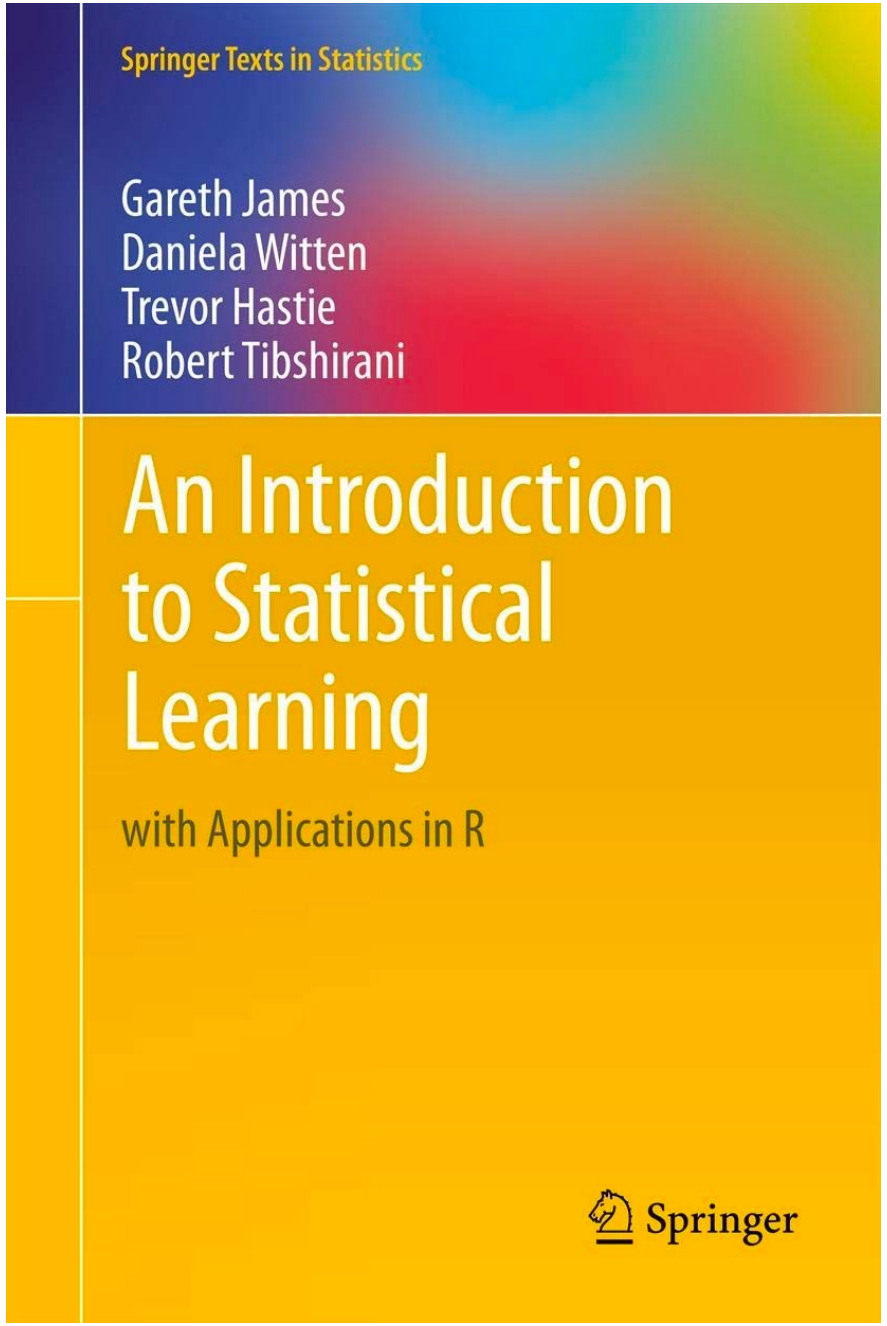
Tree-based methods

正田 備也

masada@rikkyo.ac.jp

この本の第8章を使って、
授業内容を組み立てました。
（図も同章から取っています。）

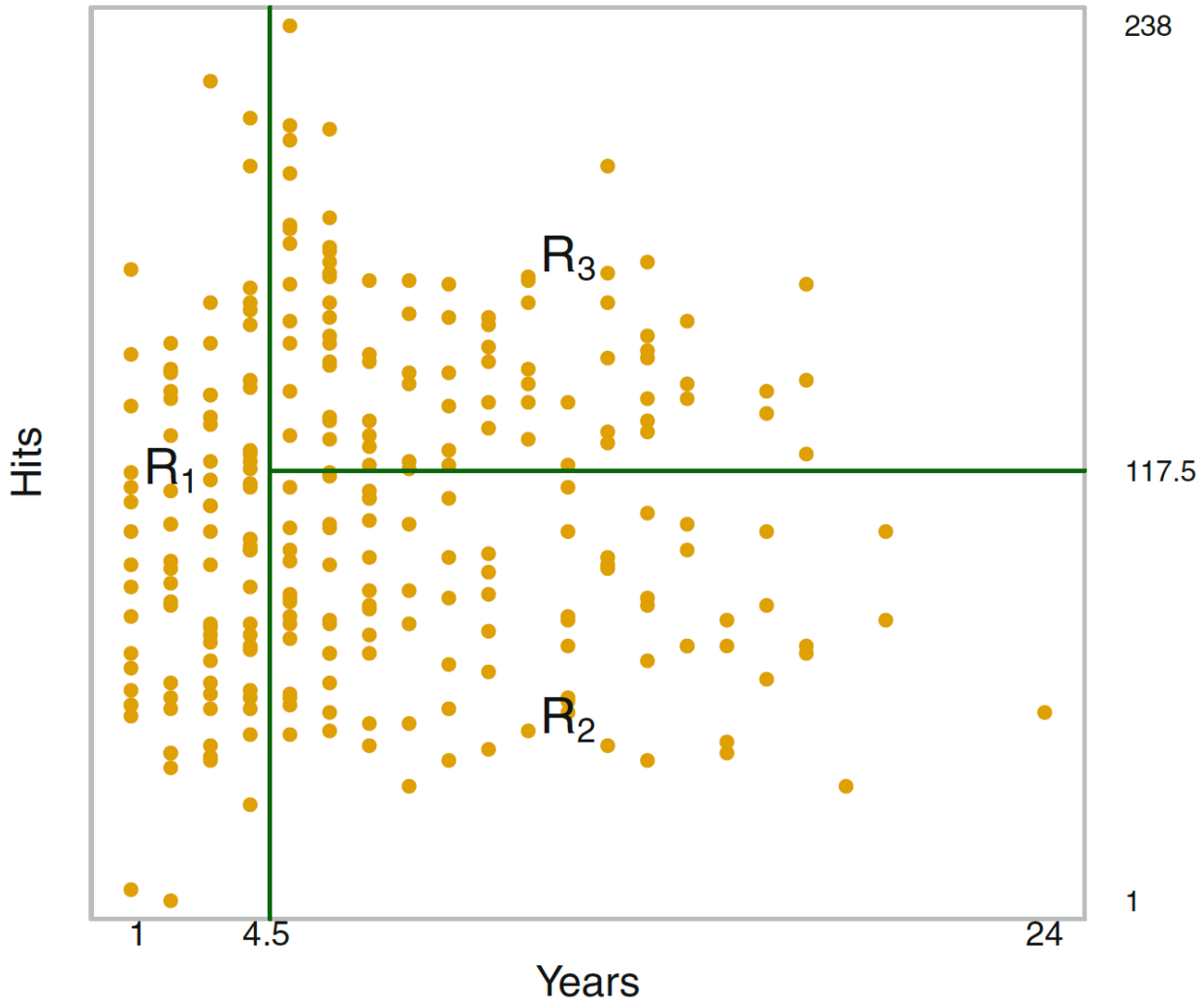


https://link.springer.com/book/10.1007/978-1-4614-7138-7

Tree-based methodsの概略

- 入力ベクトルが属する空間を、単純な領域へ分割
  - 分割していく一連のルールを、木構造で表現する
  - そのため、tree-based methodsと呼ばれる
- 未知の入力ベクトルについては、以下のように予測
  - その未知ベクトルが属する領域を、つきとめる
  - その領域に属する訓練データについて、対応する目的変数の平均値（回帰）や最頻値（分類）を求め、これを予測値とする
    - ちなみに「決定木」は、分類のためのtree-based methodsのこと。

