df.sort_values(by=['col1', 'col2'])

02/09/2020

df.mean(axis=0)

AI and ML

Linear Regression

gini = 1 − (probability of yes)² − (probability of no)²

weighted avg = $\frac{14}{14+15}(0.395) + \frac{15}{14+15}(0-$

entropy → $-\sum_i P\left(\frac{i}{t}\right)\log\left(\frac{i}{t}\right)$

$-\frac{2}{6}\log\frac{2}{6} - \frac{4}{6}\log\frac{4}{6}$

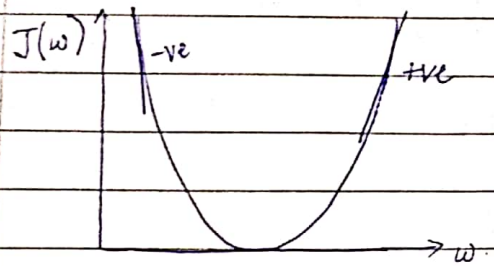
Representation of Hypothesis —

$$h(x) = w_0 + w_1 x_1$$

$$= w_0 x_0 + w_1 x_1$$

$$h(x) = w_0 + w_1 x_1 + w_2 x_2 \cdots$$

$$= \sum_{j=0}^{n} w_j x_j \quad \text{where } x_0 = 1$$



$$w_j = w_j - \alpha \frac{\partial}{\partial w_j}(J(w))$$

+ve
-ve

IG = entropy(P) − (weighted sum of entropy of children nodes)

Benefit of Square Loss — Single Global Minima (Convex Function)

ML Life Cycle

Data → Preprocessing Data

↓

Split Data (Train, test, validate)

↓

Decide problem type (Regression, Classification)

↓

Decide loss type

↓

Decide Hypothesis (Model Structure)

↓

Learning Algo (Gradient Descent/Ascent)

↓

Training $w_j = w_j - \alpha \sum_{i=0}^{n} (h_w(x_i) - y_i) \cdot x_i$

↓

$h_w(x)$ ← testing set — satisfied? — No

Scanned with CamScanner

$$J(w) = \frac{1}{2m} \sum_{i=1}^{m} \left(h(x)^{(i)} - y^{(i)}\right)^2 \leftarrow \text{squared loss.}$$

$$h(x) = w_0 x_0 + w_1 x_1 \ldots\ldots w_m x_m.$$

$$w_j = w_j - \alpha \frac{\partial}{\partial w_j}\left(J(w)\right)$$

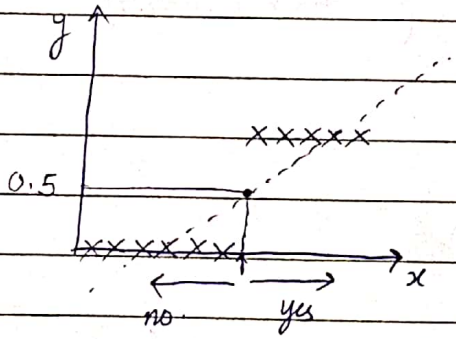
$$J(w) = \frac{1}{2}\left(h_w(x)^{(i)} - y^{(i)}\right)^2.$$

$$\frac{\partial J(w)}{\partial w_j} = 2 \times \frac{1}{2}\left(h_w(x)^{(i)} - y^{(i)}\right) \cdot \frac{\partial}{\partial w_j}\left(w_0 x_0 + w_1 x_1 \ldots\ldots\right)$$

$$= \left(h_w(x) - y\right) \cdot x_j$$

$$w_j = w_j - \alpha \left(h_w(x) - y\right) x_j$$

Classification –



$$h_w(x) = g(w^T x) = \frac{1}{1 + e^{-(w^T x)}}$$

$$h_w(x) \in [0, 1].$$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \end{bmatrix}$$

$$w^T x = w_0 x_0 + w_1 x_1 \ldots\ldots$$

likelihood.

$$P(y=1 \mid x; w) = h_w(x)$$

$$P(y=0 \mid x; w) = 1 - h_w(x)$$

$$y = \{0, 1\} \quad h_w(x) \in [0, 1]$$

only one term shall be used at once, the other becomes 0

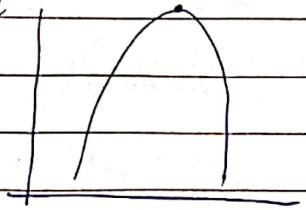
$$g(w) = \prod_{i=0}^{m} P\left(y^{(i)} \mid x^{(i)}, w\right) = \prod_{i=0}^{m} \left(\left(h_w(x^{(i)})^{y^{(i)}} + \left(1 - h_w(x)^{(i)}\right)^{(1-y^{(i)})}\right.\right.$$

$$\log g(w) = \log\left[\prod_{i=0}^{m} \left(h_w x^{(i)}\right)^{y^{(i)}} + \left(1 - h_w x^{(i)}\right)^{(1-y^{(i)})}\right]$$

Scanned with CamScanner

$$= \sum_{i=0}^{n} y^{(i)} \log h_w x^{(i)} + (1-y^{(i)}) \log (1 - h_w x^{(i)})$$

Gradient Ascent



$$w_j = w_j + \alpha \frac{\partial (l(w))}{\partial w}$$

$$+\alpha \sum_{i=0}^{m} (y^{(i)} - h_w x^{(i)}) x_j^{(i)}$$

Softmax - Multiple classes ← logits

$$\begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ C_4 \end{bmatrix} = W^T x = \begin{bmatrix} 10 \\ 8 \\ 3 \\ -1 \end{bmatrix} \rightarrow \begin{bmatrix} e^{10} \\ e^{8} \\ e^{3} \\ e^{-1} \end{bmatrix} \rightarrow \begin{bmatrix} \frac{e^{10}}{\text{mean}} \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix} \} y$$

Binary Classification → $\begin{bmatrix} C_1 \\ C_2 \end{bmatrix}$ } 0s and 1s

values b/w 0 and 1 add up to 1.

Threshold > 0.5 → 1 else 0

$$y = f(x_1, x_2)$$

$$\begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}_3^0 \quad J(y) = \begin{bmatrix} \frac{\partial y}{\partial x_1} & \frac{\partial y}{\partial x_2} \end{bmatrix}$$

$$u, v = (x_1, x_2)$$

$$J(u,v) = \begin{bmatrix} \frac{\partial u}{\partial x_1} & \frac{\partial u}{\partial x_2} \\ \frac{\partial v}{\partial x_1} & \frac{\partial v}{\partial x_2} \end{bmatrix}$$

Hessian Matrix.

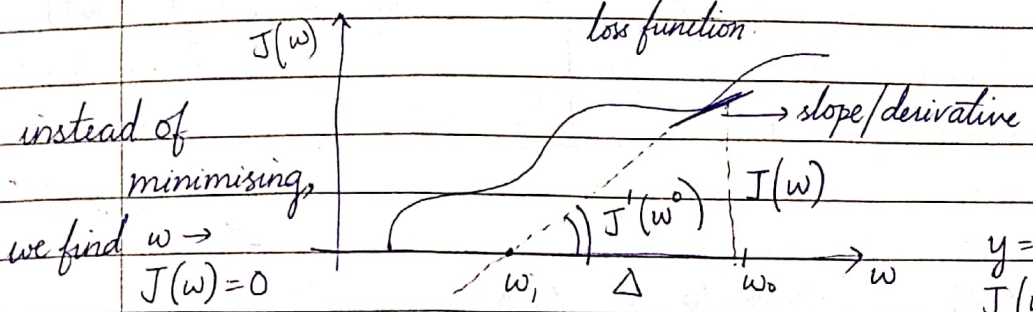
$$\frac{\partial^2 f}{\partial w^2} \qquad \frac{\partial y}{\partial (x_1, x_2)} = \begin{bmatrix} \frac{\partial y}{\partial x_1}, & \frac{\partial y}{\partial x_2} \end{bmatrix}$$

$$\begin{bmatrix} \frac{\partial^2 y}{\partial x_1 \partial x_1} & \frac{\partial^2 y}{\partial x_1 \partial x_2} \\ \frac{\partial^2 y}{\partial x_2 \partial x_1} & \frac{\partial^2 y}{\partial x_2 \partial x_2} \end{bmatrix} \rightarrow \text{Hessian Matrix.}$$

Scanned with CamScanner

Newton's Method

Logistic Regression → Want to maximize $l(w)$.



instead of minimising, we find $w \rightarrow J(w) = 0$

$y = mx$

$J(w^\circ) = J'(w^\circ) \cdot \Delta$

$\Delta = \frac{J(w^\circ)}{J'(w^\circ)}$

$w^{(1)} = w^\circ - \Delta$

$= w^\circ - \frac{J(w^\circ)}{J'(w^\circ)}$

$w^{(t+1)} = w^t - H^{-1} \nabla_w l$

$H \rightarrow n+1 \times n+1$ (n features)

H^{-1} → hessian inverse

$\nabla_w l$ → jacobian (vector of derivative)

Newton's method – large size (can't compute efficiently)

as original input $n \times n$.

$R^{n+1 \times n+1}$

Advantage – Quadratic Convergence

↓ in each step

$l(n) = 0.01$

$n+1 = 0.0001$

$n+2 = 0.00000001$

~~Decision Trees~~

Difference B/w Batch Gradient Descent and Stochastic Gradient Descent

→

Scanned with CamScanner

learning rate is hyperparameter.

