Facultad de Cs. de la Computación

Benemérita Universidad Autónoma de Puebla

"Pensar bien, para vivir mejor"

Facultad Ciencias de la Computación

Técnicas de Inteligencia Artificial

Examen Parcial 1

- Profesor: Luis Ernesto Valencia

• Alumno: Antonio Ambrosio Jesús Noé

Jueves 07/03/2024

Introducción

La inteligencia artificial (IA) ha tenido un impacto significativo en diversas áreas, incluyendo el aprendizaje automático y el análisis de datos. En este documento, se profundiza en el entrenamiento de redes neuronales artificiales mediante el algoritmo de retropropagación (backpropagation), una técnica clave en el aprendizaje supervisado de redes neuronales. El propósito principal de esta actividad es comprender en detalle cómo funciona el algoritmo de backpropagation y cómo se implementa en la programación de una red neuronal XOR básica. A lo largo de este documento, se describirá exhaustivamente el funcionamiento del algoritmo, su implementación práctica y la evaluación del rendimiento de la red neuronal entrenada.

Desarrollo

Encuadre teórico sobre redes neuronales y neuronas:

Las redes neuronales artificiales son modelos computacionales inspirados en el funcionamiento del cerebro humano. Están compuestas por capas de neuronas interconectadas, donde cada conexión tiene un peso que determina la influencia de una neurona en otra. Las redes neuronales son capaces de aprender a partir de ejemplos y de realizar tareas como clasificación, regresión, reconocimiento de patrones, entre otras.

Una neurona artificial se representa como una función matemática que toma entradas, realiza una suma ponderada de esas entradas y aplica una función de activación para producir una salida. La función de activación introduce no linealidad en la red, lo que le permite aprender relaciones complejas en los datos.

Descripción de los diferentes tipos de redes neuronales y neuronas:

Existen varios tipos de redes neuronales, cada una con una arquitectura y un propósito específico. Algunos ejemplos incluyen:

Redes neuronales feedforward: Son redes donde la información fluye en una dirección, de la entrada a la salida, sin bucles ni ciclos.

Redes neuronales recurrentes: Permiten el retroceso de la información, es decir, la salida de una capa puede volver como entrada a una capa anterior, lo que las hace adecuadas para modelar secuencias y datos temporales.

Redes neuronales convolucionales: Diseñadas para procesar datos que tienen una estructura de cuadrícula, como imágenes, donde las conexiones entre neuronas imitan la organización de la corteza visual de los animales.

Explicación breve de backpropagation:

El backpropagation, o retropropagación, es un algoritmo utilizado para entrenar redes neuronales. Consiste en propagar el error desde la salida de la red hacia atrás, ajustando los pesos de las conexiones para minimizar este error. Utiliza el gradiente descendente para encontrar el mínimo de una función de error, lo que permite que la red aprenda a partir de ejemplos etiquetados y mejore su capacidad para hacer predicciones precisas.

Entrenamiento

Captura del código

Editor - C:\Users\lunal\Downloads\Materias\7mo Semestre\TIA\V3.m

V3.m

```
% Datos de entrada y salida
X = [0 0; 0 1; 1 0; 1 1];
Y = [0; 1; 1; 0];

% Inicializar los pesos y bias
weights_input_hidden = rand(2, 3) - 0.5;
weights_hidden_output = rand(3, 1) - 0.5;
bias_hidden = rand(1, 3) - 0.5;
bias_output = rand(1, 1) - 0.5;

% Tasa de aprendizaje
learning_rate = 0.1;

% Vector para almacenar los errores
errors = zeros(1, 50000);

% Entrenamiento
for epoch = 1:50000
    % Forward pass
    hidden_layer_input = X * weights_input_hidden + bias_hidden;
    hidden_layer_output = sigmoid(hidden_layer_input);
    output_layer_input = hidden_layer_output * weights_hidden_output + bias_output;
    output = sigmoid(output_layer_input);

    % Calcular el error
    error = Y - output;

    % Almacenar el error cuadrático medio en el vector de errores
    errors(epoch) = mean(error.^2);

    % Backward pass
    d_output = error .* sigmoid_derivative(output);
    d_hidden_layer = (d_output * weights_hidden_output') .* sigmoid_derivative(hidden_layer_output);

    % Actualizar pesos y bias
    weights_hidden_output = weights_hidden_output + learning_rate * hidden_layer_output' * d_output;
    bias_output = bias_output + learning_rate * sum(d_output);
    weights_input_hidden = weights_input_hidden + learning_rate * X' * d_hidden_layer;
    bias_hidden = bias_hidden + learning_rate * sum(d_hidden_layer);
end

% Mostrar la salida después del entrenamiento
disp('Salida después del entrenamiento:');
disp(output);

% Graficar los errores
figure;
plot(errors);
title('Error cuadrático medio por época');
xlabel('Época');
ylabel('Error cuadrático medio');

% Función sigmoidal
function y = sigmoid(x)
    y = 1 ./ (1 + exp(-x));
end

% Función derivada de la sigmoidal
function y = sigmoid_derivative(x)
    y = x .* (1 - x);
end
```

Breve explicación:

Datos de entrada y salida:
Se definen los datos de entrada X (los cuatro posibles pares de entrada para XOR) y los datos de salida Y (los resultados esperados para cada par de entrada).

