

# 機械学習と主成分分析

## Ver. 1.4

2018-05-17

森下功啓

# 改訂履歴

- 2017-03-12 ver.1.1 リリース（公開開始）
- 2017-07-19 ver.1.2
  - 実用上、相関係数行列を基にした主成分分析が主に使われるため、それに合わせて資料を修正
  - 文面の表現もちょこちょこ変更している
  - 機械学習の部分はかなり書き換えた
- 2017-08-07 ver.1.3 コラムで、次元数が4と誤っていたのを3に修正
- 2018-05-17 ver.1.4 主成分得点を求める式の例の誤りを修正

# 機械学習と主成分分析の概要

# 機械学習とは

機械学習とは、データのパターンを学習することで識別や回帰による予測をソフトウェアで実現する手法である。

例えば、人の話し声を識別して文字に変換したり、株価の予想に使われている。冷房を最適化することで消費電力を4割削減した実証例もある。

機械学習では人間には認知できない程多くの変数からパターンを学習できる。

学習のアルゴリズムは、ニューラルネットワークやサポートベクターマシンやランダムフォレストなど、多数存在する。

# 特徴ベクトル

識別や回帰のために観測値を順番に並べたベクトルを特徴ベクトルという。

例えば、体重 $x_1$ ・身長 $x_2$ ・体脂肪率 $x_3$ ・平均的な髪の毛の長さ $x_4$ ・毛髪面積比（ハゲ面積/頭皮面積） $x_5$ ・髭の平均直径 $x_6$ を用いて男女を区別する場合を考えよう。 $x$ を並べたものが特徴ベクトル $\mathbf{X}$ である。個々の要素のことは特徴量と呼ぶ。

$$\mathbf{X} = (x_1, x_2, x_3, x_4, x_5, x_6)$$

観測データ1つ1つはレコードとも呼ぶ

レコード番号	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
1	69.4	152.3	16.6	16.7	0.452	0.019
2	55.9	150.7	14.0	22.3	0.468	0.021
3	65.2	162.3	15.1	20.1	0.669	0.022
4	63.7	164.0	18.8	22.8	0.581	0.023

# 教師あり学習

教師あり学習とは、特徴ベクトルと正解の関係を学習する手法である。

ここで、正解が付与されたデータを教師データと呼び、そのうち学習に用いられるデータを学習データと呼ぶ。

レコード番号	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$	$v$
1	69.4	152.3	16.6	16.7	0.452	0.019	man
2	55.9	150.7	14.0	22.3	0.468	0.021	man
3	65.2	162.3	15.1	20.1	0.669	0.022	man
4	63.7	164.0	18.8	22.8	0.581	0.023	woman

正解

教師データの例

# 教師ラベルの扱い

正解は、識別が目的の場合は文字列、回帰が目的の場合は数値となる。文字列の場合、特に**正解ラベル**又は**教師ラベル**という。

ただし、教師ラベルであっても学習時には数値に変換されて扱われている。どのように変換されるかはプログラム次第である。

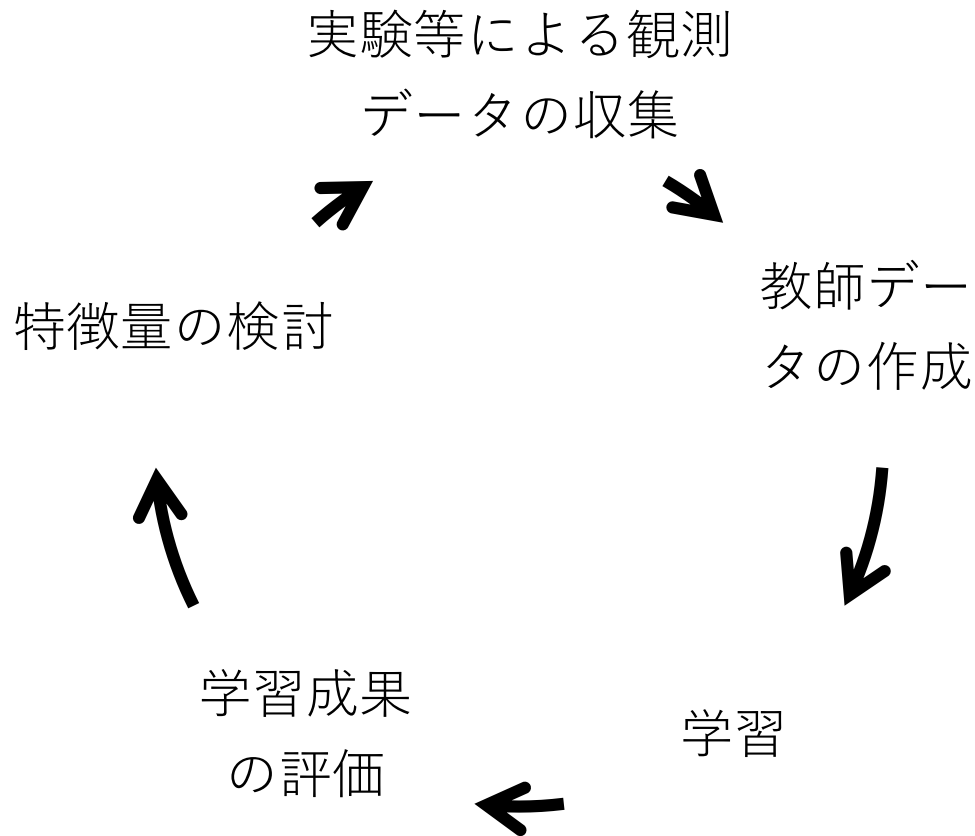
また、ソフトウェアによっては教師ラベルに数値しか受け付けないものもあり、その場合は以下のように教師ラベルを整数に変換しておく。

man	→	0
woman	→	1

ラベルの数値への変換例

# 教師あり学習の作業の流れ

教師あり学習における作業の流れを以下に示す。主成分分析は「教師データの作成」で利用されることがある。





# 次元数削減の要請

機械学習では、特徴ベクトルの次元数が増えるほど、偶然生じたパターンを学習しやすくなる。これを次元の呪いという。

また例えば、1つの特徴量につき3個のパターンがあるものとする。それぞれのパターンが独立であれば、2次元の特徴ベクトルで生じる合計のパターン数は $3^2$ で9パターンである。 $n$ 次元あれば $3^n$ パターンとなる。初等統計学で学ぶように、変数が正規分布と仮定すると1パターン当たりの最低限必要なデータ数は30個程である。すなわち、学習データ数は $30 \times 3^n$ 個必要であり、次元数が増えると指数関数的に学習に要するデータ数は増加する。

必要なデータ数を確保するために実験を指数関数的に行うとか、どMの所業である。実験数が少ないにこしたことはない。

したがって、特徴ベクトルの次元数は少ない方が良い。

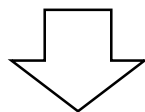
# 主成分分析 (PCA)

主成分分析 (PCA) は、次元の数を削減する「次元圧縮」や、重回帰分析の前処理として特徴量（多変量解析の世界では説明変数という）同士を無相関にする目的で利用される手法である。

PCAは、分析対象の特徴量から新たな指標を合成する。元データが $n$ 次元なら、互いに直行した $n$ 個の指標が合成される。ここで、合成された指標のことを「主成分」と呼ぶ。

