Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Matthias Schubert
Sebastian Schmoll, Sabrina Friedl

Munich, 28.01.2019

Deep Learning and Artificial Intelligence

WS 2018/19

Exercise 13: Policy Gradient Methods

Exercise 13-1 Softmax Policy

(a) Name an example of a situation in which a stochastic policy is better than a deterministic one.

(b) Given a feature vector $\mathbf{x}(s, a)$ for state s and action a, and a weight vector θ with $\mathbf{x}, \theta \in \mathbb{R}^d$. The Softmax policy π_θ parameterized by θ is defined as:

$$\pi_\theta(a \mid s) = \frac{e^{\mathbf{x}(s,a)^T\theta}}{\sum_{a' \in \mathcal{A}} e^{\mathbf{x}(s,a')^T\theta}}$$

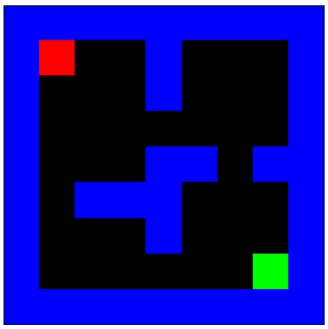
where $\mathcal{A}$ is the set of all possible actions.

Calculate the corresponding score function $\nabla_\theta \log \pi_\theta(a \mid s)$!

Exercise 13-2 REINFORCE

In this exercise, you will implement the REINFORCE algorithm, a policy gradient method that uses the return of complete episodes for the updates of the policy parameter.

On the website, you can find a zip file containing the files "rooms.py", "montecarlo_main.py" and "REINFORCE.py". The first file contains a class "RoomsEnv" which simulates the rooms domain depicted in the figure below.



The goal of the agent (red square, upper left) is to find a path to the green goal state (bottom right). The blue squares are walls that cannot be walked through. The reward is 0 at all steps and 1 when reaching the goal state. It is an episodic task and we use a discounting factor γ. The main file already implements the simulation of complete episodes (function "train") and stores the sampled states, actions and rewards received in separate arrays. After each episode, it calls `agent.update_montecarlo()`.

Task:

In the file "REINFORCE.py", implement the missing functionality for the class ReinforceAgent:

(a) The method `softmax_policy()` should return the probability $\pi(a \mid s)$ of choosing an action a in state s as given in exercise 13-1.

(b) The function `score_function()` should calculate the gradient $\nabla_\theta \log \pi_\theta(a \mid s)$

(c) The method `choose_action` should sample an action for state s according to the probabilities given by the softmax policy.

(d) Finally, the method `update_montecarlo()` should take the sampled states, actions and rewards arrays and perform the update of the policy parameter self.theta for each time step t:

$$\theta \leftarrow \theta + \alpha \; \gamma^t \; G_t \; \nabla_\theta \log \pi_\theta(a \mid s),$$

where G_t is the discounted return at time step t (for the last time step, it is just the last reward in the rewards array).

Train the agent and look at the results. You can also play around with different feature vectors. By calling `env.save_video()` you can generate a visualization of the agent acting according to the trained policy.

Exercise 13-3 Policy Gradient with Baseline

In this exercise we will take a look at the effect of the baseline in Policy Gradient algorithms. Assume we have an MDP and a stochastic policy $\pi_\theta(a|s)$ given. Recall, the goal of policy gradient is to update the parameters θ such that the expected return will be maximized. Let us consider the updates for the given state s. The feature vector $x(s, a)$ is defined as follows:

$$x(s,a) = \begin{cases} (1,0,0)^T & if a = a_1 \\ (0,1,0)^T & if a = a_2 \\ (0,0,1)^T & if a = a_3 \end{cases}$$

The average return G for action a_1, a_2, a_3 is $101, 110, 104$ respectively. Assume that we have values for θ such that the softmax-policy $\pi_\theta(a|s)$ is defined as follows:

$$\pi_\theta(a|s) = \begin{cases} 0.3 & if a = a_1 \\ 0.6 & if a = a_2 \\ 0.1 & if a = a_3 \end{cases}$$

(a) Compute the mean and the variance for the latter part of the update rule of policy gradient, that is

$$G_t \; \nabla_\theta \log \pi_\theta(a \mid s).$$

(b) What happens if we subtract a baseline $b(s) = 100$ from the return G_t? Compute the mean and variance for the formula

$$(G_t - b(s)) \; \nabla_\theta \log \pi_\theta(a \mid s).$$

(c) Compare the results of (a) and (b). What do you observe? How does the baseline effect the learning of the optimal policy?