Inicialización de pesos y bias:
Se inicializan los pesos (weights_input_hidden, weights_hidden_output) y bias (bias_hidden, bias_output) de forma aleatoria entre -0.5 y 0.5.

Tasa de aprendizaje:
Se establece la tasa de aprendizaje learning_rate en 0.1.

Vector de errores:
Se crea un vector errors para almacenar los errores cuadráticos medios en cada época.

Entrenamiento de la red neuronal:
El bucle for itera 50,000 veces (épocas).

Para cada época, se realiza un pase hacia adelante (forward pass) para calcular la salida de la red.

Se calcula el error cuadrático medio entre la salida de la red y la salida esperada.

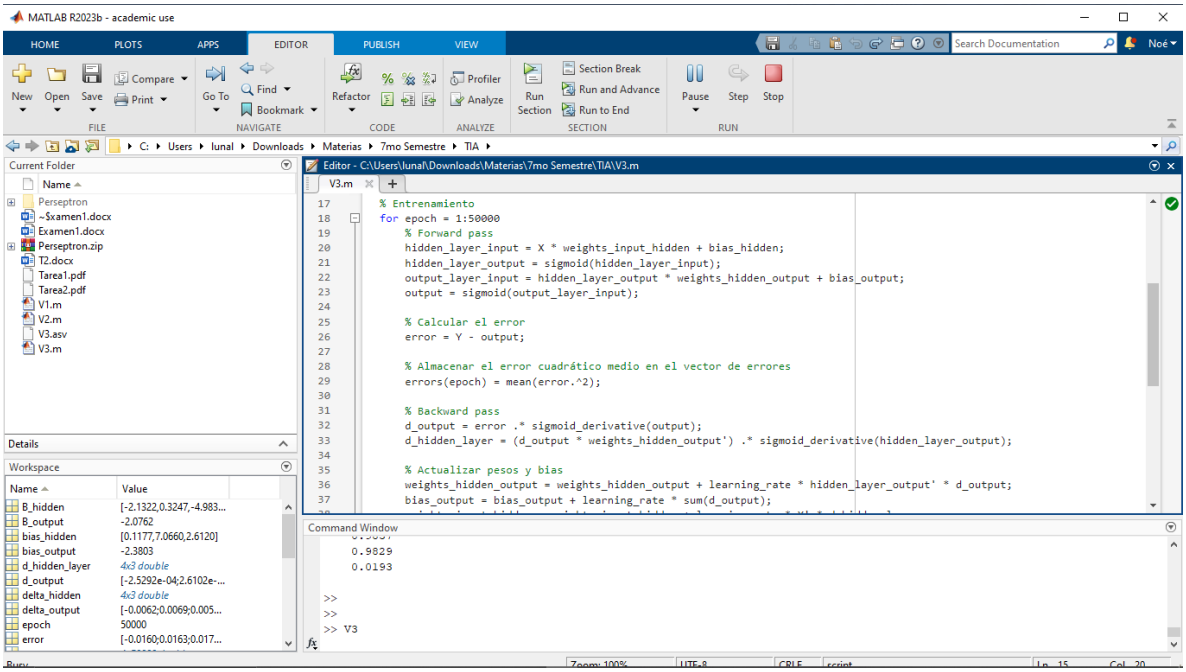
Se realiza un pase hacia atrás (backward pass) para ajustar los pesos y los bias de la red.

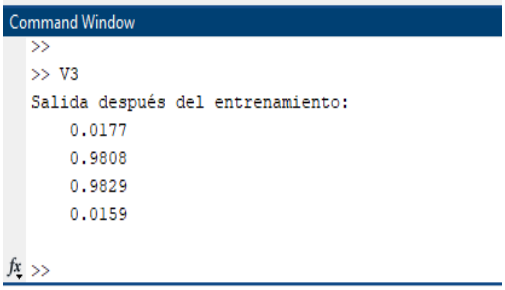
Se actualizan los pesos y bias usando el algoritmo de backpropagation y se almacena el error cuadrático medio en el vector errors.

