現場で使える機械学習・データ分析
基礎講座（DAY2）
SkillUP AI

本講座の構成

DAY1	DAY2

• 機械学習概論
  • 人工知能とは
  • 機械学習とは
  • 機械学習アルゴリズムの実装とワークフロー
  • 機械学習アルゴリズム概観
• 教師あり学習の基礎
  • 線形回帰
  • ロジスティック回帰
  • 多変量モデルへの拡張
• モデルの評価指標
  • 回帰問題（MAE/MSE/RMSE）
  • 分類問題（精度/適合率/再現率/F1-score）
	• モデルの検証・正則化
	  • 訓練誤差と汎化誤差
	  • 過学習
	  • 正則化（L2/L1）
	  • ホールドアウト法・交差検証法
	• 前処理
	  • 正規化 / 標準化
	  • 無相関化 / 白色化
	• 教師あり学習の発展的トピック
	  • サポートベクターマシン

本講座の構成

DAY3	DAY4

• 前処理
  • 特徴選択
• 教師あり学習の発展的トピック
  • 木モデル
  （決定木・ランダムフォレスト）
  • ニューラルネットワーク
	• 教師あり学習の発展的トピック
	  • 深層学習
	  • k-最近傍法
	• 教師なし学習
	  • クラスタリング
	  • 特徴抽出・次元削減
	• モデルの改善
	  • ハイパーパラメータ最適化

DAY2の目次

- グループワーク
  - 通し課題の進捗共有
- モデルの検証と正則化
  - 訓練誤差と汎化誤差
  - ホールドアウト法と交差検証法
  - オーバーフィッティングと
    アンダーフィッティング
  - L2正則化、L1正則化
- 代表的な前処理
  - 正規化 / 標準化
  - 無相関化 / 白色化
- サポートベクターマシン
  - マージンの導入
  - 線形識別関数の導入
  - サポートベクターマシンの目的関数
  - カーネル法とカーネルトリック
- グループワーク
- 質疑応答

通し課題の進捗共有

- 同じ問題を解く人同士で集まり、解法やうまくいかないポイントを共有しましょう
- データの集計でわかったことは？
  - データは線形？非線形？変数間の相関関係は？
  - 欠損値はどのように対応した？
- 説明変数は何を用いた？
  - すべて用いた？自分の持つ知識を使って、新しい説明変数は作れそう？
- 今後の方針は？

モデルの検証と正則化

1. 訓練誤差と汎化誤差
2. ホールドアウト法と交差検証法
3. オーバーフィッティングとアンダーフィッティング
4. L2正則化、L1正則化

誤差のない「完璧」なモデル？

- 線形回帰やロジスティック回帰モデルを用いて、学習用データを完璧に説明できるほとんど誤差がないモデルを作ることができたとしよう
- これは本当に完璧なモデルと言えるのだろうか？
- 本当にこのまま実運用してしまって大丈夫？

機械学習の全体像を思い出してみよう

学習フェーズ

過去のデータ

機械学習

データ・タスクに対応したモデル

運用フェーズ

新たなデータ

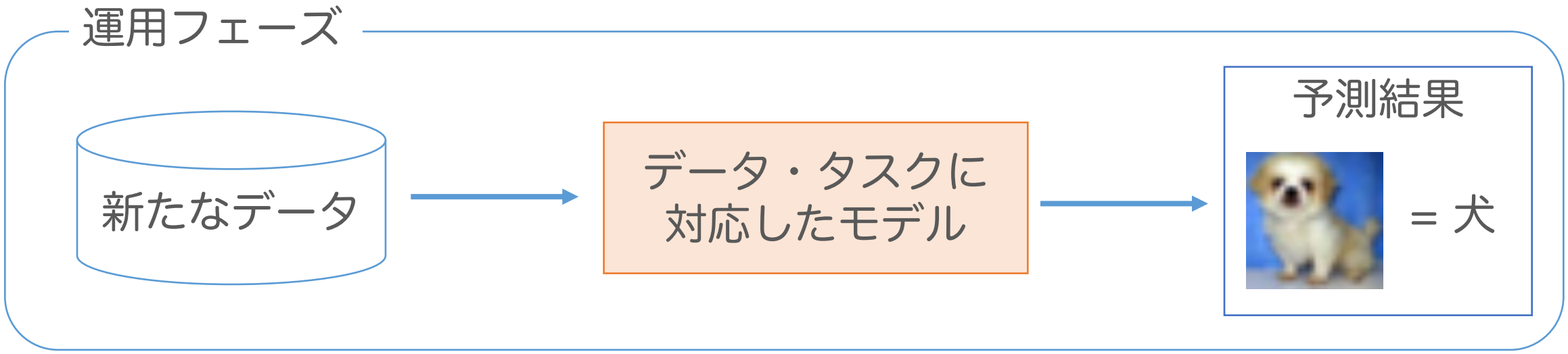
データ・タスクに対応したモデル

予測結果

＝ 犬

本当に評価すべきは「未知データ」に対する予測性能

- 本当に評価すべきは、運用フェーズでどの程度予測できるか
- 運用フェーズでは、学習に使用していない（モデルにとって）未知のデータで予測するはず
- 実運用で使えるかどうかは、未知データに対する評価指標を確認してから判断すべき！

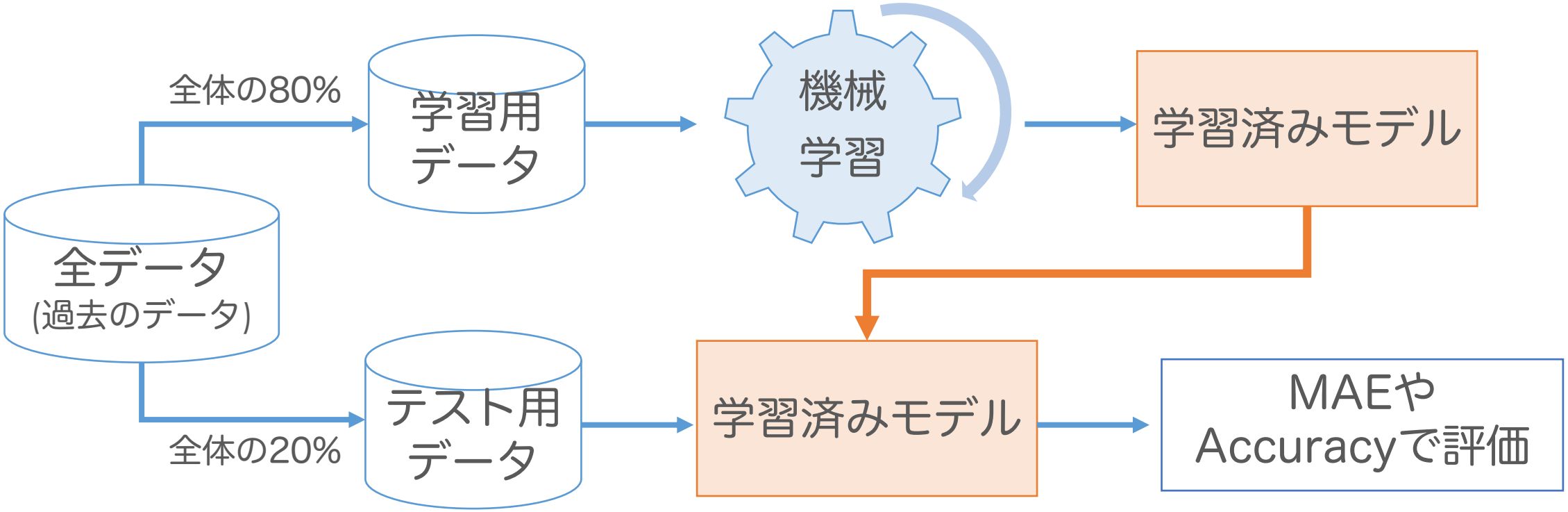


訓練誤差と汎化誤差

- 機械学習では、データに対する「誤差」を2種類にわけて考える
- 訓練誤差：学習データに対する誤差
- 汎化誤差：学習に使っていない未知データに対する誤差
  - 未知のデータに対する予測性能を汎化性能と呼ぶ
- モデルの作り込みは汎化誤差を小さくするように行うべき

汎化誤差の評価方法｜ホールドアウト法

- ホールドアウト法とは、データを事前に学習用とテスト用に分割し、テスト用データで学習済みモデルの汎化誤差を評価する方法のこと
- 実装は簡単だが、データ数が少ないと汎化誤差をうまく評価できない場合がある



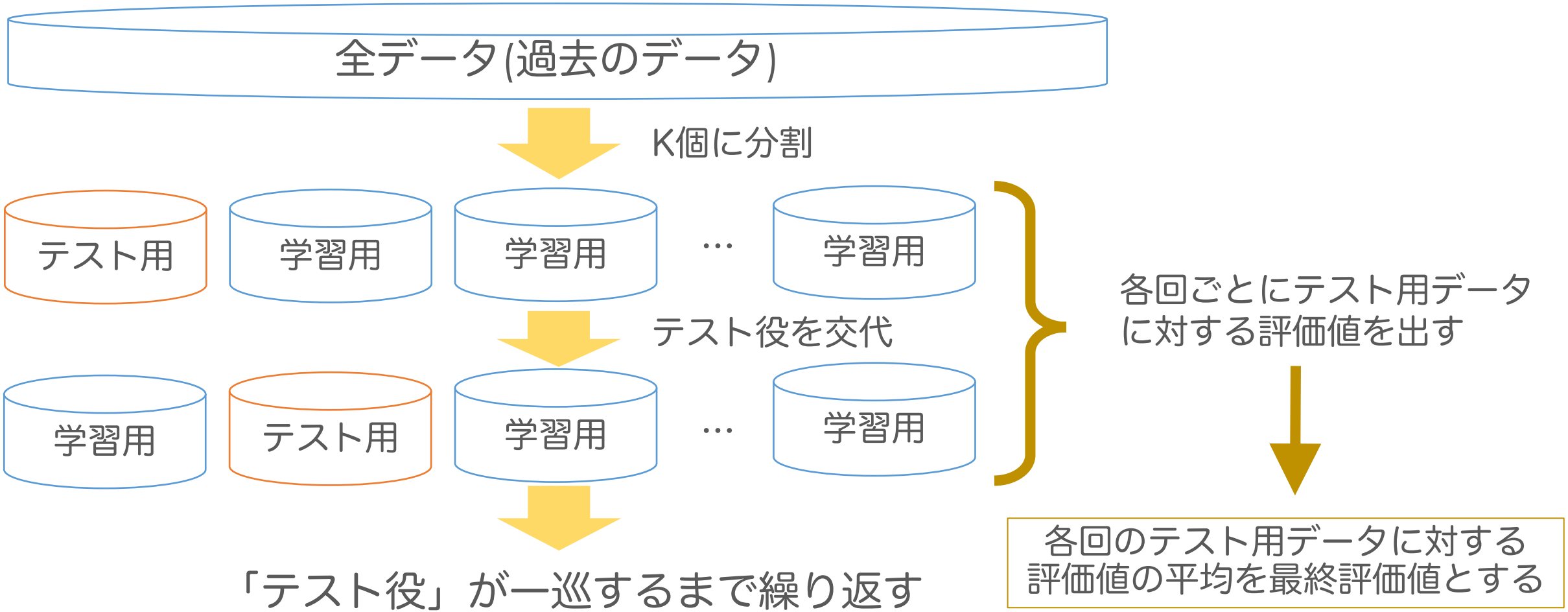
汎化誤差の評価方法｜データ数が少ないときはどうする？

- データ数が少なくとも、汎化誤差をうまく評価する方法はないだろうか？
- ホールドアウト法では、データの一部をテスト用としているが、
  全データを効率よく用いて汎化誤差をうまく評価できないだろうか？

汎化誤差の評価方法｜交差検証（クロスバリデーション）法

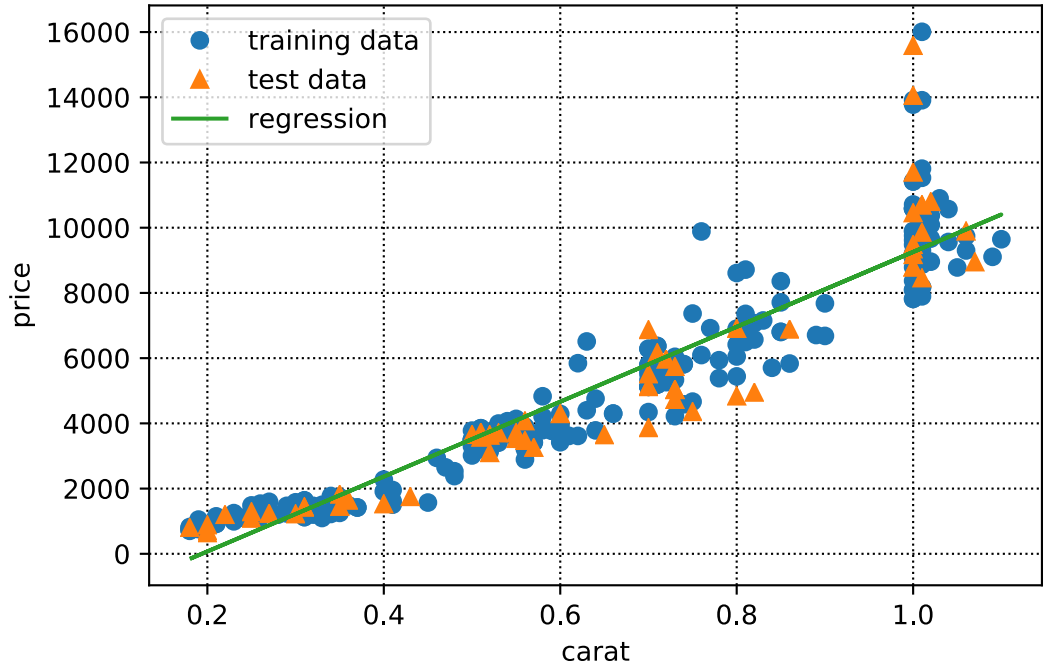
- 交差検証法とは、データを複数のグループにわけ、テスト役と学習役を交代させていくことで少ないデータでも汎化誤差を評価する方法
- データのグループ数をKとしたとき、K分割交差検証法とも呼ばれる
  - 例）グループ数が5なら、5分割交差検証法
  - 分割数を増やすほど、汎化誤差の評価は正確になっていく
- 利点：全てのデータを使うため、ホールドアウト法よりもうまく汎化誤差を評価することができる
- 欠点：学習と評価を何度も繰り返すため、時間がかかる

