Machine Learning Nanodegree - Project 01

Pedro M Duarte
pmd323@gmail.com
January 6, 2016

I. Statistical Analysis and Data Exploration

size of data	506
number of features	13
minimum price	5
maximum price	50
mean price	22.5328
median price	21.2
standard deviation	9.18801

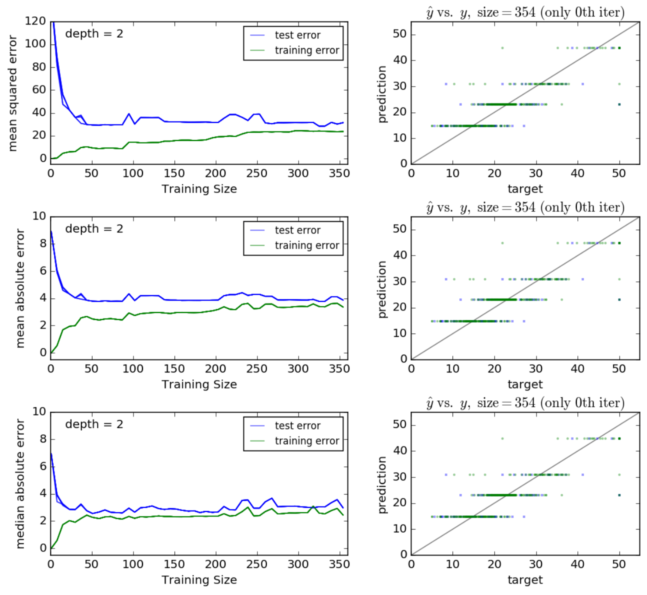

Fig. 1. DecisiontTreeRegressor with depth=2. (Top) Mean squared error. (Middle) Mean absolute error. (Bottom) Median absolute error. In the left panels, 10 independent realizations of the decision tree regressions are shown. Note that the right panel is the same for all three graphs, since the decision tree regressor does not take into account our performance metric to fit the model. The black line on the right panel is a line with slope=1 which represents the ideal model.

II. Evaluating Model Performance

Q: *Which measure of model performance is best to use for predicting Boston housing data and analyzing the errors?*

Q:*Why do you think this measurement most appropriate?*

Q:*Why might the other measurements not be appropriate here?*

A: I will answer the three questions above in the following paragraphs.

From the `sklearn` documentation we have the following choices:

- `metrics.explained_variance_score`: Explained variance regression score function
- `metrics.r2_score`: R^2 (coefficient of determination) regression score function.
- `metrics.mean_absolute_error`: Mean absolute error regression loss
- `metrics.mean_squared_error`: Mean squared error regression loss
- `metrics.median_absolute_error`: Median absolute error regression loss

The explained variance score, given by [1]

$$\text{explained variance}(y, \hat{y}) = 1 - \frac{\text{Var}\{y - \hat{y}\}}{\text{Var}\{y\}} \tag{1}$$

tells us whether our prediction and the true target are correlated. It is interesting to note that if our prediction and the true values happen to just differ by a constant for all samples, then this score will have the maximum value of 1. It is unlikely that this will happen on real (somewhat noisy) data. Nevertheless, for a target such as house price, which can be fairly linear in some of the features, the fact that an offset prediction can be falsely scored as good is sufficient reason not to use this metric.

In practice, applying the explained variance to the housing data example, I found that it behaved pretty much the same as the R^2 score. This last point brings us to R^2, the coefficient of determination. The problem that I have with this metric is the fact that it is not possible, just from the value of R^2, to get an idea of how bad our prediction will be for a particular house price. R^2 will tell us how well our features are able to describe the observed targets, but does not easily translate to an error in price units. For that reason, I would prefer to use one of the error metrics, rather than the R^2 score.

It is hard to decide, a priori, which of the three error metrics is the best. To inform the decision I simply went ahead and used the three of them on the housing data. Let's take a look at the learning curves obtained for each of the three error metrics at tree depths of 2, 5, and 10. The results are shown in Figs. 1 to 3.

