# Neural Network

CS 440 Programming Assignment 2
Wei Wei
Team Members: Yehui Huang
Feb 22 2016

## Problem Definition

Developing a Neural Network to analyze both the linear and nonlinear training data from ToyLinear and ToyMoon by using the backpropagation algorithm and predict the result by using the forwordpropagation and also used to minimal the errors. Examining accuracy of neural networks with different hidden layer and learning rate by testing the network with different hidden layers and learning rate. Upon examing the effects of the number of layers and learning rate, we discovered the overfitting problem and found different solutions for this problem.

## Method and Implementation

We implemented the Neural Network in the python with the usefull package Numpy, which is used as a powerful linear algbra caculator. Since there were something wrong with the Cannopy, we decided to use the Sublime instead, which was also helpful.

In the sketlton code, we are provided with a prefict() function which gives us a result by using the forword propagation. In addition, a funciton called compute_cost() to compute the error based on the preditct result. Our major implementation is to implement the predict functions which is the main training fucntion. The function basically involves three steps: Forward Propagation, Back Propagation and update model parameters using gradients. The Forward Propagation is to take the weights and biases to calculate the desired output of the function. The Backward Propagation takes the desired output and the actual output of the function to calculate the error between them. In the end, we calculate the gradients and update the model parameters.

## Experiments

We first trained our models using ToyLinear and ToyMoon which represents linear and nonlinear data set. We set our epsilon to be 0.01 and number of epochs to be 5000 and we used both a 2 layer neural network and another with a hidden layer. Then we performed our test based on various learning models to obtain an observation of the influence of the learning rate. Then we also tested the performance of the neural work with different hidden layers and find the overfitting problem. In order to address the overfitting problem, we use the L2 regularization to avoid overfitting.

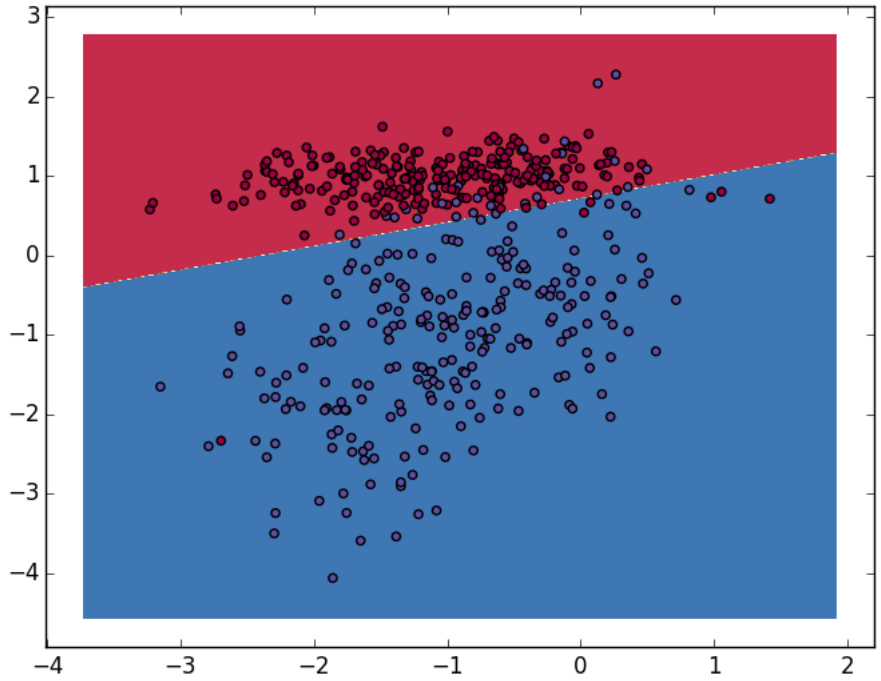
In order to define the accuracy of our neural network, we used the plotdecisionboundary function in sktltion code to viualize the accuracy of the neural network. For the learning rate, we used three learning rates as our test sample: 0.01,0.025 and 0.07, we then used the compute_cost function to calculate the error and plot them on a two dimentional axis in order to visualize the cost. For different hidden layers and the L2 regularizartion, we also used the plotdecisionboundary function to visualize our output.