汎化誤差の評価方法｜交差検証（クロスバリデーション）法



[演習] 1_how_to_validation.ipynb

- 線形回帰モデルを用いてホールドアウト法、交差検証法による汎化誤差の評価方法を確認してみましょう



Fold 1
MAE = 1341.723

Fold 2
MAE = 1239.992

Fold 3
MAE = 949.845

Fold 4
MAE = 1136.542

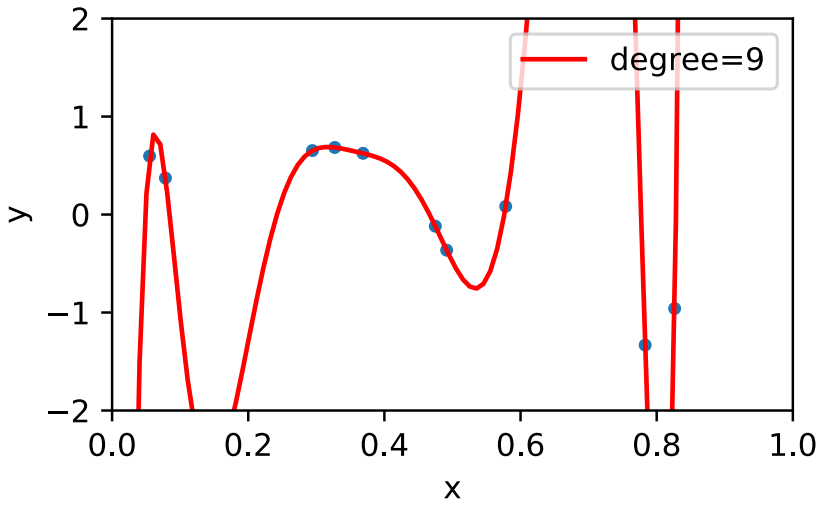
Fold 5
MAE = 1342.977

Cross Validation MAE = 1202.216

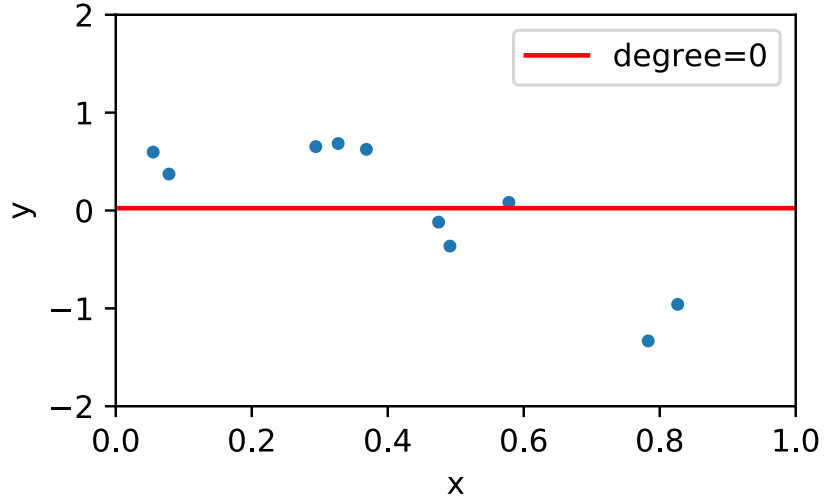
汎化誤差と訓練誤差を比べてみると…?

- 汎化誤差を評価してみると、訓練誤差に比べ非常に大きかったり、訓練誤差も汎化誤差も非常に大きいケースがあった
- この現象が起こる原因は？対処法は？

0 ≒ 訓練誤差 << 汎化誤差だったモデル

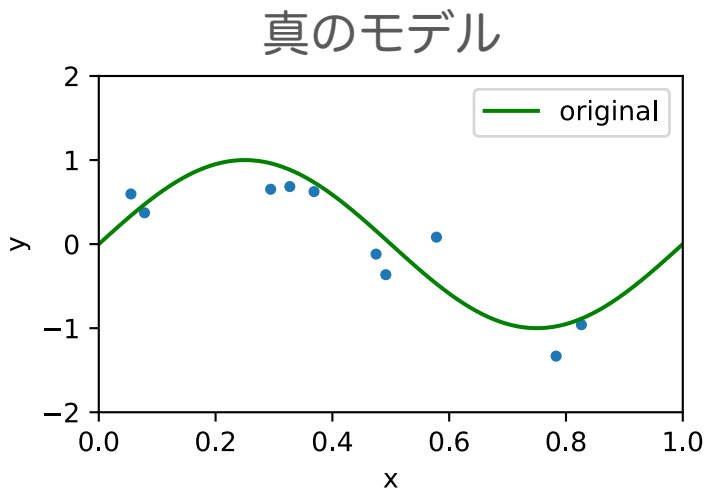


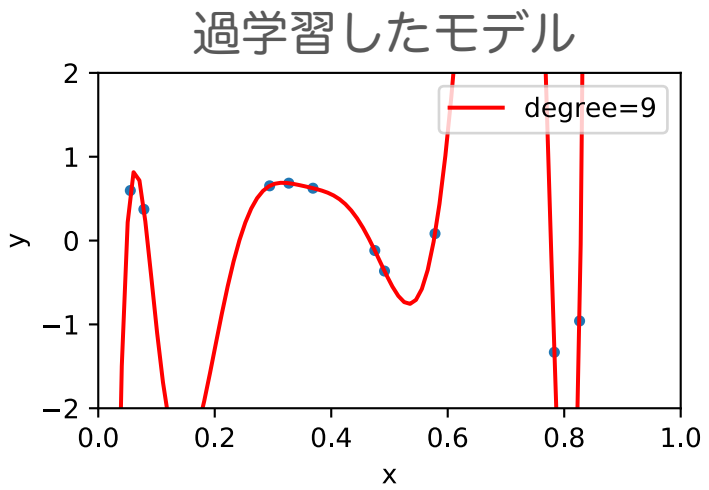
0 << 訓練誤差 ≒ 汎化誤差だったモデル



オーバーフィッティング

- 訓練誤差が十分小さいにもかかわらず、汎化誤差が大きい場合、
  「モデルは学習用データにオーバーフィッティング（過学習）している」という
  - イメージ：練習問題は完璧に解けるが、実際のテストで点が悪かった
- 原因：モデルの表現能力が高すぎる、十分なデータ数が確保できていない





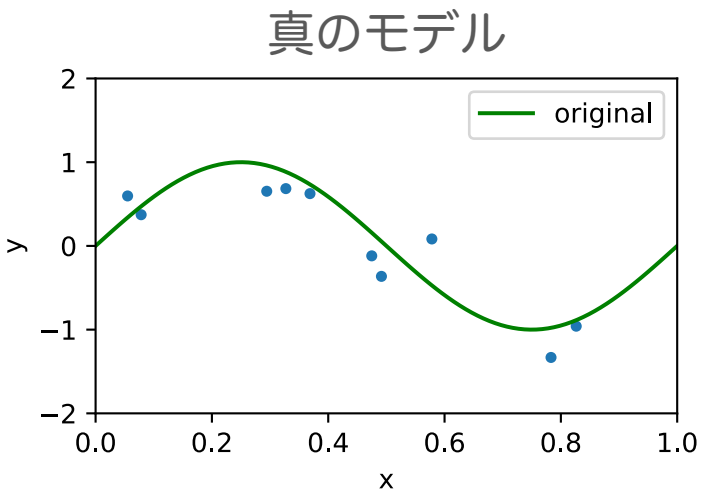
アンダーフィッティング

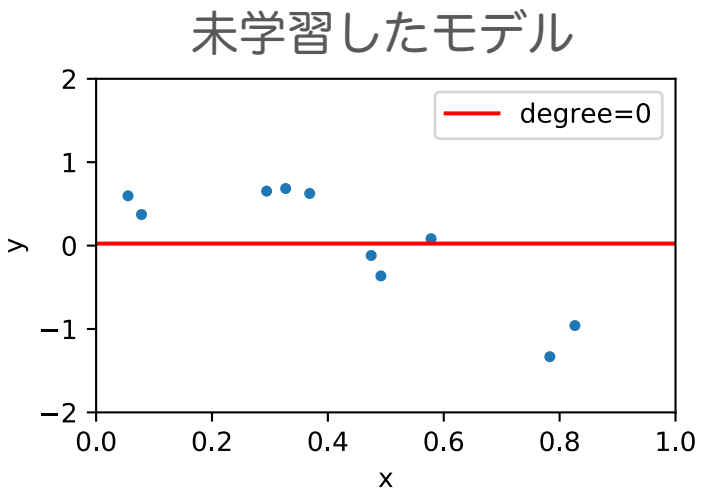
- 一方、訓練誤差と汎化誤差がともに高いままであれば

  「モデルは学習用データにアンダーフィッティング（未学習）している」という

  - イメージ：練習問題も実際のテストも点が悪かった

- 原因：データの性質とアルゴリズムやモデルが噛み合っていない





機械学習でよく問題になるのはオーバーフィッティング

- ニューラルネットワークや木モデルなど非線形データに対応した
  アルゴリズムは、特にオーバーフィッティングが起こりやすい
  - 非線形データに対応できる＝モデルの表現能力が高いとも言える
- 単純な線形回帰/ロジスティック回帰モデルでもデータ数が少ないと
  オーバーフィッティングすることがよくある

オーバーフィッティングへの対処

- 学習用データに強く当てはまりすぎるのが、オーバーフィッティング
- 当てはまりすぎないようにするにはどうすればいいだろう？
- 当てはまりの度合いを制御することはできないだろうか？
  - データへの当てはまりが弱すぎると、逆にアンダーフィッティングしてしまう…
  - いい塩梅はどうやって見つけるのか？

正則化（Regularization）

- 正則化とは、モデルに制約を加えることでオーバーフィッティングを抑制する方法のこと
  - 学習用データへの当てはまりを良くしつつも、制約を満たすように学習
  - 学習用データ「だけ」に当てはまらないように学習することでオーバーフィッティングを抑制することができる
- 正則化は機械学習でよく用いられるテクニック
  - モデルによって正則化の方法は様々

正則化（Regularization）

- 線形回帰モデルを例に正則化を考える
  - ロジスティック回帰モデルなどでも同じ考え方が使えるため
- 二乗誤差（＝目的関数）に制約を意味する「正則化項」を足し合わせる

二乗誤差

$$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - \hat{Y}^{(n)}\right)^2 + \lambda \times \text{正則化項}$$

$\hat{Y}^{(n)}$：n番目のデータの予測値
$y^{(n)}$：n番目のデータの目的変数
（教師データ）

λ：正則化の強さ
（値が大きいほど制約が強くなる）

正則化項の例｜L2正則化、L1正則化、ElasticNet

- 正則化項としてよく用いられるのは、以下の3種類
  - 正則化の内容に応じて、それぞれ手法の名前がついている

手法	正則化項	目的関数
L2正則化 (Ridge [リッジ])	$\|\mathbf{w}\|_2^2 = \sum w_i^2$	$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - \hat{Y}^{(n)}\right)^2 + \lambda\sum w_i^2$
L1正則化 (Lasso [ラッソ])	$\|\mathbf{w}\|_1 = \sum \|w_i\|$	$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - \hat{Y}^{(n)}\right)^2 + \lambda\sum \|w_i\|$
ElasticNet [エラスティックネット]	$\|\mathbf{w}\|_2^2$と$\|\mathbf{w}\|_1$の2つ	$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - \hat{Y}^{(n)}\right)^2 + \lambda_1\sum w_i^2 + \lambda_2\sum \|w_i\|$

