Artificial Intelligence
Exercises week 3 - ANN - **Solutions**

COMP3411/9814

Question 1: Computing any Logical Function with a 2-layer Network

Recall that any logical function can be converted to Conjunctive Normal Form (CNF), which means a conjunction of terms where each term is a disjunction of (possibly negated) literals. This is an example of an expression in CNF:

$$(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$$

Assuming False=0 and True=1, explain how each of the following could be constructed. You should include the bias for each node, as well as the values of all the weights (input-to-output or input-to-hidden and hidden-to-output, as appropriate).

- Perceptron to compute the OR function of m inputs,

  Answer: Set the bias weight to -0.5 and all other weights to 1. It makes sense for the input to output weights to be 1, because any of the inputs being True makes it more likely for the output to be True. In fact, the ONLY way the output can be False is if ALL the inputs are False. By setting the bias to -0.5, we insure that the linear combination is slightly negative when all of the inputs are False, but becomes positive when any of the inputs is True.

- Perceptron to compute the AND function of n inputs,

Answer: Set the bias weight to $(0.5 - n)$, all other weights to 1. The ONLY way the conjunction can be True is if ALL the inputs are True. By setting the bias to $(0.5-n)$, we insure that the linear combination is slightly positive when all of the inputs are True, but becomes negative when any of the inputs is False.

- Two-layer Neural Network to compute the function

$$(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$$

Answer: Each hidden node should compute one disjunctive term in the expression. The input to hidden weights are -1 for items that are negated, +1 for the others. The output node then computes the conjunction of all the hidden nodes, as in the previous part (see Fig. 1).

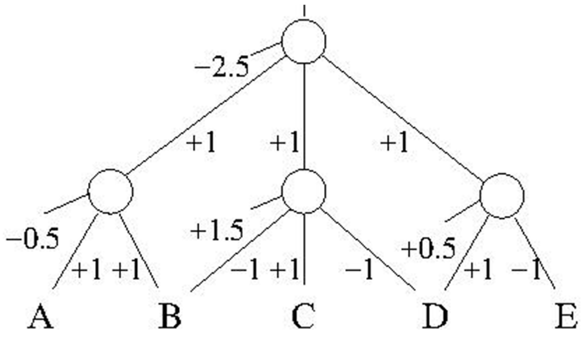


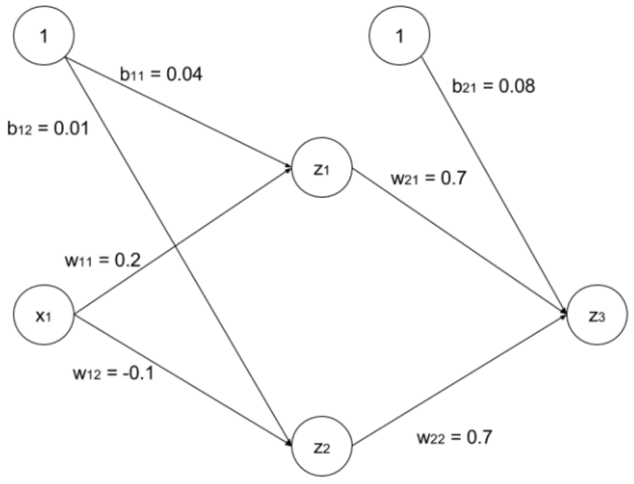
Figure 1: $(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$

With reference to this example, explain how a two-layer neural network could be constructed to compute any (given) logical expression, assuming it is written in Conjunctive Normal Form.
Hint: first consider how to construct a Perceptron to compute the OR function of m inputs, with k of the m inputs negated.

Answer: As in the example above, each hidden node should compute one disjunctive term in the expression; the output node then computes the conjunction of all these hidden nodes. The input to hidden weights should be -1 for items that are negated, +1 for the others. The bias for each hidden node should be $(k - 0.5)$ where k is the number of items that are negated in the disjunctive term corresponding to that node.

