Moving beyond linearity :

- polynomial regression
- step functions
- regression splines
- smoothing splines
- local regression
- generalized additive model

1 Polynomial regression

$y_i = \beta_0 + \beta_1 x_i + \dots \beta_d x_i^d + \varepsilon_i$

usually up until degree 3-4, after too flexible

can calculate $\mathrm{Var}\, \hat{f}(x_0)$, as least squares gives a variance for each $\hat{\beta}_i$
as well as the covariances → if $\hat{C}$ is the covariance matrix,
and $\ell_0^T = [1, x_0, x_0^2, \dots x_0^d]$, then $\mathrm{Var}\, \hat{f}(x_0) = \ell_0^T \hat{C} \ell_0$

to plot the 95% confidence interval, we calculate
the pointwise standard error and double it (~ Normal assumption) on both sides of the curve

- if we have a binary q, e.g. $Y > 250$ or not we can treat it as logistic regression

e.g. $P(Y_i > 250 \mid X_i) = \dfrac{\exp(\beta_0 + \beta_1 X_i + \dots \beta_d X_i^d)}{1 + \exp(\beta_0 + \dots \beta_d X_i^d)}$

2) Step functions

break X into bins, fit a different constant on all

~ this converts X into an ordered categorical variable

create cutpoints $c_1, \dots c_K$ and construct $K+1$ variables $C_0, \dots C_K$

st.

$$C_0(X) = I(X < c_1)$$
$$C_1(X) = I(c_1 \le X < c_2)$$
$$\vdots$$
$$C_K(X) = I(c_K \le X)$$

and then fit $\quad y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \dots + \beta_K C_K(x_i) + \varepsilon_i$

not using C_0 as redundant w/ the intercept

β_0: mean value of Y for $X < c_1$

β_j: mean increase in Y for $c_j \le X < c_{j+1}$ relative to $X < c_1$

Basis function: $y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \dots + \beta_k b_k(x_i) + \varepsilon_i$

→ poly regression: $b_j(x) = x^j$

→ step function: $b_j(x) = I(c_j \le x_i < c_{j+1})$

3, regression splines

a, piece-wise low-degree polynomials

$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \beta_3 x_i^3 + \varepsilon_i$ where

$\beta_0, \beta_1, \beta_2$ and β_3 differ in different ranges

points where coeffs change are called <u>knots</u>

e.g. we fit a new poly b/w every knot

b, we can impose continuity at the knots

- continuity of fn
- continuity of derivatives → smoothes out the fn
→ usually up until $(d-1)$th derivative

degrees of freedom: (degree +1) * (knots+1) − (constraint on derivatives) * knots
(degree +1): intercept; (constraint on derivatives): (degree − 1)

eg. cubic spline w/ K knots has $4K + 4 - 3K = K+4$ degrees of freedom.

basis representation for splines:

Cubic spline has $K+4$ degrees of freedom, use $b_{-1} \dots b_{K+3}$

(b_0 replaces β_0)

$$h(x, \xi) = (x-\xi)_+^3 = \begin{cases} (x-\xi)^3 & \text{if } x > \xi \\ 0 & \text{o/w} \end{cases}$$

this is continuous up until the 3rd derivative, where it will be discontinuous

so we fix $X, X^2, X^3, h(X, \xi_1), h(X, \xi_2), \dots h(X, \xi_K)$

where ξ_i are the knots

$\rightarrow K+4$ degrees of freedom

natural spline: we require linearity at the two outer edges

as the variance there is usually big

(this is counted as 2 knots at the bdries w/ another constraint to enforce linearity)

Choosing the location of knots

usually at uniform quantiles of the data

. choose degrees of freedom and knot number w/ CV

→ Splines are usually more stable than poly regression
better at edges if natural
don't need so high degree polys
can be more flexible where data changes rapidly
and less flexible where data more stable
by placement of knots

4) Smoothing splines

→ we want to minimize RSS but minimize overfitting by having a smooth function.

