Machine Learning and Probability

Jacob Breckenridge

October 2018

1 Introduction

1.1 Machine Learning Algorithms

1. K-Nearest Neighbor

   Looks at the K nearest point around a point $\overline{\mathbf{x}}$. Determines a likeliest probability for classification based on the probability of each classification of picking one class from the set of k points chosen.[2]

$$p(y = c|x, \mathcal{D}, \mathcal{K}) = \frac{1}{K} \cdot \sum_{i \in N_K(\overline{x}, \mathcal{D})}$$

   KNN classifiers do not perform well in high dimensionality.

   To determine the K-nearest neighbors of each point on the graph, I used a QUICKSORT algorithm .[1] Then, returned the 10 points closest to whatever given point on the graph. QUICKSORT uses the divide and conquer method of reducing time. It takes a *pivot* value in the set, then splits the set into two parts (less than the pivot and greater than the pivot). It iteratively repeats this split until the entire set has been sorted through.

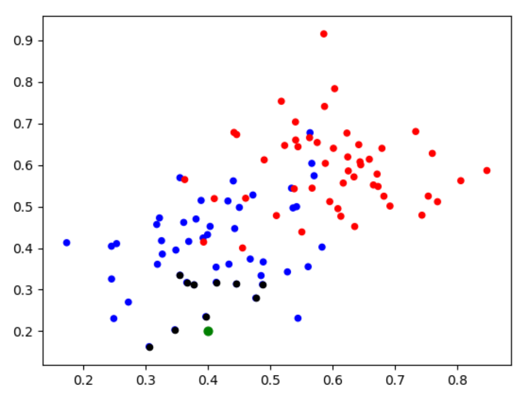

Figure 1: Green is a Random Point, Black is the 10-Nearest Neighbors. In this example, all of the points are blue.

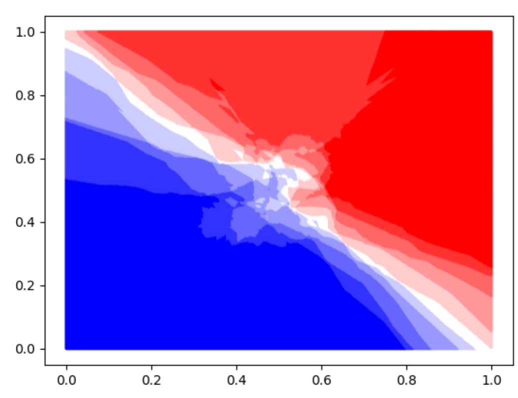

Figure 2: Estimate by Color Saturation. Where the sets tend to overlap, there is less certainty. White is 50-50 probability.

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from random import randint

np.random.seed(1080)

#function to select k nearest
def klist(x1,y1,data):
    sort_list = [[0]*100 for _ in range(4)]
    #sort the array based on distance from x1,y1
    #sorted using quicksort algorithm
    for i in range(100):
        sort_list[0][i] = data[0][i]
        sort_list[1][i] = data[1][i]
        sort_list[2][i] = distance(x1,y1,data[0][i],data[1][i])
        sort_list[3][i] = i
    quicksort(sort_list[3],sort_list[0],sort_list[1],sort_list[2],0,99)
    KNN = [[0]*10 for _ in range(3)]
    for i in range(10):
        KNN[0][i] = sort_list[0][i]
        KNN[1][i] = sort_list[1][i]
        KNN[2][i] = sort_list[3][i]
    return KNN

def quicksort(Ac,A0,A1,A2,p,r):
    if p<r:
        q = partition(Ac,A0,A1,A2,p,r)
        quicksort(Ac,A0,A1,A2,p,q-1)
        quicksort(Ac,A0,A1,A2,q+1,r)

def partition(Ac,A0,A1,A2,p,r):
    x = A2[r]
    i = p-1
    for j in range(p,r):
        if A2[j]<=x:
            i = i+1
            tempA2 = A2[j]
            A2[j] = A2[i]
            A2[i] = tempA2
            tempA0 = A0[j]
            A0[j] = A0[i]
```

```
                A0[i] = tempA0
                tempA1 = A1[j]
                A1[j] = A1[i]
                A1[i] = tempA1
                tempAc = Ac[j]
                Ac[j] = Ac[i]
                Ac[i] = tempAc
        temp2 = A2[r]
        A2[r] = A2[i+1]
        A2[i+1]= temp2
        temp0 = A0[r]
        A0[r] = A0[i+1]
        A0[i+1]= temp0
        temp1 = A1[r]
        A1[r] = A1[i+1]
        A1[i+1]= temp1
        tempc = Ac[r]
        Ac[r] = Ac[i+1]
        Ac[i+1]= tempc
        return i+1

    def distance(x1,y1,x2,y2):
        return np.sqrt((x2-x1)**2+(y2-y1)**2)

    #generate a list of data points
    mu, sigma = 0.4, 0.1 # mean and standard deviation
    a1 = list(np.random.normal(mu, sigma, 50))
    a2 = list(np.random.normal(mu, sigma, 50))
    a3 = [0.0]*50
    mu, sigma = 0.6, 0.1
    b1= list(np.random.normal(mu, sigma, 50))
    b2 = list(np.random.normal(mu, sigma, 50))
    b3 = [1.0]*50
    data= a1+b1,a2+b2,a3+b3

    fig = plt.figure()
    ax = plt.axes()

    ax.scatter(data[0], data[1], c=data[2], cmap='bwr',s=20);
    ax.scatter(0.4, 0.2, c='green',s=40);
    KNN = klist(0.4,0.2,data)
    ax.scatter(KNN[0], KNN[1], c='black',s=20);
```

References

[1] Rivest Cormen Leiserson and Stein. *Introduction to Algorithms. Third Edition*. MIT Press, 2009.

[2] Kevin P. Murphy. *Machine Learning. A Probabilistic Perspective*. MIT Press, 2012.