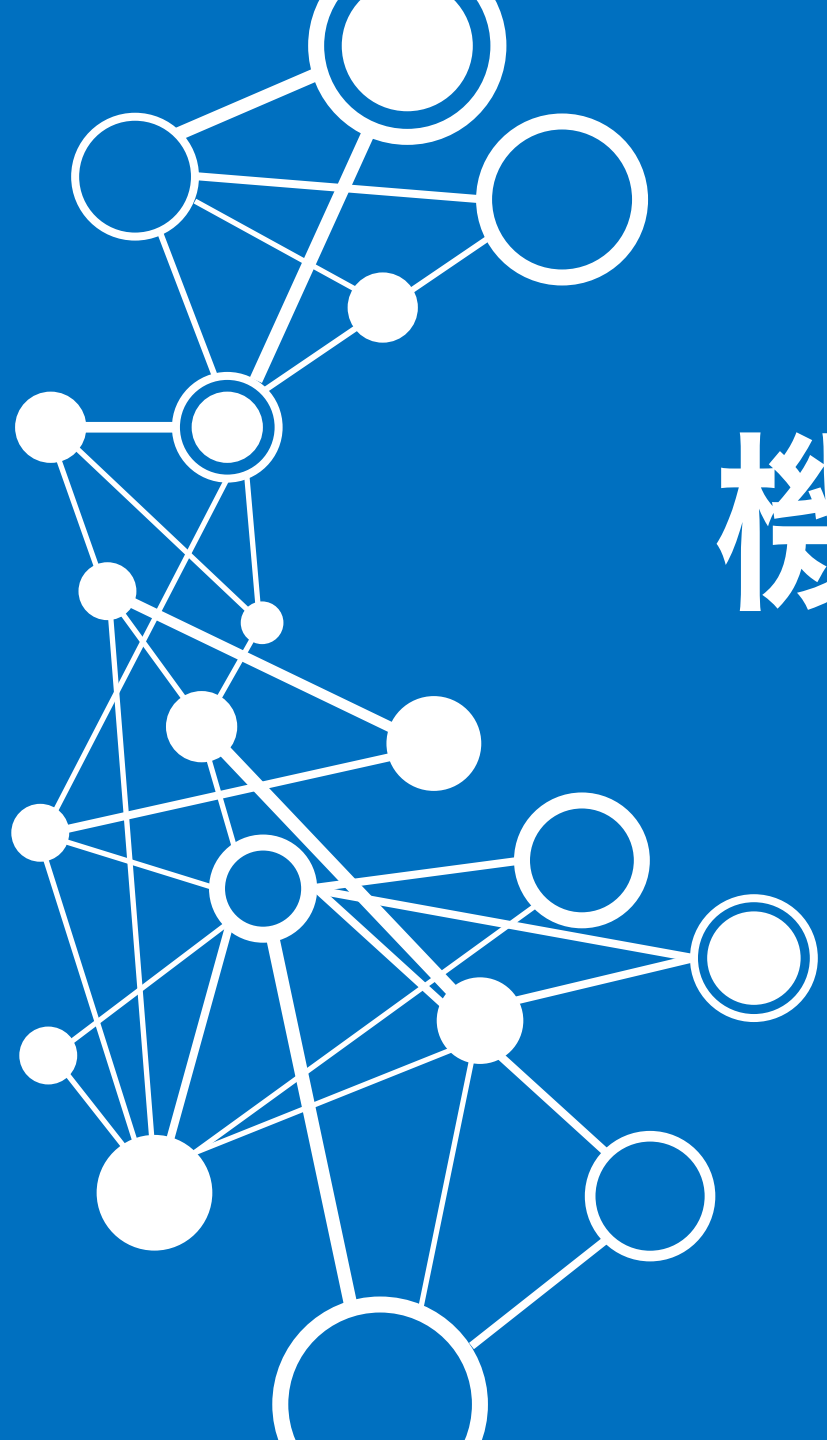


機械学習勉強会

2020-01-14 – 2020-01-17

農研機構・オリエンテーションルーム

農研機構・農業情報研究センター
孫 建強

機械学習概略

- 人工知能と機械学習
- 開発ワークフロー

機械学習概略

- 人工知能と機械学習
- 開発ワークフロー

環境問題	• 種の多様性保全 • 外来種侵入の防止 • 平均気温の上昇抑制 • グリーンエネルギー参画	虫取り網　統計モデリング 温度センサー　紫外線センサー クーロンの法則　ドローン エネルギー保存の法則
食料問題	• 環境変動に頑健な品種改良 • 食料廃棄量の削減 • 農耕地の効率的な利用 • 農家支援	車　飛行機　ベイズ理論 自動運転ロボット　衛星
医療問題	• 医療業務の効率化 • 診断治療の効率化 • 感染病の国内侵入の防止 • オーダーメイド医療	待ち行列理論　CRISPR/Cas9 de Bruijn Graph HiSeq　メンデルの法則 PCR　ボイヤー・ムーア法

社会問題 に対し

規律・秩序を制定し、

問題解決の最善方策を探索する。

社会問題

人工知能

機械学習

人工知能

- 1947 A. M. Tuning が人工知能の概念を提唱。
- 1950 チューリングテストを利用して機械が知性を持つかどうかを検証。
- 1956 Dartmouth 会議で初めて Artificial Intelligence が使われた。

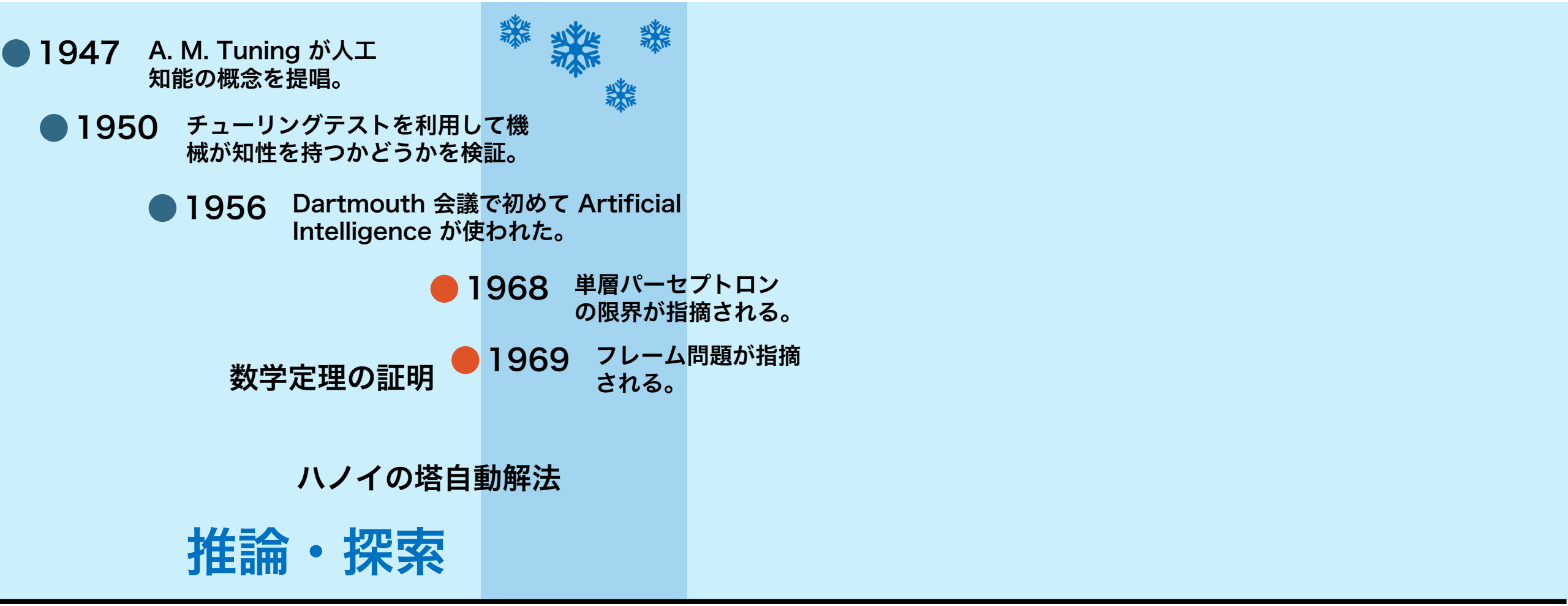
数学定理の証明

ハノイの塔自動解法

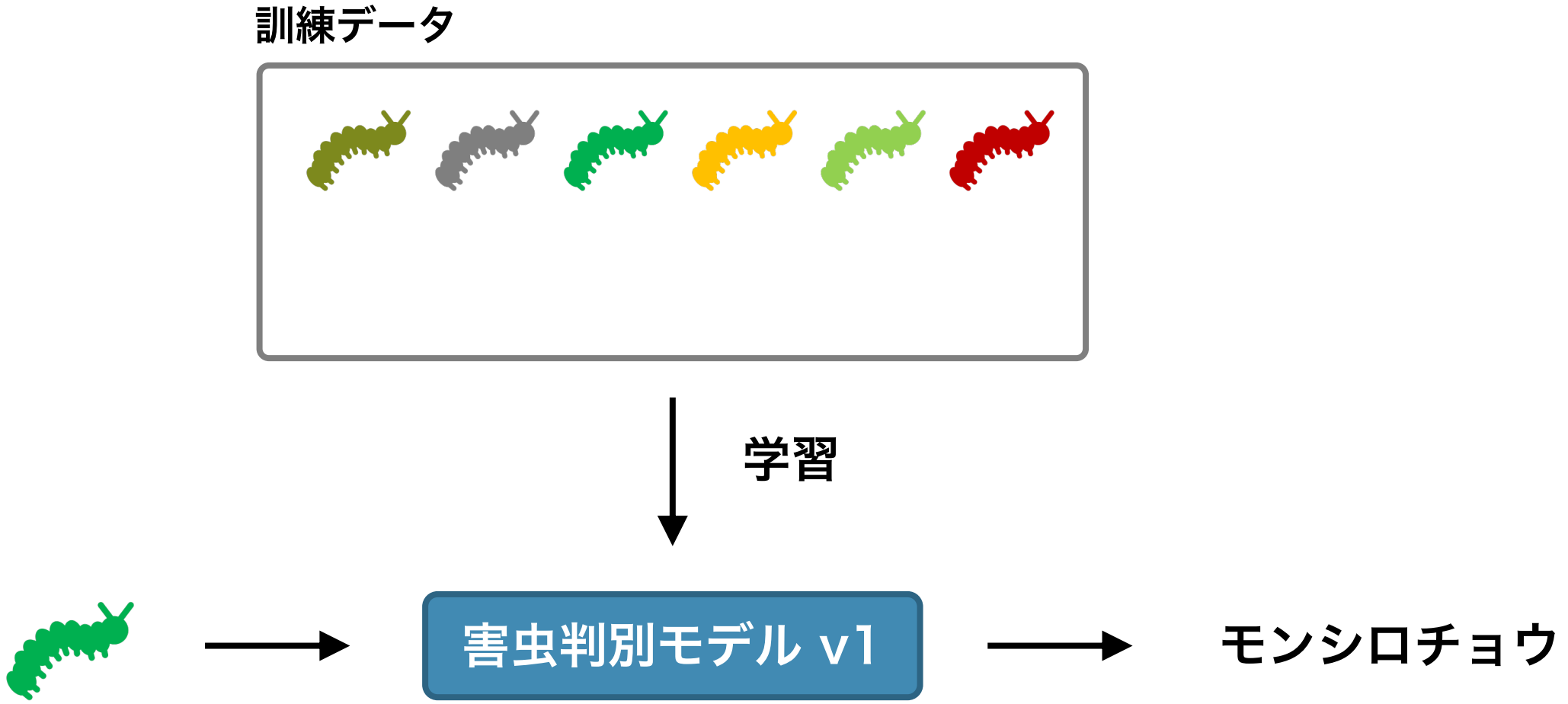
推論・探索

1950 1960 1970 1980 1990 2000 2010

人工知能

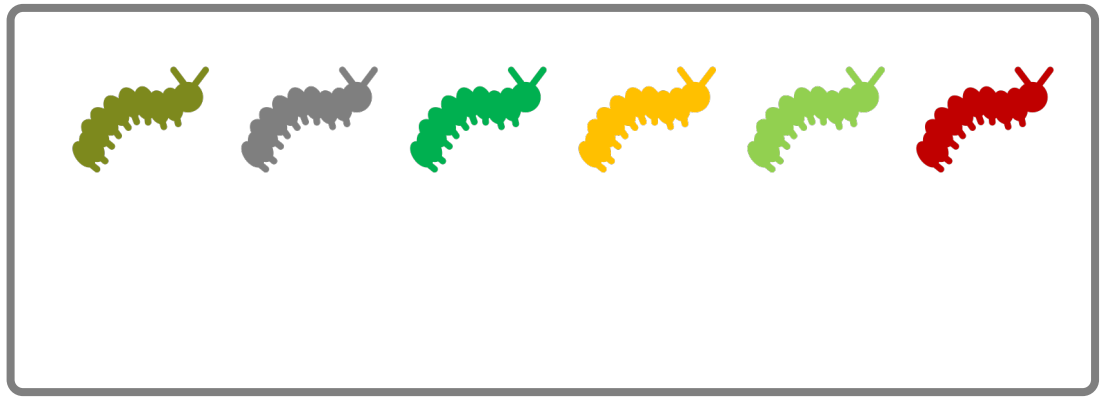

1947 A. M. Tuning が人工知能の概念を提唱。
1950 チューリングテストを利用して機械が知性を持つかどうかを検証。
1956 Dartmouth 会議で初めて Artificial Intelligence が使われた。
1968 単層パーセプトロンの限界が指摘される。
1969 フレーム問題が指摘される。
数学定理の証明
ハノイの塔自動解法
推論・探索
1950
1960
1970
1980
1990
2000
2010

