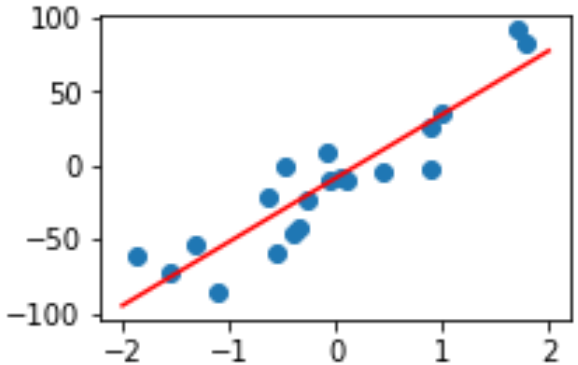
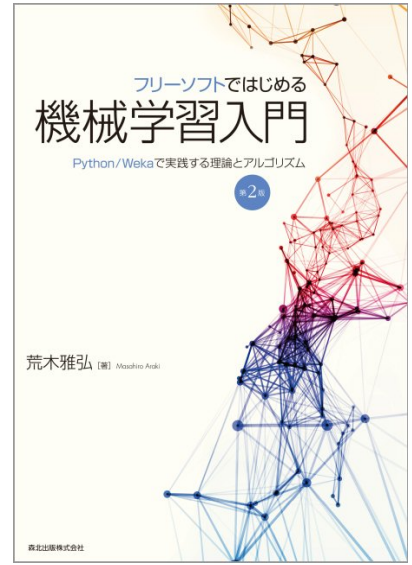
6. 回帰



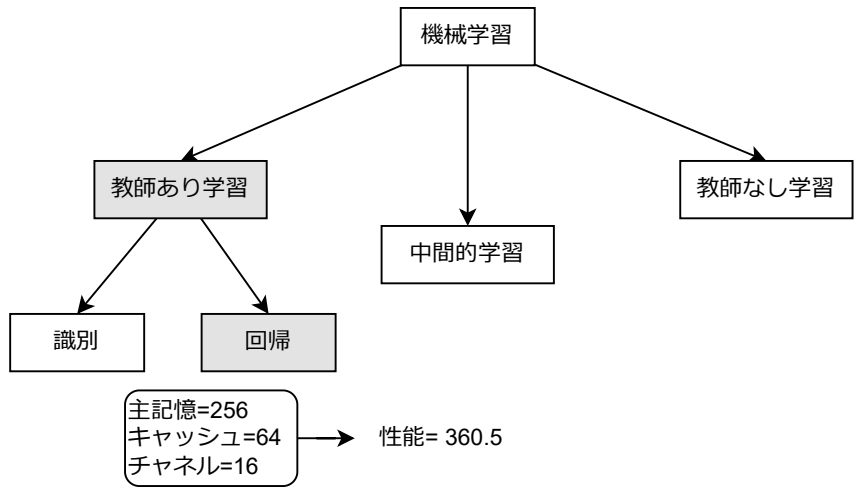
- 6.1 数値特徴に対する「教師あり・回帰」問題の定義
- 6.2 線形回帰
- 6.3 回帰モデルの評価
- 6.4 正則化
- 6.5 バイアス-分散のトレードオフ
- 6.6 回帰木
- 6.7 モデル木



- 荒木雅弘:『フリーソフトではじめる機械学習入門(第2版)』(森北出版, 2018年)
- スライドとJupyter notebook
- サポートページ

6. 回帰

- 問題設定
  - 教師あり学習
  - 数値入力 $\rightarrow$ 数値出力



6.1 数値特徴に対する「教師あり・回帰」問題の定義

- 回帰のデータ
  - 特徴ベクトル $\boldsymbol{x}$ と正解情報 y のペア

$$\{(\boldsymbol{x}_i, y_i)\}, \quad i = 1, \ldots, N$$

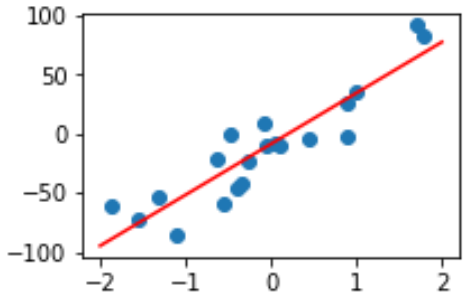
  - $\boldsymbol{x}$ は数値を要素とする d 次元ベクトル、y は数値

$$\boldsymbol{x}_i = (x_{i1}, \ldots, x_{id})^T, \quad y \in \mathbb{R}$$

  - 回帰問題の正解情報をターゲットとよぶこともある
- 回帰問題は関数 $y = \hat{c}(\boldsymbol{x})$ を求める問題と見なせる
  - 未知データに対して予測精度の高い関数を求めたい
  - どの特徴が出力値に対してどのような影響を及ぼしているかを知りたい

