AI理論と実践講座

本日のAgenda

- Cutting Edge Deep Learning Application ～ Pose Estimation
- 線形回帰復習
- PCA
- (分類問題と解法)
- Python演習

  PCA

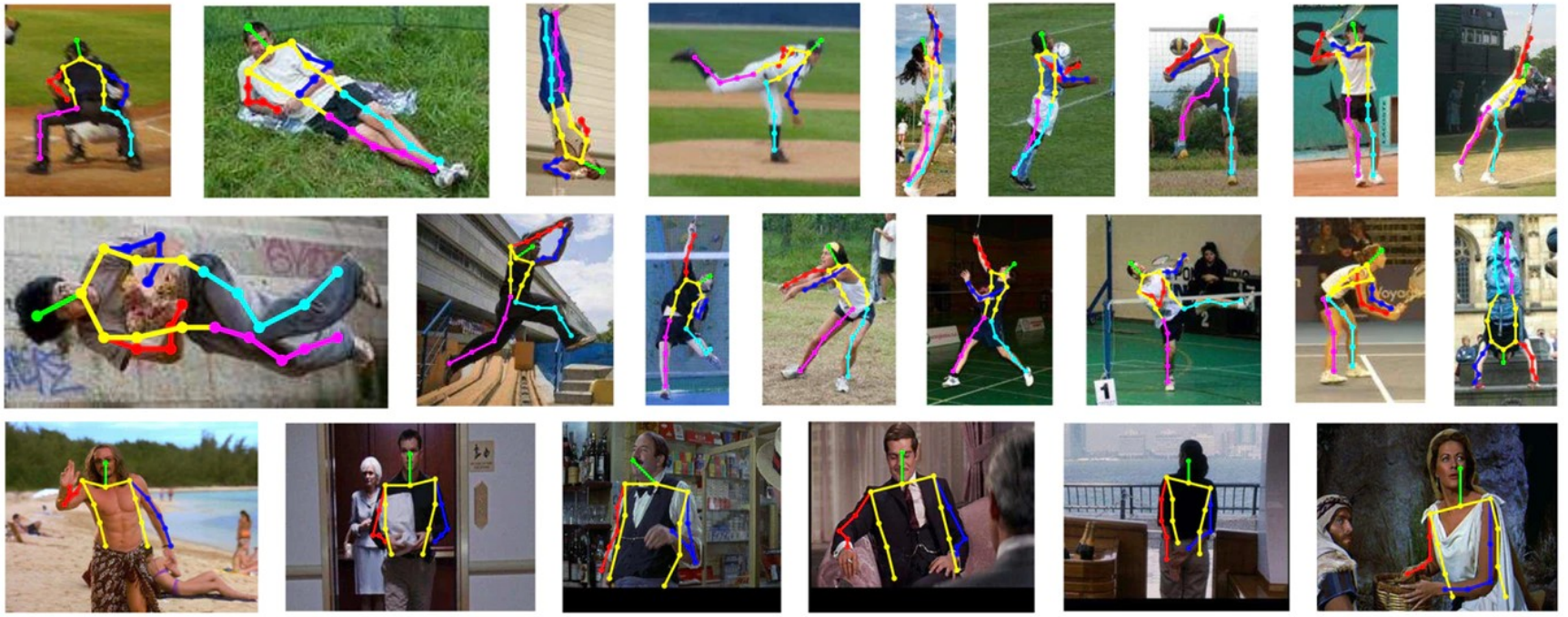
  (ロジスティック回帰、ランダムフォレスト、xGBoost)

Cutting Edge
Deep Learning Application

Human Pose Estimation (姿勢推定)

姿勢推定とは？

画像またはビデオにおける人間の関節の場所を特定するタスク



https://qiita.com/KYoshiyama/items/9f5f5a13f957e138380b

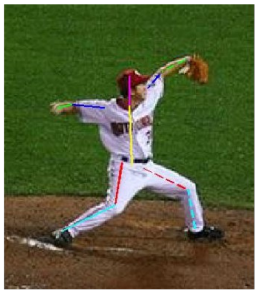
Why Human Pose Estimation?

