Artificial Neural Networks (Gerstner). Solutions for week 8

Model-Based Deep Reinforcement Learning

Exercise 1. Background Planning.

In this exercise we look again at the simple map of Europe on slide 3. You will run value iteration with goal Vienna. This means that we will keep $V(\mathsf{V}) = 0$ all the time.

a. Initialize all V- and Q-values to zero.

b. Apply the update rule of value iteration (equation 1 on slide 3) to all cities in parallel. Hint 1: keep $V(\mathsf{V}) = 0$ and don't worry, if you find e.g. $V(\mathsf{Z}) = -2$. Hint 2: "In parallel" means, that you should assume $V(s') = 0$ when running this step for the first time and take the value that you obtained in the previous iteration otherwise.

c. Repeat step b. until convergence.

d. Convince yourself that value iteration found the optimal solution. Write down, how you convinced yourself that the optimal solution was found.

Solution:

a. V(Z) = 0, V(P) = 0, V(N) = 0, V(F) = 0, V(M) = 0, V(R) = 0, V(L) = 0, V(V) = 0, V(B) = 0

b. V(Z) = -2, V(P) = -7, V(N) = -3, V(F) = -4, V(M) = -3, V(R) = -4, V(L) = -2, V(V) = 0, V(B) = -4

c. V(Z) = -4, V(P) = -11, V(N) = -6, V(F) = -6, V(M) = -5, V(R) = -8, V(L) = -4, V(V) = 0, V(B) = -8
V(Z) = -6, V(P) = -15, V(N) = -8, V(F) = -7, V(M) = -7, V(R) = -11, V(L) = -6, V(V) = 0, V(B) = -12
V(Z) = -7, V(P) = -19, V(N) = -10, V(F) = -7, V(M) = -9, V(R) = -13, V(L) = -8, V(V) = 0, V(B) = -15
V(Z) = -7, V(P) = -22, V(N) = -12, V(F) = -7, V(M) = -9, V(R) = -15, V(L) = -9, V(V) = 0, V(B) = -17
V(Z) = -7, V(P) = -24, V(N) = -12, V(F) = -7, V(M) = -9, V(R) = -15, V(L) = -9, V(V) = 0, V(B) = -19
V(Z) = -7, V(P) = -24, V(N) = -12, V(F) = -7, V(M) = -9, V(R) = -15, V(L) = -9, V(V) = 0, V(B) = -19

d. We can manually check some values. For example, the shortest path from Lausanne to Vienna goes through Zurich and costs -9, which is the final value that was found by value iteration.

Exercise 2. Exploration in MCTS

In this exercise we will have a closer look at the exploration term in Monte Carlo Tree Search. Assume that in state s_0 there are two actions a_1 and a_2. Their true values would be $\mu(s_0, a_1) = 0.8$ and $\mu(s_0, a_2) = 2/3$, but this is unknown to the learner. So far, in the expansion of the tree in MCTS, each action was taken 5 times. By an unlucky coincidence, action a_1 always led to a loss, whereas action a_2 always led to a win, i.e. the current Q-values are $Q(s_0, a_1) = 0$ and $Q(s_0, a_2) = 1$.

a. Assume that the exploration term is of the form $\sqrt{\frac{2}{N(s,a)}}$, i.e. it decreases with visitation counts but it does not increase with the total number of MCTS iterations. Show that in this case the action selection of MCTS $a^* = \arg\max_a Q(s, a) + \sqrt{\frac{2}{N(s,a)}}$ never considers action a_1 anymore and thus fails to discover that action a_1 would actually be better.

b. Let us now use the UCB1 exploration term $\sqrt{\frac{2\log(N(s))}{N(s,a)}}$ and assume that every third time a_2 is taken, a loss results and otherwise a win. After how many iterations will MCTS explore again action a_1?

