

Workout Report

最終報告会

M1 高野 修平

全体の目次

- 線形代数の基礎から主成分分析まで
- 任意の数式、モデルの実装
- 前処理の違いによるPCAの結果の変化

報告会の目的

- 自分が得たスキルセットを共有する
- 特に頑張ったところを共有する

Workout Reportメニュー表

	ビギナー	常人	玄人	達人	神
準備	Python, Git, Github	/	/	/	/
線形代数	ベクトル空間	線形変換	特異値分解と主成分分析	ガウス過程	情報幾何
ベイズ (確率)	ベイズルール	点推定	分布推定	ガウス過程	モデル選択
前処理	正規化	データセット	生データ	特徴量エンジニアリング	表現学習
モデル	実装済みモデルの利用	kerasによる実装	スクラッチ実装	問題設定 (教師あり)	問題設定 (教師なし)
検証	描画	評価指標 (教師あり)	評価指標 (教師なし)	実験デザイン (教師あり)	実験デザイン (教師なし)

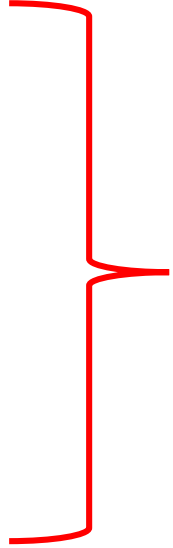
スキルセット表(線形代数)

線形 代数	ベクトル空間	線形変換	特異値分解と主成分分析	ガウス 過程	情報 幾何
	<ul style="list-style-type: none">- ベクトル空間が何かを説明できる- ベクトル空間に標準内積を導入できる	<ul style="list-style-type: none">- 写像が説明できる- 射影が説明できる- 線形変換の定義が説明できる	<ul style="list-style-type: none">- 線形次元削減の定義を説明できる- データ行列の分散共分散行列を対角化して主成分を求めることができる- SVDを用いて主成分を求めることができる- 求めた主成分を用いて次元削減ができる		
獲得した			獲得中	獲得していない	

理論の目次

線形代数

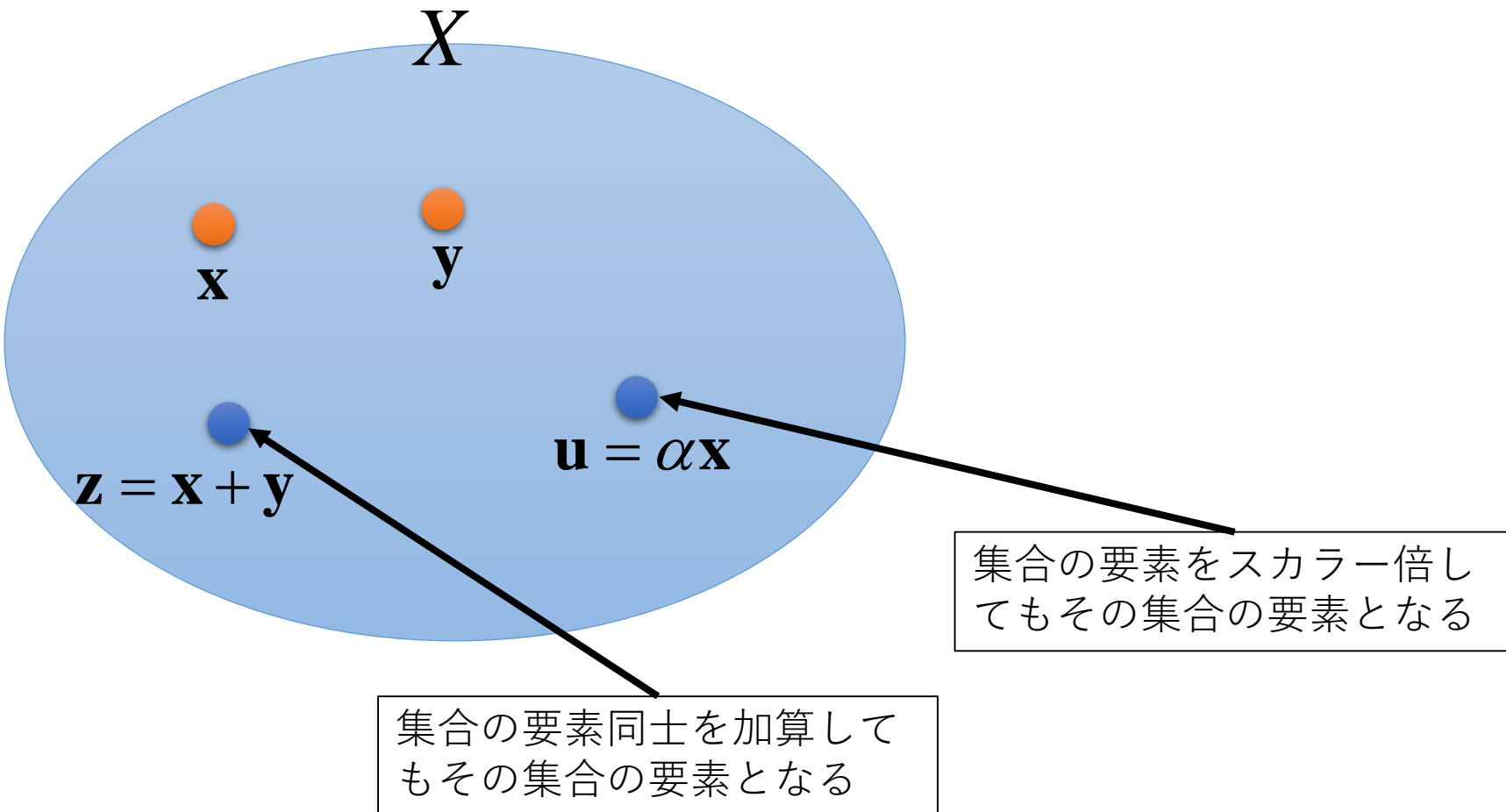
- ベクトル空間とは
- 写像とは
- 射影とは
- 線形変換の定義
- 線形変換における固有値の特性
- 求めた主成分で主成分分析



