機械学習(4)
特徴選択とL1正則化

情報科学類　佐久間　淳

再掲

wine dataの線形回帰

- [code]
  wine_linearRegressionManyFeatures.ipynb

負の影響

正の影響

	Coefficients	Name	
1	-0.193967	volatile acidity	重要な特徴
6	-0.107356	total sulfur dioxide	
4	-0.088183	chlorides	
8	-0.063842	pH	
2	-0.035553	citric acid	そうでもない特徴？
7	-0.033737	density	
3	0.023019	residual sugar	
0	0.043497	fixed acidity	
5	0.045606	free sulfur dioxide	
9	0.155277	sulphates	重要な特徴
10	0.294243	alcohol	

Mean squared err. 0.41676716722140794

再掲

文書を特徴ベクトルに：Bag-of-words

文書

In recent years, there has been a growing trend towards outsourcing of computational tasks with the development of cloud services. We propose two building blocks that work with FHE: a novel batch greater-than primitive, and matrix primitive for encrypted matrices

辞書(1.2万語)

ID	word
1168	batch
1169	bath
⋮	
1201	cloud
⋮	
1239	computation
1240	computational
⋮	
1172	primitive

評判分析にはどの語が重要か？
商品推薦にはどの語が重要か？

頻度は考えずに出現したかどうかを特徴とする

$$\boldsymbol{x}_i = (0, 0, \ldots, 0, 1, 0, 1, 0, \ldots, 0, 1, 0, \ldots, 0, 1, 0, \ldots)$$

Using Fully Homomorphic Encryption for Statistical Analysis of Categorical, Ordinal and Numerical Data
Wenjie Lu, Shohei Kawasaki, Jun Sakuma, IACR Cryptology ePrint Archive, Informal and Other Publications (2016)

特徴選択

- 特徴選択：目標変数の予測に無用な特徴を排除し、有用な特徴のみを使ってモデリングしたい

$$t = \underbrace{\begin{pmatrix} w_1 & w_2 & \dots & w_D \end{pmatrix}}_{\text{12000次元}} \begin{pmatrix} x_1 \\ x_2 \\ \dots \\ x_D \end{pmatrix} = \sum_{i=1}^{D} w_i x_i$$

Bag-of-words
ほとんどの特徴の値はゼロなのに全次元で計算するのは無駄が多い…

- なぜ特徴選択をするか
  - (予測時における)計算効率の向上
  - 目標変数の予測精度向上
    (無用な特徴によるモデリングは予測の質を下げる)
  - 学習結果の解釈性の向上 (どの変数が予測に効くか？)
- 特徴選択の実現方法
  1. フィルター法
  2. ラッパー法
  3. スパース性を導入する正則化の利用

フィルター法

- データ
  - D次元の特徴変数(実数値) $x_1, x_2, \ldots x_D$
  - 目標変数(実数値, 回帰の場合) t
- 各特徴変数と目標変数の関連の強さをスコアリング
  - 例:ピアソン相関係数 (例: s番目の特徴変数と目標変数の相関)

$$r_s = \frac{\sum_{i=1}^{N}(\bar{x}_s - x_{si})(\bar{t} - t_i)}{\sqrt{\sum_{i=1}^{N}(\bar{x}_s - x_{si})^2 \sum_{i=1}^{N}(\bar{t} - t_i)^2}}$$

- フィルター法:全特徴変数をスコアリングし、関連が強い変数のみを特徴として利用 $|r_1| > \theta?, |r_2| > \theta?, \ldots, |r_D| > \theta?$
- 二つ以上の変数がセットで目標値に影響を与える場合、フィルター法では選択できない

https://commons.wikimedia.org/wiki/File:Correlation_examples.png

(ナイーブな)ラッパー法

- データ
  - D次元の特徴変数(実数値) $x_1, x_2, \ldots x_D$
  - 目標変数(実数値, 回帰の場合) t
- 目標:特徴のindexを{1,2,3,…,D}としM個の特徴を選択
- 全列挙＋交差検証
  1. データを学習データとテストデータに分ける
  2. {1,2,3,…,D}のサイズMのべき集合を列挙 $S \subseteq 2^{\{x_1, x_2, \ldots, x_D\}}$
  3. べき集合の各要素について:
     1. 学習データにおいて, その集合で指定された特徴でモデルを学習
     2. そのモデルのテスト誤差を評価
  4. もっともよいテスト誤差をあたえる特徴部分集合を採用

(ナイーブな)ラッパー法

- メリット
  - テスト誤差の意味で最良の特徴選択が実現する
  - 「二つ以上の変数がセットで目標値に影響を与える場合」にも対応できる
- 問題点
  - 部分集合の候補数が特徴数Dに対して指数的に増加
  - E.g.,特徴数D=24, 選択する特徴数M=12
  - べき集合の要素数は$_{24}C_{12}$=約270万
  - 270万モデルの学習、k-fold CVによるテスト誤差評価→時間がかかりすぎる

ラッパー法:前向き貪欲探索

- 目標:特徴を{1,2,3,…,D}とし, 有用な特徴を選択
  1. データを学習データとテストデータに分ける
  2. i=1, F= {1}
  3. Fに含まれる特徴でモデルを学習し, そのモデルでテスト誤差を評価. テスト誤差をe^iとする
  4. F'=F ∪{i+1} とし, F'に含まれる特徴でモデル学習
  5. F'に含まれる特徴でモデル学習し, 学習モデルでテスト誤差を評価. テスト誤差をe^{i+1}とする
  6. $e^{i+1} < e^i$ならばF=F'とする. i=i+1として3へ
- 後ろ向きに選択された特徴を削除する探索も利用される
- 弱点:
  - テスト誤差の意味で最良な特徴集合が選択されない
  - 最後にいくつの特徴が選択されるか制御できない
  - 特徴の並び方で最終的な結果が異なる

