

全学教育科目展開科目 カレントトピックス科目

# 「実践 機械学習1」 2019

後期に「実践 機械学習2」(月曜5講時)をやります.

篠原 歩

東北大学 大学院情報科学研究科

東北大学 工学部 電気情報物理工学科

# 目的と概要

- 「人工知能」を支える基盤技術の一つである機械学習について学ぶ.
- 線形分類器やサポートベクトルマシン, 決定木, ニューラルネットワークなどについて, 手を動かしながらシステムを作成し, 実データを処理する過程を通じてそのエッセンスを探る.
- プログラム言語Pythonの基本と関連するライブラリの使い方も併せて習得する.

## 到達目標

- 機械学習の基礎的な知識を身に付けると同時に, データ処理の技術も身に付ける. Python言語で基本的な処理プログラムが書けるようになり, ライブラリの使い方がわかるようになる.

# 授業計画（予定）

受講生の上限:100名

- ICL演習室2（M105）の端末を使って演習を行いながら授業を進めていく.
- 自分のノートパソコン（Windows, Mac, Linux）を持ち込んで作業しても構いません.

- 人工知能と機械学習, 準備と環境構築
- 分類問題とデータ処理
- 2クラス分類問題, 多クラス分類問題
- データの可視化
- 学習アルゴリズムの作成と評価
- 次元削減
- 回帰問題
- $k$ 近傍法
- 決定木
- ランダムフォレストとブースティング
- サポートベクトルマシン
- ニューラルネットワーク

成績評価  
出席と演習の進み具合

授業時間外に予習復習が必要である.

Python言語に対する前提知識は必要としないが, 基本的なパソコンの操作には十分に慣れていることが望ましい.

※ 強化学習・深層学習は「実践 機械学習2」で行います.

# 人工知能研究の歴史

- 第1次AIブーム (1950年代～70年代)
  - ダートマス会議 (1957)  
The Dartmouth Summer Research Project on Artificial Intelligence
  - パーセプトロン, 推論と探索, ELIZA,...
- 第2次AIブーム (1980年代)
  - ニューラルネットワーク (バックプロパゲーション)
  - エキスパートシステム
  - 通産省による第五世代コンピュータプロジェクト
    - 論理型言語Prolog, 並列計算機
- 第3次AIブーム (2006年～) ⇐ いまここ
  - 深層学習 (ディープラーニング) による画像認識,...
  - ビッグデータ, ロボット, 自動運転,...
  - AlphaGo (2016),...

# 機械学習

- 人工知能の研究課題の一つ.
- 人間が自然に行っている学習能力と同様の機能を計算機で実現しようとする技術.
- 分類や判断の規則をプログラムとして明示的に与えられるのではなく, 訓練例や経験, 質問応答によって一般化された知識を自ら獲得する.
- 計算機の処理能力の向上とプログラム言語やライブラリの充実, 機械可読なデータの蓄積によって「機械学習」を手軽に試せる環境が整ってきた.

- 実際に体験してみましょう.
- その中でどんな処理が行われているのかを知りましょう.

この講義のねらい

# この講義を楽しむために

- 教科書は特に指定しませんが、参考になる書籍や雑誌がたくさん出版され、ウェブ上にも有益な資料が大量に公開されているので積極的に利用しましょう。
- 受け身にならずに、自ら学び、また互いに教え合ってください。教えることで学ぶことはとても多いです。
- Python や Jupyter (Google Colaboratory) を他の勉強にも活用してみましょう。線形代数, 微分積分, ...
- オンライン教材を活用しましょう(後述)。
- TAの方々に相談しましょう。

# 演習のキーワード

- プログラミング言語 Python
  - `scikit-learn` モジュール 機械学習
  - `numpy` 数値演算
  - `sympy` 記号演算
  - `matplotlib` モジュール 可視化, グラフ描画
  - `pandas` データ分析, 加工
  - `keras` 深層学習
- Jupyter Notebook
- Google Colaboratory



# Python のオンライン教材の紹介

Aidemy <https://aidemy.net/>

- Python入門と機械学習基礎編は無料で学べる.
- NumPyやMatplotlib, 深層学習など, より高度な技術についても学べる(有料).

Progate <https://prog-8.com/>

- Python の途中まで無料で学べる.
- 他の色々なプログラム言語やツールについても学べる.

Paiza ラーニング <https://paiza.jp/>

- Python3 入門編を無料で学べる.
- 機械学習入門編を一部無料で学べる.



# Google Colaboratory の始め方

(事前準備) Google アカウントを作っておく.

1. Chromeブラウザで  
<https://colab.research.google.com/>  
を開く.
2. notebookを新規作成するには  
【ファイル】→【Python 3 のノートブックを新規作】  
をクリックする.
3. Markdown 記法に慣れる.

# 2019年4月19日の目標 (1/2)

## 1. 本日の出欠確認

- 回覧している名簿に自筆で署名する.
  - 名簿に名前がない場合は履修登録できていません.
- 名札を作る.

## 2. ISTU <https://istu3g.dc.tohoku.ac.jp/> で 講義資料がダウンロードできることを確認する.

## 3. Google アカウントを作る.

- 既に取得済みの人はそれを使っていいです.

# 2019年4月19日の目標 (2/2)

## 4. iris データを触ってみる.

- Python の「辞書」構造の使い方を知る.
- Python の配列の使い方を知る.
- Python の for文の使い方を知る.

<http://bit.ly/JKG1a01> を開き,  
【ファイル】→【ドライブにコピーを保存する...】してから  
作業する.

- 作業結果を【ファイル】→【印刷】で pdf ファイルとして  
保存したものを ISTU にアップロードしてください.

# 機械学習 (1/2)

- 分類 (classification)
  - 正解となる, 入力データと離散的なクラスの組み合わせで学習し, 未知のデータに対してそのクラスを予測する.
- 回帰 (regression)
  - 正解となる, 入力データと数値の組み合わせで学習し, 未知のデータに対してその値 (連続値) を予測する.
- クラスタリング (clustering)
  - データを何らかの基準でグループ分けする.
- 次元削減 (dimension reduction)
  - 高次元のデータを, 可視化や計算量削減などのために, 望ましい性質を保ったまま低次元に写像する.

# 機械学習 (2/2)

- その他
  - 推薦  
ユーザが好みそうなアイテムや閲覧しているアイテムに類似したアイテムを提示する.
  - 異常検知  
不審なアクセスなど, 普段とは異なる挙動を検知する.
  - 頻出パターンマイニング  
データ中に, 高頻度に出現するパターンを抽出する.
  - 強化学習  
試行錯誤しながら, 環境からのフィードバック情報に基づいて, 望ましい行動を徐々に身に付けていく.

# 機械学習：分類

- 教師あり学習
- 予測対象は離散的な値（クラス, カテゴリ）.  
例：スパムメール, 猫の写真, ...
- クラス数が2のものを二値分類（2クラス分類）,  
3以上のものを多値分類（多クラス分類）という.

<http://scikit-learn.org/stable/modules/multiclass.html>

# 分類の代表的なアルゴリズム

- パーセプトロン (Perceptron)
- ロジステック回帰 (Logistic Regression)
- サポートベクトルマシン (Support Vector Machine)
- ニューラルネットワーク (Neural Network)
- k-近傍法 (k-Nearest Neighbor Method)
- 決定木 (Decision Tree)
- ランダムフォレスト (Random Forest)
- GBDT (Gradient Boosted Decision Tree)

Iris データベースを開いてみる.

```
from sklearn import datasets  
iris = datasets.load_iris()
```





# Python メモ

- 文字列を表すのは 'hoge' でも "hoge" でもよい.
- 辞書(ディクショナリ)は, キーと値のペアを記録する.

- 値の取り出し

```
dict[key]
```

- 値の代入

```
dict[key] = val
```

補足:これまでにでてきている例では, key が文字列型なので,  
iris['target'] のようにアクセスしている.  
Python では文字列以外のオブジェクトをkey にすることもできる.

# Iris データの形が少しわかってきた.

1	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	クラス
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
⋮	⋮	⋮	⋮	⋮	⋮
	5.9	3.0	5.1	1.8	2

# Iris データの形が少しわかってきた.

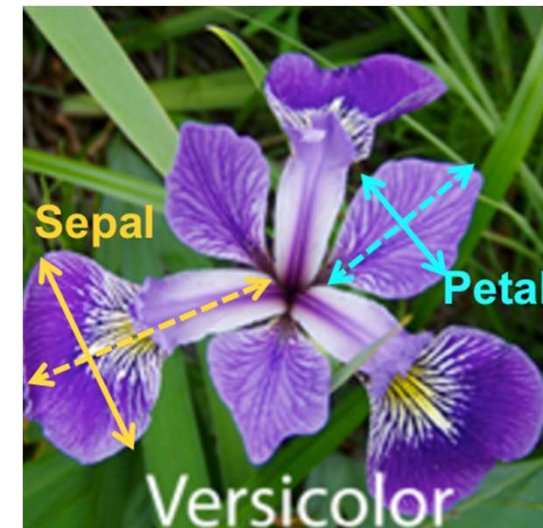
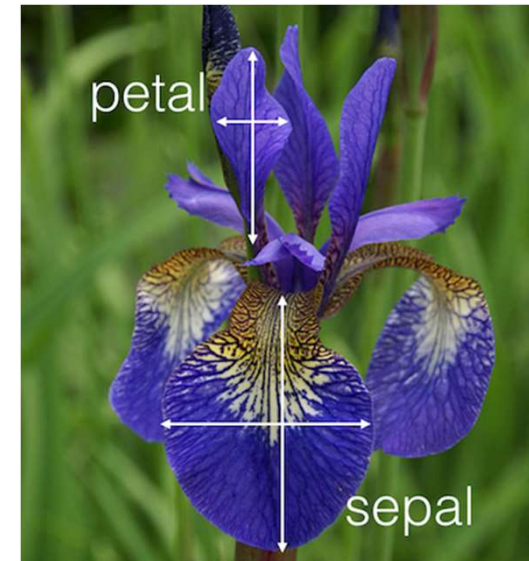
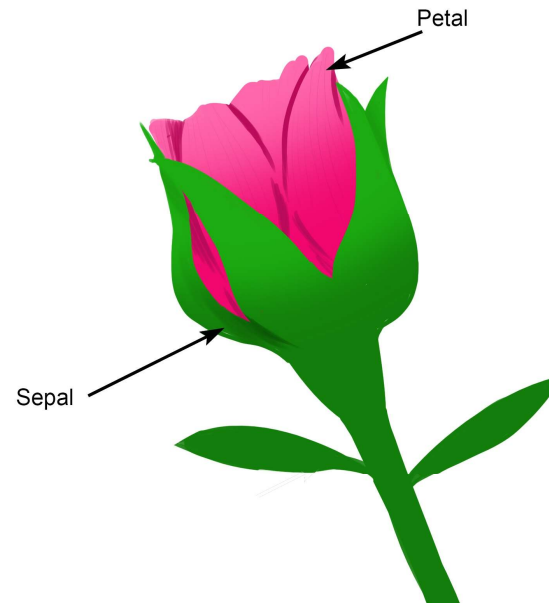
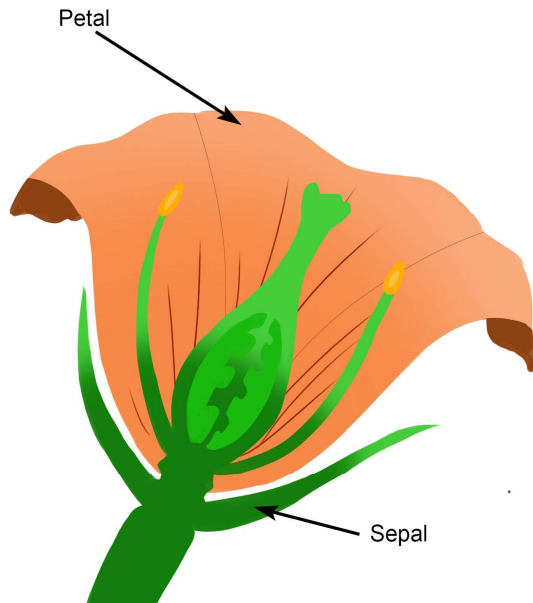
feature\_names

1	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	クラス	target_names
0	5.1	3.5	1.4	0.2	0	= setosa
1	4.9	3.0	1.4	0.2	0	
2	4.7	3.2	1.3	0.2	0	
3	4.6	3.1	1.5	0.2	0	
4	5.0	3.6	1.4	0.2	0	
5	5.4	3.9	1.7	0.4	0	
6	4.6	3.4	1.4	0.3	0	
⋮	⋮	⋮	⋮	⋮	⋮	
	5.9	3.0	5.1	1.8	2	= virginica

data

target

# Sepal (萼片<sup>がくへん</sup>) と Petal (花弁)



<https://www.quora.com/What-is-the-difference-between-sepals-and-petals>

<https://melindahiggins2000.github.io/N741UnsupervisedLearning/UnsupervisedLearning.html>

[https://holgerbrandl.github.io/kotlin4ds\\_kotlin\\_night\\_frankfurt//krangl\\_example\\_report.html](https://holgerbrandl.github.io/kotlin4ds_kotlin_night_frankfurt//krangl_example_report.html)

# Iris データベースを開いてみる.

```
from sklearn import datasets  
iris = datasets.load_iris()
```



setosa



versicolor



virginica

[http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load\\_iris.html](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html)

なお, アヤメ(菖蒲), ショウブ(菖蒲), カキツバタ(杜若)の分類とは無関係.

# Irisデータのサイズを確認しよう

```
print(iris.data.shape)
```

(150, 4)

```
print(iris.target.shape)
```

(150,)



# Python メモ

- Python の配列の要素の指定

```
a = [10, 20, 30]  
print( a[0] )  
print( a[1] )  
print( a[2] )  
print( a[-1] )  
print( a[1:2] )  
print( a[1:3] )
```

for 文を使ったループ

```
for i in range(3):  
    print(a[i])
```

# データの可視化

- Matplotlib を使うと、データの可視化を容易に行うことができる. <https://matplotlib.org/>
- さまざまなオプションが用意されているのでマニュアルや表示例を見てみよう.
- `matplotlib.pyplot.imshow( )`



# 次回の予習にどうぞ

- Scikit-learn の “The Digit Dataset” を開いて、その中を調べてみよう.

```
from sklearn import datasets  
digits = datasets.load_digits()
```

- データの大きさは？ 特徴量の数は？  
特徴量の名前は？ クラス名(ターゲット名)は？  
:

# 2019年4月26日の目標

今後も毎回行ってください

- 出席名簿に自筆で署名する.
- ISTU の難易度アンケート, ミニットペーパー, 作業記録
- まずは先週の続きを行おう. <http://bit.ly/2ZzD5zB>  
(一部誤植を修正しました)
- digits データを触ってみよう.
  - 先週 iris で行った作業の復習を兼ねて
  - print( ) を便利に使おう.
  - matplotlib を使ってデータの可視化しよう.

<https://bit.ly/2VreFJh> を開き,  
【ファイル】→【ドライブにコピーを保存する...】してから  
作業する.

作業結果を pdf ファイルとしてISTU にアップロードしてください.

# 前回4月19日の難易度・理解度

あなたにとって、本日のゼミの難易度・理解度は？

	回答数	割合
易しすぎ，知っていることばかりだった	3	7%
ちょうどよかった	28	65%
難しすぎ，ほとんどついていけなかった	12	28%

今後も出席したときは，必ず回答してください。

# 【再】Python のオンライン教材の紹介

Aidemy <https://aidemy.net/>

- Python入門と機械学習基礎編は無料で学べる.
- NumPyやMatplotlib, 深層学習など, より高度な技術についても学べる(有料).

Progate <https://prog-8.com/>

- Python の途中まで無料で学べる.
- 他の色々なプログラム言語やツールについても学べる.

Paiza ラーニング <https://paiza.jp/>

- Python3 入門編を無料で学べる.
- 機械学習入門編を一部無料で学べる.

Chainer チュートリアル <https://tutorials.chainer.org/ja/> <sup>New</sup>

- 深層学習のライブラリの説明だが, Python入門や scikit-learn の部分だけでもとても有用. 無料.
- Google Colaboratory を使っている.



# 文字列フォーマット

(print関数などで使うことが多い)

```
i = 10  
pi = 3.14  
name = "taro"
```

- % 演算子 (古典的)

```
print( "i = %d, pi = %f, name = %s" % (i, pi, name) )
```

- str.format 関数 (python 2.6～)

```
print( "i = {0}, pi = {1}, name = {2}".format(i, pi, name) )
```

- f文字列 (python 3.6～)

```
print( f"i = {i}, pi = {pi}, name = {name}" )
```

出力結果

```
i = 10, pi = 3.14, name = taro
```



# numpy の表示オプション

```
import numpy as np  
np.get_printoptions()
```

```
{'edgeitems': 3,  
 'floatmode': 'maxprec',  
 'formatter': None,  
 'infstr': 'inf',  
 'legacy': False,  
 'linewidth': 75,  
 'nanstr': 'nan',  
 'precision': 8,  
 'sign': '-',  
 'suppress': False,  
 'threshold': 1000}
```

```
np.set_printoptions(threshold=500000)
```

```
np.set_printoptions(threshold=np.inf)
```



# リストの内包表記

- `a = [ 3*i + 1 for i in range(5) ]`

- `a = [ i for i in range(15) if i % 3 == 1 ]`

`[1, 4, 7, 10, 13]`



# numpy の配列で 条件を満たすものを数える.

```
import numpy as np
a = np.array([2, 3, 2, 3, 5, 3, 3])
a3 = (a==3)
print(a3)
[False  True False  True False  True  True]
print(sum(a3))
4
```

X [y==1] という記述は, この応用だった.



# 練習問題

digits のデータにおいて,  
クラス0, クラス1, ..., クラス9 に含まれるものはそれぞれいくつかを調べよ.

(ヒント: クラス 0 は 178 個あり, 全部で 1797個ある)

# 辞書を使った解決例(1)

```
dict = {}  
for k in y:           # この in は列挙  
    if k in dict:     # この in は条件判定文  
        dict[k] += 1  
    else:  
        dict[k] = 1  
print(dict)
```

# 辞書を使った解決例(2)

## get メソッドを使う

```
dict = {}  
for k in y:  
    dict[k] = dict.get(k, 0) + 1  
print(dict)
```


dict の中に キー k が見つからなかったときには  
この値を返す.

他にも `collections.defaultdict` を使う方法などもあります.

# collections.Counter を使った解決策

```
import collections
c = collections.Counter(y)
print(c)
```

# 2019年5月10日の目標

- 出席名簿に自筆で署名する.  今後も毎回行ってください
- ISTU の難易度アンケート, ミニットペーパー
- まずは先週の続きを行おう. <https://bit.ly/2VreFJh>
- iris データを分類してみよう.
  - まずは二値分類(2クラス分類)問題として, かつ特徴量を1つに制限したところで分類問題の基本を習得.
  - matplotlib の scatter( )と hist( )
  - scikit-learn の predict( )と fit( )
  - オブジェクト指向とクラス

<http://bit.ly/2VVkwXF>を開き,

【ファイル】→【ドライブにコピーを保存する...】してから作業する.

