Modelos de Aprendizaje Profundo

Índice

1. Resumen

El presente informe tiene como objetivo presentar la documentación de los modelos de aprendizaje profundo utilizados en el trabajo práctico. Se incluye una comparación entre los modelos implementados, FNN (Feedforward Neural Network), TCN (Temporal Convolutional Network), GRU (Gated Recurrent Unit), PPO (Proximal Policy Optimization), entre otros modelos, donde se analizan las distintas métricas obtenidas, como el MAE (Mean Absolute Error), RMSE (Root Mean Square Error), R^2 (R-squared), además de una comparativa entre el uso de CPU y GPU.

2. Introducción

El presente documento reúne la documentación de los modelos de aprendizaje profundo utilizados en el trabajo práctico, así también como una breve descripción de la implementación, utilidad y resultados obtenidos de cada uno. Se incluye una comparativa entre los modelos para poder obtener una conclusión sobre cuál es el modelo más adecuado a utilizar.

3. Objetivos

El objetivo de esta sección es establecer cuáles de los modelos son adecuados para usar en el trabajo práctico, comparando las distintas métricas obtenidas y su eficiencia.

Las métricas a comparar son las siguientes:

1. **MAE** (*Mean Absolute Error*, Error Absoluto Medio) [1]: mide la magnitud promedio de los errores en un conjunto de predicciones, sin considerar su dirección. Se calcula como la media de las diferencias absolutas entre los valores predichos y los valores reales.

Fórmula MAE

$$\text{MAE} = \frac{1}{n} * \sum |y_i - \hat{y}_i|$$

donde:
- n es el número de predicciones.
- y_i es el valor real.
- $\hat{y}_i$ es el valor predicho.

Se busca **minimizar** el MAE, ya que un MAE bajo indica que el modelo tiene un buen rendimiento en la predicción de los valores, lo que significaría, en el marco de este trabajo, una predicción precisa de dosis de insulina. Idealmente, se busca obtener un valor por debajo de 0.5 unidades.

2. **RMSE** (*Root Mean Square Error*, Raíz del Error Cuadrático Medio) [2]: mide el desvío estándar de los errores de predicción. A diferencia del MAE, el RMSE penaliza los errores más grandes de maenra más significativa dado el término cuadrático, por lo que un RMSE más bajo indica que el modelo tiene errores más pequeños en general y es más consistente en sus predicciones. Se calcula como la raíz cuadrada del promedio de los cuadrados de las diferencias entre los valores predichos y los valores reales.

Fórmula RMSE

$$\text{RMSE} = \sqrt{\frac{1}{n} * \sum (y_i - \hat{y}_i)^2}$$

donde:
- n es el número de predicciones.
- y_i es el valor real.
- $\hat{y}_i$ es el valor predicho.

Se busca **minimizar** el RMSE, ya que un RMSE bajo indica que el modelo tiene un buen rendimiento en la predicción de los valores, lo que significaría, en el marco de este trabajo, una predicción precisa de dosis de insulina. Idealmente, se busca obtener un valor por debajo de 1.0 unidad.

3. **R^2** (*R-squared*, Coeficiente de Determinación) [3]: mide la proporción de la varianza total en los datos que es explicada por el modelo. Un R^2 más alto indica que el modelo explica mejor la variabilidad de los datos, lo que significa que tiene un mejor ajuste. Se calcula como:

Fórmula R^2

$$R^2 = 1 - \frac{\text{SS}_{\text{RES}}}{\text{SS}_{\text{TOT}}} = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}$$

donde:
- SS_{RES} es la suma de los cuadrados de los residuos.
- SS_{TOT} es la suma total de los cuadrados.
- y_i es el valor real.
- $\hat{y}_i$ es el valor predicho.
- $\bar{y}$ es la media de los valores reales.

Un R^2 de 1 indica que el modelo explica toda la variabilidad de los datos, mientras que un R^2 de 0 indica que el modelo no explica nada de la variabilidad. Se busca **maximizar** el R^2, idealmente por encima de 0.9, que indicaría que el modelo explica más del 90% de la varianza en los datos y reflejaría un ajuste excelente.

4. **Estabilidad por Sujeto**: se busca lograr que las métricas sean consistentes entre sujetos con especial atención en la reducción del MAE y mejorar el R^2 en casos problemáticos.

4. Feature Engineering

4.1. Extracción de la Ventana CGM

4.1.1. Propósito:

Capturar los niveles históricos de glucosa en sangre (lecturas CGM) que preceden a cada evento de bolo de insulina para proporcionar contexto temporal al modelo.

4.1.2. Implementación:

En la función `get_cgm_window`, se extrae una ventana de 2 horas de datos CGM (definida por `CONFIG["window_hours"]`) antes de cada evento de bolo. La función selecciona las 24 lecturas CGM más recientes (asumiendo intervalos de 5 minutos, 24 lecturas cubren 2 horas) para asegurar un tamaño de entrada consistente. Si hay menos de 24 lecturas disponibles, la muestra se descarta (devuelve `None`).

4.1.3. Resultado:

Resulta en una característica `cgm_window` para cada muestra, que es una lista de 24 valores CGM. Posteriormente se expande en 24 columnas separadas (`cgm_0` a `cgm_23`) en la función `preprocess_data` utilizando `pd.DataFrame(df_processed['cgm_window'].tolist())`.

4.1.4. Impacto:

- Proporciona al modelo tendencias temporales de glucosa en sangre, que son críticas para predecir dosis de insulina, especialmente para sujetos con dosis altas donde las fluctuaciones de glucosa pueden ser más pronunciadas.

4.2. Transformación Logarítmica de Características Numéricas

4.2.1. Propósito:

Reducir la asimetría de las características numéricas y estabilizar la varianza, haciendo que los datos sean más adecuados para el modelado.

4.2.2. Implementación:

En la función `preprocess_data`, las siguientes características se transforman logarítmicamente utilizando `np.log1p` (`log(1 + x)` para manejar valores cero):

- `normal` (dosis de insulina objetivo)
- `carbInput` (ingesta de carbohidratos)
- `insulinOnBoard` (insulina activa, IOB)
- `bgInput` (entrada de glucosa en sangre)
- `cgm_window` (las 24 lecturas CGM)
- `cgm_slope` (una característica derivada, ver más abajo)

Para `cgm_slope`, que puede ser negativa, la transformación se aplica como `np.log1p(abs(x)) * np.sign(x)` para preservar el signo.

4.2.3. Resultado:

Reduce el impacto de los valores extremos y hace que las distribuciones de estas características sean más gaussianas, lo cual es beneficioso para el entrenamiento de redes neuronales.

4.2.4. Impacto:

- Mejora la capacidad del modelo para aprender patrones al reducir la influencia de los valores atípicos (por ejemplo, dosis de insulina o ingestas de carbohidratos muy altas).
- Ayuda al modelo a generalizar mejor a través de diferentes rangos de dosis, particularmente para sujetos con dosis altas.

4.3. Cálculo Personalizado del Factor de Sensibilidad a la Insulina (ISF)

4.3.1. Propósito:

Derivar un factor de sensibilidad a la insulina (ISF) personalizado para cada muestra, que representa cuánto disminuye la glucosa en sangre 1 unidad de insulina.

4.3.2. Implementación:

En la función process_subject, el ISF se calcula como:

- Si la dosis de insulina (normal) es 0, se utiliza un ISF predeterminado de 50.0.
- De lo contrario, el ISF se calcula como (bg_input - 100) / normal, donde bg_input es el nivel de glucosa en sangre en el momento del bolo, y 100 es el objetivo de glucosa en sangre.

El resultado se limita entre 10 y 100 para evitar valores extremos.

4.3.3. Resultado:

Agrega una nueva característica insulinSensitivityFactor que captura la sensibilidad del sujeto a la insulina, que varía entre individuos y contextos.

4.3.4. Impacto:

- Proporciona al modelo una métrica personalizada que relaciona directamente la glucosa en sangre con los requerimientos de insulina, mejorando la precisión de la predicción para sujetos con sensibilidades variables (por ejemplo, sujetos con dosis altas como el Sujeto 49).

4.4. Característica de Pendiente CGM

4.4.1. Propósito:

Capturar la tendencia de los niveles de glucosa en sangre durante los últimos 30 minutos para ayudar al modelo a identificar patrones de glucosa en aumento o disminución, que son críticos para las decisiones de dosificación de insulina.

4.4.2. Implementación:

En la función process_subject, se calcula la pendiente de las últimas 6 lecturas CGM (que cubren 30 minutos, asumiendo intervalos de 5 minutos):

- Si al menos 6 lecturas están disponibles en la cgm_window, se extraen las últimas 6 lecturas (last_6).
- Se aplica un ajuste lineal (np.polyfit) a estas lecturas para calcular la pendiente.
- Si hay menos de 6 lecturas disponibles, la pendiente se establece en 0.0.

La característica cgm_slope luego se transforma logarítmicamente en la función preprocess_data utilizando np.log1p(abs(x)) * np.sign(x) para manejar pendientes negativas y reducir la asimetría.

4.4.3. Resultado:

Agrega una nueva característica cgm_slope que cuantifica la tasa de cambio en los niveles de glucosa en sangre durante los últimos 30 minutos.

4.4.4. Impacto:

- Ayuda al modelo a predecir mejor las dosis de insulina para sujetos con dosis altas al capturar cambios rápidos de glucosa (por ejemplo, un aumento pronunciado puede indicar la necesidad de más insulina).
- Contribuye a la mejora del rendimiento en sujetos con dosis altas como el Sujeto 49 (MAE reducido a 0.16 en la ejecución actual).

4.5. Manejo de Valores Faltantes o Cero con Imputación por la Mediana

4.5.1. Propósito:

Asegurar la robustez al manejar valores faltantes o cero en características clave, que de otro modo podrían generar errores o predicciones sesgadas.

4.5.2. Implementación:

En la función process_subject:

- **Entrada de Carbohidratos (carbInput)**: Si falta carbInput (pd.notna(row['carbInput']) es False) o es cero, se reemplaza con la mediana de los valores no cero de carbInput para el sujeto (carb_median). La mediana se calcula como non_zero_carbs.median() con un valor predeterminado de 10.0 si no existen valores no cero. El resultado se limita a un máximo de CONFIG["cap_carb"] (150).
- **Insulina Activa (insulinOnBoard)**: El IOB se calcula utilizando la función calculate_iob, pero si el resultado es 0, se reemplaza con la mediana de los valores no cero de IOB para el sujeto (iob_median). La mediana se calcula como np.median(non_zero_iob) con un valor predeterminado de 0.5 si no existen valores no cero. El resultado se limita a un máximo de CONFIG["cap_iob"] (5).
- **Entrada de Glucosa en Sangre (bgInput)**: Si falta bgInput (pd.notna(row['bgInput']) es False), se reemplaza con el valor CGM más reciente (cgm_window[-1]). El valor se limita a un mínimo de 50.0 y un máximo de CONFIG["cap_bg"] (300).
- **Dosis de Insulina (normal)**: Si falta normal (pd.notna(row['normal']) es False), se establece en 0.0. El valor se limita a un máximo de CONFIG["cap_normal"] (30).
- **Relación Insulina-Carbohidratos (insulinCarbRatio)**: Si falta insulinCarbRatio (pd.notna(row['insulinCarbRatio']) es False), se establece en un valor predeterminado de 10.0. El valor se limita entre 5 y 20.