λ：係数（学習前に分析者が決める） w：パラメータ（学習によって求める）

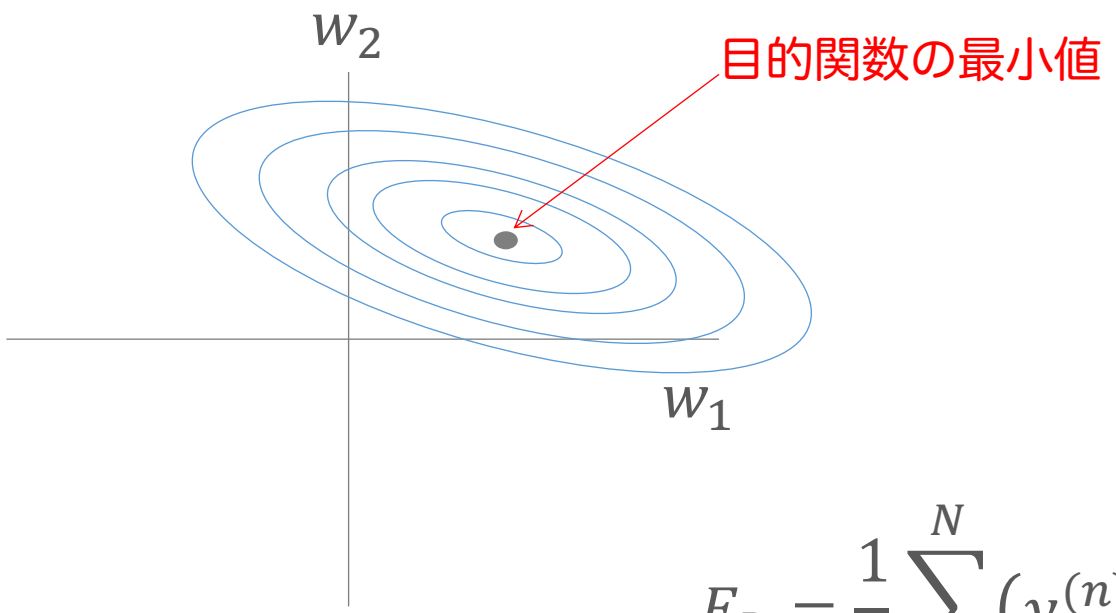
正則化項の例｜L2正則化、L1正則化、ElasticNet

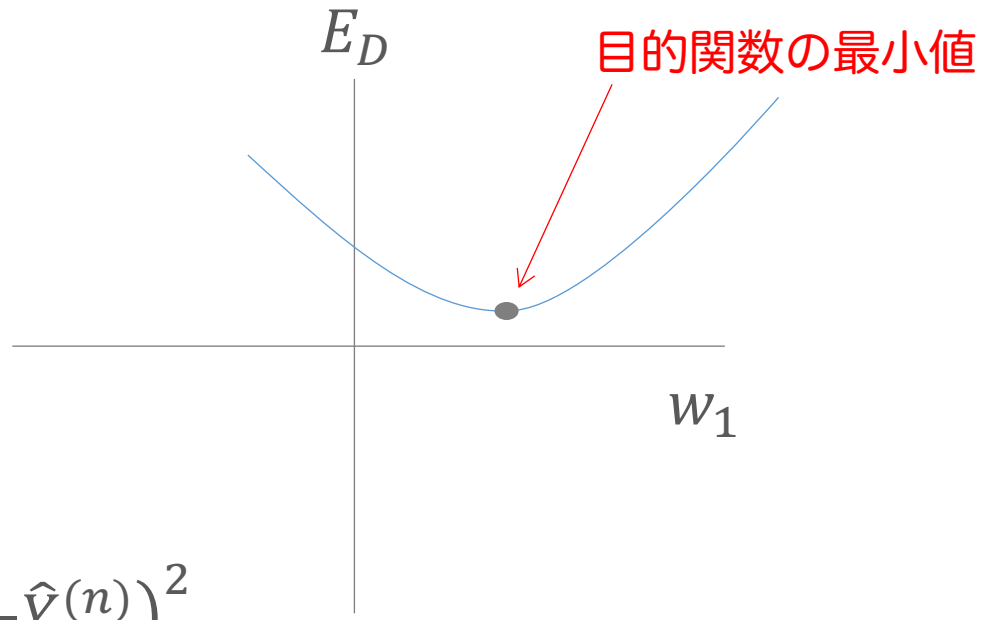
- L2正則化は、係数の値ができるだけ小さくなるような正則化
- L1正則化は、値を0とする係数が多くなるような正則化
- ElasticNetは、L2正則化とL1正則化を組み合わせたもので、両者のバランスによって性質が決まる

手法	正則化項	目的関数
L2正則化 (Ridge [リッジ])	$\|\mathbf{w}\|_2^2 = \sum w_i^2$	$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - \hat{Y}^{(n)}\right)^2 + \lambda\sum w_i^2$
L1正則化 (Lasso [ラッソ])	$\|\mathbf{w}\|_1 = \sum \|w_i\|$	$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - \hat{Y}^{(n)}\right)^2 + \lambda\sum \|w_i\|$
ElasticNet [エラスティックネット]	$\|\mathbf{w}\|_2^2$と$\|\mathbf{w}\|_1$の2つ	$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - \hat{Y}^{(n)}\right)^2 + \lambda_1\sum w_i^2 + \lambda_2\sum \|w_i\|$

最小二乗法における最小値探索のイメージ

- 正則化がどのような意味を持つか知るため、まずは最小二乗法（正則化項なし）の最適化に関するイメージをつかもう



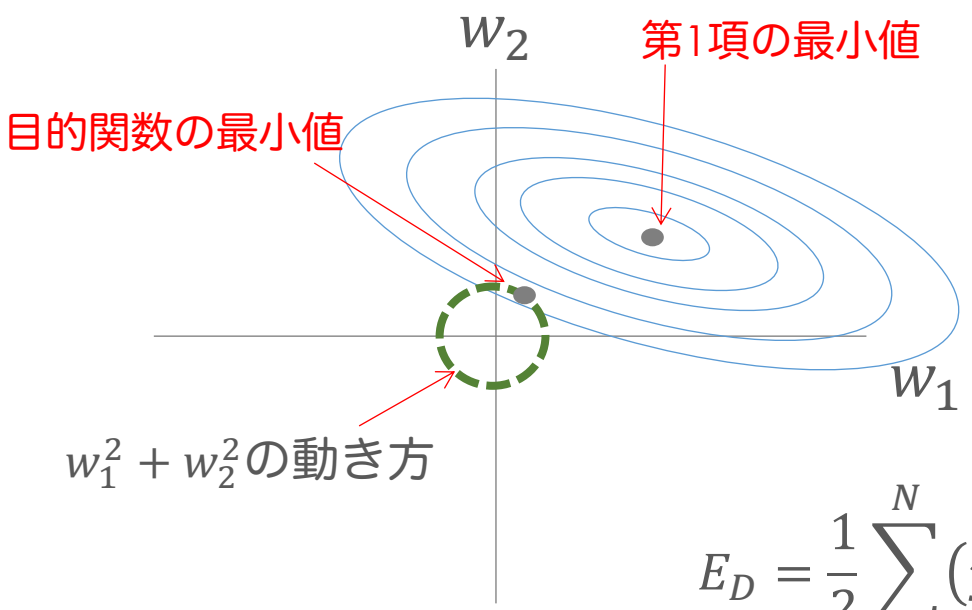


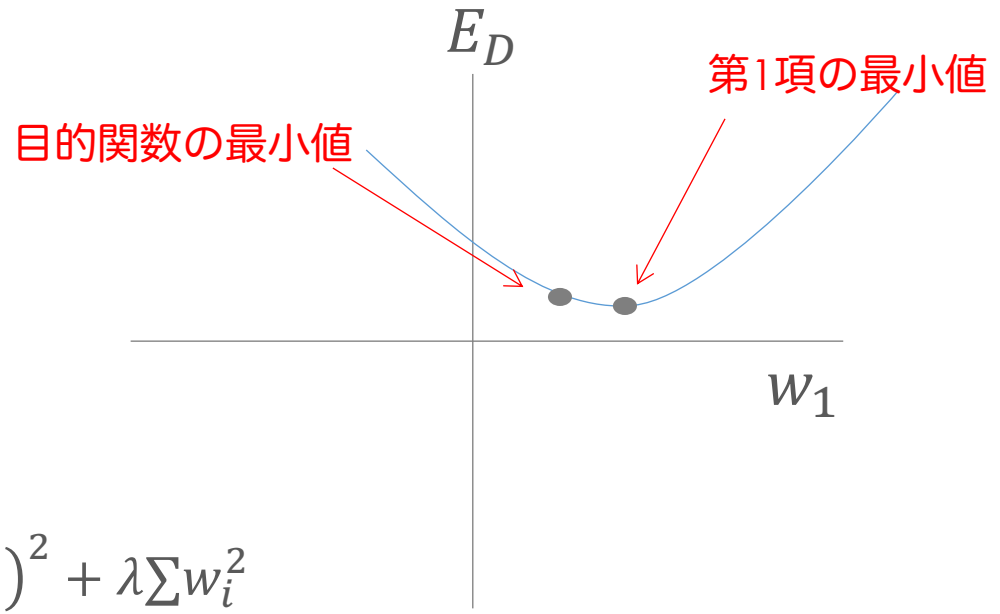
$$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - \hat{Y}^{(n)}\right)^2$$

目的関数E_Dは凸関数となるため、最小値が1つに定まる

L2正則化における最小値探索のイメージ

- 目的関数E_Dの右辺第1項、第2項はそれぞれ正の値であるため、それぞれ小さいほど
  目的関数全体が小さくなる。よって、目的関数の最小値は、第1項と第2項のバランスで決まる。
- $w_1^2 + w_2^2$が同じ値になる点をつなぐと、原点を中心とした円になる。
  つまり、 w_1とw_2は、円上のどこかに決まる。
- 円の半径の大きさはλによって決まる。 λが大きいほど、円の半径は小さくなる。

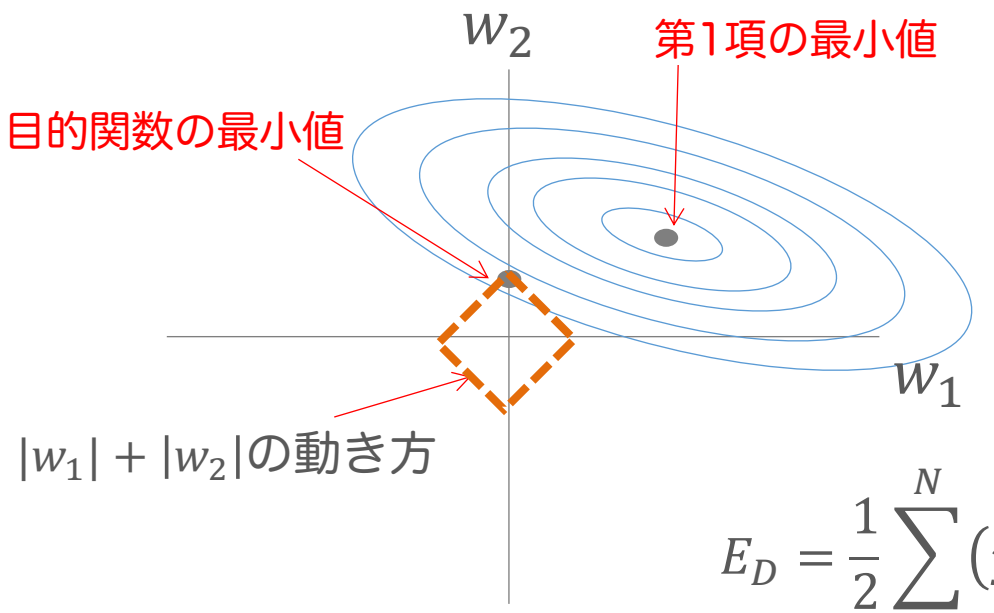


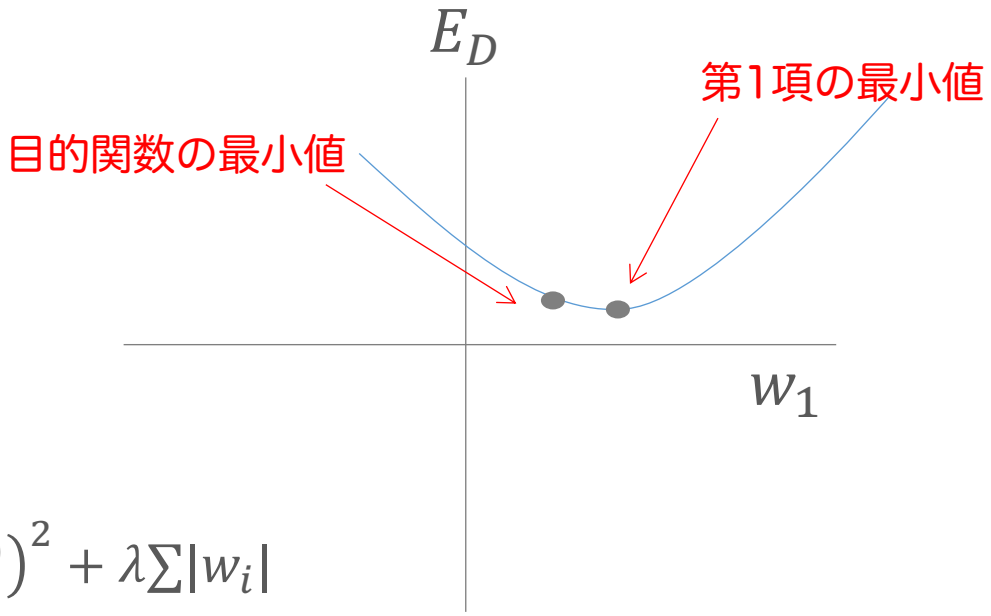


$$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - \hat{Y}^{(n)}\right)^2 + \lambda\sum w_i^2$$

L1正則化における最小値探索のイメージ

- 目的関数E_Dの右辺第1項、第2項はそれぞれ正の値であるため、それぞれ小さいほど
  目的関数全体が小さくなる。よって、目的関数の最小値は、第1項と第2項のバランスで決まる。
- $|w_1| + |w_2|$が同じ値になる点をつなぐと、原点を中心とした正方形になる。
  つまり、w_1とw_2は、正方形上のどこかに決まる。
- 正方形の1辺の長さはλによって決まる。 λが大きいほど、正方形の1辺の長さは小さくなる。





$$E_D = \frac{1}{2}\sum_{n=1}^{N}\left(y^{(n)} - \hat{Y}^{(n)}\right)^2 + \lambda\sum|w_i|$$