Solution:

a. After each action has been taken 5 times, we have $N(s_0) = 10$ and $N(s_0, a_1) = N(s_0, a_2) = 5$. Therefore, $Q(s_0, a_1) + \sqrt{\frac{2}{N(s_0,a_1)}} = 0 + \sqrt{\frac{2}{5}} \approx 0.63$ and $Q(s_0, a_2) + \sqrt{\frac{2}{N(s_0,a_2)}} = 1 + \sqrt{\frac{2}{5}} \approx 1.63$. Therefore, action a_2 would be taken in the next step. In subsequent steps $Q(s_0, a_1) + \sqrt{\frac{2}{N(s_0,a_1)}}$ stays at the value $\sqrt{\frac{2}{5}}$,

whereas $Q(s_0,a_2)+\sqrt{\frac{2}{N(s_0,a_2)}}$ approaches 2/3 from above in the limit of $N(s_0,a_2)\to\infty$ (unless $Q(s_0,a_2)$ drops below $2/3-\sqrt{\frac{2}{N(s_0,a_2)}}$ after experiencing many losses in a row, by coincidence). Because $2/3>\sqrt{\frac{2}{5}}$, action a_1 is never taken again.

b. We look for the first $N(s_0,a_2)$ such that $Q(s_0,a_2)+\sqrt{\frac{2\log(N(s_0))}{N(s_0,a_2)}}<Q(s_0,a_1)+\sqrt{\frac{2\log(N(s_0))}{N(s_0,a_1)}}=\sqrt{\frac{2\log(N(s_0))}{5}}$. We know that $N(s_0)=N(s_0,a_1)+5$ (+5 for the five times action a_2 was taken) and $Q(s_0,a_2)\approx(5+2/3(N(s_0,a_1)-5))/N(s_0,a_1)$. Solving for $N(s_0,a_1)$ we find at $N(s_0,a_1)=32$ that $Q(s_0,a_2)\approx 0.72$, $\sqrt{\frac{2\log(N(s_0))}{N(s_0,a_1)}}\approx 0.47$ and $\sqrt{\frac{2\log(N(s_0))}{5}}\approx 1.20$.

Exercise 3. AlphaZero.

In this exercise you will manually compute some steps in one iteration of AlphaZero's Monte Carlo Tree Search. Assume that from the previous Monte Carlo Tree Search you have already a tree that starts at state s_0 with $N(s_0,a_1)=8, W(s_0,a_1)=4, P(s_0,a_1)=0.2$, $N(s_0,a_2)=24, W(s_0,a_2)=16, P(s_0,a_2)=0.8$. We fix $C(s)=1$.

a. Determine the action (a_1 or a_2) that AlphaZero would take in the selection step of MCTS in state s_0.

b. Is it a greedy or an exploratory action that was taken in a?

c. Update N, P, W and Q (if it changes in the backpropagation step of MCTS) under the assumption that the expansion step led to $v=0.7$.

d. Compute the probability that AlphaZero would take the actual action a_1 now.

Solution:

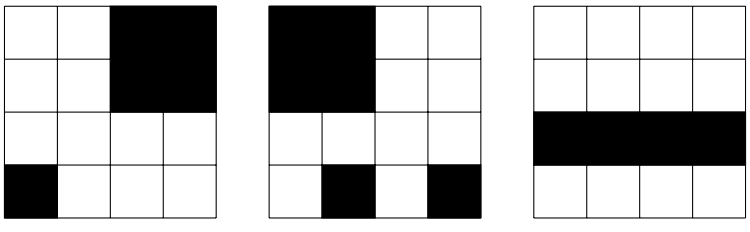
a. We need to compute the term $f(a)=Q(s_0,a)+P(s_0,a)\frac{\sqrt{N(s_0)}}{1+N(s_0,a)}$ for both actions. Using the values given in the exercise, we have $f(a_1)=\frac{4}{8}+0.2\frac{\sqrt{8+24}}{1+8}\approx 0.6257$ and $f(a_2)=\frac{16}{24}+0.8\frac{\sqrt{8+24}}{1+24}\approx 0.8477$. Therefore, action a_2 is selected.

b. It is a greedy one, since it is also equal to $\arg\max_a Q(s_0,a)$.

c. The variables N, W, and Q corresponding to action a_1 remain the same. Then we have, $N(s_0,a_2)=24+1=25$, $W(s_0,a_2)=16+0.7=16.7$, and $Q(s_0,a_2)=16.7/25\approx 0.668$. The prior action probability P remains the same, since it is only computed when s_0 was expanded for the first time.

d. The probability of the actual action is $\frac{8}{8+25}=0.24$ for a_1 and $\frac{25}{8+25}=0.76$ for a_2.

