**October 6, 2017** Lecture 8, Oct. 2, 2017

# Regression Trees

*Lecturer: Justin Domke* *Scribe: Lakshmi Vikraman*

## 1 Summary

Last class we talked about cross validation and regression trees. Today we will continue with the discussion on regression trees.

## 2 Announcements

- Quiz 2 will be held today.
- Instructor Office hours would be held in the classroom today .
- HW1 is due on Wed Oct 4.
- HW2 will be released on Wed Oct 4.
- We will start Classification Models today if time permits.
- Readings : ESL Chapter 4 (Skip sec 4.3 and 4.4.5) , 9.2.3, 9.2.4, 13.3.

## 3 Regression Trees

Recapping from last class, the equation for regression tree is

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

where the input space is divided into M regions $R_1$,$R_2$,....,$R_M$ and outputs $c_m$ if input falls in the region $R_m$.

### 3.1 How to learn trees?

In order to learn trees, we would need the constants $c_m$ as well as the splits which determine the regions $R_m$. Given a dataset, if we have predetermined split regions $R_m$ and we minimize the $MSE$ on the training data , then we set $c_m$ to be the average of the data in a particular region.

$$\hat{c}_m = average(y_i | x_i \in R_m)$$

Please note that the $\hat{c}_m$ is the average value because we are using $MSE$. If we use a different loss function then this would be different. For instance for $MAE$, $\hat{c}_m$ would be the median.

Now how to get the splits? One thing to note is that if we have the splits then we can set $c_m$ to be the average of these regions. So we can do an exhaustive search to determine the splits and set $c_m$. But this search space is huge and it wouldn't be possible to do a complete search. Therefore we are going to find what are called regression stumps. A stump would be a tree with only a single split. The intuition is to go through all possible trees with depth 1 and pick the one with the best $MSE$.

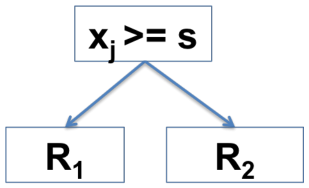

Figure 3.1: Regression Tree

For example if we have a regression tree as shown in figure 3.1, we go to $R_1$ when $x_j < s$ and $R_2$ otherwise. In general, if we have the values for $j$ and $s$, then we can put all the data into the regions and compute the averages to get $c_m$. In order to find the best tree of depth 1, we have to do a brute force search over all possible $j$ and $s$. More formally we can write this as follows,

$$\min_{j,s}(\min_{c_1} \sum_{x_i \in R_1(j,s)} (y_i - c_1)^2 + \min_{c_2} \sum_{x_i \in R_2(j,s)} (y_i - c_2)^2)$$

Here there are four constants $c_1$ , $c_2$, $j$ and $s$. $j$ would have values 1,2,...$p$, where $p$ is the dimensionality of the data, $s$ the values at which the split takes place and $c_1$ and $c_2$ are the values which the tree predicts for the regions $R_1$ and $R_2$ respectively.

We can rewrite the equation as

$$\min_{j,s}( \sum_{x_i \in R_1(j,s)} (y_i - \hat{c}_1)^2 + \sum_{x_i \in R_2(j,s)} (y_i - \hat{c}_2)^2)$$

$$\hat{c}_1 = average(y_i | x_i \in R_1(j,s))$$

$$\hat{c}_2 = average(y_i | x_i \in R_2(j,s))$$

where $\hat{c}_1$ and $\hat{c}_2$ are the averages and are functions of $j$ and $s$.

How do we minimize over all j and s? The answer is that we try all $j$ and $s$ and for every pair, get values for $c_1$ and $c_2$ and pick the best one. The number of $j$'s we consider is $p$. However the number of $s$'s is infinite. One way to resolve this to to split at the data point. A better method is to split at the midpoints between datapoints as shown in figure 3.2 . Therefore for $N$ datapoints we have $N - 1$ splits.

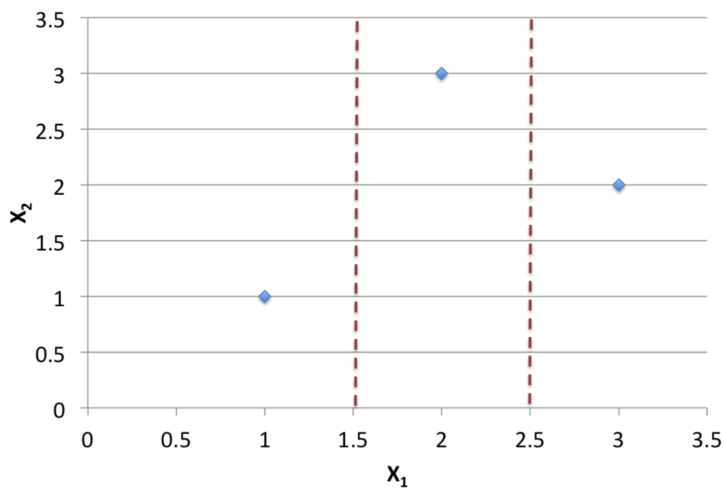

Figure 3.2: Midway Splits

The runtime of a naive implementation is $pN^2$. We can improve this by using recursive algorithms which brings the run time to $pNlogN$.

