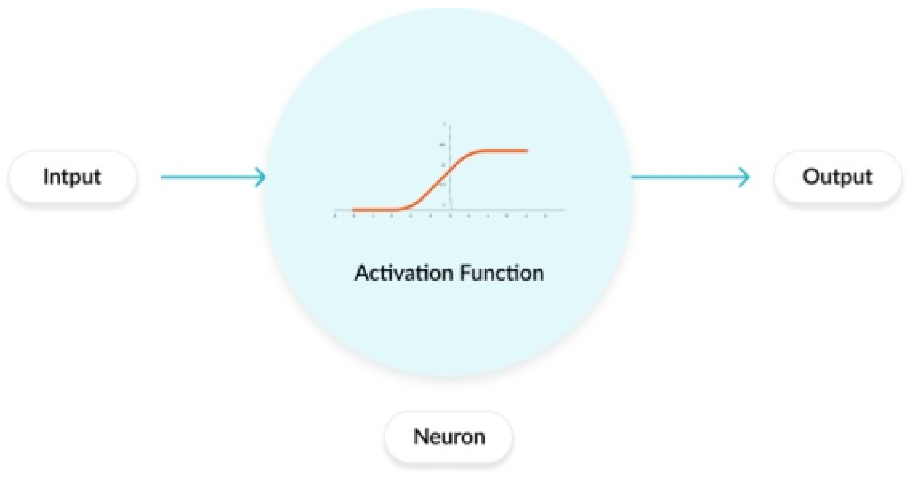
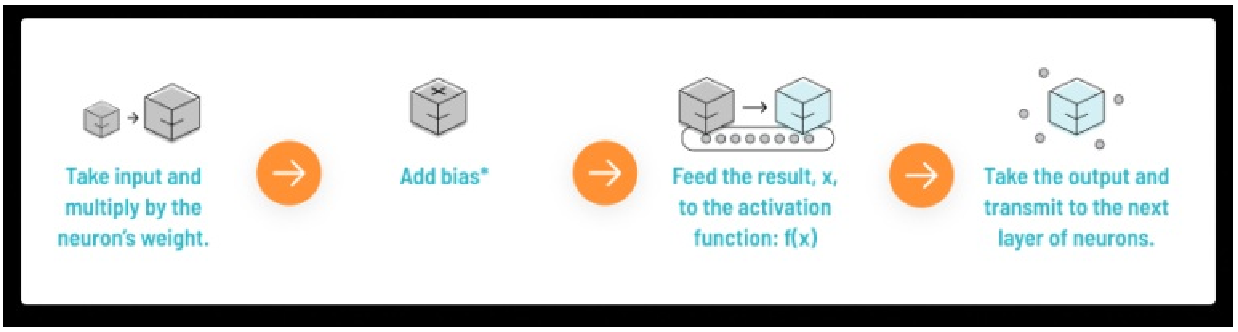
Activation Functions

Activation functions are the mathematical equations which helps us to determine the output of a neural network. Activation functions are attached to each and every neuron in the network and determines whether it should be activated or not based on wther each neuron's input is relevant for prediction of the model. Activation functions also helps us to normalize the output of each neuron to a range between 0 and 1 or between -1 and 1.

In a neural network, numeric data points (called inputs) are fed into the neurons in the input layer. Each neuron has a weight and multiplying the input number with the weight gives the output of the neuron, which is transferred to the next layer. Activation function is a mathematical gate in between the input feeding the current neuron and its output going to the next layer. It can be as simple as a step function that turns the neuron output ON and OFF depending on a rule or threshold. Or it can be a transformation that maps the input signals into output signals that are needed for the neural network to function.



Increasingly, neural networks use non-linear activation functions, which can help the network to learn complex data, compute and learn almost any function representing a question, and provide accurate predictions.


Take input and multiply by the neuron's weight.
Add bias*
Feed the result, x, to the activation function: f(x)
Take the output and transmit to the next layer of neurons.

There are so many activation functions among them few are as listed below.

1. Sigmoid Function

2. TanH / Hyperbolic Tangent Function

3. ReLU Function

4. Leaky ReLU Function

5. Exponential Linear Units (ELU) Function

6. Softmax Function

7. Parametric ReLU (PReLU)

8. Swish (A self-gated) Function

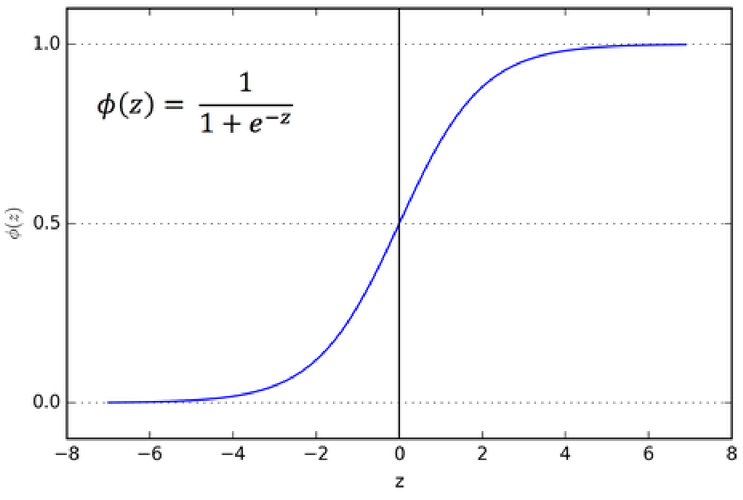
9. Maxout Function

10. Softplus Function

1. Sigmoid Function

A Sigmoid function is a bounded, differentiable, real function that is defined for all real input values and has a non-negative derivative at each point. Sigmoid function is convex for values less than 0 and it is concave for the values more than 0. Because of this, the sigmoid function and its affine compositions can process multiple optima.

Formula and chart for Sigmoid function is as follows



Many natural processes, such as those of complex system learning curves, exhibit a progression from small beginnings that accelerates a climax over timme. When a specific matheatical model is lacking, a sigmoid function is often used,

Advantages of Sigmoid Function:-

1. Clear predictions, i.e., very close to 0 or 1
2. Normalizing the output of each neuron as output values bound between 0 & 1
3. Smooth gradient, prevents jumps in output values.

Disadvantages of Sigmoid Function:-

1. Function output is not zero-centered
2. Prone to gradient vanishing
3. Powre operations are relatively time consuming
4. Computationally expensive

Uses of Sigmoid Function:-

Usually used in output layer of binary classification, where result is either 0 or 1 as value for sigmoid function lies between 0 and 1 only. So, result can be predicted easily to be 1 if value is >0.5 and 0 otherwise

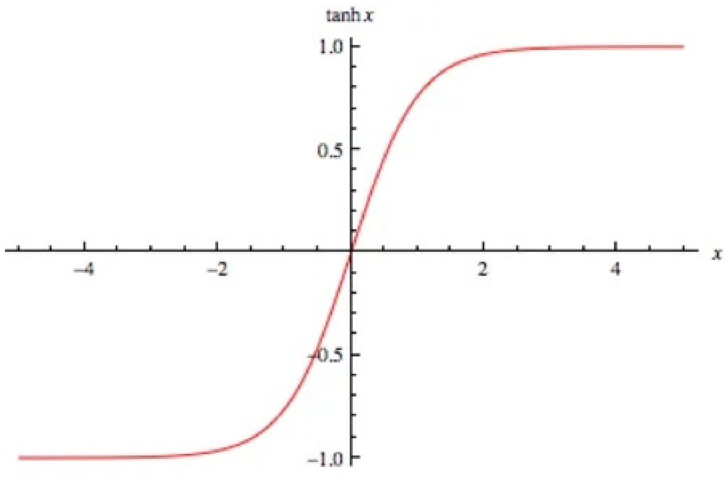
(**Note:** Above said use is not static. Specific activation function used must be analyzed according to the specic problem and / or it depends on debugging)

2. TanH / Hyperbolic Tangent Function

Though the logistic sigmoid has a nice biological interpretation, it turns out that the logisticsigmoid can cause a neural network to get "stuck" during training. This is due in part to the fact that if a strongly-negative input is provided to the logistic sigmoid, it outputs values near zero. Since neural networks use the feed-forward activations to calculate parameter gradients, this can result in model parameters that are updated less regularly than we would like, and are thus "stuck" in their current state. Alternate to logistic sigmoid function is the hyperbolic tangent (TanH), it is also "S"-shaped, but instead outputs values that range (-1,1). Thus strongly negative inputs to the TanH will map to negative outputs. Additionally, only zero-valued inputs are mapped to near-zero outputs. These properties make the network less likely to get "stuck" during training.

