

**İstanbul Aydın University**

Mehmet Eray Sürmeli,Software Engineering Department

B1605.090038,Number

SEN 437 Machine Learning

**Homework 3 Report**

**Questions**

**2-a)**

(10 points) Add the required Python lines to the existing code to create a Confusion Matrix for the Logistic Regression result from the scikit-learn library. First produce 500 blue samples and 500 red samples and later find the classification performance from these test dataset. Produce nn-from scratch-confusion.ipynb file.

Confusion matrix is a very popular measure used while solving classification problems. It can be applied to binary classification as well as for multiclass classification problems.

Confusin matrix explains that how much did correctly.

True values are bigger than others.So this is good.

**2-b)**

(5 points) Explain the Confusion Matrix output in your modified Jupyter notebook and also in your report. Briefly describe 4 numbers in the matrix and also accuracy, precision and recall values.

Code:

from sklearn.metrics import confusion\_matrix

confusion\_matrix(y, y\_pred)

Output:

array([[436, 64],

[ 70, 430]])

---------------------------------------------------------------------------------------------------------------------------

Code:

tn, fp, fn, tp = confusion\_matrix(y, y\_pred).ravel()

tn, fp, fn, tp

Output:

(436, 64, 70, 430)

True Positive = 430 True positive: correctly classified or detected.

False Negative = 70 False Positive: incorrectly classified or detected. It represents the type I error.

False Positive = 64 False negative: incorrectly rejected. It represents the type II error

True Negative = 436 True negative: correctly rejected.

* Accuracy score: Out of all the classes, how much we predicted correctly, which will be, in this case 0.82. It should be high as possible.

Code:

from sklearn.metrics import accuracy\_score

accuracy\_score(y, y\_pred)

Output:

0.866

* Precision: Out of all the positive classes we have predicted correctly, how many are actually positive. Values are good

Code:

from sklearn.metrics import precision\_recall\_curve

precision, recall, thresholds = precision\_recall\_curve(y, y\_pred)

precision

Output:

array([0.5 , 0.87044534, 1. ])

* Recall is out of all the positive classes, how much we predicted correctly. It should be high as possible. 0.83615819 is not bad.

Code:

recall

Output:

array([1. , 0.86, 0. ])

**2-c)**

(5 points) Put the output plot (decision boundary learned) obtained by Logistic Regression into your report and interpret the output. Is the classification is good enough to separate red and blue classes? Why, explain your answer.

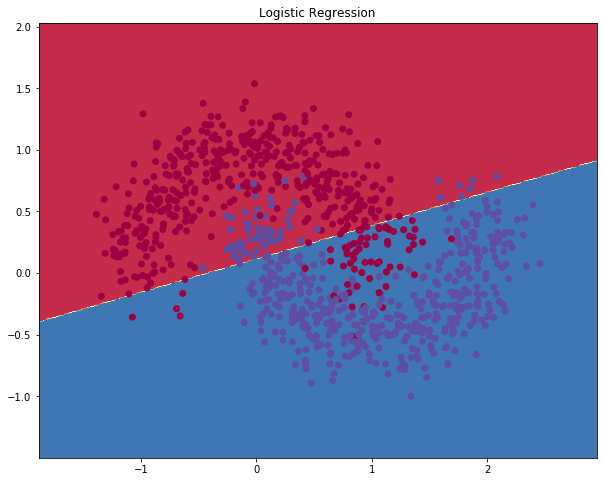


Figure 1.0

As you can see in above model separated by a single line(Figure 1.0). I think this is not enough.Because separator couldnt separate completely or near completely. There are still many blue members above the line and of course there are red members under it. But pay attention to underfit. This model is not ready right now.

**2-d)**

(5 puan) Explain the following methods build\_model, calculate\_loss ve predict and also important variables of the program (such as gradient descent parameters, epsilon, reg\_lamda.)

In build model method,while we are building model,we are calculating required mathmatical processes.