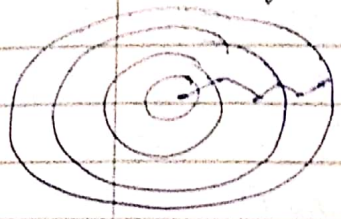
Cost function — (for single training expte)

Loss function — over the entire training set (m) (or mini-batch for mini-batch gradient descent)

→ Vanilla (Batch) G.D — (slower but smooth.)

$$\theta_j = \theta_j - \alpha \cdot \frac{\partial}{\partial \theta_j} J(\theta)$$

$$\frac{1}{m} \sum_{i=1}^{m} (\hat{y}^i - y^i) x_j^i$$



→ Stochastic Gradient D.

for i in range(m):

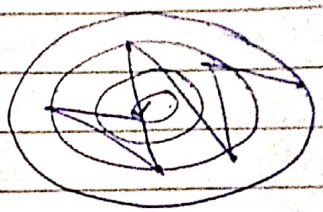
$$\theta_j = \theta_j - \alpha (\hat{y}^{(i)} - y^{(i)}) x_j^i$$

one training example

→ faster but uneven moment (zigzag)

→ noisier
→ more random

iteration of one training example at a time



Mini-

→ Batch gradient descent.

Mini-Batch size 1000.

for i in range (0, m, 1000)

$$\theta_j = \theta_j - \alpha \sum_{i=0+p}^{p+1000} (h_w(x^{(i)}) - y^{(i)}) x_j$$

instead of one sample in SGD → we take n data points in each iteration

Decision Trees :- To predict the class value or the target output by learning decision rules inferred from the prior data (training data)

prioritize which to consider first

For predicting a class label - we start from the root of the tree

→ Tree can have single root.

Scanned with CamScanner

Impurity } becomes our criteria } we need to quantify the impurity

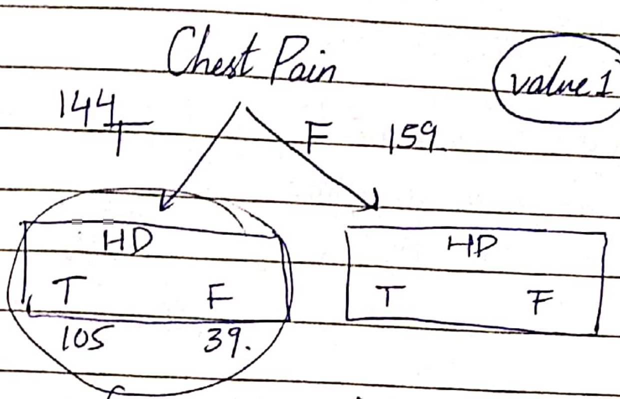
then we say → it's a good node

Information Gain – How important a given attribute of the feature vector is.

Entropy → $-\sum p(i/t)\log_2 p(i/t)$

← measures homogeneity of code

(1.1) → $-\left(\frac{2}{6}\right)\log_2\left(\frac{2}{6}\right) - \left(\frac{4}{6}\right)\log\left(\frac{4}{6}\right)$



value 1

0.0336.

$1 - (\text{probability of yes})^2 - (\text{probability of no})^2$

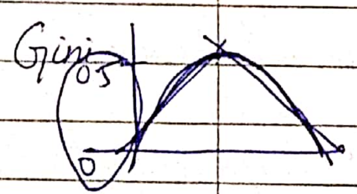
$1 - \left(\frac{105}{105+39}\right)^2 - \left(\frac{39}{105+39}\right)^2$

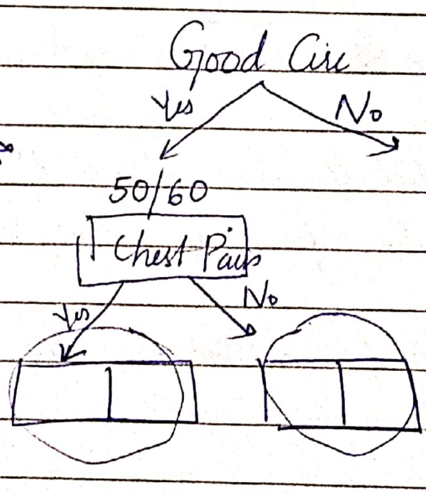
105 + 39 = 144

= 0.395

weighted avg → $\frac{144}{144+159}(0.395) + \frac{159}{144+159}(0.336)$

5, 5





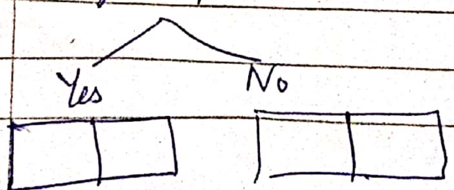
Weight	Heart Disease
220	Y
180	Y
225	Y
190	N
155	N

least value

weight < 205

weight ≤ 167.5

Weight.sort().
take avg of this and next val

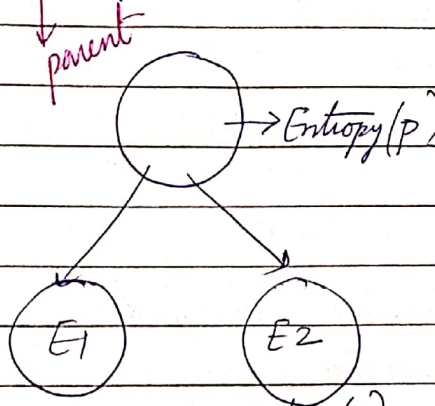


Entropy - Completely pure set → 0
" impure " → 1 b/w 0 and 1

__/__/__

Information Gain :-

Entropy(p) - (weighted sum of entropy of children nodes)



(E1, E2) → weighted sum = IG.

Entropy(p) - IG

Algorithms

CART → Entropy, IG

ID3 → Gini Score
(used for splitting)
(in decision trees)

Regression Metrics

Explained Variance (Similarity Measure)

$$1 - \frac{var(y - \hat{y})}{var(y)}$$

$$MAE = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i| \quad L_1 \text{ norm}$$

$$MSLE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} \left(\log_e(1 + y_i) - \log_e(1 + \hat{y}_i)\right)^2.$$

→ scales down data to a smaller scale

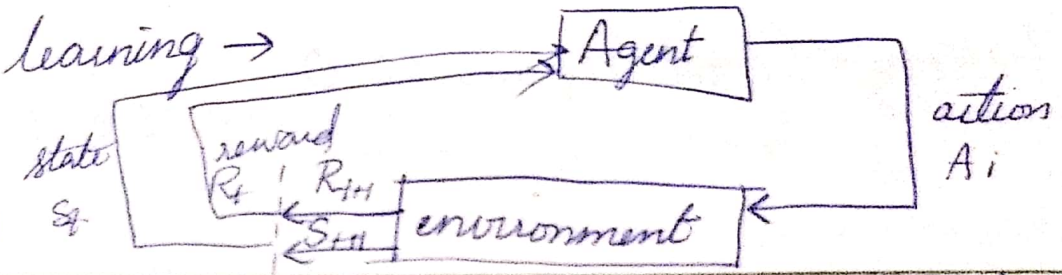
$$\text{Median Absolute Error}(y, \hat{y}) = \text{median}\left((y_1, \hat{y}_1), \ldots (y_n, \hat{y}_n)\right)$$

$$R^2 \text{ score: } \frac{1 - \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \qquad \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

↓ mean.

Scanned with CamScanner

Reinforcement learning →



__/__/____

data = pd.read_csv

Labs Revision

Agent-Environment Interface

Scanned with CamScanner

AI tools → to solve problems like calculator

AI Agents → have intelligence

Unsupervised Learning — y labels not given — tries to make sense out of data by extracting features and patterns on its own

Supervised Learning — y labels given

Search problems
Markov decision process
Adversarial games

Reflex "low level intelligence"

States

Variables → Constraint satisfaction problems
Bayesian networks.

Logic "High Level Intelligence"

(descriptive analysis) Unsupervised (Hindsight)

(predictive analysis) Supervised (Insight)

(prescriptive analysis) Reinforcement (Foresight)

(clustering) ← (regression) classification

value, difficulty →

Astronomical data analysis

Market Segmentation

Organise Computing Clusters

SNA
↓ ↓ Network
Social

Tumor size.
Malignant / Benign

Game Playing
Robot in a maze
Balance Pollin hand.
Credit assignment problem

Scanned with CamScanner

AI Material - Exam Prep. 12th Oct (11:30 - 12:30 pm)

⇒ Regression Metrics :-

(1) Error ← simplicity ↓

(2) Simplicity ← accuracy ↑

→ Mean Absolute Error

$$MAE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|$$

does a poor job when scale of the data is high.

→ Mean Squared Error

$$MSE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2$$

Squaring +ve value, sum won't be zero

" emphasizes larger differences

→ Median Absolute Error.

$$MedAE(y, \hat{y}) = median(|y_1 - \hat{y}_1|, \ldots, |y_n - \hat{y}_n|)$$

↳ does not support multioutput.

Median AE prevents outliers contributing more error to model evaluation.

Scale of prediction to be high, logs scales it down.

→ Mean Squared Logarithmic Error.

$$MSLE(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (\log_e(1+y_i) - \log_e(1+\hat{y}_i))^2$$

when targets have exponential growth → population counts, average sales of commodity over years

Method penalizes an under predicted estimate greater than over predicted estimate.

