STK-IN4300
Statistical Learning Methods in Data Science

Riccardo De Bin

debin@math.uio.no

Outline of the lecture

- Generalized Additive Models
  - Definition
  - Fitting algorithm
- Tree-based Methods
  - Background
  - How to grow a regression tree
- Bagging
  - Bootstrap aggregation
  - Bootstrap trees

Generalized Additive Models: introduction

From the previous lectures:

- linear regression models are easy and effective models;
- often the effect of a predictor on the response is not linear;

$$\downarrow$$

local polynomials and splines.

Generalized Additive Models:

- flexible statistical methods to identify and characterize nonlinear regression effects;
- larger class than the generalized linear models.

Generalized Additive Models: additive models

Consider the usual framework:

- $X_1, \ldots, X_p$ are the predictors;
- Y is the response variable;
- $f_1(\cdot), \ldots, f_p(\cdot)$ are unspecified smooth functions.

Then, an **additive model** has the form

$$E[Y|X_1, \ldots, X_p] = \alpha + f_1(X_1) + \cdots + f_p(X_p).$$

Generalized Additive Models: more generally

As linear models are extended to generalized linear models, we can generalize the additive models to the generalized additive models,

$$g(\mu(X_1, \ldots, X_p)) = \alpha + f_1(X_1) + \cdots + f_p(X_p),$$

where:

- $\mu(X_1, \ldots, X_p) = E[Y|X_1, \ldots, X_p]$;
- $g(\mu(X_1, \ldots, X_p))$ is the link function;
- classical examples:
  - $g(\mu) = \mu \leftrightarrow$ identity link $\rightarrow$ Gaussian models;
  - $g(\mu) = \log(\mu/(1-\mu)) \leftrightarrow$ logit link $\rightarrow$ Binomial models;
  - $g(\mu) = \Phi^{-1}(\mu) \leftrightarrow$ probit link $\rightarrow$ Binomial models;
  - $g(\mu) = \log(\mu) \leftrightarrow$ logarithmic link $\rightarrow$ Poisson models;
  - ...

Generalized Additive Models: semiparametric models

Generalized additive models are very flexible:

- not all functions $f_j(\cdot)$ must be nonlinear;

$$g(\mu) = X^T\beta + f(Z)$$

  in which case we talk about **semiparametric models**.

- nonlinear effect can be combined with qualitative inputs,

$$g(\mu) = f(X) + g_k(Z) = f(X) + g(V, Z)$$

  where k indexes the level of a qualitative variable V.

Fitting algorithm: difference with splines

When implementing splines:

- each function is modelled by a basis expansion;
- the resulting model can be fitted with least squares.

Here the approach is different:

- each function is modelled with a smoother (smoothing splines, kernel smoothers, ...)
- all p functions are simultaneously fitted via an algorithm.

Fitting algorithm: ingredients

Consider an additive model

$$Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \epsilon.$$

We can define a loss function,

$$\sum_{i=1}^{N} \left(y_i - \alpha - \sum_{j=1}^{p} f_j(x_{ij}) \right)^2 + \sum_{j=1}^{p} \lambda_j \int \{f_j''(t_j)\}^2 dt_j$$

- λ_j are tuning parameters;
- the minimizer is an additive cubic spline model,
  - each $f_j(X_j)$ is a cubic spline with knots at the (unique) x_{ij}'s.

Fitting algorithm: constraints

The parameter α is in general not identifiable:

- same result if adding a constant to each $f_j(X_j)$ and subtracting it from α;
- by convention, $\sum_{j=1}^{p} f_j(X_j) = 0$:
  - the functions average 0 over the data;
  - α is therefore identifiable;
  - in particular, $\hat{\alpha} = \bar{y}$.

If this is true and the matrix of inputs X has full rank:

- the loss function is convex;
- the minimizer is unique;
- simple procedure to find the solution $\rightarrow$ backfitting algorithm.

Fitting algorithm: backfitting algorithm

The **backfitting algorithm**:

1. Initialization: $\hat{\alpha} = N^{-1} \sum_{i=1}^{N} y_i$ and $\hat{f}_j \equiv 0 \ \forall j$
2. In cycle, $j = 1, \ldots, p, 1, \ldots, p, \ldots$

$$\hat{f}_j \leftarrow \mathcal{S}_j \left[\{ y_i - \hat{\alpha} - \sum_{k \neq j} \hat{f}_k(x_{ik}) \}_1^N \right]$$

$$\hat{f}_j \leftarrow \hat{f}_j - \frac{1}{N} \sum_{i=1}^{N} \hat{f}_j(x_{ij})$$

until $\hat{f}_j$ changes less than a pre-specified threshold.

$\mathcal{S}_j$ is usually a cubic smoothing spline, but other smoothing operators can be used.

Fitting algorithm: remarks

Note:

- the smoother $\mathcal{S}$ can be (when applied only at the training points) represented by the $N \times N$ smoothing matrix S,
  - the degrees of freedom for the j-th terms are trace$(S_j) - 1$;
- for the generalized additive model, the loss function is the penalized negative log-likelihood;
- the backfitting algorithm fits all predictors,
  - not feasible when $p >> N$.

Example: logistic regression for email spam data

Consider the spam data (as in Exercise 3.17).

- binary response (`email`/`spam`),
  - logistic regress., $\log \frac{Pr(Y=1|X)}{Pr(Y=0|X)} = \alpha + f_1(X_1) + \cdots + f_p(X_p)$;
- 48 percentages of words in the email (e.g. `you`, `free`, ...);
- 6 percentages of specific characters (e.g. `ch;`, `ch$`, ...);
- average length sequences of capital letters (`CAPAVE`);
- length longest sequence of capital letters (`CAPMAX`);
- sum length sequences of capital letters (`CAPTOT`).

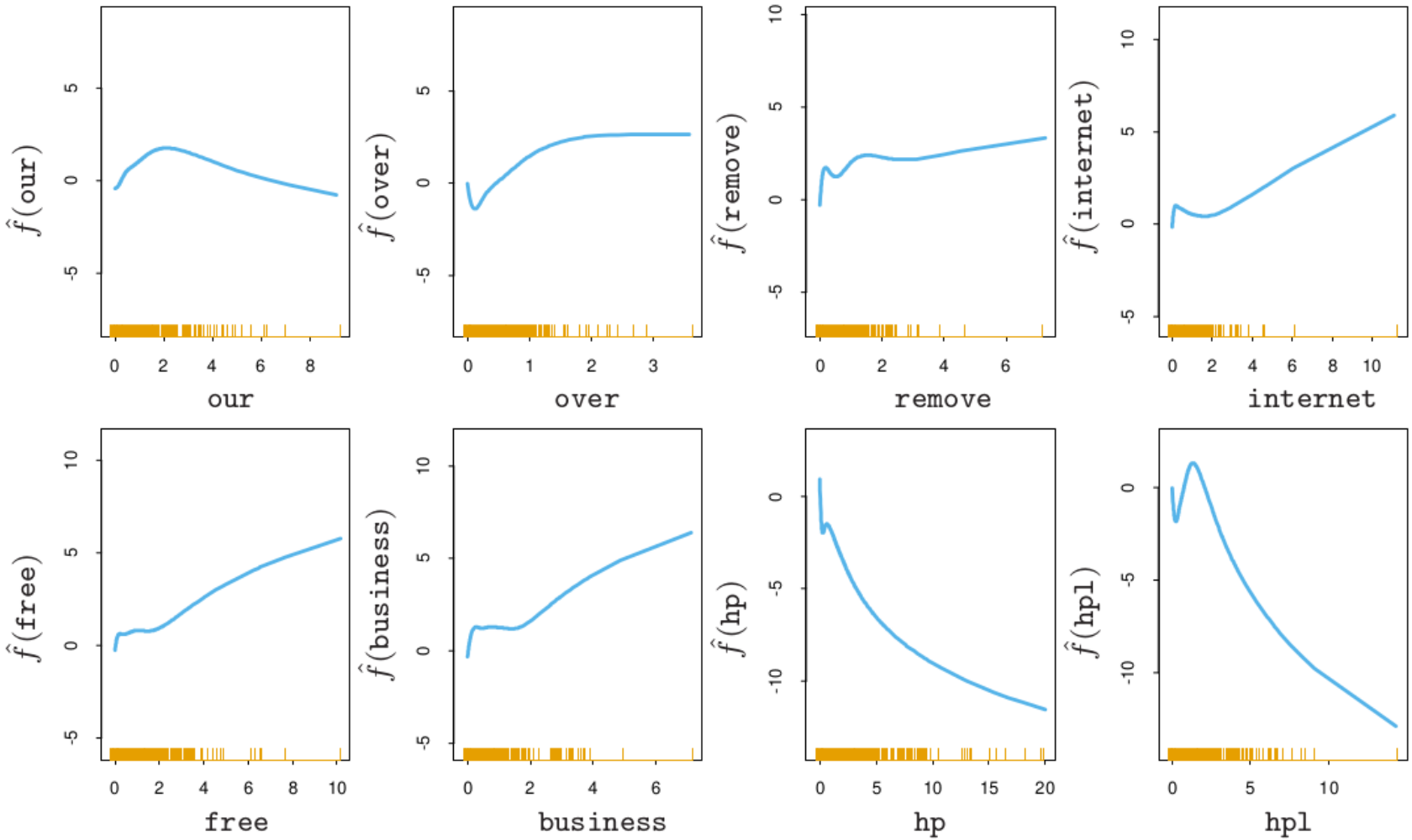
Sample size: 3065 training, 1536 test.

