Supervised learning
- Discrete output (Classification)
- Real-valued (Regression).

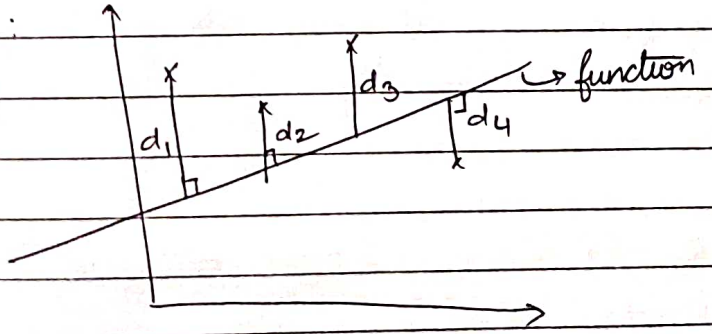
$X_1, X_2, \cdots X_D$ → features
$Y_1, Y_2, \cdots Y_K$ → outputs
$x_i \in R^D$.

Classification
- Binary
- Multi-class.

Regression

Single-output regression : $f(x_i) \rightarrow y$.
linear — $y = mx + c$.

eg :



$\min \sum_{i=1}^{N} d_i^2$.

d_i → euclidean distance.

Minkowski's Distance (L^p norm)

$$\sqrt[p]{|x_2 - x_1|^p + |y_2 - y_1|^q}$$

$p = 1 \rightarrow$ L' norm
$p = 2 \rightarrow$ euclidean distance

to account for error → minimise $\sum (\hat{y}_i - y_i)^2$ → residual error

$\hat{y}_i$ predicted, y_i actual

Here,

loss function,

$$\min_{m,c} \sum_{i=1}^{N} \underbrace{(mx_i + c}_{\hat{y}_i} - \underbrace{y_i)^2}_{y_i}$$

f(m, c) = min { sum of squared residual errors }.

Multivariate linear Regression,

$$\hat{y} = \alpha_1 x_1 + \alpha_2 x_2 + \cdots \alpha_D x_D + \underbrace{\alpha_0}_{\text{intercept}}$$

Roots → $\frac{\partial J}{\partial \alpha_0} = \frac{\partial J}{\partial \alpha_1} = \cdots \frac{\partial J}{\partial \alpha_D} = 0.$

Correlation

$$\frac{\frac{1}{N}\sum_{i=1}^{N}(a_i-\bar{a})(b_i-\bar{b})}{std(A)\,std(B)} = \frac{\frac{1}{N}\sum(a_i-\bar{a})(b_i-\bar{b})}{\sqrt{\frac{1}{N}\sum(a_i-\bar{a})^2}\sqrt{\frac{1}{N}\sum(b_i-\bar{b})^2}}$$

$$\sigma^2(A) = \frac{1}{N}\sum(a_i-\bar{a})^2$$

$$\sigma(A) = \sqrt{\sigma^2(A)}$$

$$= \frac{\sum(a_i-\bar{a})(b_i-\bar{b})}{\sqrt{\sum(a_i-\bar{a})^2}\sqrt{\sum(b_i-\bar{b})^2}}$$

* Covariance $= \frac{1}{N}\sum(a_i-\bar{a})(b_i-\bar{b})$.

* Correlation $\in [-1, 1]$.
Covariance $\in (-\infty, \infty)$.

Linear Regression for Classification

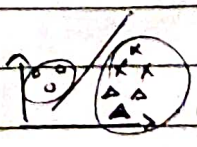
$$\hat{y}_i = \alpha_0 + \alpha_1 x_1^i + \alpha_2 x_2^i + \cdots + \alpha_D x_D^i$$

$i \to i^{th}$ row.

$$\hat{y}_i \in (-\infty, \infty).$$

Q Why do we take {0, 1} & not {1, 2} for binary classification?
It results in giving more weightage to higher class labels in order to minimize difference w/ separator.

How to use Linear Regression w/ Multiclass Dataset

1) One vs one.  (we get ${}^nC_2$ combo)
2) One vs all  (we get n combos)
3) Error correcting output code.

One vs One

Take 1 1 & find max. of all eqⁿ solⁿs.

One vs All

Find max ~~(illegible)~~ w/ most.

One v One has more data loss than One v All

* In 1v1, only $\frac{N}{k} \cdot 2$ samples are taken & $N - \frac{2N}{k}$ are lost
(v. high if K is high)

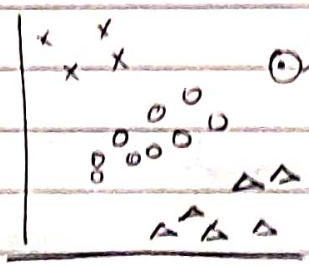
but in 1 v All, all are taken.

But, if many categories,

1 v All has way more 0 than 1 a.k.a errors.

So, both are inaccurate if categories are a lot.

Also,



will result in ○ being masked by △ & X.

Sigmoid fⁿ → $\frac{1}{1+e^{-x}}$ always returns ∈ [0, 1].

Error Correcting Output Codes

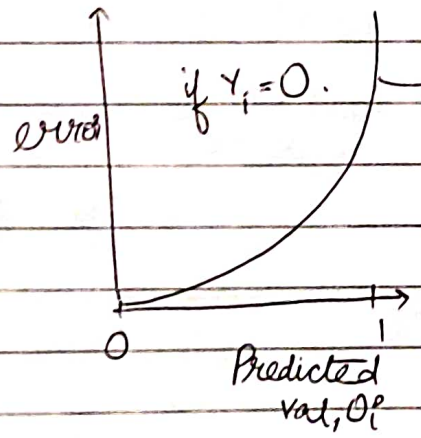
(Check later).

Logistic Regression (Binary Classifier). [here, it is better to use |x|].

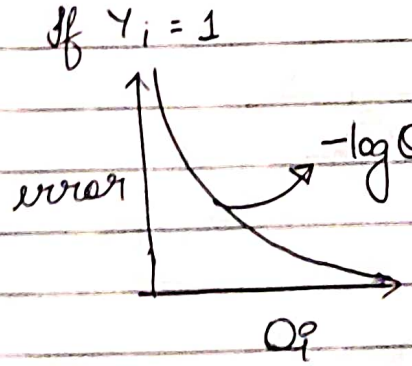
$$\hat{y}_i = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n.$$
$$= \sum \alpha_j x_j \quad \text{if } x_0 = 1.$$

To convert vals b/w $(-\infty, \infty)$, we can use sigmoid to obtain $[0, 1]$ range.

Since sq. loss f^n won't return sq. f^n for e^{-x}, we have a non convex f^n & need a convex optimisation method.



* If $f(x)$ is increasing, $\log f(x)$ is increasing.