ダイジェスト

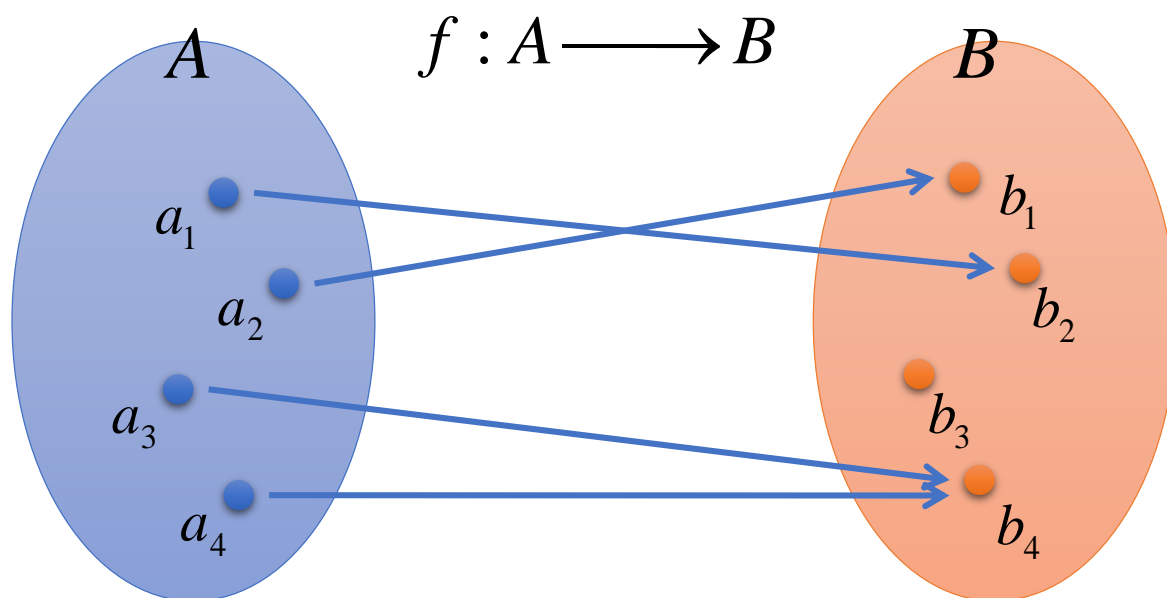
ベクトル空間とは

- 集合 X の任意の要素について加算とスカラー倍が定義できる空間(講義スライドより)



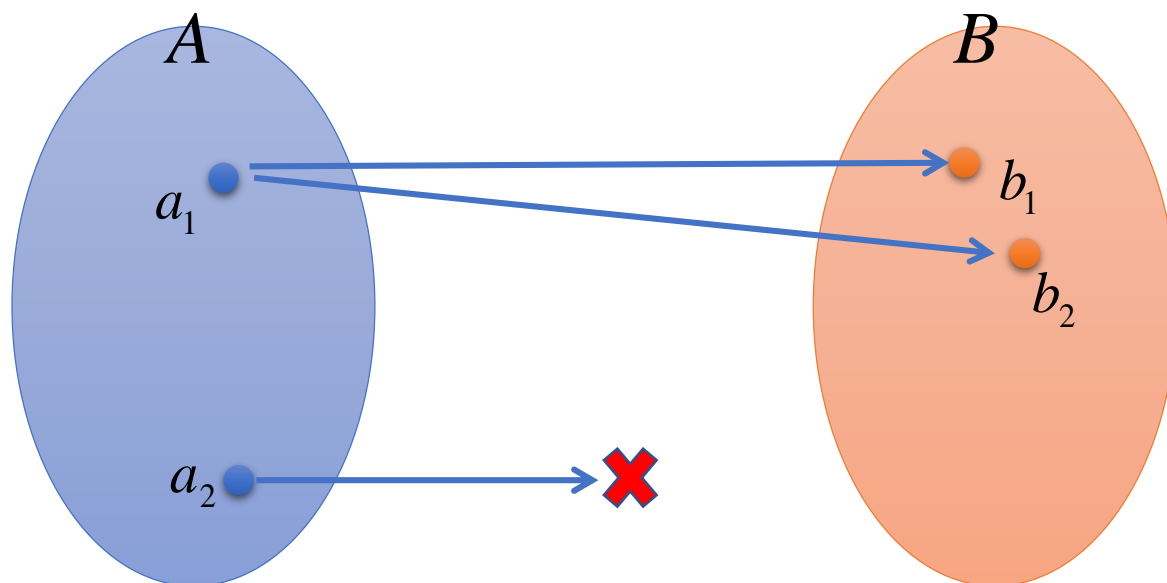
写像とは

- 集合 A, B に対し, A の各要素に対したただひとつの B の要素を対応させる規則 f が与えられたとき, f を A から B への写像と呼ぶ(講義スライドより)



