CS 6613 **Fall 2023**
HW 4 **E. K. Wong**

Total # questions = 6. Total # points = 90.

1. [5 points] Name two methods that we have learned in class to prove *entailment* in propositional logic. (No need to explain the methods. Just name them.)

2. [15 points] Given the following propositional logic sentences in a knowledge base *KB*, use *resolution* to prove that the knowledge base entails the sentence *G*; i.e., *KB* |= *G*. Show all work to get full credits.

***KB*:**

$F1 \wedge (\neg F4 \Leftrightarrow \neg F3)$
$\neg F1 \vee F3$
$F4 \Longrightarrow \neg F2$
$F3 \Longrightarrow F5$

***G*:** $\neg F2 \wedge F5$

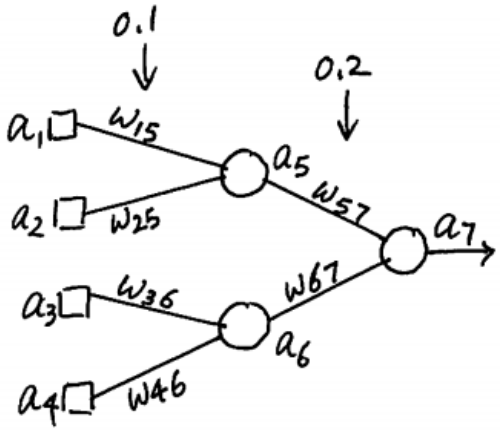
3. [20 points] Use the LEARN-DECISION-TREE algorithm in Figure 1 below (on page 3) to learn a decision tree from the restaurant training examples below, where *Hun* and *Res* are attributes. Show all work, including the computation of information gain, to get full credits.

Examples	***Hun***	***Res***	**Output**
x_1	N	Y	N
x_2	Y	N	N
x_3	N	N	Y
x_4	N	N	Y
x_5	N	N	Y
x_6	Y	N	N
x_7	N	Y	N
x_8	Y	N	Y

Y – Yes; N – No

4. [15 points] The feedforward neural network in the figure below has four input units, two neurons in the hidden layer and one neuron in the output layer. The neurons in the network do not have bias inputs and bias weights. All four connections between the input layer and the hidden layer have a weight value of 0.1 and the two connections between the hidden layer and the output layer have a weight value of 0.2. The *ReLU* function is used as the activation function in neurons 5 and 6 and the *Sigmoid* (or *Logistic*) function is used as the activation function in neuron 7. The input values for the four input units are (a1, a2, a3, a4) = (1.0, 1.5, -2.0, -2.5). Compute the output values for neurons a5, a6 and a7. Show all work to get full credits.

Note**:** *Sigmoid* function $S(x) = 1/(1 + e^{-x})$; *ReLu* function = *max*(0,*x*).



5. [20 points] We would like to train the neural network above using back propagation and gradient descent. Assume that the four connections between the input layer and the hidden layer are initialized with a value of 0.1, and the two connections between the hidden layer and the output layer are initialized with a value of 0.2. Let the learning rate be 0.1. Given the training example below, use the updating rules below to train the network **one time** (one epoch) and give the weight values for all six connections in the network after training. (Hint: note that the initial weight values and the input values are the same as in Question 4 above.)

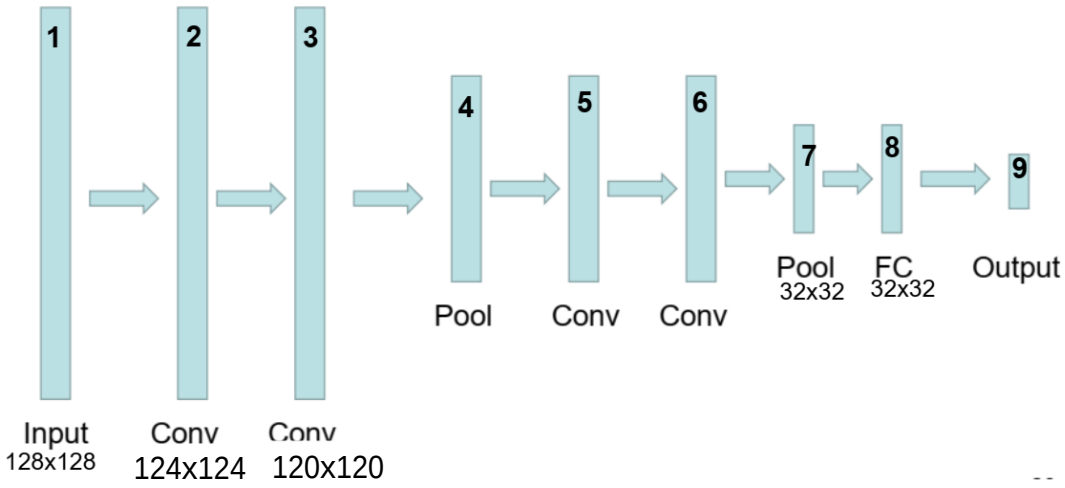
Training example: (a1, a2, a3, a4) = (1.0, 1.5, -2.0, -2.5); Output label = 1.0.

Updating rules:

$$W_{ji} \leftarrow W_{ji} - \alpha \times \frac{\partial E_i}{\partial W_{ji}} = W_{ji} + \alpha \times \Delta_i \times a_{j,} \text{ with } \Delta_i = Err_i \times g'(in_i)$$

$$W_{kj} \leftarrow W_{kj} + \alpha \times \Delta_j \times a_k, \text{with } \Delta_j = g'(in_j) \sum_i W_{ji} \Delta_i$$

6. [15 points] In the convolutional neural network below, the numbers inside the rectangles represent the layer number and the numbers at the bottom of the rectangles represent the number of units in the different layers (e.g., the input layer has $128 \times 128 = 16{,}384$ input units.) *Conv* indicates a convolutional layer; *Pool* indicates a pooling layer and *FC* indicates a fully-connected layer. Assume that each convolutional layer has only one filter of size 5×5 in the network.



(a) How many parameters need to be trained between layer 1 and layer 2, assuming that there are no bias weights for the units in layer 2?

(b) Suppose we change layer 2 to a sparsely connected layer by getting rid of *parameter sharing* in the original convolutional layer, how many parameters need to be trained? Again, assume that there are no bias weights for the units in layer 2.

(c) What is the size of layer 4 if 4-to-1 pooling (2 x 2 features become 1) is used?

(d) How many parameters need to be trained between layer 7 and layer 8? Again, assuming no bias weights for the units in layer 8.

function LEARN-DECISION-TREE($examples$, $attributes$, $parent\_examples$) **returns** a tree

if $examples$ is empty **then return** PLURALITY-VALUE($parent\_examples$)
else if all $examples$ have the same classification **then return** the classification
else if $attributes$ is empty **then return** PLURALITY-VALUE($examples$)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, $examples$)
  $tree \leftarrow$ a new decision tree with root test A
  for each value v of A **do**
    $exs \leftarrow \{e : e \in examples$ **and** $e.A = v\}$
    $subtree \leftarrow$ LEARN-DECISION-TREE(exs, $attributes - A$, $examples$)
    add a branch to $tree$ with label $(A = v)$ and subtree $subtree$
  return $tree$

Fig. 1. Decision tree learning algorithm.