

# Machine Learning with Python

～ 機械学習で、手書き判定アプリを作ろう～

Yohei Munesada



# about me

宗定 洋平  
(むねさだ ようへい)

G's ACADEMY TOKYO メンター

ブログ => <https://www.yoheim.net>



# course catalog

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ
- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践）
- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践）
- ✓ 手書き文字の判定アプリを作ろう（機械学習）
- ✓ デプロイ
- ✓ 開発演習（任意提出）

*Pre study*

*1st day*

*2nd day*



a concept of the day

機械学習の作業ステップを学ぶ

機械学習を Python で実装してみる

演習を通してコーディング力につける

機械学習モデルの中身について



# today's demo

## 手書き文字を判定しよう

(Flaskと機械学習のサンプル)

▼フリーハンドで書いてみる(一筆書き)▼

→

▼判定結果だよー▼

[消す](#)

<https://goo.gl/KMgK3e>



# agenda

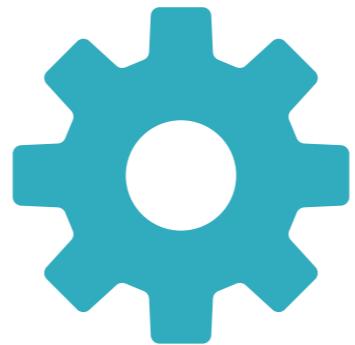
機械学習の概論

機械学習を取り巻く Python ライブラリ

SVM を用いた手書き判定アプリを作る



# Machine Learning Basic

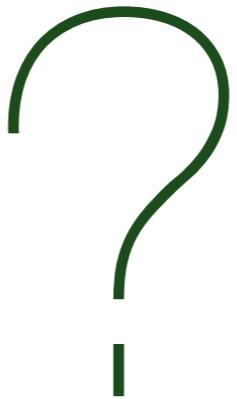


# agenda

- 機械学習とは何か？
- 回帰と分類
- 教師ありと教師なし
- 機械学習の主な4つの種類
- 機械学習のステップ



machine learning is ... ?



# machine learning is ... ?

大量データからパターンを認識する

- 明日が晴れる条件とは？
- イケメンに認定されるパターンとは？
- 明日のディズニーランドが混む条件は？

パターンを使って、未知のデータを予測する

- あの動き、もしや不審者では・・・？
- 今週末のお店への来客数は？
- 来月の電力使用量は？



# regression and classification

機械学習のモデルは、回帰(Regression)と分類(Classification)に大別されます

## 回帰(Regression)

入力されたデータから数値を予測するモデル。

例：ユーザーの購入額予測、ユーザーのスマホ利用時間予測

## 分類(Classification)

入力されたデータから分類を予測するモデル。分類器(Classifier)とも呼ばれる。

例：ユーザーが購入するか否か、画像に猫が含まれるか否か



# supervised and unsupervised

機械学習のモデルは、教師あり(Supervised)と教師なし(Unsupervised)に大別されます

## 教師あり(Supervised)

事前に与えられたデータ(トレーニングデータ)を使って学習を行い、それをもとに予測する。

例：線形回帰、ロジスティック回帰、SVM、ニューラルネット、決定木、etc

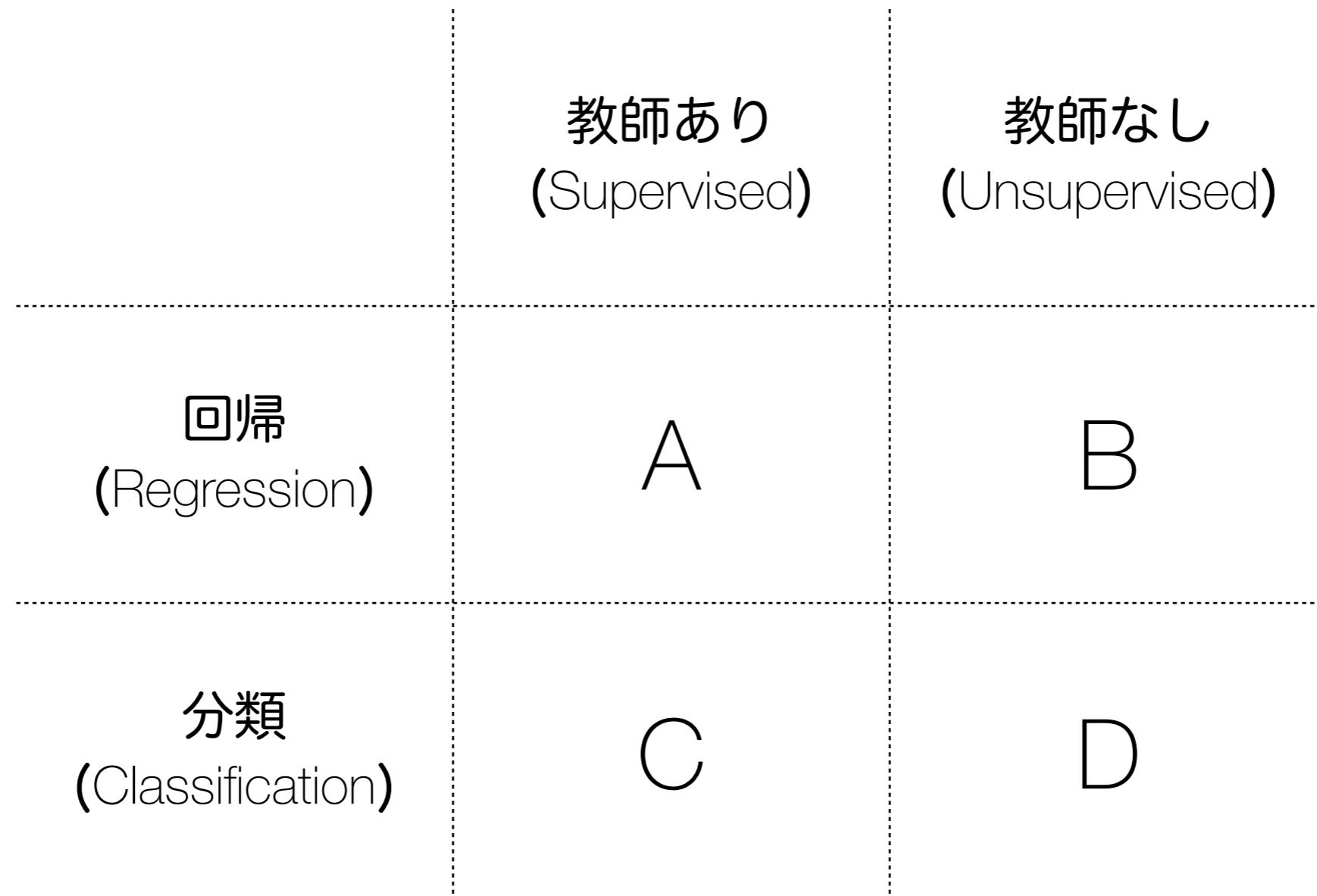
## 教師なし(Unsupervised)

事前データなしに、与えられた未知なデータから何らかの本質的な構造を導き出す。

例：クラスタリング、主成分分析、etc



# 4 types of machine learning



他にも 強化学習 や 遺伝的アルゴリズム も



ref:

## 強化学習(Reinforcement Learning)

[http://yoheim.net/work/01\\_maze/](http://yoheim.net/work/01_maze/)

<https://goo.gl/3gS3fq>

## 遺伝的アルゴリズム(Genetic Algorithm)

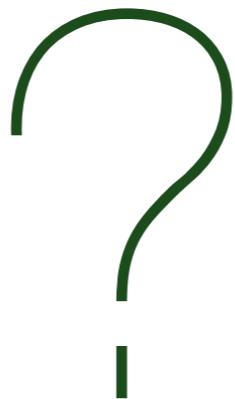
<http://www.nicovideo.jp/watch/sm18721450>