- スポーツ分析
- ヘルスケア
- 行動分析
- 危険予知
- セキュリティ

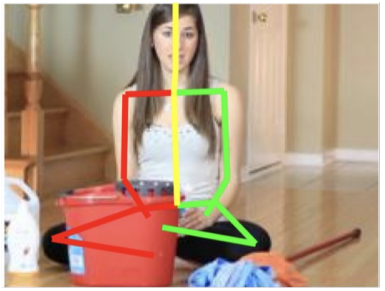
https://www.youtube.com/watch?v=hs_v3dv6OUI

Single Person vs. Multi Person

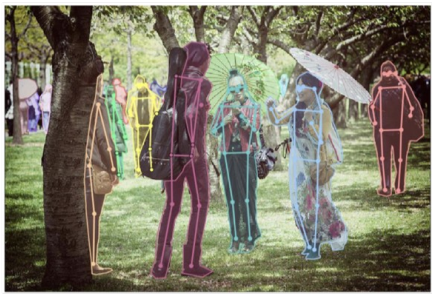
- 判定対象となる人物が一人か複数かで大きく違う
- 推定した関節を人物ごとに正しくつなげる必要がある



LSP [27]



MPII [28]



MS COCO [29]

AI Challenger [30]

Light Weight vs. Heavy Weight

応用目的によって、

・解析に時間をかけることができる

・リアルタイム性が求められる

ことがあり、それぞれ適用可能な解法が異なる

姿勢推定に限らず一般に検討しなくてはいけない重要ポイント

Approaches

先に人物を見つけてから関節を見つける

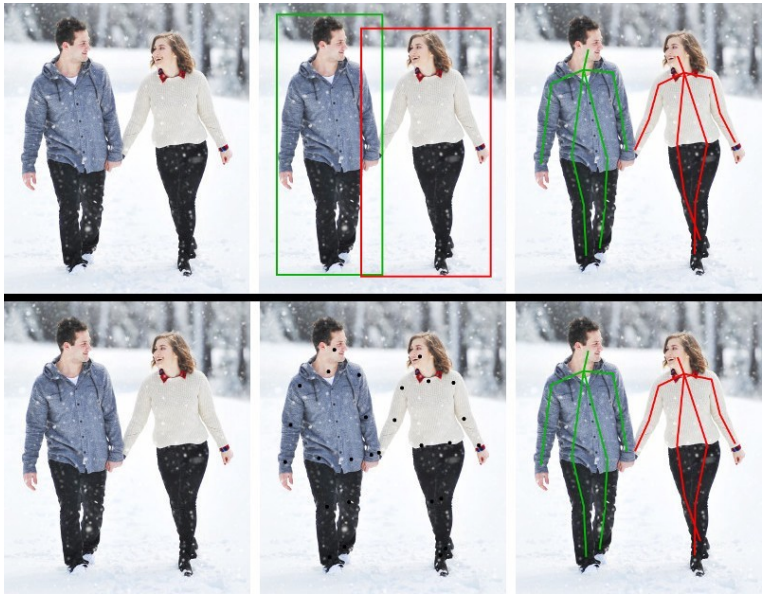
トップダウンアプローチ

先に関節を見つけてからグルーピングする

ボトムアップアプローチ

それぞれに

advantageとdisadvantageがある

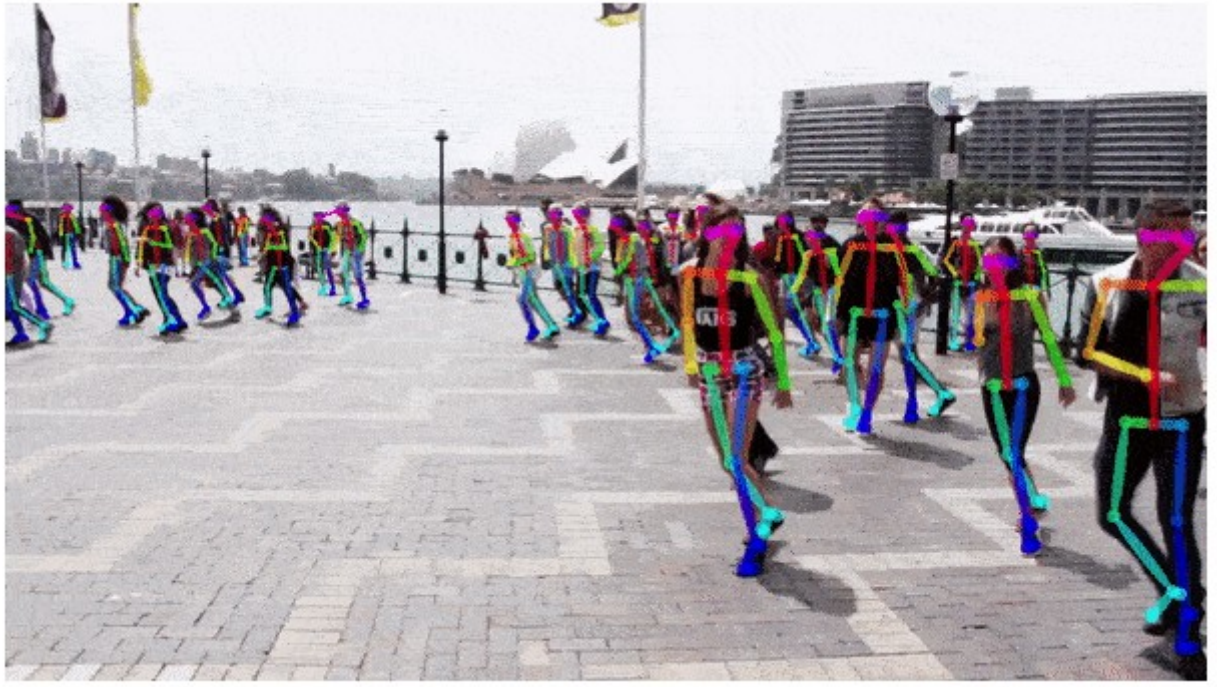


https://medium.com/beyondminds/an-overview-of-human-pose-estimation-with-deep-learning-d49eb656739b

実装例: openpose

カーネギメロン大学によるボトムアップ型の

Real Time Multi　Person Human Pose Estimation



https://github.com/CMU-Perceptual-Computing-Lab/openpose

線形回帰演習回答例

問1.

部屋数の代わりにNOx濃度を説明変数として単回帰分析を行ってください

```
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import pickle
import numpy as np



boston = load_boston()


x = boston['data']  # 物件の情報
y = boston['target']  # 家賃


x = x[:, [4]] # NOx濃度


print(x,y)
model = LinearRegression()
model.fit(x, y)


a = model.coef_
b = model.intercept_


print('a={}, b={}'.format(a, b))
```

問2.

散布図を作って回帰曲線を描画してください

```
＃問1に描画部分を付加

print('a={}, b={}'.format(a, b))

plt.scatter(x, y, c='gray', alpha=0.5) # データプロット
lreg_y = a * x + b # clf.predict(X_RM)の出力値と同じ
plt.plot(x, lreg_y, 'r') # 回帰直線プロット
plt.show()
```

問3.

すべての特徴量を説明変数として重回帰分析を行ってください

```
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import pickle
import numpy as np



boston = load_boston()


x = boston['data']  # 物件の情報
y = boston['target']  # 家賃

model = LinearRegression()
model.fit(x, y)

a = model.coef_
b = model.intercept_

print('a={}, b={}'.format(a, b))
