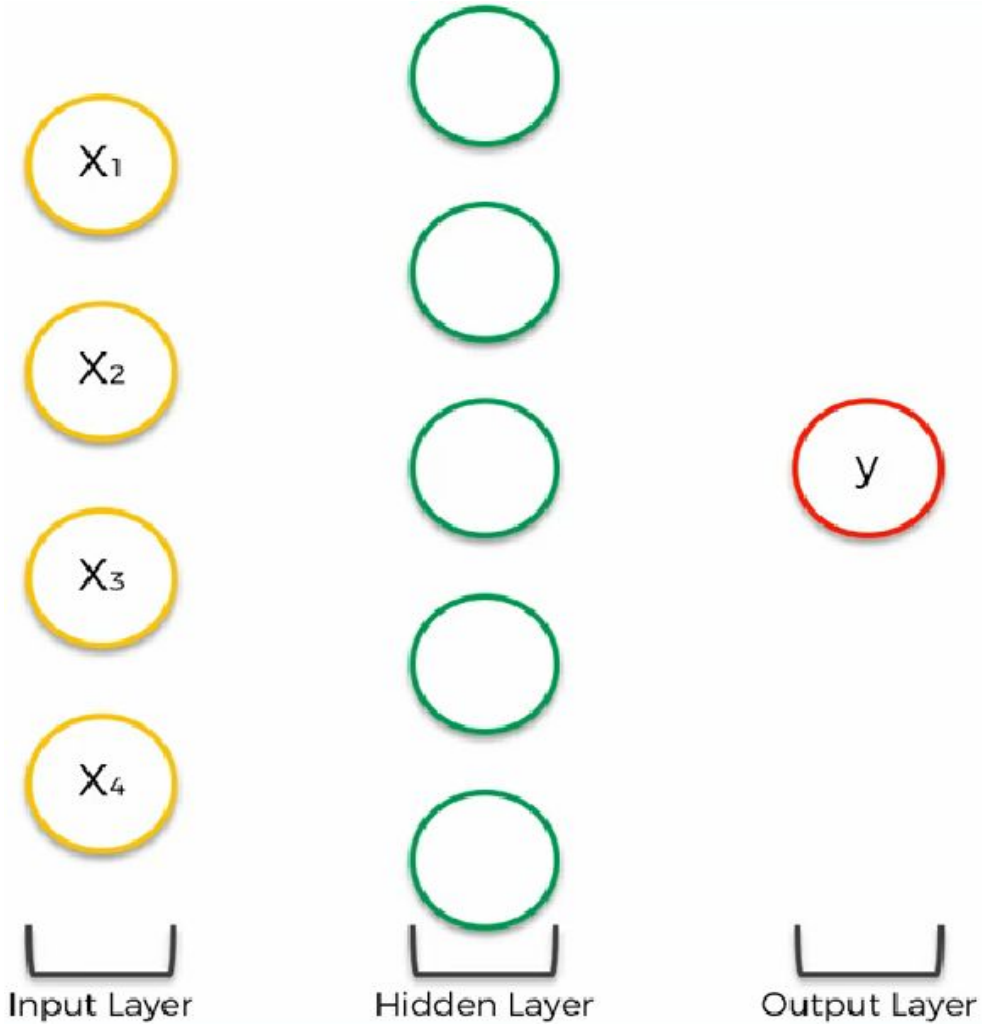
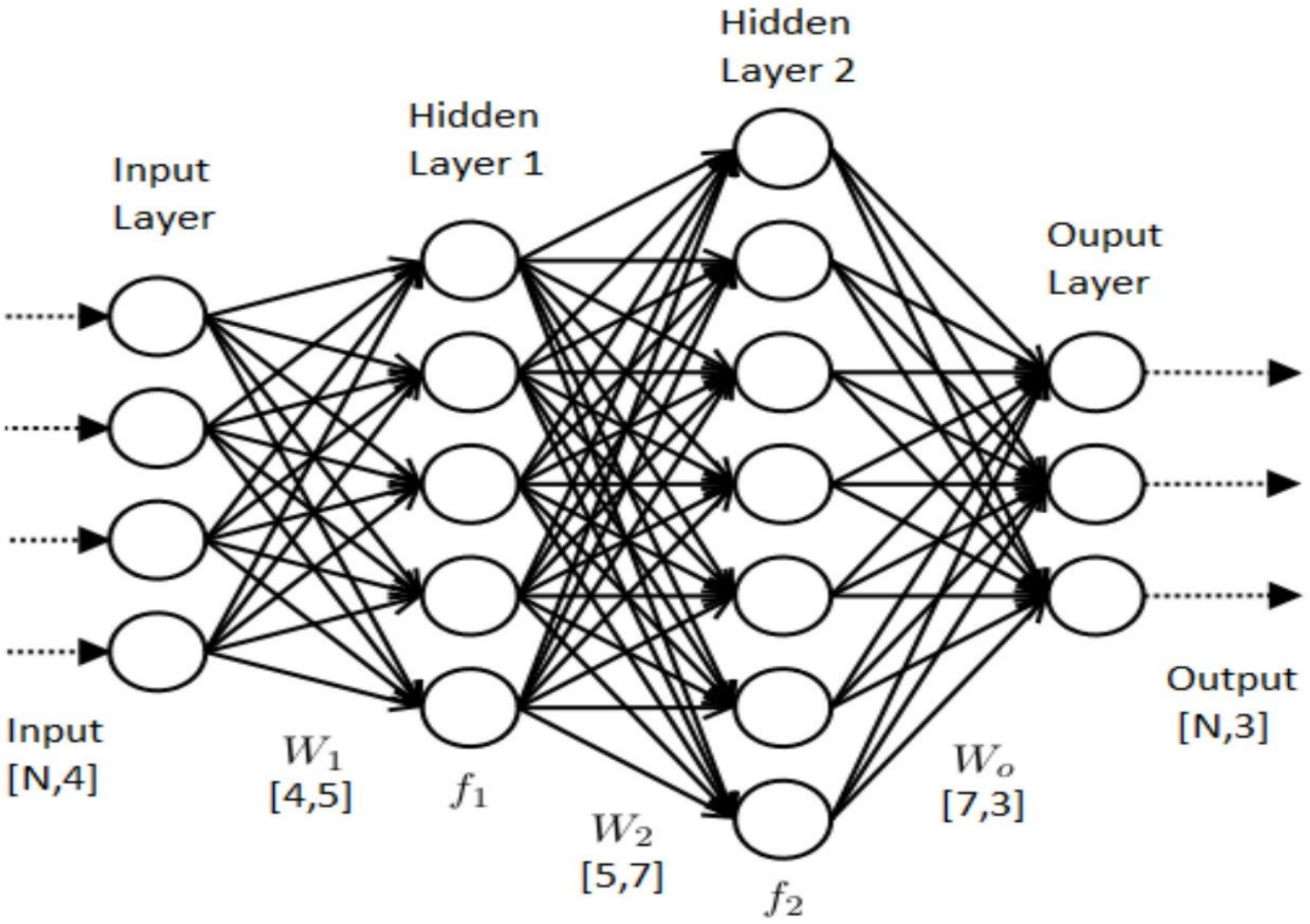
principle