ノルム (Lpノルム)

- (簡単にいうと)ベクトル空間上の距離の定義
- p-ノルム

$$\|\boldsymbol{x}\|_p = \left(\sum_{d=1}^{D} |x_d|^p \right)^{1/p}$$

- 1-ノルム
  - マンハッタン距離

$$\|\boldsymbol{x}\|_1 = \left(\sum_{d=1}^{D} |x_d| \right)$$

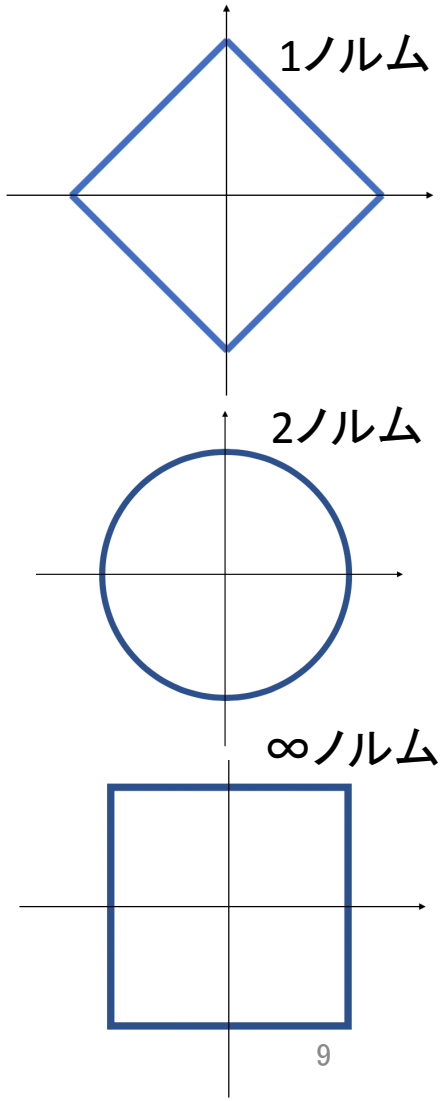
- 2-ノルム
  - ユークリッド距離

$$\|\boldsymbol{x}\|_2 = \left(\sum_{d=1}^{D} (x_d)^2 \right)^{1/2}$$

- ∞-ノルム
  - Maxノルム

$$\|\boldsymbol{x}\|_\infty = \max_d |x_d|$$

各ノルムの単位円

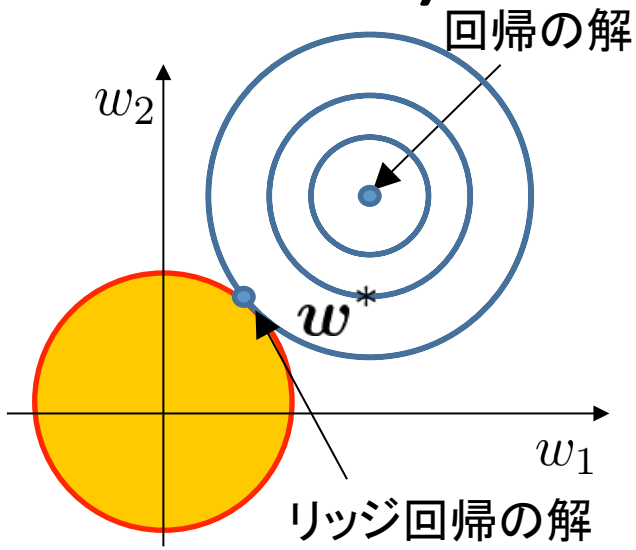


正則化の意味(L2ノルムの場合)

- 回帰 $E(\boldsymbol{w}) = \sum_{i=1}^{N} (t_i - \boldsymbol{w}^T \boldsymbol{x})^2$
- リッジ回帰

$$E(\boldsymbol{w}) = \sum_{i=1}^{N} (t_i - \boldsymbol{w}^T \boldsymbol{x})^2 + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

L2ノルムの二乗 $\|\boldsymbol{w}\|_2^2$



  - 正則化なしの回帰の解に対して、L2ノルムの意味で原点に近い解が求まる
    - E.g, w*=(2.0, 4.0) → (1.0, 2.0)
  - 正則化パラメータを大きくするほど、大きいノルムへのペナルティが大きくなる

