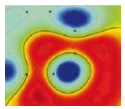
Decision Trees and Random Forrests

berlin
brain computer
interface

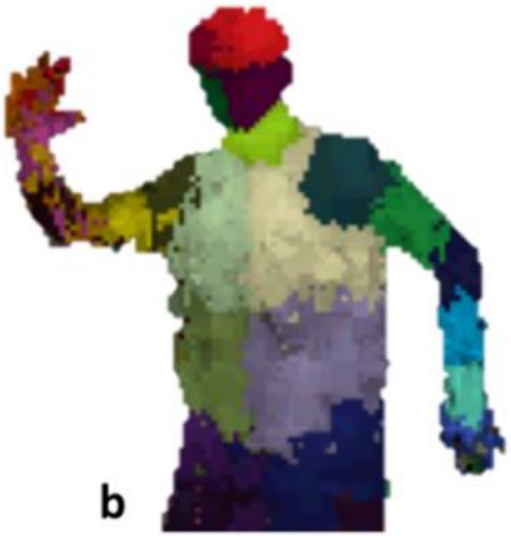
Klaus-Robert Müller

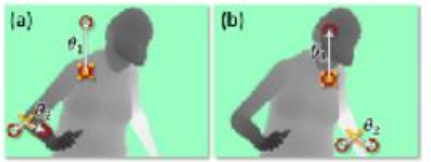


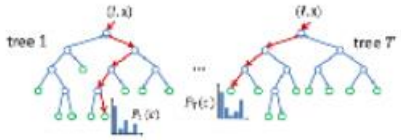
Tree-based Methods - applications

Microsoft Kinect Pose Estimation

(Classifications Forests in Kinect for XBox 360)





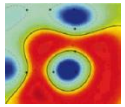


J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake. Real-Time Human Pose Recognition in Parts from a Single Depth Image (2011)

Tree-based Methods - applications

Other "real life" applications:

- Recommender systems (Facebook, Amazon, Netflix)
- Business applications (customer segmentation, target marketing)
- Medical (disease diagnosis)
- Banking (credit card issue, fraud detection)

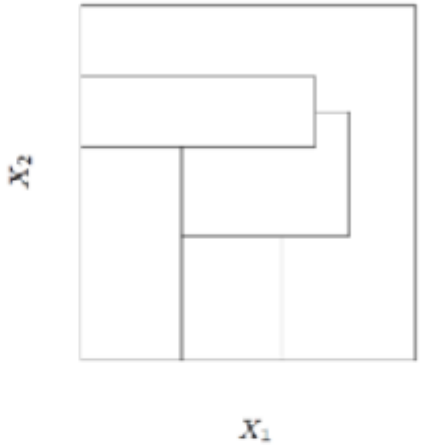


Tree-based Methods - background

Given: Features: $X_1, X_2, \ldots, X_p$
Target: Y

Key idea:

(1) Partition feature space into a set of rectangles.
(2) Fit a simple model (e.g. a constant) in each one.

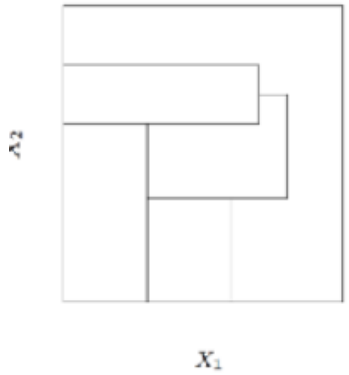


General Partition
(can not be obtained)

For each partition: model Y with a different constant.

The Elements of Statistical Learning, T. Hastie, R. Tibshirani, and J. Friedman (2001)

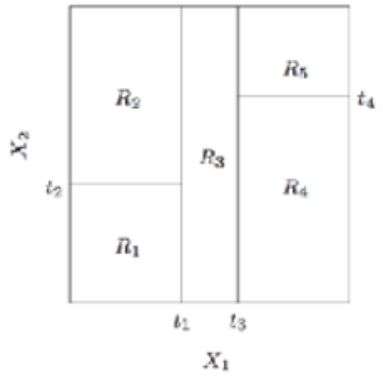
Tree-based Methods - background



General Partition
(can not be obtained)

Although each partitioning line has a simple description, e.g. X_1=c, some regions are complicated to describe.

Simplification: recursive binary partitions



Recursive
Binary Splitting

Algorithm (divide & conquer):

repeat

(1) Split space into two regions.
(2) Model response by mean of Y in each region.

The Elements of Statistical Learning, T. Hastie, R. Tibshirani, and J. Fried

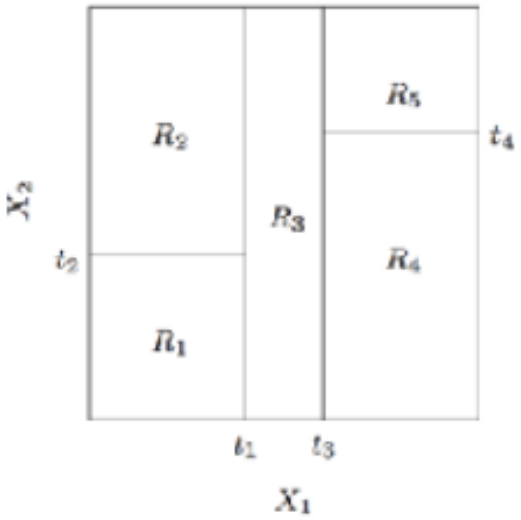
树模型方法——背景

- 虽然每条划分线都可以用简单的形式（例如 $X_1 = c$）来描述，但某些区域仍然较为复杂，不易直接表述。
- 简化思路：采用递归的二叉划分（recursive binary partitions）。

1. 将空间拆分为两个区域。
2. 在每个区域内，用目标变量 Y 的均值来构建模型。

左图：一般划分（无法直接得到） 右图：递归二叉划分

Decision Trees – conceptual construction



Recursive Binary Splitting

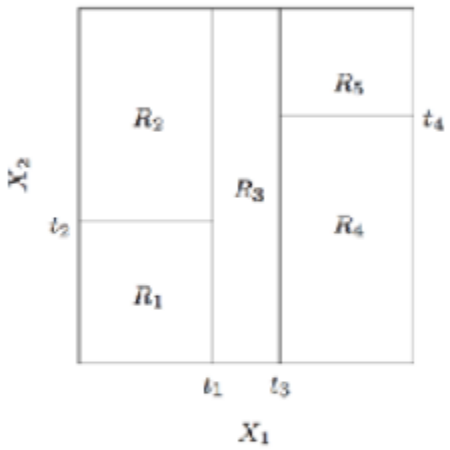
Example:

- Split at $X_1 = t_1$
- Split the region $X_1 <= t_1$ at t_2
- etc.

