

# 実用的な教師あり学習

- ニューラルネットワーク (?)
  - 深層学習への準備
- サポートベクトルマシン
- アンサンブル学習
  - 特にランダムフォレスト
- 回帰
  - 数値を予測する問題

# 6. 識別 - ニューラルネットワーク -

- 識別関数法

- 確率の枠組みにはとらわれず、

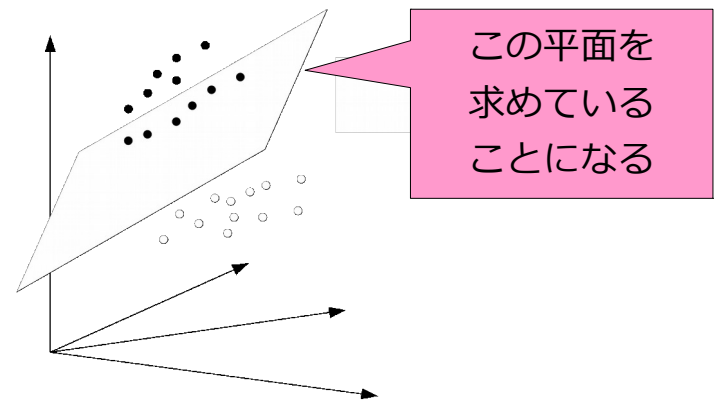
$$f_{Positive}(\boldsymbol{x}) > f_{Negative}(\boldsymbol{x})$$

