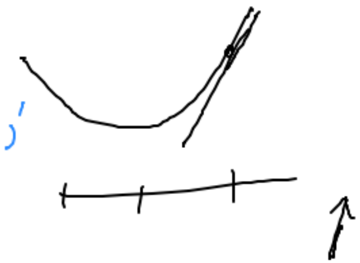
Ridge regression:

suppose: $x_{ij} > 0$
we predict too small value
i.e. $\vartheta_0 + \sum \vartheta_j x_{ij} < y_i$



minimise for some given $\lambda \geq 0$

$$L = \sum_{i=1}^{N} \Big(\underbrace{y_i - \vartheta_0 - \sum_{j=1}^{n} \vartheta_j x_{ij}}_{\text{suppose } > 0} \Big)^2 + \lambda \sum_{j=1}^{n} \vartheta_j^2$$

to find the minimiser, i.e., $\vartheta_0, \ldots, \vartheta_n$ for which L is the smallest, one can (also) use gradient descent

$$\frac{\partial L}{\partial \vartheta_k} = - \sum_{i=1}^{N} 2 \Big(\underbrace{y_i - \vartheta_0 - \sum_{j=1}^{n} \vartheta_j x_{ij}}_{> 0} \Big) \underbrace{x_{ik}}_{> 0} + 2 \lambda \vartheta_k \qquad k = 1, \ldots, n$$

-||-

$$\tilde{\vartheta}_k = \vartheta_k - c \cdot \frac{\partial L}{\partial \vartheta_k}$$

↑ adjusted ϑ_k

↑ learning constant (c)

← should be repeated many times

In the neural networks world, this corresponds to a batch size = size of the training data

- One can also use regularisation in neural networks

$$L = \sum_j (t_j - y_j)^2 + \lambda \sum w^2$$

L: loss function

t_j: target value

y_j: actual value

w: weights different than those encoding bias

$$2\lambda w_{ij}^{(k)}$$

Decision trees

1) Divide the predictor space - i.e, the set of possible values $(x_1, \dots, x_p)$ into J ~~distinct~~ disjoint regions $R_1, \dots, R_J$

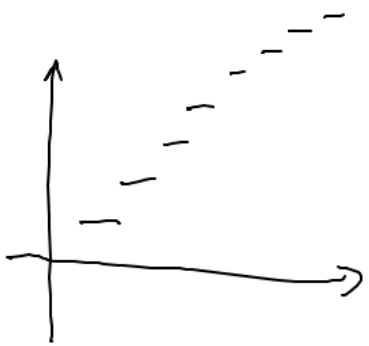
2) Prediction: for every observation that falls into R_j, we make the same prediction:

regression: mean/median of the training obs. that fell into R_j

classification: category that occurs most often among training obs. that fell into R_j (majority vote)

Model function:

$$f(x) = \sum_{j=1}^{J} c_j \mathbb{1}_{R_j}(x)$$



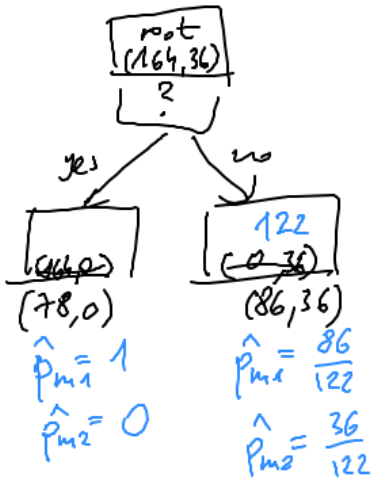
For growing the tree:

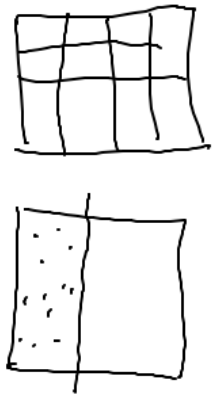
One uses a greedy algorithm:

- start from the root $\rightsquigarrow$ regions $= \{\text{whole space}\}$
- when we have a tree, with regions $\{R_1, \dots, R_k\}$, for a region $R = R_\ell$ consider all regions of the form:

$$R^{(1)}(j,s) = R_\ell \cap \{(x_1, \dots, x_p) : x_j < s\}$$

$$R^{(2)}(j,s) = R_\ell \cap \{(\dots) : x_j \geq s\}$$





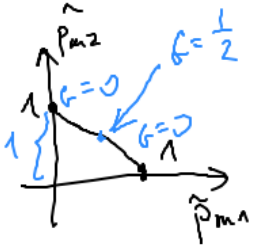
and choose one that minimises some error

- regression trees: $\sum_{i:\, x_i \in R^{(1)}(j,s)} (y_i - \hat{y}_{R^{(1)}})^2 + \sum_{i:\, x_i \in R^{(2)}(j,s)} (y_i - \hat{y}_{R^{(2)}})^2$
- classification: $\hat{p}_{mk} = \frac{\#\text{ samples in } R_m \text{ of class } k}{\#\text{ samples in } R_m}$ $\qquad \hat{p}_{m1} + \dots + \hat{p}_{mK} = 1$

minimize: $E = 1 - \max_k \hat{p}_{mk}$ — error of classification — not a good one

$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} (\hat{p}_{mk})^2$ — Gini

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



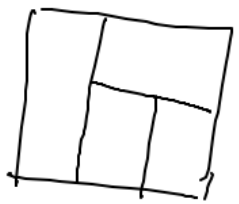
for one node / weighted average should be minimized for all leaves

We stop when all leaves have a number of samples < threshold.

Second step: pruning

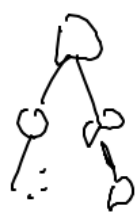
cost of complexity: $\alpha \geq 0$

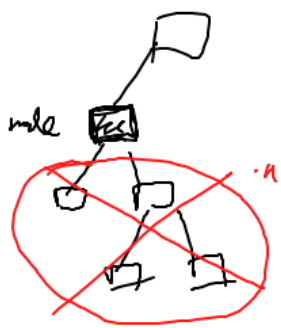
minimise $\underbrace{\sum_{m=1}^{|T|} \sum_{i: x_i \in R_m} (y_i - \hat{y}_{R_m})^2}_{\text{prediction error (on training data)}} + \alpha \underbrace{|T|}_{\text{the number of leaves}}$ ← regression



- for each node, we check for which α the cost of complexity would decrease by pruning the subtree of that node
- we prune the subtree of the node with the smallest α ⇝ obtain a new tree
- repeat for the new tree

sequence of trees!





- number of leaves will be smaller by 2
- the error of prediction will increase by e

the cost will ~~change~~ increase by

$e - 2\alpha \leq 0 \quad \alpha \geq \frac{e}{2}$