INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INGENIERÍA EN SISTEMAS COMPUTACIONALES

Inteligencia Artificial

Laboratorio 11: Redes Neuronales

Marco Isaí Flores Vicencio

Jesús Pérez González

Sebastián Reyes Núñez

Grupo:

6CV3

Introducción

Durante esta práctica, se trabajará con dos tipos de redes neuronales: el Perceptrón Multicapa (MLP) y la Red Neuronal de Función de Base Radial (RBF). Estas redes serán validadas utilizando tres métodos principales: Hold-Out (70/30), Validación Cruzada de 10 Pliegues y Leave-One-Out. Además, se medirán métricas esenciales como la exactitud y la matriz de confusión para evaluar el rendimiento de los clasificadores.

El laboratorio se llevará a cabo utilizando tres conjuntos de datos: el conocido dataset Iris Plant y otros dos de libre elección. Esto permitirá explorar la generalización de las redes neuronales en distintos dominios, resaltando la importancia de elegir modelos y validaciones adecuadas para tareas de clasificación específicas.

Marco Teórico

Redes Neuronales

Las redes neuronales son modelos computacionales diseñados para simular el comportamiento de las neuronas del cerebro humano. Estas estructuras están compuestas por capas de nodos o neuronas interconectadas que procesan datos de manera jerárquica. Existen diferentes arquitecturas de redes neuronales, cada una con capacidades y aplicaciones específicas.

El perceptrón multicapa es una de las arquitecturas más comunes y se basa en una estructura de capas: una capa de entrada, una o más capas ocultas, y una capa de salida. Las neuronas en cada capa aplican funciones de activación no lineales, lo que permite a la red aprender patrones complejos. El MLP utiliza el algoritmo de retropropagación para ajustar los pesos de sus conexiones durante el entrenamiento.

La Red Neuronal de Función de Base Radial se basa en funciones de base radial como activaciones en sus nodos. Las RBF son especialmente útiles en problemas de clasificación y regresión debido a su capacidad para generar decisiones basadas en proximidades geométricas. Su arquitectura incluye una capa de entrada, una capa oculta donde las funciones radiales transforman los datos, y una capa de salida para la predicción.

Métodos de Validación

La validación de los modelos de aprendizaje automático es crucial para evaluar su capacidad de generalización. En esta práctica se utilizan tres métodos comunes:

El primer método de validación es el Hold-Out (70/30), el cual consiste en dividir el conjunto de datos en dos partes: un 70% para entrenamiento y un 30% para prueba. Es un método simple y eficiente, aunque puede introducir sesgos si la división no es representativa del problema.

El segundo método que se usa en el desarrollo de esta práctica es el 10-Fold Cross-Validation. El conjunto de datos se divide en 10 partes iguales, utilizando 9 de ellas para entrenar y una para

probar. Este proceso se repite 10 veces, alternando las particiones. Es una técnica robusta que reduce la varianza en la evaluación.

Finalmente, el método Leave-One-Out consiste de que cada instancia del conjunto de datos se utiliza como conjunto de prueba una vez, mientras que el resto se emplea para el entrenamiento. Aunque es computacionalmente más costoso, proporciona una evaluación precisa del modelo.

Métricas de Desempeño

Para medir la calidad de los clasificadores, se utilizará la exactitud y la matriz de confusión.

La exactitud es la proporción de predicciones correctas realizadas por el modelo en comparación con el total de instancias. Se calcula como:

$$Accuracy = \frac{Número\ de\ predicciones\ correctas}{Total\ de\ predicciones\ realizadas}$$

La Matriz de Confusión es una tabla que permite visualizar el desempeño del modelo al comparar las predicciones realizadas con las verdaderas etiquetas de los datos. Proporciona información detallada sobre errores y aciertos en cada clase.