Term 1 $\rightarrow Y_i = 1$ | Term 2 $\rightarrow Y_i = 0$

$$\sum [-y_i \log O_i] + [-(1-y_i)\log(1-O_i)]$$

$$\text{error} = \sum^{k} [-y_i \log O_i - (1-y_i)\log(1-O_i)].$$

Now,

$$O_i = \frac{1}{1+e^{-\hat{y}_i}}, \quad \hat{y}_i = \alpha_0 + \alpha_1 x_1 + \cdots + \alpha_n x_n$$

Here,

$$\alpha_i^{t+1} = \alpha_i^t - \lambda \times \frac{\partial J}{\partial \alpha_i}$$

Consider term ①.

$$\sum \frac{\partial}{\partial \alpha_i} (-y_i \log O_i) = -y_i \frac{\partial}{\partial \alpha_i} \sum \log O_i$$

$$= -y_i \frac{\partial}{\partial \alpha_i} \sum \log \left(\frac{1}{1+e^{-\hat{y}_i}} \right)$$

$$= -y_i \times (1+e^{-\hat{y}_i}) \times \frac{\partial}{\partial \alpha} \frac{1}{1+e^{-\hat{y}_i}}$$

$\hookrightarrow 1/O_i$

$$= +y_i \sum \frac{1}{O_i} \times \left(\frac{1}{e^{-\hat{y}_i}+1} \right)^2 \times \frac{\partial}{\partial \alpha_i} (e^{-\hat{y}_i}+1)$$

$$= +y_i \sum \left[\frac{1}{O_i} \times \frac{1}{(1+e^{-\hat{y}_i})^2} \times e^{-\hat{y}_i} \times \frac{\partial}{\partial \alpha_i} \hat{y}_i \right]$$

$\hookrightarrow O_i^2$

$$\text{Term ①} = y_i \sum \left[O_i \times e^{-\hat{y}_i} \frac{\partial}{\partial \alpha_i} \hat{y}_i \right]$$

$\hookrightarrow x_i$

$1/o_i - 1$

$1/o_i = e^{-\hat{y}_i} + 1$

Term ①:

$$\sum \frac{y_i}{o_i} \cdot o_i^2 \times \left(\frac{1}{o_i} - 1\right) x_i$$

$$= \sum y_i o_i \left(\frac{1}{o_i} - 1\right) x_i = \sum y_i o_i \left(\frac{1 - o_i}{o_i}\right) x_i$$

$$\text{term ①} = \sum y_i x_i (1 - o_i)$$

On simplifying & adding term ②, we get:

$$\star\star \quad \alpha^{t+1} = \alpha^t - l \times \sum [o_i - y_i] x_i$$

Compare this w/ gradient descent for linear regression, $l \rightarrow$ step length.

$$\alpha^{t+1} = \alpha^t - l \times \sum (\hat{y}_i - y_i) x_i$$

* Here, decision boundary does NOT change.

o_i can also be represented by

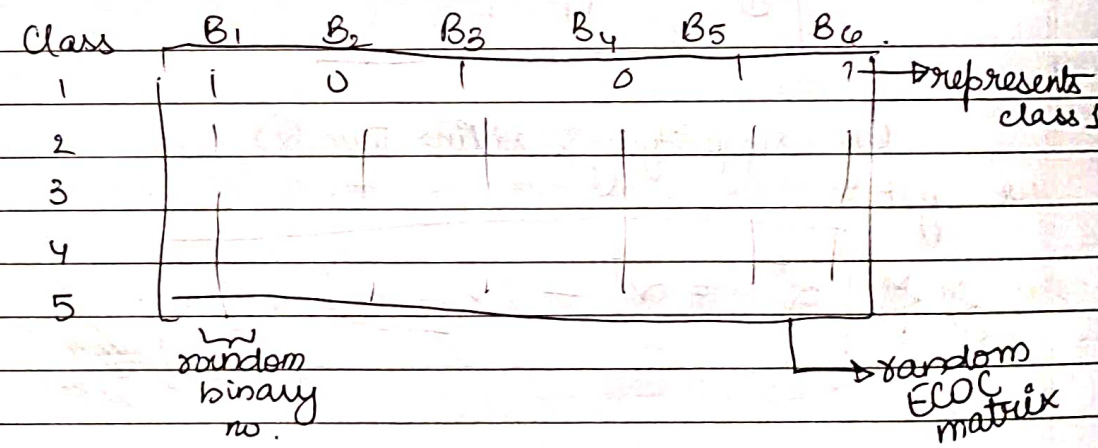
$$P(y=1 \mid \underbrace{\text{for eg.}}\ \{x_1 = 3 \cdots x_d = 5\}).$$

Error Correcting Output Codes (ECOC).

Sample length is chosen b/w K & ${}^{K}C_2$.

can go beyond this but it is not that useful.

$$K \leq l \leq {}^{K}C_2.$$



To verify that the binary labels are diff:
do XOR & count how many 1s are there.

Horizontal Separability ← The strings should be far apart by a length of $\frac{l}{d}$ minimum

Vertical Separability:
No. of ~~strings~~ 0s & 1s in a col. should approx be equal.

To classify, generate outcome string & do XOR, and then put it in the class w/ least no of 1s.

Overfitting:

If 2 models M_1 & M_2,

When,

training error

$Tr(M_1) < tr(M_2)$

But test error,

$test(M_1) > test(M_2)$.

M_1 is overfitting.

Regularisations (to avoid overfitting).

loss fn, $\left[\sum -y_i \log o_i - (1-y_i)\log(1-o_i)\right] + \lambda \times$ Penalty term

same regularisers as linear.

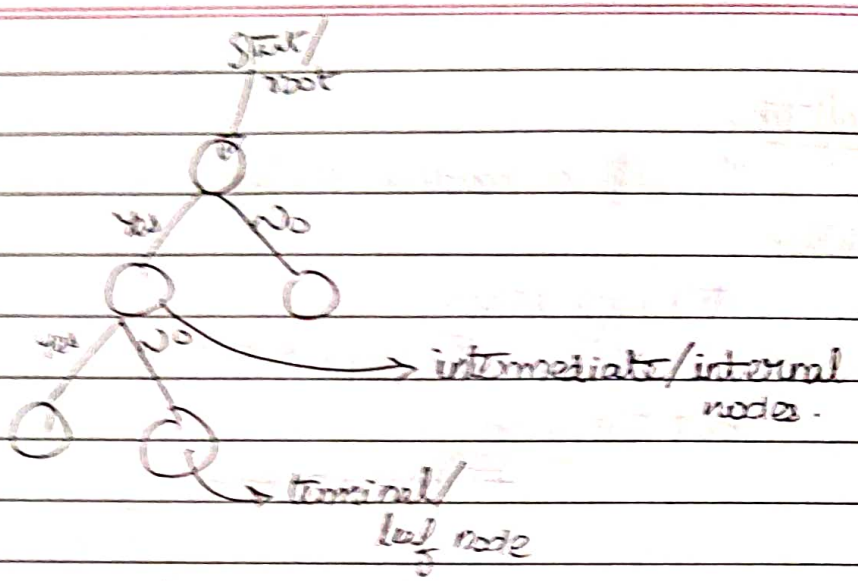
$L1 \rightarrow \sum_{j=1}^{D} |\alpha_j|$

$L2 \rightarrow \sum_{j=1}^{D} \alpha_j^2$.

Elastic net $\rightarrow$ λ_1 L1 Penalty + λ_2 × L2 Penalty

$\lambda_2 = 1 - \lambda_1$

Decision Tree



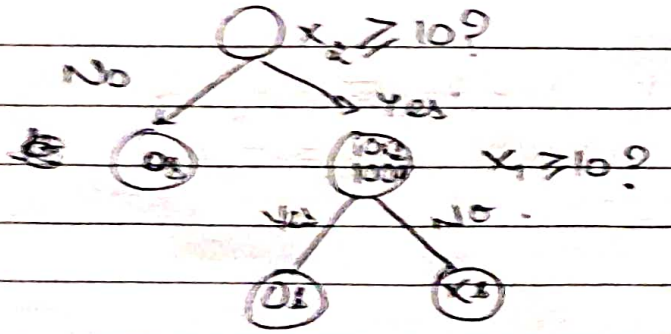
Root & internal have conditions

Leaf nodes have no conditions, they make a decision.

- It can directly be used for multi-class datasets.

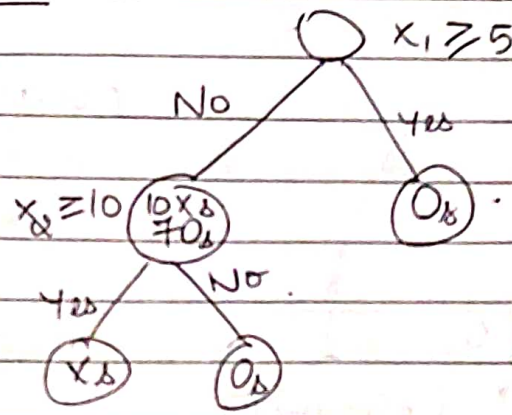
One separator line is impossible. So here there can be two lines to help us. This.

