数理生物学演習　第１３回

次元削減による解析

asai@moprphometrics.jp
数理生物学演習　修士１年　浅井駿平

本日の内容

１、 scikit-learnのデータセットを実際に触ってみよう
　　データフレームを使った次元削減

２、研究紹介
　　イチゴ果実の品種間・等級間での形状差

次元削減とは？

元のデータの意味のある要素を把握した上で、少ない特徴（次元）を使って、データの特徴を表現する方法

具体例

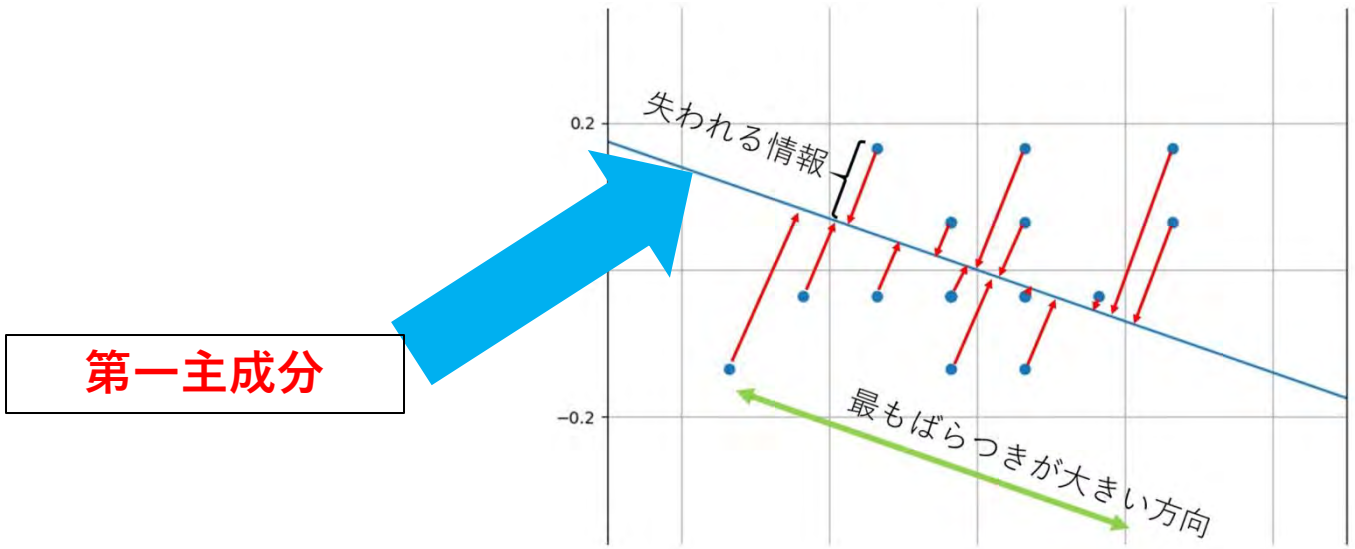
- PCA（主成分分析）
- 線形判別分析（LDA）
- t-SNE
- UMAP　　などなど

