# Neural Networks and Deep Learning

## Week-3

In Week 3, we continue from where we left in week 2 by defining logistic regression as a neural network with ‘0’ hidden units to shallow neural network with ‘1’ hidden layer.

We understood, the intuition behind the back-propagation algorithm, the forward propagation computes the final output and computes the cost(loss) and in the back propagation we try to optimize the parameters, in such a way to reduce the cost very similar to how we try to do optimize something to reduce error and make it more efficient, we try to correct it next time when we perform.

We learnt a very simple technique of how Matrix Dimensions helps us to check if the flow of our algorithm is right, which is extremely helpful while debugging.

We now started using new activation functions Tanh, Relu and Leaky Relu function in addition to sigmoid function because sigmoid function is almost linear between in certain range and if we use sigmoid function as the activation function throughout the neural network, we will simply compute an output, which is some linear combination of your input, but if we use sigmoid function in the final output layer, as it performs well as classifier.

We have seen the importance of random-initialisation instead of initialising them to zeros because if you initialise all of them to zeros all your hidden units of a given layer would compute the same value, which is redundant.

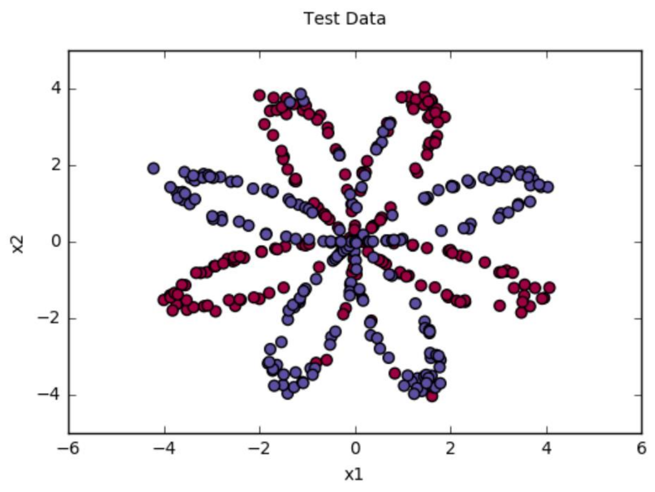
In the Programming Assignment, we generate red and blue points to form a flower.
We will then fit a neural network to correctly classify the points.

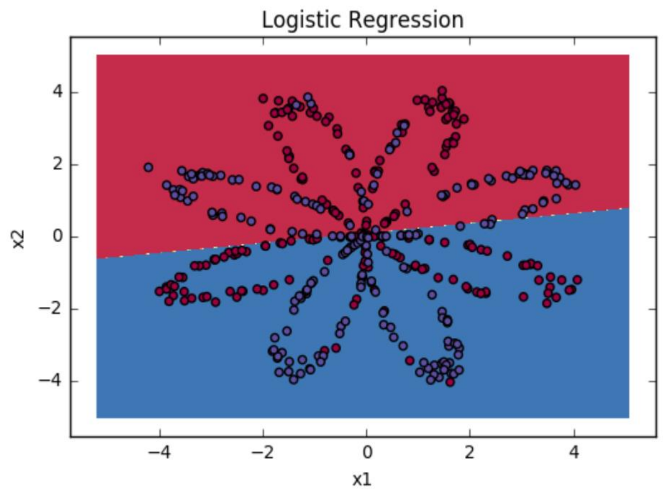
To understand the importance of neural network even more, we implemented the same problem with logistic regression and as we can see the image on the right, we get a linear boundary line where the dataset is not linearly seperable.

The decision boundary of a logistic regression is linear as long as the input features are linear.