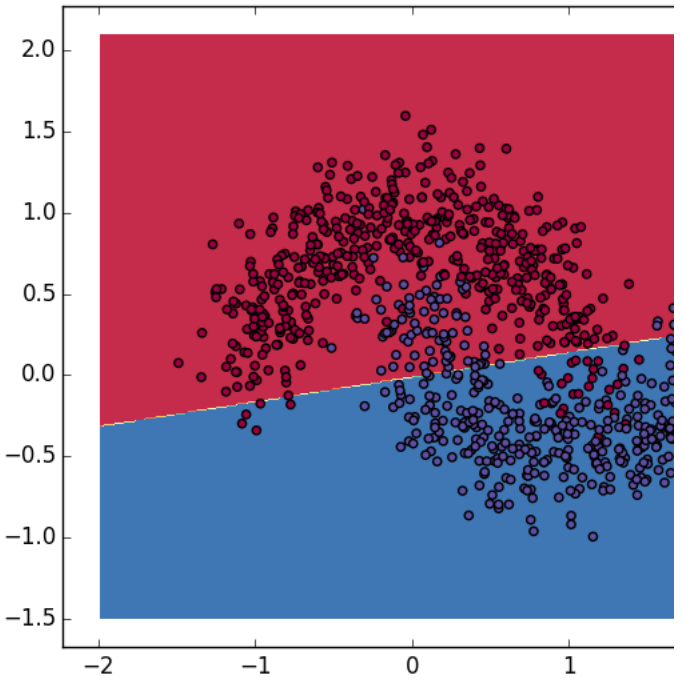
## Results

List your experimental results. Provide examples of input images and output images. If relevant, you may provide images showing any intermediate steps

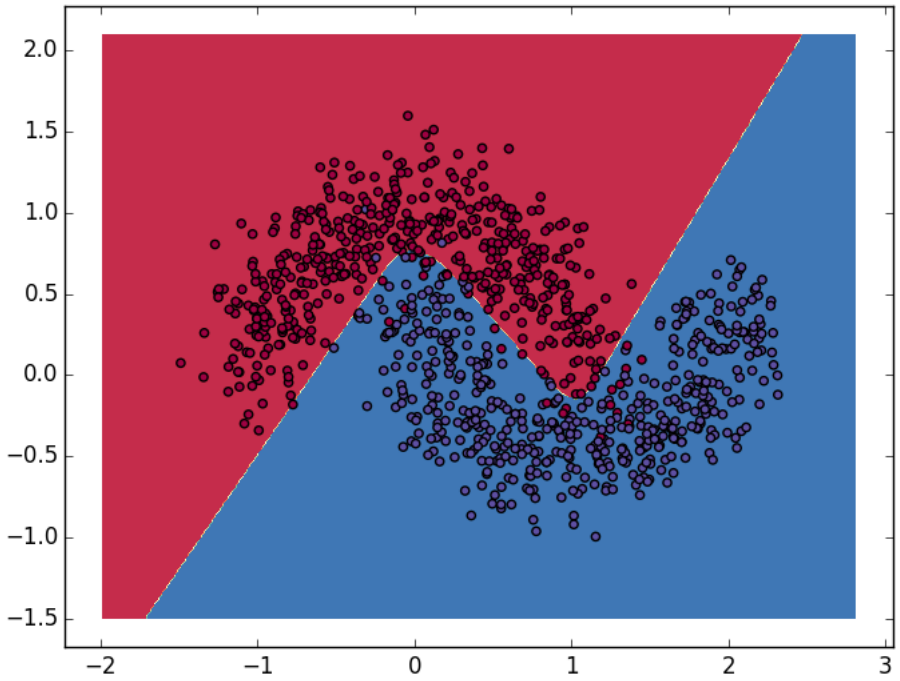
| | | Results |
|---|---|---|
| Trial | Linear | NonLinear |
| Two Layers | | |

