) * V(\text{next state}) = 1 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Change Lane}) = R(\text{Change Lane}) + \gamma * P(\text{Change Lane}) * V(\text{next state}) = 0.5 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Traffic Jam State

$$V(\text{Wait}) = R(\text{Wait}) + \gamma * P(\text{Wait}) * V(\text{next state}) = 0.5 + 0.9 * 0.8 * V(\text{next state})$$

$$V(\text{Reroute}) = R(\text{Reroute}) + \gamma * P(\text{Reroute}) * V(\text{next state}) = 1 + 0.9 * 0.2 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Parking State

$$V(\text{Park in Lot}) = R(\text{Park in Lot}) + \gamma * P(\text{Park in Lot}) * V(\text{next state}) = 2 + 0.9 * 0.5 * V(\text{next state})$$

$$V(\text{Park on Street}) = R(\text{Park on Street}) + \gamma * P(\text{Park on Street}) * V(\text{next state}) = 1 + 0.9 * 0.5 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Emergency Vehicle State

$$V(\text{Yield}) = R(\text{Yield}) + \gamma * P(\text{Yield}) * V(\text{next state}) = 2 + 0.9 * 1.0 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Roadworks State

$$V(\text{Slow Down}) = R(\text{Slow Down}) + \gamma * P(\text{Slow Down}) * V(\text{next state}) = 1 + 0.9 * 1.0 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Sharp Turn State

$$V(\text{Slow Down}) = R(\text{Slow Down}) + \gamma * P(\text{Slow Down}) * V(\text{next state}) = 1 + 0.9 * 0.9 * V(\text{next state})$$

$$V(\text{Maintain Speed}) = R(\text{Maintain Speed}) + \gamma * P(\text{Maintain Speed}) * V(\text{next state}) = -2 + 0.9 * 0.1 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Intersection State

$$V(\text{Go Straight}) = R(\text{Go Straight}) + \gamma * P(\text{Go Straight}) * V(\text{next state}) = 0.5 + 0.9 * 0.4 * V(\text{next state})$$

$$V(\text{Turn Left}) = R(\text{Turn Left}) + \gamma * P(\text{Turn Left}) * V(\text{next state}) = 0.5 + 0.9 * 0.3 * V(\text{next state})$$

$$V(\text{Turn Right}) = R(\text{Turn Right}) + \gamma * P(\text{Turn Right}) * V(\text{next state}) = 0.5 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Detailed Step-by-Step Calculations

For the 'Autonomous Car Navigation' environment, the Q-learning algorithm is applied similarly to iteratively update the Q-values using the Bellman equation. As mentioned previously, the update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (R(s_{t+1}) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

As an illustrative example, if the autonomous car is at a 'Green Light' and decides to 'Accelerate' (reward of +1, probability of 90%), with $\alpha = 0.1$, $\gamma = 0.9$, and assuming a maximum Q-value of 2 for the next state, the Q-value update would be:

$$Q(\text{Green Light}, \text{Accelerate}) \leftarrow 0 + 0.1 (1 + 0.9 \times 2 - 0) \approx 0.28$$

As mentioned previously, this iterative process refines the Q-table until convergence. The final Q-table encapsulates the learned policy, guiding the autonomous car to make decisions that maximize cumulative rewards. The car consults this Q-table to determine the optimal action in each state, selecting the one with the highest Q-value.

Stock Market Trading: Markov Decision Process

Bull Market State

$$V(\text{Buy Stock}) = R(\text{Buy Stock}) + \gamma * P(\text{Buy Stock}) * V(\text{next state}) = 3 + 0.9 * 0.6 * V(\text{next state})$$

$$V(\text{Hold Position}) = R(\text{Hold Position}) + \gamma * P(\text{Hold Position}) * V(\text{next state}) = 1 + 0.9 * 0.4 * V(\text{next state})$$

The agent selects the action with the higher expected reward, taking into account variable probabilities when applicable.