principle


Hidden
Layer 2
Hidden
Layer 1
Input
Layer
Ouput
Layer
Input
[N,4]
W_1
[4,5]
f_1
W_2
[5,7]
f_2
W_o
[7,3]
Output
[N,3]

Step

STEP 1: Randomly initialize the weights to small numbers close to 0 (but not 0)

STEP 2: Input the first observation of your dataset in the input layer, each feature in one input node.

STEP 3: Forward-Propagation: from left to right, the neurons are activated in a way that the impact of each neuron's activation is limited by the weights. Propagate the activations until getting the predicted result y.

STEP 4: Compare the predicted result to the actual result. Measure the generated error.

STEP 5: Back-Propagation: from right to left, the error is back-propagated.
Update the weights according to how much they are responsible for the error. The learning rate decides by how much we update the weights.

STEP 6: Repeat Steps 1 to 5 and update the weights after each observation (Reinforcement Learning).
Or: Repeat Steps 1 to 5 but update the weights only after a batch of observations (Batch Learning).

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of **TensorFlow, CNTK, or Theano**. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).

Supports both convolutional networks and recurrent networks, as well as combinations of the two.

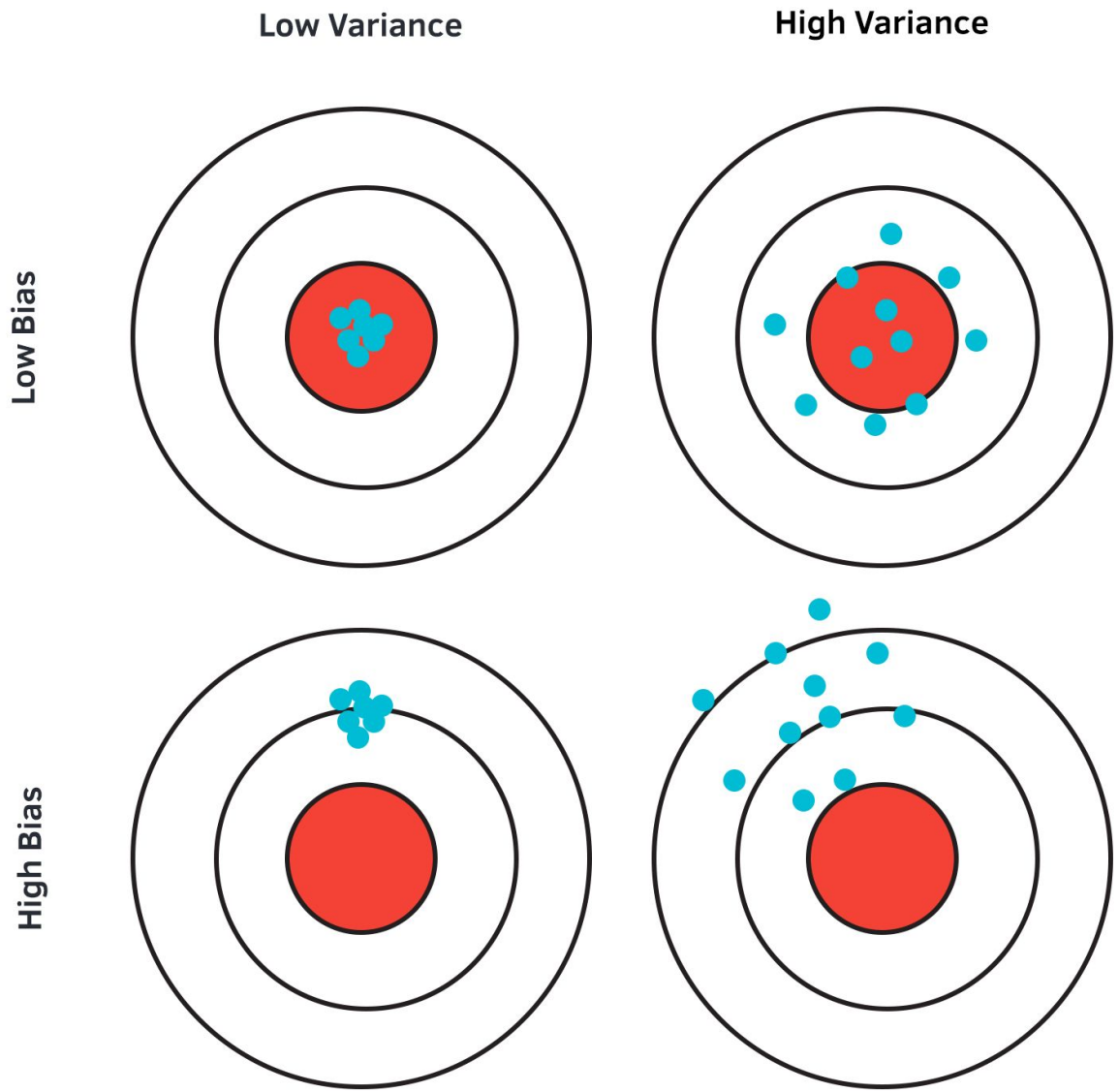
Runs seamlessly on CPU, GPU and TPU.

TensorFlow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. Wikipedia

Now its time to,

Practical Working on *Google COLAB*



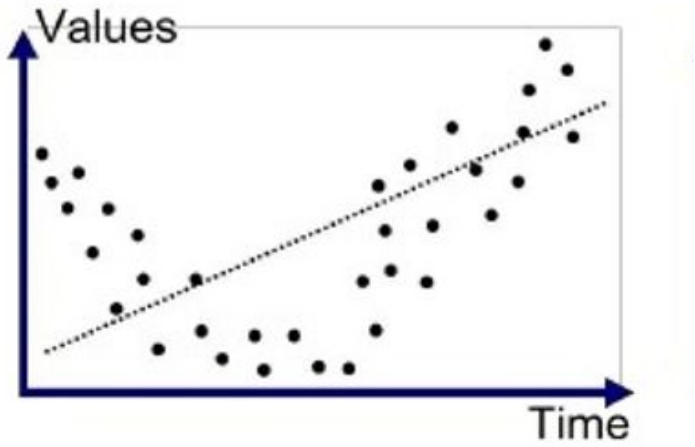
Bias-Variance Tradeoff

To Handle Overfitting Problem

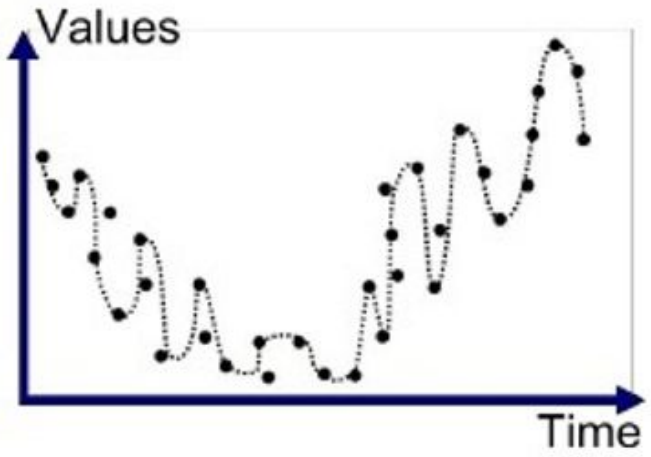
- Bias and Variance Problem (tradeoff)
    - *irreducible Error*
- Regularization (L1 Ridge Regression , L2 Lasso Regression
- Drop out Method (Regularization)

What is Bias and Variance

Underfitting and Overfitting (Bias and Variance)