フレーム問題



フレーム問題

訓練データ



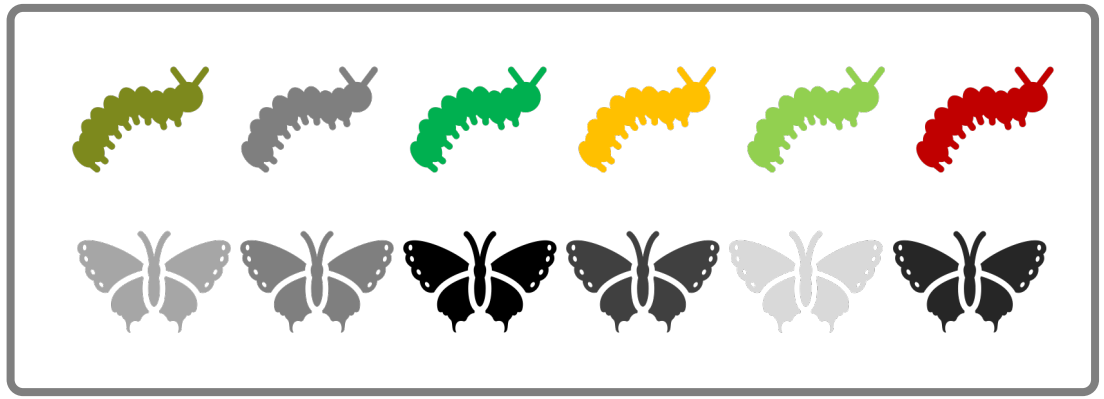
学習

害虫判別モデル v2

セスジスズメガ

フレーム問題

訓練データ



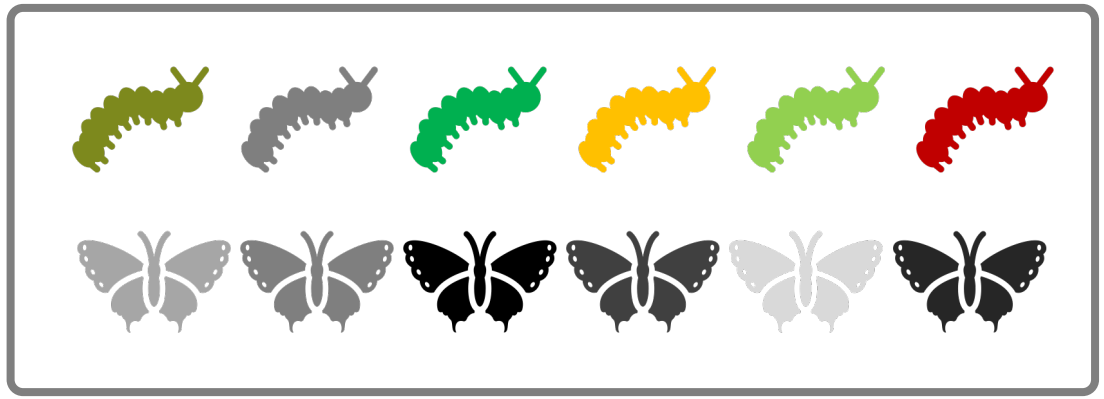
学習

害虫判別モデル v2

モンシロチョウ

フレーム問題

訓練データ



学習

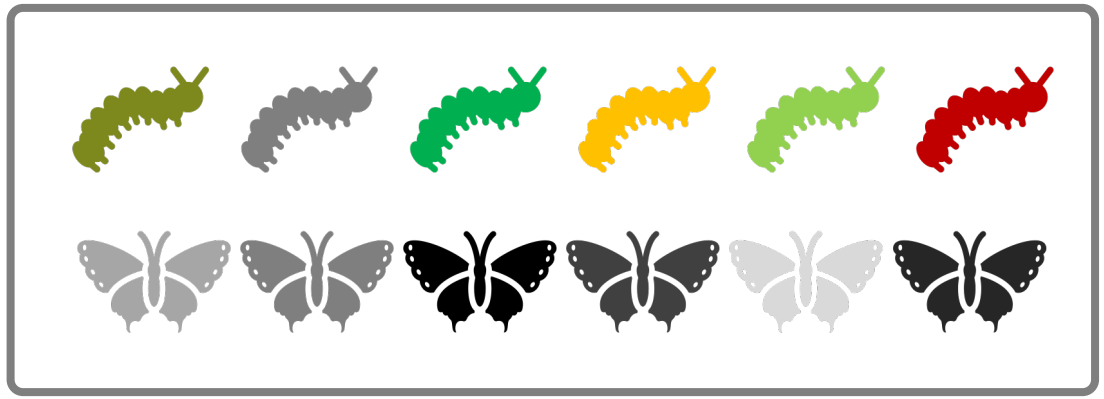
新規外来種

害虫判別モデル v2

オオゴマダラ

フレーム問題

訓練データ



+ 外来種データ
+ 蛹データ
+ 卵データ

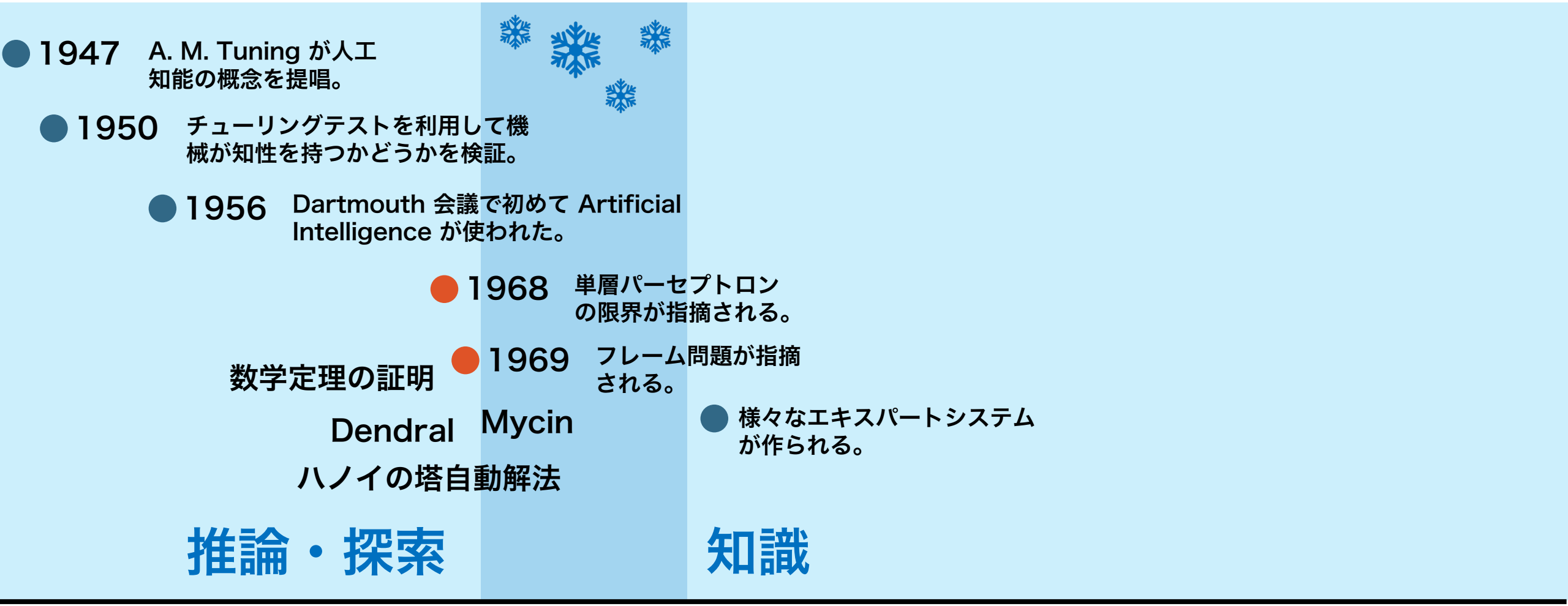
学習

新規外来種

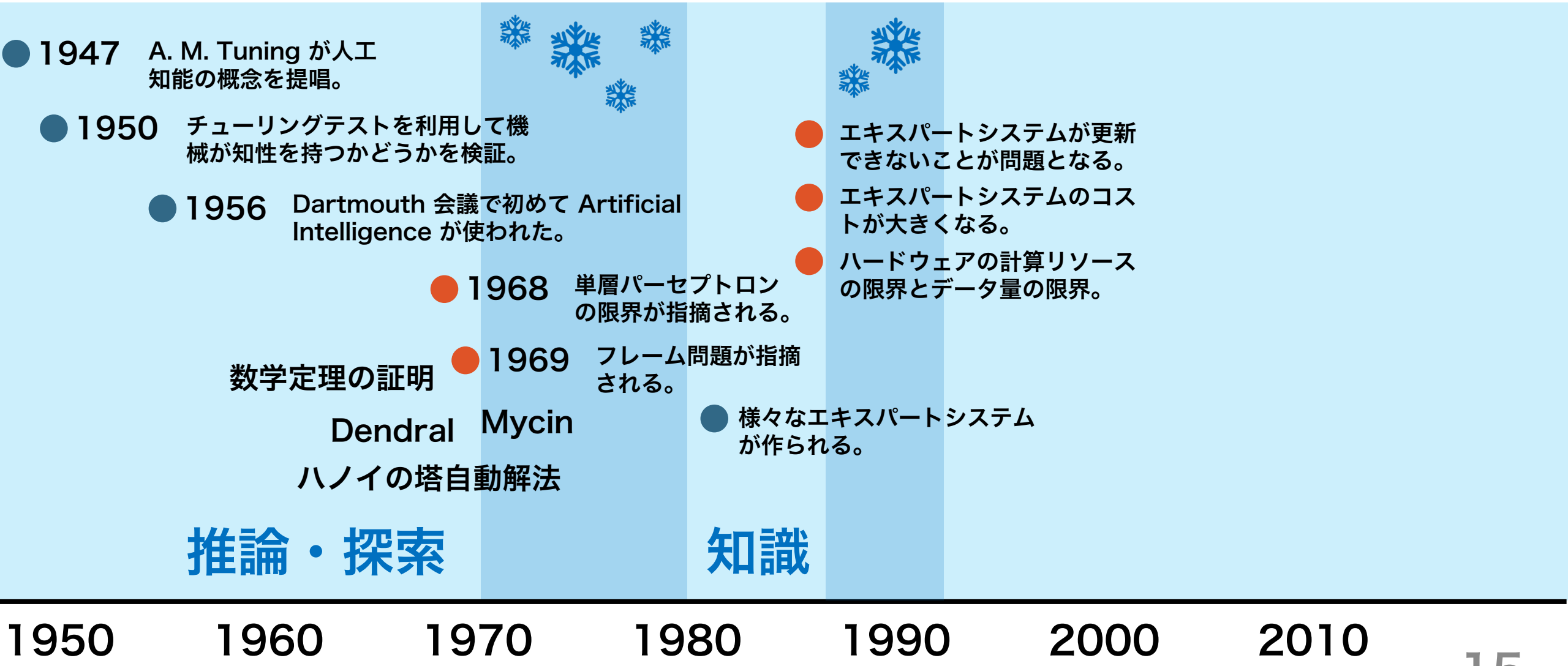
害虫判別モデル v2

オオゴマダラ

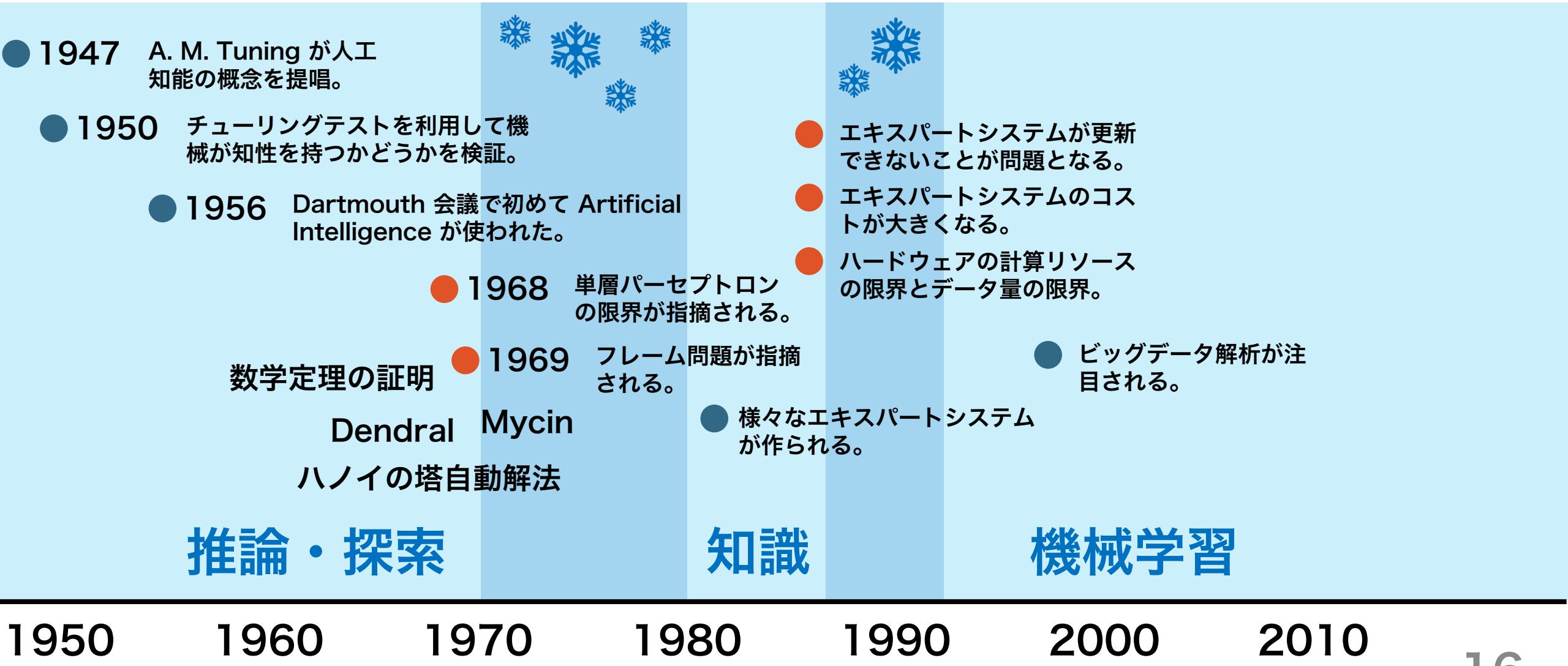
人工知能


1947 A. M. Tuning が人工知能の概念を提唱。
1950 チューリングテストを利用して機械が知性を持つかどうかを検証。
1956 Dartmouth 会議で初めて Artificial Intelligence が使われた。
1968 単層パーセプトロンの限界が指摘される。
1969 フレーム問題が指摘される。
様々なエキスパートシステムが作られる。
数学定理の証明
Dendral
Mycin
ハノイの塔自動解法
推論・探索
知識
1950
1960
1970
1980
1990
2000
2010

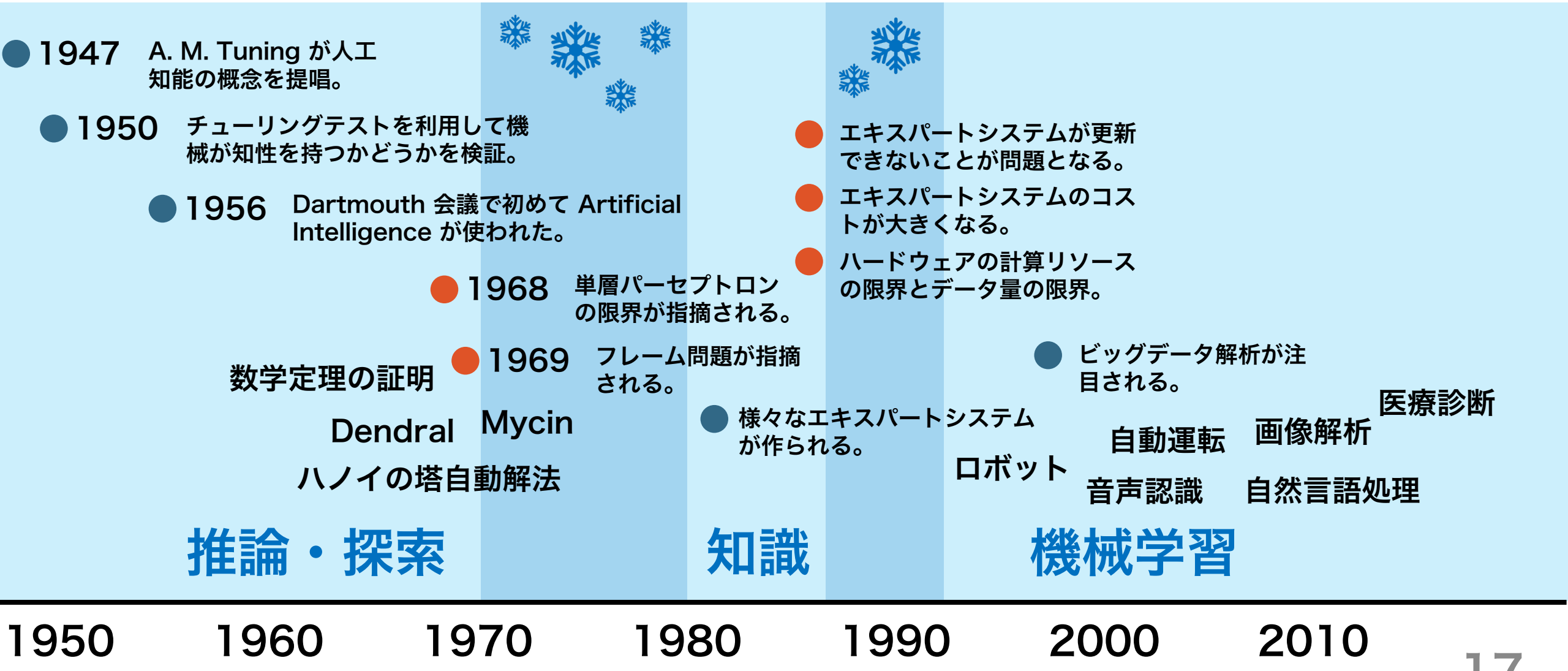
人工知能


1947 A. M. Tuning が人工知能の概念を提唱。
1950 チューリングテストを利用して機械が知性を持つかどうかを検証。
1956 Dartmouth 会議で初めて Artificial Intelligence が使われた。
1968 単層パーセプトロンの限界が指摘される。
1969 フレーム問題が指摘される。
数学定理の証明
Dendral
Mycin
ハノイの塔自動解法
様々なエキスパートシステムが作られる。
エキスパートシステムが更新できないことが問題となる。
エキスパートシステムのコストが大きくなる。
ハードウェアの計算リソースの限界とデータ量の限界。
推論・探索
知識
1950
1960
1970
1980
1990
2000
2010

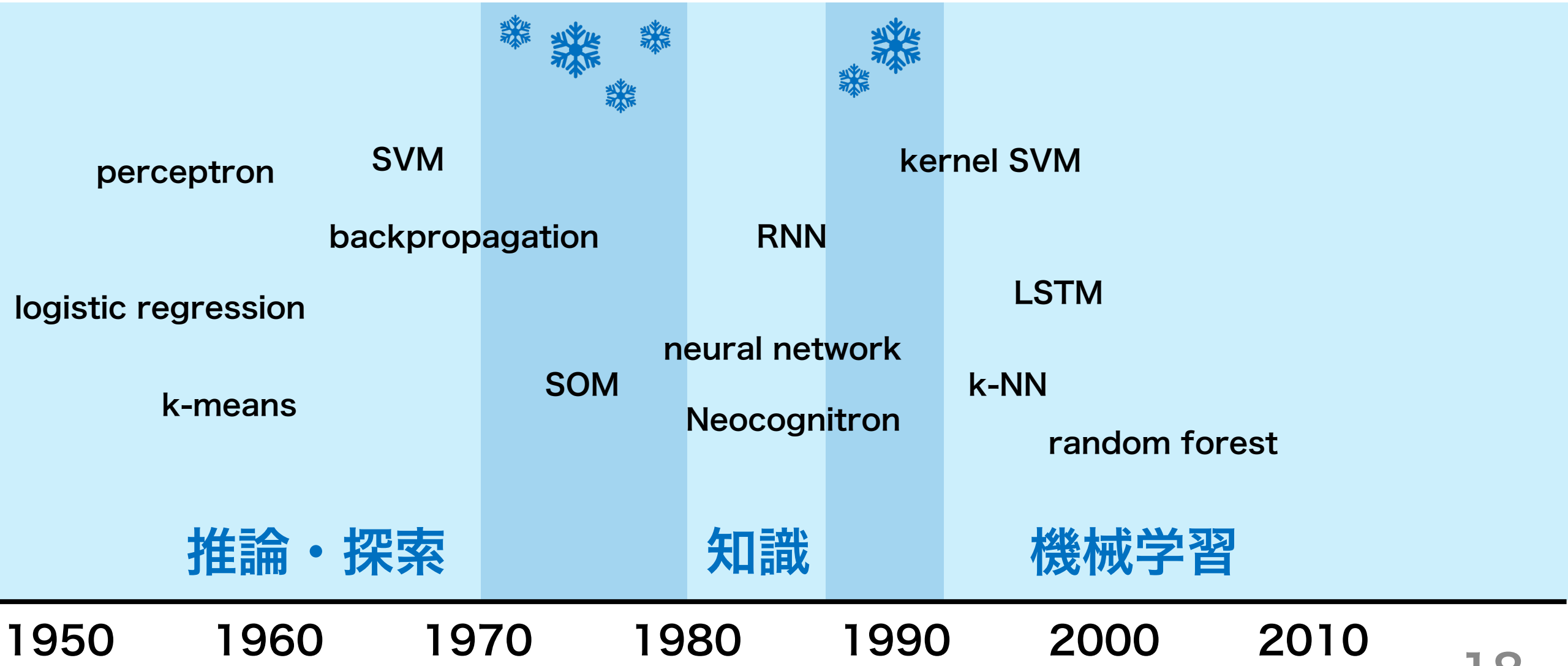
人工知能


1947 A. M. Tuning が人工知能の概念を提唱。
