KdTree y K vecinos mas cercanos

Reconocimiento de Imagenes: Gatos vs Perros

Jose Torres Lima
jtorresli1812@gmail.com

1 K-Algoritmo de vecinos más cercanos (K -NN)

Dados los datos de entrenamiento D = $(x_1, y_1), ..., (x_N, y_N)$ y un punto de prueba. Regla de predicción: obteniendo los K ejemplos de entrenamiento más similares

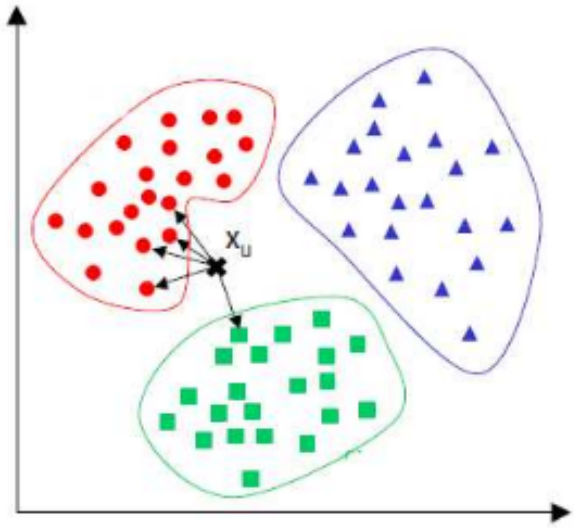

Figure 1.1: Clasificación usando KNN [2]

- Para la clasificación: asigne la etiqueta de clase mayoritaria (votación mayoritaria)
- Para regresión: asigne la respuesta promedio

El algoritmo requiere:

- El parámetro k: numero de vecinos mas cercanos a buscar
- Función de distancia: para calcular las similitudes entre ejemplos

2 Algoritmo

- Calcule la distancia del punto de prueba de cada punto de entrenamiento
- Ordenar las distancias en orden ascendente (o descendente)
- Use las distancias ordenadas para seleccionar los K vecinos más cercanos
- Usar regla de mayoría (para clasificación) o promedio (para regresión)

Nota: K -Nearest Neighbours es llamado un método no paramétrico

- A diferencia de otros algoritmos de aprendizaje supervisado, K -Nearest Neighbours no aprende un mapeo explícito f de los datos de entrenamiento
- Simplemente usa los datos de entrenamiento en el momento de la prueba para hacer predicciones

3 Elección de K - Tamaño del vecindario

K pequeño:

- Crea muchas regiones pequeñas para cada clase. Puede conducir a límites de decisión no uniformes y sobreajuste

K grande:

- Crea menos regiones más grandes.
- Generalmente conduce a límites de decisión más suaves (precaución: decisión demasiado suave el límite puede ser insuficiente)

Elección de K:

- A menudo depende de los datos y se basa en heurística
- O usando validación cruzada (usando algunos datos retenidos)
- En general, ¡una K demasiado pequeña o demasiado grande es mala!

4 KNN Ventajas

- Facil de programar.
- No se requiere optimización ni entrenamiento.
- La precisión de clasificación puede ser muy buena; puede superar a modelos más complejos

5 Problemas

- Para espacios de alta dimensión, ¡problema de que el vecino más cercano no esté muy cerca!
- Técnica basada en la memoria. Debe pasar por los datos para cada clasificación. Esto puede ser prohibitivo para grandes conjuntos de datos.

6 Implementación

6.1 Procesando las imagenes

Se eligió un conjunto de datos de imágenes de perros y gatos para la clasificación. el conjunto de datos es llamado "Dogs vs. Cats"

El archivo de entrenamiento contiene 25,000 imágenes de perros y gatos. Entrenamos el algoritmo con las imágenes de entrenamiento y probamos el algoritmo con las imágenes de la carpeta test1.zip para las pruebas dando nos como resultado(predicción) si la imagen es de un perro o un gato.

