パターン認識と学習
最近傍法

管理工学科
篠沢佳久

資料の内容

- パターン認識の基本的な流れ
  - 最近傍法（k近傍法）
- 最近傍法の例題

パターン認識の基本的な流れ

最近傍法

特徴空間

指紋照合

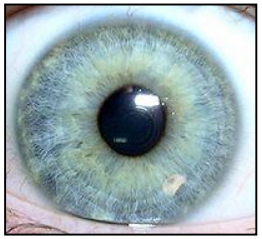
- 生体（バイオメトリクス）認証

どれと一致しますか*

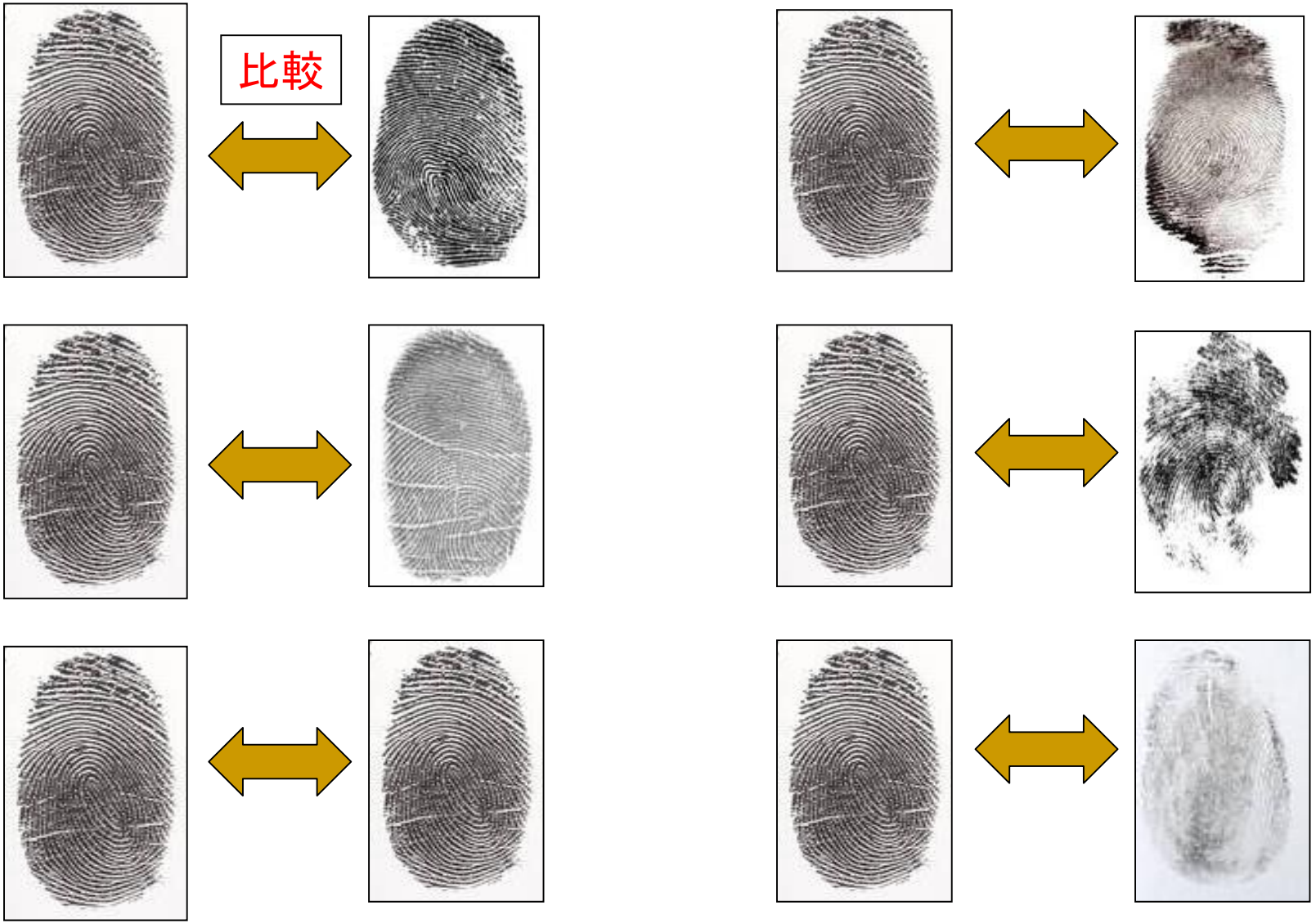
*指紋の画像の大きさ（縦，横）は全て同じです

生体（バイオメトリクス）認証

- 個体（人）によって異なる性質を利用して，個人認証を行なう
  - DNA
  - 指紋
  - 虹彩
  - 静脈（手のひら）
  - 顔
  - 耳
  - 音声
  - 筆跡



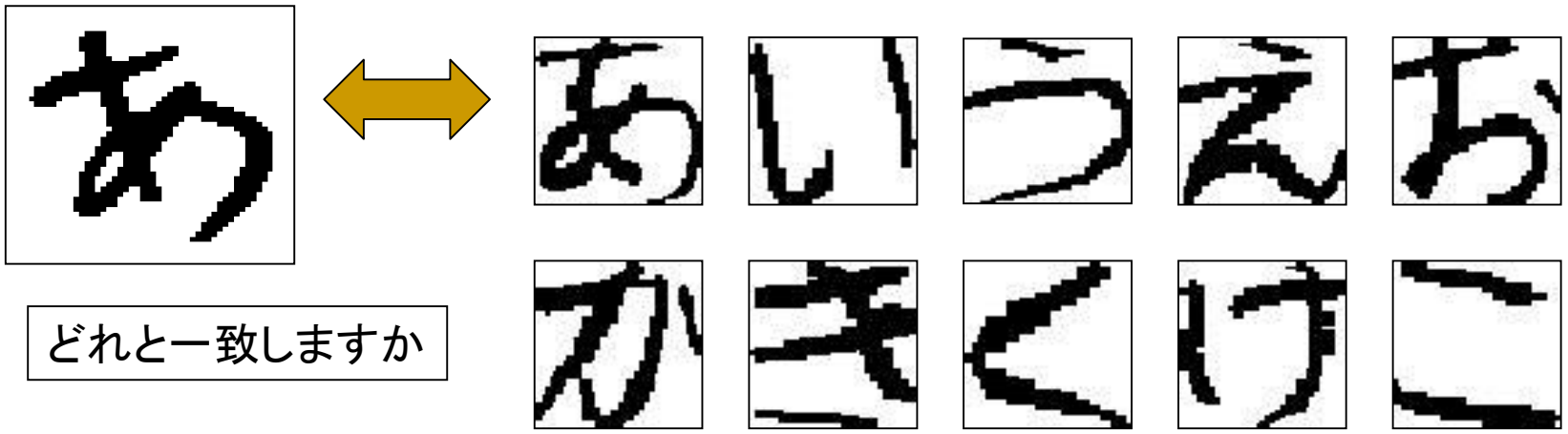
虹彩
（Wekipediaより）



照合したい指紋と全ての対象候補と比較を行なう
一致（類似）しているかを調べ，一致した（最も類似した）ものを認識結果とする

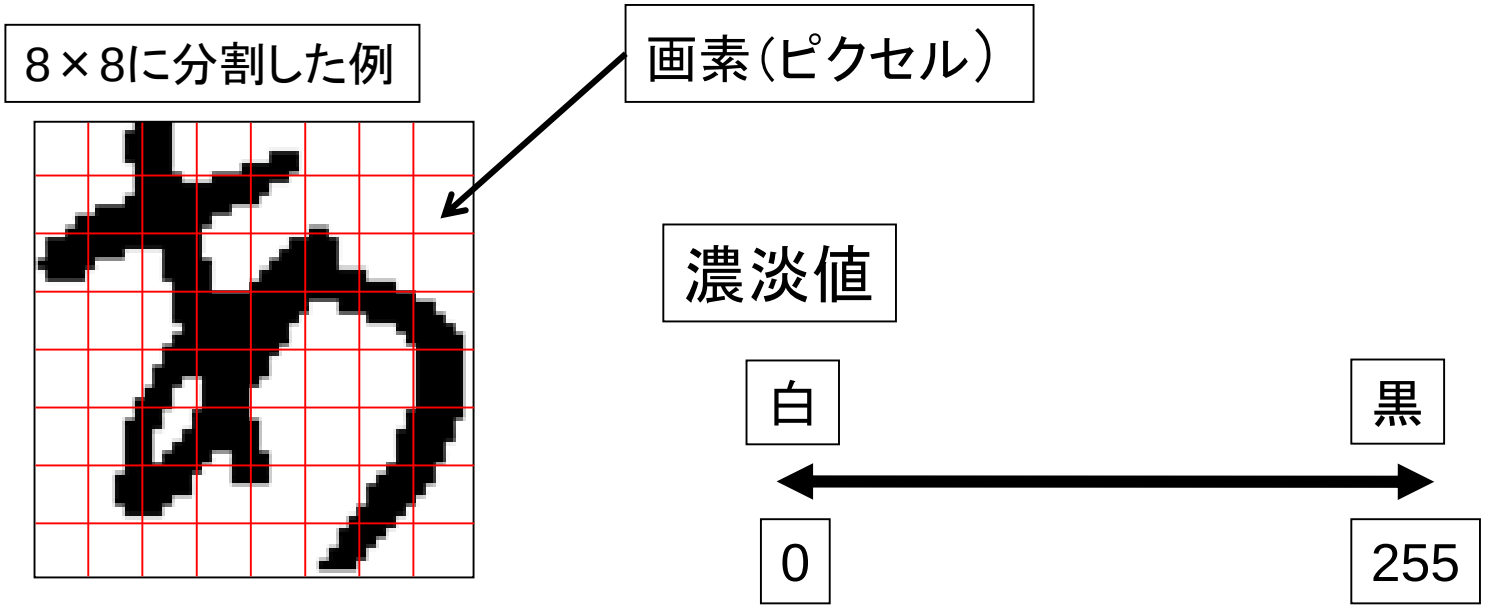
パターン認識の基本的な流れ

- 文字画像認識
  - 文字画像データベースETL9B（産業技術総合研究所）



標本化と量子化処理

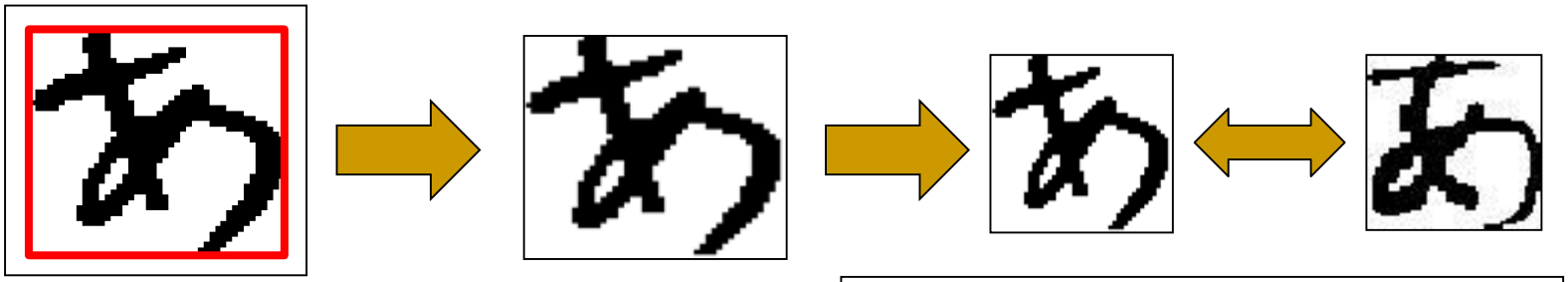
- N×N個に均等に分割（標本化処理）



- 各画素において，白と黒の割合（濃淡）を（例えば）256段階（0～255の値）で示す（量子化処理）

前処理

- 文字（と考えられる）領域を切り出す
- 文字の大きさを一定になるように変換
  - 例えばアフィン変換を用いる
- これを前処理（もしくは正規化処理）と呼ぶ



切り出し

比較したい文字画像と同じ大きさにする（この場合は縮小）

特徴ベクトル①

- 文字の濃淡値はN×Nの行列で表現できる

$$X = \begin{pmatrix} x_{11} & x_{21} & \cdots & x_{N1} \\ x_{12} & x_{22} & & x_{N2} \\ \vdots & & \ddots & \vdots \\ x_{1N} & x_{2N} & \cdots & x_{NN} \end{pmatrix}$$

x_{ij}：(i,j)メッシュの濃淡値
0～255の値

- 通常，カラー画像の場合と同様にベクトルで表現

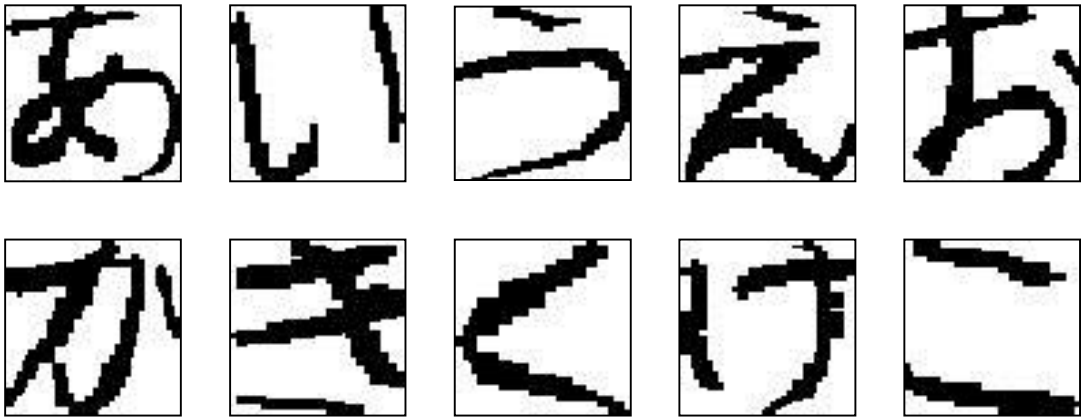
$$\mathbf{x}^t = (x_1, x_2, \cdots, x_{NN})$$

特徴ベクトル

特徴ベクトル②

調べたい文字の特徴ベクトル　$\mathbf{t}$

ベクトル$\mathbf{t}$と10個のベクトル$\mathbf{x}_p$の「類似度」をそれぞれ計算し，最も「類似度」の高い文字を認識結果とする



比較したい文字の特徴ベクトル

$$\mathbf{x}_p\,(p=1,2,\cdots,10)$$

特徴ベクトル③

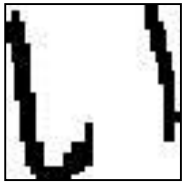
$$\mathbf{t} = (3,14,2,\cdots 12,2)^t$$

64次元

$$\mathbf{x}_1 = (5,11,4,\cdots 15,3)^t$$

64次元

$$\mathbf{x}_4 = (1,8,10,\cdots 17,9)^t$$



$$\mathbf{x}_2 = (16,10,9,\cdots 6,0)^t$$

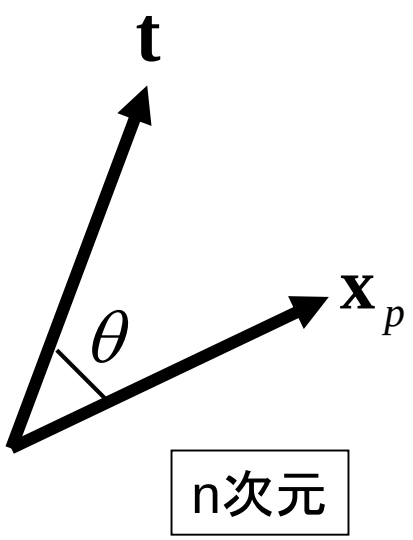
$$\mathbf{x}_3 = (1,1,2,\cdots 10,0)^t$$

$$\mathbf{x}_{10} = (5,11,10,\cdots 12,9)^t$$

（復習）類似度①

- 類似度
  - 二つのベクトルが一致する場合は $\theta = 0$
  - 従ってR$_p$は1となる

$$R_p = \cos\theta = \frac{\mathbf{t}^t \mathbf{x}_p}{\|\mathbf{t}\| \cdot \|\mathbf{x}_p\|} = \frac{\sum_{i=1}^{n} t_i x_{pi}}{\sqrt{\sum_{i=1}^{n} t_i^2} \sqrt{\sum_{i=1}^{n} x_{pi}^2}}$$



  - R$_p$が最大の画像を認識結果とする

（復習）類似度②

- 相互相関係数
  - 特徴ベクトルが n次元の場合

$$R_p' = \cos\theta = \frac{\sum_{i=1}^{n}(t_i - \bar{t})(x_{pi} - \overline{x_p})}{\sqrt{\sum_{i=1}^{n}(t_i - \bar{t})^2}\sqrt{\sum_{i=1}^{n}(x_{pi} - \overline{x_p})^2}}$$

平均値

$$\bar{t} = \frac{1}{n}\sum_{i=1}^{n} t_i$$

$$\overline{x_p} = \frac{1}{n}\sum_{i=1}^{n} x_{pi}$$

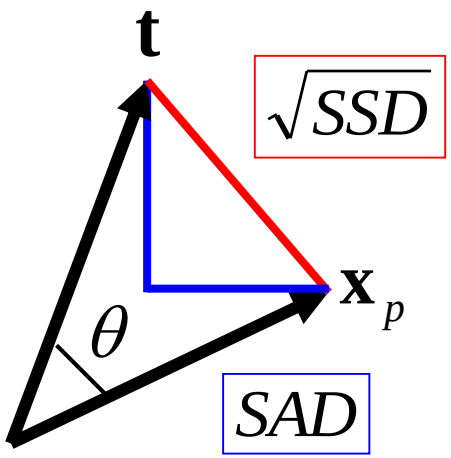
（復習）距離を用いた場合

- SSD（Sum of Squared Difference）

$$SSD = \sum_{i=1}^{n} (t_i - x_{pi})^2$$

- SAD（Sum of Absolute Difference）

$$SAD = \sum_{i=1}^{n} |t_i - x_{pi}|$$



- 距離を用いた場合，SSDもしくはSADが最小の画像を認識結果とする

（復習）距離尺度

- ユークリッド距離

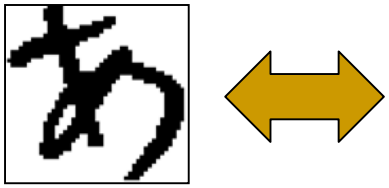
$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

- べき乗距離（ミンコスキー距離）

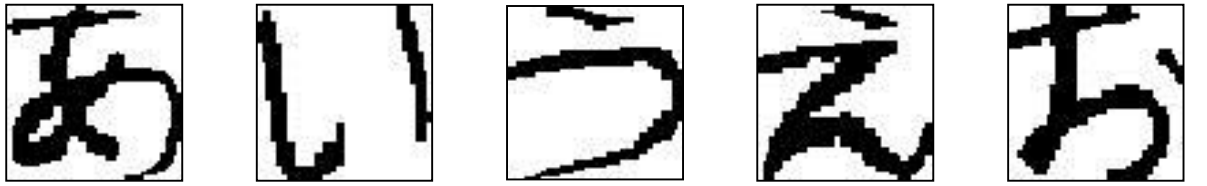
$$\left(\sum_{i=1}^{n}|x_i - y_i|^p\right)^{\frac{1}{r}}$$

p=r=1の場合，マンハッタン距離
p=r=2の場合，ユークリッド距離

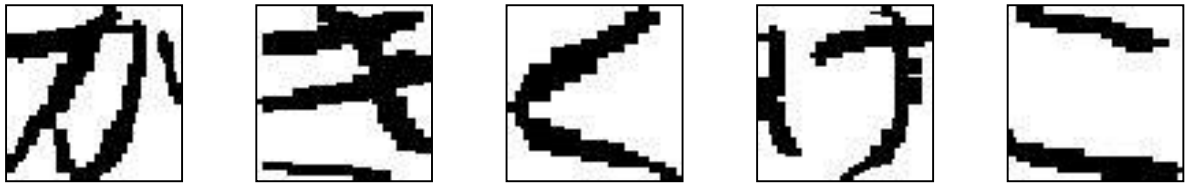
相互相関係数による類似度



相互相関係数を求めると…



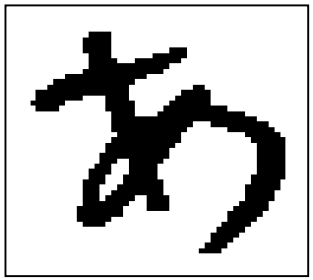
0.902	0.231	0.554	0.612	0.794



0.651	0.428	0.415	0.275	0.327

認識結果

最近傍法

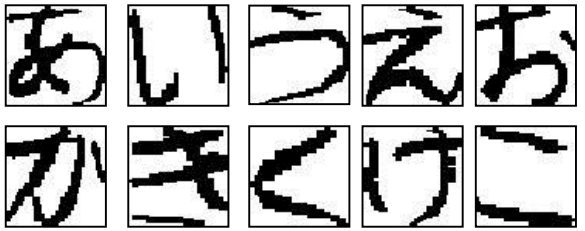


入力パターン

前処理 → 特徴抽出 → 認識処理 → 「あ」

マッチング

辞書



辞書中のマッチングされるパターンは「プロトタイプ」と呼ばれる*

辞書内のプロトタイプとマッチングを行ない，最も類似度の高いものを結果とする方法を最近傍法（Nearest Neighbor）と呼ぶ

*テンプレートとも呼ばれますが，テンプレートマッチングと混同するので講義ではプロトタイプと呼びます

最近傍法のアルゴリズム（類似度を用いる場合）

```
max = －∞
前処理（入力パターン）
t = 特徴抽出（前処理後の入力パターン）

for( p = 0 ; p < Max_p ; p++ ) {
    similarity = 類似度（t, x_p）
    if( similarity > max ) {
        max = similarity
        answer = p
    }
}
プロトタイプ answer が認識結果
```

Max_p : プロトタイプの総数
t : 入力パターンの特徴ベクトル
$\mathbf{x}_p$: プロトタイプ p の特徴ベクトル

最近傍法のアルゴリズム（距離を用いる場合）

```
min = ∞
前処理（入力パターン）
t = 特徴抽出（前処理後の入力パターン）

for( p = 0 ; p < Max_p ; p++ ) {
    dist = 距離（t, x_p）
    if( dist > min ) {
        min = dist
        answer = p
    }
}
プロトタイプ answer が認識結果
```

