機械学習

名古屋大学大学院 情報学研究科

出口 大輔

講義内容

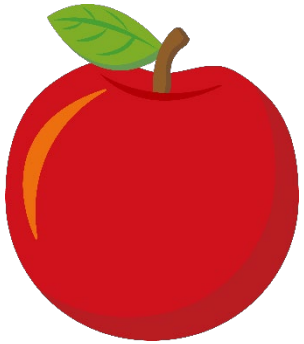
- 機械学習の基礎
  - 最近傍法
  - K最近傍法
  - 線形判別分析法

- 演習（Pythonによる実践機械学習プログラミング）
  - Pythonによる機械学習環境の構築
    - Google Colaboratory の準備
  - 手書き文字認識
    - Nearest Neighbor（最近傍法）

機械学習の基礎

まずはじめに・・・

- 自動で「りんご」を分類／認識する機械を作りたい！
- 何らかの特徴（丸さ，色，…）を測れば分類できる？
  - 特徴を数値の並び（ベクトル）として扱うことを考える



入力（画像）データ

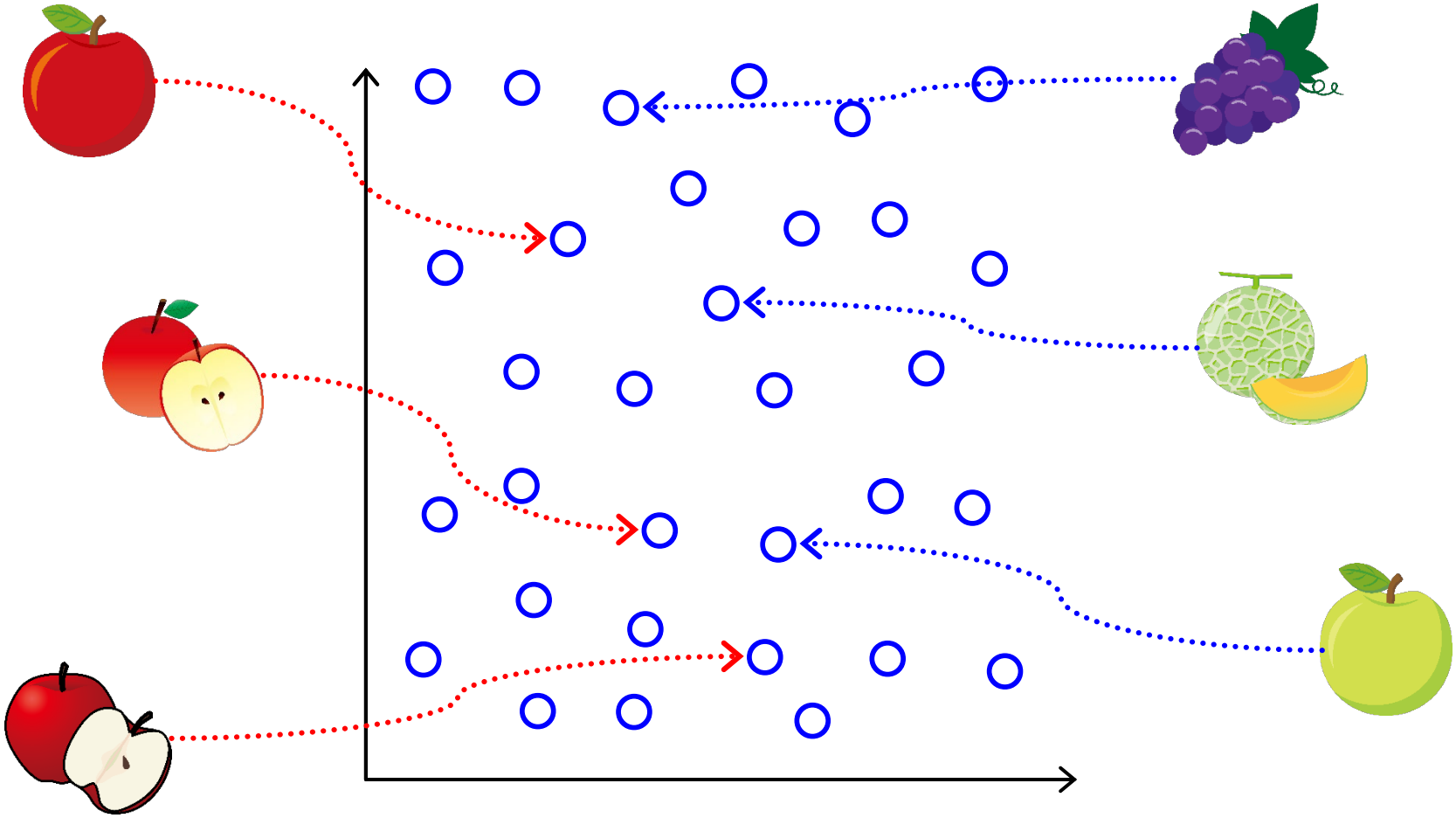
特徴量の記述

$$\begin{pmatrix} 0.8 \\ 1.0 \\ 0.2 \\ 0.2 \\ \vdots \end{pmatrix}$$

ベクトル化

