Microsoft

Reinforcement Learning with Function Approximation

KATJA HOFMANN

Researcher, MSR Cambridge

Contents

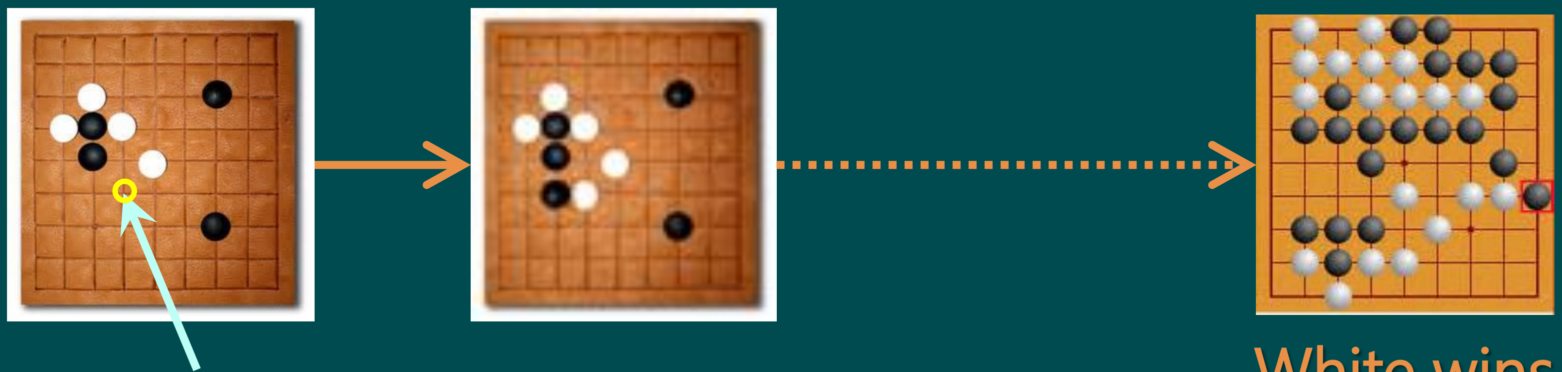
Overview

To solve reinforcement learning,

Reminder: challenges discussed in module 2

We must overcome 4 fundamental challenges:

- Representation
- Generalization
- Temporal Credit Assignment
- Exploration



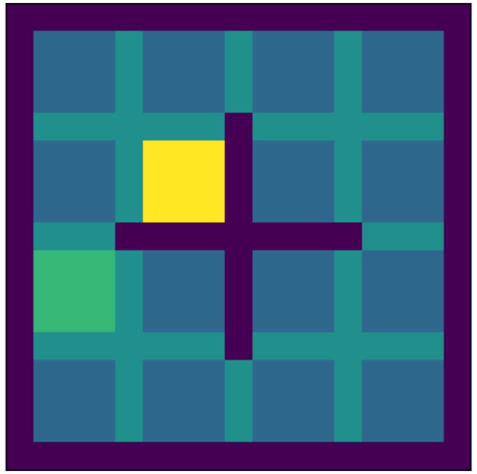
To solve reinforcement learning,

We must overcome 4 fundamental challenges:

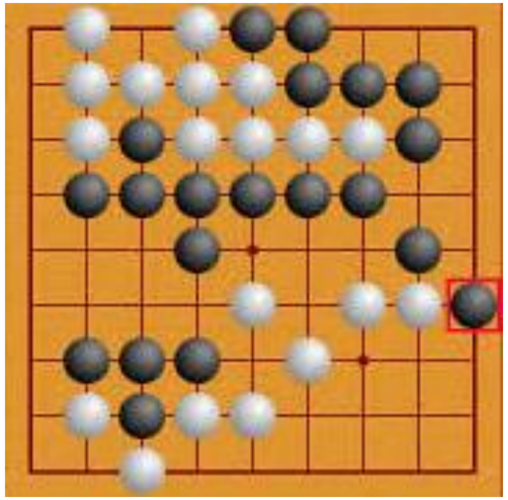
- Representation
- Generalization
- Temporal Credit Assignment
- Exploration

