Machine Learning

Unsupervised Learning

CE 5331 Machine Learning for Civil Engineers

Venki Uddameri, Ph.D. , P.E.

Recap

- What is Machine Learning
- How is it useful for Civil Engineers
- Overview of Machine Learning Methods
- Linear Regression
    - Bivariate
    - Regression interpretation
    - Multivariate
- Logistic Regression
    - Maximum likelihood estimation
    - Regularization (introduction)
- Naïve Bayesian Classifier
    - What is it
    - What makes it naïve
    - Bayes theorem
    - Prior, likelihood and posterior
- K-Nearest Neighbor
    - How does the algorithm work
    - Why is it a lazy learner
    - How to do regression and classification
- Introduction to Decision Trees
    - Fundamentals
    - Information Gain, Entropy and Gini Index
    - ID3 algorithm
    - Classification and Regression Trees (CART)
    - Multi-Adaptive Regression Splines (MARS)

- Ensemble learners
    - Introduction
- Their benefits and drawbacks
- Simple (voting) ensemble learners
- Bagging and Pasting
- Generic bagging classifiers
- Random Forest classifiers
- Bagging Classifier

Python – Introduction
Python – Functions
Python - Pandas
Python – np, scipy, statsmodels
Python – Scikit learn – linear, metrics
Python – Matplotlib, seaborn
Python – Mixed_Naive_Bayes
Python – scikit learn neighbors module
Python – scikit learn ensemble voting
Python – scikit learn bagging classifier
Python – scikit learn RandomForestClassifer

R – Classification and Regression Trees using rpart
R – Drawing trees using rpart.plot
R - Multiadaptive Regression Splines (MARS) using Earth Algorithm

Unsupervised Classifier

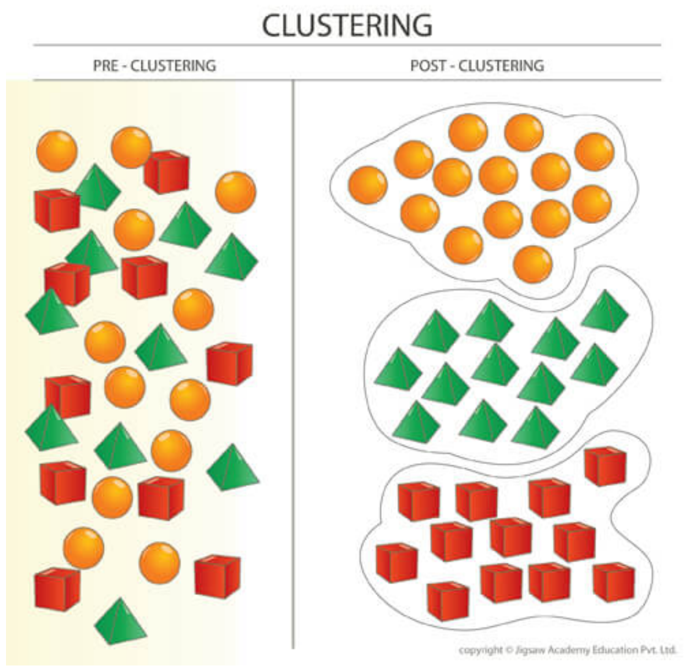
Unsupervised Learners

- Supervised learners map the relationship between inputs and outputs
- In unsupervised learning there are no outputs
  - Referred to as labels in Machine Learning literature
- Unsupervised learning can therefore not be used make predictions but are useful in many other data mining tasks
  - Clustering
  - Dimensionality Reduction
  - Anomaly Detection
  - Semi-Supervised Learning
  - Density Estimation

Unsupervised learning is a vast area but there is still a lot to be explored

Clustering

- The task of identifying instances that are similar and clustering them into groups
- Clustering has a wide range of uses in Civil Engineering
  - Clustering sites with similar characteristics
  - Clustering to study correlated load characteristics
    - Different types of loads on a structure may have same origins
  - Clustering to study freeway operating conditions
  - Clustering of hydrologic datasets
    - Rainfall, water quality