体重 $x_1$ ・身長 $x_2$ ・体脂肪率 $x_3$ ・平均的な髪の毛の長さ $x_4$ ・毛髪面積比 $x_5$ ・髭の平均直径 $x_6$

元データ



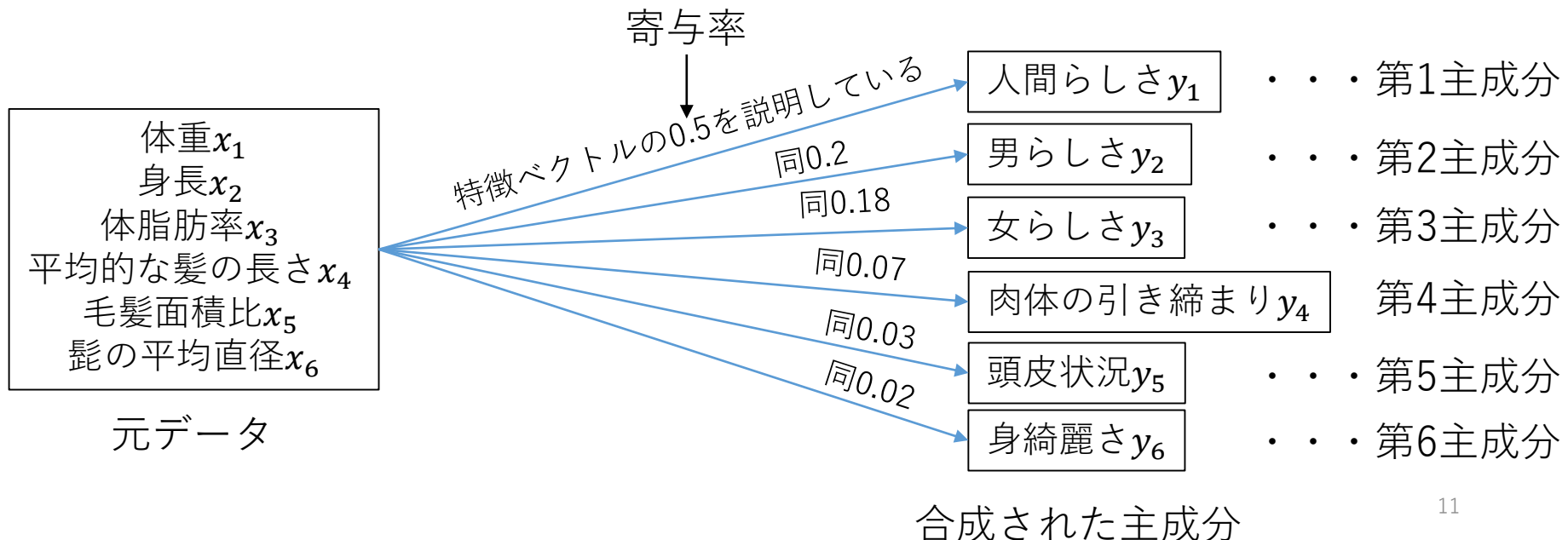
主成分分析

人間らしさ $y_1$ ・男らしさ $y_2$ ・女らしさ $y_3$ ・肉体の引き締まり $y_4$ ・頭皮状況 $y_5$ ・身綺麗さ $y_6$

合成された主成分

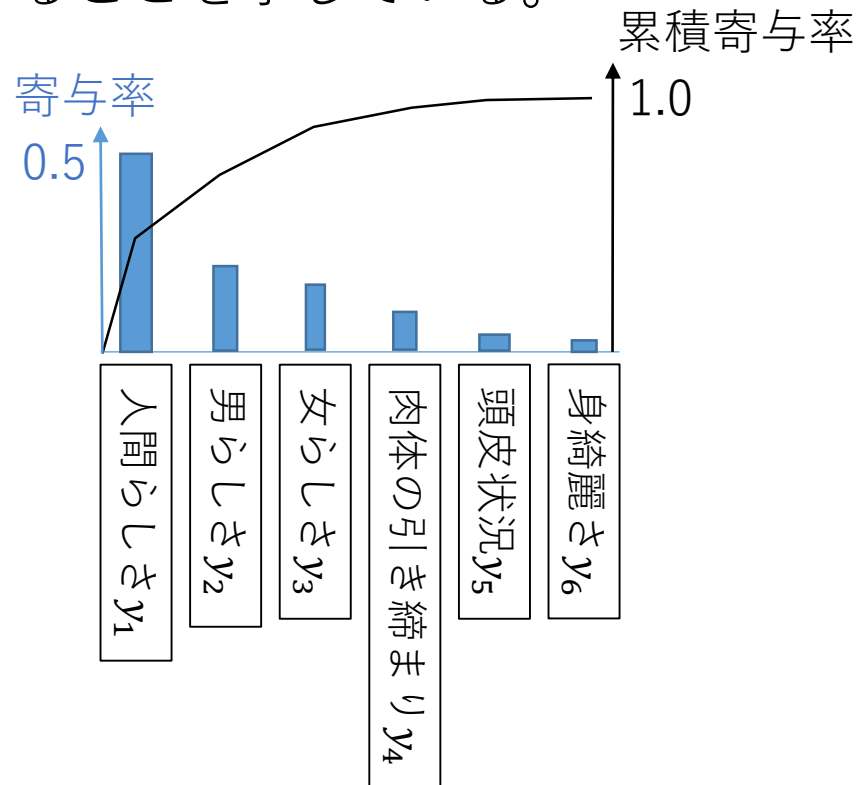
# 寄与率

主成分は、それぞれ元のデータの情報量を分担している。情報量は、支配・影響の大きさと言い直しても良い。元のデータに対する情報量の大きさを**寄与率**という。また、主成分のことを寄与率の大きい順に「第1主成分」「第2主成分」・・・と呼ぶ。



# 累積寄与率

以下に、寄与率を棒グラフで表し、その積算値として累積寄与率を折れ線グラフで表した。ここで、4つ目の主成分までの寄与率の合計は0.95となっており、元データの情報の95%を第4主成分までで表現できることを示している。



繰り返しになるが、寄与率のスライドで説明したように第1主成分が最も情報量が大きい（＝データの説明力が大きい）。その次が第2主成分だ。

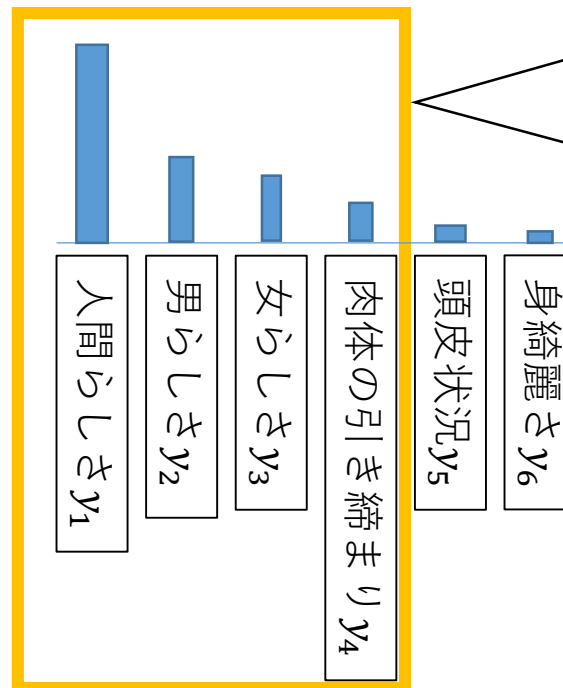
ここで、情報量の小さい主成分を捨てれば、回帰に対する計測誤差の影響を小さくできるかも知れない。

また、情報量の大きな主成分のみを学習に使えば、学習データから必要な情報が多少減ったとしても、次元数が減ったことで機械学習のトータルの性能は向上できる可能性がある。

# 何個の主成分を使うか？

機械学習においてPCAは、次元削減のために利用される。故に機械学習では元データの次元数 $n$ よりも少ない主成分を用いる。

データと分析の目的に依存するが、元データの90%を説明できれば、大抵の場合は上々である。



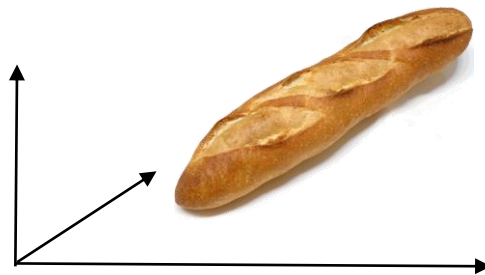
累積寄与率が90-95%もあれば普通は十分であろうから、左の例では枠内の主成分を採用する。

