CPSC 533V Assignment 1 Due Wed Jan 31, 2024

48 points total (9% of final grade)
Submit a PDF via Canvas. The PDF can be generated from LaTeX, a scan of paper version, or a PDF assembled from images.

1. (5 points) Reinforcement Learning Notation

   Give expressions for each of the following.

   (a) The return, i.e., the future cumulative rewards, G_t, when in state s at time t, for an MDP with deterministic dynamics and a deterministic policy, discount factor γ, rewards $r_t, r_{t+1}, ...$, and an infinite horizon. Use a summation.
   $G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$

   (b) Rewrite your answer above in a recursive form, i.e., express G_t in terms of G_{t+1}.
   $G_t = r_{t+1} + \gamma G_{t+1}$

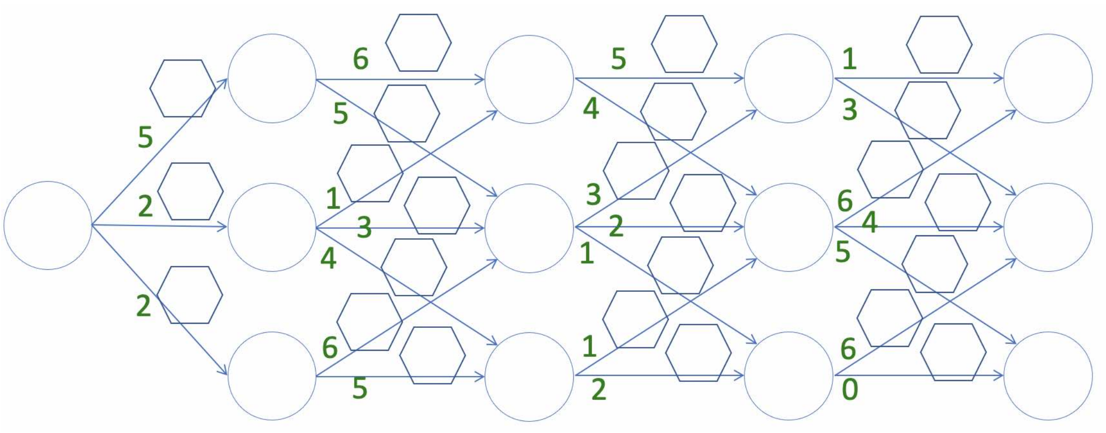
   (c) The expected future cumulative rewards, $V(s)$, when in state s at time t, for an MDP with stochastic dynamics, discount factor γ, rewards r_t, and a fixed-horizon H.
   $V(s) = E\left[\sum_{i=0}^{H-1} \gamma^i r_i | s_0 = s\right]$

   (d) The expected return from state s, if the best-available action is taken in state s, given the current value function, $V(s)$, the stochastic dynamics, $P(s'|s,a)$, the rewards $r(s,a)$, the discount factor, γ, and the use of bootstrapping. Assume discrete states and discrete actions.
   $V(s) = \max\limits_a \sum\limits_{s'} P(s'|s,a)[r(s,a) + \gamma V(s')]$

   (e) The best action to take from state s, corresponding to the above question.
   $\pi(s) = \arg\max\limits_a V(s) = \arg\max\limits_a \sum_{s'} P(s'|s,a)[r(s,a) + \gamma V(s')]$

