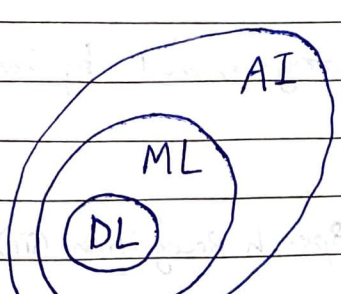
ML



What is ML ?

→ Branch of AI and CS which focuses on use of data and algos to imitate the way that humans learn, gradually improving its accuracy.

ML

Supervised
→ Uses labeled datasets to train algos that to classify data or predict outcomes accurately.

Unsupervised
→ Uses unlabeled data
→ From that data, it discovers patterns that help solve for clustering or association problems.

Semi supervised
→ When part of input data is labeled, it is semi supervised

Reinforcement
→ Taking suitable action to maximize reward in a particular situation.
→ There is no answer in dataset but the reinforcement agent decides what to do to perform given task.

Applications

NLP
- Autonomous tagging of stackoverflow ques
- Automated essay grading
- Sentence to sentence semantic similarity
- Fight online abuse
- Open domain question answering
- Social chat / conversational bots

Dataset
- StackLite
- Essays with human graded scores
- Quora ques pairs with similar ques marked
- Toxic comments on Kaggle
- NCERT books for K-12 / School students in India
- Reddit dataset

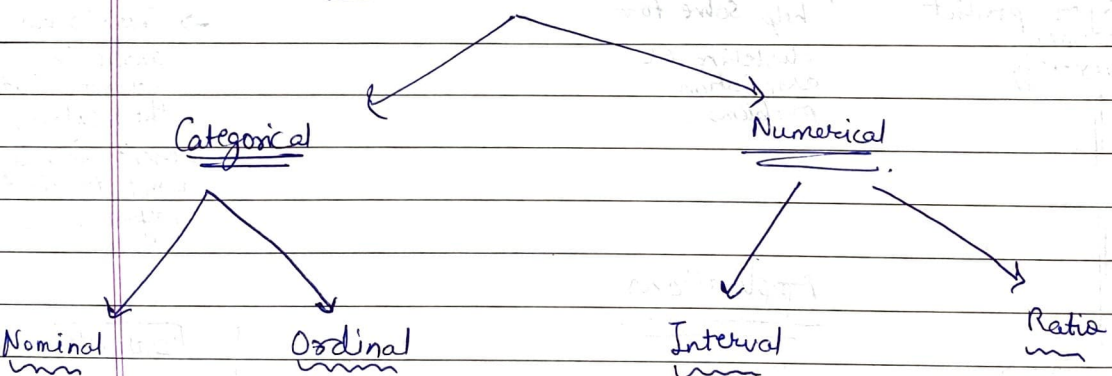
* Applications that can't be programmed by hand :-

-> Handwriting Recognition
-> NLP : (i) Chatbot (ii) Speech Recognition (iii) Text summarization
-> Part-of-Speech Tagging
-> Sentiment Analysis

Many more applications in PPT (i)

i) Personalised websites
ii) medical domain
iii) Finance
iv) Intrusion detection system
v) Robotics
vi) . . . . .

DATA TYPES



-> No order
-> No quantitative value

eg. Gender
* Male
* Female

-> Ordering matters
-> Non numeric features

eg. Good
Better
Best

- **Quantitative**
  - **Discrete**
    - → Can't be measured but can be counted
    - eg. No. of heads in 100 coin flips.
  - **Continuous** → can't be counted but can be measured
    - eg. height of person
    - **Interval**
      - → ordered units that have same diff.
      - eg. Temp. of given place
    - **Ratio**
      - → Same as interval values with difference that they do have an absolute zero
      - eg. height, weight, length
- **Qualitative**
  - **Nominal**
  - **Ordinal**

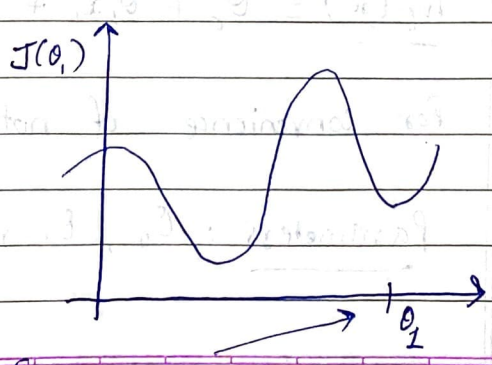
Simple Linear Regression

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x$$

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

$\Rightarrow$ Cost function, $J(\theta_0, \theta_1) = \left(\frac{1}{2m}\right) \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$

$\Rightarrow$ Minimize $J(\theta_0, \theta_1)$
θ_0, θ_1



Gradient Descent Algorithm:

$$\theta_1 := \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_1)$$

$\theta_1 \rightarrow$ Parameters

$\alpha \rightarrow$ Learning rate.