Question 2: Backpropagation

Consider the neural network architecture shown below for a binary classification problem. The values for weights and biases are shown in the figure. We define:
$a_1 = w_{11}x_1 + b_{11}$
$a_2 = w_{12}x_1 + b_{12}$
$a_3 = w_{21}z_1 + w_{22}z_2 + b_{21}$
$z_1 = ReLU(a_1), ReLU(x) = max(0, x)$
$z_2 = ReLU(a_2)$
$z_3 = \sigma(a_3), \sigma(x) = \frac{1}{1+e^{-x}}$
$Loss = \frac{1}{2}(y - z_3)^2$
Calculate the new bias b_{11} after one iteration, after performing backpropagation on the bias for a data sample $(x_1 = 1, y = 1)$ and learning rate equals to 1.



Answer:

$$a_1 = w_{11}x_1 + b_{11} = 0.2(1) + 0.04 = 0.24$$

$$z_1 = \text{ReLU}(a_1) = \max(0, 0.24) = 0.24$$

$$a_2 = w_{12}x_1 + b_{12} = -0.1(1) + 0.01 = -0.09$$

$$z_2 = \text{ReLU}(a_2) = \max(0, -0.09) = 0$$

$$a_3 = w_{21}z_1 + w_{22}z_2 + b_{21} = 0.7(0.24) + 0.7(0) + 0.08 = 0.248$$

$$z_3 = \sigma(a_3) = \frac{1}{1+e^{-0.248}} \approx 0.5617$$

$$\frac{\partial L}{\partial a_3} = (z_3 - y)\, z_3(1-z_3)$$

$$\frac{\partial a_3}{\partial z_1} = w_{21} \qquad \frac{\partial z_1}{\partial a_1} = 1 \qquad \frac{\partial a_1}{\partial b_{11}} = 1$$

$$\frac{\partial L}{\partial b_{11}} = \frac{\partial L}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_1} \cdot \frac{\partial z_1}{\partial a_1} \cdot \frac{\partial a_1}{\partial b_{11}} = (z_3 - y)\, z_3(1-z_3)\, w_{21} \cdot 1 \cdot 1$$

Plugging numbers:

$$z_3 - y \approx 0.56 - 1 = -0.44$$

$$z_3(1-z_3) \approx 0.56 \times (1-0.56) = 0.246$$

$$\frac{\partial L}{\partial b_{11}} \approx (-0.44) \times 0.246 \times 0.7 \approx -0.075$$

Gradient descent update (with $\eta = 1$):

$$b_{11}^{\text{new}} = b_{11} - \eta \frac{\partial L}{\partial b_{11}} = 0.04 - 1 \cdot (-0.075) = 0.040 + 0.075 \approx 0.115$$

Question 3: Neural design

- What would be the MLP architecture to approximate a non-linear function of 3 inputs and 2 outputs if you have available 3,000 samples? Consider 70% data for training and 30% for validation.

  Answer: What it is known from the previous description is the following:

  $Ni = 3$
  $No = 2$
  $N_{samples} = 2,100$ as we use only 70% for training.
  $Nh = ?$

  But we also know that $Nw = (Ni + 1) \times Nh + (Nh + 1) \times No$ and $Nw < N_{samples}/10$, then:

$210 = (3+1) \times Nh + (Nh+1) \times 2$
$210 = 4Nh + 2Nh + 2$
$208 = 6Nh$
$Nh = 208/6$
$Nh = 34.6 \rightarrow 34$ as the number of neurons in the hidden layer is an integer. It might be 35, however, with 34 we keep true the relationship $Nw < N_{samples}/10$.

- What would be the MLP architecture to approximate a non-linear function of 6 inputs and 3 outputs if you have available 100 samples? Consider 80% data for training and 20% for validation.

  Answer: What it is known from the previous description is the following:

  $Ni = 6$
  $No = 3$
  $N_{samples} = 80$ as we use only 80% for training.
  $Nh = ?$

  But we also know that $Nw = (Ni+1) \times Nh + (Nh+1) \times No$ and $Nw < N_{samples}/10$, then:

  $8 = (6+1) \times Nh + (Nh+1) \times 3$
  $8 = 7Nh + 3Nh + 3$
  $5 = 10Nh$
  $Nh = 5/10$
  $Nh = 0.5 \rightarrow$ it is possible that there is not enough data to correctly train the neural model as the number of neurons in the hidden layer would be less than 1. Although this depends on the complexity of the underlying function, an alternative solution would be to capture more data, if possible.