Desarrollo

Clasificador PM

El código empieza con la importación de la librerías necesarias para trabajar con el dataset de Iris, entrenar y evaluar un modelos de Perceptrón Multicapa y para la visualización de resultados. Estas librerías incluyen herramientas de manipulación de datos, partición de conjuntos, cálculo de métricas, entrenamiento del modelo y visualización.

```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split, cross_val_score, LeaveOneOut
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.neural_network import MLPClassifier
import matplotlib.pyplot as plt
import seaborn as sns
```

Después, se establece una semilla aleatoria para garantizar la reproducibilidad de resultados al dividir datos o entrenar modelos. Así, cualquier operación aleatoria, como la división del dataset, siempre generan los mismos resultados.

```
np.random.seed(42)
```

Se procede a cargar el dataset Iris de la librería 'sklearn', y sus datos y etiquetas se asignan a las variables 'x' y 'y'.

```
iris = load_iris()
X = iris.data
y = iris.target
```

El código procede a hacer la división del dataset en un 70/30. El 70% del dataset se utilizará para entrenar el modelo y el 30% para la prueba del modelo. Esta técnica de 'Hold-Out' permite evaluar el modelo con datos que no ha visto previamente en su entrenamiento.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Se inicializa un Perceptrón Multicapa con dos capas ocultas de 10 neuronas cada una y un máximo de 1000 iteraciones. Luego, se entrena con el conjunto de entrenamiento. El modelo aprende patrones en los datos de entrenamiento para predecir la especie de flor.

```
mlp = MLPClassifier(hidden_layer_sizes=(10, 10), max_iter=1000, random_state=42)
mlp.fit(X_train, y_train)
```

El modelo predice las etiquetas del conjunto de prueba, y se calcula la precisión de las predicciones. Esto mide qué tan bien el modelo generaliza para datos nuevos.

```
y_pred = mlp.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Hold-Out Accuracy (MLP):", accuracy)
```

Después, se calcula y visualiza la matriz de confusión para analizar los errores de clasificación. Esta matriz muestra cómo se distribuyen las predicciones frente a las etiquetas reales.

```
cm = confusion_matrix(y_test, y_pred)
print("Matriz de confusión (MLP):\n", cm)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Matriz de Confusión (MLP)")
plt.xlabel("Predicción")
plt.ylabel("Actual")
plt.show()
```

Se utiliza validación cruzada de 10 particiones para evaluar la estabilidad y generalización del modelo. Esto divide los datos en 10 subconjuntos, entrenando en 9 y evaluando en 1 en cada iteración.

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(mlp, X, y, cv=10)
print("10-Fold Cross-Validation Accuracy (MLP):", scores.mean())
```

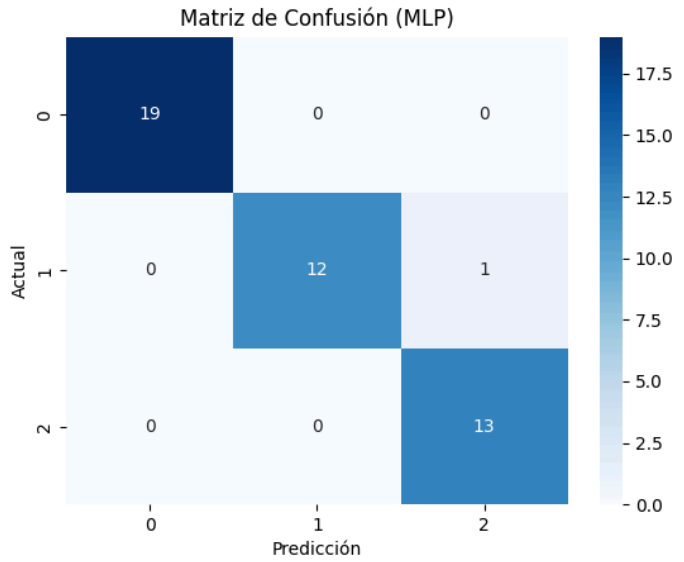
Se emplea validación Leave-One-Out para evaluar el modelo, entrenándolo con todos los datos excepto uno en cada iteración. Esta técnica ofrece una evaluación exhaustiva pero es computacionalmente costosa.

