# Artificial Neural Networks (Gerstner). Solutions for week 4

## Policy Gradient

### Exercise 1. (in Class): Single neuron as an actor

Assume an agent with binary actions $Y \in \{0,1\}$. Action $y = 1$ is taken with a probability $\pi(Y = 1|\vec{x};\vec{w}) = g(\vec{w}\cdot\vec{x})$, where $\vec{w}$ are a set of weights and $\vec{x}$ is the input signal that contains the state information. The function $g$ is monotonically increasing and limited by the bounds $0 \leq g \leq 1$.

For each action, the agent receives a reward $R(Y,\vec{x})$.

a. Calculate the gradient of the mean reward $\langle R\rangle = \sum_{Y,\vec{x}} R(Y,\vec{x})\pi(Y|\vec{x};\vec{w})P(\vec{x})$ with respect to the weight $w_j$.

   Hint: Insert the policy $\pi(Y = 1|\vec{x};\vec{w}) = g(\sum_k w_k x_k)$ and $\pi(Y = 0|\vec{x};\vec{w}) = 1 - g(\sum_k w_k x_k)$. Then take the gradient.

b. The rule derived in (a) is a batch rule. Can you transform this into an 'online rule'?

   Hint: Pay attention to the following question: what is the condition that we can simply 'drop the summation signs'?

**Solution:**

a. $\frac{d}{dw_j}\langle R\rangle = \sum_{\vec{x}} P(\vec{x})[R(y = 1,\vec{x}) - R(y = 0,\vec{x})]g' x_j$

b. If the online statistics matches the true statistics of the data in the batch, then we can drop the sum-signs. However, here this is not the case because the two outcomes $y = 1$ and $y = 0$ do not have equal probabilities. Therefore, the weight-factors in y need to be added. This can be done by the log-likelihood trick explained in class.

### Exercise 2. Subtracting the mean

You have two stochastic variables, $x$ and $y$ with means $\langle x\rangle$ and $\langle y\rangle$. Angles denote expectations. We are interested in the product $z = (x - b)(y - \langle y\rangle)$ with a fixed parameter $b$.

a. Show that $\langle z\rangle$ is independent of the choice of the parameter $b$.

b. Show that $\langle z^2\rangle$ is minimal if $b = \frac{\langle xf(y)\rangle}{\langle f(y)\rangle}$, where $f(y) = (y - \langle y\rangle)^2$.

   Hint: write $\langle z^2\rangle = F(b)$ and set $dF/db = 0$.

c. What is the optimal $b$, if $x$ and $f(y)$ are approximately independent?

d. Make the connection to policy gradient rules.

   Hint: take $x = r$ (reward) and $y$ the action taken in state $s$. Compare with the policy gradient formula of the simple 1-neuron actor. What can you conclude for the best value of $b$? Consider different states $s$. Why should $b$ depend on $s$?

**Solution:**

a.

$$\begin{aligned}\langle z\rangle &= \langle (x-b)(y-\langle y\rangle)\rangle &(1)\\ &= \langle xy\rangle - \langle x\rangle\langle y\rangle - b\langle y\rangle + b\langle y\rangle &(2)\\ &= \langle xy\rangle - \langle x\rangle\langle y\rangle &(3)\end{aligned}$$

b.

$$\begin{aligned}F(b) &= \big\langle (x-b)^2 f(y)\big\rangle &(4)\\ \Rightarrow 0 = \frac{d}{db}F(b) &= -2\,\langle (x-b)f(y)\rangle &(5)\\ \Rightarrow 0 &= \langle xf(y)\rangle - b\langle f(y)\rangle &(6)\\ \Rightarrow b &= \frac{\langle xf(y)\rangle}{\langle f(y)\rangle} &(7)\end{aligned}$$

c. If $x$ and $f(y)$ are approximately independent, $\langle xf(y)\rangle \approx \langle x\rangle\langle f(y)\rangle$ and we find $b \approx \langle x\rangle$.