1950 チューリングテストを利用して機械が知性を持つかどうかを検証。
1956 Dartmouth 会議で初めて Artificial Intelligence が使われた。
1968 単層パーセプトロンの限界が指摘される。
1969 フレーム問題が指摘される。
数学定理の証明
Dendral
Mycin
ハノイの塔自動解法
様々なエキスパートシステムが作られる。
エキスパートシステムが更新できないことが問題となる。
エキスパートシステムのコストが大きくなる。
ハードウェアの計算リソースの限界とデータ量の限界。
ビッグデータ解析が注目される。
推論・探索
知識
機械学習
1950
1960
1970
1980
1990
2000
2010

人工知能


1947 A. M. Tuning が人工知能の概念を提唱。
1950 チューリングテストを利用して機械が知性を持つかどうかを検証。
1956 Dartmouth 会議で初めて Artificial Intelligence が使われた。
1968 単層パーセプトロンの限界が指摘される。
1969 フレーム問題が指摘される。
数学定理の証明
Dendral
Mycin
ハノイの塔自動解法
様々なエキスパートシステムが作られる。
エキスパートシステムが更新できないことが問題となる。
エキスパートシステムのコストが大きくなる。
ハードウェアの計算リソースの限界とデータ量の限界。
ビッグデータ解析が注目される。
ロボット
自動運転
画像解析
医療診断
音声認識
自然言語処理
推論・探索
知識
機械学習
1950
1960
1970
1980
1990
2000
2010

機械学習アルゴリズム


perceptron
SVM
kernel SVM
backpropagation
RNN
logistic regression
LSTM
neural network
SOM
k-NN
k-means
Neocognitron
random forest
推論・探索
知識
機械学習
1950
1960
1970
1980
1990
2000
2010

no free lunch theorem

コスト関数の極値を探索するあらゆるアルゴリズムは、全ての可能なコスト関数に適用した結果を平均すると同じ性能となる

Wolpert and Macready,1995

- **問題領域に関する事前知識がなければ、どの問題に対しても最適なアルゴリズムは存在しない。**
- **問題領域の知識を有効活用し、その領域に限定した最適化アルゴリズムを使用するべき。**
- **現在のデータに対して、複数の最適化アルゴリズムを適用し、最適なアルゴリズムを選択する。**

