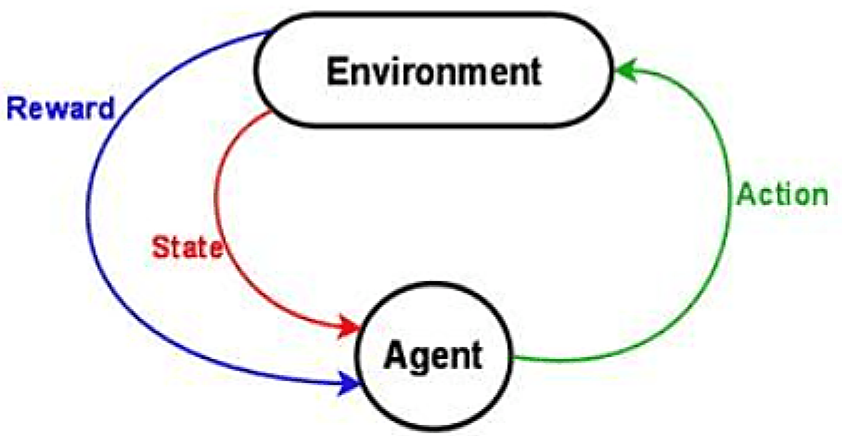
# GAME PLAYING

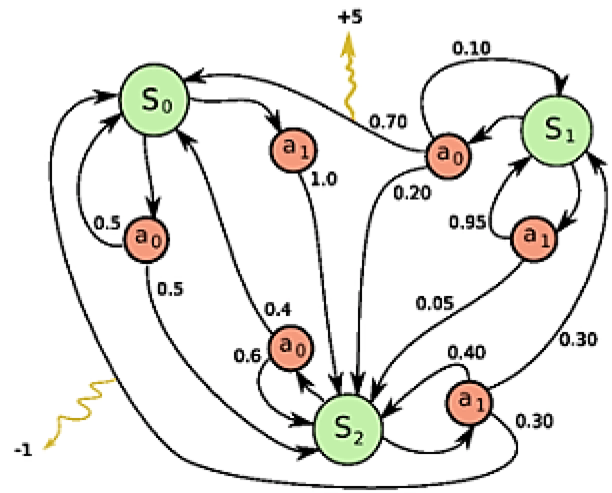
Themistoklis Diamantopoulos

# Reinforcement Learning

- Game
  - agent situated in an environment with a certain state
  - agent performs actions and receives rewards
- Markov Decision Process
  - Episode comprising states, actions, rewards

$$s_0, a_0, r_1, s_1, a_1, r_2, s_2, \ldots, s_{n-1}, a_{n-1}, r_n, s_n$$




Source: https://ai.intel.com/demystifying-deep-reinforcement-learning/

# Q-learning

- Update algorithm

Initialize $Q(s, a)$ arbitrarily
Repeat (for each episode):
  Initialize $s$
  Repeat (for each step of episode):
    Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\varepsilon$-greedy)
    Take action $a$, observe $r$, $s'$
    $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
    $s \leftarrow s'$;
  until $s$ is terminal

Exploitation vs Exploration

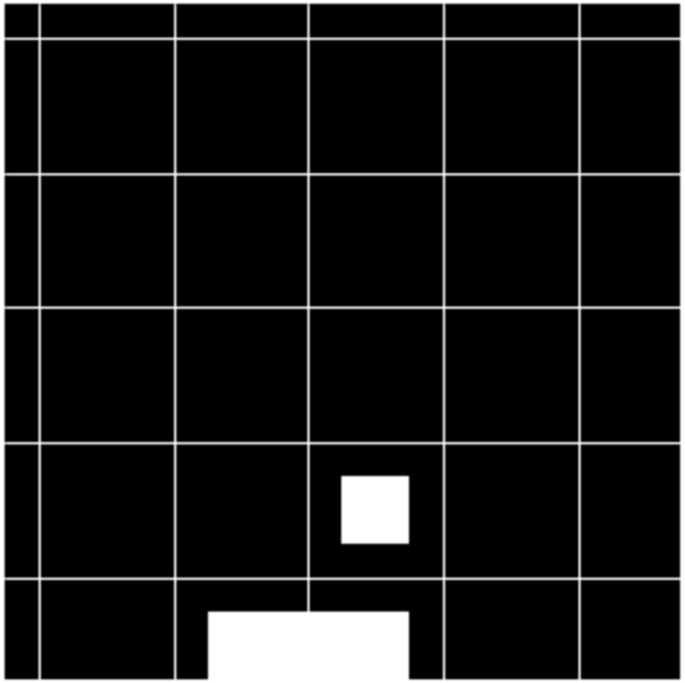
where:

s: current state
s´: next state
r: reward
a: current action
a´: next possible actions
γ: discount factor
α: learning rate