4.5.3. Resultado:

Asegura que todas las muestras tengan valores válidos para las características clave, evitando que el modelo encuentre valores NaN o poco realistas.

4.5.4. Impacto:

- Mejora la robustez del conjunto de datos al completar los datos faltantes con estimaciones razonables, lo cual es particularmente importante para sujetos con datos dispersos.
- Ayuda a mantener la consistencia entre las muestras, permitiendo que el modelo aprenda de manera más efectiva.

4.6. Normalización de Características

4.6.1. Propósito:

Estandarizar las características para que tengan una media de cero y una varianza unitaria, lo cual es esencial para el entrenamiento de redes neuronales y asegura que las características con diferentes escalas (por ejemplo, valores CGM versus hora del día) contribuyan por igual al modelo.

4.6.2. Implementación:

En la función split_data:

- **Características CGM**: Las 24 columnas CGM (cgm_0 a cgm_23) se normalizan utilizando un StandardScaler (scaler_cgm). El escalador se ajusta a los datos de entrenamiento y se aplica a los conjuntos de validación y prueba. Los datos CGM se remodelan en una matriz 3D ((n_muestras, 24, 1)) para su posible uso con arquitecturas como CNN o RNN, aunque el modelo PPO actual lo aplana.
- **Otras Características**: Las características ['carbInput', 'bgInput', 'insulinOnBoard', 'insulinCarbRatio', 'insulinSensitivityFactor', 'hour_of_day', 'cgm_slope'] se normalizan utilizando un StandardScaler separado (scaler_other). El escalador se ajusta a los datos de entrenamiento y se aplica a los conjuntos de validación y prueba.
- **Objetivo (normal)**: La dosis de insulina objetivo (normal) se normaliza utilizando un StandardScaler (scaler_y). El escalador se ajusta a los datos de entrenamiento y se aplica a los conjuntos de validación y prueba.

4.6.3. Resultado:

Todas las características y el objetivo se estandarizan para tener una media de cero y una varianza unitaria, asegurando que el modelo pueda aprender de manera efectiva.

4.6.4. Impacto:

- Evita que las características con escalas más grandes (por ejemplo, bgInput) dominen el proceso de aprendizaje.
- Asegura que las predicciones del modelo estén en un espacio normalizado, que luego se desnormalizan utilizando scaler_y.inverse_transform para la evaluación.

4.7. Característica de Hora del Día

4.7.1. Propósito:

Capturar patrones circadianos en los requerimientos de insulina, ya que la sensibilidad a la insulina y los niveles de glucosa a menudo varían a lo largo del día.

4.7.2. Implementación:

En la función process_subject, la característica hour_of_day se extrae de la marca de tiempo del bolo (bolus_time.hour) y se normaliza al rango [0, 1] dividiendo por 23 (ya que las horas varían de 0 a 23).

4.7.3. Resultado:

Agrega una característica hour_of_day que representa la hora del día como un valor continuo entre 0 y 1.

4.7.4. Impacto:

- Permite al modelo aprender patrones dependientes del tiempo en la dosificación de insulina, como dosis más altas a la hora de las comidas o dosis más bajas por la noche.
- Particularmente útil para sujetos con rutinas diarias consistentes, mejorando la capacidad del modelo para predecir dosis con precisión.

4.8. Resumen de las Características Ingenierizadas

El conjunto de datos final incluye las siguientes características ingenierizadas para cada muestra:

- **Características CGM (cgm_0 a cgm_23)**: 24 columnas que representan las últimas 2 horas de lecturas CGM, transformadas logarítmicamente y normalizadas.
- **Entrada de Carbohidratos (carbInput)**: Ingesta de carbohidratos, con valores faltantes/cero imputados utilizando la mediana del sujeto, transformada logarítmicamente y normalizada.
- **Entrada de Glucosa en Sangre (bgInput)**: Nivel de glucosa en sangre en el momento del bolo, con valores faltantes imputados utilizando la lectura CGM más reciente, transformada logarítmicamente y normalizada.
- **Insulina Activa (insulinOnBoard)**: IOB calculado, con valores cero imputados utilizando la mediana del sujeto, transformada logarítmicamente y normalizada.
- **Relación Insulina-Carbohidratos (insulinCarbRatio)**: Relación insulina-carbohidratos, con valores faltantes establecidos en un valor predeterminado de 10.0, limitada y normalizada.
- **Factor de Sensibilidad a la Insulina (insulinSensitivityFactor)**: ISF calculado a medida basado en la glucosa en sangre y la dosis de insulina, limitado y normalizado.
- **Hora del Día (hour_of_day)**: Hora del día normalizada (0 a 1), que representa patrones circadianos.
- **Pendiente CGM (cgm_slope)**: Pendiente de las últimas 6 lecturas CGM (30 minutos), transformada logarítmicamente para manejar valores negativos y normalizada.
- **Objetivo (normal)**: Dosis de insulina, transformada logarítmicamente y normalizada (utilizada como variable objetivo).

5. Modelos

5.1. Modelo Basado en Reglas

El modelo basado en reglas es un enfoque tradicional que utiliza un conjunto de reglas predefinidas para calcular la dosis de insulina. Este modelo se basa en la experiencia clínica y el conocimiento experto, y no requiere entrenamiento previo con datos históricos.

5.1.1. Baseline (Modelo Basado en Reglas)

5.1.1.1. Descripción

El modelo Baseline es un enfoque tradicional que usa un conjunto de reglas predefinidas por expertos para calcular la dosis de insulina. En este modelo, se determina la dosis mediante la siguiente fórmula:

Fórmula Baseline

$$\text{dp} = \frac{\text{ci}}{\text{icr}} + \frac{\text{bgi} - \text{tbg}}{\text{isf}}$$

donde:

- dp es la dosis predicha.
- ci (*carbInput*) es la cantidad de carbohidratos de entrada.
- icr (*insulinCarbRatio*) es la relación de carbohidratos a insulina.
- bgi (*bgInput*) es la cantidad de glucosa de entrada.
- tbg (*targetBg*) es la glucosa objetivo.
- isf (*insulinSensitivityFactor*) es el factor de sensibilidad a la insulina.

5.1.1.2. Componentes Principales

1. **Entradas**
   - **ci** (*carbInput*): cantidad de carbohidratos de entrada.
   - **bgInput** (*bgInput*): cantidad de glucosa de entrada.
   - **icr** (*insulinCarbRatio*): relación de carbohidratos a insulina.
   - **isf** (*insulinSensitivityFactor*): factor de sensibilidad a la insulina.
2. **Regla de Cálculo**
   - La dosis de insulina se calcula sumando la insulina necesaria para cubrir los carbohidratos y la insulina necesaria para corregir el nivel de glucosa actual al objetivo.

5.1.1.3. Ventajas en la Predicción de Glucosa

✅ De fácil entendimiento e implementación.
✅ No reuqiere de datos históricos extensos para su funcionamiento inicial.
✅ Puede servir como punto de referencia para comparar el rendimiento de los modelos más complejos.

5.1.1.4. Consideraciones Importantes

⚠️ La precisión depende en gran medida de la correcta configuración de las reglas y los parámetros individuales del paciente.
⚠️ Puede no adaptarse bien a la variabilidad individual y a patrones complejos en los datos de glucosa.
⚠️ No aprende de los datos ni mejora con el tiempo.

5.2. Modelos de Aprendizaje Profundo

Los modelos de aprendizaje profundo son algoritmos que utilizan redes neuronales para aprender patrones complejos en los datos. Estos modelos son capaces de capturar relaciones no lineales y dependencias temporales en los datos, lo que los hace adecuados para tareas como la predicción de dosis de insulina basándose en lecturas de glucosa.

Los modelos de aprendizaje profundo se entrenan utilizando grandes cantidades de datos históricos, lo que les permite captar y aprender patrones y relaciones complejas en los datos. A continuación, se describen los modelos de aprendizaje profundo utilizados en este trabajo práctico.

5.2.1. Attention-Only

5.2.1.1. Descripción

Attention-Only es un modelo basado en la arquitectura de atención, que se utiliza para procesar secuencias de datos. En este caso, el modelo se centra en las lecturas del monitor continuo de glucosa (CGM) y otras características relevantes para predecir la dosis de insulina.

La atención permite al modelo enfocarse en diferentes partes de la secuencia de entrada, asignando pesos a las características más relevantes para la predicción. Esto es especialmente útil en el contexto de datos temporales, donde algunas lecturas pueden ser más informativas que otras.

5.2.1.2. Componentes Principales

1. **Capa de Entrada**
   - Recibe las características relevantes para la predicción, como las lecturas de CGM recientes, ingesta de carbohidratos, nivel actual de glucosa, etc.
2. **Mecanismo de Atención**
   - Permite al modelo asignar diferentes pesos a las características de entrada, enfocándose en las más relevantes para la predicción.
   - Se calcula utilizando una función de atención que combina las características de entrada y produce un vector de contexto.
3. **Capa de Salida**
   - Produce la predicción de la dosis de insulina.
   - En el caso particular de este modelo, la capa de salida tendrá una única neurona con una función de activación adecuada para la predicción de un valor continuo (dosis de insulina).
4. **Función de Activación**
   - Introduce la no linealidad a la red, permitiendo aprender relaciones complejas entre las características de entrada y la dosis de insulina.
   - Se utiliza una función de activación como ReLU (Rectified Linear Unit) o Sigmoid en las capas ocultas para introducir no linealidades en el modelo.
   - En la capa de salida, se puede utilizar una función de activación lineal o ninguna función de activación para permitir que la red produzca valores continuos.
5. **Función de Pérdida**
   - Se utiliza una función de pérdida como el error cuadrático medio (MSE) para medir la discrepancia entre las predicciones de la red y los valores reales de dosis de insulina.
   - El objetivo del entrenamiento es minimizar esta función de pérdida ajustando los pesos de la red.

5.2.1.3. Ventajas en la Predicción de Glucosa

✅ Puede aprender relaciones no lineales complejas entre las características y la dosis de insulina.
✅ Es relativamente sencillo de implementar y entrenar.
✅ Puede utilizar diversas características como entrada para mejorar la precisión de la predicción.

5.2.1.4. Consideraciones Importantes

⚠️ No tiene memoria inherente de secuencias temporales, por lo que puede no capturar dependencias a largo plazo en los datos de glucosa.
⚠️ El rendimiento depende de la calidad y la cantidad de los datos de entrenamiento.
⚠️ La elección de la arquitectura (número de capas y neuronas) y los hiperparámetros requiere experimntación y ajuste.

5.2.2. CNN (Convolutional Neural Network)

5.2.2.1. Descripción

Convolutional Neural Network (CNN, Red Neuronal Convolucional) es un tipo de red neuronal que se utiliza principalmente para el procesamiento de datos con una estructura de cuadrícula, como imágenes. Sin embargo, también se puede aplicar a datos secuenciales, como las lecturas del monitor continuo de glucosa (CGM).

Las CNNs utilizan capas convolucionales para extraer características locales de los datos de entrada, lo que las hace efectivas para aprender patrones espaciales y temporales. En el contexto de la predicción de dosis de insulina, las CNNs pueden aprender a identificar patrones en las lecturas de CGM y otras características relevantes.

