Machine Learning:

(1) Supervised (2) Unsupervised (3) Reinforcement learning.

~~1/2 Regression~~

Sklearn → many ml algorithms already implemented for us. also provides some data sets.

(1) Linear Regression: Linear relation b/w x and y. continuous values

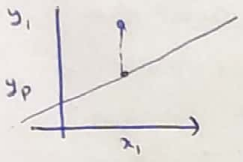
$y = m_1 x_1 + m_2 x_2 \cdots m_n x_n + b$.

We need to find $m_1, m_2 \cdots m_n, b$ ($n+1$ parameters)

Goodness of fit / R-squared / coefficient of determination

$$= 1 - \frac{\sum (y - y_{pre})^2}{\sum (y - \bar{y})^2}$$

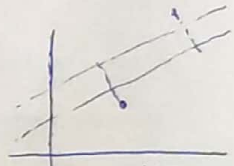
sklearn has function called ~~score~~ score. ~~to~~ find this.



Let the line $y = mx + c$

Error for pt = $y_i - (mx_i + c)$

Total error = $\sum |y_i - (mx_i + c)|$



this value will be same for 2 lines.

so we take $\sum (y_i - (mx_i + c))^2$

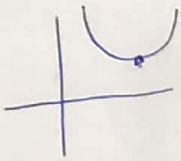
so we need find m and c for cost function to have minimum value.

cost $f^n = \sum_i (y_i - (m_1 x_1^i + m_2 x_2^i \cdots m_n x_n^i + c)^2$

where $\{x_1, x_2, x_3 \cdots x_n\}$ are ~~data~~ features.

$= f(m, c)$ (assuming 1D).

$\frac{df(m,c)}{\partial m} = 0 \qquad \frac{\partial f(m,c)}{\partial c} = 0$



solve for m and c.

$$\frac{df(m,c)}{\partial m_k} = \sum_i (y_i - (m_k x_i + c))(x_i) = 0$$

$$\Rightarrow \sum \frac{x_i y_i}{N} - m \sum \frac{x_i^2}{N} + c \underbrace{\sum \frac{x_i}{N}}_{\text{mean}} = 0$$

$$\Rightarrow (x * y).\text{mean}() - m(x^2).\text{mean}() - c.\text{mean}() = 0 \quad \text{--- (1)}$$

Diff w.r.t c: $\sum_i (y_i - (mx_i + c)) = 0$.

$$\sum c = \sum y_i - \sum m x_i$$

$$cn = \sum y_i - m \sum x_i$$

$$c = y.\text{mean}() - m.\text{mean}() \quad \text{--- (2)}$$

(i) $\Rightarrow (x^* y).\text{mean}() - m.(x^2).\text{mean}() - c^* x.\text{mean}() = 0$

(ii) $\Rightarrow c = y.\text{mean}() - m.x.\text{mean}()$.

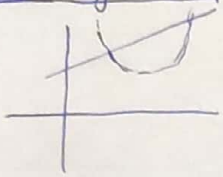
$= .(x^* y).\text{mean}() - m(x^2).\text{mean}() - y.\text{mean}()^* x.\text{mean}()$
$+ m\, x.(\text{mean}()\, x.\text{mean}() = 0$

$$m = \frac{(x^* y).\text{mean} - y.\text{mean}()^* x.\text{mean}()}{(x^2).\text{mean} - x.\text{mean}()^* x.\text{mean}()}$$

$$c = y.\text{mean}() - m^* x.\text{mean}().$$

$$y_p = mx + c$$

Finding complex decision boundary using LR.



x	y

let insert a dummy variable $x' \to x^2$

i.e

x	x'	y

Now we can do LR $y = m_1 x + m_2 x' + c$

↳ eq of plane.

Also x and x' are dependent on each other, we can do dimensionality reduction and we can get parabola as decision line instead of st. line.

So we can do many stuff → $\log x$, x^3, ... $x_1 * x_2$, $x_1 * x_2^2 * x_3$

So we can add many dummy colums looking at which are important. and may do dimensionality reduction later.

But adding many feature may lead to overfitting.

$$y = \underbrace{m_1 x_1 + m_2 x_2 \ldots}_{n+1} \; m_n x_n + \underbrace{m_{n+1} x}_{=1}$$

$$X \to \begin{bmatrix} \\ \\ \\ \end{bmatrix}_{m} \quad \theta = \begin{bmatrix} m_1 \\ m_2 \\ \vdots \\ m_{n+1} \end{bmatrix}_{n+1} \qquad Y_p = X * \theta$$

To find θ = $\boxed{(X^T X)^{-1} X^T Y_{\text{actual}} = \theta \quad \text{(Normal equation)}}$

Complexity → $O(n^3)$ as we slow it n is large calculate $X^T X$.

or $(n+1) * n$ $n * (n+1)$ $(n+1) * n$ $n * 1$

Gradient Decent

Cost fun $= f = \frac{1}{N} \sum (y_i - (m x_i + c))^2$

$m' = m - \alpha \text{ slope}_m$

$c' = c - \alpha \text{ slope}_c$

start with random value of m and c

$\alpha \rightarrow$ learning rate

when $\text{slope}_m = \frac{df}{\partial m} = \frac{+2}{N} \sum (y_i - m x_i - c)(-x_i)$

$\text{slope}_c = \frac{\partial f}{\partial c} = \frac{+2}{N} \sum (y_i - m x_i - c)(-1)$

use adaptive α, if cost value increase, decrease α.