6.2 線形回帰 (1/4)

- 目標:学習データに対してなるべく誤差の少ない直線を求める



- 問題の定義
  - 入力 $\boldsymbol{x}$ ($x_0 \equiv 1$ の $d+1$ 次元) から出力 y を求める回帰式 $\hat{c}(\boldsymbol{x})$ を1次式に限定
  - 学習データから回帰式の重み $\boldsymbol{w}$ を求める

$$\hat{c}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} = \sum_{j=0}^{d} w_j x_j$$

6.2 線形回帰（2/4）

- 最小二乗法による重みの推定
  - $E(\boldsymbol{w})$: 誤差の二乗和を損失関数（最小化の対象）として設定する
  - $\boldsymbol{X}$: パターン行列（特徴ベクトルを転置して行方向に並べたもの。N行$d+1$列）
  - $\boldsymbol{y}$: 正解ベクトル（N 次元の列ベクトル）
  - $\boldsymbol{w}$: 重みベクトル（$d+1$ 次元の列ベクトル）

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{N}(y_i - \hat{c}(\boldsymbol{x}_i))^2 = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

  - $E(\boldsymbol{w})$ を $\boldsymbol{w}$ で偏微分した値を0と置いて、損失関数が極小となる$\boldsymbol{w}$を求める

$$\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) = 0 \Leftrightarrow \boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} \Leftrightarrow \boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

6.2 線形回帰 (3/4)

- 線形回帰の精度向上
  - 基底関数 $\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_0(\boldsymbol{x}), \ldots, \phi_b(\boldsymbol{x}))$ を考える

$$\hat{c}(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) = \sum_{j=0}^{b} w_j \phi_j(\boldsymbol{x})$$

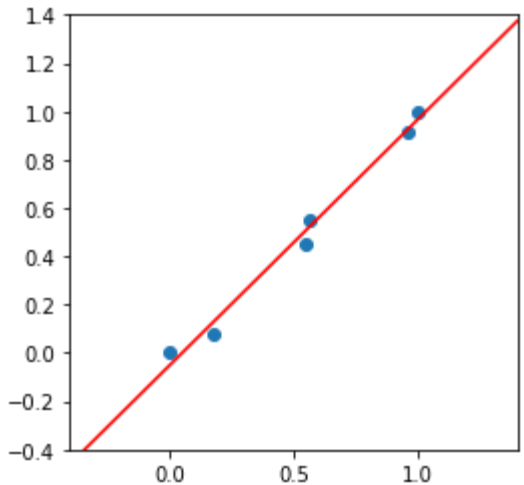
  - 基底関数の例
    - xが1次元のとき: $\boldsymbol{\phi}(x) = (1, x, x^2, \ldots, x^b)$

