Artificial Intelligence: An Introduction

Wei Wang(王伟)

Engineering Research Centre of Applied Technology on Machine Translation and Artificial Intelligence , Macao Polytechnic University

weiwang@mpu.edu.mo; 匯智樓 (WUI CHI)-5/F, N56

Feb. 02, 2026

Outline

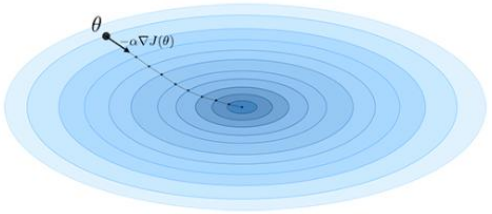
- Reflex-based Models
- Supervised Learning Classification

AI Models Types

AI models refer to different approaches or frameworks that are used to represent and solve problems in the field of AI.

These models provide a structured way to understand and analyze complex systems and make intelligent decisions.

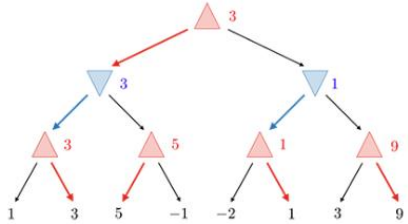
Reflex



- Feature vector, score, margin, residual
- Linear predictors, non-linear predictors
- Loss minimization, stochastic gradient descent, backpropagation, regularization

PDF Web

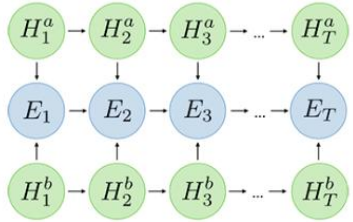
States



- Tree search, graph search, A star search
- Learning costs, relaxation
- Markov decision processes, game playing

PDF Web

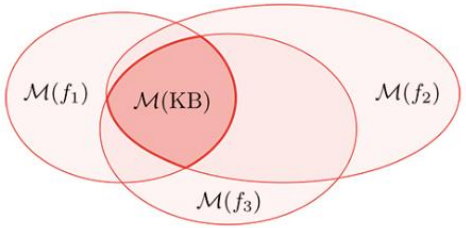
Variables



- Constraint satisfaction problem, backtracking search
- Beam search, Gibbs sampling
- Bayesian networks, probabilistic program, inference

PDF Web

Logic



- Notations, knowledge base, forward inference, rule properties
- Propositional logic, modus ponens, resolution rule, conjunctive normal form
- First-order logic, substitution, unificaion

PDF Web

Source: https://stanford.edu/~shervine/teaching/cs-221/

AI models define how an agent perceives, reasons, and acts.

Different models suit different environments and tasks.

AI Models Types

1. Logic-based models

- Symbolic representation of classes of objects.
- Deductive Reasoning.
- **Apps**: Question Answering Systems, Natural Language Understanding, Expert system
- **Options**: Propositional Logic , First-Order Logic, Knowledge Base.

2. States-based models

- Solutions are defined as a sequence of steps.
- Model a task as a graph of states and a solution as a path in the graph.
- A state captures all of the relevant information about the past in order to act in the future.
- **Apps**: Navigation and Games.
- **Options**: Tree Search (Breadth-first search, Depth-first search, and Iterative deepening), Graph search (Dynamic programming), Markov decision processes, Game playing

3. Variables-based models (Uncertainty)

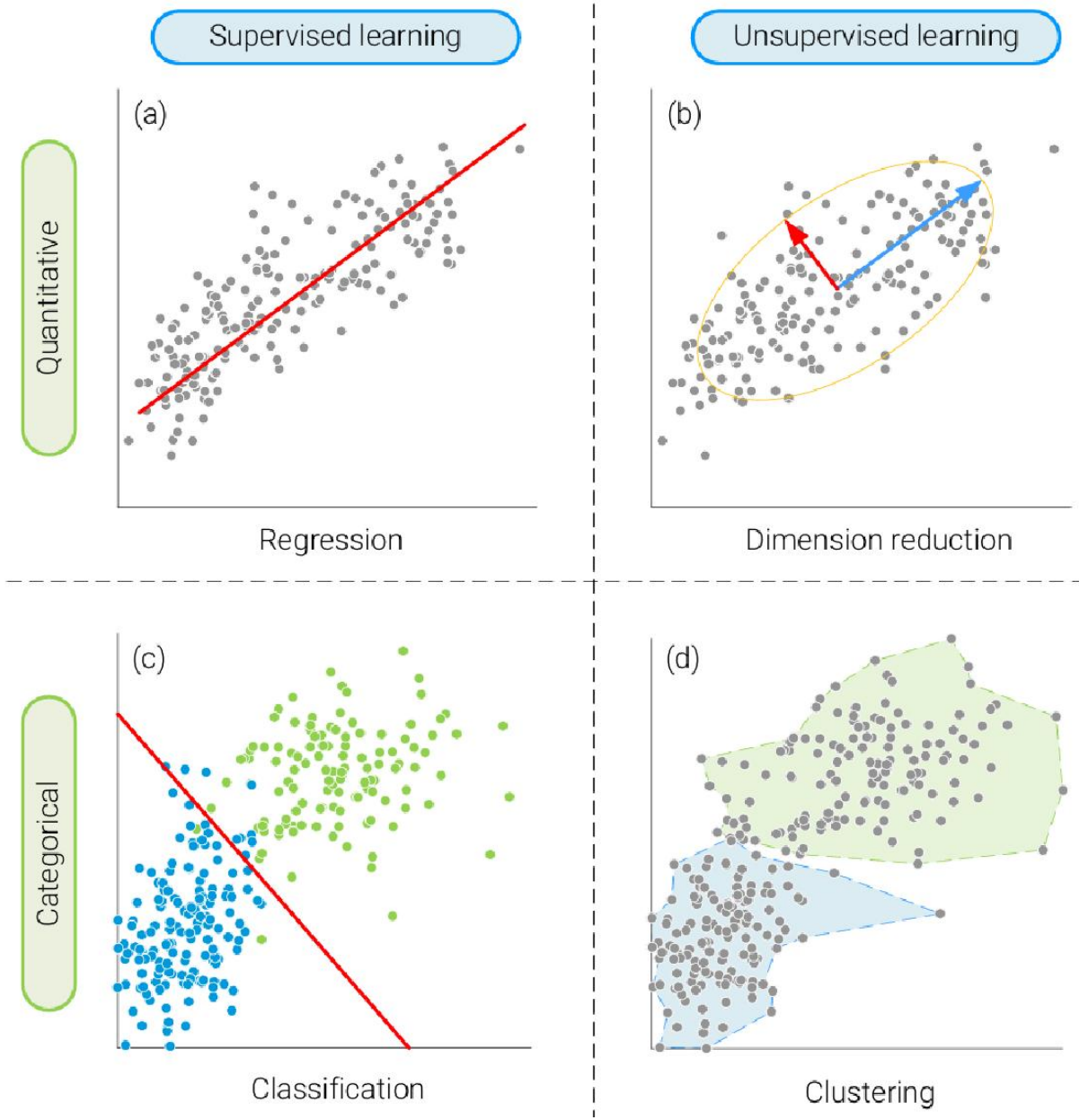
- Solution in an assignment of values for a set of variables.
- **Apps**: Soduko, Speech Recognition, and Face Recognition.
- **Options**: Convolutional Neural Networks, Constraint Satisfaction, Bayesian Networks, Factor Graphs, and Dynamic Ordering.

4. Reflex-based models

- Given a set of <Input, Output> pairs of training data, learn a set of parameters that will map input to output for future data.
- **Apps**: Classification and Regression.
- **Options**: Artificial Neural Networks (ANN), Decision Trees, Support Vector Machines, Regression, Principal Component Analysis, K-Means Clustering, and K-Nearest Neighbor

AI Models Types

Four Main Machine Learning Methods



AI Models Types

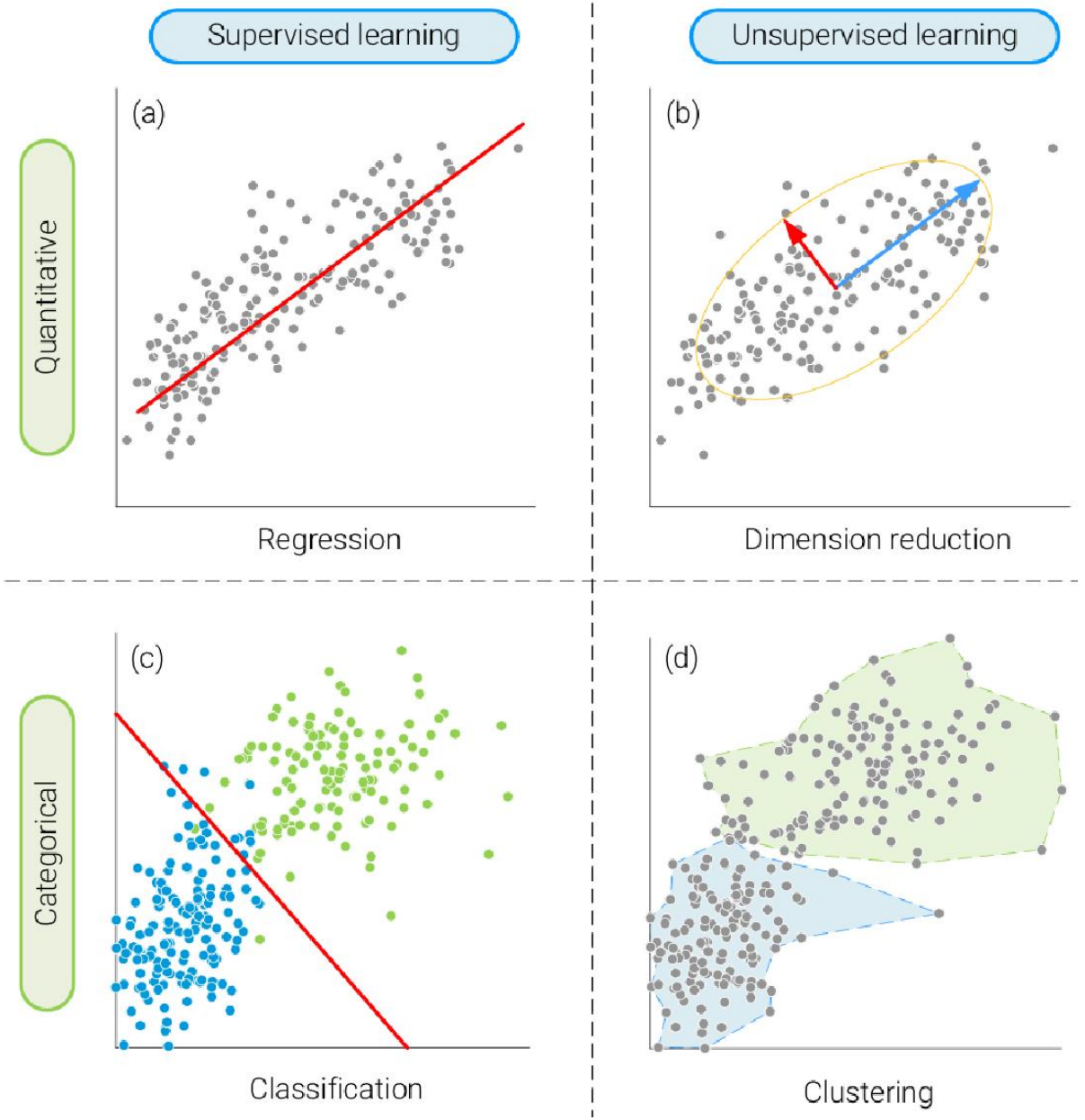
The Big Picture - Learning Paradigms

Supervised Learning: Data comes with Labels (Input X + Output Y)

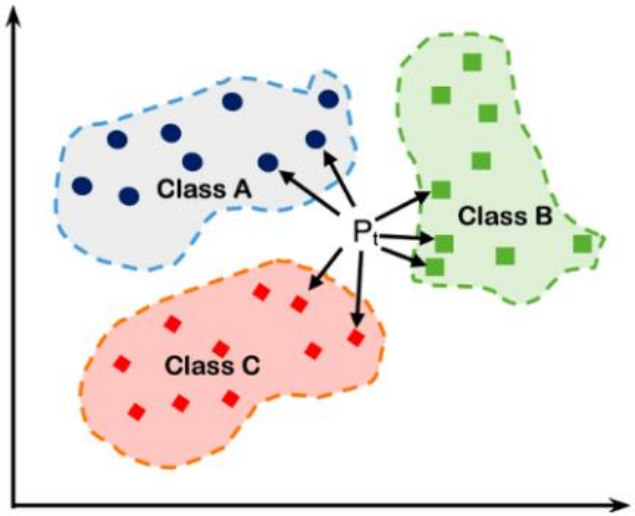
- → Regression
- → Classification

Unsupervised Learning: Data has No Labels (Input X only)

- → Clustering
- → Dimension Reduction



Classification



Source: https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4

Classification

What is Classification?

- Predicts discrete class labels
- Output belongs to a finite set
- Learns decision boundaries
- Used in decision- making systems

Formal Definition of Classification

Classification is a supervised learning process where:

- Inputs are assigned to predefined categories
- Output is categorical
- Model generalizes patterns from labeled data

Regression vs Classification (Core

Aspect	Classification	Regression
Output	Class labels	Numeric values
Nature	Discrete	Continuous
Boundary	Sharp decision	Smooth function
Example	Pass / Fail	Exam score

Classification

When to Use Classification?

- Output categories are known
- Decision matters more than magnitude
- Examples:
  - Approval decisions
  - Diagnosis
  - Recognition tasks

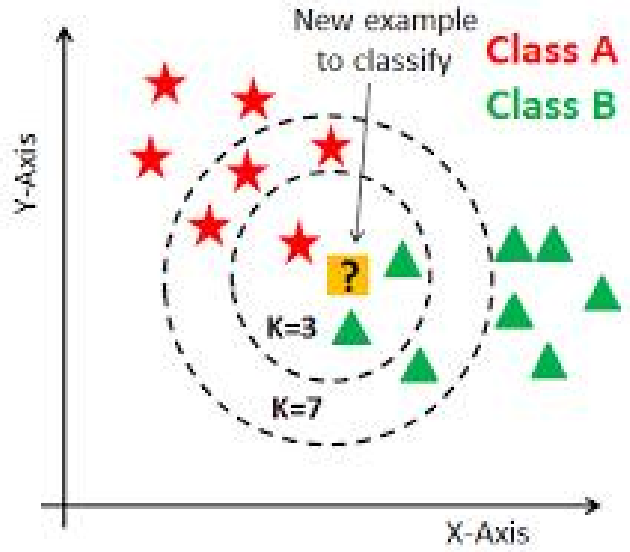
Classification Pipeline

1. Data collection
2. Feature extraction
3. Model training
4. Prediction
5. Evaluation (Accuracy, Precision, Recall)

Common algorithms:

- k-Nearest Neighbors (k-NN)
- Support Vector Machines (SVM)
- Decision Trees
- Naïve Bayes
- Neural Networks

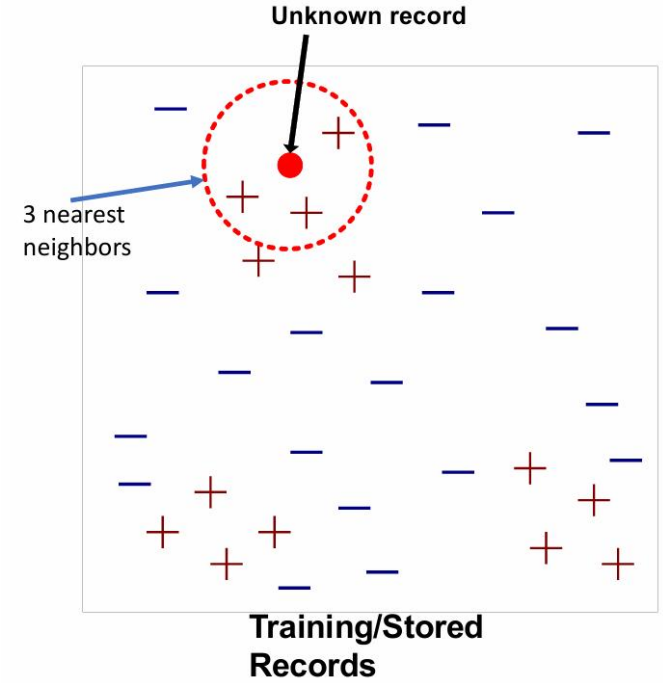
k-nearest neighbors algorithm (KNN)



k-nearest neighbors algorithm

The k-Nearest Neighbors (k-NN) algorithm is a simple, instance-based learning algorithm used for classification and regression tasks in machine learning.

- Instance-Based Learning: k-NN is an instance-based learning algorithm, meaning it memorizes the labeled instances in the training dataset to make predictions.
- **Classification and Regression: k-NN can be used for both classification and regression tasks.**
- Non-Parametric: It is considered a non-parametric and lazy learning algorithm because it does not make explicit assumptions about the form of the mapping function.



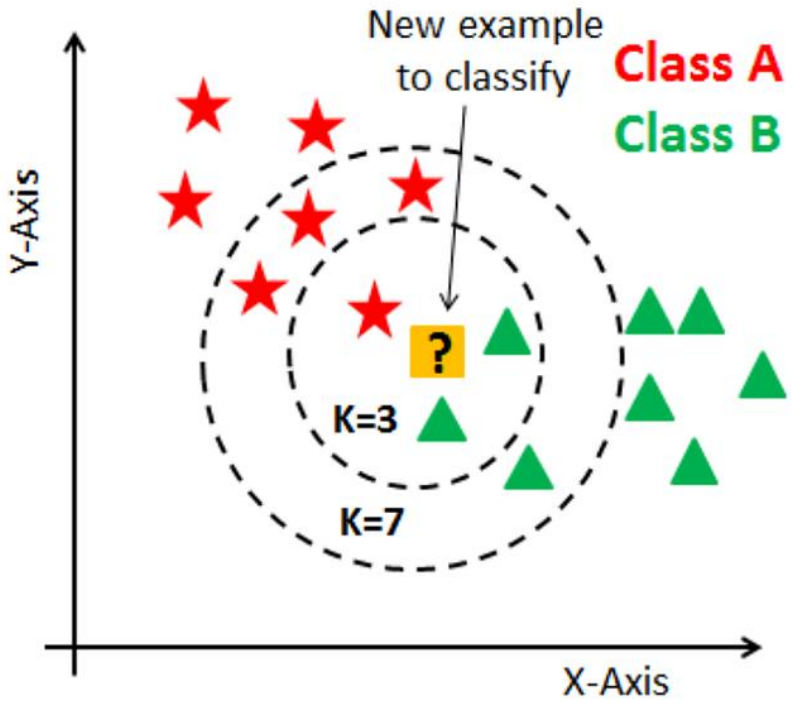
Requires three things:

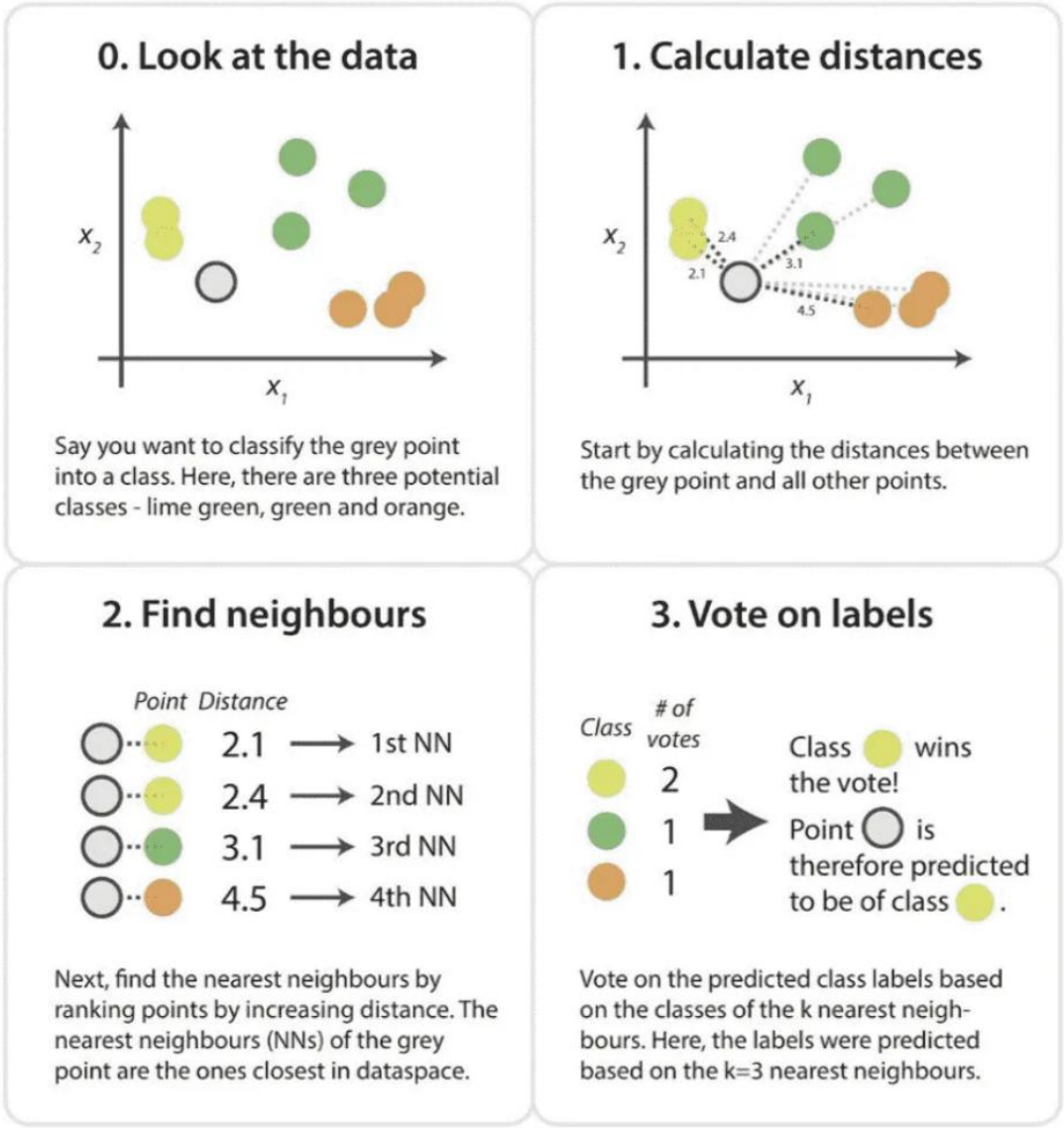
- The set of stored records
- Distance metric to compute distance between records
- The value of k, the number of nearest neighbors to retrieve

To classify an unknown record:

- Compute distance to other training records– Identify k nearest neighbors
- Use class labels of nearest neighbors to determine the class label of unknown record (e.g., by taking majority vote)

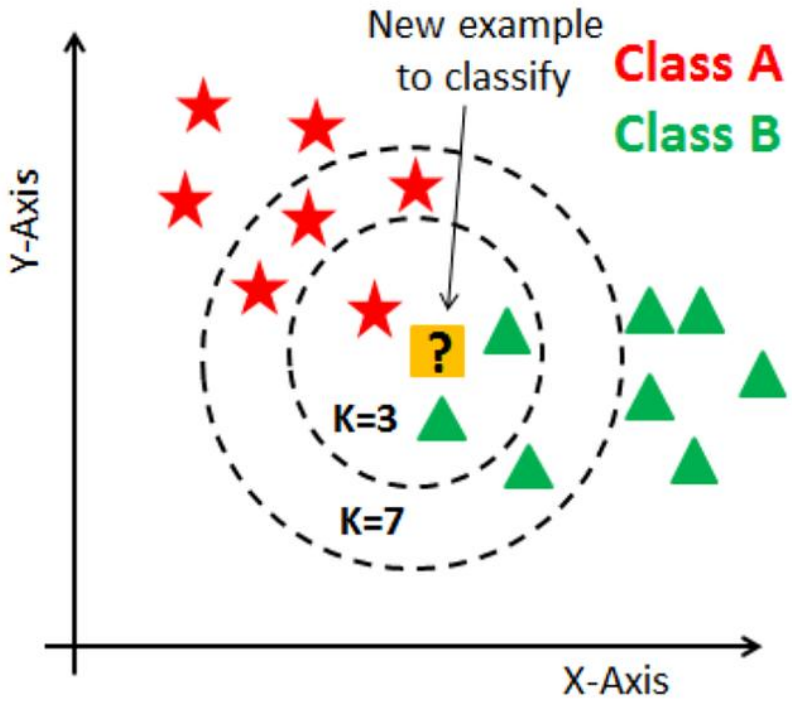
k-nearest neighbors algorithm





k-nearest neighbors algorithm

Choosing k: The choice of the parameter k (number of neighbors) is crucial. A smaller k value makes the model more sensitive to noise, while a larger k value makes the model smoother but might overlook local patterns.



kNN is one of the simplest yet powerful supervised ML algorithms. It is widely used for classification problems as well as can be used for regression problems.

The data-point is classified on the basis of its k Nearest Neighbors, followed by the majority vote of those nearest neighbors; a query point is assigned the data class which has the most representatives within the nearest neighbors of the point.

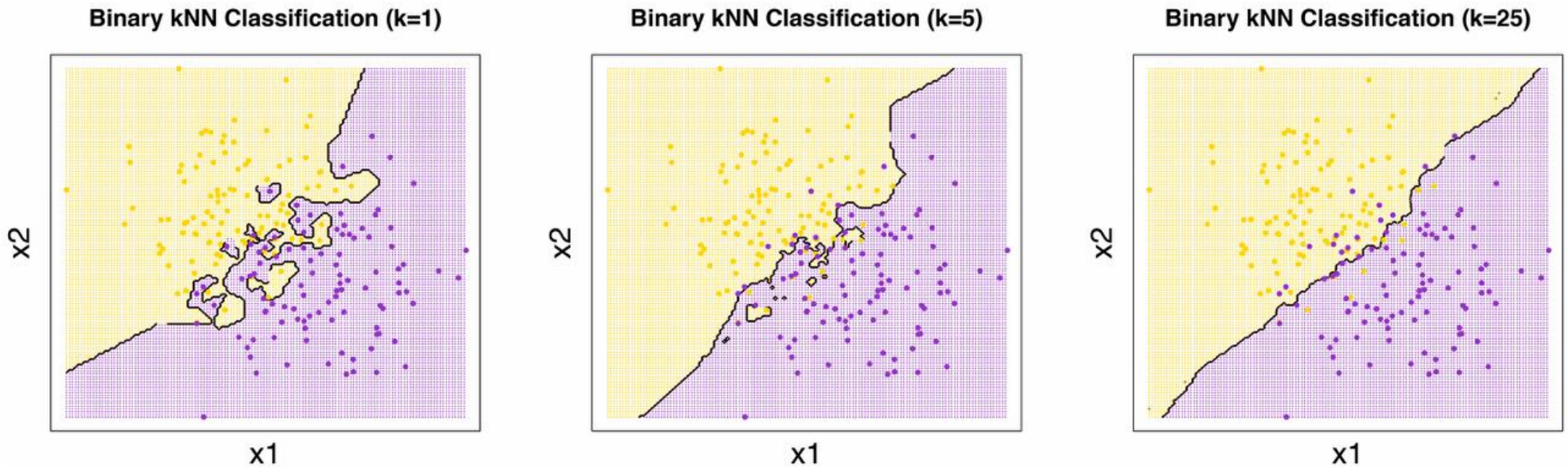
- ☐ When K=1, what is the class of new example?
- ☐ When K=3, what is the class of new example?
- ☐ When K=7, what is the class of new example?

k-nearest neighbors algorithm

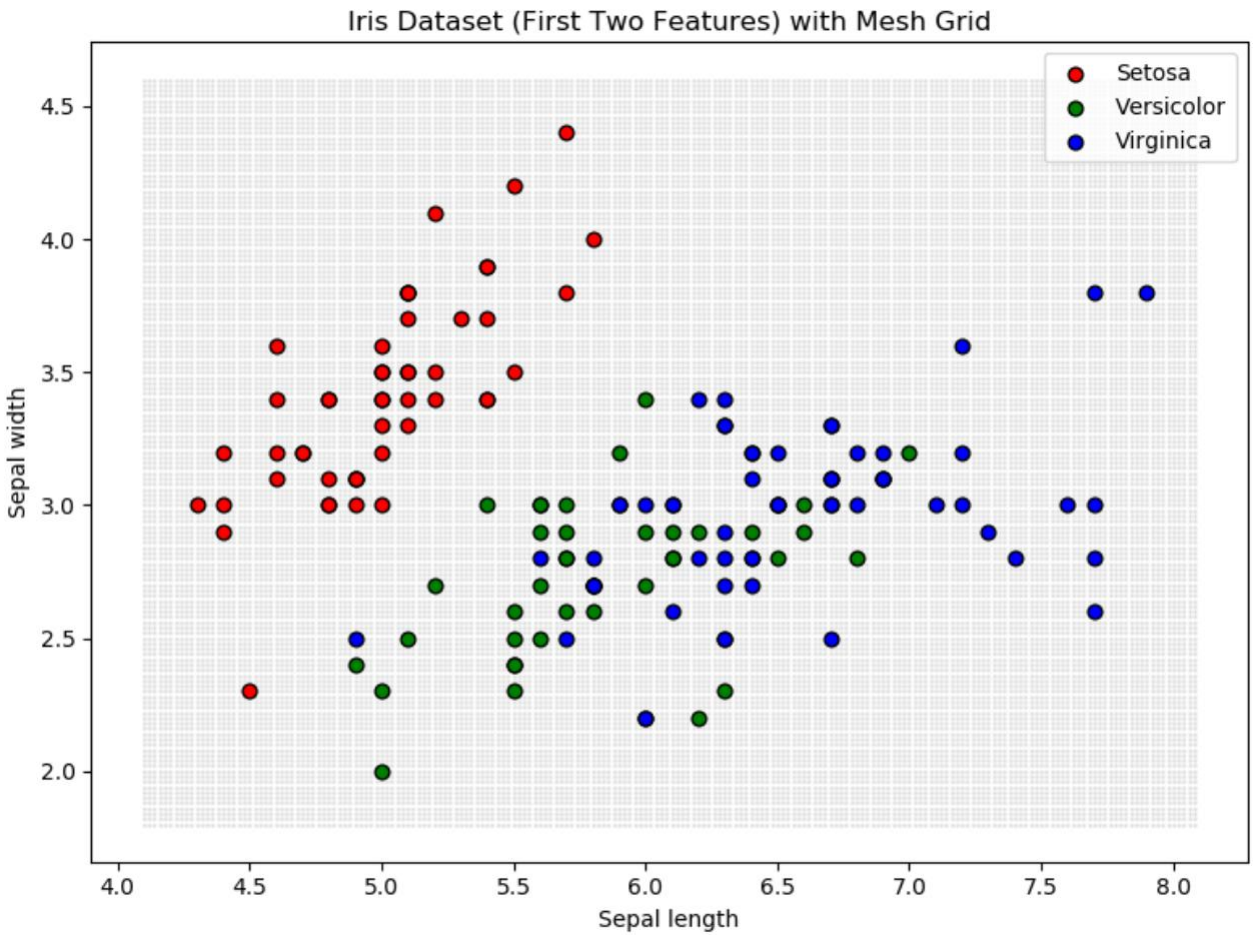
1. **Training**: The algorithm memorizes the feature vectors and their corresponding class labels from the training dataset.
2. **Prediction**:

- For a new input data point, the algorithm calculates the distance (typically Euclidean distance) to all other points in the training set.
- It then selects the k points (neighbors) in the training set that are closest to the new data point.
- For classification, it predicts the class label by **majority** voting among the k-nearest neighbors.
- For regression, it predicts the output by **averaging** the target values of the k-nearest neighbors.

K is a hyperparameter controlling the model complexity

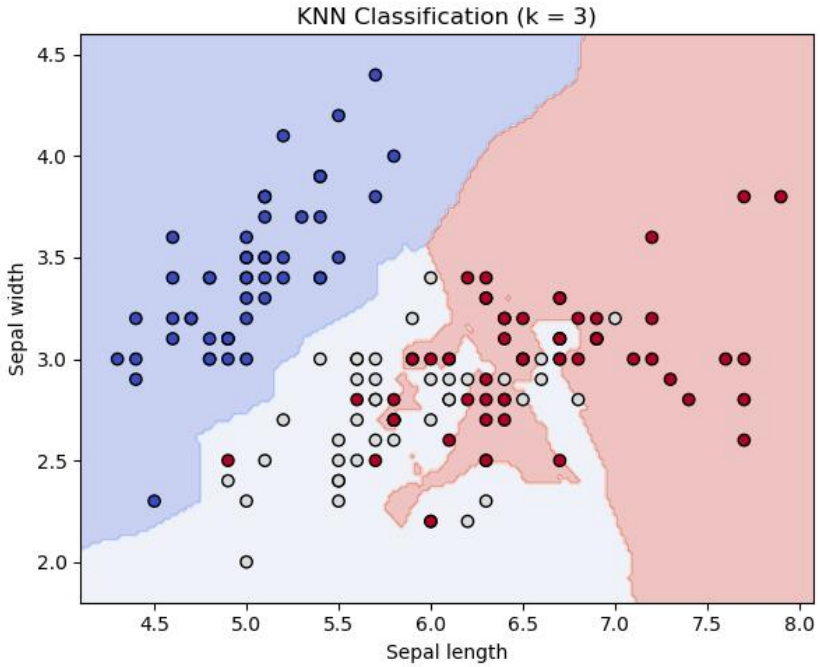


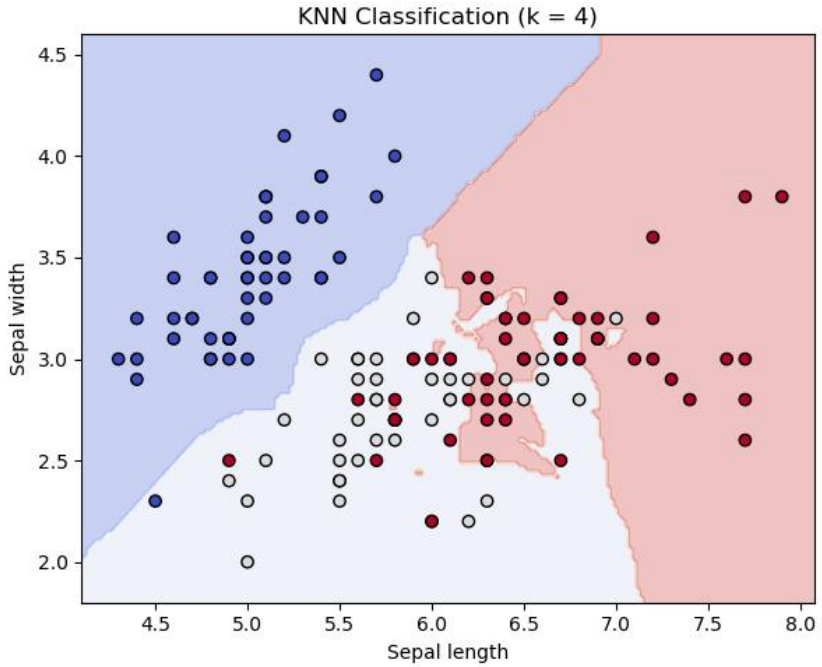
k-nearest neighbors algorithm

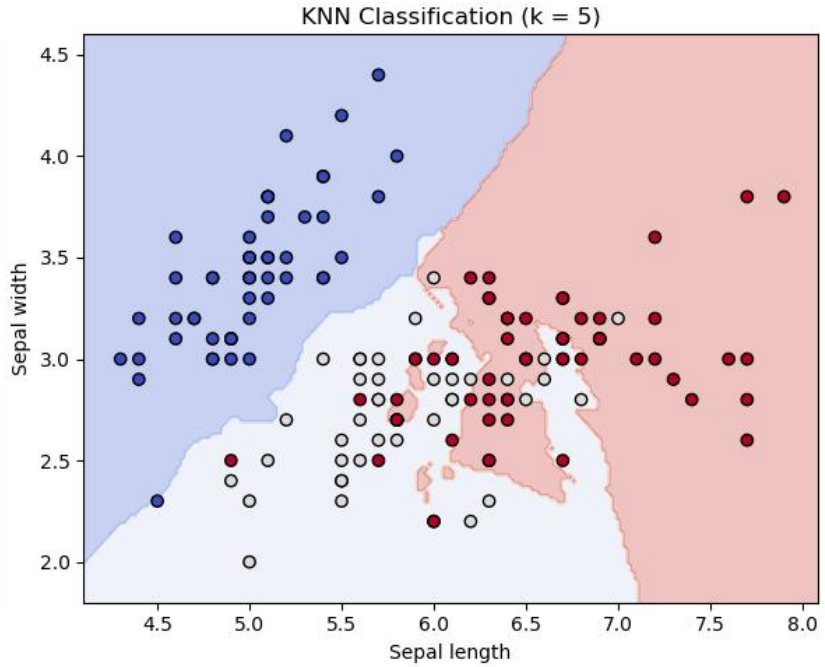



```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets

# 1. 加载 Iris 数据集
iris = datasets.load_iris()

# 2. 只取前两个特征：sepal length, sepal width
X = iris.data[:, :2]
y = iris.target

# 3. 生成 mesh 网格
h = 0.02  # 网格步长
x1_min, x1_max = X[:, 0].min() - 0.2, X[:, 0].max() + 0.2
x2_min, x2_max = X[:, 1].min() - 0.2, X[:, 1].max() + 0.2

xx1, xx2 = np.meshgrid(
    np.arange(x1_min, x1_max, h),
    np.arange(x2_min, x2_max, h)
)
```

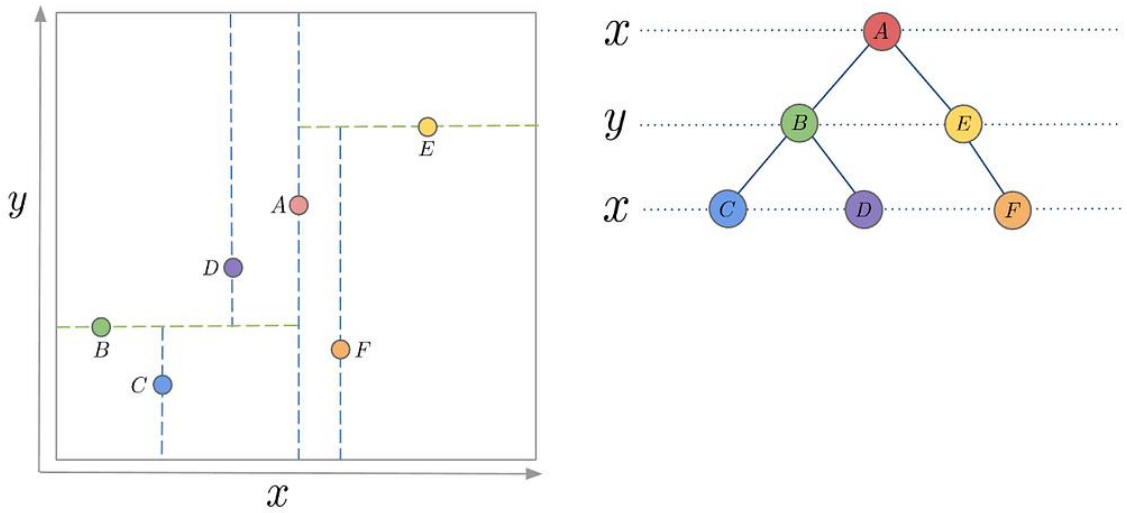
k-nearest neighbors algorithm



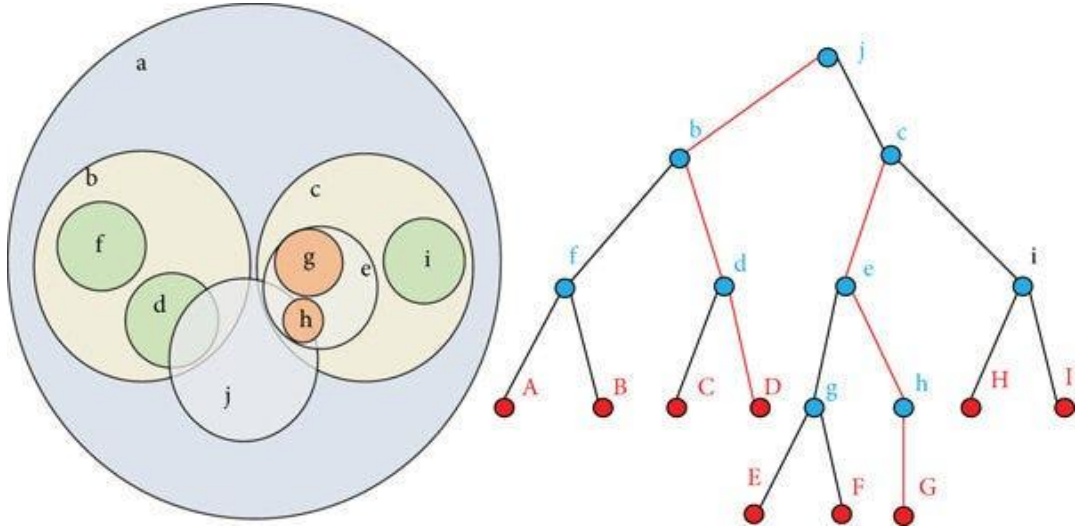




k-nearest neighbors algorithm: nearest-neighbor searching method



KD-tree: Consider splitting products according to the shelves (dimensions). A certain dimension is selected at each level to act as a partition. This hierarchical structure helps you focus your search for a product by guiding you straight to the dimension that's most relevant to you.