5.2.2.2. Componentes Principales

1. **Capa de Entrada**
   - Recibe las características relevantes para la predicción, como las lecturas de CGM recientes, ingesta de carbohidratos, nivel actual de glucosa, etc.
2. **Capas Convolucionales**
   - Aplican filtros convolucionales a las características de entrada para extraer patrones locales.
   - Cada filtro aprende a detectar características específicas en los datos, como picos o caídas en las lecturas de glucosa.
3. **Capas de Agrupamiento (Pooling)**
   - Reducen la dimensionalidad de los datos y ayudan a extraer características más abstractas.
   - Se utilizan típicamente capas de agrupamiento máximo (max pooling) o promedio (average pooling).
4. **Capa de Salida**
   - Produce la predicción de la dosis de insulina.
   - En el caso particular de este modelo, la capa de salida tendrá una única neurona con una función de activación adecuada para la predicción de un valor continuo (dosis de insulina).
5. **Función de Activación**
   - Introduce la no linealidad a la red, permitiendo aprender relaciones complejas entre las características de entrada y la dosis de insulina.
   - Se utiliza una función de activación como ReLU (Rectified Linear Unit) o Sigmoid en las capas ocultas para introducir no linealidades en el modelo.
   - En la capa de salida, se puede utilizar una función de activación lineal o ninguna función de activación para permitir que la red produzca valores continuos.
6. **Función de Pérdida**
   - Se utiliza una función de pérdida como el error cuadrático medio (MSE) para medir la discrepancia entre las predicciones de la red y los valores reales de dosis de insulina.
   - El objetivo del entrenamiento es minimizar esta función de pérdida ajustando los pesos de la red.

5.2.2.3. Ventajas en la Predicción de Glucosa

✅ Puede aprender relaciones no lineales complejas entre las características y la dosis de insulina.
✅ Es relativamente sencillo de implementar y entrenar.
✅ Puede utilizar diversas características como entrada para mejorar la precisión de la predicción.

5.2.2.4. Consideraciones Importantes

⚠️ No tiene memoria inherente de secuencias temporales, por lo que puede no capturar dependencias a largo plazo en los datos de glucosa.
⚠️ El rendimiento depende de la calidad y la cantidad de los datos de entrenamiento.
⚠️ La elección de la arquitectura (número de capas y neuronas) y los hiperparámetros requiere experimntación y ajuste.

5.2.3. FNN (Feedforward Neural Network)

5.2.3.1. Descripción

Feedforward Neural Network (FNN, Red Neuronal Feedforward) es un tipo de red neuronal artificial donde las conexiones entre los nodos (las neuronas) no forman un ciclo.La información se mueve en una sola dirección, desde la capa de entrada, a través de las capas ocultas (si las hay), hasta la capa de salida. En este contexto, el FNN se utiliza para predecir la dosis de insulina basándose en las lecturas del monitor continuo de glucosa (CGM) y otras características relevantes en un momento dado.

5.2.3.2. Componentes Principales

1. **Capa de Entrada**
   - Recibe las características relevantes para la predicción, como las lecturas de CGM recientes, ingesta de carbohidratos, nivel actual de glucosa, etc.
2. **Capas Ocultas**
   - Realizan transformaciones no lineales en los datos de entrada para aprender patrones complejos.
   - El número de capas ocultas y el número de neuronas en cada capa son hiperparámetros que se ajustan durante el entrenamiento.
3. **Capa de Salida**
   - Produce la predicción de la dosis de insulina.
   - En el caso particular de este modelo, la capa de salida tendrá una única neurona con una función de activación adecuada para la predicción de un valor continuo (dosis de insulina).
4. **Función de Activación**
   - Introducen la no linealidad a la red, permitiendo aprender relaciones complejas entre las características de entrada y la dosis de insulina.
   - Se utiliza una función de activación como ReLU (Rectified Linear Unit) o Sigmoid en las capas ocultas para introducir no linealidades en el modelo.
   - En la capa de salida, se puede utilizar una función de activación lineal o ninguna función de activación para permitir que la red produzca valores continuos.
5. **Función de Pérdida**
   - Se utiliza una función de pérdida como el error cuadrático medio (MSE) para medir la discrepancia entre las predicciones de la red y los valores reales de dosis de insulina.
   - El objetivo del entrenamiento es minimizar esta función de pérdida ajustando los pesos de la red.

5.2.3.3. Ventajas en la Predicción de Glucosa

✅ Puede aprender relaciones no lineales complejas entre las características y la dosis de insulina.
✅ Es relativamente sencillo de implementar y entrenar.
✅ Puede utilizar diversas características como entrada para mejorar la precisión de la predicción.

5.2.3.4. Consideraciones Importantes

⚠️ No tiene memoria inherente de secuencias temporales, por lo que puede no capturar dependencias a largo plazo en los datos de glucosa.
⚠️ El rendimiento depende de la calidad y la cantidad de los datos de entrenamiento.
⚠️ La elección de la arquitectura (número de capas y neuronas) y los hiperparámetros requiere experimntación y ajuste.

5.2.4. GRU (Gated Recurrent Unit)

5.2.4.1. Descripción

Gated Recurrent Unit (GRU, Unidad Recurrente Cerrada) es un tipo de red neuronal recurrente (RNN) que, al igual que el LSTM (Long Short Term Memory), está diseñada para manejar datos secuenciales y dependencias a largo plazo. Sin embargo, la GRU tiene una arquitectura más sumple con solo dos puertas: una de actualización; otra de reinicio.

La puerta de actualización controla cuánto del estado anterior debe conservarse, y cuánta nueva información debe agregarse. La puerta de reinicio determina cuánto del estado anterior debe olvidarse. Esta simplificación hace que las GRUs sean a menudo más rápidas de entrenar y tengan menos parámetros que las LSTMs, al tiempo que mantienen una capacidad similar para capturar dependencias temporales.

5.2.4.2. Componentes Principales

1. **Puerta de Actualización**
   - Controla cuánto del estado oculto anterior se mantiene en el estado oculto actual.
   - Ayuda a la red a decidir qué información del pasado debe conservarse para el futuro.
2. **Puerta de Reinicio**
   - Determina cuánto del estado oculto anterior se utiliza para calcular el nuevo estado candidato.
   - Ayuda a la red a olvidar información irrelevante del pasado.
3. **Estado Candidato**
   - Representa la nueva información que se va a agregar al estado oculto actual.
   - Se calcula utilizando la puerta de reinicio y el estado oculto anterior.
4. **Estado Oculto**
   - Almacena la información aprendida de la secuencia hasta el momento.
   - Se actualiza en cada paso de tiempo utilizando las puertas de actualización y reinicio.

5.2.4.3. Ventajas en la Predicción de Glucosa

✅ Captura dependencias temporales en los datos de glucosa.
✅ Maneja secuencias de longitud variable.
✅ Tiene menos parámetros y es más eficiente computacionalmente que el LSTM.
✅ Puede lograr un rendimiento similar al LSTM en muchas tareas de modelado de secuencias.

5.2.4.4. Consideraciones Importantes

⚠️ Puede que no capture dependencias a muy largo plazo tan bien como el LSTM en algunos casos.
⚠️ Al igual que el LSTM, requiere suficientes datos de entrenamiento y es sensible a la escala de los datos.
⚠️ La longitud de la secuencia y el número de unidades GRU afectan el rendimiento.

5.2.5. LSTM (Long Short Term Memory)

5.2.5.1. Descripción

Long Short Term Memory (LSTM, Memoria a Largo Corto Plazo) es un tipo de red neuronal recurrente (RNN) diseñada para aprender dependencias a largo plazo en datos secuenciales. A diferencia de las RNNs tradicionales, que pueden tener problemas con el desvanecimiento o explosión del gradiente, las LSTMs utilizan una arquitectura especial que incluye puertas de entrada, olvido y salida para controlar el flujo de información.

5.2.5.2. Componentes Principales

1. **Puerta de Entrada**
   - Controla cuánto de la nueva información se agrega al estado oculto.
   - Se calcula utilizando la entrada actual y el estado oculto anterior.
2. **Puerta de Olvido**
   - Determina cuánto del estado oculto anterior se olvida.
   - Ayuda a la red a decidir qué información del pasado es irrelevante y debe ser descartada.
3. **Puerta de Salida**
   - Controla cuánto del estado oculto actual se utiliza para la salida.
   - Se calcula utilizando la entrada actual y el estado oculto anterior.
4. **Estado Candidato**
   - Representa la nueva información que se va a agregar al estado oculto actual.
   - Se calcula utilizando la puerta de entrada y el estado oculto anterior.
5. **Estado Oculto**

- Almacena la información aprendida de la secuencia hasta el momento.
- Se actualiza en cada paso de tiempo utilizando las puertas de entrada, olvido y salida.

6. **Estado de Memoria**
   - Almacena información a largo plazo que puede ser utilizada en pasos de tiempo futuros.
   - Se actualiza utilizando la puerta de entrada y la puerta de olvido.

5.2.5.3. Ventajas en la Predicción de Glucosa

✅ Captura dependencias a largo plazo en los datos de glucosa.
✅ Maneja secuencias de longitud variable.
✅ Es robusto frente al desvanecimiento y explosión del gradiente.
✅ Puede aprender patrones complejos en los datos secuenciales.

5.2.5.4. Consideraciones Importantes

⚠️ Requiere más recursos computacionales y tiempo de entrenamiento que las RNNs simples.
⚠️ Puede ser propenso al sobreajuste si no se regulariza adecuadamente.
⚠️ La longitud de la secuencia y el número de unidades LSTM afectan el rendimiento.

5.2.6. RNN (Recurrent Neural Network)

5.2.6.1. Descripción

Recurrent Neural Network (RNN, Red Neuronal Recurrente) es un tipo de red neuronal diseñada para procesar datos secuenciales. A diferencia de las redes neuronales feedforward, las RNNs tienen conexiones recurrentes que les permiten mantener una memoria de estados anteriores. Esto las hace adecuadas para tareas donde el contexto temporal es importante, como la predicción de dosis de insulina basándose en lecturas de glucosa.

5.2.6.2. Componentes Principales

1. **Capa de Entrada**
   - Recibe las características relevantes para la predicción, como las lecturas de CGM recientes, ingesta de carbohidratos, nivel actual de glucosa, etc.
2. **Capa Oculta**
   - Mantiene el estado oculto que representa la información aprendida de la secuencia hasta el momento.
   - Se actualiza en cada paso de tiempo utilizando la entrada actual y el estado oculto anterior.
3. **Capa de Salida**
   - Produce la predicción de la dosis de insulina.
   - En el caso particular de este modelo, la capa de salida tendrá una única neurona con una función de activación adecuada para la predicción de un valor continuo (dosis de insulina).
4. **Función de Activación**
   - Introduce la no linealidad a la red, permitiendo aprender relaciones complejas entre las características de entrada y la dosis de insulina.
   - Se utiliza una función de activación como ReLU (Rectified Linear Unit) o Sigmoid en las capas ocultas para introducir no linealidades en el modelo.
   - En la capa de salida, se puede utilizar una función de activación lineal o ninguna función de activación para permitir que la red produzca valores continuos.
5. **Función de Pérdida**
   - Se utiliza una función de pérdida como el error cuadrático medio (MSE) para medir la discrepancia entre las predicciones de la red y los valores reales de dosis de insulina.
   - El objetivo del entrenamiento es minimizar esta función de pérdida ajustando los pesos de la red.

5.2.6.3. Ventajas en la Predicción de Glucosa

✅ Captura dependencias temporales en los datos de glucosa.
✅ Maneja secuencias de longitud variable.

✅ Es capaz de aprender patrones complejos en los datos secuenciales.