Too small ⇓ GD can be slow

Too large ⇓ GD can overshoot minimum. It may fail to converge or even diverge

repeat until convergence {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \cdot x^{(i)}$$

} update θ_0 and θ_1 simultaneously

Multiple Linear Regression

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n = \theta^T x$$

For convenience of notation, define $x_0 = 1$.

Parameters : $\theta_0, \theta_1, \dots, \theta_n$

Cost function :

$$J(\theta_0, \theta_1, \cdots, \theta_n) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Gradient descent :

Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \cdots, \theta_n)$$

} (simultaneously update for every $j = 0, \cdots, n$)

New algo ($n \geq 1$) :

Repeat {

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right) x_j^{(i)}$$

} (simultaneously update θ_j for $j = 0, \cdots, n$)

Feature Scaling

Get every feature into approximately $-1 \leq x_i \leq 1$ range

(*) Min Max normalization

$$v' = \frac{v - min_A}{max_A - min_A} (newMax_A - newMin_A) + newMin_A$$

→ For mapping to [0, 1] range, formula becomes

$$v' = \frac{v - min_A}{max_A - min_A} (1 - 0) + 0$$

$$\boxed{v' = \frac{v - min_A}{max_A - min_A}}$$

(*) Z-score normalization

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

$$\boxed{v' = \frac{v - \bar{A}}{\sigma_A}}$$

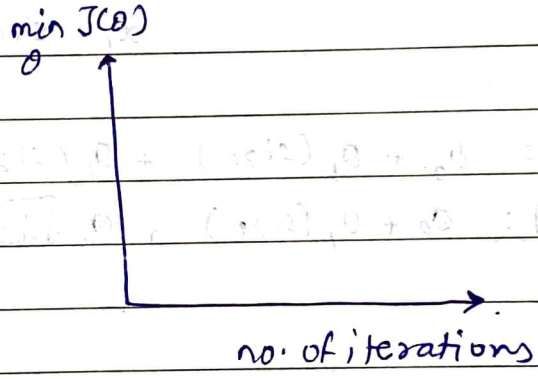
Learning rate

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

"Debugging" : How to make sure gradient descent is working correctly

How to choose learning rate α. ?

→ Declare convergence if $J(\theta)$ decreases by less than 10^{-3} in one iteration.



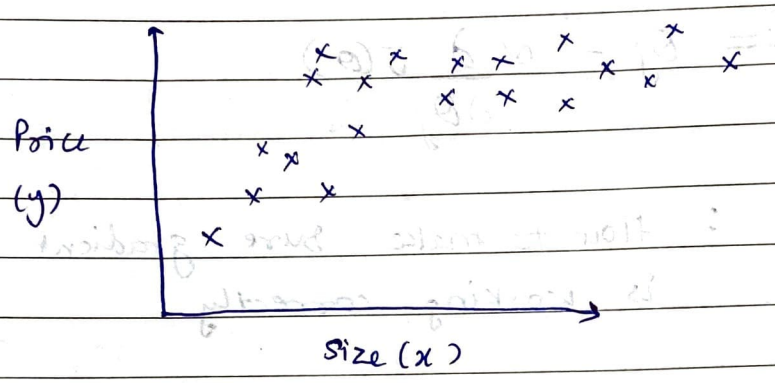
GD not working ⇒ use smaller α

⇓

For sufficiently small α, $J(\theta)$ should decrease on every iteration.

But if α is too small, G.D can be slow to converge.
if α is too large, $J(\theta)$ may not decrease on every iteration; may not converge

Features & Polynomical regression



$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$
$$= \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2 + \theta_3(\text{size})^3$$

$x_1 = \text{size}$; $x_2 = (\text{size})^2$; $x_3 = (\text{size})^3$

$$h_\theta(x) = \theta_0 + \theta_1(\text{size}) + \theta_2(\text{size})^2$$
$$h_\theta(x) = \theta_0 + \theta_1(\text{size}) + \theta_2\sqrt{\text{size}}$$

Normal Equation

Q.

x_0	x_1	x_2	x_3	x_4	y
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$X = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \qquad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$\theta = (X^T X)^{-1} X^T y$$

training examples → m

features → n

G.D	N.E
→ Need to choose α	→ No need to choose α
→ Needs many iterations	→ Don't need to iterate
→ Works well even when n is large	→ Need to compute $(X^T X)^{-1}$
	→ Slow if n is very large

What if $X^T X$ is non-invertible ?

