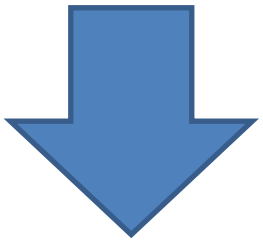
はじめてのパターン認識

第11章 識別器の組み合わせによる性能強化

新納浩幸

ノーフリーランチ定理

すべての識別問題に対して、他の識別器よりも識別性能がよい識別器は存在しない



完璧な識別器は存在しない

NFLを示す簡単な例

学習データ				テストデータ				
x	t	h_1	h_2	x	t_1	t_2	h_1	h_2
000	−1	−1	−1	011	1	−1	−1	1
001	−1	−1	−1	100	−1	1	−1	1
010	1	1	1	101	1	−1	−1	1
				110	−1	1	−1	1
				111	−1	1	−1	1

入力 X のパターンは8通り

h1 は未知データに対して -1
h2 は未知データに対して 1

左の表では未知データは 5 通り
正解パターンは 2^5 = 32 通り

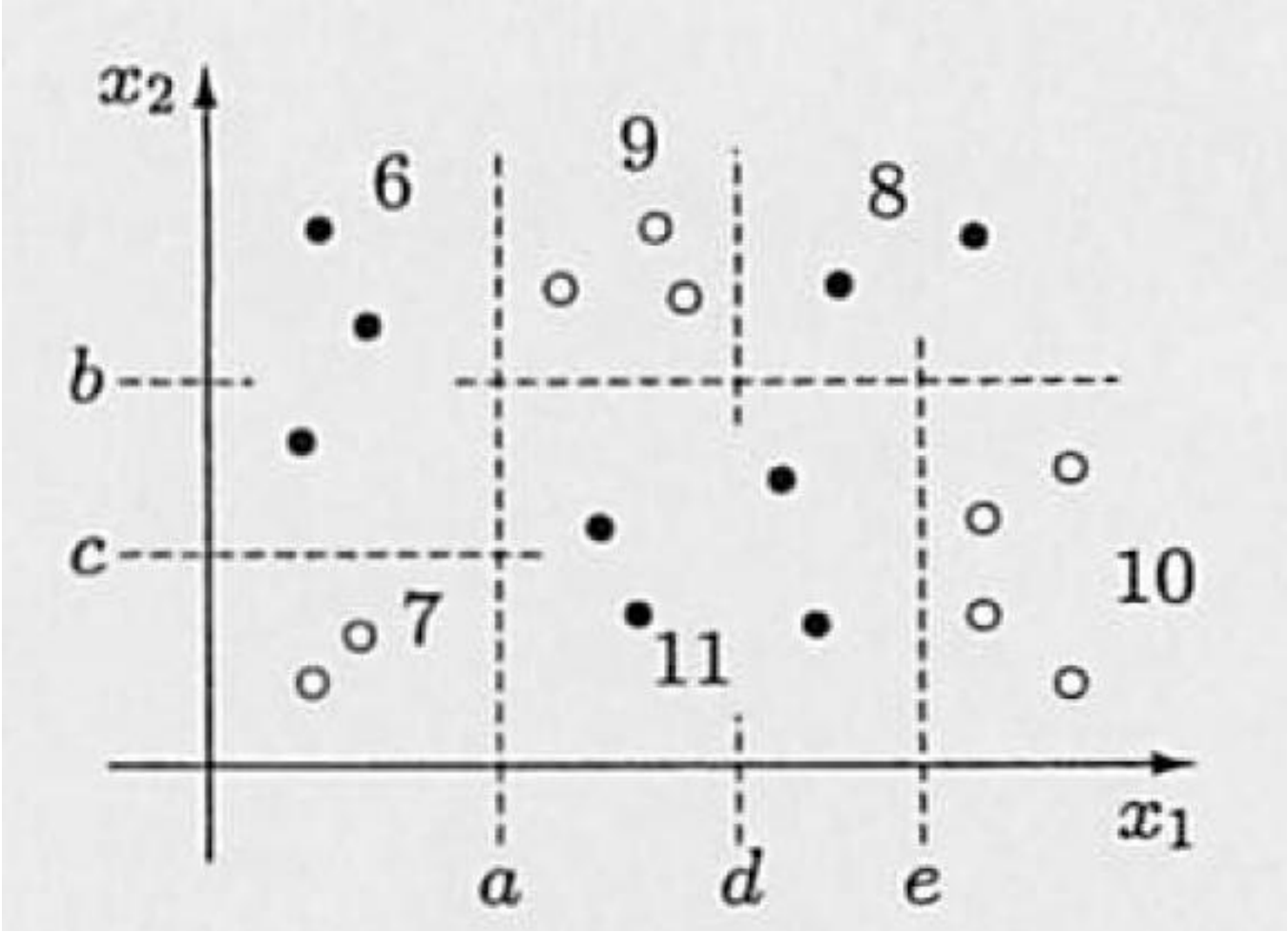
未知データ

正解のパターン1
h1 > h2

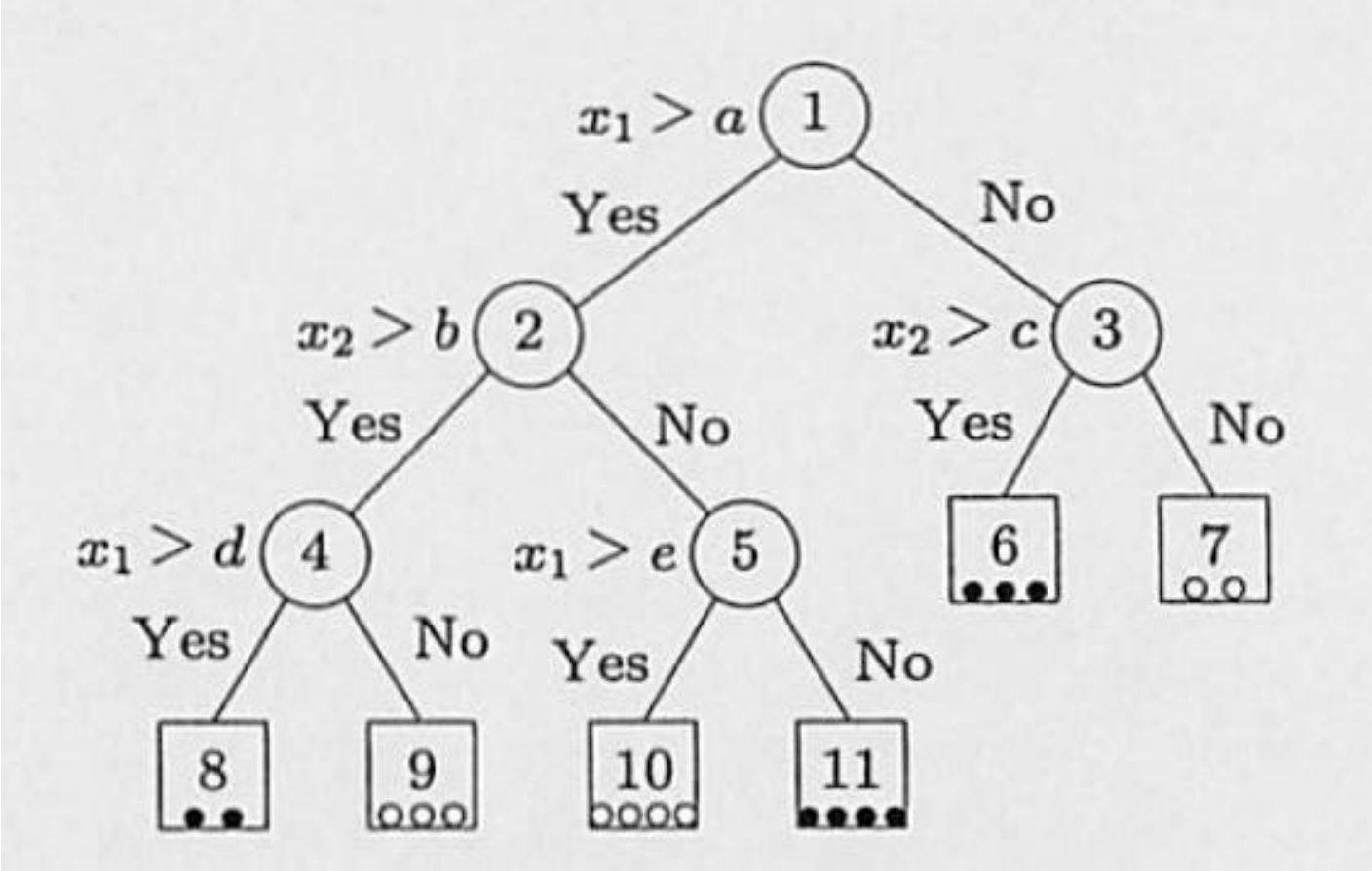
正解のパターン2（パターン1の逆）
h1 < h2

すべての正解パターンで識別器の性能を測ると全部同じ性能

決定木(1)



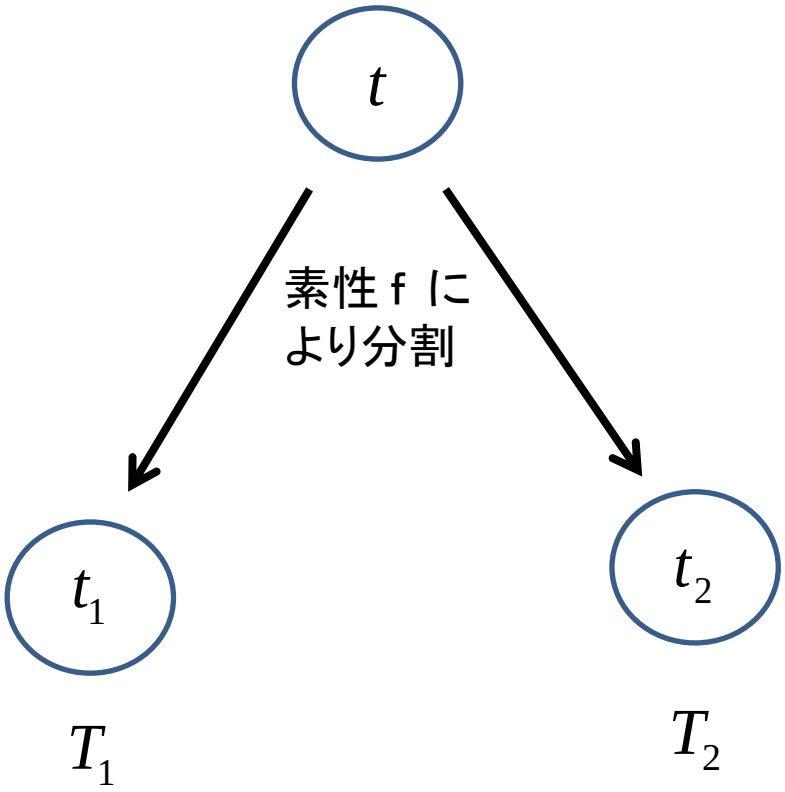
決定木(2)


$x_1 > a$
1
Yes
No
$x_2 > b$
2
$x_2 > c$
3
Yes
No
Yes
No
$x_1 > d$
4
$x_1 > e$
5
6
7
Yes
No
Yes
No
8
9
10
11

