

# 現場で使える 機械学習・データ分析基礎講座

第 1 章：機械学習概論

- 人工知能と機械学習
- 回帰と分類
- 機械学習モデルの構築・運用の流れ
- 機械学習手法の種類
- 教師あり学習とは
- 線形モデルと非線形モデル

- 人工知能、機械学習とは何かを説明できる
- 機械学習におけるタスクの意味を説明できる
- 回帰と分類の違いを説明できる
- 機械学習モデルの開発・運用の流れを説明できるようになる
- 機械学習手法の種類を 3 つ列挙し、それぞれ説明できる
- 教師あり学習の仕組みを、関数という言葉を使って説明できる
- 線形モデルと非線形モデルの違いを説明できる

# 人工知能と機械学習

---

# 人工知能（Artificial Intelligence）の定義



ジョン・マッカーシー\*  
(John McCarthy, 1927-2011)

Q. 人工知能とは何でしょうか？

知的な機械、特に、  
知的なコンピュータプログラムを作る科学と技術です<sup>[1]</sup>

Q. 知能とは何でしょうか？

実際の目標を達成する能力の計算的な部分です  
人間、動物、そして機械には、種類や水準が  
さまざまな知能があります<sup>[1]</sup>

\* : 人工知能研究の第一人者。「Artificial Intelligence」という用語を初めて公の場で  
使用した人物であり、プログラミング言語 LISP の開発者でもある

[1] 人工知能学会：人工知能のFAQ、<https://www.ai-gakkai.or.jp/whatsai/AIfaq.html> (2023/07/10 アクセス)

- 機械学習 → 人工知能の技術領域のひとつ
- 深層学習※（ディープラーニング）→ 機械学習の方法論のひとつ

「人工知能 = 機械学習 = 深層学習」ではないので要注意！！

## 人工知能 (AI)

機械学習

深層学習

SVM

線形回帰

クラスタ分析

決定木

etc…

エキスパートシステム

探索アルゴリズム

etc…

※ディープニューラルネットワーク (DNN) は、深層学習の方法論のひとつ

# 機械学習 (Machine Learning) の定義



明示的にプログラムしなくても学習する能力を  
コンピュータに与える研究分野<sup>[2]</sup>のこと

具体的には…

コンピュータプログラムが、あるタスクにおいて、  
用意されたデータを使い、性能の評価値を向上させること

アーサー・リー・サミュエル\*  
(Arthur Lee Samuel, 1901-1990)

\* : 世界初の学習型プログラム「Samuel Checkers-playing Program」を開発した  
計算機科学者。ハッシュテーブルの考案者でもある。

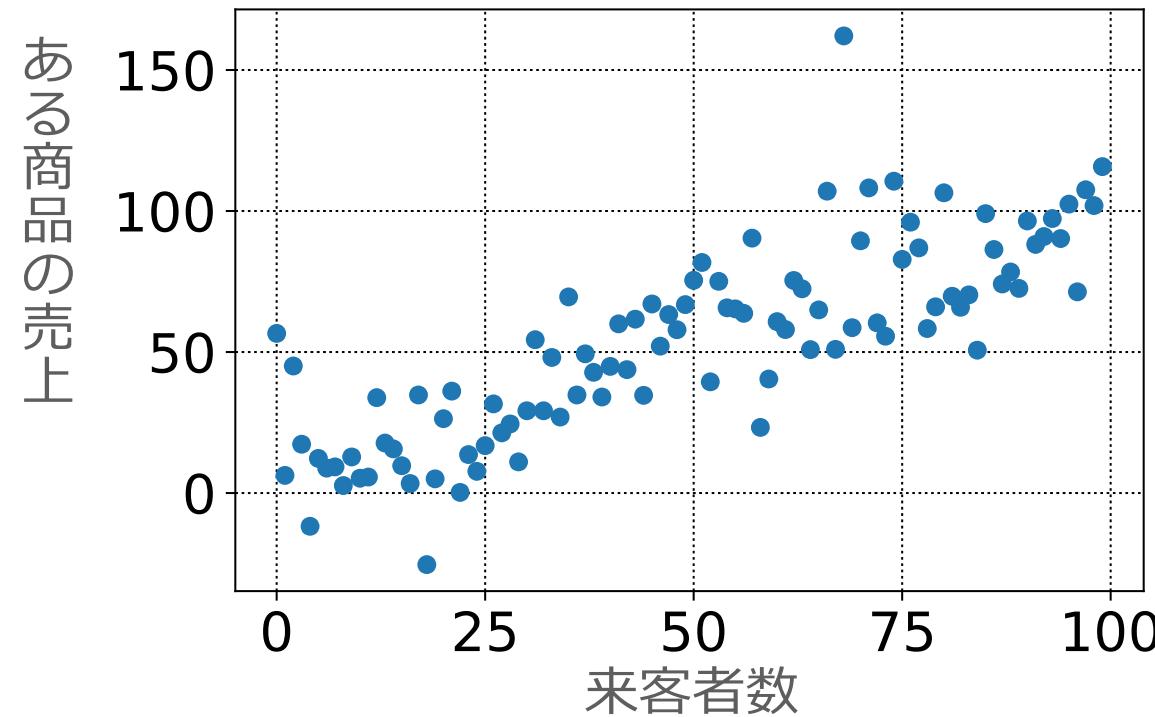
[2] Wikipedia : 機械学習、<https://ja.wikipedia.org/wiki/機械学習> (2018/09/12アクセス)

- 機械学習モデルが対象とする問題設定のこと
  - 機械学習モデルは、なんらかのタスクを解くために構築！
- 具体例
  - 画像認識
  - 物体検出
  - 文章分類
  - 機械翻訳
  - 売上予測
  - 購買予測
  - etc…

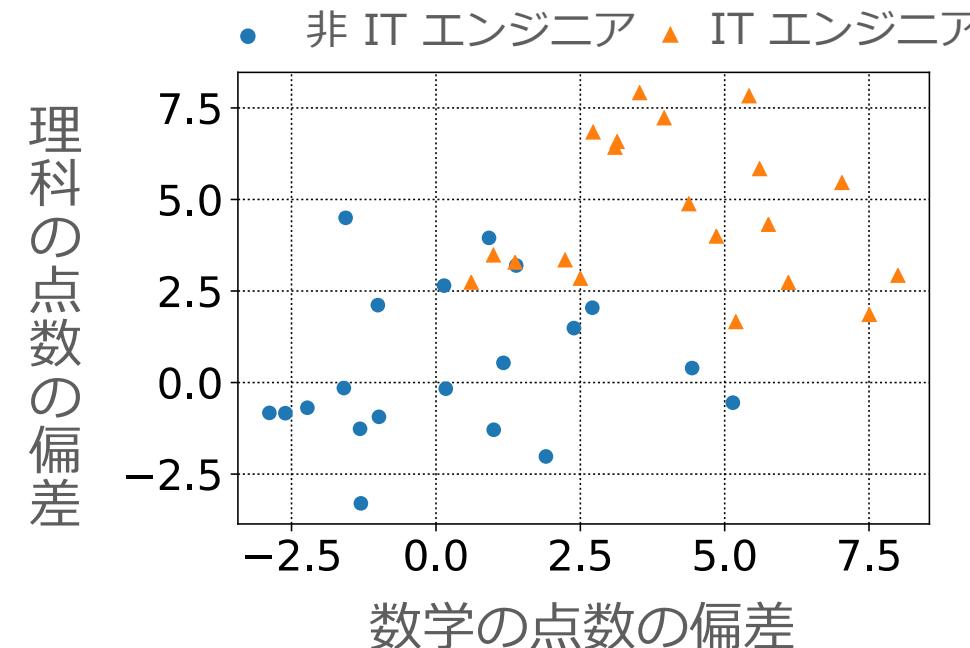
# 回帰と分類

---

- 回帰：数値を予測すること
  - ・ 例) 来客数からある商品の売上を予測
- 正解データは連続値（連続変数）



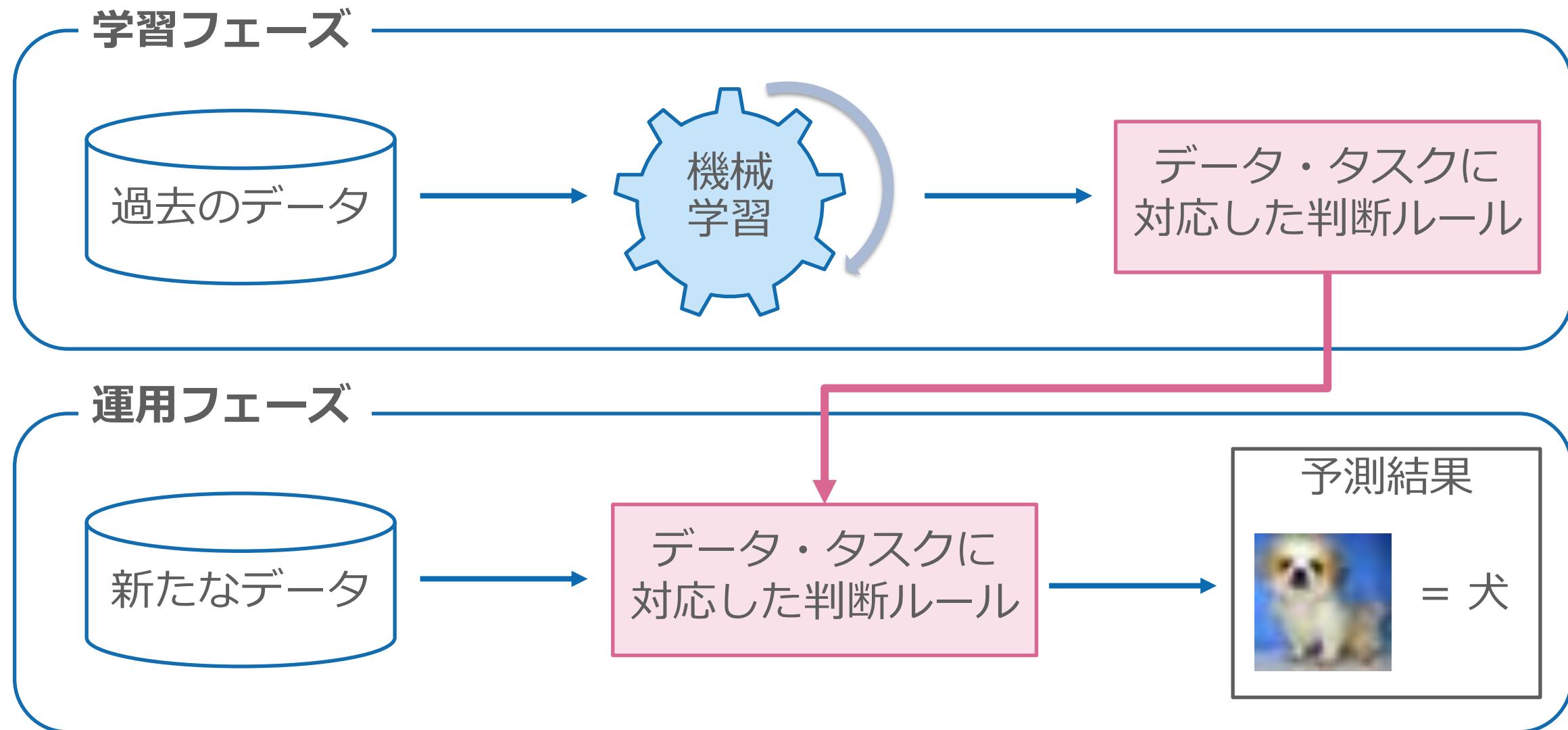
- 分類：カテゴリ（事柄の種類）を予測すること
  - 例) 理科と数学の偏差から将来 IT エンジニアになるか予測
- 正解データは離散値（離散変数）
- カテゴリ数が 2 の場合、2 クラス分類もしくは**二値分類**と呼ばれる
- 予測するカテゴリの数が 3 以上になると**多クラス分類**と呼ばれる

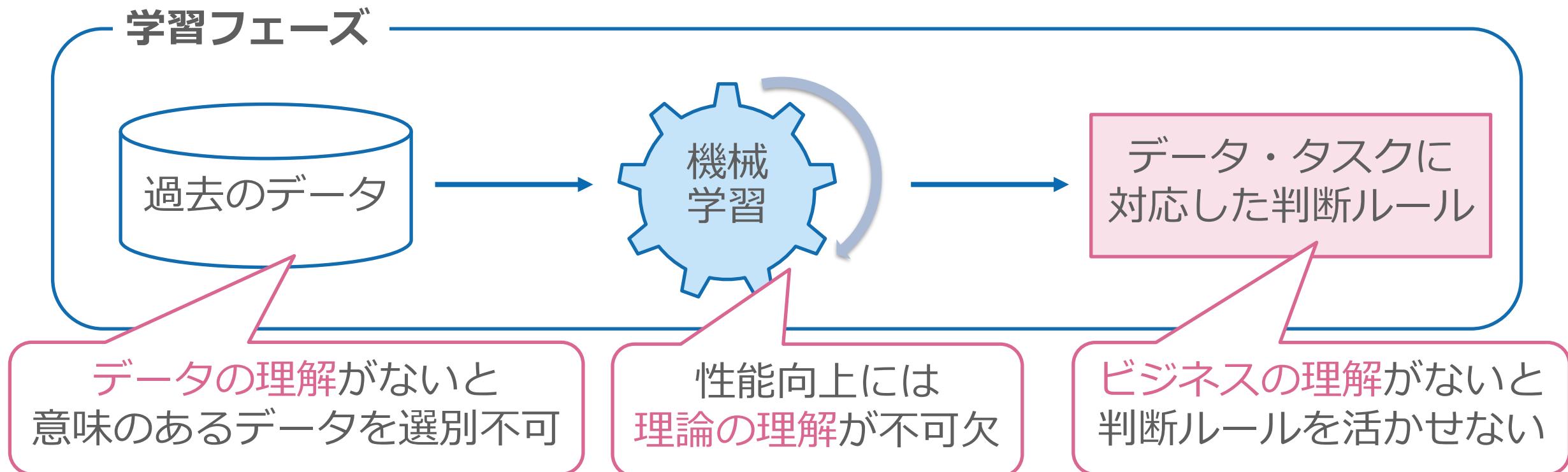


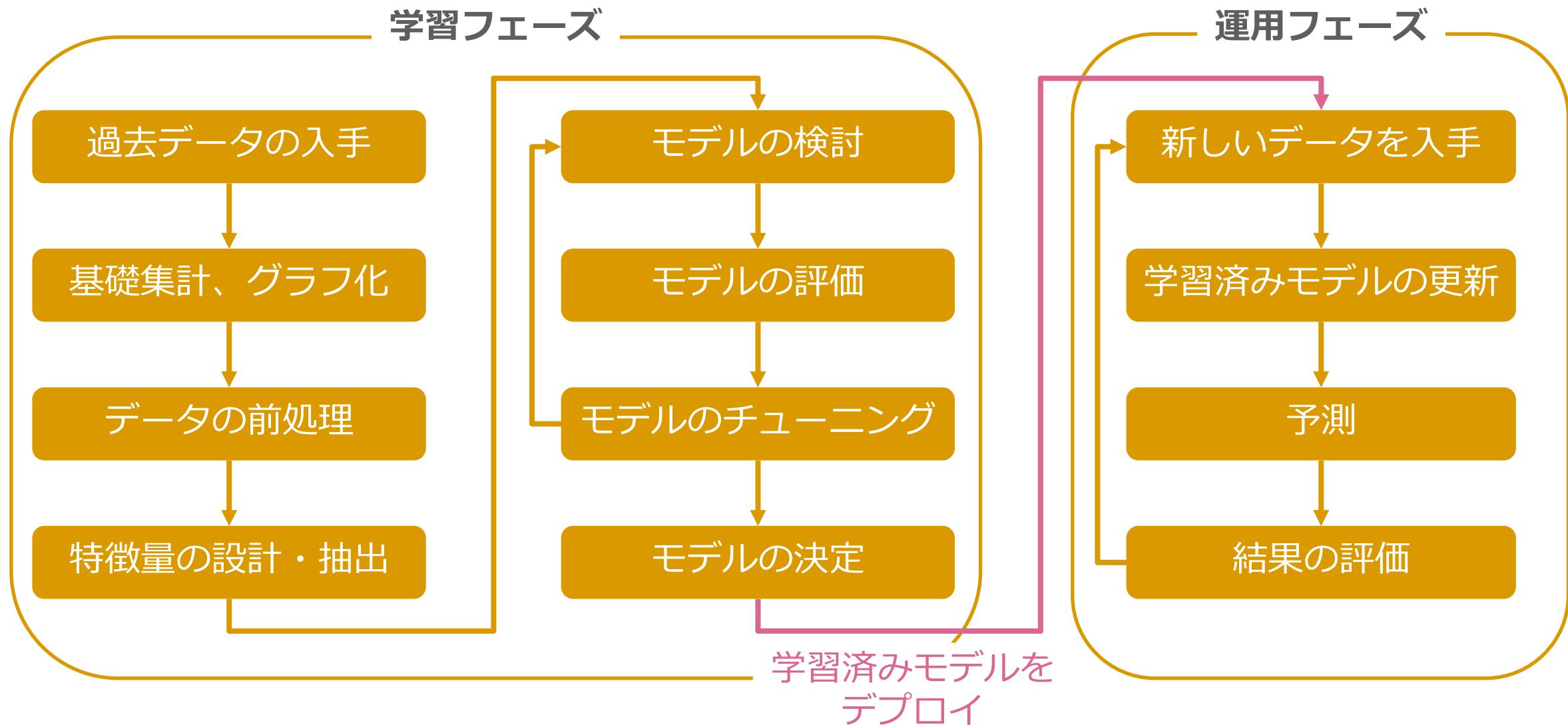
## **機械学習モデルの構築・運用の流れ**

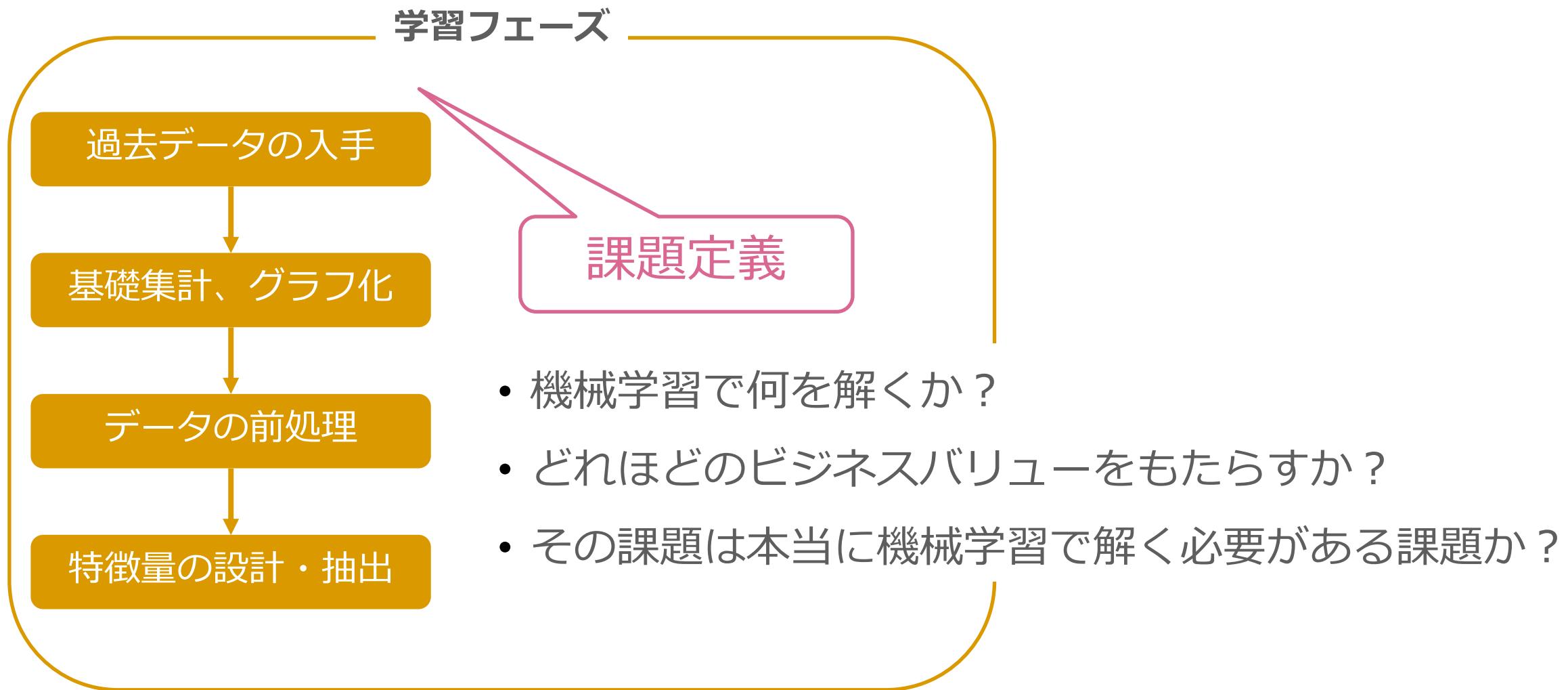
---

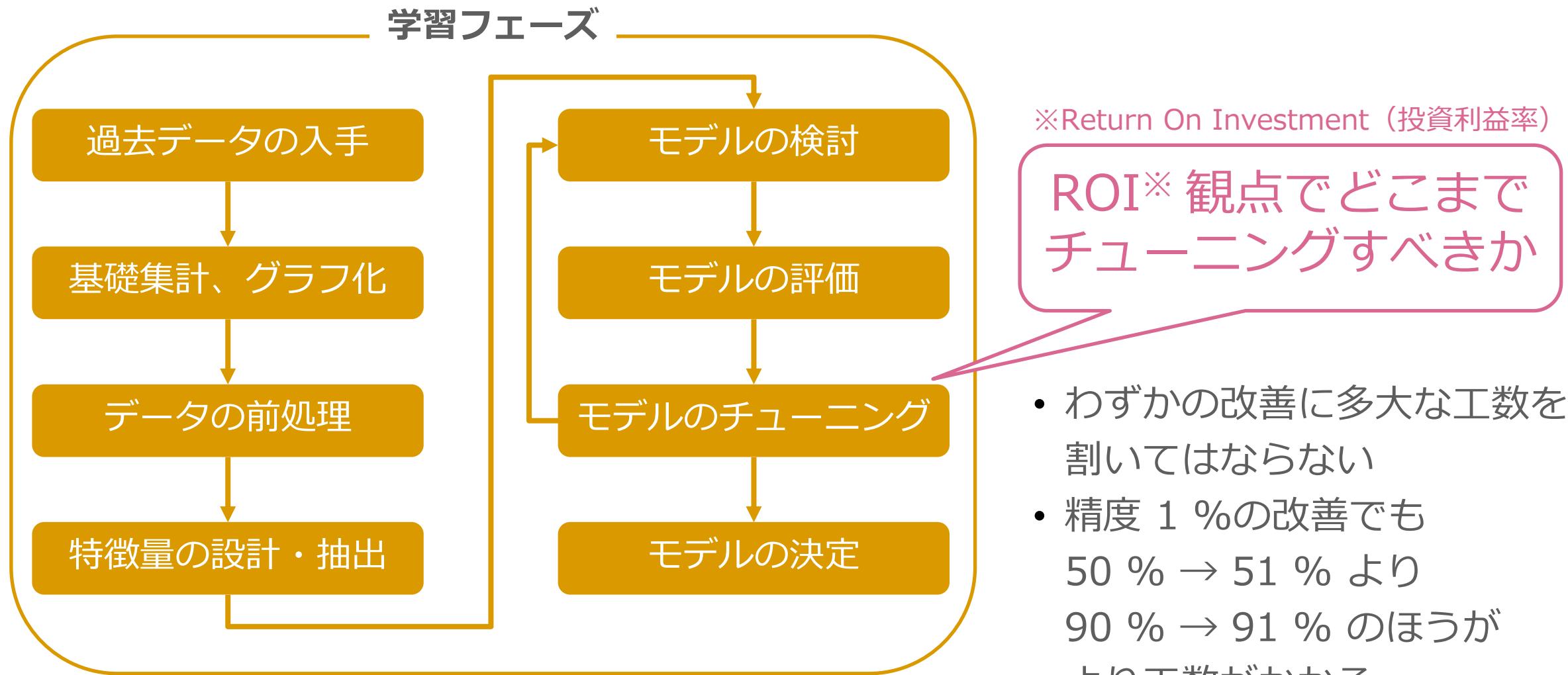
# 機械学習モデルの構築・運用の流れ | 概要









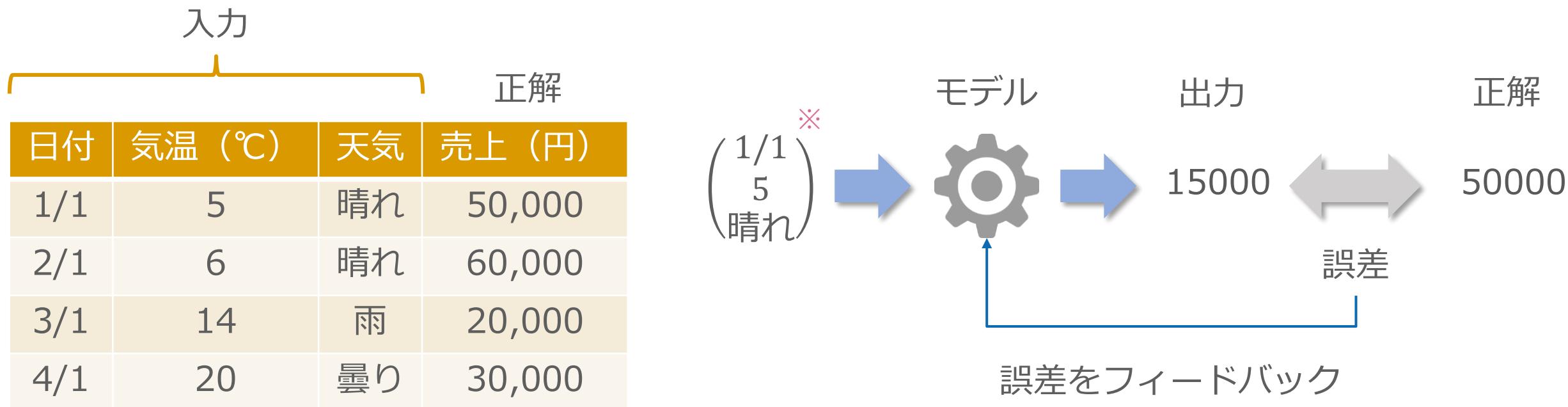


## 機械学習手法の種類

---

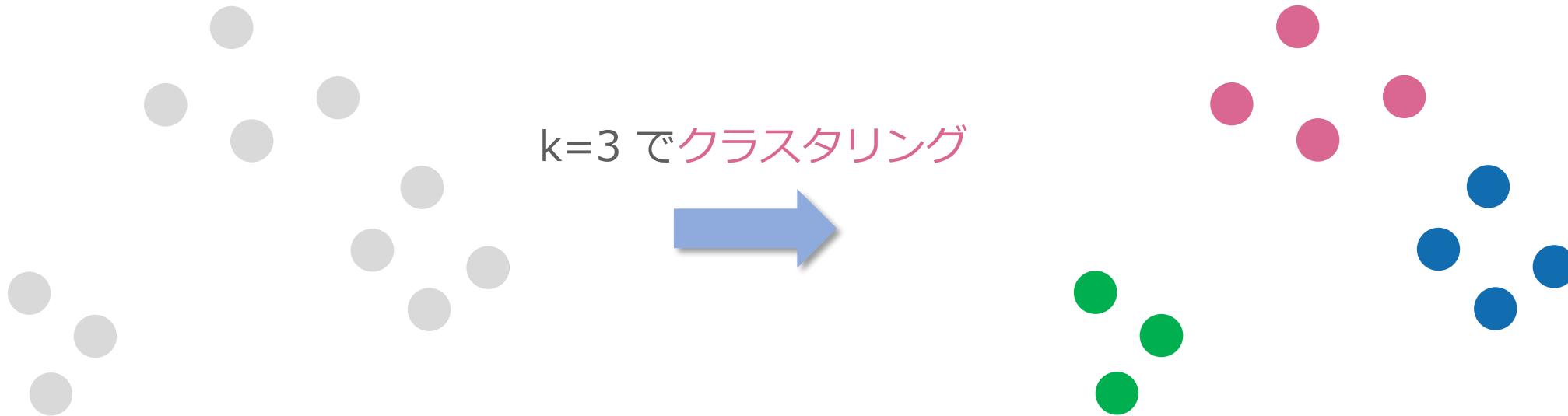
- 教師あり学習
- 教師なし学習
- 強化学習

- 入力と対応する正しい出力（正解, 教師）をモデルに与え、入出力の関係性を自動的に発見させる手法
- モデルは出力を正解に近づけるように、モデル自身が持つパラメータを変更



※実際にモデルに入力する際は、「日付」や「天気」の情報は適切な数値に置き換える必要がある

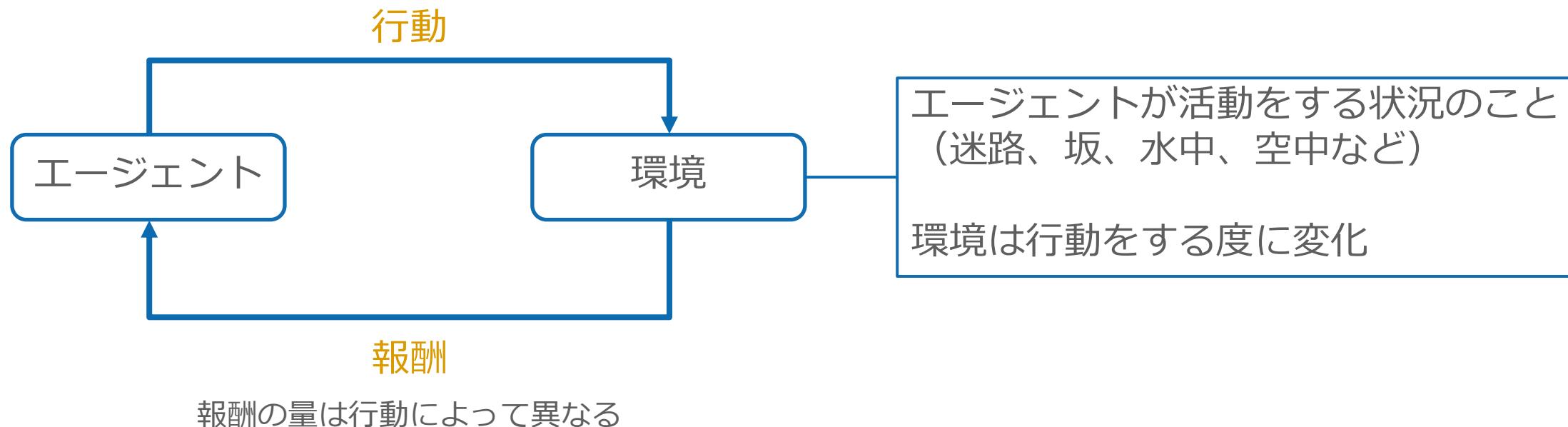
- アルゴリズムに従ってデータの関係性や構造を捉える手法
- 教師「なし」であるため、入力と正解という組みは存在しない



クラスタリングは教師なし学習の手法の 1 つ  
データからクラスタと呼ばれるデータのまとまりを発見する  
 $k=3$  はデータを 3 つのクラスタに分けるという意味

アルゴリズムに従ってクラスタを発見しているだけであり  
「正解に従って分割する」といった仕組みではない

- 学習用データを与えることなく、モデル（エージェント）に試作錯誤を繰り返し行わせることにより目的を達成させる手法
  - ・ 学習用データを与えるかわりに、行動の良さを評価できる報酬を与える
  - ・ モデルは報酬の総和を最大化するように試行錯誤を行う
- 囲碁 AI や歩行ロボットの開発に応用されている



## **教師あり学習とは**

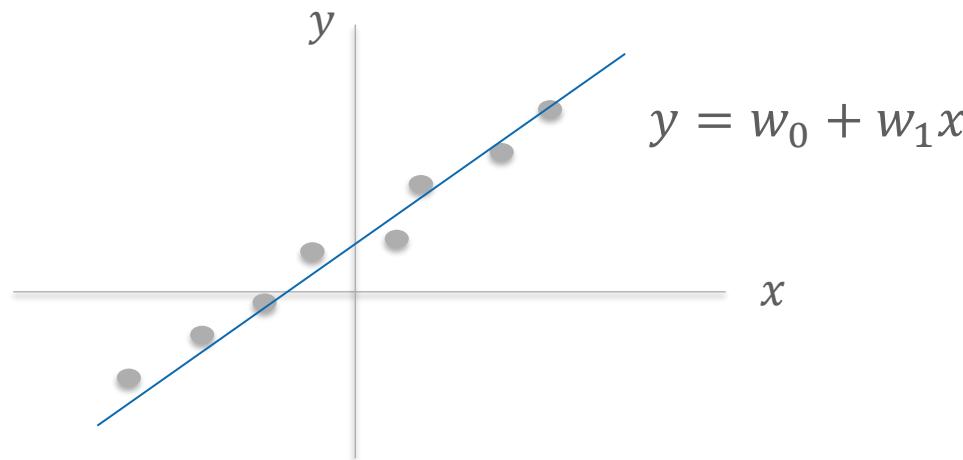
---

- 開発者が用意した入力と正解のデータを用いて、モデルに入力と正解の関係性を獲得させる方法



入力と出力に関連した用語で、**目的変数**と**説明変数**という用語がある  
**目的変数**は予測したい変数、**説明変数**は予測するために利用する変数という意味  
モデル構築時における変数の利用用途を指示する用語である

- 教師あり機械学習で利用するモデルは**関数**とみなすことができる
  - データ  $(x, y)$  における入出力の関係性に**モデル**という名の**関数を仮定**し、その**関数のパラメータ**を推定
  - 例)  $(x, y)$  に対して  $y = f(x) = w_0 + w_1x$  を仮定するのであれば。パラメータは  $w_0$  と  $w_1$
- 上手くパラメータを決定できれば、未知のデータに対する予測が可能！



$y$  が予測したい変数であれば、  
 $x$  の値を入れることで、新たな  $y$  の値を予測可能！  
 $(x, y)$  の動きを捉えた**関数**のおかげ！

## ■ モデルとして様々なものが提案されている

アルゴリズム	利用用途	線形モデル？	備考
線形回帰	回帰	Yes	シンプルな線形モデル
ロジスティック回帰	分類	Yes	「回帰」という名前だが、基本は分類のモデル (回帰の結果を分類に応用するため)
サポートベクターマシン	回帰、分類	No	計算量が非常に大きいが、 高性能が出やすい
k近傍法	回帰、分類	No	学習を行う必要がない
決定木	回帰、分類	No	正規化や標準化といった前処理を しなくても良い
ランダムフォレスト	回帰、分類	No	
ニューラルネットワーク	回帰、分類	No	深層学習※モデルに発展

※ディープニューラルネットワーク（DNN）は、深層学習の方法論のひとつ

## 線形モデルと非線形モデル

---

## ■ 目的変数 $y$ と説明変数 $x$ の関係を、以下の式で表現するモデル

$$y = f(x) = w_0 + w_1x_1 + w_2x_2 + \cdots = \mathbf{w}\mathbf{x}$$

## ■ 線形モデルの例

- $y = w_0 + w_1x_1 + w_2x_2$ 
  - 目的変数  $y$  を、説明変数  $(1, x_1, x_2)$  にパラメータ  $w$  をかけて足し合わせることで表現
  - 「1」は切片  $w_0$  を得るためにダミーの説明変数
- $y = w_0 + w_1x_1 + w_2x_1^2$ 
  - 目的変数  $y$  を、説明変数  $(1, x_1, x_1^2)$  にパラメータ  $w$  をかけて足し合わせることで表現
  - $x_1^2$  は  $x_1$  を二乗して新たに作り上げた説明変数

## ■ 線形モデル以外のモデル

- 木モデル、ニューラルネットワークなどの手法により構築可能

## ■ 非線形モデルの例

$$y = f(w_1 + w_{11}x_1 + w_{12}x_2) + f(w_2 + w_{11}x_1 + w_{12}x_2)$$

$$f(x) = \frac{1}{1 + e^{-wx}}$$

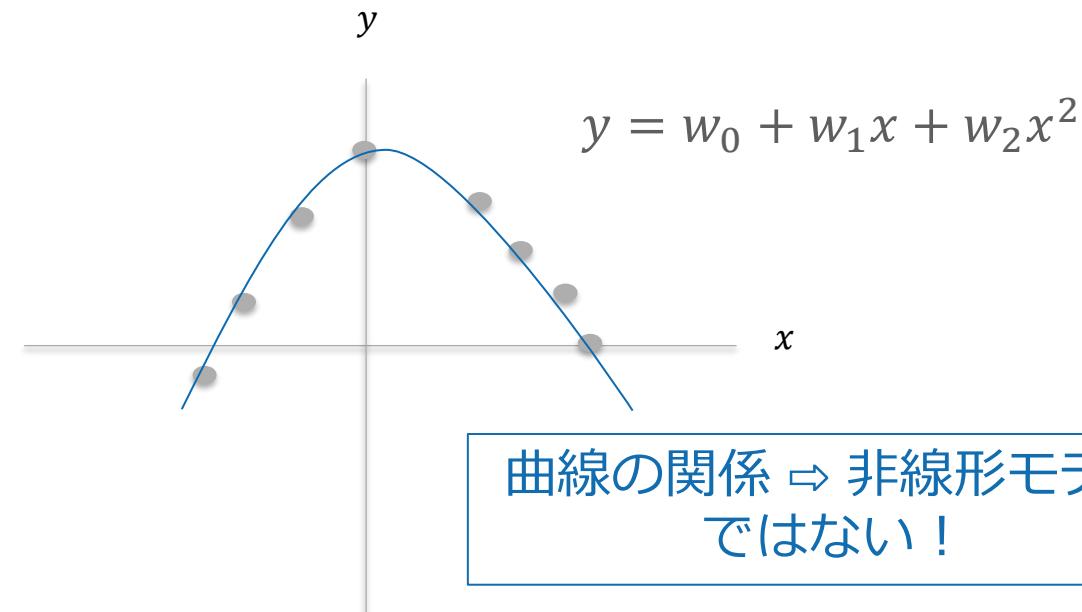
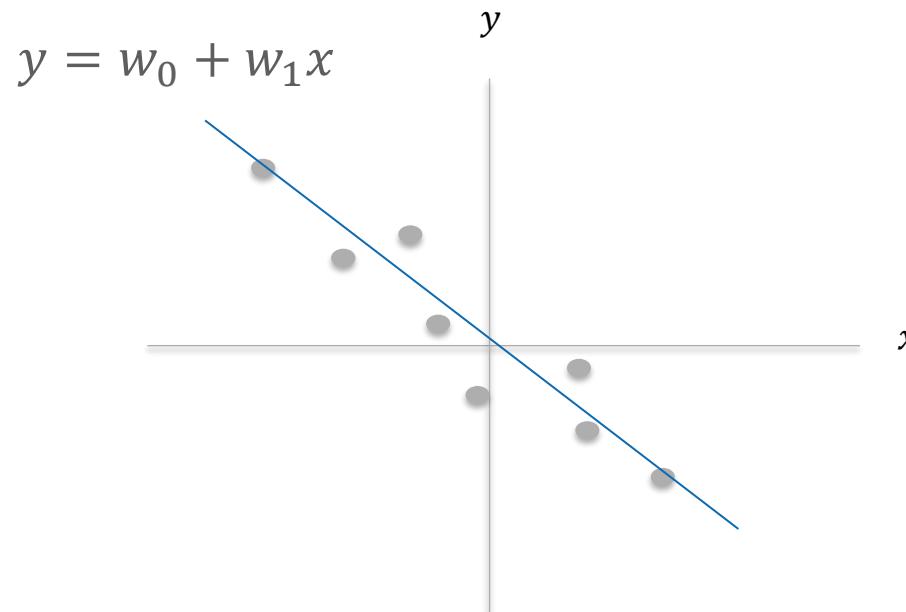
注意！

- 線形モデルで表現可能なデータ  $(x, y)$  に対しても非線形モデルは利用可能
- しかし、モデルが“あまりにも複雑な関数を表現可能である場合、モデルがデータ  $(x, y)$  に含まれるノイズ成分すらも表現してしまう可能性がある（過学習）  
→ その場合、非線形モデルでも性能が向上しにくい

# 線形モデル・非線形モデルどちらを使う？

## ■ データを可視化して、目的変数 $y$ と説明変数 $x$ との関係を把握

- $y = wx$  で表現できそうな  $y$  と  $x$  の関係が見つかれば、  
線形モデルでも十分、モデルに対する性能要件を満たすかもしれない



曲線の関係 ⇒ 非線形モデル  
ではない！

この場合、どちらも線形モデルで OK

## 本章のまとめ

---

- 人工知能と機械学習
- 回帰と分類
- 機械学習モデルの構築・運用の流れ
- 機械学習手法の種類
- 教師あり学習とは
- 線形モデルと非線形モデル

## ■ 人工知能

- 知的なコンピュータプログラムを作る科学と技術

## ■ 機械学習の目的

- データを用いてタスクを解くコンピュータプログラムを開発すること

## ■ 回帰と分類

- 回帰：連續変数（数値）を予測
- 分類：離散変数（カテゴリ）を予測

## ■ 機械学習モデルの構築・運用

- 学習フェーズ
- 運用フェーズ

## ■ 機械学習手法の種類

- 教師あり学習
- 教師なし学習
- 強化学習

## ■ 線形モデルと非線形モデル

- 線形モデル：関数  $y = wx$  で表現できるモデル
- 非線形モデル：関数  $y = wx$  で表現できないモデル

# 現場で使える 機械学習・データ分析基礎講座

第 2 章：教師あり学習の基礎

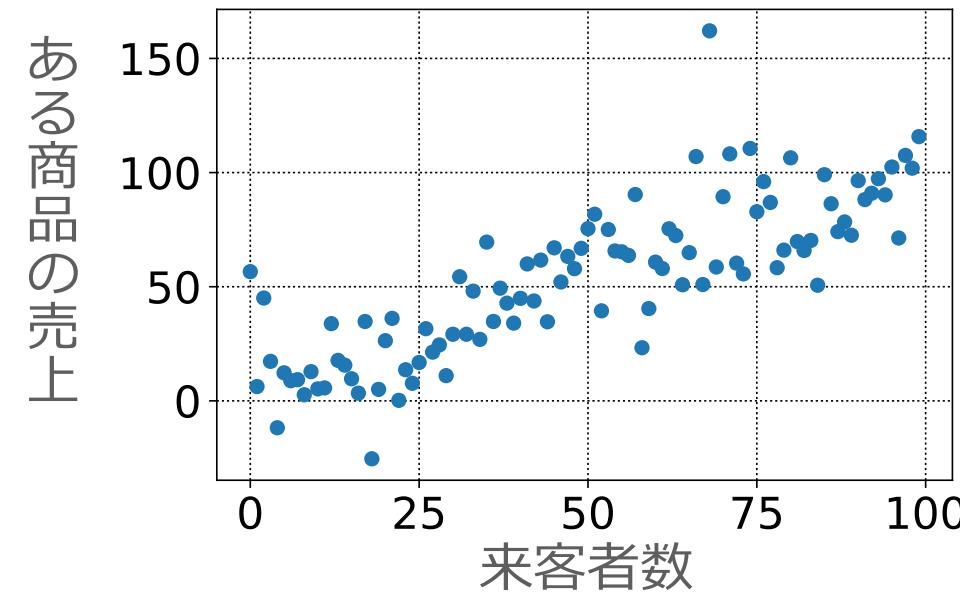
- 線形回帰モデル
- ロジスティック回帰モデル
- 多変量モデルへの拡張
- k近傍法
- ノートブック演習

- 線形回帰モデルの評価基準と最適化の方法を説明できる
- ロジスティック回帰モデルの評価基準と最適化の方法を説明できる
- 線形回帰モデルとロジスティック回帰モデルの出力を、多変量に拡張する方法を説明できる
- k近傍法を用いた回帰・分類の仕組みを説明できる

# 線形回帰モデル

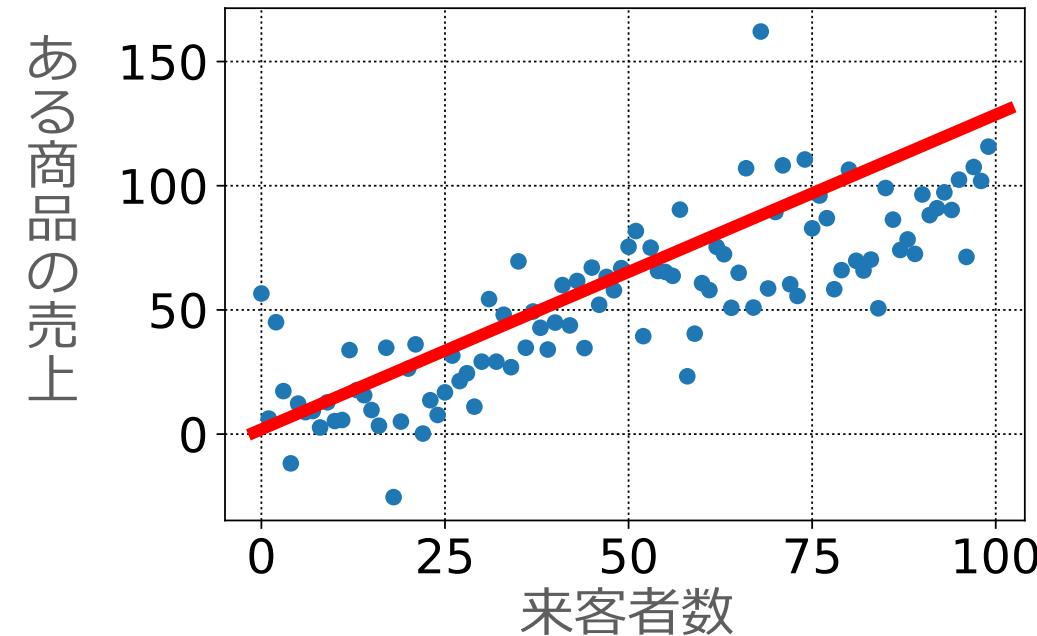
---

- ある商品の売上と来店客数の関係を調査



- 来店客数を変数として商品の売上を予測するモデルを構築したい

- 直線を引ければ売上を予測できそう？



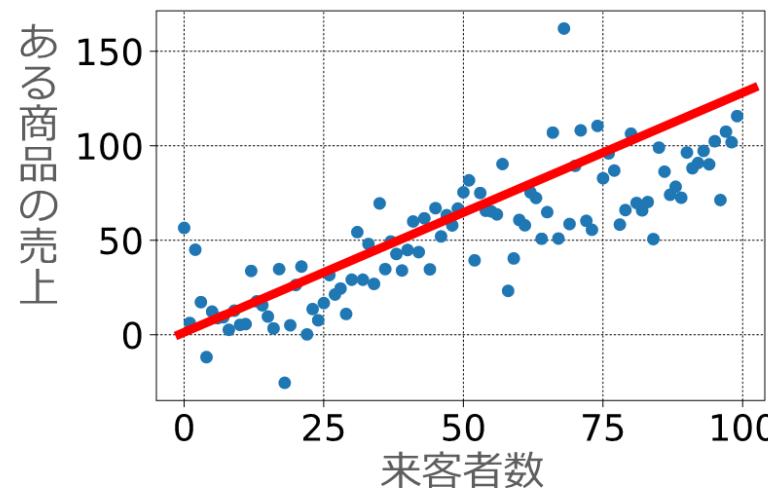
- 直線を引くモデルの 1 つが線形回帰モデル

# 線形回帰モデル (Linear Regression Model)

$$\hat{y} = f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \mathbf{x} = w_0 + w_1 x_1$$

$$\mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} 1 \\ x_1 \end{pmatrix}$$

; を用いてモデルへの入力とパラメータを区切っている  
 $\hat{y}$  は  $y$  の真の値に対する推定値という意味

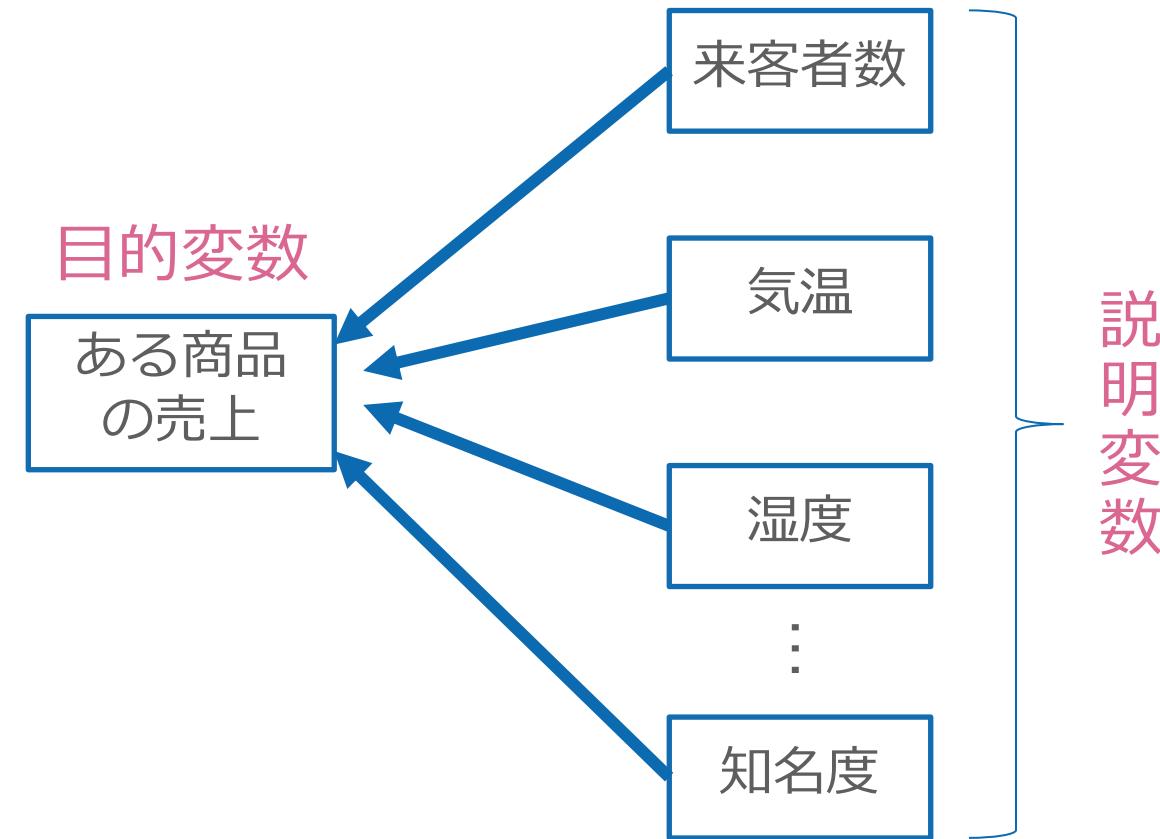


$w_0$  と  $w_1$  を上手く定めれば、  
データの動きに沿った直線を引くことが可能！

モデルのパラメータ  $w$  は、文脈によって  
変数の「重み」や「係数」とも呼ばれる

# 線形回帰 | 説明変数が多次元の場合

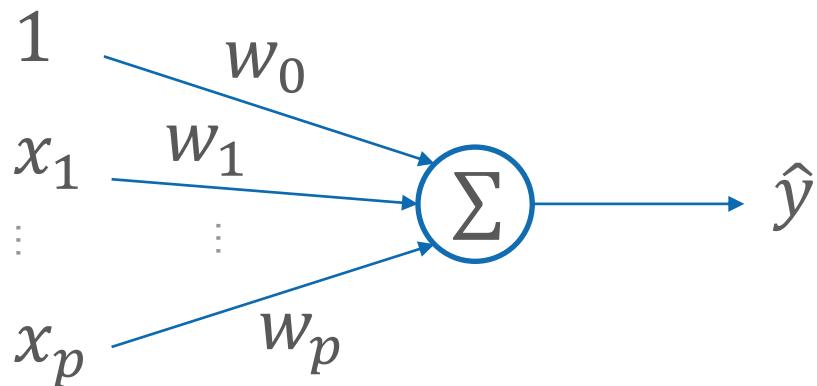
- 売上に関係がありそうな要因（説明変数）は来客者数以外にも色々ありそう
- 線形回帰モデルの考え方で、他の要因も考慮して予測できないだろうか？



$$\hat{y} = f(\mathbf{x}; \mathbf{w}) = \mathbf{w}^T \mathbf{x} = \sum_{i=0}^p w_i x_i = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_p x_p$$

$$\mathbf{w} = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_p \end{pmatrix} \quad \mathbf{x} = \begin{pmatrix} 1 \\ x_1 \\ \vdots \\ x_p \end{pmatrix}$$

線形回帰モデルの概念図



各説明変数に重みづけして  
足し合わせるシンプルなモデル  
学習データをよく説明できる  
ように重み  $w$  を学習

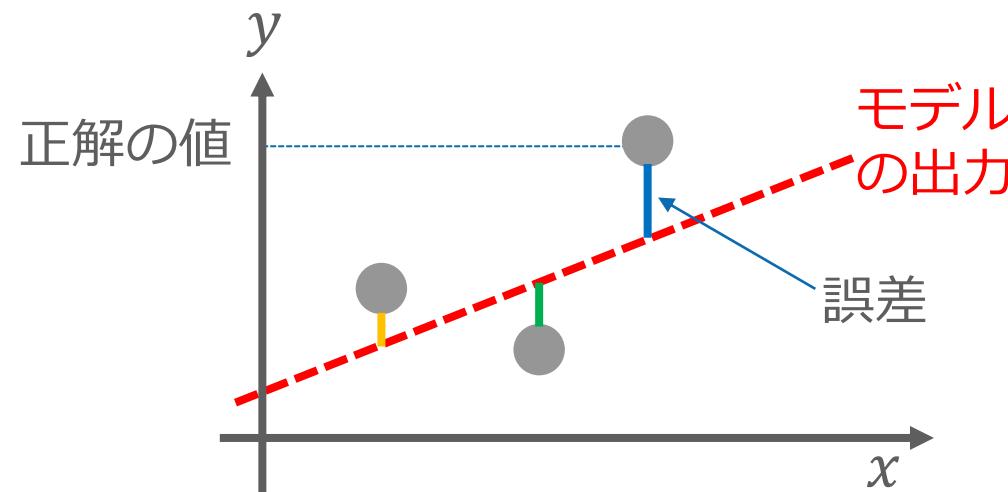
## ■ 良い $w$ とは？

- 今手にしている入力  $x$  に対して、正しい  $y$  の値を返すことができるような  $w$
- モデルの出力（=予測値）と正解との誤差を定量化すれば、 $w$  の良し悪し（=モデルの良し悪し）を把握可能
- 誤差が最小のモデルが最良の  $w$  を持ったモデル

## ■ 具体例

- 過去のデータでは来客者数が 50 の時、売上の値は 60 であったとする
- モデルを  $y = w_0 + w_1x$  とすると、 $(w_0, w_1) = \left(10, \frac{1}{5}\right)$  よりも  $(w_0, w_1) = \left(20, \frac{4}{5}\right)$  の方が良い
  - $(w_0, w_1) = \left(10, \frac{1}{5}\right)$  の場合、予測値  $\hat{y} = 10 + \frac{1}{5} \times 50 = 20$  正解の値 60 との誤差 40
  - $(w_0, w_1) = \left(20, \frac{4}{5}\right)$  の場合、予測値  $\hat{y} = 20 + \frac{4}{5} \times 50 = 60$  正解の値 60 との誤差 0

- モデルの出力と正解データとの間の二乗誤差を用いてモデルの性能を評価
- 二乗誤差：各正解データとの差の二乗の総和
- 各パラメータの値におけるモデルの性能を定量化した指標を評価基準と呼ぶ



二乗誤差の考え方  
マイナスの誤差がプラスになるよう二乗  
 $E_D = \frac{1}{2} (\text{□} + \text{+} \text{□} + \text{+} \text{□})$   
微分したときに係数が  
打ち消されるよう 2 で割る

- $N$  個の学習データのもとで二乗誤差の総和  $E_D$  は次式で定義される

$$E_D = \frac{1}{2} \sum_{n=1}^N \left( \underbrace{f(x^{(n)}; w)}_{y \text{ の推定値}} - \underbrace{y^{(n)}}_{y \text{ の正解値}} \right)^2$$

$x^{(n)}$  :  $n$  番目のデータの説明変数  
 $y^{(n)}$  :  $n$  番目のデータの目的変数  
= 正解データ

- $E_D$  は  $w$  の関数
  - $w$  を変えるとモデルの二乗誤差は変わる
  - $E_D$  を最小にするような  $w$  を求める！
- モデルとそのモデルの良し悪しを定める指標を設定した後は、  
関数に最小値を与える  $w$  を求める**数理最適化問題**となる！
- 最適化によりパラメータを求めることを機械学習では**学習**と呼んでいる！

- 二乗誤差を最小化してパラメータを求めるのことを**最小二乗法**と呼ぶ
- 線形回帰モデルの場合  
二乗誤差を最小化する  $w$  は**解析的に** (=手計算で) 導出可能！
- 二乗誤差が最小になる条件式  $\frac{\partial E_D}{\partial w} = 0$  を  $w$  について解くと…

最適なパラメータ

$$w^* = (\Phi^T \Phi)^{-1} \Phi^T y$$

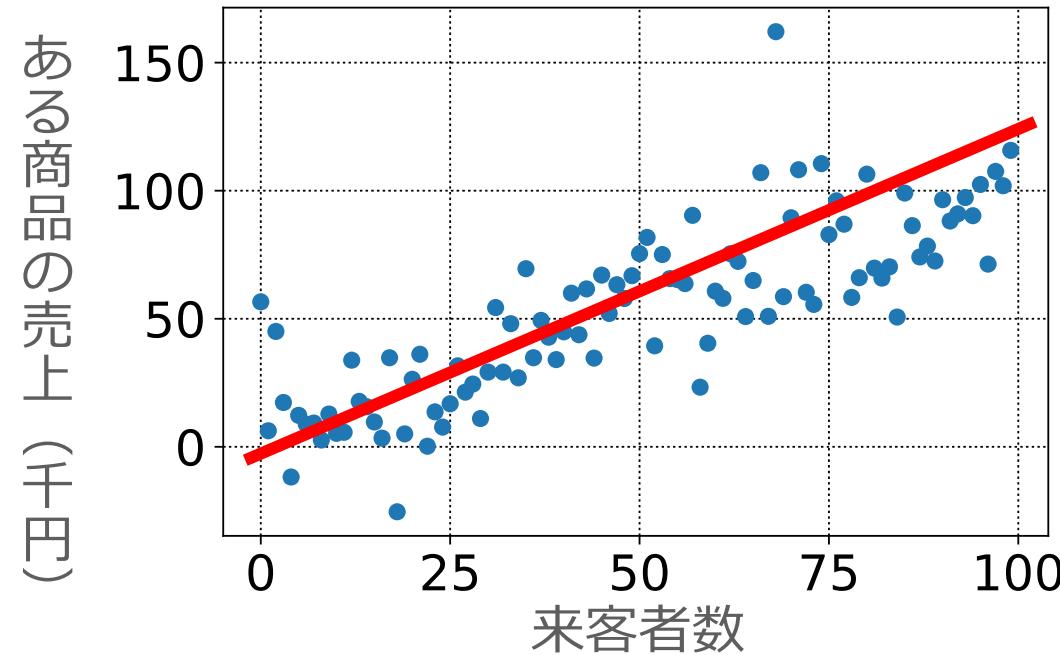
計画行列

$$\Phi = \begin{pmatrix} \mathbf{x}^{(1)\top} \\ \vdots \\ \mathbf{x}^{(N)\top} \end{pmatrix} = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_p^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_p^{(N)} \end{pmatrix}$$

2\_notebook/appendix/math\_LinearRegression.ipynb  
に導出過程を記載しています

$$y = (y^{(1)}, y^{(2)}, \dots, y^{(N)})^T$$

- 説明変数が 1 つの場合、データは 2 次元の散布図、モデルは直線で表せる
  - ・ この直線は、来客者数 50 人なら売上は約 60 千円（6 万円）になるという関係を示唆

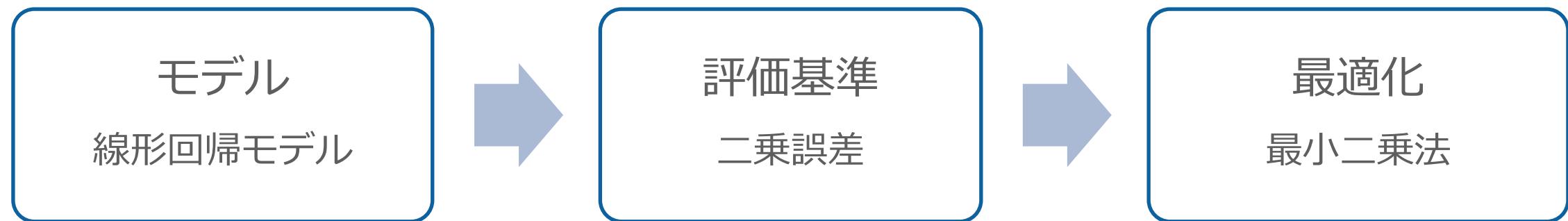


標準化については  
詳しくは第 6 章で扱う

- 説明変数が多次元の場合、説明変数と目的変数の両方を標準化しておくとモデルのパラメータ  $w$  は、各説明変数の目的変数に対する重要度を表す

## ■ 線形回帰モデル

- 目的変数と説明変数との間に直線の関係を仮定し、パラメータを求める
- 活用例
  - 1日の売上を目的変数、来客者数を説明変数に設定して、その関係性を表す直線を求める
  - この直線が求まれば、売上の値を予測可能

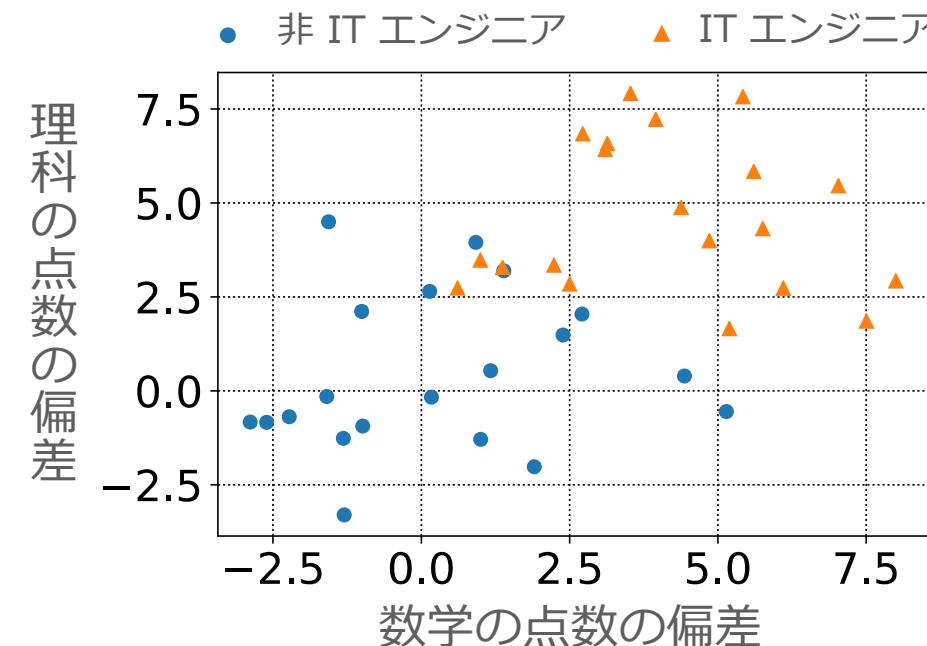


## ロジスティック回帰モデル

---

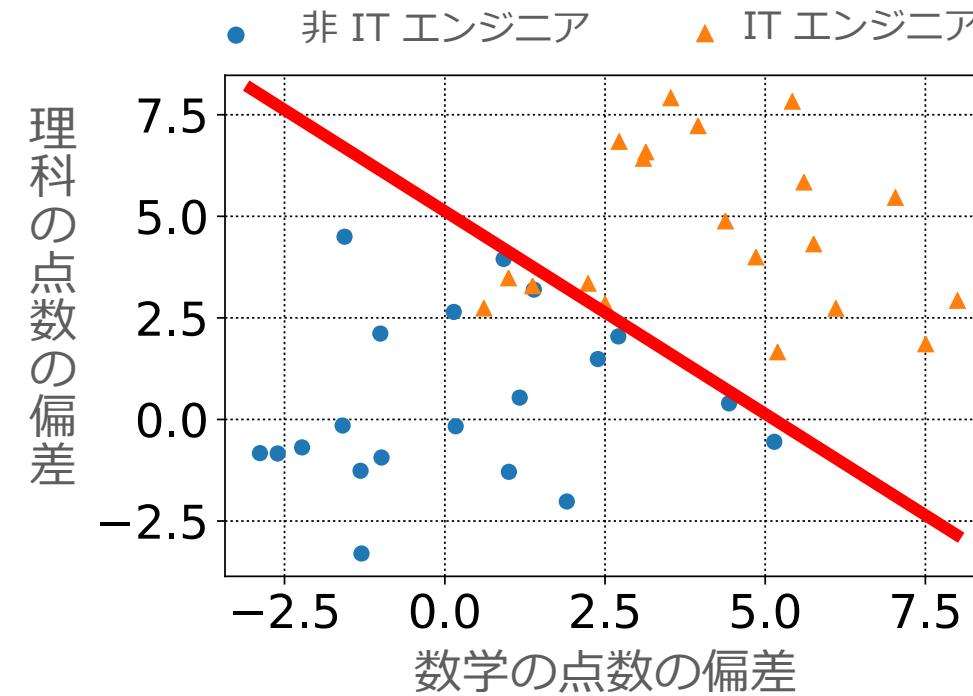
# ロジスティック回帰 | 問題の例

- ある IT エンジニアと非 IT エンジニアを対象として、学生時代の理科および数学の偏差を調査したところ、下図の結果が得られた
  - このデータから、ある学生が将来 IT エンジニアになるかを予測するモデルをつくりたい
  - どのようなモデルをつくればいいか？

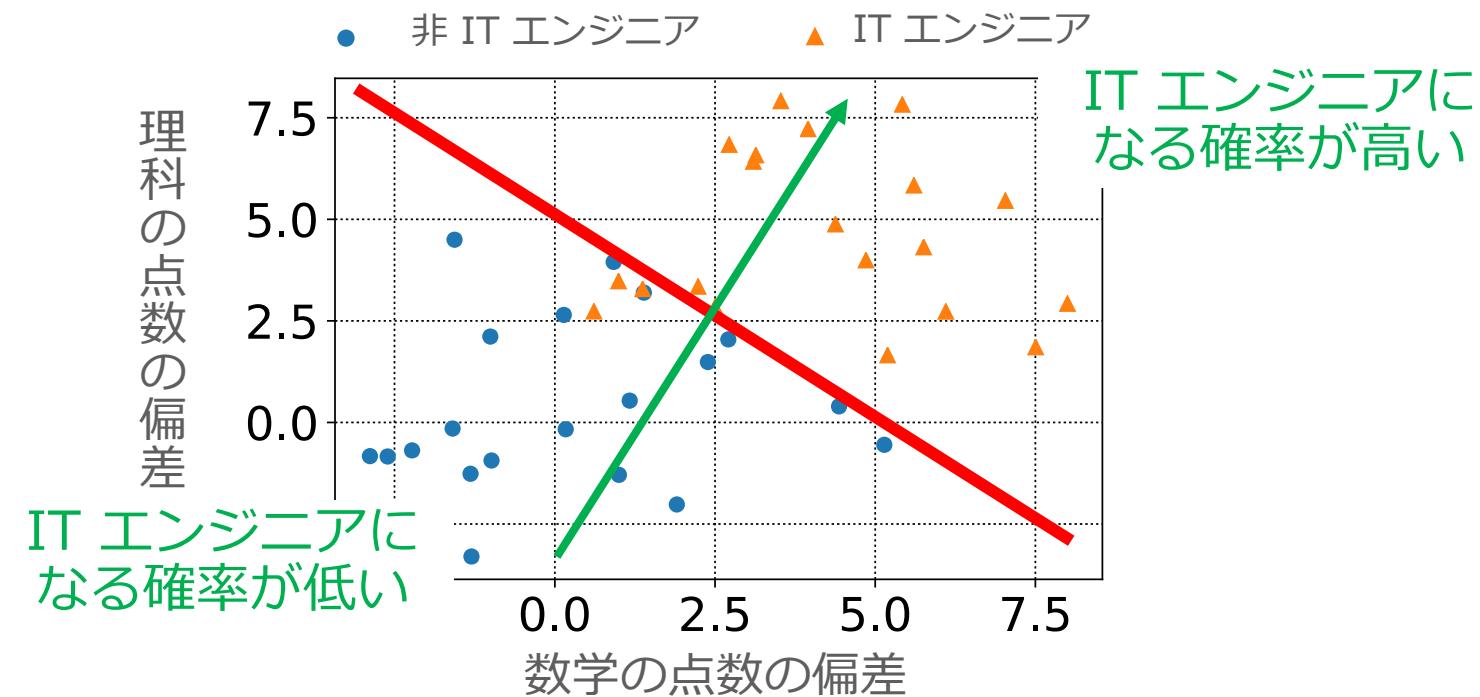


## ■ 直線の境界線を引くのがよさそう

- これには、線形回帰モデルの考え方方が使えるか？



- さらに、IT エンジニアになる確率（＝確信度）を表現できるとうれしい



## ■ データセットを構築

- 正解ラベルの定義
  - 「IT エンジニアになる」というラベル： 1
  - 「IT エンジニアにならない」というラベル： 0

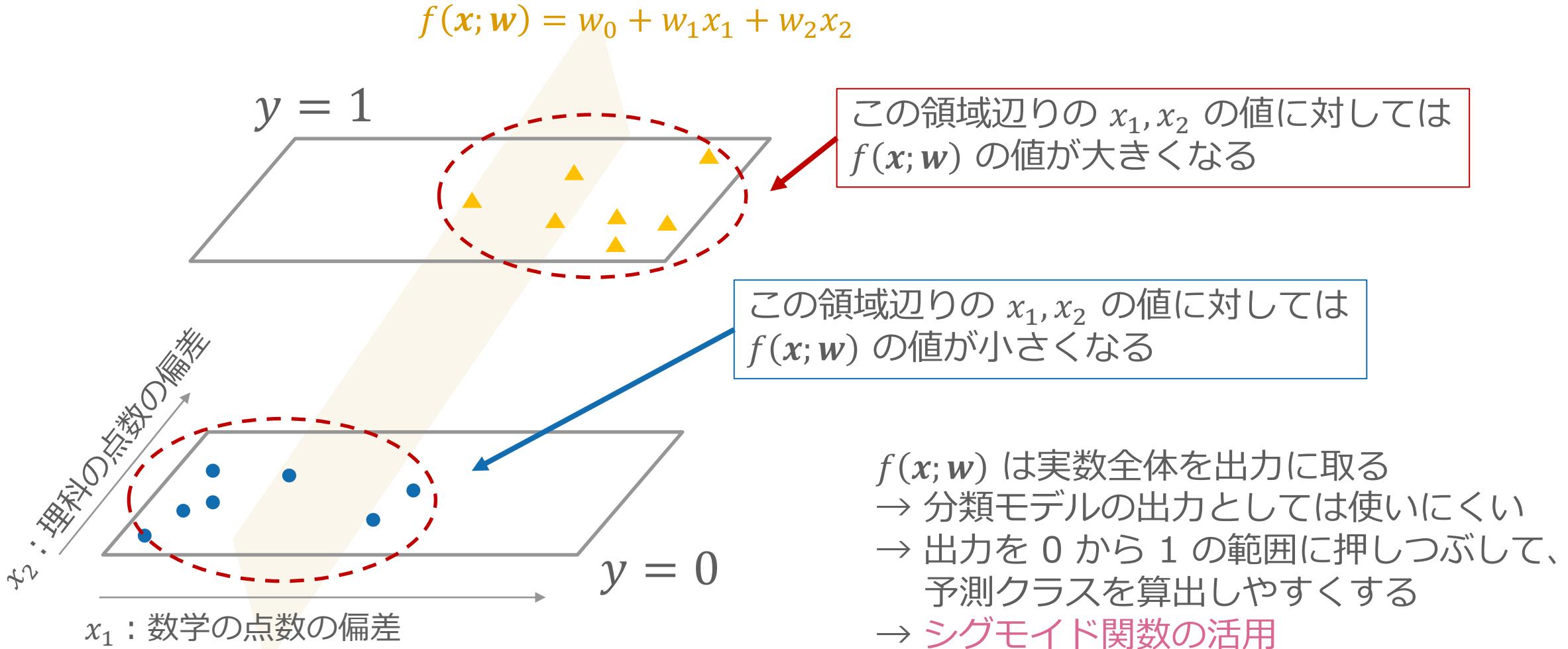
入力

正解

数学の点数の偏差	理科の点数の偏差	IT エンジニアになるか？
1.0	1.0	0
1.2	0.8	0
4.0	5.0	1

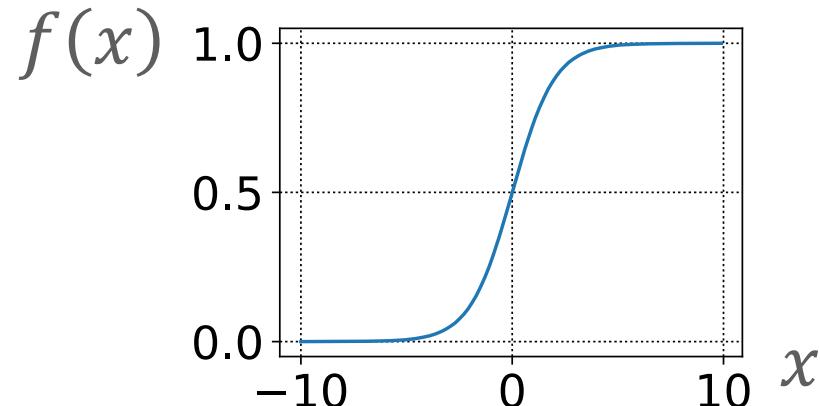
# ロジスティック回帰 | 線形回帰モデルをどう使う？

線形回帰モデル  $f(x; w) = w_0 + w_1x_1 + w_2x_2$  で回帰してはどうか？



シグモイド関数

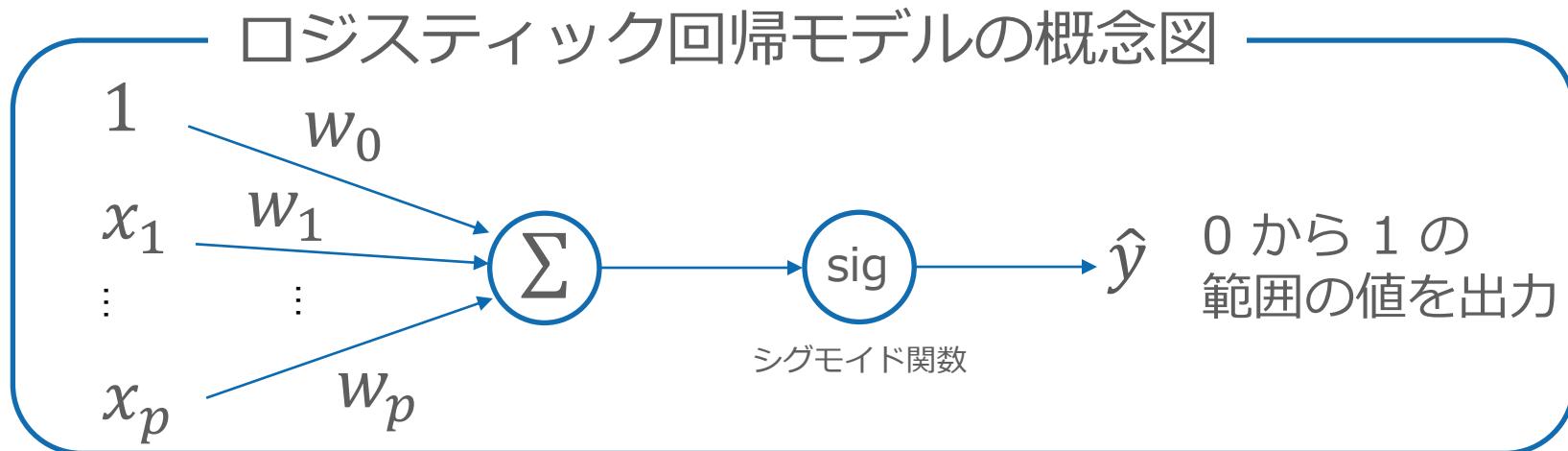
$$f(x) = \frac{1}{1 + e^{-x}}$$



- 線形回帰モデルの出力値をシグモイド関数に通すことで出力値を 0 から 1 の範囲に変換可能
- 出力値に対して、適当な閾値を定めてクラスラベルを決定する
  - 例) 閾値 0.7 とした場合：  
シグモイド関数を通した結果が 0.9 → クラス 1  
シグモイド関数を通した結果が 0.3 → クラス 0

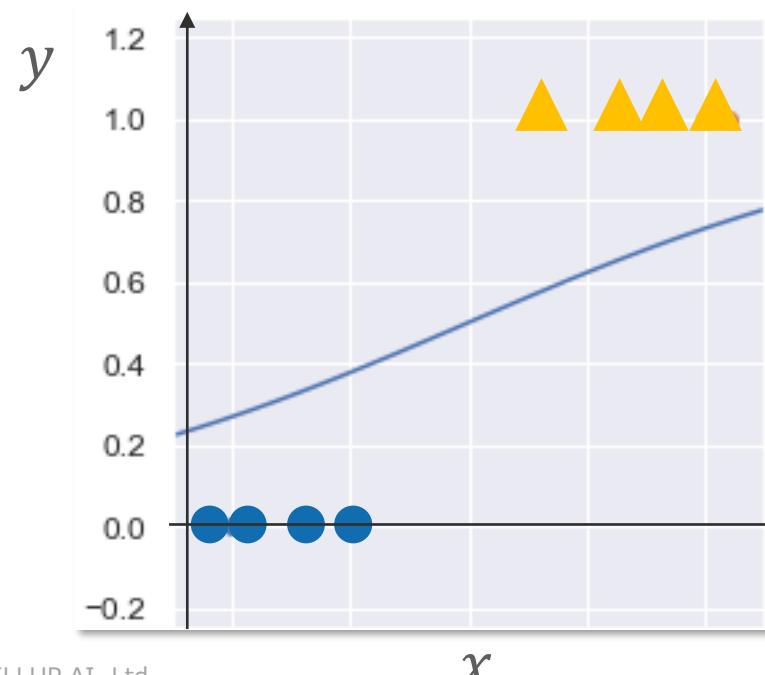
$$\hat{y} = f(\mathbf{x}; \mathbf{w}) = \frac{1}{1 + e^{-\sum_{i=0}^p w_i x_i}} = \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}}$$

線形回帰の出力を  
シグモイド関数に入力している

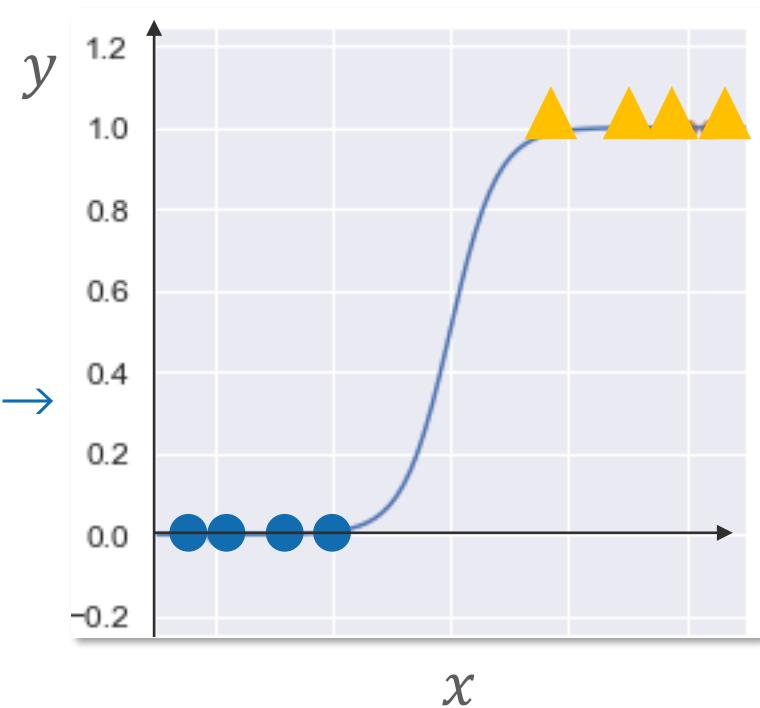


# ロジスティック回帰 | どうやって $w$ を定める？

- 説明をシンプルにするため 1 次元のロジスティック回帰モデルを考える
- モデルに  $x$  を入力し、シグモイド関数の出力と正解データ  $y$  を比べる
  - これは  $(x, y)$  のデータに対してシグモイド関数を当てはめていることと同じ
- データに最も上手く当てはまるシグモイド関数の形状を探せばよい
  - その形状を変えるのが、線形回帰モデルのパラメータ  $w$



← 傾斜の立ち上がりを  
もっと急にしたい！  
上手く当てはまっている！→



- 尤度：データとモデルの当てはまり度合いを定量化する指標の 1 つ

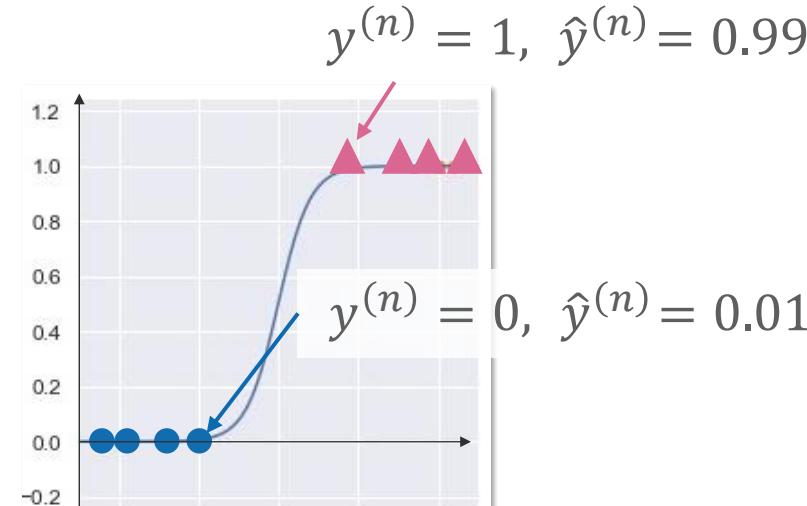
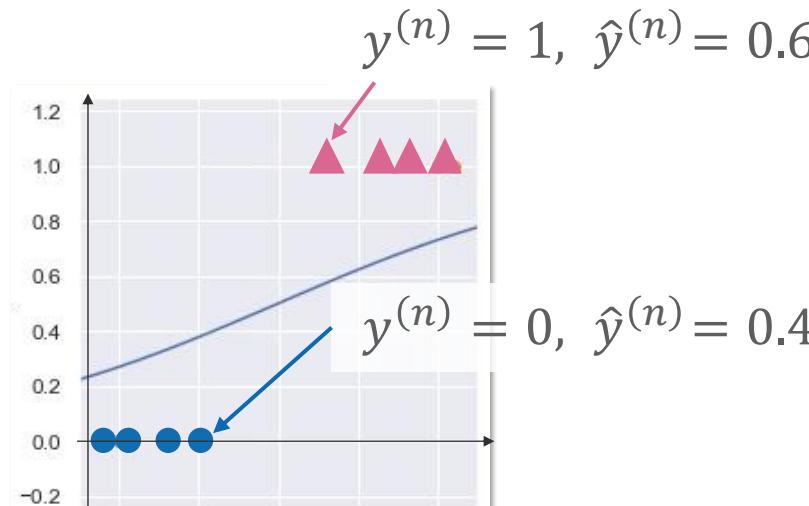
$$P(\mathbf{w}) = \prod_{n=1}^N \frac{\hat{y}^{(n)} y^{(n)}}{\text{クラス 1}} \frac{(1 - \hat{y}^{(n)})^{1-y^{(n)}}}{\text{クラス 0}}$$

$\hat{y}^{(n)}$  :  $n$  番目の入力に対する出力

$y^{(n)}$  :  $n$  番目の入力に対する正解

$\hat{y}^{(n)}$	$y^{(n)}$	$\hat{y}^{(n)} y^{(n)} (1 - \hat{y}^{(n)})^{1-y^{(n)}}$
0.99	1	0.99
0.6	1	0.6
0.01	0	0.99
0.4	0	0.6