Formula and chart for TanH function is as follows:

tanh(x) = (e^x – e^-x) / (e^x + e^-x)



Advantages of TanH Function:-

1. The gradient is stronger for TanH than Sigmoid (derivates are steeper).

Disadvantages of TanH Function:-

1. TanH also has the vanishing gradient problem

Uses of TanH Function:-

Usually used in hidden layers of a neural network as it's values lies between -1 to 1 hence the mean for the hidden layes comes out to be 0 or very close to it. Hence, helps us in centering the data by bringing mean to 0. This makes learning for the next layer much easier

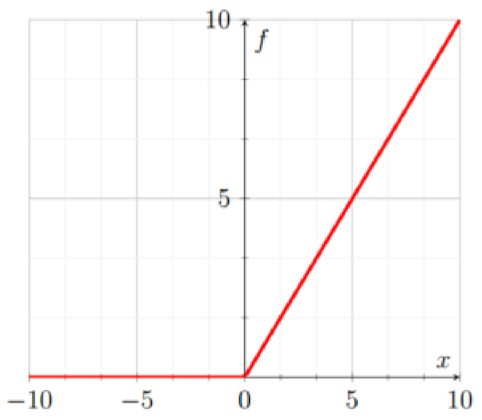
(**Note:** Above said use is not static. Specific activation function used must be analyzed according to the specic problem and / or it depends on debugging)

3. ReLU Function

ReLU stands for Rectified Linear Unit. It is the most widely used activation function. Chiefly implemented in hidden layers of Neural network. The ReLU function actually a function that takes the maximum value. Note that this is not fully interval-derivable, but we can take sub-gradient.

The ReLU function is currently more popular activation function compared with Sigmoid function and TanH function. Formula and chart for ReLU function is as follows

f(x) = max (0,x)



Advantages of ReLU Function:-

1. No gradient saturation problem when input is positive.
2. The calculation speed is much faster. The ReLU function has only linear relationship. Whether it is forward or backward, it is much faster than Sigmoid and TanH (Sigmoid and TanH need to calculate the exponent which will be slower)

Disadvantages of ReLU Function:-

1. ReLU is completely inactive when the input is negative, which means that once a negative number is entered ReLU will die. In this way, in the forward propogation process, it is not a problem. Some areas are sensitive and some are insensitive. But in the backpropagation process, if you enter a negative number, the gradient will be completely zero, which has the same problem as the Sigmoid function and TanH function.
2. ReLU function is not a 0-centric function as its output is either 0 or positive number.

Uses of ReLU Function:-

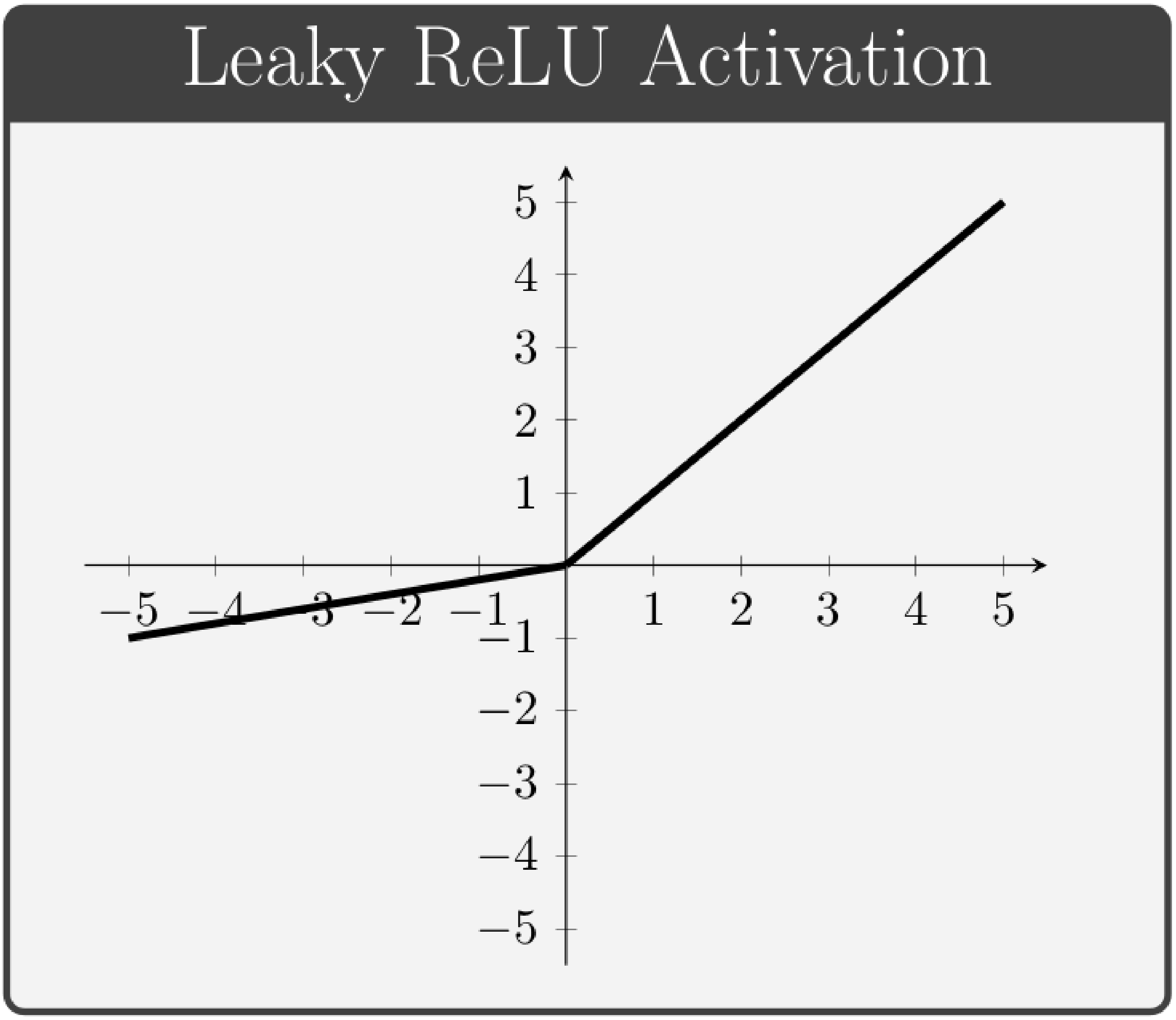
ReLU is less computationally expensive than TanH and Sigmoid because it involves simpler mathematical operations. At a time only a few neurons are activated making the network sparse making it efficient and easy for computation.