決定木の構築

どの素性で分割するかがポイント

$T = \{d_1, d_2, \cdots, d_N\}$



$T_1 \cup T_2 = T$

$T_1 \cap T_2 = \phi$

不純度　$I(t)$

$I(t) - E(I(t_i))$

が最大になるような素性 f で分割

Gini index（ジニ係数）

不純度として使われる代表的関数

$$I(t) = \sum_{j \neq i} P(C_i \mid t) P(C_j \mid t)$$

$$= 1 - \sum_{i=1}^{K} P^2(C_i \mid t)$$

ジニ係数の計算例（1）

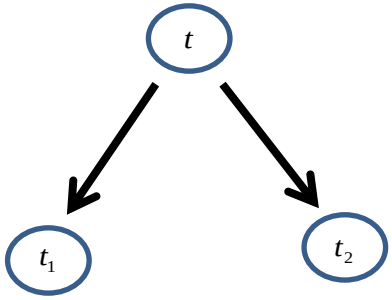
Iris データ
　データ数は 150個、
　クラスは setosa, virginica, versicolor 各 50 個
　データ4次元、花びらの長さ、幅、がく片の長さ、幅

クラスは setosa と virginica をまとめる
versicolor (c)　OR　not-versicolor (s+v)
50データ　　100 データ

ルートノード t のジニ係数　$I(t) = \frac{50}{150} \cdot \frac{100}{150} = \frac{2}{9}$

ジニ係数の計算例（2）

「花べんの長さが 2.45 以上」という素性で分割



t1: s+t 50個、 c 50 個
t2: s+t 50個、 c 0 個

$$I(t_1) = \frac{50}{100} \cdot \frac{50}{100} = \frac{1}{4} \qquad I(t_2) = \frac{50}{50} \cdot \frac{0}{50} = 0$$

$$E(I(t_i)) = \frac{100}{150} \cdot \frac{1}{4} + \frac{50}{150} \cdot 0 = \frac{1}{6}$$