[演習] 2_reguralization.ipynb

- 多項式用の線形回帰モデルを使って、オーバーフィッティングや
  アンダーフィッティングを確認してましょう
- 正則化によって、オーバーフィッティングが緩和されることを
  確認してみましょう
- 正則化の強さを変化させることで、得られるモデルにどのような変化が
  起こるか確認してみましょう

正則化項の強さはどう決める？

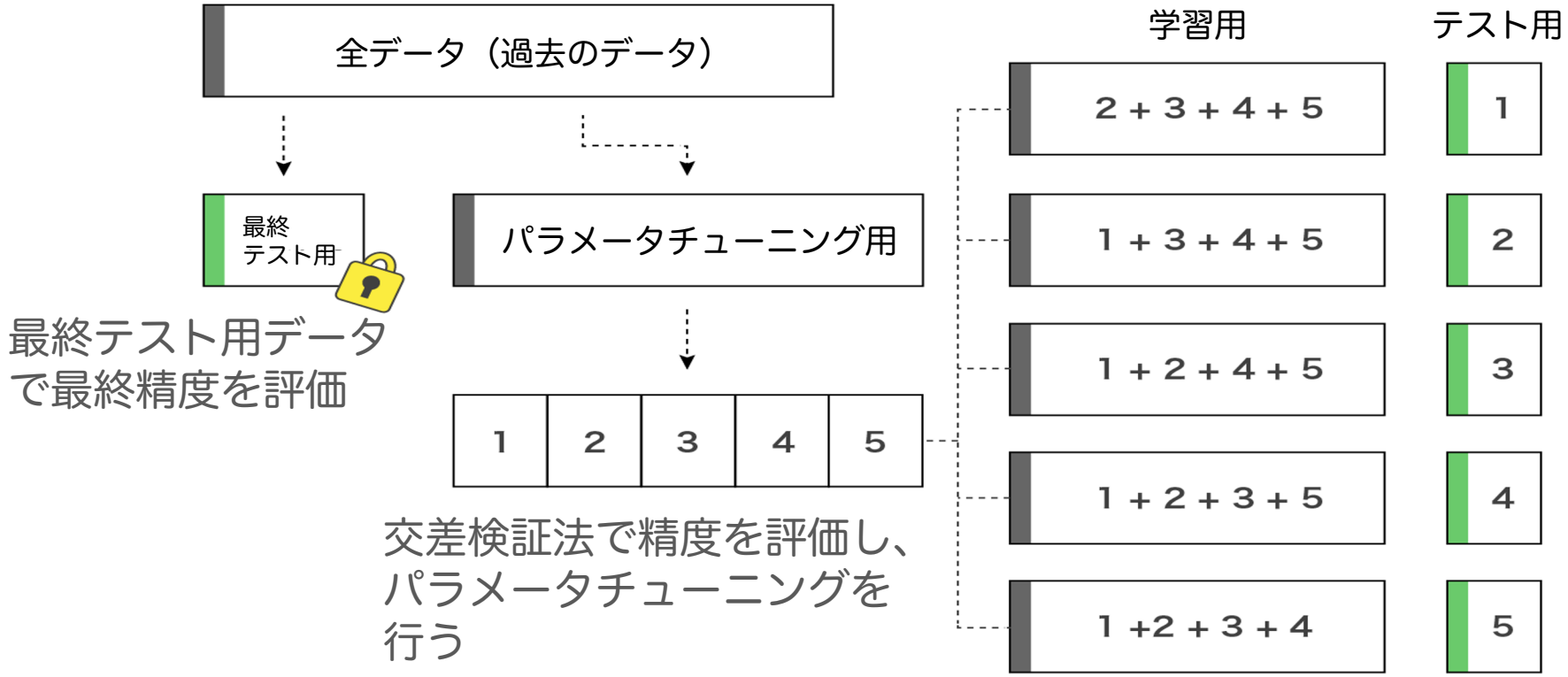
- 正則化項の強さλはどのように決めたらいいだろうか？
- λの良し悪しを決める基準は何だろうか？

正則化項の強さは汎化誤差を見て決める

- 正則化の目的は、学習用データへのオーバーフィッティングを抑制し、汎化性能を改善すること
- つまり、正則化項の強さλは汎化誤差を最小にするように決めるべき！

交差検証法を用いたパラメータチューニング

- 交差検証法によって、汎化誤差が最小になる強さλを決定する
- パラメータチューニングの際には最終テスト用データも作ることを忘れずに！



代表的な前処理

1. 正規化
2. 標準化
3. 無相関化
4. 白色化

データのスケールは気にしなくていい？

- 線形回帰モデルを用いて築年数と敷地面積から住宅価格を予測したい
- 住宅価格や築年数、敷地面積はそれぞれスケール（尺度）が異なるが、このまま学習に使って問題ないのだろうか？

住宅価格[万円]	築年数[年]	敷地面積[m^2]
8499	12	371.29
3980	2	185.38
3680	20	135.5
…	…	…
4250	2	117.24

前処理の必要性

- 一般のデータは、スケールが統一されていないことが多い
- 線形モデルの場合、データのスケールにも適応する必要があり、
  スケールの不統一は学習を不安定にさせたり、係数の解釈を難しくさせる
  - 1000万を0.01倍した結果と10を0.01倍した結果は、
    予測値への貢献度合いが全く異なるはず
- そこでデータに前処理を施し、スケールをあわせることで、
  学習を安定化させ、係数の解釈しやすさを改善する効果が期待できる

正規化

- 正規化は、データのスケールを0から1の間に収めるための前処理である
- 各変数ごとに**学習用データから**最大値と最小値を求め、以下の式によって正規化する
- 外れ値に影響されやすいので注意

$$\frac{\text{データの値} - \text{最小値}}{\text{最大値} - \text{最小値}}$$

※テスト用データの正規化に用いる最小値・最大値は
学習用データで求めた値を用いること

標準化

- 標準化は、データの平均と標準偏差を用いた前処理のこと
- 各変数ごとに**学習用データから**平均と標準偏差を求め、以下の式によって標準化する

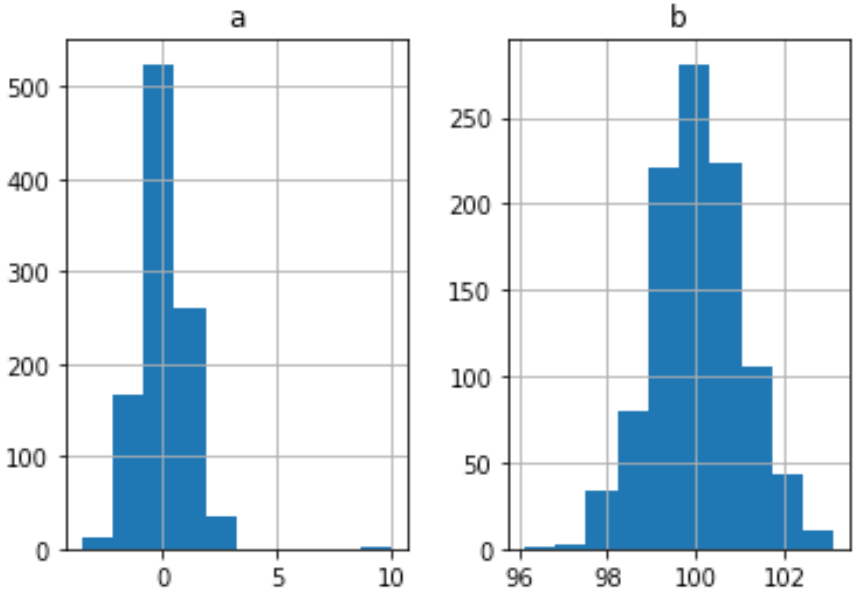
$$\frac{\text{データの値} - \text{平均値}}{\text{標準偏差}}$$

