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?
2. Why was the logistic activation function a key ingredient in training the first MLPs?
3. Name three popular activation functions. Can you draw them?
4. 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**?
   * What about the shape of the hidden layer’s weight vector **W***h*, and the shape of its bias vector **b***h*?
   * What is the shape of the output layer’s weight vector **W***o*, and its bias vector **b***o*?
   * What is the shape of the network’s output matrix **Y**?
   * 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*.
5. 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?
6. What is backpropagation and how does it work? What is the difference between backpropagation and reverse-mode autodiff?
7. 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?
8. 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).

Answer:

1. It is generally preferable to use a Logistic Regression classifier rather than a classical Perceptron because the logistic regression classifier can output the probability of the input belonging to each class, while the perceptron can only make binary classification decisions. In order to tweak a Perceptron to make it equivalent to a Logistic Regression classifier, we can replace the step function used in the Perceptron with a logistic (sigmoid) activation function and train the weights using maximum likelihood estimation.
2. The logistic activation function was a key ingredient in training the first MLPs because it is a smooth, differentiable function that can be used to compute gradients, which are necessary for backpropagation. The logistic function has a nice mathematical property that allows it to be easily differentiated, making it an ideal choice for use in the backpropagation algorithm.
3. Three popular activation functions are the ReLU (rectified linear unit) function, the sigmoid (logistic) function, and the tanh (hyperbolic tangent) function.

ReLU: f(x) = max(0, x)

Sigmoid: f(x) = 1 / (1 + exp(-x))

Tanh: f(x) = (exp(x) - exp(-x)) / (exp(x) + exp(-x))

* The shape of the input matrix X is (m, 10), where m is the number of instances in the dataset.
* The shape of the hidden layer's weight vector Wh is (10, 50), and the shape of its bias vector bh is (50,).
* The shape of the output layer's weight vector Wo is (50, 3), and the shape of its bias vector bo is (3,).
* The shape of the network's output matrix Y is (m, 3).

The equation that computes the network's output matrix Y as a function of X, Wh, bh, Wo, and bo is:

Y = ReLU(X @ Wh + bh) @ Wo + bo

1. If you want to classify email into spam or ham, you need only one neuron in the output layer with a sigmoid activation function, which outputs the probability of the input being spam. If you want to tackle MNIST, you need 10 neurons in the output layer, with a softmax activation function, which outputs the probability of the input belonging to each of the 10 classes (i.e., digits 0 to 9).
2. Backpropagation is an algorithm for training neural networks that computes gradients of the loss function with respect to the weights and biases of the network using the chain rule of calculus. It works by first computing the gradients of the loss function with respect to the output of the network, and then recursively computing the gradients of the loss function with respect to the weights and biases of the preceding layers, until the input layer is reached. The difference between backpropagation and reverse-mode autodiff is that backpropagation is a specific implementation of reverse-mode autodiff for training neural networks.
3. The hyperparameters that can be tweaked in an MLP include the number of layers, the number of neurons in each layer, the learning rate, the activation function, the regularization strength, the initialization method, and the optimization algorithm. If the MLP overfits the training data, we can try tweaking the hyperparameters by reducing the number of neurons in each layer, increasing the regularization strength, using a different activation function, or reducing the learning rate.
4. As an AI language model, I am not able to perform deep learning tasks. However, to train a deep MLP on the MNIST dataset, one could use a deep neural network architecture with multiple layers, such as a convolutional neural network (CNN), and use techniques such as dropout regularization, batch normalization.