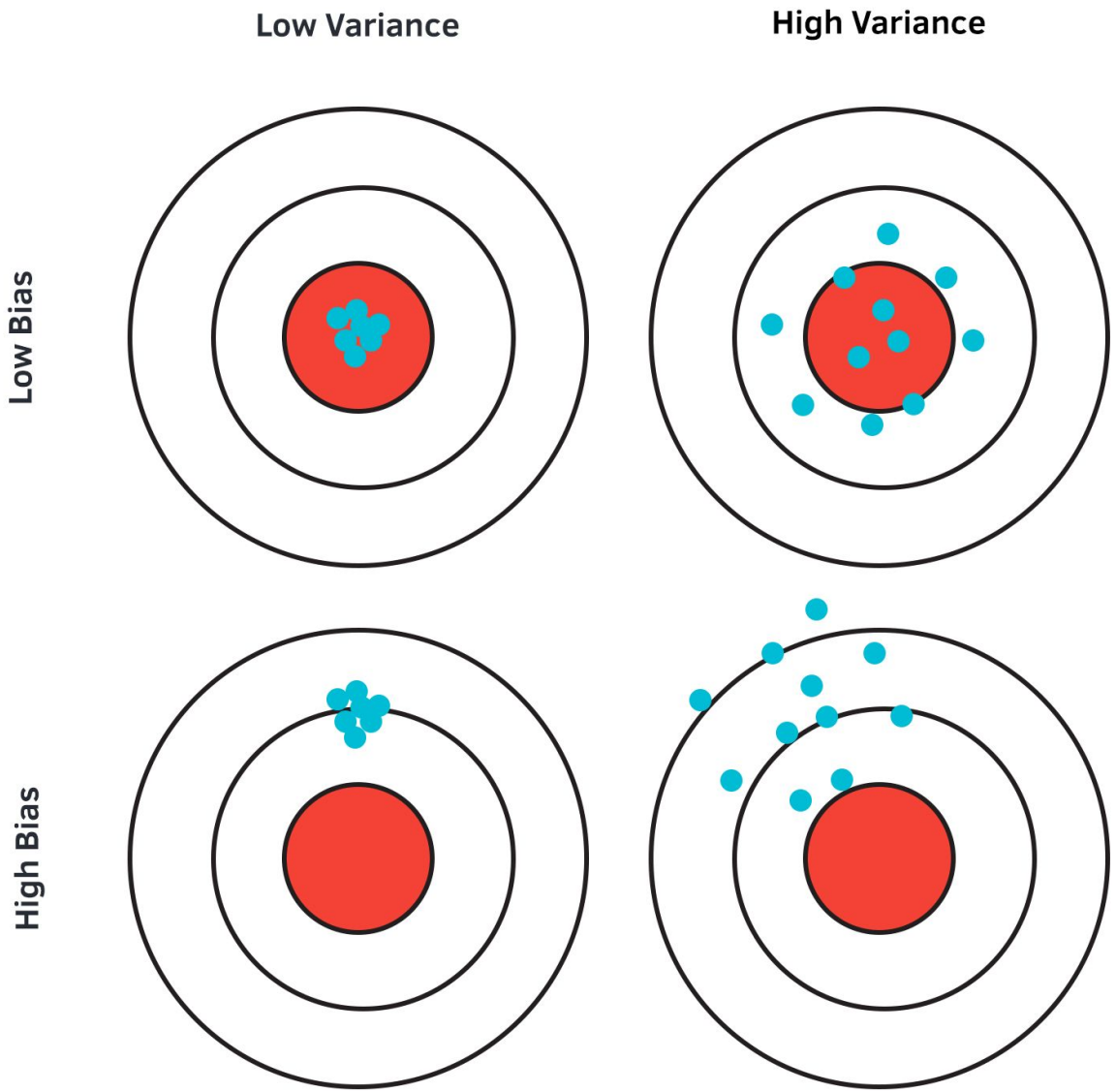
Underfitted



Overfitted

So, What will be the best model

Low Bias, Low Variance



Bias-Variance Tradeoff

So, what about the ***irreducible Error***, actually it is errors in data collected at the source. Noisiness and outlines introduced at source level from data collected!

Thanks

Helpful link:
https://towardsdatascience.com/holy-grail-for-bias-variance-tradeoff-overfitting-underfitting-7fad64ab5d76

First we create the K fold in Breast Cancer dataset first then, we understand the others Regularization then finally we move towards the Parameters Tuning…

