INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN

Tarea 05

Método de validación Hold-out

Edgar Fernando Espinosa Torres

Clasificación inteligente de patrones

Dr. Cornelio Yáñez Márquez

Ciudad de México, 30 de septiembre de 2023.

Índice general

Índice de tablas

Introducción

Un clasificador inteligente de patrones (cip) consta de cuatro elementos esenciales: el dataset, el método de validación, el algoritmo y la medida para evaluar el desempeño.

Hasta este momento, en el curso de clasificación inteligente de patrones solamente se había explorado un método de validación conocido como partición fija. En éste, los conjuntos de entrenamiento y prueba se eligen de manera arbitraria.

A continuación, se introduce un nuevo método llamado hold-out estratificado, el cual consta de una serie de pasos bien establecidos. En este método, las clases están representadas en los conjuntos de entrenamiento y prueba de manera proporcional y, además, para generar a ambos se utiliza la aleatoriedad. La arbitrariedad que caracteriza al método de partición fija queda fuera de juego.

En esta tarea, se trabaja con los datasets MNIST, Haberman's Survival y Nutt. Es importante destacar que se sigue utilizando el algoritmo del Clasificador Euclidiano. Las implementaciones de los programas realizaron en el lenguaje de programación Python.

En el caso de los datasets balanceados se calculan las medidas para evaluar el desempeño: $accuracy$ y por primera vez, se presenta la medida derivada a partir de ella, $error\ rate$.

Desarrollo y discusión

Propósito de la tarea

- Ejemplificar la aplicación del Clasificador Euclidiano en un dataset famoso del estado del arte que exhibe partición fija.
- Generar particiones 80-20 con el método de validación Hold-out y aplicar el clasificador Euclidiano usando esas particiones.

Parte 1

1.a) Descargar el dataset MNIST en formato csv (se sugiere kaggle).

1.b) Verificar que el dataset E (mnist_train.csv) es balanceado.

1.c) Verificar que el dataset P (mnist_test.csv) es balanceado.

1.d) Entrenar el Clasificador Eulidiano con el conjunto mnist_train.csv.

1.e) Probar el Clasificador Eulidiano en el conjunto mnist_test.csv.

1.f) Reportar accuracy y error rate.

Es importante recordar que un clasificador inteligente de patrones (cip) consta de cuatro elementos básicos:

- Un dataset
- Un método de validación
- Un algoritmo
- Una medida para evaluar el desempeño

1.1 Dataset

Para la parte 1 se utiliza el data set MNIST (uno de los datasets más populares en el área del RP) [1]. Sus siglas significan Modified National Institute of Standards and Technology database, MNIST es un dataset enorme de dígitos manuscritos que se utiliza habitualmente para entrenar diversos sistemas de procesamiento de imágenes.

El dataset original tiene un formato difícil de utilizar para los principiantes. El dataset utilizado es obra de Joseph Redmon quien lo donó en formato CSV.

El archivo mnist_train.csv (conjunto E) contiene 60,000 instancias. El archivo mnist_test.csv (conjunto P) contiene 10,000 instancias. El primer valor es la clase (un número del 0 al 9) y los 784 atributos restantes son los valores de los píxeles (un número del 0 al 255).

1.2 Método de validación

En esta parte se aplica el primer método de validación estudiado durante el curso, el cual tiene el nombre de partición fija, caracterizado porque la partición es hecha mediante una decisión arbitraria. En este caso, el creador del dataset estableció que patrones integran los conjuntos E y P.

1.3 Algoritmo

El algoritmo utilizado es el del Clasificador Euclidiano estudiado e implementado en la tarea 04.

Los resultados obtenidos después de ejecutar el algoritmo son los siguientes:

Patrones clasificados correctamente	**Patrones clasificados incorrectamente**
8203	1797

Tabla 1: Patrones clasificados correctamente vs incorrectamente en el dataset MNIST.

1.4 Medida para evaluar el desempeño

Ya que el dataset es balanceado con $Imbalance\ Ratio$ para el conjunto E de $IR_E = \frac{6742}{5421} = 1.24 \leq 1.5$ y para el conjunto P de $IR_P = \frac{1135}{892} = 1.27 \leq 1.5$ es posible calcular el valor $accuracy = \frac{8203}{10000} = 0.82\ (en\ porcentaje\ 82\%)$ y

$error\ rate = 1 - accuracy = 1 - 0.82 = 0.18\ (en\ porcentaje\ 18\%)$.

Parte 2

2.a) Generar una partición 80-20 con el método de validación Hold-out en el dataset Haberman's Survival Data Set del año 58.

2.b) Aplicar el Clasificador Eulidiano en la partición generada.

2.c) Reportar accuracy y error rate.

2.1 Dataset

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.

2.2 Método de validación

Se utiliza el Hold-out estratificado. A diferencia de la arbitrariedad de la partición fija, en el nuevo método se induce una partición en el conjunto D en dos conjuntos disjuntos: uno de entrenamiento E y uno de prueba P, de modo que los conjuntos E y P formen una partición del dataset D.

