Boston Housing Price Learnings

Final Project for UD675 Supervised Learning

Yi Li

ly.protegee@gmail.com

Part I Boston Housing Price estimate

1, For the given feature set: [11.95, 0.00, 18.100, 0, 0.6590, 5.6090, 90.00, 1.385, 24, 680.0, 20.20, 332.09, 12.13], I get a best house price guess at around 20.1 (in whatever unit).

I made this guess from the Adaptive Boosting method with 60 learners, and the Decision Tree method with max depth of 7, with all the available data as training data. I do each learning 10 times and take the average as my final estimate. The results are shown below.

```
Adaptive Boosting results
20.1088
20.2827956989
21.3127962085
20.4523809524
20.48
21.1974789916
19.1197183099
19.7033834586
20.7224334601
20.618128655
the avg est is:  20.3997915735
```

```
Decision Tree results
19.9974683544
19.9974683544
19.9974683544
19.9974683544
19.9974683544
19.9974683544
19.9974683544
19.9974683544
19.9974683544
19.9974683544
the avg est is:  19.9974683544
```

From the results we see the two models give quite close answers. And within each model, the learning results are quite stable, especially for the Decision Tree method, it seems for each round they generate exactly the same trees.

I choose these two models because they perform better than others. They provide low cross-validation errors, which mean they are accurate; and they provide very close estimates when varying the complexity in a small range, which means they have stable estimates.

2, to explore the variances of the estimates, I generate the average estimates for 50-90 learners for Adaptive Boosting model, and the average estimates for 5-9 max depths for Decision Tree model, and calculate the variance for each. Please check the results below.

```
Adaptive Boosting results
the avg est of  50  learners is:  20.3445791045
the avg est of  60  learners is:  20.5978278114
the avg est of  70  learners is:  20.4280251317
the avg est of  80  learners is:  20.2251508009
the avg est of  90  learners is:  20.0009137812
mean:  20.319299326 ; variance:  0.0400543570255
```

```
Decision Tree results
the avg est of  5  max depth is:  20.9677631579
the avg est of  6  max depth is:  20.7659863946
the avg est of  7  max depth is:  19.9974683544
the avg est of  8  max depth is:  18.8166666667
the avg est of  9  max depth is:  19.3272727273
mean:  19.9750314602 ; variance:  0.674605943399
```

From the results we could see that the Adaptive Boosting method is very stable for different complexities, and Decision Tree method is quite stable as well. Both of them have relatively smaller variances.

Please check codes for part I in Appendix 1.

Part II Convince a layman

I choose the models because

1, they produce more accurate estimates than others. When we use 70% of the data to train the model, produce the estimates, and compare the results with the rest 30% of data, we find the models I choose create much closer estimates, which means the models are more accurate.

2, they produce more stable estimates than others. When we produce estimates using different but close complexities of each model, the models I choose create very close estimates, with much less volatility than other models. Also, the two models I choose give very close final estimates, while other models' estimates are quite far from these numbers.