$m, c = \text{grad\_de}(\text{data}, \alpha, \text{iterations})$

For general Gradient Decent:

Cost fun $= f = \frac{1}{M} \sum_i \left(y_i - (m_1 x_i^1 + m_2 x_i^2 \ldots\ldots m_n x_i^n + \underline{c} \; m_{n+1}(1)) \right)^2$

$\frac{\partial f}{\partial m_j} = \frac{-2}{M} \sum_i \left(y_i - (m_1 x_i^1 + m_2 x_i^2 + m_3 x_i^3 \ldots m_{n+1} *) \right)(x_i^j)$

$m_j' = m_j - \alpha \frac{\partial f}{\partial m_j}$

variations of gradient Decent

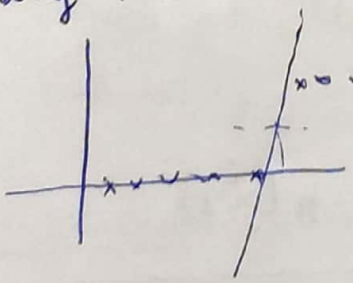
Batch Gradient: Consider whole dataset in 1 iteration

stochastic GD: Consider point by point during iteration and keep updating row.

Mini batch G.D.:

Classification discrete output.

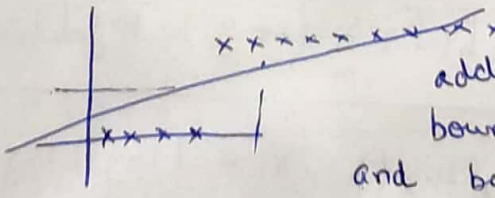
Why not linear Regression?



value is continuous instead! we want discreate.

let us assume a cutoff at $y = 0.5$

but let us have few more data points.



adding few extremes the boundary changes a lot and best fit line would spoiled.

Also we get large values for extreme x

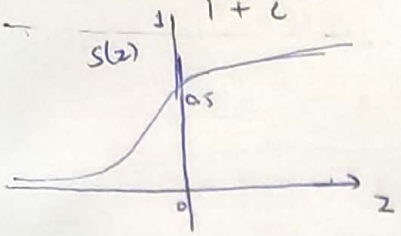
but we only need {0, 1}

Logistic Regression

$$h(x) = \frac{1}{1+e^{-g(x)}}$$

Sigmoid fn.

$$S(z) = \frac{1}{1+e^{-z}}$$



$g(x) = mx + c$ ← (basic LR line)

$= m^T x$

$m = [m_1, m_2 \cdots m_{n+1}]$

$x = [x_1\ x_2 \cdots x_n\ 1]$

hypothesis

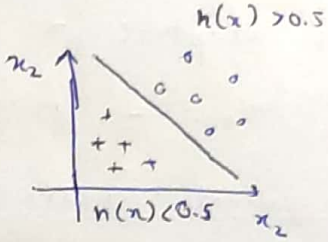
$$h(x) = \frac{1}{1+e^{-m^T x}}$$

Threshold = 0.5

$h(x) > 0.5 \Rightarrow y_{pred} = 0$

we know is $z > 0$ $S(z) > 0.5$

so if	$m^T x > 0$	$h(x) > 0.5$	$y_{pred} = 1$
Illy	$m^T x \leq 0$	$h(x) \leq 0.5$	$y_{pred} = 0$



* we will have a linear decision boundary

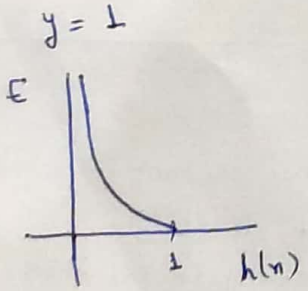
We need to find m.

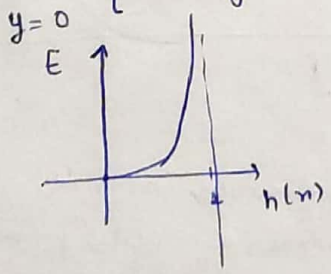
Cost function

in LR it was $\sum (y^i - h(x))$

in Logistic Regression we can use this because it has so many local minima. (Not a convex function)

So $$E(h(x), y) = \begin{cases} -\log(h(x^i)) & y = 1 \\ -\log(1-(h(x^i)) & y = 0 \end{cases}$$





$$E(h(x^i), y^i) = -y^i \log(h(x^i)) - (1-y^i) \log(1-h(x^i))$$

$$E(h(x), y) = \frac{1}{m} \sum_{i=1}^{m} \left(-y^i \log(h(x^i)) - (1-y^i) \log(1-h(x^i)) \right)$$

$$h(x^i) = \frac{1}{1+e^{-m^T x}}$$

We need to find optimal values of m.