PCAとは？

PCA（主成分分析）・・射影したときのデータの分散が最大になる軸を第一主成分とする。

必要なパッケージ

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

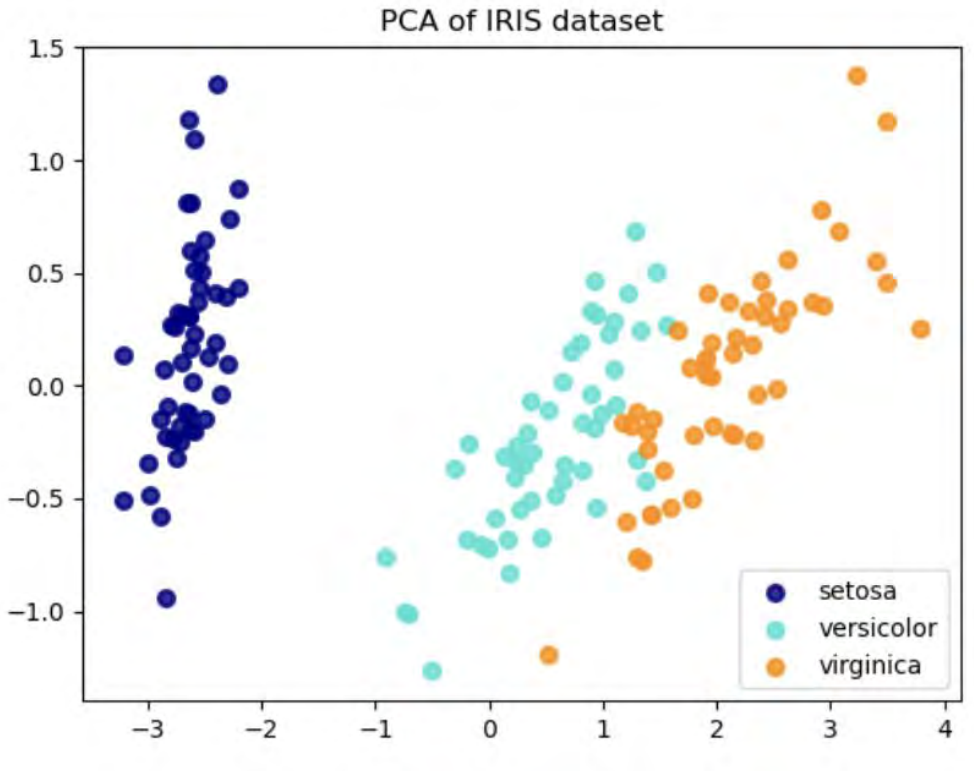


CORE CONCEPT TECHNOLOGIES INC.
主成分分析とは何なのか、とにかく全力でわかりやすく解説する,2018.08.10
https://recruit.cct-inc.co.jp/tecblog/machine-learning/pca-kaisetsu/

・高次元のデータを可視化したり、情報を保ちながら次元削減をしたい。
・変数間の相互関係を理解したい。

scikit-learnとは？

Python用の機械学習ライブラリで誰でも利用することができる。
サンプルのデータセットが豊富にある。
機械学習のプログラミングを手軽に試すことができる。



今回使うデータセット

アヤメの３種類の品種の花弁の長さと、がくの長さで構成されてれいるデータセット

```
from sklearn.datasets import load_iris
print(load_iris())
```


```
{'data': array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2],
       [5.4, 3.9, 1.7, 0.4],
       [4.6, 3.4, 1.4, 0.3],
       [5. , 3.4, 1.5, 0.2],
       [4.4, 2.9, 1.4, 0.2],
       [4.9, 3.1, 1.5, 0.1],
       [5.4, 3.7, 1.5, 0.2],
       [4.8, 3.4, 1.6, 0.2],
       [4.8, 3. , 1.4, 0.1],
       [4.3, 3. , 1.1, 0.1],
       [5.8, 4. , 1.2, 0.2],
       [5.7, 4.4, 1.5, 0.4],
       [5.4, 3.9, 1.3, 0.4],
       [5.1, 3.5, 1.4, 0.3],
       [5.7, 3.8, 1.7, 0.3],
```

このままだと扱いづらいので、処理が必要

pandasとは？

多くのデータを一つのデータフレーム（表形式）で扱うことができる。
異なる型のデータでもまとめることができる。
また、データの加工（整列、削除、列の入れ替え）がしやすい。

機械学習や解析の前段階の処理を効率的に行うことができる。

データフレームとは？

行と列で構成された表形式のデータ構造
ExcelやGoogleスプレッドシートみたいなもの

Irisデータセットのデータフレーム

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	label
0	5.1	3.5	1.4	0.2	0
1	4.9	3	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	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
7	5	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0
10	5.4	3.7	1.5	0.2	0
11	4.8	3.4	1.6	0.2	0
12	4.8	3	1.4	0.1	0
13	4.3	3	1.1	0.1	0
14	5.8	4	1.2	0.2	0
15	5.7	4.4	1.5	0.4	0
16	5.4	3.9	1.3	0.4	0
17	5.1	3.5	1.4	0.3	0
18	5.7	3.8	1.7	0.3	0
19	5.1	3.8	1.5	0.3	0
20	5.4	3.4	1.7	0.2	0
21	5.1	3.7	1.5	0.4	0
22	4.6	3.6	1	0.2	0
23	5.1	3.3	1.7	0.5	0
24	4.8	3.4	1.9	0.2	0
25	5	3	1.6	0.2	0
26	5	3.4	1.6	0.4	0
27	5.2	3.5	1.5	0.2	0