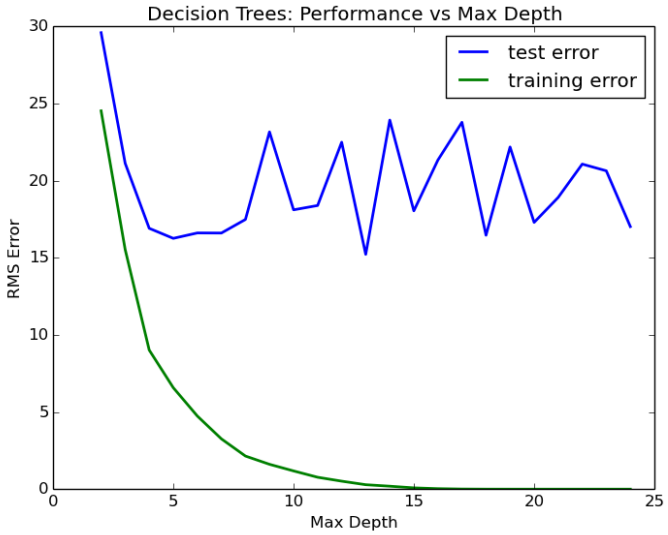
Part III Convince the Udacity coach

1, Decision Tree

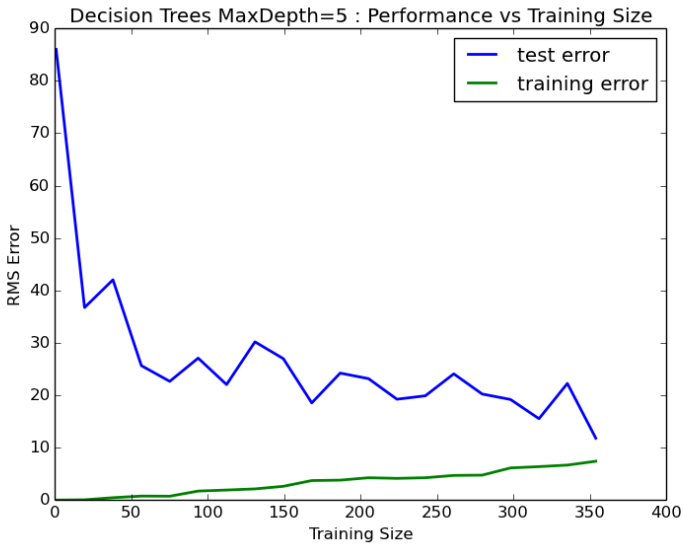
For Decision Tree, below is the Model Complexity graph. Model complexity was increased by changing the max depth of the decision tree. From it we see the model become cyclical since the max depth is larger than 5. I think a max depth of 5 would best fit the training data.

- When the max depth is lower than 5, the hypothesis under fits the data. The model is biased, it does not follow the training data;
- On the other hand, when the max depth is higher than 5, the hypothesis over fits the data. There are less bias but high variance, because they are sensitive to small changes of data.

The best max depth point when the model has neither high bias nor high variance is around 5.



Below is the learning curve for the 5 max depth decision tree. From the plot we see that the test error is obviously higher than the training error, so the model is of relatively high variance (sensitive to the training data), and over fits the data a little bit. From the plot we also see that both the training and test errors are not very high, and it following the training data quite well, so the model is not quite biased. Adding more data in this case will help to improve generalization.



For the new set of input, I create the estimate using all the data as training data, and using the decision tree max depth from 5 to 9, to create the estimates. From the results we see that the highest estimate is 20.97 from 5 max depth, the lowest estimate is 18.82 from 8 max depth, the average of these estimates is 19.98, and the variance of these estimates is 0.67, which is low. From these statistics we see the numbers are close to each other, so the different complexities give stable estimates. These results

matches the low cross-validation errors from the complexity plot and learning curve, and demonstrate again the adaptive boosting method is a good one for this case.

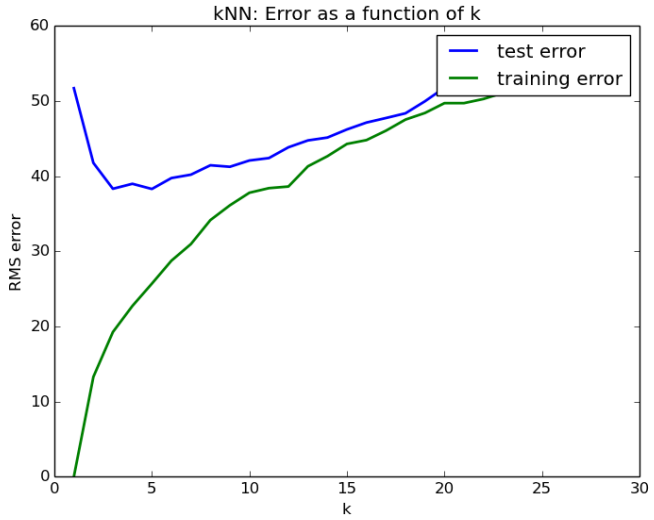
```
Decision Tree results
the avg est of  5  max depth is:  20.9677631579
the avg est of  6  max depth is:  20.7659863946
the avg est of  7  max depth is:  19.9974683544
the avg est of  8  max depth is:  18.8166666667
the avg est of  9  max depth is:  19.3272727273
mean:  19.9750314602 ; variance:  0.674605943399
```

