RECIPE FOR AN EXQUISITE NEURAL NETWORK:

PREDICTION OF CARS FUEL EFFICIENCY

WHAT WE NEED:

- We will be using the "Auto MPG Data Set" of UCI Repository.

You can find the original files in this webpage:

https://archive.ics.uci.edu/ml/datasets/auto+mpg

- Microsoft Excel

HOW TO MAKE A NEURAL NETWORK?

Open the *.DATA file with excel. Yes, I know, the format [*.data] is confusing. Change it to [*.txt] and import it as text with defined width.

Add a Header with the labels. (Remember, only one row of header)

	A	B	C	D	E	F	G	H	I
1	Mpg	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model year	Origin	Car Name
2	18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
3	15	8	350	165	3693	11.5	70	1	buick skylark 320
4	18	8	318	150	3436	11	70	1	plymouth satellite
5	16	8	304	150	3433	12	70	1	amc rebel sst
6	17	8	302	140	3449	10.5	70	1	ford torino
7	15	8	429	198	4341	10	70	1	ford galaxie 500
8	14	8	454	220	4354	9	70	1	chevrolet impala

Something like this should we in your table.

NOW YOU HAVE TO CLEAN WELL YOUR DATA:

You can find a few rows with non-numerical values with an easy filter (Just a clue, they are identify whit an ["?"]). Delete those rows. We are going to ignore those values.

IN ORDER TO MAKE PROGRESS:

"Car name" will not be used during the training. You can delete it or leave it at the right. But MPG must be at the right of all the inputs. Something like this will work. Remember the configuration of the inputs! The formula is going to request it in the same order.

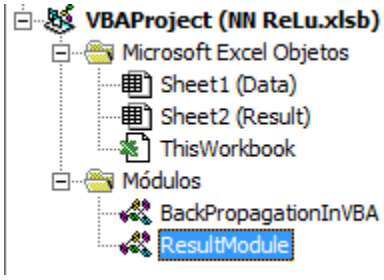
	A	B	C	D	E	F	G	H	I
1	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model year	Origin	Mpg	Car Name
2	4	121	113	2234	12.5	70	2	26	bmw 2002
3	4	98	83	2075	15.9	77	1	33.5	dodge colt m/m
4	4	119	100	2615	14.8	81	3	32.9	datsun 200sx
5	8	318	150	3940	13.2	76	1	13	plymouth volare premier v8
6	4	98	79	2255	17.7	76	1	26	dodge colt

VBA: WHERE THE MAGIC HAPPENS

Open the Visual Basic Application in the Programmer TAB. **You still don't have a Programmer TAB!** Don't worry; your secret is safe with me. Go to Options > Customize Ribbon and activate it.

Rename the sheets and paste the Code found in GitHub in two modules:

It should be something like this.



SET THE HIPERPARAMETERS:

LET'S CHECK ONE BY ONE. (YOU CAN FIND THIS IN THE PRIVATE SUB INITIALSETTINGS)

n_hid_layer = 'Number of hidden layers

I recommend starting with only one. And start adding layers. It's much more stable.

x_features = 'Number of x features

There are 7. (Number of Cylinders, Displacement, Horsepower, Weight, Acceleration, Model Year, Origin)

y_features = 'Number of y features

Only one, MPG, if you use another with more than one output remember that it will return a range. Accept the formula with Ctrl-Shift-Enter.

n_neurons_hid_layer(x) = 'Number of neurons in the hidden layer x (change the x for as many layers you have set)

I will start with 7. Just because is a pretty number.

SizeOfDataBase = 'The size of the database

Once you clean it there are 392 rows. You can save some as Test. I won't.

UploadOldModel = 'Train more an existing model.

It's the first time here. So the answer is False.

ChangeArquitecture = 'It's common, and recommended, start with a shallow neural network and start increasing/decreasing neurons/layers. If it's true, will recycle old weights and initialize new ones.

It's supposed to be False. But since UploadOldModel is False, it doesn't matter.

SamplesInBatch = 'Number of Samples on each Batch. Maximum of 1024

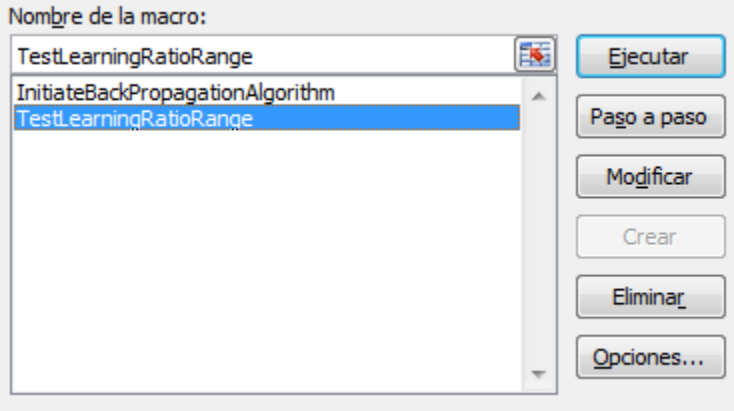
Leave it at maximum. If you want to try mini-batch GD change it. I'm afraid for those stochastic lovers that the value of one it's not available.