4. Leaky ReLU Function

Leaky ReLUs are one attempt to fix the "dying ReLU" problem. Instead of the function being zero when x < 0, a leaky ReLU will instead have a small negative slope (of 0.01 or so). That is, the function computes f(x) = { x if x>0

```
g{αx if x<=0 where α is a small constant.
```

Some people report success with this form of activation function, but the results are not always consistent.



Advantages of Leaky ReLU Function:-

1. Can avoid dead ReLU problem since we allow a small gradient when computing the derivative
2. Faster to compute as no exponential operation is included

Disadvantages of Leaky ReLU Function:-

1. It does not avoid the exploding gradient problem

2. Neural network doesn't learn the alpha value
3. Becomes linear function when it is differentaited.

5. Exponential Linear Units (ELU) Function

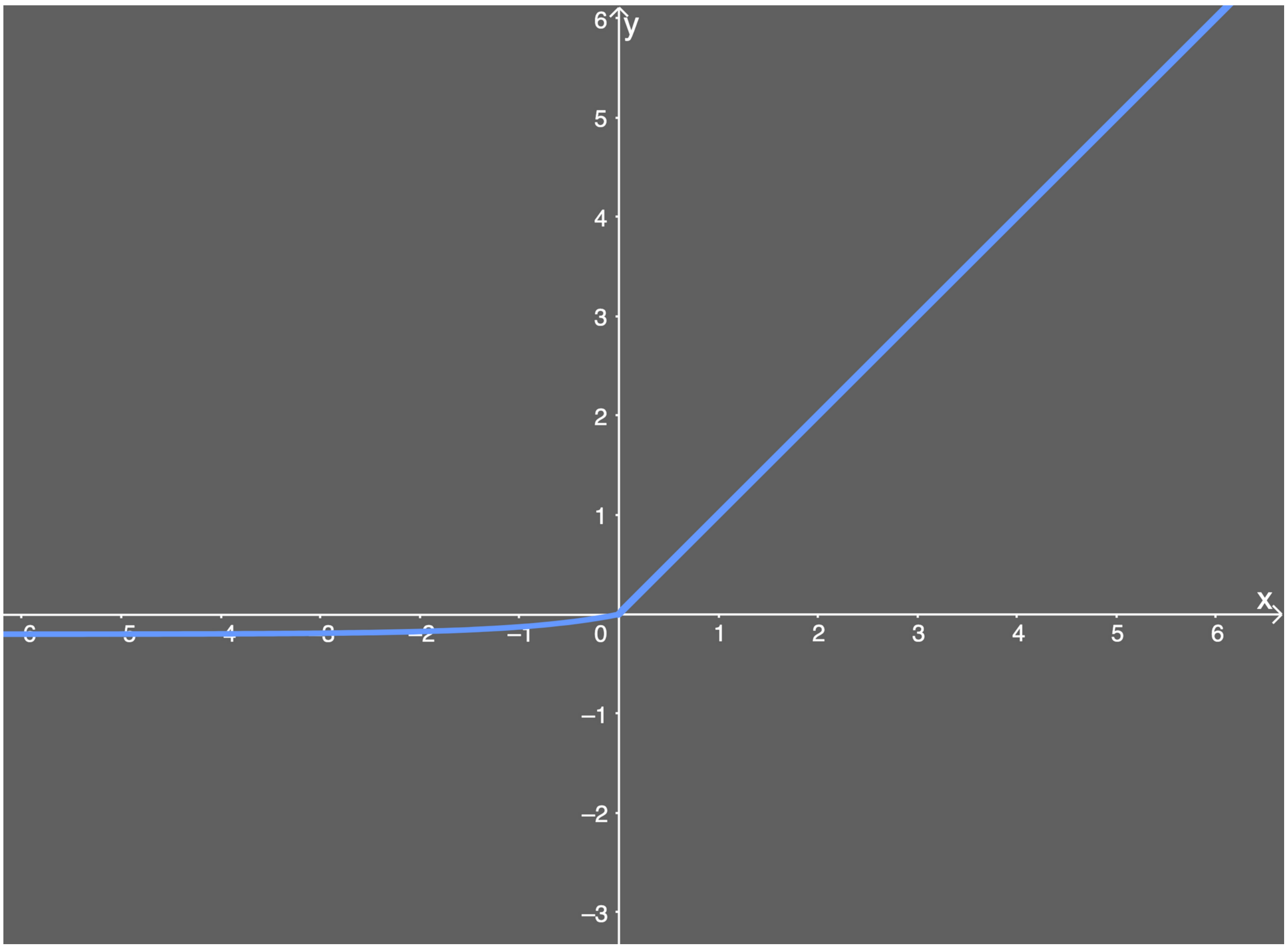
Exponential Linear Unit (ELU) is a function that tend to converge cost to zero faster and produce more accurate results. Different to other activation functions, ELU has a extra alpha constant which should be positive number. For this function an alpha value picked will be between 0.1 and 0.3

ELU(x) = { x if x > 0

```
        {α(e^x-1) if x<0
```

With the above it tends to ReLU if input value x>0 and we get slightly below 0 if input value x<0

Plot for the ELU action function



Advantages of ELU function:-

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

Disadvantages of ELU function:-

1. Introduces longer computation time because of exponential operation included
2. Does not avaoid the exploding gradient problem
3. Neural network doesn't learn the alpha value.
4. Saturates for large negative values

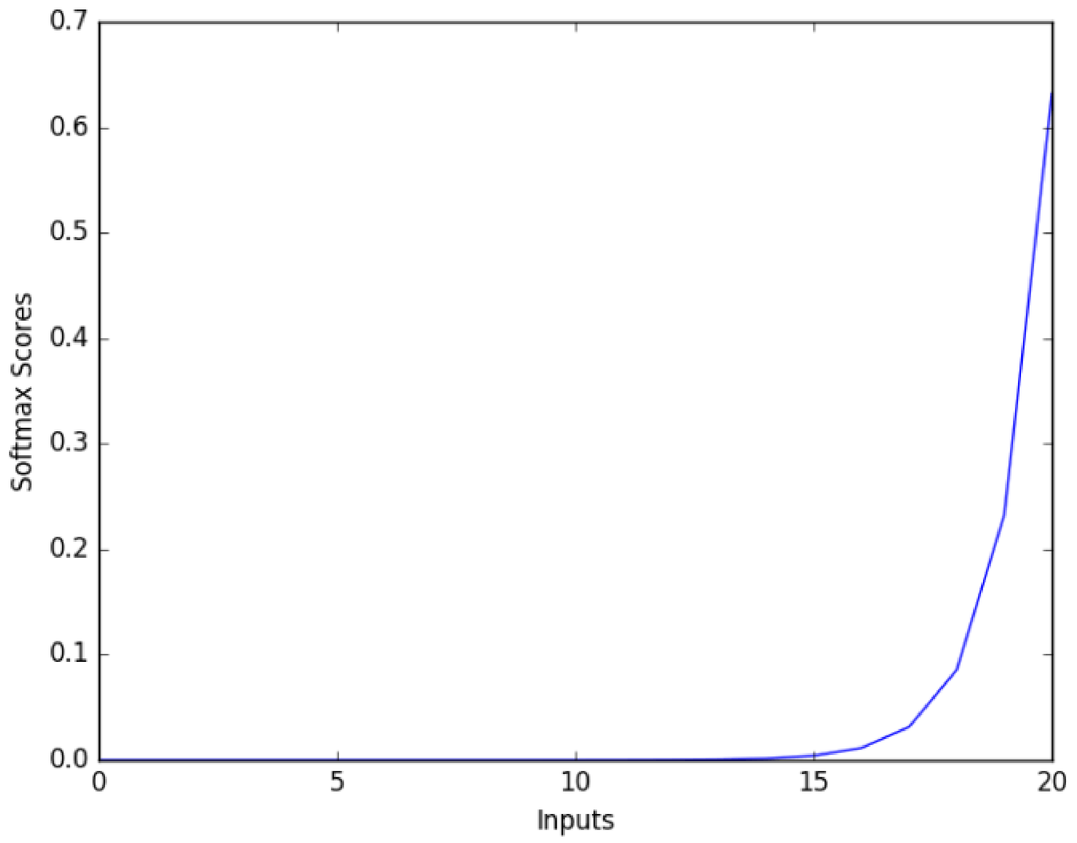
6. Softmax Function

Softmax function calculates the probabilities distribution of the event over 'n' different events. In general way of saying, this function will calculate the probabilities of each target class over all possible target classes. Later the calculated probabilities will be helpful for determining the target class for the given inputs.