Ball tree: As an alternative, think about classifying items according to their physical proximity. Every product becomes the center of a spherical zone that includes nearby products. The dynamic nature of Ball trees reflects changes in product placement over time, adapting to evolving customer preferences.

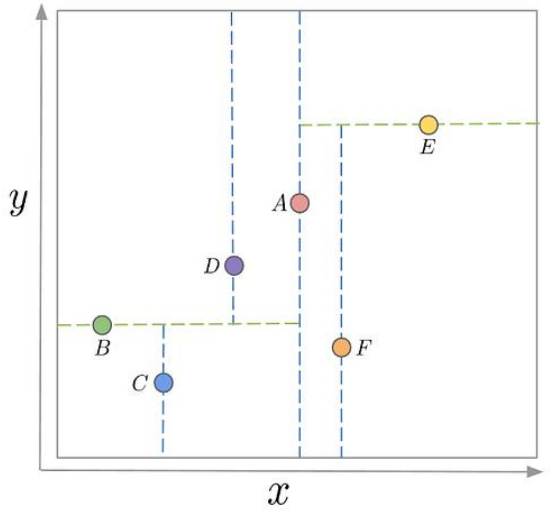
k-nearest neighbors algorithm： KD Tree

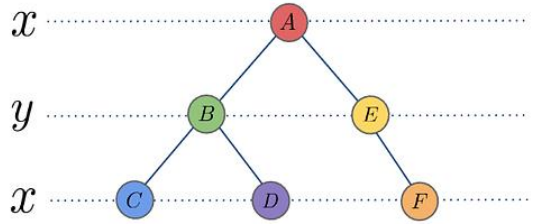
k-Dimensional Tree (kd tree)

A K-Dimensional Tree (also known as K-D Tree) is a space-partitioning data structure for organizing points in a K-Dimensional space. This data structure acts similar to a binary search tree with each node representing data in the multi dimensional space.

When this algorithm is used for k-NN classification, it rearranges the whole dataset in a binary tree structure, so that when test data is provided, it would give out the result by traversing through the tree, which takes less time than brute search.

A hierarchical data structure called a KD-Tree (K-Dimensional Tree) is used for multidimensional space partitioning. It creates a binary tree with each node representing a part of the space by iteratively dividing the space along predefined dimensions. Often used for effective closest neighbor searches.



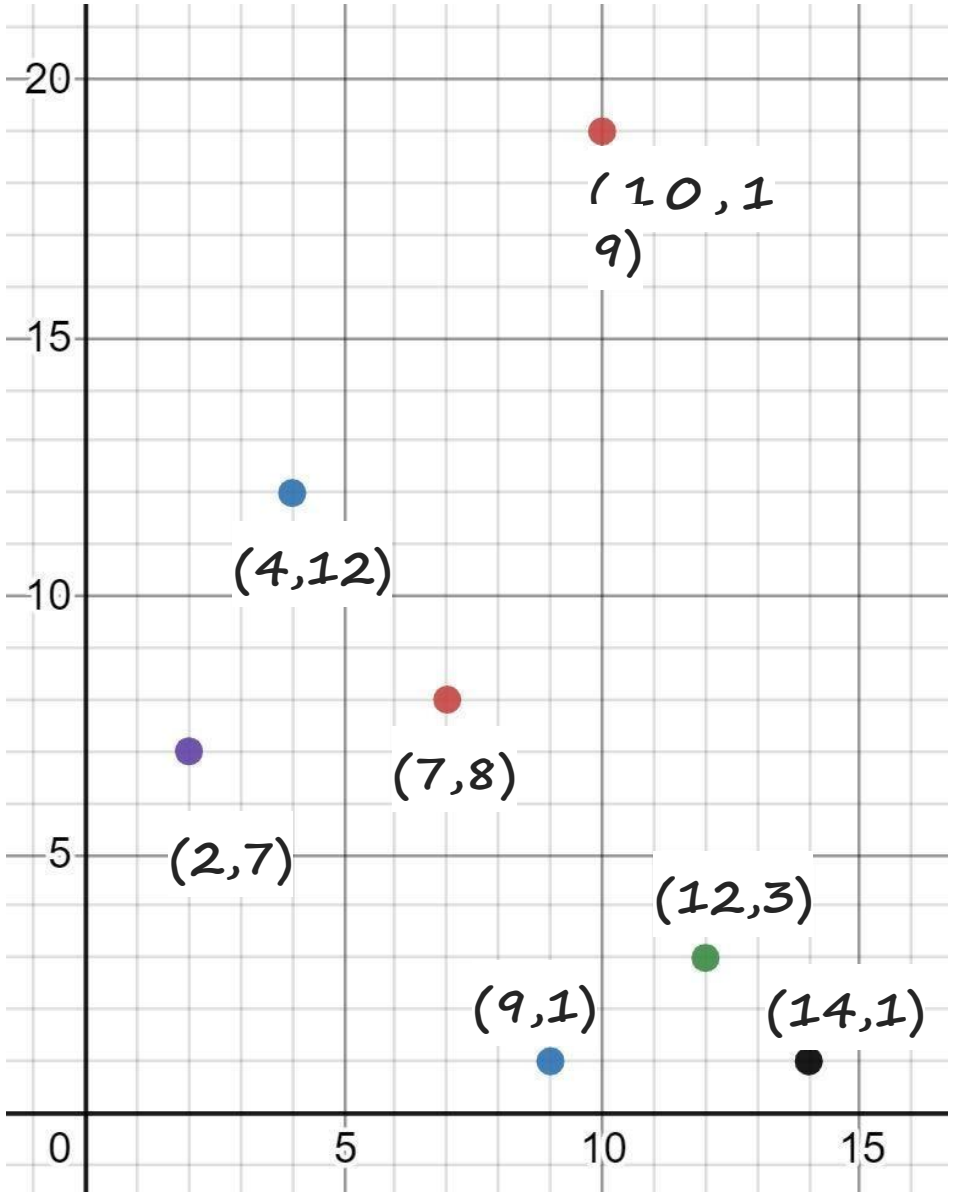


k-nearest neighbors algorithm

k-Dimensional Tree (kd tree)

A simple example to showcase the insertion into a K-Dimensional Tree, we will use a k = 2.
The points we will be adding are: (7,8), (12,3), (14,1), (4,12), (9,1), (2,7), and (10,19).

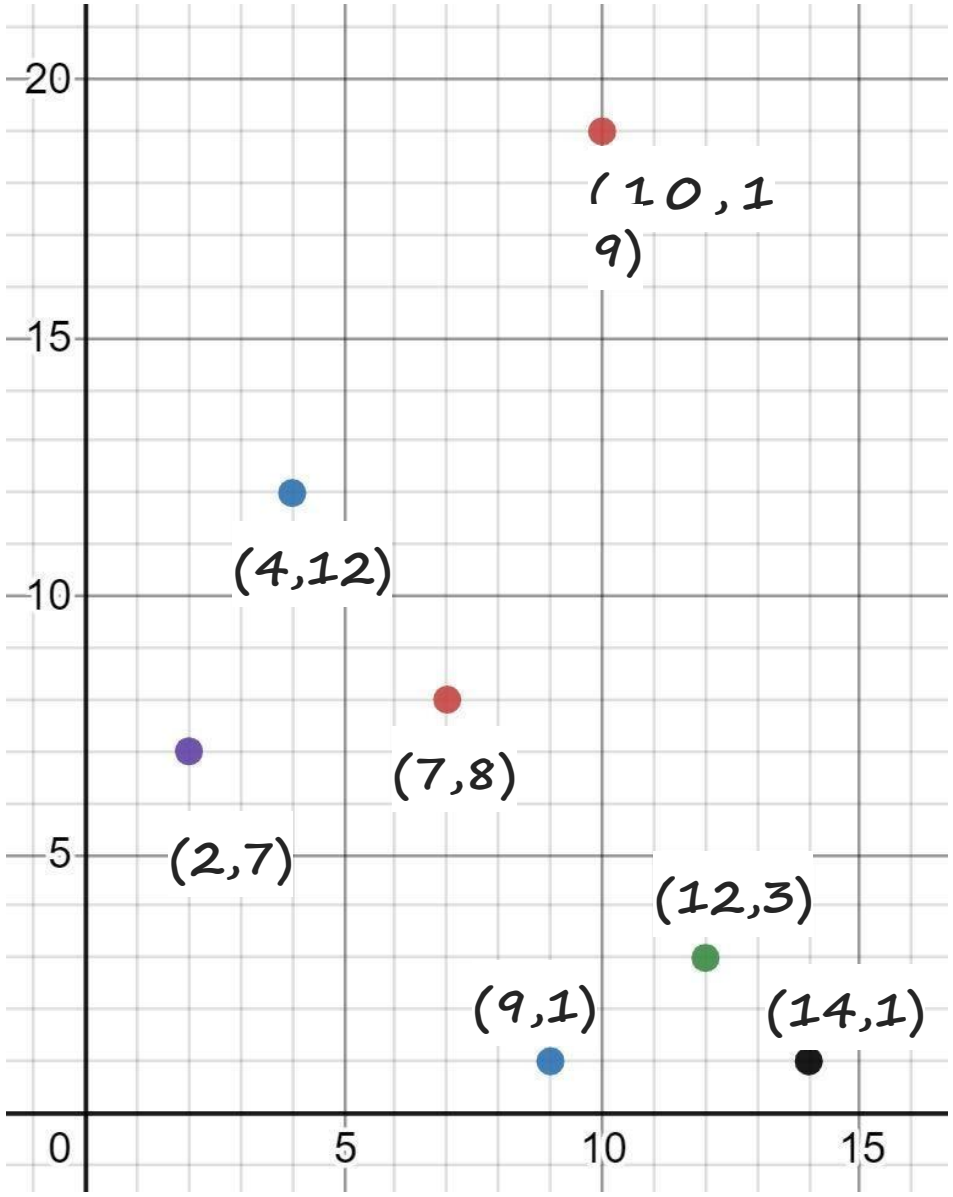
1. The first element inserted is (7,8). It will serve as the **base node** in the following K-D Tree example.
2. The second element inserted is (12,3). It is placed to the right leaf node of (7,8) because the X value, 12, is greater than the X value of base, 7.



k-nearest neighbors algorithm

k-Dimensional Tree (kd tree)

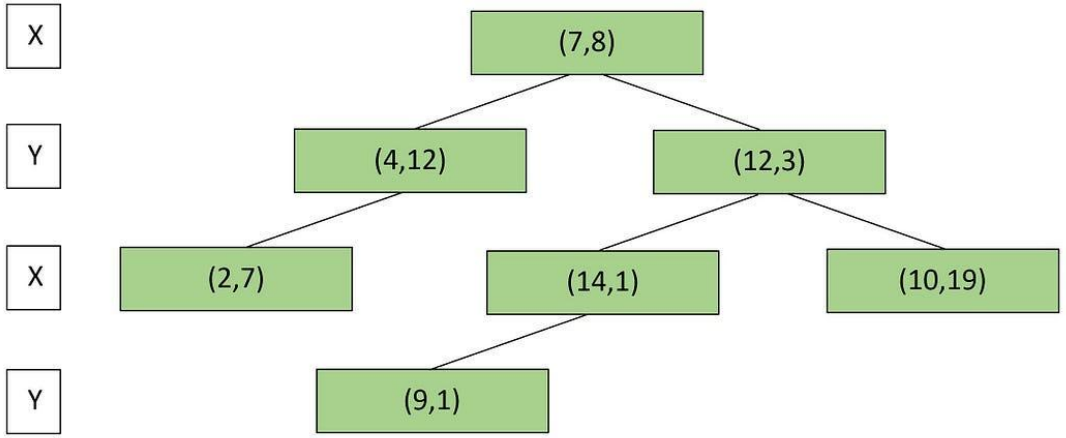
3. The next element inserted into the K-D Tree is (14,1). At the first level, we compare the X values of the set and since 14 is greater than 7 it moves to the right of the tree. Next is the comparison between (12,3) and (14,1). **Each level the comparison operator changes**, which means that we will be comparing the Y value of each set. Since the Y of the inserted set is 1 which is less than 3, it is inserted to the left leaf node of (12,3).

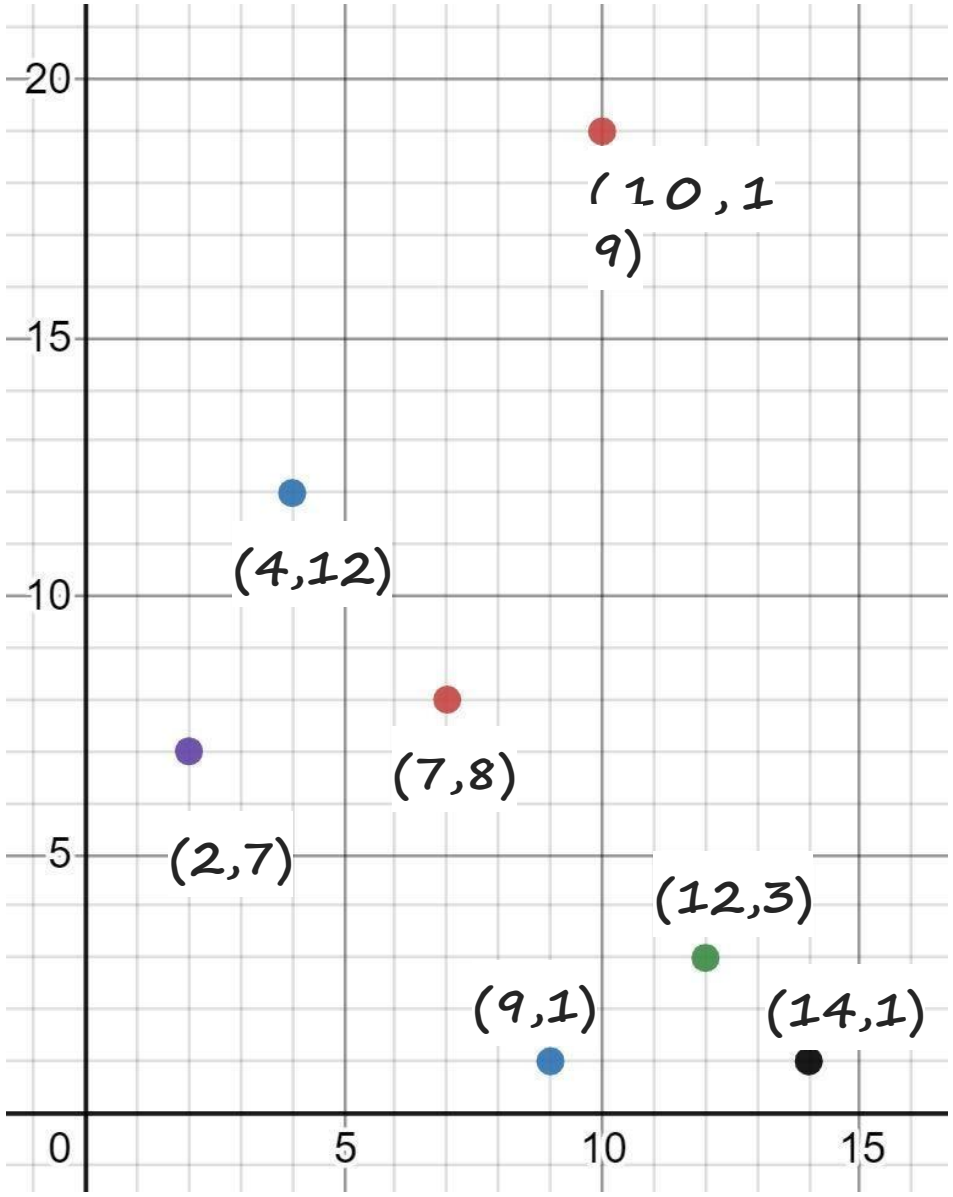


k-nearest neighbors algorithm

k-Dimensional Tree (kd tree)

4. The next set inserted is (4,12). Because the X, 4, is less than the base nodes X, which is 7, the set is inserted to the left leaf node.
5. The next set is (9,1). Since the X is greater than the base, it moves to the right leaf.
6. The next set inserted is (2,7). Since the X is less than 7 it compares to the left leaf node.
7. The next set inserted is (10,19). 10 is greater than 7 so it moves to the right leaf node.