メリット
・データの並べ替え
・行や列の削除
・データの結合

データセットの読み込みと表形式化

```
from sklearn.datasets import load_iris


import pandas as pd
import matplotlib.pyplot as plt
iris = load_iris()
df = pd.DataFrame(data = iris.data,
columns=iris.feature_names)
df['label'] = iris.target
df
```

データセットを使ったPCA解析

```
pca = PCA(n_components=2)
x = df.iloc[:, 0:4]
pca.fit(x)
pca.components_.T
```

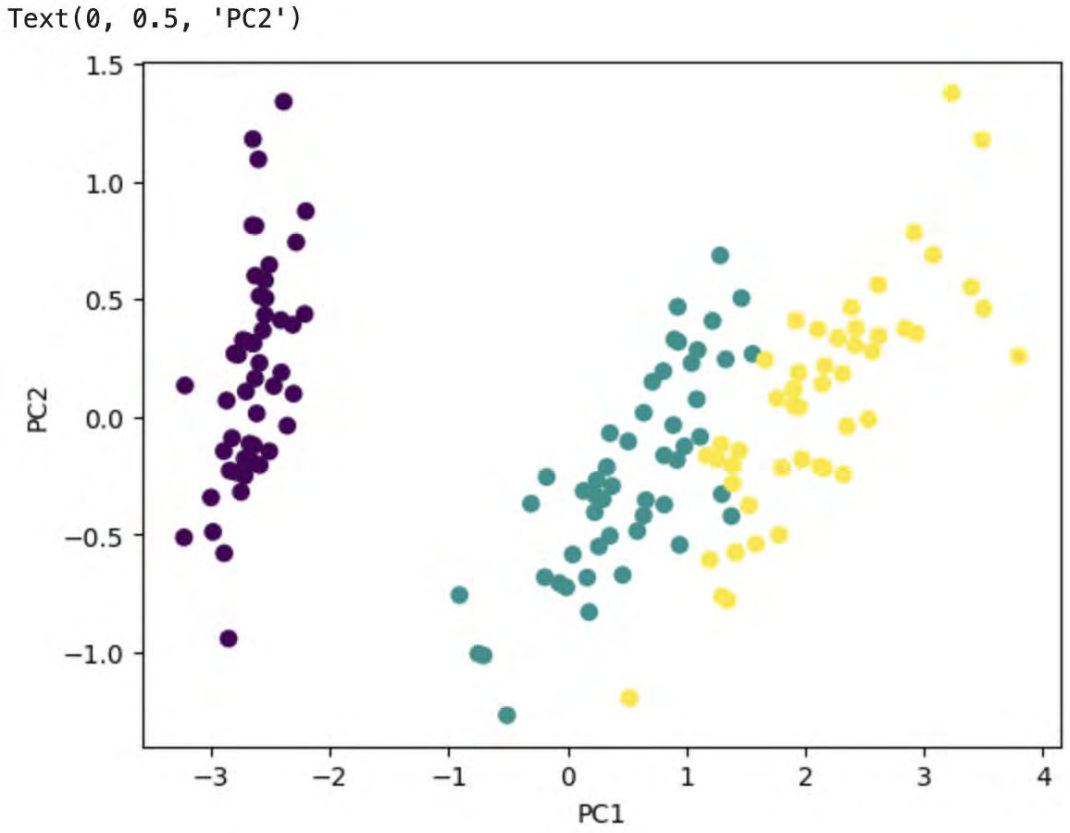
```
pca = PCA(n_components=2)
x = df.iloc[:, 0:4]
pca.fit(x)
pca.components_.T
```

```
array([[ 0.36138659,  0.65658877],
       [-0.08452251,  0.73016143],
       [ 0.85667061, -0.17337266],
       [ 0.3582892 , -0.07548102]])
```

