Wind danger detector documentaion

Marc Cervera rosell

January 2022

Contents

Chapter 1

Abstract

This project, reads an input file with different values of wind gusts. The values are expressed in meters/second.
In the other side, the program has a file with the danger level associated to the wind gusts.
Once these two files are read, the neuronal network will have an entrance layer with the wind gusts and an output layer with the level of danger.
When the user wants, he/she can enter an specific value to know what level of danger has that concrete wind gust.

Chapter 2

'.txt' files

2.1 inputs.txt

This file has a few different values each of that represents a wind gust.
The range values is from 0m/s to 200m/s. It has been considered that a wind gust greater than 200m/s is not relevant because a greater value is considered a tornado. Each value in the file is a float.

2.2 outputs.txt

This is the complementary file of the above one. This file contents the danger level associated to his corresponding "brother" in "inputs.txt".
Here, the meaning of "brother" is that each line i of the "outputs.txt" file corresponds to the line i of the "inputs.txt" file. Imagine that the "inputs.txt" file has a value of 31.25 on the line 12. Then the corresponding danger level in the "outputs.txt" file will be the value on line 12.

Chapter 3

Code

3.1 *read data file* method

This first method, stores the value of each line of the file inside a list that will be returned at the end.
The fist step is to open in *reading* mode, the file given as the only mandatory parameter of the method. Secondly, for each line in the file (data + newline character), the algorithm, stores in a variable the splited line by the non mandatory parameter. This last operation, gives us a tuple seen as:

"Value"	*"newline character"*

Table 3.1: Tuple

Finally, before closing the read file and returning the data list, the algorithm appends to the data list only the float number.
Once the two files are read, we've to convert to the corresponding type the values because at this time, the lists that contents the inputs and the outputs, have *String* and not *float* or *int*. Therefore, the algorithm converts them with a loop and making a cast for each position of the list.
Outside of the method, the two lists are converted to arrays with the function *array* from *numpy*.

3.2 Neuronal network

3.2.1 Layers and model creation

This program has an input neuron, two hidden layers and an output layer.
The hidden ones are *Dense* layers with 10 neurons each one. To create this two hidden layers the *Tensorflow* library has been used.

Finally, the algorithm creates the model as a Sequential model and it adds all the layers that it has created.

3.2.2 Compiling

To compile the model, the programmer (myself) has to choose an optimizer and a loss function. In this case the choosen optimizer is *Adam* and the choosen loss function is *mean squared error*.
The optimizer, allows the layer to adjust biases and weights efficiently for it to learn and the number tells the optimizer how much to adjust the weights and the biases.
The choosen loss function, considers that a small amount of large errors is worse than a large amount of small errors.

3.2.3 Training the model

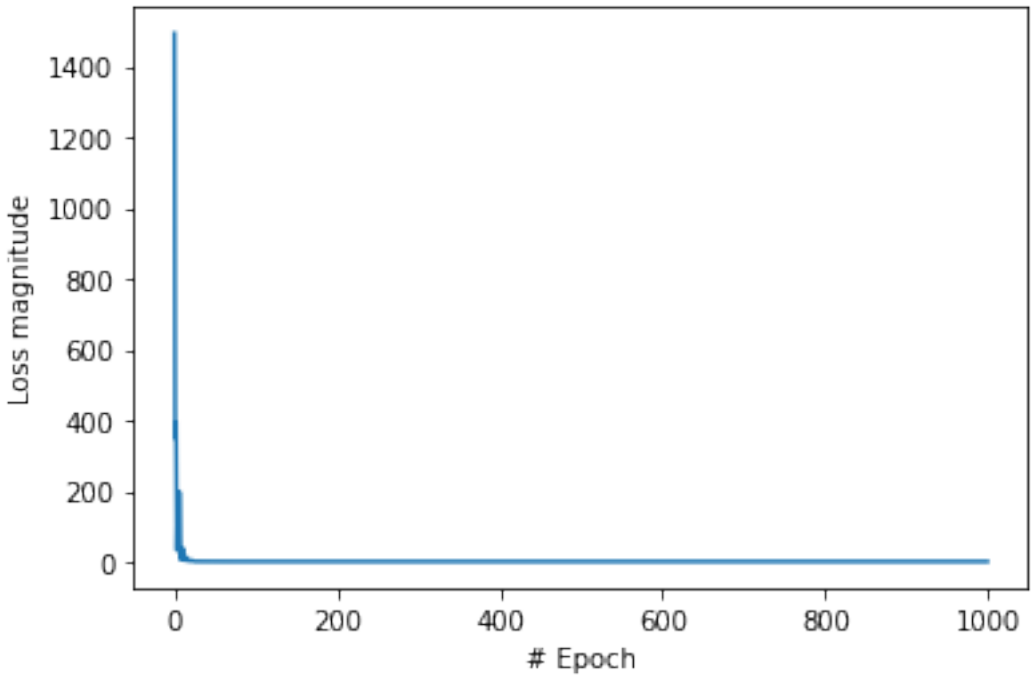
To train the model the algorithm uses the *fit* method. Two mandatory parameters are given to this method and two non mandatory more:

model.fit(inputs, outputs, epochs, verbose)

As it has been said, the two first parameters are mandatory, the other two not.

3.2.4 Printing the loss function graphic

Using the *matplotlib.pyplot* import, the program shows the graphic of the loss function.
The loss function tells the programmers/users how bad are the results of the network in each epoch. Hence, the more epochs are completed the fewer errors there are.


1400
1200
1000
800
600
400
200
0
Loss magnitude
0
200
400
600
800
1000
Epoch

Chapter 4

Playing with predictions

Finally, once all the work is done, a random user can use this program to obtain predictions.
To make a prediction, the user only has to change the parameter of the *model.predict* method and run the program. The danger level of the introduced wind gust will be shown.