→誤差を小さくするよりも、より小さいノルムを持つパラメータが選ばれる

→原点への引き寄せが大きくなる

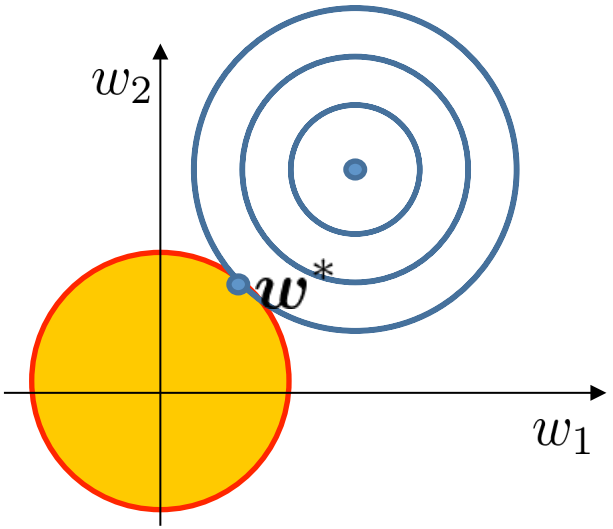
特徴選択を正則化で達成するには

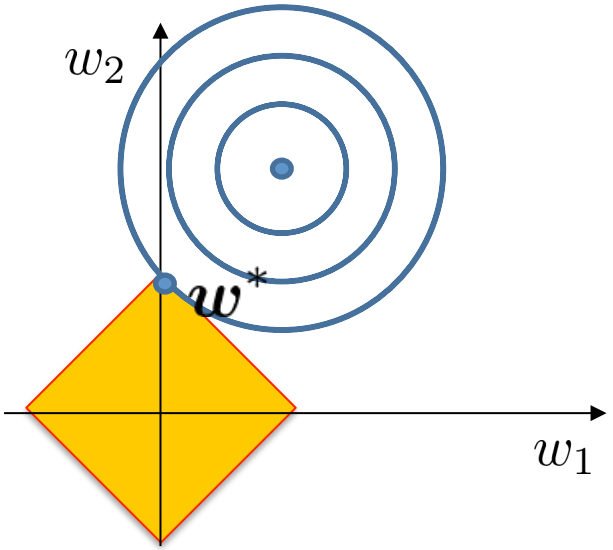
- 求まった解のうち、いくつかの次元では値が0 (軸上)に位置してほしい
  - E.g, w*=(0.1, 4.0) → (0, 3.8)
  - 1番目の特徴は予測に利用しない

  →2番目が有用な特徴として特徴選択された
- 特徴選択に適した正則化項とは？

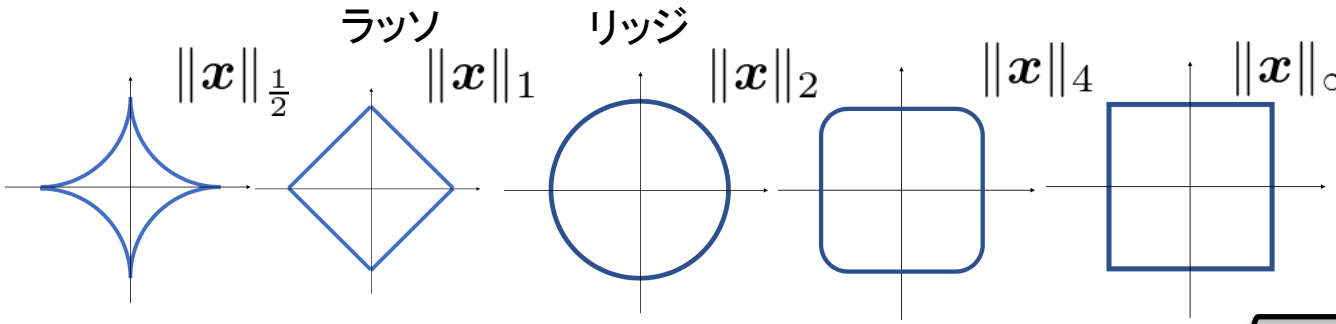
$$E(\boldsymbol{w}) = \sum_{i=1}^{N}(t_i - \boldsymbol{w}^T\boldsymbol{x})^2 + \lambda\|\boldsymbol{w}\|_p^p$$

- p=1のとき目的関数の等高線と正則化項の等高線の接する部分が「軸上」になりやすい
- このとき、係数が0になる
  - p<1でもそのような性質はあるが、凸関数でないため、最適化が困難





