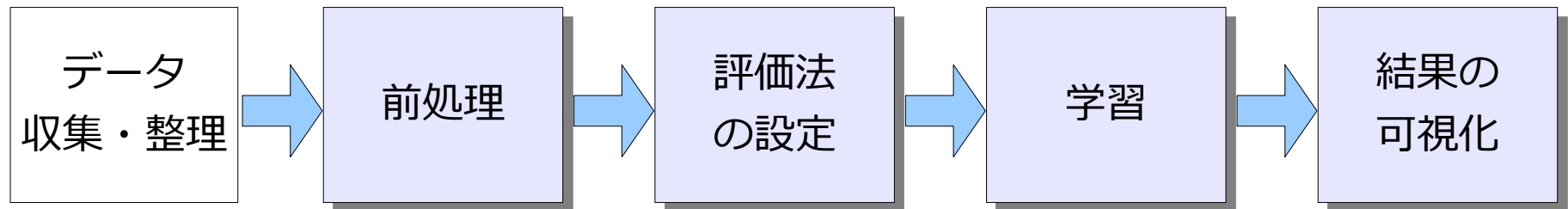



## 2. 機械学習の基本的な手順 (Python)

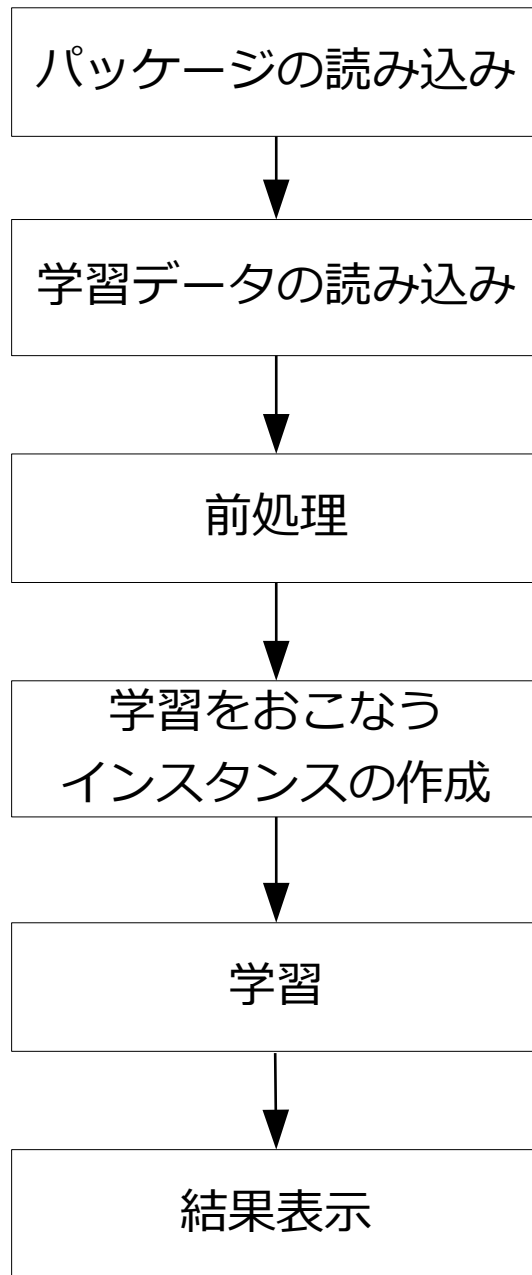


 : ライブラリによる支援が可能

## 2.2 Python による機械学習

- Python を使うメリット
  - データ処理や機械学習のパッケージが充実
    - numpy : 多次元配列を効率よく扱う
    - scipy : 高度な数値計算
    - pandas : データの読み込み・解析を支援
    - scikit-learn : 多くの機械学習アルゴリズム
    - tensorflow : 深層学習
  - グラフ表示などの可視化が容易
    - matplotlib : グラフ描画
  - Jupyter Notebook で実行手順を記録しながらコーディングが可能