Because Softmax function outputs numbers that represent probabilities, each number's value is between 0 and 1 valid value range of probabilities. The range is denoted as [0,1]. The numbers are zero or positive. The entire output vector sums to 1. That is to say when all probabilities are accounted for, that's 100%.

Formula and Chart of Softmax functions is as follows

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for } j = 1, \ldots, K.$$



Advantages of Softmax Function:-

1. The calculated probabilities will be in the range of 0 to 1
2. The sum of all the probabilities is equals to 1
3. Used for multi-classification model it returns the probabilities of each class and the target class will have the high probability.
4. Used in building neural networks in different layer level
5. If we use absolute values (modulus values) we would lose information, while the exponential intrinsically takes care of this.

Disadvantages of Softmax Function:-

1. It doesn't provide null rejection. We need to specify cases to handle null values / lables while training
2. It won't work on data which is linearly separable
3. Should not be used for multi-label classification as it can't produce more than one label with values close to 1
4. Should not be used for regression task as well

Uses of Softmax Function:-

Usually used when trying to handle multiple classes. The softmax function would squeeze the outputs for each class between 0 and 1 and would also divide by the sum of the outputs. Softmax function ideally used in the output layer of the classifier where we are actually trying to attain the probabilities to define the class of each input.

7. PReLU (Parametric ReLU)

PReLU is also an improved version of ReLU. In the negative region, PReLU has a small slope, which can also avoid the problem of ReLU death. Compared to ELU, PReLU is a linear operation in the negative region. Although the slope is small, it does not tend to 0, which is a certain advantage.

Formula for PReLU is

f(y_i) = {y_i, if y_i>0

```
   ={aᵢyᵢ, if yᵢ<=0
```

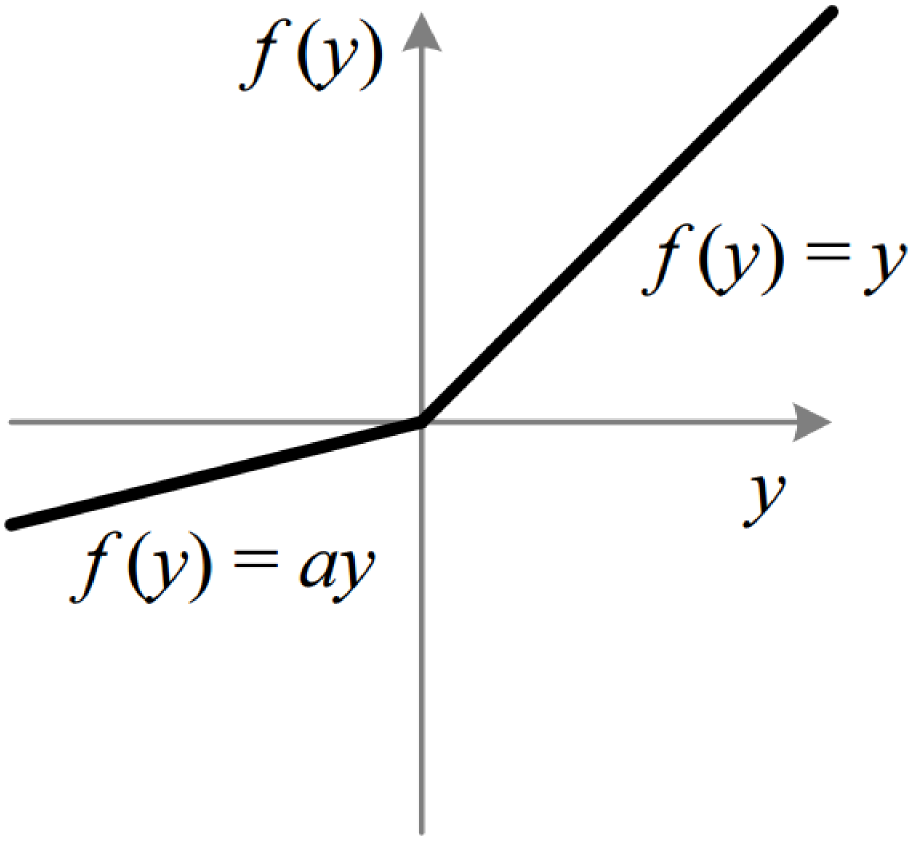
Above formula can also be written as: f(y_i) = max(0,y_i) + a_i min (0,y_i)

. if a_i = 0, f becomes ReLU

. if a_i > 0, f becomes Leaky ReLU

. if a_i is a learnable parameter, f becomes PReLU

Chart of PReLU is as follows



Advantages of PReLU Function:-

1. There will be a penalty for negative values and it will be parametric
2. It becomes generalized ReLU as "a" can be learnt
3. PReLU gives the neurons the ability to choose what slope is best in negative region

Uses of PReLU Function:-

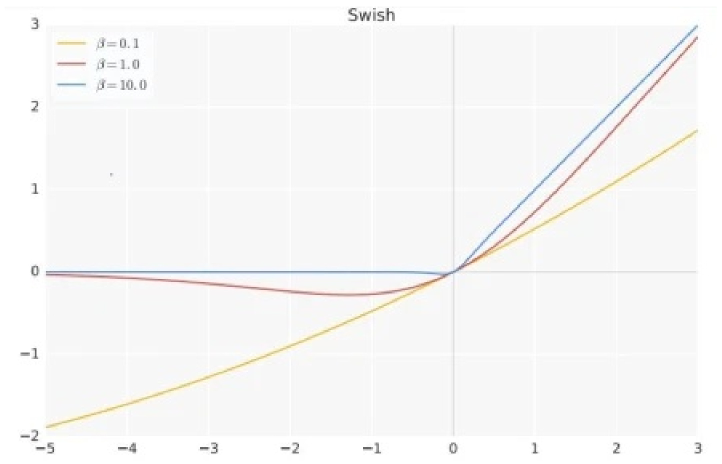
PReLU function is useful when ReLU and leaky ReLU functions fails to solve the dead neurons and the relevant information is not successfully passed to the next layer.

8. Swish (A self-gated) Function

Swish is a lesser known activation function which was discovered by researchers at Google. Swish is as computationally efficient as ReLU and shows better performance than ReLU on deeper models. The values for Swish ranges from negative infinity to infinity. The function is defined as:-

f(x) = x/ (1-e^-x). i.e., f(x) = x * Sigmoid(x)

Chart of Swish Function:-



Swish's design was inspired by the use of Sigmoid functions for gating in LSTMs and highway networks. We use the same value for gating to simplify the gating mechanism, which is called self-gating.

