JINDAL
PAGE NO.
DATE :

WEEK 7 MLT

Introduction to Binary Classification

LECTURE 1

Binary Classification :

$\{x_1, \ldots, x_n\} \quad x_i \in \mathbb{R}^d$

$\{y_1, \ldots, y_n\} \quad y_i \in \{0,1\} \;/\; \{-1,+1\}$

Goal : $h : \mathbb{R}^d \rightarrow \{0,1\}$

Loss / Error :

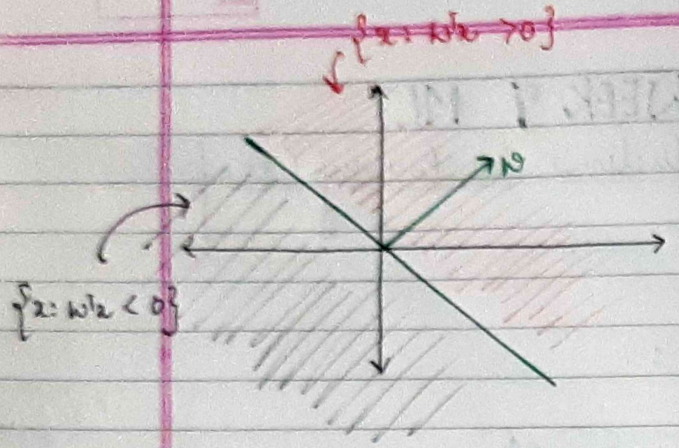
$$\text{Loss}(h) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\left(h(x_i) \neq y_i\right)$$

where $\mathbb{1}(z) = \begin{cases} 1, & \text{if } z \text{ is true} \\ 0, & \text{otherwise} \end{cases}$

If I restrict my space of functions to just the linear functions, then my goal would be to

$$\min_{h \in H_{linear}} \sum_{i=1}^{n} \mathbb{1}\left(h(x_i) \neq y_i\right)$$

$$H_{linear} = \left\{ h_w : h_w(x) = \text{sign}\left(w^T x\right) \right\} \qquad \text{sign}(z) = \begin{cases} 1, & \text{if } z > 0 \\ 0, & \text{otherwise} \end{cases}$$

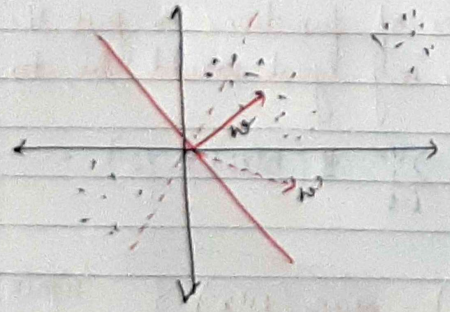


How can we solve it then?
→ it is an NP hard problem !!

- Can we use linear regression to solve classification problem ?

$\{(x_1, y_1), \ldots, (x_n, y_n)\}$ → [Lin. Reg.] → $w \in \mathbb{R}^d$ → $h_w : \mathbb{R}^d \to \{0,1\}$

→ Regression is sensitive to location of the data points & not just the 'side' on which the data lies wrt separator.



LECTURE 2 "K- nearest Neighbours"

The no. of data points that we have to store is independent of the value of k.

Simple Algorithms for Classification

- Given a test point $x_{test} \in \mathbb{R}^d$, find the closest point x^* to x_{test} in the training set.
- Predict $y_{test} = y^*$

ISSUE → This algorithm can get affected by outliers.
FIX → Ask more neighbours

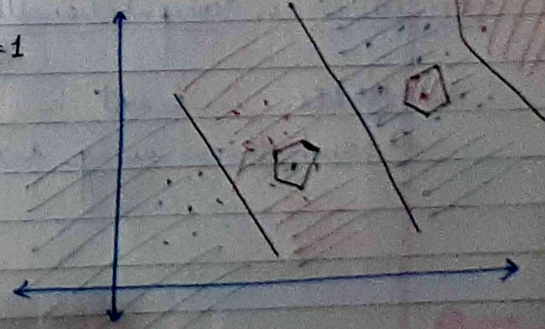
K- NN (K- nearest neighbours)

- Given x_{test}, find the k- closest points in the training set → $(x_1^*, x_2^*, \ldots, x_k^*)$
- Predict y_{test} = majority $(y_1^*, y_2^*, \ldots, y_k^*)$

How does K affect our algorithm ?

