COMP 642 — Machine Learning Jan 31st, 2023

Lecture 7

Lecturer: Anshumali Shrivastava

Scribe By: Yuhan Wei, Xiaoyu Chen, Yijun Zhou, Wanying Xu

Disclaimer: *These lecture notes are intended to develop the thought process and intuition in machine learning. The materials are not thoroughly reviewed and can contain errors.*

1 Deep Neural Networks

Machine Learning is a framework to automate (through algorithms), statistical models, like a linear regression model, to get better at making predictions, which had to get developed. Though works very well for structured data, machine learning mostly performs poorly with image, audio, and other unstructured data types. The learning portion of creating models spawned the development of artificial neural networks (ANNs).

1.1 Artificial Neural Networks

Artificial neural networks work similarly to the human brain's neural networks, which is represented as a hierarchical (layered) organization of neurons (similar to the neurons in the brain) with connections to other neurons. Each neuron is a known as perceptron and is similar to multiple linear regression. The perceptron feeds the signal produced by a multiple linear regression into an activation function that is nonlinear.

Figure 1 represents an artificial neural network with one hidden layer. The neural network has three main components: an input layer, a processing layer (hidden), and an output layer.

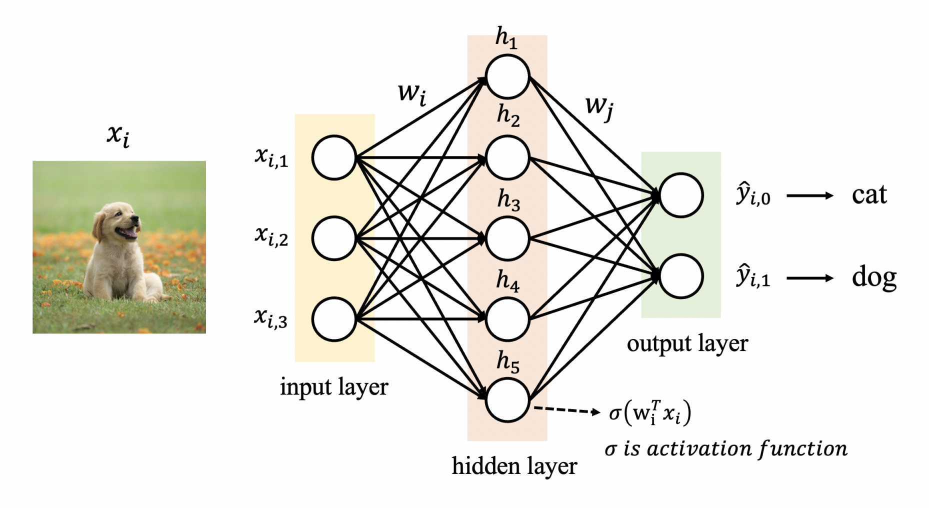

Figure 1: ANN with one hidden layer

First, the input data (x) is consumed by the input layer which then provides an output to the neurons within the next hidden layer which provides the final output ($\hat{y}$). The output might be a prediction like "Dog" or "Cat" represented in the probability of being each class.

Each layer can have one or many neurons and each will compute a small function i.e. activation function (σ), which adds non-linearity to the network. The connections between two successive layers would have associated weight (w). The weight defines the influence of the input on the output for the next neuron and eventually for the overall final output. In a neural network, the initial weights would be all random but during the model training, these weights are updated iteratively to learn to predict a correct output.

1.2 Deep Neural Networks

A deep neural network (DNN) is an ANN with multiple hidden layers between the input and output layers (figure 2). Similar to shallow ANNs, DNNs can model complex non-linear relationships without explicit formulas. Same to ANN, the neurons in DNN pass the signal to the next layer's neurons based on the received input and form a complex network that learns with some feedback mechanism (model training).