Scanned with CamScanner

→ Max Error.

$$\text{MaxError}(y, \hat{y}) = \max(|y_i - \hat{y}_i|)$$

aims to find max error made by model for single sample → finding best model that covers all samples.

Similarity Measures.

→ Explained Variance Score $(y, \hat{y})$ Best possible score → 1.0 lower values are worse

$$= 1 - \frac{\text{Var}\{y - \hat{y}\}}{\text{Var}\{y\}}$$

Estimates deviation between prediction and actual values

→ R^2 score, the coeff of determination

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

$$\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$$

$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$$

calculates unadjusted R^2 without correcting for bias in sample variance of y

Error / Dissimilar Measure (less the better)

Accuracy / Similar Measure (more the better)

Classification Metrics

	actual	predicted	
True +ve	1	1	11 → 1
False +ve	0	1	01 → 0
False -ve	1	0	10 → 0
True -ve	0	0	00 → 1

$A\bar{B} + \bar{A}B$ - EXOR.

$AB + \bar{A}\bar{B}$ EXNOR

→ Accuracy $= \frac{TP + TN}{TP + TN + FP + FN}$ } all true values / total

→ Precision $= \frac{TP}{TP + FP}$ ↑ how precise ↑ fraction of predicted +ves that are actually +ve

Scanned with CamScanner

Recall (Sensitivity) = $\frac{TP}{TP + FN}$ ← correct

FN → 0 then recall max

FN → all correct

__/__/____

Total days - 30 (one month)

actual rain - 10

Model - 9 days (5 ✓, 4 ×)

Precision = 5/9 ; Recall = 5/10

Accuracy :- (TP + TN) / total → TP+FP+TN+FN

True tve Rate :- TP / actual yes → FN+TP

(Sensitivity / Recall)

False tve Rate :- FP / actual no → TN+FP

True -ve Rate :- TN / actual no → TN+FP

(Specificity)

Precision :- TP / predicted yes → FP+TP

Misclassified Rate :- FP + FN / Total → TP+FP+TN+FN

	Predicted NO	Predicted YES
Actual NO	TN	FP
Actual YES	FN	TP

→ $F_1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 * (precision * recall)}{precision + recall}$

50% 50% focus

→ F_β score = $\frac{(1+\beta^2) \cdot (precision \cdot recall)}{(\beta^2 \cdot precision + recall)}$

$\beta = 0$ P ↑

$\beta = \infty$ R ↑

- $F_{0.5}$ (beta = 0.5) :- Precision ↑ Recall ↓
- F_1 (beta = 1.0) :- Precision 50% Recall 50%
- F_2 (beta = 2.0) :- Precision ↓ Recall ↑

Scanned with CamScanner

ROC Curve and ROC AUC Score

helps in understanding balance b/w TPR & FPR

__/__/__

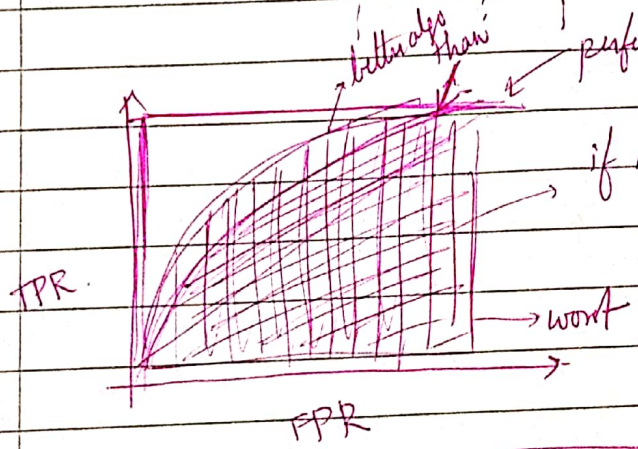
- Thresholds → all unique prediction probabilities in 'descending order'
- FPR = (FP / (FP+TN)) } for each threshold
- TPR = (TP / (TP+FN)) }

Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s.

eg: $th_1 > 0.3$ | 1 else 0

$th_2 > 0.4$ | 1 else 0

	fps	tps
th_1	x_1	y_1
th_2	x_2	y_2
th_3	x_3	y_3



if AUC ↑ model better.

Overfitting and underfitting

$x_1 \propto y$; $x_2 \propto \frac{1}{y}$; $x_3 \not\propto y$ (no impact)

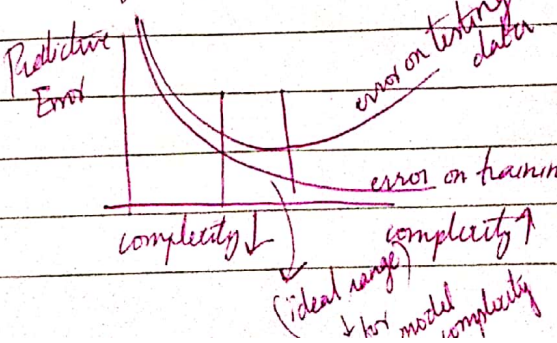
$\uparrow w_1$; $w_2 \downarrow$; $w_3 = 0$

always tries to get +ve impact on 'y'

Training Error	Testing Error
Training Loss	Testing Loss
Underfit	Overfit
Bias	Variance

→ A complex model: y = high order polynomial in X

→ A true (simpler) model: y = aX + b + noise



training acc	test acc
60	58 (HB LV)
60	40 (HB HV)
98	70 (LB HV)
98	96 (LB LV)

Scanned with CamScanner

Data Distribution to Trained Test should be uniform

like random shuffling

DT

underfit | grid split

overfit

train

training

test

Must to Split

Ways to Control Logistic Regression

→ Adjust Step Size

→ Adjust Iterations / stopping criteria for Gradient Descent

→ Regularization

hard constraints

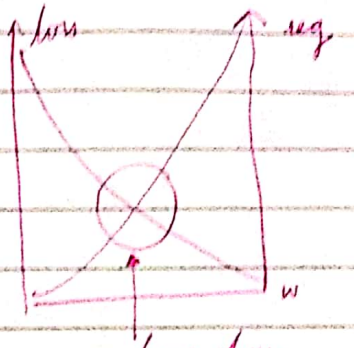
L_1 test error / test error for i iter

L_2 soft constraints

test err, > test err, (early stopping)

$$Loss(s) = \sum_i^n Loss(\hat{y}_i, y_i) + \lambda \sum_j^{\#\text{ weights}} |w_j| \rightarrow L_1 \text{ norm}$$

$w \uparrow$ $w \downarrow$



Lower down, lower value of regularizer

$$J(w) = \frac{1}{2m} \sum_{i=0}^{m} (h_w x^{(i)} - y^{(i)})^2 + \frac{\lambda}{2} \sum_{i=1}^{d} |w_i| \leftarrow L_1 \text{ norm}$$

$$J(w) = \frac{1}{2m} \sum_{i=0}^{m-1} (h_w x^{(i)} - y^{(i)})^2 + \frac{\lambda}{2} \sum_{i=0}^{d} w_j^2 \leftarrow L_2 \text{ norm}$$

$\lambda \uparrow$ (very high) $w \downarrow_0$ underfit

$\lambda \downarrow_0$ overfit

$$w_j = w_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_w x^{(i)} - y^{(i)}) x_j - \lambda w_j \right]$$

$$w_j = (1 - \alpha\lambda) w_j - \alpha \frac{1}{m} \sum_{i=0}^{m} (h_w x^{(i)} - y^{(i)}) x_j$$

regularizer factor

Scanned with CamScanner

Ways to Control Decision Tree Learning

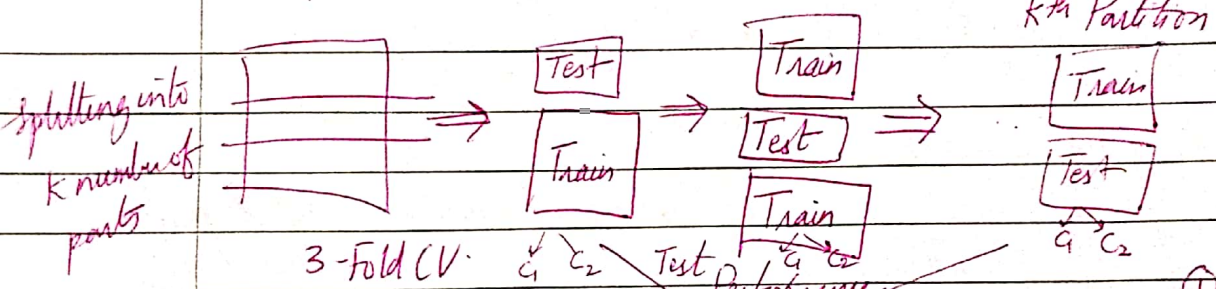
→ Increase min To split
→ Increase min Gain to Split
→ Limit total number of Nodes.
→ Penalize complexity

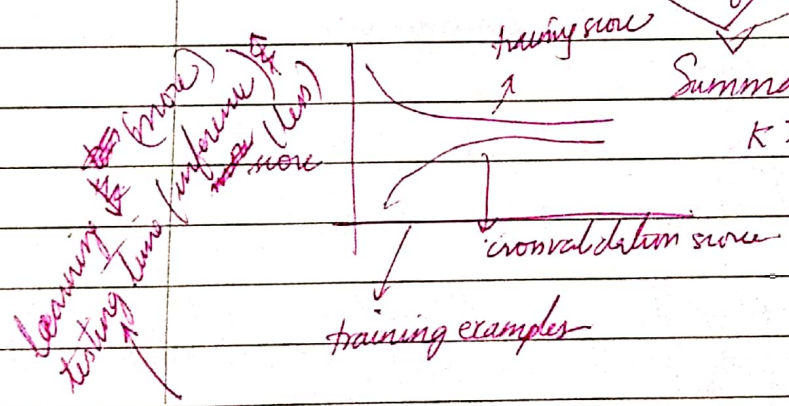
$$Loss(S) = \sum_{i}^{n} Loss(\hat{y_i}, y_i) + \lambda \log_2(\#Nodes)$$

node ↑ node ↓

Comparing Classifiers → test accuracy

K-Fold Cross Validation.





