K Nearest Neighbors Analysis

Josué Fernández Díaz
Instituto Tecnológico de Costa Rica
Ingeniería en Computación
Email: josue.0795@gmail.com

25 de Febrero del 2018

Abstract

En este proyecto se realiza el proceso denominado K Nearest Neighbors, con una base de entrenamiento de 50 mil imágenes del CIFAR-10, dadas al programa, además de un grupo de imágenes de prueba para probar la exactitud del programa a la hora realizar la distinción entre ambos conjuntos, utilizando las funciones Levenshtein, Manhattan y Chevyshev. De esto se obtiene que la primera tiene un índice de exactitud mas alto, a pesar de su enorme cantidad de tiempo para realizar los cálculos. Seguido a esto esta Manhattan que usa un tiempo promedio para una cantidad de resultados positivos, y por ultimo Chevyshev que tiene datos desfavorables en el proyecto, comparados con las otras dos.

1 Introducción

El reconocimiento de imágenes es uno de los tantos avances de la inteligencia artificial, donde se entrena un programa de forma que, a la hora de ponerlo a prueba, pueda identificar de forma lo mas acertada posible la imagen enviada, esto mediante una serie de funciones previamente especificadas, que reciben una serie de imágenes, y las comparan con su base de conocimiento, también dada a conocer previamente.
En este proyecto se tiene una serie de funciones para medir la diferencia entre imágenes, las cuales son Manhattan[4], Chevyshev[2] y Levenshtein[3], que aplican formulas entre píxeles y de esta forma pueden identificar la similitud entre dos imágenes, obteniendo la exactitud del programa.

2 Metodología

En este proyecto se utiliza una metodología de comparación de píxeles. Cada imagen es de 32x32, con tres colores RGB, agrupados en un array de 3072 posiciones, donde los primeros 1024 son Rojos, los siguientes verdes, y los últimos

azules. Además, se cuenta con tres funciones de medición de diferencias entre píxeles de las imágenes, las cuales son aplicadas de forma independiente a petición del usuario. Además se usa una variable k para tomar la cantidad de vecinos mas cercanos a la imagen de entrada, de forma que se pueda realizar una serie de "votaciones", esto para dar mas posibilidad de que el programa pueda responder de forma acertada. Es decir, si el k es 3, entonces se seleccionan las tres imágenes con mas similitud a la inicial, y obteniendo la clase de estas, se puede decir con mas exactitud la clase original de la imagen. Si dos son 'gato' y una es avión', pues el sistema dará como resultado 'gato' ya que es el que tuvo mas votaciones.
Las funciones utilizan una serie de formulas que obtienen la misma posición del mismo píxel para ambas imágenes, de forma que puedan restarlos y pasar al siguiente píxel. Cada función tiene su propia formula de obtener la distancia entre los píxeles de las imágenes, dando resultados distintos.
Se utilizó una clase donde se entrenan los datos, teniendo un campo data para el array que contiene los datos de las imágenes, y label que contiene las clases respectivas para dichas imágenes. Posteriormente, se tiene un ciclo que obtiene todas las imágenes del set de entrenamiento, y luego se ponen las imágenes de prueba en contraste con estas de entrenamiento, realizando el proceso mencionado anteriormente.

2.1 Trabajo Relacionado

Se utilizan tres funciones para obtener la diferencias entre los píxeles, las cuales pueden ser seleccionadas por el usuario indicando el tipo.
La Manhattan[4] que realiza la suma de la resta de los valores absolutos de la forma:

$$\sum_{i=1}^{n} |(p_i - q_i)| \tag{1}$$

Por otro lado, la distancia Chevyshev[2] obtiene el máximo entre la resta de estos píxeles, de igual forma con valor absoluto y presenta una forma:

$$max(|p_i - q_i|) \tag{2}$$

La Levenshtein[3] es la mas complicada de las tres, y de hecho es la que presenta mas tiempo a la hora de obtener los cálculos entre las tres. Presenta una forma:

$$lev_{a,b}(i,j) = \begin{cases} max(i,j) & \text{if } min(i,j) = 0 \\ min \begin{cases} lev_{a,b}(i-1,j) + 1 \\ lev_{a,b}(i,j-1) + 1 \\ lev_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & otherwise \end{cases} \tag{3}$$

Como guía para realizar el proyecto, se usó un Tutorial[1] de como realizar el proceso de K Nearest Neighbors, donde se explica paso a paso todo lo necesario para realizar el procedimiento.

3 Experimentos

Debido a la gran cantidad de datos, no se realizó las pruebas con todos estos, sino que se tomó una pequeña porción al azar para realizar las pruebas. Incluso con esta mínima cantidad de datos, los cálculos tardaron un tiempo considerablemente alto. Cabe destacar que se usó la misma porción para todas las pruebas, de esta forma se puede asegurar que los datos recaudados esta bien estimados, y no presentan sesgo ni disposición por ninguna función. Se realizaron primero las pruebas de Manhattan, con todos los valores k permitidos (del 1 al 3), posteriormente con Chevyshev y por ultimo con Levenshtein.

3.1 Resultados

Los resultados se pueden mostrar en las imágenes, donde se inicia por Manhattan(k=1 hasta k=3), luego Chevyshev(k=1 hasta k=3) y por ultimo Levenshtein (k=1 hasta k=3). En cuanto a Manhattan, logró un 20% entre los datos utilizados, tanto para k=1, k=2 o k=3. Chevyshev por otro lado, lo hizo mucho peor, con un 0& con todos los k utilizados en el mismo set de datos utilizado. Por otro lado, Levenshtein logró un 40% siendo el mas alto entre los tres. De igual forma, con el mismo set de datos de prueba.

4 Conclusiones

Como se mencionó anteriormente, el hecho de que se escoja un k distinto no cambia el resultado. Cuando k=1, solo tiene una posibilidad, la cual es la distancia mas corta entre el set de entrenamiento con la imagen, por lo tanto tiene un margen alto. Por otro lado, cuando k=2, escoge las dos imágenes mas cercanas, de igual forma ordenadas por la distancia entre estas con la original, sabiendo esto, el resultado siempre será igual cuando k=1.
Por otro lado, cuando k=3, presenta una mayor oportunidad de diferencia con las dos anteriores, pero como se muestra, esto no influyó en el resultado mostrado, ya que sigue mostrando el mismo cuando k=1, o bien cuando k=2.
En cuanto a las funciones utilizadas, se puede ver como Manhattan tiene una mayor precisión comparada a Chevyshev, pero superándolos está Levenshtein, aunque la cantidad de tiempo utilizada para los cálculos excede por mucho a ambas. Chevyshev y Manhattan usan, comparándolas entre si, una cantidad similar de tiempo para los cálculos, mientras que Levenshtein usa el mismo tiempo que las dos otras funciones juntas, o incluso mas, esto por la complejidad y el método que usa, pero esto también le da una mayor precisión a la hora de clasificar los cambios en las imágenes, por eso es mucho mas alto el tiempo.

References

[1] Jason Brownlee. Tutorial to implement k-nearest neighbors in python from scratch, 2014.

[2] Wikipedia. Chebyshev distance, 2016.

[3] Wikipedia. Levenshtein distance, 2016.

[4] Wikipedia. Taxicab geometry, 2018.