$$I(t) - E(I(t_i)) = \frac{2}{9} - \frac{1}{6} = \frac{1}{18} \approx 0.0556$$

木の剪定（せんてい、pruning）

木が大きくなると汎化誤差も大きくなる

不純度が十分小さくなるまで木を成長させ、
次にある許容範囲まで木を剪定する

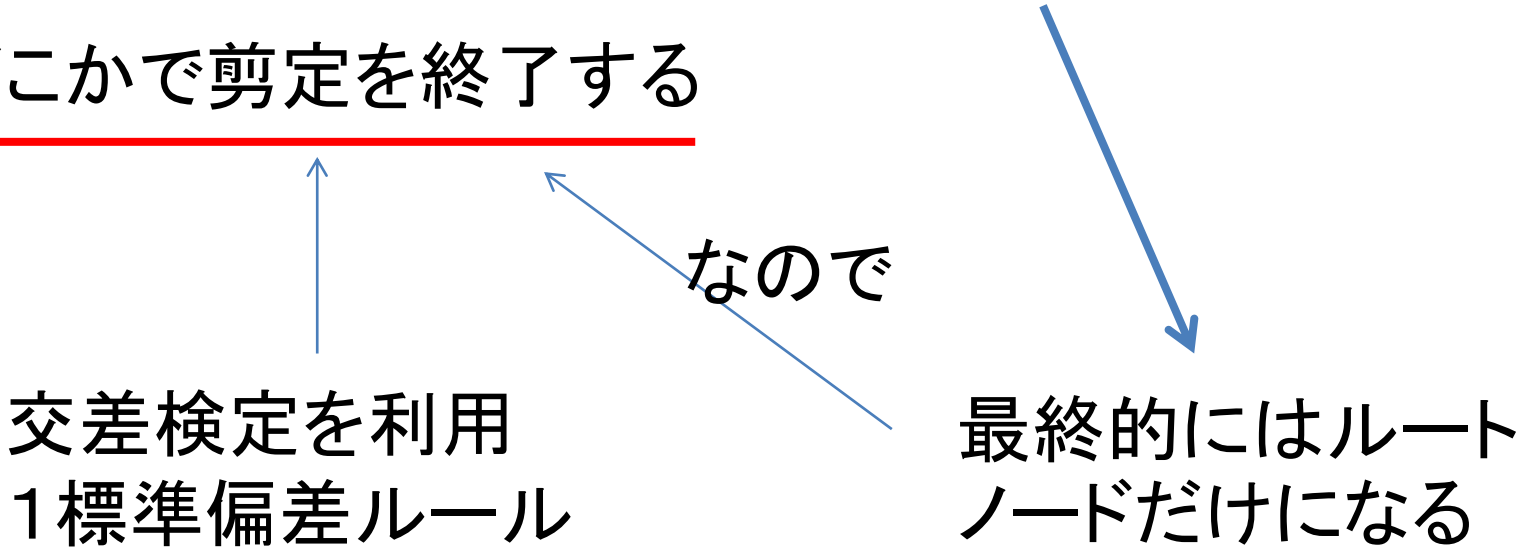
この計算方法がポイント

剪定の手順

(1) 全てのノード t に対してリンクの強さ g(t) を計算

(2) g(t) の最小値を持つノードを終端ノードとし、
その子孫のノードを剪定する、(1) へ

(3) どこかで剪定を終了する



リンクの強さ

$$g(t) = \frac{R(t) - R(T_t)}{|\tilde{T}_t - 1|}$$

$$R(t) = \frac{M(t)}{N}$$

ノード t の誤り数

全データ数

$$T_t = \{t_1, t_2, \cdots, t_K\}$$