The advantage of self-gating is that it only requires a simple scalar input, while normal gating requires multiple scalar inputs. This feature enables self-gated activation functions such as Swish to easily replace activation that take a single scalar as input (such as ReLU) without changing the hidden capacity or number of parameters.

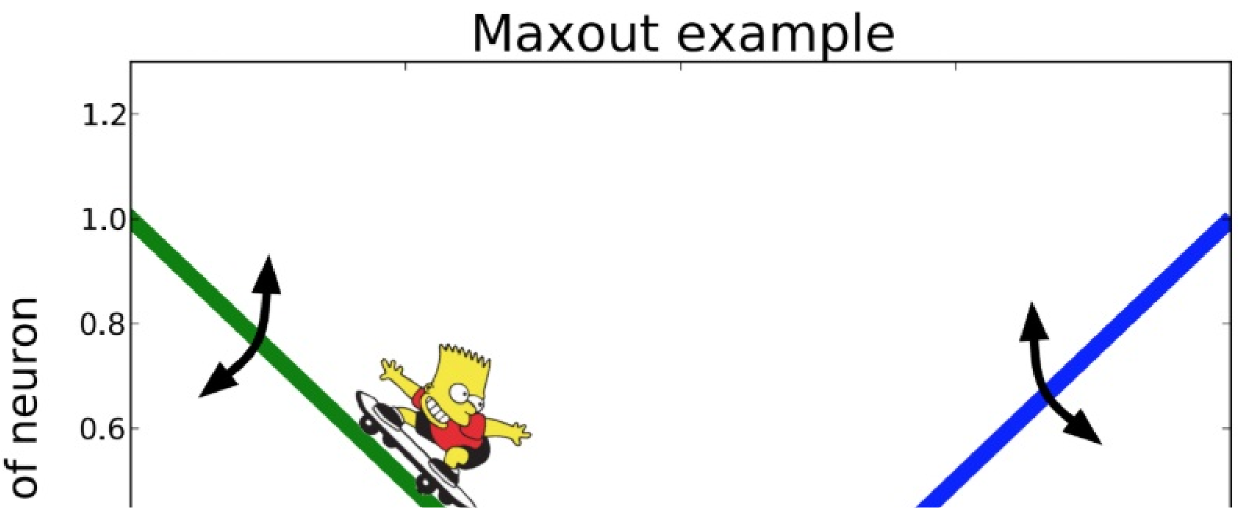
Advantages of Swish Function:-

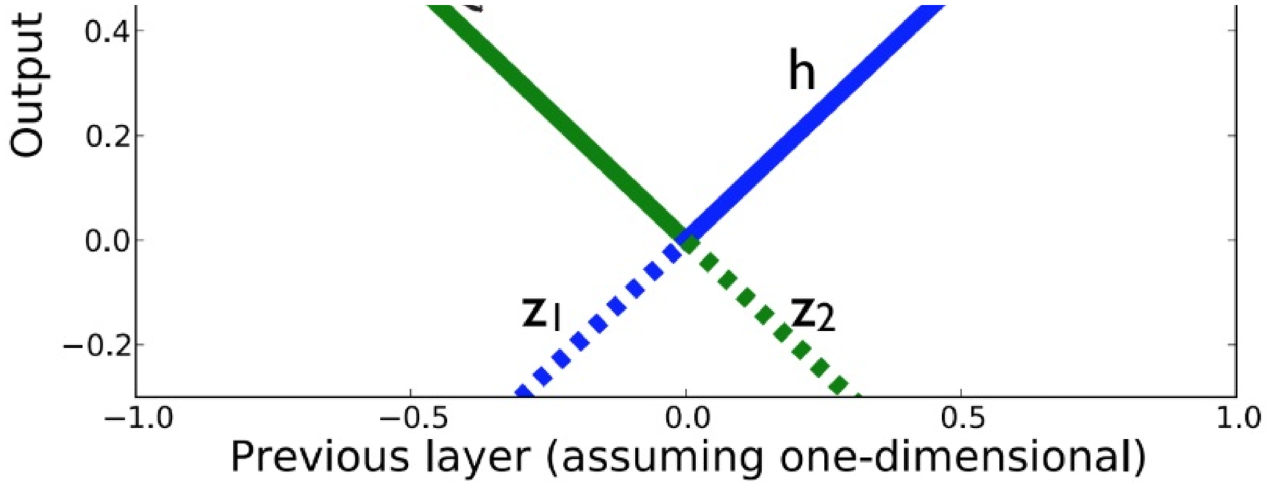
1. Unboundedness is helpful to prevent gradient from gradually approaching 0 during slow training, causing saturation. At the same time, being bounded has advantages, because bounded active functions can have strong regularization, and larger negative inputs will be resolved
2. At the same time, smoothness also plays an important role in optimization and generalization.
3. In very deep networks, swish achieves higher test accuracy than ReLU.
4. For every batch size, swish outperforms ReLU.

9. Maxout Function

Maxout activation is a generalization of the ReLU and the leaky ReLU functions. It is represented as

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$





Maxout is a learnable activation function.

Advantages of Maxout Function:-

1. It is a piecewise linear function that returns the maximum of the inputs, designed to be used in conjuction with the dropout regularization technique.
2. Both ReLU and Leaky ReLU are special cases of Maxout. The Maxout neuron, therefore, enjoys all the benifits of a ReLU unit (linear regime of operation, no saturation) and does not have its drawbacks (dying ReLU)

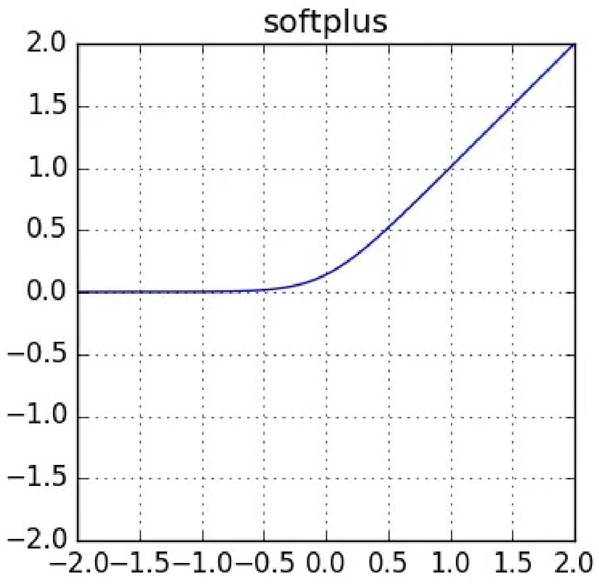
Disadvantages of Maxout Function:-

1. It doubles the total number of parameters for each neuron and hence a higher total number of parameters need to be trained

10. Softplus Function

Sofplus function is similar to ReLU function but it is relatively smooth. It is unilateral suppression like ReLU. It has a wide acceptance range (0, +∞) Formula for Softplus function is defined as f(x) = ln(1+exp x)

Chart of Softmax function:-



Advantages of Softplus Function:-

1. It has a wide acceptance domain from 0 to infinity

Disadvantages of Softplus Function:-

1. Due to exponential operation, the logarithm operation is computationally intensive and is not used.
2. Effect of ReLU is much better than Softplus

Loss Functions

A loss function is used to optimize the parameter values in a neural network model. Loss functions map a set of parameter values for the network onto a scalar value that indicates how well those parameter accomplish the task the network is intended to do.

Certainly there is a small difference between **Loss function** and **Cost function**.

Loss function is for a single training input/example and Cost function is the average loss over the entire training dataset.

