Explainable AI for Gait Analysis: Prototype Generation in Latent Spaces for Clinical Interpretability

Alber Montenegro

1 Posible resumen

La interpretabilidad de los modelos de inteligencia artificial es un desafío clave en aplicaciones clínicas, donde la confianza y la comprensión de los especialistas son fundamentales. Este trabajo propone un método novedoso para mejorar la explicabilidad de modelos de clasificación binaria en señales biomecánicas utilizando prototipos generados en espacios latentes. El enfoque consiste en entrenar clasificadores de alto desempeño, como redes neuronales multicapa y autoencoders, utilizando el conjunto de datos *GaitRec*, y proyectar las señales biomecánicas en un espacio latente representativo. A partir de este espacio, se seleccionan prototipos mediante técnicas no supervisadas, como *k-means* y estimación de densidad por núcleos (*Kernel Density Estimation, KDE*), los cuales son refinados con la validación de especialistas clínicos. Los prototipos seleccionados no solo representan características distintivas de cada clase, sino que también proporcionan ejemplos interpretables para apoyar la toma de decisiones médicas. Los resultados preliminares muestran que el uso de señales prototipo mejora significativamente la capacidad de los modelos para explicar sus predicciones, allanando el camino para una integración más confiable de la inteligencia artificial en el diagnóstico de trastornos del movimiento.

2 Introducción y contexto

En aplicaciones médicas, la interpretabilidad de los modelos de inteligencia artificial (IA) es crucial debido a la necesidad de que los especialistas comprendan y confíen en las decisiones tomadas por los sistemas automáticos. El diagnóstico de trastornos del movimiento basado en señales biomecánicas es un ejemplo clave donde las decisiones del modelo deben ser explicables para apoyar eficazmente a los especialistas clínicos.

Este trabajo se enfoca en abordar este desafío mediante la generación de señales prototipo interpretables desde representaciones latentes. Estos prototipos no solo ofrecen un contexto tangible para las decisiones del modelo, sino que también facilitan la comparación clínica directa.

3 Fases de desarrollo

3.1 Entrenamiento de Clasificadores

3.1.1 Objetivo

Entrenar modelos de clasificación binaria que no solo tengan un alto desempeño predictivo, sino que también generen representaciones latentes que capturen patrones relevantes para la separación de clases en señales biomecánicas.

3.1.2 Procedimiento

El entrenamiento de clasificadores se divide en las siguientes etapas:

1. Selección de datos:

- Se utiliza el conjunto de datos *GaitRec*, que contiene señales biomecánicas de pacientes con y sin patologías relacionadas con el patrón de marcha.
- Para garantizar un análisis justo, las clases se balancean utilizando técnicas como sobremuestreo (*oversampling*) o submuestreo (*undersampling*).
- Los datos se dividen en conjuntos de entrenamiento (70%), validación (15%) y prueba (15%). Se utiliza estratificación para mantener la proporción de clases en cada conjunto.

2. Diseño de modelos:

- **Perceptrón Multicapa (MLP):** Se utiliza una arquitectura de red neuronal densa para capturar relaciones no lineales entre las características.
  - Capas ocultas: Entre 2 y 4 capas con activaciones *ReLU*.
  - Optimización: Descenso de gradiente estocástico (*Adam*) con tasa de aprendizaje inicial de 0.001.
- **Redes Convolucionales (CNN):** Se emplean para explotar la estructura temporal y espacial de las señales.
  - Capas convolucionales: 2-3 capas con filtros de tamaño 3 y activaciones *ReLU*.
  - Capas de agrupamiento: *MaxPooling* para reducir dimensionalidad.
- **Autoencoders:** Diseñados para aprender representaciones latentes comprimidas.
  - Encoder: 3 capas con activación *ReLU*.
  - Decoder: Simétrico al encoder para reconstrucción precisa.
  - Pérdida: Error de reconstrucción (*Mean Squared Error, MSE*).
- **Autoencoders modificados:** Combina reconstrucción y clasificación.
  - Pérdida: Combinación lineal de la pérdida de reconstrucción (*MSE*) y la pérdida de clasificación (*Cross-Entropy*).
  - Resultado esperado: Separación clara de las clases en el espacio latente.
- **Autoencoders variacionales (VAE):** Extensión de los autoencoders estándar que impone una estructura probabilística en el espacio latente.
  - Latentes: Se modelan como distribuciones Gaussianas multivariadas.
  - Pérdida: Combinación de *KL-divergence* y el error de reconstrucción (*MSE*).
  - Ventaja: Genera representaciones latentes suaves y estructuradas, facilitando la generación de prototipos y nuevas muestras.
- **Encoders con pérdida tripleta:** Diseñados para maximizar la separación entre clases en el espacio latente.
  - Pérdida tripleta: Optimiza la distancia entre pares ancla, positivo y negativo.
    - **Ancla:** Punto del conjunto de entrenamiento.
    - **Positivo:** Punto de la misma clase que el ancla.
    - **Negativo:** Punto de una clase diferente.
  - Resultado esperado: Distancias intra-clase mínimas y distancias inter-clase máximas en el espacio latente.
  - Aplicación: Ideal para separar las clases en tareas donde la similitud entre puntos es clave.