- PCAの主成分を2つに抽出する。（2次元圧縮）
- 「fit」は、PCAを行うための主成分の軸、平均値などを学習している。

PCA解析の結果の可視化

```
#次元削減(PCA)
X = pca.transform(x)
#相関図の描画
plt.scatter(x=X[:, 0],
y=X[:, 1], c=iris.target)
plt.xlabel('PC1')
plt.ylabel('PC2')
```



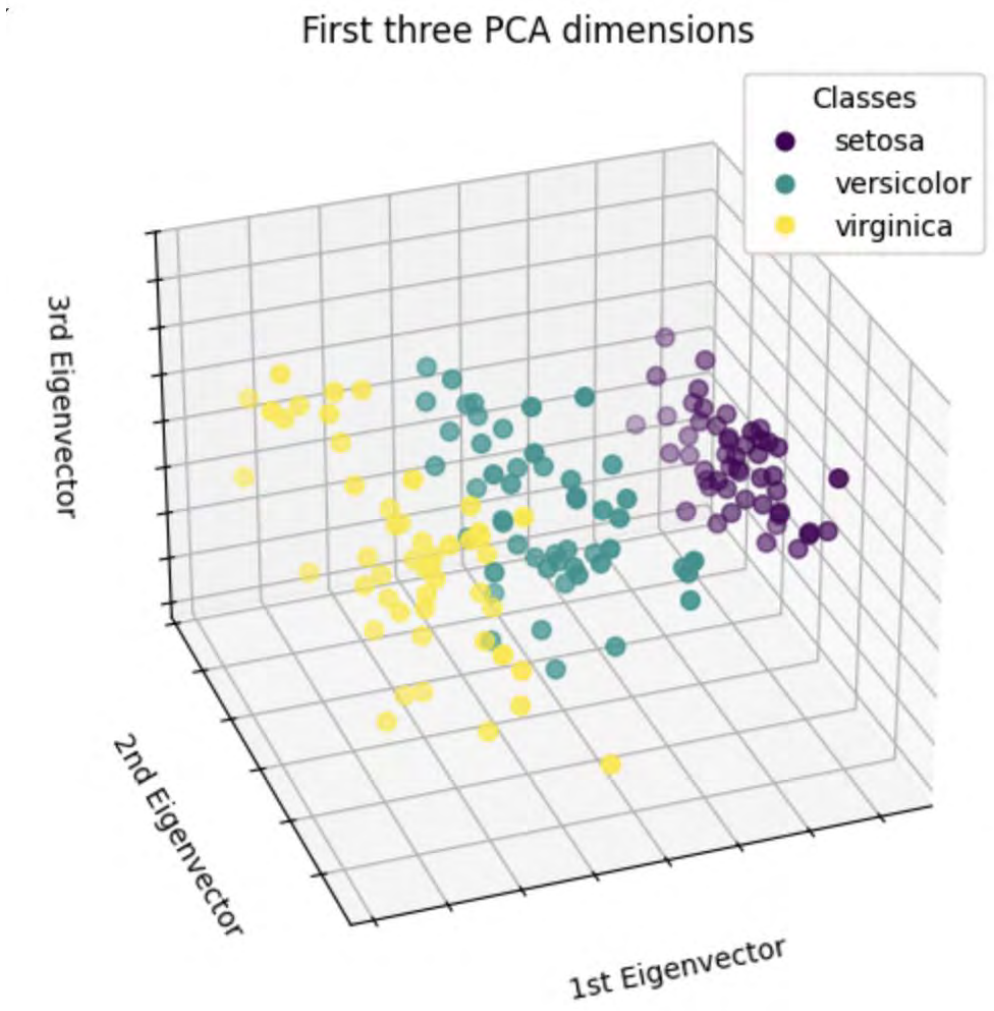
「transform」によって、データを主成分軸に沿って、変換する。

複数の主成分の可視化

```
import mpl_toolkits.mplot3d
# 描画
fig = plt.figure(1, figsize=(8, 6))
ax = fig.add_subplot(111, projection="3d", elev=-150, azim=110)
#PCA
X_reduced = PCA(n_components=3).fit_transform(iris.data)
scatter = ax.scatter(
    X_reduced[:, 0],
    X_reduced[:, 1],
    X_reduced[:, 2],
    c=iris.target,
    s=40,
)
# ラベルの追加
ax.set(
    title="First three PCA dimensions",
    xlabel="1st Eigenvector",
    ylabel="2nd Eigenvector",
    zlabel="3rd Eigenvector",)
```

```
ax.xaxis.set_ticklabels([])
ax.yaxis.set_ticklabels([])
ax.zaxis.set_ticklabels([])
# レジェンドの追加
legend1 = ax.legend(
    scatter.legend_elements()[0],
    iris.target_names.tolist(),
    loc="upper right",
    title="Classes",
)
ax.add_artist(legend1)

plt.show()
```