正解と出力が近いほど  
1 に近い値を取る



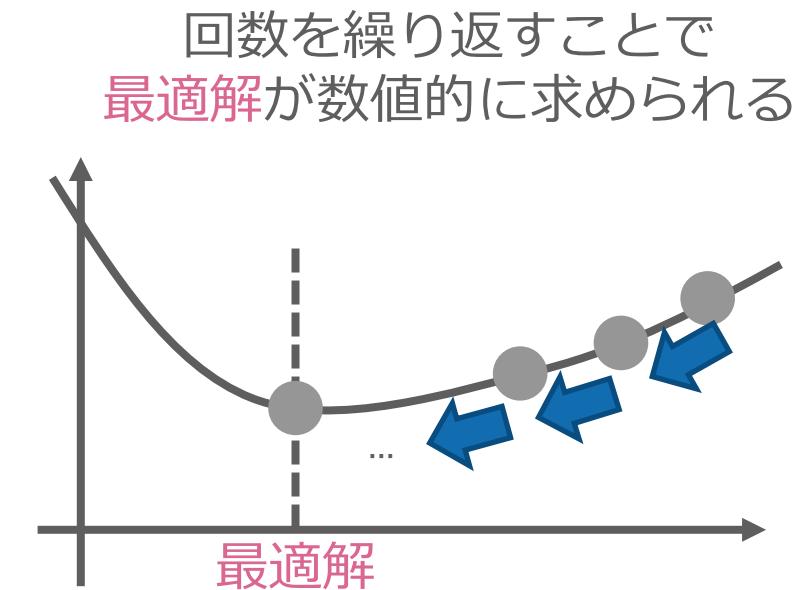
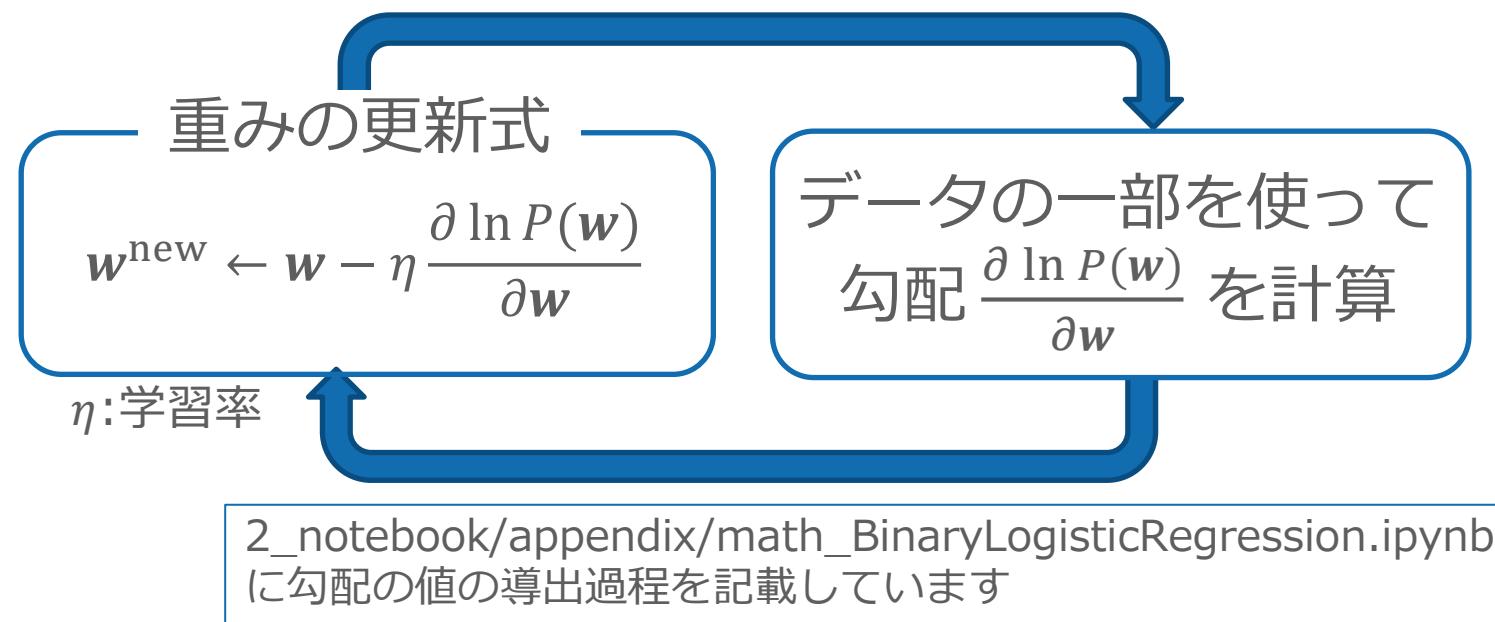
- 尤度が最大になるときの  $w$  が、シグモイド関数が最もデータによく当てはまっているときの  $w$
- 実用上は尤度を対数変換した**対数尤度**を最大化

- 尤度には**小数の掛け算**が大量に含まれるため、対数変換して**アンダーフロー**を防ぐ

$$\ln P(w) = \sum_{n=1}^N \{y^{(n)} \ln \hat{y}^{(n)} + (1 - y^{(n)}) \ln(1 - \hat{y}^{(n)})\}$$

- ロジスティック回帰では**対数尤度を評価基準として最適化を実施**
  - このことを、**最尤推定** (Maximum Likelihood Estimation) と呼ぶ
- 実際には対数尤度にマイナスを掛けて、それを基準として**最小化**を実施
  - 負の対数尤度のことを、交差エントロピー (Cross Entropy) と呼ぶ

- 負の対数尤度が最小となる条件式  $\frac{\partial \ln P(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{0}$  を解いて、最適な重みを求めたい  
→ 実は解析解は求まらない…
- そこで、繰り返し計算によって最適な値を探索する
- 勾配  $\frac{\partial \ln P(\mathbf{w})}{\partial \mathbf{w}}$  自体は解析的に求まるので、**勾配降下法**で最適化



## ■ ロジスティック回帰

- ・ 回帰によって、入力データがあるカテゴリに当てはまる確率を予測する手法
- ・ 確率値に対して閾値を定めることで、クラスの値を決定する
- ・ 活用例) 試験の偏差の情報を元に、IT エンジニアとなるかならないかを予測

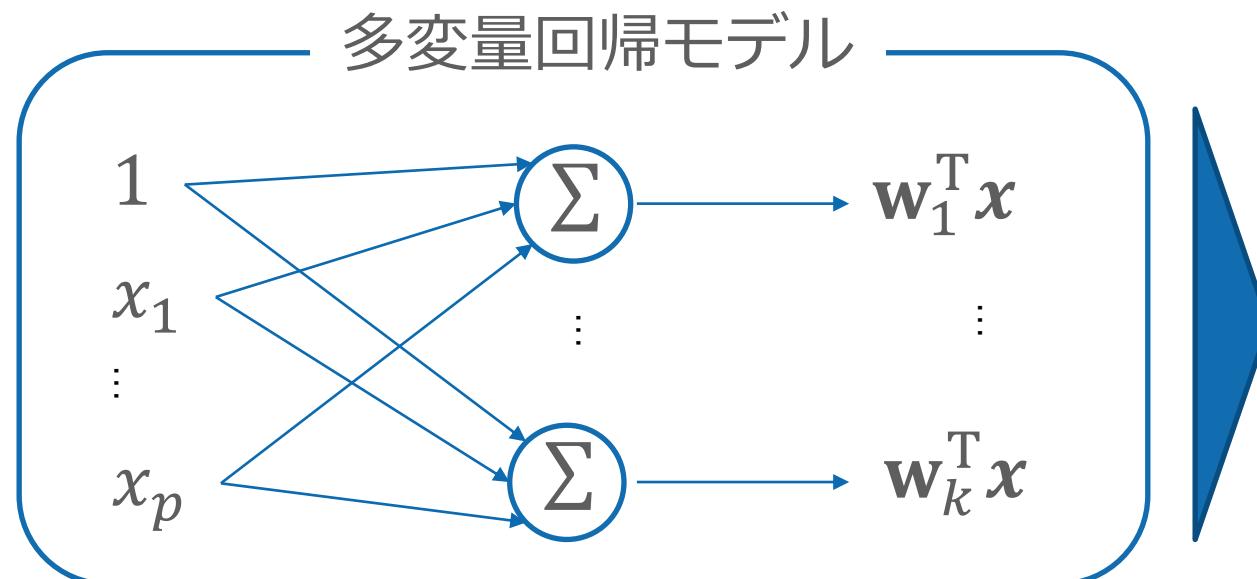


## 多変量モデルへの拡張

---

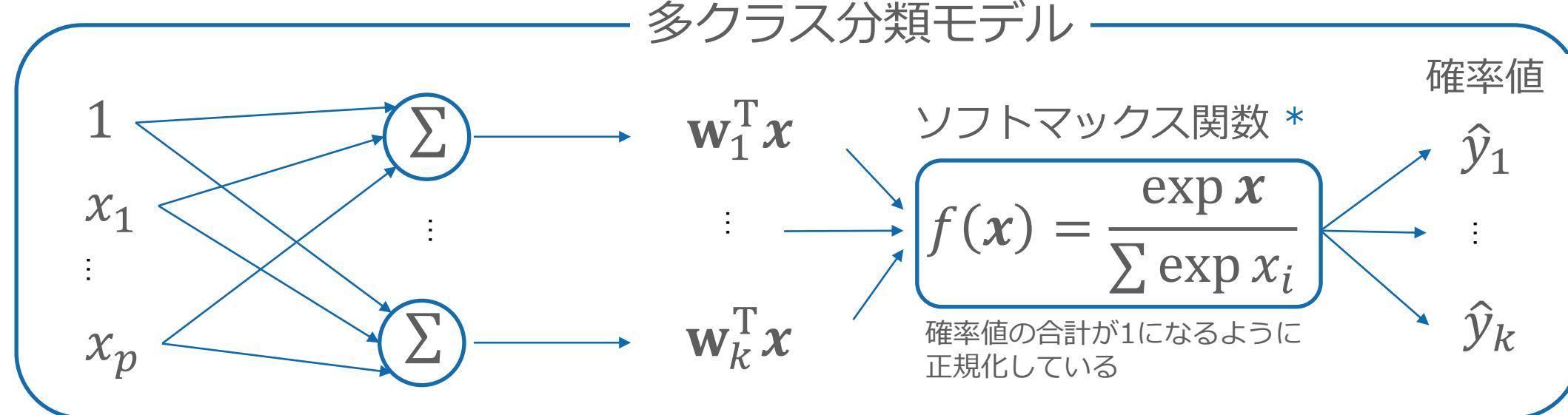
- IT エンジニア、医者、弁護士、デザイナーを対象として、学生時代の理科および数学の偏差を調査した
- このデータから、ある学生が将来これら 4 つの職業になる確率をそれぞれ求めたい
- どのようなモデルをつくればいいか？

- 今まででは目的変数が 1 次元のケースを取り扱ってきた
- しかし、実際には**目的変数が多次元**であることもありうる
- 線形回帰において出力が多次元のとき→多変量回帰



目的変数の各次元ごとに  
線形回帰モデルを置く  
評価基準もそれぞれの次元に  
対する二乗誤差の和で OK

- 分類問題のときは、各出力は「各カテゴリである確率」をそれぞれ意味する
  - ・ 1つ目はイヌである確率、2つ目はネコである確率、3つ目はウマである確率…
- このような多クラス分類モデルのときは、シグモイド関数の代わりに **ソフトマックス関数**を用いて出力を確率値に変換する



\* ソフトマックス関数には、導関数が非常にシンプルになるという利点がある

## ■ 複数の入力を正規化し、合計値が 1 になるようにする関数

- ・ 深層学習でも一般的に用いられる
- ・ 正規化する前に、入力を指数関数 ( $\exp$ ) に通している
- ・ 多クラス分類モデルの出力を、確率（に相当する値）に変換するために用いる
- ・ 入力・出力は、両方ともベクトル

### ソフトマックス関数

$$f(x) = \frac{\exp(x)}{\sum_{i=1}^k \exp(x_i)}$$

$k$  : 出力クラスの数

- 多クラス分類モデルにおける評価指標には**交差エントロピー**を用いることが多い
  - ・ 深層学習でも一般的に用いられる
- (交差エントロピー) = (対数尤度のマイナス 1 倍)
  - ・ 当てはまりの悪さを表す指標として解釈できる
  - ・ 評価基準として交差エントロピーを設定した場合には、**交差エントロピー最小化**という最適化処理を行う

## 交差エントロピー

$$-\sum_{n=1}^N \sum_{i=1}^k y_i^{(n)} \ln \hat{y}_i^{(n)}$$

全クラス分を合計する  
正解クラス以外は 0 になる

$y$  : 正解

$\hat{y}$  : 予測値

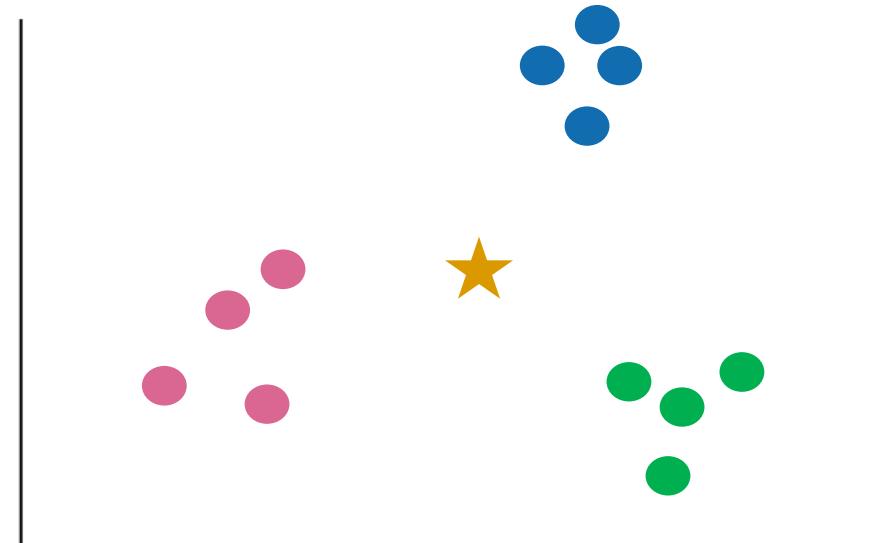
$n$  : データ数

$k$  : 出力クラスの数

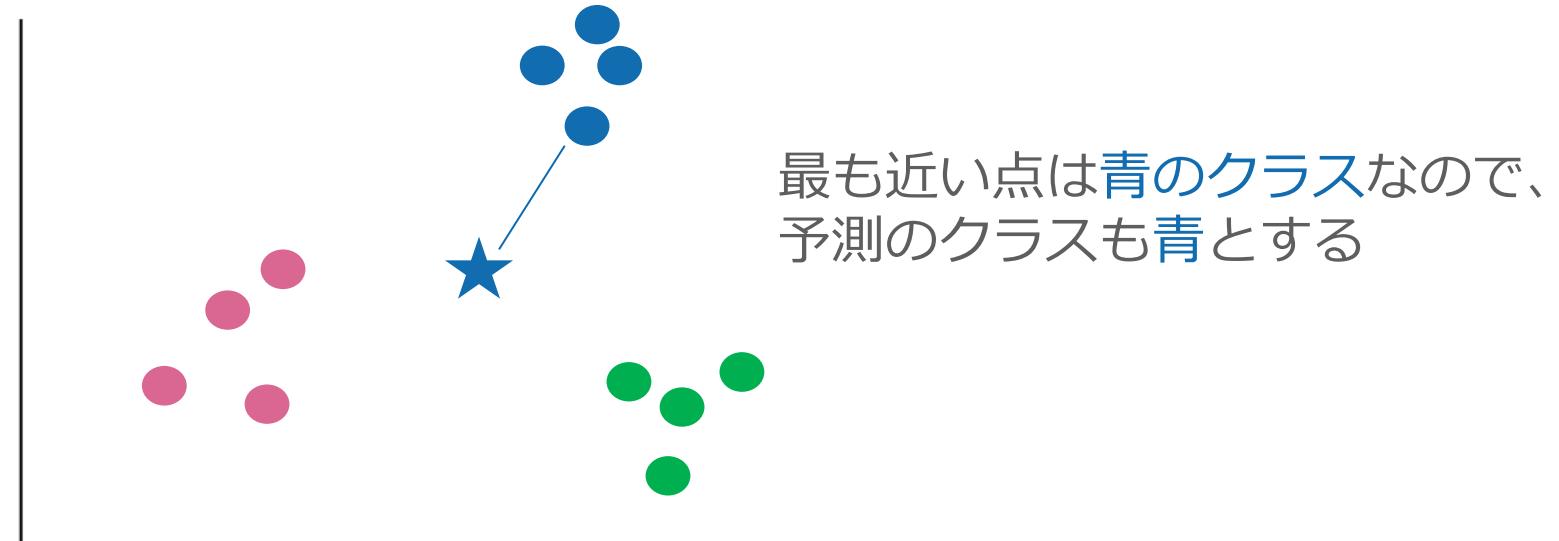
# k近傍法

---

- 既にクラスがわかっているデータが以下のように分布している
- 新しいデータ★に対して、どのクラスであるかを予測したい
- できるだけ簡単な方法で予測したいが、どのような方法が良いだろうか？

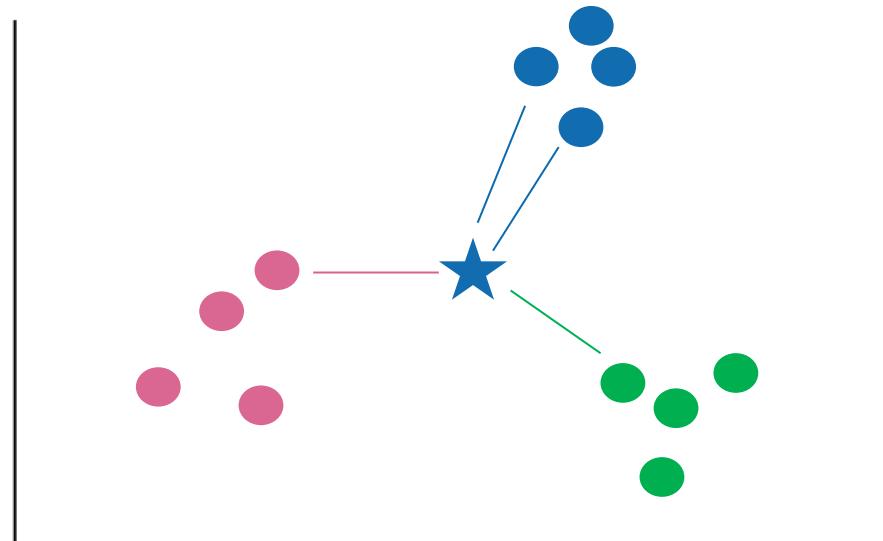


- 最もシンプルな方法は、★に最も近い点と同じクラスに割り当てる方法
- この方法は**最近傍法**と呼ばれる
- 結果をもう少し頑健にするために、見るデータの個数を増やしてみよう



# k近傍法 (k-Nearest Neighbor algorithm, k-NN)

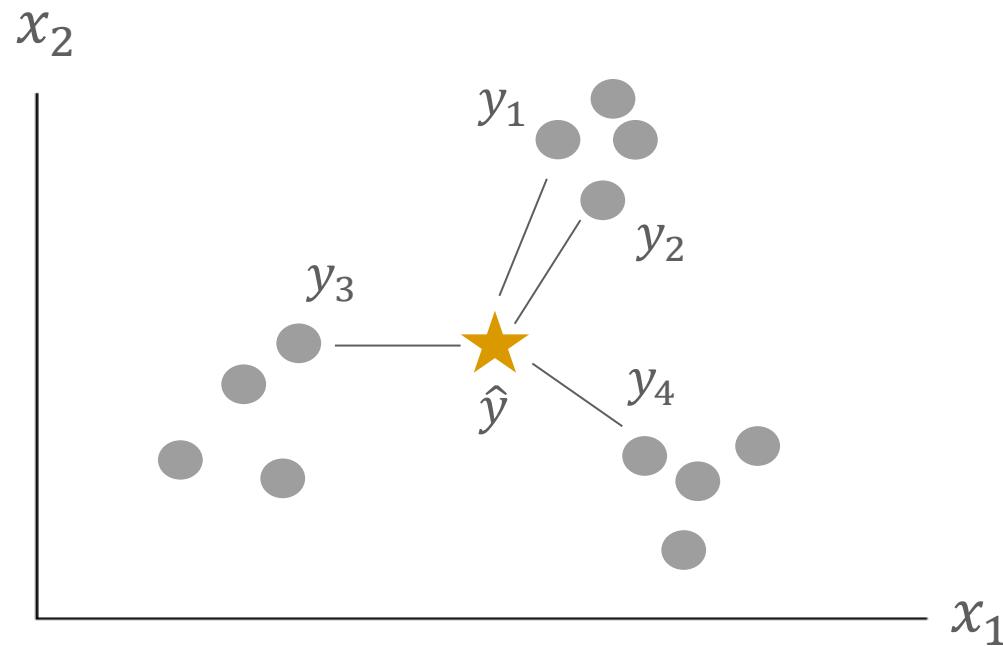
- 新たなデータ点から、近い順に  $k$  個のデータを考える
- この  $k$  個のデータのうち、最も多数派のクラスに割り当てるのが **k近傍法**
  - 全てのデータに対する距離計算が必要なため  
データ数が増えていくと、予測の出力に時間がかかるのが難点



$k=4$  のとき、★から近い順に4つのデータを見る  
このとき多数派は青のクラス  
よって★のクラスは青とする

- 近傍  $k$  個のデータを発見し、目的変数の値の平均値を予測値とする！

$$\hat{y} = \frac{1}{k} \sum_{i=1}^k y_i$$



$$\hat{y} = \frac{1}{4} \sum_{i=1}^4 y_i$$

- k の数は最も性能が高くなるように定める！
  - ・ 機械学習の目的は、未知のデータに対して性能の良いモデルを構築すること
- 様々な k の値を試して、性能が高くなる k を発見する
- 下記の 2 点は後の章で扱う
  - ・ k の値を決めるときのデータセットの作り方（モデルの検証、第 4 章）
  - ・ 候補となる k の値の決め方（ハイパーパラメータ探索、第 5 章）

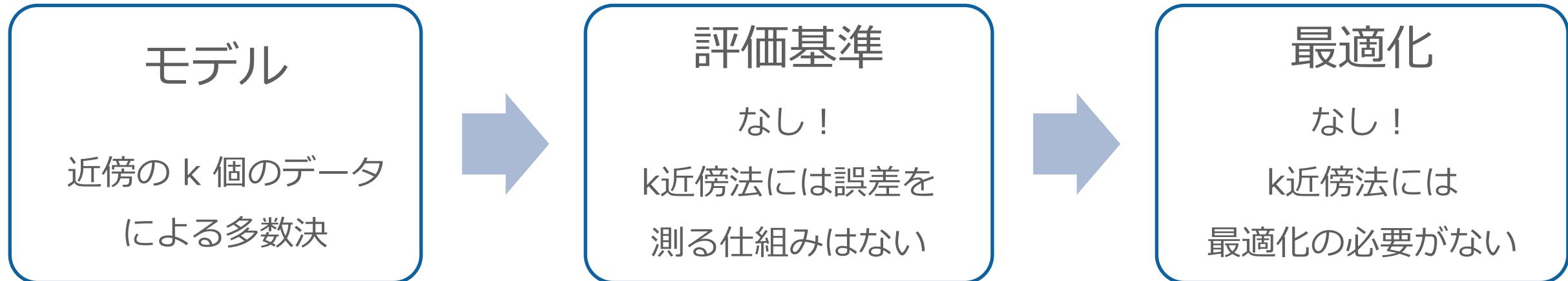
## ■ k近傍法には評価基準はない！

- ・ 新たなデータが得られたら、そのデータの近傍データを  $k$  個を発見
- ・ 近傍データの情報をを利用して、分類や回帰を実施
- ・ 「誤差」を測る仕組みはない！

## ■ 評価基準がないため、学習（最適化）も必要ない！

- ・ 最適化とは、評価基準を最小化/最大化するようなパラメータを求める処理

- あるデータの近傍データを  $k$  個 発見し、近傍データの情報をもとに回帰や分類を実施する手法
- データの数が増えると距離の計算に時間がかかるという欠点がある



## ノートブック演習

---

## ■ 線形回帰

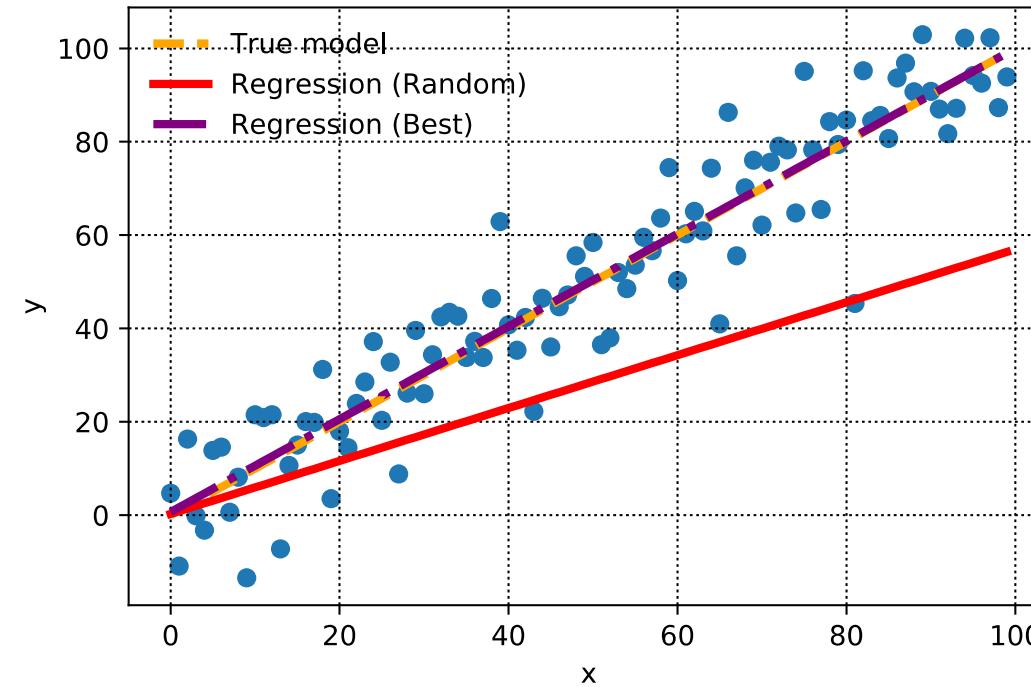
- 2-1\_linear\_regression\_psedo\_data.ipynb
- 2-2\_linear\_regression\_real\_data\_trainee.ipynb (穴埋め問題あり)
- 2-3\_linear\_regression\_multi\_psedo\_data.ipynb
- 2-4\_linear\_regression\_multi\_real\_data\_trainee.ipynb (穴埋め問題あり)

## ■ ロジスティック回帰

- 2-5\_logistic\_regression\_psedo\_data.ipynb
- 2-6\_logistic\_regression\_real\_data\_trainee.ipynb (穴埋め問題あり)

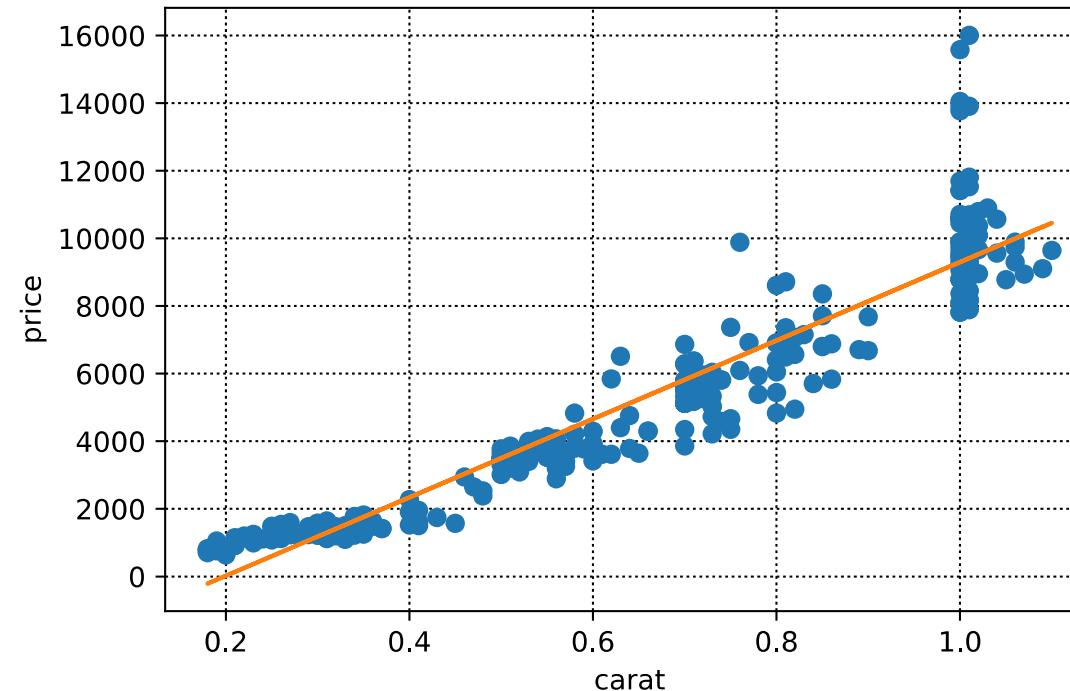
## 2-1\_linear\_regression\_psedo\_data.ipynb

- まずは簡単なデータで線形回帰を試してみよう
- ランダムに直線のパラメータを決めたものよりも、最小二乗法によって決めたパラメータの方が二乗誤差が小さくなることを確認しよう



## 2-2\_linear\_regression\_real\_data\_trainee.ipynb

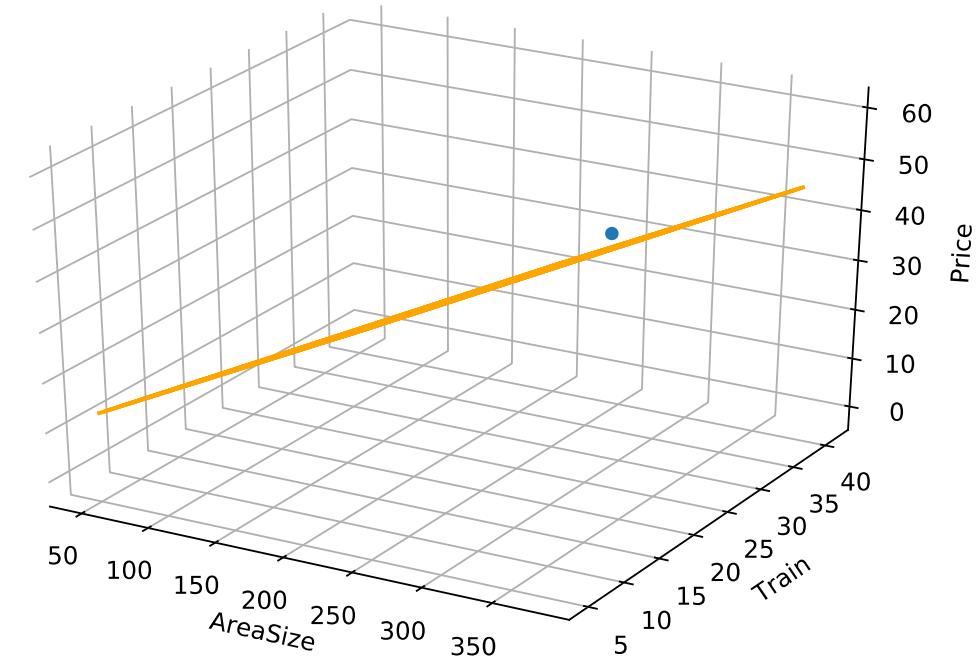
- 実際のデータで線形回帰を実行してみよう
- ダイヤモンドのカラット数からその価格を予測するモデルを作ろう



## 2-3\_linear\_regression\_multi\_psedo\_data.ipynb

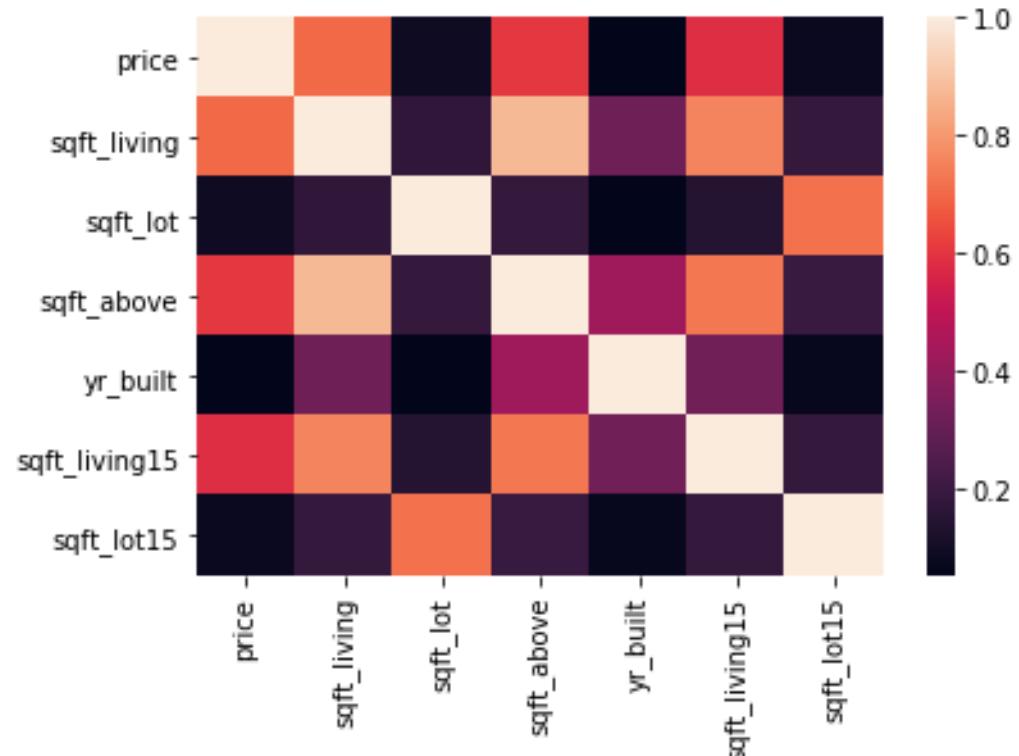
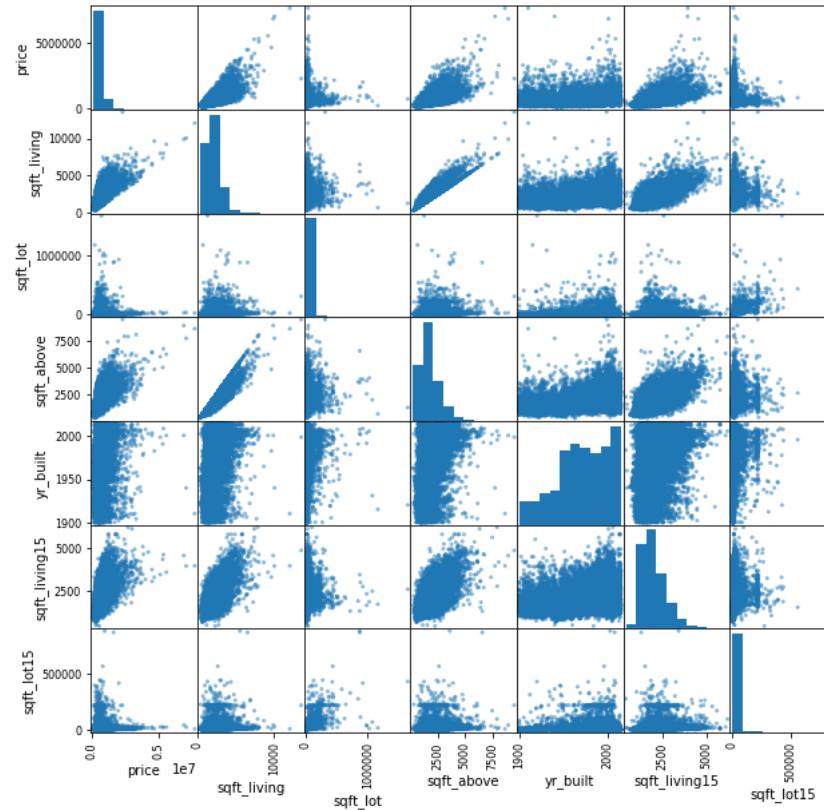
- 説明変数が多い住宅価格データセットで線形回帰をしてみよう
- さらに説明変数ごとに基本的な統計量を確認してみよう

	AreaSize	HouseSize	PassedYear	Price	Train	Walk
count	23.000000	23.000000	23.000000	23.000000	23.000000	23.000000
mean	144.139130	81.356522	9.834783	19.395652	23.260870	6.217391
std	70.086095	26.436955	4.023071	11.605151	10.247915	5.062846
min	50.000000	48.700000	3.100000	5.500000	5.000000	1.000000
25%	99.000000	66.300000	5.700000	11.600000	16.000000	2.500000
50%	139.600000	77.900000	10.500000	17.600000	19.000000	5.000000
75%	173.300000	86.750000	13.150000	25.900000	34.500000	7.500000
max	379.800000	163.700000	14.700000	59.500000	41.000000	20.000000



## 2-4\_linear\_regression\_multi\_real\_data\_trainee.ipynb

- データ数が大きな住宅価格データセットで線形回帰をしてみよう
- 統計量や散布図行列、相関係数なども確認してみよう

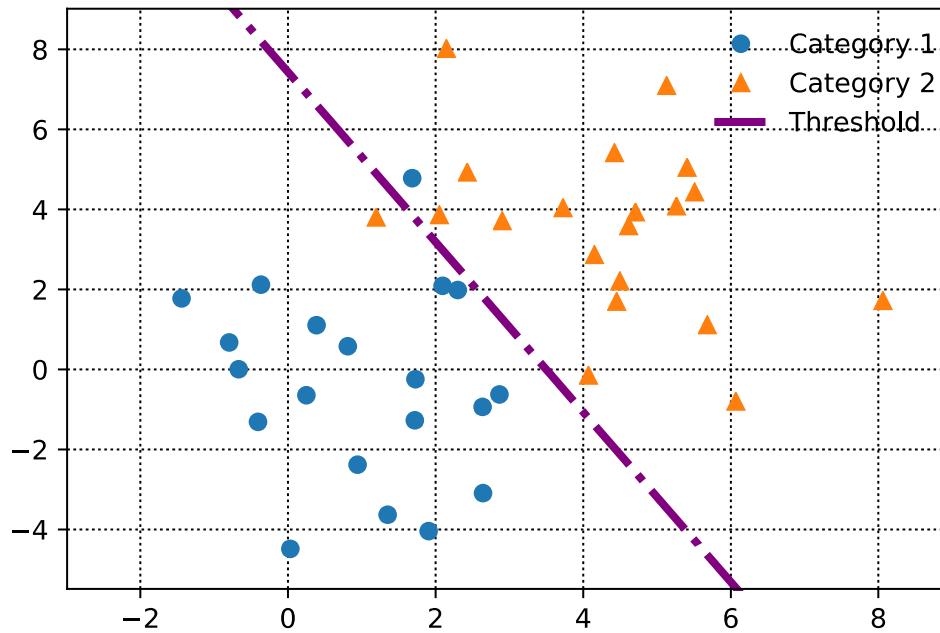


住宅価格データセットの詳細は[こちら](#)をご覧ください

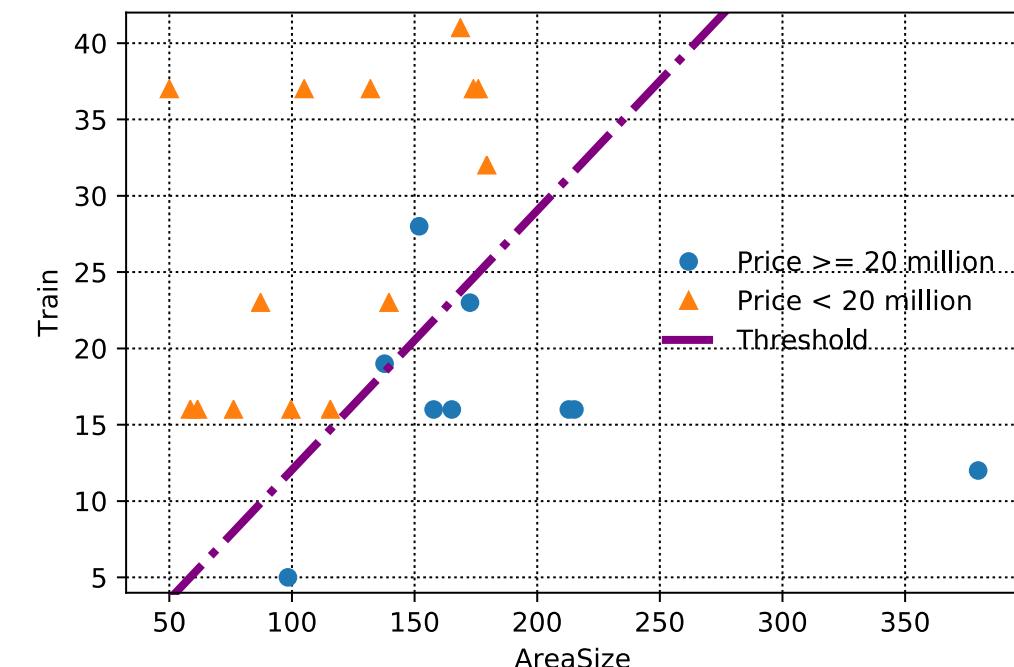
## 2-5\_logistic\_regression\_psedo\_data.ipynb

- まずは簡単なデータでロジスティック回帰を試してみよう
- IT エンジニア予測問題と住宅価格のカテゴリ予測問題でそれぞれモデルを作って確認してみよう

IT エンジニア予測



住宅価格のカテゴリ予測



## 2-6\_logistic\_regression\_real\_data\_trainee.ipynb

- 説明変数が多いデータセットでロジスティック回帰を実装してみよう
- 住宅がリノベーションされているかどうかを予測するモデルを作ってみよう

	yr_renovated	sqft_living	sqft_lot	sqft_above	yr_built	sqft_living15	sqft_lot15
0	False	1180	5650	1180	1955	1340	5650
1	True	2570	7242	2170	1951	1690	7639
2	False	770	10000	770	1933	2720	8062
3	False	1960	5000	1050	1965	1360	5000
4	False	1680	8080	1680	1987	1800	7503

予測 = リノベーションなし 予測 = リノベーション済み

正解 = リノベーションなし	20624	75
正解 = リノベーション済み	882	32

## 本章のまとめ

---

- 線形回帰モデル
- ロジスティック回帰モデル
- 多変量モデルへの拡張
- k近傍法
- ノートブック演習

## ■ 線形回帰モデル

- ・ 回帰問題に用いる
- ・ 評価基準…二乗誤差
- ・ 最適化…最小二乗法

## ■ ロジスティック回帰モデル

- ・ 分類問題に用いる
- ・ 評価基準…対数尤度
- ・ 最適化…勾配降下法

## ■ 多変量モデルへの拡張

- ・ 回帰の場合…出力の次元数と同じ数だけモデルを用意する
- ・ 分類の場合…ソフトマックス関数の計算結果を出力とする

## ■ k近傍法

- ・ 回帰の場合… k 個の近傍データの目的変数の平均値を予測値とする
- ・ 分類の場合… k 個の近傍データでクラスラベルの多数決を行い、多数派を予測クラスとする
- ・ 学習（最適化）は不要

# 現場で使える 機械学習・データ分析基礎講座

第3章：モデルの評価指標

- 回帰問題の評価指標
- 分類問題の評価指標
- 多クラス分類問題の評価指標
- ノートブック演習

- 回帰問題の評価指標のうち MAE・RMSE・MSE を説明できる
- 分類問題の評価指標のうち Accuracy・Recall・Precision・AUC を説明できる

## 回帰問題の評価指標

---

## ■ MAE (Mean Absolute Error)

- 予測と実際の差の絶対値の平均
- 平均絶対値誤差とも呼ばれる

## ■ MSE (Mean Squared Error)

- 予測と実際の差の二乗の平均
- 平均二乗誤差とも呼ばれる

## ■ RMSE (Root Mean Squared Error)

- MSE の平方根
- 二乗平均平方根誤差とも呼ばれる

MAEとRMSEは解釈性がよい

- 誤差と目的変数の単位が一致するため
- 右図ならば MAE は「赤線の平均」

$$MAE = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

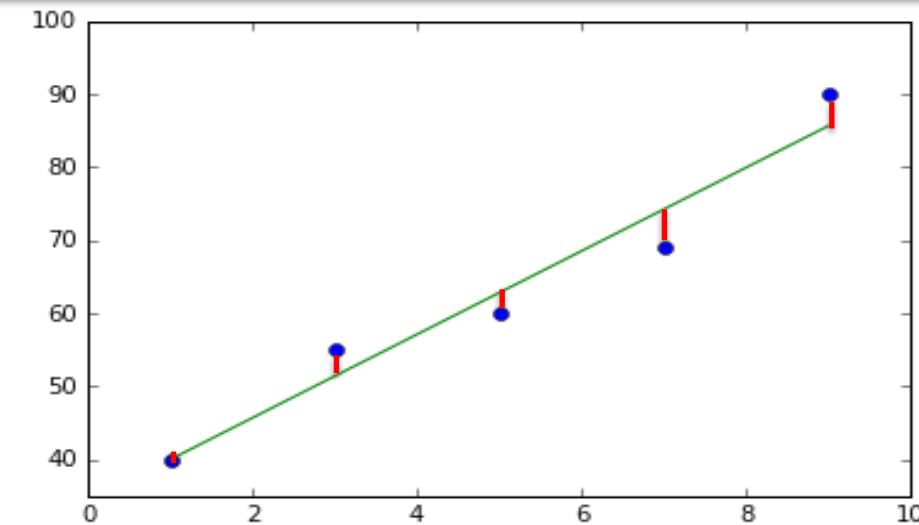
$$MSE = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

$y_i$  :  $i$  番目のデータに対する正解値

$\hat{y}_i$  :  $i$  番目のデータに対する予測値

$\bar{y}$  : 正解の平均



## ■ MAPE (Mean Absolute Percentage Error)

- 正解値に対する予測値の相対誤差
- 平均絶対パーセント誤差とも呼ばれる
- 変数のスケール（値域）が極端に大きい場合に有用
  - 例) 住宅価格

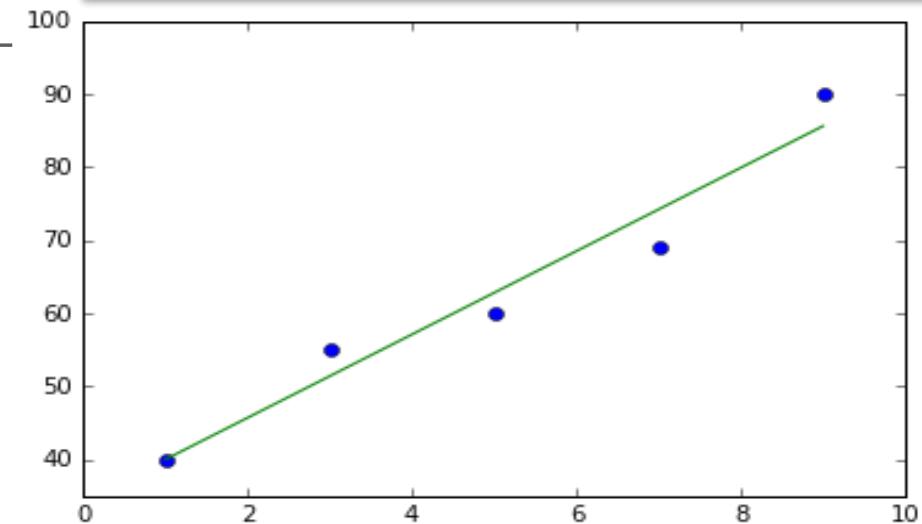
## ■ 決定係数 $R^2$

- 回帰モデルのデータに対する当てはまり具合を表す
- 値が大きいほど、説明変数を用いて目的変数を正確に説明できている

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{|y_i|}$$

$y_i$  :  $i$  番目のデータに対する正解値  
 $\hat{y}_i$  :  $i$  番目のデータに対する予測値  
 $\bar{y}$  : 正解の平均



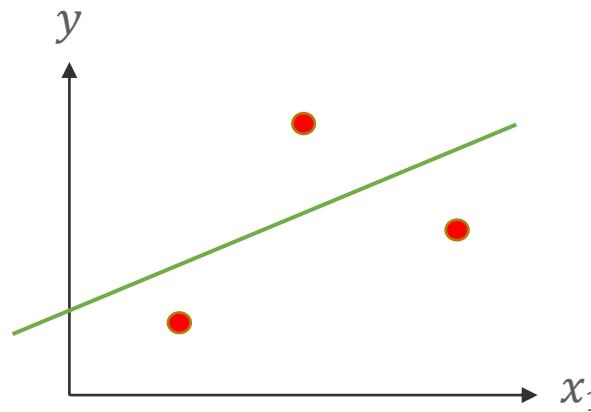
$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}{\frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{\text{二乗誤差}}{\text{目的変数の分散}}$$

- 0 から 1 までの値をとる
  - 1 に近いほど、当てはまりが高いと解釈できる
  - 二乗誤差がゼロのとき、 $R^2 = 1$
- 説明変数の数が多いほど、値が大きくなりやすい

## ■ データの数が 3 つの場合に、回帰モデルを作るとする

説明変数が 1 つの場合

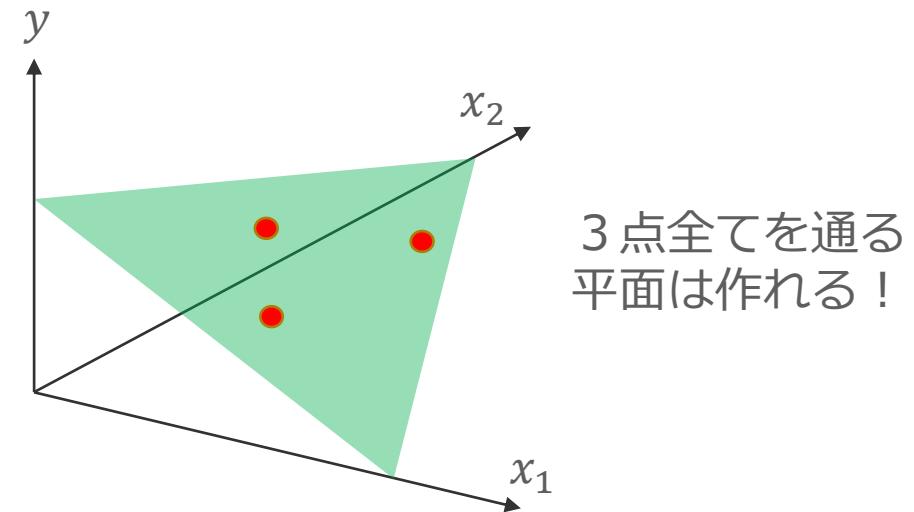
二乗誤差を 0 にする  
回帰モデルは作れない



3点全てを通る  
直線は作れない

説明変数が 2 つの場合

二乗誤差を 0 にする  
回帰モデルを作れる！



3点全てを通る  
平面は作れる！

- 決定係数は、説明変数の個数から生じる当てはめの「簡単さ」が考慮されない
- 説明変数の数が 2 個以上の重回帰の場合は、説明変数の数を考慮した「自由度調整済み決定係数」を見ると良い

(参考) 自由度調整済み決定係数  $R^{*2}$

$$R^{*2} = 1 - \frac{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n - 1 - p}}{\frac{\sum_{i=1}^n (y_i - \bar{y})^2}{n - 1}}$$

*n* : データの数  
*p* : 説明変数の数

説明変数の数が多いとき、値が小さくなるように補正されている

# 【ワーク】回帰モデルの評価指標を計算してみよう

No.	予測値 $\hat{y}$ [g]	正解値 $y_i$ [g]
1	10	15
2	8	10
3	10	5

右表の結果 ( $n = 3$ ) に対して、  
MAE、MAPE、MSE、RMSE を算出しよう

# 【ワーク】回帰モデルの評価指標を計算してみよう | 解答

No.	予測値 $\hat{y}$ [g]	正解値 $y_i$ [g]
1	10	15
2	8	10
3	10	5

右表の結果 ( $n = 3$ ) に対して、  
MAE、MAPE、MSE、RMSE を算出しよう

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| = \frac{1}{3} \{ |15 - 10| + |10 - 8| + |5 - 10| \} = \frac{5 + 2 + 5}{3} = 4 \text{ [g]}$$

平均 4 [g] の誤差があるということ  
実務では、ビジネス要件を満たすかどうかを考える上で  
着目したい指標

$$\text{MAPE} = \frac{100}{n} \sum_{i=1}^n \frac{|y_i - \hat{y}_i|}{|y_i|} = \frac{100}{3} \left\{ \frac{|15 - 10|}{|15|} + \frac{|10 - 8|}{|10|} + \frac{|5 - 10|}{|5|} \right\} = \frac{100}{3} \left( \frac{1}{3} + \frac{1}{5} + 1 \right) = \frac{100}{3} \frac{23}{15} \approx 51.1 \text{ [%]}$$

平均 51.1 [%] の誤差がある  
ということ

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 = \frac{1}{3} \{ (15 - 10)^2 + (10 - 8)^2 + (5 - 10)^2 \} = \frac{25 + 4 + 25}{3} = 18 \text{ [g}^2\text{]}$$

単位が [g<sup>2</sup>] で、解釈ができない  
他の指標を見た方がよい

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2} = \sqrt{18} = 3\sqrt{2} \text{ [g]}$$

MSE の二乗の影響が残るため、  
大きく予測を外したデータがある場合、MAE よりも大きな値になる

## 分類問題の評価指標

---

- 予測結果と真の結果でクロス集計をしたものを**混同行列**という
- Accuracy (正解率)

- 計算式:  $(TP + TN) / (TP + TN + FN + FP)$

- Recall

- 実際に正であるもの内、正であると予測した割合
- 計算式:  $TP / (TP + FN)$

- Precision

- 正と予測したもの内、正であったものの割合

計算式:  $TP / (TP + FP)$

Recall と Precision は  
トレードオフの関係

		真の結果	
		正 (Positive)	負 (Negative)
予測結果	正 (Positive)	TP 個	FP 個
	負 (Negative)	FN 個	TN 個

## ■ F1 (f1-score)

- Recall と Precision のバランスをとった評価指標
- 計算式 :  $(2 \times \text{Recall} \times \text{Precision}) / (\text{Recall} + \text{Precision})$

## ■ モデルの利用目的によって、どの評価指標を用いるかを検討

- 広告の郵送（無駄な費用をかけたくない）なら Precision 優先
- 故障の検知（見逃したくない）なら Recall 優先

		真の結果	
		正 (Positive)	負 (Negative)
予測結果	正 (Positive)	TP 個	FP 個
	負 (Negative)	FN 個	TN 個

## ■ 下記データで、2 クラス分類モデルを開発したとする

- 全データ数：1000 件
  - 正のクラス：950 件
  - 負のクラス：50 件
- } 不均衡 (invalance) なデータ  
クラス間でサンプル数に偏りがある

分類結果

		真の結果	
		正	負
予測結果	正	940	50
	負	10	0

$$\text{Accuracy} = \frac{940}{1000} = 94 [\%]$$

$$\text{Precision} = \frac{940}{990} \approx 95 [\%]$$

Accuracy と Precision は高いといえる  
しかし、この分類モデルは負のクラスのデータを全く予測できていない！

(評価指標の値だけでなく、混同行列も確認することが重要)

# 【ワーク】分類モデルの評価指標を計算してみよう

下記の混同行列を用いて、Accuracy、Precision、Recall を計算してみよう

		真の結果	
		正 (Positive)	負 (Negative)
予測結果	正 (Positive)	70	100
	負 (Negative)	180	150

# 【ワーク】分類モデルの評価指標を計算してみよう | 解答

下記の混同行列を用いて、Accuracy、Precision、Recall を計算してみよう

		真の結果	
		正 (Positive)	負 (Negative)
予測結果	正 (Positive)	70	100
	負 (Negative)	180	150

$$\text{Accuracy} = \frac{70 + 150}{70 + 100 + 180 + 150} = \frac{220}{500} = 0.44 \quad \text{「正」と「負」予測的中率は } 44\%$$

$$\text{Precision} = \frac{70}{70 + 100} = \frac{70}{170} = 0.41176\dots \quad \text{「正」と予測したとき約 } 41\% \text{ が的中する}$$

$$\text{Recall} = \frac{70}{70 + 180} = \frac{70}{250} = 0.28 \quad \text{「正」のものを } 28\% \text{ しか正しく予測できない}$$

- 分類モデルの予測値は、データが「あるクラスに該当する確率」
- 閾値次第で評価指標の値は全く異なる
- 閾値に依存しない評価方法は？→ ROC 曲線と AUC

閾値（しきい値）

確率がいくつ以上であれば  
「正例」とするか  
決定するための基準値

1	100
2	200
3	300

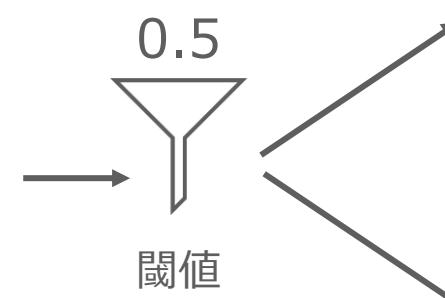
データ



予測

予測値 (0~1)		
1	100	0.81
2	200	0.42
3	300	0.35

例) 異常検知タスクの場合



$\geq 0.5$  (正例)

1	100	0.81
---	-----	------

異常あり  
(1と判断)

$< 0.5$  (負例)

2	200	0.42
3	300	0.35

異常なし  
(0と判断)

予測ラベル

## ■ ROC 曲線 (Receiver Operating Characteristic curve)

- ・ 真陽性率と偽陽性率の関係をグラフにしたもの

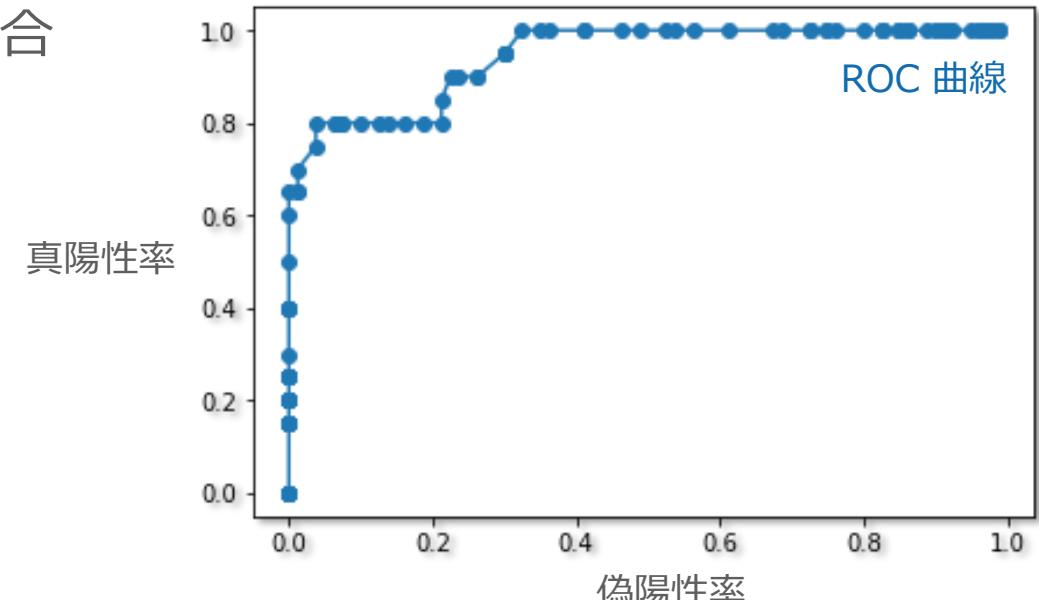
## ■ True positive rate … 真陽性率 ( $TP / (TP + FN)$ )

- ・ 実際に正であるもののうち、正と予測された割合
- ・ Recall と同じ定義

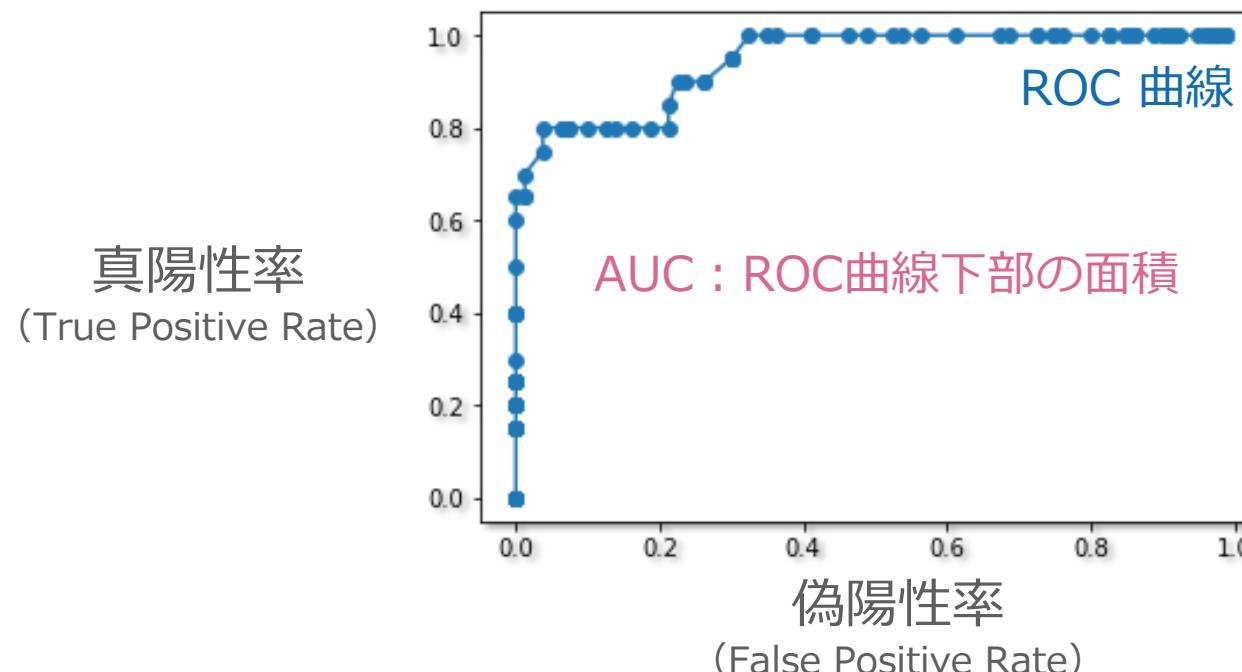
## ■ False positive rate … 偽陽性率 ( $FP / (FP + TN)$ )

- ・ 実際に負であるもののうち、正と予測された割合

		真の結果	
		正 (Positive)	負 (Negative)
予測結果	正 (Positive)	TP 個	FP 個
	負 (Negative)	FN 個	TN 個

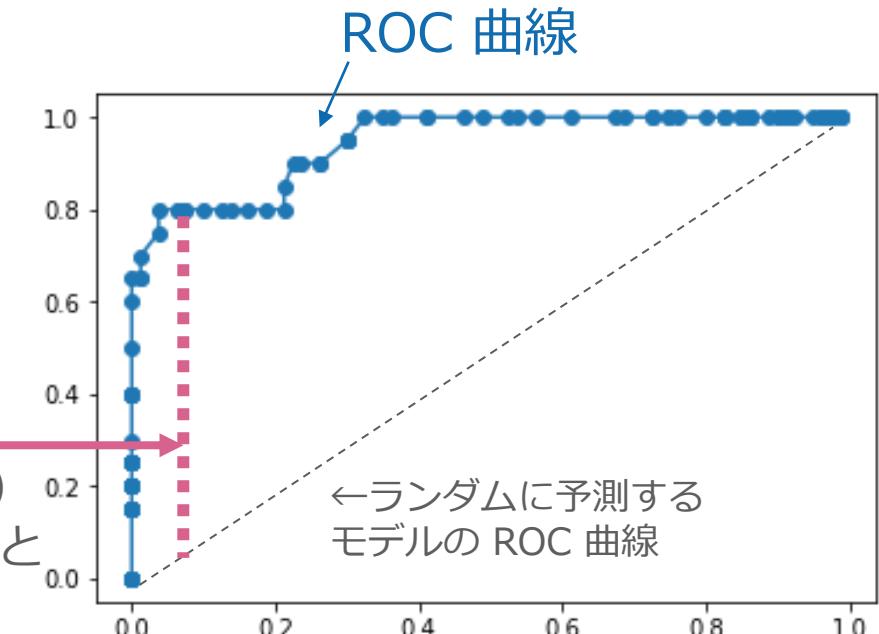


- 真陽性率と偽陽性率を様々な閾値で計算しプロット
- AUC (Area Under Curve) : ROC 曲線下部の面積
  - 値が大きいほどモデルの性能が高いと解釈
    - 最大値は 1 であり、この時最も性能が高い
    - 最小値は理論上 0.5 であり、この時最も性能が低い



- 閾値に依存しない評価ができたとはいえ、現場に導入する際には閾値を定めないとならない
- 閾値の定め方としては、以下の方法がある
  - ① 要件に基づいて決める
    - Recall や Precision、偽陽性率など、何かしらの評価指標が満たすべき要件をもとに閾値を手探りで決める
  - ② ヨーデンインデックス (Youden's Index) 法
    - ROC 曲線において、 $AUC = 0.5$  となる最悪ケースの直線から、最も離れた位置にある点に対応した閾値を採用する

この点線棒の長さ (=ヨーデンインデックス)  
が最も大きい点を見つければよい、ということ



## 多クラス分類の評価指標

---

- 多クラスの不均衡データを扱う場合は、評価指標の計算方法に注意
- 多クラス分類の評価指標
  - Micro 平均：全クラスのデータをまとめて扱い、評価指標を求める
  - Macro 平均：クラスごとに評価指標を計算し、それらの平均を求める

- 全クラスのデータをまとめて扱い、評価指標を求める
  - ・ サンプル数の大きいクラスの評価が支配的となる

		実際		
		イヌ	ネコ	ウマ
予測	イヌ	TP <sub>1</sub>	FP <sub>1</sub> , FN <sub>2</sub>	FP <sub>1</sub> , FN <sub>3</sub>
	ネコ	FP <sub>2</sub> , FN <sub>1</sub>	TP <sub>2</sub>	FP <sub>2</sub> , FN <sub>3</sub>
	ウマ	FP <sub>3</sub> , FN <sub>1</sub>	FP <sub>3</sub> , FN <sub>2</sub>	TP <sub>3</sub>

例) micro-Accuracy

$$A_{micro} = \frac{\sum_i TP_i}{\sum_i (TP_i + FP_i + FN_i + TN_i)}$$

※同じデータは 1 回しかカウントしない

※多クラス混同行列では、TN は表示できない

TP : 真陽性 (True Positive)  
 FP : 偽陽性 (False Positive)  
 FN : 偽陰性 (False Negative)  
 TN : 真陰性 (True Negative)

## ■ Micro 平均の場合、Precision は Accuracy と同じ値になる

- Recall や F1 も同様

		実際		
		イヌ	ネコ	ウマ
予測	イヌ	TP <sub>1</sub>	FP <sub>1</sub>	FP <sub>1</sub>
	ネコ	FP <sub>2</sub>	TP <sub>2</sub>	FP <sub>2</sub>
	ウマ	FP <sub>3</sub>	FP <sub>3</sub>	TP <sub>3</sub>

例) micro-Precision

$$P_{micro} = \frac{\sum_i TP_i}{\sum_i (TP_i + FP_i)}$$

$$A_{micro} = \frac{\sum_i TP_i}{\sum_i (TP_i + FP_i + FN_i + TN_i)}$$

※同じデータは 1 回しかカウントしない

※多クラス混同行列では、TN は表示できない

TP : 真陽性 (True Positive)  
 FP : 偽陽性 (False Positive)  
 FN : 偽陰性 (False Negative)  
 TN : 真陰性 (True Negative)

## ■ Micro 平均の場合、Precision は Accuracy と同じ値になる

- Recall や F1 も同様

		実際		
		イヌ	ネコ	ウマ
予測	イヌ	TP <sub>1</sub>	FN <sub>2</sub>	FN <sub>3</sub>
	ネコ	FN <sub>1</sub>	TP <sub>2</sub>	FN <sub>3</sub>
	ウマ	FN <sub>1</sub>	FN <sub>2</sub>	TP <sub>3</sub>

例) micro-Recall

$$R_{micro} = \frac{\sum_i TP_i}{\sum_i (TP_i + FN_i)}$$

$$A_{micro} = \frac{\sum_i TP_i}{\sum_i (TP_i + FP_i + FN_i + TN_i)}$$

※同じデータは 1 回しかカウントしない

※多クラス混同行列では、TN は表示できない

TP : 真陽性 (True Positive)  
 FP : 偽陽性 (False Positive)  
 FN : 偽陰性 (False Negative)  
 TN : 真陰性 (True Negative)

# 多クラス分類の評価指標 | Macro 平均

- クラスごとに分けて評価指標を計算し、それらの平均を求める
  - ・ サンプル数の影響を受けない

例) macro-Accuracy

クラス 1	イヌ	その他
イヌ	TP <sub>1</sub>	FP <sub>1</sub>
その他	FN <sub>1</sub>	TN <sub>1</sub>

$$A_1 = \frac{TP_1}{TP_1 + FP_1 + FN_1 + TN_1}$$

クラス 2	ネコ	その他
ネコ	TP <sub>2</sub>	FP <sub>2</sub>
その他	FN <sub>2</sub>	TN <sub>2</sub>

$$A_2 = \frac{TP_2}{TP_2 + FP_2 + FN_2 + TN_2}$$

クラス 3	ウマ	その他
ウマ	TP <sub>3</sub>	FP <sub>3</sub>
その他	FN <sub>3</sub>	TN <sub>3</sub>

$$A_3 = \frac{TP_3}{TP_3 + FP_3 + FN_3 + TN_3}$$

$$A_{macro} = \frac{1}{3} \sum_i A_i$$

TP : 真陽性 (True Positive)  
FP : 偽陽性 (False Positive)  
FN : 偽陰性 (False Negative)  
TN : 真陰性 (True Negative)

## ■ F1 の Macro 平均の計算方法は 2 通り存在する

- ① クラスごとに分けて F1 を計算し、平均する
- ② macro-Recall と macro-Precision を計算し、F1 の算出式に適用する

クラス 1	イヌ	その他
イヌ	TP <sub>1</sub>	FP <sub>1</sub>
その他	FN <sub>1</sub>	TN <sub>1</sub>

$$F_1 = \frac{2P_1R_1}{P_1 + R_1}$$

例) macro-F1 ①

クラス 2	ネコ	その他
ネコ	TP <sub>2</sub>	FP <sub>2</sub>
その他	FN <sub>2</sub>	TN <sub>2</sub>

$$F_2 = \frac{2P_2R_2}{P_2 + R_2}$$

クラス 3	ウマ	その他
ウマ	TP <sub>3</sub>	FP <sub>3</sub>
その他	FN <sub>3</sub>	TN <sub>3</sub>

$$F_3 = \frac{2P_3R_3}{P_3 + R_3}$$

$$F_{macro} = \frac{1}{3} \sum_i F_i$$

TP : 真陽性 (True Positive)  
 FP : 偽陽性 (False Positive)  
 FN : 偽陰性 (False Negative)  
 TN : 真陰性 (True Negative)

## ■ F1 の Macro 平均の計算方法は 2 通り存在する

- ① クラスごとに分けて F1 を計算し、平均する
- ② macro-Recall と macro-Precision を計算し、F1 の算出式に適用する

クラス 1	イヌ	その他
イヌ	TP <sub>1</sub>	FP <sub>1</sub>
その他	FN <sub>1</sub>	TN <sub>1</sub>

クラス 2	ネコ	その他
ネコ	TP <sub>2</sub>	FP <sub>2</sub>
その他	FN <sub>2</sub>	TN <sub>2</sub>

クラス 3	ウマ	その他
ウマ	TP <sub>3</sub>	FP <sub>3</sub>
その他	FN <sub>3</sub>	TN <sub>3</sub>

例) macro-F1 ②

$$R_{macro} = \frac{1}{3} \sum_i R_i$$

$$P_{macro} = \frac{1}{3} \sum_i P_i$$

$$F_{macro} = \frac{2P_{macro}R_{macro}}{P_{macro} + R_{macro}}$$

TP : 真陽性 (True Positive)  
 FP : 偽陽性 (False Positive)  
 FN : 偽陰性 (False Negative)  
 TN : 真陰性 (True Negative)

## ノートブック演習

---

## 3-1\_model\_evaluation.ipynb

- 回帰問題、分類問題それぞれにおける評価指標の実装方法を確認してみよう

```
# 値を予測
y_pred = regr.predict(X)

# MSEを計算
mse = mean_squared_error(y, y_pred)
print("MSE = %s"%round(mse,3) )

# MAEを計算
mae = mean_absolute_error(y, y_pred)
print("MAE = %s"%round(mae,3) )

# RMSEを計算
rmse = np.sqrt(mse)
print("RMSE = %s"%round(rmse, 3) )
```

MSE = 14.885  
 MAE = 3.057  
 RMSE = 3.858

```
# ラベルを予測
y_pred = clf.predict(X)

# 正答率を計算
accuracy = accuracy_score(y, y_pred)
print('正答率 (Accuracy) = {:.3f}%'.format(100 * accuracy))

# Precision, Recall, F1-scoreを計算
precision, recall, f1_score, _ = precision_recall_fscore_support(y, y_pred)

# カテゴリ「2000万以上」に関するPrecision, Recall, F1-scoreを表示
print('適合率 (Precision) = {:.3f}%'.format(100 * precision[0]))
print('再現率 (Recall) = {:.3f}%'.format(100 * recall[0]))
print('F1値 (F1-score) = {:.3f}%'.format(100 * f1_score[0]))
```

正答率 (Accuracy) = 91.304%  
 適合率 (Precision) = 87.500%  
 再現率 (Recall) = 100.000%  
 F1値 (F1-score) = 93.333%

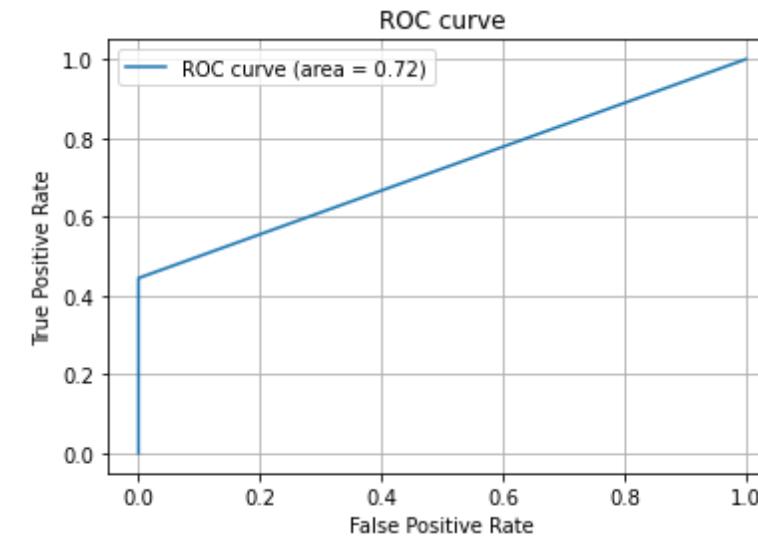
## 3-2\_model\_evaluation\_roc\_curve.ipynb

- ROC 曲線を描き、AUC を算出してみよう

```
# y_pred[:, 1] でクラス1の確率のみを取得
fpr, tpr, thresholds = roc_curve(y, y_pred[:, 1])

# AUC を計算
auc = auc(fpr, tpr)
```

```
# ROC 曲線をプロット
plt.plot(fpr, tpr, label='ROC curve (area = %.2f)' % auc)
plt.title('ROC curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.grid(True)
```



## 3-3\_micro\_macro\_average.ipynb

- 多クラス分類における評価指標の実装方法を確認してみよう

```
scores = classification_report(y, y_pred, target_names=label_txt)
print(scores)
```

	precision	recall	f1-score	support
Setosa	1.00	0.96	0.98	50
Veriscolour	0.92	0.94	0.93	50
Verginica	0.94	0.96	0.95	50
accuracy			0.95	150
macro avg	0.95	0.95	0.95	150
weighted avg	0.95	0.95	0.95	150

## 本章のまとめ

---

- 回帰問題の評価指標
- 分類問題の評価指標
- 多クラス分類問題の評価指標
- ノートブック演習

## ■ 回帰問題の評価指標

- MAE (平均絶対値誤差)
- MSE (平均二乗誤差)
- RMSE (二乗平均平方根誤差)
- MAPE (平均絶対パーセント誤差)
- 決定係数  $R^2$

## ■ 分類問題の評価指標

- 混同行列
- Accuracy
- Recall
- Precision
- F1
- AUC

		真の結果	
		正 (Positive)	負 (Negative)
予測結果	正 (Positive)	TP 個	FP 個
	負 (Negative)	FN 個	TN 個

## ■ 多クラス分類の評価指標

- Micro 平均
  - 全クラスのデータをまとめて扱い、評価指標を求める
- Macro 平均
  - クラスごとに評価指標を計算し、それらの平均を求める

# 現場で使える 機械学習・データ分析基礎講座

第4章：モデルの検証と正則化

- 訓練誤差と汎化誤差
- 汎化誤差の推定
- 過学習と未学習
- 正則化
- ノートブック演習

## ■ 下記の用語を全て説明できるようになる

- 訓練誤差と汎化誤差
- ホールドアウト法
- 交差検証法
- 過学習と未学習
- バイアス・バリアンス
- 正則化

## 訓練誤差と汎化誤差

---

## ■ 訓練誤差

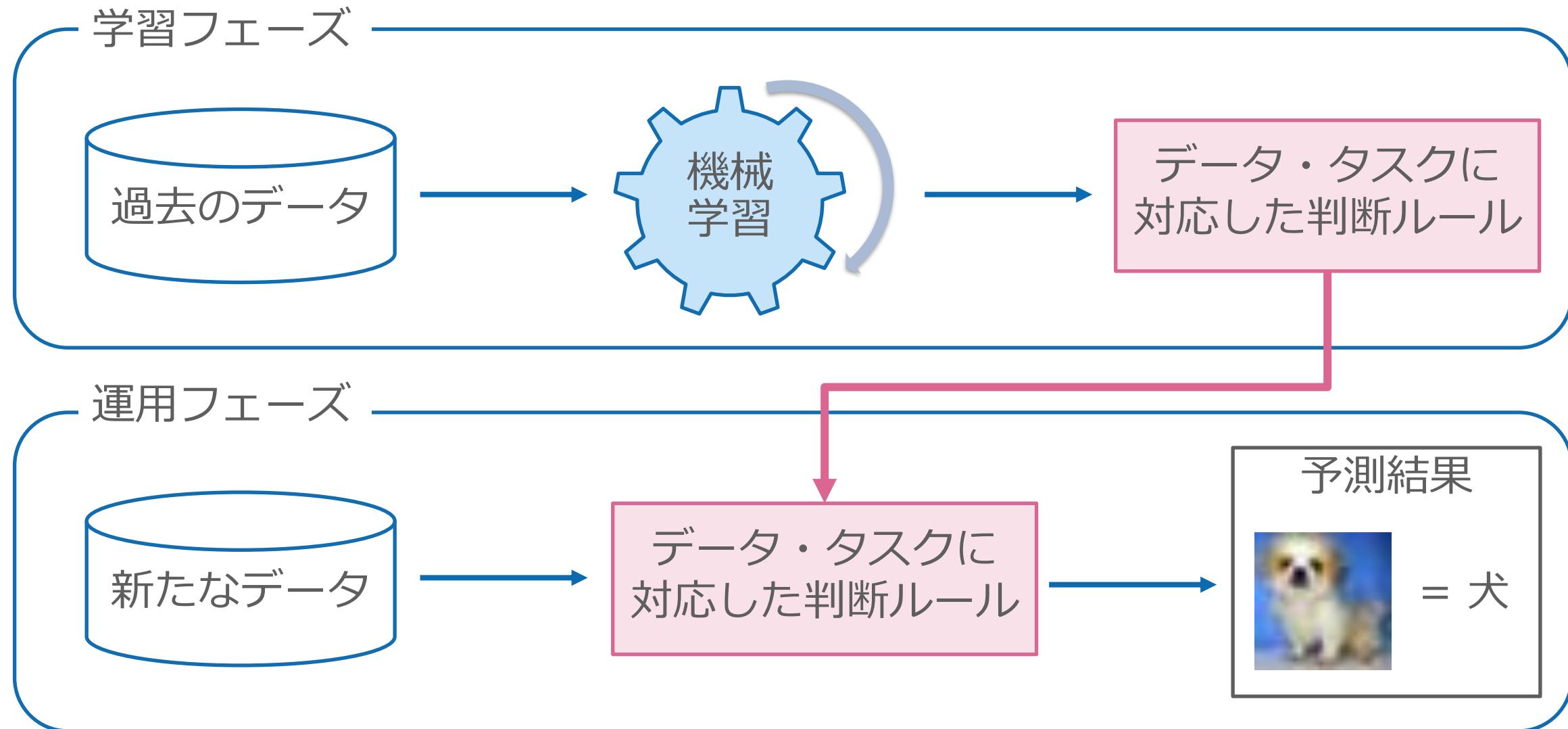
- 「モデルの学習時に利用したデータ」に対するモデルの誤差
- 例) 二乗誤差で測定した、線形モデルにおける  
学習時に利用したデータ  $(\mathbf{x}^{(1)}, y^{(1)}), \dots, (\mathbf{x}^{(N)}, y^{(N)})$  に対する訓練誤差

$$\frac{1}{2} \sum_{n=1}^N (\mathbf{w}^*{}^T \mathbf{x}^{(n)} - y^{(n)})^2$$

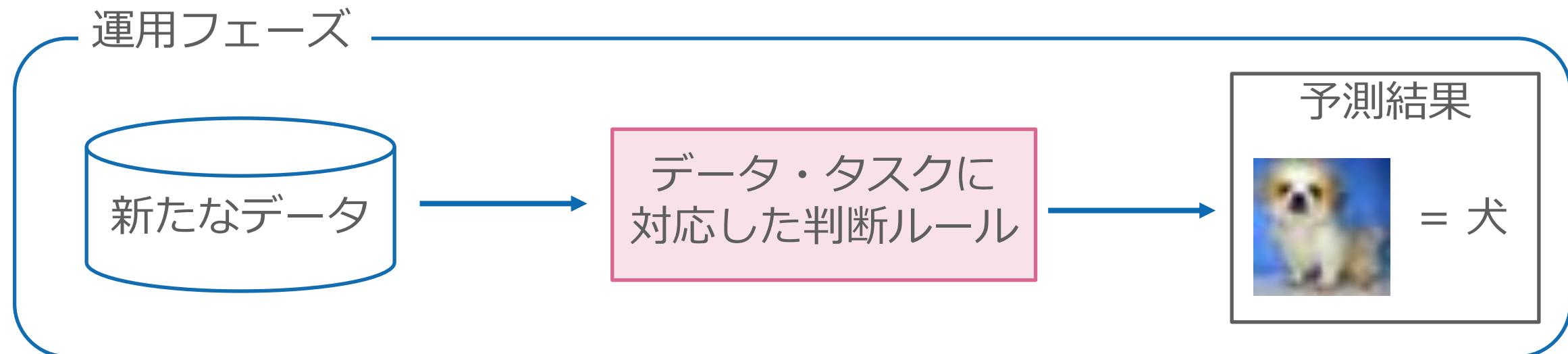
$\mathbf{w}^*$  : 学習によって得られた  
最適なパラメータ

訓練誤差がほとんど 0 のモデルを構築したとする  
このモデルは**実運用**に耐えうる完璧なモデルだろうか？

# 機械学習モデルの構築・運用の流れ



- 運用フェーズではモデルが“**学習していないデータ**”も出現しうる
  - ・ 訓練誤差がほぼ 0 でも、モデルが実際の運用で使えるかどうかは分からぬ！
- モデルが運用フェーズで使えるかどうかは、訓練誤差ではなく**汎化誤差**というもので評価する



## ■ 汎化誤差

- 未知のデータに対するモデルの誤差
- 未知のデータに対する予測性能は汎化性能と呼ばれる

## ■ 機械学習モデルを構築する際には、

汎化誤差が小さくなるように（汎化性能が最大となるように）作り込む  
これが機械学習の目的

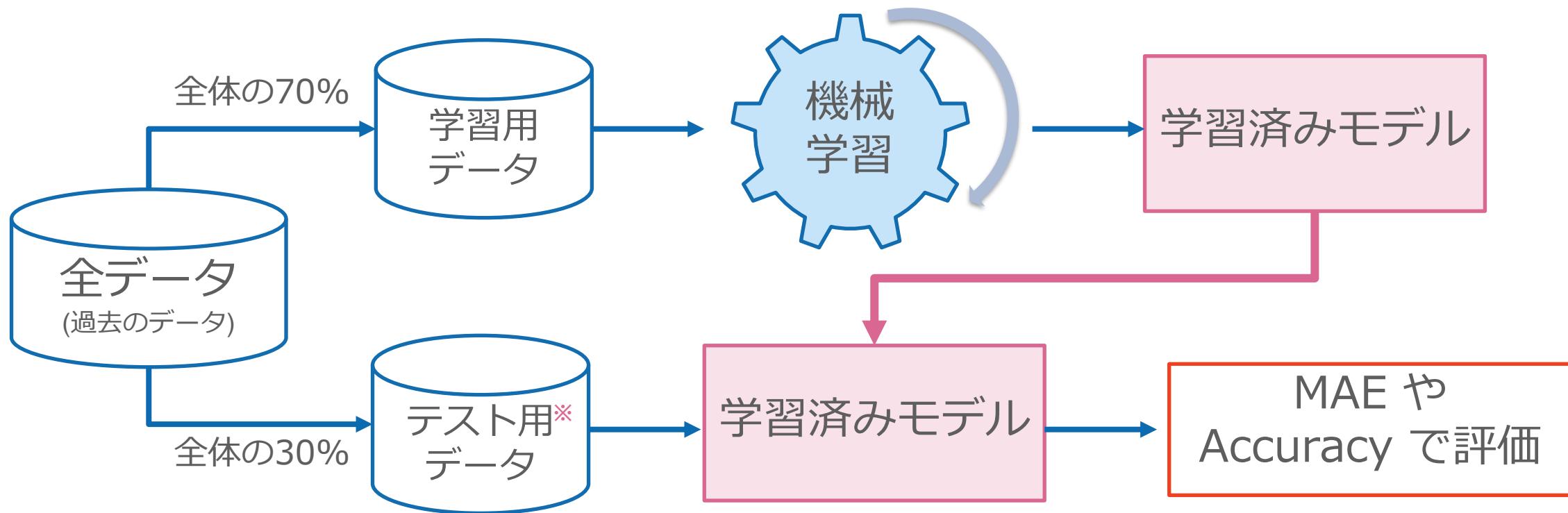
## 汎化誤差の推定

---

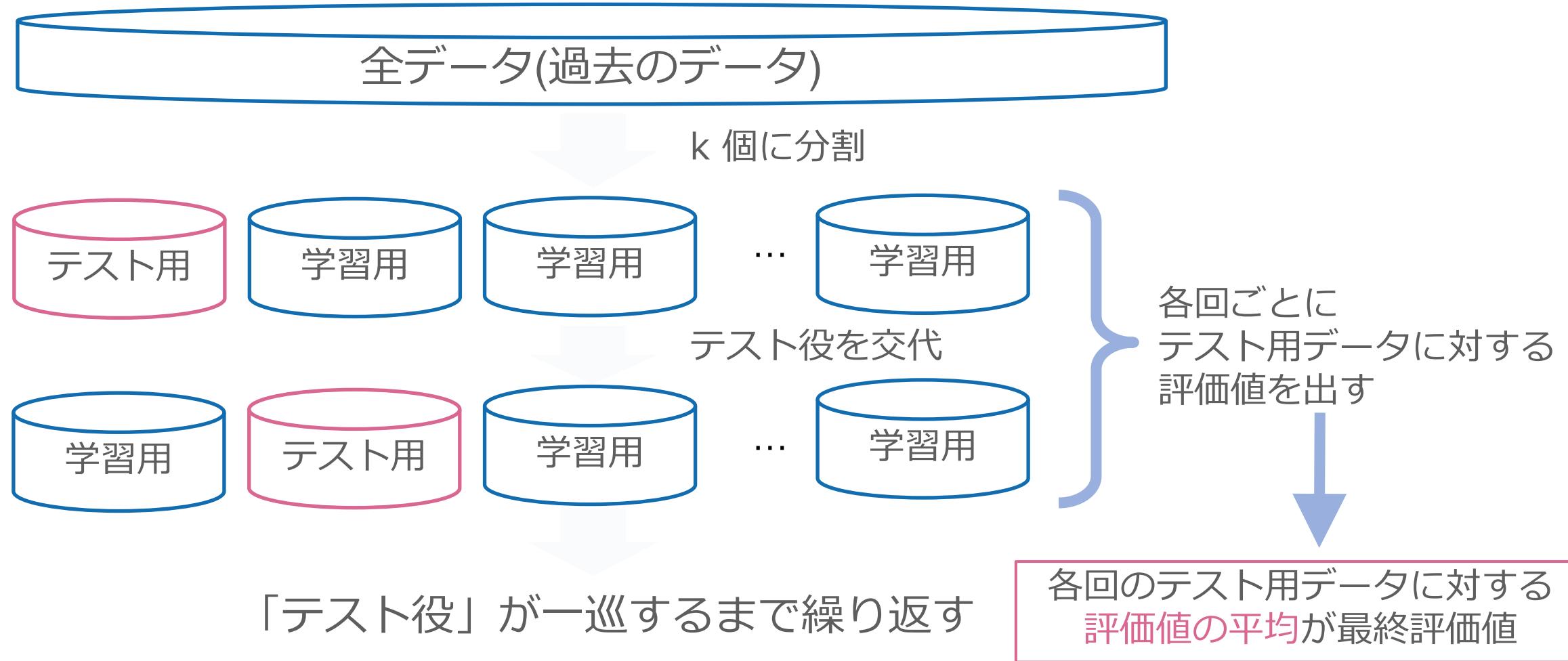
- 機械学習モデルの性能は汎化誤差で評価
- しかし、実は汎化誤差は算出不可能
  - 汎化誤差は理論上、出現しうる未知のデータ全て（＝母集団）に対してモデルの誤差を計算したものであるから
- しかし汎化誤差がどの程度であるか推定することは可能
- 代表的な汎化誤差の推定方法
  - ホールドアウト法
  - 交差検証法（クロスバリデーション法）

