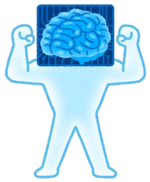


[E資格] 機械学習入門

Study AI

本講義の目的

①機械学習の**基本的な手法**を理解し実装する

- 線形回帰
- ロジスティック回帰
- 主成分分析
- K平均法など

本講義の目的

①機械学習の基本的な手法を理解し実装する

②**機械学習モデリング**の流れを理解

機械学習モデリングプロセス(大枠はDLでも同じ)

1. 問題設定

どのような課題を
機械学習に解決させるか (※)

2. データ選定

どのようなデータを使うか

3. データの前処理

モデルに学習させられるように
データを変換

4. 機械学習
モデルの選定

どの機械学習モデルを
利用するか

5. モデルの学習
(パラメータ推定)

パラメータの決め方

6. モデルの評価

ハイパーパラメータの選定
モデル精度を測る

※問題設定時に**必ずしも機械学習を使う必要はない**が本講義は機械学習を使うことを前提として進める

機械学習モデリングプロセス(大枠はDLでも同じ)

1. 問題設定

どのような課題を
機械学習に解決させるか (※)

2. データ選定

どのようなデータを使うか

3. データの前処理

モデルに学習させられるように
データを変換

4. 機械学習モデルの選定

どの機械学習モデルを
利用するか

…モデルの学習 …ラメータ推定）

…ラメー…

6. モデルの評価

ハイパーパラメータの選定
モデル精度を測る

線形回帰

ロジ回帰

SVM

深層学習

PCA

ディベロップターム
で取り扱い

※問題設定時に必ずしも機械学習を使う… …を使うことを前提として進める

本日講義で扱う内容

学習種類	タスク	機械学習モデル	パラメータの推定 (アルゴリズム)	モデル 選択・評価
教師**あり**学習	予測	線形回帰・非線形回帰	最小2乗法 (最尤法)	ホールドアウト法 交差検証法
	分類	サポートベクターマシン	マージン最大化	
		ロジスティック回帰	最尤法 + 確率的勾配降下法	
		最近傍法・K-近傍法		
教師**なし**学習	クラスタリング	K平均クラスタリング		無し
		主成分分析	固有値分解	

※シラバスに含まれる手法のみ

線形回帰とロジスティック回帰でモデリングプロセスを体験

学習種類	タスク	機械学習モデル	パラメータの推定 (アルゴリズム)	モデル 選択・評価
教師**あり**学習	予測	線形回帰・非線形回帰	最小2乗法 (最尤法)	ホールドアウト法 交差検証法
	分類	サポートベクターマシン	マージン最大化	
		ロジスティック回帰	最尤法 + 確率的勾配降下法	
		最近傍法・K-近傍法		
教師**なし**学習	クラスタリング	K平均クラスタリング		無し
		主成分分析	固有値分解	

※シラバスに含まれる手法のみ

数式は省きエッセンスのみ紹介

学習種類	タスク	機械学習モデル	パラメータの推定 (アルゴリズム)	モデル 選択・評価
教師**あり**学習	予測	線形回帰・非線形回帰	最小2乗法 (最尤法)	ホールドアウト法 交差検証法
	分類	サポートベクターマシン	マージン最大化	
		ロジスティック回帰	最尤法 + 確率的勾配降下法	
		最近傍法・K-近傍法		
教師**なし**学習	クラスタリング	K平均クラスタリング		無し
		主成分分析	固有値分解	

※シラバスに含まれる手法のみ

講義の構成

モデリングプロセス (15)	線形回帰 Lecture (30)	休憩 (10)	線形回帰 Programming (20)	非線形回帰 Lecture + programming (30)	休憩 (10)
ロジスティック回帰Lecture (30)	ロジスティック回帰 Programming (20)		SVM + 主成分分析 (20)	Numpy programming 紹介 (20)	
演習問題解説 (30)	DNN Lecture (20)		DNN + Lecture (30)	DNN + programming (20)	

多クラス分類について

※誤差逆伝播は次講義から

初めに

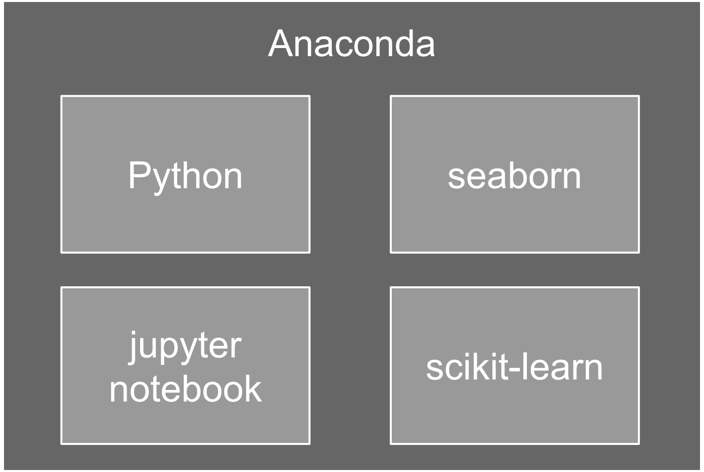
ハンズオン環境

・みなさんのPCを利用してプログラミングを行います。

・環境は事前に構築して来ていただいています (トラブルが発生した場合は、サポートスタッフまでお声がけください)

参加者のみなさま

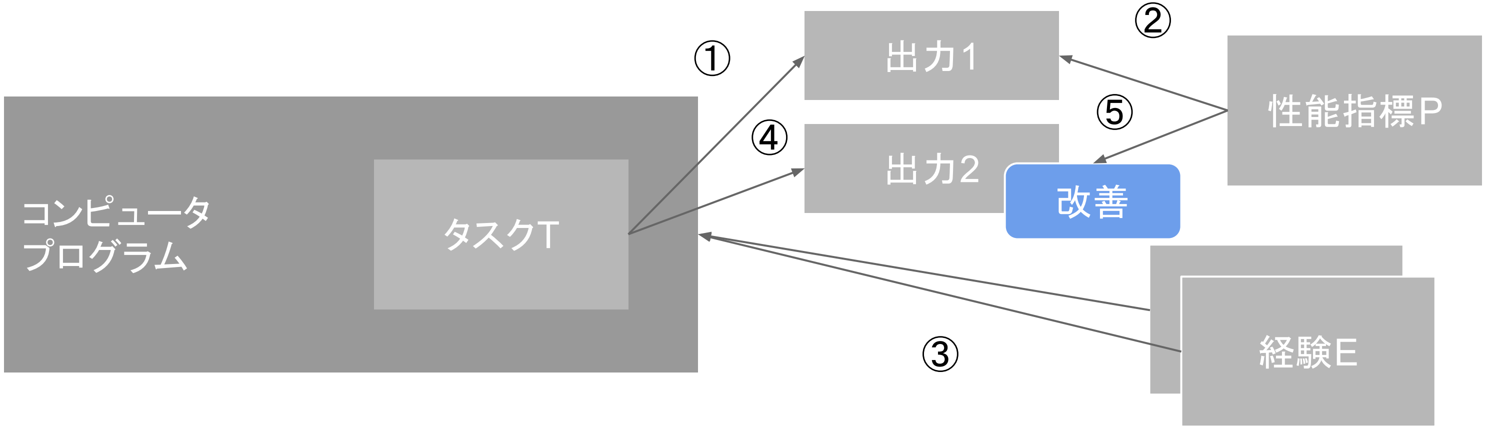
みなさんの
コンピュータ



本日の講義に必要
なライブラリ

ルールベースモデルと機械学習モデル

- 機械学習
  - コンピュータプログラムを経験によって自動的に改善していく方法について研究
  - コンピュータプログラムは、**タスクT(アプリケーションにさせたいこと)**を**性能指標P**で測定し、その性能が**経験E(データ)**により改善される場合、**タスクTおよび性能指標Pに関して経験Eから学習する**と言われている (トム・ミッチェル 1997)
  - 学習用データセット(経験E)を利用して訓練した後に、未知のデータについて正確に予測・分類できるアルゴリズムの能力をいう



線形回帰モデル

1. データの形式
2. 線形回帰モデル
3. モデル(パラメータ)の推定
4. モデルの評価
5. ハンズオン

回帰問題

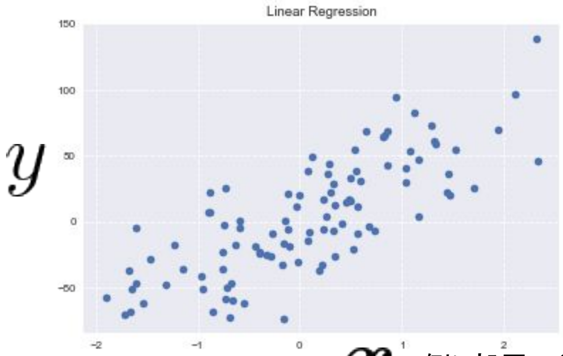
ある入力(数値)から出力(連続値)を予測する問題

- 回帰で扱うデータ
  - 入力はm次元のベクトル (m=1の場合はスカラ)
  - 入力ベクトルの各要素は説明変数 (または特徴量)と呼ぶ
  - 出力はスカラー値 (目的変数と呼ぶ)

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_m)^T \in \mathbb{R}^m \qquad y \in \mathbb{R}^1$$

- 線形回帰はデータを直線で、非線形回帰は曲線で近似したもの

例) 住宅価格



$\boldsymbol{x}$ 例) 部屋の数

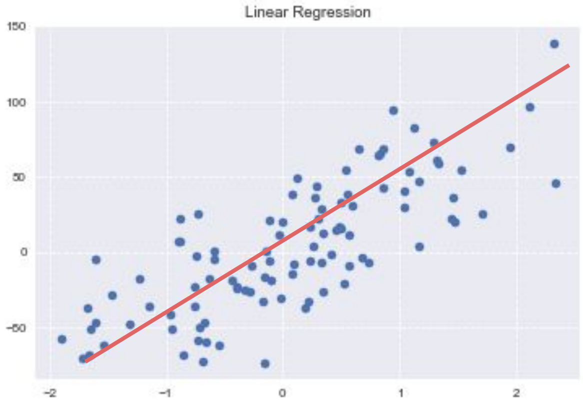
例) 住宅データ

線形回帰

回帰問題を解くための機械学習モデル

- 線形回帰モデル
  - 教師あり学習(正解付きデータから学習)
  - 入力と**m次元パラメータ**の**線形結合**を出力するモデル

$$\boldsymbol{w} = (w_1, w_2, \cdots, w_m)^T \in \mathbb{R}^m$$



$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_{j=1}^{m} w_j x_j + b$$

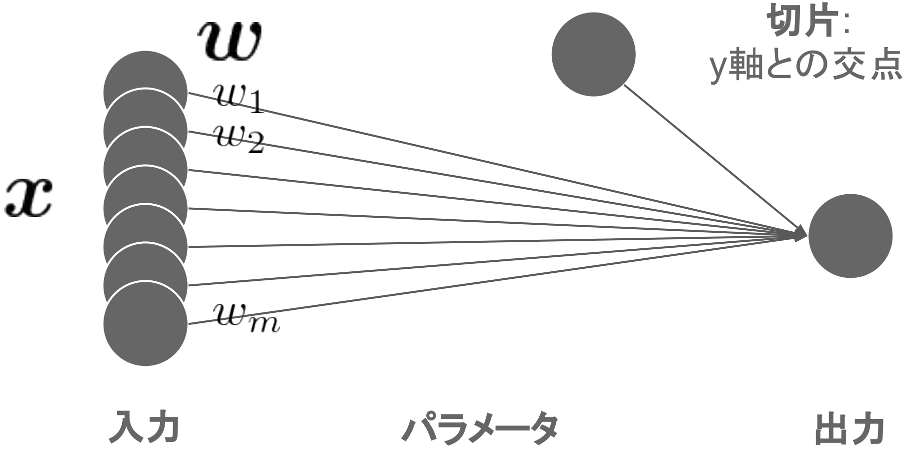
線形結合(パラメータと入力ベクトルの掛け算)

- 線形結合
  - 入力ベクトルと未知のパラメータの各要素を掛け算し足し合わせたもの
  - 入力ベクトルとの線形結合に加え、切片も足し合わせる
  - (入力のベクトルが多次元でも)出力は1次元(スカラ)となる

線形結合

$$\hat{y} = \underline{\boldsymbol{w}}^T \boldsymbol{x} + b = \sum_{j=1}^{m} \underline{w_j} x_j + b$$

未知パラメータ



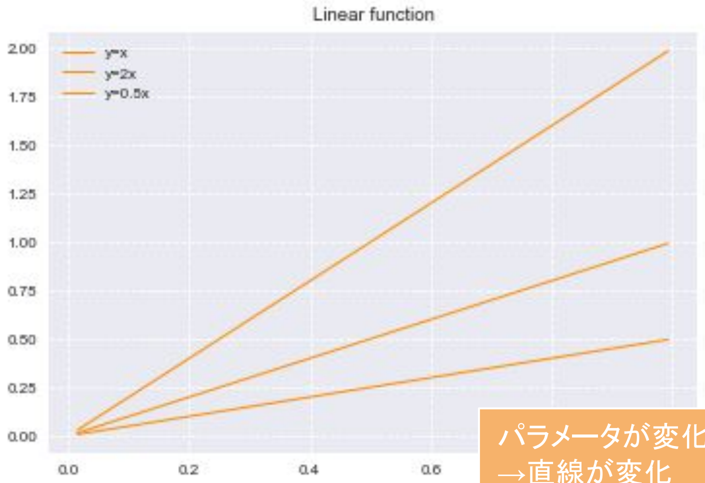
線形回帰モデルのパラメータ(推定すべき未知のもの)

- モデルのパラメータ
  - 特徴量が予測値に対してどのように影響を与えるかを決定する重みの集合
  - 正の(負の)重みをつける場合、その特徴量の値を増加させると、予測の値が増加 (減少)
  - 重みが大きければ(0であれば)、その特徴量は予測に大きな影響力を持つ (全く影響しない)

線形結合は直線・面を表現

$$\hat{y} = \boldsymbol{w}^T \boldsymbol{x} + b = \sum_{j=1}^{m} w_j x_j + b$$

未知パラメータは
最小二乗法により推定



パラメータが変化
→直線が変化
→ 全ての線を表現可能

線形単(m=1)回帰モデル(データへの仮定)

データは回帰直線に**誤差**が加わり観測されていると仮定

数式

$$y = w_0 + w_1 x_1 + \varepsilon$$

目的変数

切片

回帰係数

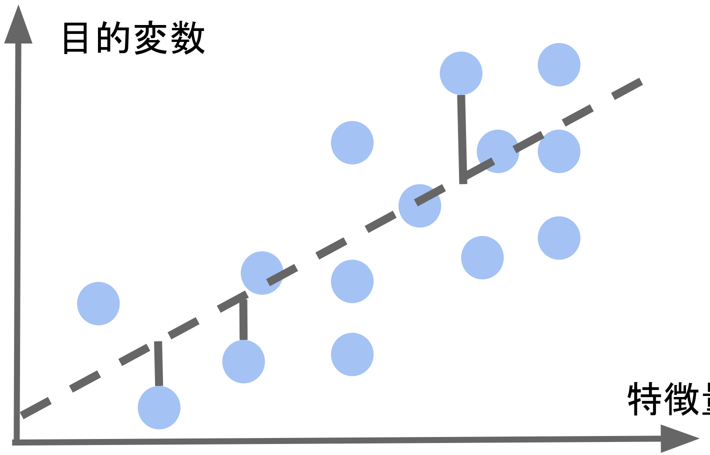
説明変数

誤差

既知:入力データ

未知:学習で決める

幾何学的な意味



線形回帰モデル(行列表現)

連立方程式は計算の簡略化のため行列で表現

連立方程式	行列表現

$$y_1 = w_0 + w_1 x_1 + \varepsilon_1$$
$$y_2 = w_0 + w_1 x_2 + \varepsilon_2$$
$$y_n = w_0 + w_1 x_n + \varepsilon_n$$

$$\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\varepsilon}$$

$$X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)^T$$

$$\boldsymbol{y} = (y_1, y_2, \cdots, y_n)^T$$

$$\boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{im})^T$$

$$\boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n)^T$$

$$\boldsymbol{w} = (w_1, w_2, \cdots, w_m)^T$$

未知パラメータは
最小二乗法により推定

線形重(m=多次元)回帰モデル

数式

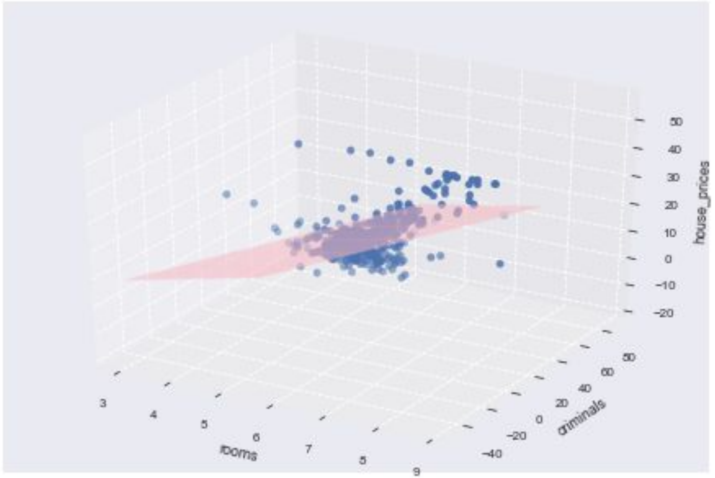
$$y = w_0 + w_1 x_1 + w_2 x_2 + \varepsilon$$

目的変数
切片
回帰係数
説明変数
回帰係数
説明変数
誤差

既知：入力データ

未知：学習で決める

幾何学的な意味



データは回帰**直面**に**誤差**が加わり観測されていると仮定

線形重(m=多次元)回帰モデル

連立方程式

$$
\begin{aligned}
y_1 &= w_0 + w_1 x_{11} + w_2 x_{12} + \cdots + w_m x_{1m} + \varepsilon_1 \\
y_2 &= w_0 + w_1 x_{21} + w_2 x_{22} + \cdots + w_m x_{2m} + \varepsilon_2 \\
\\
y_n &= w_0 + w_1 x_{n1} + w_2 x_{n2} + \cdots + w_m x_{nm} + \varepsilon_n
\end{aligned}
$$

行列表現

$$X = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n)^T \qquad \boldsymbol{x}_i = (x_{i1}, x_{i2}, \cdots, x_{im})^T$$

$$\boldsymbol{y} = X\boldsymbol{w} + \boldsymbol{\varepsilon}$$

$$\boldsymbol{y} = (y_1, y_2, \cdots, y_n)^T \qquad \boldsymbol{\varepsilon} = (\varepsilon_1, \varepsilon_2, \cdots, \varepsilon_n)^T$$

未知パラメータは
最小二乗法により推定

$$\boldsymbol{w} = (w_1, w_2, \cdots, w_m)^T$$

データの分割

- データを学習用と検証用データに分割
  - 学習用データでモデルを学習し、検証用データでモデルを検証
  - 学習データで検証した場合、一般的に当てはまりは良くなる
  - 学習データで検証した場合、**汎化性能**を測るのが困難

全てのデータ

$$\{(\boldsymbol{x}_i, y_i); i = 1, \cdots, n\}$$

モデルを学習

①学習データ

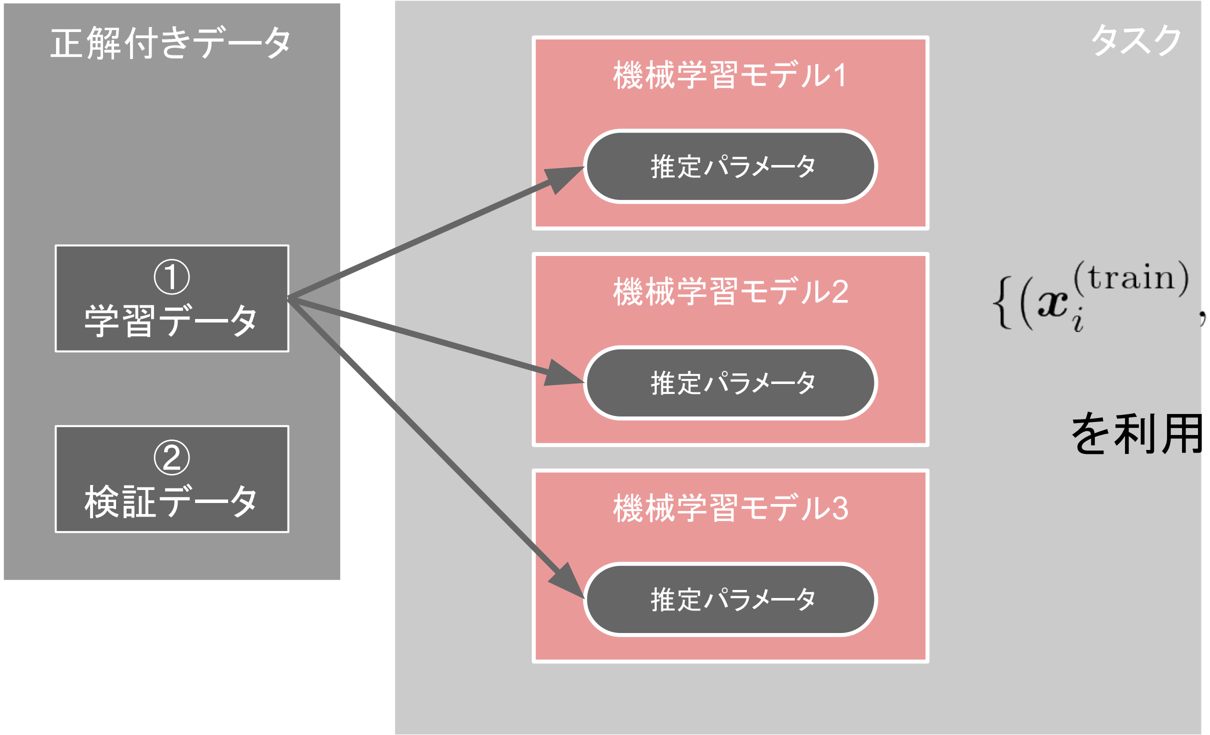
$$\{(\boldsymbol{x}_i^{(\mathrm{train})}, y_i^{(\mathrm{train})}); i = 1, \cdots, n_{\mathrm{train}}\}$$