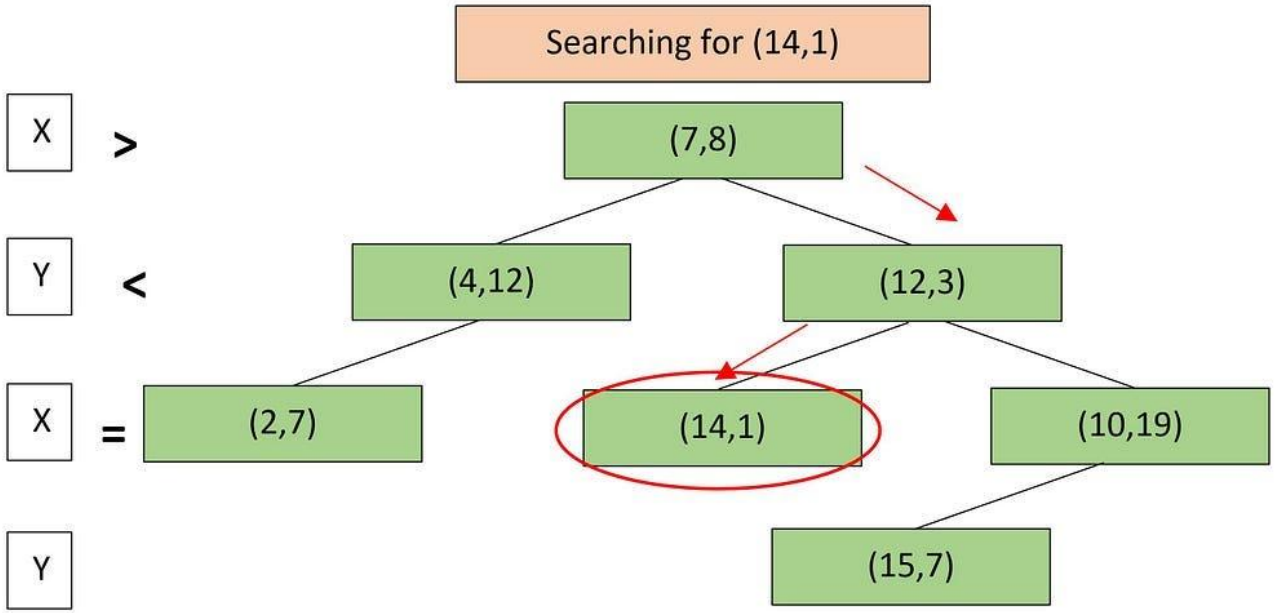


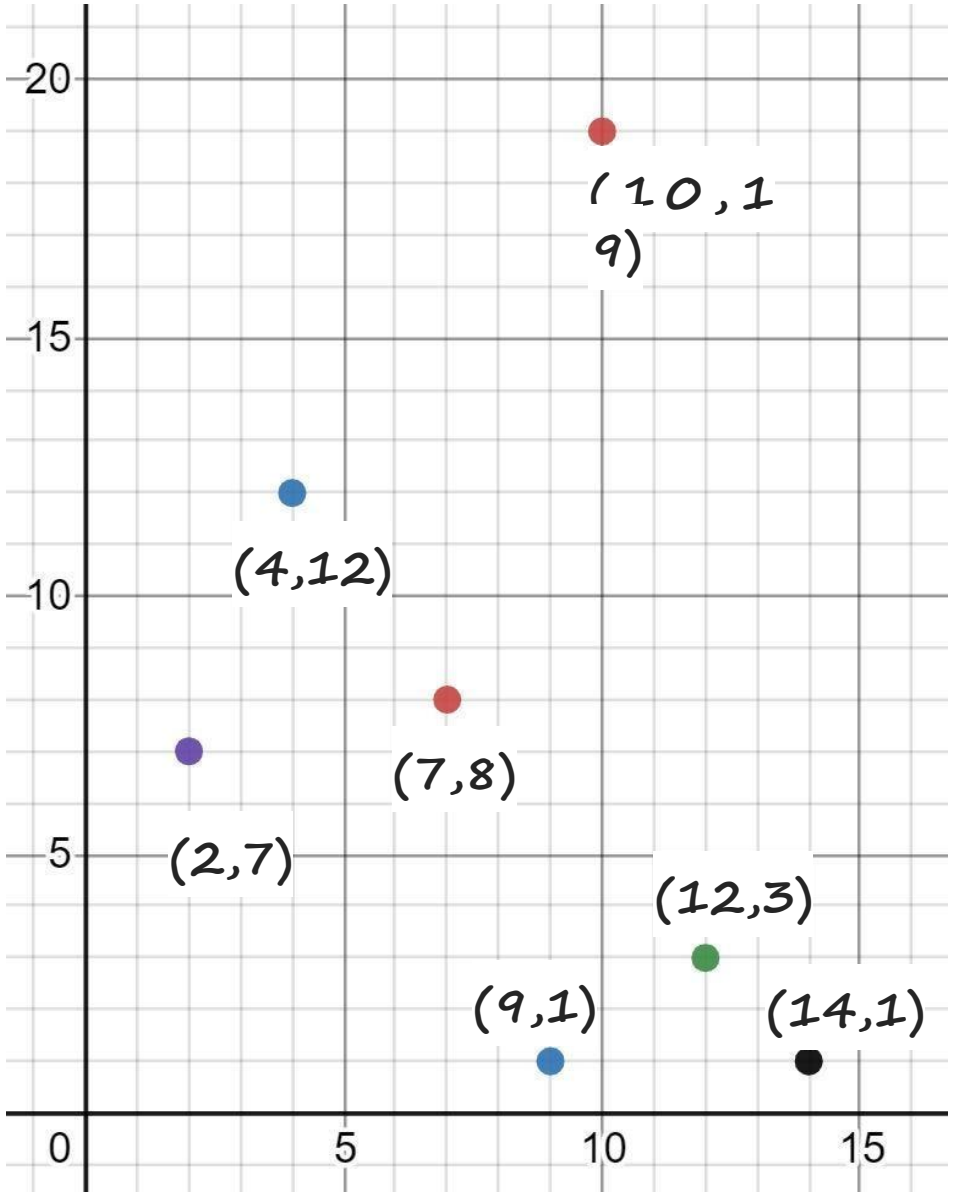


k-nearest neighbors algorithm

k-Dimensional Tree (kd tree)

Based on the above example, lets say we are searching for the point (14,1).

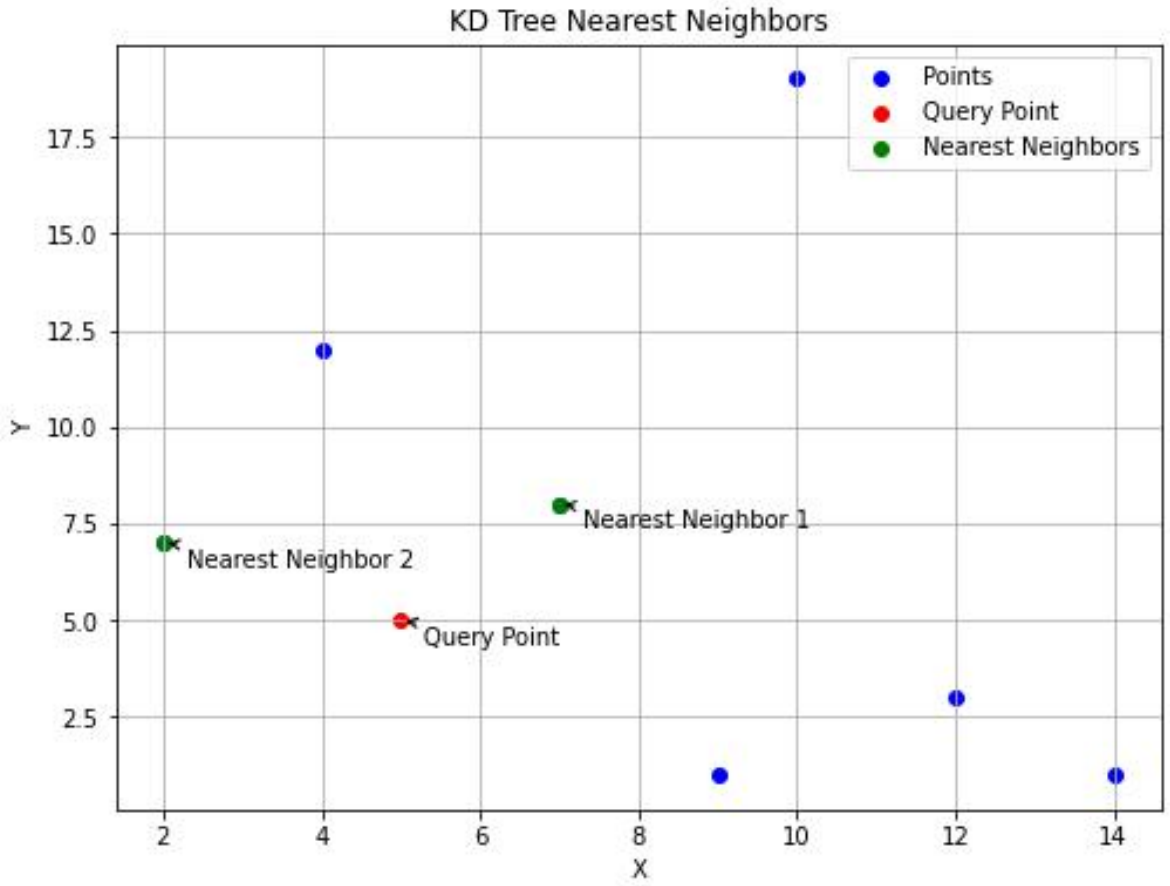




k-nearest neighbors algorithm

k-Dimensional Tree (kd tree)

```
import numpy as np
from sklearn.neighbors import KDTree

# Points to be added to the KD Tree
points = np.array([(7,8), (12,3), (14,1), (4,12), (9,1), (2,7),  (10,19)])

# Construct the KD Tree
kd_tree = KDTree(points, leaf_size=2)

# Query point
query_point = np.array([(5, 5)])

# Query the KD Tree for nearest neighbors
distances, indices = kd_tree.query(query_point, k=2)
```



k-nearest neighbors algorithm

k-Dimensional Tree (kd tree)

- **Advantages：** One of the main advantages of KD-Trees is their **ability to handle high-dimensional data**, which is often encountered in machine learning applications such as computer vision, natural language processing,

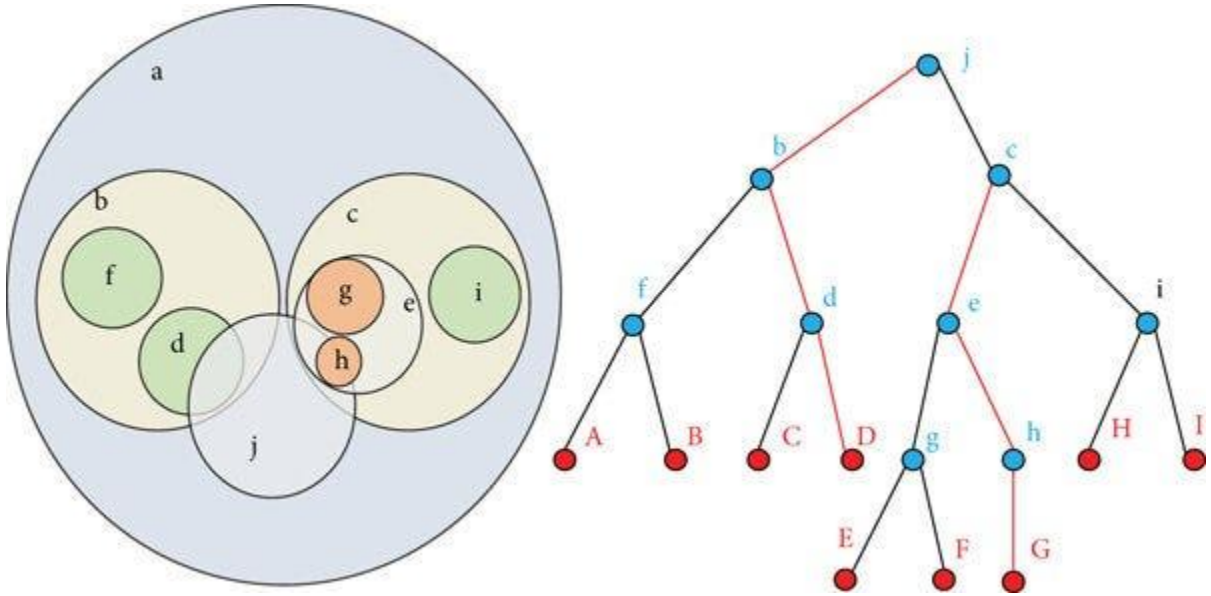
- **Disadvantages：** KD trees have some limitations, including **performance degradation as the number of dimensions increases, especially when data points are not uniformly distributed**. This can lead to unbalanced trees and inefficient search times

- k-d trees are **not suitable for efficiently finding the nearest neighbour in high-dimensional spaces**. As a general rule, if the dimensionality is k, the number of points in the data, N, should be N>> 2k.

k-nearest neighbors algorithm: Ball Tree

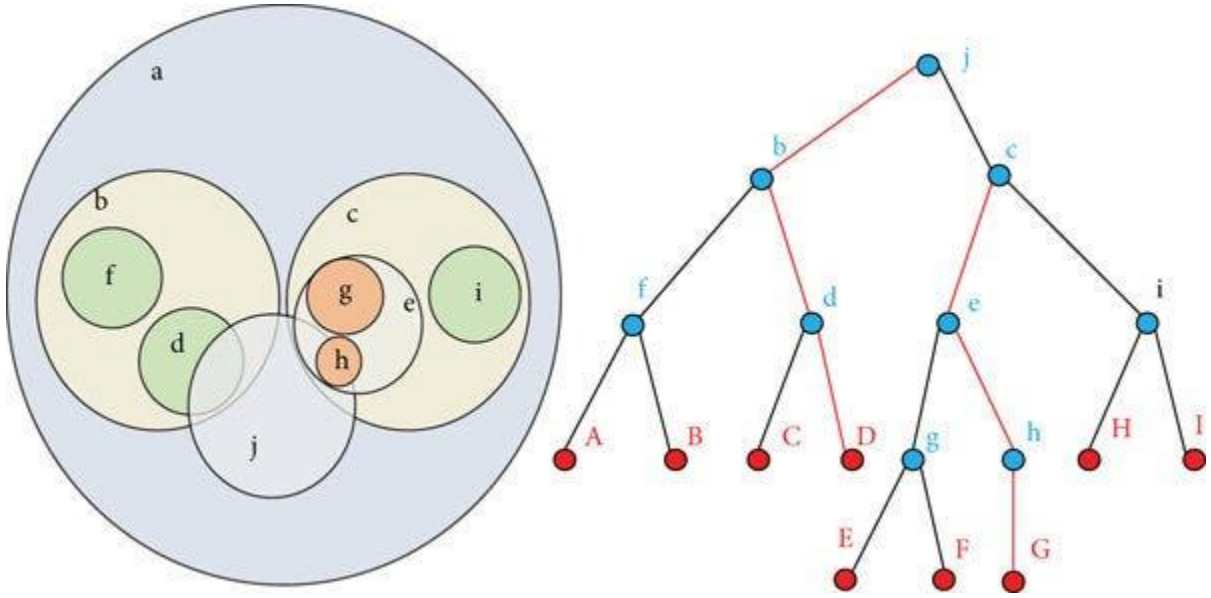
A Ball Tree is another data structure that is used for efficiently organizing and searching multidimensional data. Unlike a kd — Tree, which divides the space with axis-aligned splits, a Ball Tree groups the data points into balls (or hyperspheres) that are organized into a tree structure. This allows for more flexible and compact representation of the data, especially for high-dimensional and non-uniform data.

It is a tree-based data structure designed for organizing and querying points in multidimensional spaces. It is also a binary tree with a hierarchical (binary) structure. The ball tree data structure is particularly effective when there are a lot of dimensions. Unlike KD-trees, Ball trees use hyperspheres to represent partitions, grouping nearby points within each hypersphere.



k-nearest neighbors algorithm: Ball Tree

A Ball Tree is another data structure that is used for efficiently organizing and searching multidimensional data. Unlike a kd — Tree, which divides the space with axis-aligned splits, a Ball Tree groups the data points into balls (or hyperspheres) that are organized into a tree structure. This allows for more flexible and compact representation of the data, especially for high-dimensional and non-uniform data.



Advantages:

- Well-suited for non- Euclidean distances.
- It can adapt to dynamic dataset.
- Excel at handling high-dimensional spaces.

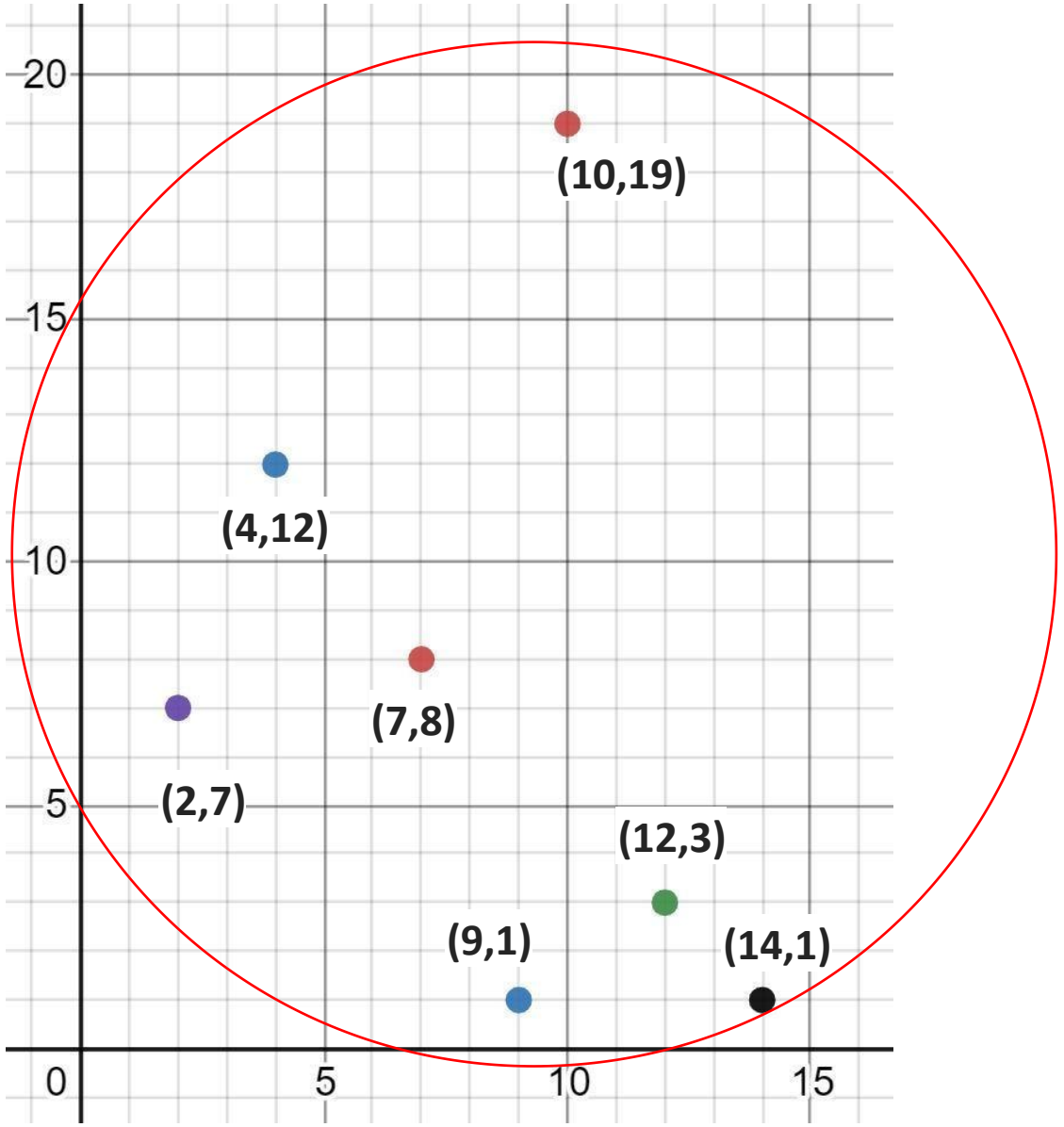
Disadvantages:

- It has complex structure.
- Querying it can be computationally expensive, especially as the dimensionality of the space increases.
- It occupies more memory.

k-nearest neighbors algorithm: Ball Tree

A simple example of a Ball — Tree with two dimensions (x and y). Suppose we have the following data points: (7,8), (12,3), (14,1), (4,12), (9,1), (2,7), and (10,19).

1. The first step is to choose a data point as the center of the hypersphere. We can start with any data point, such as (7, 8). We then find the radius that covers all the data points, which is the distance to the farthest data point, such as (10, 19). We then draw a circle with center (7,8) and radius 11.40, and enclose all the data points in the circle. This creates the root node of the tree.