2. (6 points) Deterministic MDP

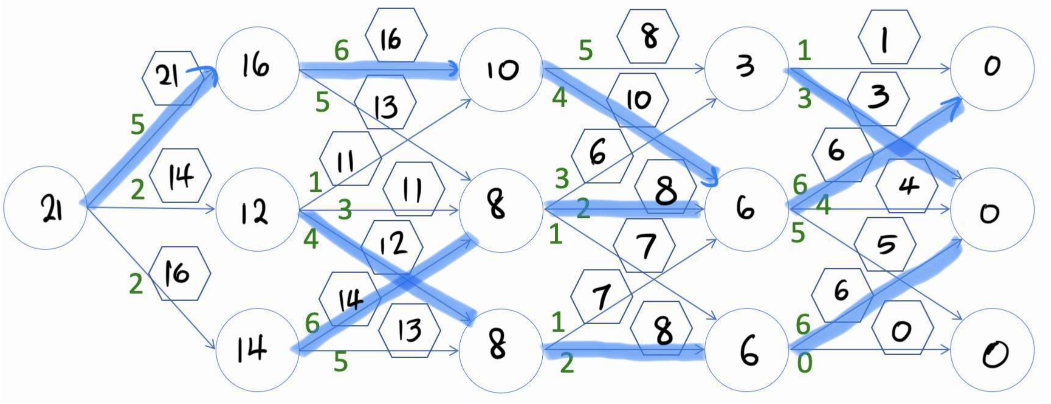
Consider the following deterministic MDP. Assume no discounting. Assume zero rewards once the final states are reached.



(a) (2 points) Compute the optimal state value function, $V(s)$, associated with each state. Write this in the circles that represent the states. Alternatively, list the values functions, in order from left-to-right, then top-to-bottom, i.e., a column-major order.

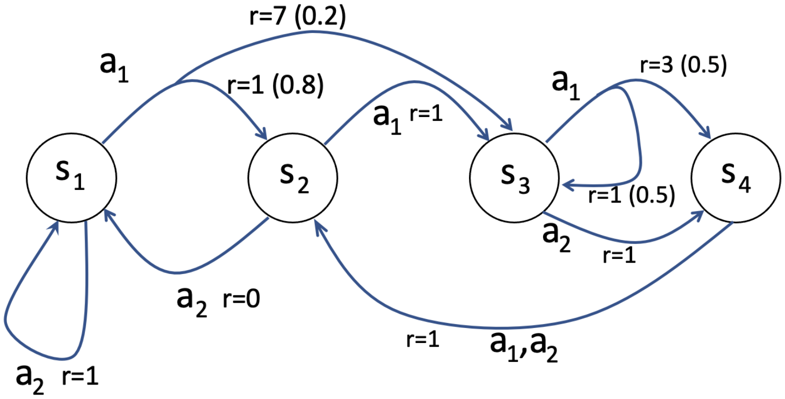
(b) (2 points) Annotate the optimal actions, using a heavier arrow. Alternatively, list the optimal actions for each state, implicitly numbered, e.g., 1,2,3, and using a column-major order for the states.

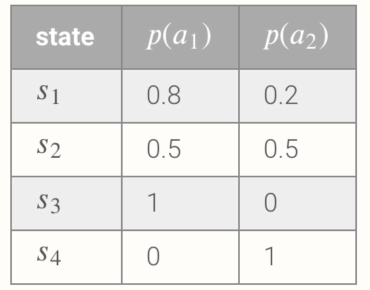
(c) (2 points) Compute the optimal state-action function, $Q(s, a)$, associated with each state. Write this in the hexagons that represent all possible action transitions. Alternatively, list these values, in order from left-to-right, then top-to-bottom, i.e., a column-major order.



3. (6 points) Stochastic MDP

Consider the following stochastic MDP, and the given stochastic policy. Assume reward discounting, with $\gamma = 0.8$, and an infinite horizon. The bracketed number on the transitions indicate the probability of taking that given transition. All other transitions are taken with probability one.





state	$p(a_1)$	$p(a_2)$
s_1	0.8	0.2
s_2	0.5	0.5
s_3	1	0
s_4	0	1

stochastic policy $\pi(s)$

(a) (3 points) Develop a transition probability matrix for the MDP, $P_{mn} : p(s_{t+1} = s_n | s_t = s_m)$ under the given policy $\pi(s)$. Element P_{mn} in column n thus represents the probability of transitioning from state m to state n. Also, develop a column vector R, where R_m is the expected immediate reward for the next transition when in state s_m, under the policy $\pi(s)$.

$$P_\pi = \begin{bmatrix} 0.2 & 0.64 & 0.16 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 1 & 0 & 0 \end{bmatrix}, R_\pi = \begin{bmatrix} 0.8 * (7 * 0.2 + 1 * 0.8) + 0.2 * (1) = 1.96 \\ 0.5 * 1 + 0.5 * 0 = 0.5 \\ 1 * (3 * 0.5 + 1 * 0.5) = 2 \\ 1 * 1 = 1 \end{bmatrix}$$