2, K Nearest Neighbors

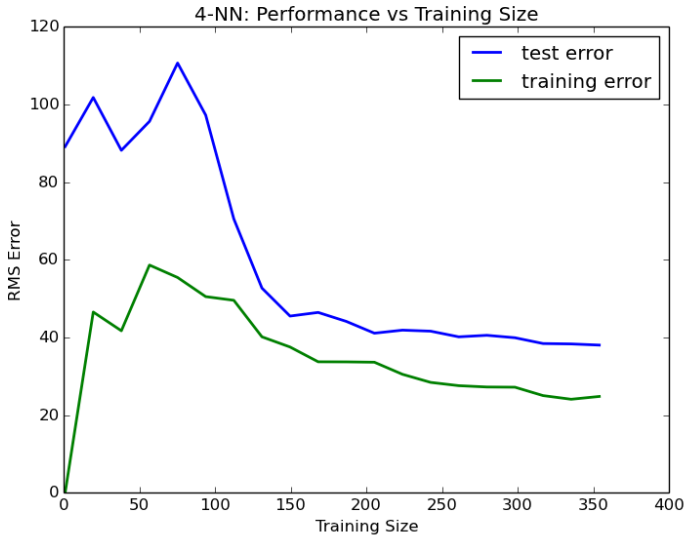
For k Nearest Neighbors, below is the Model Complexity graph. Model complexity was increased by changing the max depth of the decision tree. From it we see k=4 would best fit the training data.

- When k is smaller than 4, the training error is low while the test error is very high, the model is not accurate and is biased, and it seems to be too sensitive to the training data as well so is of high variance;
- On the other hand, when k is larger than 4, both the training and test errors are very high, so the model is biased;

The best k point to make the model less biased and with lower variance is around 4.



Below is the learning curve for the 4-NN. From the plot we see first both training and test errors are high, which means the model is not quite accurate, and of high bias. Also we see the training error is smaller than test error, which means the model is too sensitive to the training data, and of quite high variance. The model over fits the data a little bit. From the graph we also see that adding more data in this case will help to improve generalization.



For the new set of input, I create the estimate using all the data as training data, and using the 3 to 7 nearest neighbors, to create the estimates. From the results we see that the highest estimate is 27.8 from 4 nearest neighbors, the lowest estimate is 20.4 from 3 nearest neighbors, the average of these estimates is 24.22, and the variance of these estimates is 5.98, which is very high and over 20% of the mean. From these statistics we see the estimates are very volatile, and very sensitive to the change of k. The k-NN method might not give good estimates as the results are not stable.

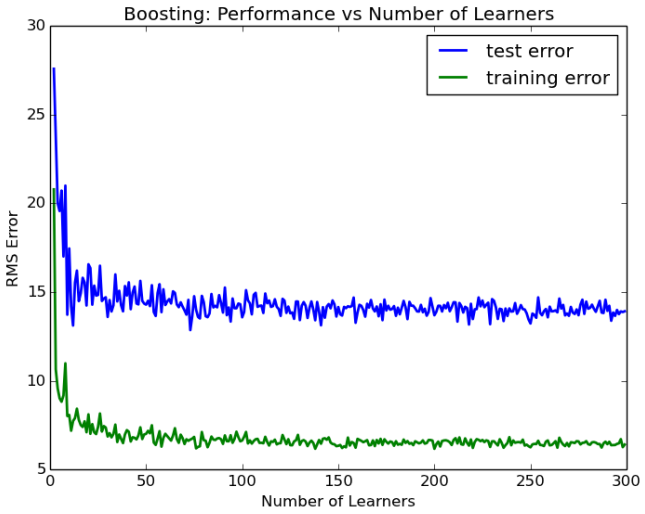
```
k Nearest Neighbors results
the avg est of  3  nearest neighbors is:  20.4
the avg est of  4  nearest neighbors is:  27.8
the avg est of  5  nearest neighbors is:  25.58
the avg est of  6  nearest neighbors is:  23.8
the avg est of  7  nearest neighbors is:  23.5285714286
mean:  24.2217142857 ; variance:  5.9825717551
```

