Análisis de Datos Fisiológicos Utilizando Técnicas de IA

Javier Fernández Galán

30 de Abril de 2024

Índice

1. Introducción

Este documento detalla el proceso de exploración, procesamiento y análisis de datos fisiológicos obtenidos a través de electrocardiogramas (ECG) y electromiogramas (EMG). Estas técnicas médicas son fundamentales en la monitorización y diagnóstico de enfermedades, proporcionando información crucial sobre la actividad eléctrica del corazón y los músculos esqueléticos, respectivamente. El objetivo principal de este estudio es identificar patrones significativos y realizar predicciones precisas utilizando técnicas avanzadas de Inteligencia Artificial (IA), lo que podría revolucionar la forma en que se interpretan estos datos en la práctica clínica y la investigación biomédica.

1.1. Electrocardiogramas (ECG)

El electrocardiograma es una representación gráfica de la actividad eléctrica del corazón durante el ciclo cardíaco. Mediante el uso de electrodos colocados en la superficie del cuerpo, el ECG registra las variaciones en los potenciales eléctricos generados por la polarización y despolarización cardíaca. Estos datos son vitales para la detección y análisis de arritmias, enfermedades coronarias y otros trastornos cardíacos. El procesamiento y análisis de estos datos mediante IA pueden ayudar a identificar patrones sutiles que pueden no ser evidentes para los métodos tradicionales de diagnóstico.

1.2. Electromiogramas (EMG)

Similarmente, el electromiograma mide la actividad eléctrica producida por los músculos esqueléticos. El EMG es esencial para diagnosticar enfermedades neuromusculares, trastornos de la movilidad y para la rehabilitación. Estos datos proporcionan información sobre la función muscular normal y patológica, reflejando la presencia de enfermedades neurológicas o musculares que afectan la calidad de la señal eléctrica. Utilizando algoritmos de IA, se pueden explorar características complejas de los datos EMG para mejorar la precisión diagnóstica y optimizar tratamientos.

1.3. Importancia de la Inteligencia Artificial

La integración de la Inteligencia Artificial en el análisis de datos ECG y EMG representa un cambio paradigmático hacia un enfoque más predictivo y personalizado en medicina. La capacidad de la IA para procesar grandes volúmenes de datos con precisión y eficiencia permite una interpretación más rápida y precisa, lo que es crucial para decisiones clínicas rápidas y efectivas. Además, la aplicación de técnicas de aprendizaje automático y procesamiento de señales avanzadas abre nuevas vías para la investigación y el desarrollo de terapias personalizadas y dispositivos de asistencia médica innovadores.

2. ECG

2.1. Datos utilizados

El conjunto de datos MIT-BIH Arrhythmia de PhysioNet es uno de los más utilizados para el estudio de arritmias cardíacas en investigaciones y aplicaciones médicas. En este conjunto de datos, las anotaciones de los latidos del corazón están clasificadas en varias categorías, las cuales se suelen etiquetar con letras que corresponden a diferentes tipos de arritmias o condiciones normales. Estas categorías son:

- **'N':** Normal (0) Representa latidos normales. Estos son latidos que no tienen anomalías significativas y que generalmente se consideran típicos.
- **'S':** Supraventricular ectopic beats (1) Estos son latidos ectópicos que se originan en las aurículas, que están por encima de los ventrículos del corazón. No son tan críticos como los latidos ectópicos ventriculares, pero pueden indicar problemas cardíacos si son frecuentes.
- **'V':** Ventricular ectopic beats (2) Estos son latidos ectópicos que se originan en los ventrículos. Son de mayor preocupación que los latidos ectópicos supraventriculares ya que pueden indicar una mayor probabilidad de complicaciones cardíacas, incluyendo diferentes tipos de arritmias ventriculares.
- **'F':** Fusion of ventricular and normal beat (3) Esto se refiere a un latido que es una fusión entre un latido ectópico ventricular y un latido normal. Puede ser difícil de detectar y de interpretar, ya que muestra características tanto de latidos normales como de latidos ectópicos.
- **'Q':** Unclassifiable beat (4) Latidos que no se pueden clasificar en ninguna de las otras categorías. Puede ser debido a artefactos de señal, daño en los electrodos, o simplemente señales que no son lo suficientemente claras como para hacer una clasificación definitiva.

