Advanced Topics on Artificial Intelligence

Alban Grastien

The Australian National University

Second Semester, 2020

Model-free decision

- We are now considering the problem of decision under uncertainty when **the model is unknown**
  - We do not know the transition function $P(s, a, s')$
  - We do not know the reward function $R(s, a, s')$
  - But the state is still fully observable

My Previous Lecture: Bandit

Two important problems:

- How to update the knowledge
  - E.g., average outcome, temporal difference
- How to balance exploration / exploitation
  - E.g., ε-greedy strategy or UCB

Today

- Evaluation of a policy
- Computing the optimal (?) policy: SARSA / Q-learning

Yesterday / today

Context: Reinforcement Learning (the P and R functions of the MDP are unknown)

- Yesterday:
  - How to estimate the value of a policy
  - How to compute a good / optimal policy anytime (SARSA, Q-Learning)

- Today: Problems with large number of states
  - Parametrised policies
  - Finding good parametrised policies:
    - Gradient descent
    - Analytical solutions

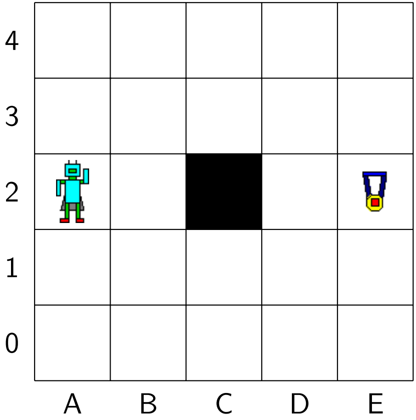
Issue with SARSA/Q-Learning

- If there are too many states (e.g., 10^9 and more), it is impossible to maintain an estimate of $V(s)$ or $Q(s,a)$ for each state/action
- These methods are also not able to generalise:
  - if two states are very similar, then the optimal action is very likely to be the same in both states

Parametrised Policies

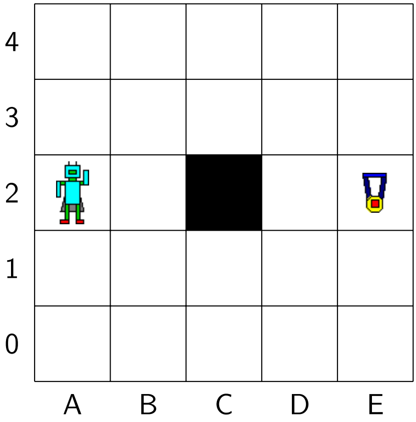
- A $\boxed{\text{parametrised policy}}$ is a function $\pi_{\boldsymbol{\theta}}(s) \to Prob(A(s))$ where $\boldsymbol{\theta}$ is a vector (list of parameters)
  - Different values to the parameters yield a different policy
- We write: $\boldsymbol{\theta} = [\theta_1, \theta_2, \ldots, \theta_n]$
- Next slides: examples

Example of Parametrised Policy (1)

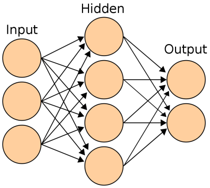


- We compute a score $f(s, a)$ for each state s and each action a:
  $f((x, y), \uparrow) = \min(\varepsilon, x\theta_1 + y\theta_2)$
  $f((x, y), \downarrow) = \min(\varepsilon, x\theta_3 + y\theta_4)$
  $f((x, y), \rightarrow) = \min(\varepsilon, x\theta_5 + y\theta_6 y)$
  $f((x, y), \leftarrow) = \min(\varepsilon, x\theta_7 + y\theta_8 y)$
- Here $\boldsymbol{\theta} = [\theta_1, \dots, \theta_8]$
- If $a \in A(s)$, the probability of applying action a from s is:
  $\pi_{\boldsymbol{\theta}}(s)(a) = \frac{f(s,a)}{\Sigma_{a' \in A(s)} f(s,a')}$

Example of Parametrised Policy (2)



Neural network



- Parameters = weights of the connections between the neurons
- What are the inputs of the NN?