NumberOfIterations = 'Number of iterations

I will start with 500. To see how everything is going.

LearningRatio = 'You can perform a range test every time you want. This value will depend on the dataset and NN architecture.

Let's perform a Range Test. It's very important to study the response of the learning rate in your model.



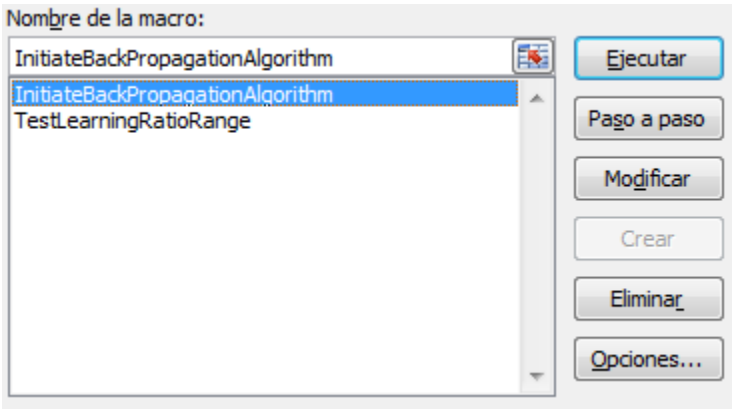
The result of the Test Range will be displayed in the immediate window. If you don't have it active press Ctrl + G.

With a Test Range you can see the relationship between the squared error and the learning ratio in only 10 iterations. Our best result was in 0.005. A common rule of thumb is to divide this value by 10.

0.0005 will be stable and effective.

```
0.00001  ---> 808.86277936321
0.0001  ---> 517.788551499356
0.001  ---> 26.8231055381561
0.005  ---> 11.8198010264647
0.01  ---> 71.838851210878
0.05  ---> 32.2938540911626
0.1  ---> 43.8146290407925
1  ---> 43.7966432205985
```

Let's set the learning rate and start training.



In the immediate window you will see the squared error evolution. If everything is Okie Dokie the number will decreasing.

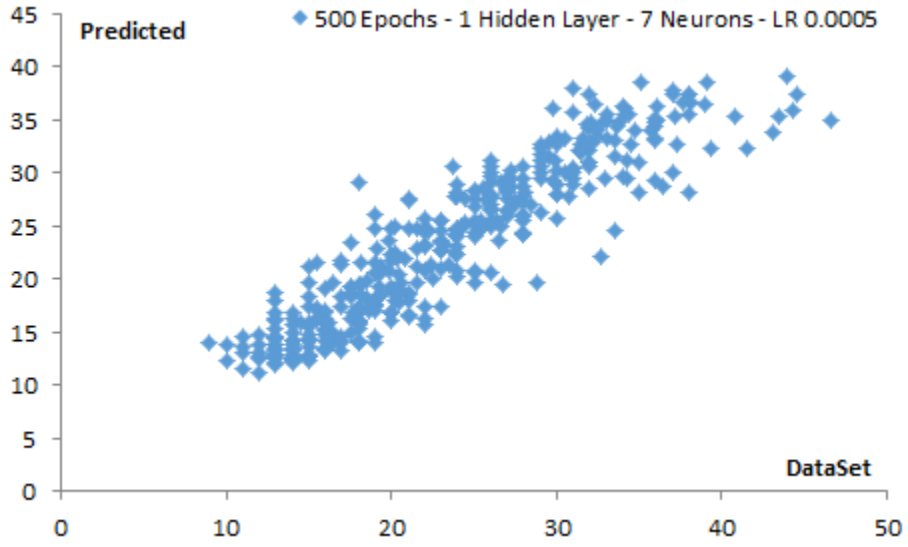
```
4615.82510273011
16.4718010326804
6.57069519893417
4.69655792384817
3.2669504247329
1.9343110071502
```

You heard 3 beeps? You see how the squared error has decreased? That mean that we are ready! Go to the "Result" Sheet and copy the content in the module "Result Module". Now a Formula named **myneuralnetwork** is ready to use in your spreadsheet!