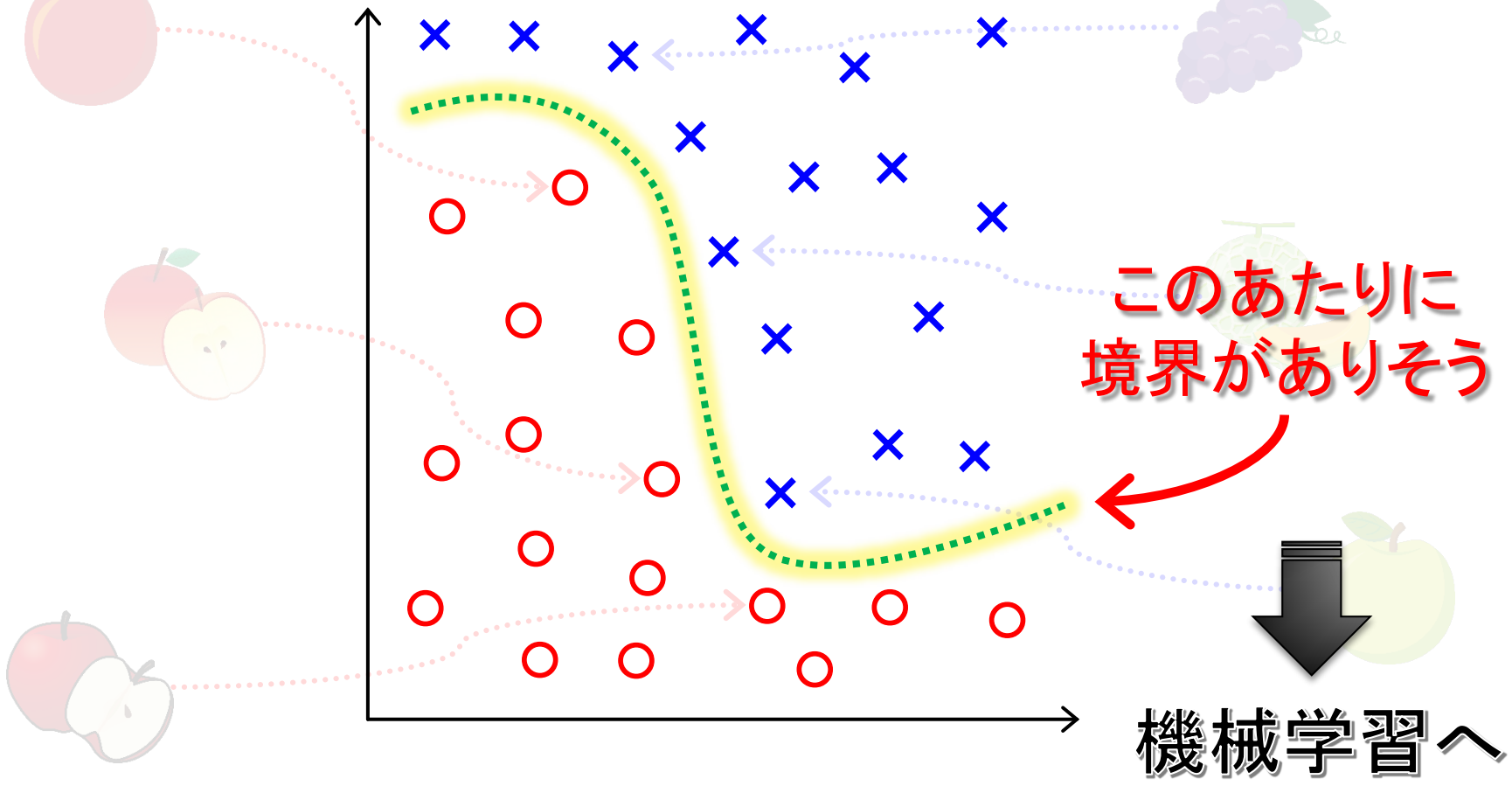
まずはじめに・・・

- いろいろな果物の特徴（ベクトル）をプロットすると？



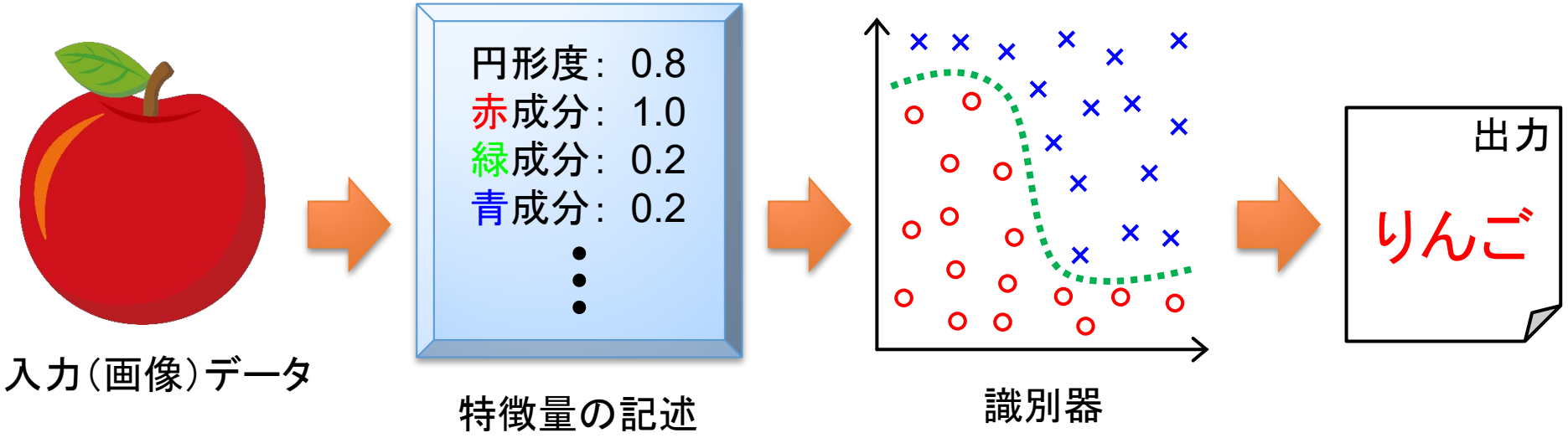
まずはじめに・・・

- いろいろな果物の特徴（ベクトル）をプロットすると？
  - 「りんご」かどうかのラベル（＝教師データ）があれば・・・



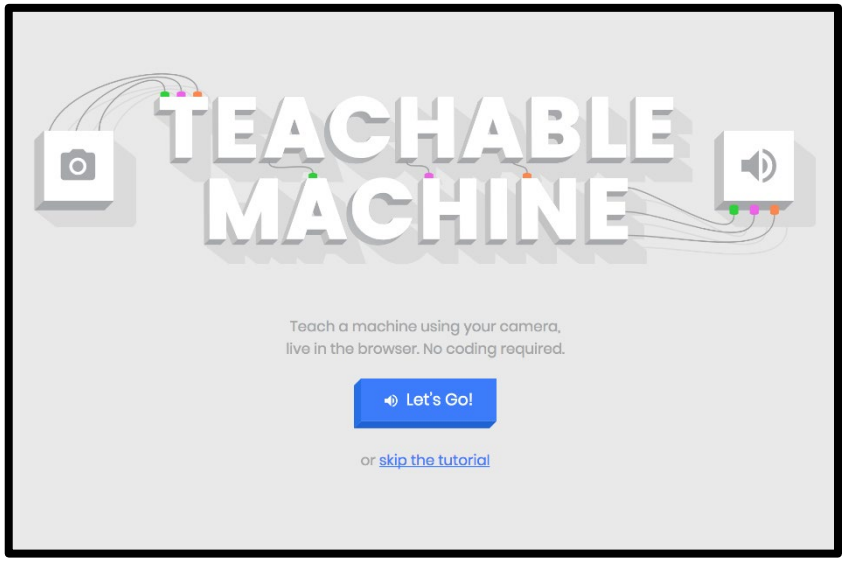
機械学習とは？

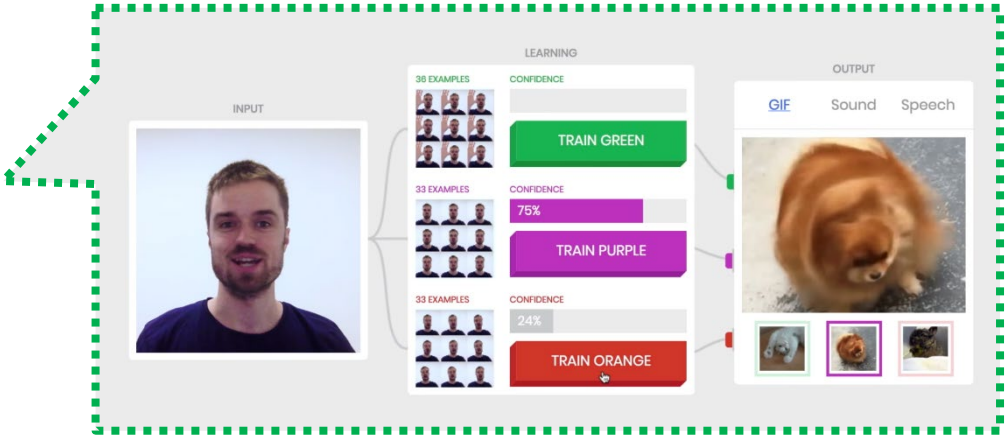
- 人間が自然に行っている学習能力と同様の機能をコンピュータで実現しようとする技術[Wikipediaより引用]
  - データの集合から何らかのルールや法則を計算機に学習させる技術



機械学習でどんなことができるの？

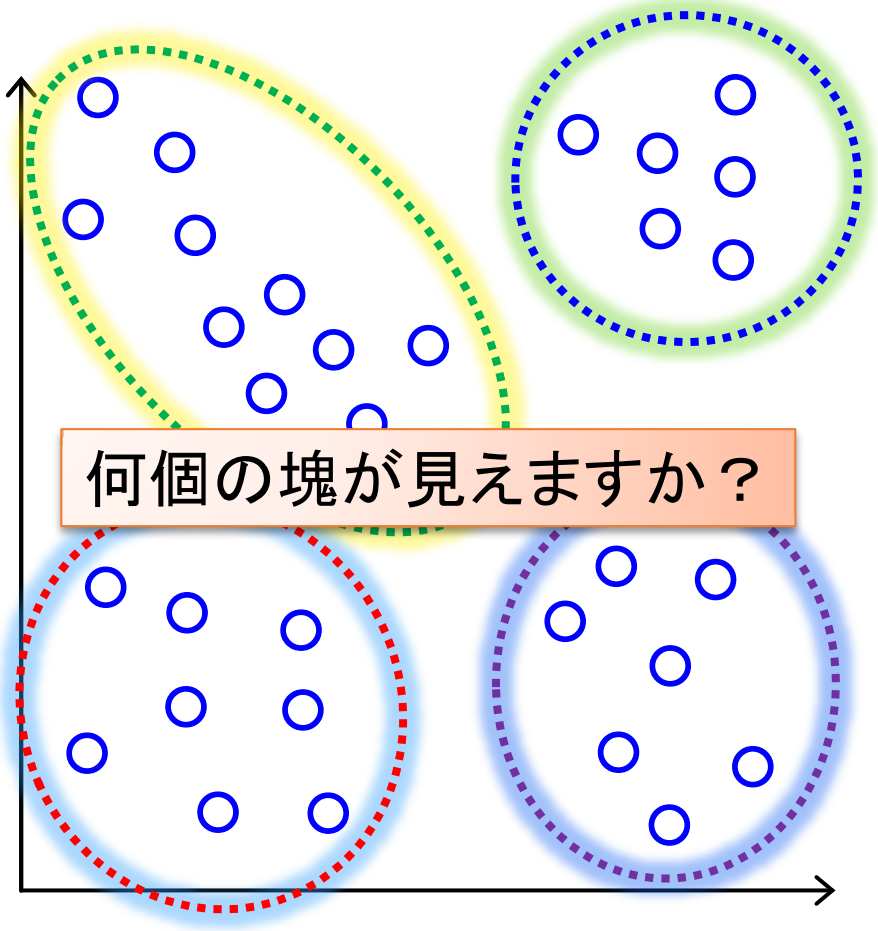
- Teachable Machine で試してみよう！
  - WEBブラウザーを使って機械学習を体験できるサイト
  - 2017年にGoogleが公開



