Last Name .... First Name..................

Artificial Neural Networks: Exam (with solutions)

29th of June 2023

- Keep your bag next to your chair, but do not open it during the exam.
- Write your name in legible letters on top of this page.
- The exam lasts 180 min. (The estimated times per section add up to 160 min)
- Write **all** your answers in a legible way on the exam (no extra sheets).
- No documentation is allowed (no textbook, no slides), except **one page A5 of handwritten notes, doublesided**.
- No calculator is allowed.
- Have your student card displayed in front of you on your desk.
- **Check that your exam has 10 pages.**

Evaluation:

1. .... / 5 pts (Section 1 - estimated time 20 min)
2. .... / 7 pts (Section 2 - estimated time 35 min)
3. .... / 8 pts (Section 3 - estimated time 35 min)
4. .... / 12 pts (Section 4 - estimated time 70 min)

Total: .... / 32 pts

This page remains empty. You can use it as free space for your calculations, do not use to write down answers.

Definitions and notations

RL stands for Reinforcement Learning.

The symbol η is reserved for the learning rate.

Bold face symbols refer to vectors, normal face to a single component or a single input/output. Unless noted otherwise, the input is N-dimensional: $\mathbf{x}^\mu \in R^N$

In the context of reinforcement learning, the symbol a refers to an action; the symbols r and R to a reward; the symbol s to a discrete state; and the symbol γ to a discount rate.

If the state space is continuous then states are also written as $\mathbf{x}$.

If algorithms are implemented as neural networks, the parameters are the weights w_{ij} from neuron j to neuron i. In a multilayer network an upper index (k) refers to the layer, so that we write $w_{ij}^{(k)}$. Hence $w_{nm}^{(l)}$ refers to the connection from of neuron m in layer $l-1$ to neuron n in layer l.

In many RL applications, there is a single goal state (terminal state). An action sequence that ends at the goal is called an episode.

How to give answers

All sections involve short calculations. **Please write the answers in the space provided for that purpose.**

We also provide some free space for calculations. We will not look at these parts for grading. You can ask for extra scratch paper. We will not look at the scratch paper.

1 3-step SARSA (5 points). Estimated time: 20 minutes

We approximate Q-values $Q(\mathbf{x}, a)$ by function approximation in a deep neural network with weights $w_{ij}^{(k)}$ where $i = 1, 2, \ldots$ and $j = 1, 2, \ldots$ are arbitrary neuron indices and k is the layer-index. The input $\mathbf{x}$ is continuous.

(i) Write down a reasonable loss function resulting from the consistency condition of the Bellman equation for the case of **3-step SARSA**. Consider both the BATCH version of the loss and the ONLINE version of the loss (the ONLINE version should contain only a minimal set of time steps necessary to implement SARSA updates). Choose an appropriate and understandable notation.

BATCH:

$$\text{Loss } L_{bt} = \left\langle \tfrac{1}{2}\Big(r + \gamma r' + \gamma^2 r'' + \gamma^3 Q(x''', a''') - Q(x, a)\Big)^2 \right\rangle$$

ONLINE:

$$\text{Loss } L_{on} = \tfrac{1}{2}\Big(r + \gamma r' + \gamma^2 r'' + \gamma^3 Q(x''', a''') - Q(x, a)\Big)^2$$

number of points: ....../ 2

(ii) Derive an online update rule for the specific weight $w_{nm}^{(k)}$ starting from the ONLINE Loss L_{on} in (i) using **semigradient** (we write Δ_{sg} and Δ_{fg} for updates with semigradient and full gradient, respectively)

$$\Delta_{\text{sg}} w_{nm}^{(k)} = \eta\Big(r + \gamma r' + \gamma^2 r'' + \gamma^3 Q(x''', a''') - Q(x, a)\Big)\tfrac{\partial Q(x,a)}{\partial w_{nm}^{(k)}}$$

number of points: ....../ 2

(iii) Write down the **DIFFERENCE** between the online update rule with *semi-gradient* $\Delta_{\text{sg}} w_{nm}^{(k)}$ on the ONLINE loss L_{on} in (i) and that with the *full gradient* $\Delta_{\text{fg}} w_{nm}^{(k)}$ (standard stochastic gradient) on the ONLINE loss L_{on} in (i).