# 前回4月26日の難易度・理解度

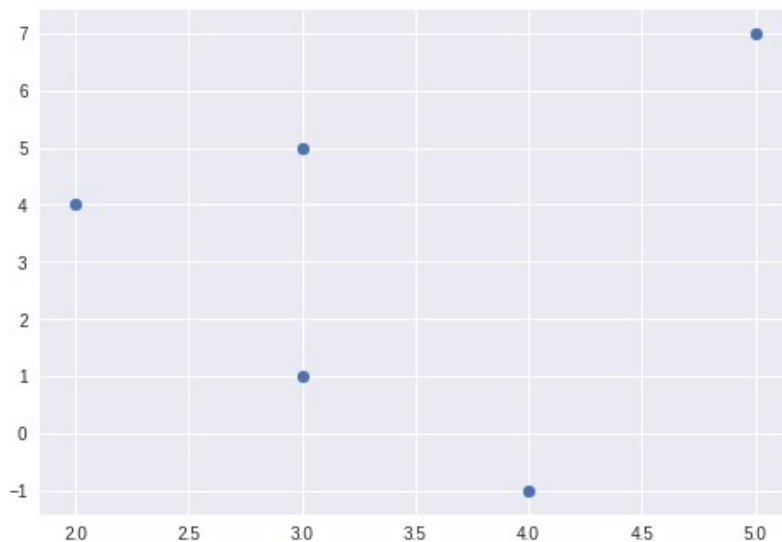
あなたにとって、本日のゼミの難易度・理解度は？

	回答数	割合
易しすぎ, 知っていることばかりだった	5	10%
ちょうどよかった	41	84%
難しすぎ, ほとんどついていけなかった	3	6%

# 散布図 matplotlib.pyplot.scatter(x, y)

```
import matplotlib.pyplot as plt
```


```
plt.scatter([2, 3, 5, 4, 3], [4, 1, 7, -1, 5])
```



x や y は list でも numpy.ndarray でもよい.

色やマーカーの形なども含めて  
豊富なオプションが用意されているので  
用途に合わせて工夫をしてみよう.

# 2019年5月17日の目標

- 出席名簿に自筆で署名する.  今後も毎回行ってください
- ISTU の難易度アンケート, ミニットペーパー
- 先週の続きを行おう <http://bit.ly/2VVkwXF>
- iris データを二値分類(2クラス分類)問題として, かつ特徴量を1つに制限したところで分類問題の基本を習得.
  - matplotlib の scatter( ) と hist( )
  - scikit-learn の predict( ) と fit( )
  - オブジェクト指向とクラス
- 次回以降の作業ファイル(予習用) <http://bit.ly/2Hwidku>



# 前回5月10日の難易度・理解度

あなたにとって、本日のゼミの難易度・理解度は？

	回答数	割合
易しすぎ, 知っていることばかりだった	0	0%
ちょうどよかった	37	82%
難しすぎ, ほとんどついていけなかった	8	18%

今回から5段階のアンケートにします.

# list と numpy.ndarray

N-dimensional array

```
a1 = [2, 3, 5]
```

```
type(a1)
```

```
> list
```

```
print(a1)
```

```
> [2, 3, 5]
```

```
import numpy as np
```

```
b1 = np.array([2, 3, 5])
```

```
type(b1)
```

```
> numpy.ndarray
```

ndarray に変換する関数

```
print(b1)
```

```
> [2 3 5]
```

```
a2 = list(b1)
```

```
type(a2)
```

```
> list
```

list に変換する関数

数値列を扱う上では似た振る舞いをするし、優れたライブラリでは必要に応じて内部で相互変換してくれるのであまり意識しなくてすむが、違いもあるので**ときどき注意が必要**。

# 2次元のデータ list と numpy.ndarray

## list

```
a1 = [[5, 6], [7, 8]]
```

```
type(a1)
```

```
> list
```

```
print(a1)
```

```
> [[5, 6], [7, 8]]
```

```
a1[1][1]
```

```
> 8
```

```
a1[1,1]
```

```
> エラー
```

## numpy.ndarray

```
b1 = np.array(a1)
```

```
type(b1)
```

```
> numpy.ndarray
```

```
print(b1)
```

```
> [[5 6]
    [7 8]]
```

```
b1[1][1]
```

```
> 8
```

```
b1[1,1]
```

```
> 8
```

# ヒストグラム matplotlib.pyplot.hist(x)

```
import matplotlib.pyplot as plt
```

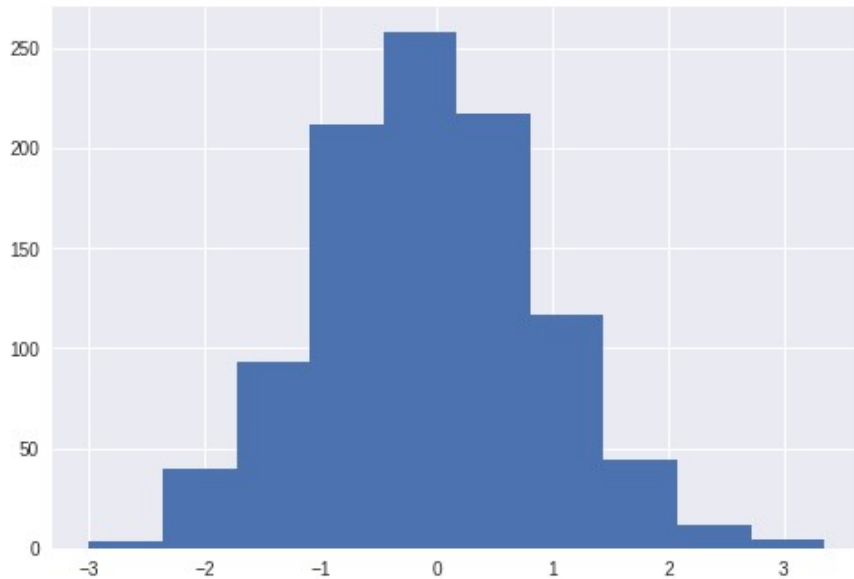
```
import numpy as np
```

```
x = np.random.randn(1000)
```

平均0, 標準偏差1 の標準正規分布にしたがってランダムに1000個の数値を生成

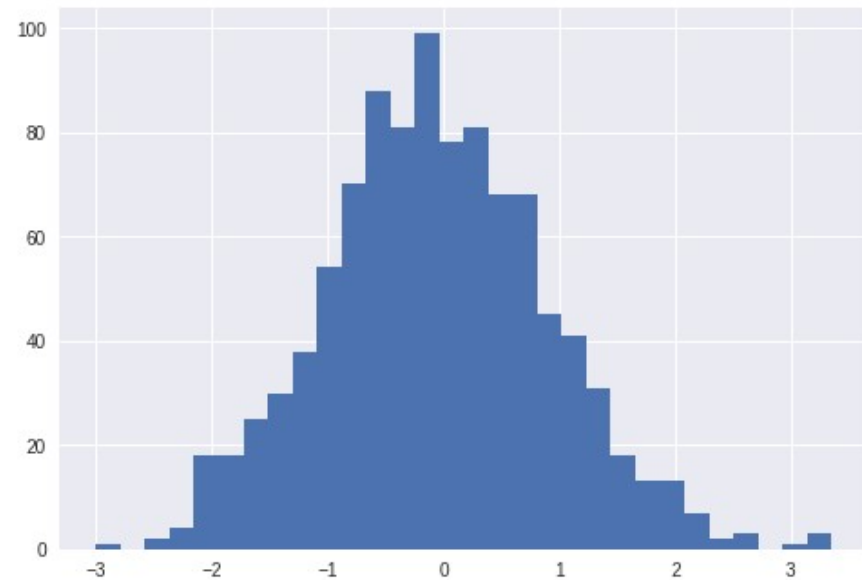
```
plt.hist(x)
```

```
plt.show()
```



```
plt.hist(x, bins=30)
```

```
plt.show()
```



他にも[豊富なオプション](#)が用意されている.



# 関数定義

```
def predict(x):
```

```
    "与えられた値 x が 5.3 より大きいときは 1 を, 小さいときは 0 を返す."
```

```
    if x >= 5.3:
```

```
        return 1
```

```
    else:
```

```
        return 0
```

ドキュメント文字列

プログラムの動作には影響しないが、  
ユーザに仕様を伝える大事な役割を果たす。  
単なるコメント文ではない。

`help(predict)`

ドキュメントの自動生成にも使われる。

Python ではブロック構造をインデントの量で表す。

- 構造が「見た目」に一致する。
- Colaboratory の中では, [Tab] でインデントを増やし,  
[Shift]+[Tab] でインデントを減らすと便利  
(複数行を選択すればまとめてインデントの調整もできる)。
- Colaboratory の中では [Ctrl]+'/' でコメント# のオンオフができる。

# scikit-learn 準拠の予測モデルの定義例

```
from sklearn.base import BaseEstimator, ClassifierMixin
```

```
class MyClassifierNaive(BaseEstimator, ClassifierMixin):
```

```
    "閾値 5.3 より大きいかどうかで 1 か 0 を返す分類器"
```

```
    def __init__(self):
```

```
        "分類器のコンストラクタ. 閾値を 5.3 と決め打ちする."
```

```
        self.threshold = 5.3
```

self は自分自身を表す  
(C++ の this に相当)  
Python では省略しない.

```
    def fit(self, X, y):
```

```
        """データへのフィッティングを行うメソッド. 実際には何もしない.
```

```
        ただし, fitメソッドは self を返すことになっている. """
```

```
        return self
```

改行を含んだ複数行の文字列は  
""" または ''' で囲む

```
    def predict(self, X):
```

```
        "予測を行う."
```

```
        pred_y = []
```

```
        for x in X:
```

```
            if x >= self.threshold:
```

```
                pred_y.append(1)
```

```
            else:
```

```
                pred_y.append(0)
```

```
        return np.array(pred_y)
```

# scikit-learn 準拠の予測モデルの定義例

```
from sklearn.base import BaseEstimator, ClassifierMixin
```

```
class MyClassifierNaive(BaseEstimator, ClassifierMixin):
```

```
    "閾値 5.3 より大きいかどうかで 1 か 0 を返す分類器"
```

```
    def __init__(self):
```

```
        "分類器のコンストラクタ. 閾値を 5.3 と決め打ちする."
```

```
        self.threshold = 5.3
```

self は自分自身を表す  
(C++ の this に相当)  
Python では省略しない.

```
    def fit(self, X, y):
```

```
        """データへのフィッティングを行うメソッド. 実際には何もしない.
```

```
        ただし, fitメソッドは self を返すことになっている. """
```

```
        return self
```

改行を含んだ複数行の文字列は  
""" または ''' で囲む

```
    def predict(self, X):
```

```
        "予測を行う. "
```

```
        pred_y = []
```

```
        for x in X:
```

```
            if x >= self.threshold:
```

```
                pred_y.append(1)
```

```
            else:
```

```
                pred_y.append(0)
```

```
        return np.array(pred_y)
```

少し上級者向けの書き方

```
def predict(self, X):
```

```
    "予測を行う. "
```

```
    return [ 1 if x >= self.threshold else 0 for x in X ]
```

内包表記と三項演算子を使えば1行でも書ける.

(C 言語の `x >= threshold ? 1 : 0` に相当)

# 予測モデルの使い方の例

X, y にデータが入っているとして

```
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data[iris.target!=2, 0]
y = iris.target[iris.target!=2]
```

```
model = MyClassifierNaive()
```

```
model.fit(X, y)
```

```
print("予測結果は", model.predict(X))
```

```
print("予測精度は{0}です".format(model.score(X, y)))
```



# 練習問題（締切:5月22日）

- ① ここまで, iris データのクラス0 (setosa) とクラス1(versicolor) の分類を, 0番目の特徴量(sepal length)を使って行ってきた. これに倣って同様に, 1番目の特徴量 (sepal width) を使った分類を行ってみよう.
- 具体的には, まず【ファイル】→【ドライブにコピーを保存】で新たなファイルとして保存してから, 冒頭の

```
X = iris.data[iris.target!=2, 0]
```

```
y = iris.target[iris.target!=2]
```

の **0** を **1** に変えれば1番目の特徴量を取り出したデータが得られる.
  - ただし単純に [shift]+[return] していきただけではうまく解析できないかもしれないので, 作業内容をよく確認しながら進もう.
    - ヒント: predict の判定文の不等号の向きを変える必要があるかも?
    - ヒント: 閾値を探す範囲を変える必要があるかも?

# 練習問題（締切：5月22日）

- ② 前問と同様に、2番目の特徴量 (`petal length`) を使った分類や3番目の特徴量 (`petal width`) を使った分類を行ってみよう.
- ③ 【中級者向け】 `predict` の中で行っている判定の不等号の向きを明示的に書き直さなくてもうまく分類できるようにしてみよう.
- ④ 【上級者向け】 さらに同様に、class 0 (`setosa`) と class 2 (`virginica`) の分類、また class 1 と class 2 の分類を行ってみよう. `fit` や `predict` をうまく書けば、クラスラベル(0 や 1 や 2など)が変わってもそのまま対応できる分類器が作れるはずである.

# 2019年5月24日の目標

今後も毎回行ってください

- 出席名簿に自筆で署名する.
- ISTU の難易度アンケート, ミニットペーパー
- iris データを引き続き分類してみよう.
  - 前回の続き <http://bit.ly/2VVkwXF>
    - 補足資料 <http://bit.ly/2I3waa7>
  - 2番目の特徴量に注目 <http://bit.ly/2VL7mrV>  
class MyClassifier の fit関数の実装例を示している.
  - 3番目 (同上) <http://bit.ly/2K173XQ>
  - 4番目 (同上) <http://bit.ly/2K6wxTA>
- 特徴量を2つに増やす. <http://bit.ly/2Hwidku>

# 前回5月17日の難易度・理解度

あなたにとって、本日のゼミの難易度・理解度は？

	回答数	割合
易しすぎ，知っていることばかりだった	0	0%
やや易しかった	1	2%
ちょうどよかった	9	22%
やや難しかった	26	62%
難しすぎ，ほとんどついていけなかった	6	14%

# 自作の fit 関数

```
def fit(X, y):
```

“データ(X, y)を最もよく説明する閾値 t を返す”

```
    best_t = None
```

```
    best_score = 0
```

```
    for t in X:
```

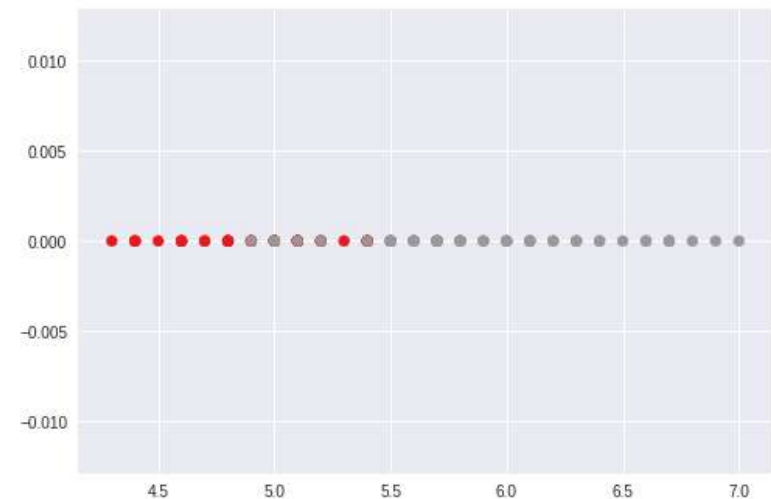
```
        new_score = score(X, y, t)
```

```
        if new_score > best_score:
```

```
            best_t = t
```

```
            best_score = new_score
```

```
    return best_t
```



# MyClassifier の実装例

```
class MyClassifier(MyClassifierNaive):
    def __init__(self, t = 0.0):
        self.threshold = t

    def fit(self, X, y):
        """データへのフィッティングを行うメソッド. """
        best_t = None
        best_score = 0
        for t in X:
            new_score = self._score(X, y, t)
            if new_score > best_score:
                best_t = t
                best_score = new_score
        self.threshold = best_t
        return self
```

```
    def _score(self, X, y, t):
        """データ x, y に対して閾値 t の正解率を調べて返す"""
        ok = 0
        for (xi, yi) in zip(X, y):
            if self._predict(xi, t) == yi:
                ok += 1
        return (ok * 100) / len(X)

    def _predict(self, x, t):
        """データx に対して閾値をtと置いたときの予測結果を返す"""
        if x >= t:
            return 0
        else:
            return 1
```

# MyClassifier の実装例

```
class MyClassifier(MyClassifierNaive):
    def __init__(self, t = 0.0):
        self.threshold = t

    def fit(self, X, y):
        """データへのフィッティングを行うメソッド. """
        best_t = None
        best_score = 0
        for t in X:
            new_score = self._score(X, y, t)
            if new_score > best_score:
                best_t = t
                best_score = new_score
        self.threshold = best_t
        return self
```

```
    def _score(self, X, y, t):
        """データ x, y に対して閾値 t の正解率を調べて返す"""
        ok = 0
        for (xi, yi) in zip(X, y):
            if self._predict(xi, t) == yi:
                ok += 1
        return (ok * 100) / len(X)

    def _predict(self, x, t):
        """データx に対して閾値をtと置いたときの予測結果を返す"""
        if x >= t:
            return 0
        else:
            return 1
```

Q: なぜ fit(self, X, y) で return self するのですか？

# MyClassifier の実装例

```
class MyClassifier(MyClassifierNaive):
    def __init__(self, t = 0.0):
        self.threshold = t

    def fit(self, X, y):
        """データへのフィッティングを行うメソッド. """
        best_t = None
        best_score = 0
        for t in X:
            new_score = self._score(X, y, t)
            if new_score > best_score:
                best_t = t
                best_score = new_score
        self.threshold = best_t
        return self
```

```
    def _score(self, X, y, t):
        "データ x, y に対して閾値 t の正解率を調べて返す"
        ok = 0
        for (xi, yi) in zip(X, y):
            if self._predict(xi, t) == yi:
                ok += 1
        return (ok * 100) / len(X)

    def _predict(self, x, t):
        "データx に対して閾値をtと置いたときの予測結果を返す"
        if x >= t:
            return 0
        else:
            return 1
```

Q: なぜ fit(self, X, y) で return self するのですか？

A: Scikit-learn で、そう書くように定められているから.



