Grid World

Worksheet, Hands-On AI for High School, grades 9 to 12

Name _____ Date _____

Read this first

You are about to train a robot. The robot is a red dot on a 6 by 6 grid, and it can do only four things: move up, down, left or right. It does not know where the goal is, and it does not know that the pits are bad. All it can do is try a move and see what number comes back.

Every move returns a reward. A penalty is simply a negative reward. The robot's only aim is to collect as much reward as it can, and it learns by remembering which moves paid off.

Event	Reward	Why
Reach the goal (star)	+10	We want this, so it pays a lot.
Fall into a pit (X)	-10	We do not want this, so it costs a lot.
Take a step anywhere else	-0.1	A small cost that encourages short paths.

How the robot learns

The method in the demo is called Q-learning. Here is the idea without the math.

- The robot keeps a table. For every square it stores four numbers, one for each direction. Each number is its current guess of how good it is to move that way from that square.
- At the start every number is zero, so the robot has no opinions.
- It picks a move, usually the one with the best number and sometimes a random one, and sees what reward comes back. Then it nudges that number up or down a little.
- This repeats thousands of times. The numbers change less and less as the robot settles on a way of moving. The arrows in the demo show the best direction from each square.

There is no map and no plan, only a table of numbers that is updated after every step.

Three settings that change everything

Epsilon (ε) is the chance that the robot makes a random move instead of the move it currently thinks is best. A high epsilon means a lot of exploring. A low epsilon means sticking with what it already knows.

Gamma (γ) sets how much the robot cares about future rewards compared with rewards right now. Near 0, it only looks one step ahead. Near 1, it plans far ahead. In practice gamma is usually set close to 1; textbook examples often use 0.9, and some popular software libraries use 0.99 by default.

Alpha (α) is the learning rate, how strongly the robot changes its table after each step. It is fixed at 0.1 in the demo. A higher alpha means faster but less steady learning.

Words to know

Term	Meaning
Agent	The thing that is learning, here the red dot. In the real world it could be a delivery drone, a game-playing program or a robot vacuum.
Environment	Whatever the agent moves around in. Here it is the 6 by 6 grid.
State	Where the agent is right now. In Grid World it is just the row and column.
Action	Something the agent can do. Here, up, down, left or right.
Reward	The number the environment returns after each action. Positive is good and negative is bad.
Episode	One attempt, from the start square until the agent reaches the goal, falls into a pit or runs out of time (100 steps).
Policy	The agent's strategy. After training, the arrows show it: from this square, go this way.
Q-value	The agent's estimate of how good a move is from a given square. Each square has four Q-values, and its color shows the best of them.

The experiments

Open Grid World in your browser from the course page, github.com/aalbata/ai-labs-for-high-school. Do the experiments in order, click Reset before each one, and write your prediction before you train.

Warm-up: just train it

Setup: the default settings (epsilon 0.20, gamma 0.90, default rewards). Click "Train 100 Episodes".

Predict: what will the arrows look like when training is done?

_____

Observe: what did you see? Where did the green color spread first? What was the success rate?

_____

Experiment 1: the greedy robot

Setup: set epsilon to 0.00 and train 100 episodes. The robot never explores on purpose.

Predict: with no random moves, will the robot find the goal? Why or why not?

_____

Observe: what was the success rate? Compare it with the warm-up. Every step costs 0.1, and every number in the table starts at zero. How might that push the robot to try moves it has not tried yet?

_____

Experiment 2: the random robot

Setup: set epsilon to 1.00, so that every move is random, and train 100 episodes.

Predict: the robot still updates its table even though it moves at random. Will it learn anything useful?

_____

Observe: what is the success rate? Do the arrows from the start lead to the goal? Now click “Train 100 Episodes” four more times without resetting. What happens to the arrows, and to the success rate?

_____

Experiment 3: short-sighted and far-sighted

Setup: set gamma to 0.10 and train. Then reset, set gamma to 0.99 and train. Compare the colors in squares far from the goal, and follow the arrows from the start square.

Predict: which setting will show brighter green far from the goal? What does that say about planning ahead?

_____

Observe: what did you notice? With gamma 0.10, do the arrows from the start square lead to the goal?

_____

Experiment 4: make every step painful

Setup: drag the step-cost slider to its most negative value (-1.00) and train 100 episodes.

Predict: the robot now wants each episode to end as quickly as possible. Will its path be shorter? Will it take more risks?

_____

Observe: do the arrows from the start lead to the goal yet? Train 200 more episodes and look again. The shortest path in this world is 10 steps; could the path get any shorter?

_____

Experiment 5: build a trap

Setup: switch to Edit World. Add pits around the goal so that only one safe square next to it is left. Switch back to Train Mode and train 100 episodes.

Predict: can the robot still reach the goal? Will it take longer than in the default world?

_____

Observe: what do the arrows near the goal look like? How does the success rate compare with the warm-up?

_____

Experiment 6: no step cost and a lot of randomness

Setup: set the step cost to 0 and epsilon to 0.50, then train 100 episodes.

Predict: with no penalty for wandering and many random moves, what will the arrows and the success rate look like?

_____

Observe: do the arrows still point to the goal? Why might the success rate be low even if the arrows are sensible?

_____

Think it through

Answer each question in a few sentences.

1. Q-learning has no idea what a goal is. It only receives +10 from somewhere. Suppose you were designing the rewards for a self-driving car. What reward would you give for reaching the destination, stopping at a stop sign, running a red light, or speeding by 5 mph? Does the size of each number matter, or only whether it is positive or negative?

_____

_____

2. Experiments 1 and 2 showed a robot that never explores and one that always does. Balancing the two is called the explore-or-exploit trade-off. Where do you face it in your own life, for example when ordering food, choosing a club or practicing an instrument?

_____

_____

3. Q-learning works here because the grid is small: 36 squares times 4 moves is 144 numbers to track. What goes wrong if you try the same table for chess or for driving a car? Why do you think modern AI uses neural networks for problems like these instead of giant tables?

_____

_____