Mean : μ Coeff. of variance = σ/μ.
Variance : σ^2 Covariance = σ_{xy}^2
Standard dev : σ Correlation = ρ_{xy}

MACHINE LEARNING

I).

Distributions

Gaussian distribution.

for mean : μ
Std. deviation = σ.

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

↓ change of variables

$Z \sim N(0,1)$ or $Z = \frac{X-\mu}{\sigma}$

Uniform

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \le x \le b \\ 0, & \text{otherwise} \end{cases}$$

$E(x) = \frac{a+b}{2}$

$V(x) = \frac{(b-a)^2}{12}$

Exponential

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & x \ge 0 \\ 0, & \text{elsewhere} \end{cases}$$

$E(x) = 1/\lambda$

$V(x) = 1/\lambda^2$

Poisson distribution

$\alpha > 0$

$$P(x) = \begin{cases} \frac{\alpha^x}{x!} e^{-\alpha} \\ 0, & \text{otherwise} \end{cases}$$

$F(x) = \sum_{i=0}^{x} \frac{\alpha^i}{i!} e^{-\alpha}$

$E(x) = \alpha = V(x)$

Transforms

Fourier transform

$$F(\lambda) = \int_{-\infty}^{\infty} f(u)\, e^{-i\lambda u}\, du$$

$$f(u) = \frac{1}{2\pi} \int_{-\infty}^{\infty} F(\lambda)\, e^{i\lambda x}\, dx$$

Sets - unique numbers

- ephro = set([3,6,5,8,2,5])
- set.add(8)

try:
  statement
except:
  pass.

classes

```
class Human:
    def __init__(self, name, age)
        self.name = name
        self.age = age.
    ↓ def
human = Human('Man', 35)
print(human.age)
```

II) Basic Python

Help Sections

i)
- dir(______) — instance — views directories and modules
- help(______) — instance.
- type(______) — view datatype
- id(identifier) — returns id in memory.

String functions

(ii)
- len("Hello").
- input("Display").
- str.replace("0", "**")
- str.strip() — remove all white elements

Loop

(iii)
for i in range(start, stop, step)
  (start ↓ def, step ↓ def.)

Random. = import random.

(iv)
- random.random() — a random number between 0-1
- random.randrange(1,7) — between 1-6.

Math module - import math.

(vi) math.sqrt()

Data types

↓

Boolean.

(vii) List. Tuples

mylist = ['one',2,"three"]

myTuple = ('one', '2', 'three')

List operators

(viii) slice: julia[2:6]

Concatenation: julia = julia + ("Eat pray love",)

repetition: print([0]*4) : [0,0,0,0]

a.count("e").

in operator: 'p' in 'apple'.

Split & Join

words = song.split("e") → list; default

'/'.join(["leaders","are","best"]) → glue; list

del a[1] → a = ['one', 'two', 'three']

strings are non-mutable, list are.

List methods.

mylist.reverse()

mylist.remove(digit)

mylist.append(digit)

pop last element: last item = mylist.pop().

mylist.insert(position, digit).

(viii) Formatting of output.

print("Hello { }. Your score is { }."
format(name, score)).

{:.2f}
precise with 2 digits

(ix) Dictionaries.

Create: engEsp = { }

engEsp['one'] = 'uno'.

inventory.get("apples", 0) instead of error.

inventory.values()

deletion: del inventory['key']

numItems = len(inventory)

inventory.keys()

for key in inventory:
print(key)

for key, val in myfruits.items()
print(key, val)

III) file handling

i) OS File Handling

- os.getcwd() // cwd = working directory
- os.makedirs("C:\\myfolder")
- os.listdir("D:\\") — list dir
- os.chdir("D:\\myfolder") — change directory
- os.rename("oldfile.txt", "newfile.txt") — Create a renamed file

f = os.open("D:\\myfolder\\file.txt", os.O_RDWR | os.O_CREAT) — flags

ii) IO Module

Reading

- content = file.read() — all as string.
- lines = file.readlines() — list of all lines.
- for line in file: print(line.strip())
- for line:

Writing

- file_obj = open("square.txt", "w") — encoding = "utf-8"
- file_obj.write("...")

Opening

- file = open("olympic.txt", "r") — mode

modes
- r : Read only mode
- rr : Read and write mode
- w : Write mode only
- wt : write only mode
- a+ : append

close
- file.close() — file is also

iii) CSV file — import csv module

Read

with open('names.csv', 'r') as csv_file:

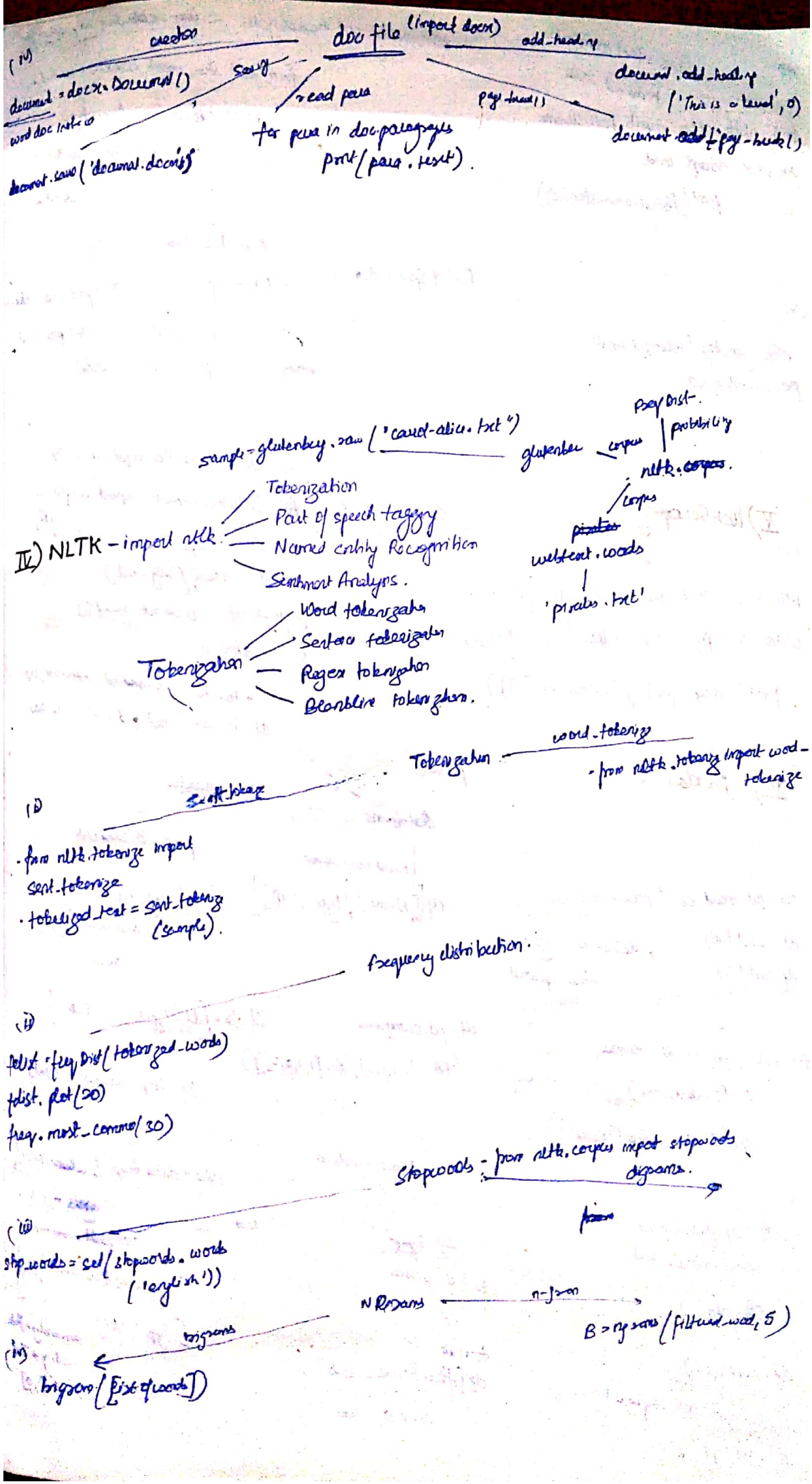
- csv_reader = csv.reader(csv_file) — object is memory
  - for line in csv_reader: print(line) = will read header → list
  - or next(csv_reader) // skips row.
- csv_reader = csv.DictReader(csv_file) — Read Dict
  - for line in csv_reader: print(line['email'])
  - or dict line['enroll']

Write

with open('new_names.csv', 'w') as new_file:

- csv_writer = csv.writer(new_file, delimiter='|')
  - csv_writer.writerow(line) — list
- csv_writer = csv.DictWriter(new_file, fieldnames=[...], delimiter='|') — fieldnames = list
  - csv_writer.writeheader()
  - csv_writer.writerow(line)

~~end for~~



(1) stem : text normalization — Stemming and lemmatization — uses vocabulary and morphological analysis

reduced words

```
from nltk.stem import PorterStemmer
for w in example_words:
    print(PorterStemmer.stem(w))
```

```
from nltk.stem import WordNetLemmatizer
WordNetLemmatizer.lemmatize
("word", pos="v/a")
adj/adv
```

(2) Part of Speech tagging. — Named entities — Relationship

```
nltk.pos_tag(tokenized_text)
pos_words = w
```

```
from nltk import ne_ch
ner_words = ne_chunk(pos_words)
print(ner_words)
```

spacy.

V) Web Scraping.

Beautiful Soup.

```
from bs4 import BeautifulSoup as soup
from urllib.request import urlopen as uReq
my_url = "https://www.flipkart.com/..."
uClient = uReq(my_url)
page_html = uClient.read()
uClient.close()
page_soup = soup(page_html, "html.parser")
containers = page_soup.findAll("div", {"class": "col 2-grid"})
print(soup.prettify(containers[0]))
```

VI) Pandas in

pandas

```
df = pd.read_csv('pokemon_data.csv')
df.head(10)
df.tail(3)
, delimiter='\t')
```
tab separated file.

```
df.to_csv('modified.txt', index=False, sep=
df.to_csv('modified.csv', index=False)
```

Dataframe.

Read header.

Read each column

```
df[['Name', 'Type 1', 'HP']]
```
new df

```
print(df.columns)
```

```
df.iloc[2,1]
```
location explicit — column — row

```
for index, row in df.iterrows():
    print(index, row)  or
    print(index, row['Name'])
```
debug.

```
df = pd.DataFrame
({'A': [1,2,3,3], 'B': [1,1,0,0]})
```

```
df.to_csv[df['Type 1'] == 'Fire']
df.drop(columns=['Total'])
```

Describing data

(1) df.describe()
shows mathematical

```
df.info()
df.nunique()
```
gives you unique values in keys

```
df.unique()
```

df.shape
list of dimensions

```
df['Name'].value_counts()
```
count of values

sort

```
df2 = data.drop(['columns-mod'],
axis=1)
horizontal, vertical
df.sort_values(['Name',
['Type 1', 'HP']
ascending=False
or ascending=True
or ascending=[1,0]
```

Pandas

(iii)
df['Total'] = df.iloc[:, 4:9].sum(axis=1))

df = df[cols[0:4] + [cols[-1]] + cols[4:12]]

String filtering

new_df = new_df.reset_index(drop=True)

inplace = True.

addition dropping

df.loc[df['Type 1'] == 'Fire', 'Type 1'] = 'Flamer'

condition

df.loc[df['Name'].str.contains('Mega')]

~ (not).

Groupby.

(iv)
df.groupby(['Type 1']).mean()

sum(), count()

where + declare

averages and groupby

- df.groupby(['Type 1']).mean().sort_values('Defense', ascending=False)
- df.groupby(['Type 1']).count()['Count']

display the count column count.

Obj data.

new_df = pd.DataFrame(columns=df.columns)

(v)
for df in pd.read_csv('modified.csv', chunksize=5):
    concat

new_df = pd.concat([new_df, results])

VII) student_data.drop(['col1', 'col2'], axis=1)

Understanding data — data.isnull().sum()

df.head() data.shape() df.columns df.unique() check null values

EDA: import pandas as pd
import numpy as np
import seaborn as sns

axis horizontal axis bars, axis=1
vertical axis bars, axis=0.

ML

Introduction

(i) Every ML Algorithm has three components. → optimization, generation of conditional figure.

- Representation
  - ensembles
  - SVM, neural networks, decision trees, Set of Rules
- Evaluation
  - Way of evaluating hypothesis
  - Accuracy, precision, recall, squared error, likelihood, posterior probability, cost margin.
- Combinatorial / Convex / constrained.

Classification, Regression, Learning

- Supervised : Training data includes desired output.
- Unsupervised : does not include
- Semi-supervised : Training data includes a few desired outputs
- Reinforcement : Rewards from sequence of actions.

Clustering, Dimensional Reduction

(ii) Data Representation — Information Systems : knowledge of Raw data for decision making
— Data table : used to represent Information
— Data structure : Integrated, consistent and cleaned data to ML algos.

Rows, Cols, values, decision attribute.

Linear Algebra

(1) training features y (matrix); $h_\theta = \theta^T x$ → Feature vector, parameter of cost function.
hypothesis → coefficient vector.

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

$x^{(i)}$ → feature vector of i^{th} instance
$y^{(i)}$ → matrix of i^{th} instance
m → no of training set.

function — cost function
optimization problem $J(\theta_0, \theta_1)$ min.
Gradient descent.

Repeat
$$\{ \theta_j = \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \}$$
or
$$\equiv \{ \theta_0 = \theta_0 - \frac{\alpha}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)$$
$$\theta_j = \theta_1 - \frac{\alpha}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$
} (Simultaneous update)

Data concepts

i) Feature scaling, divide feature by max value
$0 \le x_i \le 1$.
feature between $[-3, -\frac{1}{3}] \cup [-\frac{1}{3}, 3]$ is scaled

(ii) Mean normalization : to make features have mean = 0.
$x_i \leftarrow x_i - \mu_i$ (max-min) (Range of values)
imp. s_i
Standard deviation of $\sqrt{\sigma^2} = s$

(iii) Automatic convergence if $J(\theta)$ decreases by 10^{-3} on iteration.

Normal Equation

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

don't use when n is very large (no. of features)

When not invertible
- Redundant features
- Too many features
- Regularization (to remove overfitting)

Models: Deterministic, Probabilistic (one ϵ (Random)).

Relationship between two variables x and y can be checked by measuring the max variance from the hypothesis:

$y^{(i)} = h_\theta(x^{(i)}) + \epsilon$ // Error random probabilistic.

→ Linear Regression just uses straight line for data modelling.

→ Regression Modelling Steps
(i) Hypothesis deterministic component → defines the relation between y and x, finding hypothesis function and variables
(ii) Estimate unknown model parameters → Use gradient descent or normal equation.
(iii) Specify probability distribution of error term.
(iv) Evaluate the fitted model. → . SSE / s^2 / s
. Residual analysis
↳ Linear Assumption.
↳ Independence Assumption
↳ Normal distribution assumption
↳ Constant variance for all levels of $x^{(i)}$ datasets (homoscedasticity: variance with increase in x)

Accuracy
- → Accuracy
- → Precision
- → Recall
- → Score
- → ROC curve
- → Confusion matrix

Accuracy: Test of Slope Coefficient (t); Coefficient of Determination (r/r^2); Test for significance (regression)

Normal function:

$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$

for any $N(\mu, \sigma)$ can be converted to $N(0,1)$ by taking $z = \frac{x-\mu}{\sigma}$; mean = 0, variance = 1, always.

(v) Use for Model prediction & Estimation.

OLS

$\theta_1 = \frac{S_{xy}}{S_{xx}}$

$\theta_0 = \bar{y} - \theta_1 \bar{x}$

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

$\frac{\bar{x}\bar{y}}{m}$

↳ SSE (Sum of squared errors / unexplained sum of errors) $= \sum_{i=1}^{m} (y^{(i)} - h_{\theta}(x^{(i)}))^2$

↳ SS_{yy} (total sum of squares) $: \sum_{i=1}^{m} (y^{(i)} - \bar{y})^2$ — total variation

↳ $SS_{h_\theta(x)}$ (explained variation) : $\sum_{i=1}^{m} (h_\theta(x^{(i)}) - \bar{y})$ — $h_\theta(x)$ wrt $\bar{y}$

→ Variance of error $s^2 = \frac{SSE}{m-2}$

$s = \sqrt{\frac{SSE}{m-2}}$

Covariance / separate variance = Correlation

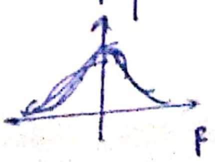
(II) Correlation

$r = \frac{SS_{xy}}{\sqrt{SS_{xx} SS_{yy}}}$

r^2: Coefficient of determination $\left(\frac{\text{explained variance}}{\text{total variance}}\right)$

(III) Test of slope coefficient

no. of datasets



$N \sim (\theta_1, s^2)$ — degree variance; ↳ mean of slopes; ↳ spread of error; observed value

$t = \frac{\theta_1}{s_{\theta_1}} = \theta_1 / \frac{s}{\sqrt{SS_{xx}}}$

IT323	B	MACHINE LEARNING
IT323	D	MACHINE LEARNING
MC305	A	OPERATION RESEARCH
MC305	D	OPERATION RESEARCH
MC307	B	OBJECT ORIENTED PROGRAMMING
MC315	C	MODERN ALGEBRA
PE307	C	FINITE ELEMENT METHOD
PE315	D	MECHATRONICS
PT305	D	PROCESS EQUIPMENT DESIGN
PT309	B	PETROLEUM REFINING ENGINEERING
PT319	C	BIOMATERIALS
SE321	B	ARTIFICIAL INTELLIGENCE
SE323	A	THEORY OF COMPUTATION

Linear Algebra

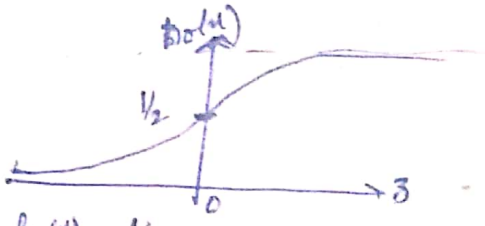
- orthogonal matrix $AA^T = I$
- orthonormal matrix $A \cdot B = 0 = A \cdot B^T$.

Logistic Regression

↳ (i) hypothesis func. , $h_\theta(x) = g(\theta^T x^{(i)}) = \frac{1}{1+e^{-\theta^T x^{(i)}}}$

$h_\theta(x) \geq 0.5$, $\theta^T x^{(i)} > 0$

$h_\theta(x) < 0.5$, $\theta^T x^{(i)} < 0$



$h_\theta(x) = \frac{1}{1+e^{-z}}$ → $z=0, \frac{1}{1+1}, 1/2$

→ $z=1, \frac{1}{1+e^{-1}} = \frac{1}{1+1/e} = \frac{e}{e+1}$

Cost function : $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

Cost$(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y=1 \text{ (for some func)} \\ -\log(1-h_\theta(x)) & \text{if } y=0 \text{ (for some func)} \end{cases}$

Compact → $-y \log(h_\theta(x^{(i)})) - (1-y) \log(1-h_\theta(x^{(i)}))$

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

Compact cost fun → $J(\theta) = -\frac{1}{2m} \sum_{i=1}^{m} \left[y \log(h_\theta(x^{(i)})) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)})) \right]$ feature mach

Gradient descent. : min $J(\theta)$ get hypothesis

Repeat

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

not correct $\theta^T x$

optimized Algorithm (Conjugate Gradient)

Cost function Quadratic or cubic just a polynomial

for multiple classes, predict $h_\theta(x^{(i)})$ for each class — pick one with highest and classify.

Overfitting

Overfitting : If we have too many features, the learned hypothesis may fit the training set very well, but fail to generalize to new examples

Addressing overfitting

↳ • plotting the hypothesis
- Reduce number of features (drop some features using con-selection).
- Model Selection Algorithm
- keep all features but reduce magnitude of coefficients.

Regularized cost function

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

↳ Regularization parameter

for the parameters you want to include some coefficients

Sum parameter coefficient in hypothesis function.

