ECE/CS 559 - Fall 2016 - Midterm and Solutions.

Erdem Koyuncu

- **Q1 (24 pts):** In the following, we follow the convention that $x_0 = 1$.

  Recall that the step-activation function $u : \mathbb{R} \to \{0, 1\}$ is defined as $u(v) = 1$ if $v \geq 0$, and $u(v) = 0$, otherwise. Then, for n inputs $x_1, \ldots, x_n$, a **perceptron** can be defined via the input-output relationship

  $$y' = u\left(\sum_{i=0}^{n} w_i x_i\right) = u(w_0 + w_1 x_1 + \cdots + w_n x_n),$$

  where y' is the perceptron output, $w_1, \ldots, w_n$ are the synaptic weights, and w_0 is the bias term.

  We define a new type of neuron, namely, a **sauron**, via the input-output relationship

  $$y = u\left(\prod_{i=0}^{n} (w_i + x_i)\right) = u((w_0 + 1)(w_1 + x_1) \cdots (w_n + x_n)),$$

  where y is called the sauron output.

  Let the real number 1 represent a `TRUE`, and the real number 0 represent a `FALSE`.

  (a) **(8 pts):** Let $n = 1$. Does there exist w_0, w_1 such that $y = 1 - x_1$ for $x_1 \in \{0, 1\}$? In other words, can a single sauron implement the `NOT` gate? If your answer is "Yes," find specific w_0, w_1 such that the sauron implements the `NOT` gate. If your answer is "No," prove that no choice for w_0, w_1 can result in a sauron that implements the `NOT` gate.

  (b) **(8 pts):** Let $n = 2$. Does there exist w_0, w_1, w_2 such that $y = x_1 x_2$ for $x_1, x_2 \in \{0, 1\}$? In other words, can a single sauron implement the `AND` gate? Justify your answer as in (a).

  (c) **(8 pts):** Let $n = 2$. Does there exist w_0, w_1, w_2 such that $y = ((x_1 + x_2) \mod 2)$ for $x_1, x_2 \in \{0, 1\}$? In other words, can a single sauron implement the `XOR` gate? Justify your answer as in (a).

Solution:

(a) Yes. We need $u((1 + w_0)(w_1 + x_1)) = 1 - x_1$ for $x_1 \in \{0, 1\}$. In other words, we need $(1 + w_0)w_1 \geq 0$ and $(1 + w_0)(1 + w_1) < 0$. One solution is $w_0 = -2$ and $w_1 = -\frac{1}{2}$.

(b) No. The following inequalities need to be satisfied:

$$\begin{aligned} (1 + w_0)w_1 w_2 &< 0, \\ (1 + w_0)(1 + w_1)w_2 &< 0, \\ (1 + w_0)w_1(1 + w_2) &< 0, \\ (1 + w_0)(1 + w_1)(1 + w_2) &\geq 0. \end{aligned}$$

Dividing the third inequality by the first inequality, we obtain $\frac{1+w_2}{w_2} > 0$. Diving the fourth inequality by the second inequality, $\frac{1+w_2}{w_2} \leq 0$, a contradiction.

(c) Yes. We need

$$\begin{aligned} (1 + w_0)w_1 w_2 &< 0, \\ (1 + w_0)(1 + w_1)w_2 &\geq 0, \\ (1 + w_0)w_1(1 + w_2) &\geq 0, \\ (1 + w_0)(1 + w_1)(1 + w_2) &< 0. \end{aligned}$$

By inspection, $w_0 = -2$, $w_1 = w_2 = -\frac{1}{2}$ will work.

- **Q2 (26 pts):** In the following, consider only neurons with the step-activation function $u(\cdot)$.

  (a) **(13 pts):** Let $\mathcal{C}_0 = \{\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right], \left[\begin{smallmatrix}-1\\1\end{smallmatrix}\right], \left[\begin{smallmatrix}0\\4\end{smallmatrix}\right]\}$ and $\mathcal{C}_1 = \{\left[\begin{smallmatrix}0\\2\end{smallmatrix}\right], \left[\begin{smallmatrix}0\\-1\end{smallmatrix}\right], \left[\begin{smallmatrix}0\\-2\end{smallmatrix}\right]\}$ as illustrated in Figure (i). Members of classes $\mathcal{C}_0$ and $\mathcal{C}_1$ are represented by black disks and crosses, respectively.