モデルの精度を測定

②検証データ

$$\{(\boldsymbol{x}_i^{(\mathrm{test})}, y_i^{(\mathrm{test})}); i = 1, \cdots, n_{\mathrm{test}}\}$$

機械学習モデルの学習

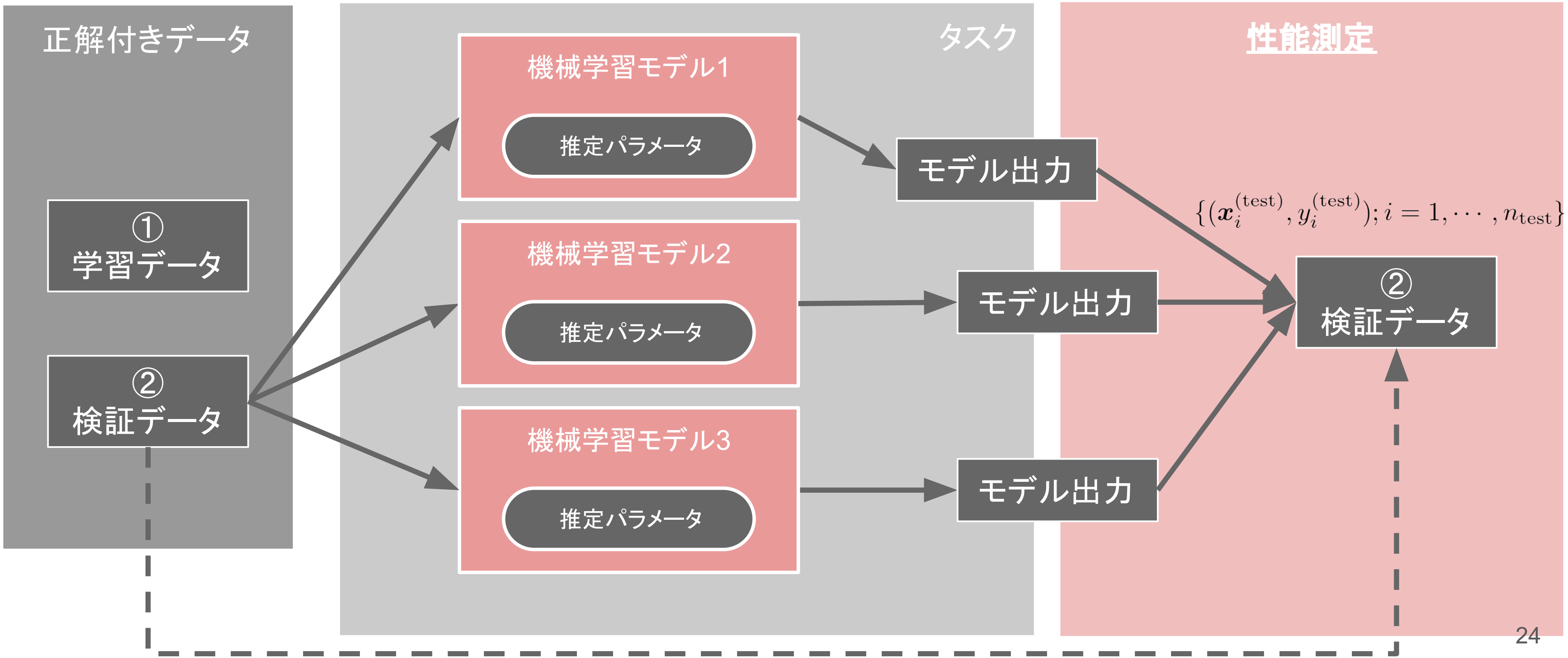


$$\{(\boldsymbol{x}_i^{(\mathrm{train})}, y_i^{(\mathrm{train})}); i = 1, \cdots, n_{\mathrm{train}}\}$$

を利用して推定パラメータを学習

モデルの評価と選択(汎化性能を測るため検証データを利用)

※学習データへの当てはまりでは無い



質疑応答

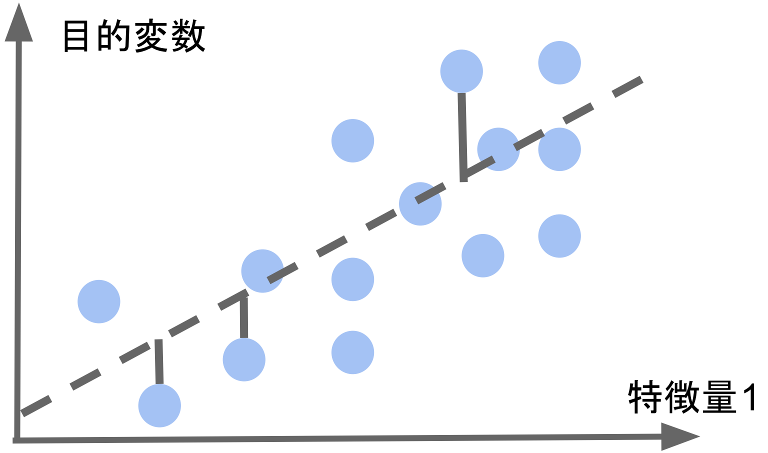
- データ分割
  - 学習用と検証用にデータを分割
  - 次スライド以降では分割したそれぞれのデータを用いて、線形回帰モデルの **学習**と**検証**の具体的な方法について説明

- ご質問がある方は只今の時間にご質問ください

パラメータの推定

- **平均二乗誤差** (Mean Squared Error)
  - データとモデル出力の二乗誤差
  - 小さいほど直線とデータの距離が近い

$$\mathrm{MSE}_{\mathrm{train}} = \frac{1}{n_{\mathrm{train}}} \sum_{i=1}^{n_{\mathrm{train}}} (\hat{y}_i^{(\mathrm{train})} - y_i^{(\mathrm{train})})^2$$



- 最小二乗法
  - 学習データの **平均二乗誤差** を最小とするパラメータを探索
  - 学習データの **平均二乗誤差** の最小化は、その **勾配が0になる点を求めれ** ば良い

パラメータの推定

平均二乗誤差を最小にするパラメータを求めるためには、
その勾配が0になる点を求めれば良い

$$\hat{\boldsymbol{w}} = \arg\min_{\boldsymbol{w}\in\mathbb{R}^m} \mathrm{MSE}_{\mathrm{train}}$$

MSEを最小にするようなw(m次元)

$$\frac{\partial}{\partial \boldsymbol{w}} \mathrm{MSE}_{\mathrm{train}} = 0$$

MSEをwに関して微分したものが 0と
なるwの点を求める

※1. 計画行列とバイアスをパラメータに含める話

パラメータの推定

行列に対して微分の操作をすれば推定量を得られる

$$\Rightarrow \frac{\partial}{\partial \boldsymbol{w}}\left\{\frac{1}{n_{\text{train}}}\sum_{i=1}^{n_{\text{train}}}(\hat{y}_i^{(\text{train})}-y_i^{(\text{train})})^2\right\}=0$$

平均二乗誤差(残差平方和)を微分

$$\Rightarrow \frac{1}{n_{\text{train}}}\frac{\partial}{\partial \boldsymbol{w}}\left\{(X^{(\text{train})}\boldsymbol{w}-\boldsymbol{y}^{(\text{train})})^T(X^{(\text{train})}\boldsymbol{w}-\boldsymbol{y}^{(\text{train})})\right\}$$

行列表現

$$\Rightarrow \frac{1}{n_{\text{train}}}\frac{\partial}{\partial \boldsymbol{w}}\left\{\boldsymbol{w}^T X^{(\text{train})T}X^{(\text{train})}\boldsymbol{w}-2\boldsymbol{w}^T X^{(\text{train})T}\boldsymbol{y}^{(\text{train})}+\boldsymbol{y}^{(\text{train})T}\boldsymbol{y}^{(\text{train})}\right\}$$

{ } 展開

$$\Rightarrow 2X^{(\text{train})T}X^{(\text{train})}\boldsymbol{w}-2X^{(\text{train})T}\boldsymbol{y}^{(\text{train})}=0$$

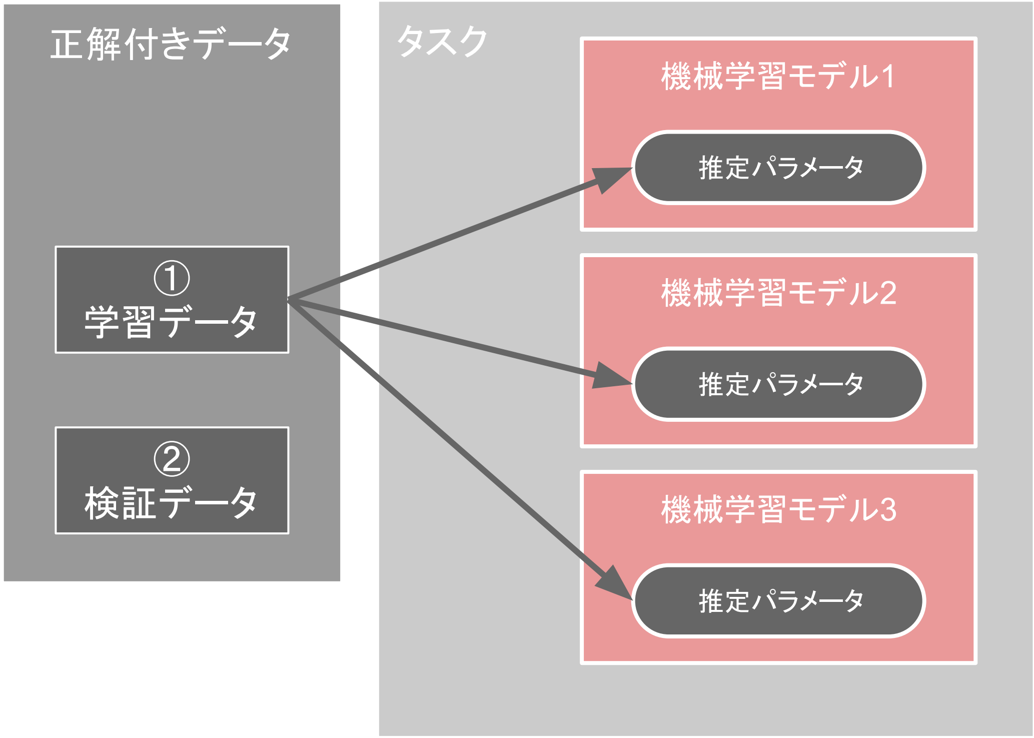
行列微分計算

$$\Rightarrow \hat{\boldsymbol{w}}=(X^{(\text{train})T}X^{(\text{train})})^{-1}X^{(\text{train})T}\boldsymbol{y}^{(\text{train})}$$

回帰係数

パラメータの推定

$$\hat{\boldsymbol{w}} = (X^{(\mathrm{train})T} X^{(\mathrm{train})})^{-1} X^{(\mathrm{train})T} \boldsymbol{y}^{(\mathrm{train})}$$



[注意]

本講義では「最小2乗法」を用いてパラメータの推定を行ったが、「**最尤法**」を用いてパラメータの推定を行うことが出来る。最尤法で推定した回帰係数の結果は最小 2乗法のそれと等しくなる。

パラメータの推定

※必要に応じて多重共線性の話を･･･

・パラメータの推定値を元に予測値は以下になる。

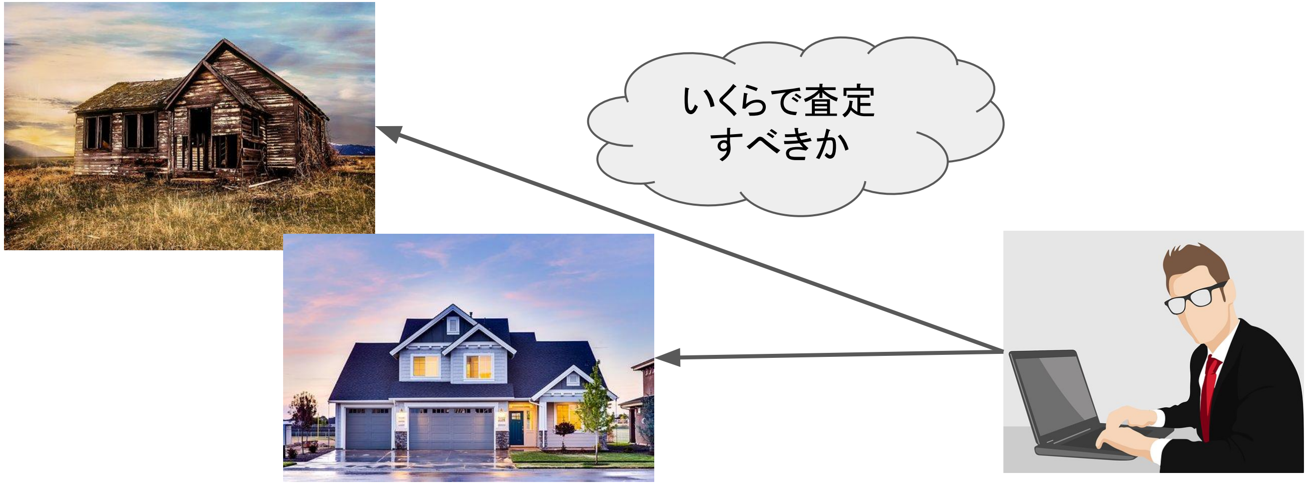
$$\hat{\boldsymbol{y}} = X(X^{(train)T}X^{(train)})^{-1}X^{(train)T}\boldsymbol{y}$$

trai
n

・データへの当てはまりの良さは決定係数を用いて評価 (本日は割愛)

ハンズオン (線形回帰モデル)

- ボストン住宅データセット分析して線形回帰モデルを作成する
- 適切な査定結果が必要
  - 高すぎても安すぎても会社に損害がある
- ルールベースではない方法で予測モデルを作成する



ボストン住宅価格データ

Boston house-prices (ボストン市の住宅価格)

・レコード数 (データをとった住宅の数)：506

・カラム数 (価格・部屋の数など)：14

・データセットの詳細：UCI Machine Learning Repository: Housing Data Set

線形回帰分析を試すデータとしては最適なデータセット！

項番	変数	記号名	備考
1	犯罪発生率	CRIM	(人口単位)
2	25000平方フィート以上の住宅区間の割合	ZN	
3	非小売業の土地面積の割合	INDUS	（人口単位）
4	チャールズ川沿いかどうかフラグ	CHAS	（川沿いなら1、そうでなければ0）
5	窒素感化物の濃度	NOX	（pphm単位）
6	一戸あたりの平均部屋数	RM	2,3など (単位は部屋)
7	1940年よりも前に建てられた家屋の割合	AGE	
8	ボストンの主な5つの雇用圏までの重み付き距離	DIS	
9	幹線道路へのアクセス指数	RAD	
10	10000ドルあたりの所得税率	TAX	
11	教師あたりの生徒の数	PTRATIO	（人口単位）
12	1000(Bk - 0.63)^2として計算される量	B	（Bkはアフリカ系アメリカ人居住者の割合（人口単位））
13	低所得者の割合	LSTAT	
14	住宅価格	PRICE	中央値（単位は1000ドル）

ハンズオン (線形回帰モデル)

jupyter notebookを起動

Daeju — -bash — 80×24

```
Last login: Sat Dec  9 16:09:48 on ttys000
kaneoohashira-no-Mac-mini:~ Daeju$ jupyter notebook
```

Daeju — jupyter-notebook — 80×24

```
Last login: Sat Dec  9 16:09:48 on ttys000
kaneoohashira-no-Mac-mini:~ Daeju$ jupyter notebook
[I 16:36:35.398 NotebookApp] JupyterLab alpha preview extension loaded from /ana
conda2/lib/python2.7/site-packages/jupyterlab
JupyterLab v0.27.0
Known labextensions:
[I 16:36:35.401 NotebookApp] Running the core application with no additional ext
ensions or settings
[I 16:36:35.410 NotebookApp] Serving notebooks from local directory: /Users/Daej
u
[I 16:36:35.410 NotebookApp] 0 active kernels
[I 16:36:35.411 NotebookApp] The Jupyter Notebook is running at: http://localhos
t:8888/?token=cfbfd4b78a5aacd91d016b99bfa5b67eba3ef664be5ade1a
[I 16:36:35.411 NotebookApp] Use Control-C to stop this server and shut down all
 kernels (twice to skip confirmation).
[C 16:36:35.412 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=cfbfd4b78a5aacd91d016b99bfa5b67eba3ef664be5
ade1a
[I 16:36:35.998 NotebookApp] Accepting one-time-token-authenticated connection f
rom ::1
```

ハンズオン (線形回帰モデル)

jupyter notebookを起動

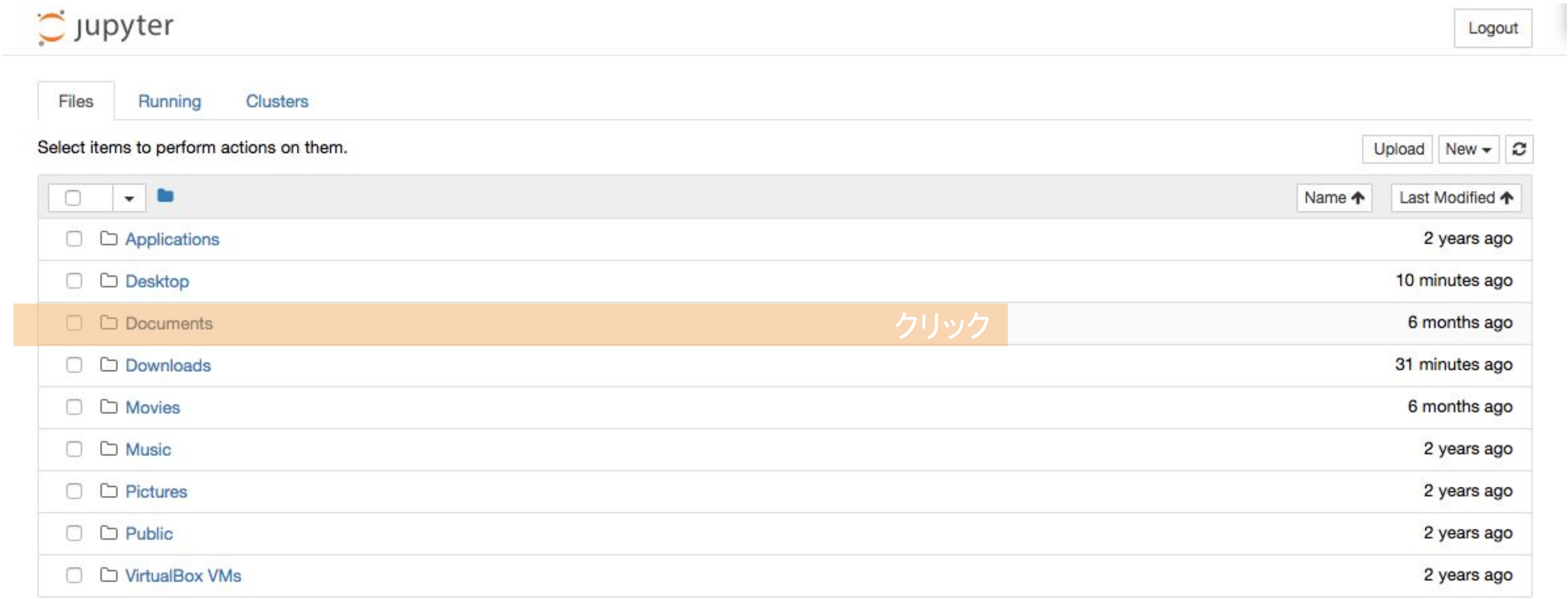
```
Daeju — jupyter-notebook — 80×24
Last login: Sat Dec  9 16:09:48 on ttys000
kaneoohashira-no-Mac-mini:~ Daeju$ jupyter notebook
[I 16:36:35.398 NotebookApp] JupyterLab alpha preview extension loaded from /ana
conda2/lib/python2.7/site-packages/jupyterlab
JupyterLab v0.27.0
Known labextensions:
[I 16:36:35.401 NotebookApp] Running the core application with no additional ext
ensions or settings
[I 16:36:35.410 NotebookApp] Serving notebooks from local directory: /Users/Daej
u
[I 16:36:35.410 NotebookApp] 0 active kernels
[I 16:36:35.411 NotebookApp] The Jupyter Notebook is running at: http://localhos
t:8888/?token=cfbfd4b78a5aacd91d016b99bfa5b67eba3ef664be5ade1a
[I 16:36:35.411 NotebookApp] Use Control-C to stop this server and shut down all
 kernels (twice to skip confirmation).
[C 16:36:35.412 NotebookApp]

    Copy/paste this URL into your browser when you connect for the first time,
    to login with a token:
        http://localhost:8888/?token=cfbfd4b78a5aacd91d016b99bfa5b67eba3ef664be5
ade1a
[I 16:36:35.998 NotebookApp] Accepting one-time-token-authenticated connection f
rom ::1
```