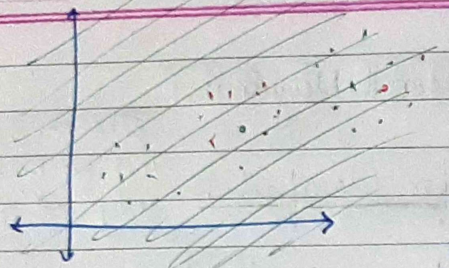
DECISION BOUNDARY :

(i) k=1



It is sensitive to outliers !!

(ii) $k = n$

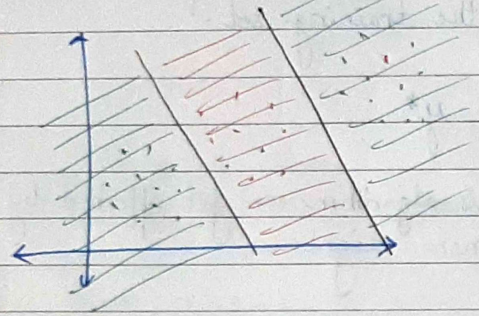


Every point is predicted $Y = -1$ (green)

TOO SMOOTH !!

(iii) k^*

a good choice



Choosing K :
- → can treat as a hyperparameter
- → smaller the k, complicated the decision boundary
- → soln : cross validate for k

Issues with K-NN :

- Choosing a distance fn
- Prediction is 'computationally expensive'
- No MODEL is learnt
  - can't throw away data after 'learning'

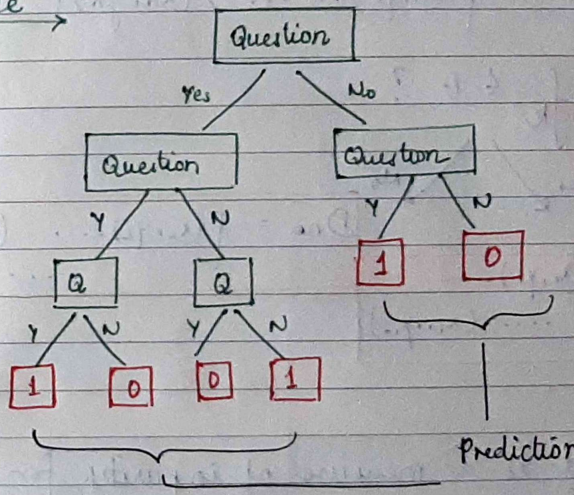
LECTURE 3 Introduction to Decision Tree

Decision Trees :

Input : Dataset $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ $\forall i\ x_i \in \mathbb{R}^d$, $y_i \in \{+1, 0\}$

Output : Decision Tree

Decision Tree →



Prediction →

Given x_{test}, traverse through the tree to reach a leaf node.

Y_{test} = value in leaf node

QUESTION → A question is a (feature, value) pair.

Eg. height ≤ 180 cm ?
(f_3) (θ)

How to measure "goodness" of a question ?

Dataset D = $\{(x_1, y_1) \ldots\ldots, (x_n, y_n)\}$

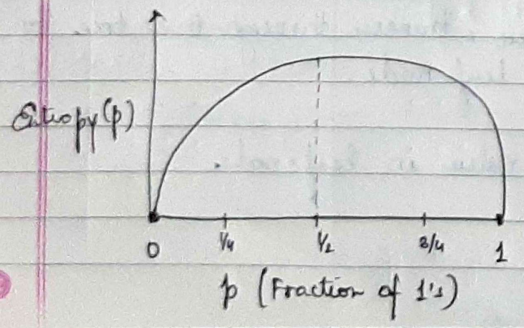
$f_k \leq \theta$?

