# A glance on Deep Learning

## 1. What is AI (Deep Learning)

Actually, the "AI" we refer to is actually always machine learning. And the state-of-art machine learning is deep learning.

Function can be used to describe reality. We assume there is an ideal function that can represent reality accurately (Let's name it real function).

Computer can construct a complex function (Deep Learning uses Neural Network) that is determined by many parameters.

Computer can make this function close to real function by adjusting its parameters automatically.

## 2. Basic ideas of deep learning

### Function can be used to describe reality. We assume there is an ideal function that can represent reality accurately (Let's name it real function).

What is function?

Function is using numbers to express real things.

First, we can denote features with numbers:

Your Weight (real feature): 60kg (number)

Your Height (real feature): 180m (number)

You have a picture (real feature):

It is connected to another feature: It is a dog (1 (Yes)/0 (No))

We can also use these numbers to denote new features, in numbers as well.

Your BMI (number that represents real feature): $Weight/Height^2$(number)

Functions take an input (real feature/number), then give an output (number). These numbers can have real meanings. Any connections between real things can be described with functions.

Numbers used to describe real features is called data. Handling data is equivalent to handling reality.

Turning reality into data is necessary for both science and engineering. Data science studies on how to deduct real features from data. Computing science studies on how to handle data.

As long as you can represent some aspect of reality with numbers (You have data), AI can handle them. That's why AI is a part of Data Science and a part of Computing science.

## Computer can construct a complex function (Neural Network) that is determined by many parameters.

There are many forms of function that can be built with computer. In deep learning, we use a structure called Neural Network.

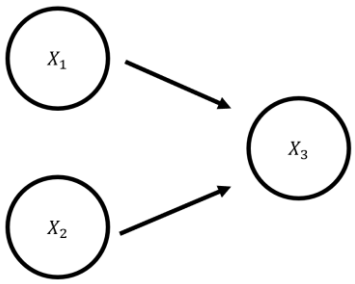
### Linear function

The Neural Network uses formula:

$$Z = W \cdot X + b$$

Where Z is the function value, X is input value, W and b are parameters. It is a typical linear function. We choose it because we can handle linear equations pretty well, with well-developed linear algebra.

Now we may introduce a graphic presentation. Like following one:

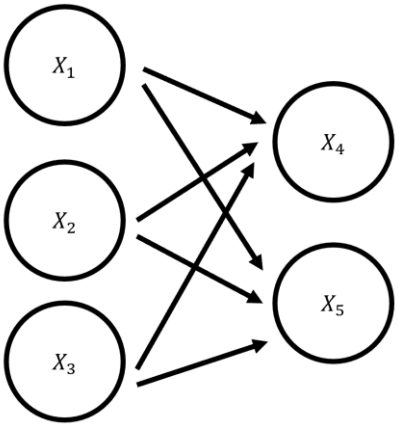



In this picture, there are three numbers (Or more generally, variables. Variables can be any numbers): $X_1, X_2, X_3$

It can be seen that the variables $X_1, X_2$ point to the variable $X_3$. When we have such points, we mean:

$$X_3 = W_1 \cdot X_1 + W_2 \cdot X_2 + b$$

Then what about this picture?




Yes, they are:

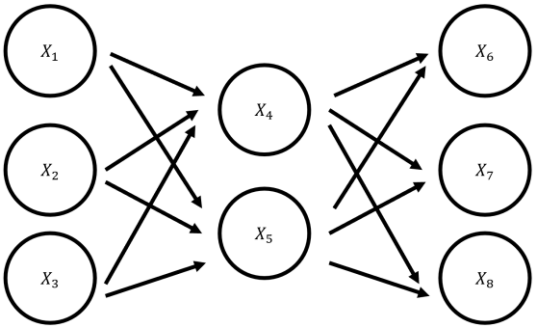
$$X_4 = W_{41} \cdot X_1 + W_{42} \cdot X_2 + b_4$$

$$X_5 = W_{51} \cdot X_1 + W_{52} \cdot X_2 + b_5$$

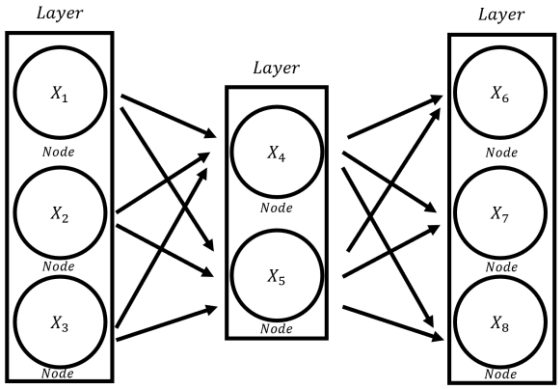
You may see that the lower index has some changes. This is just a custom to be align with linear algebra.