※テスト用データの標準化に用いる平均・標準偏差は
学習用データで求めた値を用いること

正規化と標準化

- 0付近にスケーリングしたいのであれば、標準化
- 0-1にスケーリングしたいのであれば、正規化
- 正規化は外れ値の影響を受けやすい

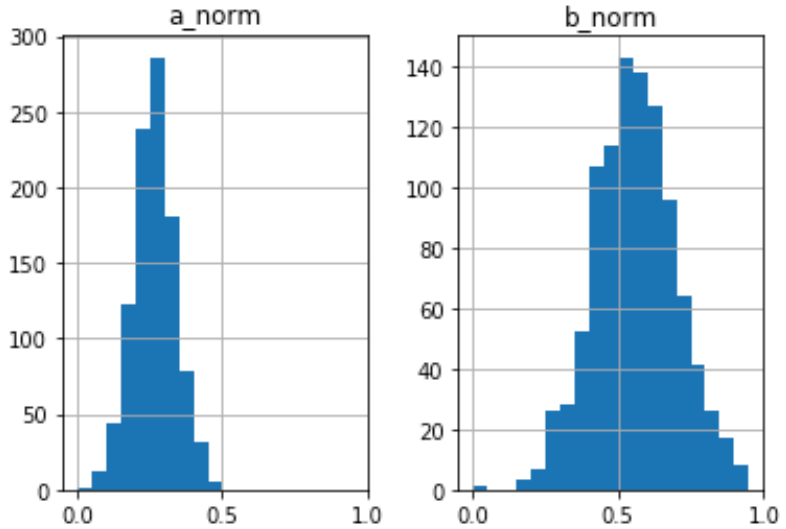
元データ



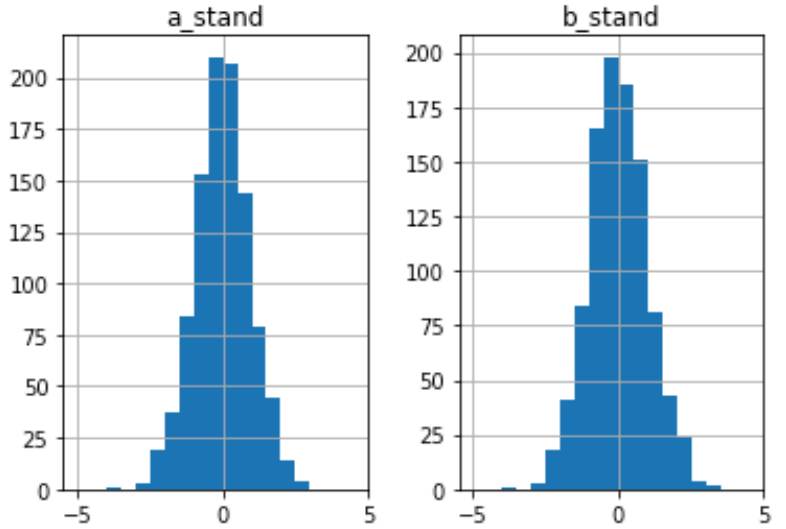
正規化

標準化

正規化の結果



標準化の結果

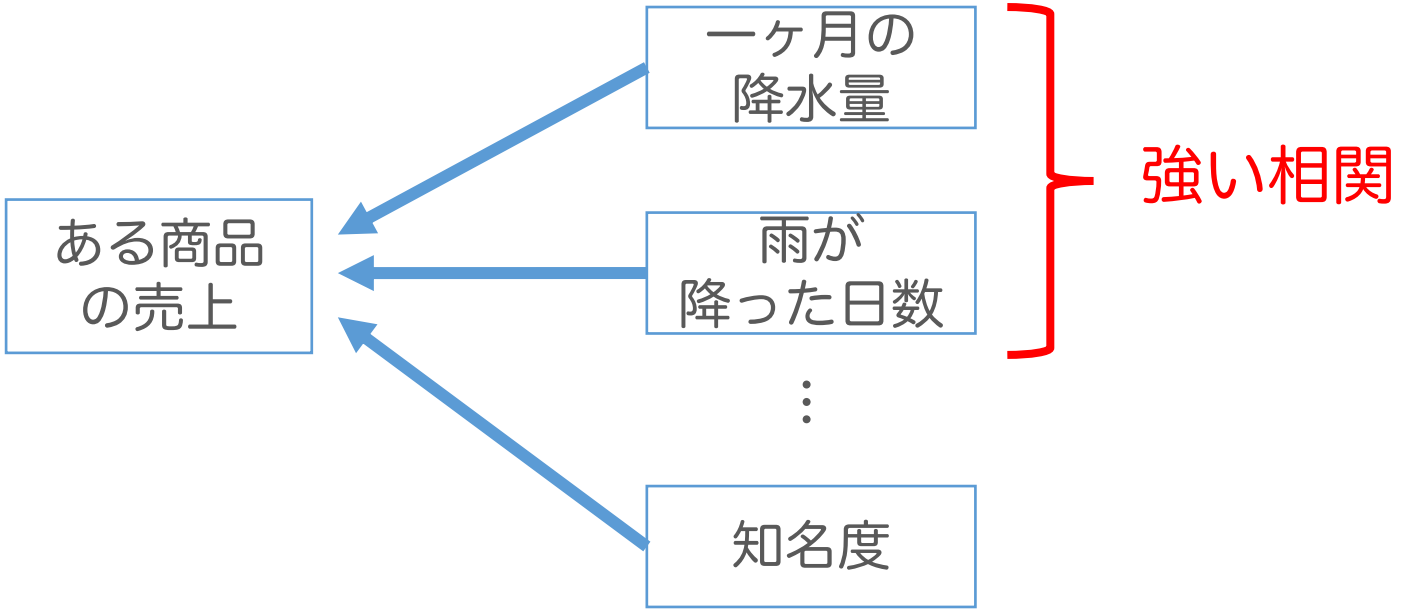


[演習] 3_normalization_and_standalization_trainee.ipynb

- 正規化、標準化の処理を確認してみましょう
- 実際のデータに対する正規化、標準化を実装してみましょう

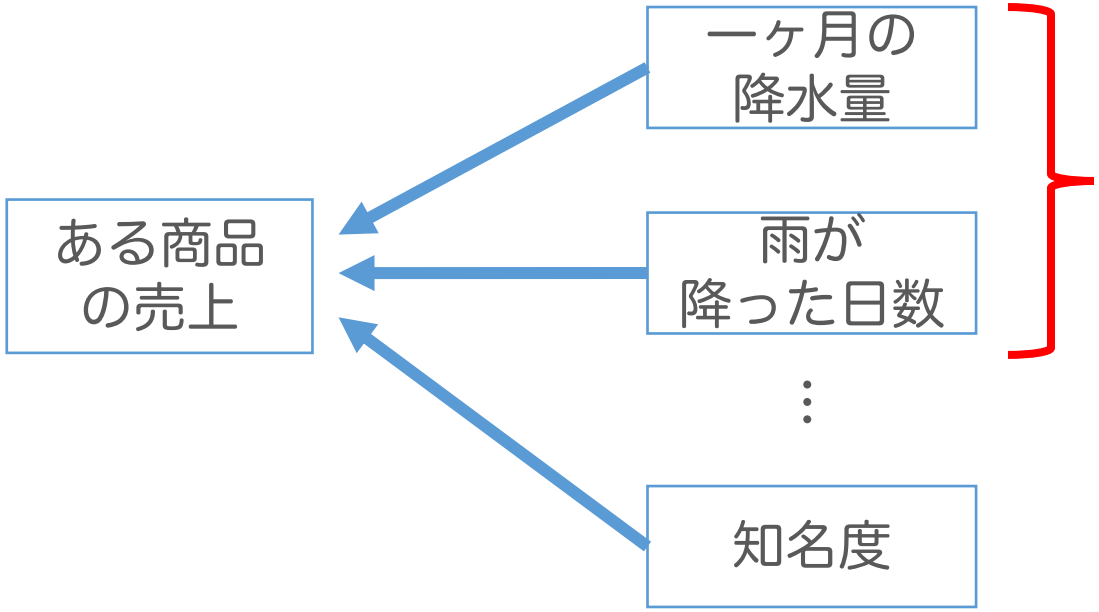
説明変数間の相関

- 線形回帰モデルを用いて、以下の説明変数からある商品の売上を予測したい
- 説明変数間の相関を見てみると「一ヶ月の降水量」と「雨が降った日数」に強い正の相関が見られた
- 説明変数間に強い相関があることは何を意味する？学習への影響は？



説明変数間の相関

- 相関係数が高い場合、一方の変数のみで事象を説明できることが多い
- これでは使える情報が減ってしまうだけでなく、説明変数に強い相関があると学習が不安定になる、係数の解釈に信用性がなくなってしまうことがある
- そこで、データに対して相関を取り除く前処理を施す



この2つの変数は
同じような情報を表現している

無相関化

- 無相関化は説明変数間の相関を取り除く前処理である
- 無相関化は**学習用データから**分散や共分散を求め、
  共分散を0にするようにデータを変換することで無相関化する
- 計算方法は「主成分分析」と類似しているため、詳細はDAY4で解説する

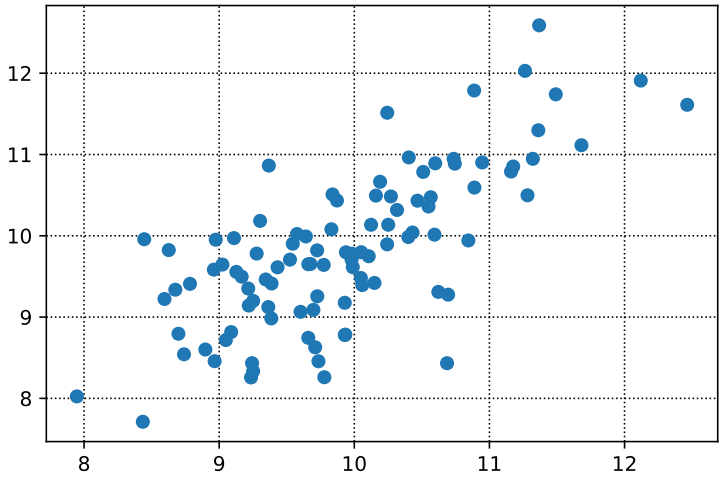
白色化

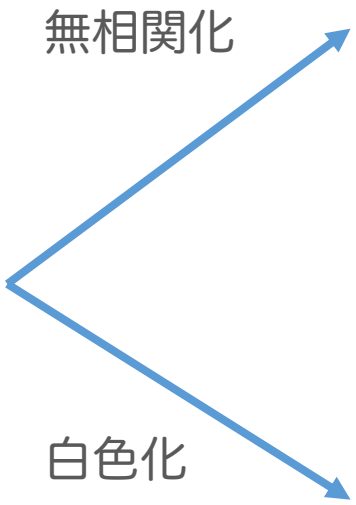
- 白色化は、無相関化したデータにさらに標準化を施す前処理である
  - 標準化→無相関化の順番で実行しても結果は同じである
- 白色化は**無相関化したデータ**にその平均、標準偏差をそれぞれ求め、以下の式によって白色化する

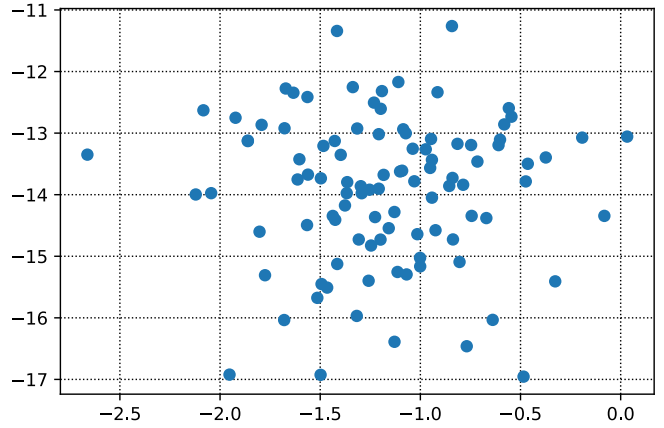
$$\frac{\text{無相関化したデータの値} - \text{無相関化したデータの平均値}}{\text{無相関化したデータの標準偏差}}$$

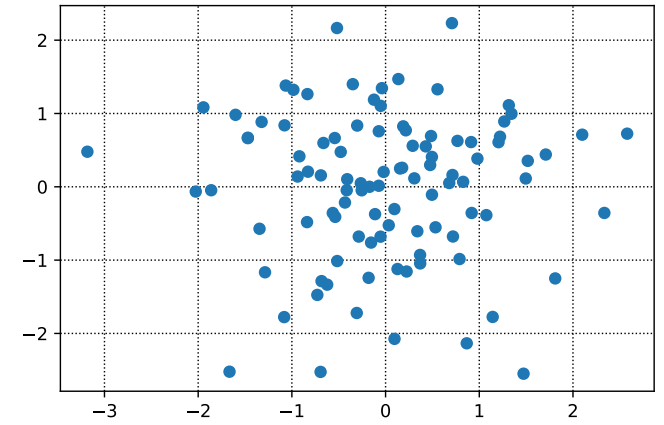
[演習] 4_decorrelation_and_whitening.ipynb

- 無相関化、白色化の処理とその結果を確認してみましょう







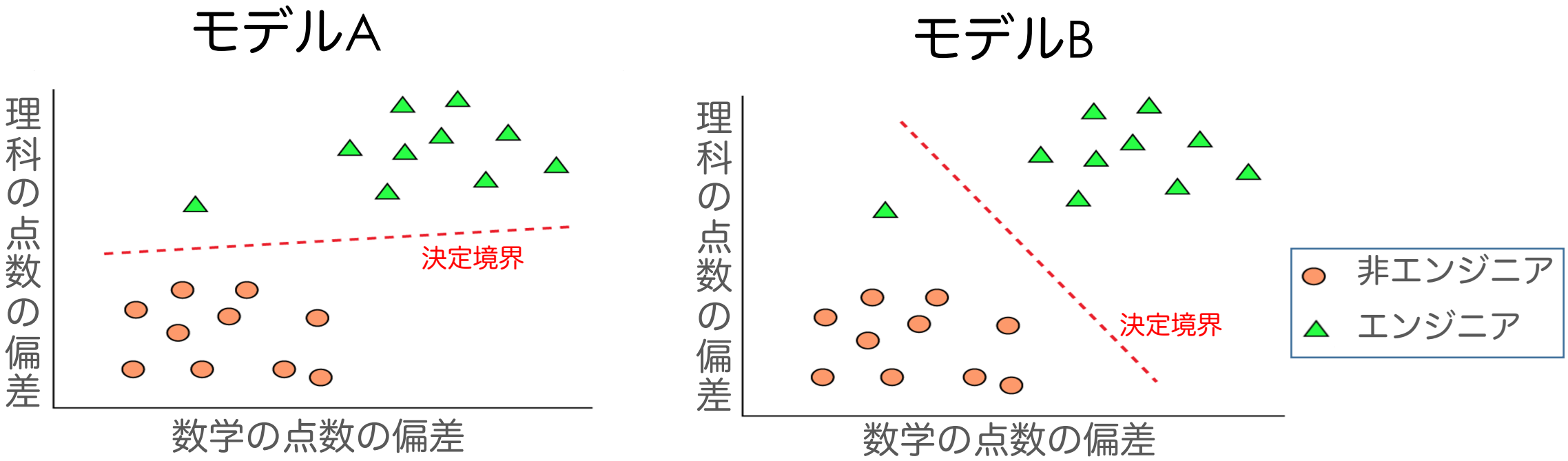


サポートベクターマシン

1. マージンの導入
2. 線形識別関数の導入
3. サポートベクターマシンの目的関数
4. カーネル法とカーネルトリック

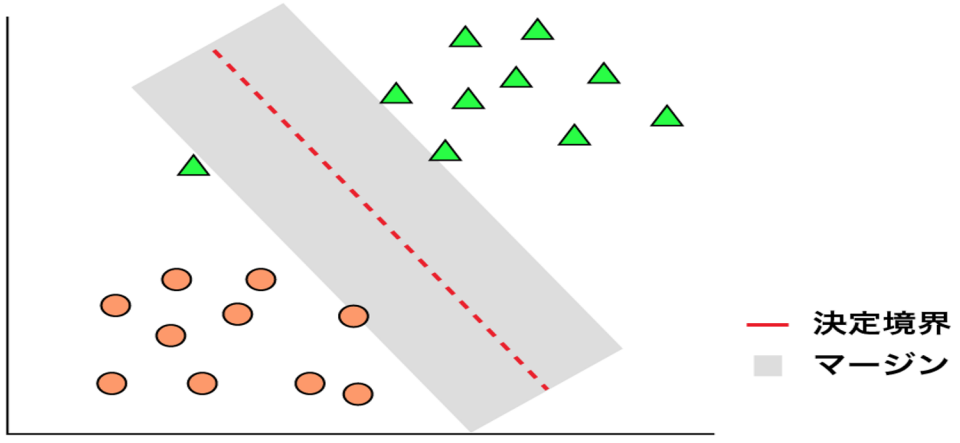
どのような決定境界が適切？

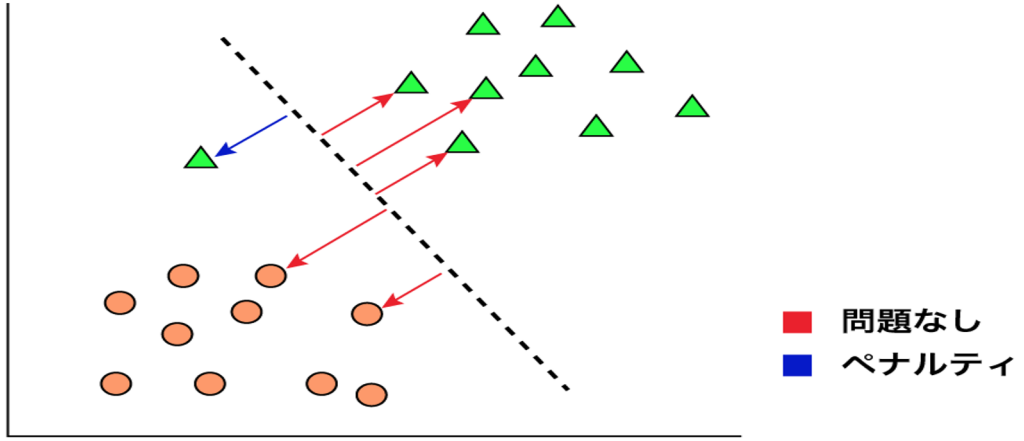
- 理科の点数の偏差、数学の点数の偏差から将来エンジニアになるかならないか分類するモデルを作成したい
- 新たなデータに対しても、できるだけ誤分類をしない境界線を決めたいが、そうなりそうなモデルはどっち？