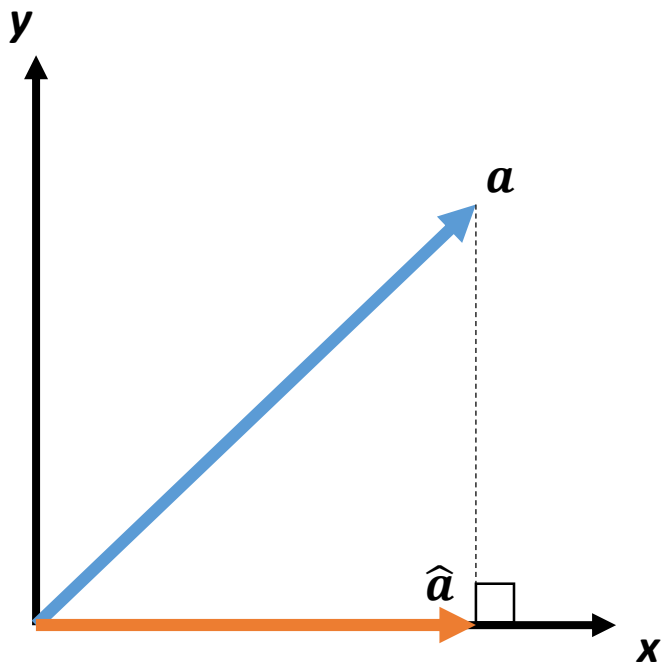
写像とは

- 集合 A, B に対し, A の各要素に対したただひとつの B の要素を対応させる規則 f が与えられたとき, f を A から B への写像と呼ぶ(講義スライドより)

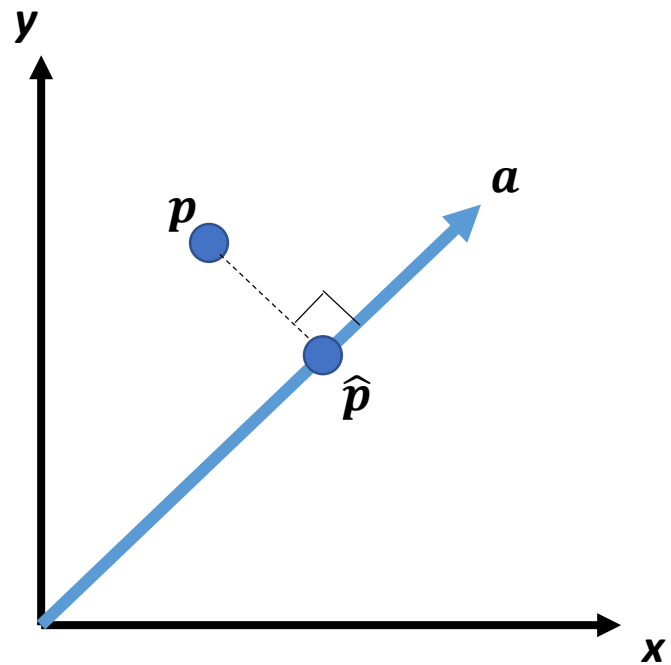


ある A の要素から B への対応する写像がなかったり、 B の2つの要素が対応してはならない

射影とは



ベクトル a を x 軸へ正射影



データ点 p をベクトル a へ正射影

線形変換の定義

- 線形変換とは

$$f(ax_1) = af(x_1)$$

入力を a 倍したら出力も a 倍

$$f(x_1 + x_2) = f(x_1) + f(x_2)$$

入力を足し合わせたら出力も足し合わされる

これら2つの性質(線形性)を満たす f のことを線形変換と呼ぶ

(例)比例 $y = ax$

$$y = a(3x) = 3ax$$

$$y = a(x_1 + x_2) = a(x_1) + a(x_2)$$

(例)行列による線形変換 $y = Ax$

$$\begin{pmatrix} 4 & -2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 6 \end{pmatrix} = 3 \begin{pmatrix} 4 & -2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 0 \\ 9 \end{pmatrix}$$