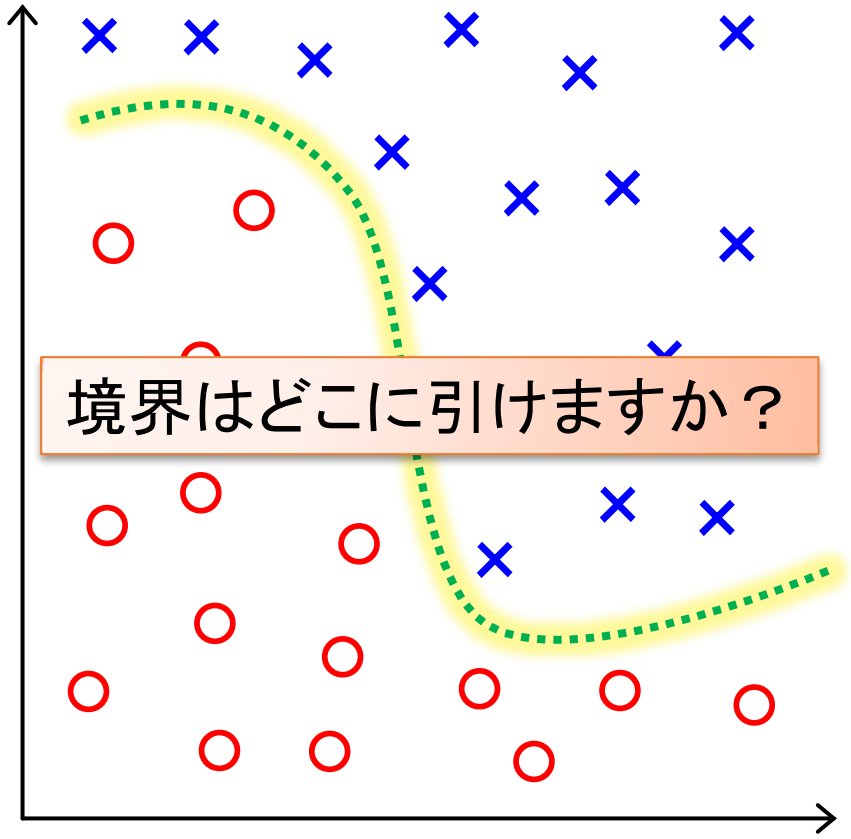


https://teachablemachine.withgoogle.com/

機械学習の種類



教師なし学習

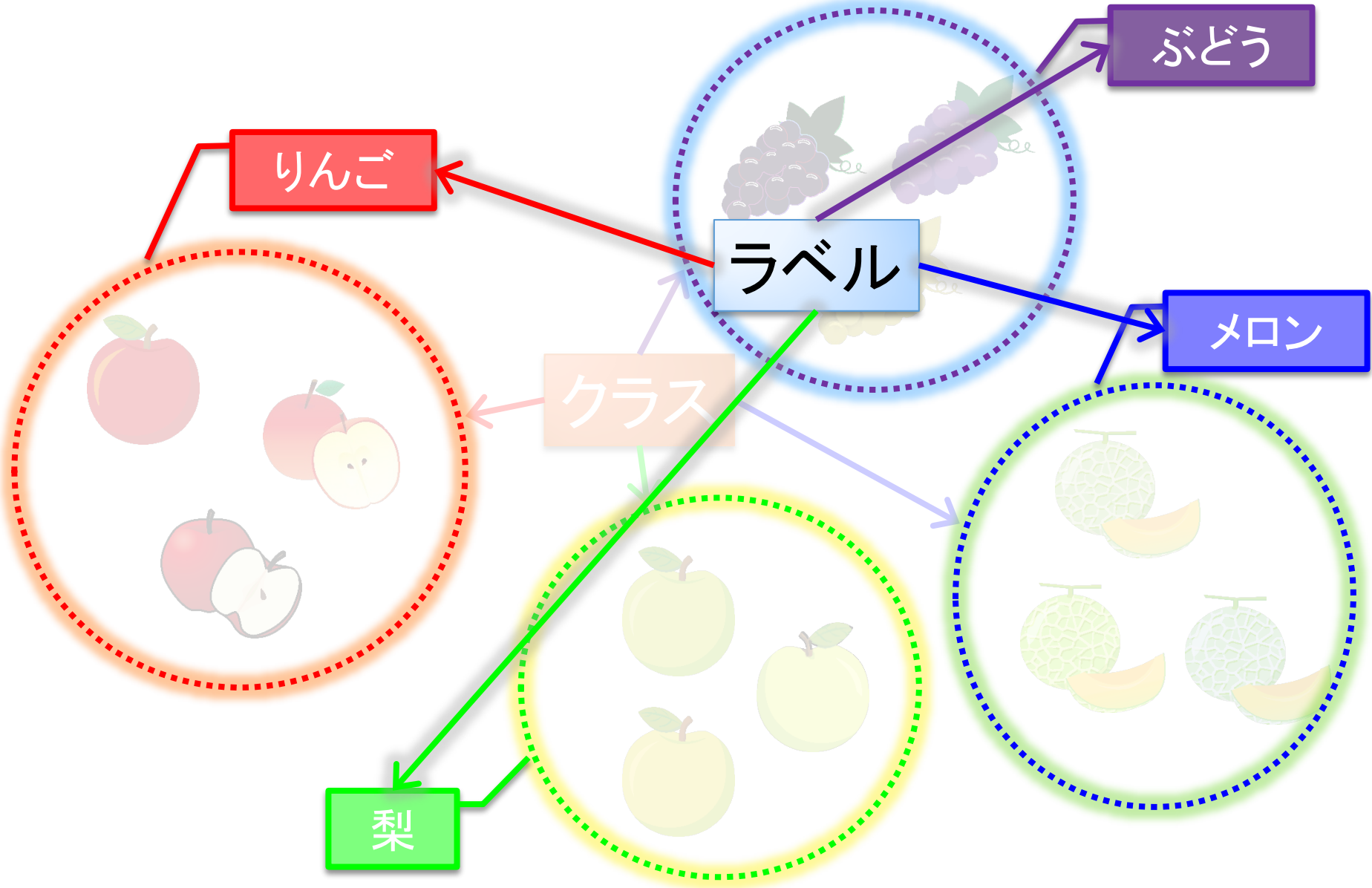


教師あり学習

機械学習の種類

- 教師あり学習
  - 教師データ(入力と出力のペア)を与えてその関係性を表現するモデルを構築する方法
    - 入力から出力への写像を求める問題(回帰，分類，など)
- 教師なし学習
  - 入力データのみからモデルを構築する方法
    - データの分類(クラスタリング)など
- その他
  - 半教師あり学習 ・・・ 教師ありと教師なしの折衷案
  - オンライン学習 ・・・ 動的に得られるデータを使って学習
  - 強化学習 ・・・ 行動に対する報酬を最大化するように学習

