|  |
| --- |
| import numpy as np |
|  |  |
|  |  |
|  | # np.random.seed(0) |
|  |  |
|  | def sigmoid(x): |
|  | return 1 / (1 + np.exp(-x)) |
|  |  |
|  |  |
|  | def sigmoid\_derivative(x): |
|  | return x \* (1 - x) |
|  |  |
|  |  |
|  | # Input datasets |
|  | inputs = np.array([[0, 0], [0, 1], [1, 0], [1, 1]]) |
|  | expected\_output = np.array([[0], [1], [1], [0]]) |
|  |  |
|  | epochs = 10000 |
|  | lr = 0.1 |
|  | inputLayerNeurons, hiddenLayerNeurons, outputLayerNeurons = 2, 2, 1 |
|  |  |
|  | # Random weights and bias initialization |
|  | hidden\_weights = np.random.uniform(size=(inputLayerNeurons, hiddenLayerNeurons)) |
|  | hidden\_bias = np.random.uniform(size=(1, hiddenLayerNeurons)) |
|  | output\_weights = np.random.uniform(size=(hiddenLayerNeurons, outputLayerNeurons)) |
|  | output\_bias = np.random.uniform(size=(1, outputLayerNeurons)) |
|  |  |
|  | print("Initial hidden weights: ", end='') |
|  | print(\*hidden\_weights) |
|  | print("Initial hidden biases: ", end='') |
|  | print(\*hidden\_bias) |
|  | print("Initial output weights: ", end='') |
|  | print(\*output\_weights) |
|  | print("Initial output biases: ", end='') |
|  | print(\*output\_bias) |
|  |  |
|  | # Training algorithm |
|  | for \_ in range(epochs): |
|  | # Forward Propagation |
|  | hidden\_layer\_activation = np.dot(inputs, hidden\_weights) |
|  | hidden\_layer\_activation += hidden\_bias |
|  | hidden\_layer\_output = sigmoid(hidden\_layer\_activation) |
|  |  |
|  | output\_layer\_activation = np.dot(hidden\_layer\_output, output\_weights) |
|  | output\_layer\_activation += output\_bias |
|  | predicted\_output = sigmoid(output\_layer\_activation) |
|  |  |
|  | # Backpropagation |
|  | error = expected\_output - predicted\_output |
|  | d\_predicted\_output = error \* sigmoid\_derivative(predicted\_output) |
|  |  |
|  | error\_hidden\_layer = d\_predicted\_output.dot(output\_weights.T) |
|  | d\_hidden\_layer = error\_hidden\_layer \* sigmoid\_derivative(hidden\_layer\_output) |
|  |  |
|  | # Updating Weights and Biases |
|  | output\_weights += hidden\_layer\_output.T.dot(d\_predicted\_output) \* lr |
|  | output\_bias += np.sum(d\_predicted\_output, axis=0, keepdims=True) \* lr |
|  | hidden\_weights += inputs.T.dot(d\_hidden\_layer) \* lr |
|  | hidden\_bias += np.sum(d\_hidden\_layer, axis=0, keepdims=True) \* lr |
|  |  |
|  | print("Final hidden weights: ", end='') |
|  | print(\*hidden\_weights) |
|  | print("Final hidden bias: ", end='') |
|  | print(\*hidden\_bias) |
|  | print("Final output weights: ", end='') |
|  | print(\*output\_weights) |
|  | print("Final output bias: ", end='') |
|  | print(\*output\_bias) |
|  |  |
|  | print("\nOutput from neural network after 10,000 epochs: ", end='') |
|  | print(\*predicted\_output) |