Main Concepts:

1) Loss Function

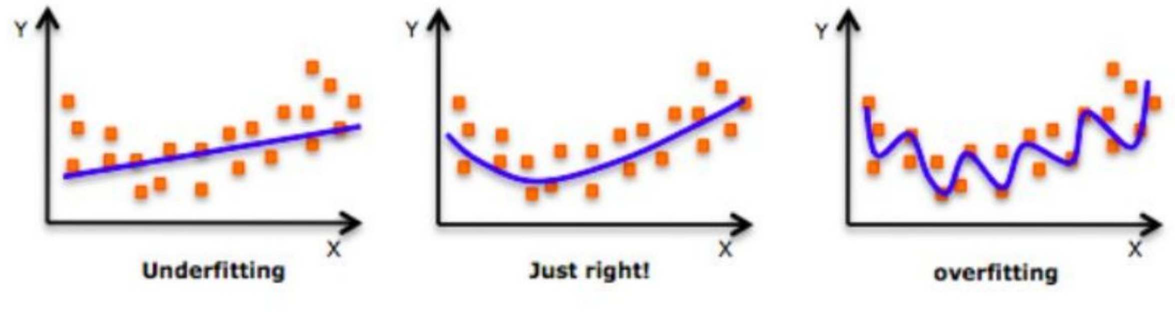
In order to train our model, we first need to define an indicator to evaluate this model is good. In fact, in machine learning, we usually define indicators to indicate that a model is bad, this indicator is called loss, and then try to minimize this indicator. A very common loss function is cross-entropy. Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label.

$$Loss = -\sum_{i=1}^{M} y_i \log \acute{y}_i$$

M - the number of classes
log - the natural log
y - the actual distribution (our one-hot vector).
y’ - our predicted probability distribution.

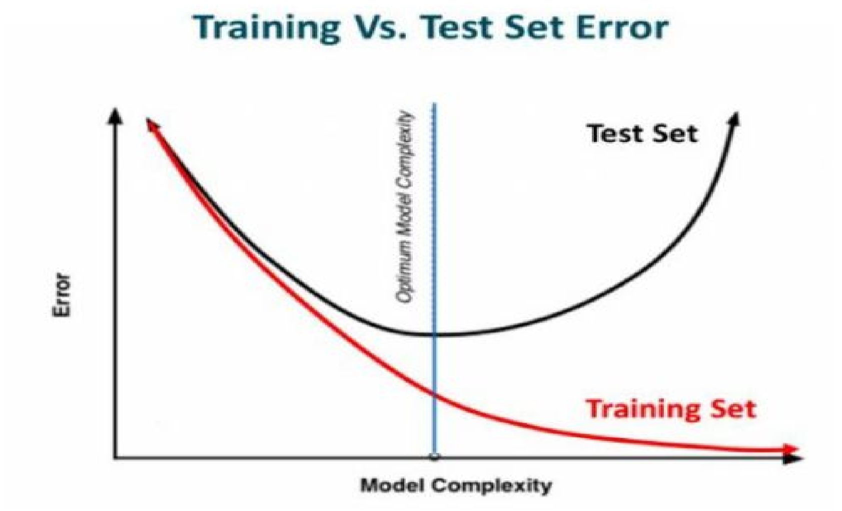
2) Regularization

Before we deep dive into the topic, take a look at this image:



As we move towards the right in this image, our model tries to learn too well the details and the

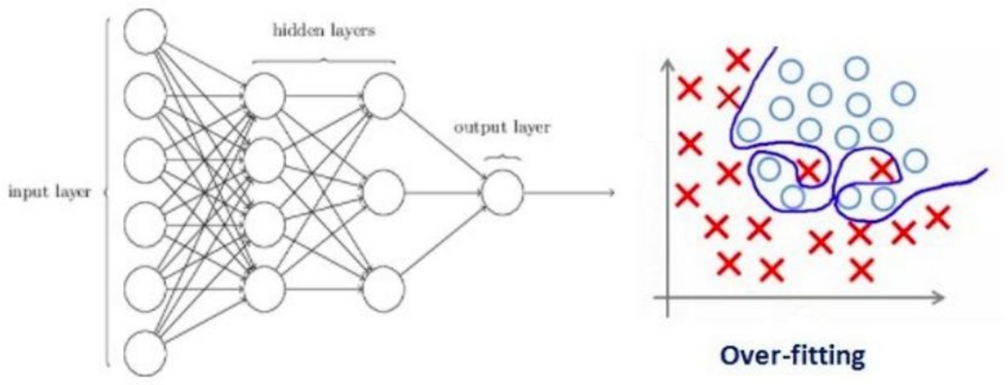
noise from the training data, which ultimately results in poor performance on unseen data. In other
words, while going towards the right, the complexity of the model increases such that the training error reduces but the testing error doesn't. This is shown in the image below.