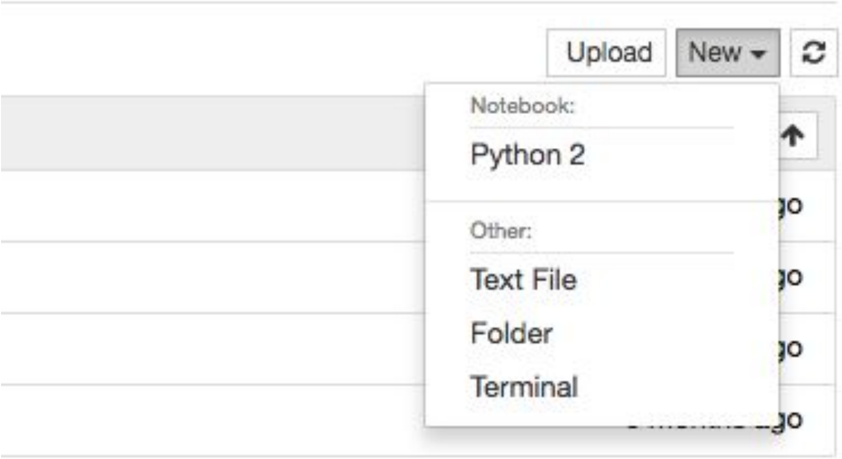
ブラウザを開きここにアクセス

ハンズオン (線形回帰モデル)

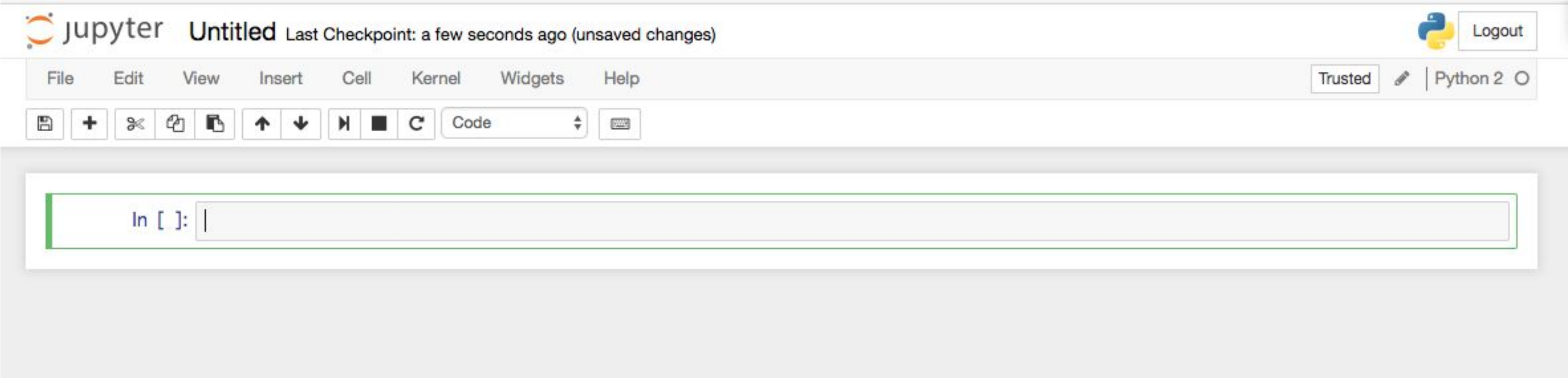
ハンズオン (線形回帰モデル)



ハンズオン (線形回帰モデル)



ハンズオン (線形回帰モデル)



ハンズオン (線形回帰モデル)

Scikit-learn: PythonのOpen-Sourceライブラリ

特徴：様々な回帰・分類・クラスタリングアルゴリズムが実装されており、 Pythonの数値計算ライブラリの NumPyとSciPyとやりとりするよう設計されている。(先ほどのPandasはデータ表示・解析に利用)

URL: http://scikit-learn.org/stable/http://scikit-learn.org/stable/

ハンズオン (線形回帰モデル)

利用するライブラリの説明

Pandas:
Excel のような2次元のテーブルを対象にしたデータ解析を支援する機能を提供するライブラリです。
DataFrame が pandasのメインとなるデータ構造で二次元のテーブルを表します。

Numpy:
値計算を効率的に行うための拡張モジュールである。効率的な数値計算を行うための型付きの多次元配列（例えばベクトルや行列などを表現できる）のサポートを Pythonに加えるとともに、それらを操作するための大規模な高水準の数学関数ライブラリを提供する。

統計処理、機械学習を行う際に共によく利用

ハンズオン (線形回帰モデル)

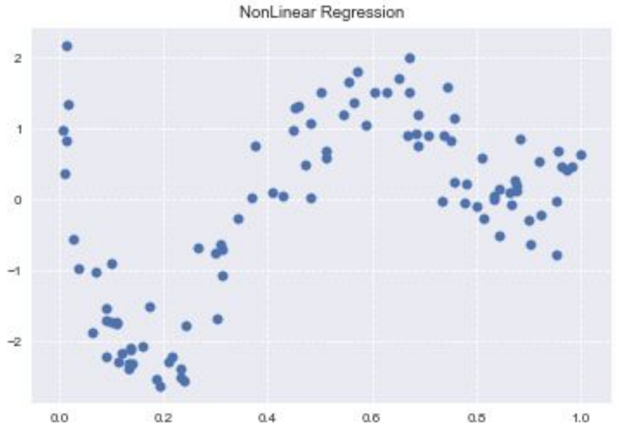
・以下GitHubにアクセスし、ご自身のPCでソースを実行してください。

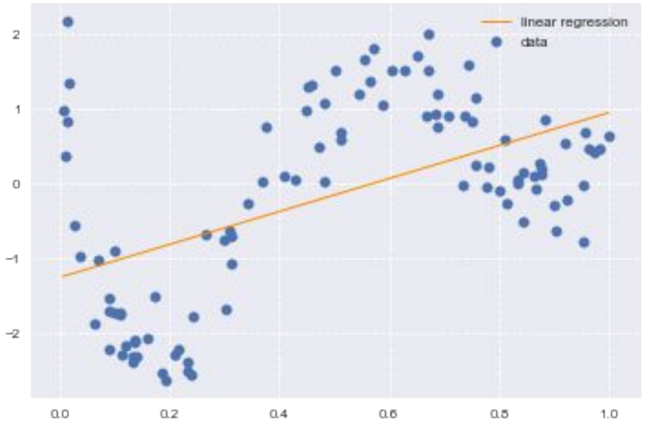
- https://github.com/studyaigit/StudyAI-M-L/tree/master/skl_MultivariateAnalysis
- /skl_MultivariateAnalysis/skl_regression.ipynb

非線形回帰モデル

- (これまで)**線形構造を内在する現象**に対して**線形回帰モデリング**を実施
  - データの構造を線形で捉えられる場合は限られる
  - 非線形な構造を捉える仕組みが必要

- 複雑な**非線形構造を内在する現象**に対して、**非線形回帰モデリング**を実施





非線形回帰モデル

- 基底展開法
  - 回帰関数として、基底関数と呼ばれる既知の非線形関数とパラメータベクトルの線型結合を使用
  - 未知パラメータは線形回帰モデルと同様に最小 2乗法や最尤法により推定

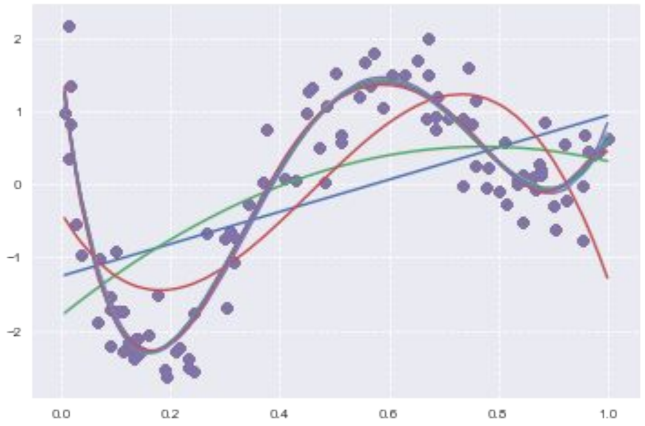
$$y_i = w_0 + \sum_{i=1}^{m} w_j \phi_j(\boldsymbol{x}_i) + \varepsilon_i$$

- よく使われる基底関数
  - 多項式関数
  - ガウス型基底関数
  - スプライン関数 / Bスプライン関数

非線形回帰モデル

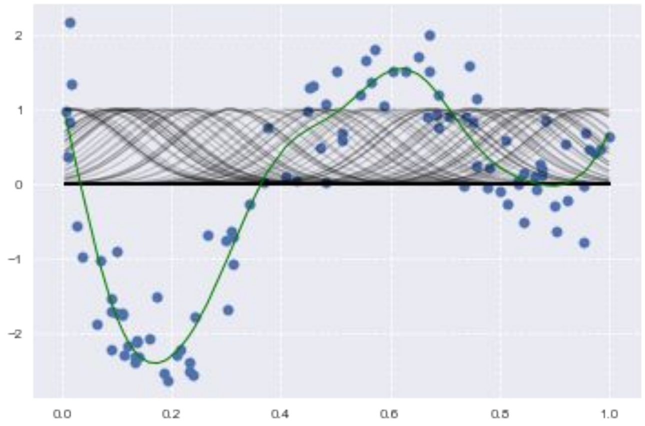
多項式 (1~9次)

$$\phi_j = x^j$$



ガウス型基底

$$\phi_j(\boldsymbol{x}) = \exp\left\{\frac{(\boldsymbol{x}-\boldsymbol{\mu}_j)^T(\boldsymbol{x}-\boldsymbol{\mu}_j)}{2h_j}\right\}$$



非線形回帰モデル

- 説明変数

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_n) \in \mathbb{R}^n$$

- 非線形関数ベクトル

$$\boldsymbol{\phi}_j(\boldsymbol{x}) = (\phi_j(x_1), \phi_j(x_2), \cdots, \phi_j(x_n))^T$$

- 非線形関数の計画行列

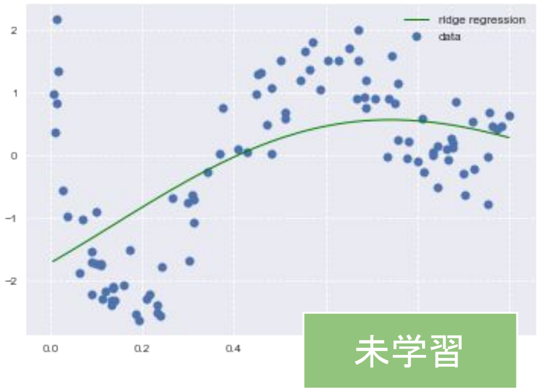
$$\Phi^{(train)} = (\boldsymbol{\phi}_1(\boldsymbol{x}), \boldsymbol{\phi}_2(\boldsymbol{x}), \cdots, \boldsymbol{\phi}_m(\boldsymbol{x}))^T$$

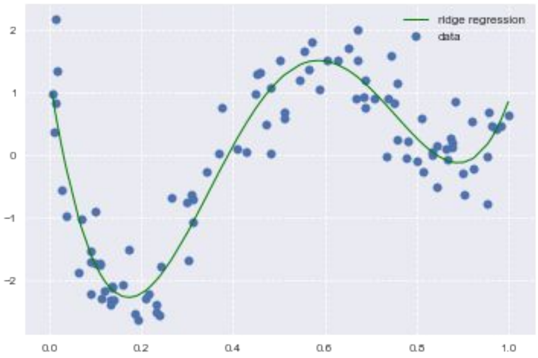
- 最尤法による予測値

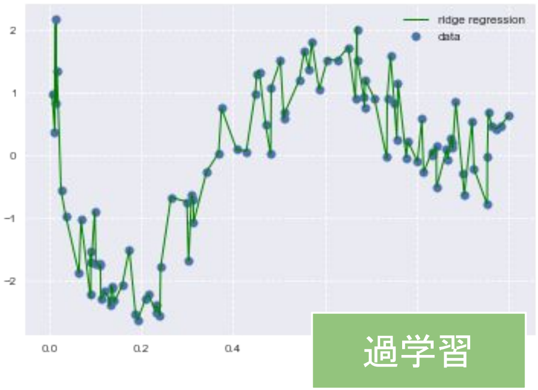
$$\hat{\boldsymbol{y}} = \Phi(\Phi^{(train)T}\Phi^{(train)})^{-1}\Phi^{(train)T}\boldsymbol{y}$$

正則化法

- 過学習(overfitting)と未学習(underfitting)
  - 学習データに対して、十分小さな誤差が得られない場合 →未学習
  - 小さな誤差は得られたけど、テスト集合誤差との差が大きい場合 →過学習
- 過学習は**正則化法**で回避(正則化しすぎると未学習)







正則化法

- 汎化性能 (Generalization)
  - 学習に使用した入力だけでなく、これまで見たことのない新たな入力に対する予測性能
  - (学習誤差ではなく)汎化誤差 (テスト誤差)が小さいものが良い性能を持ったモデル。
  - 新しい入力に対する誤差の期待値で定義される
  - 汎化誤差は通常、学習データとは別に収集された検証データでの性能を測ることで推定される
  - 必要に応じてモデルのバイアスとバリアンスの話を・・・

$$\mathrm{MSE}_{\mathrm{train}} = \frac{1}{n_{\mathrm{train}}} \sum_{i=1}^{n_{\mathrm{train}}} (\hat{y}_i^{(\mathrm{train})} - y_i^{(\mathrm{train})})^2 \Rightarrow \mathrm{MSE}_{\mathrm{test}} = \frac{1}{n_{\mathrm{test}}} \sum_{i=1}^{n_{\mathrm{test}}} (\hat{y}_i^{(\mathrm{test})} - y_i^{(\mathrm{test})})^2$$

訓練誤差:モデルの学習に使用

テスト誤差:モデルの性能の指標

正則化法

- 背景
  - 基底関数の数や基底関数のバンド幅によりモデルの複雑さが変化　(適切な調整が必要)
  - 適切な複雑さを持つモデルを適用しないと過学習や未学習の問題が発生

- 正則化法
  - モデルの複雑さに伴って、その値が大きくなる ペナルティ項(あるいは正則化項)**を課した関数の最小化**を考える
  - 正則化パラメータあるいは平滑化パラメータ : **モデルの曲線のなめらかさを調節** するとともに **推定量の安定に寄与** する役割を果たす

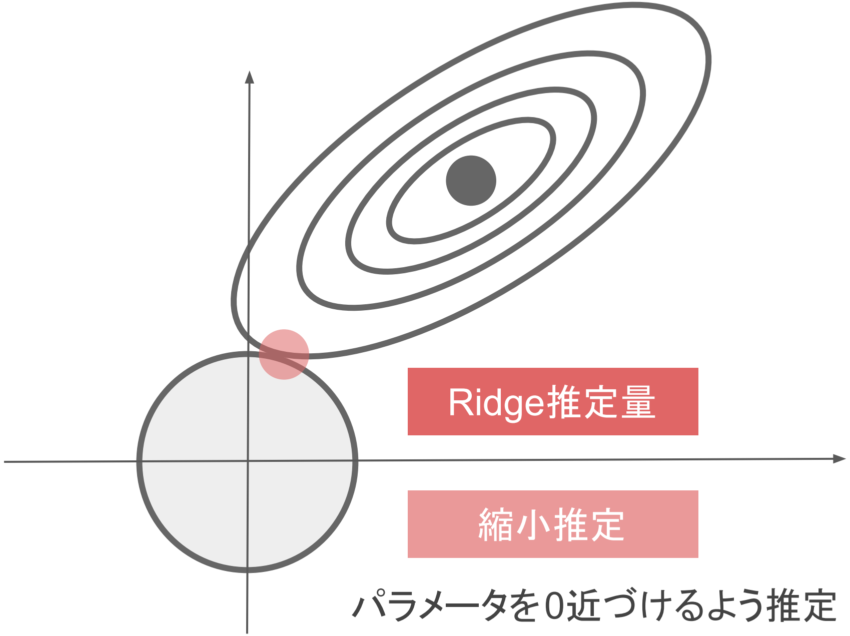
$$S_\gamma = (\boldsymbol{y} - \Phi\boldsymbol{w})^T(\boldsymbol{y} - \Phi\boldsymbol{w}) + \gamma R(\boldsymbol{w})$$

$\gamma(> 0)$

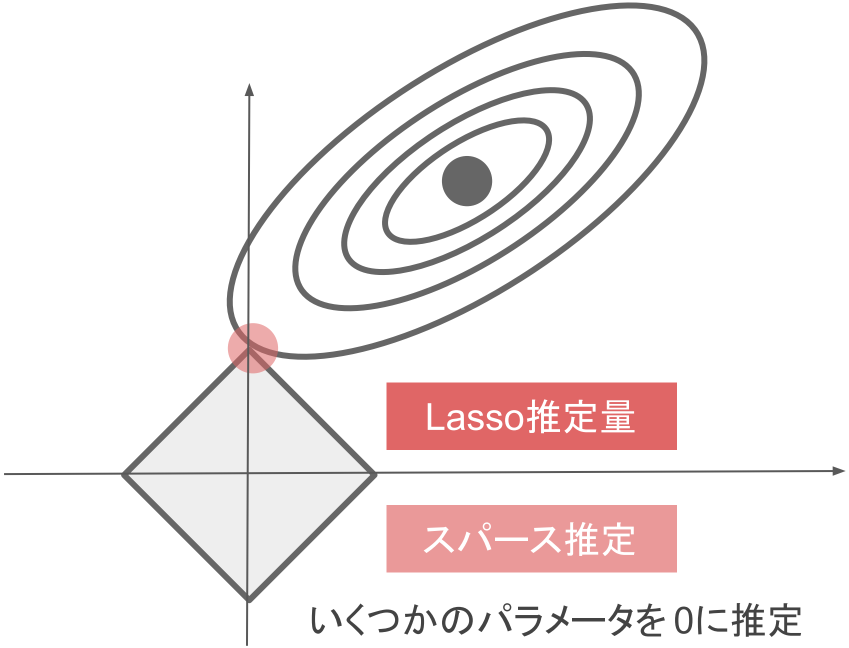
正則化法

- ペナルティ項
  - 無い場合は最小2乗推定量
  - L2ノルムを利用したものをRidge推定量
  - L1ノルムを利用したものをLasso推定量

$$\hat{\boldsymbol{y}} = \Phi(\Phi^{(train)T}\Phi^{(train)} + \gamma I)^{-1}\Phi^{(train)T}\boldsymbol{y}$$



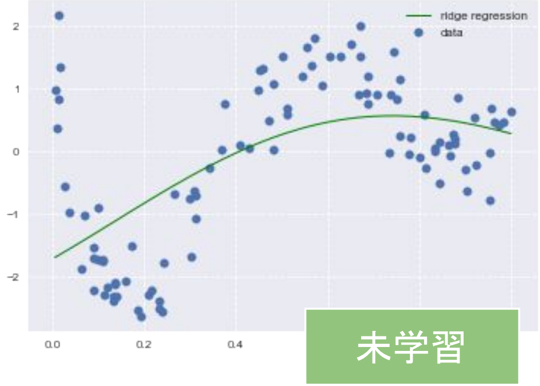
パラメータを0近づけるよう推定

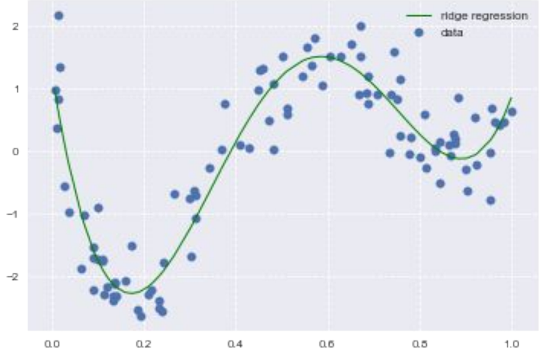


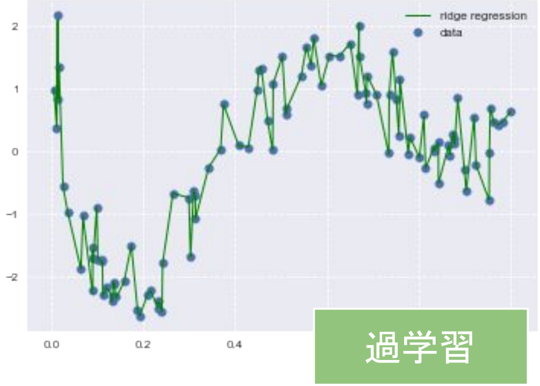
いくつかのパラメータを0に推定

モデル選択

- 過学習(overfitting)と未学習(underfitting)
  - 学習データに対して、十分小さな誤差が得られない場合 →未学習
  - 小さな誤差は得られたけど、テスト集合誤差との差が大きい場合 →過学習
- 過学習は**正則化法**で回避(正則化しすぎると未学習)
- 正則化パラメータ(基底位置・バンド幅)は**クロスバリデーション**で選択