第4主成分まで利用することにすれば、2次元減らすことに成功している。

# 主成分分析の幾何学的な解釈 1/3

正規分布に従う $n$ 次元の観測データをプロットすると、球のように分布する。 $n$ が2なら円、 $n$ が3なら球、 $n$ が4以上なら超球である。もし、特徴量間に相関があれば、分布は楕円となる。

頭のなかに楕円球を思い浮かべるのは大変だろうから、フランスパンのように分布するデータを思い浮かべよう。 $n$ が4以上なら超フランスパンである。ここで、このフランスパンはナンのように潰れていても構わない。

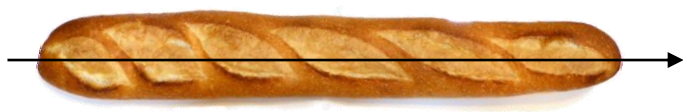


3次元空間に浮く3次元フランスパン

## 主成分分析の幾何学的な解釈 2/3

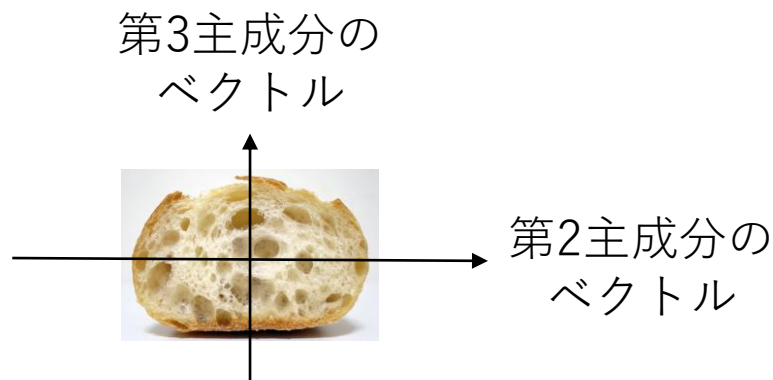
このフランスパンの長軸方向（分散が最大になる方向）が第1主成分（のベクトル）である。

第2主成分は、第1主成分のベクトルに直交し、且つ分散が最大になる方向に取られる。もし $n=3$ であれば、第3主成分は第1主成分と第2主成分に直交するという条件で自動的に決定される。



第1主成分の  
ベクトル

上から見たフランスパン



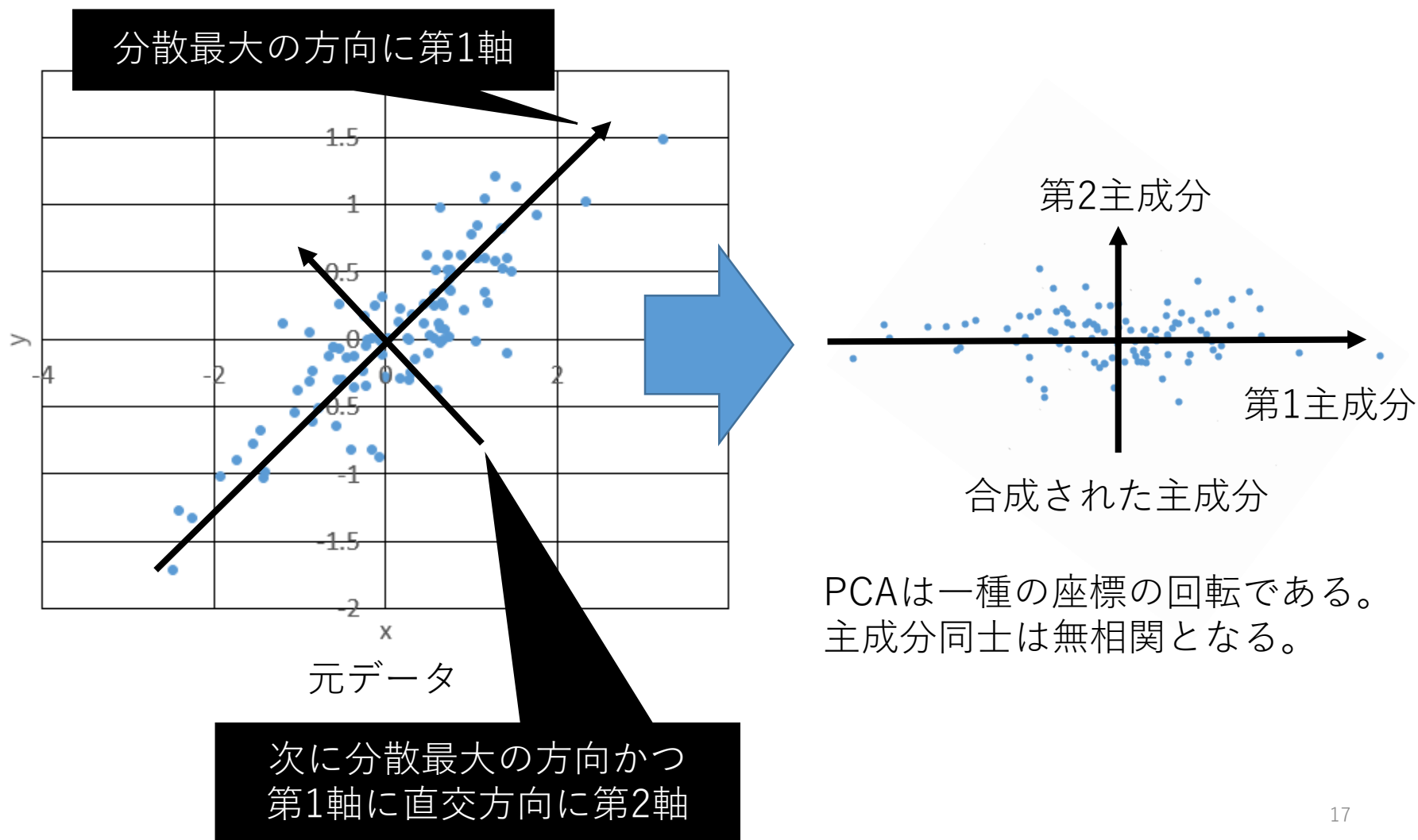
第3主成分の  
ベクトル

第2主成分の  
ベクトル

フランスパンの断面



# 主成分分析の幾何学的な解釈 3/3



# 主成分のベクトル==固有ベクトル

主成分のベクトルは、数学的には相関行列から固有ベクトルと固有値を求める事によって得られる。詳細は割愛するが、固有値は分散の大きさに比例する。また、固有ベクトルは主成分の向きである。そして最大の固有値を持つ固有ベクトルが第1主成分である。

## 第1主成分の固有ベクトル

```
Component loadings:
      Comp.1      Comp.2      Comp.3      Comp.4
Petal.Length 0.5804131 0.02449161 0.1421264 0.8014492
Petal.Width  0.5648565 0.06694199 0.6342727 -0.5235971
Sepal.Length 0.5210659 0.37741762 -0.7195664 -0.2612863
Sepal.Width -0.2693474 0.92329566 0.2443818 0.1235096

Component variances:
      Comp.1      Comp.2      Comp.3      Comp.4
2.91849782 0.91403047 0.14675688 0.02071484
```

