Introduction to Feedforward Neural Networks

1 Fully Connected Layers

A feedforward neural network is composed of layers of artificial neurons. The most basic building block of these networks is the **fully connected layer**. In such a layer, every neuron is connected to every neuron in the preceding layer through a set of weights. Mathematically, the output of a fully connected layer is represented as:

$$\mathbf{z} = \mathbf{W}\mathbf{x} + \mathbf{b},$$

where:

- $\mathbf{x}$ is the input vector,
- $\mathbf{W}$ is the weight matrix,
- $\mathbf{b}$ is the bias vector,
- $\mathbf{z}$ is the resulting output vector.

2 Activation Functions

An activation function introduces non-linearity into the network, enabling it to model complex relationships in data. Common activation functions include:

- **Sigmoid:** $\sigma(x) = \frac{1}{1+e^{-x}}$
- **Tanh:** $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- **ReLU (Rectified Linear Unit):** $f(x) = \max(0, x)$
- **Leaky ReLU:** $f(x) = \max(\alpha x, x)$, where α is a small positive constant.

3 Multi-Layer Perceptron (MLP)

A **multi-layer perceptron (MLP)** consists of multiple fully connected layers, each followed by an activation function. The input is transformed layer by layer until the output layer produces the final predictions. The network can be mathematically described using the forward equations:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)},$$
$$\mathbf{a}^{(l)} = f(\mathbf{z}^{(l)}),$$

where l denotes the layer index, and f is the activation function.

4 Gradient-Based Optimization

Training a feedforward neural network involves optimizing its parameters ($\mathbf{W}$ and $\mathbf{b}$) to minimize a **loss function**. Gradient-based optimization methods, such as stochastic gradient descent (SGD), are commonly used. The gradients are computed using **back-propagation**, which efficiently calculates partial derivatives of the loss with respect to all parameters.

5 Common Loss Functions

The choice of loss function depends on the task at hand. Popular loss functions include:

- **Mean Squared Error (MSE):**

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2,$$

  where y_i and $\hat{y}_i$ are the true and predicted values, respectively.

- **Binary Cross-Entropy Loss:**

$$\text{BCE} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right].$$

- **Categorical Cross-Entropy Loss:**

$$\text{CCE} = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_{ij}\log(\hat{y}_{ij}),$$

  where k is the number of classes.

6 Generic Training Procedure

Training a feedforward neural network involves the following steps:

1. **Sample a mini-batch of data:** Select a subset of training data for computational efficiency.
2. **Forward pass:** Compute the predictions by propagating the input through the network.
3. **Compute the loss:** Evaluate the chosen loss function based on the predictions and ground truth.
4. **Backpropagate:** Calculate gradients of the loss with respect to all network parameters using backpropagation.
5. **Update the parameters:** Adjust the parameters using a gradient-based optimization method, such as SGD or Adam.

7 Additional Reading

For additional resources, the book "Deep Learning" by Aaron Courville, Ian Goodfellow, Yoshua Bengio, full pdf: `http://alvarestech.com/temp/deep/Deep%20Learning%20by%20Ian%20Goodfellow,%20Yoshua%20Bengio,%20Aaron%20Courville%20(z-lib.org).pdf`. You will find Chapter 6 most useful, you can find the pdf for this in the folder `extra_reading`. I have also provided part of Chapter 2 of my Master's Thesis in this folder, this presents the same details but it is possibly less dense.