Cost fn: $\sum (y_i - g(x_i))^2 + \lambda \int g''(t)^2 \, dt$

$\lambda \geq 0$ tuning param

RSS loss

penalty

penalizing the change in the slope

if $\lambda = 0$, exactly fitting to the points, controls effective degrees of freedom
if $\lambda \to \infty$, straight line

the function that minimizes this cost fn is:

- piece-wise cubic polys w/ knots at $x_1, \dots, x_n$ w/ continuous derivatives
- linear at the outer regions

$\Rightarrow$ THAT'S A NATURAL CUBIC SPLINE

but not the same one we got w/ the method before as we have λ controlling the shrinkage

- LOOCV is very easily computed for smoothing splines (or any basis fn regression)

5) Local regression

- only use nearby datapoints

Algo: for $x = x_0$:

1) gather the fraction $s = \frac{k}{n}$ points who are closest to x_0
2) assign $K_{i0} = K(x_i, x_0)$ to each point of the method so closest has largest weight, furthest has smallest weight
3) fit minimizing $\sum K_{i0}(y_i - \beta_0 - \beta_1 x_i)^2$

4. Fitted value is $\hat{f}(x_0) = \hat{\beta}_0 + \hat{\beta}_1 x_0$

→ we need to do this for all datapoints

Choices: s, $k(x_i, x_0)$

↓

larger s – less wiggly $\quad 0 < s \leq 1$

smaller s – more wiggle

- for multiple predictors, for example can fit local in one variable, global in other (e.g. local in time, global in space)

  ⇒ varying coefficient model

- can use 2d / 3d neighborhoods for special data
- and perform badly above: lots of points are far away

5. Generalized additive models (GAM)

$$y_i = \beta_0 + \sum_{j=1}^{p} f_j(x_{ij}) + \varepsilon_i$$

where $f_j(x)$ is a smooth nonlinear fn

Each predictor has a separate f_j

→ Building blocks

e.g. can fit one predictor w/ splines, the other w/ poly regression, or local regression

one big regression fit w/ least squares

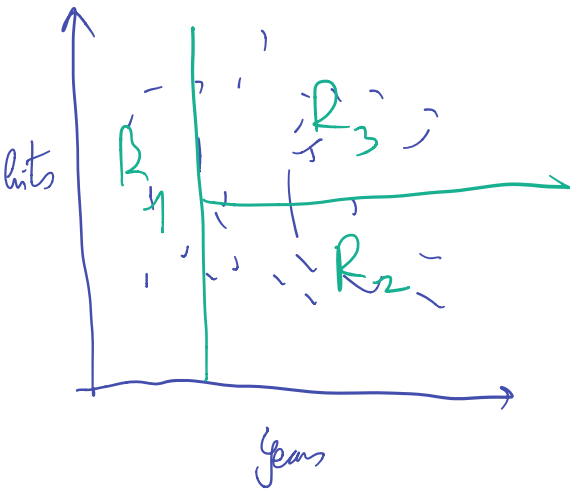
(unless smoothing splines, then w/ backfitting method but difference b/w smoothing and natural splines is small)

Pros:

- can use non-linear f_j for each X_j so model non-linear relationship w/o trying out many possible transformations
- more accurate predictions possible b/c non-linearity
- can see response to one X_j by holding all other variables fix → useful for inference

- neither of f_j can be summarized by degrees of freedom
- Can add interaction terms manually eg. $f_{jk}(X_j, X_k)$ these can be fit w/ 2d smoothers

