機械学習（4）

概要

1. 機械学習の基礎
   - 概論
     - 教師あり学習
     - 教師なし学習
     - 強化学習
2. 教師あり学習
   - ランダムフォレスト
   - サポートベクターマシン

3. 教師なし学習と確率モデリング
   - クラスタリング
   - ナイーブベイズモデル
   - 混合ガウスモデル
   - クロスバリデーション・モデル選択

4. 各種データと機械学習
   - テーブルデータ
   - 行列データ
   - 時系列データ
   - グラフデータ

補助資料：http://small-island.work/

5. 深層学習
   ニューラルネットワークの基礎
   深層学習の基礎
   画像処理・自然言語処理におけるモデル

この講義の目的

目的

1. 実際のデータに対する処理をイメージできるようになる
2. 各データに対する基本的なアプローチを理解する
3. 基本的な手法と関連手法についての概要を知っておき、使用する際には詳細を調べられるようになる

データの種類

- 画像・動画
- 音声
- 自然言語

第5回以降

- テーブルデータ
- 行列データ

今回

- 時系列データ：
  タイムスタンプの付きデータ
  信号データ：波形・振動、脳波、その他生体信号など
- グラフデータ、関係データ

次回

テーブルデータ

表形式のデータ

通常これらのデータは人が見やすいように作られているので、前処理を適切に行う必要がある

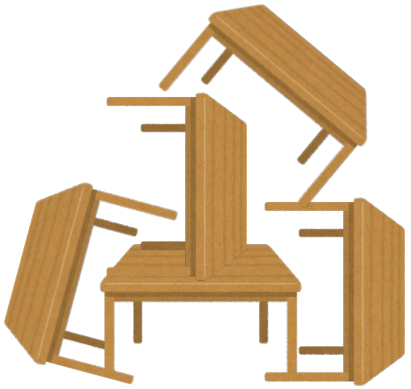
ＩＤ	氏名	年齢	身長	体重

ＩＤ	血糖値	HbA1C

ＩＤ	住所

ＩＤ	年齢	身長	体重	血糖値	HbA1C

データベースは通常複数のテーブルから構成される 関係データベースで表現されているため、それらのデータをテーブルデータとして分析する場合には適切にjoin処理などをして一つの表にまとめる前処理が必要になる



テーブルデータに対する機械学習

予測の手掛かりにしたいもの
説明変数
（特徴量）

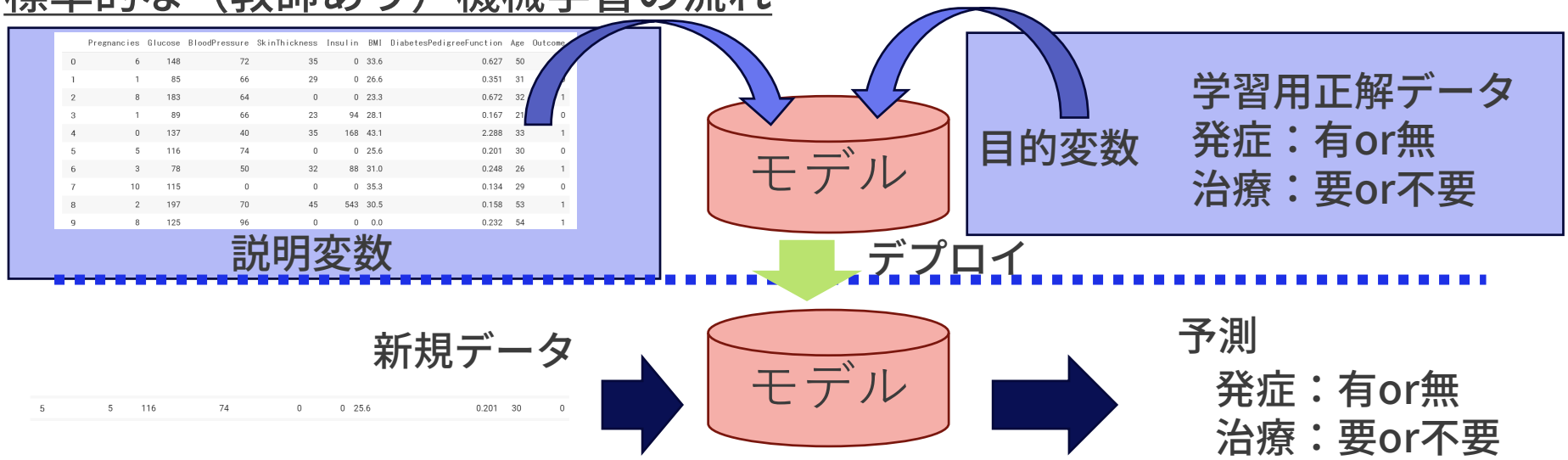
予測したいもの
目的変数

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

Pima Indiansの糖尿病発症予測データセット

標準的なテーブルデータに対する教師あり学習の流れ

標準的な（教師あり）機械学習の流れ



Automated Machine Learning：上記の流れを自動化してくれる仕組み

- Amazon SageMaker Autopilot
- Google Cloud: AutoML (Tables)
- IBM/AutoAI
- Microsoft/Automated ML
- Sony/Prediction One
- H2O Driverless AI
- Data robot
- PyCaret
- Auto Sk-learn

画像や自然言語でも
同様のサービスは多い

Automated Machine Learning

Pima Indiansの糖尿病発症予測データセットに対する結果 (by PyCaret)

評価尺度

機械学習手法たち

Model	Accuracy	AUC	Recall	Prec.	F1
Ridge Classifier	0.757900	0.000000	0.512900	0.719200	0.594600
Linear Discriminant Analysis	0.756100	0.800900	0.507600	0.716900	0.589900
Logistic Regression	0.743000	0.796000	0.508500	0.680600	0.578000
CatBoost Classifier	0.731900	0.795600	0.518700	0.655300	0.573100
Light Gradient Boosting Machine	0.726400	0.774500	0.557300	0.622800	0.584900
Random Forest Classifier	0.726300	0.745600	0.449100	0.662900	0.532900
Gradient Boosting Classifier	0.717000	0.778000	0.512900	0.620300	0.555500
Extra Trees Classifier	0.717000	0.773000	0.464900	0.633200	0.531000
Ada Boost Classifier	0.715300	0.774300	0.504100	0.610000	0.546200
Extreme Gradient Boosting	0.707700	0.772700	0.508200	0.598800	0.544600
K Neighbors Classifier	0.694800	0.721300	0.514000	0.574500	0.536400
Decision Tree Classifier	0.694800	0.669400	0.584500	0.560200	0.568700
Quadratic Discriminant Analysis	0.651800	0.000000	0.000000	0.000000	0.000000
Naive Bayes	0.647900	0.692600	0.089800	0.448100	0.144900
SVM - Linear Kernel	0.588500	0.000000	0.426300	0.442300	0.372500

各種評価値

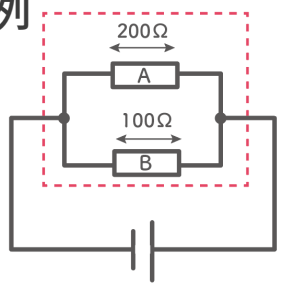
- 正解率 (精度, accuracy)：予測が実際にそうであったデータの割合
  (TP+TN)/(全データ数)
- 適合率 (precision)：正と予測したデータのうち，実際に正であった割合
  (TP)/(TP+FP)
- 再現率 (recall, 感度, sensitivity, 真陽性率)：
  実際に正のデータの内、正と予測した割合
  (TP)/(TP+FN)
- 特異度 (specificity)：実際に負のデータの内，負と予測した割合
  (TN)/(FP+TN)

	Positive のデータ	Negative のデータ
Positiveと予測	TP	FP
Negativeと予測	FN	TN

F値 (F-measure)：再現率と適合率の調和平均

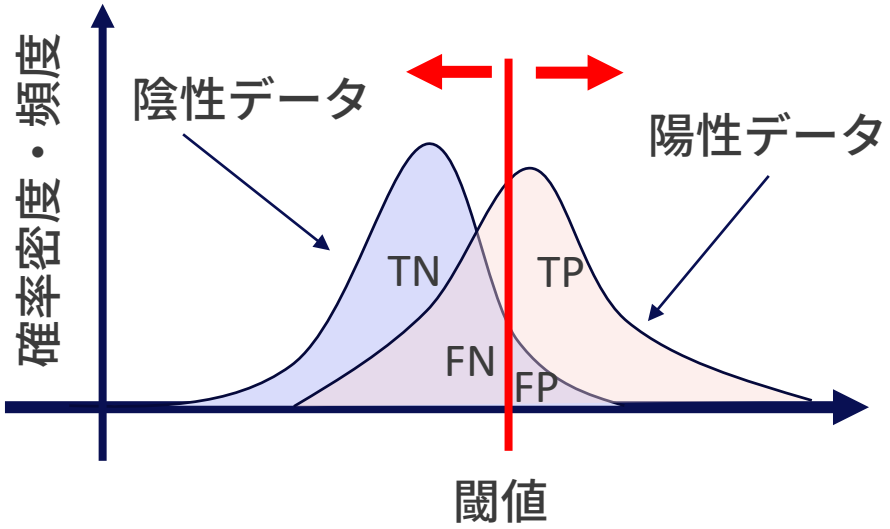
F=2/(1/recall + 1/precision)

調和平均の例



評価値: ROC, AUC (1/2)

閾値が連続的に変化する場合の評価



ROC (Receiver Operating Characteristic)曲線：

縦軸：sensitivity＝真陽性率（=TPR）

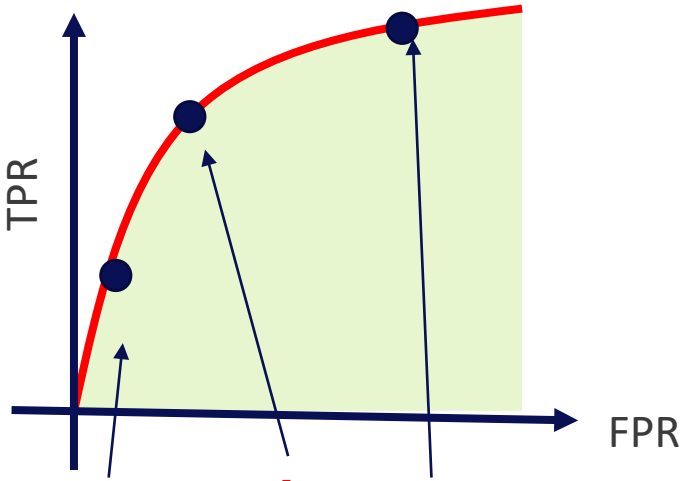
横軸：１－特異度＝偽陽性率（=FPR）

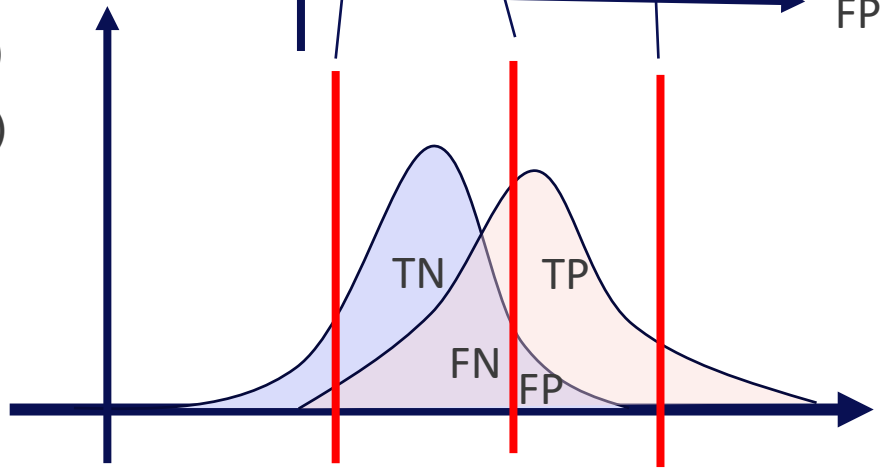
AUC, ROC-AUC (Area under curve)：

ROC曲線 の下面積

・最良の場合は1

・ランダムな場合0.5





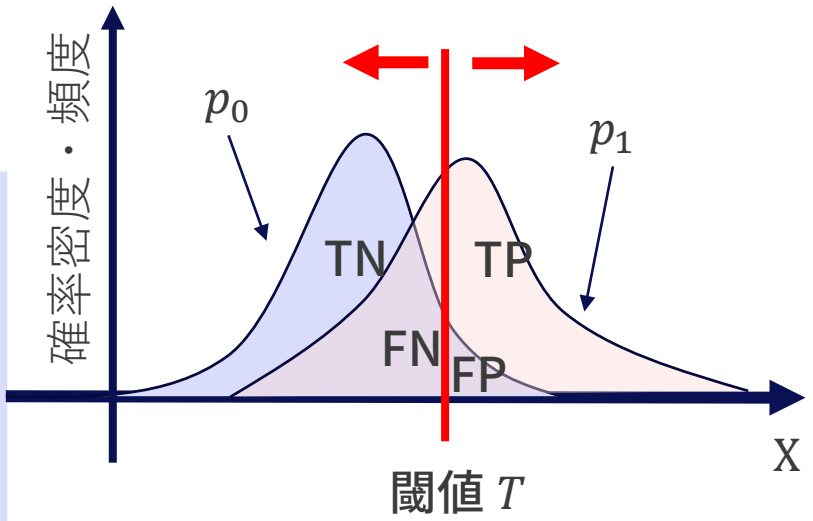
評価値: ROC, AUC (2/2)

$TPR(T)$:閾値Tの時のTP rate

$$TPR(T) = \mathrm{P}_1(\mathrm{X} > \mathrm{T}) = \int_T^{\infty} p_1(x)\, dx$$

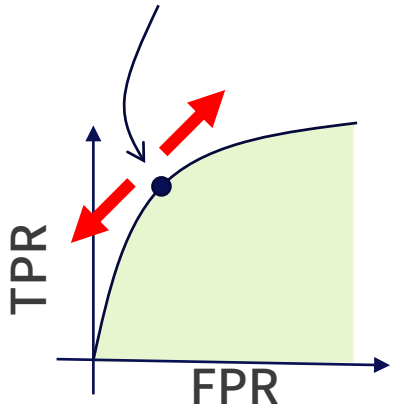
$FPR(T)$:閾値Tの時のFP rate

$$FPR(T) = \mathrm{P}_0(\mathrm{X} > \mathrm{T}) = \int_T^{\infty} p_0(x)\, dx$$



$$AUC = \int_0^1 TPR(T_{FPR})\, d\, FPR = \int_{-\infty}^{\infty} TPR(T) \frac{\partial FPR(T)}{\partial T}\, dT$$

$$FPR = FPR(T_{FPR})$$
$$d\, FPR = \frac{\partial FPR(T_{FPR})}{\partial T_{FPR}} dT_{FPR}$$



$I(condition)$: 指示関数
$condition = \mathrm{True} \rightarrow 1$
$condition = \mathrm{False} \rightarrow 0$

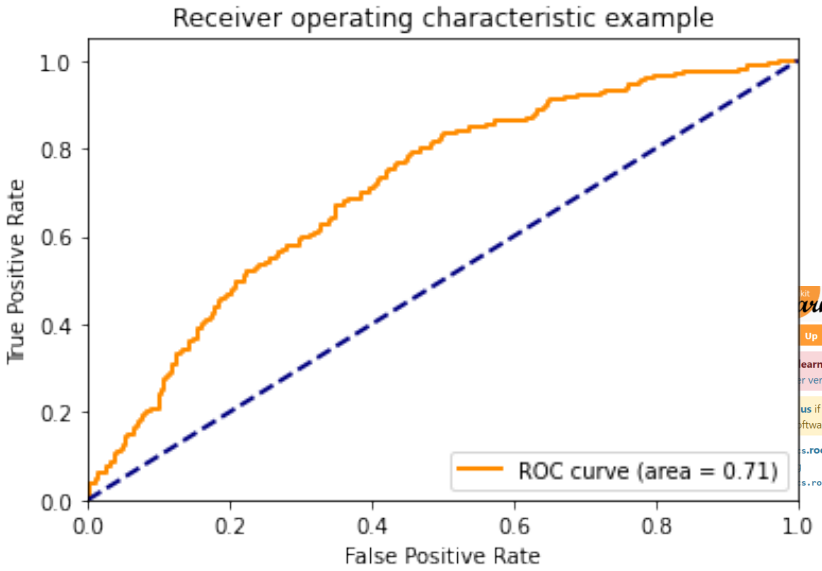
$$\begin{aligned} AUC &= \int_{-\infty}^{\infty} \int_T^{\infty} p_1(x_1)\, dx_1\, p_0(T)\ dT \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(x_1 > T) p_1(x_1)\, dx_1\, p_0(T)\ dT \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} I(x_1 > x_0) p_1(x_1)\ p_0(x_0)\, dx_1 dx_0 \\ &= P(X_1 > X_0) \end{aligned}$$

置き換え：$T = x_0$

← 陽性のスコアが陰性のスコアより高くなる確率

→ サンプル数の差の影響を受けにくい指標

例: Pima Indiansの糖尿病発症予測データセットのＲＯＣ曲線



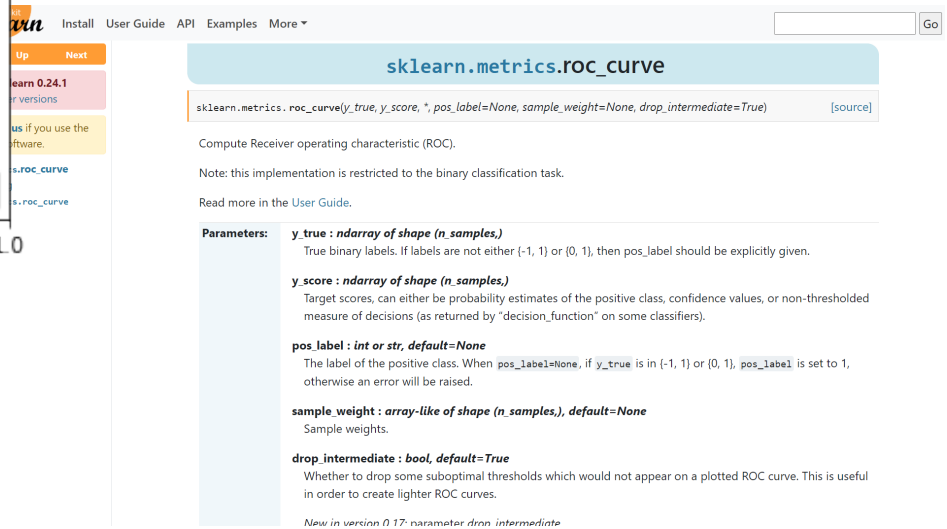
論文などでもよくある図なので

探してみるのもいいと思います

例：

DeepMind:急性腎障害を48時間前に予測

Tomašev, Nenad, et al. "A clinically applicable approach to continuous prediction of future acute kidney injury." Nature 572.7767 (2019): 116-119.