We will use gradient descent.

$$m_j := m_j - \alpha \frac{\partial E}{\partial m_j}$$

On substituting values of $h(x)$ in Error function.

$$E^i(m) = m^T x^i + y^i(-m^T x^i) + \log\left(1 + e^{-m^T x^i}\right)$$

$$E(m) = \log\left(1 + e^{m^T x_i}\right) - y^i (m^T x^i)$$

$$\frac{dE^i(m)}{\partial m_j} = \frac{1}{1+e^{m^T x_i}} e^{m^T x_i} x_j^i \qquad -y^i x_j^i$$

$$= -\left(y^i - \frac{1}{1+e^{-m^T x_i}}\right) x_j^i$$

$$= -\left(y^i - h(x_i)\right) x_j^i$$

$$\boxed{\frac{\partial E}{\partial m_j} = -\frac{1}{m} \sum \left(y^i - h(x_i)\right) x_j^i}$$

Summary $h(x) = \dfrac{1}{1+e^{-m^T x}}$

$$\boxed{E(h(x^i), y) = -\frac{1}{m} \sum \{ y \log(h(x^i)) + (1-y) \log(1 - h(x^i)) \}}$$

$$m_j := m_j - \alpha \frac{\partial E}{\partial m_j}$$

$$\frac{\partial E}{\partial m_j} = -\frac{1}{m} \sum \left(y^i - h(x_i)\right) x_j^i$$

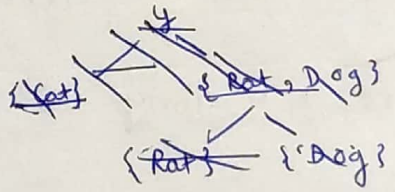
$h(x)$ is other form of Probability (P).

$h(x) \Rightarrow P(y = 1)$

Multiclass (one vs rest)

let $y_p = [\text{cat}, \text{Rat}, \text{Dog}]$

we can do



$y_1 = [\{\text{Cat}\}, \{\text{R}, \text{D}\}]$

$y_2 = [\{\text{R}\}, \{\text{C}, \text{D}\}]$

$y_3 = [\{\text{D}\}, \{\text{C}, \text{R}\}]$

So we get $P(y_1)$, $P(y_2)$, $P(y_3)$

Probability estimates, 3 models are independent.

Multinomial assume we have $\{1\ 2 \ldots k\}$ classes.

$$P(y = j) = \frac{e^{m_j x}}{\sum_{i=1}^{k} e^{m_i x}}$$

we are training $n * k$ parameters

function in sklearn model.

fit(), predict(), score(), predict_proba()

(mean accuracy)

Illy we can add dumy features to get complex boundary.

But overfitting, F!

Regularisation: "I'm gonna save you"

$$\text{cost}' = \text{cost} + \lambda \sum f(m_i)$$

$\lambda = 0$ → high λ → underfitting

$$L1 = \text{cost} + \lambda \sum |m_i| \quad \text{lasso}$$

$$L2 = \text{cost} + \lambda \sum m_i^2 \quad \text{Ridge}$$

* No regularisation on intercept.

Performance measures in classification

sklearn.metrics → confusion matrix

(1) confusion matrix

Predicted \ Actual	T	F
T	TP	FP
F	FN	TN

$$TPR = \frac{TP}{TP+FN}$$

$$FPR = \frac{FP}{FP+TN}$$

For a class (+)

$$\text{Accuracy} = \frac{TP + TN}{TP+FN+TN+FP}$$

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN} = TPR = \text{Sensitivity}$$

$$\text{Sensitivity} = \frac{TP}{TP+FN} = TPR \quad \text{(ability to detect +ve class)}$$

$$\text{Specivity} = \frac{TN}{TN+FP} = 1 - FPR.$$

Decision Tree: (You know Already)

How do we split? May be accuracy, gain ratio etc. Gini index, information gain

Greedy Algo.

→ If node is pure : make it leaf.
→ If no feature is left : output majority.
→ Find best feature to split on
→ Make recursive call.

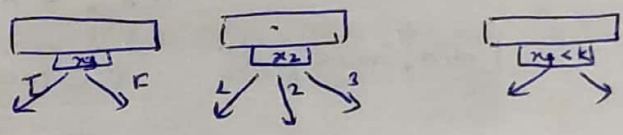
(i) Accuracy → find accuracy of Parent, try splitting on every valid possible feature and get Children's accuracy.
→ Choose feature if $\sum$ children's (for that feature) > Parent's accuracy
→ If none, make parent a leaf.

* Continuous value features:

We decide some decision line say $x < 5$ $x \geq 5$

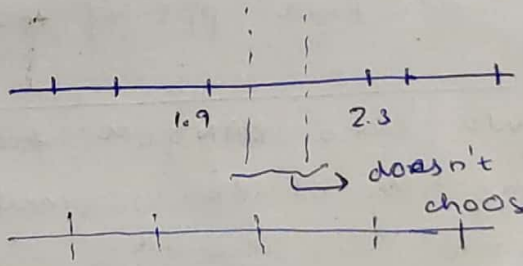
Say data is:

x_1	x_2	x_3
continuous	discrete	boolean



How we decide k?

↳ sort the data
→ get the midpoints



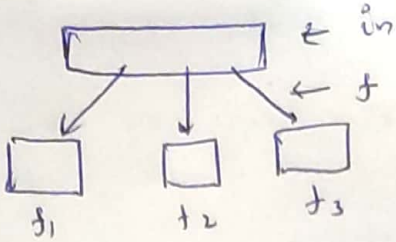
→ find accuracy or other matrix for all mid points.
and pick a mid point to make it as decision boundary.