- 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推定
- 実装が簡単であるが、データ数が少ないと汎化誤差を正しく評価できない

※実際には「検証用」と  
称することが多い  
(詳細は第5章にて解説)



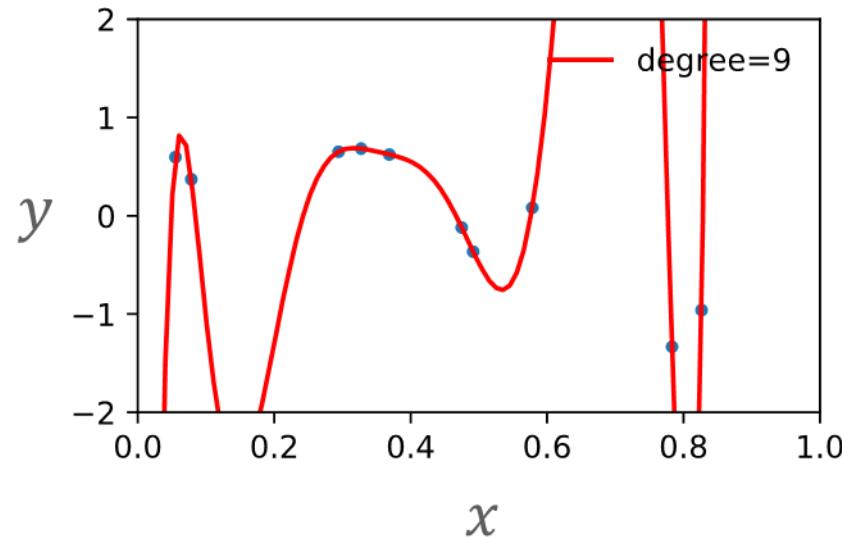
- 学習役とテスト役を交代させることで少ないデータでも汎化誤差を推定する方法
  - データを複数のグループに分割
  - 主にハイパーパラメータをチューニングする際に活用される
- データのグループ数を  $k$  としたとき、 **$k$  分割交差検証法**と呼ばれる
  - 例) グループ数が 5 なら、5 分割交差検証法
- 利点
  - 用意された全てのデータに対する、汎化性能を検証できる
    - ホールドアウト法では、分割時に偏りが生じる可能性がある
    - 例) 予測が簡単なデータが、テスト用データに集まる
- 欠点
  - 学習と推定を何度も繰り返すため、計算時間が長くなりやすい



## 過学習と未学習

---

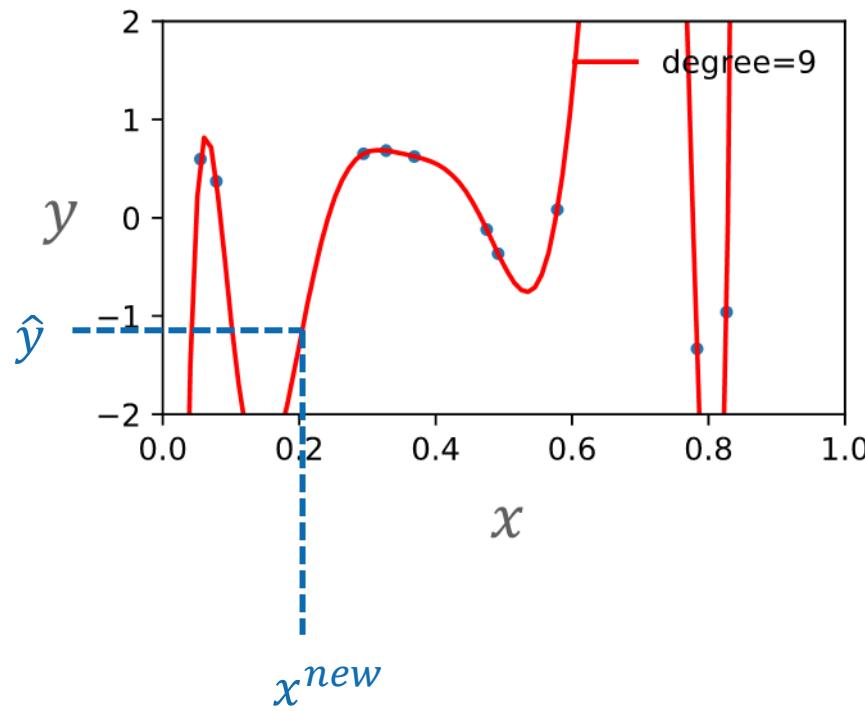
■ 機械学習モデル  $f(x; w)$  を構築した



※赤線のモデルは下記の数式で表現される  
 $\hat{y} = f(x; w) = w_0 + w_1x + w_2x^2 + \cdots + w_9x^9$

訓練誤差は小さい？大きい？  
汎化誤差は小さそう？大きそう？

■ 機械学習モデル  $f(x; w)$  を構築した



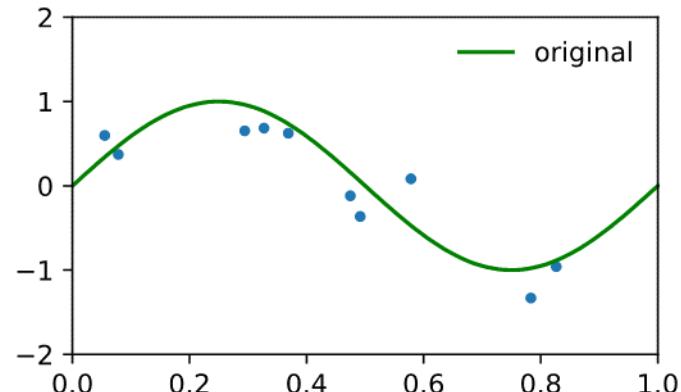
訓練誤差 → ほぼ 0

汎化誤差 → 大きそう

学習データに当てはまり過ぎて

未知のデータに対する予測が全く当たらなそう

真のモデル



$\sin$  関数から  
値をサンプリングして  
ノイズを付与

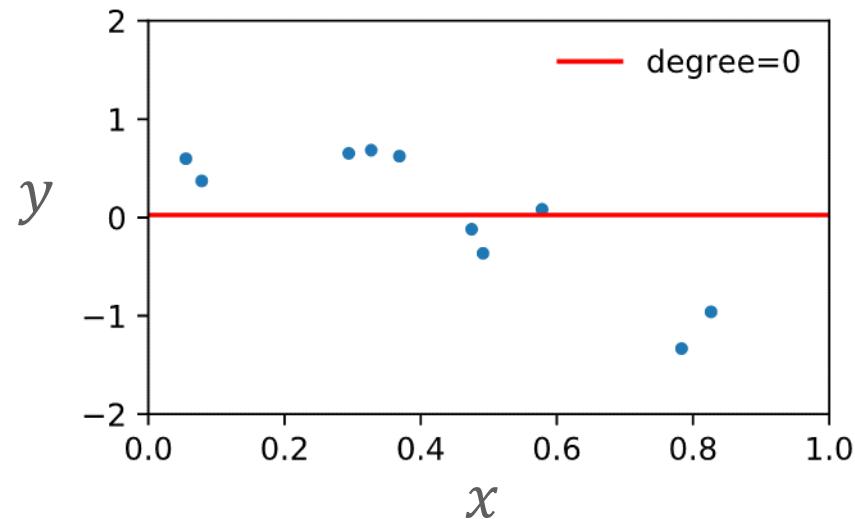
## ■ 過学習

- ・ 訓練誤差は非常に小さいが、**汎化誤差が大きく、未知のデータ**に対する予測性能が小さい状態
- ・ イメージ：練習問題の答えを丸暗記したため、テスト問題を解くことができない
- ・ オーバーフィッティングとも呼ばれる

## ■ 過学習の原因

- ・ モデルの表現能力が高すぎる
  - ・ 木モデルやニューラルネットワークなど非線形モデルと呼ばれる種類のモデルは、**非常に複雑な関数**を構成可能であるため、過学習を引き起こしやすい
- ・ 十分なデータ数が確保できていない
  - ・ 単純なモデルでも、**学習データ数が少ない場合**には過学習が起こる
  - ・ 着目している入力と出力の関係を捉えることができない

■ 機械学習モデル  $f(x; w)$  を構築した

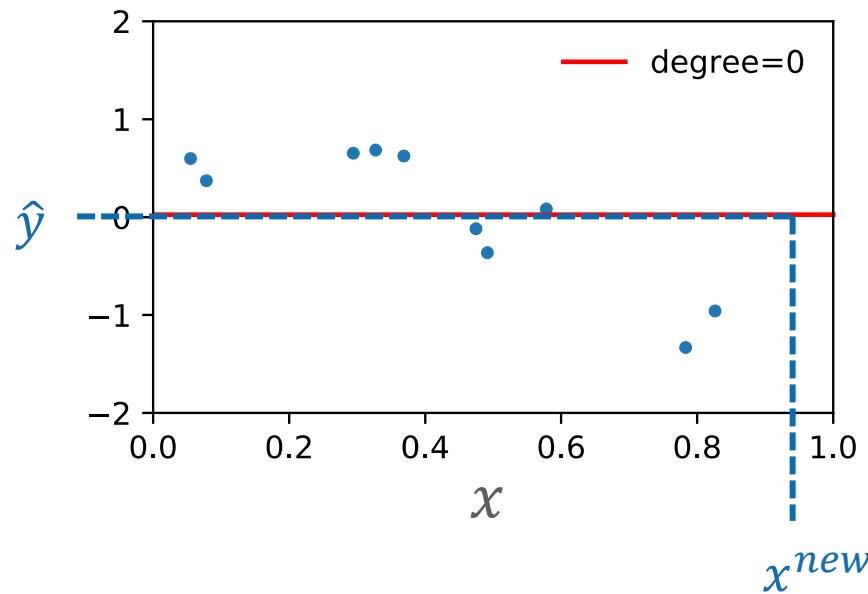


訓練誤差は小さい？大きい？  
汎化誤差は小さそう？大きそう？

※赤線のモデルは下記の数式で表現される

$$\hat{y} = f(x; w) = w_0$$

- 機械学習モデル  $f(x; w)$  を構築した



訓練誤差 → 大きい

モデルが学習データの動きを捉えきれてなさそう

汎化誤差 → 大きそう

学習もままならない状態のモデルだと、

未知のデータに対する予測もうまくいかなそう

## ■ 未学習

- 訓練誤差も汎化誤差も大きく、未知のデータに対する予測性能が小さい状態
- イメージ：練習問題の内容も学習できておらず、実際のテストも当然点を取れない
- アンダーフィッティングとも呼ばれる

## ■ 未学習の原因

- データに対するモデルの表現能力が低い
  - 例) 複雑な入出力関係のデータに対して線形回帰モデルを利用

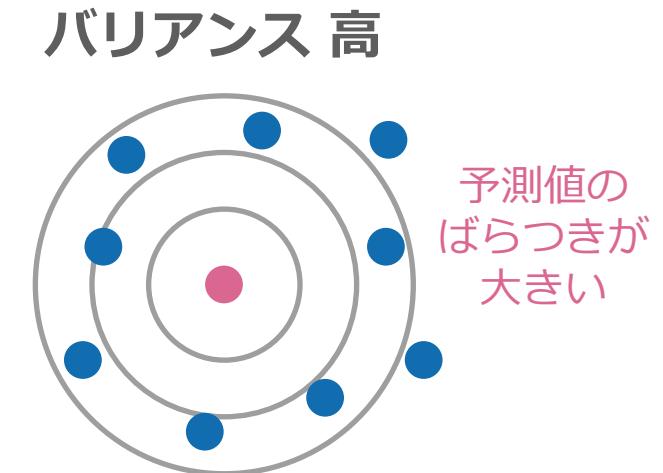
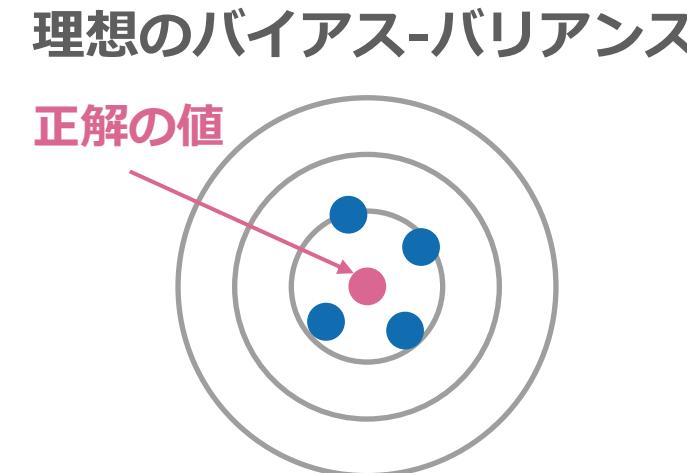
## ■ バイアス (bias)

- 予測値と正解値のズレの度合いを表す
- バイアスが大きいモデル = **未学習を起こしているモデル**

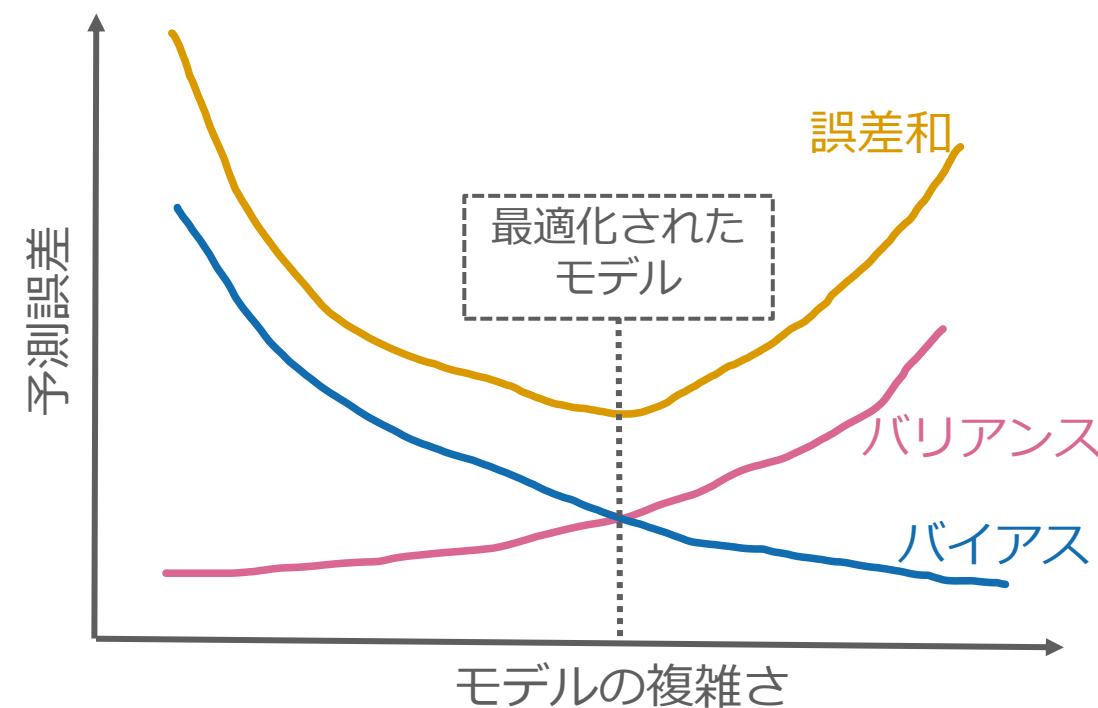
## ■ バリアンス (variance)

- 予測値のばらつき度合いを表す
- バリアンスが大きいモデル = **過学習を起こしているモデル**

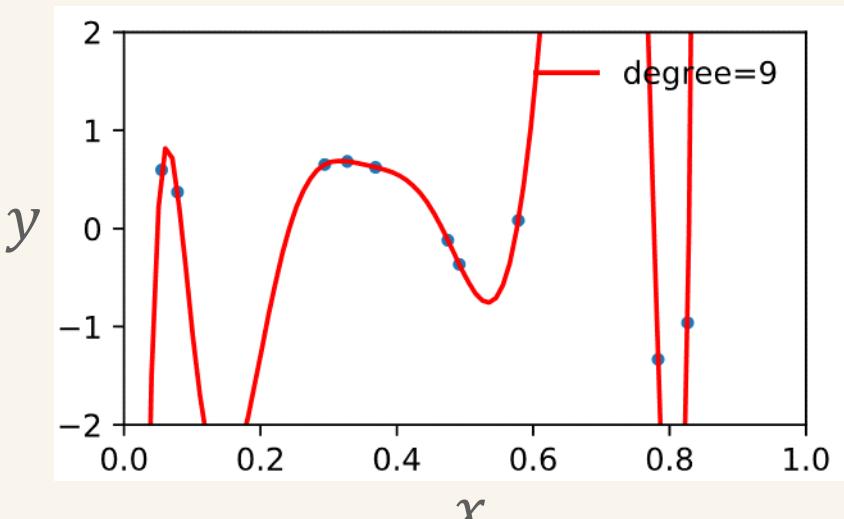
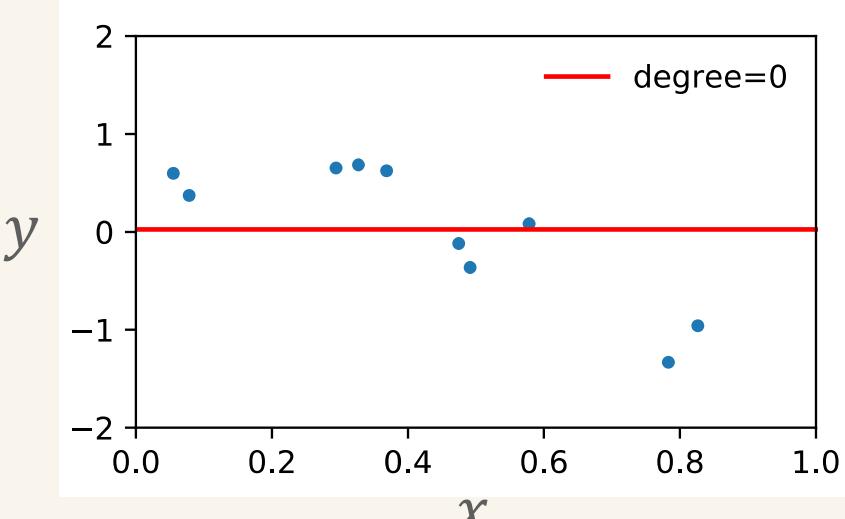
バイアスとバリアンスを「的当て」でたとえたもの



- 両方を同時に小さくすることはできない（トレードオフの関係）
  - ・ バリアンスを小さくする → バイアスが大きくなる
  - ・ バイアスを小さくする → バリアンスが大きくなる
- 正則化によって、バイアスとバリアンスのバランスをとりつつ、過学習を防ぐ



# 過学習と未学習 | まとめ

	過学習	未学習
意味	訓練誤差は小さいが汎化誤差が大きく、未知のデータに対する予測性能が低い状態 (バリアンスが大きい状態)	訓練誤差も汎化誤差も大きく、未知のデータに対する予測性能が低い状態 (バイアスが大きい状態)
イメージ	$y = w_0 + w_1x + w_2x^2 + \cdots + w_9x^9$  A scatter plot showing data points (blue dots) and a fitted curve (red line). The x-axis ranges from 0.0 to 1.0, and the y-axis ranges from -2 to 2. The curve is a high-degree polynomial (degree=9) that passes through every data point, but it exhibits extreme oscillations between the points, indicating overfitting.	$y = w_0$  A scatter plot showing data points (blue dots) and a fitted curve (red line). The x-axis ranges from 0.0 to 1.0, and the y-axis ranges from -2 to 2. The curve is a low-degree polynomial (degree=0), which is a horizontal line at y=0, failing to capture the underlying trend of the data points.
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	モデルの表現能力が低い

# 正則化

---

## ■ モデルに制約を加えることで過学習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過学習を抑制することができる

- 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適用可能
- 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

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

$\hat{y}^{(n)}$  :  $n$  番目のデータの予測値  
 $y^{(n)}$  :  $n$  番目のデータの目的変数  
(正解データ)  
 $\lambda$  : 正則化の強さ  
(値が大きいほど制約が強くなる)

## ■ 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)	$\ w\ _2^2 = \sum w_i^2$ (ユークリッド距離の二乗)	$E_D = \frac{1}{2} \sum_{n=1}^N (\hat{y}^{(n)} - y^{(n)})^2 + \lambda \sum w_i^2$
L1 正則化 (Lasso)	$\ w\ _1 = \sum  w_i $ (マンハッタン距離)	$E_D = \frac{1}{2} \sum_{n=1}^N (\hat{y}^{(n)} - y^{(n)})^2 + \lambda \sum  w_i $
ElasticNet	$\ w\ _2^2$ と $\ w\ _1$ の 2 つ	$E_D = \frac{1}{2} \sum_{n=1}^N (\hat{y}^{(n)} - y^{(n)})^2 + \lambda_1 \sum w_i^2 + \lambda_2 \sum  w_i $

$\lambda$  : 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)

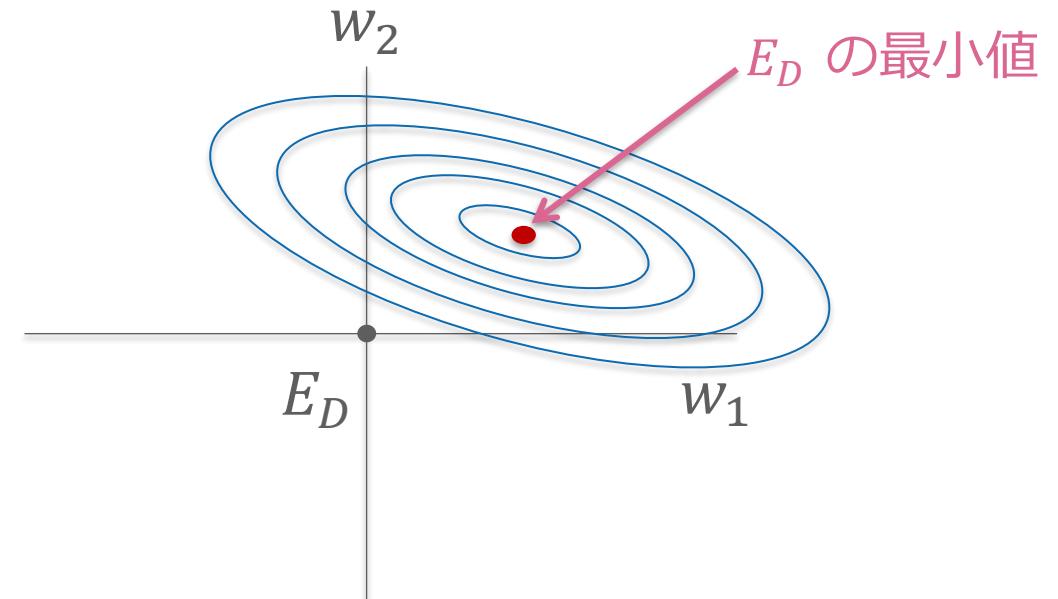
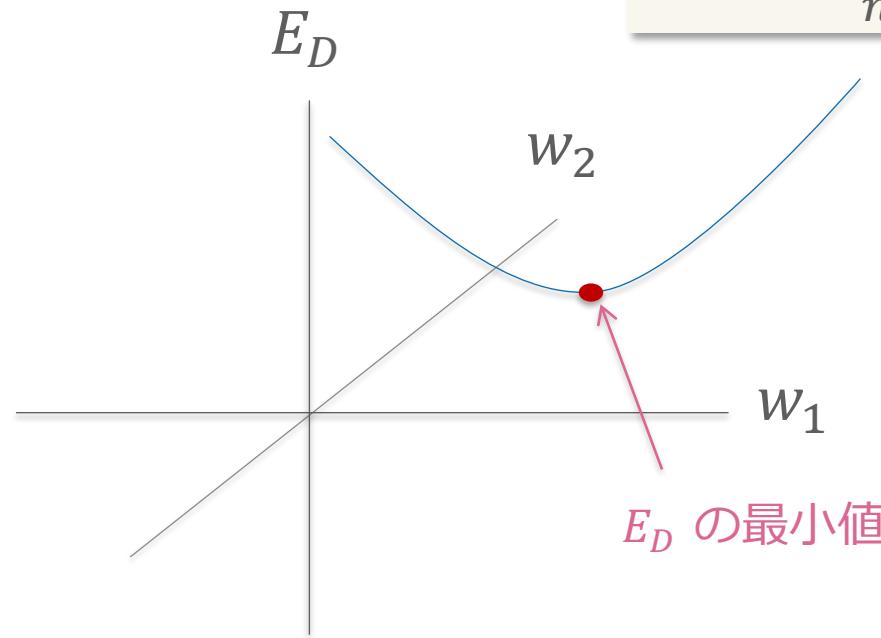
L2 正則化を加えた線形回帰を Ridge 回帰  
L1 正則化を加えた線形回帰を Lasso 回帰と呼ぶ

- L2 正則化は、係数の値ができるだけ小さくなるようにパラメータ探索を実施
- L1 正則化は、係数の値ができるだけ 0 になるようにパラメータ探索を実施
- 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)	$\ w\ _2^2 = \sum w_i^2$ (ユークリッド距離の二乗)	$E_D = \frac{1}{2} \sum_{n=1}^N (\hat{y}^{(n)} - y^{(n)})^2 + \lambda \sum w_i^2$
L1 正則化 (Lasso)	$\ w\ _1 = \sum  w_i $ (マンハッタン距離)	$E_D = \frac{1}{2} \sum_{n=1}^N (\hat{y}^{(n)} - y^{(n)})^2 + \lambda \sum  w_i $
ElasticNet	$\ w\ _2^2$ と $\ w\ _1$ の 2 つ	$E_D = \frac{1}{2} \sum_{n=1}^N (\hat{y}^{(n)} - y^{(n)})^2 + \lambda_1 \sum w_i^2 + \lambda_2 \sum  w_i $

# 正則化 | 正則化項を加える意味

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



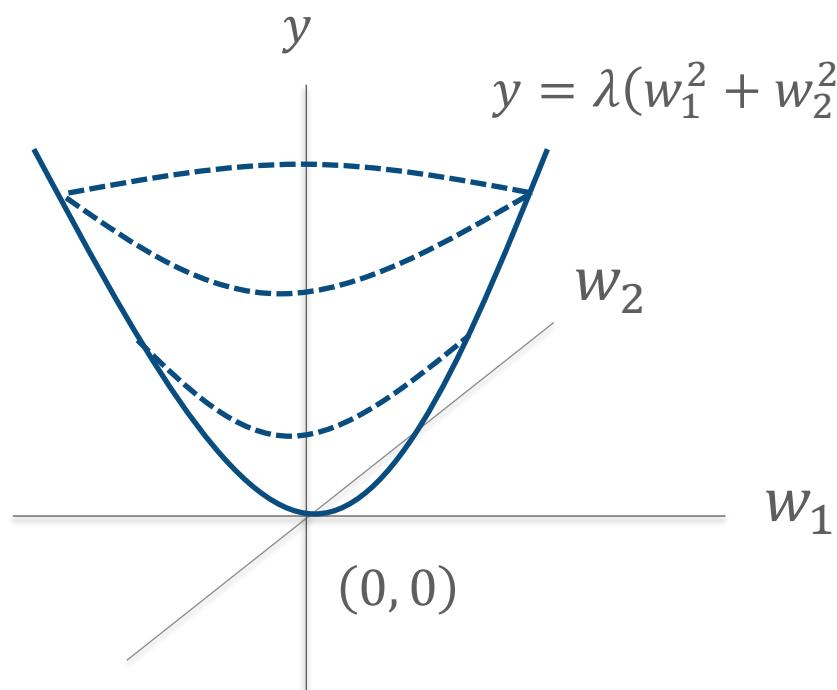
正則化項がない  $E_D$  の最小値 → 訓練誤差が最小の  $w$  を与える場所

その  $w$  では、データに対して過剰に当てはまってしまう可能性がある

訓練誤差最小ではないが、それなりに小さくなる  $w$  を見つけてはどうか？

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

$\lambda \sum w_i^2$  は左下のような図形を描く ( $i = 2$  の二変数の場合)



二乗誤差関数に  $\lambda \sum w_i^2$  を足すということは、左の図形の分だけ、二乗誤差関数の値を大きくするということ

$\lambda \sum w_i^2$  を足した後の  
関数の最小値を探索してみよう

# 正則化 | 正則化項を加える意味 (L2 正則化)

正則化項を付与することで  
 $E_D$  に最小値を与える  $w$  の場所がずれ、  
訓練誤差が小さくなり過ぎない  $w$  が得られる

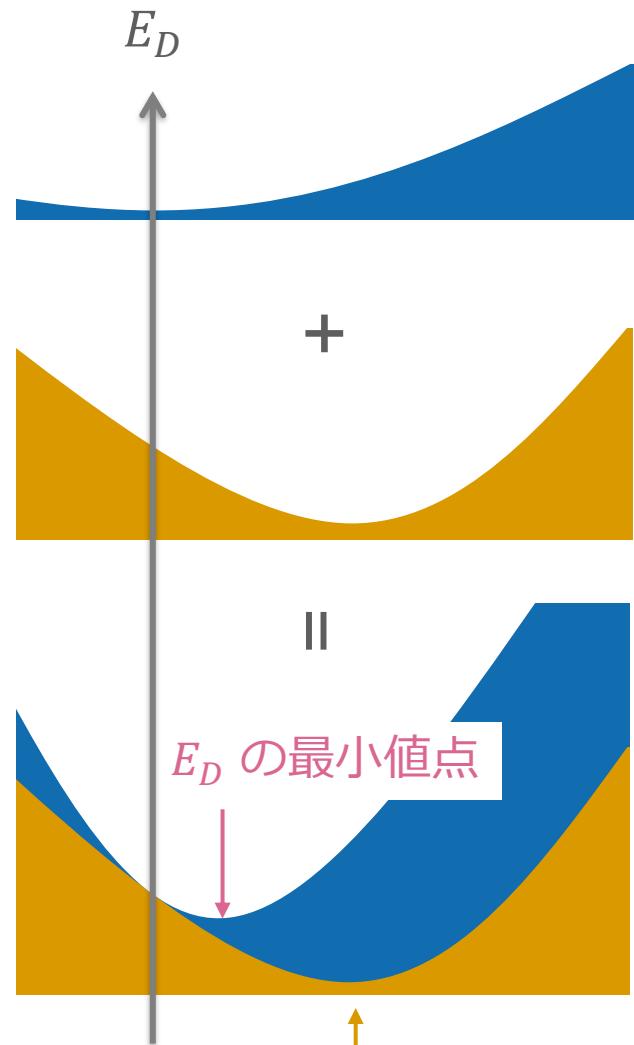
$\lambda \sum w_i^2$  は原点で最小値を取る関数



原点から離れるにつれて大きな値が  
二乗誤差関数の値に足される



正則化項を付与した  $E_D$  に  
最小値を与える  $w$  の各要素の値は  
自然と原点に寄る (= 小さくなる)



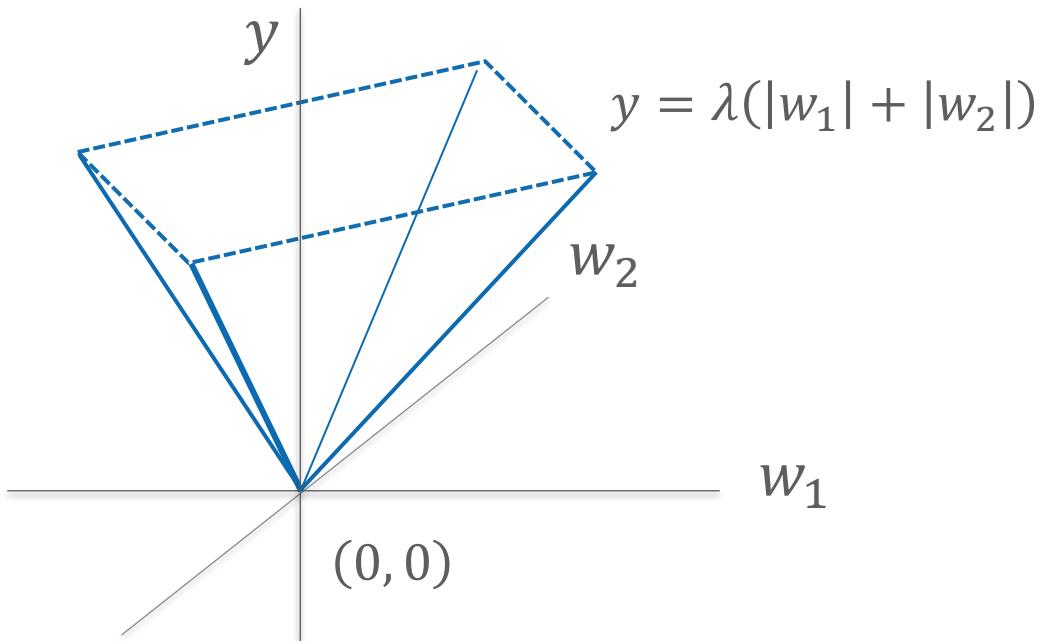
元々の二乗誤差関数の最小値点

$\lambda \sum w_i^2$   
増やす量

$\frac{1}{2} \sum_{n=1}^N (\hat{y}^{(n)} - y^{(n)})^2$   
元々の関数

$E_D$   
新しい関数

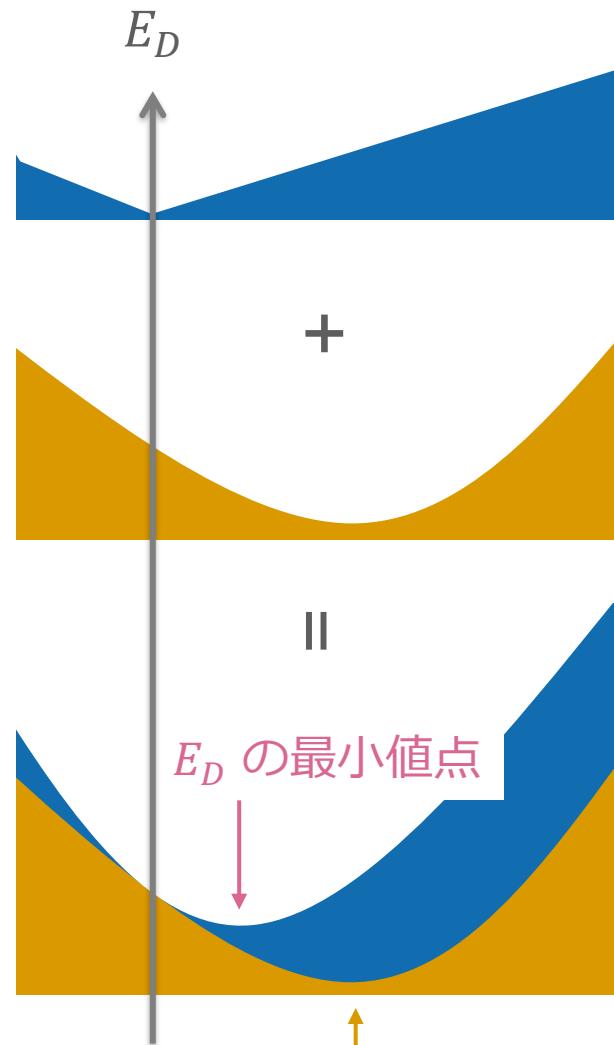
# 正則化 | 正則化項を加える意味 (L1 正則化)



$\lambda \sum |w_i|$  は原点で最小値を取り、軸上で窪む関数



正則化項を付与した  $E_D$  に  
最小値を与える  $w$  の各要素は  
窪んだ軸上に誘導されやすい  
( $w$  の各要素は 0 になりやすい)



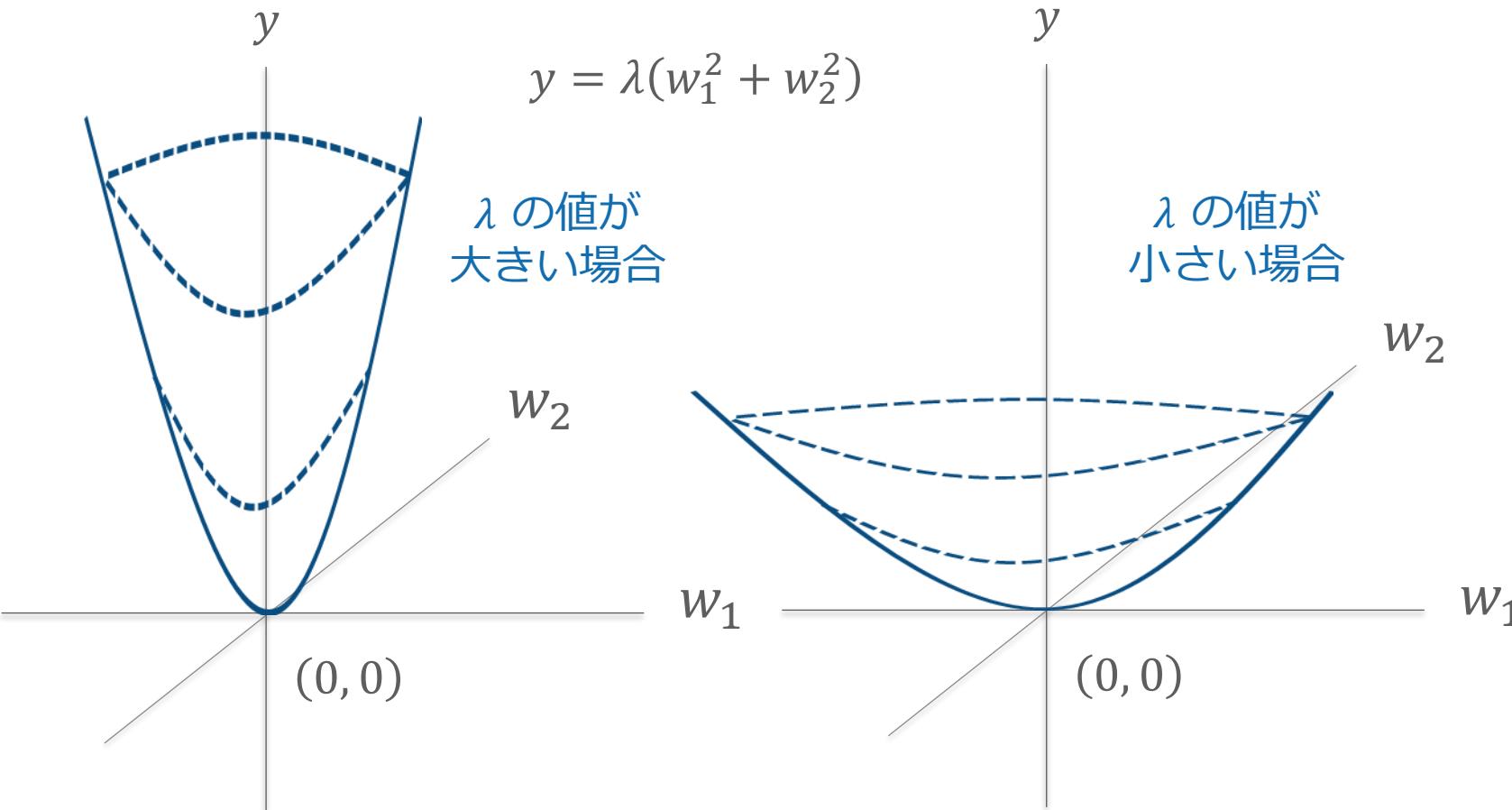
元々の二乗誤差関数の最小値点

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

$\lambda \sum |w_i|$  増やす量  
元々の関数

$E_D$   
新しい関数

## ■ 二乗誤差関数に対する値の加え方が変わる



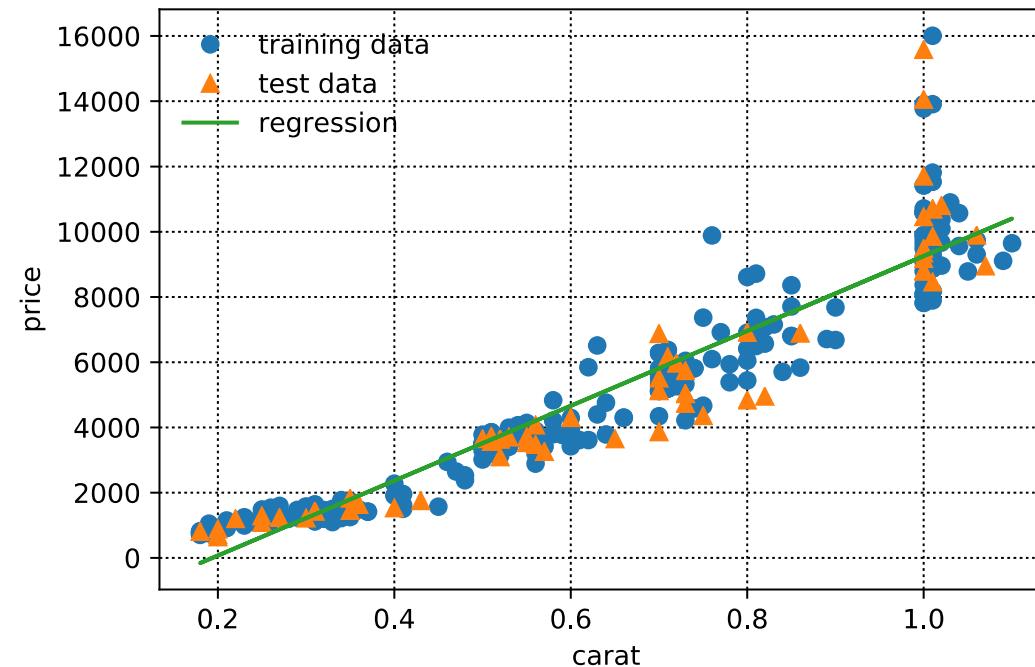
$\lambda$  の値を大きくすると  
正則化の影響が強まる  
(誤差関数全体の最小値を  
原点に誘導する力が強くなる)

## ノートブック演習

---

## 4-1\_how\_to\_validation.ipynb

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



Fold 1  
MAE = 1341.723

Fold 2  
MAE = 1239.992

Fold 3  
MAE = 949.845

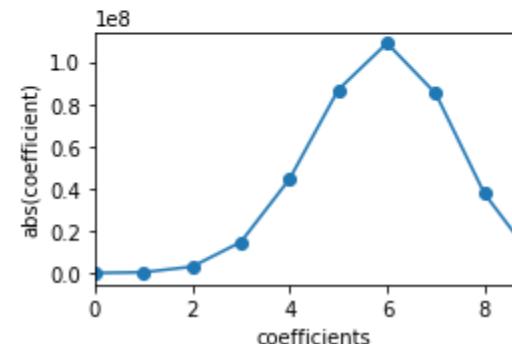
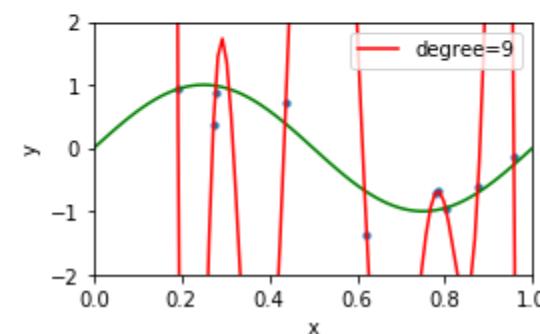
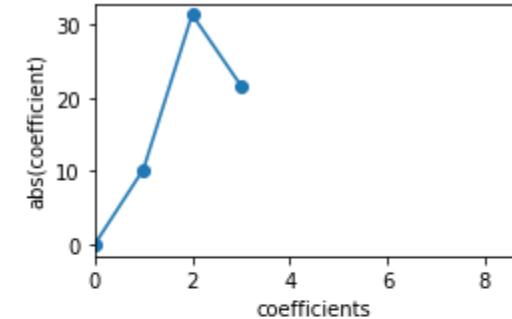
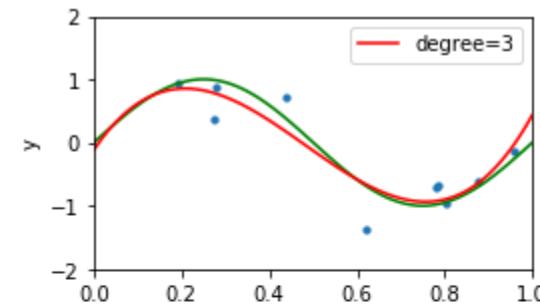
Fold 4  
MAE = 1136.542

Fold 5  
MAE = 1342.977

Cross Validation MAE = 1202.216

## 4-2\_regularization.ipynb

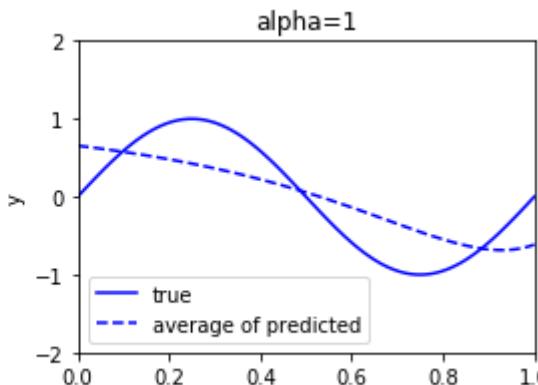
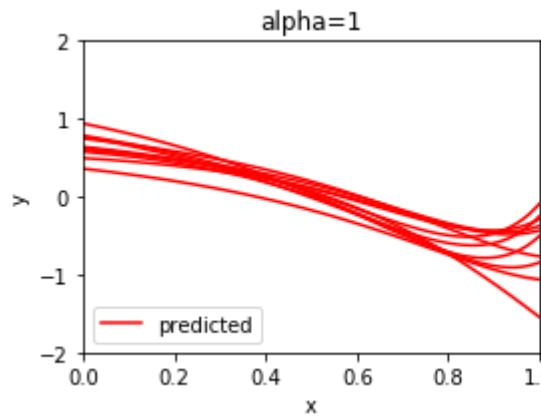
- 多項式特徴量を用いたモデルを使って、過学習や未学習を確認してみよう
- 正則化によって、過学習が緩和されることを確認してみよう
- 正則化の強さを変化させることで、得られるモデルにどのような変化が起こるか確認してみよう



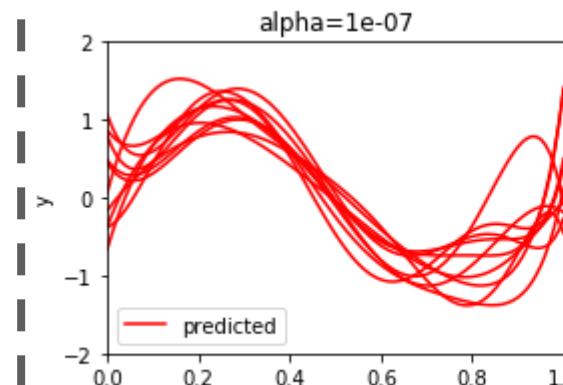
## 4-3\_bias\_variance\_noise.ipynb

- 多項式特徴量を用いたモデルを使って、バイアスの大きいモデルとバリアンスの大きいモデルの性質を確認してみよう

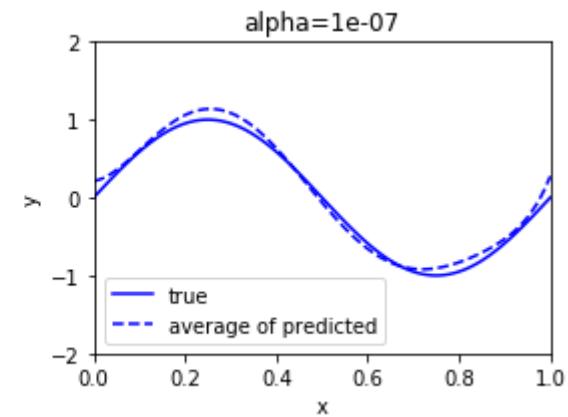
10 個のモデルに、それぞれ異なるノイズを含む  $\sin$  波を学習させた場合



予測値の誤差が大きい  
(バイアスが大きい)



予測値のばらつきが大きい  
(バリアンスが大きい)



## 本章のまとめ

---

- 訓練誤差と汎化誤差
- 汎化誤差の推定
- 過学習と未学習
- 正則化
- ノートブック演習

## ■ 訓練誤差

- 学習データに対するモデルの誤差

## ■ 汎化誤差

- 未知のデータに対するモデルの誤差
- ホールドアウト法や交差検証法を用いて値を推定可能

## ■ 過学習

- 訓練誤差は非常に小さいが、汎化誤差が大きい状態

## ■ 未学習

- 訓練誤差も汎化誤差も大きい状態

## ■ 正則化

- 機械学習モデルの過学習を防ぐ工夫の 1 つ
- 例) L1 正則化 (Lasso 回帰) 、 L2 正則化 (Ridge 回帰) 、 ElasticNet

# 現場で使える 機械学習・データ分析基礎講座

第 5 章：モデルの構築と改良

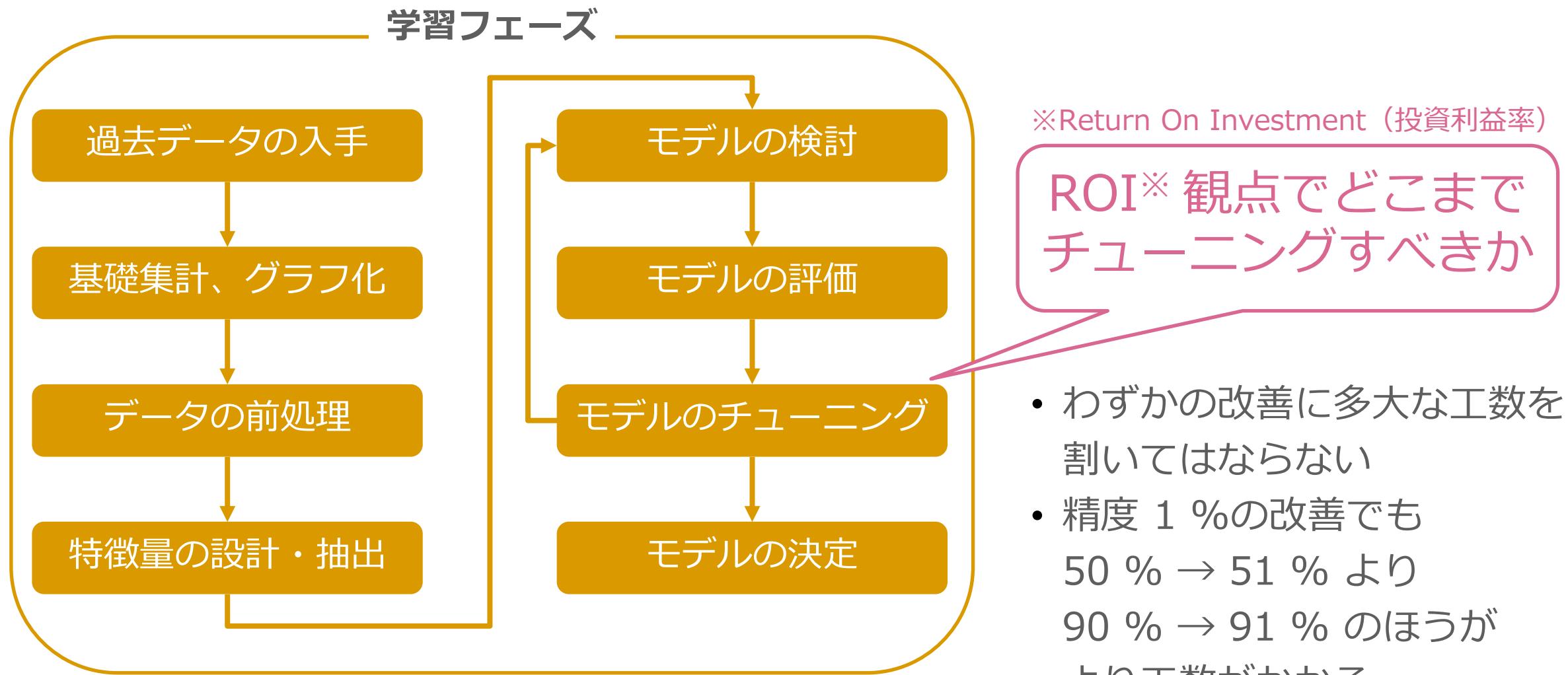
- モデルのチューニング
- ハイパーパラメータ
- ハイパーパラメータ探索

- ハイパーパラメータとは何か説明できるようになる
- ホールドアウト法もしくは k 分割交差検証法を用いた際のモデルのチューニングの流れを説明できるようになる
- ハイパーパラメータ探索の手法を 3 つ説明できるようになる

## モデルのチューニング

---

# 【復習】機械学習モデルの構築・運用の流れ



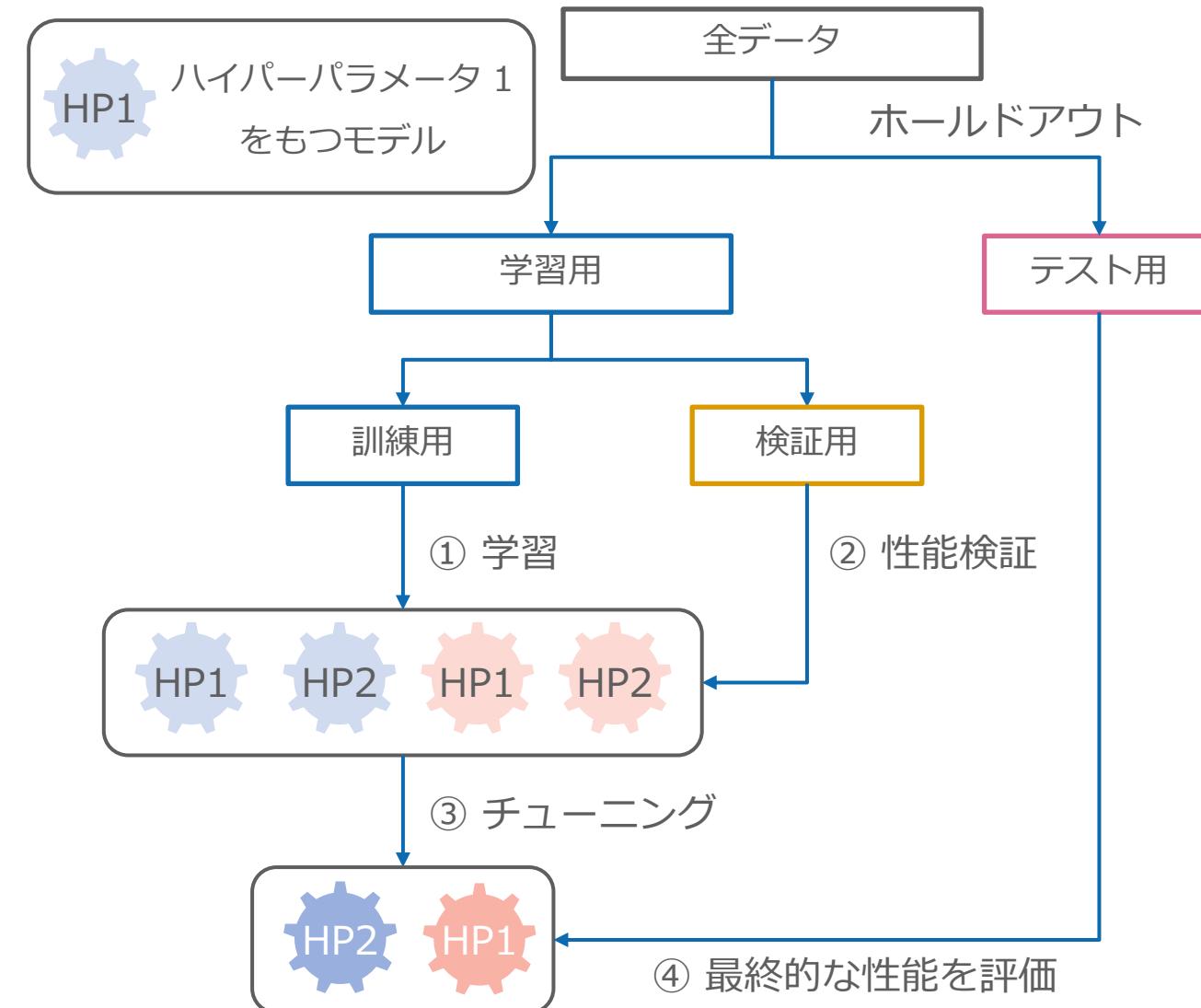
## ■ 基本的な流れ

- データの分割
  - 全データセットを学習用データとテスト用データに分割
  - 学習用データをさらに訓練用データと検証用データに分割
- モデルのチューニング
  - ベースとなるモデルとハイパーパラメータの設定
  - テスト用データを用いて最終的な性能を評価

## ■ モデルのチューニングにおいては、ベースのモデルが同じであっても、異なるハイパーパラメータをもつならば異なるモデルと考える

- 例) 下記のモデルは全て異なるモデル
  - 正則化をしない線形回帰モデル
  - $\lambda = 0.1$  の L2 正則化つき線形回帰モデル (Ridge 回帰)
  - $\lambda = 0.5$  の L2 正則化つき線形回帰モデル (Ridge 回帰)

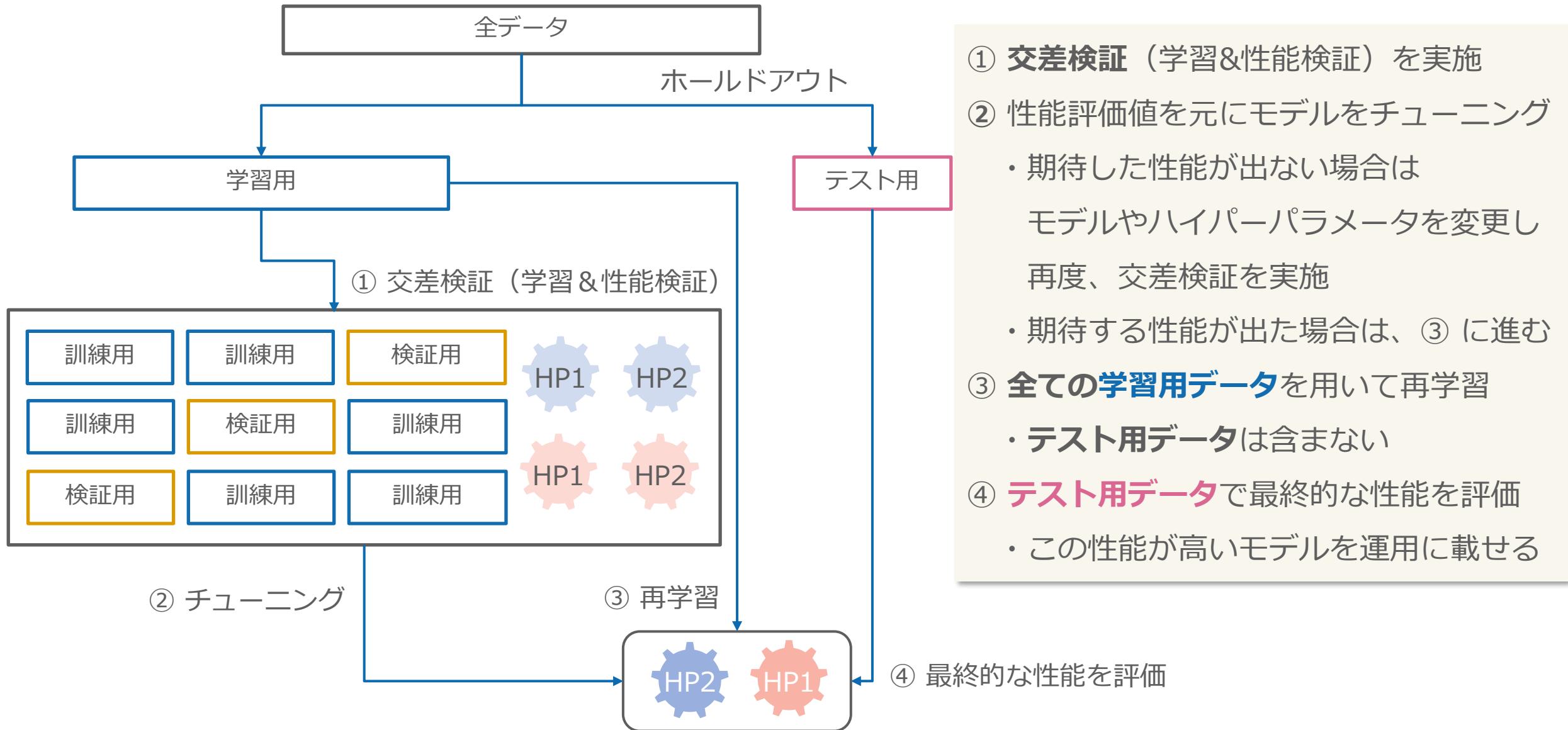
# モデルのチューニング | ホールドアウト法を用いる場合



- ① **訓練用データ**で学習
- ② **検証用データ**で性能を検証
- ③ 性能評価値を元にモデルをチューニング
  - ・期待した性能が出ない場合はモデルやハイパーパラメータを変更して、再度 ① (学習) を行う
  - ・期待する性能が出た場合は、④ に進む
- ④ **テスト用データ**で最終的な性能を評価
  - ・この性能が高いモデルを運用に載せる

**モデル選択をテスト用データで実施しないこと！**  
テスト用データに対して性能が出るような、チューニングを実施していることになり、これは一種のリーケージ※

# モデルのチューニング | 交差検証法を用いる場合



# ハイパーパラメータ

---

## ■ モデルには 2 種類のパラメータが存在

- 学習パラメータ
- ハイパーパラメータ

## ■ 学習パラメータ

- 学習（最適化）により自動的に決定されるパラメータ
- 例) 線形回帰モデルの  $f(x; w) = w^T x$  における  $w$

## ■ ハイパーパラメータ

- 学習を行う前に、分析者がモデルに与えるパラメータ
- 例)
  - k近傍法における参照近傍数  $k$  の値
  - 正則化の強さを決める  $\lambda$  の値

- ハイパーパラメータが異なると学習後のモデルの性能は異なる
- 適切なモデルを選択するためには、様々なハイパーパラメータを試し、汎化性能が良くなりそうな値を発見する必要がある

## ハイパーパラメータ探索

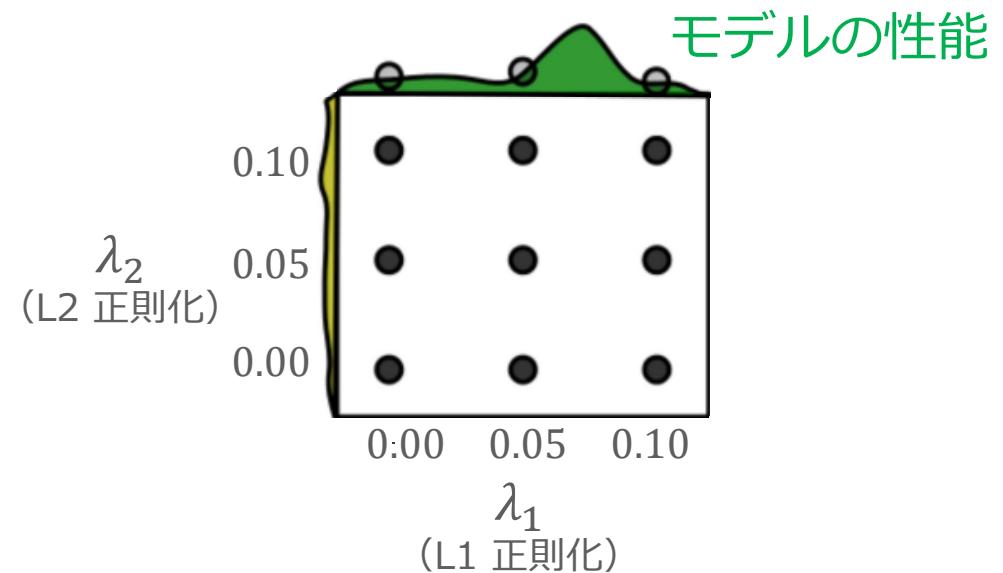
---

## ■ 代表的なハイパーパラメータ探索の手法

- グリッドサーチ
- ランダムサーチ
- ベイズ最適化

- 分析者が、探索する値の候補を事前に選択しておく方法
  - 例) ElasticNet のハイパーパラメータ  $\lambda_1, \lambda_2$  を  $[0.00, 0.05, 0.10]$  の 3 通りで試す
- 特徴
  - 総当たり手法に近く、良いハイパーパラメータを得られるまでの学習回数が大きくなる可能性がある
  - 選び出した値の間に最適なハイパーパラメータがあったとしても、見逃してしまう

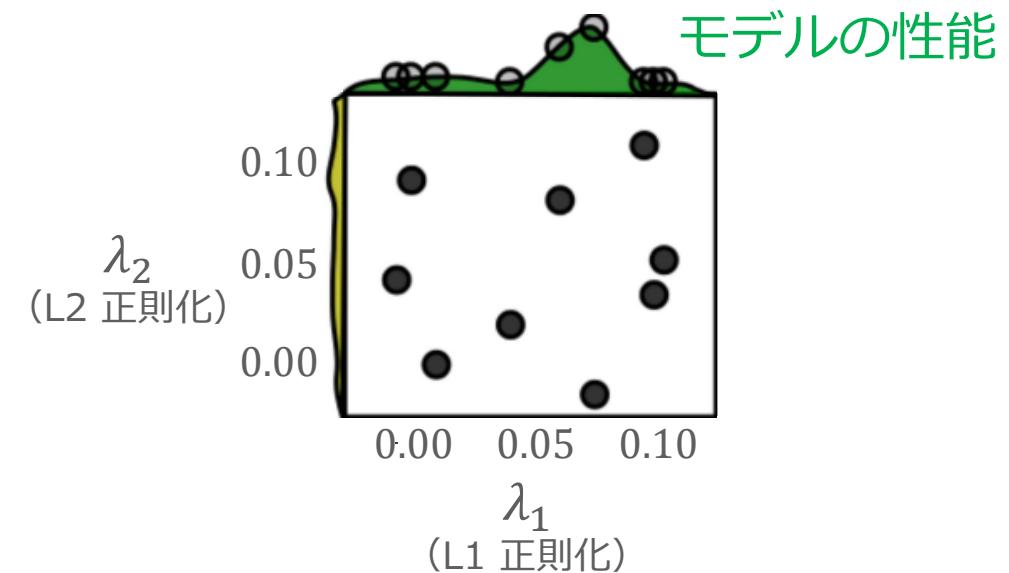
探索候補の設定が悪いと  
良い性能をとる値を見逃してしまう



出典：[Random Search for Hyper-Parameter Optimization](#)

- 亂数を用いて、探索する値の候補から値を選択する方法
  - 例) ElasticNet のハイパーパラメータ  $\lambda_1, \lambda_2$  を  $[0.00, 0.10]$  の範囲でランダムに試す
- 特徴
  - 探索に指向性がないため、良いハイパーパラメータを得られるまでの学習回数が大きくなる可能性がある
  - 偶然、良いハイパーパラメータの値を取得できる可能性がある

0.00 以上  
未満

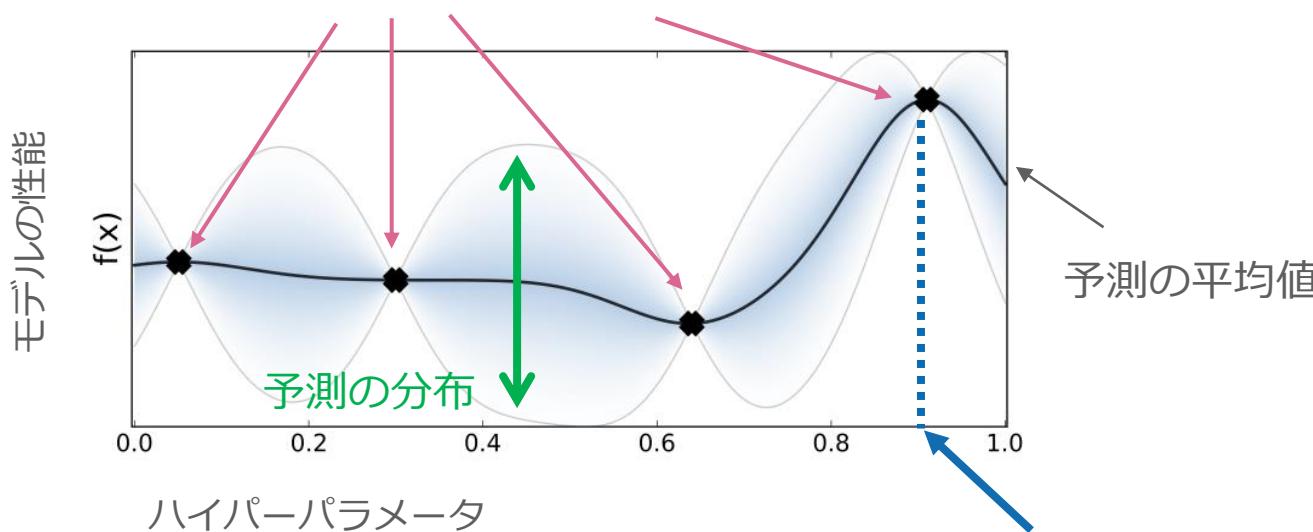


出典：[Random Search for Hyper-Parameter Optimization](#)

## ■ ハイパーパラメータの値を元に、モデルの評価値を予測するモデルを用いる方法

- ・ 高性能が出そうなハイパーパラメータの値を予測しながら、次に設定する値を決定
- ・ 少ない探索回数でも良いハイパーパラメータを見つけやすい
- ・ ニューラルネットワークなど学習に時間がかかるモデルのハイパーパラメータ探索に適する

過去のハイパーパラメータ探索によって得られた点



出典：[Introduction to Bayesian Optimization](#)

次に試すハイパーパラメータの値とする

### 参考情報

- ・ [Jasper Snoek et al., Practical Bayesian Optimization of Machine Learning Algorithms.](#)
- ・ [ガウス過程と機械学習](#)
- ・ [OPTUNA](#)
- ・ ベイズ最適化を行う Python ライブラリ

## 本章のまとめ

---

- モデルのチューニング
- ハイパーパラメータ
- ハイパーパラメータ探索

## ■ モデルのチューニングの流れ

- データ分割 → モデル選択 → テスト用データで性能評価
- 訓練用データ
- 検証用データ
- テスト用データ

## ■ ハイパーパラメータ

- 学習を行う前に、分析者がモデルに与えるパラメータ
- 例) L2 正則化における  $\lambda$  の値

## ■ ハイパーパラメータ探索の方法

- グリッドサーチ：分析者が事前に値の候補を設定
- ランダムサーチ：乱数を用いて値を設定
- ベイズ最適化：ハイパーパラメータの値を元に評価値を予測しながら、最適な値を決定

# 現場で使える 機械学習・データ分析基礎講座

第 6 章：代表的な前処理

- 欠損値処理
- 外れ値・異常値処理
- カテゴリ変数の変換
- 正規化と標準化
- 無相関化と白色化
- ノートブック演習

- 欠損値処理の具体的な方法を説明できる
- 外れ値処理の具体的な方法を説明できる
- カテゴリ変数の変換方法を説明できる
- 正規化と標準化を説明できる
- 無相関化と白色化を説明できる

# 欠損値処理

---

## ■ 欠損値

- あるデータの変数において、**値が存在しない（欠落している）** 項目
- 欠損値の含まれたデータセットを、モデルに読み込ませることはできない
- 何らかの値で**補完**するか、欠損値を含む列もしくは行を**削除**することで対処

	身長 [cm]	体重 [kg]	年齢
A	180	64	28
B	163	58	42
C	191	NaN	31

## ■ 欠損値の処理方法

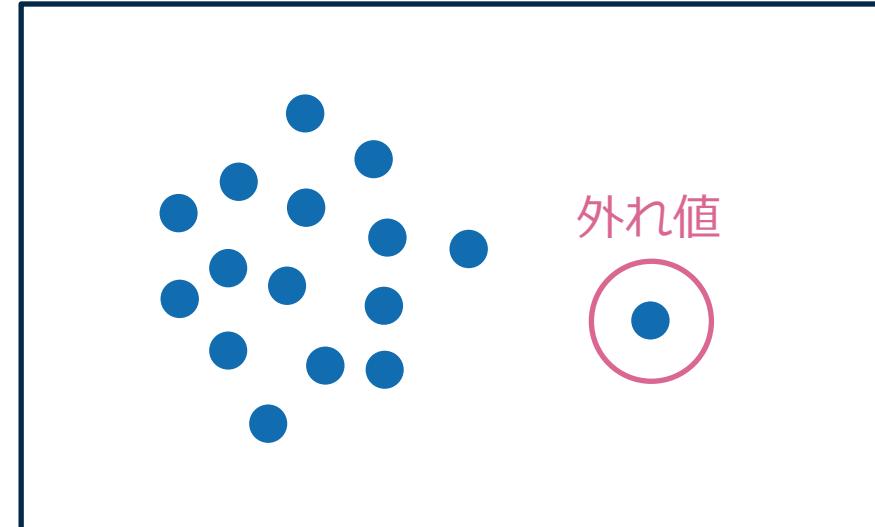
- 欠損値を含む行または列をデータセットから除外
  - 貴重なデータを失うため、なるべく欠損値の補完を実施
- 欠損値を補完
  - 量的変数（数値）の場合
    - 列の平均値や中央値などの代表値で補完
  - 質的変数（カテゴリ）の場合
    - 列内で最も頻繁に発生する値で補完
  - 時系列データの場合
    - 欠損した時刻の前後の値の平均値や移動平均などによって補完
  - 欠損値が含まれない特徴を説明変数とした回帰モデルを構築し、欠損値を予測することも可能

## 外れ値・異常値処理

---

## ■ 外れ値

- 分布の全体と見比べて、極端に値の大きさが異なるデータ
- 外れ値があると、その値に引っ張られて学習が上手くいかない場合がある
- 外れ値のうち、その値をとった原因が分かるものを、特に異常値と呼ぶことがある
  - 例) 入力ミス、測定の失敗

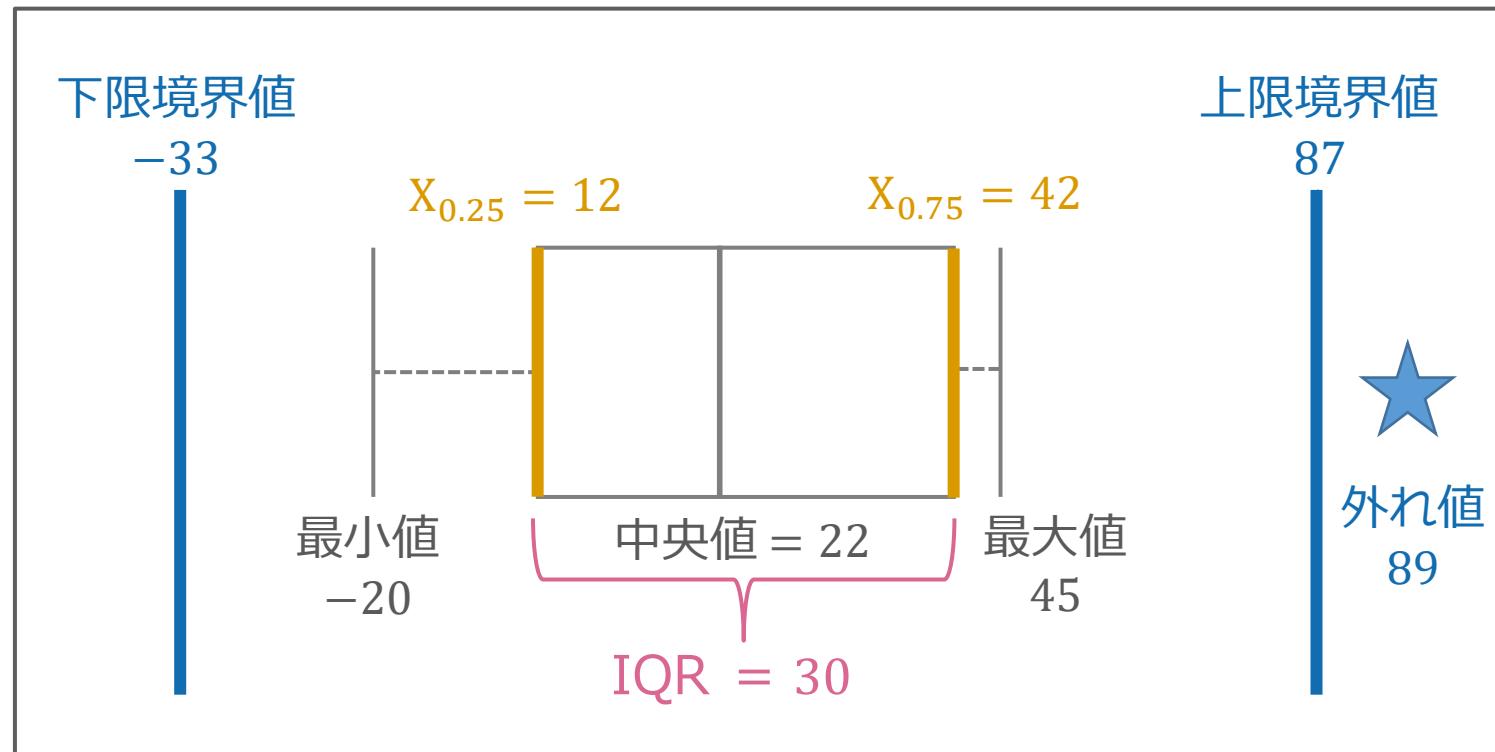


## ■ 外れ値の処理方法

- 四分位範囲 (Interquartile Range; IQR) を用いた外れ値除去
  - 以下の条件から外れたデータを除去
    - 上限境界値 : 第 1 四分位数 - 1.5 \* IQR
    - 下限境界値 : 第 3 四分位数 + 1.5 \* IQR
- ウィンザー化 (Winsorization)
  - 外れ値を一定の範囲内に丸め込む
  - 上位 5 % or 下位 5 % を超えたものは、上位 5 % or 下位 5 % の値に丸め込む

- 箱ひげ図を用いると、IQR を手軽に可視化できる

例)  $[-20, -13, 12, 17, 20, 22, 23, 39, 42, 45, 89]$  に対する箱ひげ図



中央値 : 22

第 1 四分位数 ( $X_{0.25}$ ) : 12

第 3 四分位数 ( $X_{0.75}$ ) : 42

上限境界値 : 87 ( $42 + 1.5 \times 30$ )

下限境界値 : -33 ( $12 - 1.5 \times 30$ )

## カテゴリ変数の変換

---

- カテゴリ変数（質的変数）は、そのままでは学習に利用できない
  - テキストなど
- 適切な形式の数値に変換（ダミー変数化）する必要がある
- 代表的な変換の方法
  - ワンホットエンコーディング
  - ラベルエンコーディング

## ■ カテゴリ変数をワンホットベクトルに変換

- 0と1を要素を持つ多次元ベクトル
- ベクトルの各次元がカテゴリの種類に対応

## ■ 特徴に対して重み付けを行うモデルにおいて有効な変換

- 線形回帰モデルやニューラルネットワークなど

厳密には、列を1つ削除する必要がある  
(多重共線性を回避するため)

カテゴリ変数

色
赤
赤
青
黒
黒

ワンホットエンコーディング



ワンホットベクトル

赤	青	黒
1	0	0
1	0	0
0	1	0
0	0	1
0	0	1

## ■ ワンホットベクトルを特徴量として用いた線形回帰モデル

$$\hat{y} = f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_{red} + w_2 x_{blue}$$

## ■ 特徴量 $x_{red}, x_{blue}$ が 0 or 1 のフラグとして機能する

- 入力が赤であれば出力に対して  $w_1$  だけが寄与 ( $x_{blue} = 0$  となるため、 $w_2$  は出力に寄与しない)

- カテゴリの種類毎にユニークな番号を与えて数値に変換する方法
  - ・ 変換後の数値の間に意味が生じる
- ルールベースに近い判断を実施するモデルに対して有効
  - ・ 「値が 0 であるか、1 であるか、2 であるか？」
  - ・ 木モデルなど



## ■ 特徴に対して重み付けを行うモデルでは、この変換は上手く機能しない

- ・ 線形回帰、ロジスティック回帰、ニューラルネットワークなど
- ・ 例) ラベルエンコーディング後の変数を用いた線形回帰モデル

$$\hat{y} = f(\mathbf{x}; \mathbf{w}) = w_0 + w_1 x_{color}$$

$x_{color} = 0 \rightarrow$  赤

$x_{color} = 1 \rightarrow$  青

$x_{color} = 2 \rightarrow$  黒

変換後の数値に、大小関係などの意味が生じる  
モデルに大小関係を考慮させても、学習が困難になるだけ  
(重みを最適化しにくくなる)

そもそも「0, 1, 2」には「違う色」以外の意味がない

## ■ 決定木系のモデルでは、以下のように使い分ける

- モデルを軽量化したい場合は、**ラベルエンコーディング**を用いる
  - 一般的に、ワンホットエンコーディングを用いると、メモリの消費量が多くなる
- カテゴリの種類が多い場合は、**ワンホットエンコーディング**を用いた方がよいこともある
  - ラベルエンコーディングを用いると、数値の大小関係を基準とした  
**不適切な分岐条件**ができてしまう可能性がある

決定木は第8章にて  
詳しく扱うモデル

カテゴリ変数

色
赤
橙
黄
緑
青

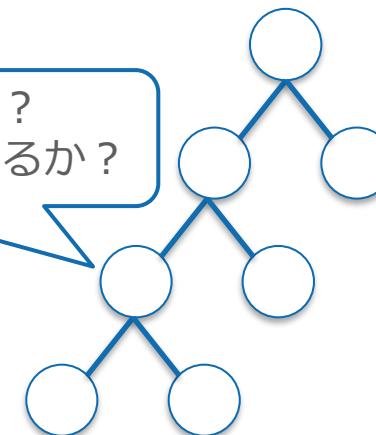
数値

色
0
1
2
3
4



OK：「色」が 2 であるか?  
NG：「色」が 2 以上であるか？

決定木の例



## ■ サポートベクターマシン (SVM) では、以下のように使い分ける

- 扱う変数が順序尺度である場合、ラベルエンコーディングを用いる
  - 「順序」の概念をもつカテゴリ変数
  - ランキングの順位、アンケートの回答（5段階評価）など
- 扱う変数が名義尺度である場合、ワンホットエンコーディングを用いる
  - 「順序」の概念をもたないカテゴリ変数
  - 国名、都道府県、性別など

SVM は第 10 章にて  
詳しく扱うモデル

## 正規化と標準化

---

- 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が異なる
  - ・ 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 <sup>2</sup> ]
8499	12	371.29
3980	2	185.38
3680	20	135.5
...	...	...
4250	2	117.24

- 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る
- スケールの不統一は学習を不安定にさせ、係数の解釈を困難にする
- データのスケールを整える前処理として、正規化もしくは標準化を施す

- 特徴  $x$  のスケールを 0 から 1 の間に収める
- 各特徴ごとに最大値と最小値を求め、以下の式による変換を実施
  - 最大値と最小値は学習用データから算出
  - テストデータに対する正規化も学習用データから求めた値を利用する
  - 外れ値に影響されやすいため注意

$$x'_i = \frac{x_i - \max}{\max - \min}$$

$x_i$  : 特徴  $x$  の  $i$  番目

$\max$  : 特徴  $x$  の最大値

$\min$  : 特徴  $x$  の最小値

- 特徴  $x$  のスケールを平均 0、標準偏差 1 に変換
- 各特徴ごとに平均と標準偏差を求め、以下の式による変換を実施
  - 平均値と標準偏差は学習用データから算出
  - テストデータに対する標準化も学習用データから求めた値を利用する

$$x'_i = \frac{x_i - \mu}{\sigma}$$

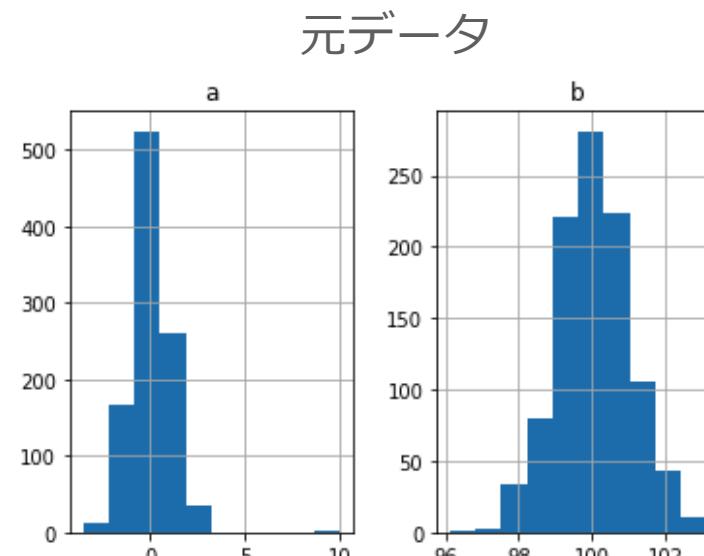
$x_i$  : 特徴  $x$  の  $i$  番目

$\mu$  : 特徴  $x$  の平均

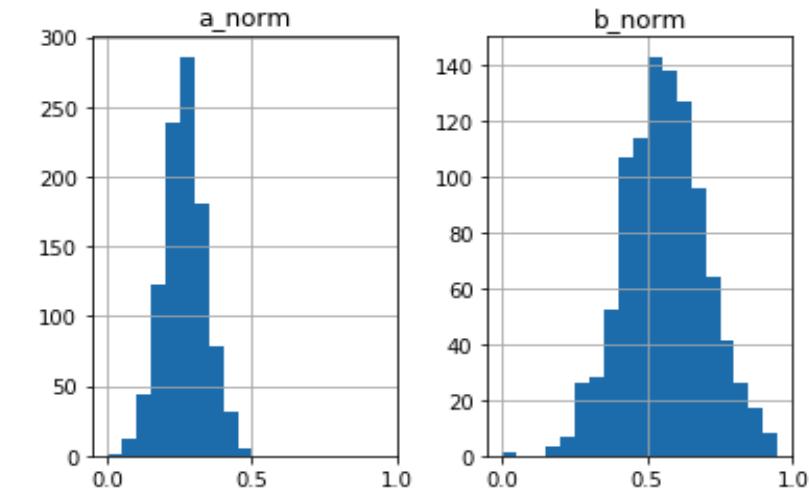
$\sigma$  : 特徴  $x$  の標準偏差

## ■ 使い分けの基準

- 0-1 にスケーリングしたい → 正規化
  - 最大値と最小値が明確に決まっている量的変数
  - 画像データの画素値など
- 0 付近にスケーリングしたい → 標準化
  - その他のテーブルデータに含まれる量的変数



