AI-Library

Adethya Srinivasan

July 14, 2024

Contents

1 High Level Implementation

Training a neural network can be broadly split into 4 distinct processes:

- Feed input
- Forward propagate
- Calculate error
- Backpropagate

  Repeat

We explain the specifics of each process in more detail below

1.1 Layers

Neural networks are composed of layers:

- Input layer
- Hidden layers
- Output layer

An input layer takes in a vector, which is generally a representation of input data (e.g. in the mnist example, each example fed into the input layer is a 784×1 vector which represents a 28×28 image of a handwritten digit).

The hidden layers is where the 'magic' happens and will be explored more in depth below. Just know that the non-linearity that allows the neural net to make predictions is introduced here.

The output layer is a vector containing the model prediction (e.g. in the mnsit example, the output layer is a 10×1 vector - we take the maximum value in the output vector to correspond to our prediction).

1.2 Activation Functions

We use activation functions to introduce non-linearity.

We model these activation functions as layers which have the same number of input neurons as output neurons. These activation functions are applied to the input to introduce complex behaviour in the neural network (otherwise we would have a bunch of linear operations, making our neural net an overly complicated linear regression model - though this behaviour can be modelled, if desired, using this framework).

1.3 Forward Propagation

Forward propagation is essentially the input vector passing through the layers of our neural network to output a prediction

1.4 Error Calculation

There are many different error calculations but these all will in some way involve calculating the difference between the target value (the true value) and the model's prediction. We use this calculation to adjust weights and biases during the backpropagation step.

1.5 Backpropagation

The backpropagation step is the crucial step which allows our model to learn. We do this by calculating the derivatives of our error with respect to output vector, weight matrix, bias vector and input, and adjusting our weights and biases accordingly, 'stepping' towards a better model - adjustments to the model are made by a 'step size' or 'learning rate' each epoch. The derivative with respect to the input vector is calculated because this is the output vector to the layer before. We calculate this and repeat the process until we reach our original input vector. The first calculation for the derivative of the output vector is the derivative of the loss function which we have already calculated.

1.6 NB

Any formal proofs will be indicated as such, but generally the maths used here can serve as a basis for intuition for many relevant proofs. The maths shown should primarily be used to aid intuition and understanding of neural networks, not as rigorous mathematical proofs themselves.

2 Fully-connected/Dense layer

A fully-connected or dense layer is a layer in which all input neurons are connected to all output neurons.

2.1 Forward Propagation

Forward propagation is the multiplication of the transpose of the weight matrix with the input vector plus the bias vector. This is equivalent to the dot product of the weight matrix and the input vector plus the bias vector. Written formally:

$$\begin{aligned} Y &= W^T X + B \\ &= W \cdot X + B \end{aligned}$$

2.2 Backpropagation

We will expand the forward propagation step for reasons that will become clear.

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_j \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1i} \\ w_{21} & w_{22} & \cdots & w_{2i} \\ \vdots & \vdots & \vdots & \vdots \\ w_{j1} & w_{j2} & \cdots & w_{ji} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_j \end{bmatrix}$$

So:

$$\begin{aligned} y_1 &= x_1 w_{11} + x_2 w_{12} + \cdots + x_i w_{1i} + b_1 \\ y_2 &= x_1 w_{21} + x_2 w_{22} + \cdots + x_i w_{2i} + b_2 \\ &\vdots \\ y_j &= x_1 w_{j1} + x_2 w_{j2} + \cdots + x_i w_{ji} + b_j \end{aligned}$$

We are given the derivative with respect to the output and we use this to calculate the derivative with respect to the weights and biases and the input.