Bear Market State

$$V(\text{Sell Stock}) = R(\text{Sell Stock}) + \gamma * P(\text{Sell Stock}) * V(\text{next state}) = 2 + 0.9 * 0.5 * V(\text{next state})$$

$$V(\text{Short Sell}) = R(\text{Short Sell}) + \gamma * P(\text{Short Sell}) * V(\text{next state}) = 3 + 0.9 * 0.5 * V(\text{next state})$$

The agent selects the action with the higher expected reward, taking into account variable probabilities when applicable.

Sideways Market State

$$V(\text{Hold Position}) = R(\text{Hold Position}) + \gamma * P(\text{Hold Position}) * V(\text{next state}) = 0.5 + 0.9 * 1.0 * V(\text{next state})$$

The agent selects the action with the higher expected reward, taking into account variable probabilities when applicable.

High Volatility State

$$V(\text{Trade Derivatives}) = R(\text{Trade Derivatives}) + \gamma * P(\text{Trade Derivatives}) * V(\text{next state}) = 4 + 0.9 * 0.5 * V(\text{next state})$$

$$V(\text{Hold Position}) = R(\text{Hold Position}) + \gamma * P(\text{Hold Position}) * V(\text{next state}) = -1 + 0.9 * 0.5 * V(\text{next state})$$

The agent selects the action with the higher expected reward, taking into account variable probabilities when applicable.

Low Volatility State

$$V(\text{Sell Options}) = R(\text{Sell Options}) + \gamma * P(\text{Sell Options}) * V(\text{next state}) = 2 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Hold Position}) = R(\text{Hold Position}) + \gamma * P(\text{Hold Position}) * V(\text{next state}) = 0.5 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward, taking into account variable probabilities when applicable.

Overbought Condition State

$$V(\text{Sell Stock}) = R(\text{Sell Stock}) + \gamma * P(\text{Sell Stock}) * V(\text{next state}) = 3 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Hold Position}) = R(\text{Hold Position}) + \gamma * P(\text{Hold Position}) * V(\text{next state}) = -2 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward, taking into account variable probabilities when applicable.

Oversold Condition State

$$V(\text{Buy Stock}) = R(\text{Buy Stock}) + \gamma * P(\text{Buy Stock}) * V(\text{next state}) = 3 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Hold Position}) = R(\text{Hold Position}) + \gamma * P(\text{Hold Position}) * V(\text{next state}) = -2 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward, taking into account variable probabilities when applicable.

Earnings Report State

$V(\text{Buy Stock}) = R(\text{Buy Stock}) + \gamma * P(\text{Buy Stock}) * V(\text{next state})$ = Variable, as it depends on specific conditions and data.

$V(\text{Sell Stock}) = R(\text{Sell Stock}) + \gamma * P(\text{Sell Stock}) * V(\text{next state})$ = Variable, as it depends on specific conditions and data.

The agent selects the action with the higher expected reward, taking into account variable probabilities when applicable.

Economic News State

$V(\text{Adjust Portfolio}) = R(\text{Adjust Portfolio}) + \gamma * P(\text{Adjust Portfolio}) * V(\text{next state})$ = Variable, as it depends on specific conditions and data.

$V(\text{Hold Position}) = R(\text{Hold Position}) + \gamma * P(\text{Hold Position}) * V(\text{next state})$ = Variable, as it depends on specific conditions and data.

The agent selects the action with the higher expected reward, taking into account variable probabilities when applicable.

Technical Breakout State

$$V(\text{Buy Stock}) = R(\text{Buy Stock}) + \gamma * P(\text{Buy Stock}) * V(\text{next state}) = 4 + 0.9 * 0.6 * V(\text{next state})$$

$$V(\text{Sell Stock}) = R(\text{Sell Stock}) + \gamma * P(\text{Sell Stock}) * V(\text{next state}) = -2 + 0.9 * 0.4 * V(\text{next state})$$

The agent selects the action with the higher expected reward, taking into account variable probabilities when applicable.