How to learn regression trees rather than stumps?

- Learn a regression stump.
- Split the data based on where it falls into and recurse.
- Stop eventually (to control capacity of the tree). In order to stop, we can define one of the following parameters : maximum depth , minimum number of datapoints which must fall under a node, maximum number of terminal nodes, minimum number of datapoints which must be allowed for a node to be split, maximum number of splits etc.

We perform this recursion and adopt a greedy strategy to search a subspace instead of the entire search space. This pushes us towards higher bias and lower variance. Also please note that the splits are horizontal or vertical axis aligned with no diagonal splits. When the data are not axis aligned, we don't have good algorithms to find good trees or even stumps. When we rotate the data, the regression tree change. Meaningful features are very helpful in getting good splits. Also, trees can have multiway splits.

## 3.2 Example Data

We will look at a few example data. These are two-dimensional datasets. To simultaneously visualize locations $\hat{x}$ and $\hat{y}$, we use a Voronoi diagram. Each pixel in the plane is colored with an intensity proportional to the value $\hat{y}$ corresponding to the nearest $\hat{x}$. The splits obtained in the first two figures 3.3 and 3.4 are reasonably good. This is because data are fairly axis aligned. In the third figure 3.5 , the splits does not seem to be good even after 15 splits due to the diagonal splits in the data.