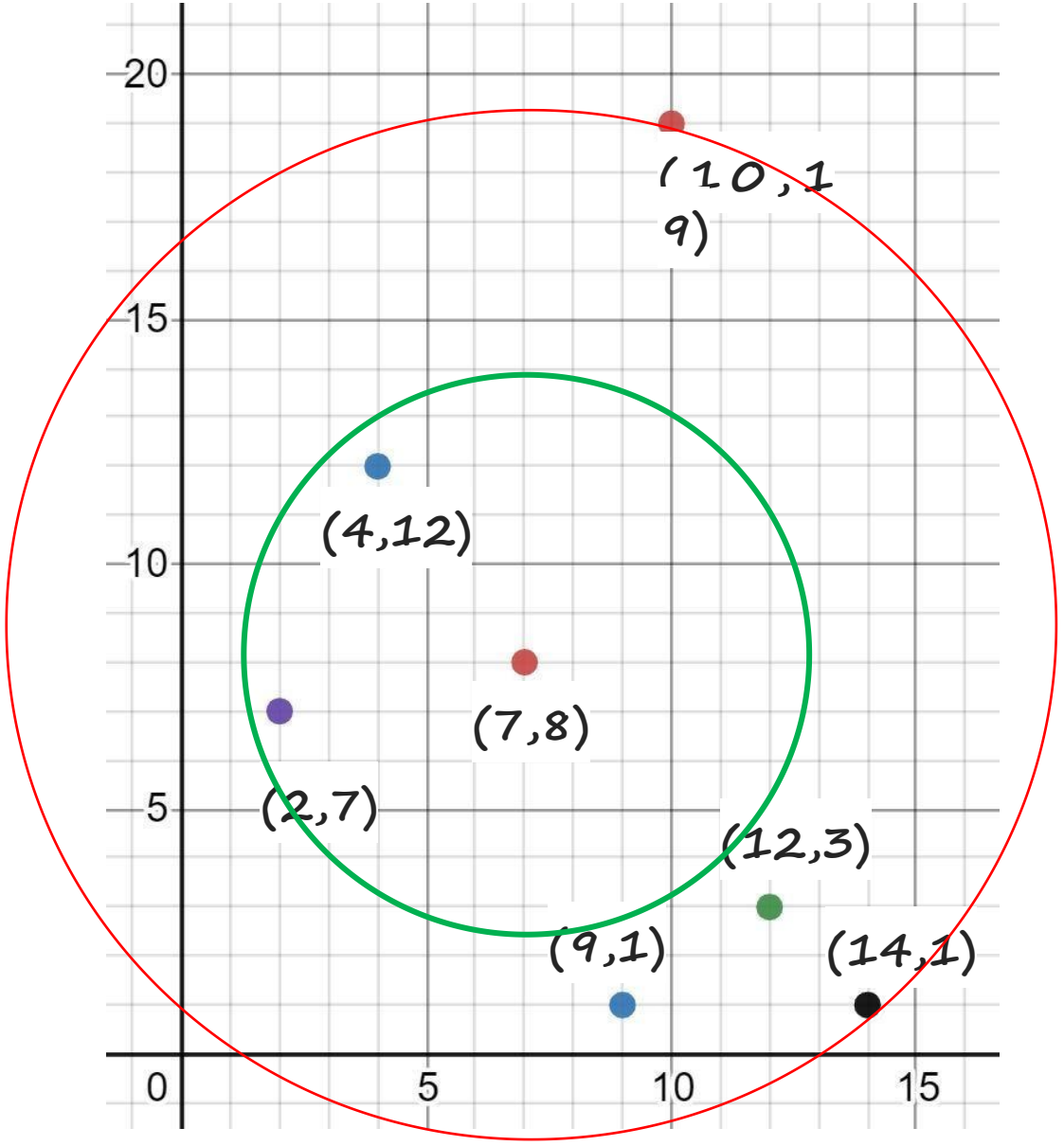
Center: Chosen as the mean of all the points:
Center=(8.29,7.29)
However, to simplify, we can choose the central point (7,8).

k-nearest neighbors algorithm: Ball Tree

A simple example of a Ball — Tree with two dimensions (x and y). Suppose we have the following data points: (7,8), (12,3), (14,1), (4,12), (9,1), (2,7), and (10,19).

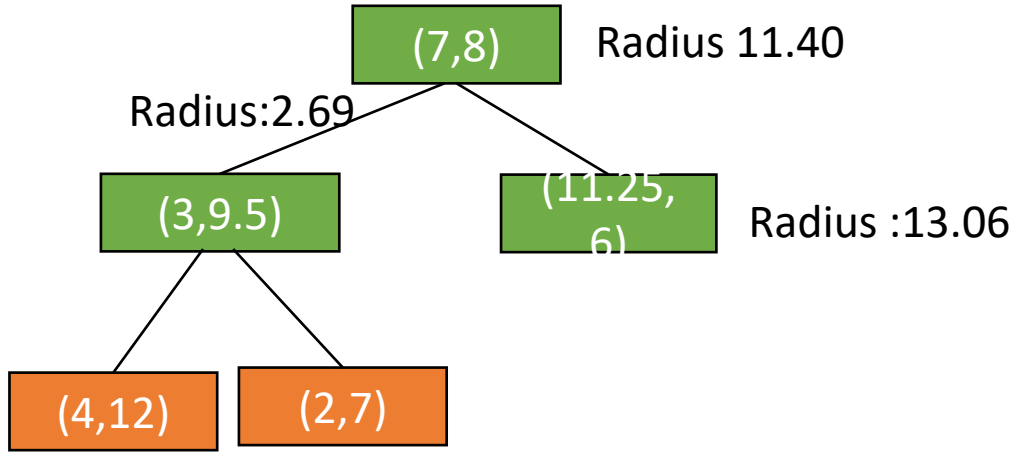
1. The first step is to choose a data point as the center of the hypersphere. We can start with any data point, such as (7, 8). We then find the radius that covers all the data points, which is the distance to the farthest data point, such as (10, 19). We then draw a circle with center (7,8) and radius 11.40, and enclose all the data points in the circle. This creates the root node of the tree.

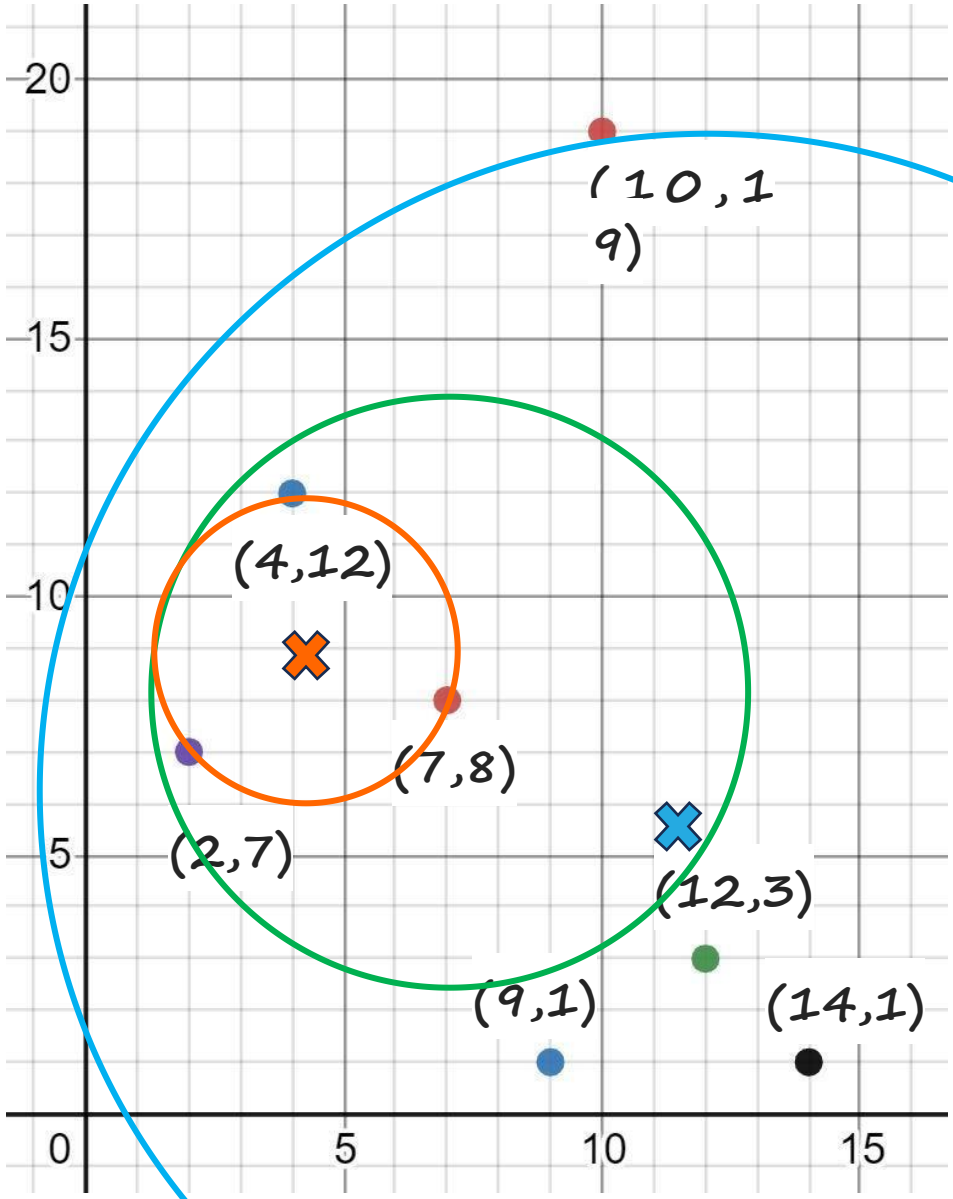
2. The next step is to split the data in the root node into two subsets. We can use any criterion to split the data, such as the distance to the center, the variance along a dimension, or the distance to a random point. For simplicity, let us use the distance to the center, and split the data into two subsets: those with distance <= 11.40 /2 and those with distance > 11.40 /2. This creates two child nodes.



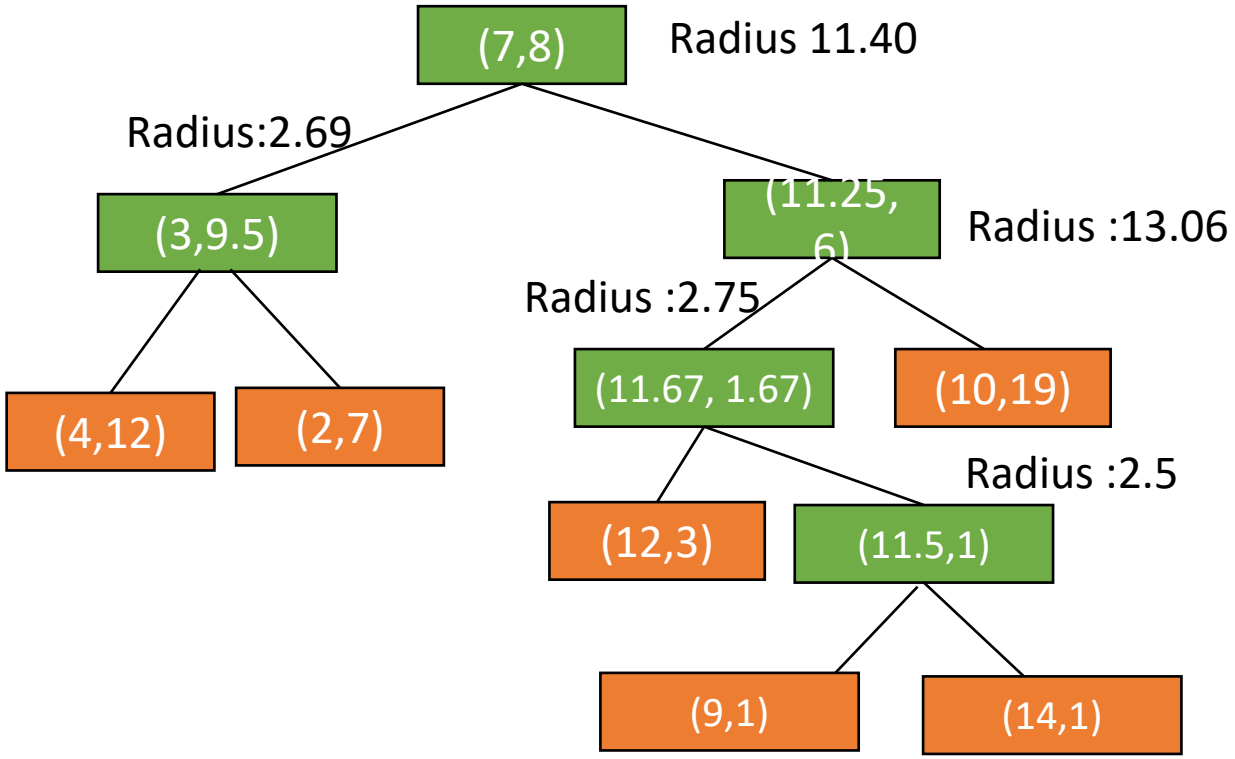
k-nearest neighbors algorithm: Ball Tree

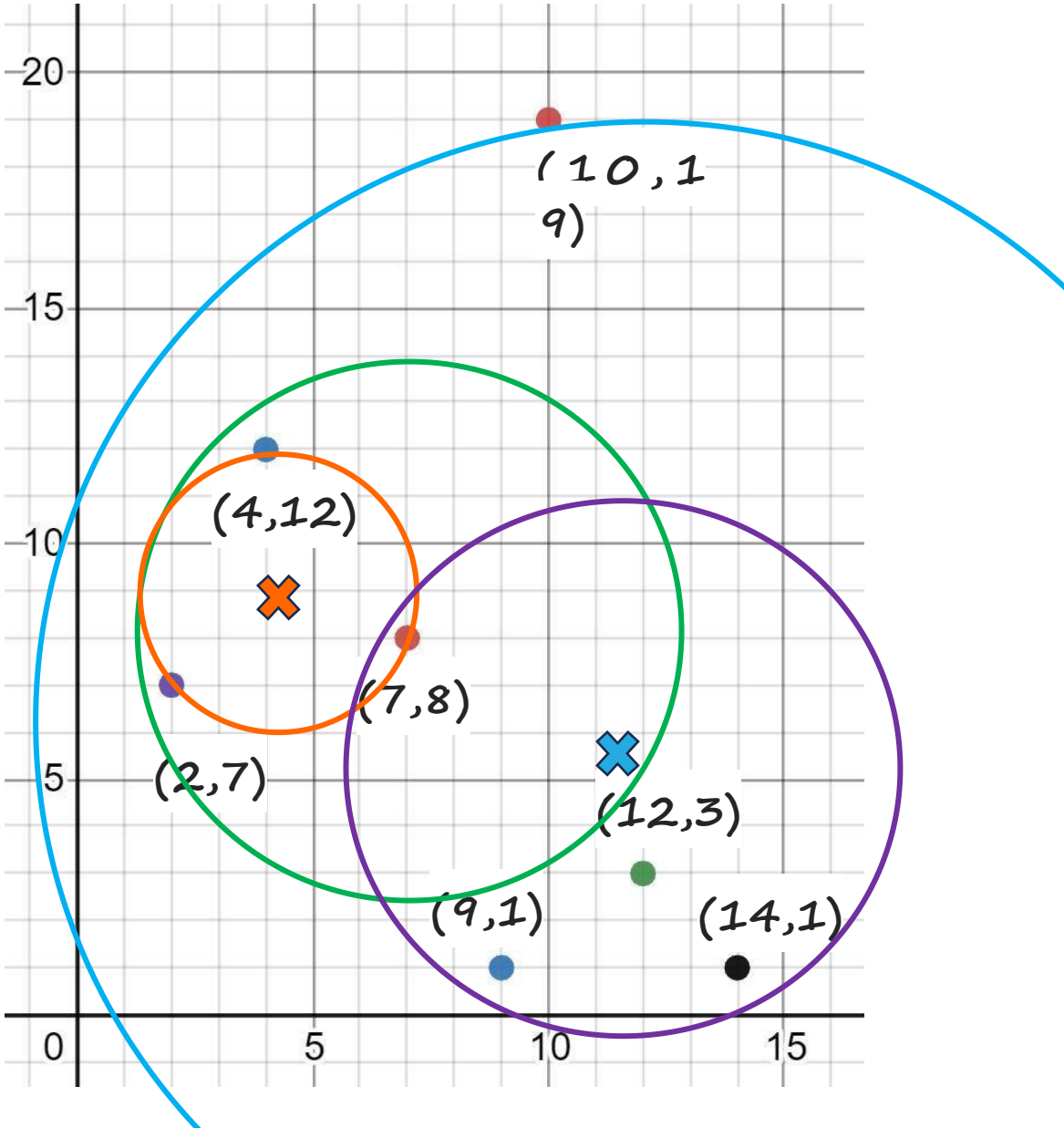
3. The next step is to choose a data point as the center of the hypersphere for each child node, and find the radius that covers all the data points in the node. For the left child node, we can choose the mean (4.33, 9) as the center, and find the radius as 3.02. We then draw a circle with center (4.33, 9) and radius 3.02, and enclose all the data points in the circle. This creates another child node.



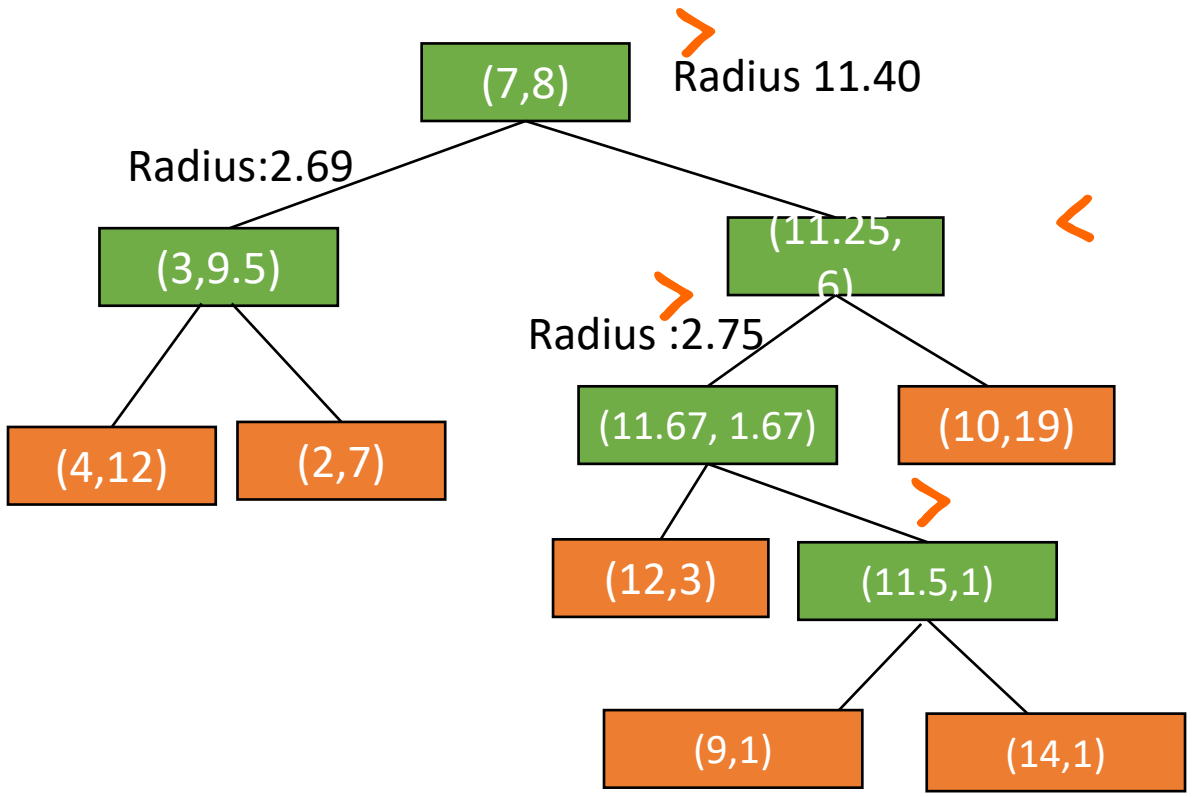


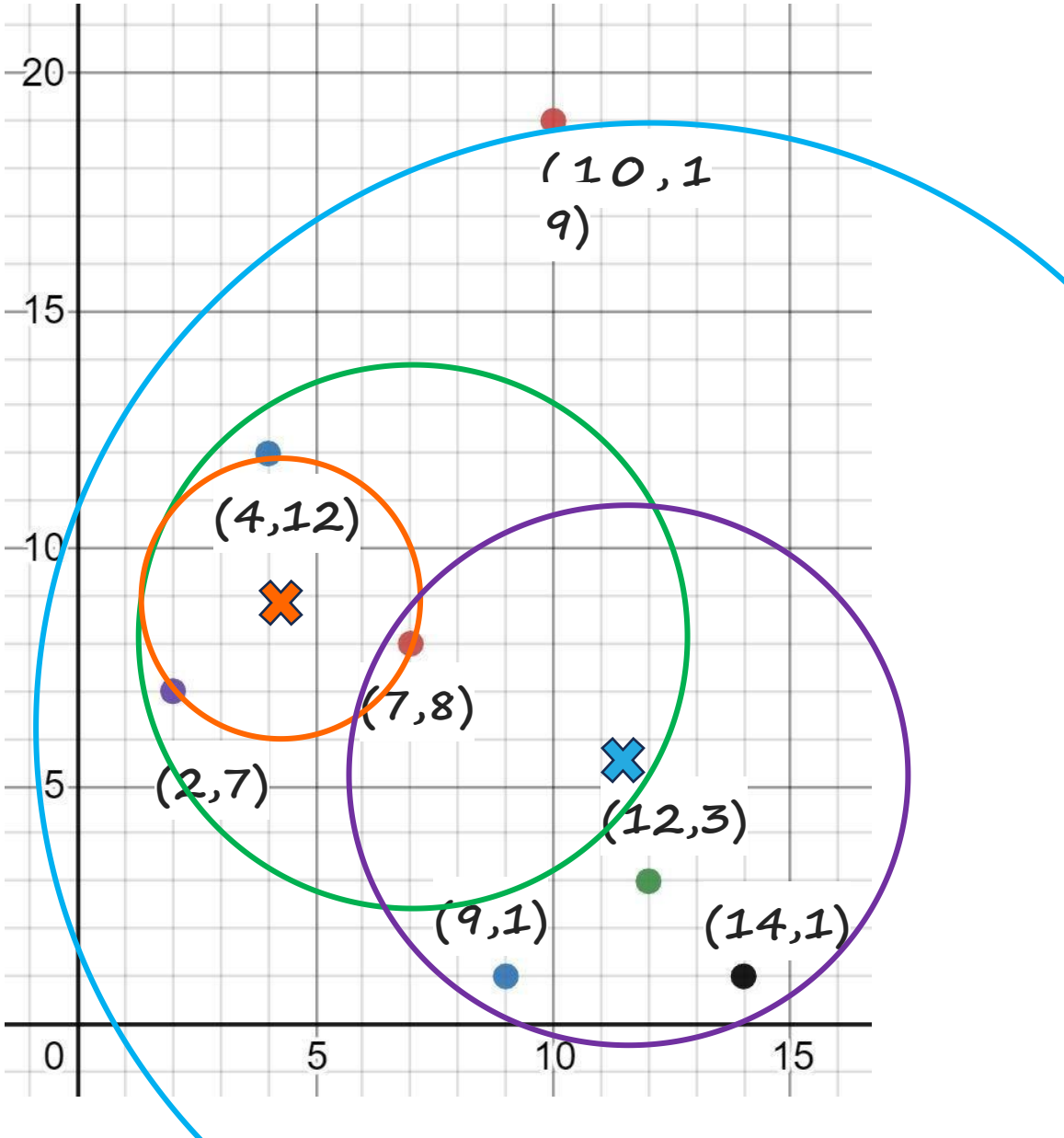
k-nearest neighbors algorithm: Ball Tree