SVM の基本原理

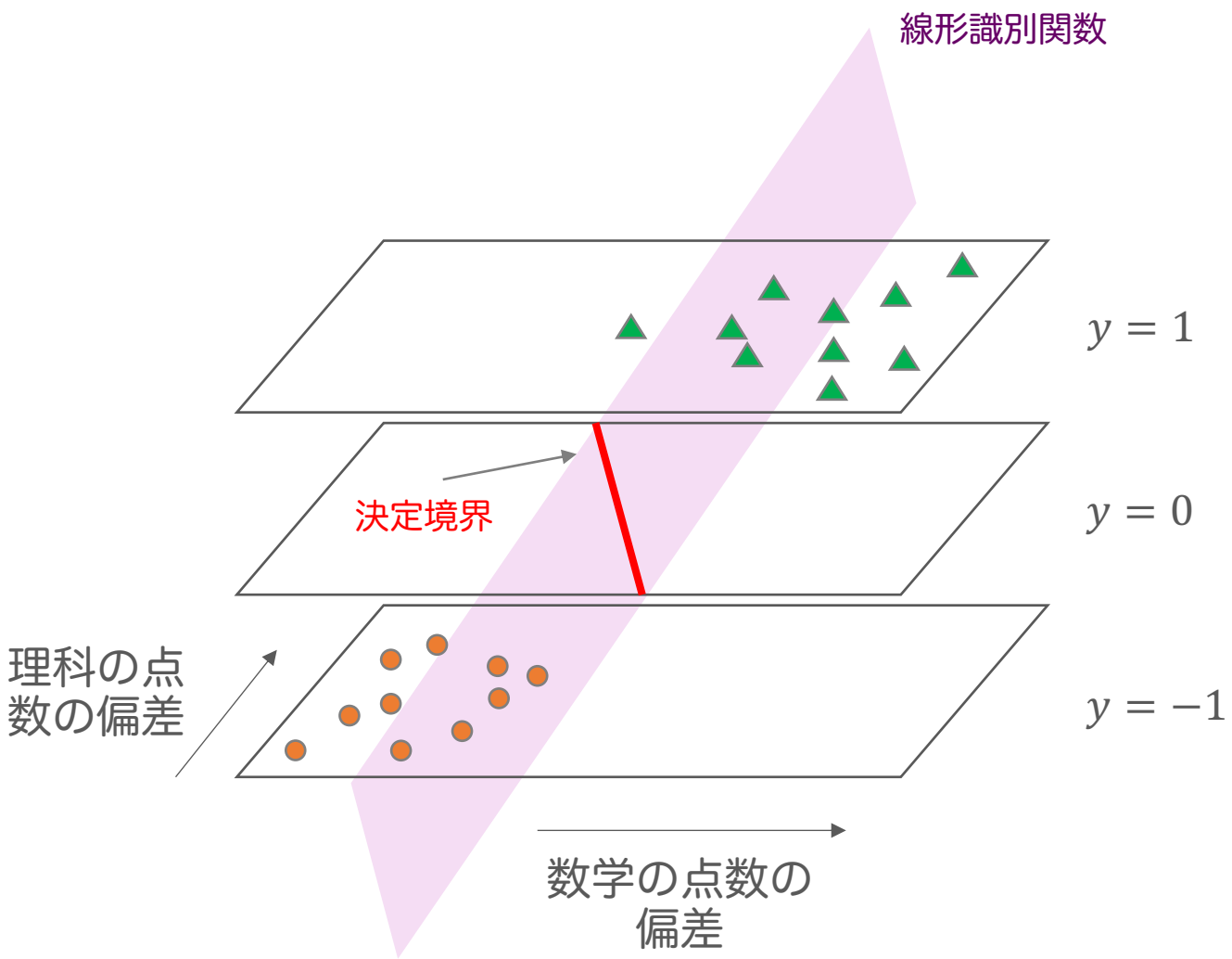
- 決定境界の両側のグレーの部分をマージンとよぶ
- この決定境界は、2値を完全に分類できないものでもよいが、誤った分類結果にはペナルティが与えられる
- SVMでは、より大きな「マージン」が得られ、ペナルティがなるべく小さくなるなような決定境界を求める





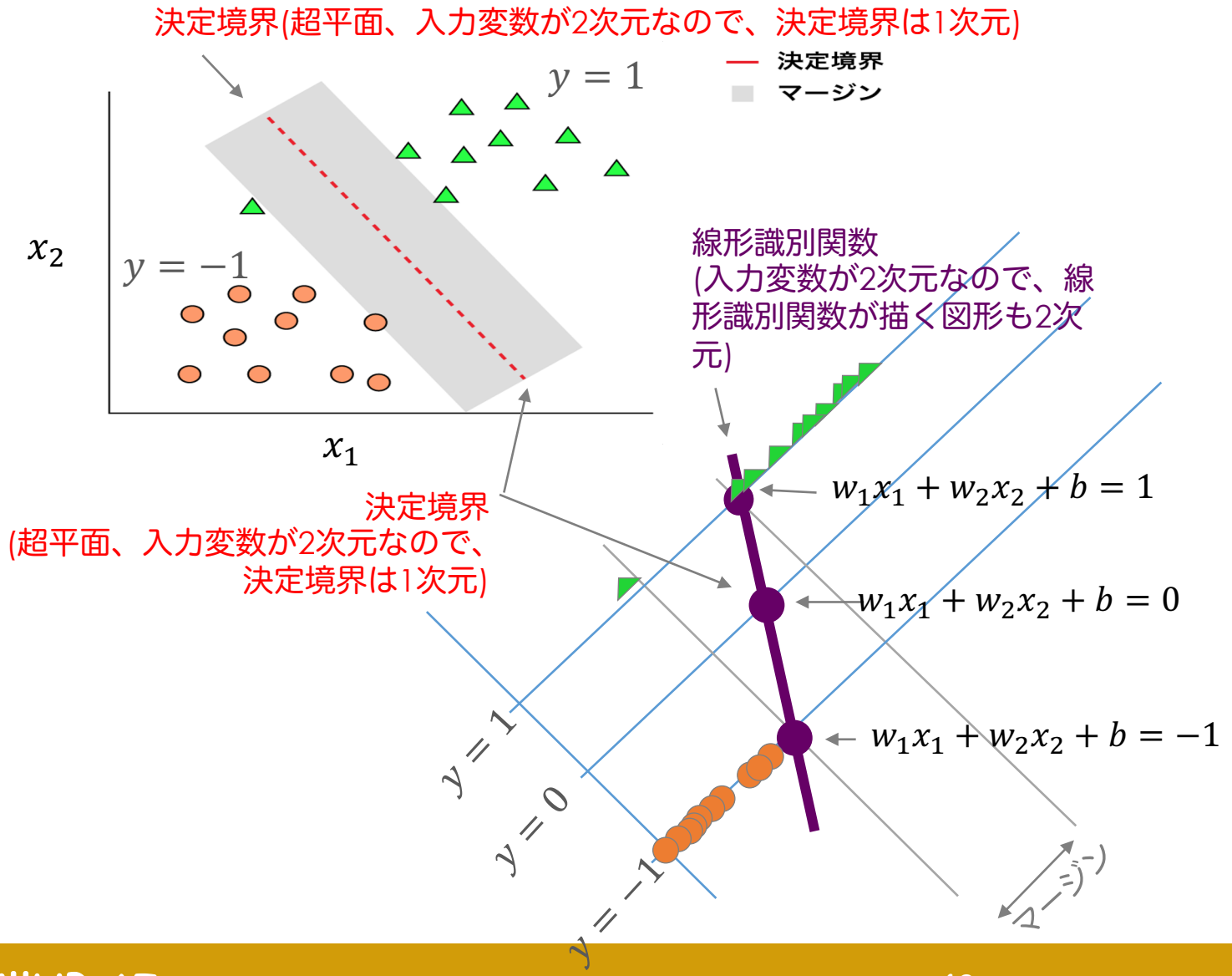
誤った分類結果に「ペナルティ」を与える

決定境界のイメージ



- 決定境界の空間的イメージは、ロジスティック回帰の境界線と似ている。(DAY1ロジスティック回帰の章を参照)
- マージンを最大化する決定境界は、傾きの小さな紫面を求めることに相当する。

最大マージンを探すために線形識別関数を考える



- 最大マージンを求めて、その中心を決定境界にする。決定境界は超平面となる。
- 超平面とは、3次元以上の平面も含めた一般化された平面の概念。
- 入力変数が2次元の場合、決定境界は直線になり、入力変数が3次元の場合、決定境界は2次元平面になる。(決定境界の次元＝入力変数の次元ー1)
- 最大マージンを求めるために、$y = w_1x_1 + w_2x_2 + b$という線形識別関数(yが描く図形は、入力変数の次元と同じ)を定義する。
- 原則的に、$y = 1$のデータは、線形識別関数よりも下にいなければならない。(制約①)
- 原則的に、$y = -1$のデータは、線形識別関数よりも上にいなければならない。(制約②)
- マージンが大きいということは、線形識別関数の傾きが緩いということになる。
- つまり、制約①②を満たしながら、最も勾配の小さい線形識別関数を求めればよいことになる。
- 最も勾配が小さいとは、$y = w_1x_1 + w_2x_2 + b$におけるx_1、x_2の影響が小さいということ。つまり、w_1、w_2の値が小さいということ。
- これは、L2ノルムとおなじ！

$\|\boldsymbol{w}\|_2^2$　L2ノルムの2乗

SVMの目的関数

2018年度 第1回E資格
出題

- マージンを最大化することを定式化し、式を整理すると、 $\boldsymbol{w}$の2ノルム2乗を最小化することになる。 $\boldsymbol{w}$とは、各説明変数に掛かる係数のこと
- 目的関数の導出の詳細は、『はじめてのパターン認識』の8.1節と8.2節をご参照ください

目的関数　$\frac{1}{2}\|\boldsymbol{w}\|_2^2$　この目的関数を最小化したい

制約条件　$t_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) \geq 1$

$\boldsymbol{w}$：各説明変数に掛かる係数
$\boldsymbol{b}$：バイアス
t：正解データ $t = \{1, -1\}$

SVMの目的関数と「C」の調整

- 線形分離可能でない場合に対応するため、目的関数と制約条件を下記のように拡張する
  - 前半がマージン最大化を意味する項、後半がペナルティ項
  - ξは、識別境界をどれだけ超えているかを表す変数
    - このξは、スラック変数と呼ばれる
    - スラック変数とは、最適化問題において，不等式制約を等式制約に変換するために導入する変数のこと
- Cが大きければ、ペナルティ項が大きくなるため第二項のξを小さくすること(誤分類をなるべく少なくすること) を重視する

目的関数 $\frac{1}{2}\|w\|_2^2 + C\left(\sum_i \xi_i\right)$

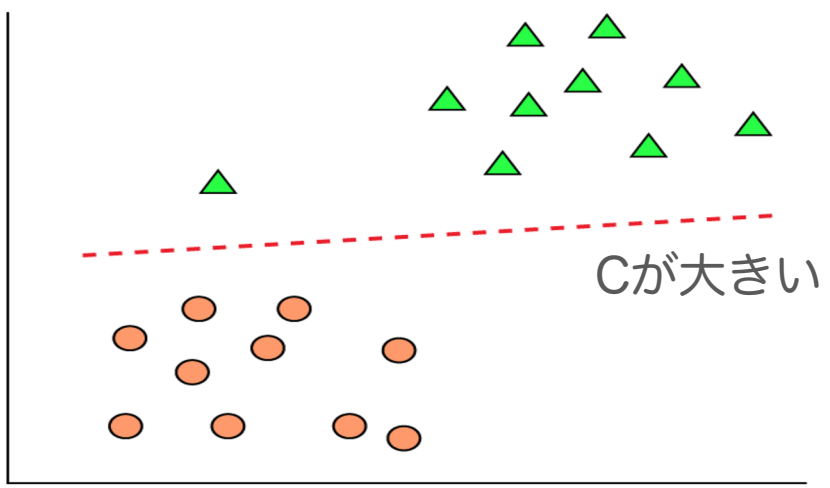
マージン最大化の項　ペナルティ項

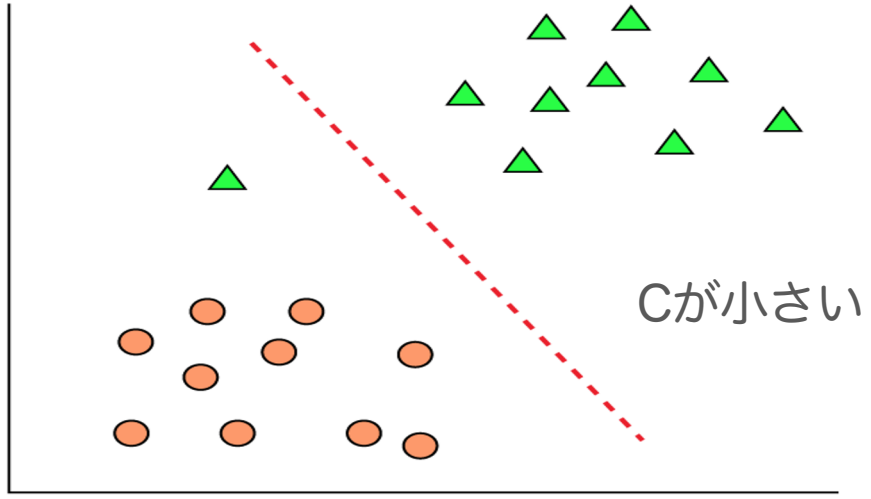
この目的関数を最小化したい

制約条件 $t_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b\right) - 1 + \xi_i \geq 0, \qquad \xi_i \geq 0$

Cの調整の影響

- Cの値が大きいほど、マージンも考慮するがペナルティの影響も大きくなるため、ペナルティを極力避ける決定境界になる





ペナルティの制御

- マージンはとにかく大きいほど良い
- ペナルティの影響力の大小が、学習結果を左右する
  - ペナルティの影響力が大きければ、できるだけペナルティが出ないようにしたい
  - ペナルティの影響力が小さければ、ペナルティを気にするより、マージンを大きく取ることに集中したい
- ペナルティの影響力は、学習開始前に自分で決定しなければいけない