In calculate\_loss while we are building our model,we are changing loss data with calculated true one. We are calculating true one as some mathmatical processes. We return the highest probability.

In the predict, we make a prediction process as it is cleared of bad data and arranged.here we train our Neural Network. It implements batch gradient descent using the backpropagation derivates we found above.

num\_examples => training set size ,

nn\_input\_dim => input layer dimensionality ,

nn\_output\_dim => output layer dimensionality and Gradient descent parameters ,

epsilon => learning rate for gradient descent ,

reg\_lambda => regularization strength

**2-e)**

(5 puan) Explain this line: hidden\_layer\_dimensions = [1, 2, 3, 4, 5, 20, 50].

Hidden layer dimensions are a term we use to make the model fit better.

Hidden layer daha cok anlat

Hidden layer number =1: not well separated - underfit

Hidden layer number =2: not well separated - underfit

Hidden layer number =3: not well separated - underfit

Hidden layer number =4: well separated - fit

Hidden layer number =5: well separated - fit

Hidden layer number =20: well separated - fit

Hidden layer number =50: overfit

**2-f)**

(5 puan) How is regularization performed in the program?

Regularization increases loss when misguided. The main purpose of the model is to minimize loss and optimize the data better. I mean regularization effects well.

**2-g)**

(5 puan) What happens as the number of hidden layers is increased in the program output? Is this good or bad for binary classification? What is its relation to overfitting? Explain these questions by putting printouts in your report.

As the number of hidden layers increases, the number of weights and bias that need to be updated also increases. Therefore, these values will be affected by ragularization and the effect of regularization will increase with the increase of the number of hidden layers. It will increase the performance of the data until a certain number of hidden layers. Good for this model. But after a point, it will cause overfit. Performance will also negatively affect.

If you see below(Figure 1.1),You will see.