<https://goo.gl/POq5eI>



# steps of machine learning

機会学習は以下のステップで実行します



# steps of machine learning

機会学習は以下のステップで実行します

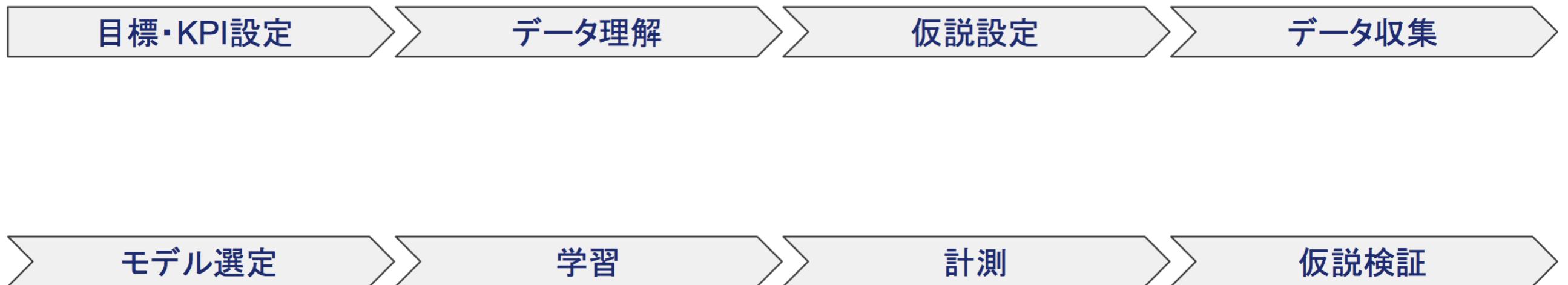
1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する



for your information

## データサイエンティストのワークフロー

### 機械学習編

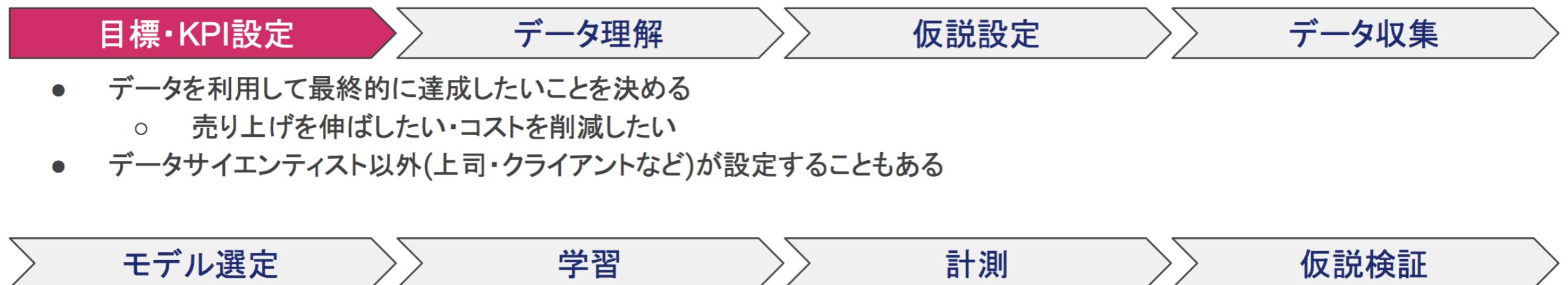


※引用：永田さん講義資料より

for your information

## データサイエンティストのワークフロー

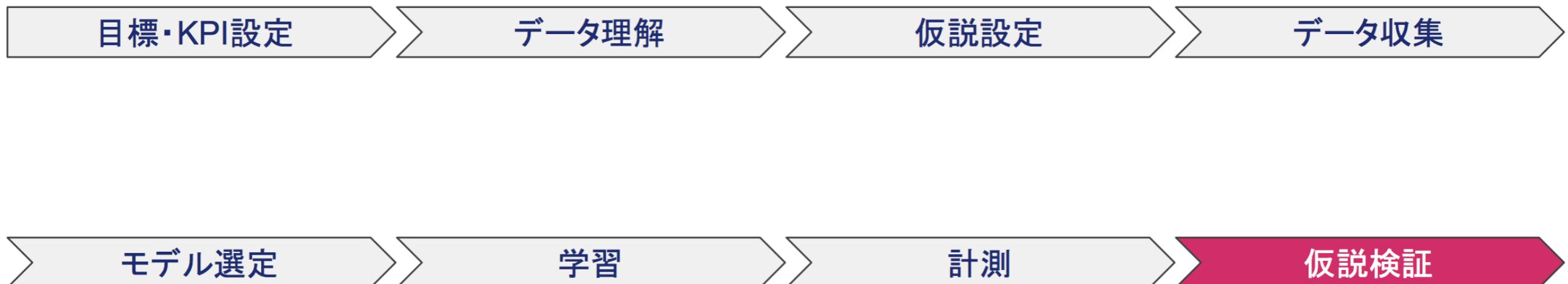
### 機械学習編



for your information

## データサイエンティストのワークフロー

### 機械学習編



- 立てた仮説が実際に目標に貢献しているか確認
  - 正確な売り上げ予測が本当に仕入れ量の最適化・コスト削減に貢献しているか確認

※引用：永田さん講義資料より

# re:agenda

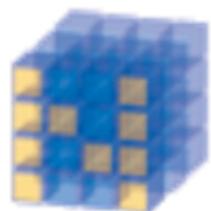
- 機械学習とは何か？
- 回帰と分類
- 教師ありと教師なし
- 機械学習の主な4つの種類
- 機械学習のステップ



python libraries for ML



# main python libraries



**NumPy**  
Base N-dimensional array  
package



**SciPy library**  
Fundamental library for  
scientific computing



**Matplotlib**  
Comprehensive 2D  
Plotting



**IPython**  
Enhanced Interactive  
Console



**Sympy**  
Symbolic mathematics



**pandas**  
Data structures & analysis



# NumPy

行列やベクトルを扱うことができ、C言語レベルで高速に処理する

```
import numpy as np

X = np.array([
    [5, 0, 1],
    [1, 6, 0],
    [0, 0, 1]
])

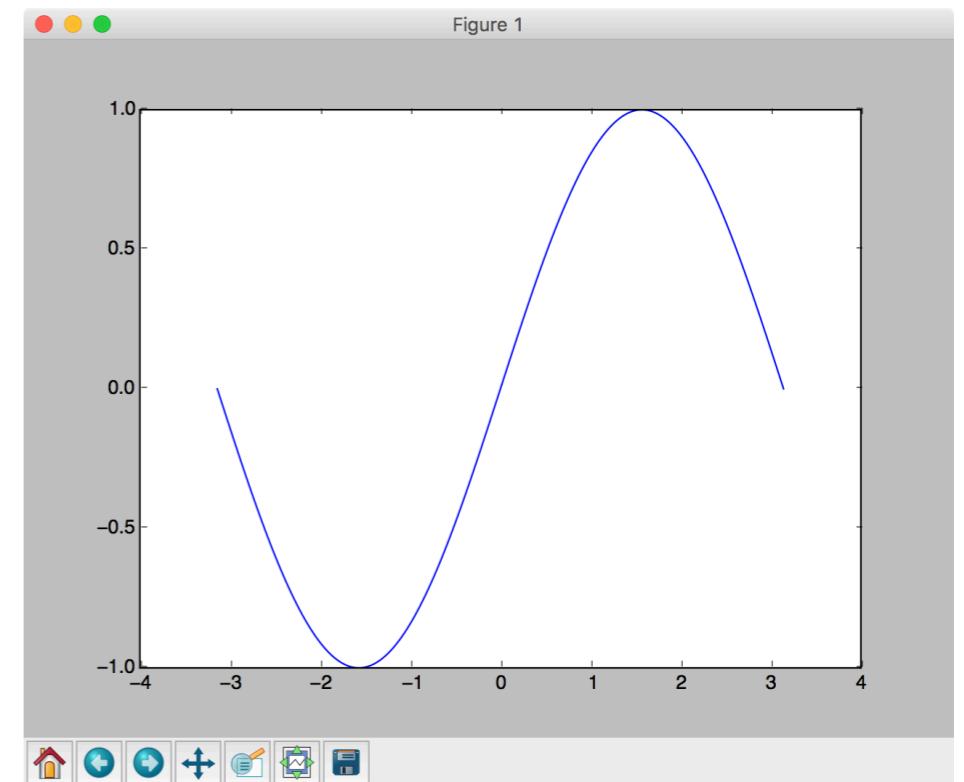
X.shape
X.T

E = np.eye(X.shape[0])
X.dot(E)
```

# Matplotlib

2Dプロットを行うライブラリです

```
import numpy as np  
import matplotlib.pyplot as plt  
  
x = np.linspace(-np.pi, np.pi, 200)  
plt.plot(x, np.sin(x))  
plt.show()
```



The image shows the official scikit-learn website. At the top left is the logo. To its right are navigation links: Home, Installation, Documentation (with a dropdown arrow), Examples, Google Custom Search, and a Search button. A 'Fork me on GitHub' button is in the top right corner. Below the header is a large blue banner. On the left side of the banner is a 3x3 grid of plots showing various machine learning models (Input data, Nearest Neighbors, Linear SVM, RBF SVM, Gaussian Process, Decision Tree, Random Forest, Neural Net) applied to different datasets. The main title 'scikit-learn' is in large white font, followed by 'Machine Learning in Python' in a smaller font. To the right of the title is a bulleted list of features:

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

## Classification

Identifying to which category an object belongs to.