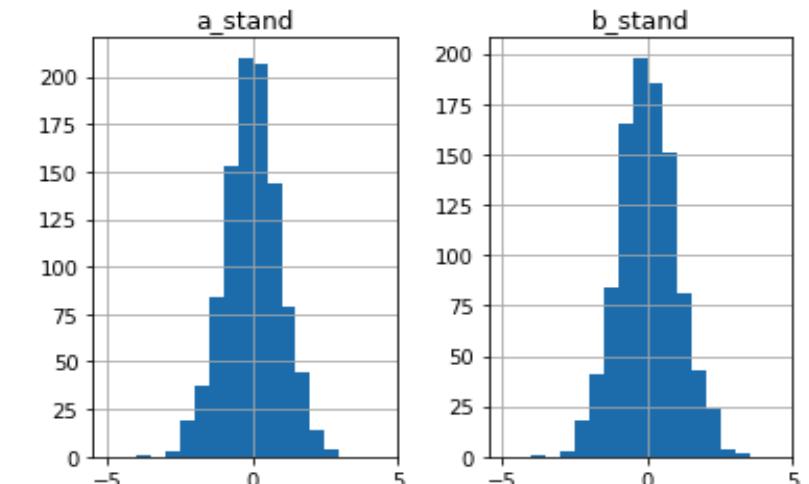
正規化の結果



正規化

標準化

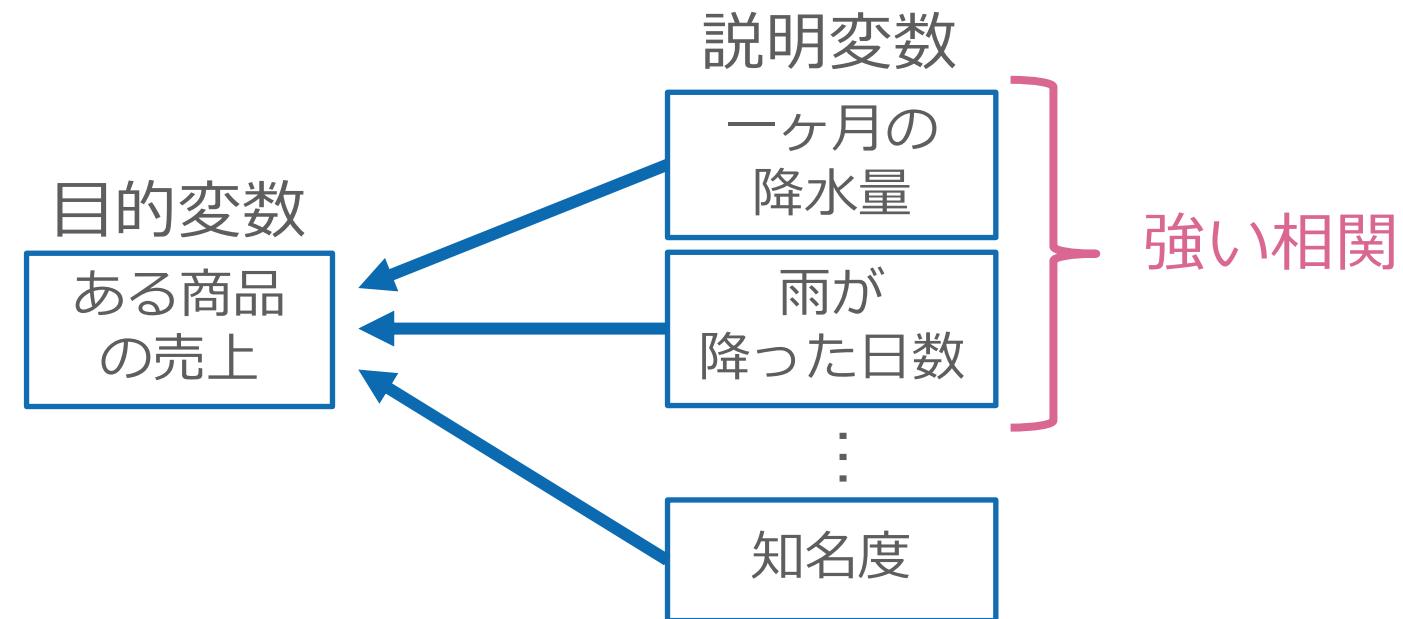
標準化の結果



## **無相関化と白色化**

---

- 線形回帰モデルを用いて、以下の特徴からある商品の売上を予測したい
- 特徴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
- 特徴の間に強い相関があることは何を意味するだろうか？学習への影響は？



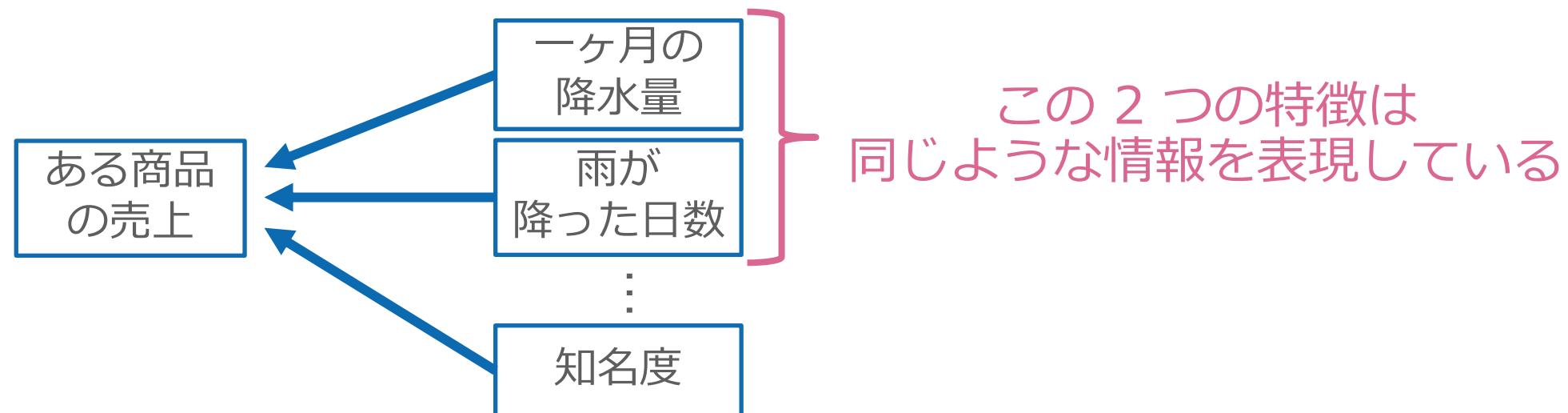
## ■ 特徴の間に強い相関があると、学習が上手くいかないことがある

- この現象のことを多重共線性という
- 相関係数が高い → 同じような情報を持った特徴がある

## ■ 係数の解釈にも信用がなくなる

- 2つの特徴のうち片方に重みに正の値、もう片方に負の値がついた場合、各特徴の目的変数への影響度合いをどのように解釈すれば良いか分からなくなる

## ■ そこで、データ全体から相関関係を取り除く前処理を施す



- 特徴間の相関を取り除くこと
- 共分散の値が 0 となるようにデータセット全体を変換
  - ・ 共分散の値が 0 → 相関係数 0

## ■ 手順

- ① データの分散共分散行列  $\Sigma$  を算出
- ② 分散共分散行列を**固有値分解**し、  
固有ベクトルの集まり  $P$  を抽出
- ③  $P$  とデータの積をとる

## ■ 分散共分散行列の固有値分解は、**主成分分析**においても用いられる

- ・ 固有値の大きい順に、 $P$  から一部の固有ベクトルを取り出すことで、次元削減

詳細は「第 14 章：教師なし学習」の  
「主成分分析」の節にて取り扱う

## ■ 無相関化したデータに対して標準化を施すこと

- 必ず無相関化 → 標準化の順番で行う
- 標準化 → 無相関化の順番で実行した場合は、異なる結果が得られる

## ■ 無相関化後のデータの平均、標準偏差をそれぞれ求め、以下の式によって変換

$$x'_i = \frac{x_i - \mu}{\sigma}$$

$x_i$  : 無相関化済みの特徴  $x$  の  $i$  番目

$\mu$  : 無相関化済みの特徴  $x$  の平均

$\sigma$  : 無相関化済みの特徴  $x$  の標準偏差

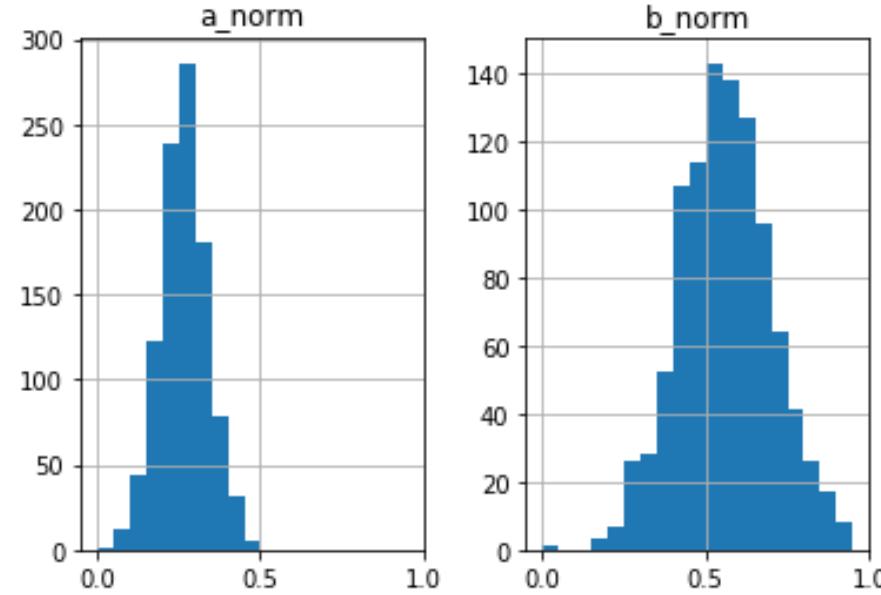
## ノートブック演習

---

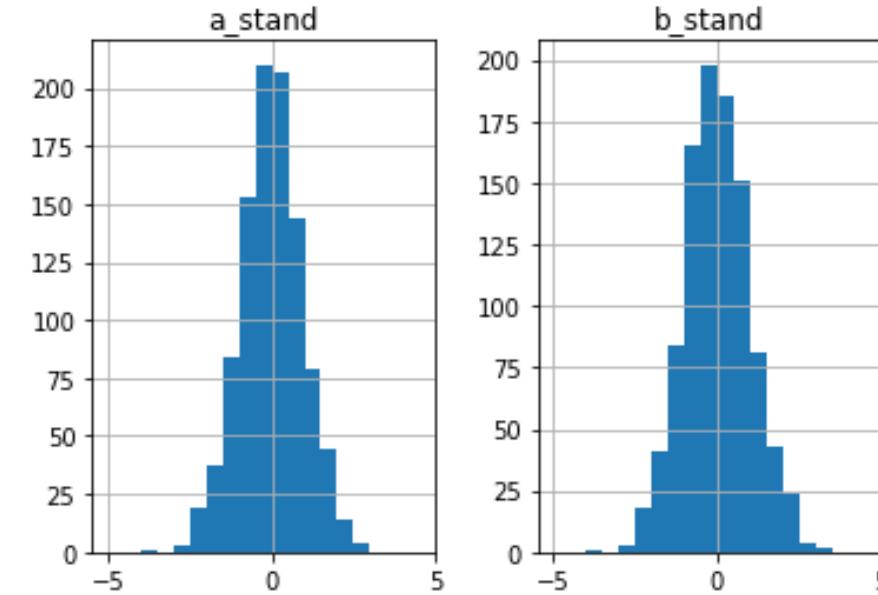
## 6-1\_normalization\_and\_standarization\_trainee.ipynb

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

正規化の結果

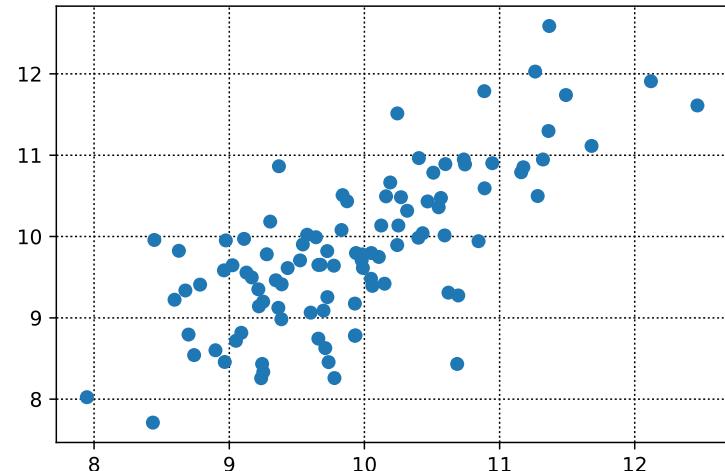


標準化の結果



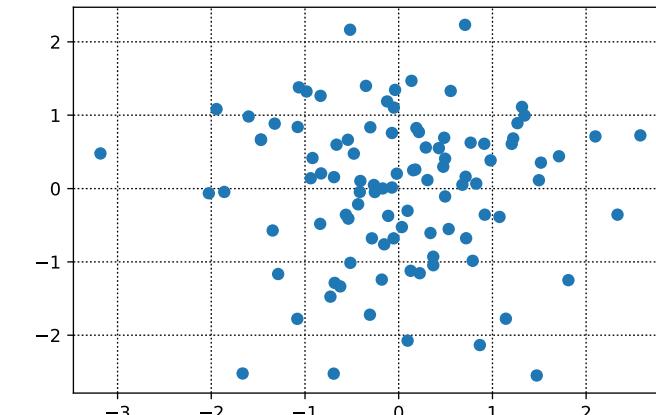
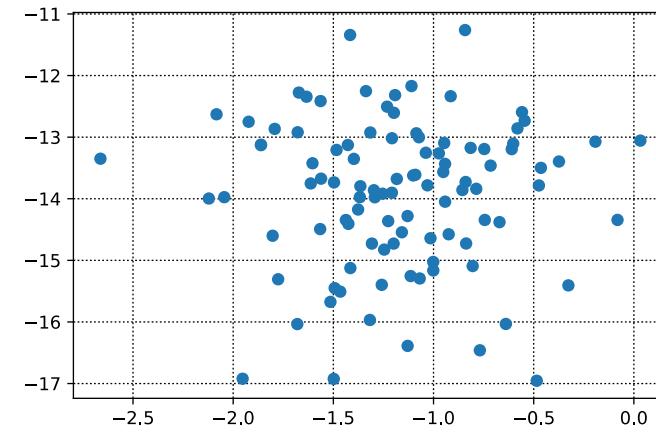
## 6-2\_decorrelation\_and\_whitening.ipynb

- 無相関化と白色化の処理を確認してみよう



無相関化

白色化



## 本章のまとめ

---

- 欠損値処理
- 外れ値・異常値処理
- カテゴリ変数の変換
- 正規化と標準化
- 無相関化と白色化
- ノートブック演習

## ■ 欠損値処理

- ・ 欠損値を含む行または列を除外
- ・ 貴重なデータを有効利用するためになるべく補完を実施

## ■ 外れ値処理

- ・ 四分位範囲 (IQR) を用いた外れ値除去
- ・ ウィンザー化 (外れ値の丸め込み)

## ■ カテゴリ変数の変換

- ・ ワンホットエンコーディング
- ・ ラベルエンコーディング

## ■ 正規化

- 特徴のスケールを 0 以上 1 以下に変換
- 外れ値に弱い

## ■ 標準化

- 特徴を平均 0、標準偏差 1 に変換

## ■ 無相関化

- データ全体から相関関係を除去
- 主成分分析でも同様の効果が得られる

## ■ 白色化

- 無相関化 → 標準化の順番でデータを変換

# 現場で使える 機械学習・データ分析基礎講座

第 7 章：特徴選択

- 次元の呪い
- 特徴選択
- ノートブック演習

- 次元の呪いとは何か説明できるようになる
- 特徴選択の方法を 3 つ列挙し、それぞれ説明できるようになる
  - ラッパー法
  - ステップワイズ法
  - 埋め込み法

# 次元の呪い

---

## ■ 次元の呪い (The curse of dimensionality)

- 高次元空間の様子は人間の直感通りにはならないことを表現した言葉
- 次元が増えると、計算時間が指数関数的に増える
- 機械学習における次元とは、**変数（特徴）の数**

## ■ 次元が大きくなりやすいデータの例

- テキストデータ
- 画像データ

## ■ 文字列の集合として表現されるデータ

- 特徴量の例) 文章における単語の出現回数（頻度）を集計して特徴とする

	単語										$m$ : 単語の個数
	id	今日	は	電車	で	会社	に	行く	…	$m$	
文章	1	1	3	1	2	2	1	1	…	0	
	2	0	2	1	0	3	0	2	…	4	
	3	1	2	0	5	0	0	1	…	0	
	4	0	0	3	0	1	2	1	…	3	
	…	…	…	…	…	…	…	…	…	…	
	$n$	0	2	1	3	0	0	0	…	4	

各文章の次元数（特徴の個数）は  $m$

文章は高次元データになりやすく、  
 文章を特徴として用いると計算負荷が高くなりやすい

- 画素と呼ばれる数値を行列形式でまとめたもの
- 画像の解像度が上がるほど、画素数が増えるため、**高次元データ**になりやすい
  - $8 \times 10$  ピクセルの小さな画像でも、ベクトルに引き伸ばせば 80 次元
  - フルHD画質の画像であれば、 $1920 \times 1080$  次元



10pixel

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	1	1	1	0	0	0	0
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0
0	0	0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0

黒い部分を0、白い部分を1と表現

- 次元の呪いによって計算時間が増大
- 人間にとて解釈が難しい
  - ・ 一体どれが重要な特徴なのか、特定するのが大変
- モデルにとて学習が難しい
  - ・ 次元が大きいほど、過学習を引き起こしやすくなる
  - ・ 過学習を防ぐには、より大量のデータを用意しなければならない

特徴の数を少なくすることはできないか？

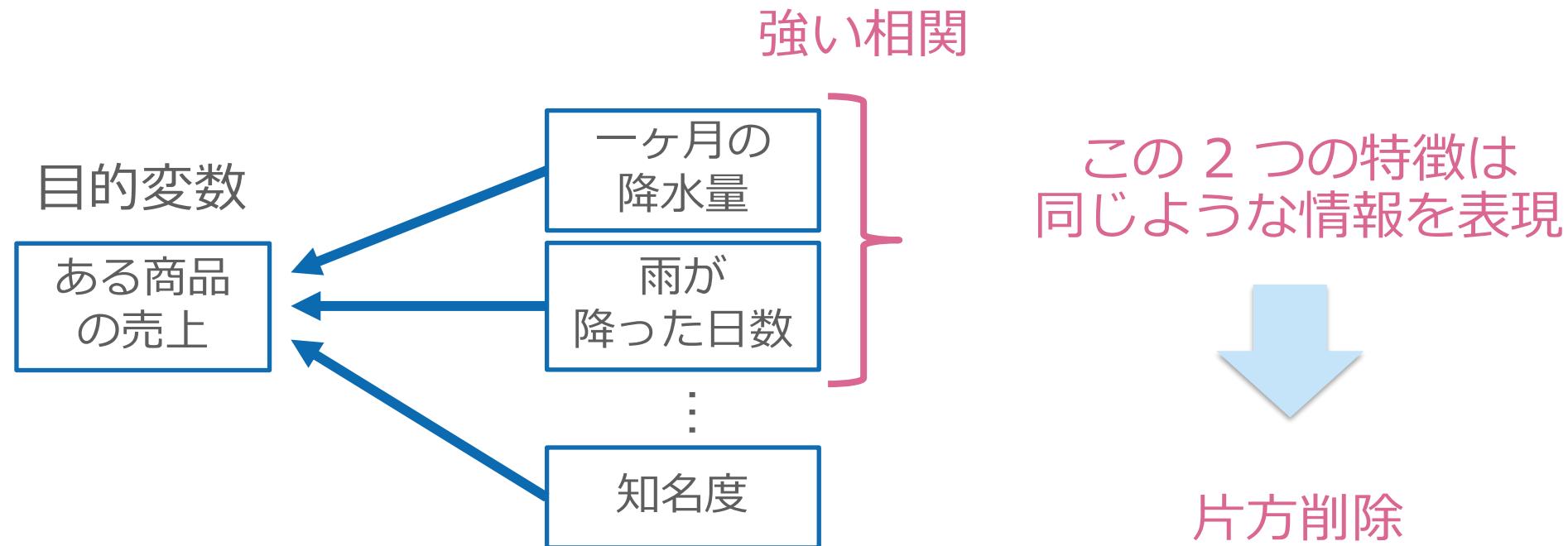
## 特徴選択

---

- 特徴の良し悪しを決める基準とアルゴリズムによって、学習に用いる特徴の数を削減すること
- 特徴選択は主に 3 つのアプローチに分けられる
  - 詳しくはこちらを参照：[An Introduction to Variable and Feature Selection](#)

	フィルタ法	ラッパー法	埋め込み法
計算時間	◎	X	○
予測精度	△	◎	○
	相関などの統計量を使って選択する  一般的に高速だが、精度の点で難がある	モデルの学習と特徴選択を何度も繰り返し、ベストな組み合わせを見つける  時間はかかるが精度は高い	モデルの学習と同時に使用する特徴を学習する  計算時間と精度のバランスがよい

- 統計量を用いて不要な変数を見つけることで、特徴選択を行う
- 例) 相関係数が高い特徴の組を見つけ、片方を削除する
  - ・ 相関係数が高い場合、一方の特徴のみで事象を説明できることが多いため削除してもよい



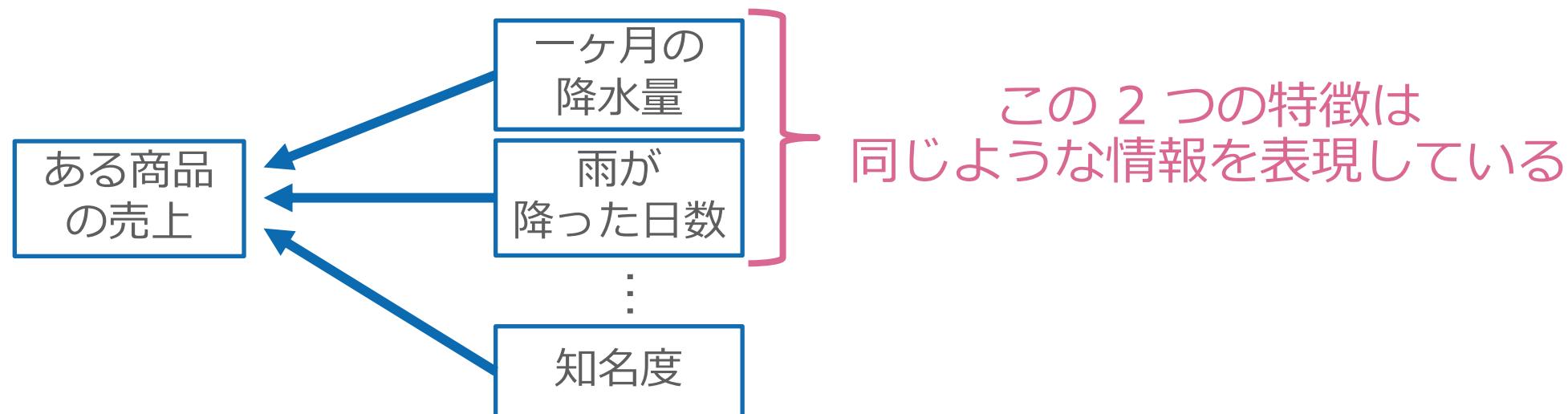
## ■ 特徴の間に強い相関があると、学習が上手くいかないことがある

- この現象のことを**多重共線性**という
- 相関係数が高い → 同じような情報を持った特徴がある

## ■ 係数の解釈にも信用がなくなる

- 2つの特徴のうち片方に重みに正の値、もう片方に負の値がついた場合、各特徴の目的変数への**影響度合い**をどのように解釈すれば良いか分からなくなる

## ■ そこで、データ全体から相関関係を取り除く前処理を施す



- 多重共線性への対処としても、**フィルタ法**が用いられる
- 利用できる統計量の例
  - 相関係数
    - 変数間の相関の強さを表す統計量
    - 相関行列を表示し、値が大きい（0.8 以上など）説明変数の一方を削除
  - 分散拡大係数 (Variance Inflation Factor; VIF)
    - 回帰係数の分散が、多重共線性のためにどれだけ増加したかを測る指標
    - 値が 10 を超える説明変数の一方を削除

$$VIF_i = \frac{1}{1 - R_i^2}$$

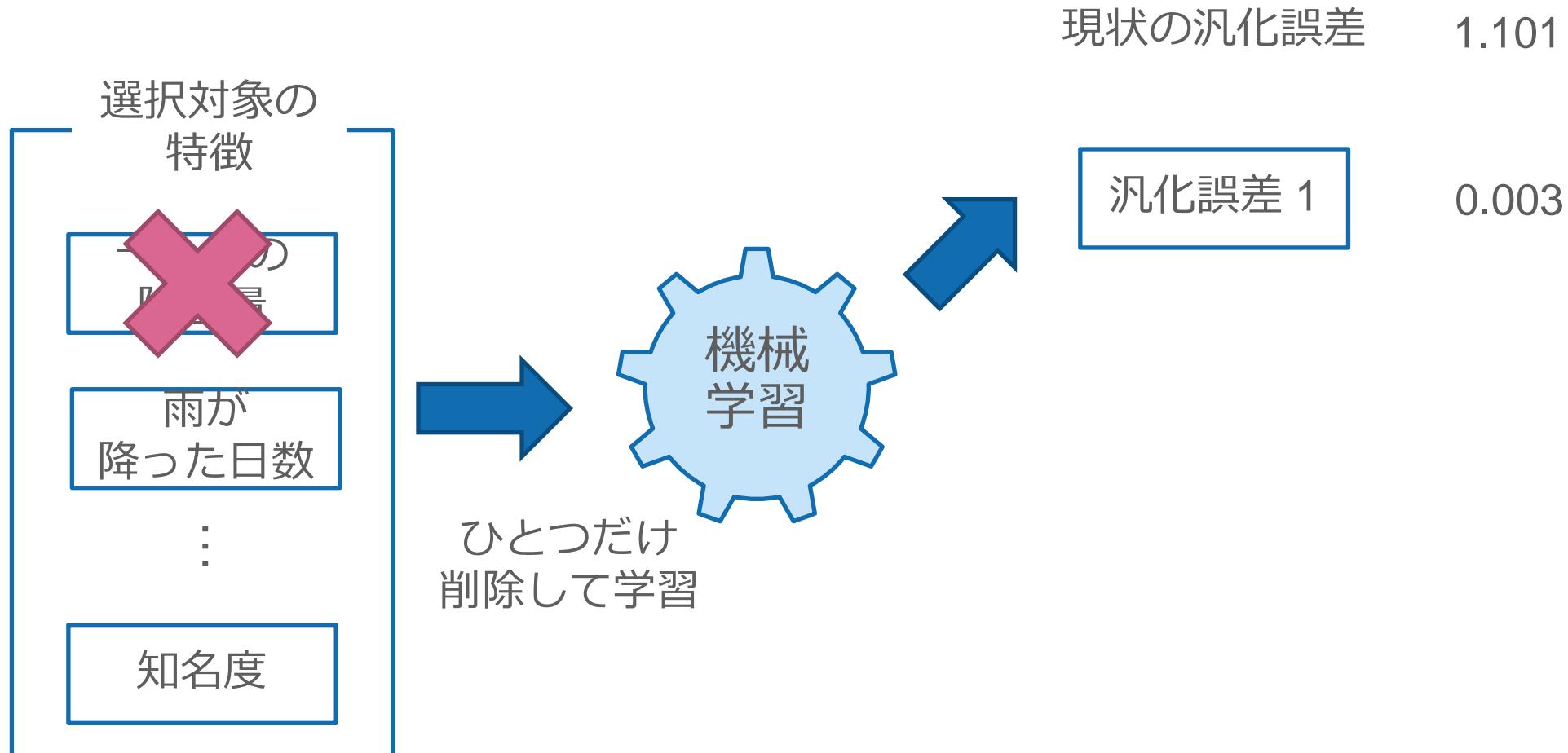
$VIF_i$  説明変数  $x_i$  の VIF

$R_i^2$ :  $x_i$  を目的変数、他の変数（元の目的変数以外）を説明変数として構築した、回帰モデルの決定係数

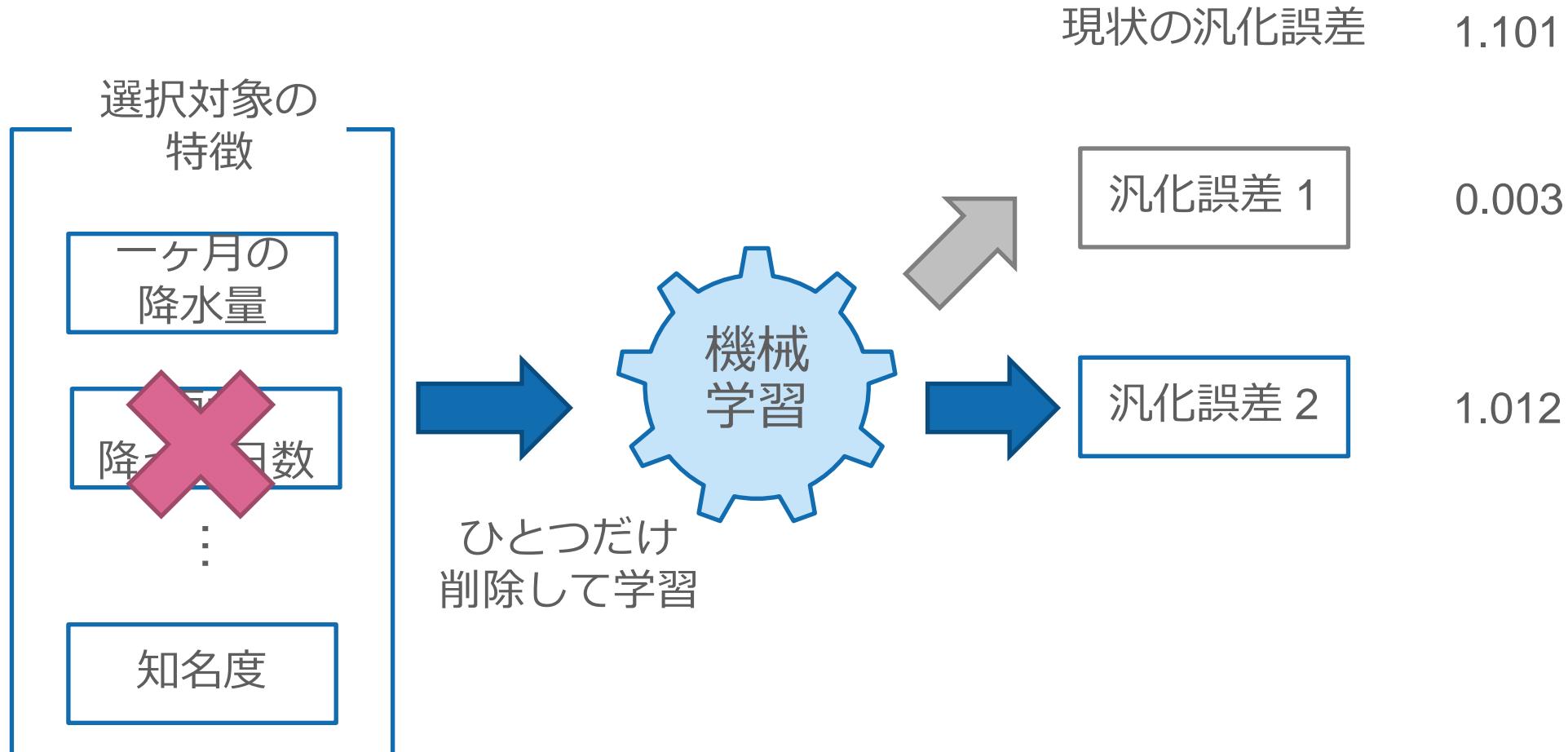
- 特徴の削除（あるいは追加）→モデルの学習→汎化誤差の評価を繰り返すことで特徴選択を行う
- 減少法（ステップワイズ法の 1 つ）
  1. 現状の汎化誤差を評価
  2. 全ての特徴について以下を繰り返し行う
    1. 特徴を 1 つ削除
    2. モデルを学習
    3. 汎化誤差を評価
    4. 特徴を元に戻して、別の特徴を削除
  3. 最も汎化誤差が減少した特徴を削除して、2 に戻る
  4. 汎化誤差の減少量が一定以下になれば終了

汎化誤差を評価する際には、モデルのチューニングと同じようにホールドアウトや交差検証を行う

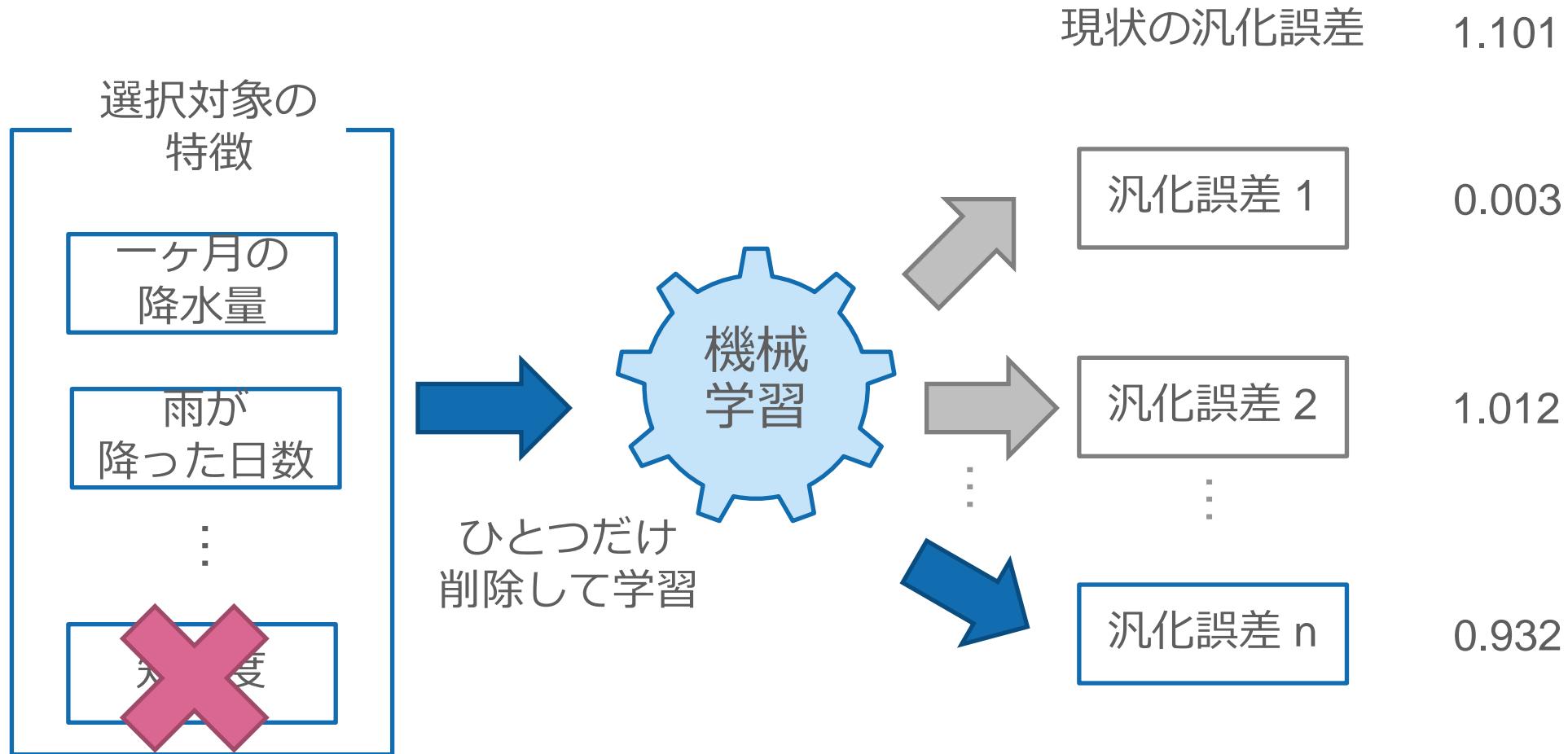
# ラッパー法 | ステップワイズ法



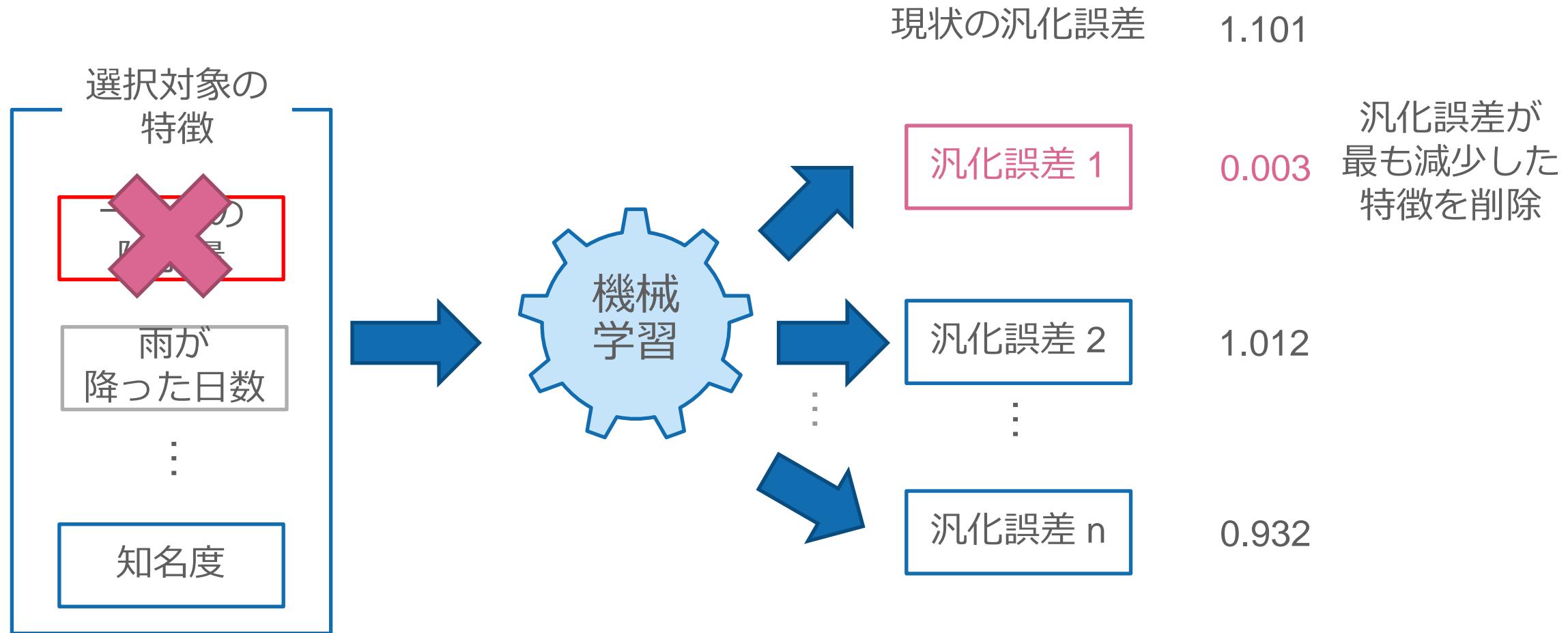
# ラッパー法 | ステップワイズ法

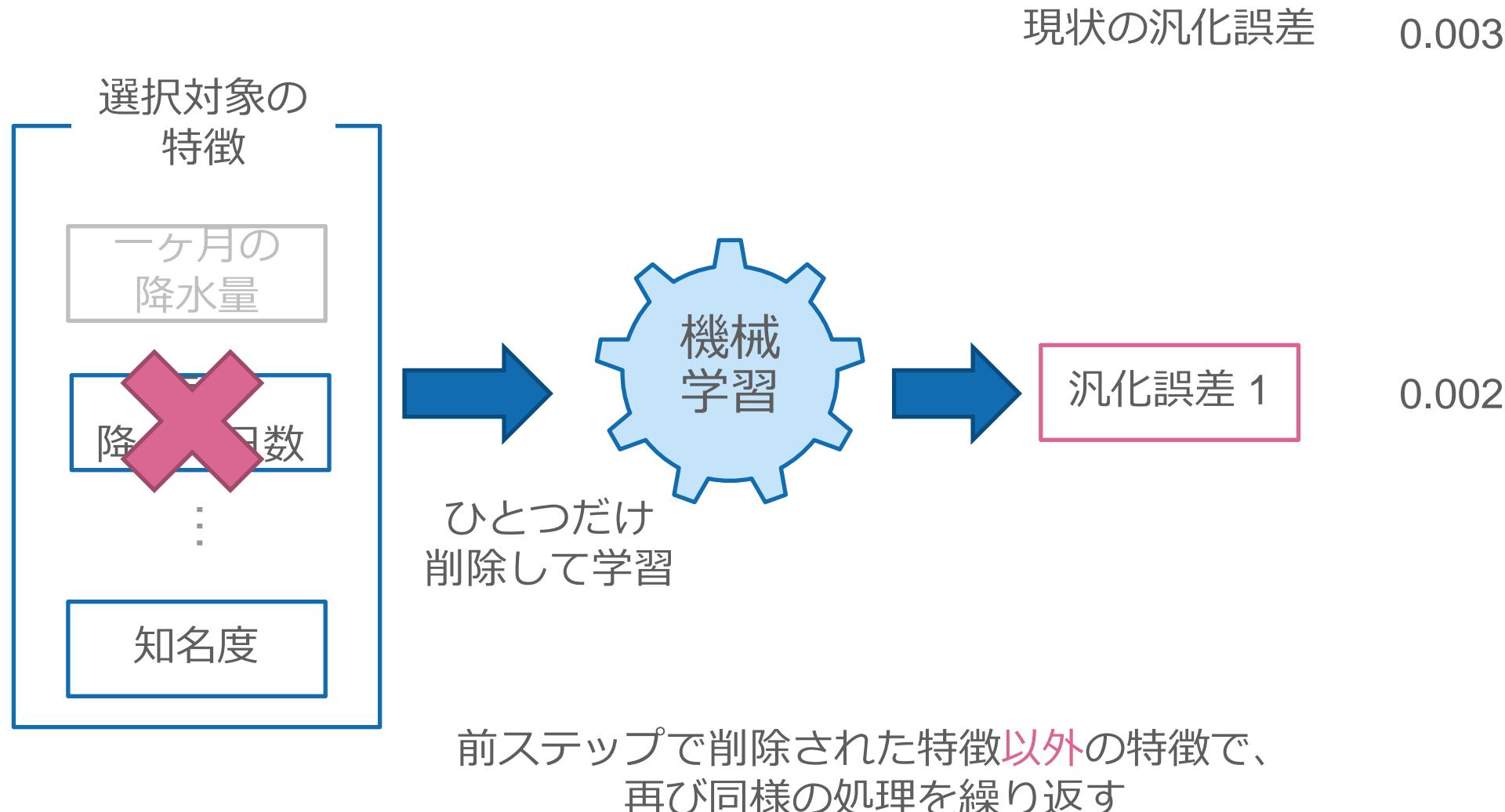


# ラッパー法 | ステップワイズ法

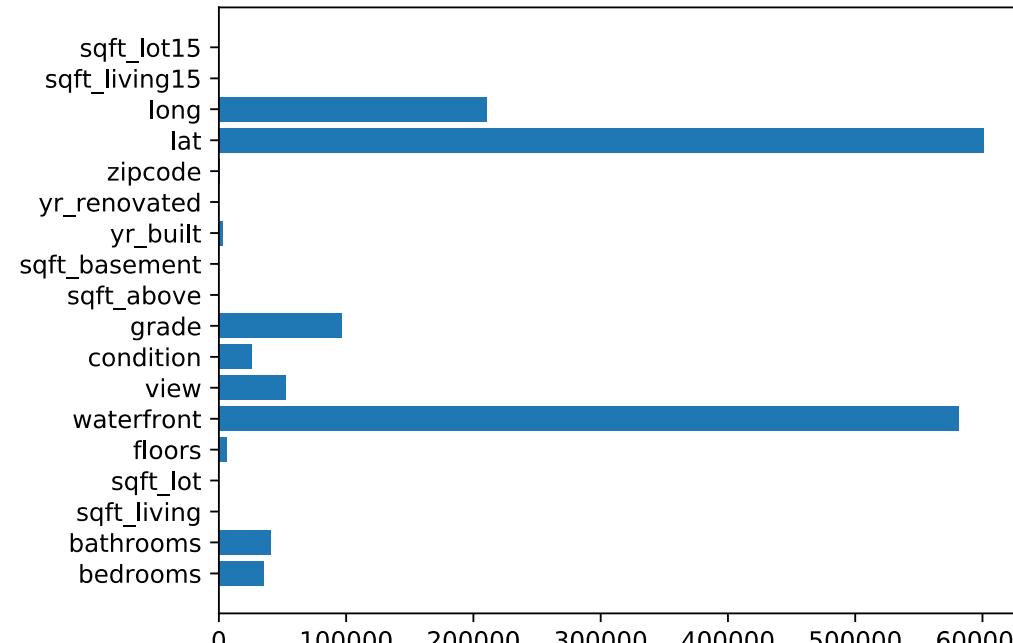


# ラッパー法 | ステップワイズ法





- Lasso は L1 正則化を特徴量選択に応用したもの
  - L1 正則化は、重要でない特徴の係数を 0 にする効果がある
- 係数が小さくなつた特徴を削除することで特徴選択に応用
  - 学習後に係数の大きさをグラフ化する（係数プロット）
  - どの特徴に対する係数が 0 であるのかを確認し、0 であるものを削除



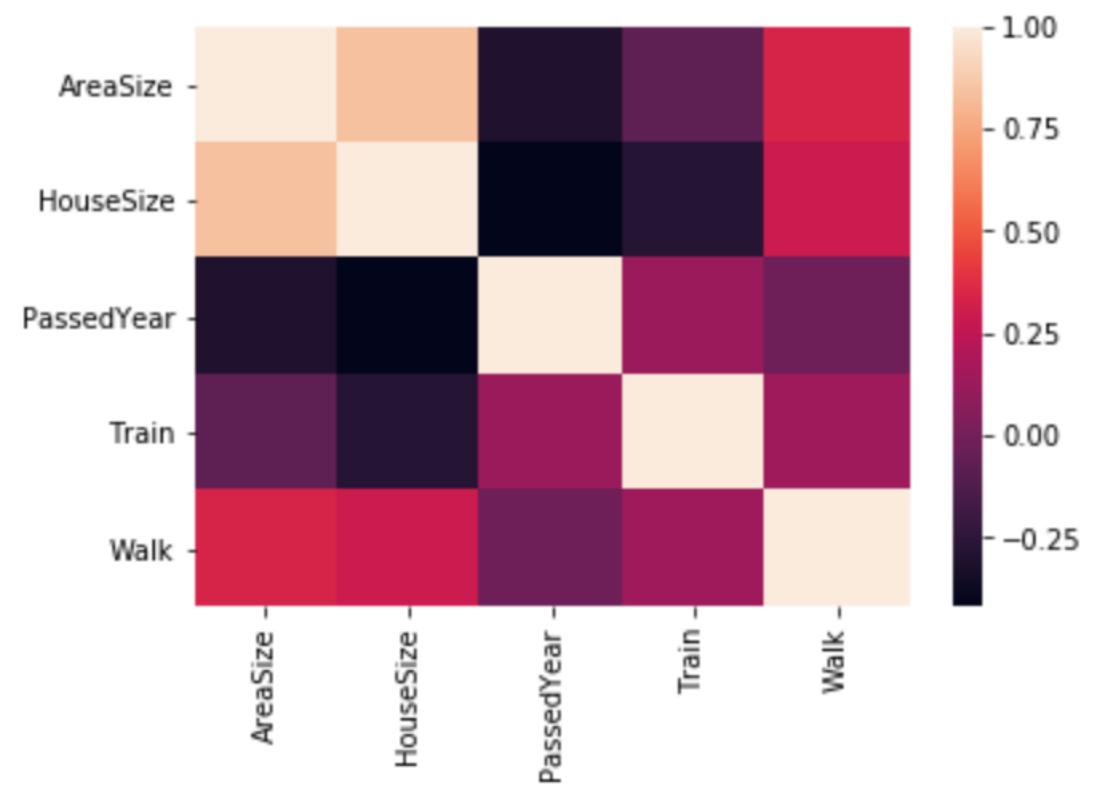
## ノートブック演習

---

## 7-1\_feature\_selection\_filter.ipynb

- 相関係数に基づいて特徴の数を削減してみよう

	AreaSize	HouseSize	PassedYear	Train	Walk
AreaSize	1.000000	0.843471	-0.303278	-0.074319	0.336687
HouseSize	0.843471	1.000000	-0.420226	-0.276636	0.291113
PassedYear	-0.303278	-0.420226	1.000000	0.124133	-0.020027
Train	-0.074319	-0.276636	0.124133	1.000000	0.138155
Walk	0.336687	0.291113	-0.020027	0.138155	1.000000



## 7-2\_feature\_selection\_wrapper.ipynb

### ■ ステップワイズ法による特徴選択を確認しよう

```
# estimatorにモデルをセット
# 今回は回帰問題であるためLinearRegressionを使用
estimator = LinearRegression(normalize=True)

# RFECVは交差検証によってステップワイズ法による特徴選択を行う
# cvにはFold (=グループ) の数, scoringには評価指標を指定する
# 今回は回帰なのでneg_mean_absolute_errorを評価指標に指定 (分類ならaccuracy)
rfecv = RFECV(estimator, cv=10, scoring='neg_mean_absolute_error')
```

```
train_label = df_house["price"]
train_data = df_house.drop("price", axis=1)

y = train_label.values
X = train_data.values

# fitで特徴選択を実行
rfecv.fit(X, y)
```

```
# 特徴のランキングを表示 (1が最も重要な特徴)
print('Feature ranking: \n{}'.format(rfecv.ranking_))
```

```
Feature ranking:
[1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 2]
```

## 7-3\_feature\_selection\_embedded.ipynb

- Lasso による特徴選択を確認しよう
- 係数をプロットし、予測値に寄与している特徴を確認しよう

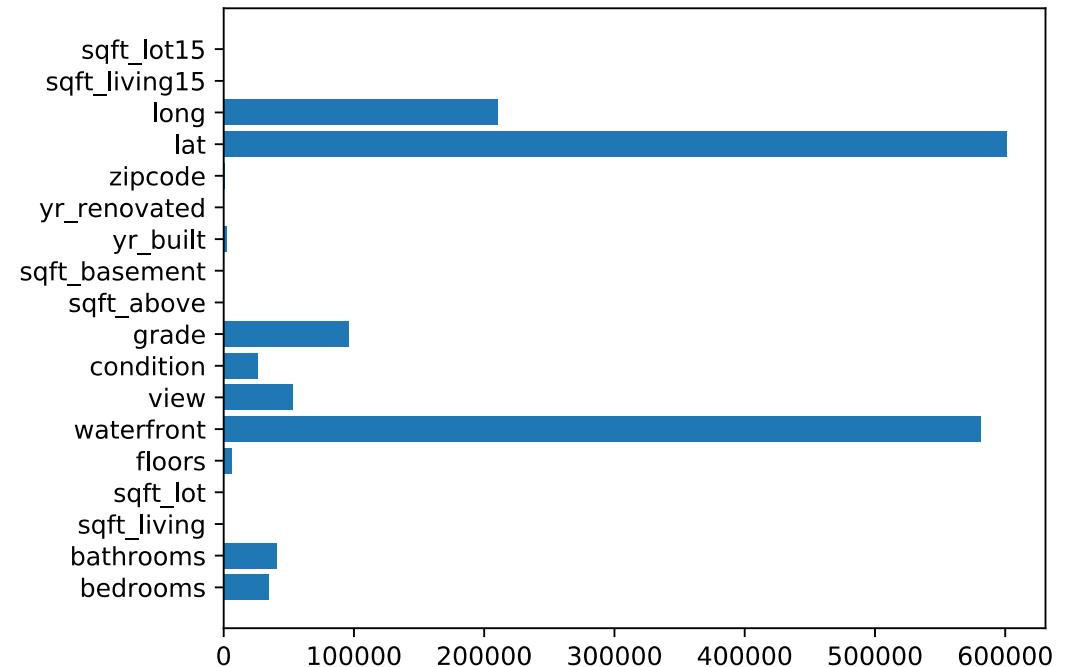
```
# estimatorにモデルをセット
# LassoCVを使って、正則化の強さは自動決定
estimator = LassoCV(normalize=True, cv=10)

# モデルの情報を使って特徴選択を行う場合は、SelectFromModelを使う
# 今回は係数が1e-5以下である特徴を削除する
# 係数のしきい値はthresholdで指定する
sfm = SelectFromModel(estimator, threshold=1e-5)

train_label = df_house["price"]
train_data = df_house.drop("price", axis=1)

y = train_label.values
X = train_data.values

# fitで特徴選択を実行
sfm.fit(X, y)
```



## 本章のまとめ

---

- 次元の呪い
- 特徴選択
- ノートブック演習

- 次元の呪い
  - データの次元数が多いと、モデルの学習が困難になる
- 特徴選択
  - フィルタ法：相関係数などの統計量を元に、モデルの学習前に特徴を削減
  - ラッパー法：モデルの学習と特徴選択を繰り返すことで、特徴を削減
  - 埋め込み法：モデルの学習と同時に、特徴を削減

# 現場で使える 機械学習・データ分析基礎講座

第 8 章：決定木

- 決定木
- 不純度の算出
- ノートブック演習

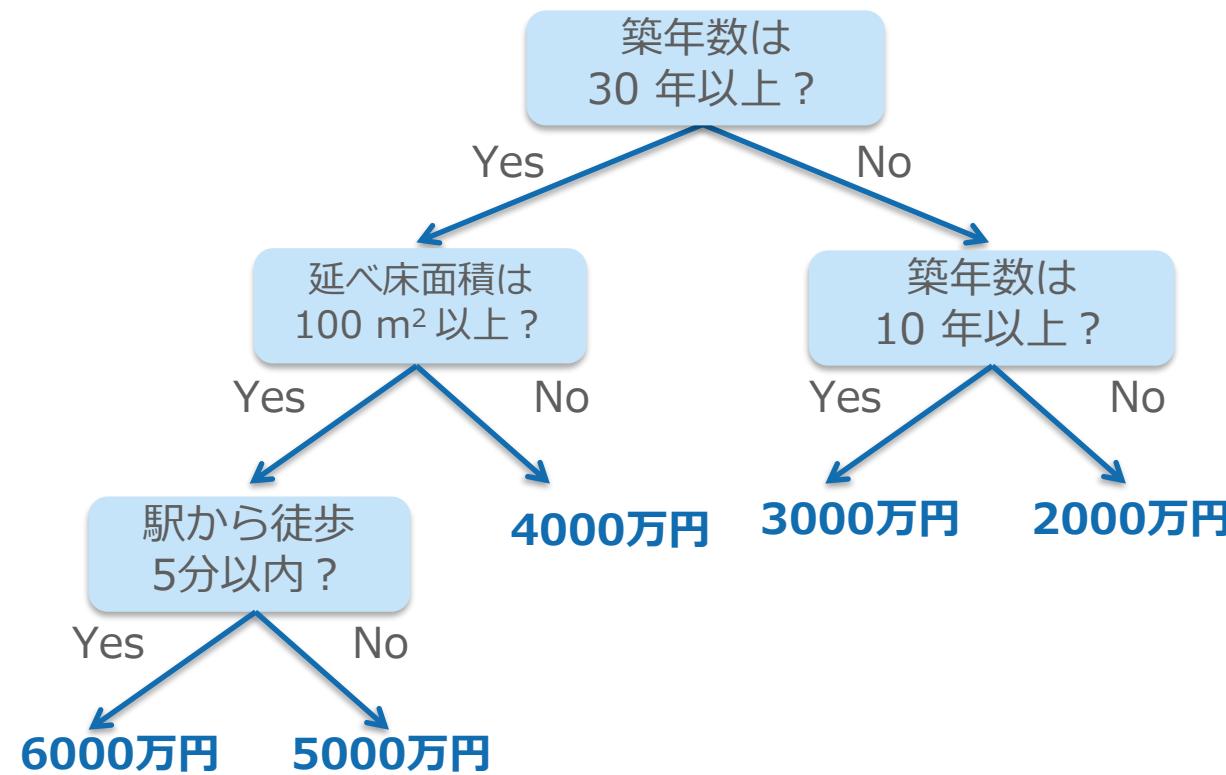
- 情報利得という単語を用いて決定木の成長を説明できるようになる
- 不純度を計算する手法を 3 つ列挙し説明できるようになる

# 決定木

---

- 住宅価格の見積もりを例に考えてみよう
- 人が行う場合は？
  - 知識を持った専門家が経験と知識を組み合わせて見積もり価格を算出
  - 今もこのように専門家による判断を用いる場面は多いだろう
- この専門家の判断プロセスはどのように構成されているだろうか？
  - 自分で判断した場面を思い出してみよう
  - どのように判断しただろうか？

- 住宅価格の見積もりであれば、以下のようなプロセスが考えられる
- いくつかのルール（条件分岐）を組み合わせて判断した場面は多いはず！



左のような構造は  
木構造（ツリー）と呼ばれる

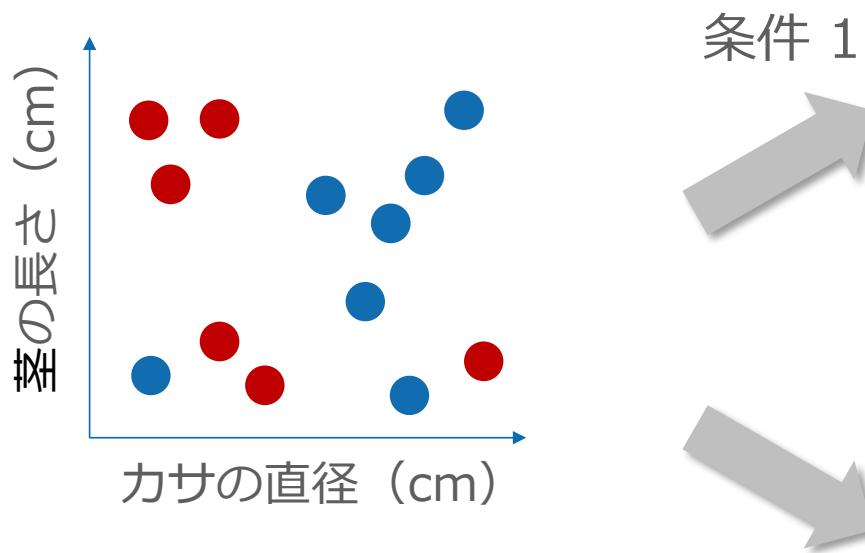
この条件分岐を正しく作れば、  
新しいデータに対しても  
住宅価格の見積もり（予測）が可能！

決定木というアルゴリズムでは  
条件分岐の木を自動作成

- データより条件分岐の木を自動構築するアルゴリズム
  - 基本的には左と右に分かれる二分木を連ねて 1 本の大きな木を構築
- 回帰と分類の両方に応用可能
- 条件分岐を作成する際の基本的な考え方  
→ 似通った目的変数の値を持つデータが集まるように分岐を作る

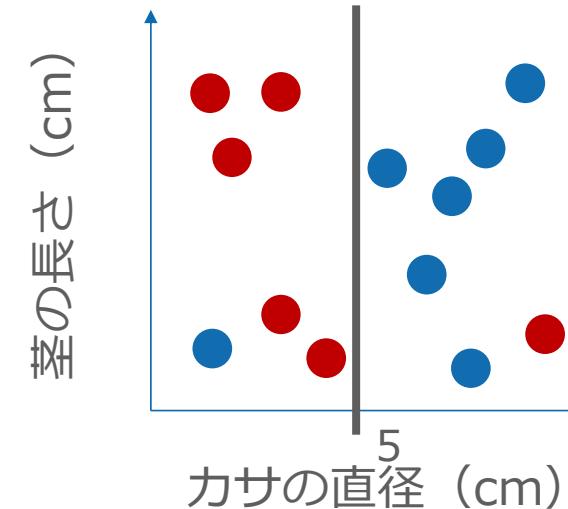
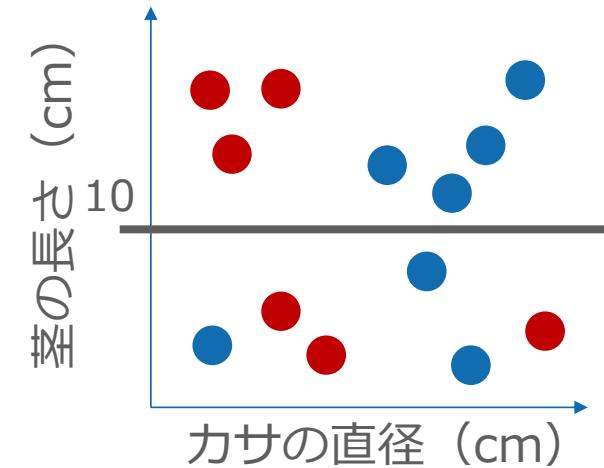
# 決定木 | 条件分岐の作り方（分類）

## ■ 例) 毒キノコの分類



目的変数

- クラス1：毒キノコではない
- クラス2：毒キノコである



条件 2 の方が  
分割後の空間で  
同じクラスのデータが  
より多く集まっている

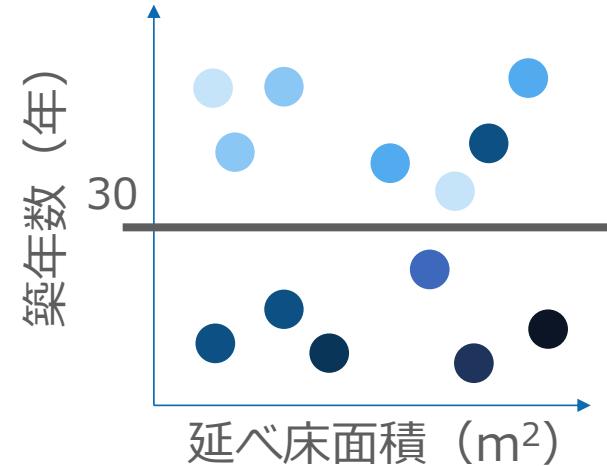
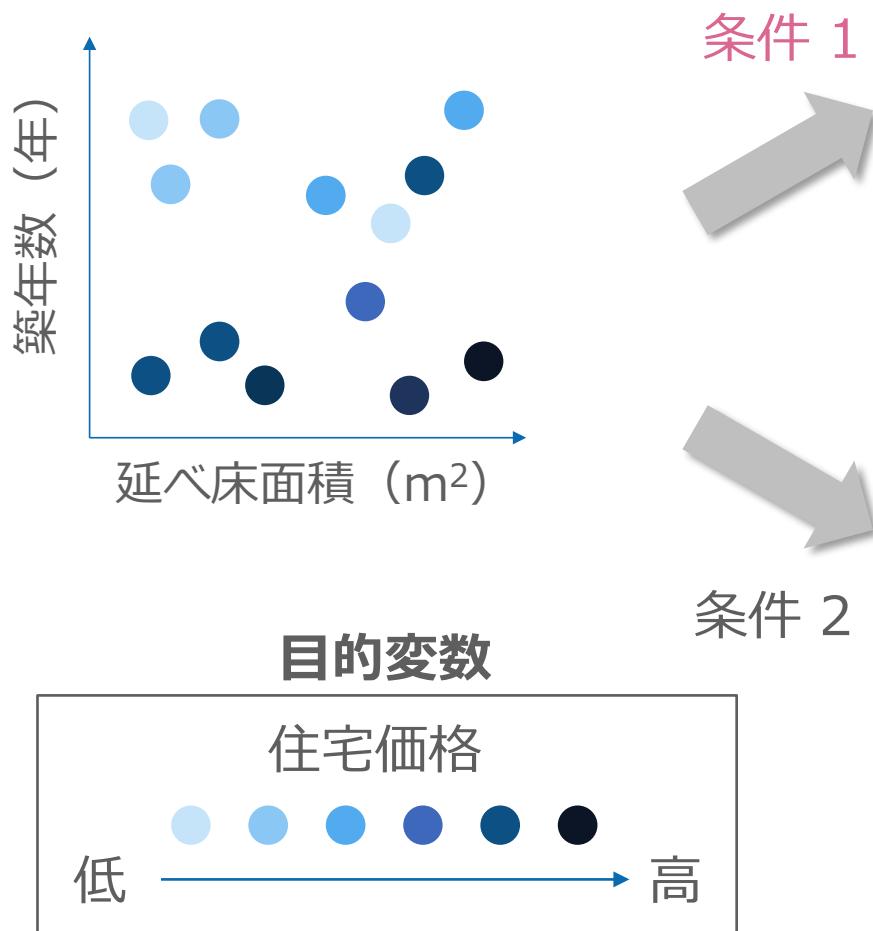
カサの直径は  
5 cm 以上か？

上記の分岐が作成される

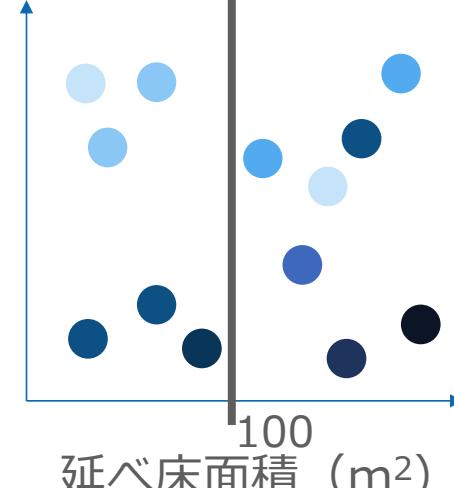
分割後の空間で、再び  
条件分岐を作成

# 決定木 | 条件分岐の作り方（回帰）

## ■ 例) 住宅価格の予測



条件 1 の方が  
分割後の空間で  
目的変数の値が近いデータが  
より多く集まっている

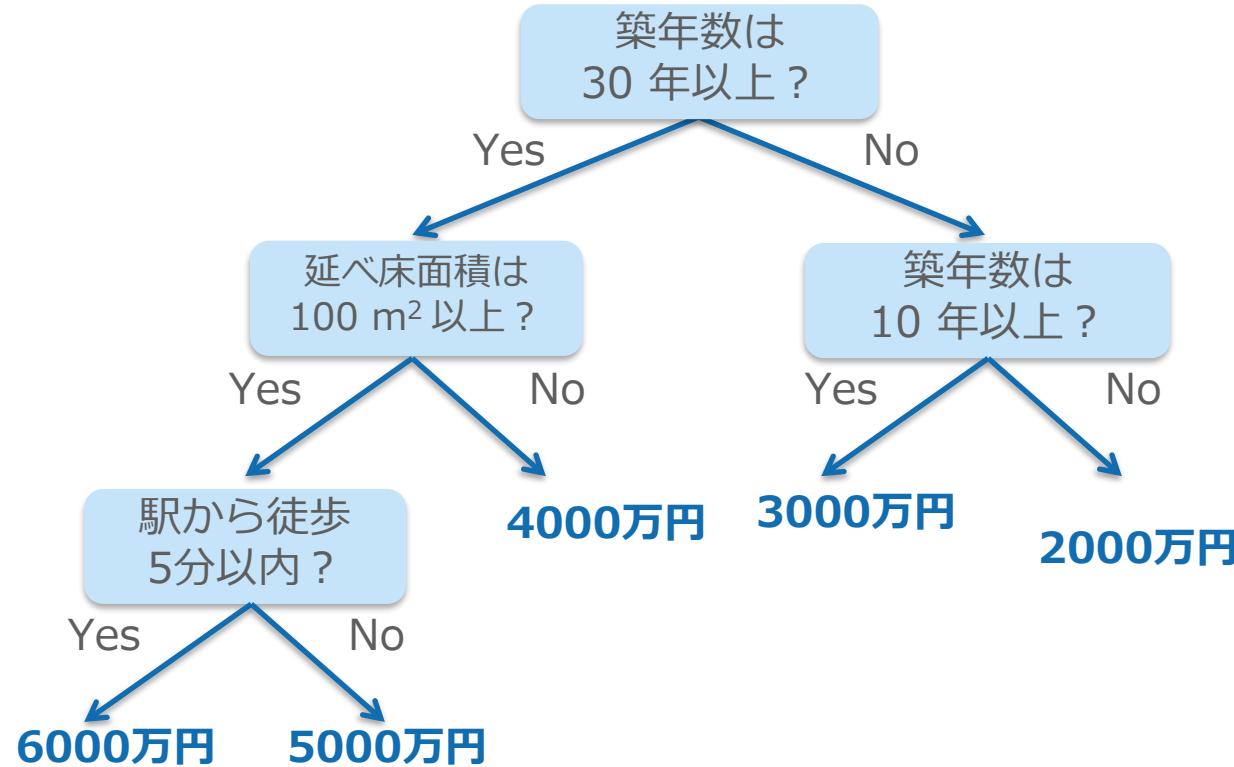


築年数は  
30 年以上か？  
上記の分岐が作成される

分割後の空間で、再び  
条件分岐を作成

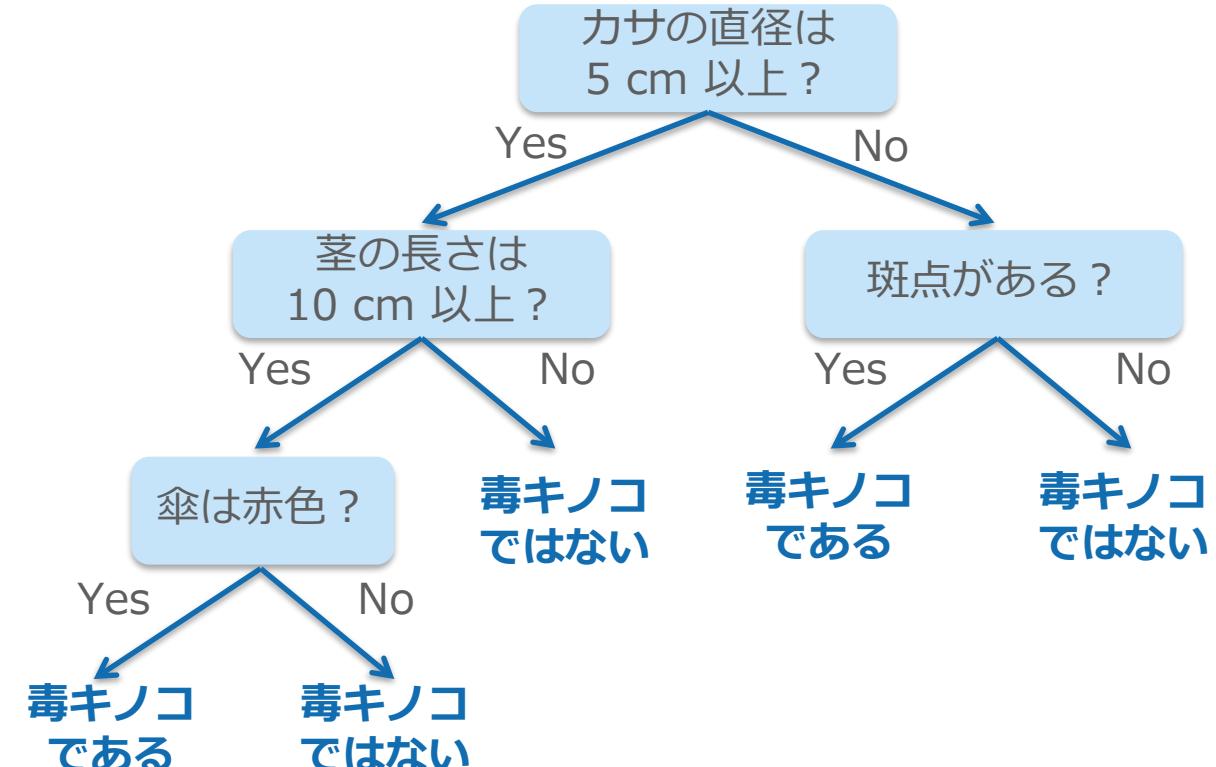
# 決定木 | 構築されたモデルの例

回帰の例



予測値は各空間内のデータの  
目的変数の平均値

分類の例



予測クラスは各空間内のデータの  
多数派クラス

- とある空間における「目的変数の値のばらつき度合い」を定量化できれば、その値を元に条件分岐の良し悪しを把握可能
- 決定木では不純度という考え方で、目的変数の値のばらつきを定量化
  - 不純度 : 0 → その空間に存在するデータは全て同じ目的変数の値を持つ
  - 不純度 : 大 → その空間に存在するデータは多種多様な目的変数の値を持つ
- 「分割前の空間における不純度」と「分割後の 2 つの空間に対する不純度」を比較し、最も大きく不純度を下げる条件を木の成長に利用

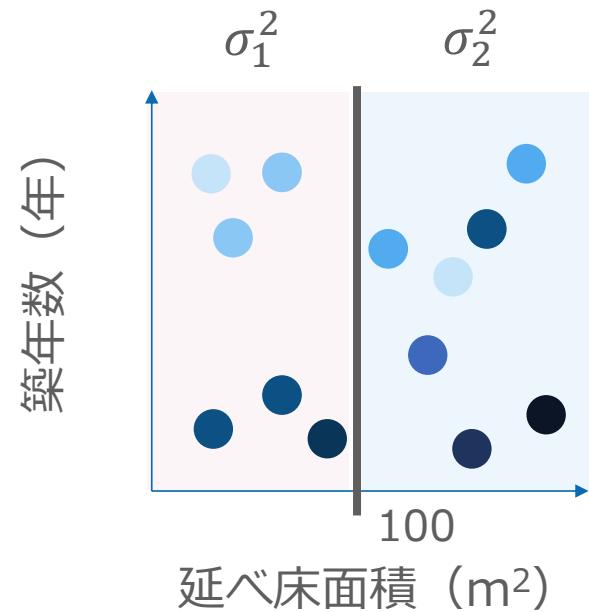
## 不純度の算出

---

- 回帰問題の場合
  - 目的変数の値の分散を用いて算出
- 分類問題の場合
  - ジニ係数（ジニ不純度）もしくはエントロピーを用いて算出

## ■ 各空間においてデータの目的変数の分散を計算

- 分散 0 → 全データの目的変数の値が同じ
- 分散 大 → 目的変数の値が多種多様



- 決定木の不純度を測る指標
- 0 から 1 までの値を取る
  - 0 の場合、空間内のデータのクラスは全て同じ
  - 値が大きいほどクラスが多種多様である

$$\text{Gini Impurity} = \sum_{k=1}^K p(C_k)(1 - p(C_k)) \quad \left( = 1 - \sum_{k=1}^K p(C_k)^2 \right)$$

$K$  : クラス数

$p(C_i)$  : 着目している空間におけるクラス  $C_i$  のデータの割合

= 空間から 1 つデータを抽出した際に  $C_i$  のデータを得る確率

## ■ ある空間におけるクラス $C_k$ のデータの割合 $p(C_k)$ と、ばらつきの関係

- $p(C_k)$  が 0 に近い
  - 空間に、そのクラスのデータがほとんどない (ばらつきが小さい)
- $p(C_k)$  が 1 に近い
  - 空間に、そのクラスのデータで完全に占められている (ばらつきが小さい)
- $p(C_k) = \frac{1}{2}$ 
  - 空間内のデータのうち、半分がクラス  $C_k$  で、残り半分が他のクラスという状態
  - 空間に他のクラスのデータも混ざっており、綺麗な状態ではない (ばらつきが大きい)

## ■ $p(C_k)(1 - p(C_k))$ は $p(C_k)$ が 0 もしくは 1 に近い場合、0 に近い値を取る

- この計算を全クラス  $k = 1, \dots, K$  で行い総和を取ることで、空間全体の不純度を評価

## ■ 事象が発生したことに対する“驚き”的度合いを定量化したもの

$$I(x) = -\log_2 p(x)$$

$I(x)$  : ある事象  $x$  に対する情報量  
 $p(x)$  : 事象  $x$  の発生確率

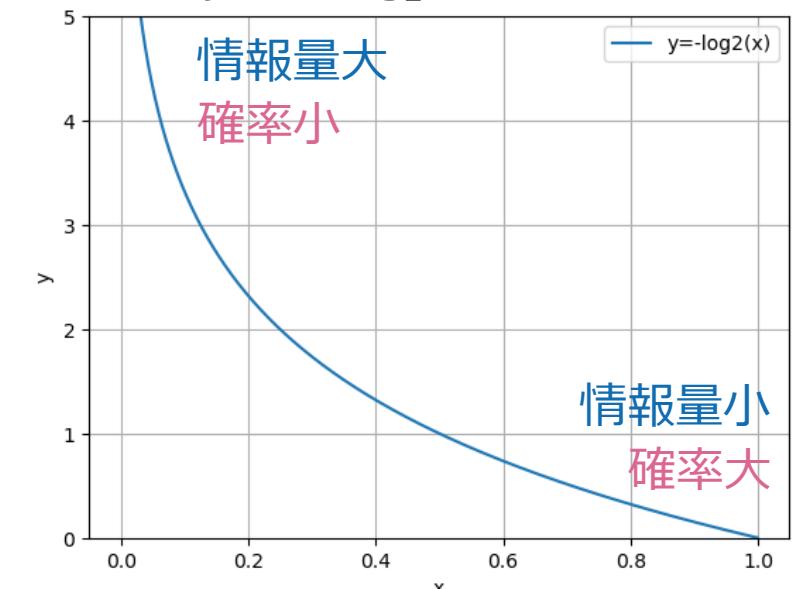
## ■ 情報量の大きいケース

- 「到底起きそうもない事象」が起きたことを知った
- 例) 砂漠に雨が降った

## ■ 情報量の小さいケース

- 「いつでも起きそうな事象」が起きたことを知った
- 例) 東京に雨が降った

$y = -\log_2 x$  のグラフ



- 系（事象の集合）に含まれる事象のばらつき具合を定量化したもの

$$H = - \sum_{k=1}^K p(C_k) \log_2 p(C_k)$$

$K$  : クラス数

$p(C_i)$  : 着目している空間におけるクラス  $C_i$  のデータの割合

= 空間から 1 つデータを抽出した際に  $C_i$  のデータを得る確率

元の定義は、**情報量の期待値**

**ジニ係数**と同じく  
値が大きいほど、空間内の  
クラスが多種多様である

- エントロピーが小さいケース

- ほとんど同じ事象しか起きない
- 例) 砂漠の天気 → ほとんどが晴れ

- エントロピーの大きいケース

- 観測するたびに事象がばらついている
- 例) 東京の天気 → 晴れだったり雨だったりとバラバラ

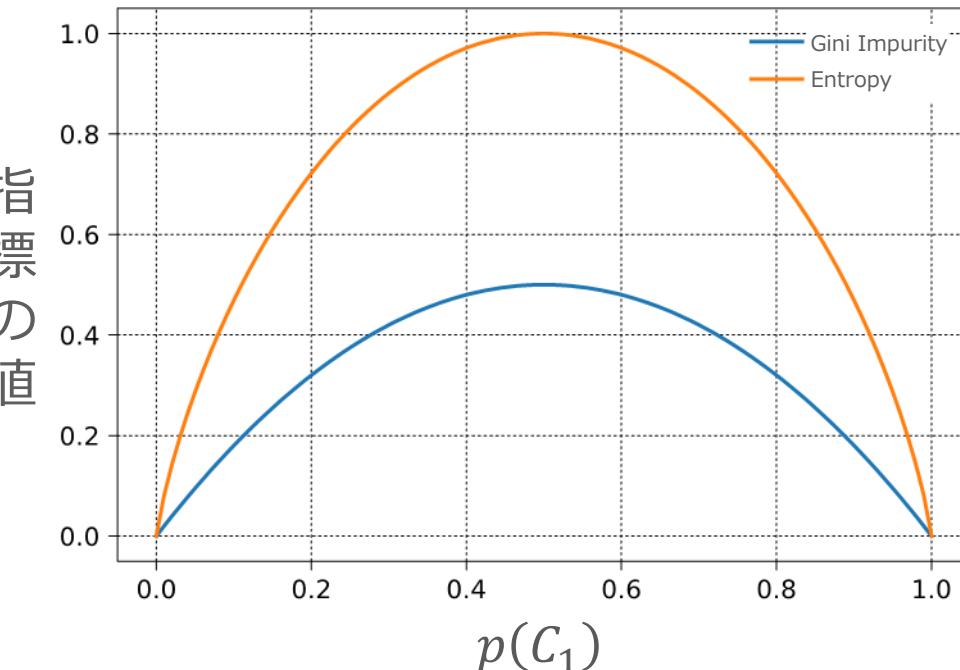
■ 一般的にジニ係数とエントロピーのどちらを用いても非常に良く似た決定木が構築される

- ・ ジニ係数によるモデル構築・エントロピーによるモデル構築の両方を行う必要はない
- ・ どちらかに決め打ちして「木の深さ」といったハイパーパラメータをチューニングする方が良い

$$\text{Gini Impurity} = \sum_{k=1}^2 p(C_k)(1 - p(C_k))$$

$$\text{Entropy} = - \sum_{k=1}^2 p(C_k) \log_2 p(C_k)$$

$K = 2$  における各指標のグラフ



- 空間における不純度を算出ができるようになった
- あとは、分割前後でどれだけ不純度が減ったのか？を算出できれば良い
  - 大量の条件に対して不純度の減少量を算出して、最も減少量の大きな条件を木の成長に利用

## ■ 不純度の減少量は下記の計算で行う

- この減少量を**情報利得**と呼ぶ
- 決定木は情報利得を最大にする条件分岐を探索するアルゴリズムと言える

$$\underline{I(D_p)} - \left( \frac{\text{重み}}{\text{分割前}} \frac{N_{D_{c1}}}{N_{D_p}} I(D_{c1}) + \frac{\text{重み}}{\text{分割後}} \frac{N_{D_{c2}}}{N_{D_p}} I(D_{c2}) \right)$$

分割前

分割後

$D_p$  : 分割前の空間に存在するデータセット

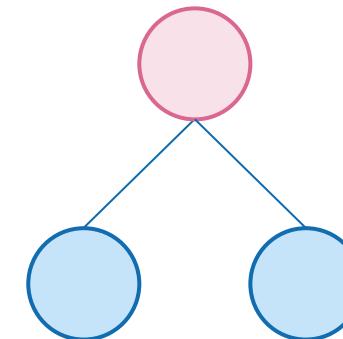
$D_{c1}$  : 分割後の空間に存在するデータセット1

$D_{c2}$  : 分割後の空間に存在するデータセット2

(  $p$  は parent、 $c$  は child の意 )

$N_*$  : データセット \* のデータ総数

$I(*)$  : データセット \* に対する不純度



親 (parent) ノード



子 (child) ノード

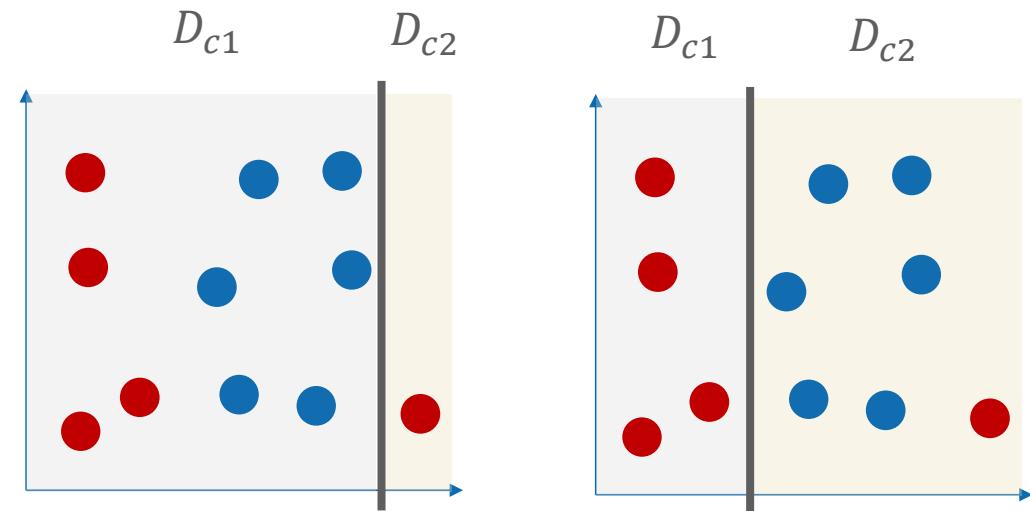
- $\frac{N_{Dc1}}{N_{Dp}}$  と  $\frac{N_{Dc2}}{N_{Dp}}$  は 分割後の各空間の不純度に対する重み

- 分割後の空間内のデータ数に応じて、大きくなる

- 重みを導入する意味

- 右図のような、良い分岐条件を探索しやすくなる
- 左図では  $D_{c2}$  に対する不純度は 0 であるが、データ 1 点のみに着目している
  - データに過剰適合している状態と言える

$D_p$  : 分割前の空間に存在するデータセット  
 $D_{c1}$  : 分割後の空間に存在するデータセット 1  
 $D_{c2}$  : 分割後の空間に存在するデータセット 2



データ 1 点のみを分けるような分割よりも  
たくさんのデータが含まれており、かつ  
不純度が小さくなる分割を採用した方が  
モデルの汎化性能が上がりやすい