主成分毎に並んだ固有ベクトル

主成分毎に並んだ固有値

RコマンダーでIrisデータを主成分分析した例

# 主成分得点（主成分スコア）

元データを次元ごとに正規化 $N(0,1)$ したデータと固有ベクトルの内積を取った値を主成分得点という。主成分得点 $y_{ij}$ は以下の式で求める。ここで、 $j$ は第 $j$ 主成分を表し、 $i$ はレコードの番号を表している。また、 $\mathbf{X}_i$ は元データを次元ごとに正規化したデータのベクトルを表し、 $\mathbf{C}_j$ は第 $j$ 主成分の固有ベクトルである。

$$y_{ij} = \mathbf{X}_i \cdot \mathbf{C}_j \quad \leftarrow \text{計算自体はベクトルの内積}$$

レコード番号	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
1	0.93	-0.78	0.38	-1.09	-1.50	-0.24
2	-1.72	-0.95	-0.95	0.77	-1.34	0.33
3	0.12	0.22	-0.41	0.06	0.67	0.60
4	-0.18	0.39	1.48	0.94	-0.20	0.77



レコード番号	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$y_6$
1	$y_{11}$	$y_{12}$	$y_{13}$	$y_{14}$	$y_{15}$	$y_{16}$
2	$y_{21}$	$y_{22}$	$y_{23}$	$y_{24}$	$y_{25}$	$y_{26}$
3	$y_{31}$	$y_{32}$	$y_{33}$	$y_{34}$	$y_{35}$	$y_{36}$
4	$y_{41}$	$y_{42}$	$y_{43}$	$y_{44}$	$y_{45}$	$y_{46}$

元データを次元ごとに正規化したデータ

主成分得点

# 主成分得点の具体的計算例

例えば、第1主成分の固有ベクトル  $\mathbf{c}_1$  を式(1)のように定義したとき、 $y_{11}$  は式(2)で求めることができる。

$$\mathbf{c}_1 = (0.8, 0.7, 0.2, -0.9, 0.5, -0.03)^T \quad (1)$$

$$y_{11} = 0.93 \times 0.8 + (-0.78) \times 0.7 + 0.38 \times 0.2 + (-1.09) \times (-0.9) + (-1.50) \times 0.5 + (-0.24) \times (-0.03) \quad (2)$$

レコード番号	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
1	0.93	-0.78	0.38	-1.09	-1.50	-0.24
2	-1.72	-0.95	-0.95	0.77	-1.34	0.33
3	0.12	0.22	-0.41	0.06	0.67	0.60
4	-0.18	0.39	1.48	0.94	-0.20	0.77



レコード番号	$y_1$	$y_2$	$y_3$	$y_4$	$y_5$	$y_6$
1	$y_{11}$	$y_{12}$	$y_{13}$	$y_{14}$	$y_{15}$	$y_{16}$
2	$y_{21}$	$y_{22}$	$y_{23}$	$y_{24}$	$y_{25}$	$y_{26}$
3	$y_{31}$	$y_{32}$	$y_{33}$	$y_{34}$	$y_{35}$	$y_{36}$
4	$y_{41}$	$y_{42}$	$y_{43}$	$y_{44}$	$y_{45}$	$y_{46}$

元データを次元ごとに正規化したデータ

主成分得点

# 主成分分析を伴う機械学習の作業手順例

ここまでの説明で機械学習と主成分分析の概要が分かったこと  
と思う。そこで、主成分分析を伴う機械学習作業手順を以下に  
整理しておく。

1. 外れ値の除去（グラフを見たり、検定により削除）
  1. グラフには、ヒストグラムや箱ひげ図や散布図行列を使う
2. 元データに教師ラベルが付いていれば取り除く
3. 主成分分析により、主成分得点を求める
4. 累積寄与率を吟味し、第何主成分まで利用するか決める
5. 主成分得点に教師ラベルを付与し、教師データを作成する
6. 学習する
7. 学習に用いなかった教師データを用いて学習成果を評価する
8. 新たに得られた未知データを識別・予測する
  1. ここで、未知データは固有ベクトルを用いて主成分得点に変換されている必要がある。

# サンプルデータとプログラム

# サンプルデータのダウンロード

以降のスライドで利用するデータを下部のリンクよりダウンロードして欲しい。含まれるデータは以下の通り。

- iris.xlsx : irisの計測データ
- iris\_pca\_result.txt : PCAの結果
- iris\_主成分得点\_mmult版.xlsx : 主成分得点を計算した例
- iris\_主成分得点\_愚直に掛けた版.xlsx : 主成分得点を計算した例
- machine\_learning : 機械学習用のプログラム

Download:

[https://www.dropbox.com/s/n6c067yapcwvgi1/pca\\_data.zip?dl=0](https://www.dropbox.com/s/n6c067yapcwvgi1/pca_data.zip?dl=0)



# 主成分分析の例

GUIツールとして、EZRを使っています

EZR download: <http://www.jichi.ac.jp/saitama-sct/SaitamaHP.files/statmed.html>



例題：Irisのデータセットを使って花の種類を予想する。その際、主成分分析を用いてより少ない特徴量で予想を実施せよ。

データには、がくの長さ[cm]，がくの幅[cm]，花弁の長さ[cm]，花弁の幅[cm]の4成分と、setosa, versicolor, virginicaという3種類の花の種類がラベルとして記載されている。

	A	B	C	D	E	F	G	H
1	No.	Sepal.Len	Sepal.Wid	Petal.Len	Petal.Wid	Species.la	Species.code	
2	1	5.1	3.5	1.4	0.2	setosa	1	
3	2	4.9	3	1.4	0.2	setosa	1	
4	3	4.7	3.2	1.3	0.2	setosa	1	
5	4	4.6	3.1	1.5	0.2	setosa	1	
6	5	5	3.6	1.4	0.2	setosa	1	
7	6	5.4	3.9	1.7	0.4	setosa	1	
8	7	4.6	3.4	1.4	0.3	setosa	1	
9	8	5	3.4	1.5	0.2	setosa	1	