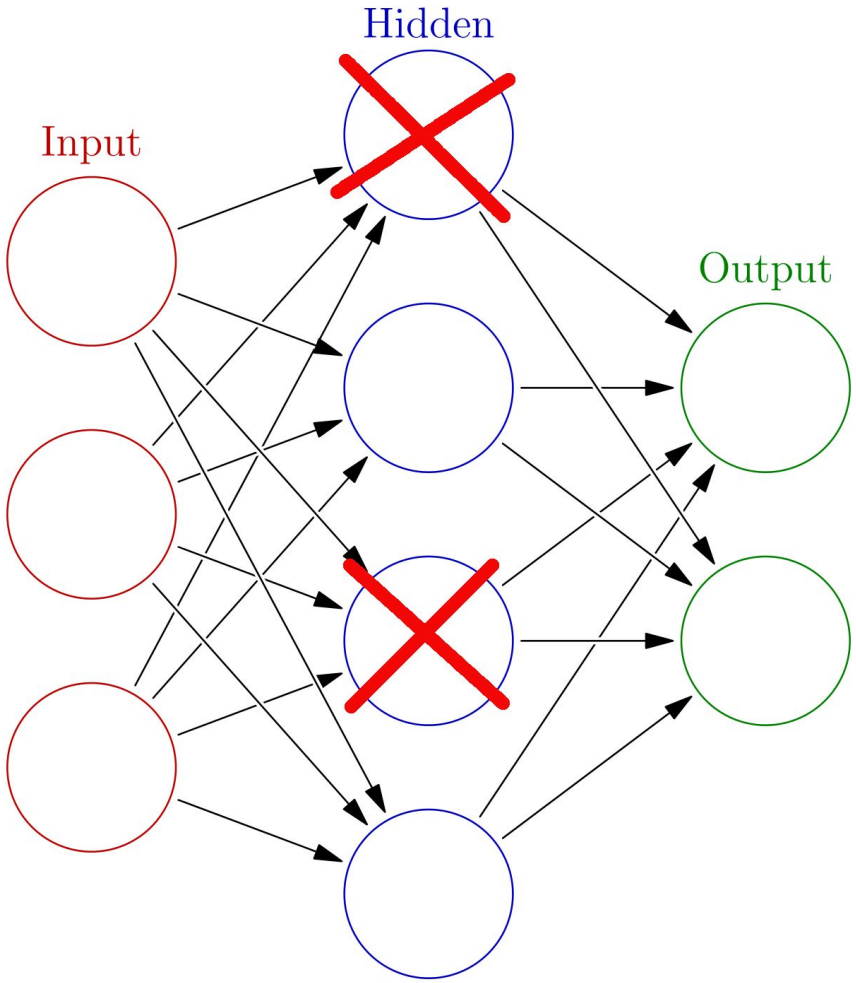
Regularization,

(L2) Ridge Regression

(L1) Lasso Regression

Regularization is a process of introducing additional information in order to prevent overfitting.

Drop out Method (Regularization)



Drop out ratio

0 <= p >= 1

Eg: = 0.2

Parameters Tuning in ANN

Parameter VS Hyper-parameters

Model Parameters are something that a model learns on its own. For example,

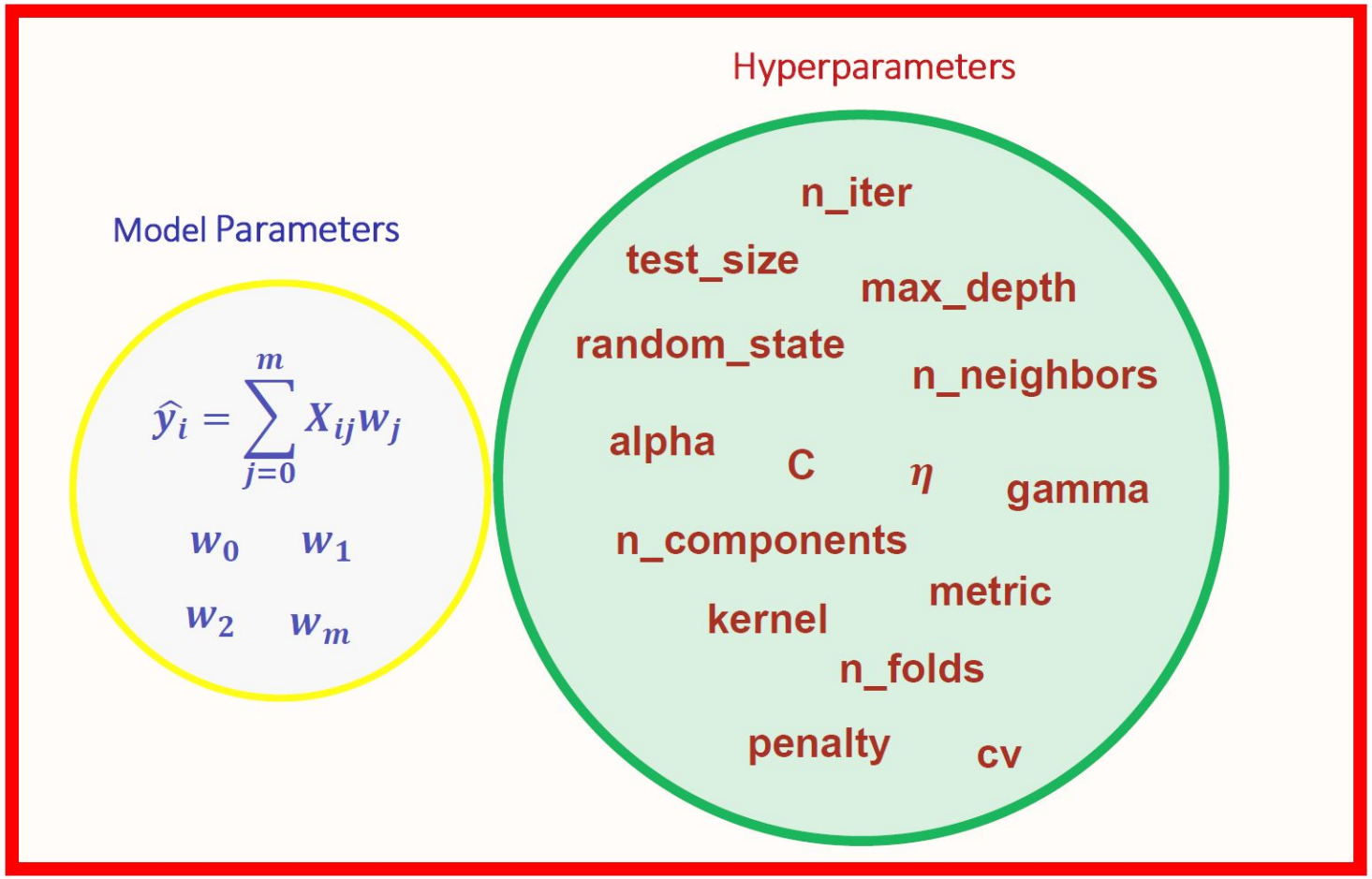
1) Weights or Coefficients of independent variables in Linear regression model.
2) Weights or Coefficients of independent variables SVM.
3) Split points in Decision Tree.