## 2.2.1 scikit-learn を用いた機械学習の手順



- 組み込みデータは datasets パッケージを利用
- 外部データは pandas の read\_csv 等を利用

- 標準化 : scale
- 主成分分析 : PCA

- 学習パラメータを与えてインスタンスを作成

- fit に学習データを与えて学習
- 分割学習法では predict で予測を得る
- 交差確認法では cross\_val\_score を実行

- 分割学習法では confusion\_matrix で混同行列を求める
- 交差確認法では、結果から平均・標準偏差などを求める

## 2.2.1 scikit-learn を用いた機械学習の手順

- パッケージの読み込み

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.decomposition import PCA
from sklearn import preprocessing
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import classification_report
```

パッケージ全体を読み込むとき

`import パッケージ名 as 略称`

`from パッケージ名 import ~` は  
サブパッケージの読み込みにも使う

特定のクラスやメソッドを読み込むとき

`from パッケージ名 import ~`

## 2.2.2 データの読み込み

- サンプルデータ : iris
  - 3 種類のアヤメ ( setosa, versicolor, virginica )  
を萼 ( がく ;sepal) の長さ・幅、花びら (petal) の長さ・幅の 4 つの特徴から分類する
  - 各クラス 50 事例ずつで計 150 事例のデータ数



setosa



versicolor



virginica

index	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
5	5.4	3.9	1.7	0.4
6	4.6	3.4	1.4	0.3
7	5.0	3.4	1.5	0.2
8	4.4	2.9	1.4	0.2
9	4.9	3.1	1.5	0.1

## 2.2.2 データの読み込み

- scikit-learn でのデータの持ち方
  - パターン行列：  $X$ 
    - 全データの特徴ベクトルを列方向に並べたもの
    - iris データの場合は 150 事例、 4 特徴の  $150 \times 4$  行列
  - 正解：  $y$ 
    - 正解ラベルを整数値にしてデータ数分並べたもの
    - iris データの場合は 150 個の数字（ 0,1,2 のいずれか）が並ぶベクトル

## 2.2.2 データの読み込み

- numpy の ndarray (n 次元テンソル) として読み込む方法

```
iris = load_iris()
print(iris.DESCR)
X = iris.data
y = iris.target
```

- load\_iris 関数の戻り値は Bunch オブジェクト
  - 特徴ベクトル, 正解データ, 特徴名, データの説明などのさまざまな情報が詰まったもの
- X や y は ndarray なので、scikit-learn の学習データとして用いることができる

## 2.2.2 データの読み込み

- pandas の DataFrame および Series として読み込む方法（ver 0.23 以降）
  - load\_iris 関数の引数：as\_frame=True
  - 実データでは、異常値・欠損値・記述ミス・不要な特徴の混入などへの対処が必要 → numpy では不十分
- pandas: データ分析・操作ツール
  - 統計的分析：describe, hist, ...
  - 異常値・欠損値 (NA) 処理：query, dropna, fillna, ...
- 最後に to\_numpy で ndarray に変換しておく



## 2.2.3 前処理

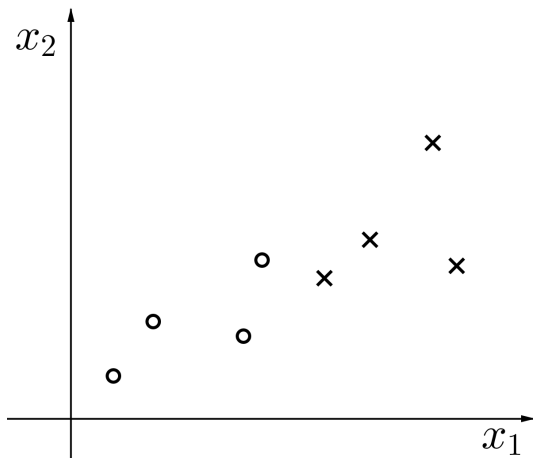
- 分析
  - 主成分分析 (PCA)
    - 高次元空間上のデータの散らばりをできるだけ保存する低次元空間へ写像を求める
    - データの次元削減に有効
  - t-SNE
    - 高次元空間での類似度を反映した低次元空間へのマッピングを求める
    - データの可視化、すなわち 2 次元または 3 次元へのマッピングに有効