→ Redundant features (linearly dependent)

eg. x_1 = size in feet^2

x_2 = size in m^2

→ Too many features (eg. $m \leq n$)

* Delete some features, or use regularization

Eqn of least squares line

$$y = w_0 + w_1 x$$

Here, $w_0 = \bar{y} - w_1 \bar{x}$

$$w_1 = \frac{\sum_{i=1}^{|D|} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{|D|} (x_i - \bar{x})^2}$$

Error measures

Absolute error : $|y_i - y_i'|$

Squared error : $(y_i - y_i')^2$

Mean absolute error : $\frac{\sum_{i=1}^{d} |y_i - y_i'|}{d}$

Mean absolute % error : $\frac{1}{d} \sum_{i=1}^{d} \left| \frac{y_i - y_i'}{y_i} \right|$

RMSE : $\sqrt{\sum_{i=1}^{d} \frac{(y_i - y_i')^2}{d}}$

NRMSE = $\frac{RMSE}{y_{max} - y_{min}}$

For Normal eqn :-

$$\theta = (X^T X + \lambda \cdot L)^{-1} X^T y$$

where

$$L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$ is square matrix of size $(n+1)$

Overfitting

→ If we have too many features, learned hypothesis may fit training set very well (i.e. $J(\theta) \simeq 0$), but fail to generalize to new examples

Options :- (i) Reduce no. of features
- Manually select features to be kept
- Model selection algo

(ii) Regularization
- Keep all features, but reduce magnitude of parameters θ_j.
- Works well when we have a lot of features, each of which contributes a bit to predicting y.

Regularization

Small values of parameters $\theta_0, \theta_1, \ldots, \theta_n$
- "Simpler" hypothesis
- Less prone to overfitting

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

For regularized Linear Regression, $\min_\theta J(\theta)$

G.D :- repeat until convergence {

$$\theta_0 := \theta_0 - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_0^{(i)}\right]$$

$$\theta_j := \theta_j\left(1 - \alpha\frac{\lambda}{m}\right) - \alpha\frac{1}{m}\left[\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}\right]$$

where $j \in \{1, 2, \ldots, n\}$

}

Logistic Regression

$h_\theta(x) = g(\theta^T x)$, where $g(z) = \frac{1}{1+e^{-z}}$

$\therefore$ ~~$h(\theta)$~~ $\boxed{h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}}$

$h_\theta(x)$ = estimated probability that $y = 1$ on input x

Accuracy Measures

Validation techniques

- → Resubstituting
- → Hold-out
- → K fold cross validation.
- → Leave one out cross validation
- → Random Subsampling.
- → Bootstrapping.

Confusion matrix

Actual class / Predicted class	C_1	$\sim C_1$
C_1	TP	FN
$\sim C_1$	FP	TN

Accuracy $= \frac{(TP + TN)}{All}$

Error rate $= 1 - \text{accuracy}$

$= \frac{(FP + FN)}{All}$

Sensitivity (TP rate) $= \frac{TP}{P}$

Specificity (TN rate) $= \frac{TN}{N\ (TN + FP)}$

$1 - \text{Specificity}$ (FP rate) $= \frac{FP}{N}$

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

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

F-score, $F = \frac{2\,(Precision)\,(Recall)}{Precision + Recall}$.

$F_\beta = \frac{(1+\beta^2)\,(Precision)\,(Recall)}{\beta^2 \times Precision + recall}$

Additional aspects for model selection

→ **Accuracy** — classifier accuracy : predicting class label

→ **Speed** — Training time ; prediction time

→ **Robustness** — handling noise & missing values

→ **Scalability** — efficiency in disk-resident databases

→ **Interpretability** — understanding & insight provided by model

Total Probability theorem :-

$$P(B) = \sum_{i=1}^{M} P(B/A_i)\,P(A_i)$$

Bayes' Theorem :-

$$P(H/X) = \frac{P(X/H)\cdot P(H)}{P(X)} = P(X/H) \times \frac{P(H)}{P(X)}$$

$P(H)$: Prior probability

$P(H/X)$: posterior probability

Gaussian Naive Bayes

$$P(C_i/X) = \frac{P(X/C_i)\,P(C_i)}{P(X)}$$

Now since $P(X)$ is constant for all classes,

$$\boxed{P(C_i/X) = P(X/C_i)\,P(C_i)}$$ needs to be maximised.