Logistic Regression

Regression

Same

Normal equation

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

} Gradient descent

$$X = \begin{bmatrix} x^{(1)T} \\ x^{2T} \\ x^{3T} \\ \vdots \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \end{bmatrix}$$

$$\theta = \left(X^TX + \lambda \begin{bmatrix} 0 & 0 & \cdots \\ 0 & 1 & 0 \\ \vdots & 0 & 1 \\ & & & \ddots \\ & & & & 1 \end{bmatrix}\right)^{-1} X^Ty$$

Size (n+1)

(i) Types of logistic Regression

- Binary (0 & 1)
- 3 or more unordered — Nominal
- 3 or more ordered — Ordinal

→ cutoff can could be 0.5

(ii) $$h_\theta(x^{(i)}) = \frac{1}{1 + e^{-\theta^T x^{(i)}}}$$

$$\theta^T x^{(i)} = \ln\left(\frac{h_\theta x^{(i)}}{1 - h_\theta x^{(i)}}\right)$$

↳ log odds Ratio, Logit Transform

Logarithm of Odds Ratio

of $\theta^T x^{(i)}$

Convolutional Neural Network

1) Layer — Convolution (⊗) - trigger filter.
- ReLU layer (⟋) — transformation function (Rectified Linear unit)
- Pooling (▬) — eg max pool

Fully Connected
↳ used for classification based on sum of activated class nodes. Choose class one.

Bayesian Decision Theory

→ Design classifiers to recommend decisions that minimize some total expected "risk". → Classification error, Risk, Cost.

Page 7 of 11

Terms:

- $p(x)$ — evidence or probability density function.
- $P(\omega_j)$ — prior. Random Variable class.
- $P(x/\omega_j)$ — likelihood, or conditional probability density. feature vector.
- $P(\omega_j/x)$ — posterior.

Bayes Rule

$$P(\omega_j/x) = \frac{P(x/\omega_j)P(\omega_j)}{p(x)}$$

posterior = likelihood × prior / evidence

$$p(x) = \sum_{j=1}^{2} P(x/\omega_j)P(\omega_j)$$

Decision Rule

→ Decide Error probability

$$P(error) = \min\left[P(\omega_1/x), P(\omega_2/x)\right]$$

- The rule is optimum if no other information is available. → minimizes the Average probability Error.
- Makes same decision all time.

→ Average probability error

$$P(error) = \int_{-\infty}^{\infty} P(error/x)\,dx$$

$$= \int_{-\infty}^{\infty} P(error/x)\,p(x)\,dx$$

all $p(x)$ contribute

Probabilities Origin

- Objective: Relative frequency — Prob. can only come from experiment.
- Subjective: Bayesian approach — Prob. may reflect degree of belief and can be based on opinion.

Zero-one loss function → Overall Risk = Average probability error.

$$\lambda(\alpha_i/\omega_j) = \begin{cases} 0, & i=j \\ 1, & i \neq j \end{cases}$$

Conditional Risk

or expected loss with taking action α_i on basis of observable x, and ω_j being correct category

$$R(\alpha_i/x) = \sum_{j=1}^{c} \lambda(\alpha_i/\omega_j)\,P(\omega_j/x)$$

Overall Risk

$$R = \int R(\alpha(x)/x)\,p(x)\,dx$$

To minimize risk, choose action with least $R(\alpha_i/x)$

$\lambda_{ij} = \lambda(\alpha_i/\omega_j)$

RISK FUNCTION

→ When a cost or loss is associated with each error

Loss function

$\lambda(\alpha_i/\omega_j)$ (depends on class)

→ Loss associated with taking action α_i when correct category is ω_j.

where $\alpha_i \in$ Actions $\{\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots\}$

$\alpha(x)$: general decision rule.

Decide ω_1 if $\frac{P(x/\omega_1)}{P(x/\omega_2)} > \frac{(\lambda_{12}-\lambda_{22})P(\omega_2)}{(\lambda_{21}-\lambda_{11})P(\omega_1)}$ — Two Category Classified

Likelihood — Threshold

(1) Risk is depended on category. Less Risk – better selection.

ANNEXURE 2

List of Open Elective Courses (OECs) for B.Tech. V Semester

Course Code	Course Title
CO357	OPERATING SYSTEM
CO361	DATABASE MANAGEMENT SYSTEM
EC353	COMPUTER VISION
EC 357	DIGITAL IMAGE PROCESSING
EC359	VLSI DESIGN
EE351	POWER ELECTRONICS SYSTEM
EE353	ELECTRICAL MACHINES AND POWER SYSTEMS
EE355	INSTRUMENTATION SYSTEMS
EE357	UTILIZATION OF ELECTRICAL ENERGY
EE359	NON-CONVENTIONAL ENERGY SYSTEM
EN353	OCCUPATIONAL HEALTH AND SAFETY MANAGEMENT
HU351	ECONOMETRICS
HU353	INTERNATIONAL TRADE
IT353	DATA STRUCTURES AND ALGORITHMS
ME353	RENEWABLE SOURCES OF ENERGY
ME359	REFRIGERATION AND AIR CONDITIONING
PE353	SUPPLY CHAIN MANAGEMENT
PE361	TOTAL QUALITY MANAGEMENT
PT361	HIGH PERFORMANCE POLYMERS
PT367	POLYMER WASTE MANAGEMENT

Naive Bayes Classifier

- Based on Naive Bayes Theorem
- Relationship on Joint probability

$$P(x) = P(x_1, x_2) = P(x_2|x_1)P(x_1)$$
$$= P(x_1|x_2)P(x_2)$$
$$= P(x_1) \cdot P(x_2)$$

(independent)

- Bayesian Rule

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)}$$

Discriminative — Generative

Classifier

- Probabilistic model of data within each class
- Naive Bayes, model based classifier

Model a classification Rule directly

- k-NN, decision trees, perceptron, SVM

Model the probability of class membership from input data

- perceptron with the cross-entropy cost.

Discriminative classifier $P(c|x)$

Generative classification $P(x|c)$

Probabilistic classifier.

→ PROBABILISTIC CLASSIFICATION

(i) Discriminative classifier : to train, all training examples of different classes must be jointly used to build a single discriminative classifier

(ii) Probabilistic classifier : Outputs L probabilities for L class labels i.e.

non-Probabilistic : single label is achieved.

(iii) Generative Model : Must be probabilistic.

- probabilistic model has to be learned independently for all classes.
- Build L probabilities for a given input with L models.
- "Generative" means that such a model produces data subject to the distribution via sampling.

→ MAP Rule : Maximum A Posteriori Classification Rule

- Choose one the label with maximum probability among $P(c|x)$ where $i \in L$, no of labels or classes

Assuming all features are class conditionally independent

$$P(y|x_1, x_2, \ldots, x_n) = \frac{[P(x_1|y)P(x_2|y)\ldots]P(y)}{P(x_1)P(x_2)\ldots P(x_n)}$$

→ Zero CONDITIONAL PROBABILITY

- If no example contains the feature value

$\hat{P}(a_{jk}|c) = 0$ → $\hat{P}(x|c)\ldots\hat{P}(a_{jk}|c)\ldots\hat{P}(x_n|c) = 0$

$\hat{P}(c|x) = 0$ for all class

↓ class conditional probabilities re-estimated.

$$\hat{P}(a_{jk}|c_i) = \frac{n_c + mp}{n+m} \quad [m\text{-estimate}]$$

prior estimate ($p = 1/t$ for t possible values of x_j)

$x_j = a_{jk}$ — x_j = attribute

n : no of training examples for which $x_j = a_{jk}$ and $c = c_i$.

n_c : no of training examples for which $c = c_i$

m : weight to prior (number of "virtual" examples)

Laplace 1:1 of # training example.

- A good candidate of a base learner in ensemble learning.

Different learning methods

→ Instance-based learning
- learning = storing all training instances.
- Classification = assigning target function to a new instance
- Referred as lazy learning.

Eager learning & explicit description of target function on the whole training set (input independent)

→ Eager learning. & Instance Based Learning

K-Nearest Neighbor Algorithm | Weighted Regression | Case-based reasoning

K-Nearest Neighbor

- Discriminative Classifier
- Solves both Regression and classification
- Also known as:
  - Memory-Based Reasoning
  - Example-Based Reasoning
  - Instance-Based Learning
  - Case-Based Reasoning
  - Lazy Learning.

K-Nearest Neighbor Algorithm

- Classification is delayed till a new instance arrives
- Classification done by comparing feature vectors of the different points.
- Target function may be discrete or real valued.

K-Select

↳ if K is too small, sensitive to noise

↳ K is too large, neighborhood may include points from other classes.

↳ Always take a odd value.

Target function = mean value of the k nearest training examples.
Continuous (Regression).

Voronoi Diagram: Creates a decision surface for the training examples.

Distance-Weighted Nearest Neighbor Algorithm

(i) Assign weights to the neighbors based on their 'distance' from the query point.
- Weight 'may' be inverse square of the distance. less distance highest weight.

(ii) • Shepard's method: All training point may influence a particular instance.

— Curse of dimensionality: when the no of features per instance is high.
- Efficient memory indexing (k dimensional tree)

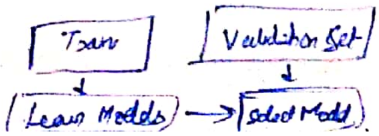
KNN Algo:
- Calculate distance for all training instance
- Sort them according to distance
- Pick the labels of selected K entries
- If regression, return the mean.
- Classification, return mode.

Evaluation of Machine Learning Classifiers

→ Learning Curve shows how accuracy of a learning method change as function of the training-set size s.

plotting:
- divide test set into sample size 's' on learning curve — repeat n times.
- randomly select s instances from training set and learn model.
- Evaluate model on test set to determine accuracy a. - plot (s, a) or (s, avg accuracy and error bars)

→ Tuning (Validation) Set: to get accuracy estimate during learning process. - divide training set into validation set.



(ii) Random Sampling: Repeatedly randomly partitioning available data into training set and test set. Make sure we may or may not proportions are maintained.

Stratified sampling: - first stratify instances by class
- Random select instances from each class proportionately

(iii) Cross Validation: Partition data into n subsamples, and iteratively train on subsample and for the test set, train Calculate average accuracy.

Stratified cross validation

Stratified sampling is used when partitioning the data. Uniform proportionality is maintained.

★ stratified always has same proportion in test, train and validation set.

(iv) Internal Cross Validation: Instead of a single validation set, use cross-validation within a training set to select a model (eg. to select choose the best level of decision pruning) or (to select k in k-NN)

Actual \ Predicted	+	-
+	t_p / a	f_n / b
-	f_p / c	t_n / d

Terms

(i) Precision = $P(\text{relevant}/\text{retrieved}) = \frac{t_p}{t_p + f_p} = \frac{a}{a+c}$

Recall = $P(\text{retrieved}/\text{relevant}) = \frac{t_p}{t_p + f_n} = \frac{a}{a+b}$

positive recognition set = $\frac{t_p}{\text{total}} = \frac{a}{a+b+c+d}$ negative recognition set = $\frac{d}{a+b+c+d}$.

(iii) TPR = Recall = $\frac{a}{a+b}$

FPR = $\frac{d}{d+c}$ = Specificity

F-Measure = $\frac{1}{\alpha \frac{1}{p} + (1-\alpha)\frac{1}{R}}$ [f measure, $\alpha = 1/2$]

(ii) Sensitivity = ~~Recall~~ = $P(\text{retrieved}/\text{relevant}) = \frac{t_p}{t_p + f_n} = \frac{a}{a+b}$; Specificity = $\frac{t_n}{t_n + f_p} = \frac{d}{d+c}$

IT427	S	INTRUSION DETECTION AND INFORMATION WARFARE
MC411	P	Data Warehousing & Data Mining
ME409	P	MECHATRONICS & CONTROL
ME411	P	I.C. ENGINES
ME413	P	METROLOGY
ME415	P	PROJECT MANAGEMENT
ME419	Q	ROBOTICS & AUTOMATION
ME421	Q	COMPUTATIONAL FLUID DYNAMICS
ME423	Q	ADVANCED MANUFACTURING PROCESSES
ME427	Q	OPERATIONS RESEARCH
ME429	R	INDUSTRIAL TRIBOLOGY
ME431	R	NON-CONVENTIONAL ENERGY SOURCES
ME433	R	COMPUTER INTEGRATED MANUFACTURING
ME435	R	OPTIMIZATION TECHNIQUES
PE411	S	COMPUTER INTEGRATED DESIGN AND MANUFACTURING
PE413	Q	ROBOTICS AND AUTOMATION
PE417	R	MATERIALS MANAGEMENT
PT415	Q	PAINT TECHNOLOGY
PT419	R	Plastics and Environment
PT427	S	SAFETY & HAZARDS IN CHEMICAL INDUSTRY
SE411	R	SOFTWARE QUALITY AND METRICS
SE417	S	DATA WAREHOUSING AND DATA MINING

Roc

TP rate vs. the FP rate as a threshold on the confidence of an instance being positive & values.

$TPR = \frac{tp}{tp+fp} = \frac{a}{a+b}$, $FPR = \frac{fp}{tp+fp} = \frac{c}{d+c}$

r random guessing, TPR & FPR.

AUC is more robust in accuracy in class imbalanced situations.

$\frac{TPR}{FPR} = \frac{1+(c/d)}{1+(b/a)}$

→ ROC curves can be used with misclassification cost.

Helps to identify operating point.

F.P.R ∝ TP.R

→ ROC creation.

- Sort test prediction acc. to confidence (+ve confidence)
- Step through sorted list from top to lower confidence.
  - Select a threshold between classes
  - Compute TPR, FPR
  - Plot.
- Can interpolate between points to get a convex hull (skip points)

— Traverse to edges to property of test set.

- Can identify global optimal class/prob threshold.

→ Prob : — misclassification cost distributions used help the AUC are different for different classifiers.

PRECISION-RECALL

- plots precision vs. Recall as a function of the confidence of a instance being positive vs. vice versa
- Well suited for tasks with lots of negative instances.

TP.R vs Precision

$TPR = \frac{a}{a+b}$; $Prec = \frac{a}{a+c}$

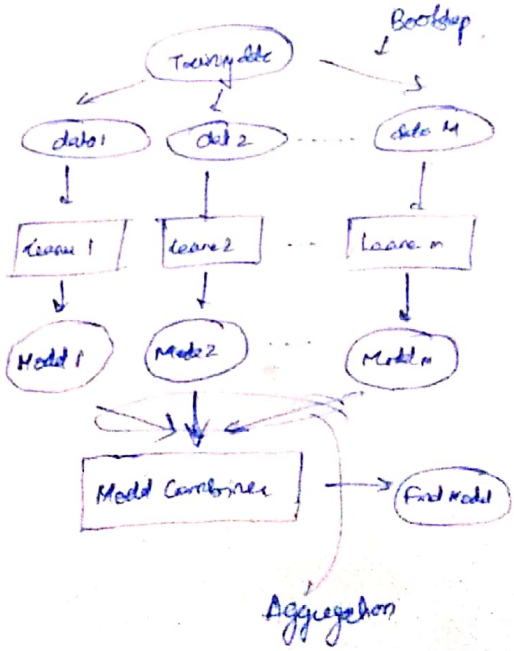
↓ $\frac{a+b}{a+c}$ big ∝

Recall ∝ T.P.R decrease as increase

Precise decreases.

ENSEMBLE LEARNING :- Learn multiple alternative definitions of a concept using different training data or different learning algorithms.

- Combine decisions of multiple definitions, e.g. using weighted voting.



Support Vector Machine

Introduction

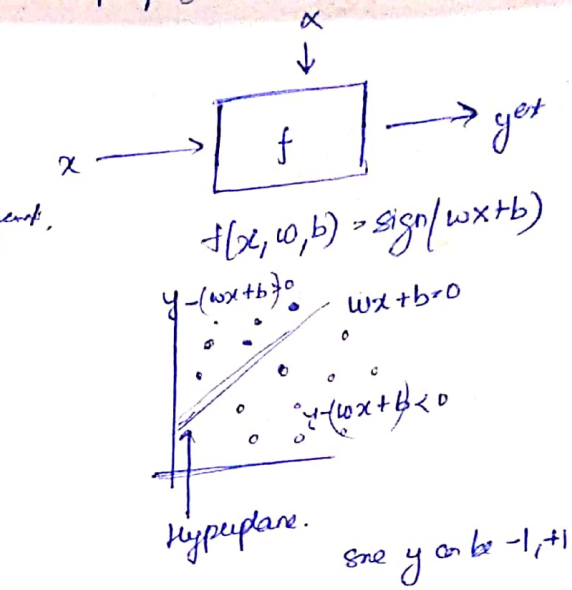
- Objective of SVM algorithm is to find a hyperplane in a N-dimensional space / High accuracy with less computation
- Can be used as regressor and Classification.

→ Define the margin of a linear classifier as the width that the boundary could be increased before hitting a data point.

- Support vectors are those datapoints that the margin pushes up against.
- LSVM is the simplest kind of SVM.

$$\text{distance} = \lambda \|w\| = \frac{2}{\sqrt{w^T w}}$$

$$\lambda = \frac{2}{w^T w} = \frac{2}{\|w\|^2}$$



Here y can be -1, +1

Goal: i) Calling all having data

If $y_i = +1$, then if $wx_i + b \geq 1$

$y_i = -1$, if $wx_i + b \leq -1$

$\therefore y_i(wx_i + b) \geq 1$

ii) Maximizing margin $M = \frac{2}{|w|}$ same as minimize $\frac{1}{2} w^T w$.

Quadratic optimization problem,

Minimize $\phi(w) = \frac{1}{2} w^t w$ subject to $y_i(wx_i + b) \geq 1 \quad \forall i$

Construct a dual problem, use Lagrange multipliers

Noisy data set

NLP — (i) text mining in NLP
(ii) DS Models in Python & All Reading
(iii) NLTK Package
(iv) Tokenization
(v) Frequency distribution
(vi) Stop-words
(vii) As-N grams.

ML project
① EDTA on textual Analysis
② pipelines in ML
③ LSTM
④ Oversampling, Undersampling
⑤ Word Vectors, Word embeddings

✓ ① Basics of NLP
✓ ② EDA on textual file
✓ ③ Base SVM, Random forest, Logistic Regression
✓ ④ N-grams
✓ ⑤ pipelines in ML
✓ ⑥ CNN, LSTM & Bases
✓ ⑦ Confusion matrix
⑧ Oversampling
⑨ Undersampling
✓ ⑩ Basics of PRAW API
⑩ Word vectors, word embeddings

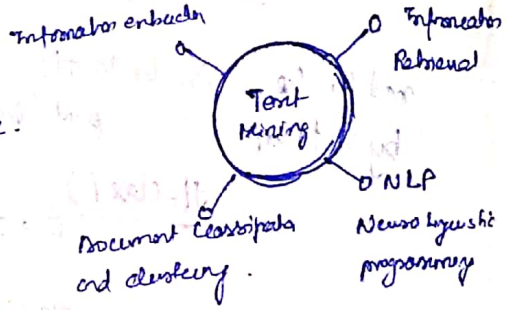
Text Mining

→ Need of text Mining. —— Unprecedented amount of unstructured text data being generated everyday

Text mining transforms unstructured data into structured data to further to be used for Analysis.

Make informed decision automated processes, market research using sentimental analysis.

Identify business insights to fuel business processes and reduce risk.



→ Natural language: Any dialect used by Humans to communicate.
NLP is a sub domain of AI

NLP text Mining.
- Anaconda is a free and open source distribution of Python and R programming languages, specifically designed to perform Data Science and Machine Learning.
- OS Module provides functionality to perform operating system dependent operations such as creating directories, creating renaming and removing files and many other tools.

use Jupyter notebook

you can import os library.

How current working directory

function: os.getcwd() : returns current working directory.

path to extract directory : / when getting directory for computer : \.

os.makedirs("C:\\myfolder")
create a new folder

read write
f = os.open("D:\\my-folder\\file.txt", os.O_RDWR | os.O_CREAT)
open a file and set flags.
os.O_CREAT flags
if file does not exist, it will automatically create it

os.listdir("D:\\")
list the entries of a folder.

os.chdir("D:\\myfolder")
change working directory

os.rename("oldfile.txt", "newfile.txt")
rename file
creates a new renamed file

The io module is used to perform file handling operations. Contains function that return a file object - called "handle" which is then used to read from or write to a file.

Creating file object → • To create a file object, we use the following function from the "io" module.

open(file, mode)

- r : Read only mode, starts at the beginning of the file
- r+ : Read and write, mode file pointer starts at the beginning of the file
- w : Write mode only, starts at the beginning of a file and overwrites the existing file.
- w+ : Also allows to read from a file apart from writing to a file.
- a+ : appends.