5.2.6.4. Consideraciones Importantes

⚠️ Puede sufrir de problemas de desvanecimiento o explosión del gradiente, lo que dificulta el aprendizaje de dependencias a largo plazo.
⚠️ Requiere más recursos computacionales y tiempo de entrenamiento que las redes feedforward.
⚠️ La longitud de la secuencia y el número de unidades RNN afectan el rendimiento.

5.2.7. TabNet

5.2.7.1. Descripción

TabNet es un modelo de aprendizaje profundo diseñado para trabajar con datos tabulares. A diferencia de otros modelos que requieren una transformación significativa de los datos, TabNet puede trabajar directamente con datos tabulares sin necesidad de convertirlos a un formato diferente.

TabNet utiliza un enfoque basado en atención para seleccionar características relevantes y aprender representaciones jerárquicas de los datos. Esto lo hace especialmente adecuado para tareas de predicción en datos tabulares, como la predicción de dosis de insulina.

5.2.7.2. Componentes Principales

1. **Capa de Entrada**
   - Recibe las características relevantes para la predicción, como las lecturas de CGM recientes, ingesta de carbohidratos, nivel actual de glucosa, etc.
2. **Mecanismo de Atención**
   - Permite al modelo asignar diferentes pesos a las características de entrada, enfocándose en las más relevantes para la predicción.
   - Se calcula utilizando una función de atención que combina las características de entrada y produce un vector de contexto.
3. **Capas de Decisión**
   - Aprenden representaciones jerárquicas de los datos y permiten al modelo tomar decisiones basadas en las características seleccionadas.
   - Utilizan un enfoque basado en atención para seleccionar características relevantes y aprender representaciones jerárquicas de los datos.
4. **Capa de Salida**
   - Produce la predicción de la dosis de insulina.
   - En el caso particular de este modelo, la capa de salida tendrá una única neurona con una función de activación adecuada para la predicción de un valor continuo (dosis de insulina).
5. **Función de Activación**
   - Introduce la no linealidad a la red, permitiendo aprender relaciones complejas entre las características de entrada y la dosis de insulina.
   - Se utiliza una función de activación como ReLU (Rectified Linear Unit) o Sigmoid en las capas ocultas para introducir no linealidades en el modelo.
   - En la capa de salida, se puede utilizar una función de activación lineal o ninguna función de activación para permitir que la red produzca valores continuos.
6. **Función de Pérdida**
   - Se utiliza una función de pérdida como el error cuadrático medio (MSE) para medir la discrepancia entre las predicciones de la red y los valores reales de dosis de insulina.
   - El objetivo del entrenamiento es minimizar esta función de pérdida ajustando los pesos de la red.

5.2.7.3. Ventajas en la Predicción de Glucosa

✅ Puede trabajar directamente con datos tabulares sin necesidad de transformación.
✅ Utiliza un enfoque basado en atención para seleccionar características relevantes.
✅ Es capaz de aprender representaciones jerárquicas de los datos.

5.2.7.4. Consideraciones Importantes

⚠️ Puede requerir más recursos computacionales y tiempo de entrenamiento que otros modelos.
⚠️ La elección de la arquitectura (número de capas y neuronas) y los hiperparámetros requiere experimntación y ajuste.
⚠️ El rendimiento depende de la calidad y la cantidad de los datos de entrenamiento.
⚠️ La interpretación de los patrones aprendidos puede ser más compleja que en otros modelos.

5.2.8. TCN (Temporal Convolutional Network)

5.2.8.1. Descripción

Temporal Convolutional Network (TCN, Red Convolucional Temporal) es una arquitectura de red neuronal diseñada específicamente para procesar datos secuenciales. A diferencia de las redes neuronales recurrentes (RNN), las TCNs usan convoluciones casuales, lo que significa que la predicción en un momento dado solo depende de los datos pasados y presentes, evitando la "mirada hacia el futuro". Además, las TCNs a menudo incorporan redes residuales para facilitar el entrenamiento de redes profundas y mitigar el problema del gradiente desvaneciente [4].

5.2.8.2. Componentes Principales

1. **Convoluciones Causales**
   - Aseguran que la salida en el tiempo t solo dependa de las entradas hasta el tiempo t.
   - Se implementan típicamente utilizando convoluciones unidimensionales con un desplazamiento adecuado.
2. **Redes Residuales**
   - Permiten que la información fliua directamente a través de las capas, facilitando el aprendizaje de identidades y mejorando el flujo de gradientes.
   - Un bloque residual típico consiste en una o más capas convolucionales seguidas de una conexión de salto que suma la entrada del bloque a su salida.
3. **Dilatación**
   - Las convoluciones dilatadas permiten que la red tenga un campo receptivo muy grande con relativamente pocas capas.
   - El factor de dilatación aumenta exponencialmente con la profundidad de la red, permitiendo capturar dependencias a largo plazo en la secuencia.

5.2.8.3. Ventajas en la Predicción de Glucosa

✅ Procesa secuencias de manera eficiente y en paralelo, pudiendo ser más rápido que las RNNs.
✅ Tiene un campo receptivo flexible que puede adaptarse a la longitud de las dependencias temporales en los datos de glucosa.
✅ Es menos susceptible al porblema de gradiente desvaneciente o explotan en comparación con las RNNs
✅ Puede capturar patrones tanto locales como globales en las series temporales.

5.2.8.4. Consideraciones Importantes

⚠️ Puede requerir más menoria que las RNNs para campos receptivos muy grandes.
⚠️ La interpretación de los patrones aprendidos puede ser más compleja que en las RNNs.
⚠️ El diseño de la arquitectura (número de filtros, capas, tasas de dilatación) puede requerir ajustes.

5.2.9. Transformer

5.2.9.1. Descripción

Transformer es una arquitectura de red neuronal que, a diferencia de las RNNs, no dependen de la secuencialidad de los datos, lo que les permite procesar secuencias completas en paralelo.

Utiliza mecanismos de atención para ponderar la importancia de diferentes partes de la entrada, lo que lo hace especialmente adecuado para tareas donde las relaciones a largo plazo son importantes.

5.2.9.2. Componentes Principales

1. **Capa de Entrada**
   - Recibe las características relevantes para la predicción, como las lecturas de CGM recientes, ingesta de carbohidratos, nivel actual de glucosa, etc.
2. **Mecanismo de Atención**
   - Permite al modelo asignar diferentes pesos a las características de entrada, enfocándose en las más relevantes para la predicción.
   - Se calcula utilizando una función de atención que combina las características de entrada y produce un vector de contexto.
3. **Capas de Decisión**
   - Aprenden representaciones jerárquicas de los datos y permiten al modelo tomar decisiones basadas en las características seleccionadas.
   - Utilizan un enfoque basado en atención para seleccionar características relevantes y aprender representaciones jerárquicas de los datos.
4. **Capa de Salida**
   - Produce la predicción de la dosis de insulina.
   - En el caso particular de este modelo, la capa de salida tendrá una única neurona con una función de activación adecuada para la predicción de un valor continuo (dosis de insulina).
5. **Función de Activación**
   - Introduce la no linealidad a la red, permitiendo aprender relaciones complejas entre las características de entrada y la dosis de insulina.
   - Se utiliza una función de activación como ReLU (Rectified Linear Unit) o Sigmoid en las capas ocultas para introducir no linealidades en el modelo.
   - En la capa de salida, se puede utilizar una función de activación lineal o ninguna función de activación para permitir que la red produzca valores continuos.
6. **Función de Pérdida**
   - Se utiliza una función de pérdida como el error cuadrático medio (MSE) para medir la discrepancia entre las predicciones de la red y los valores reales de dosis de insulina.
   - El objetivo del entrenamiento es minimizar esta función de pérdida ajustando los pesos de la red.

5.2.9.3. Ventajas en la Predicción de Glucosa

✅ Puede capturar relaciones a largo plazo en los datos de glucosa sin depender de la secuencialidad.
✅ Es altamente paralelizable, lo que puede acelerar el entrenamiento.
✅ Utiliza mecanismos de atención para ponderar la importancia de diferentes partes de la entrada.

5.2.9.4. Consideraciones Importantes

⚠️ Puede requerir más recursos computacionales y tiempo de entrenamiento que otros modelos.
⚠️ La elección de la arquitectura (número de capas, cabezas de atención) y los hiperparámetros requiere experimntación y ajuste.
⚠️ El rendimiento depende de la calidad y la cantidad de los datos de entrenamiento.
⚠️ La interpretación de los patrones aprendidos puede ser más compleja que en otros modelos.

5.2.10. WaveNet

5.2.10.1. Descripción

WaveNet es un tipo de red neuronal convolucional que se utiliza principalmente para el procesamiento de datos secuenciales, como audio y series temporales. A diferencia de las redes neuronales recurrentes (RNN), WaveNet utiliza convoluciones causales y dilatadas para capturar dependencias a largo plazo en los datos.

WaveNet es capaz de modelar secuencias de longitud variable y puede aprender patrones complejos en los datos. En el contexto de la predicción de dosis de insulina, WaveNet puede aprender a predecir la dosis óptima basándose en las lecturas del monitor continuo de glucosa (CGM) y otras características relevantes.

5.2.10.2. Componentes Principales

1. **Capa de Entrada**
   - Recibe las características relevantes para la predicción, como las lecturas de CGM recientes, ingesta de carbohidratos, nivel actual de glucosa, etc.
2. **Convoluciones Causales**
   - Aseguran que la salida en el tiempo t solo dependa de las entradas hasta el tiempo t.
   - Se implementan típicamente utilizando convoluciones unidimensionales con un desplazamiento adecuado.
3. **Convoluciones Dilatadas**
   - Permiten que la red tenga un campo receptivo muy grande con relativamente pocas capas.
   - El factor de dilatación aumenta exponencialmente con la profundidad de la red, permitiendo capturar dependencias a largo plazo en la secuencia.
4. **Capas Residuales**
   - Facilitan el flujo de gradientes a través de la red, lo que ayuda a mitigar el problema del gradiente desvaneciente.
   - Un bloque residual típico consiste en una o más capas convolucionales seguidas de una conexión de salto que suma la entrada del bloque a su salida.
5. **Capa de Salida**
   - Produce la predicción de la dosis de insulina.
   - En el caso particular de este modelo, la capa de salida tendrá una única neurona con una función de activación adecuada para la predicción de un valor continuo (dosis de insulina).
6. **Función de Activación**
   - Introduce la no linealidad a la red, permitiendo aprender relaciones complejas entre las características de entrada y la dosis de insulina.
   - Se utiliza una función de activación como ReLU (Rectified Linear Unit) o Sigmoid en las capas ocultas para introducir no linealidades en el modelo.
   - En la capa de salida, se puede utilizar una función de activación lineal o ninguna función de activación para permitir que la red produzca valores continuos.
7. **Función de Pérdida**
   - Se utiliza una función de pérdida como el error cuadrático medio (MSE) para medir la discrepancia entre las predicciones de la red y los valores reales de dosis de insulina.
   - El objetivo del entrenamiento es minimizar esta función de pérdida ajustando los pesos de la red.

5.2.10.3. Ventajas en la Predicción de Glucosa

✅ Puede capturar relaciones a largo plazo en los datos de glucosa sin depender de la secuencialidad.
✅ Es capaz de modelar secuencias de longitud variable y aprender patrones complejos.
✅ Utiliza convoluciones causales y dilatadas para capturar dependencias a largo plazo.