## ■ 決定木

- 似通った目的変数の値を持つデータが集まるような条件分岐を発見する手法
- 不純度を計算し情報利得を最大にするような条件分岐によって木を成長させる

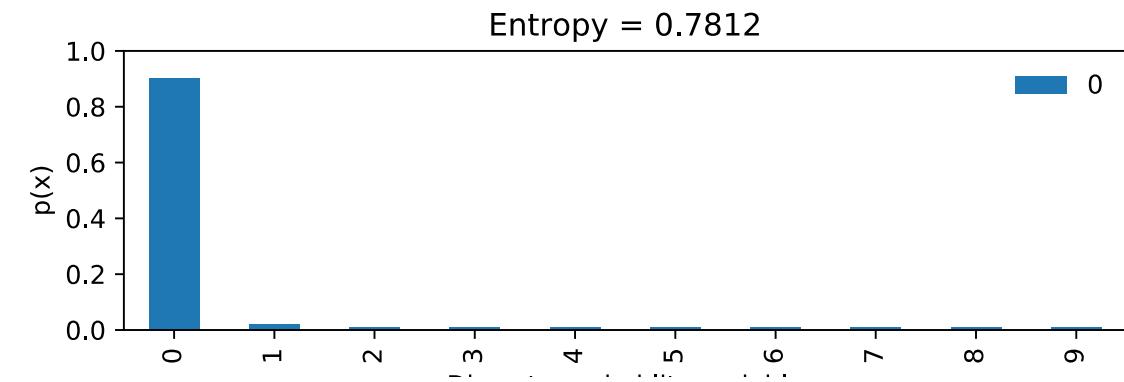
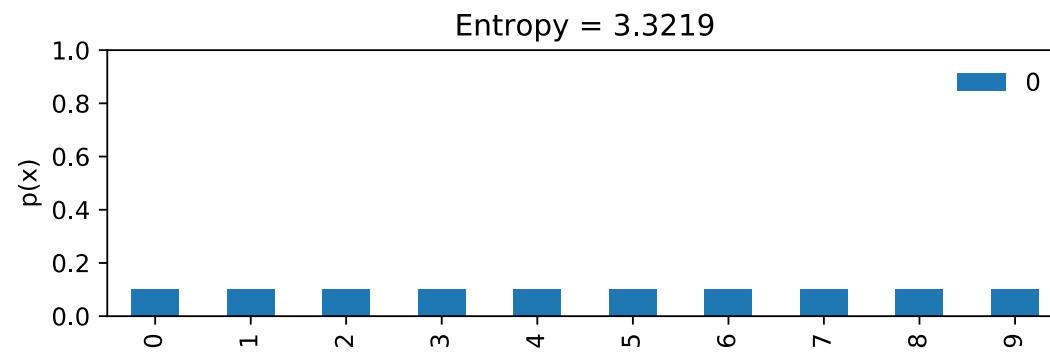


## ノートブック演習

---

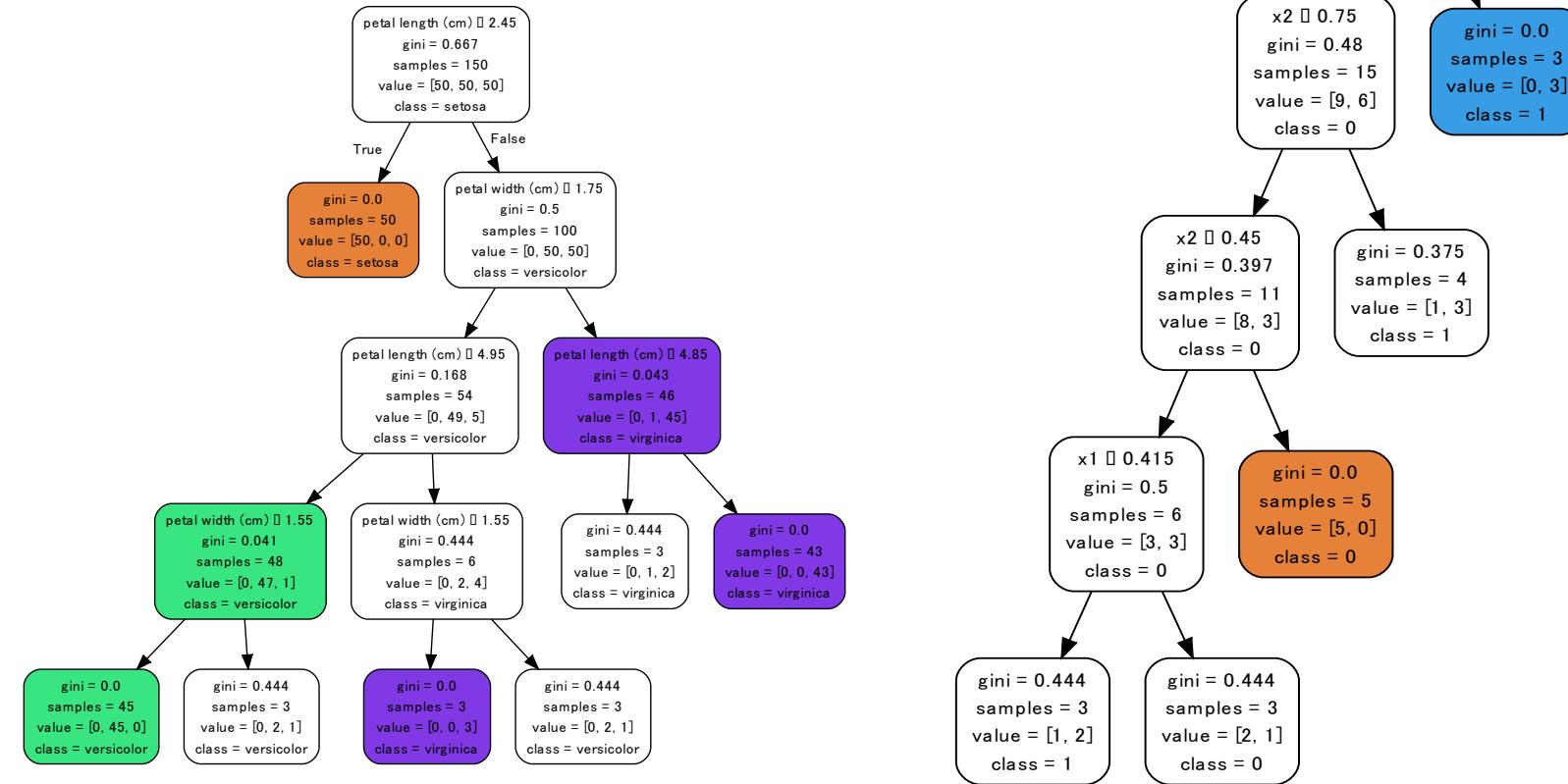
## 8-1\_entropy\_and\_gini.ipynb

- 情報量やエントロピーを実際の数値を用いて計算してみよう
- 2 クラス、多クラス問題におけるエントロピーやジニ係数の変化を確認してみよう



## 8-2\_descision\_tree.ipynb

- scikit-learn による決定木の実装を確認しよう
- 決定木を可視化し、どのような条件分岐が作られたかを見てみよう



## 本章のまとめ

---

- 決定木
- 不純度の算出
- ノートブック演習

## ■ 決定木

- 木構造を応用した機械学習モデル

## ■ 不純度

- 目的変数の値のばらつき度合いを表したもの
- 分類問題の場合、ジニ係数またはエントロピーを用いて算出

## ■ 情報利得

- 分割によって減少した不純度の量

## ■ 決定木の最適化

- 情報利得が最大となるような条件を探索

# 現場で使える 機械学習・データ分析基礎講座

第9章：アンサンブル学習

- アンサンブル学習
- ランダムフォレスト
- アダブースト
- 勾配ブースティング
- ノートブック演習

- アンサンブル学習の代表的手法を 3 つ列挙し、それぞれ説明できる
- ランダムフォレスト、アダブースト、勾配ブースティングの学習の仕組みを説明できる

# アンサンブル学習

---

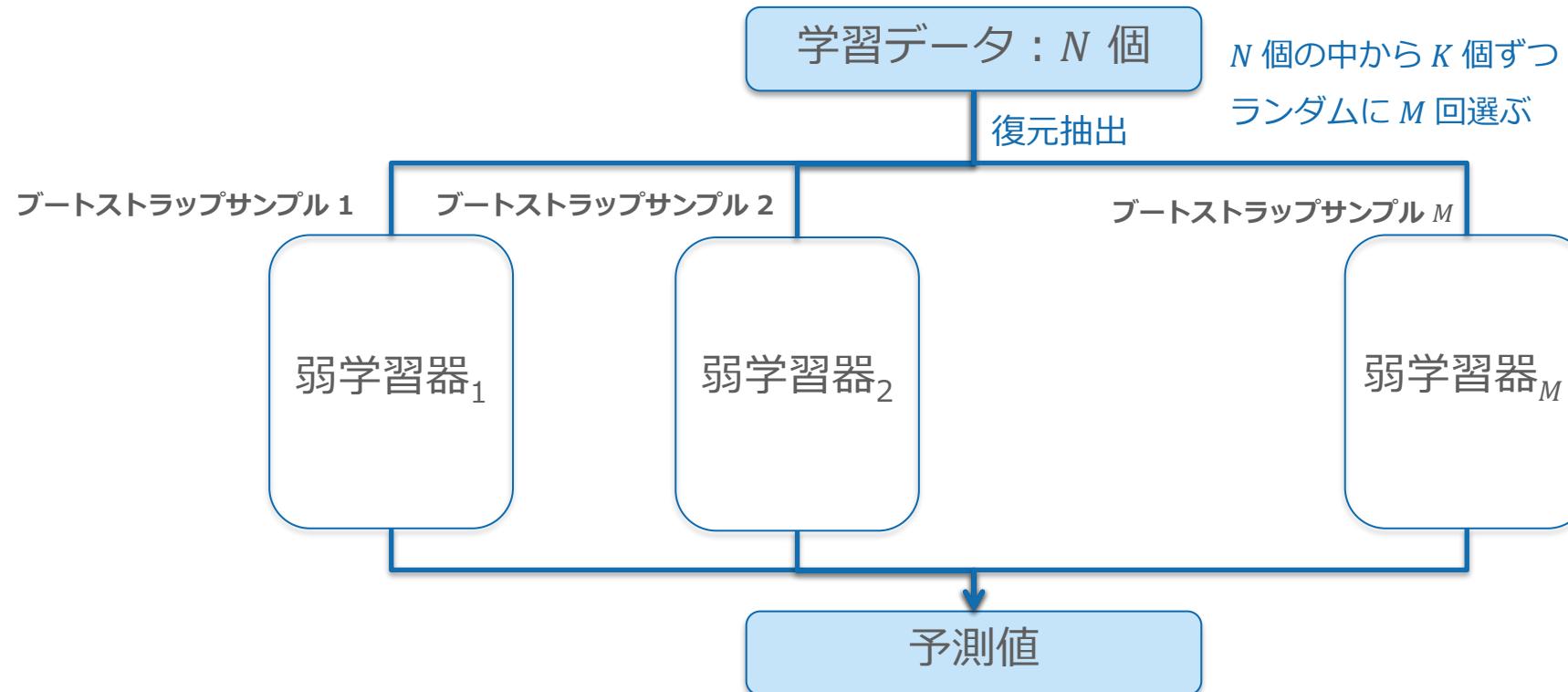
- これまで一つのモデルだけを用いて予測を実施
- 複数のモデルを組み合わせた予測ができるか?
  - 人間では複数人の意見を考慮することで、未知の事象に対して頑健な意思決定を実施（例：多数決による意思決定）

- 複数の学習済みモデルの予測結果を連携して、汎化性能の向上を狙う手法
- 過学習しやすい決定木では特に有効なテクニック
  - 決定木では、木の深さを非常に大きくすれば、学習用データ 1 つのみを判定するルールを作成可能であり、これは過学習の状態になりやすい
- 組み合わせるモデルは異なっていても構わない
  - 決定木 + ロジスティック回帰 など
- 出力（予測値）の取り扱い
  - 分類問題の場合…各モデルで予測されたクラスの多数決
  - 回帰問題の場合…各モデルの予測値の平均値

## ■ 代表的なアンサンブル学習の手法

- バギング
- ブースティング
- スタッキング

- ブートストラップサンプルを用いて複数の弱学習器を並列学習させる方法
- 弱学習器とは、単体では予測性能が出にくいモデルのこと（決定木など）



- ランダムにデータをサンプリングする手法の 1 つ
- サンプリングの手順

- $N$  個のデータ点  $X = \{x_1, x_2, x_3, \dots, x_N\}$  があるとする
- $X$  からランダムに  $K$  個のデータを**復元抽出**  
→ 新たなデータ集合  $X_b$  ができる
- この試行を  $M$  回繰り返すことにより、サイズが  $K$  のデータ集合が  $M$  個生成される

## 復元抽出

同じデータを、複数回選んでもいいという選び方

これにより、 $X$  のいくつかの点は  $X_b$  に**複数回**出現するが、 $X_b$  に入っていない点も存在することになる

元のデータ集合  $X$

データ 1	データ 2	データ 3
$\begin{pmatrix} 0.1 \\ 2 \\ \vdots \end{pmatrix}$	$\begin{pmatrix} 0.6 \\ 8 \\ \vdots \end{pmatrix}$	$\begin{pmatrix} 1.2 \\ 4 \\ \vdots \end{pmatrix}$

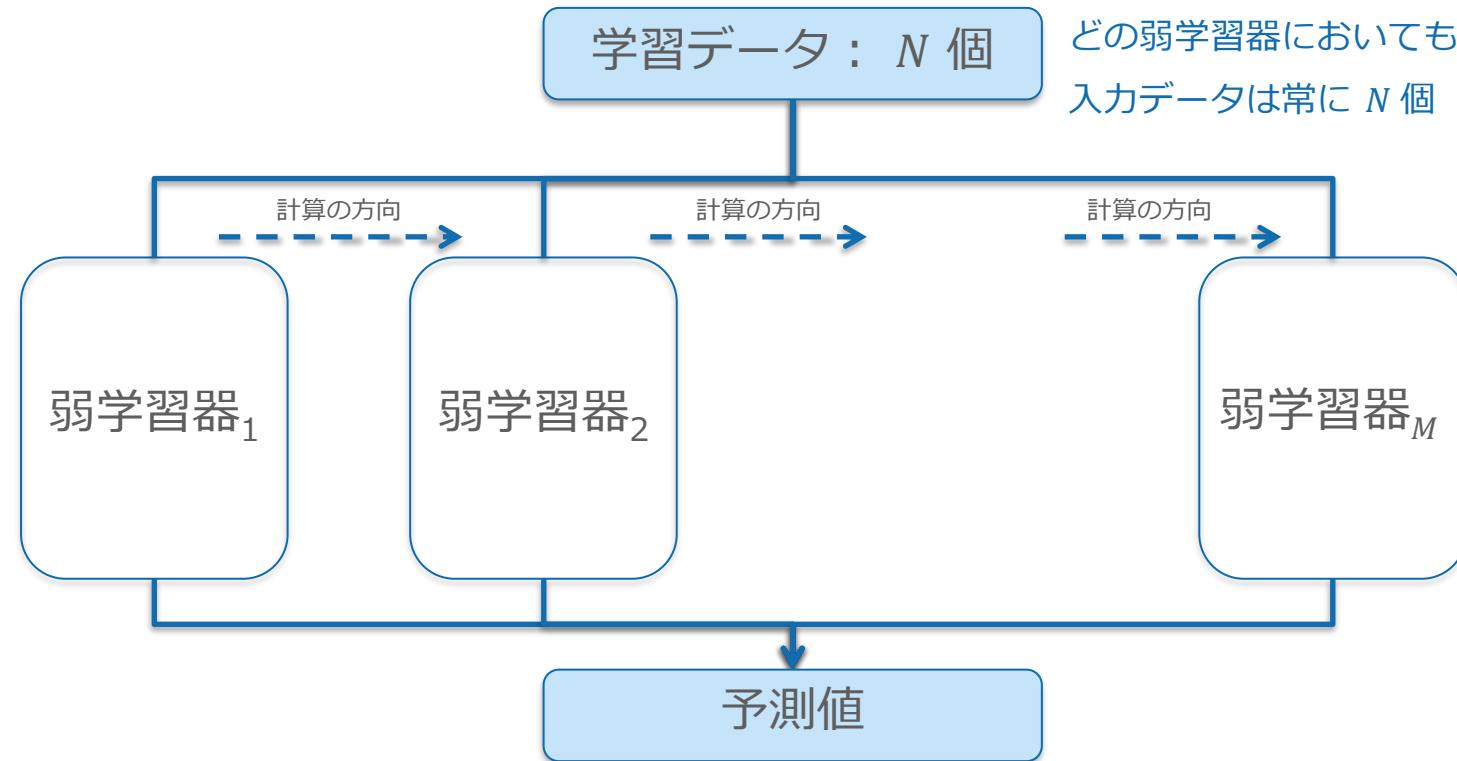


ブートストラップによって生成された  
 $M$  個のデータ集合

データ 1	データ 1	データ 3	データ 1	データ 2	データ 3
データ 2	データ 2	データ 2	データ 1	データ 3	データ 3
⋮	⋮	⋮	⋮	⋮	⋮

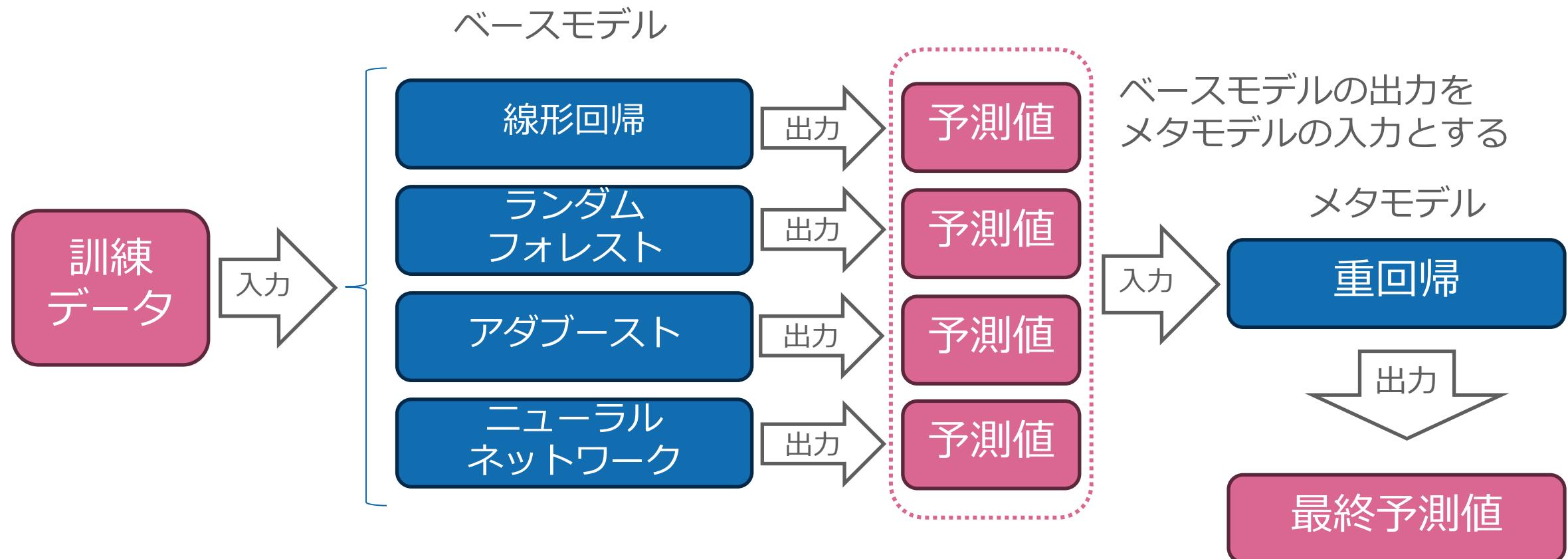
## ■ 複数の弱学習器を用意して、直列的に学習を進める方法

- より誤差が小さくなるように、次の弱学習器を学習させる
- 誤差の算出法は、各種ブースティングアルゴリズムよって様々



- バギングとブースティングはあくまでモデル構築の考え方
  - バギングというモデルがあるわけではない
- バギングとブースティングにおける弱学習器に具体的なモデルを設定することで、一つの大きな機械学習モデルとなる
  - 例) バギングの弱学習器に決定木を利用 = ランダムフォレスト
- バギングは並列計算が行えるが、ブースティングは行えない

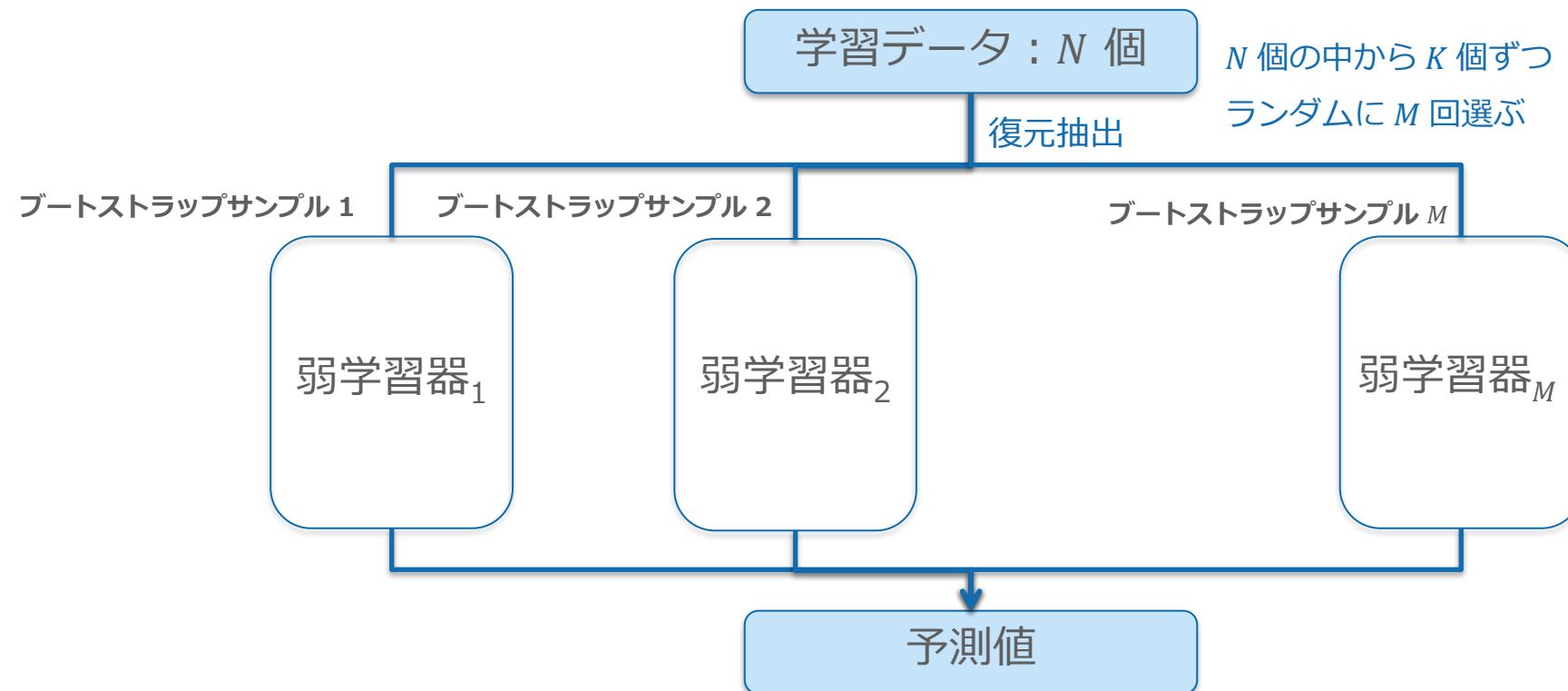
- 複数のベースモデルの予測結果を入力として、メタモデルが最終的な予測を行う
- モデルを積み重ねる（Stack する）イメージ



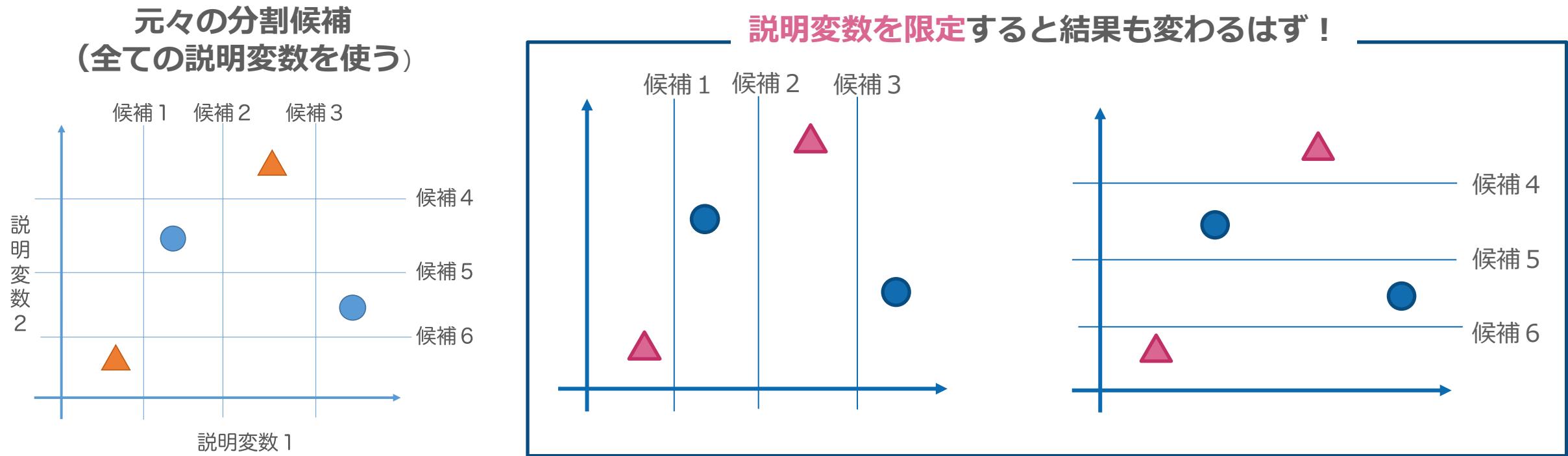
# ランダムフォレスト

---

- バギングをベースに、弱学習器に決定木を用いた方法を考えてみよう
- しかし、バギングだけでは各決定木が似たものになってしまふ
- 決定木のアルゴリズムを活かして、木の**多様性**を上げることはできないか？



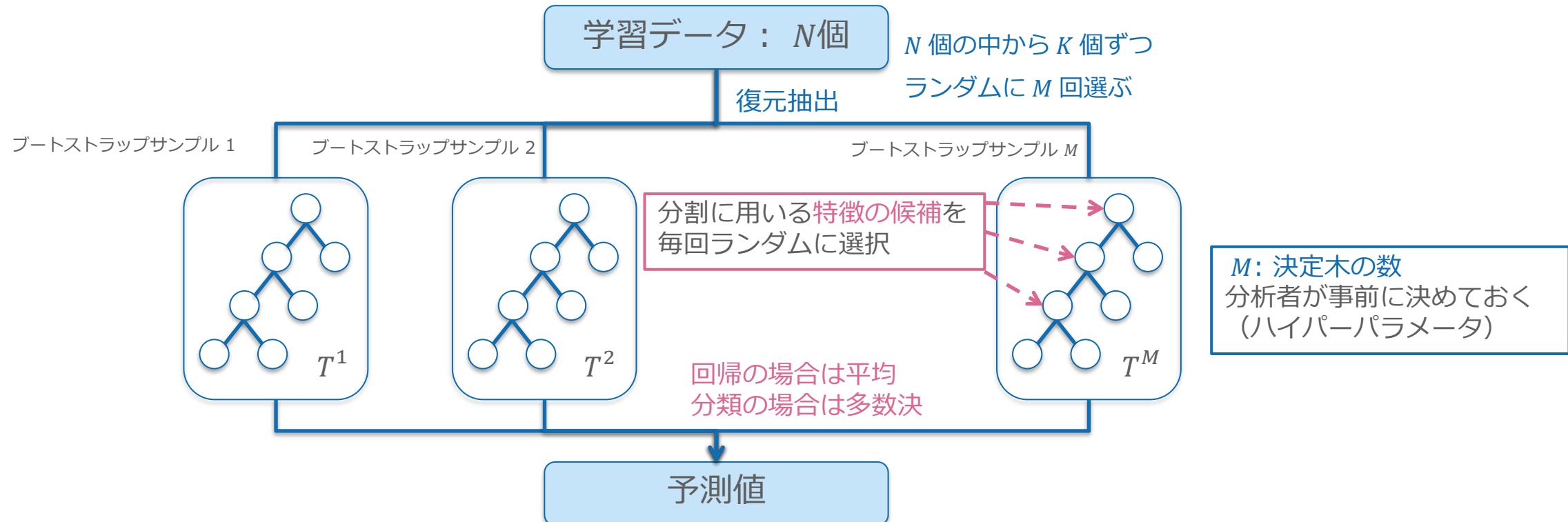
- 分割を決める際に、**使用できる説明変数**を毎回ランダムに決めることで、予測過程が異なるものが作れるはず！



# ランダムフォレスト (Random Forest)

## ■ バギングをベースに、弱学習器に決定木を用いた方法

- 分割条件を決める際に、毎回  $d$  個の中から特徴を  $d'$  個ランダムに選ぶ
  - $d'$  はハイパーパラメータであり、 $d' = \sqrt{d}$  (小数は切り捨て) が推奨されている
- モデル全体の予測値…弱学習器の予測値を平均 (回帰の場合) or 多数決 (分類の場合)



# ランダムフォレスト | 特徴重要度の算出

- ランダムフォレストで学習したあと、各決定木ごとにある説明変数が分割で使用された際の不純度の減少量の総和を計算する
- 各決定木の説明変数ごとの不純度の総減少量を平均することで、特徴の重要度を確認することができる



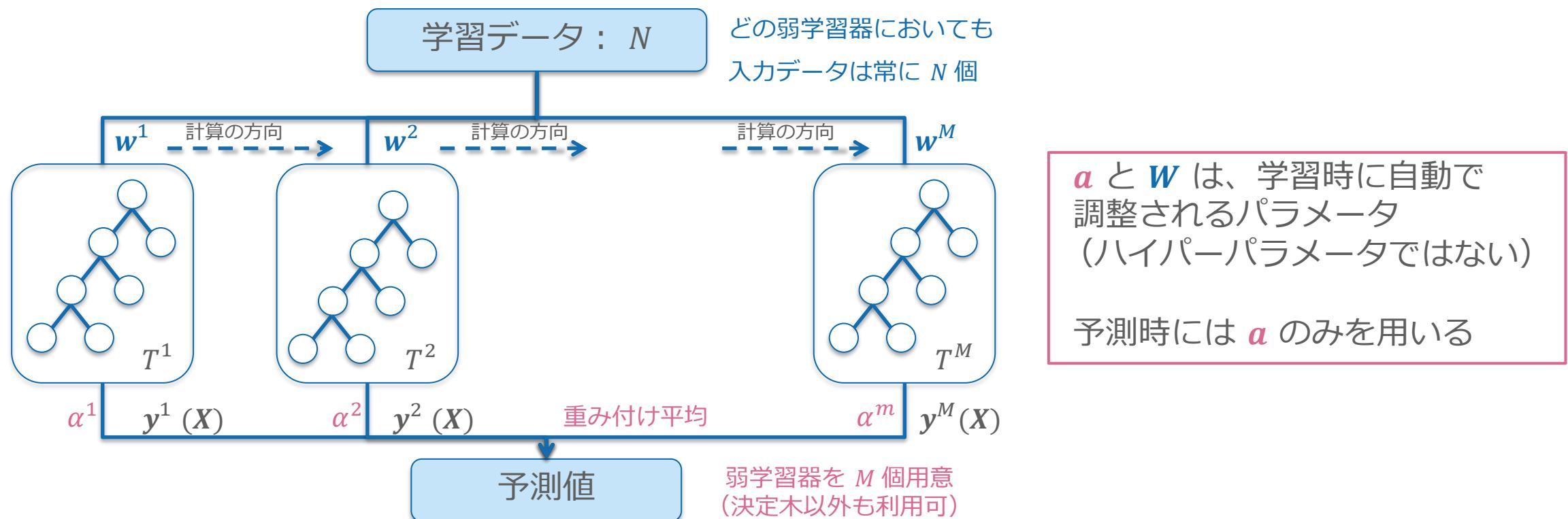
## アダブースト

---

# アダブースト (Adaptive boosting; Adaboost)

## ■ ブースティング手法の 1 つ

- 弱学習器の重み  $\alpha$  と、(弱学習器ごとの) 学習データの重み  $W$  を用意する
- 出力を誤った学習データの  $w$  を大きくして、後続の弱学習器に学習させる際に重視
- モデル全体の予測値…弱学習器の予測値を  $\alpha$  で重み付け平均



# アダブースト | 多クラス分類の場合のアルゴリズム

	数式	意味
1	$w_i^1 = \frac{1}{N}, (i = 1, 2, \dots, N)$	データの重み $w^1$ を初期化 ( $N$ 回ループ)
2	for $m = 1$ to $M$	以下の操作を $M$ 回繰り返す
3	(省略)	データを $w_i^m$ で重みづけしつつ 弱学習器 $T^m$ を学習
4	$\text{err}^m = \frac{\sum_{i=1}^N w_i^m I(y^m(X_i) \neq t_i)}{\sum_{i=1}^N w_i^m}$	弱学習器の誤り率 $\text{err}^m$ を計算
5	$\alpha^m = \log \frac{1 - \text{err}^m}{\text{err}^m} + \log(K - 1)$	弱学習器の重み $\alpha^m$ を計算 ( $K$ : クラス数)
6	$w_i^{m+1} = w_i^m \exp(\alpha^m I(y^m(X_i) \neq t_i))$ $(i = 1, 2, \dots, N)$	データの重み $w_i^m$ を更新 ( $N$ 回ループ)
7	(省略)	データの重み $w_i^{m+1}$ を 合計が 1 になるよう正規化
8	$C(X_i) = \arg \max_k \sum_{m=1}^M \alpha^m I(y^m(X_i) = k)$	クラスの予測結果 $C(X_i)$ を出力

$I(y^m(X_i) \neq t_i)$  は  
学習器  $T^m$  の出力  $y^m(X_i)$  が  
教師データ  $t_i$  と一致したとき 0、  
一致しなかったときに 1 となる関数

$M$  回ループ

$M$ : 弱学習器の数  
分析者が事前に決めておく  
(ハイパーパラメータ)

参考 : Ji Zhu et al.,  
[Multi-class AdaBoost](#)

## 勾配ブースティング

---

## ■ ブースティング手法の 1 つ

- ある弱学習器の誤差を別の弱学習器に学習させる
- 誤差には、目的関数の負の勾配を用いる

弱学習器に勾配を出力させるのが  
アダブーストとの大きな違い  
(アダブーストは予測値を出力)

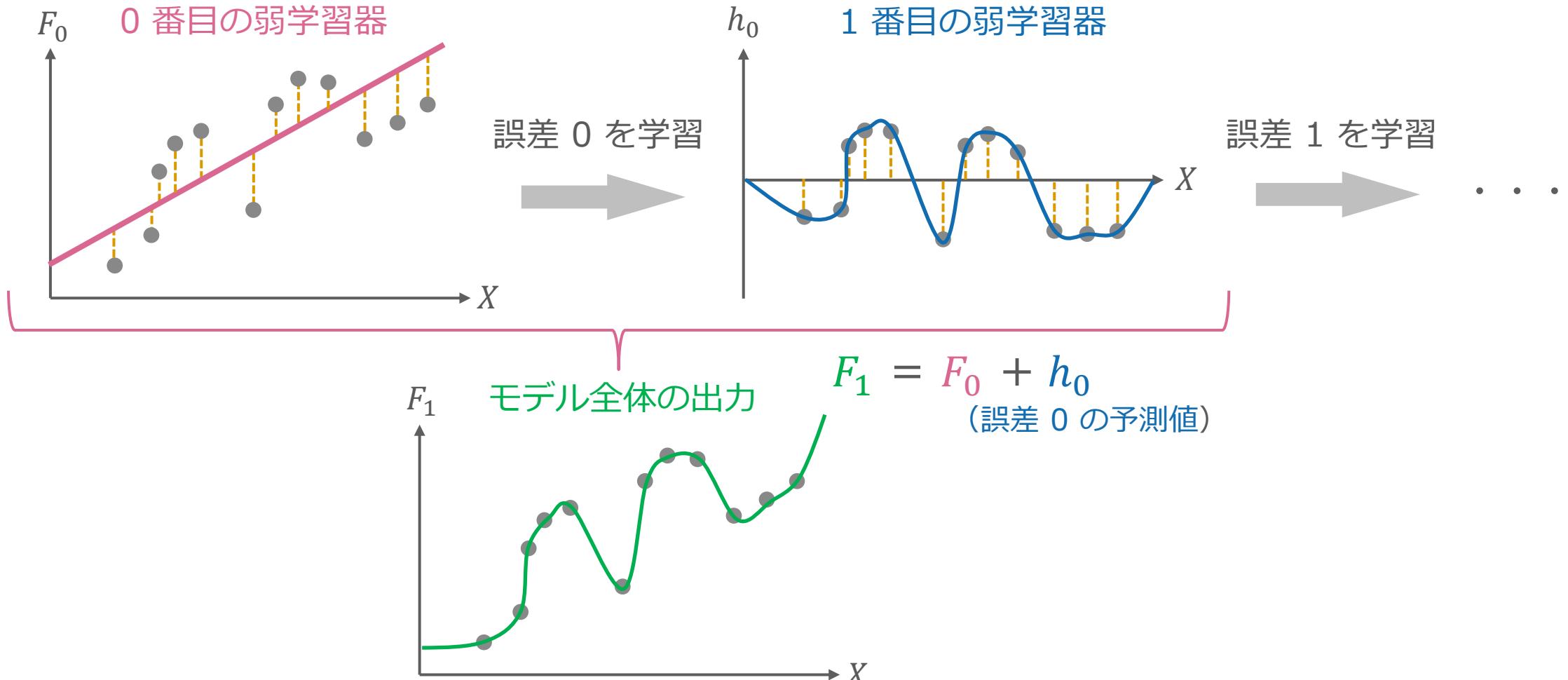
## ■ 回帰と分類の両方で利用可能

## ■ 特に、弱学習器に決定木を用いたものを

勾配ブースティング木 (Gradient Boosting Decision Tree; GDBT) という

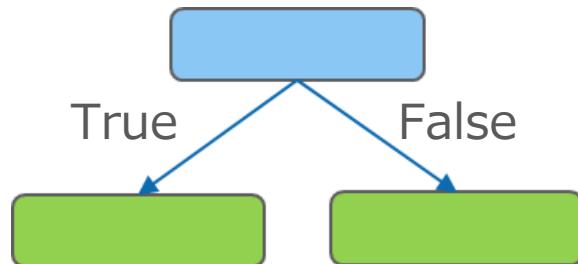
- 弱学習器は、微分可能な評価基準 (二乗誤差関数や交差エントロピーなど) を設定できるモデルであれば何でもよい

前の木の誤りを次の木が正していく



# 勾配ブースティング | 計算例

データNo.	説明変数			目的変数
	ゲーム が好きか？	ゴルフ が好きか？	...	
1	True	False	False	15
2	False	False	True	25
3	False	True	False	35
4	:	:	:	:



- いずれかの説明変数の条件で分割
- 予測値は分割後に集まった目的変数の平均

ここでは分かりやすいように、誤差を「正解と予測値の単純な引き算」と定義

データNo.	年齢	初期値	初期値との誤差	決定木1 予測値	決定木1 誤差	決定木2 予測値	最終予測値
1	15	30	-15	-10	-5	-2	18
2	25	30	-5	-10	5	3	23
3	35	30	5	2	3	3	35
4	:	:	:	:	:	:	:

本来は勾配を弱学習器の  
誤差として扱う

- 決定木 1 は初期値と正解の誤差を予測
- 決定木 2 は「初期値 + 決定木 1 予測値」の誤差を予測
- 最終的な予測値：初期値 + 決定木 1 予測値 + 決定木 2 予測値

## ■ モデルの更新方法

- $F_{t+1}(x) = F_t(x) + h_t(x)$
- 初期のモデル  $F_0(x)$  に新たな弱学習器  $h_t(x)$  を逐次的に加えていくことにより、予測値と正解値との乖離を埋めるように更新
- 弱学習器  $h_t(x)$  は、予測値と正解値との乖離が大きいときに大きな値を出力し、予測値と正解値との乖離が小さいときは小さな値を出力することが望ましい
- そこで  $h_t(x)$  を求める際に勾配情報を用いる
- $F_t(x)$  が勾配法のパラメータになる

# 勾配ブースティング | アルゴリズム

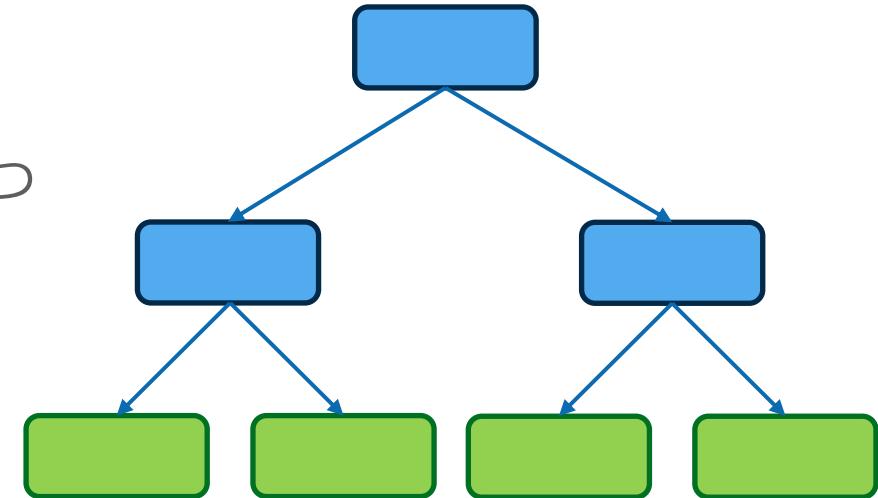
	数式	意味	
1	$F_0(x) = \arg \min_{F_1} \sum_{i=1}^N L(y_i, F_1(x_i))$	誤差が最小になるようにモデルを初期化	
2	for $t = 0$ to $T - 1$	以下の操作を $T$ 回繰り返す	
3	for $i = 1$ to $N$	データごとに以下を行う ( $N$ : データサイズ)	
4	$r_{im} = \frac{\partial L(y_i, F_t(x_i))}{\partial F_t(x_i)}$	勾配 $r_{im}$ を計算	<p><math>N</math> 回ループ</p> <p><math>T</math> 回ループ</p> <p><math>T</math>: 弱学習器の数</p> <p>分析者が事前に決めておく (ハイパーパラメータ)</p>
5	$r_m = \frac{1}{N} \sum_{i=1}^N r_{im}$	勾配の平均値 $r_m$ を計算	
6	(省略)	$r_m$ を目的変数として弱学習器 $g_t(x)$ を学習	
7	$F_{t+1}(x) = F_t(x) - \eta g_t(x)$ $= F_t(x) + h_t(x)$	$g_t(x)$ を使って次のモデル $F_{t+1}(x)$ を作る	

## ■ 最終的な予測モデルとして $F_T(x)$ を出力

- 回帰モデルの場合
  - 弱学習器の出力はスカラー
  - $F_T(x)$  に含まれる各  $h_t(x)$  の出力の総和を最終的な予測値とする
- 分類モデルの場合
  - 弱学習器の出力は、各クラスに当てはまる確率を表すベクトル
  - 出力の総和をソフトマックス関数に入力し、出力値が最大となるクラスを選択

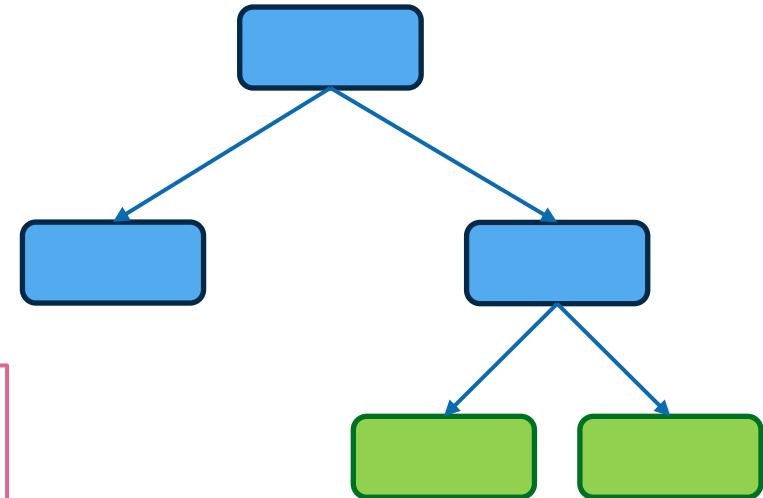
## ■ XGBoost

- 勾配ブースティング木 (GBDT) のアルゴリズムの 1 つ
- 深さ優先 (Level-wise) で決定木を生成
  - 全てのノードを分岐させる
- 処理速度が遅い



## ■ LightGBM

- XGBoost よりも軽量で新しいアルゴリズム
- 幅優先 (Leaf-wise) で決定木を生成
  - 情報利得の大きいノードのみ分岐させる
- 過学習しやすい



どちらも、実務や Kaggle で  
よく用いられるライブラリ

## ■ XGBoost の評価式

$$L(\phi) = \sum_i l(t_i, \mathbf{y}_i) + \sum_k \Omega(f_k)$$

損失関数                            罰則項

$$\mathbf{y}_i = \sum_k f_k(x_i)$$

$f_k$  :  $k$  個目の決定木  
 $\mathbf{y}_i$  : データ  $x_i$  に対するモデル全体の予測値  
 $t_i$  : データ  $x_i$  に対する正解値

## ■ XGBoost の罰則項

- 木の複雑さに制約を設けて、正則化

$$\Omega(f) = \underline{\gamma T} + \frac{1}{2} \underline{\lambda \|w\|_2^2}$$

$\gamma, \lambda$  : ハイパーパラメータ  
 $T$  : 葉（終端ノード）の数  
 $w$  : 各葉における（誤差の）予測値

$\gamma$  を大きくすると…

$T$  が小さくなる  
= 木のサイズが小さくなる  
(XGBoost の弱学習器はすべて二分木のため)

$\lambda$  を大きくすると…

$w$  の要素が小さくなる  
= 木 1 つあたりの更新量が小さくなる  
= 木の数が増える

- 勾配を使って損失が最小になるようにパラメータを更新するのが**勾配降下法**

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \frac{\partial L}{\partial \mathbf{w}_t}$$

$L$  : 誤差関数（損失関数）

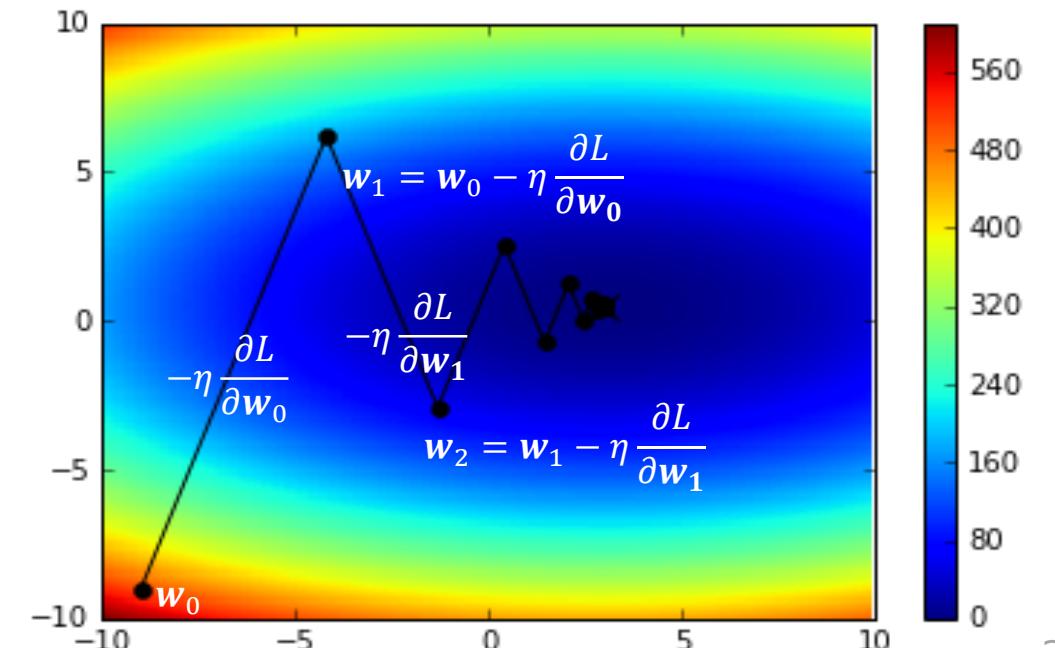
$\mathbf{w}_t$  :  $t$  ステップ目のモデルパラメータ

$\eta$  : 学習率（1回あたりの更新の程度を制御する係数）

- 勾配降下法は、今のパラメータ  $\mathbf{w}_t$  に負の勾配と学習率を掛けたもの

$-\eta \frac{\partial L}{\partial \mathbf{w}_t}$  を逐次的に加えて更新する方法とも言える

- 更新対象を「**今の学習器（の集まり）**」  
として、同じ考え方ができるだろうか？



- 勾配降下法の更新対象を学習器  $F_t(x)$  として書き直すと...

$$F_{t+1}(x) = F_t(x) - \eta \frac{\partial L}{\partial F_t(x)}$$

- $L$  を二乗誤差関数とすれば...

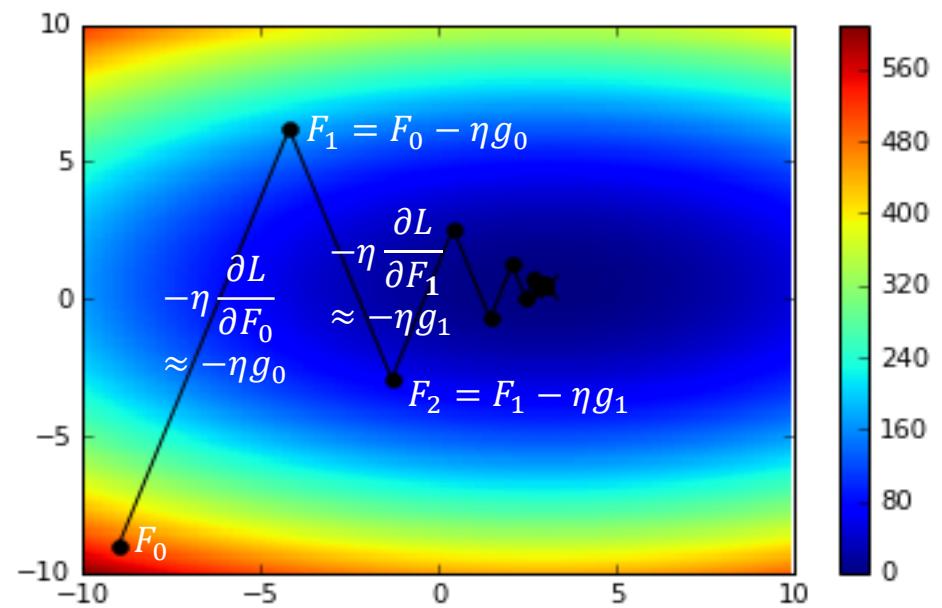
$$L = \frac{1}{2} (y - F_t(x))^2$$

$y$  : 目的変数 (教師データ)  
 $F_t(x)$  :  $t$  ステップ目の予測値

$$\frac{\partial L}{\partial F_t(x)} = -(y - F_t(x)) = F_t(x) - y$$

この勾配値を目的変数として  
新たな弱学習器  $g_t(x)$  を学習！

勾配降下法に合わせた  
勾配ブースティングのイメージ



- 勾配  $\frac{\partial L}{\partial F_t(x)}$  = 弱学習器の出力  $g_t(x)$  として、更新式を書き直す
- さらに、 $-\eta g_t(x)$  を  $h_t(x)$  に置き換える

$$\begin{aligned}F_{t+1}(x) &= F_t(x) - \eta \frac{\partial L}{\partial F_t(x)} \\&= F_t(x) - \eta g_t(x) \\&= F_t(x) + h_t(x)\end{aligned}$$

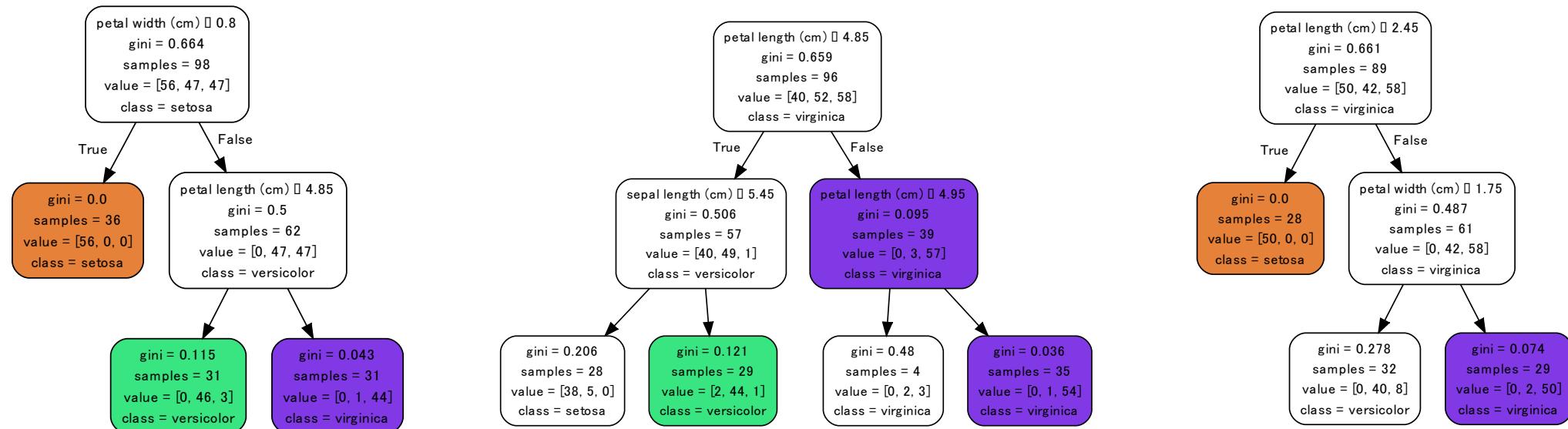
これで勾配ブースティングの  
更新式が完成！

## ノートブック演習

---

## 9-1\_random\_forest.ipynb

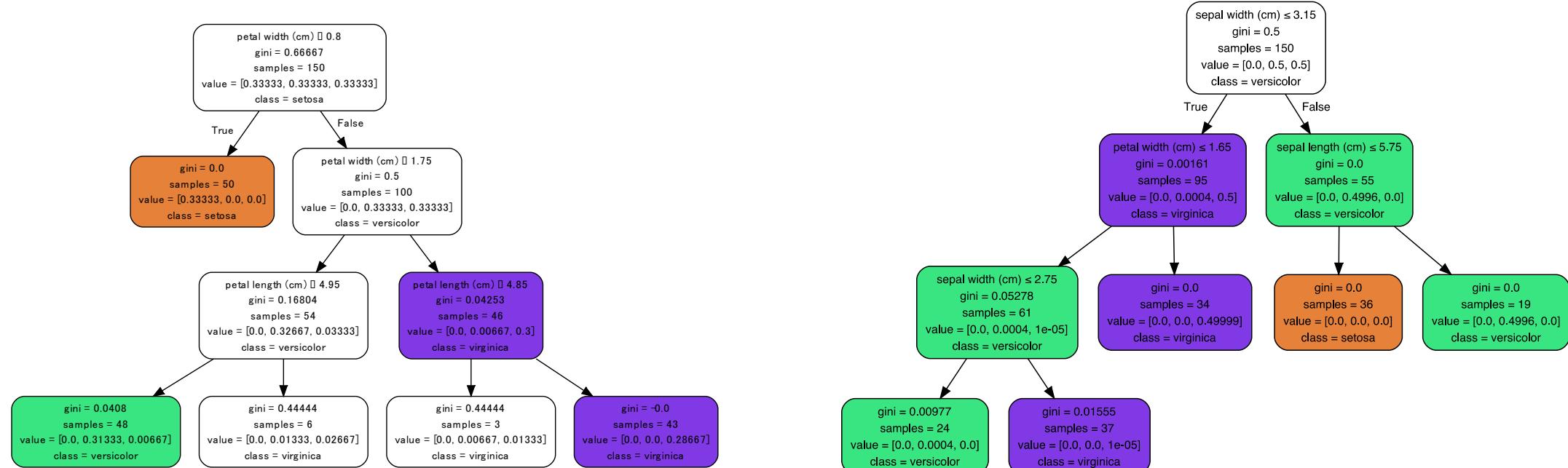
- ランダムフォレストを用いて、アヤメの分類を行ってみましょう
- 不純度の評価方法や木の深さなどのハイパーパラメータを変更し、どのような変化があるか確認しましょう



# ノートブック演習 | アダブーストと決定木

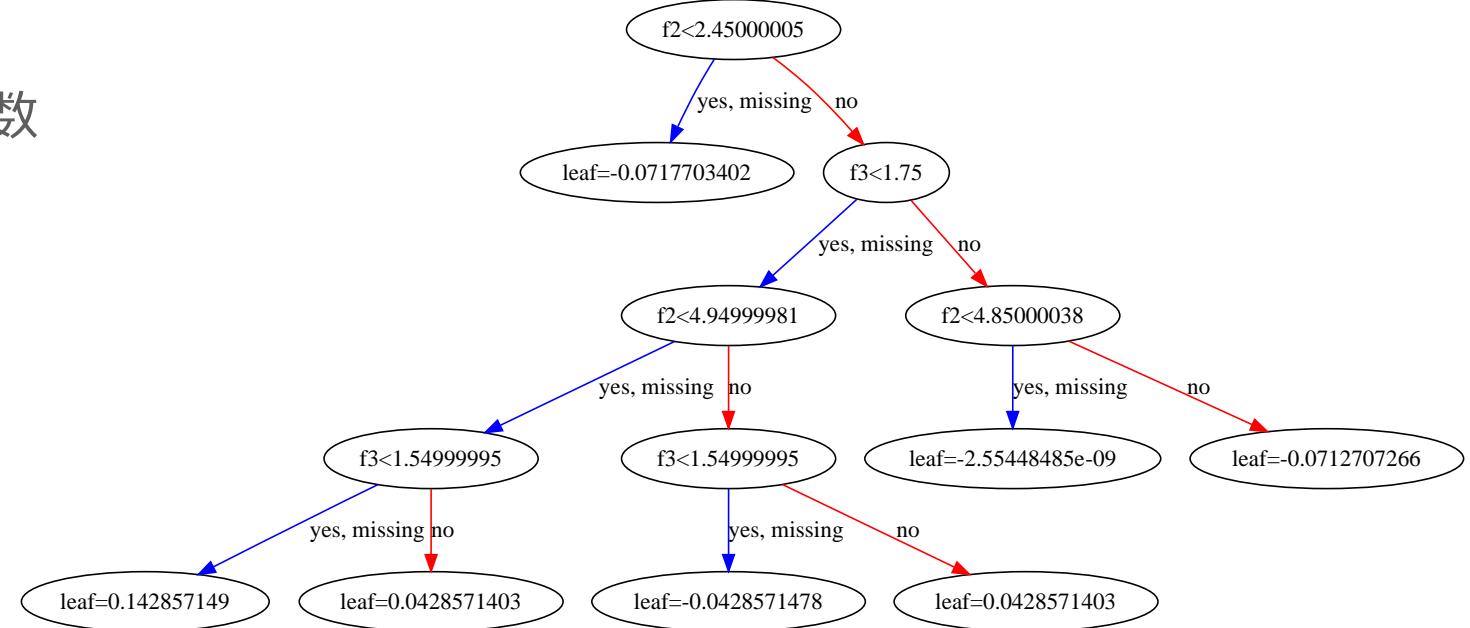
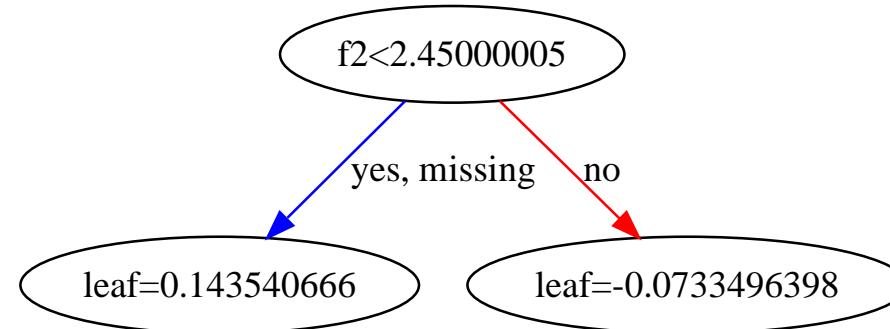
## 9-2\_adaboost.ipynb

- アダブーストと決定木を用いて、アヤメの分類を行ってみましょう
- 不純度の評価方法や木の深さなどのハイパーパラメータを変更し、どのような変化があるか確認しましょう



## 9-3\_xgboost.ipynb

- XGBoostによる勾配ブースティング+決定木を試してみましょう
- 以下のハイパーパラメータを変更し、出力の違いを確認しましょう
  - `max_depth` : 木の深さの最大値
  - `learning_rate` : 学習率
  - `n_estimators` : 追加する木の数



## 本章のまとめ

---

- アンサンブル学習
- ランダムフォレスト
- アダブースト
- 勾配ブースティング
- ノートブック演習

## ■ アンサンブル学習の手法

- バギング
  - ブートストラップサンプルを用いて
  - 複数の弱学習器を並列で学習させる
- ブースティング
  - 複数の弱学習器を直列で学習させる
  - より誤差が小さくなるように、次の弱学習器を学習させる
- スタッキング
  - ベースモデルの予測結果を入力として、メタモデルを学習させる

## ■ ランダムフォレスト

- ・ バギングに決定木を組み合わせたアルゴリズム
- ・ 分岐に使用する説明変数の候補を、毎回ランダムに決める

## ■ アダブースト

- ・ ブースティングを用いたアルゴリズムの 1 つ
- ・ 前の弱学習器が学習できなかったデータを、後続の弱学習器が優先的に学習する

## ■ 勾配ブースティング

- ・ ブースティングを用いたアルゴリズムの 1 つ
- ・ 弱学習器に決定木を用いたものを勾配ブースティング木 (GDBT) という
- ・ 前段のモデルの誤差（勾配）を目的変数として、後続のモデルが学習を行う

# 現場で使える 機械学習・データ分析基礎講座

第 10 章：サポートベクターマシン

- サポートベクターマシンの基本概念
- ハードマージン法
- ソフトマージン法
- カーネル法
- ノートブック演習

- サポートベクターマシンがどのような最適化問題に帰着するのかを説明できる
- サポートベクターマシンのハイパーパラメータ  $C$  の意味を説明できる
- サポートベクターマシンにおいてガウスカーネルを利用する意義を説明できる

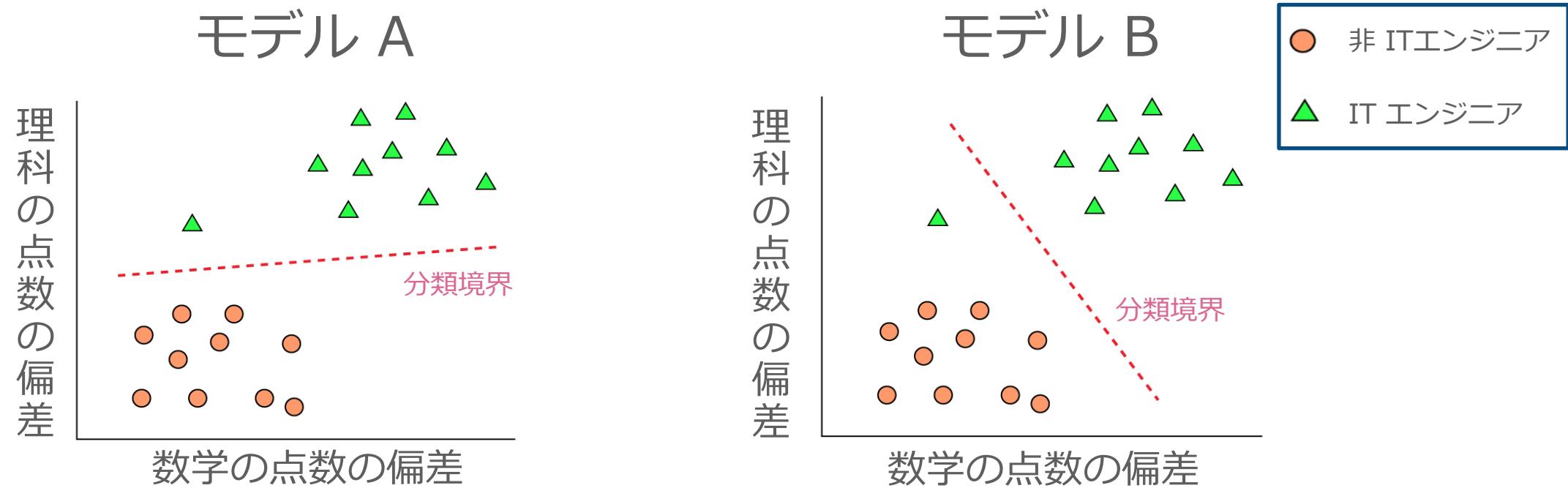
## サポートベクターマシンの基本概念

---

- 2 クラス分類問題を解くためのモデル
- scikit-learn における実装では、多クラス分類にも対応※
- サポートベクターマシンを回帰問題に応用した、サポートベクター回帰 (Support Vector Regressor; SVR) というモデルも存在

※ 「クラス A or クラス B」「クラス A or クラス C」「クラス B or クラス C」のように複数の 2 クラス分類 SVM を組み合わせて多クラス分類を行う one-vs-one という実装方法が採用されている ([参考](#))

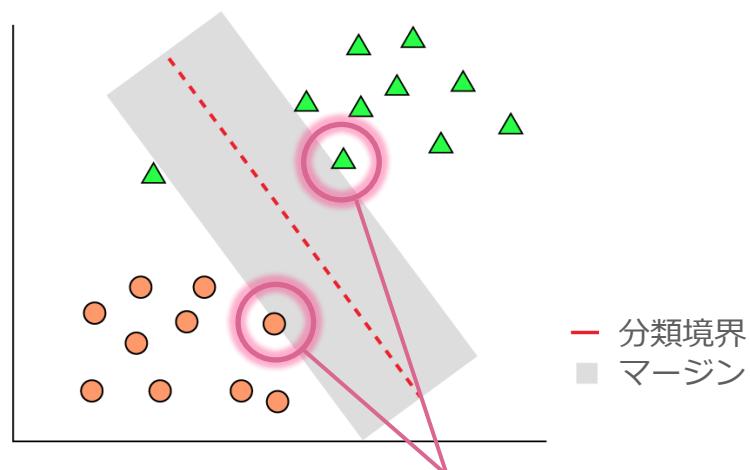
- 理科の点数の偏差、数学の点数の偏差から将来 IT エンジニアになるかならないかを分類するモデルを構築したい
  - ・ 新たなデータに対してもできるだけ誤分類をしない境界線を決めたい
  - ・ 下記モデル A と B のどちらのモデルの方が誤分類が少ないだろうか



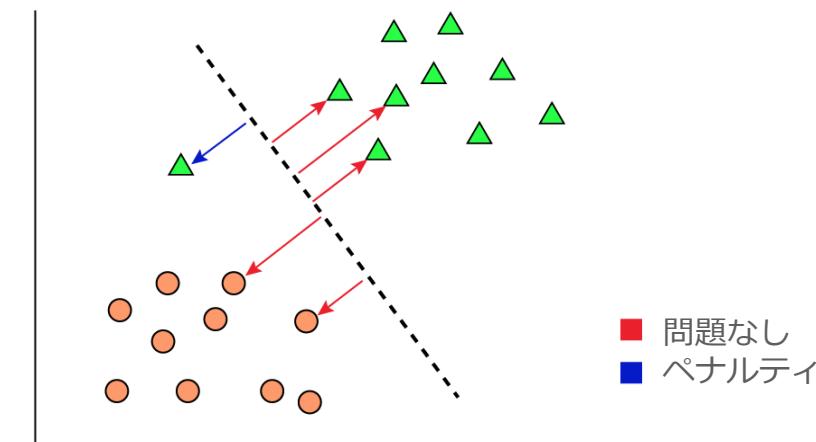
## ■ SVM の分類境界の両側にあるグレーの部分をマージンと呼ぶ

- マージンが大きい境界の方が、汎化性能が高くなりやすい
- クラスを完全に分離できなくてもよいが、誤った分類結果にはペナルティが与えられる
- マージンの両端と重なるデータをサポートベクトルと呼ぶ

## ■ マージンが大きくなり、ペナルティが小さくなるような分類境界を求める

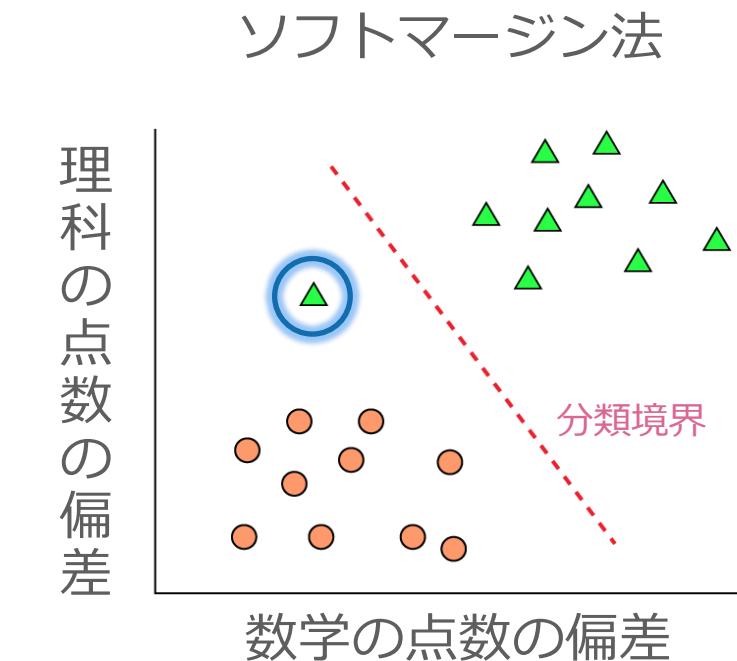
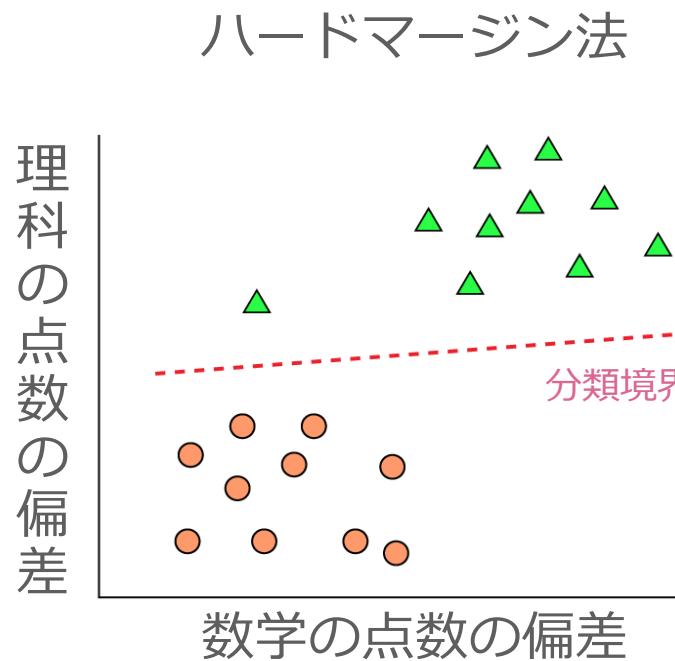


マージンの基準は「サポートベクトル」



誤った分類結果に「ペナルティ」を与える

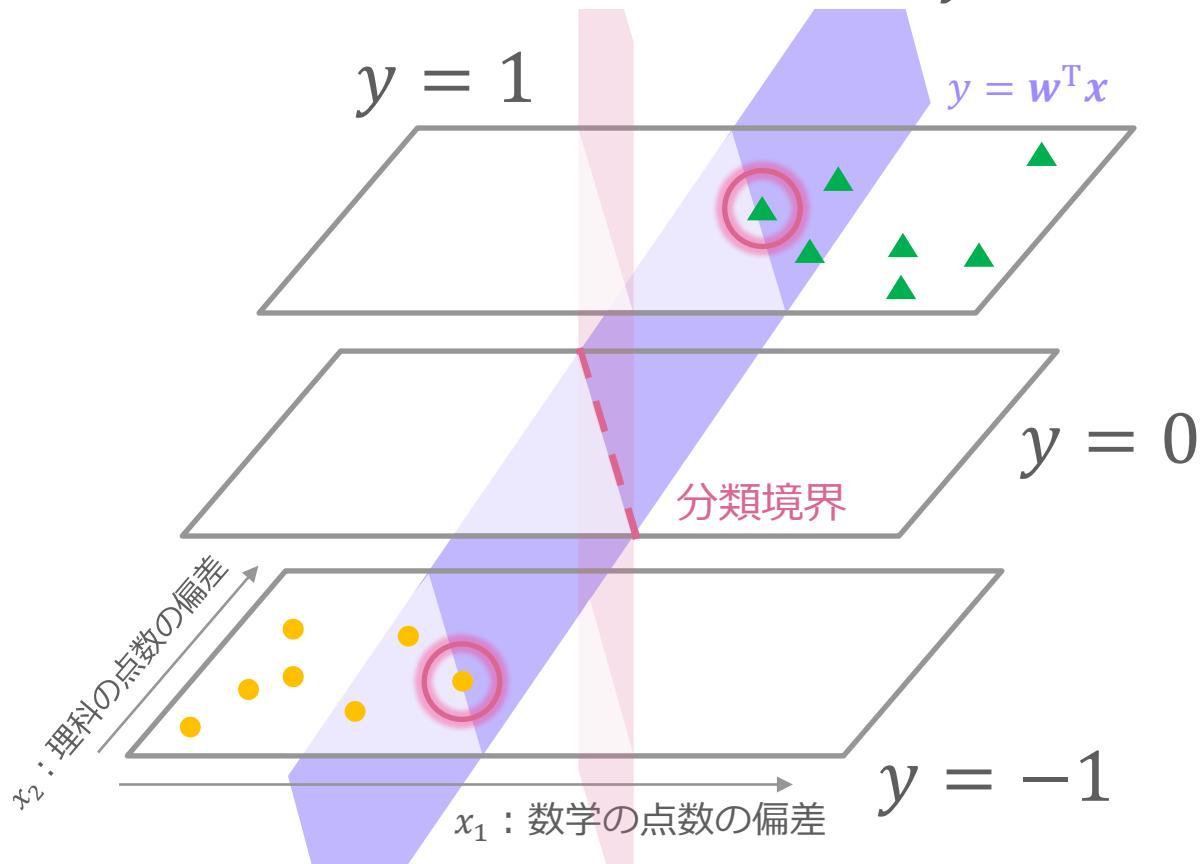
- SVM は、誤分類を許容するかどうかで 2 つの手法に分かれる
  - ・ ハードマージン法：誤分類を許容しない方法
  - ・ ソフトマージン法：ペナルティを設けて、誤分類を許容する方法



# ハードマージン法

---

- クラスを  $y = -1$  と  $y = 1$  に対応させる
- 目的変数に対する線形回帰モデルを、線形識別関数  $y = \mathbf{w}^T \mathbf{x}$  とする
- 分類境界は、線形識別関数と  $y = 0$  の交点と同じ  $x$  座標をもつ点の集まり

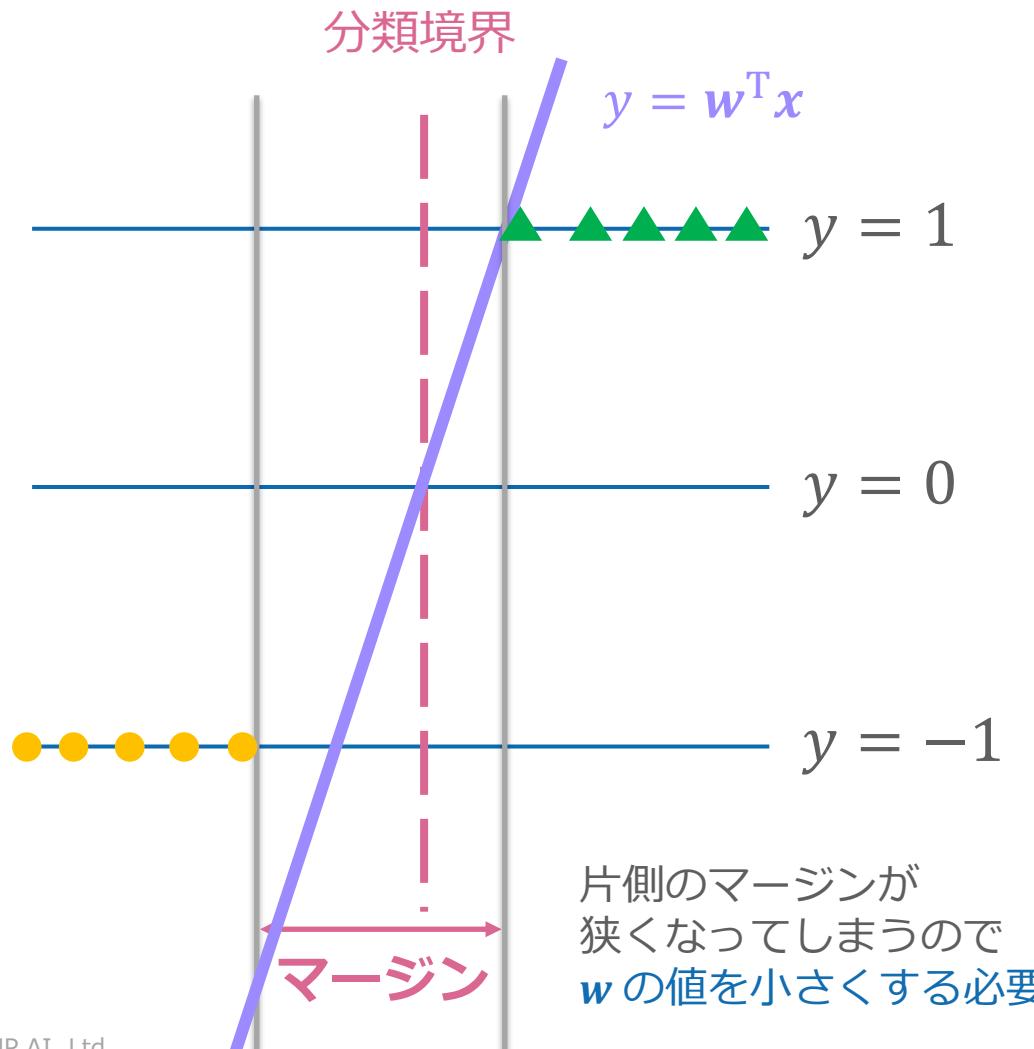


線形識別関数は  $d$  次元の超平面

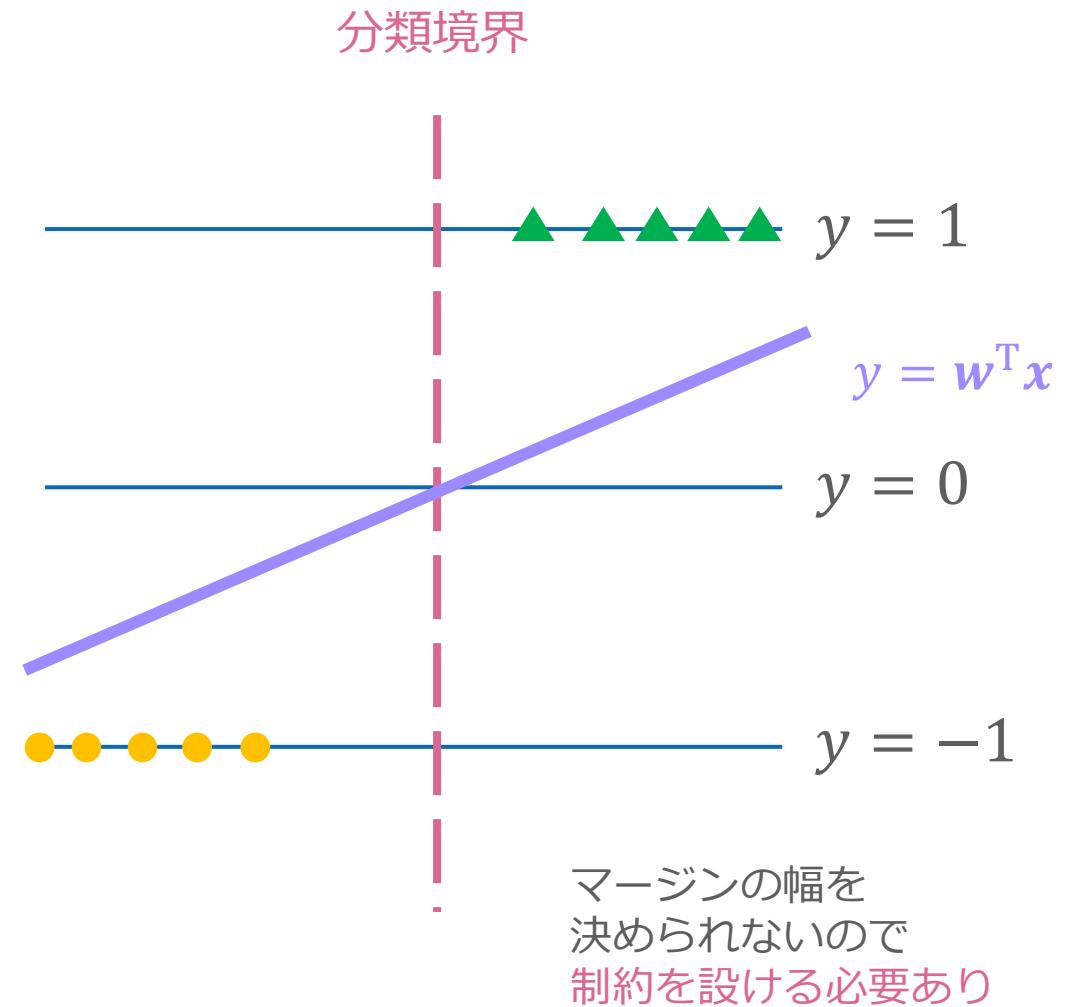
分類境界は「線形識別関数 = 0」  
を満たす  $d - 1$  次元の超平面

マージンを最大化する分類境界は、  
紫面の傾きを小さくすることで求まる

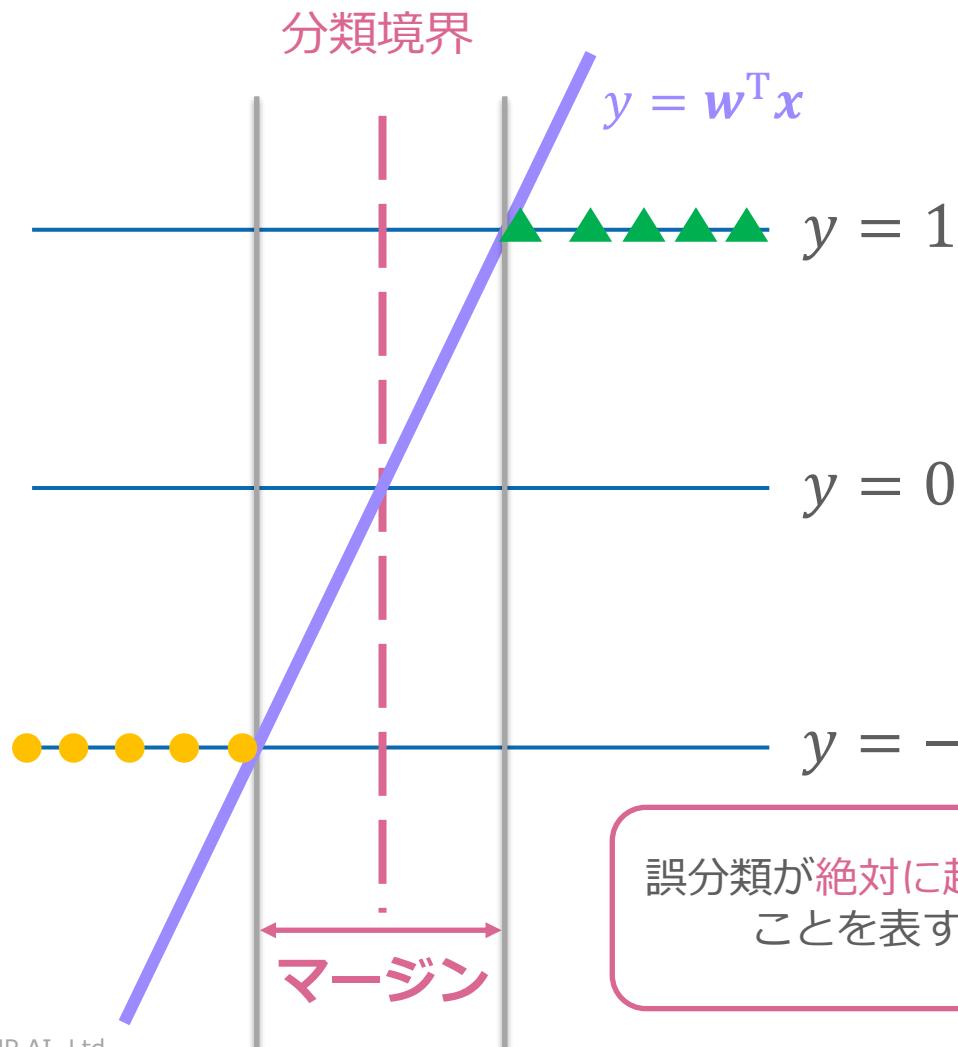
## 線形識別関数の傾きが大きい場合



## 線形識別関数の傾きが小さい場合



適切な傾きをもつ線形識別関数



$w$  の値が小さいと  $y = w^T x$  の傾きは小さくなり  
マージンが大きくなる

→  $w$  の値をなるべく小さくしたい  
→  $w$  のノルムを最小にすれば良い

マージンを最大化する問題 =  $\|w\|_2^2$  の最小化問題

ただし原則として

△ は  $y = w^T x$  の下  
○ は  $y = w^T x$  の上

→  $t_i \cdot w^T x_i \geq 1(i = 1, \dots, N)$  という制約  
 $N$  : データサイズ

データ  $i$  が クラス 1 であれば  $t_i = 1$   
データ  $i$  が クラス 0 であれば  $t_i = -1$

## ソフトマージン法

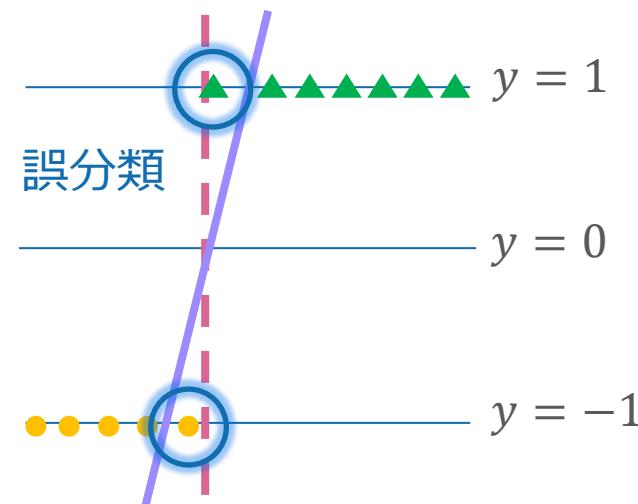
---

- ペナルティが大きくならないように注意しながら、なるべくマージンが大きくなるように傾きの小さい平面を探す
  - ・ 正しく分類できるようにしつつ、ある程度マージンを狭くとる（平面の傾きを大きくする）
- 誤分類となるのは、分類境界ではなく、線形識別関数をまたいだデータ
  - ・ マージンの内側に誤分類データが入ることもある

