Pythonで主成分分析

専門演習(B)：Pythonでデータ分析入門

主成分分析（PCA）

- 多数の変数で説明されるデータ
  → 変数を合成しより少ない変数（＝主成分）でデータを説明
  ＝データの次元圧縮

例1）身長＋体重 → 身体の大きさ

例2）年収＋役職＋勤務先 → 社会的地位

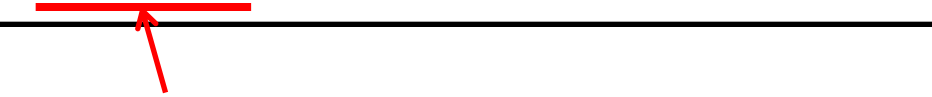
- 主成分の意味
  → どの変数がどの程度合成されているかから利用者が推定

準備

必要なライブラリのインストール

- 主成分分析に必要なライブラリ
  - numpy
    - 数値演算用のライブラリ（特に行列演算）
  - scipy
    - 科学技術計算用のライブラリ
  - scikit-learn
    - 機械学習とデータマイニング用のライブラリ
  - matplotlib
    - 主に2次元のグラフ描画用ライブラリ
- インストール手順（青字が入力）

```
$ pip install ライブラリ名 --user
```



演習用データの入力

- Excelで下のデータの数値部分を入力 → CSV形式で保存
  （Pythonのプログラムと同じ場所に保存すると良い）

名前	算数	理科	国語	英語	社会
田中	89	90	67	46	50
佐藤	57	70	80	85	90
鈴木	80	90	35	40	50
本田	40	60	50	45	55
川端	78	85	45	55	60
吉野	55	65	80	75	85
斉藤	90	85	88	92	95

CSV形式

- CSV＝Comma Separated Values
  → コンマ区切りの値
- コンマと改行で区切られたテキストデータ

対話モードで主成分分析

Pythonの起動

- 何はともあれまずはPythonの起動

```
$ python
Python 2.7.11 (default, Dec 15 2015, 19:21:39)
[GCC 4.2.1 Compatible Apple LLVM 7.0.2 (clang-700.1.81)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>>
```

Pythonの対話用プロンプト

必要なライブラリの読み込み

- numpyの読み込み（numpyだと長いから名前をnpで読み込み）

```
>>> import numpy as np
```

- 主成分分析用ライブラリ（scikit-learnの一部）を読み込み

```
>>> from sklearn.decomposition import PCA
```

データの読み込み

- numpyを利用してCSVファイルを読み込み

```
>>> data = np.loadtxt('scores.csv', delimiter=',')
```

テキストデータの読み込み　　区切り文字＝コンマ

- データラベルを別に用意
  （日本語はちょっと扱いが面倒なので今回はローマ字）

```
>>> name = ['Tanaka', 'Sato', 'Suzuki', 'Honda', 'Kawabata',
'Yoshino', 'Saito']
```

データの確認

- 全てのデータの確認

```
>>> data
```

※単に変数名を入力すると内容を表示する

- 特定の人のデータを確認

```
>>> data[name == 'Tanaka']
```

主成分分析の準備

- 主成分分析器の用意

```
>>> pca = PCA(n_components = 2, whiten = False)
```

[PCAのオプション]

- n_components
  - 主成分を幾つ求めるか（個数：上の例では2）
  - 'mle' を指定すると最尤推定により個数を自動的に求める
  - 0～1の間の実数を指定すると累積寄与率がその値になるまで主成分を求める
- whiten
  - 白色化を行うかどうか（True|False）

主成分分析の実行

- 主成分分析の実行

```
>>> pca.fit(data)
```

- 主成分の数の確認

```
>>> pca.n_components_
>>> 2
```

主成分分析の確認

- 寄与率の確認

```
>>> pca.explained_variance_ratio_
>>> array([ 0.6688013 ,  0.28791087])
```

- 因子負荷量の確認

```
>>> pca.components_
>>> array([[-0.04318455,-0.11661043, 0.55136578, 0.60073709, 0.56537406],
           [-0.84543226,-0.51948621,-0.08791982,-0.08720053, 0.00667425]])
```