機械学習の用語（クラスとラベル）

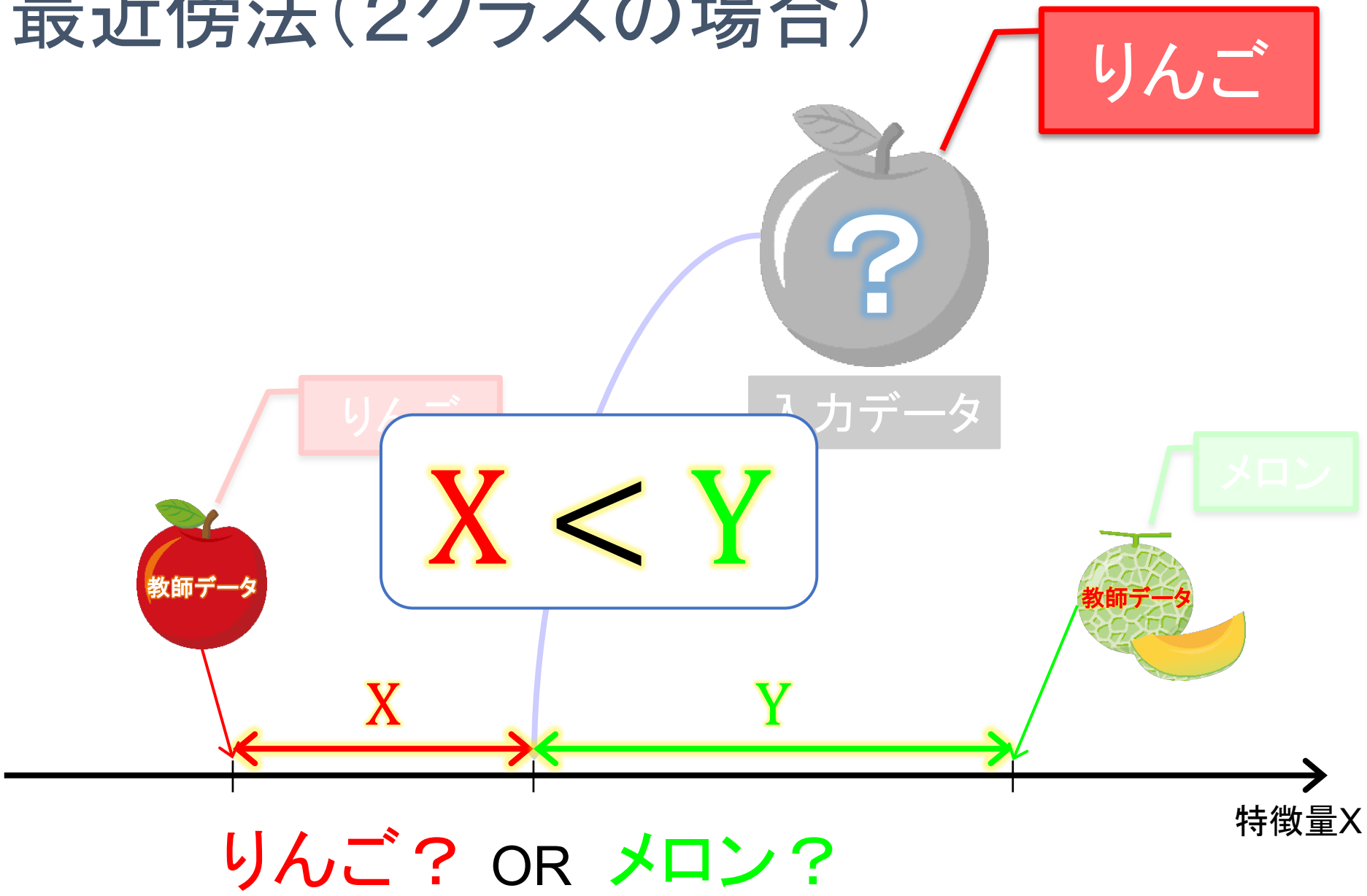


教師あり学習

まずはシンプルに

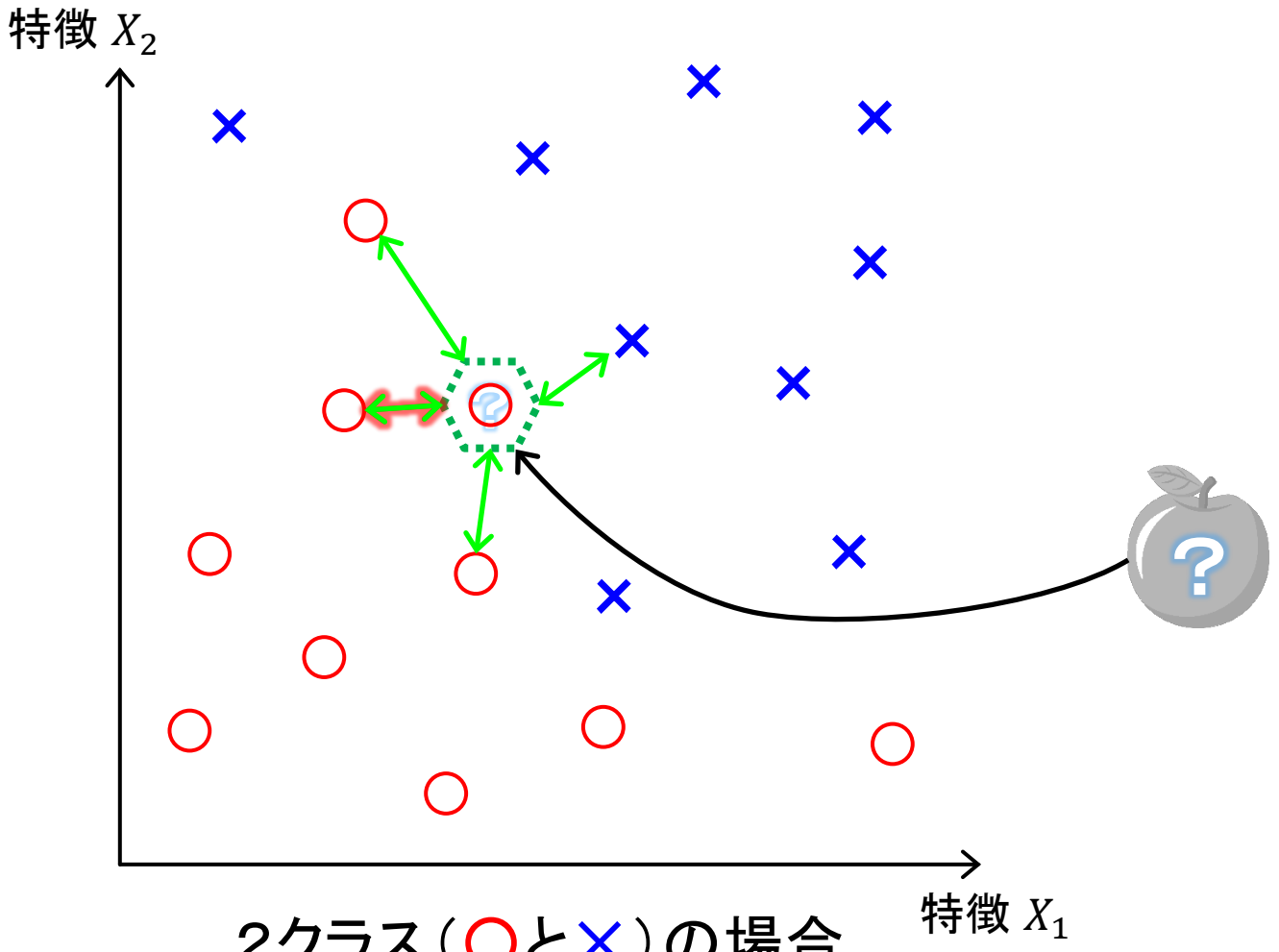
～最近傍法～

最近傍法（2クラスの場合）



最近傍法（2クラスの場合）

- 入力データに最も近い教師データを出力する手法



2クラス（○と×）の場合

最近傍法（2クラスの場合）

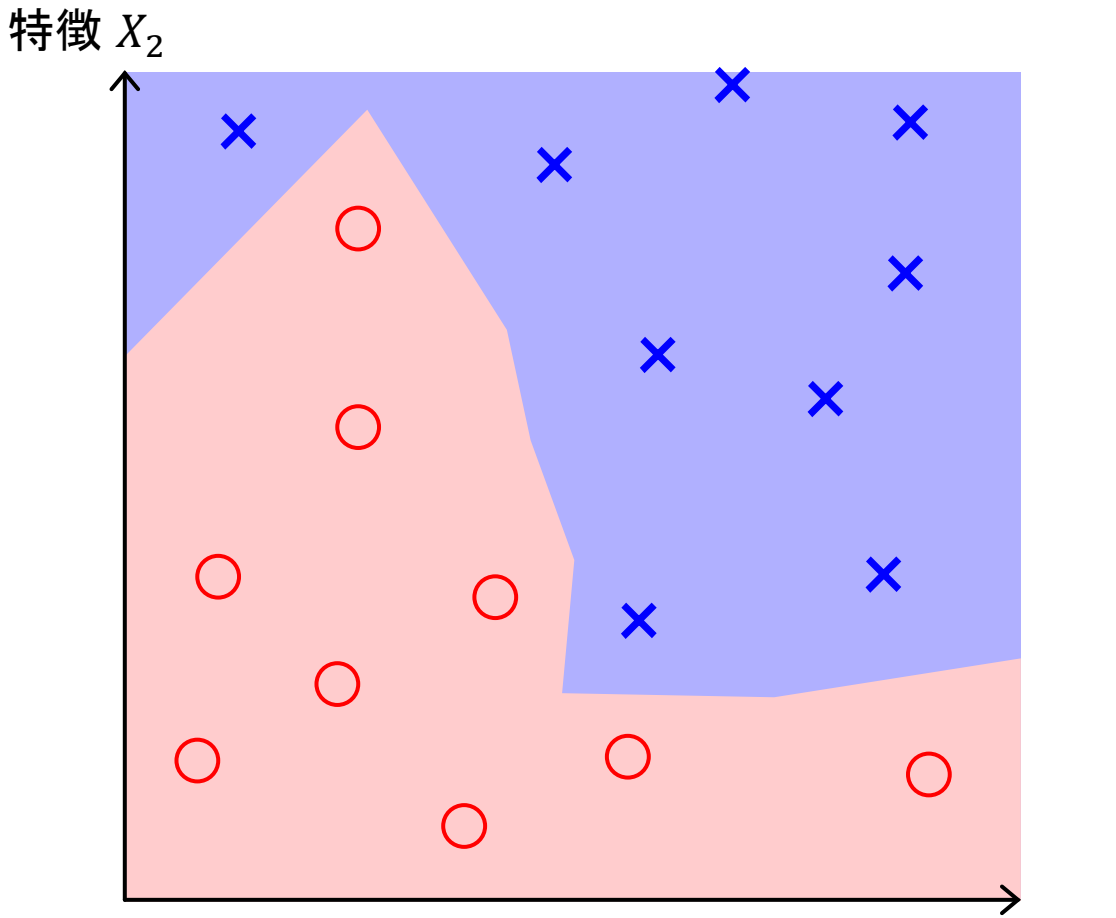
- 入力データに最も近い教師データを出力する手法
  - i 番目の教師データを $\boldsymbol{z}_i = \{\boldsymbol{x}_i, y_i\}$ とする
    - $\boldsymbol{x}_i = (x_1, x_2, x_3, \cdots)^T$ はN次元の特徴量ベクトル
    - y は $\boldsymbol{x}_i$ に対する教師ラベルであり ± 1 の値をとる

      ※ $+1$ はりんご（ポジティブデータ），-1 はメロン（ネガティブデータ）
  - 分類したいデータの特徴量を $\boldsymbol{x}$ とするとそのラベル $\hat{y}$ は？

$$\hat{y}(\boldsymbol{x}) = y_{\tilde{\imath}}$$
$$\tilde{\imath} = \underset{i}{\operatorname{argmin}} \ \|\boldsymbol{x} - \boldsymbol{x}_i\|$$

最近傍法（2クラスの場合）

- 入力データに最も近い教師データを出力する手法



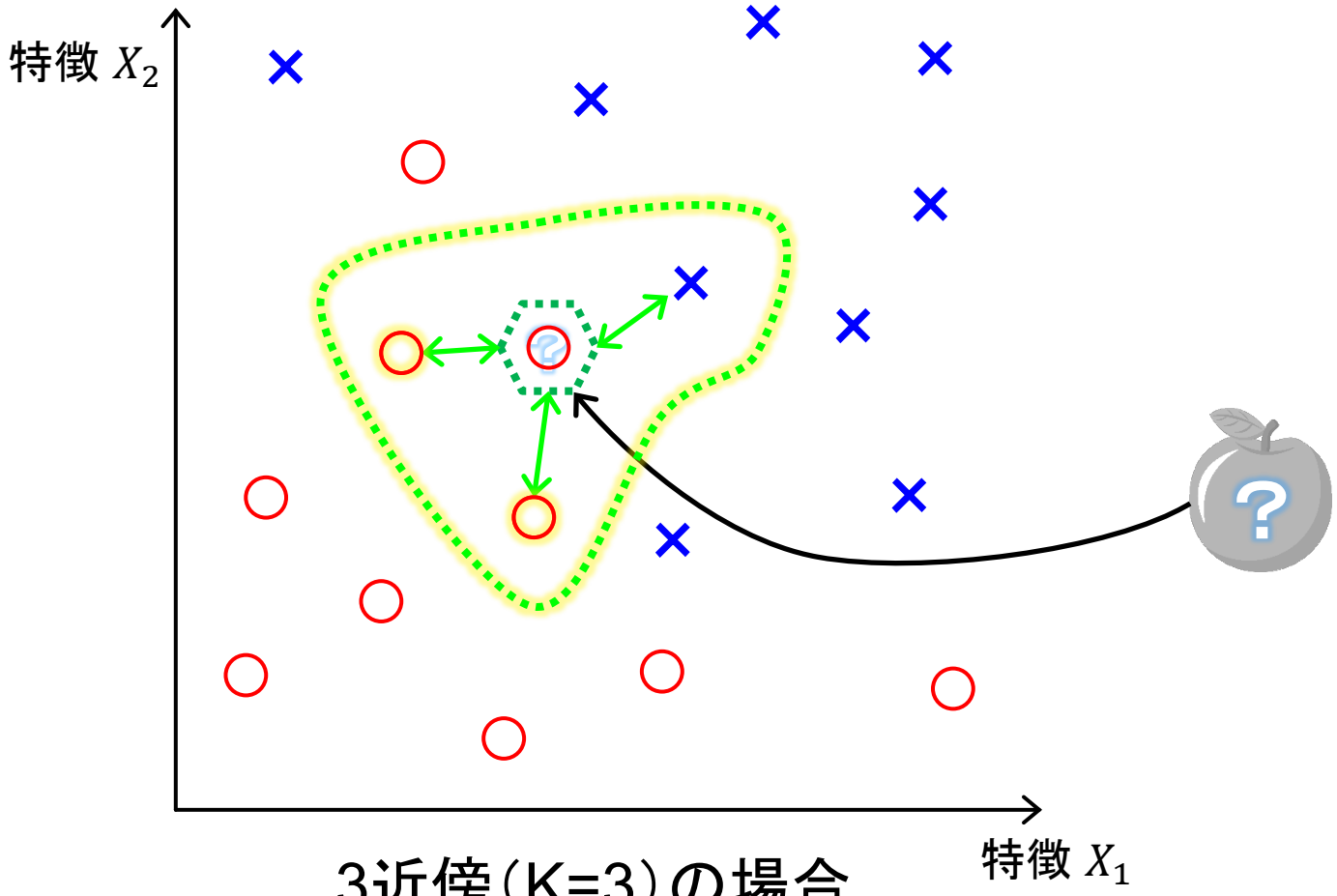
2クラス（○と×）の場合

近傍を拡張すると・・・

～K最近傍法～

K最近傍法（2クラスの場合）

- 入力データのK近傍内で最も多く現れる教師データを出力する手法



3近傍（K=3）の場合

K最近傍法（2クラスの場合）

- 入力データのK近傍内で最も多く現れる教師データを出力する手法
  - i 番目の教師データを $\boldsymbol{z}_i = \{\boldsymbol{x}_i, y_i\}$ とする
    - □ $\boldsymbol{x}_i = (x_1, x_2, x_3, \cdots)^T$ はN次元の特徴量ベクトル
    - □ y は $\boldsymbol{x}_i$ に対する教師ラベルであり ± 1 の値をとる
      ※ $+1$ はりんご（ポジティブデータ），-1 はメロン（ネガティブデータ）
  - 分類したいデータの特徴量を $\boldsymbol{x}$ とするとそのラベル $\hat{y}$ は？

$$\hat{y}(\boldsymbol{x}) = \begin{cases} +1 & \text{if } 0 \leq \dfrac{1}{K} \displaystyle\sum_{\boldsymbol{x}_i \in \mathcal{N}_K(\boldsymbol{x})} y_i \\ -1 & otherwise \end{cases}$$