空間を分割していく一連のルールを
木構造として要約したもの
（下端の数値は野球選手の年俸）



空間が分割されている様子の図示

分割の方針：RSS（残差平方和）の最小化

- 下記で定義されるRSSができるだけ小さくなるように、入力ベクトルが属する空間の分割$R_1, \ldots, R_J$を求める

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

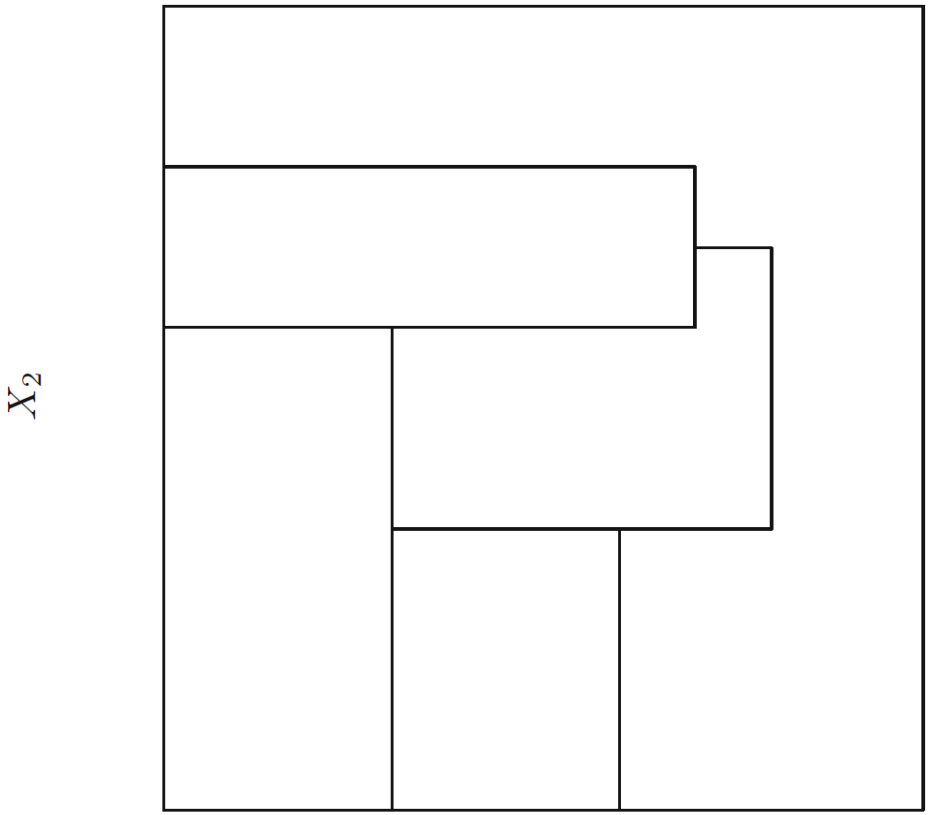
  - R_jは、第j番目の領域に含まれる訓練データの添字の集合
  - $\hat{y}_{R_j}$は、第j番目の領域に含まれる訓練データの目的変数の値の平均

学習アルゴリズム：Recursive binary splitting

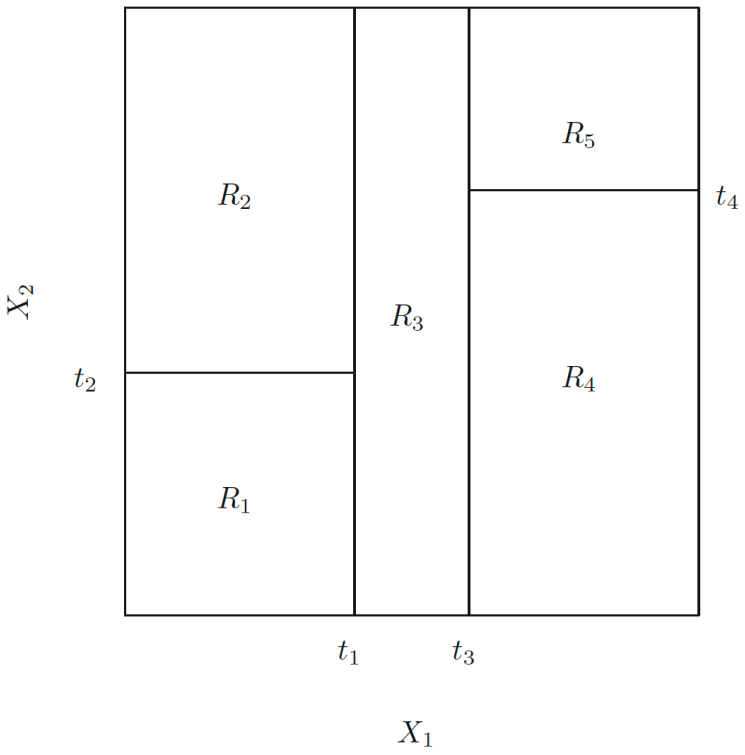
- 入力ベクトルが属する空間を、貪欲に2分割していく方法
  - 貪欲＝先のことを考えず、毎ステップで最善の答えを選択する