目的変数であるラベルを  
数値に変換したコード

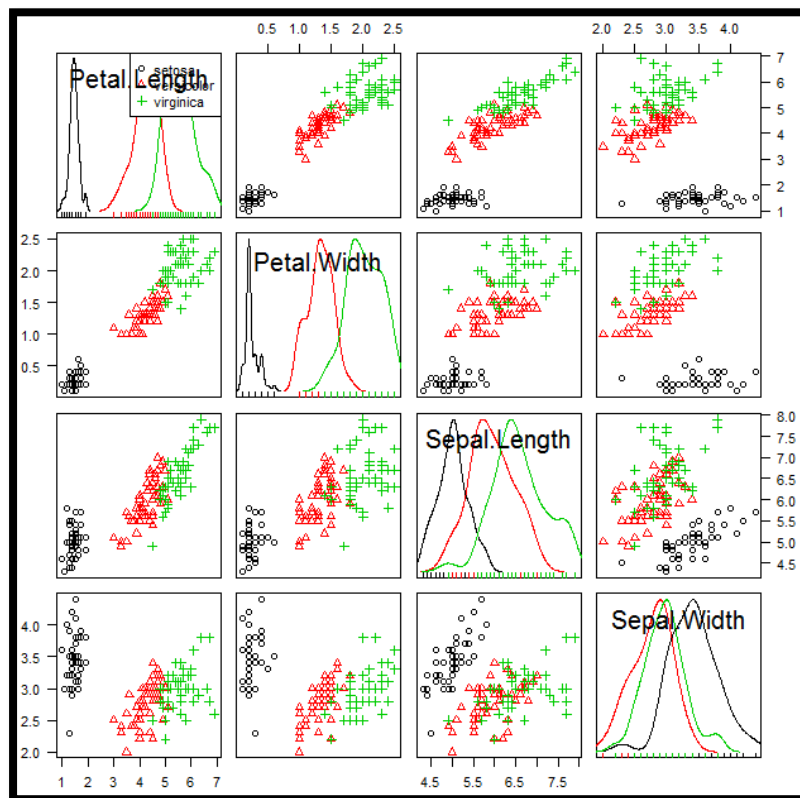
1: setosa

2: versicolor

3: virginica

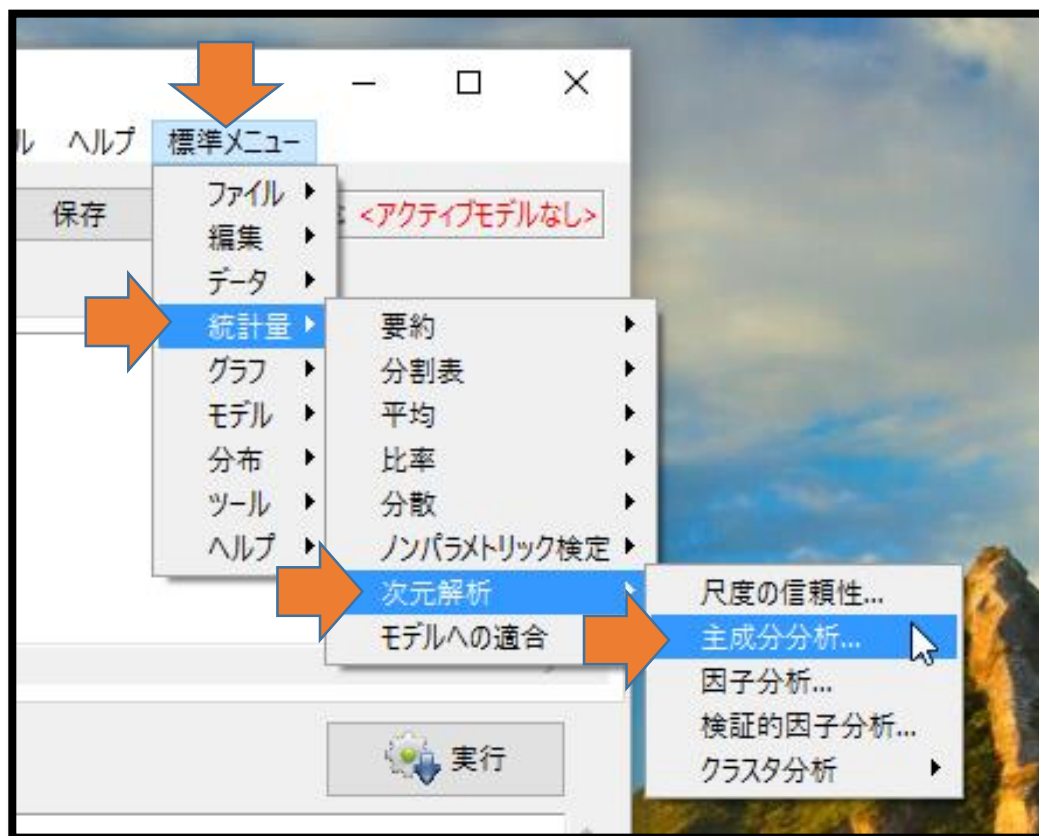
# まずはデータの分布をチェック

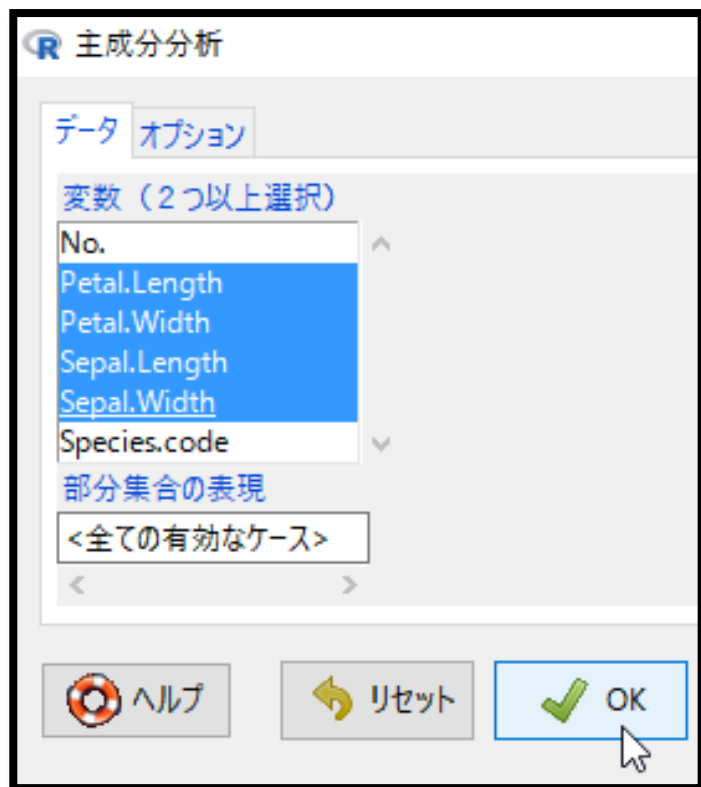
R用いて、花の種類ごとに層別で色分けした散布図行列を描いた。  
これを見ると、分布がなんとなく分かれているので、花の種類を分離できそうな気がする。



# EZRでの主成分分析

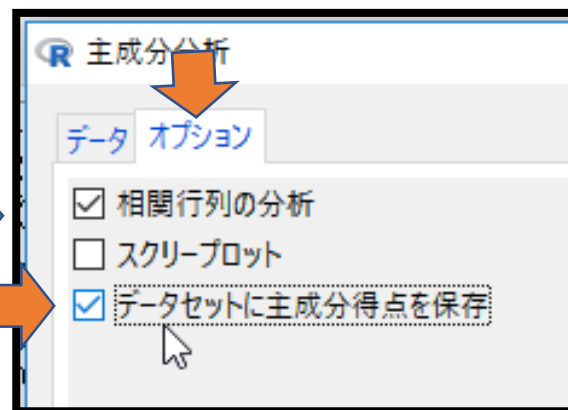
1. データ読み込み
2. 主成分分析



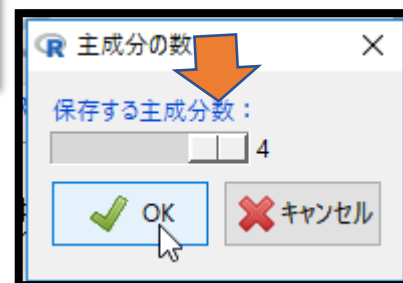


解析対象の変数を選択

目的変数を解析対象にしないこと！！



主成分得点の保存を選択



主成分数を選択

Component loadings:

	Comp.1	Comp.2	Comp.3	Comp.4
Petal.Length	0.5804131	0.02449161	0.1421264	0.8014492
Petal.Width	0.5648565	0.06694199	0.6342727	-0.5235971
Sepal.Length	0.5210659	0.37741762	-0.7195664	-0.2612863
Sepal.Width	-0.2693474	0.92329566	0.2443818	0.1235096

Component variances:

Comp.1	Comp.2	Comp.3	Comp.4
2.91849782	0.91403047	0.14675688	0.02071484

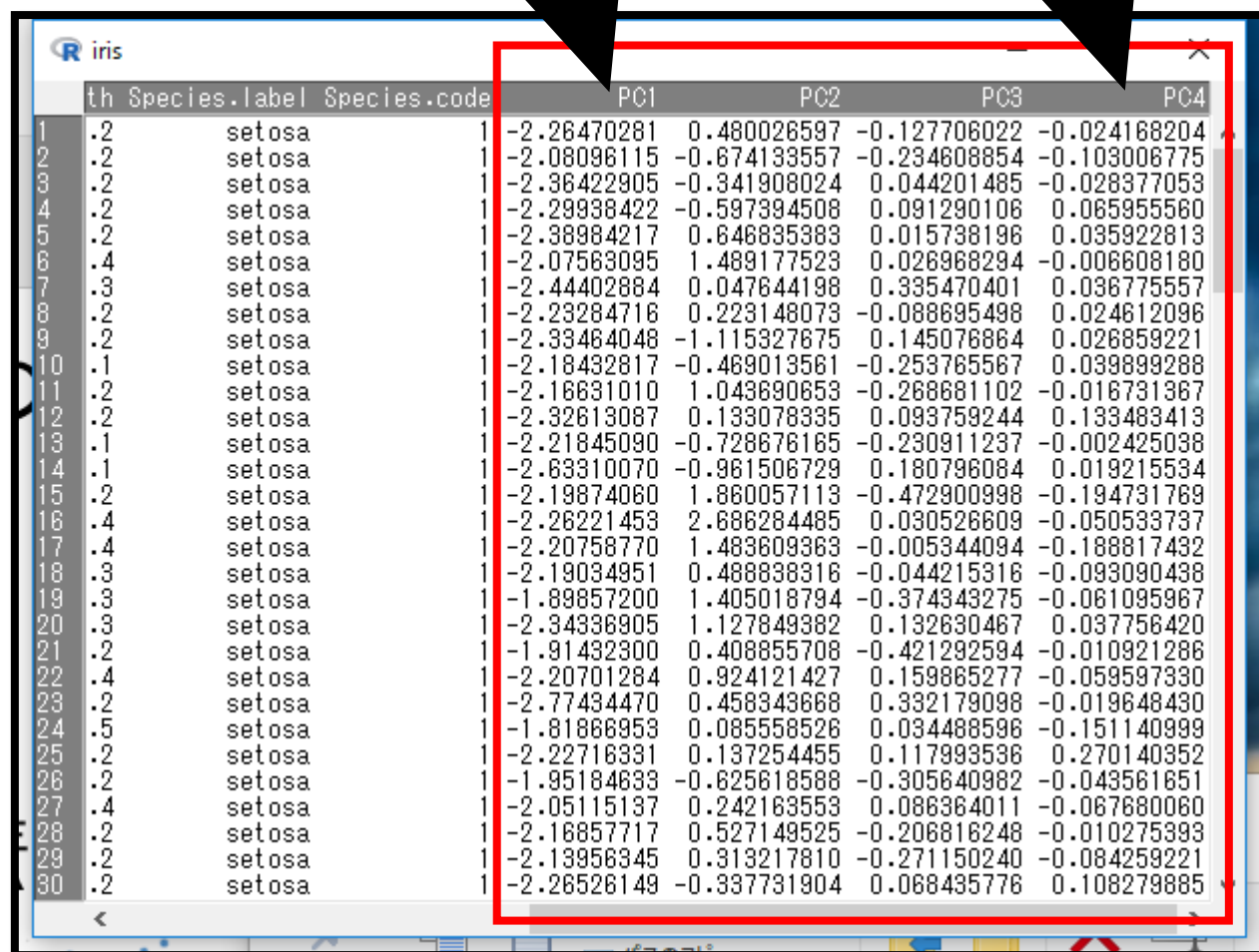
Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4
Standard deviation	1.7083611	0.9560494	0.38308860	0.143926497
Proportion of Variance	0.7296245	0.2285076	0.03668922	0.005178709
Cumulative Proportion	0.7296245	0.9581321	0.99482129	1.000000000

解析結果

第1主成分の得点

第4主成分の得点



	th	Species.label	Species.code	PC1	PC2	PC3	PC4
1	.2	setosa	1	-2.26470281	0.480026597	-0.127706022	-0.024168204
2	.2	setosa	1	-2.08096115	-0.674133557	-0.234608854	-0.103006775
3	.2	setosa	1	-2.36422905	-0.341908024	0.044201485	-0.028377053
4	.2	setosa	1	-2.29938422	-0.597394508	0.091290106	0.065955560
5	.2	setosa	1	-2.38984217	0.646835383	0.015738196	0.035922813
6	.4	setosa	1	-2.07563095	1.489177523	0.026968294	-0.006608180
7	.3	setosa	1	-2.44402884	0.047644198	0.335470401	0.036775557
8	.2	setosa	1	-2.23284716	0.223148073	-0.088695498	0.024612096
9	.2	setosa	1	-2.33464048	-1.115327675	0.145076864	0.026859221
10	.1	setosa	1	-2.18432817	-0.469013561	-0.253765567	0.039899288
11	.2	setosa	1	-2.16631010	1.043690653	-0.268681102	-0.016731367
12	.2	setosa	1	-2.32613087	0.133078335	0.093759244	0.133483413
13	.1	setosa	1	-2.21845090	-0.728676165	-0.230911237	-0.002425038
14	.1	setosa	1	-2.63310070	-0.961506729	0.180796084	0.019215534
15	.2	setosa	1	-2.19874060	1.860057113	-0.472900998	-0.194731769
16	.4	setosa	1	-2.26221453	2.686284485	0.030526609	-0.050533737
17	.4	setosa	1	-2.20758770	1.483609363	-0.005344094	-0.188817432
18	.3	setosa	1	-2.19034951	0.488838316	-0.044215316	-0.093090438
19	.3	setosa	1	-1.89857200	1.405018794	-0.374343275	-0.061095967
20	.3	setosa	1	-2.34336905	1.127849382	0.132630467	0.037756420
21	.2	setosa	1	-1.91432300	0.408855708	-0.421292594	-0.010921286
22	.4	setosa	1	-2.20701284	0.924121427	0.159865277	-0.059597330
23	.2	setosa	1	-2.77434470	0.458343668	0.332179098	-0.019648430
24	.5	setosa	1	-1.81866953	0.085558526	0.034488596	-0.151140999
25	.2	setosa	1	-2.22716331	0.137254455	0.117993536	0.270140352
26	.2	setosa	1	-1.95184633	-0.625618588	-0.305640982	-0.043561651
27	.4	setosa	1	-2.05115137	0.242163553	0.086364011	-0.067680060
28	.2	setosa	1	-2.16857717	0.527149525	-0.206816248	-0.010275393
29	.2	setosa	1	-2.13956345	0.313217810	-0.271150240	-0.084259221
30	.2	setosa	1	-2.26526149	-0.337731904	0.068435776	0.108279885