Max_p : プロトタイプの総数
t : 入力パターンの特徴ベクトル
$\mathbf{x}_p$: プロトタイプ p の特徴ベクトル

認識する前にすべきこと

- 「辞書」の作成が必要
  - プロトタイプの作成
  - 前処理
  - 特徴の抽出

- 認識したいパターン
  - プロトタイプと同様に処理を行ない，同じ特徴を抽出

特徴ベクトルによる類似性

- 二つのパターン間の類似性
  - 入力パターン，辞書中のパターン（プロトタイプ）は特徴ベクトルによって表現される
  - パターン間の類似性は特徴ベクトルの類似度によって求めることができる
- 認識結果
  - 類似度の場合 → 最大となるプロトタイプ
  - 距離の場合 → 最小となるプロトタイプ

特徴空間①

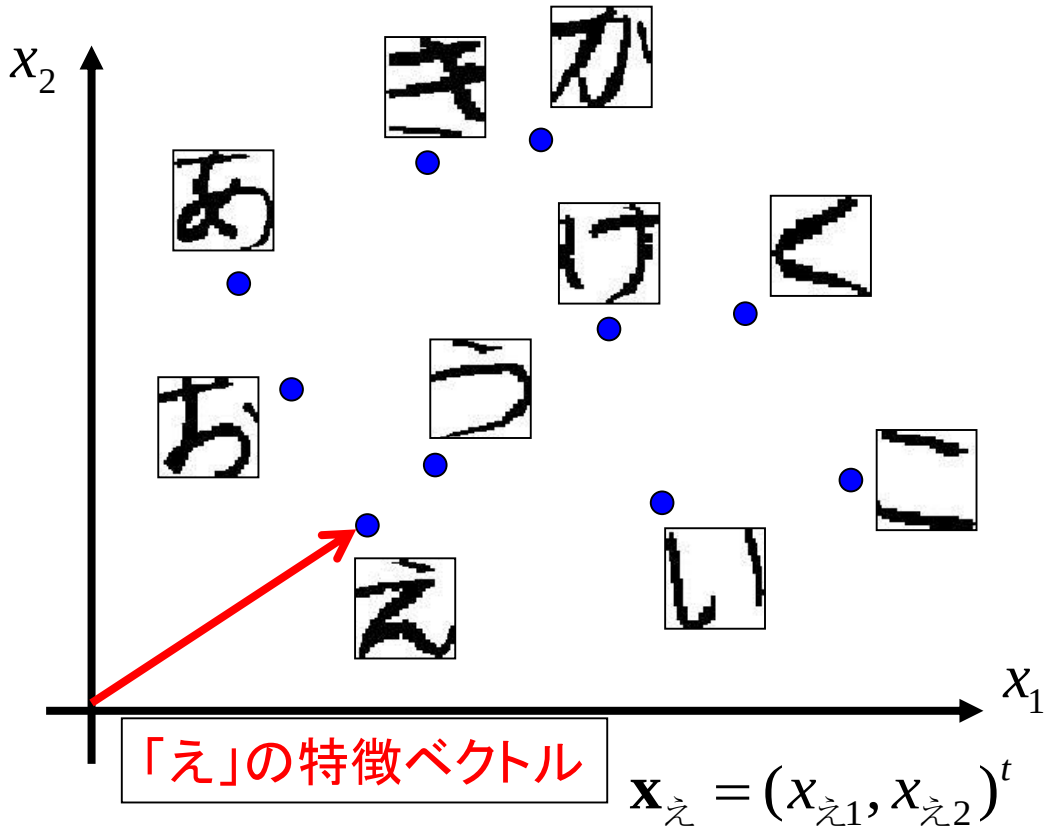
- 特徴空間
  - 特徴ベクトルによって張られる空間
  - 各パターンの特徴ベクトルは，特徴空間上の一点としても表現できる
  - 特徴空間上では，特徴ベクトルが類似しているパターンは近くに，類似していないパターンは遠くに配置される

→ 塊（クラスター）ができる

→ 必ずしも同種のパターンによってクラスターが生成されるとは限らない

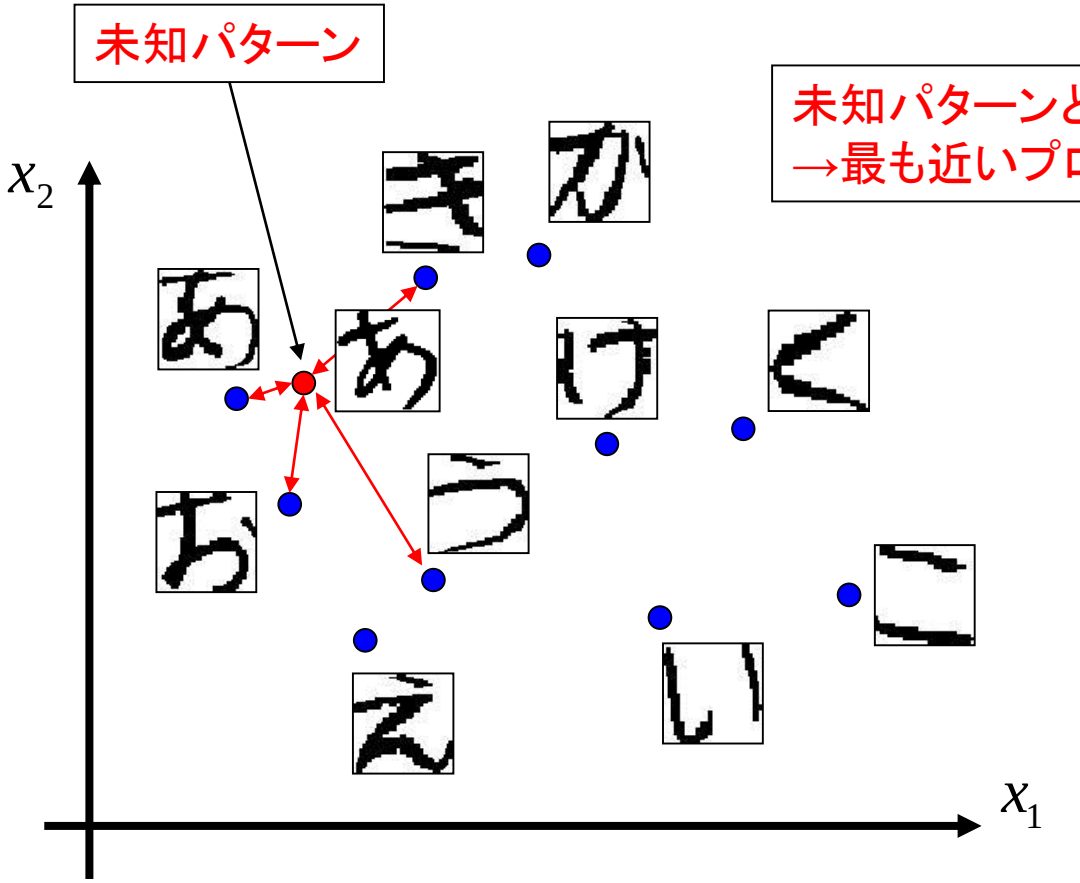
特徴空間②

- 仮に二次元上で表現できたとしたら…



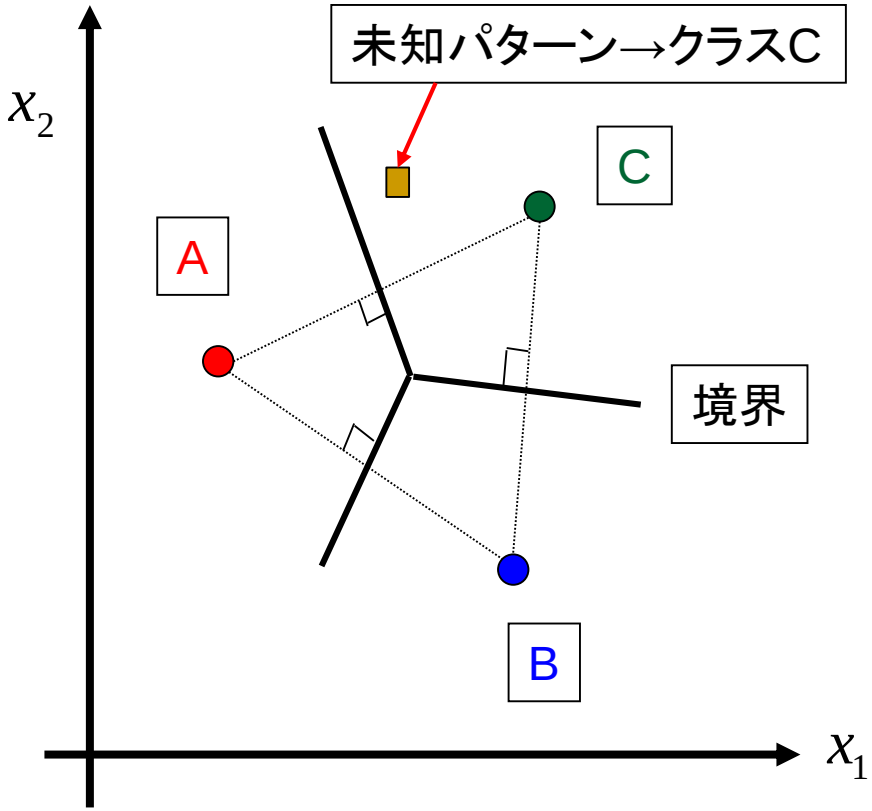
最近傍法の意味①

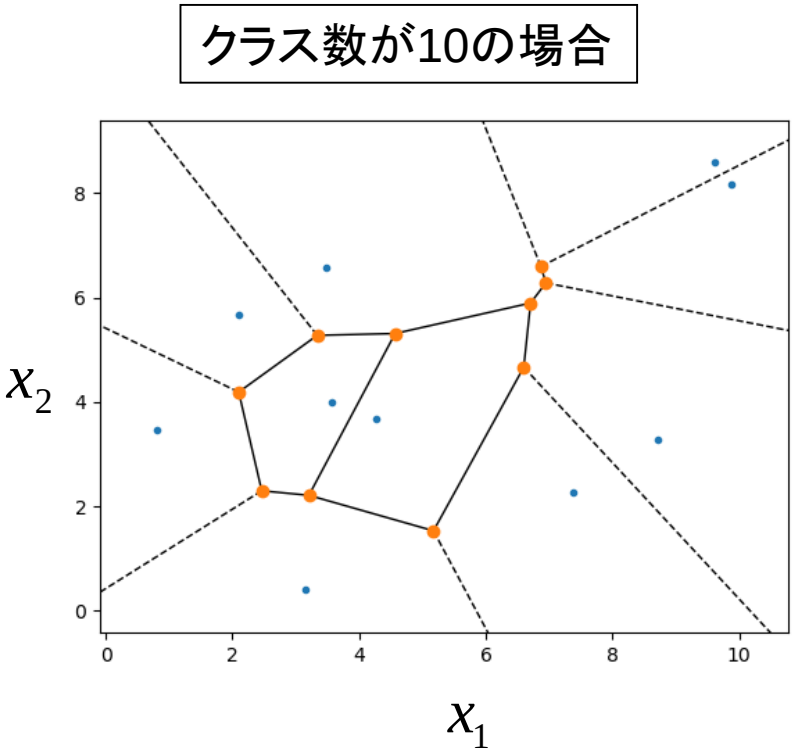
- 特徴空間上で最も距離の近いプロトタイプを探索



最近傍法の意味②

- ボロノイ図





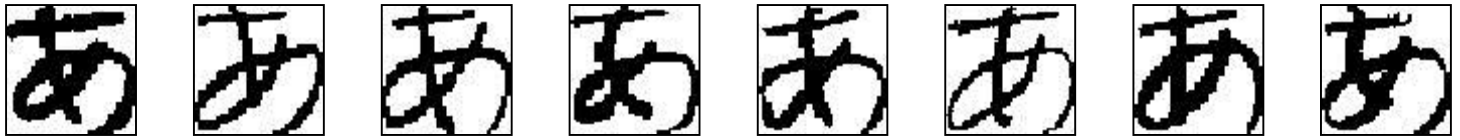
プロトタイプ

- （例えば）「あ」という文字
  - 書き手によって千差万別
  - 辞書中のプロトタイプ「あ」によって認識結果が異なる場合もある

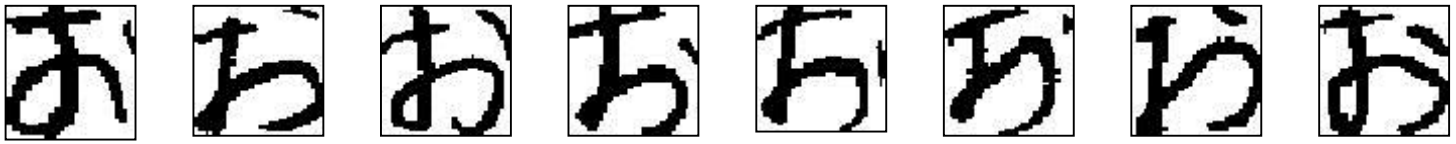
- 辞書中のプロトタイプ「あ」はどのように作成すればよいか？
  - 複数個のプロトタイプを用意