• fh.readline() is like stay tokenizer. → Reads/returns a list of lines

read file line by line for loop

```
fh = open('document', 'r')
for line in fh:
    print(line)
```

fh.close() → clears all of the file buffers

writing correctly to file

```
newfile = open("D:\\myfolder\\new-file.txt", mode="w", encoding="utf-8")
newfile.write("This is first line \n")
newfile.close()
```

to append

```
list = ["New York\n", "London\n", "Germany\n"]
new-file = open("D:\\myfolder\\newfile.txt", mode="a+",
                encoding="utf-8")
new-file.writelines(list)
```

docx → import docx.

#Creating new doc instead of read doc

```
document = docx.Document()
```

#Adds a heading to the document

```
document.add_heading('This is a Level Zero heading', 0)
```

Save document

```
document.save('document.docx')
```

to add page break

```
document.add_page_break()
```

Already existing

```
doc = Document('document.docx')
```

Reads paragraphs from the document

```
for para in doc.paragraphs
    print(para.text)
```

Natural Language tool kit (NLTK)

- Tokenization
- Part of speech Tagging
- Named entity Recognition
- Sentiment Analysis.

Corpus: is a huge collection of written text.
A compilation of corpuses is called a corpora. It is a body of written or spoken texts used for linguistic analysis and the development of NLP tools.

→ Download nltk corpora

```
import nltk
nltk.download()
```

NLTK Corpus

```
from nltk.corpus import names
```

Tokenization → Process of turning a string a text paragraph into smaller chunks or tokens such as words, phrases, keywords, symbols etc. is called Tokenization.

↓ tokenized tokens

- sent_tokenize (Sentence tokenization)
- word_tokenize (words tokenization)
- RegexpTokenizer (Regular expressions)
- BlanklineTokenizer: ignores blank lines

or. sent_tokenize [to split a body of text into sentences]

```
from nltk.tokenize import sent_tokenize
tokenized_text = sent_tokenize(sample)
print(tokenized_text)
```

list of sentences.

```
len(tokenized_text)  # will return length of the list
```

sent_tokenize

```
from nltk.corpus import gutenberg
```

Story
Gutenberg corpus
from nltk corpus.

word_tokenize

- To split a body of text into words.
- from nltk.tokenize import word_tokenize

Regexp Tokenizer →

```
from nltk.tokenize import RegexpTokenizer
capword_tokenize = RegexpTokenizer('[A-Z]\w+')  # Method to words starting with a capital letter
```

Blankline Tokenizer →

```
from nltk.tokenize import BlanklineTokenizer
sample = "text..."
BlanklineTokenizer().tokenize(sample)
```

frequency distribution. → To get the word frequency in a text. It works similar to a dictionary where the keys are the words and the values are the counts associated with the word.

```
from nltk.tokenize import word_tokenize
from nltk.corpus import gutenberg
from nltk.probability import FreqDist
sample = gutenberg.raw("carroll-alice.txt")

tokenized_words = word_tokenize(sample)
fdist = FreqDist(tokenized_words)
print(fdist)
```

→ fdist.plot(20) to see 20 words.

→ freq.most_common(30) // gives the most common 30 words

Stopwords (useless words)

→ from nltk.corpus import stopwords.

```
stopwords = set(stopwords.words("english"))
print(stop_words)
```

most common as stopwords

fdist.most_common(20)

to remove "english" stopwords.

```
from nltk.tokenize import sent_tokenize, word_tokenize
filtered_word = []
sample = gutenberg.raw("carroll-alice.txt")
tokenized_word = word_tokenize(sample)

for w in tokenized_word:
    if w not in stop_words:
        if len(w) > 3:
            filtered_word.append(w)

fdist = FreqDist(filtered_word)
print(fdist)
```

Bigrams, Trigrams → Bigrams are "two consecutive words that occur in a text

Ngrams.

To create Bigrams using "webtext" corpus.

```
from nltk.corpus import webtext, stopwords
from nltk import bigrams.
text_words = [w.lower() for w in webtext.words('pirates.txt')]
```

to lower text

words present in pirates.txt text file.

```
stop_words = set(stopwords.words("english"))
```

```
filtered_words = []
for w in test_words[]
    if not w not in stop_words:
        if len(w) > 3:
            filtered_word.append(w)

A = bigrams(filtered_word)
fdist = FreqDist(A)
fdist.most_common(20)
```

Trigrams →

Ngrams → `from nltk import ngrams`

`B = ngrams(filtered_word, 5)`

Stemming → Stemming is a process for text normalization which involves reducing words to their root / base words from their derivational affixed or derived words.

```
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize.

ps = PorterStemmer()
example_words = ["like", "likely", "liking", "liked", "unlike"]

for w in example_words:
        print(ps.stem(w))
```

Lemmatization → Lemmatization is a process what involved reducing words to their roots / base words using vocabulary and morphological analysis. Lemmatization brings context-to the words which is not done in stemming.

→ `from nltk.stem import WordNetLemmatizer`

`lemmatizer = WordNetLemmatizer()`

`print(lemmatizer.lemmatize("word"))` ⟹ `("word", pos = "a")` (adjective)

Part of Speech tagging → task is to label / tag word and in a sentence with its grammatical group such as noun, pronoun, verb, adverb, adjective, tense and many more.

'V' = same as stemming

CC - Coordinating Conjunction
IN - Preposition / Subordinating Conjunction
JJS - Adjective, superlative
NNP - Proper Noun, Singular

JJ - Adjective
JJR - Adjective, comparative
NN - Noun, singular
PRP - Personal pronoun.

perform POS tagging. nltk.pos_tag(tokenized_text)

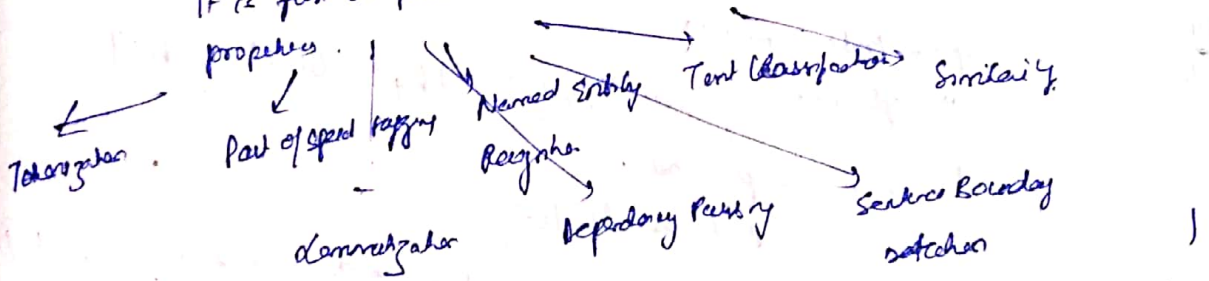
Named Entity Recognition → To identify important named entities in a text such as people, places, organizations, locations, dates, states, work of art etc.

Eg:

(i) ORGANIZATIONS: facebook, Alphabet Inc
(ii) LOCATION: Old West World, Mont Saint
(iii) GPE: India, Ukraine, South Asia
(iv) MONEY: 100 million Dollars, GBP 50
(v) PERSON: Jeorge W. Bush, Obama
(vi) DATE: July, 2019-05-2
(vii) TIME: three forty pm, 3:30 am
(viii) FACILITY: Washington Monument, Stonehenge

```
# from nltk import word_tokenize, pos_tag
pos_words = nltk.pos_tag(tokenized_text)
from nltk import ne_chunk
ner_words = ne_chunk(pos_words)
print(ner_words)
```

Introduction Spacy → free and open source library for advanced NLP in python - written in cython language (C extension of Python) which gives C like performance to Python. It is fast and provides concise APIs to access its methods and properties.



→ spacy contains language vocabularies, trained models, syntaxes and entities

```
# • import spacy
  nlp = spacy.load("en_core_web_sm")

• pass string for preprocesses.
    doc = nlp(sample)
              text file instance

• to print tokens: for token in doc:
                       print(token.text)
                   # token is a pre-processed object

Lemmatization • for lemma in doc:
                    print(lemma.text, lemma.lemma_)
```

→ part of speech tagging # for token in doc

print(token.text + "---", token.pos_ + "---", token.tag_)

Visualize → import spacy

to postag: from spacy import displacy

nlp = spacy.load("en_core_web_sm")

doc = nlp("Natural Language processing is fun!")

displacy.serve(doc, style = "dep")

~~nlp = spacy.load~~

Named entity Recognition → for ent in doc.ents

doc = print(ent.text + "---", ent.label_).

Visualize → displacy.serve(doc, style = "ent") [from spacy import displacy]

MATPLOTLIB - 2D and 3D library.

Version → from matplotlib import pylab

print(pylab.__version__)

Use numpy to create random data → x = np.linspace(0, 10, 25) # provide a random list of 25 variables, with each variable between 0-10.

y = x**2*x + 2

plotting → (i) pylab.plot(x, y, 'r') # 'r' stands for red.

(ii) Drawing a subgraph

pylab.subplot(1,2,1) # Two values of the brackets represent (rows, columns, columns)

pylab.plot(x, y, 'r--') → color and line style = '--' for dashed lines

pylab.subplot(1,2,2)

pylab.plot(y, x, 'g*-')

no. → 1 2 3 (3 columns)

(iii) To manipulate Canvas size. pylab.subplot(1,2,1) (drawing canvas) → subplot row, subplot column.

if both pylab.subplot(1,2,1) will cause overlapping of graphs.

Opacity → . import pyplot

Description. from matplotlib import pyplot as plt

Canvas → fig = plt.figure()

axis = fig.add_axis([0.5, 0.1, 0.8, 0.8]) # Control the left, right, width, height of the Canvas (from 0 to 1)

axis.plot(x, y, 'r')

ANDROID No - 26/08/2020.

Multi-Variate Linear Regression

→ Multiple features

m : no of training set.

(vector) $x^{(i)}$: input (features) of i^{th} training example (vector)

(1 number) $x_j^{(i)}$ = Value of feature j in the i^{th} training example.

Univariate hypothesis (for single feature) : $h_\theta(x) = \theta_0 + \theta_1 x$.

Multivariate hypothesis : $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$.

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

$= \theta_0 x_0 + \theta_1 x_1 \dots$

$\therefore x$ (feature vector $\in \mathbb{R}^{n+1}$) : $\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{bmatrix} \in \mathbb{R}^{n+1}$.

$\therefore h_\theta(x) = (\theta^T x)$ $(1 \times (n+1))\,((n+1) \times 1)$ matrix.

→ Gradient Descent for Multiple features.

$h_\theta(x) = \theta^T x$.

parameters for cost function : θ : $\theta_0, \theta_1 \dots \theta_n$.

~~matrix~~ of vector of dimension $(n+1)$

Cost function $J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$

Gradient descent

Repeat {

$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$ (Simultaneously update for every $j = 0 \dots n$)

}

for $n = 1$.

Repeat {

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

$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_1^{(i)}$

}

MACHINE LEARNING.

New Algorithm ($n \geq 1$):

Repeat {

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

} (simultaneous update)

Feature Scaling

→ More skewed are the contours, more time the gradient descent takes.

→ if you change scales by the max value of domain, your contour will become close to a circle

$$0 \leq x_i \leq 1.$$

or get every feature approximately to

$$-1.5 \leq x_i \leq 1.$$

A feature that takes values from -3 to 3 (scalable)

or $-\frac{1}{3}$ to $\frac{1}{3}$ (scalable).

→ Mean Normalization

Replace x_i with $x_i - \mu_i$ to make features have approximately zero mean.

$x_i \leftarrow \frac{x_i - \mu_i}{s_i}$ → μ_i → avg value of x_i in training set

s_i → range of values (max − min) or standard deviation of the variable.

Automatic convergence test

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

Features and Polynomial Regression.

Computing Parameters Analytically

Normal equation: Method to solve θ analytically.

$\theta \in \mathbb{R}^{n+1}$

$(n+1)$ dimensional vector

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

$\theta^T x^{(i)}$

by calculus $\frac{\partial}{\partial \theta_j} J(\theta) = \dots = 0$ (for every j)

Solve for $\theta_0, \theta_1, \dots \theta_n$.

General cost

m examples $(x^{(1)}, y^{(1)}), \ldots (x^{(m)}, y^{(m)})$; n features → features of data 1

$x^{(i)} = \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ x_2^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} \in \mathbb{R}^{n+1}$

X (design matrix) $= \begin{bmatrix} — & (x^{(1)})^T & — \\ — & (x^{(2)})^T & — \\ & \vdots & \\ — & (x^{(m)})^T & — \end{bmatrix}$ → features of data 2

$(n+1) * m$.

$y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix}$

Normal Equation

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

$(X^T X)^{-1}$ → inverse of matrix $X^T X$.

θ is a matrix of $\theta_0, \theta_1, \theta_2, \theta_3 \ldots$

x_1, x_2, x_3, x_4 → feature 1 of all data sets.

for m, being examples, n features.

Gradient Descent	Normal Equation
• Need to choose α	• No need to choose α.
• Need 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.
	☆ Cost of inverse $\approx O(n^3)$

Normal Equation Non Invertibility

(i) if $X^T X$ is non-invertible ?

→ Redundant features (linearly dependent).

→ Too many features (eg $n \geq m$)

- Delete some features, or use regularization.

CLASSIFICATION AND REPRESENTATION

Read

Create vectors & matrixes (inverse and transverse function) → plot data → search paths. - Gaussian distribution

add path or directory (where to function & header files) add

Vectorization

CLASSIFICATION

→ Using linear Regression, and Threshold classifier output $h_\theta(x)$ at 0.5

if $h_\theta(x) \geq 0.5$, predict "y=1"

if $h_\theta(x) < 0.5$, predict "y=0".

→ We will use Logistic Regression as a Classification Algorithm.

LOGISTIC REGRESSION

→ Want $0 \leq h_\theta(x) \leq 1$.

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

Example if $x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \text{tumor size} \end{bmatrix}$

$h_\theta(x) = 0.7$: tell patient that 70% of tumor being malignant

$$h_\theta(x) = P(y=1 \mid x;\theta)$$

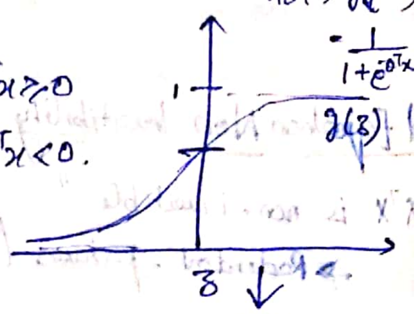
given x, parameter θ.

note: $P(y=0|x;\theta) + P(y=1|x;\theta) = 1$.

→ hypothesis function

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

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



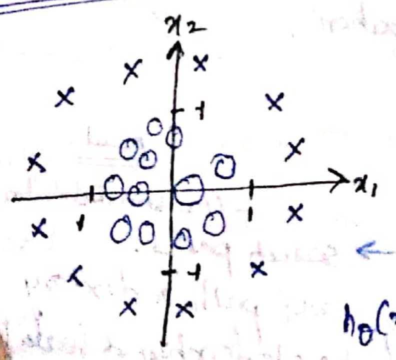
Suppose predict "y=1" if $h_\theta(x) \geq 0.5 \rightarrow \theta^T x \geq 0$

predict "y=0" if $h_\theta(x) < 0.5 \rightarrow \theta^T x < 0$.

when, $z \geq 0$, $g(z) \geq 0.5$.

ie whenever $\theta^T x \geq 0$, $g(\theta^T x) \geq 0.5$

non-linear decision boundaries



$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2)$.

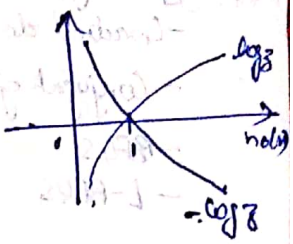
★ $h_\theta(x) = \theta_1 x$

Cost function - LRM

Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$

m examples: $x \in \begin{bmatrix} x_0 \\ x_1 \\ \cdots \\ x_n \end{bmatrix} \in \mathbb{R}^{n+1}$ $\quad x_0 = 1, y \in [0,1]$ $\quad h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

Cost function : Linear Regression $\quad J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$

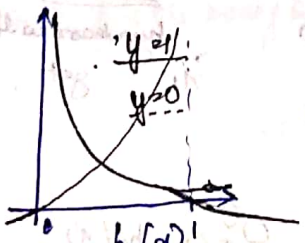


Logistic Regression Cost function :

$$\text{Cost}\left(h_\theta(x), y\right) = \begin{cases} -\log\left(h_\theta(x)\right) & \text{if } y = 1 \\ -\log\left(1 - h_\theta(x)\right) & \text{if } y = 0 \end{cases}$$

for single training example

we know, $h_\theta(x)$ values between 0 and 1



(i) Cost = 0, if $y = 1, h_\theta(x) = 1$
But as $h_\theta(x) \to 0$, cost $\to \infty$

(ii) Cost = 0, if $y = 0, h_\theta(x) = 0$
But as $h_\theta(x) \to 1$, cost $\to \infty$

Simplified Cost function and Gradient Descent

$\rightarrow J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Cost}\left(h_\theta(x^{(i)}), y^{(i)}\right)$

$$\rightarrow \text{Cost}\left(h_\theta(x), y\right) = \begin{cases} -\log\left(h_\theta(x)\right) & \text{if } y = 1 \\ -\log\left(1 - h_\theta(x)\right) & \text{if } y = 0 \end{cases}$$

Note $y = 0$ or 1 always.

compactly $\rightarrow = -y \log\left(h_\theta(x)\right) - (1-y) \log\left(1 - h_\theta(x)\right)$

$$\therefore J(\theta) = -\frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + \left(1 - y^{(i)}\right) \log\left(1 - h_\theta(x^{(i)})\right)\right]$$

To fit parameter vector θ:

$\min_\theta J(\theta)$ $\xrightarrow{\text{hypothesis will be}}$ $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

$\rightarrow$ Use gradient descent Repeat $\theta_j := \theta_j - \frac{\alpha}{m} \frac{\partial}{\partial \theta_j} J(\theta)$

Repeat {

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

}

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

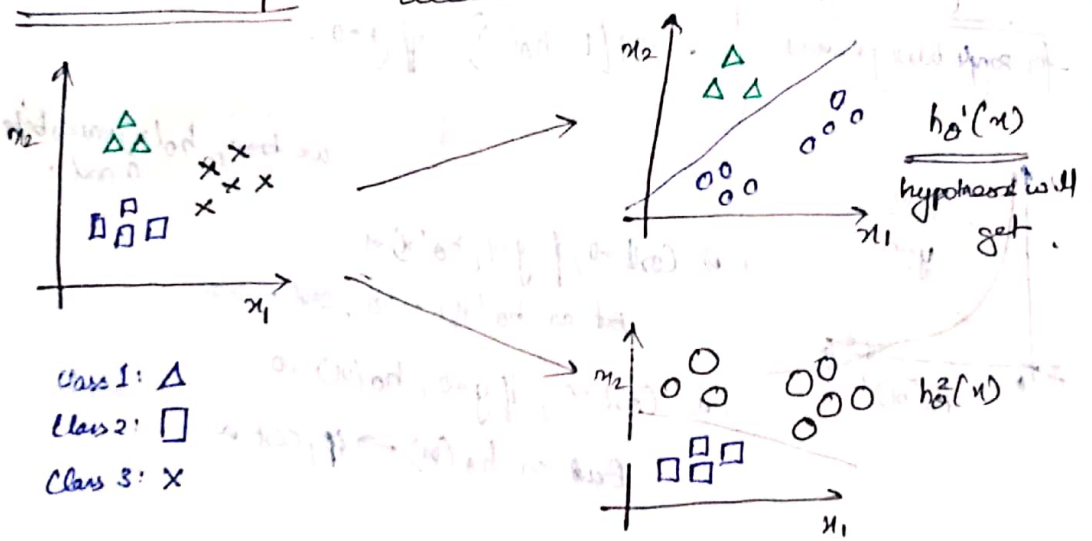
Advanced Optimization Concepts

Optimization Algorithms
- Gradient descent
- Conjugate gradient
- BFGS
- L-BFGS

Advantages:
- No need to manually pick α
- Often faster than gradient descent

Disadvantages:
- More complex.

Multiclass - Classification - One Vs all



→ we have 3 classes, $h_\theta^{(i)}(x)$ train for each class i to predict prob $y = i$

→ on a new input x, to make a prediction, pick the $h_\theta^{(i)}$ that maximizes the maximum value.

Overfitting - Regularization.

The problem of Overfitting -

→ Overfitting: If we have too many features, the learned hypothesis may fit the training set very well, but fail to generalize to new examples.

