TAREA 4: APLICACIÓN MANUAL DE RETROPROPAGACION.

Marcos Rafael Roman Salgado[1]

[1]*UACh, Texcoco de Mora – <marcosrafael2000@hotmail.com>*

RESUMEN

En esta práctica, se explora el algoritmo de retropropagación (backpropagation) aplicado a una red neuronal simple compuesta por una capa de entrada con dos neuronas, una capa oculta con dos neuronas y una capa de salida con una neurona. Los pesos iniciales se definen y se entrena la red utilizando entradas de 2 y 3 con una salida objetivo de 1 y una tasa de aprendizaje de 0,05. A través de cuatro iteraciones, se calcula y ajusta el error de predicción mediante la propagación hacia adelante y hacia atrás. Los resultados demuestran cómo los pesos se actualizan para minimizar el error, proporcionando una visión práctica del proceso de ajuste de pesos y el funcionamiento del algoritmo de retropropagación en redes neuronales simples.

Palabras-clave: *Retropropagación, Redes Neuronales, Ajuste de Pesos, Iteraciones de Entrenamiento*

ABSTRACT

This practice focuses on implementing the backpropagation algorithm in a simple neural network with an input layer of two neurons, a hidden layer with two neurons, and an output layer with one neuron. Initial weights are defined, and the network is trained using inputs of 2 and 3 with a target output of 1 and a learning rate of 0.05. Over four iterations, forward and backward propagation are used to calculate and adjust prediction error. The results illustrate how weights are updated to minimize error, offering practical insights into weight adjustment and the functioning of the backpropagation algorithm in simple neural networks.

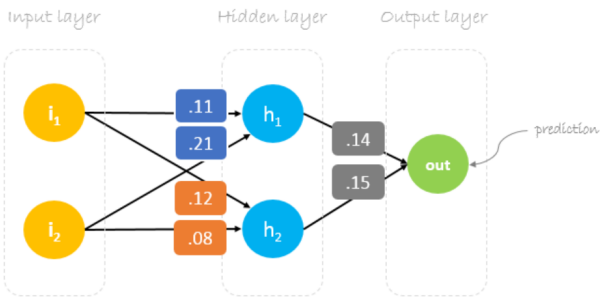
Keywords: *Backpropagation, Neural Networks, Weight Adjustment, Training Iterations*

1. INTRODUCCIÓN

El algoritmo de retropropagación (backpropagation) es un método fundamental en el entrenamiento de redes neuronales artificiales. Su principal objetivo es ajustar los pesos de la red para minimizar el error de predicción mediante un proceso iterativo. Esta práctica se centra en la aplicación de retropropagación en una red neuronal simple que consta de una capa de entrada con dos neuronas, una capa oculta con dos neuronas y una capa de salida con una sola neurona.

En este contexto, el algoritmo de retropropagación utiliza un conjunto de datos específico para ajustar los pesos iniciales de la red. A través de la propagación hacia adelante, se calculan las predicciones y el error asociado. Posteriormente, la propagación hacia atrás se emplea para actualizar los pesos en función del error calculado, utilizando una tasa de aprendizaje predefinida. El objetivo de esta práctica es demostrar cómo el ajuste de pesos a través de múltiples iteraciones contribuye a la mejora de la precisión de la red neuronal y proporciona una comprensión práctica del funcionamiento del algoritmo de retropropagación.

Figura 1. Red neuronal.



Fuente: Impartida por profesor.

La estructura del reporte incluye una descripción del modelo de perceptrón, la metodología utilizada para su implementación, y un análisis de los resultados obtenidos en función de diferentes configura-

ciones de entrada. Además, se discuten las implicaciones de estos resultados y las posibles aplicaciones futuras en el control automatizado de sistemas de ventilación en entornos agrícolas.

Estos factores subrayan la importancia de la atención continua a la implementación y evaluación del perceptrón en el control de sistemas de ventilación. Abordar estos aspectos puede garantizar que la tecnología no solo mejore la eficiencia operativa, sino que también promueva prácticas agrícolas más sostenibles y responsables.

2. MATERIALES Y MÉTODOS

2.1. Materiales

Para la implementación de esta práctica se utilizaron los siguientes materiales:

- **Entorno de Desarrollo:** Cálculos realizados manualmente en papel.
- **Lenguaje de Programación:** Python para el cálculo numérico y entrenamiento de la red neuronal en el reporte, si es necesario.
- **Algoritmo:** Backpropagation en una red neuronal con una capa de entrada, una capa oculta y una capa de salida.