Tree:



This type of decision boundary is called piecewise linear

alternative tree:



Clearly, 2nd one has a more pure root partition than 1st tree. (10:7 vs 10:10). So we would prefer this.

How we test this → algorithm.

How to decide Purity

eg: | x Xs |
| y Os |

Purest → One is 0

Most impure → x = y.

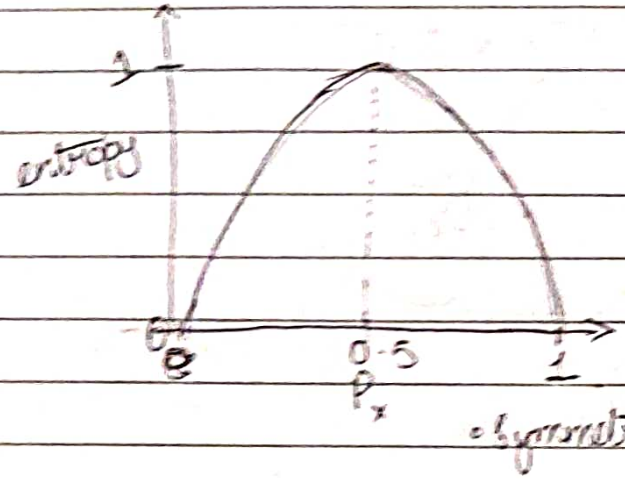
Shannon's entropy → $\sum_{i=1}^{c} -p_i \log p_i$ c → no. of classes. p → probability

For eg, 5 Xs & 5 Os → $-\frac{1}{2}\log_2\frac{1}{2} \times 2 = -\log_2 2^{-1} = +1$.

(Most impure case).

10 Xs & Os → $-\frac{1}{2}\log_2 1 = 0$.

this entropy should be minimised.



(only for 2 classes)

classes

- for 3 classes, it is > 1 in max. case.

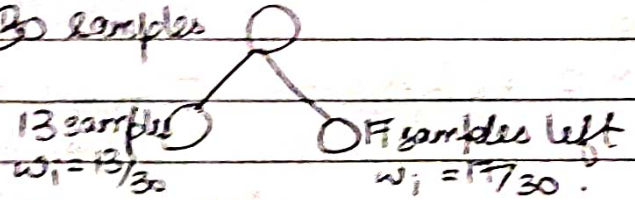
Entropy would increase w/ no. of classes.

Gain in purity/ = entropy of parent − $\sum_{\text{weighted}}$ entropy for each child

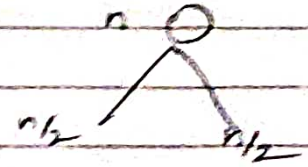
Information gain = " − $w_i \sum$ entropy of each child.

w_i = no. of samples the child uses / no. of samples the parent uses

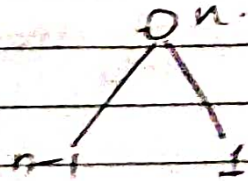
eg: 30 samples



No of partitions for categorical features = no. of categories.
To avoid overfitting & having partitions w/ (n−1) & (1) samples, we take threshold as mean.



is better than



We can also take ranges of diff ages (buckets) & create multiple children if each bucket has equal no of samples.

* Categorical features typically have higher entropy reduction (info. gain) than numerical features.

Other metrics (learn for exams)

- Gini impurity index
- Gain ratio.
- Cost sensitive ratio

Greedy Algorithm → only checks next choice at every step

Algorithm

* Recursively partitions data into pure, labelled partitions

eg dataset (in slides)

$$\text{entropy}_{\text{root}} = \left(-\frac{9}{14}\log\frac{9}{14} - \frac{5}{14}\log\frac{5}{14}\right) \} \text{ for buys computer}$$

$$= 0.9403.$$

Consider all 4 features

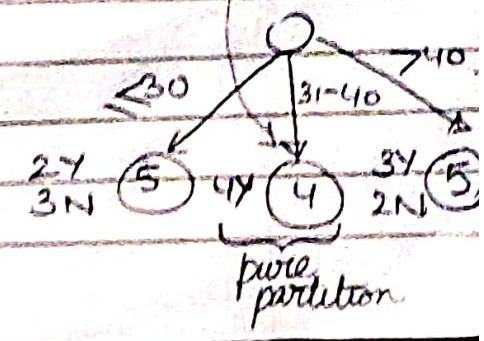
for age ? low ✗ medium high

$$\text{gain} = \text{"} = \frac{5}{14} \times \text{entropy} - \frac{4}{14} \text{ entropy} - \frac{5}{14} \text{ ent.}$$

For age?

$$\text{entropies} = \frac{2}{5}\log\frac{2}{5} - \frac{3}{5}\log\frac{3}{5} - \frac{3}{5}\log\frac{3}{5} - \frac{2}{5}\log\frac{2}{5} = 0$$

$$= \text{"} = -2\left[\frac{2}{5}\log\frac{2}{5} + \frac{3}{5}\log\frac{3}{5}\right].$$



Info. gain = parent entropy $- \left[\frac{5}{14} \left[\frac{2}{5} \varepsilon_{C_1} \right] + \frac{5}{14} \varepsilon_{C_3} + 0 \right]$

$= 1 - \frac{5}{7} \varepsilon_{C_1/C_2}$

$= 0.9403 - \frac{5}{7} \times 0.9710$

$= 0.2467$

Similarly, for student,

$IG = 0.1519$

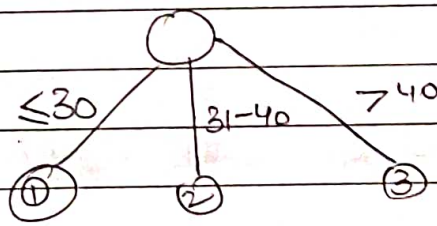
income,

$IG = 0.0292$

C.R,

$IG = 0.0481$

We choose the highest I.G & partition by age.

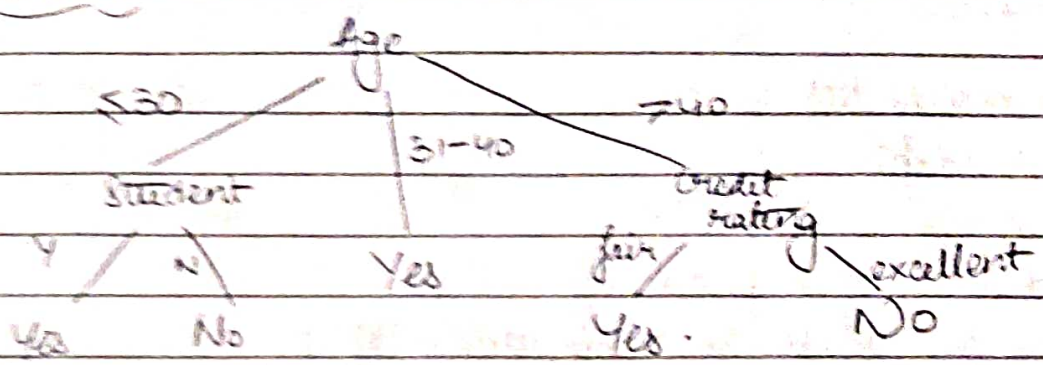


Look at node ①.