Lo que se hace es generar la partición de manera aleatoria, garantizando que todas las clases estén representadas en E y P, de manera proporcional [3].

Paso 1: porcentajes

La partición Hold-out más utilizada en la literatura es 80-20; es decir, 80% de los patrones de cada clase estarán en E y el resto, 20% de los patrones de cada clase, estarán en P. Sin embargo, es posible utilizar otras particiones con diferente porcentaje como 75-25 y 70-30.

Si los resultados de los porcentajes no son números enteros, se puede utilizar un redondeo. En esta tarea, se decidió utilizar el redondeo hacia el entero más cercano con la finalidad de manejar una cantidad entera de patrones.

Paso 2: desordenamiento aleatorio

Para cada una de las clases en D, los patrones se desordenan de manera aleatoria.

Paso 3: formación del conjunto E

Una vez que los patrones de cada clase son apilados conforme a la permutación aleatoria obtenida, se procede a la creación del conjunto E, utilizando los números adquiridos en el primer paso.

Asumamos, sin pérdida de generalidad, que estamos ante un problema biclase (con cardinalidades $N1$ y $N2$) bajo el esquema 80-20 de Hold-out. El número $e1$ es el resultado (redondeado, si es necesario) de calcular el 80% de $N1$, y el número $e2$ es el resultado (también redondeado, si es necesario) de calcular el 80% de $N2$. El conjunto E se establece mediante la unión de dos conjuntos de patrones definidos de la siguiente manera:

a) Los $e1$ patrones iniciales de la pila aleatoria de la clase 1.

b) Los $e2$ patrones iniciales de la pila aleatoria de la clase 2.

Paso 4: formación del conjunto P

Definamos $p1$ como $N1 - e1$, y $p2$ como $N2 - e2$. El conjunto P se construye mediante la unión de dos conjuntos de patrones establecidos de la forma siguiente:

a) Los patrones $p1$ de la clase 1, que son remanentes tras asignar al conjunto E los primeros e1 patrones de la pila aleatoria 1.

b) Los patrones $p2$ de la clase 2, que son remanentes tras asignar al conjunto E los primeros $e2$ patrones de la pila aleatoria 2.

2.3 Algoritmo

El algoritmo utilizado es el del Clasificador Euclidiano, estudiado e implementado en la tarea 04. Ya que el método de validación maneja aleatoriedad, se pueden obtener resultados diferentes en cada ejecución.

Ejecución	Patrones clasificados correctamente	Patrones clasificados incorrectamente
1	3	3
2	4	2
3	5	1

Tabla 2: Patrones clasificados correctamente vs incorrectamente en el dataset Haberman's Survival del año 1958.

2.4 Medida para evaluar el desempeño

Tenemos un dataset desbalanceado con $IR = \frac{20}{8} = 2.50 > 1.50$. Por lo tanto, no es posible calcular $accuracy$ y, en consecuencia, tampoco $error\ rate$.

Parte 3

3.a) Generar las siguientes particiones con el método de validación Hold-out 75-25

en el dataset Nutt (2 clases; 14 patrones por cada clase):

3.b) Aplicar el Clasificador Eulidiano en la partición generada.

3.c) Reportar accuracy y error rate.

3.1 Dataset

Este dataset se generó a partir del trabajo de Nutt en colaboración con muchos 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, mientras que los 14 patrones restantes corresponden a glioblastomas no clásicos. Cada patrón contiene 1070 atributos numéricos, los cuales representan los niveles de intensidad de expresión de los genes [4].

3.2 Método de validación

Se utilizó Hold-out 75-25.

3.3 Algoritmo

El algoritmo utilizado es el del Clasificador Euclidiano, estudiado e implementado en la tarea 04. Ya que el método de validación maneja aleatoriedad, se pueden obtener resultados diferentes en cada ejecución.

Ejecución	Patrones clasificados correctamente	Patrones clasificados incorrectamente
1	6	2
2	5	3
3	6	2

Tabla 3: Patrones clasificados correctamente vs incorrectamente en el dataset Nutt.

3.4 Medida para evaluar el desempeño

Ya que el dataset es balanceado con $IR = \frac{14}{14} = 1.00 \leq 1.50$ es posible calcular el valor $accuracy$ y $error\ rate$ para cada ejecución.

Ejecución	$\boldsymbol{accuracy}$	$\boldsymbol{error\ rate}$
1	0.75	0.25
2	0.625	0.375
3	0.75	0.375

Tabla 4: Accuracy y error rate para el dataset Nutt.

Conclusiones

En el caso del dataset MNIST fue su creador quien decidió cómo formar los conjuntos de prueba y entrenamiento y así, con estos conjuntos se logró un buen valor de $accuracy$.

No obstante, en lo general, la partición fija puede introducir un fuerte sesgo si la manera en la que se forman los conjuntos de entrenamiento y prueba no son suficientemente representativos del dataset.

El método de hold-out estratificado supone una importante mejora respecto al método de partición fija, ya que asegura que ambos conjuntos, el de entrenamiento y el de prueba son representativos de todo el dataset al considerar todas las clases y usar la aleatoriedad.

