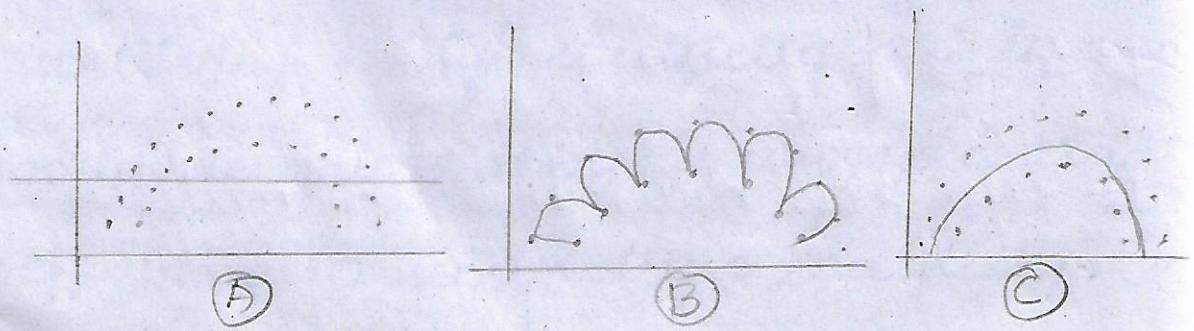
LECTURE - 54

There are some important concepts in Machine Learning. They are

1. Overfitting
2. Underfitting (BASIC MODEL)
3. Bias
4. Variance
5. Bias Variance Trade off
6. Hyperparameter Tuning.

→ In order to prevent the overfitting & Underfitting we use Regularization term i.e., where the model fits well.

*Now let us see one by one in detailed manner.

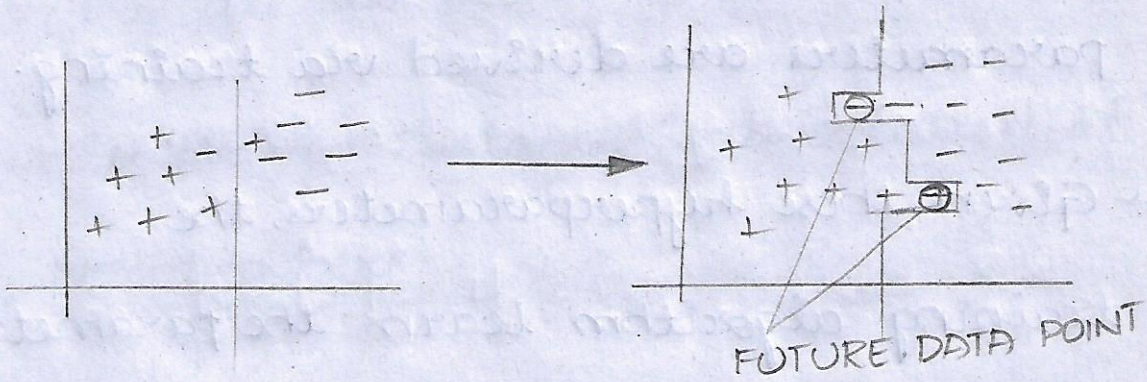


(A) → UNDERFITTING

(B) → OVERFITTING

(C) → BEST MODEL

→ For KNN, lets see the hyperparameters.



So, from the above figure, we can say that,

As K increases → Underfitting i.e.,
High Bias & low variance.

As K decreases → Overfitting i.e.,
High variance & low Bias

↓

HOW OUR MODEL IS

Here K is a hyperparameter.

WHAT IS A HYPERPARAMETER?

It is a parameter whose value is used to control the learning process.

→ By contrast, the values of other parameters are derived via training.

→ Given these hyperparameters, the training algorithm learns the parameters from the data.

HYPERPARAMETERS IN KNN:

1. What value of K (tuning)?
2. Which distance metric to use?

For Distance calculation we have,

↳ Euclidean Distance.

↳ Manhattan Distance

↳ Minkowski Distance.

Euclidean Distance = $\sqrt{\sum_{i=1}^{d} (y_i - x_i)^2}$ or

$$\left[\sum_{i=1}^{d} (y_i - x_i)^2\right]^{1/2}$$ where

$x_i - \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}$, $y_i - \begin{bmatrix} y_1 \\ \vdots \\ y_d \end{bmatrix}$.

x_i & y_i are in Vector Notation.

The Euclidean Distance is also called as

d_n Norm where

$n = 1, 2 \ldots$

d_2 Norm is written as $\|w\|_2$ - EUCLIDEAN

d_1 Norm is written as $\|w\|_1$ - MINKOWSKI

MANHATTAN DISTANCE: Walking on Grids

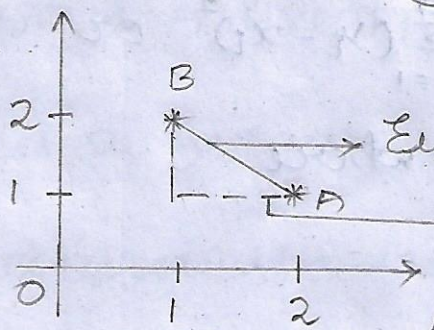
The distance between two points measured along the axes at right angles.