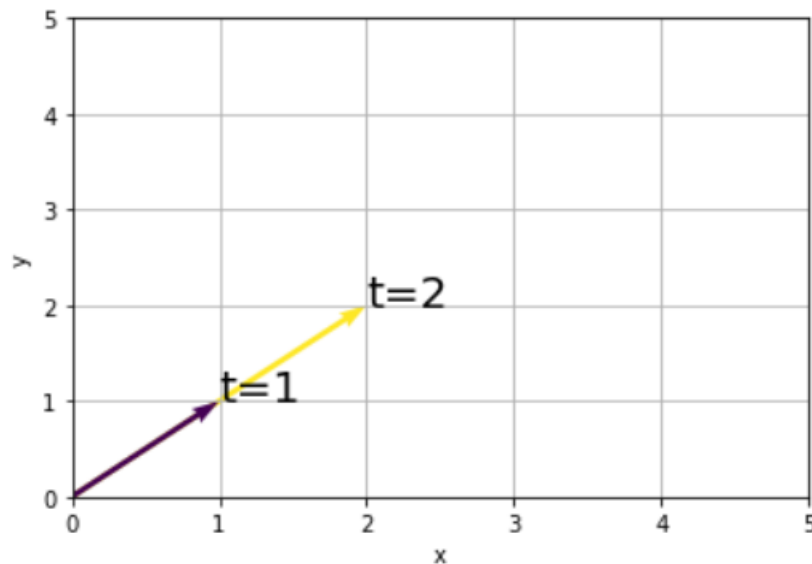
$$\begin{pmatrix} 4 & -2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1+3 \\ 2+2 \end{pmatrix} = \begin{pmatrix} 4 & -2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} + \begin{pmatrix} 4 & -2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 3 \\ 2 \end{pmatrix} = \begin{pmatrix} 8 \\ 8 \end{pmatrix}$$

固有値・固有ベクトル

- 線形変換**A**には変換後にも向きが変わらないベクトルが存在する場合がある

$$\begin{pmatrix} 4 & -2 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = 2 \cdot \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad A\mathbf{x} = \lambda\mathbf{x}$$

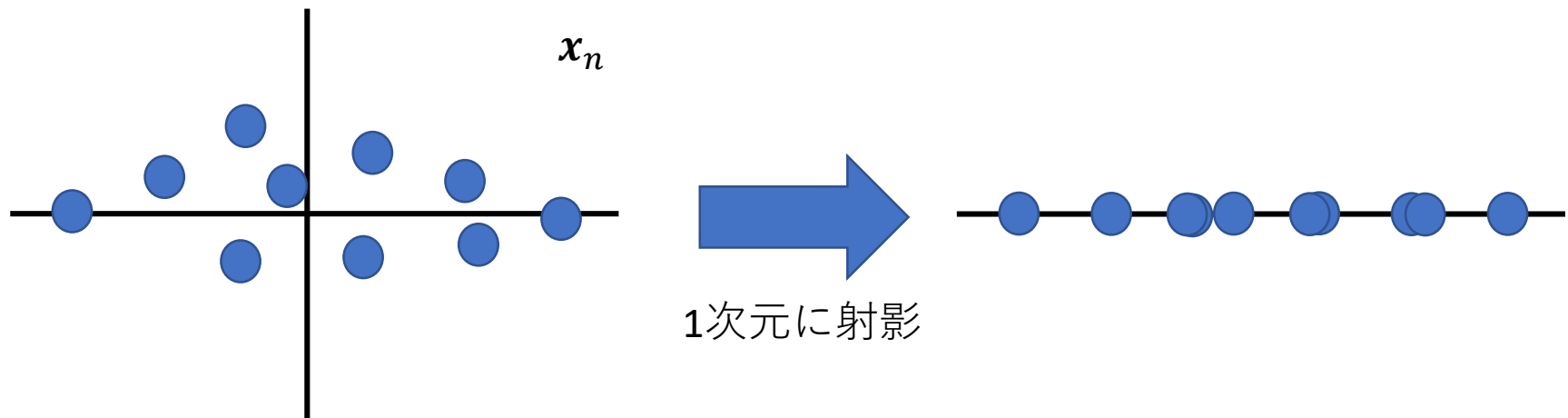
固有値問題



求めた主成分で主成分分析

- 分散最大化による主成分の導出
- 問題設定

あるデータセット $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ が与えられている
D次元データ $\mathbf{x}_n \in \mathbb{R}^D$ を1次元空間に射影する
射影後のデータの分散が大きくなるように射影する