複数個のプロトタイプ

- 「あ」の場合

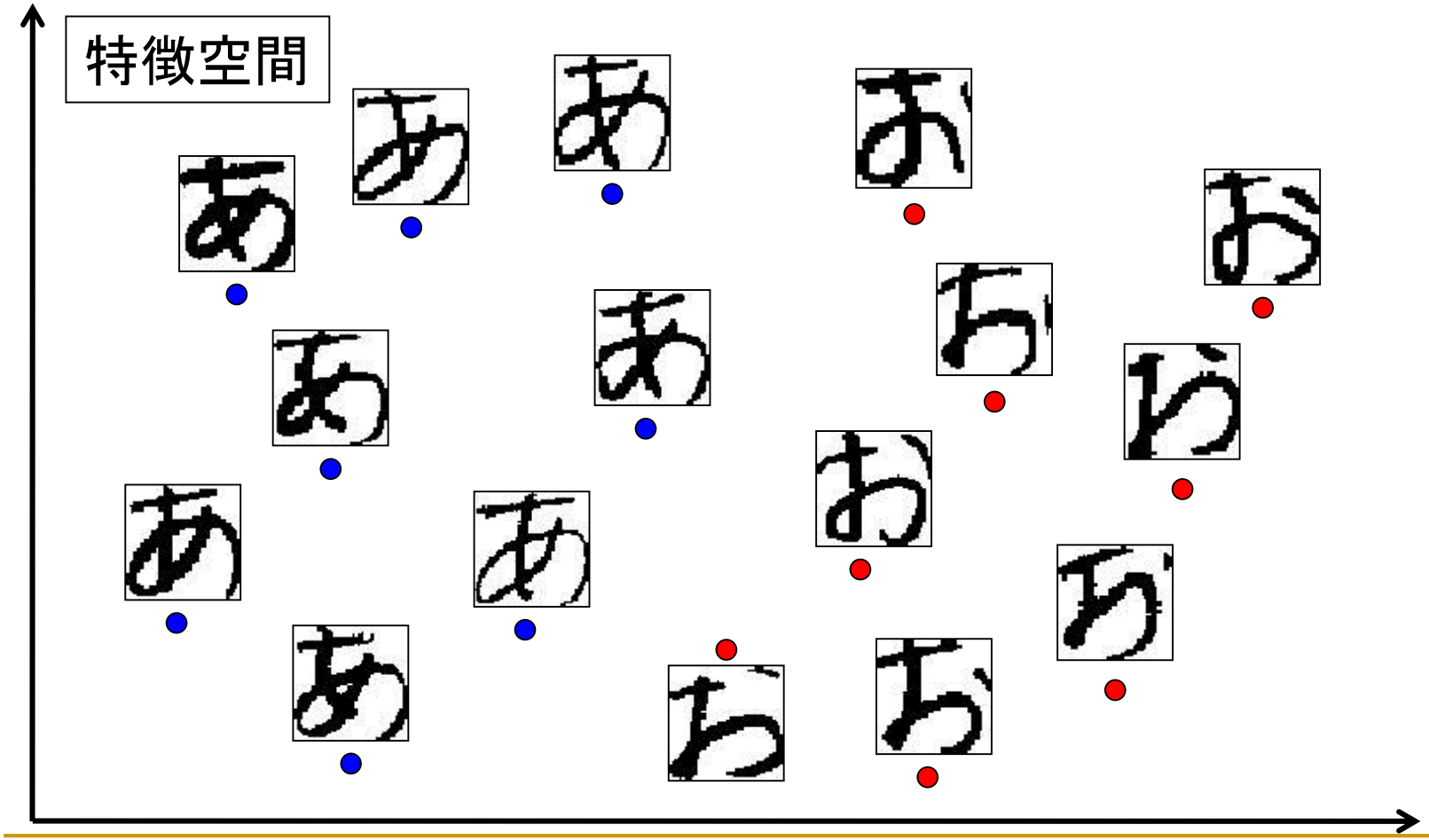


- 「お」の場合

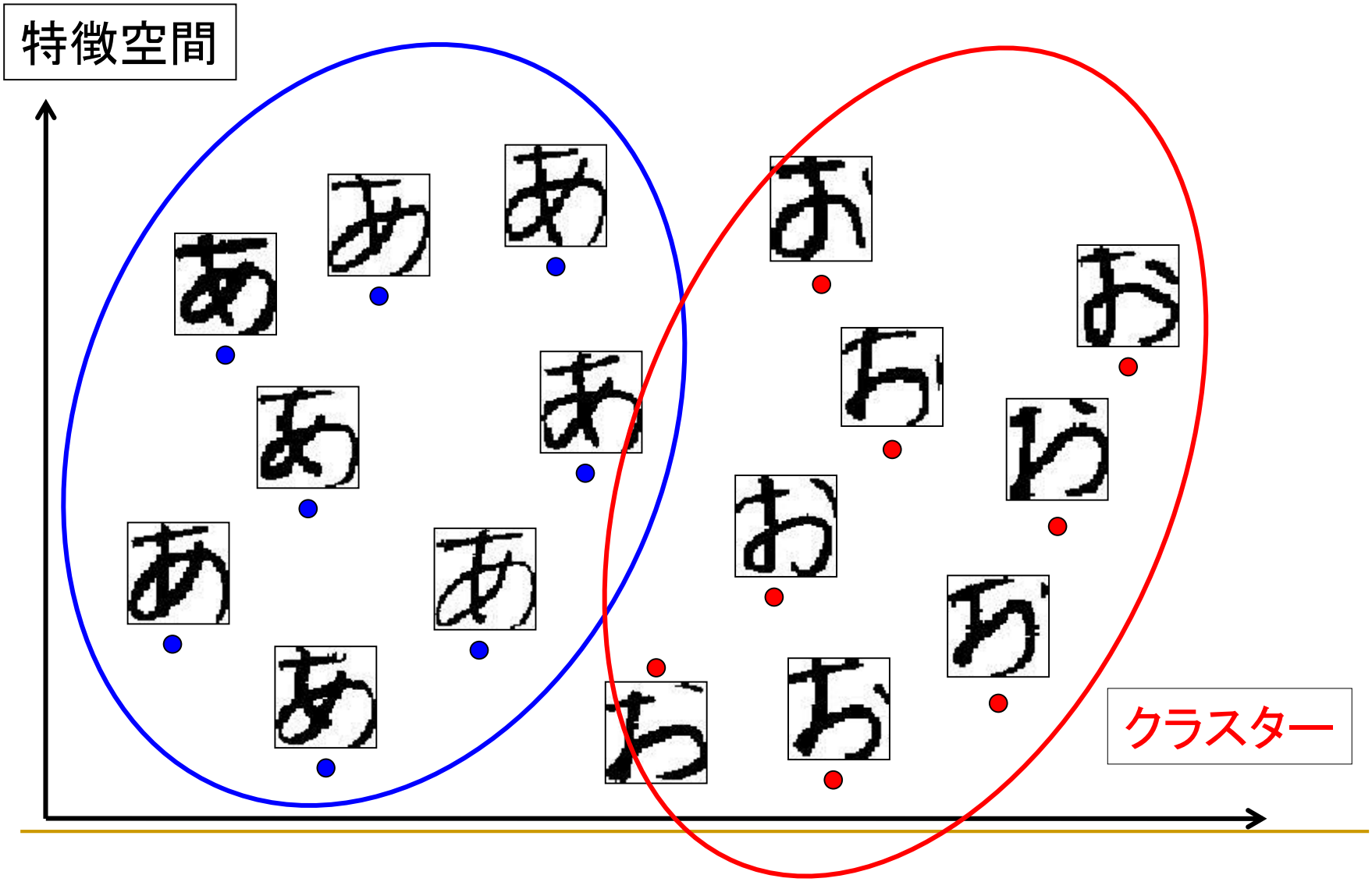


複数個のプロトタイプを利用した最近傍法①

一つのパターンについて複数個のプロトタイプを用意

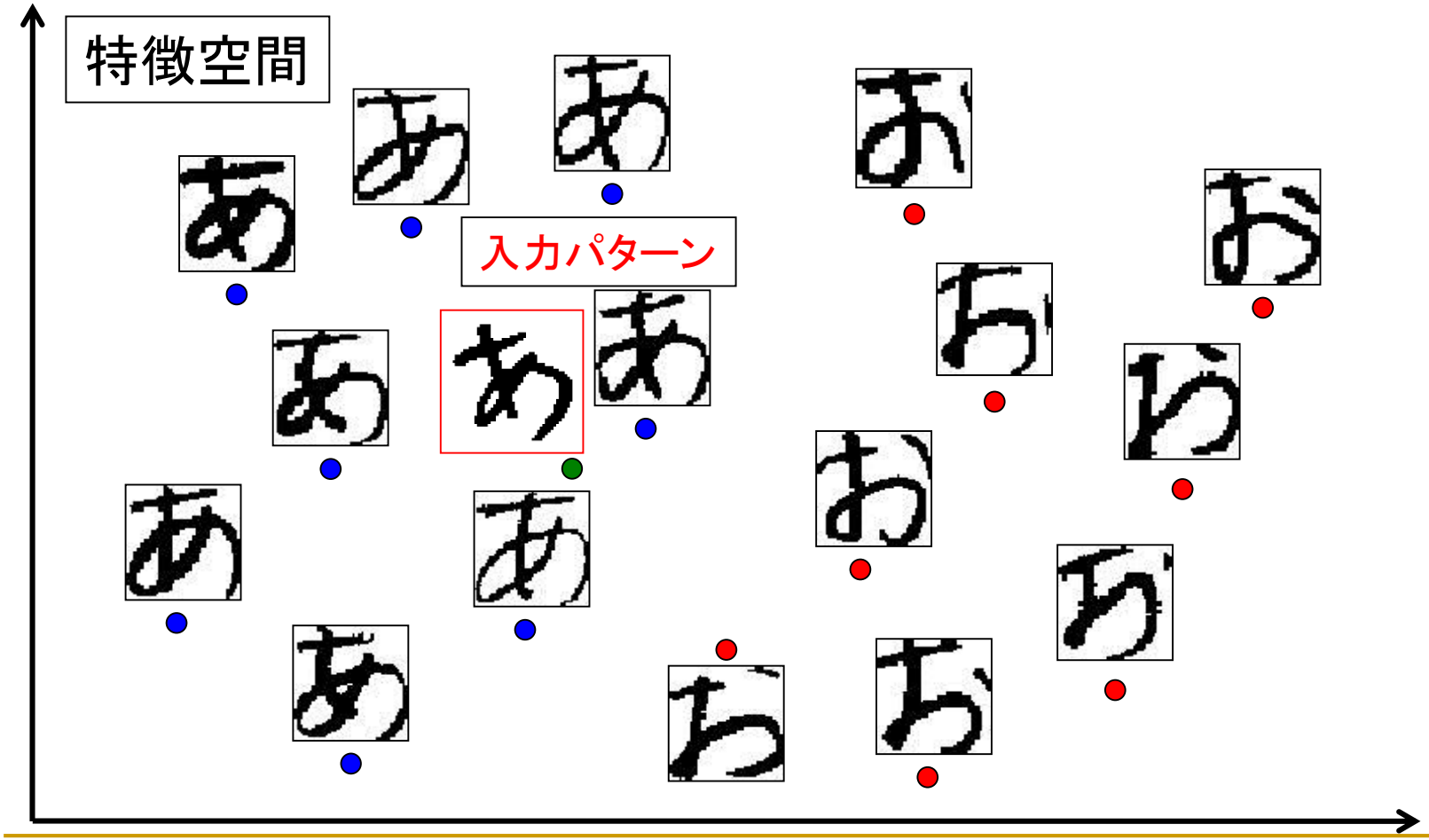


特徴空間上でのパターンの分布

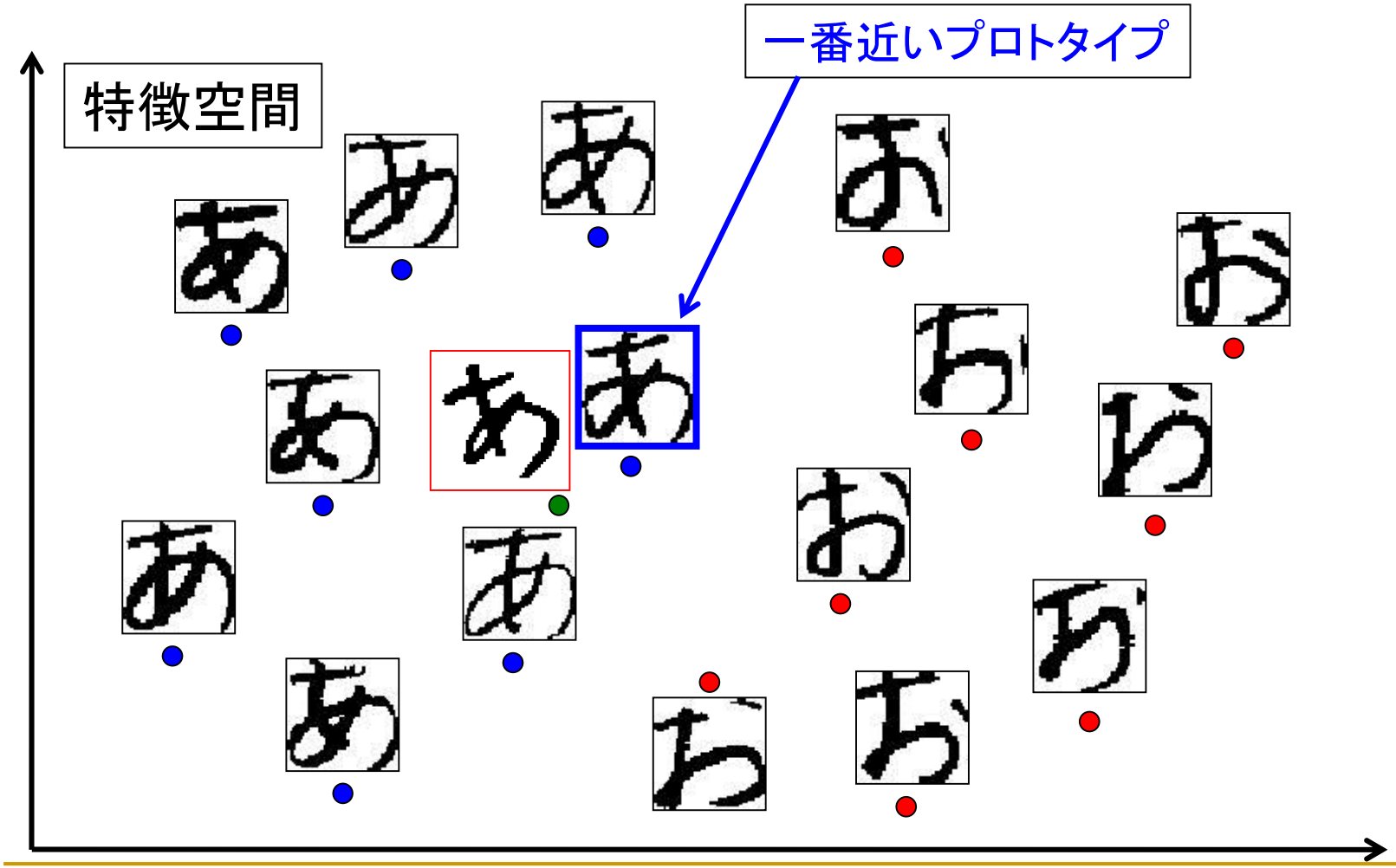


複数個のプロトタイプを利用した最近傍法②

入力パターン「あ」を特徴空間上に配置

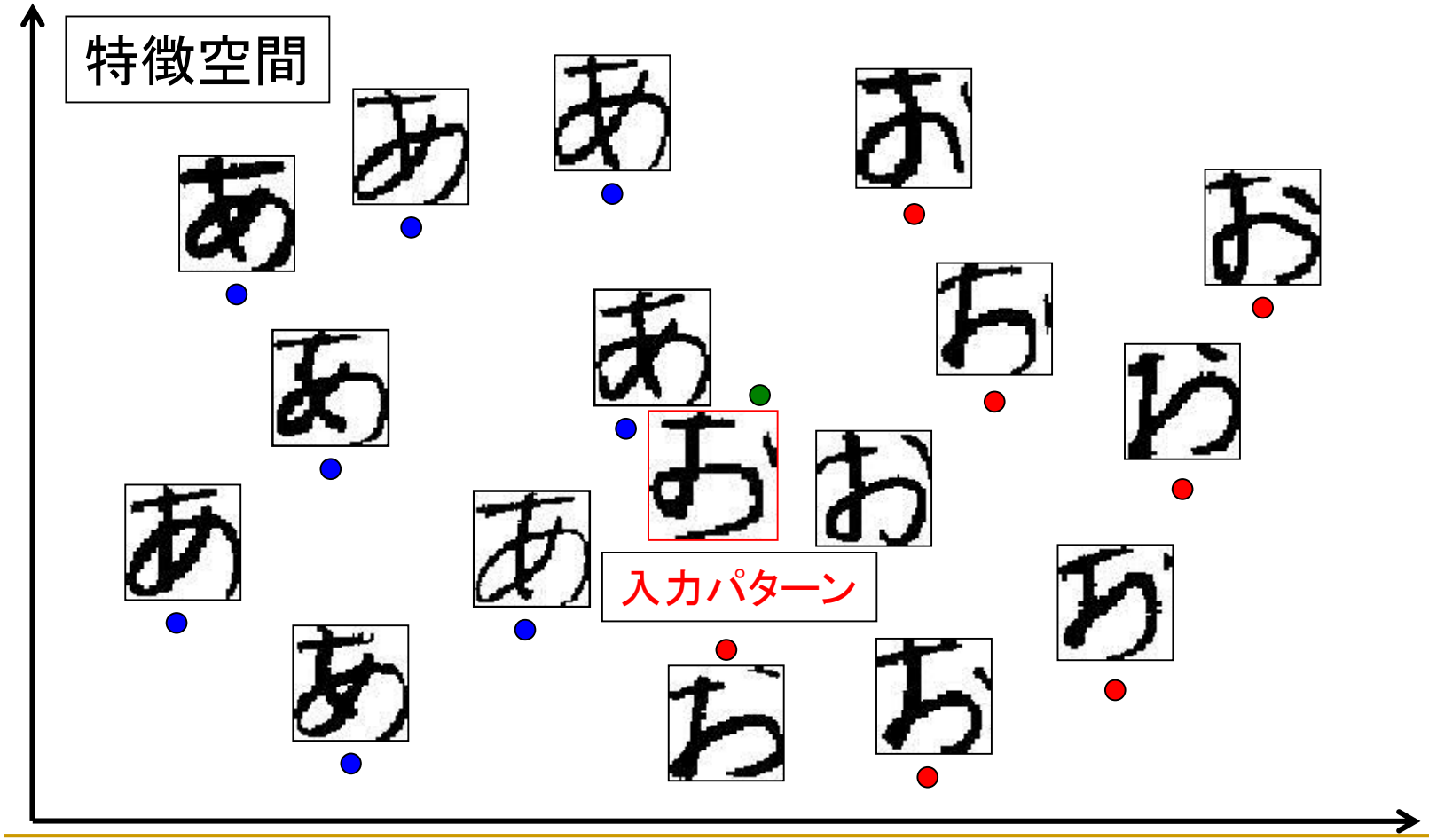


複数個のプロトタイプを利用した最近傍法③

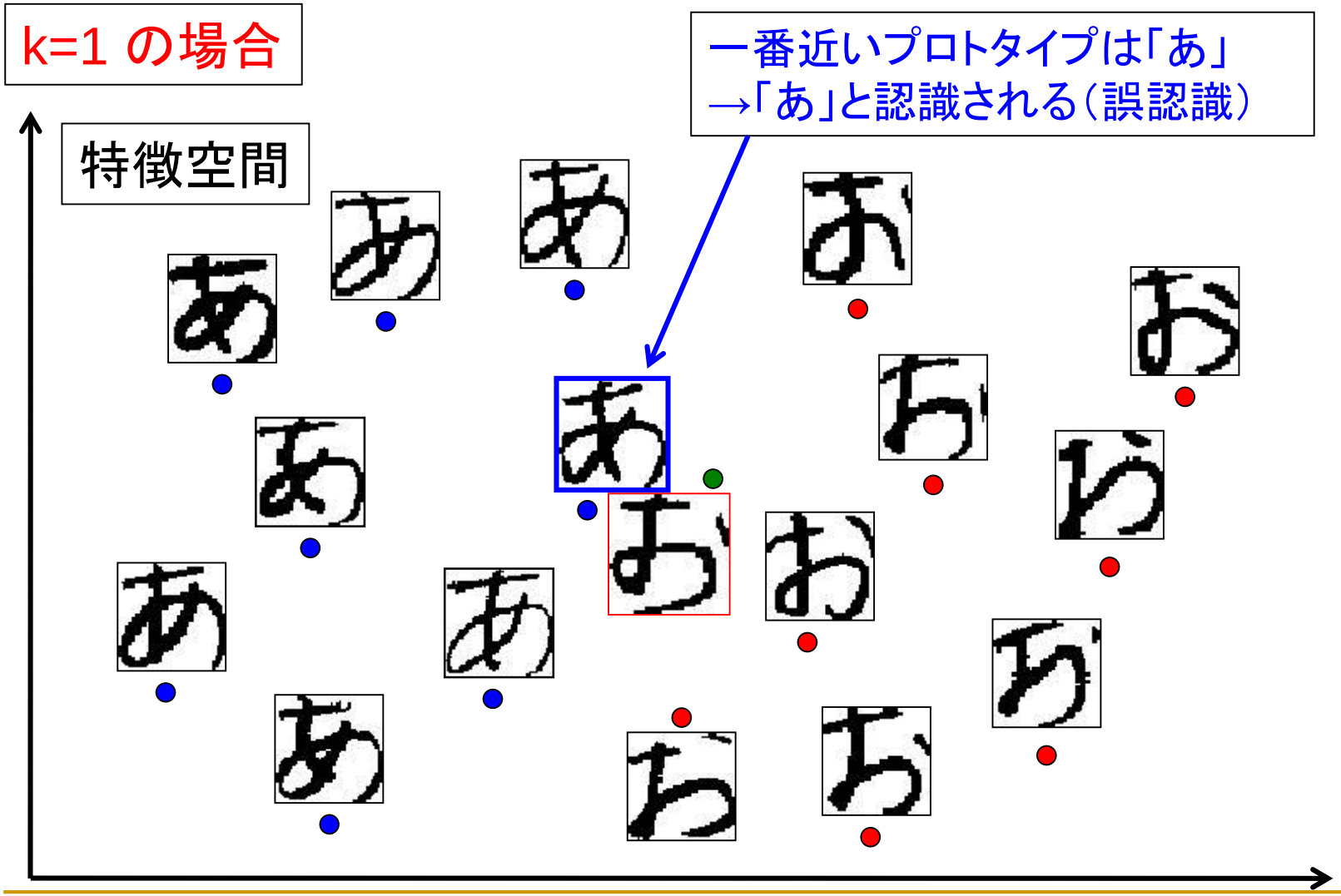


複数個のプロトタイプを利用した最近傍法④

入力パターン「お」を特徴空間上に配置

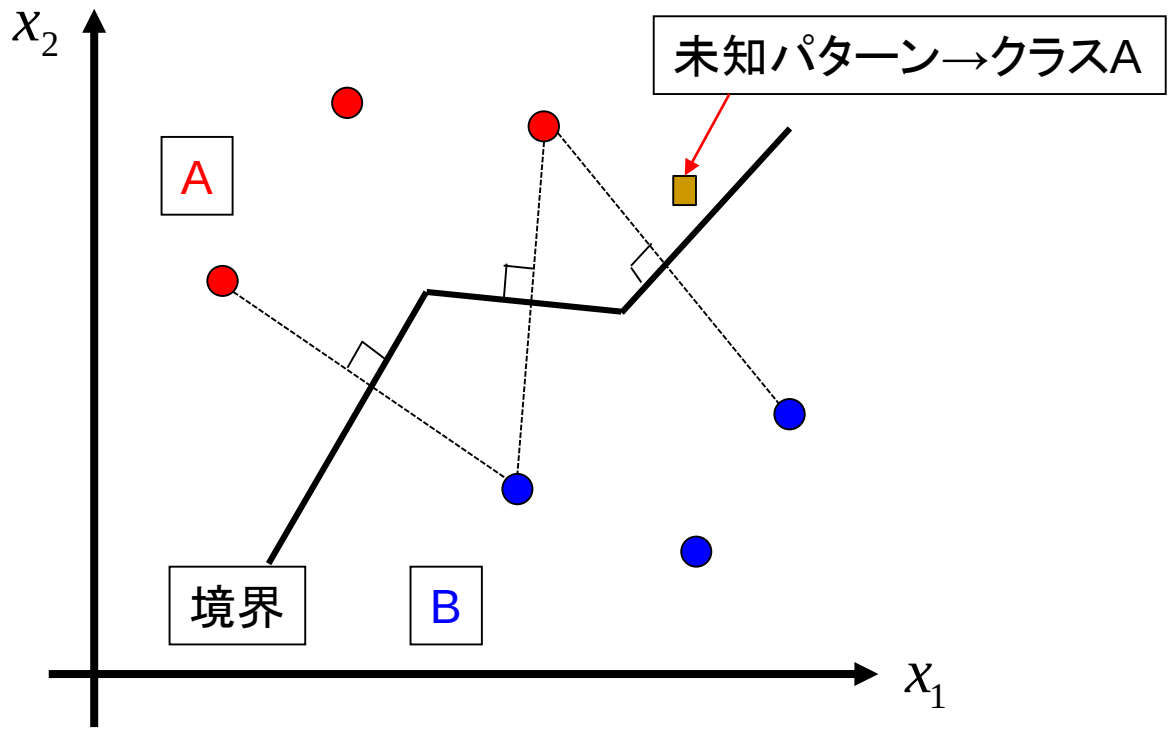


複数個のプロトタイプを利用した最近傍法⑤



ボロノイ図

- 複数個のプロトタイプを利用
  →境界が複雑に



複数個のプロトタイプを利用した最近傍法

- 利点
  - プロトタイプ数が増加するに従って，精度が向上
  - ただし，徐々に精度が一定になっていく傾向
- 問題点
  - プロトタイプ数が増加するに従って，計算時間も増加