There are several loss functions. Among them few are as listed below

1. Mean Absolute Error (L1 Loss)

2. Mean Squared Error (L2 Loss)

3. Huber Loss

4. Pseudo-Huber Loss

5. Hinge Loss

6. Cross-entropy Loss

7. Softmax-Cross-entropy Loss

8. Sigmoid-Cross-entropy Loss

9. Quantile Loss

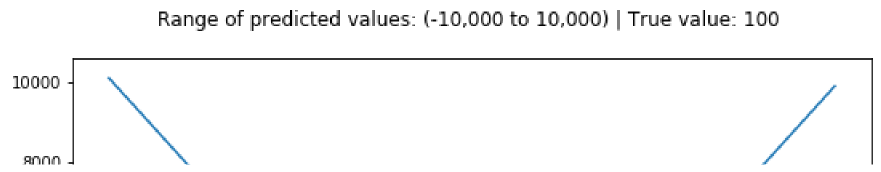
10. Log-Cosh Loss

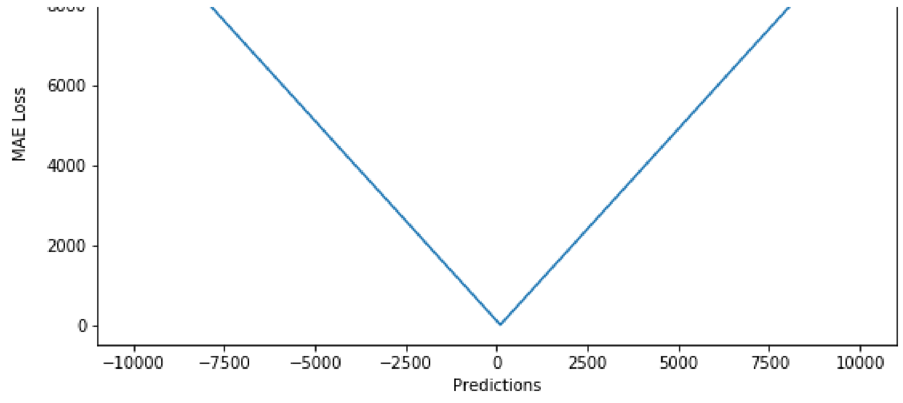
1. Mean Absolute Loss (L1 Loss)

Mean Absolute Loss (L1 Loss):

Mean Absolute Error (MAE) is used for regression models. MAE is the sum of absolute differences between our target and predicted variables. So it measures the average magnitude of errors in a set of predictions, without considering their directions (If directions also considered then it will be called as Mean Bias Error (MBE), which is sum of residuals/errors). The range is also 0 to ∞.

$$MAE = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n}$$





2. Mean Squared Error (L2 Loss)

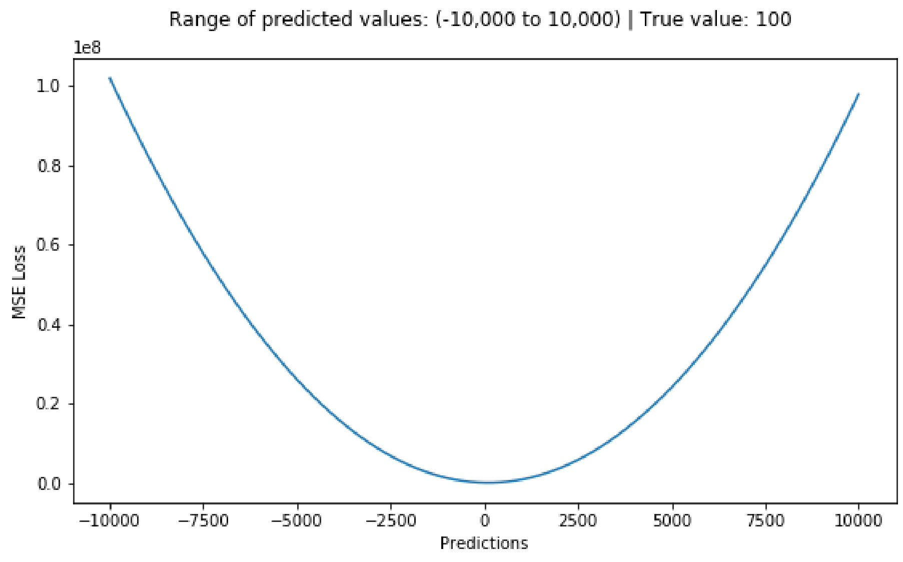
Mean Squared Error (L2 Loss):

When we have a regression task, one of the loss function we can use is Mean squared error. As the name suggests, this loss is calculated by taking the mean of squared differences between actual (target) and predicted values.

Mean Squared Error to be used where the output is a real number. i.e., it will be used when doing regression beleiving that target is conditioned on the input is normally distributed and wants large errors to be significantly (quadratically) more penalized than small ones.

$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$$

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 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

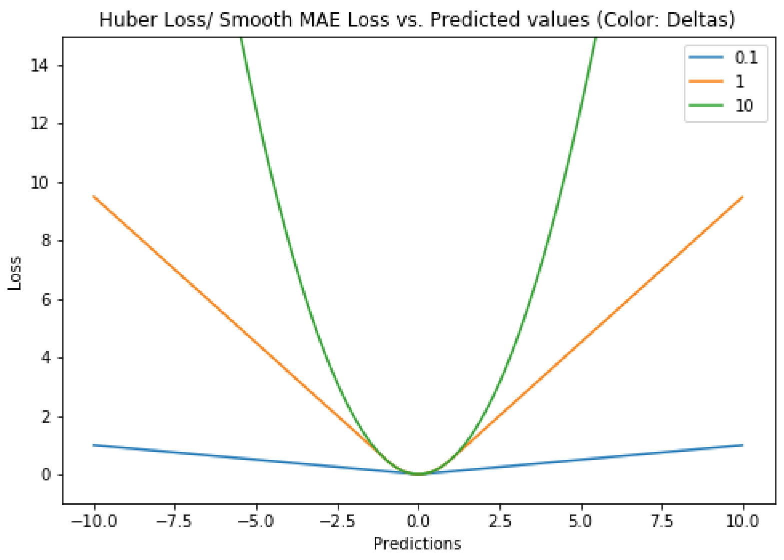
Huber loss is sensitive to outliers in data than the squared error loss. It's also differentiable at 0. It's basically absolute error, which becomes quadratic when error is small. How small that error has to be to make it quadratic depends on a hyperparameter, δ (delta), which can be tuned. Huber loss approaches MAE when δ ~ 0 and MSE when δ ~ ∞ (large numbers.)

$$L_\delta(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for} |y - f(x)| \le \delta, \\ \delta\, |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

Among them, δ is a set parameter, y represents the real values, and f(x) represents the predictedd value.

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).

The advantage of this is that when the residual is small, the loss function is L2 norm, and when the residual is large, it is a linear function of L1 norm.