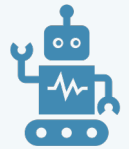
教師あり学習

回帰問題　　分類問題

半教師あり学習

クラスタリング　　次元削減

教師なし学習



強化学習

最適化問題

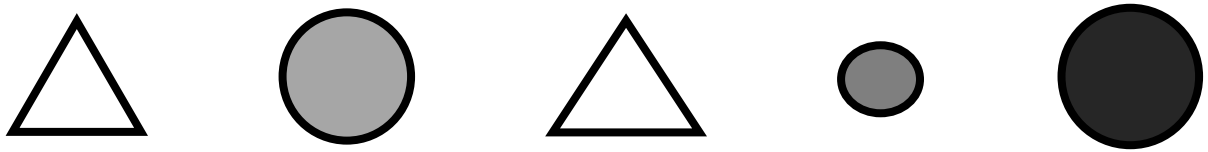
教師あり学習

教師あり学習は、訓練データに特徴量と正解ラベルがペアとして与えられるときに行う学習方法である。教師あり学習は、分類問題および回帰問題などに応用される。教師あり学習で使われる学習アルゴリズムには、次のようなものがある。

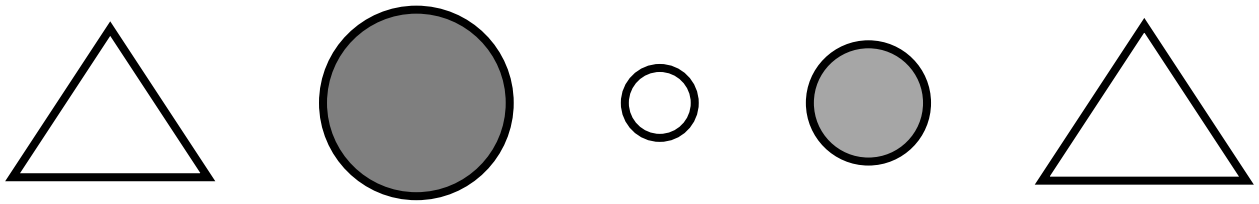
- ニューラルネットワーク
- ロジスティック回帰
- サポートベクトルマシン（SVM）
- 決定木
- ランダムフォレスト
- k 近傍法
- 線形回帰
- スパース回帰

分類問題

円形と三角形を分類したい場合、どの特徴に着目すればいいのか？



x_1	1.0	0.5	1.0	0.4	0.1
x_2	0.8	1.0	0.8	0.9	1.0



x_1	1.0	0.3	1.0	0.5	1.0
x_2	0.8	1.0	1.0	1.0	0.7

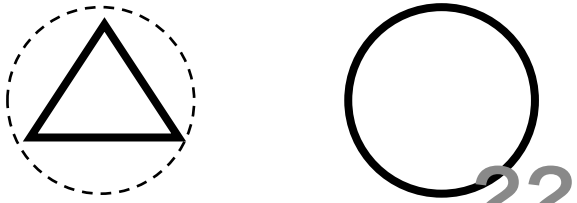
色

色に着目すると、三角形は白が多く、円形は黒い場合が多い。ここで、白を 1、黒を 0 とする色の指標を考える。

0 1

半径あたりの面積

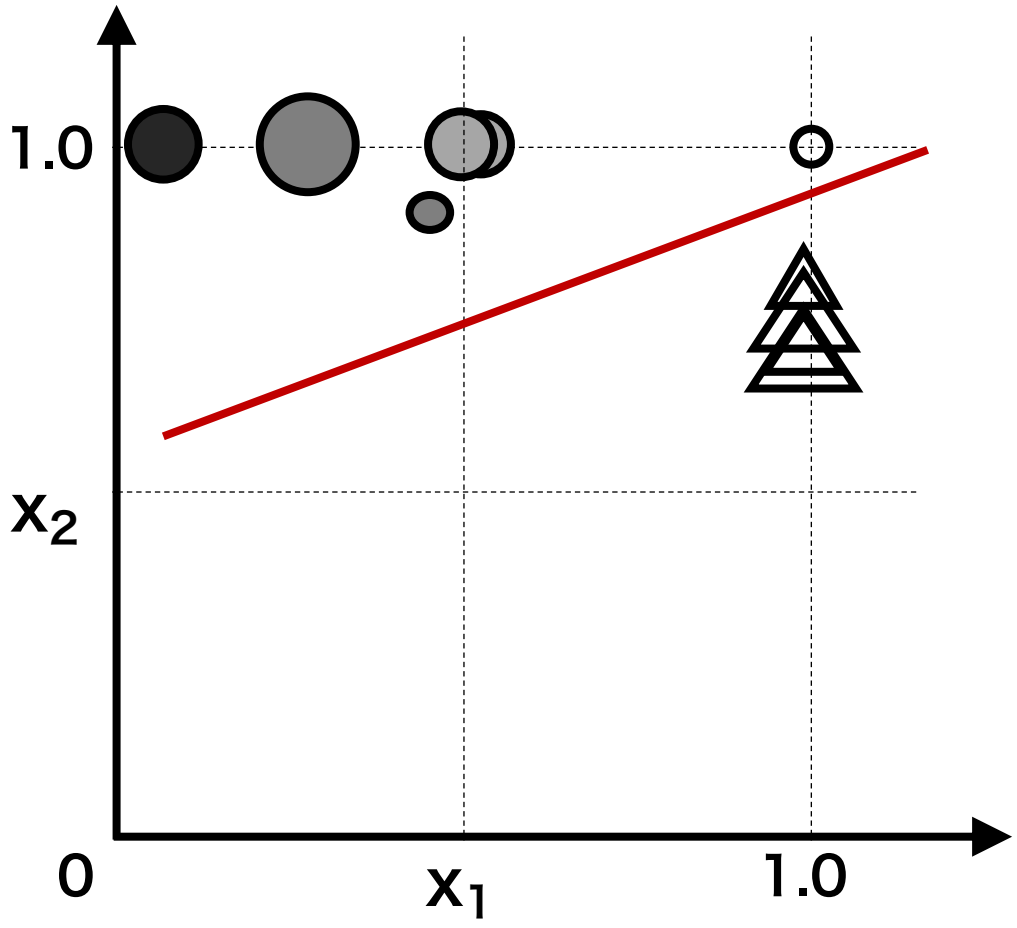
分類したい形とその外接円の面積の比に着目すると、三角形の場合は 1 よりも小さく、円形の場合はほぼ 1 になる。



分類問題

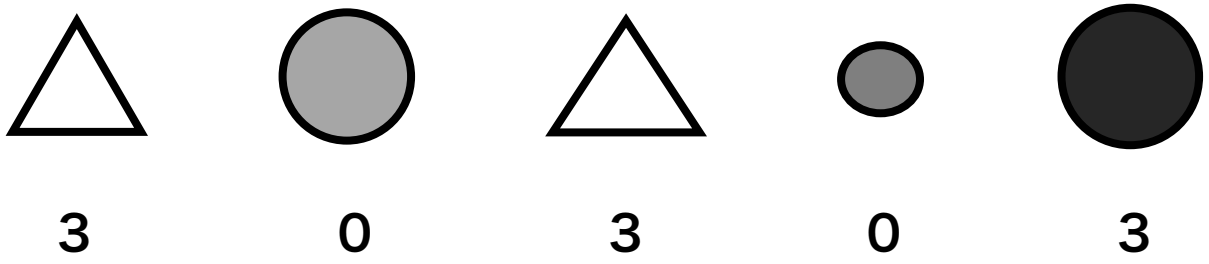
円形と三角形を分類したい場合、どの特徴に着目すればいいのか？

ラベル	X_1	X_2
1	1.0	0.8
0	0.7	1.0
1	1.0	0.8
0	0.8	0.9
0	0.9	1.0
1	1.0	0.8
0	0.8	1.0
0	1.0	1.0
0	0.7	1.0
1	1.0	0.7

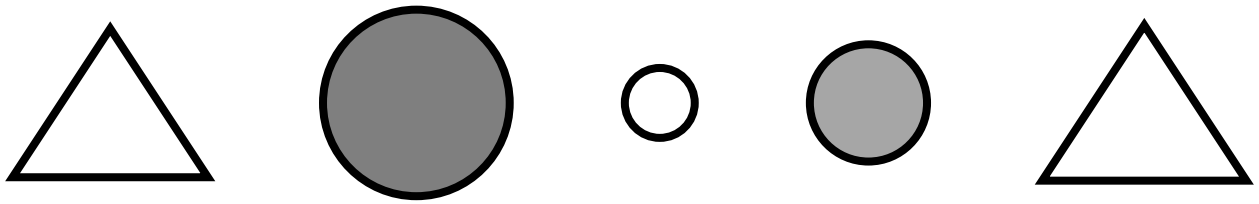


分類問題

円形と三角形を分類したい場合、どの特徴に着目すればいいのか？



x_1	3	0	3	0	3



x_1	3	0	0	0	3

角の数

円形の角は 0 個で、三角形の角の数は 3 個である。そのため、角の数を特徴量をして使えば、この 1 つの特徴量だけで円形と三角形を分けられるようになる。

回帰問題

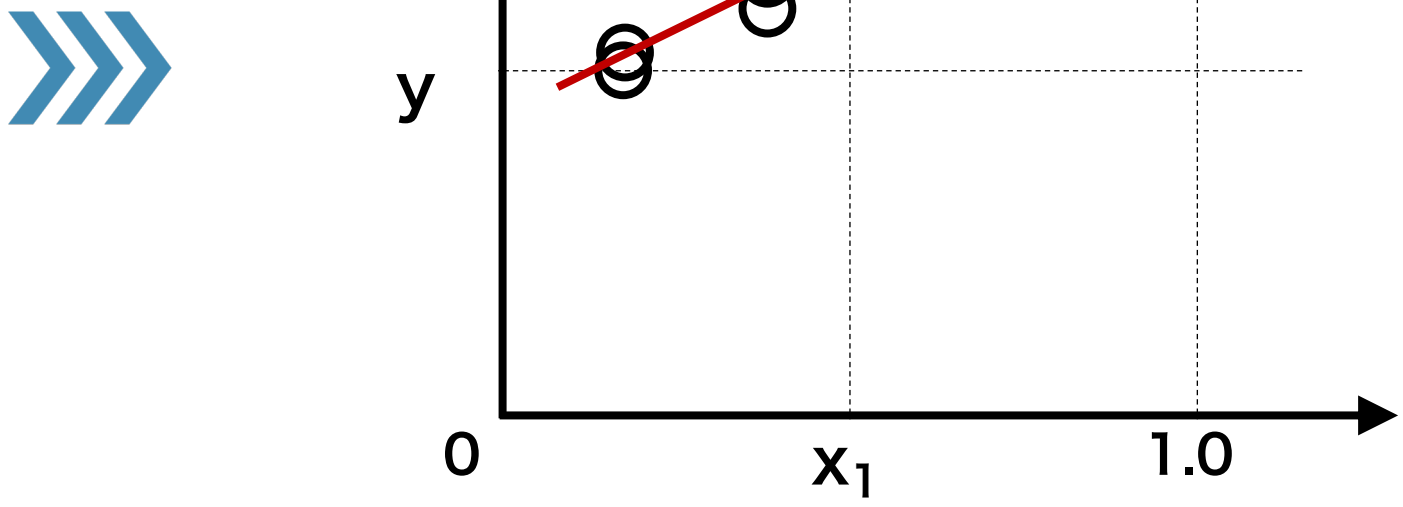
施肥量を利用して収穫量を予測するにはどうすればいいのか？

実験区画	区画 1	区画 2	区画 3	区画 4	区画 5
施肥量	0.2	0.4	0.6	0.8	1.0
収穫量					
	10.2	12.4	16.1	17.8	18.2
	10.1	11.9	17.2	18.1	18.0

回帰問題

施肥量を利用して収穫量を予測するにはどうすればいいのか？

ラベル	X_1
10.2	0.2
10.1	0.2
12.4	0.4
11.9	0.4
16.1	0.6
17.2	0.6
17.8	0.8
18.1	0.8
18.2	1.0
18.0	1.0