Exercise 4. MuZero



$o_1,\ldots,o_{16}$ $o_{17},\ldots,o_{32}$ $o_{33},\ldots,o_{48}$

Above you see some example images of an environment, where states $o_1,\ldots,o_{16}$ always have the black rectangle at the top right, while each pixel in the bottom row can be randomly 0 or 1. States $o_{17},\ldots,o_{32}$ have a similar pattern with a black rectangle at the top left and random pixels in the bottom row. States $o_{33},\ldots,o_{48}$ have a similar pattern with a bar in the second row from the bottom and random pixels in the bottom row. Assume the actual state transitions are $P^{a_1}_{o_i\to o_j}=1/16$ for $i\in\{1,\ldots,16\}, j\in\{17,\ldots,32\}$, $P^{a_1}_{o_i\to o_j}=1/16$ for $i\in\{17,\ldots,32\}, j\in\{33,\ldots,48\}$, $P^{a_1}_{o_i\to o_j}=1/16$ for $i\in\{33,\ldots,48\}, j\in\{1,\ldots,16\}$ and $P^{a_2}_{o_i\to o_j}=P^{a_1}_{o_j\to o_i}$. The rewards are $R^{a_1}_{o_i\to o_j}=1$ for $i\in\{1,\ldots,16\}, j\in\{17,\ldots,32\}$, all other rewards are zero. Episodes start in a random state and end after 10 actions have been taken.

a. How many bits does the latent representation s_t need to have at least for a model-based RL method that relies on a auto-encoder approach, where o_t has to be reconstructable from s_t?

b. How many bits does the latent representation s_t need to have at least for a model-based RL method like MuZero, where the latent state only needs to be sufficient for predicting the immediate reward, the value and the policy?

c. Can MuZero still find the optimal policy, if $P^{a_2}_{o_i \to o_j} = 1/32$ for $i \in \{17, \ldots, 32\}, j \in \{1, \ldots, 16, 33, \ldots 48\}$?

Solution:

a. There are 48 different states to be discriminated. Therefore at least $\log_2(48) \approx 5.6$ bits are required.

b. Immediate reward, value and optimal policy for each observation depend only on the group that observation is part of, i.e. all observations $o_1, \ldots, o_{16}$ have the same reward, value and optimal policy and the same holds for the group of observations $o_{17}, \ldots, o_{32}$ and $o_{33}, \ldots, o_{48}$. Therefore it is sufficient to have 3 latent states, or $log_2(3) \approx 1.6$ bits. Let us now see, if we can further reduce the number of latent states. Without loss of generality, we assume the function h_θ maps the observations to abstract states s_1, s_2, s_3, i.e. $h(o_i) = s_{\lfloor (i-1)/16 \rfloor}$. The dynamics in this abstract state space would be deterministic, e.g. $P^{a_1}_{s_1 \to s_2} = 1$. The optimal policy would be to take action a_1 in s_1, a_2 in s_2 and a_1 in s_3. To predict the optimal action, s_1 and s_3 could therefore be merged into one latent state, but s_2 would have to remain a separate state. For episodes of length 10, the values of the optimal policy would be $V(s_1) = V(s_2) = V(s_3) = 5$, for any starting state. To predict the optimal value, s_1, s_2, s_3 could all be merged into a single latent state. The immediate reward under the optimal policy is $R^{a_1}_{s_1 \to s_2} = 1$, all other rewards being zero. To predict the immediate reward s_2 and s_3 could be merged, but s_1 would have to remain a separate state. We conclude that s_1, s_2, s_3 have to remain separate latent states to successfully predict immediate rewards, values and policies.

c. With the latent states we found in the previous exercise we would have $P^{a_2}_{s_2 \to s_1} = 0.5$ and $P^{a_2}_{s_2 \to s_3} = 0.5$, but the transition function g_θ of MuZero is deterministic and incapable of learning stochastic dynamics. Despite the stochasticity, however, the optimal policy and the immediate rewards stay the same as in the previous exercise; only the values change. Therefore MuZero may still be able to find the optimal policy, despite not getting the dynamics right and making errors in the predictions during rollouts.