Source: https://www.slideshare.net/cprakash2011/reinforcement-learning-40052403

# Example 1: Catch

- Environment: grid
- State: position of the ball
- Actions: [left, stay, right]
- Reward: catch the ball
  - 1 if caught or -1 if not caught



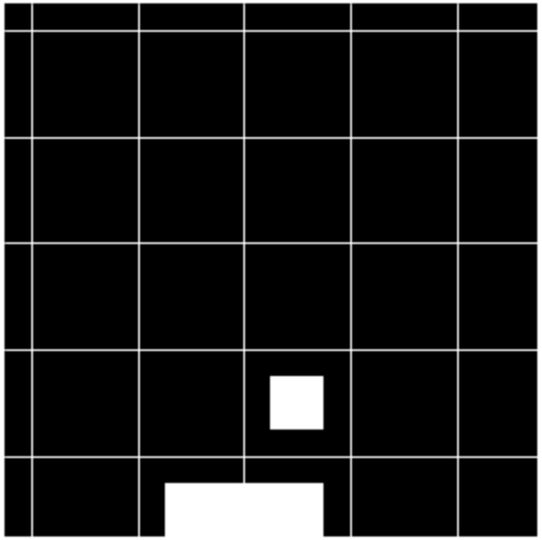
NEURAL NETWORKS ALGORITHM (s, a, r, s´)

1. For each a´ (left, stay, right) predict Q(s´, a´) (using the neural network)
2. Choose highest max{Q(s´, a´)}
3. Calculate r + γ * max{Q(s’, a’)} (this is the target value)
4. Train the network using the target value (minimize distance between predicted Q(s, a) and target)

Source: https://medium.freecodecamp.org/deep-reinforcement-learning-where-to-start-291fb0058c01

# Solution using MLP

- 3-layer fully connected network
- Input vector equal to state (full grid)
- Output layer: 3 nodes (actions)

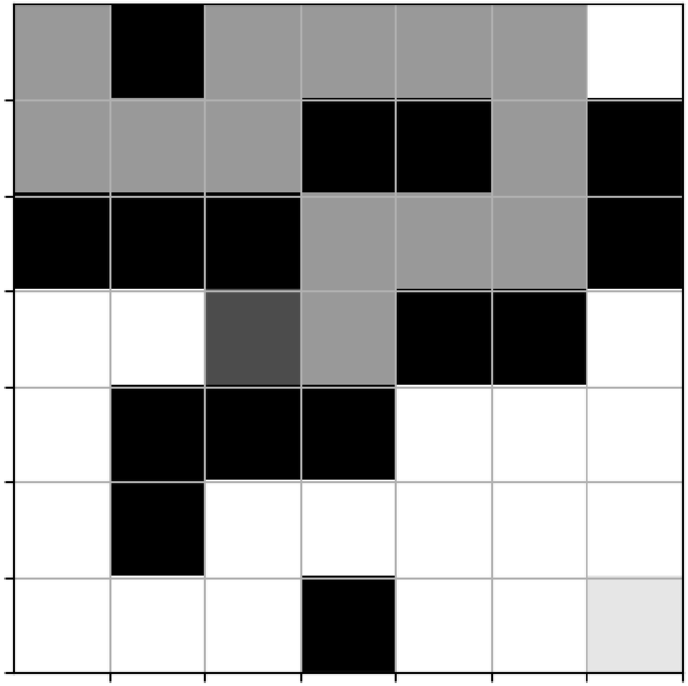


| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |

```
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_8 (Dense)              (None, 100)               10100
_________________________________________________________________
dense_9 (Dense)              (None, 3)                 303
=================================================================
```

Source: https://medium.freecodecamp.org/deep-reinforcement-learning-where-to-start-291fb0058c01