Chapter 8: Tree-based Methods

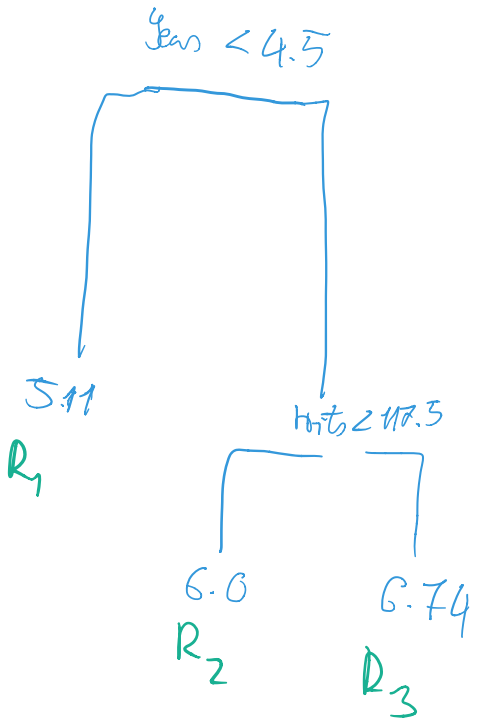


want to predict salary given years and hits (log-transformed already)

$R_1 = \{X \mid \text{Years} < 4.5\}$

$R_2 = \{X \mid \text{Years} \geq 4.5, \text{Hits} < 117.5\}$

$R_3 = \{X \mid \text{Years} \geq 4.5, \text{Hits} \geq 117.5\}$



R_x : terminal nodes or leaves

internal nods

branches

- divide space into regions
- predict the mean of observations in that region

How to find the regions?

→ want to minimize $\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2$

use greedy approach "binary tree splitting" top-down

1. Choose j and s such that region $\underbrace{\{X | X_j < s\}}_{R_1(j,s)}$ and $\underbrace{\{X | X_j \geq s\}}_{R_2(j,s)}$ minimize RSS

   e.g $\sum_{i: x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i: x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2$

2. Repeat the process by splitting one of the previously splitted regions

   Stop at a stopping criterion (regions, # of observations in a region etc.)

3. Predict using mean in that region

Tree-pruning

Above is likely to overfit

⇒ want smaller trees, we will cut some branches after creating the tree

Algo:

1) Recursive binary splitting until each terminal node has fewer than some min. number of observations
2) Apply cost complexity pruning w/ tuning parameter α
3) Choose α w/ CV

Complexity pruning:

For each α $\exists$ subtree $T \subset T_0$ s.t.

$$\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T| \quad \text{is as small as possible}$$

($|T|$ = # of terminal nodes)

$\alpha = 0 \Rightarrow T = T_0$ original tree

$\alpha \to \infty \Rightarrow$ fewer terminal nodes (until to leaf)

Nice thing : as we increase α, branches get cut in a predictable and nested way

Can make a list of branches cut as $\alpha \to \infty$
subtrees left

can use CV to find right α

Classification trees

same as regression tree but for qualitative variables

- instead of mean, we use most common class for prediction
- instead of RSS, misclassification rate

could be used but IS NOT sensitive enough for pruning

We use the Gini index instead :

$\hat{p}_{mk}$: portion of training examples in the mth class that are from the kth class

$G = \sum_{k=1}^{K} \hat{p}_{mk} (1 - \hat{p}_{mk})$, a measure of total variance across classes

small if all $\hat{p}_{mk}$ are close to 0 or 1

measure of purity – node contains observations mostly from a single class if small

Entropy:

$$D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$$

$D \approx 0$ if $\hat{p}_{mk}$'s are near 0 or 1

→ same as Gini index

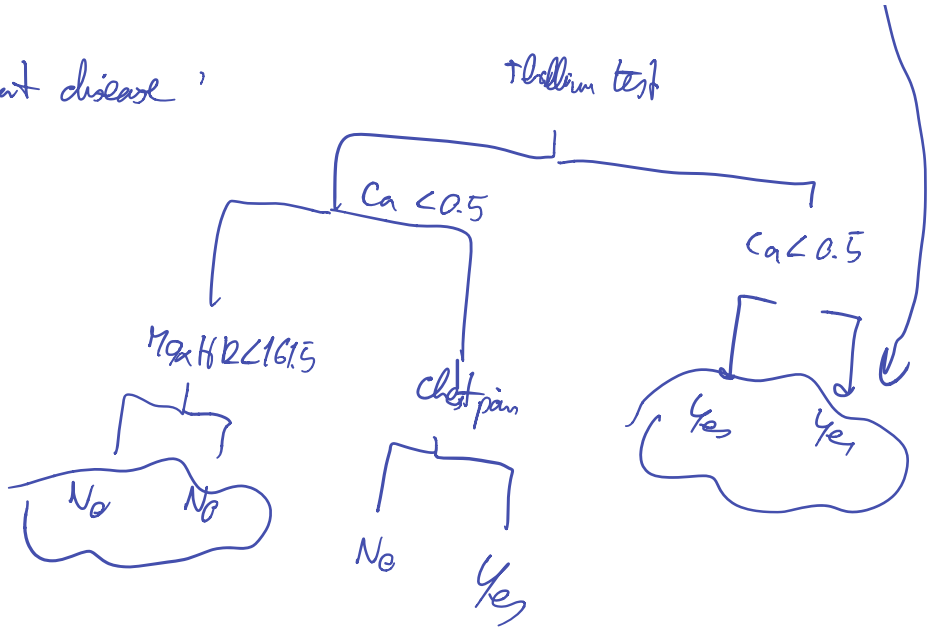
Gini index and entropy are more sensitive to node purity than misclassification error rate

Can use either for pruning

Weird thing: After pruning, we can end up w/ nodes that have the same answer at the end nodes.

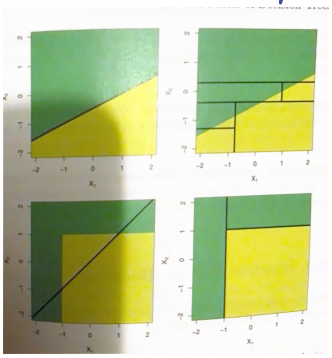
This is b/c it increases node purity (doesn't change misclassification error)

e.g. one node is 100% Yes, the other only 7/11 yes → increases overall Gini score



Tree vs. Linear Models

- depends if data has linear structure or complex relationships
- trees are easier to interpret



true boundary is linear on top
true boundary is regional on bottom row

Advantages:

- easy to explain
- mirror human decision-making
- can be displayed graphically
- can easily handle qualitative predictors

Disadvantages

- less accurate

- very non-robust, small changes result in a different tree

Bagging

Trees have high variance and are not stable

the train - test split heavily influences the outcome

bagging = bootstrap aggregation

→ used to reduce variance by bootstrapping training sets and taking the mean of the predictions (or majority for classification)

we don't prune these trees → high variance
low bias

Error estimation

On average, each bagged tree uses 2/3rd of the total observations

↳ "out of bag"

for each observation we take those trees for which it was out of bag (~1/3 of trees) and average their predictions to get a final

prediction. We can use the test error rate from the errors of these final predictions.

(Cheaper than CV)

Bagging makes us lose interpretability at the expense of accuracy.

How to find important variables:

For regression tree, RSS: total amount the RSS has decreased due to splits over a given predictor

if large, important predictor

For classification tree, Gini: how the Gini decreased due to splits over a given predictor

Random forests:

- decorrelates trees

  - when splitting your tree, you can only choose one of m randomly selected variables at each split
  - usually $m \simeq \sqrt{p}$

  → you won't always pick the same strongest variable, so your trees will look more different from each other and be less correlated

  - average of trees less variable because of the less correlation

small m is good when large number of predictors

Boosting

growing trees sequentially, using information from previous trees

Algo:

1) set $\hat{f}(x) = 0$ and $r_i = y_i$ $\forall i$ in training set

2) for $b = 1, 2, \dots, B$:

a) fit tree $\hat{f}^{(b)}$ w/ d splits to training data (X, r)

b) update $\hat{f}$ by adding a shrunken version of the new tree

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^{(b)}(x)$$

c) update the residuals

$$r_i \leftarrow r_i - \lambda \hat{f}^{(b)}(x_i)$$

3) output boosted model

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^{(b)}(x)$$

→ learns slowly, we fit the trees to the residuals of the current model instead of the outcome.

we then add this decision tree into the fitted fn and update

the residuals.