※ $\mathcal{N}_K(\boldsymbol{x})$ は $\boldsymbol{x}$ のK最近傍

$\boldsymbol{x}$ から近い順にK個集めた集合

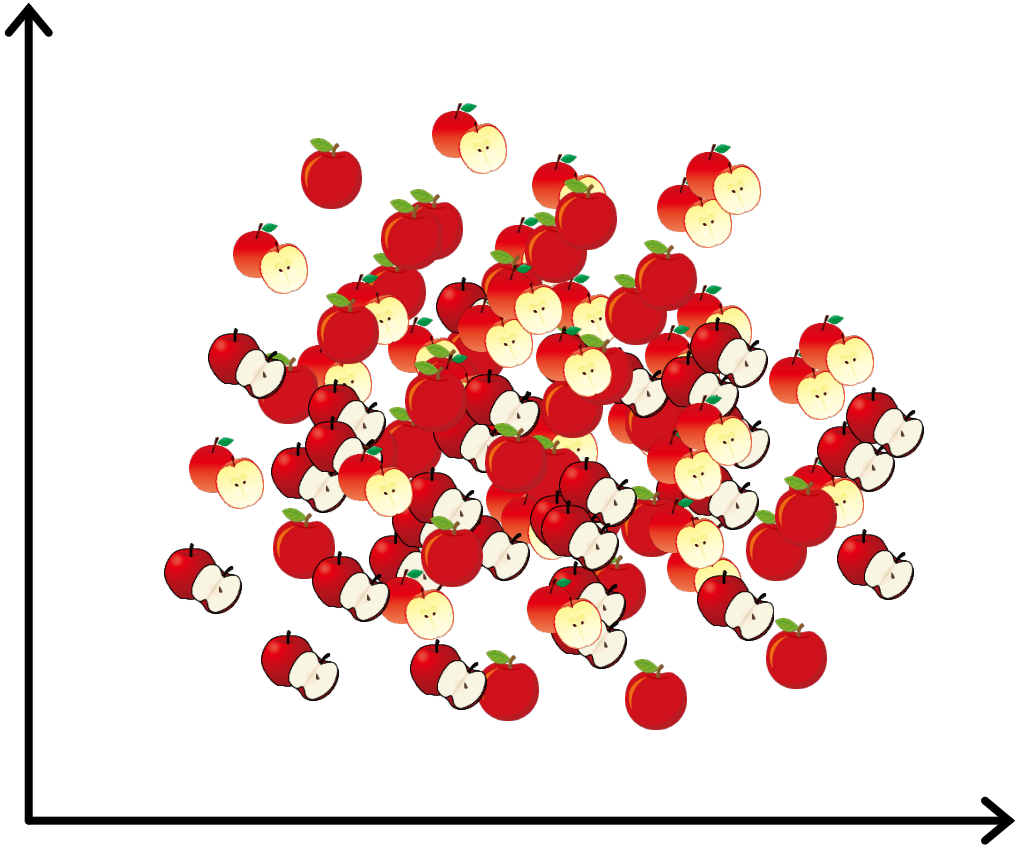
【メリット】分類（識別）境界が滑らかなる

もう少し確率的に・・・

～線形判別分析～

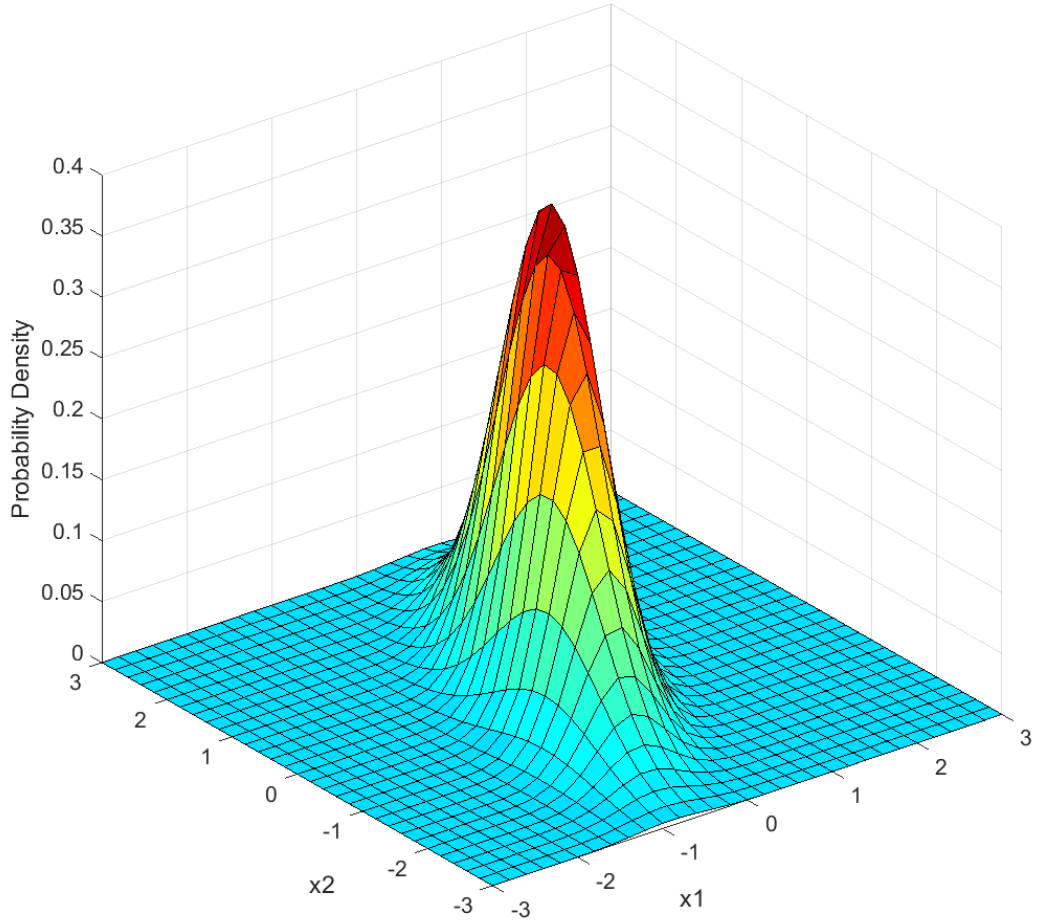
データの確率的な表現

- 大量のりんごから求めた特徴量はどんな分布？



データの確率的な表現

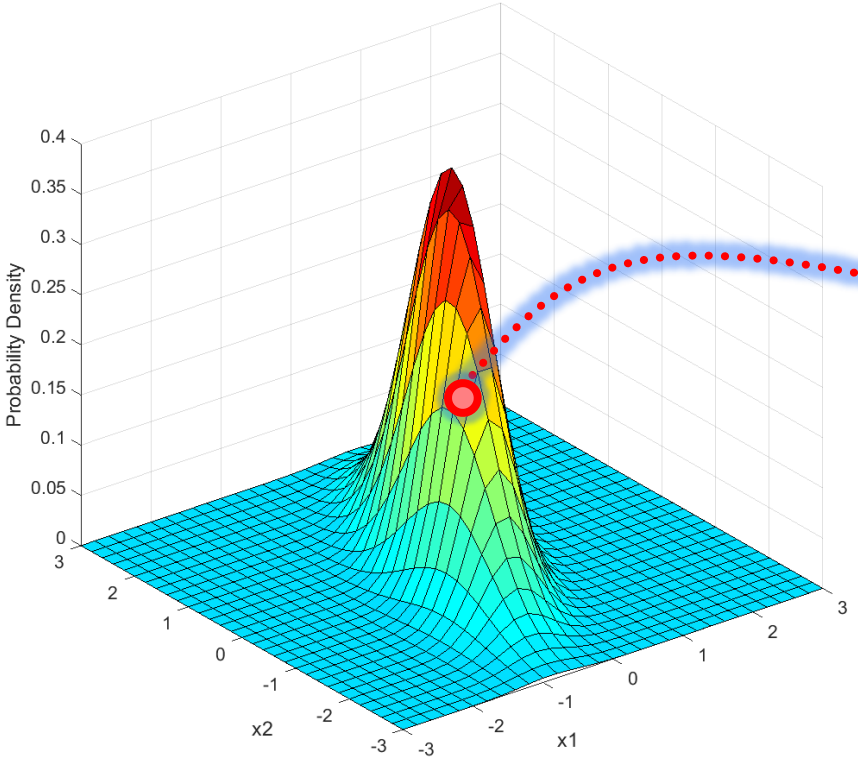
- 大量のりんごから求めた特徴量はどんな分布？
  - 中心の密度（リンゴの密集度）が高い分布になる



データの確率的な表現

- 大量のりんごから求めた特徴量はどんな分布？
  - 中心の密度（リンゴの密集度）が高い分布になる
- ある特徴量Xが与えられたら，それはどれくらい「りんご」らしいでしょう？

$= \Pr(\text{りんご}|\boldsymbol{x})$



確率60%くらいで「りんご」っぽい

クラス確率を利用して分類

線形判別分析（2クラスの場合）

- 入力 $\boldsymbol{x}$ がクラス$+1$（もしくは-1）である確率を考える
  - $+1$ である確率は $\Pr(+1|\boldsymbol{x})$
  - -1 である確率は $\Pr(-1|\boldsymbol{x})$

$\Pr(+1|\boldsymbol{x}) \geq \Pr(-1|\boldsymbol{x})$ ならば $+1$，それ以外は -1 に分類

線形判別分析（2クラスの場合）

- 入力 $\boldsymbol{x}$ がクラス$+1$（もしくは-1）である確率を考える
  - $+1$ である確率は $\Pr(+1|\boldsymbol{x})$
  - -1 である確率は $\Pr(-1|\boldsymbol{x})$

$\Pr(+1|\boldsymbol{x}) \geq \Pr(-1|\boldsymbol{x})$ ならば $+1$，それ以外は -1 に分類

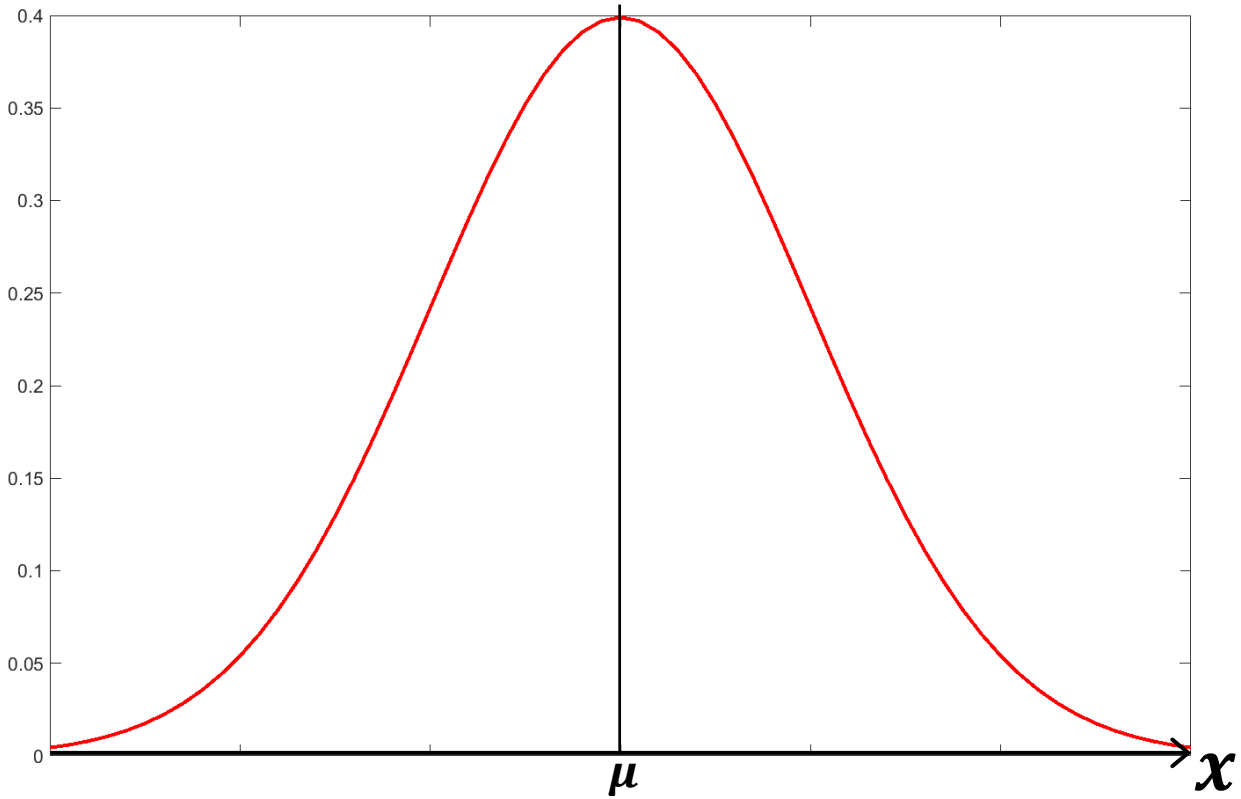
- ここで，$\Pr(\pm 1|\boldsymbol{x})$が正規分布に従うと仮定する

$$\Pr(k|\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{p}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(x-\boldsymbol{\mu}_k)}$$

※ $\boldsymbol{\mu}_k$はクラス毎の平均であり，共分散行列 $\boldsymbol{\Sigma}$ は全クラスで共通とする

正規分布？

- 確率論や統計学で非常に良く用いられる関数
  - 平均 $\boldsymbol{\mu}$ を中心，共分散行列 $\boldsymbol{\Sigma}$ で分布形状が決まる
  - 中心から離れるに従って関数値が減少



