Lecture 1. Introduction to deep learning

Young, Shen

August 11, 2024

1 Models

Considering using a linear function to predict the number of views of my youtube channel.

$$y = b + wx$$

Where:

- y: the number of views **predicted** for tomorrow;
- x: the number of views we **already have** today (feature);
- w: weight
- b: bias

However, most of the times this prediction won't behave well. Maybe considering not only today but also several days before to make the prediction will be better:

$$y = \sum_{i=1}^{n} w_i x_i + b$$

Most of the time, the original function we want to fit is not linear, which is the simplest case. Sometimes, the original function may look like this:

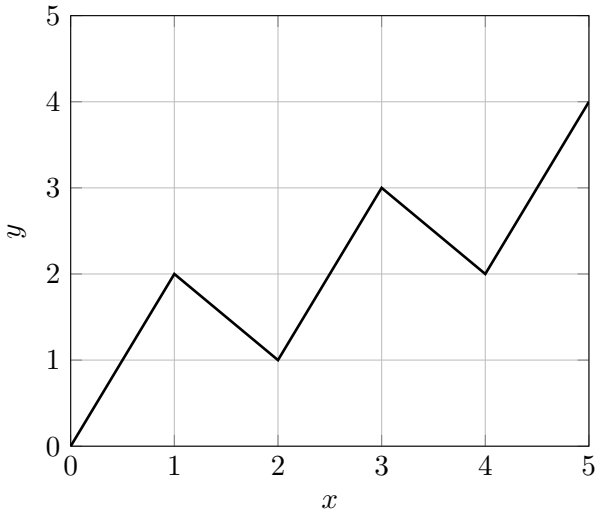

Fig. 1. Polyline

It is obvious that this function can't be expressed as $y = b + wx$, no matter what w or b is used. In this case, we need to use several hard sigmoid function(2) to combine. By changing

the gradient of the hard sigmoid function, we can use it to fit each "linear part" of the polyline. In this case (1), a total of 5 hard sigmoid funtion is needed, since there are 5 linear parts in this polyline. However, using hard sigmoid function is not convenient for some further operations on it. We usually use sigmoid function instead:

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

By changing the value of b, sigmoid function moves on x-axis, while changing the value of w

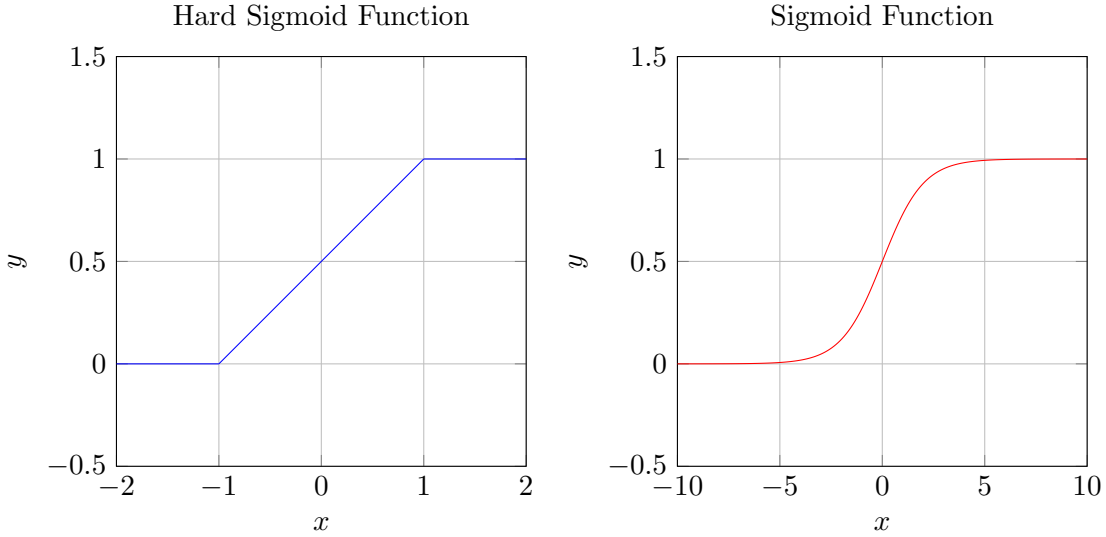

Fig. 2. Hard Sigmoid and Sigmoid Functions

influences the steepness of the function. Multiplying the function by c changes the maximum and minimum of it.