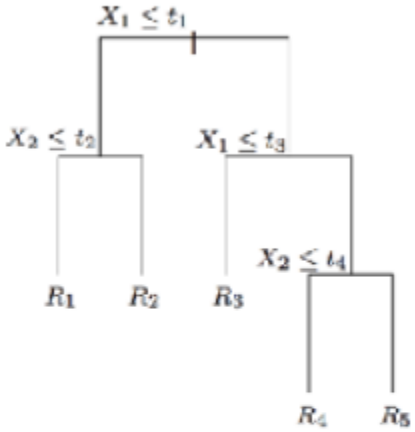
Result: Partitioning into R_1-R_5

Model: $\hat{f}(X) = \sum_{m=1}^{5} c_m I\{(X_1, X_2) \in R_m\}$

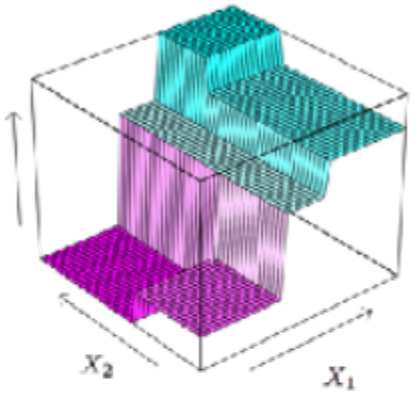
Decision Trees – conceptual construction



Recursive Binary Splitting



Decision Tree



Regression Surface

$$\hat{f}(X) = \sum_{m=1}^{5} c_m I\{(X_1, X_2) \in R_m\}$$

Decision Trees – algorithm

Given: *N* observations, *P* features

$$(x_i, y_i) \text{ for } i = 1, 2, \ldots, N, \text{ with } x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}).$$

Wanted: splitting variables *j*, split points *s*
(to partition space into M regions)

目标：在这些特征中选择用于划分的特征变量 j 以及相应的划分点 s，从而将特征空间划分为 M 个区域。

Minimization criterion: e.g. $\hat{c}_m = \text{ave}(y_i | x_i \in R_m)$ (sum of squares)

Finding best binary partition is computationally infeasible (NP hard): **use greedy approach**

Decision Trees – algorithm - splitting

Given: *N* observations, *P* features

$$(x_i, y_i) \text{ for } i = 1, 2, \ldots, N, \text{ with } x_i = (x_{i1}, x_{i2}, \ldots, x_{ip}).$$

Consider a splitting variable *j* and split point *s* and define a pair of half planes:

$$R_1(j, s) = \{X | X_j \leq s\} \text{ and } R_2(j, s) = \{X | X_j > s\}$$

We seek *j* and *s* to solve:

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

Splitting:

Inner minimization solved by: $\hat{c}_1 = \text{ave}(y_i | x_i \in R_1(j, s))$ and $\hat{c}_2 = \text{ave}(y_i | x_i \in R_2(j, s))$

I.e. scan through all *P* features and determine optimal (j,*s*)

Decision Trees – algorithm – splitting (metrics)

Information gain

Based on the concept of entropy from information theory.

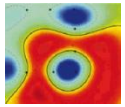
$$H(T) = I_E(p_1, p_2, \ldots, p_J) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

were $p_1, p_2, \ldots$ represent the probability of each class present in the child node that results form splitting the tree.

$$\overbrace{IG(T,a)}^{\text{Information Gain}} = \overbrace{H(T)}^{\text{Entropy(parent)}} - \overbrace{H(T|a)}^{\text{Weighted Sum of Entropy(Children)}}$$

Choose the split that results in the purest daughter nodes.

3. **选择划分依据**：在分类树中，我们选择使子节点熵最小（或信息增益最大）的划分。这等同于选择使得划分后子节点越纯的特征与阈值。



Decision Trees – algorithm – splitting (metrics)

Gini impurity (基尼不纯度)

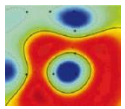
- 定义：基尼不纯度度量了如果根据当前子集的类别分布随机为一个随机挑选的样本分配标签，该样本被错误标记的概率有多大。

Gini impurtiy

Measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset

$$GI(\mathcal{X}) = \sum_{c=1}^{C} p(c)(1 - p(c)) = \sum_{c \neq \tilde{c}} p(c)p(\tilde{c})$$

Similar to entropy, zero if all samples belong to same class.



Decision Trees – algorithm – splitting (metrics)

Variance Reduction (often used for regression)

Variance reduction of a node is defined as the total reduction of the variance of the target variable due to the split at this node.

$$I_V(N) = \frac{1}{|S|^2}\sum_{i\in S}\sum_{j\in S}\frac{1}{2}(x_i - x_j)^2 - \left(\frac{1}{|S_t|^2}\sum_{i\in S_t}\sum_{j\in S_t}\frac{1}{2}(x_i - x_j)^2 + \frac{1}{|S_f|^2}\sum_{i\in S_f}\sum_{j\in S_f}\frac{1}{2}(x_i - x_j)^2\right)$$

were S, S_t, and S_f are the set of pre-split sample indices for which the split test is true and false, respectively.

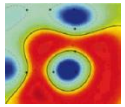
Variance Reduction (方差减少)

- 用途：常用于回归决策树。
- 定义：方差减少是指在节点处进行特征划分后，目标变量的方差（或误差）总量减少的程度。通过这种划分，我们希望子节点中数据的分布更集中，从而降低预测误差。

S 是划分前的样本集合。

S_t 和 S_f 分别是依据划分条件为真和为假时对应的子集（即划分后的两个子区域）。

方差减少即为划分前方差减去划分后两子集方差加权和。



Decision Trees – algorithm – pruning 剪枝

解决方案：先让树生长到较大规模，然后利用"成本-复杂度准则"（cost-complexity criterion）对树进行剪枝。剪枝通过折叠（collapse）一些内部节点来减少树的复杂度。

How large to grow the tree?

- too large: overfitting
- too small: not all important structure is captured

Solution: grow large tree, then prune using *cost-complexity criterion*
(collapse any number of internal (non-leaf) nodes)

$$N_m = \#\{x_i \in R_m\}, \quad \text{(training data in region)}$$

$$\hat{c}_m = \frac{1}{N_m} \sum_{x_i \in R_m} y_i, \quad \text{(prediction in region)}$$

$$Q_m(T) = \frac{1}{N_m} \sum_{x_i \in R_m} (y_i - \hat{c}_m)^2, \quad \text{(squared loss)}$$

$$|T| \quad \text{(number of terminal leaves)}$$

$$\alpha \geq 0 \quad \text{(regularization parameter)}$$

Find subtree that minimizes:

$$C_\alpha(T) = \sum_{m=1}^{|T|} \underbrace{N_m Q_m(T)}_{\text{cost}} + \underbrace{\alpha |T|}_{\text{complexity}}$$

Decision Trees – algorithm – pruning

Find subtree that minimizes:

