

Redes Neuronales

Una Introducción

Fernando Arias-Rodríguez

Banco Central de Bolivia

30 de agosto de 2024



- 1 Introducción
- 2 Conceptos Básicos
- 3 Entrenando redes neuronales
- 4 Forward y Backpropagation
- 5 Guía práctica para implementar ANN

- 1 Introducción
- 2 Conceptos Básicos
- 3 Entrenando redes neuronales
- 4 Forward y Backpropagation
- 5 Guía práctica para implementar ANN

Introducción

La inteligencia artificial (IA) es un campo de estudio que busca darle capacidades cognitivas a los computadores, con el fin de que estos aprendan a resolver problemas.

Machine Learning (ML) es una rama de IA encargada de desarrollar formas de programar computadores con el fin de que estos solucionen problemas.

Las redes neuronales, conocidas como *Artificial Neural Networks (ANN)*, surgen con la idea de imitar la función cerebral encargada de pensar soluciones.

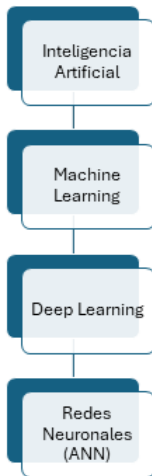
Introducción

ANN es un ejemplo de ML.

ANN es la piedra angular de lo que se conoce como *Deep Learning* (DL).

DL es un conjunto complejo de ANN, utilizados para diversas tareas como reconocimiento de imágenes, clasificación, regresión, entre otras.

Introducción



- 1 Introducción
- 2 Conceptos Básicos
Tipos de ANN
- 3 Entrenando redes neuronales
- 4 Forward y Backpropagation
- 5 Guía práctica para implementar ANN

Conceptos Básicos

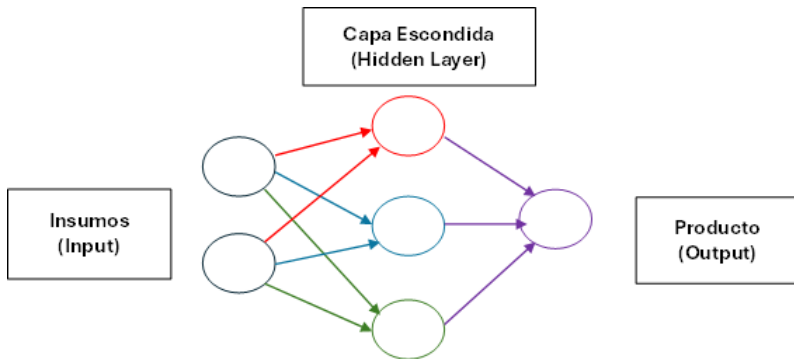
En ANN, la unidad central de procesamiento se define como **neurona**, la cual ejecuta una operación matemática determinística para generar un producto a partir de un conjunto de información de entrada.

Cada neurona ejecuta una operación simple y se activa si la señal recibida excede un umbral de activación.

La ANN es la agregación de resultados de todas las neuronas que la componen.

Conceptos Básicos

Esquemáticamente, una neurona se ve así:



Conceptos Básicos

A la arquitectura inicial, se agregan tres elementos:

- **pesos:** denotados por la letra w , estos parámetros definen la manera en la que los insumos afectan al producto. En otros términos:

$$y = \sum_{i=1}^I x_i w_i \quad (1)$$

donde I es igual a número de insumos disponibles.

Los pesos podrían asociarse con las pendientes en Mínimos Cuadrados Ordinarios.

Conceptos Básicos

- **sesgo**: es un parámetro adicional que ajusta el producto de cada neurona, de acuerdo con los insumos y los pesos. Puede asociarse con la constante en una regresión lineal tradicional.

Con este nuevo elemento, la ecuación puede escribirse como:

$$y = \sum_{i=1}^I x_i w_i + \text{sesgo} \quad (2)$$

Conceptos Básicos

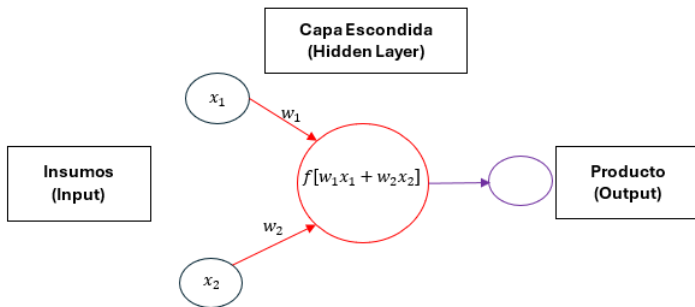
- **función de activación:** es una función matemática que se aplica a la definición de y , con la misión de activar la señal que la neurona transmite cuando el problema así lo requiera.