教師なし学習

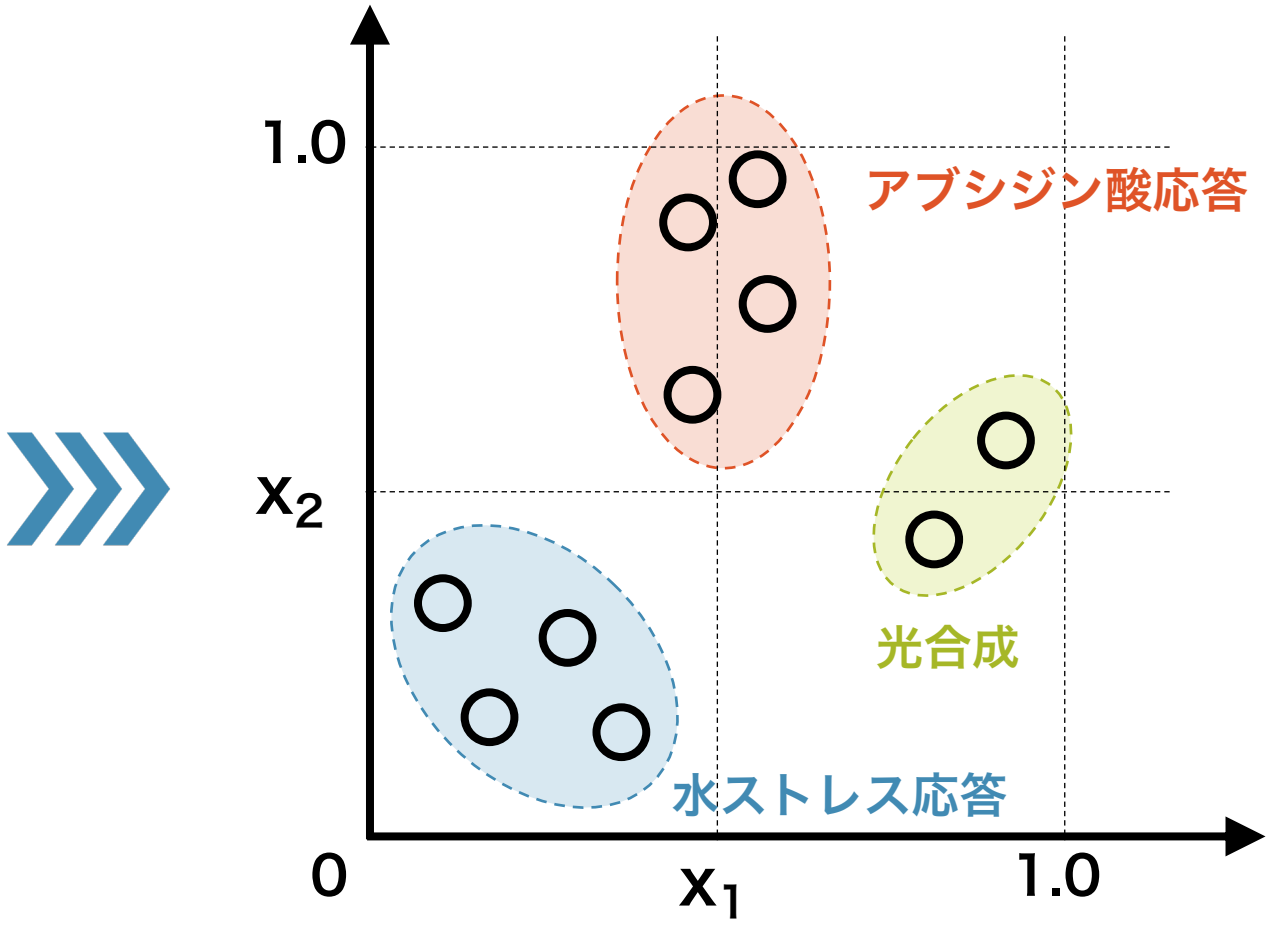
教師なし学習は、訓練データとして特徴量が与えられるときに行う学習方法である。機械が大量なデータを解釈し、データに隠されたパターンを抽出できるようになる。教師なし学習は、クラスタリングや次元削減・特徴抽出、外れ値検出などに適用される。教師あり学習で使われる学習アルゴリズムには、次のようなものがある。

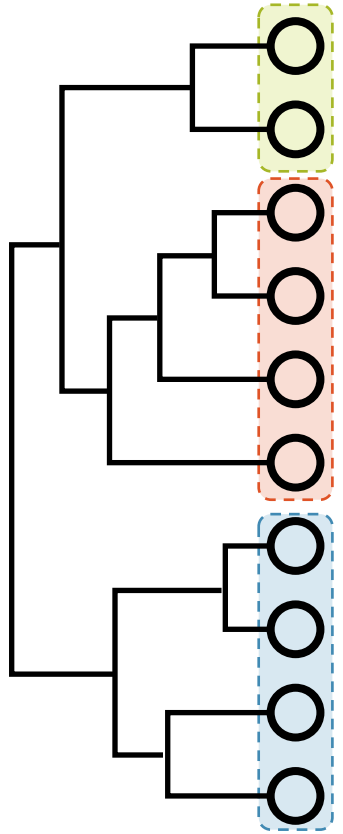
- 階層型クラスタリング
- k-means
- 自己組織化モデル（SOM）
- トピックモデル（pLSI, LDA）
- 主成分分析（PCA）
- 線形判別分析（LDA）

クラスタリング

10 個の遺伝子に対して、乾燥ストレス処理前 X_1 と処理後 X_2 の発現量を測定した。遺伝子間の関係を調べるにはどうすればいいのか？

X_1	X_2
0.12	0.38
0.92	0.57
0.39	0.21
0.52	0.98
0.49	0.88
0.26	0.25
0.34	0.36
0.49	0.61
0.53	0.73
0.82	0.46

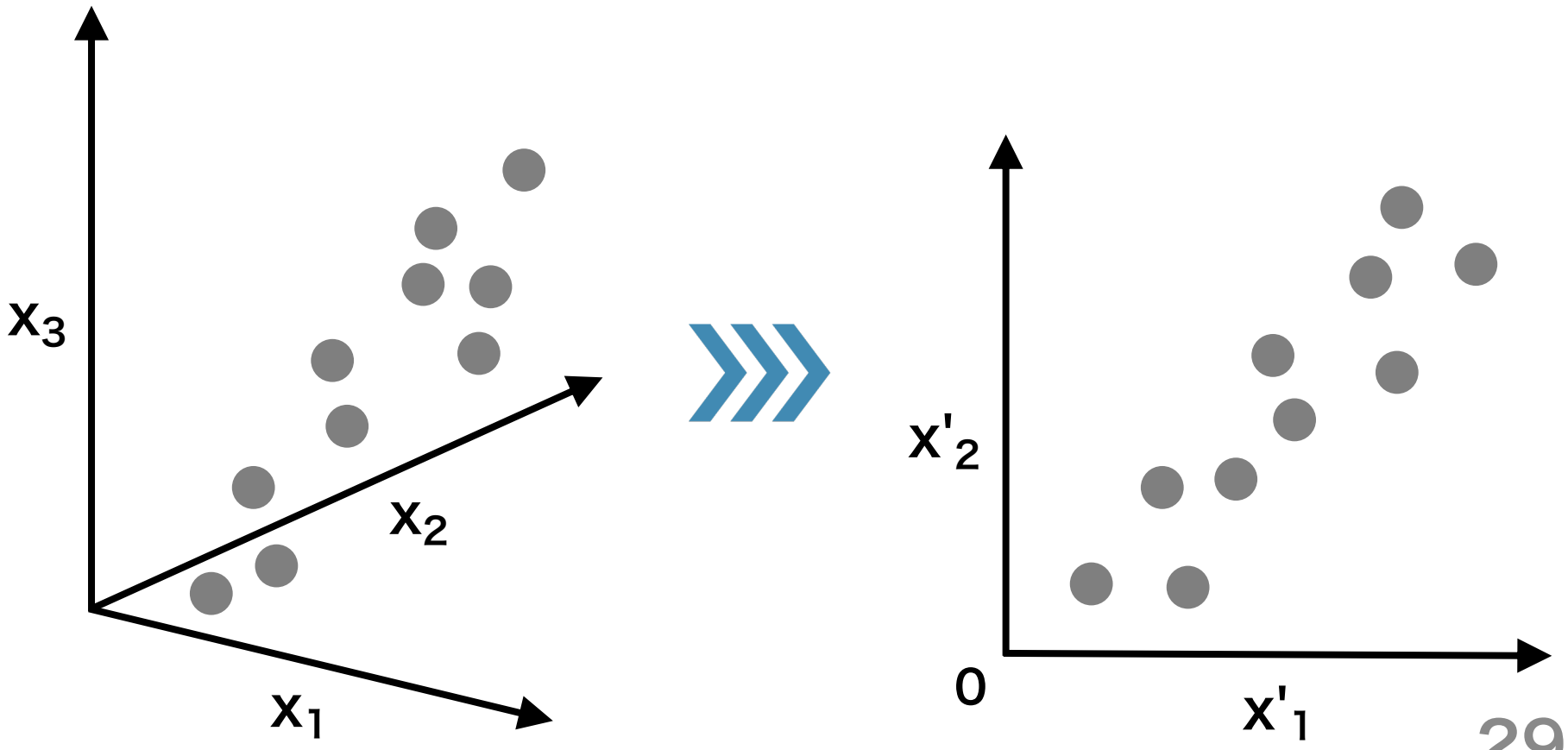




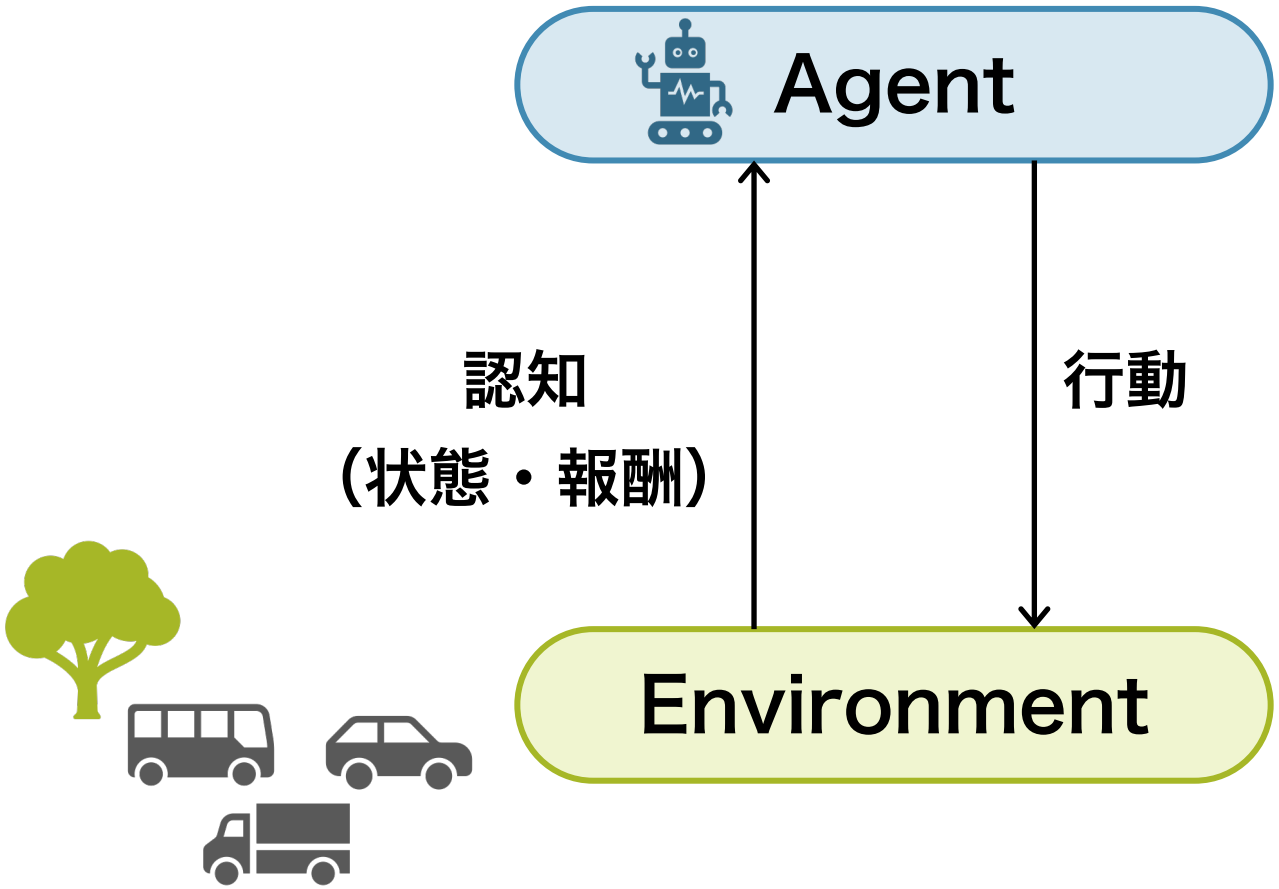
次元削減・特徴量抽出

多次元の情報を二次元の情報に落として図示したい。

x_1	x_2	x_3
0.12	0.38	0.54
0.92	0.57	0.12
0.39	0.21	0.42
0.52	0.98	0.85
0.49	0.88	0.24
0.26	0.25	0.53
0.34	0.36	0.87
0.49	0.61	0.42
0.53	0.73	0.13
0.82	0.46	0.56