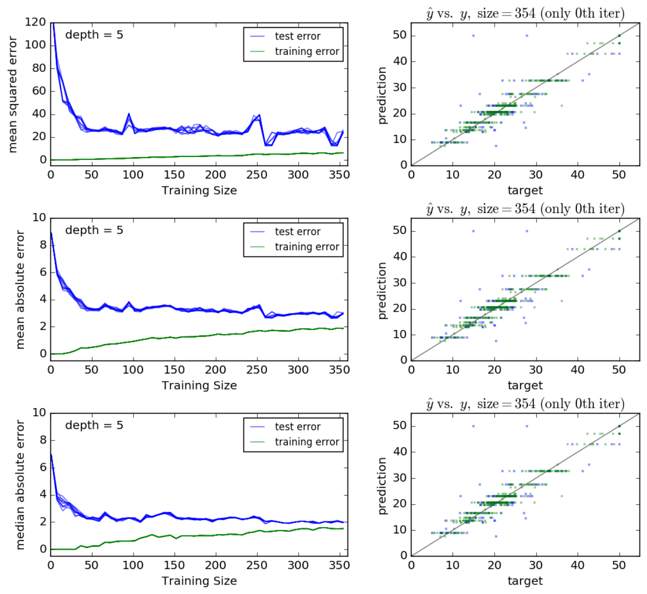

Fig. 2. DecisiontTreeRegressor with depth=5. (Top) Mean squared error. (Middle) Mean absolute error. (Bottom) Median absolute error.

At a value of depth=2 (Fig. 1), the learning curves for train and test data seem to converge to each other for all three of the performance metrics. This indicates low model variance. However, the model complexity might be too low for depth=2, and this low complexity leads to large bias. On the right panel of the graphs, we see at a glance that at depth=2 we only get four different prediction values out of our model; it is not a very complex model.

At a value of depth=5 (Fig. 2), when we use all of the training data we observe a gap between training and test error (indicative of variance issues) for all the three of the error metrics. We do note that the median absolute error has a smaller gap than the mean squared error and the mean absolute error.

Looking at the graph for $\hat{y}$ vs. y, we see that the performance metric is affected by some outlier points that do not conform to our model. We see that these problematic points are generally of one of two kinds: houses that are very expensive and our model predicts to be of lower price, and houses that we predict to be very expensive but really are not. So something about expensive houses is not right with our model.

If we were to look at only the mean squared error or the mean absolute error learning curves at a depth=5, we would be likely to say that the model is overfitting and therefore has large variance. However taking a closer look, as we explained in the previous paragraph, the error in the test data is mostly driven by a few hard-to-predict houses that our model gets very wrong. The median is a statistic that is more robust to outliers than the mean, and that is the reason why the median absolute error shows a smaller gap between training and test errors at depth=5.

Based on the arguments above, I will state that **the best measure of model performance in this case is the median absolute error**. I like the fact that the median is more robust to the presence of extreme outliers in our test predictions.

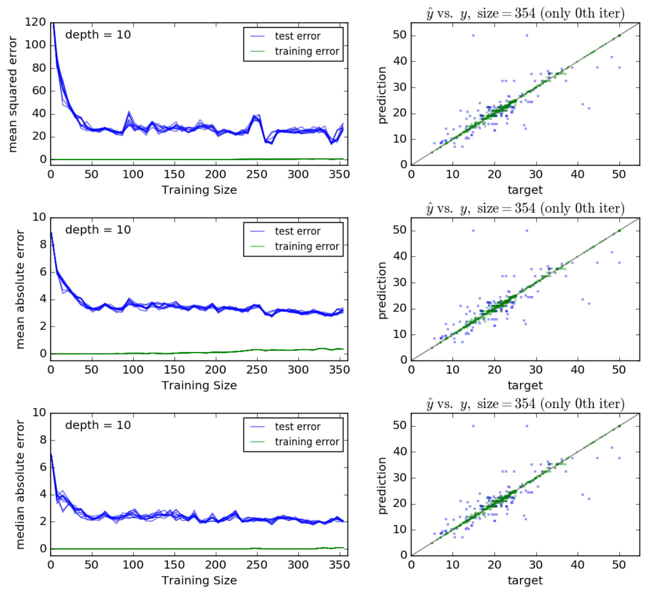

Fig. 3. DecisiontTreeRegressor with depth=10. (Top) Mean squared error. (Middle) Mean absolute error. (Bottom) Median absolute error.

Finally, to end the discussion about the learning curves, I will point out that at a value of depth=10 (Fig. 3), the high variance in the model is evident for all of the three error metrics under consideration here.

Q: *Why is it important to split the Boston housing data into training and testing data?*
Q: *What happens if you do not do this?*

A: The testing data allows us to evaluate the performance of our model. Without it it would be impossible to get an idea of what the right model complexity is for this particular application.

