libraries used

numpy

random

class ANN

__init__(self, layout)

self. layout

A *list* of integers describing the structure of the neural network. e.g., [3,4,4,2]

The first value is the input layer size. The last value is the output layer size.

The intermediate values (optional) are hidden layer sizes.

self.costs

A dictionary of all the available cost functions.

self.activations

A dictionary of all the available activation functions.

calls the ***build()*** method to initialise weights and biases of the model.

build(self)

Used to randomly initialise the weights and biases of the neural network between values of -0.5 and 0.5.

self.weights

A dictionary containing all the weight matrices of the neural network.

self.biases

A dictionary containing all the bias matrices of the neural network.

mse(self, a, y, deriv=False)

Mean Squared Error function.

$$MSE = \frac{1}{n}\Sigma(a-y)^2$$

n = length of a

deriv=False returns MSE

deriv=True returns: $a - y$

binary_cross_entropy(self, a, y, deriv=False)

Binary cross entropy function.

$$BCE = -\frac{1}{n}\Sigma(y\ log(a) + (1-y)log(1-a))$$

n = length of a

deriv=False returns BCE

deriv=True returns: $\frac{1}{n}(a-y)$

sigmoid(self, z, deriv=False)

The sigmoid activation function.

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

deriv=False return *σ(z)*

deriv=True returns: $\sigma'(z) = \sigma(z)(1-\sigma(z))$

fit(self, X, Y, activation='sigmoid', cost='mse', alpha=0.05, epochs=100)

Used to train the neural network model using features X and target Y. Stochastic gradient descent is used to optimise the model.

activation activation function to be used in each layer (default, sigmoid).

cost cost function to apply (default, mse).

alpha learning rate (default, 0.05).

epochs number of epochs (default, 100).

calls the ***forward()*** and ***backward()*** methods for respective propagations.

self.cost

The cost (error) of the model after the last epoch.

forward(self, x, activation)

Used for the forward propagation in the network.

x input features

activation activation function to be used in each layer

backward(self, a, z, activation, cost, y, alpha)

Used for the backward propagation in the network.

a the predicted activated outputs of the model

z the non-activated outputs of the model

activation activation function to be used in each layer

y target data

alpha learning rate

predict(self, X)

Returns the predicted values using the features X.

X testing features

score(self, X, Y)

Returns the accuracy of the model.

Calls the *predict()* method to predict the output using the features X.

X	testing features
Y	target values