

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

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、

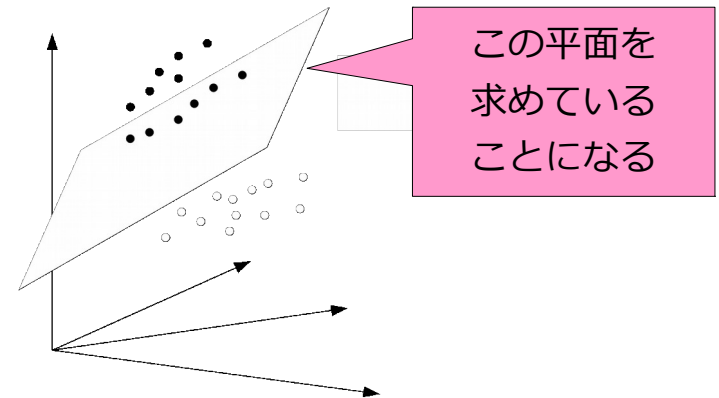
$$f_{Positive}(\boldsymbol{x}) > f_{Negative}(\boldsymbol{x})$$

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

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

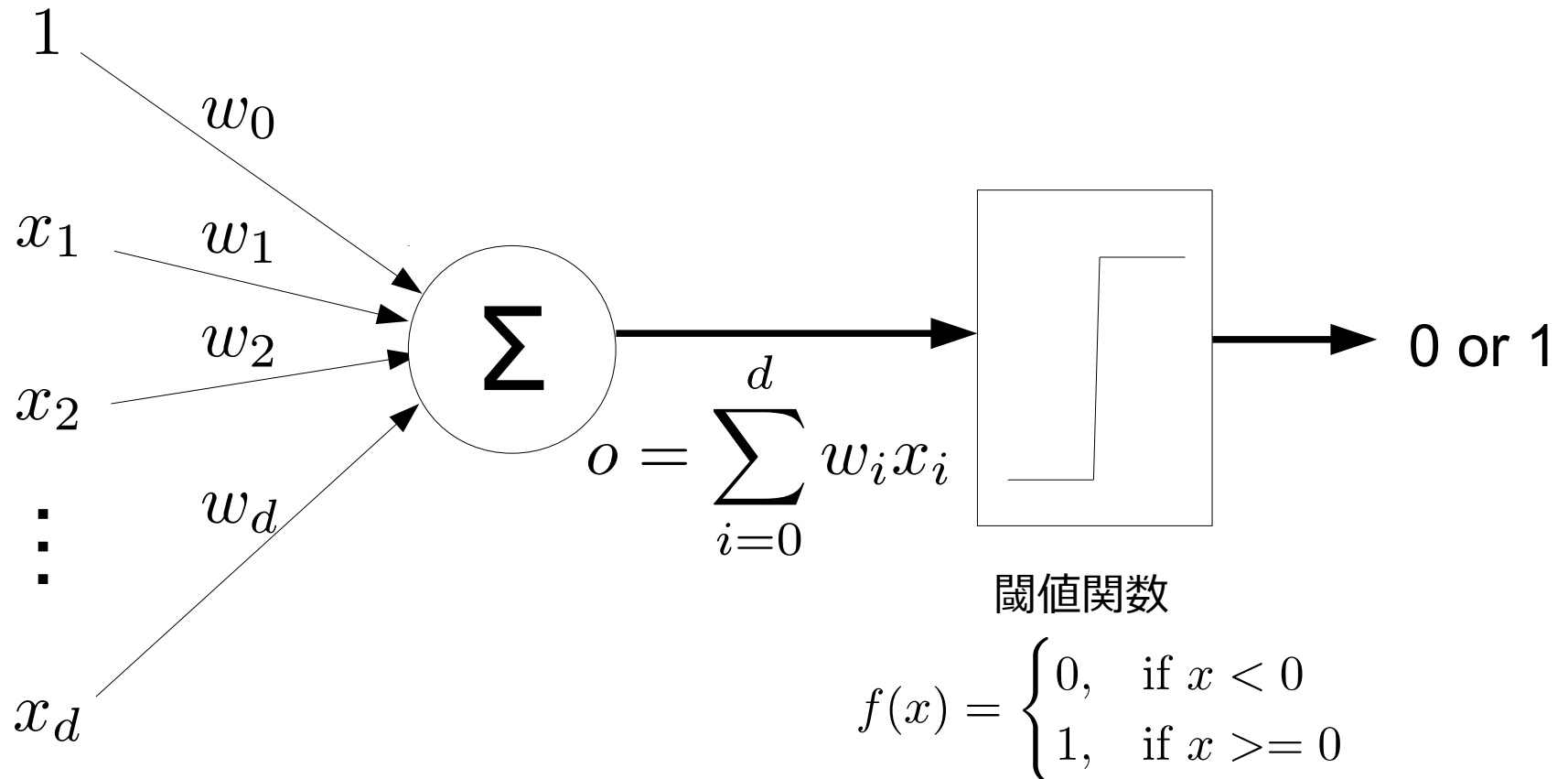