Manhattan Distance = $\sum_{i=1}^{d} abs(y_i - x_i)$

↓

absolute

If $x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $y = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$



A is 2 distance away from the B.

Mathematically, Manhattan Distance is

$$\Rightarrow \sum_{i=1}^{d} abs(y_i - x_i) = |1-2| + |2-1| = 2$$

MINKOWSKI DISTANCE:

As we know that, we use manhattan distance is to calculate the distance between the two data points in a grid like path.

→ So, As mentioned above, we use Minkowski distance formula to find Manhattan distance by setting P's values

as 1.

⇒ Minkowski Distance $= \left(\sum_{i=1}^{d} |y_i - x_i|^p\right)^{1/p}$

So, the Minkowski distance is the Generalised formula for finding distance.

NOTE:

→ CASE - I : If $P = 1 \leftrightarrow$ Manhattan Distance

→ CASE - II : If $P = 2 \leftrightarrow$ Euclidean Distance.

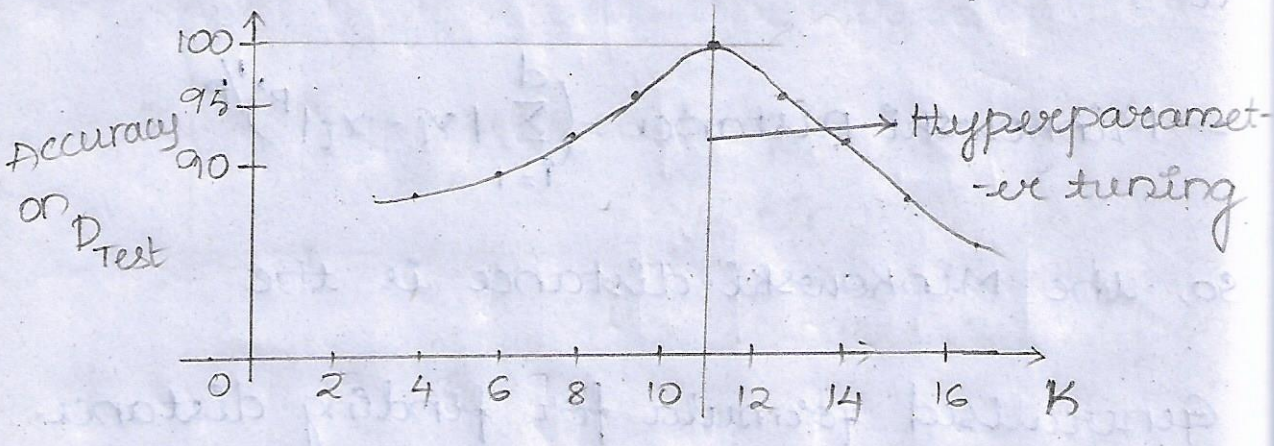
* P can be any real value, typically set to a value between 1 & 2.

EXPERIMENT :

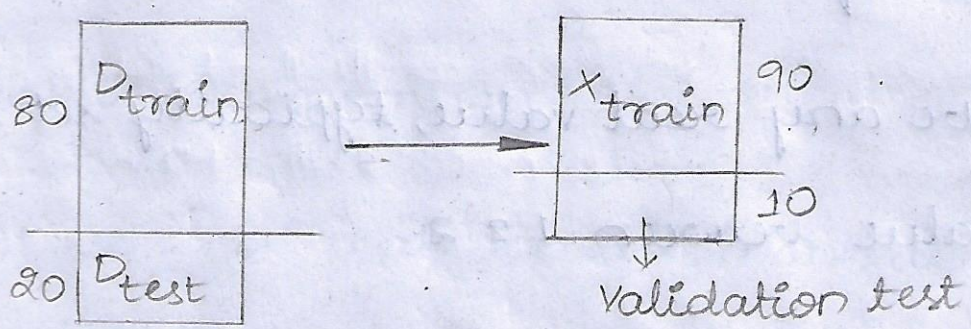
$K = [1, 3, 9, 11, 12, 13, 15]$, $P = [1, 2, 3, 4, 5, 6]$

The code to find the Accuracy :

```
model = KNC(n_neighbors= )
model.fit(x_train, y_train)
y_Pred = model.pred(x_test)
acc = metrics.acc_score(y_test, y_pred)
```



If the Accuracy is same for 2 values of K, then we will take the 1st value.



NOTE:

Model should not change with the change in the training data.

∴ KNN - Underfitting → Making basic/ Naive assumptions to simplify things leads to Bias.

∴ K has NO parameters.

Some Examples of Machine learning and hyperparameters are :

→ Learning Rate
→ Number of Epochs
→ Momentum
→ Regularization Constant
→ Number of branches in Decision tree
→ Number of clusters in a clustering Algorithm like K-means.

