Explicación Perceptrón NetLogo

* ¿Que es un perceptron ?

El perceptrón es una neurona artificial la cual esta creada para poder evaluar cómo es que los datos se comportan detectando tendencias o características de los datos de entrada.

Es un algoritmo creado para el aprendizaje supervisado el cual permite que las neuronas artificiales aprendan y traten los elementos los cuales son entregados y posteriormente nos entregue el valor esperado ya aprendido por la propia neurona y que posteriormente pueda generar predicciones por medio de situaciones o datos de los cuales ya ha aprendido.

* ¿Como funciona un perceptron y por que se rige ?

El coeficiente busca siempre los coeficientes de un peso optimo para determinar si la neurona se “enciende “ o no .

**¿Cómo funciona?**

1. **Entradas:** El perceptrón recibe varias entradas (datos), cada una con un peso asociado. Estos pesos determinan la importancia de cada entrada en la decisión final.
2. **Suma ponderada:** Multiplica cada entrada por su peso correspondiente y suma todos los resultados.
3. **Función de activación:** El resultado de la suma se pasa a una función de activación. Esta función decide si la neurona "se activa" o no, es decir, si la salida es 1 (activada) o 0 (no activada). La función de activación más común es la función escalón, que devuelve 1 si la suma ponderada es mayor o igual a un umbral, y 0 en caso contrario.
4. **Salida:** La salida del perceptrón es el resultado de la función de activación.

**¿Por qué se rige?**

El perceptrón se rige por los **pesos** asociados a cada entrada. Estos pesos se ajustan durante el proceso de aprendizaje, para que el perceptrón pueda realizar predicciones más precisas. El objetivo del aprendizaje es encontrar los pesos óptimos que minimicen el error entre las predicciones del perceptrón y los valores reales.

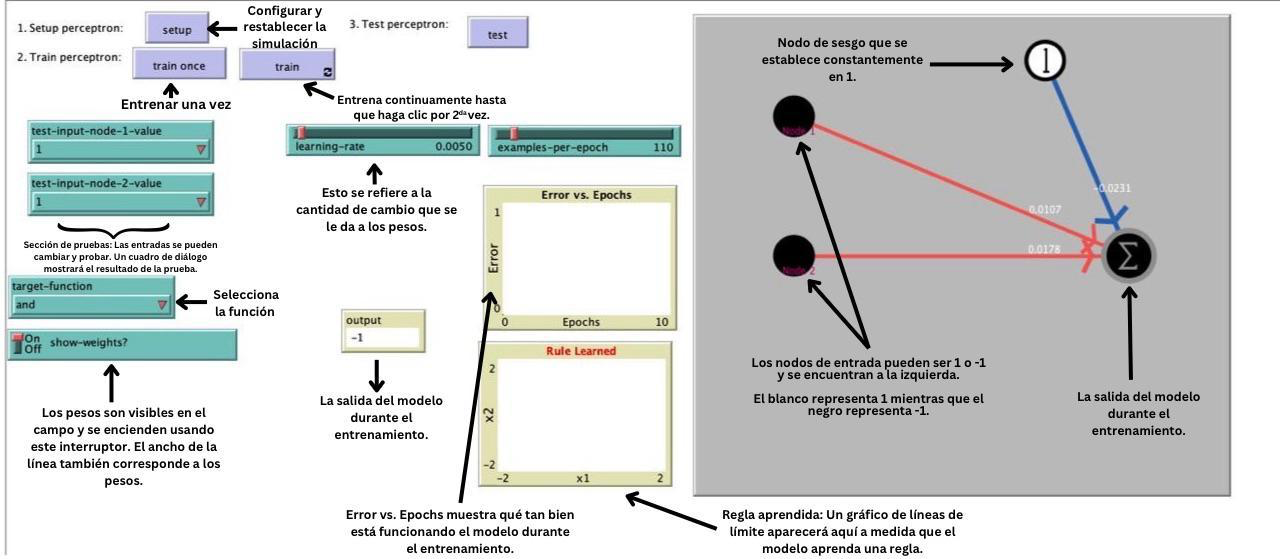


* ¿Para que sirve un perceptron ?

Los perceptrones se utilizan para resolver problemas de clasificación binaria, es decir, para clasificar datos en dos categorías (por ejemplo, spam o no spam, imagen de un gato o de un perro). Aunque un solo perceptrón es limitado, cuando se combinan en redes neuronales más grandes, pueden resolver problemas mucho más complejos.

* Explicacion programa de mesa

En la siguiente imagen se explica como funciona el perceptron hecho en netlogo y como es que cada boton o desliizador refiere a algo .



* Explicacion Codigo Mesa

Utiliza un algoritmo de aprendizaje supervisado para clasificar entradas binarias (1 o -1) basándose en operaciones lógicas como OR, AND y XOR.

