Redes Neuronales Artificiales y LSTM

# Redes Neuronales Artificiales

Las redes neuronales artificiales se estructuran en capas compuestas por neuronas. Cada neurona se conecta con otras mediante pesos, los cuales representan la importancia relativa de cada conexión en el proceso de transmisión de información. Excepto en la capa de entrada, cada neurona posee un sesgo (bias), que es un valor numérico adicional que permite desplazar la función de activación y, por ende, mejorar la capacidad de ajuste del modelo.  
  
El funcionamiento se desarrolla de manera progresiva: el valor de entrada se multiplica por los pesos correspondientes, se le suma el sesgo y este resultado se transmite hacia la siguiente capa. Este proceso se repite sucesivamente hasta llegar a la capa de salida. El valor obtenido en la salida se compara con la referencia esperada (valor real), y en función de la diferencia se realizan ajustes: si el error es considerable, las modificaciones en los parámetros serán mayores; en cambio, si el error es reducido, los ajustes serán mínimos.  
  
Un caso particular de arquitectura es la capa densa (fully connected layer), en la cual todas las neuronas de una capa se conectan con todas las neuronas de la capa siguiente.  
  
El proceso de actualización de pesos y sesgos se lleva a cabo mediante un optimizador (optimizer). La magnitud del paso de ajuste está determinada por la tasa de aprendizaje (learning rate): si esta es demasiado pequeña, los cambios serán muy lentos; si es excesivamente grande, se perderá precisión y estabilidad en la convergencia.  
  
La función de pérdida empleada frecuentemente en problemas de regresión es el error cuadrático medio (MSE, Mean Square Error), que penaliza más fuertemente los errores de gran magnitud respecto a un gran número de errores pequeños.

Donde:

* La cantidad de observaciones es representada con *n*.
* El valor real observado es *yi*.
* El valor predicho por el modelo es *y’i*.

Finalmente, el entrenamiento de una red neuronal se desarrolla a lo largo de múltiples épocas (epochs), entendidas como la cantidad de iteraciones completas sobre el conjunto de datos, durante las cuales el modelo busca minimizar la función de pérdida y aproximarse al resultado esperado.

# Redes LSTM (Long Short-Term Memory)

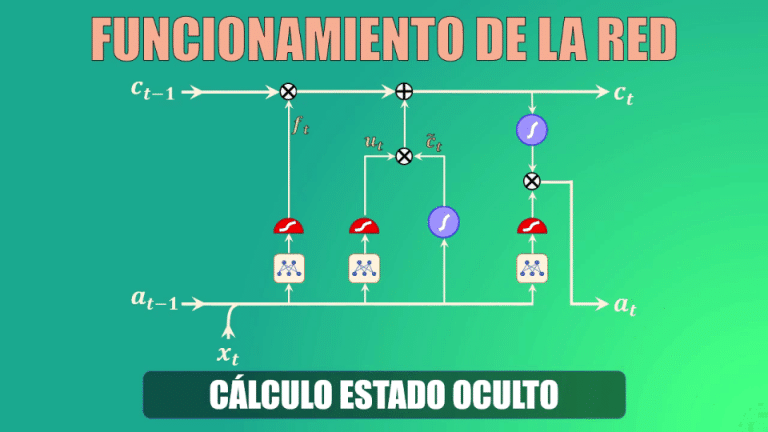
Las redes LSTM constituyen una extensión de las redes neuronales recurrentes (RNN) diseñada para superar limitaciones relacionadas con el problema del desvanecimiento del gradiente y la incapacidad de estas últimas para capturar dependencias de largo plazo. Su unidad fundamental incorpora un mecanismo interno compuesto por compuertas (gates), que regulan dinámicamente el flujo de información en cada iteración temporal.

En las redes LSTM es fundamental distinguir entre el estado de la celda (cell state) y el estado oculto (hidden state), ya que ambos cumplen roles diferentes en el flujo de información. El cell state puede considerarse como la memoria de largo plazo de la red. Se transmite de una celda a la siguiente a lo largo de la secuencia temporal, manteniendo información acumulada y solo siendo modificado por las compuertas de olvido e ingreso. De esta manera, funciona como un canal de memoria que atraviesa la red y que va siendo ajustado gradualmente.

