Lec 18 Math 390.3 4/10/18

Previously, we covered "model selection" e.g.

Mod 1: $g_1 = b_0 + b_1 x$ from $A_1, \mathcal{H}_1 = \{\dots\}$

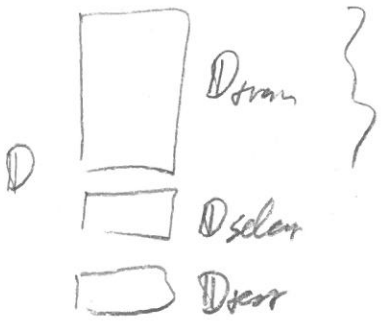
Mod 2: $g_2 = b_0 + b_1 x + b_2 x^2$ from $A_2, \mathcal{H}_2 \{\dots\}$

Mod 3: $g_3 = b_0 + b_1 x + b_2 x^2 + b_3 \log(x)$ from $A_3, \mathcal{H}_3 = \{\dots\}$

⋮

Mod M: $g_M = b_0 + b_1 x + b_2 x^2 + b_3 x^3$ from $A_M, \mathcal{H}_M = \{\dots\}$

We never discussed what models 1...M came from. Above - they were made up. Is there something smarter? There is...



Can we use this data to generate good candidates 1....M?

One such algorithm is:

For linear models, this is usually done iteratively. You start with a large # of candidate predictors (e.g. polynomials, logs, interactions, etc.), then iteratively add the "best one". (Define "definition of "best")

But if you add too many... you overfit!!!

So... use Dselect to stop once you see yourself crossing this threshold from

Under → over fitting

How would backwards stepwise work?

This is called "forward stepwise" linear modeling. DEMO

Two main problems

* (1) You still need an intelligent collection of predictors to choose from at each iteration → are you sure you can specify that?
(2) the model will still be linear. Although you can alter the A to be non-linear models.

Question!

"parametric regression" if $y \in \mathbb{R}$
"parametric classification" $y \in \{0,1\}$

How about something radically new? Whenever we specified $\mathcal{H} = \{\vec{w} \cdot \vec{x} : \vec{w} \in \mathbb{R}^{p+1}\}$ we made a "parametric assumption" i.e. the model will be of this form which is fixed. Would it be nice if $\mathcal{H}$ can adjust flexibly and allow for more complexity if n increases? This is called "non-parametric regression"

Here, the model does not take a prespecified form (like $w_0 + w_1 x$) but is constructed algorithmically to maximize fit to the data ⇒ the data provides the model space & the model estimates.

Many creative ideas for this! Let's discuss one such idea

Classification & Regression Trees (CART, 1984)

↳ classification trees $y \in \{1, \ldots, K\}$
↳ regression trees $y \in \mathbb{R}$

Let's talk about regression trees

WAIT do illustration first

Recall from midterm 2 ... assoc. but no causation conclusion

we see $f(x)$. It looks like a sine curve.

If $\mathcal{H}_1 = \{w_0 + w_1 \sin(w_2 x) : \vec{w} \in \mathbb{R}^3\}$ we would likely do very <u>well</u> with 3 parameters
A? we can still use LS. But we may need to use a numerical optimizer.

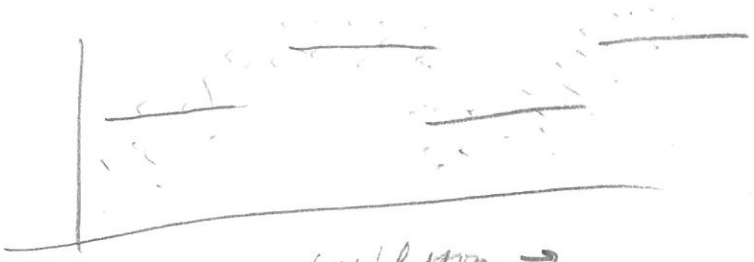
If $\mathcal{H}_2 = \{w_0 + w_1 x + w_2 x^2 ; \vec{w} \in \mathbb{R}^3\}$ we would likely do very <u>poorly</u> with 3 params.

In one dimension this is easy to pick $\mathcal{H}_1$ over $\mathcal{H}_2$. In multiple dimensions, VERY difficult!!

What if we had an algorithm that did the following:

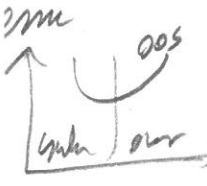
So $g(x)$ is a collection of constant subfunctions where each has a small domain. The union of all domains $= \mathcal{X}$.

