KNN from scratch

Let x_i be a point in the test set, y be list of points in the train set

pseudo-code:

- Find distance between x_i and y
- Find k closest points in y to x_i
- The most frequent labels of y is the label for x_i

Minkowski distance

$$d(x, y) = \left(\sum_{i=1}^{n} |x_i - y_i|^c \right)^{\frac{1}{c}}$$

$c = 1$: Manhattan

$c = 2$: Euclidean

Functions

In [82]:
```
import numpy as np
import pandas as pd
```

In [23]:
```
# Minkowski distance

#Input: vector x,y,c
#Output: a float
#Test: Input=([1,3],[1,2],1) Output=sqrt(5)
def Minkowski(x,y,c):
    distance=0
    for x_i,y_i in zip(x,y):
        distance+=np.abs(x_i-y_i)**c
    return distance**(1/c)
```

In [26]:
```
# Find distance of one point to all the other points

#Input: a point, list of other points, order of Minkowski distance
#Ouput: list of distance
#Test: Input=([1,1],([3,2],[3,3]),2). Output=[sqrt(5),sqrt(8)]

def distance_one_to_others(x,others,c):
    distance_list=[]
    for y_i in others:
        distance_list.append(Minkowski(x,y_i,c))
    return distance_list
```

In [60]:
```
#Find k nearest neighbors

#Input: list of number, k neighbors
#Output: index of the k shortest distance
#Test: Input=([1,2.5,0.5],2), Output=[2,0]
def k_nearest(distance_list,k):
    KNN_locations=[]
    ordered_dist=sorted(zip(range(len(distance_list)),distance_list)
                        ,key=lambda x: x[1])
    knn_indices=[i[0] for i in ordered_dist[:k]]
    return knn_indices
```

In [62]:
```
#Get the list of labels of knn of x in previous step. Get the majority v
ote

#Input: train labels, KNN indices
#Output: most frequent labels of the KNN indices

#Test: Input=([0,1,1,1,0,0],[2,0,3]), Output =1

def label_prediction(knn_indices,labels):
    knn_labels=pd.Series(np.array(labels)[knn_indices])
    predicted_label=knn_labels.mode().values[0]
    return predicted_label
```

In [65]:
```
#Predict the test labels based on all the function we have created

#Input: X_train, X_test, y_train, k, c
#Output: List of labels corresponding to each coordinate of X_test

#Test: Input=[[(1,3),(1,5),(3,4),(4,4)],
#             [(2,4)],
#             ['Apple','Apple','Orange','Orange']
#             k=3, c=2)
# Output = Apple

def knn_predict(X_train, X_test, y_train, k, c):
    prediction=[]
    for i in X_test:
        distance_list=distance_one_to_others(i,X_train,c)
        knn_indices=k_nearest(distance_list,k)
        predicted_label=label_prediction(knn_indices,y_train)
        prediction.append(predicted_label)
    return prediction

#knn_predict([(1,3),(1,5),(3,4),(4,4)],[(2,4)],['Apple','Apple','Orang
e','Orange'],k=3, c=2)
```

Test on Iris set

In [80]:
```
# Import the necessary functions
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = load_iris()
data = iris.data
target = iris.target
X_train, X_test, y_train, y_test = train_test_split(data, target,random_
state=3)

prediction=knn_predict(X_train, X_test, y_train, k=1, c=2)
print("Testing Accuracy: {}".format(accuracy_score(y_test, preds)))
```

```
Testing Accuracy: 0.9473684210526315
```