線形判別分析（2クラスの場合）

- $\Pr(+1|\boldsymbol{x}) \geq \Pr(-1|\boldsymbol{x})$ なら $+1$，それ以外は -1

$\dfrac{\Pr(+1|\boldsymbol{x})}{\Pr(-1|\boldsymbol{x})} \geq 1$ を調べることと等価

- クラス平均を $\boldsymbol{\mu}_k$ とし，共分散行列は全クラスで等しい $\boldsymbol{\Sigma}$ と仮定して左辺の対数を計算

$$\log \frac{\Pr(+1|\boldsymbol{x})}{\Pr(-1|\boldsymbol{x})} = \log \Pr(+1|\boldsymbol{x}) - \log \Pr(-1|\boldsymbol{x})$$

$$\Pr(\pm 1|\boldsymbol{x}) = \frac{1}{(2\pi)^{\frac{p}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_{\pm 1})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}-\boldsymbol{\mu}_{\pm 1})}$$

この部分は両クラス同じ

線形判別分析（2クラスの場合）

$$\log \frac{\Pr(+1|\boldsymbol{x})}{\Pr(-1|\boldsymbol{x})} = \log \Pr(+1|\boldsymbol{x}) - \log \Pr(-1|\boldsymbol{x})$$

$$= -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_{+1})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_{+1}) + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_{-1})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_{-1})$$

$$= -\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{+1} + \frac{1}{2}\boldsymbol{\mu}_{+1}{}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_{+1}{}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{+1}$$
$$+\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{-1} - \frac{1}{2}\boldsymbol{\mu}_{-1}{}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x} + \frac{1}{2}\boldsymbol{\mu}_{-1}{}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{-1}$$

$$= -\frac{1}{2}\boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1}) + \frac{1}{2}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})^T\boldsymbol{\Sigma}^{-1}\boldsymbol{x}$$
$$-\frac{1}{2}\boldsymbol{\mu}_{+1}{}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{+1} + \frac{1}{2}\boldsymbol{\mu}_{+1}{}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{-1} + \frac{1}{2}\boldsymbol{\mu}_{-1}{}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{-1} - \frac{1}{2}\boldsymbol{\mu}_{-1}{}^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_{+1}$$

$$-\frac{1}{2}\boldsymbol{\mu}_{+1}{}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1}) - \frac{1}{2}\boldsymbol{\mu}_{-1}{}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})$$

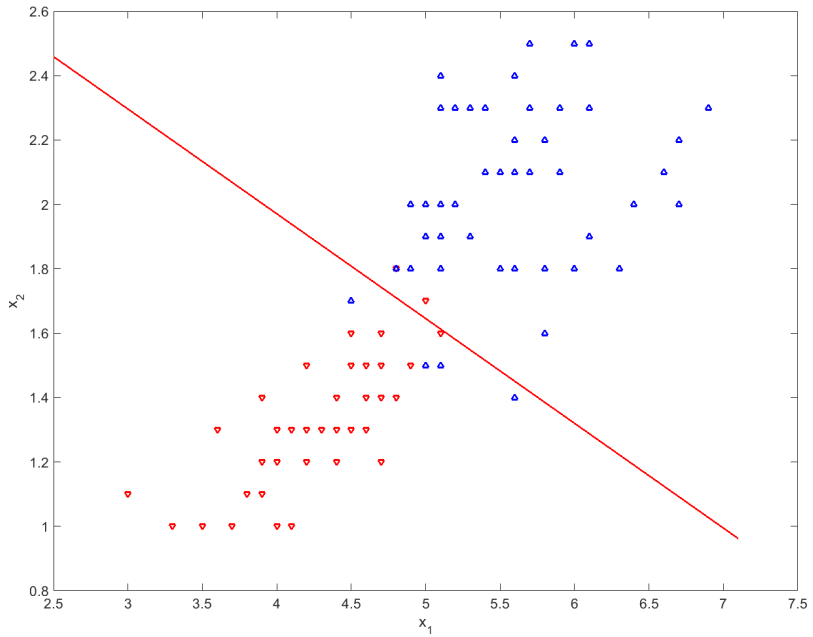
$$= \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1}) - \frac{1}{2}(\boldsymbol{\mu}_{+1}+\boldsymbol{\mu}_{-1})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})$$

線形判別分析（2クラスの場合）

- 入力 $\boldsymbol{x}$ がに対して下記を計算して分類

$$\log\frac{\Pr(+1|\boldsymbol{x})}{\Pr(-1|\boldsymbol{x})} = \boldsymbol{x}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1}) - \frac{1}{2}(\boldsymbol{\mu}_{+1}+\boldsymbol{\mu}_{-1})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_{+1}-\boldsymbol{\mu}_{-1})$$

$\log\frac{\Pr(+1|\boldsymbol{x})}{\Pr(-1|\boldsymbol{x})} \geq \log 1 = 0$ ならば $+1$，それ以外は -1



機械学習で何ができる？（その1）

https://www.cityscapes-dataset.com

※深層学習を利用したYOLOv3の結果

機械学習で何ができる？（その2）

https://www.cityscapes-dataset.com

※深層学習を利用したDeepLab v3+の結果

演習

Pythonによる実践機械学習プログラミング

まずはじめに演習の準備をしましょう(1)

- パソコンにログインし，ブラウザを開く
  - Googleアカウントが無い場合は以下よりアカウント作成
    - https://accounts.google.com/signup
  - 下記URLを入力してGoogle Colaboratoryにアクセス
    - https://colab.research.google.com

Colaboratory とは

Colaboratory（略称: Colab）では、ブラウザから Python を記述し実行できるほか、次の特長を備えています。

- 構成が不要
- GPU への無料アクセス
- 簡単に共有

Colab は、**学生**、**データ サイエンティスト**、**AI リサーチャー**の皆さんの作業を効率化します。詳しくは、Colab のご紹介をご覧ください。下からすぐに使ってみることもできます。

はじめに

ご覧になっているドキュメントは静的なウェブページではなく、**Colab ノートブック**という、コードを記述して実行できるインタラクティブな環境です。

たとえば次の**コードセル**には、値を計算して変数に保存し、結果を出力する短い Python スクリプトが含まれています。

```
[ ]  1  seconds_in_a_day = 24 * 60 * 60
     2  seconds_in_a_day
```

```
86400
```

上記のセルのコードを実行するには、セルをクリックして選択し、コードの左側にある実行ボタンをクリックするか、キーボード ショートカット「command+return」または「Ctrl+Enter」を使用します。コードはセルをクリックしてそのまま編集できます。

1 つのセルで定義した変数は、後で他のセルで使用できます。

```
[ ]  1  seconds_in_a_week = 7 * seconds_in_a_day
     2  seconds_in_a_week
```

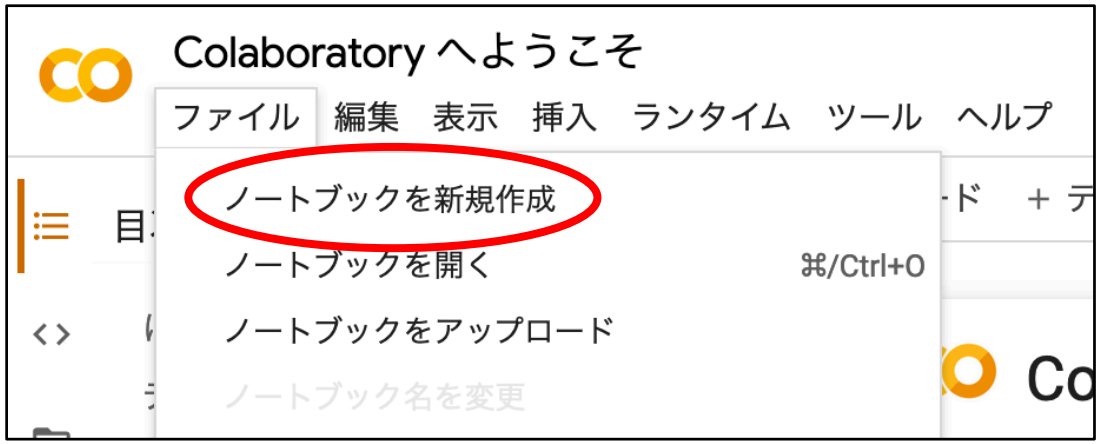
```
604800
```

Colab ノートブックを使用すると、**画像**、**HTML**、**LaTeX** などと合わせて、**実行可能コード**と**リッチテキスト**を 1 つのドキュメントに結合できます。Colab ノートブックを独自に作成すると、Google ドライブ アカウントに保存されます。Colab ノートブックは、同僚や友人と簡単に共有し、コメントの記入や編集をしてもらうことができます。詳細については Colab の概要をご覧ください。新しい Colab ノートブックを作成するには、上にある [ファイル] メニューを使用するか、新しい Colab ノートブックを作成するをクリックします。

まずはじめに演習の準備をしましょう(2)

- 演習用のノートブックを作成しましょう
  - 「ファイル」メニューをクリック

  - 「ノートブックを新規作成」をクリック



Google Colaboratory の使い方

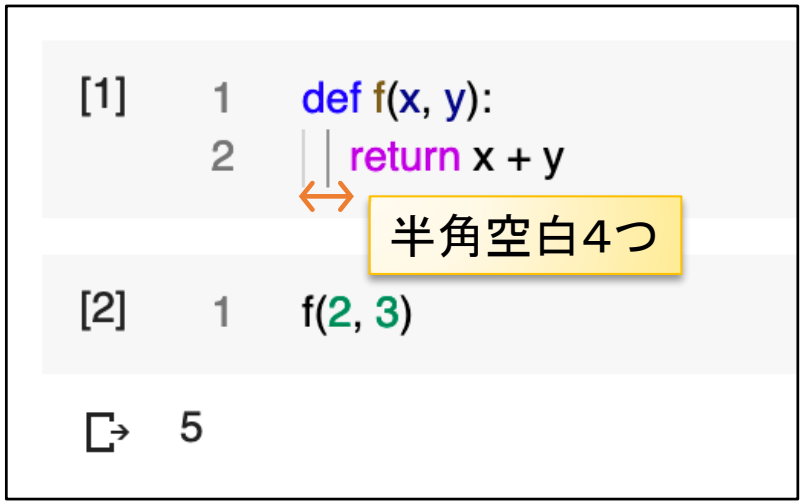
Python とは？

- インタプリタ型（PHP，Ruby等）のプログラミング言語
  - 命令を1つずつ読み込んで実行する方式

    ※コンパイル型言語（C++等）は事前に全命令を読み込んで実行可能な形式に変換する方式
- 機械学習コミュニティでは主流
  - Deep Learningのフレームワークの多くはPythonで実装
- Python の学習方法
  - Python入門
    - https://qiita.com/ponnhide/items/870f0e6b1ae50f201f07
  - Progateの無料会員コース
    - https://prog-8.com/

Python のあいうえお(1)

