INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Tarea 09

Clasificador 1-NN

Edgar Fernando Espinosa Torres

Clasificación inteligente de patrones

Dr. Cornelio Yáñez Márquez

Ciudad de México, 4 de noviembre de 2023.

Índice general

Índice de tablas

Introducción

La clasificación de patrones es uno de los pilares fundamentales del aprendizaje automático, entre los algoritmos más sencillos e intuitivos se encuentran los de los Clasificadores Euclidiano y 1-Nearest Neighbor (1-NN). Aunque ambos emplean medidas de distancias para discernir las clases, difieren en su enfoque: mientras en el Clasificador Euclidiano se utiliza un centroide para representar a cada clase de la partición de entrenamiento y se compara con cada patrón de prueba, el 1-NN se basa en la proximidad inmediata de cada patrón de prueba con los patrones de entrenamiento más cercanos (sus vecinos más cercanos).

En esta tarea, nos centramos en verificar los cálculos para el Ejemplo Nutt y aplicar el Clasificador 1-NN en las particiones generadas mediante Hold-out para los datasets Nutt y Haberman del año 1958. Ambos datasets se brindaron en el recurso didáctico 09.

También, se calcularon sus matrices de confusión y a partir de ellas, se obtuvieron las medidas de desempeño $Accuracy$ y $Error\ Rate$ si los valores de $IR \leq 1.5$ y $Sensitivity$, $Specificity$ y $Balanced\ Accuracy$ válidas para cualquier valor de IR.

Desarrollo y discusión

Propósitos de la tarea

Ejemplificar el algoritmo 1-NN.

Parte 1

En relación con el Ejemplo Nutt:

1.a) Verificar que son correctos los cálculos para la clasificación de los 8 patrones de prueba.

El dataset Nutt se generó a partir del trabajo de Nutt en colaboración con otros investigadores que estudiaban el cáncer de cerebro. Como resultado de sus investigaciones, generaron un conjunto de 28 patrones de tejido cerebral, de los cuales 14 corresponden a glioblastomas clásicos (clase Positive), mientras que los 14 patrones restantes corresponden a glioblastomas no clásicos (clase Negative).

Cada patrón contiene 1070 atributos numéricos, los cuales representan los niveles de intensidad de expresión de los genes [1].

En cuanto a la instrucción del inciso, se encontró que las distancias mostradas en la tabla del RD 09 son distintas a las que se obtuvieron utilizando la métrica euclidiana y, en consecuencia, las clasificaciones difieren.

A continuación, se presentan los resultados del 1-NN utilizando la métrica euclidiana y los mismos conjuntos E y P:

Patrón de prueba (P)	Clase del patrón de prueba	Distancia más pequeña a los patrones de entrenamiento (en E)	Vecino más cercano (NN)	Clase del vecino más cercano (NN)	Resultado
P4	Positive	4740.82	P25	Negative	FN
P6	Positive	7806.34	P8	Positive	TP
P10	Positive	5960.67	P26	Negative	FN
P14	Positive	8965.22	P17	Negative	FN
P15	Negative	1782.90	P21	Negative	TN
P18	Negative	3411.94	P20	Negative	TN

P22	Negative	2305.63	P23	Negative	TN
P28	Negative	2274.50	P21	Negative	TN

Tabla 1: Resultados del clasificador 1-NN aplicado al dataset Nutt.

1.b) Reportar accuracy, error rate, sensitivity, specificity y Balanced accuracy.

A continuación, se presenta la matriz de confusión correspondiente al ejemplo:

1	3
0	4

Tabla 2: Matriz de confusión para Nutt.

El $imbalance\ ratio$ es $IR = \frac{14}{14} = 1.0 \leq 1.5$, es decir el dataset está balanceado. Por lo tanto, podemos utilizar las medidas de desempeño $accuracy$ y $error\ rate$.

$$accuracy = \frac{TP+TN}{TP+FN+FP+TN} = \frac{1+4}{1+3+0+4} = \frac{5}{8} = 0.625$$

$$error\ rate = 1 - accuracy = 1 - 0.625 = 0.375$$

Para cualquier valor de IR es válido calcular las siguientes medidas de desempeño:

$$Sensitivity = \frac{TP}{TP + FN} = \frac{1}{1 + 3} = \frac{1}{4} = 0.25$$

$$Specificity = \frac{4}{0 + 4} = \frac{4}{4} = 1.0$$

En consecuencia, se puede obtener:

$$Balanced\ Accuracy = \frac{Sensitivity + Specificity}{2} = \frac{0.25 + 1.0}{2} = 0.625$$

Parte 2

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

Se trabaja con un dataset derivado de Haberman's Survival de UCI [2]. Este dataset contiene datos sobre la sobrevivencia de las pacientes operadas de cáncer de mama.

Al dataset original se le realizó un conjunto de técnicas conocidas como limpieza de datos para poder aislar al año 1958.

Con ello se obtiene un dataset para trabajar en single-label classification. Tiene sólo dos atributos (tipo numérico) y pocos patrones, es decir, exhibe una cardinalidad pequeña y también es biclase.

El método de validación aplicado es Hold-out 75-25. En total, se tienen 28 patrones, de los cuales 21 (75 %) conforman el conjunto de entrenamiento E y 7 (25 %) el conjunto de prueba P.

La clase Positive representa a los pacientes que no sobrevivieron los primeros 5 años y la clase Negative representa a aquellos que sobrevivieron 5 años o más.

El algoritmo utilizado es 1-NN con métrica euclidiana.