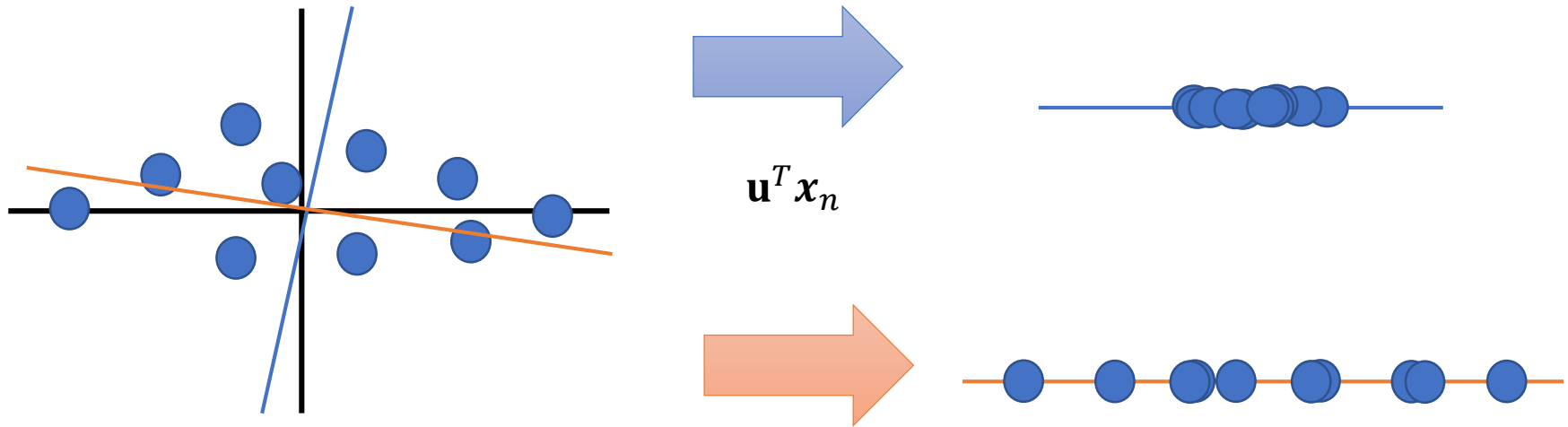


求めた主成分で主成分分析

- 分散最大化による主成分の導出

データを1次元に射影するためにD次元ベクトル $\mathbf{u} \in \mathbb{R}^D$ を導入する
 \mathbf{u} と \mathbf{x}_n の内積をとることで1次元に射影する。

射影後のデータの分散 $\frac{1}{N} \sum_{n=1}^N (\mathbf{u}^T \mathbf{x}_n - \mathbf{u}^T \bar{\mathbf{x}})^2$ できるだけ大きくしたい
 $\bar{\mathbf{x}}$: データの平均



求めた主成分で主成分分析

- 分散最大化による主成分の導出

$$\begin{aligned} \frac{1}{N} \sum_{n=1}^N (\mathbf{u}^T \mathbf{x}_n - \mathbf{u}^T \bar{\mathbf{x}})^2 &= \frac{1}{N} \sum_{n=1}^N (\mathbf{u}^T \mathbf{x}_n - \mathbf{u}^T \bar{\mathbf{x}})(\mathbf{u}^T \mathbf{x}_n - \mathbf{u}^T \bar{\mathbf{x}})^T \\ &= \frac{1}{N} \sum_{n=1}^N \{(\mathbf{u}^T \mathbf{x}_n)(\mathbf{u}^T \mathbf{x}_n)^T - (\mathbf{u}^T \mathbf{x}_n)(\mathbf{u}^T \bar{\mathbf{x}})^T - (\mathbf{u}^T \bar{\mathbf{x}})(\mathbf{u}^T \mathbf{x}_n)^T + (\mathbf{u}^T \bar{\mathbf{x}})(\mathbf{u}^T \bar{\mathbf{x}})^T\} \\ &= \frac{1}{N} \sum_{n=1}^N \{(\mathbf{u}^T \mathbf{x}_n \mathbf{x}_n^T \mathbf{u}) - 2(\mathbf{u}^T \mathbf{x}_n \bar{\mathbf{x}}^T \mathbf{u}) + (\mathbf{u}^T \bar{\mathbf{x}} \bar{\mathbf{x}}^T \mathbf{u})\} \\ &= \frac{1}{N} \sum_{n=1}^N \mathbf{u}^T (\mathbf{x}_n \mathbf{x}_n^T - 2\mathbf{x}_n \bar{\mathbf{x}}^T + \bar{\mathbf{x}} \bar{\mathbf{x}}^T) \mathbf{u} \\ &= \mathbf{u}^T \underbrace{\left\{ \frac{1}{N} \sum_{n=1}^N (\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T \right\}}_{\text{共分散行列 } S} \mathbf{u} = \boxed{\mathbf{u}^T S \mathbf{u}} \end{aligned}$$

これを最大化

求めた主成分で主成分分析

- 分散最大化による主成分の導出

$$\mathbf{u}^T S \mathbf{u}$$



制約 $\mathbf{u}^T \mathbf{u} = 1$ を追加

$$\mathbf{u}^T S \mathbf{u} + \lambda(1 - \mathbf{u}^T \mathbf{u})$$

\mathbf{u} で偏微分



$$S\mathbf{u} - \lambda\mathbf{u} = 0 \rightarrow S\mathbf{u} = \lambda\mathbf{u}$$

固有値問題 λ は S の固有値



左から \mathbf{u}^T をかけ、 \mathbf{u} を単位ベクトルとする($\mathbf{u}^T \mathbf{u} = 1$)

$$\mathbf{u}^T S \mathbf{u} = \lambda \mathbf{u}^T \mathbf{u} = \lambda$$

分散を最大化

→ 共分散行列の固有値の中で一番大きいものに対応する固有ベクトルにデータを射影する

S はデータによって決まっている
 \mathbf{u} を最大化するしかない
それも $\mathbf{u} = \infty$ とならないように

スキルセット表(モデル)

モデル	獲得した		獲得中	獲得していない	
	実装済みモデルの利用 - irisデータセットに対してPCAをかける - Boston house-pricesに対して線形回帰を行い予測できる	kerasを用いたモデルの実装 - 任意のニューラルネットワークを構築できる	スクラッチ実装 - 任意の数式をnumpy, tensorflowなどを用いて実装できる	問題設定 (教師あり)	問題設定 (教師なし)