5.2.10.4. Consideraciones Importantes

⚠️ Puede requerir más recursos computacionales y tiempo de entrenamiento que otros modelos.
⚠️ La elección de la arquitectura (número de capas, filtros, tasas de dilatación) y los hiperparámetros requiere experimntación y ajuste.
⚠️ El rendimiento depende de la calidad y la cantidad de los datos de entrenamiento.
⚠️ La interpretación de los patrones aprendidos puede ser más compleja que en otros modelos.

5.3. Modelos de Aprendizaje por Refuerzo

Los modelos de aprendizaje por refuerzo son algoritmos que aprenden a tomar decisiones secuenciales en un entorno, maximizando una recompensa acumulada a lo largo del tiempo. En el contexto de la predicción de dosis de insulina, estos modelos pueden ser utilizados para aprender políticas óptimas de administración de insulina basándose en las lecturas del monitor continuo de glucosa (CGM) y otras características relevantes.

Los modelos de aprendizaje por refuerzo son particularmente útiles en situaciones donde las decisiones deben tomarse en un contexto dinámico y donde las acciones pueden tener consecuencias a largo plazo. En este caso, el objetivo es aprender una política que maximice la recompensa acumulada, que puede estar relacionada con mantener los niveles de glucosa dentro de un rango objetivo.

5.3.1. Métodos Monte Carlo

5.3.1.1. Descripción

Los métodos Monte Carlo son una clase de algoritmos de aprendizaje por refuerzo que utilizan simulaciones aleatorias para estimar el valor de una política. En el contexto de la predicción de dosis de insulina, los métodos Monte Carlo pueden ser utilizados para evaluar diferentes políticas de administración de insulina basándose en simulaciones del comportamiento del paciente.

Estos métodos son particularmente útiles cuando el entorno es complejo y no se puede modelar fácilmente. En lugar de depender de un modelo del entorno, los métodos Monte Carlo generan episodios completos de interacción con el entorno y utilizan las recompensas obtenidas para actualizar las estimaciones de valor.

5.3.1.2. Componentes Principales

1. **Episodios**
   - Secuencias completas de interacciones entre el agente y el entorno.
   - Cada episodio termina en un estado terminal y proporciona una serie de recompensas.
2. **Valor de Estado**
   - Estimación de la recompensa futura esperada para un estado dado.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
3. **Valor de Acción**
   - Estimación de la recompensa futura esperada para una acción dada en un estado dado.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
4. **Política**
   - La estrategia que el agente sigue para seleccionar acciones en función del estado actual.
   - Puede ser una política determinista o estocástica.
5. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
6. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
7. **Exploración y Explotación**
   - El dilema entre explorar nuevas acciones para obtener más información y explotar acciones conocidas que maximizan la recompensa.
   - Los métodos Monte Carlo suelen utilizar estrategias de exploración como epsilon-greedy o softmax para equilibrar la exploración y explotación.

5.3.1.3. Ventajas en la Predicción de Glucosa

✅ No requieren un modelo del entorno, lo que los hace adecuados para entornos complejos.
✅ Pueden aprender políticas óptimas a largo plazo considerando las consecuencias futuras de las decisiones actuales.
✅ Son flexibles y pueden adaptarse a diferentes tipos de problemas de aprendizaje por refuerzo.

5.3.1.4. Consideraciones Importantes

⚠️ Pueden requerir una gran cantidad de episodios para converger a una política óptima.
⚠️ La varianza en las estimaciones de valor puede ser alta, lo que dificulta el aprendizaje.
⚠️ La exploración y explotación deben equilibrarse cuidadosamente para evitar un rendimiento subóptimo.

5.3.2. Policy Iteration

5.3.2.1. Descripción

Policy Iteration es un algoritmo de aprendizaje por refuerzo que combina la evaluación de políticas y la mejora de políticas. En el contexto de la predicción de dosis de insulina, Policy Iteration puede ser utilizado para encontrar una política óptima que maximice la recompensa acumulada a lo largo del tiempo.

El algoritmo comienza con una política inicial y alterna entre evaluar la política actual y mejorarla. Durante la evaluación, se calcula el valor de cada estado bajo la política actual. Durante la mejora, se actualiza la política para seleccionar acciones que maximicen el valor esperado.

5.3.2.2. Componentes Principales

1. **Política**
   - La estrategia que el agente sigue para seleccionar acciones en función del estado actual.
   - Puede ser una política determinista o estocástica.
2. **Valor de Estado**
   - Estimación de la recompensa futura esperada para un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
3. **Valor de Acción**
   - Estimación de la recompensa futura esperada para una acción dada en un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
4. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
5. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
6. **Exploración y Explotación**
   - El dilema entre explorar nuevas acciones para obtener más información y explotar acciones conocidas que maximizan la recompensa.
   - Los métodos de Policy Iteration suelen utilizar estrategias de exploración como epsilon-greedy o softmax para equilibrar la exploración y explotación.

5.3.2.3. Ventajas en la Predicción de Glucosa

✅ Puede encontrar políticas óptimas a largo plazo considerando las consecuencias futuras de las decisiones actuales.
✅ Es más eficiente que los métodos Monte Carlo en términos de convergencia.
✅ Puede adaptarse a diferentes tipos de problemas de aprendizaje por refuerzo.

5.3.2.4. Consideraciones Importantes

⚠️ Requiere un modelo del entorno, lo que puede ser difícil de obtener en entornos complejos.
⚠️ La convergencia puede ser lenta si la política inicial es muy subóptima.
⚠️ La exploración y explotación deben equilibrarse cuidadosamente para evitar un rendimiento subóptimo.

5.3.3. Q-Learning

5.3.3.1. Descripción

Q-Learning es un algoritmo de aprendizaje por refuerzo que busca aprender una política óptima mediante la estimación de la función de valor de acción (Q). En el contexto de la predicción de dosis de insulina, Q-Learning puede ser utilizado para aprender a seleccionar la dosis óptima en función del estado actual del paciente.

El algoritmo utiliza una tabla Q para almacenar las estimaciones de valor de acción para cada par estado--acción. A medida que el agente interactúa con el entorno, actualiza la tabla Q utilizando la recompensa obtenida y la estimación del valor futuro. El objetivo es maximizar la recompensa acumulada a lo largo del tiempo.

5.3.3.2. Componentes Principales

1. **Tabla Q**
   - Almacena las estimaciones de valor de acción para cada par estado-acción.
   - Se actualiza utilizando la recompensa obtenida y la estimación del valor futuro.
2. **Política**
   - La estrategia que el agente sigue para seleccionar acciones en función del estado actual.
   - Puede ser una política determinista o estocástica.
3. **Valor de Estado**
   - Estimación de la recompensa futura esperada para un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
4. **Valor de Acción**
   - Estimación de la recompensa futura esperada para una acción dada en un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
5. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
6. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
7. **Exploración y Explotación**
   - El dilema entre explorar nuevas acciones para obtener más información y explotar acciones conocidas que maximizan la recompensa.
   - Los métodos de Q-Learning suelen utilizar estrategias de exploración como epsilon-greedy o softmax para equilibrar la exploración y explotación.

5.3.3.3. Ventajas en la Predicción de Glucosa

✅ No requiere un modelo del entorno, lo que los hace adecuados para entornos complejos.
✅ Puede aprender políticas óptimas a largo plazo considerando las consecuencias futuras de las decisiones actuales.
✅ Es más eficiente que los métodos Monte Carlo en términos de convergencia.

5.3.3.4. Consideraciones Importantes

⚠️ Requiere una tabla Q, lo que puede ser difícil de manejar en entornos con un gran espacio de estado--acción.
⚠️ La convergencia puede ser lenta si la política inicial es muy subóptima.
⚠️ La exploración y explotación deben equilibrarse cuidadosamente para evitar un rendimiento subóptimo.

5.3.4. Reinforce (Monte Carlo Policy Gradient)

5.3.4.1. Descripción

Reinforce es un algoritmo de aprendizaje por refuerzo basado en gradientes de política que utiliza métodos Monte Carlo para actualizar la política del agente. En el contexto de la predicción de dosis de insulina, Reinforce puede ser utilizado para aprender a seleccionar la dosis óptima en función del estado actual del paciente.

El algoritmo utiliza episodios completos de interacción con el entorno y actualiza la política utilizando la recompensa obtenida al final del episodio. El objetivo es maximizar la recompensa acumulada a lo largo del tiempo.

5.3.4.2. Componentes Principales

1. **Política**
   - La estrategia que el agente sigue para seleccionar acciones en función del estado actual.
   - Puede ser una política determinista o estocástica.
2. **Valor de Estado**
   - Estimación de la recompensa futura esperada para un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
3. **Valor de Acción**
   - Estimación de la recompensa futura esperada para una acción dada en un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
4. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
5. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
6. **Exploración y Explotación**
   - El dilema entre explorar nuevas acciones para obtener más información y explotar acciones conocidas que maximizan la recompensa.
   - Los métodos de Reinforce suelen utilizar estrategias de exploración como epsilon-greedy o softmax para equilibrar la exploración y explotación.
7. **Gradiente de Política**
   - Se utiliza para actualizar la política del agente en función de la recompensa obtenida.
   - El gradiente se calcula utilizando la recompensa acumulada y la probabilidad de seleccionar la acción tomada.

5.3.4.3. Ventajas en la Predicción de Glucosa

✅ No requiere un modelo del entorno, lo que los hace adecuados para entornos complejos.
✅ Puede aprender políticas óptimas a largo plazo considerando las consecuencias futuras de las decisiones actuales.
✅ Es más eficiente que los métodos Monte Carlo en términos de convergencia.

5.3.4.4. Consideraciones Importantes

⚠️ Requiere una gran cantidad de episodios para converger a una política óptima.
⚠️ La varianza en las estimaciones de valor puede ser alta, lo que dificulta el aprendizaje.
⚠️ La exploración y explotación deben equilibrarse cuidadosamente para evitar un rendimiento subóptimo.

5.3.5. SARSA (State-Action-Reward-State-Action)

5.3.5.1. Descripción

SARSA (State-Action-Reward-State-Action) es un algoritmo de aprendizaje por refuerzo que busca aprender una política óptima mediante la estimación de la función de valor de acción (Q). A diferencia de Q-Learning, que utiliza la acción óptima para actualizar la tabla Q, SARSA utiliza la acción seleccionada por la política actual.

En el contexto de la predicción de dosis de insulina, SARSA puede ser utilizado para aprender a seleccionar la dosis óptima en función del estado actual del paciente.

El algoritmo utiliza una tabla Q para almacenar las estimaciones de valor de acción para cada par estado--acción. A medida que el agente interactúa con el entorno, actualiza la tabla Q utilizando la recompensa obtenida y la estimación del valor futuro.

El objetivo es maximizar la recompensa acumulada a lo largo del tiempo.

5.3.5.2. Componentes Principales

1. **Tabla Q**
   - Almacena las estimaciones de valor de acción para cada par estado-acción.
   - Se actualiza utilizando la recompensa obtenida y la estimación del valor futuro.
2. **Política**
   - La estrategia que el agente sigue para seleccionar acciones en función del estado actual.
   - Puede ser una política determinista o estocástica.
3. **Valor de Estado**
   - Estimación de la recompensa futura esperada para un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
4. **Valor de Acción**
   - Estimación de la recompensa futura esperada para una acción dada en un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
5. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
6. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
7. **Exploración y Explotación**
   - El dilema entre explorar nuevas acciones para obtener más información y explotar acciones conocidas que maximizan la recompensa.
   - Los métodos de SARSA suelen utilizar estrategias de exploración como epsilon-greedy o softmax para equilibrar la exploración y explotación.
8. **Gradiente de Política**
   - Se utiliza para actualizar la política del agente en función de la recompensa obtenida.
   - El gradiente se calcula utilizando la recompensa acumulada y la probabilidad de seleccionar la acción tomada.