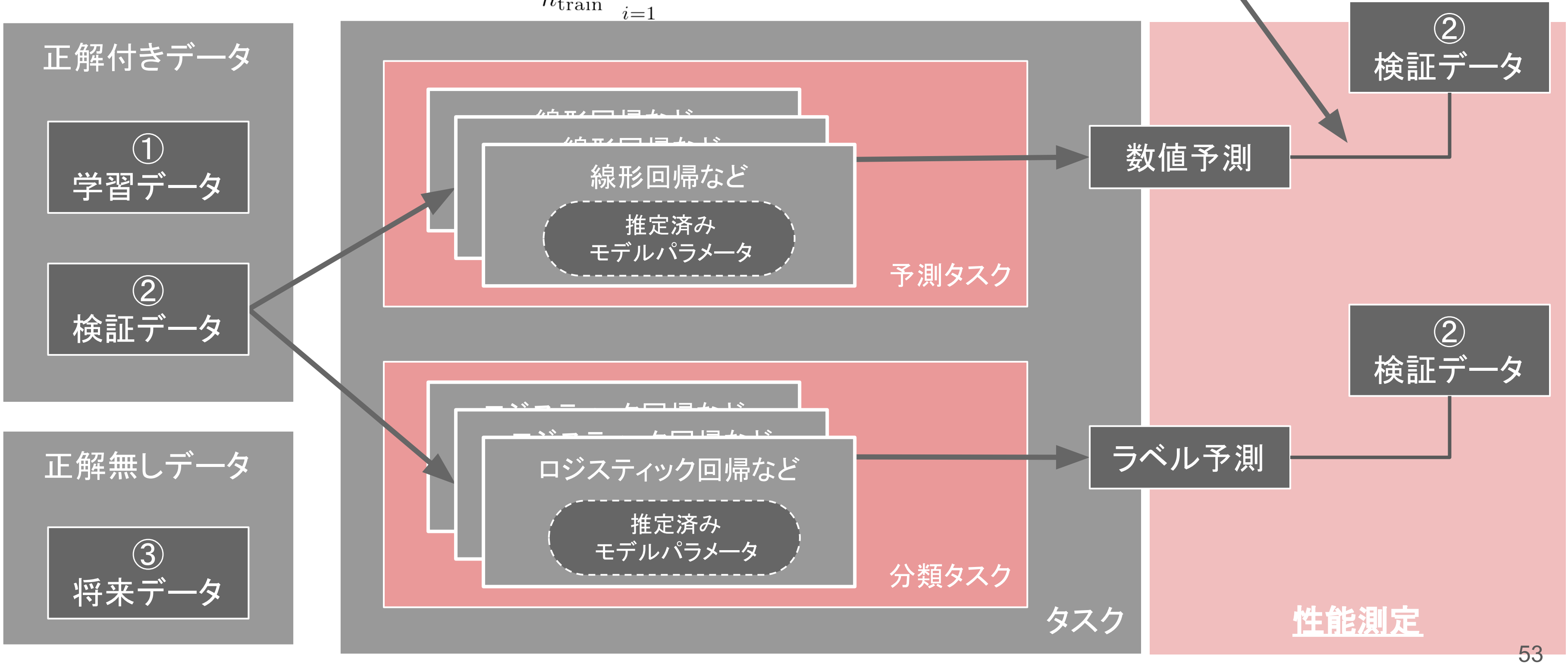
ホールドアウト法

- 手元のデータを2つに分割し、一方を学習に使い、もう一方はテストのために取り置いておき、予測精度や誤り率を推定するために使用する方法
  - 有限のデータを学習用とテスト用に分割するため、学習用を多くすればテスト用が減り、学習精度は良くなるが性能評価の精度は悪くなる
  - 逆にテスト用を多くすれば学習用が減少するので、学習そのものの精度が悪くなることになる。
  - 手元にデータが大量にある場合を除いて、良い性能評価を与えないという欠点がある。

モデルの評価と選択

$$\mathrm{MSE}_{\mathrm{test}} = \frac{1}{n_{\mathrm{test}}} \sum_{i=1}^{n_{\mathrm{test}}} (\hat{y}_i^{(\mathrm{test})} - y_i^{(\mathrm{test})})^2$$

$$\mathrm{MSE}_{\mathrm{train}} = \frac{1}{n_{\mathrm{train}}} \sum_{i=1}^{n_{\mathrm{train}}} (\hat{y}_i^{(\mathrm{train})} - y_i^{(\mathrm{train})})^2$$



クロスバリデーション(交差検証)

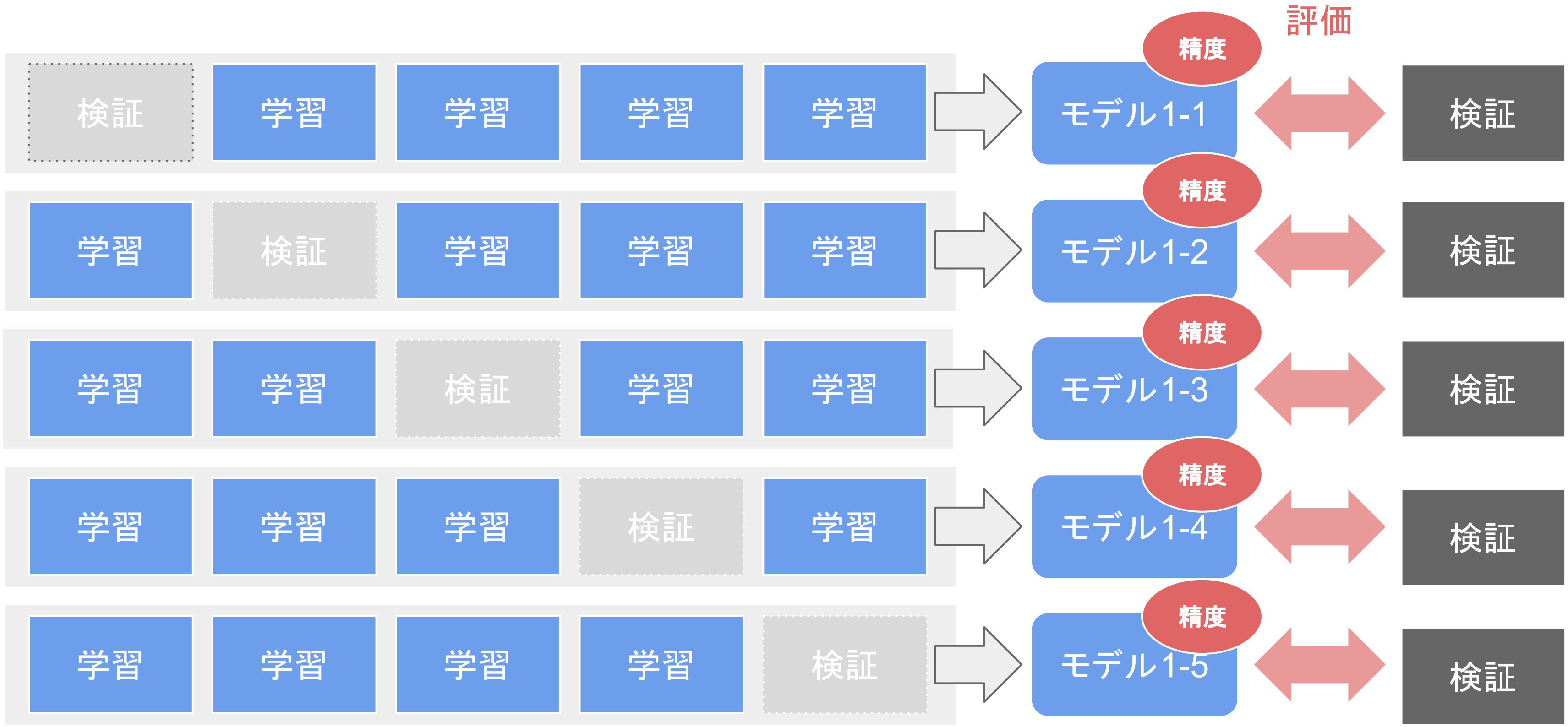
- ホールドアウト法の欠点を補うものとしてよく利用
- 手元の各クラスのデータをそれぞれm個のグループに分割し、m-1個のグループのデータを使って識別器を学習し残りの一つのグループのデータでテストを実行
- これをm回繰り返してそれらの誤り率(平均2乗誤差)の平均を性能予測値とする
- 手元になる全てのデータを学習とテストに利用するので、良い性能予測を行うことができる。

クロスバリデーション(交差検証)

データを学習用と評価用に分割 (5分割の例)

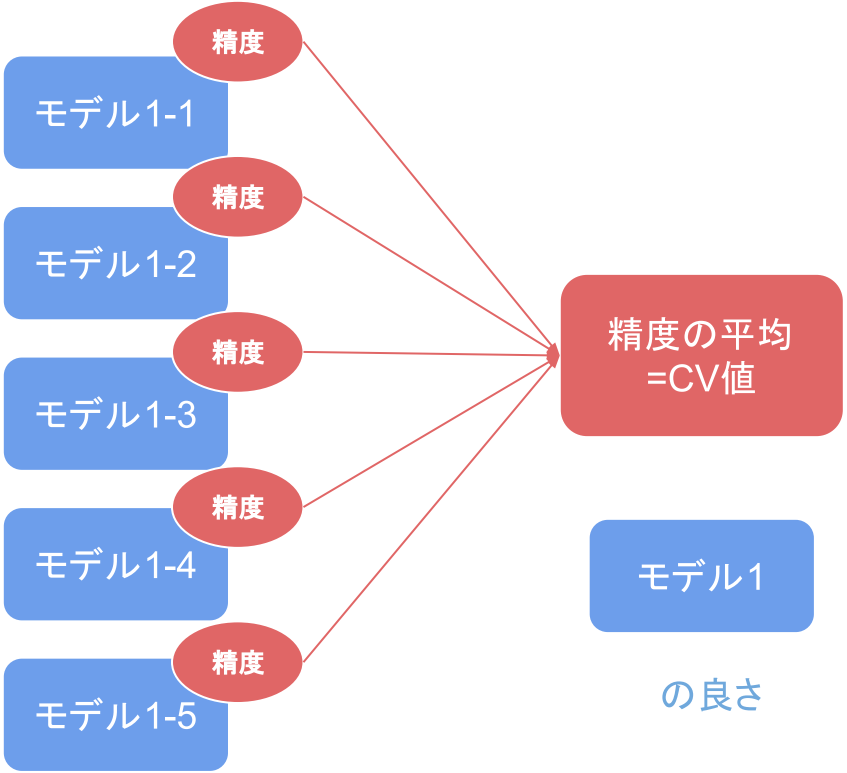
クロスバリデーション(交差検証)

検証データで各モデルの精度を計測



クロスバリデーション(交差検証)

検証データで測った各モデルの精度を算出



学習に全てのデータを使ってしまうとできないことができるようになった

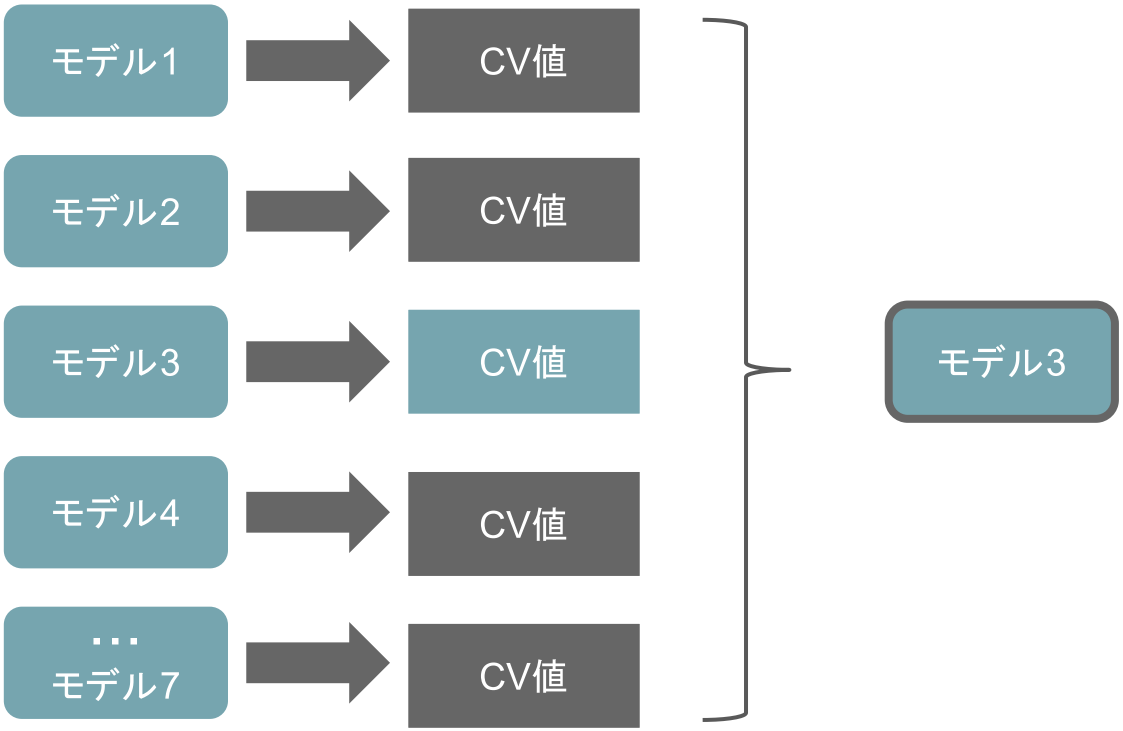
変数選択やハイパーパラメータをこの方法で決める

全てのパターンのクロスバリデーションの値を算出し、CV値が一番高いものを見る

一回だけ分割して精度を測ると分割の仕方に偏りがあるとうまく精度を計測できない

クロスバリデーション(交差検証)

CV値が小さいモデルが予測誤差の意味で良いモデル



CVの値を見比べそれが一番小さかったモデルが一番いいモデル

ビジネスで利用

ハンズオン

- 以下GitHubにアクセスし、ご自身のPCでソースを実行してください。
  - https://github.com/studyaigit/StudyAI-M-L/tree/master/skl_MultivariateAnalysis
- **Seaborn**の使用方法
- ホールドアウト法はロジスティック回帰

分類タスク

ロジスティック回帰モデル

1. データの形式
2. ロジスティック回帰モデル
3. モデル(パラメータ)の推定
4. モデルの評価
5. ハンズオン

分類問題 (クラス分類)

- ある入力(数値)からクラスに分類する問題

- 分類で扱うデータ
  - 入力はm次元のベクトル (m=1の時はスカラ)
  - 出力は0 or 1の値
  - タイタニックデータ、IRISデータなど

0か1

$$\boldsymbol{x} = (x_1, x_2, \cdots, x_m)^T \in \mathbb{R}^m \qquad y \in \{0, 1\}$$

分類問題 (クラス分類)

- ロジスティック線形回帰モデル
  - 分類問題を解くための機械学習モデル
  - 入力からそのラベルを予測するシステムを構築すること
  - 入力とパラメータの線形結合をシグモイド関数に入力
  - 出力はy=1になる確率の値になる

例) titanicデータ
・部屋のクラス
・年齢
・聖別など

生死データ
(0なら生還、1なら死亡)

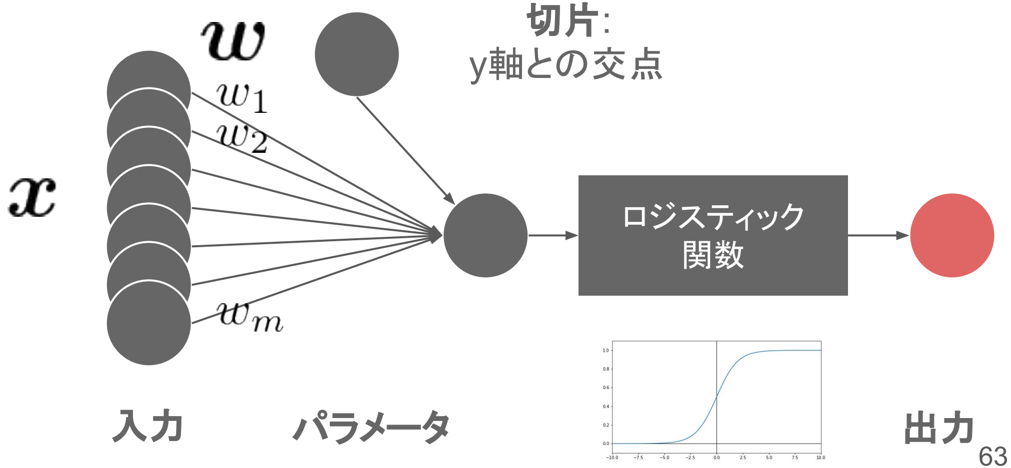
ロジスティック回帰モデル

- 入力とパラメータの線形結合を取るのは線形回帰の場合と同じ
- 線形結合の結果をロジスティック関数の入力とする (次スライド)
- ある入力に対する出力が1になる (クラス1に分類される)確率を表現

線形結合 (線形回帰と同様)

$$\hat{y} = \underline{\boldsymbol{w}}^T \boldsymbol{x} + b = \sum_{j=1}^{m} \underline{w_j} x_j + b$$

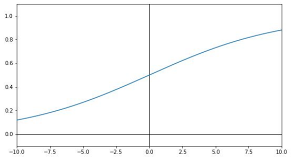
未知パラメータ



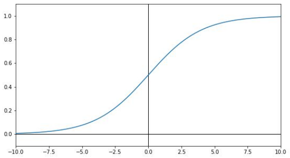
シグモイド関数

- 入力のドメインは実数空間
- 出力は必ず0~1の値になる
- パラメータが変わるとシグモイド関数の形が変わる
    - aを増加させると，x=0付近での曲線の勾配が増加
    - aを極めて大きくすると，単位ステップ関数(x<0でf(x)=0，x>0でf(x)=1となるような関数)に近づきます
    - バイアス変化は段差の位置

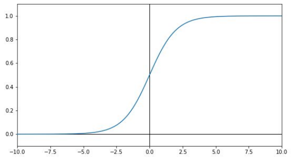
$$\sigma(x) = \frac{1}{1+\exp(-ax)}$$



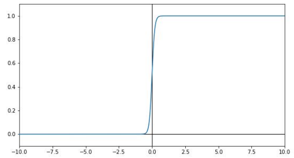
a=0.2



a=0.5



a=1



a= 10

シグモイド関数の性質

- シグモイド関数の微分は、シグモイド関数自身で表現することが可能
- 尤度関数の微分を行う際にこの事実を利用

$$\frac{\partial \sigma(x)}{\partial x} = \frac{\partial}{\partial x}\left(\frac{1}{1+\exp(-ax)}\right)$$

$$= (-1) \cdot \{1+\exp(-ax)\}^{-2} \cdot \exp(-ax) \cdot (-a)$$

連鎖律

$$= \frac{a\exp(-ax)}{\{1+\exp(-ax)\}^2} = \frac{a}{1+\exp(-ax)} \cdot \frac{1+\exp(-ax)-1}{1+\exp(-ax)}$$

$$= a\sigma(x)(1-\sigma(x))$$

ロジスティック回帰モデル

Y=1になる確率とシグモイド関数を対応させる

求めたい値

シグモイド関数

$$P(Y=1|\boldsymbol{x}) = \sigma(w_0 + w_1x_1 + \cdots + w_mx_m)$$

未知のデータが与えられた際にY=1になる確率

データのパラメーターに対する線形結合

表記

$$p_i = \sigma(w_0 + w_1x_{i1} + \cdots + w_mx_{im})$$

線形単(m=1)回帰モデル(データへの仮定)

$$Y \sim Be(p)$$

数式

$$P(Y=1|\boldsymbol{x}) = \sigma(w_0 + w_1 x_1)$$

データが与えられた時にY=1になる確率

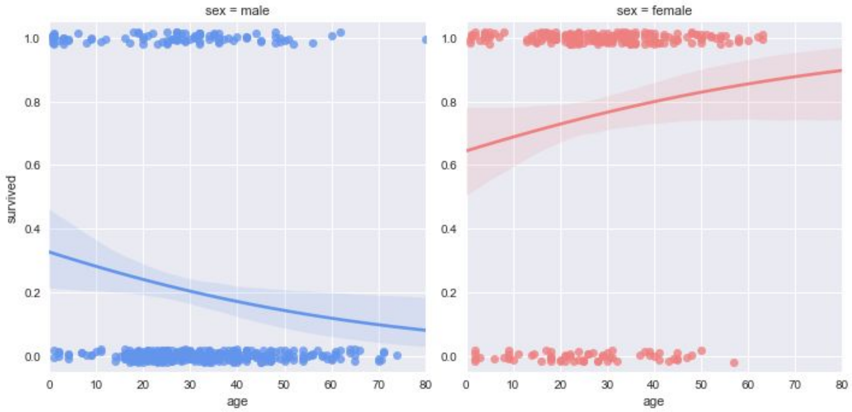
切片　回帰係数

説明変数

既知:入力データ

未知:学習で決める

幾何学的な意味



予測が可能

データYは確率が0.5以上ならば1・それより下なら0とする

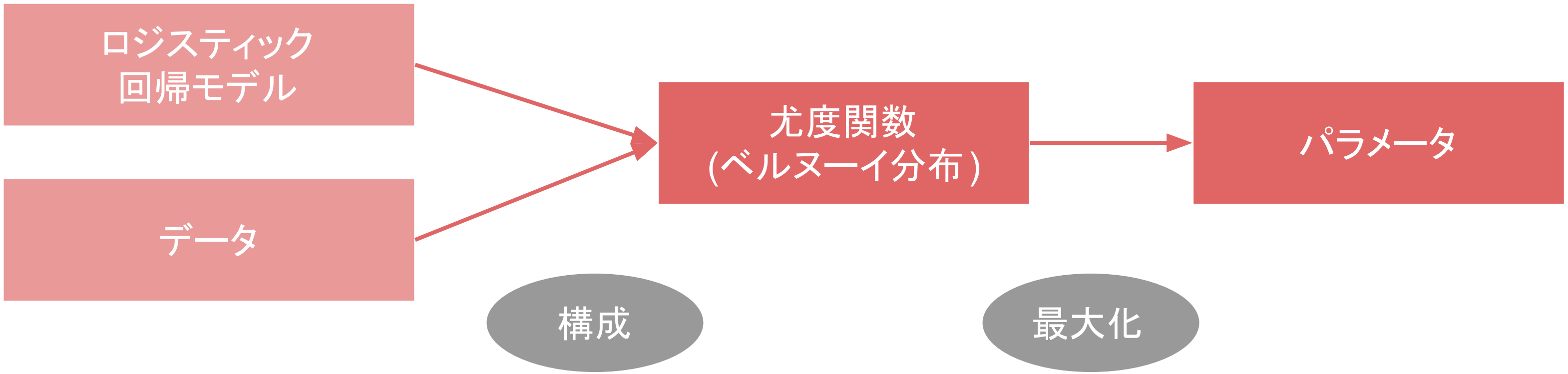
最尤推定

線形単(m=1)回帰モデル(データへの仮定)

- 確率はあるパラメータの分布から特定のデータがどれほど得られやすいかを表現したもの。
- 尤度とは，あるデータを得たときに，分布のパラメータが特定の値であることがどれほどありえそうか（尤もらしいか）を表現したもの。
- 確率はパラメータを固定してデータが変化しますが、尤度はデータを固定してパラメータが変化。
- 確率変数が独立を仮定した場合は、同時分布がそれぞれの分布の積となる。