二乗誤差項+L1正則化項=ラッソ



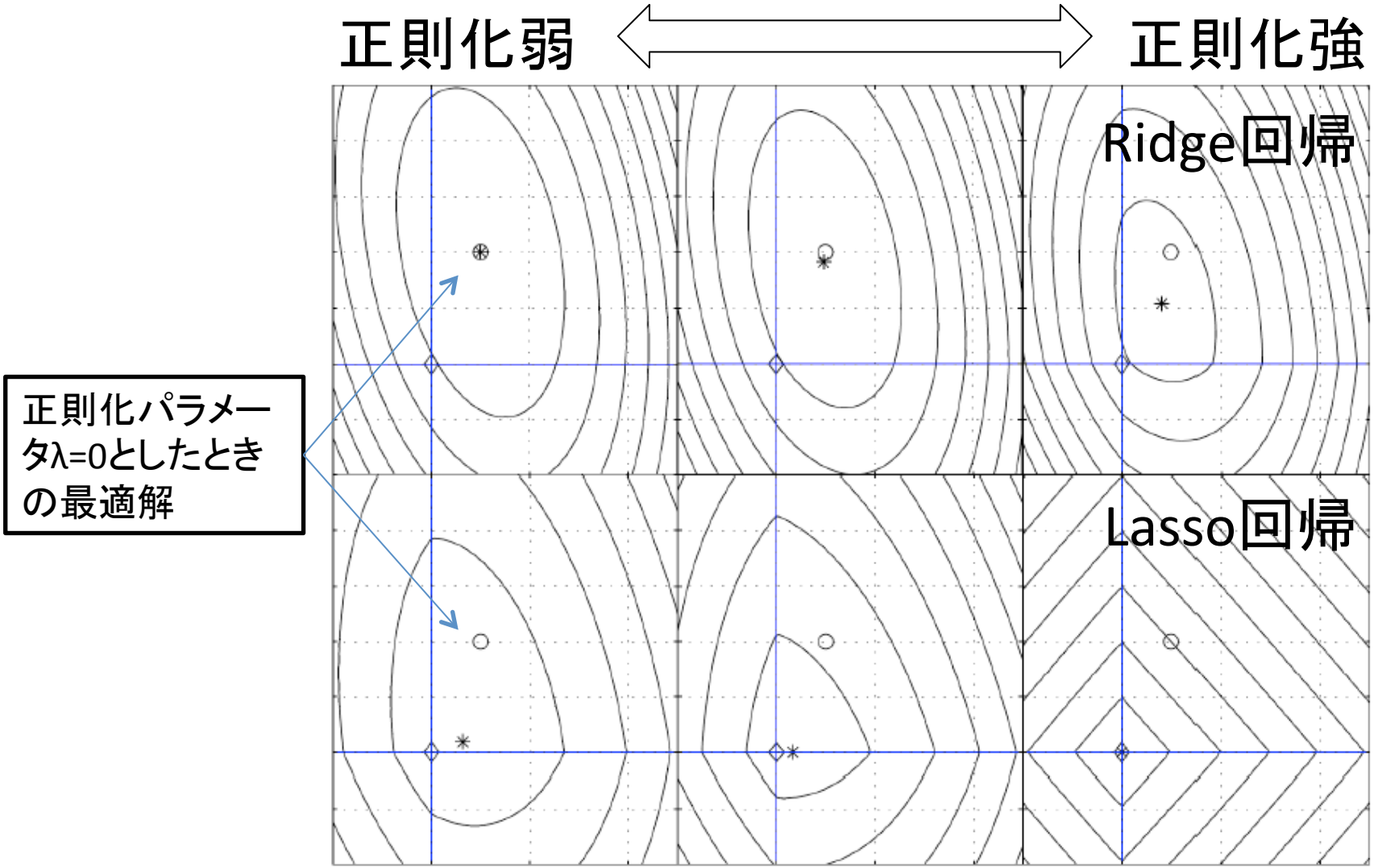
各正則化項の等高線

L1正則化項

$$E(\boldsymbol{w}) = \sum_{i=1}^{N}(t_i - \boldsymbol{w}^T\boldsymbol{x})^2 + \lambda \underbrace{\sum_{i=1}^{D}|w_i|}$$

- L1正則化: 原点からの遠さに対して線形で罰する
- 誤差項もL1正則化項も凸関数
  - 最適解は比較的求めやい、ただし
  - 原点付近で微分不可能→解析解は求まらない
  - (L2に比べ)少し解きにくい…

なぜLASSOで特徴選択できるか:
正則化による損失関数の等高線変化



www.stat.rutgers.edu/iob/bioconf07/slides/Madigan.pdf

正則化パス

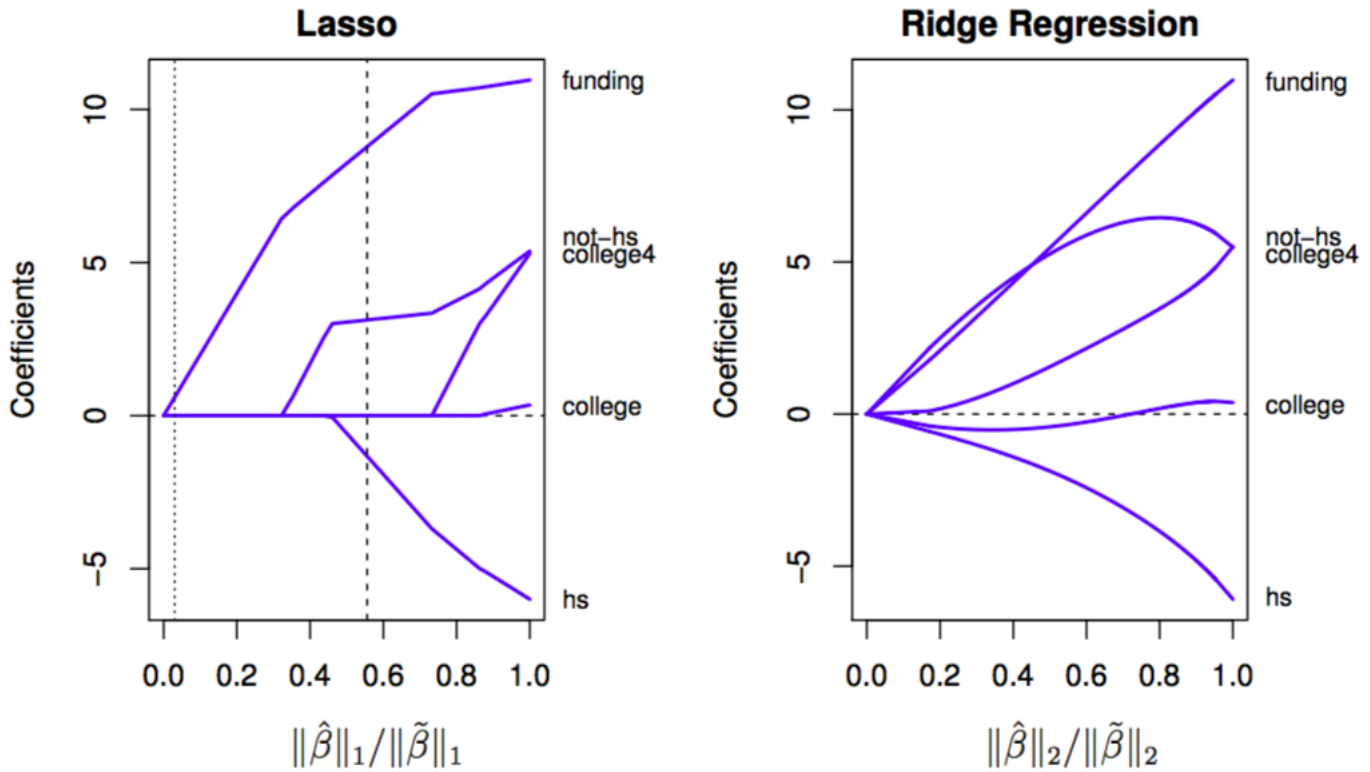
正則化係数を変化させた時に、各特徴の予測への影響度(つまり、係数の大きさ)がどう変化するか？

Table 2.1 *Crime data: Crime rate and five predictors, for $N = 50$ U.S. cities.*

city	funding	hs	not-hs	college	college4	crime rate
1	40	74	11	31	20	478
2	32	72	11	43	18	494
3	57	70	18	16	16	643
4	31	71	11	25	19	341
5	67	72	9	29	24	773
⋮	⋮	⋮	⋮	⋮		
50	66	67	26	18	16	940

