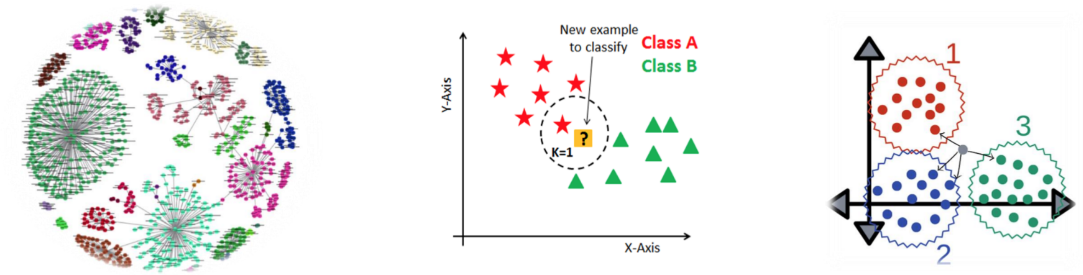


k-Nearest Neighbors (Main)

Tags	Main Lesson
Type	Done
Deliverables	M04W01_Wed_KNN.pdf
Further Reading	[Stat479].pdf
Last edited time	@August 31, 2023 10:59 AM
Lecturer	Dr. Dinh Vinh

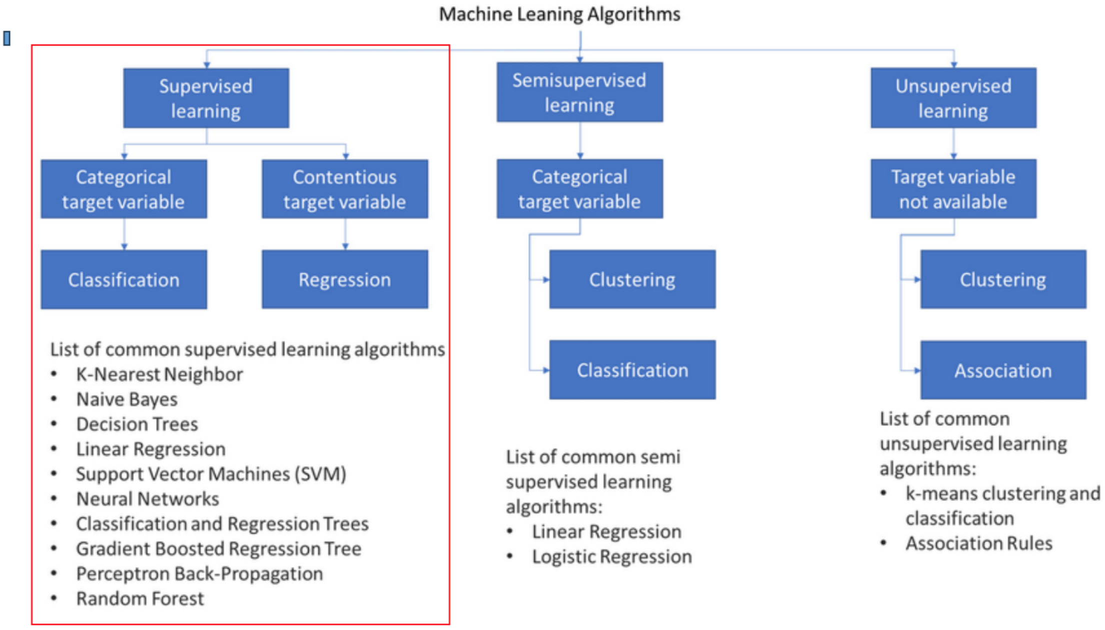
Overview of Machine Learning

Three main types:

- Supervised Learning: know input data and labels
- Unsupervised Learning: only know input data
- Reinforcement Learning

Semi-supervised Learning

Decision Tree → Random Forest → AdaBoost → **XGBoost**


Machine Leaning Algorithms
Supervised learning
Categorical target variable
Contentious target variable
Classification
Regression
List of common supervised learning algorithms
• K-Nearest Neighbor
• Naive Bayes
• Decision Trees
• Linear Regression
• Support Vector Machines (SVM)
• Neural Networks
• Classification and Regression Trees
• Gradient Boosted Regression Tree
• Perceptron Back-Propagation
• Random Forest
Semisupervised learning
Categorical target variable
Clustering
Classification
List of common semi supervised learning algorithms:
• Linear Regression
• Logistic Regression
Unsupervised learning
Target variable not available
Clustering
Association
List of common unsupervised learning algorithms:
• k-means clustering and classification
• Association Rules