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=b)
X2 = poly.fit_transform(X)
```

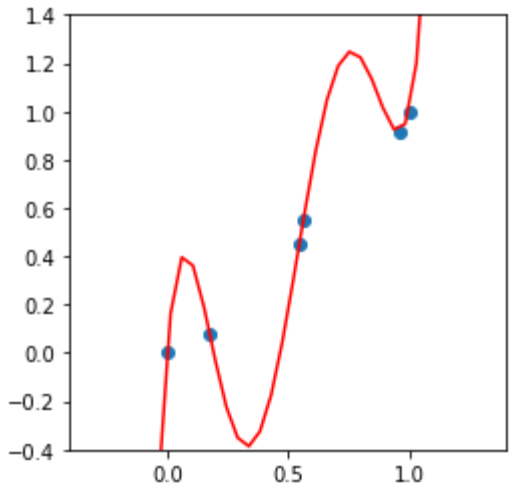
    - $\boldsymbol{x}$が2次元(x_1, x_2)のとき(`degree=2`): $\boldsymbol{\phi}(\boldsymbol{x}) = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2)$
  - 重み $\boldsymbol{w}$ が線形であれば、最小二乗法が適用可能

6.2 線形回帰（4/4）

- 基底関数を用いたときの問題点
  - 汎化性能の低下



(a) 基底関数を用いずに1次式で線形回帰



(b) 基底関数を用いて5次式で線形回帰

6.3 回帰モデルの評価 (1/2)

- 回帰モデルの評価法(1)
  - 平均二乗誤差 (MSE:Mean Squared Error)
    - 手法間の評価に有効

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{c}(\boldsymbol{x}_i))^2$$

  - 相関係数
    - 出力と正解とがどの程度似ているかを-1から1までの値で表す

$$R = \frac{\sum_{i=1}^{N}(\hat{c}(\boldsymbol{x}_i) - \tilde{c})(y_i - \tilde{y})}{\sqrt{\sum_{i=1}^{N}(\hat{c}(\boldsymbol{x}_i) - \tilde{c})^2}\sqrt{\sum_{i=1}^{N}(y_i - \tilde{y})^2}}$$

$\tilde{c}$: $\hat{c}(\boldsymbol{x}_i)$の平均, $\tilde{y}$: yの平均

6.3 回帰モデルの評価 (2/2)

- 回帰モデルの評価法(2)
  - 決定係数
    - 「正解と予測の差の二乗」と「正解の分散」との比を1から引いたもの
    - 相関係数 R の二乗と等しい

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{c}(\boldsymbol{x}_i))^2}{\sum_{i=1}^{N}(y_i - \tilde{y})^2}$$

6.4 正則化 (1/4)

- 正則化の考え方