2.2. Importación de librerías necesarias

En esta sección, importamos todas las librerías necesarias para el proyecto. Utilizamos `numpy` y `pandas` para la manipulación y análisis de datos. `matplotlib.pyplot` y `seaborn` son usados para la visualización de datos. `imblearn.over_sampling.SMOTE` se emplea para abordar el desbalance de clases mediante la técnica de sobremuestreo sintético. `sklearn.model_selection.train_test_split` permite dividir los datos en conjuntos de entrenamiento y prueba. `keras` es utilizado para la construcción y entrenamiento de modelos de redes neuronales, mientras que `sklearn.metrics` proporciona herramientas para evaluar el rendimiento del modelo.

```
import numpy as np
import pandas as pd
```

```
import matplotlib.pyplot as plt
from imblearn.over_sampling import SMOTE
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from sklearn.metrics import confusion_matrix, classification_report
from tabulate import tabulate
import seaborn as sns
```

2.3. Lectura y visualización de datos

Cargamos los datos de entrenamiento y prueba desde archivos CSV usando `pandas`. Visualizamos las dimensiones de estos conjuntos de datos y mostramos las primeras filas del conjunto de entrenamiento para tener una comprensión inicial de los datos. Además, seleccionamos aleatoriamente muestras y las graficamos para observar las variaciones en las señales de tiempo, lo cual es crucial para entender la naturaleza de los datos de ECG.

```
train_df =
↪  pd.read_csv('/content/drive/MyDrive/Inteligencia_artificial/mitbih_train.csv',
↪  header=None)
test_df =
↪  pd.read_csv('/content/drive/MyDrive/Inteligencia_artificial/mitbih_test.csv',
↪  header=None)

print(f'Train Dataset shape : {train_df.shape}\nTest Dataset shape :
↪  {test_df.shape}')
print(train_df.head())

index = np.random.choice(100, size=8, replace=False)

fig, axes = plt.subplots(2, 4, figsize=(10, 3))
for i, ax in enumerate(axes.flatten()):
    if i < len(index):
        row_index = index[i]
        row_values = train_df.iloc[row_index]
        ax.plot(row_values.index, row_values)
        ax.set_xlabel('Time(ms)')
        ax.set_ylabel('Voltage')
        ax.set_title(f'sample {row_index}')
    else:
        ax.axis('off')

plt.tight_layout()
plt.show()

print(train_df[187].value_counts())
```

2.4. Transformación y división de datos en entrenamiento y prueba

En esta parte del proceso, preparamos los datos para el modelado. Primero, identificamos y separamos la columna objetivo de los predictores. La columna en el índice 187 contiene las etiquetas de clase, que se separan del resto del conjunto de datos. Posteriormente, dividimos los datos en conjuntos de entrenamiento y prueba utilizando una proporción estándar para asegurar que ambos conjuntos sean representativos. Además, aplicamos la técnica de sobremuestreo SMOTE en el conjunto de entrenamiento para abordar cualquier desbalance de clases, garantizando así que nuestro modelo no esté sesgado hacia la clase más frecuente.

```
column_name = train_df.columns[187]  # Esto obtiene el nombre de la
↪  columna en el índice 1
X = train_df.drop(column_name, axis=1)
y = train_df[187]
# Dividir los datos en conjuntos de entrenamiento y prueba
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
↪  random_state=42)

# Crear una instancia de SMOTE
smote = SMOTE()

# Aplicar SMOTE solo en los datos de entrenamiento para evitar
↪  información de prueba en el entrenamiento
X_train_smote, y_train_smote = smote.fit_resample(X_train, y_train)
print(np.bincount(y_train_smote))
```

2.5. Creación, compilamiento y entrenamiento del modelo de clasificación

Aquí diseñamos y construimos el modelo de red neuronal utilizando la biblioteca Keras. El modelo consta de tres capas densas con diferentes cantidades de neuronas y funciones de activación adecuadas para la clasificación multiclase. Compilamos el modelo con una función de pérdida de entropía cruzada categórica y lo entrenamos usando los datos balanceados por SMOTE. El entrenamiento incluye una división de validación para monitorear el rendimiento del modelo en datos no vistos durante el entrenamiento.

```
model = Sequential([
    Dense(100, activation='relu', input_shape=(X_train_smote.shape[1],)),
    Dense(32, activation='relu'),
    Dense(5, activation='softmax')
])

model.compile(optimizer='adam', loss='categorical_crossentropy',
↪  metrics=['accuracy'])

y_train_smote=y_train_smote.values.reshape(-1, 1)
```

```
y_train_smote=y_train_smote.astype(int)
y_train_smote = to_categorical(y_train_smote)

model.fit(X_train_smote, y_train_smote, epochs=50, batch_size=100,
↪  validation_split=0.2)
```