ならば  $\boldsymbol{x}$  を Positive と判定する関数  $f$  を推定する

- 単層パーセプトロン

- 識別関数として 1 次式 (= 直線・平面) を仮定

$$f(\boldsymbol{x}) = w_0 + \boldsymbol{w} \cdot \boldsymbol{x}$$

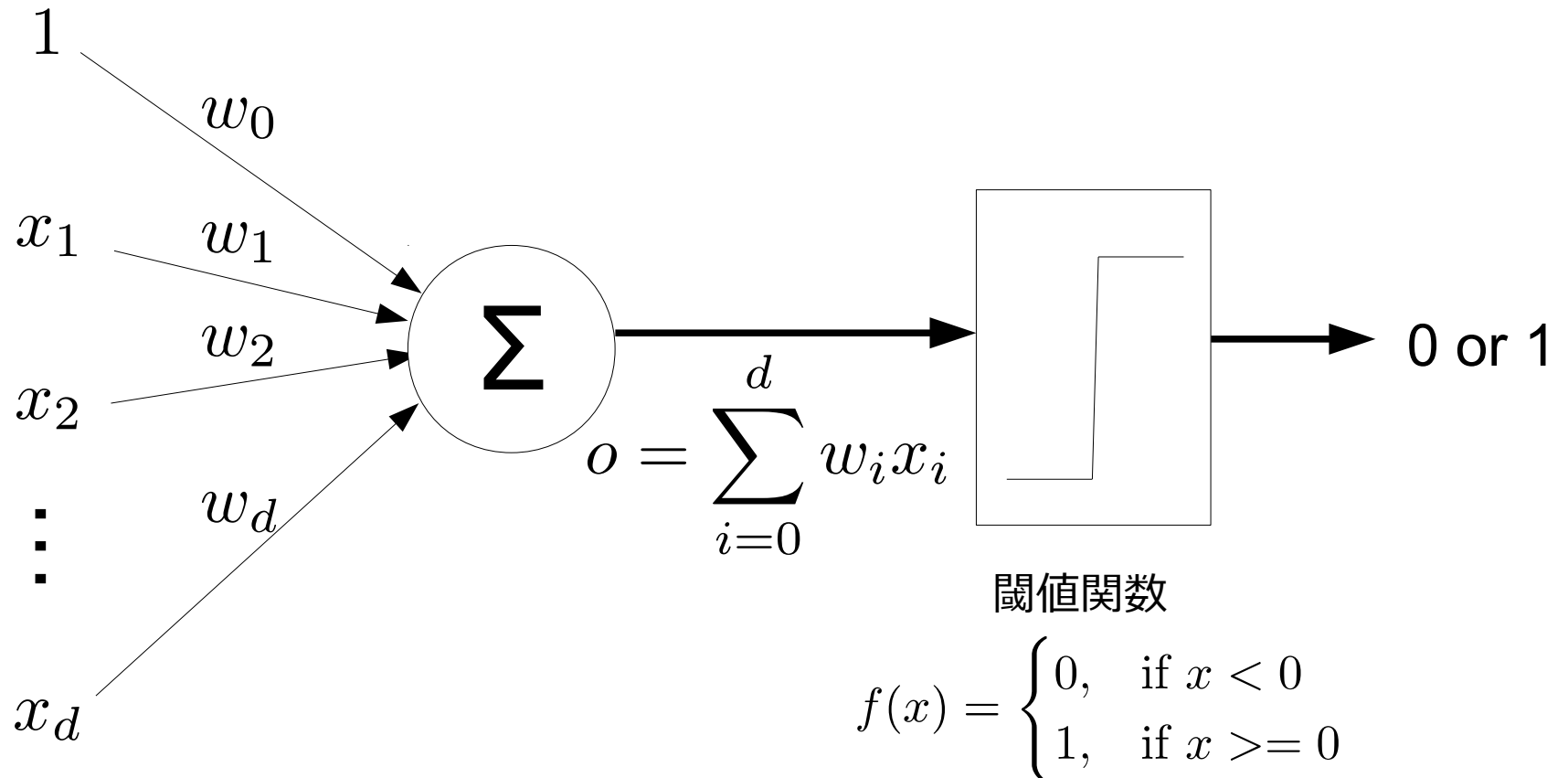


# パーセプトロンの学習

- 単層パーセプトロンの定義

以後、 $w$  は  $w_0$  を含む

- $w \cdot x = 0$  という特徴空間上の超平面を表現



# 最急降下法

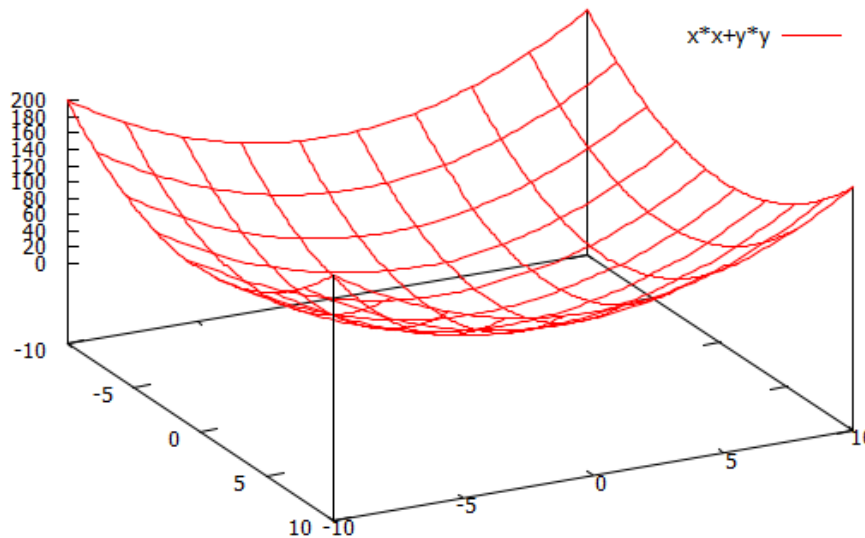
- エラーの定義

- 二乗誤差  $E(\mathbf{w}) \equiv \frac{1}{2} \sum_{x_i \in D} (y_i - o_i)^2$

全データに対する  
正解と関数の出力  
との差の2乗和

- E は w の関数

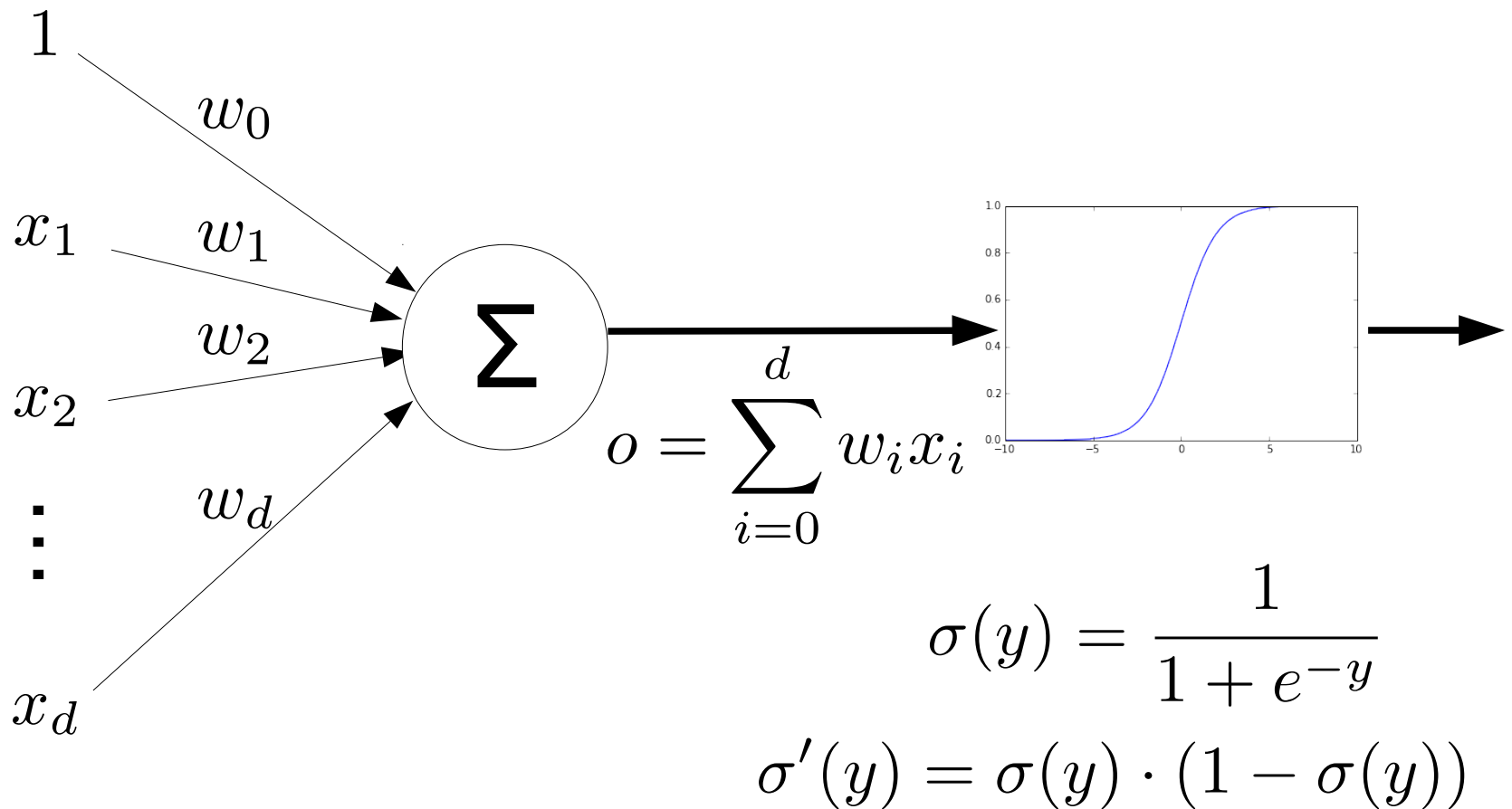
- w を E の勾配方向へ一定量だけ動かすことを繰り返して、最適解へ収束させる (→最急降下法)



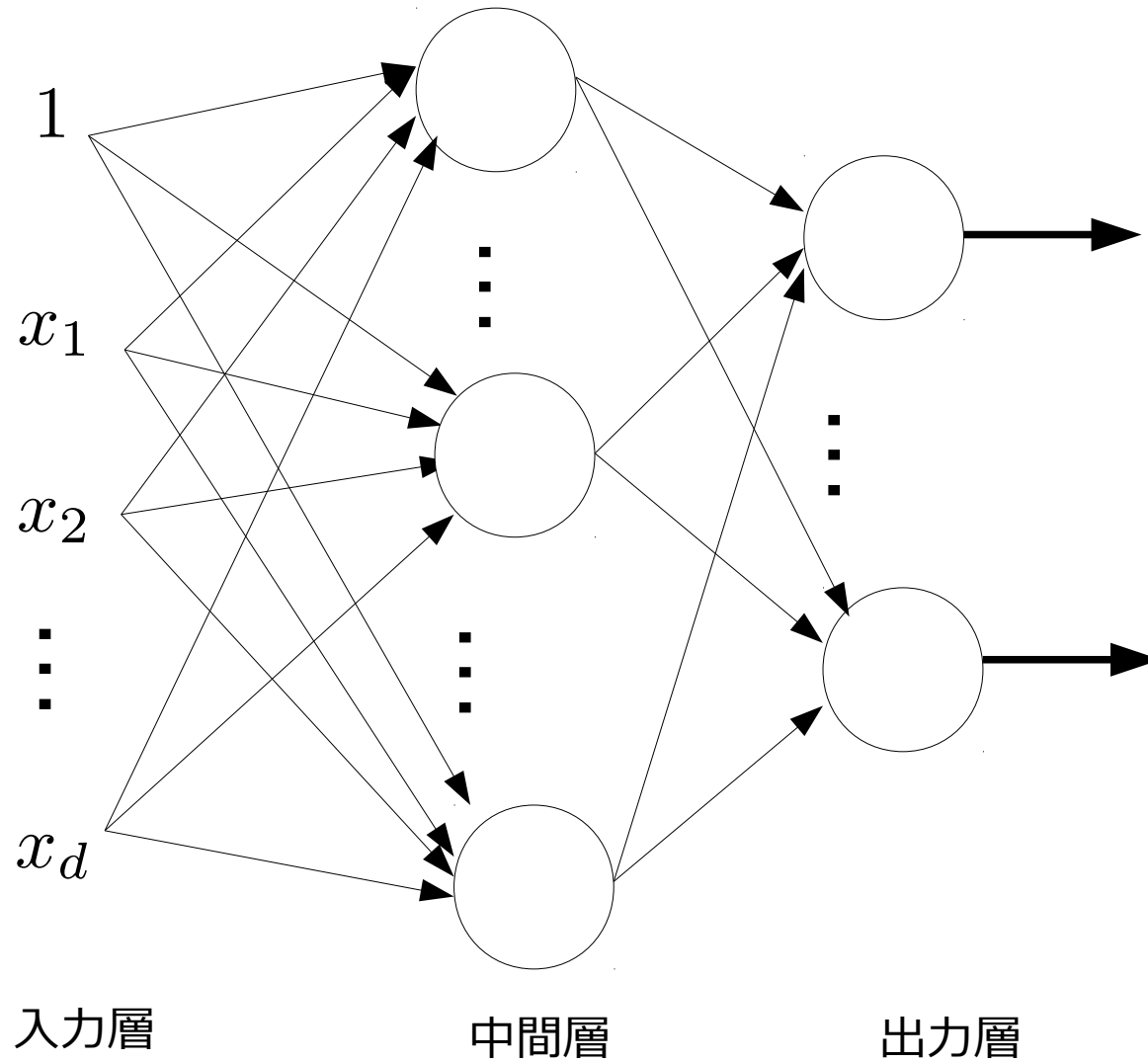
$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$$

# 多層パーセプトロンへの拡張

- シグモイド関数の適用
  - 勾配計算の際に微分可能なものを用いる



# 多層パーセプトロンの構成



# 誤差逆伝播法による学習

1. リンクの重みを小さな初期値に設定

2. 個々の学習データ  $(\mathbf{x}_i, t_i)$  に対して以下繰り返し

a) 入力  $\mathbf{x}_i$  に対するネットワークの出力  $o_i$  を計算

b) 出力層の  $k$  番目のユニットに対してエラー量  $\delta$  計算

$$\delta_k \leftarrow o_k(1 - o_k)(t_k - o_k)$$

c) 中間層の  $h$  番目のユニットに対してエラー量  $\delta$  計算

$$\delta_k \leftarrow o_k(1 - o_k) \sum_{k \in \text{outputs}} w_{kh} \delta_k$$

d) 重みの更新

$$w_{ji} \leftarrow w_{ji} + \eta \delta_j x_{ji}$$

# Weka の MultilayerPerceptron

The image shows two windows from the Weka software. The main window is 'Weka Explorer' and the smaller window is 'weka.gui.GenericObjectEditor'.

**Weka Explorer - Classifier:** MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a -G -R

**Test options:**

- Use training set
- Supplied test set (Set...)
- Cross-validation (Folds: 10)
- Percentage split (%: 66)

**Classifier output:**

```
=== Evaluation on training set ===
Time taken to test model on train
=== Summary ===
Correctly Classified Instances
Incorrectly Classified Instances
Kappa statistic
Mean absolute error
Root mean squared error
Relative absolute error
Root relative squared error
Coverage of cases (0,95 level)
Mean rel. region size (0,95 level)
Total Number of Instances

=== Detailed Accuracy By Class ===
                TP Rate  FP Rate
Weighted Avg.   0,987    0,007

=== Confusion Matrix ===
  a  b  c  <-- classified as
50  0  0  | a = Iris-setosa
 0 49  1  | b = Iris-versicolor
 0  1 49  | c = Iris-virginica
```