**Applications:** Spam detection, Image recognition.

**Algorithms:** SVM, nearest neighbors, random forest, ...

— Examples

## Regression

Predicting a continuous-valued attribute associated with an object.

**Applications:** Drug response, Stock prices.

**Algorithms:** SVR, ridge regression, Lasso, ...

— Examples

## Clustering

Automatic grouping of similar objects into sets.

**Applications:** Customer segmentation, Grouping experiment outcomes

**Algorithms:** k-Means, spectral clustering, mean-shift, ...

— Examples

## Dimensionality reduction

Reducing the number of random variables to consider.

**Applications:** Visualization, Increased efficiency

**Algorithms:** PCA, feature selection, non-

## Model selection

Comparing, validating and choosing parameters and models.

**Goal:** Improved accuracy via parameter tuning

**Modules:** grid search, cross validation,

## Preprocessing

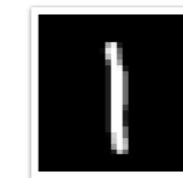
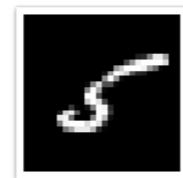
Feature extraction and normalization.

**Application:** Transforming input data such as text for use with machine learning algorithms.

**Modules:** preprocessing, feature extraction.

— Examples

# hand-written digits recognition



# agenda

1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する
8. 学習結果を保存する
9. Webサーバーから利用する



<http://yann.lecun.com/exdb/mnist/>

# THE MNIST DATABASE

## of handwritten digits

[Yann LeCun](#), Courant Institute, NYU

[Corinna Cortes](#), Google Labs, New York

[Christopher J.C. Burges](#), Microsoft Research, Redmond

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image.

It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

Four files are available on this site:

[train-images-idx3-ubyte.gz](#): training set images (9912422 bytes)

[train-labels-idx1-ubyte.gz](#): training set labels (28881 bytes)

[t10k-images-idx3-ubyte.gz](#): test set images (1648877 bytes)

[t10k-labels-idx1-ubyte.gz](#): test set labels (4542 bytes)

**please note that your browser may uncompress these files without telling you.** If the files you downloaded have a larger size than the above, they have been uncompressed by your browser. Simply rename them to remove the .gz extension. Some people have asked me "my application can't open your image files". These files are not in any standard image format. You have to write your own (very simple) program to read them. The file format is described at the bottom of this page.

# samples



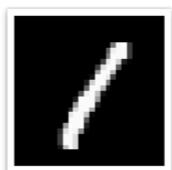
0.pgm



1.pgm



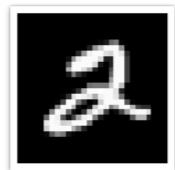
2.pgm



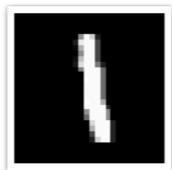
3.pgm



4.pgm



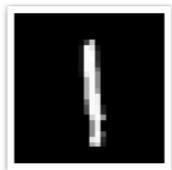
5.pgm



6.pgm



7.pgm



8.pgm



9.pgm



10.pgm



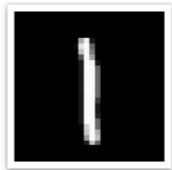
11.pgm



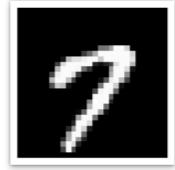
12.pgm



13.pgm



14.pgm



15.pgm



16.pgm



17.pgm



18.pgm



19.pgm



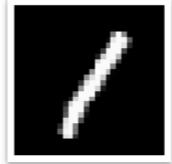
20.pgm



21.pgm



22.pgm



23.pgm



24.pgm

# download MNIST data

MNIST データは Web から取得することができます

[train-images-idx3-ubyte.gz](#): training set images (9912422 bytes)

[train-labels-idx1-ubyte.gz](#): training set labels (28881 bytes)

[t10k-images-idx3-ubyte.gz](#): test set images (1648877 bytes)

[t10k-labels-idx1-ubyte.gz](#): test set labels (4542 bytes)

<http://yann.lecun.com/exdb/mnist/>



# original format

## TRAINING SET LABEL FILE (train-labels-idx1-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.

## TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel

# agenda

1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する
8. 学習結果を保存する
9. Webサーバーから利用する

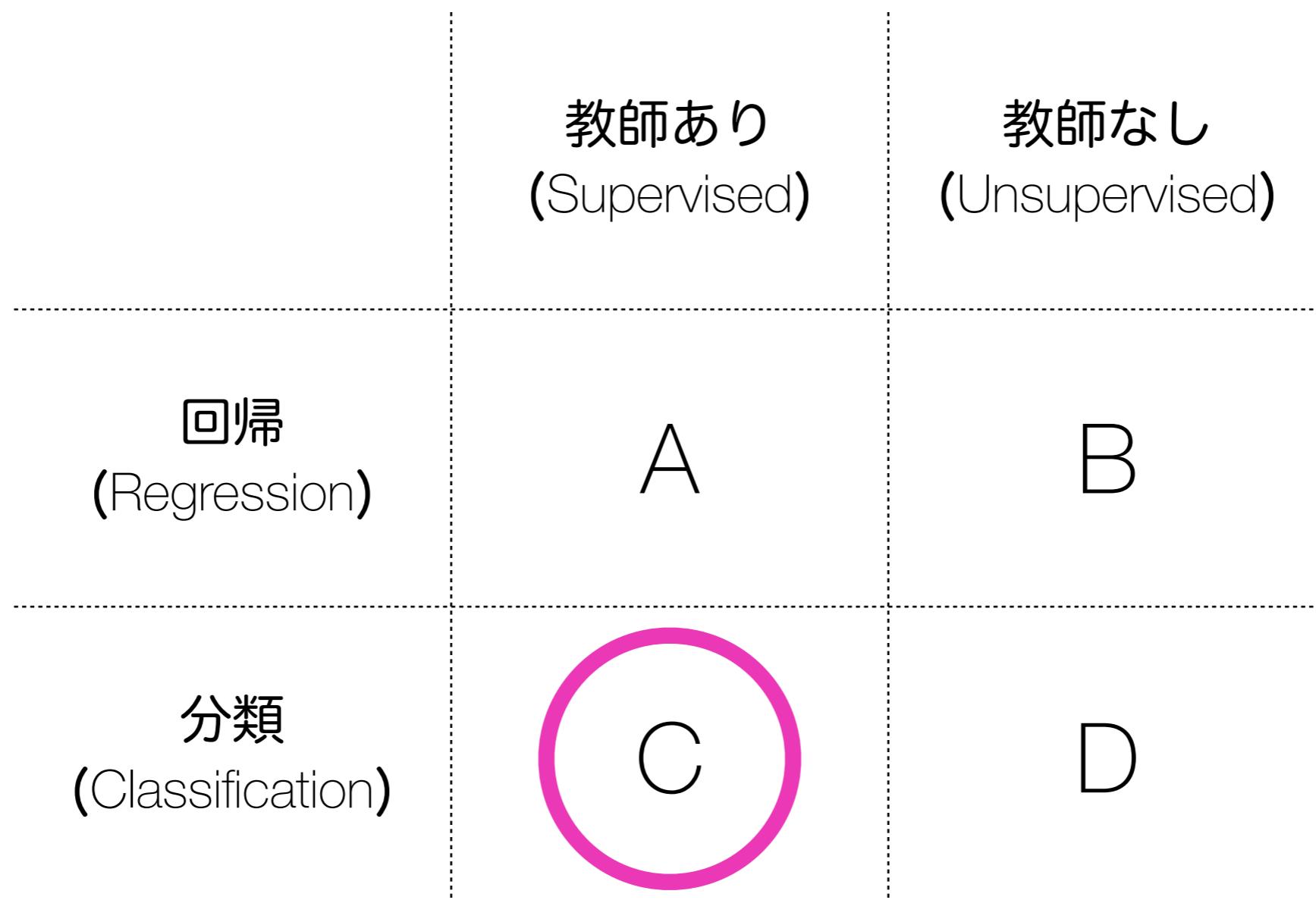


# re: 4 types of machine learning

	教師あり (Supervised)	教師なし (Unsupervised)
回帰 (Regression)	A	B
分類 (Classification)	C	D