5 samples — 3Y 2N

IG = 0.9710 — (for each feature)

final tree



* If we increase tree depth, it is likely to overfit. & disadvantage

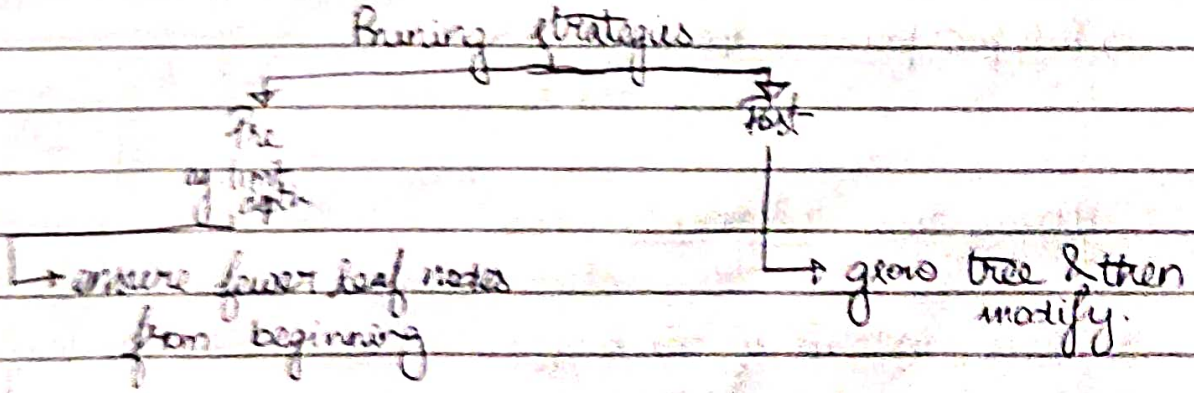
Randomised Trees

At every step take the top 'k' features & then choose best

this reduces chance of overfitting.

Other ways to avoid overfitting

1) Limit tree depth
2) Use pruning strategies & other approaches.



Pre-pruning

1) Randomised tree
2) Limit depth
3) Limit leaf nodes.
4) Put threshold on probability
5) If v. few samples in node, don't split.
6) If splitting results in v. few in 1 node & many in other, don't split } min-sampled leaf.

Post pruning

1) Merge redundant subtrees (if merging increases accuracy).

Subtree error pruning.

↓

BUT, most computer intensive

↑

most effective strategy b/c we don't assume any parameters from the start.

2) Rule post pruning (not v. effective).

we try to combine if-else conditions.

Decision trees for Regression

Since val. can't be same for regression here, instead of checking all in same class for leaf nodes, we check if they are close in value.

The metric we use to compute how diverse they are is standard deviation.

~~Support Vector Machines~~ (SVM)

⇒ Minimise $\sum_{i=1}^{N} \sum_{j=1}^{N} (x_i - x_j)^2$, $i < j$, $i \neq j$.

$= N \sum (x_i - \bar{x})^2$. This lowers it from $O(n^2)$ to $O(n)$.

⇒ Minimise $\Sigma (x_i - \bar{x})^2$

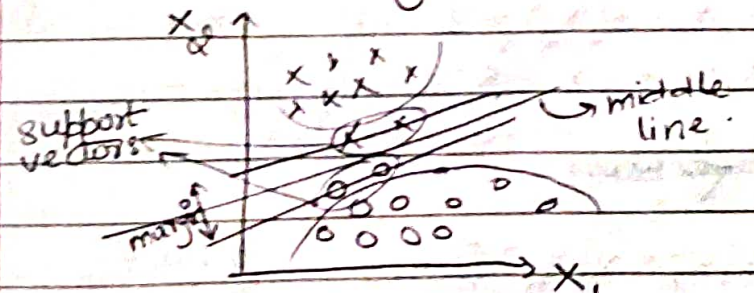
⇒ Minimise $\Sigma \frac{(x_i - \bar{x})^2}{N}$ (since N is const.)

⇒ Minimise σ^2 (variance).

* We branch by picking the feature which gives you the least variance on branching.
(we can also choose other metrics)

~~Support Vector Machines~~ (SVM) (v. math)

Unlike logistic regression, we give more imp to pts closer to line. To draw boundary,



We choose parallel lines combo. based on maximum margin.
SVM is also called max. margin classifier.

The ll line eqⁿ,

$$\alpha_1 X_1 + \alpha_2 X_2 = \text{constant}$$

both lines will have same eqⁿ w/ diff. constants.

For mathematical simplicity,

let const 1 = − const 2.

⟹ middle line = 0.

Eqⁿ's:

$$\left(\frac{\alpha_1}{K_1}\right) X_1 + \frac{\alpha_2 X_2}{K_2} = 1 \quad ; \left(\frac{\alpha_1}{K_2}\right) X_1 + \left(\frac{\alpha_2}{K_2}\right) X_2 = -1.$$

new α's. Middle line ⟶ $\alpha_1 X_1 + \alpha_2 X_2 = 0$

$$\rightarrow \alpha_1 X_1 + \alpha_2 X_2 = 1 \; ; \; \alpha_1 X_1 + \alpha_2 X_2 = -1.$$

dist. b/w the lines, $d = \frac{|c_1 - c_2|}{\sqrt{\alpha_1^2 + \alpha_2^2}} = \frac{2}{\sqrt{\alpha_1^2 + \alpha_2^2}}$

⟹

Maximise d

s.t $\alpha_1 X_1 + \alpha_2 X_2 \geq 1$ if $y_i = 1$

$\alpha_1 X_1 + \alpha_2 X_2 \leq -1$ if $y_i = -1$.

becomes

$$y_i(\alpha_1 X_1 + \alpha_2 X_2) \geq 1$$ ∴ hard SVM.

We solve this using Lagrange multipliers.

Soft SVM,

$\geq 1 - \varepsilon$ ↳ error

(allow some misclassifications to max. margin).

We maximise $\frac{2}{\sqrt{\alpha_1^2 + \alpha_2^2}} = \frac{2}{\|w\|_2}$.

is same as maximising | minimise

primal eqn $\left\{ \frac{2}{\|w\|_2^2} \right.$ | $\left. \frac{\|w\|_2^2}{2} \right\}$ dual eqn.

No. of constraints = N (no. of samples).

No. of parameters → N + d (d → no. of features)

Lagrange Multipliers

min $f(x)$ s.t $g(x)$

becomes

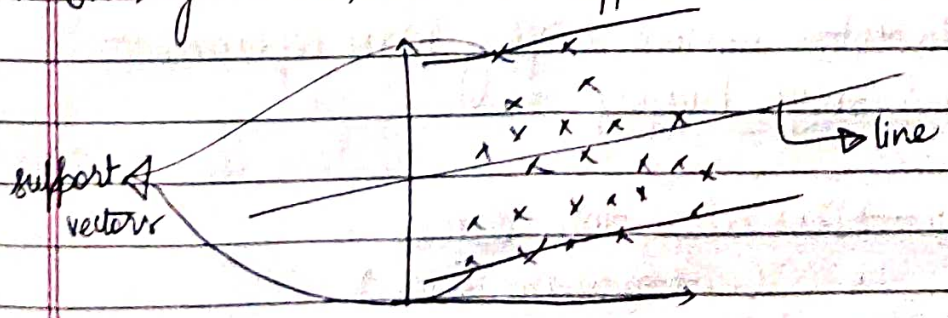
$f(x) + \lambda g(x)$

where $\lambda > 0$.

* If $g(x)$ has $\geq$ then $\lambda > 0$

$g(x)$ has $\leq$ then $\lambda < 0$.

* In regression, we take support vectors as OUTERMOST pts.



Ensemble Learning

Using multiple models based on sample. then choose a final ans. based on all results

eg → by majority (not the best way).

↓

By weighted avg (weight ∝ 1/error of that model).

• Model

Consider models

$M_1, M_2, \ldots, M_k$.

Each model should be ~~at least~~ mediocre.

These models are called weak learners. (We choose weak learners to increase diversity).

* If we give each model a lot of samples, then no. of combos are less so almost same models ⇒ no diversity

If we give too few samples then insufficient to train.

∴ We usually choose around 70% of total.

① To increase diversity, random sampling WITH replacement (called instance/sample bagging).

Weak learner → Does better than chance

(eg → for binary, ↓ 50%).

