Lecture 13: Review Notes

13 Neural Networks and Deep Learning - Overview

13.1 Why do we need neural networks?

In general, given $\boldsymbol{D} = \{(\boldsymbol{x_i}, y_i), i \in 1, ..., N\}$, $\boldsymbol{x_i} \in \boldsymbol{R^d}$, and $y_i \in \boldsymbol{R}$,we wish to find out the relationship $\boldsymbol{F}$ between $\boldsymbol{x_i}$ and y_i. we can assume some functional form and do parameterized fitting by minimizing the loss(shown in (1)) but this leads to either under fit or overfit. This is because the parameterized functional form that we have assumed may not be the representative of $\boldsymbol{F}$ (some complicated non linear function) between $\boldsymbol{x_i}$ and y_i . Here comes the neural networks that will allow us to find out these complex relationships without assuming any particular functional form.

$$\underset{\theta}{\text{minimize}} \sum_{i=1}^{N} (y_i - \boldsymbol{F_\theta(x_i)})^2 \tag{1}$$

$$\boldsymbol{F_\theta} \approx \boldsymbol{F} \tag{2}$$

13.2 Basics - General Framework of neural networks

The basic form of the neural network is given by the stacking of linear block,nonlinear activation block,linear block,etc. This is shown in Figure 1. Choice of activation function,input and output dimensions of the linear blocks are hyper parameters that can be chosen depending on the task.

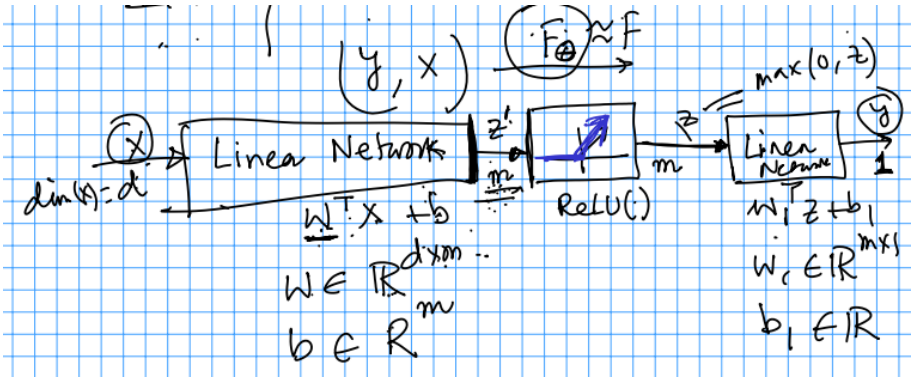

Figure 1: Linear-Activation-Linear Block

The equations for the feed forward propagation of linear-activation-linear block are,

$$\boldsymbol{z' = W^T x + b} \tag{3}$$

$$\boldsymbol{z = f(z')} \tag{4}$$

$$\boldsymbol{y' = W_1^T z + b_1} \tag{5}$$

$$\boldsymbol{L = \sum_{i=1}^{N} (y_i' - y_i)^2} \tag{6}$$

The gradient of loss function $\boldsymbol{L}$ with respect to weights and biases are computed using backpropagation(which uses chain rule) and gradient descent is used to minimize the loss function. The equations for updating the parameters are shown below.

$$w := w - \epsilon \frac{\partial L}{\partial w} \tag{7}$$

$$b := b - \epsilon \frac{\partial L}{\partial b} \tag{8}$$

According to universal function approximation theorem, any nonlinear function can be approximated by stacking together these blocks.

13.3 Recurrent Neural Networks

Recurrent neural networks, also known as RNNs, are a class of neural networks that allow previous outputs to be used as inputs while having hidden states($\boldsymbol{h}$). These are used in several sequence prediction problems such as stock market prediction,traffic forecasting,dynamical system identification etc.

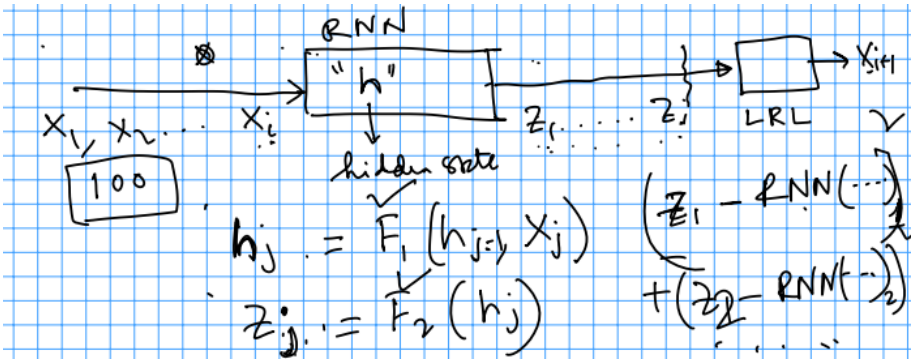

Figure 2: Basic RNN

RNNs can simulate any algorithm. Backpropagation through time(BPTT) an application of backpropagation is used to train RNN.BPTT works by unrolling all input time steps.