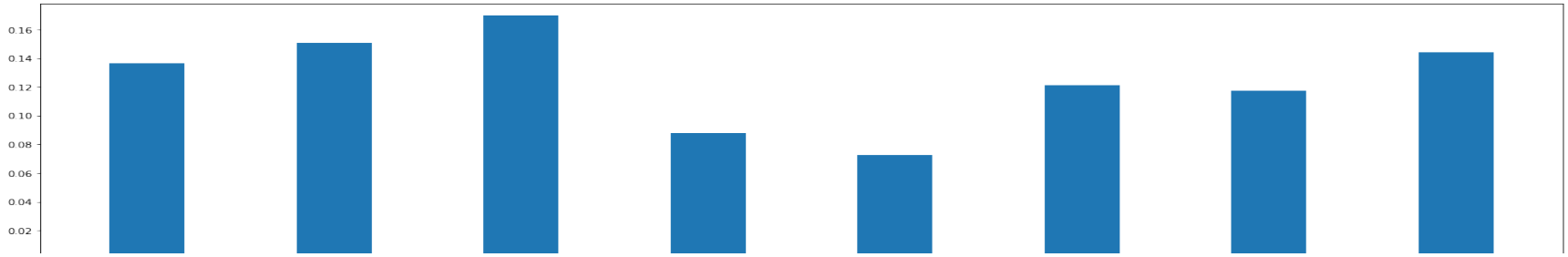
その他にも数多くの評価指標が利用できる

https://scikit-learn.org/stable/modules/classes.html#module-sklearn.metrics

応用上重要な機能：予測貢献度の表示

応用上では、単純な予測だけでなく、どの変数予測に貢献しているかを表示した

目的変数の予測に対する説明変数の貢献度



	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0

個別のサンプル（患者）ごとの貢献度 (by SHAP)

応用上重要な機能：予測貢献度の表示

予測貢献度の可視化手法たち

（XAI: Explainable AI という言葉もある）

- Partial Dependence Plots [Friedman+ 2001]
- M-Plots & ALE [Apley+ 2016]
- Individual Conditional Expectation [Goldstein+ 2017]
- Permutation feature importance [Altmann+ 2010]
- Global surrogate
- Local surrogate: LIME [Ribeiro+ 2016]
- SHAP [Lundberg+ 2017]
- Gradient-based methods
- Attention
- Graph-base: GNNExplainer [Ying+ 2019]

良い可視化手法という評価が困難ではあるが、
可視化の粒度や元のモデルに応じて使い分けることが多い

前処理：正規化

分析・予測の前に正規化の前処理をすることでうまくいくことが多い。

特に、表形式のデータは人が見やすいように単位が設定されており、単位がそろってないことも多いので、正規化で単位をそろえるとうまくいくことも多い

標準化

元データ

項目の平均

$$Z = \frac{X - \mu}{\sigma}$$

項目の標準偏差

平均０分散１になるように調整する

ID	計測値A (cm)	計測値B (mm)
1	6.0	150
2	1.0	240
3	8.0	330
	$\mu = 5.0$	$\mu = 240$
	$\sigma = 2.9$	$\sigma = 73$

標準化

ID	計測値A	計測値B
1	0.3	-1.2
2	-1.3	0.0
3	1.0	1.2

最大・最小

$$Z = \frac{X - \min(X)}{\max(X) - \min(X)}$$

最小０最大１になるように調整する

前処理：欠損値

データは何らかの理由で欠損することがある。欠損した場合にどういった表現がされるかはデータによって異なるので注意すること

ID	計測値A (cm)	計測値B(mm)
1	6.0	150
2	NAN	240
3	8.0	Error

欠損はその性質から、以下の３種類に分類されることが多い

- MCAR (Missing Completely At Random)
  完全にランダムに入る欠損：入力漏れやノイズによる欠損
- MAR (Missing at random)
  観測に応じて入る欠損：２０歳以下の人に関してはあまり測定されない項目などによる欠損
- MNAR (Missing Not At Random)
  欠損値自体の値によって欠損した欠損：体重100kg以上の人は計測器がエラーを返すなど

欠損値補完・欠損値除去

欠損除去：欠損したデータ自体を分析対象から外す
・欠損が少量の場合のみしか使えない
・依存関係のあるデータを削除すると偏りが出る可能性がある

代表値補間：平均値、中央値、最頻値などで補完する
どの代表値がいいかはデータによる

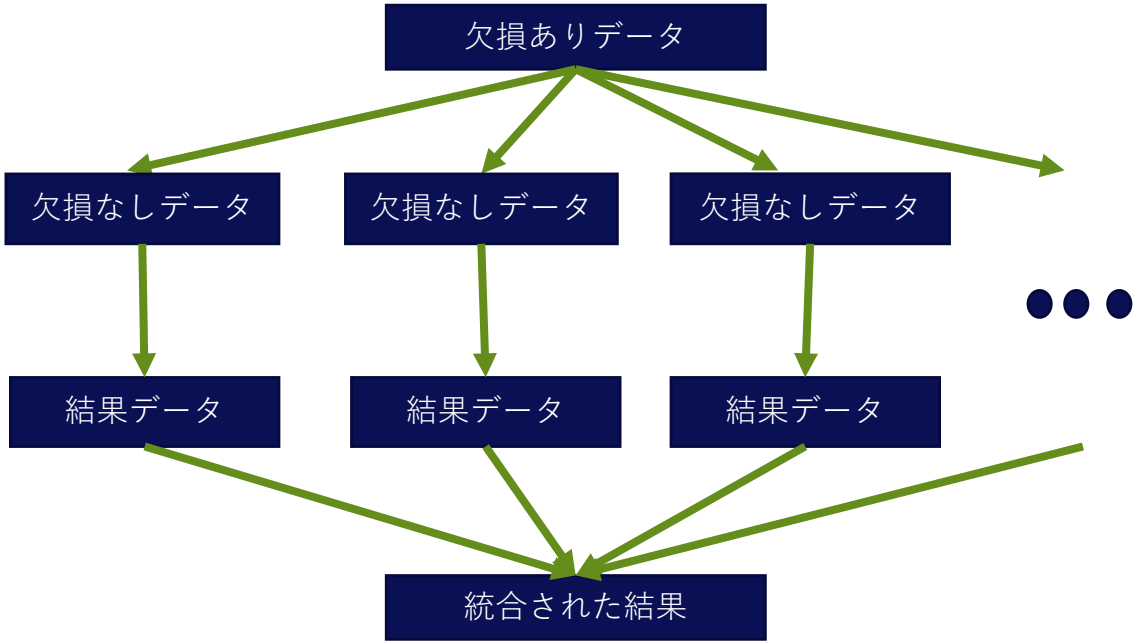
欠損値補間アルゴリズム

サンプリング（ランダムさ）を取り入れて補完する代表値補間と比べて特定の値に偏らないという利点がある
（次スライドで多重代入法について解説）

欠損値を許容する学習アルゴリズムを用いる
EMアルゴリズムやベイズアプローチは欠損値を隠れ変数とみなすことで自然に欠損値補間ができる
（第３回機械学習講義）

多重代入法

欠損部分をサンプリングして欠損を埋める
ただし、得られたサンプルによって偏ってしまうので、
複数回サンプリングを行い、それぞれで同じ分析をして最後に統合する



欠損補間のサンプリングは
事後分布$P(Z|X)$からサンプルすることに相当するため
例えばベイズ線形回帰を利用することも可（第３回機械学習講義）

X：観測値, Z：欠損値

多重代入法はscikit-learn にはないが、シンプルなプロセスなので実装は比較的容易

課題：別データへの適用

実際のデータ分析では既存のコードを参考にしつつ、手元のデータに適用していくことが多い、そのため、既存のコードを参考にして新たなデータに同じ処理を行う練習をする

「糖尿病データセットの予測モデル」を参考に
「乳がんデータセットの予測モデル」を作ってみる

参照プログラム
lecture_pred.ipynb

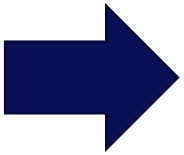
分かる人は自分でコードを読んで
初心者はこれから「糖尿病データセットの予測モデル」の方の説明をしますのでそれの後で取り組んでください

解答プログラム
lecture_pred_answer.ipynb

ヒント１：基本的には同じコードで動くはず
ヒント２：データが異なるので予測対象の列の名前などは異なるので注意

行列データ

ID	計測値A (cm)	計測値B (mm)
1	3.3	150
2	2.1	240
3	1.8	330



$$\begin{pmatrix} 3.3 & 150 \\ 2.1 & 240 \\ 1.8 & 330 \end{pmatrix}$$

テーブルデータを行列とみなすことで、線形代数の様々なツールが使える

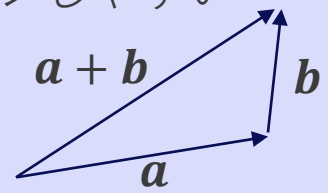
- 固有値分解(Eigenvalue Decomposition)
- 特異値分解(SVD)
- 行列分解(Matrix Factorization)
  - 非負値行列分解(Non-negative Matrix Factorization)
  - 二値行列分解(Binary Matrix Factorization)
- ロバストPCA

数学の言葉とデータの性質をうまく対応付けて考える必要がある
・利点として計算が速いことが多い
・欠損値などはあらかじめ補完しておくことが多い

固有値分解 (1/2)

モチベーション：行列による変換は人類には理解が難しいので、なんとかしてベクトルやスカラーの変換で表したい

ベクトルは
イメージしやすい

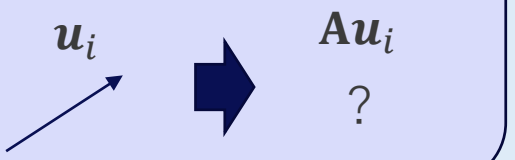


$n \times n$の行列Aの固有値分解

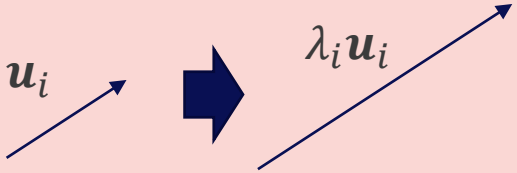
$$\mathbf{A}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$$

λ_i：固有値
$\boldsymbol{u}_i$: 固有ベクトル

行列による変換
（イメージできない）



ベクトルのスカラー倍
（矢印が伸びるイメージ）



行列による変換をスカラー倍として扱うことができるベクトルが固有ベクトル、その時のスカラー倍が固有値

固有値分解 (2/2)

固有値分解を行列で書くと

$$\mathbf{AP} = \mathbf{PD}$$

$$\mathbf{A} = \mathbf{PDP}^{-1}$$

行列Aによる変換を変換P、
軸に沿った拡大D、
Pの逆変換に分解する

$$\mathbf{A}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$$

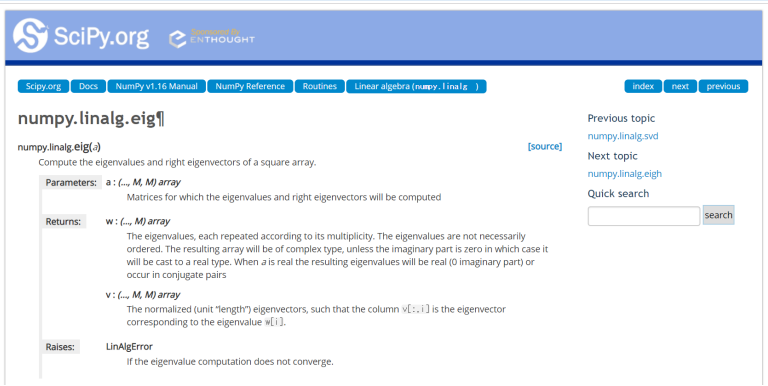
λ_i：固有値
$\boldsymbol{u}_i$：固有ベクトル

Pは一次独立な固有ベクトルを並べた行列

$$\mathbf{P} = (\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3, \dots, \boldsymbol{u}_n)$$

$$\mathbf{D} = \begin{pmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_n \end{pmatrix}$$

特に A が対称行列の場合は、
Pは直交行列となる。
つまり、 Pは回転・反転のような変換となる。



固有値分解とデータとの関係の例

分散共分散行列

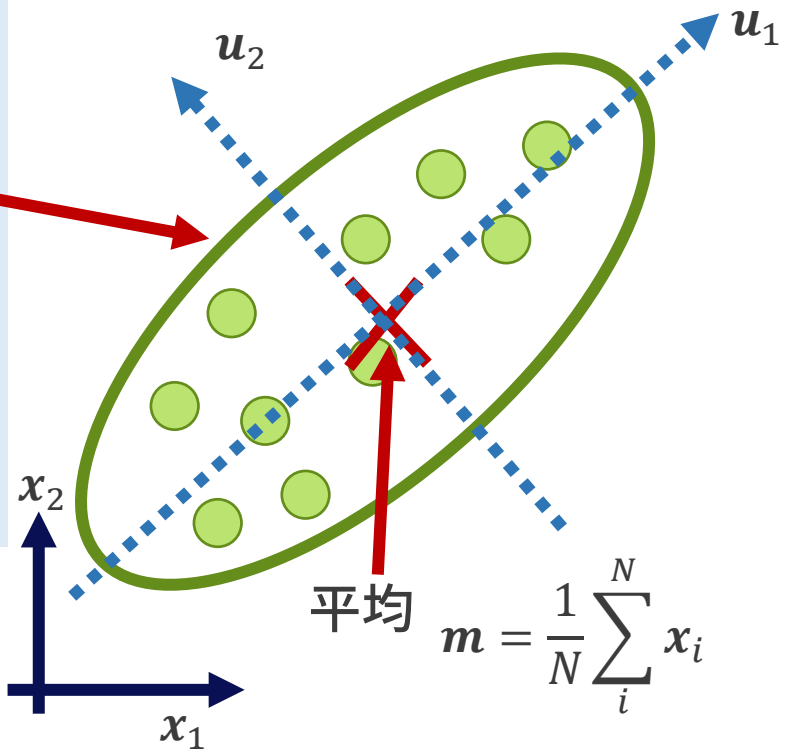
$$\mathbf{S} = \frac{1}{N}\sum_i^N (\boldsymbol{x}_i - \boldsymbol{m})(\boldsymbol{x}_i - \boldsymbol{m})^T$$

$\boldsymbol{x}_i$: データ点：
$\mathbf{X}$:$\boldsymbol{x}_i^T$を縦に並べた行列
N:データの数

Sを固有値分解

$$\mathbf{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$$

λ_i : 固有値
$\boldsymbol{u}_i$: 固有ベクトル

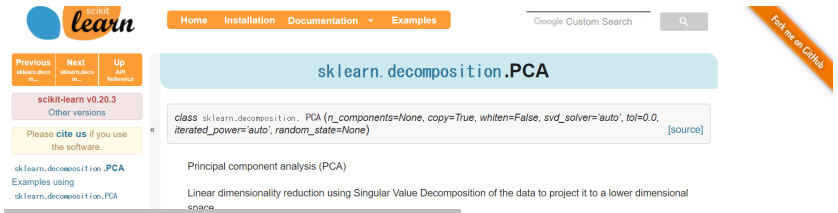


分散共分散行列
：データのばらつきに関してSはある方向がどう他の軸に影響を与えるか

固有ベクトル≒データのばらつきが、他の軸に影響を与えない方向
固有値　　≒その方向のばらつきの大きさ

データのばらつきが大きい軸からM個を選択して次元削減
→主成分分析

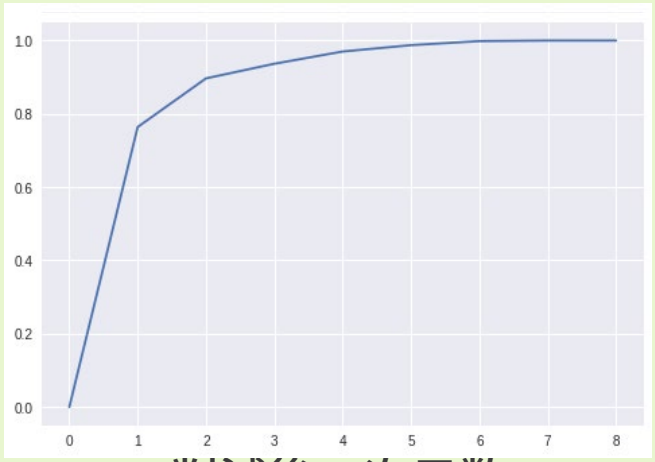
例：分散共分散行列を固有値分解≒主成分分析（ＰＣＡ）



参照プログラム
lecture_pca.ipynb

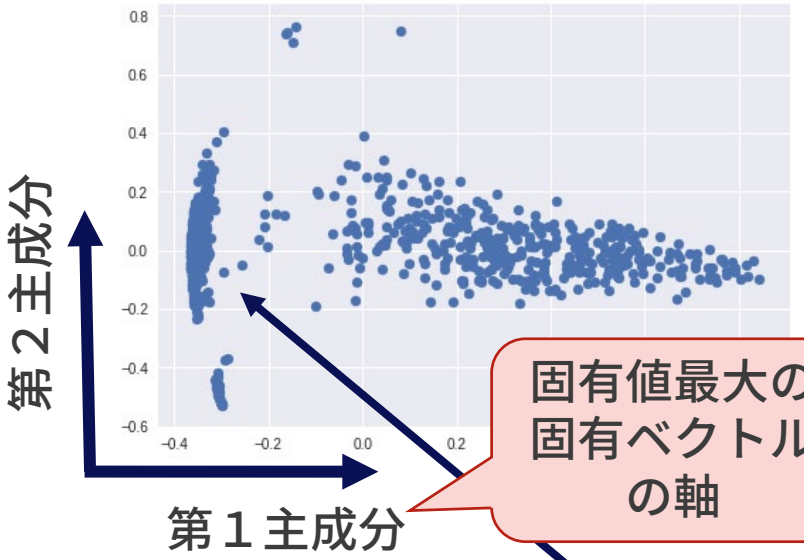
固有値

元のデータの
どれくらいを説明できるか



→２次元でもかなり元のデータの情報を持っていることがわかる

糖尿病データセットの説明変数のＰＣＡ



固有値最大の
固有ベクトル
の軸

主成分ベクトルの値

固有ベクトルの
要素の値
（固有ベクトルの向き・傾き）

元データの属性

Insulinの値がゼロ（欠損）になっていることによる分散が第１主成分と想像できる

特異値分解

モチベーション：

- 正方行列以外でも分解したい
- 常に直交行列（回転などの変換）で分解したい

$n \times m$の行列 $\mathbf{A}$ の特異値分解

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

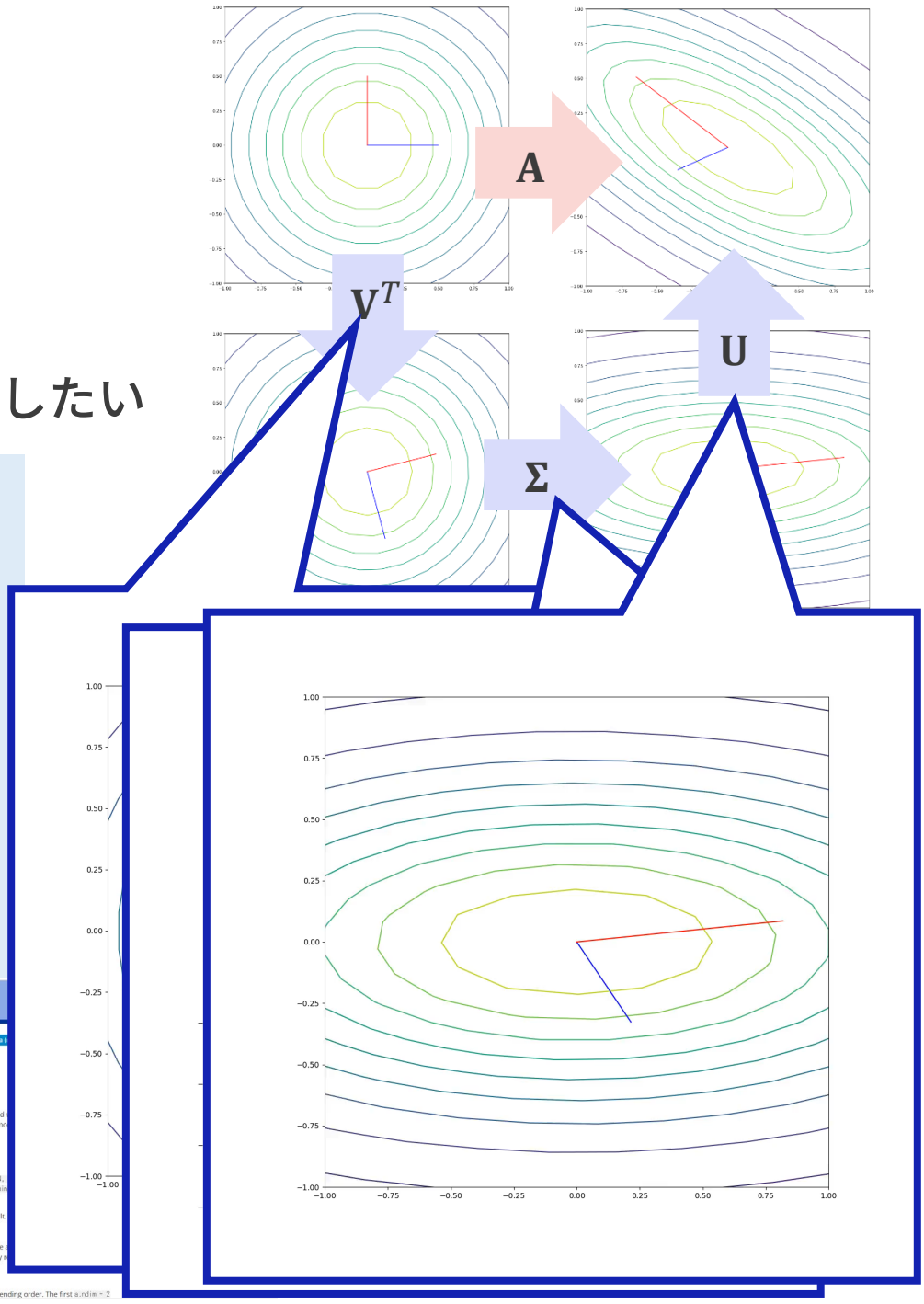
$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & 0 & 0 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & 0 & 0 & \dots & 0 \\ 0 & 0 & \ddots & 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & \sigma_n & 0 & \dots & 0 \end{pmatrix}$$