k近傍法

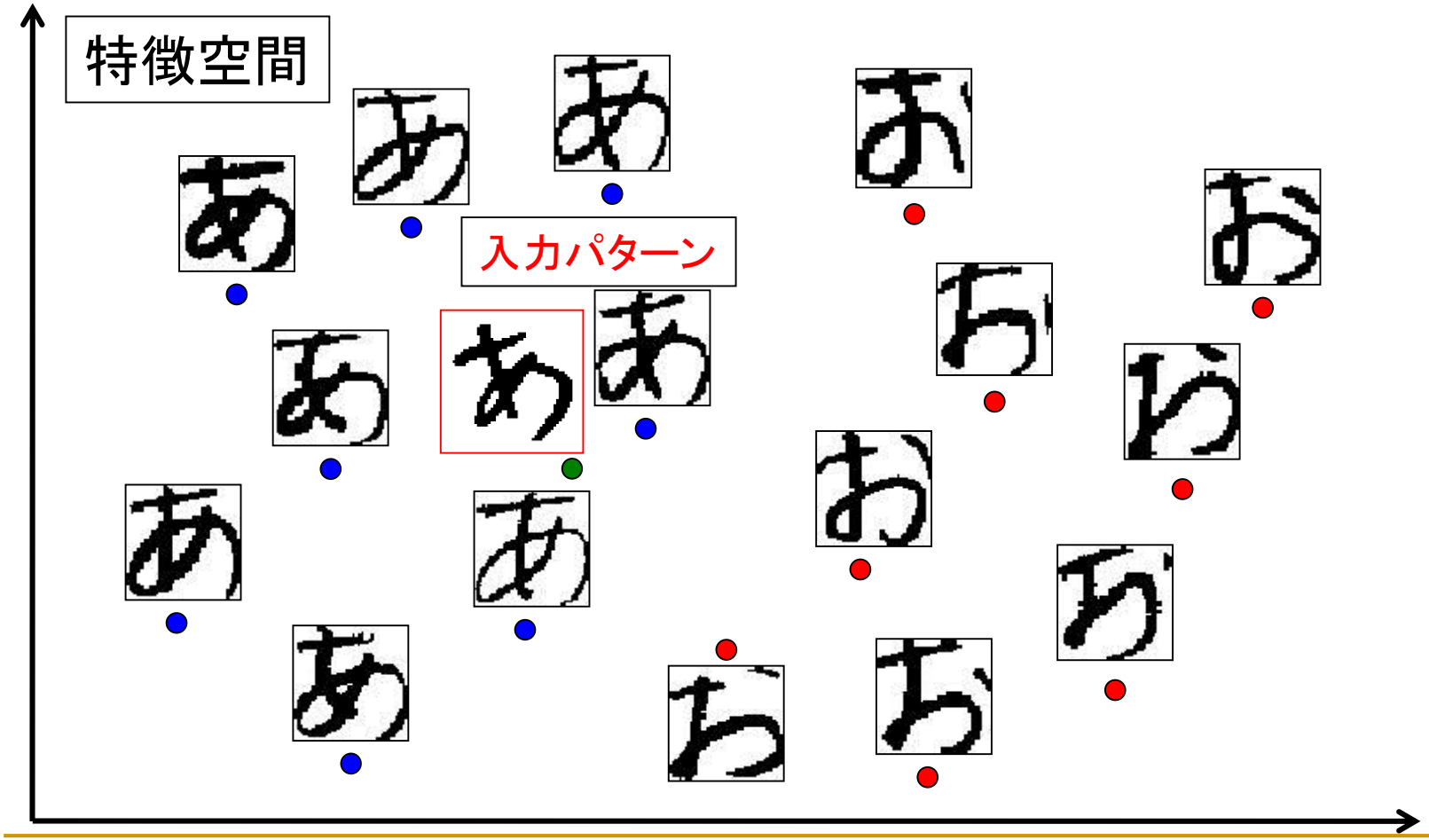
精度の評価方法

最近傍法の改良

- 複数個のプロトタイプを利用
  - 最近傍法を改良（k近傍法）→ 第k候補までの結果を利用して多数決
  - 出現確率（分布）を考慮 → 統計的パターン認識

最近傍法の改良（k近傍法）①

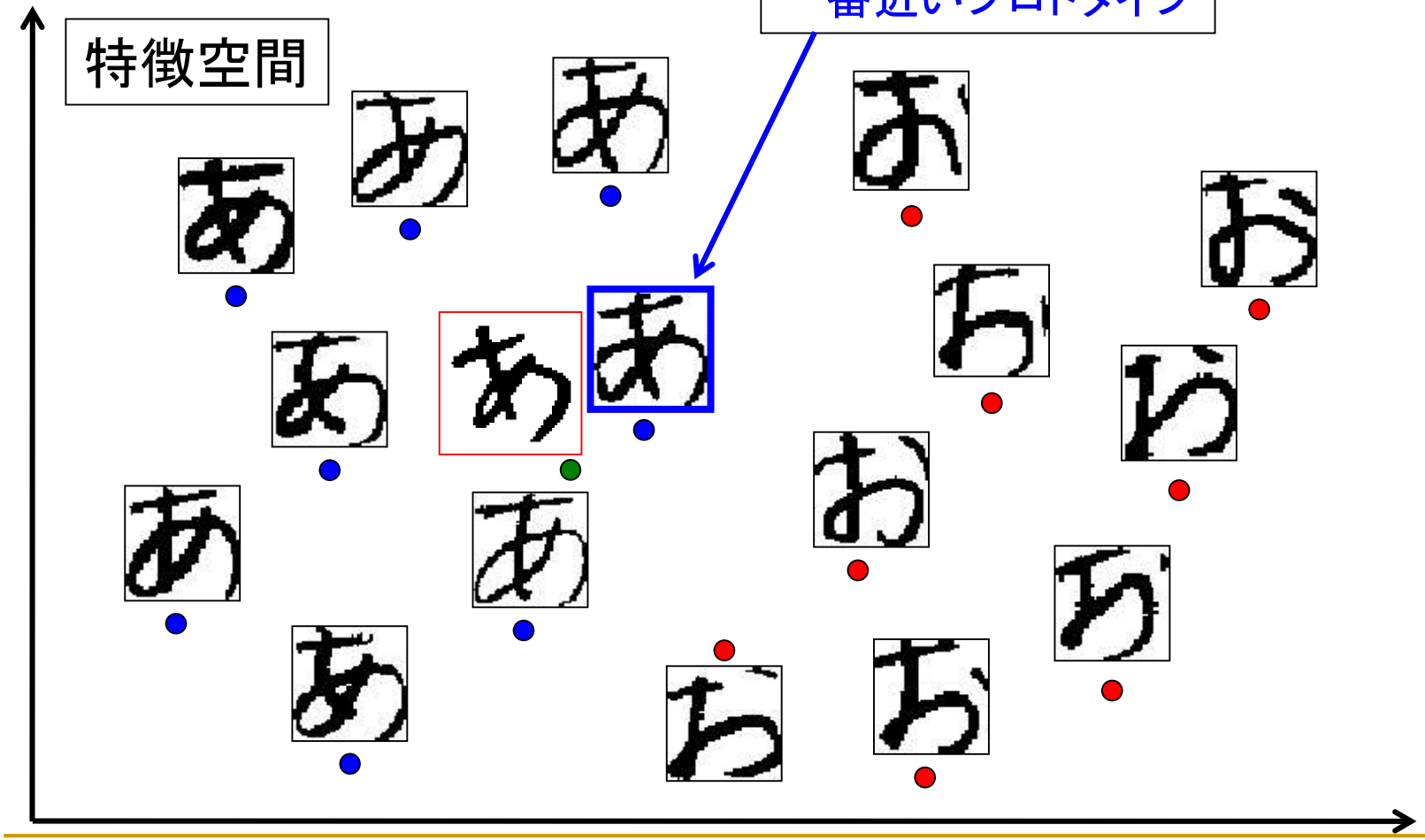
入力パターン「あ」を特徴空間上に配置



k近傍法②

一番目まで近いプロトタイプは「あ」
→「あ」と認識

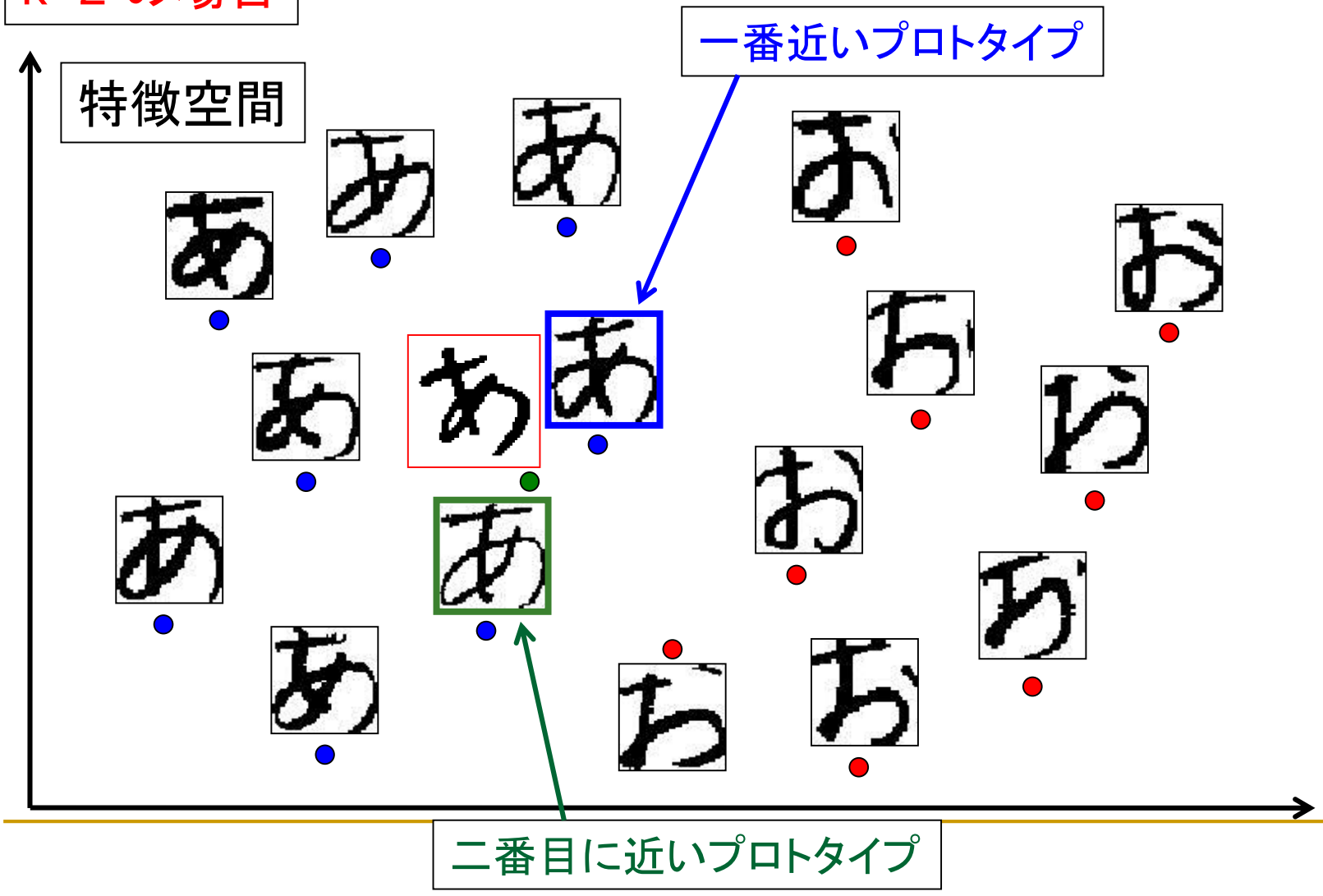
k=1 の場合（最近傍と同じ）



k近傍法③

二番目までに近いプロトタイプは「あ」が2個
→「あ」と認識

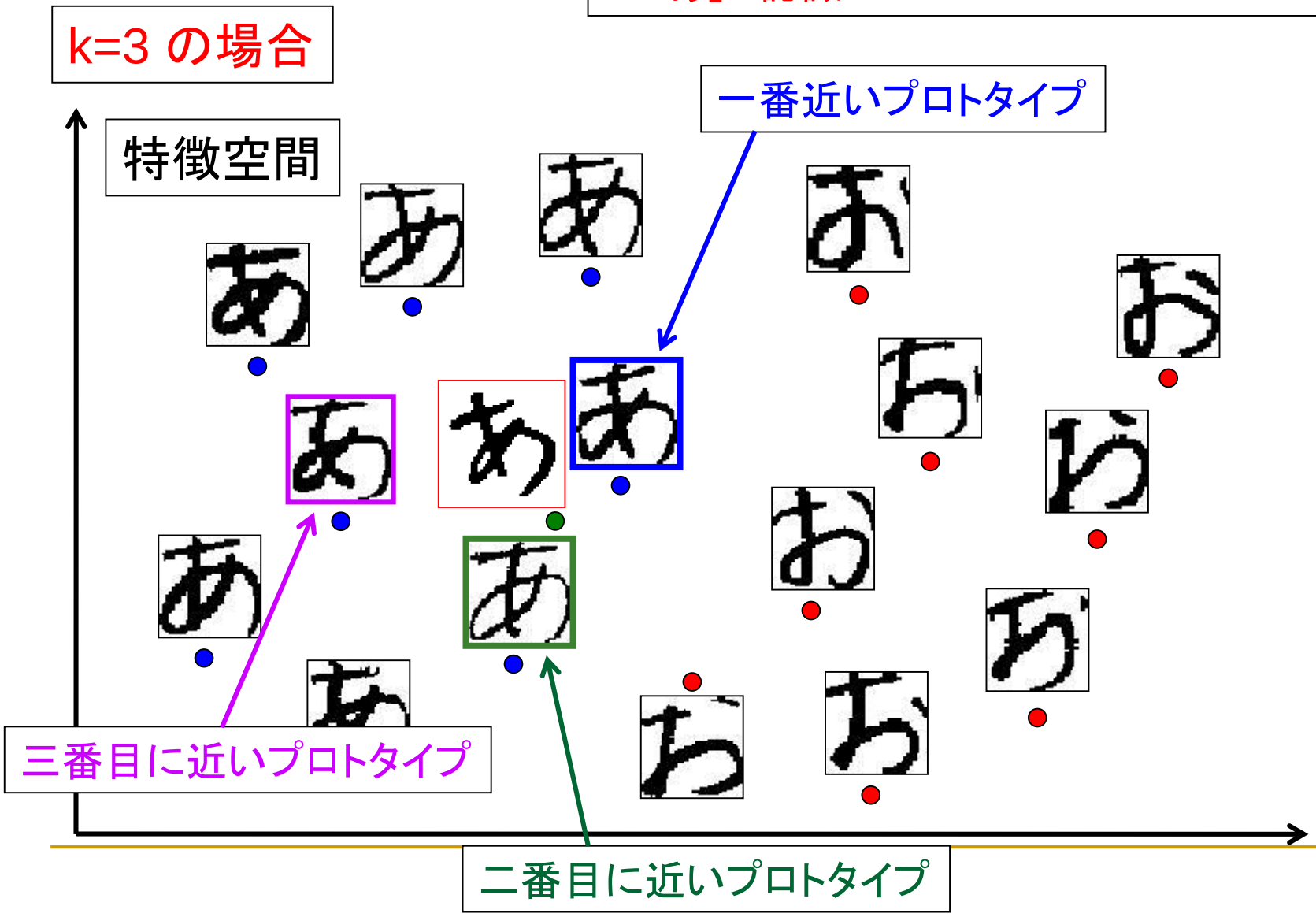
k=2 の場合



k近傍法④

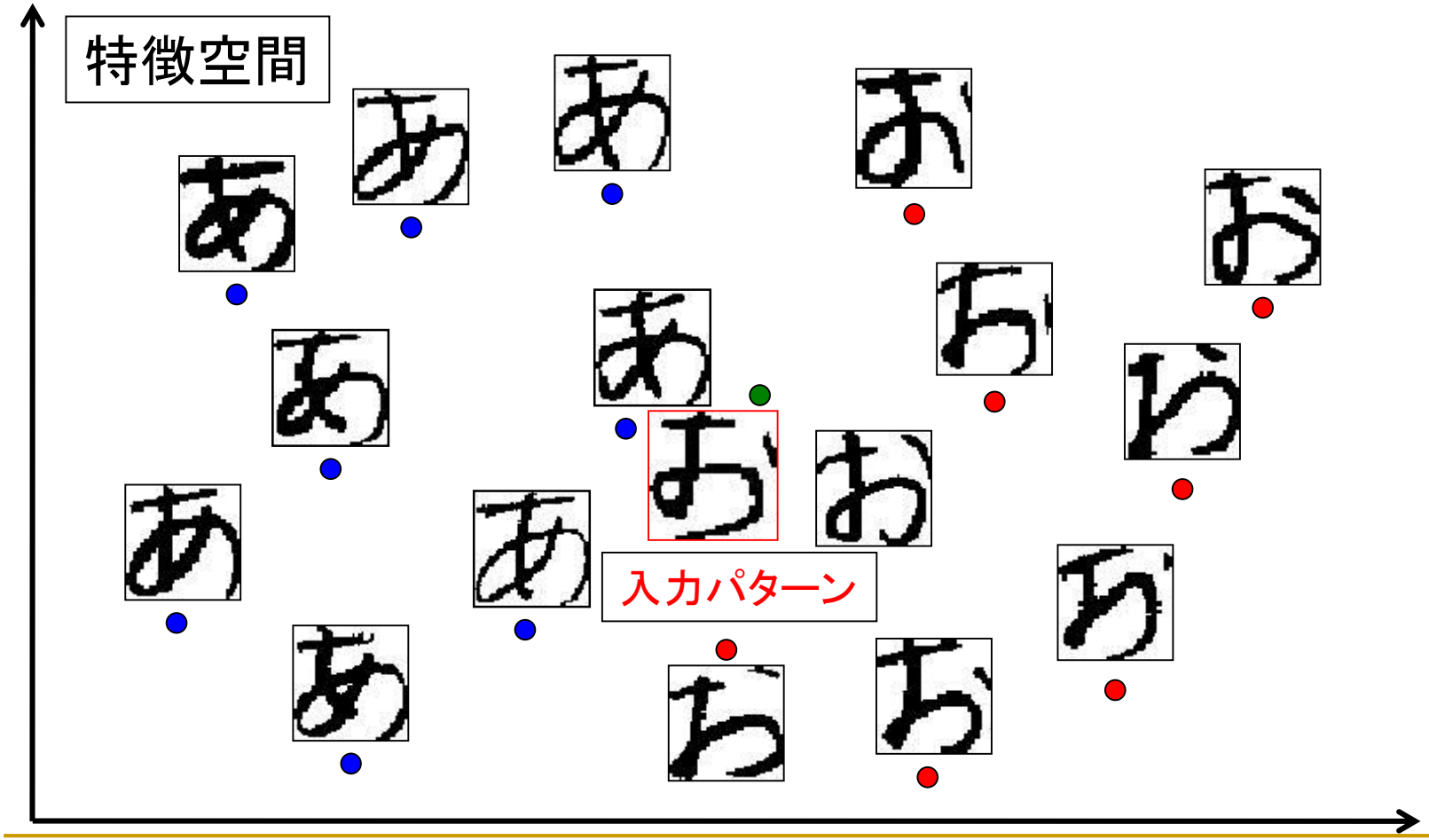
三番目までに近いプロトタイプは「あ」が3個
→「あ」と認識

k=3 の場合

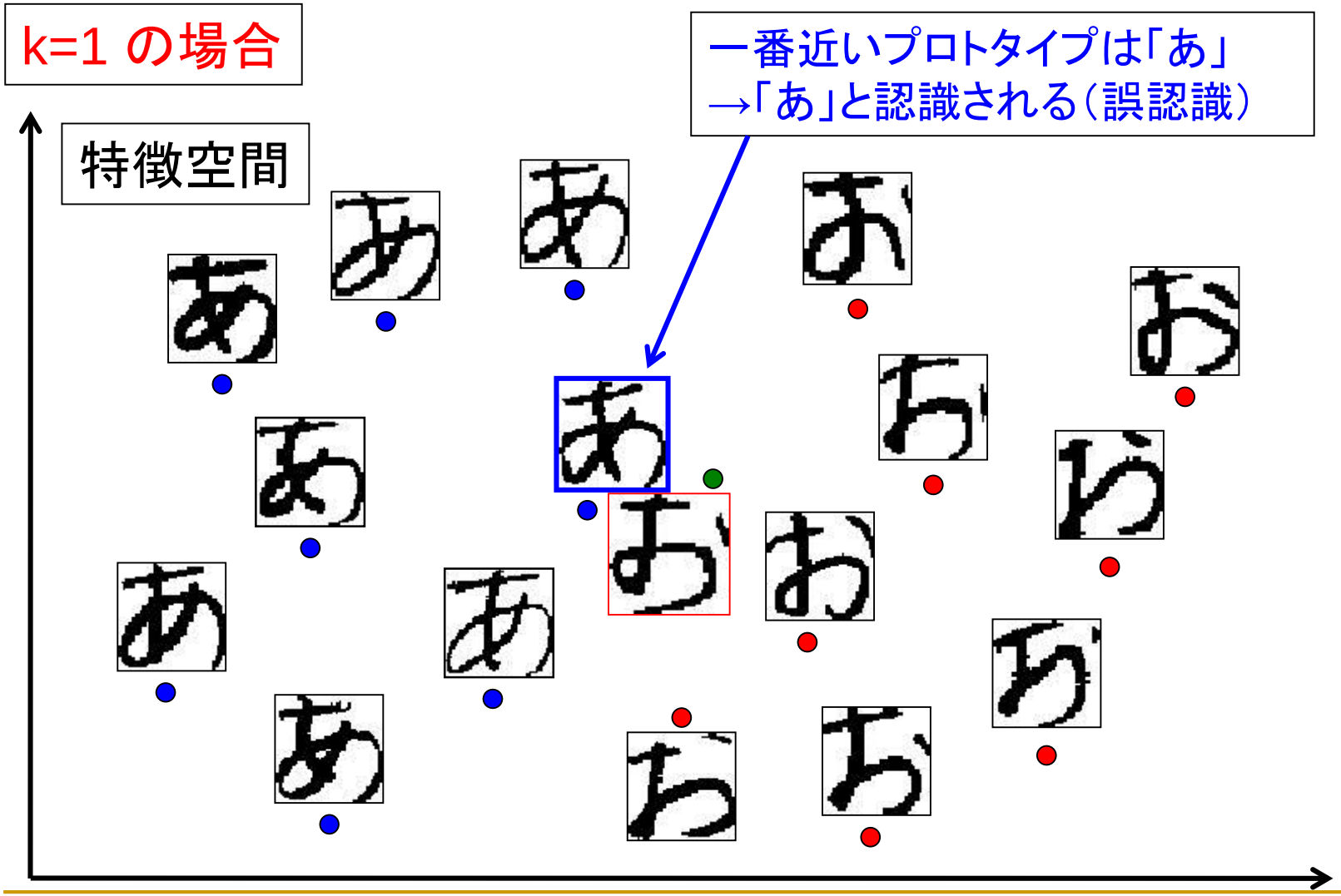


k近傍法⑤

入力パターン「お」を特徴空間上に配置



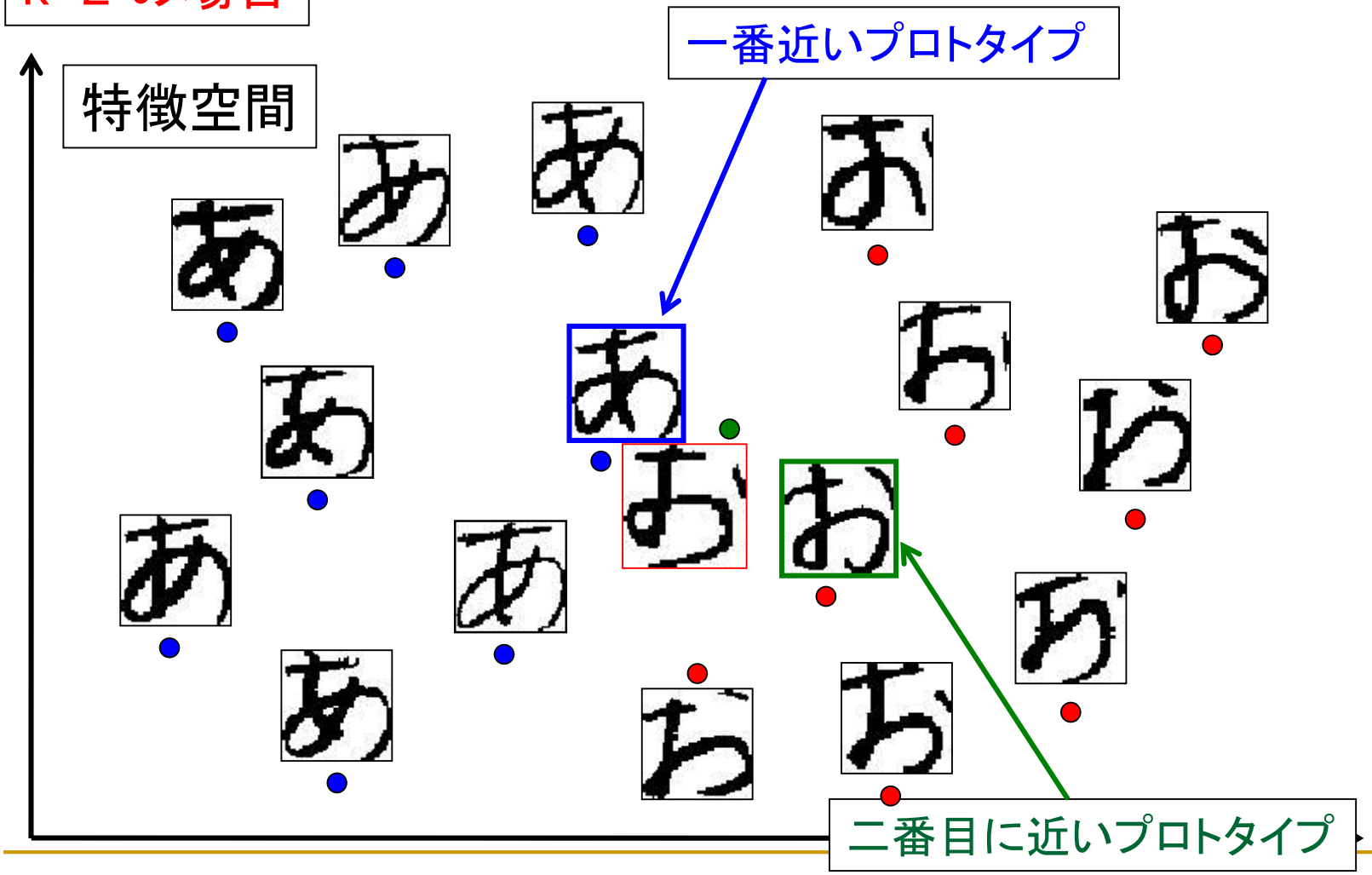
k近傍法⑥



k近傍法⑦

二番目まで近いプロトタイプを調べると「あ」が1個，「お」が1個
→確定できない

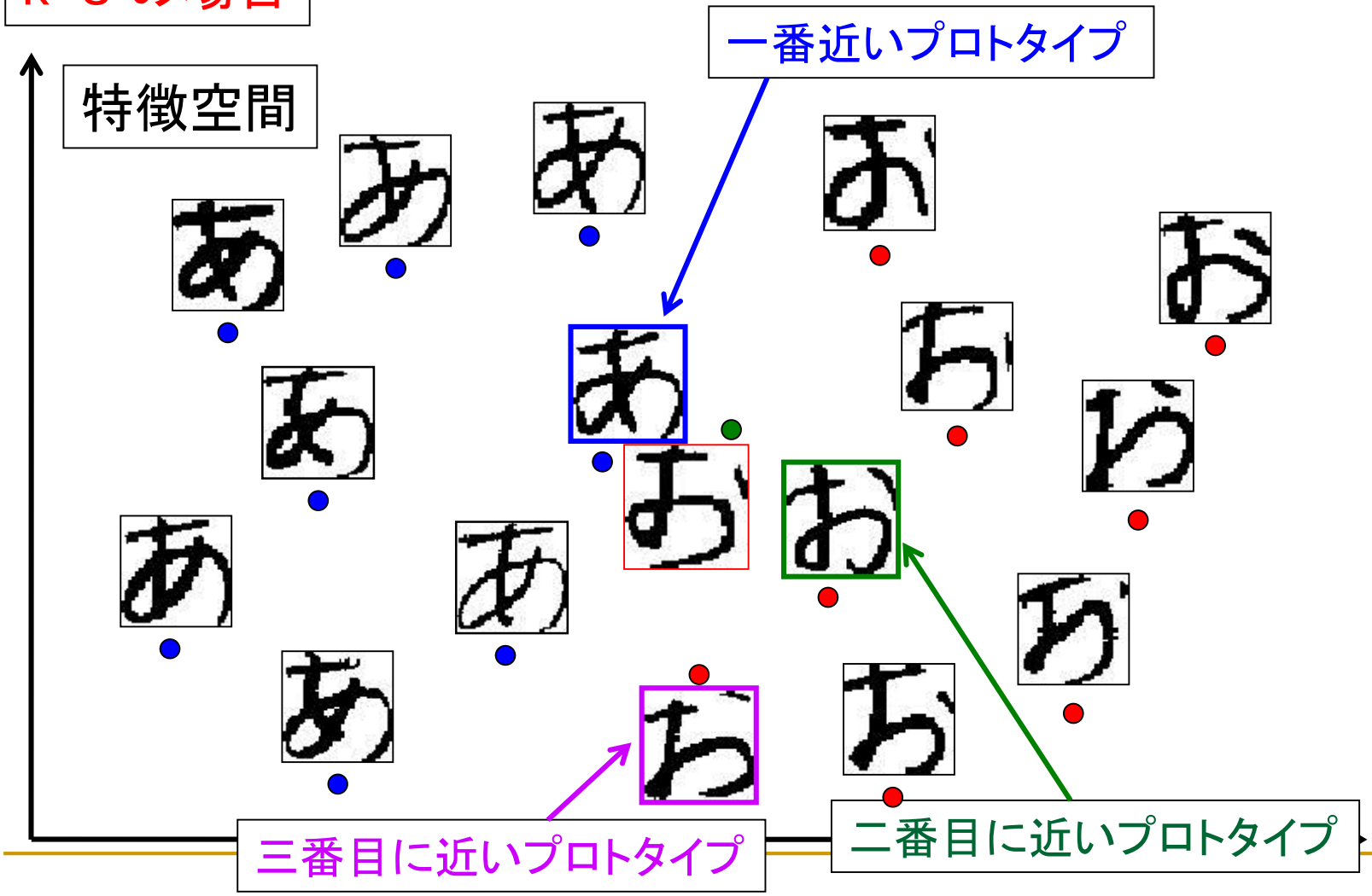
k=2 の場合



k近傍法⑧