Choice of $f_j(\cdot)$: smoothing cubic splines with $df = 4$.

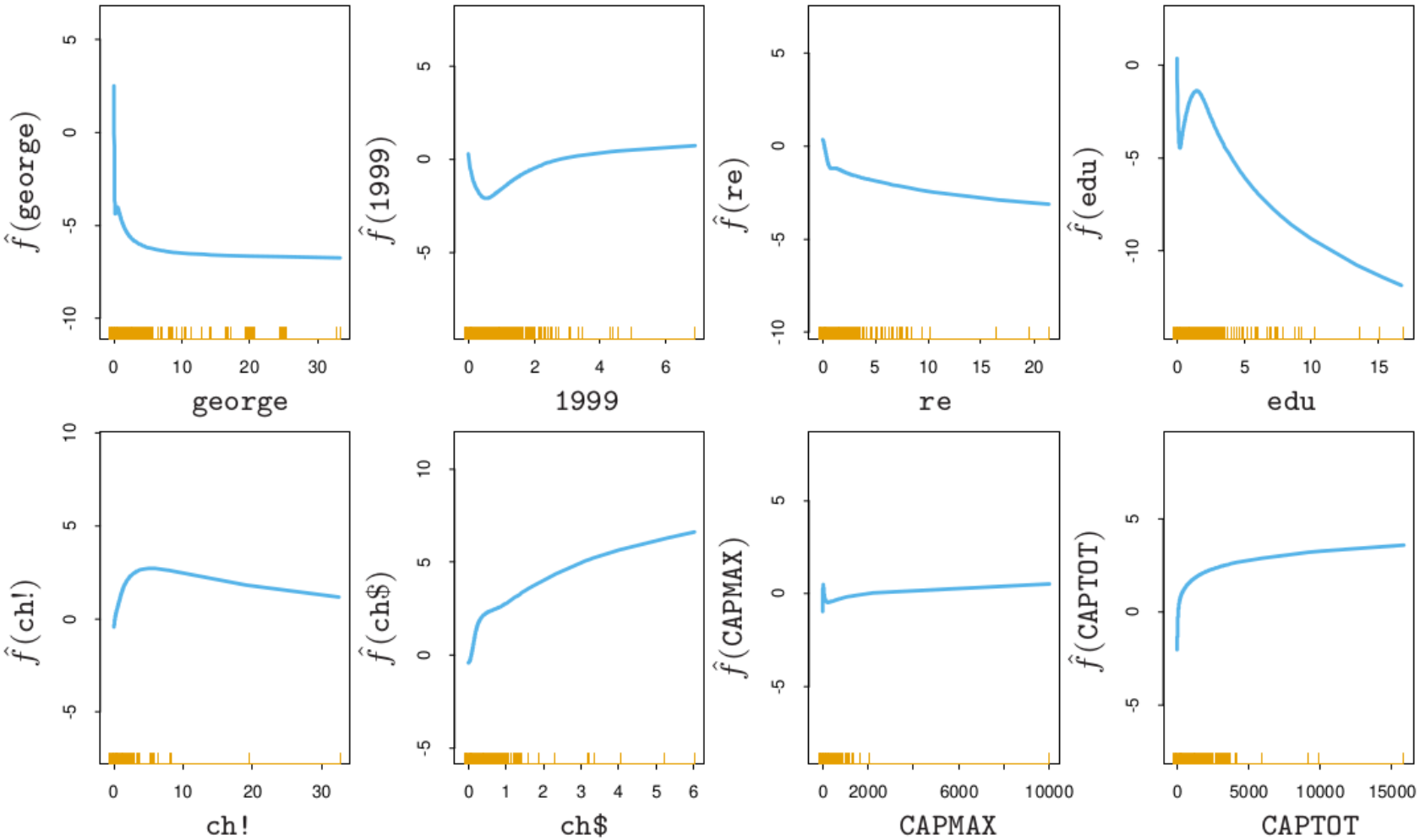
Example: logistic regression for email spam data

Name	Num.	df	Coefficient	Std. Error	Z Score	Nonlinear P-value
Positive effects						
our	5	3.9	0.566	0.114	4.970	0.052
over	6	3.9	0.244	0.195	1.249	0.004
remove	7	4.0	0.949	0.183	5.201	0.093
internet	8	4.0	0.524	0.176	2.974	0.028
free	16	3.9	0.507	0.127	4.010	0.065
business	17	3.8	0.779	0.186	4.179	0.194
hpl	26	3.8	0.045	0.250	0.181	0.002
ch!	52	4.0	0.674	0.128	5.283	0.164
ch$	53	3.9	1.419	0.280	5.062	0.354
CAPMAX	56	3.8	0.247	0.228	1.080	0.000
CAPTOT	57	4.0	0.755	0.165	4.566	0.063
Negative effects						
hp	25	3.9	−1.404	0.224	−6.262	0.140
george	27	3.7	−5.003	0.744	−6.722	0.045
1999	37	3.8	−0.672	0.191	−3.512	0.011
re	45	3.9	−0.620	0.133	−4.649	0.597
edu	46	4.0	−1.183	0.209	−5.647	0.000

Example: logistic regression for email spam data



Example: logistic regression for email spam data



Tree-based Methods: introduction

Consider a regression problem, Y the response, X the input matrix.

A tree is a recursive binary partition of the feature space:

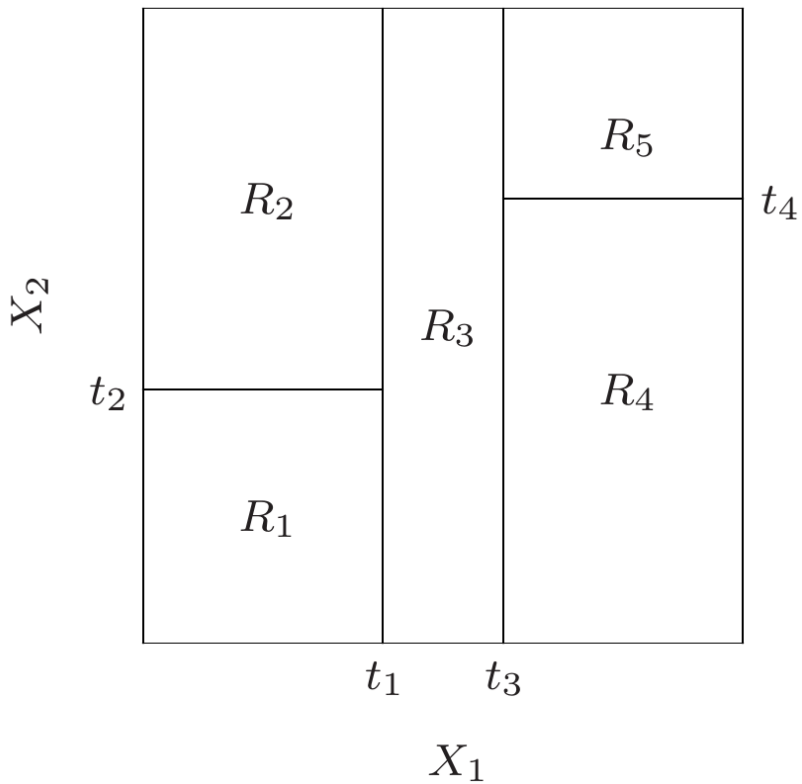
- each time, a region is divide into two or more regions;
  - until a stopping criterion applies;
- at the end, the input space is split in M regions R_m;
- a constant c_m is fitted to each R_m.