# Sckit-learnの分類器の典型的な使用例

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
model.fit(X, y)
score = model.score(X, y)
print("正解率は", score)
```

# Sckit-learnの分類器の典型的な使用例

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression()
```

```
model.fit(X, y)
```

```
score = model.score(X, y)
```

```
print("正解率は", score)
```

一行で書ける

```
score = model.fit(X, y).score(X, y)
```

「メソッド チェーン」

# 2019年5月31日の目標

今後も毎回行ってください

- 出席名簿に自筆で署名する.
- ISTU の難易度アンケート, ミニットペーパー
- 前回の復習と補足 <http://bit.ly/2Hwidku>
- 機械学習の評価と汎化性能 <http://bit.ly/2HLvh6R>
  - iris データの分類に関して
    - 丸暗記型分類器の精度について考える.
    - 精度の測り方
    - 汎化性能の評価法
      - ホールドアウト検証, k重交差検証
  - Pythonについて
    - リストとタプル
    - 疑似乱数の種(シード)

# 前回5月24日の難易度・理解度

あなたにとって、本日のゼミの難易度・理解度は？

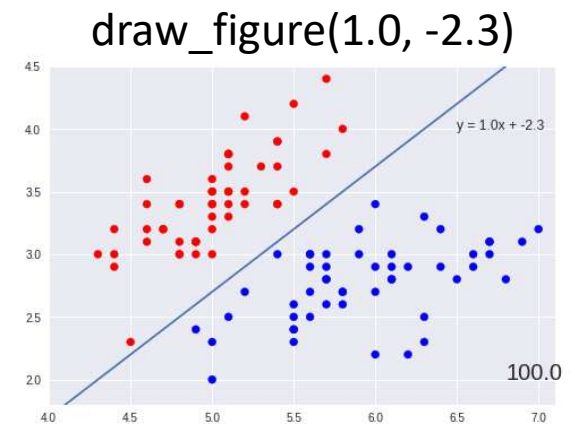
	回答数	割合
易しすぎ，知っていることばかりだった	0	0%
やや易しかった	3	6%
ちょうどよかった	22	47%
やや難しかった	20	43%
難しすぎ，ほとんどついていけなかった	2	4%

# 分離直線について

- class 0 (setosa) と class 1 (virgicolor) を  
特徴量0 (sepal length) と 特徴量 1 (sepal width) の組み  
合わせで 100%分離する直線が見つかりました.

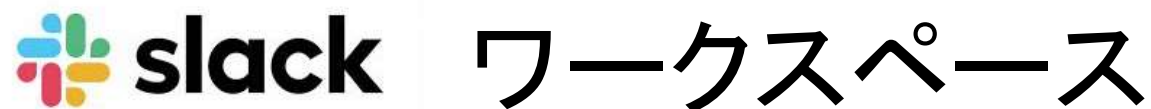
報告ありがとうございます(敬称略)

- (1.35, -4) 森山 航太
- (0.9, -1.8) 杉本 悠斗
- (0.82, -1.4) 竹谷 英久
- (1.3, -3.8) 山崎 峻平
- (1.2, -3.3) 石井 敬大
- (1, -2.3) 内田 徳之新



- plt.plot([4.0, 7.0], [0.39 × 4.0 + 0.2, 0.62 × 7.0 + 0.2]) ?? 佐々木 洸輔

なぜロジスティック回帰やパーセプトロン, 線形サポートベクトルマシンは  
これらの答えを見つけられなかったのだろうか ???



<https://jml1.slack.com> (実践 機械学習1)  
**はじめました**

- コミュニケーションツール (チャットツール)
  - ブラウザやスマホアプリで便利に使えます.
  - この講義に関する質問などを気楽に書き込んでください.
  - 質問に対する解答案も気楽に書き込んでください.
- ペンネームでの登録も可として運用してみます.  
(教員やTAは実名で登録しています)
- 登録用 招待リンク <http://bit.ly/2KnGIInh>

# 練習問題（締切:6月5日）

これまでに学んだことをじっくり復習しながら、  
いろいろなことを試してみよう。

- 我々は iris データのクラス0 (setosa) とクラス1(versicolor) の分類を, 0番目の特徴量(sepal length)と1番目の特徴量(sepal width) の2つを組合せて行ってきた.
  - 別の特徴量の組合せで分類を行ってみよう.  
さらに3つ, または4つすべてを使ってみたらどうなるだろうか？
  - クラス0 とクラス2 (virginica),  
また クラス1とクラス2の分類を試みよう.
  - クラス0 と 1 と 2 を一度に分類(3クラス分類)してみよう.
  - いろいろな分類器を使ってみよう.
- 【発展課題】 手書き文字の認識 digits をやってみよう.

# 2019年6月7日の目標

今後も毎回行ってください

- 出席名簿に自筆で署名する.
- ISTU の難易度アンケート, ミニットペーパー
- 前回の続き <http://bit.ly/2HLvh6R>
  - 精度の測り方
  - 汎化性能の評価法
    - ホールドアウト検証, k重交差検証
  - Pythonについて
    - 疑似乱数の種(シード)
- <http://bit.ly/2wFPHIf>
  - iris データと手描き文字認識
  - 多クラス分類問題



# 前回5月31日の難易度・理解度

あなたにとって、本日のゼミの難易度・理解度は？

	回答数	割合
易しすぎ，知っていることばかりだった	0	0%
やや易しかった	2	5%
ちょうどよかった	23	56%
やや難しかった	14	34%
難しすぎ，ほとんどついていけなかった	2	5%

# これまでの Colab ファイル一覧

- 4月19日 iris01 ~~<http://bit.ly/JKG1a01>~~  
<http://bit.ly/2ZzD5zB>
- 4月26日 digit01 <https://bit.ly/2VreFJh>
- 5月10日 iris\_classify01 <http://bit.ly/2VVkwXF>
- 5月17日 iris\_classify\_2features01 <http://bit.ly/2Hwidku>
  - すばらしい作品の例 <http://bit.ly/2VJ7mO1>
- 5月24日 補足資料 <http://bit.ly/2I3waa7>
  - 特徴量1に注目 iris\_classify02 <http://bit.ly/2VL7mrV>
  - 特徴量2に注目 iris\_classify03 <http://bit.ly/2K173XQ>
  - 特徴量3に注目 iris\_classify04 <http://bit.ly/2K6wxTA>
  - iris\_classify\_2features01 <http://bit.ly/2Hwidku>
- 5月31日 Classification01 <http://bit.ly/2HLvh6R>
- 6月7日 classifiers01 <http://bit.ly/2wFPHIf>

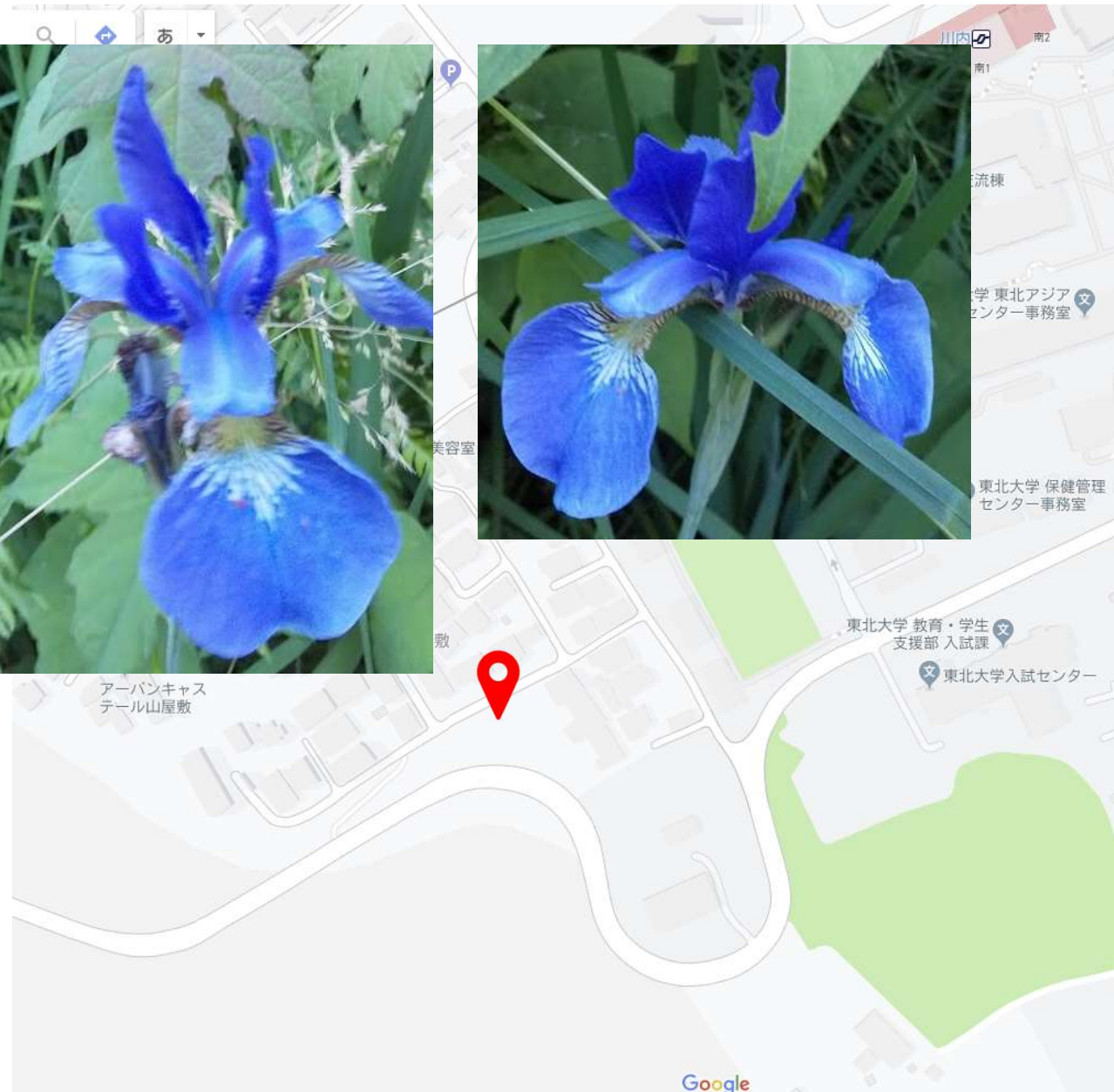
Slack 登録用 招待リンク <http://bit.ly/2KnGlnh>

# このすぐ近くで Iris 発見！ (20190606)





# このすぐ近くで Iris 発見！ (20190606)



# Iris データベースを開いてみる.

```
from sklearn import datasets  
iris = datasets.load_iris()
```



setosa



versicolor



virginica

[http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load\\_iris.html](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html)

なお, アヤメ(菖蒲), ショウブ(菖蒲), カキツバタ(杜若)の分類とは無関係.

# 今後の予習用(仮)

- 機械学習アルゴリズム
  - k 近傍法 <http://bit.ly/2wDYOcm> (仮)
  - 決定木 <http://bit.ly/2EUVgqA> (仮)
  - :

# 2019年6月14日の目標

- 出席名簿に自筆で署名する.
- ISTU の難易度アンケート, ミニットペーパー
- digits データをさまざまな分類器にかけてみよう.
  - <http://bit.ly/2MMLugU>
  - 手描き文字認識 digits の分類
  - 次元削減 (次元縮小)
- 機械学習アルゴリズム
  - k 近傍法 <http://bit.ly/2wDYOcm>

今後も毎回行ってください

今後も毎回行ってください

今後、毎回行ってください



# 前回6月7日の難易度・理解度

あなたにとって、本日のゼミの難易度・理解度は？

	回答数	割合
易しすぎ，知っていることばかりだった	0	0%
やや易しかった	2	5%
ちょうどよかった	13	33%
やや難しかった	18	46%
難しすぎ，ほとんどついていけなかった	6	15%

計 39

(45名出席しているはずなのに...)



# 2019年6月21日の目標

- 出席名簿に自筆で署名する.
- ISTU の難易度アンケート, ミニットペーパー
- 前回の補足: マウスで手書きした数字を認識
  - <http://bit.ly/2IsjxGN> (TAの市川くんの作品)
- 機械学習アルゴリズム
  - k近傍法 (KNeighborsClassifier)  
<http://bit.ly/2wDY0cm>
- 次回の予習用 決定木 <http://bit.ly/2EUVgqA>

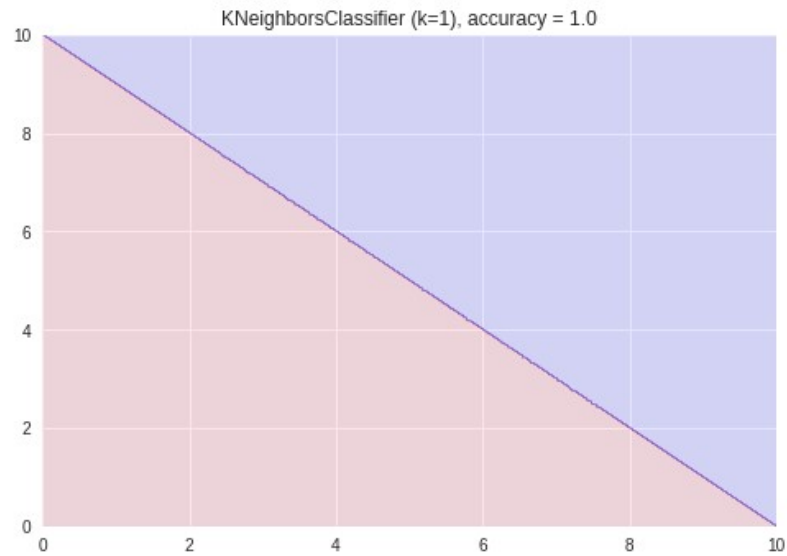
今後も毎回行ってください

# 前回6月14日の難易度・理解度

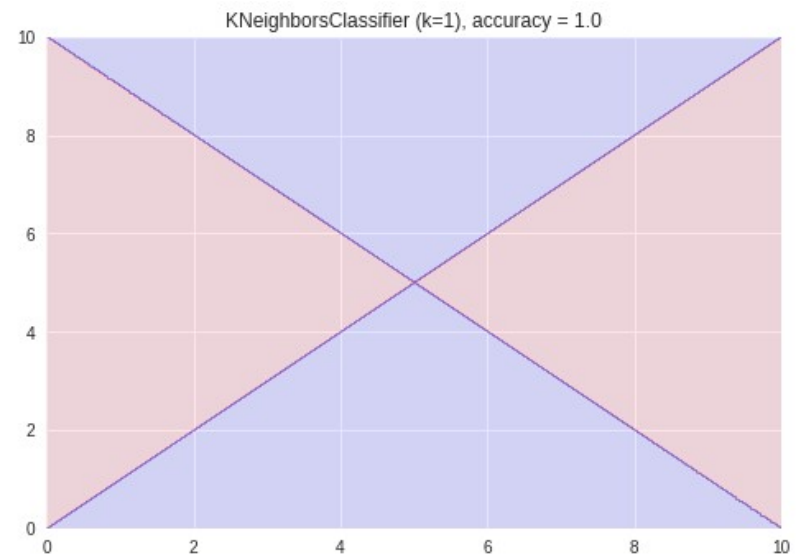
あなたにとって、本日のゼミの難易度・理解度は？

	回答数	割合
易しすぎ，知っていることばかりだった	0	0%
やや易しかった	4	9%
ちょうどよかった	22	48%
やや難しかった	18	39%
難しすぎ，ほとんどついていけなかった	2	4%

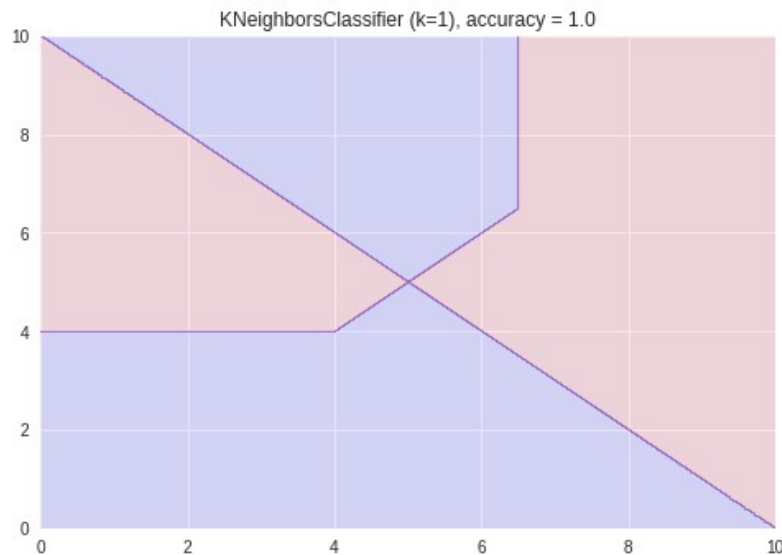
## 解答例（もちろん別解が無数に存在します）



$X = [[2, 6], [4, 8]]$   
 $y = [0, 1]$

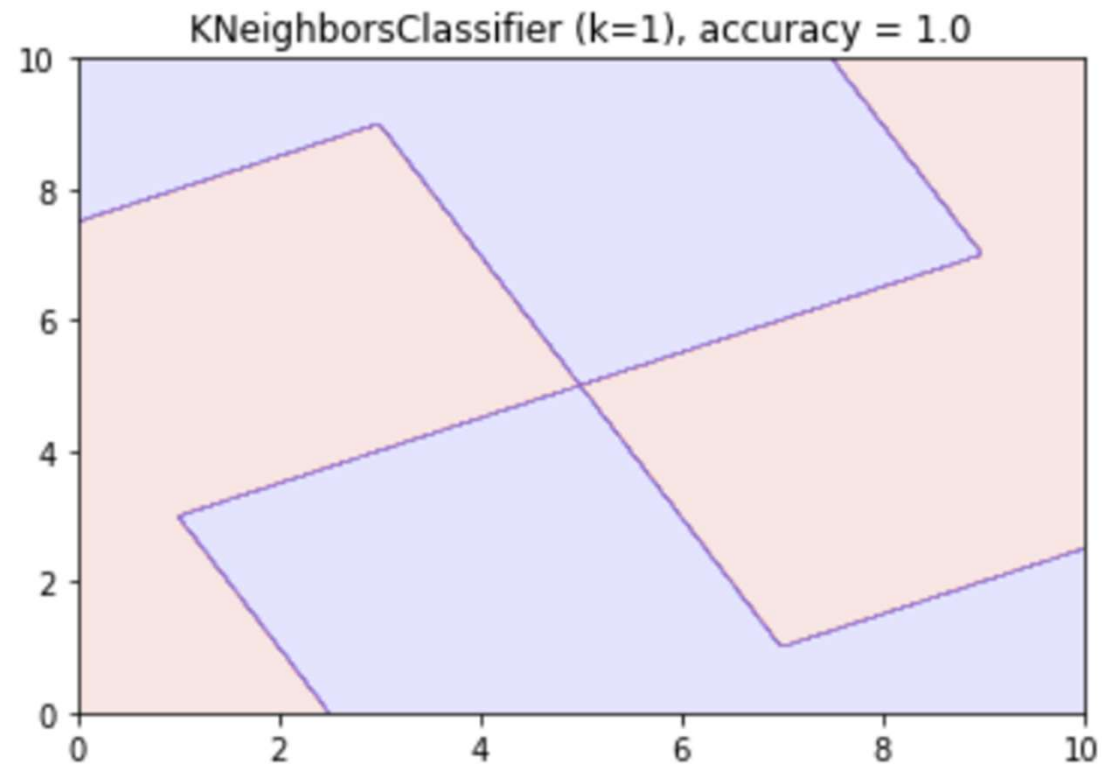


$X = [[2, 6], [4, 8], [6, 2], [8, 4]]$   
 $y = [0, 1, 1, 0]$



$X = [[2, 6], [4, 8], [6, 2], [8, 4], [2, 2], [9, 8]]$   
 $y = [0, 1, 1, 0, 1, 0]$