Summary Statistics over K test performances

① Randomize Data Set
② Perform K-fold CV

⇒ Parametric Learning Algo – Doesn't need I/p training data for inference → inference faster, learning slower
⇒ Non-Parametric Learning Algo – Locally Weighted Regression / K Nearest Neighbour

data ↑ model size ↑ (linear)
learning fast (less) & testing time (inference time) (more)

learning faster, inference slower

Scanned with CamScanner

Parametric learning Algo

Non-Parametric Learning Algo (Training data required for influence)

Recap Linear Regression.

fit w to minimize

$$\rightarrow J(w) = \sum_{i=0}^{m} \left(h_w x^{(i)} - y^{(i)}\right)^2$$

Locally Weighted Regression.

fit w to minimize

$$\sum_{i=0}^{n} \beta^{(i)} \left(y^{(i)} - h_w(x^{(i)})\right)^2$$

$x \rightarrow$ point where we want to make prediction.

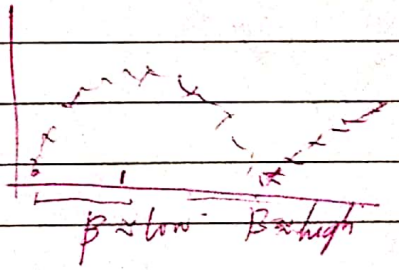
$x^i \rightarrow$ training data.

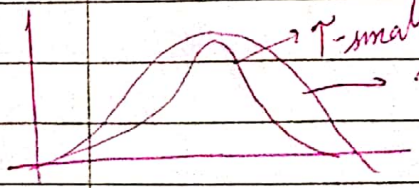
$\beta \rightarrow$ weighted function

$$\beta^{(i)} = \exp\left(-\frac{(x^i - x)^2}{2\tau^2}\right)$$

$|x^i - x|$ is small, $\beta^{(i)} \approx 1$

$|x^i - x|$ is large $\beta^{(i)} \approx 0$.

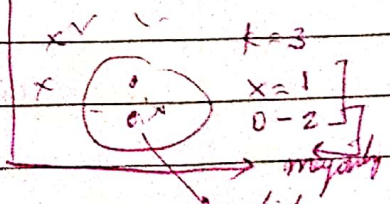




τ-small (overfit) → narrow focus

τ-large (underfit) → broader focus

K-Nearest Neighbour



k-odd

Training Phase → save the ~~model~~ example

Prediction phase → get test data x_t

find k training samples $(x_1, y_1) \ldots (x_k, y_k)$ closer to x_t value

classification → $(y_1, y_2 \ldots y_k)$ - majority prediction

regression → avg $(y_1, y_2 \ldots y_k)$ → prediction

max/min ...

Scanned with CamScanner

Voronoi Diagram

Distance (Euclidean Distance)

$x_i = \{x_{i1}, x_{i2}, \dots, x_{im}\}$

$x_j = \{x_{j1}, x_{j2}, \dots, x_{jm}\}$

$$D = \sqrt{\sum_{l=1}^{m} (x_{il} - x_{jl})^2}$$

↓ spherical → in 3D

Assumption in traditional KNN

- giving equal weight to all attributes
- scale of attribute range are same
- equal variance
- classes are spherical
- attributes have very less noise
- all attribute have same importance

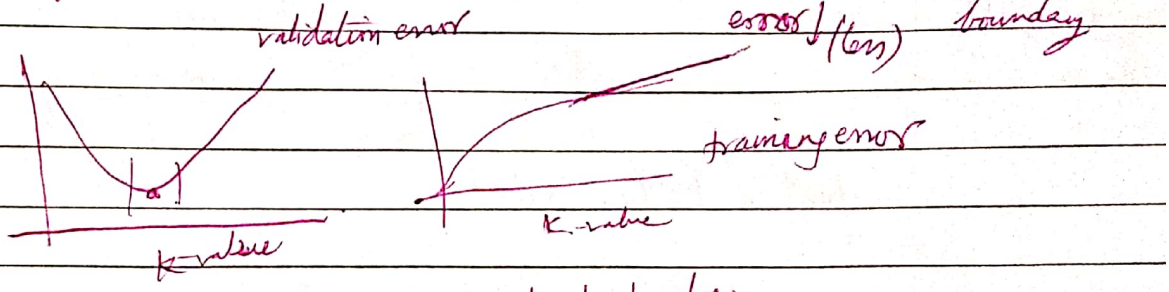
How to overcome?

- Use large K value
- Weighted distance function

Impact of K

K is less → more discrete, not smooth decision boundary (more error)

K is more → more smooth boundary (less error)



↳ Distance weighted KNN: (solution for k value)

- Take large value of K

$$\text{Prediction}_{test} = \frac{\sum_{i=0}^{k} w_i \cdot \text{class}_i}{\sum_{i=0}^{k} w_i} \qquad w_k = \frac{1}{\text{Dist}(x_k, x_{test})}$$

(more focus on closer, less focus on farther)

↳ Weighted Distance function-

$$D(x_i, x_j) = \sqrt{\sum_{i=1}^{m} w_i (x_{im} - x_{jm})^2}$$

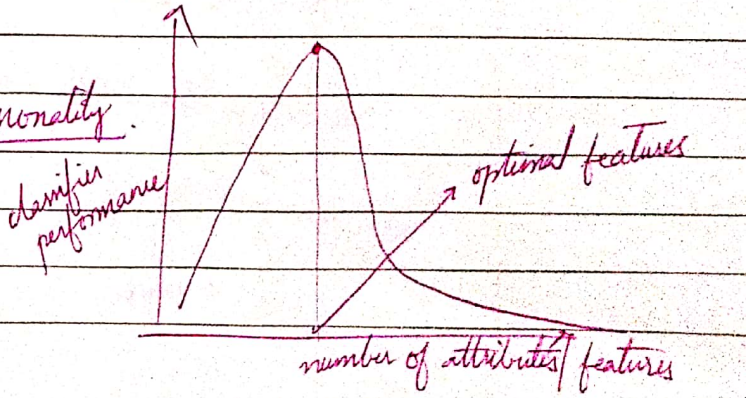
$w_i \to$ high (more imp. feature)

$w_i \to$ low (less imp. feature)

$w_i = 0$ (no impact feature)

Feature Engineering.

Curse of Dimensionality.



Scanned with CamScanner

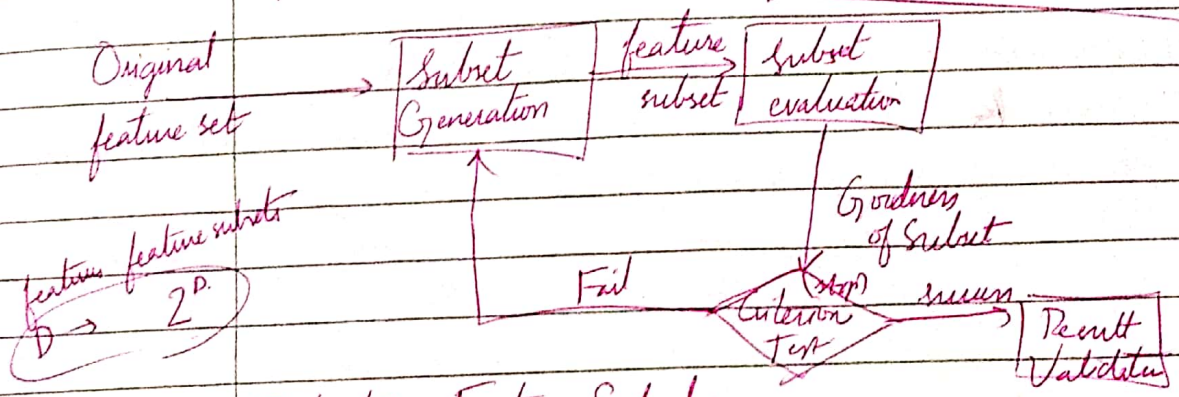
→ Irrelevant feature
→ Redundant feature
→ Limited training examples
→ Limited computational resources

Feature Selection → maintain accuracy or increase

Feature Extraction
(higher dimension to lower dimension) feature

$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ $n \geq$ $\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_n \end{bmatrix}$

Steps



Evaluation Feature Subset

- Supervised Method (Wrapper method)
  → Training using selected feature subset
  → Estimate error on validation dataset

- Unsupervised (Filter method)
  → Look at input only
  → Select the subset that has most information

Filter Method

original features
↓
importance score
↓
selecting the best
↓
learning algorithm
↓
performance