② Another way to increase diversity is by giving a subset of FEATURES to each model (resulting in diff results).

[feature bagging].

↳ ⇒ WITHOUT replacement.

If every model is a decision tree, this is called RANDOM FOREST.

Q. Why does a collection of weak learners do better? (proof).

Ans: Consider $M_1, M_2, \ldots, M_{21}$.

For misclassification, at least 11 models should misclassify.

i.e $P(11$ models misclassify$)$

Assume, accuracy of each model is 0.7.

$\Rightarrow$ error = 0.3

Assume that the predictions follow a binomial distribution.

$$P(\text{misclassification}) = {}^{21}C_{11} \cdot 0.7^{10} \cdot 0.3^{11} + {}^{21}C_{12} \cdot 0.7^{9} \cdot 0.3^{12} \cdots$$

success = misclassify

$$P[\text{exactly 11 misclassify}] = {}^{21}C_{11} \cdot 0.7^{10} \cdot 0.3^{11}$$
$$= 0.017$$

$$P[12 \text{ misclassify}] = 6.3 \times 10^{-3} = 0.0063$$

∴ this probability keeps decreasing.

∴ $P[$at least 11 misclassify$]$ is v.v low.

∴ $P[$at least 11 are CORRECT$]$ is high.

When error is 0.6 (worse than chance)

$P(\text{1 misclassify}) = 0.135$

i.e ~~not~~ fairly high

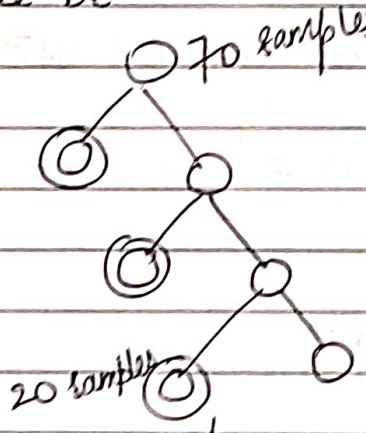
⇒ Result has high chance of being wrong.

Q. Why is majority voting not optimal?

Ans: Consider M_1, M_2, M_3, M_4, all are decision trees.

& $Y_i \in \{1, 2, 3, 4\}$. (multi class dataset).

Let tree be



No. Samples: 15, 2, 0, 3

Class 1, 2, 3, 4.

⇒ Probabilities: $\frac{15}{20}, \frac{2}{20}, \frac{0}{20}, \frac{3}{20}$

Label distribution

Look at this for all models

eg:

	1st	2nd	3rd	4th
M_1	1	4	0	3
M_2	1	4	3	2
M_3	4	3	2	1
M_4	3	4	0	1

If simple majority vote, we only look at 1st column & discard 2nd, 3rd & 4th.

∴

Output = Class 1

Instead, we should do:

1st col gets K-1 votes
↳ no. of classes.

:

last col gets 0 votes

eg, here,

1st → 3 votes
2nd → 2 votes
3rd → 1 vote
4th → 0 votes

} this method is called Borda Count.

Class 1 → 6 votes
2 → 3 votes
3 → 6 votes
4 → 9 votes
∴ Class 4 is output.

⭐ Q. When you can't use Borda Count?
We assume all models are equally strong predictors

So what to do if they aren't equal?

Bagging
↳ Instance (w/ replacement)
↳ Feature (w/o replacement).

Instance

It chooses N samples randomly & gives high imp. to more repeating.

flaw
↳ It is chance-based so imp is arbitrary.

Boosting

We should focus on hard to classify samples. (hard to learn).

eg: let models be

$M_1, M_2, \dots, M_K$

w/training sets

$T_1, T_2, \dots, T_K$

Consider 5 samples in dataset:

$\{x_1, x_2, x_3, x_4, x_5\}$

(multi-dimensional → d dimensions/features)

First, build T_1 the same way as random forest (instance bagging)

eg → take 3 samples.

let α_i^j → importance

P(sample i gets chosen for j^{th} Training set).

$\alpha_1' = 1/5, \alpha_2' = \alpha_3' = \alpha_4' = \alpha_5' = 4/5$

eg → $T_1 = \{x_4, x_1, x_2\}$

Use this to make a decision tree M_1.

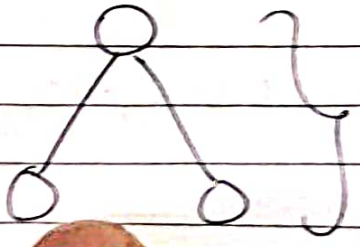
Now, test M_1 on all 5 samples.

If x_i is misclassified by M_1, give it more imp.

* In boosting, for decision tree, we ONLY go till ONE level, ie:

a) No overfitting

b) No strong learners.



Decision Stump (only 1 level trees).

eg: let x_4 be correctly predicted & x_3, x_5 be mis classified.

* Every decision stump we build should be a weak learner i.e error(m_j) < 0.5 ∀ j.

Now, we update α_i^j.

α_3^{j+1} & α_5^{j+1} increase

others decrease.

How to increase/decrease?

[Ada Boost algorithm].

Adaptive — by Freund & Schapire

formula:

$$\alpha_i^{j+1} = \alpha_i^j \times e^{\text{quantity}}$$

look up AdaBoost MH

If quantity > 0, weight increases

quantity < 0, weight decreases.

i.e for > 0, $e^{+\text{quantity}}$, < 0 $e^{-\text{quantity}}$.

$$\text{Quantity} = \ln\left(\frac{1 - \text{Error}_j}{\text{Error}_j}\right)$$

eg: $\alpha_3^2 = \frac{1}{5} \times \frac{0.6}{0.4}$ | $\alpha_5^2 = \frac{1}{5} \times \frac{0.6}{0.4} = 0.3$

$\alpha_{1,2,4} = \frac{1}{5} \times \frac{0.4}{0.6} = 0.13$

Now, we want $\sum_{i=1}^{n} \alpha_i^j = 1$, but after updating this is not always true.

So we normalise it:

$$\alpha_i^{j+1} = \frac{\text{Updated } \alpha_i^{j+1}}{\sum \alpha_i^{j+1}}$$

How to take care of probabilities?

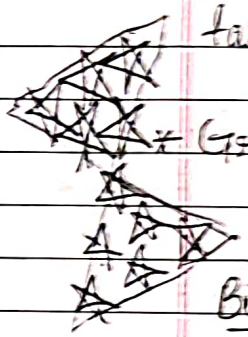
Consider 5 buckets -

B_1 B_2 B_3 B_4 B_5

$[0, \alpha^1)$ $[\alpha^1, \alpha^1+\alpha^2)$ $[0.3, 0.5)$ $[0.5, 0.8)$ $[0.85, 1]$

$\alpha^1 = 0.1$ $\alpha^2 = 0.2$

* Choose a random no. b/w 0 & 1 & check w/ bucket it falls into. Choose that sample & use it in T_2



* Gradient Boosting Algorithm -

Bias v/s Variance

Variation in output w.r.t degree of polynomial.

w.r.t test set

where

Variation = deviations in predictions of models

Bias = how far you are from the actual curve. (true hypothesis)

low bias ⇒ high degree.

⟶ low bias ⇒ high variance } trade off.

& vice versa.

Ideally, low bias & low variance, but this is not possible so we settle in the middle.

eg: Linear regression — high bias.

Logistic " for classification — high bias

KNN for 1 neighbour — high bias

↳ smaller the K, lower the bias.

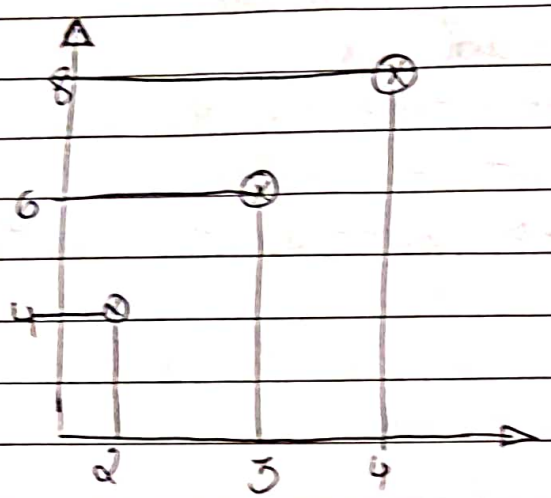
Decision tree w/ large depth — low bias.