強化学習



機械学習概略

- 人工知能と機械学習
- 開発ワークフロー

モデル構築

入力データを使用して、パラメーターチューニングを行い、最適なモデルを構築する。

データ収集

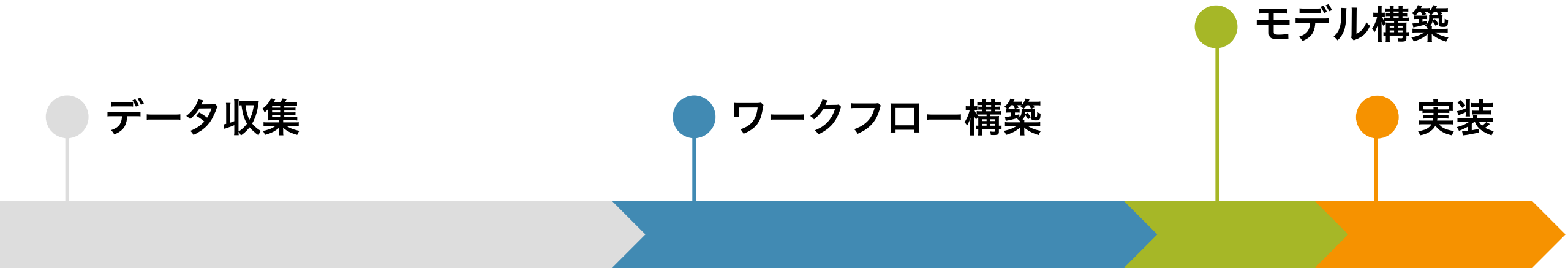
市場調査を行い、ニーズを確定し、モデル構築に必要なデータを収集する。

ワークフロー構築

収集されたデータに対して欠損値や異常値処理を行い、データを機械学習が使える形に整形し、機械学習アルゴリズムに渡すまでのワークフローを構築する。

実装

機械学習モデルをアプリ化し、スマホや専用機器に実装し、実用化する。



市場調査を行い、ニーズを確定し、モデル構築に必要なデータを収集する。

収集されたデータに対して欠損値や異常値処理を行い、データを機械学習が使える形に整形し、機械学習アルゴリズムに渡すまでのワークフローを構築する。

入力データを使用して、パラメーターチューニングを行い、最適なモデルを構築する。

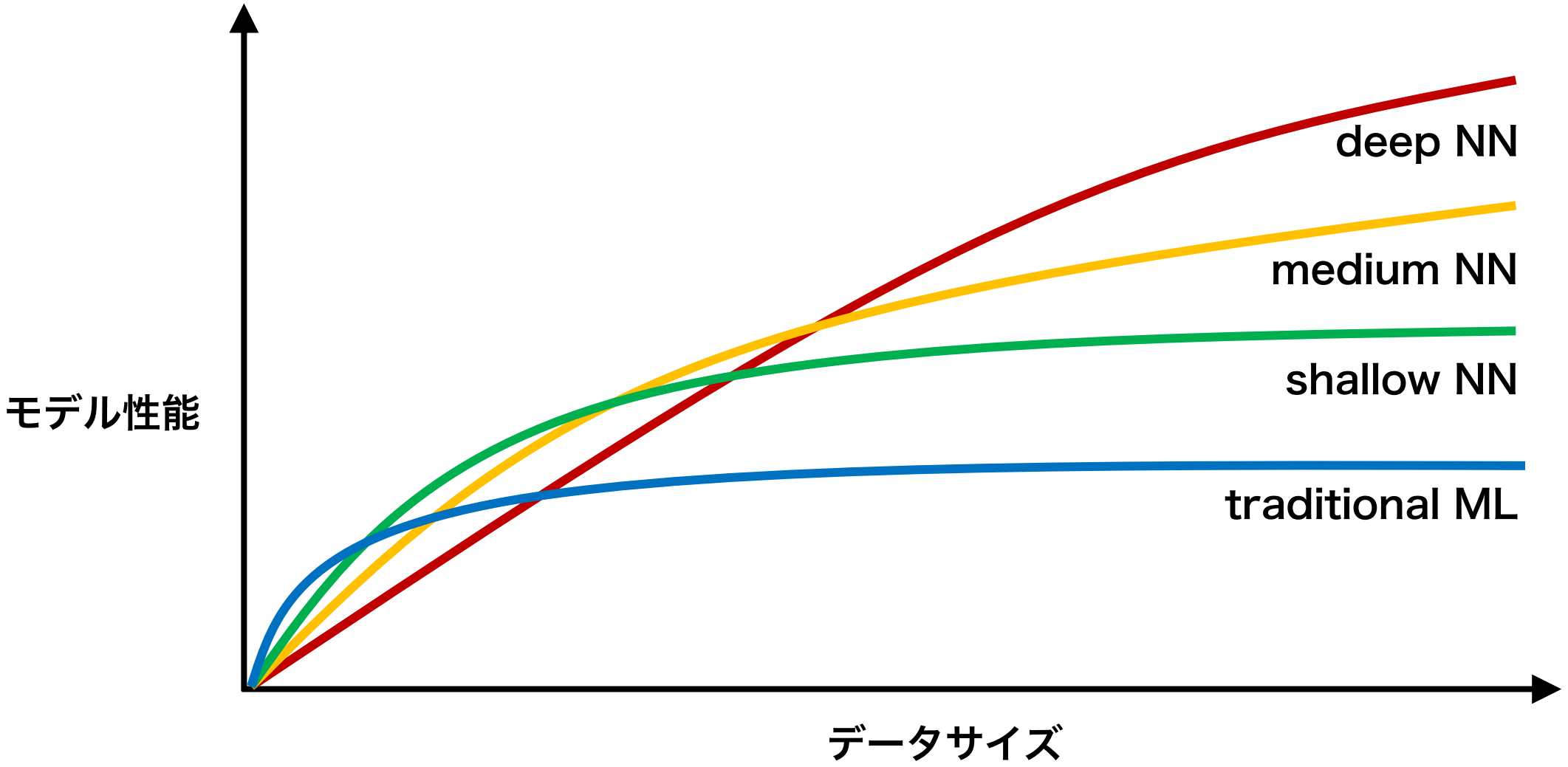
機械学習モデルをアプリ化し、スマホや専用機器に実装し、実用化する。

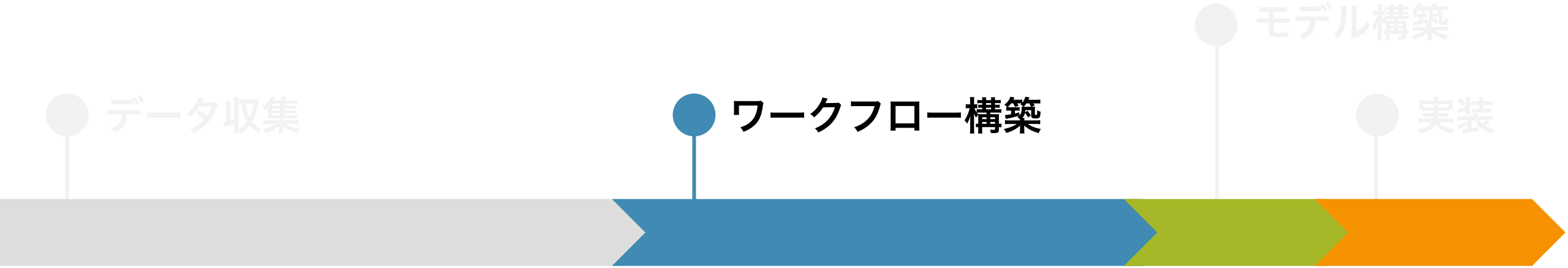
- 解決すべき問題を明確に定義する。
- 問題を解決するためのデータを収集する。
  - 機械学習の成否はデータの量と質に影響される。
  - 継続的なデータ収集フローを制定する。
- 手作業によるデータ分析を行い、効率的な機械学習アルゴリズムを検討する。

データ解析

データの重要性

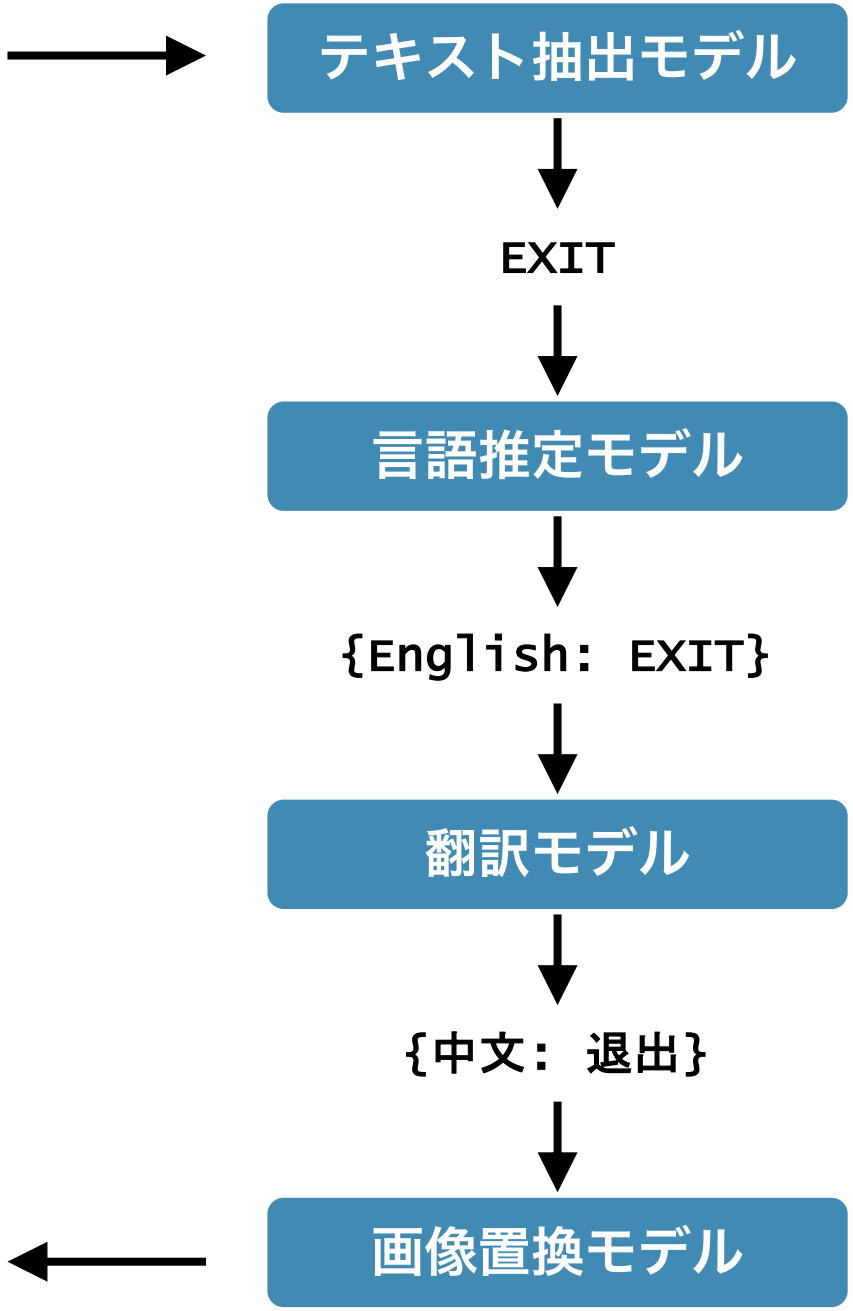
https://youtu.be/F1ka6a13S9I



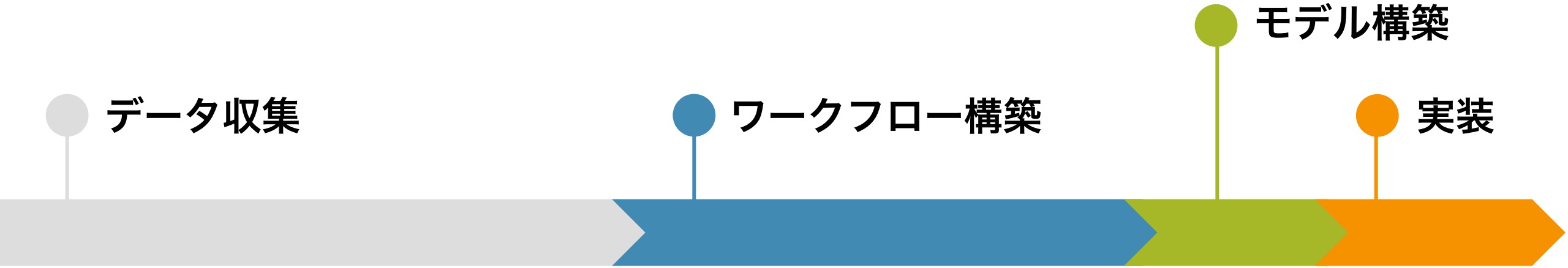


- 開発時と運用時のシステム運用方法を検討する。
  - 運用時に収集されたデータを学習に再利用できるようになっているか。
  - 運用時に専門家によるチェック過程が組み込まれているか。
- 運用時のソフトウェア・ハードウェアを検討する。
  - どのようなユーザーが、どのような使い方を想定し、どのような API を必要とするのか。

