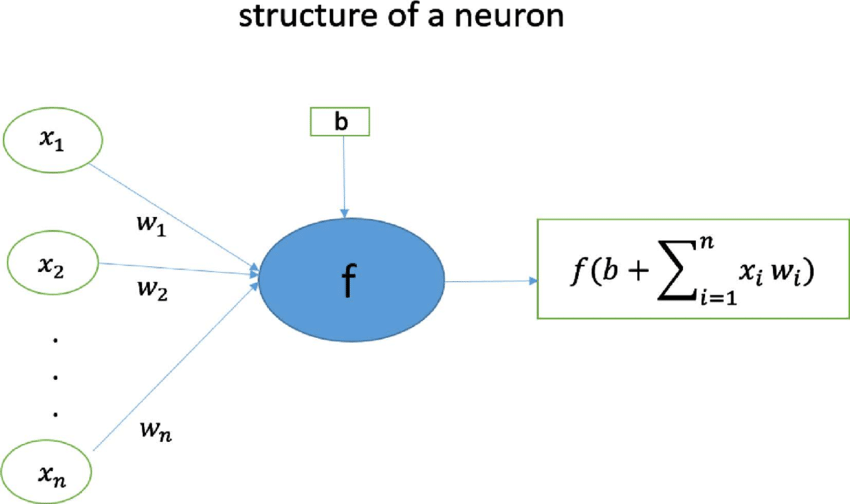
Activation Function

In order to improve the performance of neural network we generally consider a better activation function, loss function, optimizer and also hyperparameter tuning.

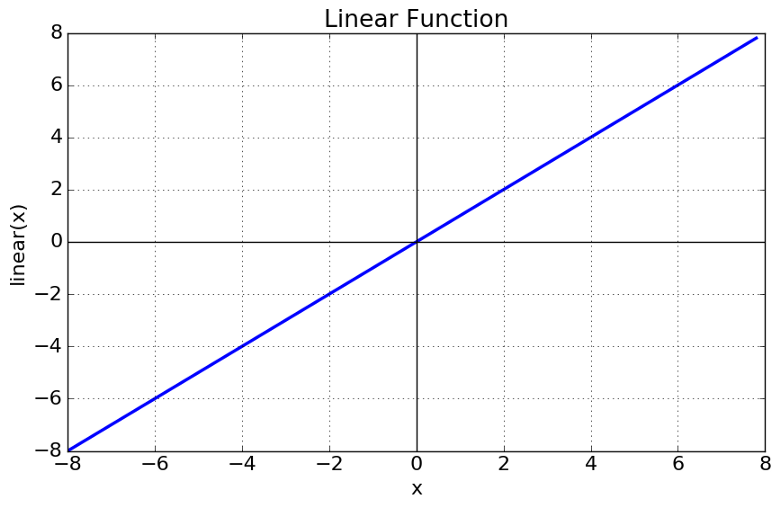


In simple words, a neural network generally calculates the WEIGHTED SUM of the input and adds bias to it. According to the problem statement we consider the activation function.

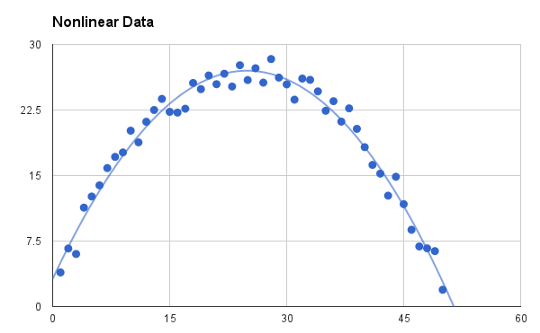
**Why do we use them?**

Moreover, it is used to determine the output of the neural network like 1 or 0.