El hidden state, en cambio, corresponde a la memoria de corto plazo y constituye la salida inmediata de cada celda. Este estado se calcula a partir del cell state, pero filtrado por la compuerta de salida y normalizado mediante una función tangente hiperbólica. Su función principal es transmitir información relevante hacia la siguiente celda, en conjunto con los nuevos datos de entrada, así como servir de salida de la red en cada instante de tiempo. En consecuencia, cuando se habla de la “celda” que entra a las compuertas, en realidad se hace referencia al cell state, ya que es este el que se ve afectado directamente por las compuertas de olvido e ingreso. El hidden state, en cambio, no es manipulado por dichas compuertas, sino que resulta de la combinación del cell state actualizado y del filtrado propio de la compuerta de salida.

Es importante aclarar que, en este contexto, una celda LSTM no equivale a una única neurona, sino que constituye un módulo completo que contiene múltiples operaciones internas. Mientras que en una red densa tradicional una neurona realiza únicamente una suma ponderada seguida de una función de activación, la celda LSTM integra varias de estas operaciones junto con compuertas específicas que regulan el flujo de información. En otras palabras, cada celda LSTM está conformada por pequeñas redes neuronales internas —las compuertas— que actúan en conjunto para decidir qué información se conserva, cuál se descarta y cuál se transmite hacia la siguiente iteración.

De este modo, puede entenderse a la celda LSTM como la unidad recurrente fundamental de la red, una estructura más compleja que una simple neurona y diseñada específicamente para manejar información secuencial con dependencias de corto y largo plazo.  
Una celda LSTM está conformada por los siguientes elementos principales:  
  
1. Forget Gate (compuerta de olvido): determina qué parte de la información proveniente del estado de la celda anterior debe ser descartada. Este proceso se logra mediante una red neuronal seguida de una función de activación sigmoide, que genera valores en el rango [0,1]. El resultado se multiplica con el estado previo: un valor cercano a 0 implica olvido total, mientras que un valor próximo a 1 implica conservación completa de la información.  
  
2. Input Gate (compuerta de entrada o actualización): decide qué nueva información debe incorporarse al estado de la celda actual. Este mecanismo combina el resultado de una función sigmoide —que regula la relevancia de la información entrante— con un vector de valores candidatos, generado a partir de una función tangente hiperbólica. La información seleccionada se suma al estado resultante de la forget gate, conformando el nuevo cell state.  
  
3. Output Gate (compuerta de salida): regula la porción de la memoria de la celda que será transmitida como salida en la iteración actual. Para ello, el cell state se transforma mediante una función tangente hiperbólica, restringiendo los valores al intervalo [-1,1]. Posteriormente, este resultado se multiplica por la señal de la compuerta de salida (activación sigmoide), permitiendo que únicamente la información relevante sea propagada al siguiente estado oculto.  
  
En conjunto, este diseño permite que las LSTM almacenen, actualicen y transmitan información a lo largo de secuencias extensas, garantizando la capacidad de modelar dependencias temporales de largo alcance. Esto las convierte en una arquitectura especialmente adecuada para problemas de predicción de series temporales, procesamiento de lenguaje natural y reconocimiento de voz, entre otras aplicaciones.



# Redes basadas en Gradient Boosting (GBR)

El Gradient Boosting Regressor (GBR) es un algoritmo de ensamble que, si bien no constituye una red neuronal en el sentido estricto, puede analizarse desde una perspectiva semejante debido a su construcción secuencial y su capacidad de aproximar funciones complejas. Su principio fundamental consiste en entrenar de manera iterativa modelos débiles (árboles de regresión en nuestro caso), de modo que cada nuevo modelo busque corregir los errores cometidos por los anteriores.

El proceso comienza con una predicción inicial, que suele ser un valor promedio del conjunto de entrenamiento. A partir de esta primera aproximación, se calculan los residuos, es decir, las diferencias entre los valores reales y las predicciones obtenidas. Un nuevo árbol se entrena con el objetivo de modelar dichos residuos en lugar de los valores originales. La salida de este árbol se combina con la predicción acumulada, ponderada por un factor denominado tasa de aprendizaje (*learning rate*), lo que permite controlar la magnitud de los ajustes. Este mecanismo se repite de forma iterativa, generando un modelo final que resulta de la suma ponderada de todos los predictores intermedios.

Desde una visión análoga al funcionamiento de una red neuronal, cada árbol puede concebirse como una capa de corrección que refina progresivamente la salida. En este esquema, la tasa de aprendizaje desempeña un papel equivalente al del parámetro homónimo en redes neuronales: valores demasiado pequeños conducen a una convergencia lenta, mientras que valores excesivos pueden ocasionar inestabilidad o sobreajuste. Asimismo, la función de pérdida guía el proceso de optimización de manera similar a como lo hace en el retropropagado de errores de una red neuronal tradicional. Por su parte, el número de iteraciones del algoritmo cumple una función comparable a las épocas en el entrenamiento de modelos neuronales, dado que define la cantidad de veces que el sistema ajusta sus parámetros para mejorar la aproximación.