三番目まで近いプロトタイプを調べると「あ」が1個，「お」が2個
→「お」と認識

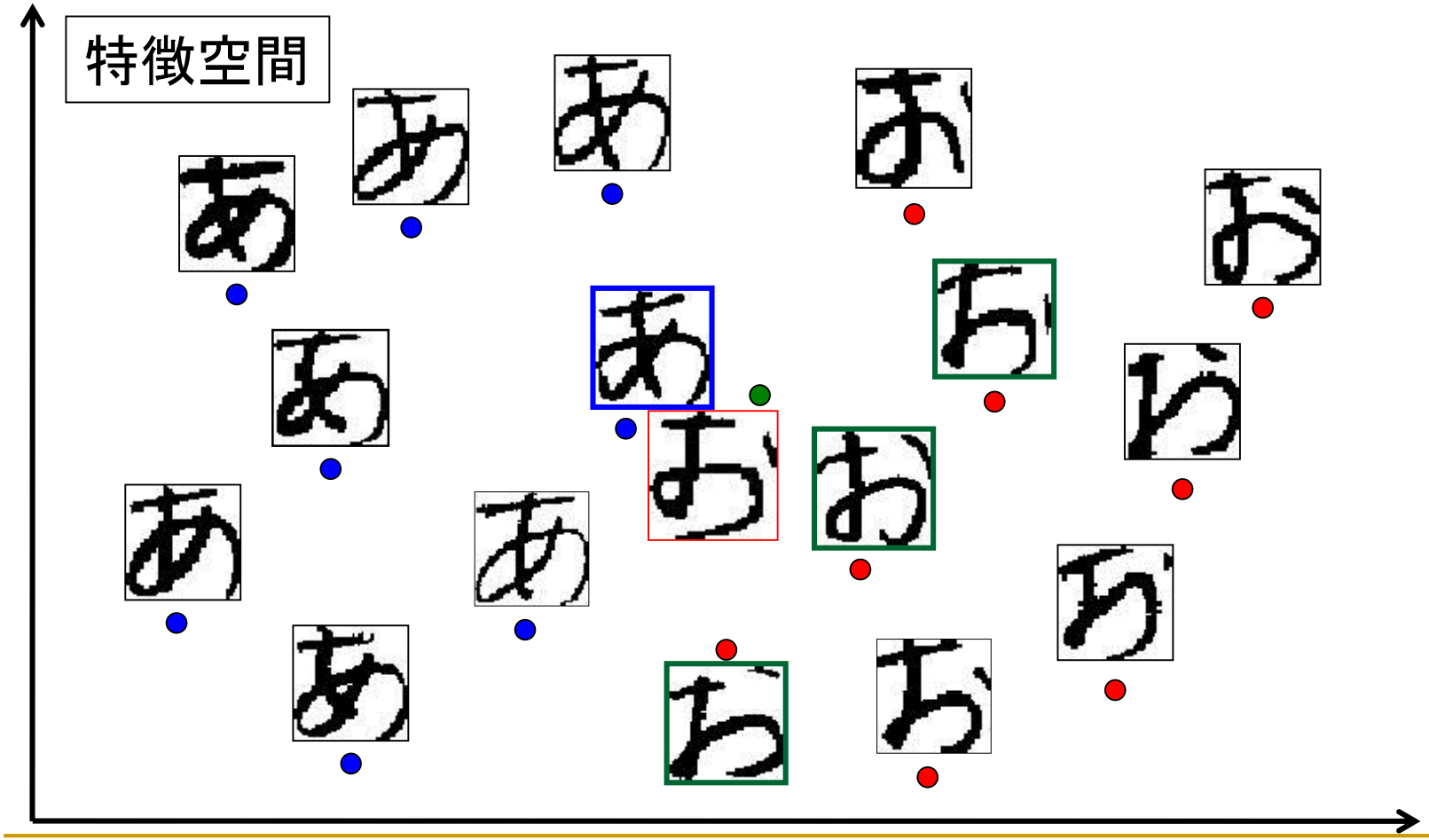
k=3 の場合



k近傍法⑨

k=4 の場合

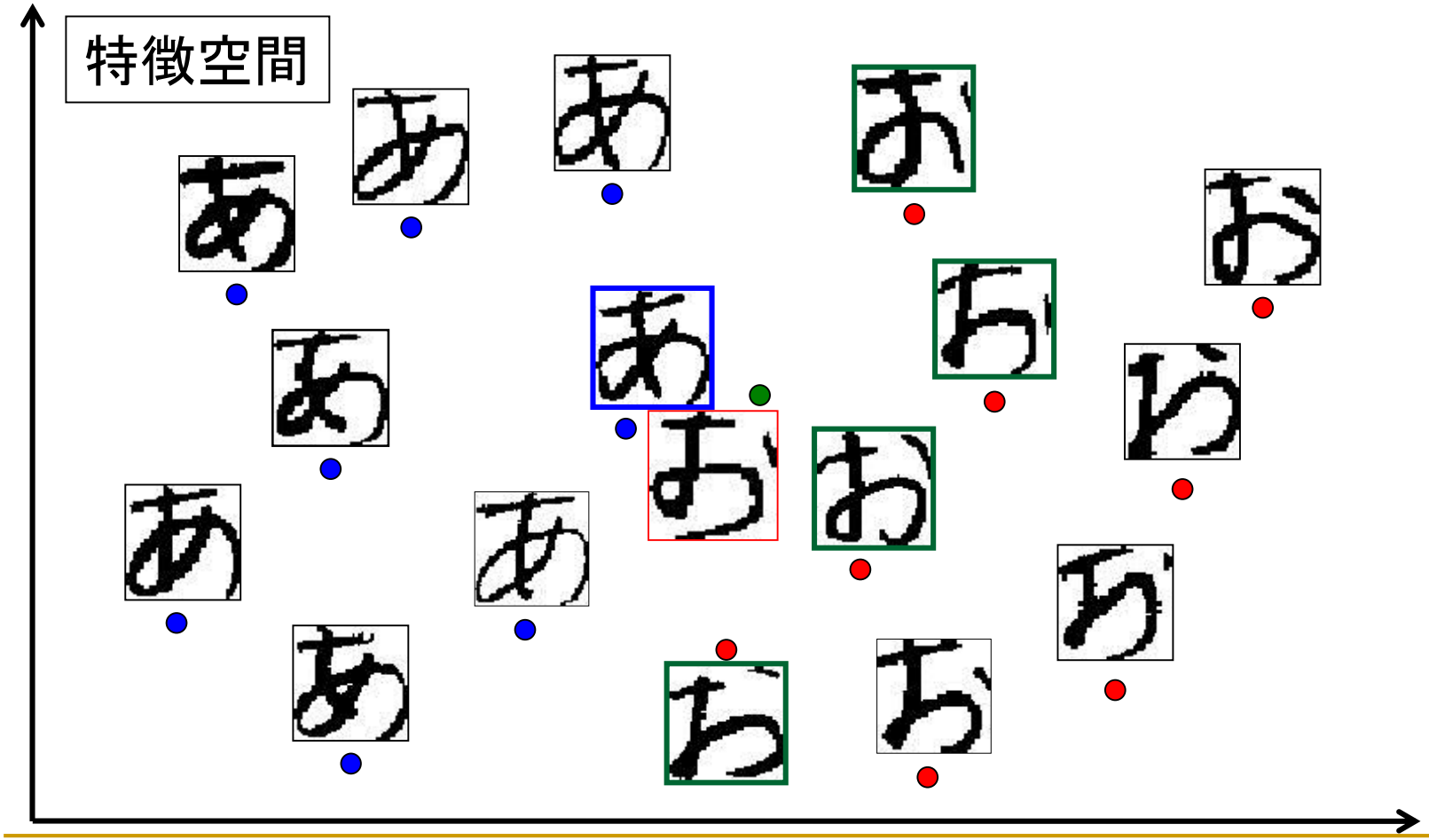
四番目まで近いプロトタイプを調べると「あ」が1個，「お」が3個
→「お」と認識



k近傍法⑩

k=5 の場合

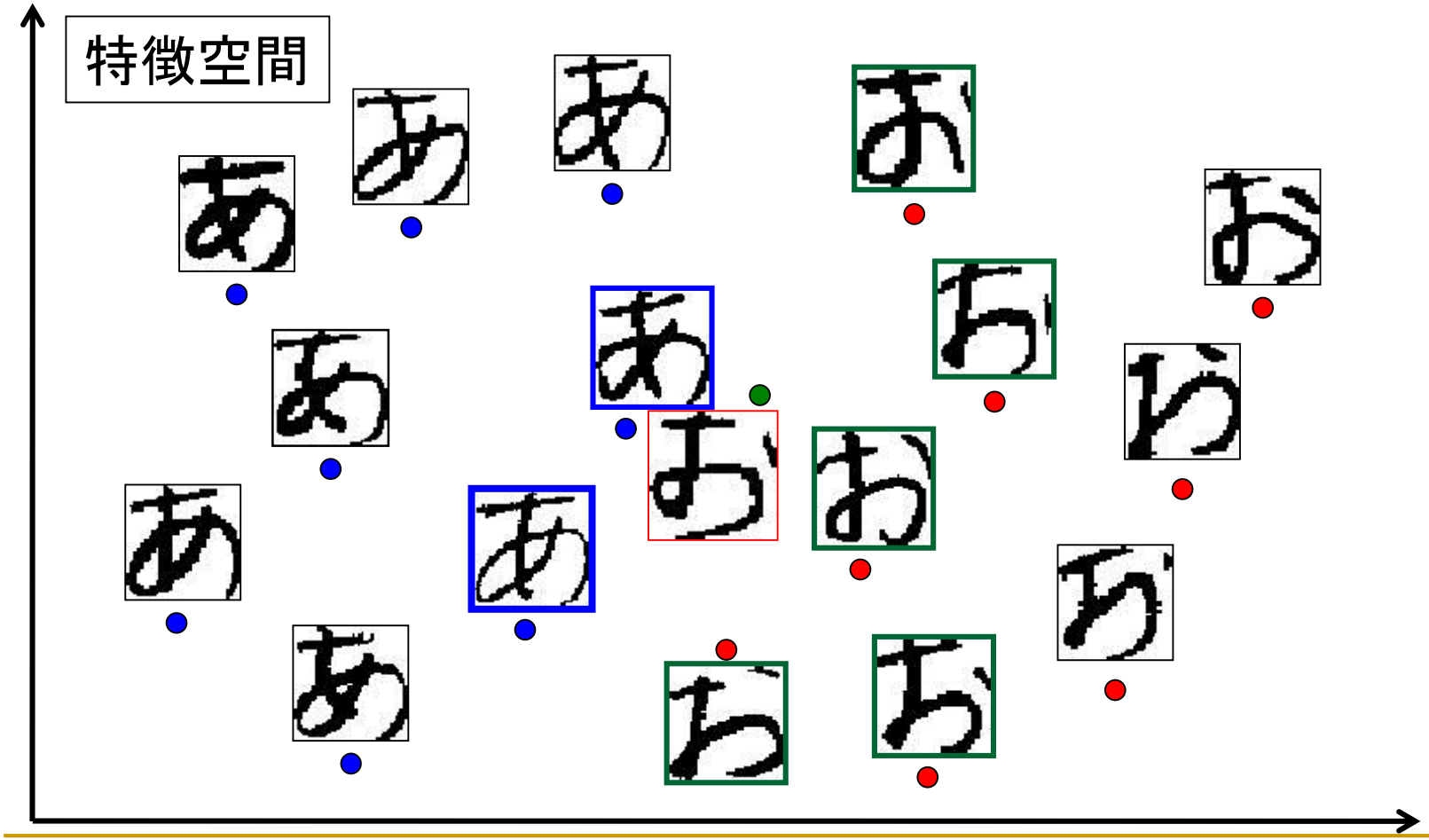
五番目まで近いプロトタイプを調べると「あ」が1個，「お」が4個
→「お」と認識



k近傍法⑪

k=7 の場合

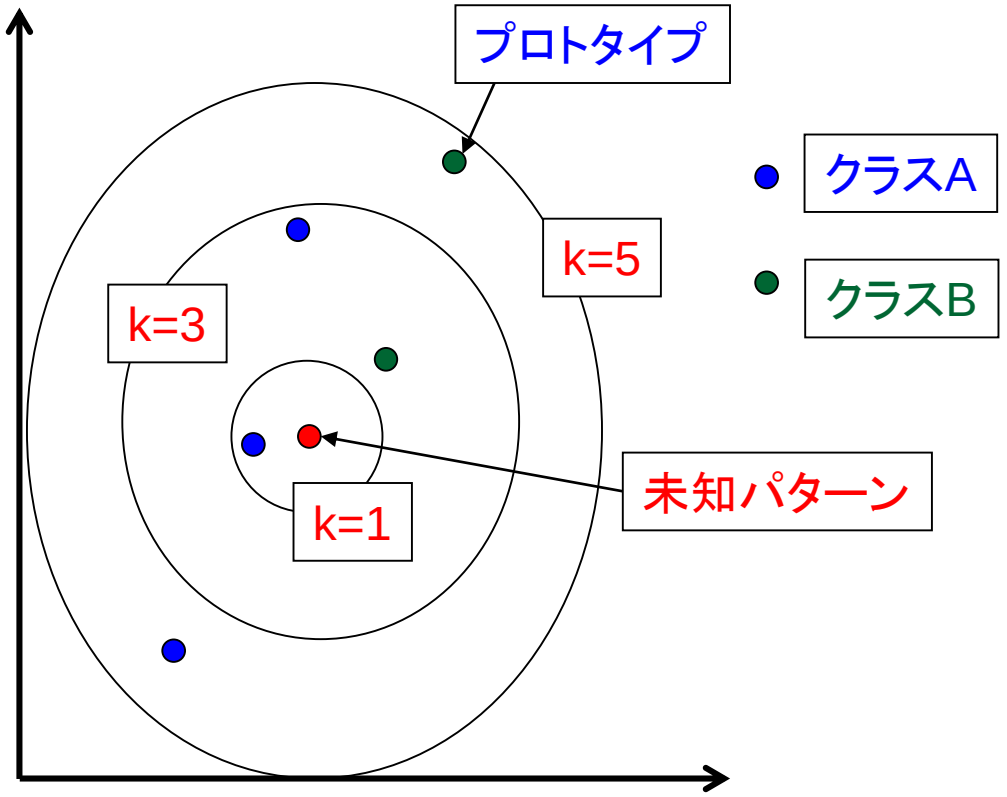
七番目まで近いプロトタイプを調べると「あ」が2個，「お」が5個
→「お」と認識



k近傍法①

- 一つのパターンについて複数個のプロトタイプを用意
- k番目までに近いプロトタイプを調べる
- k個の候補の多数決によって最終的な認識結果を決定
- k=1の場合は，最近傍法と同等

k近傍法②



k=1の場合は最近傍法

k=1
クラスAが1個→クラスA

k=3
クラスAが2個，クラスBが1個
→クラスA

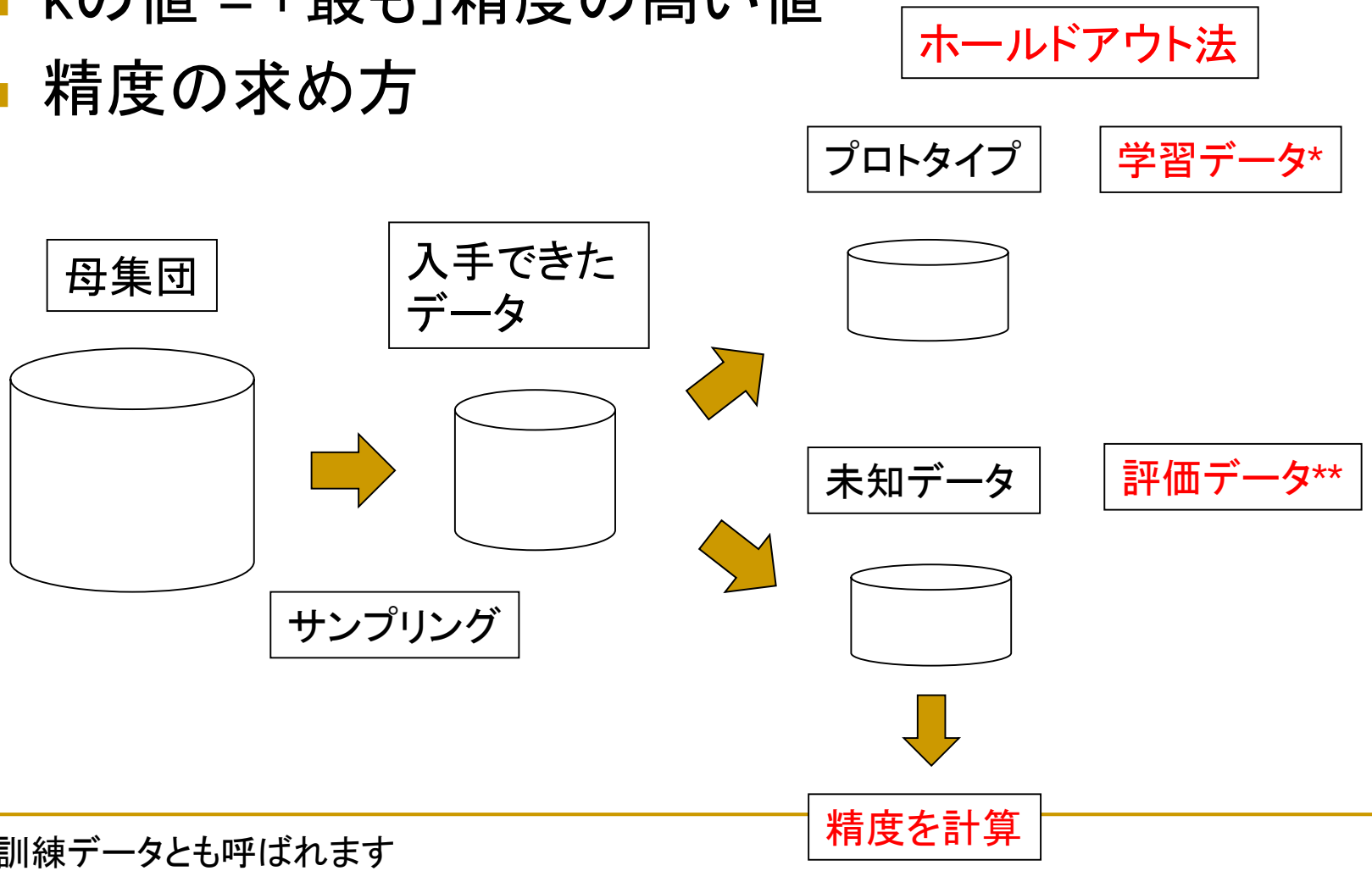
k=5
クラスAが3個，クラスBが2個
→クラスA

k近傍法③

- モデルを用いてクラスを構築する学習を行わない
- 学習データ（事例，インスタンス）を用いて，未知パターンを分類
- インスタンスベース学習（Instance Based Learning）とも呼ばれる