尤度関数

- ロジスティック回帰モデルのパラメータの最尤推定を考える
- 確率変数Y(実現値として0か1を取る)はベルヌーイ分布に従う



$$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_1), \cdots, (\boldsymbol{x}_n, y_n)$$

※実は誤差への条件を加えることで、線形回帰モデルも最尤法を利用し求めることができる(割愛)

尤度関数

・モデルの出力Yが1となる確率とYが0になる確率を以下で表記

$$P(Y=1|\boldsymbol{x})=p$$

Y=1になる確率

$$P(Y=0|\boldsymbol{x})=1-P(Y=1|\boldsymbol{x})=1-p$$

Y=0になる確率

・確率変数Yはベルヌーイ試行に従う

$$\begin{aligned}P(Y=t|\boldsymbol{x})&=P(Y=1|\boldsymbol{x})^tP(Y=0|\boldsymbol{x})^{1-t}\\&=p^t(1-p)^{1-t}\end{aligned}$$

Y=tになる確率

尤度関数

$\boldsymbol{x}_i$ を与えた時、 $Y = y_i$ になる確率 (まとめて表記)

$$P(Y = y_1|\boldsymbol{x}_1) = p_1^{y_1}(1 - p_1)^{1-y_1}$$

$$P(Y = y_2|\boldsymbol{x}_2) = p_2^{y_2}(1 - p_2)^{1-y_2}$$

…

$$P(Y = y_n|\boldsymbol{x}_n) = p_n^{y_n}(1 - p_n)^{1-y_n}$$

Y=1の時に残る　Y=0の時に残る

ロジスティック回帰モデル

y1=1の場合はなるべく高くなるように

数式(全部)

$\boldsymbol{x}_i$ を与えた時、 $Y = 1$ になる確率 (全データ分)

$$P(Y = 1|\boldsymbol{x}_1) = p_1 = \sigma(w_0 + w_1 x_{11} + \cdots + w_m x_{1m})$$

0 or 1

$$P(Y = 1|\boldsymbol{x}_2) = p_2 = \sigma(w_0 + w_1 x_{21} + \cdots + w_m x_{2m})$$

0 or 1

$$P(Y = 1|\boldsymbol{x}_n) = p_n = \sigma(w_0 + w_1 x_{n1} + \cdots + w_m x_{nm})$$

0 or 1

データがY=1になる確率を仮定

同時確率 (パラメータの関数→尤度関数)

- 学習データセットが同時に得られる確率を計算　$(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_1), \cdots, (\boldsymbol{x}_n, y_n)$
- 観測されたデータ(学習データ)を発生させる**尤もらしい**確率分布を求める

$$
\begin{aligned}
P(Y = y_1|\boldsymbol{x}_1)P(Y = y_2|\boldsymbol{x}_2)\cdots P(Y = y_n|\boldsymbol{x}_n) \\
&= p_1^{y_1}(1-p_1)^{1-y_1} p_2^{y_2}(1-p_2)^{1-y_2} \cdots p_n^{y_n}(1-p_n)^{1-y_n} \\
&= \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \\
&= \underline{L(w_0, w_1, \cdots, w_m)}
\end{aligned}
$$

同時確率

尤度関数

尤度関数を最大にするパラメータを推定パラメータとする

対数尤度関数の最大化

- 尤度関数を最大化するよりも、対数尤度関数を最大化する方が楽
  - 積が和、指数が積の演算に変換できる
  - 対数尤度関数が最大になる点と尤度関数が最大になる点は同じ (対数はは単調増加関数)
  - 平均二乗誤差は最小化、尤度関数は最大化はややこしいので、対数尤度関数にマイナスを掛けて「最小化」で統一

$$
\begin{aligned}
E(w_0, w_1, \cdots, w_m) &= -\log L(w_0, w_1, \cdots, w_m) \\
&= -\sum_{i=1}^{n} \{t_n \log p_i + (1 - t_n) \log(1 - p_i)\}
\end{aligned}
$$

勾配降下法

- なぜ必要か？
  - [線形回帰モデル (最小2乗法)] ▶︎ MSEのパラメータに関する微分が 0になる値を解析に求めることが可能
  - [ロジスティック回帰モデル (最尤法)] ▶︎ 対数尤度関数をパラメータで微分して 0になる値を求める必要があるのだが、解析的にこの値を求めることは困難である。

- 勾配降下法 (Gradient descent)
  - 反復学習によりパラメータを逐次的に更新するアプローチの一つ
  - ηは学習率と呼ばれるハイパーパラメータでモデルのパラメータの収束しやすさを調整

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} - \eta \frac{\partial E(\boldsymbol{w}, b)}{\partial \boldsymbol{w}} \qquad b^{(k+1)} = b^{(k)} - \eta \frac{\partial E(\boldsymbol{w}, b)}{\partial b}$$

勾配降下法

- 対数尤度関数を係数とバイアスに関して微分

$$
\begin{aligned}
\frac{\partial E(\boldsymbol{w}, b)}{\partial \boldsymbol{w}} &= \sum_{i=1}^{n} \frac{\partial E_i}{\partial p_i} \frac{\partial p_i}{\partial \boldsymbol{w}} \\
&= -\sum_{i=1}^{n} \left(\frac{t_i}{p_i} - \frac{1-t_i}{1-p_i} \right) \frac{\partial y_i}{\partial \boldsymbol{w}} \\
&= -\sum_{i=1}^{n} \left(\frac{t_i}{p_i} - \frac{1-t_i}{1-p_i} \right) p_i(1-p_i)\boldsymbol{x}_i \\
&= -\sum_{i=1}^{n} \left\{ t_i(1-p_i) - p_i(1-t_i) \right\} \boldsymbol{x}_i \\
&= -\sum_{i=1}^{n} (t_i - y_i)\boldsymbol{x}_i
\end{aligned}
$$

連鎖律

対数尤度関数の
pに関する微分

シグモイド関数の微分

式を整理

$$
\frac{\partial E(\boldsymbol{w}, b)}{\partial b} = -\sum_{i=1}^{n} (t_i - p_i)
$$

勾配降下法

- パラメータが更新されなくなった場合、それは勾配が0になったということ。少なくとも反復学習で探索した範囲では最適な解がもとめられたことになる。

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \eta \sum_{i=1}^{n} (t_i - p_i)\boldsymbol{x}_i \qquad b^{(k+1)} = b^{(k)} + \eta \sum_{i=1}^{n} (t_i - p_i)$$

- 勾配降下法では、パラメータを更新するのにN個全てのデータに対する和を求める必要がある。
  - nが巨大になった時にデータをオンメモリに載せる容量が足りない、計算時間が莫大になるなどの問題がある
  - 確率的勾配降下法をを利用して解決

確率的勾配降下法(SGD)

- 確率的勾配降下法
  - データを一つずつランダムに選んでパラメータを更新
  - 「確率的」とついているのはデータをランダムに選ぶから
  - 勾配降下法でパラメータを 1回更新するのと同じ計算量でパラメータを n回更新できるので効率よく最適な解を探索可能
- エポック
  - n回で勾配が0に収束する (学習が終わる)ことは稀でn個のデータ全体に対して繰り返し学習をする必要がある
  - このデータ全体に対する反復回数をエポック (epoch)と呼ぶが、各エポックごとのデータをまたシャッフルするして学習することで学習に偏りが生じにくくなり、より最適な解を得やすくなる。

$$\boldsymbol{w}^{(k+1)} = \boldsymbol{w}^{(k)} + \eta(t_n - y_n)\boldsymbol{x}_n \qquad b^{(k+1)} = b^{(k)} + \eta(t_n - y_n)$$

ミニバッチ勾配降下法

- ミニバッチ勾配降下法
  - n個のデータをm (< n)個ずつのかたまり(ミニバッチ)に分けて学習を行うもの
  - 一般的にはm=50~500くらいが用いられる
  - ミニバッチによる学習では、メモリ不足になることなく線形演算を行えるので、データ 1個ずつの繰り返しよりも計算が高速になる
  - ミニバッチのサイズをm=1とすると確率的勾配降下法に相当

確率的勾配法

- 学習率
- 動画

混同行列 (Confusion Matrix)

- True Positive：生存者を生存者と判別
- False Positive：生存者を死亡者と判別
- False Negative：死亡者を生存者と判別
- True Negative：死亡者を死亡者と判別

		検証用データの結果	
		生存	死亡
モデルの予測結果	生存	True Positive	False Positive
	死亡	False Negative	True Negative

分類の評価方法

- 正解率
  - 正解した数と予測した全データ数の割合
  - メールのスパム分類でいうと、届いたメール 100件を人の手で確認した所、スパムの数が 80件で普通のメールが20件であった場合、全てをスパムとする分類機があったとすると、正解率は 80%となる。
  - 一番精度の悪い分類機はの精度は 50%。上記のケースでは正解率は 80%だが「良い」分類器とは言えない。(全てをスパムと予測)
  - 分類したいクラスにはそれぞれ偏りがあることが多く、偏りがあるデータに対して単純な正解率はあまり意味をなさないことがほとんどです。

$$\frac{TP + TN}{TP + FN + FP + TN}$$

分類の評価方法

- 適合率(Precision)
  - 見逃しが多くてもより正確な予測をしたい場合
  - メールのスパム判定で言えば、重要なメールをスパムと誤判定をされて見逃されるよりは、たまにスパムがすり抜けても構わない。スパムと予測したものが確実にスパムである方が安心できる。

$$\frac{TP}{TP+FP}$$

- 再現率(Recall)
  - 誤りが多少多くても抜け漏れを少なくしたい場合
  - 発生する件数の少ない病気の検診で、病気であると誤判定するケースが多少あっても、再検査をすればそれでいい

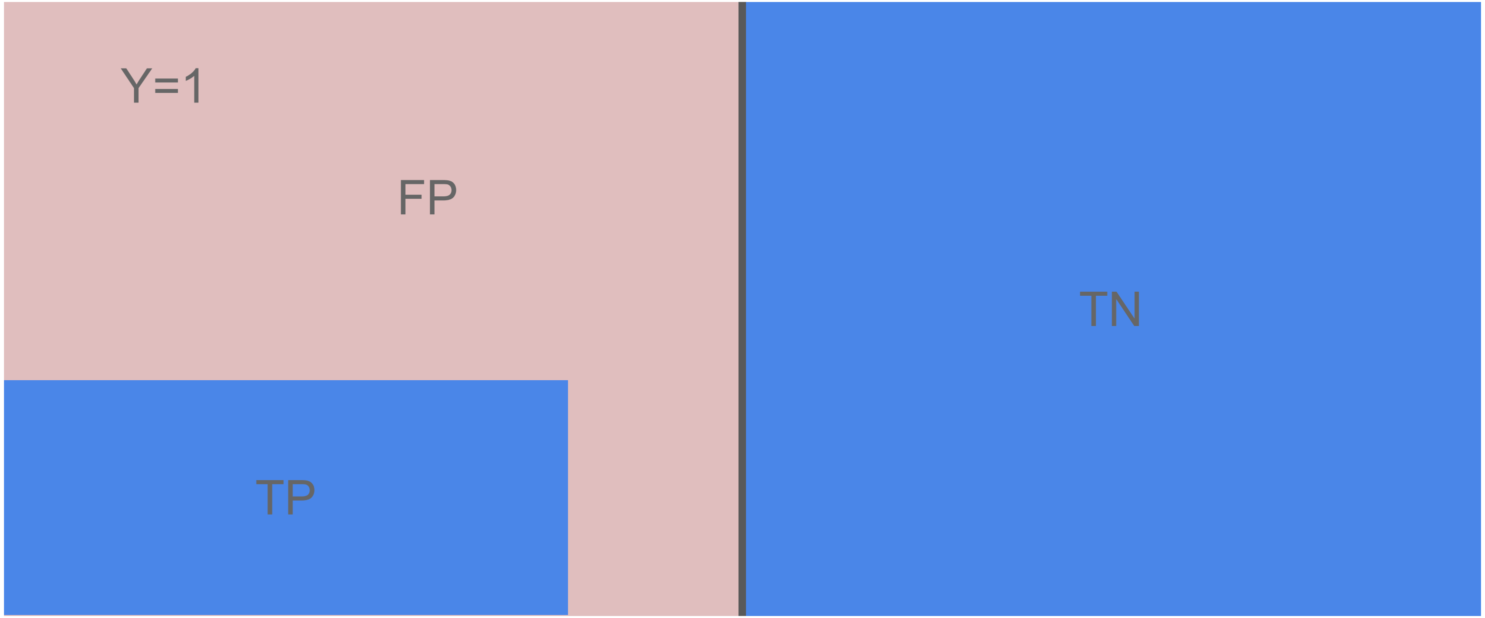
$$\frac{TP}{TP+FN}$$

- F値
  - 二つはトレードオフの関係にあり、どちらかを小さくするもでるはもう片方の値がおおきくなってしまう。そのような場合に利用

分類の評価方法

$$\frac{TP}{TP+FP}$$

(precision 低)
誤りが(多少)多くても、
抜け漏れを少なくしたい場合



FN
:なし

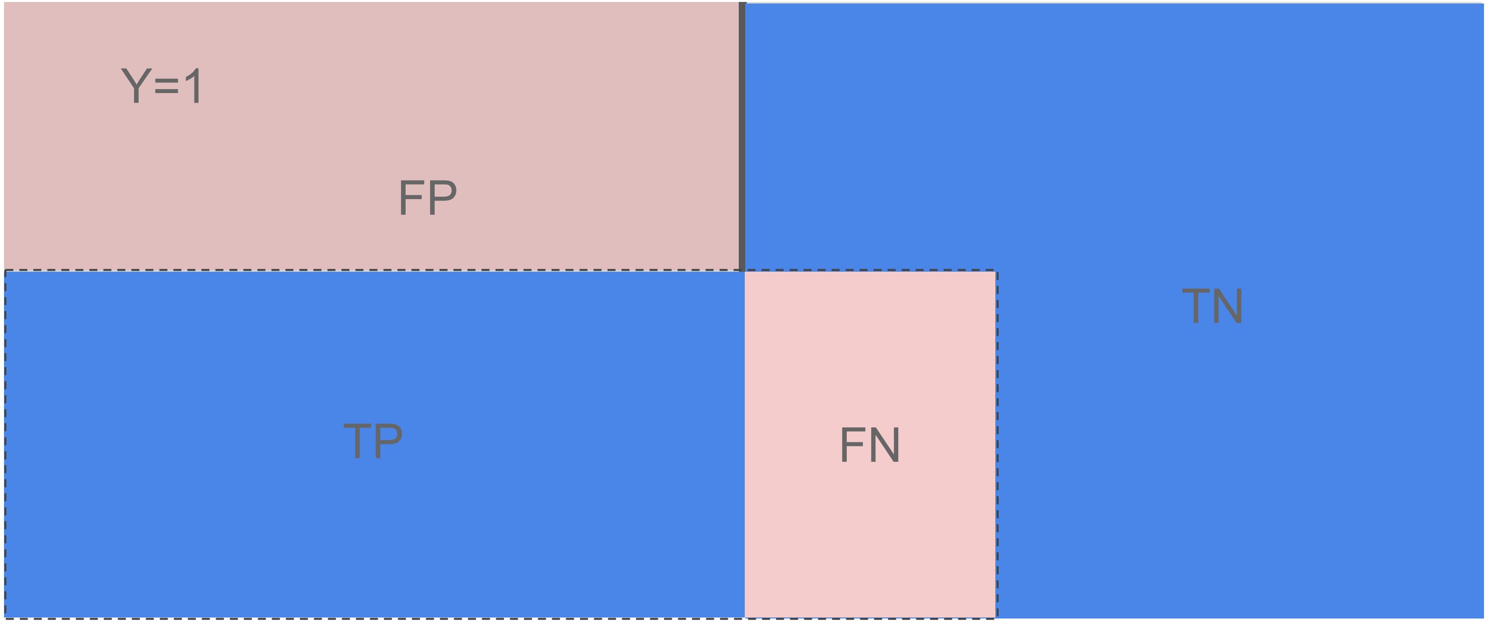
$$\frac{TP}{TP+FN}$$

(recall 高)
見逃しが多くても、
正確な予測をしたい場合

分類の評価方法

(precision 高)
誤りが(多少)多くても、
抜け漏れを少なくしたい場合

$$\frac{TP}{TP+FP}$$



(recall 低)
見逃しが多くても、
正確な予測をしたい場合

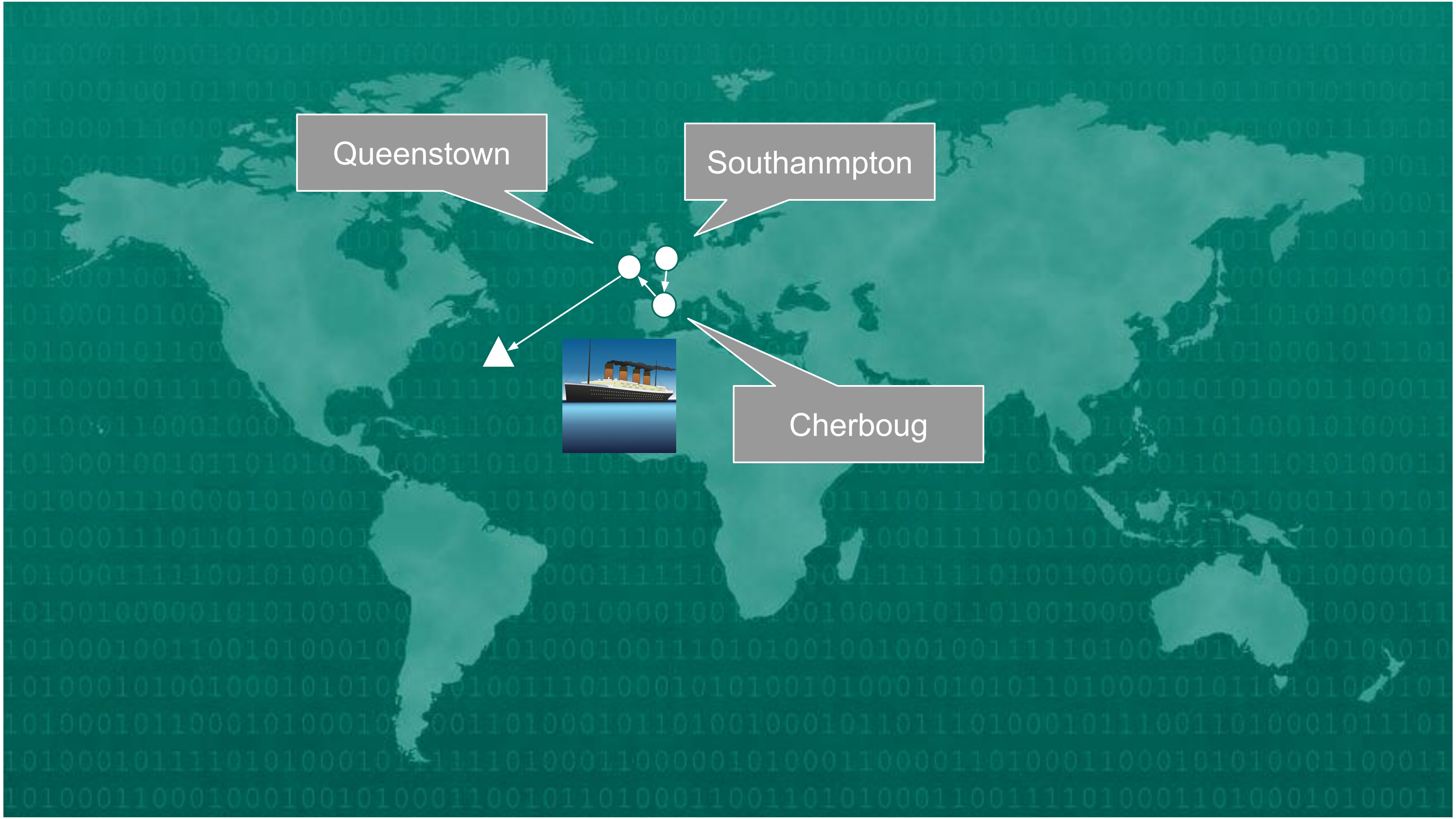
$$\frac{TP}{TP+FN}$$

ハンズオン (ロジスティック回帰モデル)

- ロジスティックモデルで分類器を作成

- タイタニックデータを利用
  - レコード数(乗客データの数):891 (train) / 417(test)
  - 891個の乗客データにはそれぞれ 11個のカラムがある
  - カラム数 (性別・チケットの価格など乗客の情報):
  - データセットの詳細: https://www.kaggle.com/c/titanic

- 以下GitHubにアクセスし、ご自身のPCでソースを実行してください。
  - https://github.com/studyaigit/StudyAI-M-L/tree/master/skl_MultivariateAnalysis


Queenstown
Southanmpton
Cherboug

項番	変数	記号名	値	
1	PassengerId	乗客ID	1, 2 ...	
2	Survived	生存・死者情報	1 -> 生存者(Alive) / 0 ->死者(Dead)	
3	Pclass	乗客の社会階級	1 (High) / 2 (Middle) / 3 (Low))	
4	Name	乗客名	-	
5	Sex	性別	男性: male / 女性: female	
6	Age	年齢	22.0 / 38.0 など	欠損あり
7	SibSp	兄弟および配偶者の数	0 / 1/ 2 など	
8	Parch	親もしくは子供の数	0 / 1 など	
9	Ticket	チケットNo	A/5 21171 など	
10	Fare	運賃	7.2500 など	
11	Cabin	船室	C85など	欠損あり
12	Embarked	乗船した港(3つ)	C: Cherbourg; Q: Queenstown, S: Southampton	テストデータには無し