→ Addressing overfitting

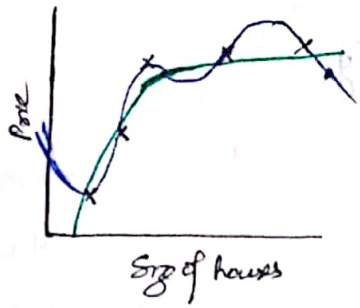
i) plotting the hypotheses.

ii) try to reduce the number of features (drop some using manual selection).

iii) Model Selection Algorithm.

(iv) - keep all the features, but reduce magnitude / values of parameter θ_j
- Works well when we have a lot of features, each of which contributes a bit to predicting y.

Regularization - Cost function



now in order to approx to θ_0, we would have to reduce values of $\theta_j \to 0$.

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

fits the training set

to simplify all my parameters.

→ regularization parameter.

keep parameter less.

If $\lambda \to \infty$, we start underfitting.

Regularized Linear Regression

→ Gradient descent

Repeat { $\theta_j := \theta_j - \alpha \frac{1}{m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)x_j^{(i)}$ $(j = 0, 1, 2, 3, \ldots n)$

}

Regularized objective

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

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

}

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

< 1 (a bit less than one)

→ Normal Equation

$$X = \begin{bmatrix} \cdots (x^{(1)})^T \cdots \\ \vdots \\ \cdots (x^{(m)})^T \cdots \end{bmatrix} \quad y = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(m)} \end{bmatrix}$$

$$\theta = \left(X^T X + \lambda \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}_{\text{size}(n+1)} \right)^{-1} X^T y.$$

Non-invertibility

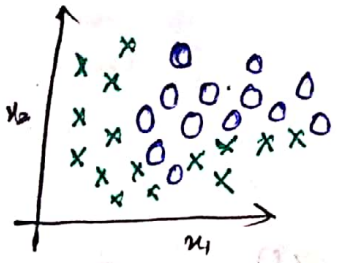
Suppose $m \leq n$

(#examples) - (#features)

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

↳ non-invertible / singular.

Regularized logistic Regression



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \cdots)$$

Cost function:

$$J(\theta) = -\left[\frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}) \right.$$

→ to Regularize

$$J(\theta) = J'(\theta) + \left[\lambda / 2m \sum_{j=1}^{n} \theta_j^2 \right]$$

$J(\theta)$ — cost function

gradient descent

Repeat {

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

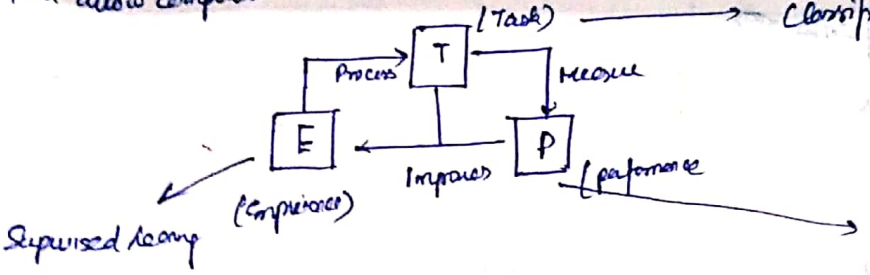
$$\theta_j := \theta_j - \alpha \left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} + \frac{\lambda}{m} \theta_j \right.$$

}

DKV - ML Introduction

Machine learning

- A branch of artificial intelligence, concerned with the design and development of algorithms that allow computers to evolve behaviours based on empirical data.



→ Classification, Regression, Clustering

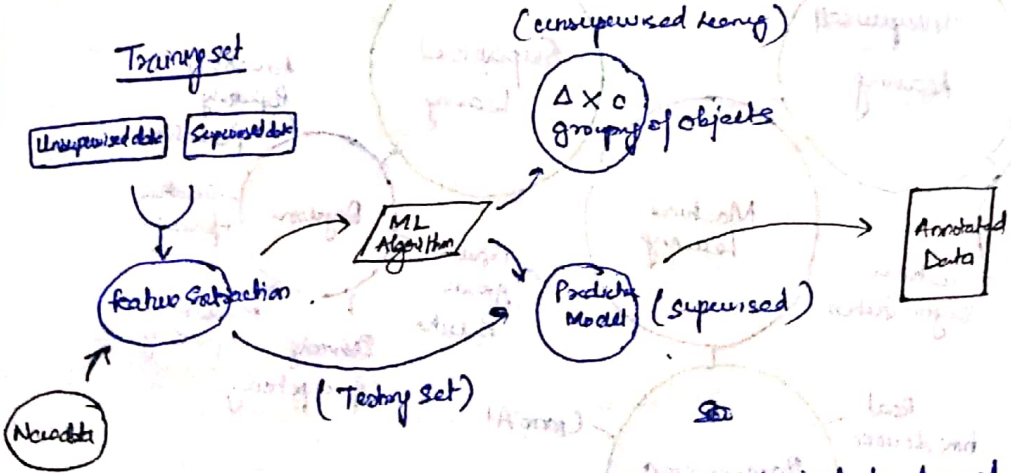
Why ML

- No human expertise.
- Black-Box human expertise (Data Abstraction)
- Rapidly changing phenomena
- Need for customization / personalization.

→ Every machine learning algorithm has three components:

- Representation
  - How to represent knowledge.
  - Eg: decision trees, sets of rules, neural networks, support vector machines, model ensembles
- Evaluation
  - Way to evaluate hypothesis.
  - Eg: Accuracy, prediction and recall, squared error, likelihood, posterior probability, cost margin.
- Optimization
  - The way candidate programs are generated (Search process).
  - Eg: combinatorial optimization, convex optimization, constrained optimization.

System



- Supervised learning (Inductive learning): Training data includes desired outputs.
- Unsupervised learning: Training data does not include desired outputs. It is hard to tell whether it is good learning or not.
- Semi-Supervised: Training data includes a few desired outputs
- Reinforcement learning: Rewards from a sequence of actions.

Representation of learning Algorithms.

i) Decision tree : Used for both classification and Regression.
(a classifier) → from a training set, we come up with a decision tree.

ii) Set of Rules / Logic programs

iii) Instances / lazy learners

iv) Graphical Models

v) Neural Networks

vi) Support Vector machines

vii) Model ensembles

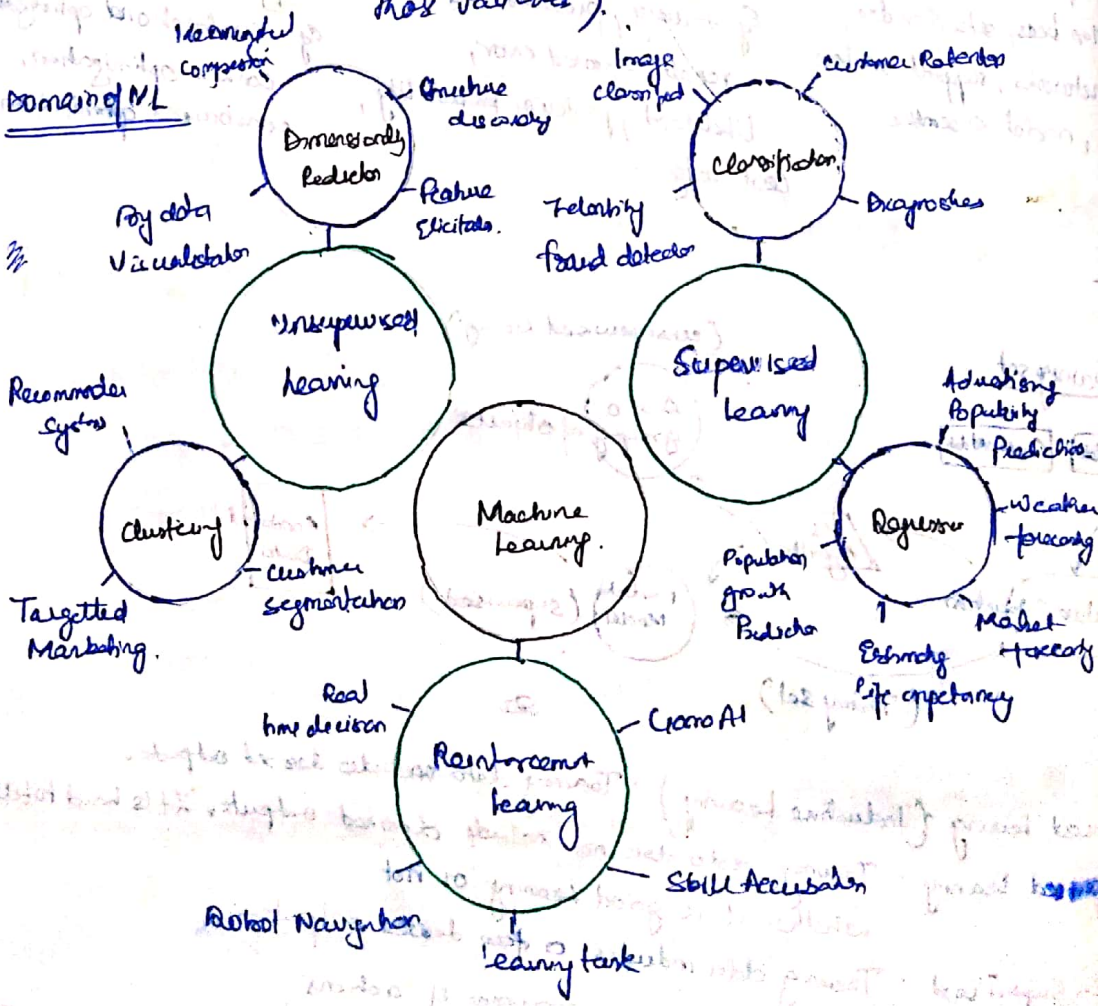
Optimization

Convex optimization
· Gradient descent
(finding function minimum)

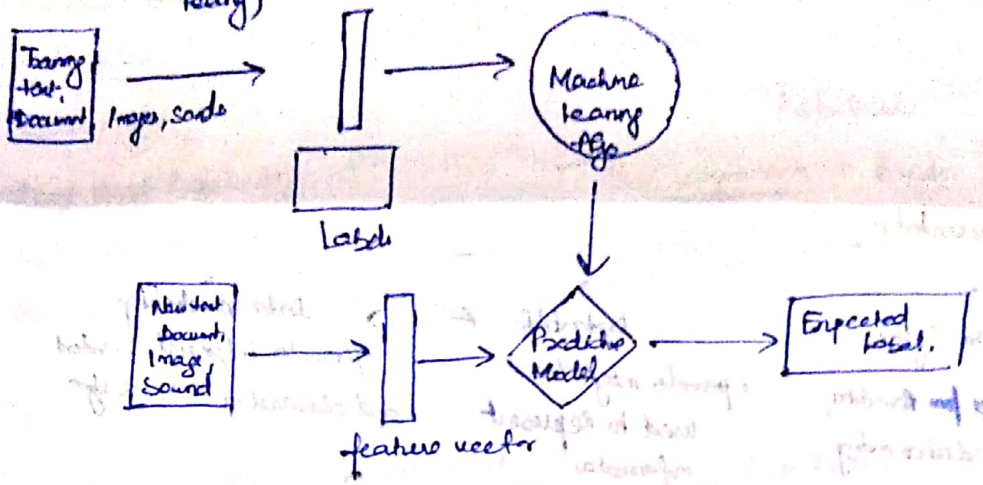
Constrained Optimization
· Linear programming.
(Optimizing an objective function with respect to some variables in presence of constraint on those variables).

Combinatorial optimization
· Greedy search
(finding optimal object from finite set of objects)

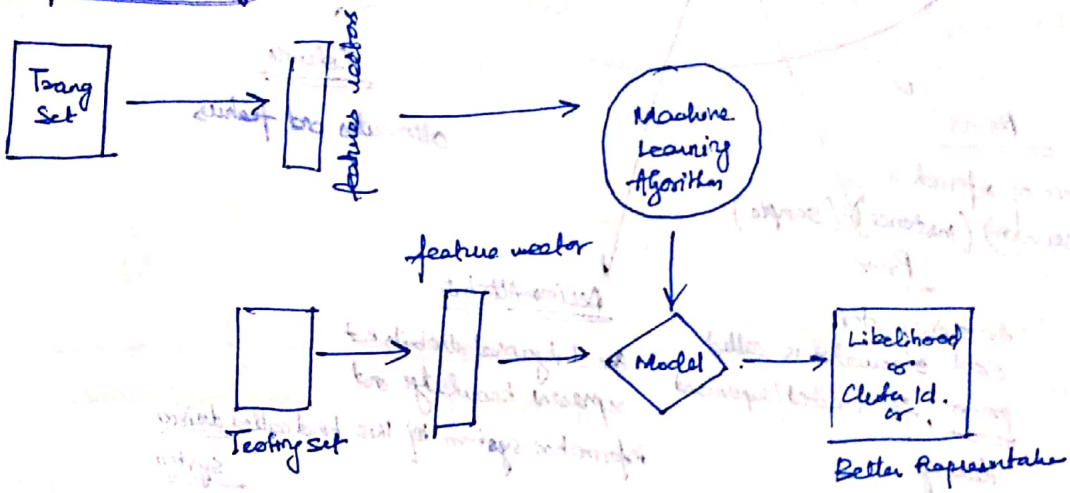
Domain of ML



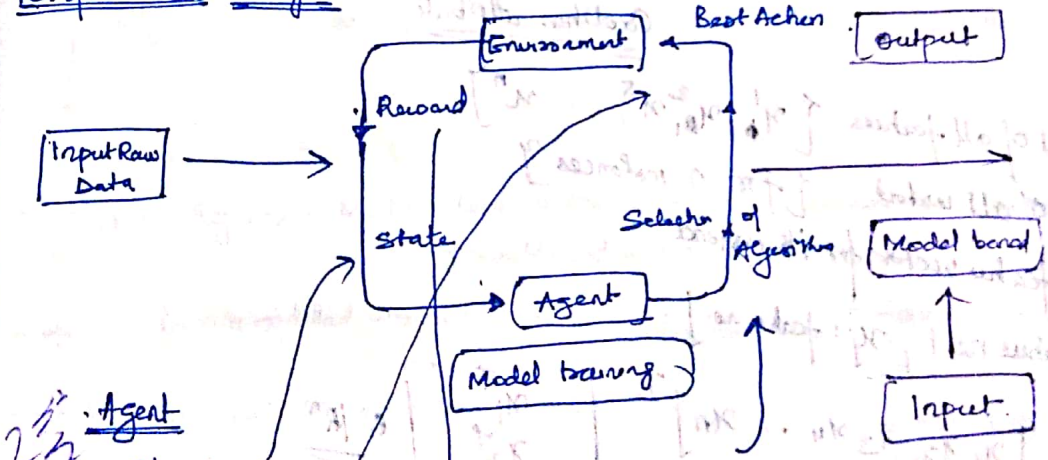
Supervised learning (Induction learning) feature vectors



Unsupervised learning



Reinforcement learning



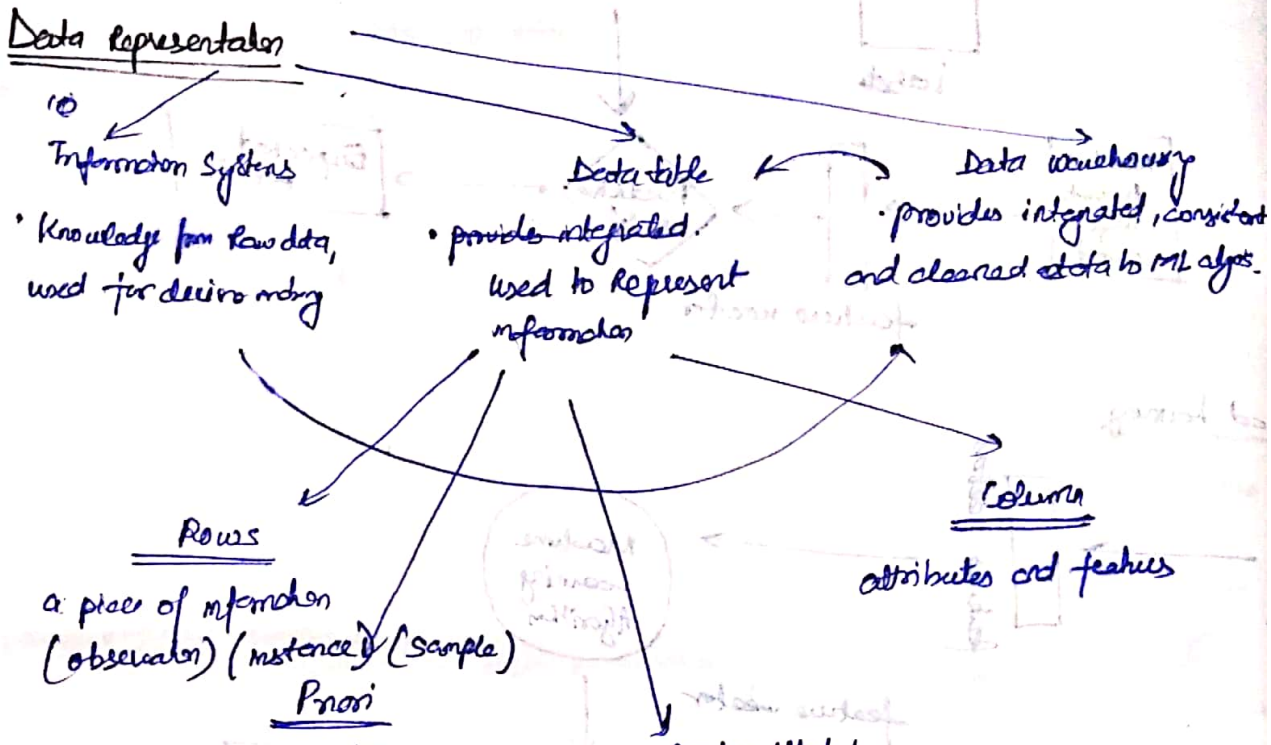
2½. Agent

1. Observe
2. Select Action using policy (uses a policy gradient)
3. Action.
4. Get Reward.
5. Update policy (learning step)
6. Iterate to get optimal policy.

Training

- Process of making system able to learn.
- No free lunch Rule:
  - Training set and testing set come from some distribution
  - Need to make some assumption or bias

Data Representation



X: set of all features $[x_1^1, x_2^1, x_1^2, x_1^3 \ldots x^n]$

S: Set of all instances [for n instances]

$x^{(i)}$: feature vector for i^{th} instance.

x_1: feature no. 1, x_j: feature no. j.

$$x^{(i)} = [x_1\ x_2\ x_3\ x_4 \ldots x_n]^T = \begin{bmatrix} x_1^T \\ x_2^T \\ x_3^T \\ \vdots \end{bmatrix} \in \mathbb{R}^n$$

State space of dimension n

(also visualised as X region: instance space $\mathbb{R}^n$)

Decision system

a tuple of (S, X, Y)

↳ Decision attribute domain.

Domain for a feature $i = \{x_j^1, x_j^2 \ldots x_j^?\}$

LINEAR REGRESSION - DKV.

Mathematical Models

Deterministic

→ Hypothesis exact Relationship.

→ For Negligible Prediction Error.

Eg: BMI from height and weight.

Probabilistic

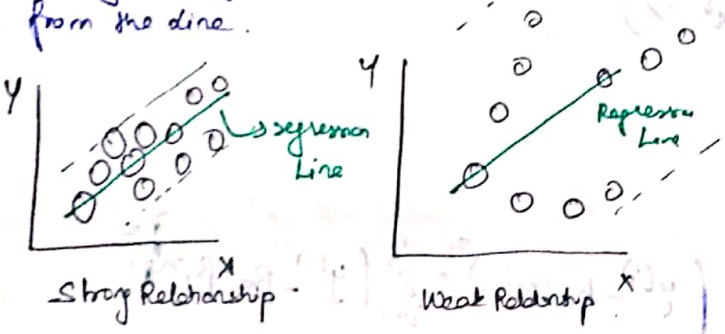
→ Two components + Random Error (ε)

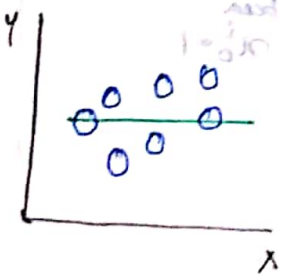
Deterministic (Hypothesis function)

Regression Modelling

→ γ: stands for calculation of Coefficient.

