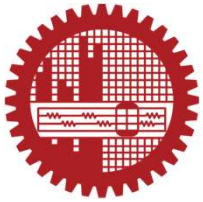
# CSE 471
# Machine Learning

## K Nearest Neighbor

Prepared by
Madhusudan Basak
Assistant Professor
CSE, BUET



# k-Nearest Neighbor

- Also known as kNN/KNN
- KNN can be used for both classification and regression predictive problems.
- Non-parametric approach: Don't assume any strong/fixed model or functional form

- For each test input point,
  - ➢considers the class/output of its nearest k number of train (available) data points and
  - ➢Determine its class by voting of the k data points
    - ✓May use different distance calculation measure (e.g., Euclidean, Manhattan)
    - ✓Voting system can be equal/weighted

# k-Nearest Neighbor

- Calculate Distance from ***point p*** to all the training points
- If k=1, nearest point = {A}

Predicted Class= Green

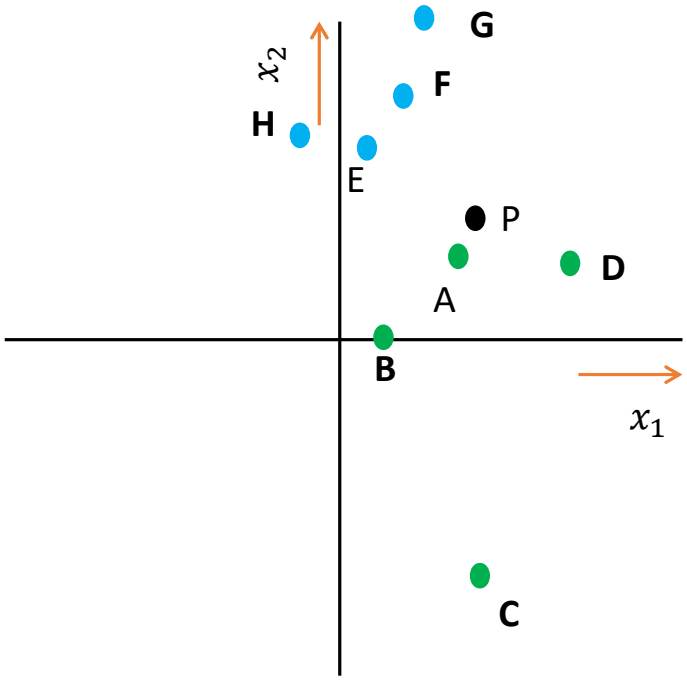
- If k=2, nearest points = {A, D}

Predicted Class= Green

- If k=3, nearest points = {A, D, E}

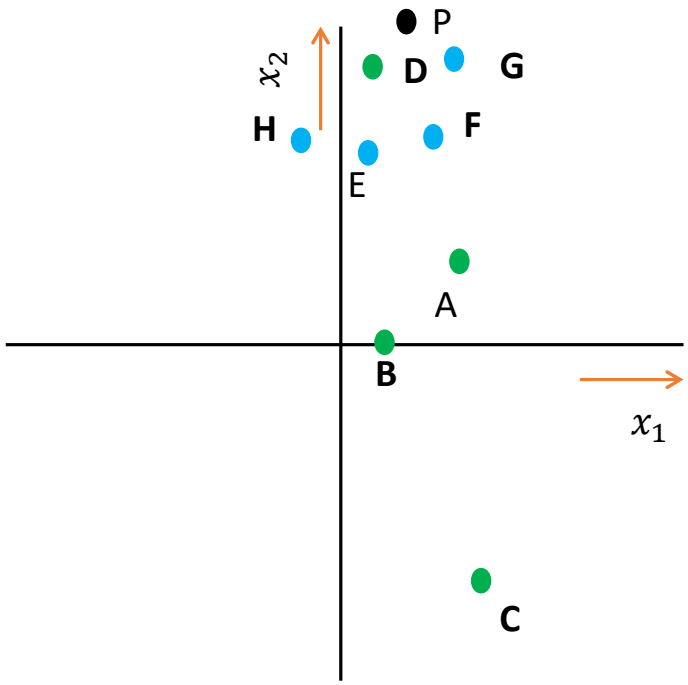
Predicted Class= Green

- What if k=7?



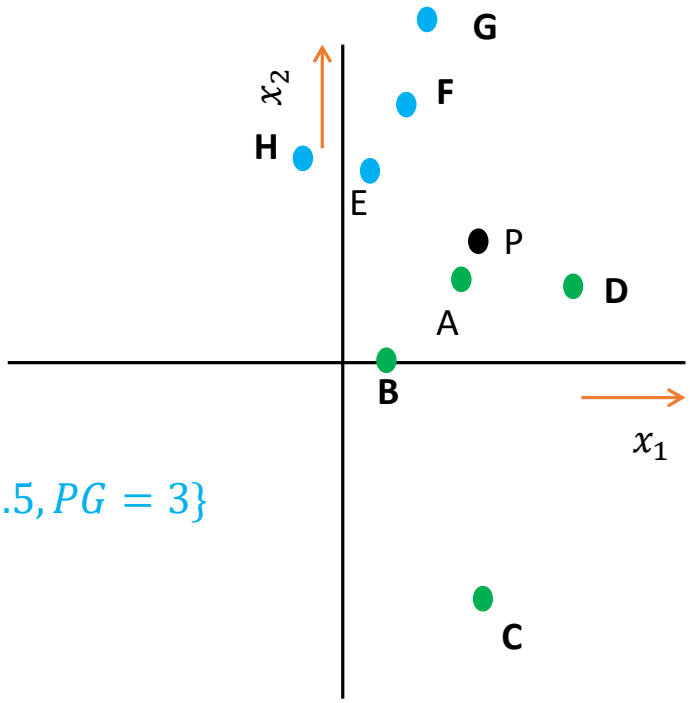
# k-Nearest Neighbor Issues

- Tuning the k (hyperparameter) value
  - ➢Too small k: sensitive to outliers (as indicated by the figure)
  - ➢Too large k: too many points from other classes
  (As the case in the previous slide)