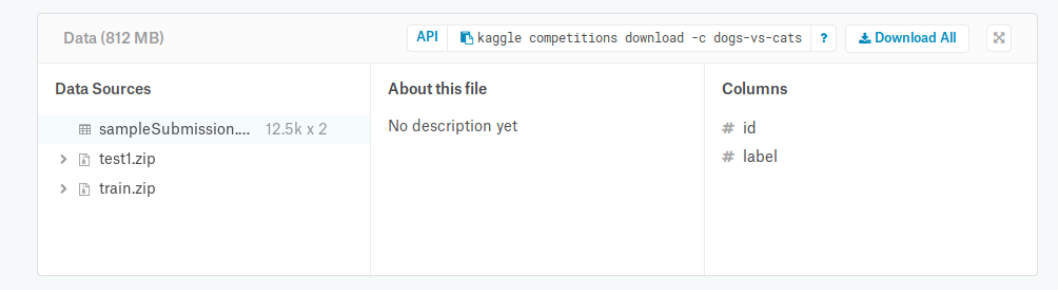

Figure 6.1: Imagen del contenido de la dataset [3]

El conjunto de datos esta disponible en [3].

```
1
2  import csv
3  import cv2
4  import numpy as np
5  import argparse
6  from imutils import paths
7  import os
8
9
10 ap = argparse.ArgumentParser()
11 ap.add_argument("-d", "--dataset", required=True,
12     help="path to input dataset")
13
14 args = vars(ap.parse_args())
15
16 print("[INFO] describing images...")
17 imagePaths = list(paths.list_images(args["dataset"]))
18
19 features = [["dat1","dat2","dat3","dat4","dat5","dat6","clase"]]
```

```
20
21 for (i, imagePath) in enumerate(imagePaths):
22
23     image = cv2.imread(imagePath)
24     image = cv2.resize(image, (32, 32))
25
26     means = cv2.mean(image)
27     means = means[:3]
28     (means, stds) = cv2.meanStdDev(image)
29     stats = np.concatenate([means, stds]).flatten()
30     print(stats)
31
32     label = imagePath.split(os.path.sep)[-1].split(".")[0]
33     stats = stats.tolist()
34     stats.append(label)
35
36     features.append(stats)
37
38
39
40 with open('cat-dog.csv', 'w', newline='') as file:
41     writer = csv.writer(file)
42     writer.writerows(features)
```

Listing 1: Procesando las imágenes

La implementación para el procesamiento de la imágenes de entrenamiento para obtener el vector característica de cada imagen la cual representaría a la imagen para cargarla en el kd tree para la parte de calcular los vecinos mas cercanos usando el algoritmo de k-vecinos mas cercanos, fue realizada en python usando la librería opencv se opto por sacar la media y la desviación estándar de los canales RGB de la imagen obteniendo un arreglo de 6 valores(3 de la media y 3 de la desviación). Se cargo esta información en un archivo cat-dog.csv

6.2 Procesando la imagen a calcular su clase

```
1 import csv
2 import cv2
3 import numpy as np
4 import argparse
5 from imutils import paths
6 import os
7
8 ap = argparse.ArgumentParser()
```

```
ap.add_argument("-i", "--image", required = True, help = "Path to the
    ↪ image")
args = vars(ap.parse_args())


features = []
image = cv2.imread(args["image"])
image = cv2.resize(image, (32, 32))
means = cv2.mean(image)
means = means[:3]
(means, stds) = cv2.meanStdDev(image)
stats = np.concatenate([means, stds]).flatten()
print(stats)

stats = stats.tolist()
features.append(stats)

with open('point.txt', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerows(features)
```

Listing 2: Procesando la imagen de consulta

Para la imagen enviada para calcular su clase, se calculo su vector característica usando la media y la desviación como se hizo para las imágenes de entrenamiento. Este arreglo de seis valores fue guardado en un archivo para cargarlo en el kd-tree para calcular la clase a la que pertenece.