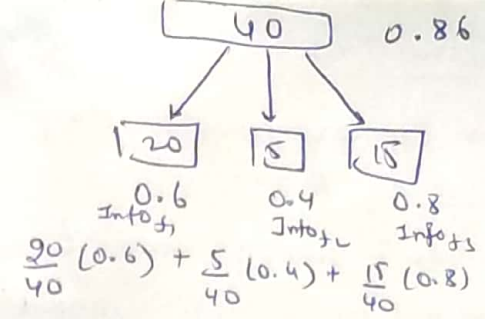
* Information Gain:

Entropy / Info req : $-\sum_{\text{classes}} p_i \log p_i$

We need minimum randomness / Entropy / Info req to reduce impurities

→ we chose feature which decrease Entropy by max.





Info gain = $Info_N() - Info_F()$

$Info_F() = \sum_{f_i} \frac{|D_i|}{|D|} Info_{D_i}()$

Pick feature which given better info gain

* better see an example.

We will be back, a little confusion in this topic

Feature scaling / Data Preprocessing

↳ scaling different feature (with variety of range) under some range.

(1) Min_Max scaling $[0, 1] = \frac{x - x_{min}}{x_{max} - x_{min}} = p$

$[min, max] = p(max - min) + min$

(2) make mean = 0
variance = 1

Read ppt of maam

Random Forest: We take RANDOM training data and features and build many trees (FOREST), result given by majority of trees is taken in account.

① Bagging → Bootstrap Aggregation Algo.

If we have m data points, we select k points with replacement. $\sum k_i = m$
Means multiple occurence of single point is there.

② Feature Selection: k feature from m feature without replacement (generally $k = \sqrt{n}$)

We can solve regression problems using decision tree, split on basis of which feature reduces the MSE and leaf node contains mean value.

Bayes Theorem : Naive Bayes :

Bayes Theorem : $P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$

$y = a_1, a_2, \ldots a_k$ (output classes)

then $Y = \max_{i=1}^{k} \left(P(y = a_i \mid x = x) \right)$

$= \max_{i=1}^{k} \left(\frac{P(x = x \mid y = a_i)\; P(y = a_i)}{P(x = x)} \right)$

$$\boxed{Y = \max_{i=1}^{k} \left(P(X = x \mid y = a_i)\; P(y = a_i) \right)}$$

Y ↓ output class

$$P(y = a_i) = \frac{|a_i|}{|y|} = \frac{|a_i|}{|a_1| + |a_2| \ldots |a_k|}$$

$P(X = x \mid y = a_i)$

Independent Events : $P(A \cap B) = P(A) * P(B)$

Given input $x \rightarrow \{x_1, x_2 \ldots x_k\}$ where x_i is value corresponding to feature f_i.

$$P(X = x \mid y = a_i) = P((x_1, x_2, x_3 \ldots x_k) \mid y = a_i)$$

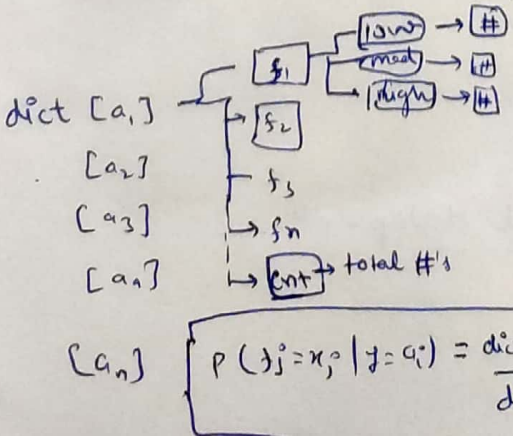
(f_1 f_2 f_3 f_k)

* Assumption → All features are independent.

$$= P(f_i = x_i \mid y = a_i) * P(f_2 = x_2 \mid y = a_i) \ldots P(f_k = x_k \mid y = a_i)$$

$$\boxed{P(X = x \mid y = a_i) = \prod_{j=1}^{k} P(f_j = x_j \mid y = a_i)}$$

$$P(f_j = x_j \mid y = a_i) = \frac{\#(f_j = x_j \text{ \& } y = a_i)}{\#(y = a_i)}$$



$$P(f_j = x_j \mid y = a_i) = \frac{dict[a_i][f_j][x_j]}{dict[a_i][cnt]}$$

Given data X_train Y_train

f_1	f_2	f_3	...	f_k	Label
low					a_1
med					a_2
hig					a_1
					a_5
					a_n

Laplace correction: $P(\ )$ can be 0 if there are no value (x_j) present in feature (F_j) given a a_i
So we modify formula

$$= \frac{dict[a_j][F_j][x_j] + 1}{dict[a_j][cnt] + |F_j|}$$

↳ values F_j can take

Continuous data?

$$P(f_j = x_j \mid y = a_i) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\left(-\frac{(x_j - \mu)^2}{2\sigma^2}\right)}$$

Naive_bayes can be used as text collection

KNN → K-Nearest Neighbour | Do feature scaling before KNN
You know what It is!

Cross validation:
Low value of k leads to overfitting
High value of k leads to underfitting
Make sure we have relevant data and independent.
→ Assign weights to feature (w_i for f_i)

$$\sum_{i=1}^{n} w_i (x_1^i - x_2^i)^2$$

→ Feature selection (Backward elimination)

Handling labeled data:
→ use 0, 1 in binary data. (2 values)
→ we can't assign 0, 1, 2, 3 .. to discrete data bcz distance would be effected.