- 単一のモデルだけで、問題をすべて解決できない。
  - 複数のモデルを組み合わせて１つの問題を解決することが多い。
- 既存のアルゴリズムを優先的に使う。
  - 既存のアルゴリズムは開発コストが少なく、応用実績も多い。
- 機械学習は必ず成功するとは限らない。


テキスト抽出モデル
EXIT
言語推定モデル
{English: EXIT}
翻訳モデル
{中文: 退出}
画像置換モデル

- アプリケーションへの実装は外注で行う。
  - 研究時間を確保するために、研究要素の少ない実装は外注する。
  - 開発会社の技術力が高く、使いやすくてセキュリティの堅い実装を可能にする。
- ユーザーによる評価やフィードバックを積極的に対応する。



市場調査を行い、ニーズを確定し、モデル構築に必要なデータを収集する。

収集されたデータに対して欠損値や異常値処理を行い、データを機械学習が使える形に整形し、機械学習アルゴリズムに渡すまでのワークフローを構築する。

入力データを使用して、パラメーターチューニングを行い、最適なモデルを構築する。

機械学習モデルをアプリ化し、スマホや専用機器に実装し、実用化する。

単層ニューラルネットワーク

- パーセプトロン
- ADALINE
- ニューラルネットワーク

単層ニューラルネットワーク

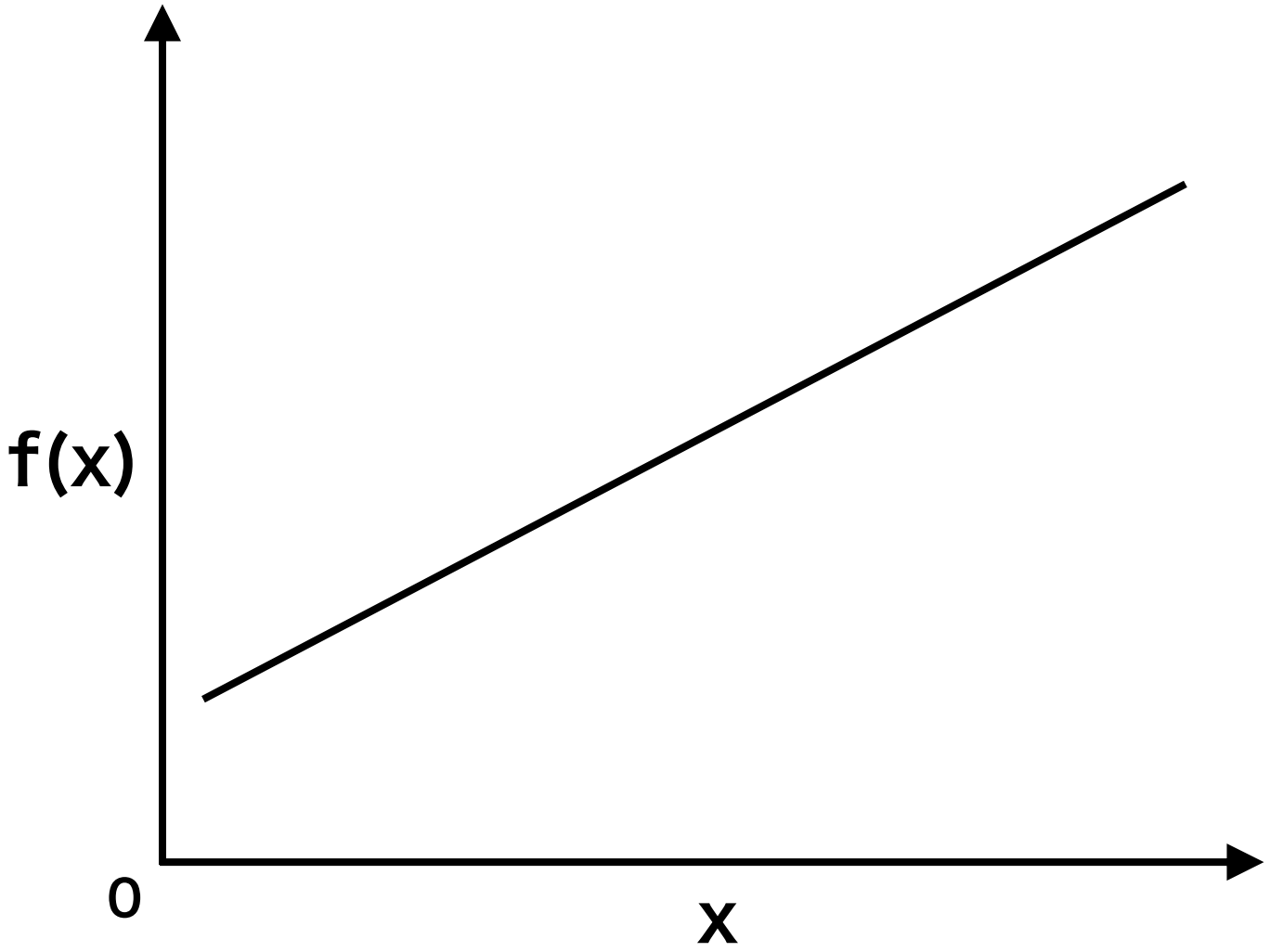
- パーセプトロン
- ADALINE
- ニューラルネットワーク

関数

１次関数 f を考える。f は入力値 x を受け取り、x に対して何らかな演算を行なって、その結果を返す関数となる。例えば、気温 x を関数 f を与えると、関数の中で気温に対して何らかの演算を行なって、穀物の収量を出力する。

$$f(x) = w_0 + w_1 x$$

▲ 収量　　▲ 気温



多変量関数

複数の値を受け取り、1 つの値を返す関数を多変量関数という。例えば、穀物の収量は、気温だけでなく、施肥量にも影響される。そこで、気温 x_1 と施肥量 x_2 を関数 f に与えると、関数の中で気温と施肥量に対して何らかの演算を行なって、穀物の収量を出力する。

f(x)

$$f(x) = w_0 + w_1 x_1 + w_2 x_2$$

▲ 収量　▲ 気温　▲ 施肥量

x_1　x_2

パーセプトロン

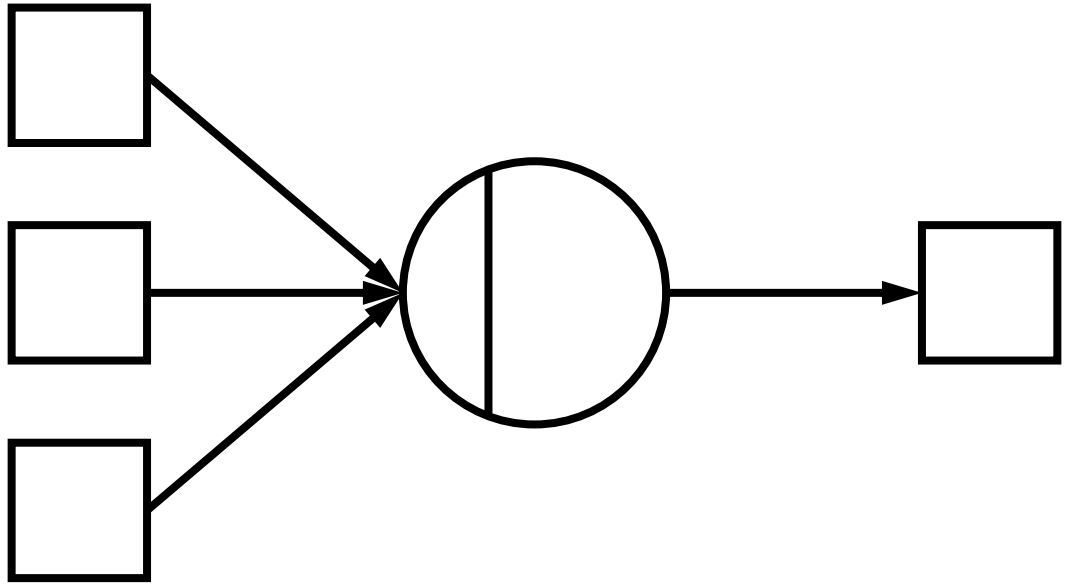
パーセプトロンは、複数の入力値に重みをかけ、その和を計算し、その和の符号に応じて値を出力するアルゴリズムである。

入力　x_1　x_2

処理　$z = w_0 + w_1 x_1 + w_2 x_2$

処理　$$\phi(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$$

出力　1



パーセプトロン

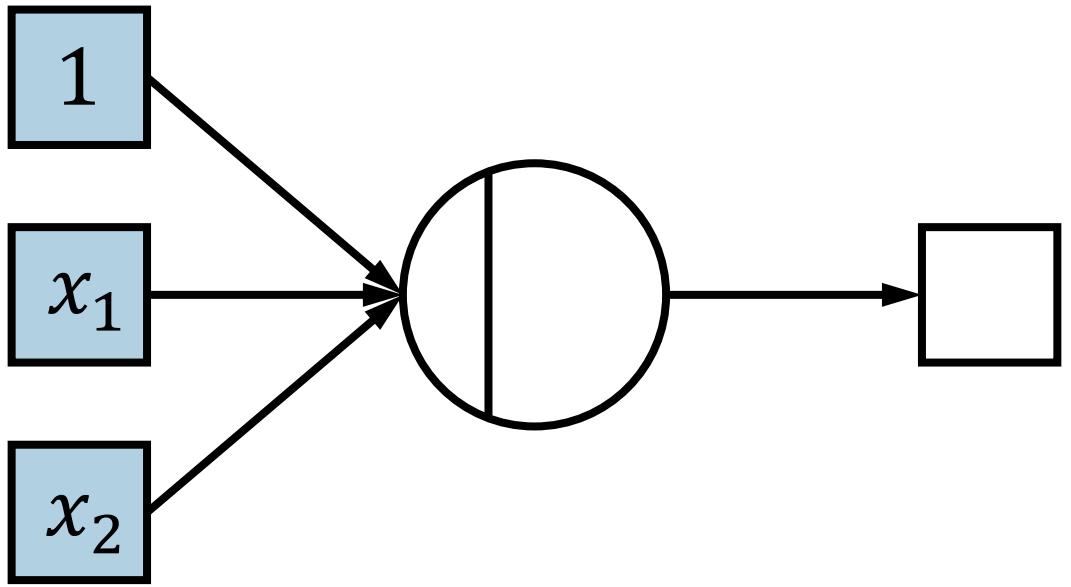
パーセプトロンは、複数の入力値に重みをかけ、その和を計算し、その和の符号に応じて値を出力するアルゴリズムである。

入力　x_1　x_2

処理　$z = w_0 + w_1 x_1 + w_2 x_2$

処理　$$\phi(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$$

出力　1



パーセプトロン

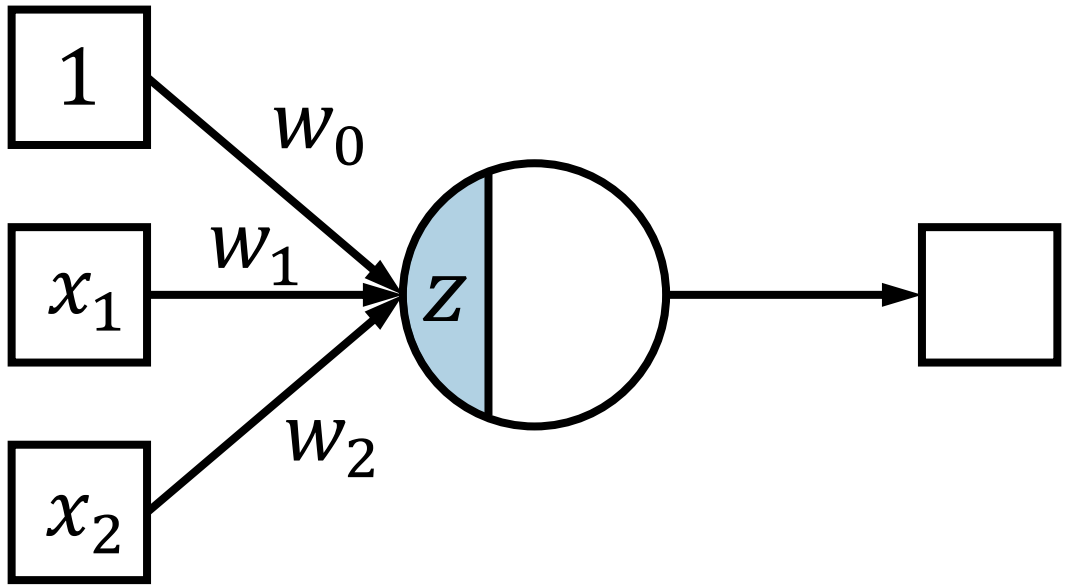
パーセプトロンは、複数の入力値に重みをかけ、その和を計算し、その和の符号に応じて値を出力するアルゴリズムである。