(b) (3 points) The value function for the MDP under the policy π is equal to the immediate rewards, plus the discounted sum of future rewards, i.e.,
$V(s) = E_\pi[\, r(s,a) + \gamma \sum_{s'} P(s'|s,a) V(s')\,]$
where $a = \pi(s)$. Give a linear algebra expression that is equivalent to this, using the P and R as determined previously. Then give a linear algebra expression that solves for $V(s)$. Lastly, solve for the value function using your favorite linear algebra package.

$V(s) = R_\pi(s) + \gamma \sum_{s'} P_\pi(ss')V(s')$ to matrix form: $V_\pi = R_\pi + \gamma P_\pi V_\pi$

To solve: $V_\pi = \begin{bmatrix} 1.96 \\ 0.5 \\ 2 \\ 1 \end{bmatrix} + 0.8 \begin{bmatrix} 0.2 & 0.64 & 0.16 & 0 \\ 0.5 & 0 & 0.5 & 0 \\ 0 & 0 & 0.5 & 0.5 \\ 0 & 1 & 0 & 0 \end{bmatrix} V_\pi$

Solve the value function we have $V_\pi = \begin{bmatrix} 7.3787 \\ 6.4214 \\ 7.4247 \\ 6.1371 \end{bmatrix}$

4. (6 points) Reinforcement Learning

(a) (1 point) Give three reasons why RL can be considered to be more difficult than supervised learning.

In supervised learning, labeled training data that one assumes to be representative even for unseen data is provided. Yet for RL, it is to learn from interaction and it is impractical to have such data available, the agent must learn from its own experience.

(b) (1 point) Are optimal policies unique? Why or why not?

No, not necessarily. The optimization landscape can always have more than one global optimal. For example, two policies that returns different actions under the same state can still result in the same expected return overall.

(c) (2 points) Suppose that an MDP returns a reward of 1 at every time step. What is the expected return for the MDP for a discount factor of γ, and assuming an infinite horizon? Show your work. Give the expected returns for $\gamma = \{0, 0.5, 0.9, 0.99\}$. Give a definition for the **effective horizon** that can be associated with a discount factor. What discount factor should we use if we wish to have a policy that has an effective horizon of 5 time steps into the future?

$$G_t = R_{t+1} + \gamma R_{t+1} + \gamma^2 R_{t+2} + ...$$

$$= 1 + \gamma + \gamma^2 + ...$$

Given discount factor $|\gamma| < 1$, from geometric series sum, we have

$$= \frac{1}{1-\gamma}$$

For $\gamma = \{0, 0.5, 0.9, 0.99\}$, we thus have $G_t = \{1, 2, 10, 100\}$.

One can define an effective horizon as such: if a process has γ (discount factor) probability to "die out" every timestep, the expected number of steps of such process's termination an be viewed as the effective horizon, which is $\frac{1}{1-\gamma}$ This is mathematically equivalent in terms of expected reward calculation compared to the "myopic" interpretation of discounted reward. For the effective horizon to be 5, $\gamma = 0.8$

(d) (2 points) Given an discrete-state, discrete action MDP with $|S|$ states and $|A|$ actions, give an expression for the number of possible distinct policies. How many deterministic policies are there for the *deterministic MDP* given above? How many deterministic policies are there for the *stochastic MDP* given above?

Assume all actions are available at each state, There are $|A|^{|S|}$ possible distinct deterministic policies and infinitely possible distinct stochastic policies.

For the *deterministic* MDP in question 2, it has $3 * 2 * 3 * 2 * 2 * 3 * 2 * 2 * 3 * 2 = 5184$ distinct deterministic policies.

For the *stochastic* MDP in question 3, it could have $2^4 = 16$ deterministic policies given the 4 states each with 2 actions.