	Red	Blue	Green
Red	0	0	1
Blue	0	0	1
Green	1	0	0
	0	1	0

→ **Other KNNs**
→ Brute force → compare with all datapoints
→ KD Trees → something like BST
→ Ball Trees

Pros of KNN → Easy to understand and code (lol)

(2) Works for multiclassification

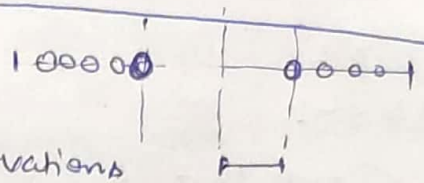
cons of KNN:

(1) Testing time is huge.

(2) It can be biased, if training data is biased.

(3) Curse of dimensionality.

SVM



The shortest distance b/w the observations and the threshold is called margin

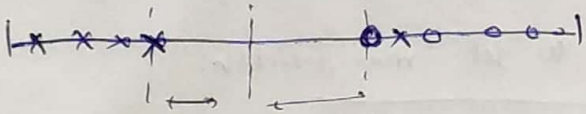
Maximal margin classifier (MMC): threshold that gives us the largest margin.

MMC is not optimal (think of outliers)

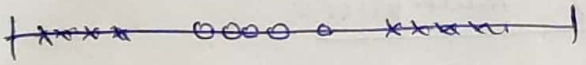
We do cross validation to find the best soft margin aka support vector classifier.

The suppor vector classifier is a hyperplane in n-dimension.

So SVC can handle outliers nicely using cross validation



But what about



so many overlaps...!

Support Vector machine: We add extra dimensions to points and then try to fit hyperplane, ~~sup~~ SVC.

~~How~~ How to transform the data? SVM uses kernels function

We can find good value of (d) dimension via cross validation

Other kernal is RBF (finds SVM in infinite direction).

Kernels just calculate the r/n b/w points in high dimension without actually trasforming them.

let $r = 1/2$ $d = 2$

Polynomial kernel $= (a \times b + r)^d$

$= ab + a^2b^2 + 1/4$

$= (a, a^2, 1/2), (b, b^2, 1/2)$

$\downarrow x \quad \downarrow y \quad \downarrow z$

MIT OCW

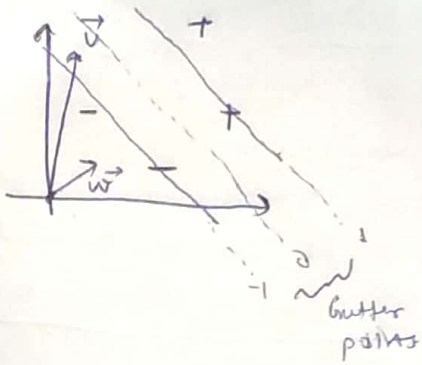
$\vec{w}$ is ⊥ to median

$\vec{u}$ be unknown sample point

$\vec{w} \cdot \vec{u} \geq c$

i.e $\vec{w} \cdot \vec{u} + b \geq 0$ for + sample

but we still dont know $\vec{w}$ and b.



$\vec{w} \cdot \vec{x}_+ + b \geq 1 \quad \ldots (1)$

$\vec{w} \cdot \vec{x}_- + b \leq -1 \quad \ldots (2)$

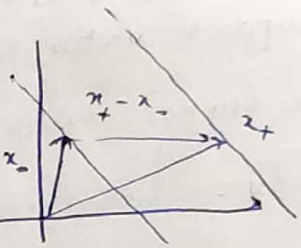
introduce y_i such that $y_i = 1$ +ve sample, -1 -ve sample

eq 1 and 2 become

$$y_i (\vec{x}_i \cdot \vec{w} + b) \geq 1$$

$$y_i (x_i \vec{w} + b) - 1 \geq 0$$

$$y_i (x_i \vec{w} + b) - 1 = 0 \rightarrow \text{For Gutter Points.}$$



$$\text{width} = (x_+ - x_-) \cdot \frac{\vec{w}}{\|w\|}$$

$$= \frac{x_+ \cdot \vec{w}}{\|w\|} - \frac{x_- \vec{w}}{\|w\|} = \frac{1-b + 1+b}{\|w\|} = \frac{2}{\|w\|}$$

we are maximising $\frac{1}{\|w\|}$ to get max width.

minimise $\|w\| \sim$ minimise $\frac{1}{2}\|w\|^2$

we have to use Langrange multipliers.

$$L = \frac{1}{2}\|w\|^2 - \sum \alpha_i [y_i (\vec{w} \cdot \vec{x}_i + b) - 1] \quad -(3)$$

$$\frac{\partial L}{\partial w} = \vec{w} - \sum \alpha_i y_i x_i = 0 \rightarrow \vec{w} = \sum_i \alpha_i y_i \vec{x}_i$$

$$\frac{\partial L}{\partial b} = \sum \alpha_i y_i = 0$$

substitute in eq 3.

$$L = \frac{1}{2}\left(\sum_i \alpha_i y_i \vec{x}_i\right) \cdot \left(\sum_j \alpha_j y_j \vec{x}_j\right) - \sum \alpha_i [y_i$$

$$L = \frac{1}{2}\left(\sum_i \alpha_i y_i \vec{x}_i\right)\left(\sum_j \alpha_j y_j \vec{x}_j\right) - \left(\sum \alpha_i y_i x_i\right)\left(\sum \alpha_j y_j \vec{x}_j\right) + \sum \alpha_i y_i b + \sum \alpha_i$$

($\sum \alpha_i y_i b = 0$)

$$L = \sum \alpha_i - \frac{1}{2} \sum_i \sum_j (\alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)) \rightarrow \text{Max of this.} \quad (4)$$

depend on dot product.