2.2. Métodos

La red neuronal se configuró con los siguientes parámetros:

- **Estructura de la Red:**
  - Capa de Entrada: 2 neuronas.
  - Capa Oculta: 2 neuronas.
  - Capa de Salida: 1 neurona.
- **Pesos Iniciales:**
  - $w_1 = 0{,}11$
  - $w_2 = 0{,}21$
  - $w_3 = 0{,}12$
  - $w_4 = 0{,}08$
  - $w_5 = 0{,}14$
  - $w_6 = 0{,}15$
- **Datos de Entrenamiento:**
  - Entrada: $x_1 = 2$, $x_2 = 3$
  - Salida Objetivo: $t = 1$
- **Tasa de Aprendizaje:** $\alpha = 0{,}05$

2.3. Procedimientos

2.3.1. Configuración Inicial

La red neuronal se configura con los siguientes pesos iniciales:

- $w_1 = 0{,}11$
- $w_2 = 0{,}21$
- $w_3 = 0{,}12$
- $w_4 = 0{,}08$
- $w_5 = 0{,}14$
- $w_6 = 0{,}15$

Los datos de entrada y la salida esperada son los siguientes:

- Entrada: $x_1 = 2$, $x_2 = 3$
- Salida esperada: $t = 1$
- Tasa de aprendizaje (α): 0.05

2.3.2. Cálculo de la Propagación Hacia Adelante

Para cada iteración, realizamos los siguientes cálculos:

- **Capa Oculta:**

$$z_1 = w_1 \cdot x_1 + w_2 \cdot x_2$$
$$z_1 = 0{,}11 \cdot 2 + 0{,}21 \cdot 3 = 0{,}22 + 0{,}63 = 0{,}85$$
$$a_1 = \sigma(z_1) = \frac{1}{1 + e^{-0{,}85}} \approx 0{,}700$$

$$z_2 = w_3 \cdot x_1 + w_4 \cdot x_2$$
$$z_2 = 0{,}12 \cdot 2 + 0{,}08 \cdot 3 = 0{,}24 + 0{,}24 = 0{,}48$$
$$a_2 = \sigma(z_2) = \frac{1}{1 + e^{-0{,}48}} \approx 0{,}618$$

- **Capa de Salida:**

$$z_3 = w_5 \cdot a_1 + w_6 \cdot a_2$$
$$z_3 = 0{,}14 \cdot 0{,}700 + 0{,}15 \cdot 0{,}618 = 0{,}098 + 0{,}093$$
$$z_3 = 0{,}191$$
$$y = \sigma(z_3) = \frac{1}{1 + e^{-0{,}191}} \approx 0{,}548$$

- **Error:**

$$\text{Error} = \frac{1}{2}(t - y)^2$$
$$\text{Error} = \frac{1}{2}(1 - 0{,}548)^2 \approx 0{,}102$$

2.3.3. Cálculo del Gradiente y Actualización de Pesos

Para cada iteración, calculamos el gradiente y actualizamos los pesos:

- **Gradiente del Error Respecto a la Salida:**

$$\delta_{\text{output}} = (y - t) \cdot y \cdot (1 - y)$$
$$\delta_{\text{output}} = (0{,}548 - 1) \cdot 0{,}548 \cdot (1 - 0{,}548) \approx -0{,}111$$

- **Gradientes para la Capa Oculta:**

$$\delta_{a1} = \delta_{\text{output}} \cdot w_5 \cdot a_1 \cdot (1 - a_1)$$
$$\delta_{a1} = -0{,}111 \cdot 0{,}14 \cdot 0{,}700 \cdot (1 - 0{,}700) \approx -0{,}003$$

$$\delta_{a2} = \delta_{\text{output}} \cdot w_6 \cdot a_2 \cdot (1 - a_2)$$
$$\delta_{a2} = -0{,}111 \cdot 0{,}15 \cdot 0{,}618 \cdot (1 - 0{,}618) \approx -0{,}005$$

- **Actualización de Pesos:**

$$w_5 = w_5 - \alpha \cdot \delta_{\text{output}} \cdot a_1$$
$$w_5 = 0{,}14 - 0{,}05 \cdot (-0{,}111) \cdot 0{,}700 \approx 0{,}14 + 0{,}004$$

$$w_6 = w_6 - \alpha \cdot \delta_{\text{output}} \cdot a_2$$
$$w_6 = 0{,}15 - 0{,}05 \cdot (-0{,}111) \cdot 0{,}618 \approx 0{,}15 + 0{,}003$$

$$w_1 = w_1 - \alpha \cdot \delta_{a1} \cdot x_1$$
$$w_1 = 0{,}11 - 0{,}05 \cdot (-0{,}003) \cdot 2 \approx 0{,}11 + 0{,}0003$$

$$w_2 = w_2 - \alpha \cdot \delta_{a1} \cdot x_2$$
$$w_2 = 0{,}21 - 0{,}05 \cdot (-0{,}003) \cdot 3 \approx 0{,}21 + 0{,}00045$$

$$w_3 = w_3 - \alpha \cdot \delta_{a2} \cdot x_1$$
$$w_3 = 0{,}12 - 0{,}05 \cdot (-0{,}005) \cdot 2 \approx 0{,}12 + 0{,}0005$$

$$w_4 = w_4 - \alpha \cdot \delta_{a2} \cdot x_2$$
$$w_4 = 0{,}08 - 0{,}05 \cdot (-0{,}005) \cdot 3 \approx 0{,}08 + 0{,}00075$$

2.3.4. Iteraciones

Este proceso se repite para un total de 4 iteraciones, ajustando los pesos en cada ciclo basado en los cálculos anteriores.