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

cost complexity

最弱链剪枝法（Weakest Link Pruning）：

1. 从完整的决策树开始，不断折叠（合并）内部节点，每次选取那个在折叠后对 $\sum_m N_m Q_m(T)$ 的节点均值增加最小的内部节点进行合并。

   结果：通过不断合并内部节点，会产生一系列的子树序列 $T_0, T_1, T_2, \ldots, T_m$，其中 T_m 是只有一个叶节点的树（最简化的树）。

2. 利用交叉验证（cross-validation）在这组子树中选择最优的 α，从而确定最优的子树 T_i。

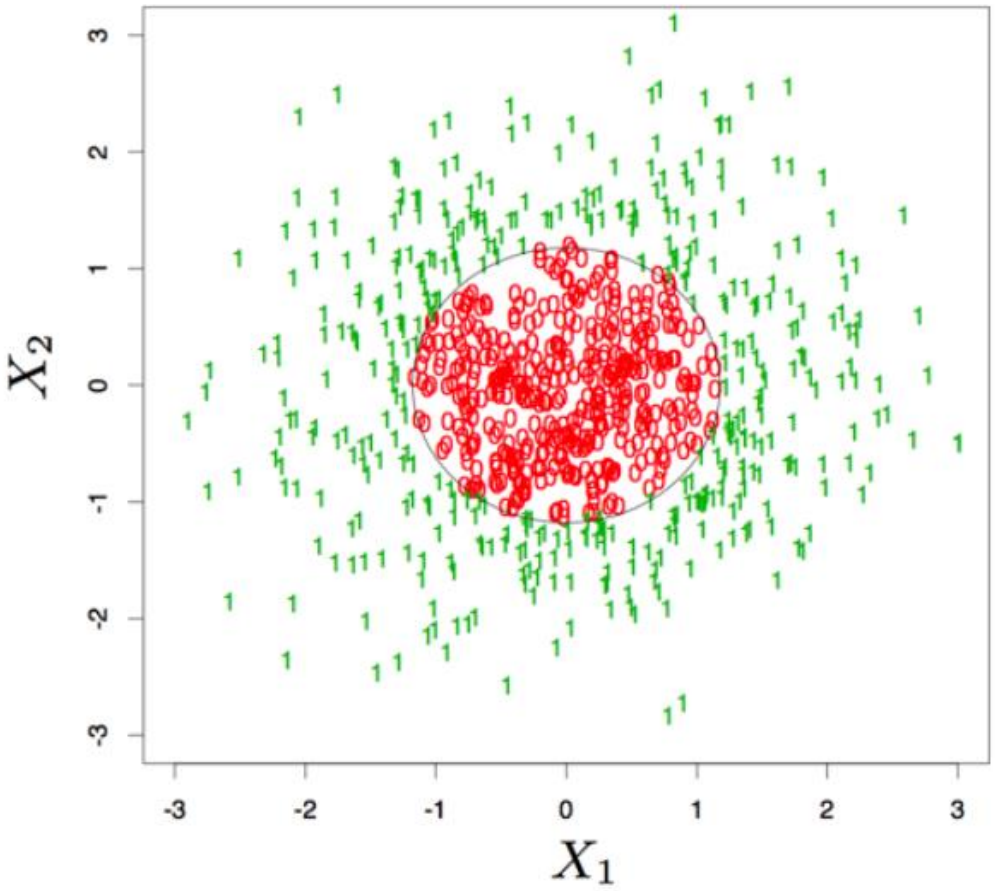
Weakest Link Pruning:

1. Starting with the full tree, successively collapse the internal node that produces the smallest per node increase in $\sum_m N_m Q_m(T)$

   Result: Sequence of trees $T_0, T_1, T_2, \ldots, T_m$ (were T_m is a single-leaf tree)

2. Choose optimal α (the optimal tree T_i) via cross-validation.

Toy Classification Problem - setup



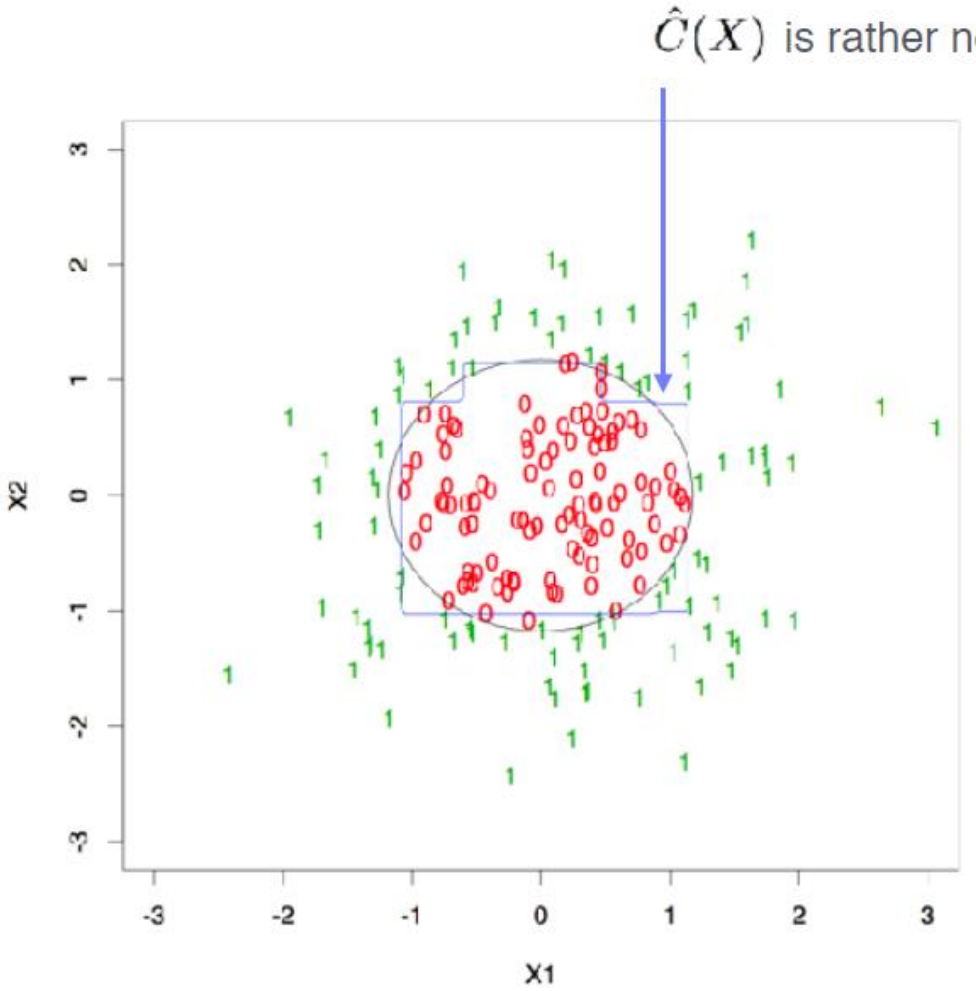
Given: N training pairs (X_i, Y_i)

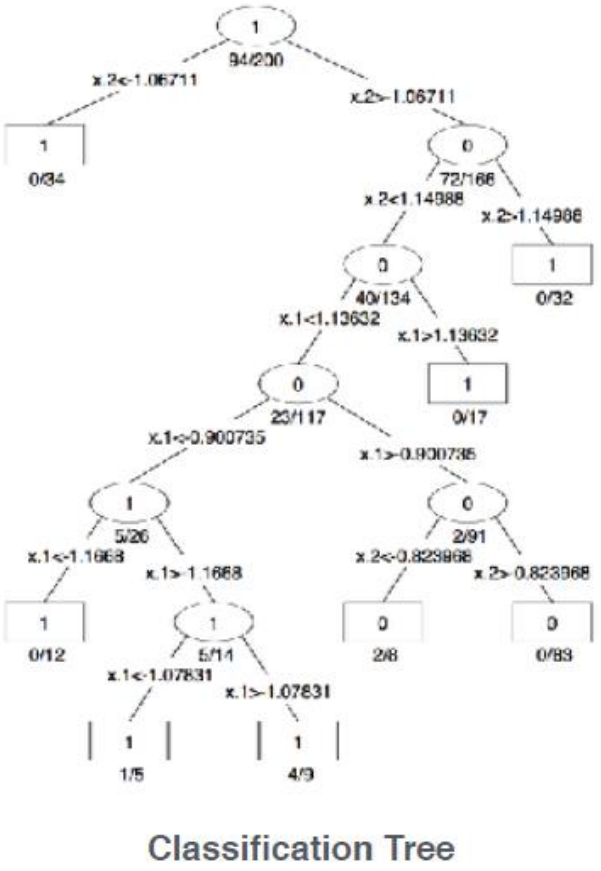
(black line: optimal decision boundary)

Goal: Produce a classifier

$$\hat{C}(X) \in \{-1, 1\}$$

Toy Classification Problem - result



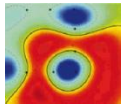


Decision Trees- advantages

- Interpretability
- Generalizes to higher dimensions
- Can handle mixed predictors: quantitative & qualitative
- Easily ignores redundant variables
- Handles missing data elegantly