Q: *What does grid search do and why might you want to use it?*

Q: *Why is cross validation useful and why might we use it with grid search?*

A: The purpose of grid search is to tune the model by trying out different model parameters and deciding which combination of parameters is the best based on some scoring metric. In addition to trying out the various parameter combinations, `GridSearchCV` will perform a k-fold cross validation for each set of parameters that it considers.

Using `GridSearchCV` we get the advantages of cross-validation, which is useful because it will allow us to leverage all of the available training data and consider it for our selection of the best model parameters. If we do only a train/test split, we never consider the test data as part of our decision on the final model, and furthermore, the split may be biased in a certain way, and may lead us to a model that does not generalize well to data we have not seen. By performing k-fold cross-validation we gain a layer of protection against these possible systematic issues.

III. Analyzing Model Performance

Q: *Look at all learning curve graphs provided. What is the general trend of training and testing error as training size increases?*

A: For a small amount of training data, the training error is small, and the testing error is large. This is due to the fact that the model can very easily fit all of the training data, but the fit results do not generalize at all to the test data.

As the training size increases the training error increases, since the model now has to actually make some compromises to try to fit the training data. For larger model complexity (depth in this case) this increase in the training error is almost negligible.

On the other hand, as the training size increases the testing error decreases. This happens because, With the ability to fit on a larger number of samples, our model does a better job at generalizing over the test data.

As training size increases to much larger number of samples, training and testing error tend to flatten out and do not change much when more training samples are incorporated. This flattening indicates that, for the particular tree depth under study, we have sufficient training data.

In the case where our model has large bias, the training and testing error may flatten out even a relatively small number of samples, and it may settle at a relatively large error value which is similar for both testing and training. On the other hand, when our model has large variance, we will most likely observe that training and testing error do not approach each other (large gap) even for large number of samples used in training.

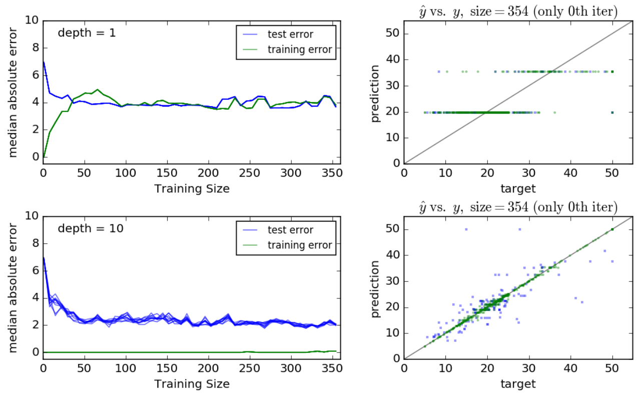

Fig. 4. (Top) depth=1. (Bottom) depth=10. The various lines on the right panels are 10 independent repetitions of the tree regression.

Q: *Look at the learning curves for the decision tree regressor with max depth 1 and 10 (first and last learning curve graphs). When the model is fully trained does it suffer from either high bias/underfitting or high variance/overfitting?*

A: The curves referred to in the question are shown in Fig. 4. For a depth=1, we see that the model does not need many samples to converge on a somewhat fixed value of the performance metric for both training and testing errors. Furthermore the error value that both training and testing converge on is rather large. These two items are an indication that the model is biased and is underfitting the data.

For a depth=10, we have a situation where the error in the training set is always really low, and there is a large gap to the error on the test data, which does not change much past 25 samples or so. The training error is small because, even for large training size, the model has sufficient complexity to fit very well through all of the training data. That overfitting does not generalize to the testing data, which results on the large testing error, which leads to the above mentioned gap between test and train curves. The situation described is indicative of high variance in our model.

Q: *Look at the model complexity graph. How do the training and test error relate to increasing model complexity? Based on this relationship, which model (max depth) best generalizes the dataset and why?*

A: In Fig. 5 we show the model complexity graph obtained using the mean squared error, as well as the model complexity graph obtained using the median absolute error. We observe that, as a general trend, the training and test error both decrease as the model complexity is increased. The training error approaches zero for large maximum depth of the tree, whereas the test error approaches a constant.