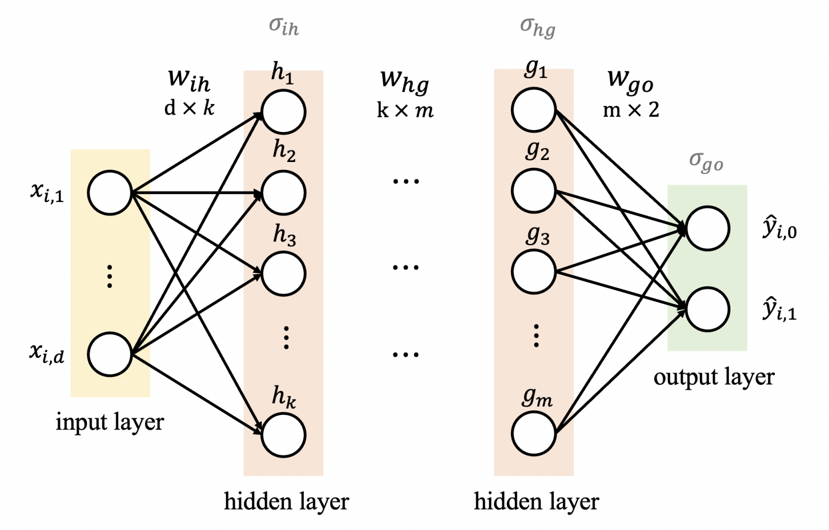

Figure 2: Deep neural network

Consider a deep neural network with 2 hidden layers h and g, the forward pass (forward propagation) of this network is

$$h_i = \phi(x_i) = \sigma_{ih}(x_i w_{ih})$$

$$g_i = \phi(h_i) = \sigma_{hg}(h_i w_{hg})$$

$$\hat{y}_i = \phi(g_i) = \sigma_{go}(g_i w_{go})$$

where,
σ_{ih}, w_{ih} are activation function and weights between input and hidden layer h,
σ_{hg}, w_{hg} are activation function and weights between hidden layer h and hidden layer g,
σ_{go}, w_{go} are activation function and weights between hidden layer g and output layer.

2 Activation Functions in Neural Networks

The active function is used to figure out the output for the next layer of nodes, which is also named as **transfer function**.
The activation functions are mainly divided into two types:

- Linear activation function(Figure 1), the output will not be confined and keep linear. In other words, it will not benefit multi-layer perceptron like the complex neural network.
- Nonlinear activation function could provide a wide range of outcome($[-\infty, \infty]$) through differential and monotonic function. There are commonly logistic, ReLU and Sigmoid functions, etc. The Logistic function is covered in previous lectures.

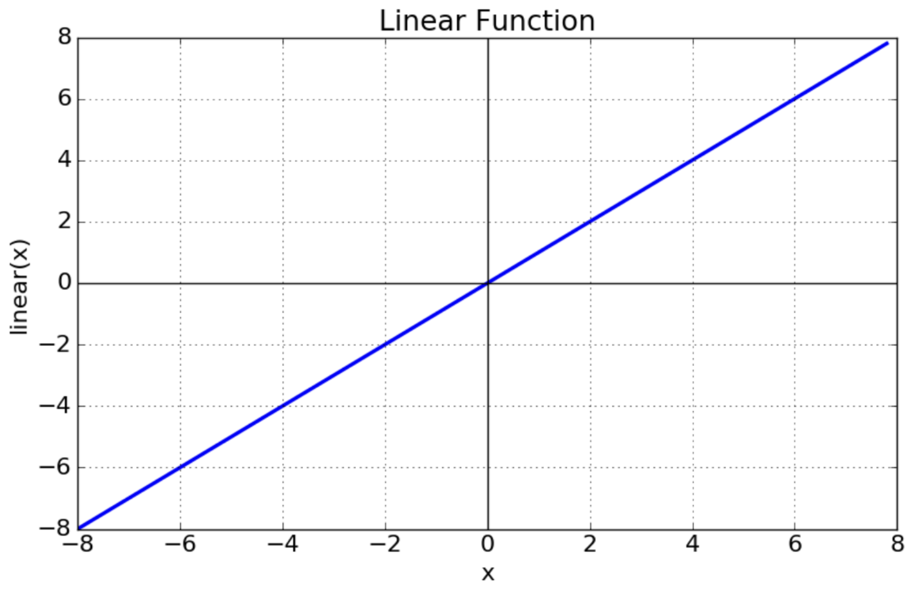

Figure 3: Linear Activation Function

2.1 ReLU(Rectified Linear Unit) Activation Function

The general formula for ReLU is

$$\sigma(x) = \begin{cases} x & x > 0 \\ 0 & \text{otherwise} \end{cases}$$

Given the figure and formula, it is easy to conclude that the output of the input directly if it is positive, otherwise, it will output zero. The output range ($[0, 1]$)

ReLU has several advantages, such as its simplicity, efficiency, and improved training speed compared to other activation functions. The ReLU activation function is widely used for a variety of data types, but it is especially well-suited for non-linear problems, such as image recognition, high-dimensional data, sparse data and non-linear problems.