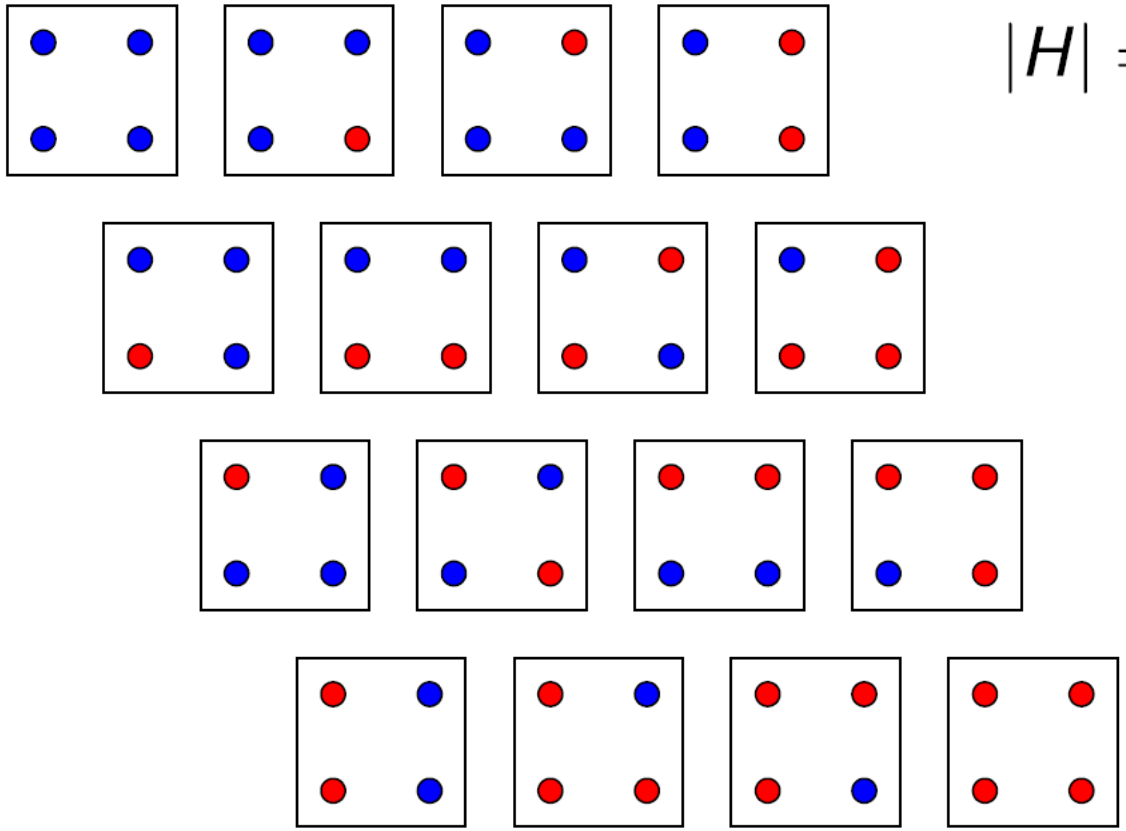
BUT: decision trees are prone to overfitting (high variance)

