Lab 11

Classic Platformer Game

## Grounded State

V(Walk Forward) = R(Walk Forward) + γ \* P(Walk Forward) \* V(next state) = 1 + 0.9 \* 0.8 \* V(next state)

V(Stay Still) = R(Stay Still) + γ \* P(Stay Still) \* V(next state) = 0 + 0.9 \* 0.2 \* V(next state)

The agent selects the action with the higher expected reward.

## Airborne State

V(Land Safely) = R(Land Safely) + γ \* P(Land Safely) \* V(next state) = 1 + 0.9 \* 0.9 \* V(next state)

V(Fall into Gap) = R(Fall into Gap) + γ \* P(Fall into Gap) \* V(next state) = -5 + 0.9 \* 0.1 \* V(next state)

The agent selects the action with the higher expected reward.

## Enemy Nearby State

V(Attack) = R(Attack) + γ \* P(Attack) \* V(next state) = 3 + 0.9 \* 0.6 \* V(next state)

V(Evade) = R(Evade) + γ \* P(Evade) \* V(next state) = 1 + 0.9 \* 0.4 \* V(next state)

The agent selects the action with the higher expected reward.

## Low Health State

V(Find Health Item) = R(Find Health Item) + γ \* P(Find Health Item) \* V(next state) = 4 + 0.9 \* 0.5 \* V(next state)

V(Avoid Danger) = R(Avoid Danger) + γ \* P(Avoid Danger) \* V(next state) = 2 + 0.9 \* 0.5 \* V(next state)

The agent selects the action with the higher expected reward.

## Obstacle Ahead State

V(Jump Over) = R(Jump Over) + γ \* P(Jump Over) \* V(next state) = 2 + 0.9 \* 0.7 \* V(next state)

V(Destroy) = R(Destroy) + γ \* P(Destroy) \* V(next state) = 3 + 0.9 \* 0.3 \* V(next state)

The agent selects the action with the higher expected reward.

## Power-Up Available State

V(Collect Power-Up) = R(Collect Power-Up) + γ \* P(Collect Power-Up) \* V(next state) = 5 + 0.9 \* 0.9 \* V(next state)

V(Ignore) = R(Ignore) + γ \* P(Ignore) \* V(next state) = 0 + 0.9 \* 0.1 \* V(next state)

The agent selects the action with the higher expected reward.

## Collectible Near State

V(Collect) = R(Collect) + γ \* P(Collect) \* V(next state) = 2 + 0.9 \* 0.8 \* V(next state)

V(Ignore) = R(Ignore) + γ \* P(Ignore) \* V(next state) = 0 + 0.9 \* 0.2 \* V(next state)

The agent selects the action with the higher expected reward.

## Boss Fight State

V(Attack Boss) = R(Attack Boss) + γ \* P(Attack Boss) \* V(next state) = 10 + 0.9 \* 0.5 \* V(next state)

V(Dodge Attacks) = R(Dodge Attacks) + γ \* P(Dodge Attacks) \* V(next state) = 3 + 0.9 \* 0.5 \* V(next state)

The agent selects the action with the higher expected reward.

## Underwater State

V(Swim Carefully) = R(Swim Carefully) + γ \* P(Swim Carefully) \* V(next state) = 2 + 0.9 \* 0.8 \* V(next state)

V(Rush Through) = R(Rush Through) + γ \* P(Rush Through) \* V(next state) = -3 + 0.9 \* 0.2 \* V(next state)

The agent selects the action with the higher expected reward.