3, Adaptive Boosting

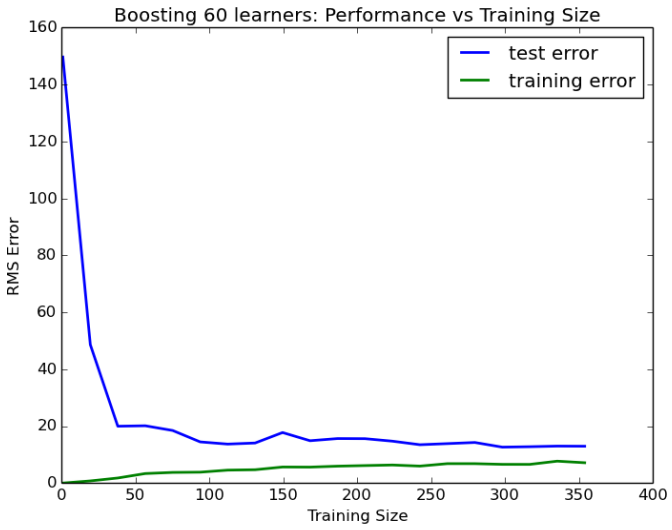
For Adaptive Boosting, below is the Model Complexity graph. Model complexity was increased by changing the max depth of the decision tree. From it we see 60 learners would best fit the training data.

- When there are less than 60 learners, the model is biased, both the training error and test error are relatively high, the model is not accurate;
- On the other hand, when there are more than 60 learners, the model performance is the same. So there is no need to have more complexity;

Under all complexities, the model is of high variance, because the training error is much smaller than the test error, which means the model is too sensitive to the training data. So I think to choose 60 learners could make the model with lowest bias.



Below is the learning curve of the Adaptive Boosting model with 60 learners. From the graph we see that both errors are quite low, so the model is quite accurate and not biased. We could also see that the test error is higher than the training error, which means the model is sensitive to the training data and is of a quite high variance. In general the model still over fits the data. Adding more data in this case will help to improve generalization.



For the new set of input, I create the estimate using all the data as training data, and using the adaptive boosting with 50 to 90 learners, to create the estimates. From the results we see that the highest estimate is 20.60 from 60 learners, the lowest estimate is 20.00 from 90 learners, the average of these estimates is 20.32, and the variance of these estimates is only 0.04, which is very low. From these statistics we see the numbers are very close to each other, so the different complexities give very stable estimates. These results matches the low cross-validation errors from the complexity plot and learning curve, and demonstrate again the adaptive boosting method is a good one for this case. Also another

point worth mentioning is that the adaptive boosting estimates are very close to the other good estimator (decision tree)'s estimates, so I believe both of them are good estimators.

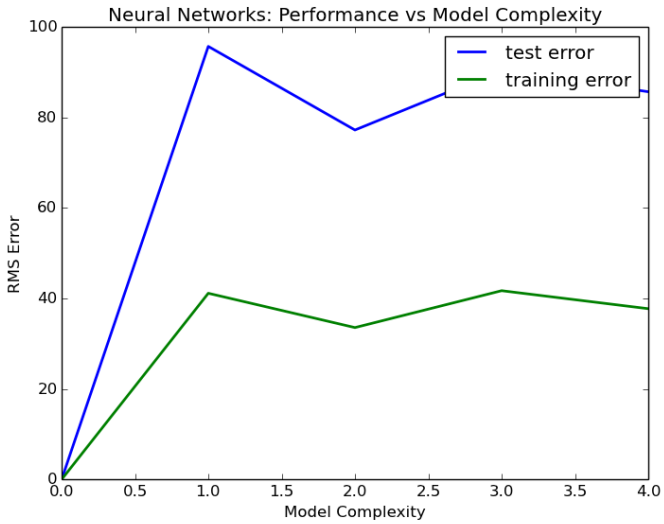
```
Adaptive Boosting results
the avg est of  50  learners is:  20.3445791045
the avg est of  60  learners is:  20.5978278114
the avg est of  70  learners is:  20.4280251317
the avg est of  80  learners is:  20.2251508009
the avg est of  90  learners is:  20.0009137812
mean:  20.319299326 ; variance:  0.0400543570255
```