5.3.5.3. Ventajas en la Predicción de Glucosa

✅ No requiere un modelo del entorno, lo que los hace adecuados para entornos complejos.
✅ Puede aprender políticas óptimas a largo plazo considerando las consecuencias futuras de las decisiones actuales.
✅ Es más eficiente que los métodos Monte Carlo en términos de convergencia.

5.3.5.4. Consideraciones Importantes

⚠️ Requiere una tabla Q, lo que puede ser difícil de manejar en entornos con un gran espacio de estado--acción.
⚠️ La convergencia puede ser lenta si la política inicial es muy subóptima.
⚠️ La exploración y explotación deben equilibrarse cuidadosamente para evitar un rendimiento subóptimo.

5.3.6. Value Iteration

5.3.6.1. Descripción

Value Iteration es un algoritmo de aprendizaje por refuerzo que busca encontrar una política óptima mediante la estimación de la función de valor de estado. A diferencia de Policy Iteration, que alterna entre evaluar y mejorar la política, Value Iteration actualiza directamente los valores de los estados.

En el contexto de la predicción de dosis de insulina, Value Iteration puede ser utilizado para encontrar una política óptima que maximice la recompensa acumulada a lo largo del tiempo.

El algoritmo comienza con una estimación inicial de los valores de los estados y actualiza iterativamente los valores utilizando la función de Bellman. Una vez que los valores convergen, se puede derivar la política óptima a partir de ellos.

5.3.6.2. Componentes Principales

1. **Valor de Estado**
   - Estimación de la recompensa futura esperada para un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
2. **Valor de Acción**
   - Estimación de la recompensa futura esperada para una acción dada en un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
3. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
4. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
5. **Exploración y Explotación**
   - El dilema entre explorar nuevas acciones para obtener más información y explotar acciones conocidas que maximizan la recompensa.
   - Los métodos de Value Iteration suelen utilizar estrategias de exploración como epsilon-greedy o softmax para equilibrar la exploración y explotación.
6. **Función de Bellman**
   - Una ecuación que relaciona el valor de un estado con los valores de los estados vecinos y las recompensas obtenidas.
   - Se utiliza para actualizar los valores de los estados en cada iteración.
7. **Política**
   - La estrategia que el agente sigue para seleccionar acciones en función del estado actual.
   - Puede ser una política determinista o estocástica.
8. **Gradiente de Política**
   - Se utiliza para actualizar la política del agente en función de la recompensa obtenida.
   - El gradiente se calcula utilizando la recompensa acumulada y la probabilidad de seleccionar la acción tomada.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.

5.3.6.3. Ventajas en la Predicción de Glucosa

✅ Puede encontrar políticas óptimas a largo plazo considerando las consecuencias futuras de las decisiones actuales.
✅ Es más eficiente que los métodos Monte Carlo en términos de convergencia.
✅ Puede adaptarse a diferentes tipos de problemas de aprendizaje por refuerzo.

5.3.6.4. Consideraciones Importantes

⚠ Requiere un modelo del entorno, lo que puede ser difícil de obtener en entornos complejos.
⚠ La convergencia puede ser lenta si la política inicial es muy subóptima.
⚠ La exploración y explotación deben equilibrarse cuidadosamente para evitar un rendimiento subóptimo.

5.4. Modelos de Aprendizaje por Refuerzo Profundo

Los modelos de aprendizaje por refuerzo profundo (Deep Reinforcement Learning, DRL) combinan el aprendizaje por refuerzo con redes neuronales profundas para abordar problemas complejos donde los espacios de estado y acción son grandes o continuos. En el contexto de la predicción de dosis de insulina, los modelos DRL pueden ser utilizados para aprender políticas óptimas de administración de insulina basándose en las lecturas del monitor continuo de glucosa (CGM) y otras características relevantes.

Los modelos DRL son particularmente útiles en situaciones donde las decisiones deben tomarse en un contexto dinámico y donde las acciones pueden tener consecuencias a largo plazo. En este caso, el objetivo es aprender una política que maximice la recompensa acumulada, que puede estar relacionada con mantener los niveles de glucosa dentro de un rango objetivo.

Los modelos DRL utilizan redes neuronales profundas para aproximar funciones de valor y políticas, lo que les permite manejar espacios de estado y acción complejos. Estos modelos son capaces de aprender representaciones jerárquicas de los datos y pueden generalizar a situaciones no vistas durante el entrenamiento.

5.4.1. A2C-A3C (Advantage Actor-Critic)

5.4.1.1. Descripción

A2C (Advantage Actor-Critic) y A3C (Asynchronous Advantage Actor-Critic) son algoritmos de aprendizaje por refuerzo profundo que combinan la estimación de la función de valor y la política. En el contexto de la predicción de dosis de insulina, estos algoritmos pueden ser utilizados para aprender a seleccionar la dosis óptima en función del estado actual del paciente.

El algoritmo A2C utiliza un enfoque de actor-crítico, donde el actor es responsable de seleccionar acciones y el crítico estima el valor de los estados. A3C es una extensión de A2C que utiliza múltiples agentes que interactúan con el entorno de manera asíncrona, lo que mejora la estabilidad y la eficiencia del entrenamiento.

5.4.1.2. Componentes Principales

1. **Actor**
   - La parte del modelo que selecciona acciones en función del estado actual.
   - Puede ser una red neuronal que toma como entrada el estado y produce una distribución de probabilidad sobre las acciones.
2. **Crítico**
   - La parte del modelo que estima el valor de los estados.
   - Puede ser una red neuronal que toma como entrada el estado y produce un valor escalar.
3. **Política**
   - La estrategia que el agente sigue para seleccionar acciones en función del estado actual.
   - Puede ser una política determinista o estocástica.
4. **Valor de Estado**
   - Estimación de la recompensa futura esperada para un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
5. **Valor de Acción**
   - Estimación de la recompensa futura esperada para una acción dada en un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
6. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.

7. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
8. **Exploración y Explotación**
   - El dilema entre explorar nuevas acciones para obtener más información y explotar acciones conocidas que maximizan la recompensa.
   - Los métodos de A2C y A3C suelen utilizar estrategias de exploración como epsilon-greedy o softmax para equilibrar la exploración y explotación.
9. **Asincronía**
   - En A3C, múltiples agentes interactúan con el entorno de manera asíncrona, lo que mejora la estabilidad y la eficiencia del entrenamiento.
   - Cada agente actualiza su propio modelo y comparte información con los demás agentes.
10. **Función de Bellman**
    - Una ecuación que relaciona el valor de un estado con los valores de los estados vecinos y las recompensas obtenidas.
    - Se utiliza para actualizar los valores de los estados en cada iteración.
11. **Gradiente de Política**
    - Se utiliza para actualizar la política del agente en función de la recompensa obtenida.
    - El gradiente se calcula utilizando la recompensa acumulada y la probabilidad de seleccionar la acción tomada.
    - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.

5.4.1.3. Ventajas en la Predicción de Glucosa

✅ Puede aprender políticas óptimas a largo plazo considerando las consecuencias futuras de las decisiones actuales.
✅ Es más eficiente que los métodos Monte Carlo en términos de convergencia.
✅ Puede adaptarse a diferentes tipos de problemas de aprendizaje por refuerzo.

5.4.1.4. Consideraciones Importantes

⚠️ Requiere un modelo del entorno, lo que puede ser difícil de obtener en entornos complejos.
⚠️ La convergencia puede ser lenta si la política inicial es muy subóptima.
⚠️ La exploración y explotación deben equilibrarse cuidadosamente para evitar un rendimiento subóptimo.

5.4.2. DDPG (Deep Deterministic Policy Gradient)

5.4.2.1. Descripción

DDPG (Deep Deterministic Policy Gradient) es un algoritmo de aprendizaje por refuerzo profundo que combina la estimación de la función de valor y la política. A diferencia de A2C y A3C, DDPG es un algoritmo off-policy que utiliza una red neuronal para representar la política y otra red neuronal para estimar el valor de los estados.

En el contexto de la predicción de dosis de insulina, DDPG puede ser utilizado para aprender a seleccionar la dosis óptima en función del estado actual del paciente.

El algoritmo utiliza un enfoque de actor-crítico, donde el actor es responsable de seleccionar acciones y el crítico estima el valor de los estados. DDPG utiliza una técnica de experiencia de repetición para almacenar experiencias pasadas y mejorar la estabilidad del entrenamiento.

5.4.2.2. Componentes Principales

1. **Actor**
   - La parte del modelo que selecciona acciones en función del estado actual.
   - Puede ser una red neuronal que toma como entrada el estado y produce una distribución de probabilidad sobre las acciones.

2. **Crítico**
   - La parte del modelo que estima el valor de los estados.
   - Puede ser una red neuronal que toma como entrada el estado y produce un valor escalar.
3. **Política**
   - La estrategia que el agente sigue para seleccionar acciones en función del estado actual.
   - Puede ser una política determinista o estocástica.
4. **Valor de Estado**
   - Estimación de la recompensa futura esperada para un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
5. **Valor de Acción**
   - Estimación de la recompensa futura esperada para una acción dada en un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
6. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
7. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
8. **Exploración y Explotación**
   - El dilema entre explorar nuevas acciones para obtener más información y explotar acciones conocidas que maximizan la recompensa.
   - Los métodos de DDPG suelen utilizar estrategias de exploración como epsilon-greedy o softmax para equilibrar la exploración y explotación.
9. **Experiencia de Repetición**
   - Una técnica que almacena experiencias pasadas en un buffer y las utiliza para entrenar el modelo.
   - Mejora la estabilidad del entrenamiento al permitir que el modelo aprenda de experiencias pasadas.
10. **Redes Neuronales**
    - Se utilizan para representar la política y la función de valor.
    - Pueden ser redes neuronales profundas que aprenden representaciones complejas de los datos.
11. **Función de Bellman**
    - Una ecuación que relaciona el valor de un estado con los valores de los estados vecinos y las recompensas obtenidas.
    - Se utiliza para actualizar los valores de los estados en cada iteración.
12. **Gradiente de Política**
    - Se utiliza para actualizar la política del agente en función de la recompensa obtenida.
    - El gradiente se calcula utilizando la recompensa acumulada y la probabilidad de seleccionar la acción tomada.
    - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.

5.4.2.3. Ventajas en la Predicción de Glucosa

✅ Puede aprender políticas óptimas a largo plazo considerando las consecuencias futuras de las decisiones actuales.
✅ Es más eficiente que los métodos Monte Carlo en términos de convergencia.
✅ Puede adaptarse a diferentes tipos de problemas de aprendizaje por refuerzo.

5.4.2.4. Consideraciones Importantes

⚠️ Requiere un modelo del entorno, lo que puede ser difícil de obtener en entornos complejos.
⚠️ La convergencia puede ser lenta si la política inicial es muy subóptima.
⚠️ La exploración y explotación deben equilibrarse cuidadosamente para evitar un rendimiento subóptimo.