I2 | =MyNeuralNetwork(A2:G2)

	A	B	C	D	E	F	G	H	I
1	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model year	Origin	Mpg	MPG
2	4	121	113	2234	12.5	70	2	26	20.6
3	4	98	83	2075	15.9	77	1	33.5	31.5
4	4	119	100	2615	14.8	81	3	32.9	29.4
5	8	318	150	3940	13.2	76	1	13	16.1
6	4	98	79	2255	17.7	76	1	26	28
7	6	225	105	3121	16.5	73	1	18	16.1

Let's compare the prediction with the given values:



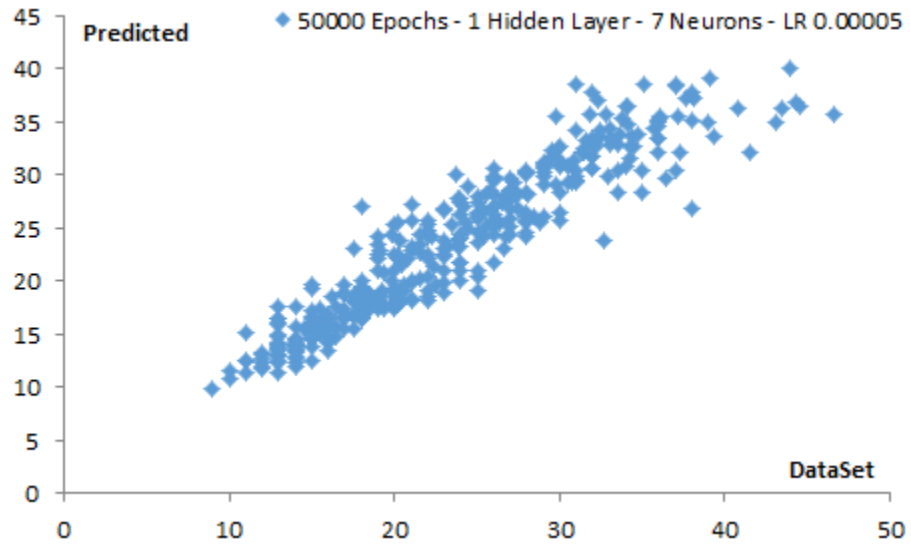
YOU WANT TO TRAIN MORE? NO PAIN NO GAIN?

Just change UploadOldModel to true and keep training.

A learning ratio reduction is commonly recommended. Or just perform another “RangeTest”!

I will go with 0.00005. Because I feel lucky!

Let’s see after 45500 iterations!



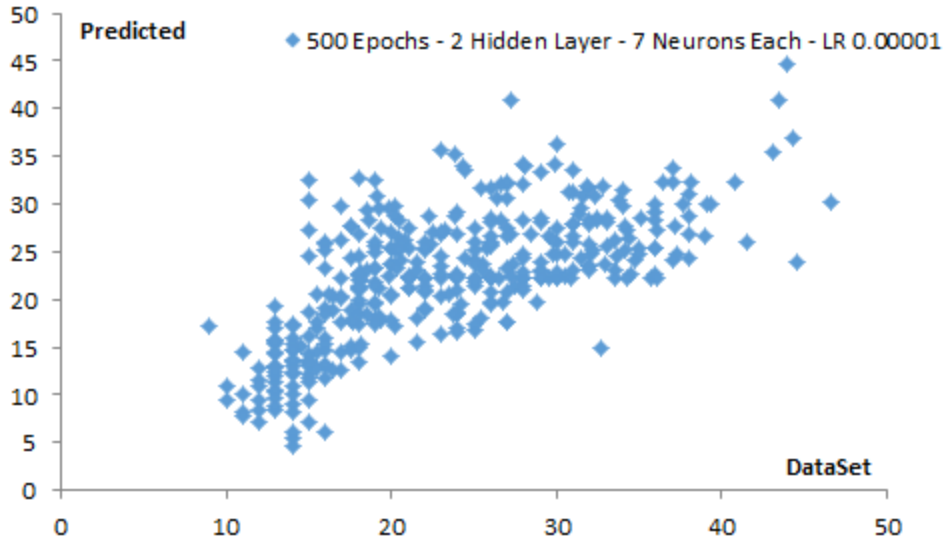
YOU WANT TO GO DEEPER? WE ARE NOT HERE TO SEE ONLY ONE HIDDEN LAYER!!!

