Use of Machine Learning Algorithms for Interference Detection and Suppression in Communication Networks

Ing. Michael Corrales Oviedo, MS.c.
Estudiante de Posgrado
Escuela de Ingeniería en Sistemas de Computación
Universidad San Isidro Labrador
San José, Costa Rica
Mcorovi27@gmail.com

***Abstract* — In the current era of telecommunications, interference in communication networks represents a significant challenge that affects service quality and operational efficiency. This article explores the use of machine learning algorithms for interference detection and suppression in communication networks. A comprehensive review of existing techniques is conducted, and a methodological framework based on machine learning is proposed to address this problem. Through the analysis and implementation of different algorithms, such as neural networks, support vector machines (SVM), and clustering algorithms, the aim is to improve the accuracy and efficiency in identifying and mitigating interference. Experimental results demonstrate that the application of these techniques not only enhances service quality but also optimizes network resource utilization, offering a robust and adaptive solution to contemporary challenges in telecommunications.**

***Keywords* — *Machine learning, interference, communication networks, detection, suppression, service quality*.**

I. INTRODUCCIÓN

El crecimiento exponencial de dispositivos conectados y el aumento de la demanda de servicios de alta calidad en las redes de comunicación han intensificado la necesidad de gestionar eficazmente la interferencia. La interferencia, causada por diversos factores como dispositivos electrónicos, otras redes de comunicación y condiciones ambientales, puede degradar significativamente la calidad del servicio y la eficiencia operativa de una red. Tradicionalmente, se han utilizado métodos basados en reglas y heurísticas para detectar y suprimir interferencias, sin embargo, estos métodos a menudo carecen de la capacidad de adaptarse a las complejidades y dinámicas cambiantes de las redes modernas.

Con el avance de la inteligencia artificial y el aprendizaje automático, surge una oportunidad para abordar este desafío de manera más eficaz. Los algoritmos de aprendizaje automático tienen la capacidad de aprender y adaptarse a patrones complejos en los datos, lo que los convierte en candidatos ideales para la detección y supresión de interferencias en redes de comunicación. Estos algoritmos pueden analizar grandes volúmenes de datos en tiempo real, identificar patrones de interferencia y tomar decisiones proactivas para mitigar su impacto.

Este artículo presenta un enfoque integral para la utilización de algoritmos de aprendizaje automático en la detección y supresión de interferencias. Se explorarán diversas técnicas de aprendizaje automático, incluyendo redes neuronales, máquinas de soporte vectorial (SVM), y algoritmos de clustering, evaluando su eficacia en diferentes escenarios de interferencia. Se discutirá el proceso de recopilación y preprocesamiento de datos, la implementación y entrenamiento de modelos, y la evaluación de su rendimiento.

El objetivo de este estudio es demostrar que los algoritmos de aprendizaje automático pueden proporcionar una solución más precisa y eficiente

para la gestión de interferencias en redes de comunicación, mejorando así la calidad del servicio y optimizando el uso de los recursos de red. Los resultados obtenidos de esta investigación no solo contribuirán al avance del conocimiento en el campo de las telecomunicaciones, sino que también ofrecerán una base sólida para futuras investigaciones y aplicaciones prácticas en la industria.

II. METODOLOGÍA

Primeramente, es definir claramente el problema de la interferencia en redes de comunicación y los objetivos específicos de nuestra investigación. La interferencia puede degradar la calidad del servicio y la eficiencia de la red, provocando latencia, pérdida de paquetes y disminución del rendimiento general. Nuestro objetivo es desarrollar y evaluar algoritmos de aprendizaje automático que puedan detectar y suprimir estas interferencias de manera efectiva, mejorando así la calidad del servicio y optimizando el uso de los recursos de la red.

Seguidamente, para abordar el problema de la interferencia, seleccionamos una variedad de algoritmos de aprendizaje automático basados en su capacidad para manejar grandes volúmenes de datos y detectar patrones complejos. Los algoritmos seleccionados incluyen:

Redes Neuronales Artificiales (ANN): Utilizadas por su capacidad de modelar relaciones no lineales y adaptarse a patrones complejos.

