TÉCNICO
LISBOA

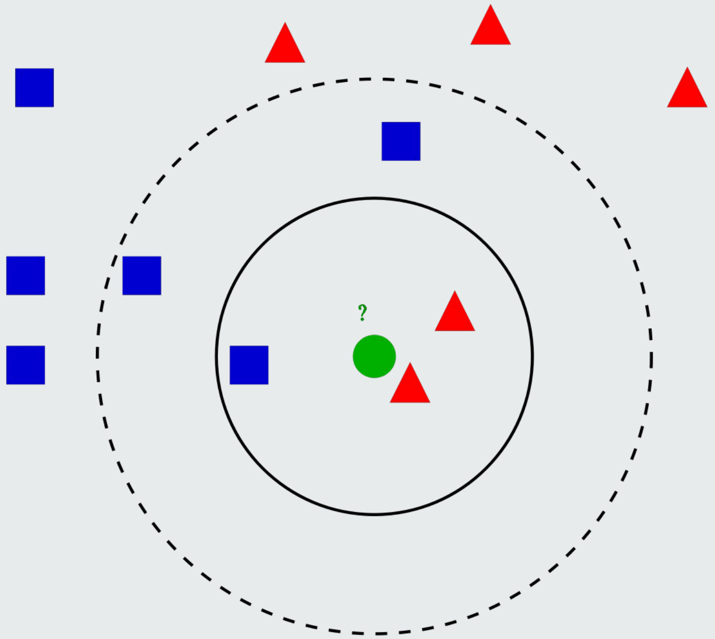
Computer Electronics

Lecture 10
The **iob-knn** core and
the **iob-soc-knn** repository

The KNN algorithm

Thomas M. Cover
KNN inventor

- Used for classifying data items
- Requires dataset of already classified data: the labeled dataset
- Will classify a set of data items: the test set