パーセプトロンの学習

- 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

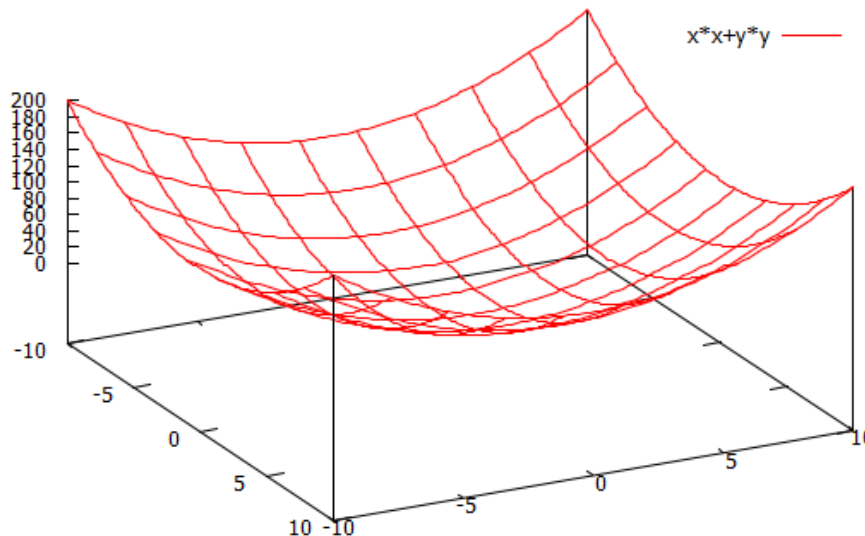
- エラーの定義

- 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

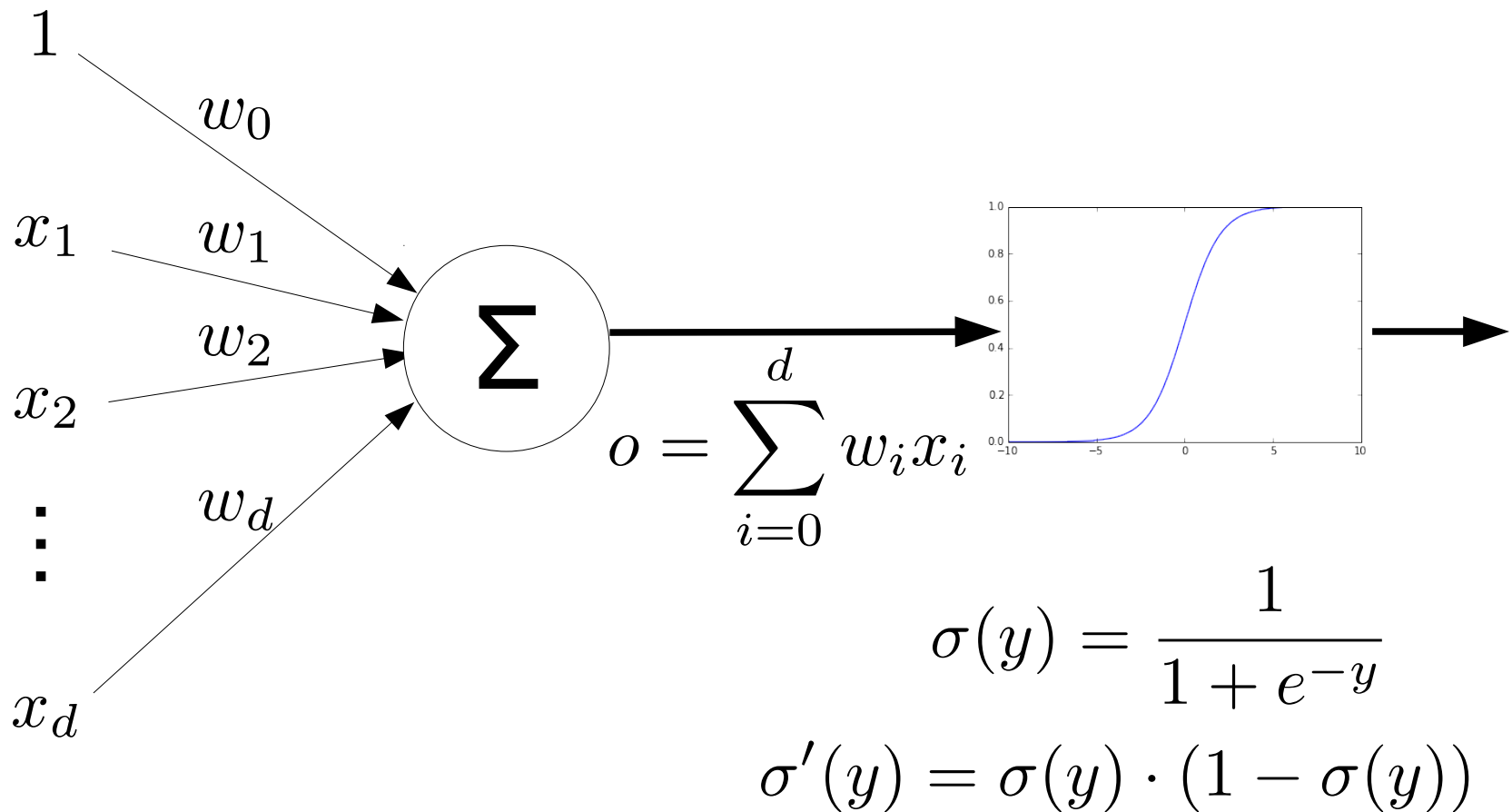
- w を E の勾配方向へ一定量だけ動かすことを繰り返して、最適解へ収束させる (→最急降下法)



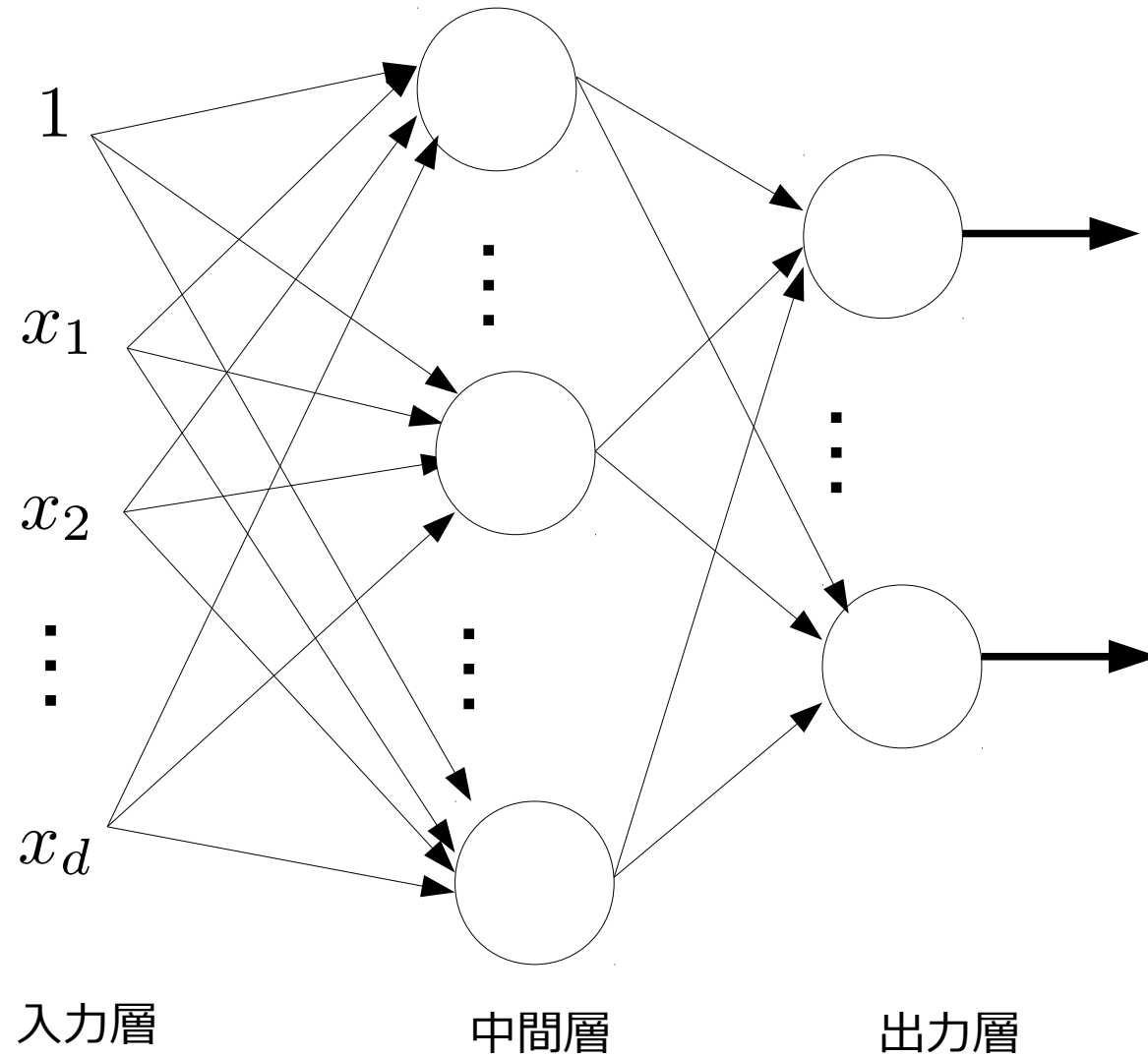
$$w_i \leftarrow w_i - \eta \frac{\partial E}{\partial w_i}$$

