INSTITUTO POLITÉCNICO NACIONAL

ESCOM

INTELIGENCIA ARTIFICIAL

LAB 9: EUCLIDIANO & 1NN

MARTÍNEZ CHÁVEZ JORGE ALEXIS

6CV3

14 NOVIEMBRE 2024

Introducción

Una de las medidas más comunes para calcular la distancia entre dos puntos en un es la distancia Euclidiana la cual mide la longitud en línea recta entre estos dos puntos; El clasificador 1-NN es una variante del algoritmo k-NN donde el valor de k se fija en 1, para cada punto de prueba, el algoritmo simplemente busca el punto de entrenamiento más cercano y le asigna la ese punto vecino.

Para los metodos de validación, el hold out divide aleatoriamente el conjunto de datos uno para entrenamiento y otro para prueba de forma 70% de los datos para entrenar el modelo y el 30% restante para evaluar su desempeño. Para el cross validation se divide el conjunto de datos en k pliegues o subgrupos y se evalúa cada uno de ellos entrenando con los restantes, repitiendo este proceso k veces para asegurar que cada punto de datos sea utilizado para prueba al menos una vez. Leave-One-Out es una forma extrema de validación cruzada en la que, para cada instancia del conjunto de datos, se utiliza un único punto de prueba y el resto de los puntos se utilizan para entrenar el modelo.

Desarrollo

Esta practica la hicimos en dos códigos, el primero para el clasificador euclidiano, a continuación debemos de cargar lo datos del dataset, para ello lo vamos a hacer desde la ruta en donde importamos el archivo data del iris.

```python
# Cargar el archivo con la ruta especificada en el código
def cargar_dataset():
    file_path = "C:/Users/jorge/OneDrive/Documentos/JORGE/ESCOM/9_SEMESTRE/IA/LABORATORIO/10_KNN_BAYES/bezdekIris.data"
    try:
        datos = pd.read_csv(file_path, header=None)
        return datos
    except Exception as e:
        print(f"Error al cargar el archivo: {e}")
        return None
```

Ahora debemos de calcular la distancia euclidianna entre 2 vectores, que es la raiz de la suma de los cuadrados de la diferencia de cada uno de estos pares de vectores.

```python
# Calcular la distancia euclidiana
def distancia_euclidiana(x1, x2):
    return np.sqrt(np.sum((x1 - x2) ** 2))
```

Ahora para hacer el clasificador euclidiano, priemero se calcula la distancia de cada uno de los puntos prueba con los puntos de entrenamiento depues se calcula el inidce del punto mas cercano y se agraga a la lista de puntos de entrenamiento mas cercanos para las predicciones y asi se hace con todos los puntos.

```python
# Clasificador basado en distancia euclidiana
def clasificador_euclidiano(X_train, y_train, X_test):
    y_pred = []
    for test_point in X_test:
        # Calcular las distancias entre el punto de prueba y todos los puntos de entrenamiento
        distancias = [distancia_euclidiana(test_point, x_train) for x_train in X_train]
        # Encontrar el índice del punto más cercano
        closest_index = np.argmin(distancias)
        # Predecir la clase del punto más cercano
        y_pred.append(y_train[closest_index])
    return np.array(y_pred)
```

Primero vamos a hacer el de Hold Out, en donde debemos de dividri 70/30 para entrenar el modelo y probar el modelo respectivamente, una vez hecha la división debemos de hacer el entrenamiento y después hacer la predicción. Por ultimo calculamos el accuracy y la matriz de confusión.

```python
# Métodos de validación
def hold_out(modelo, X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
    modelo.fit(X_train, y_train)
    y_pred = modelo.predict(X_test)
    return accuracy_score(y_test, y_pred), confusion_matrix(y_test, y_pred)
```

Ahora hacemos para cross validation, en donde el conjunto de datos lo vamos a subdividir en 10 subconjuntos, entrenamos en 9 de ellos y el sobrante lo vamos a dejar como prueba, y este proceso lo vamos a repetir 10 veces para que almenos una vez cada subconjunto se el subconjunto de prueba. Se calcula el promedio de las precisiones obtenidas en las 10 iteraciones para obtener una evaluación más estable y generalizable. Finalmente de la misma forma calculamos su accuracy y lal matriz de confusión.

```python
def cross_validation(modelo, X, y, folds):
    kf = KFold(n_splits=folds, shuffle=True, random_state=42)
    scores = cross_val_score(modelo, X, y, cv=kf)
    modelo.fit(X, y)
    y_pred = modelo.predict(X)
    conf_matrix = confusion_matrix(y, y_pred)
    return np.mean(scores), conf_matrix
```

Ahora lo hacemos para el leave one out, en donde se divide en tantas particiones como instancias tenga el dataset, se entrena el modelo con todos los datos menos una única instancia. Y se evalúa el modelo en esa instancia excluida. De la misma forma calculamos su accuracy y lal matriz de confusión.

```python
def leave_one_out(modelo, X, y):
    loo = LeaveOneOut()
    y_true, y_pred = [], []
    for train_idx, test_idx in loo.split(X):
        modelo.fit(X[train_idx], y[train_idx])
        pred = modelo.predict(X[test_idx])
        y_true.extend(y[test_idx])
        y_pred.extend(pred)
    return accuracy_score(y_true, y_pred), confusion_matrix(y_true, y_pred)
```

