OUTLINE

1) THEORY

2) IMPLEMENT THE GAME (ENVIRONMENT)

3) IMPLEMENT THE AGENT

4) IMPLEMENT THE MODEL

Reinforcement Learning

Is an area of machine learning concerned with how sotware agents (our computer player) ought to take
actions in an environment (our game in this case) in order to maximize the notion of cumulative reward.

In other words, RL is teaching a software agent how to behave in an environment by telling how good it's doing.

Deep Q Learning

This approach extends reinforcement learning by using a deep neural network to predict the actions.

OVERVIEW

AGENT

- Game
- model

store both in our agent

Action: [1,0,0] -> straight
[0,1,0] -> right turn
[0,0,1] -> left turn

Training: (loop)

- state = get_state(game)
- action = get_move(state)
  - model.predict()
- reward, game_over, score = game.play_step(action)
- new_state = get_state(game)
- remember
- model.train()

State: (11 Boolean values)
[danger straight, danger right, danger left,
direction left, dir. right, direction up, dir. down,
food left, " right, " up, " down]

GAME (Pygame)

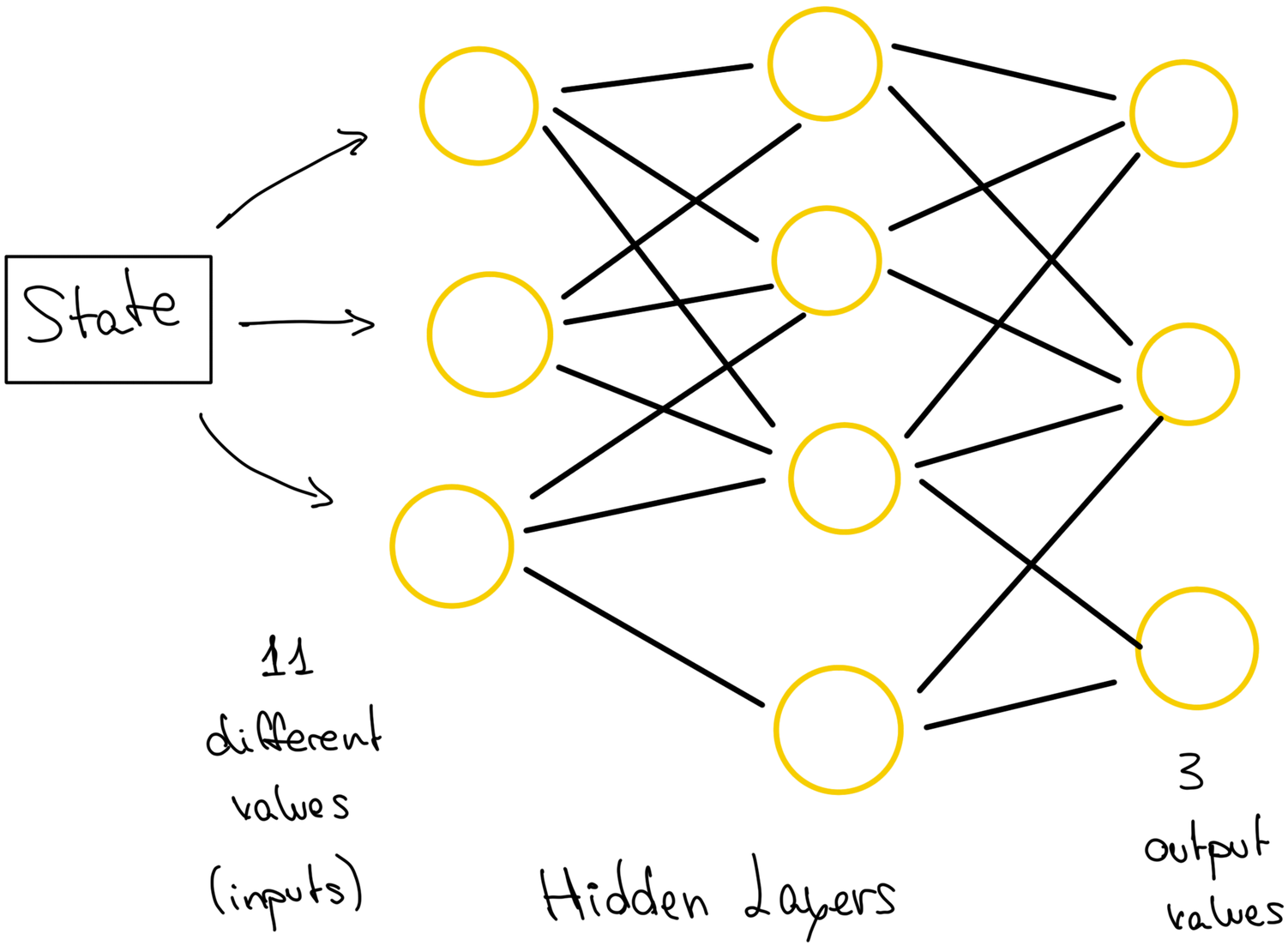
- play_step(action)
  -> reward, game_over, score

Reward:
- eat food: +10
- game over: -10
- else: 0

MODEL (PyTorch)

- Linear_QNet (DQN)
  - model.predict(state)
    -> action

MODEL



⇒ ACTION

e.g. [5.0, 2.7, 0.1]

max → [1, 0, 0]

(Deep) Q Learning

We want to improve! → Q value = Quality of action

0. Init Q value (= init model)

Repeat:

1. Choose action (model.predict(state))
2. Perform action
3. Measure reward
4. Update Q value (+ train model)

Bellman Equation (for calculating the new Q value)

state Action

$$\underbrace{New\,Q(s,a)}_{\text{New Q value for that state and that action}} = \underbrace{Q(s,a)}_{\text{Current Q value}} + \underbrace{\alpha}_{\text{Learning rate}}\left[\underbrace{R(s,a)}_{\text{Reward for taking that action at that state}} + \underbrace{\gamma}_{\text{Discount rate}} \underbrace{\max Q'(s',a')}_{\text{Max expected future reward given the new s' and all possible actions at that new state.}} - Q(s,a)\right]$$

Simplified:

$$(Old)\,Q = model.predict(state_0)$$

$$Q_{new} = R + \gamma \cdot \max(Q(state_1))$$

↑ Reward

Loss Function:

$$Loss = (Q_{new} - Q)^2$$

↳ mean squared error