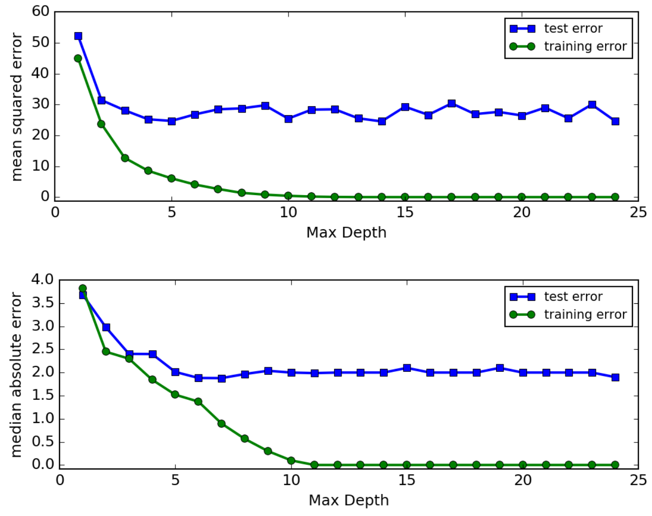

Fig. 5. (Top) Model complexity with the mean squared error performance metric. (Bottom) Model complexity with the median absolute error performance metric.

It is expected for the training error to approach zero as complexity increases, that is in fact the definition of over-fitting data: getting the training error to go to zero with an unnecessarily complex model. If the gap between the training and test error is large we can say that we have a high variance model, even if the training error has not yet approached the asymptotic value of zero.

Looking at the model complexity graph that uses the mean squared error (top of Fig. 5) one may want to use stop at a max depth of 3 or 4 at most. Beyond that point, the gap between training and test data is suspiciously large, and furthermore, the test error is not decreasing anymore with max depth.

On the other hand, if we use the median absolute error (bottom of Fig. 5), we may want to say that a max depth of 5 or 6 is more suitable. As was discussed above, for this particular data I would prefer a performance metric that is not so sensitive to outliers, so I would go with the information provided by the model complexity graph that uses the median absolute error. To be on the conservative side, **I would choose a max depth of 5 as the one that bests generalizes the dataset**.

IV. Model Prediction

Q: *Model makes predicted housing price with detailed model parameters (max depth) reported using grid search. Note due to the small randomization of the code it is recommended to run the program several times to identify the most common/reasonable price/model complexity.*

A: To answer this question requires us to go ahead and use `GridSearchCV` to tune the model parameters. The tuning with `GridSearchCV` is a more automated approach to what we just did visually by exploring the model complexity graph. We will have the chance to compare if our intuition about the best value of the max depth agrees with the results obtained from `GridSearchCV`.

To setup the `GridSearchCV` in the python code, I used randomized k-fold cross validation as provided by the scikit learn function `KFold`. I created two different scorers for the grid search, one using the default R^2 score, and one using the median absolute error.

To eliminate any effects due to the randomness of the decision tree fitting, I ran the grid search 50 times for each of the scorers considered. I collected statistics for the best depth obtained over those 50 runs. The results are shown below:

best depth statistic	R^2	Median abs. err.
mean	5.5	5.78
mode	4	6
std. dev.	1.85	0.73

Considering these results, my final choice for the depth of the tree is going to be depth=6. I used the entire data set to train the depth=6 tree and then proceeded to make a prediction:

predicted house price = 20.765

Q: *Compare prediction to earlier statistics and make a case if you think it is a valid model.*

A: The mean and median prices of houses are 22.5 and 21.2 respectively. The standard deviation of the house prices is somewhat small compared to the range of prices, so chances are that if we pick a house at random it will have a price close enough to the mean. Our prediction for the price of the house is only 1.8 away from the mean price. If the given house were to be an “average“ house picked at random, then our prediction sounds very reasonable.

Revision after first submission. The reviewer on my first submission suggested that I could try to take a look at the nearest neighbors to make an assessment of the accuracy of my prediction. I implemented that code in the file `boston_housing_simple.py`, and the results are summarized in the table below:

N	Average price for N nearest neighbors
5	25.58
10	21.52
20	18.2
30	19.79

We see that the average price of the neighbors seems to vary around 18 to 22 when we consider more than just a few neighbors (but not too many). In the case of nearest neighbors, considering very few neighbors leads to high variance, and considering too many neighbors leads to high bias.

We can at least see that the prediction is not completely outside of the range of the nearest neighbors, which makes us feel somewhat confident about it.

V. Submission details

I took the freedom to significantly modify the python code in order to produce all of the graphs included in this report and to explore a little bit around the problem. In my submission I include the python code that resulted from all my explorations (`boston_housing_complete.py`), as well as a stripped down version which contains only what was explicitly asked for in the assignment (`boston_housing_simple.py`).

References

[1] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.