$\sum \alpha_i y_i \underbrace{\vec{x_i} \cdot \vec{u}}_{\text{dot product}} + b \geq 0$ +ve sample.

Won't work with non linear decision boundary.

transformation : $\phi(\vec{x}) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$ to max

$\phi(x_i) \cdot \phi(u)$

$K(x_i, x_j) = \phi(\vec{x_i}) \cdot \phi(\vec{x_j})$

↳ kernel function. we actually don't have to do ~~dot~~ transform

so ~~$\phi(\vec{x})$~~ , all depends on dot product.

some kernels : (1) $(\vec{u} \cdot \vec{v} + r)^d$ Polynomial kernel.

(2) $e^{-\frac{\|x_i - x_j\|}{2\sigma^2}}$ Radial basis kernel

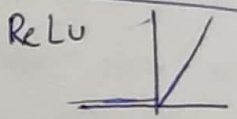
Neural Network

Activation Function → sigmoid f^n , ReLU, Soft Plus.

We get x-axis coordinate from the weight & and use activation func to get Y.

Weights come from Activation Function.

Act. (weight + biases) → new perception.



$f(x) = x \quad x \geq 0$

$= 0 \quad x < 0$

$f(x) = \max(0, x)$.

Maam PPT's after mid sem:

KNN → pattern recognition what k to chose? $k < \sqrt{n}$

→ While scaling we can add weight to the important features. w_k we can learn w_k using cross-validation.

Ex:)

Kmeans Algo: (Unsupervised learning) Stat quest PPT

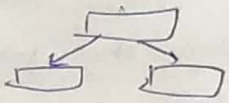
(1) Select a k (clusters to identify) (k=3)
(2) Randomly select k distinct dp. (~~centroids~~)
(3) Measure distance of remaining points to clusters and classify them to nearest cluster.
(4) Calculate mean of each cluster ← (centroid)
(5) Repeat. (Measure distance along the mean this time)
(6) ~~Read~~ When it does show its result and calculate the variance.

centroid can be real or imaginary dp

Repeat whole with different k and diff. starting points.
The best is where we have a gradual change in ~~mean~~ variance.

AdaBoost → contains stumps

resource: Stat Quest PPT



The tree with just one node and two leaves is stump.

~~Multiple~~ Forest of stumps.

A tree uses are features to make decision, but a stump would take one feature to make decision.
So 'stumps' are weak learners.

In RF, each tree has equal vote to classification
In Adaboost, each stump has different vote to classification.

In RF, each tree is independent.
In Adaboost, order is important, The error of first stump influences how second stump is made and soon.

Steps

~~(1) Make stumps of~~

(1) Assign sample weight (this will tell which dp is important)
(initially for all = 1/no. of samples)

(2) Make stumps using each feature and calculate gini index. The one with lowest would be the first stump.

(4) Now we find ... vote ... error).

Amount of vote (say) $= \frac{1}{2} \log\left(\frac{1 - \text{Total Error}}{\text{Total Error}}\right)$ Σ # of incorrectly classified with their sample weight.

(5) Now we modify sample weight.

we increase the sample weight of incorrectly classified weight of that feature, so that next time we try to classify it correctly and decrease all other sample weights.

↑ Net sample weight = sample weight × $e^{\text{amount of vote}}$

↓ sa. wt × $e^{-\text{amount of vote}}$

and calculate new sample weight.

and find rest of stumps.

Ensemble Technique → combining multiple models.

Need to add from Main

→ Bagging : Random Forest
(Bootstrap aggregation)

→ Boosting : ADA BOOST, GRADIENT BOOSTING, XgBoost.

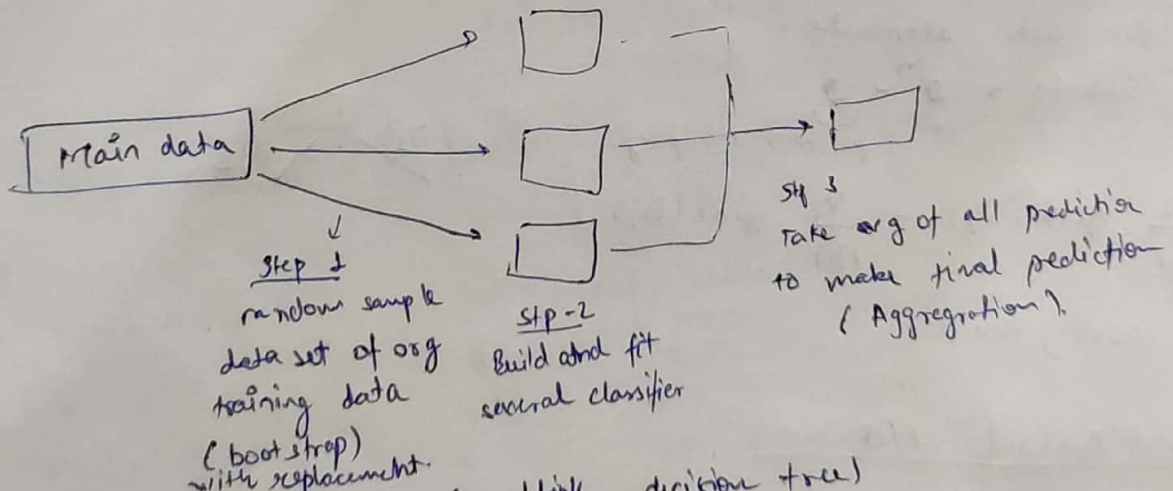
Error → Bias + variance.

