# Loss function

Diffrence btn actual minus predicted

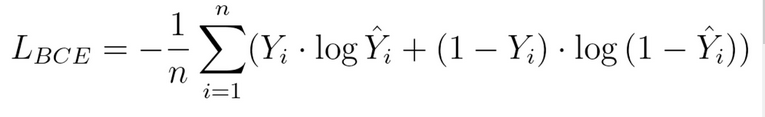
## Negative Log likelihood(torch.nn.NLLLoss)

NLL loss : -log(p\_i) where

* p\_i is the predicted probability of the correct class.
* torch.nn.CrossEntropy expect log softmax
* the lower the value of this the better the model.
* Mainly used for the classification.
* Lower NLL loss higher the predicted probablity.

## Cross-entropy loss

* It is used for classification tasks with multiple classes.
* torch.nn.CrossEntropy expect softwax
* It is calculated as the negative logarithm of the predicted probability of the correct class.
* It is often used with the softmax activation function, which produces a probability distribution over the classes.
* It is used as binary equivalent of the negative log loss likelihood loss.



## Mean squared error (MSE) :

* It is used in the regression tasks, predict is continuous value.

## Hinge loss

* Trying to maximize the seperation boundry so less likely to miss classification
* It is used for maximum-margin classification, such as support vector machines (SVMs).
* It is used for binary classification task.

## Binary cross-entropy loss

* It is used for binary classification tasks, where the goal is to predict one of two classes.
* It is calculated as the negative logarithm of the predicted probability of the correct class.

## Dice loss

* It is used with image segmentation tasks, which is a type of pixel-wise classification, where the goal is to assign a class label to each pixel in an image.
* 1 - (2\*(true positives))/(2\*(true positives) + false positives + false negatives)

## KL-divergence loss

* It is used for variational autoencoder tasks, which is a type of generative model.

Backpropagation

Parameter initialization

Speeding up gradient descent using advance optimizers

Minibatches from variables sequences