```
loo = LeaveOneOut()
scores_loo = cross_val_score(mlp, X, y, cv=loo)
print("Leave-One-Out Accuracy (MLP):", scores_loo.mean())
```

Resultados

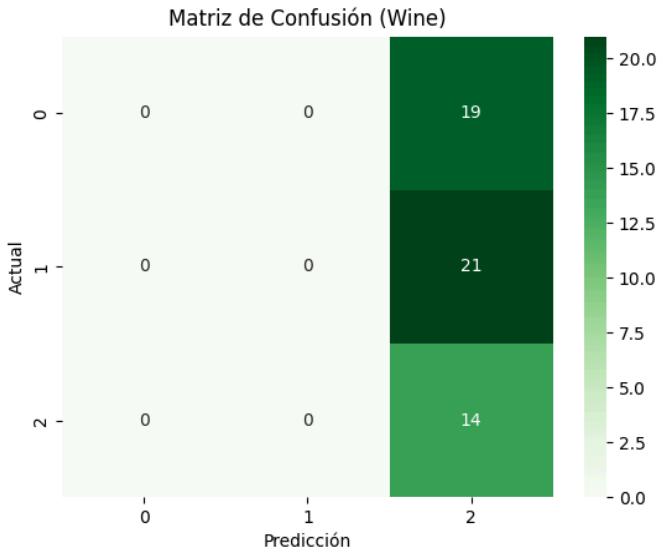
Dataset Iris

```
Hold-Out Accuracy (MLP): 0.9777777777777777
Matriz de confusión (MLP):
 [[19  0  0]
 [ 0 12  1]
 [ 0  0 13]]
10-Fold Cross-Validation Accuracy (MLP): 0.9600000000000002
Leave-One-Out Accuracy (MLP): 0.96
```



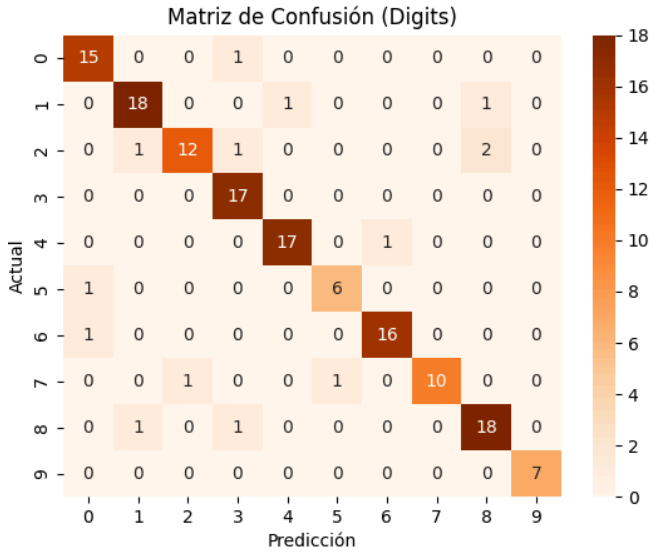
Dataset Wine

```
Hold-Out Accuracy (Wine): 0.25925925925925924
Matriz de Confusión (Wine):
 [[ 0  0 19]
 [ 0  0 21]
 [ 0  0 14]]
10-Fold Cross-Validation Accuracy (Wine): 0.26928104575163403
Leave-One-Out Accuracy (Wine): 0.2696629213483146
```



Dataset Digits