↳ higher the depth, lower the bias.

~~Next~~

NUMERICALS

Q1.



x	y
2	4
3	6
4	8

test sample : $x_p = 5.5$

$$\alpha_0 = \bar{y} - \alpha_1 \bar{x}$$

↳2 intuitively, β is 2α

$$\alpha_0 = 6 - 2(3)$$
$$= 0.$$

$$\Rightarrow y = 2x.$$

(OR)

$$(X^T X)^{-1} X^T y = \begin{bmatrix} \alpha_0 \\ \alpha_1 \end{bmatrix}$$

$x = \{x_0, x_1\}$ $\quad y = \begin{bmatrix} 4 \\ 6 \\ 8 \end{bmatrix}$

$$X = \begin{bmatrix} 1 & 2 \\ 1 & 3 \\ 1 & 4 \end{bmatrix}_{3\times 2} \qquad X^T = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 4 \end{bmatrix}_{2\times 3}$$

$$X^T X = \begin{bmatrix} 3 & 9 \\ 9 & 29 \end{bmatrix}^{-1} = \begin{bmatrix} 3 & 9 \\ 9 & 29 \end{bmatrix}$$

$$(XX^T)^{-1} = \begin{bmatrix} 18 & -9 \\ -3 & 6 \end{bmatrix}^T \times \frac{1}{(54-6)}$$

$$(XX^T)^{-1} = \begin{bmatrix} 29 & -9 \\ -9 & 3 \end{bmatrix}^T \times \frac{1}{87-81} = \frac{1}{6}$$

$$\begin{bmatrix} 29 & -9 \\ -9 & +3 \end{bmatrix}_{2\times 2} \times \begin{bmatrix} 1 & 1 & 1 \\ 2 & 3 & 4 \end{bmatrix}_{2\times 3}$$

$$= \begin{bmatrix} \end{bmatrix}$$

Weight Update Eqⁿ for Gradient Descent

$$\alpha_i^{t+1} = \alpha_i^t - l \times \sum_{i=1}^{N} (\hat{Y}_i - Y_i) X_i$$

let $\alpha_0 \to 1$ $\alpha_1 \to 3$. (assume)] → Batch gradient

$$\alpha_0^1 = \alpha_0^0 - \underbrace{l}_{\text{take as } 0.01} \times \frac{1}{\underbrace{N}_{3}} \sum_{i=1}^{3} (\hat{Y}_i - Y_i) \times \underbrace{X_0}_{1}$$

$\hat{Y}_i = 1 + 3X$.

$$= 1 - 0.01 \times \frac{1}{3} \times \underbrace{[(7-4)}_{3} + \underbrace{(10-6)}_{4} + \underbrace{(13-8)}_{5}].$$

$$= 1 - \frac{0.01}{3}[12] = 1 - 0.04 = 0.96$$

$$\alpha_1^1 = 3 - 0.1 \times \frac{1}{3} \times [\underbrace{(7-4)2}_{\to 6} + \underbrace{(10-6)3}_{\to 12} + \underbrace{(13-8)4}_{\to 20}]$$

$$= 3 - 0.1 \times \frac{1}{3}[38] = 3 - 1.266667.$$

α 3.00
−1.26
1.74

$$= 1.74.$$

∴ keep repeating this

For stochastic, remove $\frac{1}{N}$ & Σ & do for 1 eq.

For mini batch, batch size $< N$

Stacking

Take a model on training set.
Assume it predicts $\hat{y}$.

$$f(x_1, x_2, \cdots, x_D) \longrightarrow \hat{y}$$

then, train another one on $\hat{y}$

$$f(\hat{y}) \rightarrow y$$

* In 2nd model, we can also use it in addition to original features $x_1, \cdots, x_D$.

eg: Decision tree, $f(x_1, \cdots x_D) \rightarrow y$ $\hat{y}_{DT}$
log regression, $f(y, \hat{y}_{DT}) \rightarrow y$ $\hat{y}_{LR}$.
SVM, $f(y, \hat{y}_{DT}, \hat{y}_{LR}) \rightarrow y$

Blending

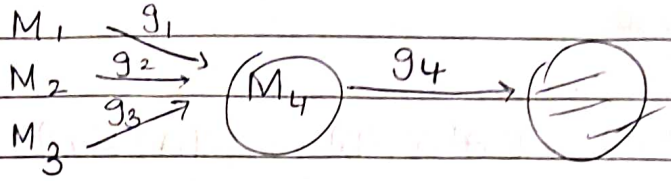
Instead of simply merging, divide training set into partitions
Each model uses the ~~same~~ data ONLY from 1 partition as training sets.

Hierarchical Mixture of Experts

eg: LR $\rightarrow M_1$ (g_1)
DT $\rightarrow M_2$ (g_2)
SVM $\rightarrow M_3$ (g_3)
$\Sigma g_i M_i$ where $\Sigma g_i = 1$.

$$\hat{y}_i = \Sigma g_i m_i.$$

$$\min \Sigma (\hat{y}_i - y_i)^2.$$



We can derive g_i by gradient descent.

$$g_j^{t+1} = g_j^t - l \times \sum_{i=1}^{N} [\hat{y}_i - y_i] m_j \quad \text{prediction}$$

g_i depends on how imp. the model is.

g_i (if we use mean sq. error),

$$\text{imp. of } M_i = \frac{\Sigma MSE_i}{MSE_{@1}}.$$

$$g_i \propto \frac{\Sigma MSE_j}{MSE_i}$$

we need to normalise this so $\Sigma g_i = 1$.

$\therefore$ Final weight, $w_i = \frac{g_i}{\Sigma g_i}$.

$$\therefore \boxed{w_i = \frac{g_i}{\sum_j g_j} \quad \text{where} \quad g_i = \frac{\Sigma MSE_j}{MSE_i}}$$

& $\hat{y}_i = \Sigma w_i M_i$.

Only for classification

Baye's theorem

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Naive Baye's theorem

eg.

$$P(\text{glucose}=m, \text{age}=31\text{-}40, \text{weight}=H \mid Y=1) \times P(Y=1)$$
$$P(m, 31\text{-}40, H)$$

(likelihood; prior probability; evidence)

posterior $= P(Y=1 \mid m, 31\text{-}40, H)$.

But if this sample doesn't exist in training set, $P=0$

To avoid this,

Laplace Correction

Give it a v. small probability $< \frac{1}{N}$.

Naive :

We assume features are all independent.

So we write

$$P(m, 31\text{-}40, H \mid Y=1) = P(m|Y=1) \times P(31\text{-}40|Y=1) \times P(H|Y=1).$$

[For numeric features, we assume normal distribution]

Consider a numerical feature (eg: BMI).

$$P(M|Y=1) \times P(31\text{-}40|Y=1) \times P(H|Y=1) \times P(35.6 \mid Y=1)$$

how to find this?

take mean & std. deviation of training set

use, $P = \frac{1}{\sqrt{2\pi}\sigma} \times e^{-(x_i-\mu)^2/2\sigma^2}$ } Gaussian naive Bayes. & compute probability.

$\text{Evidence} = \sum_{j=1}^{\text{no of classes}} P(M, 31\text{-}40, H, 35\text{-}6 \mid Y = j) \times P(Y = j)$.

(alternative form).

EVALUATION METRICS

Regression Metrics

1) Mean squared error, $MSE = \frac{1}{N} \Sigma (\hat{y}_i - y_i)^2$.