Detailed Step-by-Step Calculations

In the 'Stock Market Trading' environment, the Q-learning algorithm's application follows the same principles previously described. The update rule remains:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (R(s_{t+1}) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

For instance, if the stock trader is in a 'Bull Market' and opts to 'Buy Stock' (reward of +3, probability of 60%), with $\alpha = 0.1$, $\gamma = 0.9$, and presuming a maximum Q-value of 5 for the succeeding state, the Q-value update would be:

$$Q(\text{Bull Market}, \text{Buy Stock}) \leftarrow 0 + 0.1 (3 + 0.9 \times 5 - 0) \approx 0.54$$

As highlighted previously, this method is repeated for each action in every state, honing the Q-table until it reaches convergence. The resultant Q-table embodies the learned policy, directing the stock trader to select actions that optimize the cumulative rewards. The trader utilizes this Q-table to ascertain the best action in each state, choosing the one with the highest Q-value.

Space Exploration

Safe Orbit State

$$V(\text{Conduct Research}) = R(\text{Conduct Research}) + \gamma * P(\text{Conduct Research}) * V(\text{next state}) = 3 + 0.9 * 0.8 * V(\text{next state})$$

$$V(\text{Restock Supplies}) = R(\text{Restock Supplies}) + \gamma * P(\text{Restock Supplies}) * V(\text{next state}) = 2 + 0.9 * 0.2 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Asteroid Belt State

$$V(\text{Navigate Cautiously}) = R(\text{Navigate Cautiously}) + \gamma * P(\text{Navigate Cautiously}) * V(\text{next state}) = 2 + 0.9 * 0.9 * V(\text{next state})$$

$$V(\text{Accelerate Through}) = R(\text{Accelerate Through}) + \gamma * P(\text{Accelerate Through}) * V(\text{next state}) = -3 + 0.9 * 0.1 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Planetary Landing State

$$V(\text{Land Safely}) = R(\text{Land Safely}) + \gamma * P(\text{Land Safely}) * V(\text{next state}) = 5 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Abort Landing}) = R(\text{Abort Landing}) + \gamma * P(\text{Abort Landing}) * V(\text{next state}) = -1 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Resource Scarcity State

$$V(\text{Conserve Resources}) = R(\text{Conserve Resources}) + \gamma * P(\text{Conserve Resources}) * V(\text{next state}) = 2 + 0.9 * 0.6 * V(\text{next state})$$

$$V(\text{Seek Resources}) = R(\text{Seek Resources}) + \gamma * P(\text{Seek Resources}) * V(\text{next state}) = 4 + 0.9 * 0.4 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Alien Encounter State

$$V(\text{Communicate}) = R(\text{Communicate}) + \gamma * P(\text{Communicate}) * V(\text{next state}) = 3 + 0.9 * 0.5 * V(\text{next state})$$

$$V(\text{Flee}) = R(\text{Flee}) + \gamma * P(\text{Flee}) * V(\text{next state}) = -2 + 0.9 * 0.5 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Space Anomaly State

$$V(\text{Investigate}) = R(\text{Investigate}) + \gamma * P(\text{Investigate}) * V(\text{next state}) = 4 + 0.9 * 0.5 * V(\text{next state})$$

$$V(\text{Avoid}) = R(\text{Avoid}) + \gamma * P(\text{Avoid}) * V(\text{next state}) = 1 + 0.9 * 0.5 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Distress Signal State

$$V(\text{Provide Assistance}) = R(\text{Provide Assistance}) + \gamma * P(\text{Provide Assistance}) * V(\text{next state}) = 5 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Ignore}) = R(\text{Ignore}) + \gamma * P(\text{Ignore}) * V(\text{next state}) = -3 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Space Battle State