d. If we set $r = x$ and introduce states $s$ as a further stochastic variable, we see that $y - \langle y\rangle$ appears in the derivative of the log-policy (e.g. for a Gaussian policy $\frac{d}{dw}\log\left((1/\sqrt{2\pi})\exp(-(y-ws)^2/2)\right) = (y-ws)s$ with $ws = \langle y\rangle$; see also next exercise), and thus $(r-b)(y-\langle y\rangle) \propto (r-b)\frac{d}{dw}\log\pi(y|s;w) = \frac{d}{dw}R(y,s)$. Since $r$ and $y$ are now state dependent, the optimal baseline should also be state-dependent.

**Exercise 3. Policy gradient**

a. **Policy gradient for binary actions**: Find an online policy gradient rule for the weights $\vec{w}$ for the same setup as in exercise 1 by calculating the gradient of the log-likelihood $\log\pi(Y|\vec{x};\vec{w})$ with respect to the weights. Hint: the policy $\pi$ can be written as $\pi(Y|\vec{x};\vec{w}) = (1-\rho)^{1-Y}\rho^Y$ with $\rho = g(\vec{w}\cdot\vec{x})$.

b. **Other parameterizations**: What happens to the policy gradient rule in exercise 3.a if the likelihood $\rho$ of action 1 is parameterized not by the weights $\vec{w}$ but by other parameters: $\rho = \rho(\theta)$? Derive a learning rule for $\theta$.

c. **Generalization to the natural exponential family**: The natural exponential family is a family of probability distributions that is widely used in statistics because of its favorable properties. These distributions can be written in the form

$$p(Y) = h(Y)\exp\left(\theta Y - A(\theta)\right)\,. \tag{8}$$

This family includes many of the standard probability distributions. The Bernoulli, the Poisson and the Gaussian distribution (with fixed variance) are all member of this family. A nice property of these distributions is that the mean can easily be calculated from the function $A(\theta)$:

$$E[Y] = A'(\theta)\,. \tag{9}$$

Assume that the policy $\pi(Y|\vec{x};\theta)$ is an element of the natural exponential family. Show that the online rule for the policy gradient has the shape:

$$\Delta\theta = R(Y - E[Y])\,. \tag{10}$$

Can you give an intuitive interpretation of this learning rule?

**Solution:**

a. **Policy gradient for binary actions**: Let's first calculate the derivative of $\log\pi(Y|\vec{x};\vec{w})$ with respect to $w_j$, using the hint:

$$\begin{aligned}
\frac{d}{dw_j}\log\pi(Y|\vec{x};\vec{w}) &= \frac{1}{\pi(Y|\vec{x};\vec{w})}\frac{d}{dw_j}\pi(Y|\vec{x};\vec{w}) \\
&= \frac{1}{(1-\rho)^{1-Y}\rho^Y}\frac{d}{dw_j}\left[(1-\rho)^{1-Y}\rho^Y\right] \\
&= \frac{1}{(1-\rho)^{1-Y}\rho^Y}\left[-(1-Y)(1-\rho)^{-Y}\rho^Y + Y(1-\rho)^{1-Y}\rho^{Y-1}\right]\frac{d}{dw_j}\rho \\
&= \left[-\frac{(1-Y)(1-\rho)^{-Y}}{(1-\rho)^{1-Y}} + \frac{Y\rho^{Y-1}}{\rho^Y}\right]g'(\vec{w}\cdot\vec{x})x_j \\
&= \left[-\frac{(1-Y)}{(1-\rho)} + \frac{Y}{\rho}\right]g'(\vec{w}\cdot\vec{x})x_j.
\end{aligned}$$