Params: B – number of trees

if too large, can overfit (unlike bagging and rf)

λ – shrinkage parameter

usually 0.01 or 0.001

if very small, we need a large B

d – number of splits

$d=1$ can work well (stump trees) → additive model

controls the interaction – d splits can involve at most d variables

Exercises CH7

Show $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)_+^3$

is a cubic regression spline, (ie $x, x^2, x^3, (x-\xi)_+^3$ basis used)

a) Find cubic poly $f_1(x)$ s.t. $f(x) = f_1(x) \quad \forall\, x \le \xi$,

$(f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3)$

$f_1(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3$

b) $f_2(x)$ s.t. $f(x) = f_2(x) \quad \forall\, x > \xi$:

$f_2(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)^3$ $\quad \beta_4 x^3 - 3x^2\xi + 3x\xi^2 - \xi^3$

$= (\beta_0 - \beta_4 \xi^3) + (\beta_1 + \beta_4 3\xi^2) x + (\beta_2 - \beta_4 3\xi) x^2 + (\beta_3 + \beta_4) x^3$

c) $f_1(\xi) = f_2(\xi)$: $f(x)$ is cont. $\quad \beta_4 (x-\xi)^3 = 0$ at $x = \xi$

d) $f_1'(\xi) = f_2'(\xi)$ $\quad f'(x)$ is cont. $\quad 3\beta_4 (x-\xi)^2 = 0$ at $x = \xi$

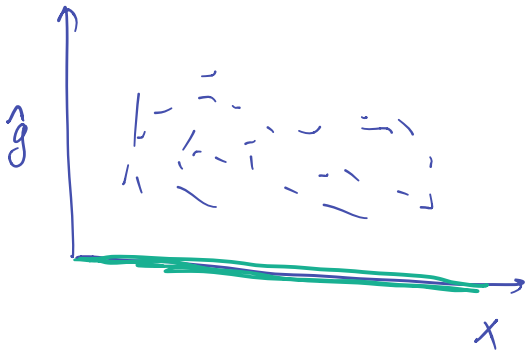
e) $f_1''(\xi) = f_2''(\xi)$ $\quad f''(x)$ is cont. $\quad 6\beta_4 (x-\xi) = 0$ at $x = \xi$

2)

$$\hat{g} = \arg\min_{g} \left(\sum (y_i - g(x_i))^2 + \lambda \int [g^m(x)]^2 dx \right)$$

→ with derivative instead of second in smoothness penalty

a)

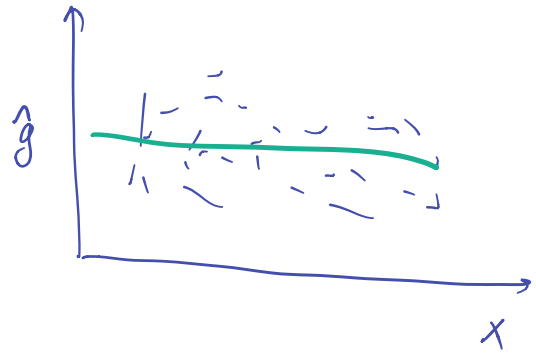


$\lambda = \infty, \; m = 0$

$\Rightarrow \hat{g} = 0$

minimizes $\int g^2 dx$

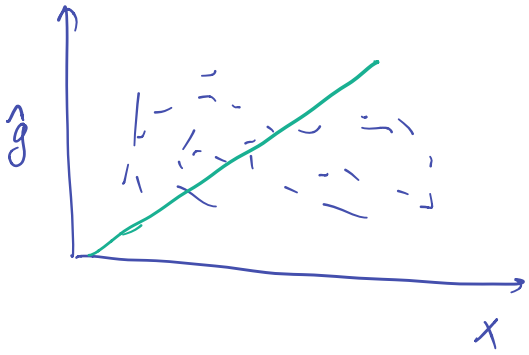
b)