Componentes Principales

* **Nodos de Entrada y Sesgo**: Los nodos de entrada son las unidades que reciben los valores (ya sea 1 o -1) y pasan esa información al nodo de salida. También hay un nodo de sesgo que siempre tiene un valor de activación de 1. Este nodo es importante porque ayuda a mejorar la precisión del perceptrón.
* **Nodo de Salida (Perceptrón)**: Este nodo toma las señales que vienen de los nodos de entrada y del sesgo. Luego, calcula su activación sumando las entradas multiplicadas por sus respectivos pesos. Usa una función de signo para decidir si la salida será 1 o -1, dependiendo de si la suma supera un cierto umbral.
* **Pesos en los Enlaces**: Los enlaces entre los nodos tienen pesos que se ajustan durante el proceso de entrenamiento. El objetivo es minimizar el error en las predicciones del perceptrón.

Entrenamiento y Activación

* **Entrenamiento (train)**: En cada ciclo de entrenamiento, se asignan valores aleatorios a los nodos de entrada y se calcula la activación del nodo de salida. Luego, el código ajusta los pesos en función del error entre la salida calculada y la respuesta correcta. Este ajuste depende de una tasa de aprendizaje que determina qué tan rápido se modifican los pesos.
* **Función de Activación (compute-activation)**: El nodo de salida calcula su activación multiplicando cada valor de entrada por su peso correspondiente y sumando todos esos productos. La función de signo decide si la salida será 1 o -1 según el umbral establecido.

Operaciones Lógicas y Pruebas

* **Operaciones Lógicas**: El modelo incluye funciones para realizar operaciones lógicas como OR, AND y XOR, que definen cómo debe comportarse el perceptrón ante diferentes combinaciones de entradas.
* **Pruebas (test)**: El código permite probar el perceptrón configurando manualmente las entradas y comparando el resultado con el valor esperado. Se informa si el perceptrón ha dado una respuesta correcta o incorrecta.
* Afectación en los resultados según cambios de variables

1. **Valores de Entrada x1*x*1​ y x2*x*2​ (1 o -1)**

Los valores de entrada x1*x*1​ y x2*x*2​ son cruciales para el entrenamiento del perceptrón. Dependiendo de estos valores, el perceptrón ajustará sus pesos para intentar clasificar correctamente las entradas. La función lógica que decidas implementar (como OR, AND o XOR) también influye en este proceso:

* Si x1*x*1​ y x2*x*2​ son siempre los mismos o cambian poco, el perceptrón tendrá menos oportunidades de aprender de diferentes ejemplos. Esto puede limitar su capacidad para generalizar.
* Para funciones lógicas como OR o AND, el perceptrón puede ajustarse fácilmente con ejemplos variados. Pero cuando se trata de XOR, necesita ver casos donde ambas entradas sean iguales y también distintos para aprender a clasificar correctamente.

1. **Tasa de Aprendizaje**

La tasa de aprendizaje es lo que determina cuánto se ajustan los pesos de los enlaces después de cada ejemplo:

* **Tasa alta**: Los pesos se ajustan rápidamente con cada error, lo que permite al perceptrón aprender rápido al principio. Sin embargo, si la tasa es demasiado alta, puede hacer que los pesos oscilen mucho y no lleguen a converger a valores óptimos, lo que genera inestabilidad.
* **Tasa baja**: Los ajustes a los pesos serán más pequeños, lo que hace que el aprendizaje sea más lento y gradual. Esto puede ser útil para llegar a soluciones estables, pero si es demasiado baja, el modelo podría necesitar muchas más épocas para alcanzar un rendimiento adecuado o incluso quedarse atrapado en valores subóptimos.

1. **Aprendizaje por Épocas**

El aprendizaje por épocas se refiere a cuántas veces el perceptrón pasa por el conjunto de ejemplos de entrenamiento. Cada época permite al modelo ajustar sus pesos según los datos:

* **Pocas épocas**: Si no se entrena lo suficiente, el perceptrón puede no ajustar los pesos correctamente, dejando un error alto. Esto es especialmente problemático si el problema necesita ver varios ejemplos para aprender patrones.
* **Muchas épocas**: Entrenar durante muchas épocas permite al perceptrón ajustar mejor los pesos y reducir el error. Sin embargo, esto también puede llevar a un sobreajuste si el modelo se expone demasiado a un conjunto específico de datos y pierde la capacidad de generalizar bien a nuevos datos.

Resumen

En resumen:

* Los valores x1*x*1​ y x2*x*2​ afectan la diversidad de ejemplos y ayudan al perceptrón a identificar patrones en diferentes combinaciones.
* La tasa de aprendizaje influye en la rapidez y estabilidad del aprendizaje, siendo clave para lograr una convergencia estable de los pesos.
* El número de épocas permite al perceptrón refinar su aprendizaje y encontrar un equilibrio entre aprender los ejemplos de entrenamiento y generalizar a nuevas entradas.