Data Mining Assinment 2

Ahmet Oral

November 2020

.

DECLARATION OF HONOR CODE:
Student ID: 180709008
Name: Ahmet
Surname: Oral
In the course of Data Mining (CENG 3521), I take academic integrity very seriously and ask you to do as well. That's why, this page is dedicated to some clear statements that defines the policies of this assignment, and hence, will be in force. Before reading this assignment booklet, please first read the following rules to avoid any possible violation on academic integrity.

- This assignment must be done individually unless stated otherwise.
- You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, you cannot copy code (in whole or in part) of someone else, cannot share your code (in whole or in part) with someone else either.
- The previous rule also holds for the material found on the web as everything on the web has been written by someone else.
- You must not look at solution sets or program code from other years.
- You cannot share or leave your code (in whole or in part) in publicly accessible areas.
- You have to be prepared to explain the idea behind the solution of this assignment you submit.
- Finally, you must make a copy of your solution of this assignment and keep it until the end of this semester.

I have carefully read every of the statements regarding this assignment and also the related part of the official disciplinary regulations of Mugla Sıtkı Kocman University and the Councilof Higher Education. By signing this document, I hereby declare that I shall abide by the rulesof this assignment to prevent any violation on academic integrity.

Signature

2 Single-layer Perceptron

2.1 Classification task

.

In this task my goal is to applying single-layer perceptron to use it on classification task and to analyse its performance with varying conditions, such as different tuple, dimension size and different max iterations for perception algorithm.I started by generating a fictional dataset with 'make_classification',Then I split data as %70 of the tuples are used for training while %30 of the tuples are used for testing.After that I applied single layer Perceptron with 100 iterations to 4 different tuple and dimension sizes.Here are the results:

Table:The effectiveness and efficacy of single-layer perceptron with respect to the varying parameter settings.
Perceptron runned with 100 iterations.
random_state = 42

	Tuple Size (m)	**Dimension Size (n)**	**Training Time (in ms)**	**Error**
a.	10,000	100	0.014002 s	0.131
b.	10,000	1,000	1.124585 s	0.16133
c.	100,000	100	0.245309 s	0.20666
d.	250,000	100	0.397727 s	0.0836

random_state = 442

	Tuple Size (m)	**Dimension Size (n)**	**Training Time (in ms)**	**Error**
a.	10,000	100	0.013003 s	0.131
b.	10,000	1,000	1.174455 s	0.12766
c.	100,000	100	0.270203 s	0.1782
d.	250,000	100	0.61883 s	0.1440266

I created 2 tables with the same parameters but I changed the random state parameter because when I test it with different random states I get different results.For a,b error increases, but after I change the random state error is decreasing a little bit.Same thing happens with c,d , when I change the tuple with random state 42 error decreases but with random state 442 error is decreasing.I test with lots of different random states but errors seems to fluctuate.I don't completely understand why but I think if I increase tuple and dimension size much bigger I would get more accurate,stable error values.Because error changes for different random states and there is no clear pattern on it I can't make accurate comments about this.Training time is of course not much affected by random state, it increases as the dimension or tuple size grows as I expected.

Here are the results of for 500 iteration:

Tuple Size (m)	**Dimension Size (n)**	**Training Time (in ms)**	**Error**
e. 10,000	100	0.023242 s	0.131
f. 10,000	1,000	0.123591 s	0.16133
g. 100,000	100	0.277184 s	0.20666
h. 250,000	100	0.397727 s	0.0836

When I changed the iteration from 100 to 500, error is not affected.For curiosity I lowered the iteration to 5 and then I compared it with 100 iteration, after that error was effected.As I lowered iteration to 5 error increased a little bit compared to the 100 iterations.Anyways, for the iteration values you asked us to test, only difference is the training time average.For smaller dimension sizes, training time increases, but after we increase dimension size, time gets faster than the one with 100 iterations.For the tuple size it is doesn't change too much.I run it more than 10 times to get an average training time but there is no significant difference I can say.Sometimes 500 iterations is faster and sometimes 100 iterations is faster.I can't tell.In dimension size, change is very clear but in tuple size it fluctuates a lot.

2.2 Visualization of decision boundary

.

For this part, I created a toy dataset with 600 sample and 9 features because you said data should have at least 500 tuples and three features.You also said two features should be informative to the ground truth vector.So I created the data with "n_informative=2" to match this requirement.Here you asked us to use single layer perceptron, but I used MLPClassifier with no hidden_layer_sizes instead of perceptron because when I used perceptron programs crashes.I fitted X_test, y_test values because you asked us to plot test values.For showing testing objects and hypothesis plane on 3-D surface I created a meshgrid and used scatter.This part really was hard for me, I tried many different approaches until I found the solution.I took a lots of online help and to be honest I don't completely understand all of the figure part.Nevertheless it is working as you asked.

Here are the results:

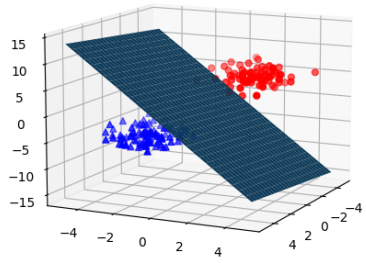

Figure 1: Testing objects and hypothesis plane on 3-D

3 Multi-layer Perceptron

3.1 Error convergence with multi-layer perceptron

.