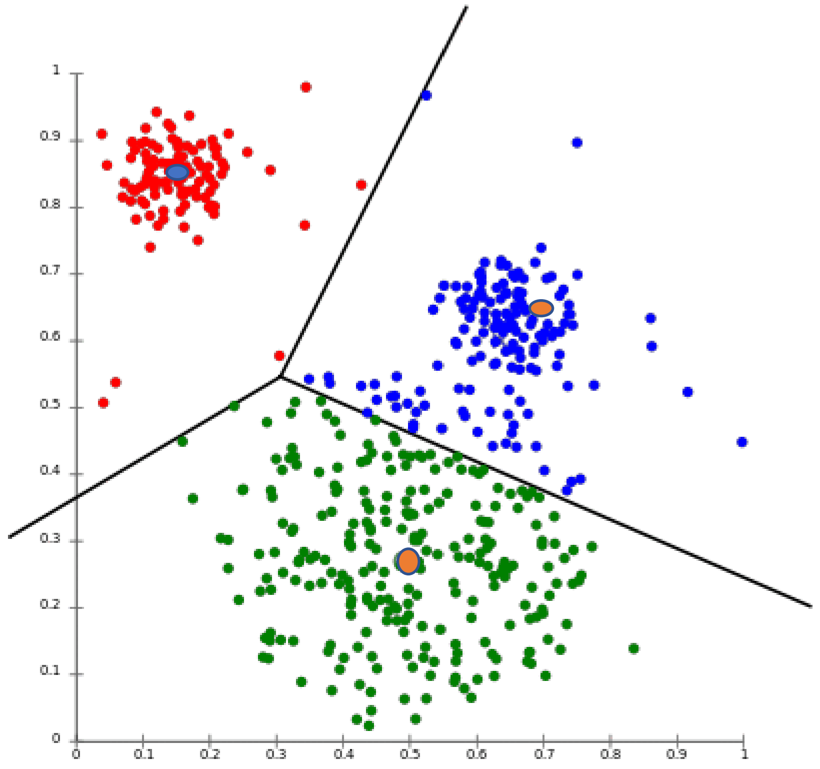
Clustering can reduce data dimensionality and help with semi-supervised classification

Clustering Algorithms

- There are several clustering algorithms and approaches
- Centroid based clustering
  - K-means is a popular algorithm
- Density based clustering
  - Dense areas are connected
  - Useful for outlier detection as outliers are not connected
- Distribution based clustering
  - Gaussian Kernels
- Hierarchical Clustering
  - Trees of clusters

K-Means Clustering

- Proposed by Stuart Lloyd at AT&T Bell Labs in 1957 but published in 1982. A similar algorithm was proposed by Edward Forgy in 1965
    - K-means is also called Lloyd-Forgy algorithm
- Specify the number of clusters in the dataset (K)
- Identify which instance belongs to which cluster



Elements belonging to the cluster are closest to the centroid of the cluster to which it belongs

The cluster centers are as far apart as possible

K-means Clustering Algorithm

- Start by placing the centroids randomly
  - Randomly pick ‘k’ instances and assign them as randomly
- Place each instance in one of the cluster
  - The instance is assigned a cluster based on the proximity to the centroid of the cluster
- Update the cluster centroid
  - Mean of the values of the instances placed within the cluster
- Use the use cluster centroid to reassign labels to each cluster
- Repeat the above two steps till the cluster centroids do not change
  - Change between successive steps is below some tolerence

K-Means Clustering

- Clustering is generally quick for most problems
  - May be slow in rare cases
- The computational speed is linear
  - Function of number of instances, number of clusters and number of dimensions
- Implementation is guaranteed to converge but may not converge to a global optimum
  - Stuck in a local optimum
  - Can be function of initial random guess

Objective Function to Minimize

$$I = \sum_{i=1}^{K} \sum_{j=1}^{N} (x_{i,j} - c_i)^2$$

Start with good initial guesses
Run the algorithm several times and keep the best solution

Inertia

- Inertia is the mean squared distance between each instance and its closest centroid
  - Smaller inertia → better model

Training instance in cluster (i)

Cluster Centroid

$$I = \frac{\sum_{j=1}^{N} (x_{i,j} - c_i)^2}{N}$$

Number of instances

Algorithmic Improvements to K-Means

- Better initialization of centroids
  - Select one centroid at random
  - Select other centroids such that there distances are further from the chosen centroids

Distance of instance (i) from chosen centroid (c_j)

$$F(X) = \frac{D[X^{(i)}]^2}{\sum_{i=1}^{N} D[X^{(i)}]^2}$$

Instances further then c_j will have larger values

Scikit.learn uses this by default

This approach called K-means++ was proposed in

Arthur, David, and Sergei Vassilvitskii. *k-means++: The advantages of careful seeding*. Stanford, 2006.