Yes → $D_{yes} = \{(x_1, y_1) \ldots\ldots (x_{10}, y_{10})\}$

No → $D_{no} = \{(x_2, y_2) \ldots\ldots (x_4, y_4)\}$

- We need is a measure of impurity for a set of labels $\{y_1, \ldots, y_n\}$

"Measure of IMPURITY for a set of labels"



One such function is :

Entropy($\{y_1, \ldots, y_n\}$) = Entropy (p)

⇒ ENTROPY (p) = $-(p \log p + (1-p) \log(1-p))$

[Convention: $\log 0 = 0$]

Now we have some way to measure how impure a bunch of Labels are.

GOAL : given a question how good is this question

D → $f_k \leq \theta$? Entropy (D)

Y → D_{yes}, Entropy (D_{yes})

N → D_{no}, Entropy (D_{no})

If Q_1 is better than Q_2 then,
$IG(Q_1) > IG(Q_2)$

Information Gain (feature, value) =

Entropy (D) − [Y · Entropy (D_{yes}) + (1−Y) Entropy (D_{no})]

weighted average of D_{yes} & D_{no}

$Y = \frac{|D_{yes}|}{|D|}$

weight of data points that fell on each side from original impurity

- Minimum depth of a decision tree for it to have bounded decision regions in $\mathbb{R}^2 \rightarrow 4$
- The impurity of a node is influenced by its ancestor.

JINDAL
PAGE NO.
DATE :

LECTURE 4 Decision Tree Algorithm

- Discretize each feature [min, max] range
- Pick the question that has the largest information gain
- Repeat the procedure for D_{yes}, D_{no}

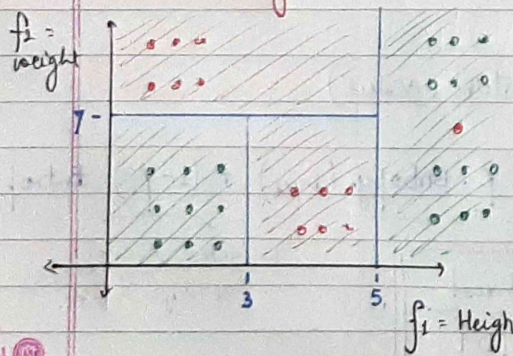
Points that we need to remember while building the tree:

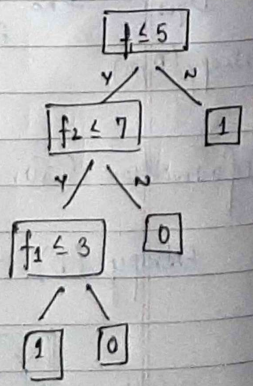
→ We can stop growing a tree if a node becomes "SUFFICIENTLY" pure

→ DEPTH of the tree is a hyperparameter

There are alternate measures for "goodness" of a question
→ GINI INDEX

Decision Boundary :





- Every question is dividing our space by a line which is parallel to one of the axes.
→ A decision tree is cutting a region into rectangles.

LECTURE 5 Generative & Discriminative Models

TYPES OF MODELING

Distribution

D over $X \times Y \longrightarrow \{+1, -1\}$ (feature space)

↳ unknown but fixed

What does this mean?
→ Every data point that you are seeing in your training set is drawn acc. to this distribution.

TRAINING SET : $\{x_1, \ldots, x_n\}$, $\{y_1, \ldots, y_n\}$ $\quad (x_i, y_i) \sim D$ (link between training & test)

Think of this distribution as some kind of a probabilistic mechanism that generates our data.

- What about the connection b/w training & test set?
→ the connection is via this underlying distribution D. The test set is also going to come from the same underlying distribution.

Two different ways to do the modelling :

Classification
- Generative Model
- Discriminative Model

Generative	Discriminative
• $P(x, y)$ modelling the joint distribution of x & y	• $P(y/x)$ modelling the prob. of y given x
• If you are modelling feature generation, this is GM.	• if you are only caring about how to discriminate whether a given feature is in label y equal to $+1$ or -1, then it is DM.
	Eg. K-nearest neighbours (KNN) Decision Trees ↳ $P(Y=1/x) = 1$ if decision tree for x says 1. $P(Y=1/x) = 1$ if majority of neighbours say 1, $= 0$ otherwise