$$V(\text{Engage Enemy}) = R(\text{Engage Enemy}) + \gamma * P(\text{Engage Enemy}) * V(\text{next state}) = 5 + 0.9 * 0.5 * V(\text{next state})$$

$$V(\text{Retreat}) = R(\text{Retreat}) + \gamma * P(\text{Retreat}) * V(\text{next state}) = -2 + 0.9 * 0.5 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Black Hole Proximity State

$$V(\text{Study Black Hole}) = R(\text{Study Black Hole}) + \gamma * P(\text{Study Black Hole}) * V(\text{next state}) = 10 + 0.9 * 0.4 * V(\text{next state})$$

$$V(\text{Escape}) = R(\text{Escape}) + \gamma * P(\text{Escape}) * V(\text{next state}) = 1 + 0.9 * 0.6 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Discovering a New Planet State

$$V(\text{Explore Planet}) = R(\text{Explore Planet}) + \gamma * P(\text{Explore Planet}) * V(\text{next state}) = 10 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Document Discovery}) = R(\text{Document Discovery}) + \gamma * P(\text{Document Discovery}) * V(\text{next state}) = 5 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Detailed Step-by-Step Calculations

For the 'Space Exploration' environment, we apply the Q-learning algorithm as outlined before. The update rule is consistent:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (R(s_{t+1}) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

For example, if the spacecraft is in a 'Safe Orbit' and decides to 'Conduct Research' (reward of +3, probability of 80%), with $\alpha = 0.1$, $\gamma = 0.9$, and assuming a maximum Q-value of 3 for the next state, the Q-value update would be:

$$Q(\text{Safe Orbit}, \text{Conduct Research}) \leftarrow 0 + 0.1 (3 + 0.9 \times 3 - 0) \approx 0.54$$

As reiterated previously, this iterative process refines the Q-table until convergence is achieved. The final Q-table signifies the learned policy, guiding the spacecraft to make decisions that maximize cumulative rewards. The spacecraft refers to this Q-table to determine the optimal action in each state, selecting the action with the highest Q-value.

Restaurant Management

Opening Hours State

$$V(\text{Greet Customers}) = R(\text{Greet Customers}) + \gamma * P(\text{Greet Customers}) * V(\text{next state}) = 1 + 0.9 * 1.0 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Peak Dining Time State

$$V(\text{Expedite Orders}) = R(\text{Expedite Orders}) + \gamma * P(\text{Expedite Orders}) * V(\text{next state}) = 3 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Hire Temporary Staff}) = R(\text{Hire Temporary Staff}) + \gamma * P(\text{Hire Temporary Staff}) * V(\text{next state}) = 2 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Low Customer Turnout State

$$V(\text{Offer Discounts}) = R(\text{Offer Discounts}) + \gamma * P(\text{Offer Discounts}) * V(\text{next state}) = 1 + 0.9 * 0.5 * V(\text{next state})$$

$$V(\text{Close Early}) = R(\text{Close Early}) + \gamma * P(\text{Close Early}) * V(\text{next state}) = -2 + 0.9 * 0.5 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Food Shortage State

$$V(\text{Restock Ingredients}) = R(\text{Restock Ingredients}) + \gamma * P(\text{Restock Ingredients}) * V(\text{next state}) = 2 + 0.9 * 0.9 * V(\text{next state})$$

$$V(\text{Simplify Menu}) = R(\text{Simplify Menu}) + \gamma * P(\text{Simplify Menu}) * V(\text{next state}) = 1 + 0.9 * 0.1 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Staff Shortage State

$$V(\text{Hire New Staff}) = R(\text{Hire New Staff}) + \gamma * P(\text{Hire New Staff}) * V(\text{next state}) = 3 + 0.9 * 0.6 * V(\text{next state})$$

$$V(\text{Offer Overtime}) = R(\text{Offer Overtime}) + \gamma * P(\text{Offer Overtime}) * V(\text{next state}) = 2 + 0.9 * 0.4 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Health Inspection State