El enfoque de Gradient Boosting presenta ventajas notables, entre ellas la capacidad de modelar relaciones no lineales complejas con elevada precisión y robustez frente al ruido, superando en desempeño a modelos basados en un único árbol de decisión. Sin embargo, también presenta limitaciones, como el costo computacional asociado al entrenamiento de un gran número de árboles y la propensión al sobreajuste cuando no se regulan adecuadamente la profundidad de los árboles y la cantidad de iteraciones.

En conjunto, el GBR puede considerarse un modelo de naturaleza híbrida que, aunque no se enmarca en la categoría de redes neuronales profundas, comparte con ellas la idea de un aprendizaje progresivo y estructurado en etapas, lo que lo convierte en una herramienta de gran relevancia en tareas de regresión y clasificación.

## Modelos débiles en Gradient Boosting

El algoritmo de Gradient Boosting se fundamenta en la construcción de un ensamble de predictores simples, conocidos como modelos débiles. En el caso de las implementaciones utilizadas en este trabajo, dichos modelos corresponden a árboles de decisión de regresión poco profundos. La elección de esta estructura no es arbitraria: los árboles de baja profundidad son lo suficientemente flexibles como para capturar relaciones no lineales en los datos, pero lo bastante simples como para no sobreajustar individualmente. Su verdadera potencia surge al ser combinados de manera secuencial.

Cada árbol se entrena con el objetivo de corregir los errores residuales cometidos por el ensamble acumulado hasta el momento. De esta forma, el primer árbol intenta explicar la diferencia entre la predicción inicial (generalmente el promedio del valor objetivo) y los valores reales. El segundo árbol se ajusta para modelar los residuos restantes después de incorporar el primer árbol, y así sucesivamente. En cada iteración, la salida del nuevo árbol se agrega a la predicción acumulada, ajustada mediante un factor de aprendizaje (learning rate) que regula la magnitud del aporte de cada modelo débil al resultado final.

El proceso iterativo de corrección convierte al conjunto de árboles en un modelo fuerte capaz de capturar dependencias complejas y patrones de difícil detección. En términos conceptuales, se puede entender a cada árbol débil como una capa adicional que refina progresivamente la aproximación, similar al rol que cumplen las capas ocultas en una red neuronal. Sin embargo, la diferencia principal radica en la forma en que se realiza la optimización: mientras que en una red neuronal la retropropagación ajusta pesos de manera global, en Gradient Boosting cada nuevo árbol se entrena localmente para minimizar el gradiente de la función de pérdida.

Esta metodología asegura que el modelo final no dependa de un único árbol complejo, sino de la combinación de múltiples árboles simples. Así, se logra un equilibrio entre sesgo y varianza, lo que convierte al Gradient Boosting en un enfoque altamente efectivo para problemas de predicción tanto en regresión como en clasificación.