Linear Activation Function

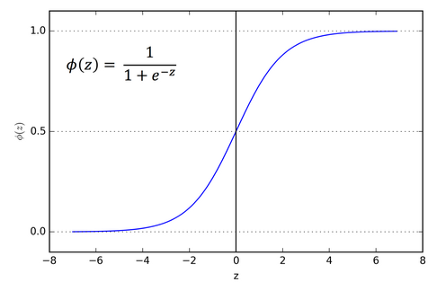


Non-linear Activation Functions



1. **Sigmoid or Logistic Activation Function**

It looks like

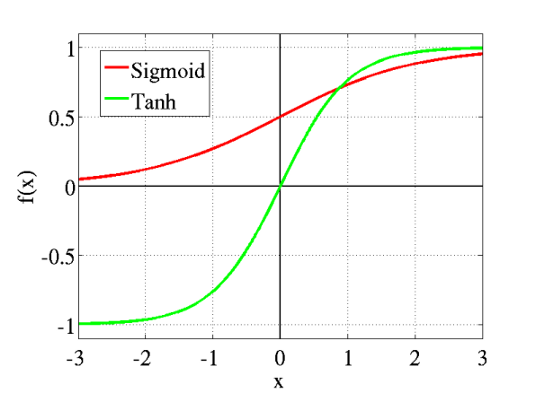


The importance of using sigmoid function is whatever the value ranges from it will automatically transforms the value between 0 and 1. So it is used to predict the probability of the output and it ranges between 0 and 1.It is defensible , so we can find the slope of curve at any two points. And the derivative of sigmoid activation is 0.25.

And sigmoid activation function is used for the **binary class classification** and SoftMax activation function is used for **multiclass classification.**

## ****2. Tanh or hyperbolic tangent Activation Function****

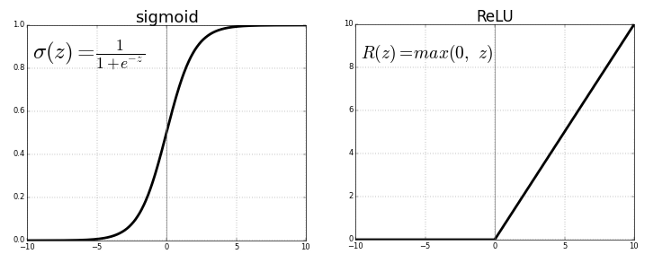
It is also similar to sigmoid function but the difference is range lies between -1 to 1.



Here in tanh the negative input is mapped to the negative and the 0 input is nearly mapped to the 0. And it is used classification between two classes, uses in feed forward networks.

## ****3. ReLU (Rectified Linear Unit) Activation Function****

It is mostly used activation in CNN and deep learning, and it looks like



As you can see, the ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.

**Range:**[ 0 to infinity)

The function and its derivative **both are** **monotonic**.

But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

**Pros:**

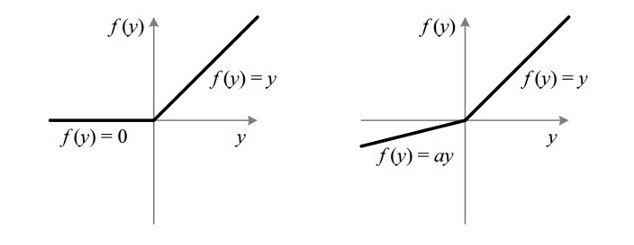
* Less time and space complexity, because of sparsity, and compared to the sigmoid, it does not evolve the exponential operation, which are more costly.
* Avoids the vanishing gradient problem.

**Cons:**

* Introduces the *dead relu* problem, where components of the network are most likely never updated to a new value. This can sometimes be also be a pro.
* ReLUs does not avoid the exploding gradient problem.

**4. Leaky ReLU**

It is an attempt to solve the dying ReLU problem





**Fig : ReLU v/s Leaky ReLU**

The leak helps to increase the range of the ReLU function. Usually, the value of **a**is 0.01 or so.

When **a is not 0.01** then it is called **Randomized ReLU**.

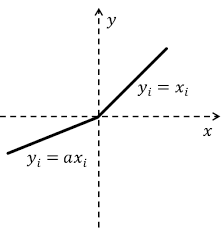
Therefore, the **range** of the Leaky ReLU is (-infinity to infinity).