4, Neural Network

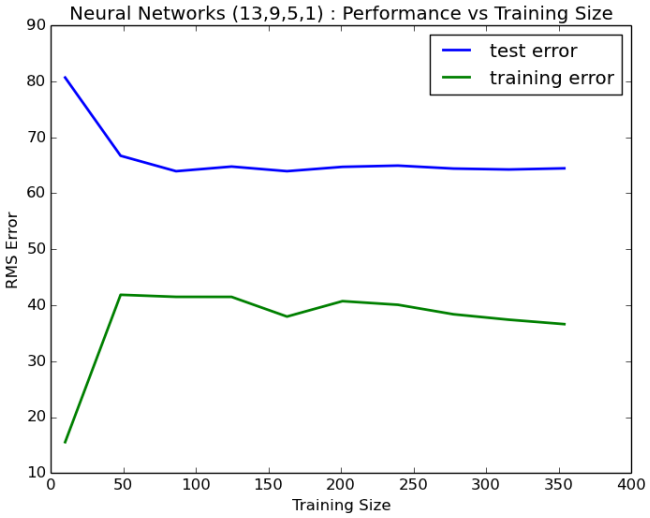
For Neural Network, below is the Model Complexity graph. Model complexity was increased by changing the max depth of the decision tree. From it we see that 2 hidden layers would best fit the training data.

- When there are only one hidden layer, the model is biased, both the training error and test error are relatively high, the model is not accurate;
- On the other hand, when there are more than 2 hidden layers, both errors increase, so the model is of high bias as well;

Under all complexities, the model is of high variance, because the training error is much smaller than the test error, which means the model is too sensitive to the training data. So I think to choose two hidden layers could make the model with lowest bias.



Below the learning curve of the Neural Network model with two hidden layers with 9 and 5 nodes each. From the plot we see that both training and test errors are high, so the model is not accurate and biased; also the test error is much higher than the training error, so the model is sensitive to the training data, and is of high variance. The model over fits the data. In addition we see that adding more data in this case doesn't help to improve generalization.



For the new set of input, I create the estimate using all the data as training data, and using the neural network with 1 to 4 hidden layers, to create the estimates. From the results we see that the highest estimate is 24.00 from 3 hidden layers, the lowest estimate is 21.54 from 1 hidden layer, the average of these estimates is 22.81, and the variance of these estimates is only 0.78. From these statistics we see neural network gives quite stable estimates, but these numbers are not quite close to what the two best estimators (adaptive boosting and decision tree) give. As the cross-validation errors are relatively high for neural network learnings, I think it is not the best method to take for case.

```
Neural Network results
the est of  1  hidden layers is:  [ 21.53912761]
the est of  2  hidden layers is:  [ 22.64134163]
the est of  3  hidden layers is:  [ 23.99911496]
the est of  4  hidden layers is:  [ 23.07183379]
mean:  22.812854497 ; variance:  0.781520255703
```

5, Conclusion

So as a conclusion for all the four methods, we see the Adaptive Boosting and Decision Tree methods produce much lower validation error, so they perform better than others. All of these models show high variance and over-fit attributes, I think it is a systematic issue to the models themselves. In general to increase the training size could significantly enhance the model performance.

Please check codes for Part III in Appendix 2.