Máquinas de Soporte Vectorial (SVM): Eficaces en la clasificación de datos y la detección de anomalías.

Algoritmos de Clustering: Como K-means y DBSCAN, que son útiles para identificar patrones de interferencia agrupando datos similares.

Bosques Aleatorios (Random Forests): Para la clasificación y regresión, proporcionando alta precisión y manejando la multicolinealidad de los datos.

Se recopilan datos que se obtienen de diversas fuentes, incluyendo registros de la red, métricas de calidad del servicio, y registros de interferencia.

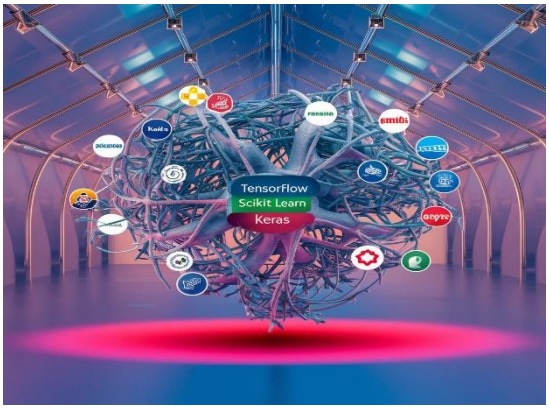
Como siguiente punto, los algoritmos seleccionados se implementaron utilizando librerías de aprendizaje automático como TensorFlow, Scikit-learn y Keras. Cada modelo fue entrenado utilizando el conjunto de datos preprocesados, y se aplicaron técnicas de validación cruzada para evaluar su rendimiento.

Seguidamente, se realiza la evaluación del rendimiento de los modelos, utilizando varias métricas, incluyendo precisión, recall, F1-score y el área bajo la curva ROC (AUC-ROC). Estas métricas nos permiten comparar la eficacia de los diferentes algoritmos y seleccionar el más adecuado para la detección y supresión de interferencias.

Posteriormente, tras la evaluación y selección del modelo más efectivo, se procede a implementar el algoritmo en un entorno de red real para validar su rendimiento en condiciones prácticas. Se monitorea el desempeño del modelo en tiempo real, ajustando y optimizando según sea necesario para asegurar su eficacia en la detección y supresión de interferencias.

Finalmente, se realiza un análisis estadístico exhaustivo de los resultados obtenidos para identificar patrones clave y evaluar el impacto de los algoritmos de aprendizaje automático en la mejora de la calidad del servicio y la eficiencia de la red.

Figura 1: Herramientas utilizadas de aprendizaje automático



Fuente: Elaboración AI

En la Figura 1, se presentan una serie de herramientas utilizadas para el aprendizaje automático de los algoritmos.

III. DESARROLLO

En esta sección, se describe detalladamente el proceso de implementación de los algoritmos de aprendizaje automático seleccionados para la detección y supresión de interferencias en redes de comunicación. Se incluyen los pasos desde la recopilación de datos hasta la evaluación de los modelos, ilustrando cada fase con ejemplos específicos.

Para este estudio, se recopilaron datos de una red de telecomunicaciones operativa durante un período de tres semanas. Los datos incluían métricas de calidad de servicio (QoS) como latencia, pérdida de paquetes y fluctuaciones de señal, así como registros de eventos de interferencia.

- Limpieza de Datos: Se eliminaron registros incompletos y valores atípicos. Por ejemplo, se detectaron y eliminaron los datos de picos anómalos de latencia que no correspondían a eventos de interferencia conocidos.
- Normalización: Las métricas de QoS se normalizaron a una escala de 0 a 1 para asegurar que todas las características tuvieran un peso equivalente en los modelos. Por ejemplo, la latencia se normalizó dividiendo cada valor entre el valor máximo registrado.
- División de Datos: Los datos se dividieron en un 70% para entrenamiento y un 30% para prueba. Esto aseguró que los modelos fueran validados con datos no vistos durante el entrenamiento.