Both Leaky and Randomized ReLU functions are monotonic in nature. Also, their derivatives also monotonic in nature.

**Why derivative/differentiation is used?**

When updating the curve, to know in **which direction** and **how much** to change or update the curve depending upon the slope. That is why we use differentiation in almost every part of Machine Learning and Deep Learning.

**5.Parameteric rectified linear unit (PReLU)**

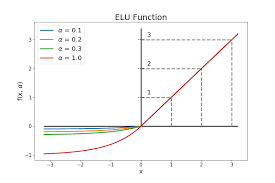




And it is advanced to leaky relu and here in this function the alpha might be any value so the range is -infinity to infinity.

6.Exponential linear unit (ELU)





**ELU**. Exponential Linear Unit or its widely known name **ELU** is a **function** that tend to converge cost to zero faster and produce more accurate results. Different to other **activation functions**, **ELU** has an extra alpha constant which should be positive number. They are both in identity **function** form for non-negative inputs.

**Pros**

* Avoids the *dead relu* problem.
* Produces negative outputs, which helps the network nudge weights and biases in the right directions.
* Produce activations instead of letting them be zero, when calculating the gradient.

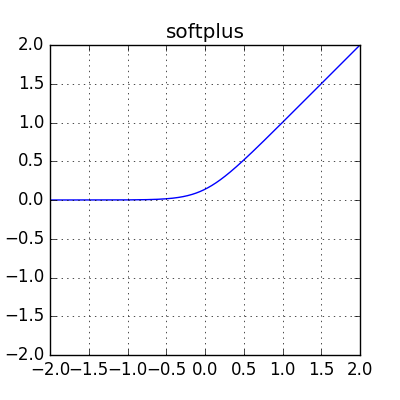
**Cons**

* Introduces longer computation time, because of the exponential operation included
* Does not avoid the exploding gradient problem
* The neural network does not learn the alpha value

7.SoftPlus activation fuction

Activation unit calculates the net output of a neural cell in [neural networks](https://sefiks.com/2017/01/15/introduction-to-neural-networks-a-mechanism-taking-lessons-from-the-past/). [Backpropagation algorithm](https://sefiks.com/2017/01/21/the-math-behind-backpropagation/) multiplies the derivative of the activation function. That’s why, picked up activation function has to be differentiable. For example, step function is useless in backpropagation because it [cannot be backpropageted](https://sefiks.com/2017/05/15/step-function-as-a-neural-network-activation-function/). That is not a must, but scientists tend to consume activation functions which have meaningful derivatives. That’s why, [sigmoid](https://sefiks.com/2017/01/21/sigmoid-function-as-an-activation-function/) and [hyperbolic tangent](https://sefiks.com/2017/01/29/hyperbolic-tangent-as-neural-network-activation-function/) functions are the most common activation functions in literature. Herein, softplus is a newer function than sigmoid and tanh. It is [firstly introduced](https://papers.nips.cc/paper/1920-incorporating-second-order-functional-knowledge-for-better-option-pricing.pdf) in 2001. Softplus is an alternative of traditional functions because it is differentiable and its derivative is easy to demonstrate. **Besides, it has a surprising derivative!**

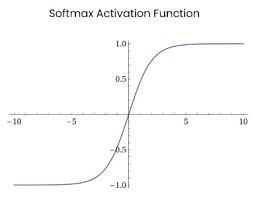
Softplus function:   f(x) = ln(1+ex)



The gradients will be smooth and it is designed by using relu.

**8.Softmx activation function**

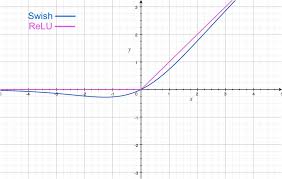
The sigmoid function can be applied easily, the ReLUs will not vanish the effect during your training process. However, when you want to deal with classification problems, they cannot help much. Simply speaking, the sigmoid function can only handle two classes, which is not what we expect.

****

So it is used for multi class classification.