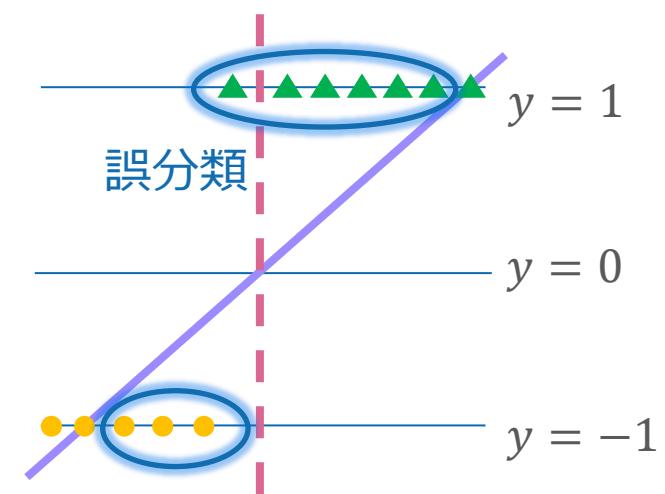
もう少しペナルティを許容して大きなマージンを確保しても良い



ペナルティの大きさとマージンの大きさのバランスが良い



マージンは十分に大きいがペナルティが大きすぎる



- サポートベクターマシンはマージンを最大化する  $w$  を探索する
- その探索は下記の最適化問題を解くことに帰着

$t_i \cdot w^T x_i \geq 1(i = 1, \dots, N)$  の制約の下で  
 $w$  の目的関数  $\frac{1}{2} \|w\|_2^2$  を最小化

- ただし、上記最適化問題ではペナルティへの考慮がない

## ■ ペナルティを考慮した SVM の最適化問題

$t_i \cdot w^T x_i \geq 1 - \xi_i$  ( $i = 1, \dots, N$ ) の制約の下で  $w$  の目的関数  $\frac{1}{2} \|w\|_2^2 + C \left( \sum_{i=1}^N \xi_i \right)$  を最小化

ペナルティ項

$\xi_i$  はデータ  $i$  に対するペナルティ ( $C$  は後ほど説明)

ある境界が引かれた後、データ  $i$  を誤分類している場合  $\xi_i$  は正の値を取る

いくつかのデータが誤分類され  $C \left( \sum_{i=1}^N \xi_i \right)$  が正の値をとったとしても

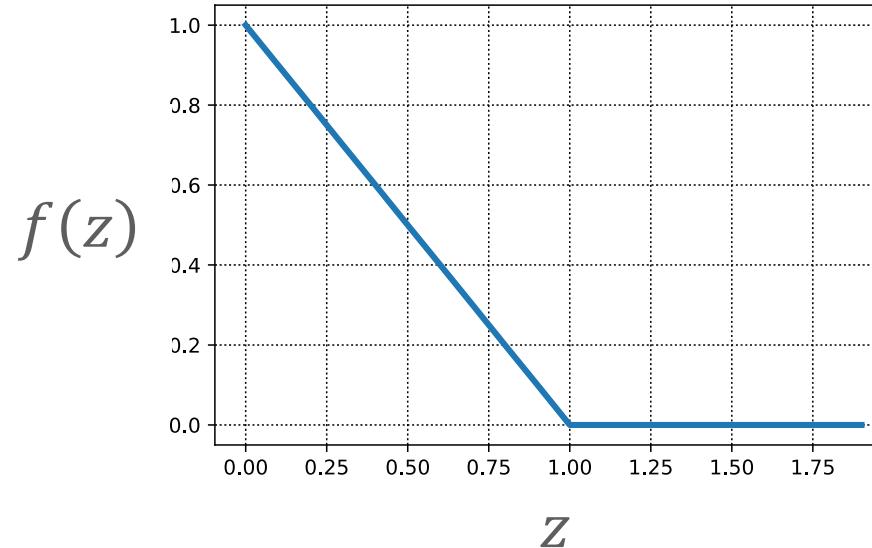
目的関数全体が最小となるのであれば、そのときの  $w$  の方が良い

- $\xi$  (グザイ) はスラック変数と呼ばれ、通常ヒンジ損失関数を用いて算出される

## ヒンジ損失関数

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

入力  $z$  が 1 以上なら 0  
1 未満なら  $1 - z$  を返す

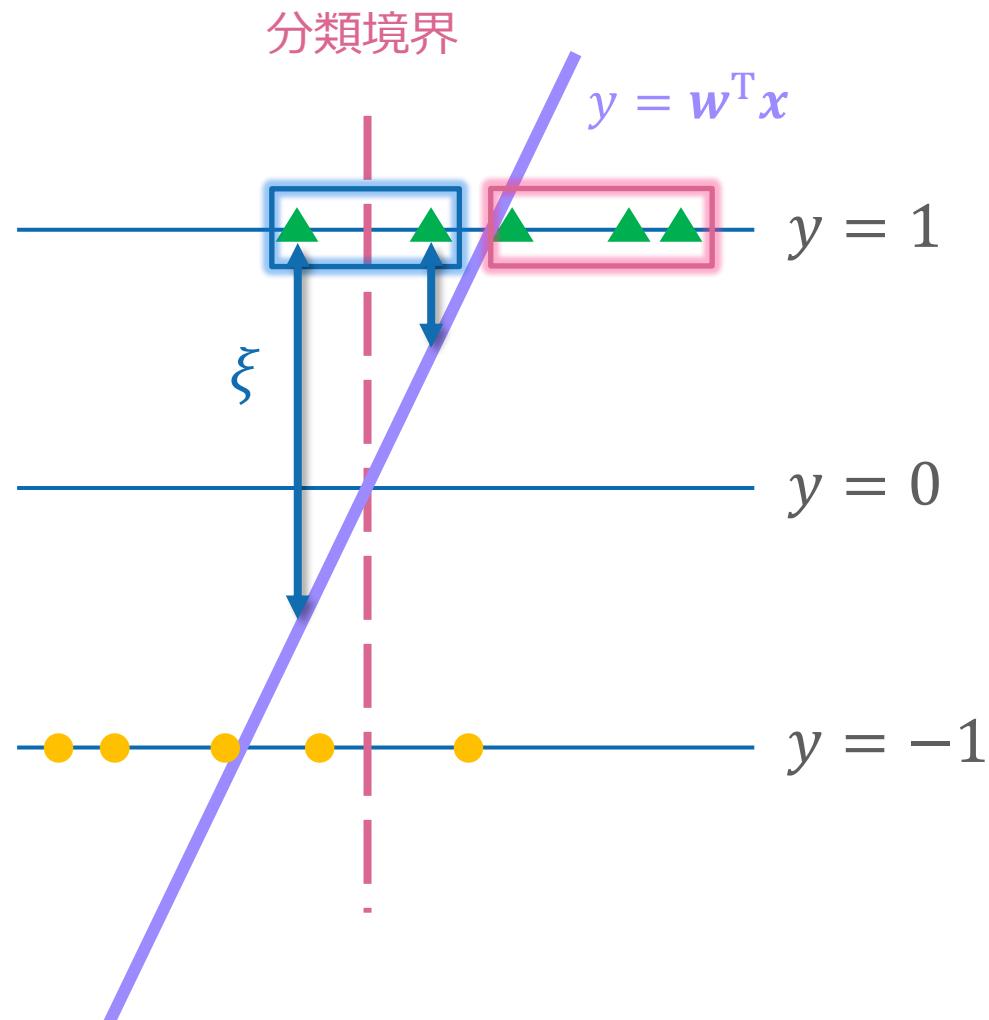


$$\xi_i = f(\underline{\mathbf{w}^T \mathbf{x}_i}) = \max[0, 1 - t_i \cdot \underline{\mathbf{w}^T \mathbf{x}_i}]$$

線形識別関数

$$t_i = \begin{cases} 1 & \text{データ } i = \text{クラス 1} \\ -1 & \text{データ } i = \text{クラス 0} \end{cases}$$

## ■ $\xi$ の具体的なイメージ (クラス 1 の場合)



$$\xi_i = f(\underline{\mathbf{w}^T \mathbf{x}_i}) = \max[0, 1 - t_i \cdot \mathbf{w}^T \mathbf{x}_i]$$

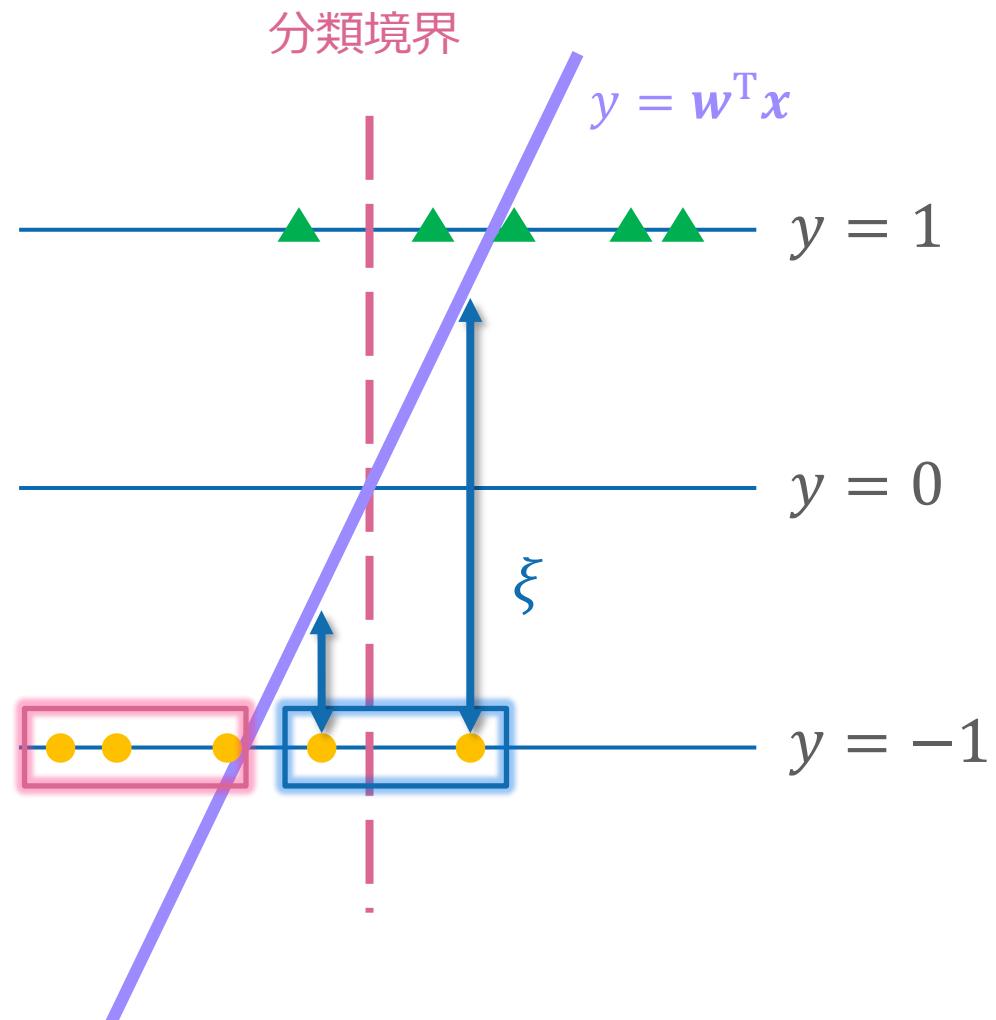
線形識別関数

$$t_i = \begin{cases} 1 & \text{データ } i = \text{クラス 1} \\ -1 & \text{データ } i = \text{クラス 0} \end{cases}$$

$t_i = 1$  のデータは  $x_i$  軸上で線形識別関数よりも右側  
 $\rightarrow \mathbf{w}^T \mathbf{x}_i \geq 1$  で、 $\xi_i = 0$

誤分類データは  $x_i$  軸上で線形識別関数よりも左側  
 $\rightarrow \mathbf{w}^T \mathbf{x}_i < 1$  で、 $\xi_i = 1 - \mathbf{w}^T \mathbf{x}_i$

## ■ $\xi$ の具体的なイメージ（クラス 0 の場合）



$$\xi_i = f(\underline{\mathbf{w}^T \mathbf{x}_i}) = \max[0, 1 - t_i \cdot \mathbf{w}^T \mathbf{x}_i]$$

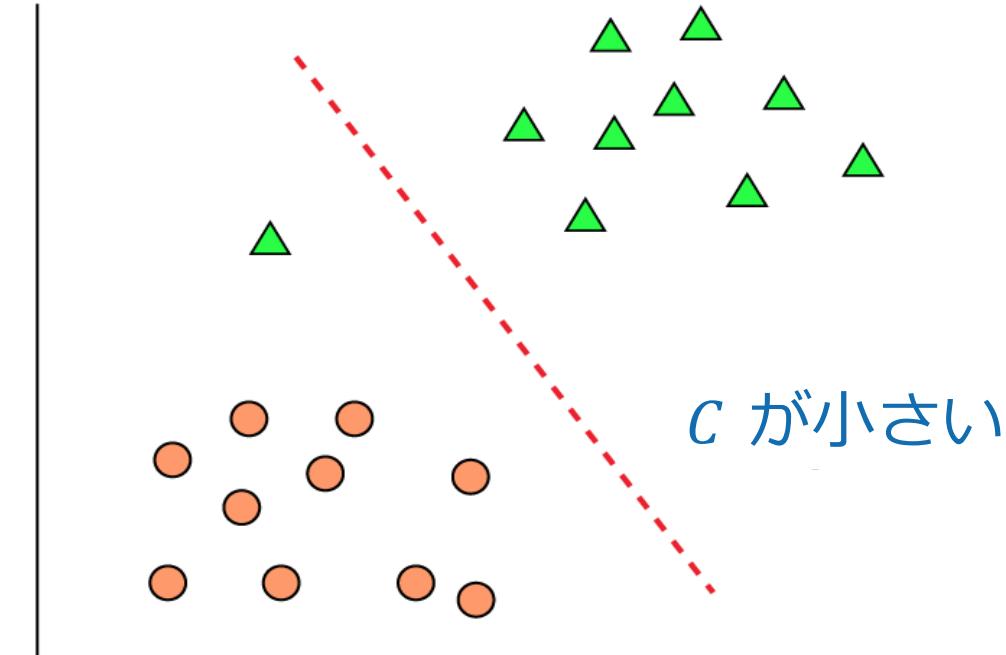
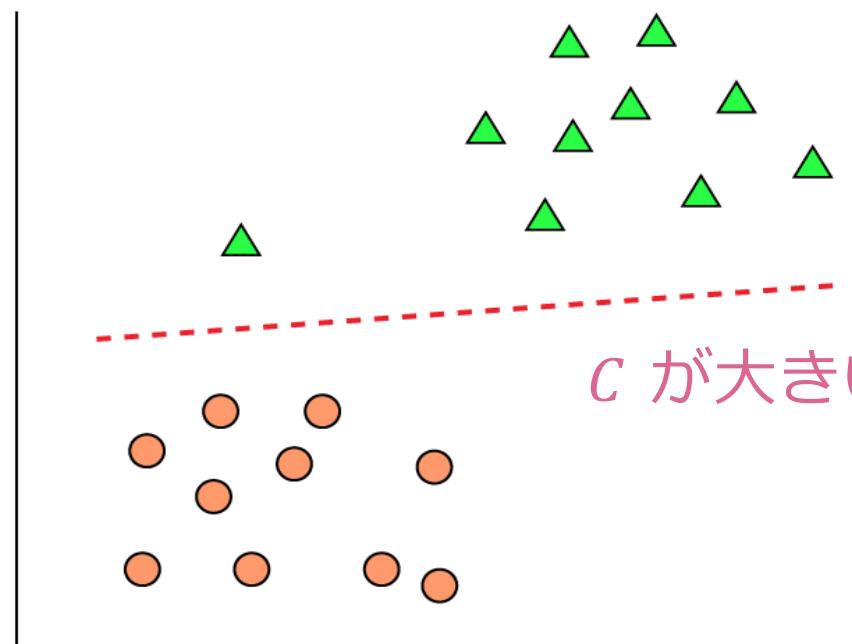
線形識別関数

$$t_i = \begin{cases} 1 & \text{データ } i = \text{クラス 1} \\ -1 & \text{データ } i = \text{クラス 0} \end{cases}$$

$t_i = -1$  のデータは  $x_i$  軸上で線形識別関数よりも左側  
 $\rightarrow \mathbf{w}^T \mathbf{x}_i \leq -1$  で、 $\xi_i = 0$

誤分類データは  $x_i$  軸上で線形識別関数よりも右側  
 $\rightarrow \mathbf{w}^T \mathbf{x}_i > -1$  で、 $\xi_i = 1 + \mathbf{w}^T \mathbf{x}_i = -(-1 - \mathbf{w}^T \mathbf{x}_i)$

- 最適化対象の  $\frac{1}{2}\|w\|_2^2 + C \left( \sum_{i=1}^N \xi_i \right)$  における  $C$  は、ペナルティ項の重みを決定するハイパーパラメータ
- $C$  が大きいと、少しの誤分類でも目的関数の値が大きくなる  
→ 誤分類を極力避けるようにマージンが決まる



## ■ 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を行う機械学習アルゴリズムと解釈可能

- ペナルティ項は、ヒンジ損失関数により線形識別関数のデータへの当てはまり具合を評価
- マージン最大化の項は、L2 正則化と同じ式

目的関数

$$\frac{1}{2} \|\mathbf{w}\|_2^2 + C \left( \sum_{i=1}^N \xi_i \right)$$

式の意味

目的関数全体を  
最小化する意味

線形識別関数の  
係数の総和

マージン最大化

誤分類による  
ペナルティ

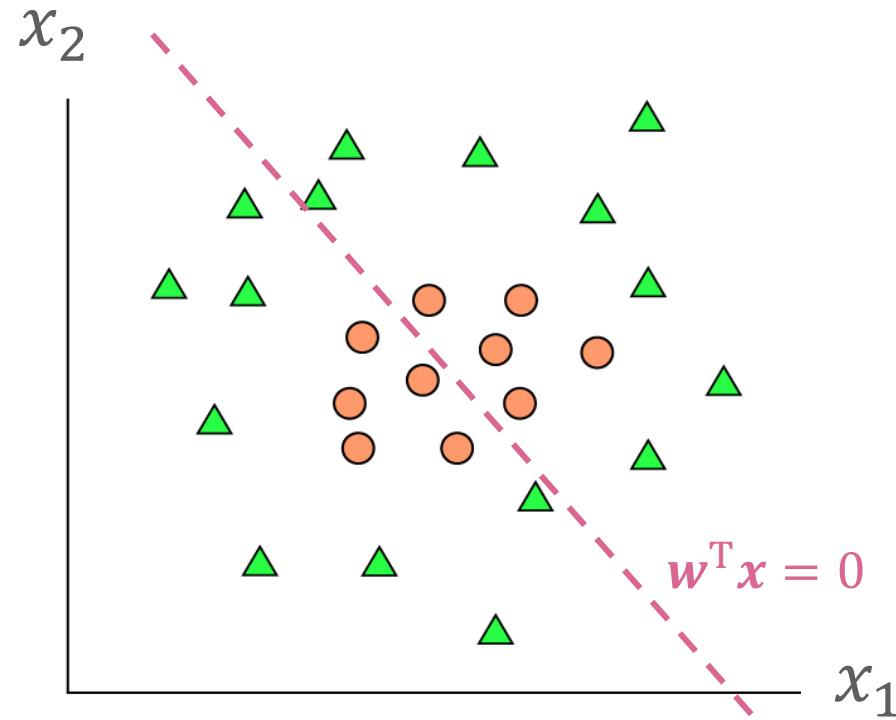
ヒンジ損失最小化

# カーネル法

---

# カーネル法 | SVM による非線形形データの分類

- オリジナルの SVM は直線的な境界 ( $w^T x = 0$ ) しか引くことができない
- 下記のようなデータに対しては上手く分類境界を引くことができない

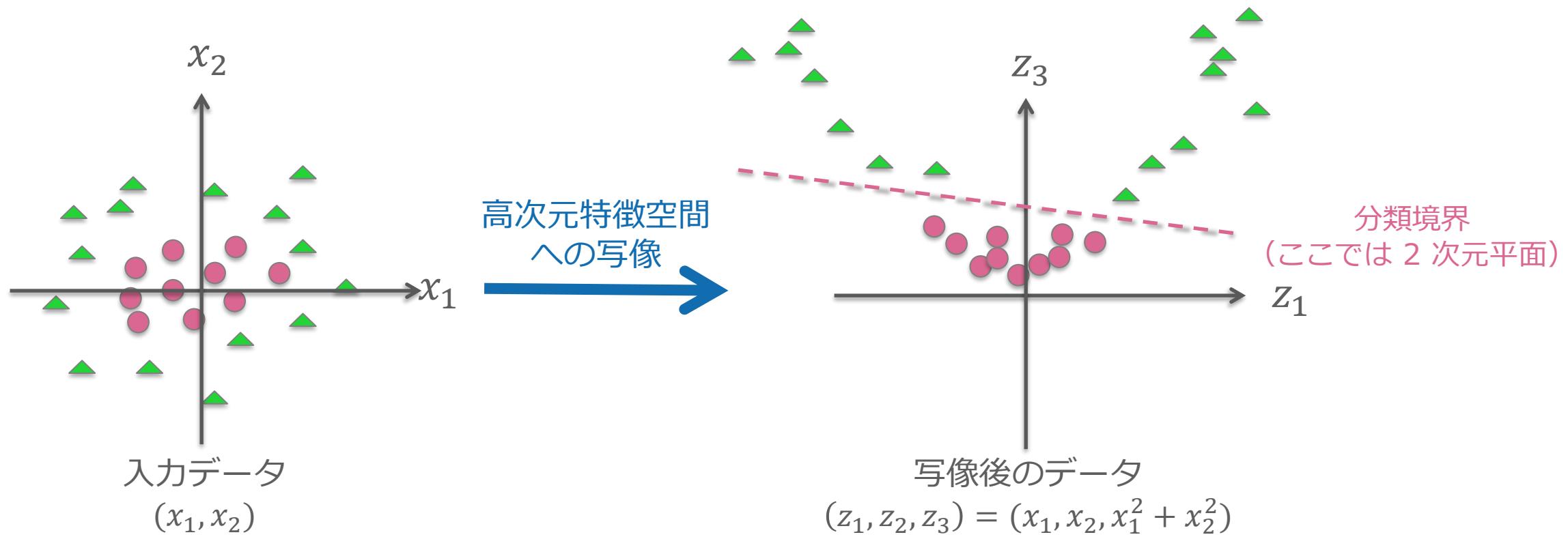


$x$  を高次元特徴空間に写像することで、上手く分類できる場合がある

- 一般に説明変数の次元数が大きいほど直線や平面で分離できる可能性が高くなる
- この性質を逆手に取り、低次元の入力データ（低次元の特徴空間）を高次元の特徴空間に写像し、高次元空間上での分離を求める

## ■ 高次元特徴空間への写像例

- 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
- 写像後の 3 次元特徴空間において、最大マージンとなる分類境界を求める



## ■ 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

- しかし  $(z_1, z_2, z_3) = (x_1, x_2, x_1^2 + x_2^2)$  という写像がいつも上手くいくとは限らない
- そもそも**良い変換**を見つけるというのは困難

## ■ そこで、良い変換をピンポイントで見つけるという考え方をやめ、「**変換を大量に用意する**」という総当たり的な考え方をする

- この考え方を適用するには 1 つ条件があるが、SVM の最適化問題はその条件を満たす
- その条件を説明するために SVM の最適化問題に少し深入りする

- SVM の最適化問題は不等式制約付き**凸最適化問題**と呼ばれる

最適化対象の目的関数  
(最小化)

$$\frac{1}{2} \|\mathbf{w}\|_2^2 + C \left( \sum_{i=1}^N \xi_i \right)$$

制約

$$t_i \cdot \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i \quad (i = 1, \dots, N)$$

- ラグランジュの未定乗数法を用いると、以下のような双対問題※に変換可能

最適化対象の目的関数  
(最大化)

$$L(\alpha) = \sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j t_i t_j x_i^T x_j$$

制約

$$\sum_{i=1}^N \alpha_i t_i = 0 \quad 0 \leq \alpha_i \leq C$$

最終的に、等式制約の下で目的関数を最大化する  $\alpha$  を求める問題に帰着

なお  $w = \sum_{i=1}^N \alpha_i t_i x_i$  である

※元の最適化問題（主問題）とは別の問題であるが、解を求めてることで主問題の解を同時に得られることが数学的に示されているもの

- 入力データ  $x$  を  $\phi(x)$  に変換することで、高次元空間に写像する
  - 例)  $x = (x_1, x_2)$  に対して  $\phi(x) = (x_1, x_2, x_1^2 + x_2^2)$
- $\phi(x)$  を入力とすると、SVM の目的関数は以下のようになる

$$\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j t_i t_j \phi(x_i)^T \phi(x_j)$$

- 高次元の入力  $\phi(x)$  を分析者が独自に設計することは困難
  - しかし目的関数をよく見ると、 $\phi(x_i)$  単体の値が必要というよりも  $\phi(x_i)^T \phi(x_j)$  の値 (= 変換後のデータの内積) が必要であるということに気づく
  - 実は一部の  $\phi$  については、 $\phi(x_i)^T$  と  $\phi(x_j)$  をそれぞれ計算しなくとも、内積  $\phi(x_i)^T \phi(x_j)$  の値を直接求めることができる (カーネルトリック)

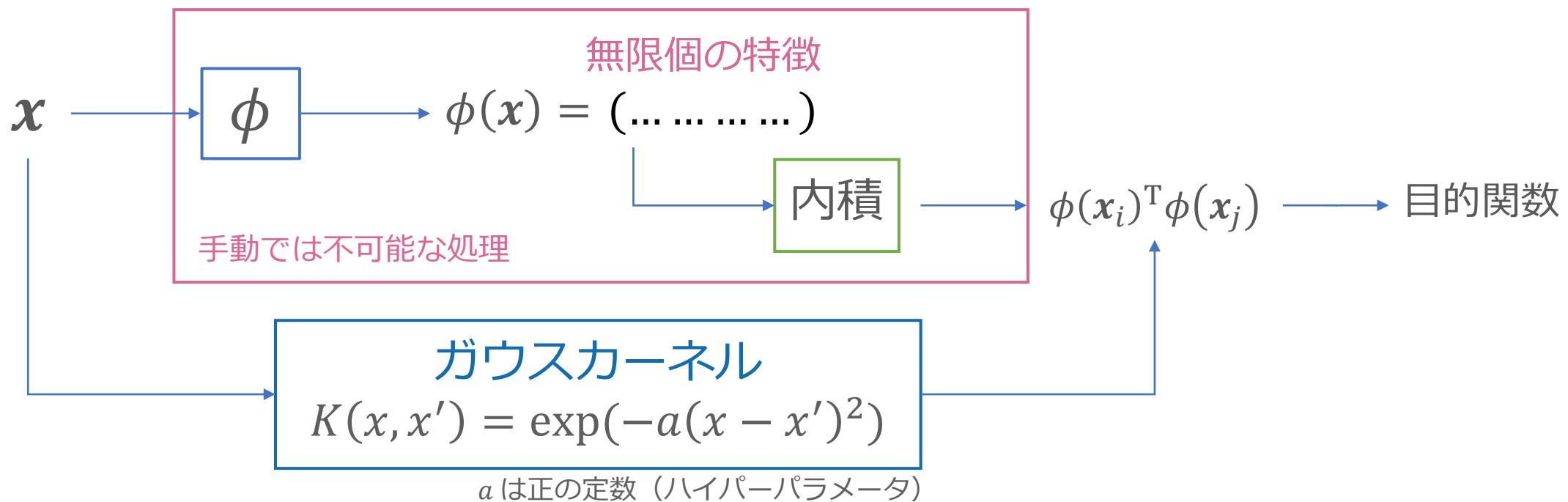
- ベクトル  $x_i, x_j$  を入力として  $\phi(x_i)$  と  $\phi(x_j)$  の内積の値を出力する関数を考える
  - $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$
  - この  $K$  をカーネル関数と呼ぶ
- カーネル関数  $K(x_i, x_j)$  を用いると、目的関数は以下のように変形できる

$$\sum_{i=1}^N \alpha_i - \frac{1}{2} \sum_{i=1}^N \sum_{j=1}^N \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

- もはや  $\phi$  を使う必要はない
- この  $K$  には大体の場合、ガウスカーネルと呼ばれる正規分布の数式に似た関数を用いると良い
- ではなぜ、ガウスカーネルが有効なのか？

## ■ カーネル関数にガウスカーネルを設定した場合

- 無数の変換を施した無限次元のベクトル  $\phi(x_i)$  を作ったことと同値になる
- その次元の中に、有用なものも含まれている可能性が高い
- 理論的には、無限次元空間で線形分離を試みることになる



- 説明変数が多く、データが少ない場合はカーネル法を用いない方が良い
  - ・ 過学習のリスクが大きいため
  - ・ SVM の実装の際には「線形カーネル」を指定 (= カーネル法なし)
- 説明変数が少なく、データ量をある程度確保可能であればガウスカーネルなどを利用してみると良い

		データサイズ	
		多い	少ない
説明変数	多い	×	×
	少ない	○ ガウスカーネル	×

## ■ カーネル法

- ガウスカーネルをはじめとした、様々なカーネル関数の性質に関する理論  
およびカーネル関数を使う方法論

## ■ カーネルトリック

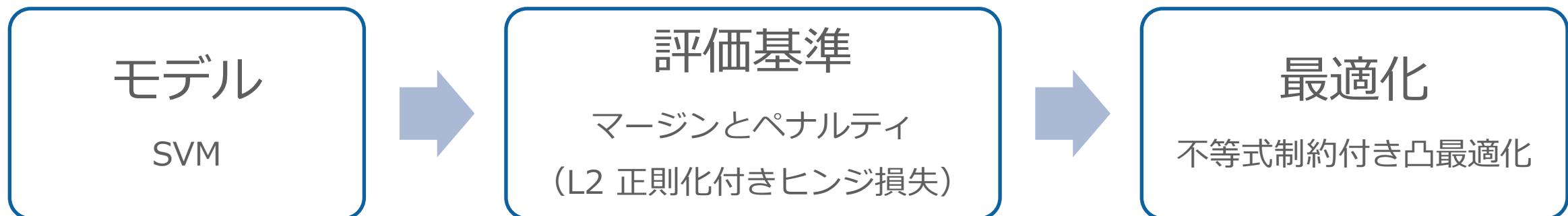
- カーネル関数を適用することにより、 $\phi(x_i)$  の計算なしで  
内積を直接求めることができる、という性質

## ■ カーネル法を用いることで、データを上手く分類できるようになる場合がある

- データを高次元特徴空間に写像する

## ■ サポートベクターマシン

- 主に 2 クラス分類を行うモデル
- マージンが最大となるような分類境界を求める
- ソフトマージン法の場合、誤分類を許容する
  - マージンを最大化しつつ、ペナルティを最小化する
- ガウスカーネルを利用すると、複雑な分類問題にも対処できる場合がある

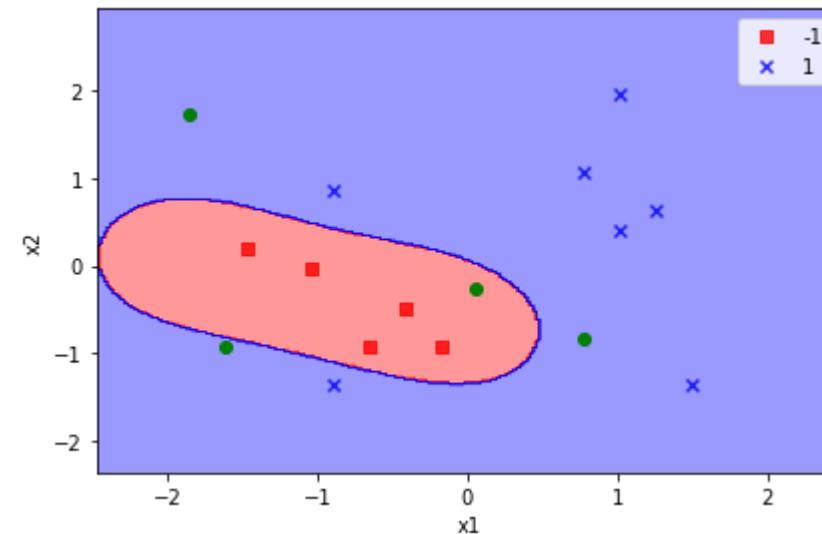
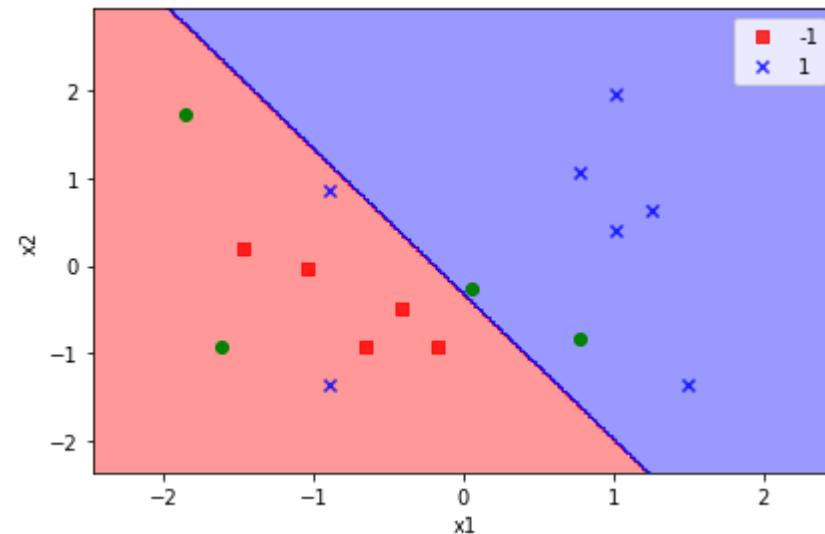


## ノートブック演習

---

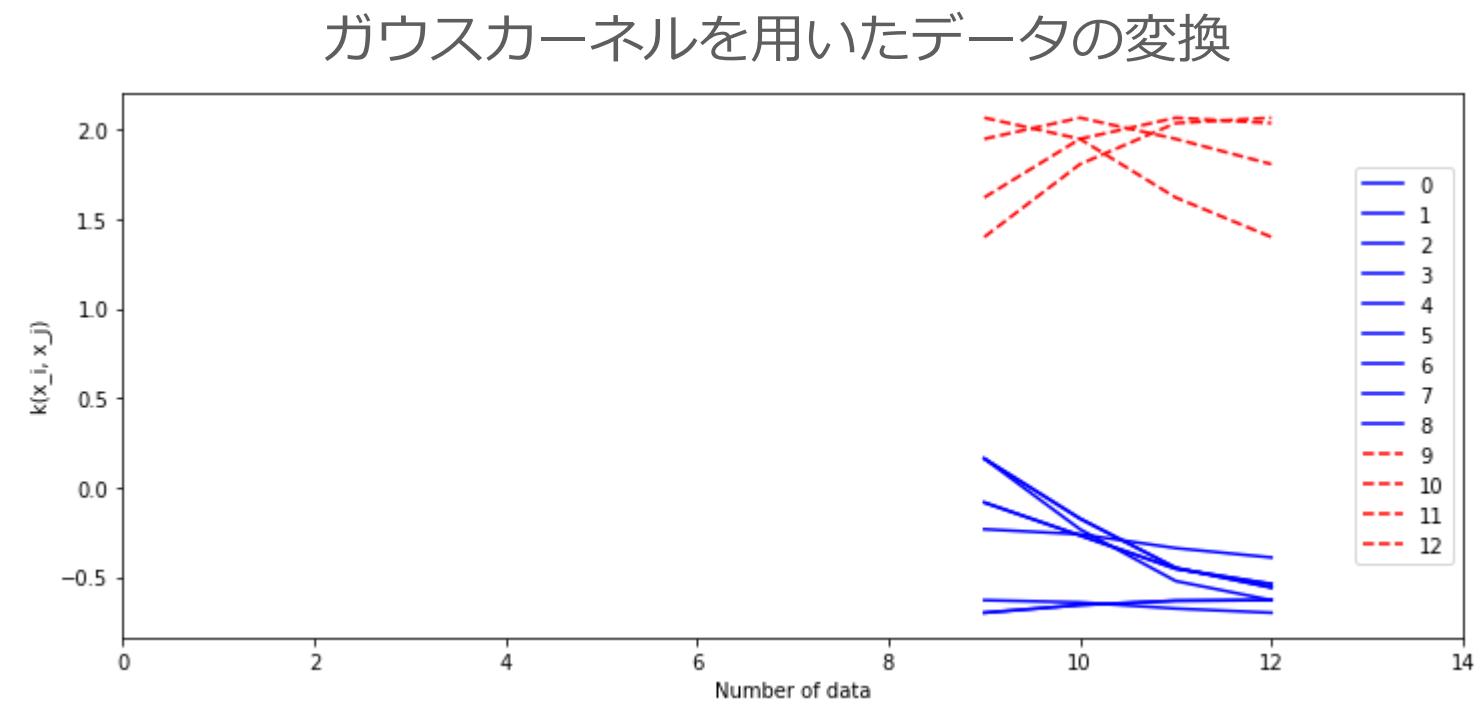
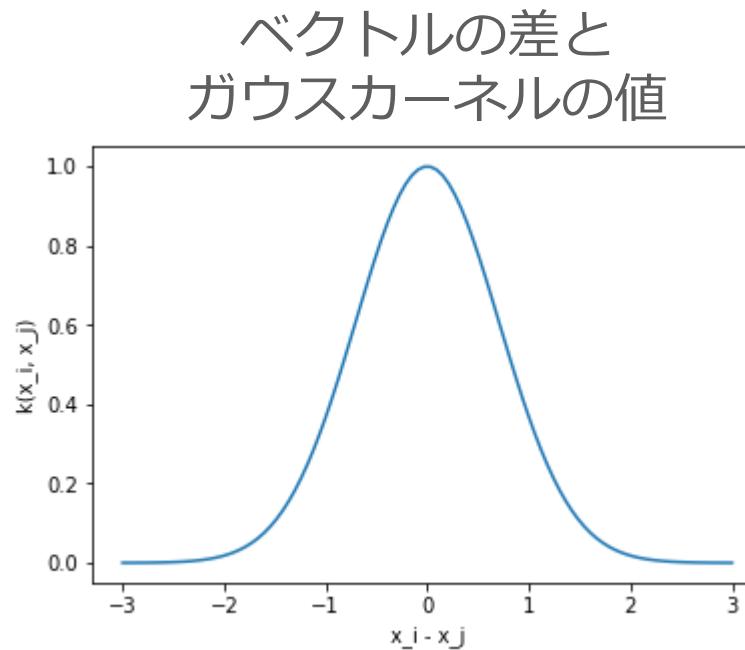
## 10-1\_SVM\_practice.ipynb

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



## 10-2\_SVM\_kernel\_function.ipynb

- 線形カーネル（単純な内積）とガウスカーネルの振る舞いを確認してみよう



## 本章のまとめ

---

- サポートベクターマシンの基本概念
- ハードマージン法
- ソフトマージン法
- カーネル法
- ノートブック演習

## ■ サポートベクターマシン

- 主に 2 クラス分類を行うモデル
- マージンが最大となる分類境界を求める
- ハードマージン法：誤分類を許容しない方法
- ソフトマージン法：ペナルティを設けて、誤分類を許容する方法
  - 目的関数は、L2 正則化付きのヒンジ損失最小化に相当

## ■ サポートベクターマシンのハイパーパラメータ $C$

- ペナルティ項の影響の強さを決める
- $C$  を大きくした場合、誤分類を極力避けるように分類境界を探索

## ■ カーネル法

- データを高次元空間上に変換して、分類しやすくする
- カーネルトリックを利用して、変換後データの内積を計算
- ガウスカーネルを使うことで、実質的に無限次元の特徴を作成できる

# 現場で使える 機械学習・データ分析基礎講座

第 11 章：深層学習の概要

- ニューロンとニューラルネットワーク
- パーセプトロン
- ニューラルネットワークの最適化
- ノートブック演習

- ニューロンの構成要素を説明できる
- ニューラルネットワークにおいて活性化関数を用いる意味を説明できる

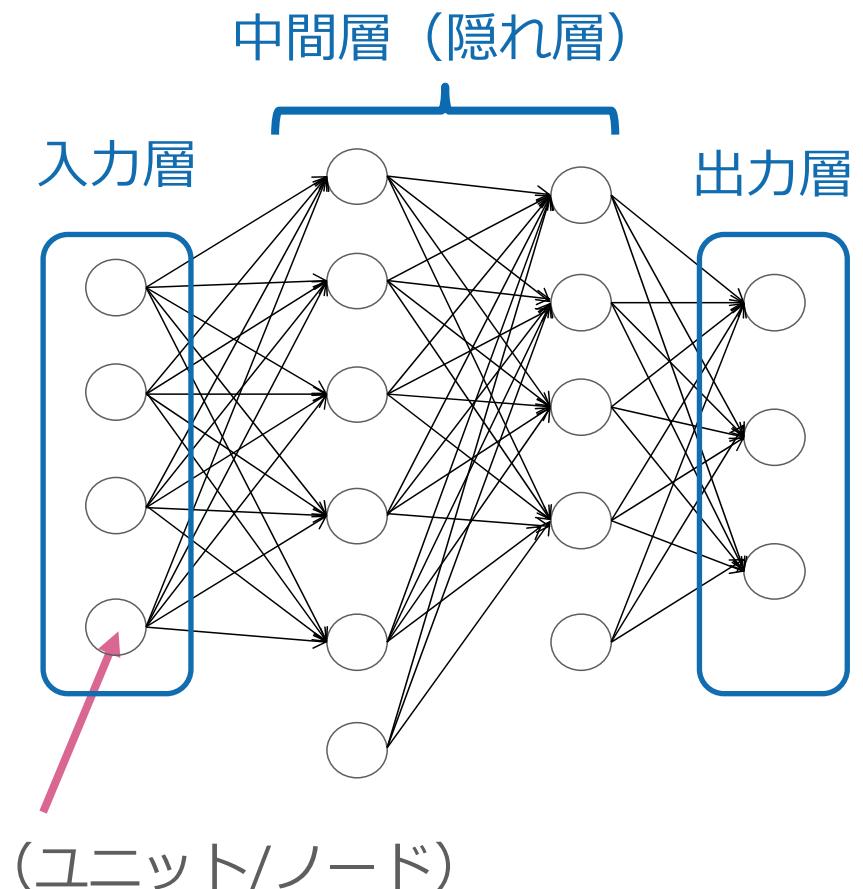
# ニューロンとニューラルネットワーク

## ■ 人間の脳内にある神経回路網を数式でモデル化したもの

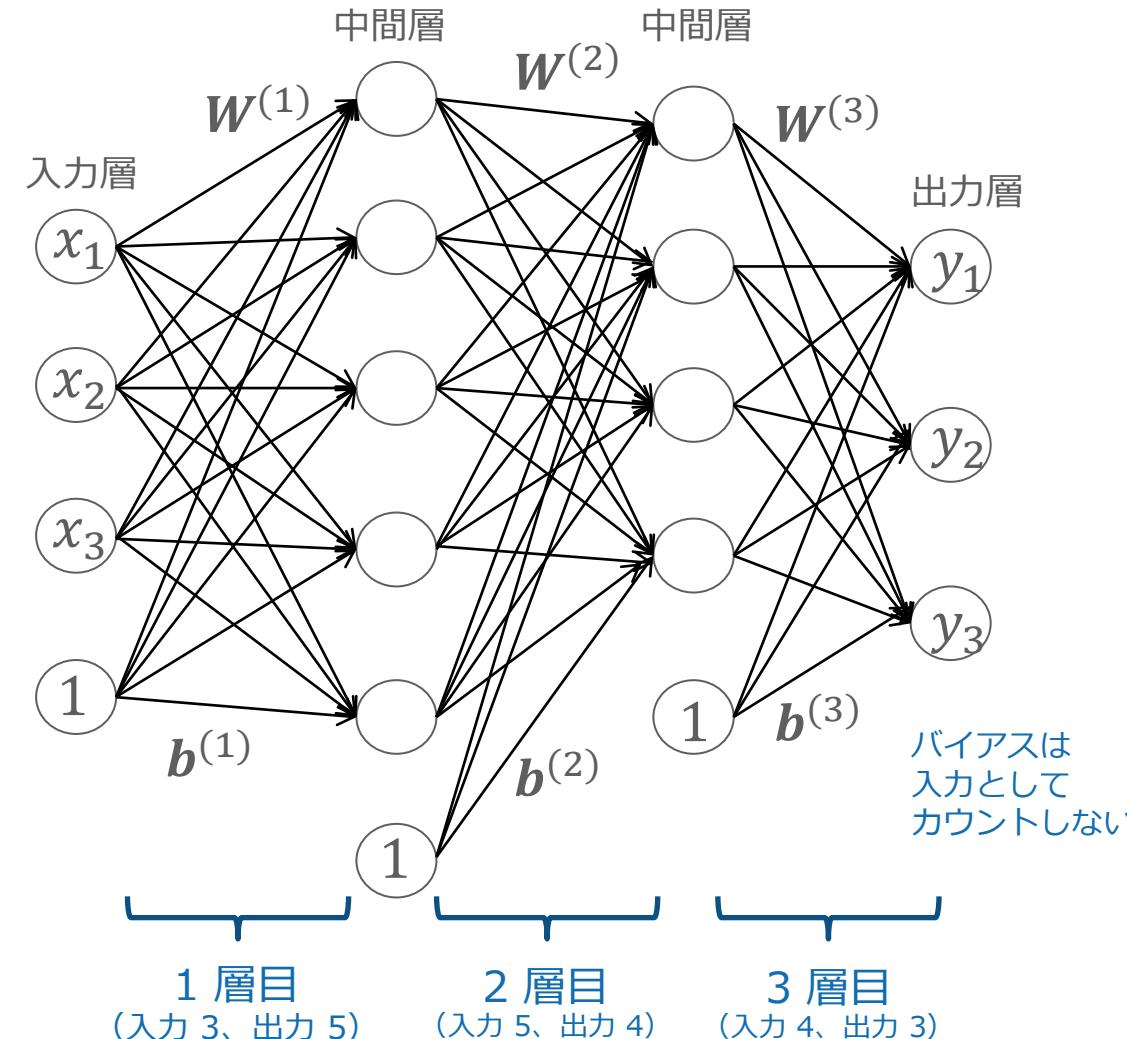
- ・ 神経細胞の活動をモデル化した**ニューロン**または**ユニット**と呼ばれる要素が**数多く結合**している
- ・ ユニット/ニューロンを**ノード**結合を**エッジ**と呼ぶこともある
- ・ ニューロンを数多く連結させることで複雑な入出力を形成

## ■ ニューロンの縦方向の並びを**層**と呼ぶ

- ・ プログラムの実装上は、ニューロン間の結合のまとまりを**層**と呼ぶことが多い



- プログラムの実装上は、ニューロン間の結合のまとまりを「層」として扱う



# ニューロンの構成要素

## ■ 入力 $x$

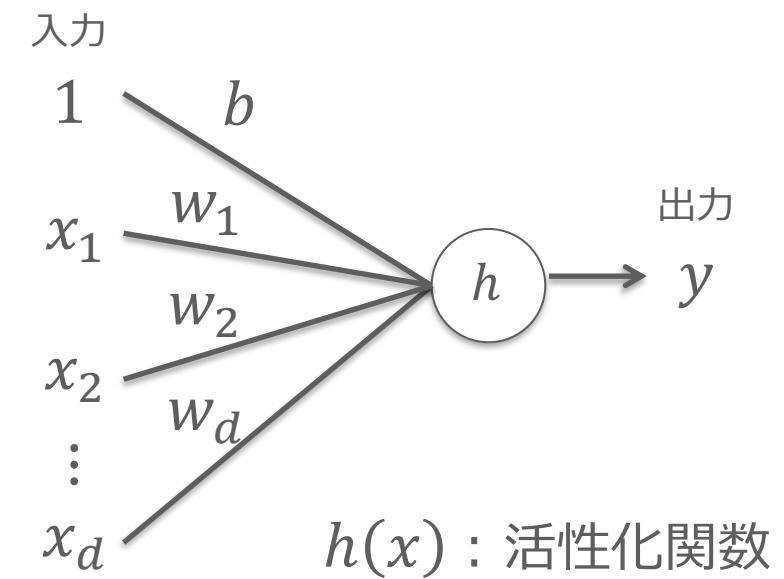
- ・ ニューロンに外から入力される値
- ・ 入力層のニューロンでは、既に中にあるとする

## ■ 重み $w$

- ・ 調整可能なパラメータ
- ・ 期待される入出力関係に近づくように調節

## ■ バイアス $b$

- ・ ダミーの入力 1 に対する重みパラメータ
- ・ 一次関数における切片の役割



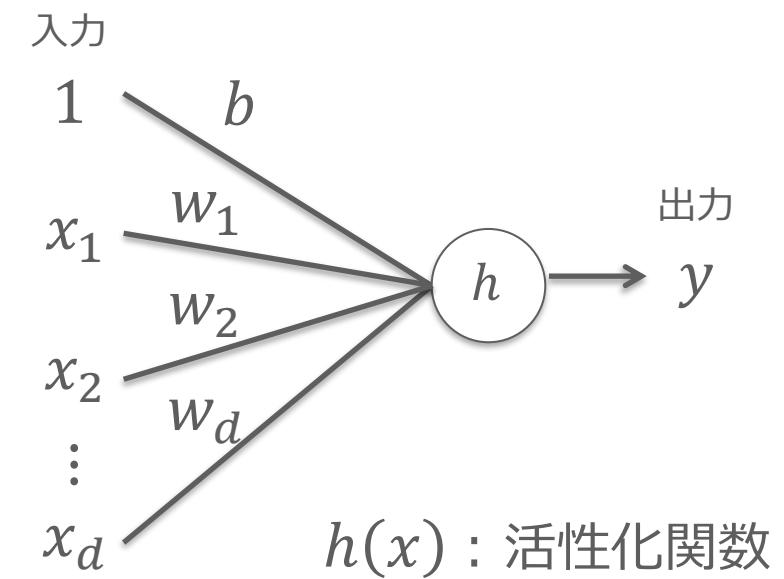
$$y = h \left( b + \sum_{i=1}^d w_i x_i \right)$$

## ■ 活性化関数 $h$

- ニューロンの入出力に複雑な動きを持たせるための関数

## ■ 出力 $y$

- ニューロンが外に出力する値
- 後続のニューロンに対する入力でもある



$$y = h \left( b + \sum_{i=1}^d w_i x_i \right)$$

## ■ 活性化関数を設定することの意味

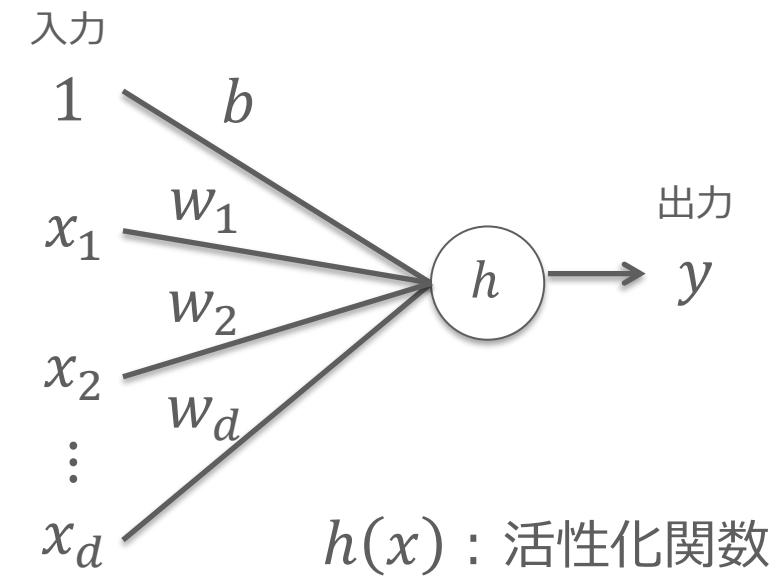
- ニューロンの入出力関係を複雑にする
- 非線形な関数を用いることが、難しい回帰や分類問題を解くポイント
- 複雑な入出力を行うニューロンを多数組み合わせて、大きな非線形モデルを構築

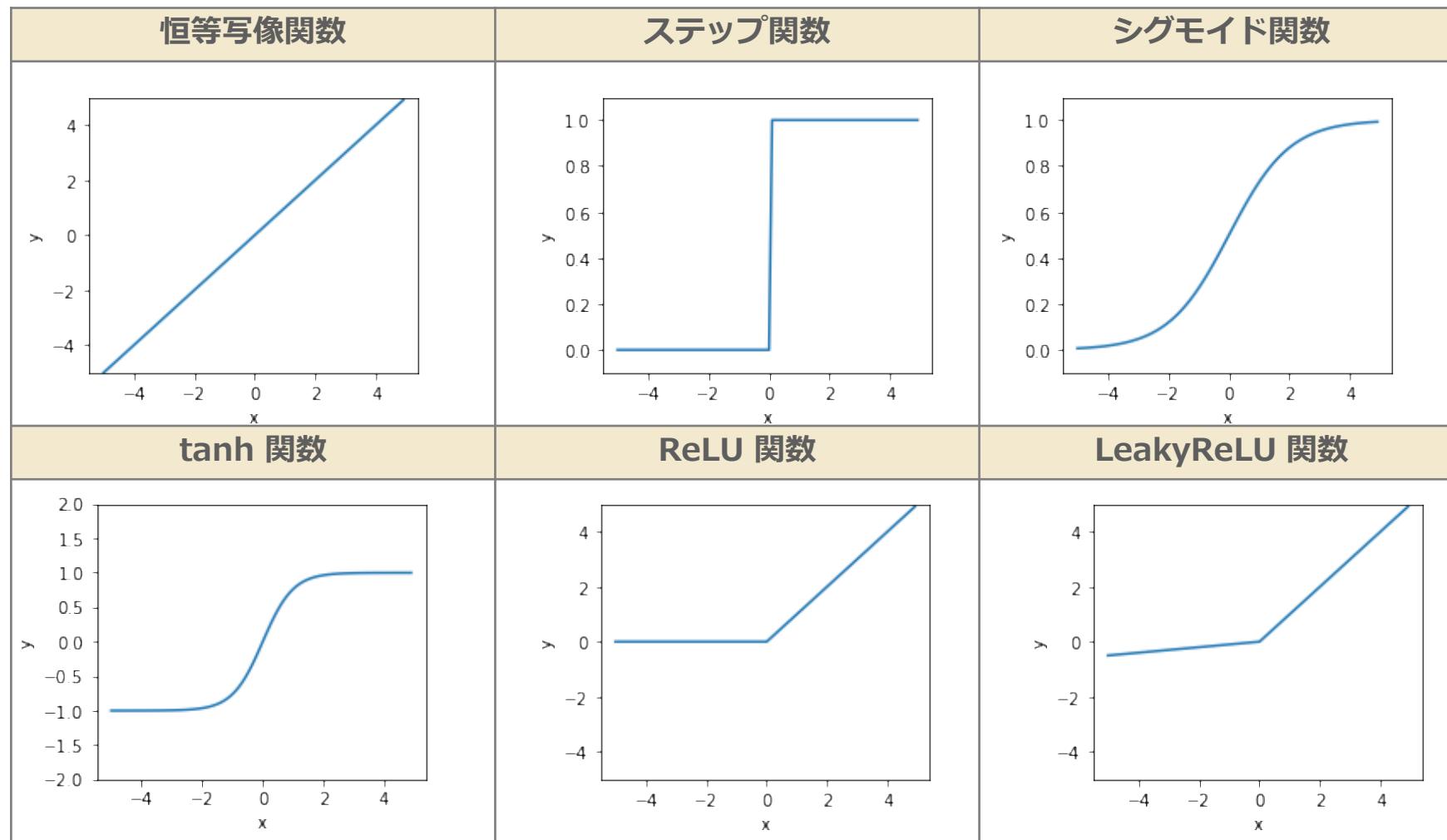
## ■ 活性化関数を用いない、もしくは恒等写像関数 $h(x) = x$ を用いた場合

- ニューロンは単純な線形モデルしか構成できない
- ニューラルネットワーク全体も線形モデルとなる

活性化関数を用いない場合

$$y = b + \sum_{i=1}^d w_i x_i$$



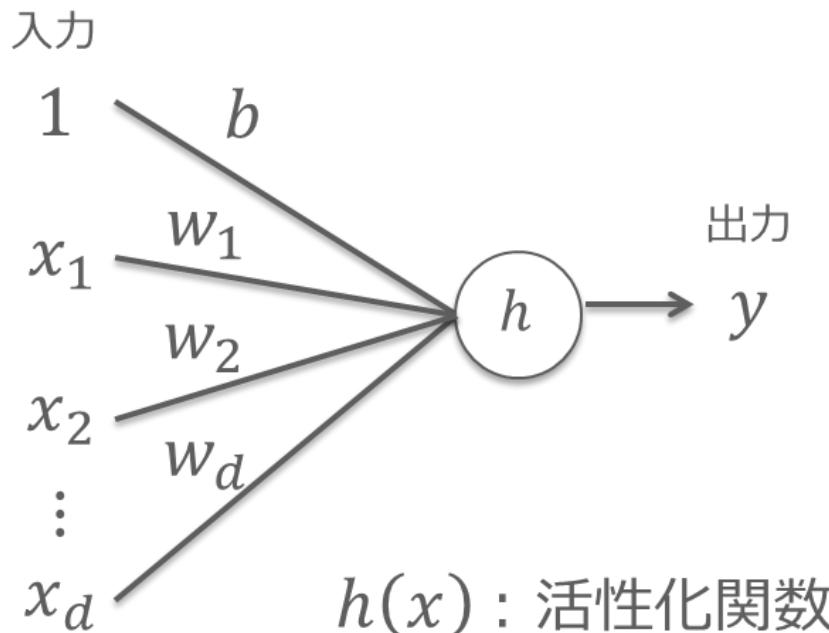


中間層には一般的に **ReLU** / **LeakyReLU** が用いられる  
 一部の NN (RNN など) ではシグモイド / tanh が用いられる

# パーセプトロン

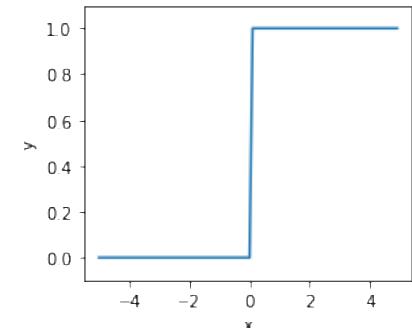
---

- 活性化関数にステップ関数を用いたニューロンに相当するもの
- 複数の信号を入力として受け取り、1つの信号を出力するアルゴリズム
  - 出力されるものは、信号を流す（1） / 流さない（0）の2値
  - 入力信号の重み付き和が、限界値（バイアス）を越えた場合にのみ 1 を出力



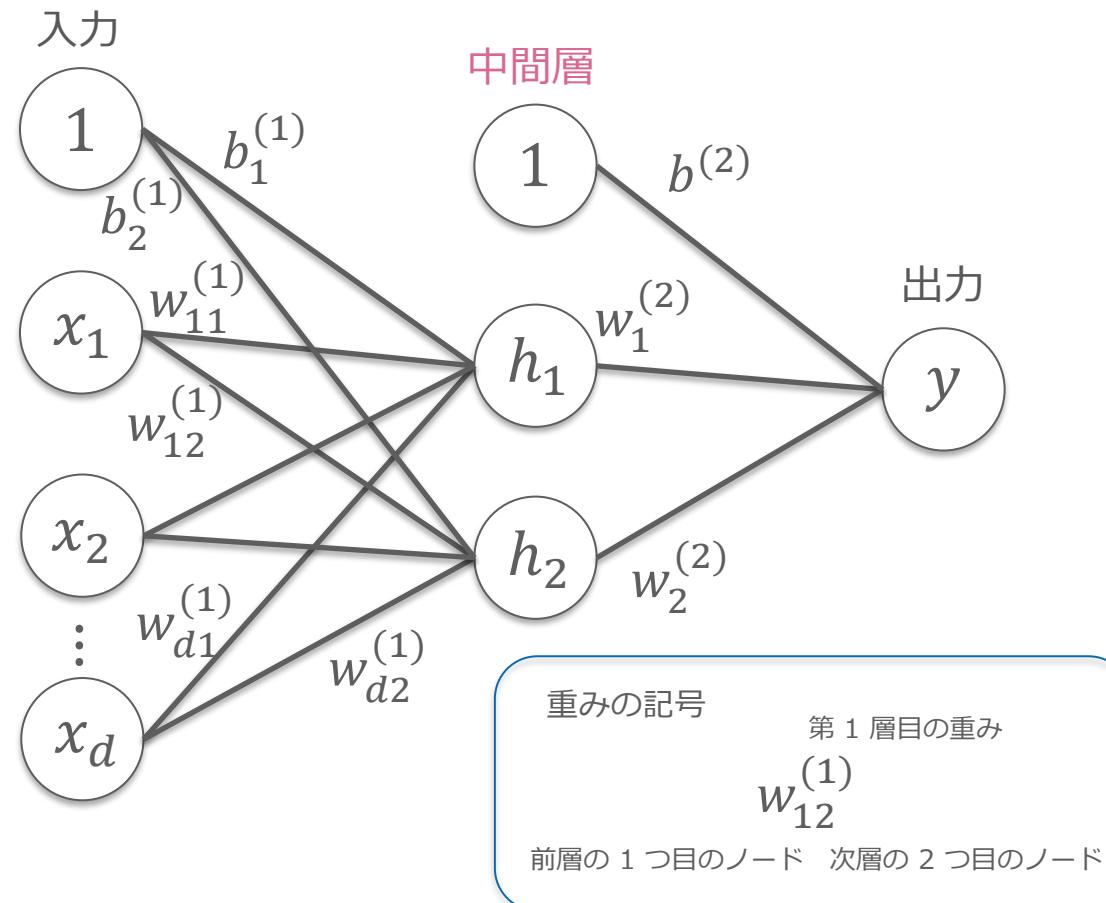
ステップ関数

$$h(x) = \begin{cases} 1 & (x \geq 0) \\ 0 & (x < 0) \end{cases}$$



$$y = \begin{cases} 1 & (\sum_{i=1}^d w_i x_i + b \geq 0) \\ 0 & (\sum_{i=1}^d w_i x_i + b < 0) \end{cases}$$

## ■ 入力層と出力層の間に中間層を持つパーセプトロン



$$y = h(b^{(2)} + \sum_{i=0}^2 w_i^{(2)} h_i)$$

$$h_1 = h(b_1^{(1)} + \sum_{i=1}^d w_{i1}^{(1)} x_i)$$

$$h_2 = h(b_2^{(1)} + \sum_{i=1}^d w_{i2}^{(1)} x_i)$$

$h()$  : 活性化関数

## ■ (全結合型) ニューラルネットワーク

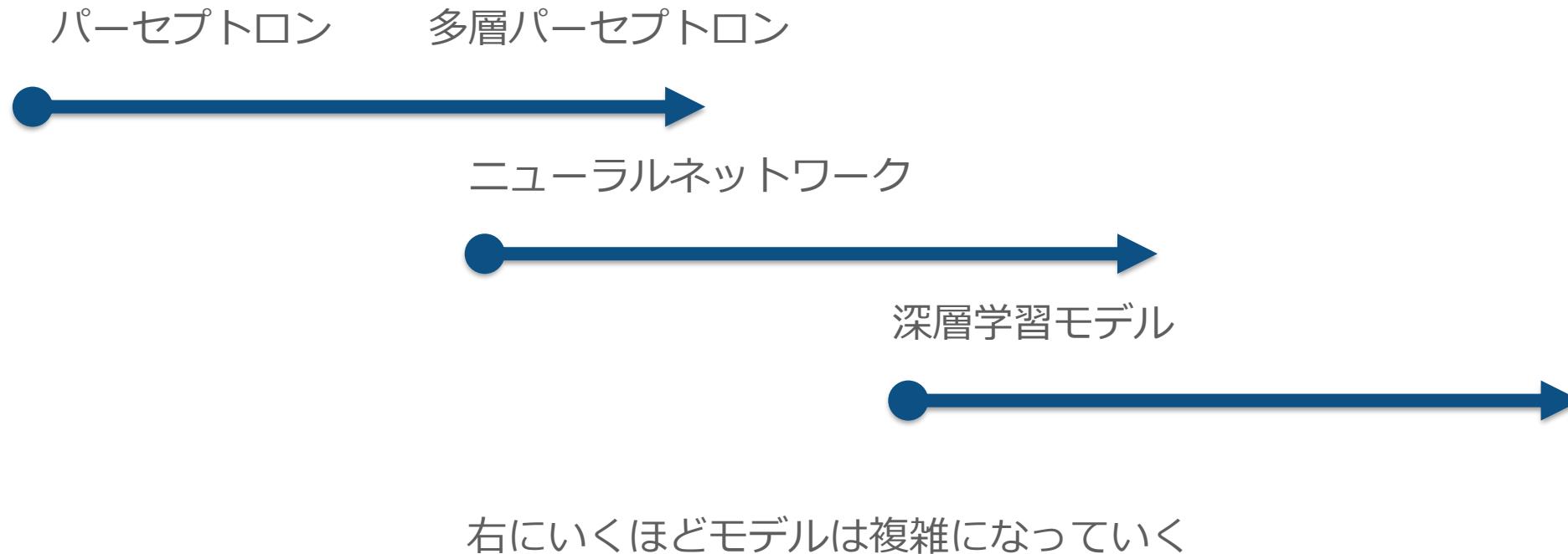
- 一般的な定義は以下の通り
- 多層パーセプトロンにおいて、活性化関数を  
ReLU 関数やシグモイド関数などの複雑な関数に設定したもの

## ■ ディープニューラルネットワーク (Deep Neural Network; DNN)

- 層の数を増やしたニューラルネットワーク
- 深層学習モデル※とも呼ばれる

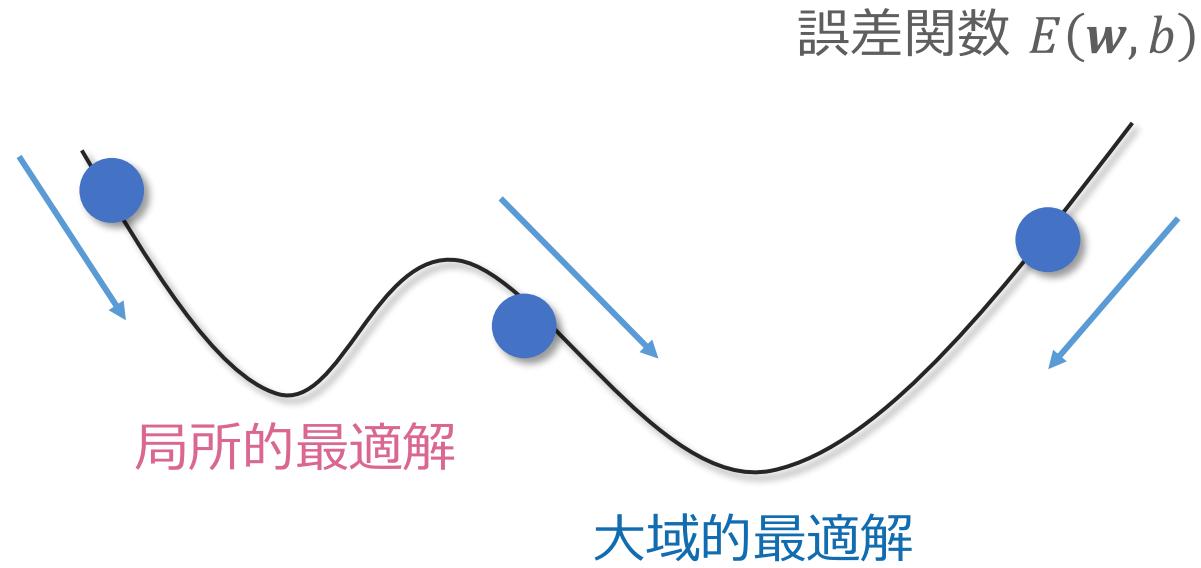
※厳密には、DNN は深層学習の方法論のひとつ

- 層の数に関して、多層パーセプトロン、ニューラルネットワークそして深層学習モデルの境界はあいまい
- 「〇〇層からは深層学習モデル」といった明確な定義はない



## ニューラルネットワークの最適化

- 回帰の場合には二乗誤差関数を設定
- 分類の場合には交差エントロピーを設定
- パラメータの最適化には勾配降下法を用いて、誤差を最小にする重み  $w$  とバイアス  $b$  を探索

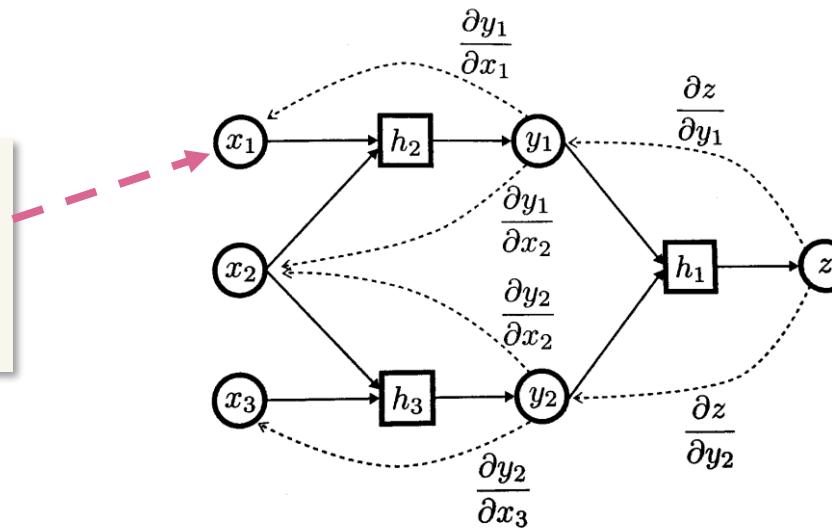


$E(w, b)$  の勾配情報を利用して  
極小値（局所的最適解）を探索

ニューラルネットワークの誤差関数は一般的に非凸関数であり、勾配法を用いて大域的最適解が得られる保証はない

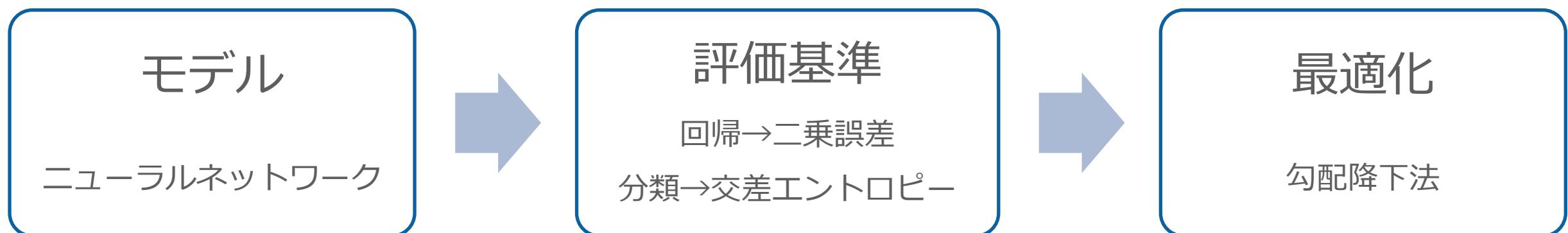
- 勾配降下法で極小値を探索するためには、重み  $w$  とバイアス  $b$  における誤差関数の偏微分値が必要
- 偏微分値を獲得するために誤差逆伝播法というアルゴリズムを使用
  - ・ ニューラルネットワークは単なる合成関数として見ることができる
  - ・ 合成関数の微分法  $\frac{df}{dx} = \frac{df}{dt} \frac{dt}{dx}$  を応用して、偏微分値を計算するために必要な情報を効率良く得ることができる

$$\frac{\partial z}{\partial x_1} = \frac{\partial z}{\partial y_1} \frac{\partial y_1}{\partial x_1}$$



## ■ ニューラルネットワーク

- ・ パーセプトロンを複数かつ多層に重ねたモデル
- ・ 活性化関数の導入により、複雑な入出力関係を持った関数を構成できる

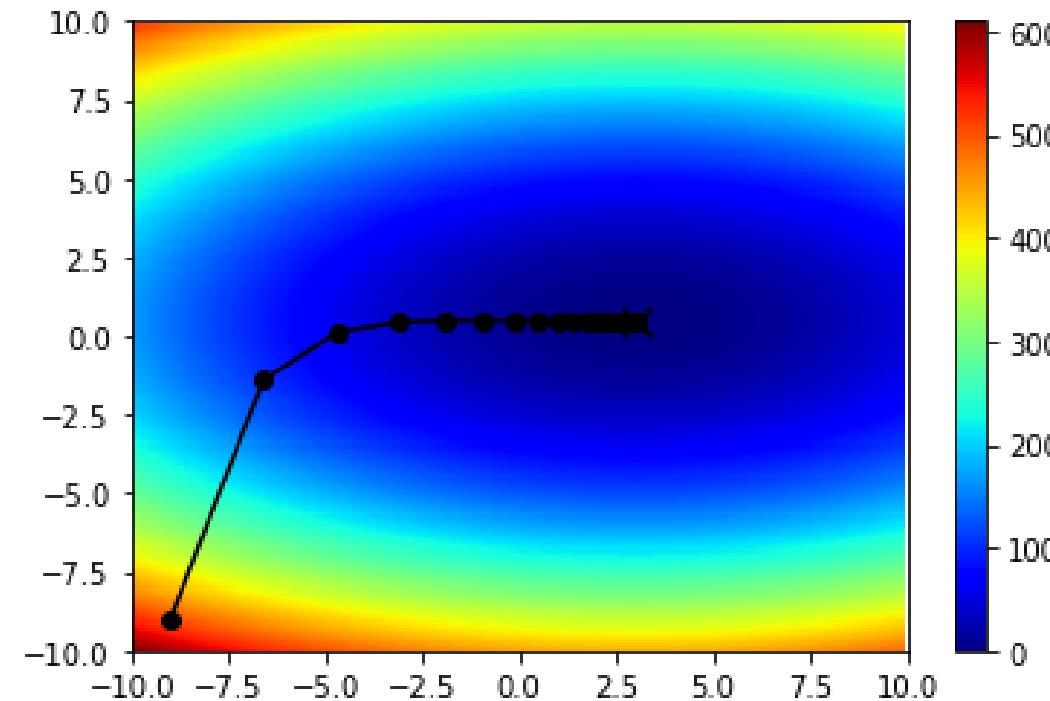


## ノートブック演習

---

## 11-1\_gradient\_descent.ipynb

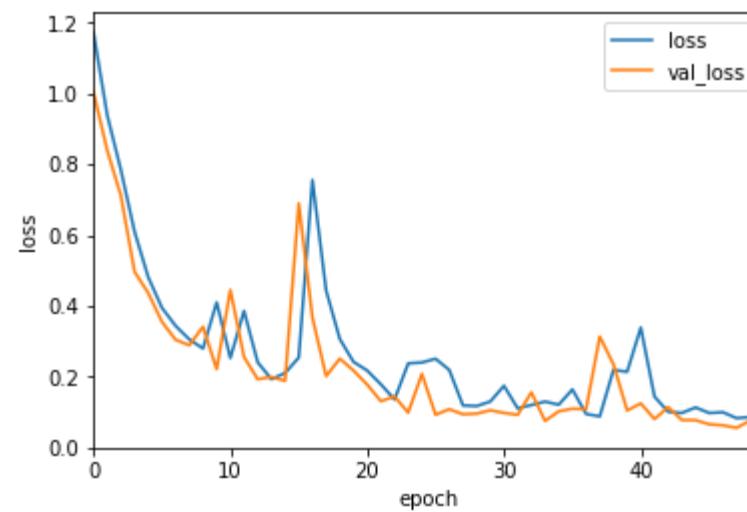
- 勾配降下法の振る舞いを、実際のプログラムで確認しましょう
- 初期値や学習率を変更したとき、どのような変化が起こるか確認してみよう



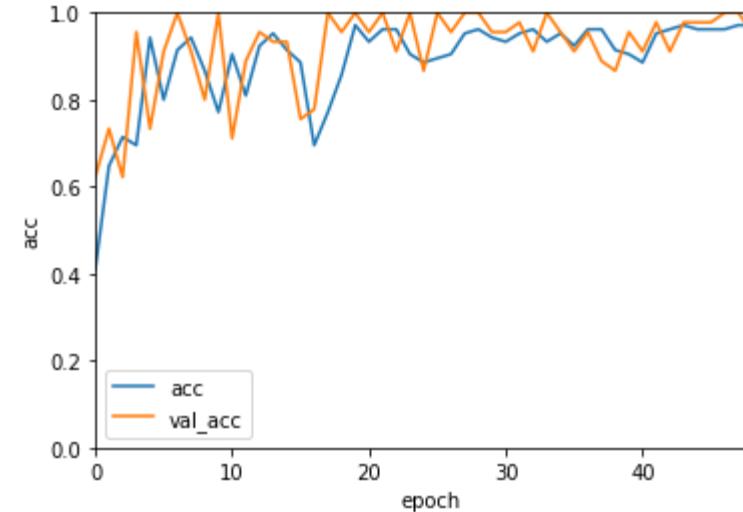
## 11-2\_NN\_practice.ipynb

- ニューラルネットワークを実装するのに便利なライブラリ TensorFlow & Keras を用いて、アヤメの分類を行ってみましょう
- ニューラルネットワークの学習に関するハイパーパラメータを変更し、変化を確認してみよう

評価基準（損失関数）

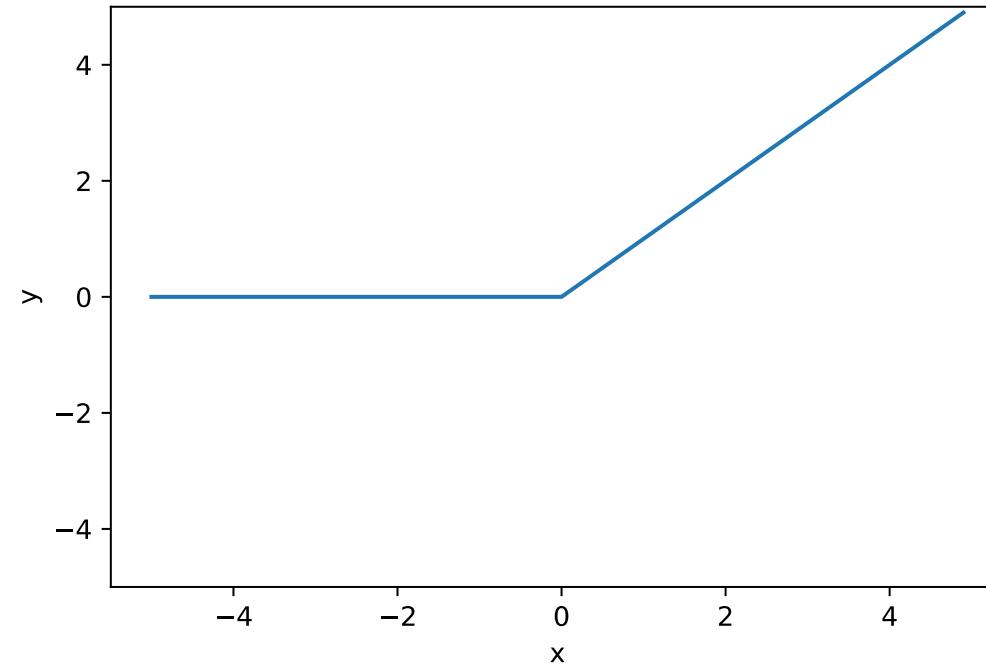


評価指標 (Accuracy)



## 11-3\_activate\_function\_trainee.ipynb

- 活性化関数の実装とその形状を確認してみましょう
- 活性化関数として最もよく用いられる ReLU 関数を実装してみよう



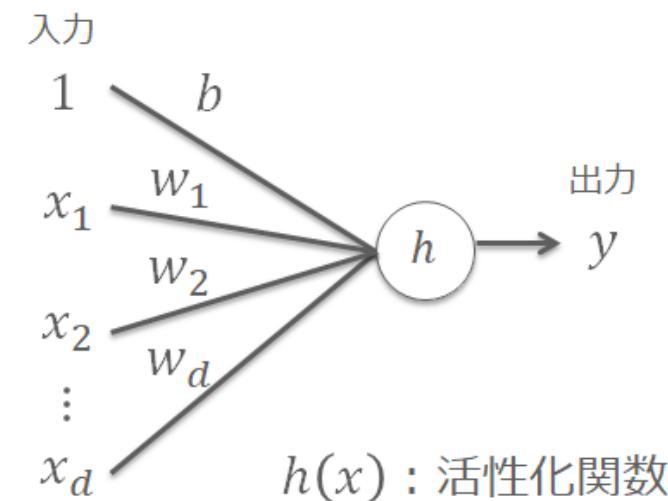
## 本章のまとめ

---

- ニューロンとニューラルネットワーク
- パーセプトロン
- ニューラルネットワークの最適化
- ノートブック演習

## ■ ニューロンの構成要素

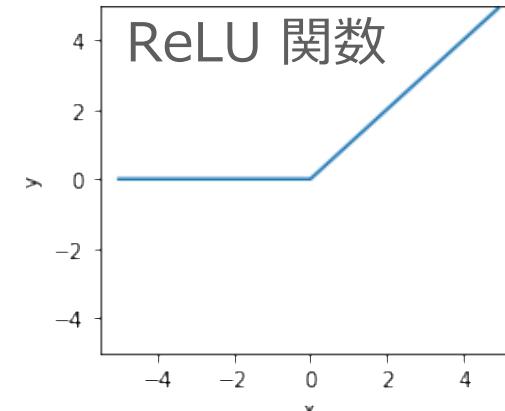
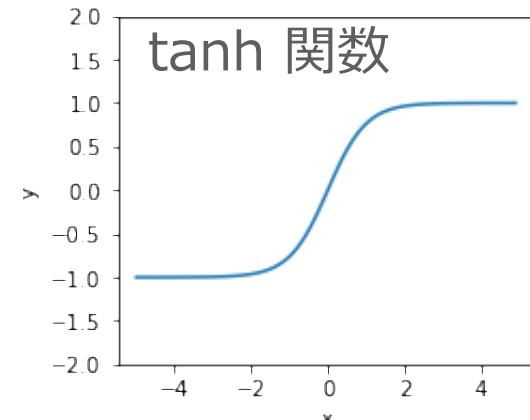
- 入力
- 重み、バイアス
- 活性化関数
- 出力



$$y = h\left(b + \sum_{i=1}^d w_i x_i\right)$$

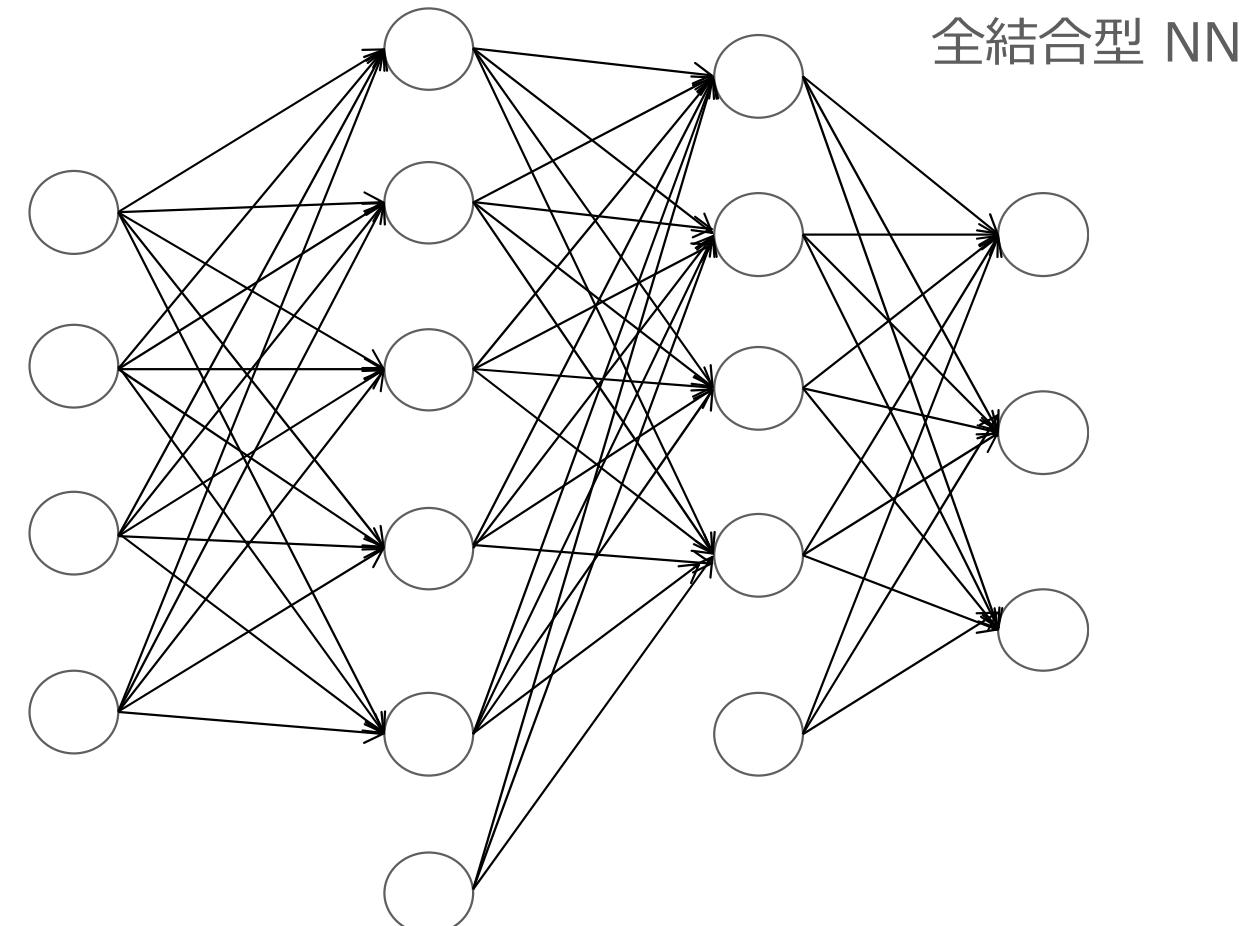
## ■ 活性化関数

- 非線形な関数を用いることで、複雑な入出力関係を捉えやすくする
- 一般的に ReLU や tanh が用いられる



## ■ 全結合型ニューラルネットワーク（全結合型 NN）

- ニューロンの入出力を数多く連結したパーセプトロン



## ■ ニューラルネットワークの最適化

- 勾配降下法を用いて、誤差関数の極小値を探索
- 誤差逆伝播法を用いて、多数の偏微分値を効率よく計算

# 現場で使える 機械学習・データ分析基礎講座

第 12 章：畳み込みニューラルネットワーク

- 置み込みニューラルネットワーク
- 置み込みとプーリング
- 著名な CNN モデル
- CNN の応用例
- ノートブック演習

- 置み込みニューラルネットワーク (CNN) の構成を説明できる
- CNN に関するタスクの例を説明できる

# 畳み込みニューラルネットワーク

## 決定的

### 階層型

全結合型ニューラルネットワーク  
Fully connected neural network

畳み込みニューラルネットワーク  
Convolutional neural network

再帰型ニューラルネットワーク  
Recurrent neural network

### 自己符号化器型

自己符号化器  
Autoencoder

雑音除去自己符号化器  
Denoising Autoencoder

変分自己符号化器  
Variational Autoencoder

スパース自己符号化器  
Sparse Autoencoder

## 確率的

### ボルツマンマシン型

ボルツマンマシン  
Boltzmann machine

制約ボルツマンマシン  
Restricted Boltzmann machine

深層信念ネットワーク  
Deep belief network

深層ボルツマンマシン  
Deep Boltzmann machine

- 「深層学習は、画像処理分野の発展に大きく貢献した」と言われることが多い
- どのような工夫があったのだろうか？
- 特徴量に着目して考えてみよう
  - 画像に特有であり、かつ、有効そうな特徴量とは何だろう？
  - それは、どのような処理を行えば得られるもの？