Parametric approach (Eager Learning)

Linear regression $y = ax + b$: Mô hình hoá các tham số

KNN Motivation

Looking for approaches that does not make any assumption on underlying data. Concretely:

- Just store the dataset without learning from it
- Start classifying data when it receives test data
- take less time to learn

KNN for Classification

1. **Select $k$**
2. **Calculate the distance**

- Euclidean Distance

Other distances:

- Manhattan distance: $|x_1 - x_2| + |y_1 - y_2|$
- Chebyshev distance: $max(|x_1 - x_2|, |y_1 - y_2|)$
- Minkowski distance (general case):

$$d(i, j) = \sqrt[p]{|x_{i1} - x_{j1}|^p + \cdots + |x_{iq} - x_{jq}|^p}$$

The choice of the distances depends on the properties of the dataset.

3. **Find out $k$ nearest neighbor based on distance metric**
4. **Voting the label and predict an output**

```
def voteTheDistances(array):
  labels = set(array)
  result = ""
  maxOccur = 0
  for label in labels:
    num = array.count(label)
    if(num > maxOccur):
      maxOccur = num
      result = label

  return result
```

How to select $k$

Try for every value of $k$ and select the one that yields the best result.

```
def computeKnearestNeighbor(trainingSet, item, k):
  distances = []
  for dataPoint in trainingSet:
    distances.append(
        {
            "label": dataPoint[-1],
            "value": computeDistance(item, dataPoint)
        }
    )
  distances.sort(key=lambda x: x["value"])
  labels = [item["label"] for item in distances]
  return labels[:k]
```

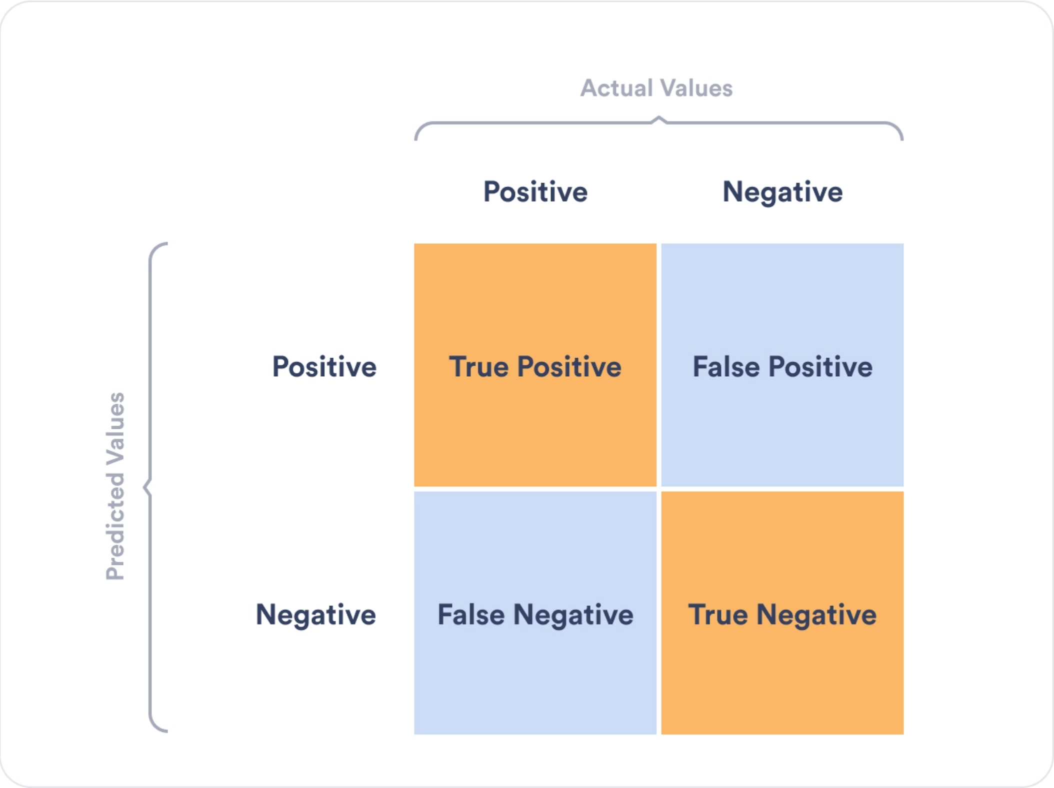
Error Analysis

Metrics:

- Accuracy: $\frac{\#truepredictedsamples}{\#samples}$

For skewed data, accuracy is not a good metric.