  [I] **(7 pts):** We wish to design a perceptron $y = u(w_0 + w_1x_1 + w_2x_2)$ such that $y = 0$ if $\left[\begin{smallmatrix}x_1\\x_2\end{smallmatrix}\right] \in \mathcal{C}_0$ and $y = 1$ if $\left[\begin{smallmatrix}x_1\\x_2\end{smallmatrix}\right] \in \mathcal{C}_1$. Suppose that we use the perceptron training algorithm for this purpose with initial weights $w_0 = 1$, $w_1 = 0$, $w_2 = 1$ and learning rate $\eta = 1$. Either prove that the algorithm will converge, or prove that it will not converge. You may use the perceptron convergence theorem.

  [II] **(6 pts):** Design a neural network (single-layer or multi-layer) such that the network provides an output of 0 if $\left[\begin{smallmatrix}x_1\\x_2\end{smallmatrix}\right] \in \mathcal{C}_0$ and an output of 1 if $\left[\begin{smallmatrix}x_1\\x_2\end{smallmatrix}\right] \in \mathcal{C}_1$.

  (b) **(13 pts)** Repeat (a) for classes $\mathcal{C}_0 = \{\left[\begin{smallmatrix}1\\1\end{smallmatrix}\right], \left[\begin{smallmatrix}-1\\1\end{smallmatrix}\right], \left[\begin{smallmatrix}0\\4\end{smallmatrix}\right]\}$ and $\mathcal{C}_1 = \{\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right], \left[\begin{smallmatrix}0\\-1\end{smallmatrix}\right], \left[\begin{smallmatrix}0\\-2\end{smallmatrix}\right]\}$ illustrated in Figure (ii). Note that the only difference is that now, instead of the point $\left[\begin{smallmatrix}0\\2\end{smallmatrix}\right]$, we have the point $\left[\begin{smallmatrix}0\\0\end{smallmatrix}\right]$ in class $\mathcal{C}_1$.

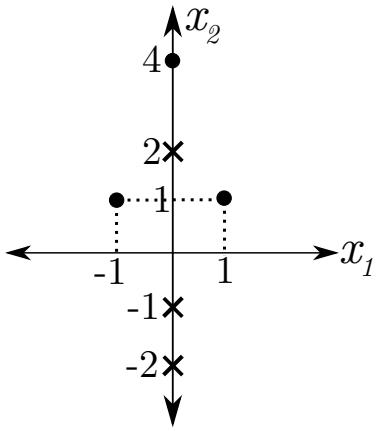


Figure (i)

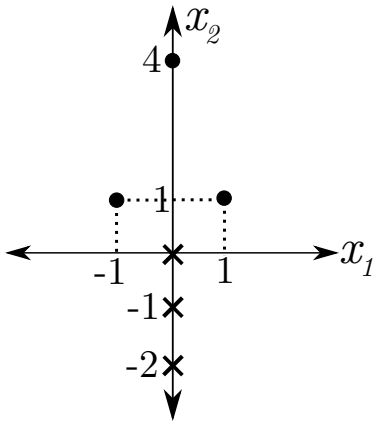


Figure (ii)

Solution:

(a) [I] PTA will not converge as the patterns are not linearly separable (the weights will always be updated).

[II] Let $y_1 = u(-x_2 - 3x_1 + 3)$ and $y_2 = u(-x_2 + 3x_1 + 3)$. The network $u(-\frac{3}{2} + y_1 + y_2)$ will work.

(b) [I] PTA will converge as the patterns are linearly separable.

[II] The same network in (a)-[II] will work! Or, a simpler network is $u(-x_2 + \frac{1}{2})$.