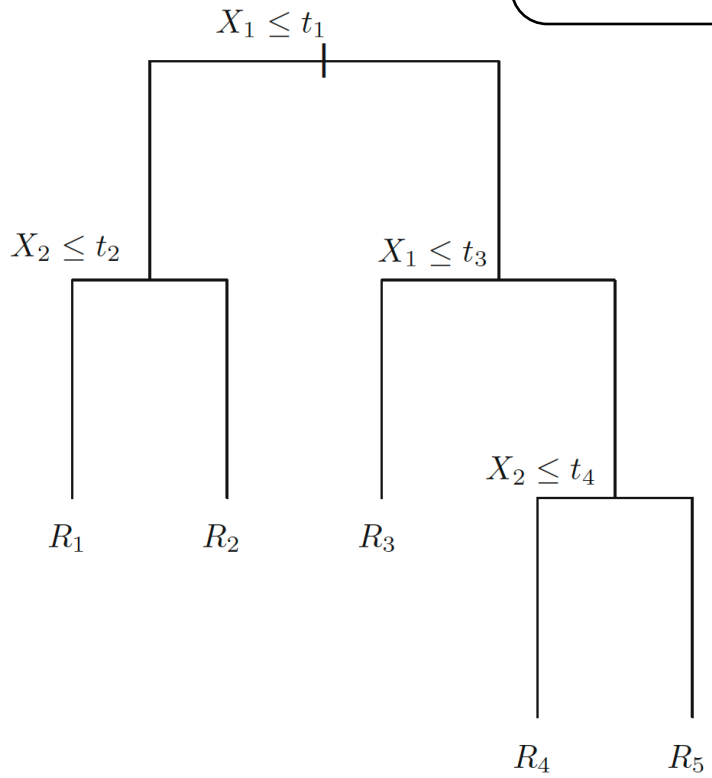
1. 説明変数のうち一つとその切断点sとを、下記の基準で決定
   - 当該の説明変数の値がs未満のものと、s以上のものとへ訓練データを分割した場合に、RSSを最も大きく減らすことができる。
2. 前もって定めた終了条件が満たされていないなら1.へ戻り、満たされていれば終了
   - 終了条件の例：「どの領域にも5個以下しか訓練データが含まれない」

