Lec: 18
04-10-24

"unsupervised Learning"

Goal of su learning → Prediction

unsupervised → finding the groups / patterns / similarities in the sample

2 methods:

i) Principal components analysis

ii) clustering

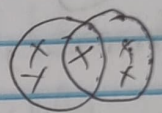
clustering:

finding subgroups / clusters

i) k-means clustering

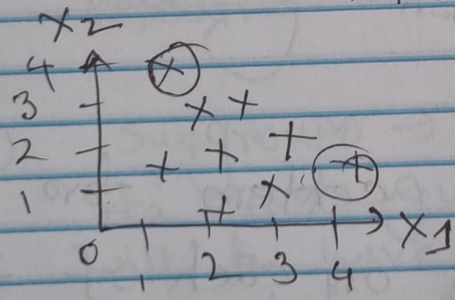
$C_1 \cup C_2$ = union →

intersection → $C_1 \cap C_2 = 1$



Shouldn't be any intersection

clusters should be distinct samples



WCV

↓

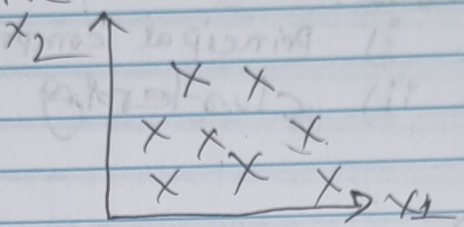
within cluster variation

Euclidean distance:

"from no clusters to
↓
samples with clusters"

k-means clustering algorithm:

i) Data received

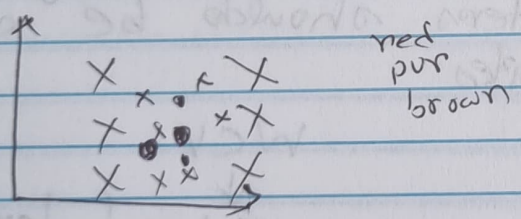


k= nu of clusters that we want to get

assume, we want to get 3 clusters

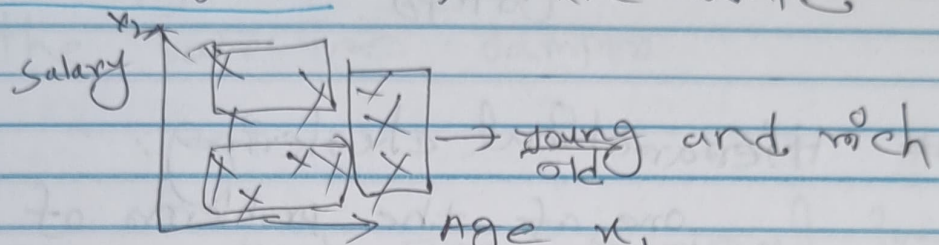
k = 3

1st step: ~~Assigning~~ randomly samples



finding the centers for ~~both~~ 3 clusters on each of the clusters

Reassigning each sample → based on the nearest distance to the sample.



Then finding the new centroid based on the new clusters

2 problems; / Disadvantages:

i) randomly selecting samples 1st step usually doesn't give the consistent results

↓

So need to run multiple times and finding out which one is repeating

ii) Subgroups → we specified 3/5 but it's not guaranteed that data will have this no of subgroups → model is doing good but underfitting

↑↑ → k ↓ more than ramp → overfitting underfitting

steps summary
→ no of clusters
→ selecting k objects randomly
→ then each observation to their closest centroid

overcoming: by doing cv in multiple steps.

Hierarchical clustering:

one of the problems of in k-means can be solved.

finding all the no of clusters within the data

↓

bottom up approach

↓

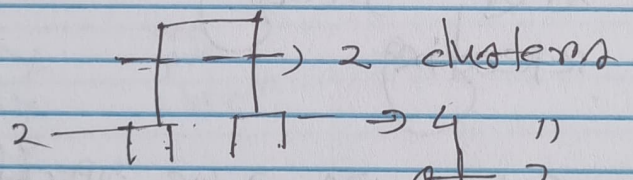
individual sample

Dendogram → will guide us

↓

to see how the clusters look like like.