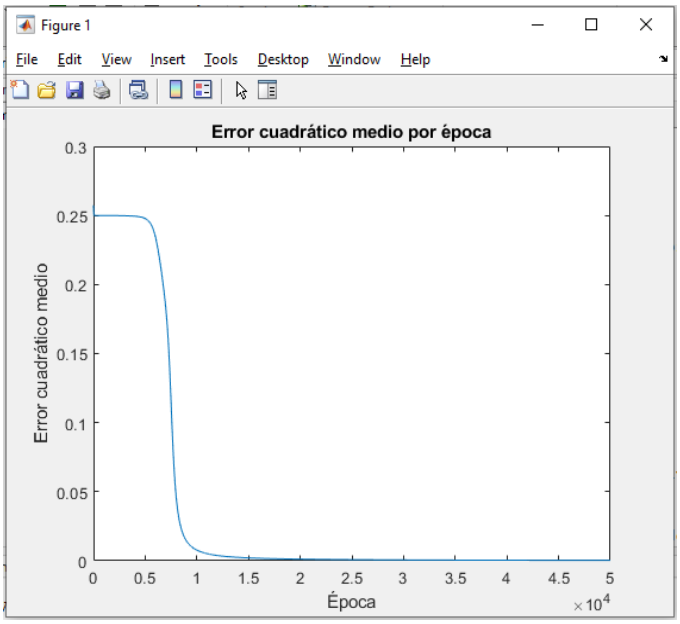
Visualización de la salida después del entrenamiento:
Se muestra en la consola de MATLAB la salida de la red después del entrenamiento.

Gráfica del error cuadrático medio por época:
Se grafica el vector errors para visualizar cómo disminuye el error cuadrático medio a lo largo de las épocas.

Ejecutando


MATLAB R2023b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Editor - C:\Users\lunal\Downloads\Materias\7mo Semestre\TIA\V3.m
V3.m
% Entrenamiento
for epoch = 1:50000
% Forward pass
hidden_layer_input = X * weights_input_hidden + bias_hidden;
hidden_layer_output = sigmoid(hidden_layer_input);
output_layer_input = hidden_layer_output * weights_hidden_output + bias_output;
output = sigmoid(output_layer_input);
% Calcular el error
error = Y - output;
% Almacenar el error cuadrático medio en el vector de errores
errors(epoch) = mean(error.^2);
% Backward pass
d_output = error .* sigmoid_derivative(output);
d_hidden_layer = (d_output * weights_hidden_output') .* sigmoid_derivative(hidden_layer_output);
% Actualizar pesos y bias
weights_hidden_output = weights_hidden_output + learning_rate * hidden_layer_output' * d_output;
bias_output = bias_output + learning_rate * sum(d_output);
Command Window
0.9829
0.0193
>>
>>
>> V3


Command Window
>>
>> V3
Salida después del entrenamiento:
0.0177
0.9808
0.9829
0.0159
>>



En la gráfica del error cuadrático medio por época, el eje "x" representa las épocas de entrenamiento, es decir, cuántas veces se ha presentado todos los ejemplos de entrenamiento a la red y ajustado los pesos. Por otro lado, el eje "y" muestra el error cuadrático medio (ECM), que dice qué tan cerca están las respuestas de la red de las respuestas reales.

La curva en la gráfica muestra cómo cambia el ECM a medida que la red aprende. Lo ideal es ver que esta curva disminuya a medida que aumentan las épocas, lo que indica que la red está aprendiendo y mejorando. Cuando la curva converge a un valor bajo y estable, significa que la red ha aprendido bien la función XOR y está produciendo resultados consistentes.

La gráfica del error cuadrático medio por época nos permite ver cómo está progresando el entrenamiento y qué tan bien está aprendiendo la red. Es una herramienta útil para monitorear la calidad del entrenamiento y asegurar de que la red esté aprendiendo correctamente.

Conclusión

En esta actividad, se exploró el entrenamiento de una red neuronal para la función XOR utilizando backpropagation. Se aprendió cómo ajustar los pesos y sesgos para minimizar el error cuadrático medio, lo que permitió lograr una convergencia exitosa.

La visualización del error por época fue crucial para monitorear el progreso y la calidad del entrenamiento. Esto enseñó la importancia de ajustar los hiperparámetros y la cantidad de épocas para obtener resultados óptimos.

El estudio y la aplicación práctica del algoritmo de retropropagación en una red neuronal básica para la función XOR ha sido fundamental para comprender los principios básicos del aprendizaje en redes neuronales.

Aspectos formales: Redacción y bibliografía

Mazur, M. (2015, 17 de marzo). A Step by Step Backpropagation Example. [Artículo]. Recuperado de https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Aggarwal, C. C. (2018). Neural Networks and Deep Learning: A Textbook. Springer.

BitBoss. (2019, 2 de mayo). Redes Neuronales - Backpropagation.

https://www.youtube.com/watch?v=boP3O89rErA

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Backpropagation: Theory, Architectures, and Applications. Psychology Press.