寄与率

- 寄与率
  - 端的に言えば各主成分の重要性を表す
  - 各主成分によって説明できるデータの割合を表す
    → 全ての主成分の寄与率を足し合わせると1.0になる
- 累積寄与率
  - 主成分の寄与率を足し合わせたもの
  - 選択した複数の主成分によって説明できるデータの割合を表す

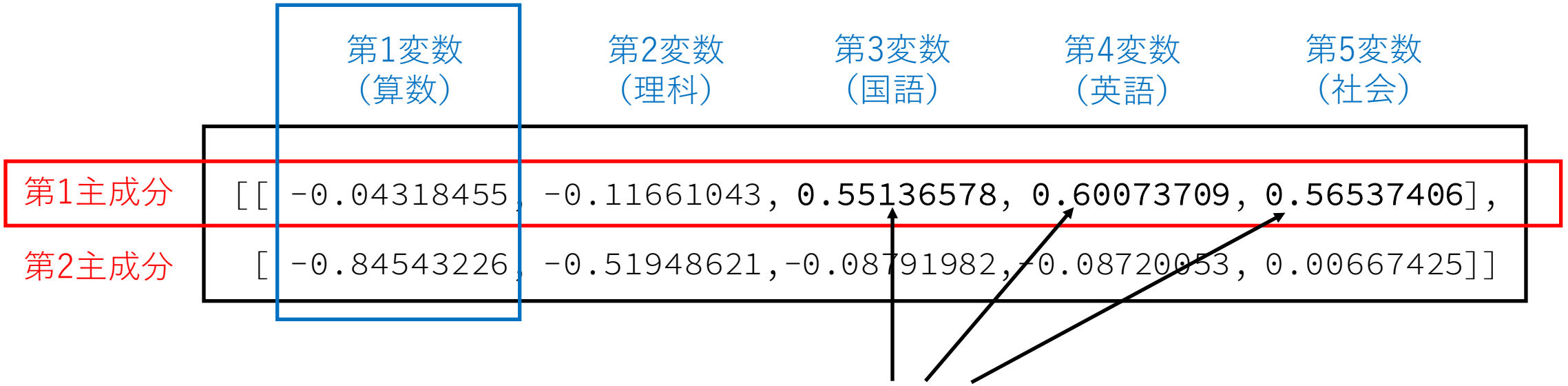
```
>>> np.cumsum(pca.explained_variance_ratio_)
>>> array([ 0.6688013 ,  0.95671218])
```

2つの主成分で95.67%のデータが説明できる

因子負荷量

- 各変数の各主成分への影響力 → 各主成分の意味の推定

［実行例の解説］



国語，英語，社会の負荷量が高い → 第1主成分は文系科目に関係あるかも

データの次元圧縮

- データを主成分空間に写像＝次元圧縮

```
>>> x = pca.transform(data)
```

- 次元圧縮後のデータを確認

```
>>> x
>>> array([[-21.21097689, -21.47715546],
       [ 35.71460142,  11.68959258],
       [-42.0704435 , -10.53162768],
       [-22.74370588,  37.14882026],
       [-21.22256751,  -8.3637958 ],
       [ 27.54978153,  16.81652223],
       [ 43.98331082, -25.28235614]])
```

データをプロットしてみる

データプロット用ライブラリの読み込み

- matplotlibの読み込み

```
>>> from matplotlib import pyplot
```

- 対話モード

```
>>> pyplot.ion()
```

- プロット面のクリア

```
>>> pyplot.clf()
```

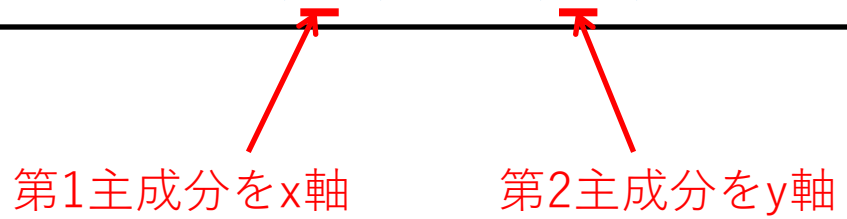
次元圧縮後のデータをプロットする

- 色の設定

```
>>> colors = [pyplot.cm.hsv(0.1 * i,1) for i in range(len(name))]
```