```

問4.

自動車データセ ット http://archive.ics.uci.edu/ml/datasets/Automobile を使って、engine-sizeを用いた単回帰分析およびlength, width, heightを用いた重回帰分析を行ってください [値段に関する分析]

Attribute Information:
Attribute: Attribute Range

1. symboling: -3, -2, -1, 0, 1, 2, 3.
2. normalized-losses: continuous from 65 to 256.
3. make: alfa-romero, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedes-benz, mercury, mitsubishi, nissan, peugot, plymouth, porsche, renault, saab, subaru, toyota, volkswagen, volvo
4. fuel-type: diesel, gas.
5. aspiration: std, turbo.
6. num-of-doors: four, two.
7. body-style: hardtop, wagon, sedan, hatchback, convertible.
8. drive-wheels: 4wd, fwd, rwd.
9. engine-location: front, rear.
10. wheel-base: continuous from 86.6 120.9.
11. length: continuous from 141.1 to 208.1.
12. width: continuous from 60.3 to 72.3.
13. height: continuous from 47.8 to 59.8.
14. curb-weight: continuous from 1488 to 4066.
15. engine-type: dohc, dohcv, l, ohc, ohcf, ohcv, rotor.
16. num-of-cylinders: eight, five, four, six, three, twelve, two.
17. engine-size: continuous from 61 to 326.
18. fuel-system: 1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi.
19. bore: continuous from 2.54 to 3.94.
20. stroke: continuous from 2.07 to 4.17.
21. compression-ratio: continuous from 7 to 23.
22. horsepower: continuous from 48 to 288.
23. peak-rpm: continuous from 4150 to 6600.
24. city-mpg: continuous from 13 to 49.
25. highway-mpg: continuous from 16 to 54.
26. price: continuous from 5118 to 45400.

```
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

car = pd.read_csv('imports-85.csv', header=None)
engine_size = car[16].values.tolist()
price = car[25].values.tolist()

x = []
for v in engine_size:
    x.append([v])

y = []
for v in price:
    if v.isdecimal():
        y.append(int(v))
    else:
        y.append(0)
```