We begin by observing that the derivative of the error with respect to the weight matrix $\frac{dE}{dW}$ can be thought of as a matrix of partial derivatives with respect to individual elements:

$$\frac{dE}{dW} = \begin{bmatrix} \frac{\partial E}{\partial w_{11}} & \frac{\partial E}{\partial w_{12}} & \cdots & \frac{\partial E}{\partial w_{1i}} \\ \frac{\partial E}{w_{21}} & \frac{\partial E}{w_{22}} & \cdots & \frac{\partial E}{w_{2i}} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial E}{w_{j1}} & \frac{\partial E}{w_{j2}} & \cdots & \frac{\partial E}{w_{ji}} \end{bmatrix}$$

Taking a closer look at $\frac{\partial E}{\partial w_{11}}$ we see that:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_1}\frac{\partial y_1}{\partial w_{11}} + \frac{\partial E}{\partial y_2}\frac{\partial y_2}{\partial w_{11}} + \cdots + \frac{\partial E}{\partial y_j}\frac{\partial y_j}{\partial w_{11}}$$

We see that the $\frac{\partial y_1}{\partial w_{11}} = x_1$ and the remaining terms are 0. Hence:

$$\frac{\partial E}{\partial w_{11}} = \frac{\partial E}{\partial y_1} x_1$$

More generally:

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial y_j} x_i$$

Rewriting $\frac{dE}{dW}$:

$$\begin{bmatrix} \frac{\partial E}{\partial y_1}x_1 & \frac{\partial E}{\partial y_1}x_2 & \cdots & \frac{\partial E}{\partial y_1}x_i \\ \frac{\partial E}{y_2}x_1 & \frac{\partial E}{y_2}x_2 & \cdots & \frac{\partial E}{y_2}x_i \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial E}{y_j}x_1 & \frac{\partial E}{y_j}x_2 & \cdots & \frac{\partial E}{y_j}x_i \end{bmatrix}$$

We can write this as a matrix multiplication:

$$\frac{dE}{dW} = \begin{bmatrix} \frac{\partial E}{\partial y_1} \\ \frac{\partial E}{\partial y_2} \\ \vdots \\ \frac{\partial E}{\partial y_j} \end{bmatrix} \begin{bmatrix} x_1 & x_2 & \cdots & x_i \end{bmatrix}$$

$$= \frac{dE}{dY} \, X^T$$

Hence we've shown:

$$\frac{dE}{dW} = \frac{dE}{dY} \, X^T$$

Using similar reasoning, we can show that:

$$\frac{dE}{dB} = \frac{dE}{dY} \text{ and } \frac{dE}{dX} = \frac{dE}{dY} W^T$$

3 Activation Functions

The activation function introduces non-linear behaviour in our neural net architecture (e.g. tanh, logistic function). We model this as a layer which applies the activation function to each input neuron to produce an output layer with the same number of neurons.

3.1 Forward Propagation

Forward propagation is simply the application of function to each input neuron.

3.2 Backpropagation

The derivative of the error with respect to the output is given so we want to find the derivative with respect to the input (note that there is no weight matrix involved in the applying the activation function). So we have $\frac{dE}{dY}$ and want to find $\frac{dE}{dX}$.

$$\frac{dE}{dY} = \begin{bmatrix} \frac{\partial E}{\partial y_1} \\ \frac{\partial E}{\partial y_2} \\ \vdots \\ \frac{\partial E}{\partial y_i} \end{bmatrix}$$

We take the derivative of the error with respect to individual elements. We will take a closer look at the derivatice with respect to x_1:

$$\begin{aligned}\frac{\partial E}{\partial x_1} &= \frac{\partial E}{\partial y_1}\frac{\partial y_1}{\partial x_1} + \frac{\partial E}{\partial y_2}\frac{\partial y_2}{\partial x_1} + \cdots + \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial x_1} \\ &= \frac{\partial E}{\partial y_1}\frac{\partial y_1}{\partial x_1}\end{aligned}$$

So generally:

$$\frac{\partial E}{\partial x_i} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial x_i}$$

We can see that this is the Hadamard product of the matrices of partial derivatives of the error with respect to the input and output:

$$\frac{dE}{dX} = \frac{dE}{dY} \circ \frac{dY}{dX}$$