**weka.gui.GenericObjectEditor - weka.classifiers.functions.MultilayerPerceptron**

**About:** A Classifier that uses backpropagation to classify instances.

**GUI:** True

**autoBuild:** True

**debug:** False

**decay:** False

**doNotCheckCapabilities:** False

**hiddenLayers:** a

**learningRate:** 0.3

**momentum:** 0.2

**nominalToBinaryFilter:** True

**normalizeAttributes:** True

**normalizeNumericClass:** True

**reset:** False

**seed:** 0

**trainingTime:** 500

**validationSetSize:** 0

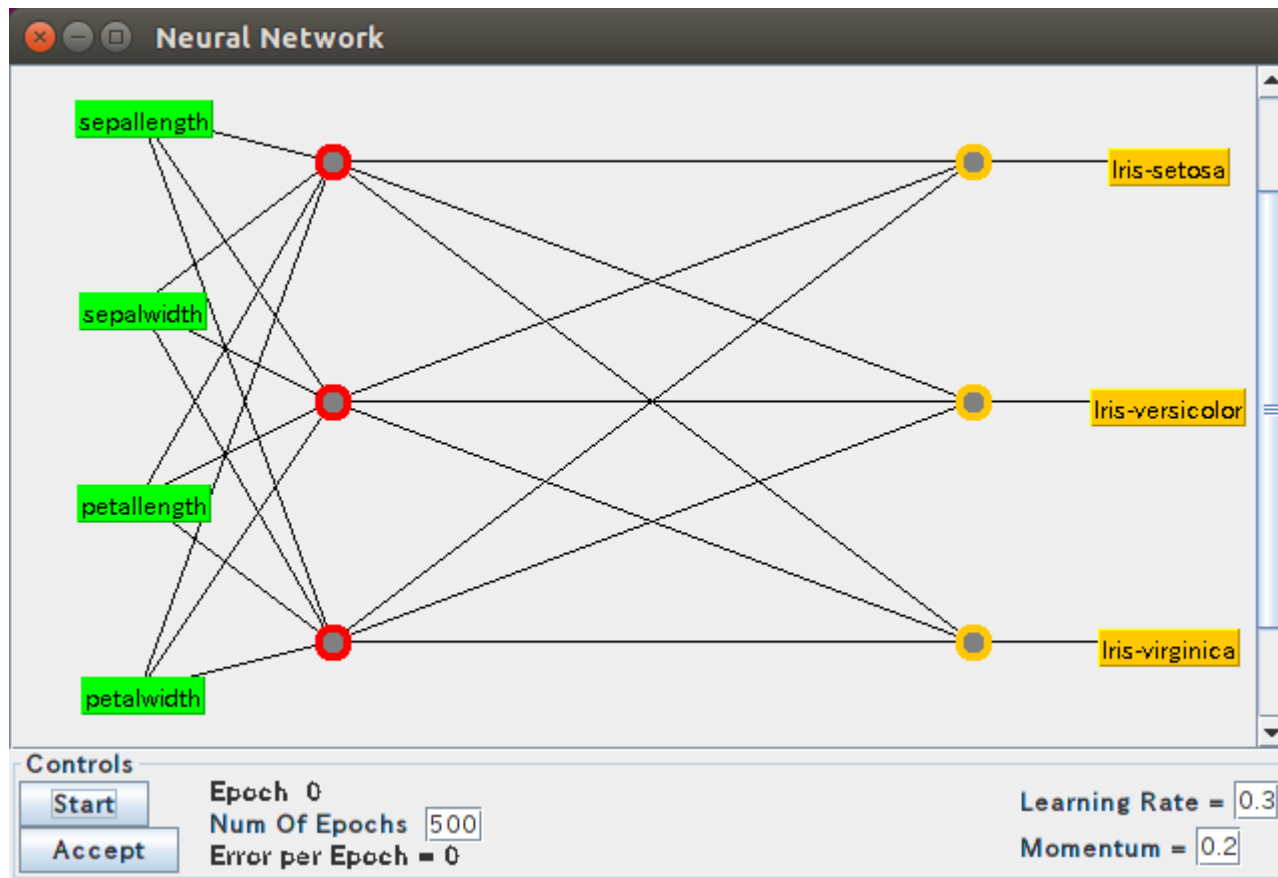
**validationThreshold:** 20

**Status:** OK

**Log:** x 0

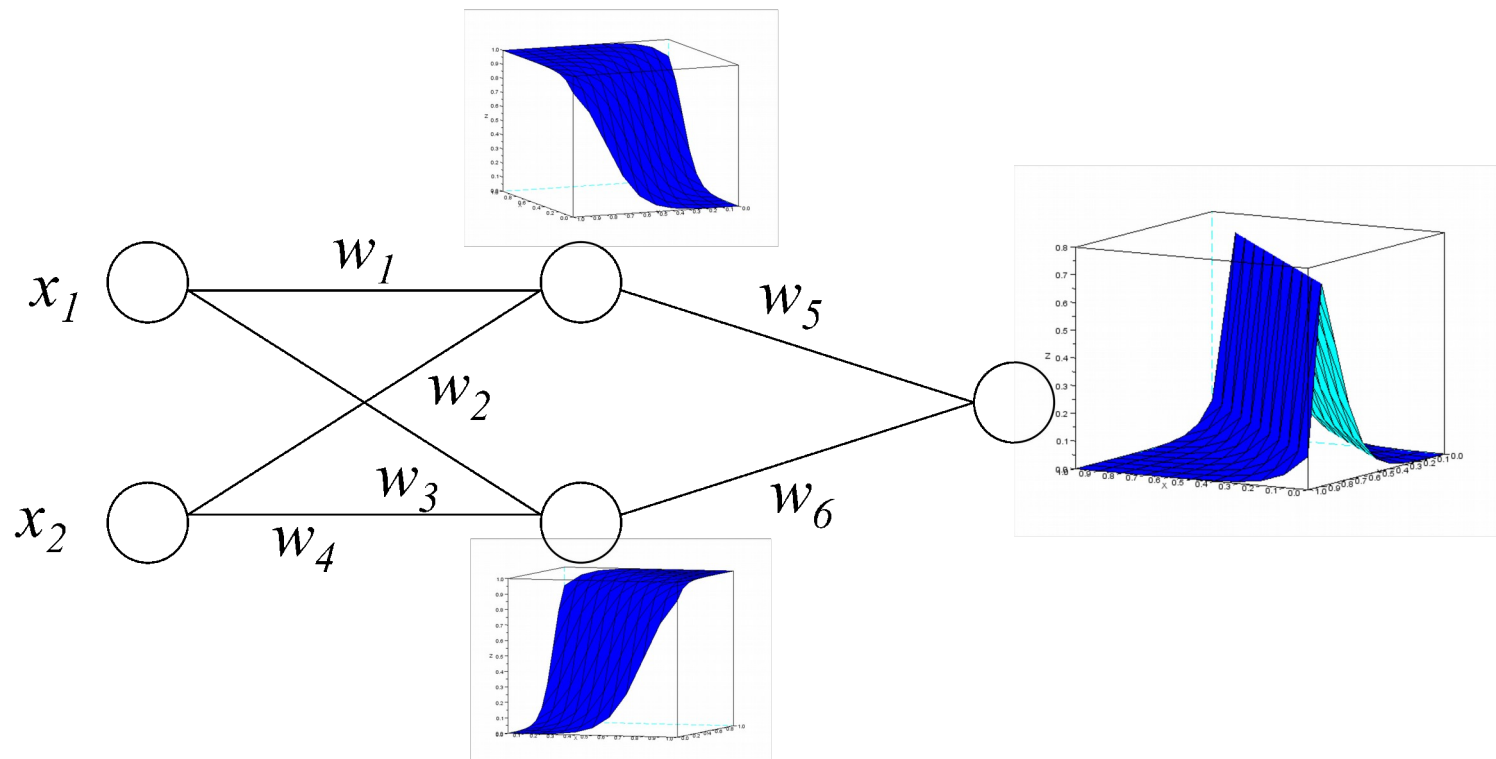


# Weka の MultilayerPerseptron



# 多層パーセプトロンの特質

- 識別面の複雑さ
  - 中間層のニューロンの個数に関する
  - シグモイド関数（非線形）を任意の重み・方向で足し合わせることで複雑な非線形識別面を構成



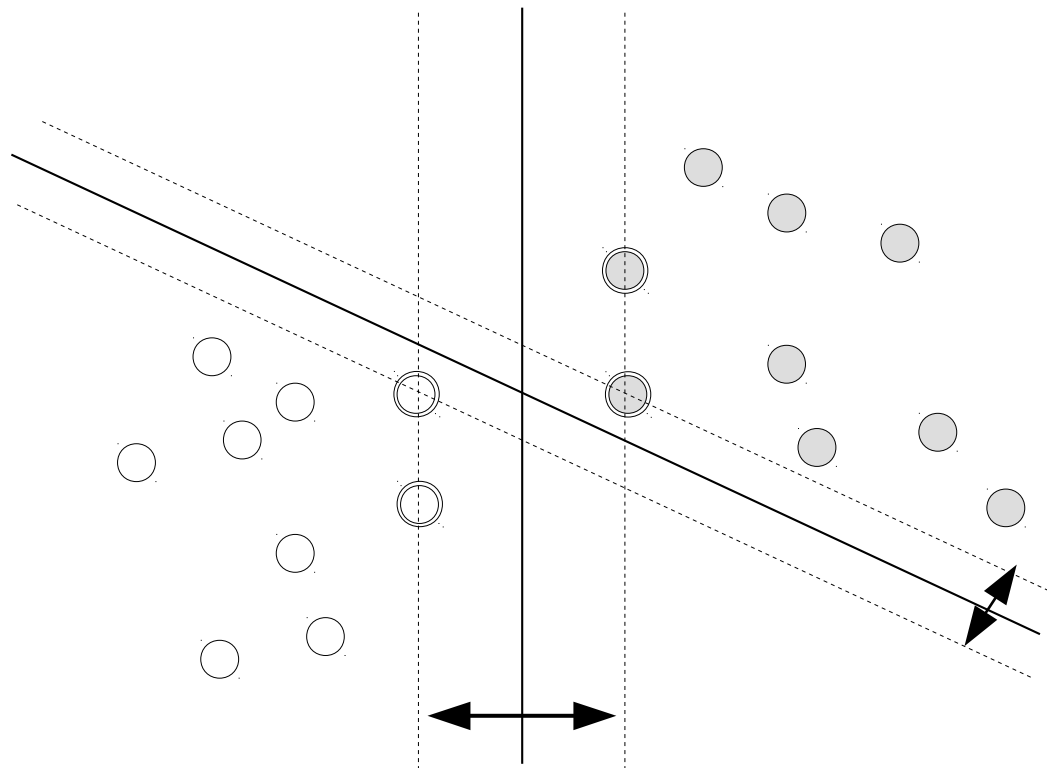
# なぜニューラルネットが流行らなかったか

- 過学習
  - 学習データに適応しすぎて、未知データに弱い
- 局所的最適解
  - 初期パラメータを変えて何度も実験
- ハイパーパラメータがいくつもあるが、挙動が理論的によくわからない
  - 中間層の数、モーメンタム ,etc.

# 7. 識別 - サポートベクトルマシン -

- マージンを最大化する識別面を求める

識別面と、最も近いデータとの距離



○ ○ : サポートベクトル

# 7.1 問題の定式化

- 学習データ

$$\{(\mathbf{x}_i, y_i)\} \quad i = 1, \dots, N, \quad y_i = 1 \text{ or } -1$$

- 識別面の式

$$\mathbf{w} \cdot \mathbf{x}_i + w_0 = 0$$

- 識別面の制約（係数を定数倍しても平面は不変）

$$\min_{i=1, \dots, N} |\mathbf{w} \cdot \mathbf{x}_i + w_0| = 1$$

- 学習パターンと超平面との最小距離

$$\min_{i=1, \dots, N} \text{Dist}(\mathbf{x}_i) = \min_{i=1, \dots, N} \frac{|\mathbf{w} \cdot \mathbf{x}_i + w_0|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

点と直線の距離の公式

$$r = \frac{|ax + by + c|}{\sqrt{a^2 + b^2}}$$

## 7.1 問題の定式化

- 目的関数：  $\min \frac{1}{2} \|\boldsymbol{w}\|^2$