Confusion matrix:



		Actual Values	
		Positive	Negative
Predicted Values	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

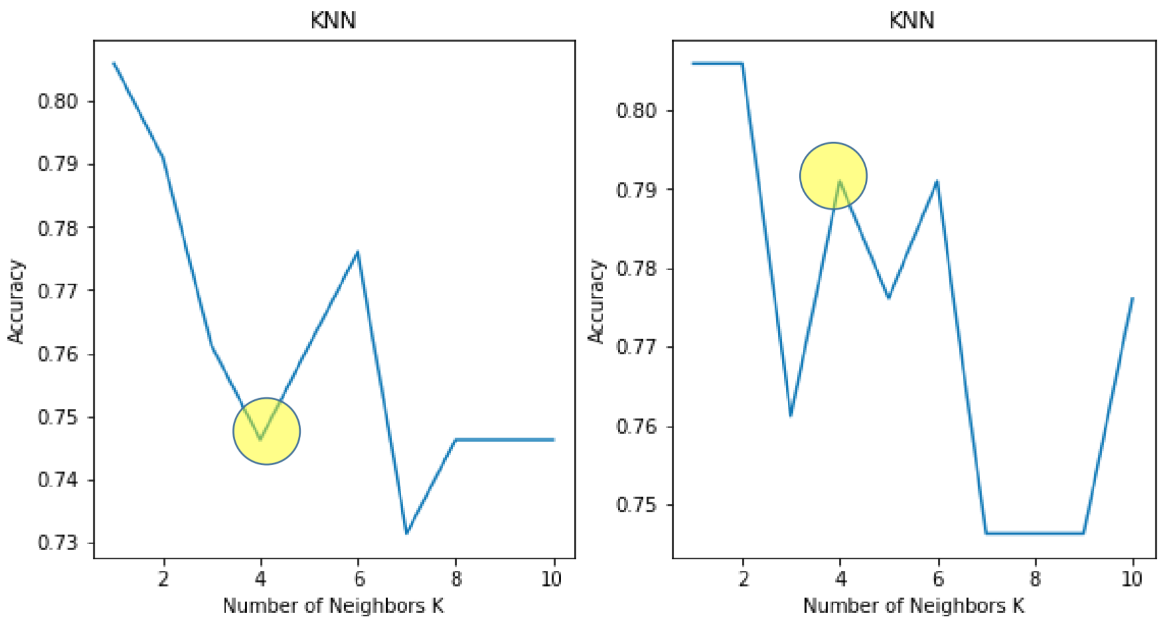
Precision = $\frac{\sum TP}{\sum TP + FP}$

Recall = $\frac{\sum TP}{\sum TP + FN}$

F1 score = $2 \times \dfrac{recall \times precision}{recall + precision}$

```
knn.fit(X_train, y_train)
pred_i = knn.predict(X_test)
```

k-NN with Brute Force



1. Load the data
2. Initialize the value of k
3. Iterate from 1 to N
4. Calculate the distance between test data and each row in the training dataset
5. Sort the calculated distance
6. Get top k rows
7. Get the most frequent class of k rows.

   By convention, we choose an odd $k$.

8. Return the predicted class

Weighted $k$-NN

Đánh trọng số cho từng neighbor

- Uniform weights (default)

  ```
  classifier = KNeighborsClassifier(n_neighbors=8, weights = 'uniform', p = 2)
  classifier.fit(X_train, y_train)
  y_pred = classifier.predict(X_test)
  ```

  where p is the p as in Minkowsky distance.