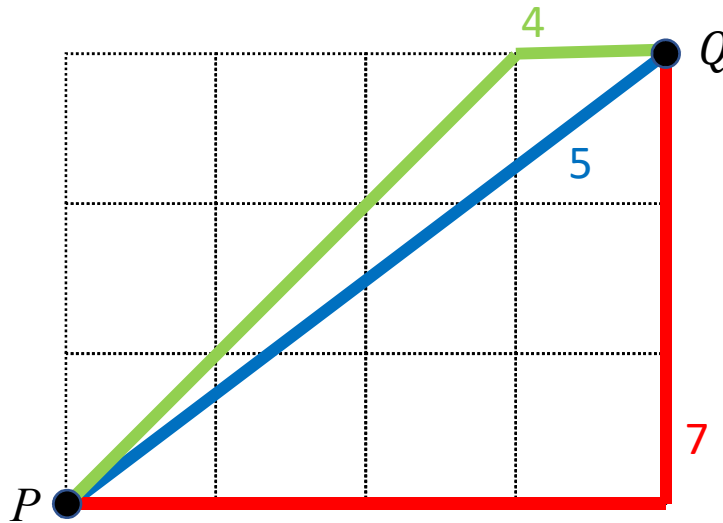
解答例（もちろん別解が無数に存在します）



$X = [[0, 0], [10, 0], [10, 10], [0, 10], [2, 6], [4, 2], [8, 4], [6, 8]]$

$y = [0, 1, 0, 1, 0, 1, 0, 1]$

# 距離の測り方



ユークリッド距離 ( $L_2$  距離)

$$d_2(P, Q) = \sqrt{\sum_{i=1}^n (q_i - p_i)^2}$$

直線距離 5

マンハッタン距離 ( $L_1$  距離)

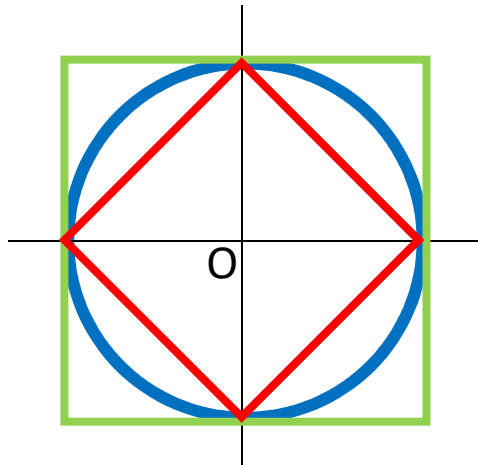
$$d_1(P, Q) = \sum_{i=1}^n |q_i - p_i|$$

縦・横 7

チェビシェフ距離 ( $L_\infty$  距離)

$$d_\infty(P, Q) = \max_{1 \leq i \leq n} \{|q_i - p_i|\}$$

縦・横・斜め 4



ミンコフスキー距離 ( $L_p$  距離)

$$d_p(P, Q) = (\sum_{i=1}^n |q_i - p_i|^p)^{\frac{1}{p}}$$

「円」: 定点 O から距離が等しい点の集合

ちょっと関連: 「スーパー楕円」

# 他にも「距離」はいろいろある.

- マハラノビス距離 (Mahalanobis Distance)  
各軸に関する分散を考慮に入れて補正したもの.
- 文字列の近さを比較するための  
編集距離 (Edit Distance) など

「距離」をうまく定義すれば k近傍法が使える.

# 2019年6月28日の目標

今後も毎回行ってください

- 出席名簿に自筆で署名する.
- ISTU の難易度アンケート, ミニットペーパー
- 機械学習アルゴリズム
  - 決定木 (DecisionTreeClassifier) <http://bit.ly/2EUVgqA>
  - 構築アルゴリズムについて

残りは3回

- 7月5日(金)
- 7月12日(金)は休講
- 7月19日(金)
- 7月26日(金)

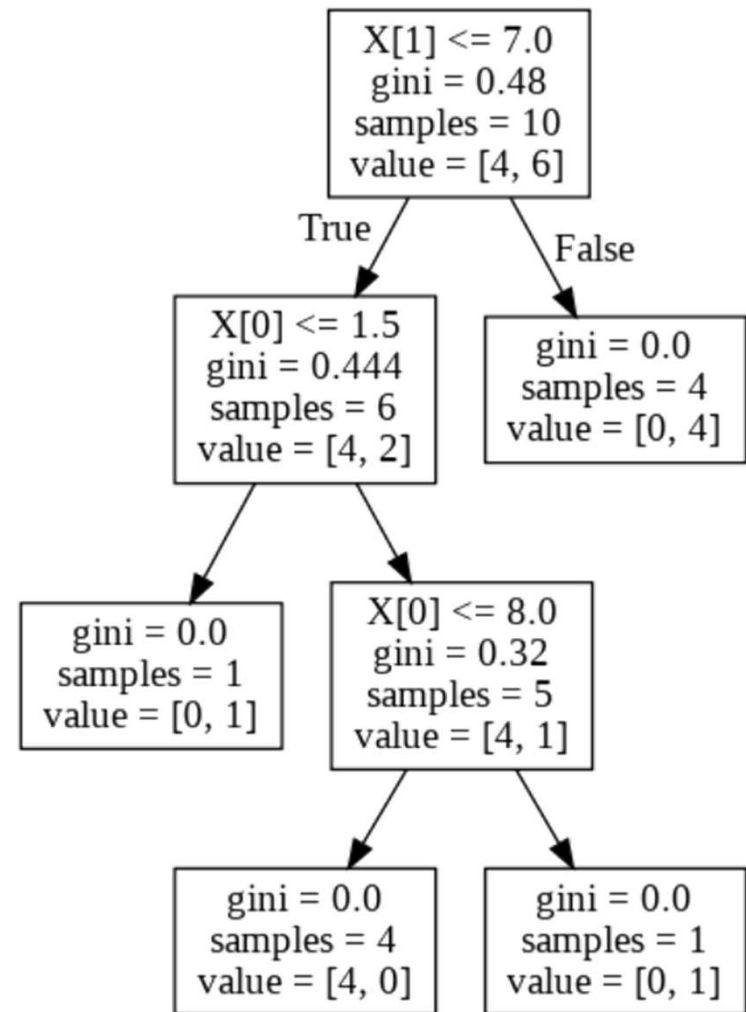
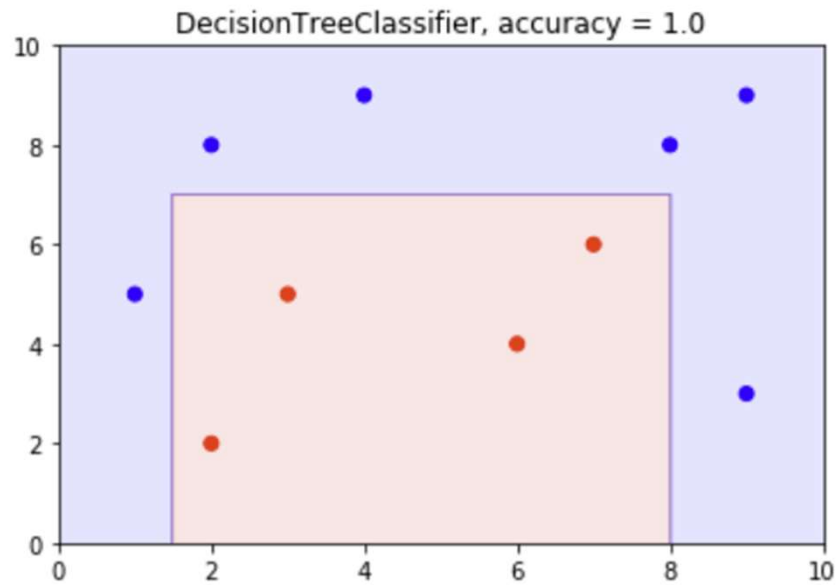
# 前回6月21日の難易度・理解度

あなたにとって、本日のゼミの難易度・理解度は？

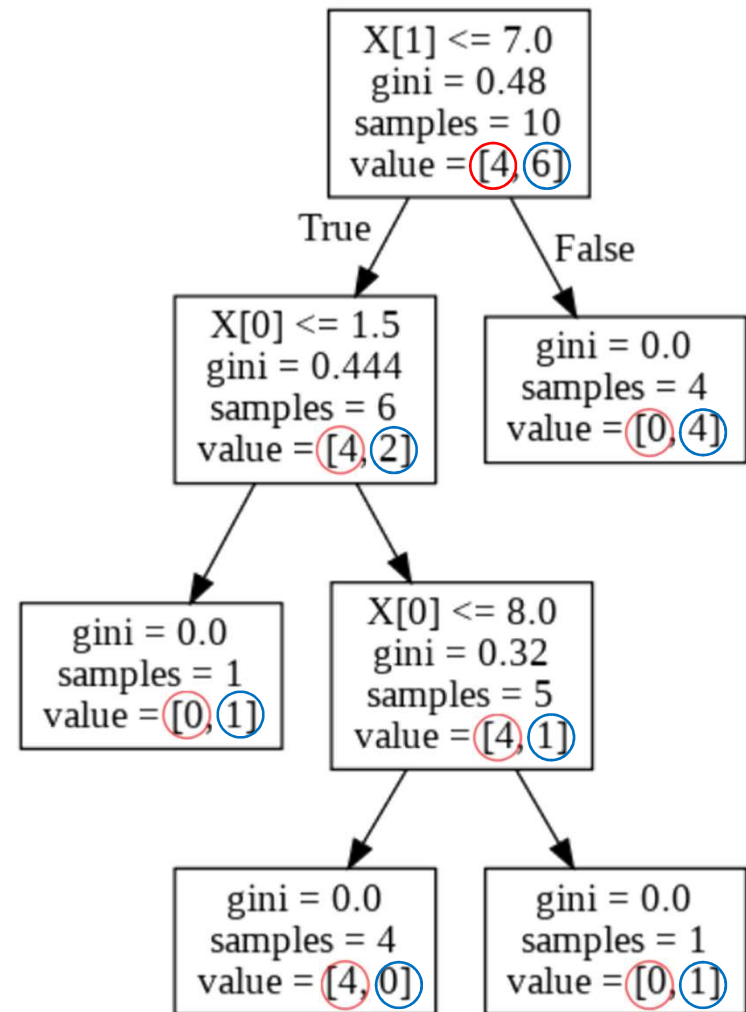
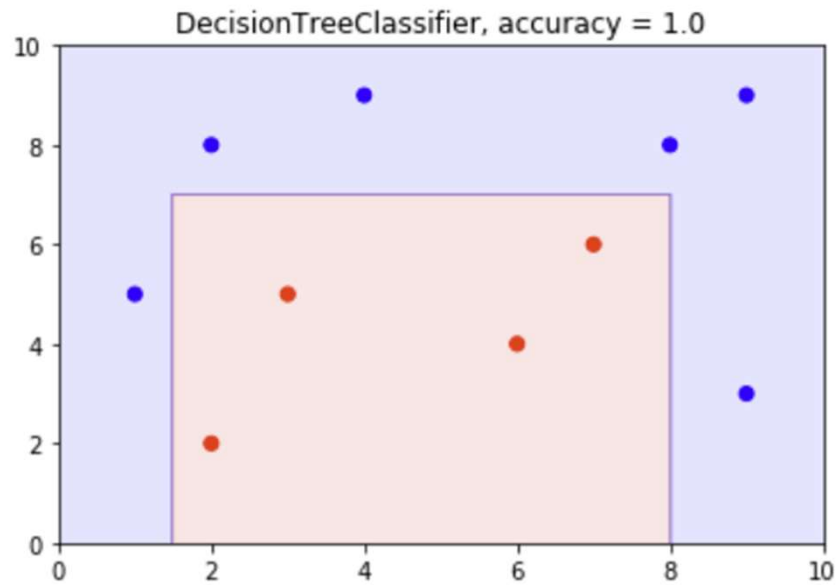
	回答数	割合
易しすぎ，知っていることばかりだった	0	0%
やや易しかった	4	8%
ちょうどよかった	29	61%
やや難しかった	14	29%
難しすぎ，ほとんどついていけなかった	1	2%



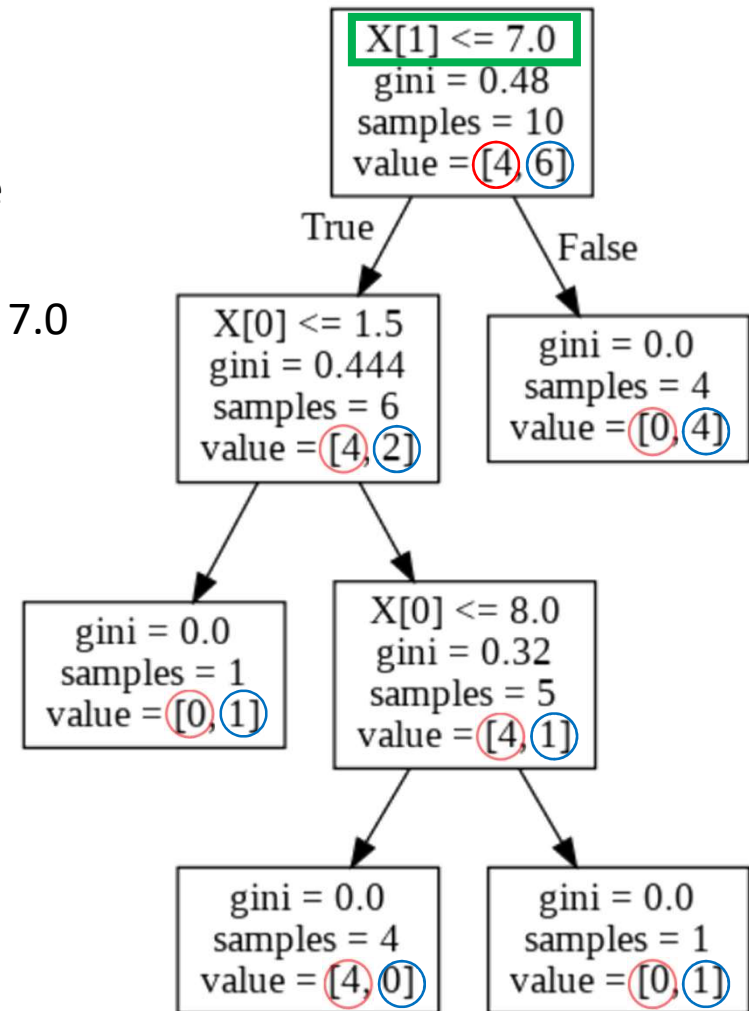
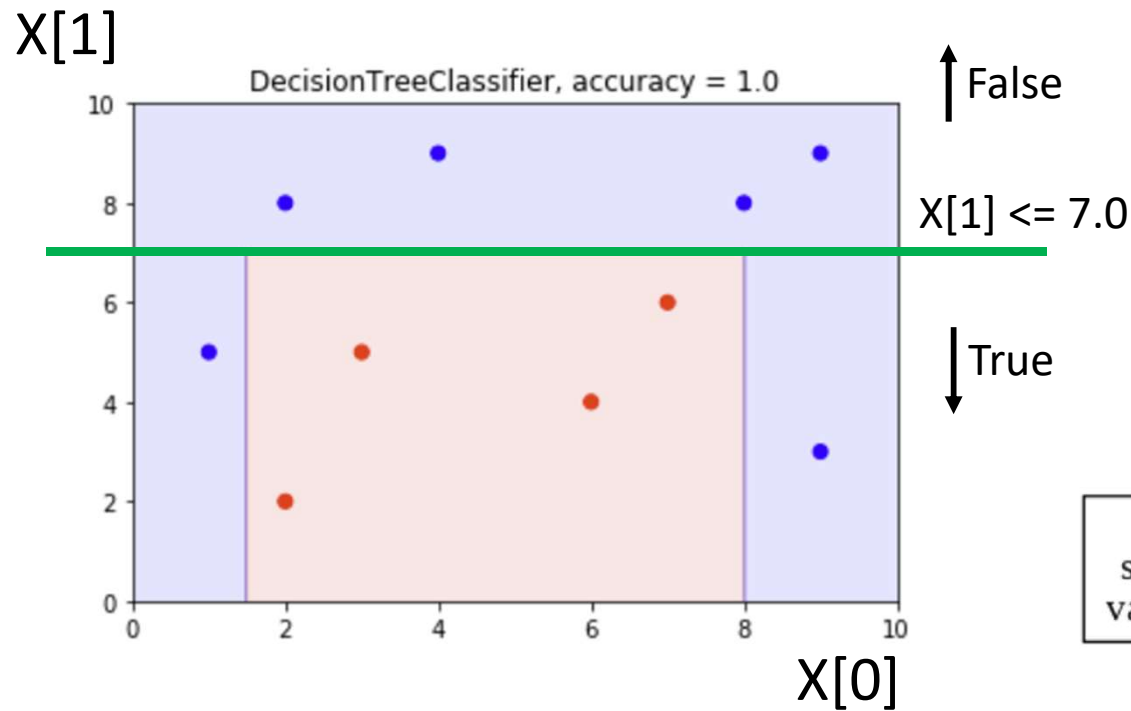
# 決定木