k-nearest neighbors algorithm: Ball Tree

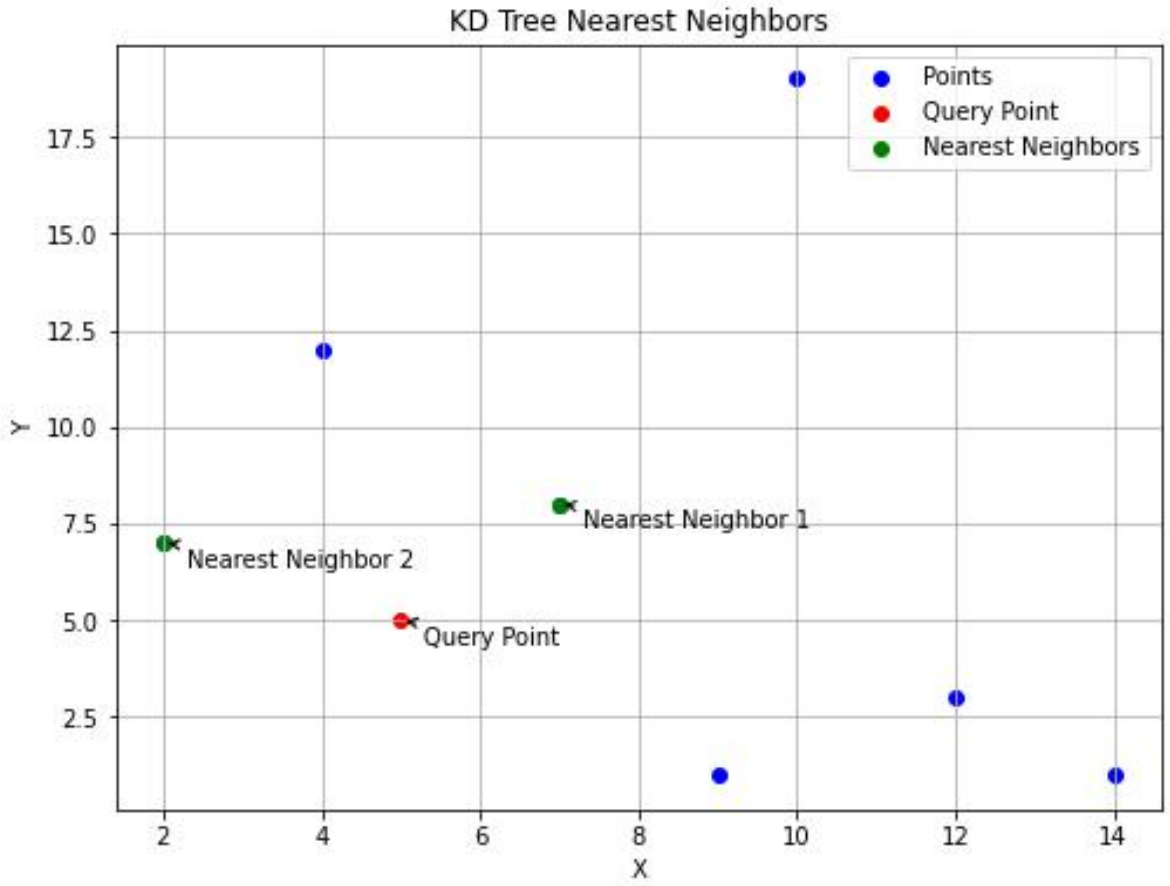




lets say we are searching for the point (14,1).

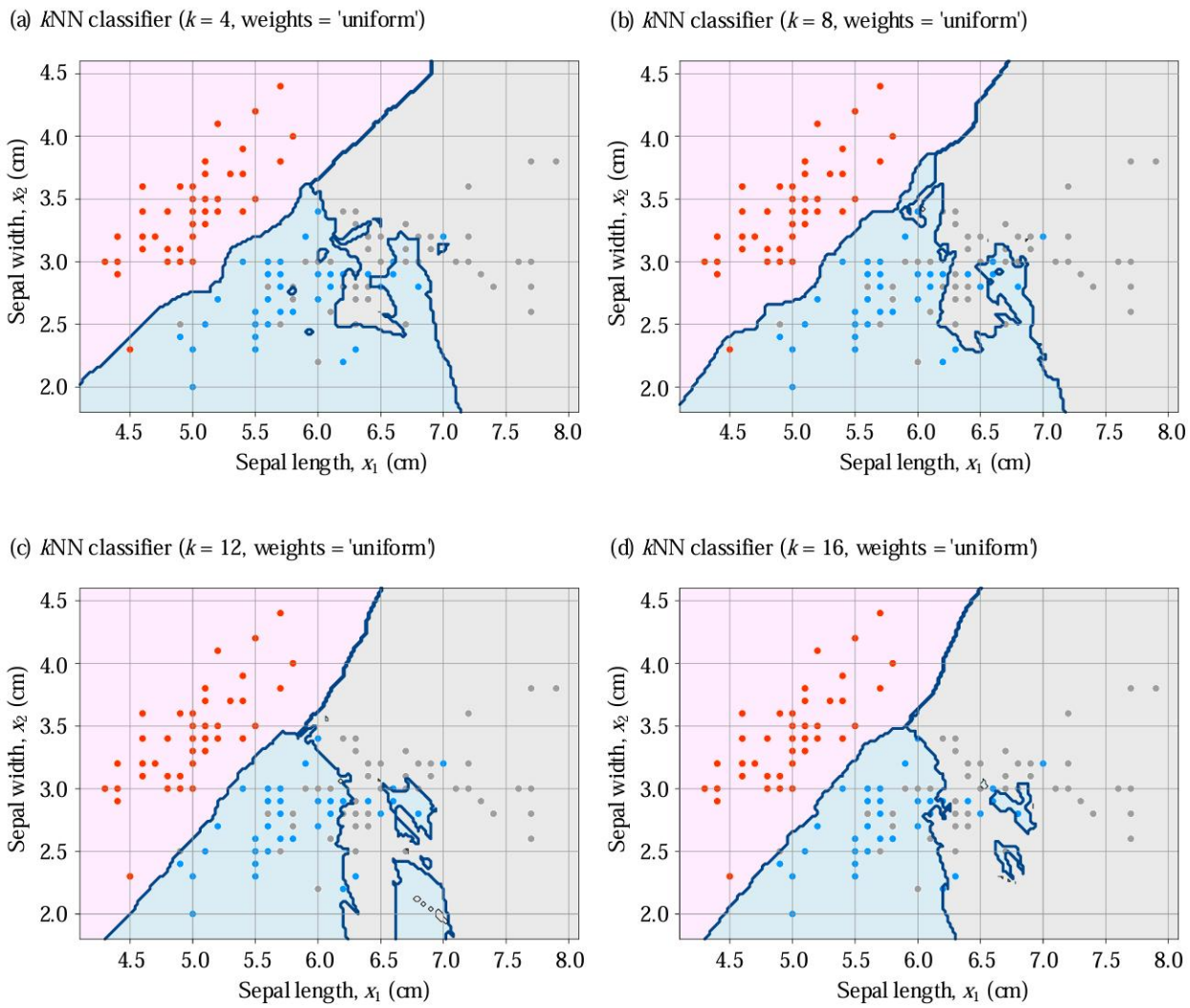
k-nearest neighbors algorithm: Ball Tree

```
import numpy as np
from sklearn.neighbors import BallTree


# Data points
points = np.array([(7, 8), (12, 3), (14, 1), (4, 12), (9, 1), (2, 7), (10, 19)])


# Create a Ball Tree using the data points
ball_tree = BallTree(points, leaf_size=2)


# Query for the nearest neighbors of a new point
new_point = np.array([[5, 5]])  # New point for which we want to find the nearest neighbors
distances, indices = ball_tree.query(new_point, k=2)  # Find the 3 nearest neighbors
```



k-nearest neighbors algorithm

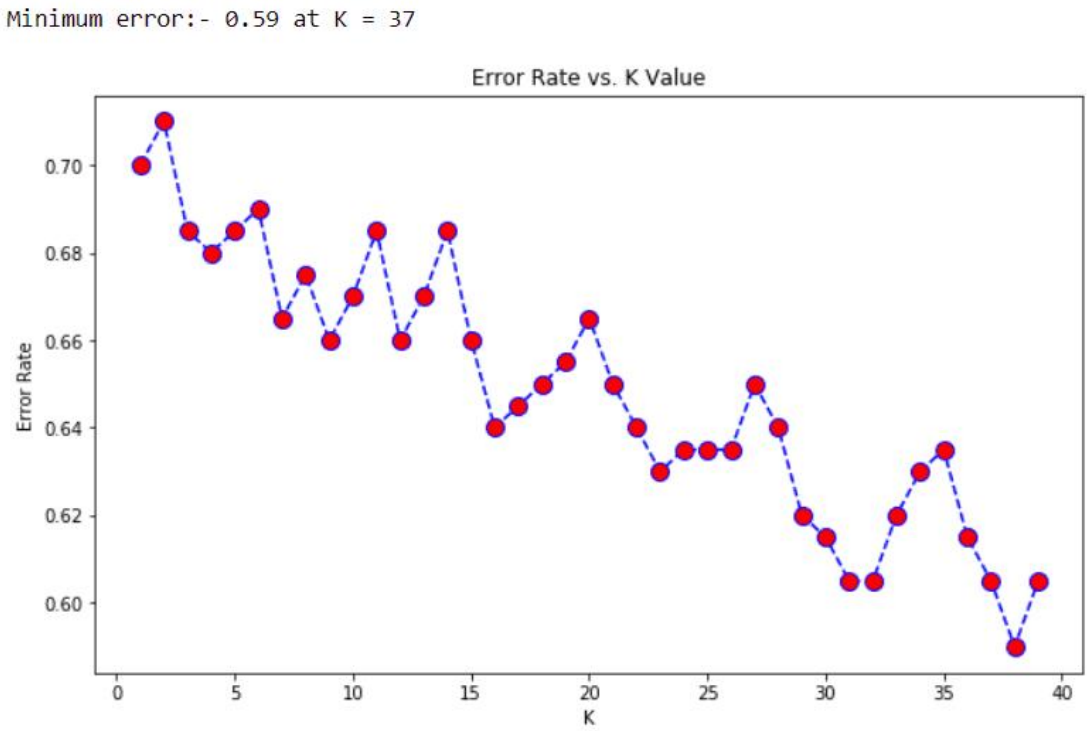
- Although selecting a smaller k value in the k-NN algorithm can accurately capture the classification patterns in the training data, the disadvantage is evident - it is susceptible to noise interference.
- As k gradually increases, the influence of local noisy samples on the boundary diminishes, and the shape of the boundary tends to become smoother.



k-nearest neighbors algorithm

How to select the optimal K value?

1. There are no pre-defined statistical methods to find the most favorable value of K.
2. Initialize a random K value and start computing.
3. Choosing a small value of K leads to unstable decision boundaries.
4. The substantial K value is better for classification as it leads to smoothening the decision boundaries.
5. Derive a plot between error rate and K denoting values in a defined range. Then choose the K value as having a minimum error rate.



k-nearest neighbors algorithm

Advantages:

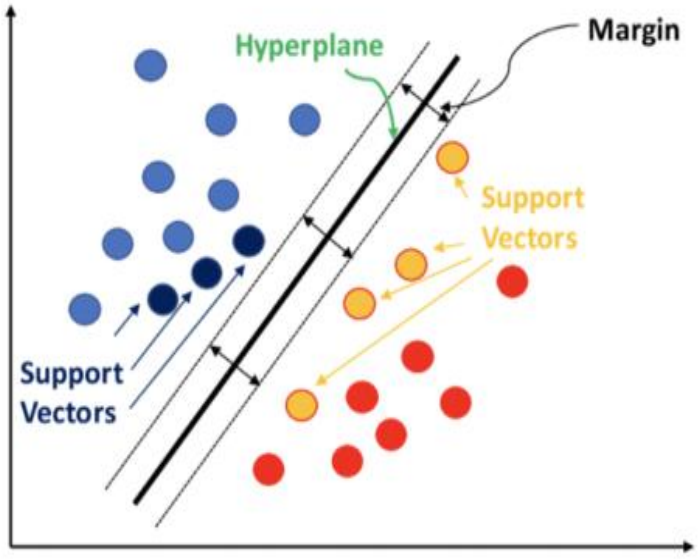
1. **No Training Period**- KNN modeling does not include training period as the data itself is a model which will be the reference for future prediction and because of this it is very time efficient in term of improvising for a random modeling on the available data.
2. **Easy Implementation**- KNN is very easy to implement as the only thing to be calculated is the distance between different points on the basis of data of different features and this distance can easily be calculated using distance formula such as- Euclidian or Manhattan
3. As there is no training period thus new data can be added at any time since it wont affect the model.

Disadvantages:

1. **Does not work well with large dataset** as calculating distances between each data instance would be very costly.
2. **Does not work well with high dimensionality** as this will complicate the distance calculating process to calculate distance for each dimension.
3. **Sensitive to noisy and missing data**
4. **Feature Scaling-** Data in all the dimension should be scaled (normalized and standardized) properly .

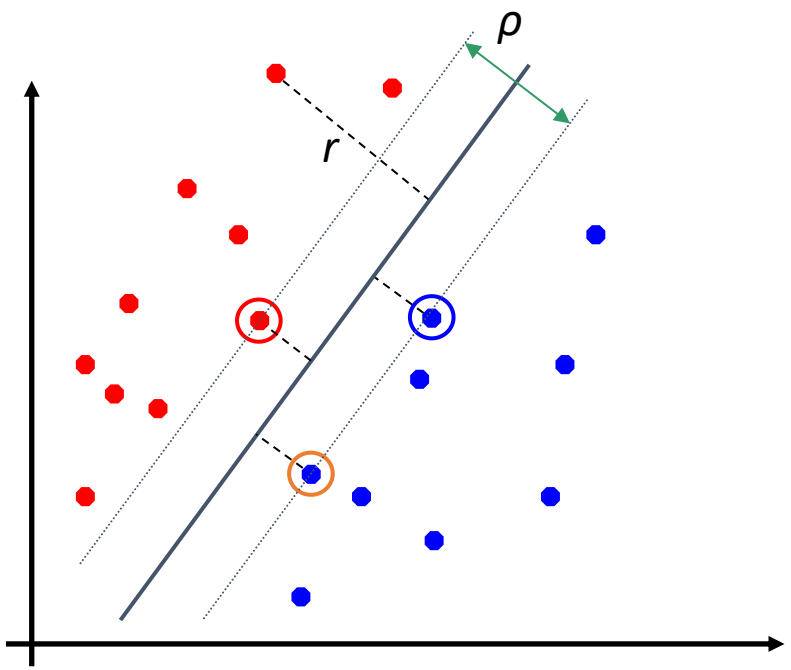
https://www.youtube.com/watch?v=gs9E7E0qOIc

Support Vector Machine (SVM)



SVM

- In 1963, Vanpik from Bell Labs first proposed the theoretical model and methods of Support Vector Machines (SVM).
- In the 1990s, emerging methods such as neural networks faced significant challenges, leading Support Vector Machines to become the mainstream statistical learning model.
- Support Vector Machines had a wide range of applications in early pattern recognition. Fields such as face detection, speech recognition, image classification, character recognition, and text classification all benefited from the application of Support Vector Machines.

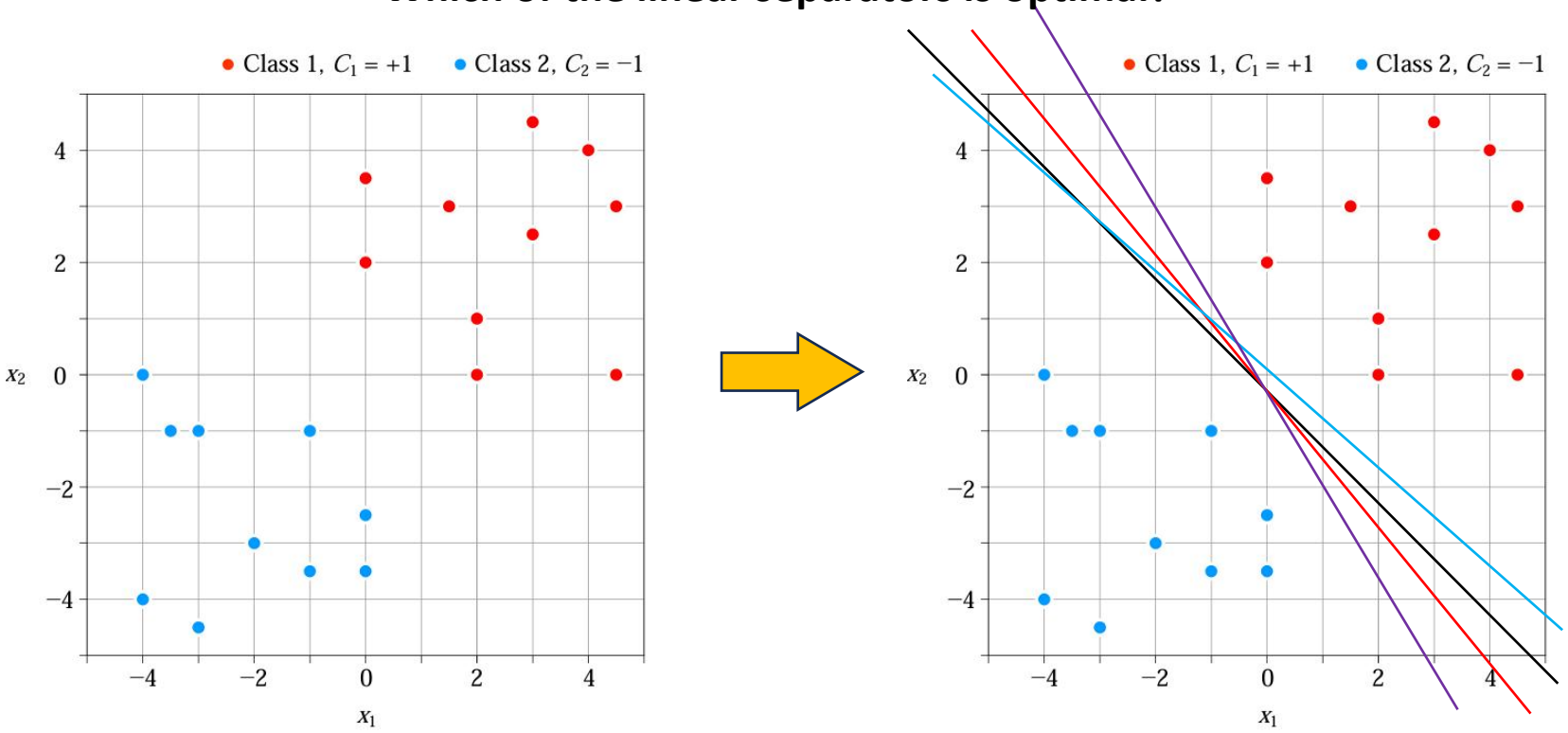


SVM

A Linear Support Vector Machine (Linear SVM) is a type of supervised machine learning algorithm used for classification tasks. SVMs are based on the concept of finding the hyperplane that best divides a dataset into classes.