特徴量:
年間警察予算,25歳以上の高校卒業率, 16-19歳の高校に行っていない人の率, 18-24歳の大学進学率, 25歳以上の4年制大学卒業率

目標値: 犯罪率



https://www.crcpress.com/Statistical-Learning-with-Sparsity-The-Lasso-and-Generalizations/Hastie-Tibshirani-Wainwright/9781498712163

問題 リッジ回帰とLASSOの比較

wine_ridgeRegression.py と wine_lasso.py を実行して以下の問いに答えなさい。Wine データを訓練事例とテスト事例に分割し、訓練事例で ridge 回帰と LASSO を学習させた。正則化パラメータ (alpha) は $2^{-16}, 2^{-15}, ..., 2^{12}$ まで変化させた。プログラムの実行結果は、正則化パラメータ、各特徴量とそれに対応する係数 (昇順にソート)、訓練誤差およびテスト誤差を表している。

1. リッジ回帰について、訓練誤差とテスト誤差を最小にする正則化パラメータはいくつか

2. ラッソについて、訓練誤差とテスト誤差を最小にする正則化パラメータはいくつか

3. リッジ回帰と LASSO について、それぞれ適切な正則化パラメータを選択せよ

4. 選択した正則化パラメータにおいて、リッジ回帰と LASSO の違いを考察しなさい

問題 リッジ回帰とLASSOの正則化パス

wine_ridgeRegression_solutionPath.py と wine_LASSO_solutionPath.py を実行して以下の問いに答えなさい。Wine データを訓練事例とテスト事例に分割し、10^6 から 10^{-3} まで正則化パラメータを変化させつつ、訓練事例で ridge 回帰と LASSO を学習させた。プログラムの実行結果は、その正則化パラメータの変化 (正則化パス) を表している。

1. リッジ回帰と LASSO の正則化パスの挙動の最大に違いは何か。その違いなぜ発生するか説明しなさい

2. 正則化パラメータを大きくした時に、その特徴の予測に与える影響 (係数の絶対値大きさ) が (多くの場合) 減少するのはなぜか

3. 正則化パラメータを大きくした時に、その特徴の予測に与える影響 (係数の絶対値の大きさ) が増加することがあるのはなぜか

特徴選択まとめ

- 多すぎる特徴は予測精度の低下や計算時間の増加を招く
- そのような特徴を絞り込むのが特徴選択
- 特徴選択手法は大きく、二つある
- フィルター法/ラッパー法
  - 様々な特徴の組み合わせを列挙して、
  - テスト誤差の意味で良い組み合わせを網羅的/実験的に選ぶ
- L1正則化
  - 利用する特徴の数をペナルティとする正則化を導入し、“特徴数をコントロールできるようにする
  - k-fold CVを使って、テスト誤差を基準に正則化パラメータをチューニングし、結果的にモデルに適切な特徴の組み合わせが利用されるようにする

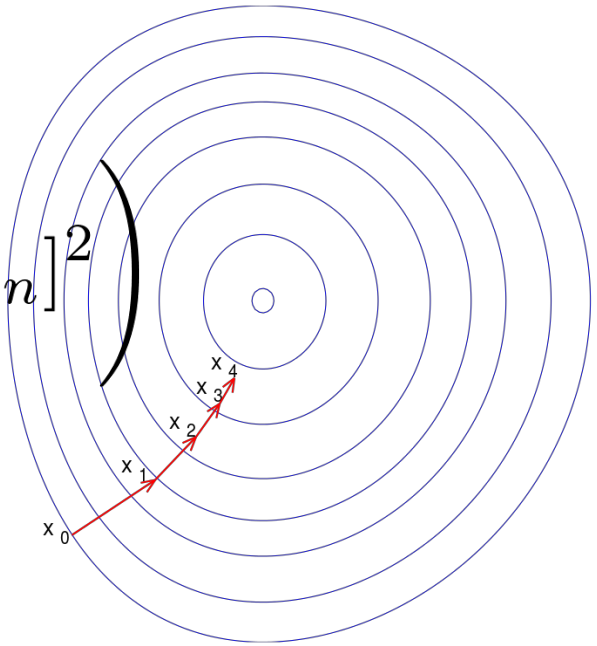
最急降下法 (勾配法, バッチ勾配法)

(gradient descent method)

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \nabla E(\mathbf{w}^{(t)})$$
$$t \leftarrow t+1 \text{ until convergence}$$

- 全事例の訓練誤差総和のwについての勾配を計算

$$\nabla E(\boldsymbol{w}^{(t)}) = \nabla \left(\sum_{i=1}^{N} [(\boldsymbol{w}^{(t)})^T \boldsymbol{x}_n - t_n]^2 \right)$$



https://commons.wikimedia.org/wiki/File:Gradient_descent.svg

- 勾配は比較的正確に計算できるが、更新あたりの計算量はサンプル数NについてO(N)
- サンプル数が非常に大きい場合には不向き

確率的勾配降下法

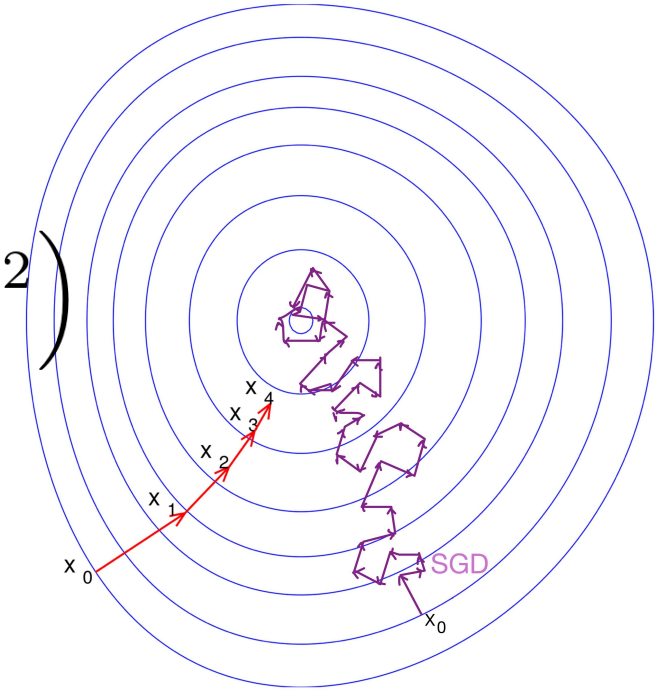
(stochastic gradient method, SDG)