$n < m$の時

$\mathbf{U}$: 直交行列 $n \times n$

$\mathbf{V}$: 直交行列 $m \times m$

$\sigma_1, \sigma_2, \dots \sigma_n$: 特異値



分散共分散行列を固有値分解≒主成分分析≒データをSVD

分散共分散行列

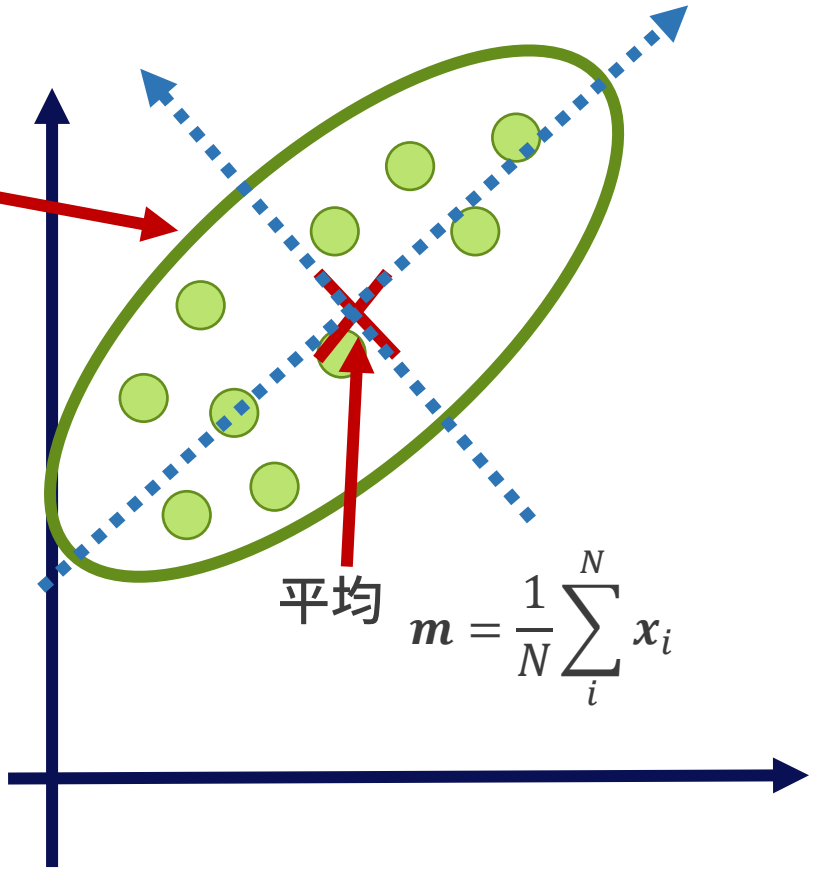
$$\mathbf{S} = \frac{1}{N}\sum_i^N (\boldsymbol{x}_i - \boldsymbol{m})(\boldsymbol{x}_i - \boldsymbol{m})^T$$

$\boldsymbol{x}_i$: データ点
$\mathbf{X}$: $\boldsymbol{x}_i^T$を縦に並べた行列
N: データの数

Sを固有値分解

$$\mathbf{S}\boldsymbol{u}_i = \lambda_i \boldsymbol{u}_i$$

λ_i: 固有値
$\boldsymbol{u}_i$: 固有ベクトル



分散共分散行列を固有値分解≒主成分分析

Sはある方向がどう他の軸に影響を与えるか
→ほかの方向に影響を与えない時は無相関
→固有ベクトルは右の点線方向
→主成分分析

分散共分散行列を固有値分解≒データXをSVD

$\boldsymbol{X}$（平均は０へ移動）の特異値分解とも等価（こちらで実装することが多い）

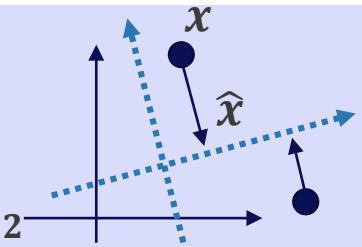
$$\mathbf{S}' = \frac{\mathbf{X}^T\mathbf{X}}{N} = \frac{\mathbf{V}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T\mathbf{V}^T}{N} = \mathbf{V}\frac{\boldsymbol{\Sigma}\boldsymbol{\Sigma}^T}{N}\mathbf{V}^{-1}$$

Xの特異値分解 $\mathbf{X} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$

$\mathbf{S}'$の固有値分解

分散最小（再構築の二乗誤差最小）と主成分分析

$$\boldsymbol{z} = \boldsymbol{U}^{\boldsymbol{T}}(\boldsymbol{x} - \boldsymbol{m})$$
$$\hat{\boldsymbol{x}} = \boldsymbol{U}\boldsymbol{z} + \boldsymbol{m}$$
$$\min \frac{1}{N}\sum_n (\hat{\boldsymbol{x}}_n - \boldsymbol{x}_n)^2$$



$\boldsymbol{U}: D \times M$

$\boldsymbol{U} = (\boldsymbol{u}_1, \boldsymbol{u}_2, .. \boldsymbol{u}_m)$

$\boldsymbol{u}_i^2 = 1$

$\boldsymbol{u}_i^T \boldsymbol{u}_j = 0$

$$\left(\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}} - \boldsymbol{I}\right)^2$$
$$= \left(\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\right)^2 - 2\boldsymbol{I}\left(\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\right) + \boldsymbol{I}$$
$$= \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}} - 2\left(\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\right) + \boldsymbol{I}$$
$$= \left(\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\right)$$

$$\frac{1}{N}\sum_n (\hat{\boldsymbol{x}}_n - \boldsymbol{x}_n)^2$$
$$= \frac{1}{N}\left(\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}(\boldsymbol{x}_n - \boldsymbol{m}) - (\boldsymbol{x}_n - \boldsymbol{m})\right)^2$$
$$= \frac{1}{N}\sum_n (\boldsymbol{x}_n - \boldsymbol{m})^T \left(\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}} - \boldsymbol{I}\right)^2 (\boldsymbol{x}_n - \boldsymbol{m})$$
$$= \frac{1}{N}\sum_n (\boldsymbol{x}_n - \boldsymbol{m})^T \left(\boldsymbol{I} - \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}\right)(\boldsymbol{x}_n - \boldsymbol{m})$$

$$= \frac{1}{N} tr\left(\sum_n -(\boldsymbol{x}_n - \boldsymbol{m})\boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}(\boldsymbol{x}_n - \boldsymbol{m})^T\right) + const$$
$$= -tr\left(\boldsymbol{U}^{\boldsymbol{T}}\boldsymbol{S}\boldsymbol{U}\right) + const$$
$$= -\boldsymbol{u}_1^T \boldsymbol{S}\boldsymbol{u}_1 - \boldsymbol{u}_2^T \boldsymbol{S}\boldsymbol{u}_2 - \cdots - \boldsymbol{u}_M^T \boldsymbol{S}\boldsymbol{u}_M + const$$

$(\hat{\boldsymbol{x}} - \boldsymbol{m}) = \boldsymbol{U}\boldsymbol{U}^{\boldsymbol{T}}(\boldsymbol{x} - \boldsymbol{m})$

ラグランジュの未定乗数法

$\boldsymbol{u}_i^2 = 1$
$\boldsymbol{u}_i^T \boldsymbol{u}_j = 0$

$$L_i(\boldsymbol{u}_i, \lambda) = -\boldsymbol{u}_i^T \boldsymbol{S}\boldsymbol{u}_i + \boldsymbol{\lambda}(\boldsymbol{u}_i^2 - \boldsymbol{1})$$

$$\frac{\partial L_i(\boldsymbol{u}_i, \lambda)}{\partial \boldsymbol{u}_i} = 2\boldsymbol{S}\boldsymbol{u}_i - 2\lambda\boldsymbol{u}_i = 0$$

$$\boldsymbol{S}\boldsymbol{u}_i = \lambda\boldsymbol{u}_i$$

最大固有値からM個とったものが再構築の二乗誤差を最小にする

再構築した二乗を最小化
≒分散共分散行列を固有値分解
（主成分分析）

PCAの確率モデルを考える:　最尤法(1/4)

・圧縮する次元数をモデル選択によって決定できる
・欠損値に対する補間を同時に行なうことができる
・他の確率モデルと同様に扱える（他のモデルとの結合などが容易）

確率的PCA

$$p(\boldsymbol{z}) = N(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$$

$$p(\boldsymbol{x}|\boldsymbol{z}) = N(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$$

$$\boldsymbol{W}: D \times M$$

先にM次元空間の$\boldsymbol{z}$の分布を考え、
データ$\boldsymbol{x}$(D次元)は$\boldsymbol{z}$の線形変換で生成される

ガウス分布の性質から

$$p(\boldsymbol{x}) = N(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

$$\boldsymbol{\Sigma} = \boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}$$

$$p(\boldsymbol{z}|\boldsymbol{x}) = N(\boldsymbol{z}|\boldsymbol{M}^{-1}\boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{\mu}), \sigma^2\boldsymbol{M}^{-1})$$

$$\boldsymbol{M} = \boldsymbol{W}^T\boldsymbol{W} + \sigma^2\boldsymbol{I}$$

$$\boldsymbol{\Sigma}^{-1} = \sigma^{-2}\boldsymbol{I} - \sigma^{-2}\boldsymbol{W}\boldsymbol{M}^{-1}\boldsymbol{W}^T$$

対数尤度：

$$\begin{aligned} L &= \textstyle\sum_n \log p(\boldsymbol{x}_n \mid \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2) \\ &= -\frac{ND}{2}\log 2\pi - \frac{N}{2}\log|\boldsymbol{\Sigma}| \\ &\quad -\frac{1}{2}\textstyle\sum_n (\boldsymbol{x}_n - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu}) \\ &= -\frac{ND}{2}\log 2\pi - \frac{N}{2}\log|\boldsymbol{\Sigma}| - \frac{N}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}) \end{aligned}$$

$$\mathbf{S} = \frac{1}{N}\textstyle\sum_n (\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^T$$

$\boldsymbol{\mu}$に関する最尤解

$$\frac{\partial L}{\partial \boldsymbol{\mu}} = \sum_n (\boldsymbol{x}_n - \boldsymbol{\mu}) = 0 \quad \Rightarrow \quad \boldsymbol{\mu}_{ML} = \frac{\sum_n \boldsymbol{x}_n}{N}$$

確率的PCA:　最尤法(2/4)

$$\boldsymbol{\Sigma} = \boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I}$$
$$\boldsymbol{W}: D \times M$$

対数尤度 ：$L = -\frac{ND}{2}\log 2\pi - \frac{N}{2}\log|\boldsymbol{\Sigma}| - \frac{N}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S})$

$$\frac{\partial \log|\boldsymbol{\Sigma}|}{\partial \boldsymbol{W}} = \left(\frac{\partial \log|\boldsymbol{\Sigma}|}{\partial \boldsymbol{\Sigma}}\right)_{ij} \frac{\partial \boldsymbol{\Sigma}_{ij}}{\partial \boldsymbol{W}} = \boldsymbol{\Sigma}^{-1}\mathbf{W}$$

$$\frac{\partial \log|\boldsymbol{\Sigma}|}{\partial \boldsymbol{\Sigma}} = \boldsymbol{\Sigma}^{-1}$$

$$\frac{\partial \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathrm{S})}{\partial \boldsymbol{W}} = \left(\frac{\partial \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathrm{S})}{\partial \boldsymbol{\Sigma}}\right)_{ij} \frac{\partial \boldsymbol{\Sigma}_{ij}}{\partial \boldsymbol{W}} = -\boldsymbol{\Sigma}^{-1}\mathrm{S}\boldsymbol{\Sigma}^{-1}\mathbf{W}$$

$$\frac{\partial \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathrm{S})}{\partial \boldsymbol{\Sigma}} = -\boldsymbol{\Sigma}^{-1}\mathrm{S}\boldsymbol{\Sigma}^{-1}$$

$$\frac{\partial L}{\partial \mathbf{W}} = -\frac{N}{2}(\boldsymbol{\Sigma}^{-1}\mathbf{W} - \boldsymbol{\Sigma}^{-1}\mathbf{S}\boldsymbol{\Sigma}^{-1}\mathbf{W}) = 0$$

$$\mathbf{W} = \mathbf{S}\boldsymbol{\Sigma}^{-1}\mathbf{W}$$

対称行列：$\boldsymbol{X}$に関して

$$\boldsymbol{X}_{ij}\frac{\partial \boldsymbol{\Sigma}_{ij}}{\partial \mathbf{W}_{kl}}$$
$$= \boldsymbol{X}_{ij}\frac{\partial \mathbf{W}_{ia}\mathbf{W}^T_{aj}}{\partial \mathbf{W}_{kl}}$$
$$= \boldsymbol{X}_{ij}\frac{\partial \mathbf{W}_{ia}}{\partial \mathbf{W}_{kl}}\mathbf{W}^T_{aj} + \boldsymbol{X}_{ij}\mathbf{W}_{ia}\frac{\partial \mathbf{W}^T_{aj}}{\partial \mathbf{W}_{kl}}$$
$$= \boldsymbol{X}_{kj}\mathbf{W}_{jl} + \boldsymbol{X}_{ik}\mathbf{W}_{il}$$
$$= 2\boldsymbol{X}_{kj}\mathbf{W}_{jl}$$

$$\frac{\partial \boldsymbol{A}^{-1}_{ij}\boldsymbol{B}_{ji}}{\partial \boldsymbol{A}_{kl}}$$
$$= \frac{\partial \boldsymbol{A}^{-1}_{ij}}{\partial \boldsymbol{A}_{kl}}\boldsymbol{B}_{ji}$$
$$= -\boldsymbol{A}^{-1}_{ia}\left(\frac{\partial \boldsymbol{A}}{\partial \boldsymbol{A}_{kl}}\right)_{ab}\boldsymbol{A}^{-1}_{bj}\boldsymbol{B}_{ji}$$
$$= -\boldsymbol{A}^{-1}_{ik}\boldsymbol{A}^{-1}_{li}\boldsymbol{B}_{ji}$$
$$= -\boldsymbol{A}^{-T}_{ki}\boldsymbol{B}^T_{ij}\boldsymbol{A}^{-T}_{jl}$$

確率的PCA:　最尤法(3/4)

Wに関する最尤法で満たすべき関係： $\mathbf{W} = \mathbf{S}\boldsymbol{\Sigma}^{-1}\mathbf{W}$

$\boldsymbol{\Lambda} = diag(\lambda_1, \dots, \lambda_M, 0, \dots, 0)$
$\boldsymbol{\Lambda}: D \times D$

$\mathbf{W} = \boldsymbol{U}\boldsymbol{\Lambda}^{1/2}\boldsymbol{V}^{\mathrm{T}}$
と特異値分解できる

$$\begin{aligned}\boldsymbol{\Sigma} &= \boldsymbol{W}\boldsymbol{W}^T + \sigma^2\boldsymbol{I} \\ &= \left(\boldsymbol{U}\boldsymbol{\Lambda}^{\boldsymbol{1/2}}\boldsymbol{V}^T\right)\left(\boldsymbol{V}\,\boldsymbol{\Lambda}^{1/2}\boldsymbol{U}^T\right) + \sigma^2\boldsymbol{I} \\ &= \boldsymbol{U}(\boldsymbol{\Lambda} + \sigma^2\boldsymbol{I})\boldsymbol{U}^T\end{aligned}$$

- $\boldsymbol{W}: D \times M$
- $\boldsymbol{U}: D \times D$
- $\boldsymbol{\Lambda}^{1/2}: D \times M$
- $\boldsymbol{V}: M \times M$

$(\boldsymbol{\Lambda} + \sigma^2\boldsymbol{I}) = diag\left((\lambda_1 + \sigma^2), \dots, (\lambda_M + \sigma^2), \sigma^2, \dots, \sigma^2\right)$

$\mathbf{W} = \mathbf{S}\boldsymbol{\Sigma}^{-1}\mathbf{W}$ に代入して

$$\mathbf{W} = \mathbf{S}\,\boldsymbol{U}(\boldsymbol{\Lambda} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{U}^T\mathbf{W}$$

$$\boldsymbol{U} = \mathbf{S}\,\boldsymbol{U}(\boldsymbol{\Lambda} + \sigma^2\boldsymbol{I})^{-1}$$

$$\boldsymbol{U}(\boldsymbol{\Lambda} + \sigma^2\boldsymbol{I}) = \mathbf{S}\,\boldsymbol{U}$$

Sに関するの固有値の式

尤度最大では、

$(\lambda_1 + \sigma^2), \dots, (\lambda_M + \sigma^2), \sigma^2, \dots, \sigma^2$　が（Sの固有値）$= (s_1, \dots, s_D)$と対応している

$\lambda_1 \dots \lambda_M$はWの特異値として導入したので、
$\mathbf{W}\mathbf{W}^{\mathrm{T}}$の固有値/固有ベクトルがこうなるようWを決定する

$$\mathbf{W}\mathbf{W}^{\mathrm{T}} = \boldsymbol{U}\boldsymbol{\Lambda}'\boldsymbol{U}^{\mathrm{T}}$$

$$\mathbf{W}\mathbf{W}^{\mathrm{T}} = \boldsymbol{U}(\boldsymbol{\Lambda}')^{1/2}\boldsymbol{R}\boldsymbol{R}^T(\boldsymbol{\Lambda}')^{1/2}\boldsymbol{U}^T$$

$$\mathbf{W}_{ML} = \boldsymbol{U}(\boldsymbol{\Lambda}')^{1/2}\boldsymbol{R}$$

$\boldsymbol{\Lambda}' = \mathrm{diag}(s_1 - \sigma^2, \dots, s_M - \sigma^2, 0, \dots, 0)$

ただし、$\boldsymbol{R}$は$\boldsymbol{R}\boldsymbol{R}^T = \mathbf{1}$、
を満たす任意の行列

確率的PCA:　最尤法(4/4)

σ^2の推定

対数尤度 ：$L = -\frac{ND}{2}\log 2\pi - \frac{N}{2}\log|\boldsymbol{\Sigma}| - \frac{N}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S})$

$$|\boldsymbol{\Sigma}| = (\lambda_1 + \sigma^2)\cdot\ldots\cdot(\lambda_M + \sigma^2)\cdot\sigma^2\cdot\ldots\cdot\sigma^2$$

$$\log|\boldsymbol{\Sigma}| = \sum_{m=1}^{M}\log(\lambda_m + \sigma^2) + (D-M)\log\sigma^2$$

$$\log|\boldsymbol{\Sigma}| = \sum_{m=1}^{M}\log s_m + (D-M)\log\sigma^2$$

$$\boldsymbol{\Sigma} = \boldsymbol{U}\boldsymbol{\Lambda}'\boldsymbol{U}^{\mathrm{T}} + \sigma^2\boldsymbol{I}$$

$$\boldsymbol{\Sigma}^{-1} = \boldsymbol{U}(\boldsymbol{\Lambda} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{U}^{T}$$

$$\boldsymbol{\Sigma}^{-1}\mathbf{S} = \boldsymbol{U}(\boldsymbol{\Lambda}' + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{U}^{\mathrm{T}}\mathbf{U}\boldsymbol{\Lambda}\boldsymbol{U}^{\mathrm{T}}$$

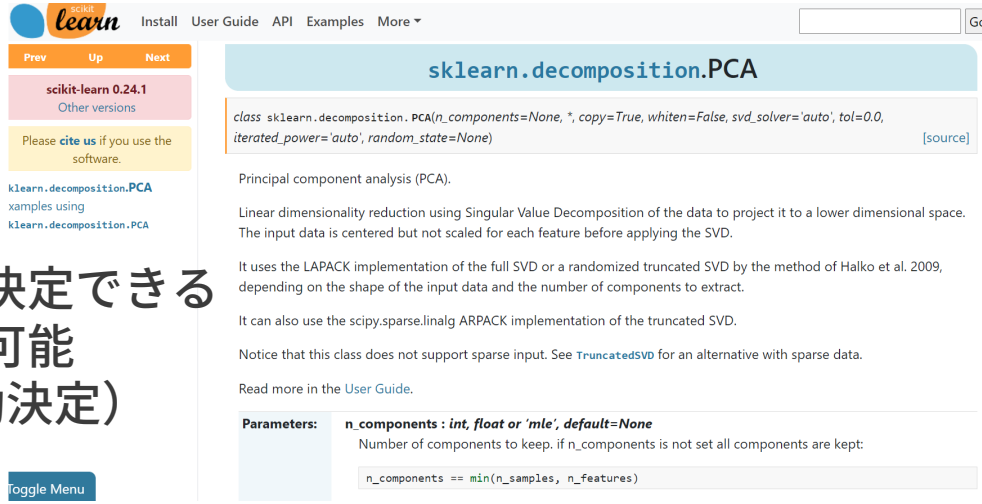
$$\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}) = M + \sum_{m=M+1}^{D} s_m/\sigma^2$$

$$L = -\frac{N}{2}\left(D\log 2\pi + \sum_{m=1}^{M}\log s_m + (D-M)\log\sigma^2 + M + \sum_{m=M+1}^{D} s_m/\sigma^2\right)$$

$$\frac{\partial L}{\partial \sigma^2} = -\frac{N}{2}\left(\frac{(D-M)}{\sigma^2} - \sum_{m=M}^{D} s_m(\sigma^2)^{-2}\right) = 0$$

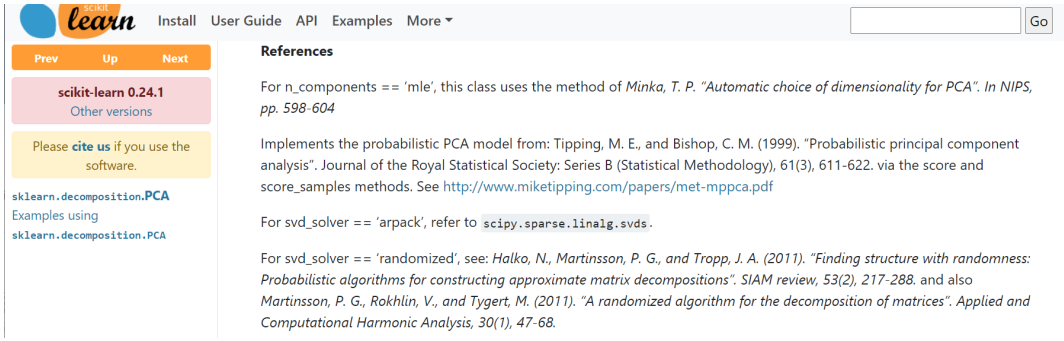
$$\sigma_{ML}^2 = \sum_{m=M+1}^{D} s_m/(D-M)$$