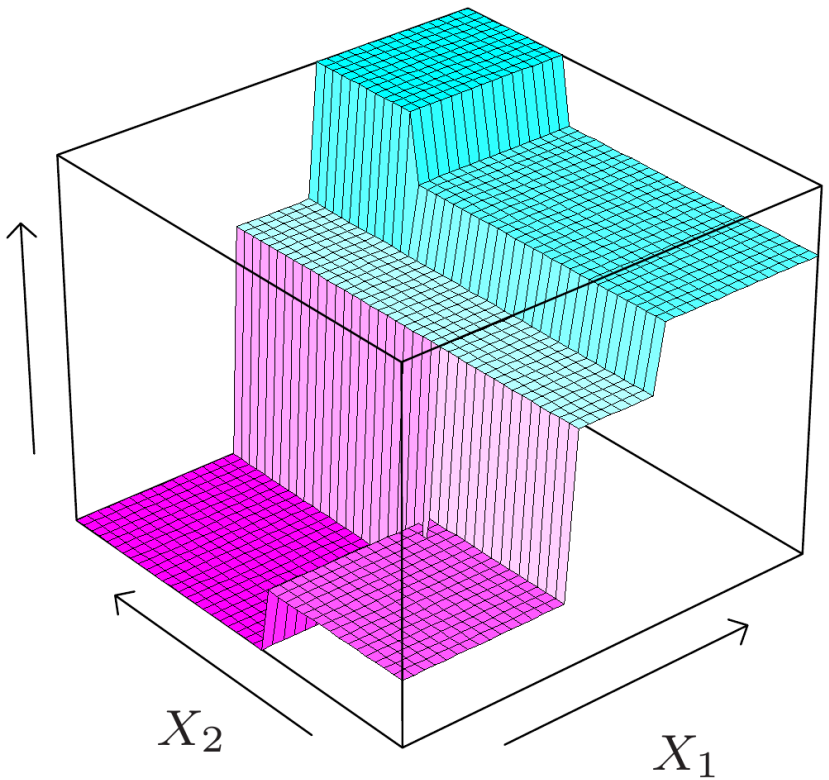
The final prediction is

$$\hat{f}(X) = \sum_{m=1}^{M} \hat{c}_m \mathbb{1}(X \in R_m),$$

where $\hat{c}_m$ is an estimate for the region R_m (e.g., ave$(y_i|x_i \in R_m)$).

Tree-based Methods: introduction





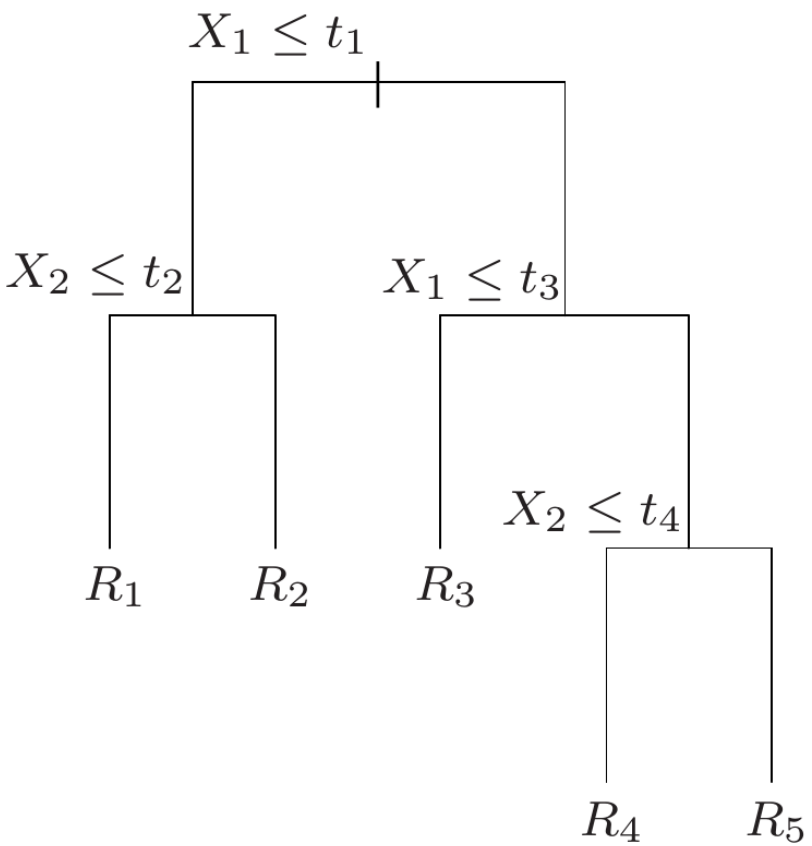
Tree-based Methods: introduction

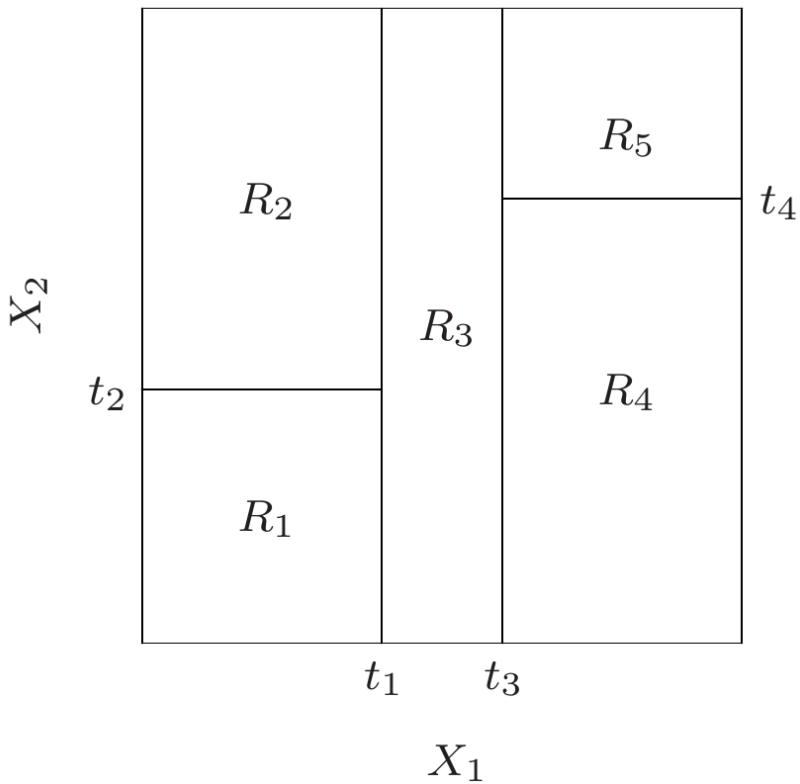
Note:

- the split can be represented as a junction of a tree;
- this representation works for $p > 2$;
- each observation is assigned to a branch at each junction;

- the model is easy to interpret.

Tree-based Methods: introduction





How to grow a regression tree: split

How to grow a regression tree:

- we need to automatically decide the splitting variables ...
- ... and the splitting points;
- we need to decide the shape (topology) of the tree.

Using a sum of squares criterion, $\sum_{i=1}^{N}(y_i - f(x_i))^2$,

- the best $\hat{c}_m = \text{ave}(y_i|x_i \in R_m)$;
- finding the best partition in terms of minimum sum of squares is generally computationally infeasible

$$\downarrow$$

go greedy

How to grow a regression tree: greedy algorithm

Starting with all data:

- for each X_j, find the best split point s
  - define the two half-hyperplanes,
    - $R_1(j,s) = \{X | X_j \leqslant s\}$;
    - $R_2(j,s) = \{X | X_j > s\}$;
  - the choice of s can be done really quickly;
- for each j and s, solve

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

- the inner minimization is solved by
  - $\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j,s))$;
  - $\hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j,s))$.
- the identification of the best (j,s) is feasible.