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \nabla E(\mathbf{w}^{(t)})$$
$$t \leftarrow t+1 \text{ until convergence}$$

- ランダムに選んだ一つの事例の訓練誤差について勾配を計算

$$\nabla E(\boldsymbol{w}^{(t)}) = \nabla \left((\boldsymbol{w}^{(t)})^T \boldsymbol{x}_n - t_n \right)^2$$

- E(w)の勾配という意味では近似的だが、更新あたりの計算量はサンプル数Nについて$O(1)$
- 全データに対する勾配との乖離があり、最適化の過程は不安定(訓練誤差は単調減少しない)



https://commons.wikimedia.org/wiki/File:Gradient_descent.svg

ミニバッチ

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \nabla E(\mathbf{w}^{(t)})$$
$$t \leftarrow t+1 \text{ until convergence}$$

- ミニバッチ D_t
  - 全データから比較的少数のデータをランダムに選んだ集合
- ランダムに選んだミニバッチの予測誤差について勾配計算

$$\nabla E(\boldsymbol{w}^{(t)}) = \nabla \left(\sum_{(\boldsymbol{x},t) \in D_t} [(\boldsymbol{w}^{(t)})^T \boldsymbol{x} - t]^2 \right)$$

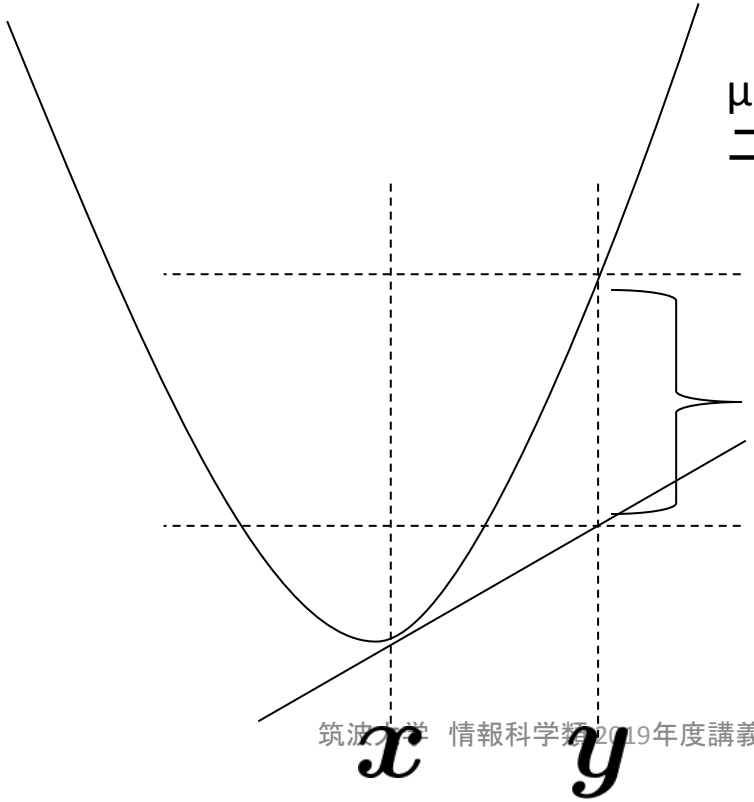
- 更新あたりの計算量はミニバッチサイズに比例
- 勾配はミニバッチについて計算するためSGDほど不安定ではない
- 並列計算器資源の有効活用
  - バッチごとに独立に勾配を計算

凸関数の特徴づけ　μ強凸

- 以下を満たすならば、f:R^D→Rはμ強凸

$$f(\boldsymbol{y}) > f(\boldsymbol{x}) + (\boldsymbol{y}-\boldsymbol{x})^T \nabla f(\boldsymbol{x}) + \frac{\mu}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$$

cf.凸関数 $f(\boldsymbol{y}) > f(\boldsymbol{x}) + (\boldsymbol{y}-\boldsymbol{x})^T \nabla f(\boldsymbol{x})$