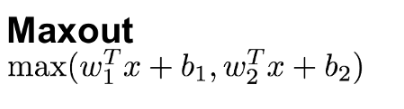
**9.Swis activation function**

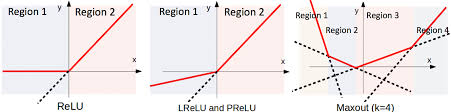
It is self-gated function and it is given by y=x\*sigmoid(x) and moreover the vanishing gradient problem is not seen here due to the equation the data is getting multiplied with sigmoid function. It is deigned based upon sigmoid and it is used in LSTM networks.



**10.Maxout activation function**

If you remember that each hidden layer is composed of a matrix multiplication of the weight matrix WT and the inputs x plus the bias. This WTx + b makes up the pre-activation of the hidden unit. In traditional neural nets, this pre-activation is passed through a sigmoid activation to convert to -1 and 1. The maxout activation instead takes the maximum value of the pre-activations and reshapes it into a vector containing only this value.

****

****

After the learning of the weights it is going to give you the output. And it designed by using relu and leaky relu.

**Loss functions**

In general loss functions are used to check the error value of the predicted output if the loss is less then the model has predicted the data as correct.

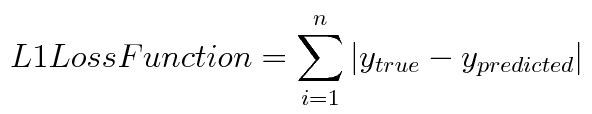
## L1 and L2 loss

L1 and L2 are two common loss functions in machine learning which are mainly used to minimize the error.

**L1 loss function** are also known as **Least Absolute Deviations** in short **LAD**. **L2 loss function** are also known as **Least square errors** in short **LS**.

Let's get brief of these two

### **1.L1 Loss function**

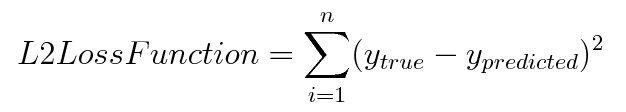
It is used to minimize the error which is the sum of all the absolute differences in between the true value and the predicted value.

It is used in most of the case and it is considered as L1 loss function

### **2.L2 Loss Function**

It is also used to minimize the error which is the sum of all the squared differences in between the true value and the predicted value.

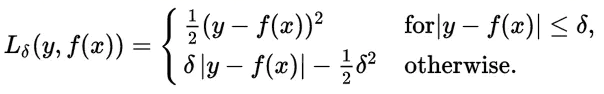
And in l2 loss function the error rate might be increased due to the we are squaring the error i.e. if are having any kind of outliers in the dataset then it might be increased in error .



**The disadvantage** of the **L2 norm** is that when there are outliers, these points will account for the main component of the loss. For example, the true value is 1, the prediction is 10 times, the prediction value is 1000 once, and the prediction value of the other times is about 1, obviously the loss value is mainly dominated by 1000.

## 3. Huber Loss

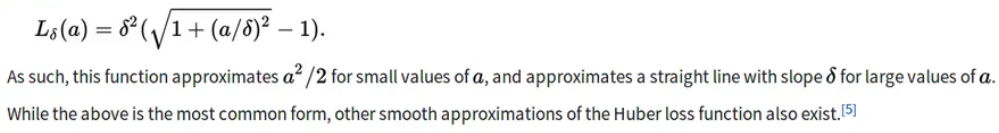
Huber Loss is often used in regression problems. Compared with L2 loss, Huber Loss is less sensitive to outliers (because if the residual is too large, it is a piecewise function, loss is a linear function of the residual).



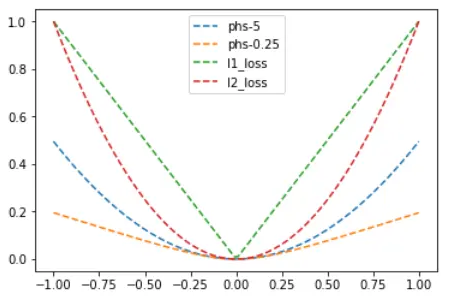
And so to avoid the problem in the L2 loss we are using the huber loss and it will be handle the outliers.

