Introduction and Building Blocks of Deep Learning for Time Series

Md Mohidul Haque

19th January, 2024

1 Foundations of Deep Learning

Deep learning involves automatic feature learning from raw data. It seeks to exploit the unknown structure in the input distribution to discover a good representation of the Data Generating process of the input data. Building a deep learning model is assembling parameterized into dynamic graphs and optimizing it with gradient-based methods. The key points are:

1. Assembling parameterized modules: Deep learning systems composed of a few submodules with a few parameters assembled into a graph-like structure.
2. Optimizing it with gradient-based methods: it can be empirically seen that most successful deep learning systems are trained using gradient-based optimization methods.

1.1 Reasons behind the popularity

Deep learning has gained a lot of popularity because of the two main reasons:

1. Increase in computing availability:
   - Computer hardware showed more improvement
   - The introduction of GPUs
2. Increase in data availability:
   - Amount of the data increasing because of the cheap cost of data storage
   - Data collection is easier than before
   - With more data, deep learning starts to outperform traditional machine learning

1.2 Perceptron

The fundamental unit of the human brain is called a neuron. Perceptron is the fundamental building block of all neural networks. Perceptron is designed to mimic a neuron.

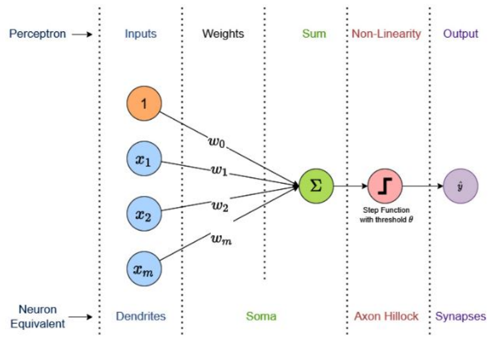

Figure 1: Perceptron Structure

Perceptron has the following components:

- Inputs: Real-valued inputs that are fed to a Perceptron. This is like the dendrites in neurons that collect the input.
- Weighted Sum: Each input is multiplied by a corresponding weight and summed up. The weights determine the importance of each input in determining the outcome. It is like soma in a neuron.
- Non-linearity: The weighted sum goes through a non-linear function. This is like Axon Hillock in a neuron.
- Output: It supplies the output as Synapses.

The mathematical form of a Perceptron:

$$\hat{y} \ = g(w_0 + \sum_{i=1}^{m} \ x_i w_i) \tag{1}$$

Equation (1) can be written as a vector form:

$$\hat{y} \ = g(\mathbf{X}_T \mathbf{W}) \tag{2}$$

2 Deep Learning System Components

Deep learning is a modular system. Deep learning is not just one model, but rather a language to express any model in terms of a few parametrized modules with these specific properties:

1. It should be able to produce an output from a given input through a series of computations.
2. If the desired output is given, they should be able to pass on information to its inputs on how to change, to arrive at the desired output.

To optimize these kinds of systems, we predominantly use gradient-based optimization methods. These parameterized modules should be differentiable functions. The most popular deep learning system paradigm starts with raw input data. The raw input data goes through blocks of linear and non-linear functions.

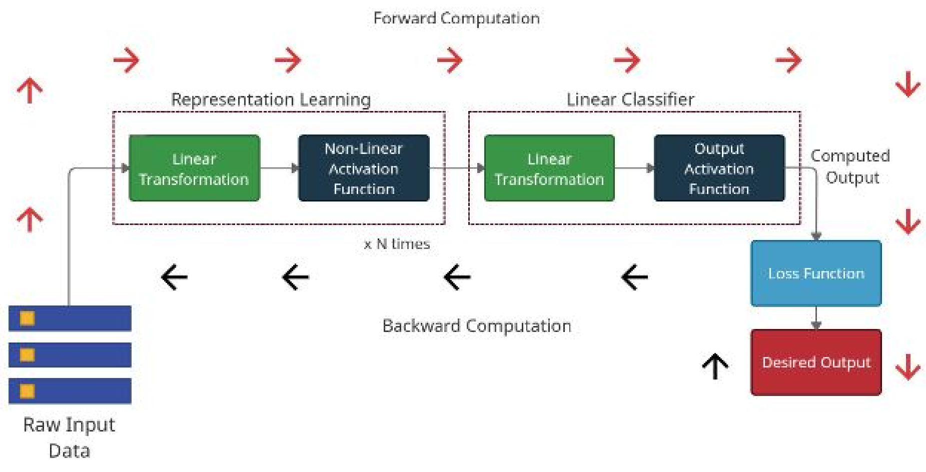

Figure 2: A Deep Learning System

2.1 Representation Learning Block

This is the first block of the deep learning system, consisting of linear transformation and non-linear activation functions. The overall function of this block is to learn a function, which transforms the raw input into good features that make non-linear problems linearly separable.

2.1.1 Linear Transformation

Linear transformation in a neural network context means affine transformations. A linear transformation fixes the origin, but an affine transformation moves the vector space. Multiple linear transformations work as a single transformation. Non-linearities are introduced by using a non-linear function, called the activation function.