# k-Nearest Neighbor Issues

- Weight of nearest neighbors
  - ➢Equal weight
  - ➢Different weights




- If k=7, Nearest points = {A, D, E, B, F, H, G}
- If equal weighted, predicted class = Blue
- If weighted voting and $weight = \frac{1}{Distance}$
- Distances ={$PA = 0.5, PD = 1.2, PE = 1.5, PB = 1.7, PF = 2, PH = 2.5, PG = 3$}
- Vote for Green =2*1+0.8*1+0.6*1 = 3.4
- Vote for Blue =0.7*1 + 0.5*1+0.4*1+0.3*1 = 1.9
- Predicted class = Green

# Document Classification

- One type of Text Classification
- kNN can be used here!
- kNN works fine on structured data but cannot handle unstructured data (Text).
- Solution?
  - Structured representation of Text (Unstructured Data)

# Structured Representation of Text

- Two Common Techniques
  - Binary Vector Representation
    - ✓ 1/0 bit is used for presence/absence of a word
  - Bag-of-Word (BoW) Representation
    - ✓ Word (hash id of the word) and frequency of the word is used here

# Binary Vector Representation of Text

- Example Train Documents

**Document 1 (Sports):** *I play cricket. I play football.*
**Document 2 (Music):** *Play this music.*
**Document 3 (Music):** *I like singing.*
**Document 4 (Biology):** *Cricket is a very small insect*

- Example Text Document

**Document t:** *I want to play music*

Vocabulary/
Feature Space

| | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_t$ |
|---|---|---|---|---|---|
| *play* | 1 | 1 | 0 | 0 | 1 |
| *cricket* | 1 | 0 | 0 | 1 | 0 |
| *football* | 1 | 0 | 0 | 0 | 0 |
| *music* | 0 | 1 | 0 | 0 | 1 |
| *like* | 0 | 0 | 1 | 0 | 0 |
| *singing* | 0 | 0 | 1 | 0 | 0 |
| *very* | 0 | 0 | 0 | 1 | 0 |
| *small* | 0 | 0 | 0 | 1 | 0 |
| *insect* | 0 | 0 | 0 | 1 | 0 |
| *want* | 0 | 0 | 0 | 0 | 1 |

- Create the full vocabulary
  - Comprising all words of all documents

- For each document, create a vector by putting 1 is the word presents in the document, otherwise put 0
- Vocabulary -> Fixed length -> Fixed input size (input feature space) -> Structured Representation of Text

# Binary Vector Representation of Text

- Example Train Documents

**Document 1 (Sports):** *I play cricket. I play football.*
**Document 2 (Music):** *Play this music.*
**Document 3 (Music):** *I like singing.*
**Document 4 (Biology):** *Cricket is a very small insect*

- Example Test Document

**Document t:** *I want to play music*

Vocabulary/
Feature Space

| | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_t$ |
|---|---|---|---|---|---|
| *play* | 1 | 1 | 0 | 0 | 1 |
| *cricket* | 1 | 0 | 0 | 1 | 0 |
| *football* | 1 | 0 | 0 | 0 | 0 |
| *music* | 0 | 1 | 0 | 0 | 1 |
| *like* | 0 | 0 | 1 | 0 | 0 |
| *singing* | 0 | 0 | 1 | 0 | 0 |
| *very* | 0 | 0 | 0 | 1 | 0 |
| *small* | 0 | 0 | 0 | 1 | 0 |
| *insect* | 0 | 0 | 0 | 1 | 0 |
| *want* | 0 | 0 | 0 | 0 | 1 |

- Calculate Hamming Distance from test document to all other train documents and select the k nearest neighbor

hd ($Dt$,$D1$)=(0+1+1+1+0+0+0+0+0+1) = 4

hd ($Dt$,$D2$) = (0+0+0+0+0+0+0+0+0+1) = 1
hd ($Dt$,$D3$) = (1+0+0+1+1+1+0+0+0+1) = 5
hd ($Dt$,$D4$) = (1+1+0+1+0+0+1+1+1+1) = 7

If k=1, Nearest Points/Documents ={Music}
Predicted Document Class/Type ={Music}

If k=3, Nearest Points/Documents = {Music, Sports, Music}
Predicted Document Class/Type ={Music}

# References

**Code Implementation Reference:**

https://machinelearningmastery.com/tutorial-to-implement-k-nearest-neighbors-in-python-from-scratch/

**Lecture References:**

https://machinelearningmastery.com/parametric-and-nonparametric-machine-learning-algorithms/

https://www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-clustering/

https://www.geeksforgeeks.org/weighted-k-nn/

Lecture Note by Madhusudan Basak