## Level Completion State

V(Reach Goal) = R(Reach Goal) + γ \* P(Reach Goal) \* V(next state) = 10 + 0.9 \* 1.0 \* V(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) ← Q(s\_t, a\_t) + α ( R(s\_{t+1}) + γ 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 α = 0.1, γ = 0.9, and an assumed maximum Q-value of 2 for the next state would be:

*Q(Grounded, Walk Forward) ← 0 + 0.1 ( 1 + 0.9 × 2 - 0 ) ≈ 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(Accelerate) = R(Accelerate) + γ \* P(Accelerate) \* V(next state) = 1 + 0.9 \* 0.9 \* V(next state)

V(Maintain Speed) = R(Maintain Speed) + γ \* P(Maintain Speed) \* V(next state) = 0.5 + 0.9 \* 0.1 \* V(next state)

The agent selects the action with the higher expected reward.

## Red Light State

V(Stop) = R(Stop) + γ \* P(Stop) \* V(next state) = 1 + 0.9 \* 1.0 \* V(next state)

The agent selects the action with the higher expected reward.

## Pedestrian Crossing State

V(Stop) = R(Stop) + γ \* P(Stop) \* V(next state) = 2 + 0.9 \* 1.0 \* V(next state)

The agent selects the action with the higher expected reward.

## Highway Driving State

V(Maintain Speed Limit) = R(Maintain Speed Limit) + γ \* P(Maintain Speed Limit) \* V(next state) = 1 + 0.9 \* 0.7 \* V(next state)

V(Change Lane) = R(Change Lane) + γ \* P(Change Lane) \* V(next state) = 0.5 + 0.9 \* 0.3 \* V(next state)

The agent selects the action with the higher expected reward.

## Traffic Jam State

V(Wait) = R(Wait) + γ \* P(Wait) \* V(next state) = 0.5 + 0.9 \* 0.8 \* V(next state)

V(Reroute) = R(Reroute) + γ \* P(Reroute) \* V(next state) = 1 + 0.9 \* 0.2 \* V(next state)

The agent selects the action with the higher expected reward.

## Parking State

V(Park in Lot) = R(Park in Lot) + γ \* P(Park in Lot) \* V(next state) = 2 + 0.9 \* 0.5 \* V(next state)

V(Park on Street) = R(Park on Street) + γ \* P(Park on Street) \* V(next state) = 1 + 0.9 \* 0.5 \* V(next state)

The agent selects the action with the higher expected reward.

## Emergency Vehicle State

V(Yield) = R(Yield) + γ \* P(Yield) \* V(next state) = 2 + 0.9 \* 1.0 \* V(next state)

The agent selects the action with the higher expected reward.

## Roadworks State

V(Slow Down) = R(Slow Down) + γ \* P(Slow Down) \* V(next state) = 1 + 0.9 \* 1.0 \* V(next state)

The agent selects the action with the higher expected reward.

## Sharp Turn State

V(Slow Down) = R(Slow Down) + γ \* P(Slow Down) \* V(next state) = 1 + 0.9 \* 0.9 \* V(next state)

V(Maintain Speed) = R(Maintain Speed) + γ \* P(Maintain Speed) \* V(next state) = -2 + 0.9 \* 0.1 \* V(next state)

The agent selects the action with the higher expected reward.

## Intersection State

V(Go Straight) = R(Go Straight) + γ \* P(Go Straight) \* V(next state) = 0.5 + 0.9 \* 0.4 \* V(next state)

V(Turn Left) = R(Turn Left) + γ \* P(Turn Left) \* V(next state) = 0.5 + 0.9 \* 0.3 \* V(next state)

V(Turn Right) = R(Turn Right) + γ \* P(Turn Right) \* V(next state) = 0.5 + 0.9 \* 0.3 \* V(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) ← Q(s\_t, a\_t) + α ( R(s\_{t+1}) + γ 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 α = 0.1, γ = 0.9, and assuming a maximum Q-value of 2 for the next state, the Q-value update would be:

*Q(Green Light, Accelerate) ← 0 + 0.1 ( 1 + 0.9 × 2 - 0 ) ≈ 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(Buy Stock) = R(Buy Stock) + γ \* P(Buy Stock) \* V(next state) = 3 + 0.9 \* 0.6 \* V(next state)