Recursive binary splittingでは実現できない分割の例

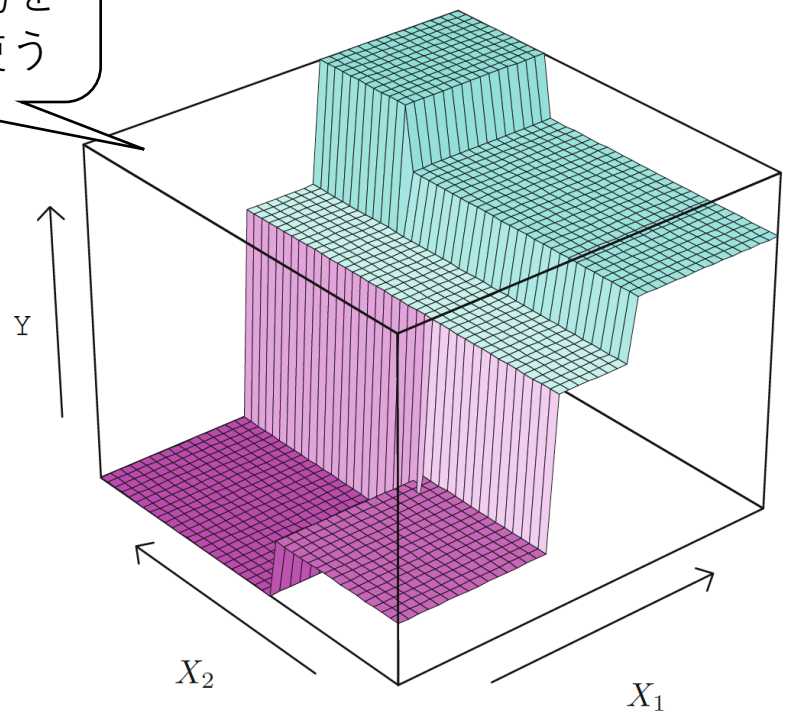


Recursive binary splittingの実行例



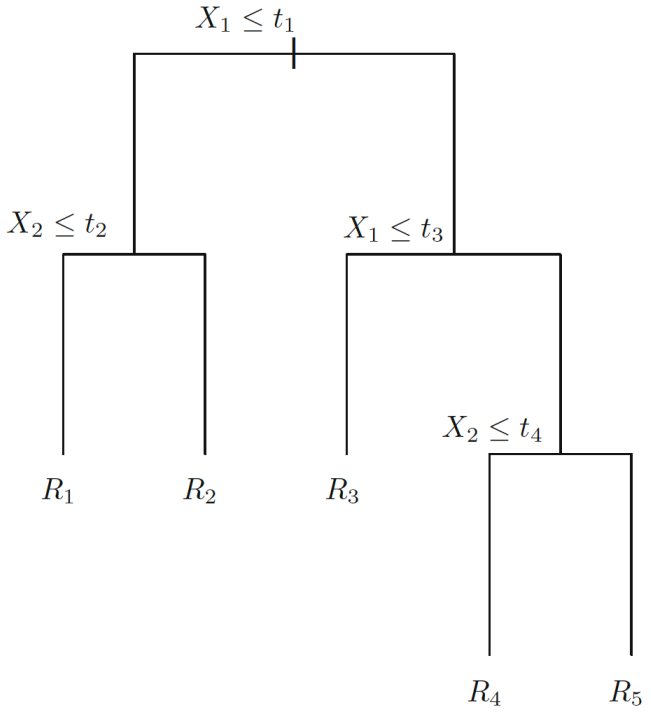


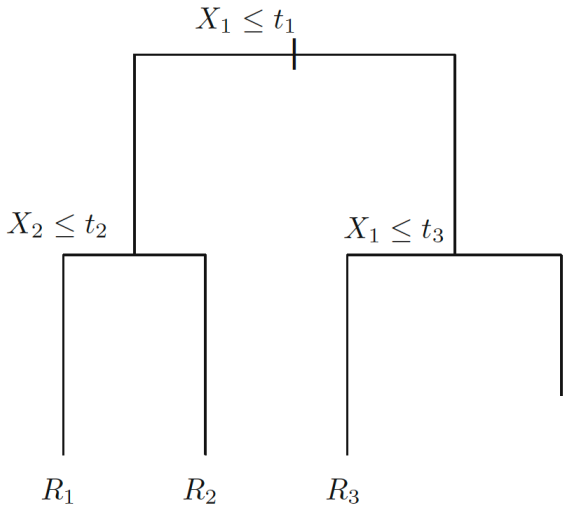
各領域での
responseの平均を
予測値として使う

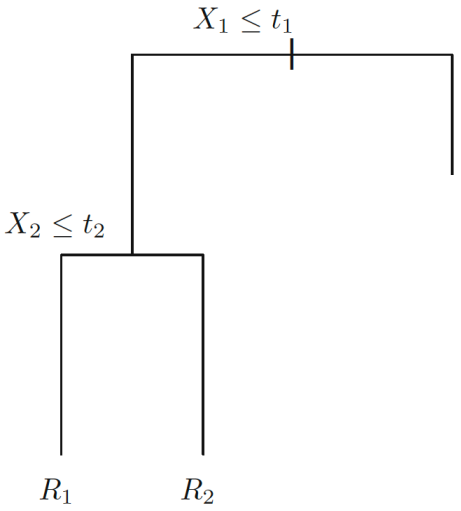