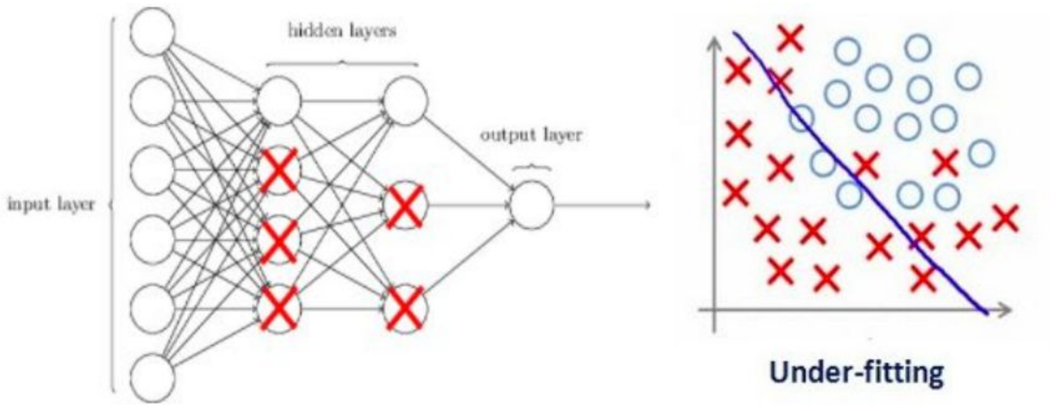
If you've built a neural network before, you know how complex they are. This makes them more prone to overfitting. Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. This, in turn, improves the model's performance on unseen data as well.

<u>How does regularization help reduce overfitting?</u>

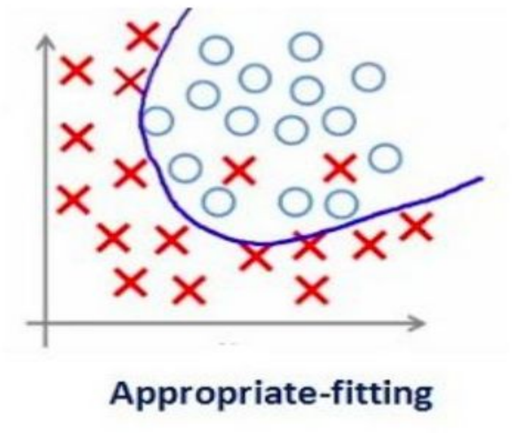
Let's consider a neural network (a more general version of logistic regression) which is overfitting on the training data as shown in the image below.



If you have studied the concept of regularization in machine learning, you will have a fair idea that regularization penalizes the coefficients. In deep learning, it actually penalizes the weight matrices of the nodes. Assume that our regularization coefficient is so high that some of the weight matrices are nearly equal to zero.



This will result in a much simpler linear network and slight underfitting of the training data Such a large value of the regularization coefficient is not that useful. We need to optimize the value of the regularization coefficient in order to obtain a well-fitted model as shown in the image below.



Different Regularization Techniques in Deep Learning:

Now that we have an understanding of how regularization helps in reducing overfitting, we'll learn a few different techniques in order to apply regularization in deep learning.

L2 & L1 regularization

L1 and L2 are the most common types of regularization. These update the general cost function by adding another term known as the regularization term.

Cost function = Loss (say, binary cross entropy) + Regularization term

Due to the addition of this regularization term, the values of weight matrices decrease because it assumes that a neural network with smaller weight matrices leads to simpler models. Therefore, it will also reduce overfitting to quite an extent.

However, this regularization term differs in L1 and L2.
In **L2**, we have:

$$\cos t\ \mathit{function} = \mathit{Loss} + \lambda \times \sum \|W\|^2$$

Here, **lambda** is the regularization hyperparameter, whose value is optimized over the validation set for better results. L2 regularization is also known as weight decay as it forces the weights to decay towards zero (but not exactly zero).

In **L1**, we have:

$$\cos t\ \mathit{function} = \mathit{Loss} + \lambda \times \sum \|W\|$$

In this, we penalize the absolute value of the weights. Unlike L2, the weights may be reduced to zero here. Hence, **it is very useful when we are trying to compress our model. Otherwise, we usually prefer L2 over it**.