UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

Documentación Algorítmo KNN

Sergio Flores Labra
Ingeniería civil en Computación

7 de septiembre de 2017

1. Introducción

El presente informe tiene por finalidad dar a conocer en resumen la implementación del algorítmo de clasificación KNN para el conjunto de datos de Iris. Además dar información sobre cómo ejecutarlo y las tecnologías empleadas para dar solución a dicho problema.

2. Desarrollo

El algorítmo de los k-vecinos más cercanos KNN [4], en inglés k-nearest neighbors, es uno de los métodos de clasificación más utilizados por el público general que desea realizar estimaciones dado un conjunto de entrenamiento. Este modelo es un método de aprendizaje de forma supervisada, ya que es necesario contar con un conjunto de datos de entrenamiento en donde se conoce la salida esperada. Para generar su pronóstico, primeramente hace uso del cálculo de la distancia euclidiana con sus vecinos, para luego seleccionar los k-vecinos más cercanos, es decir:

$$x_i = (x_{1i}, x_{2i}, ..., x_{ni}) \in X \tag{1}$$

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (x_{ri} - x_{rj})^2} \tag{2}$$

Continuando con el proceso, lo que hace este algorítmo es encontrar los k elementos que se encuentren más cercanos a él.

$$(d_1, d_2, ..., d_p) \in D \tag{3}$$

$$neighbors = sort(D).limit(k) \tag{4}$$

Por último encontrar la clase c que se repite más veces.

$$(c_1, c_2, ..., c_m) \in C \tag{5}$$

$$\forall c_i \in C,\ count_i = \sum_{j=1}^{k} a = \begin{cases} 1, \text{si } c_i == x[dj] \\ 0, \text{si } c_i \neq x[dj] \end{cases} \tag{6}$$

$$pred = C[\text{máx}(count)] \tag{7}$$

2.1. Tecnologías utilizadas

Para dar solución a este problema se utilizó el lenguaje *Python* en su versión 3, en conjunto con las siguientes librerías:

- Numpy [5]: Librería para trabajar con arreglos.
- Matplotlib [2]: Creador de gráficos.
- Tkinter [1]: Creador de ventanas gráficas.
- Math [3]: Librería para trabajar con funciones matemáticas predefinidas.

2.2. Ejecución

Existen dos formas de utilización, la primera es utilizar el clasificador por consola, ejecutando el siguiente código:

```
python3 main.py -f database -k 10 -s 10
```

Donde:

- database: Archivo de datos de entrenamiento.
- k: Número máximo de k-vecinos, 10 por defecto.
- s: Número máximo de conjuntos para la evaluación cruzada, 10 por defecto.

No es necesario incluir todos los datos, ya que poseen un valor por defecto.

Por otra parte, al no incluir el archivo de entrada, utiliza una ventana para seleccionar el archivo de entrenamiento (Figura 1), mostrando los resultados en forma de gráfico (Figura 2).

```
python3 main.py
```

Figura 1: Selección de Archivo de entrenamiento.

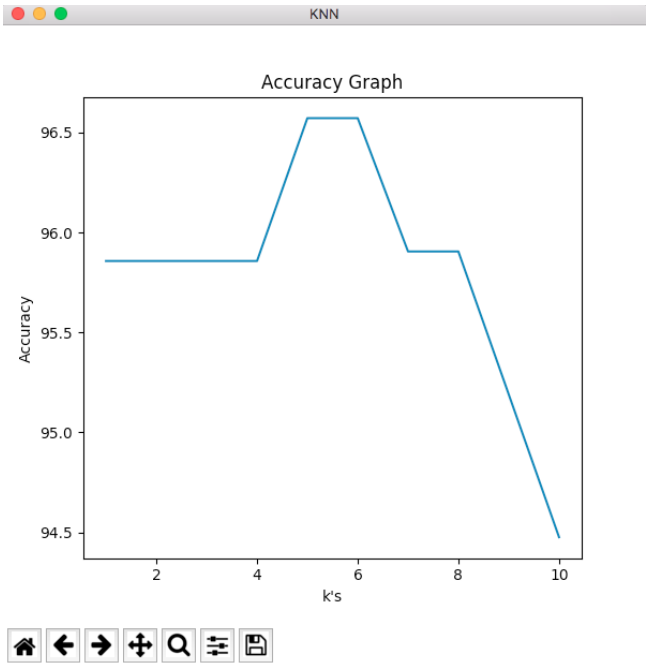

Figura 2: Gráfico de Resultados.

3. Conclusión

A modo personal, creo que es difícil determinar que valor de k es el más adecuado, ya que cada vez que se ejecuta el programa, el orden de los datos en el cuál se entrena cambia, provocando que la respuesta varíe, por lo que para $k = 3$ es el mejor valor a la siguiente ejecución puede ser $k = 7$. Por otra parte, es un algorítmo simple de programar y que puede ser muy útil tanto para clasificar, introducirse en el área de Machine Learning, como para hacer mejoras ya que se puede malear y ajustar a lo que uno desee.

Referencias

[1] Phil Hughes. Python and tkinter programming. *Linux J.*, 2000(77es), September 2000.

[2] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science and Engg.*, 9(3):90–95, May 2007.

[3] Amit Saha. *Doing Math with Python: Use Programming to Explore Algebra, Statistics, Calculus, and More!* No Starch Press, San Francisco, CA, USA, 1st edition, 2015.

[4] B. W. Silverman and M. C. Jones. E. fix and j.l. hodges (1951): An important contribution to nonparametric discriminant analysis and density estimation: Commentary on fix and hodges (1951). *International Statistical Review / Revue Internationale de Statistique*, 57(3):233–238, 1989.

[5] Stefan van der Walt, S. Chris Colbert, and Gael Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science and Engg.*, 13(2):22–30, March 2011.