- 制約条件：  $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + w_0) \geq 1 \quad i = 1, \dots, N$
- 解法：ラグランジュの未定乗数法
  - 問題  $\min f(x) \quad s.t. \quad g(x) = 0$
  - ラグランジュ関数  $L(x, \alpha) = f(x) + \alpha g(x)$ 
    - $x, \alpha$  で偏微分して 0 になる値が極値

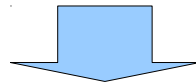
# 7.1 問題の定式化

- 計算

$$L(\mathbf{w}, w_0, \alpha) = \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^N \alpha_i (y_i (\mathbf{w} \cdot \mathbf{x}_i + w_0) - 1)$$

$$\frac{\partial L}{\partial w_0} = 0 \quad \Rightarrow \quad \sum_{i=1}^N \alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^N \alpha_i y_i \mathbf{x}_i$$

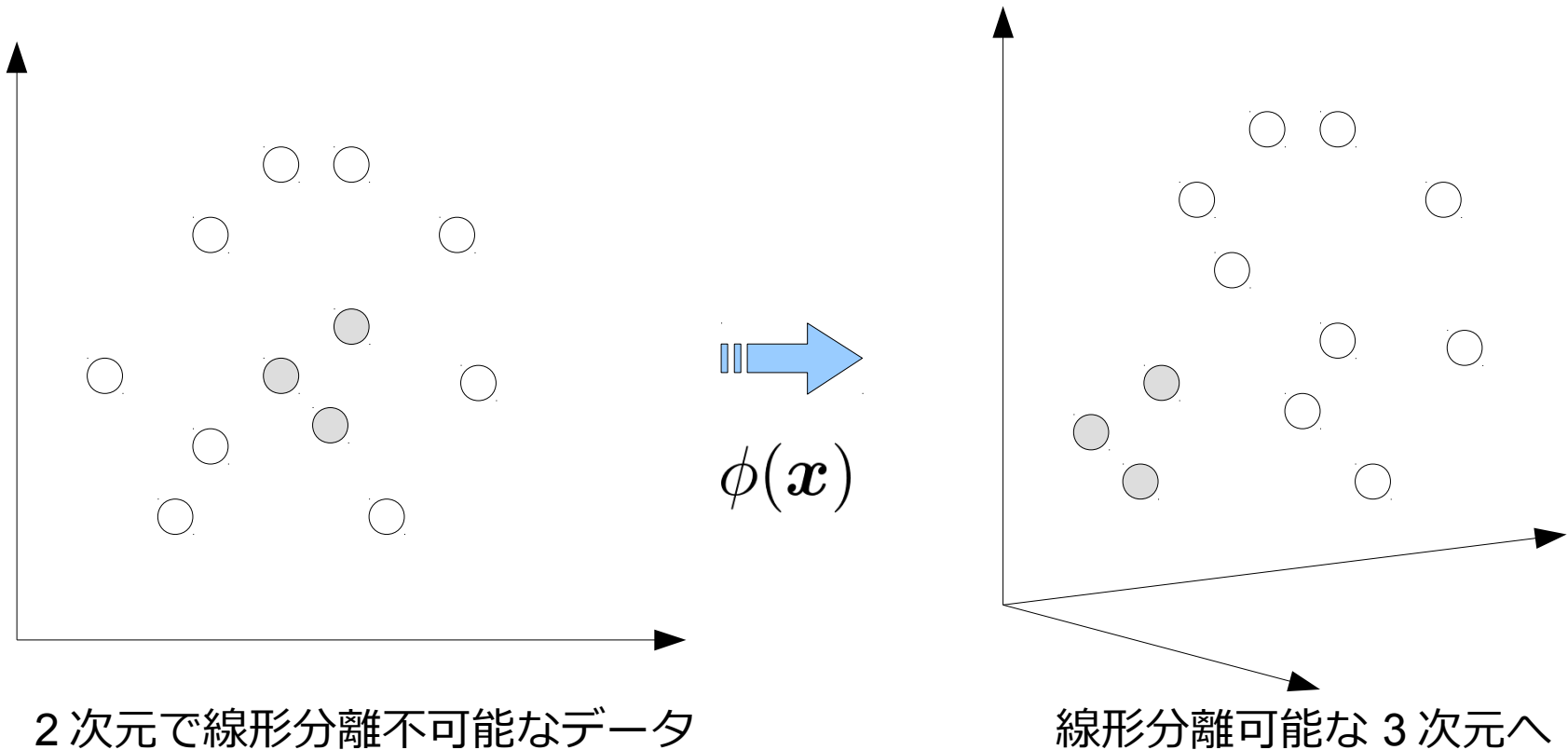


$$L(\alpha) = \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^N \alpha_i$$

α についての  
2 次計画問題

## 7.2 カーネル関数の利用

- 特徴ベクトルの次元を増やす



ただし、元の空間でのデータ間の  
距離関係は保持するように



## 7.2 カーネル関数の利用

- 非線形変換関数：  $\phi(\mathbf{x})$
- カーネル関数
  - 元の空間での距離が変換後の空間の内積に対応

$$K(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x}) \cdot \phi(\mathbf{x}')$$

- カーネル関数の例

- 多項式カーネル  $K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + 1)^p$

- ガウシアンカーネル  $K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\sigma^2}\right)$

この形であれば、対応する非線形変換が存在することが数学的に保証されている

## 7.3 カーネル関数を用いた SVM

- 変換後の識別関数 :  $g(\mathbf{x}) = \mathbf{w} \cdot \phi(\mathbf{x}) + w_0$
- SVM で求めた  $\mathbf{w}$  の値を代入

$$\begin{aligned} g(\mathbf{x}) &= \sum_{i=1}^N \alpha_i y_i \phi(\mathbf{x}) \cdot \phi(\mathbf{x}_i) + w_0 \\ &= \sum_{i=1}^N \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + w_0 \end{aligned}$$

非線形変換の  
式は不要!!!