# 決定木



# 決定木



# Iris データの形が少しわかってきた.

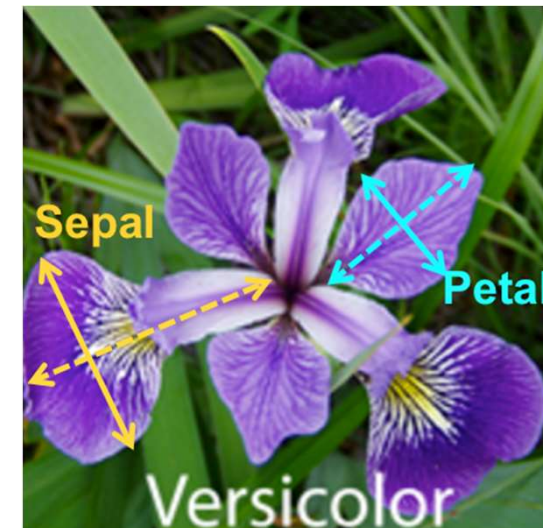
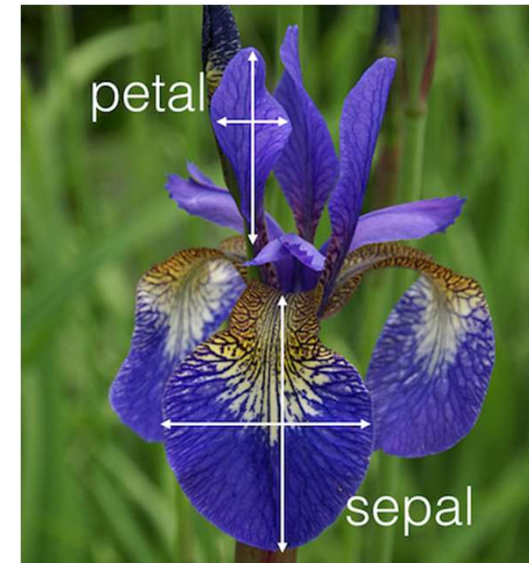
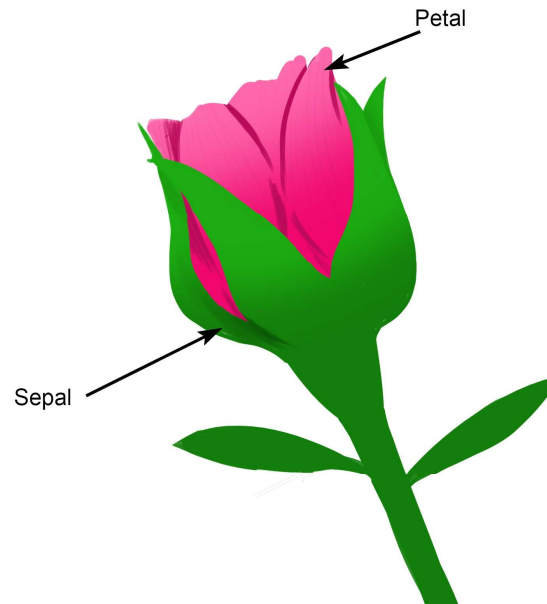
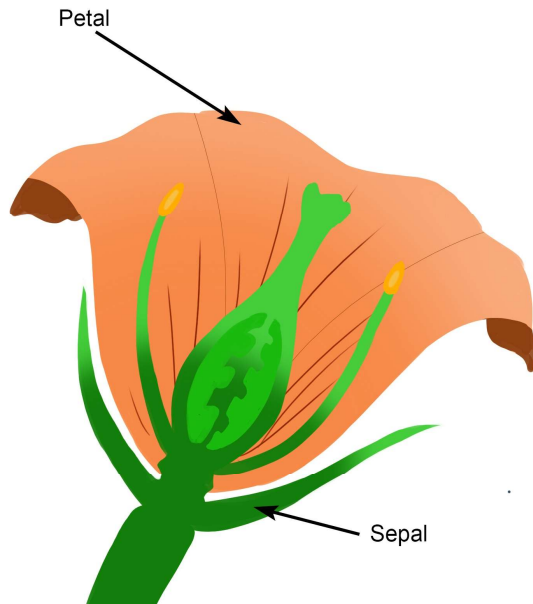
feature\_names

1	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	クラス	target_names
0	5.1	3.5	1.4	0.2	0	= setosa
1	4.9	3.0	1.4	0.2	0	再掲
2	4.7	3.2	1.3	0.2	0	
3	4.6	3.1	1.5	0.2	0	
4	5.0	3.6	1.4	0.2	0	
5	5.4	3.9	1.7	0.4	0	
6	4.6	3.4	1.4	0.3	0	= virginica
⋮	⋮	⋮	⋮	⋮	⋮	
	5.9	3.0	5.1	1.8	2	

data

target

# Sepal (萼片<sup>がくへん</sup>) と Petal (花弁) 再掲



<https://www.quora.com/What-is-the-difference-between-sepals-and-petals>

<https://melindahiggins2000.github.io/N741UnsupervisedLearning/UnsupervisedLearning.html>

[https://holgerbrandl.github.io/kotlin4ds\\_kotlin\\_night\\_frankfurt//krangl\\_example\\_report.html](https://holgerbrandl.github.io/kotlin4ds_kotlin_night_frankfurt//krangl_example_report.html)

# 2019年7月5日の目標

今後も毎回行ってください

- 出席名簿に自筆で署名する.
- ISTU の難易度アンケート, ミニットペーパー

## • 機械学習アルゴリズム

✓ 最短近傍法の実装例 <http://bit.ly/30cvrKR>

✓ 決定木

▶ サポートベクトルマシン <http://bit.ly/2RO9yi7>

- アンサンブル法, ランダムフォレスト
- ニューラルネットワーク

残りは2回

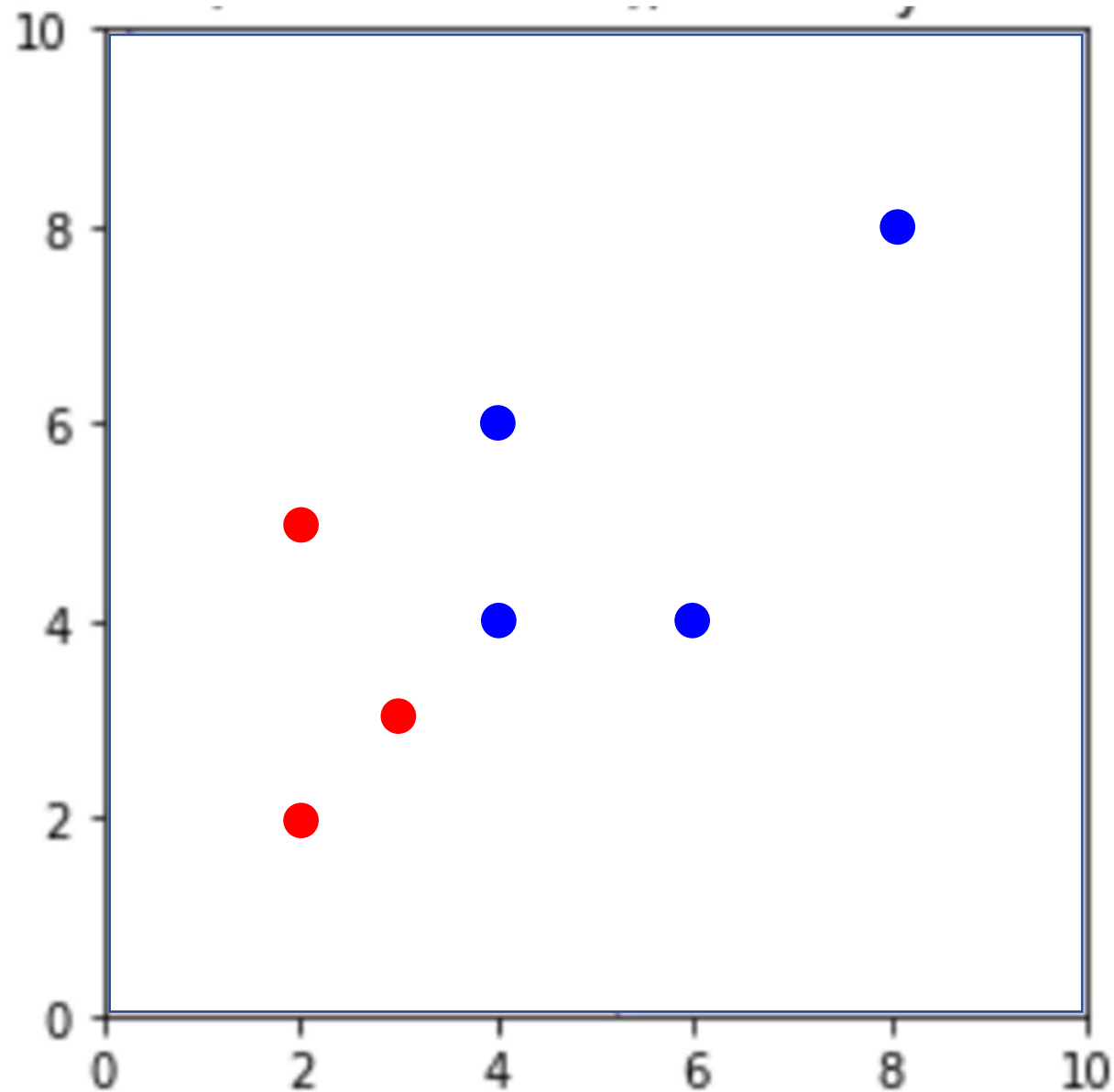
- 7月12日(金)は休講
- 7月19日(金)
- 7月26日(金)

# 前回6月28日の難易度・理解度

あなたにとって、本日のゼミの難易度・理解度は？

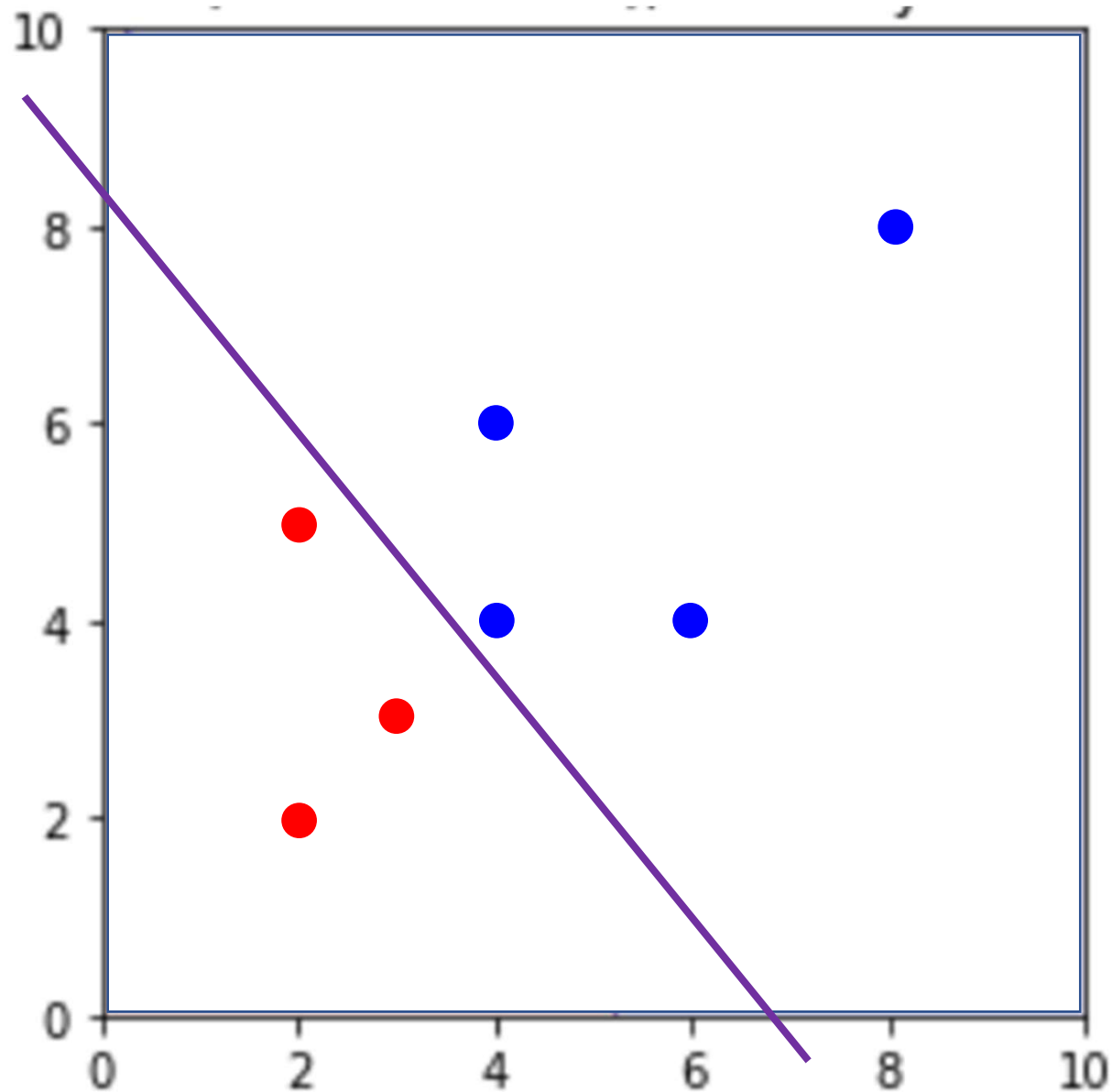
	回答数	割合
易しすぎ、知っていることばかりだった	1	3%
やや易しかった	0	0%
ちょうどよかった	19	46%
やや難しかった	18	44%
難しすぎ、ほとんどついていけなかった	3	7%