### Make a "Network" of linear functions

We can go "deeper".




Such nets are almost neural networks. You may find out that networks are only a way to build up a series of linear relation.



We call each variable surrounded by a circle a node. Each mapping is counted as a layer, as the column the nodes are in.

Notice that each mapping is also correspond to a weight, aside from indicating there connection between nodes.

### Combine non-linear function with the network: Neural Network

However, there's a drawback of our current network. That is, even though it has many layers, it is only the same function in essence. That is:

$$X_6 = W_{64}(W_{41} \cdot X_1 + W_{42} \cdot X_2 + b_4) + W_{65}(W_{51} \cdot X_1 + W_{52} \cdot X_2 + b_5) + b_6$$

Has no difference with:

$$X_6 = W_{61} \cdot X_1 + W_{62} \cdot X_2 + W_{63} \cdot X_3 + b$$

Except that they their parameters have different form.

That's the problem for linear system, preventing them from forming complex functions.

But the solution is easy. We add non-linear functions in each mapping between layers. We denote this non-linear function as $\sigma$. There are many choices for $\sigma$. For example:

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

This is a commonly used function called sigmoid function. You can also call it logistic function, which is exactly the one describing logistic curve in biology.

For more details, please check out $ReLU$ function and $th^{-1}$ function yourself

So each mapping will be like this:

$$X_6 = \sigma(W_{61} \cdot X_1 + W_{62} \cdot X_2 + W_{63} \cdot X_3 + b)$$

Then our network becomes a Neural Network. For this case, it is called "Fully Connected Neural Network" (FCNN). You may also call it Multi-Layer Perception (MLP).

The basic idea of Neural Network is using linear mapping, as well as non-linear functions, to build a complex function determined by parameters.

*If you are familiar with linear algebra, you can simply write the equation as:

$$X_i = \sigma(W_{ij} \cdot X_j + b_i)$$

### Other kinds of Neural Network

The one already introduced is fully connected one. There are partially connected ones for sure, like Graph Neural Network (GNN). There are other network structures, depending on what kind of linear mapping they are using and how is are they connected. Typical ones include Recurrent Neural Network (RNN), Convolution Neural Network (CNN), etc.

## Computer can make this function close to real function by adjusting its parameters automatically.

The key to artificial intelligence lies in how the computer adjusts a complex function to closely approximate the real underlying function. Once the computer can effectively handle and interpret data using this function, it can be considered to possess a form of intelligence.

To implement this, the deep learning adapts two strategies: loss function and gradient descent.

### Loss function

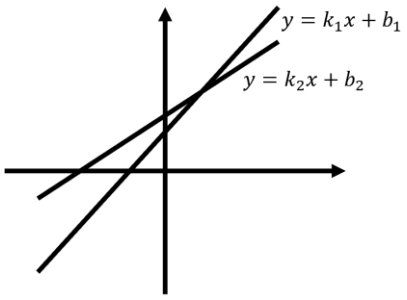
Deep learning approximates the real function by reducing the difference between constructed function and real function. To measure the difference, loss function is constructed.

However, it is usually hard to directly measure how different two functions are. Taking two linear functions for example:

$$y = k_1 x + b_1$$

$$y = k_2 x + b_2$$

Which are shown as following illustration:




At some points, the function values of two functions are similar to each other. However, the function values are different from each other from other points. Also, the real

function is always unknown and complex, forbidding us from comparing parameters to measure the different between two functions.

The strategy adapted is sampling. We take single inputs and compare the functions at those points. If they are different, then the functions are more likely to be different from each other. If they are similar, then the functions are more similar to each other.

The value of real function at these points is data (Actually, data is regarded as the pair consisting of the input value and the real function value). With the same input, the constructed function gives a value. By comparing them, we can judge whether the constructed function is close to real function, thus loss function can be constructed.

