Report HW1

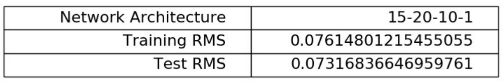
Student ID : 0845034

Regression

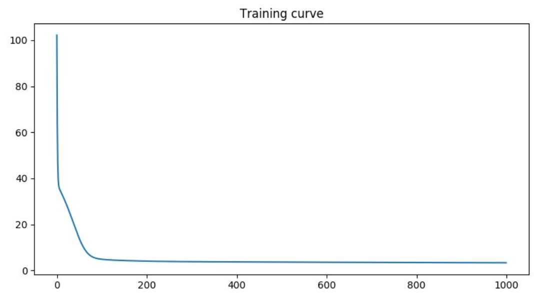
We were asked to implement a Neural Network. I used the following architecture :

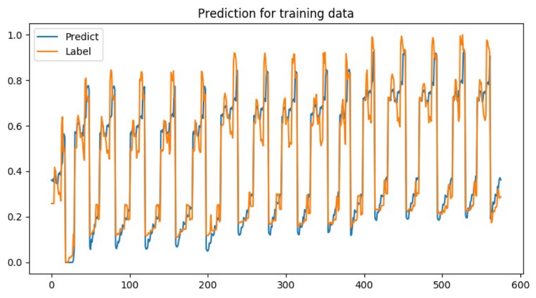
15-20-10-1 : We had 15 features because of the one-hot vectorization of 2 inital features that were categorical. We have one output corresponding to the heating we're looking for.

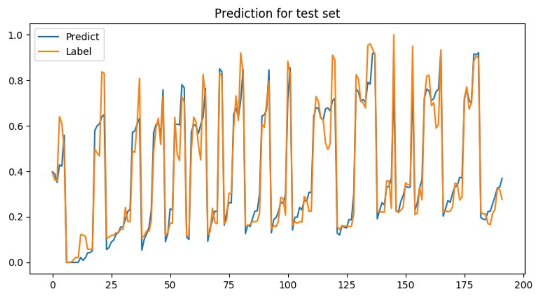
I've worked a lot on the implementation of the neural network. We must at first do a forward propagation form the input data, to have a first 'random' output. I used the ReLU activation function. Then by computing the cost and computing the gradient w.r.t different matrices of coefficient, we can update our coefficients. After several epochs, I was able to fit the train and test data. You can see on the following plots the results I had.



Network Architecture	15-20-10-1
Training RMS	0.07614801215455055
Test RMS	0.07316836646959761



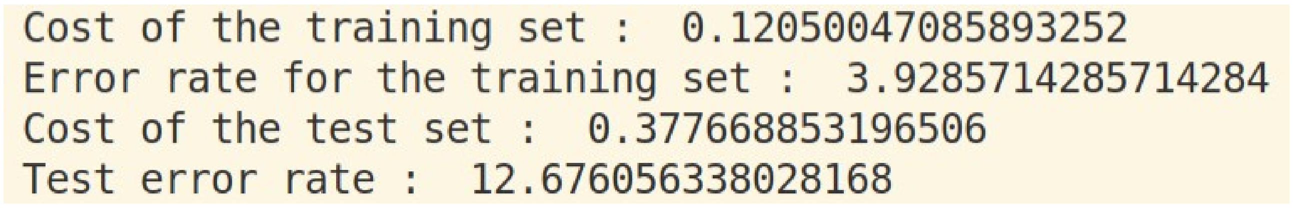




I have tried to select different features, but after the training I had no major changes. To select the features, I've been trying to delete the features that were increasing the most the cost. I defined two functions to select those features, but it was quite long to run it because it's a naive approach, I'm just testing every possibilities.

Classification

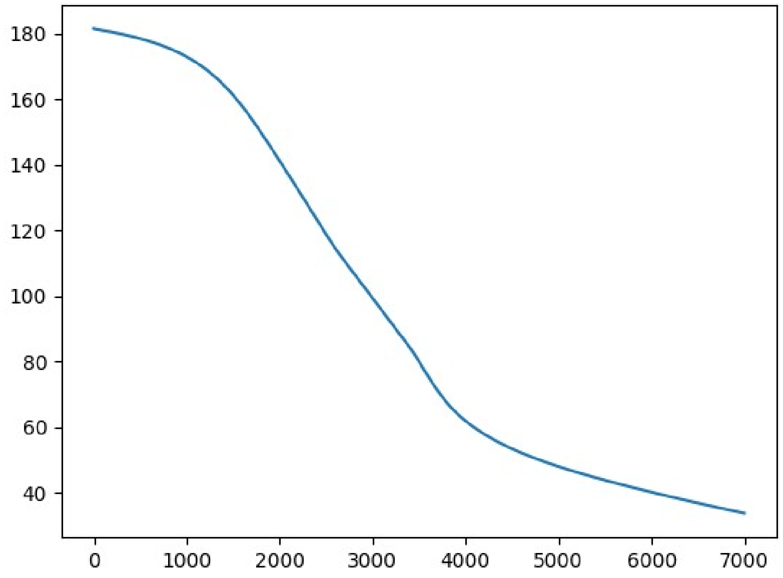
I started from my precedent program. This time, I used sigmoid function as activation function, because it's well designed for classification problem.
I had the following results :



```
Cost of the training set :  0.12050047085893252
Error rate for the training set :  3.9285714285714284
Cost of the test set :  0.377668853196506
Test error rate :  12.676056338028168
```