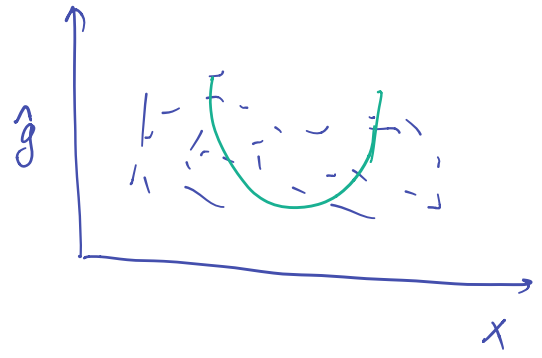
$\lambda = \infty, \; m = 1$

minimizes $\int (g')^2 dx$

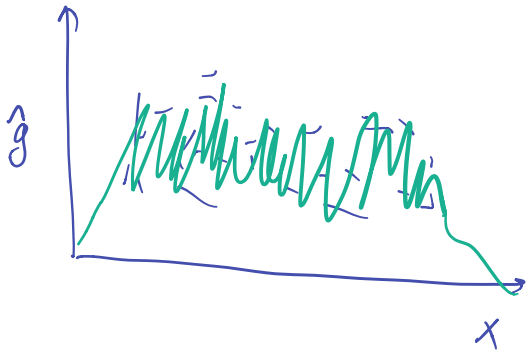
$g' = 0 \rightarrow g(x) = c$



$\lambda = \infty, \; m = 2$



$\lambda = \infty, \quad n = 3$



$\lambda = 0$

3) $e_1(x) = x$

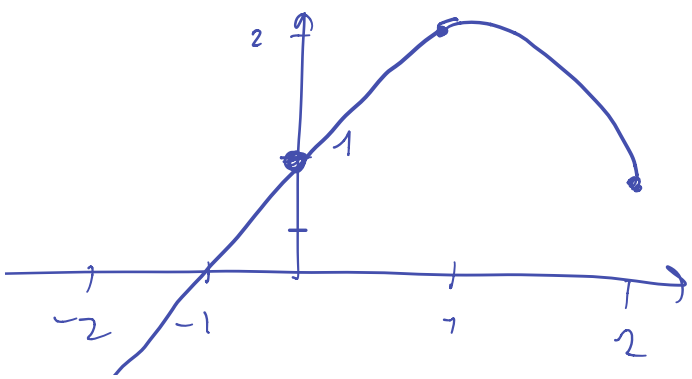
$e_2(x) = (x - \bar{x})^2 \, I(x \geq 1)$

$y = \beta_0 + \beta_1 e_1(x) + \beta_2 e_2(x) + \varepsilon$

$\hat{\beta}_0 = 1$

$\hat{\beta}_1 = 1$

$\hat{\beta}_2 = -2$



$y = 1 + x - 2(x-1)^2$ se $x \geq 1$

$1 + x$ se $x \leq 1$

4) $b_1(x) = I(0 \leq x \leq 2) - (x-1) I(1 \leq x \leq 2)$

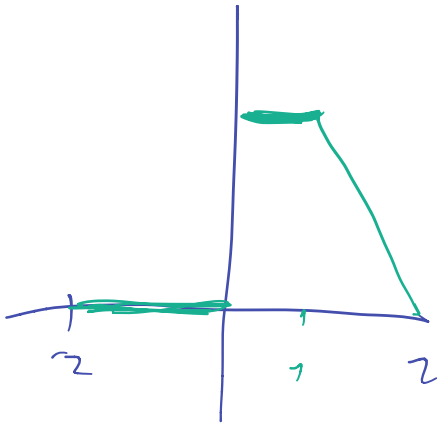
$b_2(x) = (x-3) I(3 \leq x \leq 4) + I(4 < x \leq 5)$