Wrapper Method

Original features
↓
→ generate a subset
↓
learning algo
↓
performance

selecting best subset

Scanned with CamScanner

Method for filter method - ① Select uncorrelated feature
② Forward Selection Algo
③ Backward Selection Algo

start with empty feature set, try each remaining feature, estimate error (cost/reg), select feature that gives most improvement, stop where there is no improvement

Start with full feature set, Try removing features, Drop feature with smallest impact on error

Stopping criteria → same for all

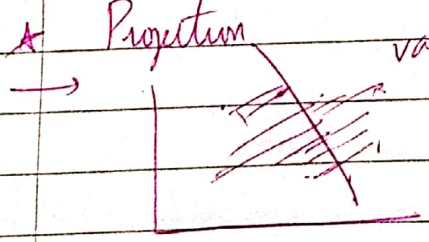
Recursive Feature Elimination $A_t \leq A_{t+1}$
- Use Linear Classifier to find W (weight of each feature)
- Steps: Compute W on all features
- Remove feature with smallest $|w_i|$ → magnitude!
- recompute W on reduced data (features)
- repeat this till stopping criteria doesn't meet

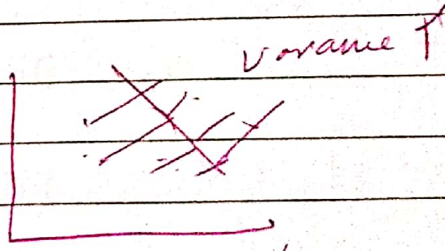
$[w_1, w_2 \dots w_n]$ → least w value (remove) → again compute → $[w_1, w_2 \dots w_{n-1}]$ → new set

Feature Extration - $\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = f\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ $N > M$

Projection of higher dimension to lower dimen
→ Uncorrelated, can not reduce further
→ high variance

Projection →





$z = w^T x$

$\dim z < \dim x$

Variance → σ^2

$$\sigma^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n}$$ (deviation, mean)

Standard deviation = sq. root of variance

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

Scanned with CamScanner

Covariance

→ $\text{Variance}(x) = \frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})(x_i-\bar{x})$

→ $\text{Covariance}(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})$

- $\text{Covariance}(x,x) = \text{var}(x)$
- $\text{Cov}(x,y) = \text{Cov}(y,x)$

→ Covariance Matrix ⟶ is symmetric

$$\text{Cov}(\Sigma) = \begin{bmatrix} \text{cov}(x_1,x_1) & \text{cov}(x_1,x_2) & \cdots & \text{cov}(x_1,x_m) \\ & \text{cov}(x_2,x_2) & & \\ \text{cov}(x_m,x_1) & & \cdots & \text{cov}(x_m,x_m) \end{bmatrix}$$

$$\text{cov}(\Sigma) = \frac{1}{n}(x-\bar{x})(x-\bar{x})^T \text{ ; where } X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

(vectorized)

→ diagonal elements are variances, i.e. $\text{cov}(x,x) = \text{var}(x)$

→ covariance matrix is symmetric

→ it is a (+ve) semi definite matrix

- All eigenvalues must be real.
- All eigenvectors corresponding to different eigenvalues are orthogonal
- All eigenvalues are greater than or equal to zero
- Covariance matrix can be diagonalized.

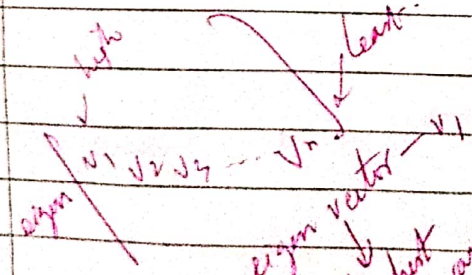
$$\text{cov} = PDP^T$$

→ How to Compute PCs → Covariance Matrix S

→ Compute its eigenvectors: $\{a_i\}_{i=1}^{d}$

→ first p eigenvectors $\{a_i\}_{i=1}^{p}$ corresponds to p largest eigenvalues from the P^T of PCs.

$$G = [a_1 a_2 \cdots a_p]$$



→ transformation G consists of

Scanned with CamScanner

→ Most weights are in fully connected layers

→ $W = USV^T$

- $W \in R^{m \times n}$, $U \in R^{m \times m}$, $S \in R^{m \times n}$, $V^T \in R^{n \times n}$

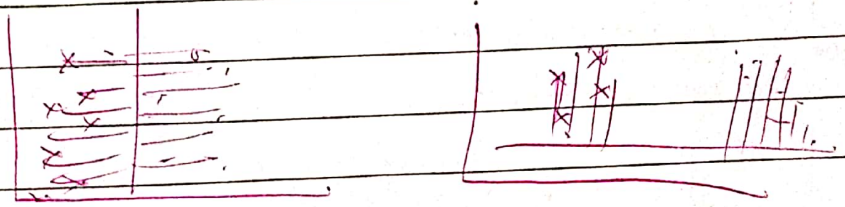
S → diagonal, decreasing magnitudes along diagonal

$$[A] = [U][W_1, W_2, W_3][V^T]$$

- $\tilde{W} \in R^{m \times n}$ - $\tilde{U} \in R^{m \times k}$, $\tilde{S} \in R^{k \times k}$, - $\tilde{V}^k \in R^{k \times n}$.

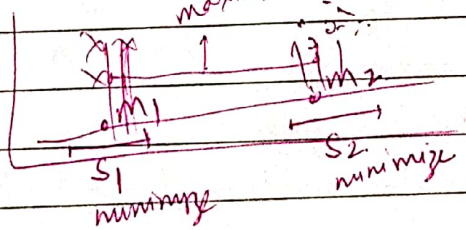
k → hyperparameter

→ PCA good for classification?



→ Linear Discrimination Analysis (LDA)

- Maximize b/w class distance
- Minimize within class distance



maximize dist b/w m_1, m_2

minimize dist within s_1, s_2

$$J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$ → means, → std dev

→ Fisher Linear Discriminants

- Means as far away as possible
- Scatter is small as possible

$$J(w) = \frac{(m_1 - m_2)^2}{s_1^2 + s_2^2}$$

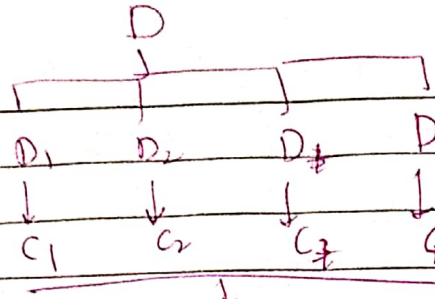
→ Ensemble Learning

Strong Learner — single or multiple

Weak Learner → just better than random guesser accuracy → > 0.5

combine →

Scanned with CamScanner



conditions → classifies outputs for same input should be diverse
→ classifier operates independent or partial independent
→ classifies have unique or local knowledge

Hybridization (when we combine)
we can combine → works if there is some 'random factor'
we can combine the same learning algo trained over diff subsets of training data

for certain, we use same algo over same data but with different weighting over data instances

Randomization of DTs.
→ Data Randomization → diff subsets of data for each tree
→ Random Subspace → each tree growing with random subset of features
→ Algorithm randomization → instead of best split attribute choose randomly

→ Aggregation Methods
- Majority Voting
- Weighted Majority Voting → can weight each classifier } o/p is class label

- mean, product, minimum, max, weighted mean, avg.
↳ o/p is numeric

→ stacking → instead of above, we can train another classifier on o/p values of base classifiers

Generate Base learners → diverse → max ensemble accuracy

Bagging / Random Forest / Boosting / AdaBoost

Bagging = Bootstrap + Aggregating

→ Uses Bootstrap resampling

↓

Sampling with replacement

Bagging — n → number of instances

↳ For each of t iterations

Sample n instances from training set (with replacement)

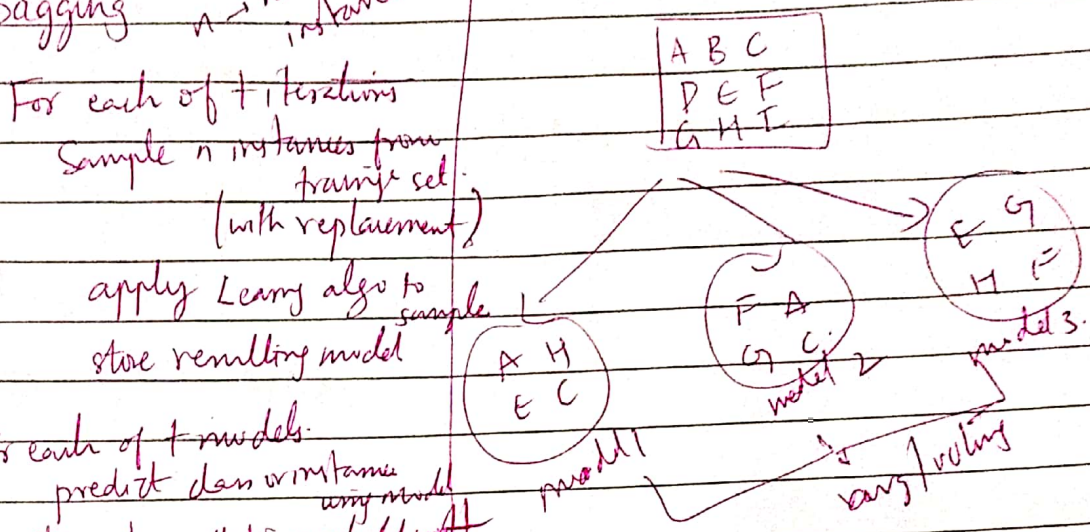
apply Learning algo to sample

store resulting model

For each of t models:

predict class of instance using model

return class that is predicted most



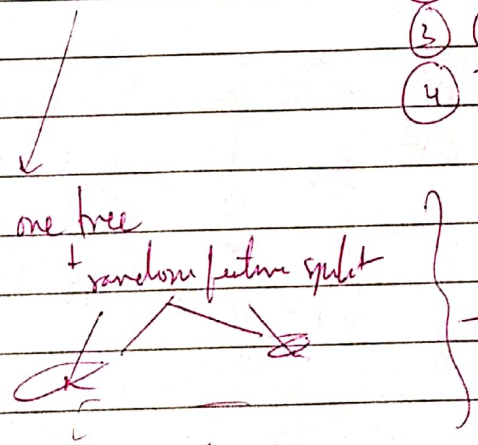
Random Forest: Input: training data set S n, T, m.