DIFFERENCE

$$\Delta_{\text{sg}} w_{nm}^{(k)} - \Delta_{\text{fg}} w_{nm}^{(k)} = \eta\Big(r + \gamma r' + \gamma^2 r'' + \gamma^3 Q(x''', a''') - Q(x, a)\Big)\gamma^3 \tfrac{\partial Q(x''',a''')}{\partial w_{nm}^{(k)}}$$

number of points: ....../ 1

Space for your calculations, not for answers

2 Policy Gradient for 1-step horizon: Contextual Bandit Problem (7 points). Estimated time 35 min

The agent (bandit) can choose amongst N actions $a = 1, 2, \ldots, N$. Inputs are denoted by $\mathbf{x}$. Action $a = i$ (with $1 \leq i \leq N$) is chosen with probability

$$\pi(a = i|\mathbf{x}; \mathbf{w}_1, \ldots, \mathbf{w}_N) = \frac{[\sum_{k=1}^{10} w_{ik} y_k]^\beta}{\sum_j [\sum_{k=1}^{10} w_{jk} y_k]^\beta}, \tag{1}$$

where $\beta \in \{2, 4, 6, \ldots\}$ is a metaparameter, $\mathbf{w}_i = (w_{i1}, \ldots, w_{i10})$ is the weight vector for action $a = i$, and $y_k = f(\mathbf{x} - \mathbf{c}_k)$ with $1 \leq k \leq 10$ is a set of ten basis functions. The agent reacts to an input $\mathbf{x}$ with an action a and receives a deterministic reward $r(\mathbf{x}, a)$. At each time step a new input $\mathbf{x}$ is drawn from a stationary distribution $P(\mathbf{x})$.

(i) Write down the expected reward $E[r(\mathbf{x}, a)|\, \mathbf{w}_1, \ldots, \mathbf{w}_N]$ for the policy with parameters $\mathbf{w}_1, \ldots, \mathbf{w}_N$:

$E[r(\mathbf{x}, a)|\, \mathbf{w}_1, \ldots, \mathbf{w}_N] = \int p(\mathbf{x}) \sum_{i=1}^N \pi(a = i) r(\mathbf{x}, i) d\mathbf{x}$

number of points: ....../ [1]

(ii) What is the stochastic online version of the gradient $E[r(\mathbf{x}, a)|\mathbf{w}_1, \ldots, \mathbf{w}_N]$ with respect to a weight w_{nm}? Write down the result.

$\Delta w_{nm} = \eta r(\mathbf{x}, i) \frac{\partial \ln \pi(a=i)}{\partial w_{nm}} = \eta r(\mathbf{x}, i) \frac{\beta y_m}{\sum_k w_{nk} y_k} [\delta_{n,i} - \pi(a = n)]$ with i the selected action

number of points: ....../ [2]

(iii) Show that in expectation your online version in (ii) leads back to correct gradient of $E[r(\mathbf{x}, a)|\mathbf{w}_1, \ldots, \mathbf{w}_N]$

$$\begin{aligned}
\mathbb{E}\left[r(\mathbf{x}, a) \frac{\partial \ln \pi(a)}{\partial w_{nm}}\right] &= \int p(\mathbf{x}) \sum_{i=1}^N \pi(a = i) r(\mathbf{x}, i) \frac{\partial \ln \pi(a = i)}{\partial w_{nm}} d\mathbf{x} \\
&= \int p(\mathbf{x}) \sum_{i=1}^N \pi(a = i) r(\mathbf{x}, i) \frac{1}{\pi(a = i)} \frac{\partial \pi(a = i)}{\partial w_{nm}} d\mathbf{x} \\
&= \int p(\mathbf{x}) \sum_{i=1}^N r(\mathbf{x}, i) \frac{\partial \pi(a = i)}{\partial w_{nm}} d\mathbf{x} = \frac{\partial \mathbb{E}[r(\mathbf{x}, a)]}{\partial w_{nm}}
\end{aligned}$$

number of points: ....../ [2]

(iv) Take your result from (ii), separate a common factor $\eta\beta/(\sum_k w_{nk} y_k)$ from the remaining terms, and use Eq. (1) to find an elegant and compact formula for the update of the weight w_{nm}:

$\Delta w_{nm} = \frac{\eta\beta}{\sum_k w_{nk} y_k} y_m r(\mathbf{x}, i) [\delta_{n,i} - \pi(a = n)]$

number of points: ....../ [2]

3 1-step and 2-step Q-learning in the tabular setting (8 points). Estimated time: 35 min

In class we have discussed the SARSA algorithm and shown that, if the Q-values have converged, the resulting set of Q-values solves in expectation the Bellman equation. Here the task is to do a related calculation for Q-learning.