using Bagging we can decrease variance

using Boosting ——— bias

simplilearn
sklearn
PPT

Bagging :→ bootstrap aggregation reduces variance of an estimate by taking mean of multiple estimates.



eg Random forest (multiple decision tree)

Boosting reduces bias by training weak learners sequentially, each trying to correct predecessor.

eg: → ADA BOOST (multiple stumps)
→ Gradient Boosting (GBM)
→ XGBoost

CART Algo [Classification and Regression Tree]

~~ID3~~ ~~Gain ratio~~

$$GINI(D) = 1 - \sum_{i=1}^{m} p_i^2$$

→ Binary Tree
→ Uses GINI Index

→ If a dat set D examples from n classes,

$$gini(D) = 1 - \sum_{i=1}^{n} p_i^2.$$

→ If a data set D is split on A (Attribute) into two subsets D_1 and D_2

$$gini_A(D) \rightarrow \frac{|D_1|}{|D|} gini(D_1) + \frac{|D_2|}{|D|} gini(D_2)$$

* Reduction in impurity:

$$\Delta gini(A) = gini(D) - gini_A(D)$$

The one has minimum Gini index, is selected as the splitting attribute.

(1) Calculate gini of whole Dataset.

$$Gini(D) = 1 - \sum p_i^2$$

(2) Calculate for each attribute.

→ Total subsets → $2^n - 2$ (power + empty).

$$\frac{D_1}{|D|} gini(D_1) + \frac{D_2}{|D|} gini(D_2)$$

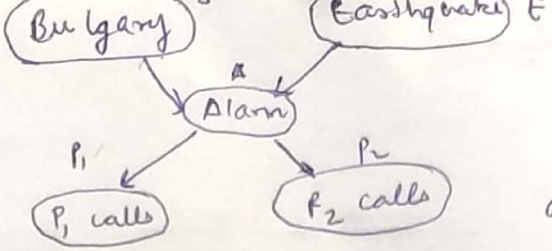
and keep doing this.

→ Bayesian Belief Network

(1) Directed acyclic graph
(2) Conditional Probability Table.

convenient for representing probabilistic relation b/w multiple events.

Ex



Given $P(B=T) = 0.001$
$P(B=F) = 0.999$
$P(E=T) = 0.002$
$P(E=F) = 0.998$
and other poopebility

see video 5 minute engineering.

* Rule Based classification.

'IF' 'AND' 'THEN'

Rules are extracted using these three keywords.

eg: IF outlook = 'sunny' AND
Humidity = 'High'
THEN Play = 'NO'.

* Association Rule Mining (Market-Basket Analysis)

~~color~~ endeseba!

A ⇒ B
If Then

3 Metrics which help in Association
(1) support (2) confidence (3) lift

$$\text{support} = \frac{\text{freq}(A, B)}{N}$$

$$\text{confidence} = \frac{\text{freq}(A, B)}{\text{freq}(A)}$$

To turn rules into association rules

How frequent both items are bought together.

$$\text{Lift} = \frac{\text{support}}{\text{supp}(A) \times \text{supp}(B)}$$

where A, B occurs independently

eg:

let transactions be

T_1	A	B	C
T_2	A	C	D
T_3	B	C	D
T_4	A	D	E
T_5	B	C	E

lets create rules.

	support	confidence	lift
A → D	2/5	2/3	10/9
C → A	2/5	2/4	5/6
A → C	2/5	2/3	5/6
B & C → A	1/5	1/3	5/9

Aprion Algorithm: uses frequent item set to generate ~~item sets~~. association rules. It is based on concept that a subset of a frequent itemset must also be frequent itemset.

Frequent item set is an itemset whose support value is greater than a threshold value.

Eg.

TID	Items
T1	1 3 4
T2	2 3 5
T3	1 2 3 5
T4	2 5
T5	1 3 5

→

Itemset	Support
{1}	3
{2}	3
{3}	4
~~{4}~~	~~1~~
{5}	4

1st Iteration

→

Itemset	Support
~~{1,2}~~	~~1~~
{1,3}	3
{1,5}	2
{2,3}	2
{2,5}	3
{3,5}	3

→

Itemset	support
{1,3,5}	2
{2,3,5}	2

↓ ↑

~~{1,2,3,5}~~	~~1~~

2nd iteration.

We can do some pruning

for eg: we can discard whose data set containing {1,2}

eg = {1,2,3} {1,2}, {1,5}, {2,5}. (Spurious Eng)

so Frequent Items are {1,3,5}, {2,3,5}

Now ~~who~~ check their subset

Eg for I = {1,3,5} → {1} {3} {5} {1,3} {1,5} {3,5}.