2) Sum of squared errors, $= N \times MSE$.

3) Mean absolute errors, $\frac{1}{N} \Sigma |\hat{y}_i - y_i|$.

4) R square (co-efficient of determination). Used for goodness of the fit.

How much the f^n can adapt

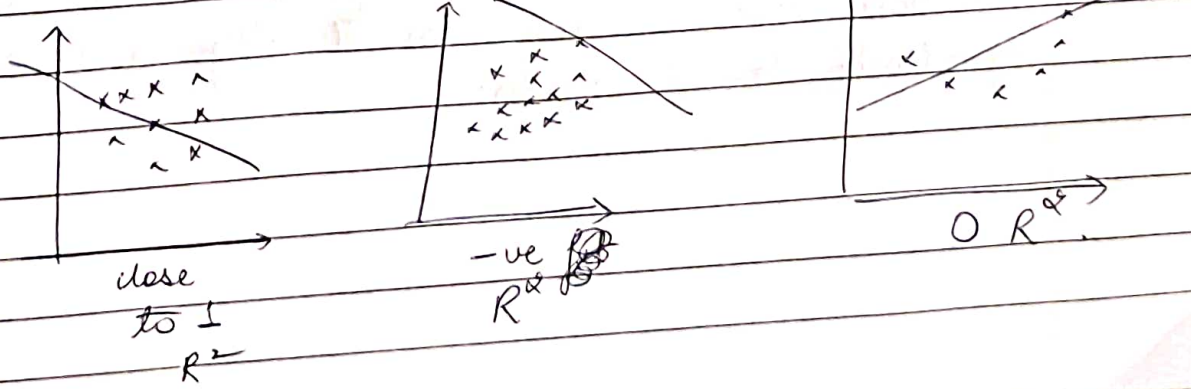
It exists b/w a given range.

It is used to compare diff. degree of polynomials.

-ve → it hasn't learned so far

0 → not a good fit

1 → v. good fit.



$$R^2 \text{ error} = 1 - \left[\frac{\sum_{i=1}^{N} (\hat{Y}_i - Y_i)^2}{\sum (\bar{Y}_i - \bar{Y})^2} \right]$$

sum of squared errors/residuals

total sum of squares

If $\hat{Y}_i = \bar{Y}$ i.e only α_0 i.e underfitting

$R^2 = 0$

If we predict perfectly,

$R^2 = 1$ (since predicted = actual)

If sq. error is v.v large

$R^2 < 0$.

* R^2 is ONLY for training set, NOT for test set.

* R^2 is the ONLY metric where higher is better.

Classification Metrics

Indicator variable $[\hat{Y}_i == Y_i]$.

[if equal, 1 else 0]

① Accuracy, $= \frac{\sum_{i}^{N} [\hat{Y}_i == Y_i]}{N}$

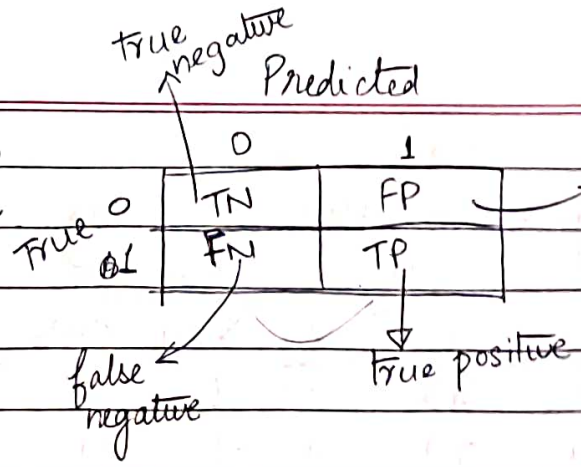
Error = 1 − accuracy.

We don't prefer accuracy when data is skewed.

eg: 99.9% healthy & 0.1% cancerous

We only use accuracy when all categories have almost equal proportion of samples.

(2) Confusion Matrix



Here, (a) accuracy $= \frac{TP+TN}{TP+TN+FP+FN}$

(b) precision, $= \frac{TP}{TP+FP}$

(c) recall, $= \frac{TP}{TP+FN}$.

In case of skewed data,
try to MAX. TP / MIN. FP, based on the situation & requirement.

(3) Binary cross entropy loss (for non-skewed data)

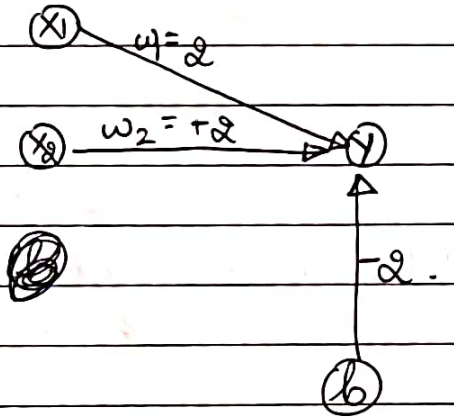
$$= \sum_{i=0}^{\text{samples}} -y_i^0 \log o_i - (1-y_i^0) \log(1-o_i^0)$$

$$\Rightarrow \sum_{\substack{j \text{ over} \\ \text{classes}}} -y_i^j \log o_i^j$$

Q. Design AND gate using Hebbian learning.

inputs →	x_1	x_2	bias b	y	Outputs Δw_1	Δw_2	Δb	w_1 new	w_2 new	b new
S1	1	1	0	1	1	1	1	1	1	1
	1	-1	0	-1	-1	+1	-1	0	2	0
	-1	1	0	-1	+1	-1	-1	1	+1	-1
	-1	-1	0	-1	1	1	-1	2	2	-2

final weights

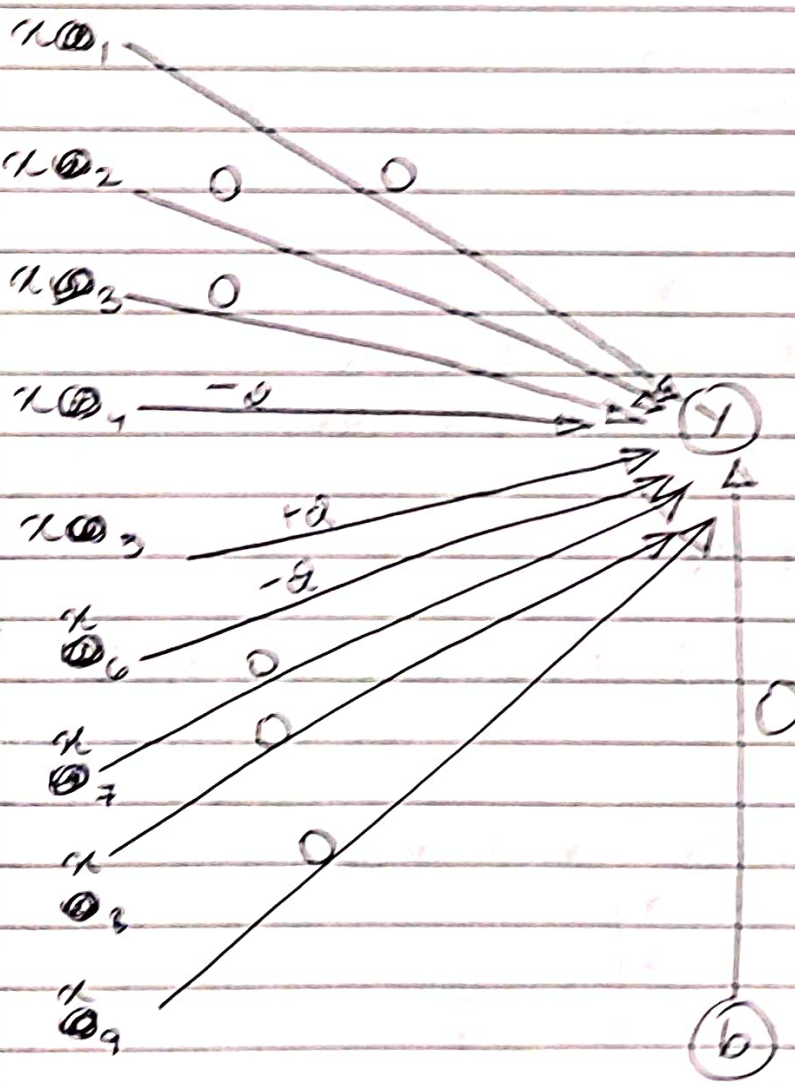


Q.