# ハードマージン（線形分離可能な場合）

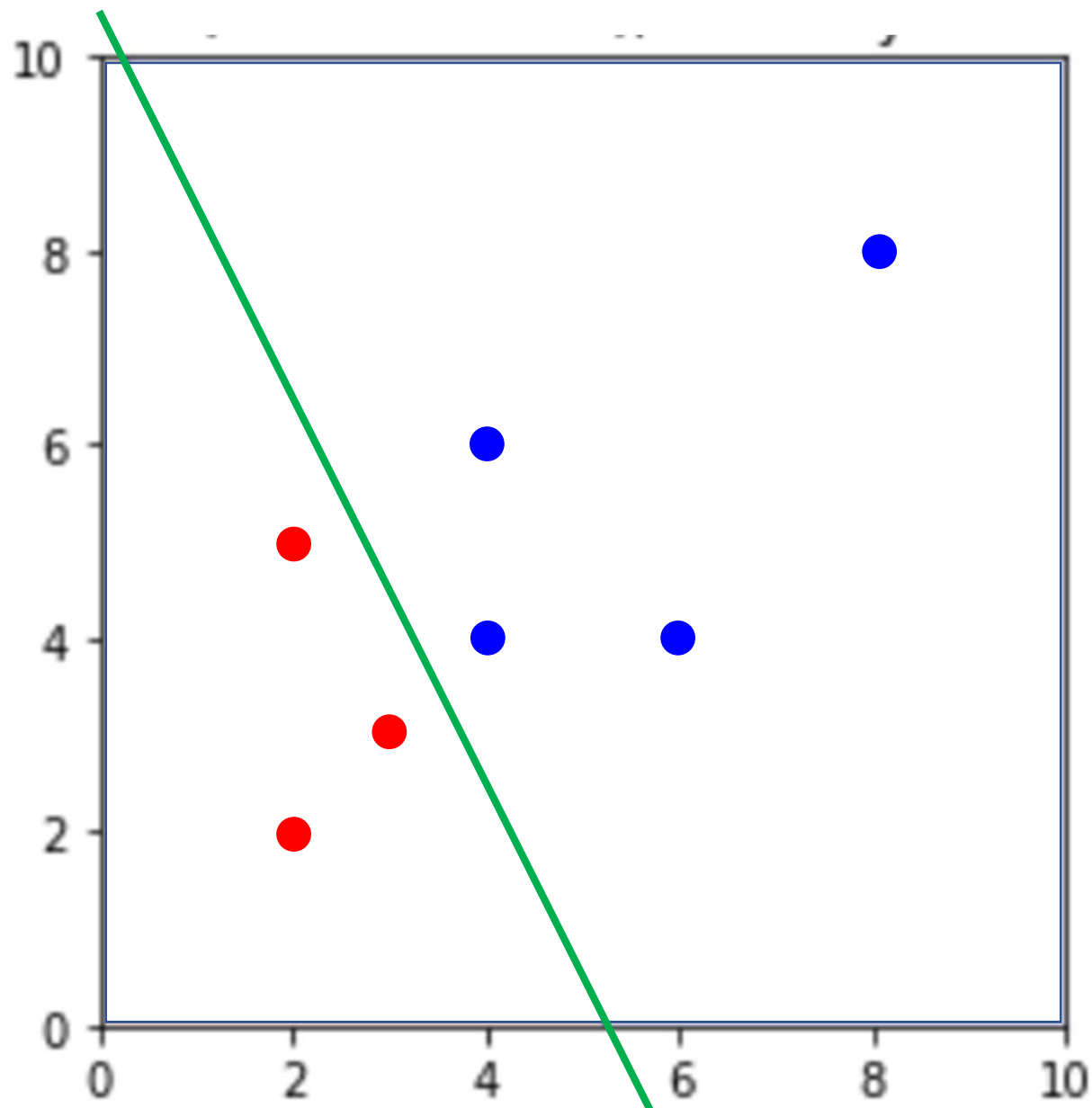




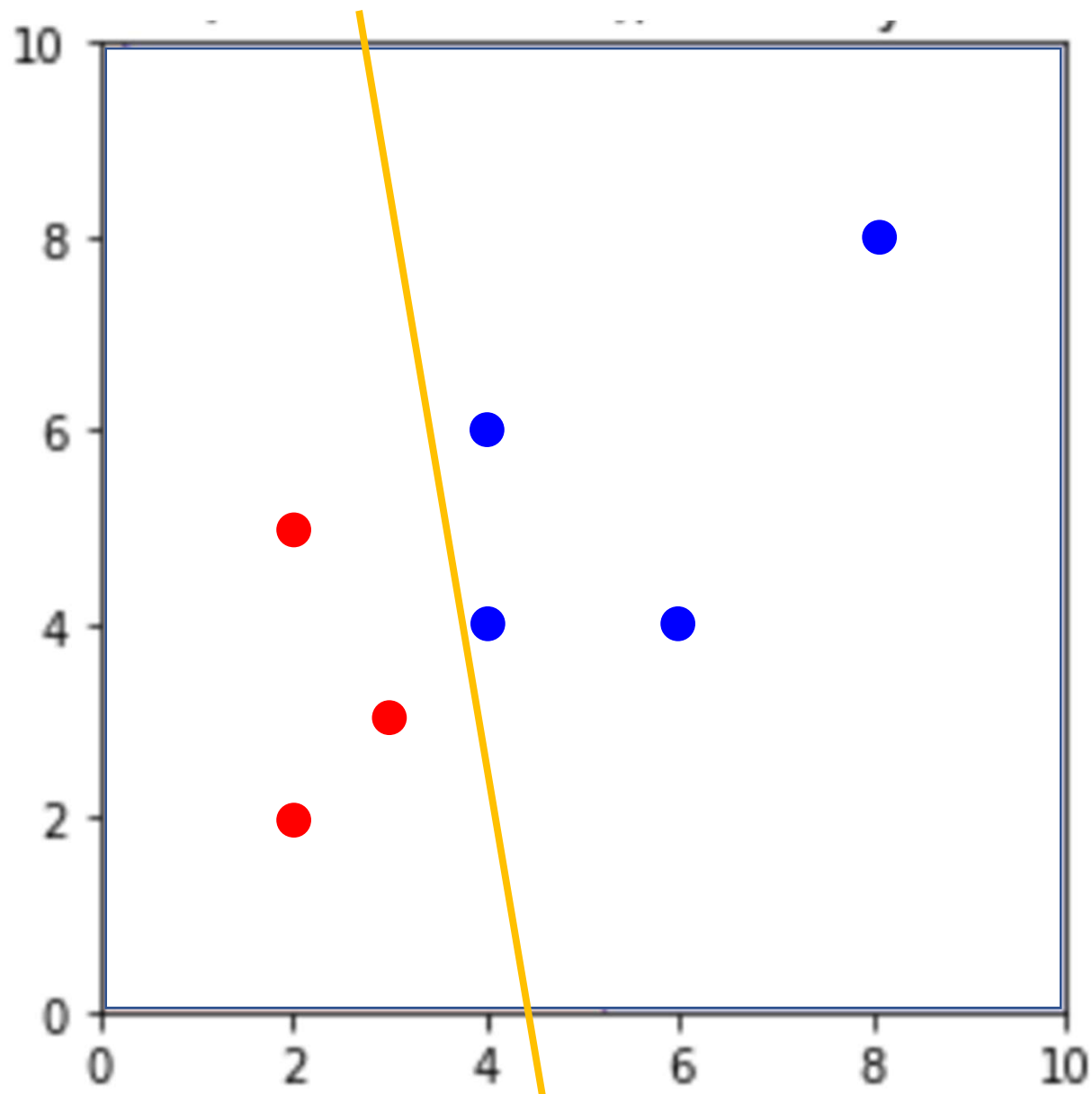
# ハードマージン（線形分離可能な場合）



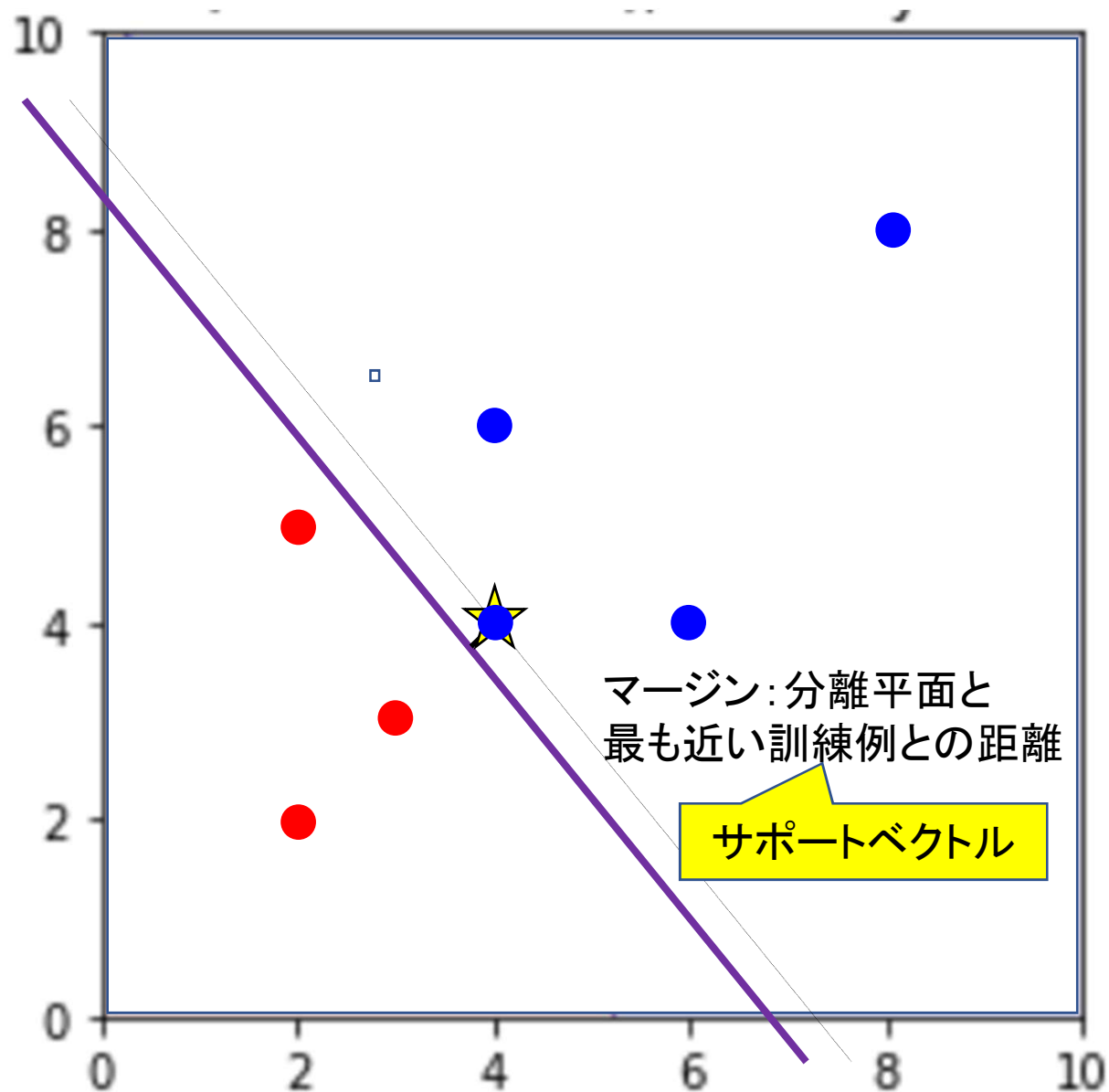
# ハードマージン（線形分離可能な場合）



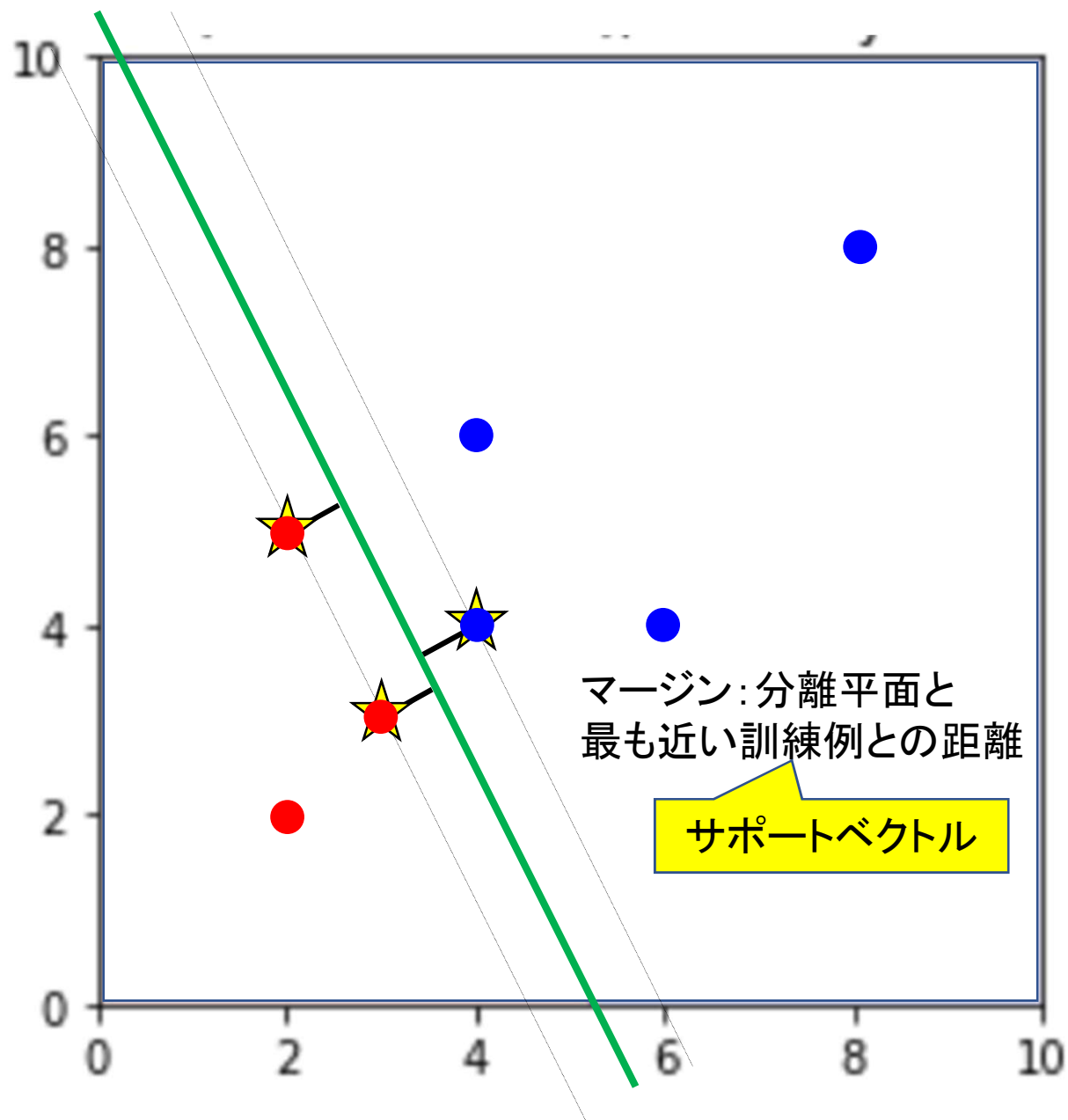
# ハードマージン（線形分離可能な場合）



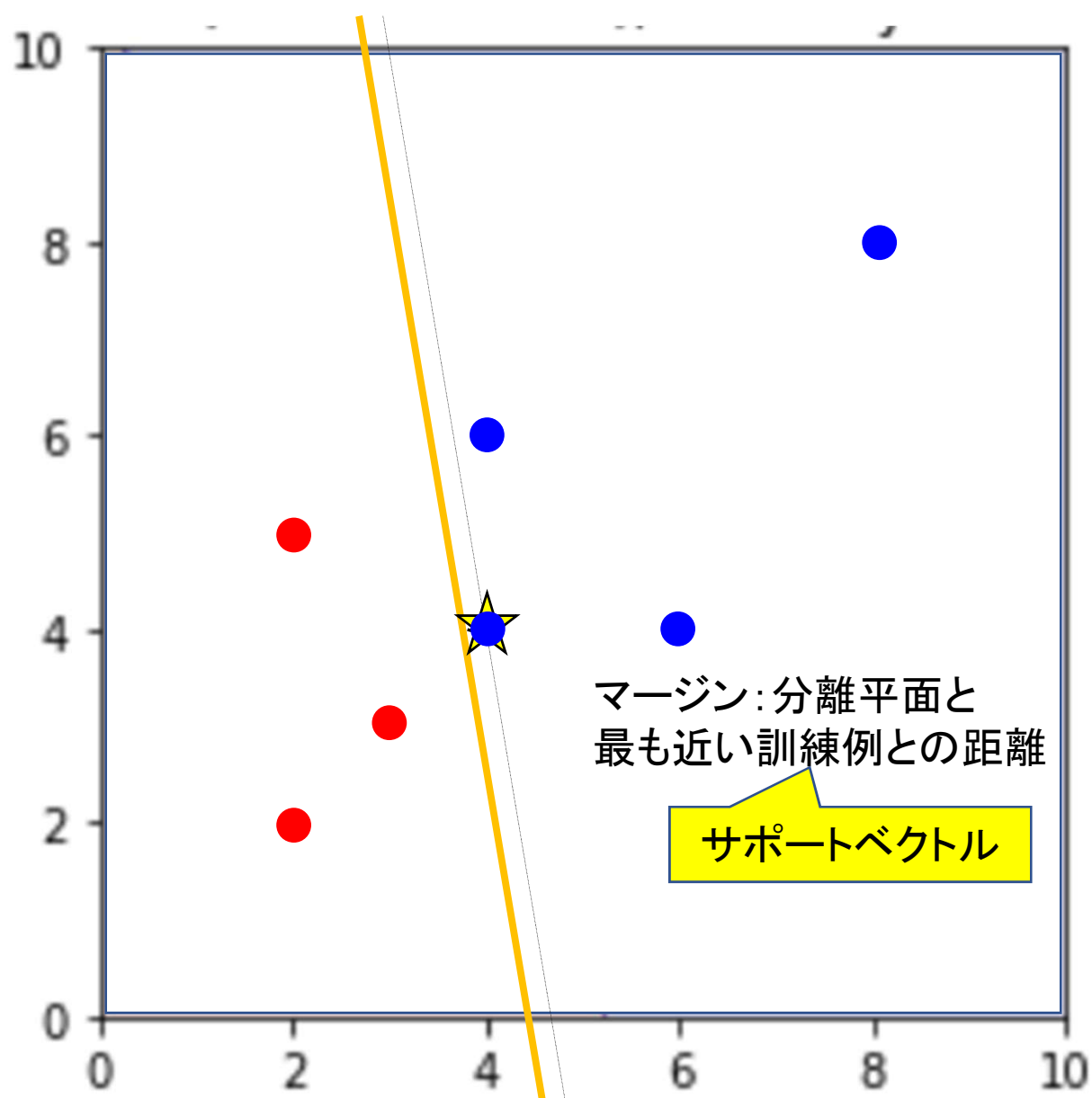
# ハードマージン（線形分離可能な場合）



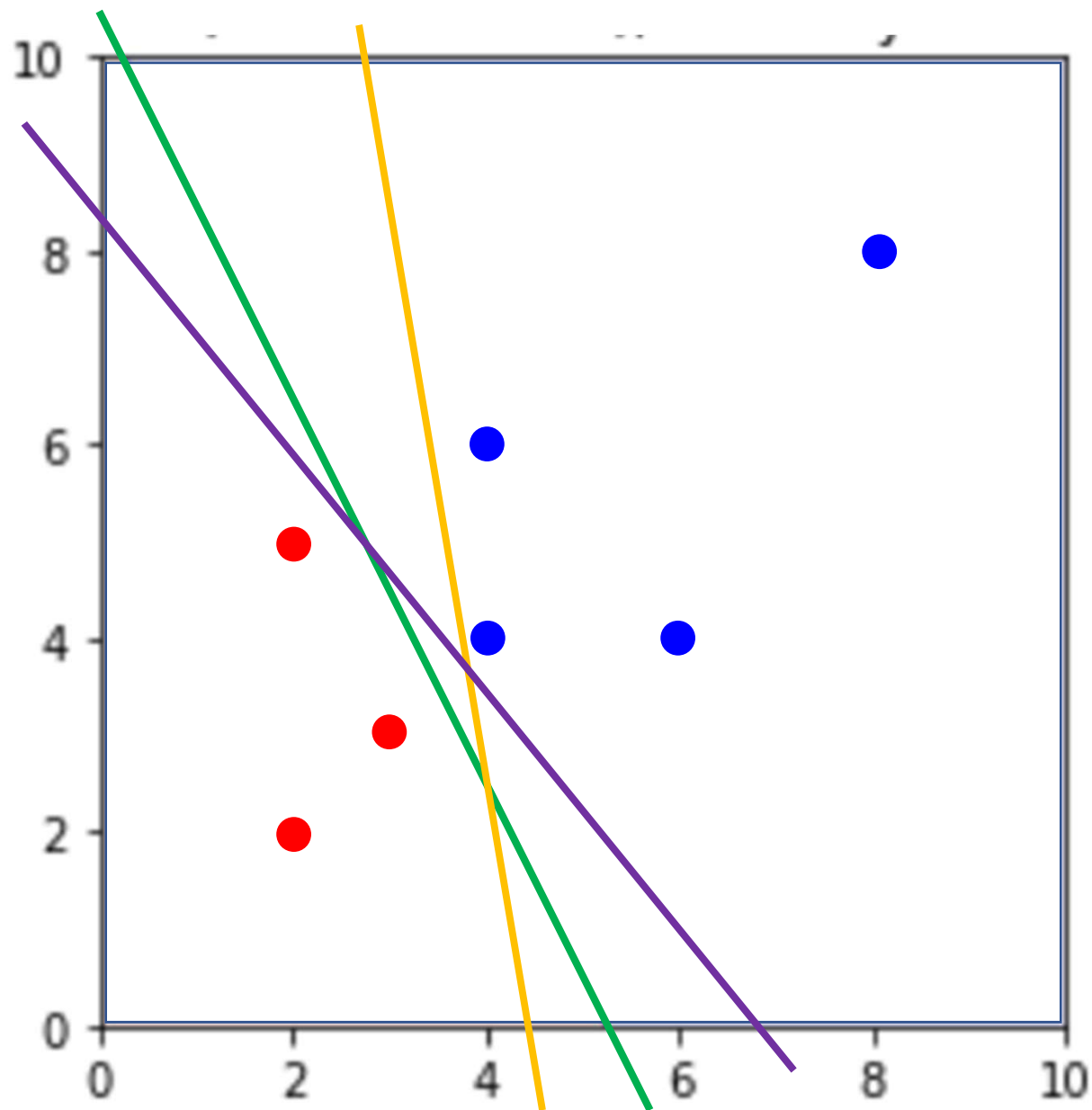
# ハードマージン（線形分離可能な場合）



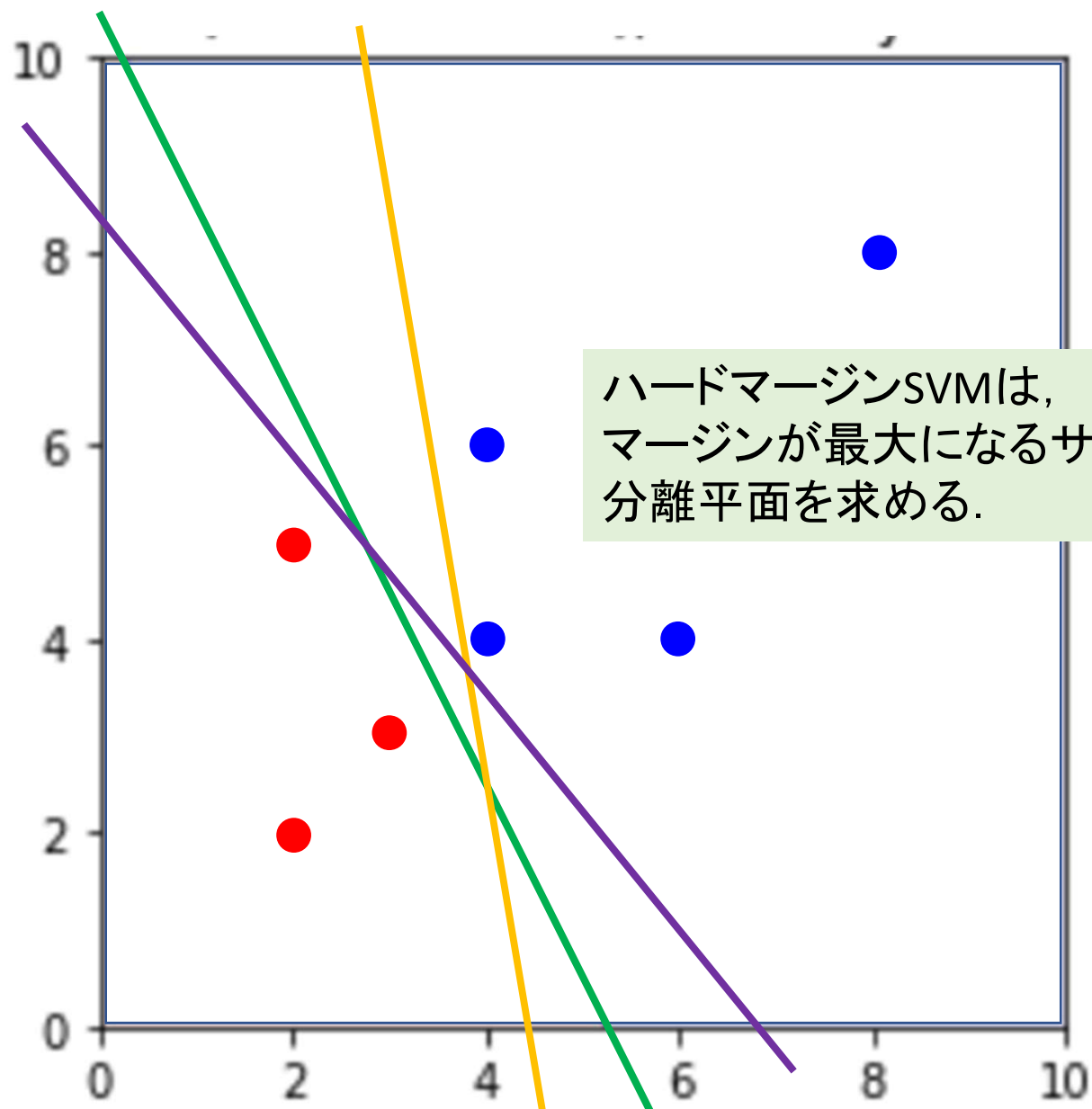
# ハードマージン（線形分離可能な場合）



# ハードマージン（線形分離可能な場合）



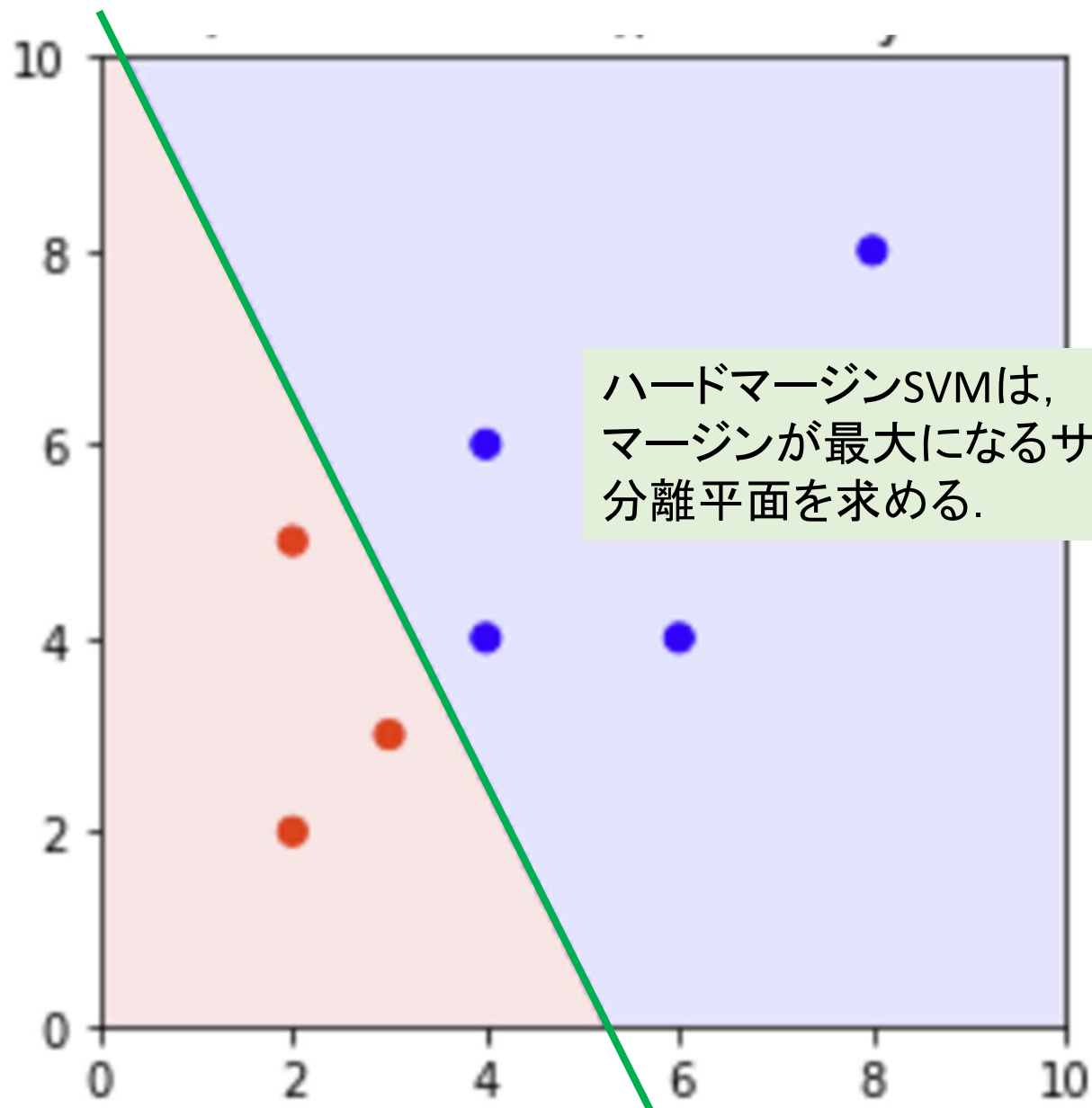
# ハードマージン（線形分離可能な場合）



ハードマージンSVMは、  
マージンが最大になるサポートベクトルと  
分離平面を求める。

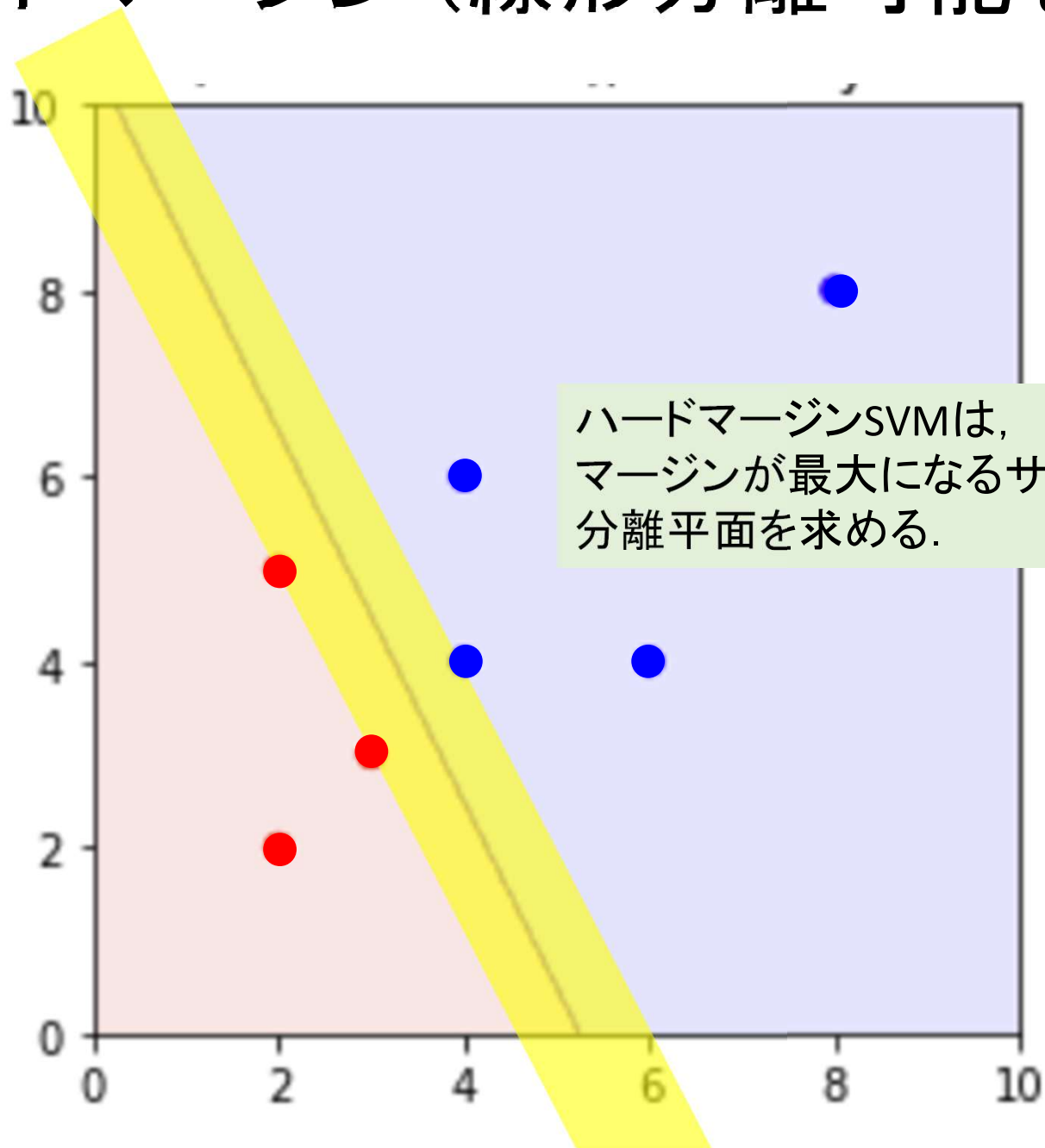


# ハードマージン（線形分離可能な場合）



ハードマージンSVMは、  
マージンが最大になるサポートベクトルと  
分離平面を求める。

# ハードマージン（線形分離可能な場合）



ハードマージンSVMは、  
マージンが最大になるサポートベクトルと  
分離平面を求める。

# 点と直線の距離の公式を思い出そう

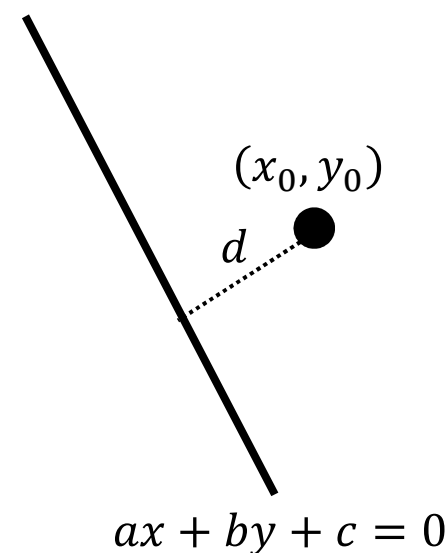
- 点  $(x_0, y_0)$  と 直線  $ax + by + c = 0$  の距離  $d$

$$d = \frac{|a x_0 + b y_0 + c|}{\sqrt{a^2 + b^2}}$$

$$\mathbf{w} = (a, b)$$

$\mathbf{x} = (x, y)$  とすると