5. (6 points) Math Review: Evaluate the following expressions.

$$v = \begin{bmatrix} \pi \\ e \\ 1 \end{bmatrix}, \quad w = \begin{bmatrix} 2 \\ 3 \\ -1 \end{bmatrix}, \quad A = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}, \quad \alpha = \frac{\pi}{4}$$

(a) $v^T w + \|v\|$ (2-norm)

(b) $v^T A v$

(c) $A^T A$

(d) $\min_{x \in [0,1]} \; x(1-x)$

(e) $\max_{x \in [0,1]} \; x(1-x)$

(f) $\text{argmax}_{x \in [0,1]} \; x(1-x)$

$$(a) = [\pi \;\; e \;\; 1]\begin{bmatrix} 2 \\ 3 \\ -1 \end{bmatrix} + \sqrt{\pi^2+e^2+1} = 2\pi + 3e - 1 + \sqrt{\pi^2+e^2+1}$$

$$(b) = [\pi \; e \; 1]\begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix}\begin{bmatrix} \pi \\ e \\ 1 \end{bmatrix} = [\pi\cos\alpha - \sin\alpha \;\; e \;\; \pi\sin\alpha + \cos\alpha]\begin{bmatrix} \pi \\ e \\ 1 \end{bmatrix}$$

$$= \pi^2\cos\alpha \underline{- \pi\sin\alpha} + e^2 + \underline{\pi\sin\alpha} + \cos\alpha$$

$$= (\pi^2 + 1)\cos\alpha + e^2$$

$$= \frac{\sqrt{2}}{2}\pi^2 + \frac{\sqrt{2}}{2} + e^2$$

$$\cos\frac{\pi}{4} = \frac{\sqrt{2}}{2}$$

$$(c) = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix}\begin{bmatrix} \cos\alpha & 0 & \sin\alpha \\ 0 & 1 & 0 \\ -\sin\alpha & 0 & \cos\alpha \end{bmatrix} = \begin{bmatrix} \cos^2\alpha + \sin^2\alpha & 0 & 0 \\ 0 & 1 & 0 \\ 0 & & \sin^2\alpha + \cos^2\alpha \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

(d)
(e)
(f)

$$\frac{d}{dx} x(1-x) = -x + 1 - x = 1 - 2x \qquad 1 - 2x = 0 \Rightarrow x = 1/2$$

$$\frac{d^2}{dx^2} x(1-x) = -2 < 0$$

$$\therefore \min_{x\in[0,1]} x(1-x) = 0$$

$$\max_{x\in[0,1]} x(1-x) = \frac{1}{2}\cdot\frac{1}{2} = \frac{1}{4}$$

$$\text{argmax}_{x\in[0,1]} x(1-x) = \frac{1}{2}$$

6. (2 points) Normal distribution derivatives

The stochastic policies we will use in reinforcement learning for continuous action problems will often use a normally-distributed action, with the policy output consisting of the mean, μ, and variance, σ^2, i.e., $a \sim \mathcal{N}(\mu, \sigma^2)$, where the normal distribution is given by:

$$P(a) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a-\mu}{\sigma})^2}$$

We will see that to use backpropagation, we will need to know the derivative of the normal distribution probability with respect to μ and σ^2.

Show how to compute the derivatives of the normal distribution with respect to the mean and variance.

$$\frac{\partial}{\partial \mu} P(a) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a-\mu}{\sigma})^2} \cdot \frac{\partial}{\partial \mu}\left(-\frac{1}{2}\left(\frac{a-\mu}{\sigma}\right)^2\right) \quad \text{using chain rule.}$$

$$= \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a-\mu}{\sigma})^2} \cdot \left(-\cancel{\frac{1}{2}}\right) \cdot \cancel{2} \cdot \left(\frac{a-\mu}{\sigma}\right) \cdot \left(-\frac{1}{\sigma}\right)$$

$$= \frac{a-\mu}{\sigma^2} P(a)$$

$$\frac{\partial}{\partial \sigma^2} P(a) = \frac{\partial}{\partial \sigma^2}\left((\sigma^2)^{-\frac{1}{2}} \cdot \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{1}{2}(a-\mu)^2 \cdot (\sigma^2)^{-1}}\right)$$