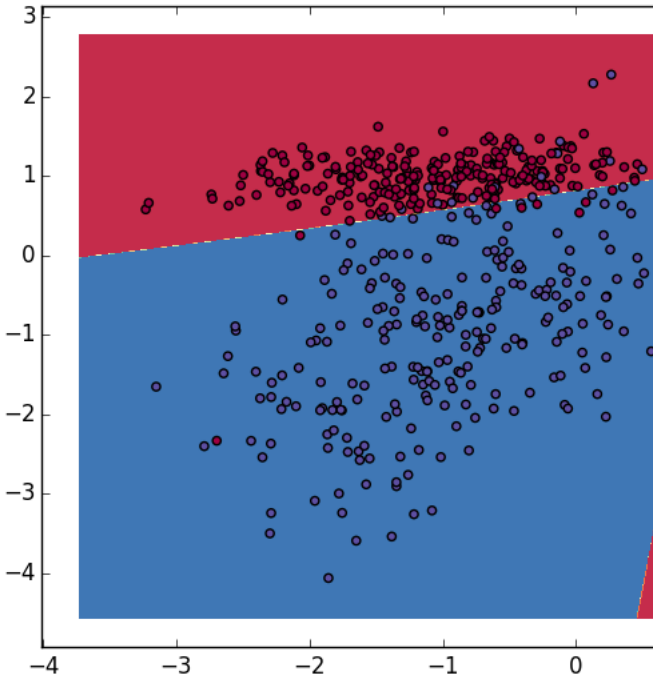
Result



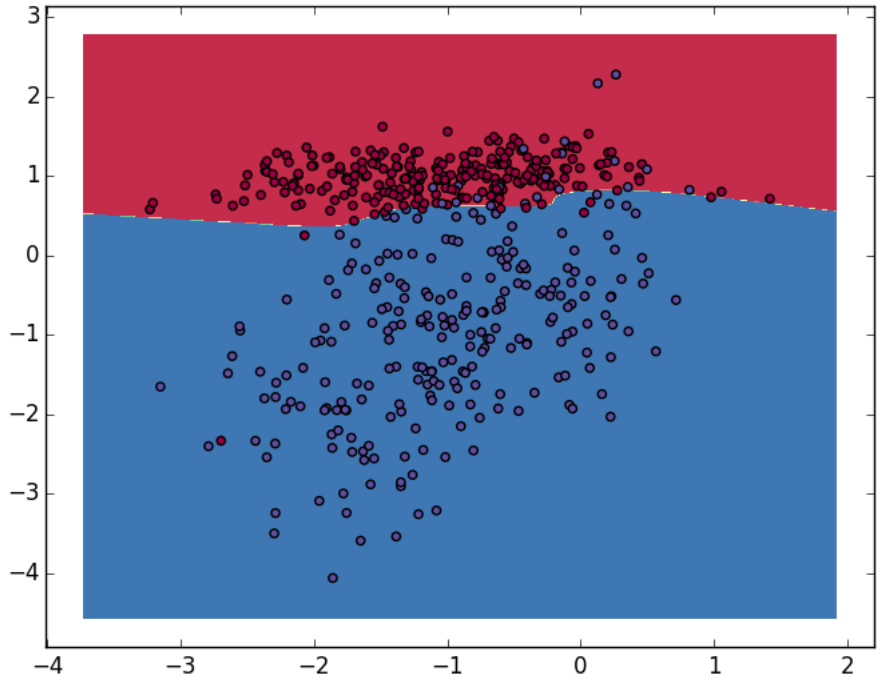



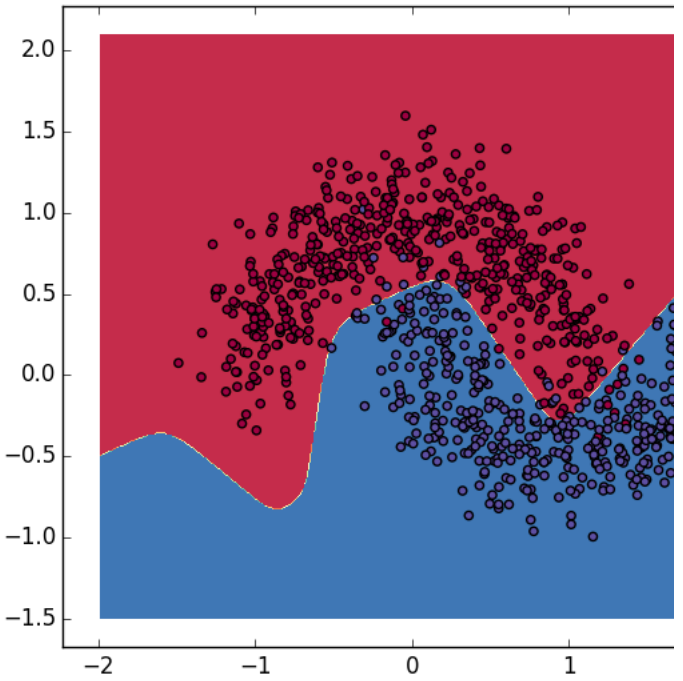

Three
Layers
Result




Five
Layers
Result




## L2 Regularization

The differences between using L2 Regularization to prevent the overfitting

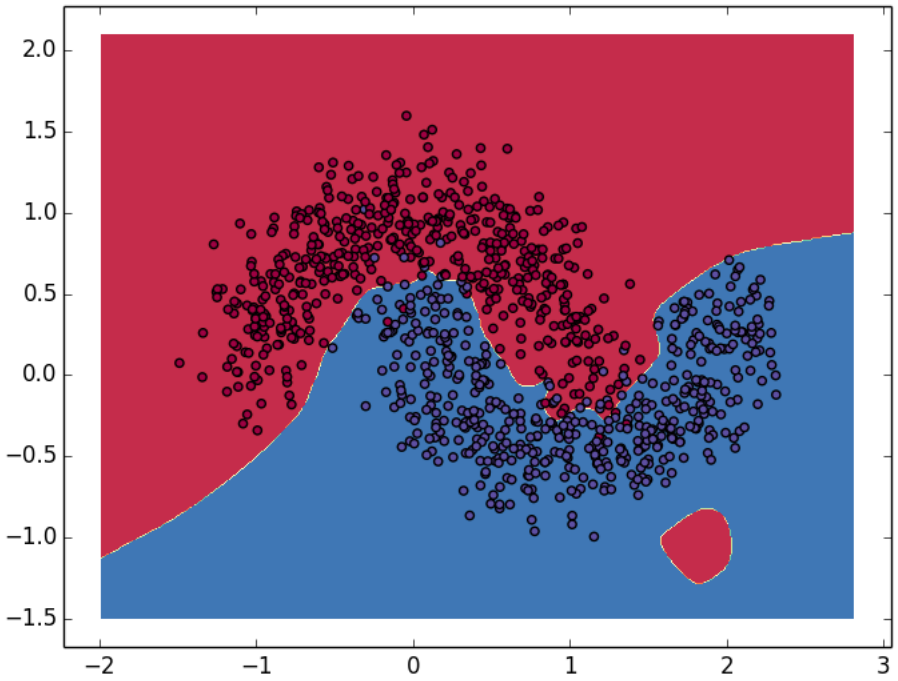
**Results**

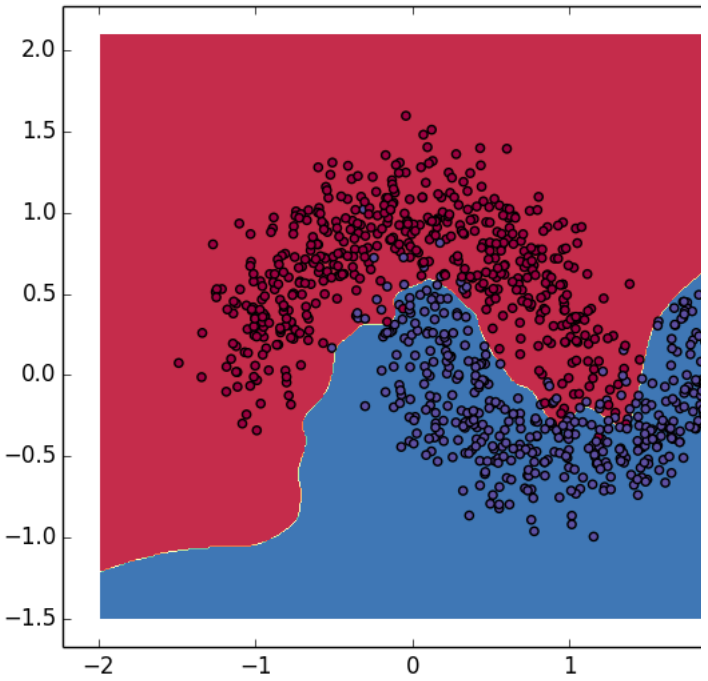
Trial No Regularization Regularization

80

Layers

Result



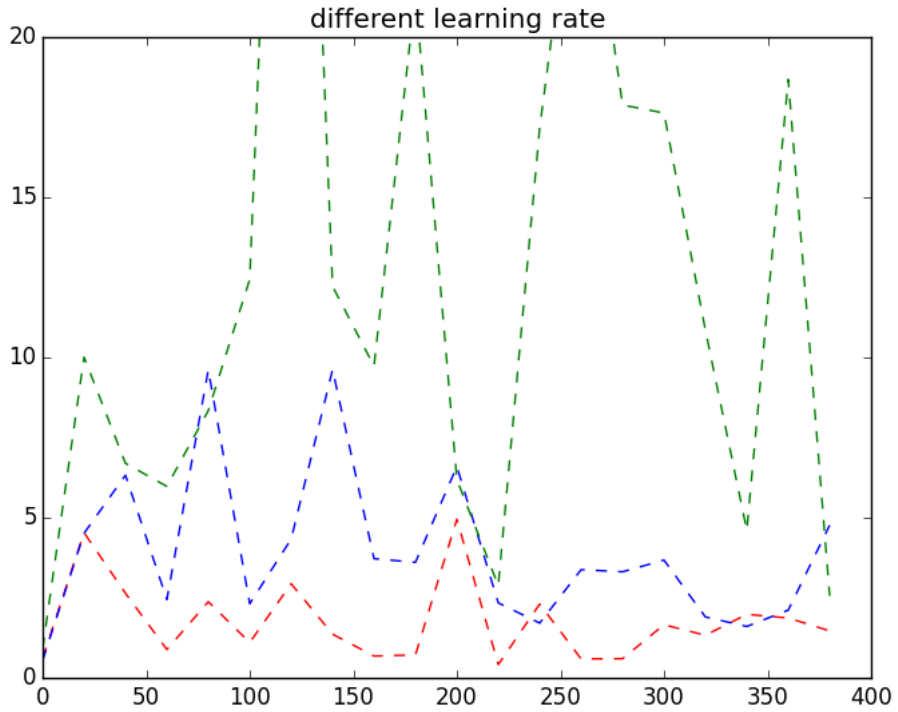

## Various Learning Rate

The different outputs effected by different learning rate

**Results**

Trial  Learning rates: 0.01, 0.025, 0.07

Result




## Discussion

Discuss your method and results:

- The neural network we implemented is reliable and from the results shown above, the network can be trained properly to produce the desired output. Our experiment results also shows how the learning rate affect the performance of the neural network, which a smaller learning rate are more likely to produce a smoother and more accurate curve. Also the test with different hidden layer also shows that more layer does not necessarily mean the output would be more accurate. The L2 Regularization we implemented can also be shown from the result that it would help improve the accuracy of the neural work and avoid the overfitting problem.

- By testing the performance of our neural network with various different parameters, we encountered a problem called overfitting, which is an situation which occurs when building a very complex model, it is quite easy to perfectly fit our data set. But when we evaluate such a complex model on new data, it performs very poorly. In other words, the model does not generalize well. Upon the problem we encountered, we used a method called L2- Regularization to reduce the overfitting problem. The way that the L2-Regularization is to add a regularization term (including parameter lambda) into our linear regression.