確率的PCAの利用法



確率的PCAの利点

- 圧縮する次元数をモデル選択によって決定できる
  ラプラス近似に基づくモデル選択が可能
  （n_components=“mle”とすると自動決定）



参照プログラム
lecture_prob_pca.ipynb

- 欠損値に対する補間を同時に行なうことができる
- 他の確率モデルと同様に扱える（他のモデルとの結合などが容易）

確率的PCA のEMアルゴリズム(1/2)

$$p(\boldsymbol{z}) = N(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$$
$$p(\boldsymbol{x}|\boldsymbol{z}) = N(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$$
$$\boldsymbol{W}\text{: } D \times M$$

$$p(\boldsymbol{z}|\boldsymbol{x}) = N(\boldsymbol{x}|\boldsymbol{M}^{-1}\boldsymbol{W}^T(\boldsymbol{x} - \boldsymbol{\mu}), \sigma^2\boldsymbol{M}^{-1})$$
$$\boldsymbol{M} = \boldsymbol{W}^T\boldsymbol{W} + \sigma^2\boldsymbol{I}$$

ＥＭアルゴリズムで考える尤度の下限（Ｅステップ）：

$$Q = E_{p(z|x;\boldsymbol{W}_{old},\sigma_{old})}[\log p(\boldsymbol{x}, \boldsymbol{z} \mid \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2)]$$
$$= E[\log p(\boldsymbol{z})] + E[\log p(\boldsymbol{x}|\boldsymbol{z})]$$

$$E[\boldsymbol{z}] = \boldsymbol{M}^{-1}\boldsymbol{W}_{old}{}^T(\boldsymbol{x} - \boldsymbol{\mu})$$
$$E[\boldsymbol{z}\boldsymbol{z}^T] = \sigma_{old}^2\boldsymbol{M}^{-1} + E[\boldsymbol{z}]E[\boldsymbol{z}^T]$$
$$\boldsymbol{M} = \boldsymbol{W}_{old}{}^T\boldsymbol{W}_{old} + \sigma_{old}^2\boldsymbol{I}$$

- $E[\log p(\boldsymbol{z})] = -\frac{M}{2}\log 2\pi - \frac{1}{2}\mathrm{tr}(E[\boldsymbol{z}\boldsymbol{z}^T])$
- $E[\log p(\boldsymbol{x}|\boldsymbol{z})]$

$$= -\frac{D}{2}\log 2\pi\sigma^2 - \frac{1}{2}\mathrm{tr}\left(\sigma^{-2}(\boldsymbol{x} - \boldsymbol{W}\boldsymbol{z} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{W}\boldsymbol{z} - \boldsymbol{\mu})^{\mathrm{T}}\right)$$
$$= -\frac{D}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\left(\mathrm{tr}\left(E[\boldsymbol{z}\boldsymbol{z}^T]\boldsymbol{W}^T\boldsymbol{W}\right) - 2\mathrm{tr}\left(\boldsymbol{W}E[\boldsymbol{z}](\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}}\right) + (\boldsymbol{x} - \boldsymbol{\mu})^{\mathrm{T}}(\boldsymbol{x} - \boldsymbol{\mu})\right)$$

$$E[\boldsymbol{z}\boldsymbol{z}^T] - E[\boldsymbol{z}]E[\boldsymbol{z}^T] = E[(\boldsymbol{z} - E[\boldsymbol{z}])(\boldsymbol{z} - E[\boldsymbol{z}])^T]$$

確率的PCA のEMアルゴリズム（2/2）

Qの最大化（Mステップ）：

Update: $\boldsymbol{W}$

$$\frac{\partial Q}{\partial \boldsymbol{W}} = -\frac{1}{2\sigma^2}\left(2\boldsymbol{W}E[\boldsymbol{z}\boldsymbol{z}^T] - 2(\boldsymbol{x}-\boldsymbol{\mu})E[\boldsymbol{z}]^{\mathrm{T}}\right) = 0$$

$$\boldsymbol{W} = (\boldsymbol{x}-\boldsymbol{\mu})E[\boldsymbol{z}]^{\mathrm{T}} \cdot E[\boldsymbol{z}\boldsymbol{z}^T]^{-1}$$

$$\boldsymbol{W}E[\boldsymbol{z}\boldsymbol{z}^T] - (\boldsymbol{x}-\boldsymbol{\mu})E[\boldsymbol{z}]^{\mathrm{T}} = 0$$
$$\boldsymbol{W}E[\boldsymbol{z}\boldsymbol{z}^T] = (\boldsymbol{x}-\boldsymbol{\mu})E[\boldsymbol{z}]^{\mathrm{T}}$$

Update: σ^2

$$\frac{\partial Q}{\partial \sigma^2} = -\frac{D}{2\sigma^2} + \frac{1}{2\sigma^4}\left(\mathrm{tr}\left(E[\boldsymbol{z}\boldsymbol{z}^T]\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W}\right) - 2\mathrm{tr}\left(\boldsymbol{W}E[\boldsymbol{z}](\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\right) + (\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{\mu})\right) = 0$$

$$\sigma^2 = \frac{1}{2D}\left(\mathrm{tr}\left(E[\boldsymbol{z}\boldsymbol{z}^T]\boldsymbol{W}^{\boldsymbol{T}}\boldsymbol{W}\right) - 2\mathrm{tr}\left(\boldsymbol{W}E[\boldsymbol{z}](\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}\right) + (\boldsymbol{x}-\boldsymbol{\mu})^{\mathrm{T}}(\boldsymbol{x}-\boldsymbol{\mu})\right)$$

一見複雑に見えるが、$D \times D$の行列は出てこない

確率的PCA のEMアルゴリズムのpython実装と実行結果

EMアルゴリズムの基本的な流れ

E-step

各サンプルごとに

$E[\boldsymbol{z}]$

$E[\boldsymbol{z}\boldsymbol{z}^T]$

の値を計算する

M-step

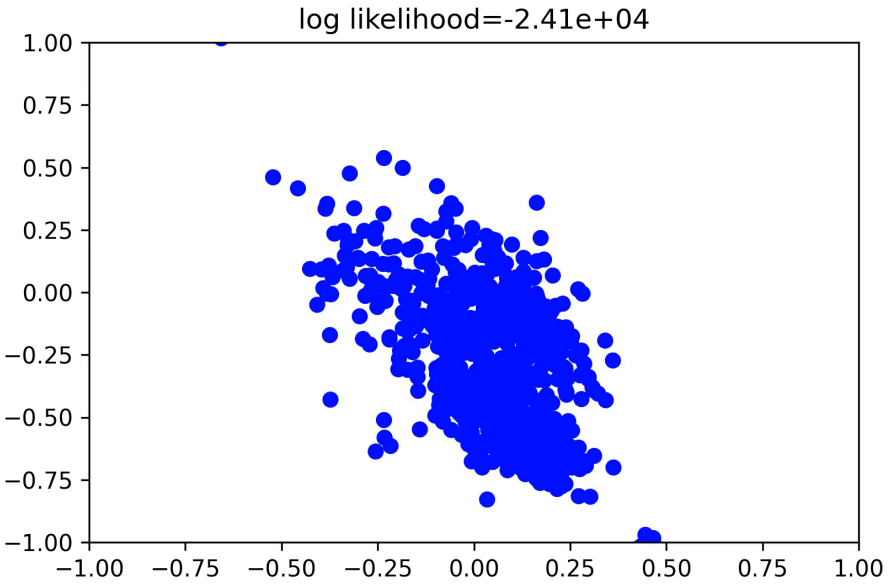
Update: $\boldsymbol{W}$

Update: σ^2

修正

直交化： $\boldsymbol{W}$

糖尿病データセットの説明変数の
確率的ＰＣＡのEM学習



参照プログラム
lecture_prob_pca_em.ipynb

ＰＣＡまとめ

主成分分析

≒分散共分散行列を固有値分解

≒データをSVD

≒再構築した二乗を最小化

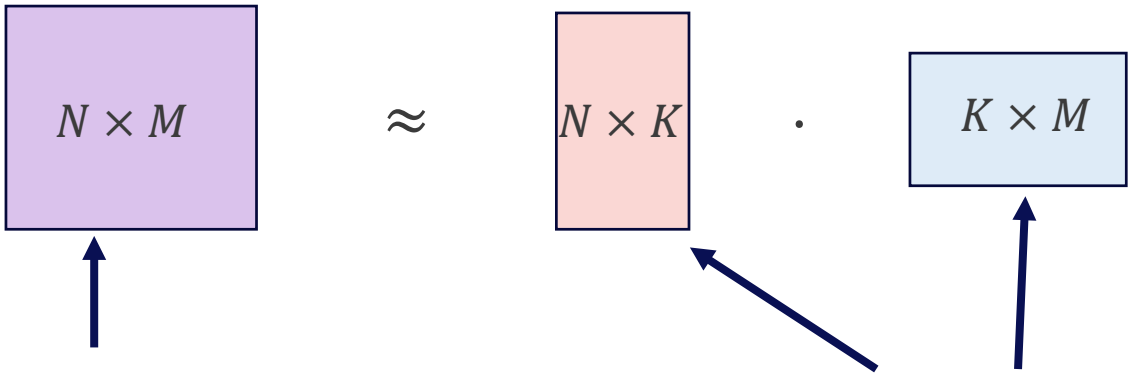
≒ガウス分布に従う潜在変数からの
線形変換によるデータ生成モデルの最尤推定

$$p(\boldsymbol{z}) = N(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$$
$$p(\boldsymbol{x}|\boldsymbol{z}) = N(\boldsymbol{x}|\boldsymbol{W}\boldsymbol{z} + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})$$

事前分布をおいて、ベイズ推定も可能

Non-negative Matrix Factorization

ある非負の値を要素に持つ行列を二つの非負の行列の積に分解する



(Kは指定する)

非負値のデータ
- 遺伝子発現
- 音声スペクトルのパワー
- 画像

データのパターンのようなものに分解できる可能性がある

データによって（NとMの次元の意味によって）分解された行列が持つ意味は異なる

濃度や何かをカウントしたデータやパワーの値など0以上の値しかとらないデータの分析によく利用される

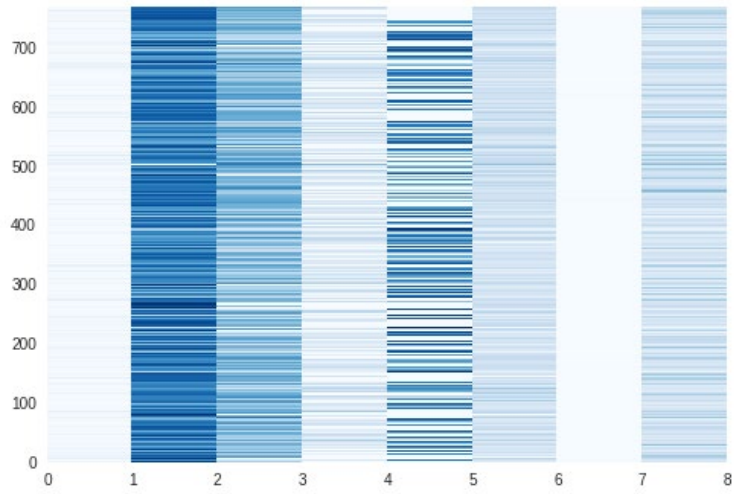
Non-negative Matrix Factorization

$$N \times M \approx N \times K \cdot K \times M$$

参照プログラム
lecture_nmf.ipynb

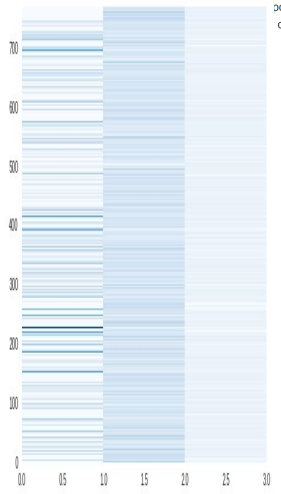
糖尿病データセットのNMF

サンプル番号



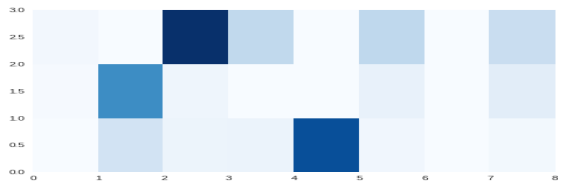
糖尿病データセットの説明変数

$\approx$



基底の数 $(K = 3)$

$\cdot$

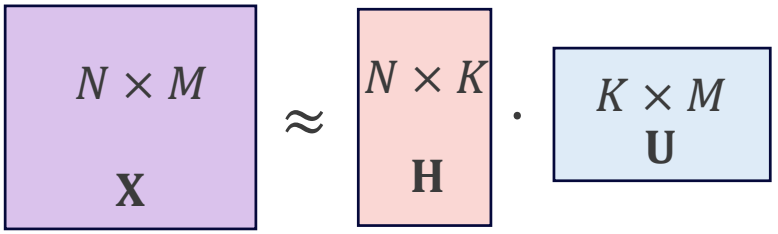


糖尿病データセットの説明変数

経験的に**スパース（大きい値を持っている要素は一部のみ）**になりやすい

NMFのアルゴリズム(1/3)

XとHUの値が近くなるように学習する



Iダイバージェンス（一般化KLダイバージェンス）

$$d(\mathbf{X}, \mathbf{HU}) = \sum_{ij} \mathbf{X}_{ij} \log \frac{\mathbf{X}_{ij}}{(\mathbf{HU})_{ij}} - \mathbf{X}_{ij} + (\mathbf{HU})_{ij}$$

$$s.t.\ \mathbf{H}_{ij} \geq 0$$
$$\mathbf{U}_{ij} \geq 0$$

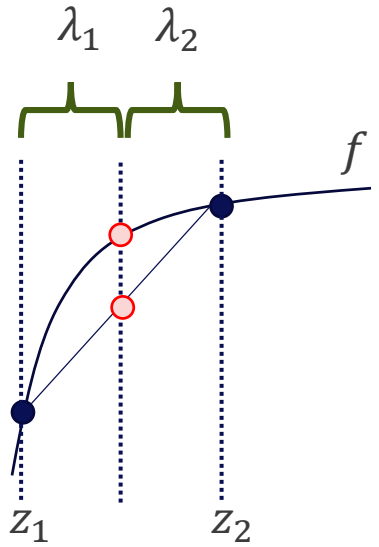
一般化KLダイバージェンス

$$d(f(x), g(x)) = \int f(x) \log \frac{f(x)}{g(x)} - f(x) + g(x)\, dx$$

$$\begin{aligned} Loss &= \textstyle\sum_{ij} -\mathbf{X}_{ij} \log (\mathbf{HU})_{ij} + (\mathbf{HU})_{ij} \\ &= \textstyle\sum_{ij} \left(-\mathbf{X}_{ij} \log \sum_{\boldsymbol{k}} \mathbf{H}_{ik} \mathbf{U}_{kj} + \sum_{\boldsymbol{k}} \mathbf{H}_{ik} \mathbf{U}_{kj} \right) \\ &\leq \textstyle\sum_{ij} \left(-\mathbf{X}_{ij} \sum_k \lambda_{ijk} \log \frac{\mathbf{H}_{ik} \mathbf{U}_{kj}}{\lambda_{ijk}} + \sum_{\boldsymbol{k}} \mathbf{H}_{ik} \mathbf{U}_{kj} \right) = Q \end{aligned}$$

<u>イェンセンの不等式</u>　fが上に凸、ここではlog

$$f\left(\sum_k z_k\right) \geq \sum_k \lambda_k f\left(\frac{z_k}{\lambda_k}\right) \qquad s.t. \sum_k \lambda_k = 1$$



NMFのアルゴリズム(2/3)

Qの最小化（補助関数法）

$$Q = \sum_{ij}\left(-\mathbf{X}_{ij}\sum_{k}\lambda_{ijk}\log\frac{\mathbf{H}_{ik}\mathbf{U}_{kj}}{\lambda_{ijk}} + \sum_{k}\mathbf{H}_{ik}\mathbf{U}_{kj}\right) \qquad s.t.\sum_{k}\lambda_{ijk} = 1$$

ラグランジュの未定乗数法：$L_Q = Q + \mathrm{L}_{\mathrm{ij}}\left(\sum_k \lambda_{ijk} - 1\right)$

Update:λ_{ijk}

$$\frac{\partial L_Q}{\partial \lambda_{ijk}} = \frac{\partial}{\partial \lambda_{ijk}}\left(-\mathbf{X}_{ij}\lambda_{ijk}\log \mathbf{H}_{ik}\mathbf{U}_{kj} + \mathbf{X}_{ij}\lambda_{ijk}\log \lambda_{ijk} + \mathrm{L}_{\mathrm{ij}}\right) = 0$$

$$-\mathbf{X}_{ij}\log \mathbf{H}_{ik}\mathbf{U}_{kj} + \mathbf{X}_{ij}\left(\log \lambda_{ijk} + 1\right) + \mathrm{L}_{ij} = 0$$

整理

$$\mathbf{X}_{ij}\log \lambda_{ijk} + \mathbf{X}_{ij} + \mathrm{L}_{ij} = \mathbf{X}_{ij}\log \mathbf{H}_{ik}\mathbf{U}_{kj}$$

$$\log \lambda_{ijk} = \log \mathbf{H}_{ik}\mathbf{U}_{kj} - 1 - \frac{\mathrm{L}_{ij}}{\mathbf{X}_{ij}}$$

$$\lambda_{ijk} = \frac{\mathbf{H}_{ik}\mathbf{U}_{kj}}{e+\exp\left(\mathrm{L}_{ij}/\mathbf{X}_{ij}\right)}$$

$$\lambda_{ijk} = \frac{\mathbf{H}_{ik}\mathbf{U}_{kj}}{\sum_k \mathbf{H}_{ik}\mathbf{U}_{kj}}$$

$$\sum_k \lambda_{ijk} = 1$$

$$\frac{\sum_k \mathbf{H}_{ik}\mathbf{U}_{kj}}{e+\exp\left(\mathrm{L}_{ij}/\mathbf{X}_{ij}\right)} = 1$$

$$\sum_k \mathbf{H}_{ik}\mathbf{U}_{kj} = e + \exp\left(\mathrm{L}_{ij}/\mathbf{X}_{ij}\right)$$

NMFのアルゴリズム(3/3)

$$Q = \sum_{ij} \left(-\mathbf{X}_{ij} \sum_{k} \lambda_{ijk} \log \frac{\mathbf{H}_{ik}\mathbf{U}_{kj}}{\lambda_{ijk}} + \sum_{\boldsymbol{k}} \mathbf{H}_{ik}\mathbf{U}_{kj} \right)$$

Update:H

$$\frac{\partial Q}{\partial \mathbf{H}_{il}} = \sum_{j} \frac{\partial}{\partial \mathbf{H}_{il}} \left(-\mathbf{X}_{ij} \sum_{k} \lambda_{ijk} \log \mathbf{H}_{ik} + \sum_{\boldsymbol{k}} \mathbf{H}_{ik}\mathbf{U}_{kj} \right) = 0$$

$$\sum_{j} \left(-\mathbf{X}_{ij}\lambda_{ijk} \left(\frac{1}{\mathbf{H}_{il}} \right) + \mathbf{U}_{lj} \right) = 0$$

$$\mathbf{H}_{il} = \frac{\sum_j \mathbf{X}_{ij}\lambda_{ijk}}{\sum_j \mathbf{U}_{lj}}$$

Update:U

$$\frac{\partial Q}{\partial \boldsymbol{U}_{lj}} = \sum_{i} \frac{\partial}{\partial \mathbf{U}_{lj}} \left(-\mathbf{X}_{ij} \sum_{k} \lambda_{ijk} \log \mathbf{U}_{kj} + \sum_{\boldsymbol{k}} \mathbf{H}_{ik}\mathbf{U}_{kj} \right) = 0$$

$$\sum_{i} \left(-\mathbf{X}_{ij}\lambda_{ijk} \left(\frac{1}{\mathbf{U}_{lj}} \right) + \mathbf{U}_{lj} \right) = 0$$

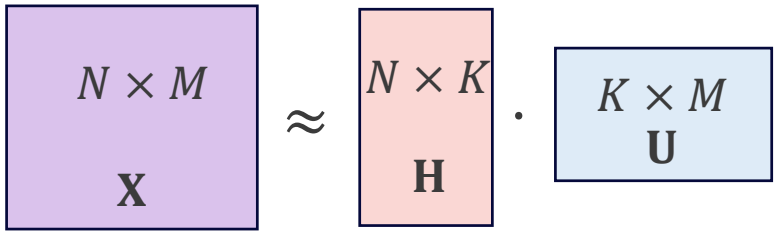
$$\mathbf{U}_{lj} = \frac{\sum_i \mathbf{X}_{ij}\lambda_{ijk}}{\sum_j \mathbf{U}_{lj}}$$

NMFの損失関数：βダイバージェンス

損失関数：$d_\beta(\mathbf{X}, \mathbf{HU})$

βダイバージェンス

$$d_\beta(x,y) = \frac{y^\beta}{\beta(\beta-1)} + \frac{x^\beta}{\beta} - \frac{yx^{\beta-1}}{\beta-1}$$



beta_loss : *float or {'frobenius', 'kullback-leibler', 'itakura-saito'}, default='frobenius'*

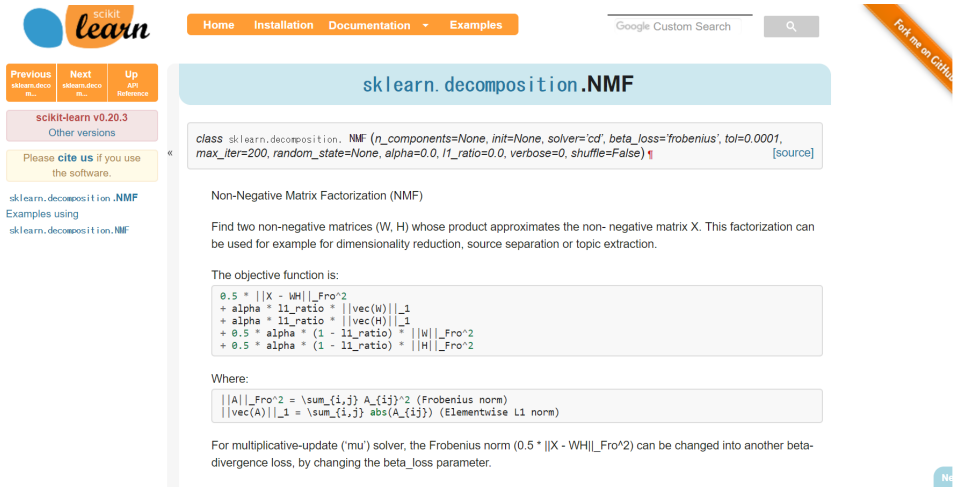
Beta divergence to be minimized, measuring the distance between X and the dot product WH. Note that values different from 'frobenius' (or 2) and 'kullback-leibler' (or 1) lead to significantly slower fits. Note that for beta_loss <= 0 (or 'itakura-saito'), the input matrix X cannot contain zeros. Used only in 'mu' solver.