3. Entrenamiento de modelos:

- Se utiliza validación cruzada *k-fold* (k=5) para evaluar la estabilidad y generalización de los modelos.
- Las métricas utilizadas para evaluar el desempeño incluyen:
  - *Accuracy:* Proporción de predicciones correctas.
  - *F1-score:* Métrica balanceada entre precisión y sensibilidad.
  - *AUC-ROC:* Área bajo la curva ROC para evaluar la capacidad del modelo para separar clases.
- Los hiperparámetros de los modelos se optimizan utilizando herramientas avanzadas como *Keras Tuner* con el método *Hyperband*, además de enfoques tradicionales como búsqueda en cuadrícula (*Grid Search*) o búsqueda aleatoria (*Random Search*).
  - **Keras Tuner:** Herramienta eficiente para la exploración de hiperparámetros.
    - Se utiliza para ajustar parámetros como el número de capas ocultas, neuronas por capa, tasa de aprendizaje y funciones de activación.
    - Ofrece una API flexible y fácil de integrar con modelos Keras.
  - **Hyperband:** Algoritmo basado en principios de bandit adaptativo, que acelera la búsqueda de hiperparámetros reduciendo el tiempo de entrenamiento innecesario.
    - Divide los recursos computacionales entre múltiples configuraciones y refina iterativamente las mejores opciones.
    - Ventaja: Encuentra configuraciones óptimas más rápido que *Grid Search* o *Random Search*.
  - **Proceso de optimización:**
    - Define un rango de valores para cada hiperparámetro, como:
      - Número de neuronas: [32, 64, 128].
      - Tasa de aprendizaje: [0.001, 0.01, 0.1].
      - Funciones de activación: *ReLU*, *tanh*.
    - Ejecuta *Hyperband* para seleccionar las configuraciones más prometedoras basadas en métricas de validación (e.g., *F1-score*, *accuracy*).
    - Elige la configuración óptima para el modelo final y entrena con todos los datos de entrenamiento.

4. Validación del desempeño:

- Se realiza una evaluación final en el conjunto de prueba para obtener un estimado realista del desempeño del modelo.

3.2 Preparación del Espacio Latente

3.2.1 Objetivo

Extraer representaciones latentes signific ativas de los modelos entrenados.

3.2.2 Procedimiento

1. **Obtención del espacio latente:** Extraer activaciones de las últimas capas antes de la salida del clasificador.
2. **Visualización inicial:** Utilizar métodos como t-SNE o UMAP para proyectar el espacio latente en 2D o 3D.

3.3 Generación de Prototipos

3.3.1 Objetivo

Seleccionar prototipos representativos para cada clase en el espacio latente.

3.3.2 Procedimiento

1. **Técnicas de selección inicial:**
   - **Clustering (k-means):** Agrupar puntos latentes por clase y seleccionar los centroides como prototipos.
   - **Estimación de densidad (KDE):** Identificar puntos en regiones de alta densidad.
2. **Mapeo de prototipos al espacio original:** Seleccionar los datos originales más cercanos a cada prototipo.
3. **Verificación inicial:** Evaluar si los prototipos reflejan características distintivas de cada clase.

3.4 Validación de los Prototipos

3.4.1 Objetivo

Garantizar que los prototipos sean representativos y útiles para la práctica clínica.

3.4.2 Procedimiento

1. **Validación cuantitativa:**
   - Distancia intra-clase: Promedio de las distancias entre puntos y prototipos dentro de cada clase.
   - Separación inter-clase: Distancia promedio entre prototipos de diferentes clases.
2. **Validación cualitativa:** Visualizar los prototipos en el espacio latente y analizar su coherencia.
3. **Revisión clínica:** Consultar a especialistas para validar la relevancia de los prototipos.

3.5 Refinamiento de los Prototipos

3.5.1 Objetivo

Mejorar la calidad de los prototipos basándose en los resultados de la validación.

3.5.2 Procedimiento

1. **Ajustes técnicos:** Refinar el número de clusters (k) o los parámetros de KDE.
2. **Ajustes clínicos:** Incorporar criterios clínicos adicionales para seleccionar prototipos relevantes.
3. **Evaluación iterativa:** Repetir los pasos de validación y refinamiento hasta obtener prototipos robustos.

3.6 Integración y Explicabilidad

3.6.1 Objetivo

Incorporar los prototipos al modelo como herramientas de explicabilidad.

3.6.2 Procedimiento

1. **Prototipos como explicaciones:** Asociar el prototipo más cercano al paciente para cada predicción del modelo.
2. **Evaluación del modelo explicable:** Medir el impacto de los prototipos en la comprensión de las predicciones por parte de los especialistas.

3.7 Discusión y Perspectivas

- Analizar las limitaciones actuales del enfoque, como la generalización a datos de diferentes fuentes.
- Proponer extensiones futuras, como la incorporación de señales multimodales o técnicas avanzadas como KDM o GANs para refinar los prototipos.