Appendix 1

```
from sklearn.utils import shuffle
from sklearn.metrics import mean_squared_error
from sklearn import datasets
from sklearn.ensemble import AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsRegressor

from pybrain.structure import FeedForwardNetwork
from pybrain.tools.shortcuts import buildNetwork
from pybrain.datasets import SupervisedDataSet
from pybrain.supervised.trainers import BackpropTrainer


boston = datasets.load_boston()
X, y = shuffle(boston.data, boston.target)

X_train, y_train = X, y
X_est=[[11.95,0.00,18.100,0,0.6590,5.6090,90.00,1.385,24,680.0,20.20,332.09,12.13]]
```

```
####### PART I ########
### 1 AB
print "Adaptive Boosting results"
y_est=zeros(10)
for k in range(0,10):
    regressor = AdaBoostRegressor(n_estimators=60)
    regressor.fit(X_train, y_train)
    y_est[k]=regressor.predict(X_est)
    print y_est[k]
y_est_exp=mean(y_est)
print "the avg est is: ",y_est_exp

### 2 DT
print "Decision Tree results"
y_est=zeros(10)
for k in range(0,10):
    regressor =DecisionTreeRegressor(max_depth=7)
    regressor.fit(X_train, y_train)
    y_est[k]=regressor.predict(X_est)
    print y_est[k]
y_est_exp=mean(y_est)
print "the avg est is: ",y_est_exp
```

Appendix 2

```
####### PART III ########
### 1 AB
print "Adaptive Boosting results"
y_est_all=[]
for num in range(50,100,10):
    y_est=zeros(10)
    for k in range(0,10):
        regressor = AdaBoostRegressor(n_estimators=num)
        regressor.fit(X_train, y_train)
        y_est[k]=regressor.predict(X_est)
    y_est_exp=mean(y_est)
    print "the avg est of ",num," learners is: ",y_est_exp
    y_est_all.append(y_est_exp)
y_est_all_exp=mean(y_est_all)
y_est_all_var=var(y_est_all)
print "mean: ",y_est_all_exp,"; variance: ",y_est_all_var


### 2 DT
print "Decision Tree results"
y_est_all=[]
for num in range(5,10):
    y_est=zeros(10)
    for k in range(0,10):
        regressor = DecisionTreeRegressor(max_depth=num)
        regressor.fit(X_train, y_train)
        y_est[k]=regressor.predict(X_est)
    y_est_exp=mean(y_est)
    print "the avg est of ",num," max depth is: ",y_est_exp
    y_est_all.append(y_est_exp)
y_est_all_exp=mean(y_est_all)
y_est_all_var=var(y_est_all)
print "mean: ",y_est_all_exp,"; variance: ",y_est_all_var


### 3 kNN
print "k Nearest Neighbors results"
y_est_all=[]
for num in range(3,8):
    y_est=zeros(10)
    for k in range(0,10):
        neigh = KNeighborsRegressor(n_neighbors=num)
        neigh.fit(X_train, y_train)
        y_est[k]=neigh.predict(X_est)
    y_est_exp=mean(y_est)
    print "the avg est of ",num," nearest neighbors is: ",y_est_exp
    y_est_all.append(y_est_exp)
y_est_all_exp=mean(y_est_all)
y_est_all_var=var(y_est_all)
print "mean: ",y_est_all_exp,"; variance: ",y_est_all_var
```

```
### 4 NN
print "Neural Network results"
net = []
net.append(buildNetwork(13,7,1))
net.append(buildNetwork(13,9,5,1))
net.append(buildNetwork(13,9,7,3,1))
net.append(buildNetwork(13,9,7,3,2,1))
net_arr = range(0, len(net))

max_epochs = 50
train_err = zeros(len(net))
ds = SupervisedDataSet(13, 1)
for j in range(1, len(X_train)):
    ds.addSample(X_train[j], y_train[j])

y_est_all=[]
for i in range(0, len(net)):
    trainer = BackpropTrainer(net[i], ds)
    for k in range(1, max_epochs):
        train_err[i] = trainer.train()
    y_est= net[i].activate(X_est[0])
    print "the est of ",i+1," hidden layers is: ",y_est
    y_est_all.append(y_est)
y_est_all_exp=mean(y_est_all)
y_est_all_var=var(y_est_all)
print "mean: ",y_est_all_exp,"; variance: ",y_est_all_var
```