# Árboles de decisión

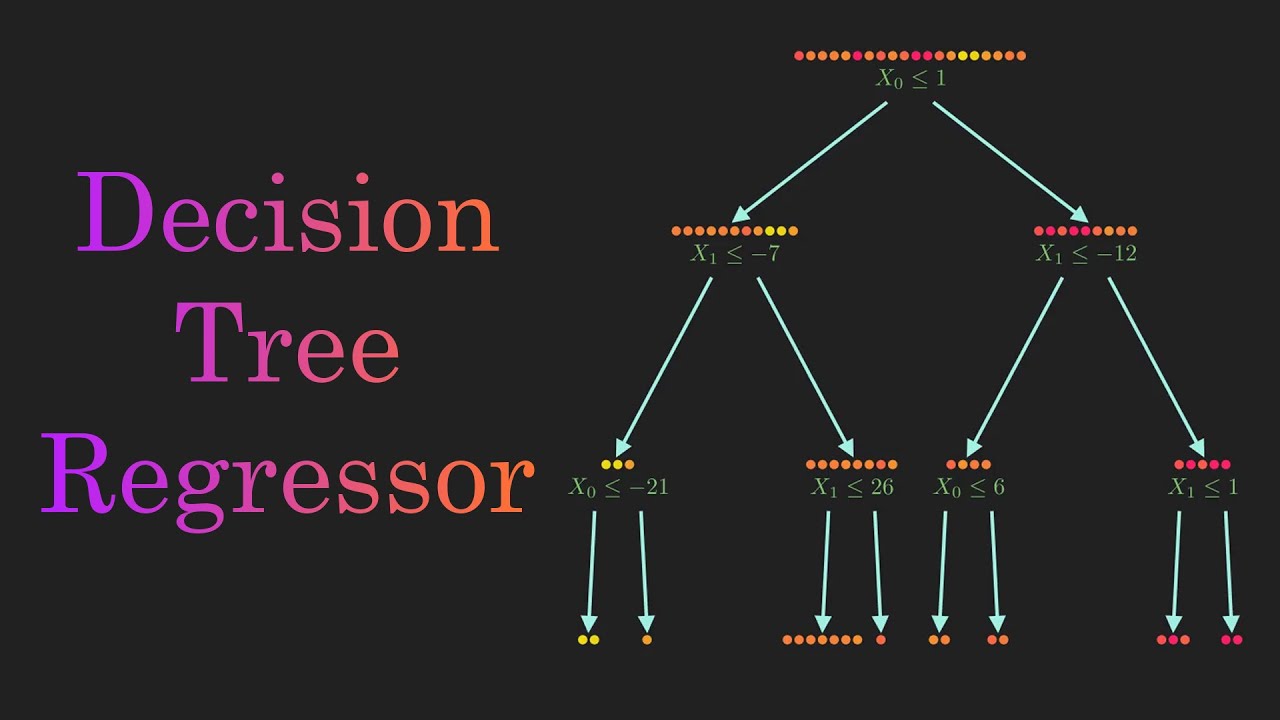
Los árboles de decisión constituyen uno de los modelos más utilizados y versátiles en el campo del aprendizaje automático. Su funcionamiento se basa en representar el proceso de toma de decisiones mediante una estructura jerárquica en forma de árbol, en la cual los nodos internos corresponden a condiciones sobre los atributos de entrada, las ramas representan los resultados de esas condiciones, y las hojas indican la predicción final.

En el caso de la clasificación, las hojas contienen etiquetas de clase y cada decisión en el recorrido del árbol conduce a asignar un ejemplo a una categoría determinada. En el caso de la regresión, las hojas almacenan valores numéricos y el árbol predice un valor continuo. En ambos contextos, la idea central es dividir recursivamente el espacio de datos en regiones cada vez más homogéneas en relación con la variable objetivo.

El entrenamiento de un árbol de decisión se realiza mediante un proceso denominado partición recursiva. En cada nodo, el algoritmo selecciona la característica y el umbral de división que mejor separen los datos según un criterio de impureza o error. Entre los criterios más utilizados se encuentran el índice Gini y la entropía para problemas de clasificación, mientras que para regresión suele emplearse el error cuadrático medio. Una vez definida la mejor división, los datos se reparten en ramas y el proceso se repite hasta alcanzar un criterio de detención, como puede ser la profundidad máxima del árbol, el número mínimo de muestras por nodo o la ausencia de mejora en la partición.

Los árboles de decisión presentan varias ventajas: son fáciles de interpretar, ya que la estructura resultante puede representarse gráficamente y entenderse como un conjunto de reglas de tipo “si... entonces”. Además, pueden manejar tanto variables categóricas como numéricas y requieren poca preparación de los datos. Sin embargo, también poseen limitaciones, entre las que se destacan la tendencia al sobreajuste cuando el árbol es demasiado profundo, la inestabilidad frente a pequeñas variaciones en los datos de entrenamiento, y la dificultad para modelar relaciones demasiado complejas sin volverse excesivamente grandes.

En síntesis, los árboles de decisión ofrecen un equilibrio entre simplicidad interpretativa y capacidad predictiva, lo que los convierte en un modelo de base ampliamente utilizado, tanto de forma individual como como componente esencial en métodos de ensamble más avanzados, tales como Random Forest o Gradient Boosting.



# Bibliografía

* <https://en.wikipedia.org/wiki/Gradient_boosting>
* <https://scikit-learn.org/stable/modules/ensemble.html>
* <https://neptune.ai/blog/gradient-boosted-decision-trees-guide>
* <https://medium.com/%40toprak.mhmt/gradient-boosting-and-weak-learners-1f93726b6fbd>
* <https://machinelearningmastery.com/strong-learners-vs-weak-learners-for-ensemble-learning/>