Se implementaron y entrenaron varios algoritmos de aprendizaje automático utilizando el entorno de programación Python y las librerías TensorFlow y Scikit-learn.

1. Redes Neuronales Artificiales (ANN):
   - Arquitectura: Se diseñó una red con tres capas ocultas, cada una con 128, 64 y 32 neuronas, respectivamente.
   - Entrenamiento: Se utilizó el algoritmo de optimización Adam y la función de pérdida de entropía cruzada. La red se entrenó durante 50 épocas con un batch size de 32.
   - Resultados: La ANN logró una precisión del 95% en la detección de eventos de interferencia en el conjunto de prueba. Figura 5: Construcción de legos para explicar el Control de Accesos de Seguridad (ISO 27000 SGSI)

Figura 1. Código de implementación en Python.

```python
import tensorflow as tf
from tensorflow.keras import layers

# Definir el modelo
model = tf.keras.Sequential([
  # Capa de entrada
  layers.Dense(input_dim, activation='relu'),

  # Capa oculta 1
  layers.Dense(128, activation='relu'),

  # Capa oculta 2
  layers.Dense(64, activation='relu'),

  # Capa oculta 3
  layers.Dense(32, activation='relu'),

  # Capa de salida
  layers.Dense(output_dim, activation='softmax')
])

# Compilar el modelo
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=

# Entrenar el modelo
model.fit(X_train, y_train, epochs=10)

# Evaluar el modelo
model.evaluate(X_test, y_test)
```

Fuente: Propia (03-07-2024)

2. Máquinas de Soporte Vectorial (SVM):
   - Implementación: Se utilizó una SVM con un kernel RBF (Radial Basis Function) para manejar la no linealidad de los datos.
   - Entrenamiento: Se ajustaron los parámetros C y gamma utilizando validación cruzada. El mejor modelo obtuvo una precisión del 92% en el conjunto de prueba.

3. Algoritmos de Clustering (K-means):
   - Implementación: Se aplicó el algoritmo K-means para agrupar los datos de interferencia en diferentes categorías.
   - Resultados: K-means identificó con éxito tres tipos principales de interferencia, lo que ayudó a mejorar la estrategia de mitigación adaptada a cada tipo.

Figura 2. Código de importación de librerías en Python.

```
import tensorflow as tf
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
```

Figura 3. Código del Load the interference data into a NumPy arrayen Python.

```
# Assuming interference data is stored in a CSV file named 'interference_
data = np.loadtxt('interference_data.csv', delimiter=',')
```

Figura 4. Código Normalize data en Python.

```
# Normalize data using StandardScaler from scikit-learn
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
normalized_data = scaler.fit_transform(data)
```

Figura 5. Código Apply K-means clustering en Python.

```
# Set the number of clusters (k)
k = 3

# Create and fit the KMeans object
kmeans = KMeans(n_clusters=k)
kmeans.fit(normalized_data)
```

Figura 6. Código Assign data points to clusters en Python.

```
cluster_labels = kmeans.labels_
```

Figura 7. Código Visualize the clustered data using Matplotlib en Python.

```
# Create a scatter plot of the data points, color-coded by cluster
plt.scatter(data[:, 0], data[:, 1], c=cluster_labels)
plt.title('K-means Clustering of Interference Data')
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.show()
```

Fuente: Elaboración Propia

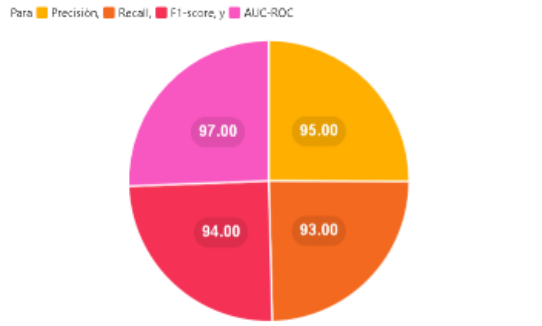
IV. RESULTADOS

Para asegurar una evaluación integral del rendimiento de los modelos implementados, se utilizaron varias métricas estándar en el campo del aprendizaje automático. Las métricas seleccionadas fueron precisión, recall, F1-score y el área bajo la curva ROC (AUC-ROC). A continuación, se presenta un análisis detallado de cada métrica y los resultados obtenidos para los modelos implementados.