Now $$P(X/C_i) = \prod_{k=1}^{n} P(x_k/C_i)$$

$$= P(x_1/C_i) \times P(x_2/C_i) \times \dots \times P(x_n/C_i)$$

Now if A_k is categorical $\Rightarrow P(x_k / C_i)$

$$= \frac{\#\text{ of tuples in } C_i \text{ having value } x_k \text{ for } A_k}{\#\text{ of tuples of } C_i \text{ in } D}$$

If A_k is continous $\Rightarrow P(X/C_i) = g(x_k, \mu_{C_i}, \sigma_{C_i})$

where $g(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\,\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$

(*) In order to avoid zero probability problem, add Laplace correction i.e. add 1 to each case

(*) For a remedy of zero conditional probability problem, conditional probabilities estimated with (m-estimates) are used.

$$\hat{P}(X_j = a_{jk} / C = C_i) = \frac{n_c + mp}{n + m}$$

$n_c \rightarrow$ no. of tuples for which $X_j = a_{jk}$ & $C = C_i$

$n \rightarrow$ no. of tuples for which $C = C_i$

$p \rightarrow$ prior estimate [usually, $p = 1/t$ for t possible values of X_j]

$m \rightarrow$ weight to prior (no. of "virtual" examples, $m \geq 1$)

Multinomial Naive Bayes'

$$\hat{P}(x_i/\omega_j) = \frac{\sum tf(x_i, d \in \omega_j) + \alpha}{\sum N_{d\in\omega_j} + \alpha V}$$

where,

$x_i \rightarrow$ Word from feature vector x of particular sample

$\sum tf(x_i, d\in\omega_j) \rightarrow$ sum of raw term freq. of word x_i from all documents belonging to class ω_j.

$\sum N_{d\in\omega_j} \rightarrow$ Sum of all term frequencies in training dataset for class ω_j.

$\alpha \rightarrow$ An additive smoothing parameter ($\alpha = 1$ for laplace smoothing)

$V \rightarrow$ no. of different words in training set.

Multivariate Bernoulli Naive Bayes'

$$P(X/\omega_j) = \prod_{i=1}^{m} P(x_i/\omega_j)^b \cdot (1 - P(x_i/\omega_j))^{(1-b)} \qquad (b \in 0, 1)$$

Now, $$\hat{P}(x_i/\omega_j) = \frac{df_{x_i,y} + 1}{df_y + 2}$$

where,

$df_{x_i,y} \rightarrow$ no. of documents that contain x_i and belong to class ω_j

$df_y \rightarrow$ no. of documents belong to class ω_j.

$+1$ and $+2 \rightarrow$ Laplace smoothing.

Doubt Why formula in theory and sum solving are diff?

Formula in problem solving :-

$$P(S/b_i) \propto P(S) \prod_{i=1}^{8} [b_{fi} P(w_i/S) + (1-b_{ii})(1-P(w_i/S))]$$

DECISION TREES

ID3 Algorithm

→ Select attribute that has highest information gain

→ $\text{Entropy}(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$ where p_i = proportion of S belonging to class i

→ For collection S having +ve and -ve examples, entropy is given by,

$$E(S) = -p^+ \log_2 p^+ - p^- \log_2 p^-$$

where p^+ is proportion of +ve examples,
p^- is proportion of -ve examples

→ $E(S) = 0$ if all members belong to same class
$E(S) = 1$ if all members are split equally

Information gain for feature column A is calculated as:

$$IG(S, A) = Entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} \times Entropy(S_v)$$

$S_v \rightarrow$ set of rows in S for which feature column A has value v

$|S_v| \rightarrow$ no. of rows in S_v

$|S| \rightarrow$ no. of rows in S

C4.5 algorithm

$$SplitInfo_A(D) = -\sum_{j=1}^{v} \frac{|D_j|}{|D|} \log_2 \frac{|D_j|}{|D|}$$

$$Gain\ Ratio(A) = \frac{Gain(A)}{SplitInfo(A)}$$

→ Attribute with maximum gain ratio is selected as splitting attribute.

CART Algorithm

If dataset D contains examples from n classes, gini index, gini(D) is defined as

$$gini(D) = 1 - \sum_{j=1}^{n} p_j^2$$; $p_j \rightarrow$ relative frequency of class j in D

If dataset D is split on A into two subsets D_1 and D_2, the gini index gini(D) is defined as

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

Reduction in impurity :

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

→ Attribute provides smallest $gini_{split}(D)$ (or largest reduction in impurity) is chosen to split the node.