- 変数:データを記憶する箱(数学の変数のようなもの)
  - 数値, 文字列, クラス, ・・・
  - 変数名はアルファベットと数字の組み合わせが利用可能
    - ただし先頭は必ずアルファベットで, 大/小文字を区別
- 関数:ある一定の処理を機能として再利用可能な形でまとめたもの(数学の関数のようなもの)
  - def 関数名: の形で宣言する
- インデント(半角空白4つ)で処理のまとまりを判断
  - 異なるインデントは異なる処理ブロックとして扱われる



Python のあいうえお(2)

- 配列：複数のデータをまとめて管理するもの
  - 配列内のデータはインデックスを使ってアクセス（例: a[0]）
    - インデックスは 0 から始まる
  - 角括弧 [ ] で囲むことで配列を作れる
    - データはカンマ(,)区切り

```
[37]  1  a=[1, 2, 3, 4, 5]
      2  a[3]

      4
```

- 連想配列：キーと値のペアでデータを管理する配列
  - 波括弧 { } で囲むことで配列を作れる
  - キーと値をコロン(:)で対応付ける
    - 例: "key1" : 3.0
  - データへのアクセスは角括弧にキーを指定する

```
[38]  1  d = {"a": 2, "b": 9}
      2  d["a"]

      2
```

Python のあいうえお(3)

- 繰り返し:ある処理のまとまりを繰り返し実行

```
for 変数 in データの集まり:
    なんらかの処理
```

  - 「データの集まり」の各要素が順に「変数」に代入されながら「なんらかの処理」が実行される
  - 「データの集まり」には配列や連想配列などを指定可能

- 例:1から10までの総和

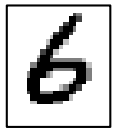
```
[44]  1  sum = 0
      2  for a in [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]:
      3      sum = sum + a
      4  sum

      55
```

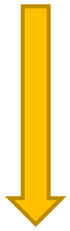
対象：文字認識

- 0～9の手書き数字認識にチャレンジ

学習データ

これは何という数字？





「6」

最近傍法の実装

- 演習用データのダウンロードと展開
- 必要なモジュールの読み込み
- 画像の扱いについて
- 学習画像の準備
- 認識しよう(最近傍法)
- 沢山の画像を認識して認識率を計算しよう

演習用データのダウンロードと展開

- Google Colaboratoryでは，先頭に！をつけることでLinuxのコマンドを実行することが可能
- 以下のコードを入力して実行（Shift+Enter）

```
# 演習用データのダウンロード

!wget https://bit.ly/2PVNqUi -q -O info3.zip
!unzip -q info3.zip
!ls
```

実行結果

演習用のディレクトリに移動

- 演習を実行する際の作業ディレクトリを %cd コマンドを使って「info3」フォルダに変更
- 以下のコードを入力して実行（Shift+Enter）

```
# 作業ディレクトリを変更

%cd info3
!ls
```

実行結果

必要なモジュールを読み込み

- 以下のコードを入力して，Shift+Enterキーを押して実行します

```
# モジュールの読み込み

%matplotlib inline
import numpy as np
from skimage.io import imread, imshow
from skimage.feature import hog
from sklearn.metrics import accuracy_score, confusion_matrix
from scipy.stats import mode
import matplotlib.pyplot as plt
from print_cmx import print_cmx
```

画像の読み込み

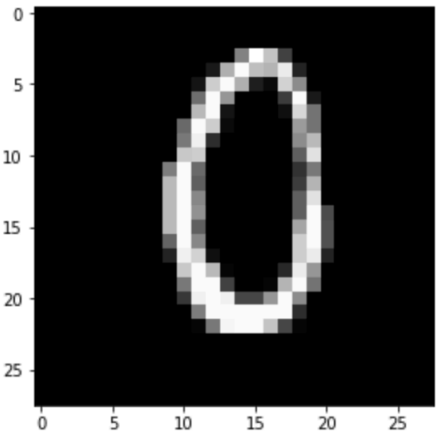
- 画像の読み込み

```
# 画像の読み込み
image = imread("classes/cls0_0.png")

# 表示
imshow(image)
```

実行結果

```
[4]  1  # 画像の読み込み
     2  image = imread("classes/cls0_0.png")
     3
     4  # 表示
     5  imshow(image)
```

```
<matplotlib.image.AxesImage at 0x7f7f04835240>
```



画像からHOG特徴量を抽出

- HOG特徴量の抽出処理

```
# HOG (Histograms of Oriented Gradient) 特徴量を計算
feat, hog_image = hog(image, visualize=True)
```

```
imshow(hog_image)
```

とだけ入力して Shift+Enter で画像が表示される

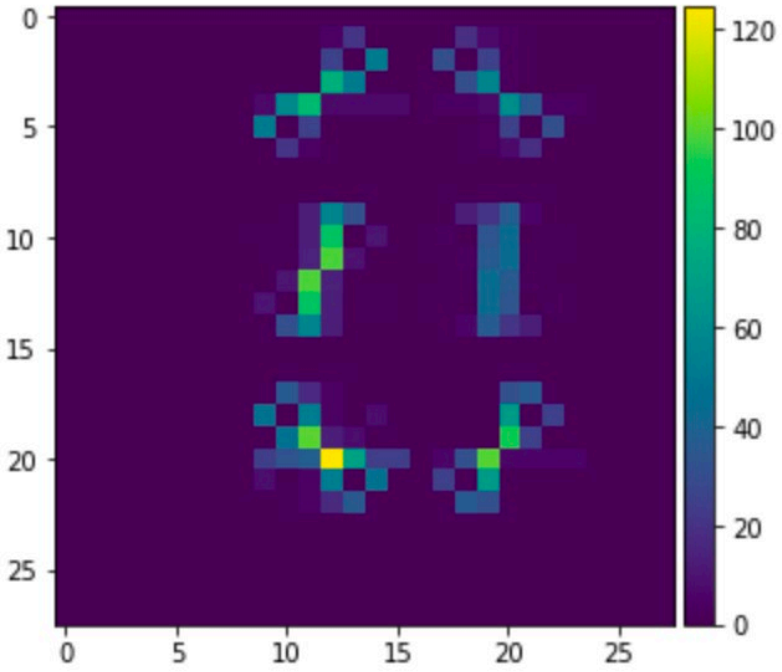
- 特徴量の可視化

```
feat
```

と入力して Shift+Enter を押すと特徴ベクトルの値が表示される

```
imshow(hog_image)
```

でHOG特徴量を画像化したものが表示される



学習画像の準備

- samples_per_class
  - クラスごとの学習画像枚数（後で変更：1～100）

```
samples_per_class = 1

train_images = [] # 入力画像
train_features = [] # 特徴量
train_labels = [] # 対応するクラスラベル

for c in range(10):
    for i in range(samples_per_class):
        img = imread("classes/cls%d_%d.png" % (c, i))
        feat = hog(img)

        train_images.append(img)
        train_features.append(feat)
        train_labels.append(c)
```

学習画像の準備

- samples_per_class
  - クラスごとの学習画像枚数（後で変更：1～100）

```
samples_per_class = 1

train_images = [] # 入力画像
train_features = [] # 特徴量
train_labels = [] # 対応するクラスラベル

for c in range(10):
    for i in range(samples_per_class):
        img = imread("classes/cls%d_%d.png" % (c, i))
        feat = hog(img)

        train_images.append(img)
        train_features.append(feat)
        train_labels.append(c)
```

スペース4個

スペース8個

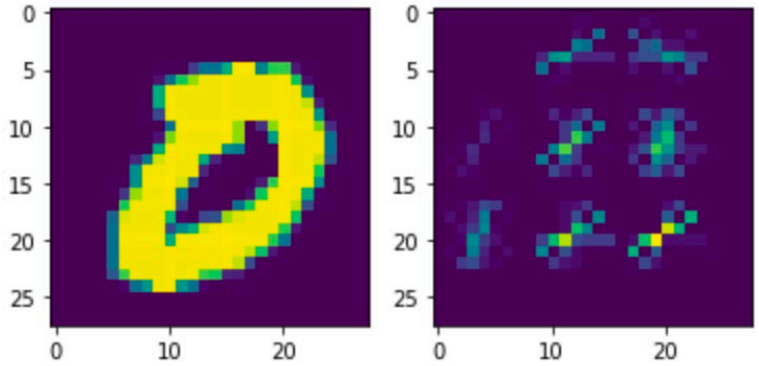
認識したい画像の読み込み・特徴抽出

- imagesフォルダ内の適当な画像を読み込む
  - 000.png～099.png まであります

```
test_image = imread("images/005.png")
test_feature, test_feature_image = hog(test_image, visualize=True)
```

- 読み込んだ画像を表示してみよう

```
plt.subplot(1,2,1)
plt.imshow(test_image)
plt.subplot(1,2,2)
plt.imshow(test_feature_image)
```



※ matplotlibのsubplotを使うと画像やグラフを並べて表示できる

Python Tips #1

- 配列を簡単に初期化したい
  - 通常のやり方

```
y = []
for x in range(10):
    y.append(x)
```

  - 簡単な書き方（実行時間も早くなる）

```
y = [x for x in range(10)]
```

各クラスの代表との距離を計算

```
# train_featuresの中の全ての学習サンプルとの距離をそれぞれ計算する

distances = [np.linalg.norm(feat - test_feature) \
                                    for feat in train_features]
```

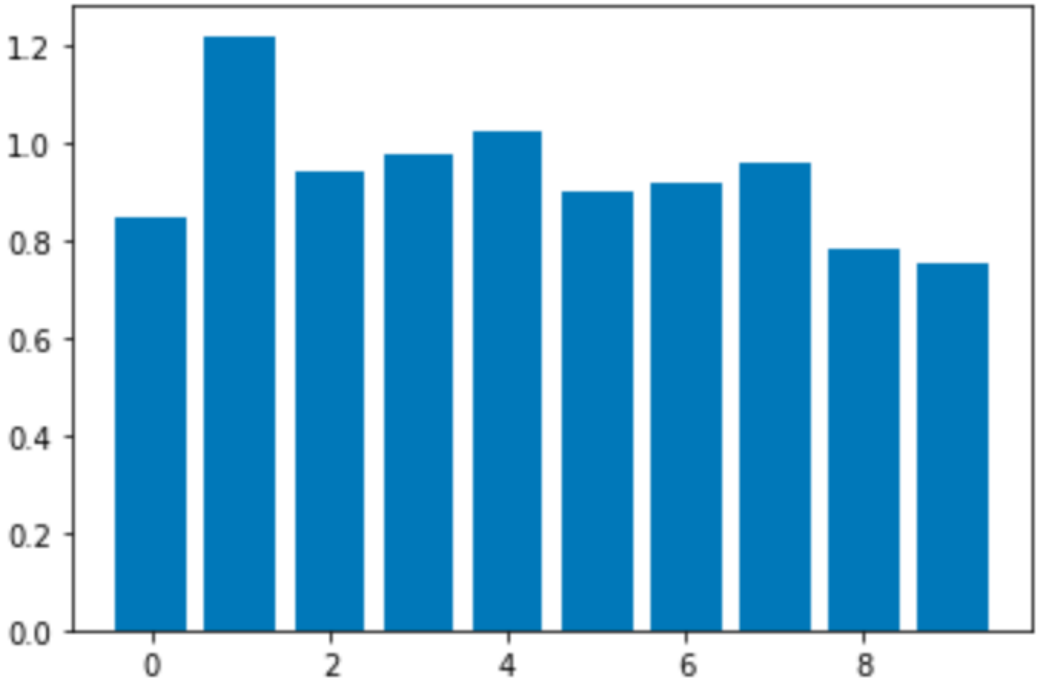
途中で改行する時は \ を入力

```
# 結果を見てみよう

distances
```