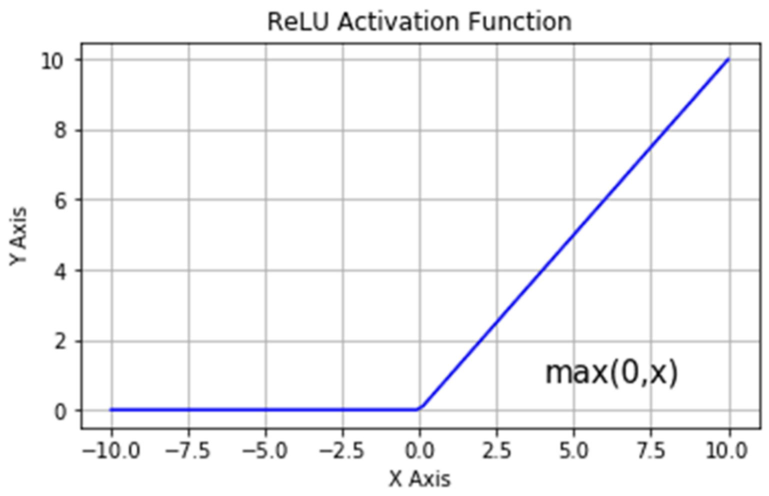

Figure 4: ReLU activation function

2.2 Sigmoid Activation Function

The general formula for Sigmoid is

$$\sigma(x) = \frac{1}{1 + exp(-x)}$$

Based on the formula, it is differentiable. Besides this, the function is monotonic. These advantages contribute to the generalization or adaption of data and output. The output is also from $[0, 1]$. Another one is called Tanh (hyperbolic tangent) function which ranges $[-1, 1]$. These two activations are usually used in feed-forward nets.

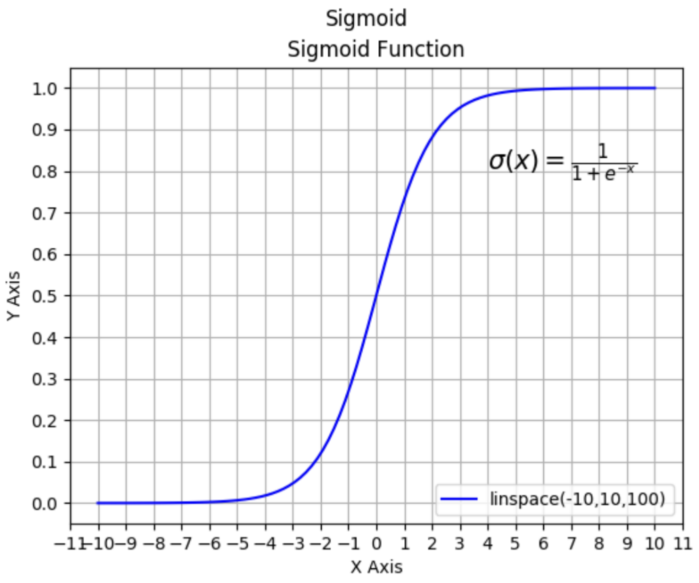

Figure 5: Sigmoid activation function

3 How to Train Model

The main idea of training a model is to minimize the loss function. We can use the Stochastic Gradient Descent Algorithm to make loss function approaches 0. In stochastic (or "on-line") gradient descent, the true gradient of Q(w) is approximated by a gradient at a single sample:

$$w := w - \eta \nabla Q_i(w).$$

In pseudocode, stochastic gradient descent can be presented as

- Choose an initial vector of parameters w and learning rate η.
- Repeat until an approximate minimum is obtained:
- Randomly shuffle samples in the training set.
- For i=1,2,3,4...n, do:
- $w := w - \eta \nabla Q_i(w)$.