$$V(\text{Maintain Cleanliness}) = R(\text{Maintain Cleanliness}) + \gamma * P(\text{Maintain Cleanliness}) * V(\text{next state}) = 5 + 0.9 * 1.0 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Customer Complaint State

$$V(\text{Offer Apology}) = R(\text{Offer Apology}) + \gamma * P(\text{Offer Apology}) * V(\text{next state}) = 2 + 0.9 * 0.8 * V(\text{next state})$$

$$V(\text{Ignore}) = R(\text{Ignore}) + \gamma * P(\text{Ignore}) * V(\text{next state}) = -5 + 0.9 * 0.2 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Equipment Malfunction State

$$V(\text{Repair Equipment}) = R(\text{Repair Equipment}) + \gamma * P(\text{Repair Equipment}) * V(\text{next state}) = 3 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Use Alternative}) = R(\text{Use Alternative}) + \gamma * P(\text{Use Alternative}) * V(\text{next state}) = 1 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

High Expenses State

$$V(\text{Reduce Costs}) = R(\text{Reduce Costs}) + \gamma * P(\text{Reduce Costs}) * V(\text{next state}) = 4 + 0.9 * 0.8 * V(\text{next state})$$

$$V(\text{Increase Prices}) = R(\text{Increase Prices}) + \gamma * P(\text{Increase Prices}) * V(\text{next state}) = -2 + 0.9 * 0.2 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Menu Update State

$$V(\text{Promote New Items}) = R(\text{Promote New Items}) + \gamma * P(\text{Promote New Items}) * V(\text{next state}) = 3 + 0.9 * 0.6 * V(\text{next state})$$

$$V(\text{Gather Customer Feedback}) = R(\text{Gather Customer Feedback}) + \gamma * P(\text{Gather Customer Feedback}) * V(\text{next state}) = 2 + 0.9 * 0.4 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Detailed Step-by-Step Calculations

In the 'Restaurant Management' environment, the Q-learning algorithm's approach is similar to that described previously. The Q-value update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (R(s_{t+1}) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

For example, if the restaurant manager is during 'Peak Dining Time' and decides to 'Expedite Orders' (reward of +3, probability of 70%), with $\alpha = 0.1$, $\gamma = 0.9$, and assuming a maximum Q-value of 5 for the next state, the Q-value update would be:

$$Q(\text{Peak Dining Time}, \text{Expedite Orders}) \leftarrow 0 + 0.1 (3 + 0.9 \times 5 - 0) \approx 0.78$$

As mentioned in previous examples, this iterative process continues until the Q-table converges to a stable set of values. The final Q-table provides the learned policy, enabling the restaurant manager to make decisions that optimize cumulative rewards. The manager uses the Q-table to determine the optimal actions in each state, based on the highest Q-values.

Disaster Management

Flood Alert State

$$V(\text{Strengthen Levees}) = R(\text{Strengthen Levees}) + \gamma * P(\text{Strengthen Levees}) * V(\text{next state}) = 4 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Evacuate Areas}) = R(\text{Evacuate Areas}) + \gamma * P(\text{Evacuate Areas}) * V(\text{next state}) = 3 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Earthquake Aftermath State

$$V(\text{Search and Rescue}) = R(\text{Search and Rescue}) + \gamma * P(\text{Search and Rescue}) * V(\text{next state}) = 5 + 0.9 * 0.8 * V(\text{next state})$$

$$V(\text{Assess Damage}) = R(\text{Assess Damage}) + \gamma * P(\text{Assess Damage}) * V(\text{next state}) = 2 + 0.9 * 0.2 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Hurricane Warning State

$$V(\text{Evacuate Areas}) = R(\text{Evacuate Areas}) + \gamma * P(\text{Evacuate Areas}) * V(\text{next state}) = 5 + 0.9 * 0.9 * V(\text{next state})$$