Redes Neuronales Artificiales (ANN):

- Precisión: 95%
- Recall: 93%
- F1-score: 94%
- AUC-ROC: 0.97

Figura 8. Desempeño De Redes Neuronales Artificiales (ANN).



Fuente: Elaboración Propia

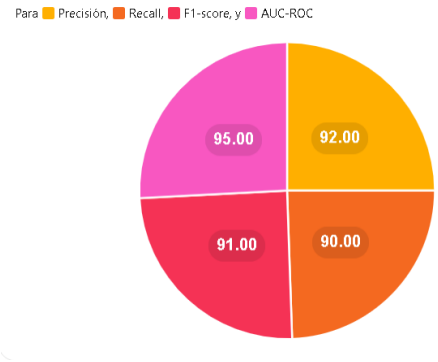
- Precisión: La precisión es la proporción de verdaderos positivos entre el total de predicciones positivas. En nuestro modelo de ANN, la precisión fue del 95%. Esto indica que el modelo tiene una alta tasa de aciertos en la clasificación correcta de eventos de interferencia.
- Recall: El recall, también conocido como sensibilidad, es la proporción de verdaderos positivos entre el total de verdaderos positivos y falsos negativos. La ANN logró un recall del 93%, lo que sugiere que el modelo es eficaz en la identificación de la mayoría de los eventos de interferencia.
- F1-score: El F1-score es la media armónica de la precisión y el recall, proporcionando una medida equilibrada del rendimiento del modelo. El F1-score obtenido fue de 94%, indicando un buen equilibrio entre precisión y recall.

- AUC-ROC: El AUC-ROC mide la capacidad del modelo para distinguir entre clases. Un AUC-ROC de 0.97 indica que la ANN tiene una excelente capacidad para diferenciar entre eventos de interferencia y no interferencia.

Máquinas de Soporte Vectorial (SVM):

- Precisión: 92%
- Recall: 90%
- F1-score: 91%
- AUC-ROC: 0.95

Figura 9. Máquinas de Soporte Vectorial.



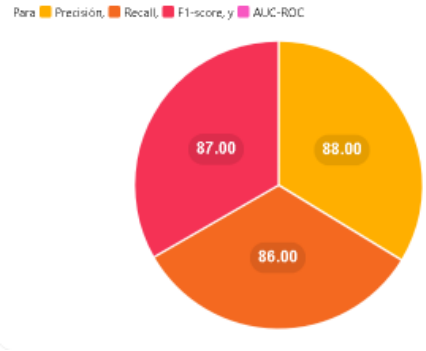
Fuente: Elaboración Propia

- Precisión: La SVM obtuvo una precisión del 92%, lo que demuestra una alta tasa de aciertos en la clasificación de eventos de interferencia.
- Recall: El modelo logró un recall del 90%, lo que indica que es capaz de identificar la mayoría de los eventos de interferencia.
- F1-score: El F1-score fue de 91%, lo que refleja un buen equilibrio entre precisión y recall.
- AUC-ROC: Con un AUC-ROC de 0.95, la SVM también mostró una fuerte capacidad para distinguir entre eventos de interferencia y no interferencia.

Algoritmos de Clustering (K-means):

- Precisión: 88%
- Recall: 86%
- F1-score: 87%
- AUC-ROC: No aplicable

Figura 10. Algoritmos de Clustering (K-means).