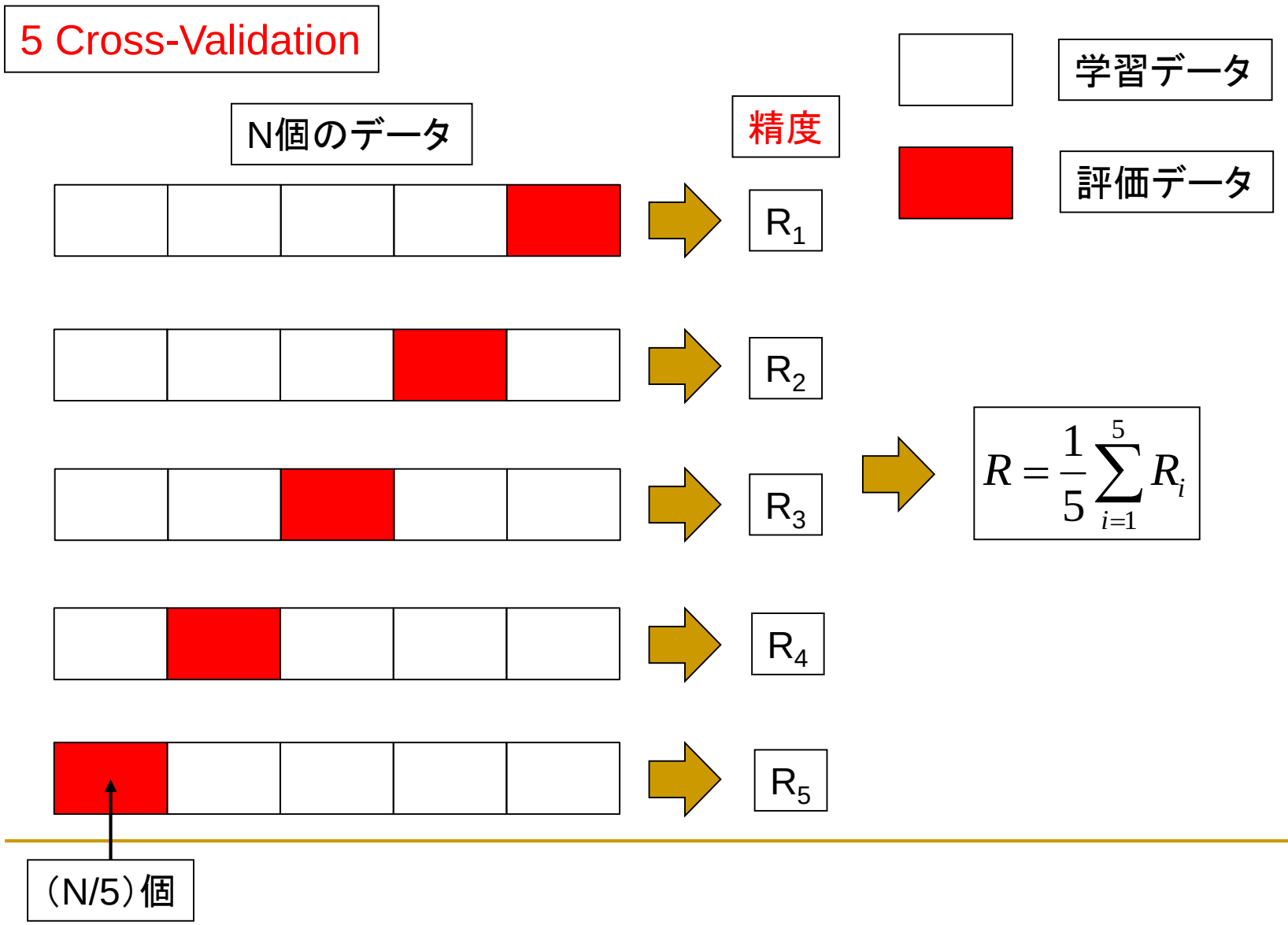
kの値の決め方

- kの値 =「最も」精度の高い値
- 精度の求め方

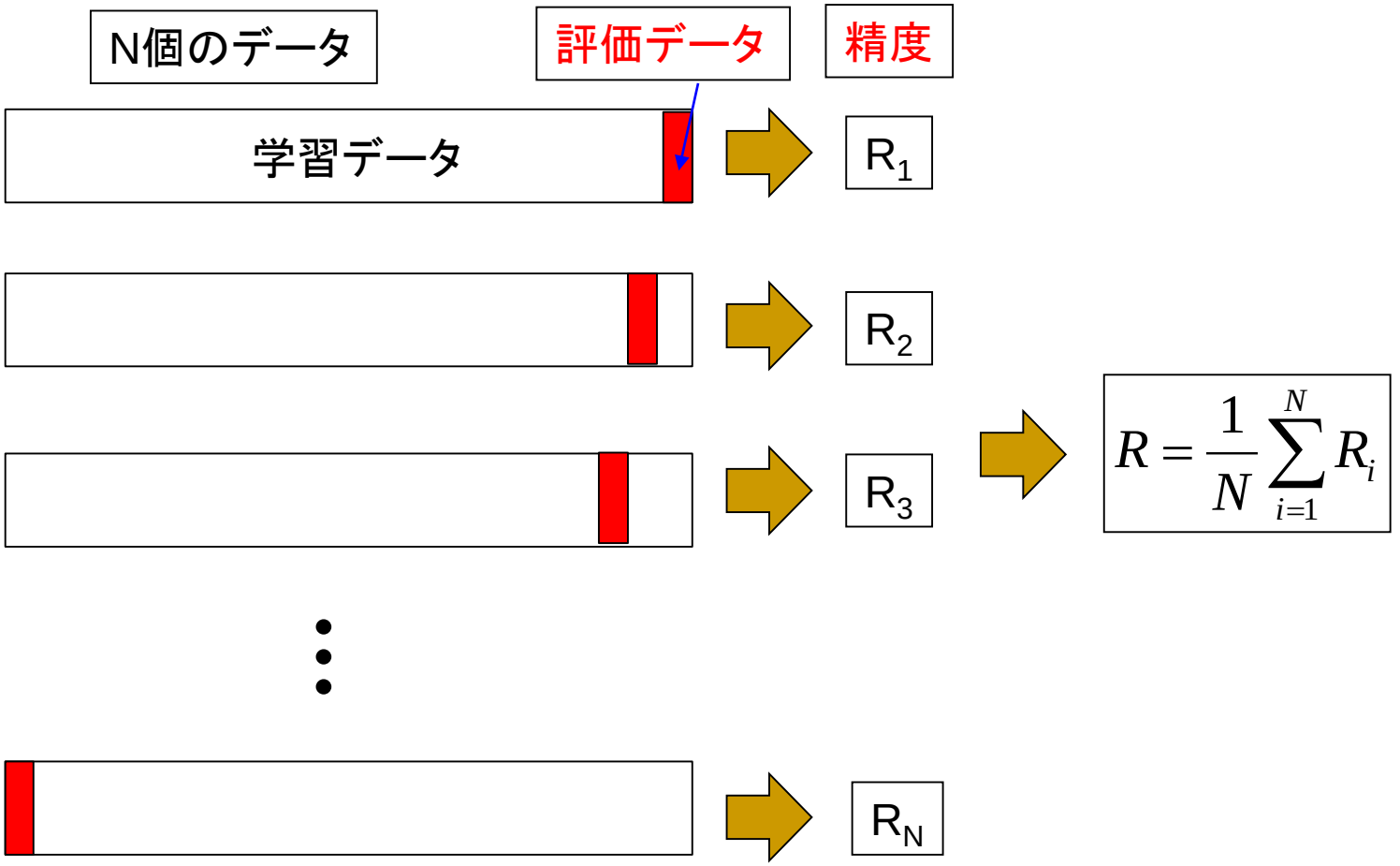


*訓練データとも呼ばれます
**テストデータとも呼ばれます

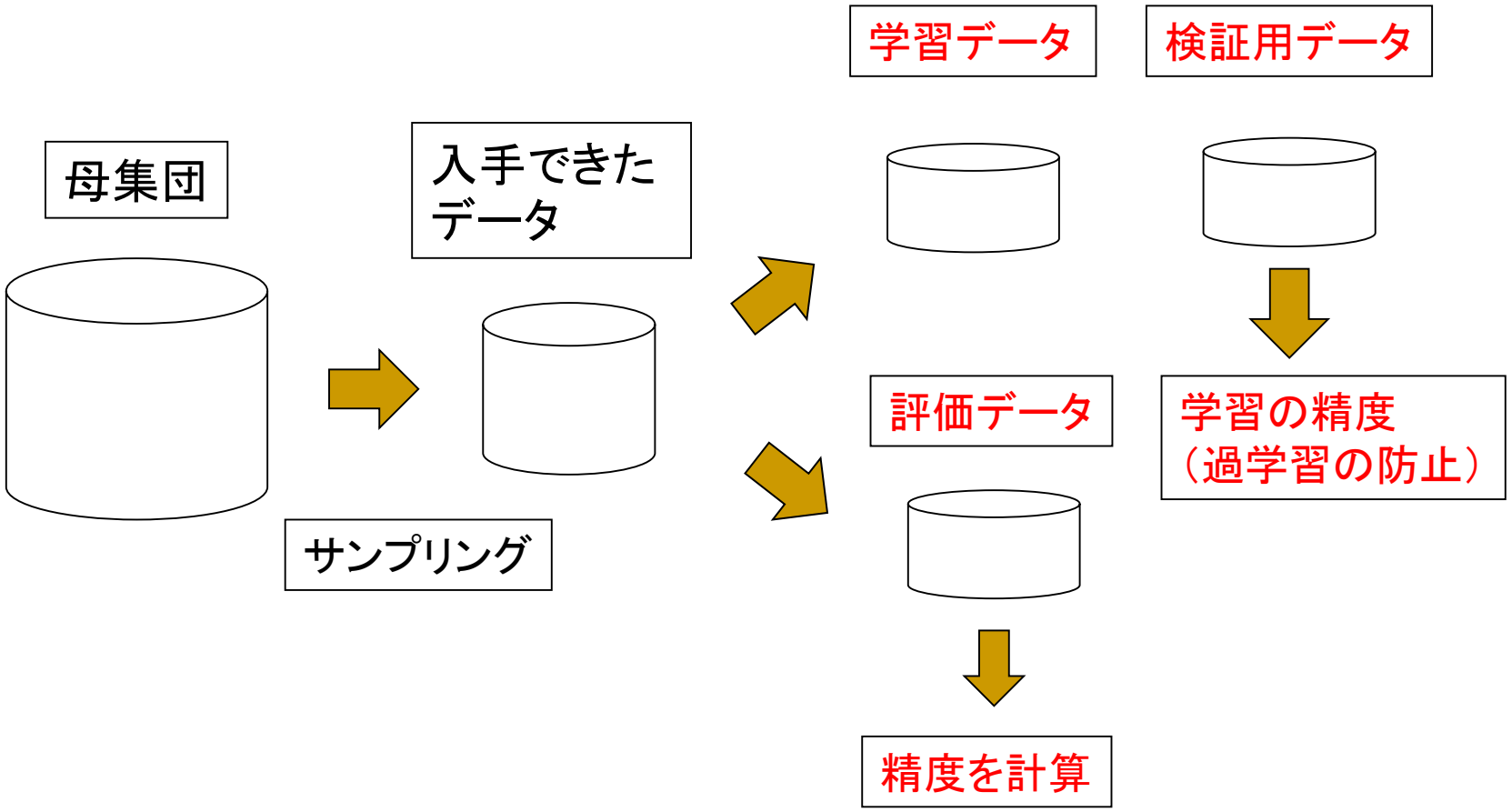
交差検証（Cross Validation）



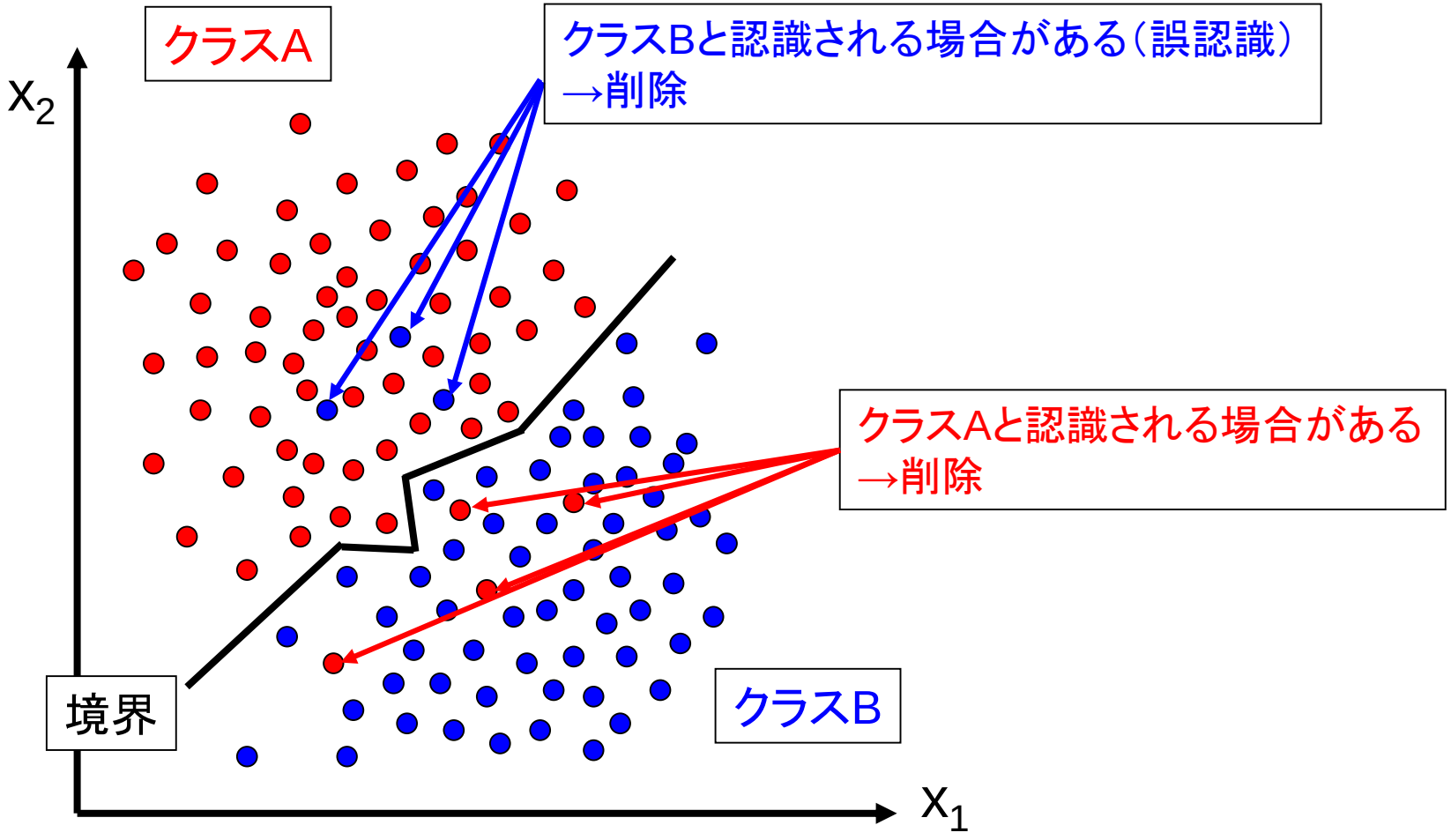
一つ抜き法（Leave One Out）



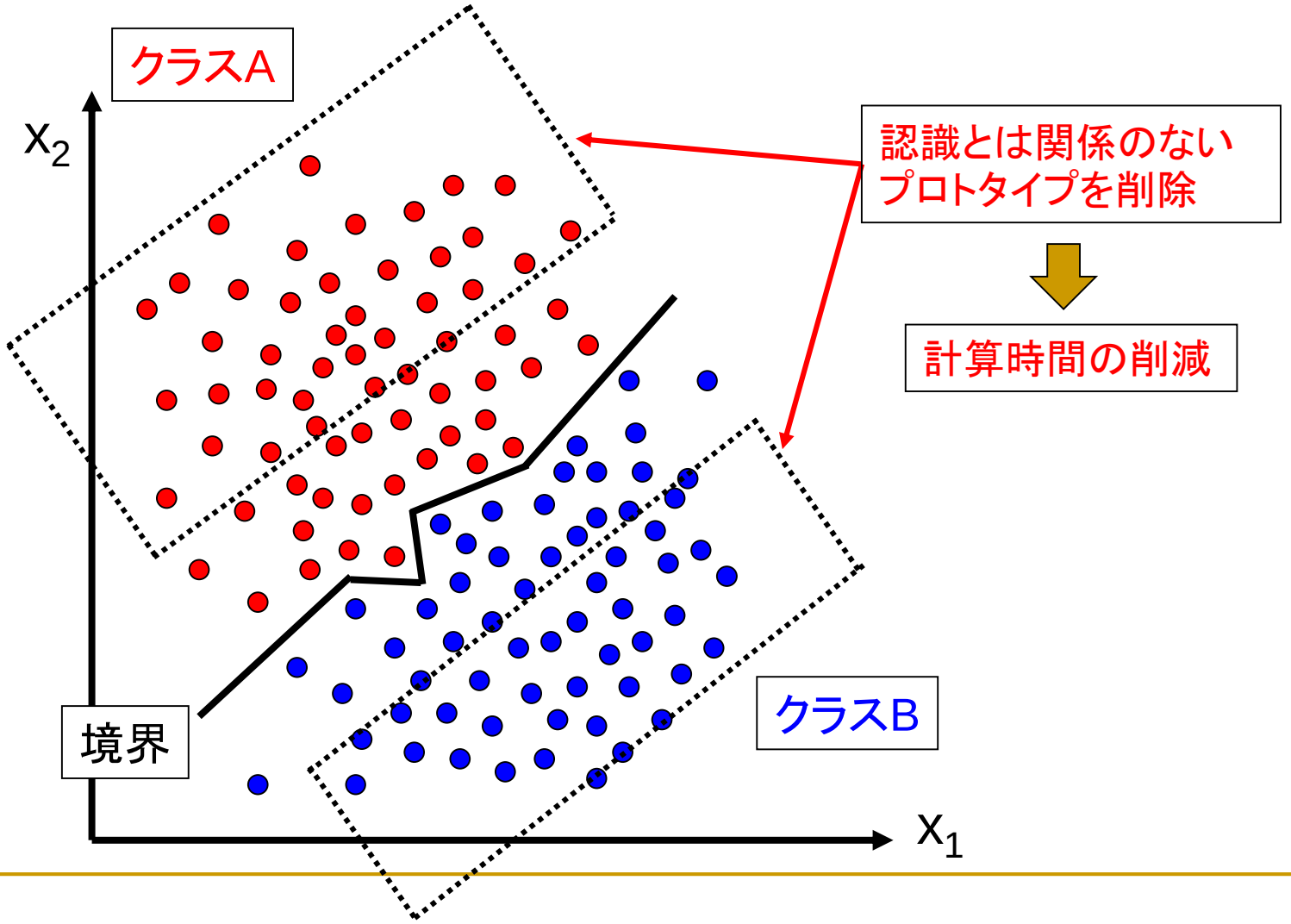
検証用データの利用



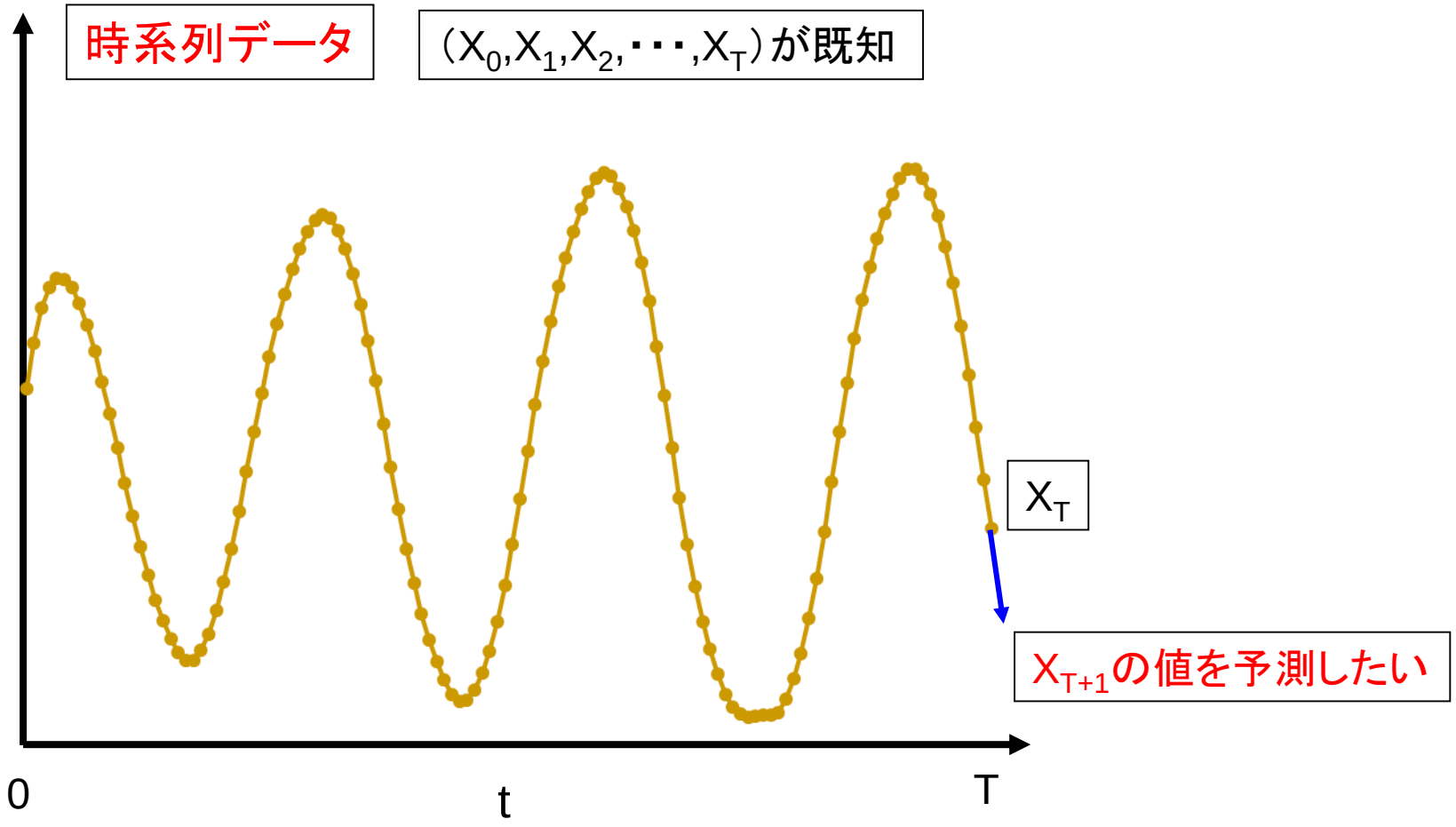
k近傍法の改良①（誤り削除）



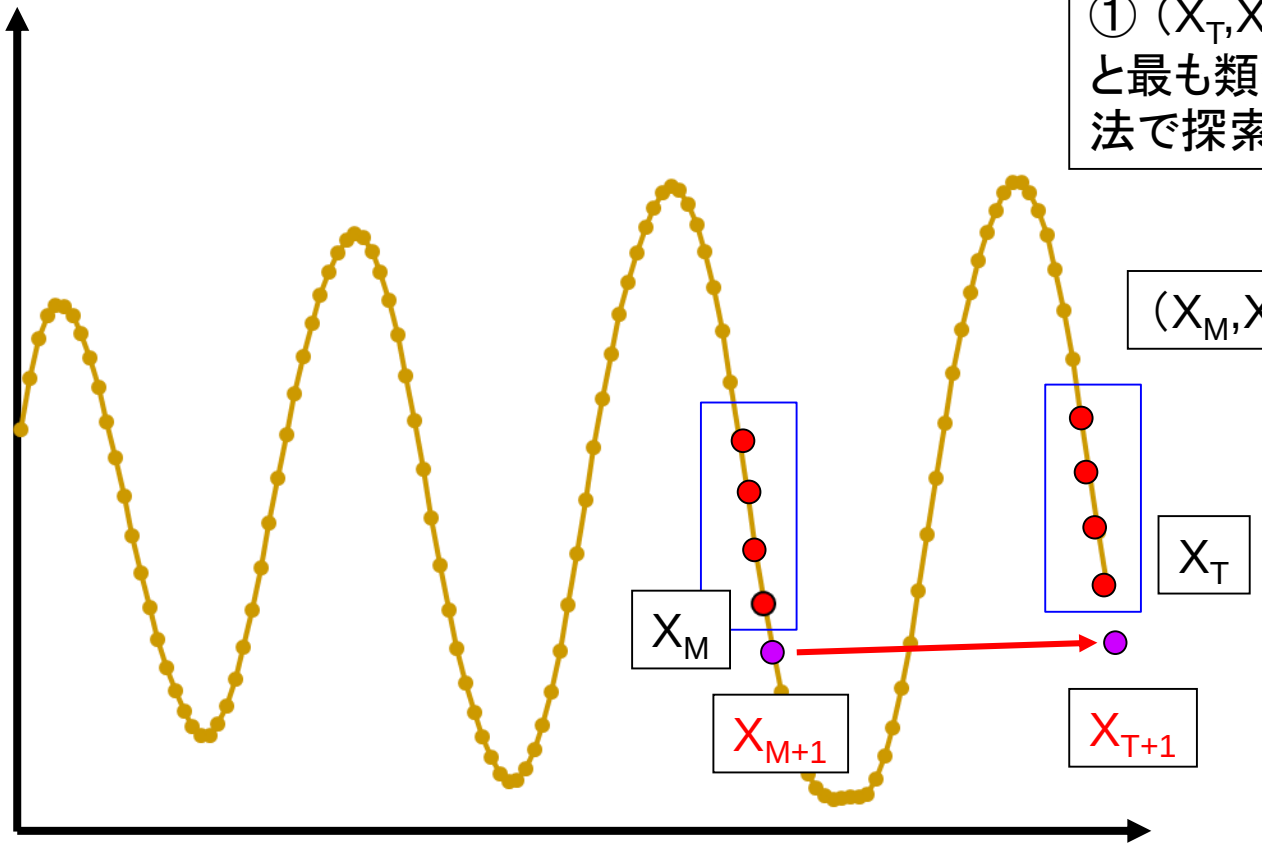
k近傍法の改良②（圧縮）



最近傍法を用いた予測①



最近傍法を用いた予測②



① $(X_T, X_{T-1}, X_{T-2}, \cdots, X_{T-n})$ と最も類似した系列を最近傍法で探索

$(X_M, X_{M-1}, X_{M-2}, \cdots, X_{M-n})$

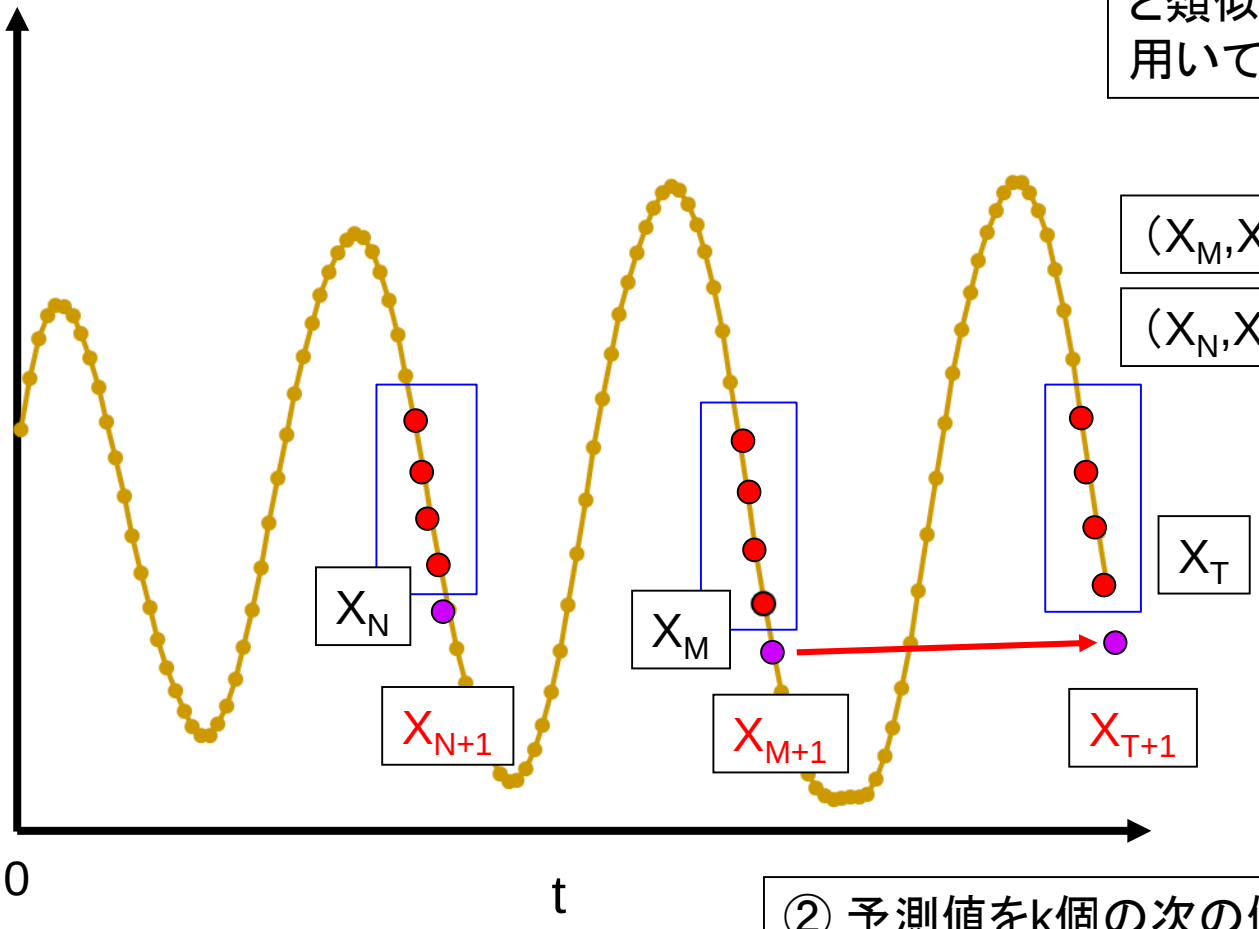
② 予測値をX_Mの次の値(X_{M+1})とする
$X_{T+1}=X_{M+1}$

最近傍法を用いた予測③

① $(X_T, X_{T-1}, X_{T-2}, \cdots, X_{T-n})$ と類似した系列をk近傍法を用いて探索（図はk=2）

$(X_M, X_{M-1}, X_{M-2}, \cdots, X_{M-n})$

$(X_N, X_{N-1}, X_{N-2}, \cdots, X_{N-n})$

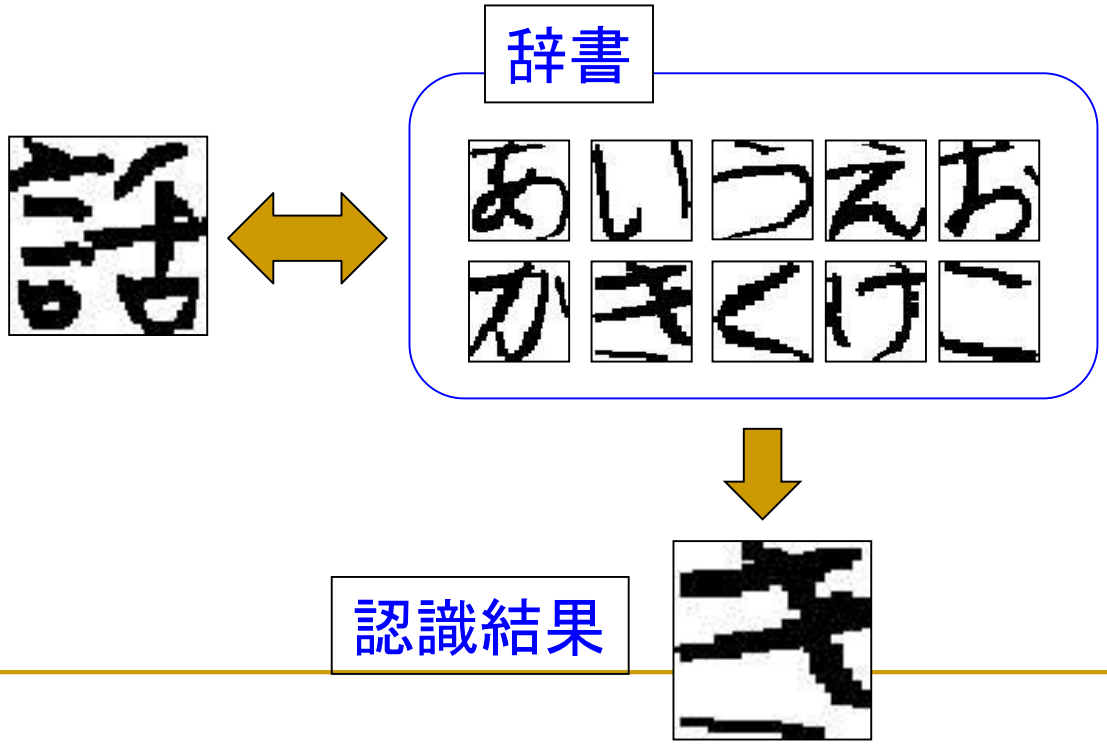


② 予測値をk個の次の値の平均値とする

$X_{T+1}=(X_{M+1}+X_{N+1})/2$

k近傍法（最近傍法）の問題点

- 「辞書」に存在しないパターンは認識できない
  - 「辞書」に存在しないパターンは，最も類似したプロトタイプとして認識

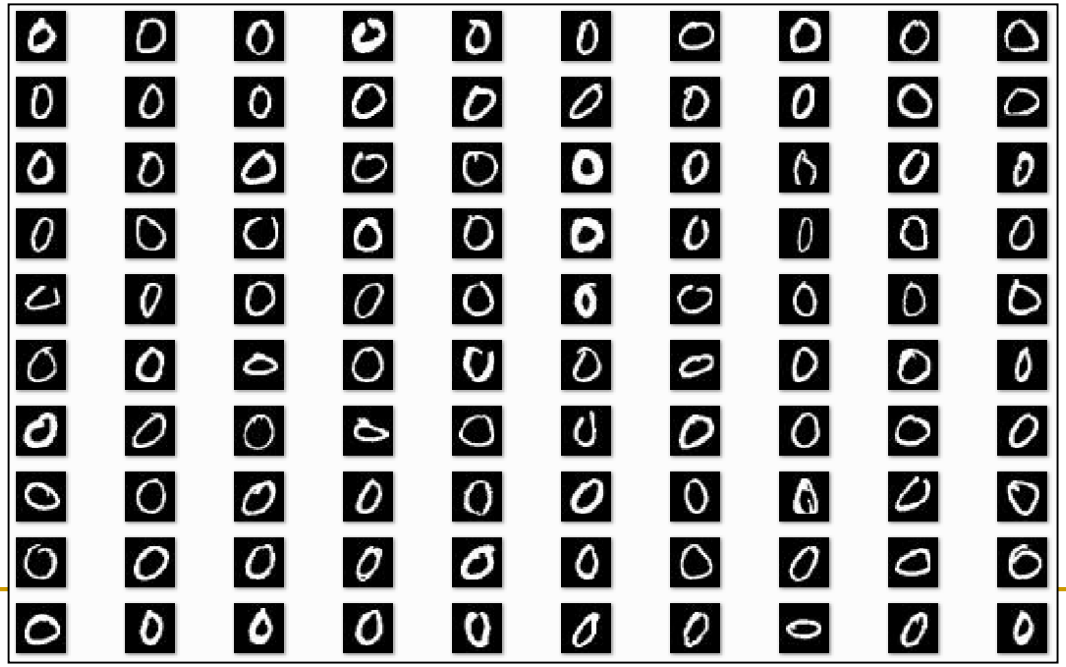


最近傍法の例題

MNIST

MNIST

- 数字画像データベース
  - Mixed National Institute of Standards and Technology database
  - http://yann.lecun.com/exdb/mnist/



MNIST

- 学習データ
  - mnist/train/以下
  - 100文字×10文字種
- 評価データ
  - mnist/test/以下
  - 100文字×10文字種

*実データは学習データが60,000文字，評価データが10,000文字あります

最近傍法のプログラム

- NN.py
  - プロトタイプが一字種につき一文字
- NN-1.py
  - プロトタイプが一字種につき複数文字

実行方法

- 必要なパッケージ（Anacondaにはインストール済）
  - numpy，PIL（pillow）が必要
  - インストール方法
  - > pip install numpy
  - > pip install pillow

- 実行方法
  - > python NN.py