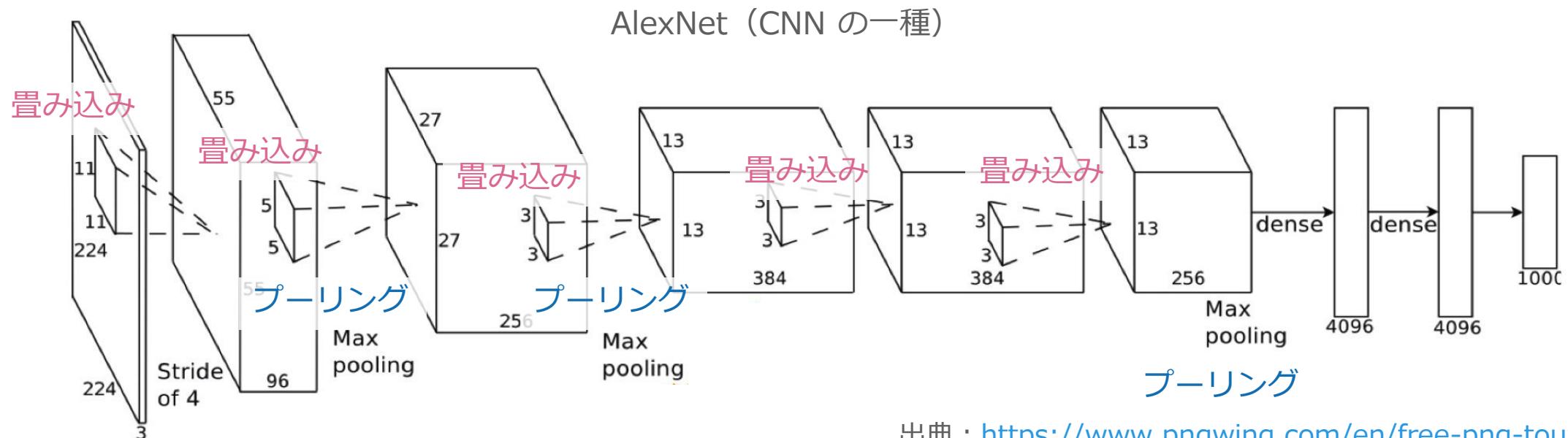
この画像から「トラ」であることを認識するためにはどのような特徴が必要？

## ■ 置み込み演算を取り入れたニューラルネットワーク

- CNN は *Convolutional Neural Network* の略

## ■ 置み込み層とプーリング層を用いて、画像から効果的な特徴を抽出

- 脳の視覚野に関する神経科学の知見をヒントにしている
- 置み込みに使うフィルタ（カーネルと呼ばれることがある）の値が学習対象のパラメータ



出典：<https://www.pngwing.com/en/free-png-touzp>

## ■ 画像のエッジ（輪郭）抽出などに用いられる処理

- ・ エッジは、画像分類において有用な特徴量の一つ

## ■ 処理手順

1. データにフィルタを重ねて要素ごとの積を取り、その合計を求める
2. フィルタを重ねる位置をずらしながら、画像全体に対して繰り返す

1	2	3	0
0	1	2	3
3	0	1	2
2	3	0	1

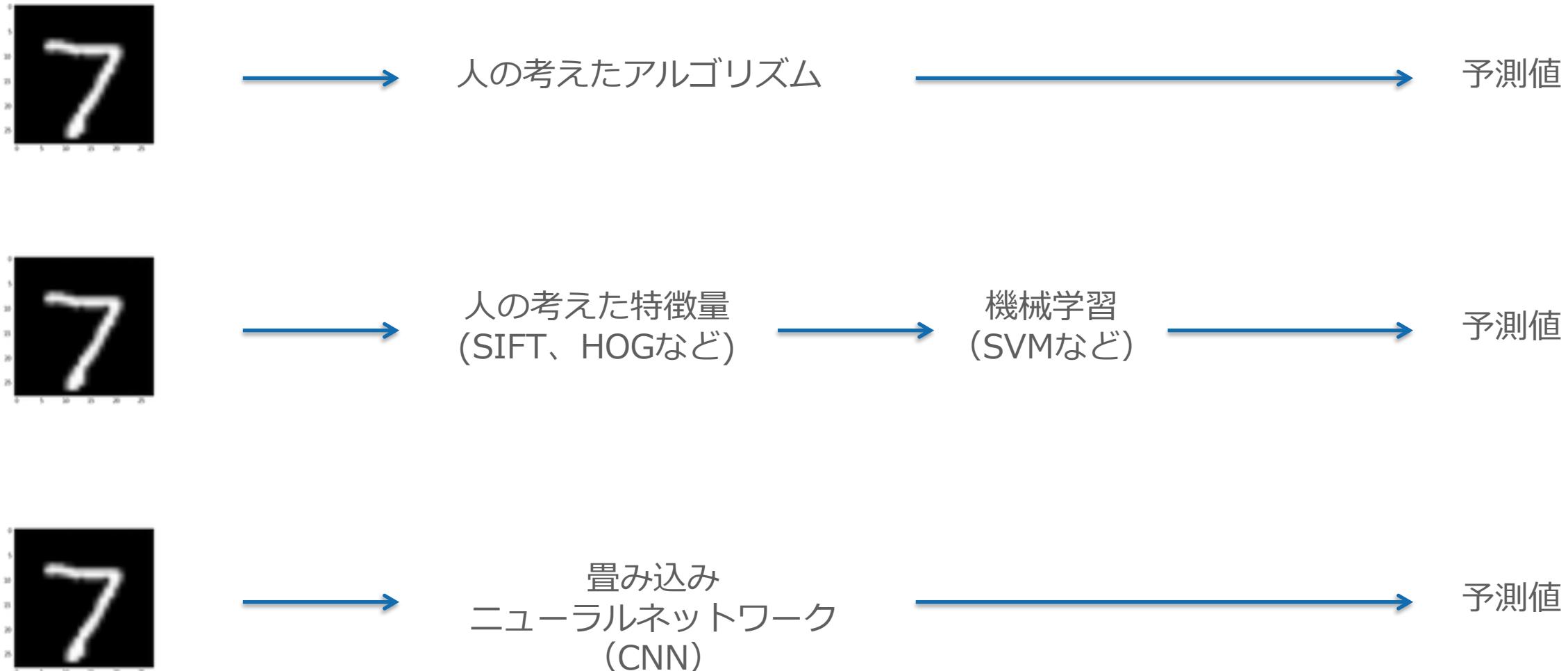
\* フィルタ

$$\begin{matrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{matrix} \rightarrow \begin{matrix} -2 & -2 \\ 2 & -2 \end{matrix}$$

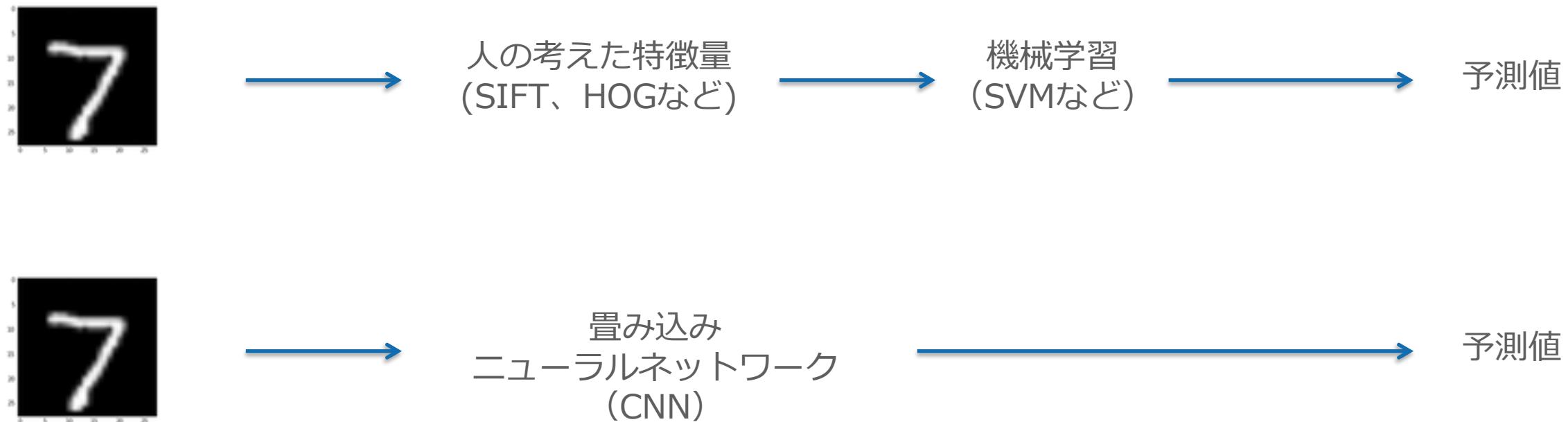
\* : 置み込み演算



# CNN | 画像分類における従来手法と CNN の違い



- CNN は、他の機械学習手法と異なり、**特徴量の設計を自動で行っているもの**と見ることができる



## 決定的

### 階層型

全結合型ニューラルネットワーク  
Fully connected neural network

畳み込みニューラルネットワーク  
Convolutional neural network

再帰型ニューラルネットワーク  
Recurrent neural network

### 自己符号化器型

自己符号化器  
Autoencoder

雑音除去自己符号化器  
Denoising Autoencoder

変分自己符号化器  
Variational Autoencoder

スパース自己符号化器  
Sparse Autoencoder

## 確率的

### ボルツマンマシン型

ボルツマンマシン  
Boltzmann machine

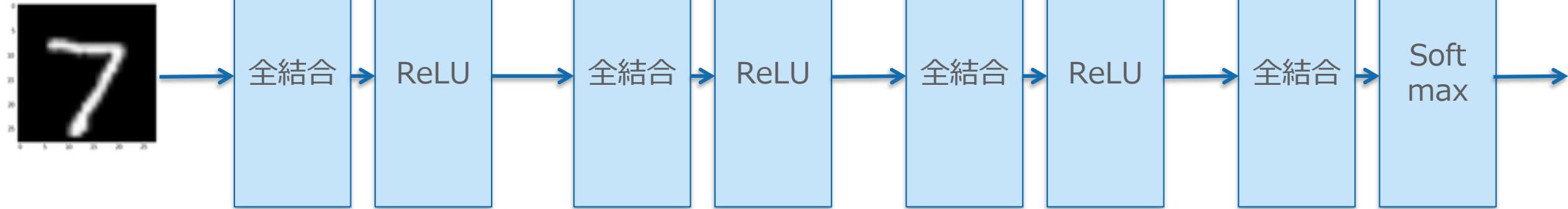
制約ボルツマンマシン  
Restricted Boltzmann machine

深層信念ネットワーク  
Deep belief network

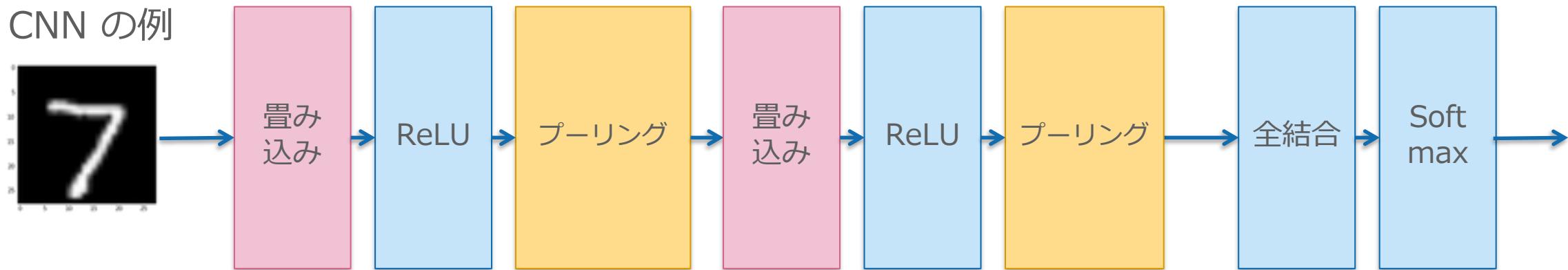
深層ボルツマンマシン  
Deep Boltzmann machine

参考：『深層学習』(神鳶ら、近代科学社)

全結合 NN の例



CNN の例



- ・「畠み込み・プーリング」はセットにする必要はない
- ・「畠み込み → ReLU → 畠み込み → ReLU → プーリング」のように複数回の畠み込みを行った上で 1 回だけプーリングする構成でも OK
- ・同様に全結合層も、1 つだけでなく複数用いててもよい

## 畳み込みとプーリング

## ■ 以下の手順で畳み込み演算を行う

1. データにフィルタを重ねて  
要素ごとの積を取り、その合計を求める
2. フィルタを重ねる位置をずらしながら、  
画像全体に対して繰り返す

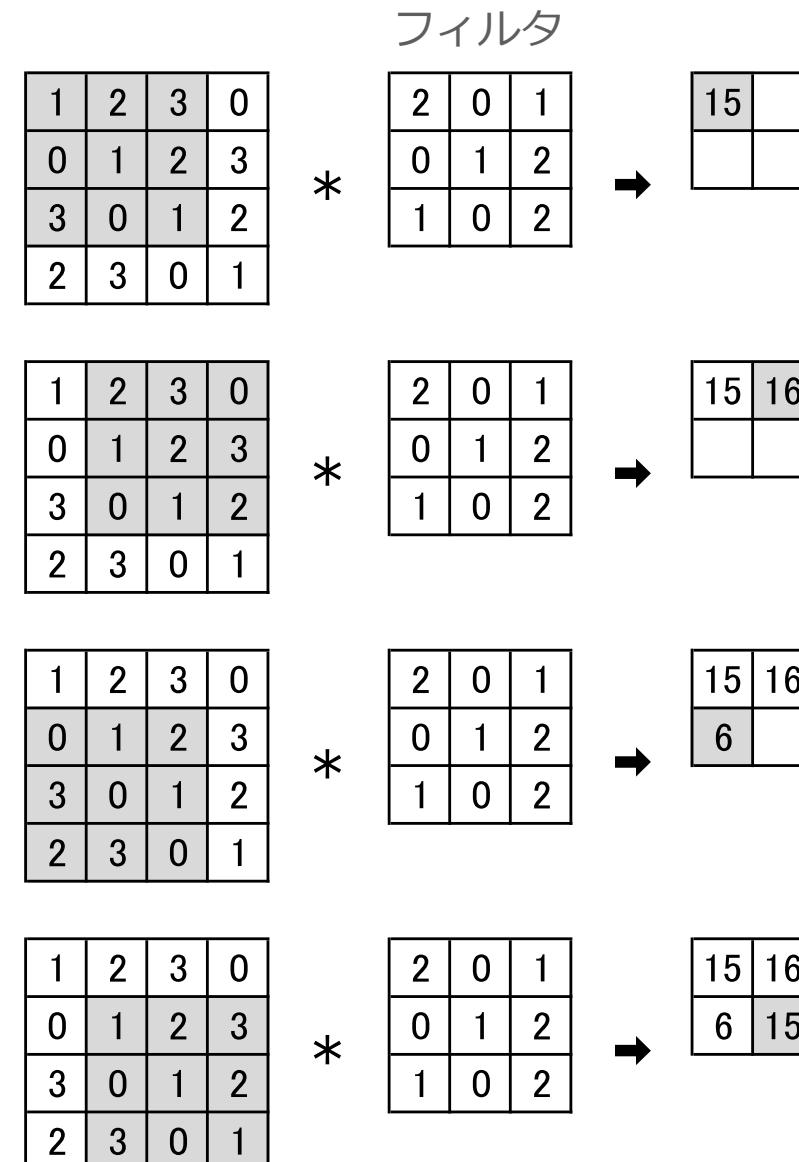
## ■ フィルタの値を誤差逆伝播法で学習

- 複数のフィルタを用意することで、

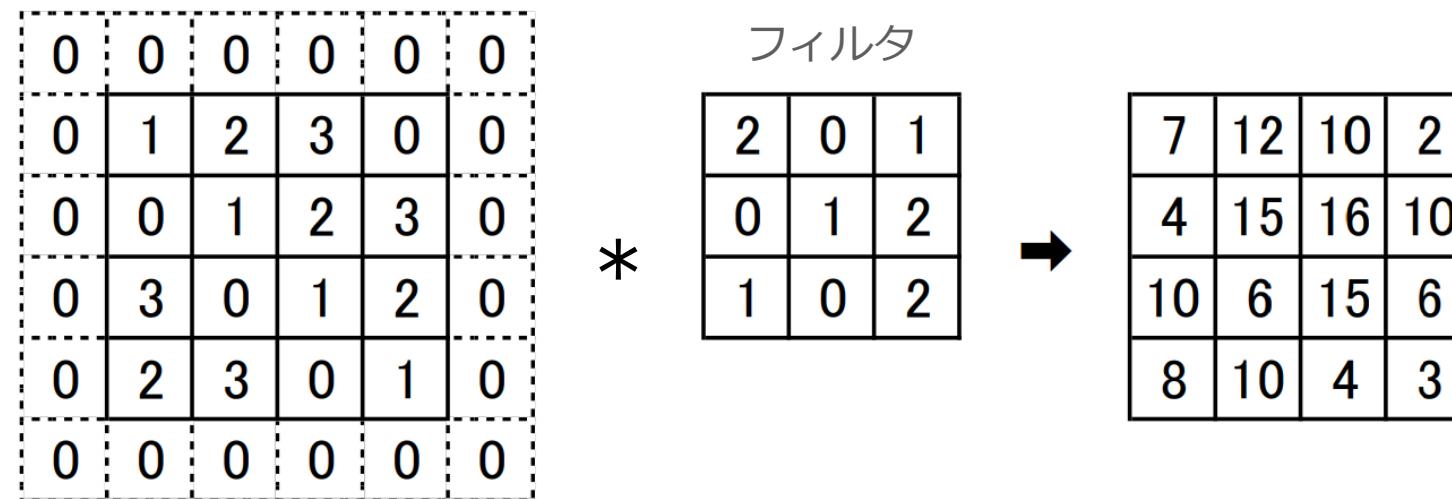
様々な特徴を抽出

## ■ 畳み込み層のハイパーパラメータ

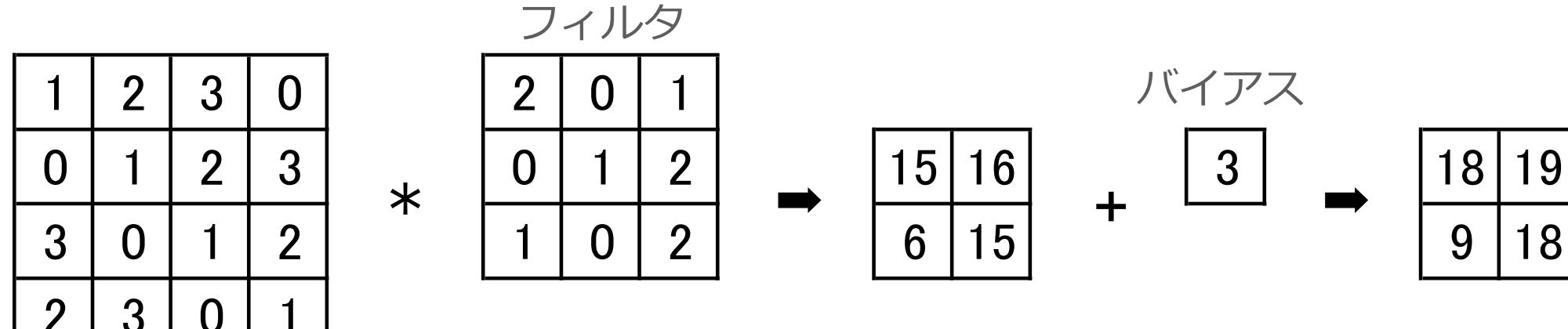
- フィルタのサイズ（大きさ）
- フィルタの枚数
- フィルタをずらす幅（ストライド）
- パディング処理の幅と値



- 畳み込み演算を行う前に画像の周辺に要素を追加すること
- 一般的に、0 を用いて埋めることが多い（ゼロパディング）
- パディングを行うメリット
  - 出力サイズを入力サイズと揃えることができる
  - 画像の周辺部の特徴をとらえやすくなる



- 畳み込み層にも、バイアス項を導入することができる
- 処理された画像に決まった値を足し込むことでバイアスとする



\* : 畳み込み演算

- ほぼ同じデータでも、位置がずれないと畳み込みの結果が変わってしまう
  - ・ 認識したい物体の移動に対して頑健にするにはどうすればよいだろうか？

$$\begin{array}{|c|c|c|c|c|c|} \hline
 1 & 2 & 0 & 7 & 1 & 0 \\ \hline
 0 & 9 & 2 & 3 & 2 & 3 \\ \hline
 3 & 0 & 1 & 2 & 1 & 2 \\ \hline
 2 & 4 & 0 & 1 & 0 & 1 \\ \hline
 6 & 0 & 1 & 2 & 1 & 2 \\ \hline
 2 & 4 & 0 & 1 & 8 & 1 \\ \hline
 \end{array}
 \begin{array}{l}
 \times \\
 \text{フィルタ}
 \end{array}
 \begin{array}{|c|c|c|} \hline
 0 & 1 & 0 \\ \hline
 0 & 0 & 0 \\ \hline
 0 & -1 & 0 \\ \hline
 \end{array}
 \rightarrow
 \begin{array}{|c|c|c|c|} \hline
 2 & -1 & 5 & 0 \\ \hline
 5 & 2 & 2 & 2 \\ \hline
 0 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & -8 \\ \hline
 \end{array}$$

$$\begin{array}{|c|c|c|c|c|c|} \hline
 1 & 1 & 2 & 0 & 7 & 1 \\ \hline
 3 & 0 & 9 & 2 & 3 & 2 \\ \hline
 2 & 3 & 0 & 1 & 2 & 1 \\ \hline
 3 & 2 & 4 & 0 & 1 & 0 \\ \hline
 2 & 6 & 0 & 1 & 2 & 1 \\ \hline
 1 & 2 & 4 & 0 & 1 & 8 \\ \hline
 \end{array}
 \begin{array}{l}
 \times \\
 \text{フィルタ}
 \end{array}
 \begin{array}{|c|c|c|} \hline
 0 & 1 & 0 \\ \hline
 0 & 0 & 0 \\ \hline
 0 & -1 & 0 \\ \hline
 \end{array}
 \rightarrow
 \begin{array}{|c|c|c|c|} \hline
 -2 & 2 & -1 & 5 \\ \hline
 -2 & 5 & 2 & 2 \\ \hline
 -3 & 0 & 0 & 0 \\ \hline
 0 & 0 & 0 & 0 \\ \hline
 \end{array}$$

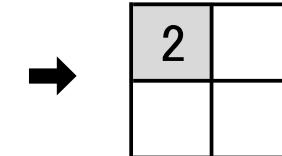
右に 1 ピクセルずれている

## ■ 画像の一部分の最大値や平均値をとる操作

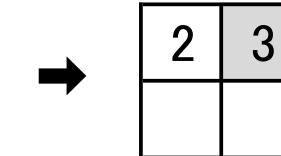
- CNN ではプーリングを用いることで、位置に対する不変性を獲得している
- 以下はサイズ  $2 \times 2$  の最大値プーリング (Max Pooling) の例
  - プーリングのサイズはハイパーパラメータである

1	2	1	0
0	1	2	3
3	0	1	2
2	4	0	1

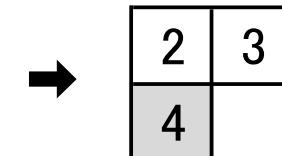
灰色の部分の最大値を  
取り出している



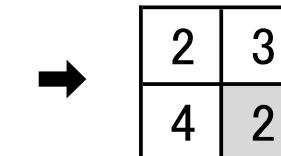
1	2	1	0
0	1	2	3
3	0	1	2
2	4	0	1



1	2	1	0
0	1	2	3
3	0	1	2
2	4	0	1



1	2	1	0
0	1	2	3
3	0	1	2
2	4	0	1



## ■ プーリングにおいて、位置がずれても同じ出力が得られるか確認してみよう

- 例) p17 の畳み込みの出力に、サイズ  $2 \times 2$  の最大値プーリングを行った場合
  - 両方とも同じ結果が得られる
  - 位置によって出力が変化しないことがわかる

2	-1	5	0
5	2	2	2
0	0	0	0
0	0	0	-8



5	5
0	0

プーリング演算の  
結果は同じ

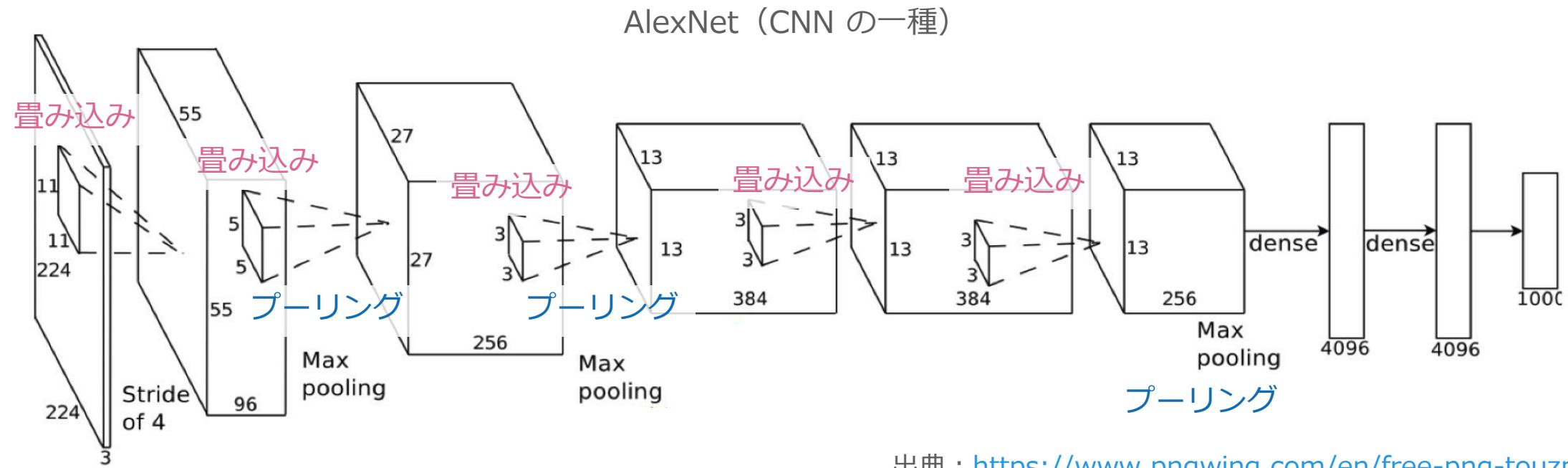
-2	2	-1	5
-2	5	2	2
-3	0	0	0
0	0	0	0



5	5
0	0

右に 1 ピクセルずれている

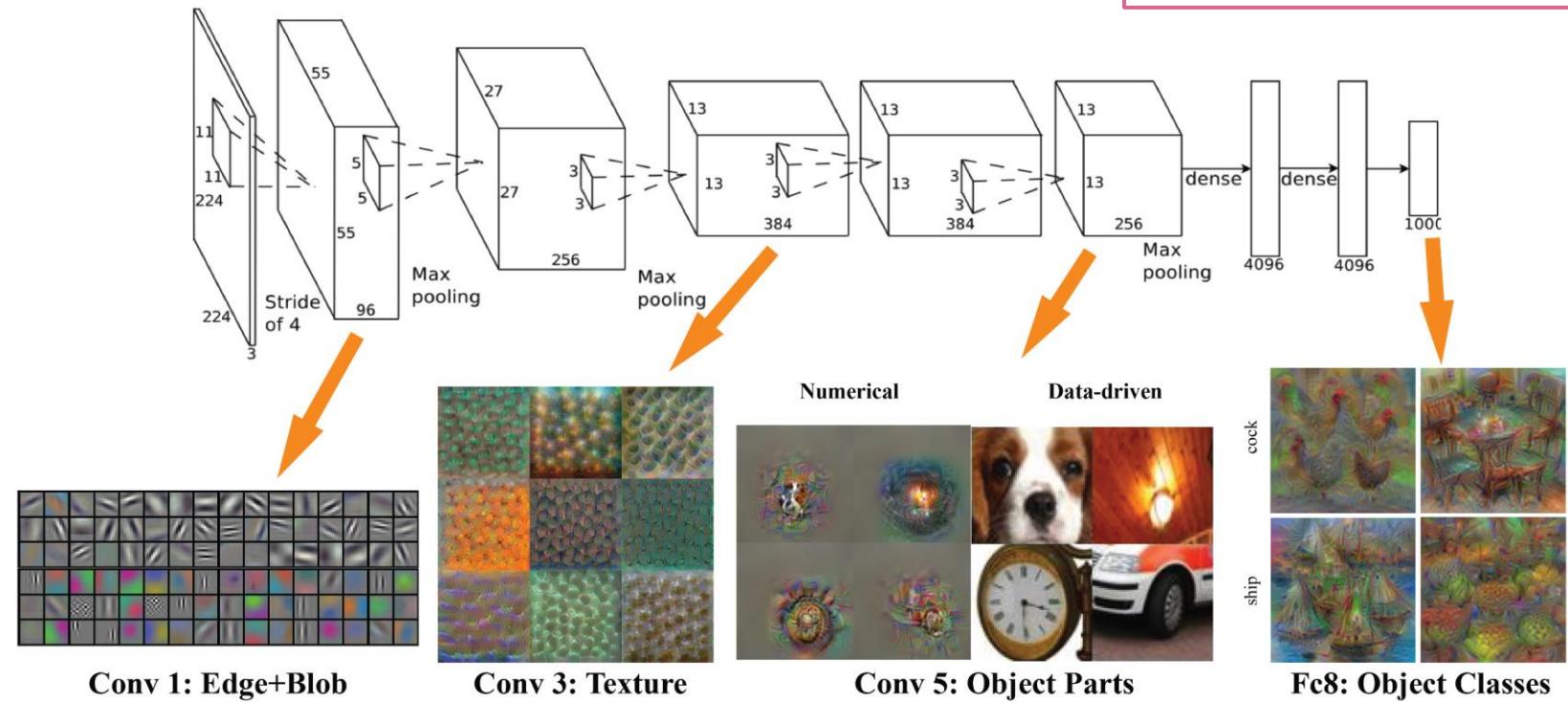
- CNN では、複数の畳み込み層を積み重ねている
- それぞれの畳み込み層で抽出されている特徴は、  
どのようにになっているのだろうか？
- AlexNet と呼ばれる 8 層の CNN で、抽出された特徴を覗いてみよう



出典：<https://www.pngwing.com/en/free-png-touzp>

- 入力側の 1 層目 (Conv1) は、輪郭 (edge) や塊 (blob) に関する特徴
- 3 層目 (Conv3) は、質感 (texture) に関する特徴
- 5 層目 (Conv5) は、物体の部品に関する特徴
- 出力層 (Fc8) では、物体のクラスを出力

出力層に近くなるほど、  
複雑な特徴を抽出している



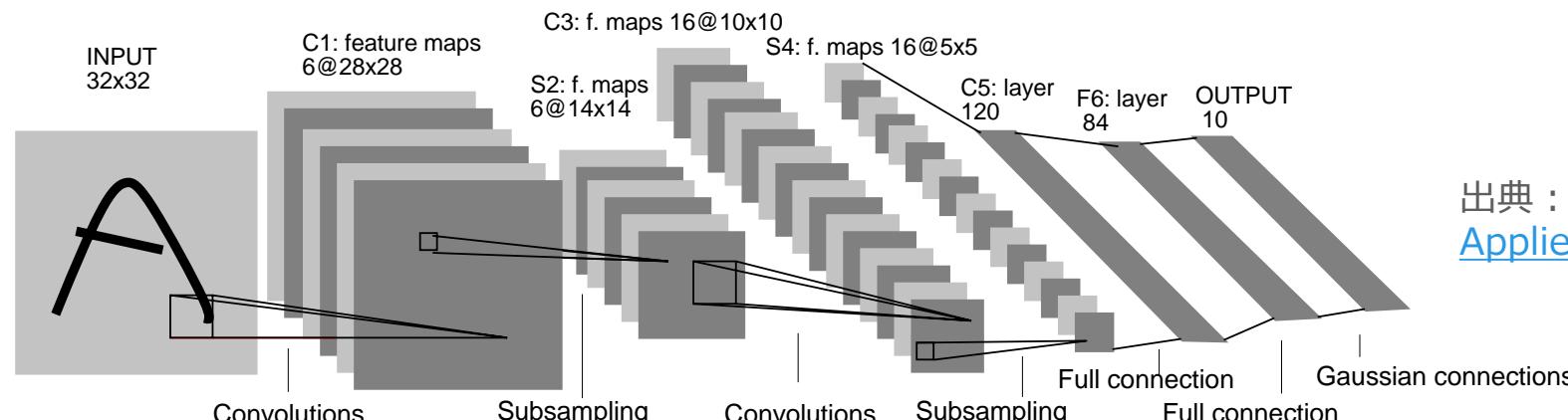
出典：<https://www.pngwing.com/en/free-png-touzp>

## **著名な CNN モデル**

---

- CNN を初めて実用化したネットワーク
- 1998 年ごろ、ヤン・ルカンらによって開発された
  - 1 層目 : サイズ  $5 \times 5$  のフィルタを 6 個持つ畳み込み層
  - 2 層目 :  $2 \times 2$  の領域にサブサンプリングを行うプーリング層
  - 3 層目 : サイズ  $6 \times 6$  のフィルタを 16 個持つ畳み込み層
  - 4 層目 :  $2 \times 2$  の領域にサブサンプリングを行うプーリング層
  - 5 層目 : サイズ  $6 \times 6$  のフィルタを 120 個持つ畳み込み層
  - 最終層 (6 層目) : 全結合層

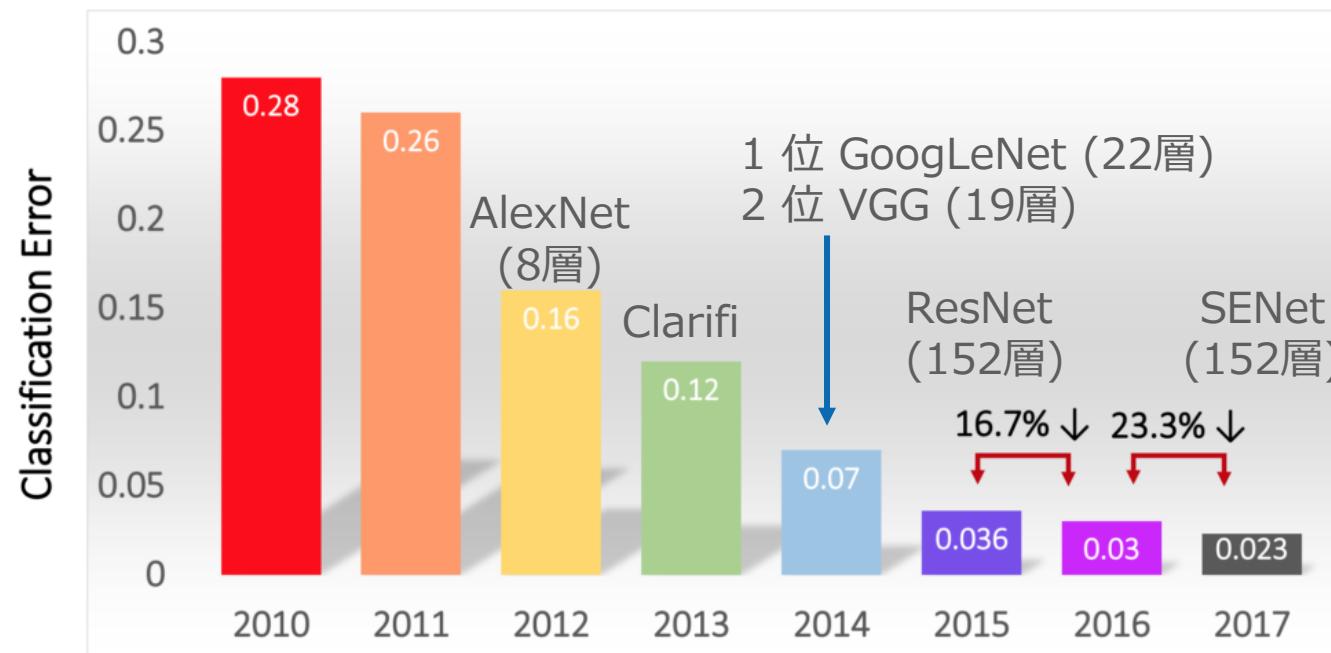
1980 年には、福島邦彦によって  
[ネオコグニトロン](#)という  
CNN の原型が提唱されている



出典 : [Gradient-Based Learning Applied to Document Recognition](#)

## ■ ImageNet という大規模なデータセットを用いた、画像分類のコンテスト

- 画像分類モデルの性能を高めることを目的として、2010 年より開催
- 2012 年、CNN を用いた手法である AlexNet が優勝し、**深層学習ブーム**の火付け役となる
- モデルの性能が十分なものに達したため、2018 年に終了
- 参考：[画像解析関連コンペティションの動向（2017年5月）](#)

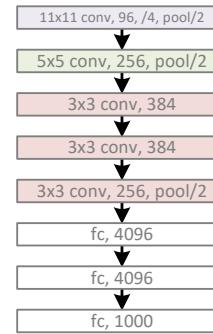


出典：[ImageNet Winning CNN Architectures \(ILSVRC\) | Kaggle](#)

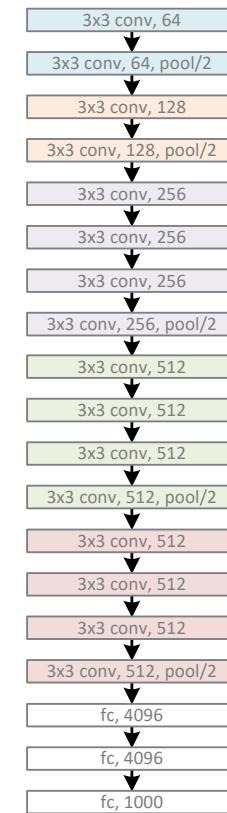
# 著名な CNN モデル | 2012 年 ~ 2014 年

■ ILSVRC 優勝モデルは 2012 年に 8 層、2014 年には 22 層まで深くなっている

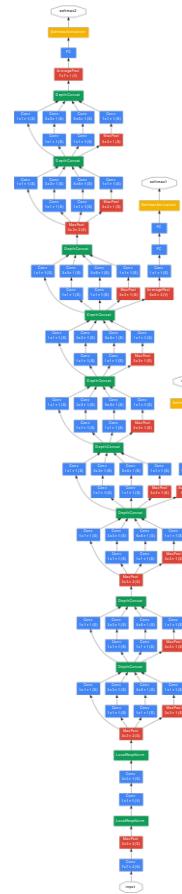
AlexNet, 8 layers  
(ILSVRC 2012)



VGG, 19 layers  
(ILSVRC 2014)



GoogleNet, 22 layers  
(ILSVRC 2014)



出典 : [https://kaiminghe.github.io/ilsvrc15/ilsvrc2015\\_deep\\_residual\\_learning\\_kaiminghe.pdf](https://kaiminghe.github.io/ilsvrc15/ilsvrc2015_deep_residual_learning_kaiminghe.pdf)

# 著名な CNN モデル | 2012 年 ~ 2015 年

- なんと 2015 年の ILSVRC 優勝モデルは 152 層！
- 今後は更に層数が多く増えていく可能性も…

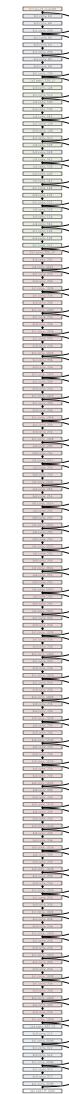
AlexNet, 8 layers  
(ILSVRC 2012)



VGG, 19 layers  
(ILSVRC 2014)



ResNet, 152 layers  
(ILSVRC 2015)



出典 : [https://kaiminghe.github.io/ilsvrc15/ilsvrc2015\\_deep\\_residual\\_learning\\_kaiminghe.pdf](https://kaiminghe.github.io/ilsvrc15/ilsvrc2015_deep_residual_learning_kaiminghe.pdf)

## CNN の応用例

---

## ■ 物体検出 (Object detection)

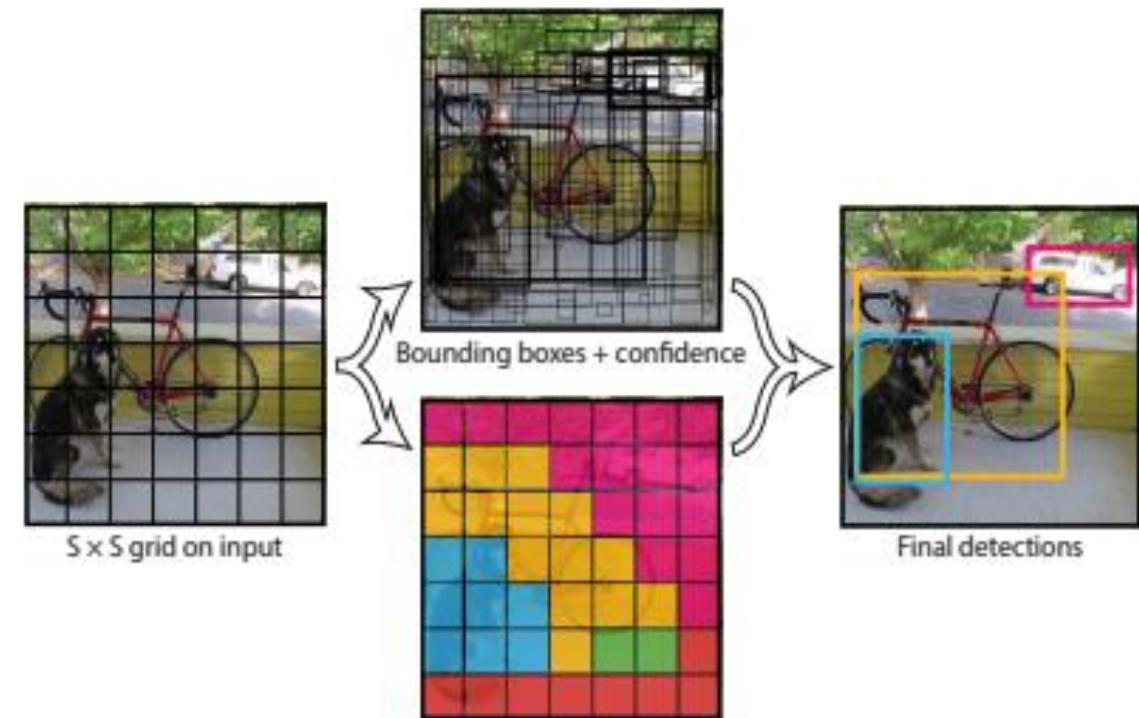
- ・ 画像の中から物体を検出し、その物体のクラスを予測するタスク
- ・ 物体が含まれる領域をバウンディングボックスという

## ■ YOLO (You Only Look Once)

- ・ 物体検出モデルの 1 つ
- ・ 画像全体をグリッドに分割
- ・ 各領域毎に物体のクラスとバウンディングボックスを求める

## ■ 参考動画

- ・ [YOLO 9000 Object Detection #7](#)



出典：[You only look once: Unified, real-time object detection](#)

## ■ セグメンテーション (Segmentation)

- ・ 画像をクラスごとに塗り分けるタスク
- ・ 画素（ピクセル）単位の分類問題
- ・ 自動運転や医用画像の分野において重要な技術
- ・ 通常は Semantic Segmentation のことを指す

## ■ ICNet

- ・ セグメンテーションモデルの 1 つ
- ・ リアルタイム性を重視した軽量なモデル

## ■ 参考動画

- ・ [ICNet for Real-Time Semantic Segmentation on High-Resolution Images](#)

同じクラスの物体をまとめて扱う方法 (Semantic Segmentation) と  
物体一つ一つを区別する方法 (Instance Segmentation) がある

Image      Instance Segmentation      Semantic Segmentation



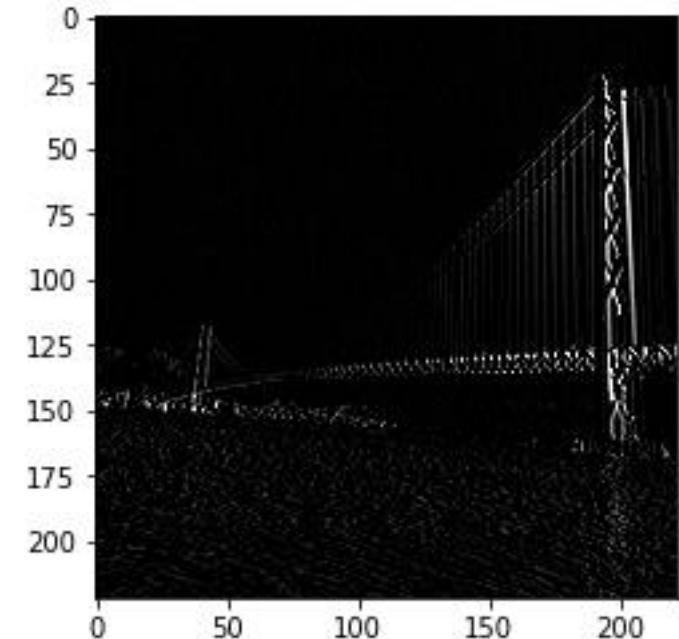
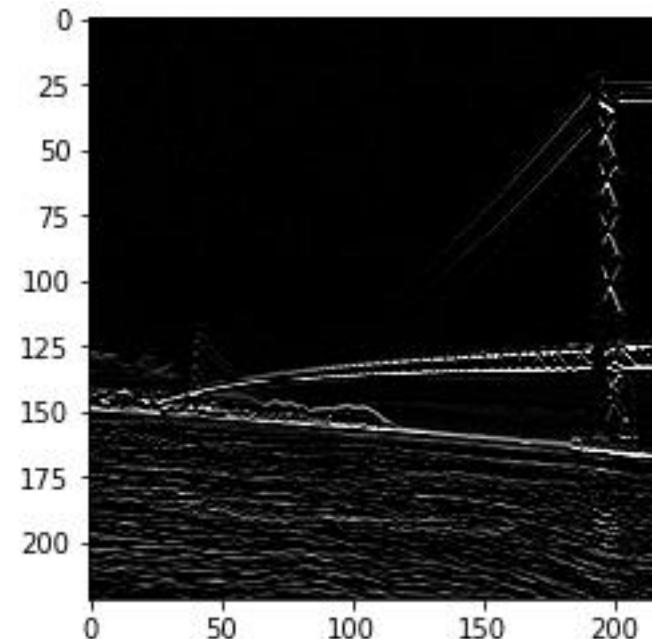
出典：[PASCAL VOC2011 Example Segmentations](#)

## ノートブック演習

---

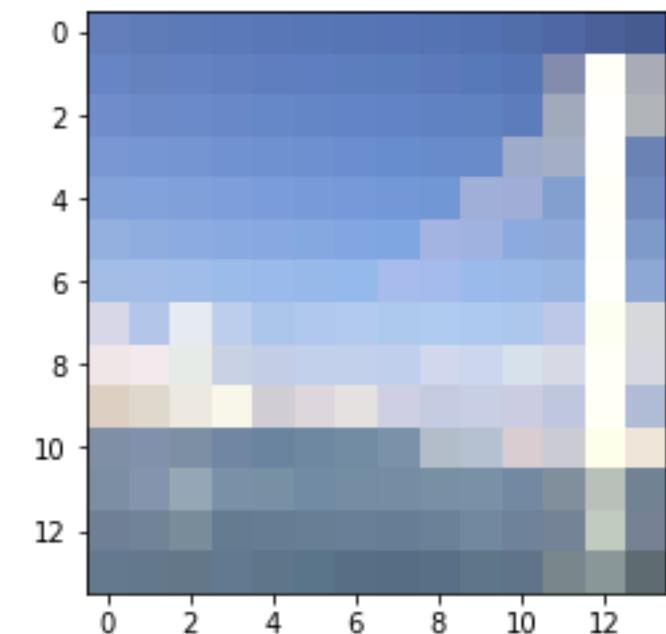
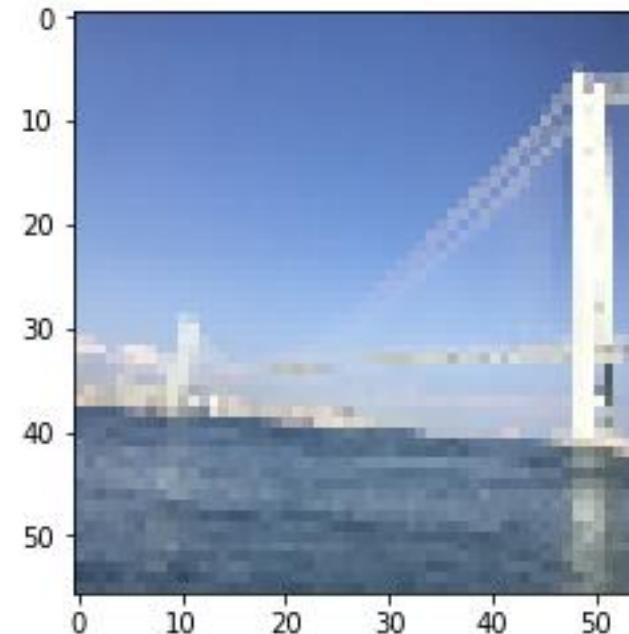
## 12-1\_convolution.ipynb

- 置み込み演算の振る舞いを確認してみよう



## 12-2\_pooling.ipynb

- プーリングの振る舞いを確認してみよう



## 12-3\_CNN.ipynb

- CNN は学習済みモデルが多く配布されており、TensorFlow、Keras などのライブラリを用いて比較的簡単に利用できる
- EfficientNet の学習済みモデルを読み込んで、画像分類を行ってみよう
  - EfficientNet は、予測の速さと性能の良さを両立する目的で設計された CNN

入力画像



予測結果

1番、信頼スコア：0.09、	クラス名：lakeside, lakeshore
2番、信頼スコア：0.08、	クラス名：balloon
3番、信頼スコア：0.07、	クラス名：seashore, coast, seacoast, sea-coast
4番、信頼スコア：0.04、	クラス名：parachute, chute
5番、信頼スコア：0.04、	クラス名：flagpole, flagstaff

## 本章のまとめ

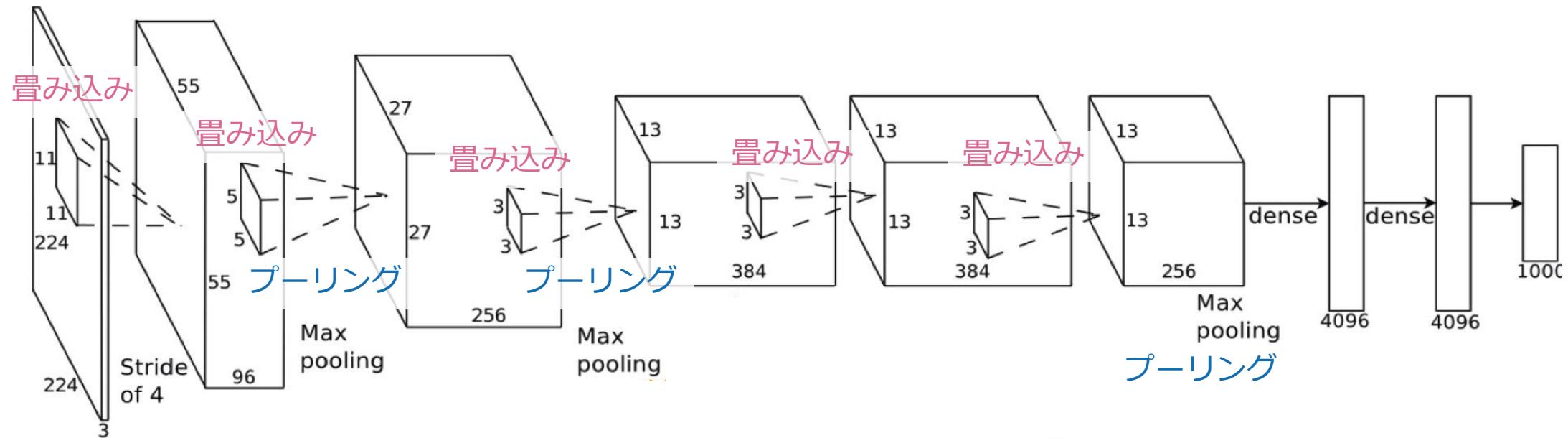
---

- 置み込みニューラルネットワーク
- 置み込みとプーリング
- 著名な CNN モデル
- CNN の応用例
- ノートブック演習

## ■ 置み込みニューラルネットワーク (CNN)

- ・ 置み込み層とプーリング層を備えたニューラルネットワーク
- ・ 画像から特徴を上手く抽出することができる

AlexNet (CNNの一種)



## ■ 置み込みとプーリング

- 置み込み層
  - フィルタを用いて特徴を抽出
- プーリング層
  - 画像の一部分から最大値や平均値を抽出

## ■ CNN の応用例

- 物体検出
  - 画像の中から物体を検出し、その物体のクラスを予測するタスク
- セグメンテーション
  - 画像をクラスごとに塗り分けるタスク

# 現場で使える 機械学習・データ分析基礎講座

第 13 章：深層学習の代表的な手法

- 再帰型ニューラルネットワーク
- 敵対的生成ネットワーク

- 再帰型ニューラルネットワークの構成を説明できる
- 敵対的生成ネットワークによる生成の流れを説明できる

# 再帰型ニューラルネットワーク

---

## 決定的

### 階層型

全結合型ニューラルネットワーク  
Fully connected neural network

畳み込みニューラルネットワーク  
Convolutional neural network

再帰型ニューラルネットワーク  
Recurrent neural network

### 自己符号化器型

自己符号化器  
Autoencoder

雑音除去自己符号化器  
Denoising Autoencoder

変分自己符号化器  
Variational Autoencoder

スパース自己符号化器  
Sparse Autoencoder

## 確率的

### ボルツマンマシン型

ボルツマンマシン  
Boltzmann machine

制約ボルツマンマシン  
Restricted Boltzmann machine

深層信念ネットワーク  
Deep belief network

深層ボルツマンマシン  
Deep Boltzmann machine

参考：『深層学習』(神鳴ら、近代科学社)

- CNN は画像に適した処理方法である、畳み込みを組み込むことで効果的な特徴を抽出していた
- 次は自然言語や音声などの**時系列データ**を考えてみよう
- ある時点におけるデータは、以前のデータと何らかの関係を持つはず
  - 自然言語であれば、ある時点の単語はその前に出た単語と関係があるはず
- **以前のデータとの関係**も使って、予測値を決めるにはどのようにすればよいだろうか？

## ■ 時間方向に状態を引き継ぎながら計算を進めるニューラルネットワーク

- 自然言語処理や音声認識などに利用される
- 時系列データ（順番に並んでいるデータ）に向いている
- 可変長データ（長さが毎回異なるデータ）を扱う場合に向いている

## ■ RNN 以外でも、時系列データを扱うことは可能

- 自己回帰モデルは、通常の全結合型ニューラルネットワークでも実現できる
- 2017年以降は、再帰構造をもたないモデルである Transformer が主流



**自己回帰モデル**  
前時刻（あるいは前の単語）に対する出力を入力として、繰り返し計算を行うモデル

## ■ シンプルな RNN

- 単純な全結合層を用いて状態を更新
- 長い系列を学習しても、過去の情報がほとんど反映されない

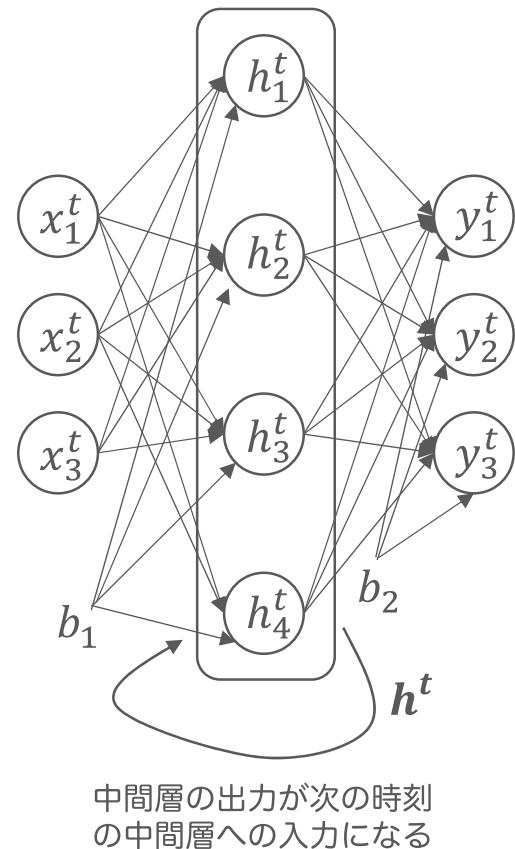
## ■ LSTM (Long Short-Term Memory unit)

- シンプルな RNN の欠点を解決したモデル
- ゲートと呼ばれる仕組みを導入（入力ゲート・出力ゲート・忘却ゲート・記憶セル）
- 必要な情報だけを次の状態に引き継ぐことで、情報を長く記憶できる

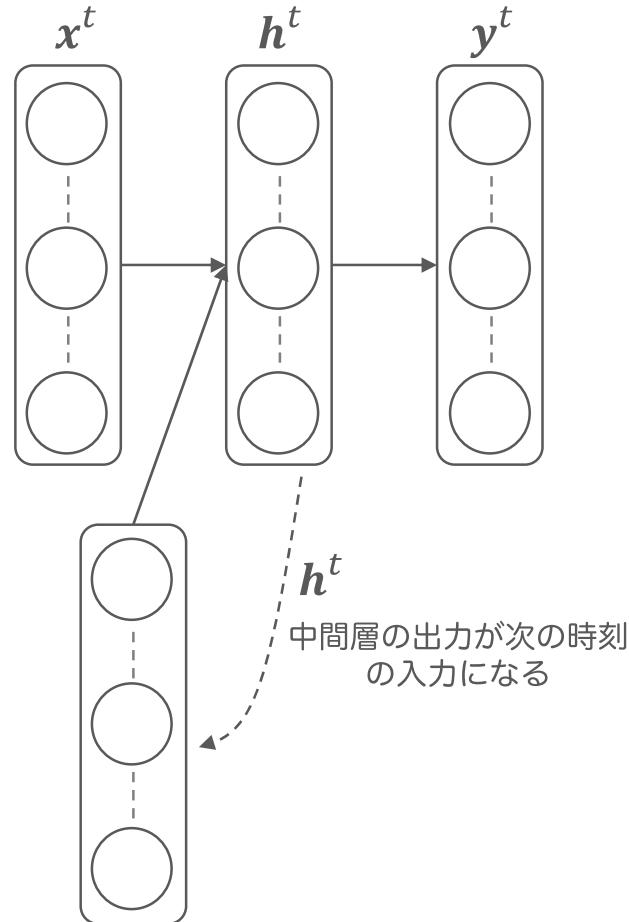
## ■ GRU (Gated Recurrent Unit)

- LSTM をシンプルにしたモデル
- LSTM よりもパラメータの数が少ない
- リセットゲートと更新ゲートのみで構成される

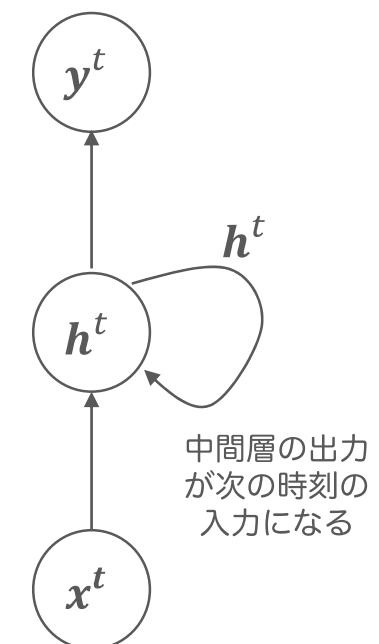
- 中間層の出力が次の時刻の入力となる、再帰的な接続を持つ



簡略化する  
→



もっと  
簡略化する  
→



## ■ Bi-directional RNN

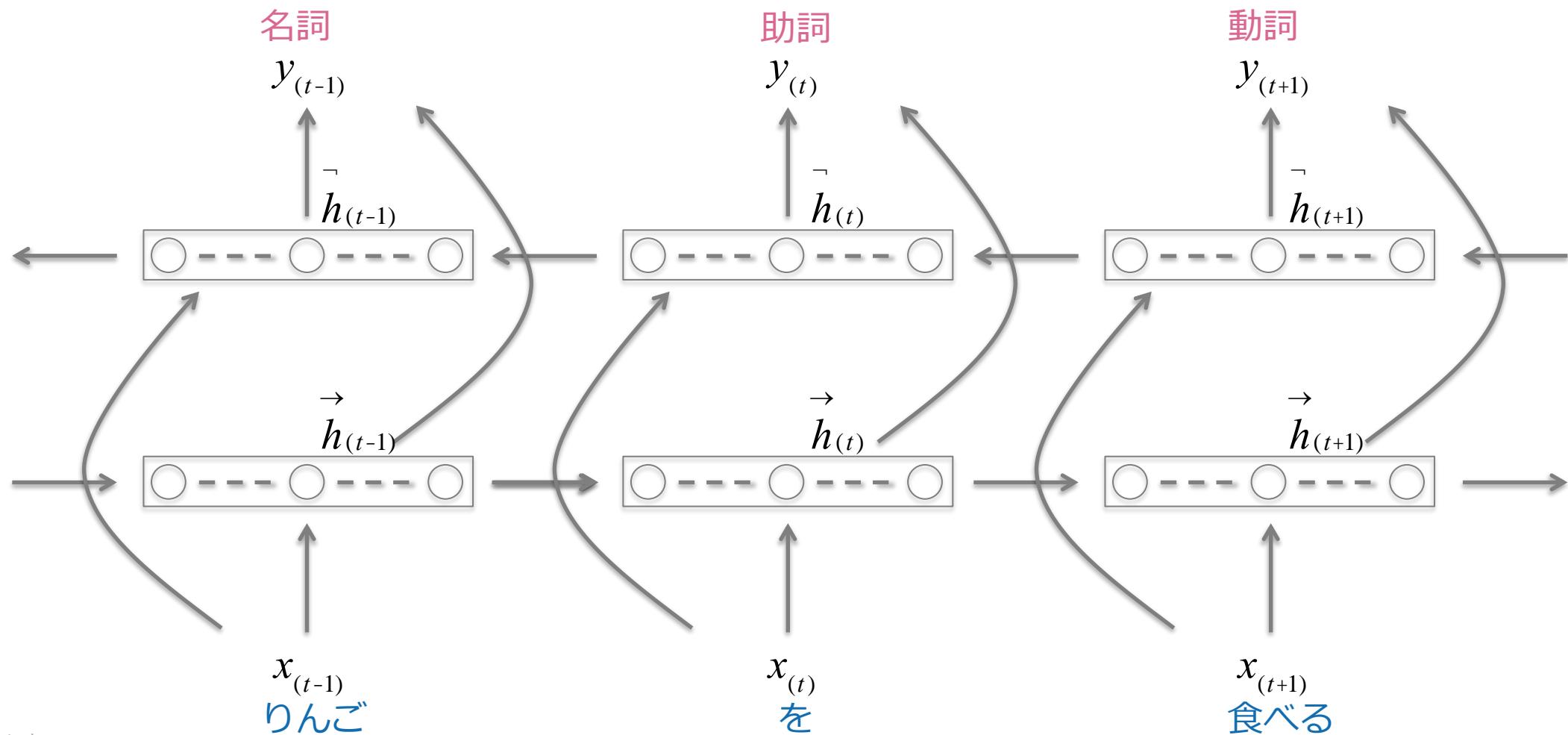
- ・ 過去から未来だけでなく、**未来から過去への方向**も考慮した RNN
- ・ すでに手元にあるデータに対して、何らかの予測を行う場合に用いる
- ・ 例) 空欄に入る単語の予測、単語の品詞推定

## ■ Encoder-Decoder モデル

- ・ 入力と出力が**両方とも時系列データ**であるモデル
- ・ Sequence-to-sequence model (Seq2seq) とも呼ばれる
- ・ 例) 機械翻訳（和文→英文など）、音声認識（音声→テキスト）

■ 「過去←未来」の方向にも、隠れ状態を伝える

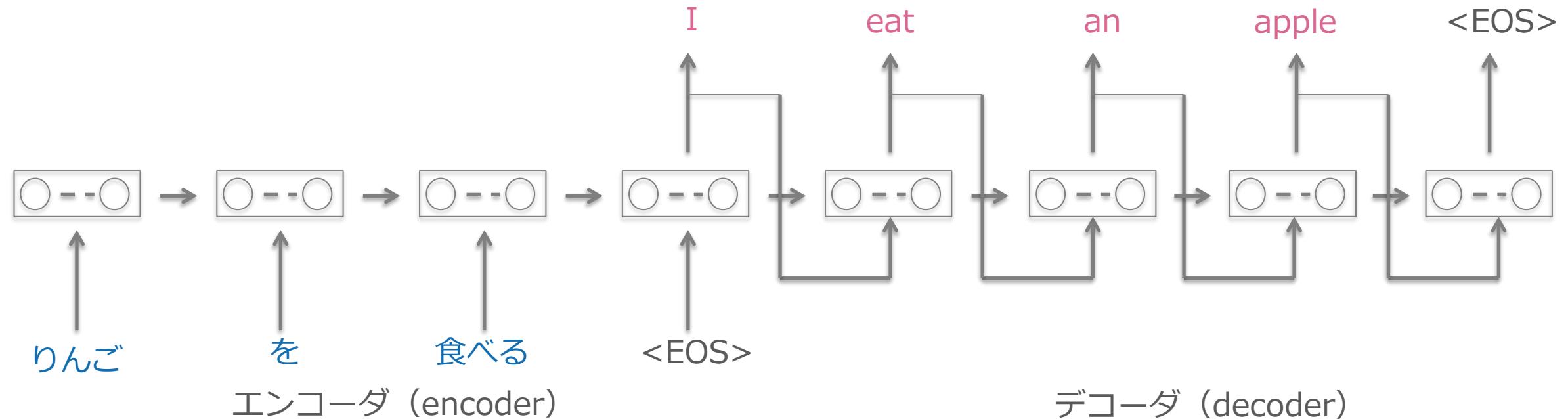
Bi-directional RNNの例



## ■ 入力と出力は共に時系列データ

- 入力の系列長（例：単語数）と出力の系列長は一致していなくてよい
- 系列長は、データごとに異なっていてもよい

RNN Encoder-Decoderの例



- RNN 系モデルと CNN 系モデルを組み合わせたネットワークを構築
  - 画像とテキストを同時に扱うタスクに応用できる
- タスクの例
  - 画像キャプション生成
  - 画像質問応答

## ■ 画像の内容を自然言語で記述するタスク

- CNN と LSTM を組み合わせたモデルを利用

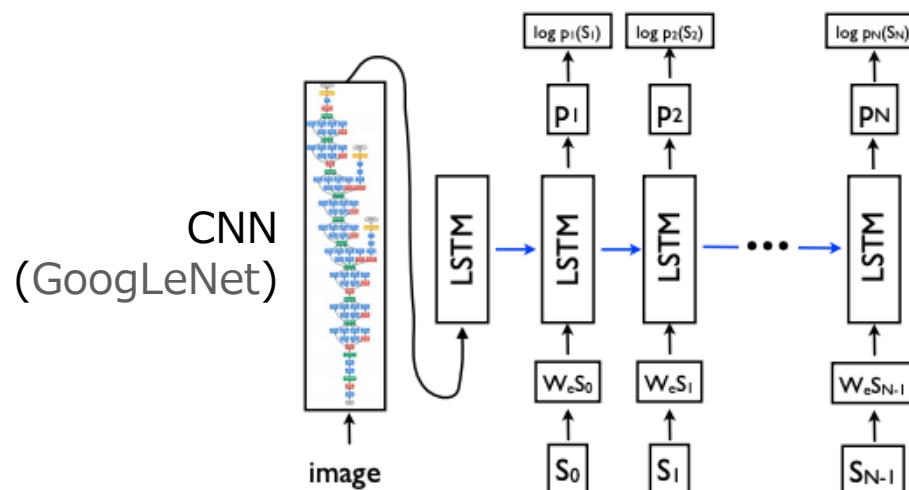


Figure 3. LSTM model combined with a CNN image embedder (as defined in [12]) and word embeddings. The unrolled connections between the LSTM memories are in blue and they correspond to the recurrent connections in Figure 2. All LSTMs share the same parameters.



Figure 5. A selection of evaluation results, grouped by human rating.

出典：<https://arxiv.org/pdf/1411.4555.pdf>

## ■ 画像に関する質問に答えるタスク

- Visual Question Answering (VQA)
- CNN と LSTM を組み合わせたモデルを利用

## ■ 専用のデータセットも公開されている

- <https://visualqa.org/>

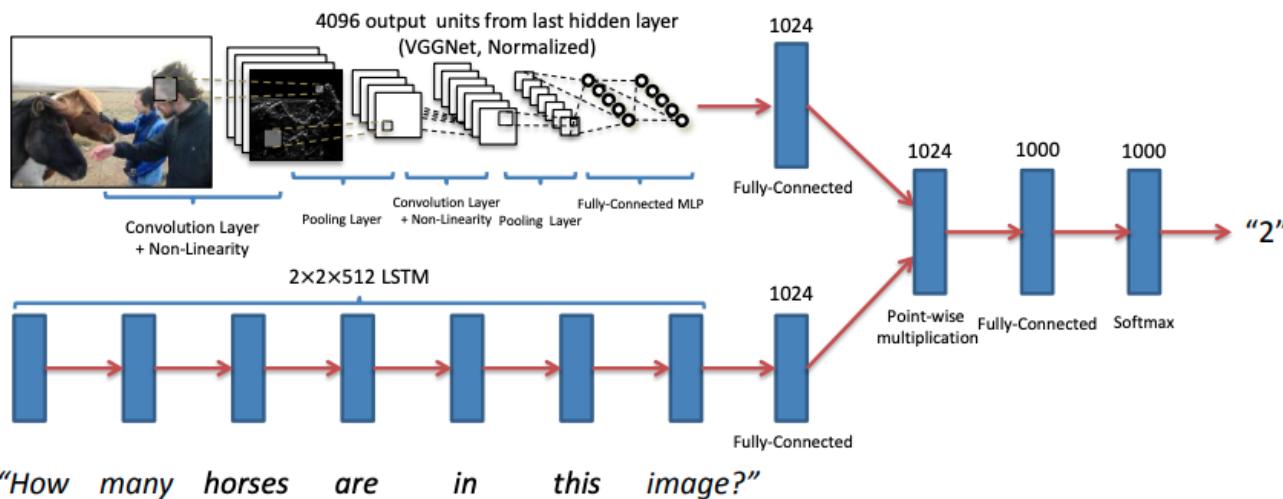


Fig. 8: Our best performing model (deeper LSTM Q + norm I). This model uses a two layer LSTM to encode the questions and the last hidden layer of VGGNet [48] to encode the images. The image features are then  $\ell_2$  normalized. Both the question and image features are transformed to a common space and fused via element-wise multiplication, which is then passed through a fully connected layer followed by a softmax layer to obtain a distribution over answers.



What color are her eyes?  
What is the mustache made of?



How many slices of pizza are there?  
Is this a vegetarian pizza?



Is this person expecting company?  
What is just under the tree?



Does it appear to be rainy?  
Does this person have 20/20 vision?

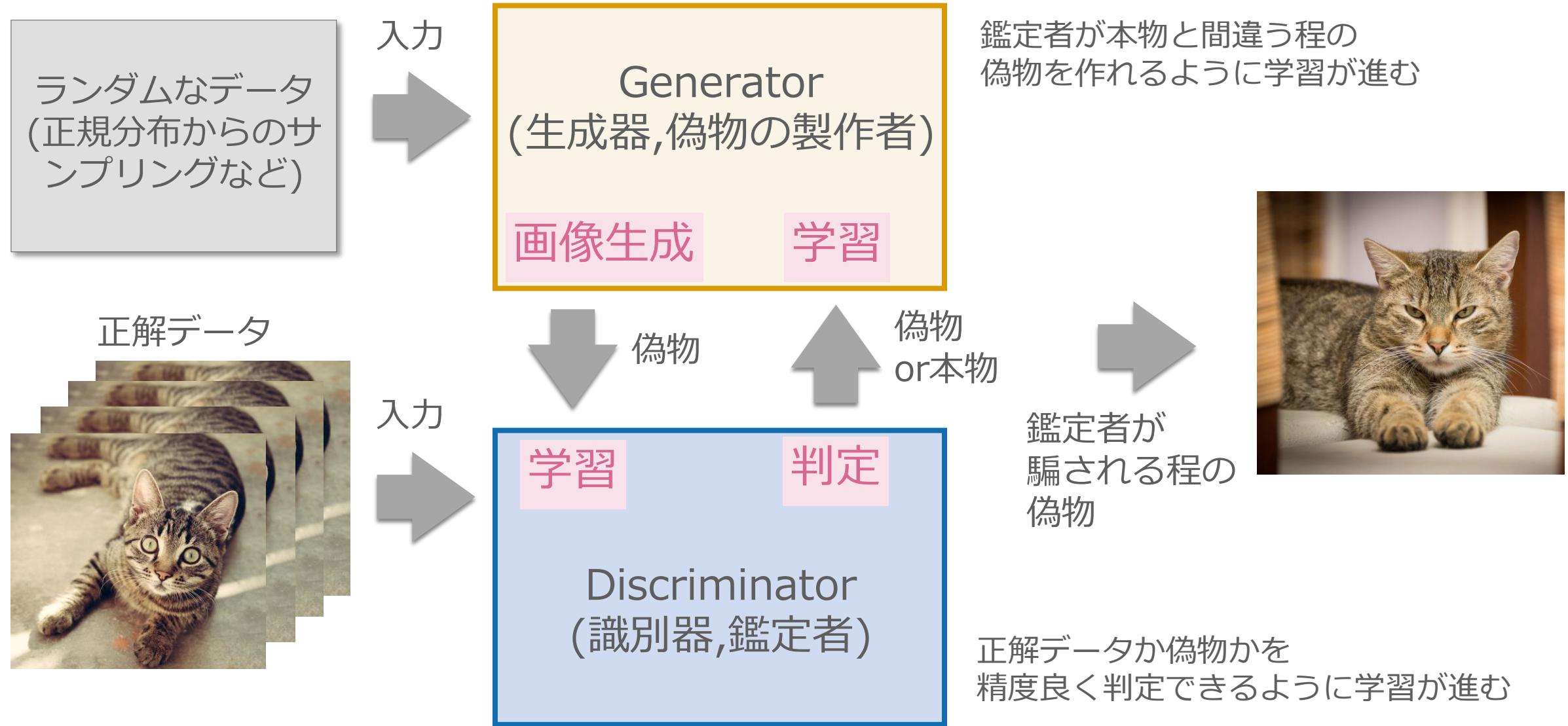
Fig. 1: Examples of free-form, open-ended questions collected for images via Amazon Mechanical Turk. Note that commonsense knowledge is needed along with a visual understanding of the scene to answer many questions.

出典：<https://arxiv.org/pdf/1505.00468.pdf>

## 敵対的生成ネットワーク

---

- 新しいデータを生成する「生成モデル」の一種
  - ・ 他の生成モデル : VAE、フローベースモデル、拡散モデル
- 2つのネットワークを競い合うように学習させることが特徴
  - ・ 生成器 (Generator)
    - ・ 偽物を作る側
    - ・ Discriminator を騙すことが目的
    - ・ 本物そっくりの偽物を作るように学習を進める
  - ・ 識別器 (Discriminator)
    - ・ 偽物を見破る側
    - ・ Generator の作った偽物を見破ることが目的
    - ・ ほんのわずかな違いも見破れるように学習を進める



## ■ 高解像度の画像を生成する GAN

- 生成器に CNN を用いる

生成された画像



生成器のネットワーク

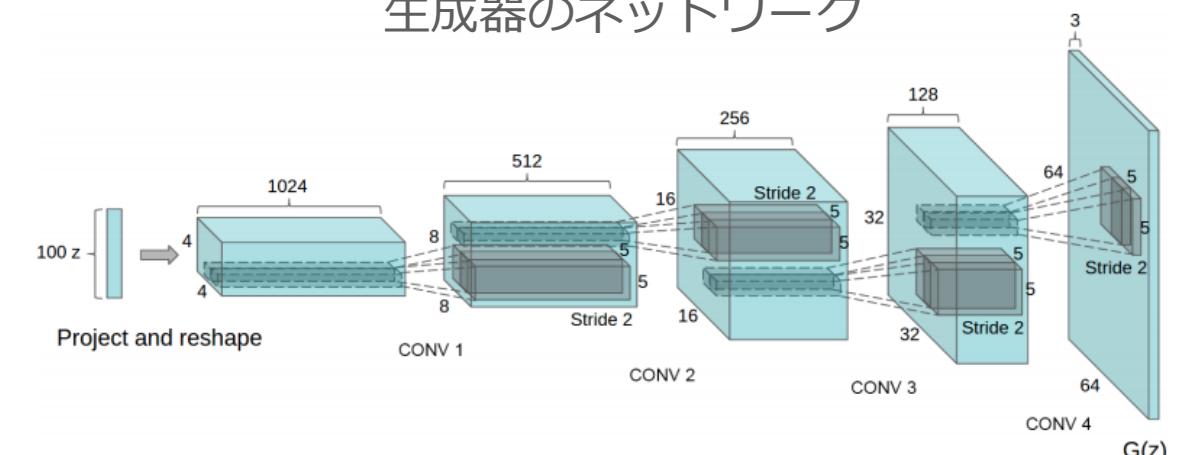
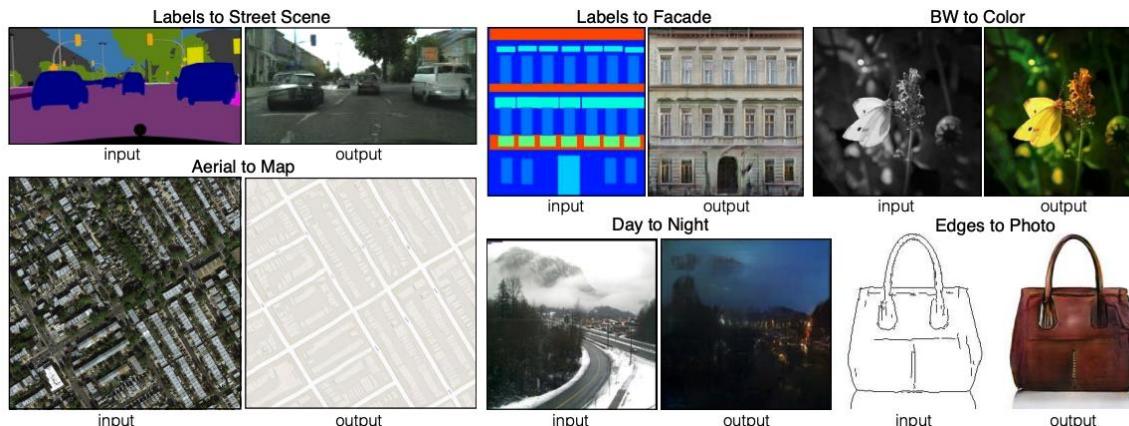


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution  $Z$  is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a  $64 \times 64$  pixel image. Notably, no fully connected or pooling layers are used.

出典：<https://arxiv.org/pdf/1511.06434.pdf>

## ■ 画像を別の画像に変換するモデル

- GAN の枠組みで学習
- 変換前画像  $x$  と変換後画像  $y$  のペアを用意する必要がある



出典：<https://phillipi.github.io/pix2pix/>

Dには、変換前画像 $x$ と  
生成された変換後画像 $G(x)$ を入れる

Dには、変換前 $x$ と  
本物の変換後画像 $y$ を入れる

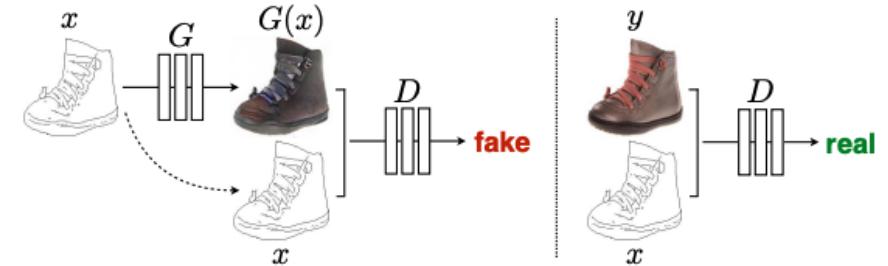


Figure 2: Training a conditional GAN to map edges→photo. The discriminator,  $D$ , learns to classify between fake (synthesized by the generator) and real {edge, photo} tuples. The generator,  $G$ , learns to fool the discriminator. Unlike an unconditional GAN, both the generator and discriminator observe the input edge map.