2.1.2 Non-Linear Activation Functions

Activation functions are non-linear differentiable functions. It transforms linearly inseparable input vector space to a linearly separable vector space. Examples of popular activation functions, where x is the input space:

Example 2.1 ***Sigmoid***

$$g(x) = \frac{1}{1+e^{-x}}$$

Characteristics:

- Most common, and known as the logistic function
- Continuous function and therefore it is differentiable everywhere
- The derivative is also computationally simpler to calculate.
- Squashes the input between 0 and 1

Main drawback:

- Saturating of the activation: the gradients tend to zero on the flat portions of the sigmoid

Example 2.2 ***Hyperbolic Tangent (*****tanh*****)***

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)}$$

Characteristics:

- Can express tanh as a function of sigmoid
- Outputs value is in between -1 and 1, and symmetrical around the origin

Main drawback:

- Saturating function

Example 2.3 ***ReLU***

$$g(x) = \max(x, 0)$$

Characteristics:

- A linear function, with a kink at zero
- Any value greater than zero is retained as is, but all values below zero are squashed to zero

- This squashing is what gives the non-linearity to the activation function
- Squashes the input between 0 and 1
- Non-saturating function
- Computations of activation function and its gradients are cheap
- Training is faster with this function
- Sparsity in the network: having the activation as zero, a large majority of neurons in the network can be turned off

Main drawbacks:

- Dead ReLUs: if the input is less than zero, then the gradients become zero, and the unit will not learn anymore
- Average output of a ReLU unit is positive and when we stack multiple layers, this might lead to a positive bias in the output.

Example 2.4 ***Leaky ReLU***

Characteristics:

- No dead ReLUs: makes sure the gradients are not zero when input is less than zero

Main drawback:

- Sparsity is lost: no zero output that turns off a unit

2.2 Linear Classifier

This is the second block of the deep learning system, consisting of linear transformation and output activation functions.

2.2.1 Output Activation Functions

This kind of function has a deeper connection with maximum likelihood estimation (MLE) and the chosen loss function. A linear activation function is used for regression. Sigmoid and tanh activation functions are used for binary classification, whereas the softmax activation function is more suitable for multiclass classification.

Example 2.5 ***Softmax (K Classes)***

$$\text{Softmax}(\mathbf{x}_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}}$$

Characteristics:

- Standard output activation for multiclass classification problems
- Converts the raw output from a network into something that resembles a probability across possible classes

2.3 Loss Function

The last major component in the deep learning system. The loss function is a way to tell how good the predictions of the model are. Loss function should be differentiable in the deep learning context.

3 Train Deep Learning System

Forward propagation gets the output, and by using the loss function we can measure how far we are from the desired output. Then backward propagation is used to calculate the gradient concerning all the parameters. With the gradient of the loss function, gradient descent helps us to optimize the loss function. Negative gradient is used to minimize a loss function, which will point us in the direction of the steepest descent. The learning rate multiplied by the gradient defines the step we take in each iteration. After each iteration, all the parameters in the network are updated to get the optimal loss. There are three popular variants of gradient descent:

1. Batch gradient descent
2. Stochastic gradient descent (SGD)
3. Mini-batch gradient descent

4 Deep Learning for Time Series

Deep learning learns good features using representation learning and then uses the learned features to map history to the future. This idea can be refined to the time series perspective by using the encoder-decoder paradigm. Latent space is an abstract vector space that encodes a meaningful internal representation of the feature/input space. An encoder-decoder architecture has two main parts:

1. Encoder: takes input vector, and encodes it into a latent space, consumes the history of time series
2. Decoder: decodes latent vector/space into output, generates the forecast

4.1 Feed-forward Networks (FFNs)

Most basic encoder-decoder paradigm, fully connected neural networks for forecasting time series. It is created by stacking multiple perceptions.

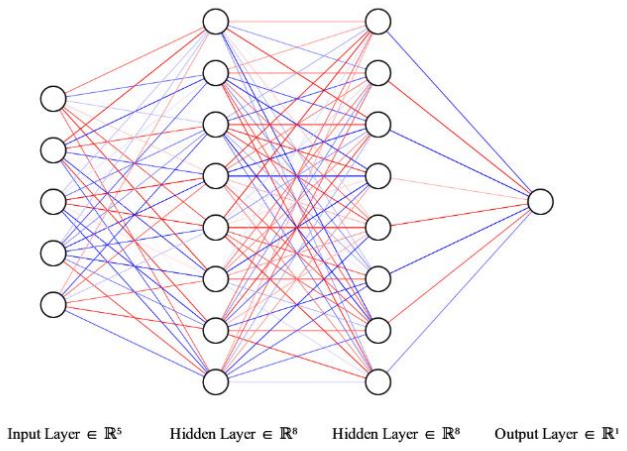
Input Layer $\in \mathbb{R}^5$
Hidden Layer $\in \mathbb{R}^8$
Hidden Layer $\in \mathbb{R}^8$
Output Layer $\in \mathbb{R}^1$

Figure 3: Feed-forward network

Characteristics:

- Takes a fixed-size input vector and passes it through a series of computational layers leading up to the desired output
- Information is fed forward through the network
- Fully connected network