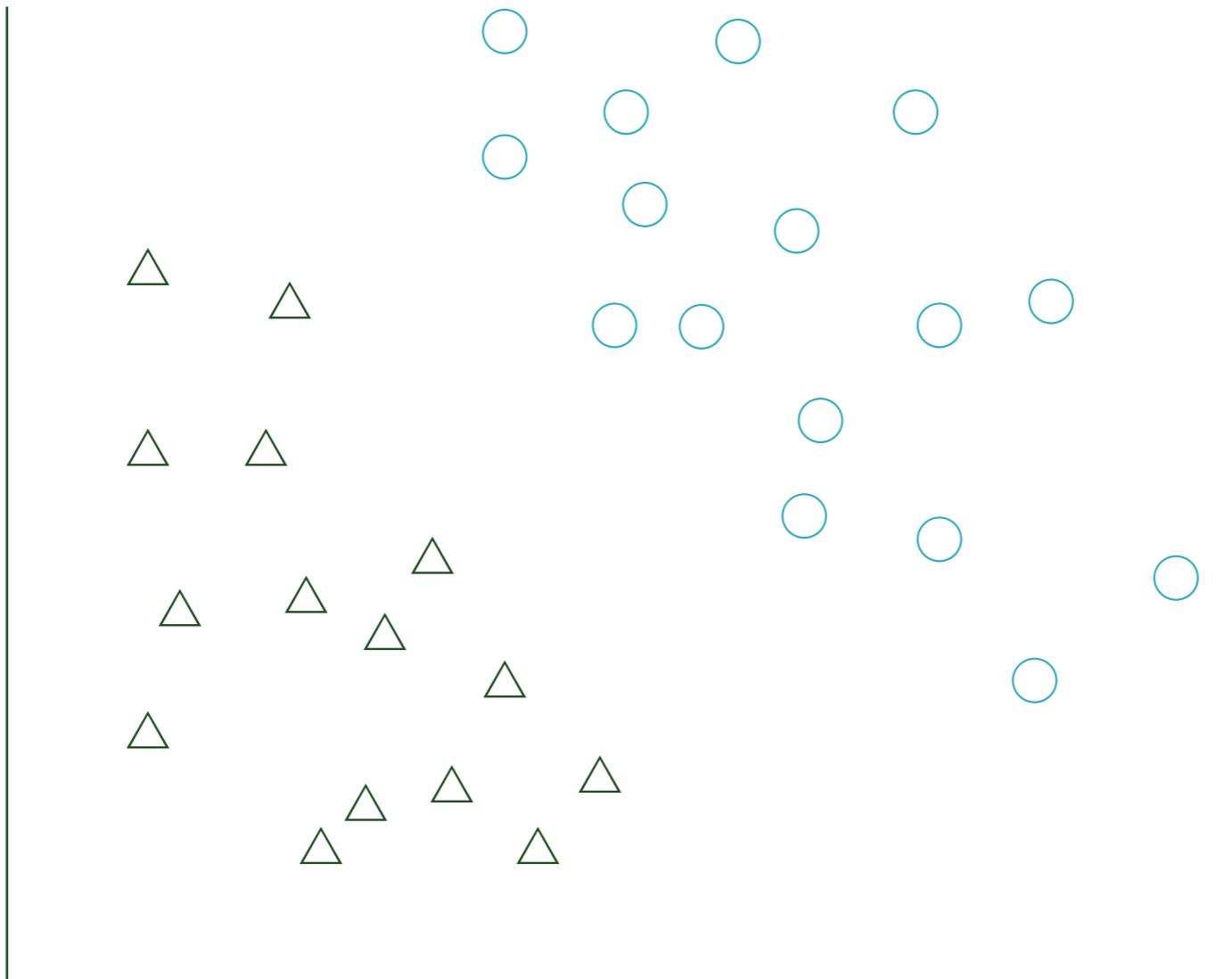


# re: 4 types of machine learning



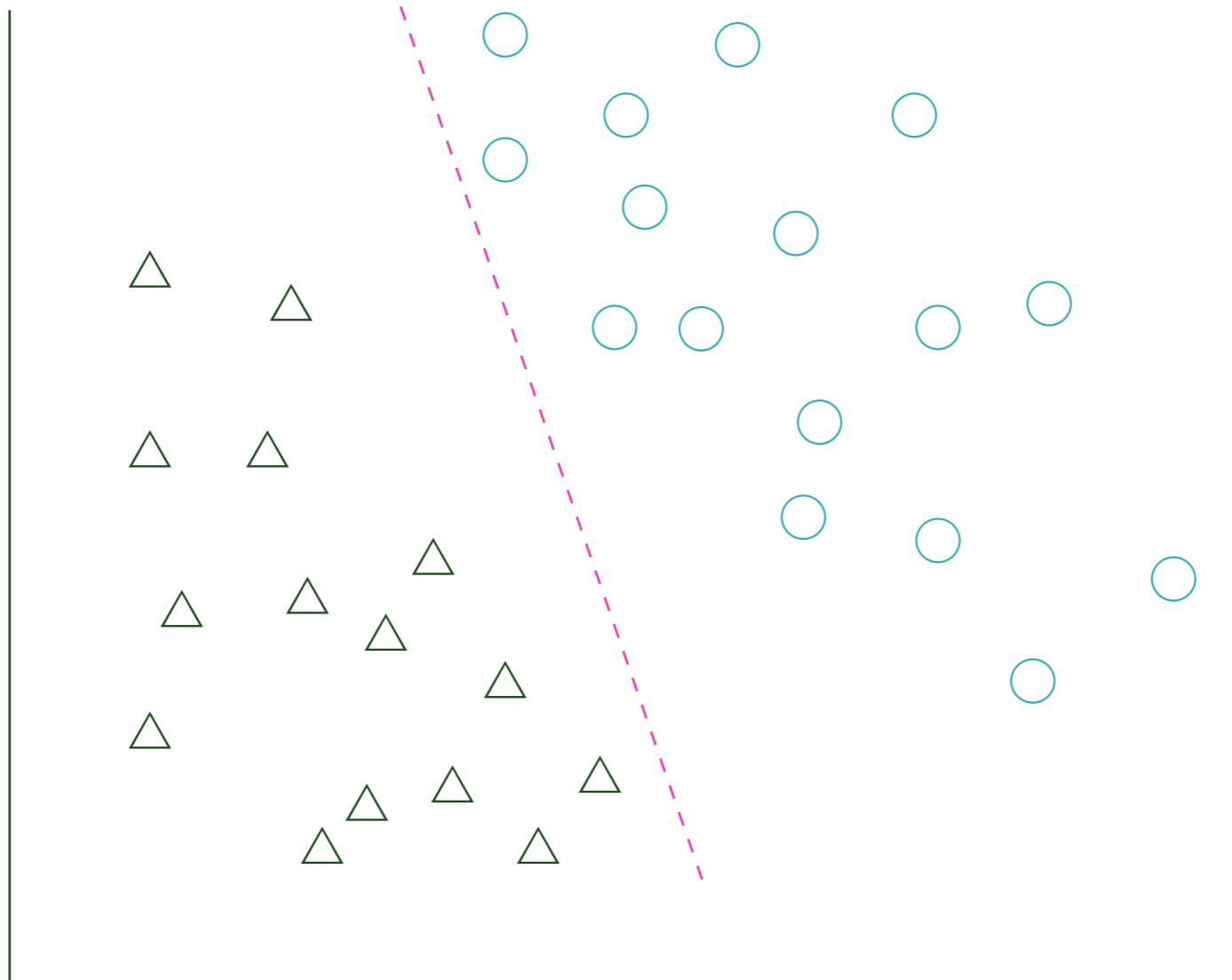
# Support Vector Machine

教師あり、分類問題を解くアルゴリズムです



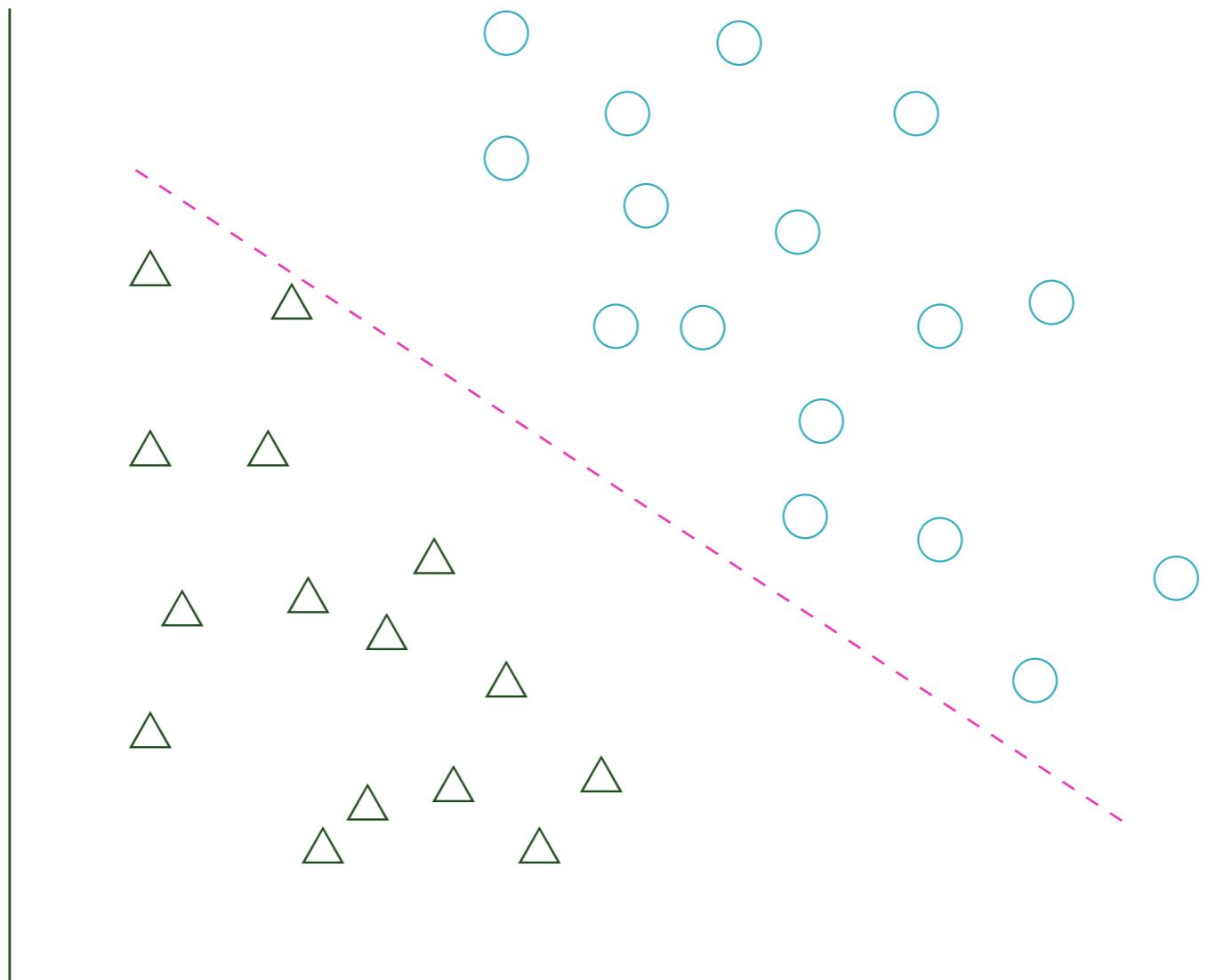
# Support Vector Machine

教師あり、分類問題を解くアルゴリズムです



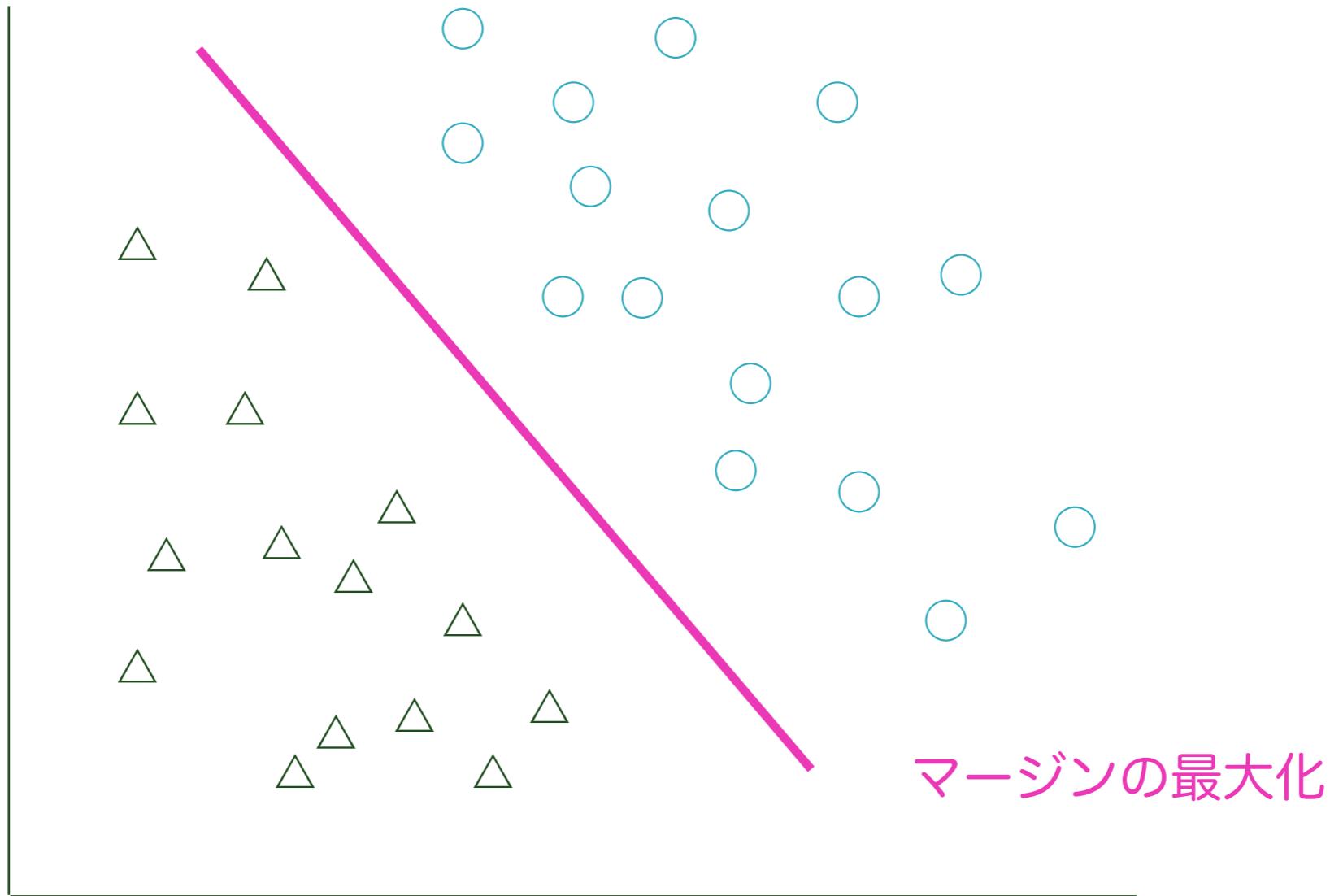
# Support Vector Machine

教師あり、分類問題を解くアルゴリズムです



# Support Vector Machine

教師あり、分類問題を解くアルゴリズムです



# Wanna dive into deep ?

機械学習の基礎（数式編）

<https://goo.gl/JE8VwX>

サポートベクターマシン

<https://goo.gl/ATGxjM>



# agenda

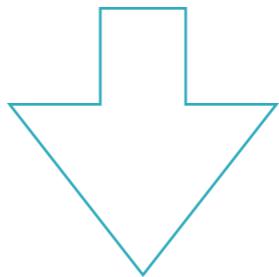
1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する
8. 学習結果を保存する
9. Webサーバーから利用する