In a binary classification problem, a hyperplane is a line that divides a feature space into two classes. In two dimensions, this is a straight line, and in higher dimensions, it's a plane.

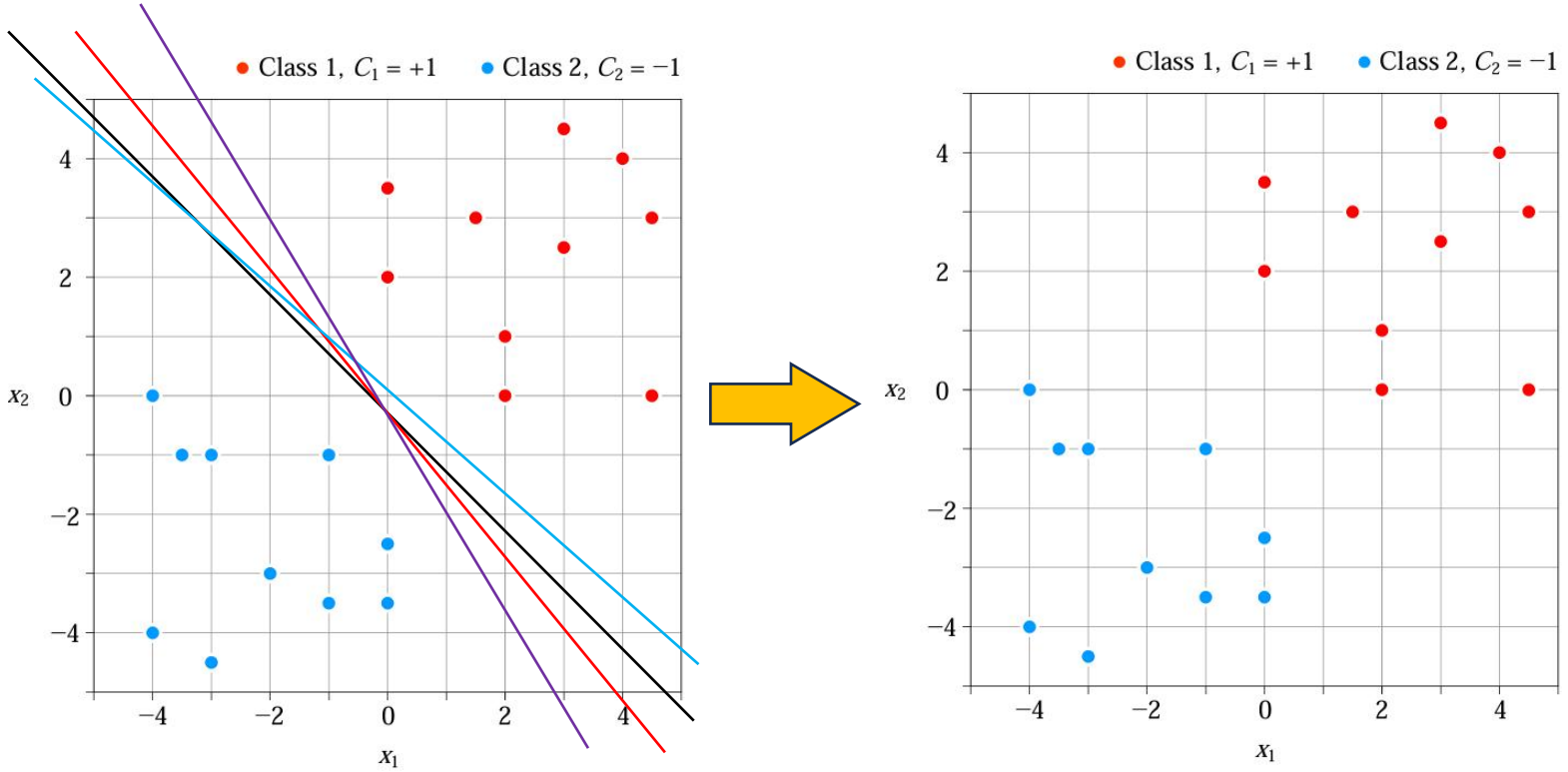
Which of the linear separators is optimal?



Support Vector Machine (SVM)

Support Vector Machine aims to select a separating hyperplane with the maximum margin between instances on both sides from an infinite number of solutions derived from the perceptron algorithm.

When the training data is linearly separable, Support Vector Machine finds the optimal separating hyperplane by maximizing the hard margin. When the training data is approximately linearly separable, Support Vector Machine finds the optimal separating hyperplane by maximizing the soft margin.



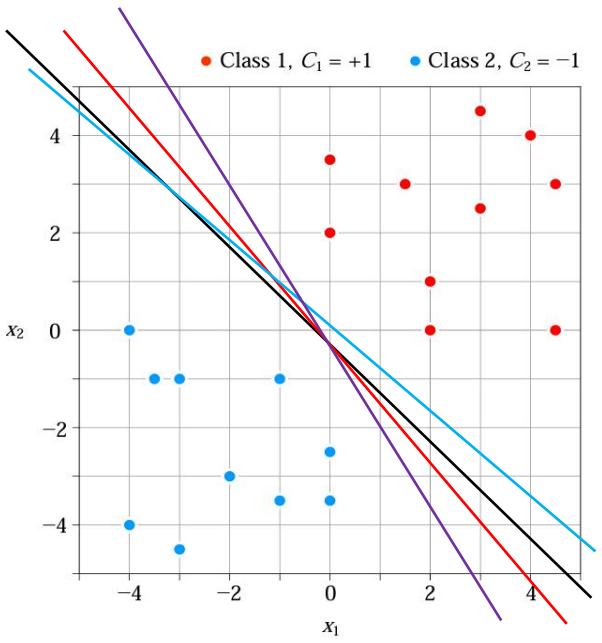
$$w_1 x_1 + w_2 x_2 + b = 0$$

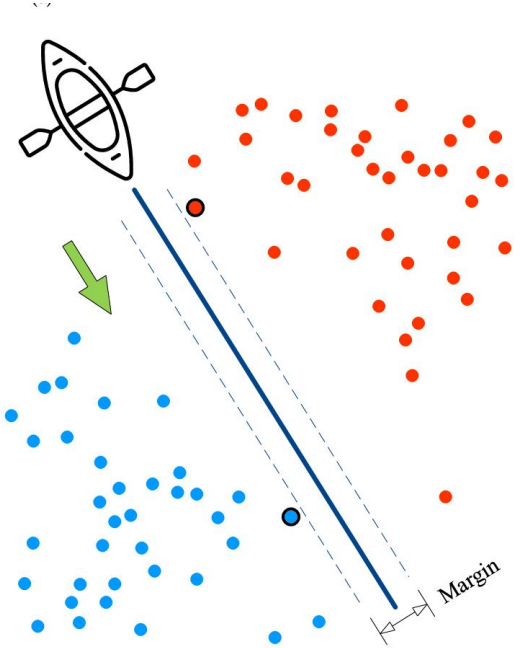
$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

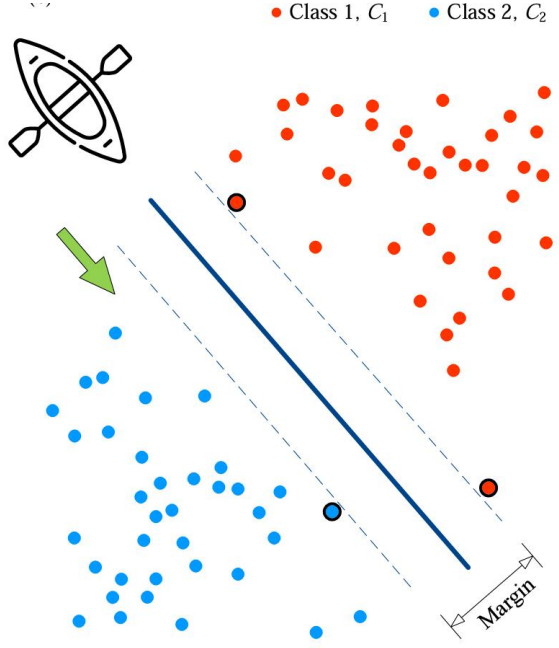
Support Vector Machine (SVM)

Support Vector Machine aims to select a separating hyperplane with the maximum margin between instances on both sides from an infinite number of solutions.

From a data perspective, when two classes of data can be separated by a straight line, this type of data is called linearly separable. Linearly separable problems are addressed using hard margin, which essentially means that there are no data points within the margin.





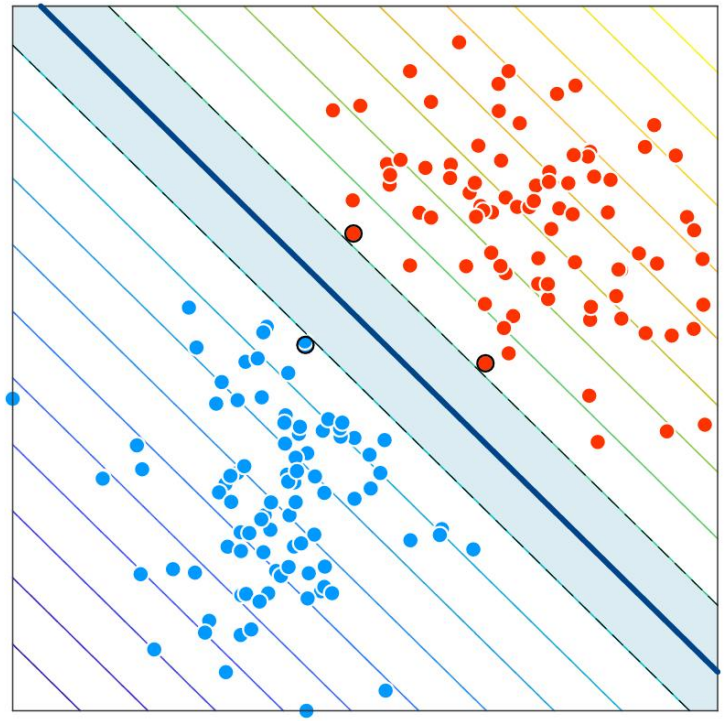


$$w_1 x_1 + w_2 x_2 + b = 0$$

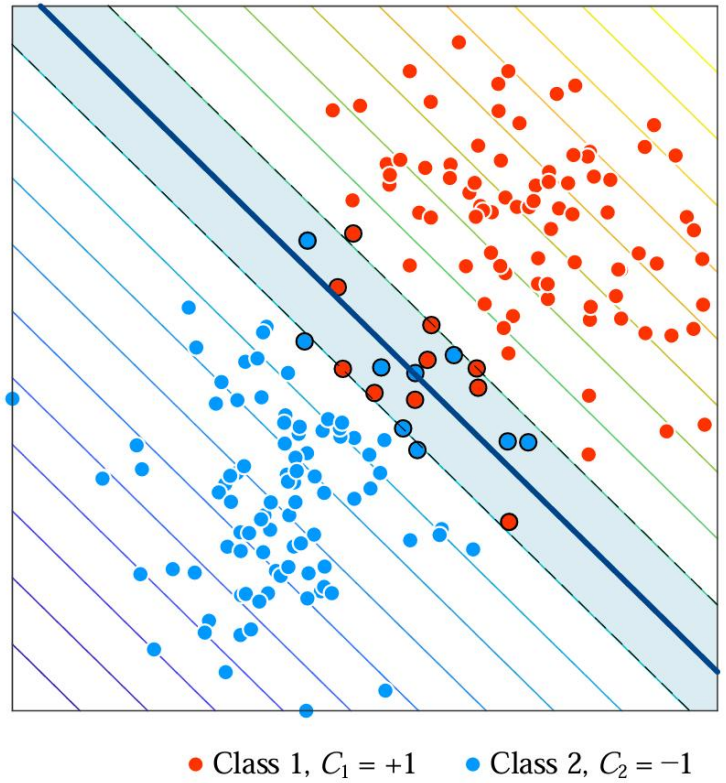
$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{b}{w_2}$$

Support Vector Machine (SVM)

For linearly inseparable problems, two methods are introduced — soft margin and kernel trick.
Soft margin acts as a buffer zone. When a soft margin is present, while separating data with a decision boundary, data points intrude into the margin, and may even extend beyond the margin band.



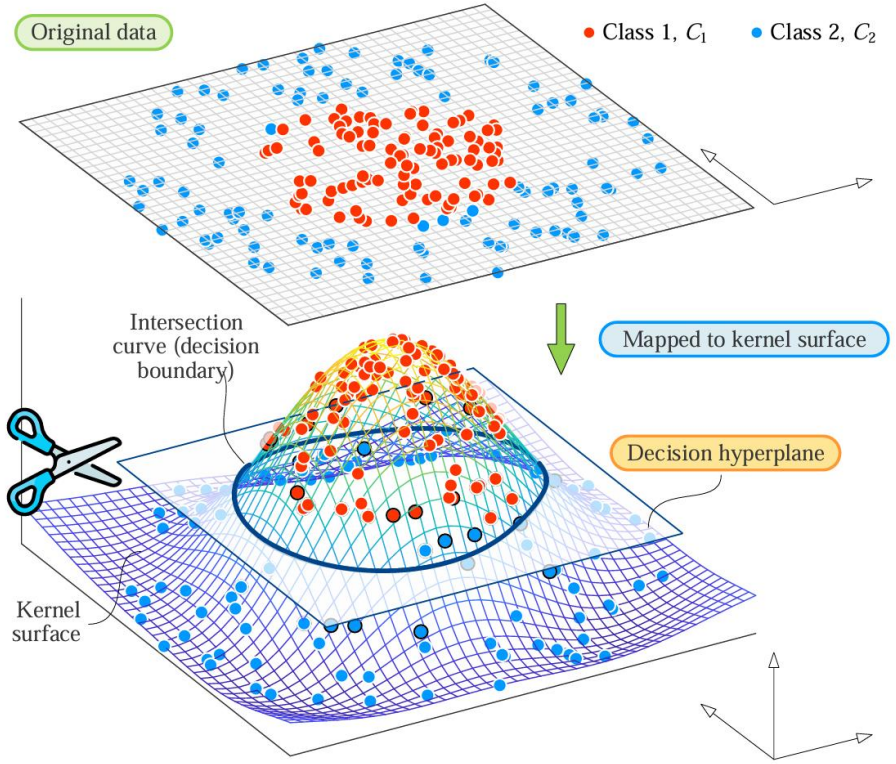
Hard Margin

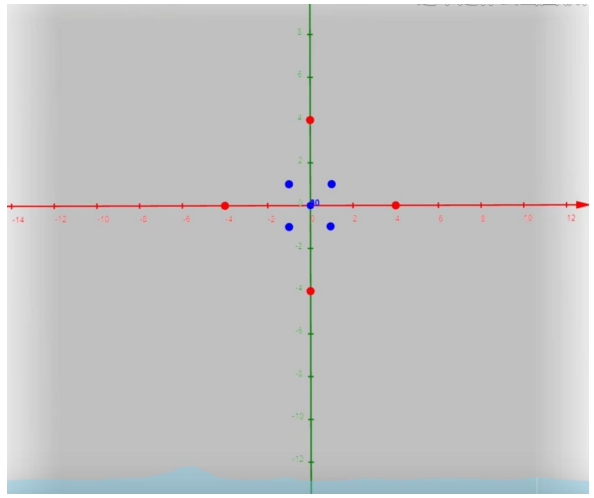


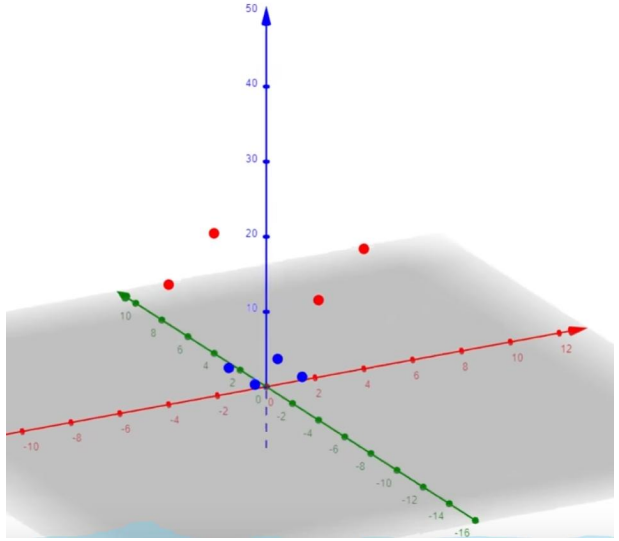
Soft Margin

Support Vector Machine (SVM)

For linearly inseparable problems, two methods are introduced — soft margin and kernel trick.
The kernel trick involves mapping data to a higher-dimensional feature space, which is a form of data dimensionality expansion.



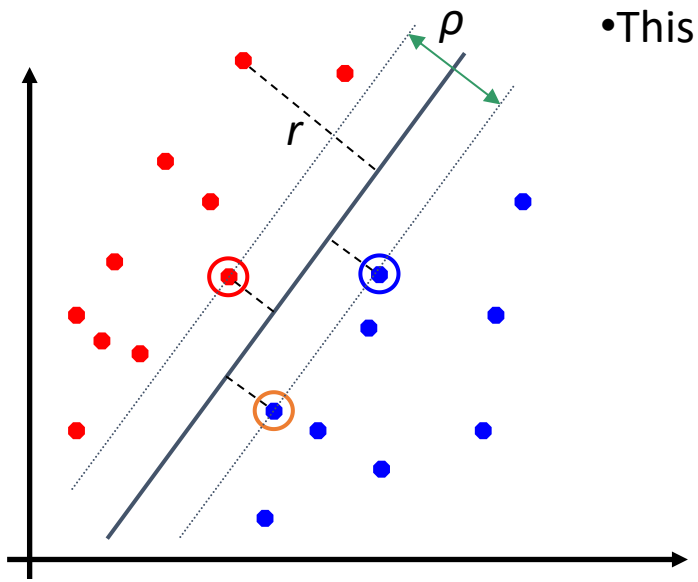




Support Vector Machine (SVM)

What is a Support Vector Machine?