Con esto, nuestra expresión es igual a:

$$y = f(x) = f\left(\sum_{i=1}^I x_i w_i + \text{sesgo}\right) \quad (3)$$

Conceptos Básicos

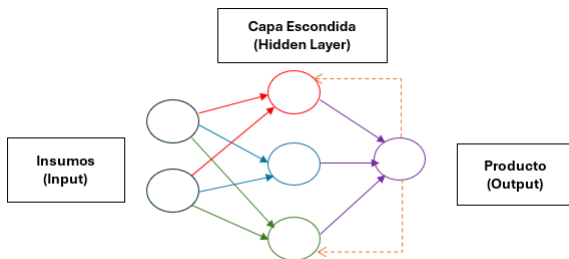
Incorporando los nuevos elementos, una red podría esquematizarse así:



- 1 Introducción
- 2 Conceptos Básicos
Tipos de ANN
- 3 Entrenando redes neuronales
- 4 Forward y Backpropagation
- 5 Guía práctica para implementar ANN

- Unidireccional, como en los diagramas ya presentados.
- Recurrentes, donde hay una retroalimentación entre las neuronas.

Un esquema de una ANN recurrente puede ser el siguiente:



- La recurrencia aparece cuando una ANN modela un comportamiento dinámico, como problemas de reconocimiento de patrones o, en estadística, modelación de series de tiempo.
- Estas ANN son difíciles de entrenar. Así, por lo general tienen pocas capas, para disminuir su complejidad.
- En economía se suelen usar la ANN conocidas como *Long Short Term Memory*.

- 1 Introducción
- 2 Conceptos Básicos
- 3 Entrenando redes neuronales**
 - Funciones de activación
 - Elección de funciones de activación
- 4 Forward y Backpropagation
- 5 Guía práctica para implementar ANN

Entrenando redes neuronales

- Entrenar una red neuronal significa introducir información muestral y reestimar los parámetros de manera que se pueda aproximar lo mejor posible el producto deseado.
- Existen dos tipos de entrenamientos:
 - ① Aprendizaje supervisado: se le suministra a la red tanto insumos como producto.
 - ② Aprendizaje no supervisado: se le suministra a la red solo insumos.

Entrenando redes neuronales

- Bajo aprendizaje supervisado es posible modificar los pesos de tal manera que se reduzca la diferencia entre lo observado y lo predicho por el modelo.
- En aprendizaje no supervisado, el modelo ajusta los pesos de manera autónoma, con la idea de que insumos similares deben producir productos similares.
- El proceso de proveer insumos, calcular pesos, sesgos y generar un resultado con funciones de activación se conoce como *epoch*. Para entrenar a una red es necesario implementar muchos *epochs* - decenas de miles incluso.

- 1 Introducción
- 2 Conceptos Básicos
- 3 Entrenando redes neuronales**
 - Funciones de activación
 - Elección de funciones de activación
- 4 Forward y Backpropagation
- 5 Guía práctica para implementar ANN

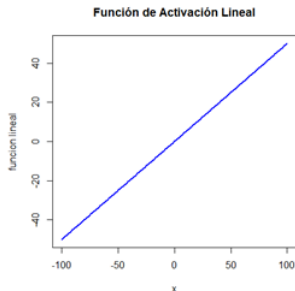
- Una función de activación es una función matemática que convierte los insumos en productos, ya sean para alimentar otra neurona o para arrojar la solución final.
- Sin una función de activación, una red neuronal funcionaría de manera similar a una función lineal.
- Sin embargo, las redes neuronales intentan solucionar problemas **no lineales y de naturaleza compleja**.
- En esta subsección se presentarán las funciones de activación más usadas.

Función lineal

- Se representa a través de la fórmula:

$$y = f(x) = x$$

- Gráficamente puede verse como:



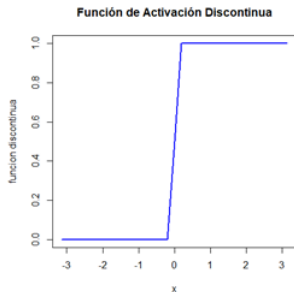
- Es usada como función de activación de la capa de producto.

Función escalonada

- Su forma funcional está dada por

$$f(x) = \begin{cases} 0 & \text{cuando } x < 0 \\ 1 & \text{cuando } x \geq 0 \end{cases}$$

- Gráficamente, esta luce como:



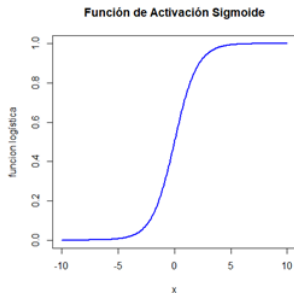
- Esta función es útil para problemas de naturaleza binaria: 0 valores negativos; 1 positivos.