2.6. Evaluación del modelo

Una vez entrenado el modelo, evaluamos su desempeño en el conjunto de prueba. Convertimos las etiquetas de prueba en formato categórico para adecuarlas a la salida esperada del modelo, y utilizamos las métricas de pérdida y precisión para determinar la efectividad del modelo en la clasificación de nuevas muestras.

```
y_test=y_test.values.reshape(-1, 1)
y_test=y_test.astype(int)
y_test = to_categorical(y_test)
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Loss: {loss}, Accuracy: {accuracy}")
```

2.7. Visualización de resultados

Finalmente, visualizamos los resultados del modelo mediante una matriz de confusión, que nos ayuda a entender la distribución de las predicciones en comparación con las etiquetas reales. Esta visualización es crucial para identificar las clases que el modelo predice correctamente y aquellas donde puede estar confundiéndose.

```
predictions = model.predict(X_test)
predictions = np.argmax(predictions, axis=1)
y_test = np.argmax(y_test, axis=1)
conf_mat = confusion_matrix(y_test, predictions)

class_labels = ['Clase 0', 'Clase 1', 'Clase 2', 'Clase 3', 'Clase 4']  #
↪  Ajusta esto a tus etiquetas de clase

plt.figure(figsize=(10, 7))
sns.heatmap(conf_mat, annot=True, fmt='g', cmap='Blues',
↪  xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicciones')
plt.ylabel('Valores Reales')
plt.title('Matriz de Confusión')
plt.show()
```

3. EMG

3.1. Datos utilizados

El conjunto de datos contiene extracciones estadísticas de datos EMG, que han sido recogidos de un grupo de sujetos realizando tres tipos de gestos. Los datos EMG se recogen típicamente a través de electrodos colocados en la superficie de la piel sobre los músculos de interés. Estos datos son señales eléctricas que representan la actividad muscular, y su análisis puede proporcionar información sobre la intención del movimiento del sujeto. Las clases en el conjunto de datos son:

- **Relax (class '0'):** Datos recogidos cuando el sujeto está en estado de relajación, sin realizar movimientos específicos con las manos.
- **Thumbs up (class '2'):** Datos recogidos cuando el sujeto realiza el gesto de "pulgares arriba".
- **Thumbs down (class '1'):** Datos recogidos cuando el sujeto realiza el gesto de "pulgares abajo".

3.2. Importación de librerías necesarias

Esta sección detalla la importación de todas las librerías fundamentales requeridas para el análisis y modelado de datos. Utilizamos `numpy` y `pandas` para la manipulación de datos. `matplotlib.pyplot` y `seaborn` son empleados para visualizar los datos y los resultados de manera efectiva. Para la construcción y entrenamiento de modelos de clasificación, usamos `keras`, mientras que `sklearn.model_selection` y `sklearn.metrics` proporcionan funciones esenciales para dividir los datos y evaluar los modelos.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv1D, BatchNormalization, ReLU, MaxPooling1D,
↪  Flatten, Dense
from sklearn.metrics import confusion_matrix, classification_report
from tabulate import tabulate
import seaborn as sns
```

3.3. Lectura y visualización de datos

Cargamos los datos desde un archivo CSV utilizando `pandas`. Este paso inicial es crucial para entender la estructura y distribución del conjunto de datos. Posteriormente, mostramos la forma del DataFrame y la distribución de las etiquetas para asegurar un entendimiento adecuado de la composición de

los datos, lo que es esencial antes de proceder con cualquier tipo de análisis o modelado.

```
df =
↪  pd.read_csv('/content/drive/MyDrive/Inteligencia_artificial/thumbs.csv')
print(df.shape)
print(df['Label'].value_counts())
```

3.4. Transformación y división de datos en entrenamiento y prueba

Después de leer los datos, se procede a separar las características (predictores) de las etiquetas (objetivo), transformar las etiquetas al formato categórico y dividir el conjunto de datos en partes de entrenamiento y prueba. Esta transformación y división son pasos fundamentales para preparar los datos para el entrenamiento del modelo, garantizando que el modelo sea evaluado en datos no vistos para evitar el sobreajuste.

