Escuela Colombiana de Ingeniería Julio Garavito
Análisis Computacional de Datos
Final Project 2020-1
Profesor Cristian C. Garzón Alfonso

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

Instructions

1. The project must be submitted in Moodle until May Friday 15th at 11:59 PM.
2. A Python file named ***asserts.py*** is available to check if your implementation code is programmed correctly. Your grade will be based on the number of asserts that your code passes correctly.

Project Deliverables

1. **File 1:** A PDF file named ***FinalProjectReport_{Student1LastName}_{Student2Last Name}.pdf***[1] containing a brief introduction about Backpropagation, your answers to questions 1 and 2, and computed accuracy measure score values.
2. **File 2:** A Python file named ***project.py*** containing the classes and main implementing the source code as a solution to points 3, 4, and 5.
3. **File 3:** The provided text file ***data.txt***.

Project Definition

1. Write a class *errors* without a constructor containing a static method to compute the Mean Squared Error (MSE) between two vectors. The signature of this function must be ***mse(real, predicted)***. Use the formula shown below:

$$MSE(v_{real}, v_{forecasted}) = \sqrt{\sum_{i=1}^{N} \frac{(v_{real,i} - v_{forecasted,i})^2}{N}}$$

2. The sigmoid function with $\sigma = 1$ is defined as follows:

$$S(x) = \frac{1}{1 + e^{-x}}$$

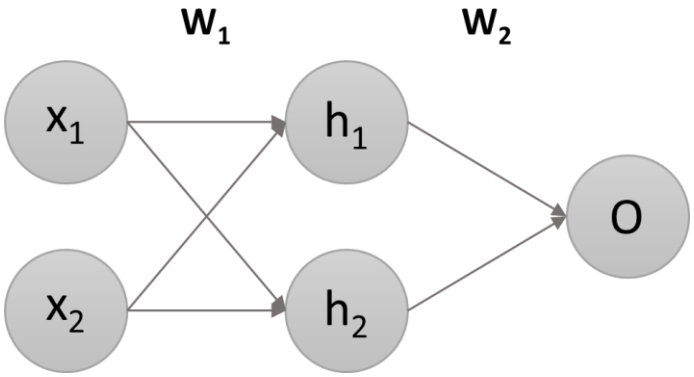
   a. Demonstrate that the derivative of $S(x)$, $\frac{dS}{dx}$ is equal to the following:

$$\frac{dS}{dx} = S(x)(1 - S(x))$$

   b. Write a class *activations* without a constructor containing two static methods to compute the sigmoid and its derivative, respectively. The signatures of these functions must be ***sigmoid(x)*** and ***sigmoid_derivative(x)***, respectively.

[1] Example: ***FinalProjectReport_GarzonAlfonso_RosaDeJesus.pdf***

3. Consider the following architecture:

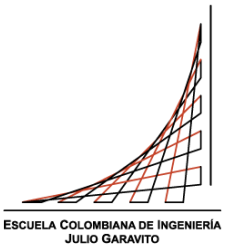


Write a class *network* with the following:

a. The constructor with four arguments, including the inputs, targets, number of neurons in the hidden layer, and learning rate. The last two arguments have default values of 2 and 0.5, respectively. The signature of this constructor must be ***__init__(self, inputs, targets, hidden_layer_neurons = 2, learning_rate = 0.5)***.
   i. This constructor must set the inputs, targets, hidden_layer_neurons, and learning_rate arguments to their associated instance field variables.
   ii. The constructor will also set the following instant field variables:
      1. ***weights_layer_1***: This variable represents the weight matrix of the neurons in the input layer and the neurons in the hidden layer, and it must be initiated with random values. Its dimension is the number of columns in the inputs by the number of hidden layer neurons plus one.
      2. ***weights_layer_2***: This variable represents the weight matrix between the neurons in the hidden layer and the neurons in the output layer, and it must be initiated with random values. Its dimension is the number of hidden layer neurons plus one by one, since our architecture will only have one output.
      3. ***activations***: This variable is a pointer to the class *activations*.
      4. ***errors***: This variable is a pointer to the class *errors*.

b. The *fit* method with one argument, the number of epochs[2]:
   i. For each epoch do the following:
      1. Forward Pass[3]
         a. Compute the dot product between the inputs and the weight matrix of the neurons in the input layer and the neurons in the hidden layer.

[2] Read Chapter 6 Section 1 to have a good understanding about backpropagation.
[3] The output of steps 1.b and 1.d must be held somewhere as they will be used in the backward pass.



   b. Activate the output in 1.a using the sigmoid function.
   c. Compute the dot product between the output in 1.b and the weight matrix of the neurons in the hidden layer and the neuron in the output layer.
   d. Activate the output in 1.c. using the sigmoid function.
2. **Backward Pass (Backpropagation)**
   a. Output Layer
      i. Compute the local error of the output layer by subtracting the predicted values in 1.d from their associated target values.
      ii. Deltas
         1. Compute the derivative of sigmoid for the predicted values in 1.d.
         2. Multiply the local error in 2.a.i by the output in 2.a.ii.1.
   b. Hidden Layer
      i. Compute the dot product between the output in 2.a.ii.2 and the transpose of the weight matrix of the neurons in the hidden layer and the neuron in the output layer.
      ii. Deltas
         1. Compute the derivative of sigmoid for the output in 1.b.
         2. Multiply the output in 2.b.i by the output in 2.b.ii.1.
   c. Adjusts weights
      i. Compute the dot product between the transpose of the output in 1.b and the output in 2.a.ii.2.
      ii. Multiply the output in 2.c.i by the learning rate and add it to the weight matrix between the neurons in the hidden layer and the neurons in the output layer.
      iii. Compute the dot product between the transpose of the inputs and the output in 2.b.ii.2.
      iv. Multiply the output in 2.c.iii by the learning rate and add it to the weight matrix of the neurons in the input layer and the neurons in the hidden layer.

c. The *predict* method with one argument, the inputs:
   i. Compute the dot product between the inputs and the weight matrix of the neurons in the input layer and the neurons in the hidden layer.
   ii. Activate the output in c.i using the sigmoid function.
   iii. Compute the dot product between the output in c.ii and the weight matrix of the neurons in the hidden layer and the neuron in the output layer.
   iv. Activate the output in c.iii using the sigmoid function.

   v. Return the output in c.iv.

4. Write the *main* method for your program:
   a. Read text files in row-column format:
      i. The first line contains the number of input-target data.
      ii. The subsequent lines contain the bias, x_1, x_2, and target separated by spaces.
      iii. **DO NOT MODIFY THE TEXT FILES PROVIDED TO YOU!**
   b. For this project, your inputs array must have a dimension equals to the number of input-target data by three that corresponds to the cardinality of a row vector containing its corresponding bias, x_1, and x_2 values.
   c. For instance, if your file contains the following:

```
4
1 0 0 0
1 0 1 0
1 1 0 0
1 1 1 1
```

   Your inputs array must have a dimension equals to (4, 3):

   Inputs =

```
[[1,0,0],
[1,0,0],
[1,0,0],
[1,0,0]]
```

   and your target array must have a dimension equals to (4, 1).

   Targets =

```
[[0],
[0],
[0],
[1]]
```

5. Compute the Precision, Recall, F1, and Accuracy score values.