主成分得点がアクティブデータセット内に保存されている

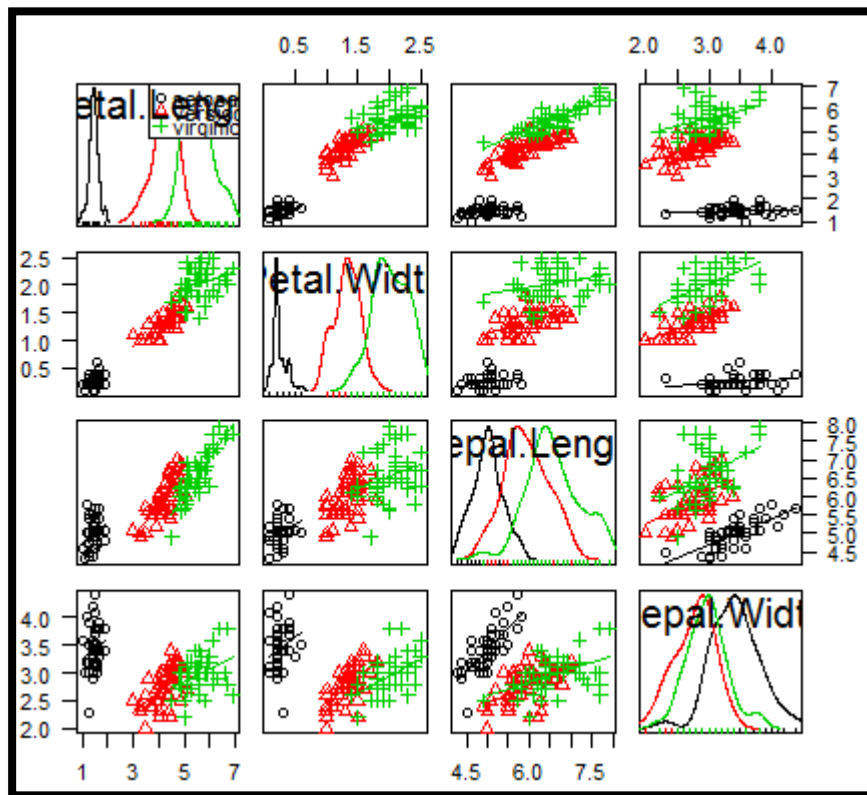
## 使う主成分数を決める

累積寄与率に赤色の下線を付けた。これによると、第3主成分までで寄与率が0.99を超えており、十分に次元数を減らせるようだ。

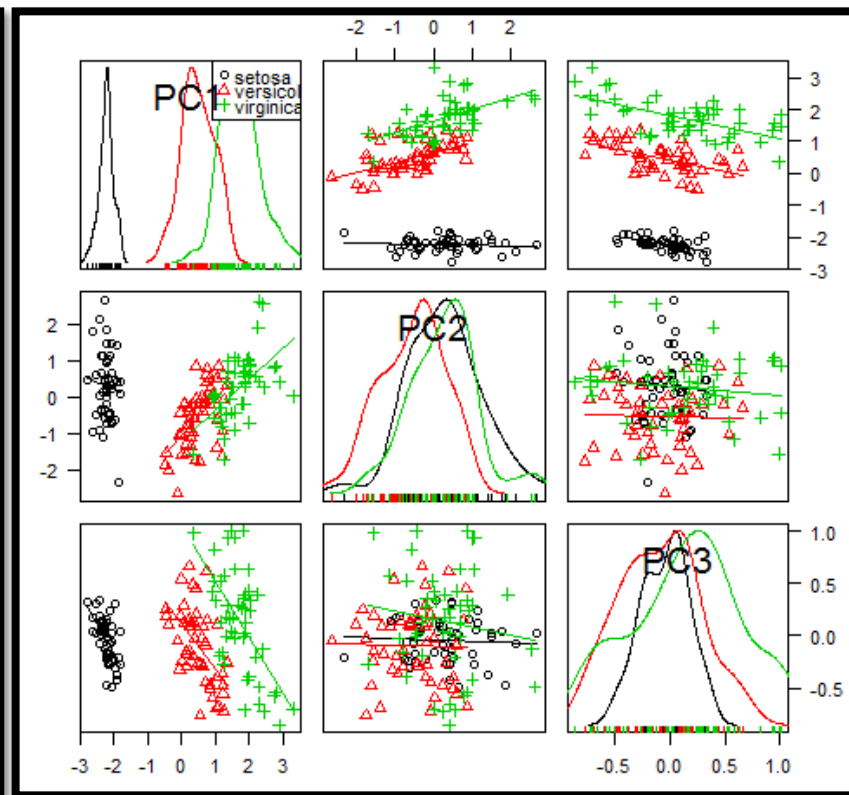
Importance of components:

	Comp.1	Comp.2	Comp.3	Comp.4
Standard deviation	1.7083611	0.9560494	0.38308860	0.143926497
Proportion of Variance	0.7296245	0.2285076	0.03688922	0.005178709
Cumulative Proportion	0.7296245	0.9581321	0.99482129	1.000000000

ひとまず、散布図を描いてみる



元データの散布図行列



主成分得点の散布図行列

全体として、各軸間で無相関であることが分かる。

# 一応、Excelで主成分得点を計算する

計算の手順を確認するために、Excelを使って主成分得点を計算した。

SUM		✕ ✓ <i>f<sub>x</sub></i>		=I10*B\$2+\$J10*B\$3+\$K10*B\$4+\$L10*B\$5													
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		Comp.1	Comp.2	Comp.3	Comp.4												
2	Sepal.Length	0.5211	0.3774	-0.7196	-0.2613												
3	Sepal.Width	-0.2693	0.9233	0.2444	0.1235	(計算しやすいように固有ベクトルの行を少し入れ替えている)											
4	Petal.Length	0.5804	0.0245	0.1421	0.8014												
5	Petal.Width	0.5649	0.0669	0.6343	-0.5236												
6		↑ Rで求めた固有ベクトル															
7																	
8	元データ							正規化 (次元毎に平均0, 標準偏差1.0に変換)				主成分得点					
9	No.	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species .label	Species .code	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	PC1	PC2	PC3	PC4		
10	1	5.1	3.5	1.4	0.2	setosa	1	-0.89767	1.0156	-1.3358	-1.31	.10*B\$3	0.478	-0.13	-0.02		
11	2	4.9	3	1.4	0.2	setosa	1	-1.1392	-0.132	-1.3358	-1.31	-2.07	-0.67	-0.23	-0.1		
12	3	4.7	3.2	1.3	0.2	setosa	1	-1.38073	0.3273	-1.3924	-1.31	-2.36	-0.34	0.044	-0.03		
13	4	4.6	3.1	1.5	0.2	setosa	1	-1.50149	0.0979	-1.2791	-1.31	-2.29	-0.6	0.091	0.066		



次に、次元を減らしても  
識別に問題がないことを  
機械学習で確認してみる

# 概要

- 先の第1主成分～第3主成分を使って、花の種類を求める
- 3クラス分類問題である
- 機械学習器にはランダムフォレストを用いる

# 学習用のプログラム learning.py

データ読み込み関数

```
10 #-----
11 import numpy
12 import pandas
13 #from sklearn.ensemble import RandomForestRegressor as ml # こちらは回帰問題用
14 from sklearn.ensemble import RandomForestClassifier as ml # こちらは分類問題用
15
16
17 def read_training_data(teaching_file_path):
18     # 学習に必要な教師データを読み出す
19     data = pandas.read_csv(teaching_file_path, na_values='None', header=None, delimiter="\t")
20     data = data.dropna()
21
22     #print(data)
23     features = (data.iloc[:, 0:-1]).values # transform to ndarray
24     labels = (data.iloc[:, -1:]).values
25     labels = [flatten for inner in labels for flatten in inner] # transform 2次元 to 1次元 ぽいこと
26     #
27     print(features)
28     print(labels)
29     return (features, labels)
```

学習

```
32 def learn(teaching_data, save_file_path):
33     # 学習実行
34     trainFeature, trainLabel = teaching_data
35     clf = ml(max_features="auto")#, max_depth=7)
36     clf.fit(trainFeature, trainLabel)
```

学習済みオブジェクトの保存

```
37
38 # 学習成果を保存
39 import pickle
40 with open(save_file_path, 'wb') as f:
41     pickle.dump(clf, f)
```

正常かどうか確認

```
42
43 # 学習結果を確認のために表示
44 test = clf.predict([trainFeature[2]]) # 試しに一つ確認
45 print(test)
46 result = clf.score(trainFeature, trainLabel) # 学習データに対する、適合率
47 print(result)
48 print(clf.feature_importances_) # 各特徴量に対する寄与度を求める
49
50 return clf
```

```
51
52
53 def main():
54     training_data = read_training_data("feature_for_learning.csv")
55     learn(training_data, 'entry_temp.pickle')
56
57 if __name__ == '__main__':
58     main()
```

```

prediction.py
10 #-----
11 import pandas
12 import pickle
13
14
15 def read_training_data(teaching_file_path):
16     """ 学習に必要な教師データを読み出す
17     """
18     data = pandas.read_csv(teaching_file_path, na_values='None', header=None, delimiter="\t")
19     data = data.dropna()
20
21     #print(data)
22     features = (data.iloc[:, 0:-1]).values # transform to ndarray
23     labels = (data.iloc[:, -1:]).values
24     labels = [flatten for inner in labels for flatten in inner] # transform 2次元 to 1次元 ぽいこと
25     #print(labels)
26     return (features, labels)
27
28
29 def predict(clf, test_data):
30     """ 識別結果を返す
31     """
32     features, labels = test_data
33     results = clf.predict(features)
34
35     # 予測結果を保存
36     with open("result_temp.csv", "w") as fw:
37         fw.write("real value, Predicted value\n")
38         for predict_result, label in zip(results, labels):
39             fw.write(str(label))
40             fw.write(",")
41             fw.write(str(predict_result))
42             fw.write("\n")
43
44
45 def main():
46     # 機械学習オブジェクトを生成
47     clf = None
48     with open('entry_temp.pickle', 'rb') as f: # 学習成果を読み出す
49         clf = pickle.load(f) # オブジェクト復元
50
51     # テストデータの読み込み
52     test_data = read_training_data("feature_for_verify.csv")
53     predict(clf, test_data)
54
55 if __name__ == '__main__':
56     main()