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

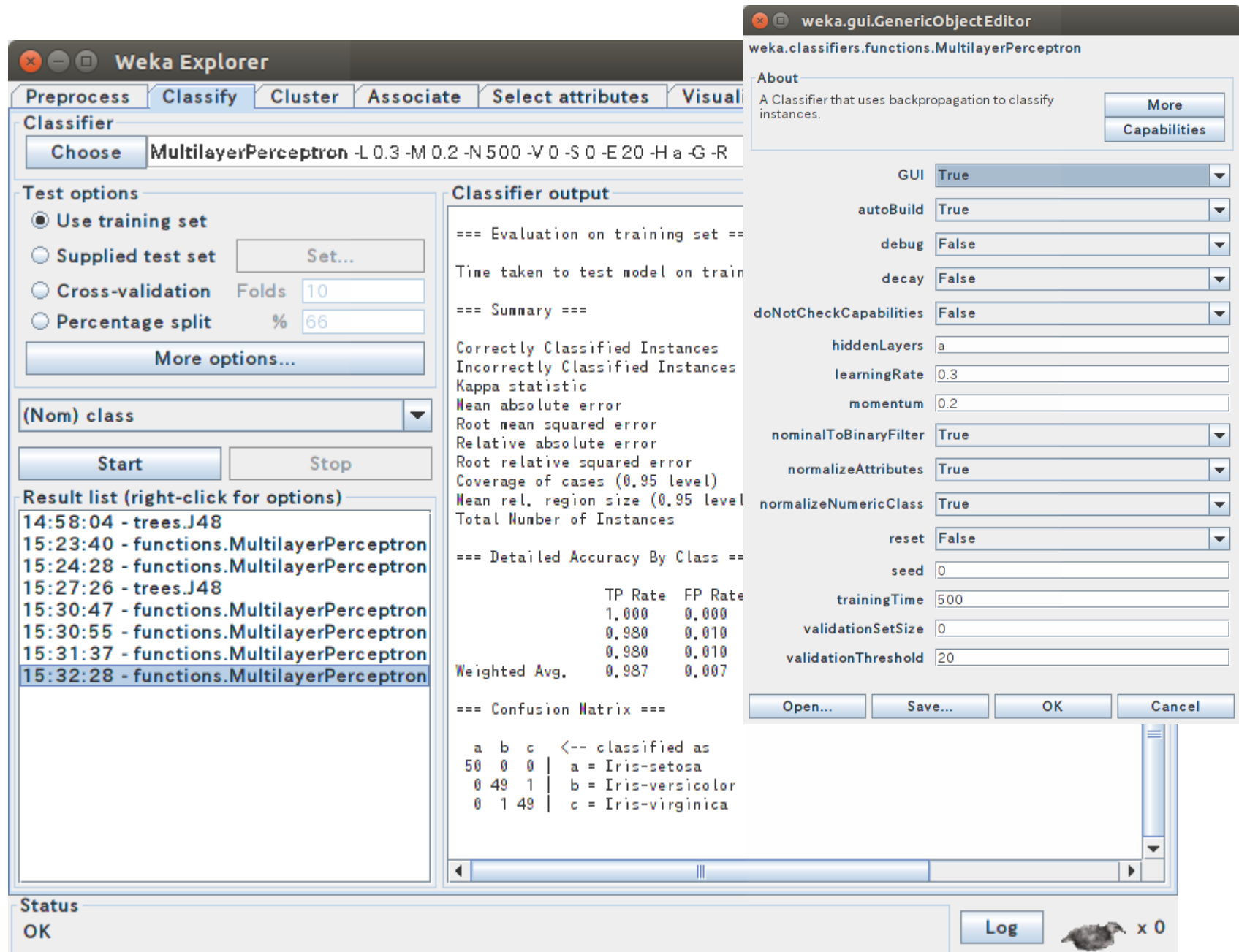


誤差逆伝播法による学習

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 \mathbf{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 δ 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 δ 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 the Weka Explorer interface with the MultilayerPerceptron classifier selected. The 'Classify' tab is active, and the 'Test options' section shows 'Use training set' selected. The 'Classifier output' section displays the results of the evaluation on the training set, including a summary of performance metrics and a detailed accuracy by class table.