Accelerated K-Means

- K-means involves many distance calculations
  - Not all of which are really necessary due to triangle inequality
    - The straightline is the shortest distance between two points
  - By knowing the upper and lower bounds of distances between instances and centroids in a cluster many distance calculations can be avoided
  - This approach was first proposed by - Elkan, Charles. "Using the triangle inequality to accelerate k-means." In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pp. 147-153. 2003.

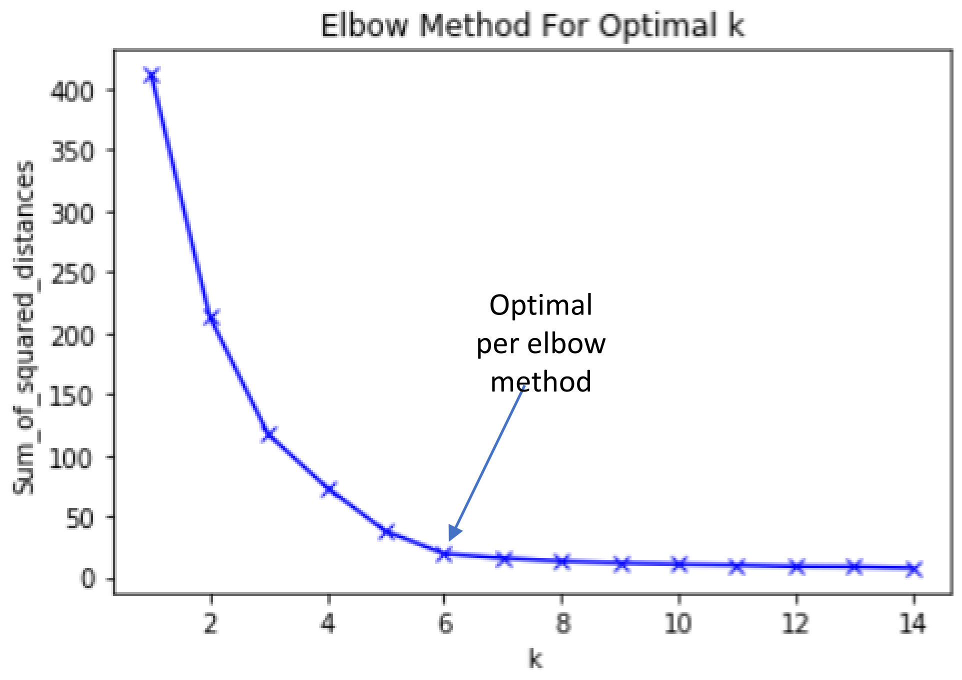
  - Scikit.learn uses this by default

Mini-Batch K-means

- Use of mini-batches of data rather than full dataset at each iteration
  - Centroids move only slightly each iteration
  - Useful when dealing with big data
- Mini-batch training of K-means is faster than conventional approach
  - But the inertia is often worse
- Mini-batch training was proposed in the following paper
  - Sculley, David. "Web-scale k-means clustering." In *Proceedings of the 19th international conference on World wide web*, pp. 1177-1178. 2010.
- Scikit.learn has a function called MiniBatchKmeans