倾向于



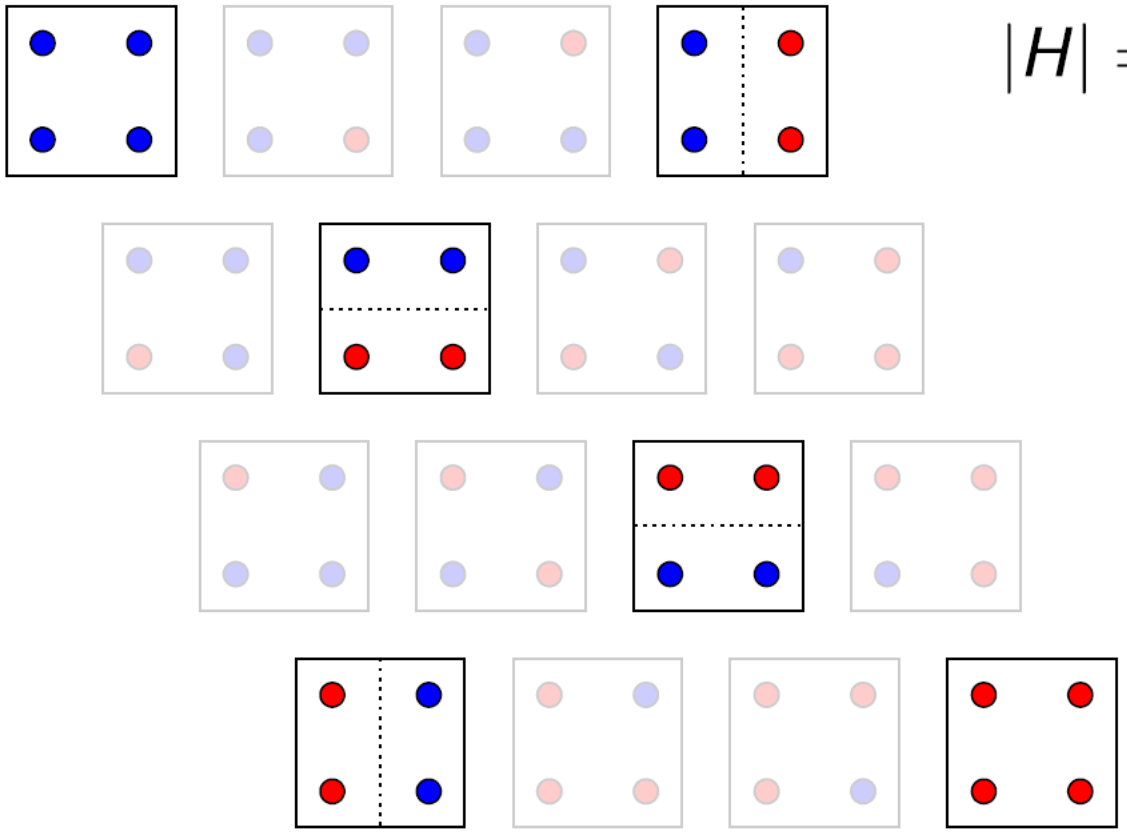
learning theory for decision trees

Hypothesis Space – general classifiers



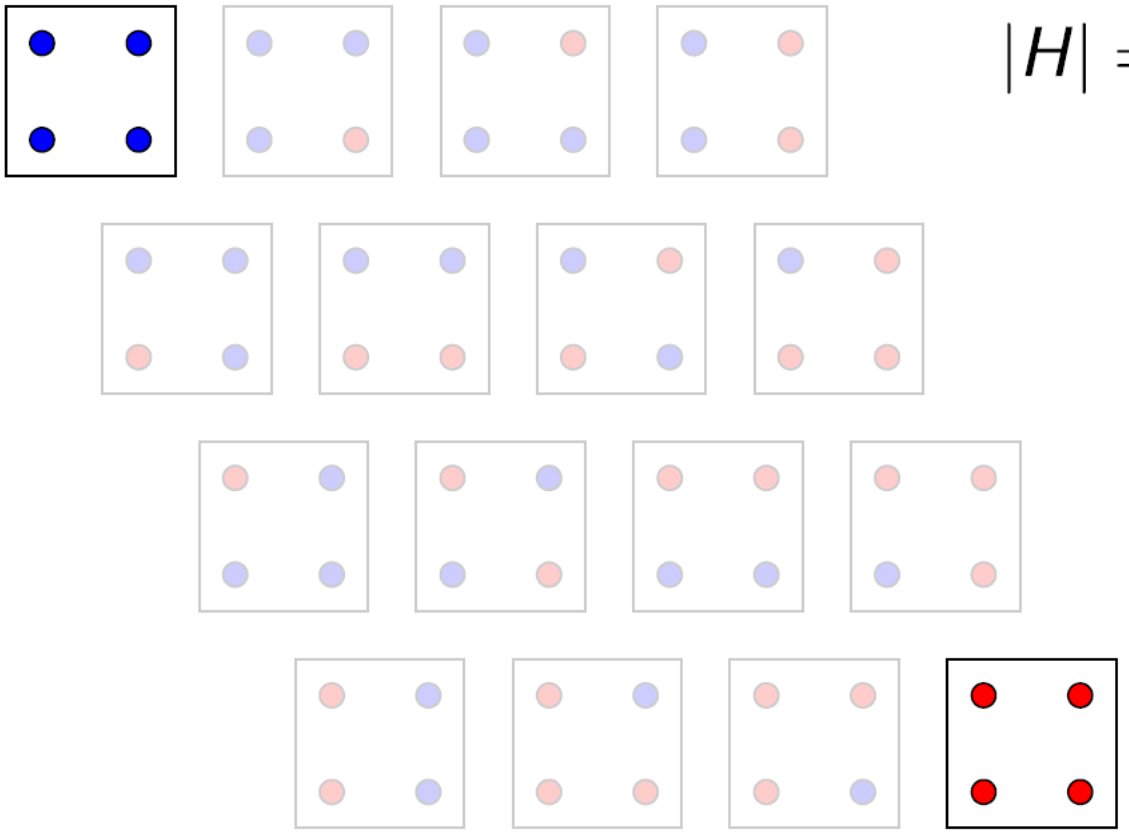
$$|H| = 16$$

Hypothesis Space – decision trees (level 1)



$$|H| = 6$$

Hypothesis Space – constant classifiers



$$|H| = 2$$

Bounding Generalization Error

[Haussler'88]: When the hypothesis space H is finite, then, the generalization error of a consistent (i.e. correctly classifying) hypothesis h can be bounded as:

$$P(\text{error}_{\mathcal{X}}(h) > \varepsilon) \leq |H| e^{-m\varepsilon}$$

... where $|H|$ is the size of the hypothesis space, and m is the number of iid samples in the training set.

Observation: generalization error grows with the number of hypotheses and decreases with the number of data points.

Bounding Generalization Error

If we take the bound from the previous slide and consider the probability to be constant

$$\underbrace{P(\text{error}_{\mathcal{X}}(h) > \varepsilon)}_{\delta} \leq |H| e^{-m\varepsilon}$$

then, we can rewrite the bound in a more convenient way as

$$\text{error}_{\mathcal{X}}(h) \leq \frac{\log |H| + \log \frac{1}{\delta}}{m}$$

i.e. generalization error grows with model complexity $|H|$ and decreases with the training set size m.

Bounding Generalization Error

Limitation of Haussler's bound:

- It applies only to set of hypotheses that perfectly classify the data. What if we use a simple model (e.g. shallow decision tree or linear classifier?)

Another bound (derived from the Chernoff inequality) for the generalization error is given by:

$$\text{error}_{\mathcal{X}}(h) \leq \underbrace{\text{error}_{\mathcal{D}}(h)}_{\text{training error}} + \underbrace{\sqrt{\frac{\log|H| + \log\frac{1}{\delta}}{2m}}}_{\text{error due to model complexity}}$$

training error ~ **bias**

error due to model complexity ~ **variance**

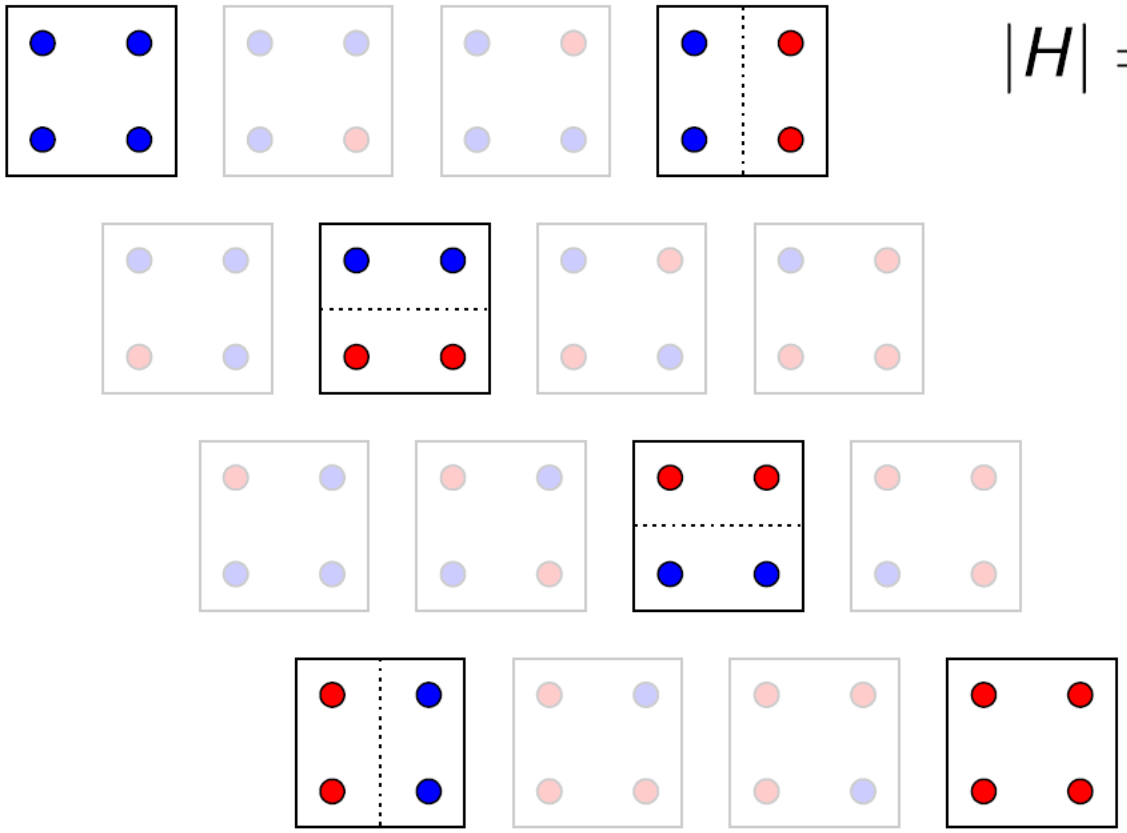
Application to Decision Trees

Question: Can we bound the error of decision trees?

How large is the hypothesis space of decision trees?

$$\text{error}_{\mathcal{X}}(h) \leq \underbrace{\text{error}_{\mathcal{D}}(h)}_{\sim \text{ bias}} + \underbrace{\sqrt{\frac{\log |H| + \log \frac{1}{\delta}}{2m}}}_{\sim \text{ variance}}$$

Hypothesis Space – decision trees (level 1)



$$|H| = 6$$

Complexity of Decision Trees

- **Example 1:** General decision tree with n binary features (i.e. expanded until it perfectly classifies data):

$$|H| = 2^{2^n}$$

i.e. all possible lookup tables over n binary features.

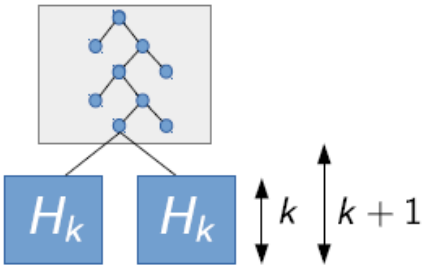
- **Example 2:** Decision tree of depth 0:

$$|H| = 2$$

H = {always labeling "0", always labeling "1"}

Complexity of Decision Trees

Solution for trees of depth k: Use induction



- Initial condition: $|H_0| = 2$
- Recursion: $|H_{k+1}| = n \times |H_k| \times |H_k|$

choice of root attribute

left tree

right tree

- Result $\log_2 |H_k| = (2^k - 1)(1 + \log_2 n) + 1$

Wrap-up

How large is the hypothesis space of decision trees?