```

# 検証用のプログラム prediction.py

データ読み込み関数

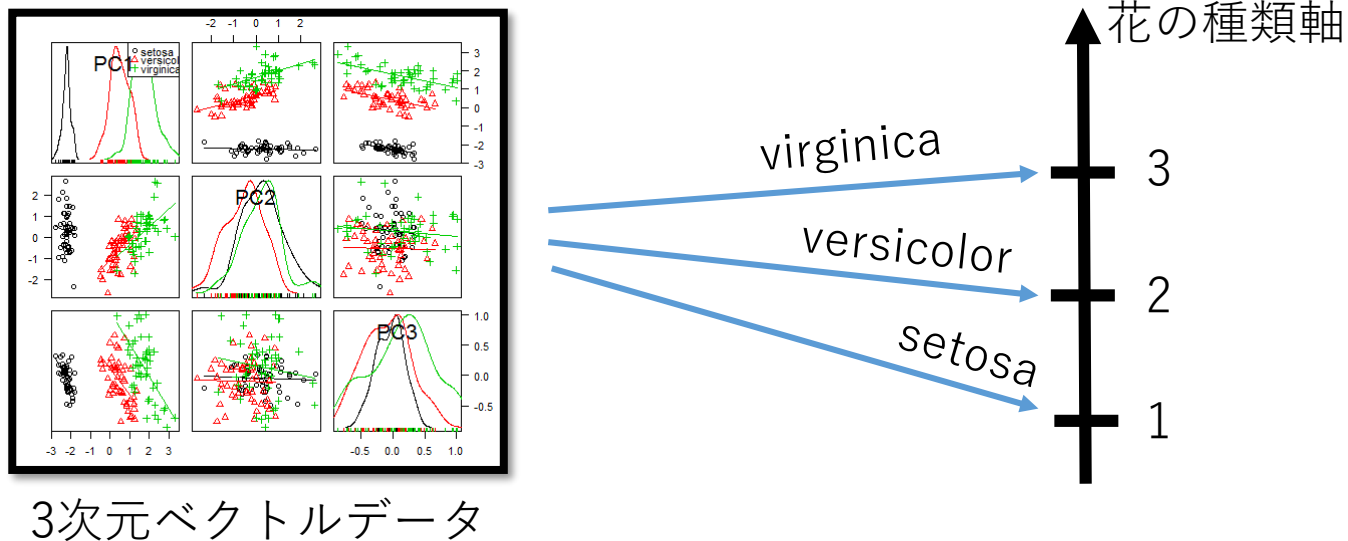
検証

検証結果の保存

保存したデータのロード

## \* コラム

ソフトウェアが数学的に何を指すのか眺めておきたい。



本計算例で利用する機械学習器は、3次元のベクトルを1次元に写像することを目指して、パターンを学習する。学習済み機械学習器は伝達関数そのものである。

# 学習用と検証用のデータ

学習成果の確認のために、tsvファイルの中身を「feature\_for\_learning.csv」と「feature\_for\_verify.csv」に分割保存する<sup>†</sup>。本計算例では、全データをランダムに並び替えた上で、検証用に2割のデータを抽出した。

feature_for_learning.csv				
1	-2.257141078	0.478423823	-0.127279553	1
2	-2.545111953	-0.47750103	0.304745622	1
3	2.252232135	1.914596315	0.396224434	3
4	0.158279772	-0.789451009	-0.301028699	2
5	-2.211043638	-0.726243194	-0.230140176	1
6	1.864257787	0.385674047	0.255418098	3
7	1.782378712	-0.186735626	0.269754249	3

学習用データ

feature_for_verify.csv				
1	-2.257141078	0.478423823	-0.127279553	1
2	0.574463756	-0.154356486	-0.27074337	2
3	0.255873363	-0.596852286	0.091572382	2
4	1.359620604	0.690443412	0.283661734	3
5	1.856483749	-0.177953327	0.352966182	3
6	0.069355967	-0.218770434	0.29060573	2
7	0.560231402	-1.75883213	-0.763214589	2

検証用データ

<sup>†</sup> csvファイルは本来ならカンマ区切りのテキストファイルを指すが、日本だけタブ区切りでも良いことになっている。

# 注意：想定しているデータのフォーマット

ここで、想定している教師データのフォーマットについて説明しておく。配布しているプログラムが想定しているフォーマットでは、1行目にヘッダー（項目名）はない。

以下は「learning.py」内に記述された、教師データを読み込む命令である。引数の中に「header=None」とあるが、これは「ヘッダー（項目名）なし」を指示している。もし、1行目に列の名前を入れる場合は、「header=0」とする。

```
data = pandas.read_csv(teaching_file_path, na_values='None', header=None, delimiter="\t")
```

\*引数に名前を指定して渡すやり方を「名前付き引数」という。モダンな言語は実装している。

# 学習

「learning.py」を用いて、学習データを学習させる。  
「learning.py」を実行すれば、自動的に  
「feature\_for\_learning.csv」が読み込まれ、記述された命令に従って学習が行われる。学習済みの機械学習器はバイナリ形式で「entry\_temp.pickle」に保存される。



# 検証

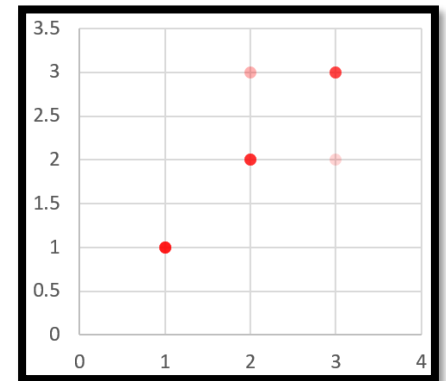
「prediction.py」を用いて、学習済みの機械学習機を用いて検証用のデータを識別した。識別結果は「result\_temp.csv」に保存されている。Excelでこれを開いた様子を以下に示す。

図において、A列が真値（教師ラベル）を表しており、B列が予測結果を示している。散布図を見る限り、誤判定はあるもののほぼうまく判定できているようだ。

これにより、主成分分析によって必要な情報を落とさずに次元の数を減らすことができたことが分かる。

	A	B	C
1	real value	Predicted value	
2	1	1	
3	2	2	
4	2	2	
5	3	3	
6	3	3	
7	2	2	
8	2	2	
9	1	1	

result\_temp.csv



真値と予測の散布図

# 参考文献

機 械 学 習 による  
データ分析 **まわり** のお話

@canard0328

◀ 1 of 75 ▶



<http://www.slideshare.net/canard0328/ss-44288984>

決定係数とは？

## 高校数学の基本問題

[http://www.geisya.or.jp/~mwm48961/koukou/index\\_m.htm#linear](http://www.geisya.or.jp/~mwm48961/koukou/index_m.htm#linear)

\* 寄与率に関する数学的な定義も  
記述されていて、わかり易い。