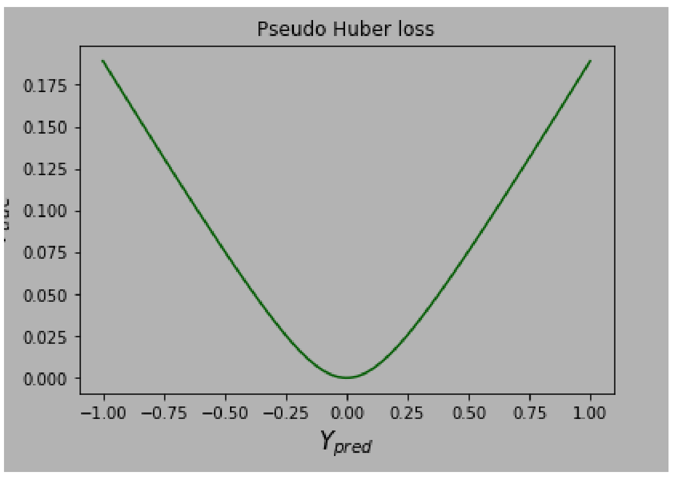


4. Pseudo-Huber Loss Function

It is a smooth approximation to the Huber Loss function. Huber loss is, a loss function used in robust regression, that is less sensitive to outliers in data than the squared error loss [LSE]. This loss function attempts to take the best of the L1 and L2 by being convex near the target and less steep for extreme values. The form depends on an extra parameter, delta, which dictates how steep it will be.

$$S = \sum_{i=1}^{n} \delta^2 \left(\sqrt{1 + \left(\frac{y_i - f(x_i)}{\delta} \right)^2} - 1 \right)$$

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



5. Hinge Loss

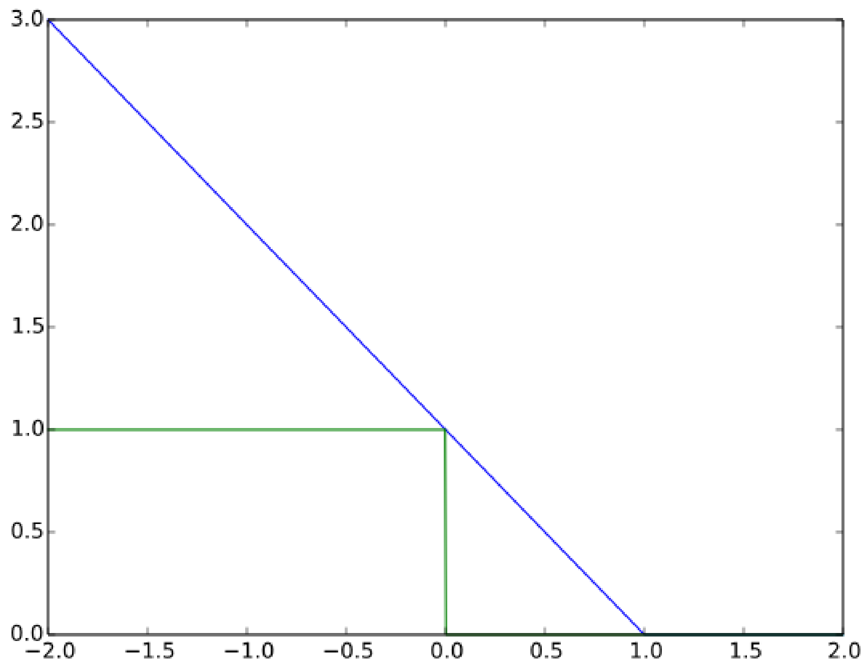
Hinge Loss is often used for binary classification, such as ground true: t=1 or -1, predicted value y = wx + b

In the SVM classifier, the definition of hinge loss is l(y) = max(0, 1-t.y)

In other words, the closer the y is to t, the smaller the loss will be.

The Hinge loss is used for "maximum margin" classification, most notably for SVMs

------------ Plot of hinge loss on Y-axis and " predicted y " on X-axis ----------



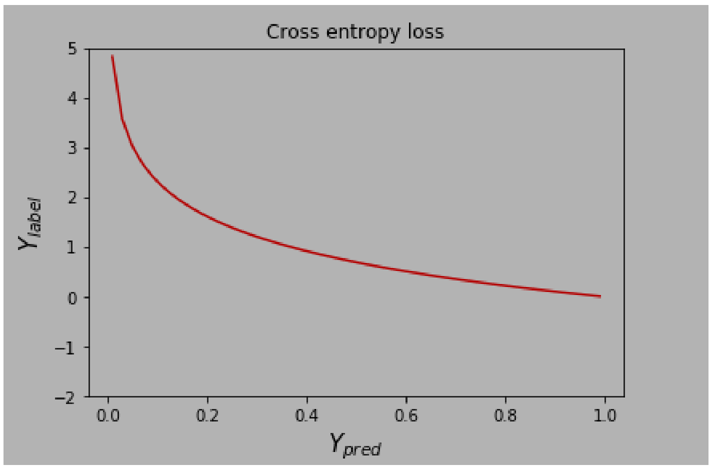
6. Cross-entropy Loss

Cross-entropy loss is a measure of the difference between two probability distributions for a given random variable or set of events.

$$CE = -\sum_{i=1}^{C'=2} t_i log(s_i) = -t_1 log(s_1) - (1-t_1) log(1-s_1)$$

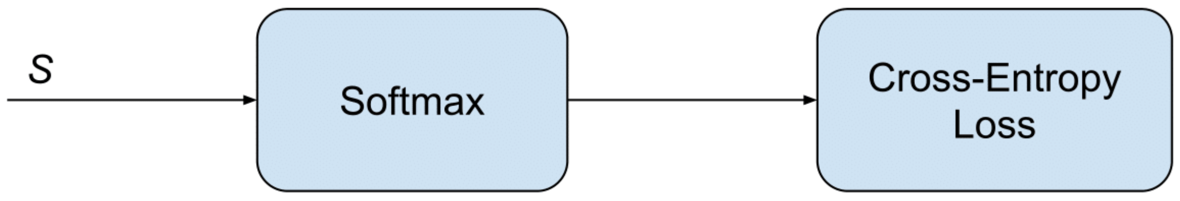
Cross-entropy loss is mainly applied to binary classification problems. The predicted value is a probability value and the loss is defined according to the crross entropy.

Note the value range of the above value: the predicted value of y should be a probability and the value range is [0,1]



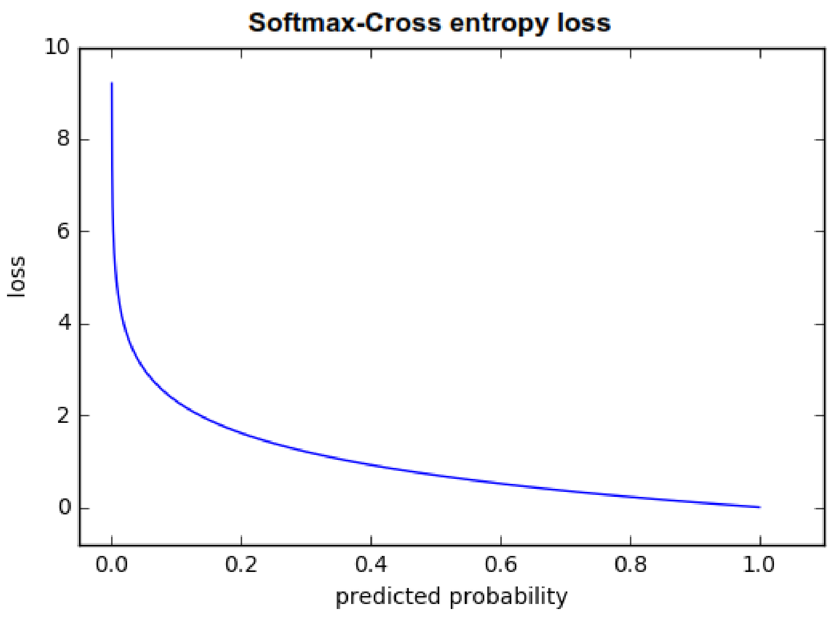
7. Softmax-Cross-entropy Loss

Also called Categorical Cross-Entropy Loss. It is a Softmax activation plus a Cross-Entropy loss. If we use this loss, we will train a CNN to output a probability over the C classes for each image. It is used for multi-class classification.