スラック変数ξの定義

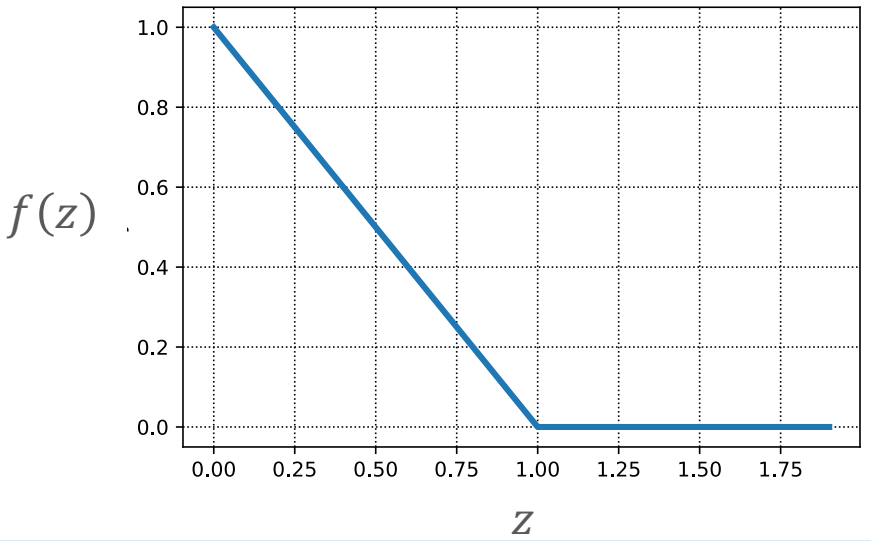
- スラック変数ξは、制約条件を紐解くと、下式の形になる
- この関数fは、 135°に開いている蝶番に似ていることからヒンジ損失関数と呼ばれる

$$\xi_i = f\left(w^{\mathrm{T}}x_i\right) = \max[0, 1 - y_i(w^{\mathrm{T}}x_i)] \qquad y_i\text{:正解データ(-1 or 1)}$$

ヒンジ損失関数

$$f(z) = \max[0, 1 - z]$$

1以上なら0を返し、
1未満なら1-Zを返す関数



スラック変数ξの定義

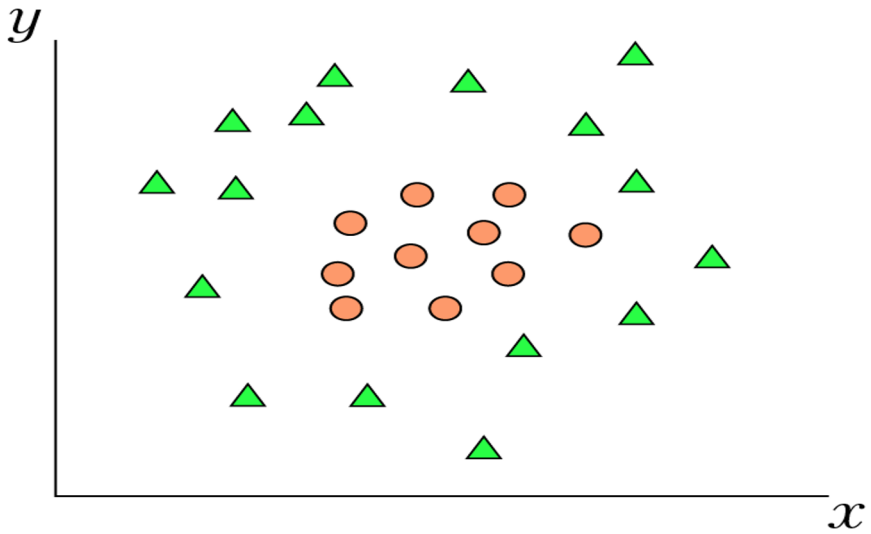
- 前頁の定義式を用いると、結果的にスラック変数ξは、以下の値になる
  - あるデータx_iが正しく分類され、マージンよりも遠くにあるとき、ξ_iは0
  - あるデータx_iが正しく分類され、 x_iがマージン内にあるとき、ξ_iは0を超え1以下の値
  - あるデータx_iが誤分類されるとき、ξ_iは1を超える値
- これより、スラック変数ξは、誤分類に対するペナルティとして機能していることがわかる

サポートベクターマシンの解釈

- スラック変数ξの定義から、ペナルティ項はヒンジ損失関数によってデータへの当てはまり具合を評価したものと見ることができる
- また、マージン最大化の項は、L2正則化と同じ式で表現される
- つまり、サポートベクターマシンはL2正則化付きのヒンジ損失最小化学習をする機械学習アルゴリズムである
- 最適化は制約付き凸最適化手法を用いるが、複雑であるため割愛
  （詳しくは『機械学習のための最適化講座』で！）

SVMによる非線形の分類

どのような決定境界が望ましい？

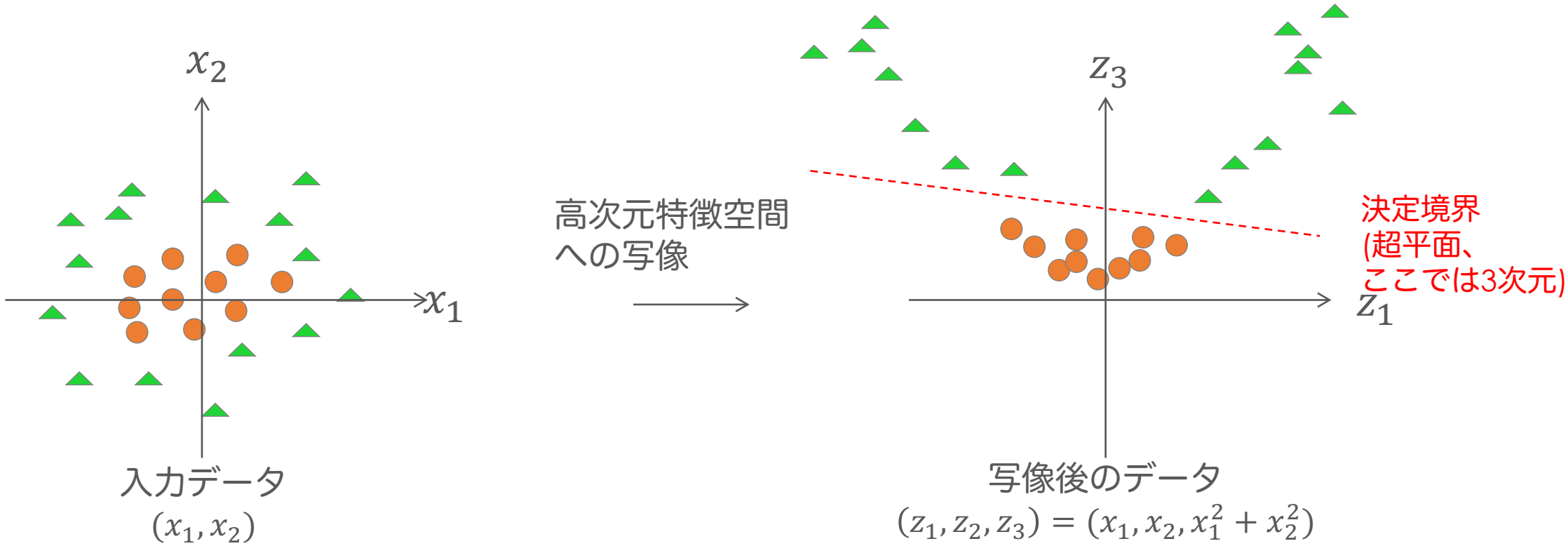


「発想の転換」

- 一般に、説明変数の次元数が大きいほど、線形識別面が存在する可能性が高くなる
- この性質を逆手にとって、低次元の入力データ(低次元特徴空間)を
  高次元空間に写像し、その高次元特徴空間上で識別平面を求めることにする
- この発想の転換により、SVMの枠組みにおいて非線形の決定境界を実現できる

高次元特徴空間への写像

- 高次元特徴空間への写像例を以下に示す
- 2次元の入力データ(x_1, x_2)を$(z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$の3次元空間に写像する
- 写像後の3次元特徴空間において、最大マージンとなる決定境界を求める



高次元特徴空間への写像

- 前頁の高次元特徴空間への写像は、一見すばらしいアイデアに見えるが、
  $(z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$という写像がいつもうまくいくとは限らない
- 入力変数の次元が大きくなると高次元特徴空間の次元数が指数関数的に増え、
  高次元特徴空間への写像に膨大な時間がかかるようになる
  - うまくいくようになっても、計算時間がかかりすぎては実運用で使えない
- そこでカーネル法を用いることで、高次元特徴空間への写像というコンセプトは
  そのままで、高次元特徴空間へ写像させる計算が不要になる
  - これはカーネルトリックとも呼ばれている

SVMとカーネル法の関係

- SVMでは、最適解を求める際に内積計算を行なっている
- 内積とは、原点からみて同じような方向と位置にあるかを測る方法であり、
  その値によってデータどうしの類似度を表すことができる
  - この内積計算部分は、線形カーネル関数とも呼ばれる
- この内積計算部分を非線形カーネル関数に置き換えることで、高次元特徴空間へ
  写像後のデータについてデータどうしの類似度を測っていることと同じ意味になる
- 詳しくは、『はじめてのパターン認識』8.3節をご参照ください

カーネル関数とは

- カーネル関数には様々な種類があり、非線形カーネルでは
  ガウスカーネル関数や多項式カーネル関数が有名
- カーネル法はSVMだけでなく、様々なアルゴリズムで使用されている
- カーネル法を用いてもある程度計算処理が重くなるため、
  データ数が多い場合は注意する必要がある

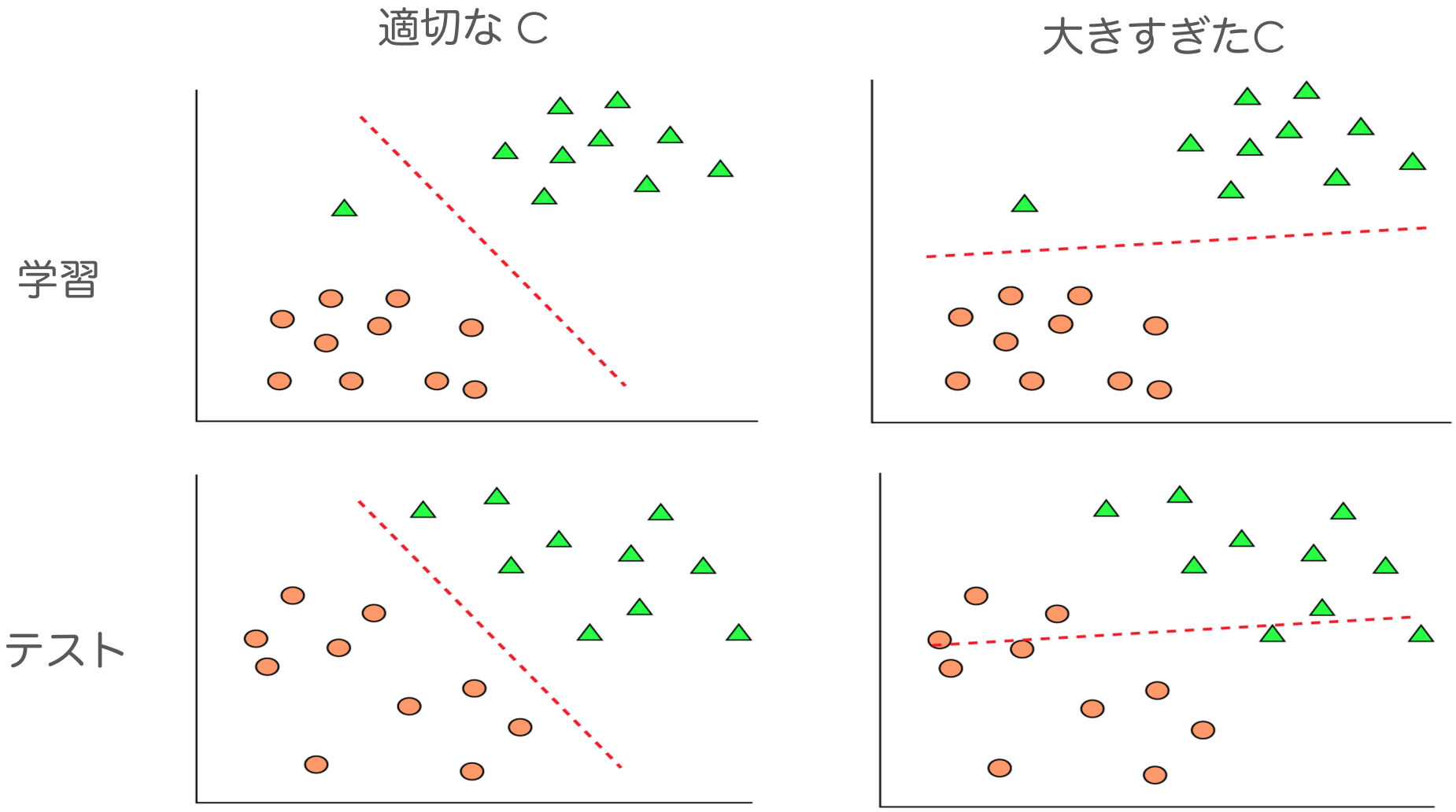
カーネルの使い分け

- 説明変数が多く、データが少ない場合は、過学習のリスクがあるためカーネル法を用いない（＝線形カーネルを用いる）方がよい
- 説明変数が少なくデータ量がそこそこあれば、非線形カーネル関数を利用してみることで予測性能が改善することがある