Optimal Number of Clusters

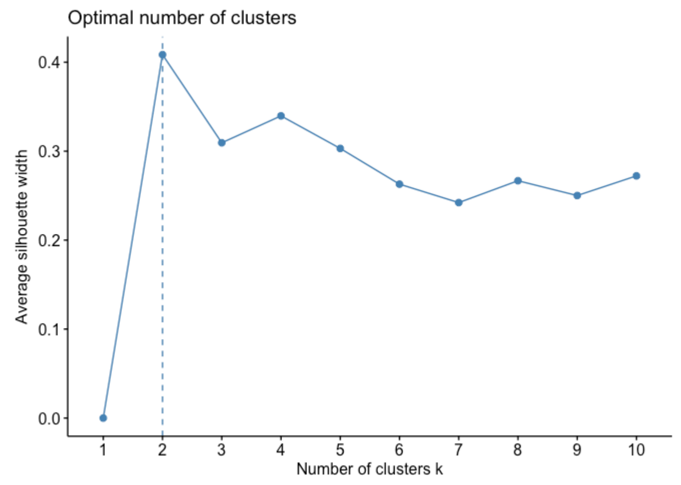
- Inertia is a not a good performance metric to pick the number of clusters
  - Inertia keeps getting lower with increasing number of clusters
  - The error drops quickly initially and slows down
    - Creating an elbow type graph
  - Pick the number of clusters – where the drop becomes gradual



Scikit-learn computes negative inertia using the ‘score’ metric

Optimal Number of Clusters

- The elbow method is a rather crude way to identify the number of clusters
- The *silhouette coefficient* is one rigorous approach
  - Albeit computationally expensive
- Scikit-learn has a silhouette score function in the sklearn.metrics module



Mean distance to instances in the next closest cluster

Mean distance to other instances in the same cluster

$$ss = \frac{(b - a)}{max(a, b)}$$

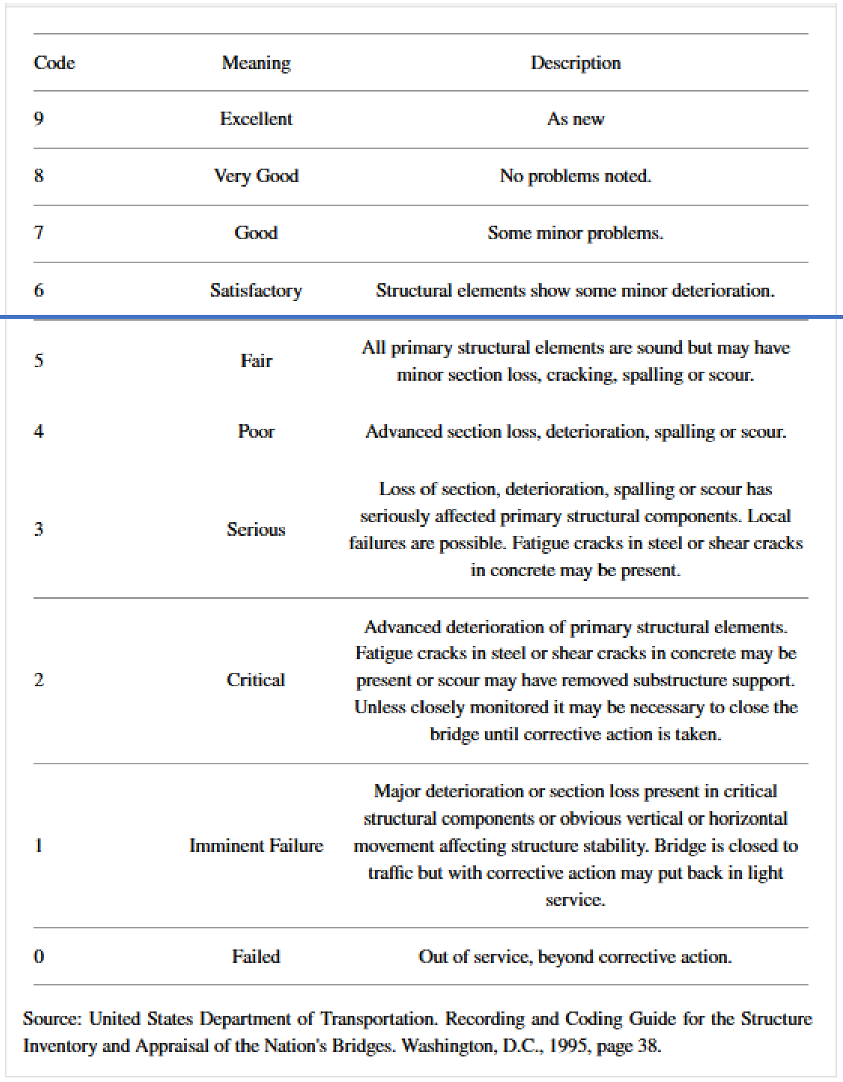
silhouette coefficient varies between -1 and +1 (+1 it is in its own cluster; 0 it is in the cluster boundary and -1 it is in the wrong cluster)