cutting



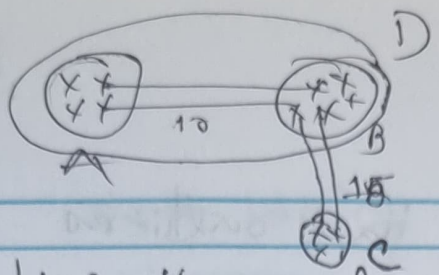
** Types of linkage:

linkage method

i) complete linkage method:

clusters are closers → similars

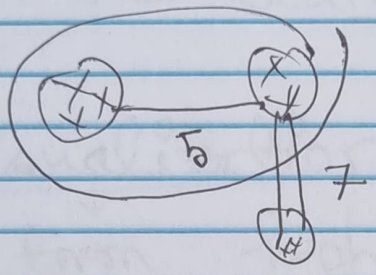
find the maximal intercluster dissimilarity



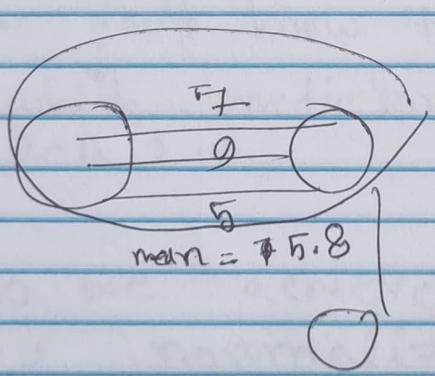
→ now 2 clusters
↓
D and C

finding the largest distance between the pairwise samples

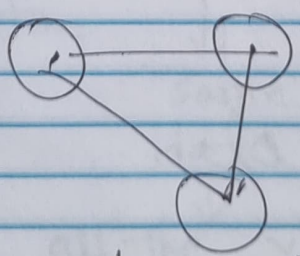
ii) single : finds the minimal interclass dissimilarities



iii) Avg : mean inter-cluster dissimilarity



iv) centroid : also sensitive to the outlier



complete and single → are sensitive to outliers

If data has outliers

↓

then centroid and avg good

scaling metrics: mean / normalized

Lec: 19
04-17-24

Deep-learning

PCA:

Reducing the dimension for example from 6000 to only 2.

Principle component analysis: Advantage:

i) for reducing the dimension correlated features are putting together

ii) visualization

from high dimension to 2-dimension

Disadvantage:

i) we don't know the details which predictors are making PCA 1 →

How do we achieve PCA:

step 1: normalize / standardizing the data

same scale of the data (not vast difference)

all the values will fall between 0 and 1.

Standardization

[0, 1]

Min Max

↓

0

SD → 1

upper/ lower value can vary

↓

Bringing the values around the mean of the numerical values

↓

If the data's are sensitive to min or max, then we need to normalize.

Step 2: calculating the co-variance matrix

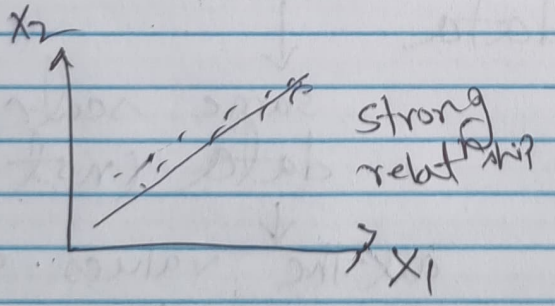
↓

how different the predictors are relating to each other etc

	X_1	X_2	X_3
X_1	1	0.9	0.1
X_2			
X_3			

→ we wanna measure how strongly X_1 measures X_1

3X3 PXP matrix



Data : $X_{n \times p}$

normalizing

finding co-variance

$cov_{p \times p}$

Step 3 :

Taking cov matrix

↓

Calculating Eigenvectors and Eigenvalues from covariance matrix

Eigenvectors

↓

Represent the principled component

Shape : P X P

X_1 X_2 X_3

	PC_1	PC_2	PC_3
PC_1			
PC_2			
PC_3			

Eigenvalues

↓

Represent the variance explained by the principled components

1 X P

PC_1	PC_2	PC_3

PC_1 captured information from PC_1

↓

If, $PC_1 > PC_2$

↓

then PC_1 will contain the more inform

from the data

Step 4: Project the principal component onto the normalized or standardized data

eigenvectors $\odot$ scaled data

$P \times P \quad n \times P^T$

matrix transpose:

$$A = \begin{bmatrix} 2 & 1 \\ 2 & 4 \\ 3 & 1 \end{bmatrix}$$

$A^T = 2 \times 3 \times 2$

2×3

$$\begin{bmatrix} 3 & 2 & 2 \\ 1 & 4 & 1 \end{bmatrix}$$

$P \times P \odot P \times n = P \times n$

Scaled data — eigenvectors

$n \times P \quad P \times P$

$= n \times P$

no need to matrix transpose as internal P·P are same

Summary:

→ normalizing (only numerical features not categorical")

→ covariance matrix → eigenvectors → PC.

eigenvalues → amount of info that each PC carries

→ Projecting PC onto the standardized data

Then selecting the principal component, M, where $M < P$

$\Rightarrow n \times M$

selecting M based on eigenvalues.

Select the principle component, M,
where $M < P$

$\Rightarrow n \times M$

selecting M based on eigen values.

youtube:

Eigen vectors and eigen values:

(i) covariance matrix between features;

(ii) Eigen vectors and eigen values will be found out from this covariance matrix

(iii) Eigen vector → eigen value →

magnitude of the eigen vector ↓ capture the max variance

Linear transformation of matrix

↓

$Av = \lambda v$

Steps to calculate eigen value and vectors:

① covariance of features:

$[x, y] \rightarrow X^1$

$$cov(X,Y) = \sum_{i=1}^{w} \frac{(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

→ standardizatⁿ

→ covariance matrix of x and y

→ find out eigen vectors and values

$$Av = \lambda v$$

$A \rightarrow [\lambda_1, \lambda_2]$ ⇒ eigen values

λ_1 → PC1, λ_2 → PC2

Gaussian distrt
outliers

standardizatⁿ → to transform the data to have a mean of zero and SD of 1

mean of 0 and SD - 1 or z-score normalizatⁿ

lower/upper any values

normalizatⁿ → to make feature values range from 0 to 1.

↓ always range from 0 to 1

↓ in case of neural network

Project ***

Bean classes are : 7
SEKER BAR BOM CALI HOR
DERMASON, SIRA

(I) Data Import + Descriptn

(II) " Preprocessing:
→ Data cleaning
→ Dropping the duplicated items
→ Null data
→ Data split + normalizatn
→ Data - visualizatn

Train → LR (penalty = 'l2', verbose = 1, n_jobs = -1)

Penalty 'l2' = L2 regularizatn
Ridge
"
this adds a penalty term to the loss funct^n
which penalizes the ↑↑ co-efficient values
so that overfitting can be minimized

verbose=1 ; Detailed output during the training process
('Iterat^n + convergence)

n_jobs = -1; the number of CPU cores
all available cores

these parameters help improve the efficiency and effectiveness of the LR model for classificat^{n}.

model.get_params().keys() → returning a list of all the parameter names

.solver → LR model → optimizat^{n} algorithm

lbfgs → Limited memory Broyden-Fletcher Goldfarb-Shanno
↓
optimizat^{n} algorithm

model_select^{n}. GridSearchCV;
Dimensionality Reduct^{n}
Linear_model.LR → Machine model classifier

null → mean + median of specific column

dup → backup

outli → proposed classifiers doesn't deviate from its predictn

normalizatn → to standardize the scale of features

data was divided to a training set, 80%
and a test set 20%

pearson correlatn matrix;
analysing the inter-relationships between the variables

Analysis: 16 fea
correlatn + Pearson plot

04-22-24

Deep Learning

Traditional / classical ML models:
*** SVM, Decision tree
KNN, Hierarchical
came from statistical

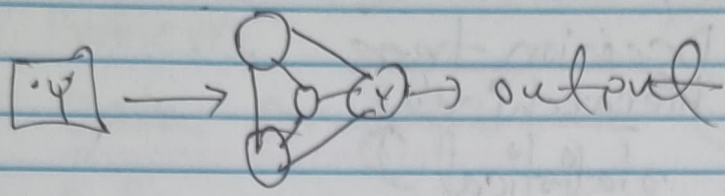
1990 → Deep learning:
at that time computational / hardware was scarce
↓
so classical deep-learning