In the following, $\log(\cdot)$ is the natural logarithm, e is the base of the natural logarithm, $|\cdot|$ is the absolute value ($|x| = x$ if $x \geq 0$, and $|x| = -x$ if $x < 0$), and $\mathbb{R}$ is the set of real numbers. Recall $\frac{\partial e^x}{\partial x} = e^x$, and $e^{\log x} = x$, $x > 0$.

- **Q3 (25 pts):** Consider the activation function

$$f(v) = \begin{cases} 1 - e^{-v}, & v \geq 0, \\ -1 + e^{-|v|}, & v < 0. \end{cases}$$

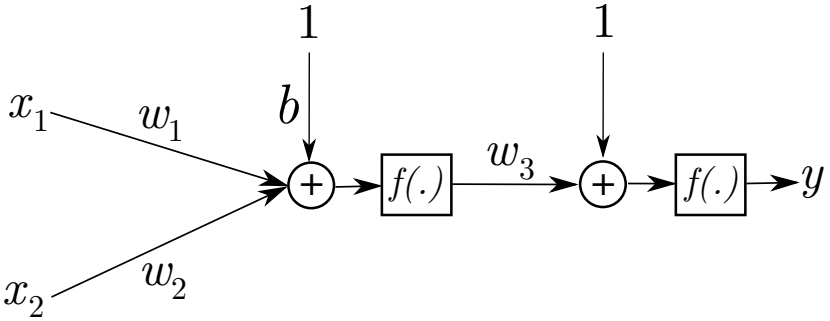
For (a)-(c), consider a single-neuron network with input-output relationship $y = f(b + \mathbf{w}^T\mathbf{x})$, where y is the network output, b is the bias term, $\mathbf{w} = \left[\begin{smallmatrix} w_1 \\ w_2 \end{smallmatrix}\right]$ are the synaptic weights, and $\mathbf{x} = \left[\begin{smallmatrix} x_1 \\ x_2 \end{smallmatrix}\right]$ is the network input.

(a) **(3 pts):** Calculate the derivative $f'(v)$ in closed form.

(b) **(7 pts):** Let $E = (d - y)^2$, where d is a constant (a generic desired output). Find the delta-learning rule (the gradient-descent update equations) for b, w_1, w_2 given learning parameter $\eta = \frac{1}{2}$. Remember that you can write a single (vectorized) update equation that can handle all variables. The final update expression(s) may however contain only the terms/functions $d, y, f', f, \mathbf{w}, w_1, w_2, b$.

(c) **(6 pts):** Consider the same delta-learning setup as in (b). Consider the training vectors $\mathbf{x}_1 = \left[\begin{smallmatrix} \log 2 \\ \log 3 \end{smallmatrix}\right]$, $\mathbf{x}_2 = \left[\begin{smallmatrix} 0 \\ \log 2 \end{smallmatrix}\right]$, with desired outputs $d_1 = \frac{2}{3}$, $d_2 = \frac{5}{2}$, respectively. Find the updated bias and the updated synaptic weights after one epoch of online learning given initial conditions $b = 0$, $\mathbf{w} = \left[\begin{smallmatrix} 0 \\ 1 \end{smallmatrix}\right]$.

(d) **(9 pts)** Consider now a multi-layer network as shown below.



Let $E = e^{(d-y)^4}$. Find the gradient-descent update equations for b, w_1, w_2, w_3 given learning parameter $\eta = \frac{1}{4}$. The final expression may only contain the terms/functions $d, y, f', f, \mathbf{w}, w_1, w_2, b, w_3$.
Hint: You do not have to apply some algorithm. Just write the input-output relationship of the network, and use chain rule. This is more of a calculus problem than a neural network problem!

Solution:

(a) By calculus, $f'(v) = e^{-|v|}$.

(b) The same delta-rule in class: $\left[\begin{smallmatrix} b \\ w_1 \\ w_2 \end{smallmatrix}\right] \leftarrow \left[\begin{smallmatrix} b \\ w_1 \\ w_2 \end{smallmatrix}\right] + (d - y)f'(v)\left[\begin{smallmatrix} 1 \\ x_1 \\ x_2 \end{smallmatrix}\right]$, where $v = b + \mathbf{w}^T\mathbf{x}$.