Por ultimo simplemente ejecutamos cada uno de los metodos de validación y calculamos su matriz de confusión y el accuracy para despues visualizar los datos.

```python
# Ejecutar clasificador Euclidiano
def ejecutar_clasificador_euclidiano(X, y):
    validaciones = {
        "Hold-Out (70/30)": hold_out,
        "10-Fold Cross-Validation": lambda m, X, y: cross_validation(m, X, y, 10),
        "Leave-One-Out": leave_one_out
    }

    resultados = []
    for metodo, funcion_validacion in validaciones.items():
        accuracy, matriz_confusion = funcion_validacion(clasificador_euclidiano, X, y)
        resultados.append((metodo, accuracy, matriz_confusion))
```

Con el sigueinte resultado en pantalla, en donde nos dice que para hold out se tuvo un accuracy del 100% en donde la primera clase tuvi 19 isntancias con 0 errores, en la clase 2 tuvo 13 instancias con 0 errores y por ultimo en la clase 3 tuvo 13 instancias con 13 errores. Para el cross validation en la promera clase tuvo 50 instancias con 0 errores en la segunda clase tuvo 47 instancias con 3 errores de confusion con la clase 3 y en la clase 3 tuvo 47 instancias con 3 errores de confusion con la clase 2 lo que da

un accuracy del 96%. Para leave one out se tuvo un acurracy del 96% en donde en la primera clase tuvo una 50 instancias con 0 errores, en la segunda clase tuvo 47 instancias con 3 errores de confusion con la clase 3 y enla clase 3 tuvo 47 instancias con 3 errores de confusion con la clase 2.

```
Método de Validación: Hold-Out (70/30)
Accuracy: 1.00
Matriz de Confusión:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Método de Validación: 10-Fold Cross-Validation
Accuracy: 0.96
Matriz de Confusión:
[[5.  0.  0. ]
 [0.  4.7 0.3]
 [0.  0.3 4.7]]
Método de Validación: Leave-One-Out
Accuracy: 0.96
Matriz de Confusión:
[[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]
PS C:\Users\jorge\OneDrive\Documentos\JORGE\ESCOM\9 SEMESTRE>
```

Para la segunda parte de la practica utilizamos todo el codigo anterior excepto la parte del clasificador dado que ahora lo haremos por 1nn, es un clasificador de kneighbors en donde solo le se asigna 1 de forma que solo se pueda ver al vecino mas cercano, despues se entrena al modelo con los datos de entrenameinto y se realizan las pruebas de predicciones.

```python
# Clasificador 1-NN usando KNeighborsClassifier de scikit-learn
def clasificador_1nn(X_train, y_train, X_test):
    # Creamos el clasificador 1-NN
    model = KNeighborsClassifier(n_neighbors=1)
    model.fit(X_train, y_train)  # Entrenamos el modelo con los datos de entrenamiento
    y_pred = model.predict(X_test)  # Hacemos predicciones en el conjunto de prueba
    return y_pred
```

Ahora cuando hacemos la prueba de ejecución obtenemos lo sigueintes resultados en donde para hold out hubo 19 isntancias con 0 errores, en la clase 2 tuvo 13 instancias con 0 errores y por ultimo en la clase 3 tuvo 13 instancias con 13 errores lo que nos da

un accuracy del 100%. Para el cross validation en la promera clase tuvo 50 instancias con 0 errores en la segunda clase tuvo 47 instancias con 3 errores de confusion con la clase 3 y en la clase 3 tuvo 47 instancias con 3 errores de confusion con la clase 2 lo que da un accuracy del 96%. Para leave one out se tuvo un acurracy del 96% en donde en la primera clase tuvo una 50 instancias con 0 errores, en la segunda clase tuvo 47 instancias con 3 errores de confusion con la clase 3 y enla clase 3 tuvo 47 instancias con 3 errores de confusion con la clase 2.

```
Método de Validación: Hold-Out (70/30)
Accuracy: 1.0000
Matriz de Confusión:
[[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
Método de Validación: 10-Fold Cross-Validation
Accuracy: 96.0000
Matriz de Confusión:
[[5.  0.  0. ]
 [0.  4.7 0.3]
 [0.  0.3 4.7]]
Método de Validación: Leave-One-Out
Accuracy: 0.9600
Matriz de Confusión:
[[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]
PS C:\Users\jorge\OneDrive\Documentos\JORGE\ESCOM\9_SEMESTRE>
```

Enlace

A continuación el enlace donde se encuentra el repositorio de todas las practicas, esta en especifico se encuentra en la carpeta de lab 9
https://github.com/Jorge300403/IA_6CV3_MartinezChavez/tree/main/LAB_9_EUCLIDIANOS_1NN