モデルの目次

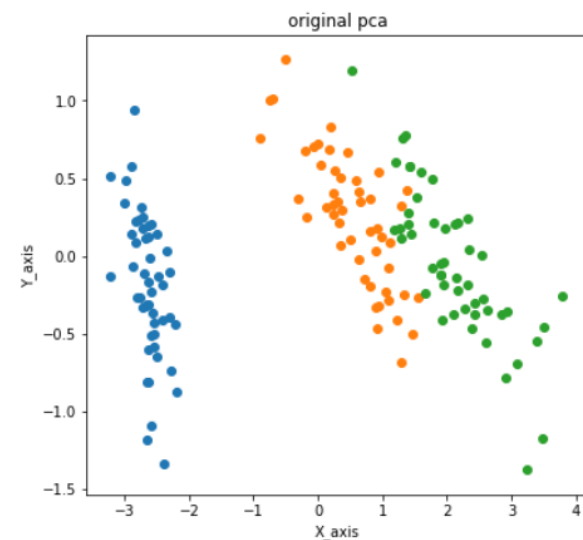
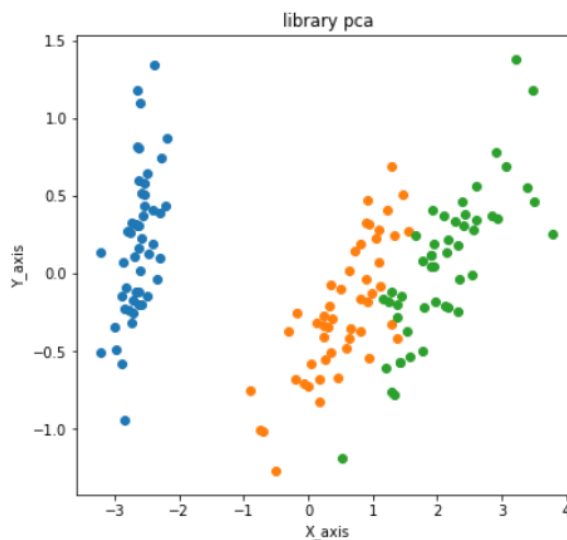
- 任意の数式を実装(PCA)
- 任意の数式を実装(SOM)
- 任意のニューラルネットワークを構築(VAE)

任意の数式を実装

- 主成分分析

```
import numpy as np#numpyをインストール
import matplotlib.pyplot as plt#グラフを書く
from mpl_toolkits.mplot3d import Axes3D#3D
from sklearn import decomposition#PCAのラ
from sklearn import datasets#datasetsを取

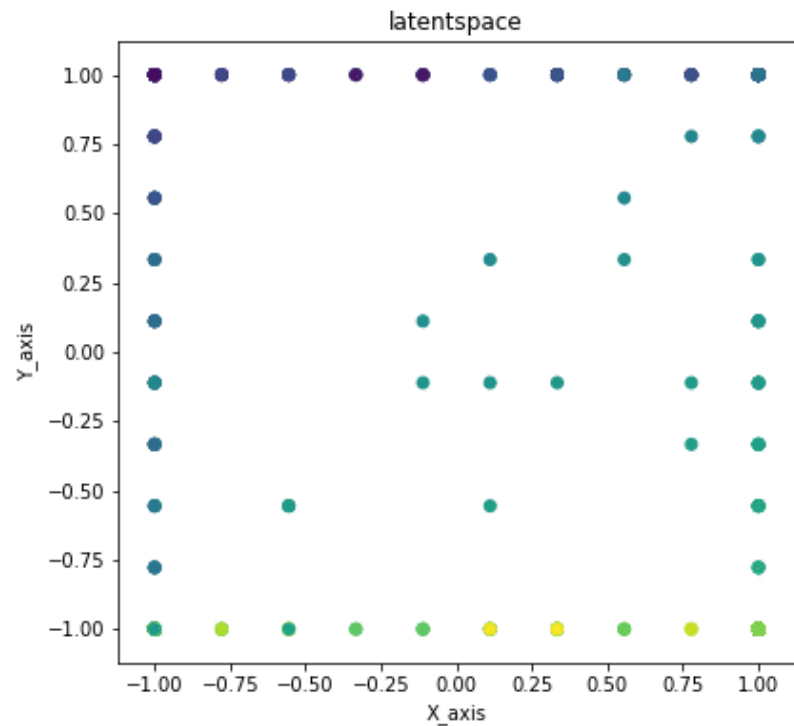
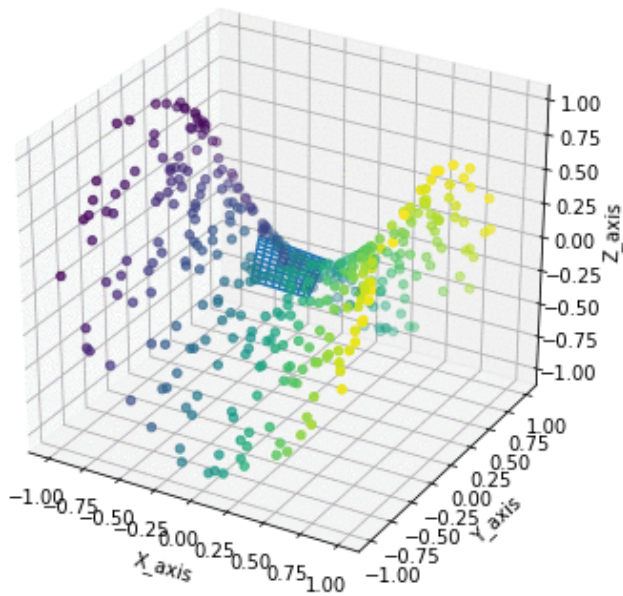
n_component=2
np.set_printoptions(suppress=True)
fig = plt.figure(2, figsize=(14, 6))
iris = datasets.load_iris()
X = iris.data
y = iris.target
for i in range(4):#各データの平均を元のデー
    mean = np.mean(X[:,i])
    X[:,i]=(X[:,i]-mean)
X_cov=np.dot(X.T,X)#共分散行列を生成
w,v=np.linalg.eig(X_cov)#共分散行列の固有値
for i in range(n_component):#固有値の大き
    Xpc[i]=v[:,i]
Xpc=np.array(Xpc)
Xafter=np.dot(X,Xpc.T)#取り出した固有ベクトル
```