V(Hold Position) = R(Hold Position) + γ \* P(Hold Position) \* V(next state) = 1 + 0.9 \* 0.4 \* V(next state)

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

## Bear Market State

V(Sell Stock) = R(Sell Stock) + γ \* P(Sell Stock) \* V(next state) = 2 + 0.9 \* 0.5 \* V(next state)

V(Short Sell) = R(Short Sell) + γ \* P(Short Sell) \* V(next state) = 3 + 0.9 \* 0.5 \* V(next state)

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

## Sideways Market State

V(Hold Position) = R(Hold Position) + γ \* P(Hold Position) \* V(next state) = 0.5 + 0.9 \* 1.0 \* V(next state)

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

## High Volatility State

V(Trade Derivatives) = R(Trade Derivatives) + γ \* P(Trade Derivatives) \* V(next state) = 4 + 0.9 \* 0.5 \* V(next state)

V(Hold Position) = R(Hold Position) + γ \* P(Hold Position) \* V(next state) = -1 + 0.9 \* 0.5 \* V(next state)

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

## Low Volatility State

V(Sell Options) = R(Sell Options) + γ \* P(Sell Options) \* V(next state) = 2 + 0.9 \* 0.7 \* V(next state)

V(Hold Position) = R(Hold Position) + γ \* P(Hold Position) \* V(next state) = 0.5 + 0.9 \* 0.3 \* V(next state)

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

## Overbought Condition State

V(Sell Stock) = R(Sell Stock) + γ \* P(Sell Stock) \* V(next state) = 3 + 0.9 \* 0.7 \* V(next state)

V(Hold Position) = R(Hold Position) + γ \* P(Hold Position) \* V(next state) = -2 + 0.9 \* 0.3 \* V(next state)

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

## Oversold Condition State

V(Buy Stock) = R(Buy Stock) + γ \* P(Buy Stock) \* V(next state) = 3 + 0.9 \* 0.7 \* V(next state)

V(Hold Position) = R(Hold Position) + γ \* P(Hold Position) \* V(next state) = -2 + 0.9 \* 0.3 \* V(next state)

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

## Earnings Report State

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

V(Sell Stock) = R(Sell Stock) + γ \* P(Sell Stock) \* V(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(Adjust Portfolio) = R(Adjust Portfolio) + γ \* P(Adjust Portfolio) \* V(next state) = Variable, as it depends on specific conditions and data.

V(Hold Position) = R(Hold Position) + γ \* P(Hold Position) \* V(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(Buy Stock) = R(Buy Stock) + γ \* P(Buy Stock) \* V(next state) = 4 + 0.9 \* 0.6 \* V(next state)

V(Sell Stock) = R(Sell Stock) + γ \* P(Sell Stock) \* V(next state) = -2 + 0.9 \* 0.4 \* V(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) ← Q(s\_t, a\_t) + α ( R(s\_{t+1}) + γ 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 α = 0.1, γ = 0.9, and presuming a maximum Q-value of 5 for the succeeding state, the Q-value update would be:

*Q(Bull Market, Buy Stock) ← 0 + 0.1 ( 3 + 0.9 × 5 - 0 ) ≈ 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(Conduct Research) = R(Conduct Research) + γ \* P(Conduct Research) \* V(next state) = 3 + 0.9 \* 0.8 \* V(next state)

V(Restock Supplies) = R(Restock Supplies) + γ \* P(Restock Supplies) \* V(next state) = 2 + 0.9 \* 0.2 \* V(next state)

The agent selects the action with the higher expected reward.

## Asteroid Belt State

V(Navigate Cautiously) = R(Navigate Cautiously) + γ \* P(Navigate Cautiously) \* V(next state) = 2 + 0.9 \* 0.9 \* V(next state)

V(Accelerate Through) = R(Accelerate Through) + γ \* P(Accelerate Through) \* V(next state) = -3 + 0.9 \* 0.1 \* V(next state)

The agent selects the action with the higher expected reward.