5.4.3. DQN (Deep Q-Network)

5.4.3.1. Descripción

DQN (Deep Q-Network) es un algoritmo de aprendizaje por refuerzo profundo que utiliza redes neuronales para aproximar la función de valor de acción (Q). A diferencia de los métodos tradicionales de Q-Learning, DQN utiliza una red neuronal profunda para representar la función Q, lo que permite manejar espacios de estado-acción más grandes y complejos.

En el contexto de la predicción de dosis de insulina, DQN puede ser utilizado para aprender a seleccionar la dosis óptima en función del estado actual del paciente.

El algoritmo utiliza una técnica de experiencia de repetición para almacenar experiencias pasadas y mejorar la estabilidad del entrenamiento. Además, DQN utiliza un enfoque de red objetivo para estabilizar las actualizaciones de la red Q.

5.4.3.2. Componentes Principales

1. **Red Neuronal**
   - Se utiliza para aproximar la función de valor de acción (Q).
   - Puede ser una red neuronal profunda que aprende representaciones complejas de los datos.
2. **Función de Valor de Acción**
   - Estimación de la recompensa futura esperada para una acción dada en un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
3. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
4. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
5. **Exploración y Explotación**
   - El dilema entre explorar nuevas acciones para obtener más información y explotar acciones conocidas que maximizan la recompensa.
   - Los métodos de DQN suelen utilizar estrategias de exploración como epsilon-greedy o softmax para equilibrar la exploración y explotación.
6. **Experiencia de Repetición**
   - Una técnica que almacena experiencias pasadas en un buffer y las utiliza para entrenar el modelo.
   - Mejora la estabilidad del entrenamiento al permitir que el modelo aprenda de experiencias pasadas.
7. **Red Objetivo**
   - Una copia de la red Q que se utiliza para estabilizar las actualizaciones de la red Q principal.
   - Se actualiza periódicamente para reflejar los cambios en la red Q principal.
8. **Función de Bellman**
   - Una ecuación que relaciona el valor de un estado con los valores de los estados vecinos y las recompensas obtenidas.
   - Se utiliza para actualizar los valores de los estados en cada iteración.
9. **Política**
   - La estrategia que el agente sigue para seleccionar acciones en función del estado actual.
   - Puede ser una política determinista o estocástica.
10. **Gradiente de Política**
    - Se utiliza para actualizar la política del agente en función de la recompensa obtenida.
    - El gradiente se calcula utilizando la recompensa acumulada y la probabilidad de seleccionar la acción tomada.
    - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.

5.4.3.3. Ventajas en la Predicción de Glucosa

✅ Puede aprender políticas óptimas a largo plazo considerando las consecuencias futuras de las decisiones actuales.
✅ Es más eficiente que los métodos Monte Carlo en términos de convergencia.
✅ Puede adaptarse a diferentes tipos de problemas de aprendizaje por refuerzo.

5.4.3.4. Consideraciones Importantes

⚠️ Requiere un modelo del entorno, lo que puede ser difícil de obtener en entornos complejos.
⚠️ La convergencia puede ser lenta si la política inicial es muy subóptima.
⚠️ La exploración y explotación deben equilibrarse cuidadosamente para evitar un rendimiento subóptimo.

5.4.4. PPO (Proximal Policy Optimization)

5.4.4.1. Descripción

Proximal Policy Optimization (PPO, Optimización de Políticas Proximal) es un algoritmo de aprendizaje por refuerzo que se usa para entrenar agentes que toman decisiones secuenciales. En el contexto de la predicción de dosis de insulina, el agente (modelo PPO) aprende una política, que es una función que mappea el estado actual del paciente (por ejemplo, lecturas de CGM, ingesta de carbohidratos, actividad física) a una acción (la dosis de insulina a administrar).

El objetivo del agente es aprender una política que maximice una recompensa acumulada a lo largo del tiempo donde la recompensa está diseñada para reflejar el mantenimiento de los niveles de glucosa dentro de un rango saludable.

PPO es un algoritmo *on-policy*, lo que significa que aprende de las experiencias generadas por la política actual y actualiza la política de manera tal que los nuevos comportamientos no se desvíen demasiado de los antiguos, ayudando a estabilizar el entrenamiento.

5.4.4.2. Componentes Principales

1. **Agente**
   - El modelo que aprende a tomar decisiones sobre la dosis de insulina a administrar en función del estado actual del paciente.
   - Aprende a través de la interacción con el entorno (paciente) y la retroalimentación recibida.
2. **Entorno**
   - La simulación del paciente y su respuesta a las dosis de insulina en función de sus datos (CGM, comidas, etc.)
3. **Política**
   - La función que el agente aprende para mappear el estado del entorno a las acciones (dosis de insulina).
   - En PPO, la política suele estar representada por una red neuronal.
4. **Función de Valor**
   - Estima la recompensa futura esperada para un estado dado.
   - Se usa para reducir la varianza en las estimaciones de la ventaja.
5. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
6. **Optimización Proximal**
   - El mecanismo clave de PPO que limita la magnitud del cambio en la política durante cada actualización para evitar grandes caídas en el rendimiento.
   - Utiliza una función objetivo recortada para asegurar que la nueva política no sea demasiado diferente de la política anterior.

5.4.4.3. Ventajas en la Predicción de Glucosa

✅ Puede aprender políticas óptimas para la administración de insulina a largo plazo, considerando las consecuencias futuras de las decisiones actuales.
✅ Se adapta a la dinámica compleja y a la variabilidad individual de los pacientes.
✅ Puede incorporar múltiples factores y objetivos en la función de recompensa. Por ejemplo, mantener la glucosa en rango, minimizar la hipoglucemia, hiperglucemia, etc.

5.4.4.4. Consideraciones Importantes

⚠️ El entrenamiento de modelos de aprendizaje por refuerzo puede ser complejo y requerir una gran cantidad de datos y simulación del entorno.
⚠️ La definición de la función de recompensa es crucial y puede afectar el comportamineto del agente.
⚠️ La interpretabilidad de la política aprendida puede ser un desafío.
⚠️ La estabilidad del entrenamiento puede ser un problema, y se requieren técnicas como la optimización proximal para mejorarla.

5.4.5. SAC (Soft Actor-Critic)

5.4.5.1. Descripción

Soft Actor-Critic (SAC) es un algoritmo de aprendizaje por refuerzo que combina la optimización de políticas y la estimación de funciones de valor. A diferencia de otros algoritmos, SAC busca maximizar tanto la recompensa esperada como la entropía de la política, lo que fomenta una exploración más amplia y evita el sobreajuste a políticas específicas.

En el contexto de la predicción de dosis de insulina, SAC puede ser utilizado para aprender a seleccionar la dosis óptima en función del estado actual del paciente.

El algoritmo utiliza un enfoque de actor-crítico, donde el actor es responsable de seleccionar acciones y el crítico estima el valor de los estados. SAC utiliza una técnica de experiencia de repetición para almacenar experiencias pasadas y mejorar la estabilidad del entrenamiento.

5.4.5.2. Componentes Principales

1. **Actor**
   - La parte del modelo que selecciona acciones en función del estado actual.
   - Puede ser una red neuronal que toma como entrada el estado y produce una distribución de probabilidad sobre las acciones.
2. **Crítico**
   - La parte del modelo que estima el valor de los estados.
   - Puede ser una red neuronal que toma como entrada el estado y produce un valor escalar.
3. **Política**
   - La estrategia que el agente sigue para seleccionar acciones en función del estado actual.
   - Puede ser una política determinista o estocástica.
4. **Valor de Estado**
   - Estimación de la recompensa futura esperada para un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
5. **Valor de Acción**
   - Estimación de la recompensa futura esperada para una acción dada en un estado dado bajo la política actual.
   - Se actualiza utilizando las recompensas obtenidas en los episodios.
6. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
7. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.

- Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.

8. **Exploración y Explotación**
   - El dilema entre explorar nuevas acciones para obtener más información y explotar acciones conocidas que maximizan la recompensa.
   - Los métodos de SAC suelen utilizar estrategias de exploración como epsilon-greedy o softmax para equilibrar la exploración y explotación.
9. **Experiencia de Repetición**
   - Una técnica que almacena experiencias pasadas en un buffer y las utiliza para entrenar el modelo.
   - Mejora la estabilidad del entrenamiento al permitir que el modelo aprenda de experiencias pasadas.
10. **Redes Neuronales**
    - Se utilizan para representar la política y la función de valor.
    - Pueden ser redes neuronales profundas que aprenden representaciones complejas de los datos.
11. **Función de Bellman**
    - Una ecuación que relaciona el valor de un estado con los valores de los estados vecinos y las recompensas obtenidas.
    - Se utiliza para actualizar los valores de los estados en cada iteración.
12. **Gradiente de Política**
    - Se utiliza para actualizar la política del agente en función de la recompensa obtenida.
    - El gradiente se calcula utilizando la recompensa acumulada y la probabilidad de seleccionar la acción tomada.
    - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.

5.4.5.3. Ventajas en la Predicción de Glucosa

✅ Puede aprender políticas óptimas a largo plazo considerando las consecuencias futuras de las decisiones actuales.
✅ Es más eficiente que los métodos Monte Carlo en términos de convergencia.
✅ Puede adaptarse a diferentes tipos de problemas de aprendizaje por refuerzo.

5.4.5.4. Consideraciones Importantes

⚠️ Requiere un modelo del entorno, lo que puede ser difícil de obtener en entornos complejos.
⚠️ La convergencia puede ser lenta si la política inicial es muy subóptima.
⚠️ La exploración y explotación deben equilibrarse cuidadosamente para evitar un rendimiento subóptimo.

5.4.6. TRPO (Trust Region Policy Optimization)

5.4.6.1. Descripción

TRPO (Trust Region Policy Optimization) es un algoritmo de aprendizaje por refuerzo que se utiliza para entrenar agentes que toman decisiones secuenciales. En el contexto de la predicción de dosis de insulina, el agente (modelo TRPO) aprende una política, que es una función que mappea el estado actual del paciente (por ejemplo, lecturas de CGM, ingesta de carbohidratos, actividad física) a una acción (la dosis de insulina a administrar).

El objetivo del agente es aprender una política que maximice una recompensa acumulada a lo largo del tiempo donde la recompensa está diseñada para reflejar el mantenimiento de los niveles de glucosa dentro de un rango saludable.

TRPO es un algoritmo *on-policy*, lo que significa que aprende de las experiencias generadas por la política actual y actualiza la política de manera tal que los nuevos comportamientos no se desvíen demasiado de los antiguos, ayudando a estabilizar el entrenamiento.

5.4.6.2. Componentes Principales

1. **Agente**

- El modelo que aprende a tomar decisiones sobre la dosis de insulina a administrar en función del estado actual del paciente.
- Aprende a través de la interacción con el entorno (paciente) y la retroalimentación recibida.