$Z = w^T X + b,\ Y = \sigma(Z)$ -> Linear Decision Boundary as Input is Linear

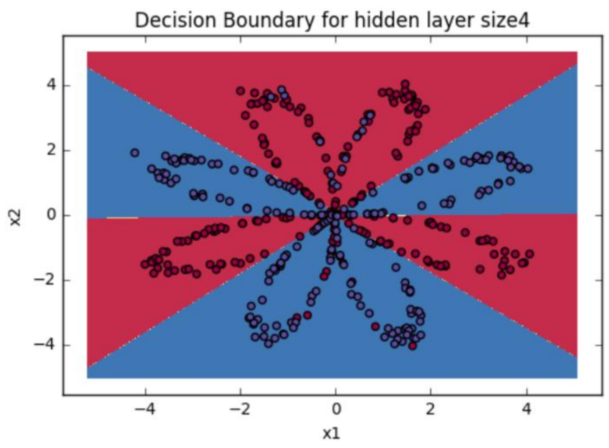
$Say\ X = f^2, Z = w^T X + b,\ Y = \sigma(Z)$ -> Non- Linear Decision Boundary as Input is Non-linear





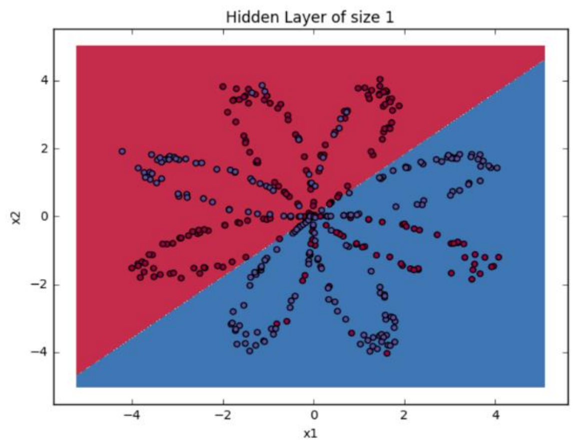
We built a Neural Network with
1 Input layer (2 features x1, x2)
1 hidden layer (4 units)
1 Output Layer.

Result:

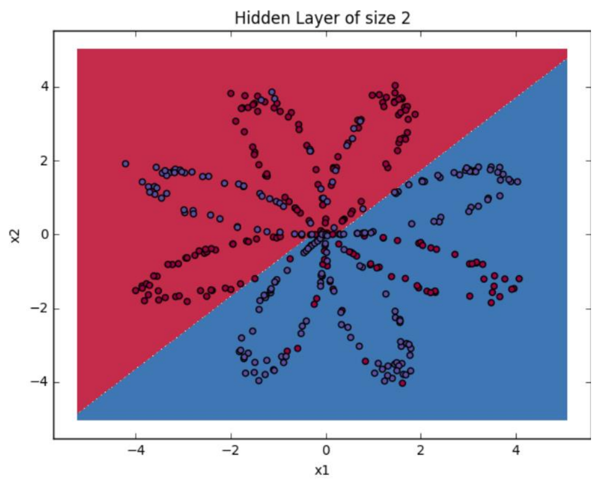


Accuracy for 4 hidden units: 90.5%

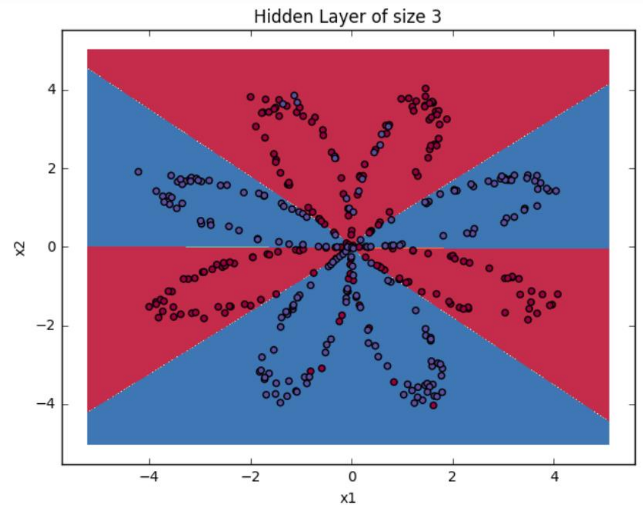
Hidden Layers of Different Values:



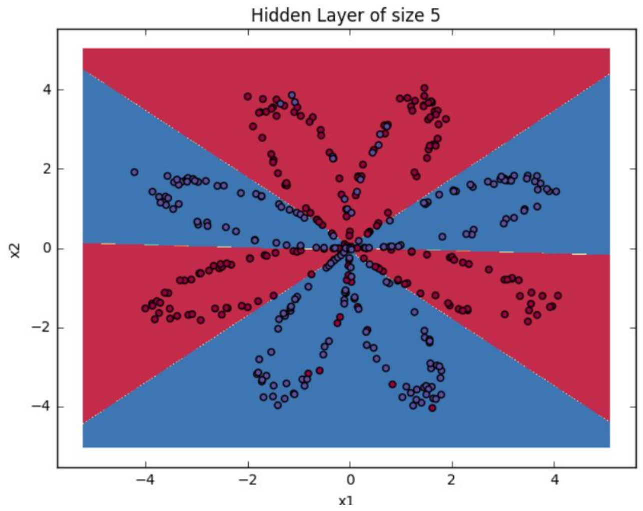
Accuracy for 1 hidden unit: 67.5%



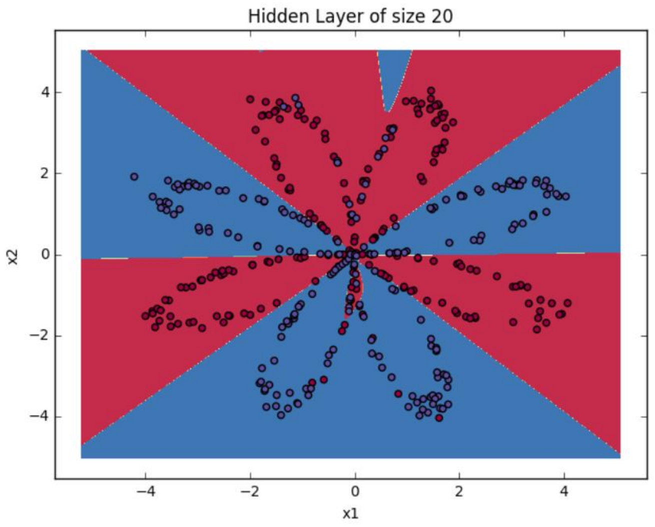
Accuracy for 2 hidden units: 67.25%



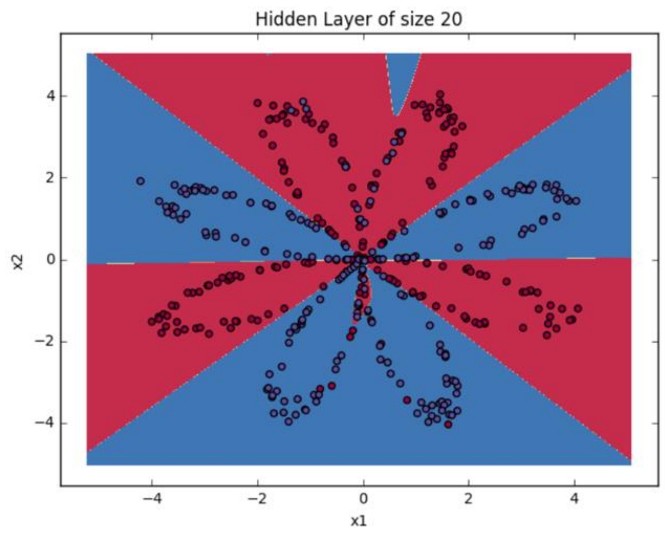
Accuracy for 3 hidden units: 90.75%



Accuracy for 5 hidden units: 91.25%



Accuracy for 20 hidden units: 90.0%



Accuracy for 50 hidden units: 90.25%

Conclusion:

- The larger models (with more hidden units) are able to fit the training set better, until eventually the largest models overfit the data.
- The best hidden layer size seems to be around n_h = 5. Indeed, a value around here seems to fit the data well without also incurring noticeable overfitting.