Creating Intelligent Agents with Reinforcement Learning

Deep Q Learning

E. van Biljon Prof. J.A. du Preez

Department of Electrical and Electronic Engineering
Stellenbosch University

April 25, 2018

UNIVERSITEIT
STELLENBOSCH
UNIVERSITY

Outline

UNIVERSITEIT
STELLENBOSCH
UNIVERSITY

Outline

1. Reinforcement Learning
2. Deep Q Learning
3. Code vs Zombies
4. Experiments and Results

Traditional ML

Supervised Learning:

- input to output examples
- find mapping from input to known output

Unsupervised Learning:

- input examples
- find underlying pattern

Current predictions **do not** affect future predictions!

How RL differs

Reinforcement Learning:

- current input and current reward
- find optimal behaviour (series of decisions) to perform a task

Current predictions **do** affect future predictions!

Reinforcement Learning
Terminology

- Agent: entity that is making the decisions (our program)
- Environment: the world the agent interacts with
- State: current configuration of environment
- Action: a way in which the agent can interact with the environment
- Reward: the feedback the environment gives us

Reinforcement Learning

What is RL?

Branch of machine learning that deals with:

- making sequential decisions
- modelling behaviour $\rightarrow$ policy

Model a setup where some agent interacts with some environment.

Reinforcement Learning
Approaches

In what ways can we do this?

- Model-based
  Learn how the environment reacts to the agent's actions
- Value-based / Policy-search
  (Learn how to decide what action to take)
  - Policy-search:
    $\rightarrow$ Directly predict the action that results in highest cumulative reward
  - Value-based:
    $\rightarrow$ Assign value to states (value = cumulative reward)
    $\rightarrow$ perform action that would put the agent in the state with the highest value

Outline

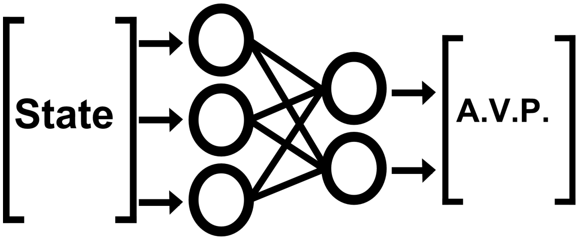
Deep Q Learning

Value-based

Learn a function that maps state to action-values using a neural network

Action-value: value of state agent will be in if specific action is performed

This is then called a Q network



UNIVERSITEIT STELLENBOSCH UNIVERSITY

Deep Q Learning

How is it done in practice?

Episode: session of interactions with the environment

Algorithm 1 Q Learning

```
while environment.episode_in_progress():
    s = environment.get_state()
    a = Q.choose_action(s)
    environment.update(a)
    new_s, r = environment.observe()
    Q.update(s, a, r, new_s)
```

UNIVERSITEIT STELLENBOSCH UNIVERSITY

Outline

1. Reinforcement Learning
2. Deep Q Learning
3. Code vs Zombies
4. Experiments and Results

Code vs Zombies
Description

Entities:

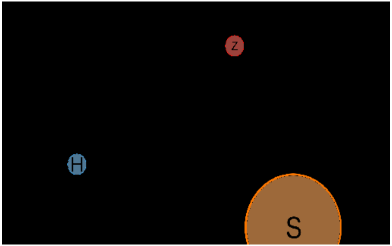
- a shooter → move towards player specified point
- humans → static
- zombies → move towards closest human (shooter is human)

Interactions:

- zombies kill humans → reduction in score
- shooter kills zombies → increase in score

Terminating states:

- no more humans → fail
- no more zombies → win

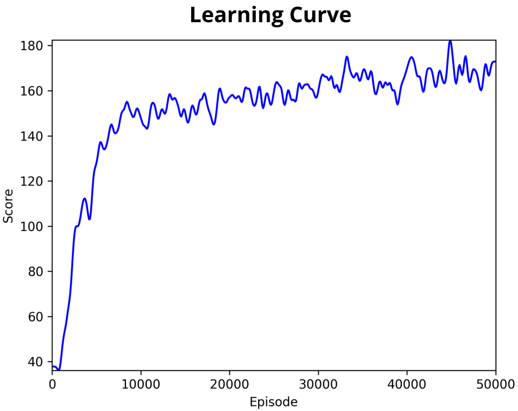


Outline

Experiments
Validation

Validation: time vs accuracy trade off

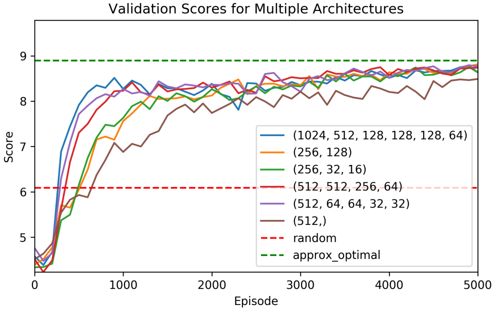
- validation score: average over 100 on-policy games



Results

Optimal score on some constrained versions:

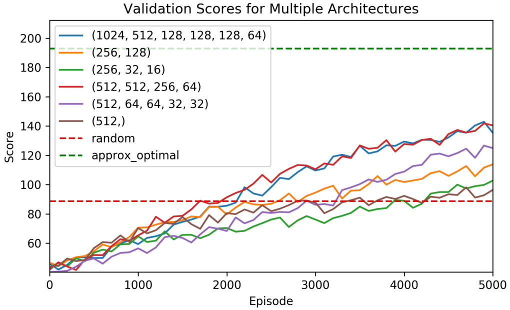
- 1 human
- 1 zombie



Results

Optimal score on some constrained versions:

- 3 humans
- 3 zombies



Want to find out more?

- Textbook:
  "Reinforcement Learning: An Introduction" by Sutton, R.S. and Barto, A.G.
- Papers:
  - Mnih, Volodymyr et al. (2015). Human-level control through deep reinforcement learning. In: Nature 518, pp. 529542
  - Hessel, M. et al. (2017). Rainbow: Combining Improvements in Deep Reinforcement Learning. arXiv:1710.02298
- Me:
  `https://elanvb.github.io/CV/`

UNIVERSITEIT STELLENBOSCH UNIVERSITY

Demo

Questions?

Value-based method

Q learning

We focus on value-based methods:

- Value function: measure state desirability

Q learning:

- assign value to state-action pairs
- action-value function $\rightarrow$ Q function

$$Q(s,a) := r + \gamma \cdot \max_a Q(s', a)$$

Q learning

What are the problems with this approach?

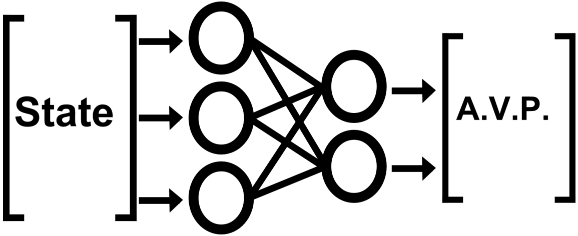
- usually modelled with Q matrix: $Q[s][a] = Q(s, a)$
- possible states as rows
- possible actions as columns
- as number of states and actions grow, matrix becomes very large
  $\rightarrow$ difficult to populate full matrix
  $\rightarrow$ does not generalise to unseen states

Deep Q Learning

Use a neural network to approximate the Q function:

$$\widehat{Q}(s, a, \theta) = NN(s, a, \theta)$$

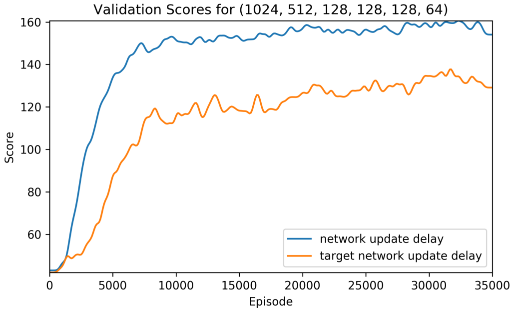
This is then called a Q network



Experiments

Training Methods

- target network update delay: use 2 networks for solution stability (Deep Mind - 2015)
- network update delay: achieve same benefits and more with a single network (Ours)



UNIVERSITEIT STELLENBOSCH UNIVERSITY