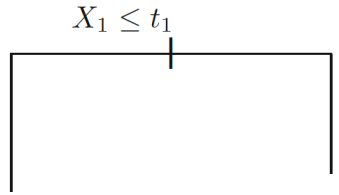
Tree pruning

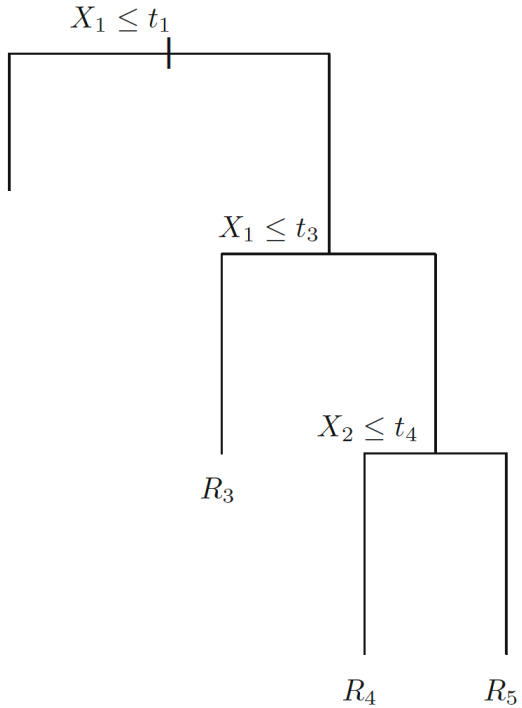
- 上記の手法で得た分割は、訓練データに合いすぎていて、訓練データ以外のデータでうまくいかないことが多い
  - いわゆる過学習。
- そこで、枝刈りをする
- 枝刈りの単純な方法
  - Recursive binary splittingで木を求める。この木をT_0と呼ぶ。
  - T_0の全ての部分木Tを交差検証で評価し、最も良い部分木を選ぶ。
  - しかし、この方法は時間がかかりすぎる。