$$= \frac{1}{\sqrt{2\pi}} \cdot \left((\sigma^2)^{-\frac{1}{2}} \cdot \frac{\partial}{\partial \sigma^2}\left(e^{-\frac{1}{2}(a-\mu)^2 \cdot (\sigma^2)^{-1}}\right) + \frac{\partial}{\partial \sigma^2}\left((\sigma^2)^{-\frac{1}{2}}\right) \cdot e^{-\frac{1}{2}(a-\mu)^2 \cdot (\sigma^2)^{-1}}\right)$$

$$= \frac{1}{\sqrt{2\pi}} \cdot \left(\sigma^{-1} \cdot e^{-\frac{1}{2}(\frac{a-\mu}{\sigma})^2} \cdot \left(-\frac{1}{2}(a-\mu)^2\right) \cdot (-1)(\sigma^2)^{-2} + \left(-\frac{1}{2}\right)\underset{\sigma^{-3}}{(\sigma^2)^{-\frac{3}{2}}} \; e^{-\frac{1}{2}(\frac{a-\mu}{\sigma})^2}\right)$$

$$= \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a-\mu}{\sigma})^2}\left(\frac{1}{2}(a-\mu)^2\sigma^{-5} - \frac{1}{2}\sigma^{-3}\right)$$

$$= \underbrace{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{a-\mu}{\sigma})^2}}_{P(a)}\left(\frac{1}{2}(a-\mu)^2\sigma^{-4} - \frac{1}{2}\sigma^{-2}\right)$$

$$= \frac{1}{\sqrt{2\pi}} e^{-\frac{(a-\mu)^2}{2\sigma^2}}\left(-\frac{1}{2}(\sigma^2)^{-\frac{3}{2}} + \frac{(a-\mu)^2}{2}(\sigma^2)^{-\frac{5}{2}}\right)$$

7. (5 points) Understanding auto-differentiation

Deep reinforcement learning methods will make use of neural-networks to model the policy, $\pi(s)$, the state value function, $V(s)$, and the state-action value function, $Q(s, a)$. This question is about computation graphs and automatic differentiation. Training a neural network is about finding a set of parameters (weights) that optimizes a particular objective function. Gradient descent is key to this and thus automatic differentiation algorithms play a key role in being able to compute the gradients of loss functions with respect to neural network weights.

For the following vector function, answer the questions below.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \mathbf{f}(x_1, x_2) = \begin{bmatrix} ln(x_1) + x_1 sin(3{x_2}^2) \\ e^{x_1 x_2} + x_1 \end{bmatrix}$$

In the space below:

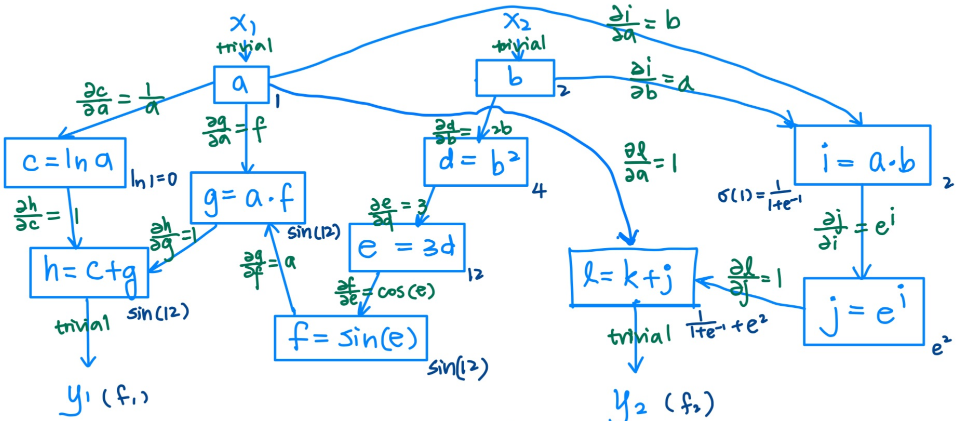
(a) Draw the computation graph.

(b) Annotate derivatives on each path of graph.

(c) Compute the value of $\mathbf{f}(x_1, x_2)$ at $x_1 = 1$ and $x_2 = 2$.