6.3 Obteniendo la clase, en el archivo kdtree.js

```


var data =[]
var i=1;

const csv = require('csv-parser');
const fs = require('fs');

var point = fs.readFileSync('point.txt').toString().split(",");
point = point.slice(0,k)

for(var i = 0 ; i<point.length; i++) {
    point[i] = parseFloat(point[i]);
}

```

```
16
17 fs.createReadStream('cat-dog.csv', {star:10})
18     .on('error', () => {
19     })
20
21     .pipe(csv())
22     .on('data', (row) => {
23
24
25         let str = `${row["SepalLengthCm"]} ${row["SepalWidthCm"]} ${row["
               ↪ PetalLengthCm"]} ${row["PetalWidthCm"]}`;
26
27         var a = parseFloat(`${row["dat1"]}`);
28         var b = parseFloat(`${row["dat2"]}`);
29         var c = parseFloat(`${row["dat3"]}`);
30         var d = parseFloat(`${row["dat4"]}`);
31         var e = parseFloat(`${row["dat5"]}`);
32         var f = parseFloat(`${row["dat6"]}`);
33         var g = `${row["clase"]}`;
34         data.push([a,b,c,d,e,f,g]);
35
36         i++;
37
38     })
39
40     .on('end', () => {
41         var cat=0;
42         var dog =0;
43         var root = build_kdtree(data);
44         console.log("K: ",kn);
45
46         console.log("Punto a consultar: ");
47         for(var i = 0 ; i<point.length; i++) {
48             console.log(point[i]);
49         }
50
51
52         console.log("\n");
53
54         knearest=[]
55         knearestpoints(root, point , knearest);
56
57         for(var i = 0 ; i<knearest.length-1; i++)
58         { if(knearest[i][k]=='cat')
```

```
59                  cat++;
60              else
61                  dog++;
62          }
63
64          if(cat>dog)
65              console.log("cat")
66          else
67              console.log("dog")
68
69      })
70
71      .on('close', function(){
72
73      })
```

Listing 3: Obteniendo la clase usando kdtree y KNN

En el archivo kdtree.js se carga el archivo cat-dog.csv donde están los vectores característicos de cada imagen como también la clase a la que pertenecen, usados para calcular las distancias a la imagen en consulta la cual también se guardo su vector característico en un archivo, los archivos son cargados y se calculan las distancias usando los vectores característicos. El uso de un kdtree optimiza la búsqueda en un algoritmo de k-vecinos mas cercanos. Las etiquetas de las clases se usaron para determinar a que clase pertenece la imagen a consulta una vez obtenido los k-vecinos mas cercanos esto por mayoría de votos.

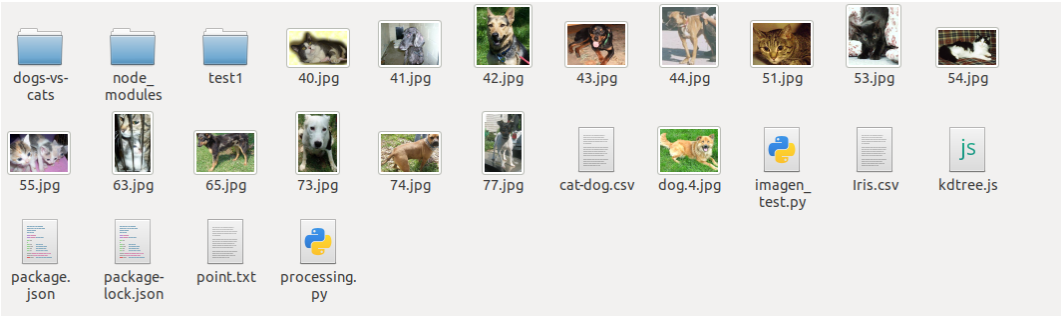

Figure 6.2: Los archivos usados como también las carpetas de imágenes

En la figura anterior se muestran las carpetas de las imágenes para el entrenamiento, como las imágenes para las pruebas, además de los archivos con las implementaciones de los algoritmos.