Challenge: Representation

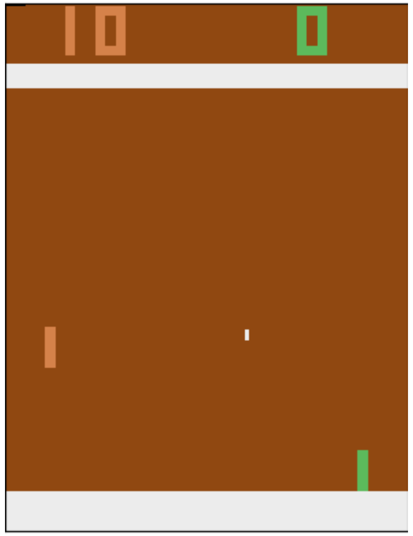
Grid Maze



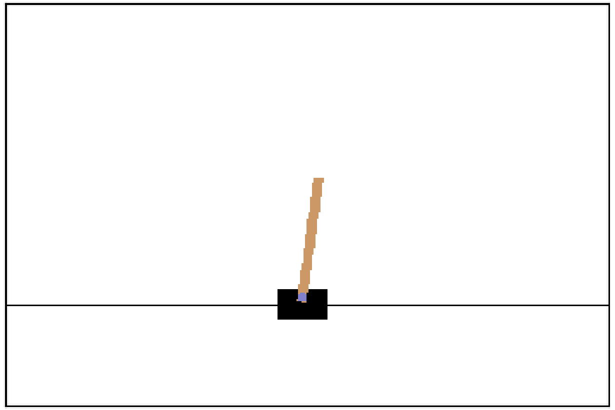
Go



Atari (Pong)



Cart Pole



Aim: generalizable representations

- From state (state-action) values to state (state-action) value functions V($\phi(s)$) and $Q(\phi(s,a))$, with k-dimensional observations

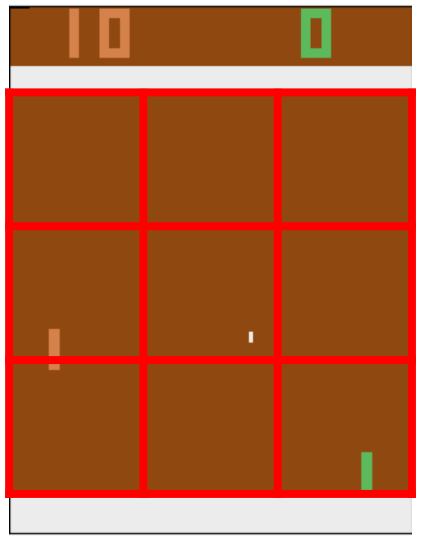
$$\phi(s) = \begin{pmatrix} \phi_{11} & \cdots & \phi_{1k} \\ \vdots & \ddots & \vdots \\ \phi_{n1} & \cdots & \phi_{nk} \end{pmatrix}$$

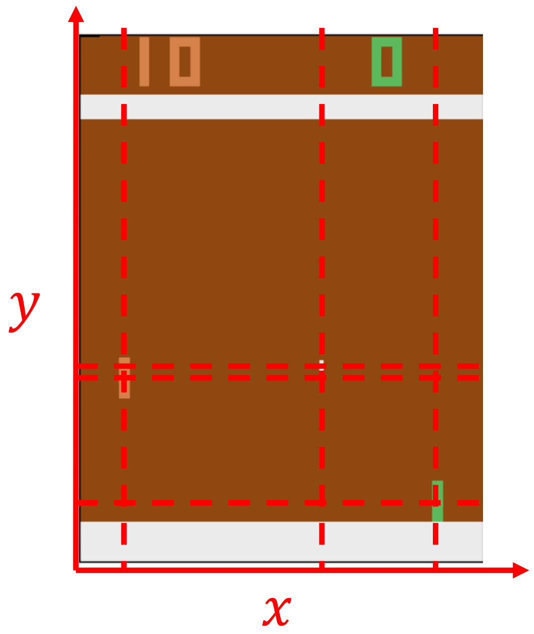
- Why can this work?
  Consider high dimensional observations vs lower dimensional underlying state
- Ideal representation: sample efficient training and good generalization to unseen parts of the observation space

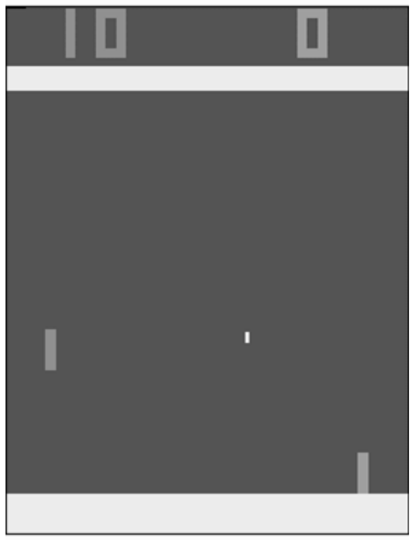
Starting point: define observation space