```
pca = PCA(n_components=2)  
X2 = pca.fit_transform(X)
```

```
tsne = TSNE(perplexity=5)  
X3 = tsne.fit_transform(X)
```

# 主成分分析の考え方

データが多次元正規分布していると仮定



共分散行列 $\Sigma$ の計算

$\bar{x}_1, \bar{x}_2$  : 平均値、 $N$  : データ数

対角成分は分散、  
非対角成分は相関を表す

$$\Sigma = \frac{1}{N} \begin{pmatrix} \sum (x_1 - \bar{x}_1)^2 & \sum (x_1 - \bar{x}_1)(x_2 - \bar{x}_2) \\ \sum (x_1 - \bar{x}_1)(x_2 - \bar{x}_2) & \sum (x_2 - \bar{x}_2)^2 \end{pmatrix}$$

$\Sigma$ は

半正定値(→固有値がすべて0以上の実数)

対称行列(→固有ベクトルが実数かつ直交)

であるので、以下のように分解できる

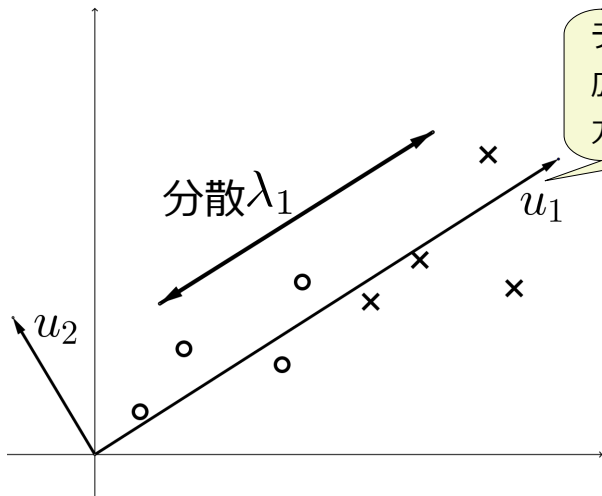
$$\Sigma' = U^T \Sigma U = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$$

$\lambda$ は固有値の大きい順、 $U$ は対応する固有ベクトル $U_1, U_2$ を並べたもの