The agent uses an epsilon-greedy policy. The update step is

$$\Delta Q(s_t, a_t) = \eta \left[r_t + \gamma max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right] \tag{2}$$

The update step is applied after *every* action.

(i) Assume that the Q-learning algorithm has converged in expectation, i.e., the expecation of the update step is $E[\Delta Q(s_t, a_t)] = 0$. Show that $E[\Delta Q(s, a)] = 0$ for all pairs (s, a) implies that the set of Q-values solves the optimal Bellman equation.

(1) $E\left[\Delta Q(s_t, a_t)|s_t, a_t\right] = 0$

(2) $E\left[Q(s_t, a_t)|s_t, a_t\right] = E\left[r_t + \gamma \max_{a'} Q(s_{t+1}, a')|s_t, a_t\right]$

(3) $Q(s_t, a_t) = E\left[r_t + \gamma \max_{a'} Q(s_{t+1}, a')|s_t, a_t\right]$

(4) $Q(s_t, a_t) = \sum_{s'} p(s'|s_t, a_t)\left[R^{a_t}_{s_t \to s'} + \gamma \max_{a'} Q(s', a')\right]$

Line (4) is the final result. For the transitions from lines (1) to (2) to (3) to (4) I used the following identities or arguments:

(1) to (2) Equation 2.

(2) to (3) $Q(s_t, a_t)$ is constant conditioned on s_t and a_t.

(3) to (4) Definitions of expectation and the expected reward.

number of points: ....../ [2]

(ii) So far we have looked at the standard version of Q-learning which we call Variant A_1 in the following. Your friend Bob proposes an alternative (Variant B) of the algorithm as follows: As before we run an epsilon-greedy policy, but to be sure that we remain optimal, we apply the update step (given above in Eq. (2)) only if the *best* action is chosen at time t, i.e., $a_t = a^*(s_t) = max_{a'} Q(s_t, a')$. Your friend Alice says that convergence of Q-values $E[\Delta Q(s, a)] = 0$ for all pairs (s, a) in Variant B **does not guarantee** that the set of Q-values is consistent with the Bellman equation.

Is Alice right? Give a mathematical argument or an intuitive interpretation.

Alice is right! In variant B, the argument for going from (1) to (2) in our proof above does not hold because $E\left[\Delta Q(s_t, a_t)|s_t, a_t\right] = 0$ can also imply that $a_t \neq a^*(s_t)$.

number of points: ....../ [2]

(iii) Now we repeat the same arguments for 2-step Q-learning. Again we consider two variants.

In variant A$_2$ the update step is applied after *every* action (for all time steps $t > 2$). Your friend Claire proproses an alternative (Variant C) where the update rule is applied only if the epsilon-greedy algorithm chooses the greedy action for the intermediate step of the algorithm, i.e. $a_{t+1} = a^*(s_{t+1}) = max_{a'} Q(s_{t+1}, a')$.

Your friend Dominique claims that (at least) one of the Variants A$_2$ or C is consistent with the Bellman equation. To check whether Dominique is correct, let us assume that the 2-step Q-learning algorithm has converged in expectation, i.e., the expecation of the update step is $E[\Delta Q(s_t, a_t)] = 0$.

Choose either variant A$_2$ or C (whichever works best for your arguments) and show that $E[\Delta Q(s_t, a_t)] = 0$ implies that the set of Q-values solves the two-step Bellman equation. Write down the two-step algorithm, then three steps of the argument and then your final result:

Variant C: $\Delta Q(s,a) = \eta\, \mathbb{1}_{\{a'=a^*(s')\}} \delta$ with $\delta = [r + \gamma r' + \gamma^2 \max_{a''} Q(s'', a'') - Q(s,a)]$.

$E[\Delta Q(s,a)|s,a] = \eta \sum_{r,s',a',r',s''} p(r,s'|s,a)\pi(a'|s')p(r',s''|s',a')\mathbb{1}_{\{a'=a^*(s')\}}\delta = 0 \Rightarrow$

$\sum_{r,s',r',s''} p(r,s'|s,a) \underbrace{\pi(a^*(s')|s')}_{1-\epsilon=\text{constant}} p(r',s''|s',a^*(s'))\delta = 0 \Rightarrow$

$\sum_{r,s',r',s''} p(r,s'|s,a)p(r',s''|s',a^*(s'))\delta = 0 \Rightarrow$

$Q(s,a) = \sum_{s'} p(s'|s,a) \left(R^a_{s\to s'} + \gamma \sum_{s''} p(s''|s',a^*(s')) \left(R^{a^*(s')}_{s'\to s''} + \gamma^2 Q(s'', a^*(s''))\right)\right)$

number of points: ....../ [3]

(iv) Would the mathematical calculation also work for the other variant? The answer (yes/no) needs to be supported by a mathematical argument or an intuitive reason.