項番	変数	記号名	値
1	PassengerId	乗客ID	1, 2 ...
2	Survived	生存・死者情報	1 -> 生存者(Alive) / 0 ->死者(Dead)
3	Pclass	乗客の社会階級	1 (High) / 2 (Middle) / 3 (Low))
4	Name	乗客名	-
5	Sex	性別	男性: male / 女性: female
6	Age	年齢	22.0 / 38.0 など
7	SibSp	兄弟および配偶者の数	0 / 1/ 2 など
8	Parch	親もしくは子供の数	0 / 1 など
9	Ticket	チケットNo	A/5 21171 など
10	Fare	運賃	7.2500 など
11	Cabin	船室	C85など
12	Embarked	乗船した港(3つ)	C: Cherbourg; Q: Queenstown, S: Southampton

平均

欠損あり

欠損あり

テストデータ
には無し

項番	変数	記号名	値
1	**PassengerId**	乗客ID	1, 2 ...
2	**Survived**	生存・死者情報	1 -> 生存者(Alive) / 0 ->死者(Dead)
3	**Pclass**	乗客の社会階級	1 (High) / 2 (Middle) / 3 (Low))
4	[illegible]**me**	乗客名	-
[illegible]	[illegible]	性別	男性: male / 女性: female
[illegible]	**Age**	年齢	22.0 / 38.0 など
[illegible]	**SibSp**	兄弟および配偶者の数	0 / 1/ 2 など
8	**Parch**	親もしくは子供の数	0 / 1 など
9	**Ticket**	チケットNo	A/5 21171 など
10	**Fare**	運賃	7.2500 など
11	**Cabin**	船室	C85など
12	**Embarked**	乗船した港(3つ)	C: Cherbourg; Q: Queenstown, S: Southampton

まずは2変量に絞る

ラベル

入力データ

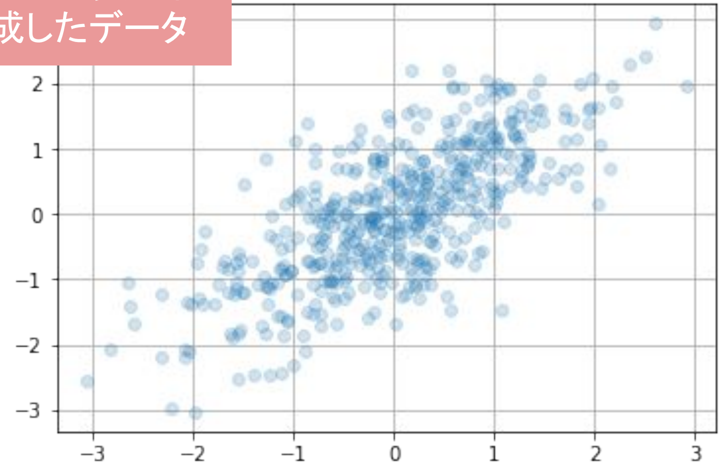
主成分分析

1. データの形式
2. アウトライン
3. 主成分
4. 寄与率

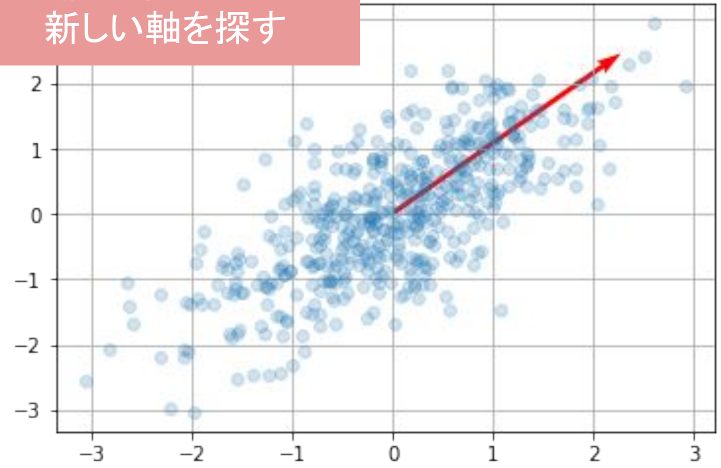
主成分分析

- 多変量データの持つ構造をより少数個の指標にまとめる。
  - 変量の個数を減らすことに伴う情報の損失はなるべく小さくする必要がある。
  - **学習データの分散が最大になる方向への線形変換** を求める手法である
  - 少数変数を利用した分析や可視化 (2･3次元の場合)が実現可能

相関0.8の正規分布
から生成したデータ



分散が最大になる
新しい軸を探す



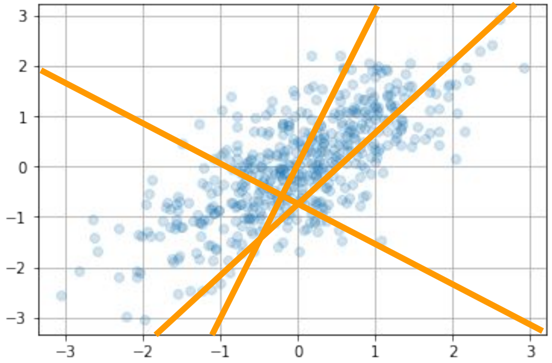
主成分分析

- 学習データ: $$\boldsymbol{x} = (x_1, x_2, \cdots, x_m)^T \in \mathbb{R}^m$$
- 平均ベクトル: $$\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{x}_i$$
- (平均ベクトルを引き算した) データ行列: $$\bar{X} = (\boldsymbol{x}_1 - \bar{\boldsymbol{x}}, \cdots, \boldsymbol{x}_n - \bar{\boldsymbol{x}})^T$$
- 分散共分散行列: $$\Sigma = Var(\bar{X}) = \frac{1}{n}\bar{X}^T\bar{X}$$
- n個のデータを係数ベクトルを用いて線形変換:

$$\boldsymbol{s}_j = (s_{1j}, \cdots, s_{nj})^T = \bar{X}\boldsymbol{a}_j$$

主成分分析

- 係数ベクトルが変われば線形変換後の値が変わる
  - どのような線形変換を求めるば良いか？
- 情報の量を分散の大きさ捉え、変換後の分散が最大となる射影軸(線形変換)を探索



$$\boldsymbol{s}_j = (s_{1j}, \cdots, s_{nj})^T = \bar{X}\boldsymbol{a}_j$$

線形変換後の分散

$$Var(\boldsymbol{s}_j) = \frac{1}{n}\boldsymbol{s}_j^T\boldsymbol{s}_j = \frac{1}{n}(\bar{X}\boldsymbol{a}_j)^T(\bar{X}\boldsymbol{a}_j)$$
$$= \frac{1}{n}\boldsymbol{a}_j^T\bar{X}\bar{X}\boldsymbol{a}_j = \boldsymbol{a}_j^T Var(\bar{X})\boldsymbol{a}_j$$

主成分分析

- 以下の最適化問題を解く(ノルムに制約を入れないと無限に解がある)

$$\underset{\boldsymbol{a} \in \mathbb{R}^m}{\arg\max} \ \boldsymbol{a}_j^T Var(\bar{X}) \boldsymbol{a}_j \qquad \text{subject to} \qquad \boldsymbol{a}_j^T \boldsymbol{a}_j = 1$$

- 上記の制約付き最適化問題はラグランジュ関数を最大にする係数ベクトルを見つければいい。
  - 係数ベクトルで微分して0とおき解を求める
  - 下記が分散を最大にする係数ベクトル (**分散共分散行列の固有ベクトル** が解)

ラグランジュ関数

$$E(\boldsymbol{a}_j) = \boldsymbol{a}_j^T Var(\bar{X}) \boldsymbol{a}_j - \lambda(\boldsymbol{a}_j^T \boldsymbol{a}_j - 1)$$

$$\frac{\partial E(\boldsymbol{a}_j)}{\partial \boldsymbol{a}_j} = 2Var(\bar{X})\boldsymbol{a}_j - 2\lambda \boldsymbol{a}_j = 0$$

解

$$Var(\bar{X})\boldsymbol{a}_j = \lambda \boldsymbol{a}_j$$

主成分分析

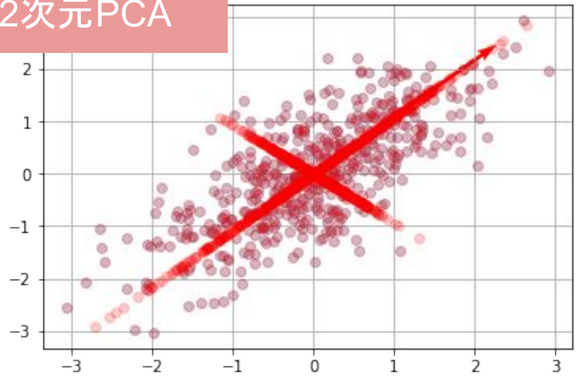
- 射影ベクトルの向きは固有ベクトル (前スライドより)
- 分散の値は、固有値と一致

$$Var(\boldsymbol{s}_1) = \boldsymbol{a}_1^T Var(\bar{X}) \boldsymbol{a}_1 = \lambda_1 \boldsymbol{a}_1^T \boldsymbol{a}_1 = \lambda_1$$

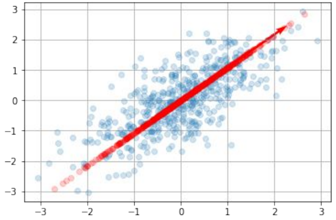
- 分散共分散行列は実対称行列なので、固有ベクトルは全て直交

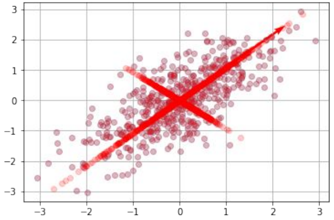
1次元PCA

2次元PCA



主成分分析





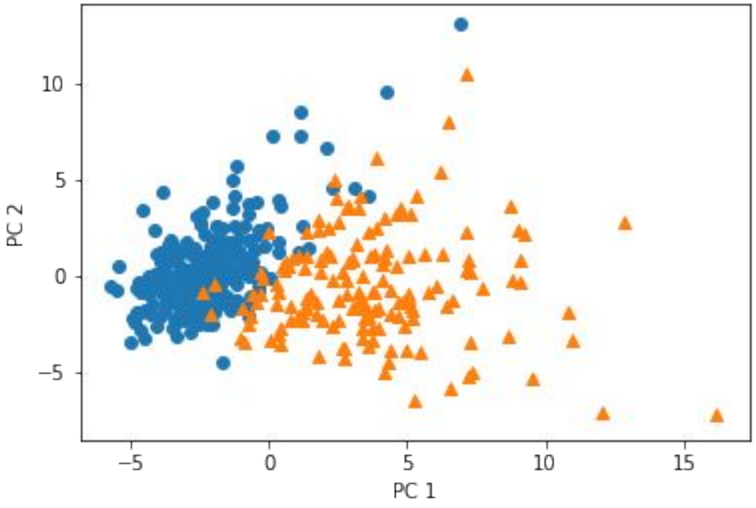
- 主成分
  - 最大固有値に対応する固有ベクトルで線形変換された特徴量を第一主成分と呼ぶ。
  - k番目の固有値に対応する固有ベクトルで変換された特徴量を第 k主成分と呼ぶ

- 寄与率
  - 変換された特徴量の分散は固有値に一致するので、全分散量は元データの持つ全分散量とも一致 (元次元と同じ次元数の主成分を使えば、情報量は一致 : 図は 2次元PCAの場合で完全に復元)
  - 第k主成分の分散の全分散に対する割合を第 k成分の寄与率という (第k主成分が持つ情報量の割合)
  - 第k成分での累積寄与率 (第1-k主成分まで圧縮した際の情報損失量の割合)

$$V_{total} = \sum_{i=1}^{m} \lambda_i \qquad c_k = \frac{\lambda_k}{\sum_{i=1}^{m} \lambda_i} \qquad r_k = \frac{\sum_{j=1}^{k} \lambda_j}{\sum_{i=1}^{m} \lambda_i}$$

ハンズオン

・プログラミング結果だけ表示 (乳がん検査データ)

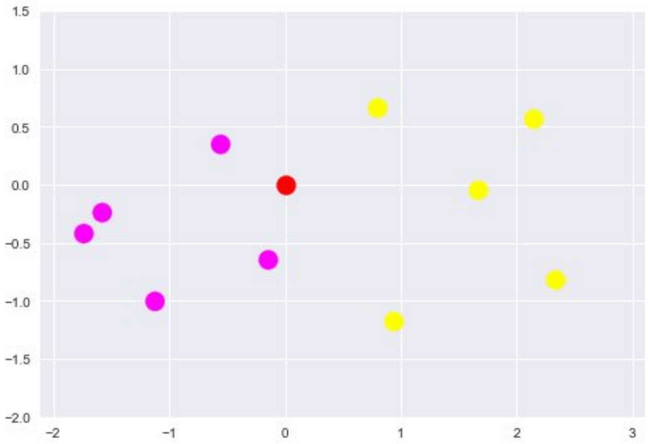


k近傍法

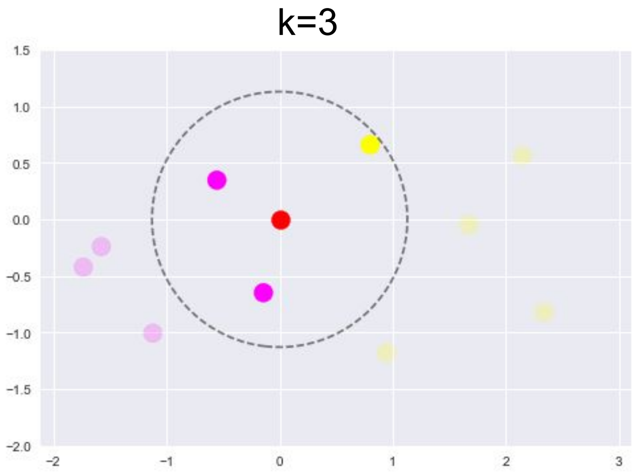
1.

k近傍法(kNN)

- 分類問題のための機械学習手法
- 近傍k個のデータから識別する
- k個のクラスラベルの中で最も多いラベルを割り当てる



新しいデータ(赤色)を分類する

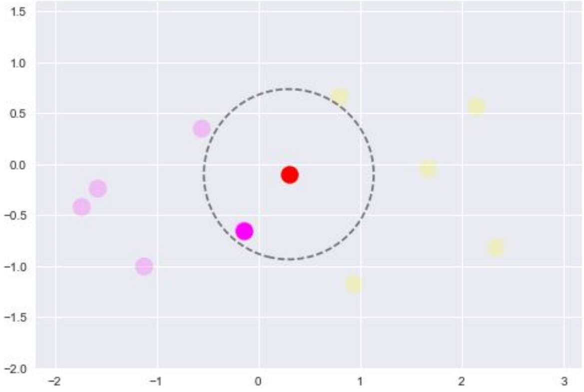


近傍の点は紫2個、黄1個なので紫クラスに分類する

k近傍法(kNN)

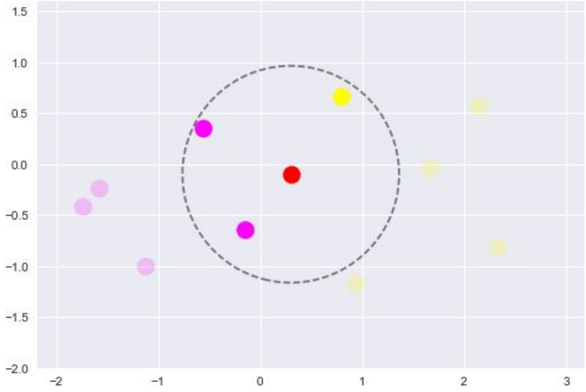
- kを変化させると結果も変わる

k=1 (最近傍法)



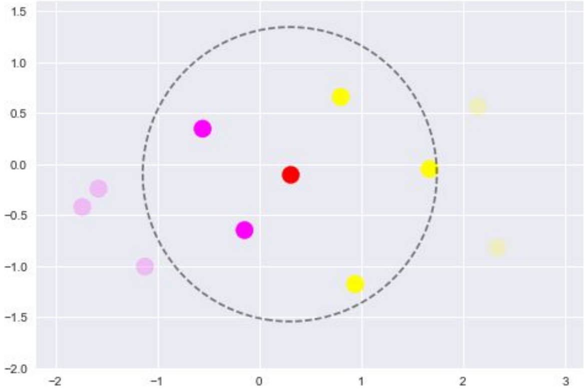
紫1個、黄0個で
紫クラスに分類

k=3



紫2個、黄1個で
紫クラスに分類

k=5

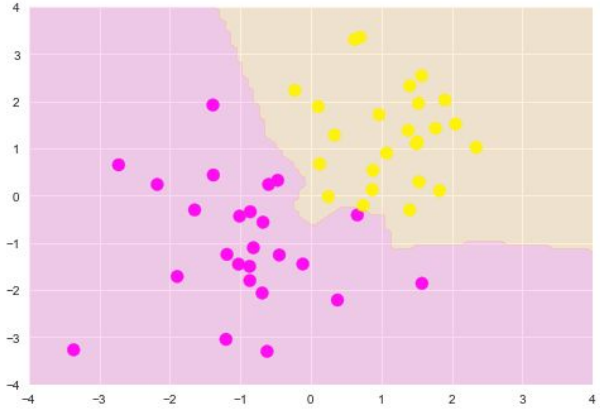


紫2個、黄3個で
黄クラスに分類

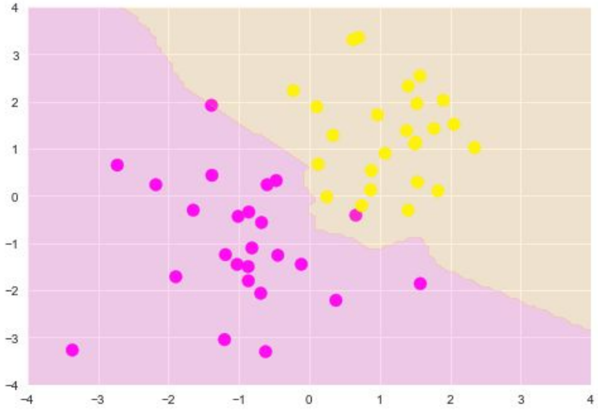
k近傍法(kNN)

- kを大きくすると決定境界は滑らかになる

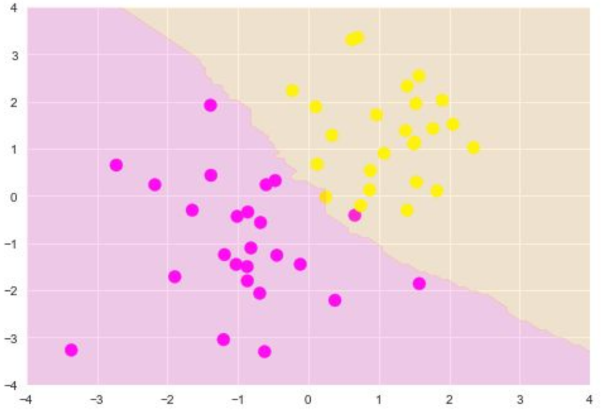
k=1 (最近傍法)



k=3



k=10



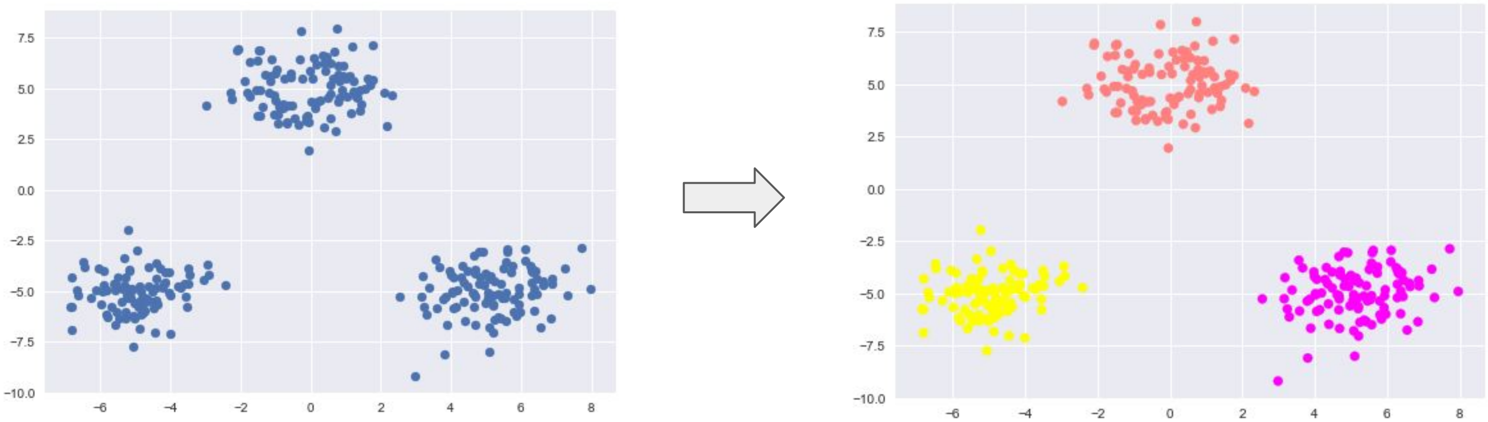
k平均法

1.

k平均法(k-means)