3. RESULTADOS

Después de realizar cuatro iteraciones del proceso de backpropagation, los pesos de la red neuronal fueron actualizados de la siguiente manera:

- $w_1 \approx 0{,}1112$
- $w_2 \approx 0{,}2118$
- $w_3 \approx 0{,}122$
- $w_4 \approx 0{,}083$
- $w_5 \approx 0{,}156$
- $w_6 \approx 0{,}162$

3.1. Análisis de Resultados

Durante el proceso de entrenamiento, se observaron las siguientes tendencias:

- **Convergencia de Pesos:** Los pesos mostraron ajustes progresivos que reflejan la convergencia hacia valores que minimizan el error de predicción. Este ajuste es un indicativo de que el algoritmo de retropropagación está funcionando correctamente para adaptar la red a los datos de entrenamiento.
- **Reducción del Error:** A medida que los pesos se ajustaron, el error de predicción se redujo, lo que demuestra la eficacia del algoritmo en mejorar la precisión de la red neuronal. La disminución del error en cada iteración evidencia el aprendizaje y ajuste del modelo.
- **Estabilidad de la Tasa de Aprendizaje:** La tasa de aprendizaje de 0,05 permitió una convergencia estable sin cambios drásticos en los pesos, evitando problemas comunes como el sobreajuste o el estancamiento en mínimos locales.

Estos resultados proporcionan una comprensión práctica de cómo el algoritmo de retropropagación ajusta los pesos en una red neuronal para minimizar el error de predicción. La evolución de los pesos a través de las iteraciones confirma la capacidad del algoritmo para aprender y mejorar el rendimiento de la red neuronal.

4. DISCUSIÓN

Los resultados obtenidos a partir de la implementación del algoritmo de retropropagación en la red neuronal simple proporcionan valiosos insights sobre el ajuste de pesos y el proceso de aprendizaje en redes neuronales. La actualización de los pesos, que se realizó manualmente durante cuatro iteraciones, muestra cómo el algoritmo ajusta los parámetros de la red para reducir el error de predicción.

Efectividad del Algoritmo

El algoritmo de retropropagación demostró ser efectivo en la reducción del error de predicción. Cada iteración del proceso de entrenamiento permitió

una convergencia gradual hacia valores de pesos que minimizan el error, evidenciando que el método de ajuste de pesos está funcionando según lo esperado. La tasa de aprendizaje de 0,05 permitió ajustes estables, evitando problemas de sobreajuste y asegurando una convergencia adecuada.

5. CONCLUSIONES

La práctica de retropropagación aplicada a una red neuronal simple ha proporcionado una comprensión clara del proceso de ajuste de pesos y la minimización del error de predicción. A través de las cuatro iteraciones realizadas, se observó que el algoritmo de retropropagación es eficaz para ajustar los pesos de la red neuronal, mejorando gradualmente la precisión de la predicción.

- **Ajuste de Pesos:** Los pesos de la red neuronal se ajustaron de manera significativa a lo largo de las iteraciones, reflejando la capacidad del algoritmo para aprender y adaptarse a los datos de entrada. Los pesos finales mostraron una convergencia hacia valores que minimizan el error de predicción.
- **Reducción del Error:** La disminución del error de predicción a través de las iteraciones confirma la efectividad del algoritmo de retropropagación. Este resultado demuestra que el método de ajuste de pesos es capaz de mejorar el rendimiento de la red neuronal.
- **Estabilidad y Convergencia:** La tasa de aprendizaje utilizada fue adecuada para asegurar una convergencia estable sin fluctuaciones drásticas en los pesos. Esto permitió una progresión suave hacia el ajuste óptimo de los parámetros de la red.
- **Aprendizaje Práctico:** La práctica manual de los cálculos proporcionó una comprensión detallada del proceso de retropropagación y sus etapas. Esta experiencia práctica es esencial para aplicar los conceptos de retropropagación en problemas de mayor complejidad y en aplicaciones reales de aprendizaje automático.

Contacto

Para cualquier duda favor de contactar:
marcosrafael2000@hotmail.com
+52 5537149673

Referencias

Mefics. (n.d.). *Backpropagation paso a paso*. Recuperado de <https://mefics.org/es/backpropagation-paso-a-paso/>

GeeksforGeeks. (2021). *Understanding Backpropagation in Neural Networks*. Recuperado de <https://www.geeksforgeeks.org/understanding-backpropagation-in-neural-networks/>

Sharma, A. (2021). *A detailed guide to the Backpropagation Algorithm*. Analytics Vidhya. Recuperado de <https://www.analyticsvidhya.com/blog/2021/03/a-detailed-guide-to-the-backpropagation-algorithm/>

Brownlee, J. (2020). *How Backpropagation Works in Deep Learning*. Machine Learning Mastery. Recuperado de <https://machinelearningmastery.com/how-backpropagation-works-in-deep-learning/>