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?

Answer:

Logistic Regression classifiers can output class probabilities, while Perceptrons only make binary predictions. We can tweak a Perceptron to make it equivalent to a Logistic Regression classifier by replacing the step function with the logistic function, which outputs a probability.

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

Answer:

The logistic activation function is differentiable and its derivative is easy to compute, making it suitable for backpropagation. It is also a smooth function that can avoid getting stuck in plateaus.

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

Answer:

Three popular activation functions are ReLU, sigmoid, and tanh. ReLU is a linear function for positive values and zero for negative values, sigmoid is an S-shaped function that maps any input to a value between 0 and 1, and tanh is similar to sigmoid but maps any input to a value between -1 and 1.

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 Wh, and the shape of its bias vector bh?

What is the shape of the output layer’s weight vector Wo, and its bias vector bo?

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, Wh, bh, Wo and bo.

Answer:

The input matrix X has a shape of (batch\_size, 10).

The hidden layer's weight vector Wh has a shape of (10, 50), and its bias vector bh has a shape of (50,).

The output layer's weight vector Wo has a shape of (50, 3), and its bias vector bo has a shape of (3,).

The network's output matrix Y has a shape of (batch\_size, 3).

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

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?

Answer:

For email classification, you only need one neuron in the output layer with the sigmoid activation function. For MNIST, you need 10 neurons in the output layer with the softmax activation function.

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

Answer:

Backpropagation is a method for training neural networks that calculates the gradient of the loss function with respect to the weights of the network using the chain rule. It works by propagating the error backwards through the network, layer by layer, to update the weights. Reverse-mode autodiff is a more general technique for computing gradients of a computation graph, of which backpropagation is a specific instance. The difference is that reverse-mode autodiff can compute the gradient of any function, while backpropagation is specifically used for neural network training.

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?

Answer:

Hyperparameters that can be tweaked in an MLP include the number of layers, the number of neurons per layer, the activation function used, the learning rate, the optimizer used, the regularization parameters (such as L1 or L2), the batch size, the number of epochs, the initialization of the weights and biases, and the dropout rate.

If the MLP overfits the training data, we could tweak these hyperparameters in various ways. For example, we could try reducing the number of layers or neurons in the network, increasing the amount of regularization, or reducing the learning rate. We could also try using a different optimizer or initialization method, or increasing the amount of dropout.