Weka Explorer

Classifier

Choose **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
- ☐ Cross-validation Folds
- ☐ Percentage split %

(Nom) class

Result list (right-click for options)

- 14:58:04 - trees.J48
- 15:23:40 - functions.MultilayerPerceptron
- 15:24:28 - functions.MultilayerPerceptron
- 15:27:26 - trees.J48
- 15:30:47 - functions.MultilayerPerceptron
- 15:30:55 - functions.MultilayerPerceptron
- 15:31:37 - functions.MultilayerPerceptron
- 15:32:28 - functions.MultilayerPerceptron

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
1.000	0.000	
0.980	0.010	
0.980	0.010	
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

autoBuild

debug

decay

doNotCheckCapabilities

hiddenLayers

learningRate

momentum

nominalToBinaryFilter

normalizeAttributes

normalizeNumericClass

reset

seed

trainingTime

validationSetSize

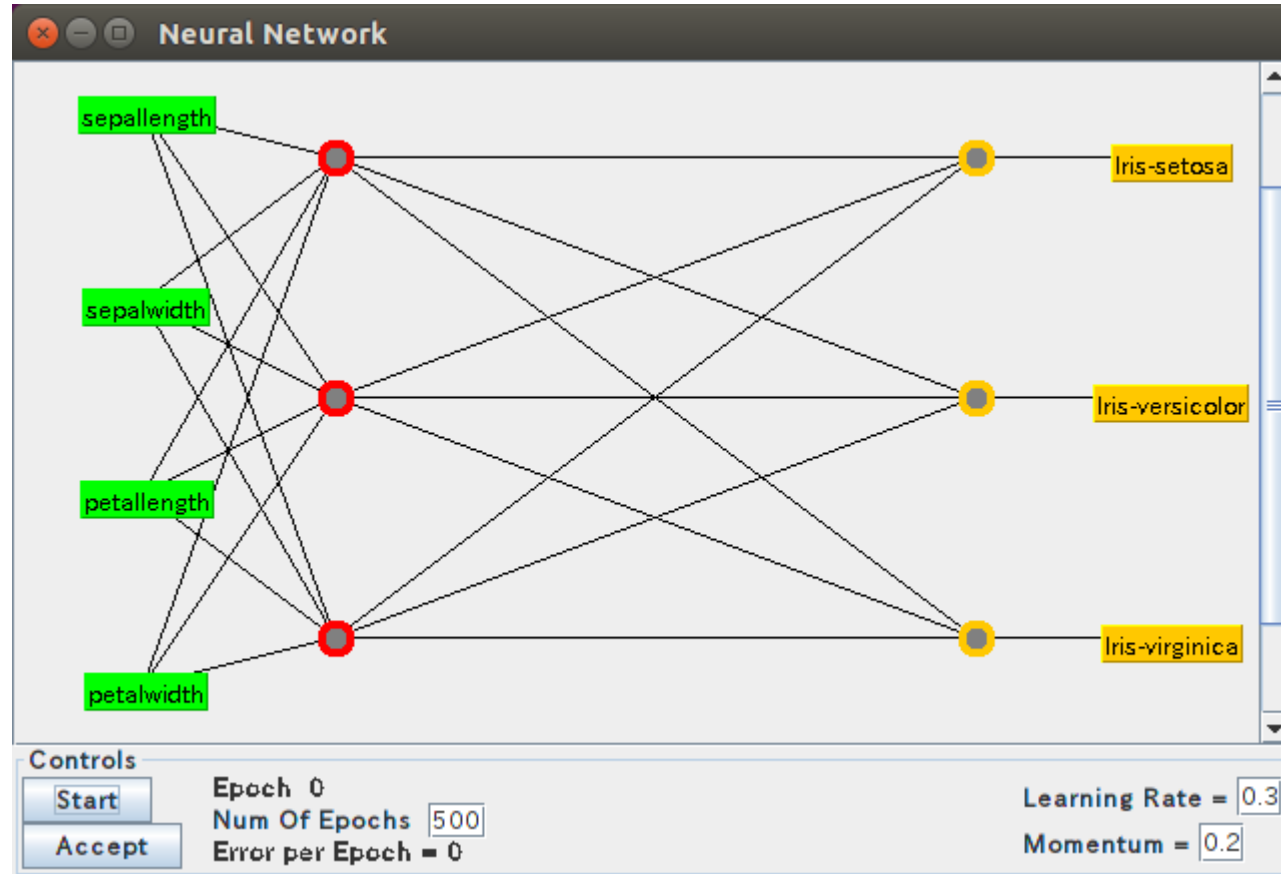
validationThreshold

Status

OK

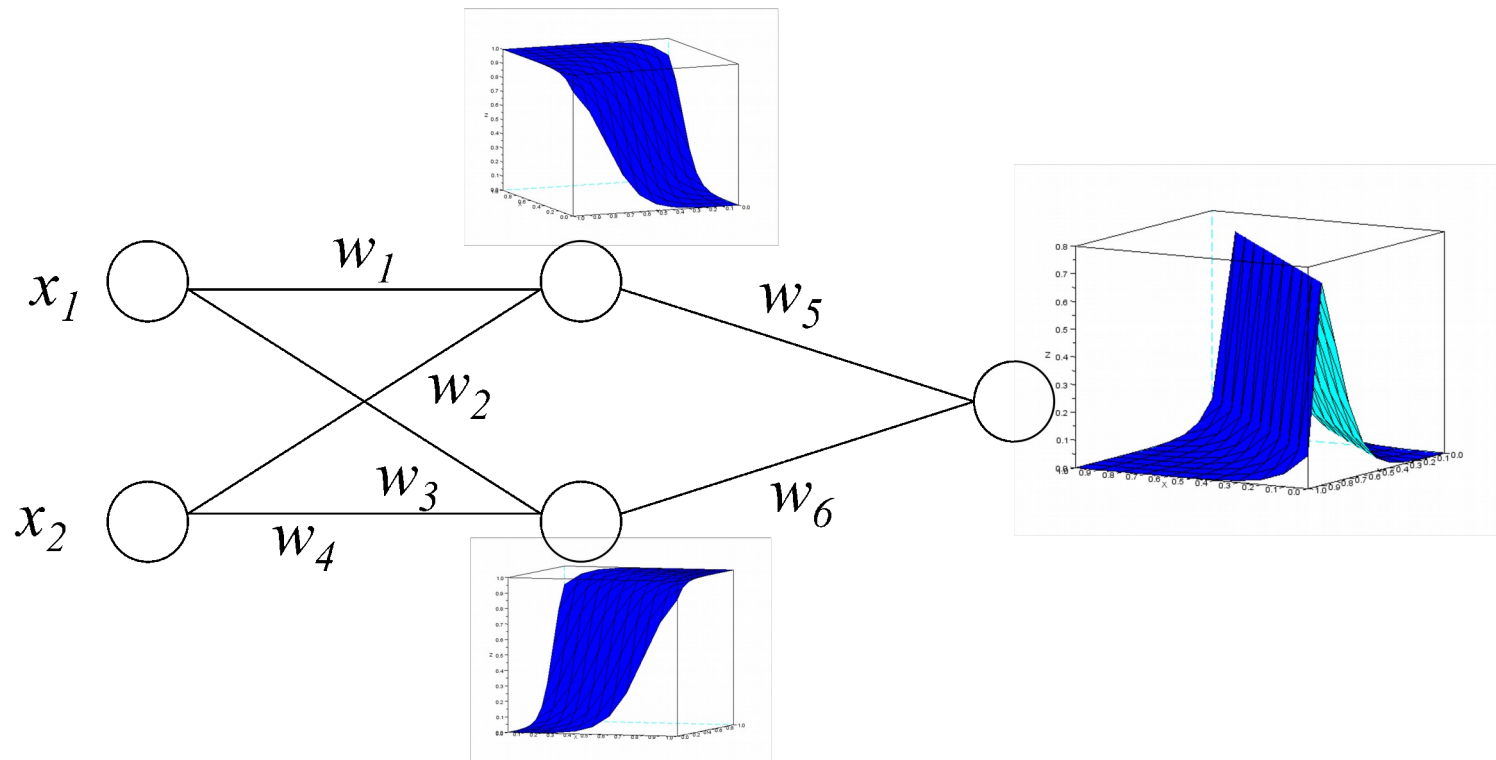
x 0

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



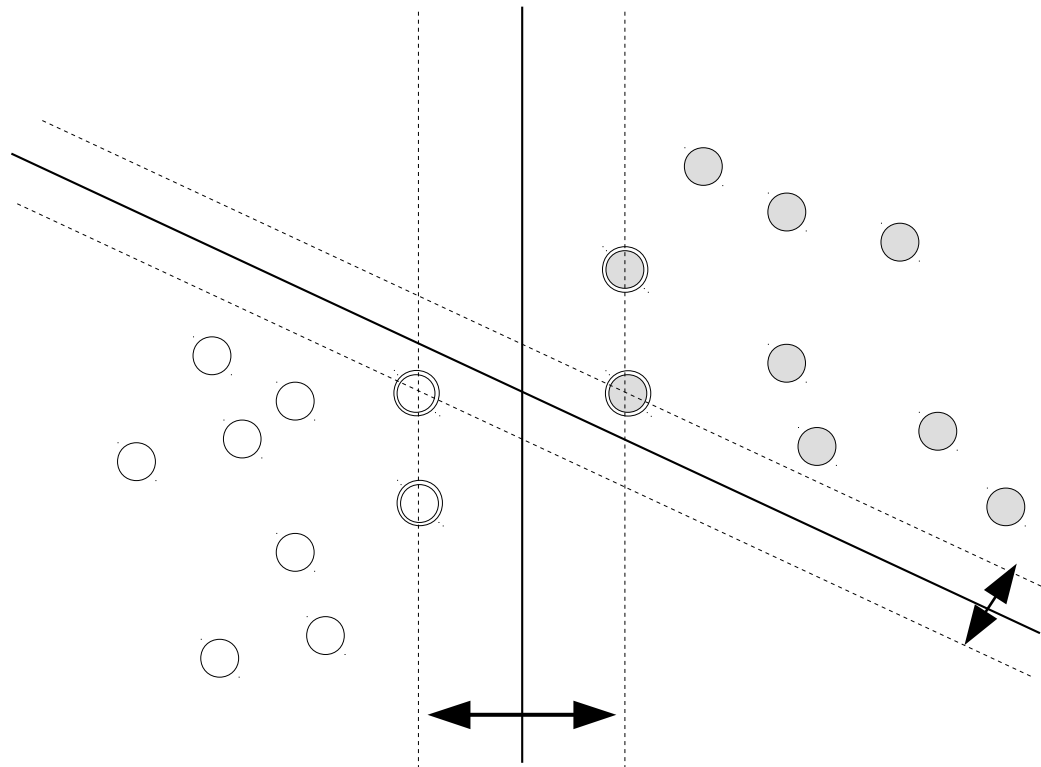
なぜニューラルネットが流行らなかったか

- 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かあるが、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