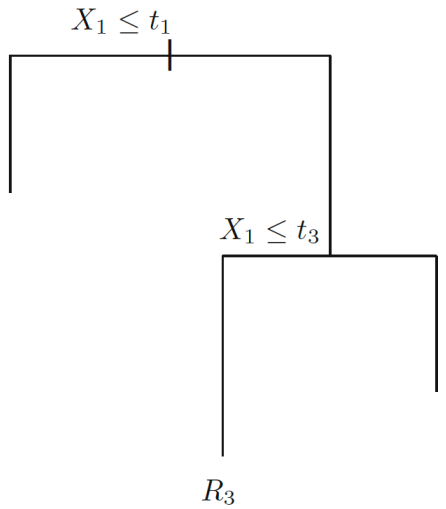












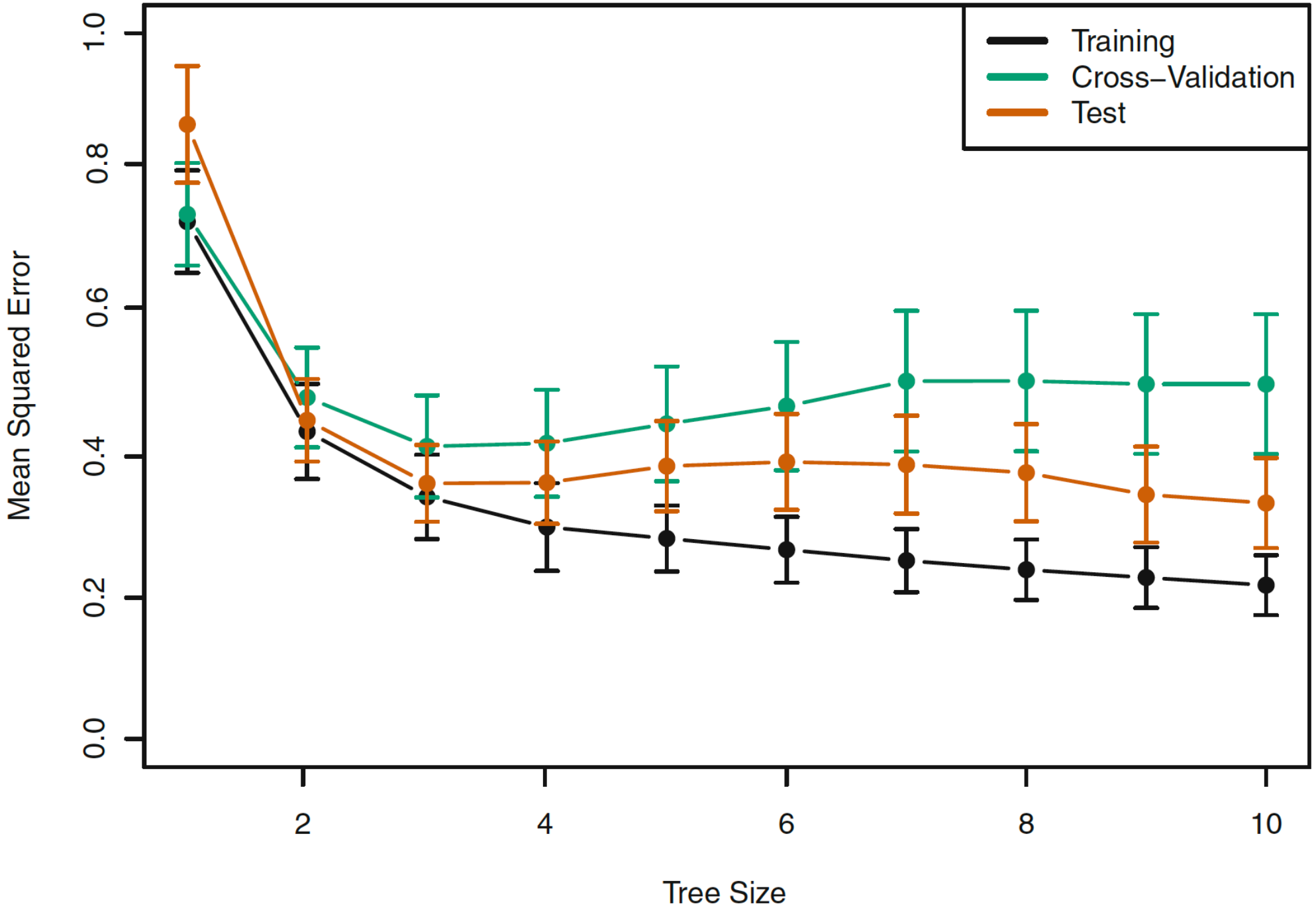
元の木（左端）の
部分木の例

木の複雑さをコントロールする

- RSSに、木の複雑さを表す項を追加する

$$\sum_{j=1}^{|T|}\sum_{i\in R_j}(y_i-\hat{y}_{R_j})^2+\alpha|T|$$

- 上記の関数を最小化するαを、交差検証で決定する
  - $|T|$ は木 T の葉ノードの個数。つまり分割によって得られる領域の個数。
  - $\alpha|T|$は、木が複雑になることによるペナルティを表す項。


Training
Cross–Validation
Test
Mean Squared Error
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Tree Size

分類にtree-based methodsを使う場合

- 予測：各領域で最多数派を予測に使う（これはこれでいいとして…）

- 学習：領域を分割していくとき、何を最小化するか？
  - 領域を表す添字をj、クラスを表す添字をkとする。

1. 最多数派以外のデータ点の割合：$1 - \max_k(\hat{p}_{jk})$
   - これではうまくいかないらしい。
2. Gini係数（分散に相当）：$\sum_{k=1}^{K} \hat{p}_{jk}\left(1 - \hat{p}_{jk}\right) = 1 - \sum_{k=1}^{K} {\hat{p}_{jk}}^2$
3. エントロピー：$-\sum_{k=1}^{K} \hat{p}_{jk} \log \hat{p}_{jk}$