- Distance weights: $1/distance$

  ```
  classifier = KNeighborsClassifier(n_neighbors=8, p = 2, weights="distance")
  classifier.fit(X_train, y_train)
  y_pred = classifier.predict(X_test)
  ```

- User-defined weights

  ```
  def customizeWeight(distances):
    sigma = 0.5
    return np.exp(-distances**2/sigma)

  classifier = KNeighborsClassifier(n_neighbors=4, p = 2, weights=customizeWeight)
  classifier.fit(X_train, y_train)
  y_pred = classifier.predict(X_test)
  ```

KNN for Regression

1. Choose the value for $k$
2. Calculate the distance

3. Find the $k$ nearest neighbors
4. Calculate the prediction

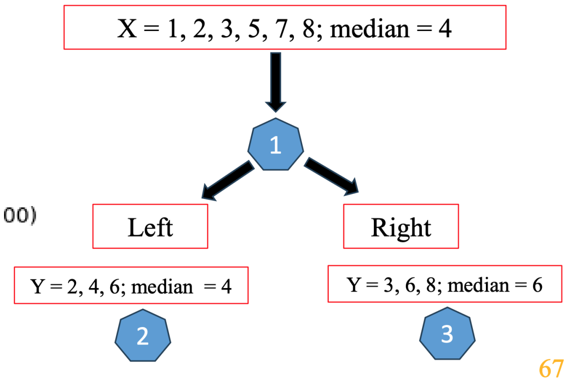
Advanced KNN

Brute Force

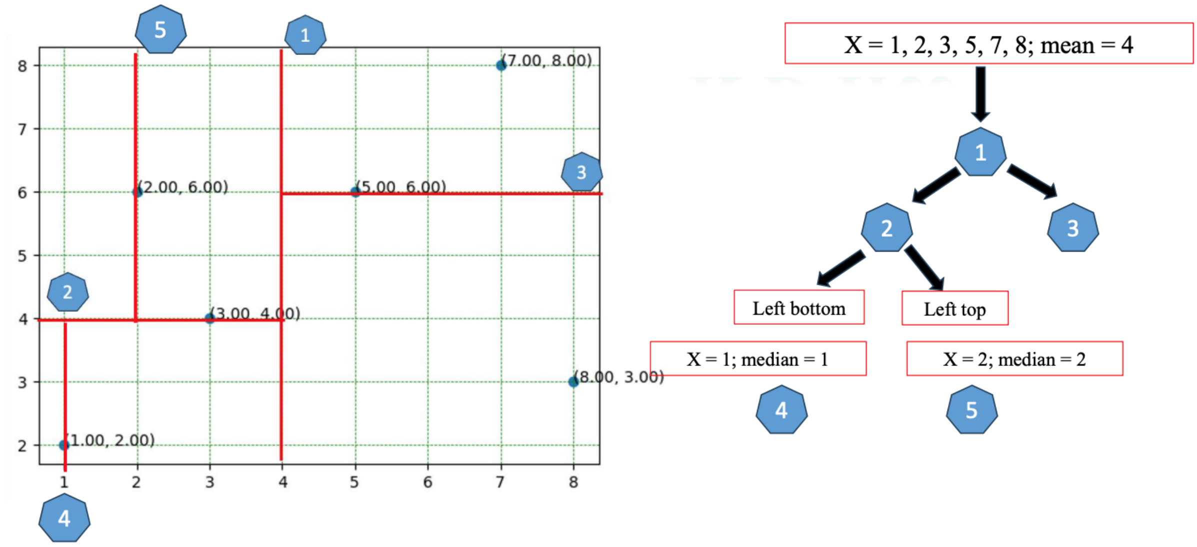
Vét cạn: tính khoảng cách từ data mới đến tất cả các điểm dữ liệu còn lại ⇒ computationally expensive

K-D Tree

`knn.fit()` tìm cách sắp xếp lại dữ liệu sao cho quá trình học nhanh hơn.



1. Pick any one feature at random
2. Find median
3. Split dataset in approximate equal halves
4. Pick next feature and repeat step #2,3
5. Contiue until all data points are partitioned



Ball Tree

1. Take any point
2. Find a point farthest to that point
3. Find a point farthest to this fatherst point
4. Project all points on the line joining farthest points
5. Find the median to divide the space into two halves
6. Find the centroid in each half
7. Draw ball (cirlce of radius equal to the distance to the farthest point in that half)