**4.** **Pseudo-Huber loss function**

A smooth approximation of Huber loss to ensure that each order is differentiable**.**



Where 𝛿 is the set parameter, the larger the value, the steeper the linear part on both sides.

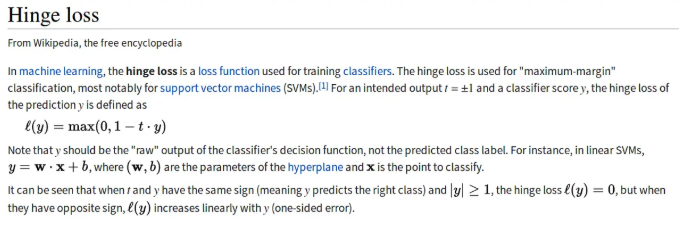


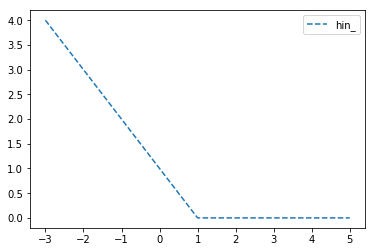
## 5.Hinge Loss

Hinge loss is often used for binary classification problems, such as ground true: t = 1 or -1, predicted value y = wx + b

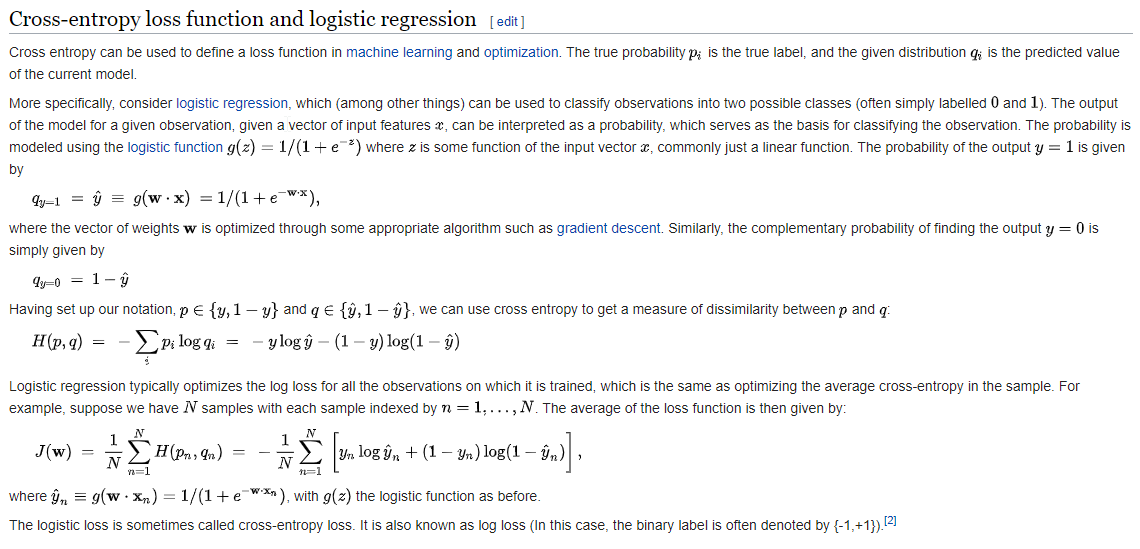
In the svm classifier, the definition of hinge loss is.

In most of the time Hinge loss is used to calculate the loss in support vector machine.