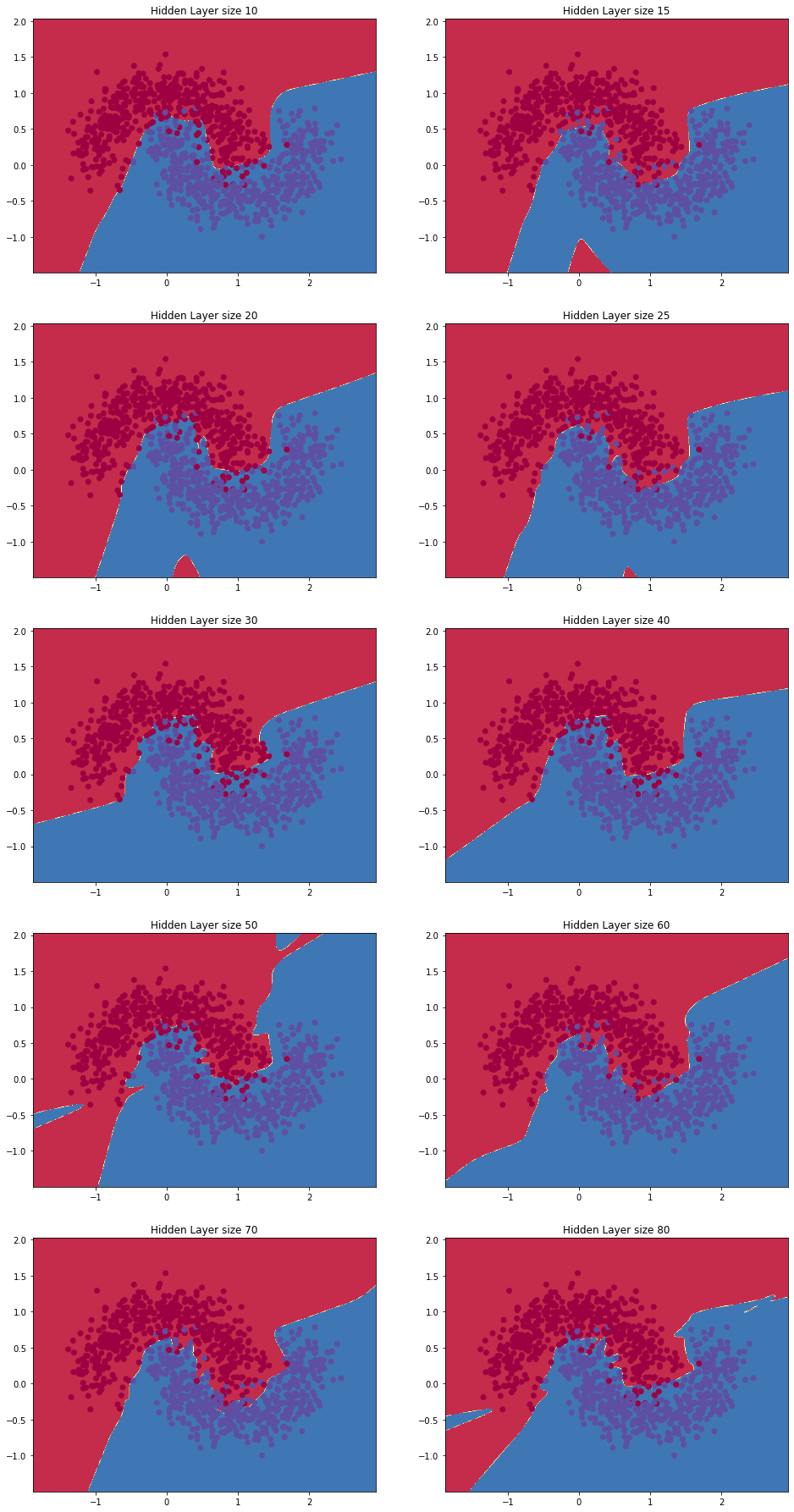
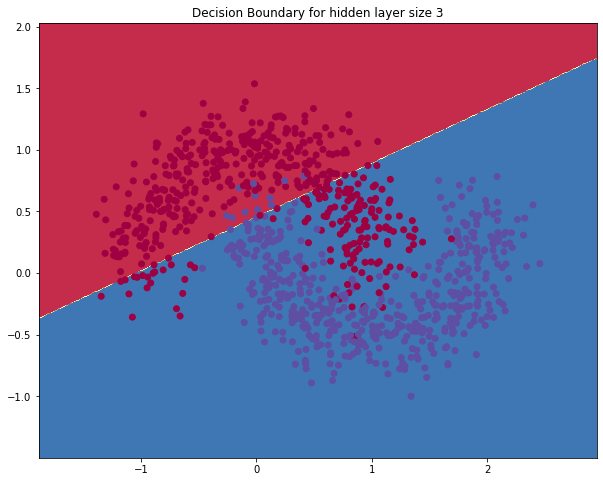