$\lambda_1$ に対応する固有ベクトル $U_1$ で  
2次元データを1次元に射影

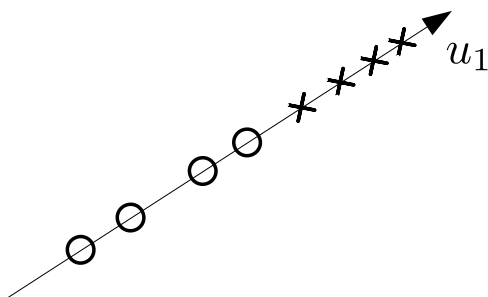
$$u_1 = U_1^T x$$

$$\text{寄与率} = \frac{\lambda_1}{\lambda_1 + \lambda_2}$$



分散 $\lambda_1$

データが一番  
広がっている  
方向



# t-SNE の考え方

- 高次元空間
  - データ  $x_i$  と  $x_j$  の類似度  $p_{ij}$  を正規分布で表現
    - 正規分布の分散（どのくらいの数のデータを類似度計算の対象である「近く」とみなすか）をパラメータとして与える
- 低次元空間
  - データ  $y_i$  と  $y_j$  の類似度  $q_{ij}$  を t 分布で表現
    - t 分布は正規分布よりも値の大きい範囲が広い
- 最適化
  - $p_{ij}$  と  $q_{ij}$  を近くするために、分布の距離 (KL-divergence)

$$KL(P, Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

を最小化するように  $Y=\{y_1, \dots, y_n\}$  の位置を逐次更新

## 2.2.3 前処理

- 特徴のスケーリング
  - 特徴の各次元のスケールが著しく異なると、特徴の扱いが不公平になる
- 標準化：すべての次元を平均 0、分散 1 に揃える