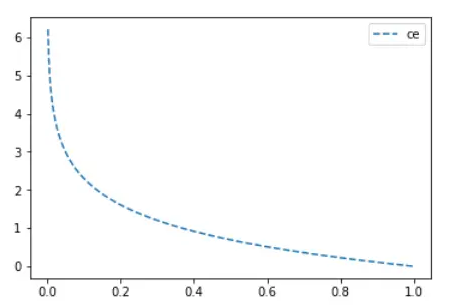




6.Cross Entropy loss

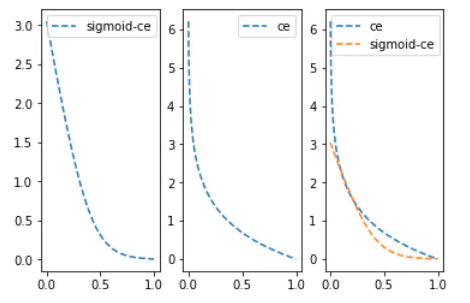


The above is mainly to say that cross-entropy loss is mainly applied to binary classification problems. The predicted value is a probability value and the loss is defined according to the cross entropy. Note the value range of the above value: the predicted value of y should be a probability and the value range is [0,1]



**7.Sigmoid-Cross-entropy loss**

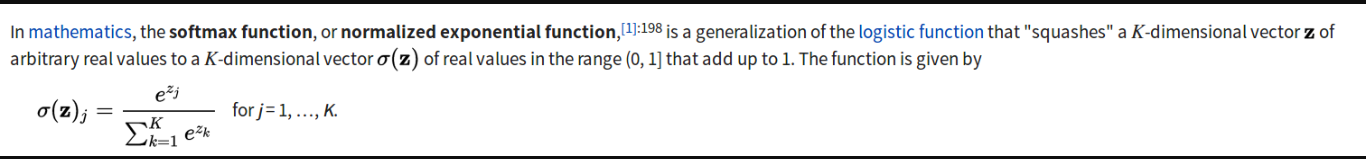
The above cross-entropy loss requires that the predicted value is a probability. Generally, we calculate $scores = x\*w + b$. Entering this value into the sigmoid function can compress the value range to (0,1).



It can be seen that the sigmoid function smoothes the predicted value(such as directly inputting 0.1 and 0.01 and inputting 0.1, 0.01 sigmoid and then entering, the latter will obviously have a much smaller change value), which makes the predicted value of sigmoid-ce far from the label loss growth is not so steep.

**8.Softmax cross-entropy loss**

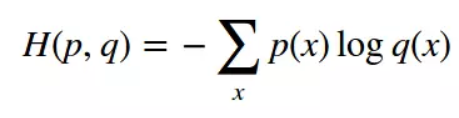
First, the softmax function can convert a set of fraction vectors into corresponding probability vectors. Here is the definition of softmax function



As above, softmax also implements a vector of 'squashes' k-dimensional real value to the [0,1] range of k-dimensional, while ensuring that the cumulative sum is 1.

According to the definition of cross entropy, probability is required as input.Sigmoid-cross-entropy-loss uses sigmoid to convert the score vector into a probability vector, and softmax-cross-entropy-loss uses a softmax function to convert the score vector into a probability vector.

According to the definition of cross entropy loss.



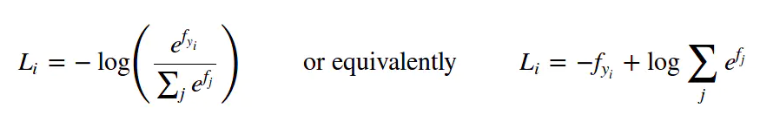
where $p(x)$ represents the probability that classification $x$ is a correct classification, and the value of $p$ can only be 0 or 1. This is the prior value

$q(x)$ is the prediction probability that the $x$ category is a correct classification, and the value range is (0,1)

So specific to a classification problem with a total of C types, then $p(x\_j)$, $(0 <\_{=} j <\_{=} C)$ must be only 1 and C-1 is 0(because there can be only one correct classification, correct the probability of classification as correct classification is 1, and the probability of the remaining classification as correct classification is 0)

Then the definition of softmax-cross-entropy-loss can be derived naturally.

Here is the definition of softmax-cross-entropy-loss.



Where $f\_j$ is the score of all possible categories, and $f\_{y\_i}$ is the score of ground true class