Model hyper-parameters are used to optimize the model performance. For example

1)Kernel and slack in SVM
2)Value of K in KNN
3)Depth of tree in Decision trees.

Parameters Tuning in ANN

Parameter VS Hyper-parameters



Lets understand the this Hyper parameters (optimal) size or quantity in ANN using Parameter Tuning!

Parameters Tuning in ANN
Parameter VS Hyper-parameters

Grid searching of hyperparameters

Grid search is an approach to hyperparameter tuning that will methodically build and evaluate a model for each combination of algorithm parameters specified in a **grid**.

What is Optimizer and different types it!

Optimizers are algorithms or methods used to change the attributes of your neural network such as weights and learning rate in order to reduce the losses.
Optimization algorithms or strategies are responsible for reducing the losses and to provide the most accurate results possible

Role of an optimizer

Optimizers update the weight parameters to minimize the loss function. Loss function acts as guides to the terrain telling optimizer if it is moving in the right direction to reach the bottom of the valley, the global minimum.

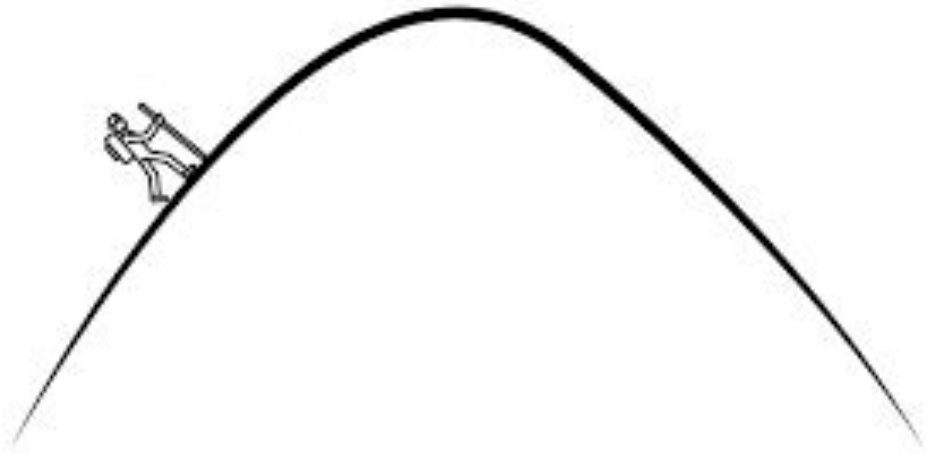
What is Optimizer and different types it!