3.1 Forward Propagation

In order to train the model, it is important to discuss how the neural network calculates its output given a set of inputs. Forward propagation can be visualized as a long series of nested equations. Such as:

$$g(x_i) = \sigma_1(\sigma_2(x_i * w_1) * w_2)$$

, which means we have 2 linear layers and two non-linear layers. The σ_1, σ_2 means the activation function.

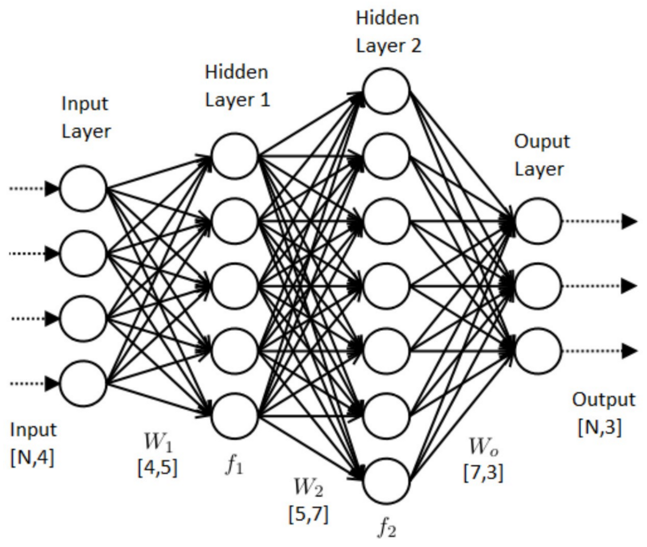

Figure 6: Forward Propagation

Forward propagation is where input data is fed through a network, in a forward direction, to generate an output. The data is accepted by hidden layers and processed, as per the activation function, and moves to the successive layer. During forward propagation, pre-activation and activation take place at each hidden and output layer node of a neural network. The pre-activation function is the calculation of the weighted sum.

3.2 Backward Propagation

In order to be trained, a neural network relies on both forward and backward propagation. Backpropagation is used in machine learning and data mining to improve prediction accuracy through backward propagation calculated derivatives. Backward Propagation is the process of moving from the right (output layer) to the left (input layer).

To calculate the partial derivative from multilayers, we need to use the chain rule. The chain rule states that the derivative of f(g(x)) is f'(g(x))*g'(x), which means we need to calculate partial derivatives from back to forward, multiply the back derivative f'(g(x)) by the previous one g'(x) and use it to update the weights. The whole process is:

- Find the partial derivative $\nabla Q_i(w)$ of layer k
- Update the weights of layer k with $w := w - \eta \nabla Q_i(w)$.
- Store the derivative in the memory and pass it to previous layer k-1
- Multiply the partial derivative of layer k-1 by the derivative of layer k and update weights accordingly.

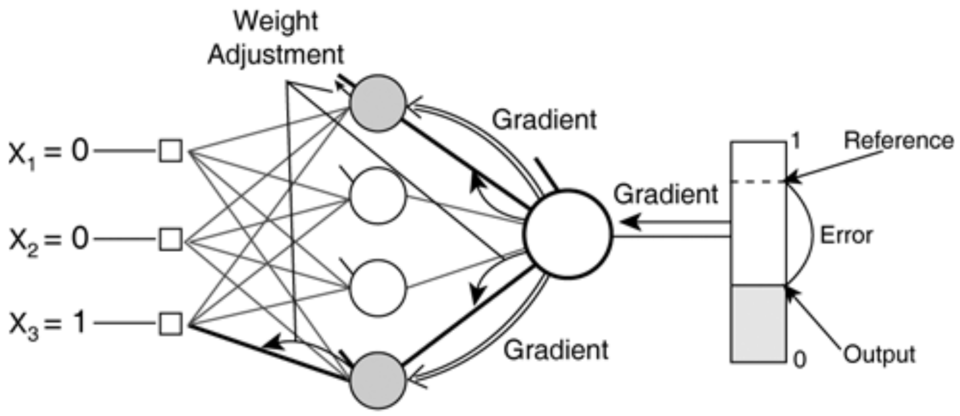

Figure 7: Backward Propagation