2. **Entorno**
   - La simulación del paciente y su respuesta a las dosis de insulina en función de sus datos (CGM, comidas, etc.)
3. **Política**
   - La función que el agente aprende para mappear el estado del entorno a las acciones (dosis de insulina).
   - En TRPO, la política suele estar representada por una red neuronal.
4. **Función de Valor**
   - Estima la recompensa futura esperada para un estado dado.
   - Se usa para reducir la varianza en las estimaciones de la ventaja.
5. **Función de Recompensa**
   - Define el objetivo del agente. En este caso, puede ser una función que otorga recompensas por mantener los niveles de glucosa dentro de un rango objetivo y penaliza las desviaciones.
6. **Optimización de Región de Confianza**
   - El mecanismo clave de TRPO que limita la magnitud del cambio en la política durante cada actualización para evitar grandes caídas en el rendimiento.
   - Utiliza una función objetivo recortada para asegurar que la nueva política no sea demasiado diferente de la política anterior.
7. **Gradiente de Política**
   - Se utiliza para actualizar la política del agente en función de la recompensa obtenida.
   - El gradiente se calcula utilizando la recompensa acumulada y la probabilidad de seleccionar la acción tomada.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
8. **Estimación de Valor**
   - Proceso de actualizar las estimaciones de valor utilizando las recompensas obtenidas en los episodios.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.
9. **Función de Bellman**
   - Una ecuación que relaciona el valor de un estado con los valores de los estados vecinos y las recompensas obtenidas.
   - Se utiliza para actualizar los valores de los estados en cada iteración.
   - Se utiliza para mejorar la política del agente y guiar su comportamiento en el entorno.

5.4.6.3. Ventajas en la Predicción de Glucosa

✅ Puede aprender políticas óptimas para la administración de insulina a largo plazo, considerando las consecuencias futuras de las decisiones actuales.
✅ Se adapta a la dinámica compleja y a la variabilidad individual de los pacientes.
✅ Puede incorporar múltiples factores y objetivos en la función de recompensa. Por ejemplo, mantener la glucosa en rango, minimizar la hipoglucemia, hiperglucemia, etc.

5.4.6.4. Consideraciones Importantes

⚠️ El entrenamiento de modelos de aprendizaje por refuerzo puede ser complejo y requerir una gran cantidad de datos y simulación del entorno.
⚠️ La definición de la función de recompensa es crucial y puede afectar el comportamineto del agente.
⚠️ La interpretabilidad de la política aprendida puede ser un desafío.
⚠️ La estabilidad del entrenamiento puede ser un problema, y se requieren técnicas como la optimización proximal para mejorarla.

6. Herramientas Utilizadas

6.1. Lenguajes de Programación

6.1.1. Python

Python

Se eligió Python como lenguaje de programación para el desarrollo del trabajo práctico debido al gran ecosistema que ofrece para el procesamiento y análisis de datos, así como para el aprendizaje automático y el aprendizaje por refuerzo. Python es ampliamente utilizado en la comunidad científica y tiene una gran cantidad de bibliotecas y herramientas que facilitan la implementación de modelos complejos.

Entre las librerías que se usaron se encuentran NumPy, Pandas, TensorFlow, Keras y PyTorch. Estas librerías proporcionan funcionalidades avanzadas para el manejo de datos, la construcción de modelos y la optimización de hiperparámetros, que se detallarán más adelante.

Se debe usar Python 3.8 o superior.

6.1.2. Julia

Julia

Julia es un lenguaje de programación de alto rendimiento y fácil de usar, especialmente diseñado para la computación técnica y científica. Se eligió Julia para el desarrollo del trabajo práctico debido a su capacidad para manejar cálculos numéricos intensivos y su sintaxis clara y concisa.

Julia es particularmente adecuada para tareas de aprendizaje automático y aprendizaje por refuerzo, ya que ofrece un rendimiento comparable al de lenguajes como C, al tener un compilador JIT (Just in Time). Además, Julia ofrece varios paquetes para realizar análisis de datos y aprendizaje automático, como Flux.jl y MLJ.jl, que permiten construir y entrenar modelos de manera eficiente.

6.2. Librerías

6.2.1. Procesamiento y Análisis de los Datos

6.2.1.1. NumPy

NumPy

NumPy es una biblioteca fundamental para la computación científica en Python. Proporciona un objeto de matriz multidimensional y funciones para trabajar con estos arrays de manera eficiente. NumPy es ampliamente utilizado para realizar operaciones matemáticas y lógicas sobre arrays, así como para manipular datos numéricos.

En el trabajo práctico, se utilizó NumPy para realizar cálculos matemáticos y manipulación de datos, como la normalización y transformación de los datos de entrada.

6.2.1.2. Pandas

Pandas

Pandas es una biblioteca de Python que proporciona estructuras de datos y herramientas para el análisis de datos. Permite manipular y analizar datos tabulares de manera eficiente, facilitando la carga, limpieza y transformación de datos.

En el trabajo práctico, se utilizó Pandas para cargar los datos, realizar operaciones de limpieza y transformación, así como para explorar y analizar los datos antes de entrenar los modelos.

6.2.1.3. Polars

Polars

Polars es una biblioteca de análisis de datos en Python que se centra en la velocidad y la eficiencia. Utiliza un motor de ejecución paralelo y optimizado para realizar operaciones sobre grandes conjuntos de datos de manera rápida.

En el trabajo práctico, se utilizó Polars para realizar operaciones de análisis y manipulación de datos, aprovechando su rendimiento superior en comparación con otras bibliotecas como Pandas.

6.2.2. Visualización de Datos

6.2.2.1. MatPlotLib

MatPlotLib

Matplotlib es una biblioteca de visualización de datos en Python que permite crear gráficos estáticos, animados e interactivos. Proporciona una amplia variedad de tipos de gráficos y opciones de personalización.

En el trabajo práctico, se utilizó Matplotlib para crear gráficos y visualizaciones de los resultados obtenidos por los modelos, facilitando la interpretación y análisis de los datos.

6.2.2.2. Seaborn

Seaborn

Seaborn es una biblioteca de visualización de datos basada en Matplotlib que proporciona una interfaz de alto nivel para crear gráficos atractivos y informativos. Seaborn facilita la creación de gráficos estadísticos y la visualización de relaciones entre variables.

En el trabajo práctico, se utilizó Seaborn para crear gráficos estadísticos y visualizaciones más complejas, mejorando la presentación de los resultados.

6.2.2.3. Plotly

Plotly

Plotly es una biblioteca de visualización interactiva que permite crear gráficos y dashboards interactivos. Proporciona una amplia variedad de tipos de gráficos y opciones de personalización.

En el trabajo práctico, se utilizó Plotly para crear visualizaciones interactivas que permiten explorar los resultados de manera más dinámica.

6.2.3. Aprendizaje Automático

6.2.3.1. TensorFlow

TensorFlow

TensorFlow es una biblioteca de código abierto para el aprendizaje automático y el aprendizaje profundo. Proporciona herramientas y recursos para construir y entrenar modelos de aprendizaje automático, así como para implementar redes neuronales profundas.

En el trabajo práctico, se utilizó TensorFlow para construir y entrenar los modelos de aprendizaje profundo, aprovechando su flexibilidad y rendimiento.

6.2.3.2. Keras

Keras

Keras es una API de alto nivel para construir y entrenar modelos de aprendizaje profundo. Se integra con TensorFlow y proporciona una interfaz sencilla y fácil de usar para crear redes neuronales.

En el trabajo práctico, se utilizó Keras para construir y entrenar los modelos de aprendizaje profundo, facilitando la implementación y ajuste de los hiperparámetros.

6.2.3.3. PyTorch

PyTorch

PyTorch es una biblioteca de aprendizaje profundo de código abierto que proporciona una interfaz flexible y dinámica para construir y entrenar modelos. Es ampliamente utilizada en la investigación y la industria debido a su facilidad de uso y rendimiento.

En el trabajo práctico, se utilizó PyTorch para construir y entrenar los modelos de aprendizaje profundo, aprovechando su flexibilidad y capacidad para manejar datos dinámicos.

6.2.3.4. Scikit-learn

Scikit-learn

Scikit-learn es una biblioteca de aprendizaje automático en Python que proporciona herramientas para la construcción y evaluación de modelos. Incluye algoritmos de clasificación, regresión y agrupamiento, así como herramientas para la selección y evaluación de modelos.

En el trabajo práctico, se utilizó Scikit-learn para realizar tareas de preprocesamiento de datos, selección de características y evaluación de modelos.

6.2.3.5. JAX

JAX

JAX es una biblioteca de Python que permite la diferenciación automática y la ejecución en GPU/TPU. Proporciona herramientas para realizar cálculos numéricos de manera eficiente y flexible.

En el trabajo práctico, se utilizó JAX para realizar cálculos numéricos y optimización de modelos, aprovechando su capacidad para ejecutar operaciones en paralelo.

6.2.4. Aprendizaje por Refuerzo

6.2.4.1. Stable Baselines3

Stable Baselines3

Stable Baselines3 es una biblioteca de aprendizaje por refuerzo en Python que proporciona implementaciones de algoritmos populares de aprendizaje por refuerzo. Facilita la construcción y entrenamiento de agentes de aprendizaje por refuerzo.

En el trabajo práctico, se utilizó Stable Baselines3 para implementar y entrenar el modelo PPO, aprovechando su facilidad de uso y rendimiento.

6.2.5. Simulación de Entornos

6.2.5.1. OpenAI Gym

OpenAI Gym

OpenAI Gym es una biblioteca que proporciona un entorno para el desarrollo y evaluación de algoritmos de aprendizaje por refuerzo. Ofrece una variedad de entornos simulados para entrenar y evaluar agentes.

En el trabajo práctico, se utilizó OpenAI Gym para simular el entorno del paciente y evaluar el rendimiento del modelo PPO.

6.2.5.2. TensorFlow Agents

TensorFlow Agents

TensorFlow Agents es una biblioteca de aprendizaje por refuerzo que proporciona herramientas y recursos para construir y entrenar agentes de aprendizaje por refuerzo en entornos simulados.

En el trabajo práctico, se utilizó TensorFlow Agents para implementar y entrenar el modelo PPO, aprovechando su integración con TensorFlow.

6.2.6. Paralelismo

6.2.6.1. Joblib

Joblib

Joblib es una biblioteca de Python que proporciona herramientas para la paralelización y el almacenamiento en caché de funciones. Facilita la ejecución de tareas en paralelo y la gestión de recursos computacionales.

En el trabajo práctico, se utilizó Joblib para paralelizar procesamiento de los datos y optimizar el uso de recursos computacionales.

7. Dataset

El dataset elegido para realizar este análisis es el DiaTrend [5].

TODO

Completar con el análisis hecho sobre el dataset.

8. Metodología de Entrenamiento

9. Resultados y Análisis

10. Conclusiones

11. Bibliografía

[1] ScienceDirect, «Mean Absolute Error». Accedido: 10 de diciembre de 2023. [En línea]. Disponible en: https://www.sciencedirect.com/topics/engineering/mean-absolute-error

[2] Oracle, «Root Mean Square Error». Accedido: 10 de diciembre de 2023. [En línea]. Disponible en: https://docs.oracle.com/cloud/help/es/pbcs_common/PFUSU/insights_metrics_RMSE.htm

[3] S. Gupta, «R-Squared Formula Explanation». Accedido: 10 de diciembre de 2023. [En línea]. Disponible en: https://medium.com/analytics-vidhya/r-squared-formula-explanation-6dc0096ce3ba

[4] Hernandez Rodriguez, Adrián, «Problema de desvanecimiento del gradiente (Vanishing gradient problem)». Accedido: 10 de diciembre de 2023. [En línea]. Disponible en: https://mlearninglab.com/2018/05/06/problema-de-desvanecimiento-del-gradiente-vanishing-gradient-problem/

[5] T. Prioleau, A. Bartolome, R. Comi, y C. Stanger, «DiaTrend: A dataset from advanced diabetes technology to enable development of novel analytic solutions», *Scientific Data*, vol. 10, p. , 2023, doi: 10.1038/s41597-023-02469-5.