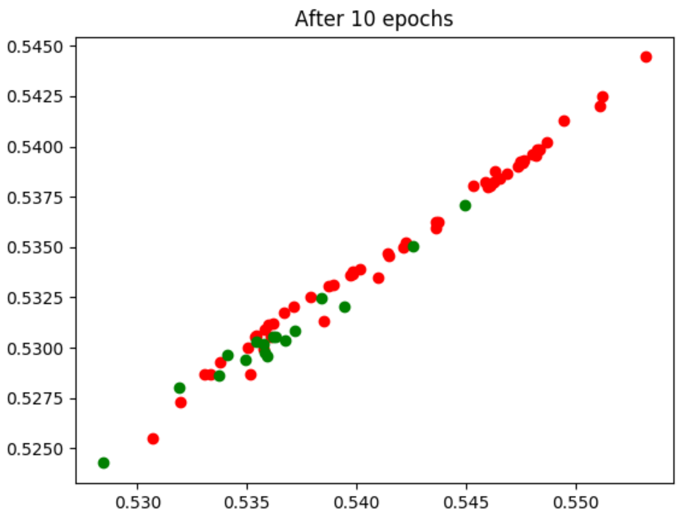
You may notice that the test error rate is higher than the error rate for training set. This is because I did not use regularization, therefore I tend to overfit the training data. I had difficulties to make the model work, because small changes in the weight initialisation, or in the learning were giving me really bad results. The costs are quite low !

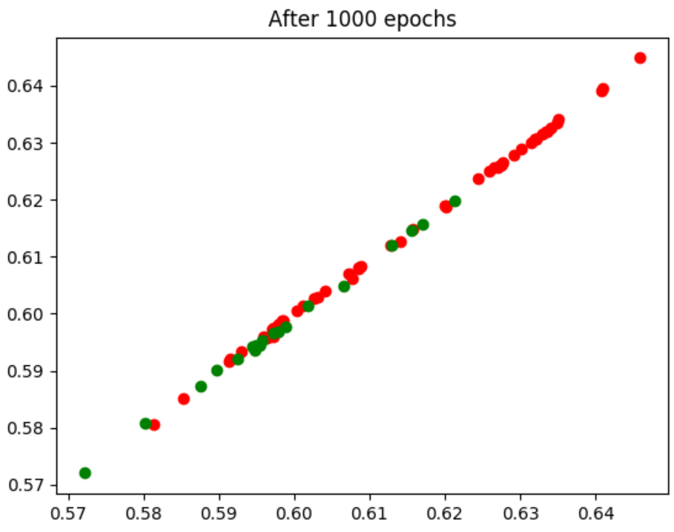
For this exercise, I used the following architecture :
34-6-2-1 : We had 34 features in this data set, and one output corresponding to the probability of having an exemple in the label 1. I had the following learning curve :



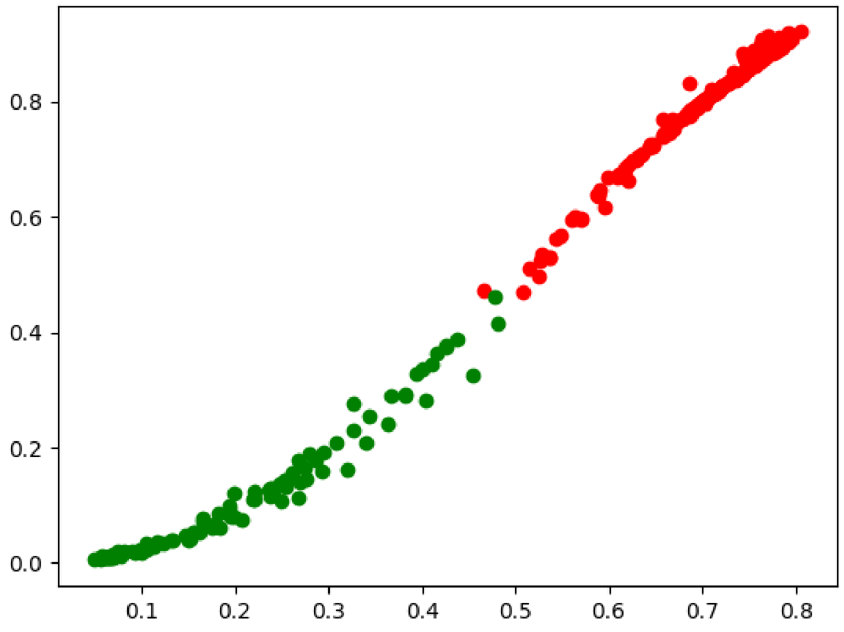
Training curve

The 2 neurons in the layer before the ouptut made me able to plot a 2D graph of the data (test data here) :





We can notice that after 10 epochs, the labels are still mixed. But as we go through more epochs, they are regrouping. In the end, they are clearly separated :



2D-feature representation

I haven’t computed the 3D representation, it would be the same way, with a 3 neuron layer instead of a 2 neuron layer before the ouptut.