No, it does not work for variant A$_2$ because a' and a'' in the update rule correspond to two different policies (a' to ϵ-greedy and a'' to greedy).

number of points: ....../ [1]

4 Importance Sampling in an n-step actor-critic model: V-Trace approach (12 points). Estimated time: 70 min.

Background information, not necessary to solve the exercise: The V-Trace Actor-Critic Method is an important part of the popular IMPALA agent. It is an *off-policy* modification of an n-step actor-critic policy gradient method. In this section we analyze aspects of this practically relevant algorithm.

Problem setting: We consider trajectories as sequences of states $\mathbf{x}_t, \mathbf{x}_{t+1}, \mathbf{x}_{t+2}, \ldots$ and actions $a_t, a_{t+1}, a_{t+2}, \ldots$ that are generated by an actor following some policy $\mu(a|\mathbf{x})$. We are interested in the V-values $V(\mathbf{x})$ that correspond to another policy $\pi(a|\mathbf{x})$. The state at time t is denoted by $\mathbf{x}_t$; the state at time s is denoted by $\mathbf{x}_s$.

The critic is optimized by semi-gradient on a loss function $L_{on} = (V(\mathbf{x}_s) - v_s)^2$ that compares, at each time point s, the V-value $V(\mathbf{x}_s)$ in state $\mathbf{x}_s$ with a target v_s. The actor is optimized by policy gradient.

V-trace target: We define the n-step 'V-Trace target' for $V(\mathbf{x}_s)$ as

$$v_s := V(\mathbf{x}_s) + \sum_{t=s}^{s+n-1} \gamma^{t-s} \left(\prod_{i=s}^{t-1} c_i \right) \delta_t \tag{3}$$

where

$$\delta_t := [r_t + \gamma V(\mathbf{x}_{t+1}) - V(\mathbf{x}_t)]\, \rho_t \tag{4}$$

is a temporal difference and $\rho_t := \min\left(\bar{\rho}, \frac{\pi(a_t|\mathbf{x}_t)}{\mu(a_t|\mathbf{x}_t)}\right)$ and $c_i := \min\left(\bar{c}, \frac{\pi(a_i|\mathbf{x}_i)}{\mu(a_i|\mathbf{x}_i)}\right)$ are truncated importance sampling weights with truncation levels $\bar{\rho}$ and $\bar{c}$.

Note: To simplify the writing, we make use of the notation $\prod_{i=s}^{t-1} c_i = 1$ for $s = t$.

(i) We start with the on-policy case, assuming $\bar{\rho} = \bar{c} = 1$. Rewrite the V-Trace target v_s defined by Eq. 3 in terms of $V(\mathbf{x}_{s+n})$ and intermediate rewards $r_s, r_{s+1}, \ldots r_{s+n}$.

Hint1: After the rewrite, $V(\mathbf{x}_s)$ should disappear on the righ-hand-side of Eq. 3.

Hint2: on-policy means $\mu = \pi$.

$v_s = \left(\sum_{t=s}^{s+n-1} \gamma^{t-s} r_t\right) + \gamma^n V(\mathbf{x}_{s+n})$

number of points: ....../ 3

Space for your calculations, not for answers

(ii) We turn to the critic and study the off-policy scenario. Hence the policy $\pi \neq \mu$.

Expectation values will be denoted by E_μ if the expection is over trajectories generated by the policy μ and E_π if the expection is over trajectories generated by the policy π .

For the calculation we assume $\bar{\rho} = \bar{c} > \max_t \frac{\pi(a_t|\mathbf{x}_t)}{\mu(a_t|\mathbf{x}_t)}$.

Given the current state $\mathbf{x}_s$, relate the Expectiation E_μ of V-Trace target v_s from Eq. 3 to the Expectation E_π of V-values and, possibly, rewards.

Show the main steps of your mathematical calculation and write the final result in the last line.