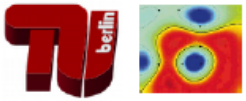
$$\text{error}_{\mathcal{X}}(h) \leq \underbrace{\text{error}_{\mathcal{D}}(h)}_{\sim \text{ bias}} + \underbrace{\sqrt{\frac{\log |H| + \log \frac{1}{\delta}}{2m}}}_{\sim \text{ variance}}$$

Decision tree:

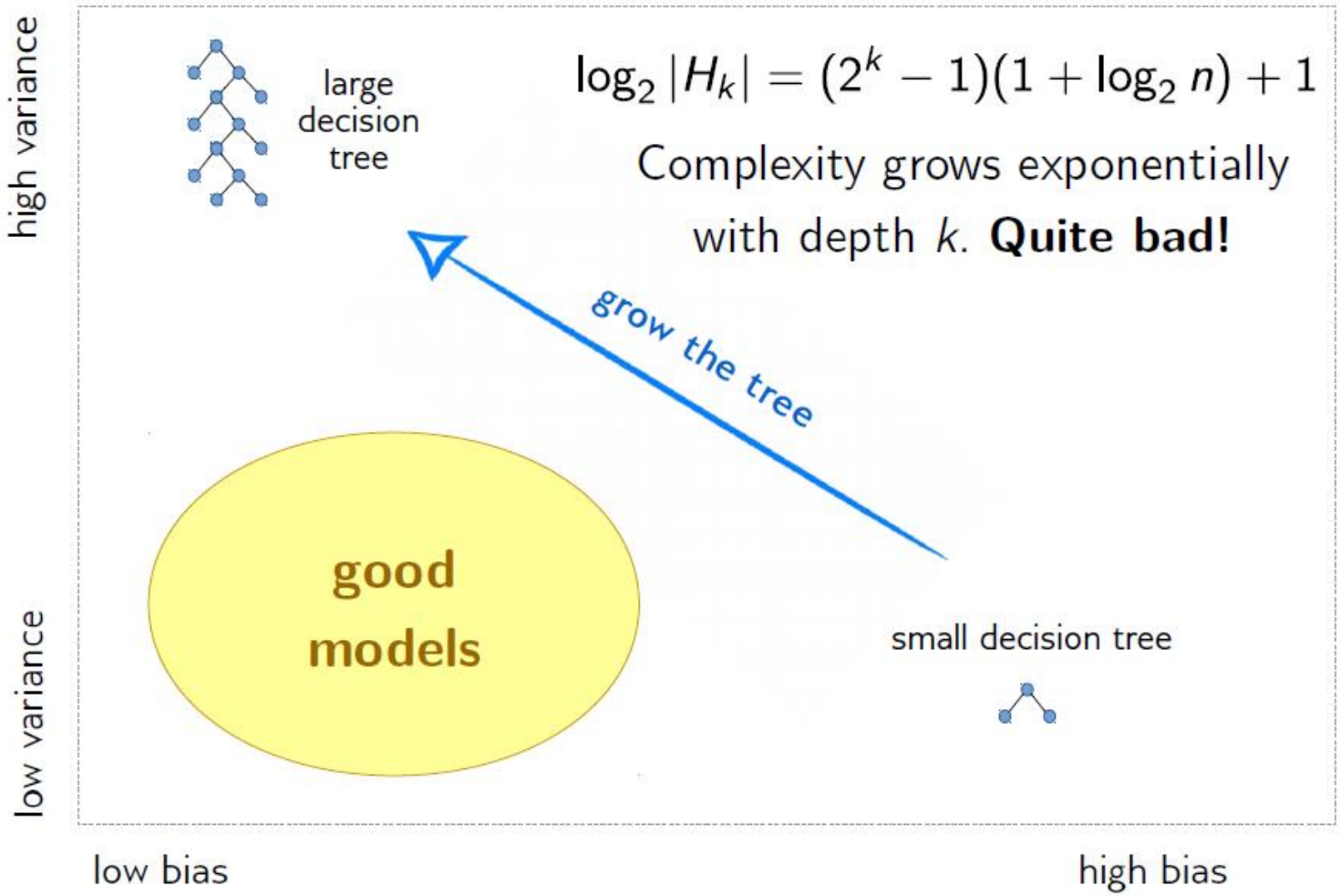
- Depth 0 (constant labeling) $\log_2 |H_0| = 1$
- Depth k $\log_2 |H_k| = (2^k - 1)(1 + \log_2 n) + 1$
- Unrestricted $\log_2 |H| = 2^n$

high bias / low variance

low bias / high variance



Decision Trees and Bias Variance



→ **We need better capacity control.**

Idea: combine decision trees with averaging, boosting.

DT- regularization via model averaging (ensembles)

Generalization error: $\mathrm{E}\left[\left(y-\hat{f}(x)\right)^2\right] = \mathrm{Bias}\left[\hat{f}(x)\right]^2 + \mathrm{Var}\left[\hat{f}(x)\right] + \sigma^2$

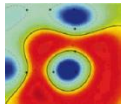
bias: low (if sufficiently deep)

\+ **variance:** high (sensitive to choice of splits)

\+ **residual error**

Consequence: decision trees often produce noisy or weak classifiers!

Idea: Combine the predictions of several randomized trees into single model.



Toy Classification Problem

1. **自助抽样（Bootstrap Sampling）**
   - 从原始训练集 S 通过 **有放回** 的方式随机抽取多个大小为 N 的子样本集 $S^*1, S^*2, ..., S^*B$（即 bootstrap 采样）。
   - 由于是有放回抽样，部分数据可能会被重复选中，而部分数据可能不会被选中。
2. **训练多个模型**
   - 对每个采样的数据集 S^*b 训练一个弱模型（通常是决策树）。
   - 由于不同的样本集包含不同的训练数据，每个模型可能学到不同的特征。

Bagging (**B**ootstrap **agg**regat**ing**)

Classifier $C(\mathcal{S}, x)$
training data S

Train B models on different dataset:

Draw $\mathcal{S}^{*1}, \ldots \mathcal{S}^{*B}$
samples sets of length *N*
(bootstrap samples: duplicates allowed!)

Prediction:

$\hat{C}_{bag}(x) = \text{Majority Vote } \{C(\mathcal{S}^{*b}, x)\}_{b=1}^{B}$

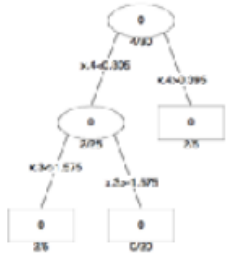
(in classification; regression: avergage)