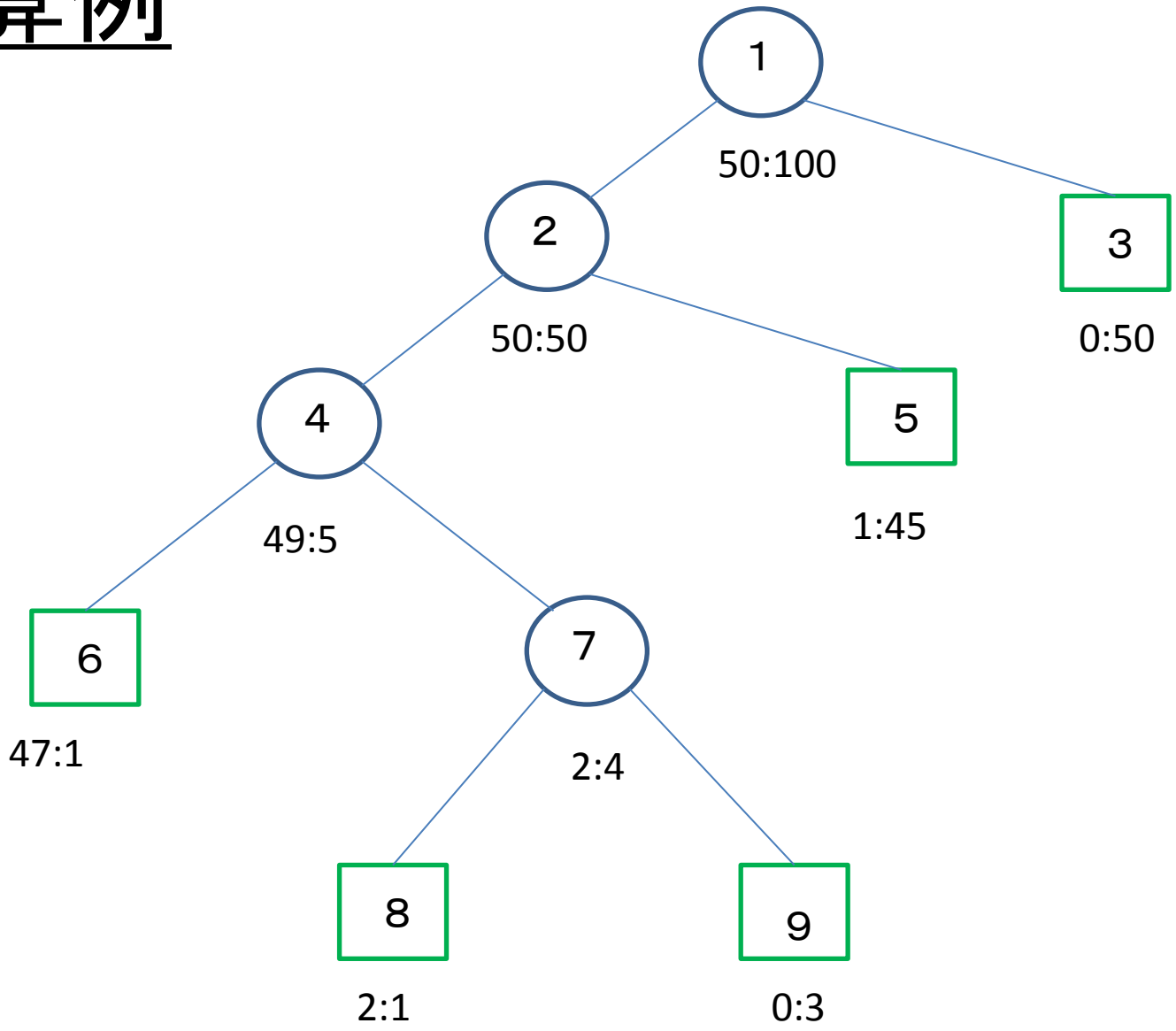
ノード t の子ノードの中の
葉ノードの集合

$$R(T_t) = \sum_{i=1}^{K} R(t_i)$$

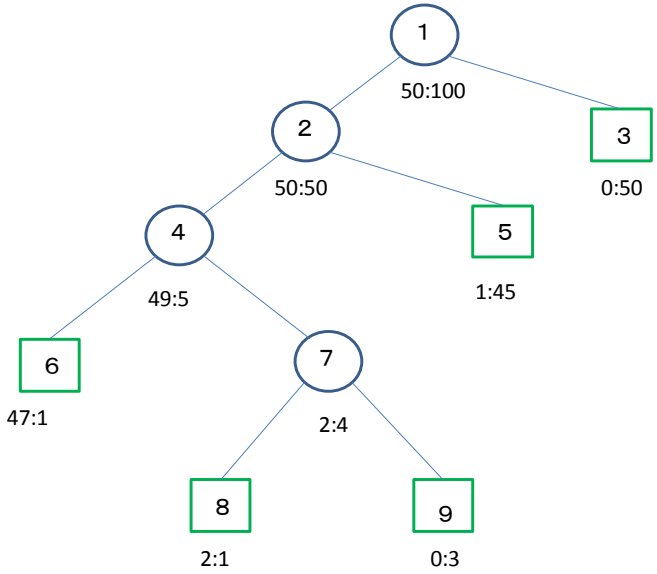
その要素数

$$|\tilde{T}_t| = K$$

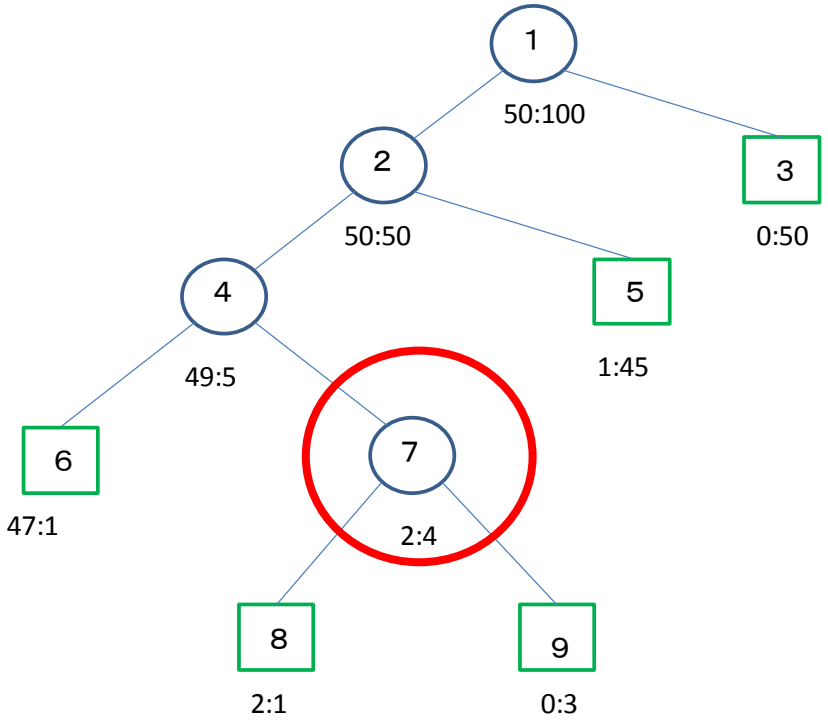
計算例



誤り率は簡単

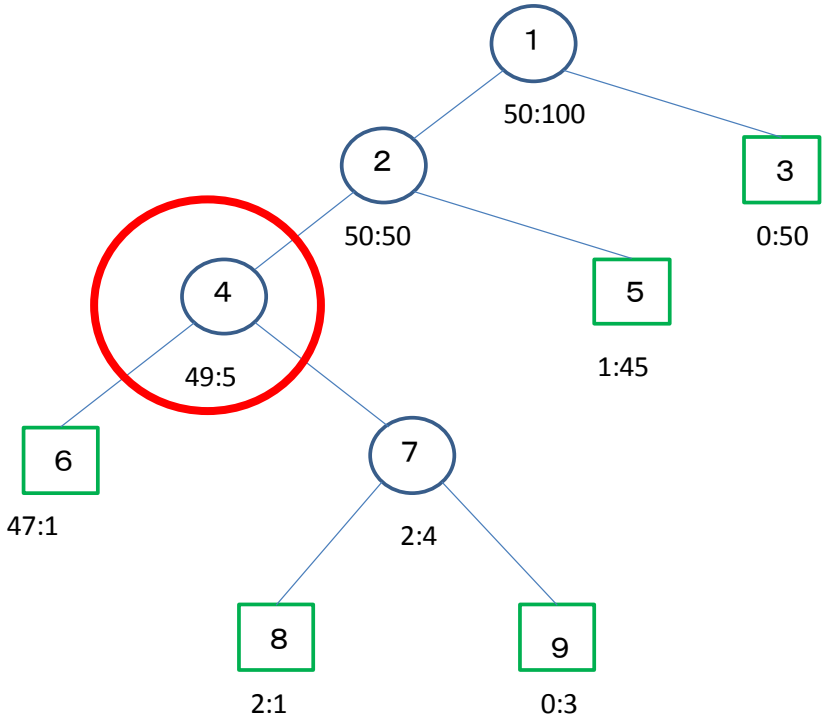


$$R(1) = 50/150 = 0.33$$
$$R(2) = 50/150 = 0.33$$
$$R(3) = 0/150 = 0$$
$$R(4) = 5/150 = 0.033$$
$$R(5) = 1/150 = 0.0067$$
$$R(6) = 1/150 = 0.0067$$
$$R(7) = 2/150 = 0.013$$
$$R(8) = 1/150 = 0.0067$$
$$R(9) = 0/150 = 0$$