How to grow a regression tree: when to stop

The tree size:

- is a tuning parameter;
- it controls the model complexity;
- its optimal values should be chosen from the data.

Naive approach:

- split the tree nodes only if there is a sufficient decrease in the sum-of-squares (e.g., larger than a pre-specified threshold);
  - intuitive;
  - short-sighted (a split can be preparatory for a split below).

Preferred strategy:

- grow a large (pre-specified # of nodes) or complete tree T_0;
- prune it (remove branches) to find the best tree.

How to grow a regression tree: cost-complexity pruning

Consider a tree $T \subset T_0$ computed by pruning T_0 and define:

- R_m the region defined by the node m;
- $|T|$ the number of terminal nodes in T;
- N_m the number of observations in R_m, $N_m = \#\{x_i \in R_m\}$;
- $\hat{c}_m$ the estimate in R_m, $\hat{c}_m = N_m^{-1} \sum_{x_i \in R_m} y_i$;
- $Q_m(T)$ the loss in R_m, $Q_m(T) = N_m^{-1} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2$.

Then, the **cost complexity criterion** is

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m Q_m(T) + \alpha |T|.$$

How to grow a regression tree: cost-complexity pruning

The idea is to find the subtree $T_{\hat{\alpha}} \subset T_0$ which minimizes $C_\alpha(T)$:

- $\forall \alpha$, find the unique subtree T_α which minimizes $C_\alpha(T)$;
- through **weakest link pruning**:
  - successively collapse the internal node that produces the smallest increase in $\sum_{m=1}^{|T|} N_m Q_m(T)$;
  - until the single node tree;
  - find T_α within the sequence;
- find $\hat{\alpha}$ via cross-validation.

Here the tuning parameter α:

- governs the trade-off between tree size and goodness of fit;
- larger values of α correspond to smaller trees;
- $\alpha = 0 \rightarrow$ full tree.

Classification trees: definition

No major differences between regression and classification trees:

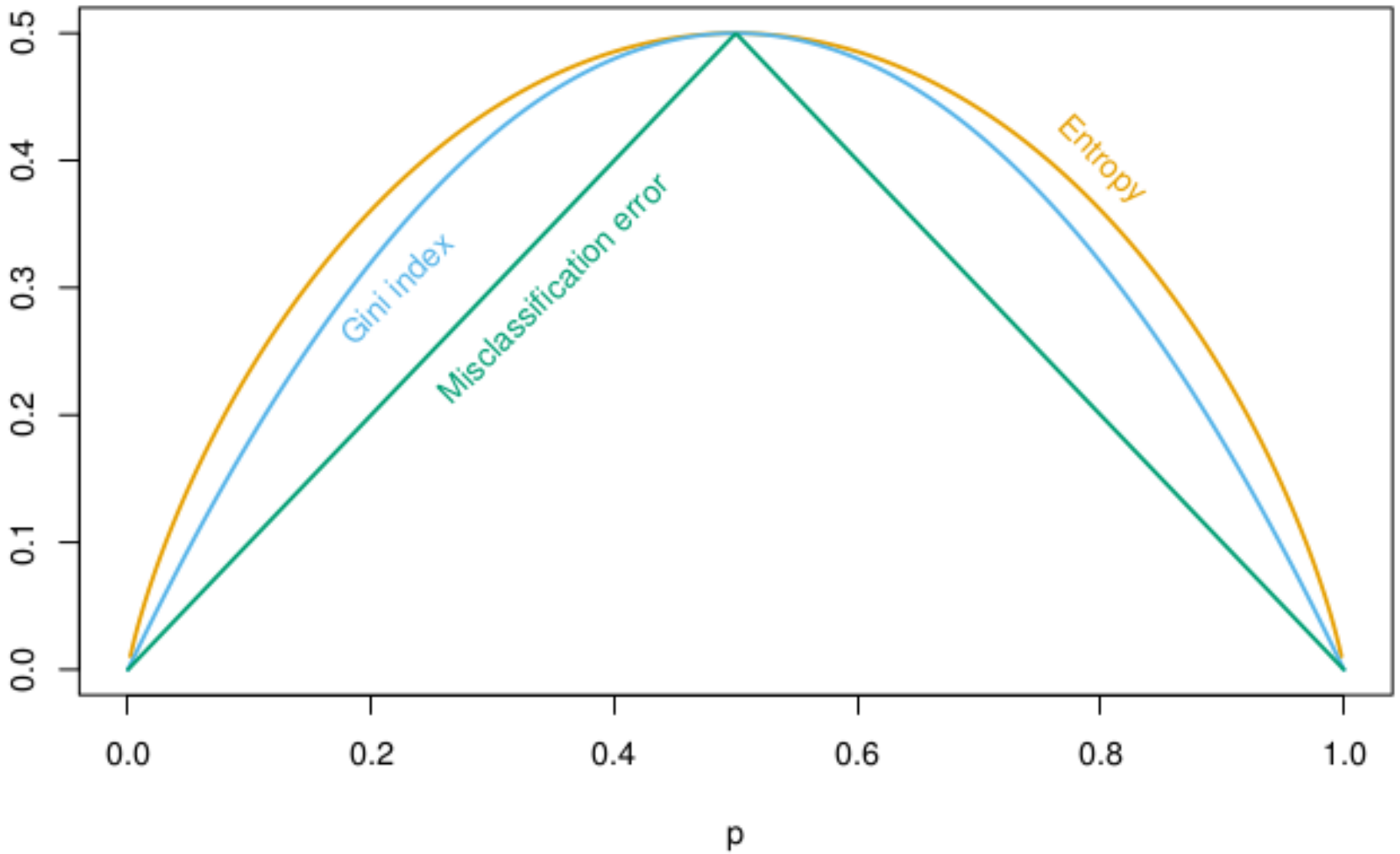
- define a class $k \in \{1, \ldots, K\}$ for each region,

$$k_m = \text{argmax}_k \, \hat{p}_{mk} = \text{argmax}_k \left\{ N_m^{-1} \sum_{x_i \in R_m} \mathbb{1}(y_i = k) \right\};$$

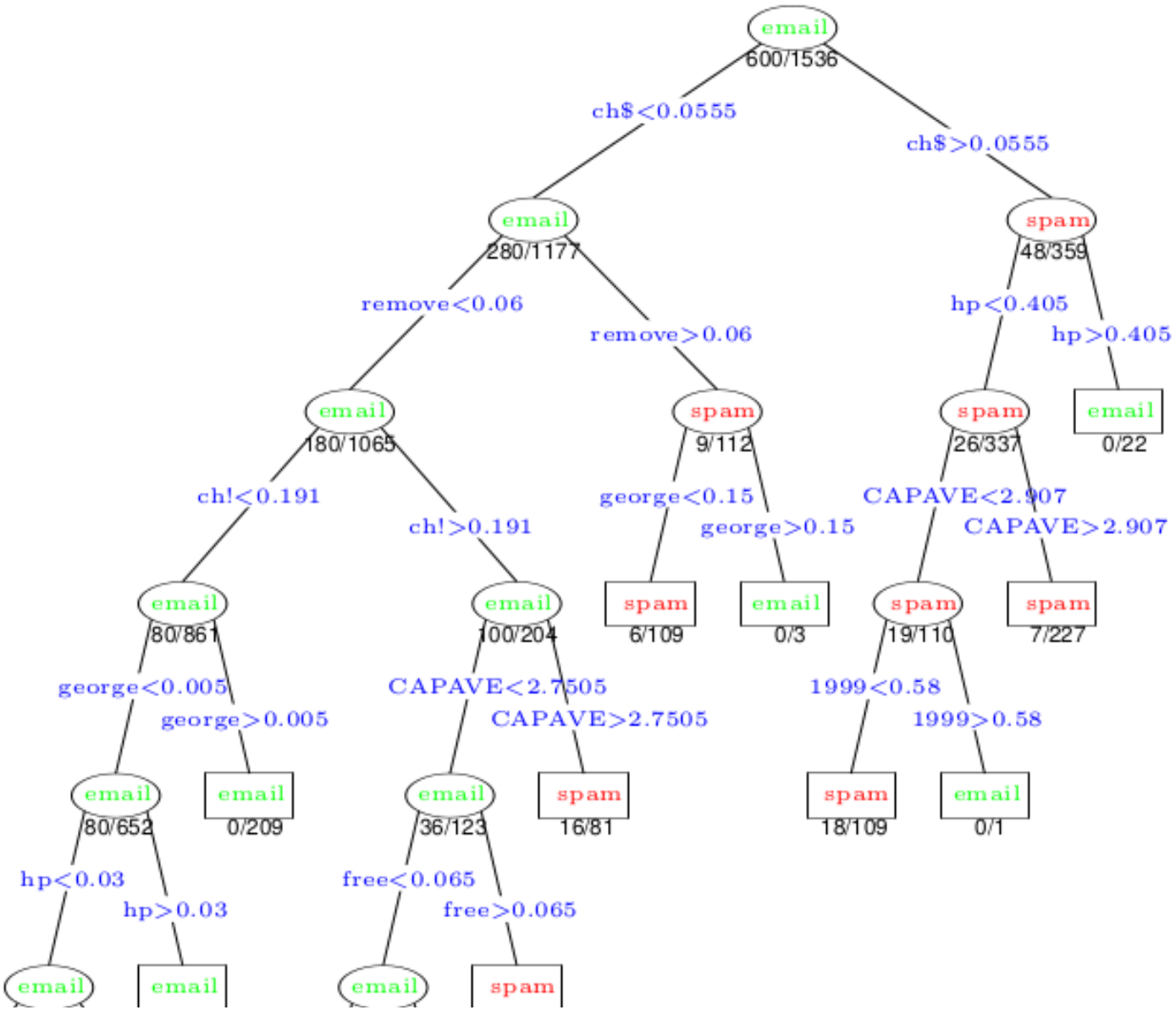
- change the loss function from $Q_m(T)$ to:
  - 0-1 loss: $N_m^{-1} \sum_{x_i \in R_m} \mathbb{1}(y_i \neq k_m)$;
  - Gini index: $\sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$;
  - deviance: $\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$;