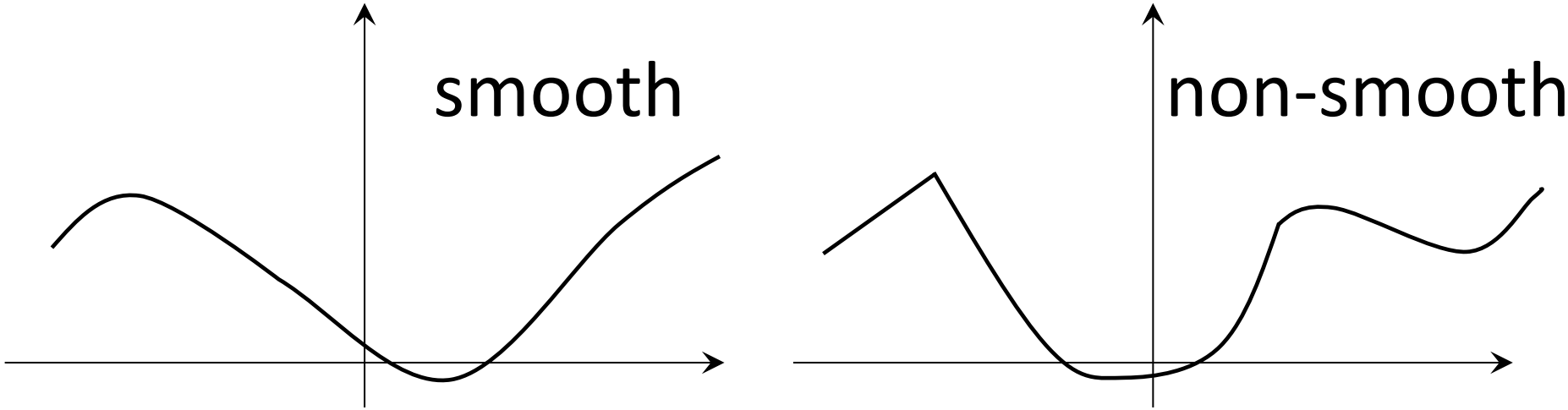


μは凸性の強さを表す
二次関数を引いてもなお凸性があるかどうか

このギャップが $\frac{\mu}{2}\|\boldsymbol{x}-\boldsymbol{y}\|_2^2$ 以上あればよい

凸関数の特徴づけ: L平滑

- 以下を満たすならば、f:R^D→RはL平滑 (L>0)
  - 勾配の変化が入力の変化の定数倍で抑えられる
    $\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_2 < L\|\boldsymbol{x} - \boldsymbol{y}\|_2$
  - 近接した点の勾配は、互いに近接している



勾配降下法の収束

定理：$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w})$ とする。また関数Eは微分可能な凸関数とする。関数EがL平滑(L>0)で、ステップサイズがη<1/Lならば以下が成立：

$$E(\boldsymbol{w}^{(t)}) - E(\boldsymbol{w}^*) < \frac{2L\|\boldsymbol{w}^{(0)} - \boldsymbol{w}^*\|_2^2}{t+4}$$

- 収束速度はステップ数tについてO(1/t)
- 誤差をε以下にするには、O(1/ε)回の更新が必要
  - (ラフに言えば)ε =0.01程度の誤差をT回更新で達成したとしたら、ε =0.001程度の誤差は10T回程度の更新で達成できる

勾配降下法の収束(強凸)

定理: $\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} E(\boldsymbol{w})$ とする。また関数Eは微分可能な凸関数とする。関数Eがμ強凸 (μ>0)かつL平滑L(>0) で、η=1/Lならば以下が成立:

$$E(\boldsymbol{w}^{(t)}) - E(\boldsymbol{w}^*) < \left(1 - \frac{\mu}{L}\right)^t [E(\boldsymbol{w}^{(0)}) - E(\boldsymbol{w}^*)]$$

- 収束速度はステップ数tについてO(c^t)
- 誤差をε以下にするには、O(log(1/ε))回の更新でよい
  - (大雑把に言えば)ε =0.01程度の誤差を2T回更新で達成したとしたら、ε =0.001程度の誤差は3T回程度の更新で達成できる
- 強凸の方がずっと収束が早い(一次収束)
  - 回帰、ridge回帰は強凸かつL平滑
  - ロジスティック回帰はL平滑だが強凸ではない(あとで出てくる)
  - Lasso, SVMはL平滑ではない(あとで出てくる)

ステップサイズパラメータ(学習率)

- 学習率の選択
  - 小さすぎると、収束は遅くなる
  - 大きすぎると、収束は早いが、最適解周辺で振動
  - 非常に大きいと、発散する
- 学習率のスケジューリング
  - 更新回数が増加するにつれて、ステップサイズを減少させる
  - 更新量が減少するにつれて、ステップサイズを減少させる
  - パラメータごとに、学習率を適応的に変化させる
- 適切な学習率を理論的に決定することは困難
- 問題ごとにチューニングする必要がある

劣勾配(sub-gradient))

- 微分不可能な項を含む目的関数をどうやって最適化する？

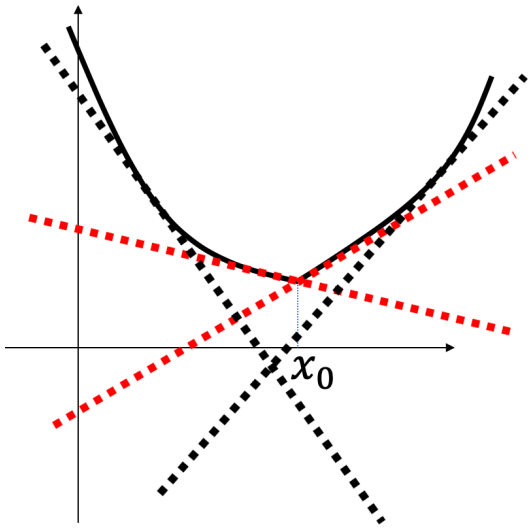
$$E(\boldsymbol{w}) = \sum_{i=1}^{N}(t_i - \boldsymbol{w}^T\boldsymbol{x})^2 + \lambda\sum_{i=1}^{D}|w_i|$$

L1正則化項は微分不可能

- 劣勾配 $\partial f(\mathbf{w}_0) = \{\mathbf{g} \in \mathbf{R}^D | \forall \mathbf{w}, f(\mathbf{w}) - f(\mathbf{w}_0) \geq (\mathbf{w} - \mathbf{w}_0)^T\mathbf{g}\}$
  - 例　$f(\mathbf{w}) = |w_1| + 2|w_2|$

$\mathbf{w}_0 = (1,0)^T$での劣微分

$$\partial f(\mathbf{w}_0) = \left\{ \begin{pmatrix} 1 \\ 2z \end{pmatrix} \middle| z \in [1,-1] \right\}$$



劣勾配降下法(sub-gradient descent)

- 最急降下法における勾配を劣勾配に置き換え

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \partial E(\mathbf{w}^{(t)})$$