① Choose T — no. of trees

② Choose m — no. of variables to split

③ Grow T trees.

④ To classify point X, collect votes from each tree



→ many trees by bootstrapped data

used for testing purposes → Out of the Bag } Samples not in the training data

↓

proportion of these samples wrongly classified

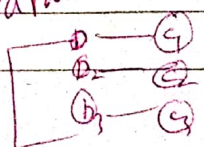
↓

"Out of Bag Error"

Bagging

↓

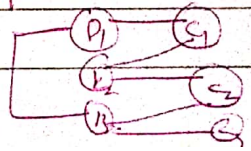
Parallel



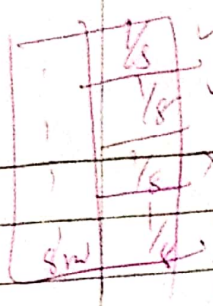
Boosting

↓

Sequential



Scanned with CamScanner



$D \rightarrow L_1 \alpha_1$

increase $D_2 \rightarrow L_2 \alpha_2$ to wrongly classified

Boosting

Classification $(-1, +1)$

$D_1 = \frac{1}{m}$ m data points

For $t = 1$ to T

Train learner using D_t

Get Weak classifier h_t

Choose/find $\alpha_t \leftarrow f(\epsilon)$ (error)

update $D_{t+1} = f(D_t)$

previous correct always $\sum D = 1$

output $\rightarrow H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

~~$D_{t+1}(i)$~~

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

$-1/+1 \quad -1/+1$

$Z_t \rightarrow$ normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

y_i	$h_t(x_i)$	
-1	-1	$e^{+\alpha}$
-1	$+1$	$e^{+\alpha}$
$+1$	$+1$	$e^{-\alpha}$
$+1$	$+1$	$e^{-\alpha}$
-1	$+1$	
-1	$+1$	

y_i same

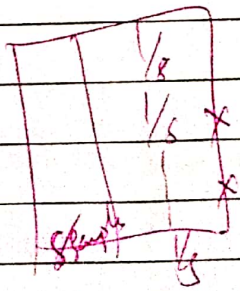
choose $\alpha_t \rightarrow$ to minimize training error

$\epsilon \in \mathbb{R}$

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

$$\epsilon_t = \sum_{i=1}^{m_t} D_t(i) \delta(h_t(x_i) \neq y_i)$$

not classified correctly



$\epsilon_t = \frac{1}{8} + \frac{1}{8}$

$$\sum_{i=1}^{m_t} D_t = 1$$

Scanned with CamScanner

Random Forest

→ full sized trees

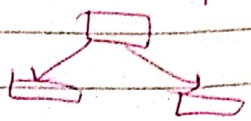
→ equal vote on final classification

Forest of Trees (Adaboost)

Stumps

usually one node and 2 leaves

(known as stumps)

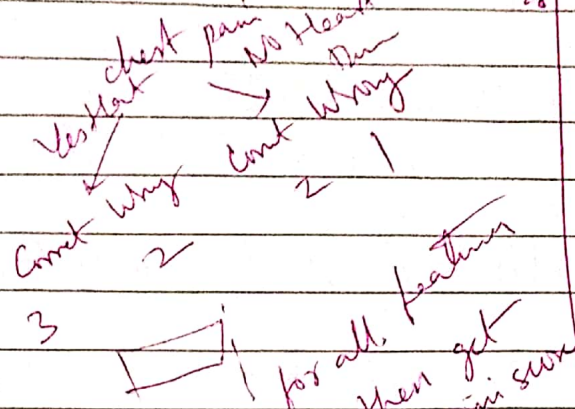


stumps → weak learner

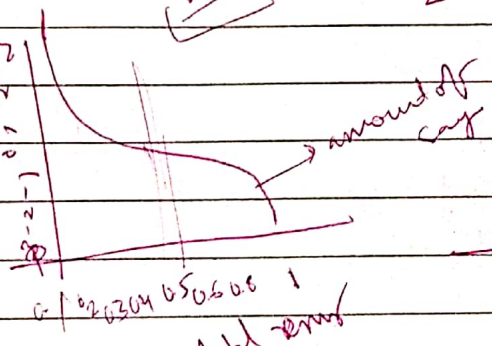
→ some stumps may get more say in final classification than another

$d_1, d_2, \ldots, d_n$

Initially sample weights all same like 1/8



$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right) \rightarrow \alpha = \frac{1}{2}\log\left(\frac{1-TE}{TE}\right)$$



New sample weight = sample weight × $e^{\text{amount of say}}$

$$D_{t+1}(i) = D_t(i) \cdot e^{(-\alpha_t y_i h_t(x_i))} \rightarrow Z_t$$

old weight | New weight | normalized

old weight	New weight	normalized
1/8	0.07	0.07
1/8	0.07	0.07
1/8		0.49
	0.33	
		0.07

sum all of them and divide each one by it

final new weight

maybe even multiple times

0.21 + 0.49 = 0.70

1/8 b/w 0.21 and 0.70 we add

8 random numbers b/w 0 and 1

small range

1/8 in between 0 to 0.07 then include b/w 0.07 to 0.14 also

finally we want

$H(x_i) = \text{sign}\left(\sum \alpha_t h_t(x_i)\right)$

proceed focusing on errors we have in subsequent stumps

Gradient Boosting

= Gradient Descent + Boosting.

$$h(x) = avg\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \leftarrow \text{Regression}$$

$$h(x) \overset{or}{=} sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right) \leftarrow \text{Binary classification}$$

AdaBoost : Shortcoming of Previous Model.
↳ high weighted data.

GB → Shortcomings are identified using Gradient

$F(x) = 0.6$ | $y_1 = 0.8$
$F(x_2) = 1.3$ | $y_2 = 1.2$
→ we want to improve current modelling

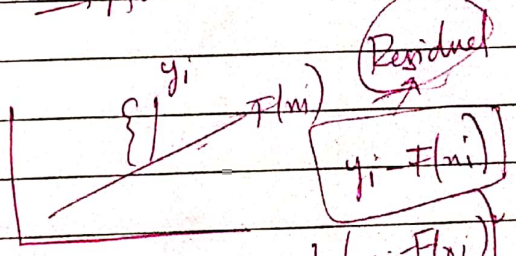
↳ You can't change $F(x)$
→ Add an h to F

$F(x) + h(x)$ → final model

$F(x_1) + h(x_1) \approx y_1$

$F(x_2) + h(x_2) \approx y_2$

$h(x_1) = y_1 - F(x_1)$



we want to approximate $y_1 - F(x_1)$ to achieve $h(x)$.

∴ we can fit regression tree h to data

$(x_1, y_1 - F(x_1)), (x_2, y_2 - F(x_2)) \ldots$

$h(x)$ → not able to satisfied → add more trees

$h_1(x) \rightarrow h_2(x)$

$y_1 - (F(x_1) + h_1(x_1))$

Scanned with CamScanner

Review G.D

$$w_i = w_i - \alpha \frac{\partial J(w)}{\partial w_i}$$

$$L(w) = \frac{1}{2}\left(y - h_w(x)\right)^2$$

we want to minimize

$$J(w) = \sum_{i=1}^{m} L(w)$$

$$L(y, F(x)) = \frac{1}{2}\left(y - F(x)\right)^2$$

minimize

$$J = \sum_{i=1}^{m} L(y_i, F(x_i))$$

adjust $F(x)$

$$\frac{\partial J}{\partial F(x_i)} = \frac{\partial \frac{1}{2}\left(y - F(x)\right)^2}{\partial F(x)}$$

$$= F(x) - y$$

$$y_i - F(x_i) = -\frac{\partial J}{\partial F(x_i)}$$

$$F(x) = F(x) + h(x)$$

$y - F(x)$

$$F(x) = F(x) - \eta \frac{\partial J}{\partial F(x)}$$

Gradient Boost

$F(x)$ + 0.1 x [First tree] (input LR)

based on residual

+ 0.1 x [Second tree]