**カーネルトリック**

# Weka の SMO

The image shows the Weka Explorer interface with the SMO classifier selected. The Classifier output pane displays the following results:

```
Time taken to build model: 0.001
=== Stratified cross-validation
=== Summary ===
Correctly Classified Instances 100
Incorrectly Classified Instances 0
Kappa statistic 1.000
Mean absolute error 0.000
Root mean squared error 0.000
Relative absolute error 0.000
Root relative squared error 0.000
Coverage of cases (0.95 level) 1.000
Mean rel. region size (0.95 level) 1.000
Total Number of Instances 100

=== Detailed Accuracy By Class ===

```

|               | TP Rate | FP Rate |
|---------------|---------|---------|
| 1             | 1.000   | 0.000   |
| 2             | 0.980   | 0.050   |
| 3             | 0.900   | 0.010   |
| Weighted Avg. | 0.960   | 0.020   |

```
=== Confusion Matrix ===
 a b c <-- classified as
50 0 0 | a = Iris-setosa
 0 49 1 | b = Iris-versicolor
 0 5 45 | c = Iris-virginica
```

The GenericObjectEditor window shows the following configuration for weka.classifiers.functions.SMO:

- buildLogisticModels: False
- c: 1.0
- checksTurnedOff: False
- debug: False
- doNotCheckCapabilities: False
- epsilon: 1.0E-12
- filterType: Normalize training data
- kernel: Choose PolyKernel -E 1.0 -C 250007
- numFolds: -1
- randomSeed: 1
- toleranceParameter: 0.001

Buttons: Open..., Save..., OK, Cancel

Status: OK

Log x 0

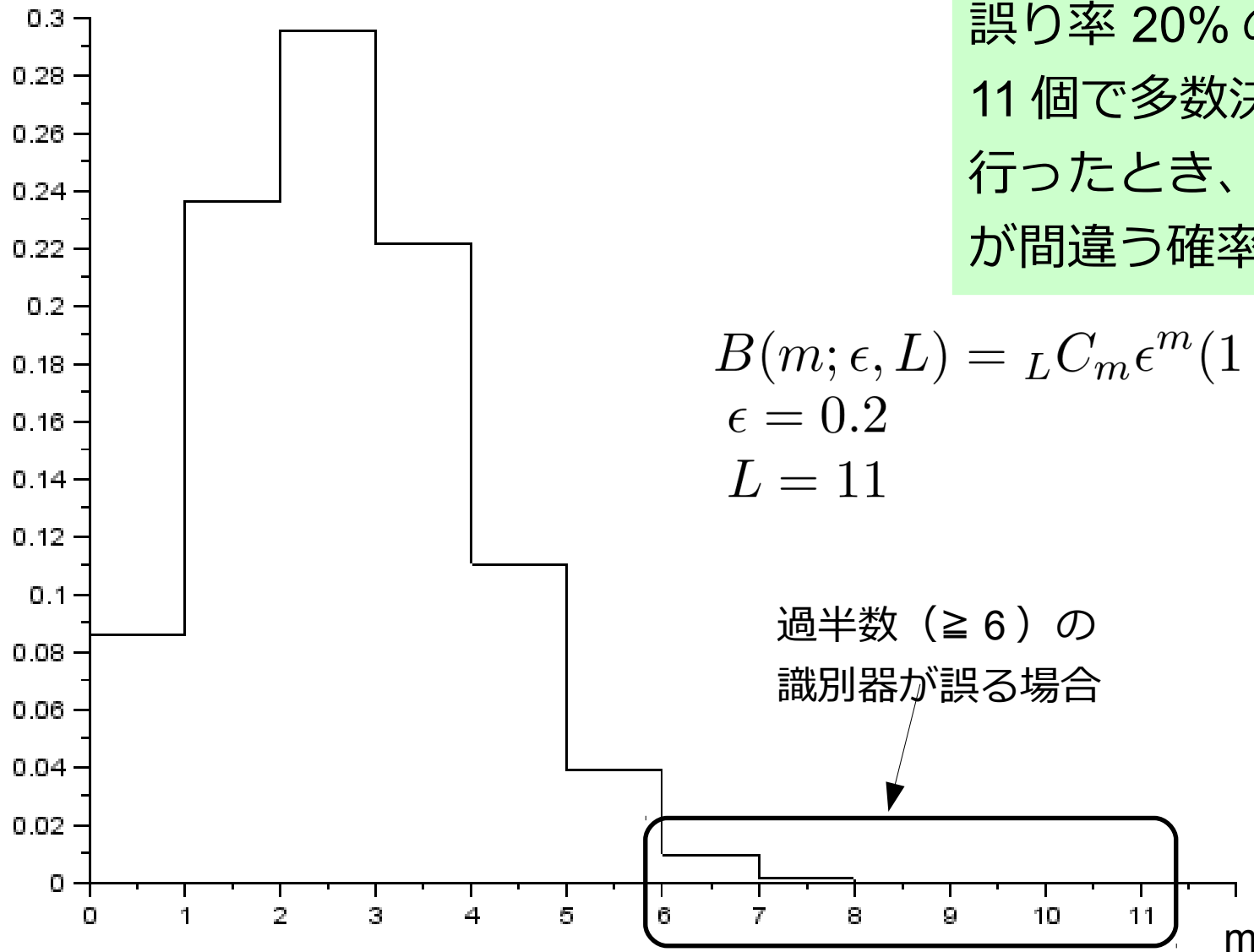
# なぜ SVM が流行ったか

- 大局的最適解
- 高次元特徴量に強い
  - 文書分類などでは数万次元になることも
- カーネルトリック
  - 文書・グラフ構造など、とにかくカーネル関数が定義できればよい

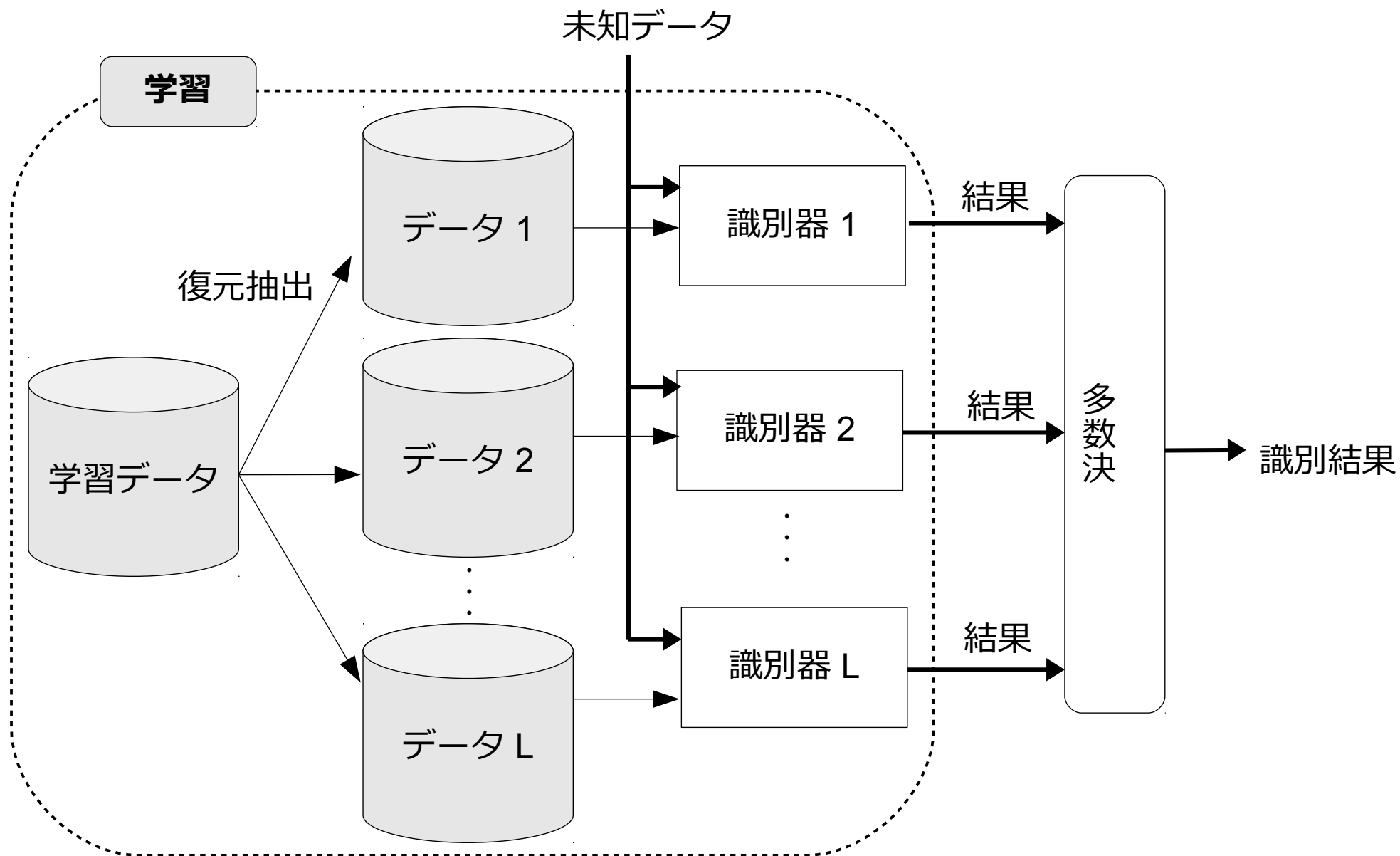
# 9. アンサンブル学習

- アンサンブル学習とは
  - 分類器を複数組み合わせ、それらの結果を統合することで個々の分類器よりも性能を向上させる方法
- アイディア
  - 訓練例集合から全く独立に  $L$  個の分類器 ( 誤り率  $\epsilon$ , 誤りは独立 ) を作成
    - $m$  個の分類器が誤る確率は二項分布  $B(m; \epsilon, L)$
    - $\epsilon < 0.5$  のとき、 $m > L/2$  となる  $B$  は小さい値