Letter	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	y
I	+1	+1	+1	-1	+1	-1	+1	-1	+1	+1
O	+1	+1	+1	-1	-1	-1	+1	+1	+1	-1

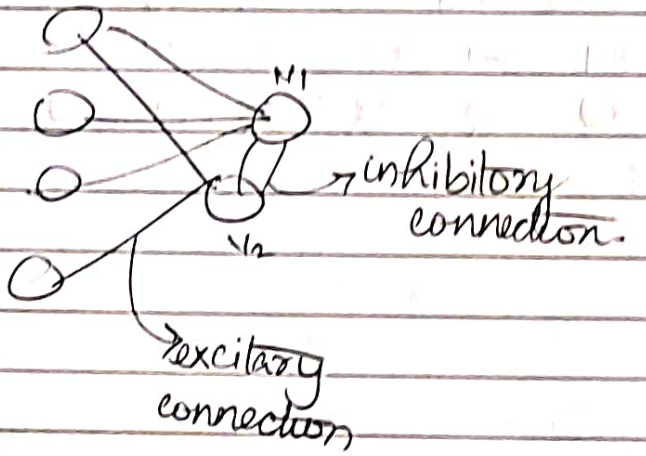
Output

	w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	Bias
I	+1	+1	+1	-1	+1	-1	-1	-1	-1	1
O										



Competitive learning

* Winner-takes-all.
* Only winner's weights updated



$$v_k = \sum x_j w_{kj}$$

Then,

$$\Delta w_{kj} = \begin{cases} \eta(x_j - w_{kj}) & \text{, if } k \text{ wins} \\ 0 & \text{, if } k \text{ loses.} \end{cases}$$

eg: $\eta = 0.5$ y_1, y_2.

Inputs → [0 0 1 1], [1 0 0 0], [0 1 1 0], [0 0 0 1].

$\sum w_{kj} = 1$.

Initialise the weights b/w 0 to 1.

let $w = \begin{bmatrix} 0.2 & 0.9 \\ 0.4 & 0.7 \\ 0.6 & 0.5 \\ 0.8 & 0.3 \end{bmatrix}$ ← input rows.

↑ outputs

$V_1 = [0\ 0\ 1\ 1] \times 0.2 + [1\ 0\ 0\ 0] \times 0.4 + [0\ 1\ 1\ 0] \times 0.6 + [0\ 0\ 0\ 1] \times 0.8$

$= [0\ 0\ 0.2\ 0.2] + [0.4\ 0\ 0\ 0] + [0\ 0.6\ 0.6\ 0] + [0\ 0\ 0\ 0.8]$

$V_1 = [0.4\ \ 0.6\ \ 1.6\ \ 1.0]$

Pattern 1

$V_1 = 0 \times 0.2 + 0 \times 0.4 + 1 \times 0.6 + 1 \times 0.8$

$= 1.4$

$V_2 = 0 \times 0.9 + 0 \times 0.7 + 1 \times 0.5 + 1 \times 0.3$

$= 0.8$

V_1 wins

Pattern 2

$V_1 = 0.2 \quad V_2 = 0.9$

V_2 wins

Pattern 3:

$V_1 = 1.0 \quad V_2 = 1.2$

V_2 wins

Pattern 4

$V_1 = 0.8 \quad V_2 = 0.3$

V_1 wins

PATTERN 1

V_1 wins.

$\Delta w_{11} = 0.5 \times (0 - 0.2) = -0.1$

$\Delta w_{12} = 0.5 \times (0 - 0.4) = -0.2$

$\Delta w_{13} = 0.5 \times (1 - 0.6) = +0.2$

$\Delta w_{14} = 0.5 \times (1 - 0.8) = +0.1$

$\Rightarrow w_{11} = +0.1 \quad w_{12} = +0.2 \quad w_{13} = +0.8 \quad w_{14} = +0.9$

Since node 2 loses, there is no change in weight.

Pattern 2

V_2 wins

$\Delta W_{21} = 0.5(1-0.9) = 0.05$ | $\Delta W_{22} = 0.5\times(0-0.7) = -0.35$ | $\Delta W_{23} = 0.5\times(0-0.5) = -0.25$ | $\Delta W_{24} = 0.5(0-0.$ $= -0.15$

$W_{21} = 0.95$

Pattern 5:

V_2 wins.

$W_{22} = +0.35$

$\Delta W_{21} = 0.5(0-0.9) = -0.45$ | $\Delta W_{24} = 0.5\times(1-0.7) = 0.15$

$W_{23} = +0.25$

$W_{24} = +0.15$

Since node 1 lost, no change.

New weight matrix:

$$W = \begin{bmatrix} 0.1 & 0.95 \\ 0.2 & 0.35 \\ 0.8 & 0.25 \\ 0.9 & 0.15 \end{bmatrix}$$

Pattern 3:

V_2 wins.

$\Delta W_{21} = 0.5\times(0-0.95) = -0.475$ | $\Delta W_{22} = 0.5(1-0.35) = +0.325$ | $\Delta W_{23} = 0.5(1-0.25) = +0.375$ | $\Delta W_{24} = 0.5(0-0.15) = -0.075$

Pattern 4

V_1 wins.

$\Delta w_{11} = 0.5(0-0.1) = -0.05$ | $\Delta w_{12} = 0.5(0-0.2) = -0.10$ | $\Delta w_{13} = 0.5(1-0.8) = +0.10$ | $\Delta w_{14} = 0.5(1-0.9) = +0.05$

$w_{11} = 0.05$ $w_{12} = 0.$ $w_{13} = 0.35$ $w_{14} =$

Final weight matrix

$$w = \begin{bmatrix} 0.05 & 0.95 \\ 0.10 & 0.35 \\ 0.10 & 0.25 \\ 0.05 & 0.15 \end{bmatrix}$$

New weights after pattern 2

$$w = \begin{bmatrix} 0.1 & 0.95 \\ 0.2 & 0.35 \\ 0.8 & 0.25 \\ 0.9 & 0.15 \end{bmatrix}$$

For pattern 3,

$V_1 = 0 + 1 \times 0.2 + 1 \times 0.8 + 0 = 1.0$

$V_2 = 0 + 0.35 + 0.25 + 0 = 0.60$

V_1 wins

$\Delta w_{11} = 0.5(0-0.1) = -0.05$ | $\Delta w_{12} = 0.5(1-0.2) = 0.40$ | $\Delta w_{13} = 0.5(1-0.8) = +0.10$ | $\Delta w_{14} = 0.5(0-0.9) = -0.45$

$$W = \begin{bmatrix} 0.05 & 0.95 \\ 0.6 & 0.35 \\ 0.9 & 0.25 \\ 0.45 & 0.15 \end{bmatrix}.$$

Pattern 4

$v_1 = 0.45 \quad v_2 = 0.15$

v_1 wins.

$$\Delta w_{11} = 0.5(-0.05) = -0.025 \quad \Delta w_{12} = 0.5(-0.6) = -0.3 \quad \Delta w_{13} = 0.5(-0.9) = -0.45 \quad \Delta w_{14} = 0.5(0.55) = 0.275$$

$$W = \begin{bmatrix} 0.025 & 0.95 \\ 0.3 & 0.35 \\ 0.45 & 0.25 \\ 0.725 & 0.15 \end{bmatrix}$$