## Planetary Landing State

V(Land Safely) = R(Land Safely) + γ \* P(Land Safely) \* V(next state) = 5 + 0.9 \* 0.7 \* V(next state)

V(Abort Landing) = R(Abort Landing) + γ \* P(Abort Landing) \* V(next state) = -1 + 0.9 \* 0.3 \* V(next state)

The agent selects the action with the higher expected reward.

## Resource Scarcity State

V(Conserve Resources) = R(Conserve Resources) + γ \* P(Conserve Resources) \* V(next state) = 2 + 0.9 \* 0.6 \* V(next state)

V(Seek Resources) = R(Seek Resources) + γ \* P(Seek Resources) \* V(next state) = 4 + 0.9 \* 0.4 \* V(next state)

The agent selects the action with the higher expected reward.

## Alien Encounter State

V(Communicate) = R(Communicate) + γ \* P(Communicate) \* V(next state) = 3 + 0.9 \* 0.5 \* V(next state)

V(Flee) = R(Flee) + γ \* P(Flee) \* V(next state) = -2 + 0.9 \* 0.5 \* V(next state)

The agent selects the action with the higher expected reward.

## Space Anomaly State

V(Investigate) = R(Investigate) + γ \* P(Investigate) \* V(next state) = 4 + 0.9 \* 0.5 \* V(next state)

V(Avoid) = R(Avoid) + γ \* P(Avoid) \* V(next state) = 1 + 0.9 \* 0.5 \* V(next state)

The agent selects the action with the higher expected reward.

## Distress Signal State

V(Provide Assistance) = R(Provide Assistance) + γ \* P(Provide Assistance) \* V(next state) = 5 + 0.9 \* 0.7 \* V(next state)

V(Ignore) = R(Ignore) + γ \* P(Ignore) \* V(next state) = -3 + 0.9 \* 0.3 \* V(next state)

The agent selects the action with the higher expected reward.

## Space Battle State

V(Engage Enemy) = R(Engage Enemy) + γ \* P(Engage Enemy) \* V(next state) = 5 + 0.9 \* 0.5 \* V(next state)

V(Retreat) = R(Retreat) + γ \* P(Retreat) \* V(next state) = -2 + 0.9 \* 0.5 \* V(next state)

The agent selects the action with the higher expected reward.

## Black Hole Proximity State

V(Study Black Hole) = R(Study Black Hole) + γ \* P(Study Black Hole) \* V(next state) = 10 + 0.9 \* 0.4 \* V(next state)

V(Escape) = R(Escape) + γ \* P(Escape) \* V(next state) = 1 + 0.9 \* 0.6 \* V(next state)

The agent selects the action with the higher expected reward.

## Discovering a New Planet State

V(Explore Planet) = R(Explore Planet) + γ \* P(Explore Planet) \* V(next state) = 10 + 0.9 \* 0.7 \* V(next state)

V(Document Discovery) = R(Document Discovery) + γ \* P(Document Discovery) \* V(next state) = 5 + 0.9 \* 0.3 \* V(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) ← Q(s\_t, a\_t) + α ( R(s\_{t+1}) + γ 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 α = 0.1, γ = 0.9, and assuming a maximum Q-value of 3 for the next state, the Q-value update would be:

*Q(Safe Orbit, Conduct Research) ← 0 + 0.1 ( 3 + 0.9 × 3 - 0 ) ≈ 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(Greet Customers) = R(Greet Customers) + γ \* P(Greet Customers) \* V(next state) = 1 + 0.9 \* 1.0 \* V(next state)

The agent selects the action with the higher expected reward.

## Peak Dining Time State

V(Expedite Orders) = R(Expedite Orders) + γ \* P(Expedite Orders) \* V(next state) = 3 + 0.9 \* 0.7 \* V(next state)

V(Hire Temporary Staff) = R(Hire Temporary Staff) + γ \* P(Hire Temporary Staff) \* V(next state) = 2 + 0.9 \* 0.3 \* V(next state)

The agent selects the action with the higher expected reward.

