HyperionDev - Tuorial - Neural Networks

May 26, 2024

Overview

This document describes a simple feedforward neural network with bias terms included. It covers the architecture of the neural network, forward pass computations, backward pass computations, regularisation techniques, and weight updates.

Although there are multiple libraries like TensorFlow, Keras, PyTorch, and Scikit-Learn, it is important to know the way, under the hood of how a neural network is trained.

This will allow you to at least describe it, understand it and eventually debug it if need be.

You should also be able to understand all the moving parts here, like the parameters involved: *learning rate*, *initialiser*, *regularisation* ... Most of the libraries have default parameters that may not suit your work. You must know what you are changing and how it affects training in a Machine Learning environment.

Neural Network Diagram

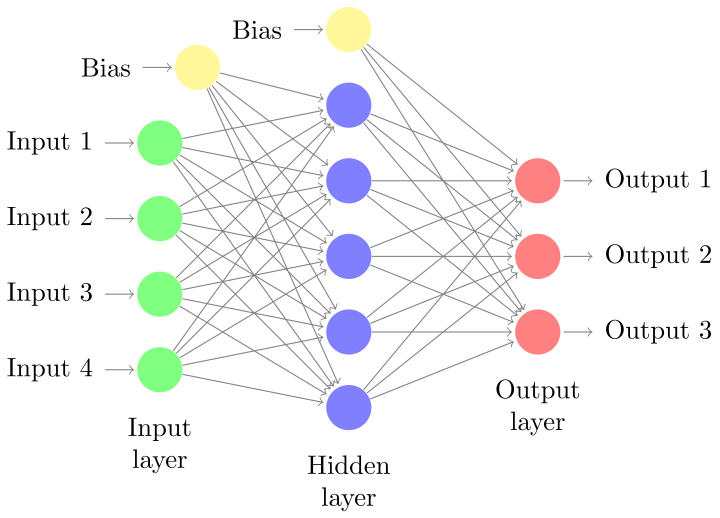

Figure 1: A simple feedforward neural network with one hidden layer. Bias nodes are shown in yellow.

Forward Pass

Given:

- Input data $\mathbf{X}$ of shape (m, n), where m is the number of samples and n is the number of input features.
- Weights and biases for the hidden layer: $\mathbf{W1}$ of shape (n, h) and $\mathbf{b1}$ of shape $(1, h)$, where h is the number of hidden units.
- Weights and biases for the output layer: $\mathbf{W2}$ of shape (h, c) and $\mathbf{b2}$ of shape $(1, c)$, where c is the number of classes.

1. Compute the input to the hidden layer:

$$\mathbf{Z1} = \mathbf{XW1} + \mathbf{b1}$$

2. Apply the sigmoid activation function to get the hidden layer activations:

$$\mathbf{A1} = \sigma(\mathbf{Z1}) = \frac{1}{1 + e^{-\mathbf{Z1}}}$$

3. Compute the input to the output layer:

$$\mathbf{Z2} = \mathbf{A1W2} + \mathbf{b2}$$

4. Apply the softmax activation function to get the output layer activations (probabilities):

$$\mathbf{A2} = \mathrm{softmax}(\mathbf{Z2}) = \frac{e^{\mathbf{Z2}}}{\sum_j e^{\mathbf{Z2_j}}}$$

Backward Pass

Given:

- The predicted outputs $\mathbf{A2}$ and the true labels $\mathbf{Y}$.

1. Compute the error at the output layer:

$$\mathbf{dZ2} = \mathbf{A2} - \mathbf{Y}$$

2. Compute the gradient for the output layer weights $\mathbf{W2}$:

$$\mathbf{dW2} = \frac{1}{m}\mathbf{A1}^T\mathbf{dZ2}$$

3. Compute the gradient for the output layer biases $\mathbf{b2}$:

$$\mathbf{db2} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{dZ2_i}$$

4. Backpropagate the error to the hidden layer:

$$\mathbf{dA1} = \mathbf{dZ2}\mathbf{W2}^T$$

5. Compute the gradient of the sigmoid activation function:

$$\sigma'(\mathbf{Z1}) = \mathbf{A1} \circ (1 - \mathbf{A1})$$

6. Compute the error at the hidden layer:

$$\mathbf{dZ1} = \mathbf{dA1} \circ \sigma'(\mathbf{Z1})$$

where $\circ$ denotes the element-wise product (Hadamard product).

7. Compute the gradient for the hidden layer weights $\mathbf{W1}$:

$$\mathbf{dW1} = \frac{1}{m}\mathbf{X}^T\mathbf{dZ1}$$

8. Compute the gradient for the hidden layer biases $\mathbf{b1}$:

$$\mathbf{db1} = \frac{1}{m}\sum_{i=1}^{m}\mathbf{dZ1_i}$$

Regularisation

L1 Regularisation

Add the derivative of the L1 regularisation term to the weight gradients:

$$\mathbf{dW2}+=\lambda\,\mathrm{sign}(\mathbf{W2})$$

$$\mathbf{dW1}+=\lambda\,\mathrm{sign}(\mathbf{W1})$$

L2 Regularisation

Add the derivative of the L2 regularisation term to the weight gradients:

$$\mathbf{dW2}+=\lambda\mathbf{W2}$$

$$\mathbf{dW1}+=\lambda\mathbf{W1}$$

Weight Updates

Update the weights and biases using gradient descent:

1. Update weights and biases for the hidden layer:

$$\mathbf{W1}-=\alpha\mathbf{dW1}$$

$$\mathbf{b1}-=\alpha\mathbf{db1}$$

2. Update weights and biases for the output layer:

$$\mathbf{W2}-=\alpha\mathbf{dW2}$$

$$\mathbf{b2}-=\alpha\mathbf{db2}$$