- 教師なし学習
- クラスタリング手法
- 与えられたデータをk個のクラスタに分類する

クラスタリング・・・特徴の似ているもの同士をグループ化



k平均法(k-means)のアルゴリズム

1) 各クラスタ中心の初期値を設定する

2) 各データ点に対して、各クラスタ中心との距離を計算し、
最も距離が近いクラスタを割り当てる

3) 各クラスタの平均ベクトル（中心）を計算する

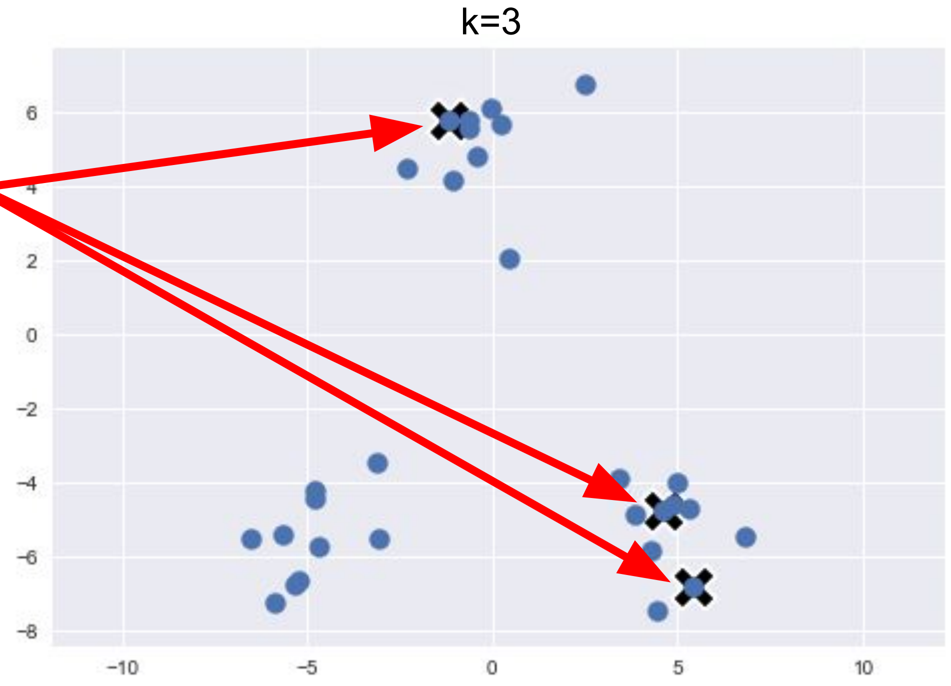
4) 収束するまで2, 3の処理を繰り返す

次から各手順の詳細を説明する

手順①

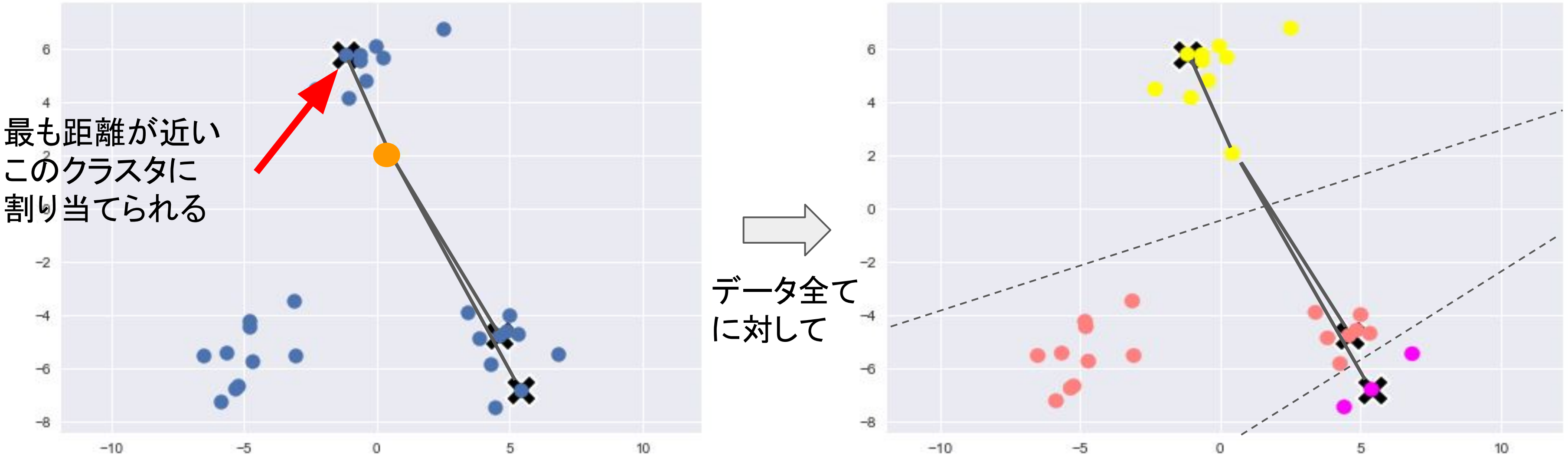
● 各クラスタ中心の初期値を設定する

最初のクラスタ中心を
ランダムに選ぶ



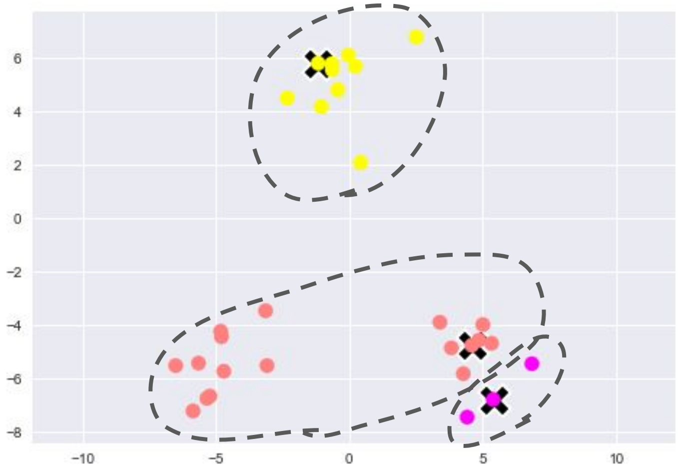
手順②

- 各データ点に対して、各クラスタ中心との距離を計算し、最も距離が近いクラスタを割り当てる

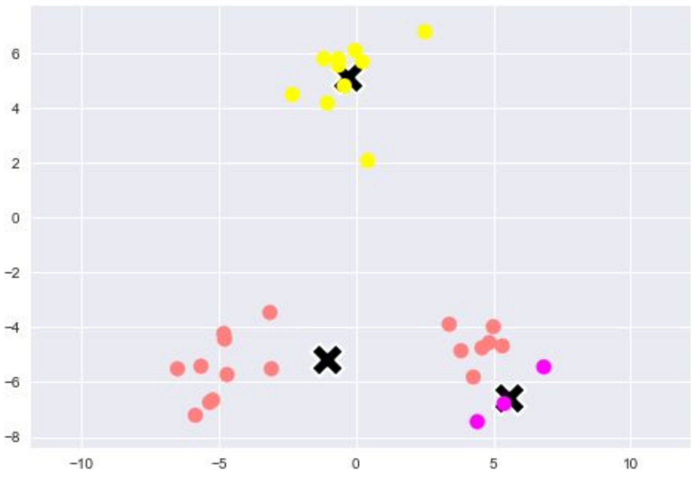


手順③

- 各クラスタの平均ベクトル（中心）を計算する

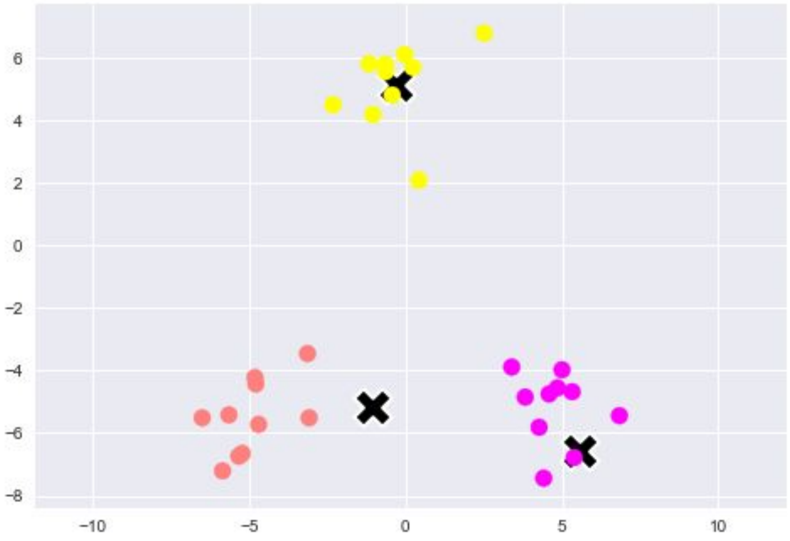


中心を更新

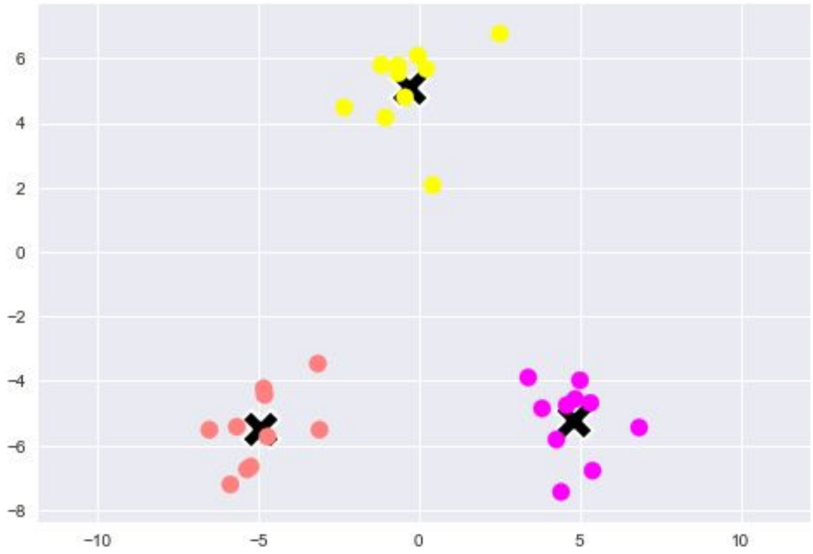


手順④

- クラスタの再割り当てと、中心の更新を繰り返す



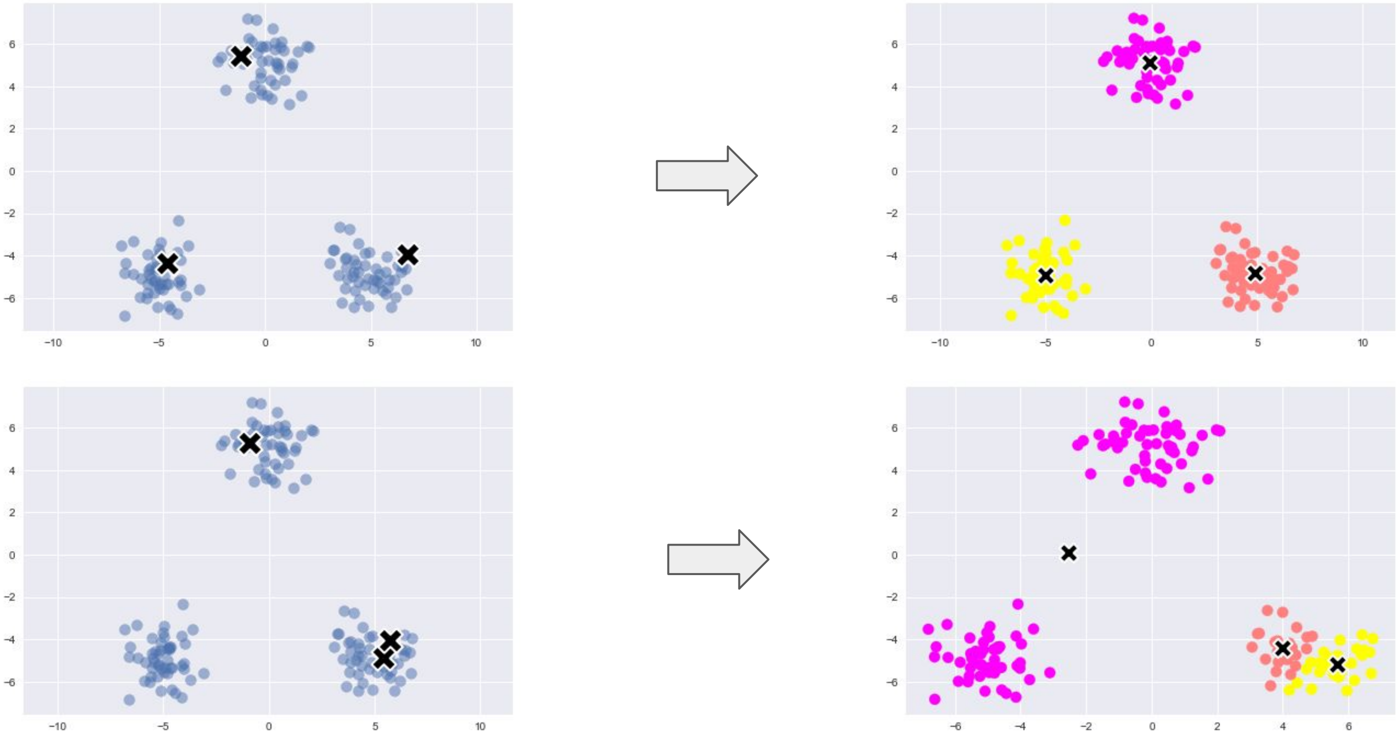
クラスタの再割り当て



中心の更新

k平均法(k-means)

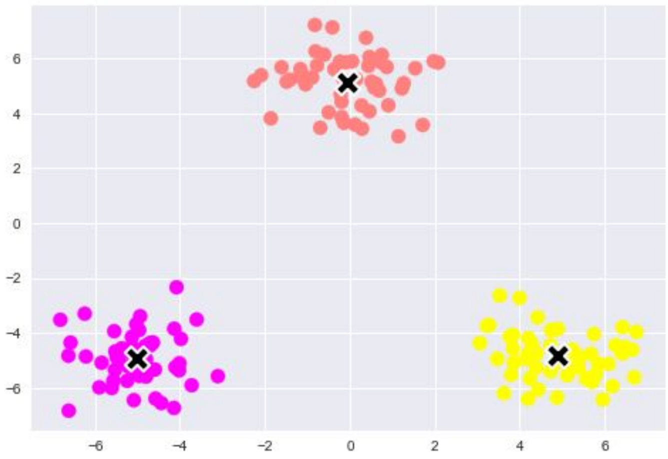
- 中心の初期値を変えるとクラスタリング結果も変わりうる



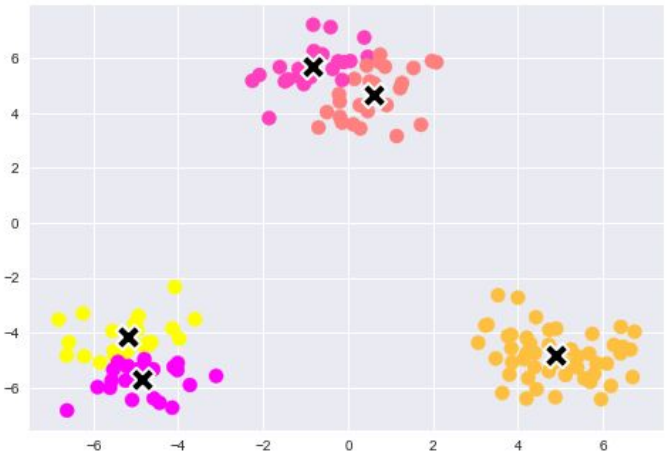
k平均法(k-means)

- kの値を変えるとクラスタリング結果も変わる

k=3



k=5



Numpyによる実装

講師と一緒に実装と動作を確認してみましょう。

Appendix

参考文献

- 仕事ではじめる機械学習 (有賀康顕)
- 多変量解析入門 (小西貞則)
- http://scikit-learn.org/stable/
- 深層学習 (岡谷貴之)
- 深層学習 (Ian goodflow)
- はじめてのパターン認識 (平井 有三)

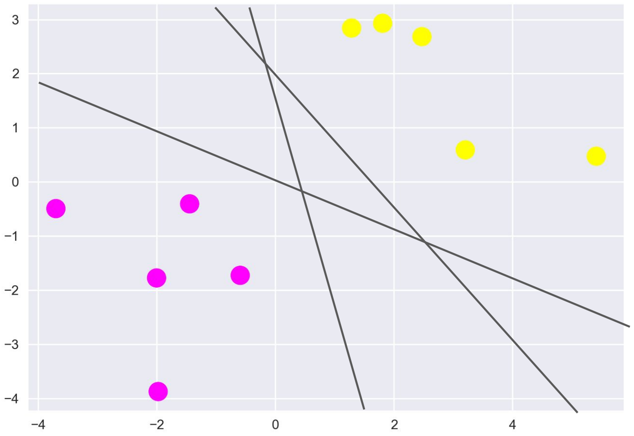
サポートベクターマシン

1. データの形式
2. アウトライン
3. サポートベクターマシン
4. マージン最大化
5. (主問題と双対問題)
6. (カーネルトリック)
7. サポートベクター

アウトラインのみ

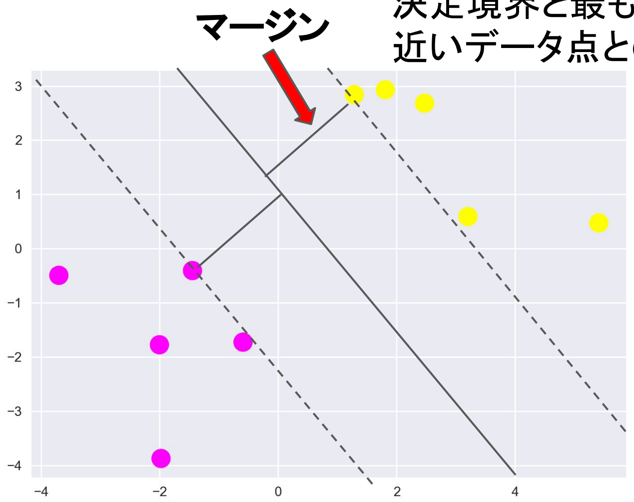
サポートベクターマシン(SVM)

- 2クラス分類のための機械学習手法
- マージンを最大化する決定境界(識別面)を求める



決定境界はいくつも考えられる

決定境界と最も
近いデータ点との距離

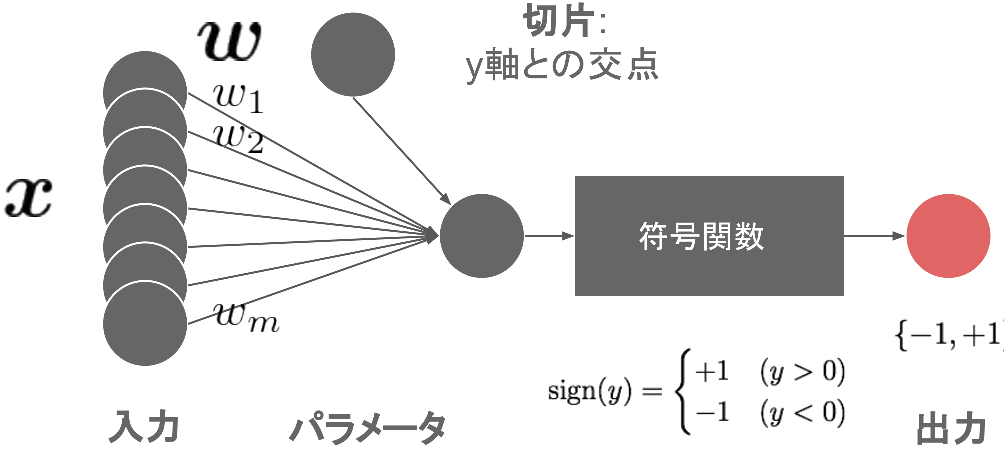


マージンが最大となる決定境界を求める

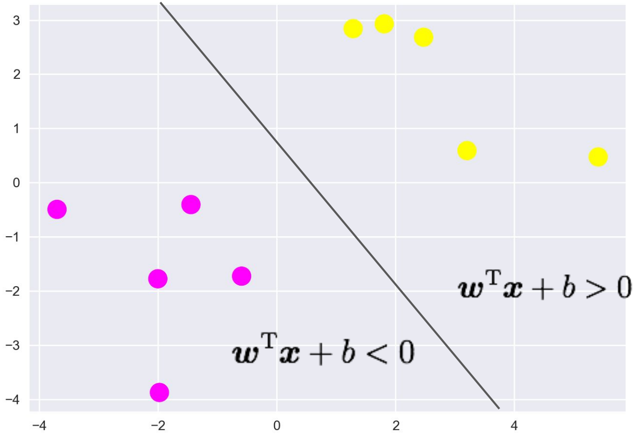
サポートベクターマシン(SVM)

●線形モデルの正負で2値分類

$$y = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = \sum_{j=1}^{m} w_j x_j + b$$



$$\mathrm{sign}(y) = \begin{cases} +1 & (y > 0) \\ -1 & (y < 0) \end{cases}$$



決定境界

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0$$

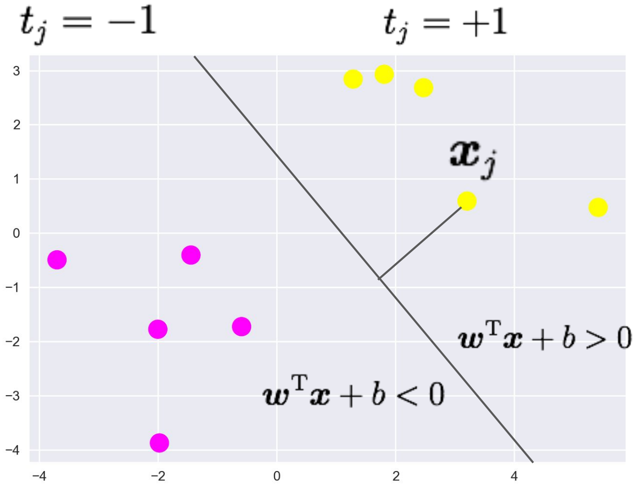
目的関数の導出

各点と決定境界との距離は、点と直線との距離の公式から

$$\frac{|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b|}{||\boldsymbol{w}||} = \frac{t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b)}{||\boldsymbol{w}||}$$

マージンとは決定境界と最も距離の近い点との距離なので、

$$\min_i \frac{t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b)}{||\boldsymbol{w}||}$$