SklearnのPCAは第1主成分の固有ベクトルの第1要素が正となるように固有ベクトルを定めている

任意の数式を実装

- SOM



任意のニューラルネットワークを構築

- VAEをKerasで構築した
- CIFAR10で精度が出なかったため精度の向上に執着した
- エンコーダー、デコーダーの層を変化させたり、潜在空間の次元を変えたりして、精度の向上を目指した

エンコーダー:CNN4層,全結合層1層

デコーダー:CNN2層,全結合層1層

潜在空間の次元:2次元



任意のニューラルネットワークを構築

エンコーダー:CNN4層,全結合層1層

デコーダー:CNN2層,全結合層1層

潜在空間の次元:64次元



エンコーダー:CNN5層,全結合層1層

デコーダー:CNN5層,全結合層3層

潜在空間の次元:64次元



任意のニューラルネットワークを構築

エンコーダー:CNN5層,全結合層1層

デコーダー:CNN5層,全結合層3層

潜在空間の次元:64次元



エンコーダー:CNN5層,全結合層1層

デコーダー:CNN5層,全結合層3層

潜在空間の次元:64次元

最適化関数:Adam→RMSProp



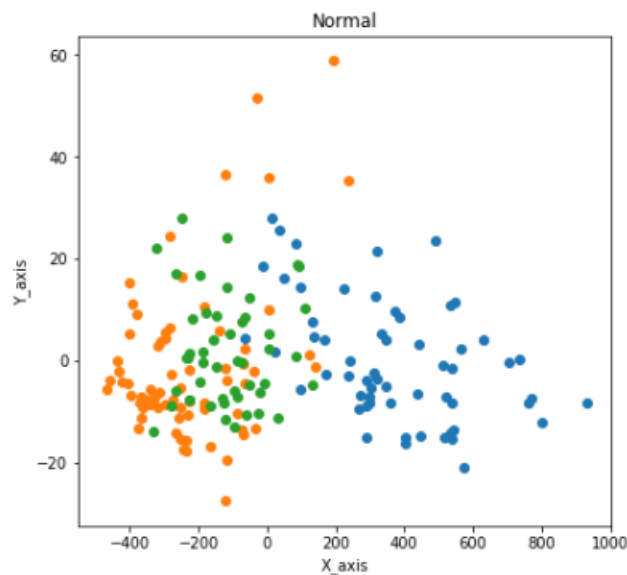
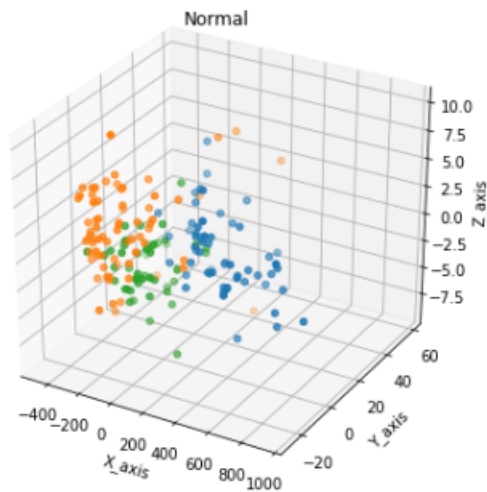
スキルセット表(前処理)

前処理	獲得した		獲得していない		
	正規化	データセット	生データ	特徴量エンジニアリング	表現学習
	<ul style="list-style-type: none">- 与えられたnumpy arrayを平均0, 分散1に標準化できる	<ul style="list-style-type: none">- sklearnやUCIなどの任意の公開データセットデータを「使える」- タスクに応じて適切なデータセットを調べられる- 標準的な形式のファイルを読み込める	<ul style="list-style-type: none">- KaggleやEnronデータセットなどの生データからデータ整形・前処理を行い使用できる- 任意のデータセットを整形できる- 整形の工程管理ができる		

sklearnを使える(前処理：常人)