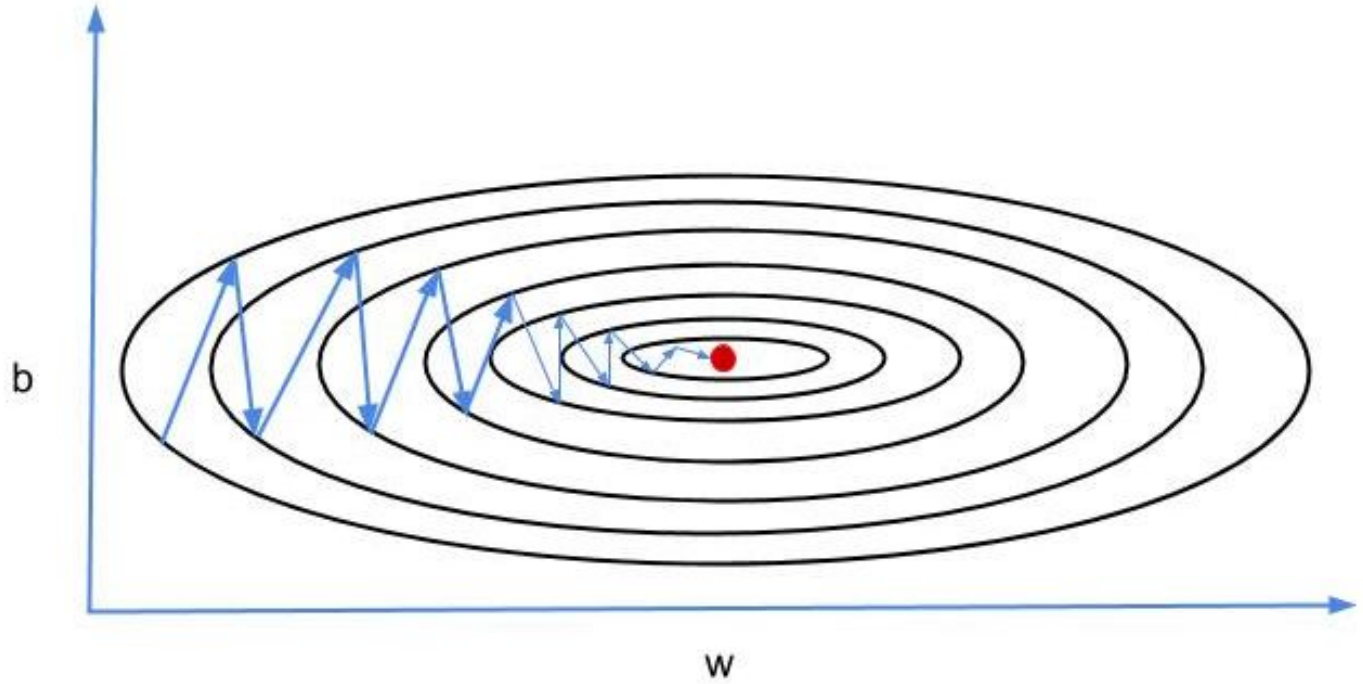
Types of Gradient Descent: *Different types of Gradient descents are*

- **Batch Gradient Descent or Vanilla Gradient Descent**
- **Stochastic Gradient Descent**
- **Mini batch Gradient Descent**

Gradient Descent with Momentum

Mini-batch gradient descent makes a parameter update with just a subset of examples, the direction of the update has some variance, and so the path taken by mini-batch gradient descent will “oscillate” toward convergence. Gradient Descent with Momentum considers the past gradients to smooth out the update. It ***computes an exponentially weighted average of your gradients,*** and then use that gradient to update your weights instead. It works faster than the standard gradient descent algorithm.





Gradient Descent with Momentum considers the past gradients to smooth out the update. It computes an exponentially weighted average of your gradients, and then use that gradient to update your weights instead.

Gradient Descent with Momentum

During backward propagation, we use dW and db to update our parameters W and b as follows:

$$W = W - \text{learning rate} * dW$$
$$b = b - \text{learning rate} * db$$

In momentum, instead of using dW and db independently for each epoch, we take the exponentially weighted averages of dW and db.

$$V_{dW} = \beta \times V_{dW} + (1 - \beta) \times dW$$
$$V_{db} = \beta \times V_{db} + (1 - \beta) \times db$$

Where beta 'β' is another hyperparameter called momentum and ranges from 0 to 1. It sets the weight between the average of previous values and the current value to calculate the new weighted average.
After calculating exponentially weighted averages, we will update our parameters.

$$W = W - \text{learning rate} * V_{dW}$$
$$b = b - \text{learning rate} * V_{db}$$

Adagrad Optimizer

Adagrad is an **optimizer** with parameter-specific learning rates, which are adapted relative to how frequently a parameter gets updated during training. The more updates a parameter receives, the smaller the learning rate.

AdaGrad or adaptive gradient allows the learning rate to adapt based on parameters. It performs larger updates for infrequent parameters and smaller updates for frequent one. Because of this it is well suited for sparse data (NLP or image recognition). Another advantage is that it basically eliminates the need to tune the learning rate. Each parameter has its own learning rate and due to the peculiarities of the algorithm the learning rate is monotonically decreasing. This causes the biggest problem: at some point of time the learning rate is so small that the system stops learning

Two Features in Learning stage in (NN)

Dense (Matrix) most of value in matrix is NOT zero (0)

Sparse (Matrix) most of value in matrix is zero (0)

taking cumulative sum of squared

Adadelta & RMSProp Optimizer

RMSprop

Root mean square prop or RMSprop is another adaptive learning rate that is an improvement of AdaGrad. Instead of taking cumulative sum of squared gradients like in AdaGrad, we take the exponential moving average of these gradients.

Like RMSprop, Adadelta (2012) is also another improvement from AdaGrad, focusing on the learning rate component. Adadelta is probably short for 'adaptive delta', where *delta* here refers to the difference between the current weight and the newly updated weight.