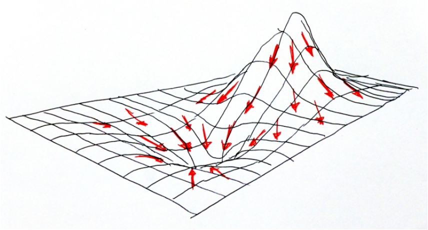
Optimal Parameters

- Given the parameters $\boldsymbol{\theta}$, the value of the initial state s_0 for $\pi_{\boldsymbol{\theta}}$ is $V_{\boldsymbol{\theta}}(s_0) := V_{\pi_{\boldsymbol{\theta}}}(s_0)$
- The optimal parameters $\boldsymbol{\theta}^*$ are those such that

$$V_{\boldsymbol{\theta}^*}(s_0) \geq V_{\boldsymbol{\theta}}(s_0) \qquad \text{for all parameters } \boldsymbol{\theta}$$

- There may be no parameter that is optimal **for all states**
- From now on, I will use the notation $V(\boldsymbol{\theta})$ to represent $V_{\boldsymbol{\theta}}(s_0)$

Gradient Descent — General Idea



- The derivative $\frac{\partial f}{\partial x}(x)$ of function $f(x)$ indicates in which direction to go in order to increase / decrease $f(x)$
- The gradient is similar for several parameters

Gradient Descent

- Assume that $\pi_{\boldsymbol{\theta}}(s)$ is differentiable wrt $\boldsymbol{\theta}$
- The gradient of the $V(\boldsymbol{\theta})$ is the vector:

$$\nabla V(\boldsymbol{\theta}) = \left[\frac{\partial V(\boldsymbol{\theta})}{\partial \theta_1}, \frac{\partial V(\boldsymbol{\theta})}{\partial \theta_2}, \ldots, \frac{\partial V(\boldsymbol{\theta})}{\partial \theta_n} \right]$$

- Let α be a step size.
- Then we update the parameters $\boldsymbol{\theta}' := \boldsymbol{\theta} + \alpha \nabla V(\boldsymbol{\theta})$
- If α is small enough, then $V(\boldsymbol{\theta}')$ is better than $V(\boldsymbol{\theta})$

How to Compute the Gradient?

First method: Estimate

- Let $\mathbf{1}_i$ be the vector $[\underbrace{0,\ldots,0}_{i-1},1,0,\ldots,0]$
- By definition:

$$\frac{\partial V(\boldsymbol{\theta})}{\partial \theta_i} = \lim_{\varepsilon \to 0} \frac{V(\boldsymbol{\theta}') - V(\boldsymbol{\theta})}{\varepsilon}$$

  where $\boldsymbol{\theta}' := \boldsymbol{\theta} + \varepsilon \mathbf{1}_i$ is a perturbation of the parameter vector $\boldsymbol{\theta}$

- Just choose ϵ small enough and assume

$$\frac{\partial V(\boldsymbol{\theta})}{\partial \theta_i} \simeq \frac{V(\boldsymbol{\theta}') - V(\boldsymbol{\theta})}{\epsilon}$$

Issues of this Method for Gradient Descent

- Requires a lot of computation, because $V(\boldsymbol{\theta})$ needs to be accurate
- Gradient descent often ends up in local optima

How to Compute the Gradient?

Second method: Analytically

- Assumptions & Notations
  - Episodic problems (with final states)
  - A run, or a history, is $h = [s_0, a_1, s_1, a_2, \ldots, s_{k_h}]$
  - The set of all histories is $\mathcal{H}$

- The value associated with parameters $\boldsymbol{\theta}$ is

$$V(\boldsymbol{\theta}) = \Sigma_{h \in \mathcal{H}} \; P(h \mid \boldsymbol{\theta}) \times R(h).$$

- We want to compute

$$\nabla_{\boldsymbol{\theta}} V(\boldsymbol{\theta})$$

- Notice that $P(h \mid \boldsymbol{\theta})$ is unknown (and $R(h)$ too)!