A typical loss function is Mean Squared Error function (MSE) Loss:

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

Which can be taken as the average of Euclidean distance. $y_i$ is the data, and the $\hat{y}_i$ is the function value of the constructed function.

Another one is Binary Cross-Entropy Loss function (Log Loss):

$$L = -\frac{1}{N}\sum_{N} y_i \log \hat{y}_i + (1 - y_i)\,(\log(1 - \hat{y}_i))$$

Loss function has its limits. When the loss function is small, the constructed function is not necessarily close to the real function. On contrast, when the loss function is too small at some points, the constructed function may be highly different from the real function. This is called overfitting. Vice versa. This is due to that the loss function constructed is taking samples to estimate the difference.

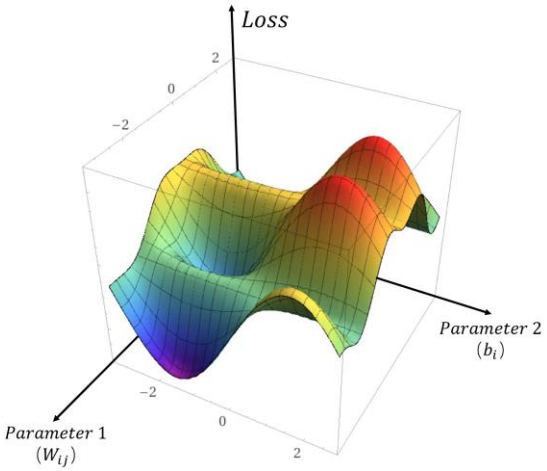
However, loss function is still a good estimation when there is certain amount of data. Lower the loss function is, better the constructed function may be.

## Gradient descent

"Nearly all of deep learning is powered by one very important algorithm: Stochastic Gradient Descent (SGD)" – Goodfellow et al.

With loss function to minimize (the opposite of the loss function to be maximized), we want to adjust the parameter so that the loss function can be as little as possible.

Let's consider how different parameters would lead to different loss function values. We use graphs to illustrate this. Assume at $x = x_0$, where $x_0$ is a random number, we introduce axis of parameters and plot out relevant loss function value (Loss for abbreviation) of different points. It turns to be a graph like:




In this illustration, we have two different parameters, you may take them as $W_{ij}$ and $b_i$ in the linear function mentioned before. Loss is different with different choice of these parameters. We aim to pick out a set of parameters so that the loss will be minimized (Loss should be optimized overall, but it doesn't hurt deducing the loss at a single value).

We may adapt to following strategy: Each time we make some adjustment to the parameters, so that it will "go down" along the curve (The loss reduces).

This can be implemented according to gradient concept in math, which tells how to change the parameters may receive a rapid change to target (Loss) function:

$$W_i = W_{i-1} - \alpha \frac{\partial L}{\partial W}$$

$$b_i = b_{i-1} - \alpha \frac{\partial L}{\partial b}$$

Such parameter updating strategy is called gradient descent, since it uses gradient concept. By iteratively conducting gradient descent with different input values, the constructed function automatically approximates the real function.

Notice that we are not iterate on the same graph. In practice, we take different input values, which correspond to a Parameters-Loss graph respectively. On each graph we make the parameters to make a small shift according to gradient.

Precisely, this should be called stochastic gradient descent.

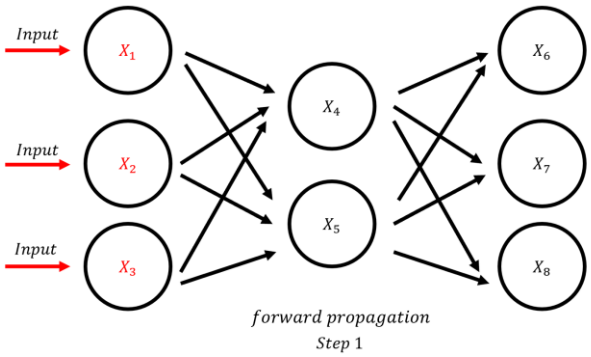
Although it doesn't make sure the algorithm is sure to find a set of suitable parameters, it works well in practice when amount of data is sufficient and there are many iterations.