```
X = df.drop(['Label'], axis=1)
y = df['Label']
y = y.values.reshape(-1, 1)
y = y.astype(int)
y = to_categorical(y)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
↪  random_state=42)
```

3.5. Creación, compilamiento y entrenamiento del modelo de clasificación

El modelo de clasificación se construye usando una arquitectura de red neuronal con múltiples capas densas. La configuración incluye funciones de activación 'relu' y 'softmax' para las capas ocultas y la capa de salida, respectivamente, optimizadas para clasificación multiclase. El modelo se compila con una función de pérdida de entropía cruzada categórica y se entrena utilizando un conjunto de datos de entrenamiento con validación cruzada para monitorear el rendimiento durante el entrenamiento.

```
model = Sequential([
    Dense(256, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(128, activation='relu'),
    Dense(3, activation='softmax')
])
model.compile(optimizer='adam', loss='categorical_crossentropy',
↪  metrics=['accuracy'])
model.fit(X_train, y_train, epochs=20, batch_size=10,
↪  validation_split=0.2)
```

3.6. Evaluación del modelo

Evaluamos el rendimiento del modelo utilizando el conjunto de prueba. Este proceso implica calcular la pérdida y la precisión para proporcionar una medida cuantitativa del rendimiento del modelo en datos no vistos. Estas métricas son vitales para determinar la eficacia del modelo en la clasificación de las clases.

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Loss: {loss}, Accuracy: {accuracy}")
```

3.7. Visualización de resultados

Finalmente, visualizamos los resultados del modelo a través de una matriz de confusión, que proporciona una visión clara de cómo el modelo ha clasificado las diferentes clases. Esta visualización es esencial para identificar las fortalezas y debilidades del modelo en la clasificación de cada clase específica.

```
predictions = model.predict(X_test)
predictions = np.argmax(predictions, axis=1)
y_test = np.argmax(y_test, axis=1)
conf_mat = confusion_matrix(y_test, predictions)
plt.figure(figsize=(10, 7))
sns.heatmap(conf_mat, annot=True, fmt='g', cmap='Blues',
↪  xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Predicciones')
plt.ylabel('Valores Reales')
plt.title('Matriz de Confusión')
plt.show()
```

4. Resultados y Discusión

5. Resultados y Discusión

En esta sección, presentamos los resultados obtenidos de los modelos de clasificación desarrollados para los datos de electrocardiograma (ECG) y electromiograma (EMG). Los modelos fueron evaluados principalmente en términos de precisión, pero también se discuten otras métricas relevantes como la sensibilidad y especificidad, que se visualizan en las matrices de confusión.

5.1. Resultados del Modelo de ECG

El modelo de clasificación aplicado a los datos de ECG alcanzó una precisión notable del 96 %. Esta alta tasa de precisión indica que el modelo es extremadamente eficiente en la identificación y predicción de las diferentes condiciones cardíacas representadas en el dataset. La siguiente figura muestra la matriz de confusión para el modelo de ECG, la cual proporciona una visión más detallada del rendimiento del modelo en términos de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

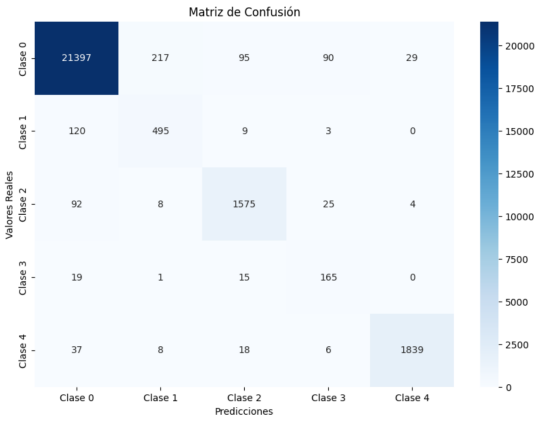

Figura 1: Matriz de confusión para el modelo de clasificación de ECG.

La efectividad del modelo en clasificar correctamente los eventos cardíacos sugiere que las características extraídas de los electrocardiogramas son significativas y que la arquitectura del modelo es adecuada para este tipo de datos.