→ Strength of Relationship can be checked by noticing the longest perpendicular distance range from the line.





$h_\theta(x) = \theta_0 + \varepsilon$.

Steps

1. Hypothesis deterministic Components
(Eg: → finding suitable features and their coefficients)
2. Specify probability distribution of Random Error term.
(Calculate minima of cost function or Estimate standard deviation of Error)
3. Evaluate the fitted model
4. Use Model for prediction & Estimation.

→ Linear Regression Model: Used to show the linear relationship between a dependent variable and one or more independent variables.

→ • Data is modelled using a straight line.

→ Logistic: Data Modelled using sigmoid.

To Measure Accuracy

→ Accuracy
→ Precision
→ Recall
→ F1 Score
→ ROC curve
→ Confusion Matrix.

Linear Regression Model

$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$ ← Random Error.

- Y_i: Dependent Variable (Response / Label)
- β_0: population Y-intercept
- β_1: population slope.
- X_i: Independent Variable (Explanatory) (feature)
- Best hypothesis, which minimizes cost function

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

m → sample space
x^i → feature vector of.

Diff notation

$\beta_0 \to \theta_0$

$\beta_1 \to \theta_1$

$\underline{\beta} \to \underline{\theta}$

vector vector

$$\sum_{i=1}^{m} \left(y^i - h_\theta(x^{(i)}) \right)^2 = \sum_{i=1}^{m} \xi_\theta^{(i)^2}$$

→ Least Square Method minimizes the sum of the squared differences (errors) [SSE]

Final Calculation.

hypothesis (predicted) equation (For single feature)

$$\therefore h_\theta(x) = \theta_0 + \theta_1 x$$

$$\theta_1 = \frac{SS_{xy}}{SS_{xx}} = \frac{\sum_{i=1}^{m} (x_i^i - \bar{x})(y^i - \bar{y})}{\sum (x^i - \bar{x})^2}$$

$$\theta_0 = \bar{y} - \theta_1 \bar{x} \to \text{mean}$$

n x^2 → mean.

Derivation: LS (Not gradient descent)

$$\sum_{i=1}^{n} \xi^{(i)^2} = \sum_{i=1}^{n} \left(y^{(i)} - h_\theta(x^{(i)}) \right)^2 = \sum_{i=1}^{n} \left(y^{(i)} - \beta_0 - \beta_1 x_1^{i^2} \right)$$

$$0 = \frac{\partial \sum \xi_i^{(i)^2}}{\partial \beta_0} = \frac{\partial \sum \left(y^{(i)} - \beta_0 - \beta_1 x_1^i \right)^2}{\partial \beta_0}$$

beca $x_0^i = 1$

$$= -2 \left(n\bar{y} - n\beta_0 - n\beta_1 \bar{x} \right)$$

or $\beta_0 = \bar{y} - \beta_1 \bar{x}_1$ ——— (1)

$$0 = \frac{\partial \sum \xi_i^2}{\partial \beta_1} = -2 \sum x_1^i \left(y^{(i)} - \beta_0 - \beta_1 x_1^{(i)} \right)$$

$$= -2 \sum x_1^{(i)} \left(y_0^{(i)} - \bar{y} + \beta_1 \bar{x}_1 - \beta_1 x_1^i \right)$$

$$\to \beta_1 \sum x^i (x^i - \bar{x}_1) = \sum x^{(i)} \left(y^{(i)} - \bar{y} \right)$$

$$\therefore \beta_1 \sum (x^i - \bar{x}_1)(x_i - \bar{x}_1) = \sum (x_1^{(i)} - \bar{x}_1)(y^{(i)} - \bar{y})$$

$$\beta_1 = \frac{SS_{xy}}{SS_{xx}}$$

Sum of Square of error

parameter estimation Quick

Calculate table, $x^{(i)}$, $y^{(i)}$, $x^{(i)2}$, $y^{(i)2}$, $x^{(i)}y^{(i)}$, $(\sum x^{(i)})(\sum y^{(i)})$

now, $SS_{xy} = \sum_{i=1}^{n}(x_1^{(i)} - \bar{x}_1)(y^{(i)} - \bar{y}) = \sum_{i=1}^{n} x_1^{(i)} y^{(i)} - \frac{(\sum x^{(i)})(\sum y^{(i)})}{n}$

$SS_{xx} = \sum_{i=1}^{n}(x_1^{(i)} - \bar{x}_1)^2 = \sum_{i=1}^{n} x_1^{(i)2} - \frac{(\sum x^{(i)})^2}{n}$

Example

Fish no	Bird weight
1	1
2	1
3	2
4	2
5	4

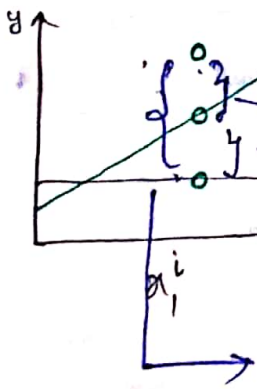
$SS_{xy} = (1+2+6+8+20) - \frac{(15)(10)}{5}$

$= 37 - 30 - 7$

$SS_{xx} = (1+4+9+16+25) - \frac{(15)^2}{5}$

$= 55 - \frac{45}{5}$

Goodness: Variation Measures (i) Estimation of σ^2



$\sum_{i=1}^{n}(y^{(i)} - h_\theta(x^{(i)}))^2$ [unexplained sum of squares]

$\sum_{i=1}^{n}(h_\theta(x^{(i)}) - \bar{y})^2$ [explained sum of squares]

Total sum of Square $SS_y = \sum_{i=1}^{n}(y^{(i)} - \bar{y})^2 = \sum y^{(i)2} - \frac{(\sum y^{(i)})^2}{m}$

Sum Squared error = Unexplained Sum of squares = SSE

$s^2 = \frac{SSE}{n-2}$

Standard deviation of error $\leftarrow s = \sqrt{s^2} = \sqrt{\frac{SSE}{n-2}}$

(ii) Residual Analysis

$e_i = y^{(i)} - h_\theta(x^{(i)})$ [difference between observed & predicted value]

→ check assumption of Regression by examining the residuals

- Examine for linearity assumption → residuals vs x
- Evaluate independence assumption → residuals vs x (Independent)
- Evaluate normal distribution assumption
- Examine for constant variance for all levels of X (Homoscedasticity) for all values of x.

Evaluating the Model - Testing for Significance

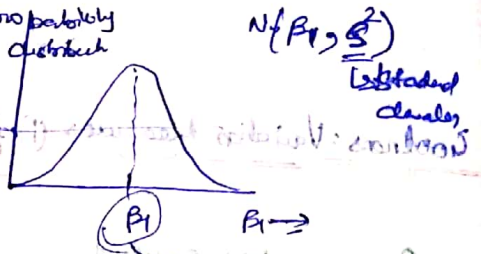
Regression Modelling Steps

1. Hypothesize deterministic component
2. Estimate unknown model parameters
3. Specify probability distribution of random error term.
   - Estimate $\sigma = \sqrt{\frac{SSE}{n-2}}$
4. Evaluate Model
5. Use Model for prediction and estimation.

(i) Test of Slope Coefficient

- Shows if there is a linear relationship between x and y
- Involves population slope β_1
- Hypotheses
  - → $H_0: \beta_1 = 0$ (No linear Relationship)
  - → $H_1: \beta_1 \neq 0$ (Linear Relationship)
- Sample distribution of slope
- $h_0(x)$: population line $(\Theta_0 + \Theta_1 x)$

Ideal graph:



→ Slope coefficient test statistic

$$t = \frac{\hat{\beta}_1 - \beta_1}{S_{\hat{\beta}_1}} = \frac{\Theta_1}{S_{\Theta_1}} = \frac{\Theta_1}{\sigma / \sqrt{SS_{xx}}}$$

$$SS_{xx} = \sum_{i=1}^{n} (x^{(i)} - \bar{x})^2 = \sum_{i=1}^{n} x^{(i)2} - \frac{\left(\sum_{i=1}^{n} x^{(i)}\right)^2}{n}$$

Degree of freedom = $n-2$ $\quad \alpha = 0.05$

Critical value: $\pm$ Find out from graph.

If t in range of critical value, accept that there is No linear Relationship, ie H_0 is true

If not, assume there is a linear relationship

(ii) CORRELATION MODELS

→ The correlation coefficient r estimates the true linear relationship between two variable

- Some correlation coefficient denoted r.
- Values range from -1 to +1
- Measure degree of association
- Does not indicate cause effect relationship.

$$r = \frac{SS_{xy}}{\sqrt{SS_{xx} SS_{yy}}}$$

(iii) Coefficient of Determination

Proportion of variation explained by relationship between x and y.

$$r^2 = \frac{\text{Explained Variation}}{\text{Total Variation}} = \frac{SS_{yy} - SSE}{SS_{yy}} \quad (0 \le r^2 \le 1)$$

(Coefficient of correlation)^2

ie, determination coefficient = (Coefficient of Correlation)^2

Interpretation: About r^2 of the sample variation in y can be explained by using x to predict ~~Sales~~ y in the linear model.

LOGISTIC REGRESSION

(i) Extend idea of linear Regression to categorical classification. y takes two values: Success (1), failure (0)

x should be continuous independent variable

(ii) Categorical Response: 2 level → binary: 0 and 1; 3 level or more → Nominal or ordinal

x_i can be binary, categorical or continuous.

(iii) Logistic curve is a sigmoid curve

given by function $h_\theta(x^{(i)}) = \frac{1}{1 + e^{-\theta^T(x^{(i)})}}$ ~~$\theta^T(x^{(i)})$~~

$\because \theta^T(x^{(i)}) = \ln\left(\frac{h_\theta(x^{(i)})}{1 - h_\theta(x^{(i)})}\right)$

LOGIT transformation

(iv) Logit transformation

~~$\theta^T(x) = \text{logit}$~~

$\theta^T x^{(i)} = \text{logit}(h_\theta(x^{(i)})) = \ln\left(\frac{h_\theta(x^{(i)})}{1 - h_\theta(x^{(i)})}\right)$

natural log; odds ratio; log odds ratio.

(v) Steps

(i) • estimate of the probability of belonging to each class. $P(y=1)$
• probability of belonging to class 1

(ii) • a cutoff value of these probabilities in order to classify each case to one of the classes
• a cutoff of 0.5 means that cases with $P(y=1) > 0.5$ belong to class 1.
• Cutoff need not be at 0.5 always

(vi) Types — Binary: 2 possible outcomes
— Multinomial: Three or more categories without ordering (Veg, Non-Veg, Vegan)
— Ordinal: Three or more categories with ordering.

Convolutional Neural Networks

(I) Layer – Convolutional
– ReLU Layer
– Pooling
– Fully Connected.

(II) CNN compares the images piece by piece. The pieces that it looks for are called features.
- We put small pieces of the bigger image. i.e. we choose a feature and put it on the input image, if it matches the the image is classified correctly.

(III) Convolutional layer ($\otimes$)
- Move filter on the image at every possible position
- Multiply Add and Average → put value in the new bigger image (reduced dimension)

(IV) ReLU : Activation layer. ( )
- In this layer we remove every negative value from the filtered images and replace with zero's.
- This is done to avoid the values from summing upto zero.

$$f(x) = \begin{cases} 0, & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases}$$

→ ReLU (Rectified Linear Unit) transforms function only activates a node if the input is above a certain quantity

(V) Pooling Layer. – Shrinking image stand into a smaller size

Steps
(i) Pick a window size (usually 2 or 3)
(ii) Pick a stride (usually 2)
(iii) walk window across your filtered images
(iv) for max pool – take max value

(VI) fully connected layer
- This is where final classification happens. – give weights to different layers.
- then we flatten our final output layer into vector

final : Prediction : Calculate the sum for both classes
↓
of activation probabilities indeed based on class activation values

the highest one wins

Bayesian Decision Theory

→ Design classifier to recommend decisions that minimize some total expected "risk" (cost or classification error)

terminology
- random variable : state of nature ω : ω_1 for sea bass, ω_2 for salmon
- Probabilities $P(\omega_1)$ and $P(\omega_2)$ (priors).
- Probability density function $p(\underline{x})$ (evidence) : → lightness
  feature value

- Likelihood, $P(x/\omega_1)$ → pattern belongs to ω_i
  given feature vector x → Poors

∴ different distributions : $P(x/\omega_i)$ ⟶ can be salmon / sea-bass

- Conditional probability $P(\omega_j/x)$ (posterior):

Decision Rule ⟶ $P(error) = \min[P(\omega_1), P(\omega_2)]$

if we decide ω_2 ← if we decide ω_1

feature vector corresponds to ω_2 class

This rule will be making same decision all time
- optimum if no other information is available.

Decision rule using conditional probability ⟶ $P(\omega_j/x) = \frac{P(x/\omega_j)\,P(\omega_j)}{P(x)} = \frac{\text{likelihood} \times \text{prior}}{\text{evidence}}$

$P(x) = \sum_{j=1}^{2} P(x/\omega_j)\,P(\omega_j) = 1$ (most of the times)

Probability of error ⟶ (i) $P(error/x) = \begin{cases} P(\omega_1/x) & \text{if we decide } \omega_2 \\ P(\omega_2/x) & \text{if we decide } \omega_1 \end{cases}$

or $P(error/x) = \min[P(\omega_1/x), P(\omega_2/x)]$

(ii) Average probability error

$$P(error) = \int_{-\infty}^{\infty} P(error, x)\,dx = \int_{-\infty}^{\infty} P(error/x)\,p(x)\,dx$$

$P(error \cap x)$

Bayes Rule is optimum (minimizes average probability error).

Where do probabilities come from ⟶ (i) Relative frequency (objective) approach (from experiments)

(ii) Bayesian (Subjective) approach
- may reflect degree of belief and can be based on opinion

Prior and Posterior Probabilities

- $P(A)$ and $P(B)$ are called prior probabilities
- $P(A|B)$, $P(B|A)$ are called posterior probabilities
- likelihood $P(x|\omega_i)$ is called class conditional prob.

process

- Collect data
- determine prior $P(\omega_i)$
- determine likelihood $P(x|\omega_i)$
- Calculate posterior for each class $P(\omega_i|x) = \frac{P(x|\omega_i)P(\omega_i)}{P(x)}$

(x for one class)

$P(x) = \sum_{i=1}^{n} P(x|\omega_i)P(\omega_i)$

or $P(x|\omega_1)P(\omega_1) + P(x|\omega_2)P(\omega_2)$

- General → allow rejection category
- Employ a more general error function

risk function → assoc. a "cost" or "loss" function with each error

Loss function $\lambda(\alpha_i|\omega_j)$

- A finite set of actions $\alpha_1, \alpha_2, \ldots \alpha_l$
- loss function
  - the cost associated with taking action α_i when correct classification category is ω_j.

Conditional Risk or Expected loss

Conditional risk with taking action α_i is

$$R(\alpha_i|x) = \sum_{j=1}^{c} \lambda(\alpha_i|\omega_j) P(\omega_j|x)$$

↓ correct category assigned

Overall Risk

- Suppose $\alpha(x)$ is general decision rule that determines which action $\alpha_1, \alpha_2, \ldots \alpha_l$ to take for each x;

overall risk

$$R = \int R(\alpha(x)|x) P(x)\,dx.$$

determine or basis of x, what decision to take.

- The optimum decision rule is the Bayes rule. When overall risk is least.

minimizes R by

(i) computing $R(\alpha_i|x)$ for α_i given x

(ii) choosing the action α_i with the minimum $R(\alpha_i|x)$.

$R^* = \min R$ ← Bayes Risk

$\omega_4 = \frac{2}{4} \times \frac{4}{10} \times 2$

$= \frac{1}{2}$

Minimum Risk decision rule

Decide w_1 if $R(a_1/x) < R(a_2/x)$; else decide w_2

Zero one loss function → $\lambda(\alpha_i/w_j) = \begin{cases} 0 & i=j \\ 1 & i \neq j \end{cases}$

$$R(\alpha_i/x) = 1 - P(w_i/x)$$

Thus overall risk is the average probability error.

Evaluation of Classifiers

→ Params: Precision, Recall, Accuracy, F-Measure, TPR, FPR, Sensitivity, ROC

learning curve →
- How does the accuracy of a learning method change as a function of the training set size? (Accuracy vs. Sample size)
- Can be assessed by plotting learning curve
- plot $(\frac{S}{2}, a)$ or $(S, \text{avg accuracy and error bars})$ (S: sample size)

Validation (Tuning) set →
- Consider we want unbiased estimate of accuracy during the learning process (e.g. to decide the best level of decision-tree pruning)
- ↓ training data is partitioned into separate validation/training sets

Limitations of single hold-out / Test partition →
- Larger sets give more reliable estimate of accuracy (Lower variance estimate)
- A single training set doesn't tell us how sensitive accuracy is to a particular training sample

Random Sampling →
- Repeatedly randomly partitioning the available data into training and testing sets
  - (class proportions maintained in each set)
  - ↳ Stratified sampling
  - ↳ training, validation and test (e.g. positive/negative = 3/2)

Cross validation →
- Or Partition data into n subsamples

Internal Cross validation →
- Instead of a single validation set, we can use cross-validation within a training set to select a model.

Precision and Recall

Precision: fraction of retrieved docs that are relevant

$= P(\text{relevant} \mid \text{retrieved})$

$= \frac{tp}{tp+fp} = \frac{a}{a+c}$

Recall: fraction of relevant docs that are retrieved

$= P(\text{retrieved} \mid \text{relevant})$

$= \frac{tp}{tp+fn} = \frac{a}{a+b}$

Precision Recall curve plots precision vs recall as a threshold on the confidence of an instance being positive is varied.

	+	−
+	$tp(a)$	$fn(b)$
−	$fp(c)$	$tn(d)$

Default prec $= \frac{a+b}{a+b+c+d}$

Ideal pnt TPR = Prec = Recall = 1.0

Issues: • Prec-Recall set may give same prec vs recall (does not take tn into account)

Combined Measure (F)

• Assesses precision/recall tradeoff is F measure (weighted harmonic mean)

$$F = \frac{1}{\alpha \frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

Balanced F measure is with $\alpha = \frac{1}{2}$

Accuracy

• $\text{Accuracy}(\%) = \frac{tp+tn}{(tp+tn+fn+fp)} \times 100 = \frac{a+d}{a+b+c+d}$

• Misleading with non-uniform distribution set

Recognition

Positive recognition rate: $\frac{a}{a+b+c+d}$

Negative recognition: $\frac{d}{a+b+c+d}$

Accuracy Adequacy

• Accuracy not good if there is large class skew

• different misclassification costs

• We are most interested in a subset of high-confidence predictions

Sensitivity Specificity

$\text{Sensitivity} = \frac{TP}{TP+FN} = \frac{a}{a+b} = P(\text{retrieves} \mid \text{relevant})$

$\text{Specificity} = \frac{TN}{TN+FP} = \frac{d}{b+c} = P(\text{not to be retrieved} \mid \text{not relevant})$

TPR

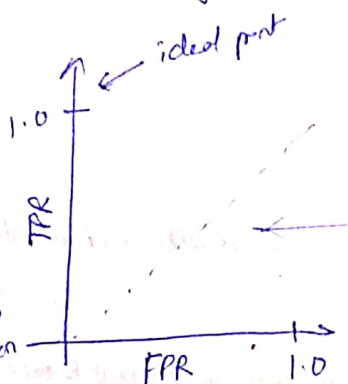
True positive rate / Recall

and False positive rate

$$TPR = \frac{TP}{TP + FN} = P(\text{retrieved} \mid \text{relevant}) = \frac{a}{a+b}$$

$$FPR = \frac{FP}{\text{actual neg}} = \frac{c}{c+d} = P(\text{false retrieval} \mid \text{non relevant})$$

ROC / AUC curve → Receiver Operation Characteristic (ROC) & Area Under curve plots the TPR/Sensitivity/Recall vs the FPR as threshold on the confidence of an instance being positive is varied.



Cost is associated with b and c. (fp and fn)

Expected curve for random guessing

Confidence ~~percentage error~~ ~~confidence that~~

95% confidence interval defines a range of values that you can be 95% confident contains the population mean.

- Mere Robust in class imbalanced situations
- 10 Negative : FP costs 10 × FN
  10 Positives : FN = 10 × FP
  Equal : Cost of misclassifying positive and negative is equal.