```
Hold-Out Accuracy (Digits): 0.9066666666666666
Matriz de Confusión (Digits):
 [[15  0  0  1  0  0  0  0  0  0]
 [ 0 18  0  0  1  0  0  0  1  0]
 [ 0  1 12  1  0  0  0  0  2  0]
 [ 0  0  0 17  0  0  0  0  0  0]
 [ 0  0  0  0 17  0  1  0  0  0]
 [ 1  0  0  0  0  6  0  0  0  0]
 [ 1  0  0  0  0  0 16  0  0  0]
 [ 0  0  1  0  0  1  0 10  0  0]
 [ 0  1  0  1  0  0  0  0 18  0]
 [ 0  0  0  0  0  0  0  0  0  7]]
10-Fold Cross-Validation Accuracy (Digits): 0.9319999999999998
Leave-One-Out Accuracy (Digits): 0.934
```



Red Neuronal RBF

El código comienza importando las librerías necesarias para cargar datos, transformar características con un kernel RBF, entrenar un clasificador, evaluar el modelo y generar visualizaciones.

```
from sklearn.datasets import load_iris
from sklearn.kernel_approximation import RBFSampler
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split, cross_val_score, LeaveOneOut
from sklearn.metrics import accuracy_score, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

El dataset Iris se carga desde 'sklearn' y se descompone en variables de entrada 'x' y etiquetas de salida 'y'.

```
iris = load_iris()
X, y = iris.data, iris.target
```

Se divide el dataset en un 70% para entrenamiento y 30% para prueba mediante 'train_test_split'. Esto asegura que el modelo sea evaluado con datos no vistos durante el entrenamiento.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

Se crea un transformador RBF para mapear las características originales a un espacio de mayor dimensión, y se aplica a los datos de entrenamiento y prueba. El parámetro 'gamma' controla la amplitud de las funciones RBF, afectando la complejidad de la transformación.

```
rbf_feature = RBFSampler(gamma=1, random_state=42)
X_train_rbf = rbf_feature.fit_transform(X_train)
X_test_rbf = rbf_feature.transform(X_test)
```

Se utiliza un clasificador SGD para entrenar el modelo con las características transformadas. SGD es un clasificador lineal optimizado que funciona bien con datos transformados por kernels.

```
clf_rbf = SGDClassifier(random_state=42)
clf_rbf.fit(X_train_rbf, y_train)
```

El modelo predice las etiquetas del conjunto de prueba transformado, y se calcula la precisión de las predicciones. Esto mide la efectividad del modelo al generalizar en datos nuevos.

```
y_pred = clf_rbf.predict(X_test_rbf)
accuracy = accuracy_score(y_test, y_pred)
print("Hold-Out Accuracy (Iris RBF):", accuracy)
```

Se calcula y visualiza la matriz de confusión para entender cómo se distribuyen los errores de clasificación.

```
cm = confusion_matrix(y_test, y_pred)
print("Matriz de Confusión (Iris RBF):\n", cm)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title("Matriz de Confusión (Iris RBF)")
plt.xlabel("Predicción")
plt.ylabel("Actual")
plt.show()
```

Se realiza validación cruzada de 10 particiones en todo el conjunto de datos para evaluar la estabilidad del modelo. Esto mide la precisión promedio del modelo en diferentes divisiones del conjunto de datos.

```
X_rbf = rbf_feature.fit_transform(X)
scores = cross_val_score(clf_rbf, X_rbf, y, cv=10)
print("10-Fold Cross-Validation Accuracy (Iris RBF):", scores.mean())
```