# 9.1 なぜ性能が向上するのか



# 9.2 バギング



## 9.2 バギング

- 特徴

- 訓練例から復元抽出することで、元のデータと同じサイズの独立なデータ集合を作成する。

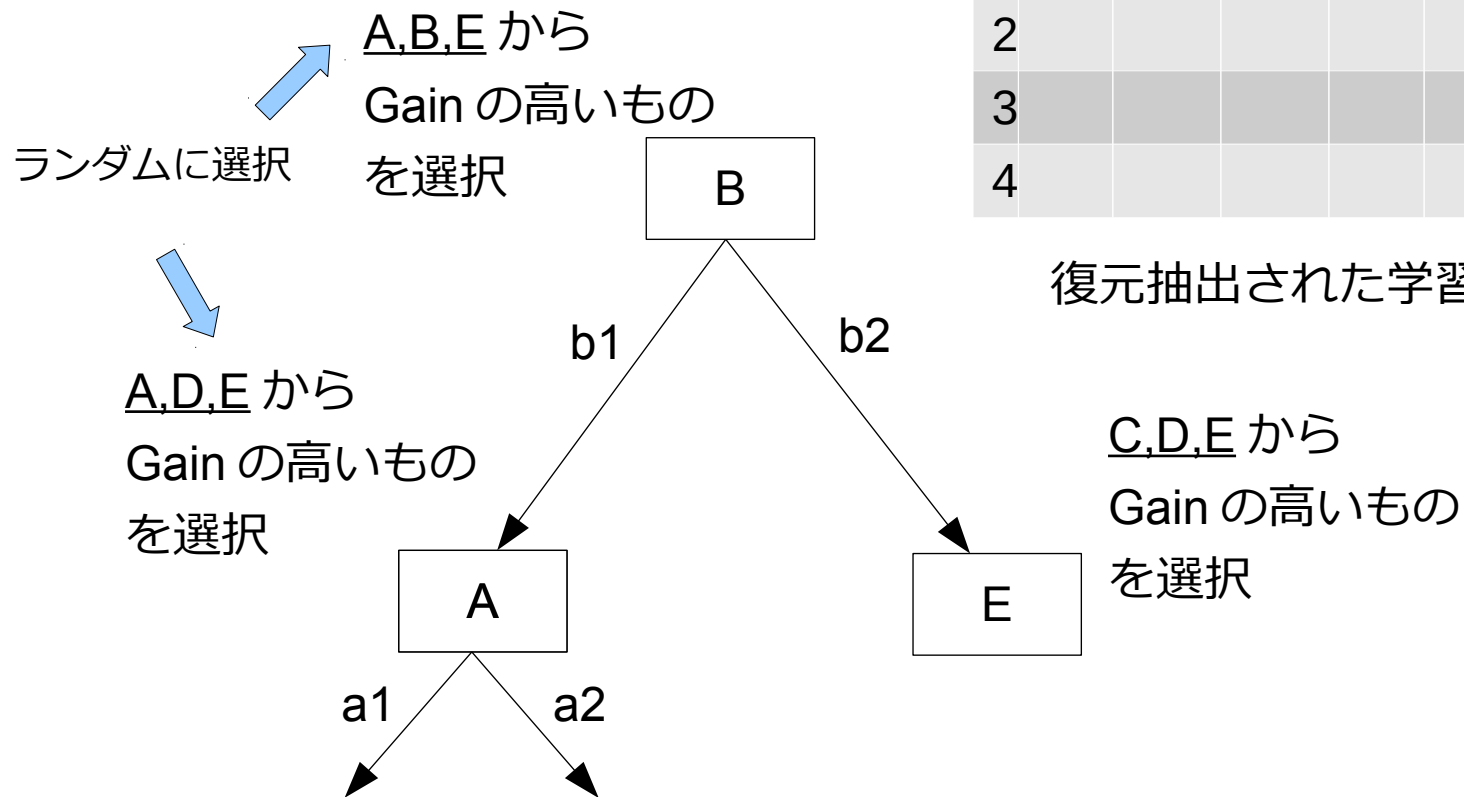
n回行って、あるデータが抽出されない確率： $(1 - \frac{1}{n})^n$

$n \rightarrow \infty$  で  
約 0.368

- 各々のデータに対して同じアルゴリズムで分類器を作成する
  - アルゴリズムは不安定 (学習データの違いに敏感) な方がよい
    - 例) 枝刈りをしない決定木
- 結果の統合は多数決



# 9.3 ランダムフォレスト



|   | A | B | C | D | E | class |
|---|---|---|---|---|---|-------|
| 1 |   |   |   |   |   |       |
| 2 |   |   |   |   |   |       |
| 3 |   |   |   |   |   |       |
| 4 |   |   |   |   |   |       |

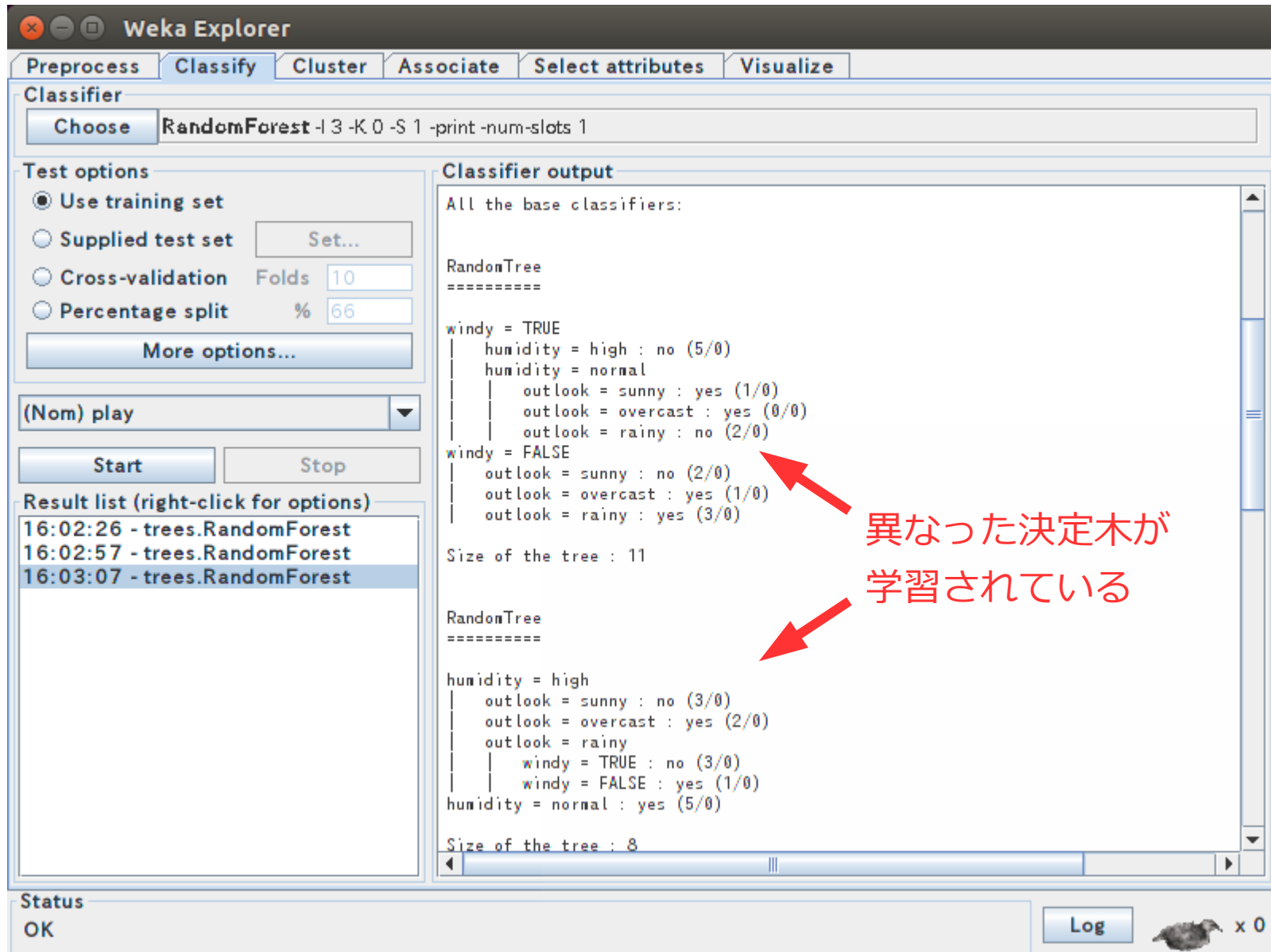
復元抽出された学習データ

葉が単一クラスになるまで (過) 学習

## 9.3 ランダムフォレスト

- 特徴
  - バギングと同様、学習データは復元抽出
  - 識別器作成に使用できる特徴をランダムに制限することで、各抽出データ毎に全く異なった識別器ができる
  - 識別器は意図的に過学習させる