With this, we can fit the polyline in this form:

$$y = \sum_{i=1}^{6} c_i \sigma(w_i x + b_i) + b$$

Where b indicates beginning y-value, in this case, 0.

But what if the function is a smooth function(3)? There don't seem to be turning points any more.

In fact, calculus tells us that if you divide it into infinite parts, each part can be represented by a hard sigmoid function. For the same reason before, we use sigmoid function instead.

$$y = \sum_{i=1}^{\infty} c_i \sigma(w_i x + b_i) + b$$

In practice, we only use finite sigmoid function to model. The more function you use to model the original function, the more accurate it will be. (Be careful about the overfitting problem)

Aside from sigmoid function, another function that is often used for modelling original function is called ReLU(Rectified Linear Unit) :

$$ReLU(x) = max(0, x)$$

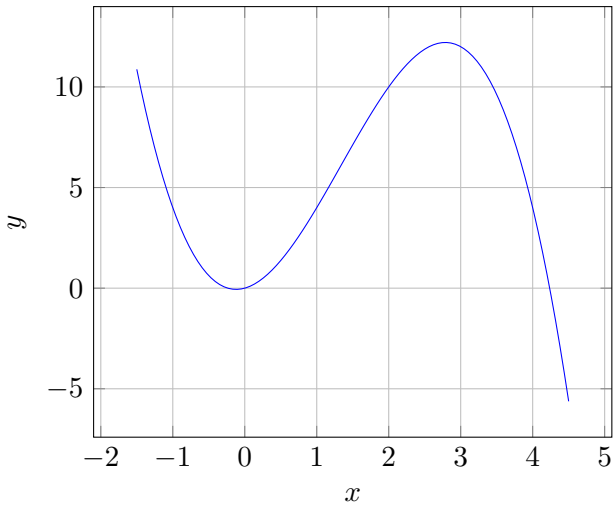

Fig. 3. Smooth Function

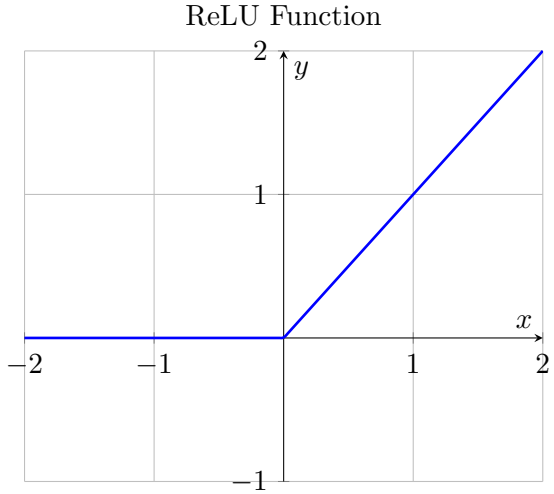

Fig. 4. ReLU Function

It's obvious that 2 ReLUs serve the same function as hard sigmoid function, so the function(1) can be expressed using ReLU as:

$$y = \sum_{i=1}^{12} ReLU(w_i x + b_i) + b$$

Note that using ReLU, there are 12 functions in total, which is the twice of that using hard sigmoid function.

