Centro de investigación en Computación

Clasificación inteligente de Patrones

Clasificador 1-NN

Autor:
Diego Noguez Ruiz

Profesor:
Dr. Cornelio Yáñez Márquez

7 de abril de 2023

Índice general

Índice general **1**
1. Introducción 2
2. Desarrollo 2
   2.1. Propósito de la tarea 2
   2.2. Parte 1 2
   2.3. Parte 2 3
3. Conclusiones y trabajo futuro 4

1. Introducción

A estas alturas del curso, nos encontramos en un nivel taxonómico elevado en la escala de aprendizaje. Previamente, hemos estudiado los elementos básicos del cip y hemos indagado en la matriz de confusión debido a su gran importancia en medidas de desempeño; sin embargo, en este momento, solo conocemos un clasificador, el euclidiano. Este clasificador es simple, por lo cual, fue de gran ayuda para empezar a construir conocimiento. No obstante, ha llegado el momento de estudiar un nuevo clasificador, el 1-NN. Entender este clasificador será útil para introducir la generalización del algoritmo, llamado k-NN.
En esta tarea, trabajaremos con el dataset *Haberman* [1] para poner en práctica este clasificador.

2. Desarrollo

2.1. Propósito de la tarea

Ejemplificar el algoritmo 1-NN

2.2. Parte 1

Dado que en esta tarea se va a trabajar con el clasificador 1-NN, es importante definir como es que el algoritmo decide que clase asignar a un patrón, ver definición 1

Definición 1. *Sea $p \in P$, donde P es el conjunto de patrones de prueba. El clasificador 1-NN le asignará la clase C sii C es la clase de e_0 y $d(x, e_0) = \min_{e \in E}\{d(x, e)\}$, donde E es el conjunto de patrones de entrenamiento.*

En relación con el contenido del presente RD 09

- Parte 1.a) **Instrucciones:** Verificar que son correctos los cálculos para la clasificación de los tres patrones de prueba de la clase negative.
  Programando el clasificador en python obtenemos los siguientes resultados para las patrones de prueba de la clase negativa.
  - Patrón de prueba: [41, 0]
    El patrón [41, 0] fue clasificado BIEN en la clase negativa, su distancia más corta fue: 1.0
  - Patrón de prueba: [42, 0]
    El patrón [42, 0] fue clasificado BIEN en la clase negativa, su distancia más corta fue: 1.0
  - Patrón de prueba: [64, 22]
    El patrón [64, 22] fue clasificado MAL en la clase positiva, su distancia más corta fue: 3.6055512754639896

Por lo tanto tenemos la fila inferior de la matriz de confusión con los siguientes valores:

- $TN = 2$
- $FP = 1$

- Parte 1.b) **Instrucciones:** Realizar los cálculos para la clasificación de los tres patrones de prueba de la clase positive.
  Usando el mismo código que en la parte 1.a) obtenemos que:

  - Patrón de prueba: [53, 1]
    El patrón [53, 1] fue clasificado MAL en la clase negativa, su distancia más corta fue: 3.6055512754639896
  - Patrón de prueba: [74, 3]
    El patrón [74, 3] fue clasificado MAL en la clase negativa, su distancia más corta fue: 3.6055512754639896
  - Patrón de prueba: [78, 1]
    El patrón [78, 1] fue clasificado MAL en la clase negativa, su distancia más corta fue: 3.6055512754639896

  Por lo tanto tenemos la primera fila de la matriz de confusión:

  - $TP = 0$
  - $FN = 3$

  Por lo tanto tenemos que la matriz de confusión es la siguiente:

$$M = \begin{bmatrix} 0 & 3 \\ 1 & 2 \end{bmatrix}$$

  La cual coincide con la obtenida en clase.

2.3. Parte 2

Aplicar el algoritmo 1-NN en la partición Hold-out 75-25 del Haberman 58 que se incluye en la próxima diapositiva.

- Parte 2.a) **Instrucciones:**) Reportar la matriz de confusión.

  Se procedió a realizar el clasificador 1-NN para los datos que nos solicitan; para esto, dado que de la primera parte ya se contaba con el código en python, simplemente se hicieron los ajustes necesarios para este caso particular obteniendo la siguiente matriz de confusión.

$$M = \begin{bmatrix} 1 & 1 \\ 1 & 4 \end{bmatrix}$$

- Parte 2.b) **Instrucciones:**)Reportar los valores de las medidas de desempeño estudiadas en el curso, cuando sea adecuado.

  Para este dataset se tiene $IR = 2.5$, entonces es un dataset desbalanceado, por lo tanto no tiene sentido hablar de *Accuracy*, sin embargo si podemos calcular las otras medidas de desempeño.

  - *Recall*: 0.5
  - *Specificity*: 0.8
  - *Balanced Accuracy*: 0.65

3. Conclusiones y trabajo futuro

Con esta tarea aprendimos a trabajar con el clasificador k-NN en su forma más sencilla, cuando $k = 1$. Esto nos permitió sentar las bases para estudiar en breve la generalización con un valor de $k \geq 1$.

Considero que este clasificador no es tan malo, ya que compara un patrón de prueba con todos aquellos que son de experimento, y usualmente todos aquellos patrones que son de una clase deben tener atributos en común, lo cual implica que la mayoría de patrones de una clase deben estar cerca en algún sentido, y es por ello que este algoritmo debería clasificar correctamente muchos patrones. Sin embargo, su desventaja es la amplia complejidad que tiene, $O(n * m)$, donde n es el número de patrones de entrenamiento y m es el número de patrones del conjunto de prueba. Esto no es óptimo cuando se trabaja con grandes cantidades de datos.

Personalmente, había escuchado el nombre de este clasificador antes, y creo que es uno de los más conocidos en la comunidad. Espero que la generalización del k-NN ayude a mejorar las desventajas del clasificador actual.

Referencias

[1] UCI ML, "Haberman's survival data set." https://archive.ics.uci.edu/ml/datasets/haberman's+survival. Visitado en Marzo 2023.