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

- 識別面の式

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

- 識別面の制約（係数を定数倍しても平面は不変）

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

- 学習パターンと超平面との最小距離

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

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, α で偏微分して 0 になる値が極値

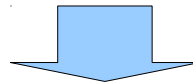
7.1 問題の定式化

- 計算

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

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

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

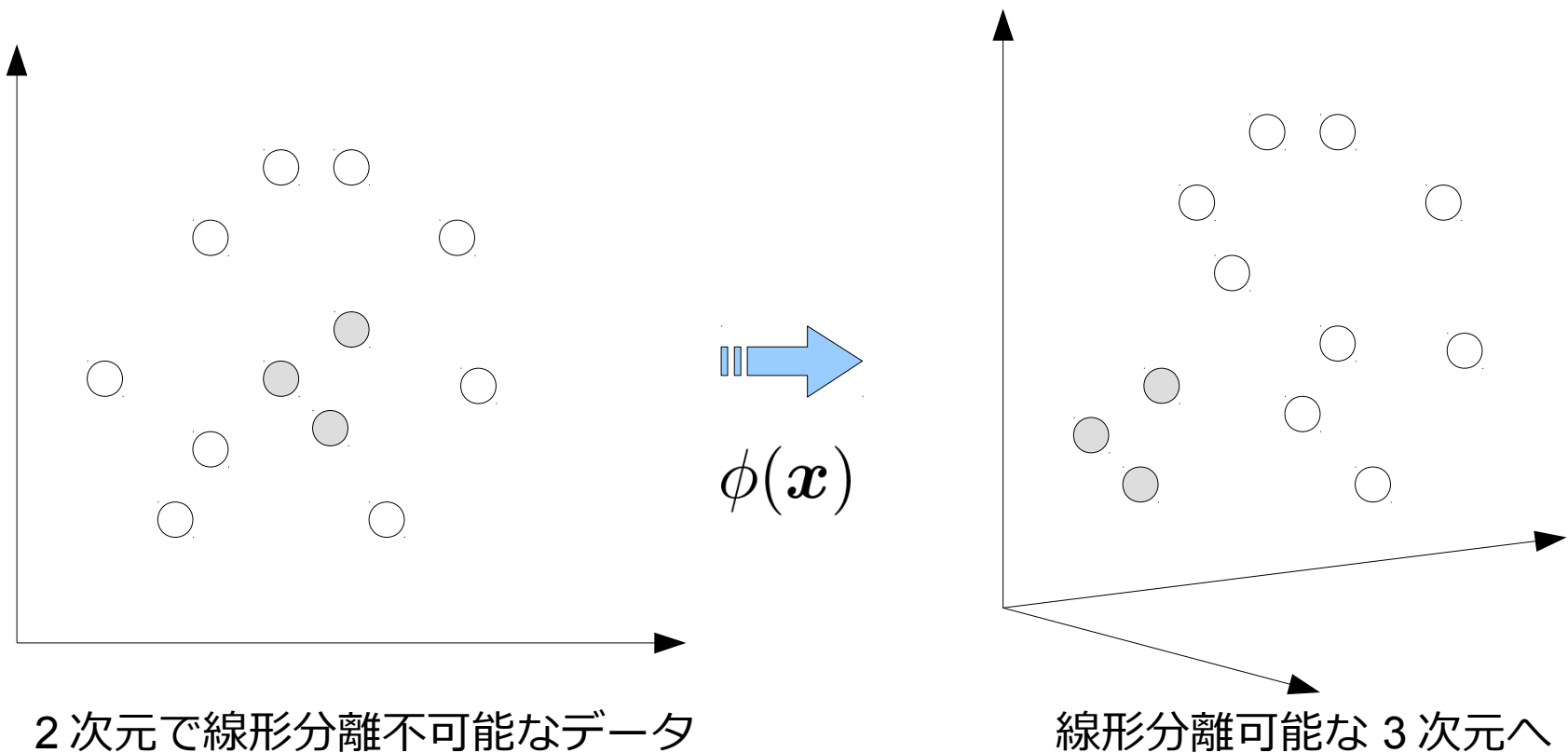


$$L(\alpha) = \frac{1}{2} \sum_{i,j=1}^N \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \sum_{i=1}^N \alpha_i$$

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： $\phi(\boldsymbol{x})$
- 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

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

- カーネル関数の例

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

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

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 'Classify' tab is active, and the 'Test options' section shows 'Cross-validation' selected with 10 folds. The 'Classifier output' section displays the results of the cross-validation, including a summary of performance metrics and a detailed accuracy by class table.

Weka Explorer

Classifier

Choose SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K *weka.classifi

Test options

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

More options...

(Nom) class

Start Stop

Result list (right-click for options)

16:00:33 - functions.SMO

Classifier output

Time taken to build model: 0.0

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances
Incorrectly Classified Instance
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
a	1.000	0.000
b	0.980	0.050
c	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

weka.gui.GenericObjectEditor

weka.classifiers.functions.SMO

About

Implements John Platt's sequential minimal optimization algorithm for training a support vector classifier.

More
Capabilities

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

Open... Save... OK Cancel

Status OK

Log x 0

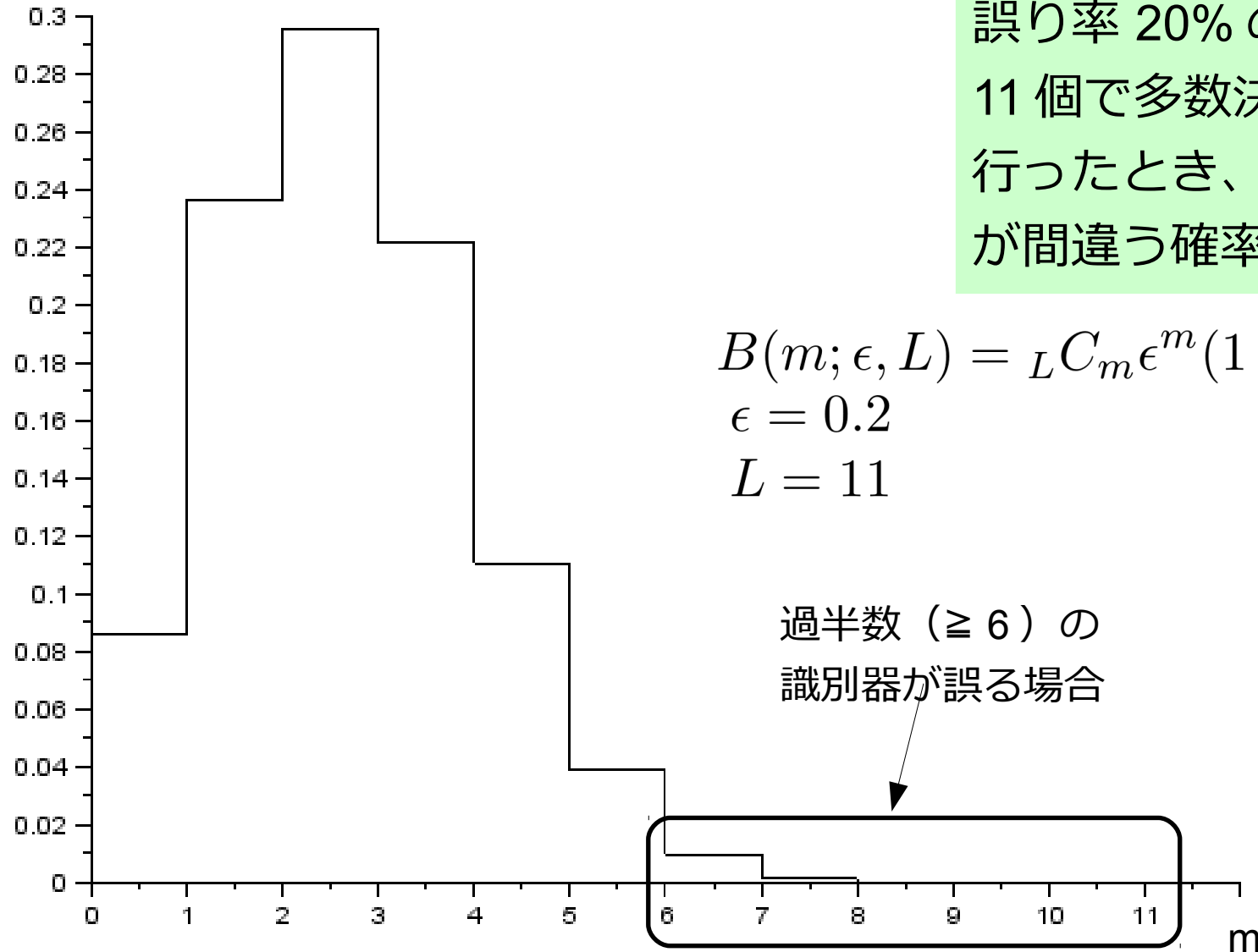
なぜ SVM が流行ったか

- 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 ϵ , 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