Sin embargo, hold-out estratificado no mostró tanta $accuracy$ para los datasets utilizados. Como se mencionó en clase, existen otros métodos de validación que pueden brindar un mejor valor para esta medida de desempeño. Por lo tanto, es necesario seguir explorando y estudiando otros métodos que puedan ser más fiables o incluso estables, es decir, que los conjuntos de entrenamiento y prueba no dependan de la aleatoriedad y en que contextos se pueden utilizar.

Referencias bibliográficas

[1] MNIST in CSV. (2018, 19 mayo). Kaggle.
https://www.kaggle.com/datasets/oddrationale/mnist-in-csv

[2] *UCI Machine Learning Repository*. (s. f.-c).
https://archive.ics.uci.edu/dataset/43/haberman+s+survival

[3] Yáñez Márquez, C. (2023). RD 05: Método de validación Hold-out [Diapositivas de clase]. Centro de Investigación en Computación.

[4] 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.

Anexos

```python
#Parte 1
import numpy as np
import pandas as pd

def leer_archivo(nombre_archivo):
    df = pd.read_csv(nombre_archivo, usecols=lambda column : column not in ['label'])
    return df.to_numpy()

def obtener_etiquetas(nombre_archivo):
    labels = pd.read_csv(nombre_archivo, usecols=['label'])
    return labels.to_numpy().flatten()

def calcular_centroide(data):
    return np.mean(data, axis=0)

def distancia_euclidiana(a, b):
    return np.sqrt(np.sum((a - b) ** 2))

def clasificar(vector, centroides):
    distancias = [distancia_euclidiana(vector, centroide) for centroide in
centroides]
    return np.argmin(distancias)

def main():
    entrenamiento_data = leer_archivo('mnist_train.csv')
    entrenamiento_labels = obtener_etiquetas('mnist_train.csv')

    centroides = []
    for label in np.unique(entrenamiento_labels):
        data_label = entrenamiento_data[entrenamiento_labels == label]
        centroide = calcular_centroide(data_label)
        centroides.append(centroide)

    prueba_data = leer_archivo('mnist_test.csv')
    prueba_labels = obtener_etiquetas('mnist_test.csv')
    resultados = [clasificar(vector, centroides) for vector in prueba_data]

    correctos = sum(resultados == prueba_labels)
    incorrectos = len(prueba_data) - correctos

    print(f"Patrones correctamente clasificados: {correctos}")
    print(f"Patrones incorrectamente clasificados: {incorrectos}")

if __name__ == '__main__':
    main()
```

```python
#Parte 2 y 3

import pandas as pd
import numpy as np

# Leer el archivo
def leer_archivo(nombre_archivo):
    return pd.read_excel(nombre_archivo)

# Calcular el centroide
def calcular_centroide(grupo):
    return grupo.mean()

# Distancia Euclidiana
def distancia_euclidiana(a, b):
    return np.sqrt(np.sum((a - b) ** 2))

# Clasificar las instancias
def clasificar(instancia, centroides):
    distancias = {etiqueta: distancia_euclidiana(instancia, centroide) for etiqueta,
centroide in centroides.items()}
    return min(distancias, key=distancias.get)

# Dividir los datos
def dividir_datos(df, test_size):
    etiquetas = df['label'].unique()
    entrenamiento, prueba = pd.DataFrame(), pd.DataFrame()
    for etiqueta in etiquetas:
        grupo = df[df['label'] == etiqueta].sample(frac=1).reset_index(drop=True)
        limite = int(len(grupo) * (1 - test_size))
        entrenamiento = pd.concat([entrenamiento, grupo.iloc[:limite]])
        prueba = pd.concat([prueba, grupo.iloc[limite:]])
    return entrenamiento, prueba

# Función Principal
def main():
    df = leer_archivo('Nutt_classic_nonclassic_excel.xlsx')
    test_size = float(input("Ingrese el tamaño del conjunto de prueba (por ejemplo,
0.2 para el 20%): "))
    entrenamiento, prueba = dividir_datos(df, test_size)

    centroides = {etiqueta: calcular_centroide(grupo.drop(columns=['label'])) for
etiqueta, grupo in entrenamiento.groupby('label')}

    predicciones = [clasificar(instancia.drop('label'), centroides) for index,
instancia in prueba.iterrows()]

    # Imprimir patrones de prueba, clase predicha y clase real
    for (index, instancia), predicción in zip(prueba.iterrows(), predicciones):
        etiqueta_real = instancia['label']
```

```
        print(f"Patrón de Prueba: {instancia.drop('label').to_list()}")
        print(f"Clase Predicha: {predicción}")
        print(f"Clase Real: {etiqueta_real}")
        if predicción == etiqueta_real:
          print("CORRECTO")
        else:
          print("INCORRECTO")
        print("-----")

    comparacion = np.array(predicciones) == prueba['label'].to_numpy()
    correctos = np.sum(comparacion)
    incorrectos = len(prueba) - correctos

    print(f"Patrones correctamente clasificados: {correctos}")
    print(f"Patrones incorrectamente clasificados: {incorrectos}")

if __name__ == '__main__':
    main()
```