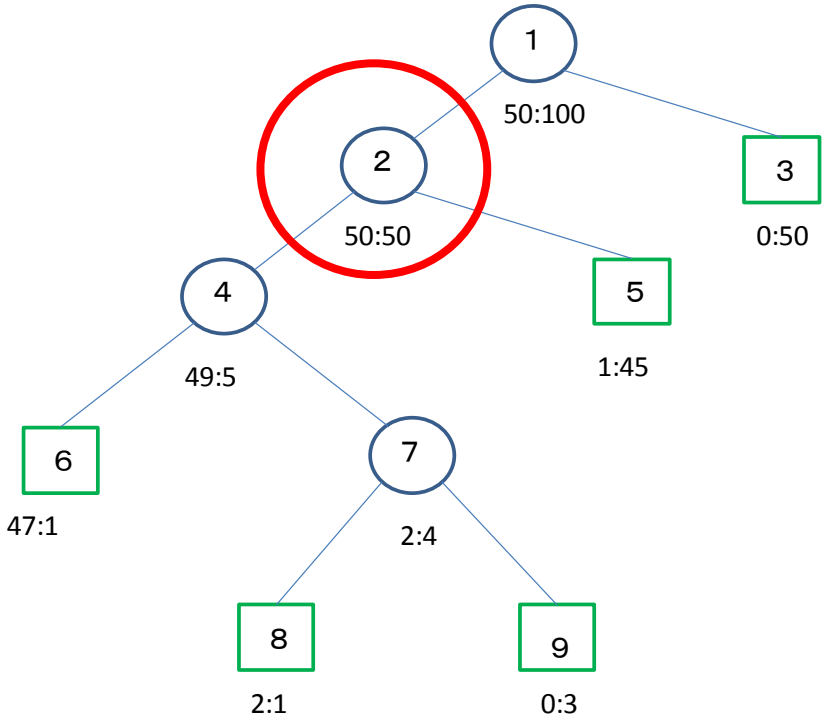
$$T_7 = \{8,9\}$$

$$g(7) = \frac{R(7) - \left(R(8) + R(9)\right)}{2-1} = 0.0067$$



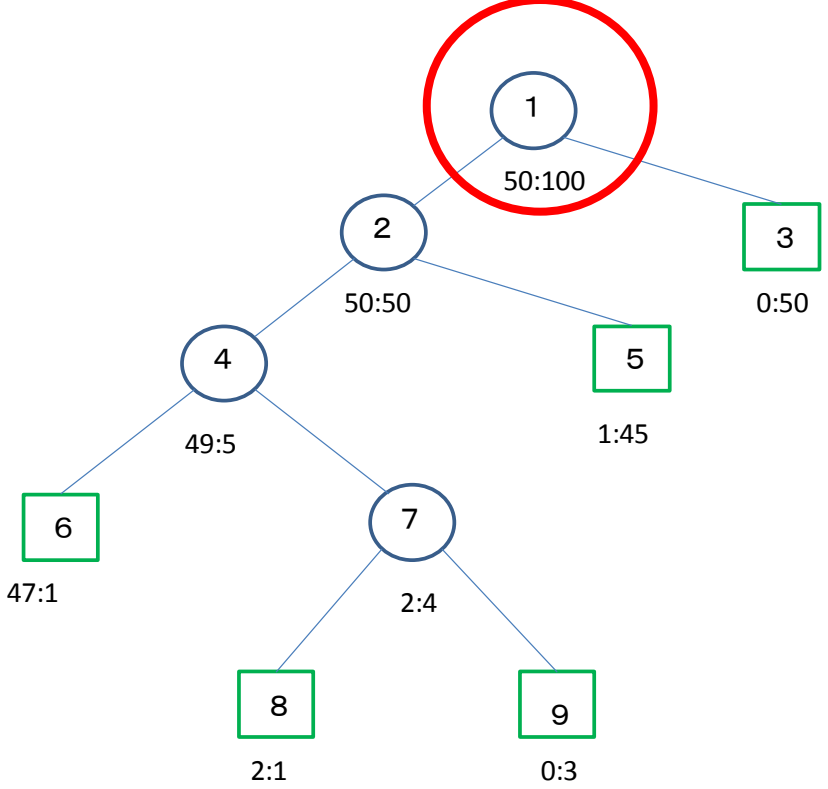
$$T_4 = \{6,8,9\}$$

$$g(4) = \frac{R(4) - \left(R(6) + R(8) + R(9)\right)}{3-1} = 0.01$$



$$T_2 = \{5,6,8,9\}$$

$$g(2) = \frac{R(2) - \left(R(5) + R(6) + R(8) + R(9)\right)}{3-1} = 0.1044$$



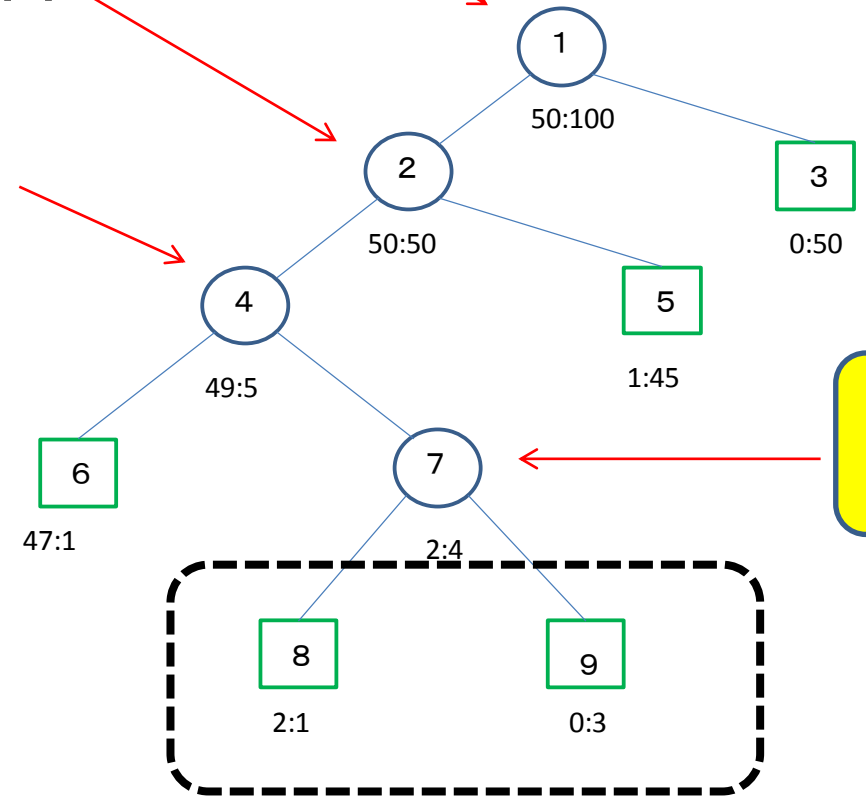
$$T_1 = \{3,5,6,8,9\}$$

$$g(1) = \frac{R(1) - \left(R(3) + R(5) + R(6) + R(8) + R(9)\right)}{5-1} = 0.0783$$