Analytical method (cont.)

$\nabla_{\boldsymbol{\theta}} V(\boldsymbol{\theta})$

- $= \nabla_{\boldsymbol{\theta}} \Sigma_{h \in \mathcal{H}} \left(P(h \mid \boldsymbol{\theta}) \times R(h) \right)$
- $= \Sigma_{h \in \mathcal{H}} \left(R(h) \times \nabla_{\boldsymbol{\theta}} P(h \mid \boldsymbol{\theta}) \right)$
- $= \Sigma_{h \in \mathcal{H}} \left(R(h) \times \frac{P(h|\boldsymbol{\theta}) \times \nabla_{\boldsymbol{\theta}} P(h|\boldsymbol{\theta})}{P(h|\boldsymbol{\theta})} \right)$
- $= \Sigma_{h \in \mathcal{H}} \left(R(h) \times P(h \mid \boldsymbol{\theta}) \times \nabla_{\boldsymbol{\theta}} \log(P(h \mid \boldsymbol{\theta})) \right)$
- $\simeq \frac{1}{m} \Sigma_{i \in \{1,\dots,m\}} \left(R(h^i) \times \nabla_{\boldsymbol{\theta}} \log(P(h^i \mid \boldsymbol{\theta})) \right)$ where $h^1, \dots, h^m$ are sampled trajectories

Analytical method (cont.)

Looking at a single history $h = [s_0, a_1, s_1, a_2, \ldots, s_k]$

$\nabla_{\boldsymbol{\theta}} \log(P(h \mid \boldsymbol{\theta}))$

- $= \nabla_{\boldsymbol{\theta}} \log(P(s_0 \mid \boldsymbol{\theta}) \times P(a_1 \mid s_0, \boldsymbol{\theta}) \times P(s_1 \mid s_0, a_1, \boldsymbol{\theta}) \times P(a_2 \mid s_1, \boldsymbol{\theta}) \ldots)$
- $= \nabla_{\boldsymbol{\theta}} \Big(\log(P(s_0 \mid \boldsymbol{\theta})) + \log(P(a_1 \mid s_0, \boldsymbol{\theta})) + \log(P(s_1 \mid s_0, a_1, \boldsymbol{\theta})) + \log(P(a_2 \mid s_1, \boldsymbol{\theta})) + \ldots \Big)$
- $= \nabla_{\boldsymbol{\theta}} \log(P(s_0 \mid \boldsymbol{\theta})) + \nabla_{\boldsymbol{\theta}} \log(P(a_1 \mid s_0, \boldsymbol{\theta})) + \nabla_{\boldsymbol{\theta}} \log(P(s_1 \mid s_0, a_1, \boldsymbol{\theta})) + \nabla_{\boldsymbol{\theta}} \log(P(a_2 \mid s_1, \boldsymbol{\theta})) \ldots$
- $= 0 + \nabla_{\boldsymbol{\theta}} \log(P(a_1 \mid s_0, \boldsymbol{\theta})) + 0 + \nabla_{\boldsymbol{\theta}} \log(P(a_2 \mid s_1, \boldsymbol{\theta})) + \ldots$

This value can be computed because we know $P(a \mid s, \boldsymbol{\theta})$ for each s, a, $\boldsymbol{\theta}$

AlphaGo

Silver, David, et al. "**Mastering the game of go without human knowledge**." *Nature* 550.7676 (2017): 354-359.

Three ingredients:

- Monte Carlo Tree Search
- A neural network that computes good moves (used for simulation by MCTS)
- A neural network that predicts the winner of a game (to prune the search depth)

Last Word

https://www.youtube.com/watch?v=_cQITY0SPiw

Discussion about past, present, and future of AI