Creation : • Sort test set predicted probability to confidence that each instance is positive

- Step through the list from high to low confidence
  - → Locate a threshold between instances with opposite classes
  - → Compute TPR and FPR for instances above threshold
  - → Output (FPR, TPR) coordinate.

Precision/recall curve → • plots precision vs recall as a threshold on the confidence of an instance being positive is varied.

PYTHON

(i) eda on tent → load, mob, fe...

WEEK1

Operator and Operands

FunctionCalls

-> type(data types) :
return class of data type

→ String literals:

- i) ' '
- (ii) " "
- (iii) """ """
- (iv) ''' '''

- if you know the string going to have a quotation mark in between,
  → use triple quotation marks.
- Triple Quotation marks, allow us to write strings in lines

Help Section

```
>> dir(__builtins__)
>> dir(datatype)
>> help( str.upper )
```

method

__x__

datatype

Type Conversion functions

→ int (3.14) → will covert 3.14 to 3

→ int (string) → (should not contain floating point)
if the string has all alphanumerics, it will be converted to integer.

~~→ Similarly a number can be converted into a string~~

Variables

Statements and Expressions

→ print(len("hello"))
↳ implicit function.

Hard Coding

Input

→ n = input ("Please enter your name: ") // always a string.

TURTLE GRAPHICS

Tue First turtle program

```
import turtle                    # allows us to use the turtles library
wn = turtle.Screen()             # creates a graphics window
alex = turtle.Turtle()           # create a turtle named alex.
alex.forward(150)                # tell alex to move forward by 150 units
alex.left(90)                    # turn by 90 degrees
alex.forward(75)                 # complete the second side of a rectangle.
```

alex is an instance of an object.

setting attributes

```
alex.salary = 50000
```

Instances: A Herd of Turtles

```
wn = turtle.Screen()
wn.bgcolor("lightgreen")
```

Repetition with a for loop — gap → proper indentation is required

```
for _ in range(3):
    print("This line will execute three times")
    print("This line will also execute three times").
```

gap

→ python range function start counting from 0 and ends at n-1

range(start, stop, step)

default, default

More Turtle Methods

```
import turtle
wn = turtle.Screen()
wn.bgcolor("lightgreen")
tess = turtle.Turtle()
tess.color("blue")
tess.shape("turtle")
dist = 5
tess.up()                        //doesn't put the line stamp
for _ in range(30):
    tess.stamp()
    tess.forward(dist)
    tess.right(24)
    dist = dist+2
wn.exitonclick()
```

Importing Modules

The random module → module

```
>>> import random
prob = random.random()   # a random number between 0-1
print(prob)

diceThrow = random.randrange(1,7)   # returns an int, one of 1,2,3,4,5,6
print(diceThrow)
```

OR

```
from random import randrange, random   # become local names
prob = random()
print(prob)
diceThrow = randrange(1,7)
print(diceThrow)
```

The way of a programmer

Syntax error

i) EOF in multi-line statement on line 3.
ie python can't figure out the syntax.

Runtime Errors

ii) Illegal operation.

Know your error messages

→ import math → math module in sth
math.sqrt(x)

Common Errors

Week 2

sequences - Strg

List : general sequences

```
myList = ["one", 2, "three"]
```

Tuples (immutable)

```
myTuple = ('one', 2, 'three')
```

= Square brackets with parenthesis

Note, myTuple = (100) would be an integer

To create tuple 100,

```
myTuple = (100,)
```

→ empty tuple

```
myTuple = ()
```

Index Operator

```
s = "Python"
print(s[0])
```

index begins from 0

→ len(s) would return length.

also s[-1] = n

moves from the back ie is equivalent to n-1 index.

The Slice Operator

Eg: julia = ["Julia", "Roberts", 1967, "Duplicity"], 2009, "Actress", "Atlanta, Georgia").

print(julia[2]) → 1967

print(julia[2:6]) → from 1967 to Actress

→ from index 2-5

print(len(julia)) → 7

julia = julia[:3] + ("Eat Pray Love", 2010) + julia[5:]

from 0,3

tuple

'julia', 'Roberts', 1967

tuple.

print(julia

slicing

---

alist = [3, 67, "cat", [56, 57, "dog"], [], 3.14, False]

Concatenation and Repetition

→ print([0]*4) = [0,0,0,0]

([0,1]*4) = [0,1,0,1,0,1,0,1]

Count and Index

→ a = "I have had an apple on my desk before!"

print(a.count("e"))

print(a.count("ha"))

a.index(item)

→ index finds where you item is.

→ if the item is not on list index is gonna give you an error.

Split & Join

→ song = "The rain in Spain..."

words = song.split() // returns a list.

→ if we call,

so words = song.split("e")

: ['l', 'ad', ...]

→ "/".join(["Loades", "ord", "Lest"]).
glue

Iteration

for loop

→ for name in ["Joe", "Amy", "Brad", "Angelina", "Zuki", "Thandi", "Paus"]:
only valid variable
print("Hi", name, "Please come to my party on saturday!")

→ for ocha in "Go spot Go":
print(ochar)
takes character by character.

The Accumulator pattern

nums = [1,2,3,4,5,6,7,8,9,10]
accumo
for no in nums:
→ tab accum = accum + no
print(accum)

* ~~sum~~(91)
won't include 11

The Range function

WEEK 3

Boolean Expressions

→ print(True) → Boolean datatype as different from strings.
print(type(True))

{True, False}

→ Comparison operator
- 5 == 5 returns True or false
- !=, <, <=, >, >=

logical operators

→ L and R | L and R should be boolean expression
L or R
* not B

The in and not in operators

→ print('p' in 'apple') → true

Similarly not in

Precedence of Operators

CONDITIONAL EXECUTION

Conditional Execution

i) x = 15

```
if x%2==0:
    print(x, "is even")
else:
    print(x, "is odd")
```

Omitting the Else Clause : Unay Sekhon

(i) x = 5
y = 10

```
if x<y: print('x is less than y')
else:
    if x>y:
        print('x is greater than y')
    else: print('x is equal to y').
```

or

explain elseif → elif

The Accumulator Pattern with Conditionals and Accumulating a maximum value

→ find largest number in this list of numbers.

WEEK 4

Mutability

→ list are mutable.

→ strings are non-mutable.

greeting = "Hello, world!"
newGreeting = 'J' + greeting[1:] (slicing)
print (newGreeting)
print (greeting).

List Element Deletion

→ a = ['one', 'two', 'three']
del a[1] — takes a position
print (a)

Objects and References

→ a = "banana"
b = "banana" — pointer for string

a is b : ~~true~~ ~~false~~ true , a == b is true — false for list

a and b point to same

→ a = [list] ~~pointer~~ : if bea — b and a a aliases
b = [list] — not aliases

Aliasing

Cloning list

a = [81, 82, 83]

b = a[:] — makes a clone copy too.

Methods on Lists and String

→ mylist.append(digit) — appends an item into the list

→ mylist.reverse()

→ mylist.insert(~~digit, position~~) (position, digit)

→ mylist.remove(digit)

→ lastitem = mylist.pop() pop. — returns the last element of

Append vs Concatenate

→ concatenate → creates a new list object

```
oiglist = [45,32,88]
oiglist = oiglist + ["cat"]
```

| += does not mutates

→ id(identifier) returns id in memory

Non Mutating Methods on Strings

-> str.strip()

removes any white elements in the end and the beginning

→ str.replace("0", "++")

↳ to_replace → replace_with

All methods on the string are non-mutable.

→ scores = [("Rodney Dangerfield", -1), ("Marlon Brando", 1), ("You", 100)]

```
for person in scores:
    name = person[0]
    score = person[1]
    print("Hello {}. Your Score is {}.".format(name, score))
```

String

→ formatting

{} {:.2f}

precision with 2 digits

Methods on lists and strings

Accumulating Lists and Strings

→ num = [3, 5, 8]

accum = []

```
for x in nums:
    x = (x+2)   → square
    accum.append(x)
print(accum)
```

Accumulator pattern with strings

```
s = input("Enter some text")
ac = ""
for c in s:
    ac = ac + c + "..." + c + "..."
```

```
x = [1,2,3]
y = x
x += [4,5]
y = (y + [6])
print(x)
print(y)
```

→ concatenates returns a new list

→ append does operation on the same

→ creates a new list

PYTHON FUNCTIONS, FILES, AND DICTIONARIES

FILES AND CSV OUTPUT

FILES

Reading a file

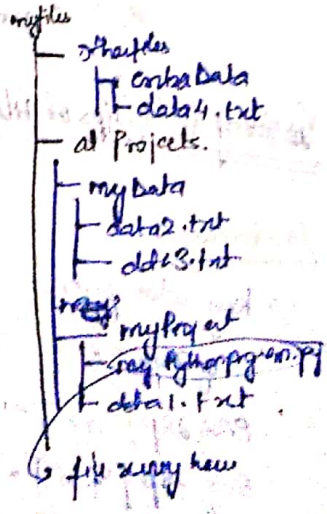
(i) filevar = open("olympics.txt", "r")
- "olympics.txt" = name of file
- "r" = what to do with file
- filevar = file object

filevar.close()

(ii) contents = filevar.read()
print(contents[:100])

(iii) lines = filevar.readlines()
print(lines[:4]) ⇒ returns a list, each string end with \n character.

or for ln in filevar:
    print(ln.strip())



FINDING A FILE IN YOUR FILES SYSTEM

Relative path
open('../myData/data2.txt', 'r')

Direct → open('/Users/pesnick/myfiles/all Projects/myData/data2.txt', 'r')

Working text files

'\n' is actually stores in file.

```
file_obj = open("squares.txt", "w")
for number in range(13):
    square = number * number
    file_obj.write(str(square))
file_obj.close()
```

or `file_obj.write(str(square) + '\n')`

with for files

```
with open('mydata.txt', 'r') as md:
    for line in md:
        print(line)
```

CSV format (comma seperated values)

Reading a data from CSV file

```
file_connection = open("olympics.txt", 'r')
lines = file_connection.readlines()
for ln in lines[:6]:
    print(ln.strip())
print("-------")
header = lines[0]
field_names = header.strip().split(',')
print(field_names)
for row in lines[1:]:
    vals = row.strip().split(',')
    if vals[5] != "NA":
        print("{}: {};".format(vals[0], vals[4], vals[5]))
```

DICTIONARIES

Dictionary

→ key value pair.

```
eng2sp = {}
eng2sp['one'] = 'uno'
eng2sp['two'] = 'dos'
eng2sp['three'] = 'tres'
print(eng2sp)
```

→ • dictionary are unordered.

ex code • capitals = { 'three': 'tres', 'one': 'uno', 'two': 'dos' }.

Dictionary basics / Dictionary operations

→ del inventory['pears']
↳ get rid of a key value pair.

num_items = len(inventory)
↳ number of key value pairs in a dictionary.

Dictionary methods

→ inventory = { 'apples': 430, 'bananas': 312, 'oranges': 525, 'pears': 217 }

for key in inventory.keys():
    print(key, "has the value", inventory[key])

keys = list(inventory.keys())
print(keys).

→ or, for key in inventory
    print(keys)
    only print keys.

→ list(inventory.values()) → only values

→ list(list(inventory.items()))
↳ list of tuples.

→ inventory.get("apples")
returns values.

also. inventory.get("cheeses") → returns None if not present

or inventory.get("cheeses", 0) → 0 instead of None

Dictionary-Accumulation

→ f = open('scarlet.txt', 'r')
text = f.read()
one long string containing all the characters.

Python full Tutorial

Identations

lines should be in the same vertical line.

Best practice is to use Tab

List

Array.

- Add more values in the list add append.

→ List.append('element')

→ mylist.pop()
removes last element

.pop(1) = at index 1.

→ list is mutable,
strings and tuples are not.

Slicing → getting partial information from your data.

——→ myfruits[1:3] // exclusive slice.
include 1, 2.

Set

→ only stores unique number

. we can convert a list into a set

SurveyAns = [3,6,6,5,8,3,5,1,6,9]
opinionSelected = set(surveyAns)
print(opinionselected)

{1,3,5,8,9}

for, set we have add., myset.add(3) won't append value

dichonary

nested dichonary → priceList = {'iphone8': {'price':600, 'size':4.7},
'iphone9': 700,
'iphoneX': 800,
'iphone11': 900}

we can use nested dichonary to create 2-D Arrays.

for loop

iteration → myfic = { 'Oreyo': 25, 'Apple': 56, 'Mongo': 65 }

```
for key, val in myfavits.item():
    print(key, val)
```

Functions

Catching Errors

→ atmPin = input('Please enter your pin') - user.

try: statement

except:
    pass.

Modules nd packages

→ Modules add functionality to our program

package is a group of modules

package
|
Sub-package
|
Module.

use → import utlis
one module. file or module.

or from utlis import sayHello
method name.

use → from Resources.MyModules.Utlis import sayHello
multiple folder main folder folder file. method.

Classes & Objects

made in a module

```
class Human:
    def __init__(self, name, age, height):
        self.name = name
        self.age = age
```

constructor

main file import

```
from Kumaclass import Human
human1 = Kuma('Mwitoge', 35, 6)
print(human1.age)
```

files

f = open('myFile.txt', 'r')
permission.
or 'w+'
'w'.

\ : special characters
/ : not special.

```
for x in range(0,10):
    f.write('Current line : ' + str(x))
                                or + '\n'
f.close().
```

append → f = open('myfile.txt', 'a+')

→ f.close - myfile.read()

CSV files

→ CSV files can have multiple delimiter values.

→ import csv
module.

```
with open('names.csv', 'r') as csv_file:
    csv_reader = csv.reader(csv_file)
    object in memory.
    for line in csv_reader:
        print(line)
```

a list or array. — or print(line[2]).

(or)

```
next(csv_reader)  → move read pointer
for line in csv_reader:
    print(line[2]).
```

to write →

```
with open('new_names.csv', 'w') as new_file:
    csv_writer = csv.writer(new_file, delimiter='-').
    write object.                    delimiter.
    for line in csv_reader:
        csv_writer.writerow(line)
```

- CSV reader automatically counts to skip to not confuse between delim

Using dictionary Reader and writer

→ with open('names.csv', 'r') as csv_file:
csv_reader = csv.DictReader(csv_file)

for line in csv_reader:
print(line)

[field names are now keys].

for line in csv_reader:
print(line['email']).

dict writer →

with open('new_names.csv', 'w') as new_file:
fieldnames = ['first_name', 'last_name', 'email']
csv_writer = csv.DictWriter(new_file, fieldnames = fieldnames, delimiter = '\t')
csv_writer.writeheader() # write fieldnames as first line

for line in csv_reader:
csv_writer.writerow(line)

{ or del line['email'] email does not exist.

open() function →

- r for reading
- w for writing
- a for appending
- r+ for both reading and writing.

PANDAS

Load data into Pandas →

import pandas as pd.
df = pd.read_csv('pokemon_data.csv')
df = dataframe
print(df)

Reading some data →

- df.head(10) top 10 rows
- df.tail(3) // bottom 3 rows.
- df = pd.read_csv('pokemon_data.txt', delimiter = '\t') tab separated file. ↑ to anything separating to your column.

Read header → print(df.columns) could also use df.Name.

Read col column → print(df['Name'][0:5]) (top 5 rows names)

or print(df[['Name', 'Type 1', 'HP']])

Print each Row → print(df.head(4))

also, print(df.iloc[1:4])

Reading specific location → print(df.iloc[2,1])

Iterate through rows →
```
for index, row in df.iterrows():
    print(index, row)  or
    print(index, row['Name'])
```

df.loc[df['Type 1'] == "Fire"]

Sorting / Describing Data →
- df.describe() → returns count, mean, std, min, 25%, 50%, 75%, max
- df.sort_values('Name')
  - for descending, ascending = False

for more tables ['Type 1', 'HP'] → ascending = [1, 0]
- Type 1 ascending
- HP descending.

Making changes to data →
- df['Total'] = df['HP'] + df['Attack'] + df['Defense'] + df['Sp. Atk'] + df['Sp. Def'] + df['Speed'] (new column.)
- df = df.drop(columns = ['Total']) (update data frame.)
- df['Total'] = df.iloc[:, 4:9].sum(axis=1) // adding horizontally (4 to 9 columns)

Reorder Columns →
- cols = list(df.columns.values) // adding vertically
- df = df[['Total', 'HP', 'Defense']] or
- df = df[cols[0:4] + [cols[-1]] + cols[4:12]] // → single column may be headed as list.

Saving our data → df.to_csv('modified.csv')
- index = False (does not store index.)

- df.to_excel('modified.xlsx', index=False)
- df.to_csv('modified.txt', index=False, sep='|')

filtering of the data → We can use '&' and '|' etc

- to reset index new_df = new_df.reset_index()
  - drop=True → drops previous index
  - or new_df.reset_index(drop=True, inplace=True)

→ Story filtering.

df.loc[df['Name'].str.contains('Mega')]

for reverse → df.loc[~df['Name'].str.contains('Mega')]

~ not

- Import ~~regex~~ re.

df.loc[df['Type1'].str.contains('fire|Grass', regex=True)]

Conditional Changes.

- df.loc[df['Type 1']=='fire', 'Type1'] = 'Flamer'.
  - based on condition.
  - Chnge Type 1
  - flags = re.I → ignore case.
- df.loc[df['Total']>500, ['Generation', 'Legendary']] = ['TEST VALUE']
  - both to test value

Groupby Aggregate.

df = pd.read_csv('modified.csv')

- df.groupby(['Type1']).mean() // averages for type 1 pokemon
- df.groupby(['Type1']).mean().sort_values('Defence', Ascending=False)
- df.groupby(['Type1']).sum()
  - based on.
- df.groupby(['Type1']).count() // won't work with NAN
- df['count']=1
  df.groupby(['Type1']).count()['count'] or
  df.groupby(['Type1','Type2']).count()['count']

Value counts → Returns the counts of values

df['Flair'].value_counts()

- df.to_excel('modified.xlsx', index=False)
- df.to_csv('modified.txt', index=False, sep='|')

Filtering of the data → We can use '&' and '|' or

- to reset index new_df = new_df.reset_index() → drop=True (drops previous index)
  or new_df.reset_index(drop=True, inplace=True)

→ String filtering.

for reverse: df.loc[df['Name'].str.contains('Mega')]
df.loc[~df['Name'].str.contains('Mega')] (~ = not)

- Import ~~regex~~ re.

df.loc[df['Type1'].str.contains('fire|Grass', regex=True)]

Conditional Changes. → flags = re.I (ignore case)

- df.loc[df['Type 1']=='fire', 'Type1'] = 'Flamer'.
  (based on condition. changes Type 1)
- df.loc[df['Total']>500, ['Generation', 'Legendary']] = ['TEST VALUE'] (both to test value)

Groupby Aggregate:

df = pd.read_csv('modified.csv')

- df.groupby(['Type1']).mean() // average for type 1 pokemon
- df.groupby(['Type1']).mean().sort_values('Defense', ascending=False)
- df.groupby(['Type1']).sum() (based on)
- df.groupby(['Type1']).count() // won't work NAN
- df['Count']=1
  df.groupby(['Type1']).count()['Count']
  df.groupby(['Type1','Type2']).count()['Count']

Value Counts → Returns the counts of values

df['Flair'].value_counts()

Large Amount of Data → df = pd.read_csv('modified.csv')

= or

pd.read_csv('modified.csv', chunksize = 5) // 5 files passed at a time

- for df in pd.read_csv('modified.csv', chunksize = 5):
- new_df = pd.DataFrame(columns = df.columns) — new data frame

```
for df in pd.read_csv('modified.csv', chunksize = 5):
    results = df.groupby(['Type 1']).count()
    new_df = pd.concat([new_df, results])
```

Column analysis

EDA

Libraries → import pandas as pd
import numpy as np
import seaborn as sns

Understanding the data
- data.head()
- data.shape()
- data.columns
- data.nunique() → unique values for all colums
- data.isnull().sum()
  check null values.

drop → student = data.drop(['race/ethnicity', 'parental level of education'], axis=1).

- Outlier - data point which differs significantly from its other. Can cause serious problems in statistical analysis.

Relationship analysis → i) Correlation matrix shows correlation coefficient between variables.

correlation = student.corr()

Heatmap ii) sns.heatmap(correlation, xticklabels = correlation.columns, yticklabels = correlation.columns, annot = True)

pair plot iii) sns.pairplot(student)

Scatterplot iv) sns.relplot(x = 'math score', y = 'reading score', hue = 'gender', data = student)

Histogram v) sns.displot(student['reading score'], bins = 5)
(shape, spread and continuous sample data)
(5 divisions)

box plot (vi) sns.catplot(x = 'gender', kind = 'box', data = student)
values of female & male.