$$d = \frac{|\overset{\text{内積}}{\mathbf{w}} \cdot \mathbf{x} + c|}{\|\mathbf{w}\|}$$



2次元以上でも成り立つ

# 点と直線の距離の公式を思い出そう

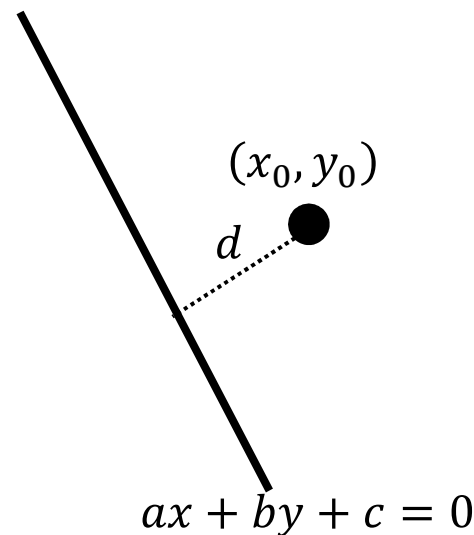
- 点  $(x_0, y_0)$  と直線  $ax + by + c = 0$  の距離  $d$

$$d = \frac{|a x_0 + b y_0 + c|}{\sqrt{a^2 + b^2}}$$

$$\mathbf{w} = (a, b)$$

$\mathbf{x} = (x, y)$  とすると

$$d = \frac{|\mathbf{w} \cdot \mathbf{x} + c|}{\|\mathbf{w}\|}$$



これは2次元以上でも成り立つ

$\mathbf{w}$  の定数倍は分子・分母で相殺されるので、境界面に一番近い点で  $|\mathbf{w} \cdot \mathbf{x} + c| = 1$  が成り立つと仮定できる。

そうするとマージンを最大化する式は  $\frac{1}{\|\mathbf{w}\|}$  を最大化することになる。

つまり訓練例  $\{(x_i, y_i)\}$  に対して ( $y_i \in \{-1, +1\}$ )

$y_i (\mathbf{w} \cdot \mathbf{x}_i + c) \geq 1$  の条件下で  $\frac{1}{\|\mathbf{w}\|}$  を最大にする  $\mathbf{w}$  と  $c$  を求める問題になる。

$y_i (\mathbf{w} \cdot \mathbf{x}_i + c) \geq 1$  の条件下で  $\frac{1}{\|\mathbf{w}\|}$  を最大にする  $\mathbf{w}$  と  $c$  を求める問題は,

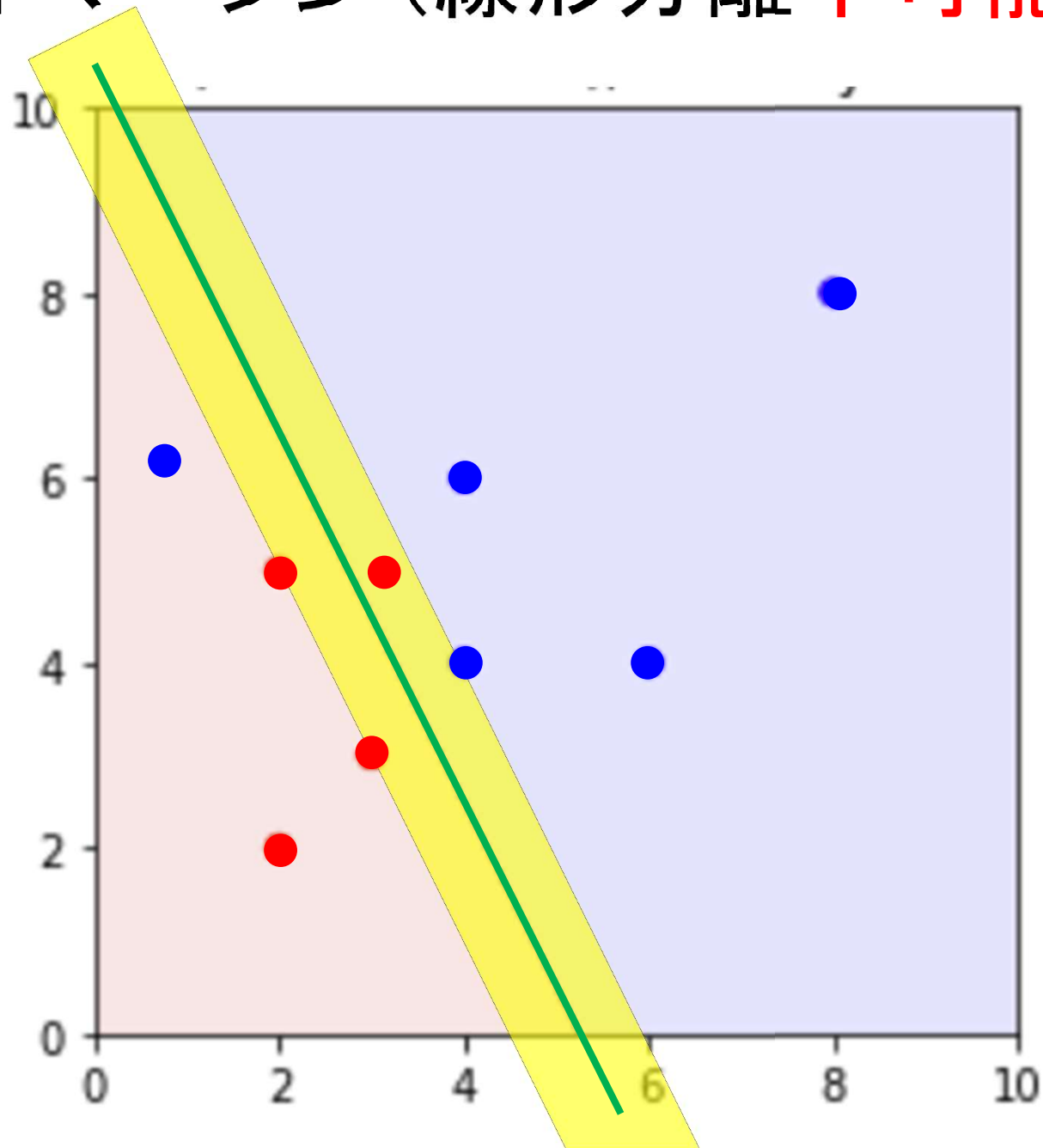
同じ条件下で  $\frac{1}{2} \|\mathbf{w}\|^2$  を最小化する問題に置き換えられる.