Función sigmoide

- Es una de las más usadas. Su forma funcional está dada por

$$f(x) = \frac{1}{(1 + \exp(-x))}$$

- Gráficamente, esta función se ve así:



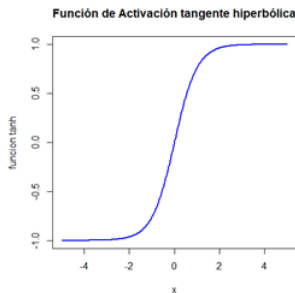
- Esta ajusta cualquier insumo a un valor entre 0 y 1, como un modelo logístico.

Función tangente hiperbólica

- Esta es una versión reescalada de la sigmoide. Su forma funcional está dada por

$$f(x) = \tanh(x)$$

- Gráficamente luce como:



- Esta función es no lineal y definida en el rango $(-1, 1)$.

Función rectificadora lineal

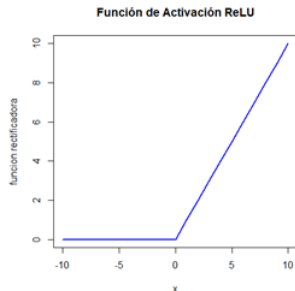
- Las funciones sigmoide y tangente hiperbólica sufren del problema de pendiente perdida (*missing slope*).
- el problema: a medida que se añaden capas a la red, los gradientes de la función de pérdida se acercan a cero, haciendo que esta sea más difícil de entrenar.
- Estas funciones generan ese problema, al comprimir el espacio de información de los insumos a un espacio mucho más reducido (0 a 1 o -1 a 1).

Función rectificadora lineal

- Converge más rápido y soluciona el problema expuesto. Su forma funcional está dada por

$$f(x) = \begin{cases} 0 & \text{cuando } x < 0 \\ x & \text{cuando } x \geq 0 \end{cases}$$

- Puede escribirse sucintamente como $f(x) = \max(0, x)$.
- También se conoce como ReLU y gráficamente luce como:



Función *softplus*

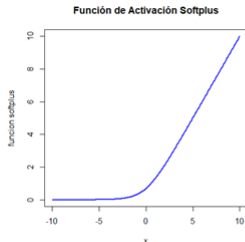
- Es una variación de ReLU. Su forma funcional está dada por

$$f(x) = \ln(1 + \exp(x))$$

- Lo interesante de esta función es que su primera derivada es igual a la función de activación sigmoide:

$$f'(x) = \frac{\exp(x)}{[1 + \exp(x)]} = \frac{1}{[1 + \exp(-x)]}$$

- Gráficamente luce como:



- 1 Introducción
- 2 Conceptos Básicos
- 3 Entrenando redes neuronales**
 - Funciones de activación
 - Elección de funciones de activación
- 4 Forward y Backpropagation
- 5 Guía práctica para implementar ANN

Una ANN estándar utiliza la siguiente configuración para sus funciones de activación:

- En las capas ocultas se prefiere el uso de ReLU.
- Para productos de naturaleza logística, la capa de producto puede seguir una función *softmax*, que es igual a una función multinomial:

$$f(x_i) = \frac{\exp(x_i)}{\sum_{j=0}^k \exp(x_j)}, i = 0, 1, \dots, k.$$

- Si se modela el problema como una regresión, la función de activación del producto es una lineal.

- 1 Introducción
- 2 Conceptos Básicos
- 3 Entrenando redes neuronales
- 4 Forward y Backpropagation**
- 5 Guía práctica para implementar ANN

Forward propagation

- Es el procesamiento de la red desde la capa de insumos a las capas escondidas y luego a la capa de producto.
- En cada capa se calcula la cantidad $\sum_i w_i x_i + \text{sesgo}$ y con la función de activación se llega al producto que va a otra neurona o se convierte en el producto final.
- Este procedimiento es estándar para ANN con un gran número de capas ocultas. Estas redes se conocen como Deep Neuronal Network (DNN).

Forward propagation

- En la última capa es necesario computar el error, diferencia entre lo observado y predicho.
- Este error es el insumo para recalcular los pesos y sesgos que se usarán en la nueva etapa de *Forward Propagation*.
- En la minimización del error entra a jugar el procedimiento de *Backpropagation*.

