1. Why is it generally preferable to use a Logistic Regression classifier rather than a classical Perceptron (i.e., a single layer of linear threshold units trained using the Perceptron training algorithm)? How can you tweak a Perceptron to make it equivalent to a Logistic Regression classifier?

ANS:

A classical perceptron will only converge if the data is linearly seperable. It also cannot compute class probabilities. The logistic regression classifier is able to converge on non-linear data and outputs class probabilities.

1. Why was the logistic activation function a key ingredient in training the first MLPs?

ANS:

The logistic activation function was a key ingredient in training the first MLPs because its derivative is always nonzero, so Gradient Descent can always roll down the slope. When the activation function is a step function, Gradient Descent cannot move, as there is no slope at all.

1. Name three popular activation functions. Can you draw them?

ANS:

* Sigmoid (Logistic)
* Hyperbolic Tangent (Tanh)
* Rectified Linear Unit (ReLU)
* Leaky ReLU.
* Parametric Leaky ReLU (PReLU)
* Exponential Linear Units (ELU)

1. Suppose you have an MLP composed of one input layer with 10 passthrough neurons, followed by one hidden layer with 50 artificial neurons, and finally one output layer with 3 artificial neurons. All artificial neurons use the ReLU activation function.
   * What is the shape of the input matrix **X**?

ANS:

The shape of X will be (m x 10) where m is the batch size. The logic of a passthrough neuron is that it should have no effect at all on the input layer. Therefore, there must be one neuron for each input feature that allows the input feature to pass through to the next layer without altering its value in any way.

* + What about the shape of the hidden layer’s weight vector **W***h*, and the shape of its bias vector **b***h*?

ANS:

Since the hidden layer has 50 neurons and there are 10 features, its weight vector will have the shape of 10 x 50. Because each of the 10 features will need to be multiplied by a weight which is connected to each of the 50 hidden layer neurons.

The bias vector will have a length of 50. This is because one bias unit is added to the output of the passthrough layer. That unit is then multiplied once for each of the 50 artificial neurons.

* + What is the shape of the output layer’s weight vector **W***o*, and its bias vector **b***o*?

ANS:

The output layer has 3 neurons. The input to the output layer is the output of the hidden layer.The output of the hidden layer has 50 neurons, therefore the shape of the output layer's weight vector is 50 x 3.

The bias vector will have a length of 3 because one bias unit is added to the output of the hidden layer. That unit is then multiplied once for each of the 3 neurons in the output layer.

* + What is the shape of the network’s output matrix **Y**?

ANS:

The shape of the output matrix is going to be m x 3, where m is the batch size, and 3 because each example will compute a probability that it belongs to one of three classes.

* + Write the equation that computes the network’s output matrix **Y** as a function of **X**, **W***h*, **b***h*, **W***o* and **b***o*.

ANS:

Y = (X \* Wh + bh) \* (Wo + bo)

When adding a bias vector to a matrix it is added to every single row. This is called broadcasting.

1. How many neurons do you need in the output layer if you want to classify email into spam or ham? What activation function should you use in the output layer? If instead you want to tackle MNIST, how many neurons do you need in the output layer, using what activation function?

ANS:

Email classification is a binary classification problem, so you would only need one neuron in the output layer. This neuron would indicate the probability that the email is spam or ham. You'd most likely use the sigmoid activation function in the output layer.

For the MNIST problem you would need 10 output neurons in the final layer, one for each digit. You would then replace the logistic function with the softmax function which can output one probability per class per digit.

Predicting housing prices is a linear regression problem. You'd only need one output neuron in the final layer. You wouldn't need to use an activation function at all.

1. What is backpropagation and how does it work? What is the difference between backpropagation and reverse-mode autodiff?

ANS:

Backpropagation is an algorithm used to train neural networks. It first computes the gradients of the cost function with regards to every model parameter then it performs a gradient descent step using these gradients.

This backpropagation step is performed until the model parameters converge to values that hopefully minimize the cost function.

Backpropagation refers to the whole process of training a neural network. Reverse-mode autodiff is a technique to compute the gradients efficiently. It is used by the backprop algorithm.

1. Can you list all the hyperparameters you can tweak in an MLP? If the MLP overfits the training data, how could you tweak these hyperparameters to try to solve the problem?

ANS:

In general, the hyperparameters of a neural network you can adjust are the number of hidden layers, the number of neurons in each hidden layer, and the activation function used by each neuron.

For binary classification, use the logistic activation function. For a multi-class problem, use softmax. For a linear regression problem, don't use an activation function.

Some simple ways to try and solve overfitting are reducing the number of hidden layers or the number of neurons.

1. Train a deep MLP on the MNIST dataset and see if you can get over 98% precision. Try adding all the bells and whistles (i.e., save checkpoints, restore the last checkpoint in case of an interruption, add summaries, plot learning curves using TensorBoard, and so on).

ANS:

**from** \_\_future\_\_ **import** division, print\_function, unicode\_literals

*# Common imports*

**import** numpy **as** np

**import** os

**try**:

*# %tensorflow\_version only exists in Colab.*

**%tensorflow\_version** 1.x

**except** Exception:

**pass**

*# to make this notebook's output stable across runs*

**def** reset\_graph(seed**=**42):

tf**.**reset\_default\_graph()

tf**.**set\_random\_seed(seed)

np**.**random**.**seed(seed)

*# To plot pretty figures*

**%matplotlib** inline

**import** matplotlib

**import** matplotlib.pyplot **as** plt

plt**.**rcParams['axes.labelsize'] **=** 14

plt**.**rcParams['xtick.labelsize'] **=** 12

plt**.**rcParams['ytick.labelsize'] **=** 12

*# Where to save the figures*

PROJECT\_ROOT\_DIR **=** "."

CHAPTER\_ID **=** "ann"

IMAGES\_PATH **=** os**.**path**.**join(PROJECT\_ROOT\_DIR, "images", CHAPTER\_ID)

os**.**makedirs(IMAGES\_PATH, exist\_ok**=True**)

**def** save\_fig(fig\_id, tight\_layout**=True**, fig\_extension**=**"png", resolution**=**300):

path **=** os**.**path**.**join(IMAGES\_PATH, fig\_id **+** "." **+** fig\_extension)

print("Saving figure", fig\_id)

**if** tight\_layout:

plt**.**tight\_layout()

plt**.**savefig(path, format**=**fig\_extension, dpi**=**resolution)