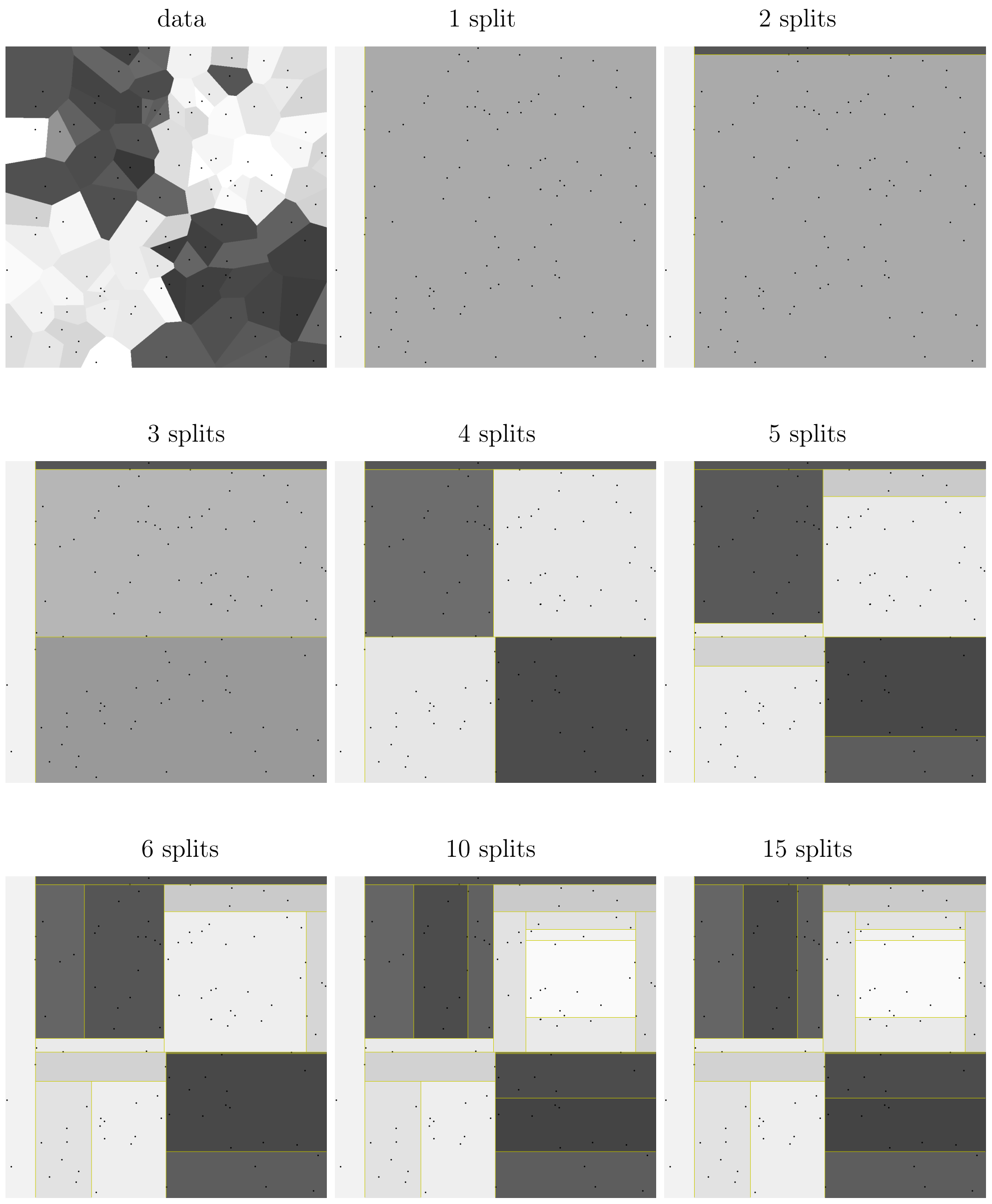

Figure 3.3: Example 1

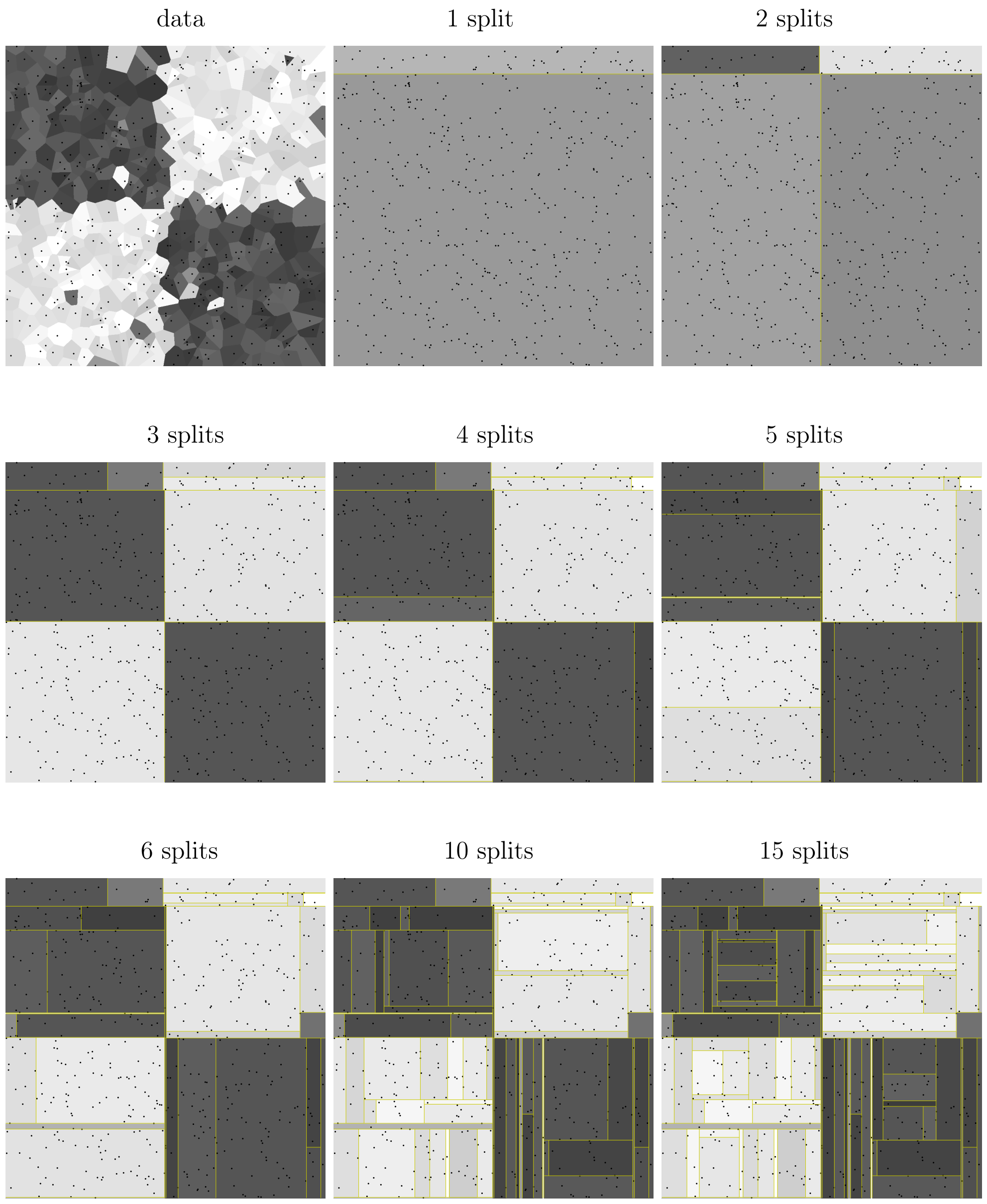

Figure 3.4: Example 2

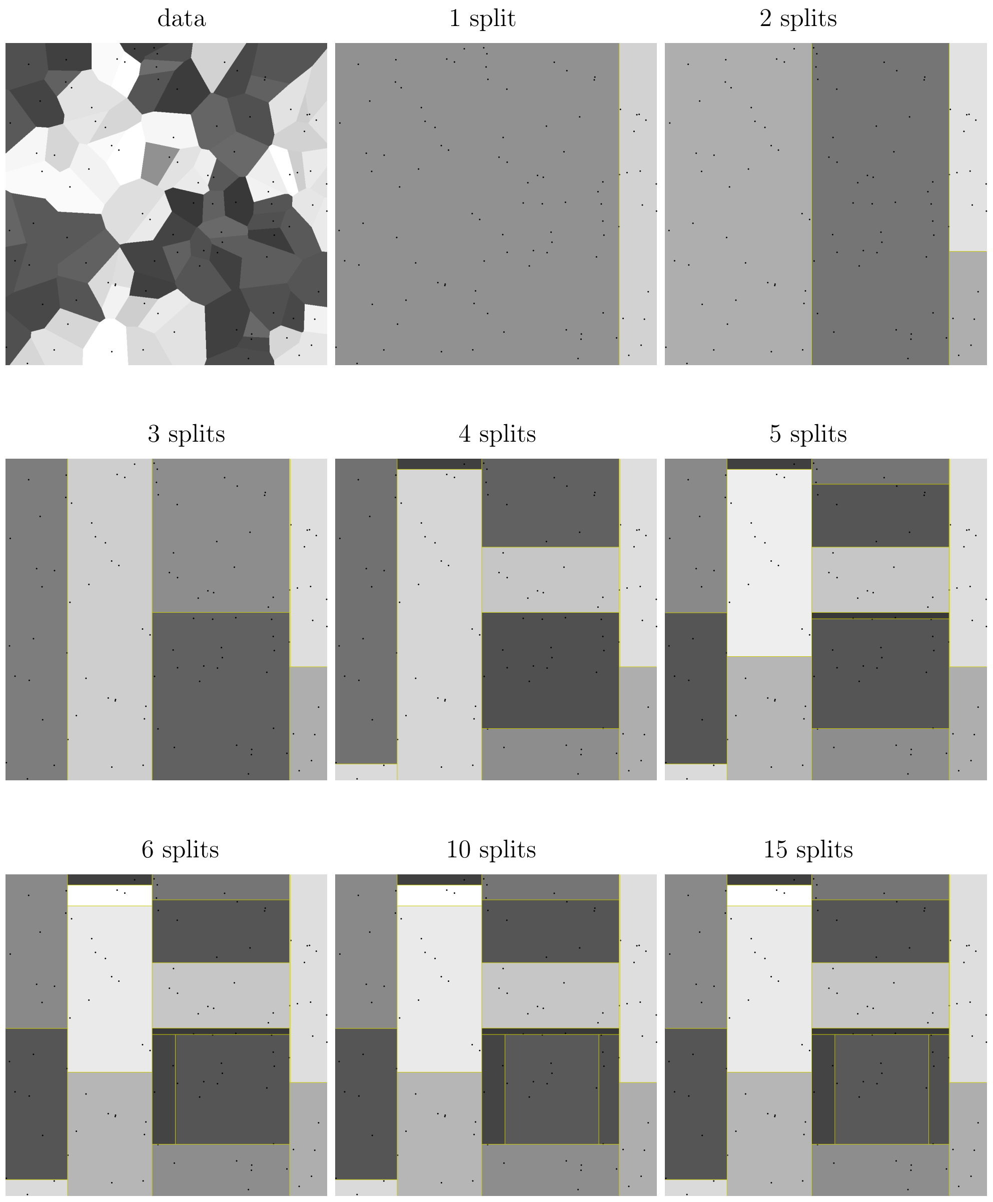

Figure 3.5: Example 3

## 4 Discussion

- The splits in regression trees are axis aligned.
- They are fast to learn and very fast to evaluate. For instance, the complexity to evaluate a tree with depth d is d.
- They can be used for "distillation". This is a technique where a larger model like neural network or ensemble (which may have better generalization properties) is first trained on the data. Next, given a large UNLABELED dataset, the first predictor is used to generate data which is then used to train the tree. The hope is that this will give similar predictions to the first predictor, but be faster since trees are so fast.
- They are a common ingredient in ensembles. Two common methods are boosting and bagging. Boosting averages trees to reduce bias. Bagging fits a bunch of trees which results in a reduction in variance.