The KNN algorithm steps

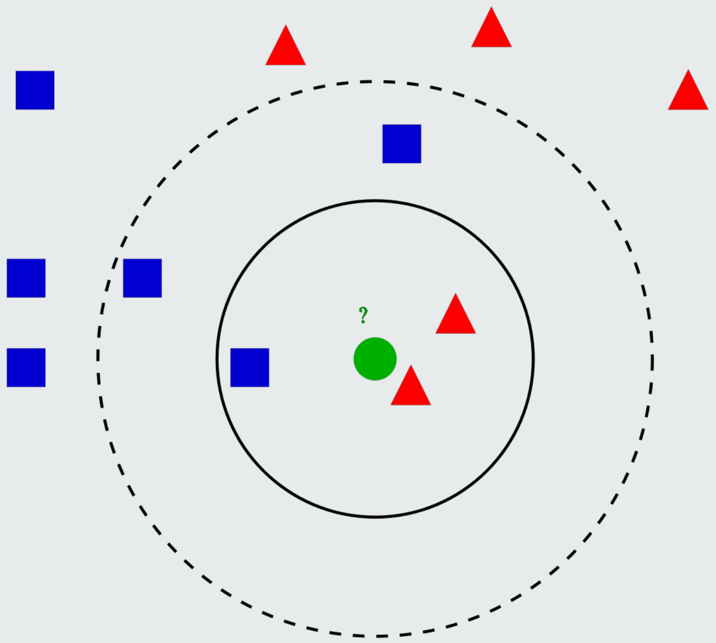
1)Set K, the number of neighbors

2)For each test item

3)Compute the “distance” to all dataset items

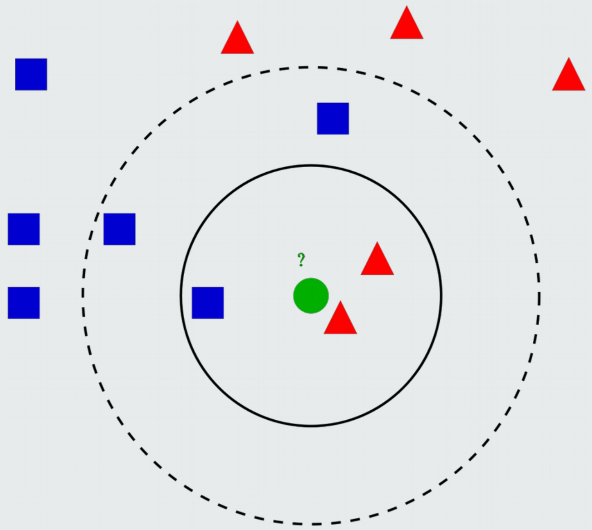
4)Keep the k-nearest neighbours

5)Set the label of the test item as the most common label among the k neighbors



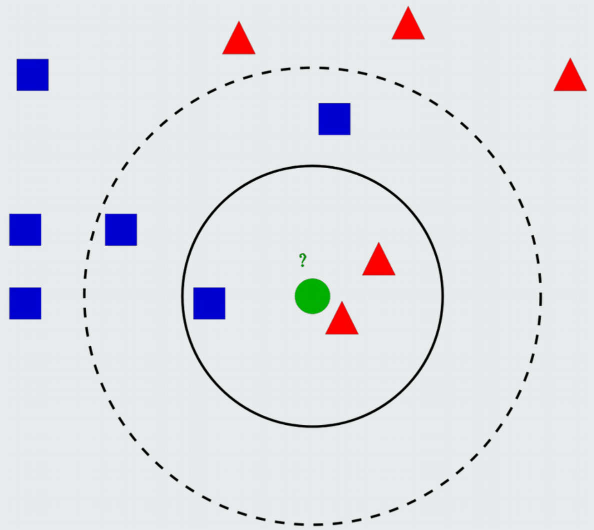
The IOb-KNN core

- Git repository: github.com/IObundle/**iob-knn**
- Let’s visit it

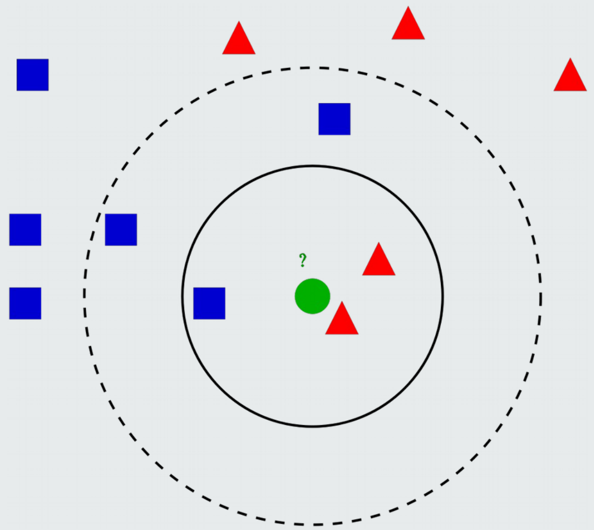


The IOb-KNN core software implementation

- Constants
  - S: random seed
  - N: data set size
  - K: number of neighbours (K)
  - C: number of data classes
  - M: samples to be classified (test set size)



The IOb-KNN core software implementation



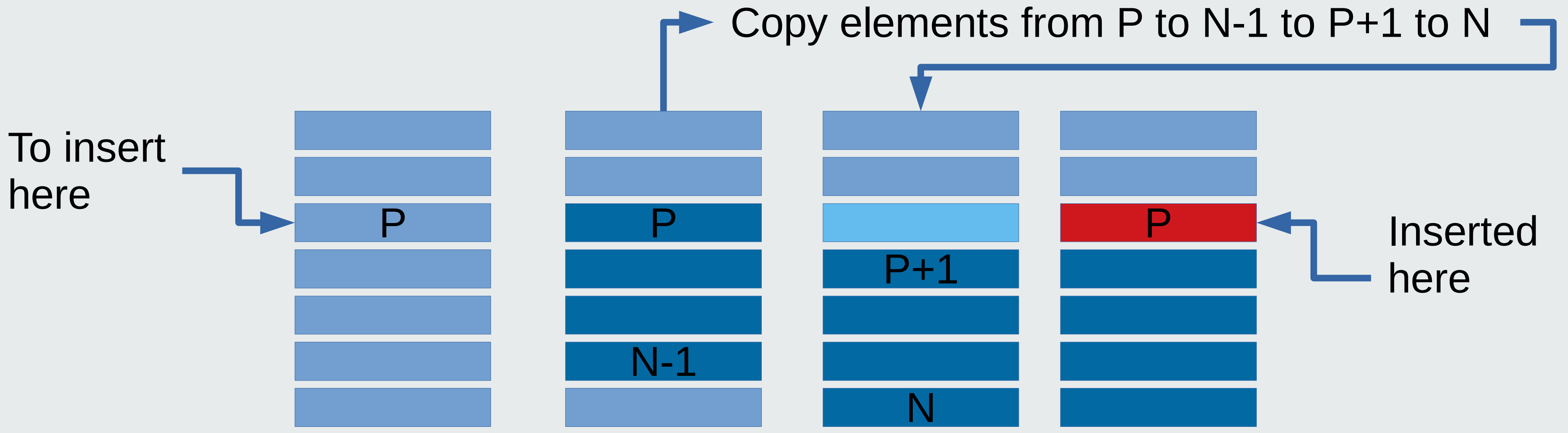
- Data structures
  - Datum:
    - X coordinate
    - Y coordinate
    - Label (unknown for test set)
    - Both dataset and testset are arrays of Datum elements
  - Neighbor:
    - Index in dataset array
    - Distance to test point
    - A k-neighbor array is formed for each test-point
  - Votes:
    - Array of C positions, one per class, to hold the votes in each class

The IOb-KNN core software implementation

$$d=\sqrt{(x_A-x_B)^2+(y_A-y_A)^2}$$

$$d^2=(x_A-x_B)^2+(y_A-y_A)^2$$

- Functions
  - Square distance:
    - Euclidean distance not needed
    - Square distance is enough for comparisons
  - Insert element in position P of an array



The IOb-KNN core software implementation (random numbers)

- Used to create the dataset and testset internally and avoid communication with host
- Used custom random number generator functions
- Does the stdlib srand and rand functions work? How complex it is? How long do they take to execute?
- To avoid these questions, we used our own
- Uses generate polynomial, Linear Feedback Shift Register (LFSR)
  - Algorithm not explained
  - Can be implemented in hardware
- 2 Functions
  - random_init(seed): similar to srand
  - cmwc_rand(): similar rand, returns next pseudo random number

The IOb-KNN core software implementation of the algorithm

1)Initialize dataset and testset randomly

x,y coordinates, dataset has label, testset has not

2)Process each test point in test set

The IOb-KNN core software: process each test point

1) Init list of neighbors
    - leave index blank
    - assign distance to Infinity
2) Find the k neighbors; for each point in dataset
    - compute distance to test point
    - find P, the position of the first farther neighbor in list
    - insert data point at position P in neighbor list
3) Compute the label using the k neighbors

The Iob-SoC-KNN system

- Git repository: github.com/IObundle/**iob-soc-knn**
- Let’s visit it