# Weka の RandomForest



Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose RandomForest -I 3 -K 0 -S 1 -print -num-slots 1

Test options

- Use training set
- Supplied test set
- Cross-validation Folds
- Percentage split %

(Nom) play

Result list (right-click for options)

- 16:02:26 - trees.RandomForest
- 16:02:57 - trees.RandomForest
- 16:03:07 - trees.RandomForest

Classifier output

All the base classifiers:

```
RandomTree
=====
windy = TRUE
| humidity = high : no (5/0)
| humidity = normal
| | outlook = sunny : yes (1/0)
| | outlook = overcast : yes (0/0)
| | outlook = rainy : no (2/0)
windy = FALSE
| outlook = sunny : no (2/0)
| outlook = overcast : yes (1/0)
| outlook = rainy : yes (3/0)

Size of the tree : 11

RandomTree
=====
humidity = high
| outlook = sunny : no (3/0)
| outlook = overcast : yes (2/0)
| outlook = rainy
| | windy = TRUE : no (3/0)
| | windy = FALSE : yes (1/0)
humidity = normal : yes (5/0)

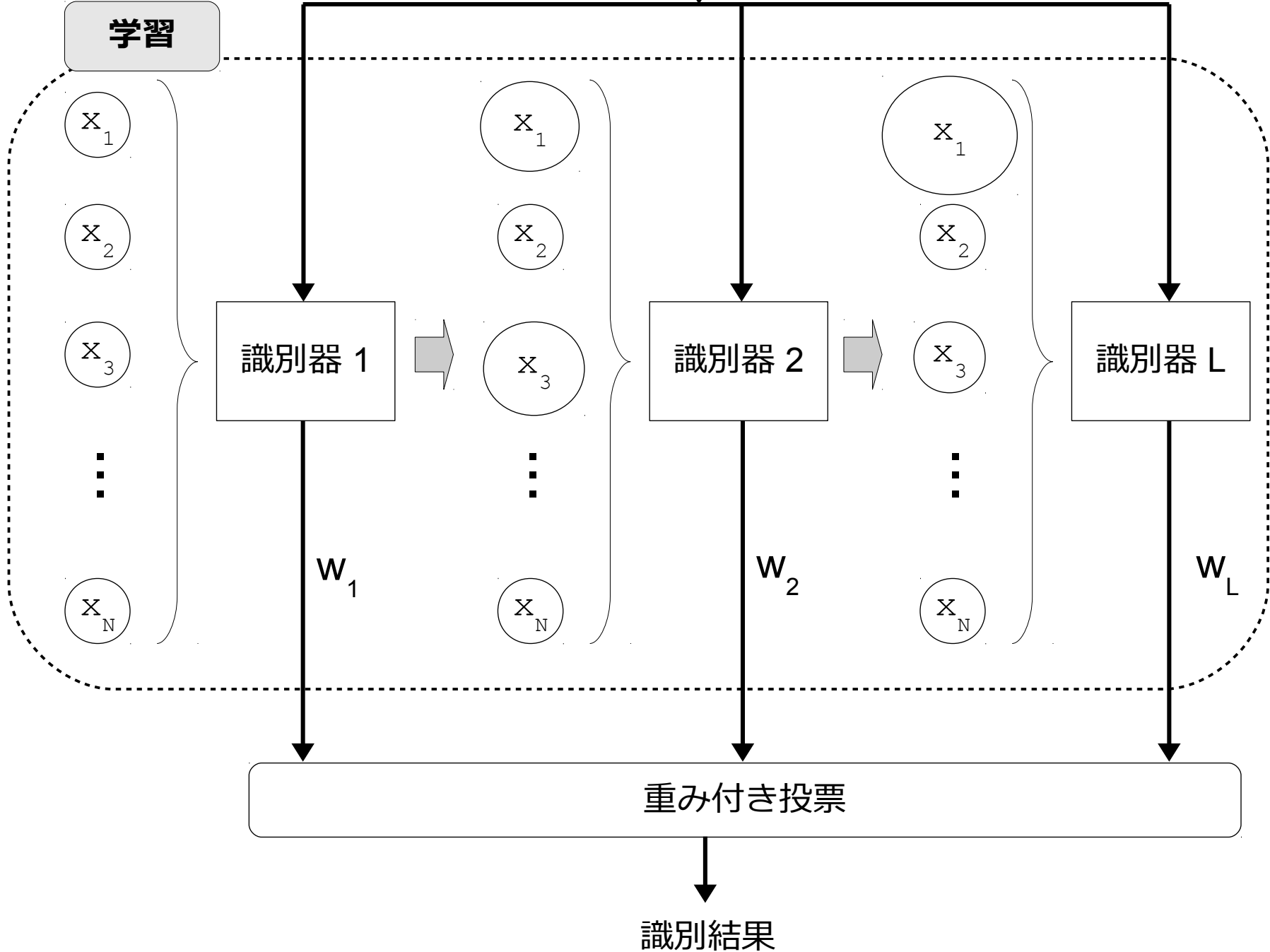
Size of the tree : 8
```

異なった決定木が学習されている

Status OK  x 0

# 9.4 ブースティング

未知データ

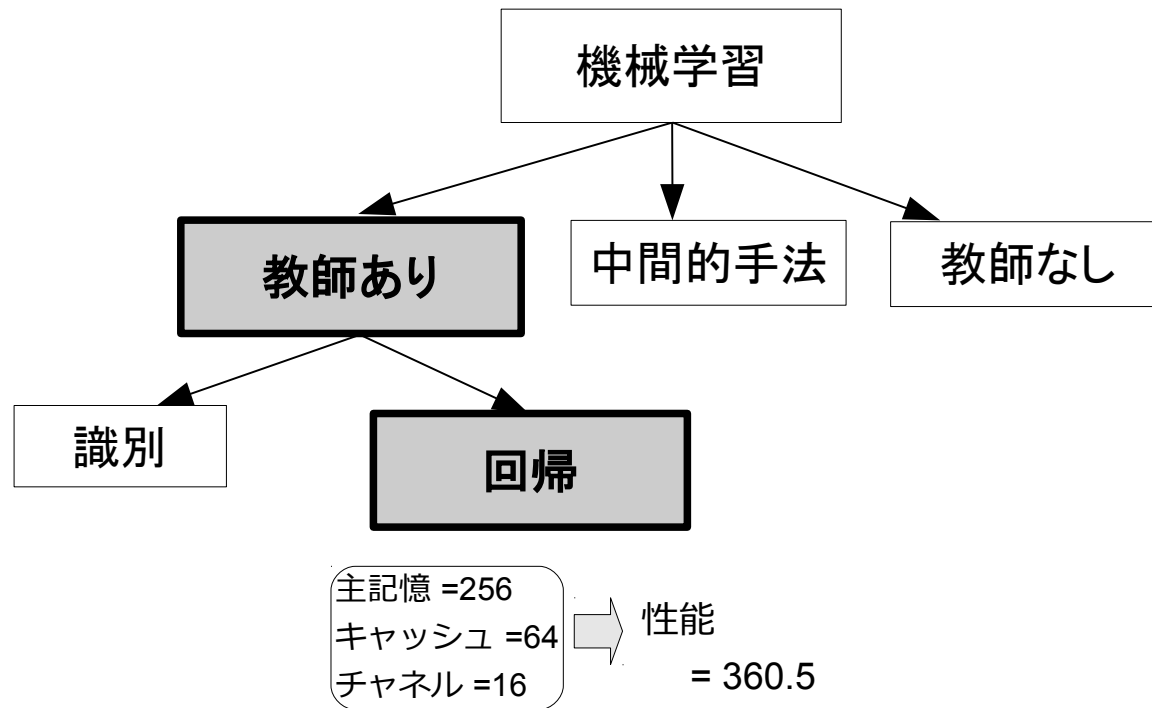


## 9.4 ブースティング

- 特徴
  - 逐次的に相補的な分類器を作成
  - 以前の分類器が誤った事例に重みを付けて次の分類器を学習
  - 学習アルゴリズムが重みに対応していない場合は、重みに比例した数を復元抽出
  - 結果は分類器に対する重み付き投票