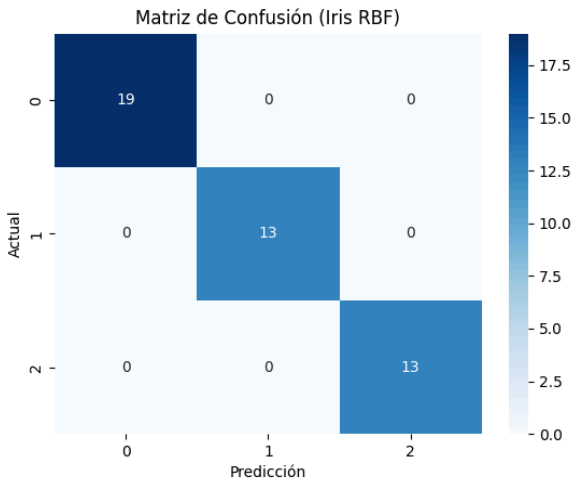
Se realiza una evaluación exhaustiva utilizando Leave-One-Out, dejando un dato fuera para prueba en cada iteración. Aunque computacionalmente es más costoso, esto ofrece una evaluación precisa del rendimiento del modelo.

```
loo = LeaveOneOut()
scores_loo = cross_val_score(clf_rbf, X_rbf, y, cv=loo)
print("Leave-One-Out Accuracy (Iris RBF):", scores_loo.mean())
```

Resultados

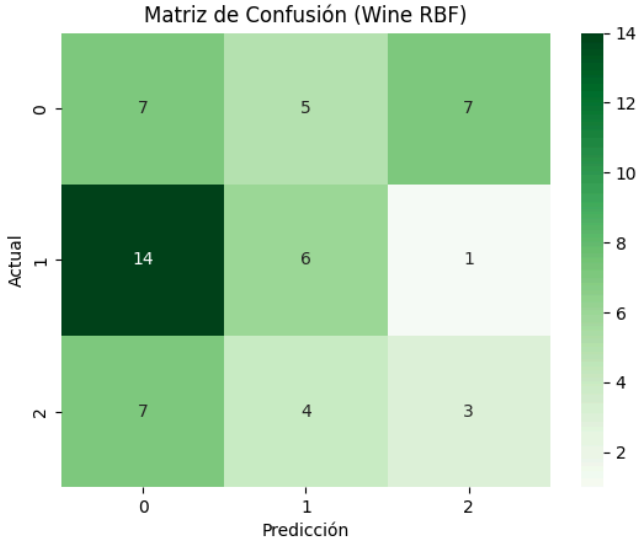
Dataset Iris

```
Hold-Out Accuracy (Iris RBF): 1.0
Matriz de Confusión (Iris RBF):
 [[19  0  0]
 [ 0 13  0]
 [ 0  0 13]]
10-Fold Cross-Validation Accuracy (Iris RBF): 0.9533333333333334
Leave-One-Out Accuracy (Iris RBF): 0.9533333333333334
```



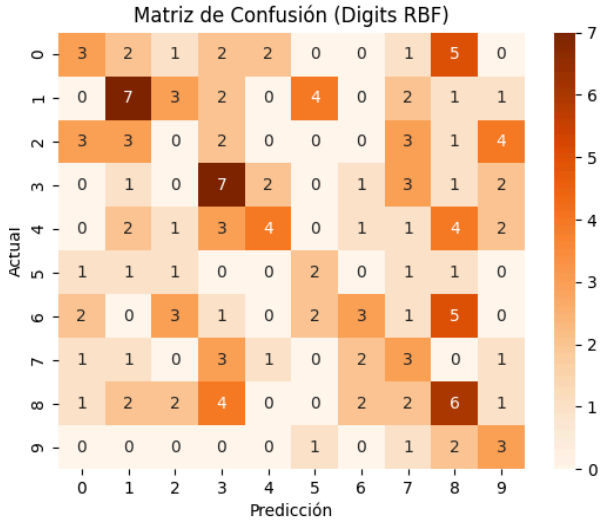
Dataset Wine

```
Hold-Out Accuracy (Wine RBF): 0.2962962962962963
Matriz de Confusión (Wine RBF):
 [[ 7  5  7]
 [14  6  1]
 [ 7  4  3]]
10-Fold Cross-Validation Accuracy (Wine RBF): 0.330718954248366
Leave-One-Out Accuracy (Wine RBF): 0.33707865168539325
```