出典：<https://arxiv.org/pdf/1611.07004.pdf>

## ■ INAI MODEL

- ・ 全身モデルの画像を生成するサービス
- ・ ポーズの指定も可能
- ・ 広告やサンプルなどに利用できる

## ■ 参考

- ・ [AIがモデルのライバルに？「全身画像」量産サービスが始動](#)
- ・ [\[DataGrid\] Model generation AI | YouTube](#)



出典：<https://Imagenavi.jp/topics/inaimodel/>

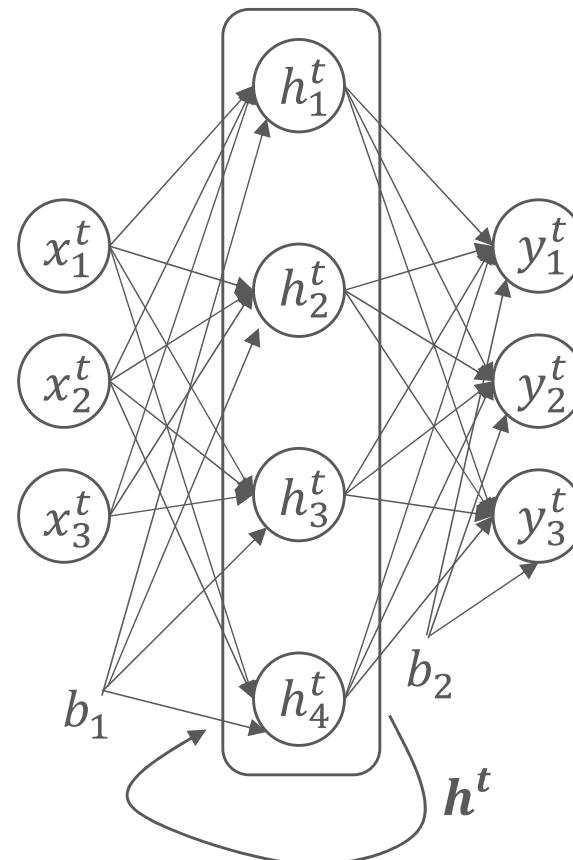
## 本章のまとめ

---

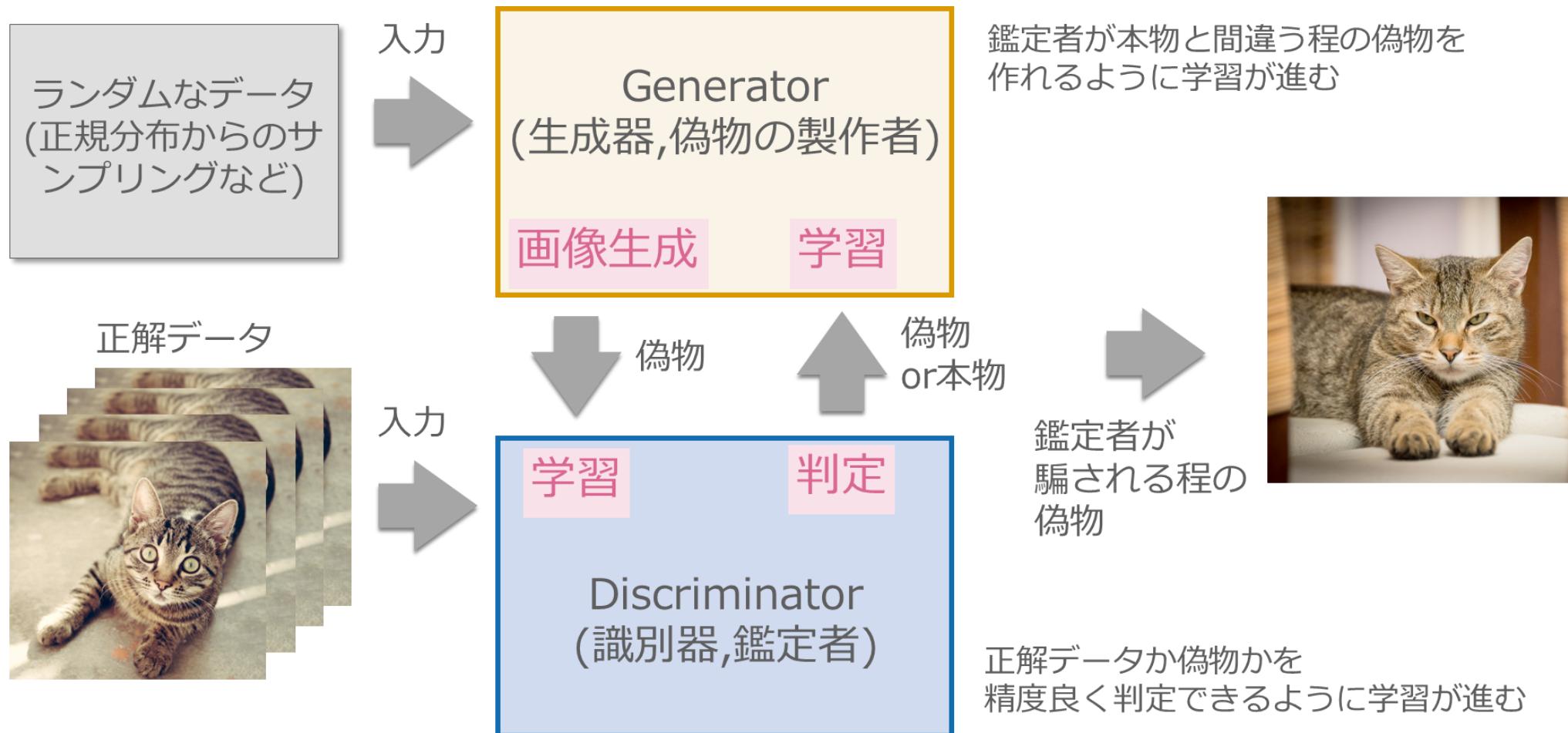
- 再帰型ニューラルネットワーク
- 敵対的生成ネットワーク

## ■ 再帰型ニューラルネットワーク (RNN)

- 再帰構造をもつニューラルネットワーク
- 可変長の時系列データの処理に向いている



## ■ 敵対的生成ネットワーク (GAN)



# 現場で使える 機械学習・データ分析基礎講座

第 14 章：教師なし学習

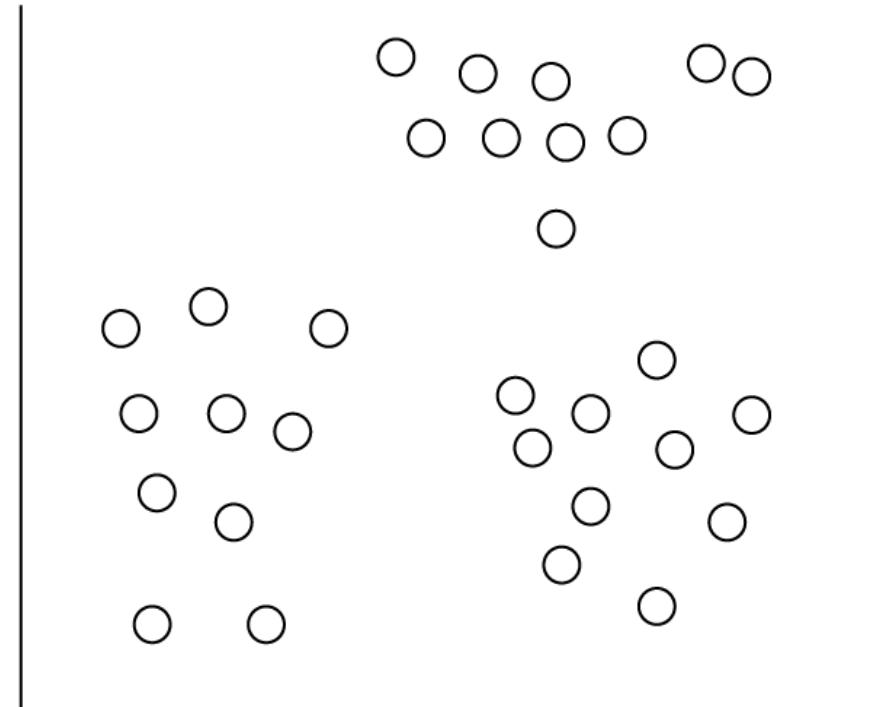
- クラスタリング
- k-means 法
- 主成分分析
- 自己符号化器
- ノートブック演習

- k-means 法の学習の進め方と用途を説明できる
- 主成分分析の用途を説明できる
- 自己符号化器の用途を説明できる

# クラスタリング

---

- グループの数はいくつ？
- それぞれのグループの、どこに中心がある？



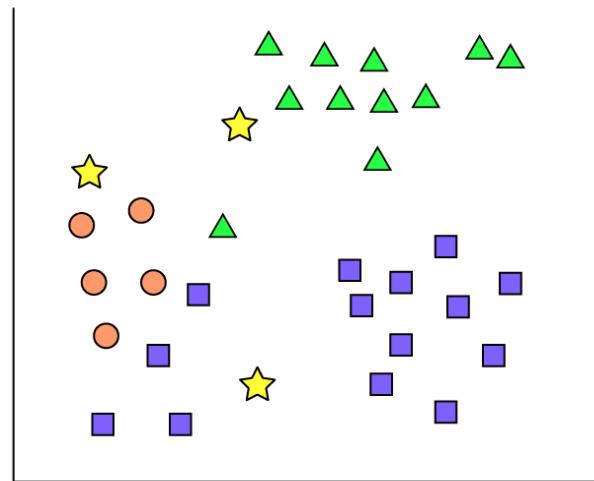
- 似ているデータを集めて群（クラスタ）を作り、対象をグループ分けする方法
  - ・ クラスタという言葉は、マーケティング分野でよく使われる
- グループ分けの結果に正解は無い
  - ・ 各グループがどのような特徴をもっているのかは、出力結果を見て分析者が解釈する必要がある
- 今回は代表的なクラスタリングの手法である k-means 法を学習

# k-means 法

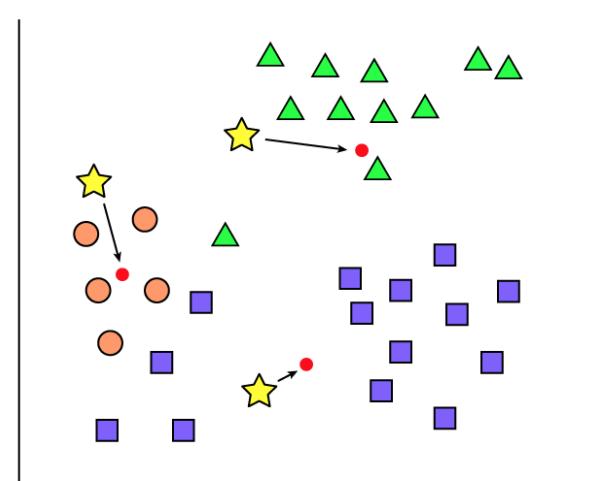
---

# k-means 法 | 基本原理

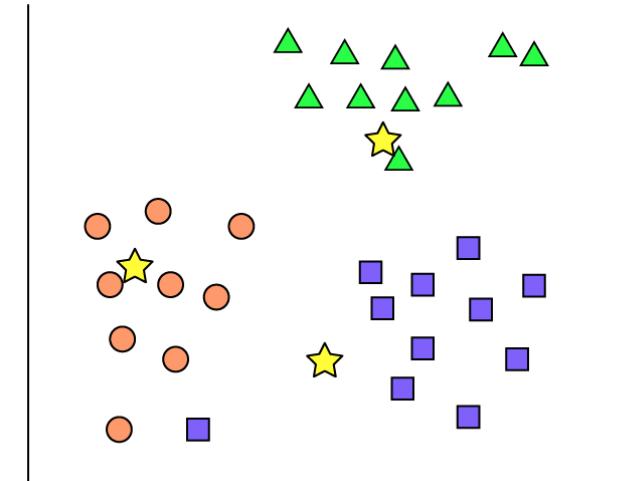
- 事前に  $k$  個のグループがあることを想定
- 最初に、グループの中心点をランダムに配置
- 中心点が動かなくなるまで、以下の処理を繰り返す
  1. データをグループに割り当てる
  2. グループ内の中心点の更新



中心点をランダムに配置  
データを初期グループに割り当て

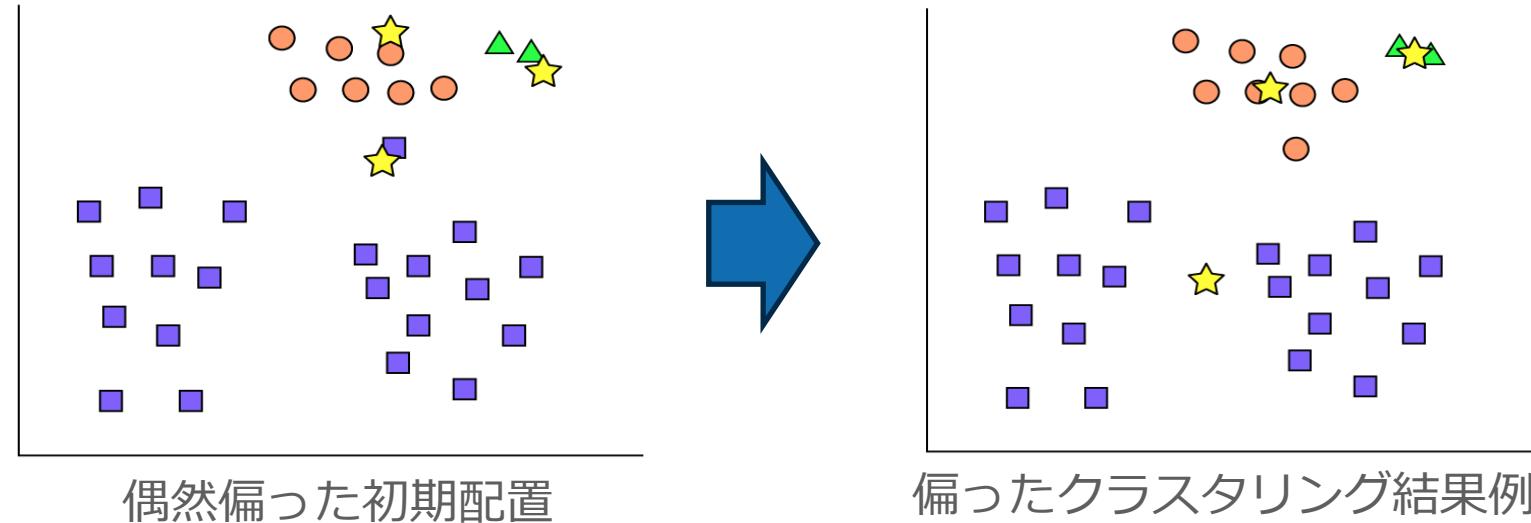


グループの中心点を更新



データをグループに再割り当てる

- 中心点の初期配置によっては、満足な結果が得られない
- 対策
  - k-means++ 法を用いる
  - 初期配置を変えて複数回実行し、誤差平方和が最小になるものを選択する



- ランダム性を保ちつつも、できるだけ最初の中心点間の位置を離しておく方法
  - scikit-learn の KMeans クラスでは、デフォルトで **k-means++ 法**を使用
- アルゴリズム
  1. 入力データの中から初期の中心点をランダムに 1 個選択
  2. 中心点以外の入力データを対象に、**各中心点との距離の 2 乗**を求める
  3. 中心点以外の入力データを対象に、**最近傍中心点からの距離の 2 乗**に比例する**重み**を割り当てる
  4. その重みを考慮しながら、次の中心点をランダムに選択
    - これにより、現在の中心点から遠い点ほど次の中心点に選ばれやすくなる
  5. ステップ 2~4 を、**中心点が k 個になるまで**繰り返す
  6. ステップ 5 の結果を初期値に、k-means 法を実行

## ■ クラスタリング後のデータのばらつきを表す指標

- クラスタの中心点に対する二乗誤差
- クラスタ内の分散に相当

$$SSE = \sum_{i=1}^n \sum_{j=1}^k w^{(i,j)} \|x^{(i)} - \mu^{(j)}\|_2^2, \quad w^{(i,j)} = \begin{cases} 1 & x^{(i)} \in C_j \\ 0 & x^{(i)} \notin C_j \end{cases}$$

$n$  : データ数

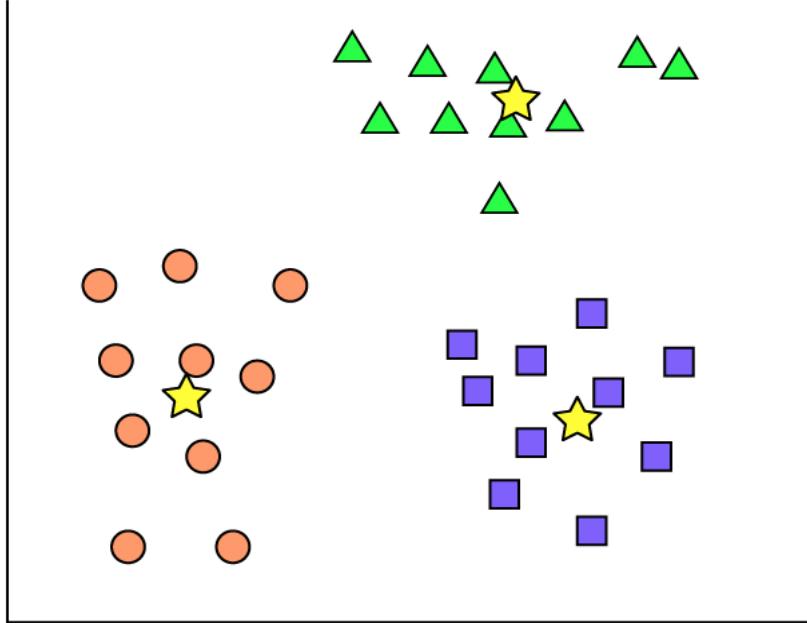
$k$  : クラスタ数

$x^{(i)}$  :  $i$  番目のサンプル点

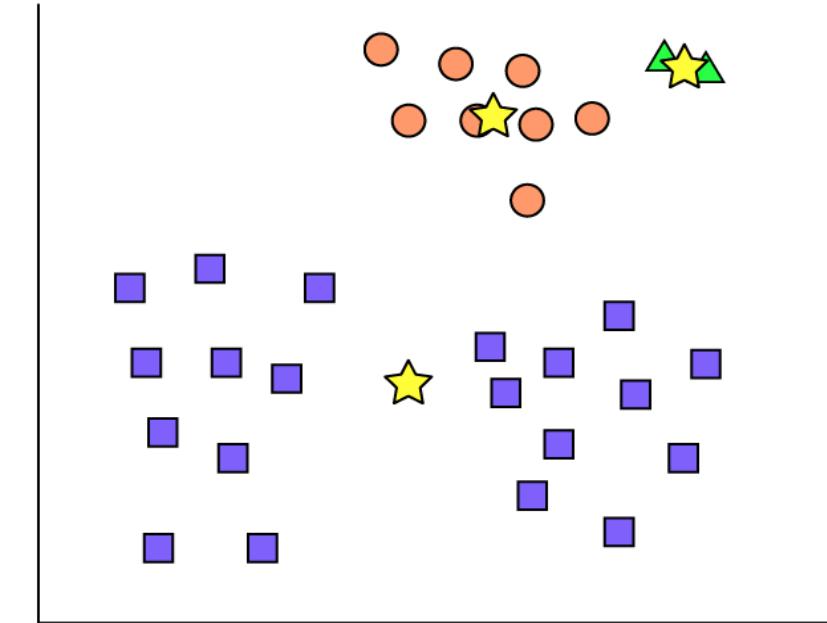
$\mu_j$  :  $j$  番目のクラスタの中心点

$C_j$  :  $j$  番目のクラスタ

理想的なケース

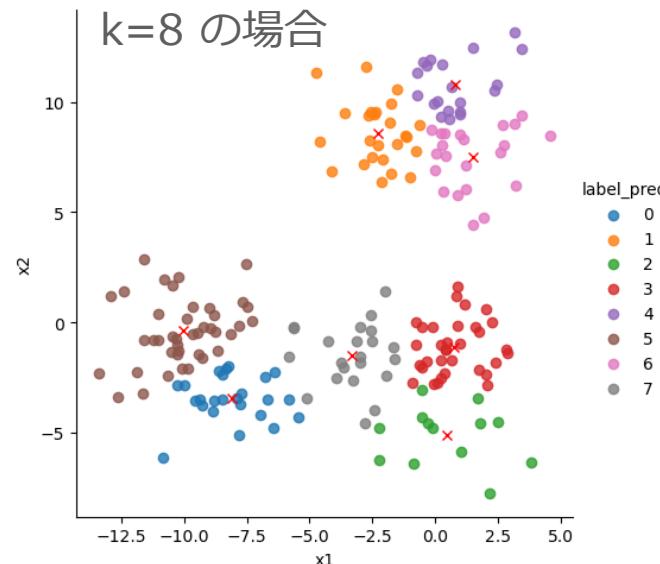


分散が大きいまま収束

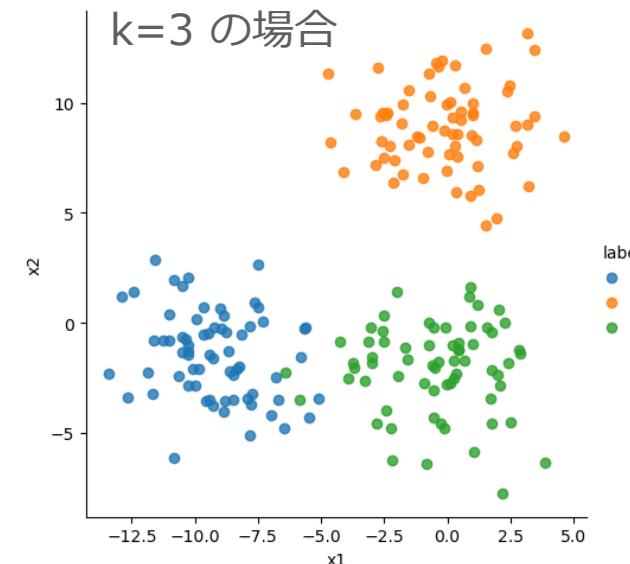


# k-means 法 | 最適な k の決定

- 中心点は、クラスタ内誤差平方和が最小となる位置に更新される
  - 値が小さいほど、まとまりが良い（ばらつきが少ない）と考えられる
- ただし「ばらつきが少ないほど良い」というケースばかりではない点に注意
  - **k の値が大きければ**、ばらつきは少なくなる
  - 分布に規則性が見られるなら、**規則への適合度合い**が重要



規則性を無視している  
(k の値が大きすぎる)



分布の規則に適合している  
(k の値が適正)

## ■ 所定のビジネスニーズがある場合は、その内容に従う

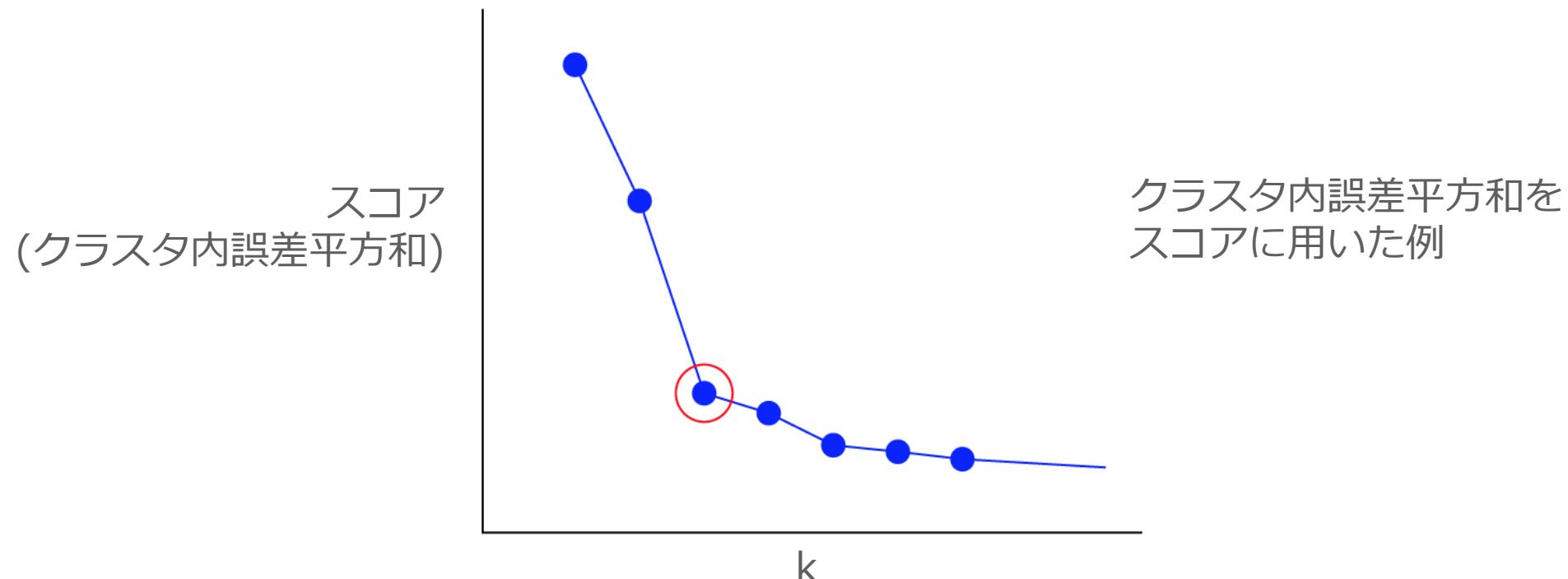
- 例) A, B, C の 3 つの広告プランがあり、それに合うように顧客を分けたい場合
  - まずは  $k = 3$  に設定してクラスタリングを行う
  - 出力が要件に合わない場合は、 $k$  を 3 以上に設定して実行
    - 何らかの尺度に基づき、クラスタを再度 3 つに分類する

## ■ 事前に k を決められない場合は、地道に探索する

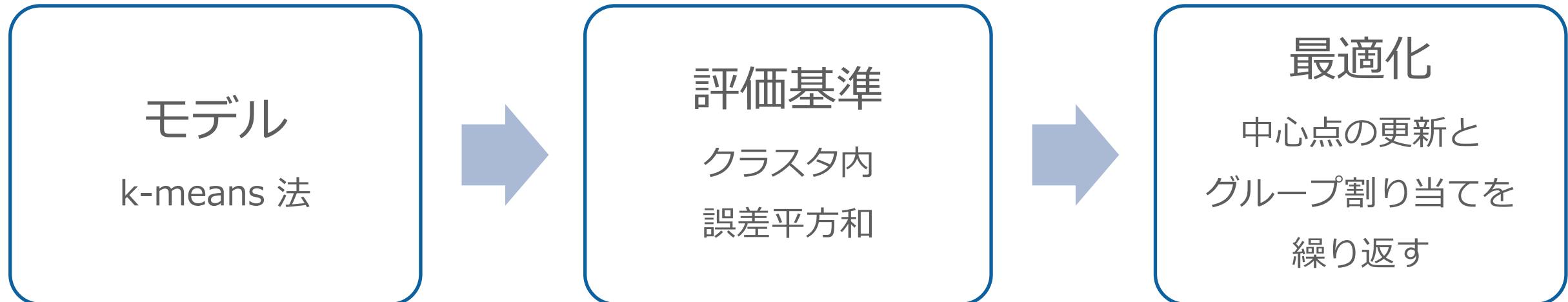
- 例) クラスタリングが次の分析への前工程である場合
  - $k$  をある範囲で動かして、クラスタリングをまとめて行う
    - 後でスコアを一覧して良いものを選択
    - エルボー法を用いて、最適な  $k$  の目星を付ける
  - 逐次スコアを確認しながら、スコア改善が見られなくなるまで  $k$  を増加させ続ける

## ■ 最適なクラスタ数 $k$ を探索するための手法

- ・ 「妥当な」  $k$  の値まで実際に学習をおこない、スコアを後から評価する
- ・ スコアが急激に減少する部分（肘）をターゲットの  $k$  とする
- ・ 実際はきれいに「肘」が見えるケースは稀で、参考程度



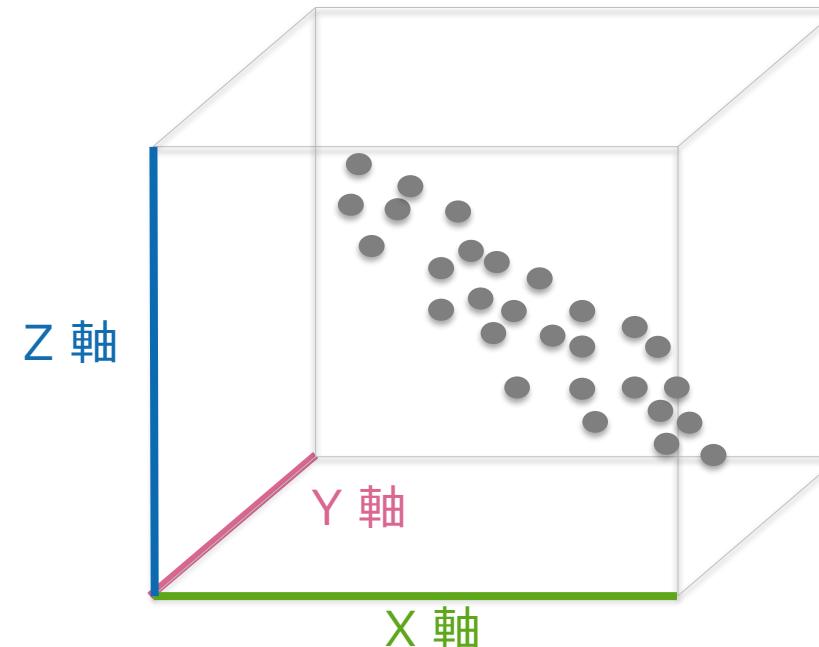
- クラスタリングの手法の一つ
- 以下の操作を繰り返して、データを  $k$  個のグループに分ける
  - グループの中心点の配置
  - データのグループへの割り当て
  - グループ内の中心点の更新
- グループの個数  $k$  は分析者が決定



# 主成分分析

---

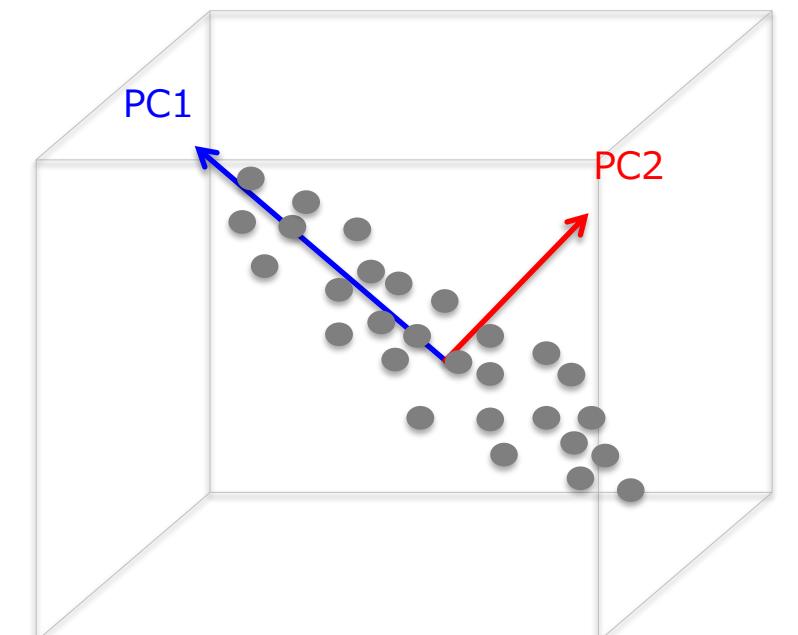
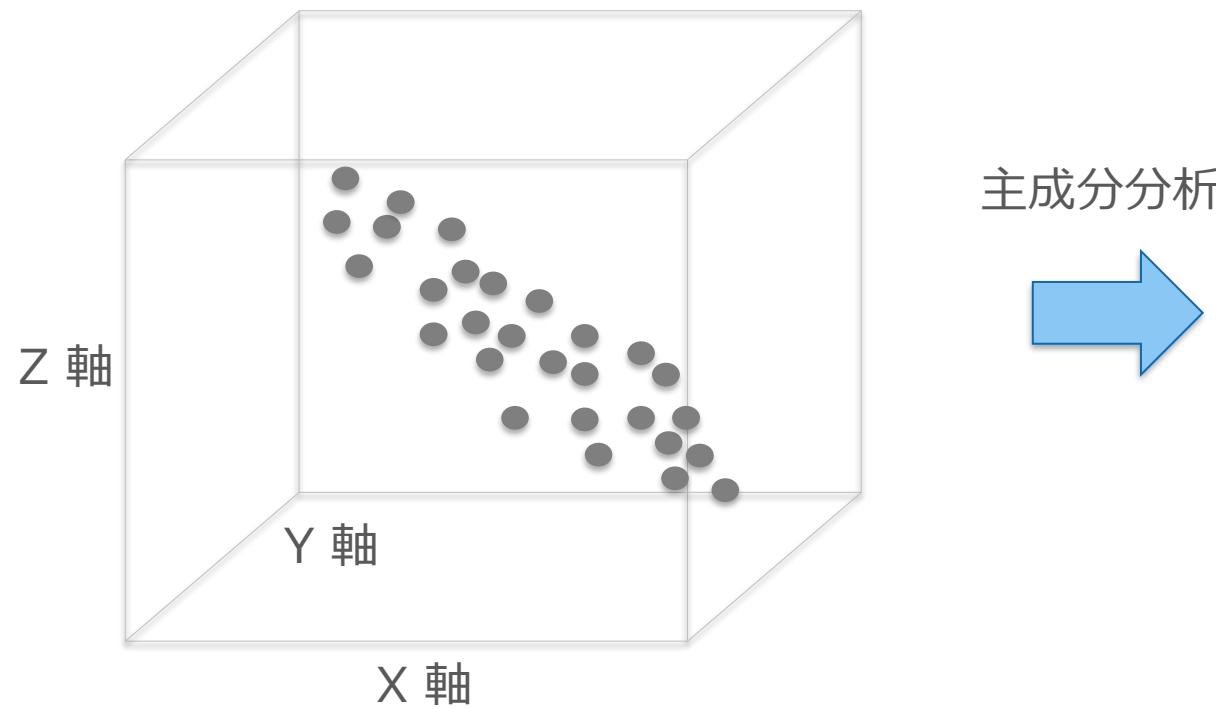
- 以下のように、X 軸・Y 軸・Z 軸で表現される 3 次元データがある
- データの次元を減らして、低次元でも意味のある表現を獲得したい
- もし、自由に軸を設定できるとすればどのように軸を置くべき？
  - 1 次元（=軸 1 本）だったらどこに置く？ 2 次元（=軸 2 本）だったら？



- 元の変数に重みをかけた新しい合成変数（主成分）を作成し、次元を減らす手法
- 主成分軸は「**固有値問題**」という問題を解くことで求まる
- 主な用途
  - データの次元削減
  - 多次元データを低次元（2 次元のグラフなど）で可視化

## ■ X 軸、Y 軸、Z 軸の 3 次元データを、2 次元平面に変換

- 分散が最大となるような、新しい軸を探索する
- 分散が大きい=そのデータを特徴づける方向



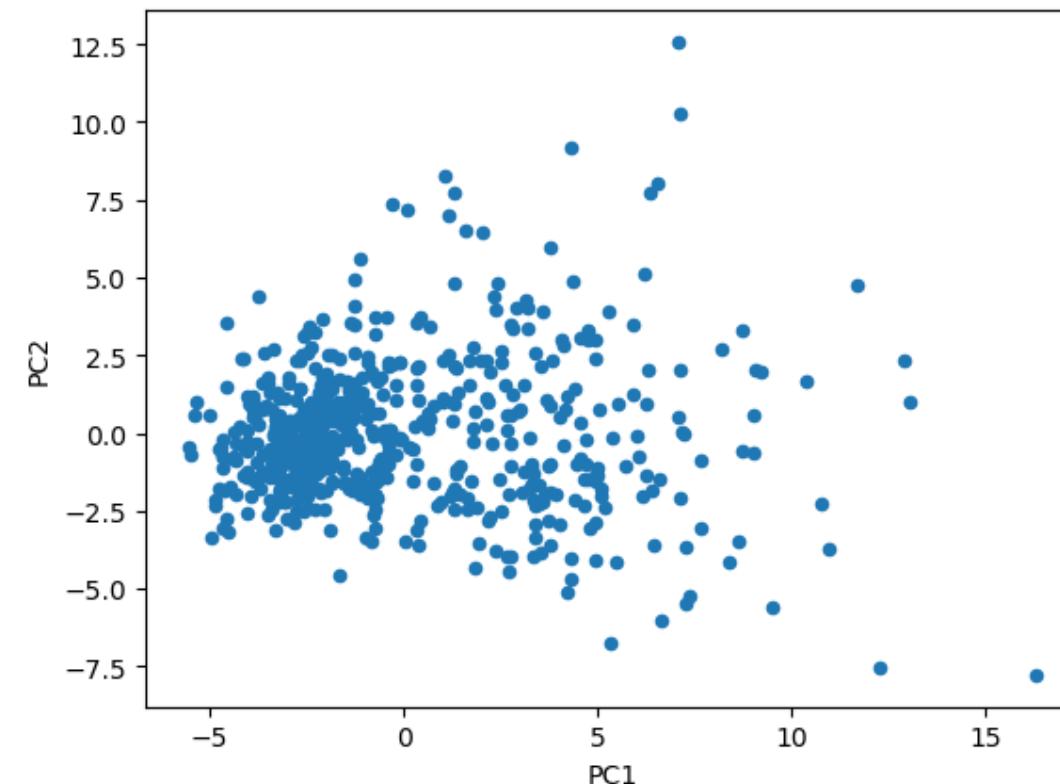
PC1 軸と PC2 軸の  
2 次元空間（平面）に変換

- PC1 は第一主成分軸、PC2 は第二主成分軸と呼ばれる
  - 第一主成分軸は最も分散が大きい方向を表現
  - 第二主成分軸は 2 番目に分散が大きい方向を表現
  - 元データが N 次元の場合、第 N 主成分まで求めることが可能
- 各主成分軸は必ず直交する（内積が 0 になる）
  - 固有ベクトルを用いているため
- 一部の主成分軸のみを取り出せば、次元削減が可能
  - 高次元データを低次元データに変換して、可視化できる

## ■ 各データ点における、主成分の値

- この値を用いて、データの性質をある程度把握することができる
- 例) 30 個の変数をもつ (30 次元の) データを 2 次元に圧縮した場合

主成分得点の散布図



## ■ ある主成分に対する固有値を、全主成分の固有値の合計で割ったもの

- 値が 1 に近いほど、変換前のデータを再現するにあたって重要であることを表す
- 主成分得点の分散が大きいほど、寄与率も大きい
- 寄与率の高い順に、各主成分を第一主成分、第二主成分…と呼ぶ

## ■ 第 N 主成分までの寄与率の合計を累積寄与率という

- 次元削減の際に、主成分軸をいくつ取り出すべきかの判断基準に用いる
- この値が低くなりすぎないように軸を選べば、適切な次元削減を行える

- 分散が最大となるような、新しい軸を探索する
- 処理手順（一部、無相関化と共通）

- ① データ  $X$  の分散共分散行列  $\Sigma$  を算出
- ② 分散共分散行列を固有値分解し（**固有値問題**を解く）  
固有ベクトルの集まり  $P$  と固有値の集まり  $D$  を抽出
- ③ 固有値の大きい順に、 $P$  から一部の**固有ベクトル**を取り出して、**主成分**とする

5次元以上のデータに対しては  
解析的に解くことができない  
(出力は近似値となる)

$$\textcircled{1} \quad \Sigma = E[(X - \mu)(X - \mu)^T]$$

$$\textcircled{2} \quad P^T \Sigma P = D$$

$\mu$  : データ  $X$  の各次元における平均  
 $P$  :  $\Sigma$  の固有ベクトルを列とする直行行列  
 $D$  :  $\Sigma$  の固有値を対角成分にもつ行列

データ  $X$  の例

変数 1	変数 2	変数 3	...
0.839	3.145	9.642	...
1.246	0.536	1.325	...
...	...	...	...
9.789	8.568	1.235	...

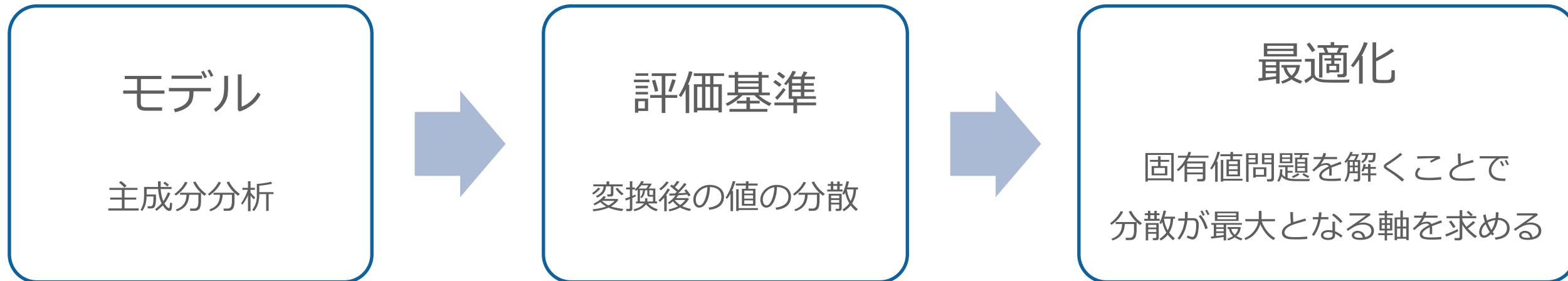
- 数理的には、元の変数と**主成分**の関係を表すモデルのパラメータ  $w$  を求めることに相当

- $PC1 = w_{11}x_1 + w_{21}x_2 + w_{31}x_3$
- $PC2 = w_{12}x_1 + w_{22}x_2 + w_{32}x_3$
- $PC3 = w_{13}x_1 + w_{23}x_2 + w_{33}x_3$

主成分 (PC1~3) の分散が最大となるように  $w$  の各値を求める

- この  $w$  を**主成分負荷量**という
  - 各主成分の計算に、どの変数をどのくらい使用しているかを表す値
  - 各主成分がどのような意味合いをもつのか解釈する際に利用
- 一度このモデルを作成しておけば、固有値問題を解くことなく新たなデータに同様の変換を適用できる

- 元の変数に重みをかけた合成変数（主成分）を作成し、次元軸を減らす手法
- データの次元削減やデータの特性を抽出するのに役立つ
  - 累積寄与率が高くなるように次元を選ぶ
- 多次元データを低次元（2次元のグラフなど）で可視化できるようになる



## 自己符号化器

---

## 決定的

### 階層型

全結合型ニューラルネットワーク  
Fully connected neural network

畳み込みニューラルネットワーク  
Convolutional neural network

再帰型ニューラルネットワーク  
Recurrent neural network

### 自己符号化器型

自己符号化器  
Autoencoder

雑音除去自己符号化器  
Denoising Autoencoder

変分自己符号化器  
Variational Autoencoder

スパース自己符号化器  
Sparse Autoencoder

## 確率的

### ボルツマンマシン型

ボルツマンマシン  
Boltzmann machine

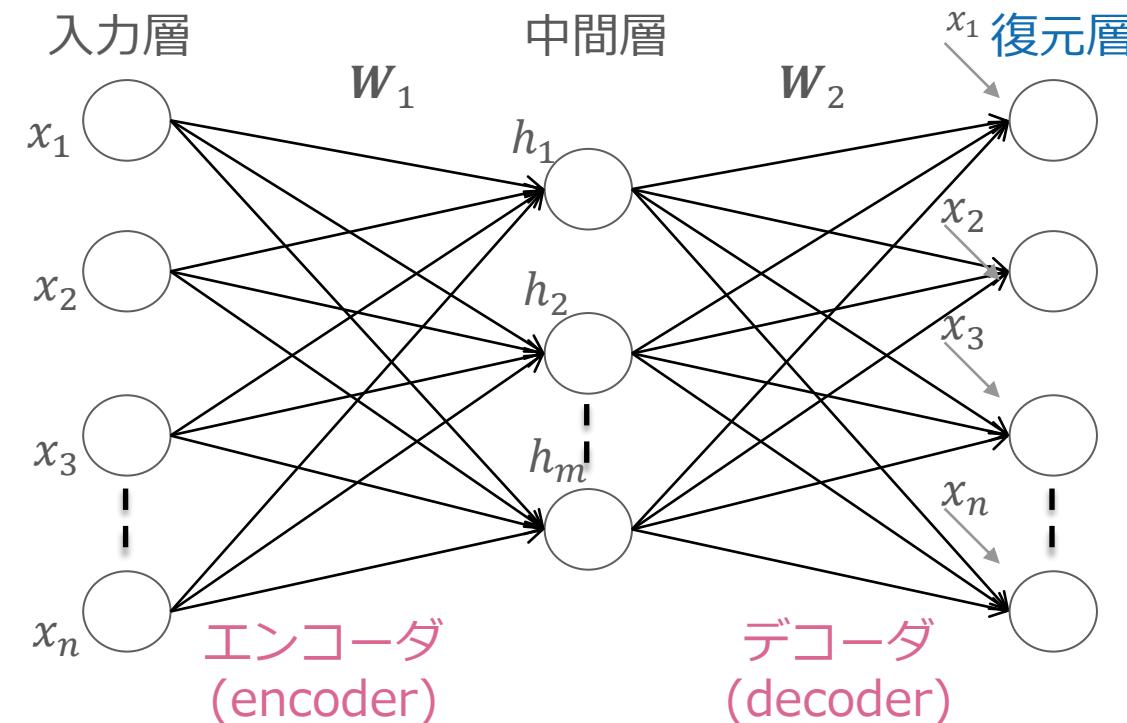
制約ボルツマンマシン  
Restricted Boltzmann machine

深層信念ネットワーク  
Deep belief network

深層ボルツマンマシン  
Deep Boltzmann machine

参考：『深層学習』(神鳴ら、近代科学社)

- 入力層・中間層・復元層で構成されるニューラルネットワーク
  - 入力層～中間層をエンコーダ、中間層～復元層をデコーダと呼ぶ場合もある
- 正解ラベルの代わりに、入力データと同じものを与えて学習
  - 評価基準は、通常の回帰問題と同じく二乗誤差
  - 勾配降下法を用いて最適化



## ■ データを入力した際の、中間層の出力 $h$ を利用

- 特徴抽出
- 次元圧縮

## ■ データを入力した際の、復元層の出力を利用

- ノイズ除去
- 異常検知（復元誤差を利用）

## ■ 雑音除去自己符号化器 (Denoising Autoencoder)

- ・ ノイズを付加したデータを入力層に入れる方法
- ・ 復元層の出力が、ノイズを付加する前のデータに近くなるように学習させる

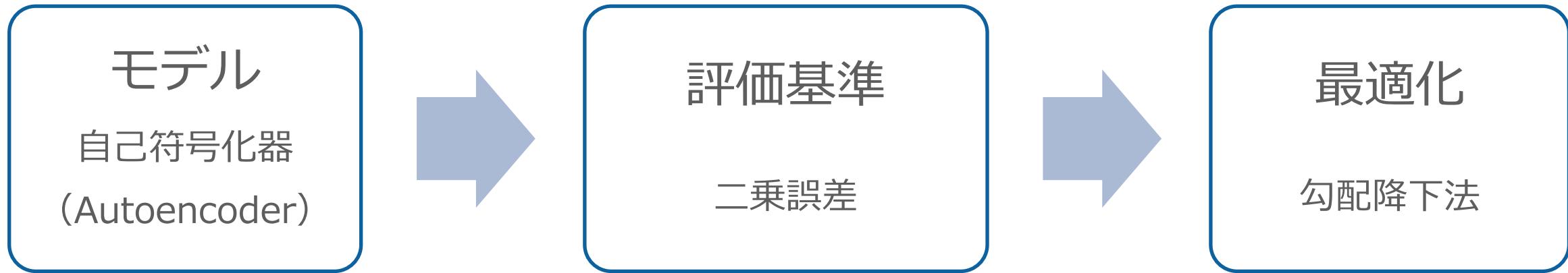
## ■ 変分自己符号化器 (Variational Autoencoder)

- ・ 新たなデータを生成する、生成モデルの一種
- ・ 中間層の出力を、潜在変数の確率分布として用いる

## ■ スパース自己符号化器 (Sparse Autoencoder)

- ・ 自己符号化器に正則化項を加えて学習させる方法
- ・ 効率的にユニット数（次元数）を少なくできる

- 入力層・中間層・復元層で構成されるニューラルネットワーク
  - 入力層と復元層のユニット数は同じ
  - 中間層のユニット数を少なくしておく
- 入力データを復元できるように学習
  - 正解ラベルの代わりに、入力データと同じものを与える
  - 中間層の出力を取り出すことで、次元削減に利用できる

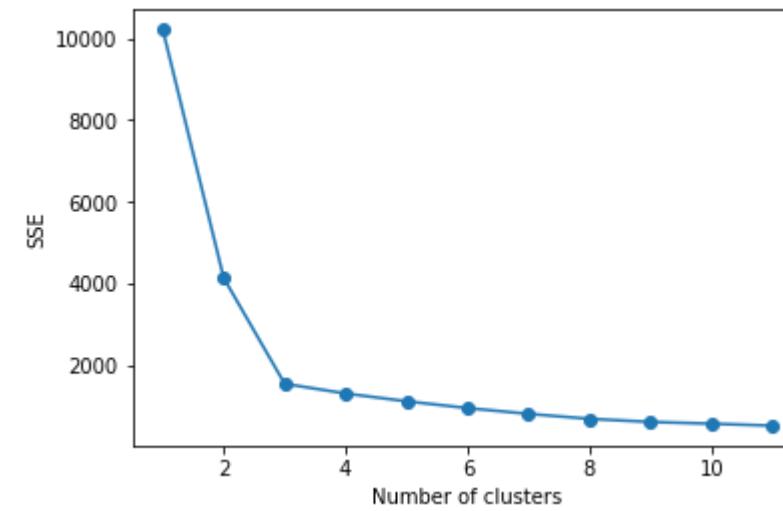
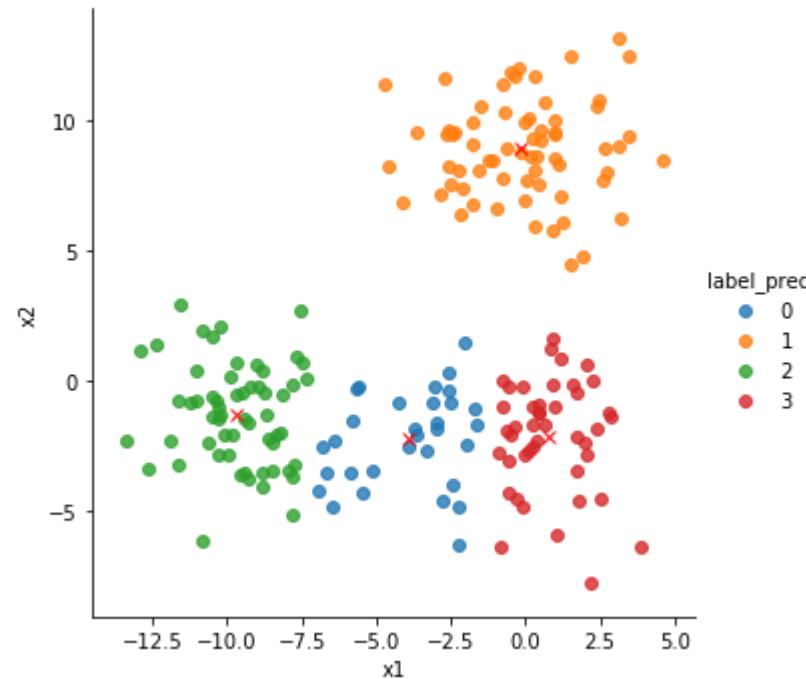


## ノートブック演習

---

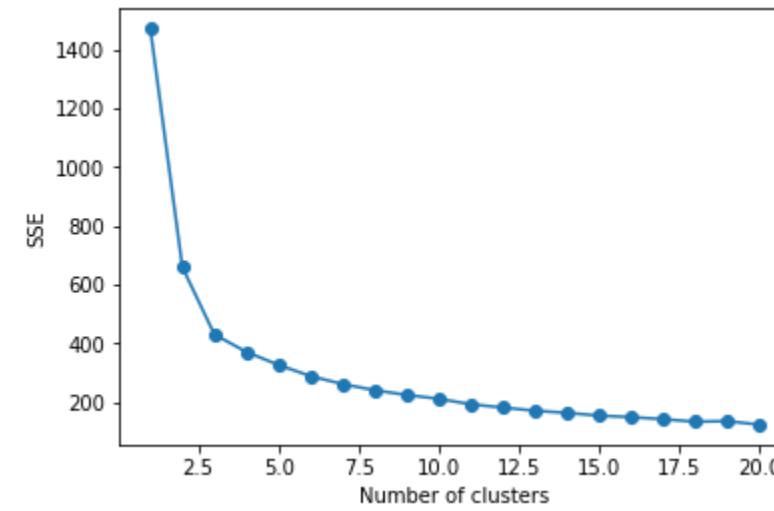
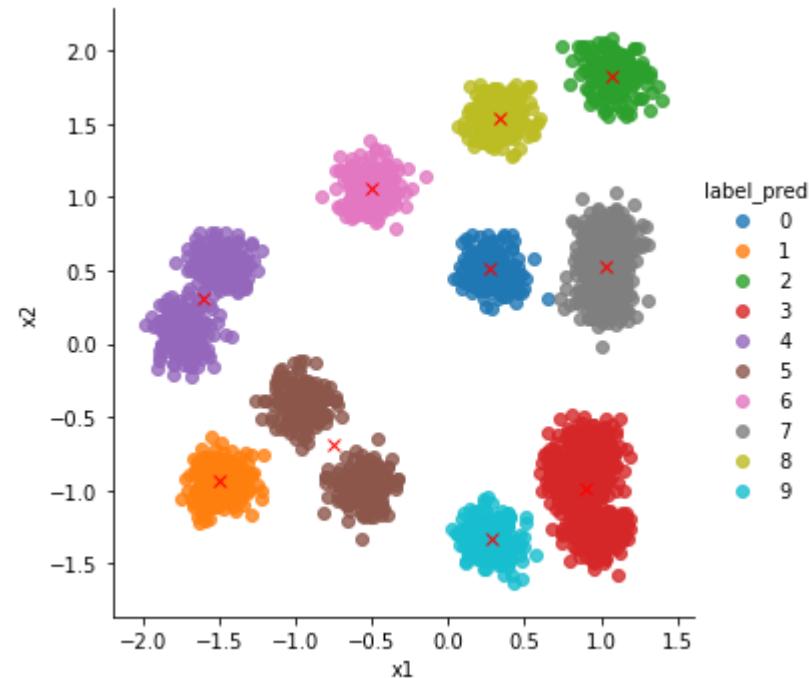
## 14-1\_k-means\_artificial.ipynb

- k-means 法の実装を確認してみよう
- 人工的なデータを用いて、実際に k-means 法を試してみよう



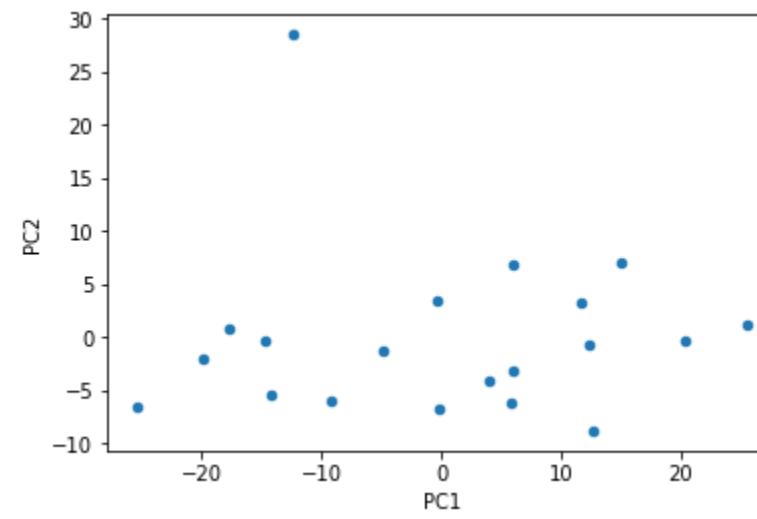
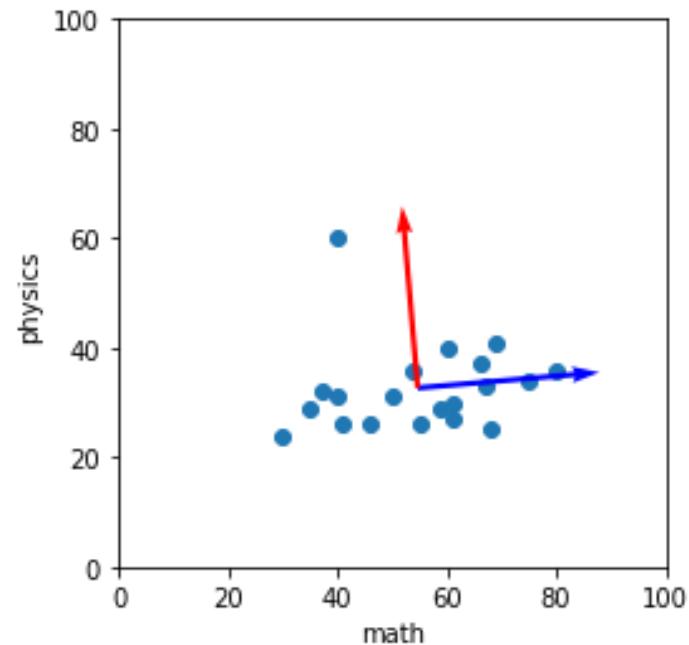
## 14-2\_k-means\_real.ipynb

- 実際のデータを用いて、k-means 法でクラスタリングしてみよう
- 分割数が既知、未知のそれぞれのケースで k-means 法によるクラスタリングとデータの分類を行ってみよう



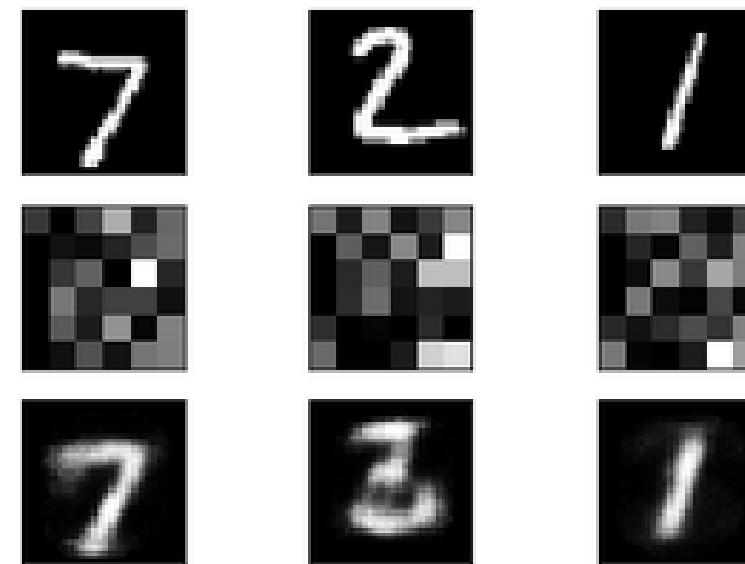
## 14-3\_PCA.ipynb

- 主成分分析によって、データの次元削減を行ってみよう
- 寄与率を確認し、重要な特徴とはなにかを考えてみよう



## 14-4\_AutoEncoder.ipynb

- 自己符号化器を用いて、手書き数字画像のデータを圧縮してみよう
- パラメータを変更すると、デコーダから再構成される画像がどのように変化するか試してみよう



## 本章のまとめ

---

- クラスタリング
- k-means 法
- 主成分分析
- 自己符号化器
- ノートブック演習

## ■ k-means 法

- クラスタリングの代表的な手法
- 以下の操作を繰り返して、データを  $k$  個のグループに分ける
  - グループの中心点の配置
  - データのグループへの割り当て
  - グループ内の中心点の更新
- グループの個数  $k$  は分析者が決定

## ■ 主成分分析

- ・ 各次元の分散が最大となるようにデータを変換
- ・ 次元削減に用いる
- ・ 累積寄与率が高くなるように次元軸を選ぶ

## ■ 自己符号化器

- ・ 入力データを復元できるように学習させる
- ・ 中間層の出力を取り出すことで、次元圧縮を行える
- ・ ノイズ除去や異常検知への応用が可能

# 現場で使える 機械学習・データ分析基礎講座

第 15 章 : AutoML

- AutoML とは？
- 代表的な AutoML サービス
- 代表的な AutoML 用 Python ライブラリ
- 講座全体のまとめ

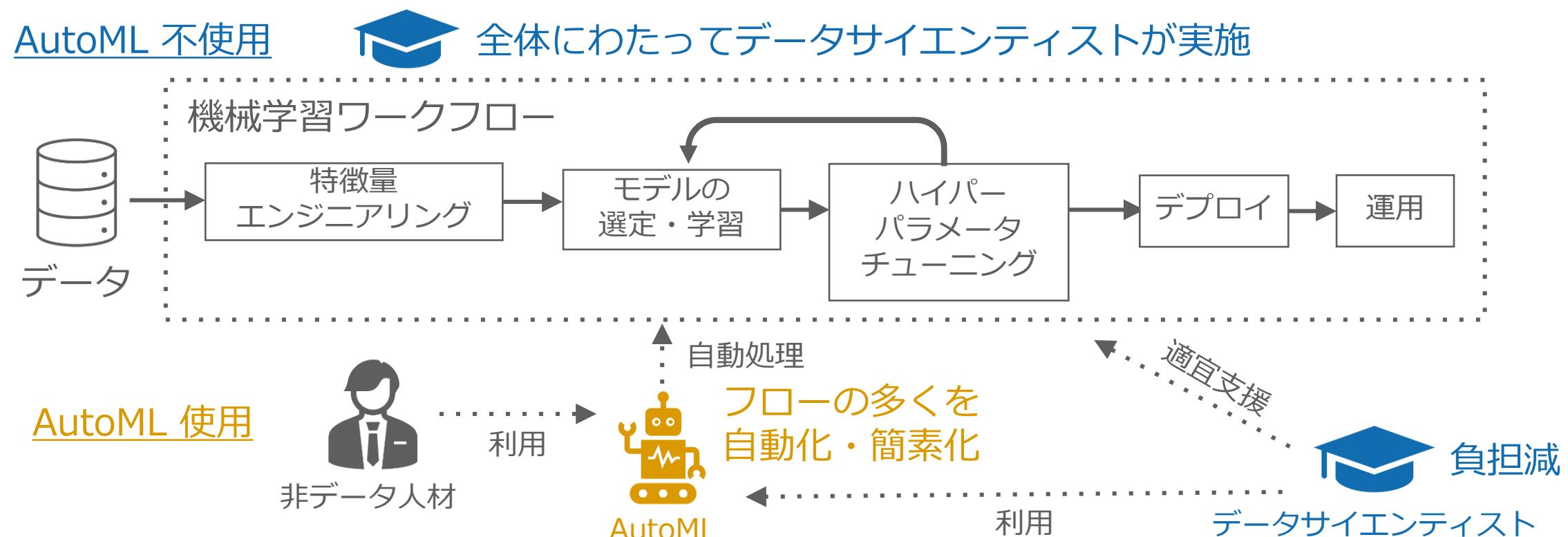
- AutoML に関する概要と、導入する利点を説明できる
- 代表的な AutoML サービスを列挙し、説明できる
- 代表的な AutoML 用 Python ライブラリを列挙し、説明できる

# AutoML とは？

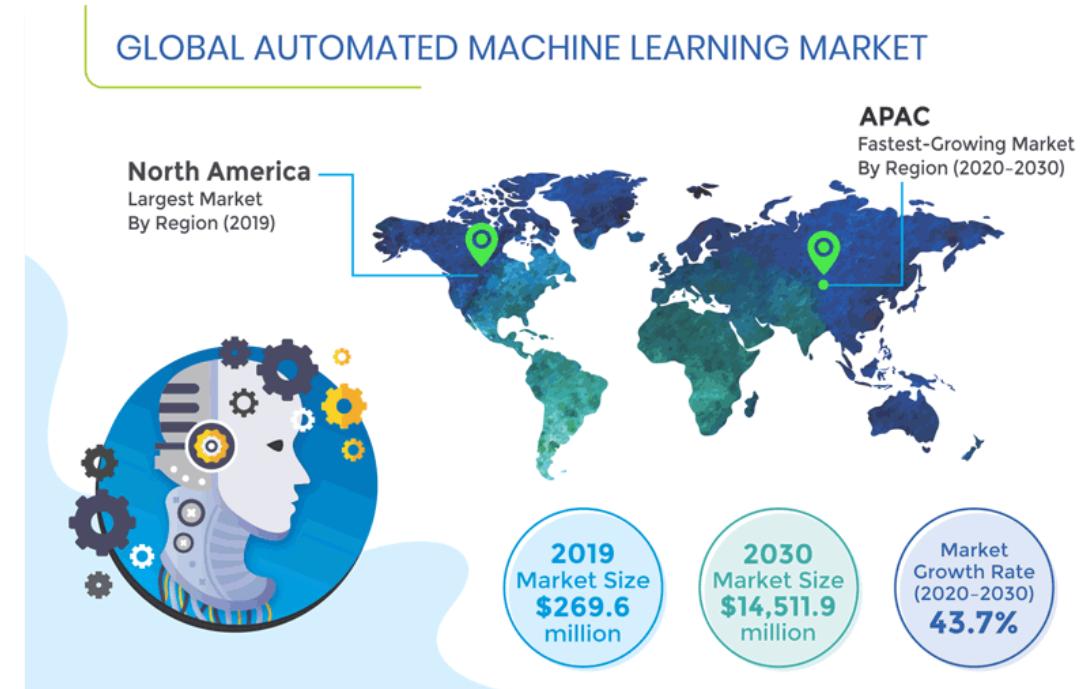
---

## ■ AutoML = Automated Machine Learning (自動化された機械学習)

- ・ 非データサイエンティストにも機械学習を利用した課題解決の一部が実現可能
- ・ データサイエンティストが反復的な作業から解放され、生産的なタスクに集中できる



- 世界的に見ても AutoML への期待やその注目度は高い
- 世界の AutoML 市場の収益は、2030 年に約 145 億ドル（約 1 兆 6000 億円）になると言われている
  - 市場調査会社 Prescient & Strategic Intelligence のレポート（2020 年）より



出典：[Automated Machine Learning Market Share Forecast Report, 2030](#)

1

## データサイエンティストの育成・採用コストを下げられる※

- 非データサイエンティストからの育成が容易
- データサイエンティストの採用コストを下げやすい
- データサイエンティストの育成・採用にかかる時間を短縮できる

2

## AI の内製化を実現しやすい

- ノウハウが社内に蓄積される
- PDCA サイクルを何度も回せる（試行錯誤できる）
- モデルやデータの権利問題が発生しづらい

3

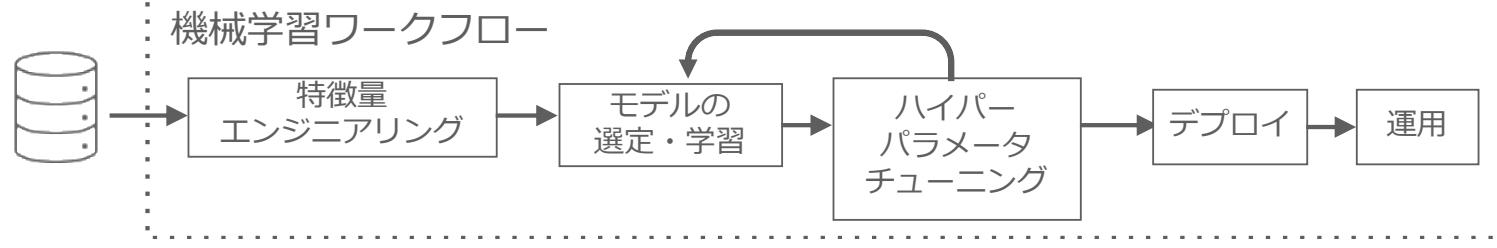
## データサイエンティスト個人への依存度を下げられる

- モデル作成における技術レベルの均質化ができる（誰がやっても一定のレベルが保証できる）
- 事業継続性がある組織を作りやすい（退職に対してロバストな組織にしやすい）

※育成・採用が「不要」ではないことに要注意！

- 物体検出や自然言語処理など、深層学習を中心とした高度な AI 技術には、現時点の AutoML ツールは、ほぼ対応していない
- そのため高度な AI 技術が必要な場合には採用・育成は必須
- そうでなくとも、AutoMLを使いこなすためのリテラシーは必要（G検定・DS検定リテラシーレベル合格程度）

- 「AutoML」が指す内容は、企業が提供するサービスやシステムの場合と学術分野・OSSの場合で違いがある
- 必要なものを正しく見定めるためにもまずは「AutoML」の指す内容の違いを理解しよう

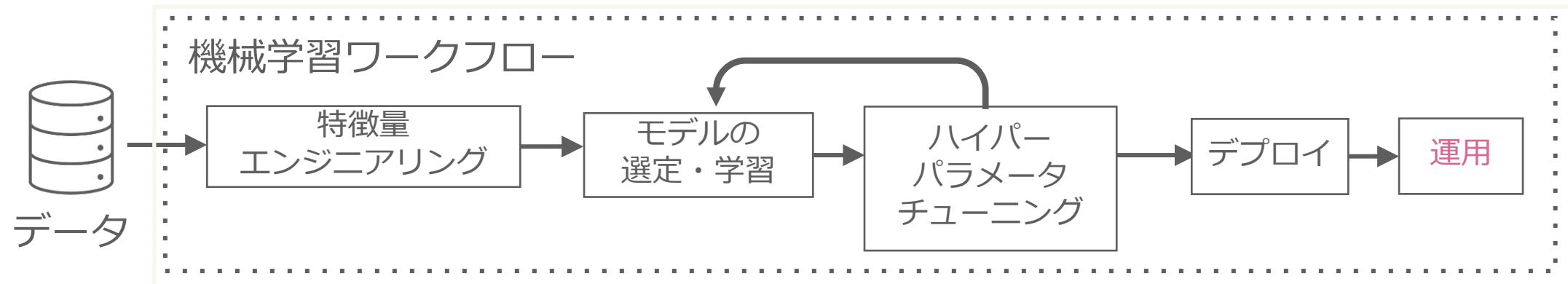


それぞれの場合における違いは  
何だろうか？



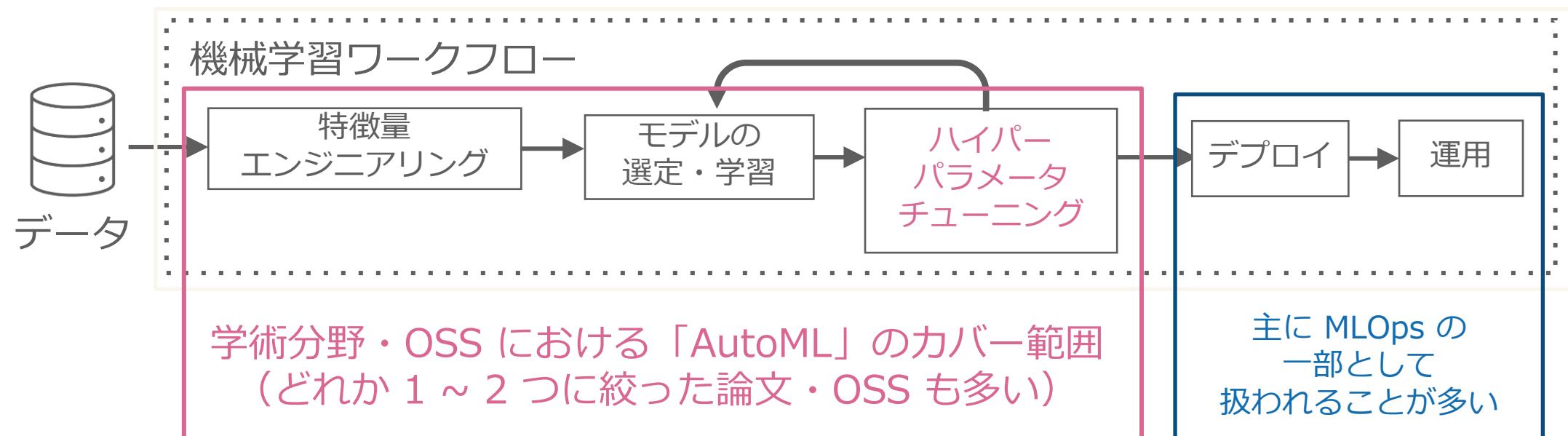
- いわゆる「AutoML サービス」と呼ばれるものは、特徴量エンジニアリングから運用までトータルでサポート
- 機械学習ワークフロー全体を支援することが目的

## 「AutoML サービス」のカバー範囲



- 一方で、学術分野やオープンソースソフトウェア（OSS）では、特徴量エンジニアリングからハイパーパラメータチューニングを指すことが多い
- あくまで「モデルの作成」を支援することが目的

## 「AutoML サービス」のカバー範囲



## **代表的な AutoML サービス**

---

## ■ 特徴

- ・ 「ビジネス活用」に特化した ROI 算出機能

## ■ 実行環境

- ・ クラウド / オンプレミス

## ■ 対応するタスク

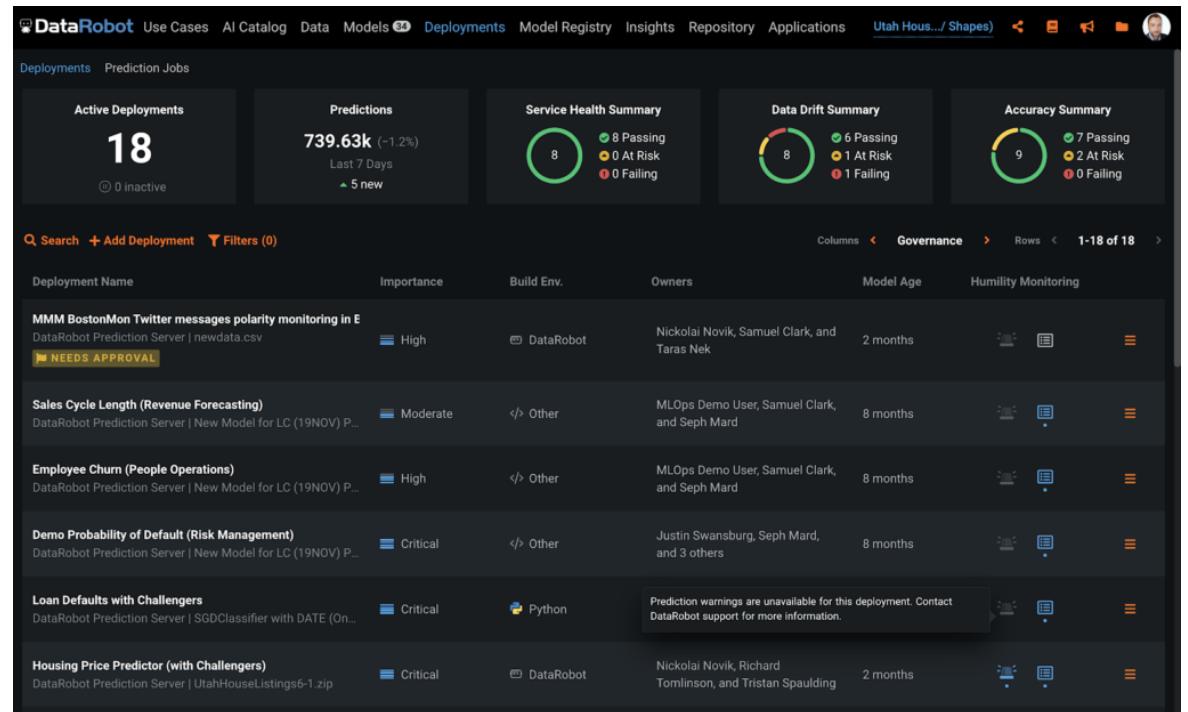
- ・ 回帰、分類

## ■ 対応するデータ形式

- ・ テーブル、画像、時系列、テキスト

## ■ 價格

- ・ 販売代理店・プランによって異なる
- ・ 30 日間の無料トライアルあり



出典：<https://www.datarobot.com/jp/>

## ■ 特徴

- 予測結果の理由を、理解しやすい形式で表示

## ■ 実行環境

- クラウド / オンプレミス

## ■ 対応するタスク

- 回帰、分類

## ■ 対応するデータ形式

- テーブル、時系列

## ■ 價格

- [公式サイト](#)を参照
- 無料体験版あり



出典：<https://predictionone.sony.biz/>

## ■ 特徴

- ・ 入出力において、様々なツール（Excel や BI ツールなど）と連携可能

## ■ 実行環境

- ・ クラウド

## ■ 対応するタスク

- ・ 回帰、分類、需要予測

## ■ 対応するデータ形式

- ・ テーブル、時系列

## ■ 價格

- ・ [公式サイト](#)を参照



出典：<https://www.datarobot.com/jp/>

## ■ 特徴

- ・ 情報量の多いダッシュボード、幅広いデータ形式への対応

## ■ 実行環境

- ・ クラウド / オンプレミス

## ■ 対応するタスク

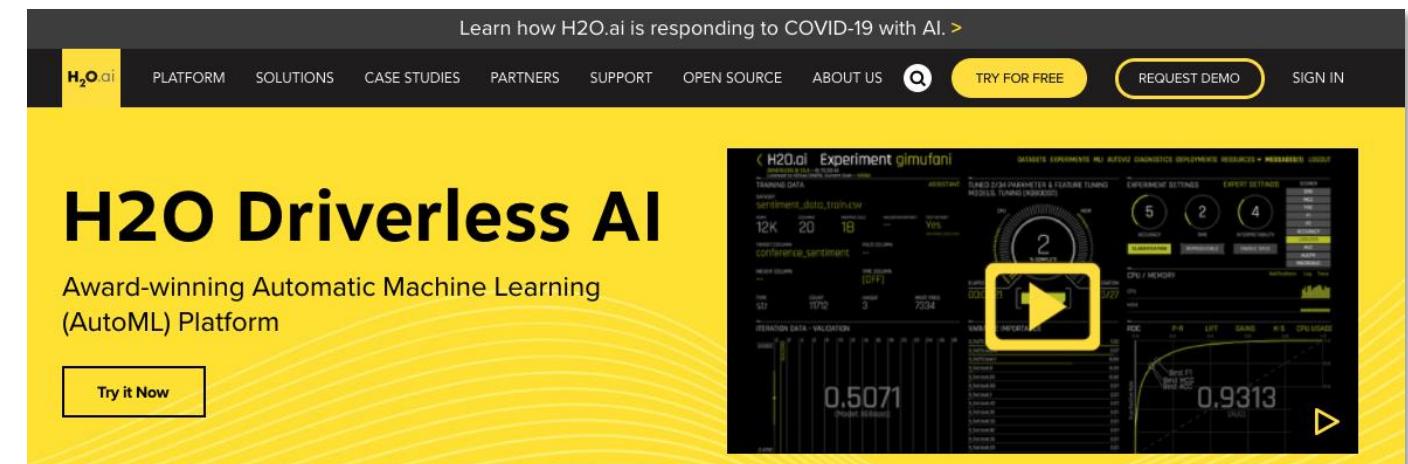
- ・ 回帰、分類

## ■ 対応するデータ形式

- ・ テーブル、画像、時系列、  
テキスト、動画、音声、グラフ

## ■ 価格

- ・ 非公開
- ・ 21 日間の無料トライアルあり



出典：<https://www.h2o.ai/products/h2o-driverless-ai/>

## ■ 特徴

- 独自開発の特徴量自動設計技術

## ■ 実行環境

- クラウド / オンプレミス

## ■ 対応するタスク

- 回帰、分類、推薦

## ■ 対応するデータ形式

- 非公開

## ■ 価格

- 非公開
- 無料トライアルあり（日数は非公開）



出典：<https://dotdata.com/>

## ■ 特徴

- Azure の各種サービスとの連携や、XAI・公平性・プライバシー機能との連携

## ■ 実行環境

- クラウド

## ■ 対応するタスク

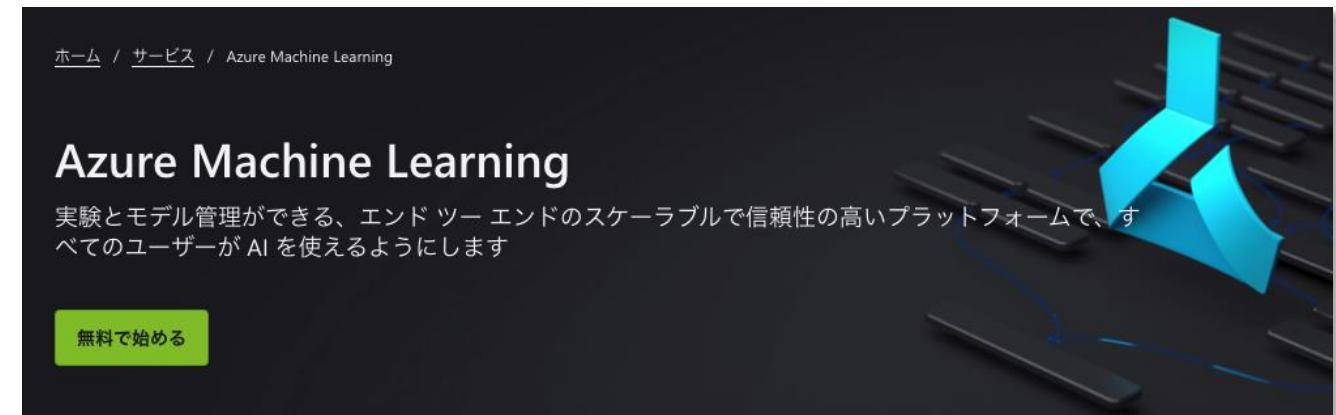
- 回帰、分類

## ■ 対応するデータ形式

- テーブル、時系列

## ■ 価格

- AutoML 機能は無料
- 学習時に利用するインスタンスやストレージの利用料金は別途必要



出典：<https://docs.microsoft.com/ja-jp/azure/machine-learning/concept-automated-ml>

## ■ 特徴

- SageMaker に属する MLOps に関するサービスとの連携

## ■ 実行環境

- クラウド

## ■ 対応するタスク

- 回帰、分類

## ■ 対応するデータ形式

- テーブル

## ■ 価格

- SageMaker Autopilot 自体は無料
- 学習時に利用するインスタンスやストレージの利用料金は別途必要



出典：<https://aws.amazon.com/jp/sagemaker/autopilot/>

## ■ 特徴

- 各種タスクに対応した Auto ML サービス

## ■ 実行環境

- クラウド

## ■ 対応するタスク

- 回帰、分類、翻訳、物体検出

## ■ 対応するデータ形式

- テーブル、画像、動画、テキスト

## ■ 価格

- 価格はサービスによって異なる
- 学習時に利用するインスタンスやストレージの利用料金は別途必要

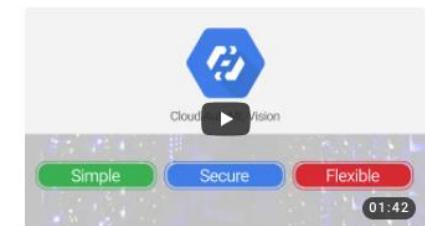
### AutoML

最小限の労力と機械学習の専門知識で、高品質のカスタム機械学習モデルをトレーニングできます。

[ドキュメントを見る](#)

#### カスタム機械学習モデルのトレーニング

AutoML を使用すると、機械学習の専門知識が限られていても、ビジネスニーズに合った高品質のモデルをトレーニングできます。数分で独自のカスタム機械学習モデルを構築します。



AutoML の概要

出典：<https://cloud.google.com/automl>

## 代表的な AutoML 用 Python ライブラリ

---

## ■ 特徴

- 複数の機械学習モデルの学習、比較、ハイパーパラメータチューニングなどを簡単なコードで実行できる

## ■ 対応するタスク

- 回帰、分類、クラスタリング、異常検知、自然言語処理に対応している
- AutoML 機能を提供しているのは、現時点で「回帰、分類」のみ

## ■ 対応するデータ形式

- テーブル、テキスト

## ■ ライセンス

- [MITライセンス](#)
  - 商用利用 OK
  - 利用・改変・再頒布の場合は、著作権表示とライセンスの全文をどこかに記載すること
  - 利用時も含めソースコード公開は不要



出典：<https://pycaret.org/>

## ■ 特徴

- Apache MXNet をベースに作成されたライブラリ
- 幅広いタスクをカバーする

## ■ 対応するタスク：

- 回帰、分類、画像分類、物体検出、テキスト分類/回帰

## ■ 対応するデータ：

- テーブル、画像、テキスト、マルチモーダル

## ■ ライセンス

- Apache 2.0 ライセンス
  - 商用利用 OK
  - 改変・再頒布の場合は、著作権表示とライセンスの全文をどこかに記載すること
  - 利用時も含めソースコード公開は不要



出典：<https://github.com/awslabs/autogluon>

## ■ 特徴

- scikit-learn をベースに作成された AutoML ライブラリ
- 新しい特徴量の構築や次元削減、特徴選択なども自動で実施してくれる

## ■ 対応するタスク

- 分類・回帰

## ■ 対応するデータ

- テーブル

## ■ ライセンス

- [GPL 3.0 ライセンス](#)
  - 商用利用 OK
  - 頒布時には、著作権・ライセンス全文を含めた上で、ソースコードの公開が必要
  - 単に社内で利用する分には公開不要



出典：<https://github.com/EpistasisLab/tpot>

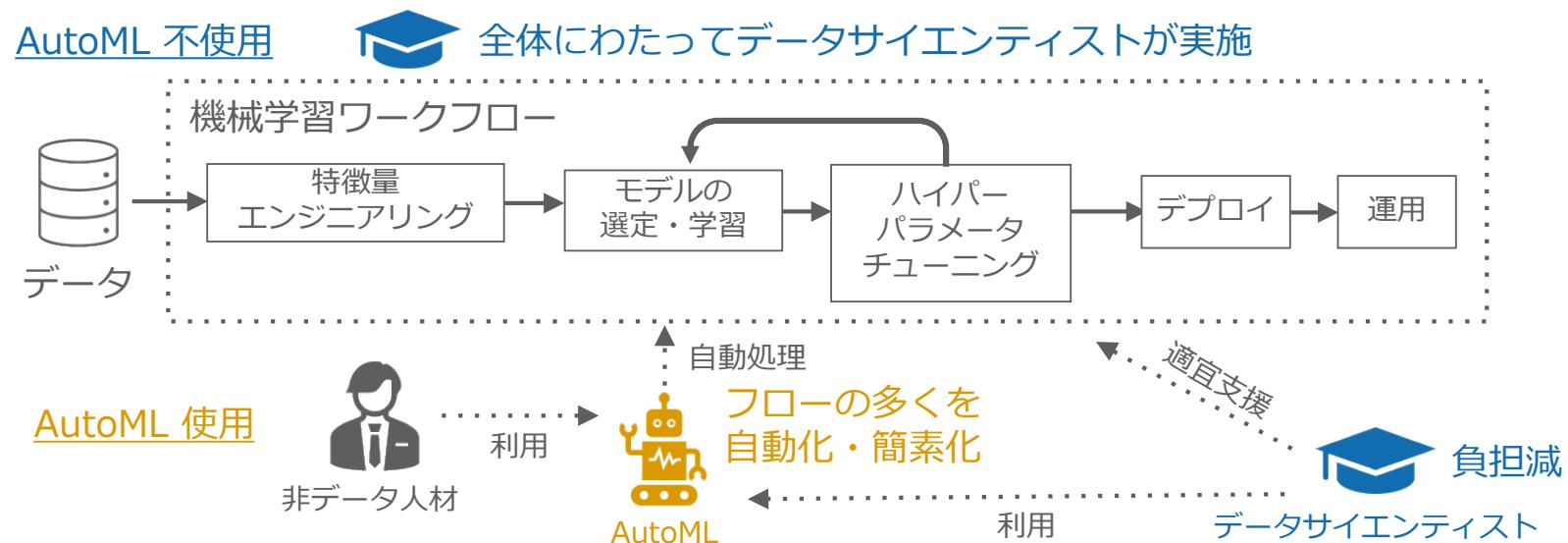
## 本章のまとめ

---

- AutoML とは？
- 代表的な AutoML サービス
- 代表的な AutoML 用 Python ライブラリ

## ■ AutoML

- Automated Machine Learning (自動化された機械学習)
- 導入のメリット
  - データサイエンティストの育成・採用コストを下げられる
  - データサイエンティスト個人への依存度を下げられる
  - AI の内製化を実現しやすい



## ■ 代表的な AutoML サービス

- DataRobot
- Prediction One
- UMWELT
- H2O Driverless AI
- dotData
- Azure Machine Learning AutoML
- Amazon SageMaker Autopilot
- GCP Vertex AI AutoML

## ■ 代表的な AutoML 用 Python ライブラリ

- PyCaret
- AutoGluon
- TPOT

# 講座全体のまとめ

- 第 1 章：機械学習概論
- 第 2 章：教師あり学習の基礎
- 第 3 章：モデルの評価指標
  
- 第 4 章：モデルの検証と正則化
- 第 5 章：モデルの構築と改良
- 第 6 章：代表的な前処理
- 第 7 章：特徴選択

- 第 8 章：決定木
- 第 9 章：アンサンブル学習
- 第 10 章：サポートベクターマシン
  
- 第 11 章：深層学習の概要
- 第 12 章：畳み込みニューラルネットワーク
- 第 13 章：深層学習の代表的な手法
- 第 14 章：教師なし学習
- 第 15 章：AutoML

## ■ 代表的な機械学習モデル

- 教師あり学習
  - 線形回帰モデル
  - ロジスティック回帰モデル
  - k近傍法
  - 木モデル（決定木、ランダムフォレスト、アダブースト、勾配ブースティング木）
  - サポートベクターマシン
  - ニューラルネットワーク（全結合型 NN、CNN、RNN、GAN）
- 教師なし学習
  - k-means 法
  - 主成分分析
  - 自己符号化器

## ■ 機械学習モデルの評価指標

- 回帰
  - MAE
  - MSE
  - RMSE
  - MAPE
  - 決定係数
- 分類
  - Accuracy
  - Recall
  - Precision
  - F1
  - AUC

## ■ 代表的な前処理手法

- 欠損値処理
- 外れ値処理
- カテゴリ変数の変換
- 正規化
- 標準化
- 無相関化、白色化

## ■ 特徴選択

- フィルタ法
- ラッパー法（ステップワイズ法）
- 埋め込み法

## ■ 機械学習モデルの正しい構築手順

- データの可視化
- データセットの分割
  - 訓練用、検証用、テスト用
- データの前処理
- モデルの構築・学習
  - 正則化を行い、過学習を抑える
- モデルの性能を検証
  - ホールドアウト法/交差検証法
  - 結果を踏まえてチューニングを行う
  - ハイパーパラメータ探索の手法を活用
- モデルの最終評価
  - テスト用データを用いる