Categorical values can be changed to numerical values.

Numpy

→ Numpy uses fixed type and all features than list

5 → 0000 0101 (Binary) → Numpy → 00000000 00000000 00000000 00000 (Int32)

List → Size, Reference, Object Type, Object Value

- Numpy is faster to read less bytes of memory.
- No type checking while moving through objects.

→ Numpy is contiguous memory.

- list contains pointers to information scattered across the memory.
- Numpy array uses contiguous memory.

Benefit: • SIMD Vector processing makes process faster.
• Effective Cache Utilization.

- Arrays in numpy

```
a = np.array([1,3,5])
b = np.array([1,2,3])
a*b = np.array([1,6,15])
```

Applications → • MATLAB Replacement
• Plotting (Matplotlib)
• Tensor libraries are connected to numpy.

(Sci-fi library also has math functions)

Basics →

```
a = np.array([1,2,3])        # One D array
print(a)
b = np.array([[9.0,8.0,7.0],[6.0,5.0,4.0]])   # 2D array   (8 bit floats)
a.ndim  # Get dimensions. (One d array, 2d array, 3d array)
a.shape()  # Returns shape.
a.dtype  # 'int32' by default
a = np.array([1,2,3], type='int16')
a.itemsize  # returns bytes.
a.size  # total number of items
a.nbytes  # total size
```

Accessing / Changing specific elements, rows, columns, etc.

- a = np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])
- [r, c] notation for specific index (row-column index)
- a[0, 1:-1:2] — start index : end index : step size
- a[:2] = [1,2] — for second row; for first row
- b = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
- Replace b[:,1,:] = [[9,9],[8,8]]

Initializing different types of arrays:

(i) np.zeros(5) // vector of length 5 with float 0. # All zeros.
np.zeros((2,3,3)) // 3D.

(ii) np.ones((4,2,2), dtype='int32'). # All ones.

(iii) np.full((2,2), 99) # Other value — datatype = 'float'

(iv) full_like method
np.full(a.shape, 4) — like a // [coloring]
or
np.full_like(a, 4) → fills with 4.

(v) Making of random number
np.random.rand(4,2). — random numbers between 0 and 1.
or np.random.random_sample(a.shape)

(vi) np.random.randint(7, size=(3,3))
or
np.random.randint(4, 7, size(3,3)) → exclusive.

Identity matrix (vii) np.identity(5).

Repeating array (viii)
arr = np.array([[1,2,3]])
r1 = np.repeat(arr, 3, axis=0)
print(r)

1 → horizontal
0 → vertical.

Note arr = np.array([1,2,3])
r = np.repeat(arr, 3, axis=1) : 111222333

Mathematics (i) a+2 → would add, minus, multiply, divide each element by number

a = np.array([1,2,3]) b = np.array([1,0,1])

a+b → if same size adds elements index wise

(ii) taking sin of each values. np.sin(a) } element wise computation.

Linear Algebra a = np.full((2,3),2) or a = np.ones((2,3)) | b = np.full((3,2),2) print(b)

(i) ~~a+b~~ np.matmul(a,b)

determinant (ii) np.linalg.det(a) (linalgebra)

Statistics (i) np.min(stats). // whole min value

np.min(stats, axis=1) // on horizontal axis bars

np.min(stats, axis=0) // on basis of vertical axis

(ii) before = np.array([[1,2,3,4],[5,6,7,8]]) → row to column mapped.

print(before)

after = before.reshape((2,2,2))

print(after)

Vertically stacking (iii) v1 = np.array([1,2,3,4])

v2 = np.array([5,6,7,8])

np.vstack([v1,v2]) // same horizontal stacking can be done np.hstack.

File load np.genfromtxt('data.txt', delimiter=',')

filedata = filedata.astype('int32') → integer 32 bit format.

makes a copy.

Advanced Indexing & Boolean Masking

- filedata > 50 → returns a matrix of true or false.
- filedata[filedata > 50] → gives a list of all values greater than 50.
- a[[1,2,8]] // returns a list with values of a at index 1,2,8

copying.

Multithreading

(1) Breaking one task into multiple process, and again joining those process using threads

Most of the threads are used on different cores.

(i)
```
class Hello:
    def run(self):
        for i in range(5):
            print("Hello")
```

```
class Hi:
    def run(self):
        for i in range(5):
            print("Hi")
```

```
t1 = Hello()  // Hello object
t1.run()      ---- run let if called first
```

```
t2 = Hi()  // Hi object
t2.run()
```
↳ run second if called second.

Execute them parallelly.

- This execution is done by main thread
- from time import sleep
- from threading import *

```
class Hello(Thread):
    def run(self):
        for i in range(300):
            print("Hello")
            sleep(1)
```
gives 1 second.

```
class Hi(Thread):
    def run(self):
        for i in range(300):
            print("Hi")
            sleep(1)
```

```
t1 = Hello()
t2 = Hi()
t1.start()
t2.start()
```
sleep(0.2) — so and they dont go into collision.

print("Bye") — in between.

if you want this to wait at end

```
t1.join()
t2.join()
print("Bye")
```
after end of other threads execute.

Copying numpy array

```
a = np.array([1,2,3])
b = a
b[0] = 100
print(a)
```
pointer.

genuine

```
a = np.array([1,2,3])
b = a.copy
b[0] = 100
```

- np.all(filedata>50, axis=0) = takes vertical set of numbers from the array.
  all filedata>50.
- (~((filedata>50) & (filedata<100)))
  = not.

Web Scraping

Uses → (i) E-Commerce portal
(ii) Market Research
(iii) Social Media

Steps → DOCUMENT LOAD
↳ PARSING
↳ EXTRACTION
↳ TRANSFORMATION.

Packages : Pattern, Scrapy, Mechanize, Beautiful Soup, requests. (popular)

Libraries →

```
from bs4 import BeautifulSoup as soup
from urllib.request import urlopen as uReq.
url = 'https://www.flipkart.com'
uclient = uReq(my_url)
page_html = uClient.read()
uClient.close()
page_soup = soup(page_html, "html.parser")
containers = page_soup.findAll("div", {"class": "col-2-grid"})

print(soup.prettify(containers[0]))
```

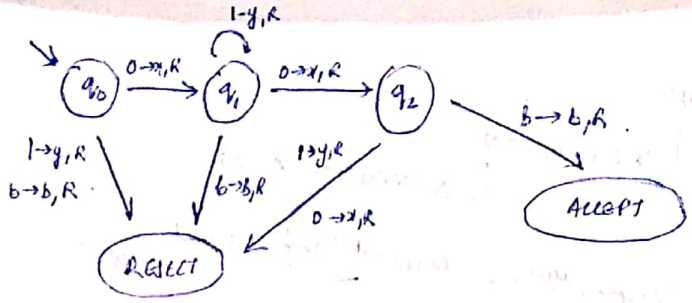
NLP

TOC Assignment

~ Aman Jha (2K18/IT/018)

Ans 1. $L = 01^*0$

turing machine will be $(Q, \Sigma, \Gamma, \delta, q_0, b, F)$



$Q \rightarrow \{q_0, q_1, q_2, REJECT, ACCEPT\}$

$\Sigma \rightarrow \{0, 1\}$

$\Gamma \rightarrow \{x, y, b\}$

$q_0 \rightarrow q_0$

$b \rightarrow b$

$F \rightarrow \{REJECT, ACCEPT\}$

δ : $\delta(q_0, 0) \rightarrow (q_1, x, R)$; $\delta(q_0, 1) \rightarrow (REJECT, y, R)$;

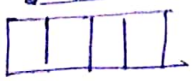
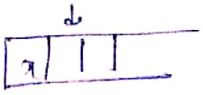
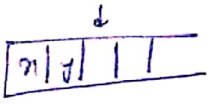
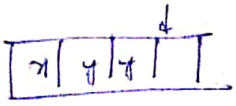
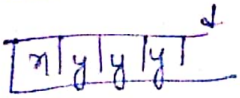
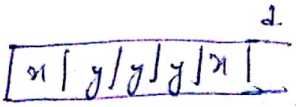
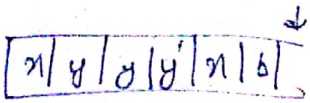
$\delta(q_0, b) \rightarrow (REJECT, b, R)$; $\delta(q_1, 1) \rightarrow \delta(q_1, y, R)$;

$\delta(q_1, 0) \rightarrow (q_2, x, R)$; $\delta(q_1, b) \rightarrow (REJECT, b, R)$;

$\delta(q_2, 0) \rightarrow (REJECT, x, R)$; $\delta(q_2, 1) \rightarrow (REJECT, y, R)$;

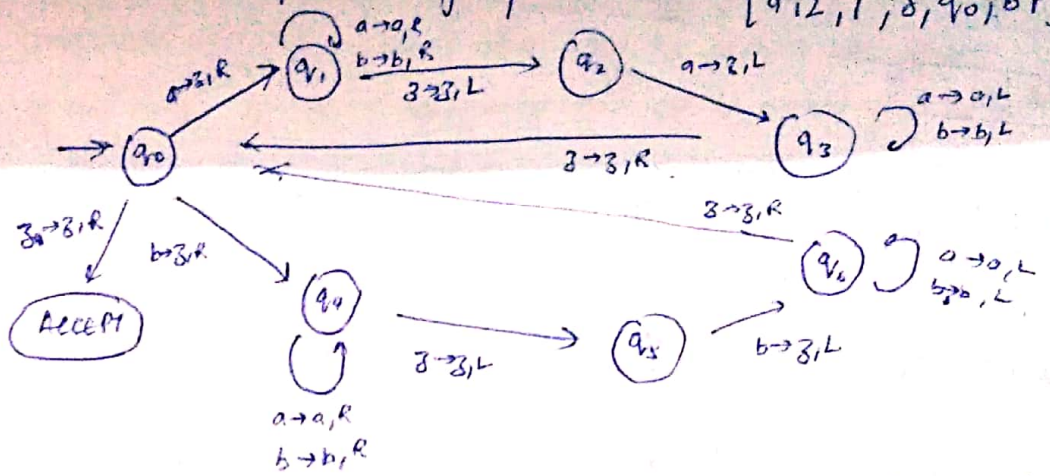
$\delta(q_2, b) \rightarrow (ACCEPT, b, R)$.

For example ⇒ 01110 should be accept. It can be written as 01110bb.

Tape	Current State	Input
	q_0	01110bb
	q_1	01110bb
	q_1	01110bb
	q_1	01110bb
	q_1	01110bb
	q_2	01110bb
	ACCEPT	ACCEPTED

Hence, string is accepted.

Ans 2: Turing Machine for even length palindrome is : $\{Q, \Sigma, \Gamma, \delta, q_0, b, F\}$



All states except q_0 go to REJECT state on the unmentioned input (Unmentioned edge δ)

$Q \rightarrow \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$

$\Sigma \rightarrow \{a, b\}$

$\Gamma \rightarrow \{a, b\}$

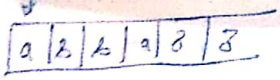
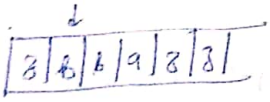
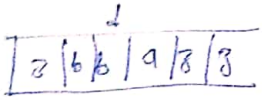
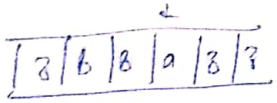
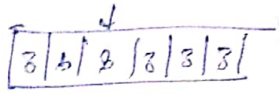
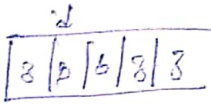
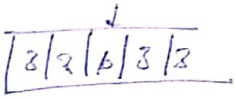
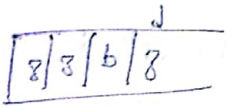
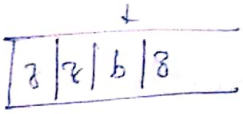
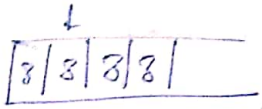
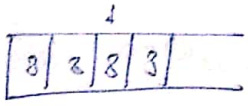
$q_0 \rightarrow q_0$

$b \rightarrow B$ (BLANK)

$F \rightarrow \{$ ACCEPT, REJECT $\}$

$\delta \rightarrow \delta(q_0, a) \rightarrow (q_1, B, R)$; $\delta(q_0, b) \rightarrow (q_4, B, R)$;
$\delta(q_0, B) \rightarrow$ (ACCEPT, B, R) ; $\delta(q_1, a) \rightarrow (q_1, a, R)$;
$\delta(q_1, b) \rightarrow (q_1, b, R)$; $\delta(q_1, B) \rightarrow (q_2, B, L)$;
$\delta(q_2, a) \rightarrow (q_3, B, L)$; $\delta(q_2, b) \rightarrow$ (REJECT, b, R) ;
$\delta(q_2, B) \rightarrow$ (REJECT, B, R) ; $\delta(q_3, a) \rightarrow (q_3, a, L)$;
$\delta(q_3, b) \rightarrow (q_3, b, L)$; $\delta(q_3, B) \rightarrow (q_0, B, R)$;
$\delta(q_4, a) \rightarrow (q_4, a, R)$; $\delta(q_4, b) \rightarrow (q_4, b, R)$;
$\delta(q_4, B) \rightarrow (q_5, B, L)$; $\delta(q_5, a) \rightarrow$ (REJECT, a, R) ;
$\delta(q_5, b) \rightarrow (q_6, B, L)$; $\delta(q_5, B) \rightarrow$ (REJECT, B, R) ;
$\delta(q_6, a) \rightarrow (q_6, a, L)$; $\delta(q_6, b) \rightarrow (q_6, b, L)$;
$\delta(q_6, B) \rightarrow (q_0, B, R)$

for example : abba BB

Tape	Current State	Input
	q_0	abba$
	q_1	abba$
	q_1	abba$
	q_1	abba$
	:	
	q_3	abba$
	q_0	
	q_4	
	q_4	
	q_5	
	q_6	
	q_0	
	ACCEPT	Hence Accepted.

Q8. Variants of Turing Machine:

1) Multi-tape turing machine: It has multiple tapes and each is accessed via a separate head and each head is independent of others. At first, the first tape is occupied by input and other tapes are kept blank. Then, the machine reads consecutive symbols under its head and the machine prints a symbol on each tape and its head is moved.

2) Multi-Track turing machine: This is a special type of multi-tape turing machine which has multiple track but just one tape head. Here a single tape head reads n symbols from n tracks at each step.

3) Non-Deterministic Turing Machine: To this type of Turing Machine, for every state and symbol, there are a group of actions that the machine can perform. This computation of this machine is a tree of configuration that can be reached from start of the configuration. Input is accepted if one leads to accept configuration.

Ans: $A = \{100, 0, 1\}$

$B = \{1, 100, 00\}$

This can be represented as $\boxed{\frac{100}{1}}$, $\boxed{\frac{0}{100}}$, $\boxed{\frac{1}{00}}$

(X) (A) (B)

① Choosing X, we get $\frac{100}{1}$

② Choosing Z, we get $\frac{1001}{100}$

③ Choosing X, we get $\frac{1001100}{1001}$

④ Choosing X, we get $\frac{1001100100}{10011}$

⑤ Choosing Y, we get $\frac{100110010010}{10011001000}$

⑥ Choosing Y, we get $\frac{1001100100100}{10011001000100}$

⑦ Choosing Z, we get $\frac{100110010010001}{1001100}$

Hence PCP corresponding problem is solved.

Ans 5 Undecidability of Universal Language:

Let L_u, universal language be a recursively enumerable language as well as let it be recursive.

Then by definition, L_u' complement of L_u, is also recursive language.

This implies a turing machine M' exists for L_u'.

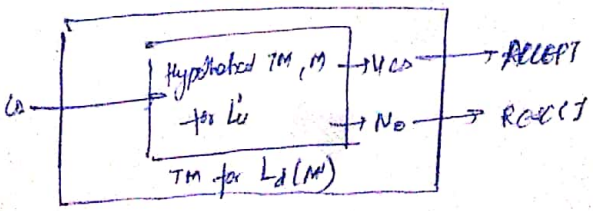
This implies a turing machine M' exists for L_2, diagonalization language for L_u and L_u'

But by definition $L_2 = \{w_i | w_i \text{ is not in } L(M_i)\}$ where M_i is any turing machine.

This means L_d not recursively enumerable.

Hence no assumption had L_u was recursive is wrong.

Hence L_u is recursively enumerable but not recursive.



COMPUTER VISION

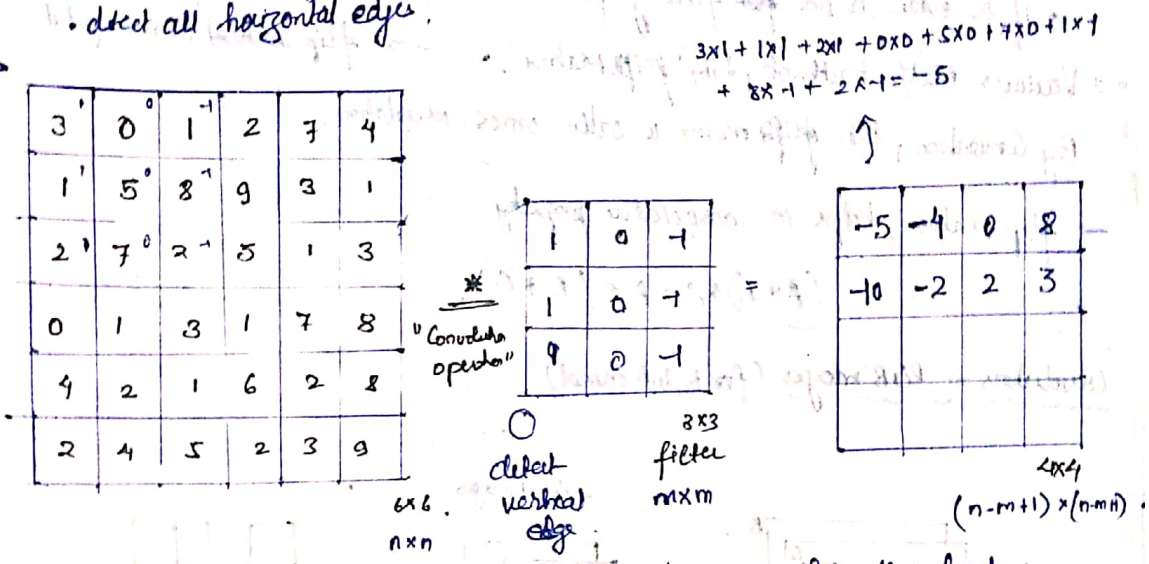
Convolutional operation - Basics of Neural Networks

CONVOLUTIONAL NEURAL NETWORKS

Edge Detection Example

→ Problem
- detect all vertical edge
- detect all horizontal edges.

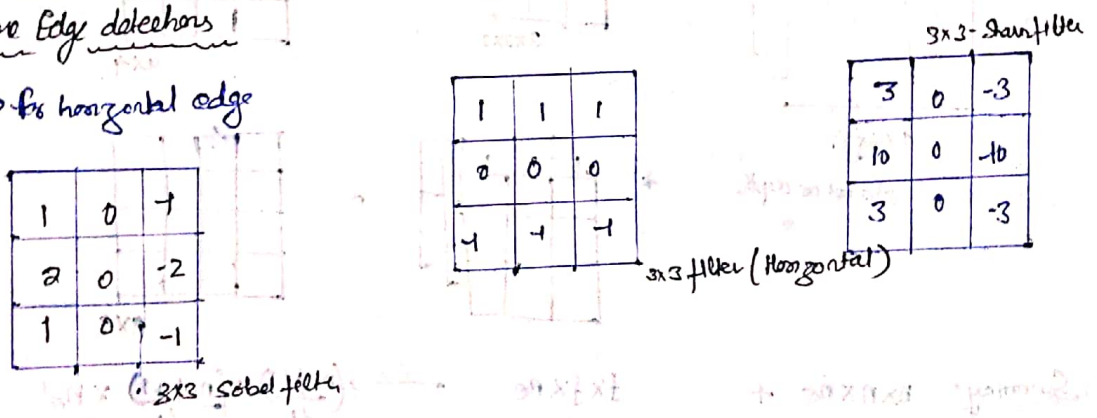
→



python : conv_forward and other deep-learning frameworks have other functions.