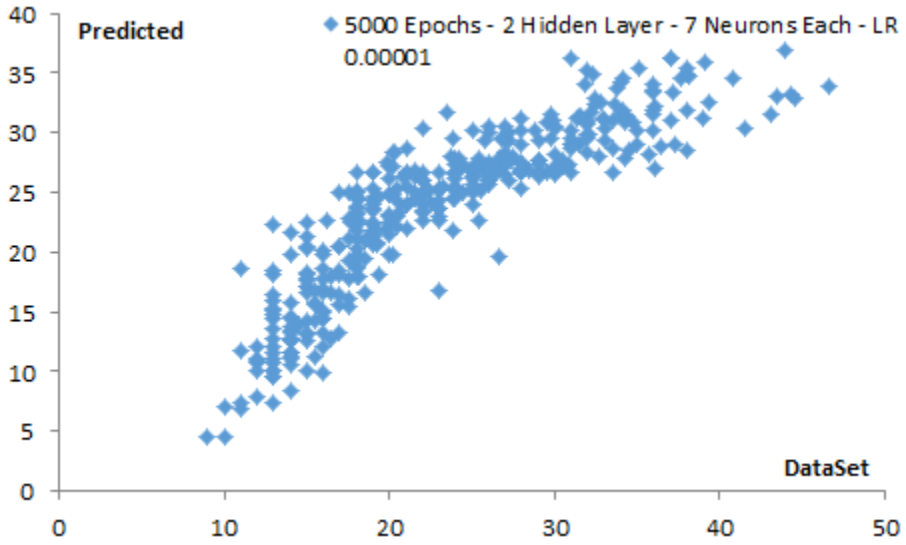
Make sure that “ChangeArquitecture” is true and change “n_hid_layer”. If you don’t want to explode your neural network adding layers progressively is recommended.

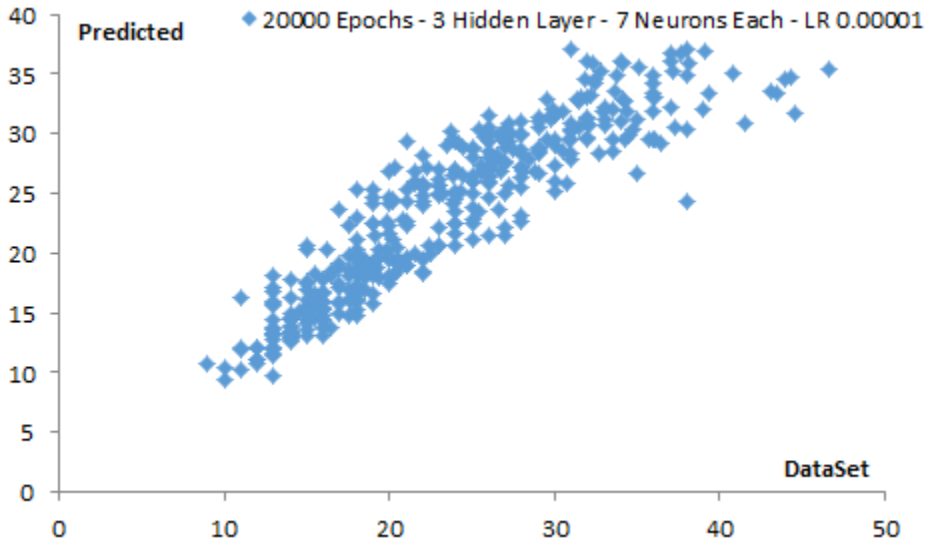
ATTENTION: REMEMBER TO ADD THE NUMBER OF HIDDEN LAYER OF EVERY LAYER!

```
n_hid_layer = 10
x_features = 7
y_features = 1
ReDim n_neurons_hid_layer(n_hid_layer + 1)
n_neurons_hid_layer(0) = x_features
n_neurons_hid_layer(1) = 5
n_neurons_hid_layer(2) = 5
n_neurons_hid_layer(3) = 5
n_neurons_hid_layer(4) = 5
n_neurons_hid_layer(5) = 5
n_neurons_hid_layer(6) = 5
n_neurons_hid_layer(7) = 5
n_neurons_hid_layer(8) = 5
n_neurons_hid_layer(9) = 5
n_neurons_hid_layer(10) = 5
n_neurons_hid_layer(n_hid_layer + 1) = y_features
```

Here are some predictions. They were taken on a rush. No optimum intended.


Predicted
500 Epochs - 2 Hidden Layer - 7 Neurons Each - LR 0.00001
DataSet
0
5
10
15
20
25
30
35
40
45
50
0
10
20
30
40
50


Predicted
5000 Epochs - 2 Hidden Layer - 7 Neurons Each - LR 0.00001
DataSet
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50


Predicted
20000 Epochs - 3 Hidden Layer - 7 Neurons Each - LR 0.00001
DataSet
0
5
10
15
20
25
30
35
40
0
10
20
30
40
50