  - 各次元（軸）に対して平均値を引き、標準偏差で割る

$$x'_i = \frac{x_i - m_i}{\sigma_i} \quad m_i, \sigma_i : \text{軸 } i \text{ の平均、標準偏差}$$

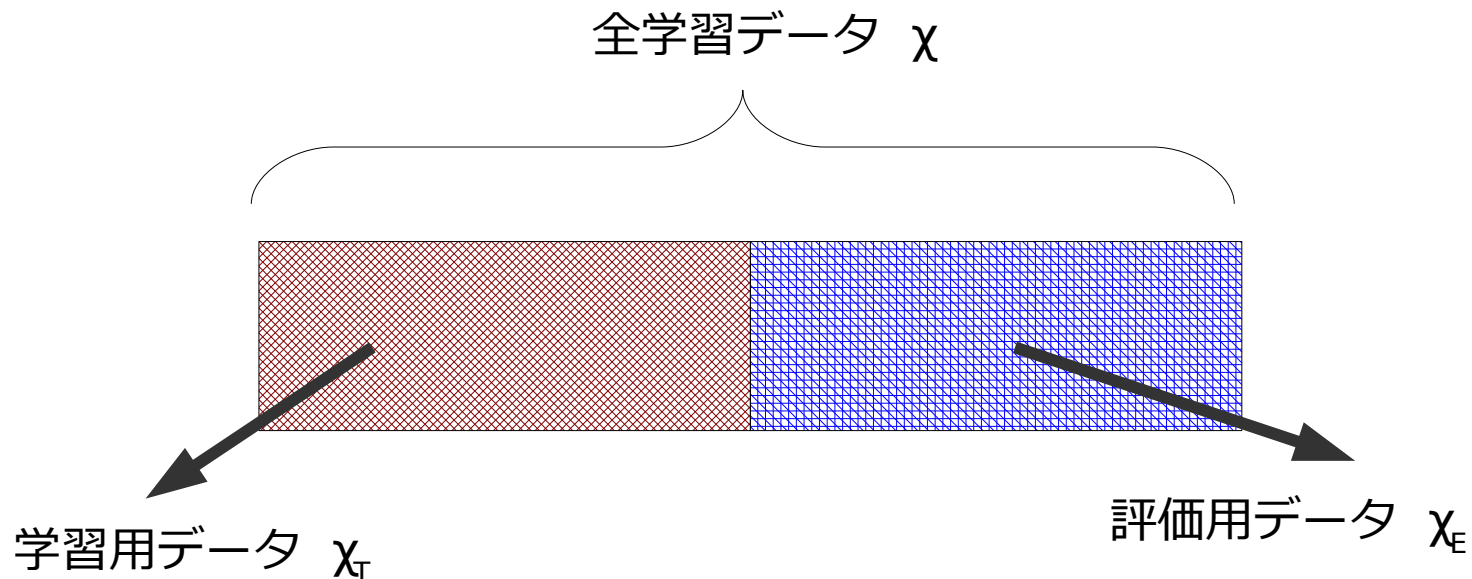
```
X_scaled = preprocessing.scale(X)
```

## 2.2.4 評価基準の設定と学習

- 分割学習法
  - データを学習用と評価用に適切な割合で分ける
  - ハイパーパラメータを調整する場合は、学習用・検証用・評価用に分ける
- 交差確認法
  - データを  $m$  個の集合に分割し、 $m-1$  個の集合で学習、残りの 1 個の集合で評価を行う
  - 評価する集合を入れ替え、合計  $m$  回評価を行う
  - 分割数をデータ数とする場合を一つ抜き法とよぶ

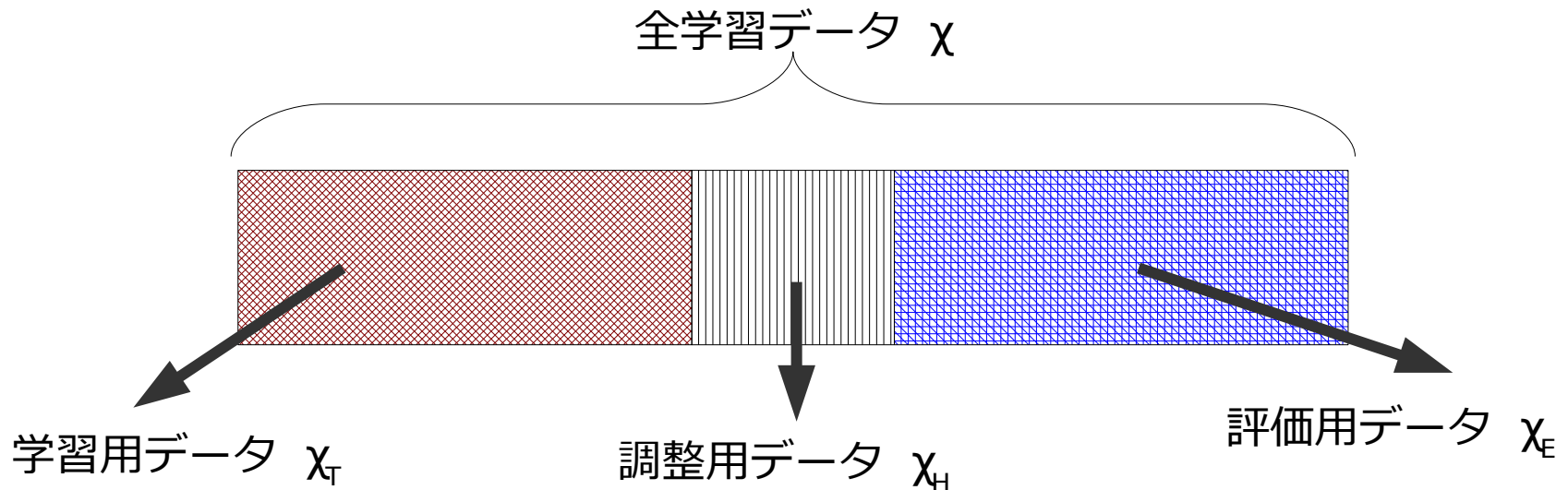
# 評価基準の設定

- 分割学習法（１）
  - 全学習データ  $\chi$  を学習用データ集合  $\chi_T$  と評価用データ集合  $\chi_E$  に分割する
  - $\chi_T$  を用いて識別機を設計し、 $\chi_E$  を用いて誤識別率を推定する



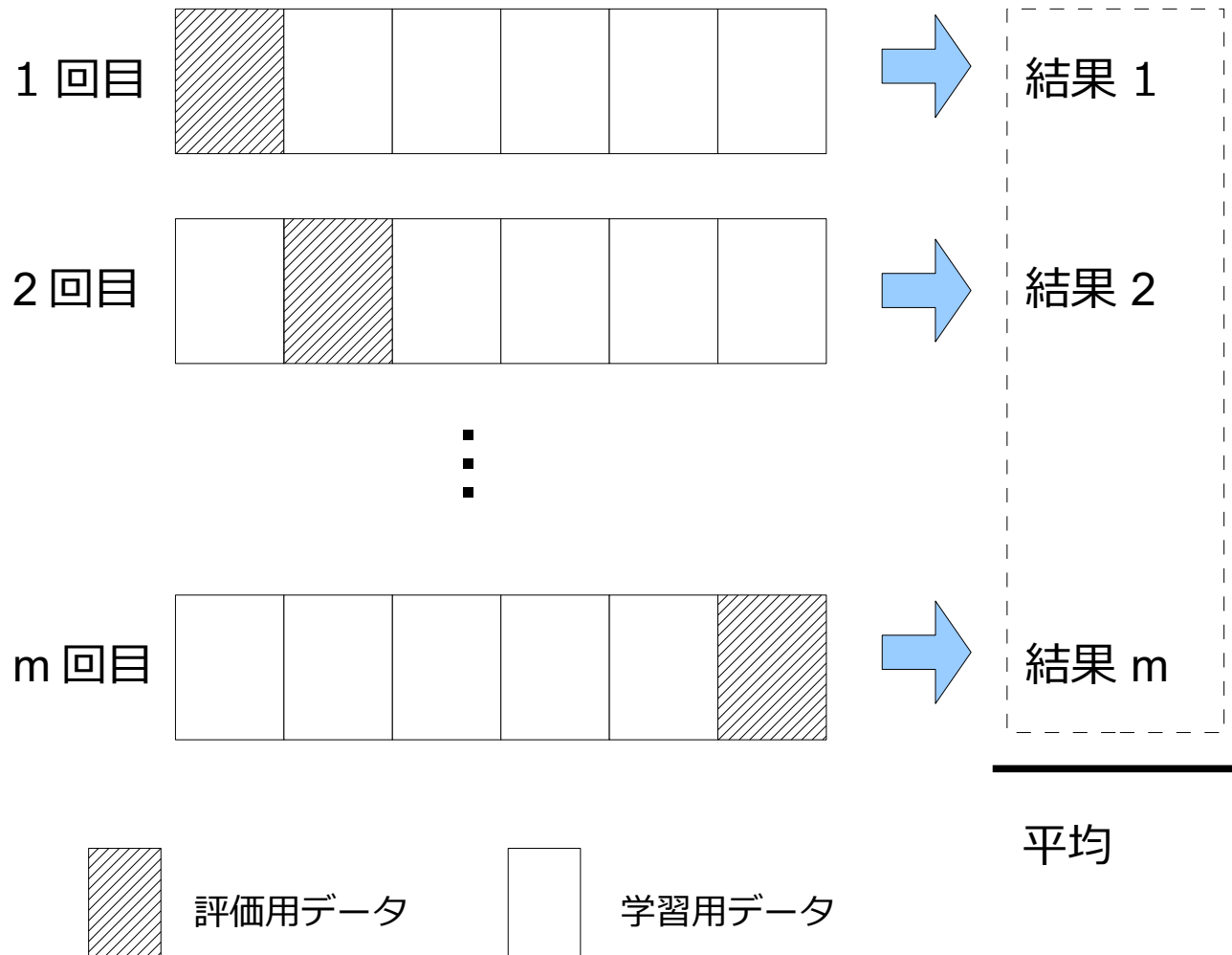
# 評価基準の設定

- 分割学習法（2）
