Machine Learning, Homework 3
Leon Lixing Yu
In collaboration with: Junaid Nomani, Faisal Zaghloul

1 Boosting and overfitting

The decision tree I used is matlab function $fitctree$, which takes inputs, Y, X, $weights$, and $maxNumSplits$. The maximum number of splits are used to control the depth of the tree.
If the $maxNumSplits$ is 3, the depth of the tree is 2. likewise, if the $maxNumSplits$ is 4, the depth of the tree is 3. Since we are given the depth of the tree as 3, I choose the $maxNumSplits = 4$.
The code for this problem is attached in **Appendix-A**.The figures below shows the test errors while changing the number of iterations with fixed tree depth.

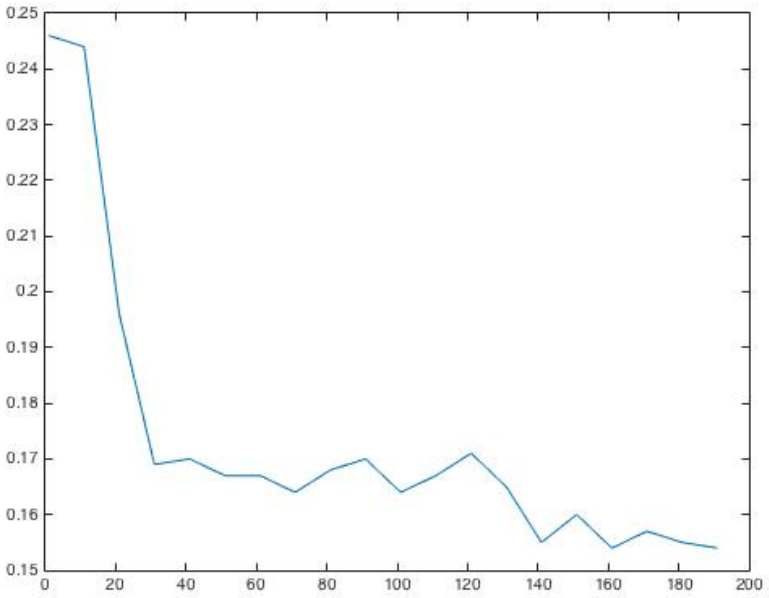

Figure 1: test error for up to 200 iterations.I can see a dip at about 30th iteration. The testing error fluctuates alone the similiar domain.It is possible a overfitting case after 30 iterations. To clearify, I plotted training error to see the convergence.

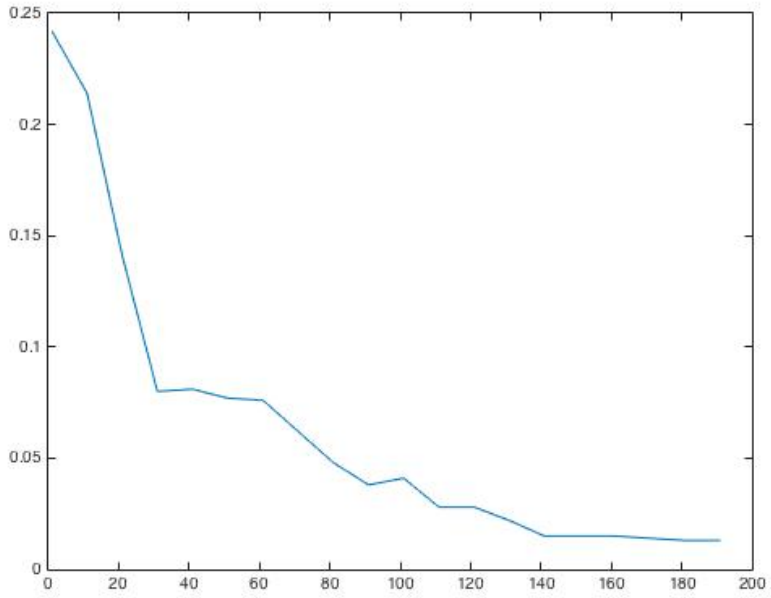

Figure 2: training error for up to 200 iterations.I can see at around 30th iteration, The training takes a huge leap, yet the progression on training accuracy slows down afterwards. I suspect that the training is overfitted to the dataset. after 30th iterations.