- Cf. 最急降下法

$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} - \eta \nabla E(\mathbf{w}^{(t)})$$

FOBOSによるLassoの解法[Duchi09+]

- 劣勾配降下法ではwがスパースにならないことがある
- 目的関数 $E(\boldsymbol{w}) = L(\boldsymbol{w}) + r(\boldsymbol{w}) \quad \text{where } r(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|_1$
  - L:二乗誤差和(微分可能), r:L1正則化項(微分できない)
- 2stepによる解法
  1. 勾配法による損失(誤差)の更新

$$\boldsymbol{w}^{(t+\frac{1}{2})} = \boldsymbol{w}^{(t)} - \eta_t \nabla_{\boldsymbol{w}} L(\boldsymbol{w}^{(t)})$$

  2. 正則化の適用

$$\boldsymbol{w}^{(t+1)} \leftarrow \arg\min_{\boldsymbol{w}} \left\{ \frac{1}{2} \|\boldsymbol{w} - \boldsymbol{w}^{(t+\frac{1}{2})}\|^2 + \eta_{t+\frac{1}{2}} r(\boldsymbol{w}) \right\}$$

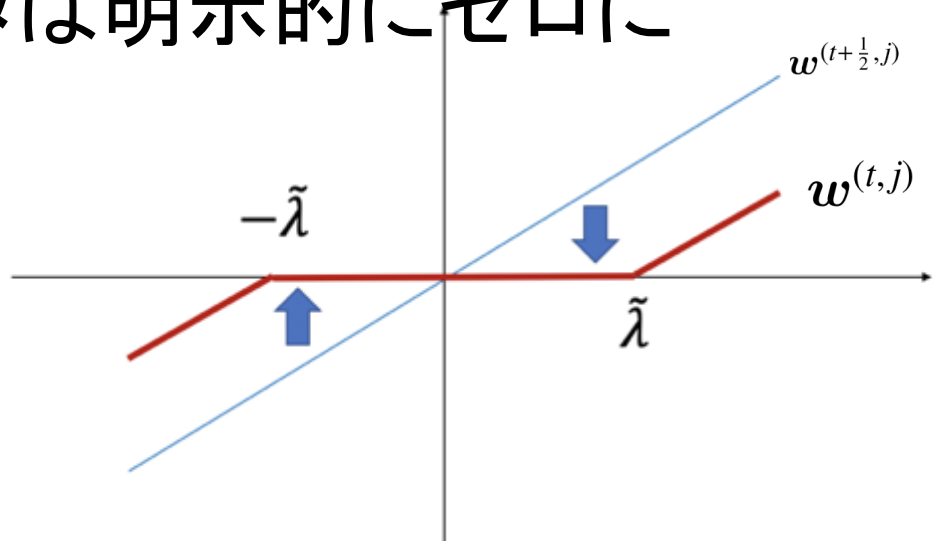
  - L1正則化の明示的な適用がありwがスパースになりやすい
  - 収束の保証がある(講義では省略)

Step 2をどう解くか？

- Soft-thresholdingと呼ばれる手法
- Proximal作用素として一般化される

$$\begin{aligned} \boldsymbol{w}^{(t+1,j)} &= \mathrm{sign}(\boldsymbol{w}^{(t+\frac{1}{2},j)}) \left[|\boldsymbol{w}^{(t+\frac{1}{2},j)}| - \tilde{\lambda} \right]_+ \text{ where } \tilde{\lambda} = \eta_{t+\frac{1}{2}} \cdot \lambda \\ &= \mathrm{sign}\left(\boldsymbol{w}^{(t,j)} - \eta_t \nabla_{\boldsymbol{w}^{(t,j)}} L(\boldsymbol{w}) \right) \left[|\boldsymbol{w}^{(t,j)} - \eta_t \nabla_{\boldsymbol{w}^{(t,j)}} L(\boldsymbol{w})| - \eta_{t+\frac{1}{2}} \cdot \lambda \right]_+ \end{aligned}$$

- 値の小さいパラメータは明示的にゼロに



Further reading
http://postd.cc/optimizing-gradient-descent/

演習 最急降下法と確率的勾配法

3.4　最急降下法と確率的最急降下法

wine_GD.ipynb　と wine_SGD.ipynb を実行して以下の問いに答えなさい。プログラムは Wine データを訓練事例とテスト事例に分割し、訓練事例で ridge 回帰を勾配降下法 (GD) と確率的勾配降下法 (SGD) によって最適化した。 いずれも複数のステップサイズパラメータ η で実行した。

1. GD ではステップサイズにかかわらすアルゴリズム停止時にはほぼ同じ誤差を達成しているが、ステップサイズによって停止までにかかるステップ数が異なるのはなぜか

2. GD では目的関数が単調減少するのに対して、SGD ではそうならないのはなぜか

3. GD は t 回目の更新における誤差と、$t+1$ 回目の更新における誤差の差分が ϵ より小さくなったときに収束したと判定し、アルゴリズムを停止するように設定されているが、SGD では GD と同じ収束判定法は使えない。それはなぜか。またどのような収束判定を使うと良いと考えられるか、アイディアを考えて説明しなさい。

演習

4.4　リッジ回帰の最急降下法と確率的最急降下法

$\boldsymbol{x}_i = \begin{pmatrix} 1 \\ x_{i1} \\ \vdots \\ x_{iD} \end{pmatrix}, \boldsymbol{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \vdots \\ \boldsymbol{x}_N^T \end{pmatrix}, \boldsymbol{t} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_N \end{pmatrix}, \boldsymbol{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_D \end{pmatrix}$ とする。事例群 $\{(\boldsymbol{x}_i, t_i)\}_{i=1}^N$ を使ってリッジ回帰による線形モデルを求めることを考える。

1. リッジ回帰の最急降下法の更新式を示せ
2. リッジ回帰の確率勾配降下法の更新式を示せ