出力結果



PCAの複数の主成分を同時に可視化することができる。

複数の主成分同士の関係を可視化

```
import seaborn as sns
import pandas as pd
from sklearn.preprocessing import StandardScaler
# データ準備
iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names


# 標準化 → PCA(3次元)
X_scaled = StandardScaler().fit_transform(X)
pca = PCA(n_components=3)
X_pca = pca.fit_transform(X_scaled)
```

```
# データフレームに変換
df_pca = pd.DataFrame(X_pca, columns=['PC1', 'PC2', 'PC3'])
df_pca['label'] = y
```

```
#データフレーム確認
df_pca
```

	PC1	PC2	PC3	label
0	-2.264703	0.480027	0.127706	0
1	-2.080961	-0.674134	0.234609	0
2	-2.364229	-0.341908	-0.044201	0
3	-2.299384	-0.597395	-0.091290	0
4	-2.389842	0.646835	-0.015738	0
...	...	...	...	...
145	1.870503	0.386966	-0.256274	2
146	1.564580	-0.896687	0.026371	2
147	1.521170	0.269069	-0.180178	2
148	1.372788	1.011254	-0.933395	2
149	0.960656	-0.024332	-0.528249	2

150 rows × 4 columns

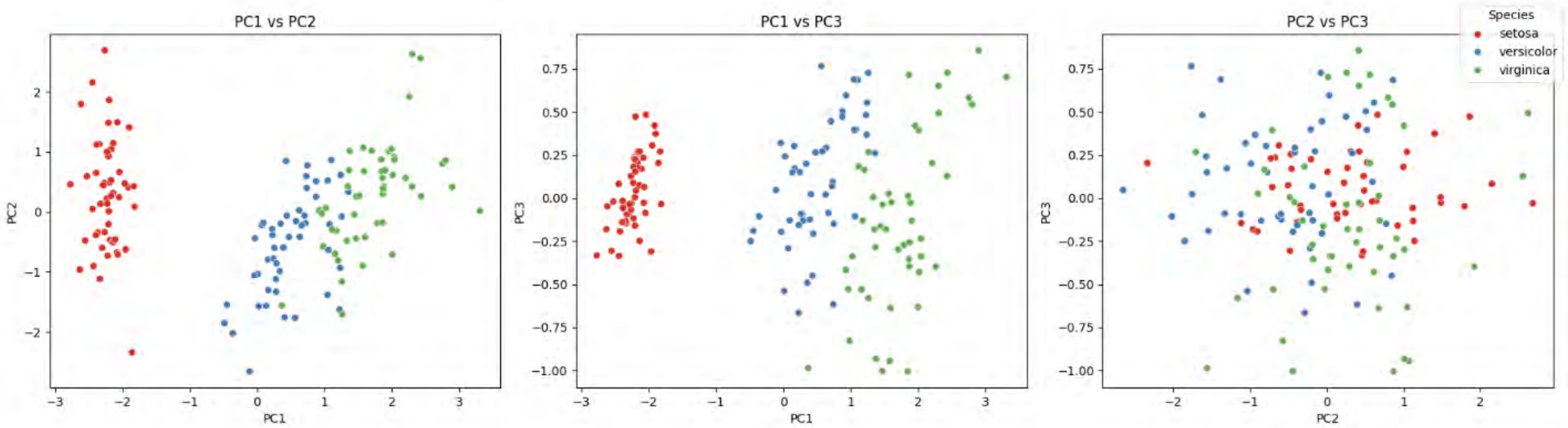
複数の主成分同士の関係を可視化

```
# 描画
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
#PC1とPC2の描画
sns.scatterplot(ax=axes[0], data=df_pca, x='PC1',
y='PC2', hue='label', palette='Set1')
axes[0].set_title('PC1 vs PC2’)
#PC1とPC3の描画
sns.scatterplot(ax=axes[1], data=df_pca, x='PC1',
y='PC3', hue='label', palette='Set1')
axes[1].set_title('PC1 vs PC3’)
#PC2とPC3の描画
sns.scatterplot(ax=axes[2], data=df_pca, x='PC2',
y='PC3', hue='label', palette='Set1')
axes[2].set_title('PC2 vs PC3’)
```

```
# 凡例をひとつにまとめる
handles, labels =
axes[0].get_legend_handles_labels()
for ax in axes:
    ax.legend_.remove()
fig.legend(handles, [target_names[int(label)]
for label in set(y)], title="Species",
loc='upper right')


plt.tight_layout()
plt.show()
```

