Artificial Neural Networks (Gerstner). Exercises for week 2

Reinforcement Learning: Q-value and SARSA

Exercise 1. Recap of Online, Batch, and Expectations in ML (regression problem)[1]

We consider data coming from a distribution $p(x, y)$ where x is one-dimensional input and y the output. We have three possible x-values with $p(x = 0) = 0.2$ and $p(x = 1) = p(x = -1) = 0.4$. The output probabilities are

– $p(y = 1|x = 0) = p(y = 5|x = 0) = 0.5$

– $3 \cdot p(y = 1|x = 1) = p(y = 5|x = 1) = 0.75$

– $p(y = 1|x = -1) = 3 \cdot p(y = 5|x = -1) = 0.75$

The aim is to look at convergence of a linear estimator $f(x) = ax + b$ with a quadratic loss function $l\big(y, f(x)\big) = \big(y - f(x)\big)^2$, using gradient descent either in batch or in online mode.

a. Before you start, strengthen your intuitions. Suppose that we have drawn 50 points that match the statistical weights of the distribution. In total we have 50 points:
   – $\big\{(x_k, y_k) = (0, 1)\big\}_{k=1}^{5}$
   – $\big\{(x_k, y_k) = (0, 5)\big\}_{k=6}^{10}$
   – $\big\{(x_k, y_k) = (1, 1)\big\}_{k=11}^{15}$
   – $\big\{(x_k, y_k) = (-1, 5)\big\}_{k=16}^{20}$
   – $\big\{(x_k, y_k) = (-1, 1)\big\}_{k=21}^{35}$
   – $\big\{(x_k, y_k) = (1, 5)\big\}_{k=36}^{50}$

   Take a piece of paper, draw the points, and draw by eye the best linear fit. Trough which y-value should the line pass at $x = 0$ or at $x = 1$?

b. Here you can decide whether you would like to check the result in simulation (i) OR in theory (ii), i.e., you do not need to do both. Question (c) is again for everybody.

   (i) Simulation: Implement the task in Python: Use gradient descent with a small but fixed learning rate η. Follow the convergence of the parameters a and b to their final values. Compare batch mode (full batch of 50 samples) with online mode (one sample at a time). How do the fluctuations $\langle(\Delta b)^2\rangle$ scale with learning rate η? e.g., linear, quadratic, or other?

   Instead of the 50 handpicked examples above, draw N examples randomly and independently from the distribution $p(x, y)$. Run BATCH mode for a small η and observe the final result as a function of N. How close is it to the optimal solution?

   Start at the optimal solution for b that defines the minimum of the loss function for $N \to \infty$ but use only a few randomly picked samples. Do you stay at the exact minimum or does the solution move away? How about if you average your solution over 100 update steps? How does the answer to the last two questions change as a function of N?

   (ii) Theory: Solve for optimal parameters a^* and b^* analytically in the infinite data limit starting from the loss function. (*Hint:* you can also construct from general theorems of L2 loss and symmetries of the problem at hand).

   Then define the update step for the batch rule starting from arbitrary (a, b). Show that the update step is equal to 0 at (a^*, b^*).

   Start at the exact solution (a^*, b^*), but use the online rule by picking randomly from the 50 examples above. How big are the fluctuations? Based on calculations, how do the fluctuations $\langle(\Delta b)^2\rangle$ scale with the learning rate? e.g., linear, quadratic, or other?

[1] The result of Exercise 1 will be used in the first lecture of week 2.

c. Can you link the above results to the update of Q-values in the bandit problem? Can we relate Q-values to parameters? What would be states, actions, and rewards?

Exercise 2. SARSA algorithm

In the lecture, we introduced the SARSA (state-action-reward-state-action) algorithm, which (for discount factor $\gamma = 1$) is defined by the update rule

$$\Delta Q(s,a) = \eta \left[r - \big(Q(s,a) - Q(s',a') \big) \right] , \tag{1}$$

where s' and a' are the state and action subsequent to s and a. In this exercise, we apply a greedy policy, i.e., at each time step, the action chosen is the one with maximal expected reward, i.e.,

$$a_t^* = \arg\max_a Q_t(s,a) . \tag{2}$$

If the available actions have the same Q-value, we take both actions with probability 0.5.

Consider a rat navigating in a 1-armed maze (=linear track). The rat is initially placed at the upper end of the maze (state s), with a food reward at the other end. This can be modeled as a one-dimensional sequence of states with a unique reward ($r = 1$) as the goal is reached. For each state, the possible actions are going up or going down (Figure 1). When the goal is reached, the trial is over, and the rat is picked up by the experimentalist and placed back in the initial position s and the exploration starts again.

a. Suppose we discretize the linear track by 6 states, $s_1, \ldots, s_6$ where s_1 is the initial state and s_6 is the goal state. Initialize all the Q-values at zero. How do the Q-values develop as the rat walks down the maze in the first trial?

b. Calculate the Q-values after 3 complete trials. How many Q-values are non-zero? How many trials do we need so that information about the reward has arrived in the state just 'below' the starting state?

c. What happens to the learning speed if the number of states increases from 6 to 12? How many Q-values are non-zero after 3 trials? How many trial do we need so that information about the reward has arrived in the state just 'below' the starting state?

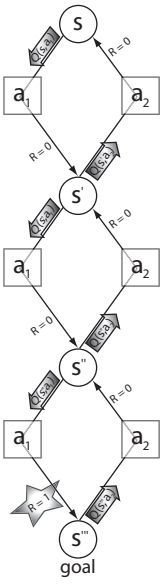

Figure 1: A linear maze.

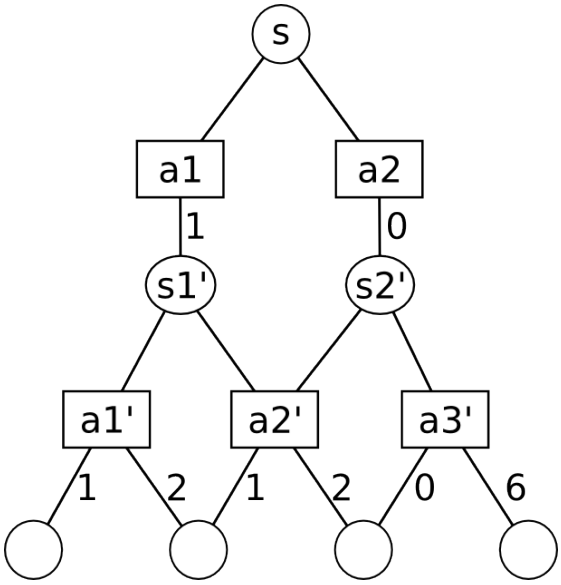

Figure 2: A tree–like environment.

Exercise 3. Bellman equation

Use the Bellman equation to calculate $Q(s, a1)$ and $Q(s, a2)$ for the environment shown in Figure 2. Consider two different policies:

- Total exploration: All actions are chosen with equal probability.
- Greedy exploitation: The agent always chooses the best action.

Note that the rewards/next states are stochastic for the actions $a1'$, $a2'$ and $a3'$. Assume that the probabilities for the outcome of these actions are all equal, and the discount factor γ is 1.

Exercise 4. Computer exercises: Environment 1 (part 1)[1]

Download the Jupyter notebook of the 1st computer exercise, setup your Python environment, and complete 'Exercise 0: One step horizon'.

[1] Start this exercise in the second exercise session of week 2.