(d) Compute the Jacobian, $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$, at $x_1 = 1$ and $x_2 = 2$ using finite differences.

(e) Compute the Jacobian in closed form and evaluate it at the same point, i.e., $x_1 = 1$ and $x_2 = 2$.


x_1
trivial
x_2
trivial
$\frac{\partial i}{\partial a} = b$
a
1
b
2
$\frac{\partial i}{\partial b} = a$
$\frac{\partial c}{\partial a} = \frac{1}{a}$
$\frac{\partial g}{\partial a} = f$
$\frac{\partial d}{\partial b} = 2b$
$\frac{\partial l}{\partial a} = 1$
$c = \ln a$
$\ln 1 = 0$
$g = a \cdot f$
$d = b^2$
4
$i = a \cdot b$
2
$\sigma(1) = \frac{1}{1+e^{-1}}$
$\frac{\partial h}{\partial c} = 1$
$\frac{\partial h}{\partial g} = 1$
sin(12)
$\frac{\partial e}{\partial d} = 3$
$e = 3d$
12
$\frac{\partial j}{\partial i} = e^i$
$\frac{\partial g}{\partial f} = a$
$h = c + g$
$\frac{\partial f}{\partial e} = \cos(e)$
$l = k + j$
$\frac{\partial l}{\partial j} = 1$
$j = e^i$
sin(12)
trivial
$f = \sin(e)$
trivial
$\frac{1}{1+e^{-1}} + e^2$
e^2
sin(12)
y_1 (f_1)
y_2 (f_2)

(c). $f(x_1, x_2)\big|_{x_1=1, x_2=2} = \begin{bmatrix} \ln(1) + \sin(3 \cdot 2^2) \\ e^{1 \cdot 2} + 1 \end{bmatrix} = \begin{bmatrix} \sin(12) \\ e^2 + 1 \end{bmatrix}$

(d) (Using a forward approximation)

$$\nabla f = \lim_{h \to 0} \begin{bmatrix} \frac{f_1(x_1+h, x_2) - f_1(x_1, x_2)}{h} & \frac{f_1(x_1, x_2+h) - f_1(x_1, x_2)}{h} \\ \frac{f_2(x_1+h, x_2) - f_2(x_1, x_2)}{h} & \frac{f_2(x_1, x_2+h) - f_2(x_1, x_2)}{h} \end{bmatrix}$$

Approximate using $h = 0.01$ at $x_1 = 1, x_2 = 2$

$$\approx \begin{bmatrix} 100 \cdot (\ln(1.01) + 1.01 \sin(12) - \sin(12)) & 100(\ln 1 + \sin(12.12) - \sin(12)) \\ 100 \cdot (e^{1.01 \times 2} + 1.01 - e^2 - 1) & 100(e^{1 \times 2.01} + 1 - e^2 - 1) \end{bmatrix} \approx \begin{bmatrix} 0.458 & 10.51 \\ 15.93 & 7.43 \end{bmatrix}$$

(e) $$\nabla f = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{\partial h}{\partial c} \cdot \frac{\partial c}{\partial a} + \frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial a} & \frac{\partial h}{\partial g} \cdot \frac{\partial g}{\partial f} \cdot \frac{\partial f}{\partial e} \cdot \frac{\partial e}{\partial d} \cdot \frac{\partial d}{\partial b} \\ \frac{\partial l}{\partial a} + \frac{\partial l}{\partial j} \cdot \frac{\partial j}{\partial i} \cdot \frac{\partial i}{\partial a} & \frac{\partial l}{\partial j} \cdot \frac{\partial j}{\partial i} \cdot \frac{\partial i}{\partial b} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{x_1} + \sin(3x_2^2) & x_1 \cos(3x_2^2) \cdot 6x_2 \\ e^{x_1 x_2} \cdot x_2 + 1 & e^{x_1 x_2} \cdot x_1 \end{bmatrix}$$

At $x_1 = 1$, $x_2 = 2$ $= \begin{bmatrix} 1 + \sin(12) & 12 \cos(12) \\ 2e^2 + 1 & e^2 \end{bmatrix}$

Verified: Close to the numerical approximation in (d)

8. (2 points) Gradient descent

For $f(x) = x^2$, starting from $x = 2$, show how you can find the minimum of $f(x)$ using gradient descent step-by-step. Show the first two iterations using a gradient step size of 0.5.