- Important aspect of problem formulation
- Opportunity to encode expert knowledge
- Trade-off: sample efficient learning vs generality

Example: observation space for Pong







Example: $k\ =\ 3\ x\ 9$

indicate if center of each paddle and ball is in a given grid cell

Example: $k\ =\ 3\ x\ 2$

indicate x, y coordinate of ball and paddles

Example: $k\ = (84\ x\ 84)256$

84x84 8-bit gray-scale pixel values of scaled image

Linear Function Approximation

Linear function approximators

- Recall – recursive formulation of the value function:

$$v^{\pi}(s_t) = \sum_{s_{t+1}} R(s_t) + P(s_t, s_{t+1}; \pi)\gamma v^{\pi}(s_{t+1})$$
$$= \mathbb{E}_{\pi}[R(s_t) + \gamma v^{\pi}(s_{t+1})]$$

- And assume linear model:

$$v^{\pi}(s_t) = \theta^T \phi(s_t)$$

  where θ denotes the model parameters

- Formulate temporal difference error:

$$\delta_{t+1} = R_t + \gamma \theta^T \phi(s_{t+1}) - \theta^T \phi(s_t)$$

Iterative algorithm

- Convenient update given samples s_t, r_t, s_{t+1}

$$\delta \leftarrow r_t + \gamma \theta^T \phi(s_{t+1}) - \theta^T \phi(s_t)$$

$$\theta \leftarrow \theta + \alpha \delta \phi(s_t)$$

- Approximately minimizes the squared loss (under certain conditions):

$$\mathrm{J}(\theta) = \|r_t + \gamma \theta^T \phi(s_{t+1}) - \theta^T \phi(s_t)\|^2$$

Extension to linear Q-learning

- Approximate state-action instead of actions values and update given samples s_t, a_t, r_t, s_{t+1} and feature extractor $\phi(s, a)$

$$\delta \leftarrow r_t + \gamma \max_{a \in A} \theta^T \phi(s_{t+1}, a) - \theta^T \phi(s_t, a_t)$$

- Corresponding to the squared loss:

$$\mathrm{J}(\theta) = \left\| r_t + \gamma \max_{a \in A} \theta^T \phi(s_{t+1}, a) - \theta^T \phi(s_t, a) \right\|^2$$

- For a complete Q-learning algorithm with linear function approximation, follow by policy improvement step to derive policy $\pi(s)$

$$\pi(s) \leftarrow \max_{a \in A} \theta^T \phi(s, a)$$

Summary

- RL with function approximation, aim: representation that can be learned efficiently and generalizes well
- Trade-off between manually constructed and learned components
- RL with Linear function approximation – approximate values using linear weighted feature combinations

Further Reading

- **Reinforcement Learning – An Introduction** (Sutton and Barto, 2017):
  - http://www.incompleteideas.net/sutton/book/the-book-2nd.html
  - Part II: approximate solution methods + references
- **Algorithms for Reinforcement Learning** (Szepesvári, 2010). *Synthesis Lectures on Artificial Intelligence and Machine Learning:*
  - More details on least squares methods (section 3) + overview of convergence results + references

Lab 1

RL with Deep Neural Networks

Beyond linear function approximation

Recall Q-learning with linear function approximation minimizes squared loss between current Q and targets. More generally:

$$\mathrm{J}(\theta_i) = \left\| r_t + \gamma \max_{a \in A} Q\left(\phi(s_{t+1}, a); \theta_i^{-}\right) - Q\left(\phi(s_t, a); \theta_i\right) \right\|^2$$

Update parameters θ using stochastic gradient descent (SGD):

$$\nabla_{\theta_i} \mathrm{J}(\theta_i) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1} \sim D}\left[r_t + \gamma \max_{a \in A} Q\left(\phi(s_{t+1}, a); \theta_i^{-}\right) - Q\left(\phi(s_t, a_t); \theta\right) \right] \nabla_{\theta_i} Q\left(\phi(s_t, a_t); \theta\right)$$