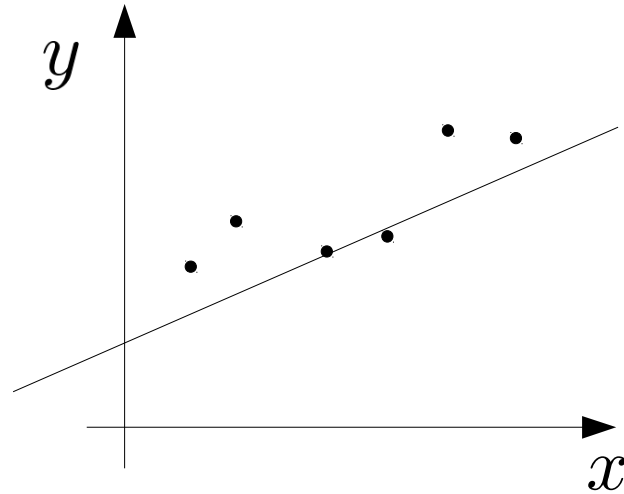
# 8. 回帰

- 問題設定
  - 教師あり学習
  - 数値入力 → 数値出力



## 8.2 線形回帰

- 目標：なるべく誤差の少ない直線を求める



- 線形回帰の定義
  - 入力  $\mathbf{x}$  から出力  $y$  を求める回帰式を 1 次式に限定
  - 学習データから係数  $w$  を求める

$$\hat{c}(\mathbf{x}) = \sum_{i=0}^d w_i x_i$$

## 8.2 線形回帰

- 最小二乗法による係数の推定
  - 推定の基準：誤差の二乗和  $E$  を最小化

$$E(\mathbf{w}) = \sum_{i=1}^N (y_i - \hat{c}(\mathbf{x}_i))^2$$

$$= (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w})$$

$\mathbf{X}$ : 全学習データを並べた行列

$\mathbf{w}$ : 係数のベクトル表現

- $w$  で微分した値が 0 となるのは

$$\mathbf{X}^T (\mathbf{y} - \mathbf{X}\mathbf{w}) = 0$$

$$\Leftrightarrow \mathbf{w} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$$

$w$  が解析的に  
求まる



## 8.2 線形回帰

- 最小二乗法の精度向上

例  $\phi(x) = (1, x, x^2, \dots, x^b)$

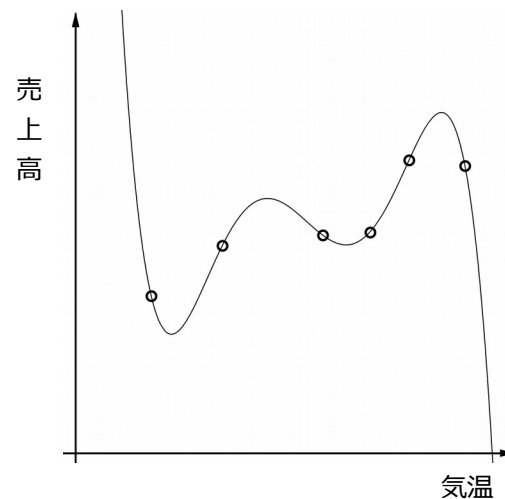
- 基底関数  $\phi(\mathbf{x}) = (\phi_1(\mathbf{x}), \dots, \phi_b(\mathbf{x}))$  を考える

$$\hat{c}(\mathbf{x}) = \sum_{j=0}^b w_j \phi_j(\mathbf{x})$$

- 係数が線形であれば、最小二乗法が適用可能

- 問題点

- 汎化性能の低下



## 8.2 線形回帰

- 正則化

- 正則化項の導入

- 複雑なパラメータ  $w$  (過学習) の回避

- L1 ノルム  $|w|$  : 0 となるパラメータが多くなる

Lasso

- L2 ノルム  $\|w\|^2$  : パラメータを 0 に近づける

Ridge

- リッジ回帰

- 誤差の二乗和に L2 ノルム正則化項を加える

$$E(w) = (y - Xw)^T (y - Xw) + \lambda w^T w$$

$\lambda$  : 誤差の二乗和と正則化項とのバランス

$$w = (X^T X + \lambda I)^{-1} X^T y$$

$w$  が解析的に  
求まる

## 8.2 線形回帰

- ラッソ回帰

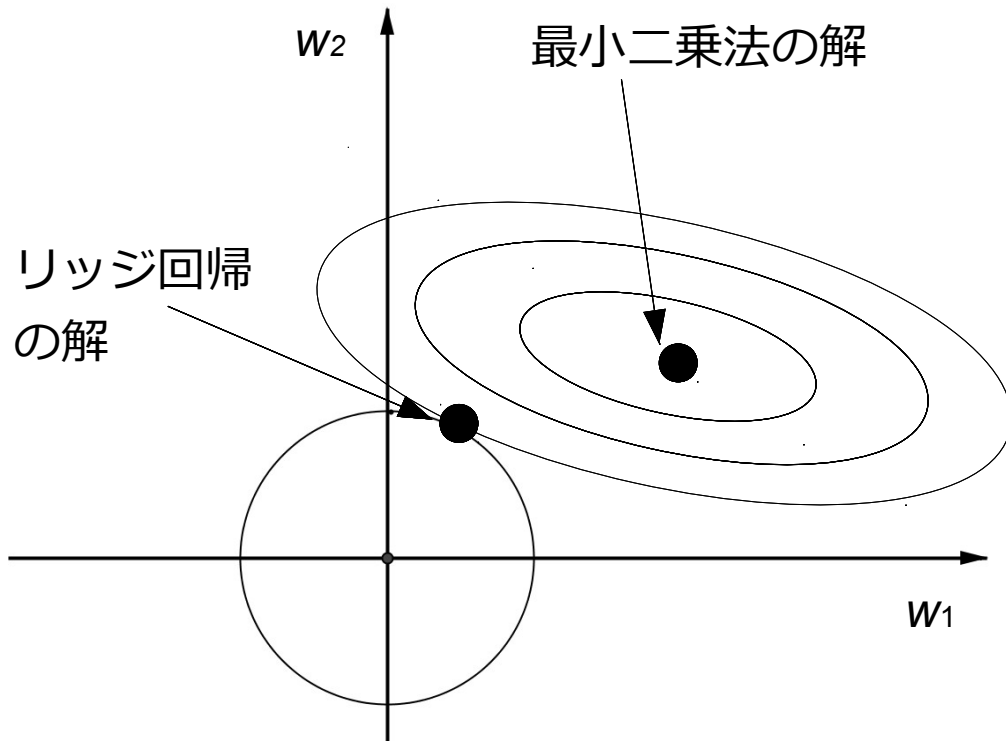
- 誤差の二乗和に L1 ノルム正則化項を加える

$$E(\mathbf{w}) = (\mathbf{y} - \mathbf{X}\mathbf{w})^T (\mathbf{y} - \mathbf{X}\mathbf{w}) + \lambda \underbrace{\sum_{j=1}^d |w_j|}$$

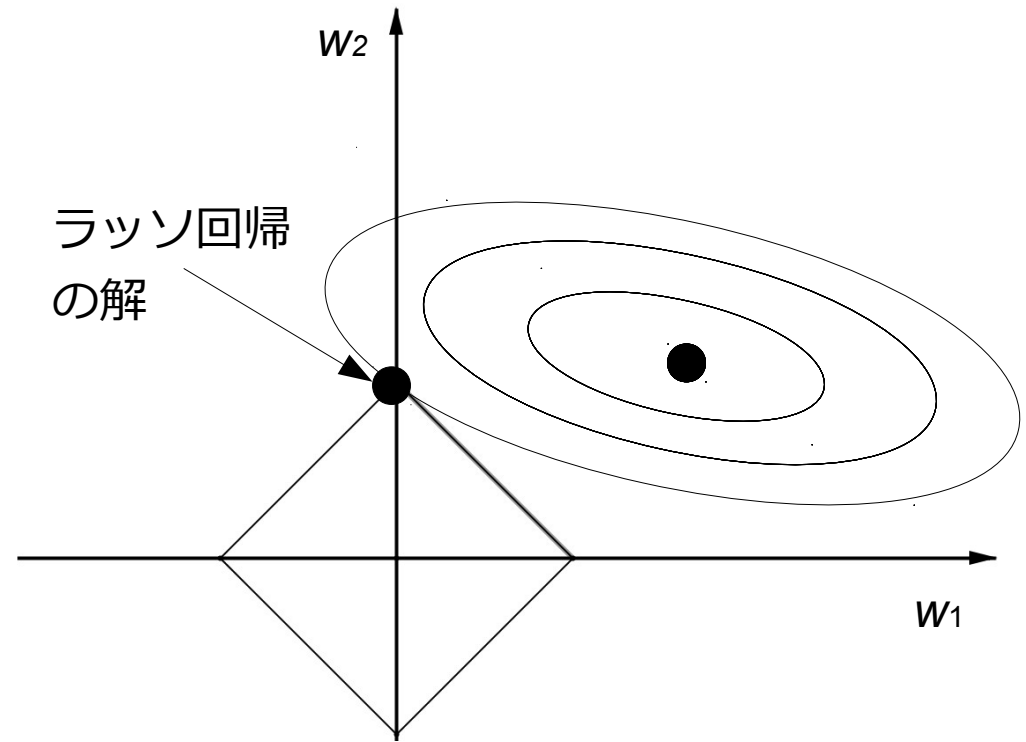
- 微分不可能な点があるため、解析的に解を求めることができない
  - 適当な初期重みから始め、リッジ回帰で上界を押さえる逐次更新アルゴリズムを用いる

# 8.2 線形回帰

- リッジ回帰とラッソ回帰



パラメータを0に近づけている



0となるパラメータを多くしている