$\nabla f(x) = 2x$

First iteration :

$x = 2. \quad f(x) = 4.$

$x \to x - \alpha \cdot \nabla f(x) \qquad \alpha = 0.5$

$2 - 0.5 \cdot 2 \cdot 2$

$\to 0$

Second iteration :

$x = 0, \quad f(x) = 0$

$x \to x - \alpha \cdot \nabla f(x)$

$0 - 0.5 \cdot 2 \cdot 0$

$\to 0$

$\therefore$ minimum of $f(x) = f(x)|_{x=0} = 0^2 = 0$

9. (6 points) This question is a hands-on exercise on a neural network's forward and backward pass.

   Numerous good examples of forward and backward propagation for neural networks can be found online, e.g., :
   `https://theneuralblog.com/forward-pass-backpropagation-example/`.
   A visual website to play with neural network architectures and weights is available here:
   `https://playground.tensorflow.org/`.]

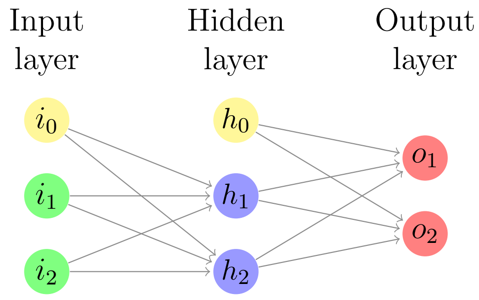


Figure 1: Neural network representation. i_0 and h_0 are the bias units.

Let us define a neural network with two inputs, two outputs, one hidden layer with two neurons, as depicted in fig. 1. Both layers have a bias of 1 (i_0 and h_0) and let's assume for simplicity that their weights are fixed. Use a ReLU activation on the hidden. The output layer is a linearly weighted combination of the hidden layer outputs. The loss function is L2. For updating the weights, we use stochastic gradient descent with a learning rate of 0.5.

The weights are initialized as follows:

$$\begin{aligned}
w_{i_1-h_1} &= w_1 = 0.5\\
w_{i_2-h_1} &= w_2 = 0.55\\
w_{i_1-h_2} &= w_3 = 0.6\\
w_{i_2-h_2} &= w_4 = 0.65\\
w_{h_1-o_1} &= w_5 = 0.7\\
w_{h_2-o_1} &= w_6 = 0.75\\
w_{h_1-o_2} &= w_7 = 0.8\\
w_{h_2-o_2} &= w_8 = 0.85\\
w_{i_0-h_1} = w_{i_0-h_2} &= b_1 = 0.35\\
w_{h_0-o_1} = w_{h_0-o_2} &= b_2 = 0.60
\end{aligned}$$

Use the following sample point:

$$\begin{aligned}
X &= [0.3, 0.5]\\
Y &= [0.01, 0.99]
\end{aligned}$$

For each of the following parts, show your work for at least one of the values requested. For the others, the result is sufficient.

(a) Forward pass: Compute the values of o_1 and o_2 using X as input;

(b) Loss computation: Compute the total error E_{total} when the target value is Y;

(c) Backward pass (output layer): Compute the partial errors with respect to the output layer $\frac{\partial E_{total}}{\partial w_5}$, $\frac{\partial E_{total}}{\partial w_6}$, $\frac{\partial E_{total}}{\partial w_7}$, $\frac{\partial E_{total}}{\partial w_8}$;

(d) Optimization step (output layer): Using the values computed in the previous point, update the weights w_5, w_6, w_7, w_8;

(e) Backward pass (hidden layer): Compute the partial errors with respect to the hidden layer $\frac{\partial E_{total}}{\partial w_1}$, $\frac{\partial E_{total}}{\partial w_2}$, $\frac{\partial E_{total}}{\partial w_3}$, $\frac{\partial E_{total}}{\partial w_4}$;

(f) Optimization step (hidden layer): Using the values computed in the previous point, update the weights w_1, w_2, w_3, w_4.