Figure 1.1

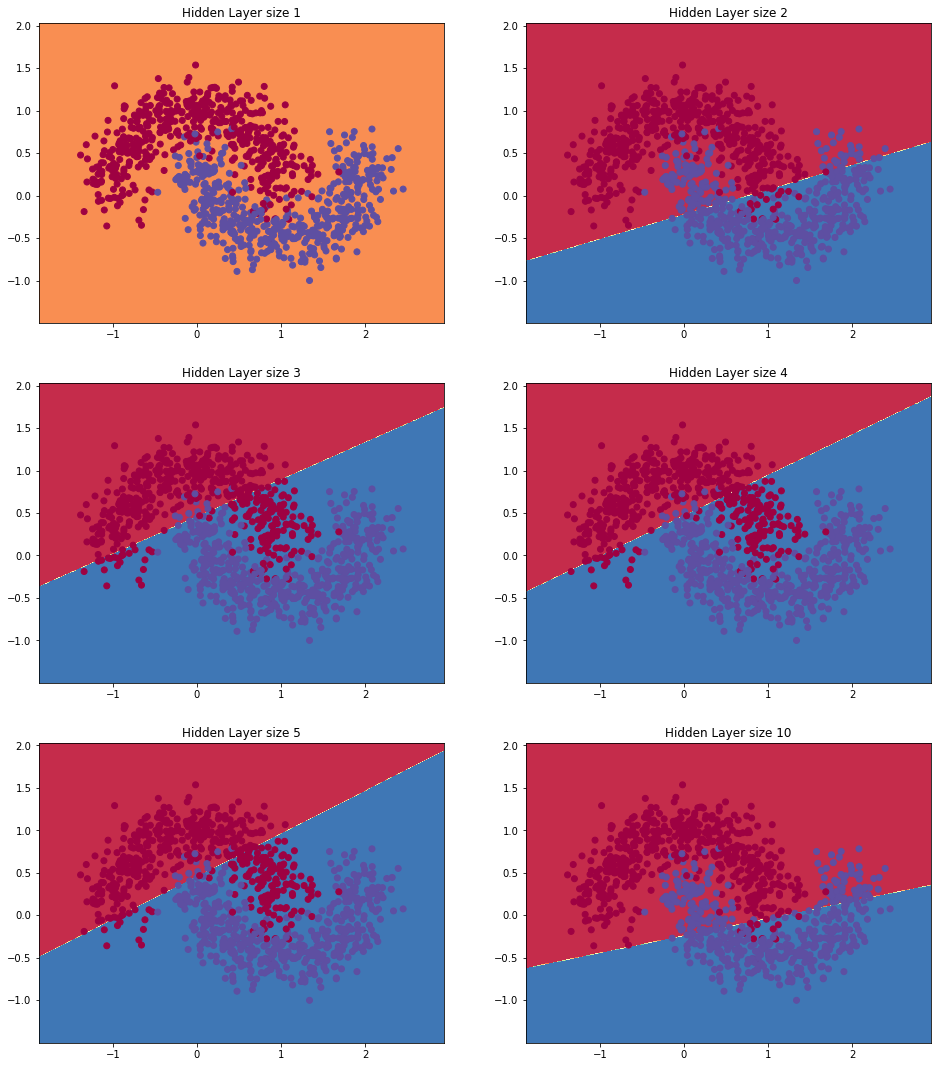
Here you can't see an overfit plot that repeats itself until the 30th layer. But after 30 it starts to repeat itself, that is to memorize it. We call this event overfit. It is something we do not want. And for this model, the number of hidden layers should be maximum 30.