This parts task was to apply multi-layer perceptron to obtain error values throughout the training.I loaded the digits dataset and split it using train test split with 0.3 test_size ratio.I applied MLPClassifier with 1 hidden layer size with 50 neurons and 100 iterations.I used plt.plot(clf.loss_curve_) to create figure.

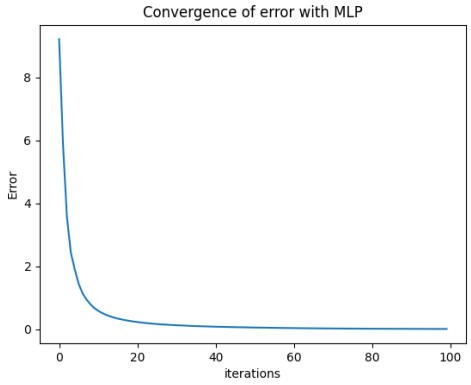

Figure 2: Convergence plot for error values.

The figure above shows error values as a function of iteration.We can see the error significantly drops after around the 10 iterations.After around 40 iterations error stabilizes.We can see how important the iteration effects the error very clearly from the drop of error.

3.2 Effects of multi-layet perceptron structure on train test scores

.

In this part, my task was to analyse train and test scores (accuracy) as a function of hidden layer size and neurons and plot score (accuracy) values.As usual I loaded the digits dataset and split it with 0.3 test ratio.The fun part here was to apply multi-layer perceptron network with one to 'H' hidden layer size.H determines the hidden layer size and how many neurons we are going to put in those hidden layers.User input determines the H so I was going to have to write a function to calculate hidden layers and neurons according to H.

I created a function named h_values to return the list I am going to put in the hidden_layer_sizes parameter.

```
def h_values(h):
    list=[]
    for i in range(h):
        i=i+1
        list.append(2**i)
    list.reverse()
    return list
```

This is the function.(I put a lot's of comments in my code to explain everything so more details can be found in my .py file).For example if user enter H value as 5, it returns [32,16,8,4,2].I used this list in the hidden_layer_sizes parameter to create hidden layers as you asked us.

Than I created a for loop that runs H times to take the test and train scores for all hidden layer sizes.Here is the output:

```
Please enter the 'H' value: 10
Hidden layers:  [2] Train scores:  0.2211614956245028 --Test scores:  0.2351851851851852
Hidden layers:  [4, 2] Train scores:  0.5839299920445505 --Test scores:  0.562962962962963
Hidden layers:  [8, 4, 2] Train scores:  0.10421638822593476 --Test scores:  0.07962962962962963
Hidden layers:  [16, 8, 4, 2] Train scores:  0.5632458233890215 --Test scores:  0.525925925925925
Hidden layers:  [32, 16, 8, 4, 2] Train scores:  0.994431185361973 --Test scores:  0.8851851851851852
Hidden layers:  [64, 32, 16, 8, 4, 2] Train scores:  0.8965791567223548 --Test scores:  0.8018518518518518
Hidden layers:  [128, 64, 32, 16, 8, 4, 2] Train scores:  0.8918058870326173 --Test scores:  0.8444444444444444
Hidden layers:  [256, 128, 64, 32, 16, 8, 4, 2] Train scores:  0.42482100238663484 --Test scores:  0.4203703703703704
Hidden layers:  [512, 256, 128, 64, 32, 16, 8, 4, 2] Train scores:  0.10342084327764518 --Test scores:  0.08703703703703704
Hidden layers:  [1024, 512, 256, 128, 64, 32, 16, 8, 4, 2] Train scores:  0.6945107398568019 --Test scores:  0.6814814814814815
```

You can see the all hidden layers with their train and test scores.

After that, I created the plot with score values as a function of hidden layer size as you asked.Here is the plot:

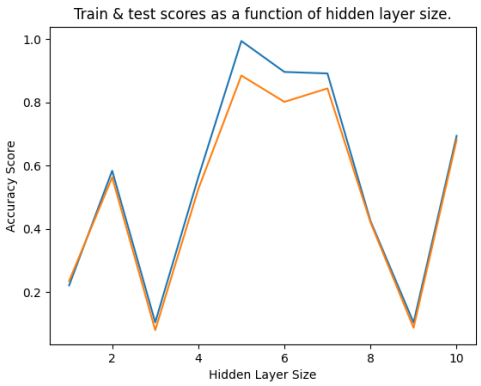

Figure 3: Convergence plot for error values.

Conclusion

In this assignment I tried to approach problems in different ways and I always tried to improve my code.I used my own knowledge and when it is not enough to solve a problem I learned new methods to solve it.In the first task because I had different error values for different random_states.There was no visible pattern so I couldn't make a comment about it, but other than that I tried to explain and show what I understood as much as I could.Apart from this report I always put as much comment as possible in my code.I explained how my code works in the comments.I tried my best while coding but there might be big problems that I am not aware of.There might be things I implemented wrongly or any other problem that I am not aware of.So if there is,please tell me where I did wrong or what I could improve.(I tried to keep my report simple to not exceed 5 pages but still it has 6 pages because of the images.)
Thanks for reading.