New in version 0.19.

$\beta = 2$: 二乗誤差（Frobeniusノルム）

$\beta = 1$: 一般化KLダイバージェンス

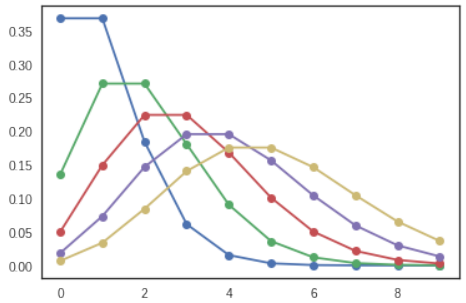
$\beta = 0$: 板倉斎藤ダイバージェンス



NMFと確率モデル

一般化KLダイバージェンスから導出される損失

$$Loss = \sum_{ij} -\mathbf{X}_{ij}\log\left(\mathbf{HU}\right)_{ij} + \left(\mathbf{HU}\right)_{ij}$$



ポアソン分布を考えると

$$Poisson\left(X_{ij}\,|(\mathbf{HU})_{ij}\right) = \prod_{ij} \frac{(\mathbf{HU})_{ij}^{\mathbf{X}_{ij}} \exp -(\mathbf{HU})_{ij}}{\mathbf{X}_{ij}!}$$

$$-\log Poisson\left(X_{ij}\,|(\mathbf{HU})_{ij}\right) = -\log \prod_{ij} \frac{(\mathbf{HU})_{ij}^{\mathbf{X}_{ij}} \exp -(\mathbf{HU})_{ij}}{\mathbf{X}_{ij}!}$$

$$= \sum_{ij} -\mathbf{X}_{ij}\log(\mathbf{HU})_{ij} + (\mathbf{HU})_{ij} + \log \mathbf{X}_{ij}!$$

一致

$\beta = 2$: 二乗誤差（Frobeniusノルム）　ガウス分布を仮定した最尤法と一致

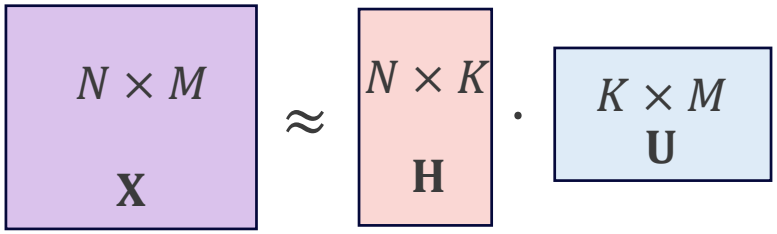
$\beta = 1$: 一般化KLダイバージェンス　ポアソン分布を仮定した最尤法と一致

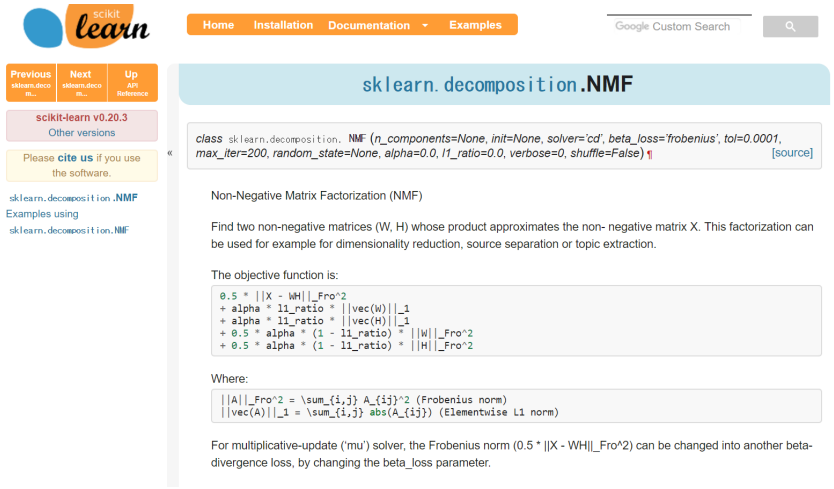
$\beta = 0$: 板倉斎藤ダイバージェンス　指数分布を仮定した最尤法と一致

基底の数：Kをどうやって決めるのか？

- クロスバリデーション
- 情報量規準
- Bayesian NMF

（第３回機械学習講義）





Baysian NMFはscikit-learn には未実装

まとめ（前半）

テーブルデータ

Automated Machine Learningなどを使った予測モデルの構築方法と評価値や説明可能ＡＩ技術の使い方などを紹介した

実際に上記技術を用いての予測モデル構築の演習

行列データ

行列分解を用いたデータ解析方法の紹介

- ＰＣＡの詳細と使い方
- ＮＭＦの紹介

後半は、時系列や関係データなどのより複雑なデータに関する話

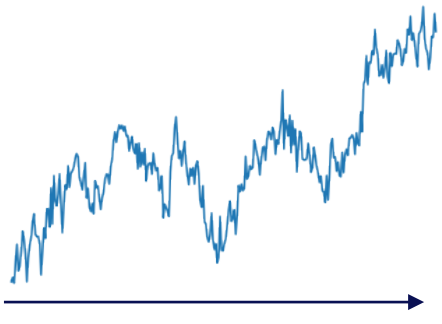
系列データ

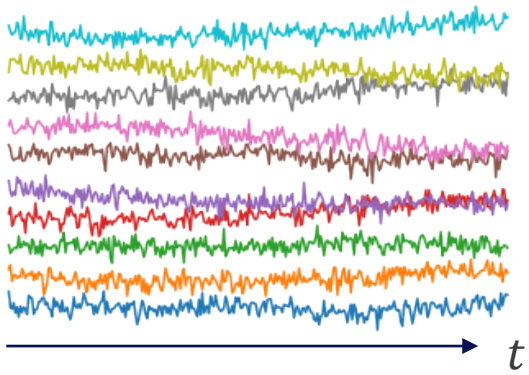
テーブルデータや行列データとして表現されていることも多い

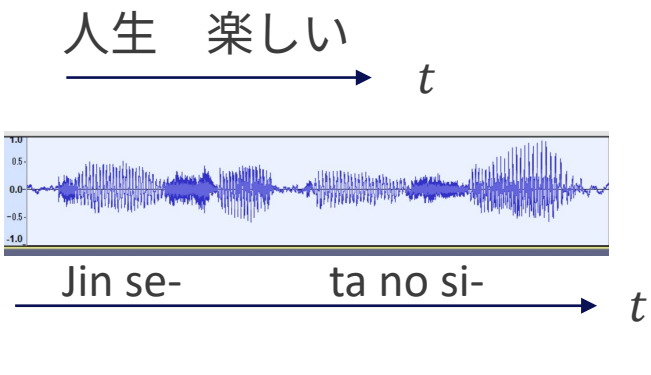
- 時系列データ
- 信号データ
  - センサデータ
  - 多チャンネルデータ
  - 音声、動画
- シンボル系列
  - ログや履歴
  - テキスト

日付	項目A	項目B
2019/01/02	2.2	A
2019/01/03	2.4	B
2019/01/05	2.1	C
2019/01/08	1.8	B
2019/01/10	1.7	B

t





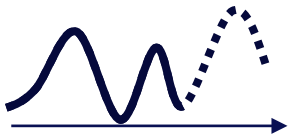


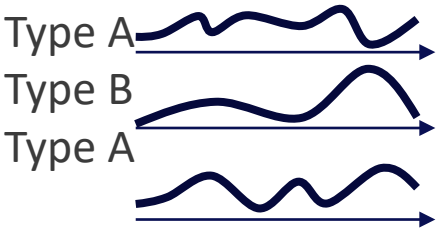
系列データに対する機械学習の流れ

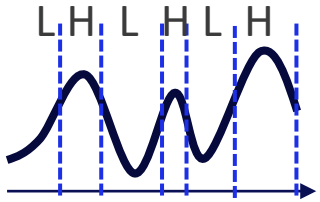
系列データの前処理

系列データの識別

- 将来の予測
- 系列単位での識別
- 各時刻に対する識別タグ付け







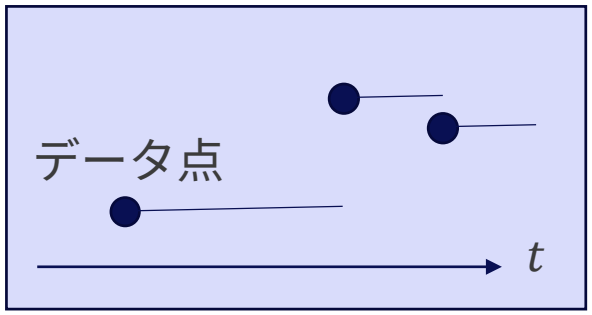
系列のクラスタリング

- 系列単位でのクラスタリング
- 各時刻に対する教師なしタグ付け

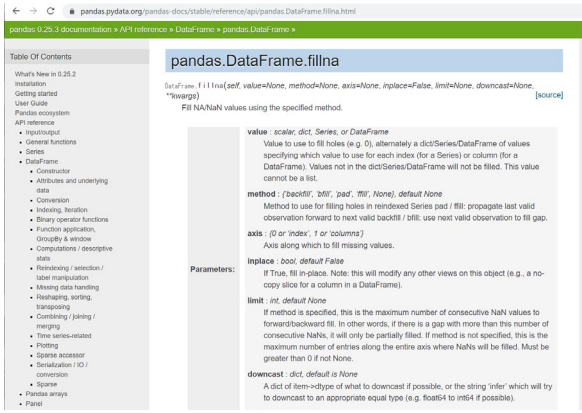
前処理（1/2）：欠損値補間

欠損の補間時に時間方向を考慮する必要がある

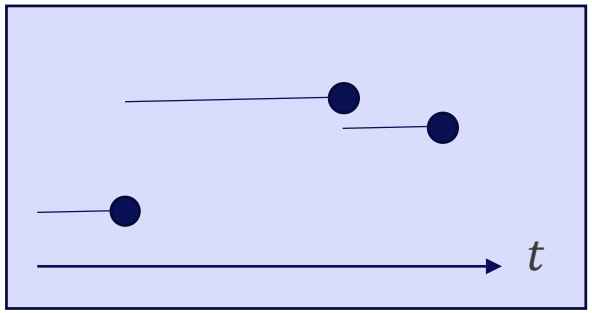
前方向の補間



fillna(method='**ffill**')

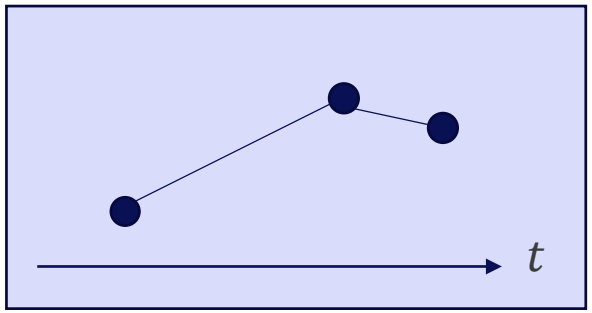


後ろ方向の補間



fillna(method='**bfill**')

線形補間



interpolate(method='**linear**')

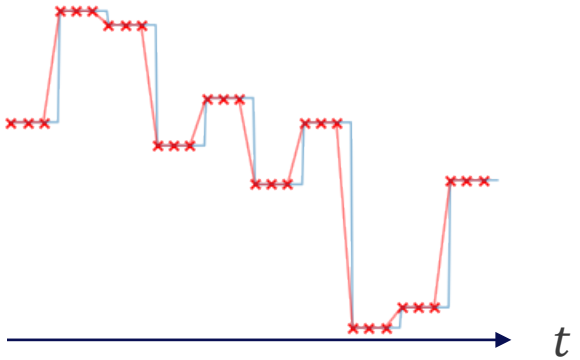
前処理（2/2）:リサンプリング

時系列データを元データより高い頻度または低い頻度で再度サンプリングすることをリサンプリングという

他のシステムやデータとの整合性をとるために使う

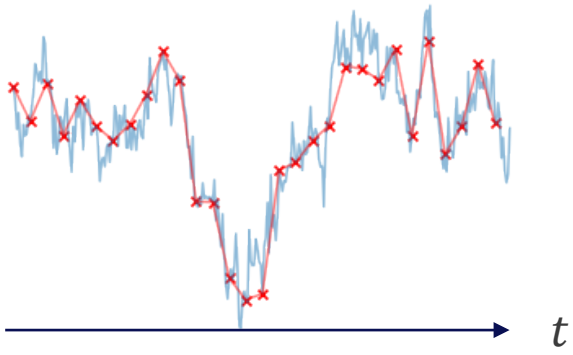
アップサンプリング（オーバーサンプリング）

- より高い頻度（短い周期）でリサンプリング



ダウンサンプリング（アンダーサンプリング）

- より低い頻度（長い周期）でリサンプリング
- 欠損を減らす目的でも利用される



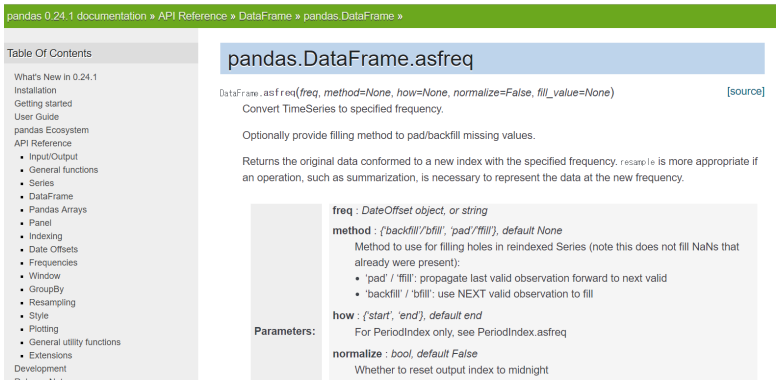

pandas 0.24.1 documentation » API Reference » DataFrame » pandas.DataFrame »

Table Of Contents

What's New in 0.24.1
Installation
Getting started
User Guide
pandas Ecosystem
API Reference
- Input/Output
- General functions
- Series
- DataFrame
- Pandas Arrays
- Panel
- Indexing
- Date Offsets
- Frequencies
- Window
- GroupBy
- Resampling
- Style
- Plotting
- General utility functions
- Extensions

Development

pandas.DataFrame.asfreq

DataFrame.asfreq(*freq*, *method=None*, *how=None*, *normalize=False*, *fill_value=None*) [source]

Convert TimeSeries to specified frequency.

Optionally provide filling method to pad/backfill missing values.

Returns the original data conformed to a new index with the specified frequency. resample is more appropriate if an operation, such as summarization, is necessary to represent the data at the new frequency.

Parameters:

freq : *DateOffset object, or string*

method : *{'backfill'/'bfill', 'pad'/'ffill'}, default None*
Method to use for filling holes in reindexed Series (note this does not fill NaNs that already were present):
- 'pad' / 'ffill': propagate last valid observation forward to next valid
- 'backfill' / 'bfill': use NEXT valid observation to fill

how : *{'start', 'end'}, default end*
For PeriodIndex only, see PeriodIndex.asfreq

normalize : *bool, default False*
Whether to reset output index to midnight

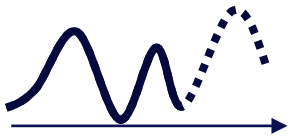
系列データに対する機械学習の流れ

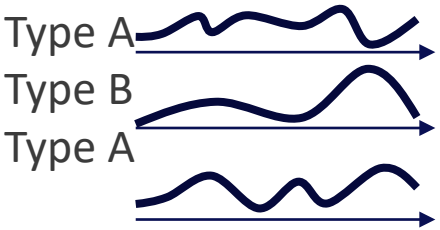
系列データの前処理

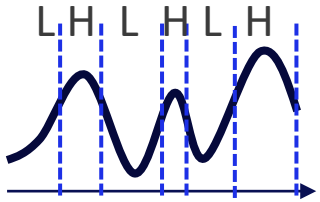
欠損値補間
リサンプリング

系列データの識別

- 将来の予測
- 系列単位での識別
- 各時刻に対する識別タグ付け







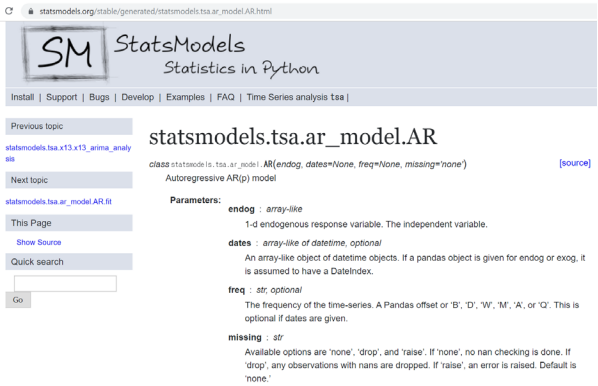
系列のクラスタリング

- 系列単位でのクラスタリング
- 各時刻に対する教師なしタグ付け

系列データの予測

自己回帰モデル（Auto regression model）

系列データの未来の予測に利用できる
最もシンプルな方法


statsmodels.org/stable/generated/statsmodels.tsa.ar_model.AR.html

StatsModels
Statistics in Python

Install | Support | Bugs | Develop | Examples | FAQ | Time Series analysis tsa |

Previous topic

statsmodels.tsa.x13.x13_arima_analysis

Next topic

statsmodels.tsa.ar_model.AR.fit

This Page

Show Source

Quick search

Go

statsmodels.tsa.ar_model.AR

class statsmodels.tsa.ar_model.AR(endog, dates=None, freq=None, missing='none') [source]

Autoregressive AR(p) model

Parameters:

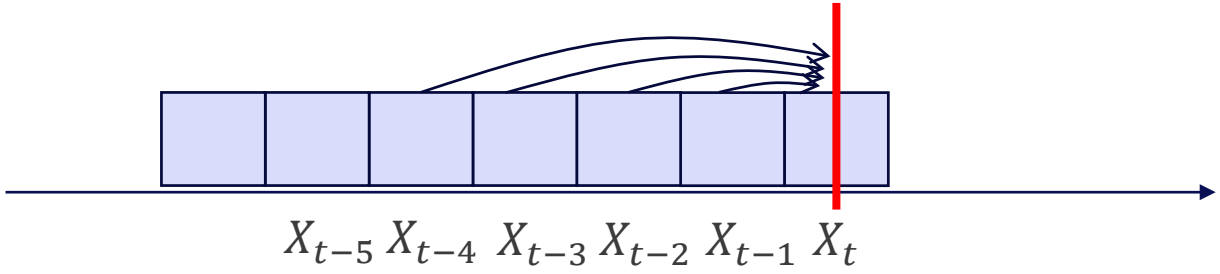
endog : array-like
1-d endogenous response variable. The independent variable.

dates : array-like of datetime, optional
An array-like object of datetime objects. If a pandas object is given for endog or exog, it is assumed to have a DateIndex.

freq : str, optional
The frequency of the time-series. A Pandas offset or 'B', 'D', 'W', 'M', 'A', or 'Q'. This is optional if dates are given.

missing : str
Available options are 'none', 'drop', and 'raise'. If 'none', no nan checking is done. If 'drop', any observations with nans are dropped. If 'raise', an error is raised. Default is 'none.'

