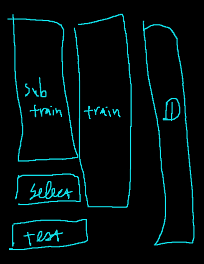


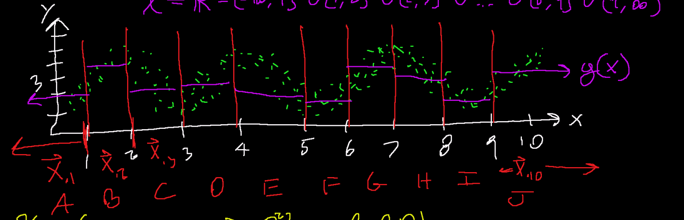
Big question: How to provide the M models to select from in the model selection procedure? How to provide the transformations that define the global search space in the forward stewpise metaalgorithm? In other words, how do we provide an algorithm that can flexible create curlyH for us on-the-fly without overfitting? This problem is the main thing that's addressed by "non-parametric" machine learning.

We will study tree models. The name of the algorithm is called Classification and Regression Trees (CART, 1984) which is really two algorithms:

Classification Trees for $\mathcal{Y} = \{C_1, C_2, \ldots, C_K\}$
Regression Trees for $\mathcal{Y} \subseteq \mathbb{R}$

We will start with regression trees and build it up step-by-step. First, consider the following data with p_raw = 1:

$\mathcal{X} = \mathbb{R} = (-\infty, 1] \cup (1,2] \cup (2,3] \cup \ldots \cup (8,9] \cup (9,\infty)$



$\mathcal{H}_1 = \{w_0 + w_1 x : \vec{w} \in \mathbb{R}^2\}$ BAD!

$\mathcal{H}_2 = \{w_0 + w_1 x + w_2 x^2 : \vec{w} \in \mathbb{R}^3\}$ BAD!

$\mathcal{H}_3 = \{w_0 + w_1 \sin(w_2 x) : \vec{w} \in \mathbb{R}^3\}$ Good!

You can figure this out with p = 1. But with high p... forget it! We need a simple, generalizable idea. Consider:

$$X = \begin{bmatrix} x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ \vdots \\ x_{1n} \end{bmatrix} \longrightarrow X_{tr} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 1 \\ \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

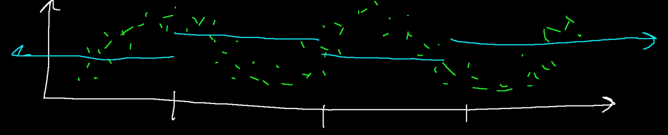
$\mathbb{1}_{x \in (-\infty, 1]}$ $\mathbb{1}_{x \in (1,2]}$ .... $\mathbb{1}_{x \in [9,\infty)}$

Now, let's fit an OLS model to this space which is

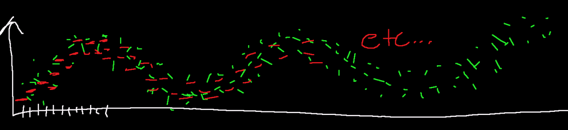
$$\mathcal{H} = \{w_1 \mathbb{1}_{x \leq 1} + w_2 \mathbb{1}_{x \in (1,2]} + \ldots + w_9 \mathbb{1}_{x \in (8,9]} + w_{10} \mathbb{1}_{x > 9} : \vec{w} \in \mathbb{R}^{10}\}$$

$\Rightarrow g(x) =$

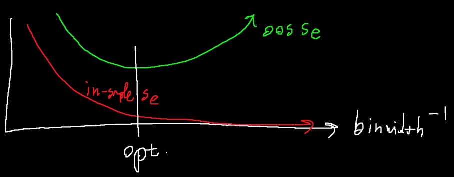
What "choice" (hyperparameter) was necessary to fit this model? The bin width. If the bin width was larger... we underfit!



If the bin width was much smaller... we overfit!



This can be solved with the model selection procedure by using oos data to find the best binwidth on this curve:



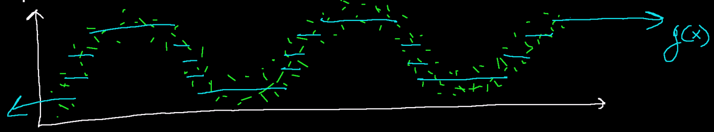
Small binwidth => high degrees of freedom; large binwidth => low degrees of freedom.

There's a huge problem with this modeling procedure!

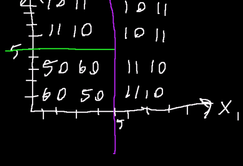
In two dimensions, bins are square. And you fit a level at a certain height (yhat). In three dimensions, bins are cubes and each cube has its own yhat. The problem is: in high p (dimension), the number of different coefficients grows exponentially.

E.g. Boston Housing data with p = 13. If each variable had only 5 bins, then there would be 5^13 = 1,220,703,125 i.e. a billion parameters. Clearly this p_tr > n and it cannot be estimated... and if it were to be estimated it would be overfit.

What about an algorithm that only considers bins that improve predictive performance? No need for bins to be equally sized or spaced.

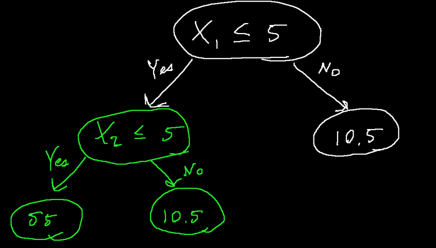


Let's design such an algorithm. Let's consider two dimensions:



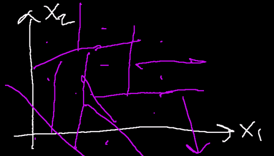
Restrict bins to be separated by an orthogonal-to-axis split of the entire space.

Step 1: Get best two-bin model.
Step 2: Get best three bin model assuming we split a bin's subspace.



At some point we have to stop splitting into new bins and call it quits with out "binary tree model".

You can also consider non-orthogonal to split bin models or curved bin models such as:



but such models are hard to fit... We will consider in this course only orthogonal-to-axis-split hyperrectangular regions.