Q1. What is a loss function?

هي معادلة بتحسب اد ايه ال MODEL دا كويس في انه ي فت الداتا بتاعتنا

evaluating how well your algorithm is modeling your dataset.

A loss function is a mathematical function that quantifies the difference between predicted and actual values in a machine learning model. It measures the model's performance and guides the optimization process by providing feedback on how well it fits the data.

1. Regression
   - MSE(Mean Squared Error)
   - MAE(Mean Absolute Error)
   - Hubber loss

1. Mean Squared Error/Squared loss/ L2 loss

2. The Mean Squared Error (MSE) is the simplest and most common loss function. To calculate the MSE, you take the difference between the actual value and model prediction, square it, and average it across the whole dataset.

$$MSE = \frac{1}{N}\sum_{i}^{N}(Yi - \hat{Y}i)^2$$

Mean Absolute Error/ L1 loss

The Mean Absolute Error (MAE) is also the simplest loss function. To calculate the MAE, you take the difference between the actual value and model prediction and average it across the whole dataset.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |Y_i - \hat{Y}_i|$$

Q2. What is loss and cost function in deep learning?

In deep learning, “loss function” and “cost function” are often used interchangeably. They both refer to the same concept of a function that calculates the error or discrepancy between predicted and actual values. The cost or loss function is minimized during the model’s training process to improve accuracy.

What is L1 loss function in deep learning?

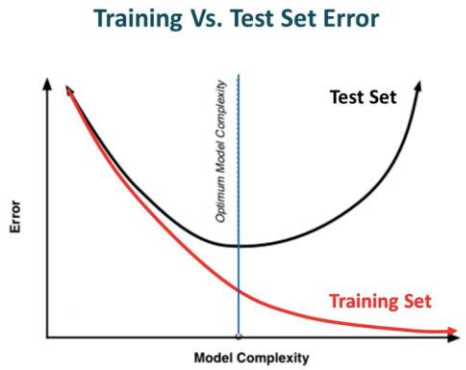
L1 loss function, also known as the mean absolute error (MAE), is commonly used in deep learning. It calculates the absolute difference between predicted and actual values. L1 loss is robust to outliers but does not penalize larger errors as strongly as other loss functions like L2 loss.

What is Regularization?

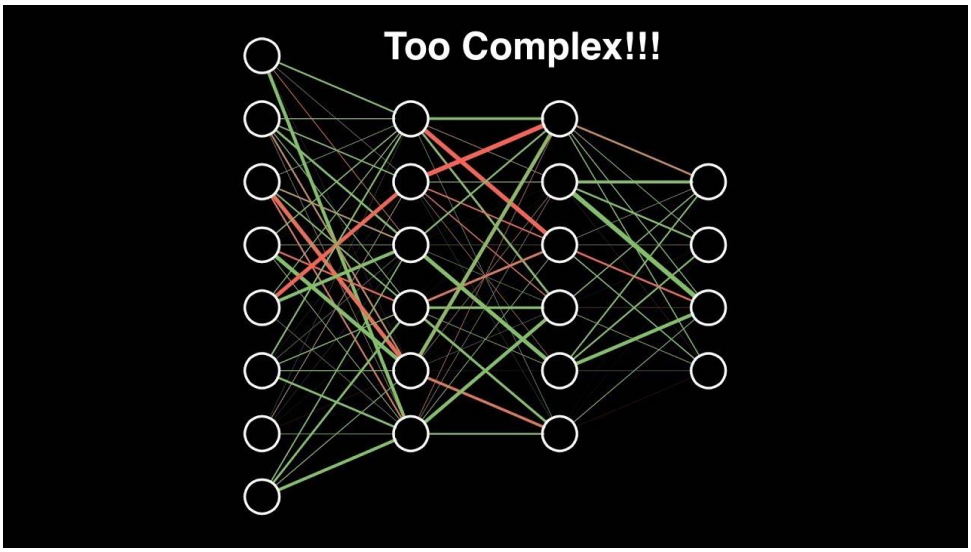
Regularization is a technique used in machine learning and deep learning to prevent overfitting and improve the generalization performance of a model. It involves adding a penalty term to the loss function during training.

This penalty discourages the model from becoming too complex or having large parameter values, which helps in controlling the model's ability to fit noise in the training data. Regularization methods include L1 and L2 regularization, dropout, early stopping, and more. By applying regularization, models become more robust and better at making accurate predictions 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.