**2-i)**Sigmoid

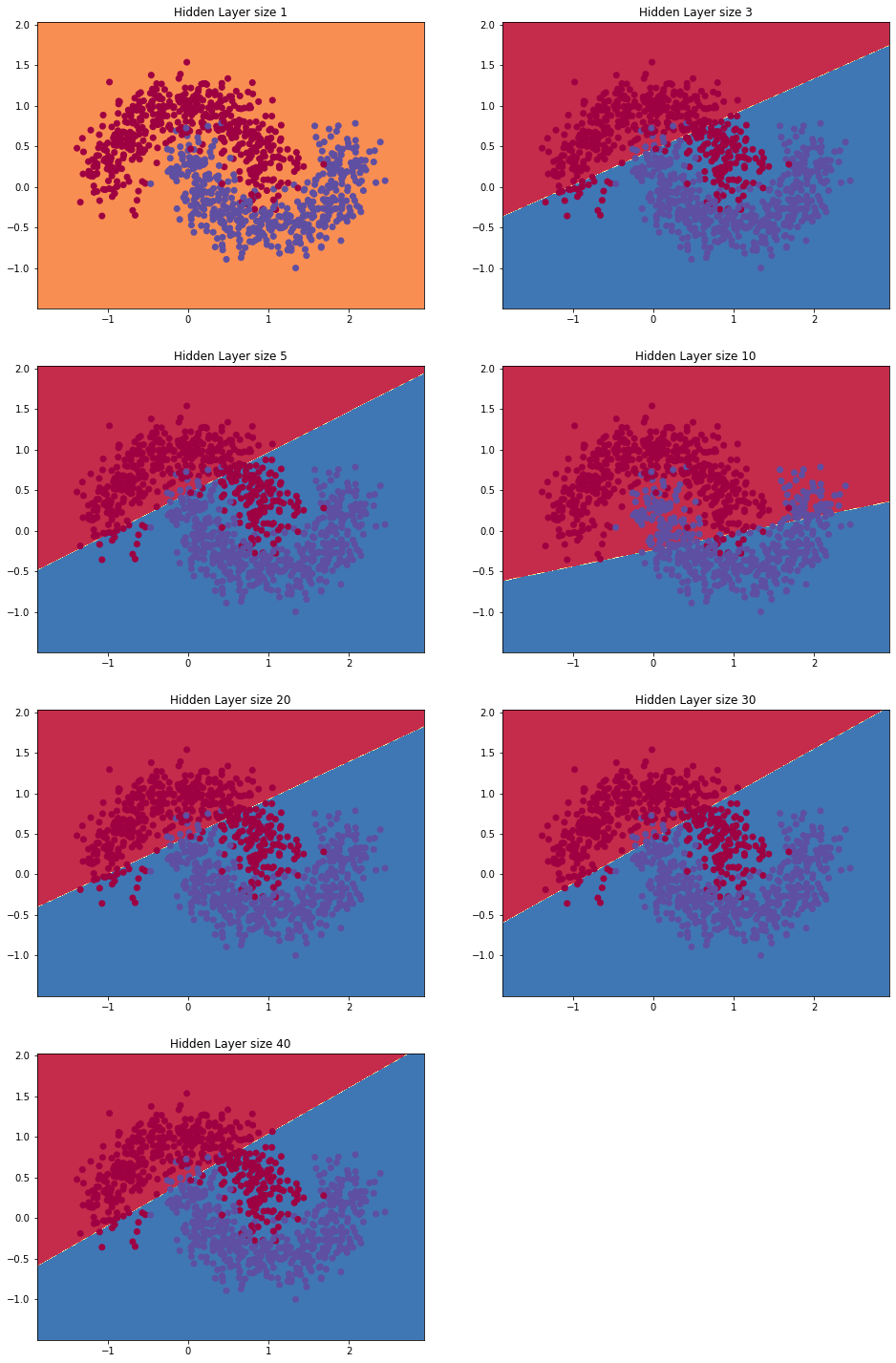
Hidden layer size = 3



Hidden layer size = [1, 2, 3, 4, 5,10]



Hidden layer size = [1, 3, 5, 10, 20, 30, 40]