  - all three can be extended to consider different error weights.

Classification trees: loss functions



Classification trees: example



Tree-based Methods: remarks

Tree-based methods:

- fast to construct, interpretable models;
- can incorporate mixtures of numeric and categorical inputs;
- immune to outliers, resistant to irrelevant inputs;
- lack of smoothness;
- difficulty in capturing additive structures;
- highly unstable (high variance).

Bagging: Galton (1907)

17°·0 at Moyeni, Basutoland, on August 23. The mean yearly value of the absolute maxima was 86°·9, and of the corresponding minima 41°·6. The mean temperature for the year was 0°·9 below the average. The stormiest month was October, and the calmest was April.

We have also received the official meteorological year-books for South Australia (1904) and Mysore (1905). Both of these works contain valuable means for previous years.

Forty Years of Southern New Mexico Climate.—Bulletin No. 59 of the New Mexico College of Agriculture contains the meteorological data recorded at the experimental station from 1892 to 1905 inclusive, together with results of temperature and rainfall observations at other stations in the Mesilla Valley for most of the years between 1851 and 1890, published some years ago by General Greely in a "Report on the Climate of New Mexico." The station is situated in lat. 32° 15′ N., long. 106° 45′ W., and is 3868 feet above sea-level. The data have a general application to those portions of southern New Mexico with an altitude less than 4000 feet. The mean annual temperature for the whole period was 61°·6, mean maximum (fourteen years) 76°·8, mean minimum 41°·4, absolute maximum 106° (which occurred several times), absolute minimum 1° (December, 1895). The mean annual rainfall was 8·8 inches; the smallest yearly amount was 3·5 inches, in 1873, the largest 17·1 inches, in 1905. Most of the rain falls during July, August, and September. The relative humidity is low, the mean annual amount being about 51 per cent. The bulletin was prepared by J. D. Tinsley, vice-director of the station.

Meteorological Observations in Germany.—The results of the observations made under the system of the Deutsche Seewarte, Hamburg, for 1905, at ten stations of the second order, and at fifty-six storm-warning stations, have been received. This is the twenty-eighth yearly volume published by the Seewarte, and forms part of the series of German meteorological year-books. We have frequently referred to this excellent series, and the volume in question is similar in all respects to its predecessors; it contains most valuable data relating to the North Sea and Baltic coasts. We note that the sunshine at Hamburg was

Distribution of the estimates of the dressed weight of a particular living ox, made by 787 different persons.

Degrees of the length of Array 0°—100°	Estimates in lbs.	Centiles: Observed deviates from 1207 lbs.	Centiles: Normal p.e =37	Excess of Observed over Normal
°				
5	1074	−133	−90	+43
10	1109	− 98	−70	+28
15	1126	− 81	−57	+24
20	1148	− 59	−46	+13
q_1 25	1162	− 45	−37	+ 8
30	1174	− 33	−29	+ 4
35	1181	− 26	−21	+ 5
40	1188	− 19	−14	+ 5
45	1197	− 10	− 7	+ 3
m 50	1207	0	0	0
55	1214	+ 7	+ 7	0
60	1219	+ 12	+14	− 2
65	1225	+ 18	+21	− 3
70	1230	+ 23	+29	− 6
q_3 75	1236	+ 29	+37	− 8
80	1243	+ 36	+46	−10
85	1254	+ 47	+57	−10
90	1267	+ 52	+70	−18
95	1293	+ 86	+90	− 4

q_1, q_3, the first and third quartiles, stand at 25° and 75° respectively.
m, the median or middlemost value, stands at 50°.
The dressed weight proved to be 1198 lbs.

According to the democratic principle of "one vote one value," the middlemost estimate expresses the *vox populi*, every other estimate being condemned as too low or too high by a majority of the voters (for fuller explanation see "One Vote, One Value," NATURE, February 28, p. 414). Now the middlemost estimate is 1207 lb., and the weight of the dressed ox proved to be 1198 lb.;

Bagging: Galton (1907)

In 1907, Sir Francis Galton visited a country fair:

A weight-judging competition was carried on at the annual show of the West of England Fat Stock and Poultry Exhibition recently held at Plymouth. A fat ox having been selected, competitors bought stamped and numbered cards [. . .] on which to inscribe their respective names, addresses, and estimates of what the ox would weigh after it had been slaughtered and "dressed". Those who guessed most successfully received prizes. About 800 tickets were issued, which were kindly lent me for examination after they had fulfilled their immediate purpose.

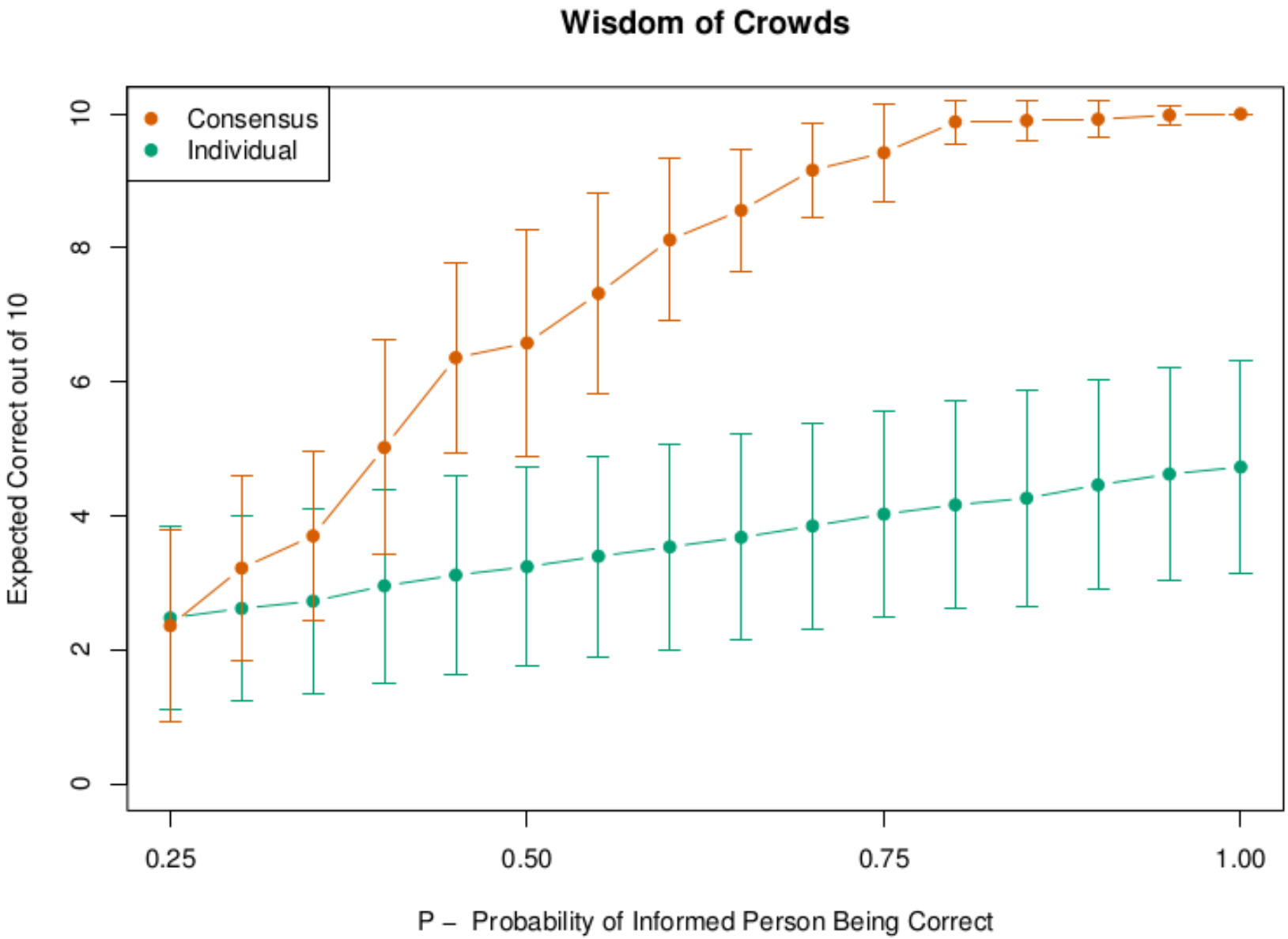
Bagging: Galton (1907)