- SklearnのワインデータをPCAにかけた

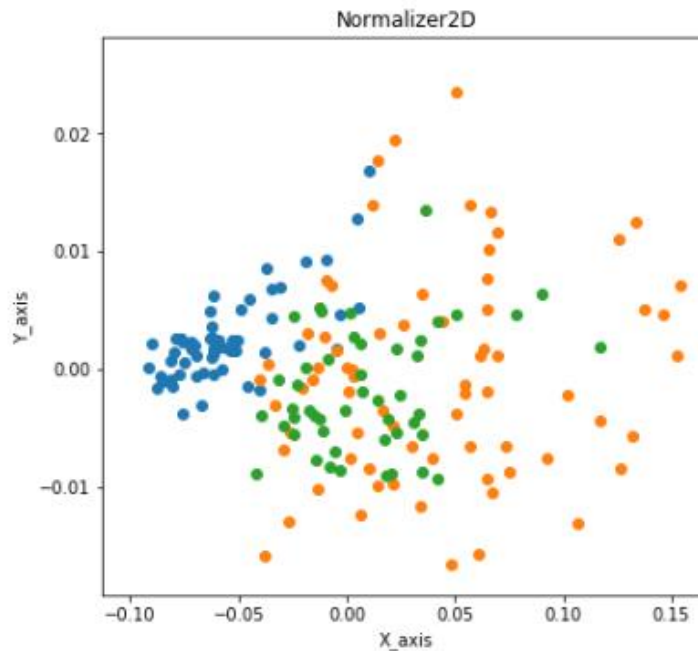
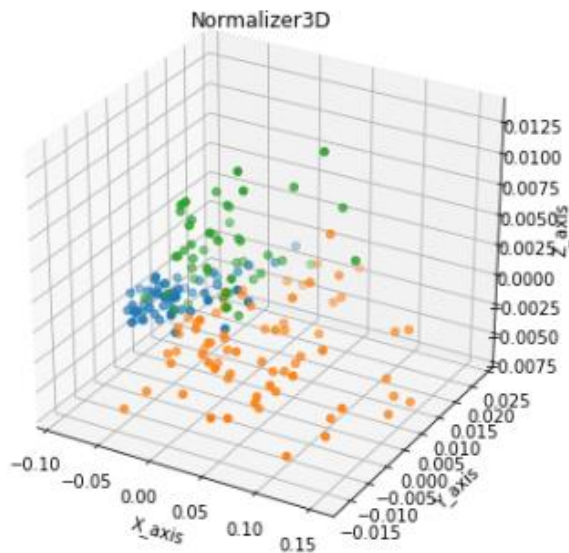
各次元の寄与率: [9.981e-01 1.736e-03 9.496e-05]
累積寄与率: 0.9999221050741546
加工なし



sklearnを使える

- SklearnのワインデータにNormalizer(特徴量のノルムを1にする)の前処理をしてPCA

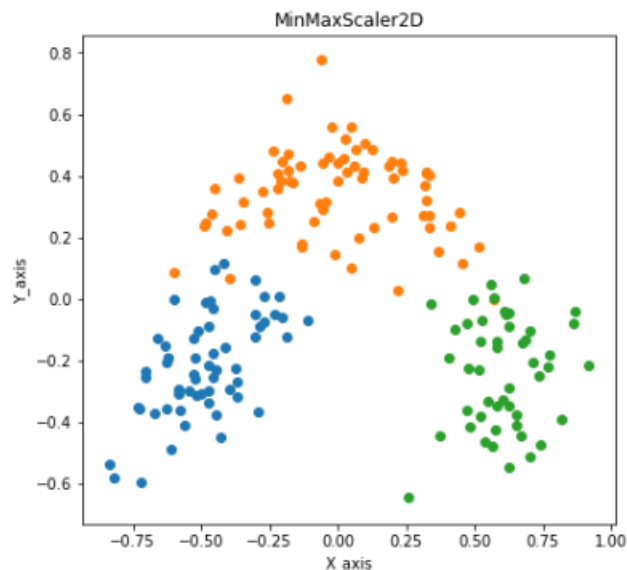
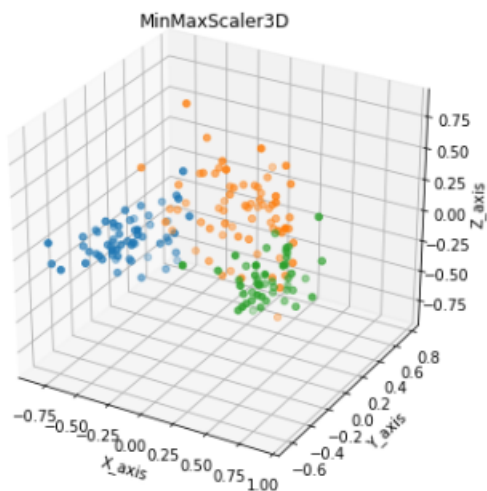
各次元の寄与率: [0.977 0.013 0.005]
累積寄与率: 0.9952310653624501



sklearnを使える

- SklearnのワンデータにMinMaxScaler(データが0~1になる)変換の前処理をしてPCA

各次元の寄与率: [0.407 0.19 0.086]
累積寄与率: 0.6828150695968223
正規化



まとめ

- 線形代数、モデル、前処理などのスキルセットを得るために色々やりました
- ベイズのスキルセットをもっと得たい

ディスカッション

- なぜMinMaxScalarだとうまくいったのか