  - 全学習データ  $\chi$  を学習用データ集合  $\chi_T$ 、調整用データ集合  $\chi_H$ 、評価用データ集合  $\chi_E$  に分割する
  - $\chi_T$  を用いて識別機を設計、 $\chi_H$  を用いてハイパーパラメータを調整、 $\chi_E$  を用いて誤識別率を推定する



# 評価基準の設定

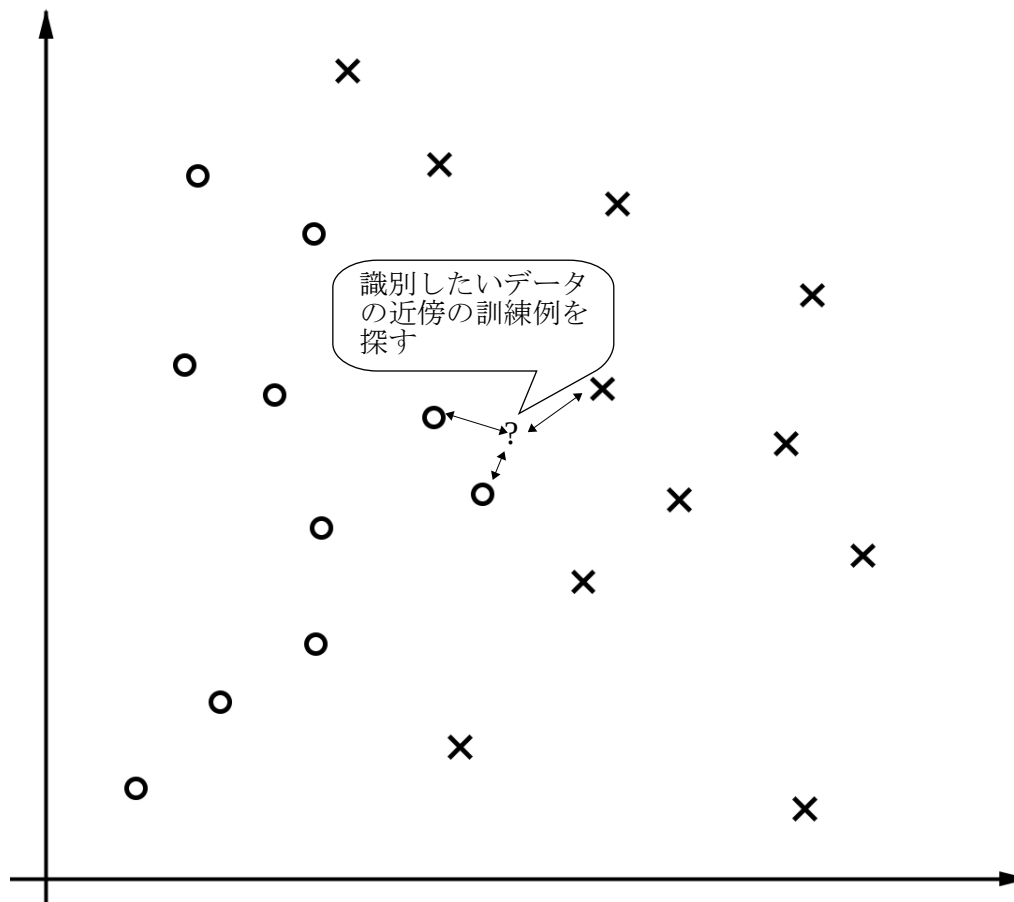
- 交差確認法





## 2.2.4 評価基準の設定と学習

- k-NN 法
  - 識別したいデータの近傍の  $k$  個の学習データを探し、属するクラスの多数決で識別



「正解付きデータを多数集めること」= 学習  
とみなす方法

## 2.2.4 評価基準の設定と学習

- k-NN 法のパラメータ
  - 近傍として探索するデータ数：k
    - k が 1 の場合にもっとも複雑な境界
    - k が増えるに従って境界は滑らかになるが、あまり大きいと識別性能が低下する
  - 距離尺度
    - 通常はユークリッド距離
    - 値を持つ次元が少ない場合はマンハッタン距離
  - 探索方法
    - 入力と全データとの距離を計算してソート
    - データが多い場合は事前にデータを木構造化

## 2.2.5 結果の表示

- 学習したモデル
  - 式、木構造、ネットワークの重み、 etc.
- 性能
  - 正解率、適合率、再現率、 F 値
  - グラフ
    - パラメータを変えたときの性能の変化
    - 異なるモデルの性能比較

## 2.2.5 結果の表示

- 混同行列

	予測+	予測-
正解+	true positive(TP)	false negative(FN)
正解-	falsepositive(FP)	true negative(TN)

- 正解率  $Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$

- 適合率  $Precision = \frac{TP}{TP + FP}$

「精度」から変更

- 再現率  $Recall = \frac{TP}{TP + FN}$

- F 値  $F-measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}$

正解の割合  
クラスの出現率に  
偏りがある場合は不適

正例の判定が  
正しい割合

正しく判定された  
正例の割合

トレードオフ

適合率と再現率の  
調和平均

# パイプライン

- パイプラインとは
  - 複数の前処理と学習モジュールなど、連続した処理をパイプラインとして結合して、ひとつの識別器のインスタンスとみなせる

```
estimators = [('reduce_dim', PCA()), ('clf', KNeighborsClassifier())]  
pipe = Pipeline(estimators)
```

- パイプラインのメリット
  - 処理をカプセル化して実行を簡単にできる
  - ハイパーパラメータ調整を一度に行える
  - テストデータが混入していないことを保証できる