## Low Customer Turnout State

V(Offer Discounts) = R(Offer Discounts) + γ \* P(Offer Discounts) \* V(next state) = 1 + 0.9 \* 0.5 \* V(next state)

V(Close Early) = R(Close Early) + γ \* P(Close Early) \* V(next state) = -2 + 0.9 \* 0.5 \* V(next state)

The agent selects the action with the higher expected reward.

## Food Shortage State

V(Restock Ingredients) = R(Restock Ingredients) + γ \* P(Restock Ingredients) \* V(next state) = 2 + 0.9 \* 0.9 \* V(next state)

V(Simplify Menu) = R(Simplify Menu) + γ \* P(Simplify Menu) \* V(next state) = 1 + 0.9 \* 0.1 \* V(next state)

The agent selects the action with the higher expected reward.

## Staff Shortage State

V(Hire New Staff) = R(Hire New Staff) + γ \* P(Hire New Staff) \* V(next state) = 3 + 0.9 \* 0.6 \* V(next state)

V(Offer Overtime) = R(Offer Overtime) + γ \* P(Offer Overtime) \* V(next state) = 2 + 0.9 \* 0.4 \* V(next state)

The agent selects the action with the higher expected reward.

## Health Inspection State

V(Maintain Cleanliness) = R(Maintain Cleanliness) + γ \* P(Maintain Cleanliness) \* V(next state) = 5 + 0.9 \* 1.0 \* V(next state)

The agent selects the action with the higher expected reward.

## Customer Complaint State

V(Offer Apology) = R(Offer Apology) + γ \* P(Offer Apology) \* V(next state) = 2 + 0.9 \* 0.8 \* V(next state)

V(Ignore) = R(Ignore) + γ \* P(Ignore) \* V(next state) = -5 + 0.9 \* 0.2 \* V(next state)

The agent selects the action with the higher expected reward.

## Equipment Malfunction State

V(Repair Equipment) = R(Repair Equipment) + γ \* P(Repair Equipment) \* V(next state) = 3 + 0.9 \* 0.7 \* V(next state)

V(Use Alternative) = R(Use Alternative) + γ \* P(Use Alternative) \* V(next state) = 1 + 0.9 \* 0.3 \* V(next state)

The agent selects the action with the higher expected reward.

## High Expenses State

V(Reduce Costs) = R(Reduce Costs) + γ \* P(Reduce Costs) \* V(next state) = 4 + 0.9 \* 0.8 \* V(next state)

V(Increase Prices) = R(Increase Prices) + γ \* P(Increase Prices) \* V(next state) = -2 + 0.9 \* 0.2 \* V(next state)

The agent selects the action with the higher expected reward.

## Menu Update State

V(Promote New Items) = R(Promote New Items) + γ \* P(Promote New Items) \* V(next state) = 3 + 0.9 \* 0.6 \* V(next state)

V(Gather Customer Feedback) = R(Gather Customer Feedback) + γ \* P(Gather Customer Feedback) \* V(next state) = 2 + 0.9 \* 0.4 \* V(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) ← Q(s\_t, a\_t) + α ( R(s\_{t+1}) + γ 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 α = 0.1, γ = 0.9, and assuming a maximum Q-value of 5 for the next state, the Q-value update would be:

*Q(Peak Dining Time, Expedite Orders) ← 0 + 0.1 ( 3 + 0.9 × 5 - 0 ) ≈ 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(Strengthen Levees) = R(Strengthen Levees) + γ \* P(Strengthen Levees) \* V(next state) = 4 + 0.9 \* 0.7 \* V(next state)

V(Evacuate Areas) = R(Evacuate Areas) + γ \* P(Evacuate Areas) \* V(next state) = 3 + 0.9 \* 0.3 \* V(next state)

The agent selects the action with the higher expected reward.