Â The new term in the equation is the sum of squares of the coefficients (except the bias term) multiplied by the parameter lambda.Lambda = 0 is a super over-fit scenario and Lambda = Infinity brings down the problem to just single mean estimation. Optimizing Lambda is the task we need to solve looking at the trade-off between the prediction accuracy of training sample and prediction accuracy of the hold out sample. We also found another three methods to avoid overfitting: Early Stopping: we divided the data into three subsets in this technique. The first subset is the training set which computes the gradient and update the network weights and biases. The second subset is the validation set. we monitor during the data training. When the networks begins to overfit the data, the error begins to rise. When the validation error increases for a number of iterations, we stop the training and return the weights and biases at the minimum error. Regularization: this method modifies the performance function, usually equals to the sum of squares of the network errors on the training set. Using this performance function causes the network to have smaller weights and biases, and this forces the network response to be smoother and less likely to overfit. Dropout: The key idea of Dropout method is to randomly drop units during the training. This prevents the tranining from adapting too much. The specific way to do is to drop samples from an exponential number of different networks. At test time, it is easy to find out the effect of averaging the predictions of the thinned network and this method will reduces overfitting.

## Conclusions

By visualizing the 2 layer neural network we implemented, we concluded that the network could learn the linear and nonlinear decision boundaries. Since the red and blue dot could be separated by the decision boundary. However, the decision boundary is jsut a straight line without any curve so it is not very accurate at predicting the outputs. While we then implemented our 3 layer neural network with 1 hidden layer. The decision boundaries of both the linear and nonlinear model appears to be more smooth and with curve, which means a more accurate prediction. We also tested our neural network with different learning rate and we concluded from our results that a smaller learning rate will make less error and a larger learning rate tends to be more unstable. We also tested our network with various hidden layers, upto 20. We have observed that when we first increase the number of the layers, the functions seems to have a more accurate predict. However, as the layer size becomes larger and larger, the predict seems like to be very unstable and do not make very accurate predicts.

---

## Credits and Bibliography

Cite any papers or other references you consulted while developing your solution. Citations to papers should include the authors, the year of publication, the title of the work, and the publication information (e.g., book name and publisher; conference proceedings and location; journal name, volume and pages; technical report and institution). Material on the web should include the url and date of access.

Credit any joint work or discussions with your classmates.

---