  - 誤差関数に正則化項を導入 $\rightarrow$ 複雑な重み $\boldsymbol{w}$ (過学習)の回避
  - L1ノルム $|\boldsymbol{w}|$: 0となる重みが多くなる
  - L2ノルム $\|\boldsymbol{w}\|^2$: 重みを0に近づける
- リッジ回帰
  - 誤差の二乗和にL2ノルム正則化項を加える

$$E(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \alpha \boldsymbol{w}^T\boldsymbol{w} \qquad \alpha : \text{正則化項の重み}$$

  - $\boldsymbol{w}$ が解析的に求まる

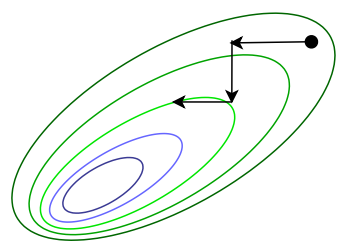
$$\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X} + \alpha \boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

6.4 正則化 (2/4)

- ラッソ回帰
  - 誤差の二乗和にL1ノルム正則化項を加える

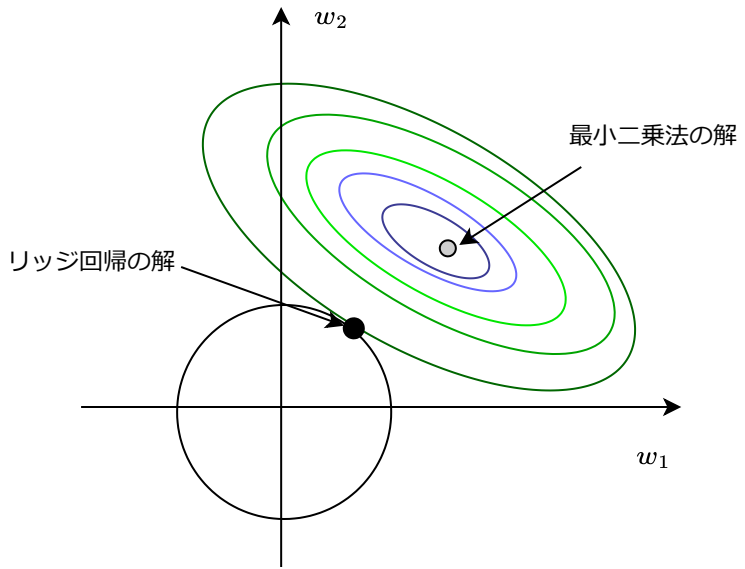
$$E(\boldsymbol{w}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}) + \alpha \sum_{j=1}^{d} |w_j|$$

  - 微分不可能な点があるため、解析的に解を求められない
  - 解法の例：coordinate descent algorithm
    - 1つの変数(軸)の値だけを誤差が減る方向に変更することを繰り返す

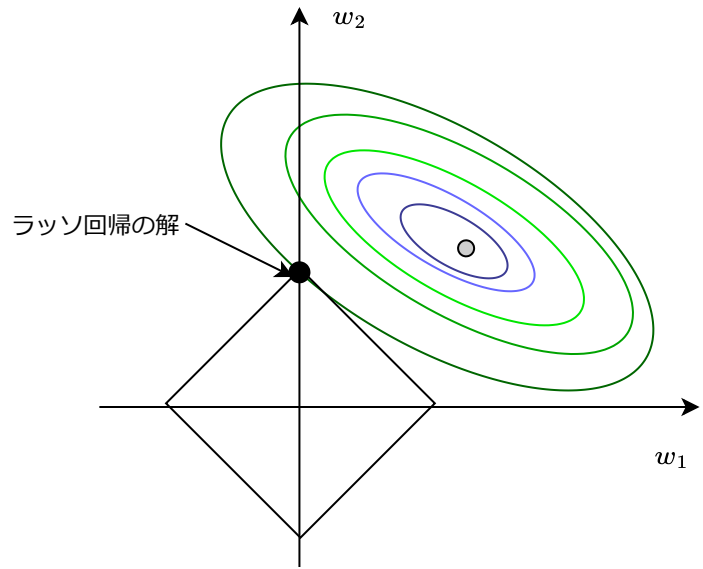


6.4 正則化 (3/4)

- リッジ回帰とラッソ回帰



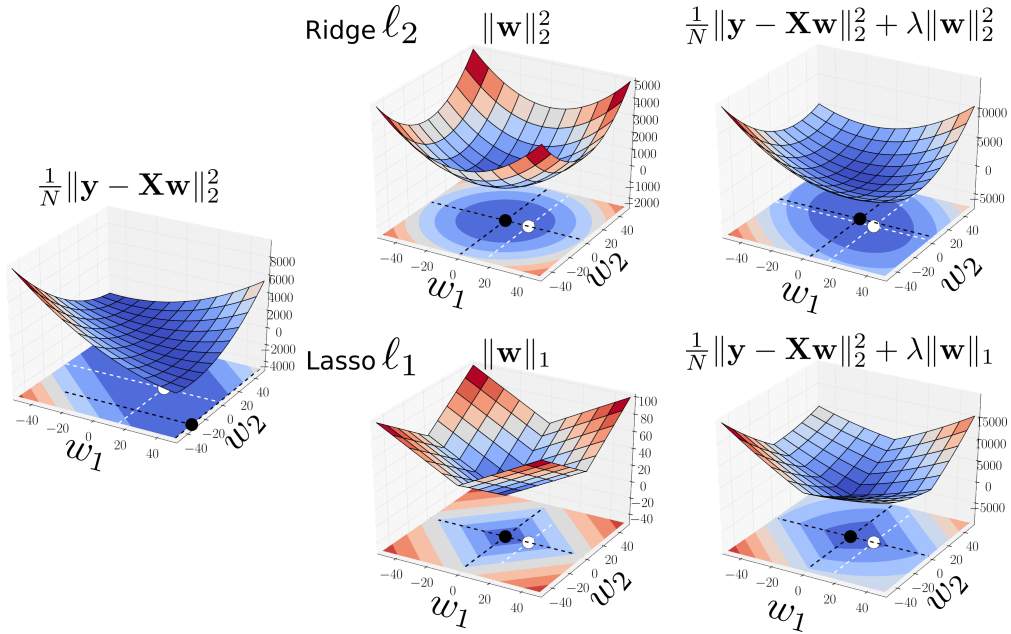
重みを0に近づけている



0となる重みを多くしている

6.4 正則化 (4/4)

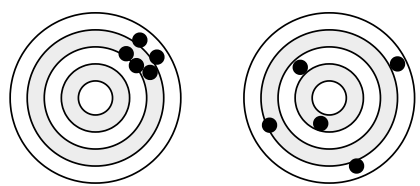
- リッジ回帰とラッソ回帰



  - https://duchesnay.github.io/pystatsml/machine_learning/linear_regression.html

6.5 バイアスー分散のトレードオフ