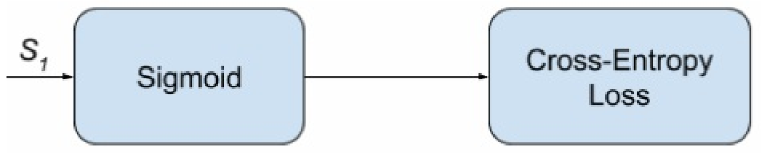
$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

In the specific (and usual) case of Multi-Class classification the labels are one-hot, so only the positive class Cp keeps its term in the loss. There is only one element of the Target vector t which is not zero ti=tp. So discarding the elements of the summation which are zero due to target labels



8. Sigmoid-Cross-entropy Loss

Also called Binary Cross-Entropy loss. It is a Sigmoid activation plus a Cross-Entropy loss. Unlike Softmax loss it is independent for each vector component (class), meaning that the loss computed for every CNN output vector component is not affected by other component values. That's why it is used for multi-label classification, were the insight of an element belonging to a certain class should not influence the decision for another class. It's called Binary Cross-Entropy Loss because it sets up a binary classification problem between C′=2 classes for every class in C, as explained above. So when using this Loss, the formulation of Cross Entroypy Loss for binary problems is often used:

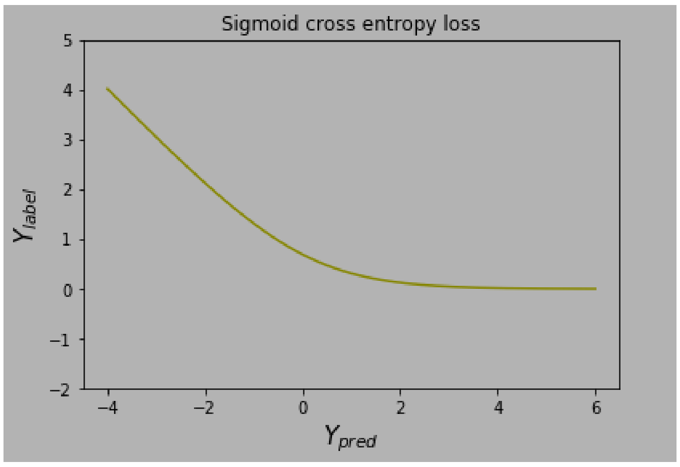


$$CE = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$$

$$f(s_1) = \frac{1}{}$$

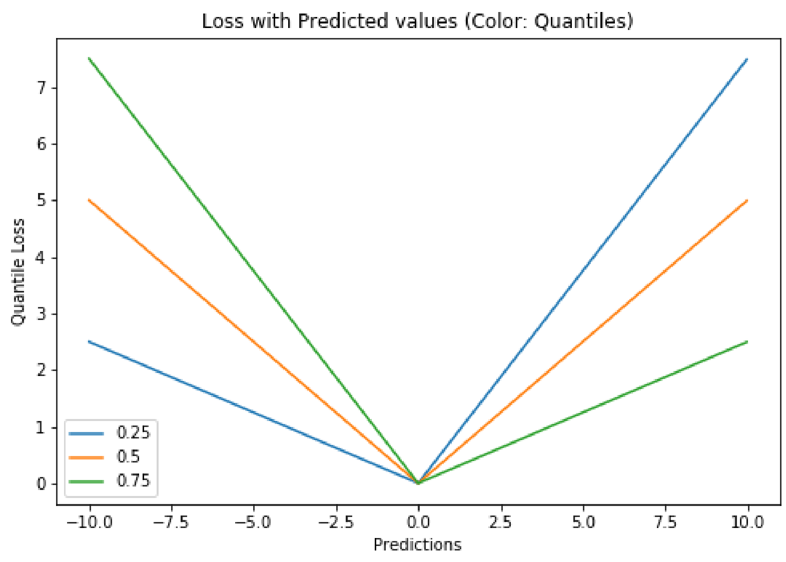
$$f(s_i) = \frac{1}{1+e^{-s_i}}$$

This would be the pipeline for each one of the C clases. We set C independent binary classification problems (C′=2). Then we sum up the loss over the different binary problems: We sum up the gradients of every binary problem to backpropagate, and the losses to monitor the global loss. s1 and t1 are the score and the gorundtruth label for the class C1, which is also the class Ci in C. s2=1−s1 and t2=1−t1 are the score and the groundtruth label of the class C2, which is not a “class” in our original problem with C classes, but a class we create to set up the binary problem with C1=Ci.



9. Quantile Loss

Quantile loss functions turn out to be useful when we are interested in predicting an interval instead of only point predictions. Prediction interval from least square regression is based on an assumption that residuals (y-y') have constant variance across values of independent variables. We can not trust linear regression models which violate this assumption. We can not trust linear regression models which violate this aasumption. We can not also just throw away the idea of fitting linear regression model as baseline by saying that such situations would always be better modeled using non-linear functions or tree based models. This is where quantile loss and quantile regression come to rescue as regression based on quantile loss porovides sensible prediction intervals even for residuals with non-constant variance or non-normal distribution.



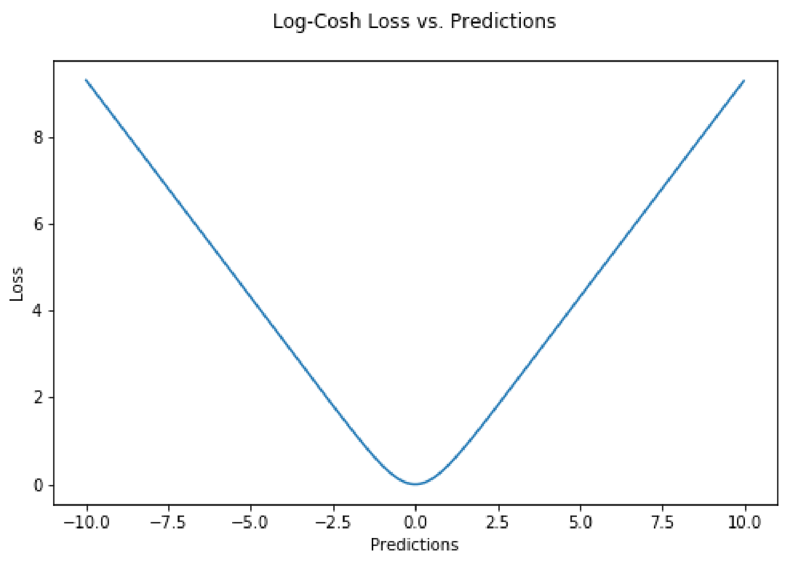
Quantile loss is actually just an extension of MAE (when quantile is 50th percentile, it's MAE)

10. Log-Cosh Loss

Log-cosh is another function used in regression tasks that's smoother than L2 loss. Log-cosh is the logarithm of the hyperbolic cosine of the prediction error.

n

$$L(y, y^p) = \sum_{i=1} \log(\cosh(y_i^p - y_i))$$



Advantage: log(cosh(x)) is approximately equal to (x**2)/2 for small x and to abs(x) - log(2) for large x. This means that 'logcosh' works mostly like the mean squared error, but will not be so strongly affected by the occassional wildly incorrect prediction. It has all the advantages of Huber loss, and it's twice differentiable everywwhere, unlike Huber loss.

But Log-cosh loss isn't perfect. It still ssuffers from the problem of gradient and hessain for very large off-target predictions being constant, therefore resulting in the absence of splits for XGBoost.

```
In [ ]:
```