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?

Logistic Regression is generally preferable to a classical Perceptron because it can produce probability estimates for binary classification problems, allowing it to make more nuanced predictions. A Perceptron, on the other hand, can only make binary decisions. To make a Perceptron equivalent to a Logistic Regression classifier, you can modify the output of the Perceptron by passing it through a logistic (sigmoid) function. This process transforms the Perceptron's output into a probability between 0 and 1, which is the key characteristic of Logistic Regression.

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

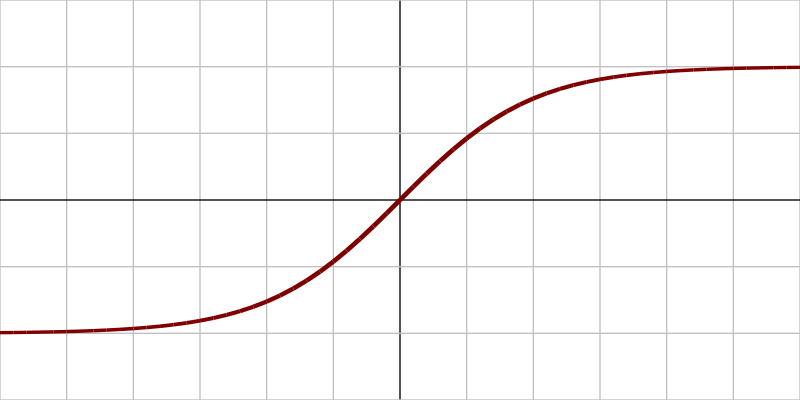
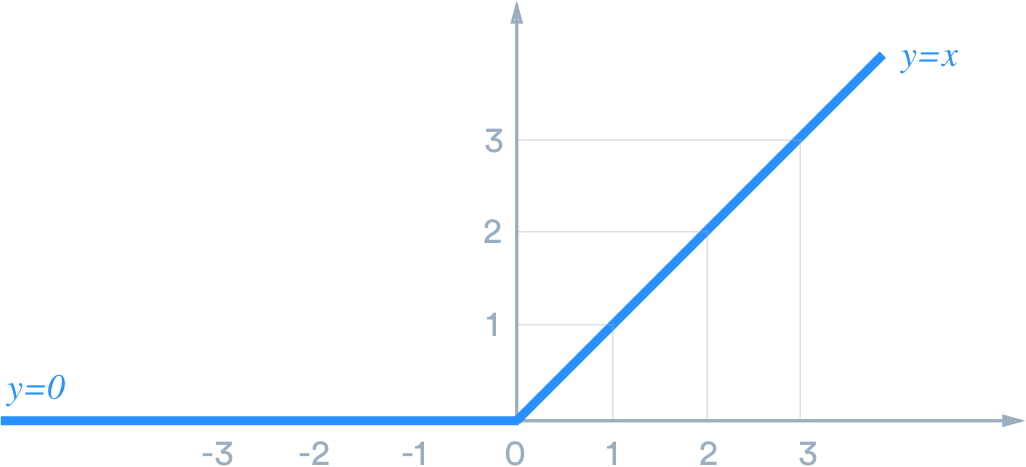
The logistic activation function was a key ingredient in training the first MLPs because it is differentiable, and its derivative can be computed easily. This differentiability is crucial for applying gradient-based optimization techniques like backpropagation, which is used to train neural networks. The logistic function has a smooth gradient, which allows for more stable and efficient training.

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

 Three popular activation functions are:

* Sigmoid (Logistic) function: f(x)=11+e−xf(x)=1+e−x1​
* Rectified Linear Unit (ReLU): f(x)=max⁡(0,x)f(x)=max(0,x)
* Hyperbolic Tangent (tanh) function: f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}

Here are plots of these functions:



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**?
   * 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*.

 For an MLP with ReLU activation:

* The shape of the input matrix XX is (m,10)(m,10) where mm is the number of instances.
* The shape of the hidden layer's weight vector WhWh is ((10, 50).
* The shape of its bias vector bhbh is ((50,).
* The shape of the output layer's weight vector WoWo is ((50, 3).
* The shape of its bias vector bobo is ((3,).
* The shape of the network's output matrix YY is ((m, 3).

The equation to compute the network's output matrix YY is: Y=ReLU(X⋅Wh+bh)⋅Wo+boY=ReLU(X⋅Wh+bh)⋅Wo+bo

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?

For classifying email into spam or ham, you typically need two neurons in the output layer, one for each class (spam and ham). You should use the softmax activation function in the output layer.

For MNIST, you need ten neurons in the output layer, one for each digit (0 through 9). The softmax activation function is also used in this case.

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

Backpropagation is the process of computing gradients with respect to the network's weights during training. It works by applying the chain rule of calculus to compute gradients layer by layer, from the output layer to the input layer. This allows for the optimization of the network's weights using gradient-based techniques. The difference between backpropagation and reverse-mode autodiff is that backpropagation is a specific method for computing gradients for neural networks, while reverse-mode autodiff is a more general automatic differentiation technique that can be used for various mathematical expressions.

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?

 Hyperparameters you can tweak in an MLP include:

* Learning rate
* Number of layers
* Number of neurons in each layer
* Activation functions
* Batch size
* Weight initialization
* Regularization techniques (e.g., dropout)
* Optimizer choice (e.g., SGD, Adam)
* Training epochs
* Early stopping criteria
* Learning rate scheduling

If the MLP overfits the training data, you can try:

* Reducing the number of neurons or layers
* Applying regularization (e.g., dropout)
* Reducing the learning rate
* Using early stopping

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).