**2-j)**

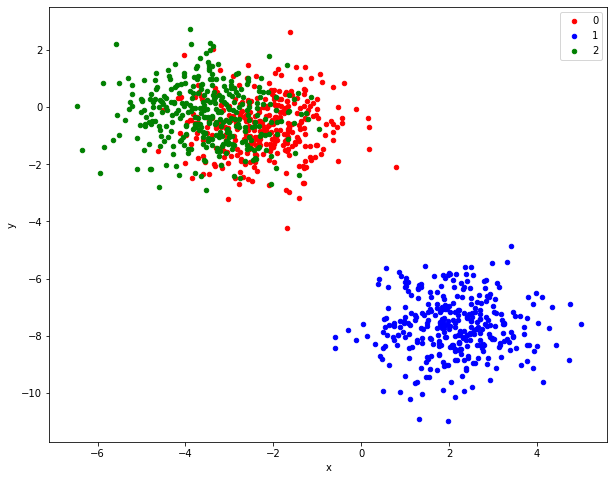


Figure 1.2

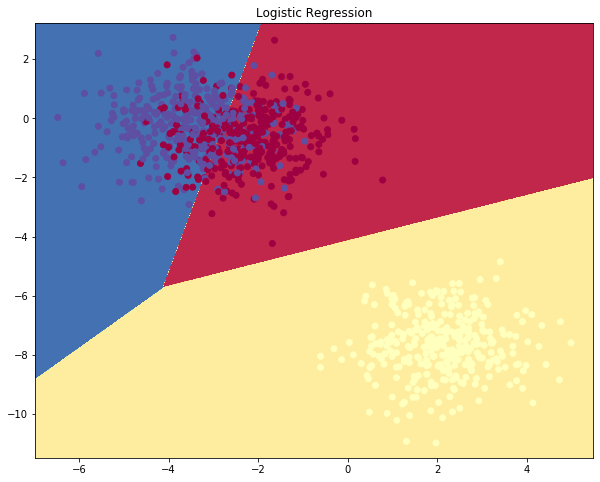


Figure 1.3

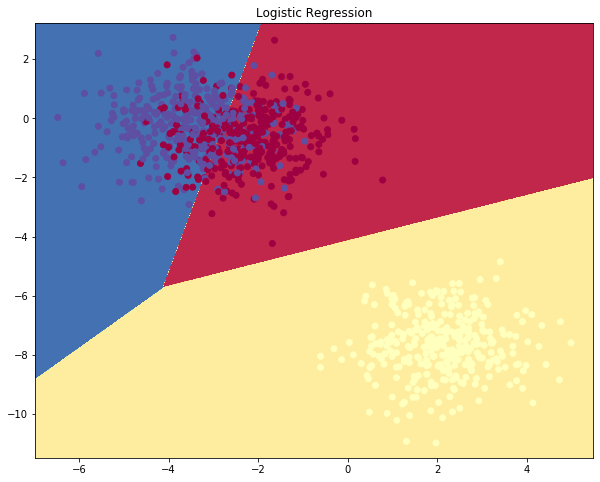


Figure 1.4 - Prediction of logistic Regression

**2-k)**

(optional for extra credit) Extend the network to 4 layers. Experiment with the layer size. Adding

another hidden layer means you will need to adjust both the forward propagation as well as the

backpropagation code.

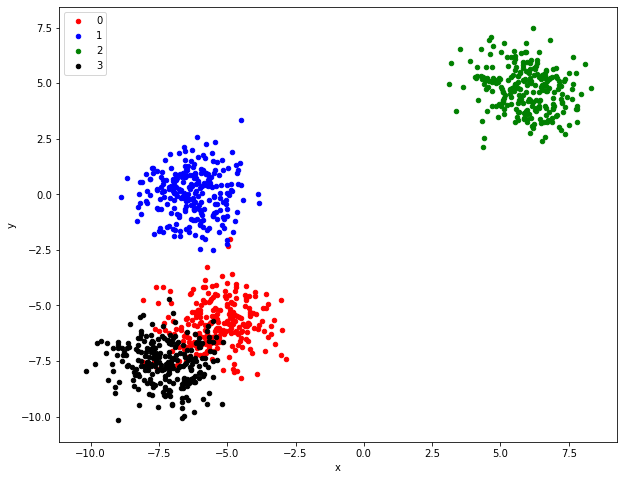


Figure 1.5

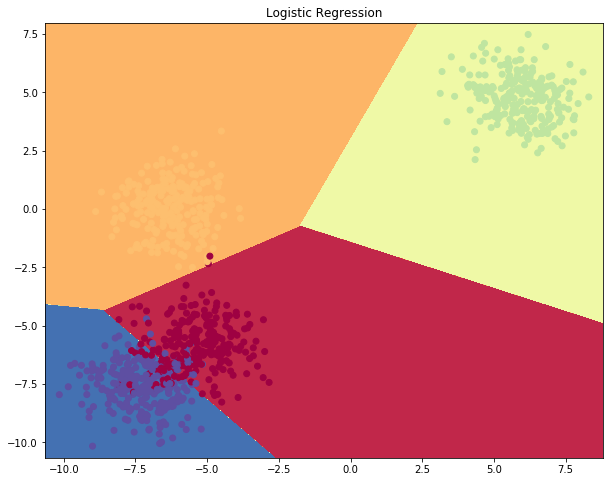


Figure 1.6