Let $\sigma(\cdot)$ denote relu activation function

(a) $h_1 = \sigma(0.35 + 0.5 \times 0.3 + 0.55 \times 0.5) = 0.775$

$h_2 = 0.855$

$o_1 = 0.6 + 0.7h_1 + 0.75h_2 = 1.784$

$o_2 = 1.947$

(b) $E_{total} = \frac{1}{2}\sum(\text{target} - \text{output})^2 = \frac{1}{2}\left[(y_1 - o_1)^2 + (y_2 - o_2)^2\right] = 2.031$

(c) $\dfrac{\partial E_{total}}{\partial w_5} = \dfrac{\partial E_{total}}{\partial o_1} \cdot \dfrac{\partial o_1}{\partial w_5} = (o_1 - y_1) \cdot h_1 = 1.375$

$\dfrac{\partial E_{total}}{\partial w_6} = 1.517$, $\dfrac{\partial E_{total}}{\partial w_7} = 0.741$, $\dfrac{\partial E_{total}}{\partial w_8} = 0.818$

(d) $w_5' = w_5 - \partial \cdot \dfrac{\partial E_{total}}{\partial w_5} = 0.013$

$w_6' = -0.008$, $w_7' = 0.429$, $w_8' = 0.441$

$\sigma' = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$

(e) $\dfrac{\partial E_{total}}{\partial w_1} = \dfrac{\partial E_{total}}{\partial h_1} \cdot \dfrac{\partial h_1}{\partial w_1} = \left(\dfrac{\partial E_{total}}{\partial o_1} \cdot \dfrac{\partial o_1}{\partial h_1} + \dfrac{\partial E_{total}}{\partial o_2} \cdot \dfrac{\partial o_2}{\partial h_1}\right) \cdot \sigma'\left([i_0\ i_1\ i_2]\begin{bmatrix} b_1 \\ w_1 \\ w_2 \end{bmatrix}\right) \cdot i_1$

$= \left[(o_1 - y_1)w_5 + (o_2 - y_2) \cdot w_7\right] \cdot i_1$

≈ 0.602

Similarly $\dfrac{\partial E_{total}}{\partial w_2} \approx 1.00$, $\dfrac{\partial E_{total}}{\partial w_3} \approx 0.643$, $\dfrac{\partial E_{total}}{\partial w_4} \approx 1.072$

(f) $w_1' = w_1 - \alpha \dfrac{\partial E_{total}}{\partial w_1} \approx 0.199$

$w_2' \approx 0.048$

$w_3' \approx 0.282$

$w_4' \approx 0.114$

(any number roughly in range is acceptable)

10. This section contains conceptual questions. They will not be graded.

    (a) Why do we randomly initialize the weights of a network network? Relatedly, what happens if we initialize all weights to 0? What happens if we initialize all weights to the same non-zero value?
    Random initialization of weights helps the hidden units to get non-trivial and different signals. All hidden units would get 0 signal regardless of the input if all weights are set to 0. The network would not learn. If they are all the same non-zero value, all units will get the same signal, hence diminishing the point of having multiple neurons.

    (b) What happens if we were to use a purely linear activation function?
    Combinations of pure linear layers would result in an overall linear network, equivalent to have only one layer.

    (c) Why do we use stochastic gradient descent instead of gradient descent?
    It is faster as it calculate on a small batch of data at each iteration.

    (d) Why is ReLU more commonly used than a sigmoid?
    Sigmoid suffers from issue like vanishing gradient while ReLU does not. ReLU is also computationally lower cost.

11. (4 points) Reading

    Read the paper: "A critique of pure learning and what artificial neural networks can learn from animal brains" `https://www.nature.com/articles/s41467-019-11786-6.pdf`.

    (a) (2 points) Give two implications (of the many possible implications) of the proposed point-of-view for reinforcement learning.

    (b) (2 points) Give an argument that goes at least partly against the given point of view.
    Keep your answers short, i.e., one paragraph for each implication and each argument. Be original!