WHY DO WE NEED TO SET HYPERPARAMETER?

→ Hyperparameters are important because they directly control the behaviour of the training algorithm and have a significant impact on the performance of the model is being trained.

→ Efficiently search the space of possible hyperparameters.

→ then its easy to manage a large set of experiments for hyperparameter tuning.

HOW TO CHOOSE A HYPERPARAMETER?

STEP-1: Split the data at hand into the training and test subsets.

STEP-2: Repeat optimization loop a fixed number of times or until ~~and~~ a condition is met.

STEP-3: Compare all metric values and choose the hyperparameter set that yeilds the best metric value.

↳ For Decision Tree, lets see the hyperparameter.

* Hyperparameter – Depth

→ As Depth increases – Overfitting
↓ ↕
pure nodes High variance, low Bias

→ As Depth decreases – Underfitting
↕
High Bias, low Variance.

↳ For Random Forest, lets see the hyperparameter.

* Hyperparameters → Depth, Number of Base learners.

→ As Depth increases – Decision Tree gets overfitted byt not Random Forest.

→ As Depth decreases – Decision Tree gets underfitted but not Random Forest

↳ For logistic Regression, lets see the hyperparameter.

$$w^* = \underset{w}{\arg\min} \sum_{i=1}^{n} \log\left(1+\exp\{-y_i \cdot \vec{w}^T \cdot x_i\}\right) + \lambda \|w\|_2^2$$

↓ Cost function

↳ From Optimization theory - Regularization

* Hyperparameter - λ
* parameters - w

→ As λ increases, ↔ Underfitting

↓

High variance, low Variance
Bias

→ As λ decreases (or) equal to zero -

Overfitting - High variance, low Bias.

NOTE:

1. If we use L_2 norm i.e., $\|w\|_2^2$ then it is called as RIDGE REGRESSION.

2. If we use L_1 norm i.e., $\|w\|$ then it is called as LASSO REGRESSION.

3. If we use None, then it takes Overfitting by default.

REGULARIZATION:

The regularization term (or) penalty, imposes a cost on the optimization function for overfitting the function (or) to find an optimal solution.

→ It is any modification one makes to a learning algorithm that is intended to reduce its generalization error but not its training error.

WHAT IS HYPERPARAMETER USED IN REGULARIZATION?

The regularization parameter lambda that we just introduced is another commonly used hyperparameter.

→ When both the parameters and hyperparameters are tuned, the final Performance of the model is evaluated as with a test set, the same way we did before.

→ The regularization is used to increase the magnitude of parameter values in order to reduce overfitting.

→ The larger 'λ' is the less likely it is

that the parameters will be increased in magnitude simply to adjust for small perturbations in the data.

↳ For linear Regression, lets see the hyperparameter.

$$w^* = \arg\min \sum_{i=0}^{n} (y_i - \{w^T \cdot x_i\})^2 + \lambda \text{ term}$$

λ term → α_1, α_2, ElasticNet

$$\lambda_1 * \|w\|_1^2 + \lambda_2 * \|w\|_2^2$$

* Hyperparameter - λ
* parameter - w

→ As λ increases - underfitting

↕

High Bias, low Variance

→ As λ decreases (or) equal to zero, -

Overfitting - High variance, low Bias.

ELASTIC NET:

It is a popular type of regularized linear regression that combines two popular penalities, specifically α_1 & α_2 Norm functions.

→ This is an extension of Linear Regression that adds regularization penalities to the loss function during training.

↳ For SVM, lets see the hyperparameter.

Soft Margin - $w^* = \arg\min \left\{ \frac{2}{\|w\|} + c * \frac{1}{n} \sum_{i=1}^{n} \zeta_i \right\}$

* Hyperparameter - C
* parameter - w.

→ As C increases - Overfitting
↓
High variance, low Bias

→ As C decreases (or) equal to zero,-

Underfitting - High Bias, low Variance.

NOTE:

SVM ~~par~~ hyperparameter is opposite of logistic Regression. i.e.,

$$C = 1/\lambda$$

Dual Form -

$$\arg\max \sum_{i=0}^{n} \alpha_i - \frac{1}{2} \sum_{i=0}^{n} \sum_{j=0}^{n} \alpha_i \alpha_j y_i y_j \cdot K(x_i^T \cdot x_j)$$

$$s.t \quad 0 \le \alpha_i \le C \quad \& \quad \sum \alpha_i y_i = 0$$

→ If $K(x_i, x_j) = x_i \cdot x_j$ ↔ linear SVM

$$\downarrow$$

$$x_i^T \cdot x_j$$

For RBF - $K(x_i, x_j) = \exp\left\{ \frac{-\|x_i - x_j\|^2}{2\sigma^2} \right\}$

$$= \exp\{-\gamma \|x_i - x_j\|\}$$

where $\gamma = \frac{1}{2\sigma^2}$

If it is a linear Regression, no need of using γ