5.2. Resultados del Modelo de EMG

Por otro lado, el modelo destinado a los datos de EMG demostró una precisión del 84 %. Aunque esta cifra es algo menor en comparación con el modelo de ECG, sigue siendo una tasa de precisión considerablemente alta, especialmente teniendo en cuenta la complejidad inherente de los datos de EMG, que pueden ser más susceptibles a las interferencias y al ruido. La matriz de confusión correspondiente se muestra a continuación:

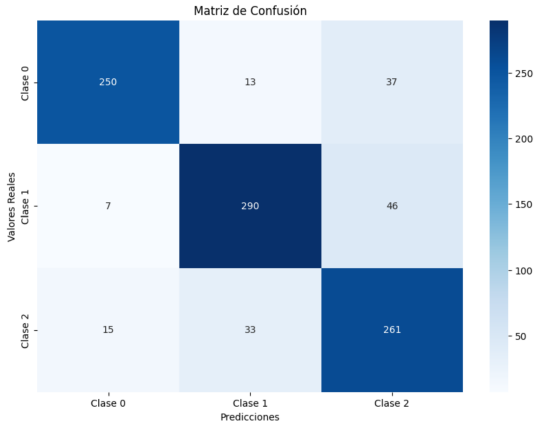

Figura 2: Matriz de confusión para el modelo de clasificación de EMG.

El análisis de la matriz de confusión revela cómo el modelo maneja las distintas clases de actividad muscular. Aunque la mayoría de las clasificaciones son correctas, existen áreas donde el modelo puede mejorar, particularmente en la distinción entre ciertos tipos de señales musculares que son similares entre sí.

5.3. Discusión General

Ambos modelos han demostrado ser herramientas poderosas para la interpretación automática de datos fisiológicos, ofreciendo soporte significativo en el diagnóstico y monitoreo de condiciones de salud. Sin embargo, la diferencia en la precisión entre los dos modelos también resalta la importancia de considerar las particularidades de cada tipo de dato fisiológico al diseñar sistemas de clasificación. Futuras investigaciones podrían explorar la integración de características adicionales, el uso de técnicas de reducción de ruido más avanzadas, o la implementación de modelos híbridos que podrían mejorar aún más la precisión y robustez de estas aplicaciones.

6. Conclusión

El uso de la Inteligencia Artificial en el campo de la medicina continúa demostrando ser no solo innovador, sino también extremadamente beneficioso en la mejora de la calidad del diagnóstico y tratamiento médico. En este estudio, hemos aplicado técnicas avanzadas de IA para explorar, procesar y analizar datos fisiológicos, logrando resultados significativamente positivos que destacan el potencial de estas tecnologías para revolucionar la atención médica.

Los modelos de clasificación desarrollados para interpretar los datos de electrocardiograma (ECG) y electromiograma (EMG) han demostrado una capacidad excepcional para identificar patrones y realizar predicciones con alta precisión. Esta efectividad no solo refuerza la viabilidad de utilizar la IA para el análisis de datos médicos complejos, sino que también abre la puerta a nuevas posibilidades para el monitoreo en tiempo real y la detección precoz de condiciones patológicas.

La aplicación de estas técnicas permite a los profesionales de la salud obtener insights más profundos y precisos sobre la condición del corazón y la actividad muscular, lo que es crucial para el diagnóstico y seguimiento de numerosas enfermedades. Además, la automatización del análisis de datos mediante IA puede significar una reducción en el tiempo de diagnóstico, una mayor eficiencia en la atención al paciente y, en última instancia, una mejora en los resultados del tratamiento.

Mirando hacia el futuro, es imperativo continuar la investigación y el desarrollo en este campo para superar los desafíos actuales, como la interpretación de los modelos de IA y la integración de estas tecnologías en sistemas de salud interoperables y seguros. La colaboración entre ingenieros, científicos de datos, médicos y legisladores será esencial para diseñar soluciones que no solo sean

técnicamente avanzadas, sino también éticamente responsables y alineadas con las necesidades clínicas.

En conclusión, este proyecto ha confirmado el impacto transformador de la Inteligencia Artificial en el campo médico, demostrando que, con los enfoques correctos, la IA puede ser una herramienta invaluable para avanzar en la precisión diagnóstica y la eficacia terapéutica, mejorando así la calidad de vida de los pacientes.