When there are more than one feather we care about, the operation is the same as before:

$$y = \sum_{i=1}^{n} ReLU(\sum_{j=1}^{k} w_{ij} x_j + b_i) + b$$

Where k is the number of features, n is the number of ReLU(or sigmoid) functions you choose to model the original function. Define:

$$r_i = \sum_{j=1}^{k} w_{ij} x_j + b_i$$

then(assume k=6):

$$\mathbf{r} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ r_6 \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{41} & w_{42} & w_{43} \\ w_{51} & w_{52} & w_{53} \\ w_{61} & w_{62} & w_{63} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \end{bmatrix} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

Understanding the parameters:

- w_{ij}: the weight of jth feature to the ith r
- b_i: the bias of the ith r

To visualize this, the structure can be shown as Fig.5(5):

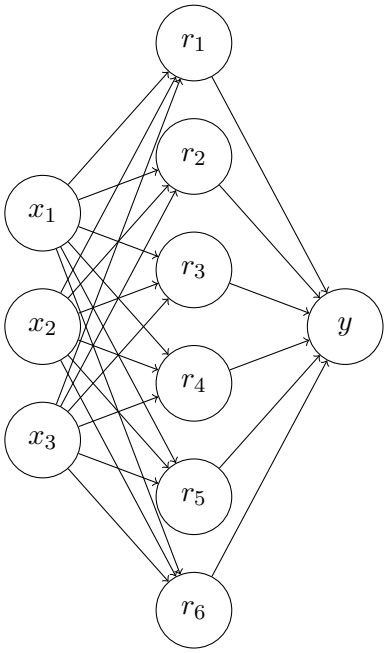

Fig. 5. DNN with one hidden layer

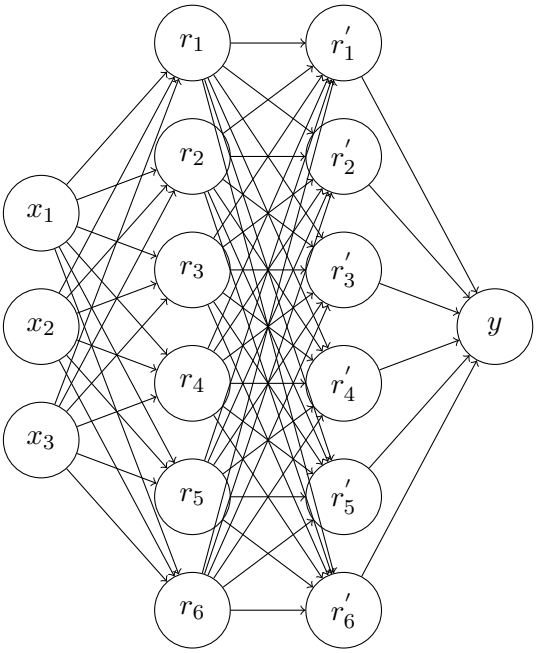

Fig. 6. DNN with multiple hidden layers

In fig.5, the output layer y is only the sum of the ReLUs from the hidden layer plus bias. As mentioned before, if enough nodes in Hidden layer is used, y can be a good fit for any continuous function.

However, to model any continuous function is not enough. For Deep Learning, we do the same operation to the hidden layer,creating mutiple hidden layers.A simple illustration of a DNN with 2 hidden layers is shown(6). With the increase of hidden layers, the network gets deeper. And this is why it is called **Deep Learning**.

Since the network with only one layer can model any continuous functions well, why do we need multiple layers? Why don't we just increase the number of nodes in a hidden layer? This remains unknown to me for now.

2 Loss Function

To measure how good out model is doing, we define **Loss Function**. Most widely used loss functions are:

$$L(\theta) = \sum_{i=1}^{n} |\hat{y}_i - y_i|$$

$$L(\theta) = \sum_{i=1}^{n} (\hat{y}_i - y_i)^2$$

$$H(P, Q) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$

Note the third expression is also called **Cross-entropy**, detailed explanation will be included in later lecture notes.

During training, our goal is to minimize the loss function, one commonly used approach is **Gradient Descent**:

$$\boldsymbol{\theta_j} = \boldsymbol{\theta_i} - \alpha \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta})$$

Where α stands for **learning rate**, which is a hyperparameter in deep learning.