4つ前までの時刻に依存したARモデルをAR(4)のように書く



AR(p)

$$X_t = w_0 + \sum_{i=1}^{p} w_i X_{t-i} + \epsilon_t \qquad \epsilon:ノイズ$$

w_i：に関しては最小二乗法など通常の線形モデルの方法で
推定することができる

信号処理の分野のフィルタとも深い関係がある

ARモデルの仲間

AR(p)

$$X_t = w_0 + \sum_{i=1}^{p} w_i X_{t-i} + \epsilon_t$$

ϵ:ノイズ

$$E[\epsilon_t] = 0$$
$$E[\epsilon_t^2] = \sigma^2$$
$$E[\epsilon_i \epsilon_j] = 0 \quad i \neq j$$

MA(q)

$$X_t = w_0 + \sum_{i=0}^{q} \beta_i \epsilon_{t-i}$$

ARMA(q)

$$X_t = w_0 + \sum_{i=1}^{p} w_i X_{t-i} + \sum_{i=0}^{q} \beta_i \epsilon_{t-i}$$

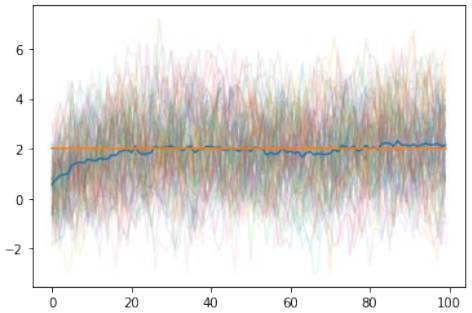
$$E[X_t] = w_0 + \sum_{i=1}^{p} w_i E[X_{t-i}] + \sum_{i=0}^{q} \beta_i E[\epsilon_{t-i}]$$
$$= w_0 + \sum_{i=1}^{p} w_i E[X_{t-i}]$$

$$E[X_t] = E[X_{t-i}] = \text{const}$$

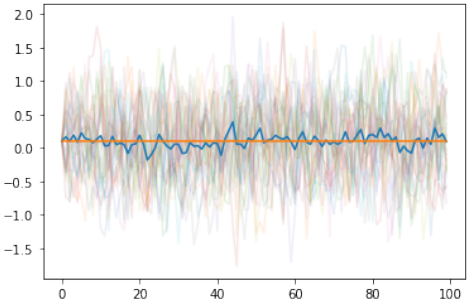
$$E[X_t] = \frac{w_0}{1 - w_1 - w_2 - \cdots w_p}$$

自分自身で予測を
繰り返してプロット

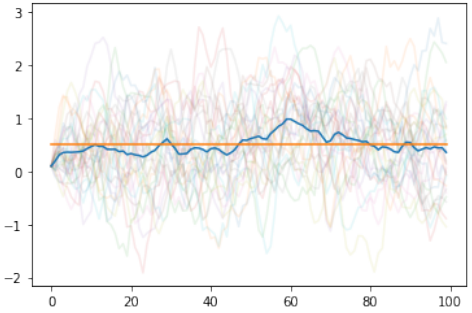
AR



MA

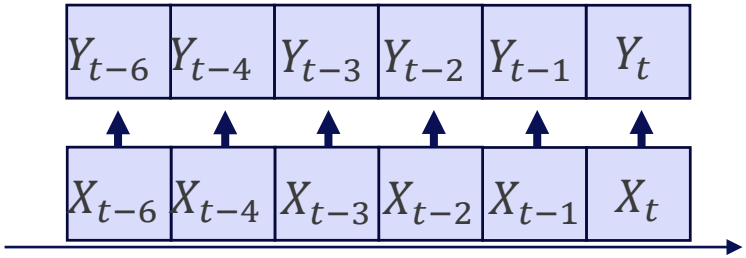


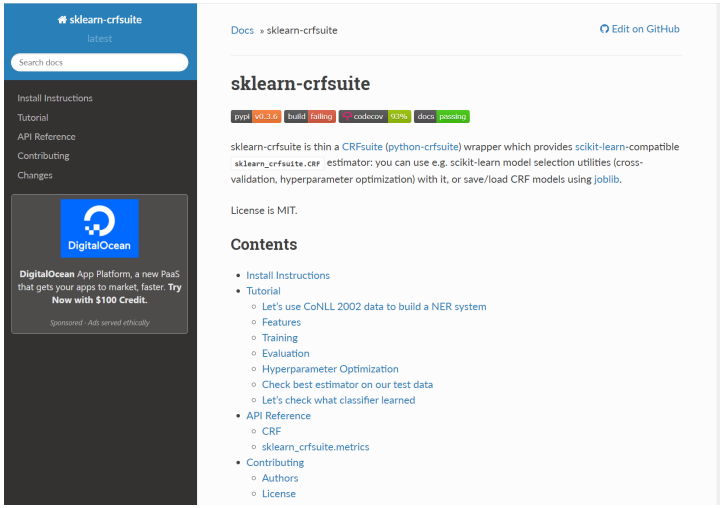
ARMA



系列の教師あり学習

Linear chain CRF(Conditional Random Fields)
系列を予測する識別モデル
（未来を予測するのではなく系列のタグを予測する場合によく使われる）





$$P(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x}, y)} \exp(\boldsymbol{w} \cdot \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{y}))$$

$\boldsymbol{\phi}$：系列中の特徴
$\boldsymbol{w}$：特徴の重み
Z：正規化項

最近だと深層学習ライブラリを使うことも多いかもしれない
tensorflow_addons.layers.CRF
pytorch-crf

形態素解析（文章から品詞を識別する問題）への応用が有名

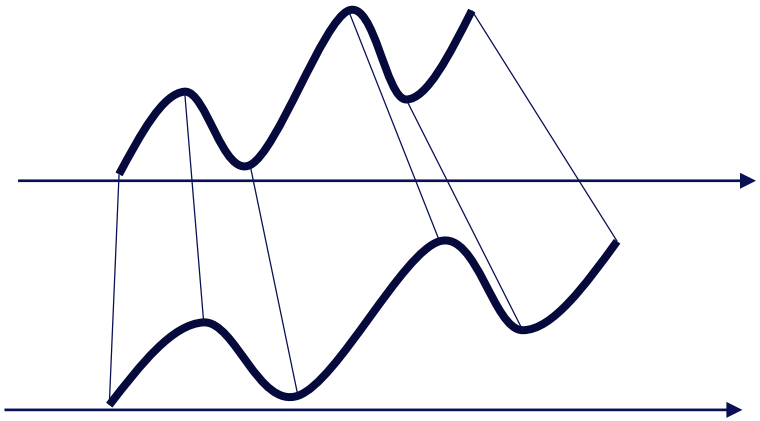
$\boldsymbol{y}$　　0.3　0.2
　　名詞—助詞—形容詞
　0.5　0.5　0.5
$\boldsymbol{x}$　　人生　は　楽しい

$\boldsymbol{\phi}$:エッジに対応

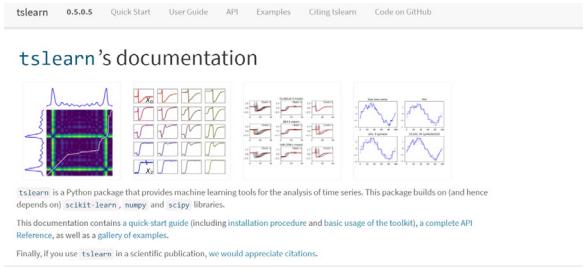
系列の識別・クラスタリング

距離に基づくクラスタリング・識別： DTW・レーベンシュタイン距離（編集距離）とクラスタリング・識別手法を合わせて使う

Dynamic Time Warping (DTW)

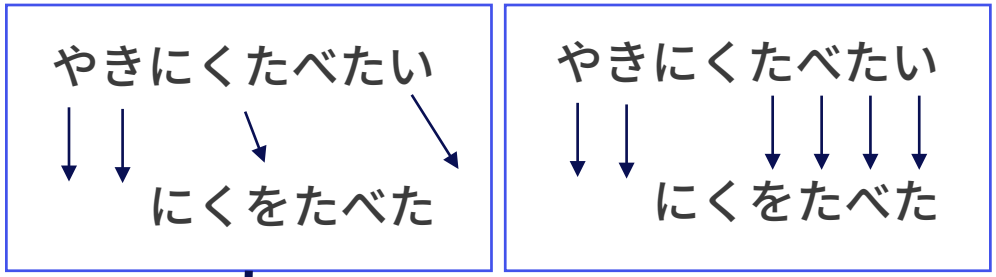


伸長部分を合わせながら二つの系列データの距離を測ることができる
信号のマッチングなどで使われる



レーベンシュタイン距離（編集距離）

一文字挿入(insertion), 一文字削除(deletion), 一文字置換(substitution)
を最小何回で変換できるかを距離とみなす



最小はこちら

レーベンシュタイン距離＝４

実装
- python-levenshtein
- editdistance

系列データに対する機械学習の流れ

系列データの前処理

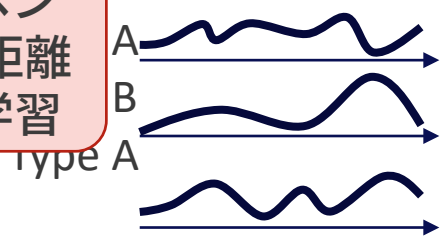
欠損値補間
リサンプリング

系列データの識別

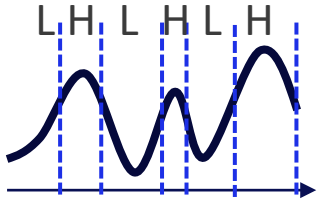
- 将来の予測
- 系列単位での識別
- 各時刻に対する識別タグ付け

ARモデル

DTW,レーベンシュタイン距離
+教師あり学習



Linear-chain
CRF



系列のクラスタリング

- 系列単位でのクラスタリング
- 各時刻に対する教師なしタグ付け

DTW,レーベンシュタイン距離
+クラスタリング

系列データに対する確率モデル（生成モデル）

（離散時間）マルコフモデル

- 一つ前の時刻に依存して次の値が確率的に決まるモデル
- k時刻前～一つ前の時刻に依存して次の値が決まる場合はセミマルコフモデルと呼ぶ

隠れマルコフモデル

- マルコフモデルに従う状態が直接観測できないモデル

系列データに対する機械学習の流れ

系列データの前処理

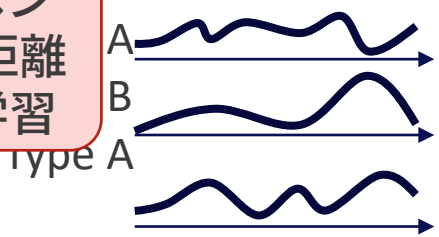
欠損値補間
リサンプリング

系列データの識別

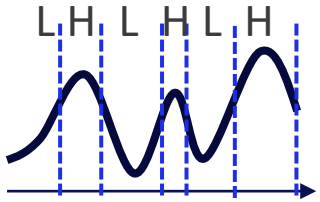
- 将来の予測
- 系列単位での識別
- 各時刻に対する識別タグ付け

ARモデル
マルコフモデル

DTW,レーベンシュタイン距離
+教師あり学習



Linear-chain
CRF



系列のクラスタリング

- 系列単位でのクラスタリング
- 各時刻に対する教師なしタグ付け

DTW,レーベンシュタイン距離
+クラスタリング

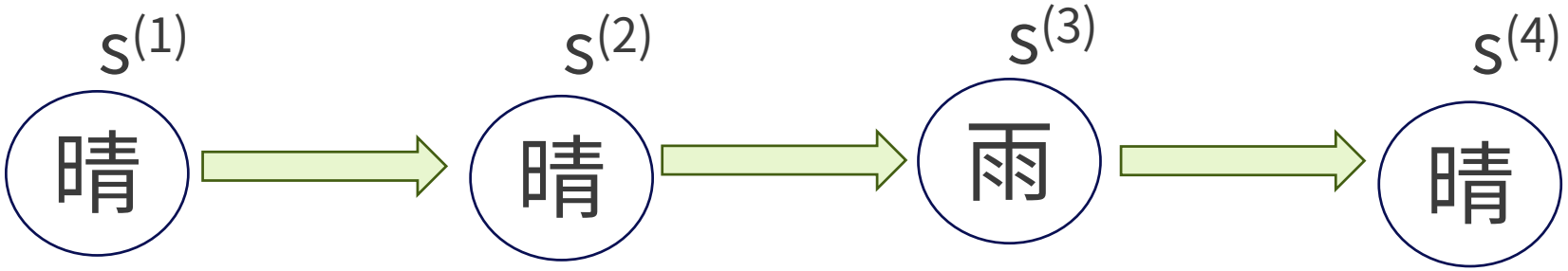
隠れマルコフモデル

離散時間ー離散状態マルコフモデル

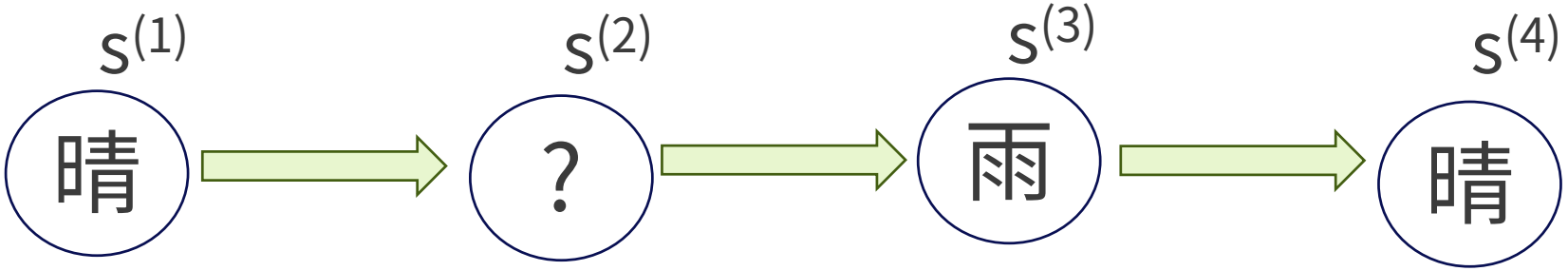
一つ前の時刻に依存して次の値が確率的に決まるモデル

今の状態

次の状態	晴	雨
晴	0.9	0.2
雨	0.1	0.8



この系列が得られる確率は：$0.9 \times 0.1 \times 0.2 = 0.018$

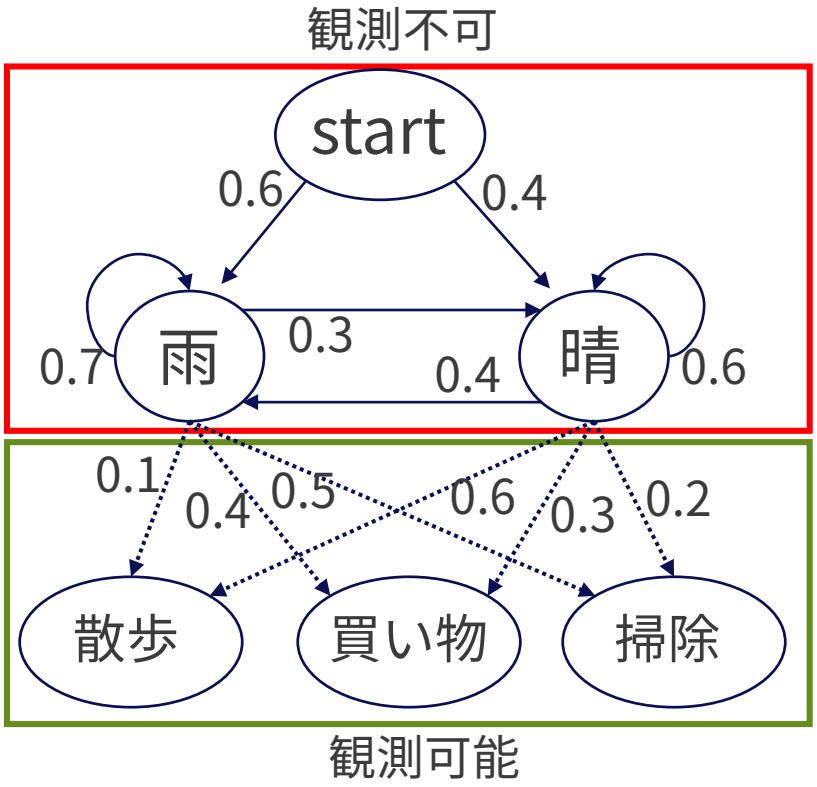


この系列が得られる確率は：$0.9 \times 0.1 \times 0.2 + 0.1 \times 0.8 \times 0.2 = 0.018 + 0.016$

隠れマルコフモデル
(HMM: Hidden Markov Model)

離れた友人と電話をしてその日何をしたかを聞く。彼の毎日の行動に基づいて、その日の天気を推測。

まずは右図の確率はわかっているとする



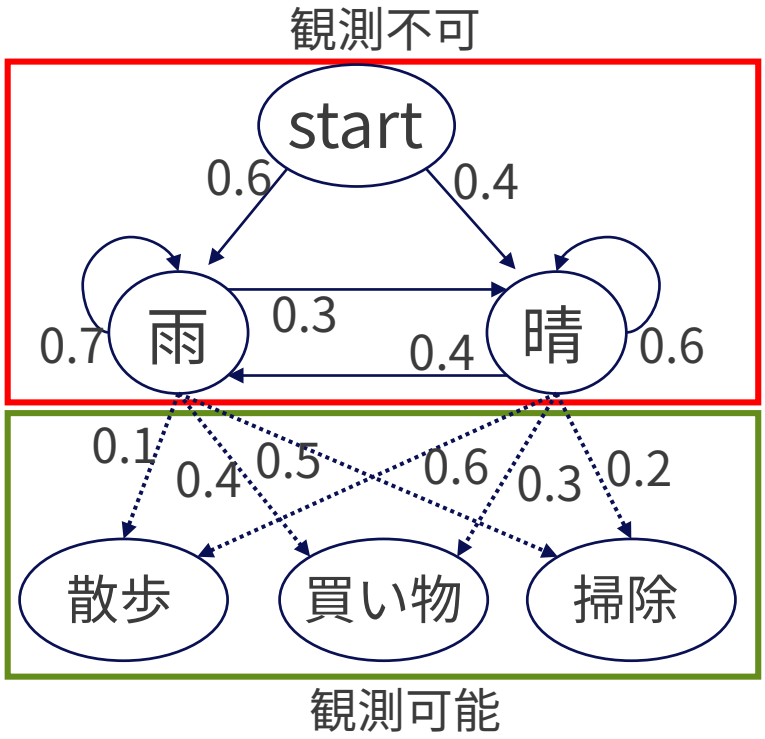
一部が観測不可で一部が観測可能な状況で広く応用されている

- 系列のセグメンテーション
  - 音声認識（波形が観測されるが、文字は観測されない）
- 系列の検索(profile HMM)
  - 文字列検索
- 系列のクラスタリング

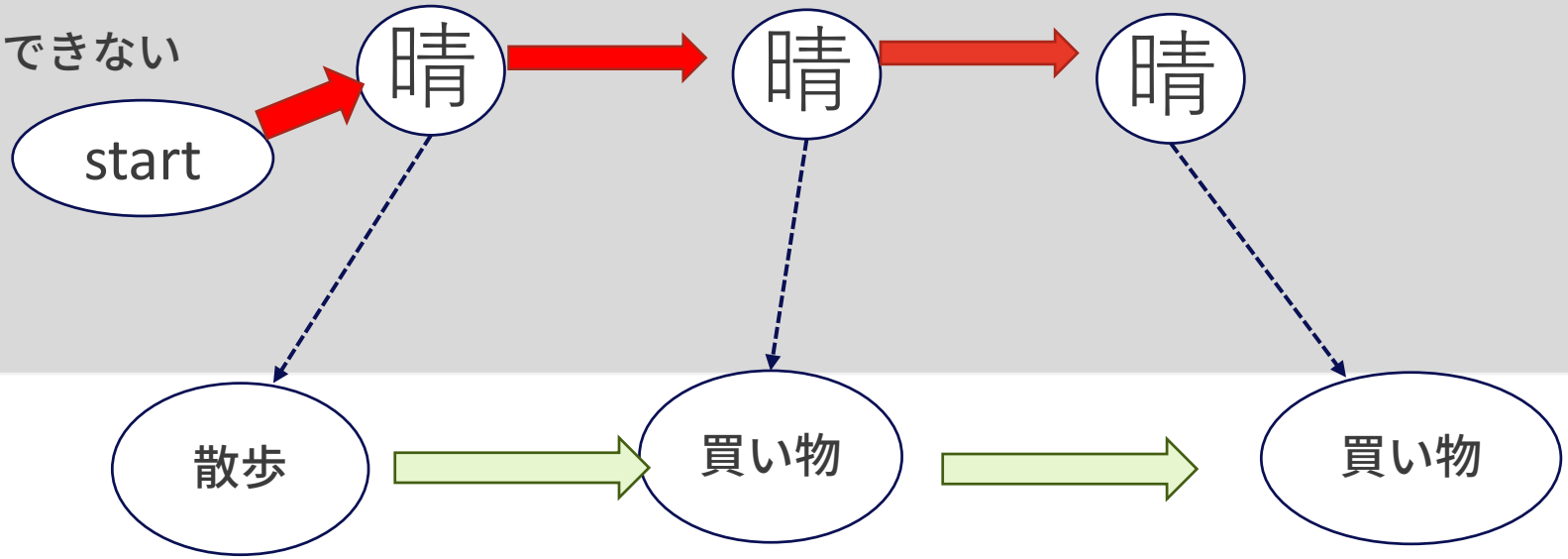
隠れマルコフモデル
(HMM: Hidden Markov Model)