## Backpropagation

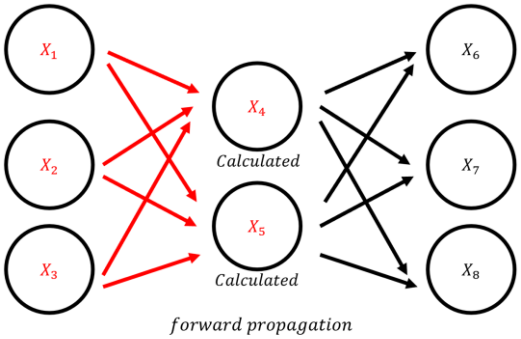
The gradient descent process of Neural Network is called backpropagation.

This is because the Neural Networks are sets of linear mappings. When calculating gradient, we need to use chain rule and take the derivatives from back to front, and update the upcoming parameters.

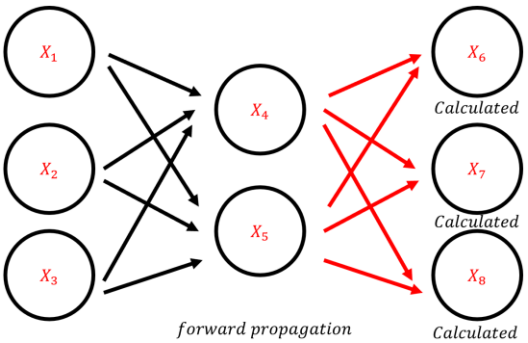
Actually, these derivatives always involve the function value of the constructed function at each point. We take the input, and calculate the function values gradually from front to back, to use these values to implement gradient descent. This process is called forward propagation (Sometimes it is called forward process).



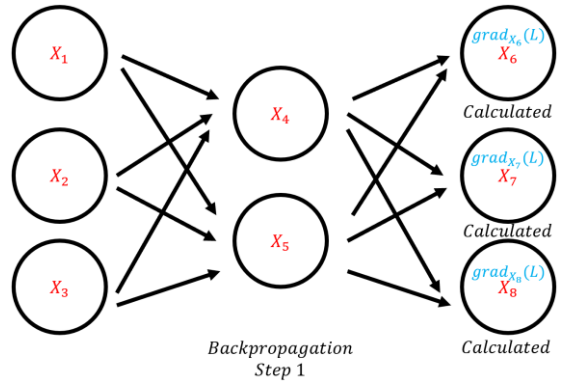

*forward propagation*
*Step 1*



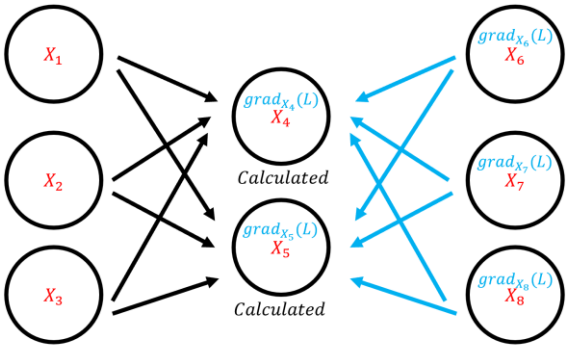

*forward propagation*
*Step 2*




*forward propagation*
*Step 3*

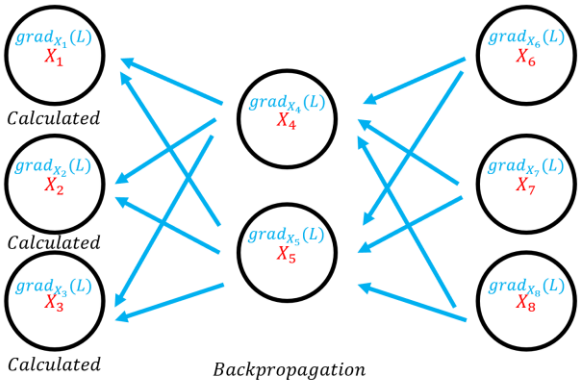



*Backpropagation Step 1*




*Backpropagation Step 2*

Blue arrow means the gradient of backside node is used to calculate the gradient of frontside node




*Backpropagation Step 3*

Blue arrow means the gradient of backside node is used to calculate the gradient of frontside node

In the essence, both the forward and back propagation serves to implement the gradient descent.

The architecture of a Neural Network is well-suited for the application of gradient descent. With these dedicate underlying structure, deep learning is able to approximate reality in a effective way and handle real problems.