 # priceには'?'が含まれるのでクレンジングする

```
plt.scatter(x, y, c='gray', alpha=0.5)
plt.show()

model = LinearRegression()
model.fit(x, y)

a = model.coef_
b = model.intercept_

print('a={}, b={}'.format(a, b))

plt.scatter(x, y, c='gray', alpha=0.5)
lreg_y = a * x + b
plt.plot(x, lreg_y, 'r')
plt.show()
```

```
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

car = pd.read_csv('imports-85.csv', header=None)

length = car[10].values.tolist()
width = car[11].values.tolist()
height = car[12].values.tolist()
price = car[25].values.tolist()

x = []
for l, w, h in zip(length, width, height):
    x.append([l,w,h])

y = [int(p) if p != '?' else 0 for p in price ]

model = LinearRegression()
model.fit(x, y)

a = model.coef_
b = model.intercept_

print('a={}, b={}'.format(a, b))
```

 # priceには'?'が含まれるのでクレンジングする

Principal Component Analysis
PCA

主成分分析

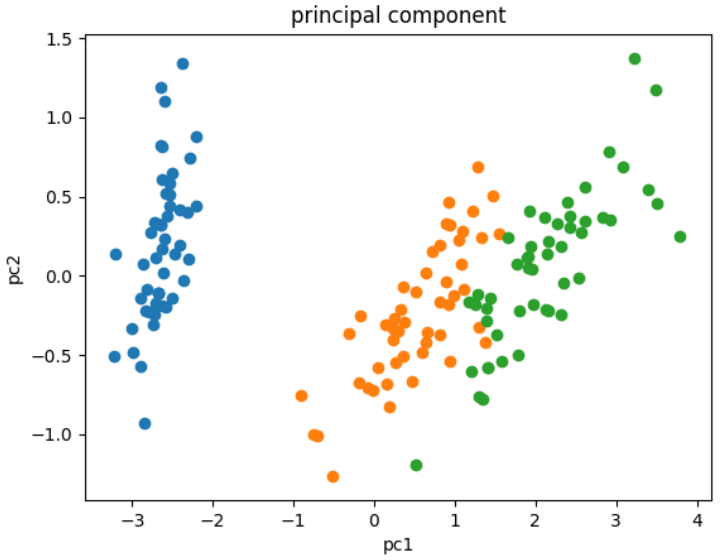
主成分分析は、相関のある多数の変数から相関のない少数で全体のばらつきを最もよく表す主成分と呼ばれる変数を合成する多変量解析の一手法。データの次元を削減するために用いられる。

https://ja.wikipedia.org/wiki/%E4%B8%BB%E6%88%90%E5%88%86%E5%88%86%E6%9E%90

代表的な教師なし学習手法

Why PCA?

データ解析ではデータ項目を横断的に解釈する必要があり、情報の損失を抑えながら、データ項目をできるだけ少ない数に置き換える主成分分析は、たいへん有効。主成分分析は多くのデータ項目を少数の項目に置き換えることで、データを解釈しやすくしてくれます。このような手続きを「次元の縮約」と呼び、具体的には、複数のデータ項目から新しい合成データ項目を作りだします。作られたデータ項目は寄与度の大きなものから第一主成分、第二主成分とよびます。



四次元のデータ項目を二次元に置き換えた例
三次元以下であれば可視化できることも大きな意味がある

Pythonによる線形回帰とPCA

iris data

```
$ python
Python 3.6.8 (default, Feb 11 2019, 22:40:57)
[GCC 7.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from sklearn.datasets import load_iris
>>> iris = load_iris()
>>> print(iris.DESCR)
.. _iris_dataset:

Iris plants dataset
--------------------


**Data Set Characteristics:**


    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica
<snip>
```

三種のアヤメについて、四つのデータ項目、sepal length (ガクの長さ), sepal width (ガクの幅), petal length (花弁の長さ), petal width (花弁の幅)を収集したデータ