After having arrayed and analyzed the data, Galton (1907) stated:

It appears then, in this particular instance, that the vox populi is correct to within 1 per cent of the real value, and that the individual estimates are abnormally distributed in such a way that it is an equal chance whether one of them, selected at random, falls within or without the limits of -3.7 per cent and +2.4 per cent of their middlemost value.

Concept of "**Wisdom of Crowds**" (or, as Schapire & Freund, 2014, "how it is that a committee of blockheads can somehow arrive at a highly reasoned decision, despite the weak judgement of the individual members.")

Bagging: wisdom of crowds



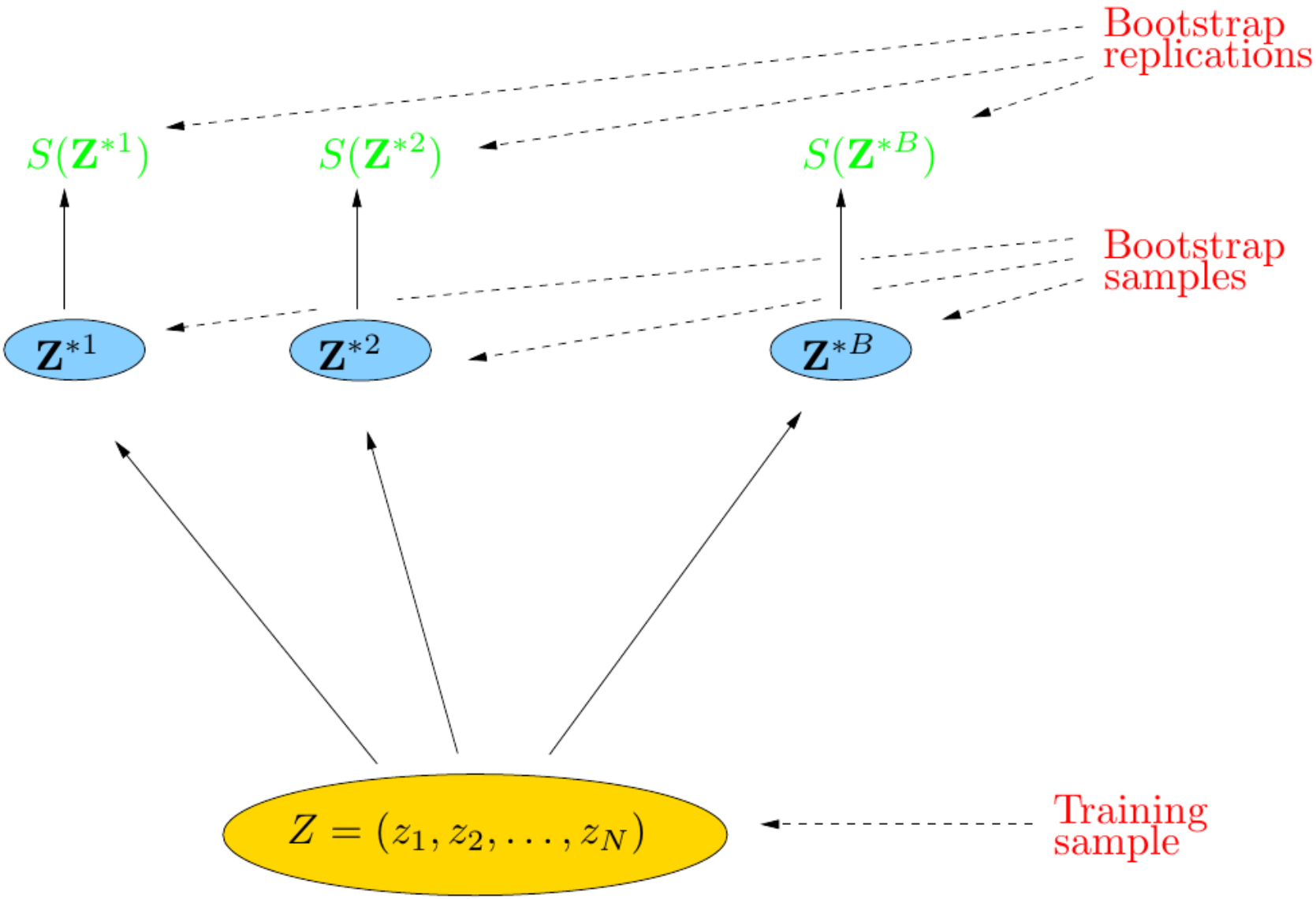
Bagging: translate this message into trees

How do can we translate this idea into tree-based methods?