Hint: You start with $E_\mu[v_s|\mathbf{x}_s]$ on the left-hand side; and end with an expectation $= E_\pi[\dots]$ over an appropriate expression on the right-hand side.

$$E_\mu[v_s|\mathbf{x}_s] = E_\mu\left[V(\mathbf{x}_s)\middle|\mathbf{x}_s\right] + \sum_{t=s}^{s+n-1} \gamma^{t-s} E_\mu\left[\prod_{i=s}^{t} \frac{\pi(a_i|\mathbf{x}_i)}{\mu(a_i|\mathbf{x}_i)} \left(r_t + \gamma V(\mathbf{x}_{t+1}) - V(\mathbf{x}_t)\right) \middle|\mathbf{x}_s\right]$$

$$= E_\pi\left[V(\mathbf{x}_s)\middle|\mathbf{x}_s\right] + \sum_{t=s}^{s+n-1} \gamma^{t-s} E_\pi\left[\left(r_t + \gamma V(\mathbf{x}_{t+1}) - V(\mathbf{x}_t)\right)\middle|\mathbf{x}_s\right]$$

$$E_\mu[v_s|\mathbf{x}_s] = E_\pi\left[\sum_{k=0}^{n-1} \gamma^k r_{s+k}\middle|\mathbf{x}_s\right] + \gamma^n E_\pi\left[V(\mathbf{x}_{s+n})\middle|\mathbf{x}_s\right]$$

number of points: ....../ 4

Space for your calculations, not for answers

Exam continues on next page

(iii) We now turn to the actor. We consider the term

$$A := \rho_s \nabla_w \log \pi_w(a_s|\mathbf{x}_s)(r_s + \gamma v_{s+1} - V(\mathbf{x}_s)) \tag{5}$$

and study its expectation $E_\mu[A|\mathbf{x}_s]$. We assume $\bar{\rho} = \bar{c} > \max_t \frac{\pi(a_t|\mathbf{x}_t)}{\mu(a_t|\mathbf{x}_t)}$.

Show that $E_\mu[A|\mathbf{x}_s]$ is equal to the expection E_π of the policy-gradient rule of the **on-policy model**, i.e., of an agent with policy π.

Show the main steps of your mathematical arguments:

(1) $E_\mu[A|\mathbf{x}_s] = E_{a_s\sim\mu}\left[\frac{\pi(a_s|\mathbf{x}_s)}{\mu(a_s|\mathbf{x}_s)}\nabla_w \log \pi_w(a_s|\mathbf{x}_s)(E_\mu[r_s + \gamma v_{s+1}|a_s, \mathbf{x}_s] - V(\mathbf{x}_s))\middle|\mathbf{x}_s\right]$

(2) $= E_{a_s\sim\pi}\left[\nabla_w \log \pi_w(a_s|\mathbf{x}_s)(E_\mu[r_s + \gamma v_{s+1}|a_s, \mathbf{x}_s] - V(\mathbf{x}_s))\middle|\mathbf{x}_s\right]$

(3) $= E_{a_s\sim\pi}\left[\nabla_w \log \pi_w(a_s|\mathbf{x}_s)\left(E_\pi\left[r_s + \sum_{k=0}^{n-1}\gamma^{k+1}r_{s+1+k} + \gamma^{n+1}V(\mathbf{x}_{s+1+n})\middle|a_s, \mathbf{x}_s\right] - V(\mathbf{x}_s)\right)\middle|\mathbf{x}_s\right]$

(4) $= E_\pi\left[\nabla_w \log \pi_w(a_s|\mathbf{x}_s)\left(\sum_{k=0}^{n}\gamma^k r_{s+k} + \gamma^{n+1}V(\mathbf{x}_{s+n+1}) - V(\mathbf{x}_s)\right)\middle|\mathbf{x}_s\right]$

Line 4, is the final result. For the transitions from line 1 to 2 to 3 to 4, I used the following identities:

(1) to (2) Absorbing $\frac{\pi(a_s|\mathbf{x}_s)}{\mu(a_s|\mathbf{x}_s)}$ to change the outer expecation of a_s from μ to π.

(2) to (3) Using part ii to rewrite $E_\mu[v_s|\mathbf{x}_s, a_s]$ as an expecation with respect to π.

(3) to (4) Combining the two expecation using the law of total expecations.

number of points: ....../ 4

(iv) Why are the findings in parts (ii) and (iii) important for practical applications?

1. They allow training policy π with samples generated by another policy μ.
2. They allow using replay buffer in on-policy methods (but with re-weighting the samples).
3. The idea is similar to PPO.

number of points: ....../ 1

Space for your calculations, not for answers