For every subsets of I, output the rule:

S → (I − S) i.e (S recommends I−S)

if support(I) / support(S) >= min_conf value.

so we check which rule is selected

say {1,3} → {{1,3,5} − {1,3}} [1 & 3 → 5]

✓ = support(1,3,5) / support(1,3) = 2/3 > 60%.

~~Eg~~

{1} → {1,3,5} − {1} [1 → 3 & 5]

✓ = sup(1 3 5) / sup(1) = 2/3 > 60%.

{5} → {1,3,5} − {5} (5 → 1 & 3)

✗ supp(1,3,5) / sup(5) = 2/4 < 60%.

Frequent Patterns:

eg { milk, bread } , { computer, Antivirus }

type: Itemset , Sequential

FP Tree Construction:

min support → 2

TID	Items
1	b d c a
2	e d c
3	a b
4	a c d
5	f g d b

→

Item	Support
d	4
a	3
b	3
c	3
e	1
f	1
g	1

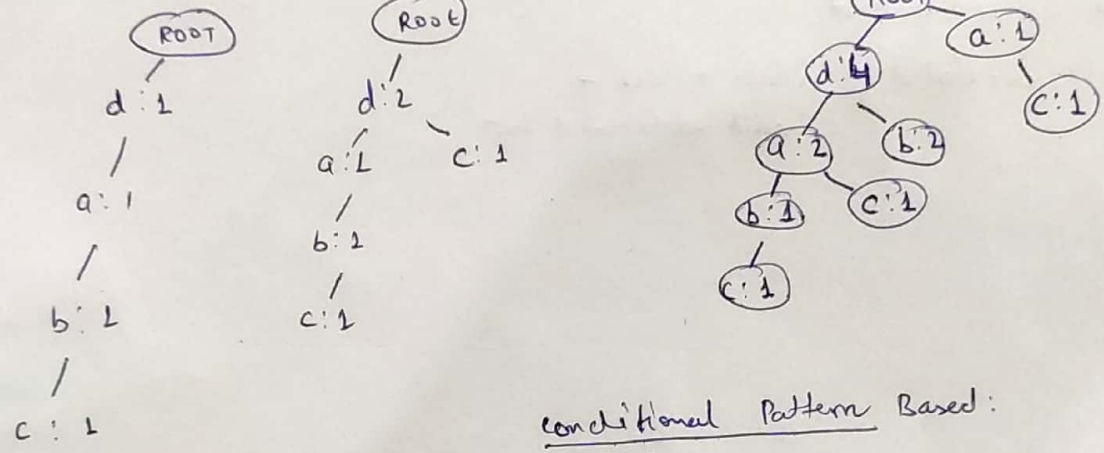
↓ Priority

ordered frequent items

TID	Items
1	d a b c
2	d c a
3	a b
4	d a c
5	d b

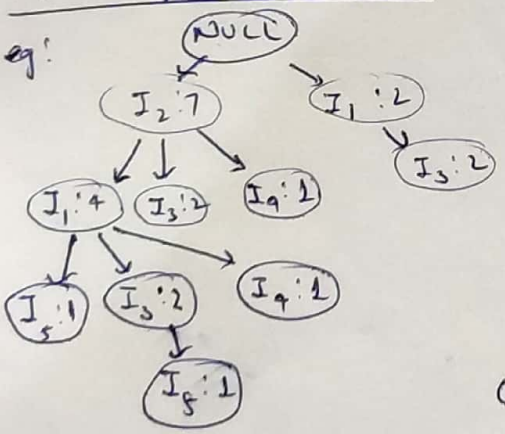
construct FP tree

Transition 1 T2 ..... so on T5



FP tree growth Algorithm:

eg:



I_5 : 2

I_4 : 2

I_3 : 6

I_1 : 6

I_2 is ignored as its connected to root

Conditional Pattern Based:

For I_5:

$\{I_2, I_1 : 1\}$ $\{I_2 I_1 I_3 : 1\}$

For I_4

$\{I_2 I_1 : 1\}$ $\{I_2 : 1\}$

For I_3

$\{I_2 I_1 : 2\}$ $\{I_2 : 2\}$ $\{I_1 : 2\}$

For I_1

$\{I_2 : 4\}$

Conditional FP-Tree.

Find common prefix

min support = 2

For I_5:

$\{I_2 : 2, I_1 : 2\}$ ~~$\{I_5 : 1\}$~~

For I_4

$\{I_2 : 2\}$ ~~$\{I_1 : 1\}$~~

For I_3

$\{I_2 : 4, I_1 : 2\}$ $\{I_1 : 2\}$

For I_1

$\{I_2 : 4\}$

Frequent Pattern Generation!

Sets to vapas add karde

eg I_5 I_2 I_1 I_5 → $\{I_2, I_5 : 2\}$
$\{I_1, I_5 : 2\}$
$\{I_2, I_1, I_5 : 2\}$

See video! 5 minute explanation.

LIFT and χ^2

$lift(A,B) = 1$ A, B are independent
> 1 positive correlated
< 1 -ve correlated

$$lift(A,B) = \frac{confidence(A \rightarrow B)}{support(B)}$$

$$= \frac{support(A \cup B)}{support(A) \times support(B)}$$

$$= \frac{P(A \cup B)}{P(A)\,P(B)}$$

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

$\chi^2 = 0$: independent
> 0 correlated, either + or -
need additional test