Now let's consider the term $\frac{d}{dw_j}\langle R\rangle$ again. We can write

$$\begin{aligned}
\frac{d}{dw_j}\langle R\rangle &= \sum_{Y,\vec{x}} R(Y,\vec{x})\frac{d}{dw_j}\pi(Y|\vec{x};\vec{w})P(\vec{x}) \\
&= \sum_{Y,\vec{x}} R(Y,\vec{x})\pi(Y|\vec{x};\vec{w})\underbrace{\frac{1}{\pi(Y|\vec{x};\vec{w})}\frac{d}{dw_j}\pi(Y|\vec{x};\vec{w})}_{\frac{d}{dw_j}\log\pi(Y|\vec{x};\vec{w})}P(\vec{x}) \\
&= \langle R\frac{d}{dw_j}(\log\pi)\rangle\,,
\end{aligned}$$

where we multiplied by $\pi(\cdot)/\pi(\dot{)} = 1$ and identified the derivative of the log. This suggest an online rule with an update term:

$$\Delta w_j = R\frac{d}{dw_j}\log\pi(Y|\vec{x};\vec{w}) = R\left[-\frac{(1-Y)}{(1-\rho)} + \frac{Y}{\rho}\right]g'(\vec{w}\cdot\vec{x})x_j. \tag{11}$$

b. **Other parameterizations**: Replacing $\vec{w}$ by $\theta$, we can follow the same development as in 3.a. The only difference comes in the expression of $\frac{d\rho}{d\theta}$, for which we don't have an explicit expression anymore. The learning rule is:

$$\Delta\theta = R\left[-\frac{(1-Y)}{(1-\rho)} + \frac{Y}{\rho}\right]\rho'(\theta).$$

c. **Generalization to the natural exponential family**: Let's calculate $\frac{d}{d\theta}\log p(Y)$:

$$\begin{aligned}\frac{d}{d\theta}\log p(Y) &= \frac{d}{d\theta}\log\left[h(Y)\exp\left(\theta Y - A(\theta)\right)\right]\\ &= \frac{1}{h(Y)\exp\left(\theta Y - A(\theta)\right)}\cdot h(Y)\exp\left(\theta Y - A(\theta)\right)\cdot\left(Y - A'(\theta)\right)\\ &= Y - A'(\theta) = (Y - E[Y]).\end{aligned}$$

With that simple expression, the online rule of Eq. (11) becomes:

$$\Delta\theta = R\frac{d}{d\theta}\log P(y) = R(Y - E[Y])\,. \tag{12}$$

This learning rule will look for correlation between the reward and the deviations of $Y$ from its expectation value. If $R$ is systematically *higher* when $Y$ is higher than its expectation value, theta will increase, leading to higher probabilities of higher $Y$. Inversely, if $R$ is systematically lower when $Y$ is higher than its expectation value, theta will decrease and the probability of lower $Y$ will decrease.

**Exercise 4. Debugging of RL algorithms**

You work with an implementation of 2-step SARSA and have doubts whether your algorithm performs correctly.

You have 2 possible actions from each state. You read-out the values after $n$ episodes and find the following values:

$Q(1,a1) = 0$, $Q(2,a1) = 5$ $Q(3,a1) = 3$ $Q(4,a1) = 4$ $Q(5,a1) = 6$ $Q(6,a1) = 12$ $Q(7,a1) = 10$ $Q(8,a1) = 11$ $Q(9,a1) = 9$ $Q(10,a1) = 10$

$Q(1,a2) = 1$, $Q(2,a2) = 1$ $Q(3,a2) = 3$ $Q(4,a2) = 2$ $Q(5,a2) = 1$ $Q(6,a2) = 4$ $Q(7,a2) = 2$ $Q(8,a2) = 6$ $Q(9,a2) = 11$ $Q(10,a1) = 10$

You run one episode and observe the following sequence (state, action, reward)

$(1,a2,1)$ $(2,a2,1)$ $(3,a1,0)$ $(5,a1,4)$ $(6,a1,1)$ $(8,a2,1)$

What are the updates of 2-step SARSA that the algorithm should produce?