- SVM is a supervised learning algorithm used for:
  - Classification
  - Regression
- Core idea:
  - Find an optimal separating hyperplane
- The hyperplane maximizes the margin between classes
- Only a subset of points affects the solution



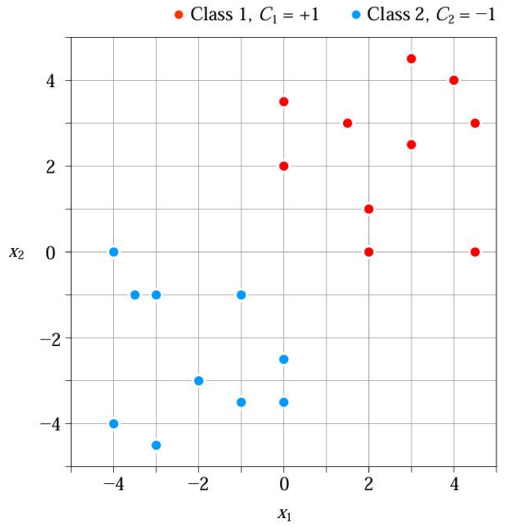
Support Vectors

- Support vectors are:
  - Data points closest to the decision boundary
- They:
  - Define the margin
  - Control the position of the hyperplane
- Removing non-support vectors does not change the model
- This makes SVM memory-efficient and stable

Support Vector Machine (SVM)

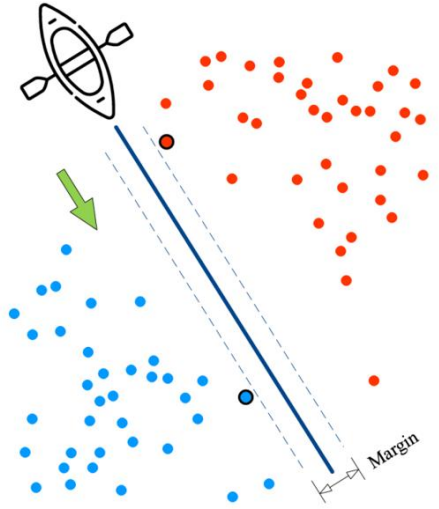
Linearly Separable Data

- Data is linearly separable if:
  - A straight line (or plane) can perfectly separate classes
- No misclassification exists
- Ideal scenario for Hard Margin SVM
- Rare in real-world datasets



Hard Margin SVM: Definition

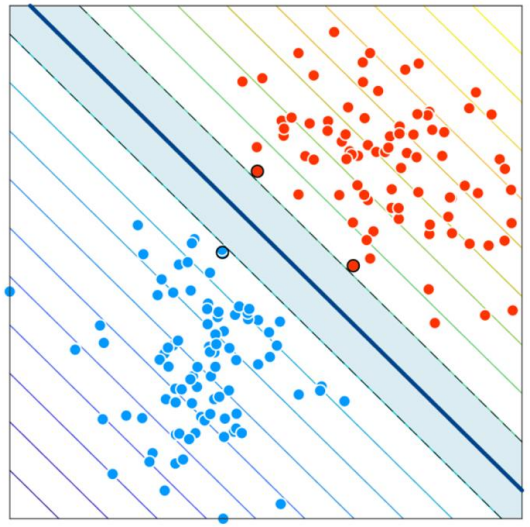
- Hard margin SVM assumes:
  - Data is perfectly linearly separable
- Objective:
  - Maximize margin
  - Zero classification errors
- Strict constraints:
  - All data points must lie outside the margin



Support Vector Machine (SVM)

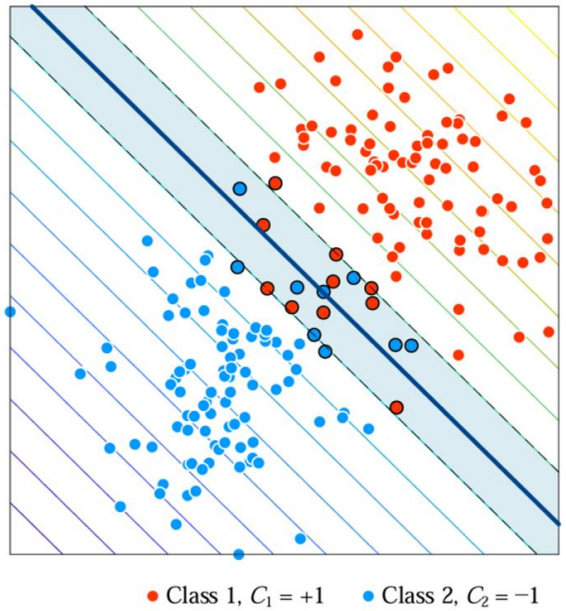
Hard Margin SVM: Characteristics

- Advantages:
  - Clear and mathematically elegant solution
  - Strong theoretical guarantees
- Limitations:
  - Extremely sensitive to noise
  - Fails if even one point is misclassified
- Not suitable for noisy or real-world data



Why Soft Margin SVM?

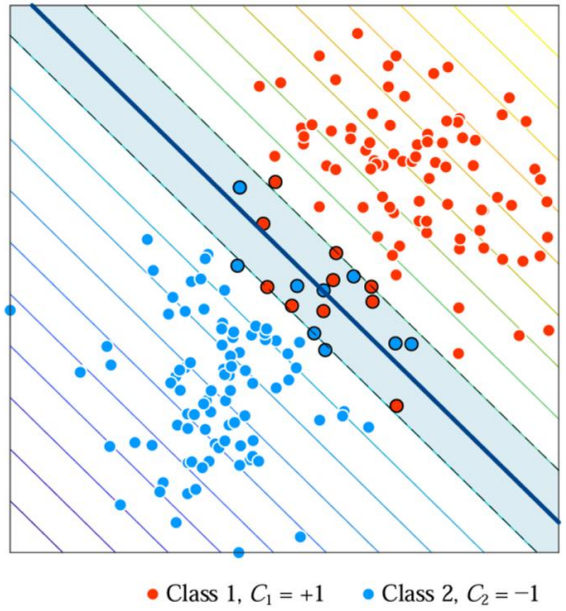
- Real-world data often contains:
  - Noise
  - Overlapping classes
  - Outliers
- Hard margin constraints are too rigid
- Soft margin SVM introduces flexibility



Support Vector Machine (SVM)

Soft Margin SVM: Concept

- Allows:
  - Some data points inside the margin
  - Some misclassifications
- Introduces slack variables
- Objective balances:
  - Margin maximization
  - Classification error minimization



Regularization Parameter (C)

- Parameter C controls trade- off:
  - Large C → fewer misclassifications, smaller margin
  - Small C → larger margin, more tolerance to errors
- High C:
  - Risk of overfitting
- Low C:
  - Better generalization
- Choosing C is critical

Support Vector Machine (SVM)

Motivation for Kernel Trick

- Some data is not linearly separable
- Example:
  - **XOR problem**
- Linear SVM fails in original feature space
- Solution:
  - Transform data into higher dimensions

Definition of XOR

The XOR operation outputs true (1) only when the two inputs are different.

Input A	Input B	XOR Output
0	0	0
0	1	1
1	0	1
1	1	0

Inputs: two binary features x_1 and x_2

Output: class label $y \in \{0, 1\}$

Data points:

- Class 1: (0,1), (1,0)
- Class 0: (0,0), (1,1)

When plotted in 2D feature space:

- Points belonging to class 1 lie diagonally opposite
- Points belonging to class 0 lie on the other diagonal
- No single straight line can separate the two classes

This means XOR is not linearly separable!

Support Vector Machine (SVM)

Motivation for Kernel Trick

- Some data is not linearly separable
- Example:
  - XOR problem
- Linear SVM fails in original feature space
- Solution:
  - Transform data into higher dimensions

Kernel Trick: Core Idea

- Kernel trick:
  - Computes similarity in higher dimensions without explicit transformation
- Avoids computational cost
- Enables SVM to:
  - Learn non-linear decision boundaries
- Common kernels:
  - Linear
  - Polynomial
  - Radial Basis Function (RBF)

Support Vector Machine (SVM)

Summary

SVM is a powerful margin- based classifier

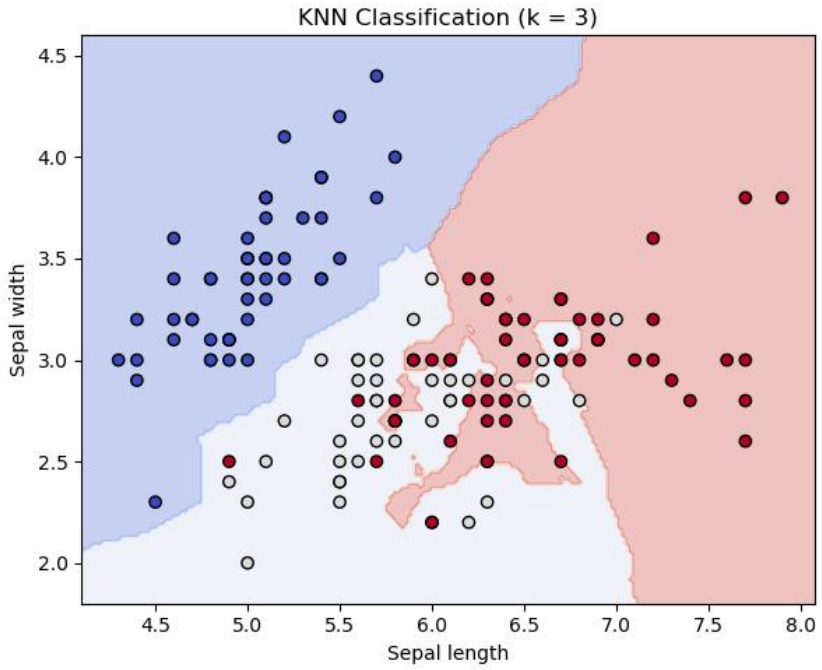
- Hard margin:
  - Works only for clean, separable data
- Soft margin:
  - Handles noise and overlaps
- Kernel trick:
  - Enables non-linear classification
- SVM performance depends on:
  - Kernel choice
  - Parameter C
  - Data quality

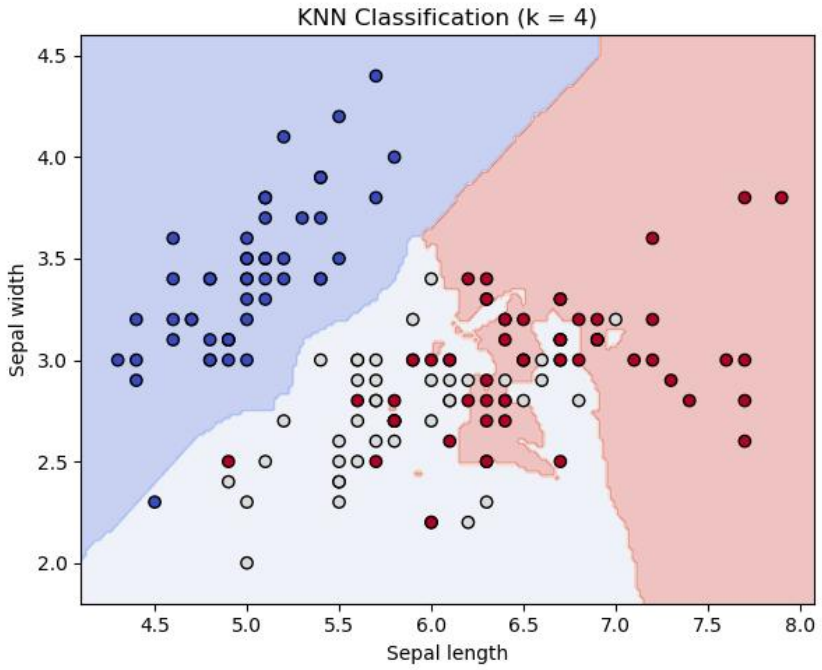
https://www.youtube.com/watch?v=ny1iZ5A8ilA

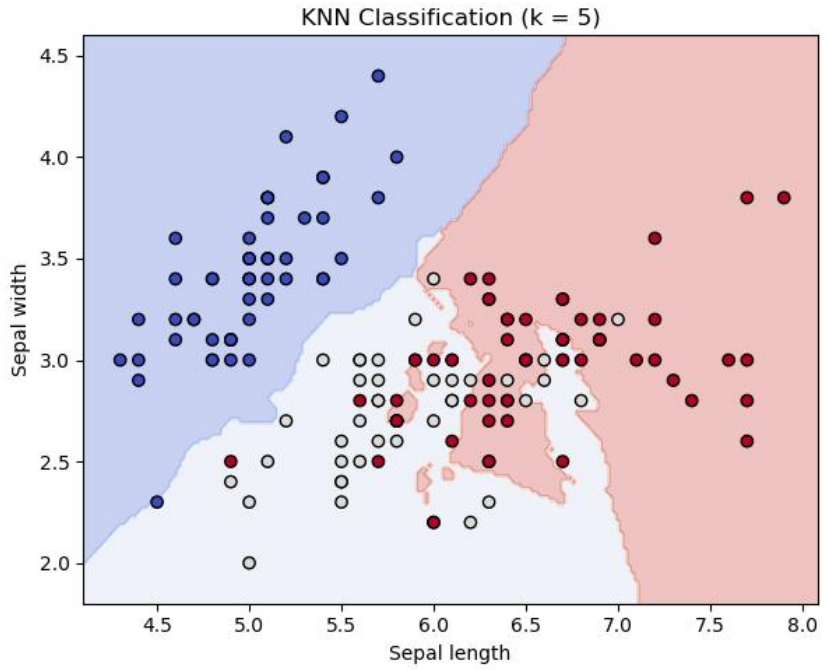
Classwork

Please use the first two features of the iris dataset from scikit-learn to perform KNN classification, where K equals 3, 4, and 5 respectively. Output a visualization scatter plot.

You can sign out at any time after **completing this assignment**.
OR
If you do not complete it, you can sign out **at 5 PM**.







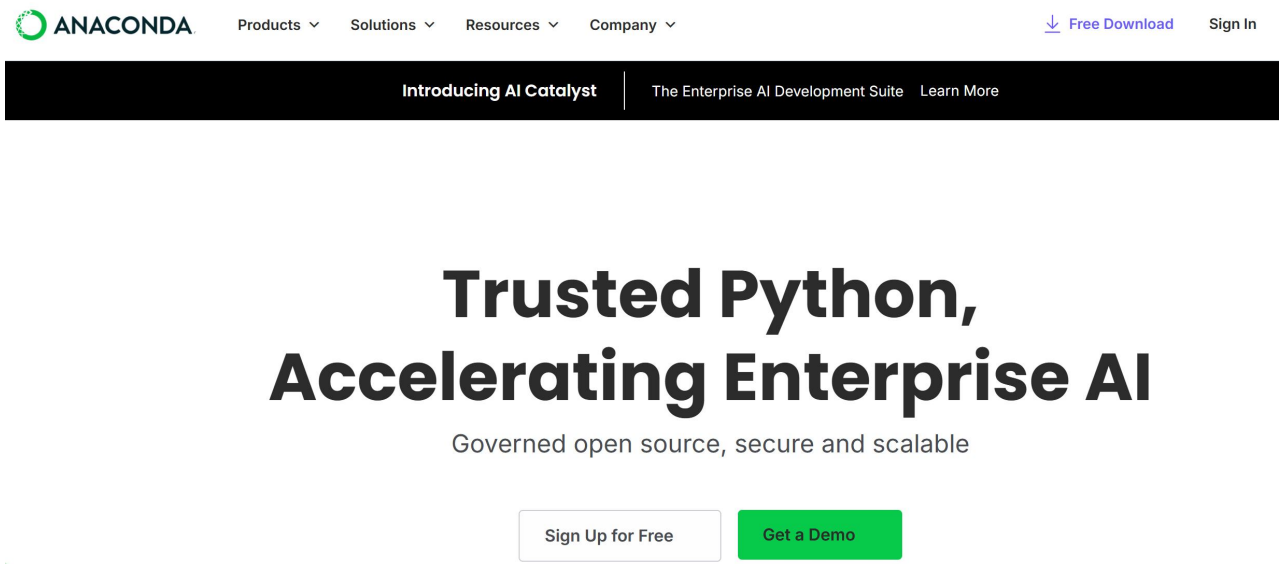
This assignment has nothing to do with the grade!

Run the Logistic Regression of Iris

https://mirror.tuna.tsinghua.edu.cn/help/anaconda/

1. What is Anaconda?
Anaconda is a Python data science distribution, essentially an ‘integrated toolkit’. Its core components include:
- **Python interpreter**: The foundational program for directly executing Python code.
- **Hundreds of pre-installed libraries**: Covering common tools for data processing, numerical computation, visualisation, machine learning, and more.
Conda package manager: for installing, updating, and uninstalling libraries, offering greater capabilities than Python's built-in pip (manages non-Python dependencies); Environment management tool: enables creation of multiple independent virtual environments (e.g., one environment using Python 3.8, another using 3.10), preventing dependency conflicts between different projects. 2. How to install Anaconda
1. Download from the official website
2. Download via mirror sites

Anaconda Install



Run the Logistic Regression of Iris

https://www.jetbrains.com/pycharm/

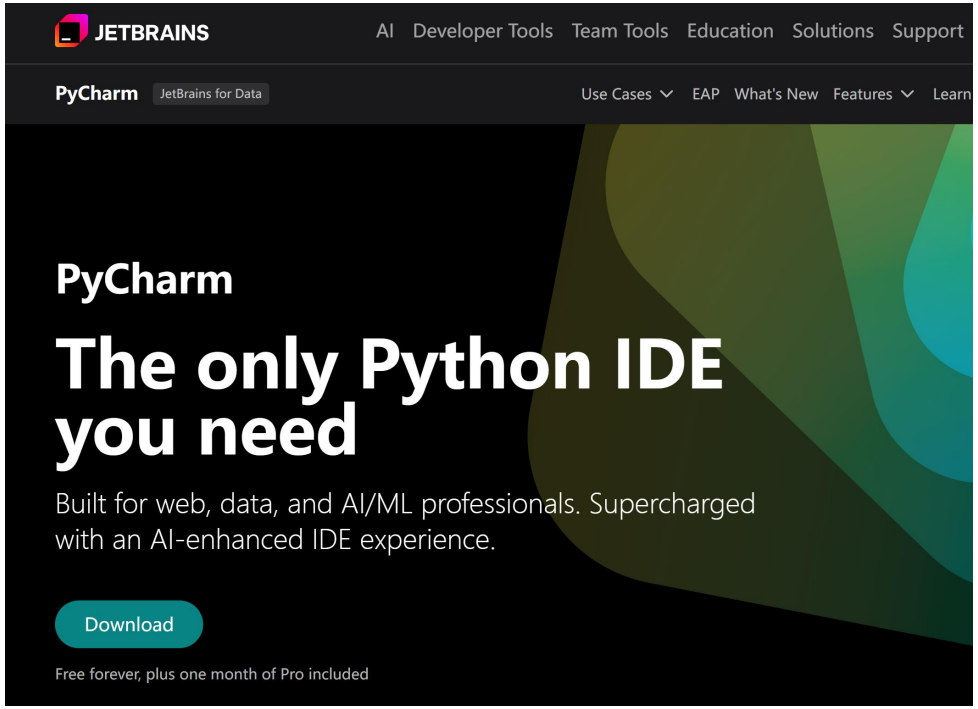
What is PyCharm?

PyCharm is a professional Python integrated development environment (IDE) developed by JetBrains, specifically designed for the Python programming language.
It offers extensive features and tools aimed at enhancing developers' programming efficiency and code quality.

It is widely used across various Python development domains, including web development, data analysis, artificial intelligence, and scientific computing.

It is available in two editions:
Community Edition and **Professional Edition**.

Opt for a **more stable cracked version** during PyCharm installation.



Pycharm Install

Assignment 2 (Grades account for 15%)

Using AI as a tool, design and implement any software/feature/system/webpage/algorithm that interests you. Document the interaction process with AI in detail, compile development documentation, create a PowerPoint presentation, and present it (5-minute presentation, 2-minute Q&A). **No grouping**.

Required submissions:

1. Development documentation (requirements analysis, feasibility analysis, design, implementation, and the entire process of interaction with AI)

2. PPT slides (demonstrating how to use AI as a tool, from conception to design and implementation) **The order of the PPT presentations will be randomly generated.**

3. Declaration of Use of Generative AI (signed)

Detailed information will be released via Canvas soon.

Canvas Submission DDL: 15 April 2026

Some Important Dates

Midterm Test: 14:30- 15:30 23 March (**Q&A** on 13 April)

Final Exam: 19:00 - 22:00 30 April (**Q&A** on 27 April, Last Class)

Schedule Adjustment: 13 April (May be Rescheduled, TBD)

Presentations (3 Classes): 20 April, 27 April, 13 April (Rescheduled, TBD)

Thank you!

Innovating into the Future