各クラスの代表との距離を計算

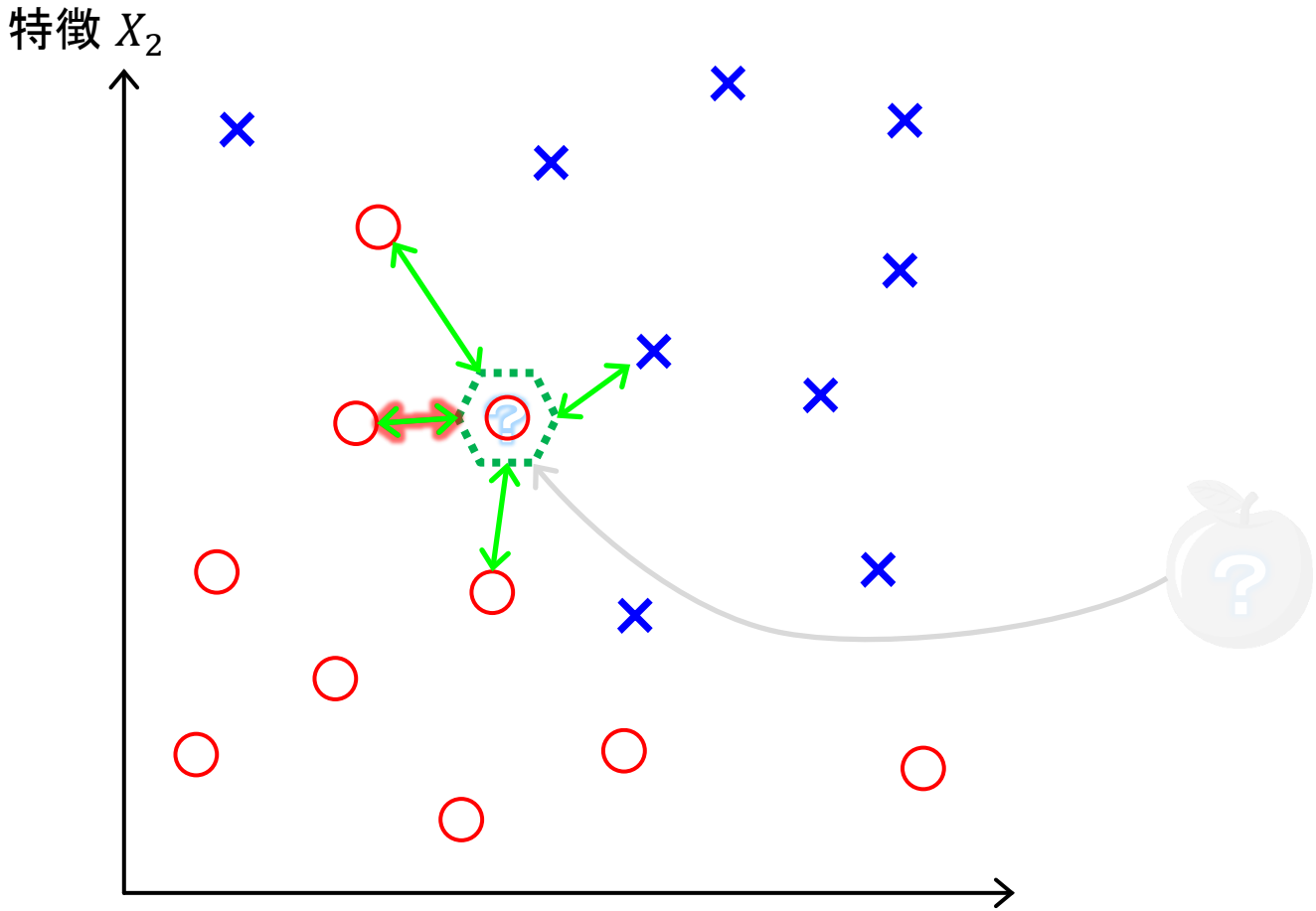
- 距離のばらつきを可視化してみる

```
# matplotlibを用いたグラフ化

plt.bar(range(len(distances)), distances)
```



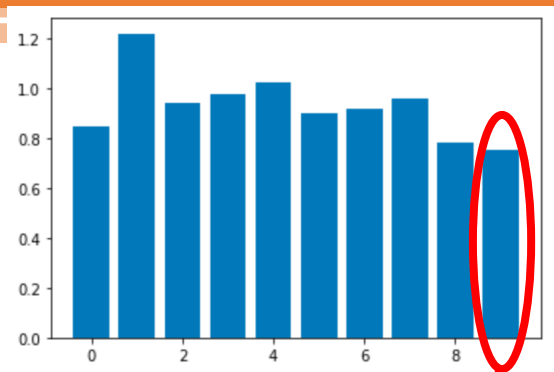
【復習】最近傍法（2クラスの場合）

- 入力データに最も近い教師データを出力する手法



2クラス（○と×）の場合

一番近いデータを求める




```
# 最も近いのは何番目のデータなのかを求める
idx = np.argmin(distances)

# idx 番目の教師データを参照
train_labels[idx]
```

配列内で最も小さい値のインデックスを返す

成功した？

```
# 入力画像の確認
imshow(test_image)
```

認識処理の関数化

```
# 使いやすいよう，認識処理を関数化

def predict(test_feature, train_features, train_labels):
    distances = [np.linalg.norm(feat - test_feature) \
                 for feat in train_features]
    idx = np.argmin(distances)
    return train_labels[idx]
```

```
# 関数を使って認識してみる

predict(test_feature, train_features, train_labels)
```

認識したいデータ

学習データ

学習データのラベル

認識率を求める

- 多数の画像がどのくらい正しく認識できるかを調べる
  - 画像を読み込む

```
# 評価用画像の読み込み関数（100枚）

def load_images():
    images = []
    features = []
    for x in range(100):
        img = imread("images/%03d.png" % x)
        feat = hog(img)
        images.append(img)
        features.append(feat)

    return images, features
```

認識率を求める

- 多数の画像がどのくらい正しく認識できるかを調べる
  - 正解ラベルの読み込み

```
# 正解ラベルの読み込み関数

def load_labels():
    labels = []
    with open("labels.txt", "r") as fin:
        for x in range(100):
            l = int(fin.readline().strip())
            labels.append(l)
    return labels
```

```
# 評価用画像，正解ラベルの読み込み
test_images, test_features = load_images()
test_labels = load_labels()
```

認識率を求める(1)

- 全てのテスト画像を使って認識率を計算

```
# test_features の全ての評価用画像について，認識を行う

results = [predict(feat, train_features, train_labels) \
        for feat in test_features]
```

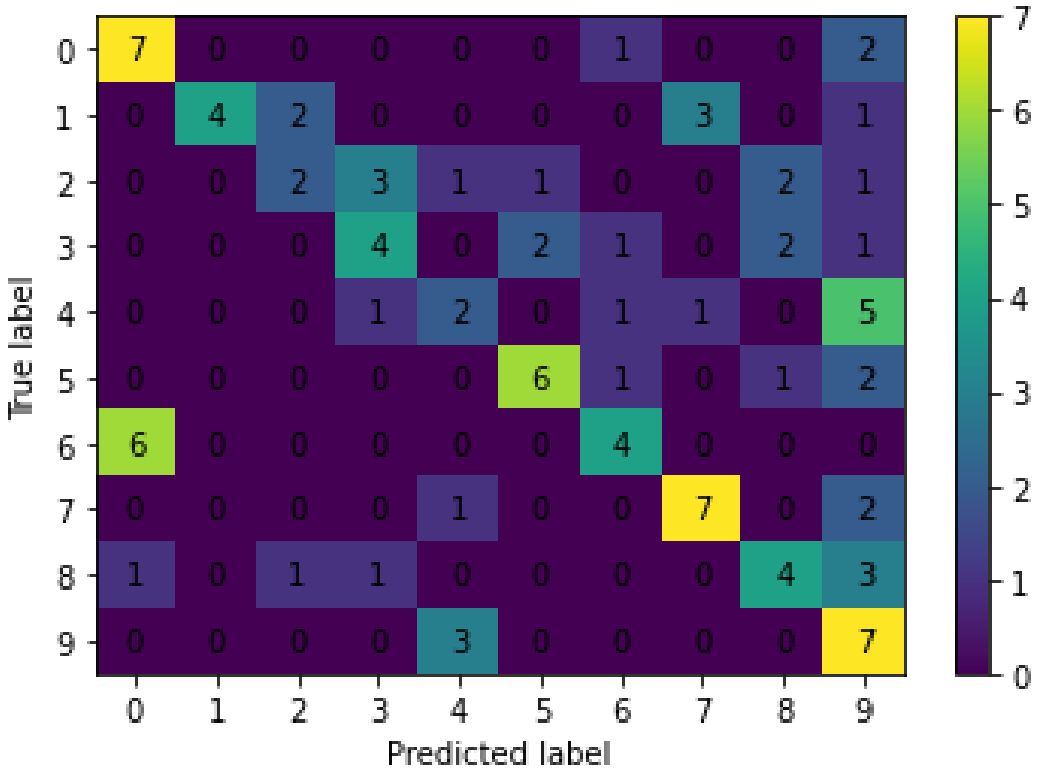
```
# 認識率の計算

accuracy_score(test_labels, results)
```

```
[34]    1   # 認識率の計算
        2
        3   accuracy_score(test_labels, results)

    0.47
```

認識率を求める(2)

- 混同行列による認識誤りの傾向分析

```
# 混同行列を表示（正しく分類したか？どの文字をどの文字に間違えたか？）

print_cmx(test_labels, results)
```



学習データ数を増やすと認識率はどうなる？

- 学習画像の準備のところの samples_per_class の値を増やして確認してみよう
  - 「認識率を求める」のコードの再実行を忘れずに

```
samples_per_class = 100 # ←この部分を 1 から 100 までの値に変更！

train_images = [] # 入力画像
train_features = [] # 特徴量
train_labels = [] # 対応するクラスラベル

for c in range(10):
    for i in range(samples_per_class):
        img = imread("classes/cls%d_%d.png" % (c, i))
        feat = hog(img)

        train_images.append(img)
        train_features.append(feat)
        train_labels.append(c)
```

まとめ

- 機械学習の基礎
  - 最近傍法
  - K最近傍法
  - 線形判別分析法

- 演習（Pythonによる実践機械学習プログラミング）
  - Pythonによる機械学習環境の構築
    - Google Colaboratory の準備
  - 手書き文字認識
    - Nearest Neighbor（最近傍法）