Tree-based methodsの長所と短所

□人に説明しやすい

□人間のdecision-makingに近い（？）

□可視化しやすい（解釈が容易）

□カテゴリカル変数の扱いも簡単（ダミー変数不要）

■他の回帰手法や分類手法に比べて性能が悪いことがある

■データ集合の少しの変更が木の構造を大きく変えることがある

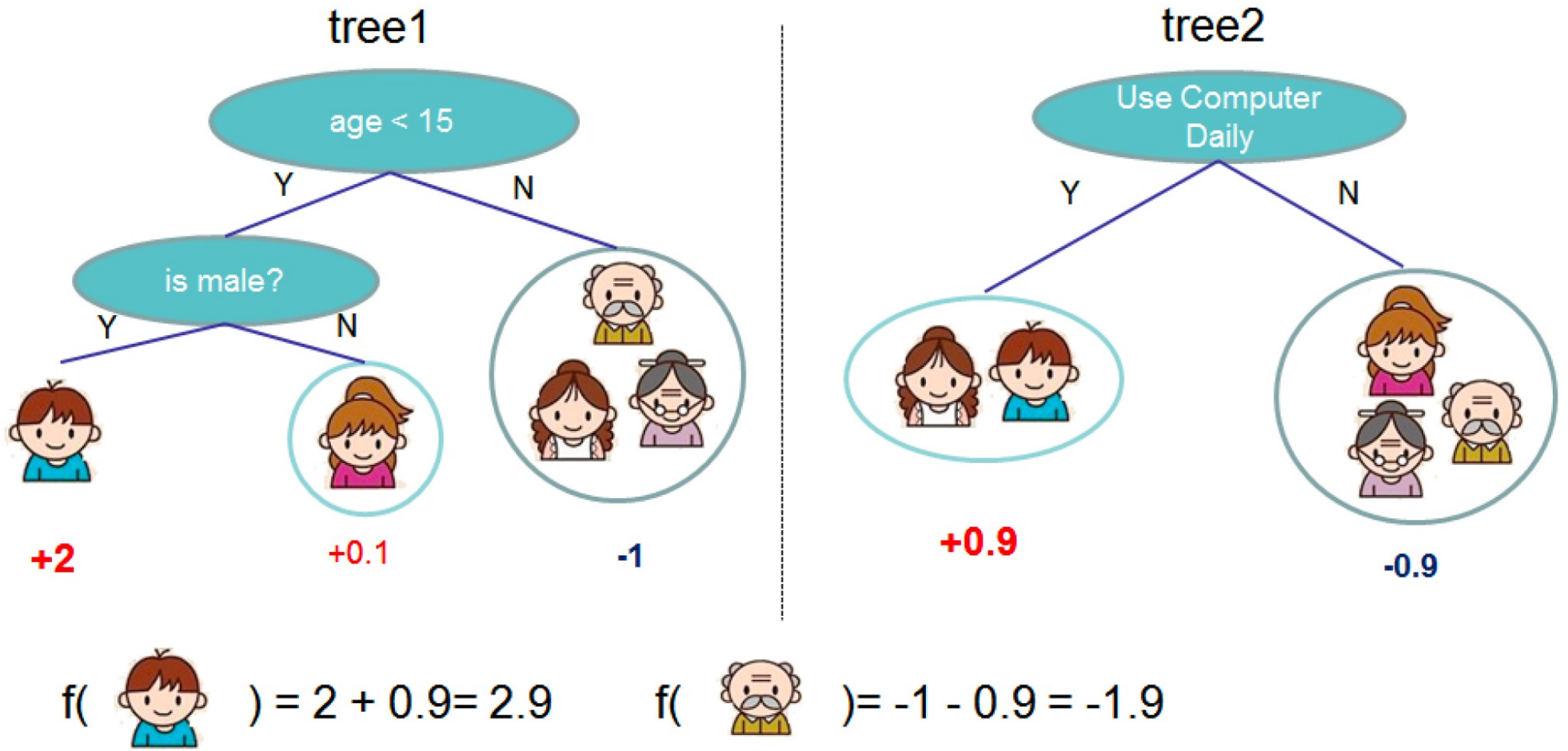
- つまり過学習を起こしやすい

短所を克服する3つの方法

1. バギング (bagging)
2. ランダム・フォレスト (random forests)
3. ブースティング (boosting)

- いずれも複数の木を使う（アンサンブル学習）

Tree ensemble model



Chen and Guestrin. XGBoost: A Scalable Tree Boosting System. arXiv:1603.02754.

バギング (bagging)

- 例えば、訓練データをランダムに等分してそれぞれで木を作ると、かなり違った木になることが多い
  - こういう現象をhigh varianceと呼ぶ。
- そこで、訓練データからランダムに部分集合をいくつも採って、それぞれで木を作り、これらの出力の平均を予測に使う
  - 部分集合をランダムに選ぶときは、復元抽出をおこなう。
  - 分類の場合は、複数の木の出力で多数決をとり、予測に使う。
  - 元はhigh varianceでも、平均をとるとvarianceを下げられる。
- 補足：バギングは他の分類手法でも使える

ランダム・フォレスト (random forests)

- バギングとほとんど同じ
- 違うのは、領域を分割するときに、p個ある説明変数すべてではなく、ランダムに選んだm個の説明変数だけを考慮する点
  - バギングでは、すべての説明変数を考慮する。
    - つまり、p通りの分割の可能性のすべてを考える。
  - いくつの説明変数を考慮するかについては、例えば$m \approx \sqrt{p}$と設定する。
- 一部の説明変数だけを考慮することで、たくさん作る木がお互いに似てきてしまうことを防ぐ

ブースティング (boosting)

- 訓練データに変更を加えつつ、順番にたくさんの木を作る
- 最初は、元の訓練データそのものを使って、木を作る
- 次は、いま作った木による予測の、訓練データからのズレ（残差）を、新たな訓練データと思って、二つ目の木を作る
- その次は、いま作った木による予測の、直前の訓練データからのズレを、あらなた訓練データと思って、三つ目の木を作る
- 以下、この繰り返し

- 入力：訓練データ$(\boldsymbol{X}, \boldsymbol{y})$　ただし$\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$、$\boldsymbol{y} = \{y_1, \ldots, y_N\}$

1. モデルを$\hat{f}(\boldsymbol{x}) = 0$と初期化、また、全ての$\boldsymbol{x}_i$について$r_i = y_i$とする
2. for b in $[1, \ldots, B]$:
   a. 木$\hat{f}^b$に、深さがdになるまで、訓練データ$(\boldsymbol{X}, \boldsymbol{r})$ で学習させる
   b. 予測に使うモデル$\hat{f}$を、以下のように更新（λは0.01や0.001などの値を使う）

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

   c. 残差を、以下のように更新

$$r_i \leftarrow r_i - \lambda \hat{f}^b(\boldsymbol{x}_i)$$

3. 以下のモデルを出力する

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

Gradient tree boosting

- 以下の論文を参照
  - https://arxiv.org/abs/1603.02754