# feature is

機械学習のアルゴリズムに入力する特徴量(=feature)です

モテ率 = ベース値 + 身長 + ルックス + 性格  
(判定結果) (特徴1) (特徴2) (特徴3)



$$h_{\theta}(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$$



# the features of MNIST

MNISTの画像の特徴量とは何でしょうか？



0,0,0,3,18,18,18,126,136,175,26,166,255,247,127,0,0,0,0,0,  
,0,0,0,0,0,30,36,94,154,170,253,253,253,253,253,225,172,  
253,242,195,64,0,0,0,0,0,0,0,0,0,0,49,238,253,253,253,253  
,253,253,253,253,251,93,82,82,56,39,0,0,0,0,0,0,0,0,0,0,0,0,1  
8,219,253,253,253,253,253,253,198,182,247,241,0,0,0,0,0,0,0,  
0,0,0,0,0,0,0,0,80,156,107,253,253,205,11,0,43,154,0,0,0,  
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,14,1,154,253,90,0,0,0,0,0,0,0,  
...

1ピクセルがそれぞれ特徴量（784コ）



# agenda

1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する
8. 学習結果を保存する
9. Webサーバーから利用する



# data pre-processing

MNIST データから、モデルに読み込ませるデータを作ります

1. CSV形式に変換する
2. サンプル画像を出力してみる (加工結果の確認)



# convert to csv format

MNIST のバイナリーデータをCSV形式に変換します

が、その前に…



# binary data with Python

バイナリデータを struct モジュール(標準ライブラリ)で扱います

## 目次

- 7.1. struct — バイト列をパックされたバイナリデータとして解釈する
  - 7.1.1. 関数と例外
  - 7.1.2. 書式文字列
    - 7.1.2.1. バイトオーダ、サイズ、アライメント
    - 7.1.2.2. 書式指定文字
    - 7.1.2.3. 使用例
  - 7.1.3. クラス

## 前のトピックへ

7. バイナリデータ処理

## 次のトピックへ

7.2. codecs — codec レジストリと基底クラス

## このページ

バグ報告  
ソースの表示

## 7.1. struct — バイト列をパックされたバイナリデータとして解釈する

ソースコード: [Lib/struct.py](#)

このモジュールは、Python の値と Python `bytes` オブジェクトとして表される C の構造体データとの間の変換を実現します。このモジュールは特に、ファイルに保存されたり、ネットワーク接続を経由したバイナリデータを扱うときに使われます。このモジュールでは、C 構造体のレイアウトおよび Python の値との間で行いたい変換をコンパクトに表現するために、[書式文字列](#) を使います。

**注釈:** デフォルトでは、与えられた C の構造体をパックする際に、関連する C データ型を適切にアラインメント (alignment)するために数バイトのパディングを行うことがあります。この挙動が選択されたのは、パックされた構造体のバイト表現を対応する C 構造体のメモリレイアウトに正確に対応させるためです。プラットフォーム独立のデータフォーマットを扱ったり、隠れたパディングを排除したりするには、サイズ及びアラインメントとして `native` の代わりに `standard` を使うようにします: 詳しくは [バイトオーダ、サイズ、アラインメント](#) を参照して下さい。

いくつかの `struct` の関数 (および `Struct` のメソッド) は `buffer` 引数を取ります。これは [バッファプロトコル \(buffer Protocol\)](#) を実装していて読み取り可能または読み書き可能なバッファを提供するオブジェクトのことです。この目的のために使われる最も一般的な型は `bytes` と `bytearray` ですが、バイトの配列とみなすことができるような他の多くの型がバッファプロトコルを実装しています。そのため、それらは `bytes` オブジェクトから追加のコピーなしで読み出しや書き込みができます。

<https://docs.python.org/ja/3/library/struct.html>

# binary data with Python

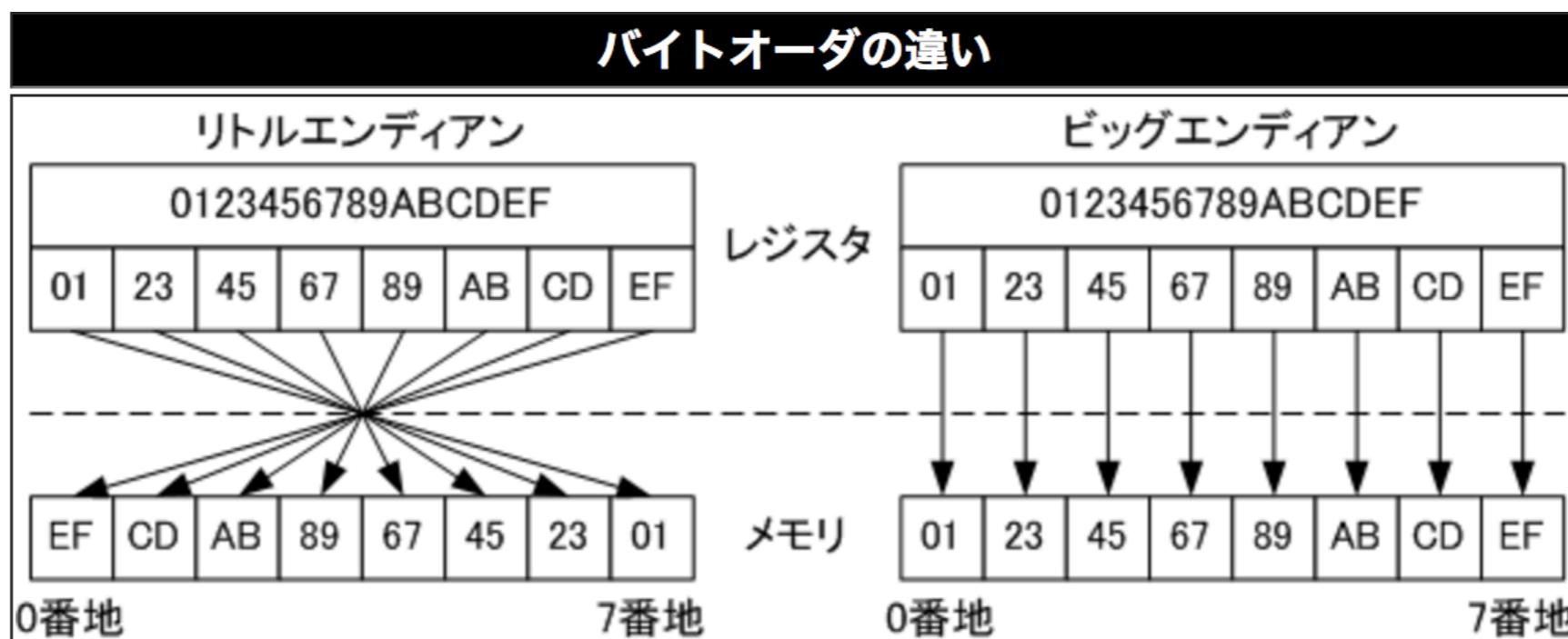
バイナリデータを struct モジュール(標準ライブラリ)で扱います

```
import struct  
f = open("file.binary", "rb")  
num1, num2 = struct.unpack(">II", f.read(8))
```

Format	C Type	Python type	Standard size	Notes
x	pad byte	no value		
c	char	bytes of length 1	1	
b	signed char	integer	1	(1),(3)
B	unsigned char	integer	1	(3)
?	_Bool	bool	1	(1)
h	short	integer	2	(3)
H	unsigned short	integer	2	(3)
i	int	integer	4	(3)
I	unsigned int	integer	4	(3)
L	long	integer	4	(3)
l	unsigned long	integer	4	(3)

# big-endian and little-endian

1 00



(参考) エンディアンについて：<http://www.ertl.jp/~takayuki/readings/info/no05.html>

# convert to csv format

MNIST のバイナリーデータをCSV形式に変換します



※ ラベルデータ

## TRAINING SET LABEL FILE (train-labels-idx1-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000801(2049)	magic number (MSB first)
0004	32 bit integer	60000	number of items
0008	unsigned byte	??	label
0009	unsigned byte	??	label
.....			
xxxx	unsigned byte	??	label

The labels values are 0 to 9.