  - > python NN-1.py

最近傍法のプログラム（NN.py）①

```
import sys
import os
import random
import numpy as np
from PIL import Image

train_img = np.zeros((10,28,28), dtype=np.float32)

for i in range(10):
    num = random.randint(1,100)
    train_file = "mnist/train/" + str(i) + "/" + str(i) + "_" + str(num) + ".jpg"

    train_img[i] =
            np.asarray(Image.open(train_file).convert('L')).astype(np.float32)
```

ライブラリイのインポート

プロトタイプ（10枚）の画像の読み込み

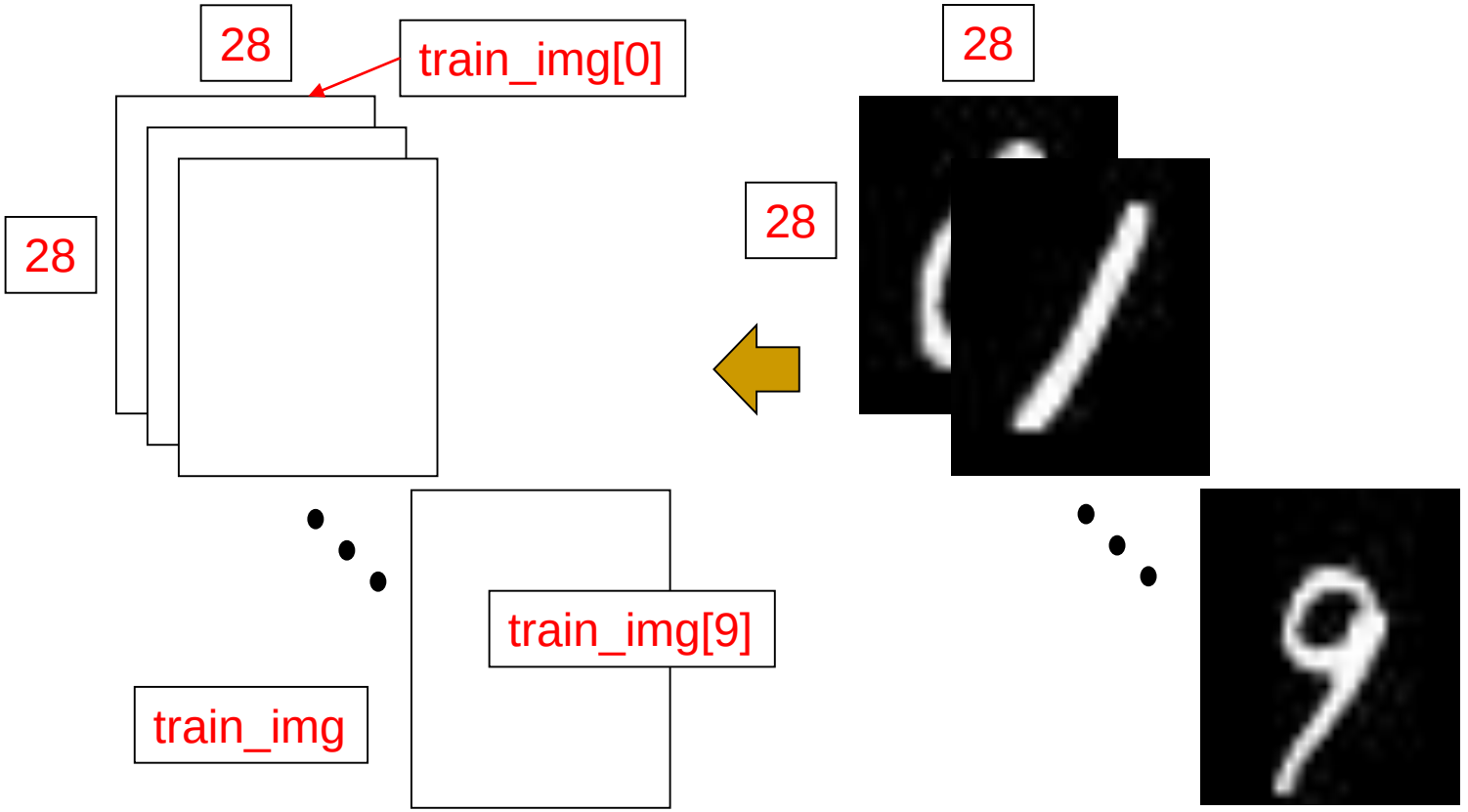
100枚の画像からランダムに選択

読み込む画像のファイル名

グレースケール画像として読み込み→numpyに変換

プロトタイプの読み込み

```
train_img = np.zeros((10,28,28), dtype=np.float32)
```




```
train_img[i] =
np.asarray(Image.open(train_file).convert('L')).astype(np.float32)
```

最近傍法のプログラム（NN.py）②

未知パターンの読み込み

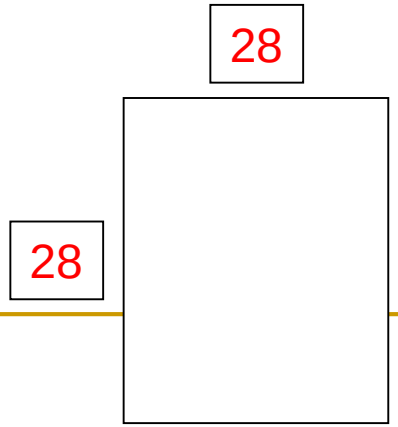
```
result = np.zeros((10,10), dtype=np.int32)
for i in range(10):
    for j in range(1,101):
        pat_file = "mnist/train/" + str(i) + "/" + str(i) + "_" + str(j) + ".jpg"
        pat_img = np.asarray(Image.open(pat_file).convert('L')).astype(np.float32)
```

読み込む画像のファイル名

グレースケール画像として読み込み→numpyに変換

配列result：混合行列（10×10の大きさ）

pat_img：未知パターン



最近傍法のプログラム（NN.py）③

```
        min_val = float('inf')
        ans = 0
        for k in range(10):
            t = train_img[k].flatten()
            p = pat_img.flatten()
            dist = np.dot( (t-p).T , (t-p) )
            if dist < min_val:
                min_val = dist
                ans = k

        result[i][ans] +=1
        print( i , j , "->" , ans )

print( "[混合行列]" )
print( result )
print( "正解数 ->" ,  np.trace(result) )
```