- データのプロット

```
>>> for i in range(len(name)):
...     pyplot.scatter(x[i,0], x[i,1], c=colors[i], label=name[i])
```



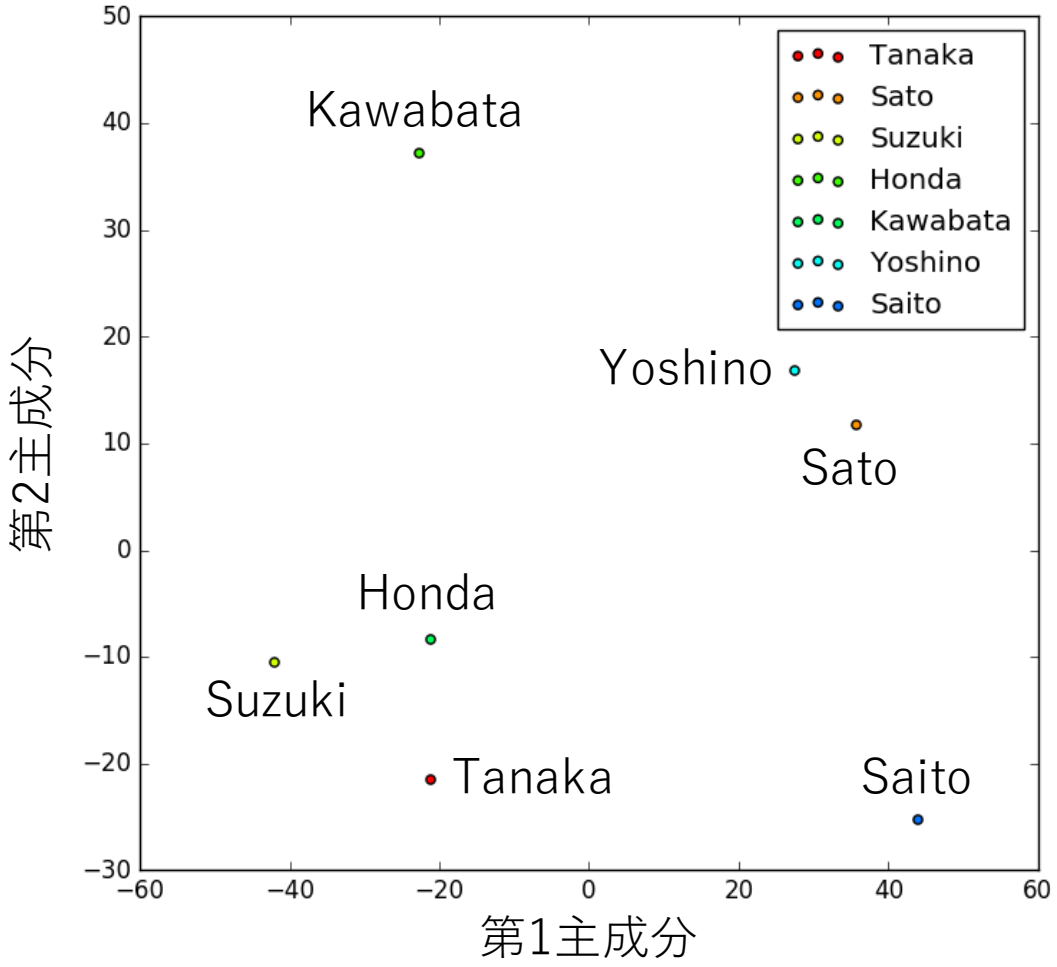
- 凡例の表示

```
>>> pyplot.legend()
```

プロット例

- 因子負荷量からの意味づけ
  - 第1主成分＝文系科目
  - 第2主成分＝理系科目

- 理系が強い → 斉藤
- 文系が強い → 川端
- どちらもそこそこ → 佐藤，吉野
- どちらもいまいち → その他



※データラベルは後で追記しています。