**Solution:**

The update algorithm for 2-step SARSA is

$$\Delta Q(s_t,a_t) = \alpha(r_{t+1} + \gamma r_{t+2} + \gamma^2 Q(s_{t+2},a_{t+2}) - Q(s_t,a_t)) \tag{13}$$

with step size/learning rate $\alpha$ and discount factor $\gamma$. As a result, the update for the episode above should be

$$\begin{aligned}\Delta Q(1,a2) &= \alpha(1 + 1\gamma + 3\gamma^2 - 1)\\ \Delta Q(2,a2) &= \alpha(1 + 0\gamma + 6\gamma^2 - 1)\\ \Delta Q(3,a1) &= \alpha(0 + 4\gamma + 12\gamma^2 - 3)\\ \Delta Q(5,a1) &= \alpha(4 + 1\gamma + 6\gamma^2 - 6)\\ \Delta Q(6,a1) &= \alpha(1 + 1\gamma - 12)\\ \Delta Q(8,a2) &= \alpha(1 - 6).\end{aligned}$$

Here, we use the fact that no rewards can be received after the episode ends to truncate the summation. This can be thought of as a special "terminal" state at the end of each episode, that always transitions into itself with reward 0, and all Q–values equal to 0.

**Exercise 5. Analysis of RL algorithms**

Your friend proposes the following algorithm, using the pseudocode convention of Sutton and Barto.

Initialize $Q(s,a) = 0$ for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $\pi$ to be $\varepsilon$-greedy
Parameters: step size $\alpha \in (0,1]$, small $\varepsilon > 0$
All store and access operations (for $S_t$, $A_t$, and $R_t$) can take their index mod $4$

Repeat (for each episode):
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$
  $T \leftarrow 10000$
  For $t = 0, 1, 2, \ldots$ :
  | If $t < T$, then:
  |   Take action $A_t$
  |   Observe and store the next reward as $R_{t+1}$ and the next state as $S_{t+1}$
  |   If $S_{t+1}$ is terminal, then:
  |     $T \leftarrow t+1$
  |   else:
  |     Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
  | $\tau \leftarrow t - 3$
  | If $\tau \geq 0$:
  |   $X \leftarrow \sum_{i=\tau+1}^{\min(\tau+4,T)} \gamma^{i-\tau-1} R_i$
  |   If $\tau + 4 < T$, then $X \leftarrow X + \gamma^4 Q(S_{\tau+4}, A_{\tau+4})$
  |   $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\,[X - Q(S_\tau, A_\tau)]$
  |
  Until $\tau = T - 1$

a. Is the algorithm On-Policy or Off-Policy?

   Answer: ..........

b. What does the variable X represent?

   Answer .............

c. Is this algorithm novel, similar to, or equivalent to an existing algorithm?

   Answer (fill in/choose)

   This algorithm is identical/very similar to .... .

   There is no difference to the named algorithm/the main difference is ....

d. Is this algorithm a TD algorithm? What is the reason for your answer?

   Answer: Yes/No, because ....

**Solution:**

a. Is the algorithm On-Policy or Off-Policy?

   The algorithm is On–Policy. In the third–to–last line, the value is bootstrapped using the Q–value estimate $Q(s_{t+4}, a_{t+4})$, i.e. the action that was taken in state $s_{t+4}$ according to the agent's actual policy.

b. What does the variable X represent?

The variable X represents the 4–step truncated discounted returns. That is, X is a sample from the distribution over the returns that the agent can expect from taking action $A_\tau$ in state $S_\tau$; the agent estimates the mean of this distribution with $Q(S_\tau, A_\tau)$.

The agent gets this sample using the actual (discounted) rewards observed in the episode over the first 4 steps, plus an estimate of the average discounted returns from step 5 onwards (given by $\gamma^4 Q(S_{\tau+4}, A_{\tau+4})$).

c. Is this algorithm novel, similar to, or equivalent to an existing algorithm?

The algorithm is equivalent to 4–step SARSA, which itself is very similar to the more commonly used 1–step SARSA.

d. Is this algorithm a TD algorithm? What is the reason for your answer?

The algorithm is a TD algorithm because it uses bootstrapping (updating estimates from other, later estimates) to estimate the target (the Q–value function).