GPU → can compute heavy matrices and run
↓
then based on this → deep learning models (large dataset)
↓
started/ experimenting

GANS → can generate images from text

single layer NN

→ let the computer learn from data



each neuron → responsible for particular task

Artificial neuron that will learn from the input

$$y = mx + b$$
$$= Wx + b$$

W → weights/parameters
x → input
b → bias

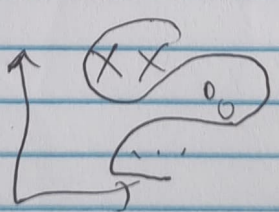
example:

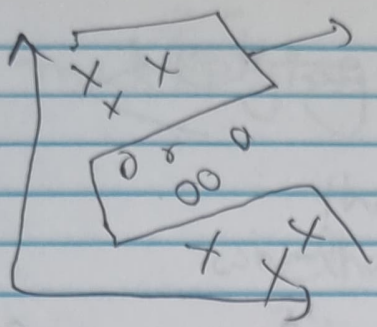
$$A_1 = w_1x_1 + w_2x_2 + \cdots w_4x_4 + b$$

those parameter values need to checked to the true values

* Adding nonlinear behavior from linear func.

Activation:



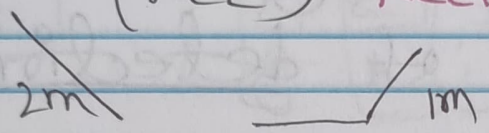


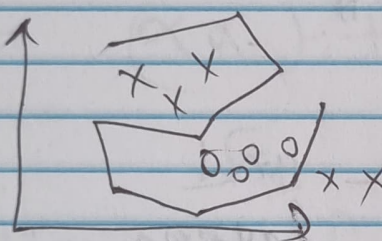
five different neuron each can go on one direction, can't bend.

↓

So it's difficult to capture non-linearity

Rectified Linear (ReLU) Rectified linear unit:





How works?

↓

Relu = max(A_i^0, 0)

if the line is not good → then (-)

↓

If not 0, then A_i^0

↓

good learn

↓

don't learn from this

Sigmoid:

↓

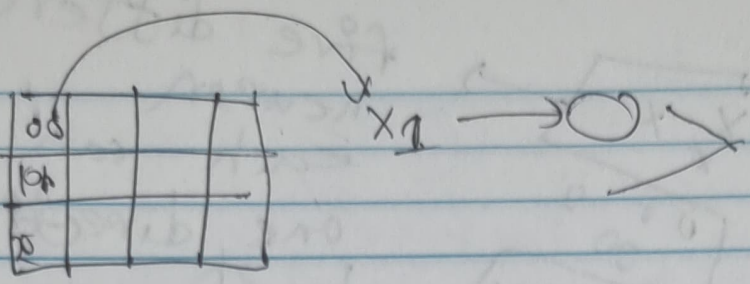
How it works?

S shaped ∫

↓

between 0 and

$$\int_0^{0.5} 1$$



convolutional Neural network

↓ CNN

for image classification

any kind of detection

↓

from CNN

2 main operations

→ convolution and

→ Pooling layers

convolution filters

image → matrices → elements → number

↓

pixels value

↓

Grey scale

↓

white ← 0 — 255

↓ ↓

closer to 0 darker → 1 pixels value

multiplying by special matrix
↓
that can identify
whether tiger / cat

convolut^n filter = $\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix}$

$$\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} \cdot \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} = \begin{bmatrix} F_1 & F_2 \\ F_3 & F_4 \end{bmatrix}$$

$$(\alpha a) + (b \times \beta) + (\gamma \times d) + (\delta \times e)$$

F_1

Hierarchy of filters

cat pic: || → vertical → lower pixel → white is more visible

Pooling

reducing the dimension from 4x4 to 2x2.

max pooling / avg pooling

convolve → pool → convolve → pool → flatten

↓

like basic neural network

↓

output

AI model → review

↓

differentiating whether + / −

featurization: Bag-of-Words

→ Dictionary of frequently used words → p predictors

	x_1 good	x_2 sad	x_3 happy
0	0	0	
1	4	0	
2	2	0	
⋮			
n			

n samples

→ Then learning from the sentences punctuation of the word in

a sentence (meaningful)

Recurrent neural Network Archi

↓

mothers of chatgpt