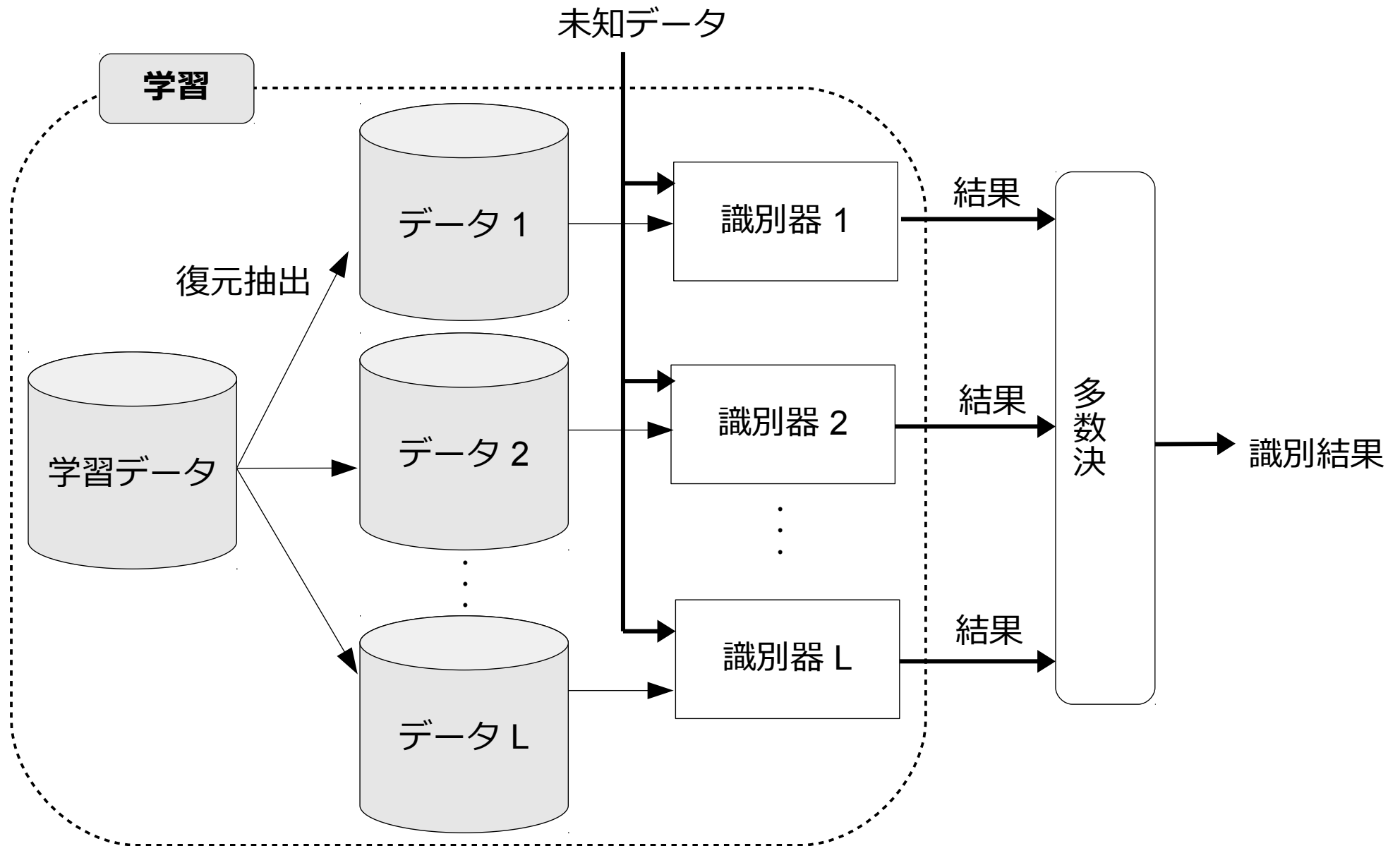
誤り率 20% の分類器
11 個で多数決統合を
行ったとき、6 個以上
が間違ふ確率は **1.2%**