Backpropagation

- Utiliza las derivadas parciales de la función de activación en cada neurona para identificar la pendiente en la dirección de cada uno de los pesos.
- El gradiente sugiere la intensidad en la que cambiará el error ante un cambio en los pesos.
- *Backpropagation* modifica los pesos hasta que la reducción en los errores sea óptima. En cada etapa de interacción (*Epoch*), el algoritmo modifica los pesos en una cantidad conocida como la tasa de aprendizaje o *learning rate*.

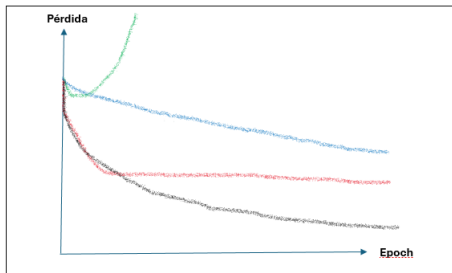
Backpropagation

La tasa de aprendizaje es un parámetro que determina el tamaño del paso en el que este se mueve hacia la minimización de la función de pérdida.

Este parámetro determina el tamaño del ajuste necesario para reducir los errores vs la rapidez en la convergencia del algoritmo.

Backpropagation

Gráficamente, la interacción entre la variación de la función de pérdida y la cantidad de repeticiones del proceso de optimización luce así:



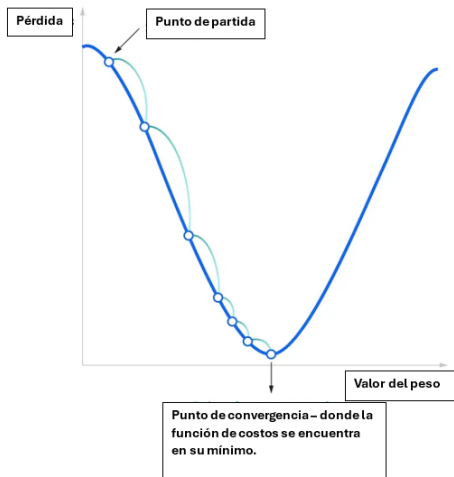
- Una tasa de aprendizaje alta (verde) es perjudicial para el ajuste de la red.
- Una tasa muy baja (azul) termina por producir un estancamiento en la optimización.
- Una tasa adecuada (negra) produce un ajuste suave en la minimización del error ante cada *epoch*.

Gradient Descent

- Mismo principio que la minimización de errores durante mínimos cuadrados ordinarios.
- Se tiene una función de error convexa y se busca encontrar el mínimo global de esa función, a partir del procedimiento:
 - 1 Se tiene valores para pesos y sesgos. Se calcula la pendiente de la función de activación con esos puntos, usando el gradiente.
 - 2 Se calcula cuánto deben cambiar estos parámetros para tener, en el siguiente paso (o siguiente *epoch*), una pendiente menos empinada.
 - 3 El procedimiento continua hasta encontrar la pendiente que asegure estar lo más cercano al mínimo. El algoritmo para si esto sucede.

Gradient Descent

Gráficamente, este procedimiento puede verse así:



Gradient Descent

Este procedimiento puede aplicarse al total de la muestra de entrenamiento o a una muestra de esta.

- *Full Batch Gradient Descent* usa toda la muestra de entrenamiento. Este acercamiento es más eficiente.
- *Stochastic Gradient Descent* utiliza una muestra aleatoria de la muestra de entrenamiento. Este acercamiento es más rápido.

- 1 Introducción
- 2 Conceptos Básicos
- 3 Entrenando redes neuronales
- 4 Forward y Backpropagation
- 5 Guía práctica para implementar ANN

En la construcción de una ANN, se siguen los pasos:

- 1 Disponer los datos de entrada en matrices (hecho por todos los programas).
- 2 Construir una arquitectura para la ANN.
- 3 Inicializar los pesos y sesgos con datos aleatorios - valores iniciales.
- 4 Aplicar los insumos a la ANN.
- 5 Calcular el producto final de cada neurona, a partir de las capas escondidas y la función de activación.

- 6 Calcular el error (la diferencia entre lo observado y lo predicho por el modelo).
- 7 Usar el error calculado para computar señales de error, usando la derivada parcial de la función de activación.
- 8 Usar las señales de error para calcular ajustes en los pesos y sesgos.
- 9 Aplicar los ajustes a los pesos y sesgos.
- 10 repetir los pasos 4 a 9 para cada patrón de entrenamiento o hasta que se minimice el error.

- Los pasos 4 y 5 corresponde a *Forward Propagation*.
- Los pasos 6 a 9 conformen el proceso de *Backpropagation*.
- Los pasos 4 a 10 conforman lo que se conoce como un ciclo de entrenamiento o *epoch*.