More Edge detections

→ for horizontal edge



Padding

→ padding the image, with an additional border, to prevent
- Shrinking output
- To prevent throwing away information from edge

→ p : padding amount, Convolved Eg matrix converts to $(n+2p-f+1)(n+2p-f+1)$

→ Valid Convolution and Same Convolution

↓ no padding

$n \times n * f \times f \rightarrow (n-f+1)(n-f+1)$

Pad so that output size is the same as the input size : or $n+2p-f+1 = n$

ie $2p = f-1$ or $p = \frac{f-1}{2}$

→ padding can be symmetric or asymmetric.

Strided Convolution

→ Rather than taking one step, we take s (no of stride) steps.

$$\text{output} : \left(\frac{n+2p-f}{s}+1\right)^2$$

• if the value is not a floating point number, then we use the floor function.

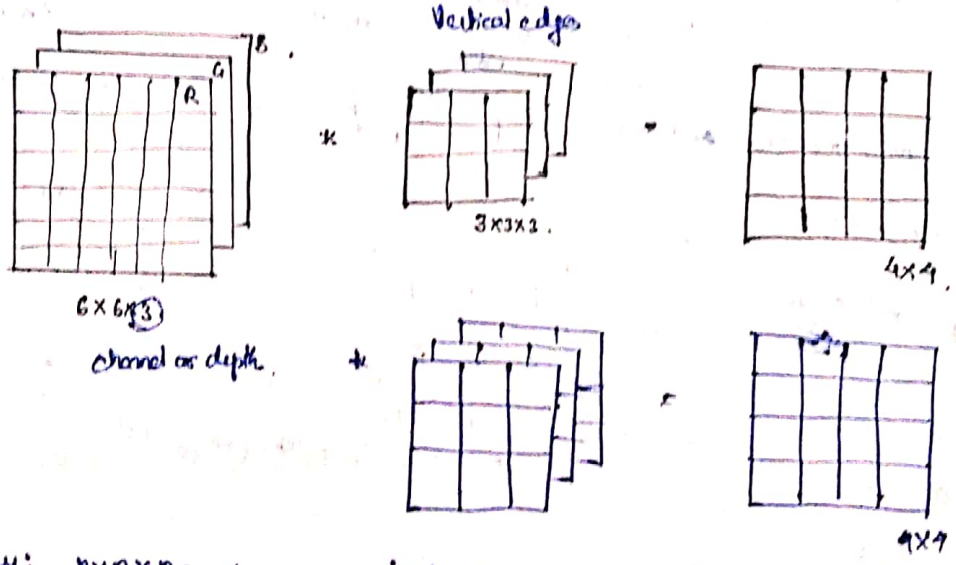
→ Various Math textbook, take 'flip matrix'. ⟶ flip vertical then horizontal

By convention, this flip matrix is called cross-correlation.

→ flip matrix, helps in associative property

$$(A * B) * C = A * (B * C).$$

Convolutions on RGB images (for R, G, B channel)



Summary: $n \times n \times n_c$ + $f \times f \times n_c$ ⟶ $\left(\frac{n-f+1}{1}\right) \times \left(\frac{n-f+1}{1}\right) \times n_c'$

$6 \times 6 \times 3$ $\quad$ $3 \times 3 \times 3$ $\quad$ ↓ # filters

One Layer of a Convolutional Network

for one layer,

$$\underset{\text{non-linearity}}{\text{Relu}}\left(\underbrace{\boxed{\#}}_{4\times 4} + \underbrace{b_1}_{\text{bias}}\right)$$

Computer Vision

→ Randomness → Stationary → strict sense stationary
→ wide sense stationary.
→ Non stationary.

• A stochastic of random process - is, a change in the state of some system over time - whose course depends on chance and for which the probability of a particular course is not defined..

(i) $Y(t) = Y(t-1) + \varepsilon(t)$ (Pure Random walk process) → variance evolved on time, and goes to infinity

(ii) $Y(t) = \alpha + Y(t-1) + \varepsilon t$ (Random walk & Drift process)
white term noise
↓ does not react to a long-run mean and has variance dep. on time

(iii) $Y(t) = \alpha + \beta(t) + \varepsilon t$ (Deterministic trend)
drift
deterministic trend
mean is constant.

(iv) $Y(t) = \alpha + Y(t-1) + \beta(t) + \varepsilon t$ (Drift + Deterministic)

(i) Stationary process, $Y(t) = \alpha + \varepsilon t$.

• ε_t : Stochastic component or white noise, means ε_t is independent and identically distributed with mean "0" and variance "σ^2"

$$\varepsilon_t \sim N(0, \sigma^2)$$

Probability density function

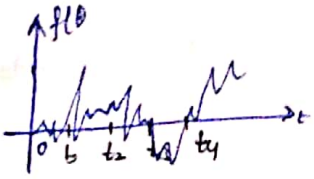
$f(x)$ = U.D.f → uniform density function, → $H = \frac{a+b}{2}$, $\sigma^2 = \frac{(b-a)^2}{12}$

G.D.f → Gaussian density function → H. σ^2.

R.D.f → Rayleigh Distribution function → $E(H) = \sigma\sqrt{\frac{\pi}{2}}$

Var(H) $= \sigma^2\left(\frac{4-\pi}{2}\right)$

Stationary



$m_1(0 < t < t_1) = m_1$

$m_2(t_1 < t < t_2) = m_2$

$m_{n-1}(t_{n-1} < t < t_n) = m_n$

$m_1 = m_2 = \ldots m_n$

(***) power operator

→ A statistical ensemble is a probability distribution for the state of the system.

If the ensemble size / data size of the non-stationary / stationary data is very large then the behaviour of the random signal is represented as gaussian.

(common n ≥ 25) Central limit theorem → k: no. of samples

$\bar{x}_1, \bar{x}_2, \bar{x}_3, \ldots \bar{x}_4 \ldots \bar{x}_k$ (sample means)

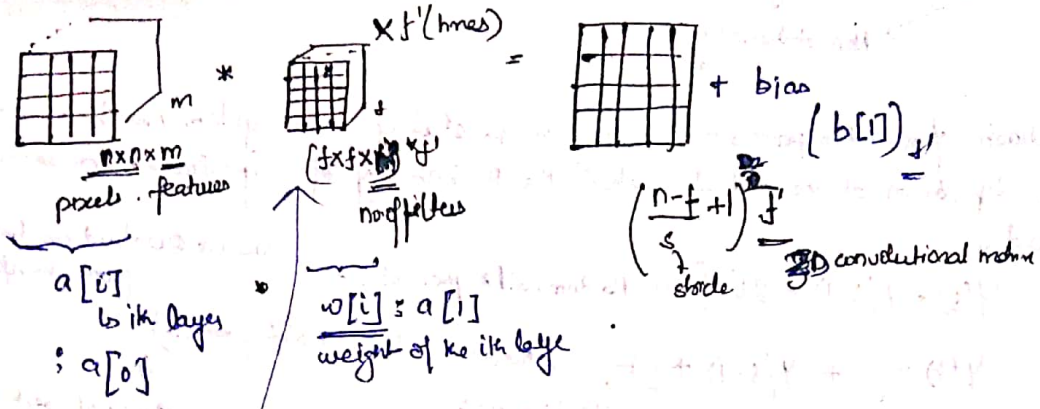
and $\bar{x}_{-\infty,\infty} = H.$, $\bar{\sigma}^2_{x(-\infty,\infty)} = \sigma^2$

for $k \to \infty$, $n \to \infty$, ...

approximation $N \sim \left(H, \frac{\sigma^2}{n}\right)$

plot graph for means, it will follow approximately normal distribution $N \sim \left(H, \frac{\sigma^2}{n}\right)$ where n: no. of data in a sample

— Continue : One layer of Convolutional Network.



$\therefore Z[i] = w[i]\,a[0] + b[i]$

$a[i] = g^{Z[i]}$ — non-linear

Notation

Each filter is: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]}$

- $f^{[l]}$ = filter size
- $p^{[l]}$ = padding
- $s^{[l]}$ = stride
- $n_c^{[l]}$ = number of filters

$a^{[l]} \rightarrow n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$ (activations)

Input : $n_H^{[l-1]} \times n_w^{[l-1]} \times n_c^{[l-1]}$ — increase height and width differ. — no. of channels.

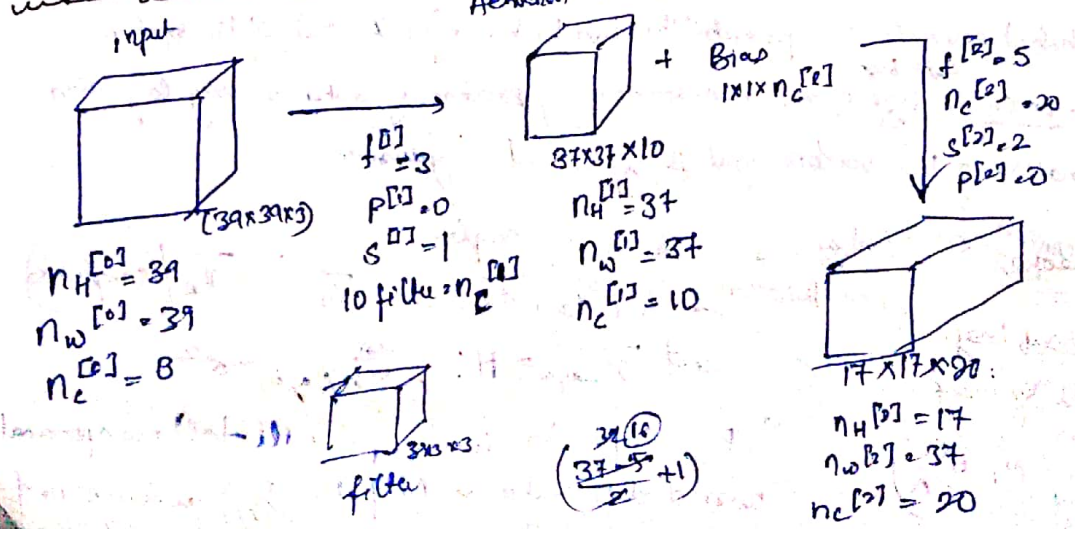
Output : $n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$

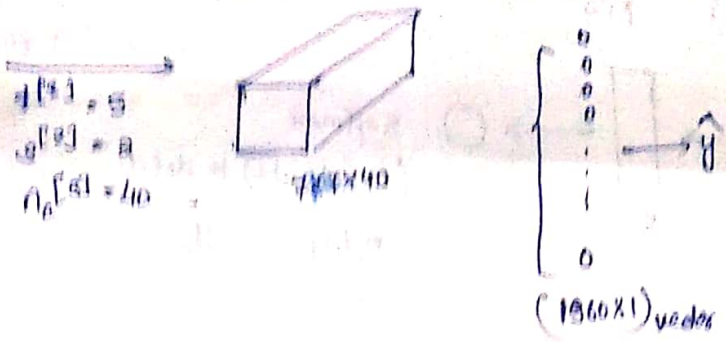
$$\rightarrow n_{H/w}^{[l]} = \left[\frac{n_{H/w}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1\right]$$

$\rightarrow n_c^{[l]} =$ no of filters $n_c^{[l]}$

- $A^{[l]} \rightarrow m \times n_H^{[l]} \times n_w^{[l]} \times n_c^{[l]}$
- Weights $\rightarrow f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$ — no of filters.
- Bias $\rightarrow 1 \times 1 \times n_c^{[l]}$ = only 3 axes

Simple Convolutional Network

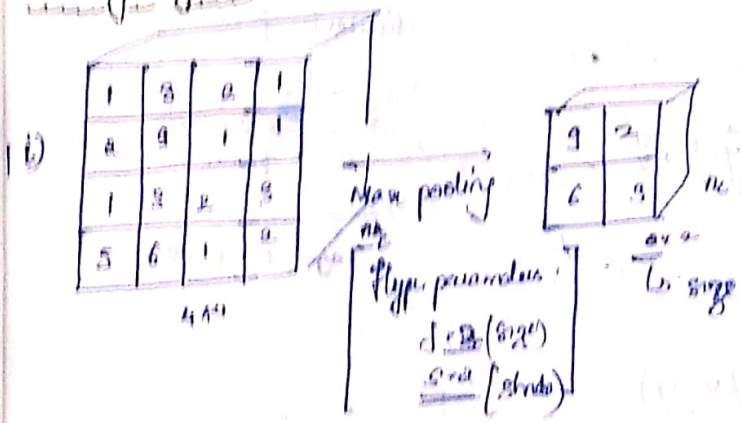




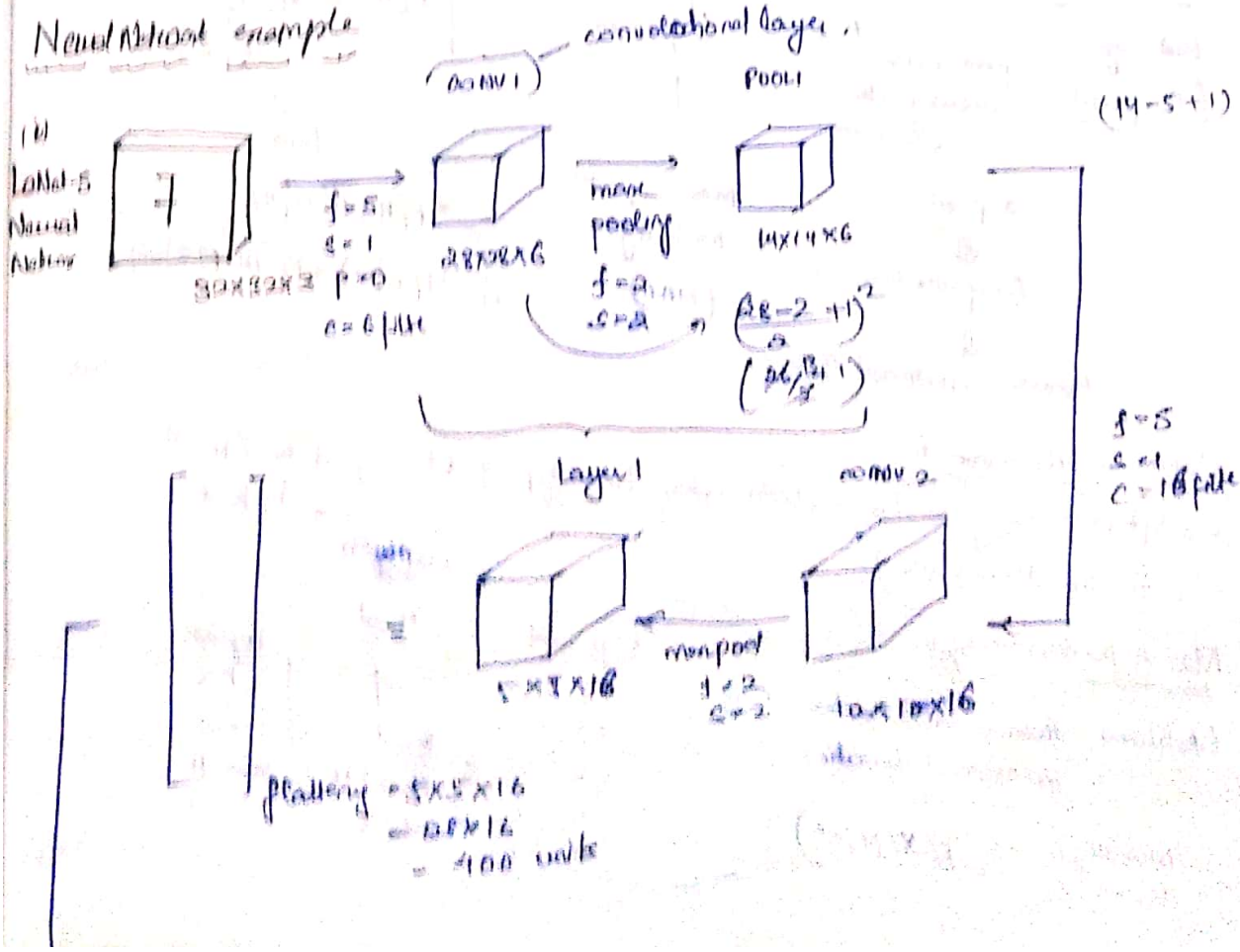
Types of layer in a Convolutional network:

i) Convolutional (conv)
ii) Pooling (pool)
iii) Fully connected (FC).

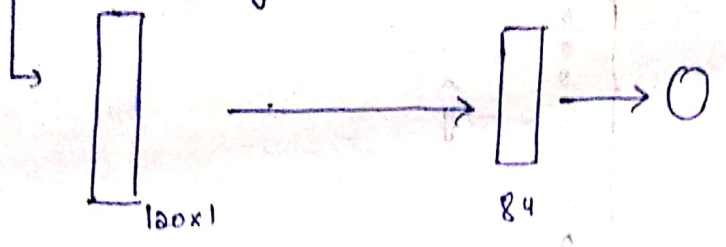
Pooling layers



Neural Network example



FC3 (fully connected layer) FC4



$W^{[3]} = (120, 400)$

Why Convolutions?

Advantages:

i) parameter sharing

ii) sparsity of connections.

(In each layer, each output value depends only on a small number of inputs)

Training set: (Cat detector).

→ Any set of networks, gives you a cost function

$$\text{Cost } J = \frac{1}{m}\sum_{i=1}^{m} L\left(\hat{y}^{(i)}, y^{(i)}\right)$$

→ fourier transform

rotating vector = 0.5 cycle/second.

or 1 cycle → 2 sec.

(good) → frequency.

1 cycle per frequency time.

Bayes theorem

X → Real Image

Y → Noisy Image

$$P(x) \cdot P(y|x) = P(y) \cdot P(x|y)$$

Real image (A-priori)

priori noise occurs in the given image

A-priori

a-posteriori

a-priori → Computer Vision → Maximum Likelihood Algo.

Max. a posteriori prob. Algo (MAP)

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E)}$$

Priori.

$$= \frac{P(H)\,P(E|H)}{P(H)\,P(E|H) + P(\neg H)\,P(E|\neg H)}$$

Maximum Likelihood Algo

→ optimum way to fit a distribution to a type of data (find Mean, σ^2 for the distribution)

Max A-posterior Algo

Likelihood → Assume data is derived using gaussian distribution.

Minimize $-\log P(X; \mu, \sigma^2)$ μ, σ^2

Distribution

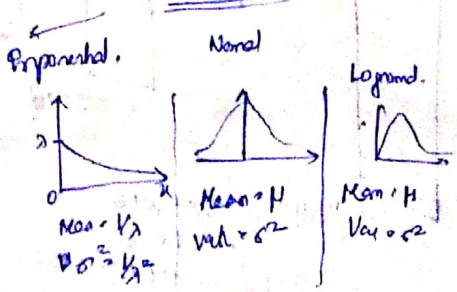


Image processing

Digital Image (space signal).

- two-dimensional signal $x[n_1, n_2]$, $n_1, n_2 \in \mathbb{Z}$
- Indices locate a point on a grid → pixel
- Grid is usually regularly spaced.
- Values $x[n_1, n_2]$ refer to the pixel's appearance

Grayscale Image vs. Color

▲ grayscale images : scalar pixel values
▲ Color images : multi-dimensional pixel values in a color space (RGB, HSV, YUV etc)
▲ We can consider the single components separately.

We can present using cartesian plot

Most common representation : Black and white.

★ Image could be unrolled (vectorized representation)

Decomposed likelihood

$$-\log p(X; \mu, \sigma^2) = \sum_{i=1}^{n} \frac{1}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}(x_i - \mu)^2 = \frac{n}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2$$

Assume IID.

also, $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$

minimizer for $\mu = \frac{1}{n}\sum_{i=0}^{\infty} x_i$

posterior probability

$$-\log p(w|X) = -\log p(X|w) - \log p(w) + c$$

(Maximum a posteriori estimator)

$$\min_w \; -\log p(X; \mu, \sigma^2) - \log p(w)$$

maximum a-posteriori

$$\min_w \; -\log p(w|X, y)$$

$$\Leftrightarrow \quad \min_w \; \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - h(x_i, w))^2 + \frac{1}{2\sigma^2}\|w\|^2 + \text{const}$$

$$\min_w \; \frac{1}{2n}\sum_{i=1}^{n}(y_i - h(x_i, w))^2 + \frac{\lambda}{2}\|w\|^2$$

— Gaussian Noise.

gaussian noise is indepent and identically distributed

WGN

Maximum Likelihood Algo

$x(t) = s(t) + w(t)$ [time Series]

white and gaussian.

A time series with

- Mean = 0
- ST. Dev is constant with time
- Correlation Between days is 0.