Original Tree

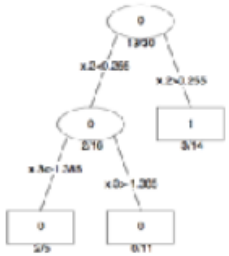
Bootstrap Tree 1

Bootstrap Tree 2

Bootstrap Tree 3

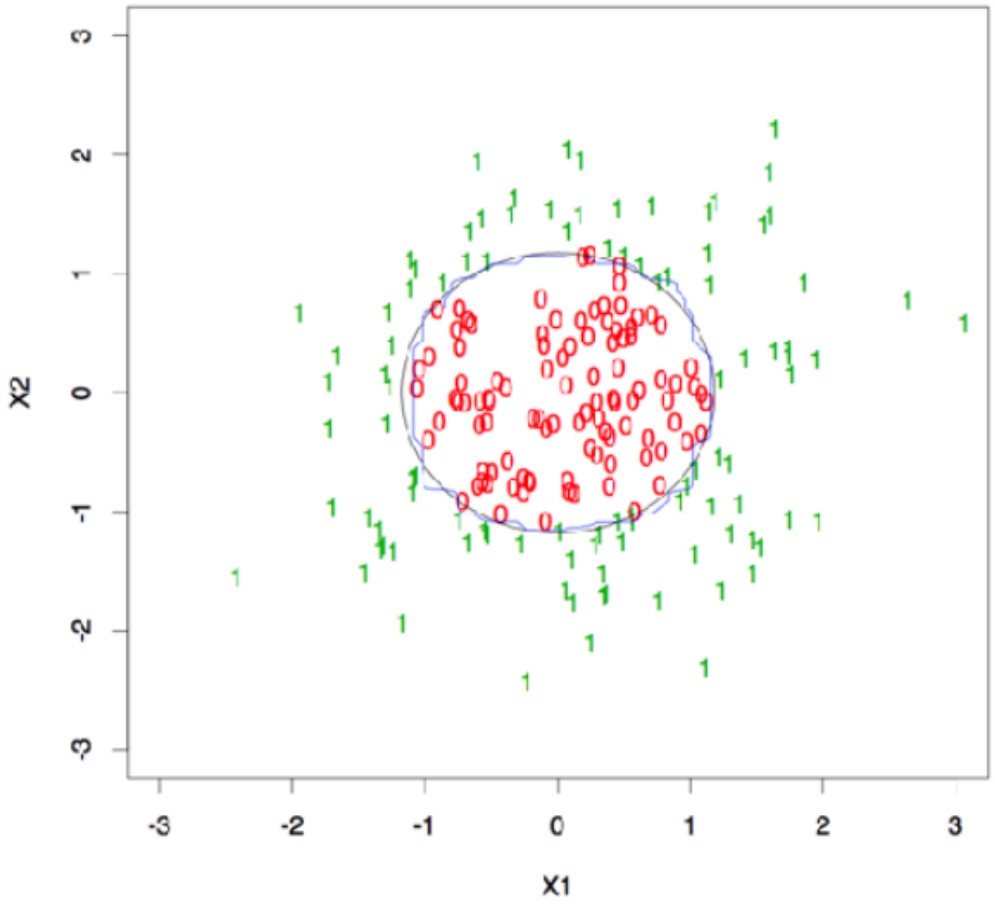


Bootstrap Tree 4



Bootstrap Tree 5

Toy Classification Problem - bagging



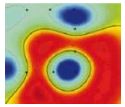
Improved prediction via reduced variance

Bagging averages trees for smoother decision boundaries.

Bagging - limitations

- **Datasets highly overlapping:**
  Predictions of different trees become highly correlated.

- **Variance not reduced as much as desired!**



Random Forrests - overview

Random Subspace Method (Tin Kam Ho, 1998):
Randomly sample features $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ without replacement

➡ De-correlates estimators and decreases variance of the aggregate.

Random Forest (Breiman, 1999):
Combination of Bagging + Random Subspace Method applied to decision trees

- Complexity control through "out-of-bag" control
  (estimate generalization error using untrained samples)
- Random feature selection either at tree or split level

Random Subspace Method（随机子空间方法）（Tin Kam Ho, 1998）：

- 在给定特征向量 $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ 的情况下，从全部特征中不放回随机抽取一部分特征进行建模。
- 此方法可以降低不同模型（基学习器）之间的相关性，并减少整体模型预测结果的方差。

Random Forest（随机森林）（Breiman, 1999）：

- 随机森林是将 Bagging（自助法）与 Random Subspace Method（随机子空间方法）应用于决策树的一种集成模型。
- 它通过"袋外样本（out-of-bag）"来进行复杂度控制，这是一种无需单独验证集即可对模型泛化误差进行估计的方法。
- 随机森林在建立每棵树时都会进行随机特征选择，无论是在整棵树层面还是在每一次划分（split）时。

Random Forrests - algorith

对 $b = 1$ 到 B（即构建 B 棵树）重复以下步骤：

1. 从训练数据中采用有放回的随机抽样（bootstrap sample）方法抽取大小为 N 的样本集。
2. 基于此自助样本集生长一棵决策树 T_b，直到节点达到最小大小为止。具体步骤：
   - 随机选择 m 个特征作为候选特征子集。
   - 在这 m 个特征中选出最优特征及最优划分点。
   - 根据该划分点将节点分成两个子节点。

For *b = 1* to *B*:

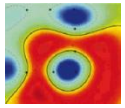
(a) Draw a bootstrap sample (random sampling with replacement) of size *N* from the training data.
(b) Grow a tree T_b to the bootstrapped data until minimum node size reached:
   i. Select *m* features at random.
   ii. Pick best variable/split-point among the *m*.
   iii. Split node into two daughter nodes.

Result: ensemble of *B* trees.

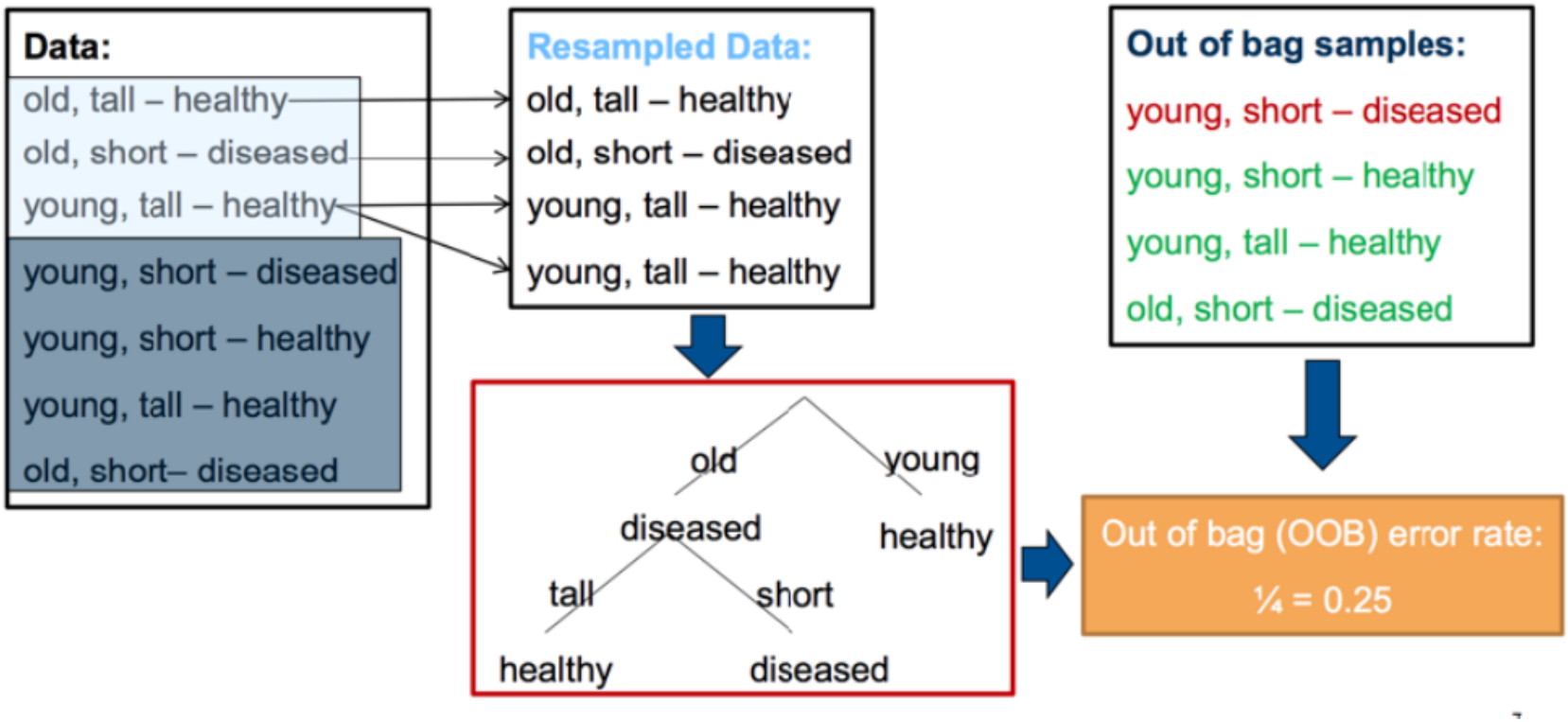
Prediction:

$$\hat{f}_{\mathrm{rf}}^{B}(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \quad \text{(regression)}$$

$$\hat{C}_{\mathrm{rf}}^{B}(x) = majority\ vote\ \{\hat{C}_b(x)\}_1^B \quad \text{(classification)}$$