Dataset Digits

```
Hold-Out Accuracy (Digits RBF): 0.25333333333333335
Matriz de Confusión (Digits RBF):
 [[3 2 1 2 2 0 0 1 5 0]
 [0 7 3 2 0 4 0 2 1 1]
 [3 3 0 2 0 0 0 3 1 4]
 [0 1 0 7 2 0 1 3 1 2]
 [0 2 1 3 4 0 1 1 4 2]
 [1 1 1 0 0 2 0 1 1 0]
 [2 0 3 1 0 2 3 1 5 0]
 [1 1 0 3 1 0 2 3 0 1]
 [1 2 2 4 0 0 2 2 6 1]
 [0 0 0 0 0 1 0 1 2 3]]
10-Fold Cross-Validation Accuracy (Digits RBF): 0.24400000000000005
Leave-One-Out Accuracy (Digits RBF): 0.248
```



Conclusiones

En esta práctica se implementaron y evaluaron dos tipos de redes neuronales: el Perceptrón Multicapa y la Red Neuronal de Función de Base Radial , utilizando tres conjuntos de datos diferentes: Iris, Wine y Digits. A través de este laboratorio, se exploraron las capacidades de estas arquitecturas para resolver problemas de clasificación y su desempeño bajo distintas estrategias de validación. Los resultados obtenidos evidenciaron que cada modelo tiene fortalezas específicas dependiendo de las características del dataset y los parámetros de configuración utilizados. El Perceptrón Multicapa demostró ser versátil y adecuado para manejar datos con estructuras no lineales, mientras que las Redes RBF sobresalieron en problemas donde la proximidad geométrica entre puntos es crucial para una correcta clasificación.

La comparación de los métodos de validación (Hold-Out, 10-Fold Cross-Validation y Leave-One-Out) permitió destacar la importancia de elegir el enfoque adecuado según el tamaño y las características del conjunto de datos. Mientras que Hold-Out es eficiente para conjuntos grandes, Leave-One-Out proporciona una evaluación más precisa a costa de mayor costo computacional. Por último, la medición de métricas como la precisión y el análisis de la matriz de confusión proporcionaron una visión más clara del desempeño de los modelos, permitiendo identificar patrones de errores y áreas de mejora.

Referencias

Buitrago, B. (2021, 16 diciembre). Redes neuronales — Perceptrón Multicapa i - iWannaBeDataDriven - Medium. *Medium*. https://medium.com/iwannabedatadriven/redes-neuronales-perceptr%C3%B3n-multicapa-i-d8c05a88857e

¿Cuál es el método de retención en la minería de datos? (2023, 25 septiembre). www.linkedin.com. https://es.linkedin.com/advice/1/what-holdout-method-data-mining-skills-data-mining?lang=es#:~:text=El%20m%C3%A9todo%20holdout%20es%20una,y%20un%20conjunto%20de%20prueba.

GeeksforGeeks. (2023, 20 abril). *LOOCV (Leave One Out CrossValidation) in R Programming*. GeeksforGeeks. https://www.geeksforgeeks.org/loocvleave-one-out-cross-validation-in-r-programming/

GeeksforGeeks. (2024, 7 agosto). *Cross Validation in Machine Learning*. GeeksforGeeks. https://www.geeksforgeeks.org/cross-validation-machine-learning/

¿Qué es una red neuronal? - Explicación de las redes neuronales artificiales - AWS. (s. f.). Amazon Web Services, Inc. https://aws.amazon.com/es/what-is/neural-network/

Redes de funcion de base radial (RBF). (s. f.). https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x185.html

SPSS Statistics Subscription - Classic. (s. f.-a). https://www.ibm.com/docs/es/spss-statistics/saas?topic=networks-multilayer-perceptron

SPSS Statistics Subscription - Classic. (s. f.-b). https://www.ibm.com/docs/es/spss-statistics/saas?topic=networks-radial-basis-function