Illustrative Example

- Predicting damage to culverts in Texas
- Use the ADT and SVCYR to cluster the data into 5 clusters
- As an exercise find the optimal number of clusters

Satisfactory

Unsatisfactory



Code	Meaning	Description
9	Excellent	As new
8	Very Good	No problems noted.
7	Good	Some minor problems.
6	Satisfactory	Structural elements show some minor deterioration.
5	Fair	All primary structural elements are sound but may have minor section loss, cracking, spalling or scour.
4	Poor	Advanced section loss, deterioration, spalling or scour.
3	Serious	Loss of section, deterioration, spalling or scour has seriously affected primary structural components. Local failures are possible. Fatigue cracks in steel or shear cracks in concrete may be present.
2	Critical	Advanced deterioration of primary structural elements. Fatigue cracks in steel or shear cracks in concrete may be present or scour may have removed substructure support. Unless closely monitored it may be necessary to close the bridge until corrective action is taken.
1	Imminent Failure	Major deterioration or section loss present in critical structural components or obvious vertical or horizontal movement affecting structure stability. Bridge is closed to traffic but with corrective action may put back in light service.
0	Failed	Out of service, beyond corrective action.

Source: United States Department of Transportation. Recording and Coding Guide for the Structure Inventory and Appraisal of the Nation's Bridges. Washington, D.C., 1995, page 38.

Implementation

```
# Step 1: Import Libraries
import os
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.cluster import MiniBatchKMeans
from sklearn.metrics import silhouette_score
import seaborn as sns
from matplotlib import pyplot as plt
```

```
# Step 2: Ccange working directory
dir = 'D:\\Dropbox\\000CE5333Machine Learning\\Week8-
UnsupervisedMethods\\Code'
os.chdir(dir)

# Step 3: Read the dataset and add a variable
a = pd.read_csv('TXculvertdata.csv') # read our dataset
features = ['SVCYR','ADT'] # INPUT DATA FEATURES
X = a[features] # DATAFRAME OF INPUT FEATURES
scaler = StandardScaler() # Initialize standardscaler
X_scl = scaler.fit_transform(X)

# Step 4: Split into training and testing data
X_train,X_test = train_test_split(X_scl,test_size=0.30,
                                  random_state=10)
```

Implementation

```
#Step 5: Make a run with an arbitrary K = 5 clusters
k = 5
kmeans5 = KMeans(n_clusters=k) # Instantiate an object
y_pred = kmeans5.fit_predict(X_train) # Fit the K-means model
kmean5_cent = kmeans5.cluster_centers_ # Get centroids
y_tst = kmeans5.predict(X_test)
kmeans5.inertia_  # Compute inertia
kmeans5.score(X_train) # Score is the reciprocal of inertia
silhouette_score(X_train,kmeans5.labels_)
```

```
# Step 6: Perform Minibatch processing
minibatchK5 = MiniBatchKMeans(n_clusters=k) # initialize
y_mini = minibatchK5.fit_predict(X_train)
mini_cent = minibatchK5.cluster_centers_
y_tst = minibatchK5.predict(X_test)
minibatchK5.inertia_  # Compute inertia
minibatchK5.score(X_train) # Score is the reciprocal of inertia
silhouette_score(X_train,minibatchK5.labels_)
```

You should Know

- What is unsupervised classification
- What are its uses
- What is K-means algorithm
- What are its hyperparameters
- How does the K-means algorithm work
    - What are its limitations
- What are some enhancements made to overcome its limitations
    - Minibatch processing
    - Good initial conditions
- How do we find the optimal number of clusters
    - Elbow method
    - Silhouette Score