この問題を, ラグランジュ未定乗数法を使って

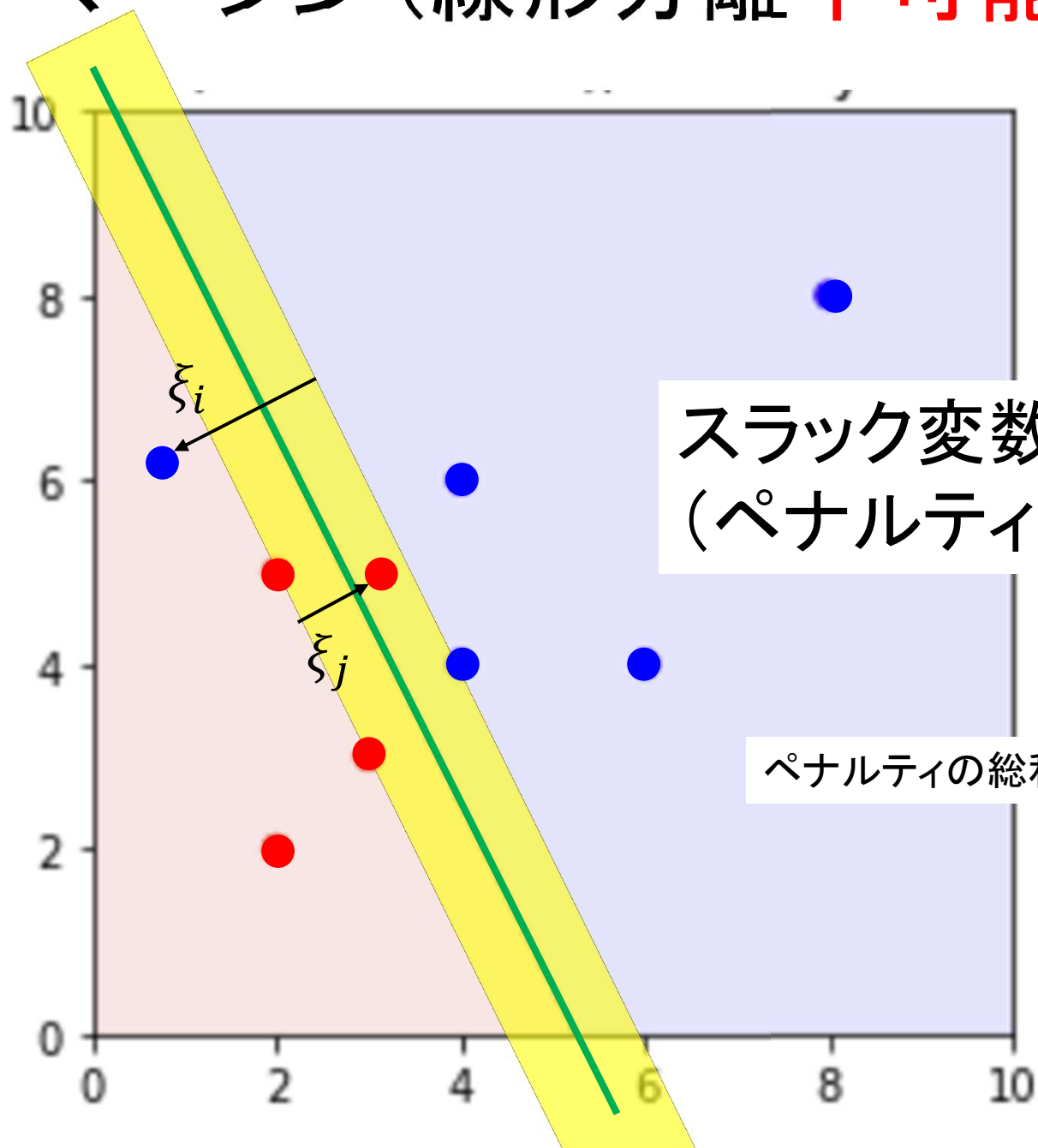
$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

を最大化する係数  $\alpha = (\alpha_1, \dots, \alpha_n)$ ,  $\alpha \geq 0$  を求める問題に置き換えることができる.

# ソフトマージン（線形分離不可能な場合）



# ソフトマージン（線形分離不可能な場合）



スラック変数  $\xi_i$   
(ペナルティ)を導入

ペナルティの総和を小さく抑える

# ソフトマージン（線形分離不可能な場合）

$y_i (\mathbf{w} \cdot \mathbf{x}_i + c) \geq 1 - \xi_i, \quad \xi_i \geq 0$  の条件下で

$\frac{1}{2} \|\mathbf{w}\|^2 + \underbrace{C \sum_i \xi_i}_{\text{正則化のパラメータ}}$  を最小化する問題に置き換えられる。

この問題も、ラグランジュ未定乗数法を使って

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

を最大化する係数  $\alpha = (\alpha_1, \dots, \alpha_n)$ ,  $\alpha_i \geq 0$ ,  $\sum_i \alpha_i y_i = 0$  を求める問題に置き換えることができる。



# カーネルトリック

$$\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \boxed{\mathbf{x}_i \cdot \mathbf{x}_j}$$

$\mathbf{x} = (x_1, x_2)$ ,  $\mathbf{y} = (y_1, y_2)$  に対して

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}) &= (1 + \mathbf{x} \cdot \mathbf{y})^2 && \text{多項式カーネル} \\ &= (1 + x_1 y_1 + x_2 y_2)^2 \\ &= 1 + x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 y_1 + 2 x_2 y_2 + 2 x_1 x_2 y_1 y_2 \\ &= (1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2) \cdot (1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1 y_2) \end{aligned}$$

# 2019年7月19日の目標

今後も毎回行ってください

- 出席名簿に自筆で署名する.
- ISTU の難易度アンケート, ミニットペーパー
- 機械学習アルゴリズム
  - ✓ 最短近傍法の実装例
  - ✓ 決定木
  - ✓ サポートベクトルマシン
  - アンサンブル法, ランダムフォレスト <http://bit.ly/2JDUIs7>
    - ニューラルネットワーク

残りは1回  
• 7月26日(金)

# 7月5日の撮影協力ありがとうございました

<https://www3.nhk.or.jp/tohoku-news/20190719/6000006253.html>

東北 NEWS WEB

(1週間ぐらいで消えると思うので視聴はお早めに！)

## 東北大学 AI 授業を必修に

07月19日 09時50分



東北大学は、A I = 人工知能の活用が幅広い分野に広がっていることから、その仕組みや理論について学ぶ授業を来年度の新入生から文系・理系を問わずすべての学部で必修とすることに

しています。

東北大学では、A I に関する知識が次世代のリーダーには欠かせないとして、工学部や農学部、それに生物学科を除く理学部のすべての学科などで A I の仕組みや理論に関する授業を必修としていたほか、そのほかの学部でも希望に応じて受講できるようにしていました。

しかし、A I の活用が言語の研究や医療現場など幅広い分野に広がっていることから、来年度の新入生から文系・理系を問わずすべての学部で必修とすることにしています。

A I に関する授業は、大学1年生から2年生までの一般教養科目の一部として導入され、学生たちは、A I の開発に不可欠なプログラミングの基礎知識や理論、それにデータの処理方法などを学ぶということです。

東北大学で教育・学生支援を担当する滝澤博胤理事は、「実際に A I の開発まで携わるのは一部の学生だと思うが、その仕組みを知ること、学生が自ら将来の仕事や研究などで A I をどう使うことができるか考えるための土台を提供できると思う」と話しています。

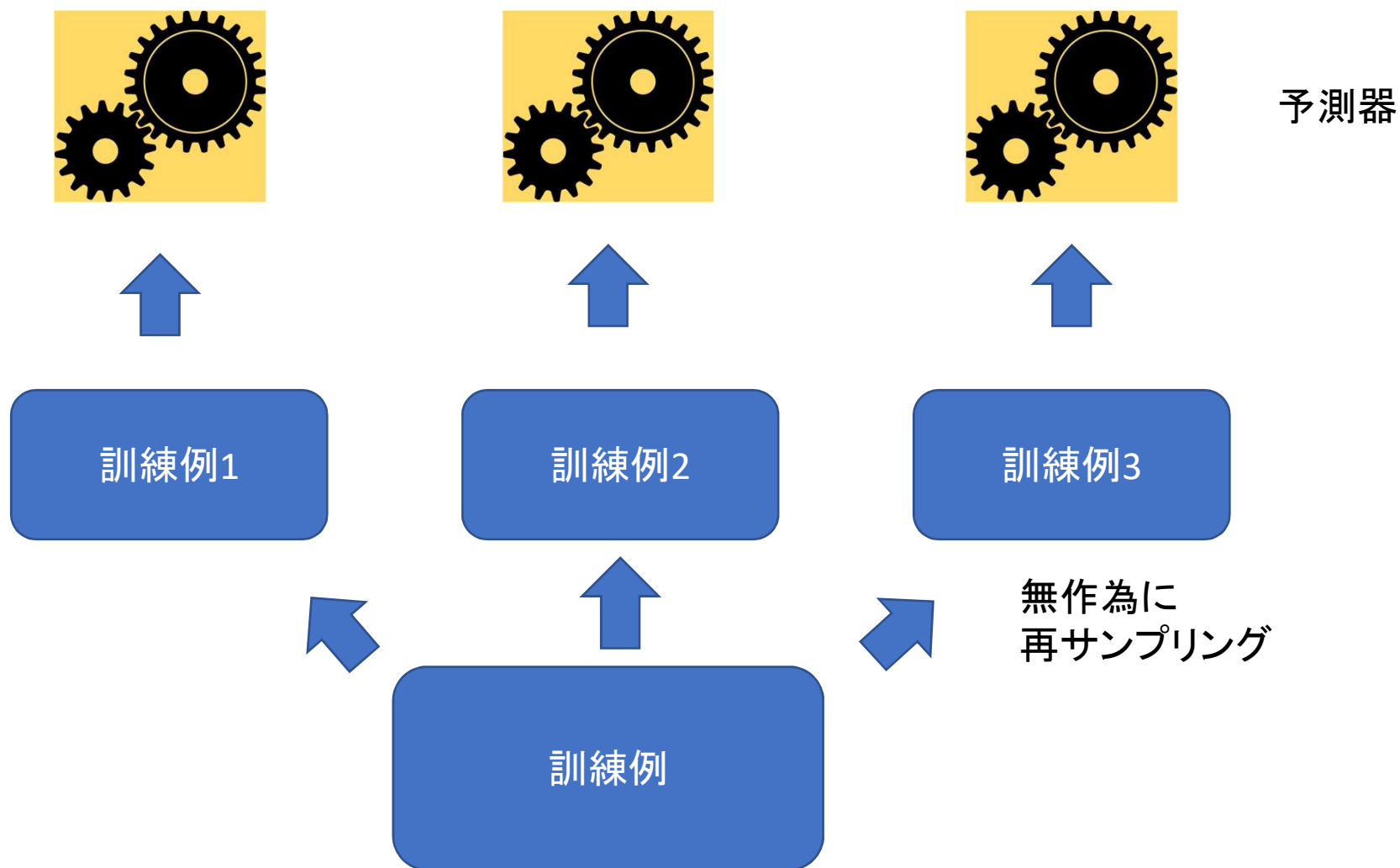
今朝の「おはよう宮城 (NHK, 7:45～)」で放送されました。

# 前回7月5日の難易度・理解度

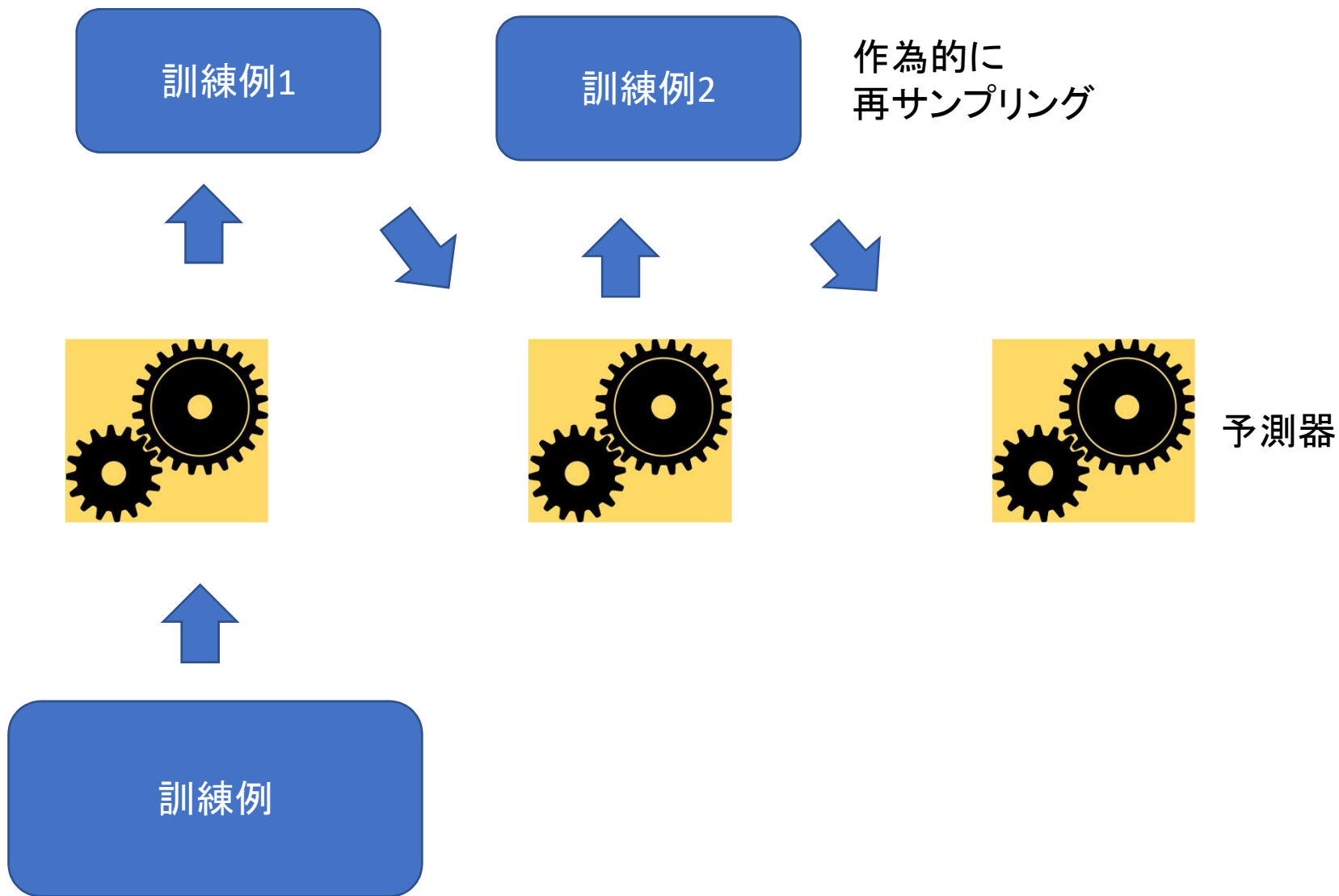
あなたにとって、本日のゼミの難易度・理解度は？

	回答数	割合
易しすぎ，知っていることばかりだった	0	0%
やや易しかった	2	5%
ちょうどよかった	16	40%
やや難しかった	19	48%
難しすぎ，ほとんどついていけなかった	3	7%

# Bagging



# Boosting



# 2019年7月26日(最終回)の目標

- 出席名簿に自筆で署名する.
- ISTU の難易度アンケート, ミニットペーパー
- 機械学習アルゴリズム
  - ✓ 最短近傍法の実装例
  - ✓ 決定木
  - ✓ サポートベクトルマシン
  - ✓ アンサンブル法, ランダムフォレスト <http://bit.ly/2JDUIs7>
  - ニューラルネットワーク <http://bit.ly/2LCwm44>
  - 深層学習のイントロ <http://bit.ly/2XX3n1x>

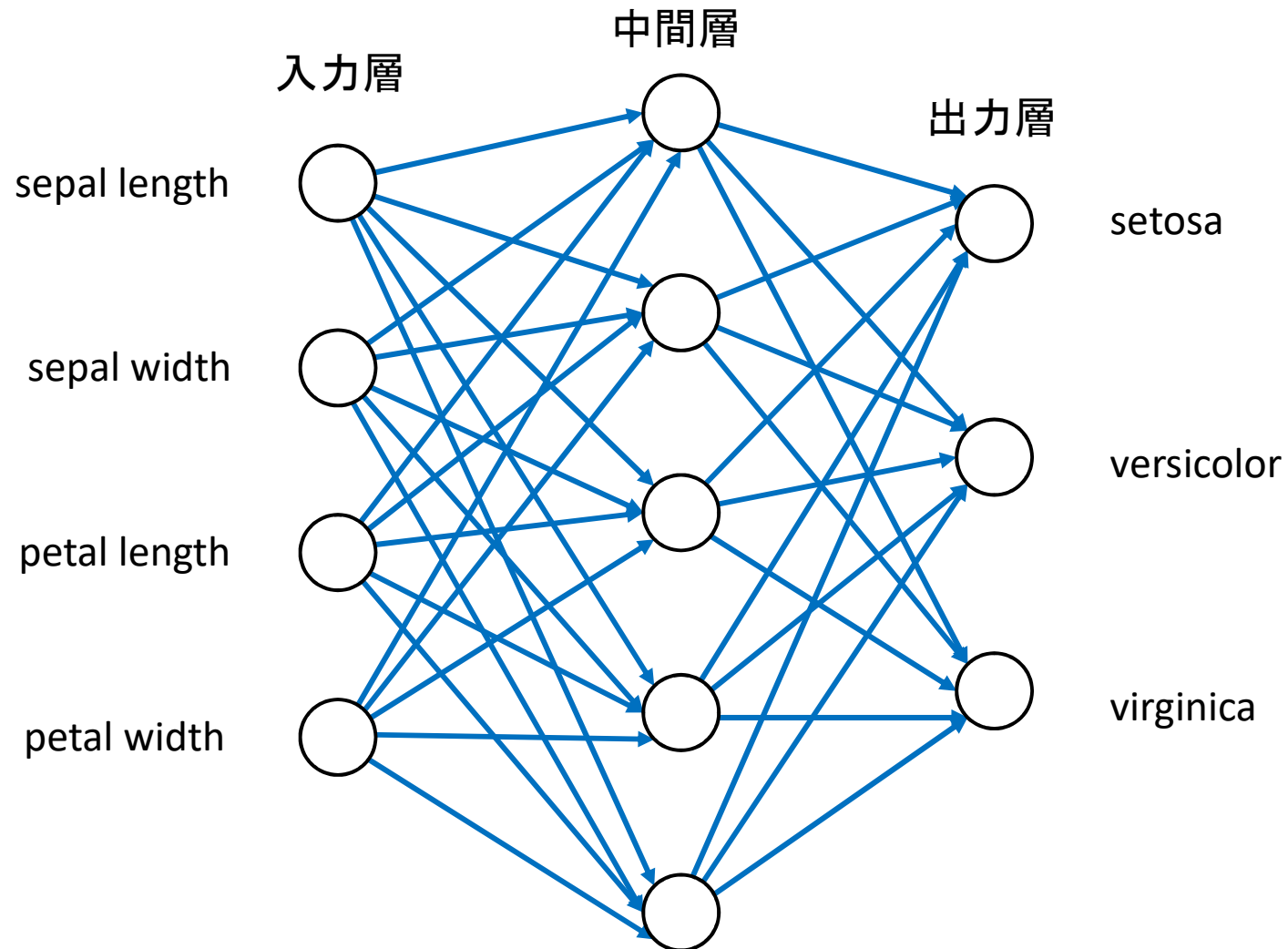
# 前回7月19日の難易度・理解度

あなたにとって、本日のゼミの難易度・理解度は？

	回答数	割合
易しすぎ，知っていることばかりだった	0	0%
やや易しかった	0	0%
ちょうどよかった	24	62%
やや難しかった	13	33%
難しすぎ，ほとんどついていけなかった	2	5%



# 2層のニューラルネットワーク



# まとめ

- Python は使いやすい.
- Scikit-learn を使えば機械学習が楽に行える.
- Numpy をうまく使えば, 高速な処理が可能になる.
- Google Colaboratory を使えば, 計算機パワーを要する作業を手軽に行い, その過程が残せる.  
また共有できる.

➡ 今後の学生生活, 研究生活で大いに活用してください.

- **A I** とか **人工知能** と呼ばれているものの  
中身が少しわかったような気がする....かな?