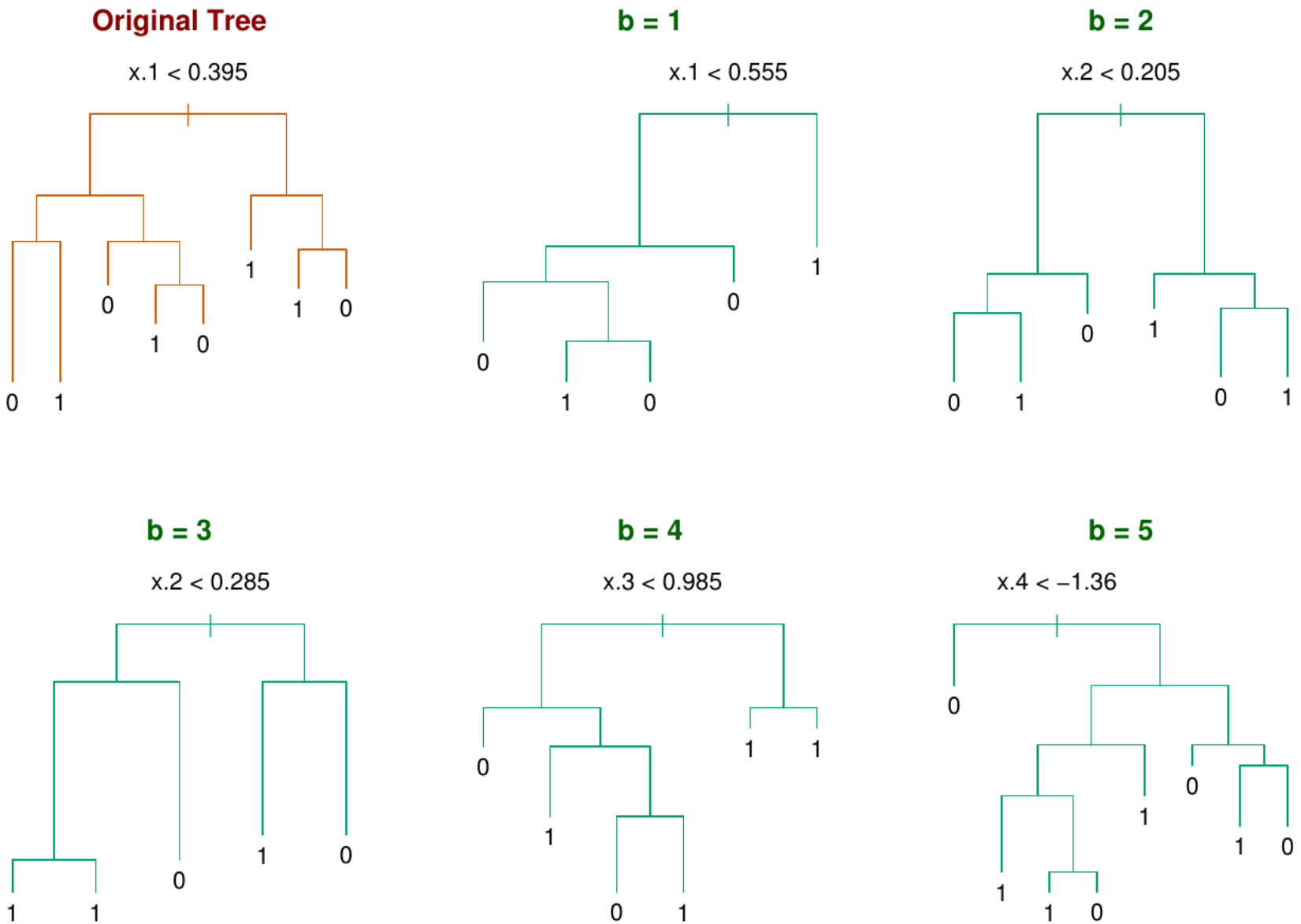
- we can fit several trees, then aggregate their results;
- problems:
  - "individuals" are supposed to be independent;
  - we have only one dataset ...

How can we mimic different datasets while having only one?

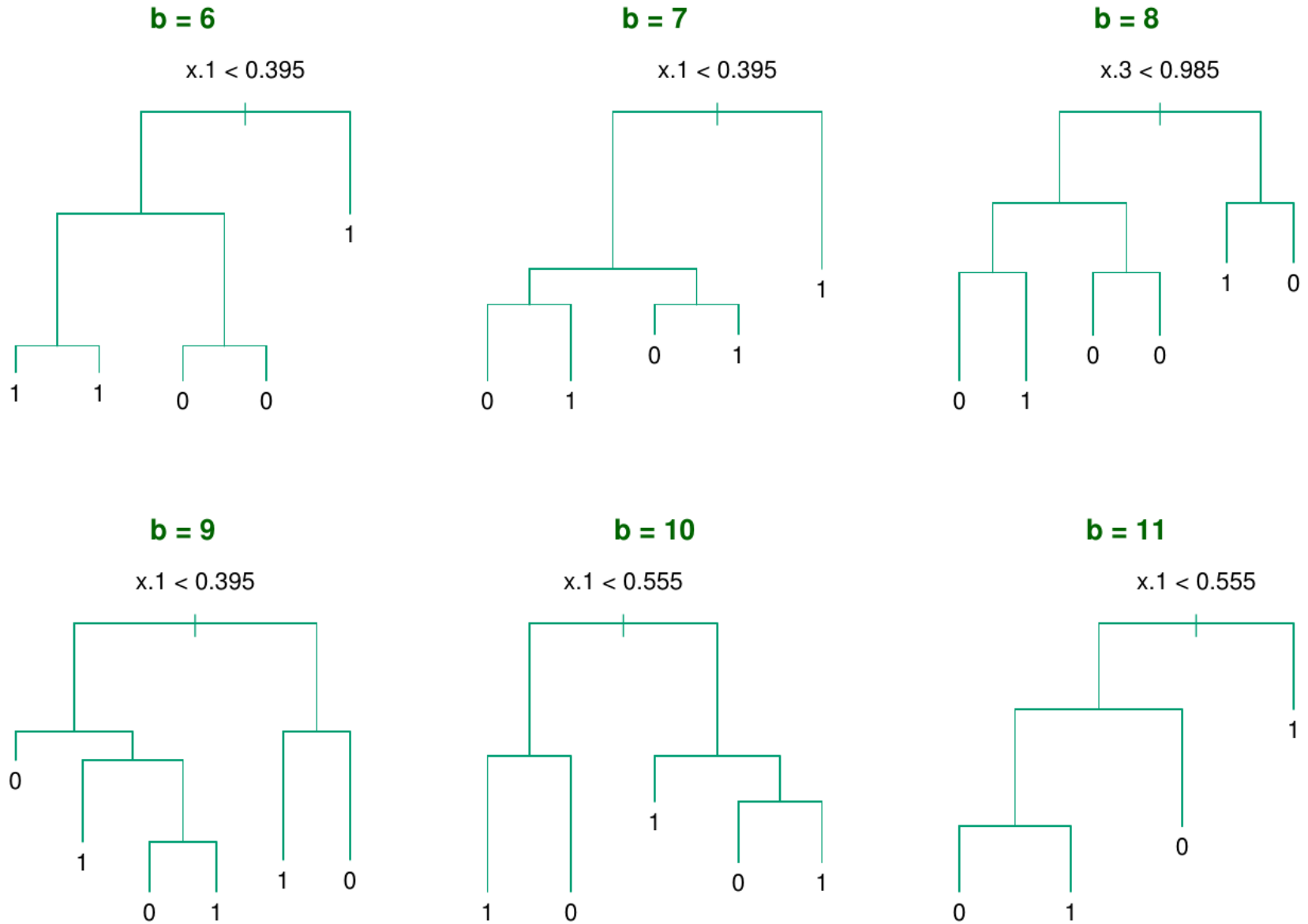
Bagging: the solution is . . .



Bagging: bootstrap trees



Bagging: bootstrap trees



Bagging: bootstrap trees

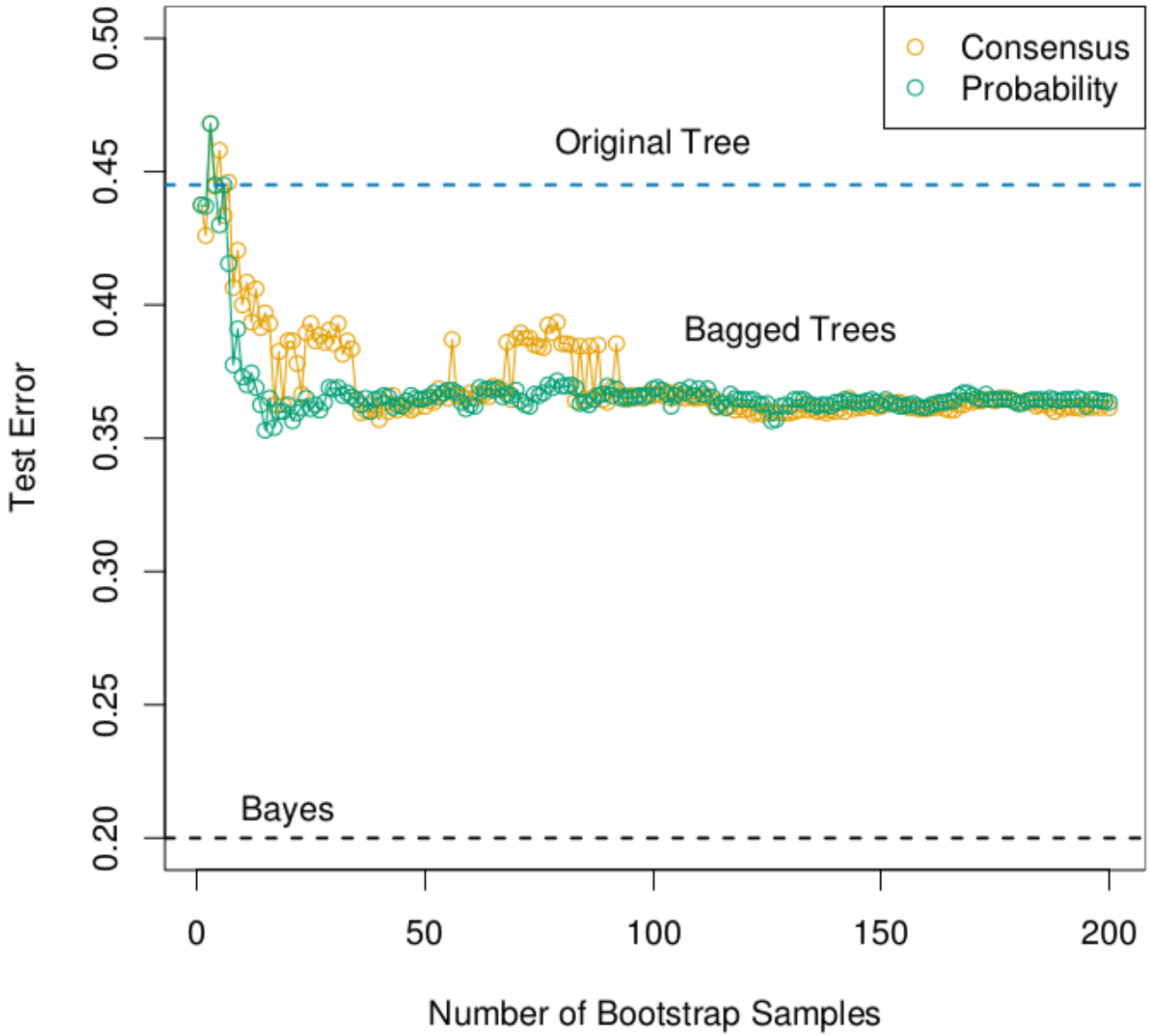
The procedure so far:

- generate bootstrap samples;
- fit a tree on each bootstrap sample;
- obtain B trees.

At this point, aggregate the results. How?

- **consensus**: $\hat{G}(x) = \text{argmax}_k \, q_k(x)$, $k \in \{1, \ldots, K\}$,
  - where $q_k(x)$ is the proportion of trees voting for the category k;
- **probability**: $\hat{G}(x) = \text{argmax}_k \, B^{-1} \sum_{b=1}^{B} p_k^{[b]}(x)$, $k \in \{1, \ldots, K\}$,
  - where $p_k^{[b]}(x)$ is the probability assigned by the b-th tree to category k;

Bagging: bootstrap trees



Bagging: general

In general, consider the training data $Z = \{(y_1, x_1), \ldots, (y_N, x_N)\}$. The **bagging** (**b**ootstrap **agg**regat**ing**) estimate is defined by

$$\hat{f}_{\text{bag}}(x) = E_{\hat{\mathcal{P}}}[\hat{f}^*(x)],$$

where:

- $\hat{\mathcal{P}}$ is the empirical distribution of the data (y_i, x_i);
- $\hat{f}^*(x)$ is the prediction computed on a bootstrap sample Z^*;
- i.e., $(y_i^*, x_i^*) \sim \hat{\mathcal{P}}$.

The empirical version of the bagging estimate is

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^*(x),$$

where B is the number of bootstrap samples.

Bagging: variance

Bagging has smaller prediction error because it reduces the variance component,

$$\begin{aligned} E_{\mathcal{P}}[(Y - \hat{f}^*(x))^2] &= E_{\mathcal{P}}[(Y - f_{\text{bag}}(x) + f_{\text{bag}}(x) - \hat{f}^*(x))^2] \\ &= E_{\mathcal{P}}[(Y - f_{\text{bag}}(x))^2] + E_{\mathcal{P}}[(f_{\text{bag}}(x) - \hat{f}^*(x))^2] \\ &\geqslant E_{\mathcal{P}}[(Y - f_{\text{bag}}(x))^2], \end{aligned}$$

where $\mathcal{P}$ is the data distribution.

Note that this does not work for 0-1 loss:

- due to non-additivity of bias and variance;
- bagging makes better a good classifier, worse a bad one.

Bagging: from bagging to random forests

The average of B identically distributed r.v. with variance σ^2 and positive pairwise correlation ρ has variance

$$\rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.$$

- as B increases, the second term goes to 0;
- the bootstrap trees are p. correlated $\rightarrow$ first term dominates.

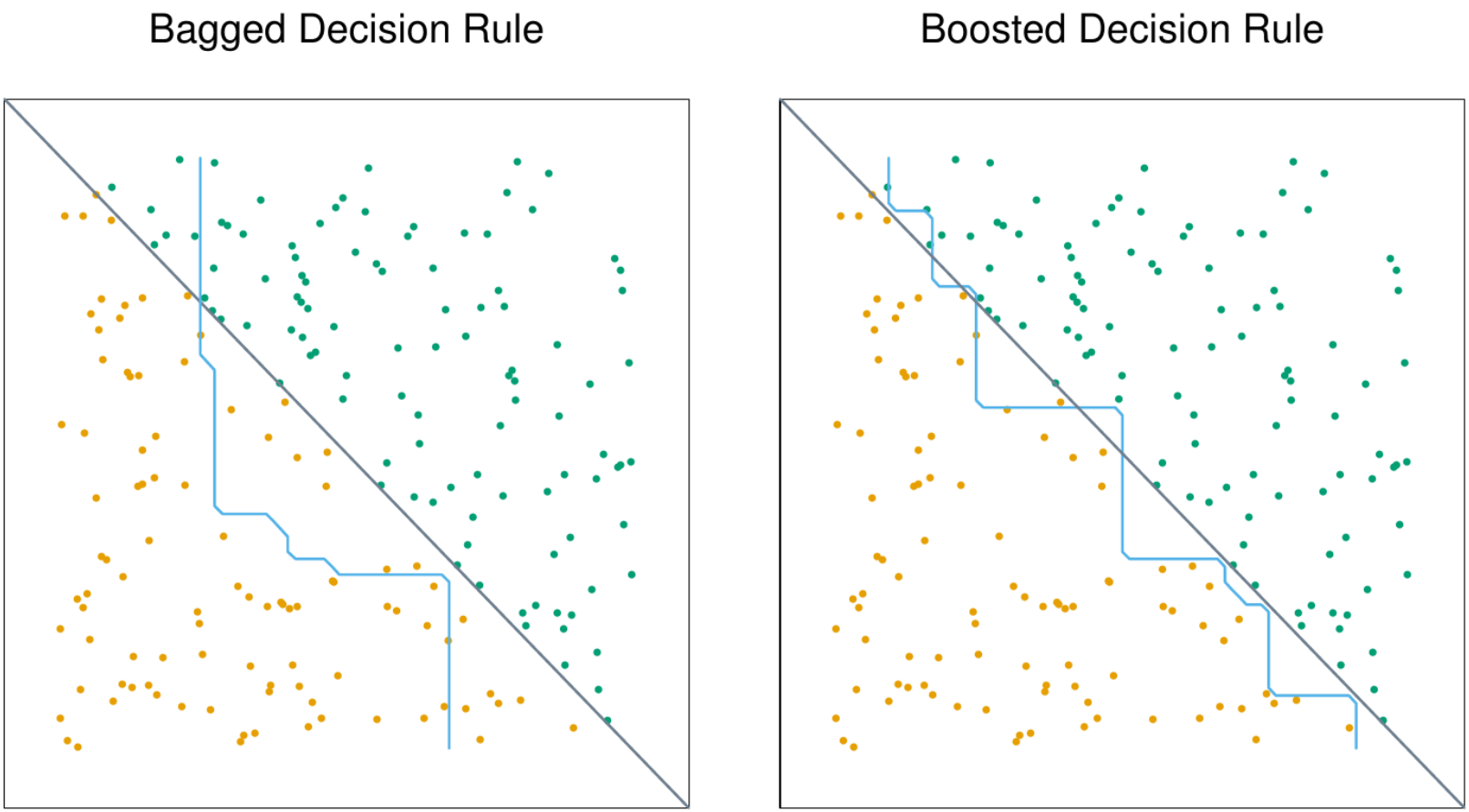
$\downarrow$

construct bootstrap tree as less correlated as possible

$\downarrow$

random forests

Bagging: from bagging to boosting



References I

Galton, F. (1907). Vox populi. *Nature* **75**, 450–451.

Schapire, R. E. & Freund, Y. (2014). *Boosting: Foundations and Algorithms*. MIT Press, Cambridge.