かなり広く応用されている

- 系列のセグメンテーション
  - 音声認識
- 系列の検索(profile HMM)
  - 文字列検索
- 系列のクラスタリング



本当は観測できない

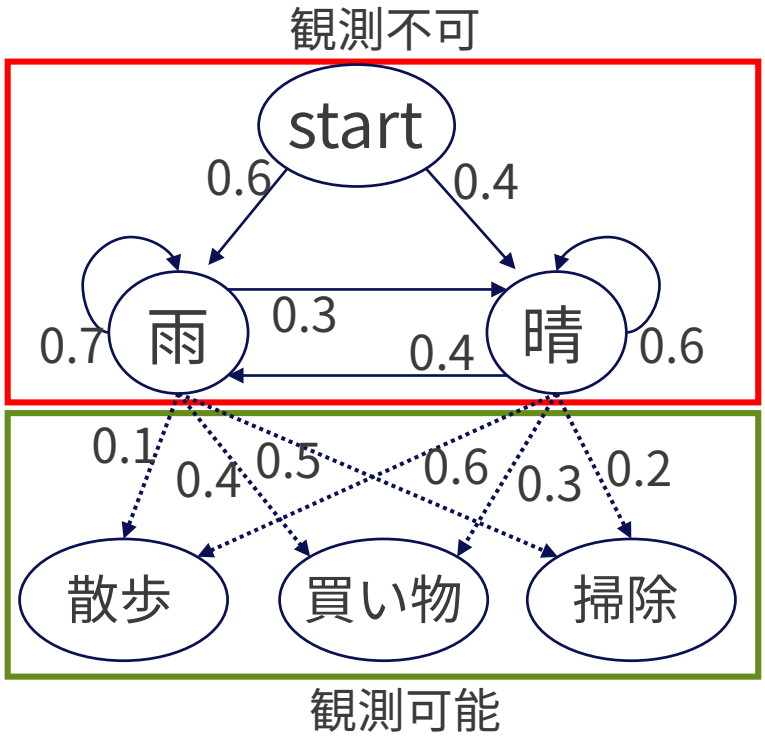


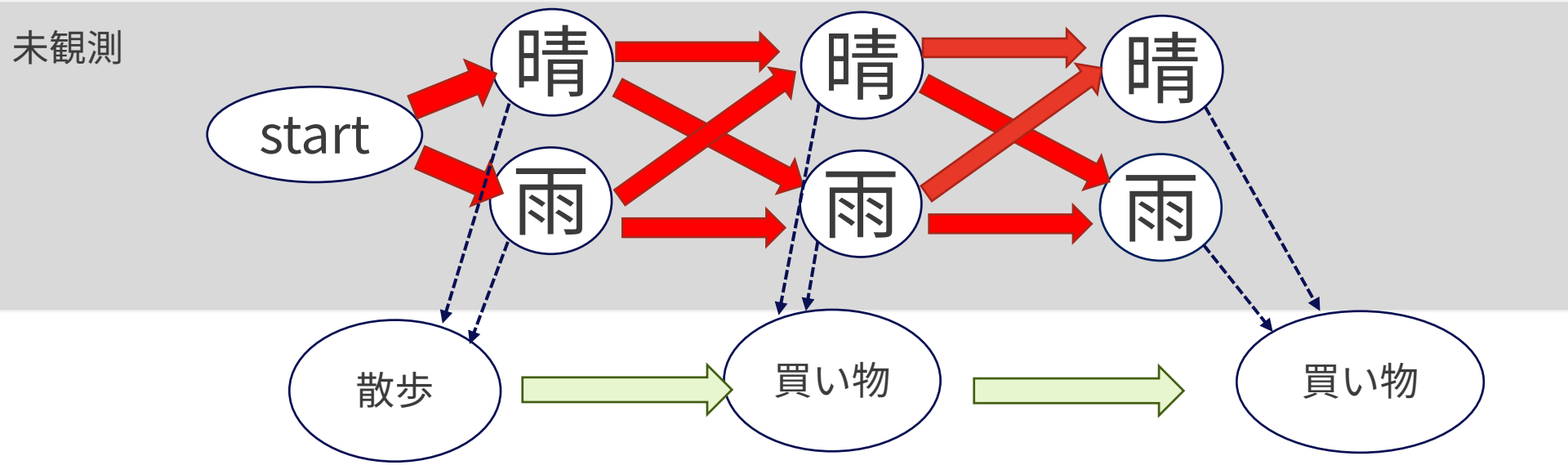
この事象（パス）が起こる確率は：$0.4 \times 0.6 \times 0.6 \quad \times \quad 0.6 \times 0.3 \times 0.3$

隠れマルコフモデル
(HMM: Hidden Markov Model)

かなり広く応用されている

- 系列のセグメンテーション
  - 音声認識
- 系列の検索(profile HMM)
  - 文字列検索
- 系列のクラスタリング



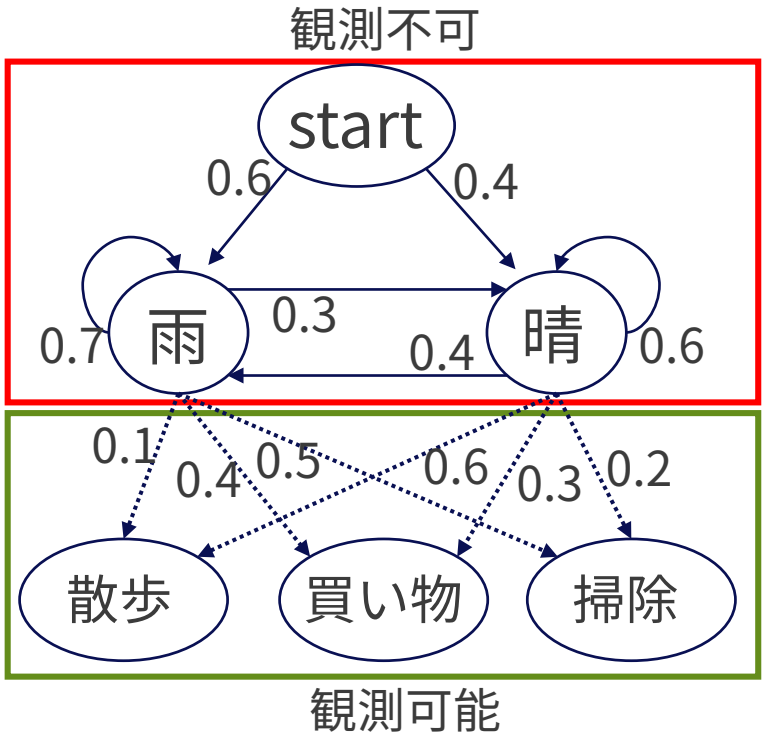


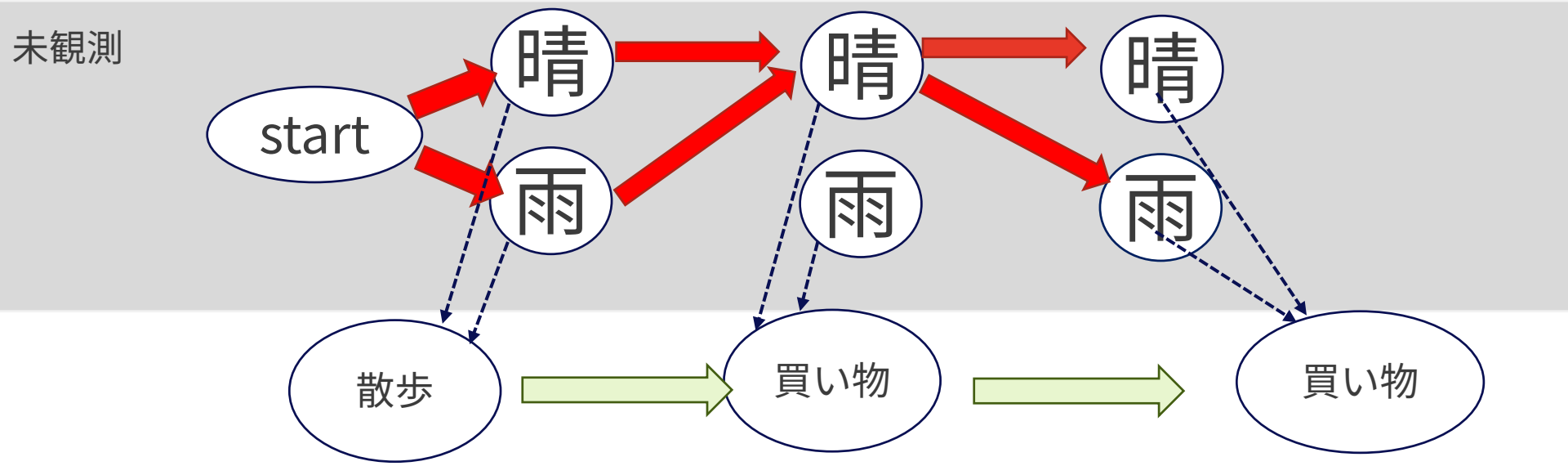
この系列が得られる確率は$2 \times 2 \times 2$通りのパスの確率の足し算

隠れマルコフモデル
(HMM: Hidden Markov Model)

かなり広く応用されている

- 系列のセグメンテーション
  - 音声認識
- 系列の検索(profile HMM)
  - 文字列検索
- 系列のクラスタリング

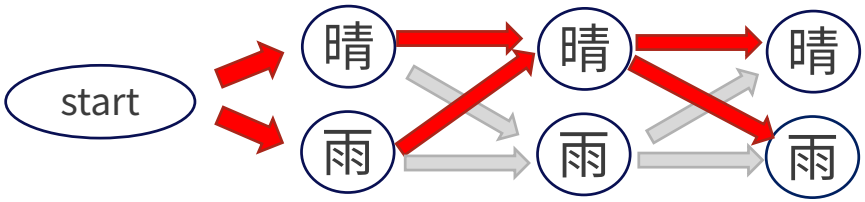




この系列が得られたときに二日目が晴れの確率は2×2通りのパスの確率の足し算

二日目が晴れの確率の計算

時刻２で晴の確率が知りたい時

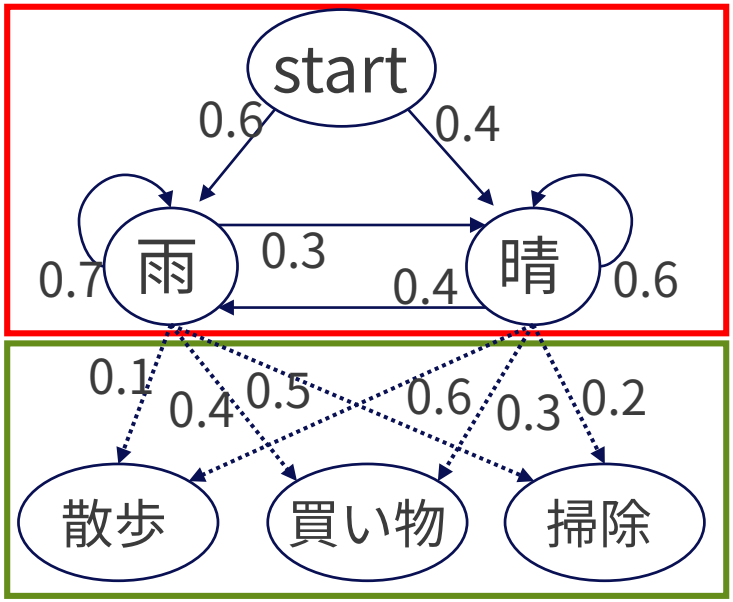


観測：　散歩 ⇨ 買い物 ⇨ 買い物

$P_t(A)$：時刻tで晴の確率　A:晴、B:雨

晴晴晴：$P_1(A) \times P_2(A|A) \times P_3(A|A)$
晴晴雨：$P_1(A) \times P_2(A|A) \times P_3(B|A)$
晴雨晴：$P_1(A) \times P_2(B|A) \times P_3(A|B)$
晴雨雨：$P_1(A) \times P_2(B|A) \times P_3(B|B)$
雨晴晴：$P_1(B) \times P_2(A|B) \times P_3(A|A)$
雨晴雨：$P_1(B) \times P_2(A|B) \times P_3(B|A)$
雨雨晴：$P_1(B) \times P_2(B|B) \times P_3(A|B)$
雨雨雨：$P_1(B) \times P_2(B|B) \times P_3(B|B)$

観測不可



観測可能

晴晴雨 の確率
$P_1(A) = 0.4 \times 0.6$
$P_2(A|A) = 0.6 \times 0.3$
$P_3(B|A) = 0.4 \times 0.4$

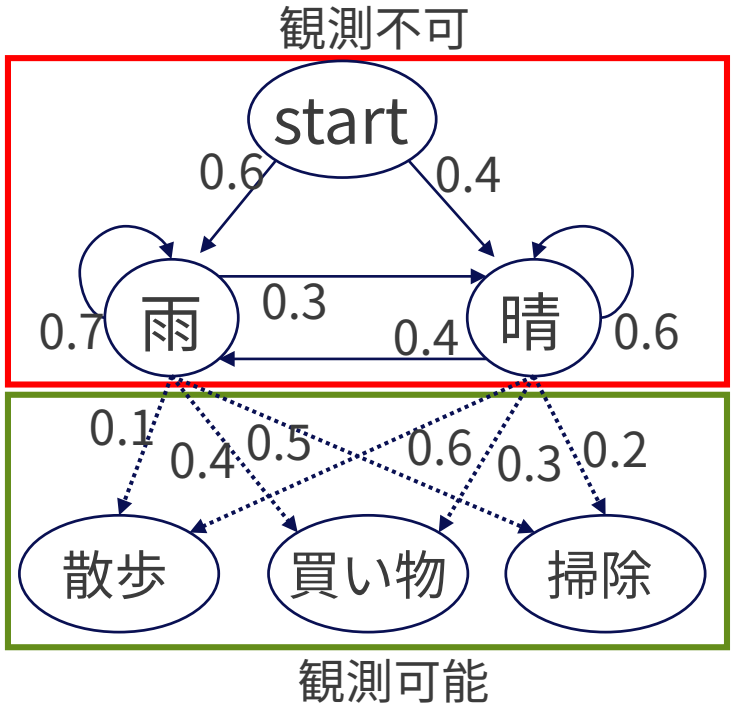
時刻２で晴の確率はこれらの和
掛け算8回、足し算3回

積和計算を少なくできる
（Forward-backwardアルゴリズム）

$$\left(P_1(A) \times P_2(A|A) + P_1(B) \times P_2(A|B)\right) \times \left(P_3(A|A) + P_3(B|A)\right)$$

掛け算3回、足し算2回

隠れマルコフモデル
(HMM: Hidden Markov Model)



隠れマルコフモデルに対して可能なこと

- 確率計算
- トポロジーが決まっているとき観測系列だけから確率値を計算

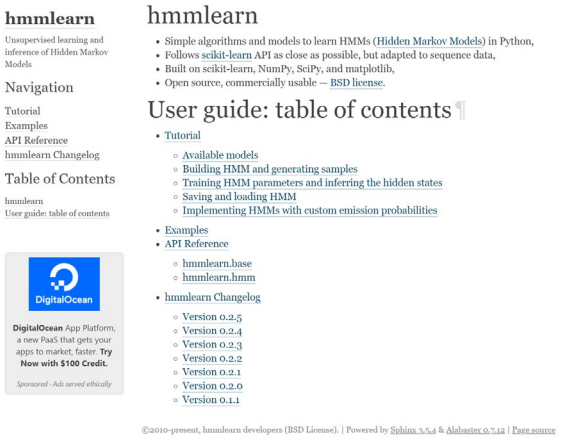
（EM アルゴリズム）

- トポロジーと確率が決まっているとき最も確率の高い状態を計算

（Viterbi アルゴリズム）

- ベイズ推論によりトポロジーを評価・探索することで、観測系列からトポロジーと確率を推定

（e.g. 変分ベイズHMM）



系列データに対する機械学習の流れ

系列データの前処理

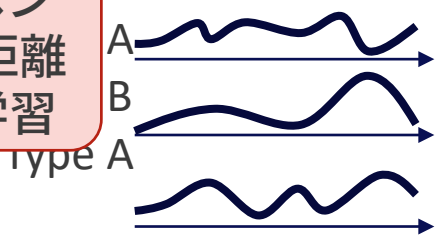
欠損値補間
リサンプリング

系列データの識別

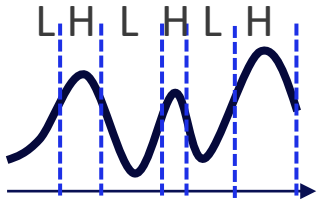
- 将来の予測
- 系列単位での識別
- 各時刻に対する識別タグ付け

ARモデル
マルコフモデル

DTW,レーベンシュタイン距離
+教師あり学習



Linear-chain
CRF



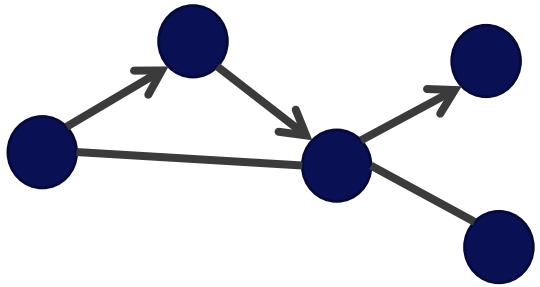
系列のクラスタリング

- 系列単位でのクラスタリング
- 各時刻に対する教師なしタグ付け

DTW,レーベンシュタイン距離
+クラスタリング

隠れマルコフモデル

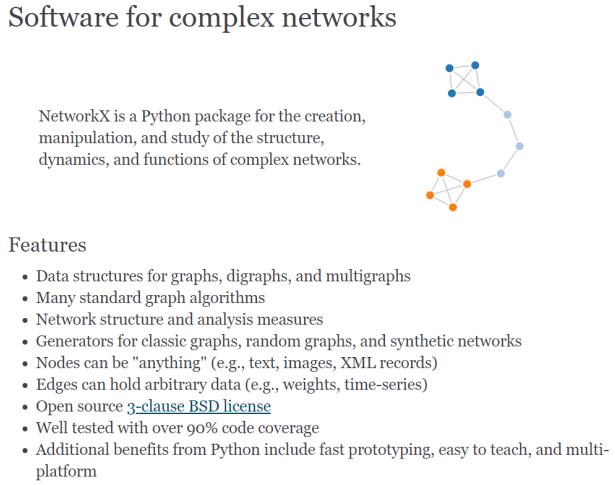
グラフデータ



ノードの集合とその間のエッジの集合
- 有向グラフ
- 無向グラフ
- 非巡回グラフ：ループのないグラフ
- 二分グラフ：２群の間のグラフ
- ハイパーグラフ：複数ノードをつなげる
  エッジを許すグラフ

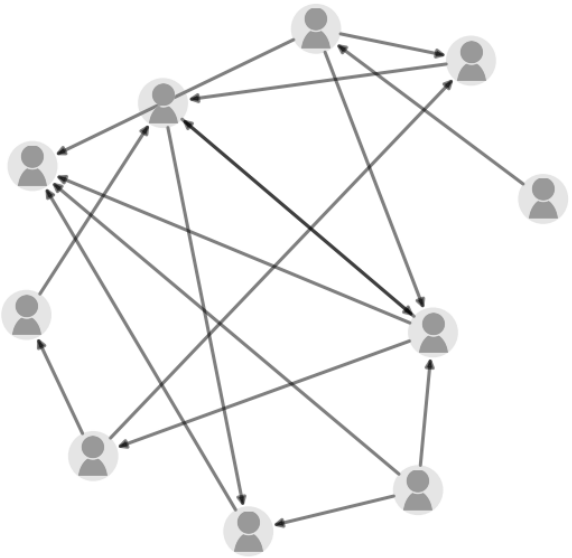
ネットワーク
- グラフのエッジに重みがあるもの

ノードの次数
- そのノードにつながってるエッジの数
- 有効の場合は入次数・出次数



<u>応用</u>
- ソーシャルネットワーク
- 多次元データから距離を用いてグラフを構築する（Similarity graph）
- 化合物（構造式）
- 知識グラフ

グラフの例(1/2)



社会ネットワーク

(SNSのフォローフォロワー関係)

応用例：

- 影響力の調査（PageRank）
- コミュニティ、クラスタリング
- 推薦
- ボットや犯罪者などの検出

ユーザと購入商品のグラフ

二部グラフの例

応用例：

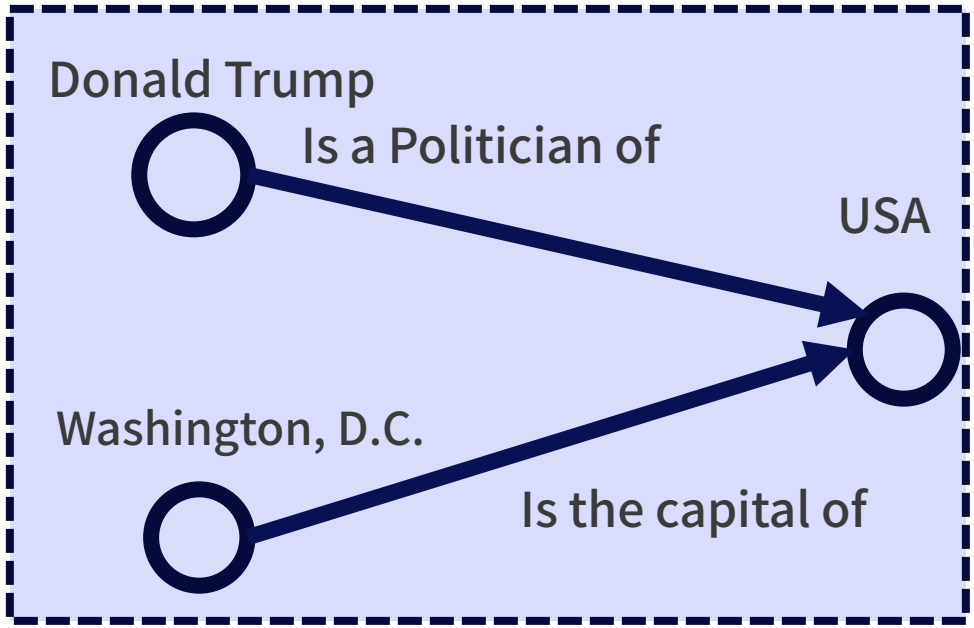
推薦（新規リンクの予測）

協調フィルタリング

グラフの例(2/2)