## ※ ラベルデータ

```
import struct
import gzip

# Read MNIST `label`.
fpath = "./data_mnist/train-labels-idx1-ubyte.gz"
with gzip.open(fpath, "rb") as f:
    magic_number, img_count = struct.unpack(">II", f.read(8))
    labels = []
    for i in range(img_count):
        label = str(struct.unpack("B", f.read(1))[0])
        labels.append(label)

# Write as csv.
outpath = './csv/train-labels.csv'
with open(outpath, "w") as f:
    f.write("\n".join(labels))
```



※ 画像データ

## TRAINING SET IMAGE FILE (train-images-idx3-ubyte):

[offset]	[type]	[value]	[description]
0000	32 bit integer	0x00000803(2051)	magic number
0004	32 bit integer	60000	number of images
0008	32 bit integer	28	number of rows
0012	32 bit integer	28	number of columns
0016	unsigned byte	??	pixel
0017	unsigned byte	??	pixel
.....			
xxxx	unsigned byte	??	pixel



## ※ 画像データ

```
import struct
import gzip

# Read MNIST `images`.
fpath = "./data_mnist/train-images-idx3-ubyte.gz"
with gzip.open(fpath, "rb") as f:
    _, img_count = struct.unpack(">II", f.read(8))
    rows, cols = struct.unpack(">II", f.read(8))
    images = []
    for i in range(img_count):
        binary = f.read(rows * cols)
        images.append(",".join([str(b) for b in binary]))

# Write as csv.
outpath = './csv/train-images.csv'
with open(outpath, "w") as f:
    f.write("\n".join(images))
```



# output as images

一部を画像に出力してCSV結果が正しいことを確認します

```
with open("./csv/train-images.csv") as f:  
    images = f.read().split("\n")  
  
for i, image in enumerate(images[:10]):  
    with open("./image/%d.pgm" % i, "w") as f:  
        s = "P2 28 28 255\n"  
        s += " ".join(image.split(","))  
        f.write(s)
```



# output as images

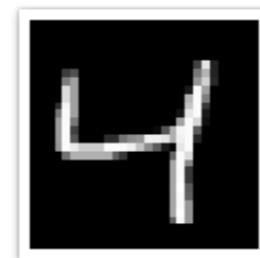
一部を画像に出力してCSV結果が正しいことを確認します



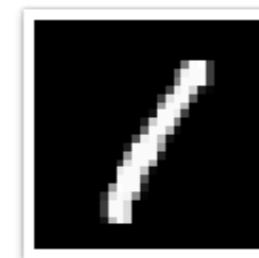
0.pgm



1.pgm



2.pgm



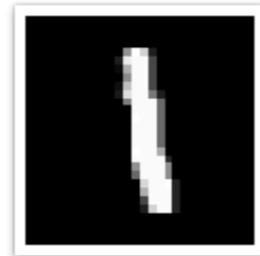
3.pgm



4.pgm



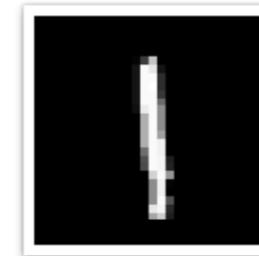
5.pgm



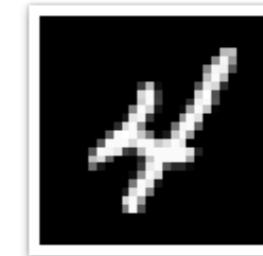
6.pgm



7.pgm



8.pgm



9.pgm



# agenda

1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する
8. 学習結果を保存する
9. Webサーバーから利用する



# use Support Vector Machine

それではSVMでパターン認識を行いたいと思います

```
from sklearn import svm

# Load training data.

with open("./csv/train-images.csv") as f:
    images = f.read().split("\n")[:500]
with open("./csv/train-labels.csv") as f:
    labels = f.read().split("\n")[:500]

# Convert data.

images = [[int(i)/256 for i in image.split(",")] for image in images]
labels = [int(l) for l in labels]

# Use SVM.

clf = svm.SVC()
clf.fit(images, labels)
```



# agenda

1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する
8. 学習結果を保存する
9. Webサーバーから利用する



# evaluate the model

テストデータを用いて学習した結果から予測を行い、精度を評価します

```
from sklearn import metrics

test_images = #読み込み処理は省略（今回は500件読み込む）#
test_labels = #読み込み処理は省略（今回は500件読み込む）#

# Predict.
predict = clf.predict(test_images)

# Show results.
ac_score = metrics.accuracy_score(test_labels, predict)
print("Accuracy:", ac_score)
cl_report = metrics.classification_report(test_labels, predict)
print(cl_report)
```



# evaluate the model

テストデータを用いて学習した結果から予測を行い、精度を評価します

precision	recall	f1-score	support
0.96	0.96	0.96	50
0.72	0.95	0.82	66
1.00	0.75	0.86	52
0.81	0.88	0.85	50
0.89	0.90	0.90	52
0.89	0.64	0.75	39
0.95	0.89	0.92	45
0.88	0.71	0.79	52
1.00	0.72	0.84	39
0.66	0.93	0.77	55
avg / total	0.87	0.84	500

# agenda

1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する
8. 学習結果を保存する
9. Webサーバーから利用する



# how to improve the model

## どうやって改善しますか？

Accuracy: 0.844

	precision	recall	f1-score	support
0	0.96	0.96	0.96	50
1	0.72	0.95	0.82	66
2	1.00	0.75	0.86	52
3	0.81	0.88	0.85	50
4	0.89	0.90	0.90	52
5	0.89	0.64	0.75	39
6	0.95	0.89	0.92	45
7	0.88	0.71	0.79	52
8	1.00	0.72	0.84	39
9	0.66	0.93	0.77	55
avg / total	0.87	0.84	0.84	500



# how to improve the model

トレーニングデータを増やす

アルゴリズムを変更する

特徴を増やす/減らす

正則化項の影響度 ( $\lambda$ ) を増やす/減らす

データの正規化

など



# how to improve the model

トレーニングデータを増やす

アルゴリズムを変更する

特徴を増やす/減らす

正則化項の影響度 ( $\lambda$ ) を増やす/減らす

データの正規化

など



# how to improve the model

学習データを 500 → 5,000 件に増やします

```
from sklearn import svm

# Load training data.

with open("./csv/train-images.csv") as f:
    images = f.read().split("\n")[:5000]
with open("./csv/train-labels.csv") as f:
    labels = f.read().split("\n")[:5000]

# Convert data.

images = [[int(i)/256 for i in image.split(",")] for image in images]
labels = [int(l) for l in labels]

# Use SVM.

clf = svm.SVC()
clf.fit(images, labels)
```



# evaluate the training result

学習データを 500 → 5,000 件に増やすと、精度が 0.84 → 0.93 に改善しました

Accuracy: 0.932

	precision	recall	f1-score	support
0	0.94	1.00	0.97	50
1	0.94	0.95	0.95	66
2	1.00	0.85	0.92	52
3	0.96	0.90	0.93	50
4	0.90	1.00	0.95	52
5	0.86	0.92	0.89	39
6	0.96	0.98	0.97	45
7	0.86	0.94	0.90	52
8	0.97	0.90	0.93	39
9	0.96	0.87	0.91	55
avg / total	0.94	0.93	0.93	500

# agenda

1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する
8. 学習結果を保存する
9. Webサーバーから利用する



# save the training result

学習結果を保存し、のちにWebサーバーから利用できるようにします

```
# Save the training result.  
import joblib  
joblib.dump(clf, "./result/svm.pkl")
```



# agenda

1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する
8. 学習結果を保存する
9. Webサーバーから利用する