Average Weight → of all column → $F(x)$

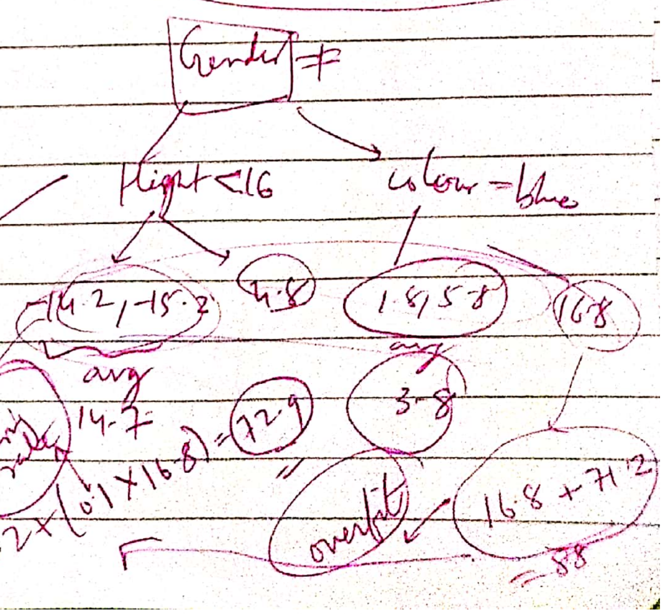
avg weight = 71.2

Adaboost → small tree (called stump)

Gradient Boost → we are starting from one leaf and restricting on no. of leaves

trees larger than stump

Height	Fav. Colour	Gender	Weight	Residual
			88	16.8
			76	
			56	
			73	
			77	
			57	



tree made with simple DT method

restrict leaves = 4

learning rate

$88 / 71.2 \times (0.1 \times 16.8) = 72.9$

overfit

$16.8 + 71.2 = 88$

Scanned with CamScanner

__/__/____

F(x) F(x) + $h_1(x)$.
Residual

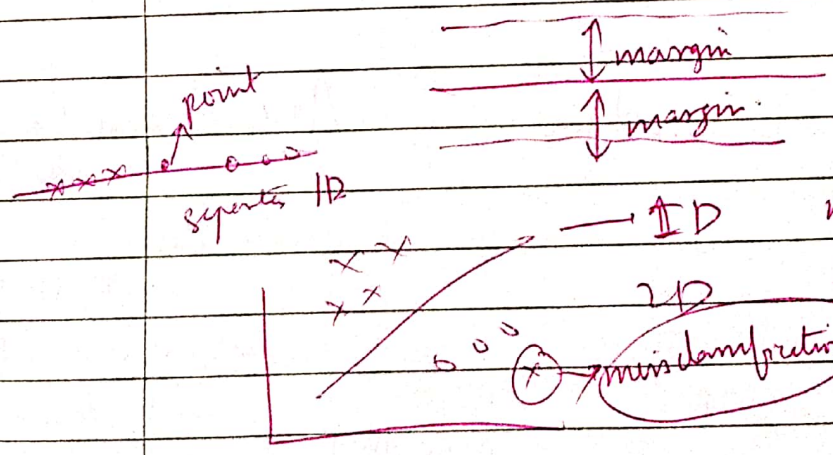
16.8 → 15.1
4.8 → 4.3
-15.2 → -13.7
1.8 → 1.4
5.8 → 5.4
-14.2 → -12.7

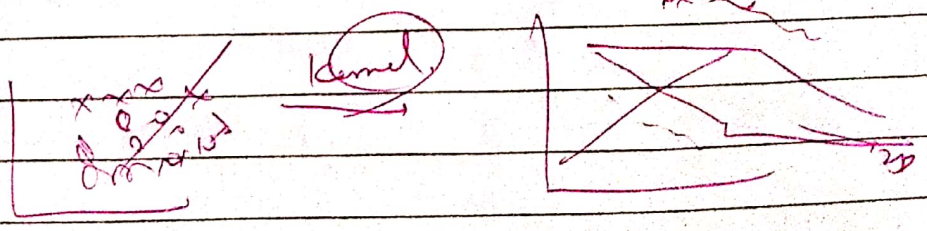
values decrease (overall)

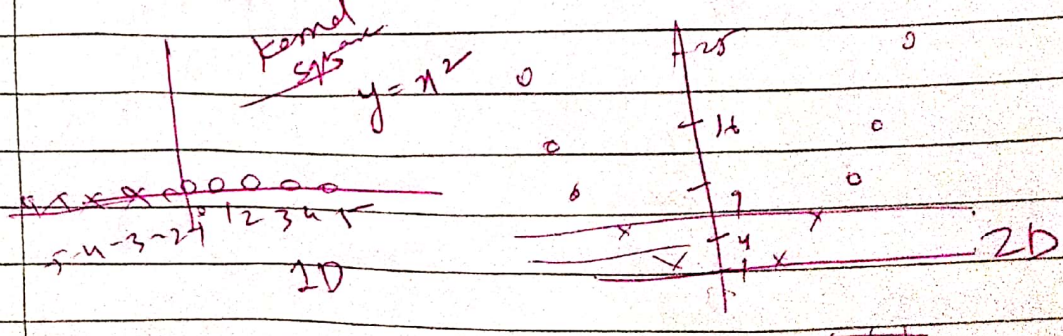
↳ small step in right direction (minimize residual)

$F(x) + \alpha_1 h_1(x) + \alpha_2 h_2(x)$

Support Vector Machine







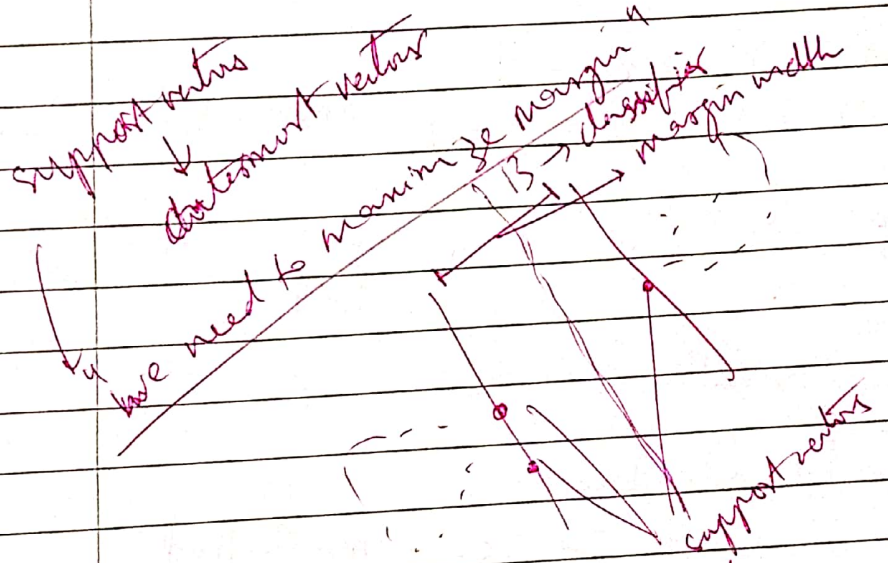
Scanned with CamScanner

SVM.

① Define optimal hyperplane is (taking into account that it needs to be computed efficiently)
↓
maximize margin

② Generalize to non-linearly separable problems.
have a penalty term for misclassification

③ Map data to high dimensional space where it is easily to classify with linear decision surfaces
↓
reformulate so that data are mapped implicitly to this space



Linear Separating Hyperplanes

→ this linear separating plane need not pass through the origin of our feature space, i.e. it does not need to include the zero vector as an entity within the plane. Such hyperplanes are called 'affine'

→ Real dimensional ~~p-plane~~ p-dimensional feature space, known mathematically as $\mathbb{R}^p$ ↓ our linearly separating plane is an affine p-1 dimensional space embedded within it

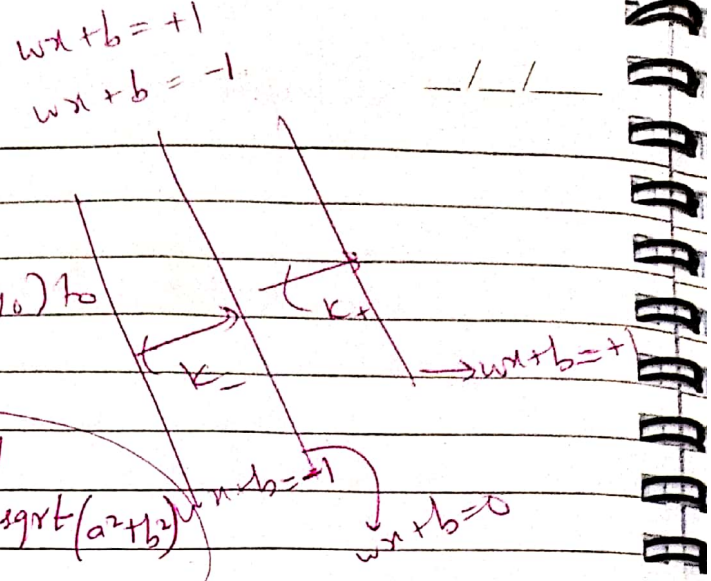
→

$y_i \in \{-1, +1\}$

$f(x) = \text{sign}(w \cdot x + b)$

$wx + b > 0$

$wx + b < 0$



~~Dist~~ Dist from a point (x_0, y_0) to a line

$ax + by + c = 0$

$$\frac{|ax_0 + by_0 + c|}{\text{sqrt}(a^2 + b^2)}$$

$x_i \cdot w + b \geq +1$ when $y_i = +1$

$x_i \cdot w + b \leq -1$ when $y_i = -1$

} combined $y_i (x_i \cdot w) \geq 1$

$$\frac{w \cdot x + b}{\sqrt{\sum_i w^i}} \rightarrow k = \frac{1}{\|w\|} \rightarrow 2k = \frac{2}{\|w\|}$$

$$k = \frac{1}{\sqrt{\sum_i w^i}}$$

$$\max \frac{2}{\|w\|}$$

So the problem becomes →

$$\max \frac{2}{\|w\|} \quad \text{or} \quad \min \frac{1}{2} \|w\|^2$$

$$s.t.\ y_i (w \cdot x_i + b) \geq 1,\ \forall x_i$$

Linear, Hard-Margin SVM Formulation

Find w, b that solve $\min \frac{1}{2} \|w\|^2$

$s.t.\ y_i (w \cdot x_i + b) \geq 1$

Quadratic program: quadratic objective, linear (inequality constraints)