SSDの計算

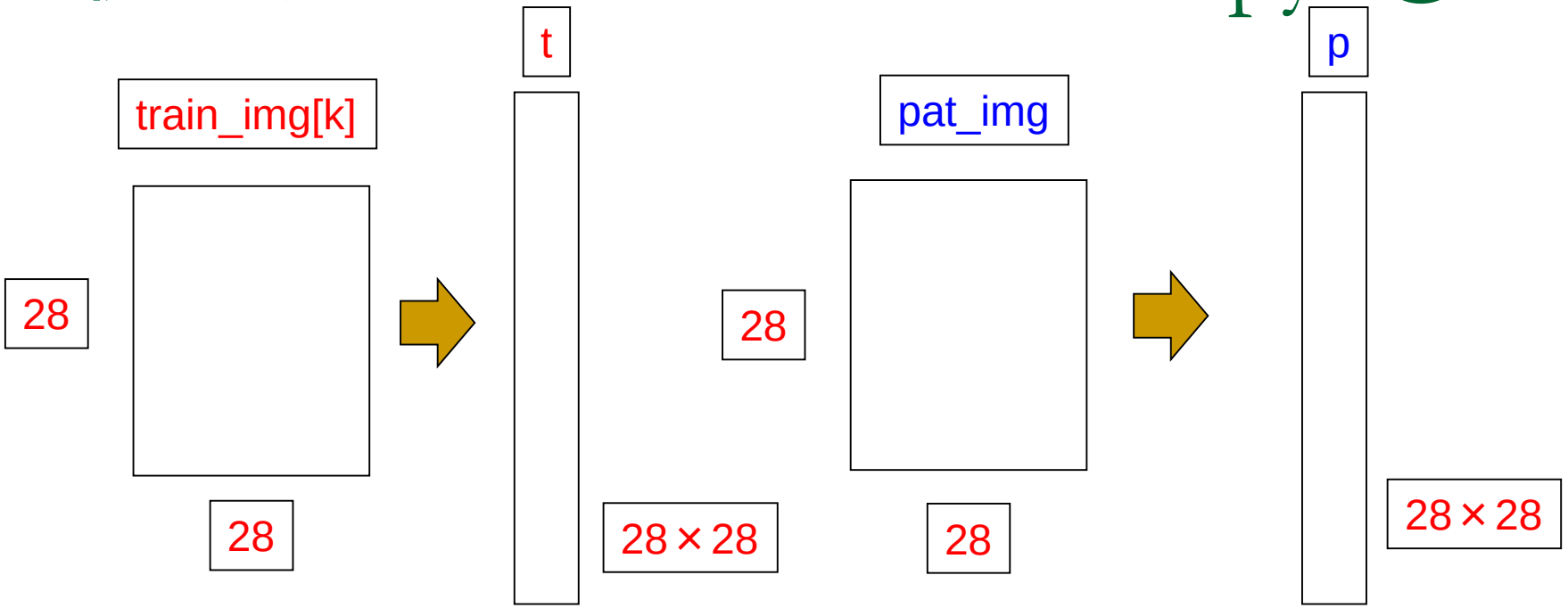
最小値の探索

結果の出力

混合行列の出力

正解数の出力

最近傍法のプログラム（NN.py）④



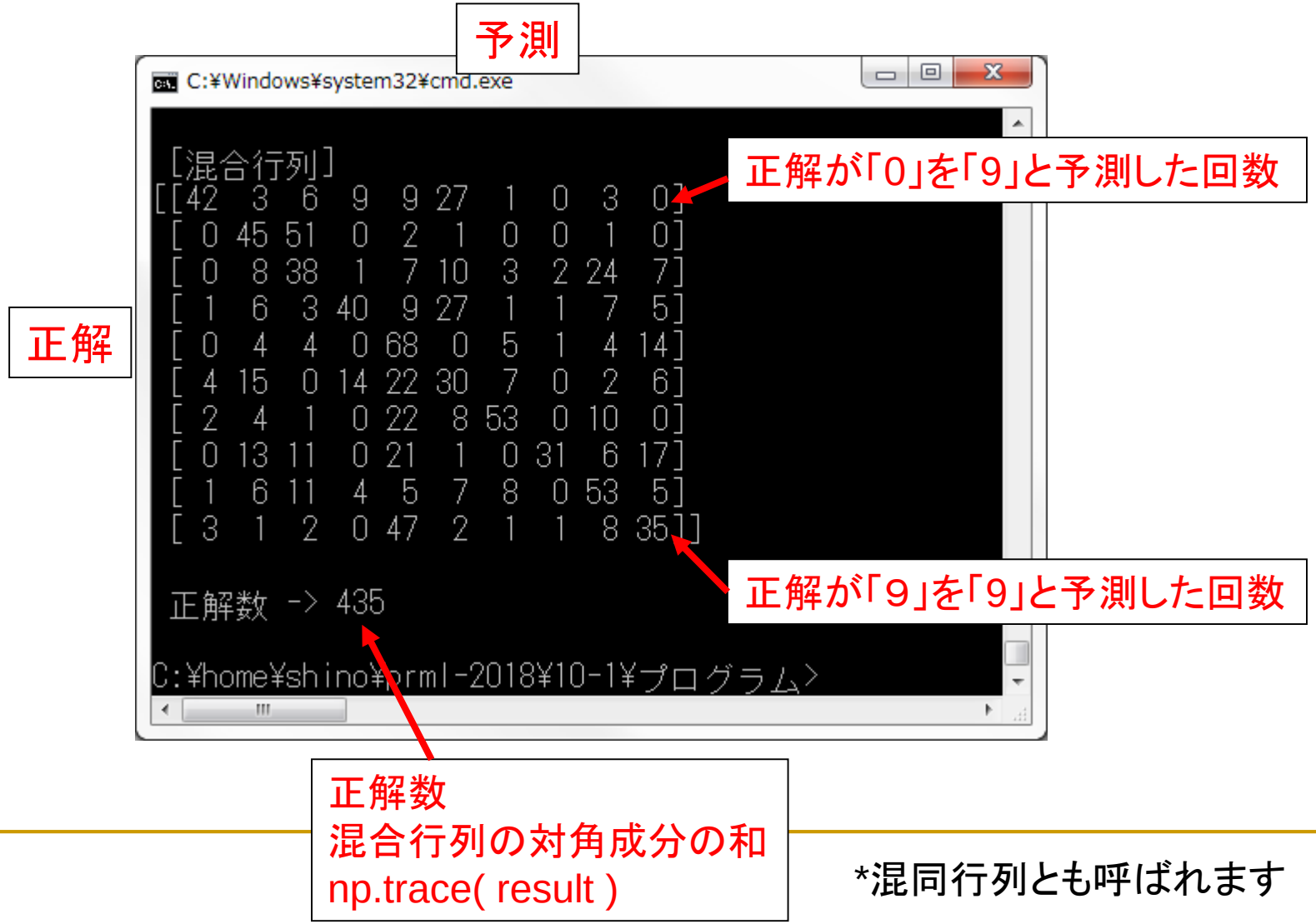
t = train_img[k].flatten()

p = pat_img.flatten()

SSDの計算

dist = np.dot((t-p).T , (t-p))

混合行列*(Confusion Matrix)



*混同行列とも呼ばれます

最近傍法のプログラム（NN-1.py）①

```
import sys
import os
import numpy as np
from PIL import Image

train_num = 10
train_img = np.zeros((10,train_num,28,28), dtype=np.float32)

for i in range(10):
    for j in range(1,train_num+1):
        train_file = "mnist/train/" + str(i) + "/" + str(i) + "_" + str(j) + ".jpg"
        train_img[i][j-1] =
            np.asarray(Image.open(train_file).convert('L')).astype(np.float32)
```

ライブラリイのインポート

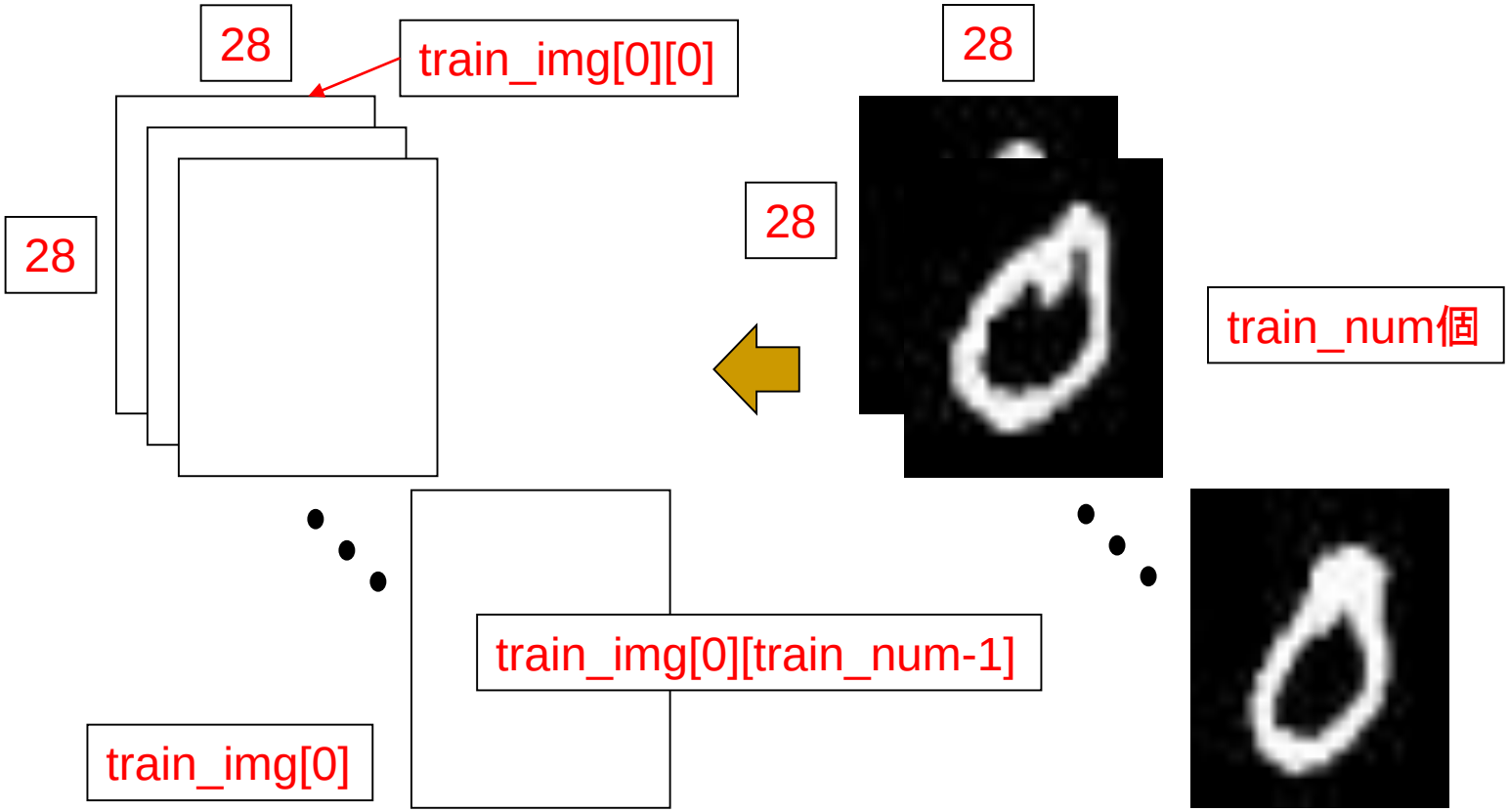
一字種につきtrain_num個読み込む

読み込む画像のファイル名

グレースケール画像として読み込み→numpyに変換

プロトタイプの読み込み

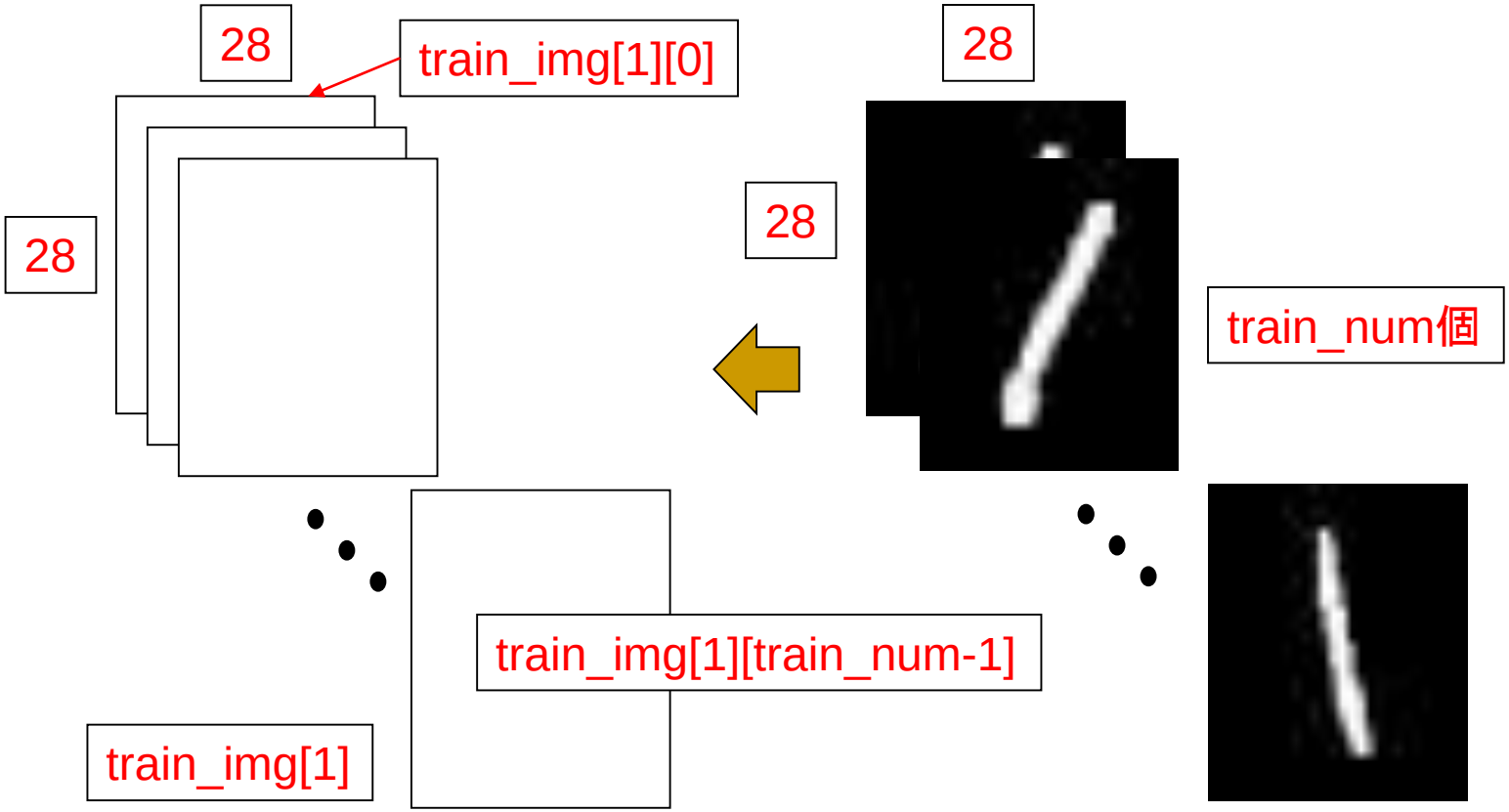
```
train_img = np.zeros((10,train_num,28,28), dtype=np.float32)
```




```
train_img[i][j-1] =
np.asarray(Image.open(train_file).convert('L')).astype(np.float32)
```

プロトタイプの読み込み

```
train_img = np.zeros((10,train_num,28,28), dtype=np.float32)
```




```
train_img[i][j-1] =
np.asarray(Image.open(train_file).convert('L')).astype(np.float32)
```

最近傍法のプログラム（NN-1.py）②

未知パターンの読み込み

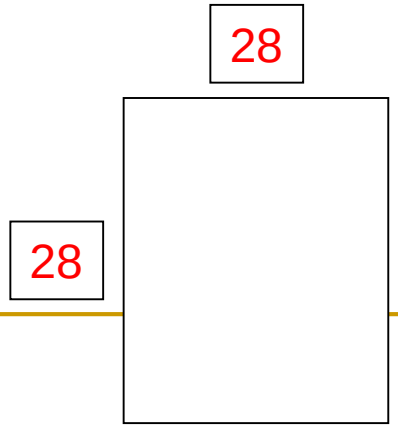
```
result = np.zeros((10,10), dtype=np.int32)
for i in range(10):
    for j in range(1,101):
        pat_file = "mnist/test/" + str(i) + "/" + str(i) + "_" + str(j) + ".jpg"
        pat_img = np.asarray(Image.open(pat_file).convert('L')).astype(np.float32)
```

読み込む画像のファイル名

グレースケール画像として読み込み→numpyに変換

配列result：混合行列（10×10の大きさ）

pat_img：未知パターン



最近傍法のプログラム（NN-1.py）③

```
        min_val = float('inf')
        ans = 0
        for k in range(10):
            for l in range(1,train_num+1):
                t = train_img[k][l-1].flatten()
                p = pat_img.flatten()
                dist = np.dot( (t-p).T , (t-p) )

                if dist < min_val:
                    min_val = dist
                    ans = k

        result[i][ans] +=1
        print( i , j , "->" , ans )

print( "¥n [混合行列]" )
print( result )
print( "¥n 正解数 ->" ,  np.trace(result) )
```

SSDの計算

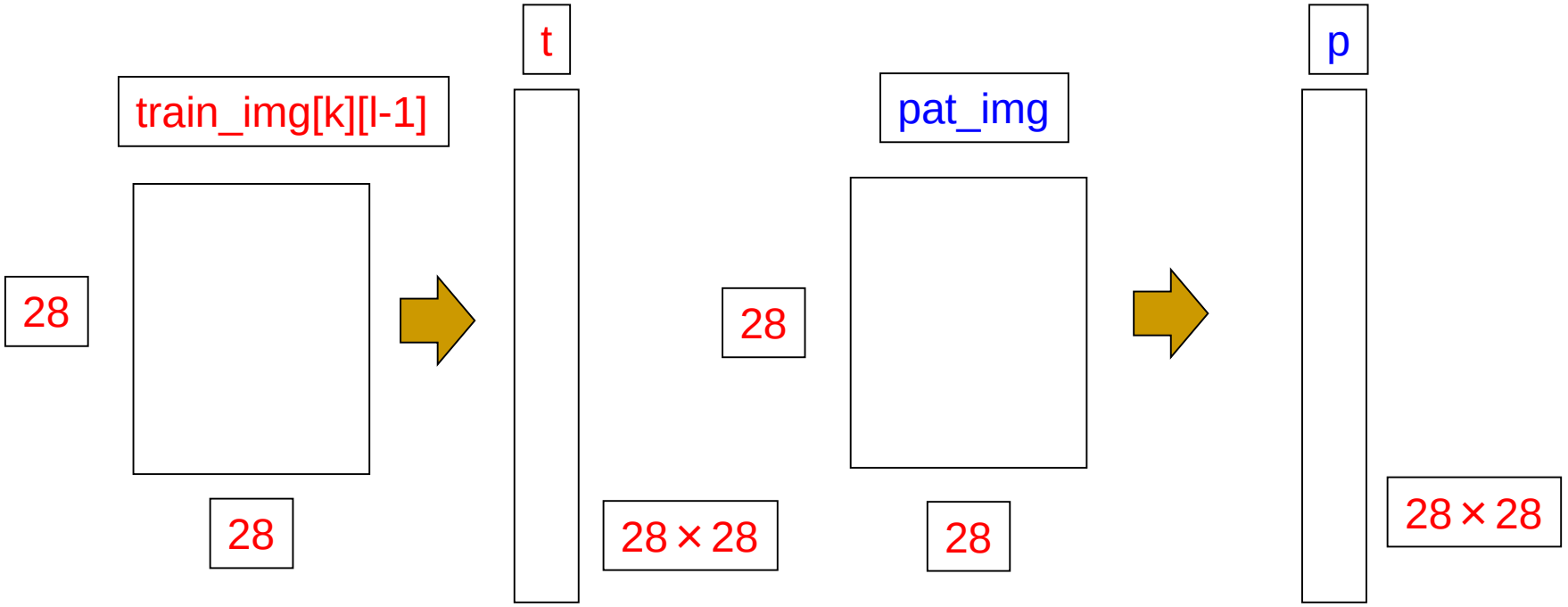
最小値の探索

結果の出力

混合行列の出力

正解数の出力

最近傍法のプログラム（NN-1.py）④



t = train_img[k][l-1].flatten()

p = pat_img.flatten()

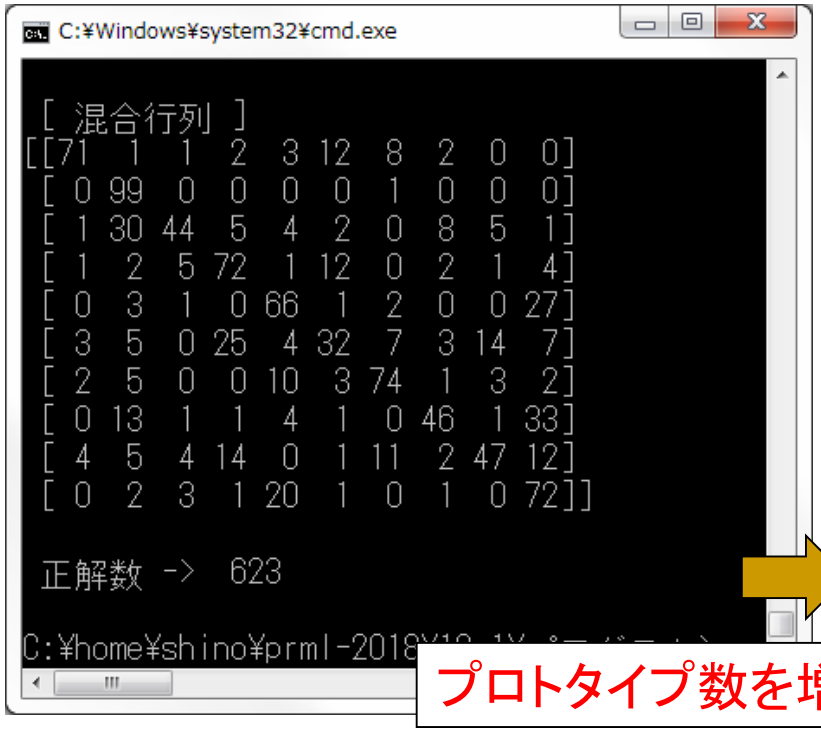
SSDの計算

dist = np.dot((t-p).T , (t-p))

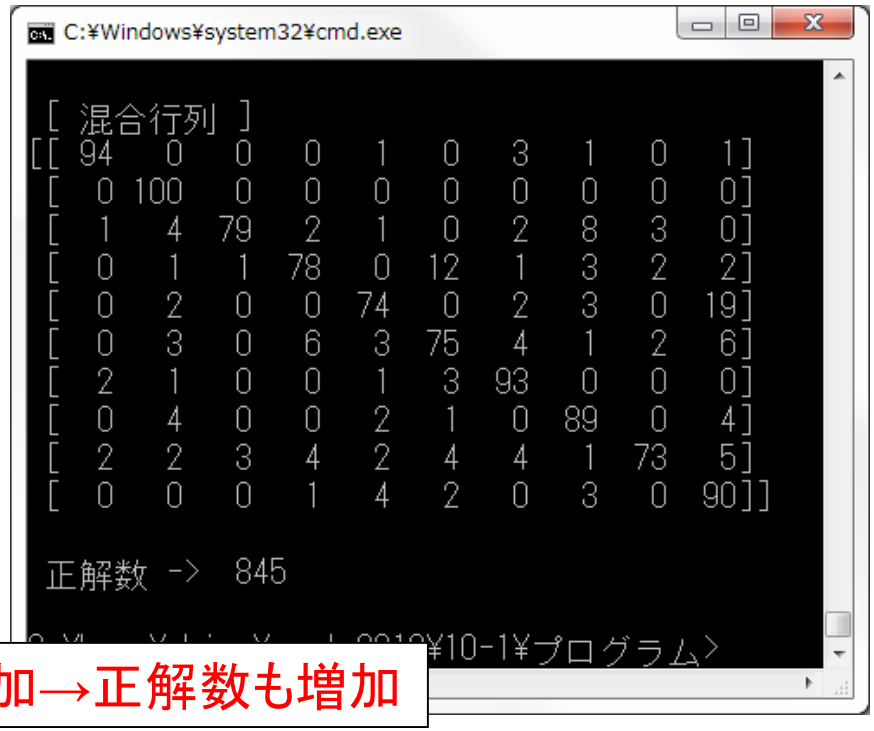
結果の出力

- 混合行列

train_num = 10



train_num = 100



プロトタイプ数を増加→正解数も増加

宿題②

- NN-1.pyを元にk近傍法のプログラムを作成しなさい.
  - kはキーボードから入力できるようにするとよいです
  - ただし，k=2,3,・・・としても正解率が必ずしも向上するわけではありません

（本日の）参考文献

- 石井健一郎他：わかりやすいパターン認識，オーム社（1998）
- 森健一：パターン認識，電子情報通信学会（1988）
- 平井有三：はじめてのパターン認識，森北出版（2012）