## Earthquake Aftermath State

V(Search and Rescue) = R(Search and Rescue) + γ \* P(Search and Rescue) \* V(next state) = 5 + 0.9 \* 0.8 \* V(next state)

V(Assess Damage) = R(Assess Damage) + γ \* P(Assess Damage) \* V(next state) = 2 + 0.9 \* 0.2 \* V(next state)

The agent selects the action with the higher expected reward.

## Hurricane Warning State

V(Evacuate Areas) = R(Evacuate Areas) + γ \* P(Evacuate Areas) \* V(next state) = 5 + 0.9 \* 0.9 \* V(next state)

V(Secure Buildings) = R(Secure Buildings) + γ \* P(Secure Buildings) \* V(next state) = 3 + 0.9 \* 0.1 \* V(next state)

The agent selects the action with the higher expected reward.

## Wildfire Spread State

V(Create Firebreaks) = R(Create Firebreaks) + γ \* P(Create Firebreaks) \* V(next state) = 4 + 0.9 \* 0.7 \* V(next state)

V(Evacuate Areas) = R(Evacuate Areas) + γ \* P(Evacuate Areas) \* V(next state) = 3 + 0.9 \* 0.3 \* V(next state)

The agent selects the action with the higher expected reward.

## Power Outage State

V(Restore Power) = R(Restore Power) + γ \* P(Restore Power) \* V(next state) = 3 + 0.9 \* 0.9 \* V(next state)

V(Provide Generators) = R(Provide Generators) + γ \* P(Provide Generators) \* V(next state) = 2 + 0.9 \* 0.1 \* V(next state)

The agent selects the action with the higher expected reward.

## Evacuation Order State

V(Organize Transport) = R(Organize Transport) + γ \* P(Organize Transport) \* V(next state) = 4 + 0.9 \* 0.8 \* V(next state)

V(Set Up Shelters) = R(Set Up Shelters) + γ \* P(Set Up Shelters) \* V(next state) = 3 + 0.9 \* 0.2 \* V(next state)

The agent selects the action with the higher expected reward.

## Search and Rescue State

V(Deploy Teams) = R(Deploy Teams) + γ \* P(Deploy Teams) \* V(next state) = 5 + 0.9 \* 1.0 \* V(next state)

The agent selects the action with the higher expected reward.

## Resource Allocation State

V(Distribute Food and Water) = R(Distribute Food and Water) + γ \* P(Distribute Food and Water) \* V(next state) = 3 + 0.9 \* 0.7 \* V(next state)

V(Distribute Medical Supplies) = R(Distribute Medical Supplies) + γ \* P(Distribute Medical Supplies) \* V(next state) = 4 + 0.9 \* 0.3 \* V(next state)

The agent selects the action with the higher expected reward.

## Infrastructure Repair State

V(Fix Roads) = R(Fix Roads) + γ \* P(Fix Roads) \* V(next state) = 3 + 0.9 \* 0.5 \* V(next state)

V(Restore Communication) = R(Restore Communication) + γ \* P(Restore Communication) \* V(next state) = 4 + 0.9 \* 0.5 \* V(next state)

The agent selects the action with the higher expected reward.

## Public Panic State

V(Broadcast Calm Messages) = R(Broadcast Calm Messages) + γ \* P(Broadcast Calm Messages) \* V(next state) = 2 + 0.9 \* 0.8 \* V(next state)

V(Ignore) = R(Ignore) + γ \* P(Ignore) \* V(next state) = -5 + 0.9 \* 0.2 \* V(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) ← Q(s\_t, a\_t) + α ( R(s\_{t+1}) + γ 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 α = 0.1, γ = 0.9, and assuming a maximum Q-value of 5 for the next state, the Q-value update would be:

*Q(Flood Alert, Strengthen Levees) ← 0 + 0.1 ( 4 + 0.9 × 5 - 0 ) ≈ 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.