$y = \beta_0 + \beta_1 b_1(x) + \beta_2 b_2(x) + \varepsilon$

$\hat{\beta}_0 = 1$

$\hat{\beta}_1 = 1$

$\hat{\beta}_2 = 3$



$y = 1$ se $x < 0$

$y = 1 + 1 = 2$ se $0 \leq x < 1$

$y = 2 - (x-1)$ se $1 \leq x < 2$

$= 3 - x$

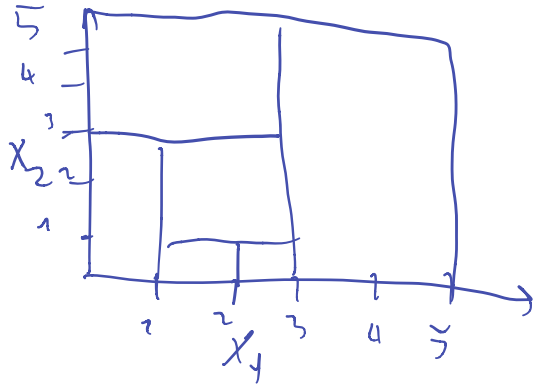
5) a) $\hat{g}_2$ have small RSS on training set
→ higher order poly

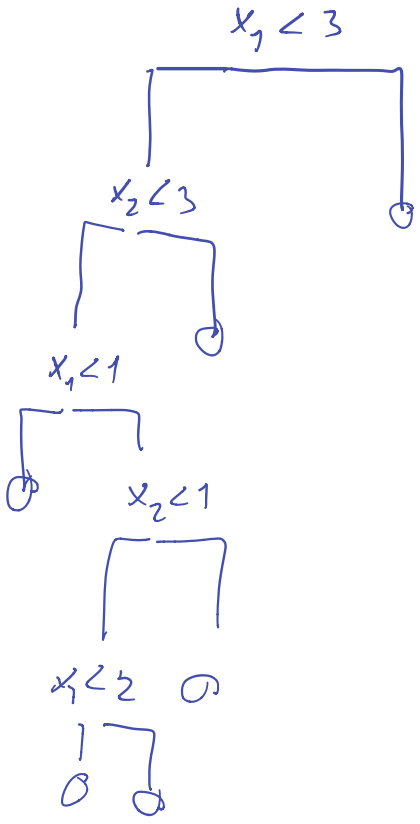
c) either, can't possibly $\hat{g}_1$ as overfitted on

c) $\hat{g}_1 = \hat{g}_2$ if $\lambda = 0$

Chapter 8 exercises

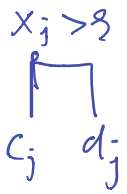
1)





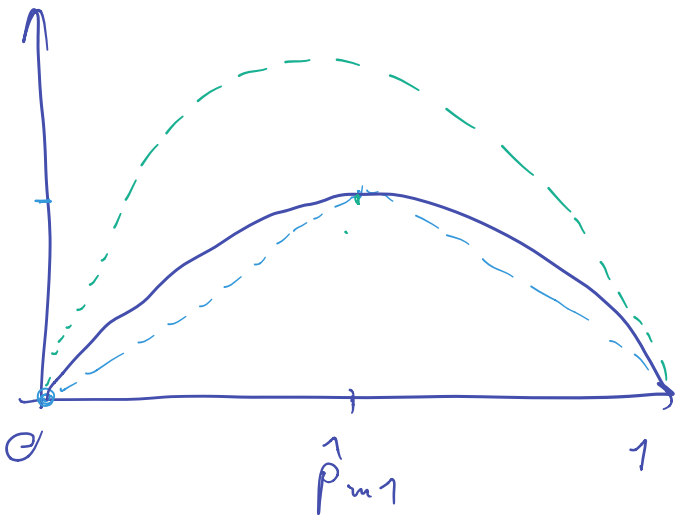
2) Stumps is an additive model

$$\hat{f}(x) = \sum \lambda \hat{f}^{(e)}(x)$$

Stump: 