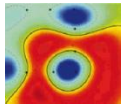


Random Forrests - algorithm



Random Forrests – differences to single tree

- Train each tree on bootstrap resample of data.
- For each split: consider only a subset of *m* randomly selected features. (typically $m = \sqrt{p}$ or $\log_2 p$ were p is the total number of features)
- No pruning.
- Fit *B* trees this way and aggregate predictions.



Random Forrests – differences to single tree

Single Trees

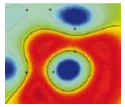
+ yield insight into decision rules
+ rather fast
+ easy to tune parameters

- predictions with high variance

Random Forests

+ smaller prediction variance: better performance
+ easy to tune parameters

- slower
- “black box”



Boosting - conceptually

Boosting 的基本思想：
给定一个弱学习器，将其在（重新加权的）训练数据上多次运行，然后对得到的多个分类器进行投票。通过反复迭代训练弱分类器，并根据其错误情况调整样本权重的方式，使后来的分类器在困难样本上更加用力，从而提升整体分类器的性能。

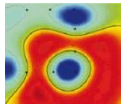
在每次迭代中：

像 **Bagging** 一样，从数据中有放回抽样一批观测值。
与 **Bagging** 不同的是，这里的采样并非随机均匀抽样，而是根据样本权重进行有偏抽样。较难分类的样本会被赋予更高的权重，因此在下一次迭代中被抽到的概率更大。
每个训练样本的权重由其分类困难度决定。分类越困难（被误分类越多），该样本的权重就越高。

Idea: given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote.

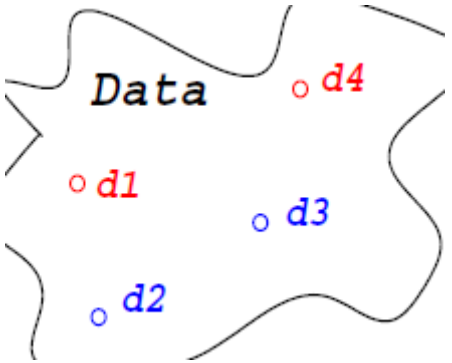
On each iteration:

- Like in bagging, draw a sample of the observations from data (with replacement).
- Unlike in bagging, observations are not sampled randomly
  Higher weight observations are more likely chosen.
- Weight each training example by how incorrectly it was classified.
  Weight is higher for examples that are harder to classify.



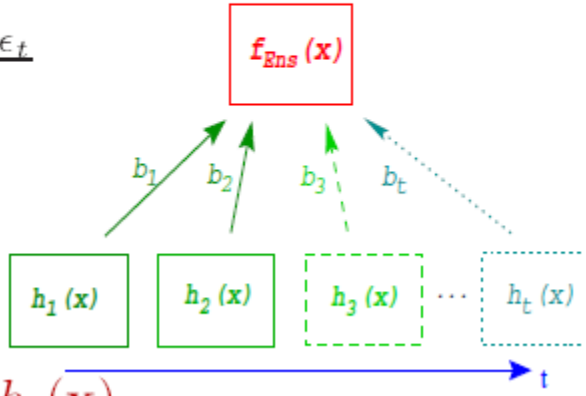
The Adaboost Algorithm

Input: N examples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$
Initialize: $d_i^{(1)} = 1/N$ for all $i = 1 \ldots N$
Do for $t = 1, \ldots, T$,



1. Train base learner according to example distribution $\mathbf{d}^{(t)}$ and obtain hypothesis $h_t : \mathbf{x} \mapsto \{\pm 1\}$.
2. compute weighted error $\epsilon_t = \sum_{i=1}^{N} d_i^{(t)} \mathrm{I}(y_i \neq h_t(\mathbf{x}_i))$
3. compute hypothesis weight $\alpha_t = \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
4. update example distribution

$$d_i^{(t+1)} = d_i^{(t)} \exp\left(-\alpha_t y_i h_t(\mathbf{x}_i)\right) / Z_t$$



Output: final hypothesis $f_{\mathrm{Ens}}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$

Gradient Tree Boosting - conceptually

Idea: Iteratively improve a weak learner by correcting it: $F_{m+1}(x) = F_m(x) + h(x)$

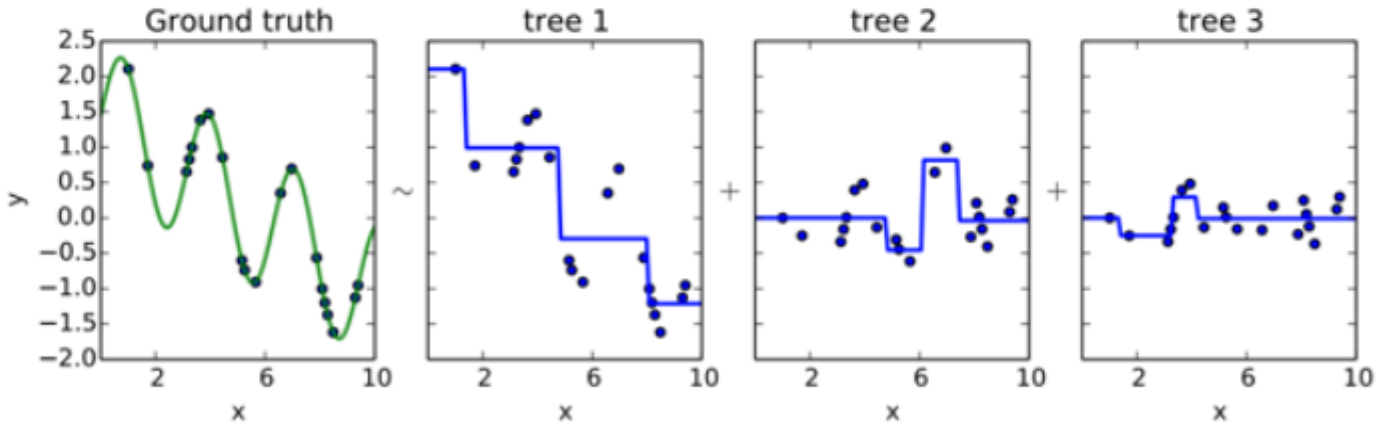
A perfect correction h(x) implies: $F_{m+1}(x) = F_m(x) + h(x) = y$

i.e. $h(x) = y - F_m(x)$

Hence we fit a new tree to the residual: $y - F_m(x)$ → $\frac{1}{2}(y - F(x))^2$

(negative gradient of quadratic loss function)

Residual fitting:



Gradient boosting is a gradient descent algorithm and can be trivially generalized with other loss functions and their gradients.

Decision Trees and Bias Variance