知識グラフ

h	ℓ	t
Donald Trump	Is a politician of	USA
Washington, D.C.	Is the capital of	USA



Google knowledge graph

- 70 billion facts(2016)

https://www.theverge.com/2016/10/4/13122406/google-phone-event-stats

WolframAlpha

- Siriにも含まれている

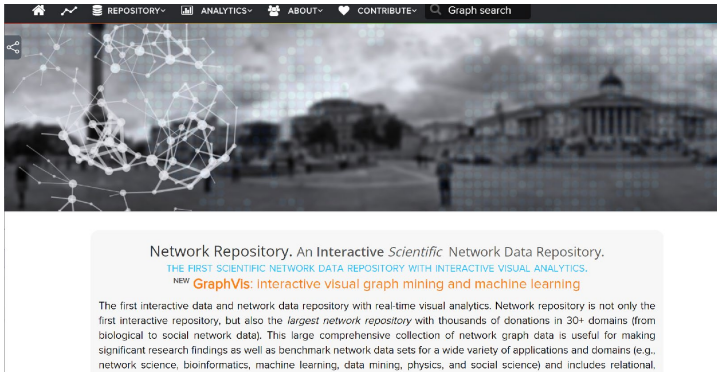
NELL

- 知識の自動獲得

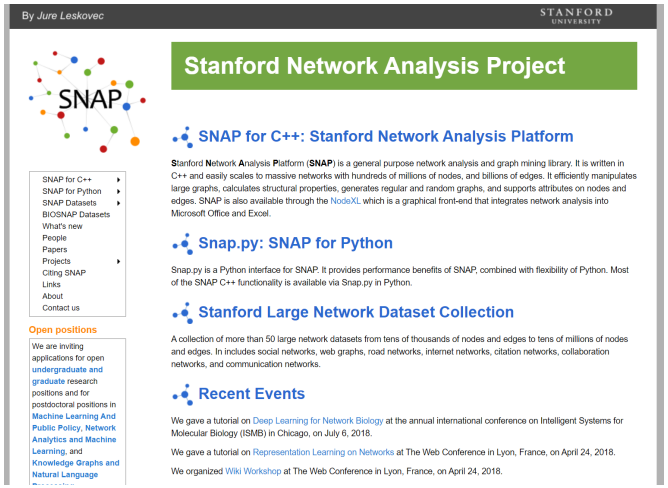
グラフデータの取得

グラフの解析を試したい場合は、例えば以下のサイトから
各種グラフデータがダウンロードできる

http://networkrepository.com/



http://snap.stanford.edu/



グラフデータの代表的な保存方法
　基本的には3列のテーブルで保存されていることが多い

ノード（始点）	関係	ノード（終点）
グー	勝つ	チョキ
チョキ	勝つ	パー
パー	勝つ	グー

グラフデータに対する処理

グラフデータの前処理、グラフの構築
- グラフの構築法：相関行列、類似度行列
- グラフの表現方法：隣接行列、接続行列

→ グラフ上の重要な統計量
- 平均次数、次数分布、スケールフリー性など

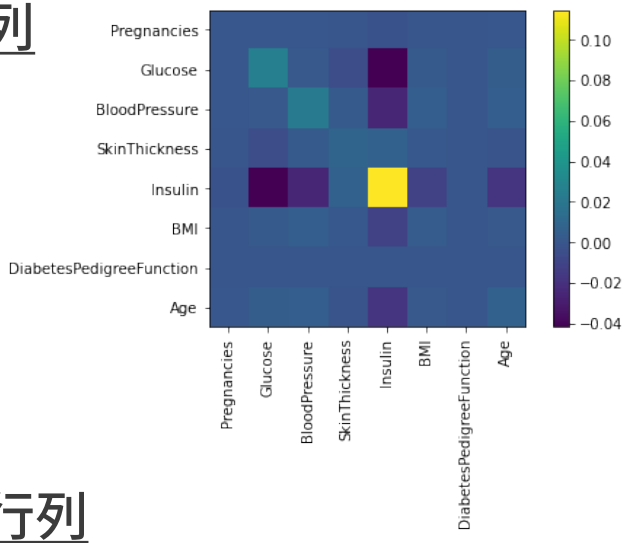
→ グラフに関する予測
- グラフ上の予測
  - リンク予測
  - ノード属性の予測
- グラフ群の識別
  - グラフカーネル

→ グラフデータのクラスタリング
- グラフ上のノード群のクラスタリング
  - スペクトルクラスタリング
  - コミュニティ抽出
- グラフ群のクラスタリング
  - グラフカーネル

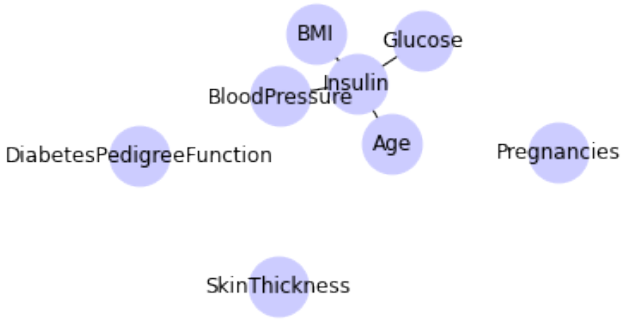
グラフデータの作成：相関行列、類似度行列から作る

例：糖尿病データセットをグラフ化

相関行列

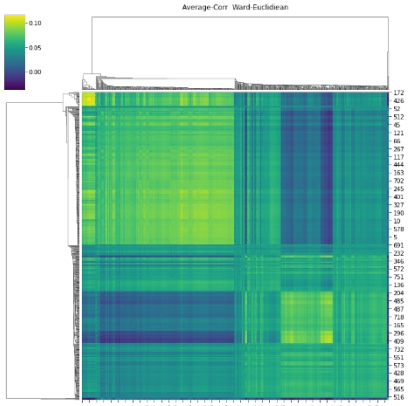


閾値

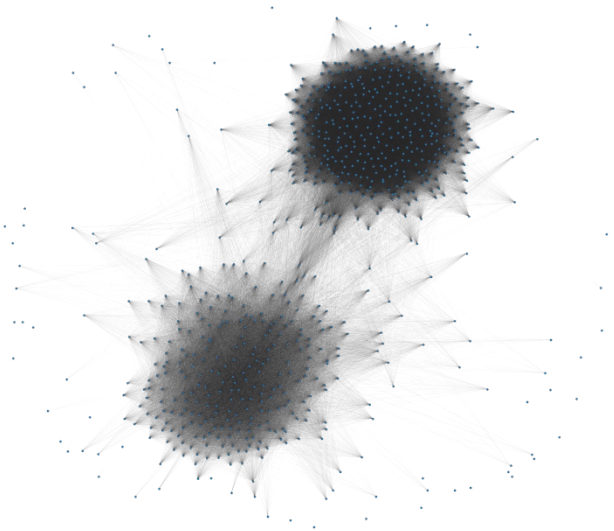


変数間のつながりのグラフ

類似度行列



閾値



各サンプルの8次元のベクトルの内積（類似度）の全組み合わせ

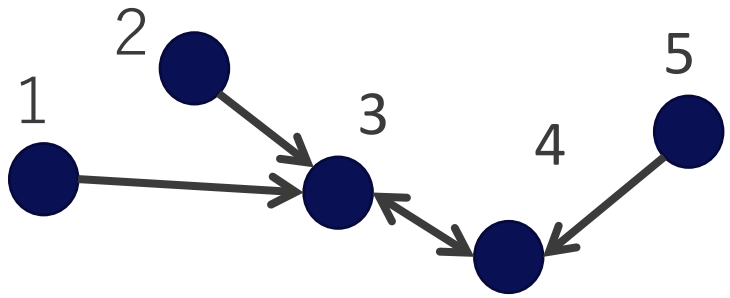
サンプル間のつながりのグラフ

グラフの定式化

隣接行列

ノードiとノードjに
エッジがあれば１なければ０

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



隣接行列をこのように定義すると到達可能かどうかを
行列の積で表すことができる

2回の移動で辿りつけるかどうかの0/1変数$b_{i,j}$

$$b_{i,j} = \sum_k a_{i,k} a_{k,j}$$

$b_{i,j}$を要素に持つ行列$\boldsymbol{B}$

$$\boldsymbol{B} = \boldsymbol{A}^2$$

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

グラフデータに対する処理：まとめ

グラフデータの前処理、グラフの構築

- グラフの構築法：相関行列、類似度行列
- グラフの表現方法：隣接行列、接続行列

→ グラフ上の重要な統計量

- 平均次数、次数分布、スケールフリー性など

→ グラフに関する予測

- グラフ上の予測
  - リンク予測
  - ノード属性の予測
- グラフ群の識別
  - グラフカーネル・グラフ類似度

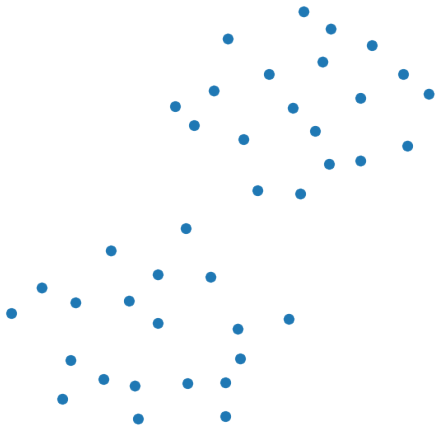
→ グラフデータのクラスタリング

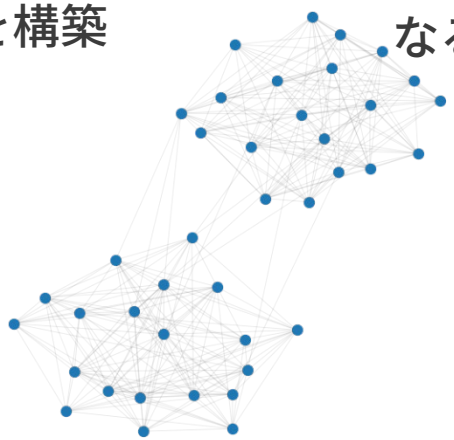
- グラフ上のノード群のクラスタリング
  - グラフカット・スペクトルクラスタリング
  - コミュニティ抽出
- グラフ群のクラスタリング
  - グラフカーネル・グラフ類似度

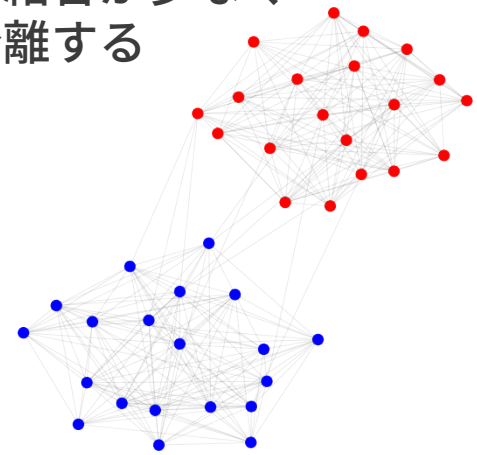
スペクトラルクラスタリング

類似度行列
→グラフを構築

クラスタリング
クラスタ間の結合が少なく
なるように分離する







scikit learn

Home Installation Documentation Examples

Google Custom Search

Previous Next Up

scikit-learn v0.20.3
Other versions

Please cite us if you use the software.

sklearn.cluster.SpectralClustering
Examples using sklearn.cluster.SpectralClustering

sklearn.cluster.SpectralClustering

class sklearn.cluster. SpectralClustering (*n_clusters=8, eigen_solver=None, random_state=None, n_init=10, gamma=1.0, affinity='rbf', n_neighbors=10, eigen_tol=0.0, assign_labels='kmeans', degree=3, coef0=1, kernel_params=None, n_jobs=None*) [source]

Apply clustering to a projection to the normalized laplacian.

In practice Spectral Clustering is very useful when the structure of the individual clusters is highly non-convex or more generally when a measure of the center and spread of the cluster is not a suitable description of the complete cluster. For instance when clusters are nested circles on the 2D plan.

If affinity is the adjacency matrix of a graph, this method can be used to find normalized graph cuts.

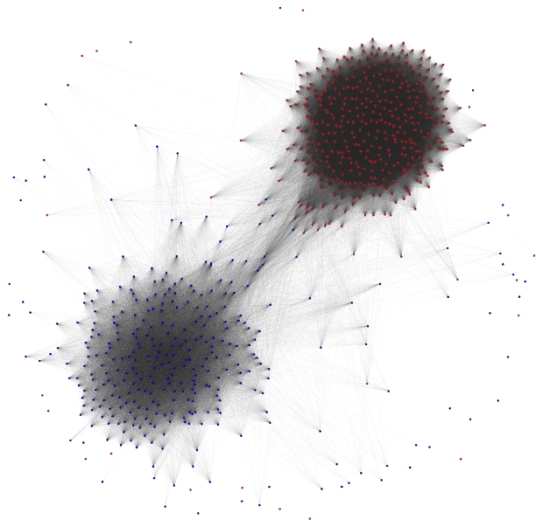
When calling `fit`, an affinity matrix is constructed using either kernel function such the Gaussian (aka RBF) kernel of the euclidean distanced `d(X, X)`:

```
np.exp(-gamma * d(X,X) ** 2)
```

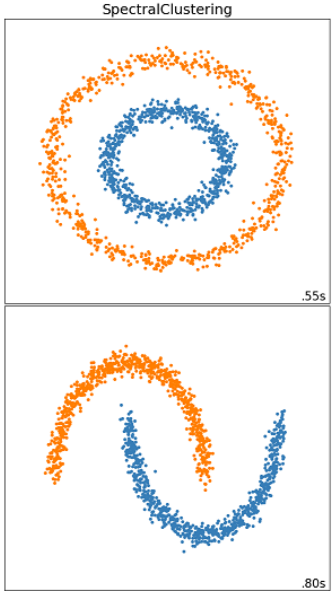
or a k-nearest neighbors connectivity matrix.

Alternatively, using `precomputed`, a user-provided affinity matrix can be used.

Fork me on GitHub

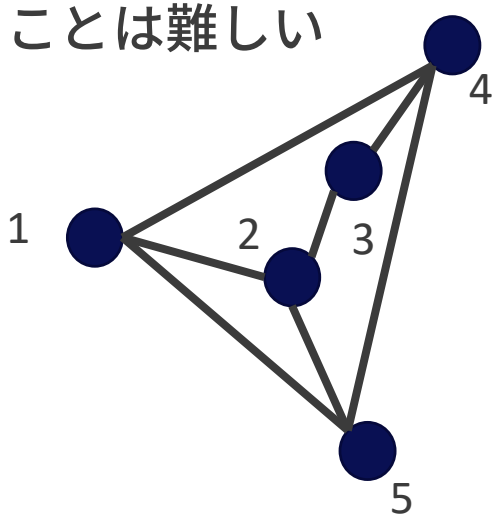


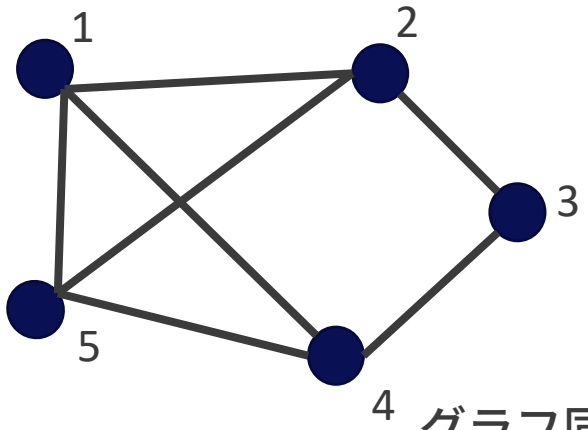
SpectralClustering



グラフの識別・クラスタリング：グラフカーネル

前提：グラフに一意のIDがない場合、グラフが同じかどうかを判定することは難しい



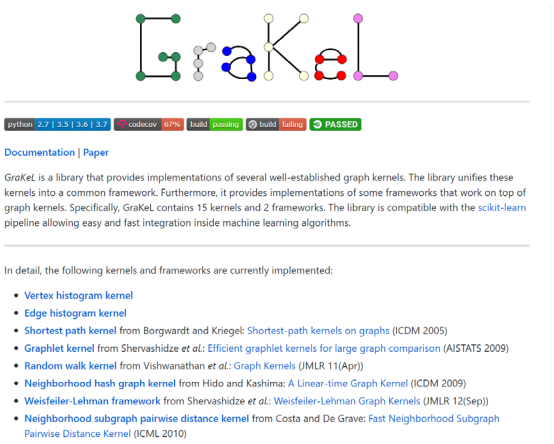


グラフ同型性判定問題
→NP-hard（実用的ではない）

グラフの類似度を定義：グラフカーネル

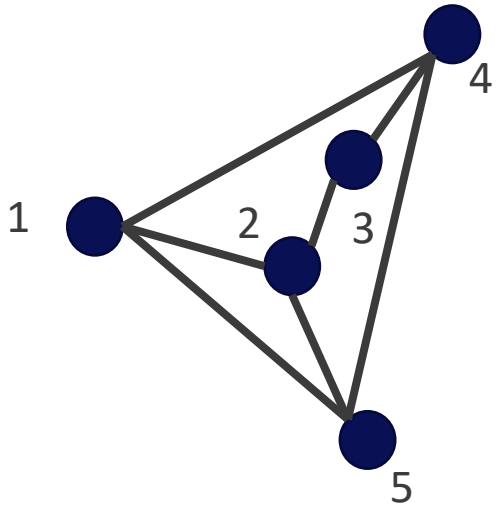
さまざまな手法がある

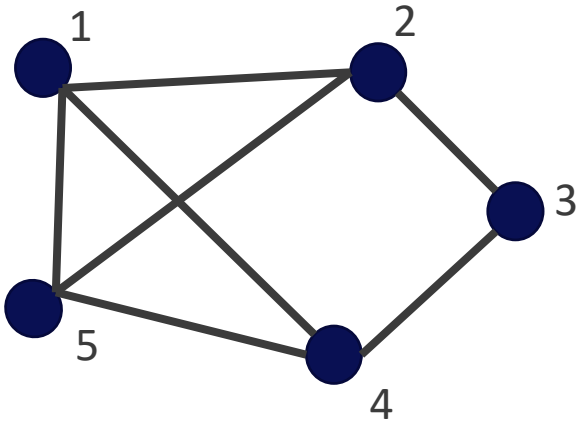
- shortest-path kernel
- random walk kernel
- Weisfeiler-Lehman graph kernels



グラフの識別・クラスタリング：グラフカーネル

ノードの対応がとれている場合によく使われる方法

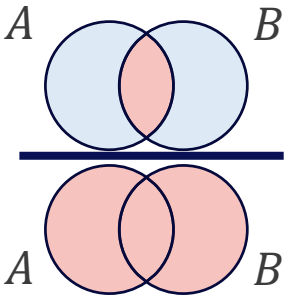




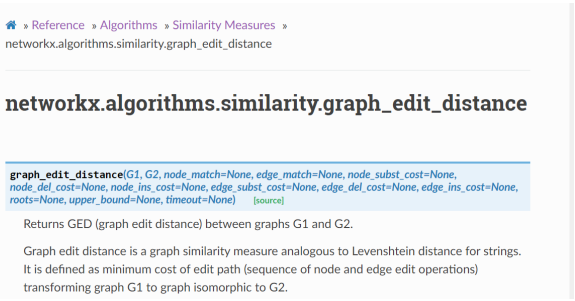
グラフ編集距離：エッジの追加・削除を何回行ったか

エッジ集合間のJaccard係数：

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} =$$



二つのグラフのエッジの集合をA,Bとする



グラフデータに対する処理：まとめ

グラフデータの前処理、グラフの構築

- グラフの構築法：相関行列、類似度行列
- グラフの表現方法：隣接行列、接続行列

→ グラフ上の重要な統計量

- 平均次数、次数分布、スケールフリー性など

→ グラフに関する予測

- グラフ上の予測
  - リンク予測
  - ノード属性の予測
- グラフ群の識別
  - グラフカーネル・グラフ類似度

→ グラフデータのクラスタリング

- グラフ上のノード群のクラスタリング
  - グラフカット・スペクトルクラスタリング
  - コミュニティ抽出
- グラフ群のクラスタリング
  - グラフカーネル・グラフ類似度

まとめ

- 様々なデータがあるので、それに応じた対応をする必要がある
  - テーブルデータ・行列データ・系列データ・グラフデータ

- 元が同じデータでもどういった見方で分析するかは分析者の腕が試される

- それぞれのデータに応じて様々な手法があるので、大まかには知っておくと便利