# use the model

先ほど保存した学習結果を、別のプログラムから利用します

```
import joblib

# Load the training result.
pklfile = "./result/svm.pkl"
clf = joblib.load(pklfile)

# Predict.
predict = clf.predict([data])
number = str(predict.tolist()[0])
```



# re:agenda

1. データを入手して理解する
2. アルゴリズムを決める
3. 特徴量を決める
4. データのクレンジング/整形をする
5. 学習する
6. モデルを評価する
7. モデルを改善する
8. 学習結果を保存する
9. Webサーバーから利用する



# 2nd demo app

手書き数値判定アプリを作ろう



# today's demo

## 手書き文字を判定しよう

(Flaskと機械学習のサンプル)

▼フリーハンドで書いてみる(一筆書き)▼

→

▼判定結果だよー▼

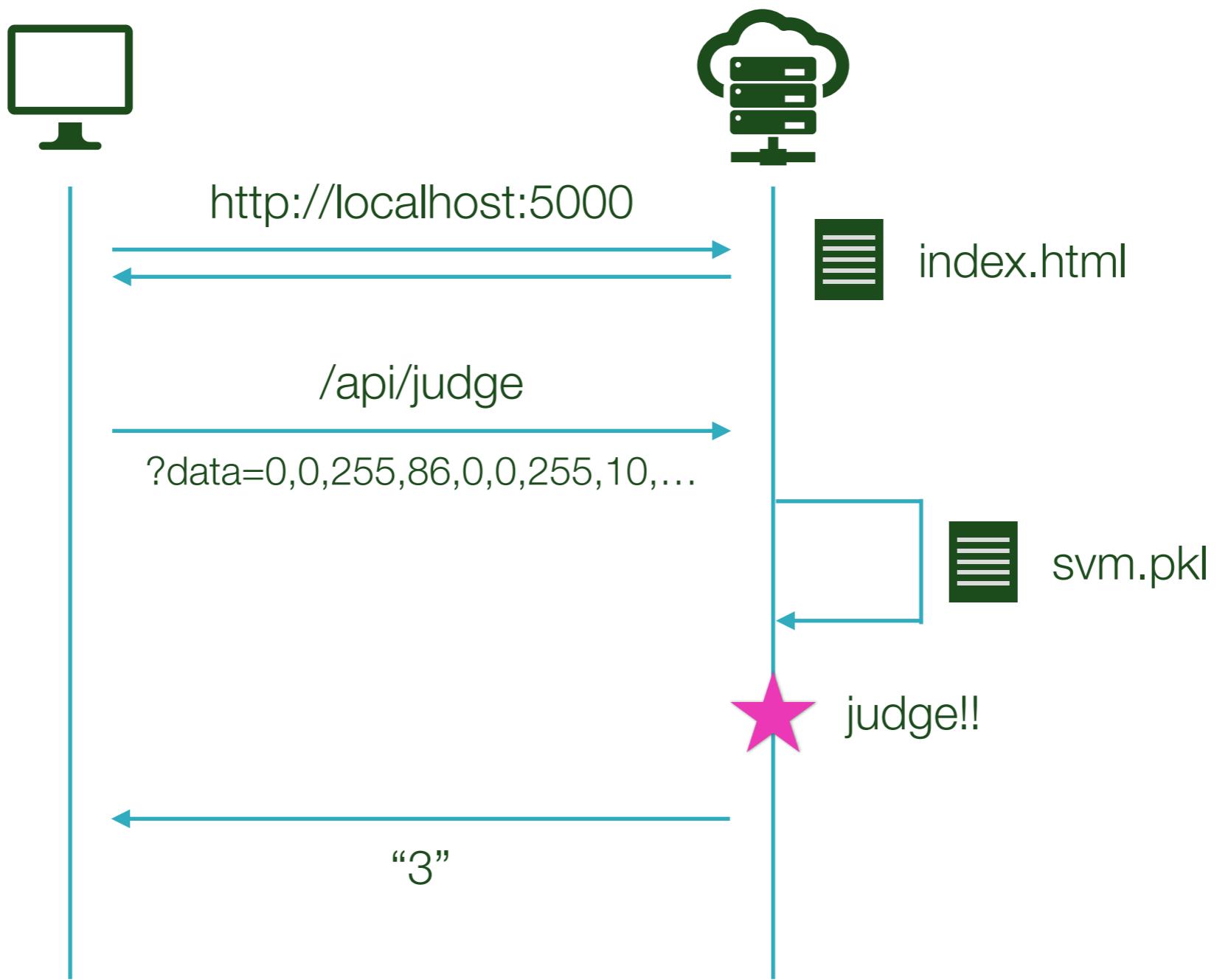
[消す](#)

<https://goo.gl/KMgK3e>



# how it works

Flask アプリケーションから保存した学習結果を利用します



# how to create

1. レポジトリをクローンする
2. ライブラリー一覧を読み込む
3. 起動してみる
4. 基礎課題を実装する
5. 動作テストをする

reference



<https://www.coursera.org/learn/machine-learning>

- Overview
- Syllabus
- FAQs
- Creators
- Ratings and Reviews

## Machine Learning

**Enroll Now**  
Started Feb 20

Financial Aid is available for learners who cannot afford the fee. [Learn more and apply.](#)

Home > Data Science > Machine Learning

# Machine Learning

**About this course:** Machine learning is the science of getting computers to act without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome. Machine learning is so pervasive today that you probably use it dozens of times a day without knowing it. Many

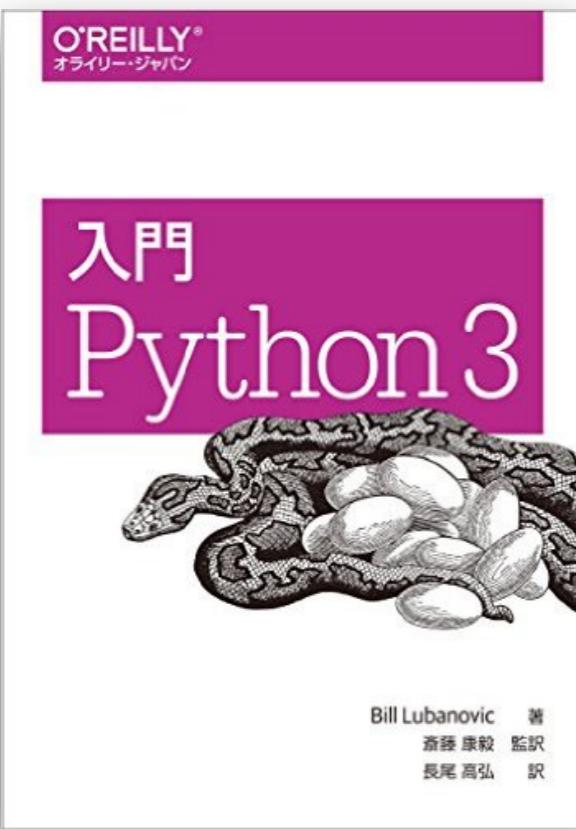
▼ More

**Created by:** Stanford University



**Taught by:** Andrew Ng, Associate Professor, Stanford University; Chief Scientist, Baidu; Chairman and Co-founder, Coursera

# next step



<https://goo.gl/ccYxtS>

<https://goo.gl/POq5el>



# next step



<https://goo.gl/EXcphV>



<https://goo.gl/51viVt>



next step

## 機械学習の基礎 & Python

G's Academy Tokyo Pro Course

Munesada Yohei



<https://goo.gl/JE8Vvx>



# next step



<https://goo.gl/J6jXhp>



# Course Summary



# course catalog

- ✓ Pythonとは
- ✓ Python基本編
- ✓ モジュールとパッケージ
- ✓ Webスクレイピング（基礎）
- ✓ Webスクレイピング（実践）
- ✓ Flaskを用いたWebアプリケーション（基礎）
- ✓ Flaskを用いたWebアプリケーション（実践）
- ✓ 手書き文字の判定アプリを作ろう（機械学習）
- ✓ デプロイ
- ✓ 開発演習（任意提出）

*Pre study*

*1st day*

*2nd day*



# 宿題

1. 本日の手書き判定アプリを完成させてください
2. Heroku アカウントを作成してください  
<https://signup.heroku.com/dc>



# Thank you



<http://www.yoheim.net>



@yoheiMune

<https://flic.kr/p/mzmQK2>