Deep Q-Learning: algorithm outline

- Policy evaluation step:

$$\delta_t \leftarrow r_t + \gamma \max_{a \in A} Q\left(\phi(s_{t+1}, a); \theta_i^-\right) - Q\left(\phi(s_t, a); \theta_i\right)$$
$$\theta_{t+1} = \theta_t + \alpha(\delta_t - Q(\phi(s,a);\theta))\nabla_{\theta_t} Q(\phi(s,a);\theta_t)$$

- Policy improvement step:

$$\pi_t(s) \leftarrow \max_{a \in A} Q(\phi(s,a);\theta_t)$$

- In practice: use standard deep learning frameworks to compute gradients and apply parameter updates

Beyond linear function approximation

Stochastic gradient methods – key to using powerful function approximators, in particular deep neural nets

Benefit: use tools from supervised learning – deep learning frameworks, optimizers

Challenge: stability / robustness

Target Network

In deep Q-learning, regression targets are non-stationary:

$$J(\theta_i) = \left\| r_t + \gamma \max_{a \in A} Q\left(\phi(s_{t+1}, a); \theta_i^-\right) - Q\left(\phi(s_t, a); \theta_i\right) \right\|^2$$

Achieve temporary stability, by using a separate target network θ_i^- to compute update targets

[Mnih et al. 2015]

Replay Memory

Updates use minibatches sampled from a data buffer D

$$\nabla_{\theta_i} \mathrm{J}(\theta_i) = \mathbb{E}_{s_t, a_t, r_t, s_{t+1} \sim D}\left[r_t + \gamma \max_{a \in A} Q\left(\phi(s_{t+1}, a); \theta_i^{-}\right) - Q\left(\phi(s_t, a_t); \theta\right)\right] \nabla_{\theta_i} Q\left(\phi(s_t, a_t); \theta\right)$$

Implemented as large capacity "replay buffer" of recent interactions

[Mnih et al. 2015]

Summary

- Deep Q-learning formulation enables use of powerful function approximators, e.g., Deep Neural Networks
- Leverage tools from supervised learning for effective learning
- Challenge: no convergence guarantee, need to stabilize learning

Further Reading

- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., … Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.
  - Introduce DQN, breakthrough result on learning to play Atari from pixels

- Riedmiller, M. (2005). Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. ECML 2005, 317-328.
  - Propose use of neural nets for Q-learning, Rprop optimizer – promising results

Lab 2

Extensions and Practical Considerations

Extensions of Deep Q-Learning

Focus on increasing robustness, reducing bias and/or variance

Here: Double DQN, prioritized replay memory

Double DQN

Start from target computation for Deep Q-Learning:

$$Y^{DQN} = r_t + \gamma \max_{a \in A} Q(\phi(s_{t+1}, a); \theta_i^-)$$

Problem: biased estimator – $\max\limits_{a \in A}$ and $Q(\phi(s_{t+1}, a); \theta_i^-)$ rely on the same estimator, parameterized by θ_i^-. Proposed algorithm – compute targets using:

$$Y^{DDQN} = r_t + \gamma Q(\phi(s_{t+1}, \operatorname*{argmax}_{a \in A} Q(\phi(s_{t+1}, a); \theta_i); \theta_i^-)$$

[van Hasselt et al. 2016]

Prioritized Replay Memory

Idea – instead of uniform sampling from replay buffer D, prioritize transitions with high loss.

Proposed approach – sample with

$$p(s_t, a_t, r_t, s_{t+1}) \propto r_t + \gamma \max_{a \in A} Q(\phi(s_{t+1}, a); \theta_i^-)$$

[Schaul et al. 2016]

Practical considerations

For SGD / network architecture – see supervised deep learning approaches

Additional consideration for Deep Q-learning: exploration, replay memory size

Data efficiency – interactions may be slow / costly; reference work on multi-task / meta-learning

Further Reading

- van Hasselt, H., Guez, A., & Silver, D. (2016). Deep Reinforcement Learning with Double Q-learning. AAAI 2016, 2094-2100 http://arxiv.org/abs/1509.06461
- Schaul, T., Quan, J., Antonoglou, I., Silver, D. (2016). Prioritized Experience Replay. ICLR 2016 https://arxiv.org/abs/1511.05952
- Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). A Brief Survey of Deep Reinforcement Learning. IEEE SPM Special Issue on Deep Learning for Visual Understanding http://arxiv.org/abs/1708.05866