$g(1) = 0.0783$

$g(2) = 0.1044$

$g(4) = 0.01$

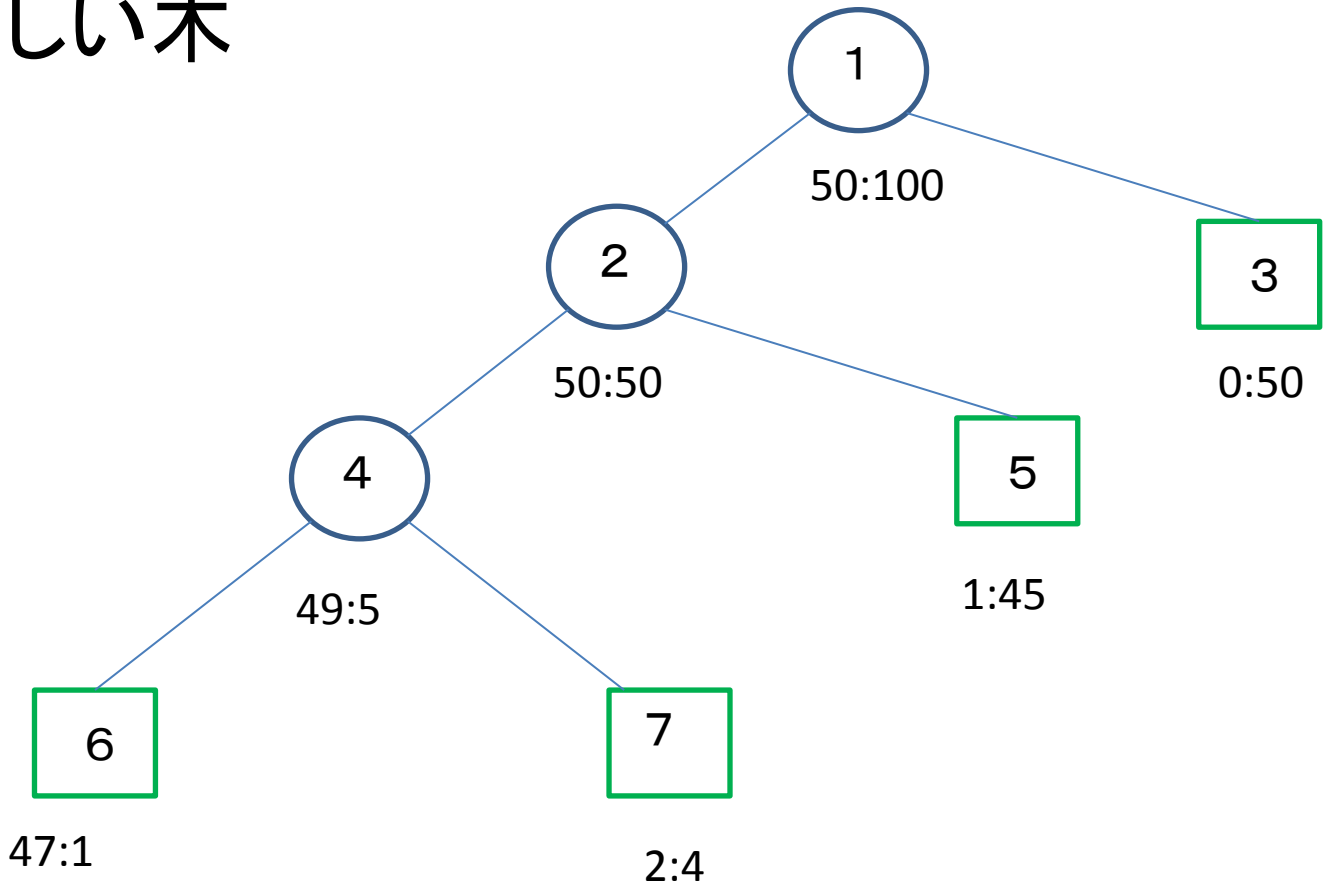


$g(7) = 0.0067$

最小

剪定

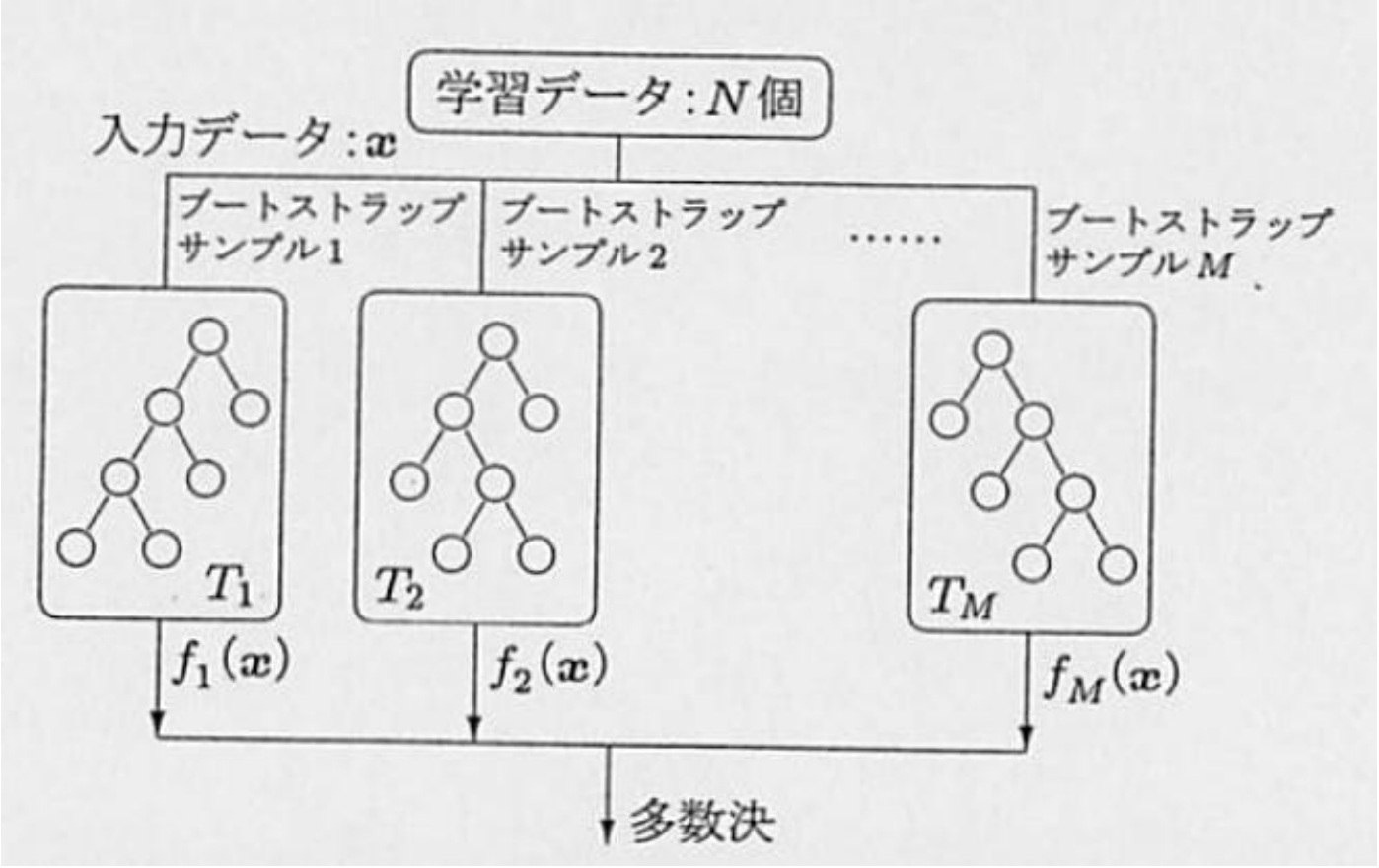
新しい木



リンクの強さの計算を繰り返す

Bagging（バギング）

複数の識別木を組み合わせる方法の1つ



ブートストラップサンプル

N 個のデータから重複を許して
N 個をサンプリング

 新しいデータ  推定

これを何度も繰り返すと、推定したい値の
バラツキがわかる

AdaBoost（アダブースト）

データ D0 から 弱学習器　f0　を学習

F0 で D0 を識別、誤ったデータの重みを大きく、
正しく識別されたデータの重みを小さくした
データ D1 を作成

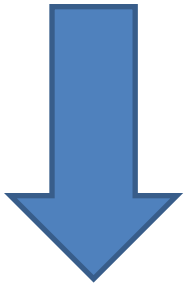
データ D1 から 弱学習器　f1　を学習

繰り返して、f1,f2,・・・,fM を構築、
識別はそれらの重み付き多数決

ランダムフォレスト（１）

バギングの問題

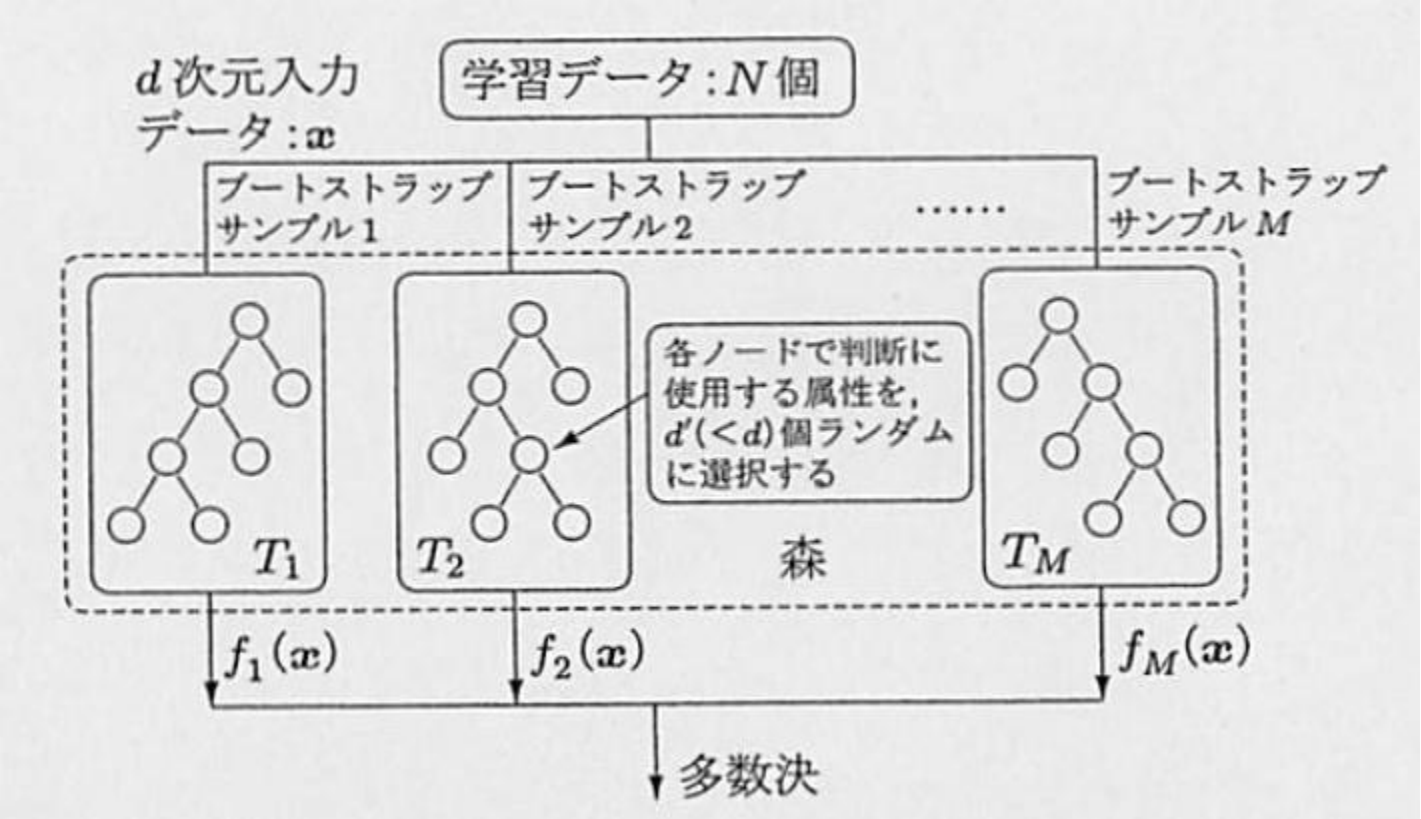
各識別器の相関が高いと性能が強化できない



ランダムフォレスト

決定木のバギングを改良した手法。
決定木の各非終端ノードにおいて識別に用いる
特徴を予め決められた数だけランダムに選択。
これによって相関の低い多様な決定木が生成される

ランダムフォレスト（2）


d次元入力
データ：x
学習データ：N個
ブートストラップ
サンプル1
ブートストラップ
サンプル2
……
ブートストラップ
サンプルM
各ノードで判断に
使用する属性を，
d'(<d)個ランダム
に選択する
T_1
T_2
森
T_M
$f_1(x)$
$f_2(x)$
$f_M(x)$
多数決

randomForest

R のランダムフォレストのパッケージ、非常に有名

```
> library(randomForest)
randomForest 4.6-7
Type rfNews() to see new features/changes/bug fixes.
 警告メッセージ:
 パッケージ ''randomForest'' はバージョン 2.15.3 の R の下で造られました
> train <- iris[sample(c(1:150),20),]
> rf <- randomForest(Species~ .,train)
> ans <- iris[,5]
> kekka <- predict(rf,test)
> sum(ans == kekka)
[1] 145
>
```