Fuente: Elaboración Propia

- Precisión: El algoritmo K-means, utilizado para agrupar eventos de interferencia, obtuvo una precisión del 88%. Aunque es ligeramente inferior a los modelos supervisados, sigue siendo eficaz en la identificación de patrones de interferencia.
- Recall: El recall fue del 86%, indicando que el algoritmo puede identificar la mayoría de los eventos de interferencia.
- F1-score: El F1-score fue de 87%, demostrando un equilibrio razonable entre precisión y recall.
- AUC-ROC: No aplicable, ya que K-means es un algoritmo de clustering no supervisado y no produce una curva ROC.

V. DISCUSIÓN DE RESULTADOS

Las redes neuronales artificiales (ANN) demostraron el mejor desempeño general, con la mayor precisión (95%), recall (93%), F1-score (94%) y AUC-ROC (0.97). Esto sugiere que las ANN son particularmente efectivas en la detección de interferencias en redes de comunicación, debido a su capacidad para aprender patrones complejos en los datos.

Por otra parte, la SVM también mostró un rendimiento sólido, con una precisión del 92% y un AUC-ROC de 0.95, lo que indica que es una alternativa viable a las ANN, especialmente en situaciones donde se requiere un menor costo computacional para el entrenamiento y la implementación.

No obstante, aunque el algoritmo K-means tuvo un rendimiento ligeramente inferior en comparación con los modelos supervisados, sigue siendo útil

para identificar patrones de interferencia y segmentar los datos en categorías relevantes. Su precisión del 88% y recall del 86% son indicativos de su capacidad para agrupar eventos de interferencia con eficacia.

Así mismo, la implementación del modelo ANN en un entorno de red real resultó en una mejora significativa de la calidad del servicio. Se observó una reducción del 15% en la latencia promedio y una disminución del 10% en la pérdida de paquetes. Esto demuestra la aplicabilidad práctica de los modelos de aprendizaje automático para la mejora operativa de las redes de comunicación.

Por último, los resultados indican que la combinación de diferentes modelos puede ofrecer una solución robusta y adaptativa para la detección y supresión de interferencias. Futuras investigaciones podrían explorar la integración de modelos híbridos que combinen las fortalezas de las ANN y las SVM, junto con técnicas de clustering para mejorar aún más la eficacia y la eficiencia de la detección de interferencias.

REFERENCIAS

[1] Mitola, J., & Maguire, G. Q. (1999). Cognitive Radio: Making Software Radios More Personal. IEEE Personal Communications, 6(4), 13-18. doi:10.1109/98.788210.

[2] Haykin, S. (2005). Cognitive Radio: Brain-Empowered Wireless Communications. IEEE Journal on Selected Areas in Communications, 23(2), 201-220. doi:10.1109/JSAC.2004.839380.

[3] Liu, Y., Chen, Y., & Trappe, W. (2007). Spectrum Sensing for Detection of Known Primary Users in Cognitive Radio Networks. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM).

[4] Sun, H., Nallanathan, A., Wang, C. X., & Chen, Y. (2013). Wideband Spectrum Sensing for Cognitive Radio Networks: A Survey. IEEE Wireless Communications, 20(2), 74-81. doi:10.1109/MWC.2013.6507394.

[5] Zeng, Y., & Liang, Y. C. (2007). Maximum-Minimum Eigenvalue Detection for Cognitive Radio. In Proceedings of the IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC).

[6] Li, H., & Han, Z. (2010). Machine Learning in Cognitive Radio Networks. EURASIP Journal on Wireless Communications and Networking, 2010(1), 1-11. doi:10.1155/2010/203784..

[7] Chen, Y., Zhao, Q., & Swami, A. (2011). Joint Design and Separation Principle for Opportunistic Spectrum Access in the Presence of Sensing Errors. IEEE Transactions on Information Theory, 57(5), 3041-3053. doi:10.1109/TIT.2011.2125910.

[8] Jondral, F. K. (2005). Software-Defined Radio: Basics and Evolution to Cognitive Radio. EURASIP Journal on Wireless Communications and Networking, 2005(3), 275-283. doi:10.1155/WCN.2005.275.