where $\hat{f}^{(e)}(x) = c_j I(x_j < s_j) + d_j I(x_j \geq s_j)$

So $\lambda \hat{f}^{(e)}(x) = f(x_j)$ for some j, it only depends on one predictor

3) 

Gini
entropy
classification error

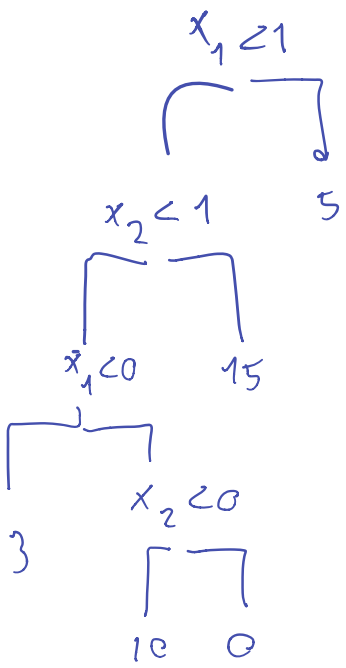
two classes
$\hat{p}_{m2} = 1 - \hat{p}_{m1}$

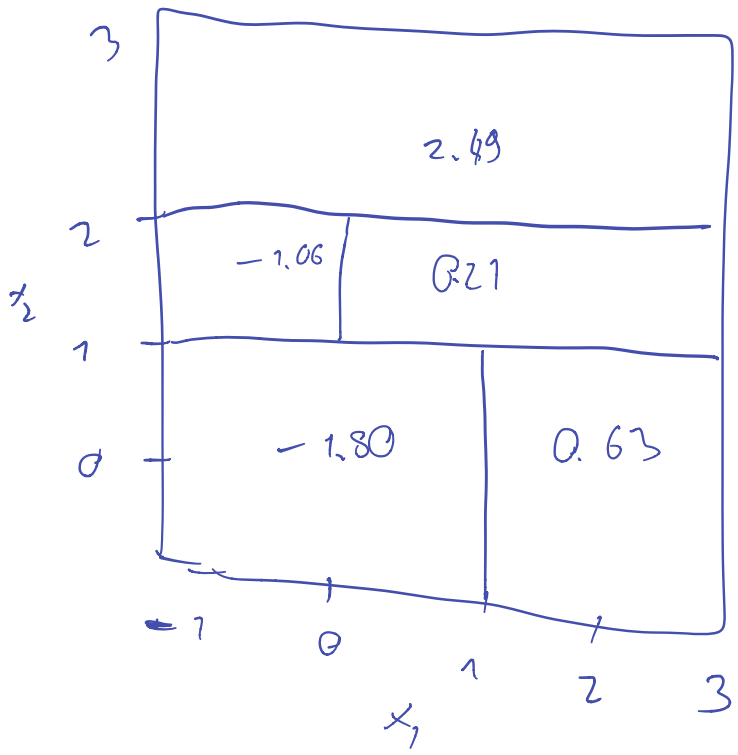
Gini: $2(\hat{p}_{m1}(1-\hat{p}_{m1}))$

entropy: $-\hat{p}_{m1}\log\hat{p}_{m1} - (1-\hat{p}_{m2})\log(1-\hat{p}_{m1})$

Classification error: $1 - \max(\hat{p}_{m1}, \hat{p}_{m2})$

4,





5, Majority vote: red

Avg: 0.43 ⇒ Green

6,