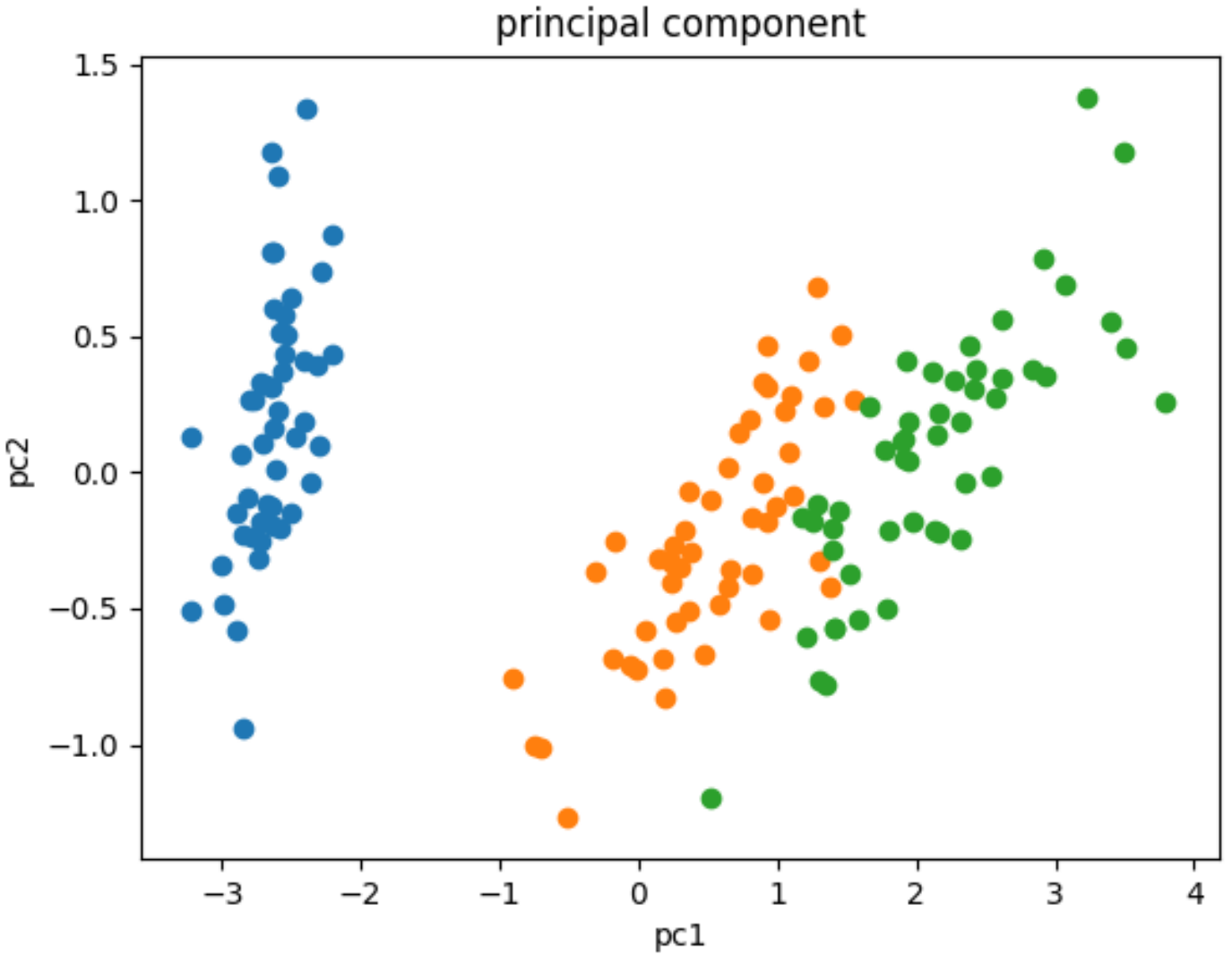
scikit-learnによるPCA

```
import numpy as np
from matplotlib import pyplot as plt
from sklearn.decomposition import PCA
from sklearn import datasets
def main():
    dataset = datasets.load_iris()
    features = dataset.data
    targets = dataset.target
    # 主成分分析する
    pca = PCA(n_components=2)
    pca.fit(features)
    # 分析結果を元にデータセットを主成分に変換する
    transformed = pca.fit_transform(features)
    # 主成分をプロットする
    for label in np.unique(targets):
        plt.scatter(transformed[targets == label, 0],
                    transformed[targets == label, 1])
    plt.title('principal component')
    plt.xlabel('pc1')
    plt.ylabel('pc2')

    print('各次元の寄与率{0}'.format(pca.explained_variance_ratio_))
    print('累積寄与率: {0}'.format(sum(pca.explained_variance_ratio_)))

    plt.show()
if __name__ == '__main__':
    main()
```

四つのデータ項目をPCAで二次元に圧縮


principal component
pc2
pc1
1.5
1.0
0.5
0.0
−0.5
−1.0
−3
−2
−1
0
1
2
3
4

演習

問1.

自動車データセットhttp://archive.ics.uci.edu/ml/datasets/Automobile を使って、engine-size, length, width, heightの四次元のデータをPCAを用いて二次元に次元圧縮してください。また結果を散布図で表示してください。

plotされた結果が何を意味するのか考えてみましょう

分類問題とその解法

Q&A