Artificial Neural Network

This is the documentation of what I did and how I did my honors contract project which was to create an Artificial Neural Network using python.

Why did I choose this project? I chose this project because I wanted to know what machine learning is and what really goes inside the code and I think I got the surface level idea by creating a neural network.

Some of the python libraries I used are:

1. Numpy: a library for numerical computing with arrays and matrices.

2. Pandas : a library for data manipulation and analysis, providing easy-to-use data structures and data analysis tools

3. matplotlib.pyplot : a plotting library that provides a MATLAB-like interface to create a variety of plots such as line plots, scatter plots, and bar plots

Main(Imp) function explanations:

- init_params()

```
def init_params():
    W1 = np.random.rand(10, 784) - 0.5
    b1 = np.random.rand(10, 1) - 0.5
    W2 = np.random.rand(10, 10) - 0.5
    b2 = np.random.rand(10, 1) - 0.5
    return W1, b1, W2, b2
```

This function initializes the weights and biases for a two-layer neural network

- ReLU(Z)

```
def ReLU(Z):
    return np.maximum(Z, 0)
```

This function implements the rectified linear unit activation function aka ReLU, which is a common non-linear activation function used in neural networks. It takes an input Z and returns the element-wise maximum of Z and 0.

- softmax(Z)

```
def softmax(Z):
    A = np.exp(Z) / sum(np.exp(Z))
    return A
```

This function implements the softmax activation function, which is often used as the final activation function to normalize the output

- forward_prop()

```
def forward_prop(W1, b1, W2, b2, X):
    Z1 = W1.dot(X) + b1
    A1 = ReLU(Z1)
    Z2 = W2.dot(A1) + b2
    A2 = softmax(Z2)
    return Z1, A1, Z2, A2
```

Forward Propagation is the way to move from the Input layer (left) to the Output layer (right) in the neural network

- ReLU_deriv(Z)

```
def ReLU_deriv(Z):
    return Z > 0
```

This function computes the derivative

- One_hot

```
def one_hot(Y):
    one_hot_Y = np.zeros((Y.size, Y.max() + 1))
    one_hot_Y[np.arange(Y.size), Y] = 1
    one_hot_Y = one_hot_Y.T
    return one_hot_Y
```

This function converts a vector of class labels Y into a one-hot encoded matrix. It creates a matrix of zeros

- Backward_prop

```
def backward_prop(Z1, A1, Z2, A2, W1, W2, X, Y):
    one_hot_Y = one_hot(Y)
    dZ2 = A2 - one_hot_Y
    dW2 = 1 / m * dZ2.dot(A1.T)
    db2 = 1 / m * np.sum(dZ2)
    dZ1 = W2.T.dot(dZ2) * ReLU_deriv(Z1)
    dW1 = 1 / m * dZ1.dot(X.T)
    db1 = 1 / m * np.sum(dZ1)
    return dW1, db1, dW2, db2
```

Backpropagation is just a way of propagating the total loss back into the neural network to know how much of the loss every node is responsible for

- Update_params

```
def update_params(W1, b1, W2, b2, dW1, db1, dW2, db2, alpha):
    W1 = W1 - alpha * dW1
    b1 = b1 - alpha * db1
    W2 = W2 - alpha * dW2
    b2 = b2 - alpha * db2
    return W1, b1, W2, b2
```

This function updates the weight matrices and bias vectors using the gradients computed during the backward propagation

- Gradient_descent

```python
def gradient_descent(X, Y, alpha, iterations):
    W1, b1, W2, b2 = init_params()
    for i in range(iterations):
        Z1, A1, Z2, A2 = forward_prop(W1, b1, W2, b2, X)
        dW1, db1, dW2, db2 = backward_prop(Z1, A1, Z2, A2, W1, W2, X, Y)
        W1, b1, W2, b2 = update_params(W1, b1, W2, b2, dW1, db1, dW2, db2, alpha)
        if i % 10 == 0:
            print("Iteration: ", i)
            predictions = get_predictions(A2)
            print(get_accuracy(predictions, Y))
    return W1, b1, W2, b2
```

function is the main training function that implements the gradient descent algorithm to train the neural network. It takes in the input features X, the true class labels Y, the learning rate alpha, and the number of iterations to train for iterations. It initializes the parameters of the neural network using the init_params() function and then repeatedly performs forward propagation, backward propagation, and updates the parameters using the update_params() function.

Resources i used:

1. Neural networks from scratch in Pyhton(sentdex)
2. Building neural network from scratch
3. What is Neural Network?
4. Make your own neural network