The figure below shows the graph for testing error with incremental tree depth while fixing the iteration number to 3

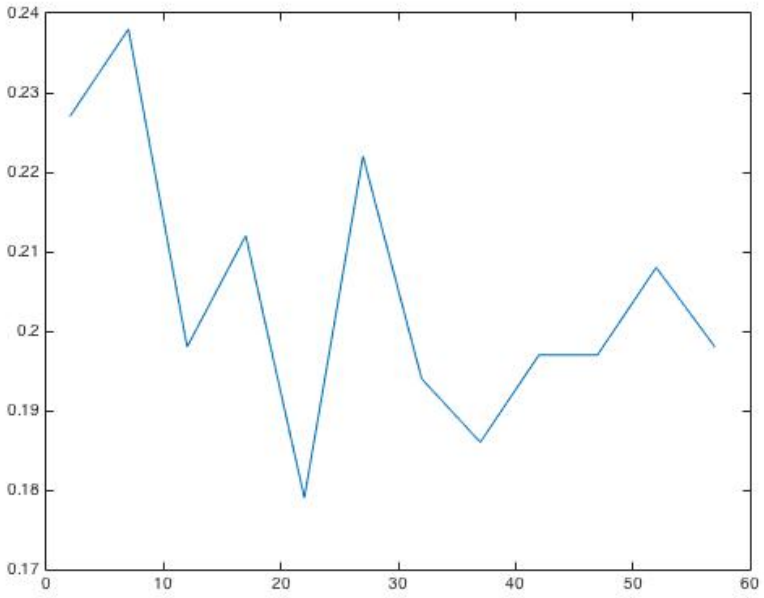

Figure 3: Training error for upto 60 splits. The iteration is fixed to 10. I see that there are a few fluctuation yet the overall error rate is improving slightly

2 boosting for regression

The algorithm is shown below:

```
The initial residual is set to be the training label, Y.

for each iteration
    I train a decision tree of depth 3 to fit the residual
    Get the prediction based on trained tree for testing set and training set
    Update F = F + prediction (for testing)
           Ftrain = Ftrain + predictionTrain ( for training)
    Update residual to the error between the prediction and labels for both testing and training set

The error is the 2 norm of of the residual after all iterations.
The F in the end of all iterations are the prediction for the testing set
```

The implementation of the algorithm is shown in **Appendix-B**. The graph below shows the testing error.

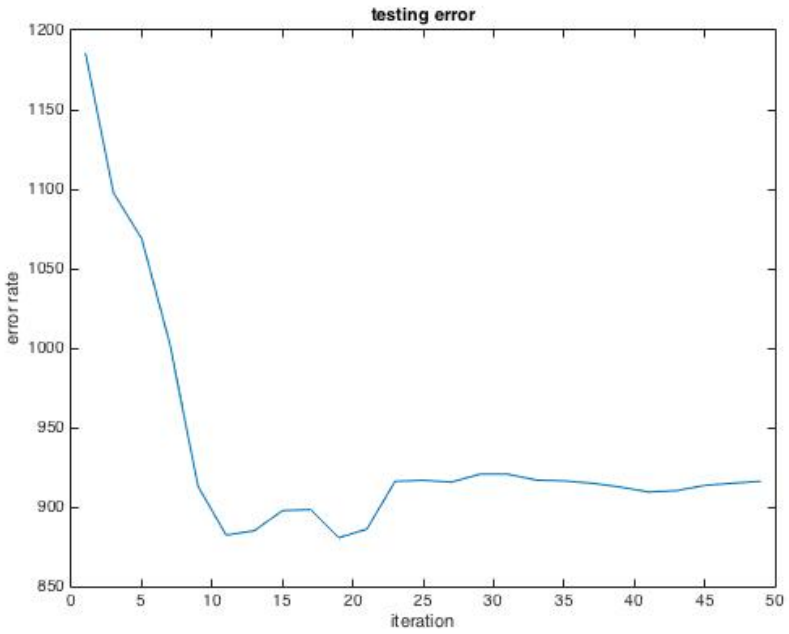

Figure 4: test error for upto 50 iterations.I can see the improvement on higher iteration choices, yet at iteration = 10, the performance gain stopped; thus I suspect overfitting if I go beyond 10 iterations. To verify my claim, I also plot the training error.