(c) For initial bias/weights $\left[\begin{smallmatrix} 0 \\ 0 \\ 1 \end{smallmatrix}\right]$, if the input is $\mathbf{x}_1$, we obtain $v = \log 3$ so that $y = 1 - e^{-\log 3} = \frac{2}{3}$. Since $d_1 = \frac{2}{3}$, there will be no update. For the input x_2, with same bias and weights we have started with, we obtain $v = \log 2$, so that $y = \frac{1}{2}$, and $f'(v) = \frac{1}{2}$. This gives $\left[\begin{smallmatrix} 0 \\ 0 \\ 1 \end{smallmatrix}\right] + (\frac{5}{2} - \frac{1}{2})\frac{1}{2}\left[\begin{smallmatrix} 1 \\ 0 \\ \log 2 \end{smallmatrix}\right] = \left[\begin{smallmatrix} 1 \\ 0 \\ 1+\log 2 \end{smallmatrix}\right]$ as the final weights after epoch 1.

(d) We have $y = f(1 + w_3 f(b + \mathbf{w}^T\mathbf{x}))$. By calculus/chain rule, we obtain

$$w_3 \leftarrow w_3 - \eta\frac{\partial e^{(d-y)^4}}{\partial w_3} = w_3 + (d - y)^3 e^{(d-y)^4} f'(1 + w_3 f(b + \mathbf{w}^T\mathbf{x}))f(b + \mathbf{w}^T\mathbf{x})$$

$$\left[\begin{smallmatrix} b \\ w_1 \\ w_2 \end{smallmatrix}\right] \leftarrow \left[\begin{smallmatrix} b \\ w_1 \\ w_2 \end{smallmatrix}\right] + (d - y)^3 e^{(d-y)^4} f'(1 + w_3 f(b + \mathbf{w}^T\mathbf{x}))w_3 f'(b + \mathbf{w}^T\mathbf{x})\left[\begin{smallmatrix} 1 \\ x_1 \\ x_2 \end{smallmatrix}\right]$$

- **Q4 (25 pts):** Design a (possibly multi-layer) neural network with two inputs $x_1, x_2 \in \mathbb{R}$ and a single output y such that $y = 1$ if $x_1 = 3$ and $x_2 = 2$, and $y = 0$, otherwise. In other words, the network output should assume a value of 1 only at the single point $(x_1, x_2) = (3, 2)$ of the input space $\mathbb{R}^2$. Note that the inputs can be any real numbers. The neuron(s) of the network should use the step-activation function $u : \mathbb{R} \to \{0, 1\}$ given by $u(v) = 1$ if $v \geq 0$, and $u(v) = 0$, otherwise. The network should consist of **5 neurons at most**. Spoilers and their partial credits:

  (a) **(3 pts):** Design a network with input x_1 and output y such that $y = 1$ if $x_1 \geq 3$, and $y = 0$, otherwise.

  (b) **(4 pts):** Design a network with input x_1 and output y such that $y = 1$ if $x_1 \leq 3$, and $y = 0$, otherwise.

  (c) **(6 pts):** Design a network with input x_1 and output y such that $y = 1$ if $x_1 = 3$, and $y = 0$, otherwise.

  (d) **(6 pts):** Solve the original problem without any restrictions on the number of neurons.

  (e) **(6 pts):** Solve the original problem.

  You are not required to do (a)-(d) provided you can provide a correct solution to (e).

Solution:

(a) $y_1 = u(x_1 - 3)$.

(b) $y_2 = u(-x_1 + 3)$.

(c) $u(y_1 + y_2 - \frac{3}{2})$, i.e. AND the two previous networks.

(d) and (e) We do (a) and (b) for $x_2 = 2$ as well, i.e. let $z_1 = u(x_2 - 2)$ and $z_2 = u(-x_2 + 2)$. Now, AND y_1, y_2, z_1, z_2, i.e. let $y = u(y_1 + y_2 + z_1 + z_2 - \frac{7}{2})$.