決定境界

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = 0$$

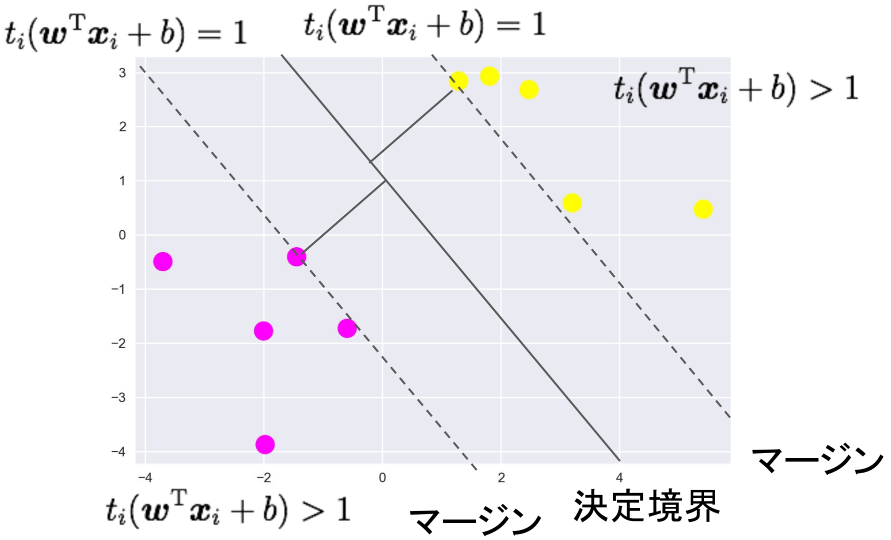
目的関数の導出

SVMはマージンを最大化することを目標なので目的関数は、

$$\max_{\boldsymbol{w},b}[\min_i \frac{t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b)}{||\boldsymbol{w}||}]$$

マージン上の点において、 $t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) = 1$ が成り立つとすると、すべての点に対して、 $t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1$ が成り立ち、目的関数は以下のようになる。

$$\max_{\boldsymbol{w},b} \frac{1}{||\boldsymbol{w}||}$$



SVMの主問題

・主問題の目的関数と制約条件

$$\min_{\boldsymbol{w},b} \frac{1}{2}||\boldsymbol{w}||^2$$

$$t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) \geq 1 \qquad (i = 1, 2, ..., n)$$

上の最適化問題をラグランジュ未定乗数法で解くことを考える

ラグランジュ未定乗数法

- 制約付き最適化問題を解くための手法

制約 $g_i(\boldsymbol{x}) \geq 0(i = 1, 2, ..., n)$のもとで、$f(\boldsymbol{x})$が最小となる　は$\boldsymbol{x}$、

変数 $\lambda_i \geq 0(i = 1, 2, ..., n)$を用いて新たに定義したラグランジュ関数

$$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^{n} \lambda_i g(\boldsymbol{x})$$

において

$$\frac{\partial L}{\partial x_j} = 0 \ (j = 1, 2, ..., m)$$

を満たす

KKT条件

- 制約付き最適化問題において最適解が満たす条件

制約 $g_i(\boldsymbol{x}) \geq 0 (i = 1, 2, ..., n)$のもとで、$f(\boldsymbol{x})$が最小となる $\boldsymbol{x}^*$は、以下の条件を満たす

$$\frac{\partial L}{\partial x_j}|_{\boldsymbol{x}^*} = 0 \ (j = 1, 2, ..., m)$$

$$g_i(\boldsymbol{x}^*) \leq 0, \ \lambda_i \geq 0, \ \lambda_i g_i(\boldsymbol{x}^*) = 0 \ (i = 1, 2, ..., n)$$

（Lはラグランジュ関数、λはラグランジュ乗数である）

ラグランジュ未定乗数法

・ラグランジュ未定乗数法を適用

ラグランジュ関数

$$L(\boldsymbol{w}, b, \boldsymbol{a}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} a_i \left(t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1\right)$$

$a_i \geq 0(i = 1, 2, ..., n)$ はラグランジュ乗数である。
そして、最小となる $\boldsymbol{w}, b$ は以下を満たす。

$$\frac{\partial L}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} a_i t_i \boldsymbol{x}_i = \mathbf{0}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{n} a_i t_i = 0$$

$$\boldsymbol{w} = \sum_{i=1}^{n} a_i t_i \boldsymbol{x}_i$$

$$\sum_{i=1}^{n} a_i t_i = 0$$

双対問題

- ラグランジュ関数から $\boldsymbol{w}, b$ を消去

$$\tilde{L} = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} a_i \left(t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1\right)$$

$$= \frac{1}{2}(\sum_{i=1}^{n} a_i t_i \boldsymbol{x}_i)^{\mathrm{T}}(\sum_{j=1}^{n} a_j t_j \boldsymbol{x}_j) - \sum_{i=1}^{n} a_i t_i(\sum_{j=1}^{n} a_j t_j \boldsymbol{x}_j)^{\mathrm{T}}\boldsymbol{x}_i - b\sum_{i=1}^{n} a_i t_i + \sum_{i=1}^{n} a_i$$

$$= \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j t_i t_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$$

双対問題

- 双対問題の目的関数と制約条件

$$\max_{\boldsymbol{a}} \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j t_i t_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$$

$$\sum_{i=1}^{n} a_i t_i = 0$$

$$a_i \geq 0 \qquad (i = 1, 2, ..., n)$$

主問題と双対問題

- 主問題の最適解と双対問題の最適解は一対一対応

$$\boldsymbol{w} = \sum_{i=1}^{n} a_i t_i \boldsymbol{x}_i$$

$$b = \frac{1}{t_i} - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i$$

（KKT条件の相補性条件 $a_i \left(t_i(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i + b) - 1 \right) = 0$ より

$a_i > 0$ となる i に対して ）

予測

・SVMの決定関数

$$y(\boldsymbol{x}) = \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b = \sum_{i=1}^{n} a_i t_i \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} + b$$

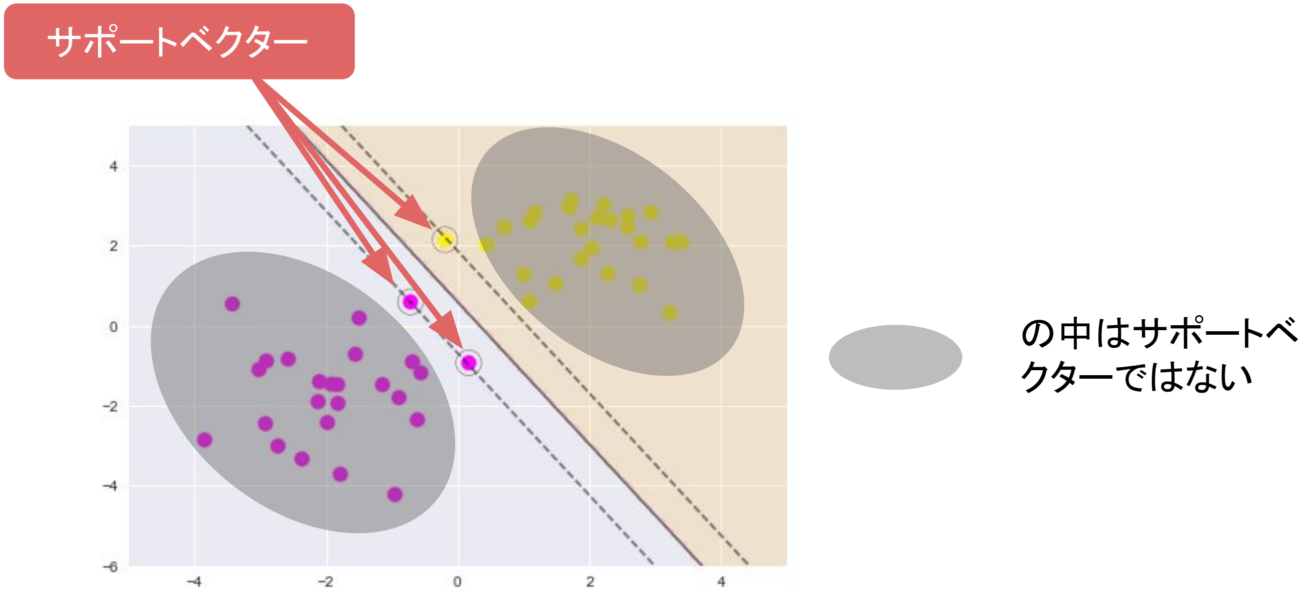
KKT条件の相補性条件 $a_i\left(t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) - 1\right) = 0$ から、

・ $t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b - 1) > 0$ のとき、つまりマージンの外側のデータでは $a_i = 0$ となり予測に影響を与えない。

・ $t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b - 1) = 0$ のとき、つまりマージン上にあるデータでは $a_i > 0$ となり、予測に影響を与える。(サポートベクター)

サポートベクター

- 分離超平面を構成する学習データは、サポートベクターだけで残りのデータは不要

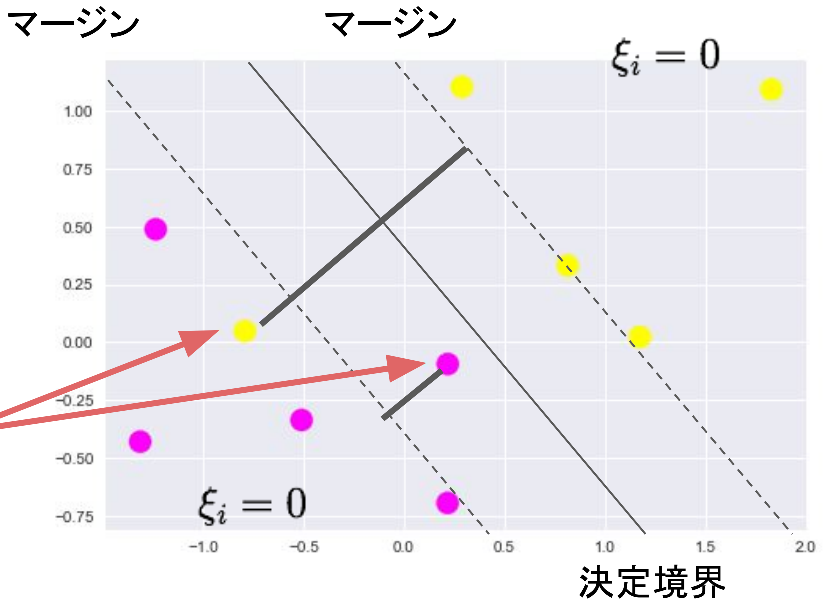


ソフトマージンSVM

- サンプルを線形分離できないとき
- 誤差を許容し、誤差に対してペナルティを与える

マージン内に入るデータや誤分類されたデータに対して誤差を表す変数
ξ_i を導入する

$$\xi_i = 1 - t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b) > 0$$



ソフトマージンSVMの目的関数と制約条件

ソフトマージンSVMの目的関数と制約条件は以下のようになる

$$\underbrace{\frac{1}{2}||\boldsymbol{w}||^2}_{\text{マージンの大きさ}} + C \underbrace{\sum_{i=1}^{n} \xi_i}_{\text{誤差に対するペナルティ}}$$

マージンの大きさ

誤差に対するペナルティ

トレードオフを制御するパラメータ

$$t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b) \geq 1 - \xi_i, \qquad \xi_i \geq 0$$

双対表現

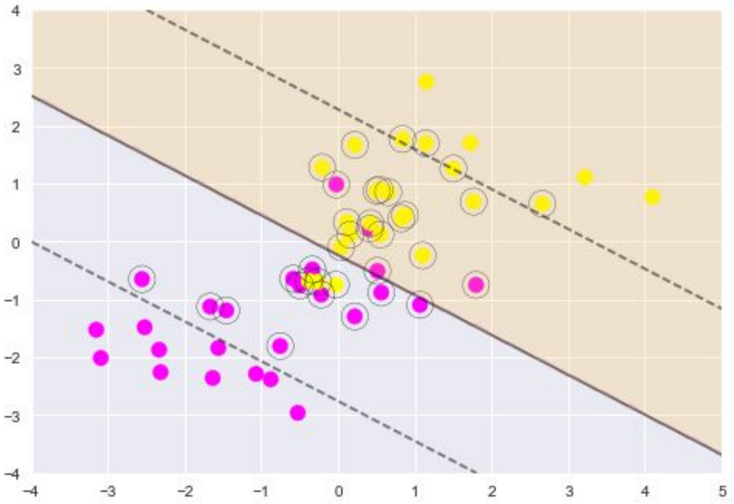
$$\max_{\boldsymbol{a}} \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j t_i t_j \boldsymbol{x}_i^{\mathrm{T}} \boldsymbol{x}_j$$

$$\sum_{i=1}^{n} a_i t_i = 0$$

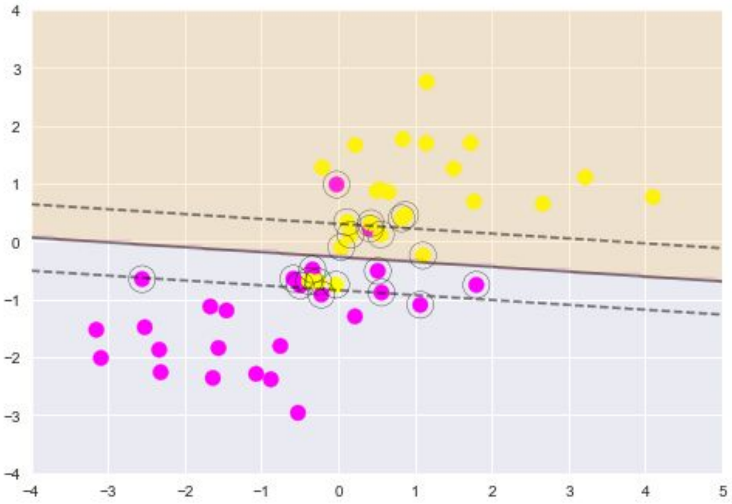
$$0 \leq a_i \leq C \qquad (i = 1, 2, ..., n)$$

ソフトマージンSVM

- 線形分離できない場合でも対応
- パラメータCの大小で決定境界が変化



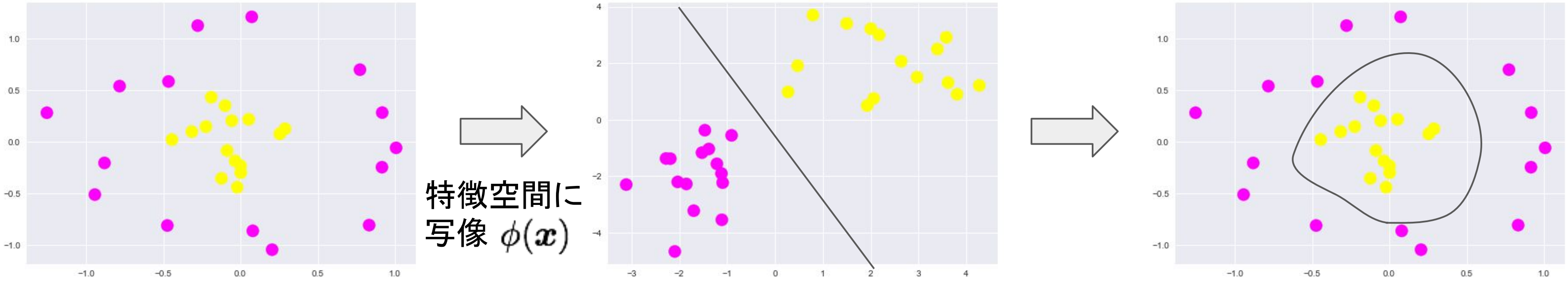
Cが小さい時（誤差をより許容する）



Cが大きい時（誤差をあまり許容しない）

非線形分離

- 線形分離できないとき
- 特徴空間に写像し、その空間で線形に分離する



線形に分離できない

特徴空間上では
線形に分離可能

もとの空間上では
非線形な分離

カーネルトリック

目的関数は以下のように変わる

$$\max_{\boldsymbol{a}} \sum_{i=1}^{n} a_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_i a_j t_i t_j \underline{\phi(\boldsymbol{x}_i)^{\mathrm{T}} \phi(\boldsymbol{x}_j)}$$

ここのみ変化

カーネルトリック

・カーネル関数　$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)^{\mathrm{T}} \phi(\boldsymbol{x}_j)$

・高次元ベクトルの内積をスカラー関数で表現

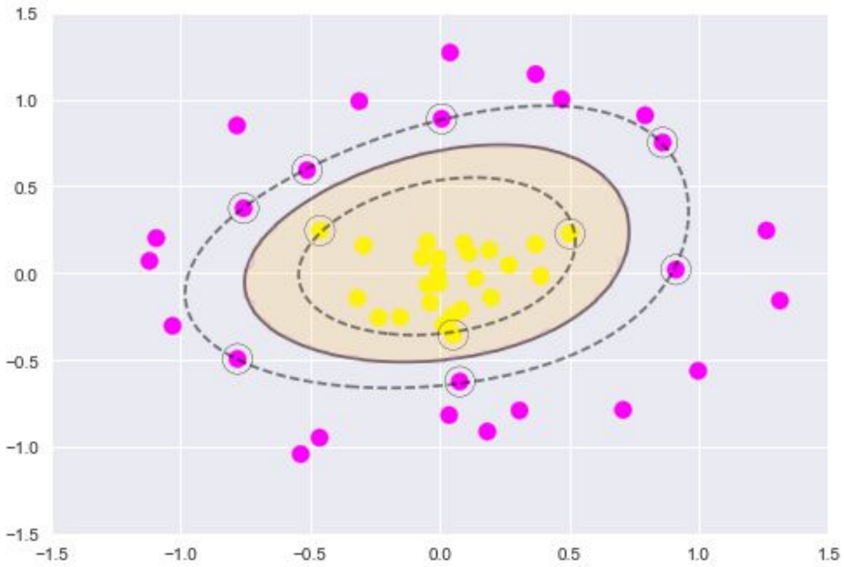
・特徴空間が高次元でも計算コストを抑えられる

非線形カーネルを用いたSVM

- 非線形な分離が可能

放射基底関数カーネル（RBFカーネル・ガウシアンカーネル）を用いる

$$k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||^2}{2\sigma^2}\right)$$



合成関数

$$\sigma(x) = \frac{1}{1 + \exp(-ax)}$$

$$g_1(x) = -ax \quad g_2(g_1(x)) = \exp(g_1(x)) + 1 \quad g_3(g_2(x)) = \frac{1}{g_2(x)}$$

合成関数で表現

$$\sigma(x) = g_3(g_2(g_1(x)))$$

合成関数の微分は
それぞれの微分の積で表現可能

$$\frac{\partial \sigma(x)}{\partial x} = \frac{\partial g_3(g_2(g_1(x)))}{\partial g_2(g_1(x))} \cdot \frac{\partial g_2(g_1(x))}{\partial g_1(x)} \cdot \frac{\partial g_1(x)}{\partial x}$$

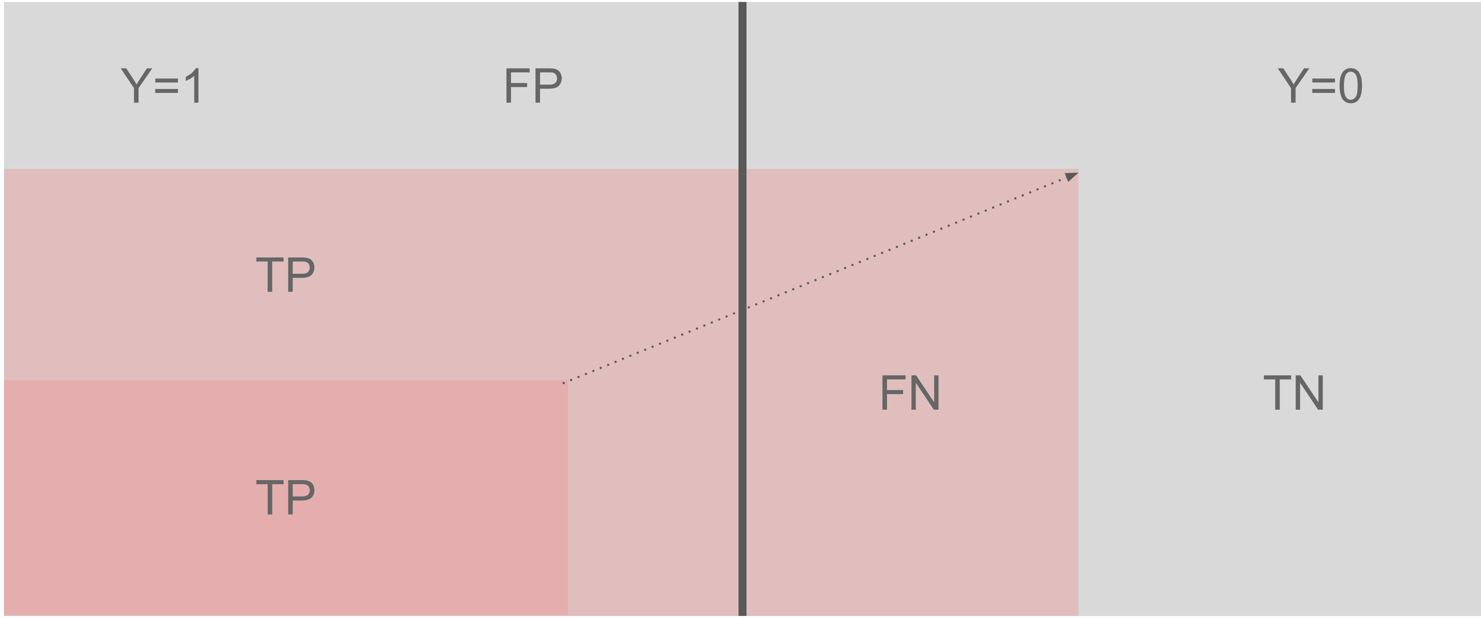
連鎖律 (chain rule)

ニューラルネットワークやディープラーニングで
頻出するので必ず覚えておく

分類の評価方法

(precision 高)
誤りが(多少)多くても、
抜け漏れを少なくしたい場合

$$\frac{TP}{TP+FP}$$



(recall 高)
見逃しが多くても、
正確な予測をしたい場合

$$\frac{TP}{TP+FN}$$