$$B(m; \epsilon, L) = {}_L C_m \epsilon^m (1 - \epsilon)^{L-m}$$

$\epsilon = 0.2$
 $L = 11$

過半数 (≥ 6) の
識別器が誤る場合

9.2 バギング



9.2 バギング

- 特徴

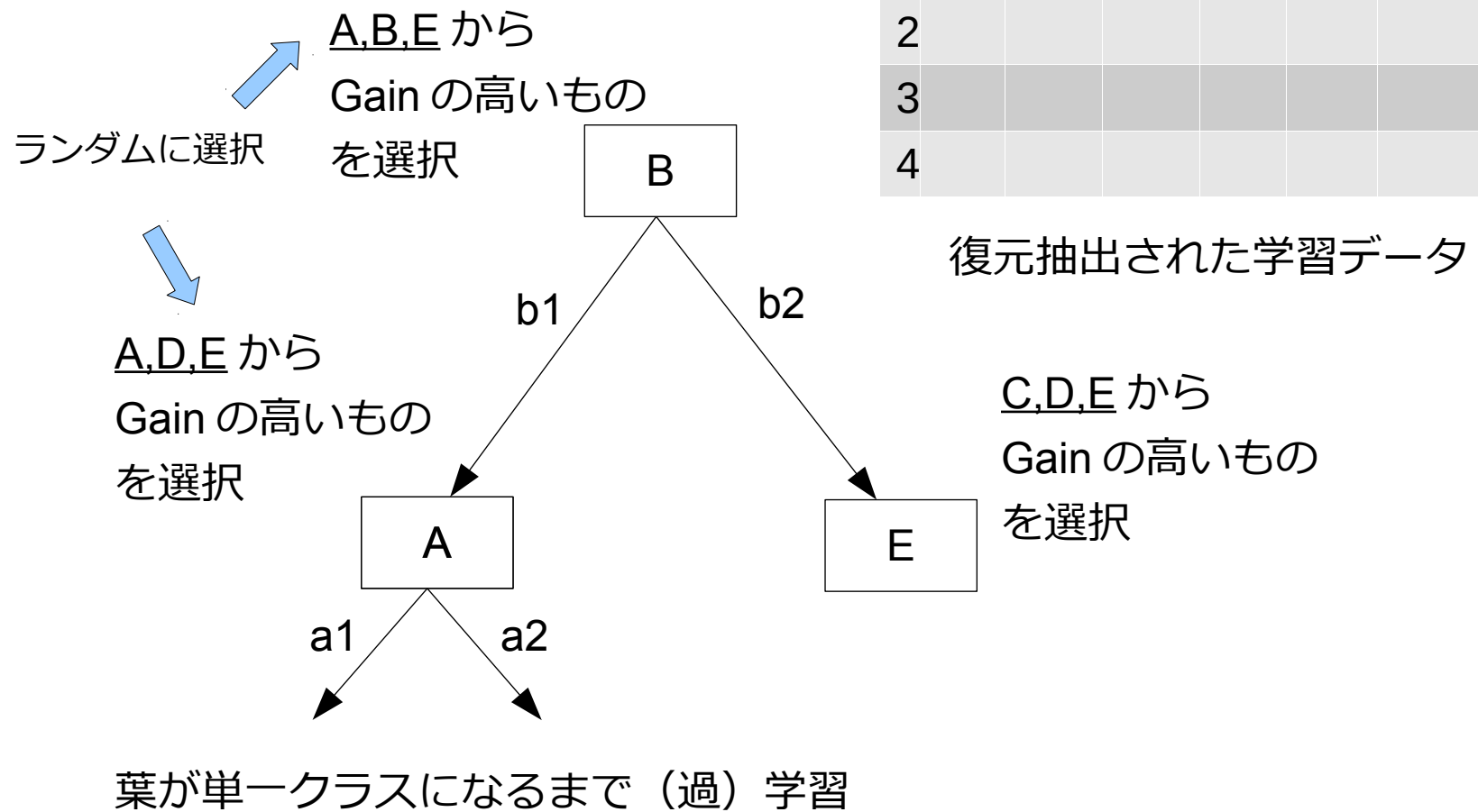
- 訓練例から復元抽出することで、元のデータと同じサイズの独立なデータ集合を作成する。

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



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

The screenshot shows the Weka Explorer interface with the 'Classify' tab selected. The classifier chosen is 'RandomForest' with parameters '-I 3 -K 0 -S 1 -print -num-slots 1'. The 'Test options' section shows 'Use training set' selected. The 'Result list' on the left shows three entries for 'trees.RandomForest' at different times. The 'Classifier output' pane on the right displays the output of the first two trees. Two red arrows point to the output of the first two trees, with the text '異なった決定木が学習されている' (Different decision trees are being learned) written in red.

Classifier output

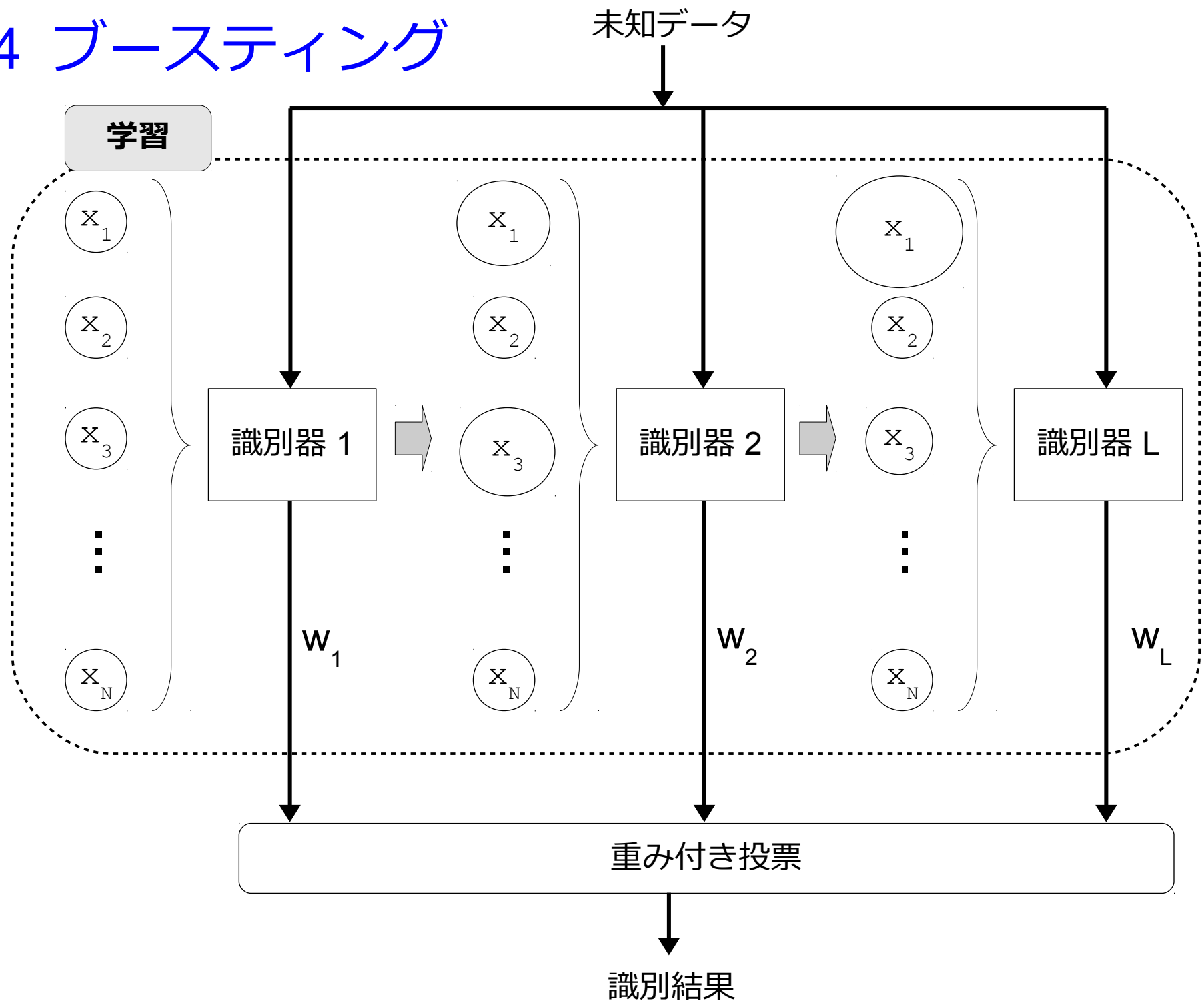
All the base classifiers:

```
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

9.4 ブースティング



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