$$V(\text{Secure Buildings}) = R(\text{Secure Buildings}) + \gamma * P(\text{Secure Buildings}) * V(\text{next state}) = 3 + 0.9 * 0.1 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Wildfire Spread State

$$V(\text{Create Firebreaks}) = R(\text{Create Firebreaks}) + \gamma * P(\text{Create Firebreaks}) * V(\text{next state}) = 4 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Evacuate Areas}) = R(\text{Evacuate Areas}) + \gamma * P(\text{Evacuate Areas}) * V(\text{next state}) = 3 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Power Outage State

$$V(\text{Restore Power}) = R(\text{Restore Power}) + \gamma * P(\text{Restore Power}) * V(\text{next state}) = 3 + 0.9 * 0.9 * V(\text{next state})$$

$$V(\text{Provide Generators}) = R(\text{Provide Generators}) + \gamma * P(\text{Provide Generators}) * V(\text{next state}) = 2 + 0.9 * 0.1 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Evacuation Order State

$$V(\text{Organize Transport}) = R(\text{Organize Transport}) + \gamma * P(\text{Organize Transport}) * V(\text{next state}) \\ = 4 + 0.9 * 0.8 * V(\text{next state})$$

$$V(\text{Set Up Shelters}) = R(\text{Set Up Shelters}) + \gamma * P(\text{Set Up Shelters}) * V(\text{next state}) = 3 + 0.9 * 0.2 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Search and Rescue State

$$V(\text{Deploy Teams}) = R(\text{Deploy Teams}) + \gamma * P(\text{Deploy Teams}) * V(\text{next state}) = 5 + 0.9 * 1.0 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Resource Allocation State

$$V(\text{Distribute Food and Water}) = R(\text{Distribute Food and Water}) + \gamma * P(\text{Distribute Food and Water}) * V(\text{next state}) = 3 + 0.9 * 0.7 * V(\text{next state})$$

$$V(\text{Distribute Medical Supplies}) = R(\text{Distribute Medical Supplies}) + \gamma * P(\text{Distribute Medical Supplies}) * V(\text{next state}) = 4 + 0.9 * 0.3 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Infrastructure Repair State

$$V(\text{Fix Roads}) = R(\text{Fix Roads}) + \gamma * P(\text{Fix Roads}) * V(\text{next state}) = 3 + 0.9 * 0.5 * V(\text{next state})$$

$$V(\text{Restore Communication}) = R(\text{Restore Communication}) + \gamma * P(\text{Restore Communication}) * V(\text{next state}) = 4 + 0.9 * 0.5 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Public Panic State

$$V(\text{Broadcast Calm Messages}) = R(\text{Broadcast Calm Messages}) + \gamma * P(\text{Broadcast Calm Messages}) * V(\text{next state}) = 2 + 0.9 * 0.8 * V(\text{next state})$$

$$V(\text{Ignore}) = R(\text{Ignore}) + \gamma * P(\text{Ignore}) * V(\text{next state}) = -5 + 0.9 * 0.2 * V(\text{next state})$$

The agent selects the action with the higher expected reward.

Detailed Step-by-Step Calculations

In the 'Disaster Management' environment, we apply the Q-learning algorithm using the same principles described earlier. The update rule is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (R(s_{t+1}) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t))$$

As an example, if the disaster response team is facing a 'Flood Alert' and decides to 'Strengthen Levees' (reward of +4, probability of 70%), with $\alpha = 0.1$, $\gamma = 0.9$, and assuming a maximum Q-value of 5 for the next state, the Q-value update would be:

$$Q(\text{Flood Alert}, \text{Strengthen Levees}) \leftarrow 0 + 0.1 (4 + 0.9 \times 5 - 0) \approx 0.85$$

As reiterated in previous sections, this iterative process refines the Q-table until it converges. The final Q-table represents the learned policy, guiding the disaster response team to take actions that maximize cumulative rewards. The team uses this Q-table to determine the optimal action in each state, choosing the one with the highest Q-value.