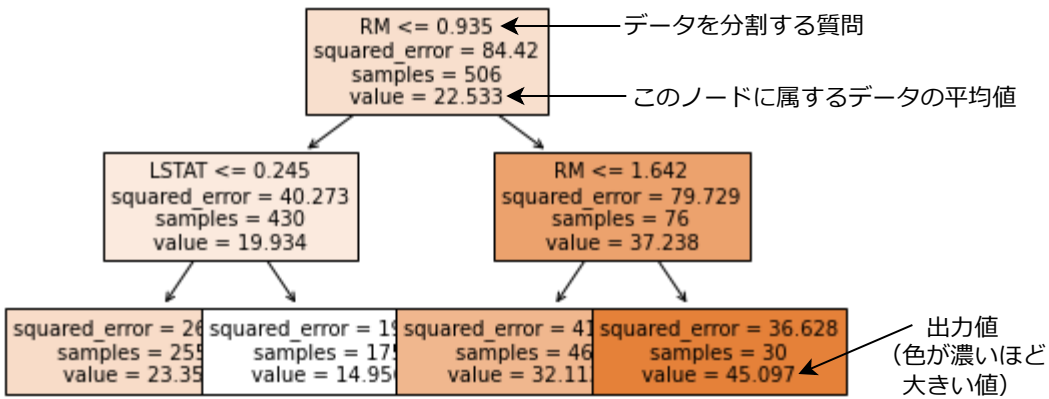
- バイアスと分散
  - バイアス：正解（＝真のモデル）からのズレ
  - 分散：求まるモデルの安定性
  - 単純なモデル
    - 正解をカバーしていないかもしれない → バイアス **大**
    - データが多少ぶれても結果は似ている → 分散 **小**
  - 複雑なモデル
    - 正解をカバーしている可能性が高い → バイアス **小**
    - データが少し違えば結果が大きく異なる → 分散 **大**（正則化項はこれを小さくしようとしている）



単純なモデル　　複雑なモデル

6.6 回帰木（1/3）

- 回帰木とは
  - 識別における決定木の考え方を回帰問題に適用
  - ターゲット値の分散が小さくなるように分割



6.6 回帰木（2/3）

- CART（classification and regression tree）
  - 木の構造を二分木に限定
  - 識別の際のデータの分類基準はジニ不純度
    - 2クラスの場合のジニ不純度

$$\mathrm{Gini}(p) = 2p(1-p) \quad p：正例の割合$$

    - クラスの出現が等確率のとき最大
  - 回帰に用いるときのデータの分類基準：ターゲット値の分散
    - 子ノードの重み付き分散和が最小＝分割後の分散の減少量が最大 となる特徴を選ぶ

6.6 回帰木 (3/3)

- CARTの特徴選択基準
  - データ D の分散

$$SS(D) = \sum_{i=1}^{N} (y_i - \tilde{y})^2$$

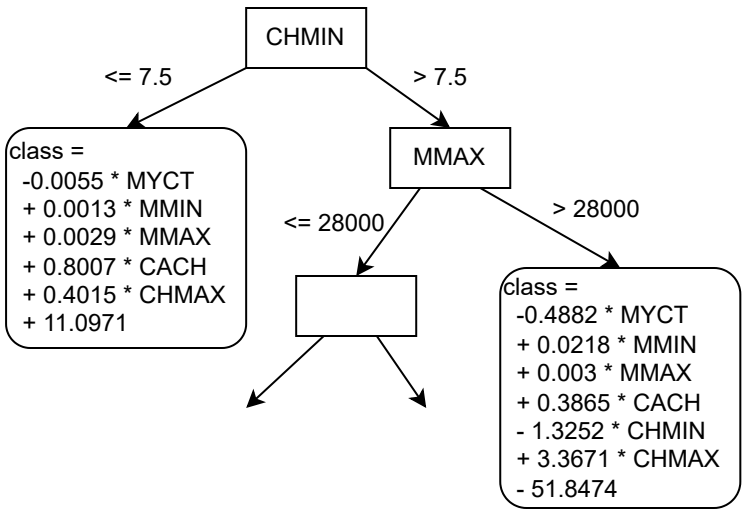
  - 分割後の分散の減少量が最大となる特徴を選ぶ

$$\Delta SS(D) = SS(D) - P_L \cdot SS(D_L) - P_R \cdot SS(D_R)$$

    - D_L：左部分木のデータ
    - D_R：右部分木のデータ
    - P_L：左部分木のデータ数の割合
    - P_R：右部分木のデータ数の割合

6.7 モデル木

- モデル木とは
  - リーフを線形回帰式にした回帰木



6.8 まとめ

- 線形回帰
  - 最小二乗法で回帰関数のパラメータを求めることができる
  - 基底関数によって複雑な関数も表現可能
  - 正則化で過学習を回避
- 回帰木
  - 決定木を用いて回帰を行う
  - 振る舞いが異なるデータが混合しているときに有効
  - リーフを線形回帰式にしたものがモデル木