- There is also a unique minimizer, i.e. w and b values to provide the minimum.
- No solution if the data are not ~~linearly separable~~

→

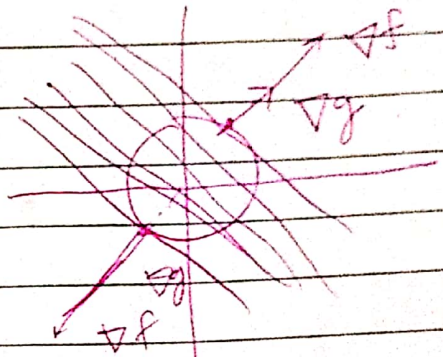
Lagrange Multipliers

$$\underset{w,b,\alpha}{\text{Minimize}}\ \phi(w) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i y_i (w x_i + b) + \sum_i \alpha_i$$

- Convex quadratic programming problem
- Duality theory applies

$f(x,y) = 3x + 4y \leftarrow$ Max/Min

s.t $x^2 + y^2 \leq 1$



$\nabla f = \lambda \nabla g$

$\nabla f - \lambda \nabla g = 0$ max/min

$f = 3x + 4y$ $\quad g = x^2 + y^2 - 1$ $\quad \frac{\partial g}{\partial x} = 2x + 0 + 0$

$\nabla f = \lambda \nabla g$

$\nabla f - \lambda \nabla g = 0$

$\frac{\partial f}{\partial x} - \lambda \frac{\partial g}{\partial x} = 0$ $\quad \frac{\partial f}{\partial y} - \lambda \frac{\partial g}{\partial y} = 0$

$3 - \lambda 2x = 0$ $\quad 4 - \lambda 2y = 0$

$x = \frac{3}{2\lambda}$ $\quad y = \frac{4}{2\lambda}$

$\frac{9}{4\lambda^2} + \frac{16}{4\lambda^2} = 1$ $\quad 4\lambda^2 = 25$

$\lambda = \pm 5/2$

max $f(x,y)$

$\lambda = +5/2$

$x = 3/5, y = 4/5$

min $f(x,y)$

$\lambda = -5/2$

$x = -3/5, y = -4/5$

→

Scanned with CamScanner

$\min \frac{1}{2}\|w\|^2$

s.t $y_i(w \cdot x_i + b) \geq 1, \forall x_i$

$L = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i [y_i(wx_i + b) - 1]$

$\frac{\partial L}{\partial w} = w - \frac{\partial}{\partial w}\left[\sum \alpha_i y_i w x_i + \sum_i \alpha_i y_i b - \sum \alpha_i\right]$

$\Rightarrow w - \sum_i \alpha_i y_i x_i = 0$ $\quad \frac{\partial L}{\partial b} = \sum_i \alpha_i y_i = 0$

$w = \sum_i \alpha_i y_i x_i$

$w = \sum_i \alpha_i y_i x_i$ $\quad \sum \alpha_i y_i = 0$

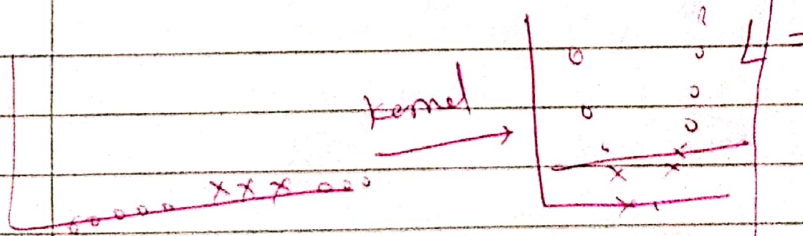
$L = \frac{1}{2}\left(\sum_i \alpha_i y_i x_i\right)\left(\sum_j \alpha_j y_j x_j\right) - \sum_i \alpha_i y_i w x_i + \sum_i \alpha_i y_i b + \sum_i \alpha_i$

$= \frac{1}{2}\left(\sum_i \alpha_i y_i x_i\right)\sum_j \alpha_j y_j x_j - \left(\sum_i \alpha_i y_i x_i\right)\left(\sum_j \alpha_j y_j x_j\right) + \sum_i \alpha_i$

$= -\frac{1}{2}\left(\sum_i \alpha_i y_i x_i\right)\left(\sum_j \alpha_j y_j x_j\right) + \sum_i \alpha_i$ $\quad \left[\sum_i \alpha_i\right]$

$L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j x_i x_j$

↓ minimize



$L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$

$\alpha_1 > 0, \alpha_2 > 0 \cdots$

$\alpha_i > 0$

$\phi(x_i)\,\phi(x_j) \rightarrow$ map

$L = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$

↓ we pass through kernel

once you pass through kernel simply gives out dot product

$= \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle \phi(x_i) \cdot \phi(x_j) \rangle$

$K(x_i, x_j) \in \phi(x_i) \cdot \phi(x_j)$

$k(x_i, x_j) = \langle \phi(x_i) \cdot \phi(x_j) \rangle$

⟹ Polynomial Kernel

$k(x_i, x_j) = (x_i \cdot x_j - 1)^P$

⟹ Gaussian Kernel → also in sklearn library uses

$k(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$

→ Find max and min values of $f(x,y) = 81x^2 + y^2$ subject to constraint $4x^2 + y^2 = 9$

$f(x) = 81x^2 + y^2$ $g(x) = 4x^2 - y^2 = 9$ $L = (81x^2 + y^2) - \lambda(4x^2 - y^2 - 9)$

$\frac{\partial L}{\partial x}$ ↗ $162x - \lambda 8x = 0$ $2x(81 - 4\lambda) = 0$

$x = 0, \lambda = 81/4$

$4x^2 + y^2 = 9$ → $y^2 = 9$ $y = \pm 3$ → $(0, +3)$ $(0, -3)$

$\frac{\partial L}{\partial y}$ ↘ $2y - 2\lambda y = 0$

$2y(1 - \lambda) = 0$

$y = 0, \lambda = 0$

$4x^2 = 9$ $x = \pm 3/2$ → $(+\frac{3}{2}, 0)$ $(-3/2, 0)$

$f(\pm 3/2, 0)$ $81x^2 + y^2$ (min)

min value $= 81 \times 9/4$ (max) $81 \times +9/4$

$-\frac{729}{4}$ $+\frac{729}{4}$

$f(x,y,z) \Rightarrow 8xyz^2$ $200(x+y+z)$

The company produces 100 units annually

$g(x,y,z) \Rightarrow x+y+z = 100$

$L = f(x,y,z) - \lambda g(x,y,z)$

$\frac{\partial L}{\partial x} =$ ①

$\frac{\partial L}{\partial y} =$ ②

$\frac{\partial L}{\partial z}$ ③

(eq1, eq2, eq3) = no soln → not going to work; single soln; ∞ solns ↘ multiple choices (check out, some problem)

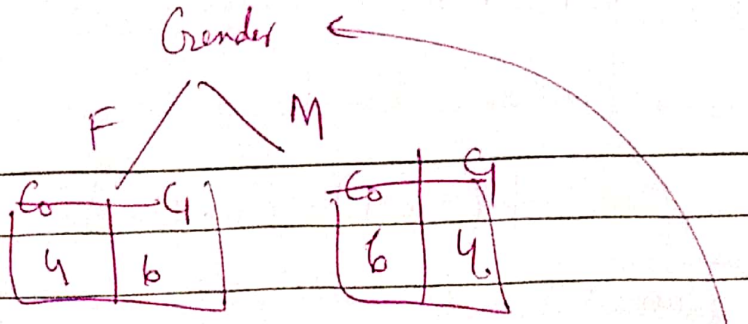
revenue model

$f(h,s) = 200h^{2/3}s^{1/3}$

$g(h,s) = 20h + 170s = 20000$

budget constraint

Scanned with CamScanner

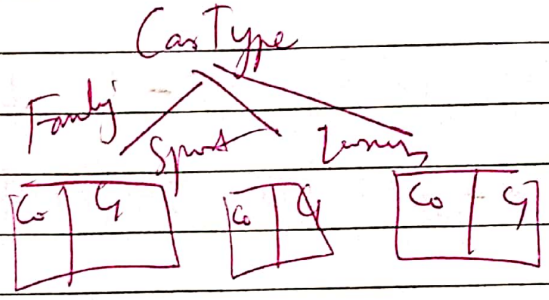


$$G_1 = 1 - \left(\frac{4}{10}\right)^2 - \left(\frac{6}{10}\right)^2$$

$$G_2 = 1 - \left(\frac{6}{10}\right)^2 - \left(\frac{4}{10}\right)^2$$

weighted sum

$$\frac{(10)\,G_1 + G_2(10)}{20}$$



Information gain ↑, Split that

Gini impurity ↓ split that

Scanned with CamScanner