hue・・色分けに使う関数

複数の主成分同士の関係を可視化



同時に複数の主成分をプロットすることで、それぞれの
主成分同士の相関を調べることができる。

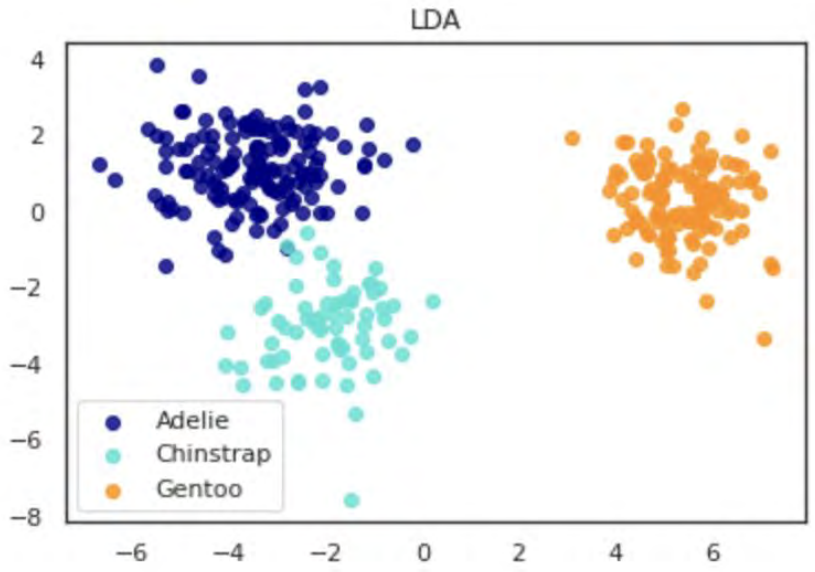
次元削減とは？

元のデータの意味のある要素を把握した上で、少ない特徴（次元）を使って、データの特徴を表現する方法

具体例
- PCA（主成分分析）
- 線形判別分析（LDA）
- t-SNE
- UMAP　　などなど

LDAとは？

LDA（線形判別分析）・・グループ間の分散を最大に、グループの分類を最小にする軸を選ぶ。



必要なパッケージ

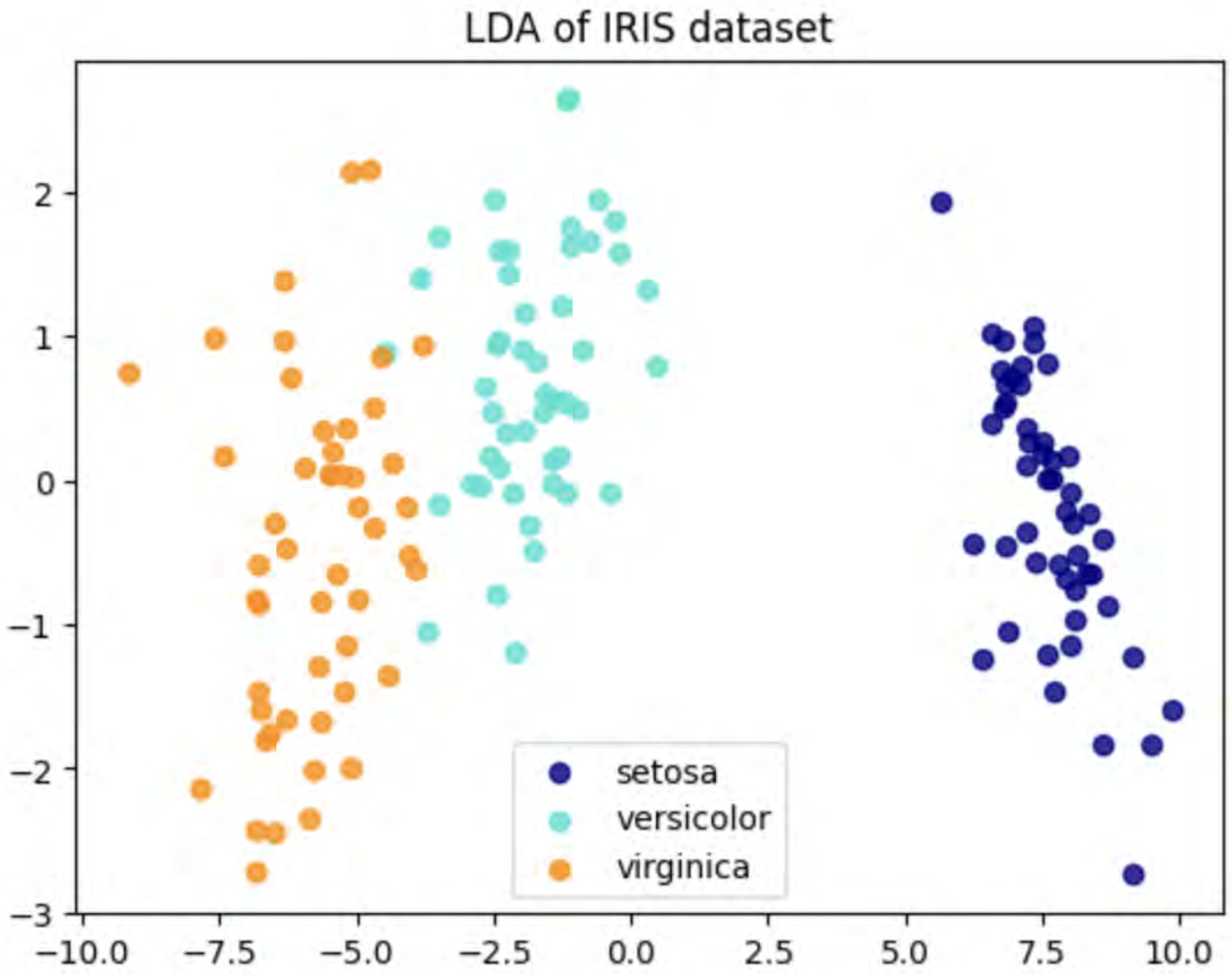
```
from sklearn.discriminant_analysis
import LinearDiscriminantAnalysis
```

・グループ間の違いを強調した図を作りたい。
・クラス（品種など）分類の前処理として使いたい。

LDAの可視化

```
import matplotlib.pyplot as plt

from sklearn import datasets

#データセットの用意
iris = datasets.load_iris()

X = iris.data
y = iris.target
target_names = iris.target_names

#データセットにLDAを適用
lda = LinearDiscriminantAnalysis(n_components=2)
X_r2 = lda.fit(X, y).transform(X)

# 可視化
for color, i, target_name in zip(colors, [0, 1, 2], target_names):
plt.scatter(
X_r2[y == i, 0], X_r2[y == i, 1], alpha=0.8, color=color, label=target_name
)
plt.legend(loc="best", shadow=False, scatterpoints=1)
plt.title("LDA of IRIS dataset")

plt.show()
```

LDAの可視化


LDA of IRIS dataset
setosa
versicolor
virginica
2
1
0
−1
−2
−3
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

課題

Normal

1、サンプルコードを実行して、プロットされたPCAとLDAの違いと、プロットから読み取れるペンギンの種の違いについて簡単に記述してください。
2、質問、感想などお願いします。

Hard

https://scikit-learn.org/stable/datasets.html
に載っているscikit-learnのデータセット（アイリス（アヤメ）の種類以外）を使って、PCAのプロットをしてください。
使い方の参考：https://note.nkmk.me/python-sklearn-datasets-load-fetch/