# Example 2: Maze

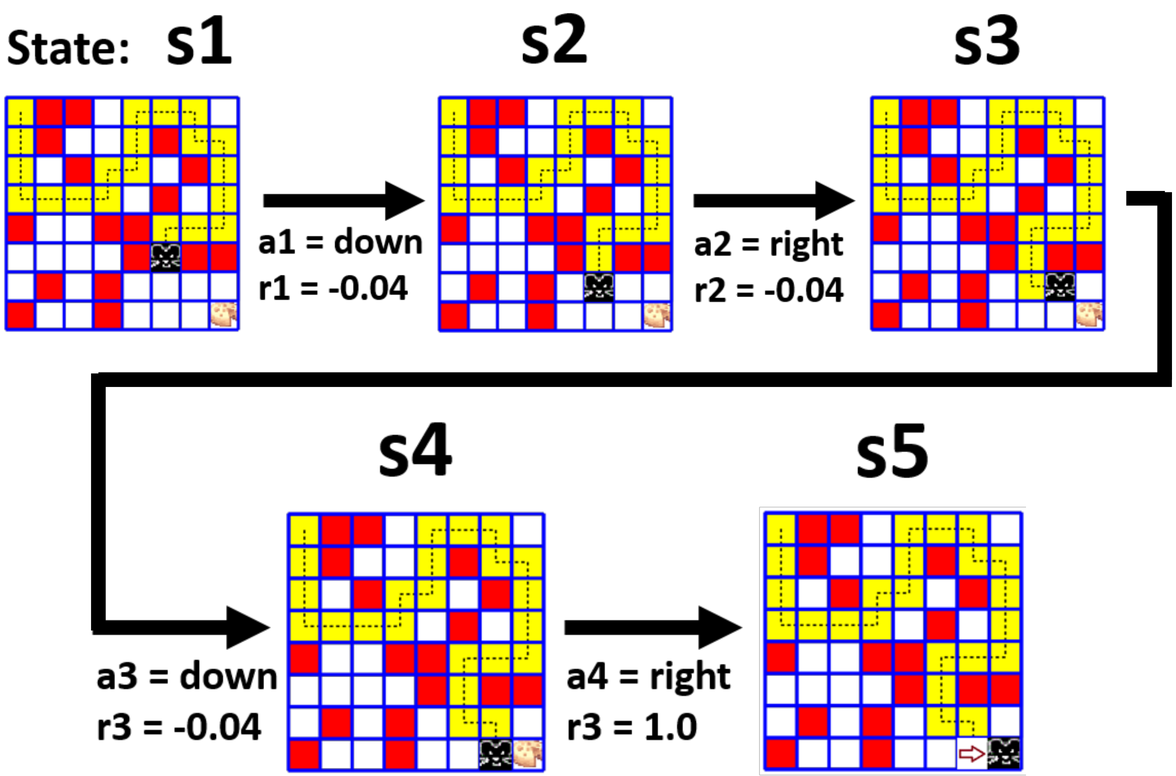


- Environment: grid, walls (■)
- State: position of player (■)
- Actions:
  - [left–0, up–1, right–2, down–3]
- Rewards:
  - 1 for catching the cheese (■)
  - -0.04 for each move to an open cell
  - -0.75 for trying to move into a wall (■)
  - -0.8 for trying to move outside the maze
  - -0.25 for moving to already visited cell (■)
- Game ends if cheese is caught or if reward < –mazesize/2

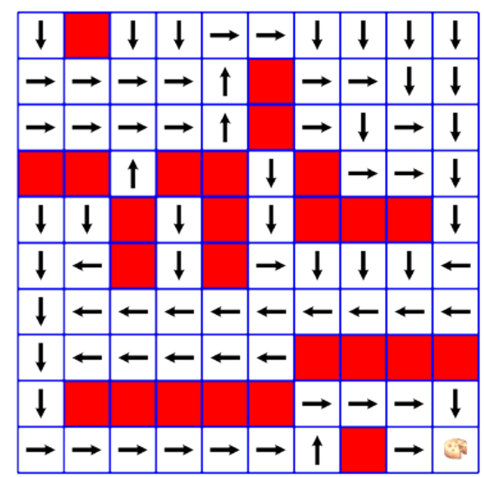
Source: http://www.samyzaf.com/ML/rl/qmaze.html

# Problem Modeling

- States, actions, and rewards
- Try to find a policy



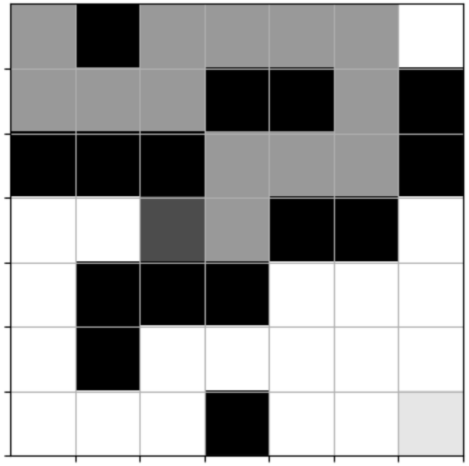

Policy



Source: http://www.samyzaf.com/ML/rl/qmaze.html

# Solution using MLP

- 3-layer fully connected network
- Input vector equal to state (full grid)
- Output layer: 4 nodes (actions)



| Layer (type) | Output Shape | Param # |
|---|---|---|
| dense_1 (Dense) | (None, 49) | 2450 |
| p_re_lu_1 (PReLU) | (None, 49) | 49 |
| dense_2 (Dense) | (None, 49) | 2450 |
| p_re_lu_2 (PReLU) | (None, 49) | 49 |
| dense_3 (Dense) | (None, 4) | 200 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | | | 1 | |
| | | | 1 | 1 | 1 | |
| 1 | 1 | 1 | 1 | | | 1 |
| 1 | | | | 1 | 1 | 1 |
| 1 | | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | | 1 | 1 | 1 |

Source: http://www.samyzaf.com/ML/rl/qmaze.html