Cの決定

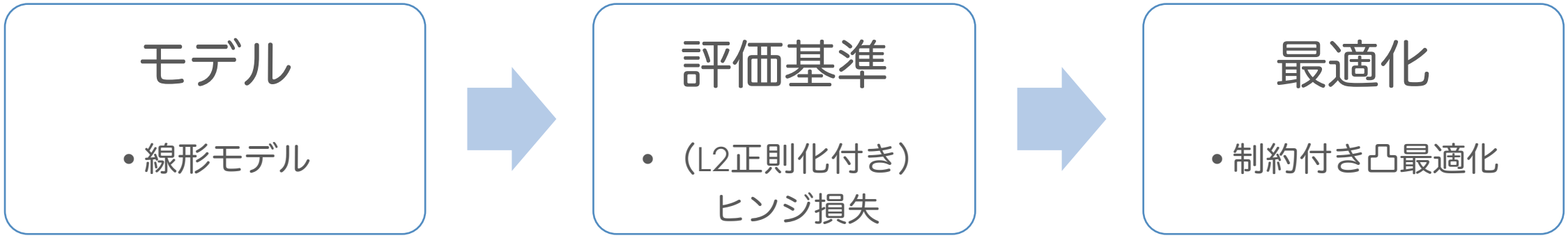
- ここまで、パラメータ C は試行錯誤の末に決定していた
- 学習にかかる時間が増えるほど、手動で探索的に値を決めていくことは面倒が多くなるため、探索行為自体は自動的におこなえるようにしたい
- そういう場合は、グリッドサーチ（交点探索）を実施する
- グリッドサーチとは、パラメータをいろいろ変えて、最適なパラメータを探す手法のこと
- グリッドサーチの際には、パラメータの良し悪しを交差検証法によって評価した汎化誤差によって決める

最適なCの決定



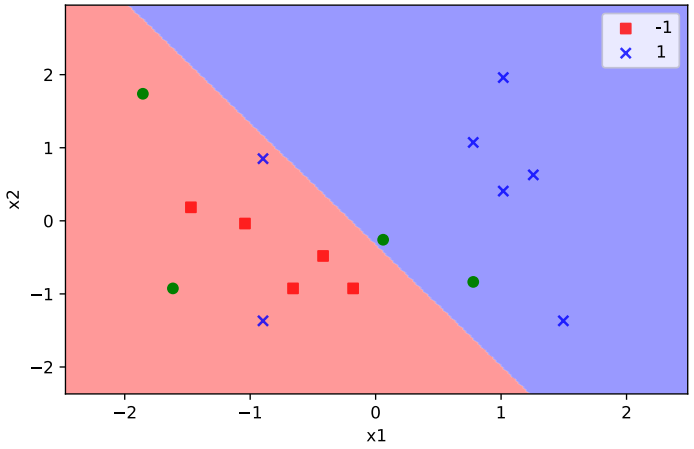
サポートベクターマシンまとめ（モデル、評価基準、最適化の観点から）

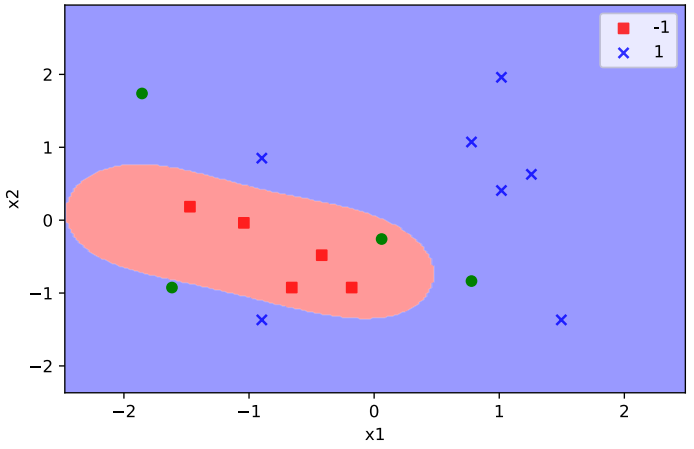
- サポートベクターマシン：マージンを最大化する境界線を学習するモデル
- L2正則化付きのヒンジ損失最小化と等価となる
- カーネル法によって非線形データに対応することができる



[演習] 5_SVM_practice.ipynb

- サポートベクターマシンの実装方法を確認してみましょう
- さらにカーネル関数を適用して決定境界がどのように変化するのか確認してみましょう
- 最後にペナルティ項の強さをチューニングしてみましょう





[参考Notebook] 6_SVM_kernel_function.ipynb

- カーネル関数の動作・振る舞いを確認するNotebookです
- このNotebookでは多項式カーネルとガウスカーネルを取り上げます

[グループワーク] サポートベクターマシンとロジスティック回帰の比較

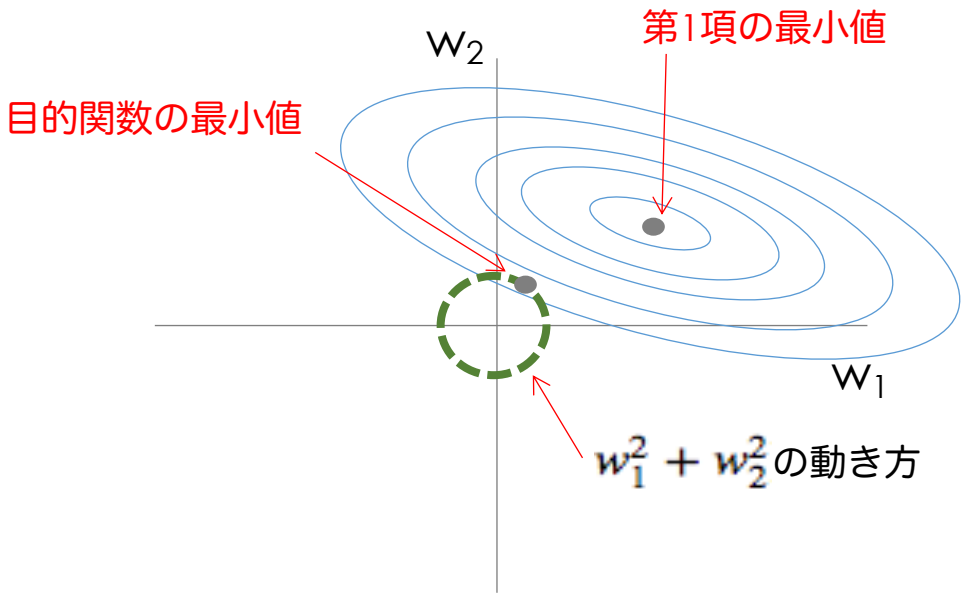
- 分類問題を対象にサポートベクターマシンを紹介しました
- サポートベクターマシンにできて、ロジスティック回帰にできないことを挙げてみましょう（5分）
- それをもとに、サポートベクターマシンが適切な問題とロジスティック回帰が適切な問題をできるだけ多く挙げましょう（10分）
- 最後に以上2つで議論した結果を全体に向けて発表しましょう（各グループ2分ずつ）

Any Questions?

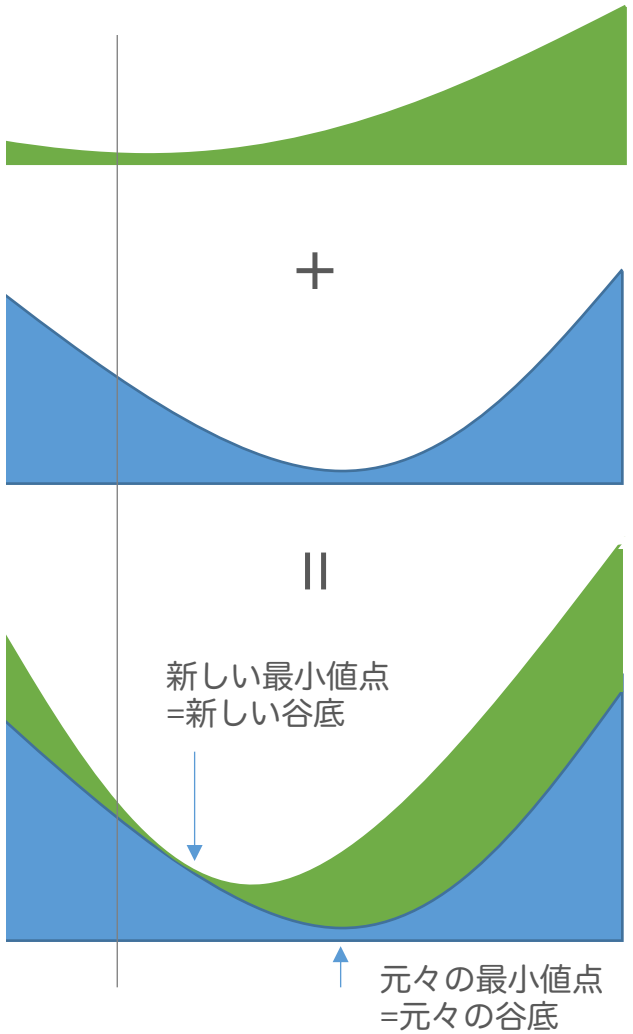
Appendix

L2正則化とL1正則化における最小値探索のイメージ

L2正則化における最小値探索のイメージ



元々の谷底にたどり着く前にボールは止まる。
つまり、学習用データに完全にフィットする
前に計算が止まるため、過学習が抑えられる。

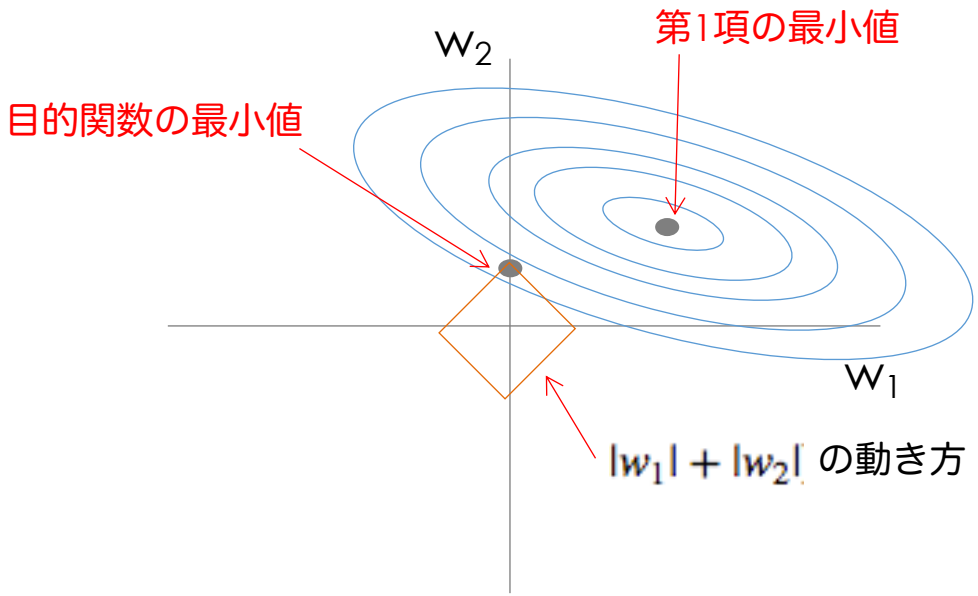


正則化項
=山全体に土を被せる。
被せる土の傾斜はλに
よって決まる

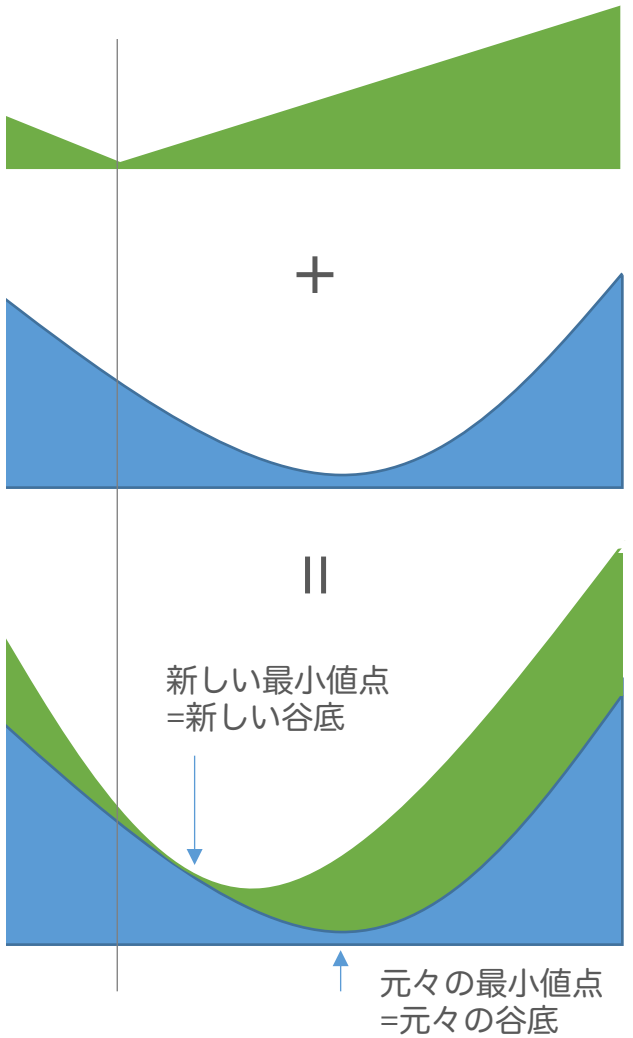
元々の目的関数
=元々の山

新しい目的関数
=新しい山

L1正則化における最小値探索のイメージ



元々の谷底にたどり着く前にボールは止まる。
つまり、学習用データに完全にフィットする
前に計算が止まるため、過学習が抑えられる。



正則化項
=山全体に土を被せる。
被せる土の傾斜はλに
よって決まる

元々の目的関数
=元々の山

新しい目的関数
=新しい山