The difference between Adadelta and RMSprop is that Adadelta removes the use of the learning rate parameter completely by replacing it with D, the exponential moving average of squared deltas.

Adaptive moment estimation, or Adam Optimizer

Adaptive moment estimation, or Adam (2014), ***is a combination of momentum and RMSprop.***

It acts upon,

(i) The gradient component by using V, the exponential moving average of gradients (like in momentum) and

(ii) The learning rate component by dividing the learning rate α by square root of S, the exponential moving average of squared gradients (like in RMSprop).

W = W – learning rate * dW

$V_{dW} = \beta \times V_{dW} + (1 - \beta) \times dW$
$V_{db} = \beta \times V_{db} + (1 - \beta) \times db$

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{S}_t} + \epsilon} \cdot \hat{V}_t$$

where

$$\hat{V}_t = \frac{V_t}{1 - \beta_1^t}$$

$$\hat{S}_t = \frac{S_t}{1 - \beta_2^t}$$

are the bias corrections, and

$$V_t = \beta_1 V_{t-1} + (1 - \beta_1) \frac{\partial L}{\partial w_t}$$

$$S_t = \beta_2 S_{t-1} + (1 - \beta_2) \left[\frac{\partial L}{\partial w_t} \right]^2$$

with V and S initialised to 0.

Converulation neural network (CNN)

But understanding first understand the two core field of A.I

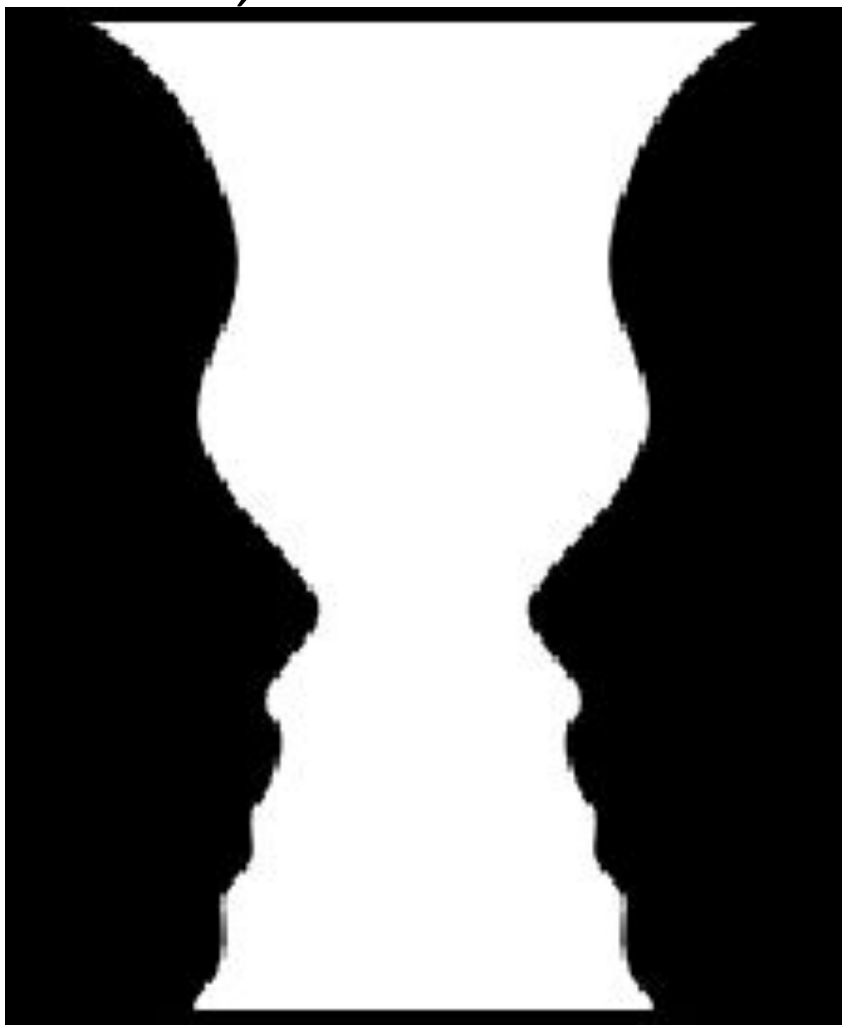
- ***Image Processing***
- ***Computer vision***

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies. It forms core research area within engineering and computer science disciplines too.

Computer Vision, often abbreviated as CV, is defined as a field of study that seeks to develop techniques to help computers “see” and understand the content of digital images such as photographs and videos.

Converulation neural network (CNN)

So, what is CNN



Converulation neural network (CNN)

So, what is CNN

Converulation neural network (CNN)

So, what is CNN

A convolutional neural network (**CNN**) is a specific type of artificial neural network that uses perceptrons, a machine learning unit algorithm, for supervised learning, to analyze data. CNNs apply to image processing, natural language processing and other kinds of cognitive tasks.