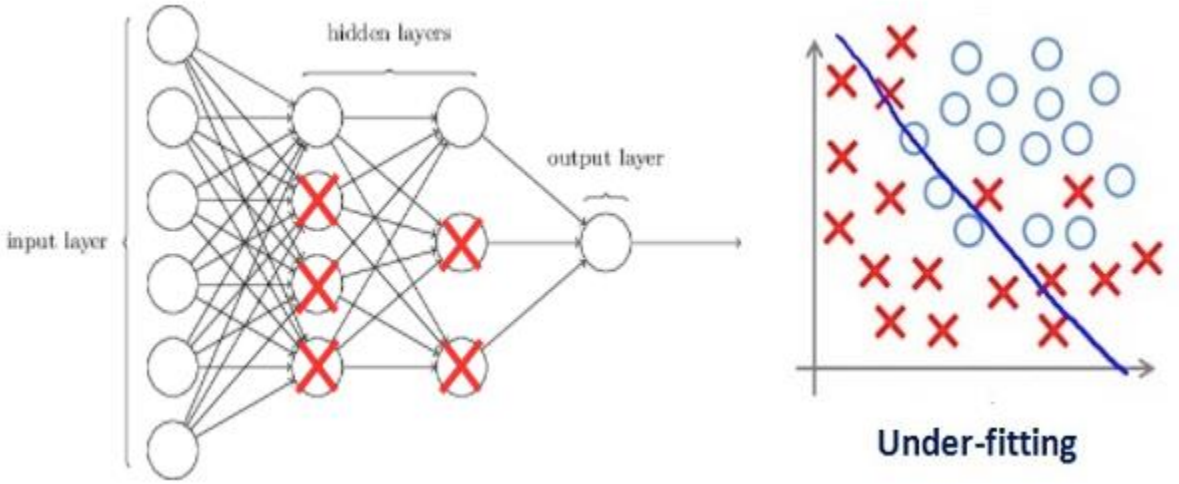
If you've built a neural network before, you know how complex they are. This makes them more prone to overfitting.



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 the unseen data as well.

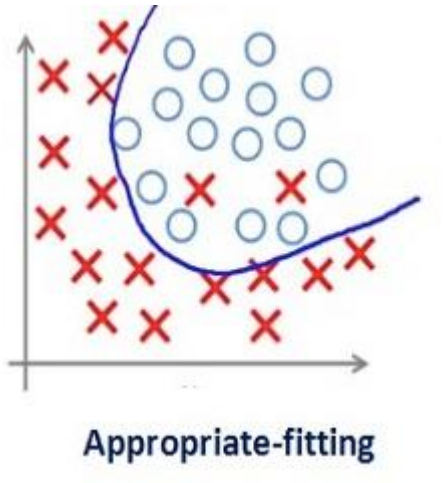
If you have studied the concept of regularization **in machine learning, you will have a fair idea that regularization penalizes the coefficients. In deep learning, it actually penalizes the weight matrices of the nodes.**

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 regularization coefficient in order to obtain a well-fitted model as shown in the image below.



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:

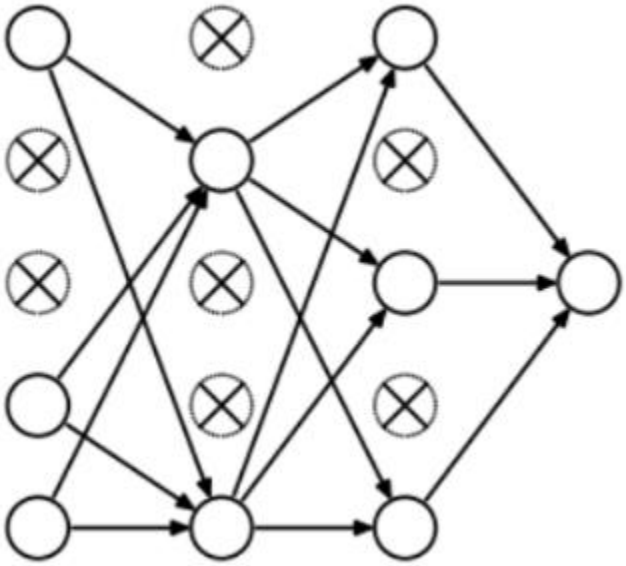
$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum ||w||^2$$

Here, **lambda** is the regularization parameter. It is the hyperparameter whose value is optimized 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:

$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum ||w||$$

So what does dropout do? At every iteration, it randomly selects some nodes and removes them along with all of their incoming and outgoing connections as shown below.



Data Augmentation

The simplest way to reduce overfitting is to increase the size of the training data. In machine learning, we were not able to increase the size of training data as the labeled data was too costly.

But, now let's consider we are dealing with images. In this case, there are a few ways of increasing the size of the training data – rotating the image, flipping, scaling, shifting, etc. In the below image, some transformation has been done on the handwritten digits dataset.

This technique is known as data augmentation. This usually provides a big leap in improving the accuracy of the model. *It can be considered as a mandatory trick in order to improve our predictions.*

In *keras*, we can perform all of these transformations using ImageDataGenerator. It has a big list of arguments which you you can use to pre-process your training data.

Below is the sample code to implement it.

```
from keras.preprocessing.image import ImageDataGenerator
datagen = ImageDataGenerator(horizontal flip=True)
datagen.fit(train)
```

Early stopping

Early stopping is a kind of cross-validation strategy where we keep one part of the training set as the validation set. When we see that the performance on the validation set is getting worse, we immediately stop the training on the model. This is known as early stopping.

In the above image, we will stop training at the dotted line since after that our model will start overfitting on the training data.