Patrón de prueba (P)	**Clase del patrón de prueba**	**Distancia más pequeña a los patrones de entrenamiento (en E)**	**Vecino más cercano (NN)**	**Clase del vecino más cercano (NN)**	**Resultado**
54 1	Negative	1.000000	55 1	Negative	TN
46 3	Negative	4.472136	50 1	Negative	TN
72 0	Negative	2.000000	70 0	Positive	FP
47 3	Negative	3.605551	50 1	Negative	TN
53 1	Negative	2.000000	55 1	Negative	TN
46 2	Positive	4.123106	50 1	Negative	FN
48 11	Positive	4.472136	44 9	Positive	TP

Tabla 3: Resultados del clasificador 1-NN aplicado al dataset Haberman 58.

2.a) Reportar la matriz de confusión.

1	1
1	4

Tabla 4: Matriz de confusión para Haberman 58.

2.b) Reportar *accuracy*, *error rate*, *sensitivity*, *specificity* y *Balanced accuracy*, cuando sea adecuado.

El $imbalance\ ratio$ es $IR = \frac{20}{8} = 2.5 > 1.5$, es decir el dataset está desbalanceado. Por lo tanto, no podemos utilizar las medidas de desempeño $accuracy$ ni $error\ rate$.

Sin embargo, para cualquier IR es válido calcular:

$$Sensitivity = \frac{TP}{TP + FN} = \frac{1}{1+1} = \frac{1}{2} = 0.5$$

$$Specificity = \frac{TN}{FP + TN} = \frac{4}{4+1} = \frac{4}{5} = 0.8$$

En consecuencia, se puede obtener:

$$Balanced\ Accuracy = \frac{Sensitivity + Specificity}{2} = \frac{13}{20} = 0.65$$

Conclusiones

Antes de esta tarea 09, solamente habíamos estudiado un algoritmo basado en el cálculo de centroides para cada clase durante la fase de aprendizaje. Con esto, se hace referencia al del Clasificador Euclidiano, el cual utiliza un enfoque eager (ansioso).

Por otro lado, el Clasificador 1-NN que hemos aplicado a los datasets Nutt y Haberman 58 utiliza un enfoque lazy (perezoso), es decir, no realiza ninguna operación durante la fase de aprendizaje.

Las medidas de desempeño calculadas aplicando 1-NN a ambos datasets revelan resultados que, si bien no son deficientes, tampoco resultan sobresalientes.

Una desventaja significativa del 1-NN en comparación al Euclidiano radica en su elevado costo computacional. Este algoritmo debe comparar la distancia (en este caso euclidiana) de cada patrón de prueba con todos los patrones de entrenamiento para encontrar su vecino más cercano, lo cual incrementa considerablemente el tiempo de ejecución cuando se trata de datasets de gran volumen o alta dimensionalidad.

Referencias bibliográficas

[1] Nutt, C. L. et al. (2003). Gene expression-based classification of malignant gliomas correlates better with survival than histological classification. Cancer Research, 63(7), 1602–1607.

[2] Yáñez Márquez, C. (2023). RD 09: Clasificador 1-NN [Diapositivas de clase]. Centro de Investigación en Computación.

Anexos

```
import pandas as pd

# Datos proporcionados
negative_e_data = [
    [40, 0, 'Negative'],
    [37, 0, 'Negative'],
    # ... (datos omitidos para brevedad)
    [64, 0, 'Negative']
]

negative_p_data = [
    [54, 1, 'Negative'],
    # ... (datos omitidos para brevedad)
    [53, 1, 'Negative']
]

positive_e_data = [
    [44, 9, 'Positive'],
    # ... (datos omitidos para brevedad)
    [70, 0, 'Positive']
]

positive_p_data = [
    [46, 2, 'Positive'],
    [48, 11, 'Positive']
]

# Crear DataFrames a partir de los datos proporcionados
negative_e_df = pd.DataFrame(negative_e_data, columns=['Age',
'Axillary nodes', 'Survival status'])
negative_p_df = pd.DataFrame(negative_p_data, columns=['Age',
'Axillary nodes', 'Survival status'])
positive_e_df = pd.DataFrame(positive_e_data, columns=['Age',
'Axillary nodes', 'Survival status'])
positive_p_df = pd.DataFrame(positive_p_data, columns=['Age',
'Axillary nodes', 'Survival status'])

# Concatenar para obtener DataFrames de entrenamiento y prueba
training_df = pd.concat([negative_e_df, positive_e_df],
ignore_index=True)
test_df = pd.concat([negative_p_df, positive_p_df], ignore_index=True)
```

```
# Función para calcular la distancia euclidiana
def euclidean_distance(point1, point2):
    sum_squared = sum((p1 - p2) ** 2 for p1, p2 in zip(point1,
point2))
    return sum_squared ** 0.5

# Función para encontrar el vecino más cercano
def nearest_neighbor(training_points, single_test_point):
    closest_distance = float('inf')
    closest_label = None

    for train_point, label in training_points:
        distance = euclidean_distance(train_point, single_test_point)
        if distance < closest_distance:
            closest_distance = distance
            closest_label = label

    return closest_label

# Preparación de datos para la clasificación
training_points = [(row[:2], row[2]) for row in
training_df.itertuples(index=False, name=None)]
test_points = [row[:2] for row in test_df.itertuples(index=False,
name=None)]

# Clasificación de la partición de prueba
predictions = [nearest_neighbor(training_points, test_point) for
test_point in test_points]

# Agregar las clasificaciones (predicciones) al DataFrame de prueba
test_df['Predicted status'] = predictions

# Mostrar el DataFrame actualizado
print(test_df)
```