INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

Reconocimiento de Patrones

Algoritmo KNN

Emiliano González López

21/Mayo/2020

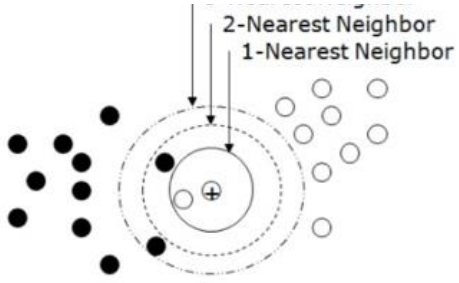
Marco teórico

La idea es realmente sencilla: el algoritmo clasifica cada dato nuevo en el grupo que corresponda, según tenga k vecinos más cerca de un grupo o de otro. Es decir, calcula la distancia del elemento nuevo a cada uno de los existentes, y ordena dichas distancias de menor a mayor para ir seleccionando el grupo al que pertenecer. Este grupo será, por tanto, el de mayor frecuencia con menores distancias.

El K-NN es un algoritmo de **aprendizaje supervisado**, es decir, que a partir de un juego de datos inicial su objetivo será el de clasificar correctamente todas las instancias nuevas. El juego de datos típico de este tipo de algoritmos está formado por varios atributos descriptivos y un solo atributo objetivo (también llamado clase).

Destacar que K-NN es muy sensible a:

La **variable *k***, de modo que con valores distintos de *k* podemos obtener resultados también muy distintos. Este valor suele fijarse tras un proceso de pruebas con varias instancias.
La **métrica** de similitud utilizada, puesto que esta influirá, fuertemente, en las relaciones de cercanía que se irán estableciendo en el proceso de construcción del algoritmo. La métrica de distancia puede llegar a contener pesos que nos ayudarán a calibrar el algoritmo de clasificación, convirtiéndola, de hecho, en una métrica personalizada.



Web Data Mining, de Bing Liu

Código de implementación

```
import matplotlib.pyplot as plt
import numpy as np
import math as m


print("        Hola bienvenid@ a la practica ")
print("Para comenzar ingresa el numero de clases que habrá.")
n = int(input("Numero = "))
n_por_clase = []

for i in range(n):
    n_por_clase.append( int(input("Ingresa el numero de características para
 la clase C" + str(i)+" = ")))

print("Cada clase tiene el siguiete numero de características: "+ str(n_por_
clase))
print("\n\nAhora que tenemos las clases procederemos a llenarlas con las car
acterísticas.")

c = 0
clases = []
arr_aux = []
clases = []
ft_class = []
classes = []
#--------------------Llenar clases-------------------------------------
while c < n:
    c2 = n_por_clase[c]
    print("\nClase a registrar características C"+ str(c))
    ft_class = []
    for x in range(c2):
        features = list(map(int, input("Ingresa (X"+str(x)+"), coordenadas (
digitos) separadas por Espacio ").split()))
        ft_class.append(features)
    classes.append(ft_class)
    c = c+1


print(classes)
c=0
xaxis = []
yaxis = []
x_per_class =[]
```

```
y_per_class =[]

#---------------------Imprimir clases---------------------------------
while c < n:
    c2 = n_por_clase[c]
    for x in range(c2):
        xaxis.append(classes[c][x][0])
        yaxis.append(classes[c][x][1])
    plt.scatter(xaxis, yaxis, label = ('Clase C'+ str(c)))
    x_per_class.append(xaxis)
    y_per_class.append(yaxis)
    xaxis = []
    yaxis = []
    c = c+1

plt.legend()
plt.show()
#---------------------- Calcular distancias --------------------------
distancias_por_clase = []
def calcular_distancia(muestra, k):
    global n
    global classes
    global distancias_por_clase
    c=0
    while  c < n:
        caux = n_por_clase[c]
        aux = []
        for x in range(caux):
            res = ((muestra[0]-classes[c][x][0])**2) + ((muestra[1]-
classes[c][x][1])**2)
            res = m.sqrt(res)
            aux.append(res)
        distancias_por_clase.append(aux)
        xaxis = [aux]
        c = c+1
    print("Distancias")
    print(distancias_por_clase)
    #++++++++++++++++++++++++++++ Evauluar distancia ++++++++++++++++++++
    c = 0
    distancias_finales = []
    while  c < n:
        distancias_por_clase[c].sort()
        distancias_finales.append(np.sum(distancias_por_clase[c][0:k]))
        c += 1
    distMin = min(distancias_finales)
```

```
    print("La distancia total minima pertenerce a la clase C{}".format(dista
ncias_finales.index(distMin)))
    print("Las distancias totales separadas por clase son: ")
    print(distancias_finales, end="\n\n")


#----------------------------- Menú ----------------------------------
op = 0
while op != 2:
    print("\nAhora que tienes los vectores podemos trabajar con ellos"
         +"\n Digita la opción que desees realizar.")
    print("1.- Ingresar una muestra ")
    print("2.- Salir")
    op = input("Ingresa la opción: ")
    if op == "1":
        muestra = list(map(float, input("Ingresa una coordenada a evaluar (d
igitos) separadas por Espacio ").split()))
        print("Para cauntos vecinos k quieres evaluar? (k) /n No puedes ser
mas de {}".format(max(n_por_clase)))
        k = int(input("K= " ))
        print("Todos los calculos serán basados en la distancia eulidea")
        #distancia = int(input("Ingresa seleccion "))
        calcular_distancia(muestra, k)

    if op == "2":
        break
```

Referencias

Soucy, P., & Mineau, G. W. (2001, November). A simple KNN algorithm for text categorization. In *Proceedings 2001 IEEE International Conference on Data Mining* (pp. 647-648). IEEE.

Deng, Z., Zhu, X., Cheng, D., Zong, M., & Zhang, S. (2016). Efficient kNN classification algorithm for big data. *Neurocomputing*, *195*, 143-148.