You can make even smaller domains for better resolution.



underfitting →
not enough complexity in $\mathcal{H}$ rel. to c_f

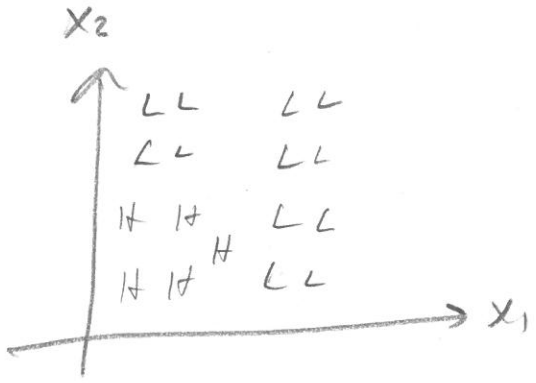
overfitting → too much complexity in $\mathcal{H}$ rel. to c_f & simple size



ex: large bounds
what type of error is prevalent here?
misspecification error
($\mathcal{H}$ not expressive enough)

ex: small bounds
what type of error is prev. here?
estimation error
(not enough data in each bin to make a good decision about what to put the constant value)

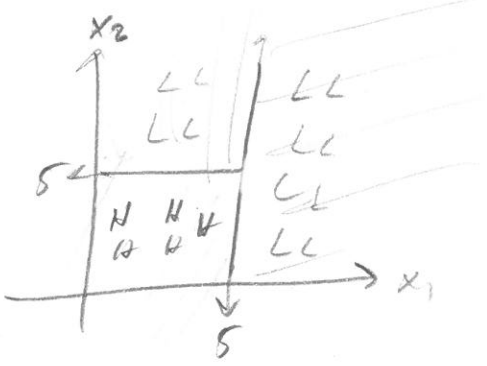
we will discuss how to find a nice balance between underfitting/overfitting later. First let's talk about what happens if $p=2$



let $\mathcal{Y} = \mathbb{R}$
but for purposes of illustration, let's denote L: low value
H: high value

What do bounds look like in 2-dim? Shapes!

Let's make a simplification: shapes must be rectangles whose sides are parallel to the axes. (and always may be infinite) What "bound split" makes most sense here?



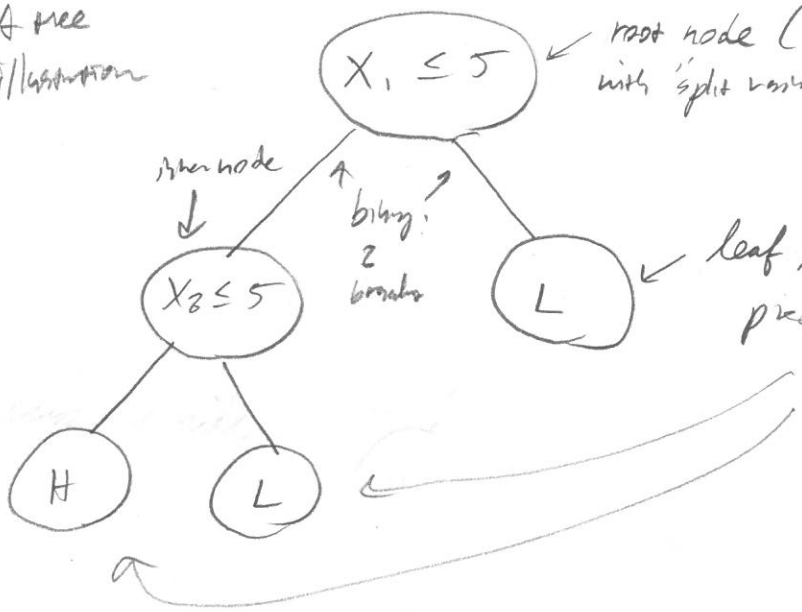
What is this model? 3 leaves (all values in size)

$$g(\vec{x}) = L\,\mathbb{1}_{x_1 > 5} + L\,\mathbb{1}_{x_1 \le 5}\,\mathbb{1}_{x_2 > 5} + H\,\mathbb{1}_{x_1 \le 5}\,\mathbb{1}_{x_2 \le 5}$$

It's technically a linear model...

Is there a better way to visualize this? A "binary tree"

A tree illustration



Is all $\mathcal{X}$ accounted for? Yes. There is a $\hat{y}$ $\forall \vec{x} \in \mathcal{X} = \mathbb{R}^2$.

Where is the rectangle-parallel-to-axis assumption seen?

Rules must be of the form "$X_j \le m$" Creating two rectangles parallel to the X_j direction.

$\mathcal{H} = \{ ? \}$ good question ... hm...

What is $\mathcal{A}$? There are ∞ possible models of this form. How to select (1) splits (2) predictions

model structure (discrete) parameters

MANY!!

One such $\mathcal{A}$.

① Begin with all training data $X, \vec{y}$

② For every possible split at the current node divide the data into $X_l, \vec{y}_l$ & $X_r, \vec{y}_r$ and calculate

$$SSE_l = \sum (y_l - \bar{y}_l)^2 \quad , \quad SSE_r = \sum (y_r - \bar{y}_r)^2$$

over # of data pts under the split

③ Find the split with the lowest overall avg SSE

$$SSE_{avg} = \frac{n_l SSE_l + n_r SSE_r}{n_l + n_r}$$

④ Create the split, split data into two nodes

⑤ Repeat steps 2-4 until "STOP"

STOP: node has $\leq N_0$ data pts inside.

There are many, many variations of the above.

If $N_0 = 1$ ... tree is grown to fit a separate parameter for each data pt. $R^2 = ?$ 100%! Then tree is "grossly overfit" back to not

How is N_0 picked? Via on Model selection procedure

Kind of like backwards stepwise regression.