7 Resultados

Para la parte de pruebas primero calculamos el vector característico de la imagen a evaluar, para luego calcular la clase a la que pertenece. la imagen debajo muestra los resultados.

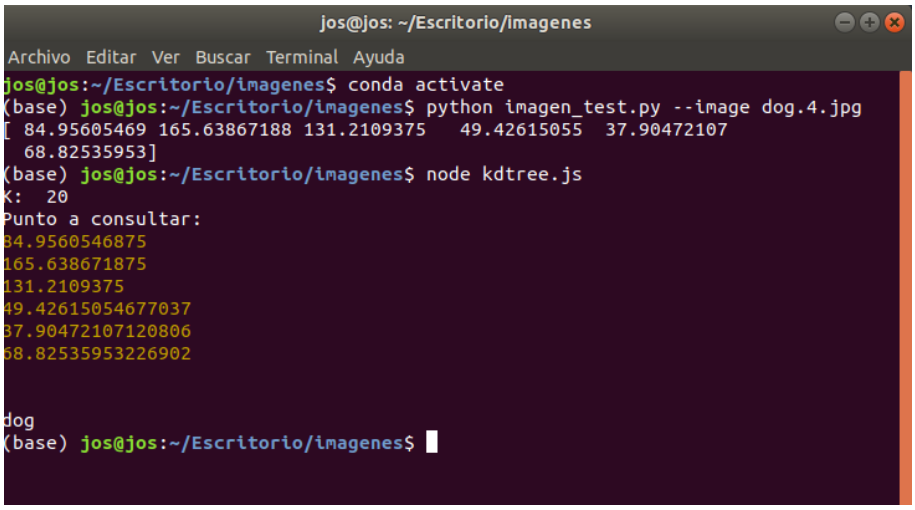

Figure 7.1: Mostrando los resultados

Figure 7.2: Imagen tomada para la prueba anterior

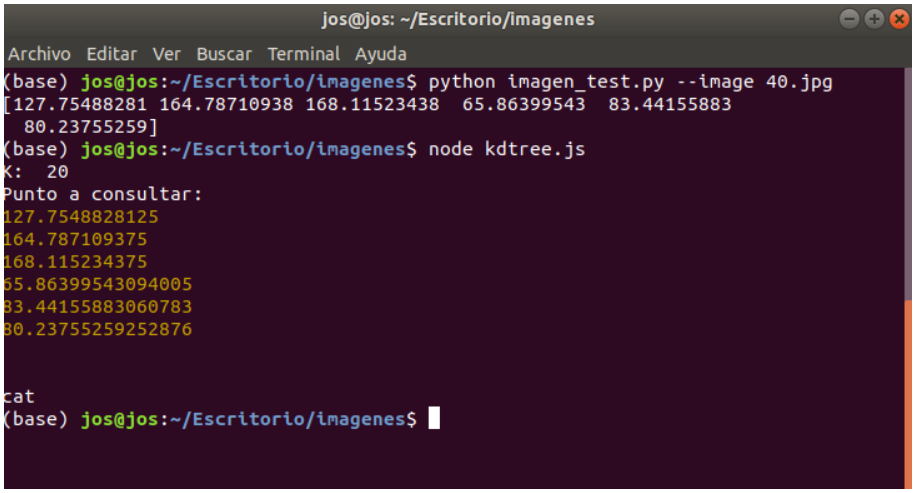

Figure 7.3: Mostrando los resultados

Figure 7.4: Imagen tomada para la prueba anterior

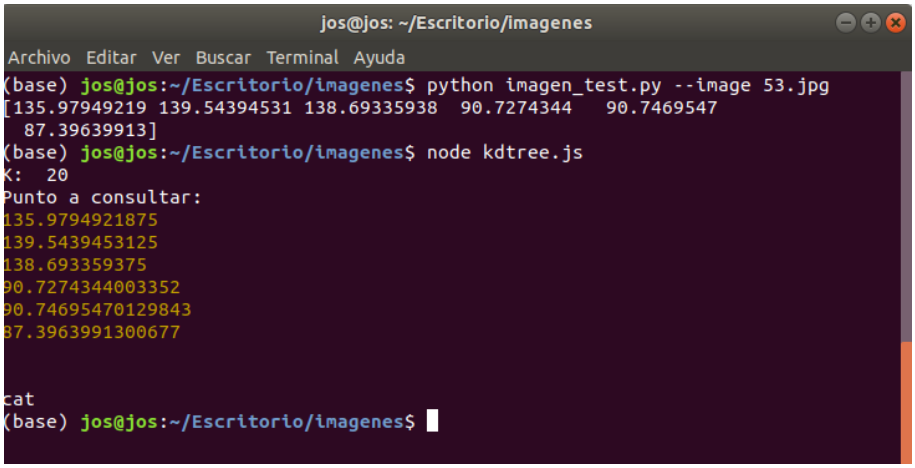

Figure 7.5: Mostrando los resultados

Figure 7.6: Imagen tomada para la prueba anterior

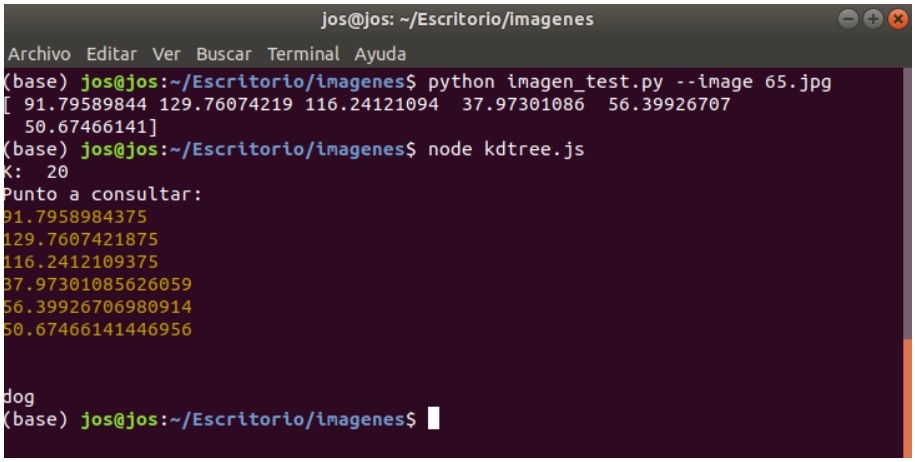

Figure 7.7: Mostrando los resultados

Figure 7.8: Imagen tomada para la prueba anterior

8 Conclusiones

Concluyendo este documento se mostró la clasificación de imágenes usando al algoritmo de los k-vecinos mas cercanos el cual es un algoritmo supervisado, para su optimización en la búsqueda de los vecinos se integro una estructura kdtree. Por otro lado los resultados muestran que se puede clasificar imágenes usando este algoritmo. Con lo que respecta a las dimensiones por la cual una imagen puede ser representada hay algoritmos que nos permiten reducir esta dimensionalidad como es el algoritmo PCA para reducir las dimensiones de las imágenes y así obtener resultados mas precisos. La elección de los k

vecinos también es un valor de interés para la clasificación.

References

[1] Nearest neighbor methods. David Sontag. Disponible en http://people.csail.mit.edu/dsontag/courses/ml12/slides/lecture10.pdf

[2] http://www.cs.cornell.edu/courses/cs4758/2013sp/materials/cs4758-knn-lectureslides.pdf

[3] Dogs vs. Cats. Disponible en: https://www.kaggle.com/c/dogs-vs-cats/overview