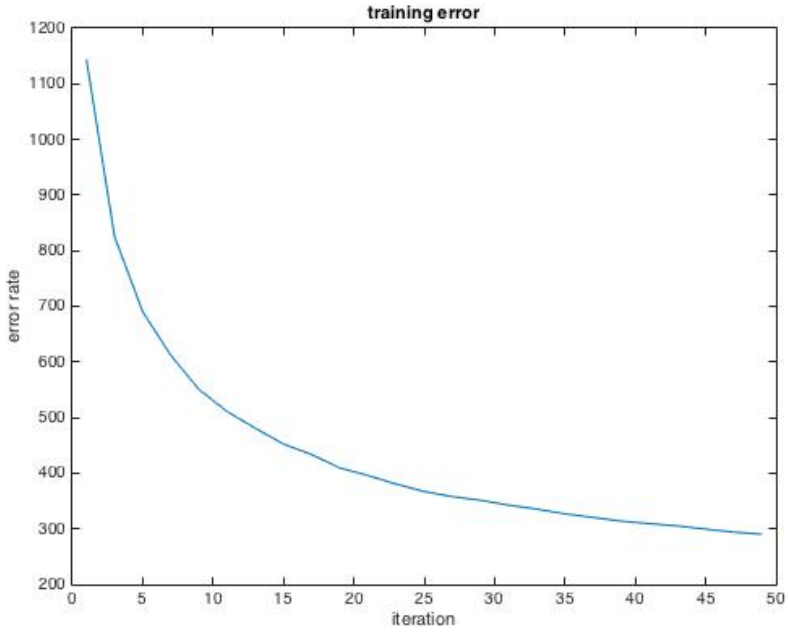

Figure 5: training error for upto 50 iterations. I can see the convergence happening, yet since the testing error is hiccupped at iteration = 10, I think the training is overfitted to the training set after 10th iteration.

3 Model Selection

The code for this section is shown in **Appendix-C**. I use forward selection for this dataset as when I tried LARS, it did not work as well. The forward selection is working much better. The figure below shows the plot comparsion between real labels and fitted functions

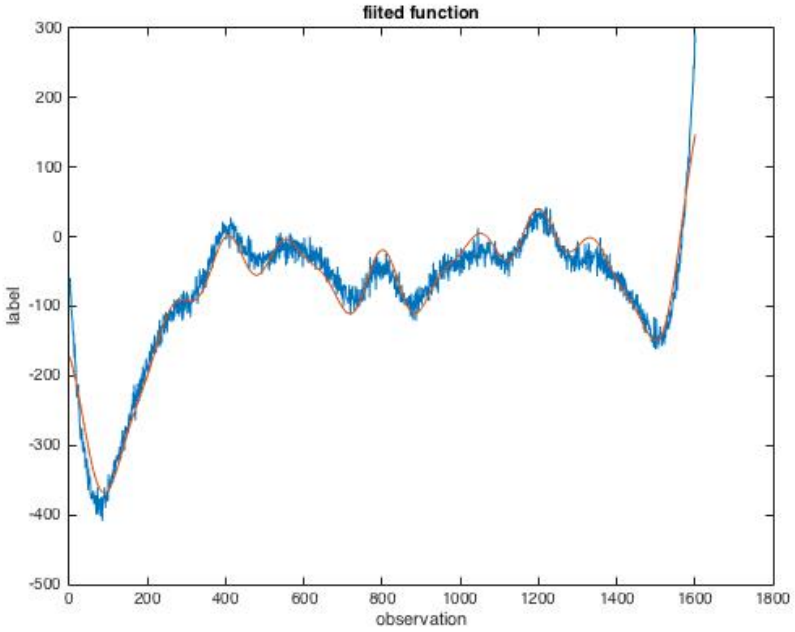

Figure 6: The thin curve is fitted linear combination from the forward selection, and the data points are the labels. I see that the fitted function well fits the data points.

The fitted results are:
x^2 with coeff 2.3543
x^3 with coeff 0.3120
$cos(wx) \quad where\ w \in 2\pi i/1000 \quad where \quad i = 0, 149, 220, 291, 298, 503, 757$ with respective coeffs $-140.0, -141.993, 9.3862, -164.25411, 120.5357, 24.3701, 16.3127$
$exp(-x^2/(2\sigma^2)) \quad where\sigma \in i/100 \quad where \quad i = 88$ with respective coeffs 256.5708.

4 term project

My term project partner is Junaid Nomani. Therefore, my project description would be the same as his.

Ai for game playing.
Partner: Junaid Nomani
Our final project is to use machine learning techniques to get an AI to learn how to play 4x4 tic-tac-toe without any user defined features. We hope to use online learning where the AI improves its performance just by playing. As input it sees the current board state, and as output it chooses the position to place its next token. It trains based only on if it won a game or lost a game. It can play against a human to learn, but to make it faster and so we dont have to play thousands of games of tic-tac-toe we will use hand-coded AI for the learning AI to play against.
We hope to try a few algorithms to see which work best, but will probably prefer using feed-forward neural networks and maybe SVMs. We will test the AIs performance by looking at its winrate against the hand-coded AI for recent games it has played.