入力　x_1　x_2

処理　$z = w_0 + w_1 x_1 + w_2 x_2$

処理　$$\phi(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$$

出力　1



パーセプトロン

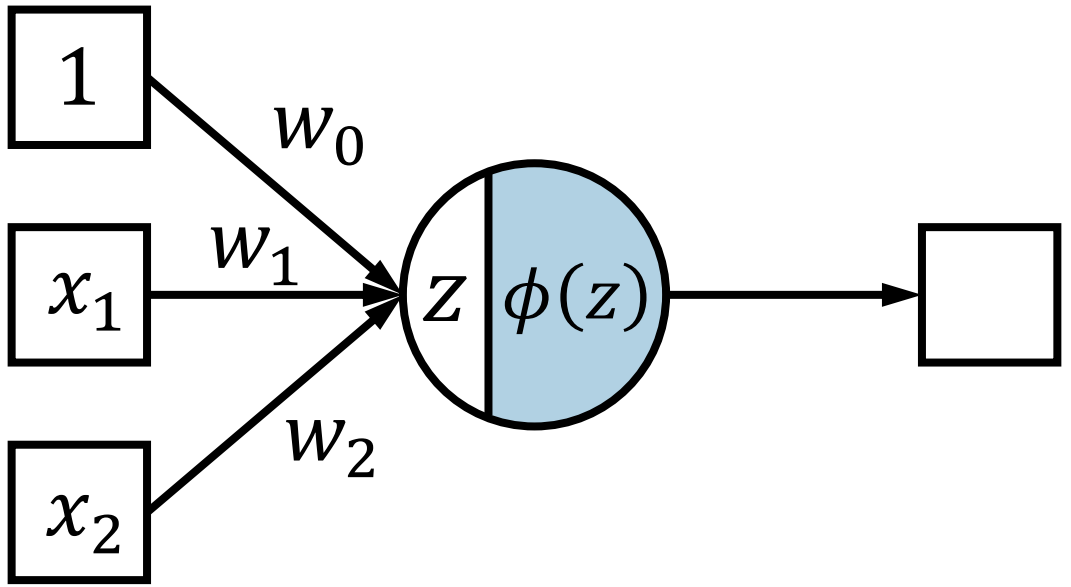
パーセプトロンは、複数の入力値に重みをかけ、その和を計算し、その和の符号に応じて値を出力するアルゴリズムである。

入力　x_1　x_2

処理　$z = w_0 + w_1 x_1 + w_2 x_2$

処理　$\phi(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$

出力　1



パーセプトロン

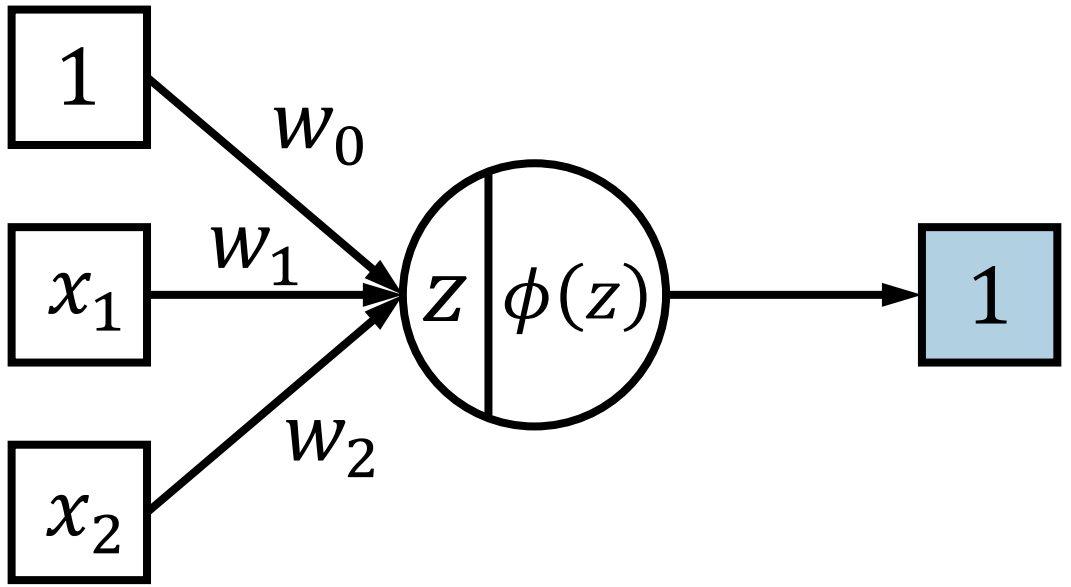
パーセプトロンは、複数の入力値に重みをかけ、その和を計算し、その和の符号に応じて値を出力するアルゴリズムである。

入力　x_1　x_2

処理　$z = w_0 + w_1 x_1 + w_2 x_2$

処理　$\phi(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$

出力　1

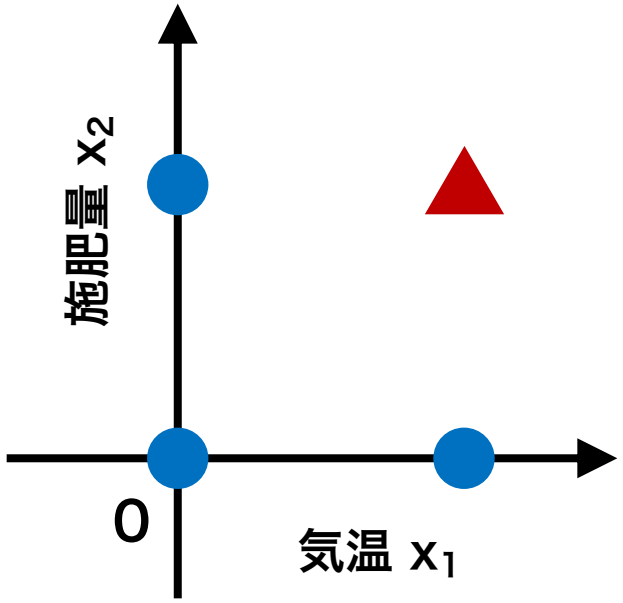


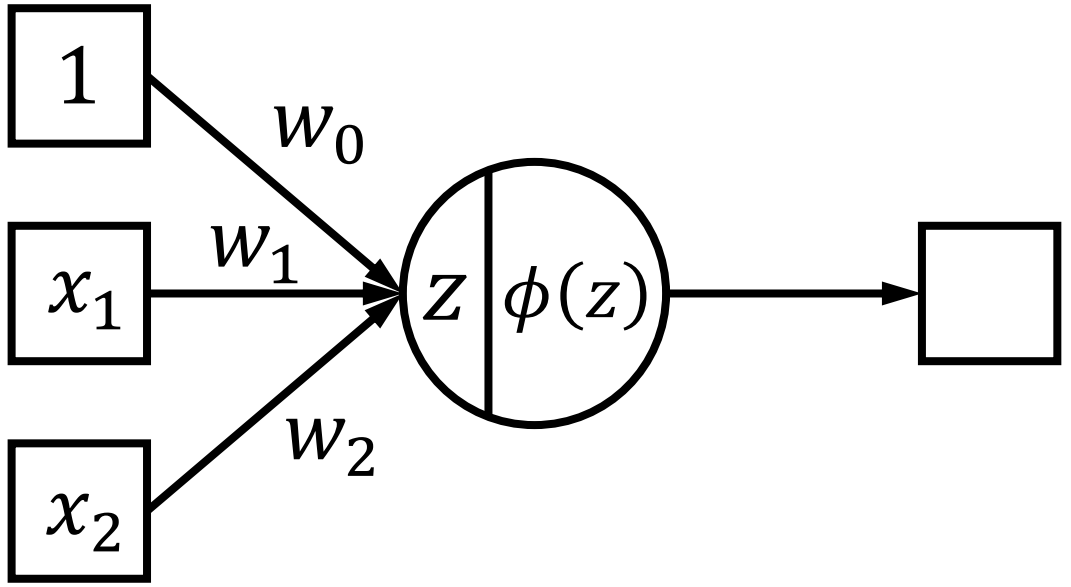
パーセプトロン学習則

学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1





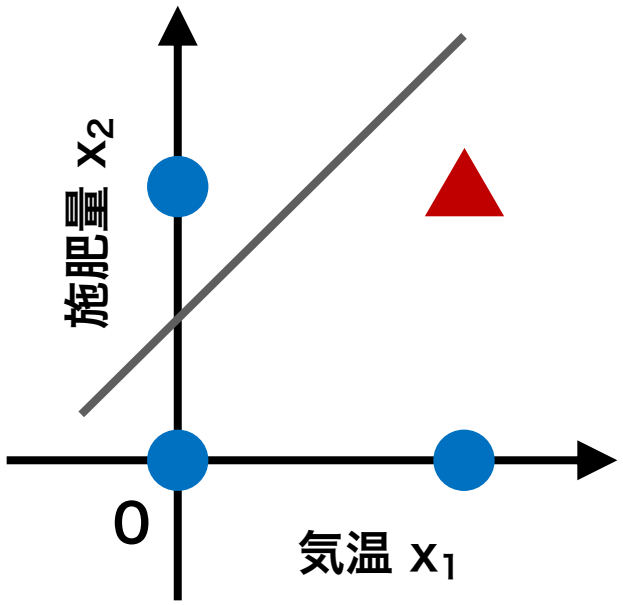
パーセプトロン学習則

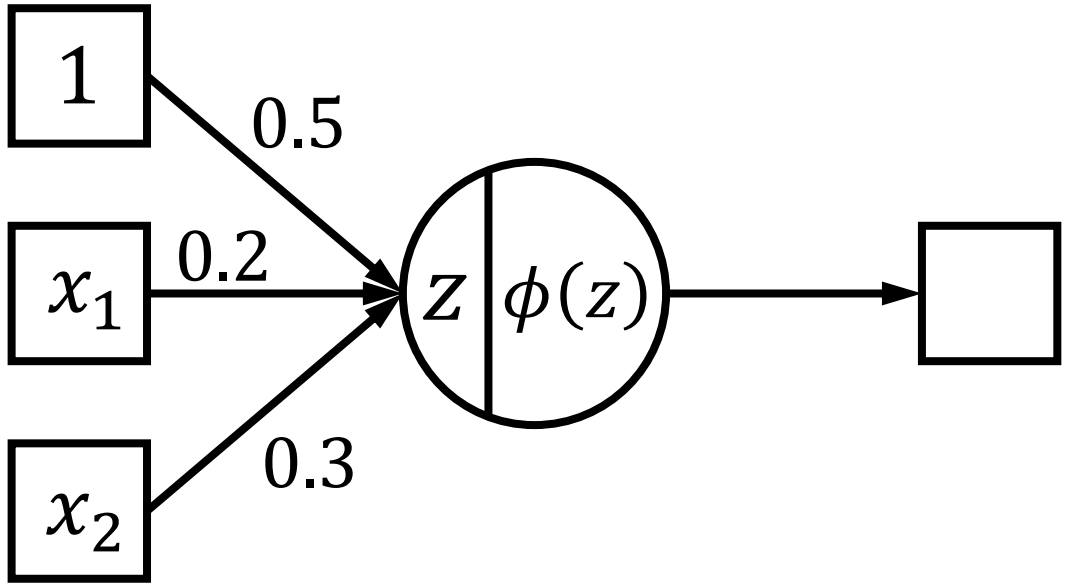
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





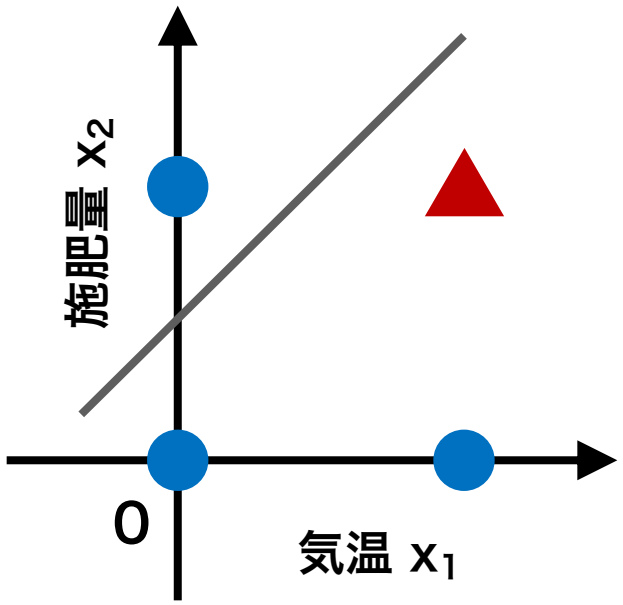
パーセプトロン学習則

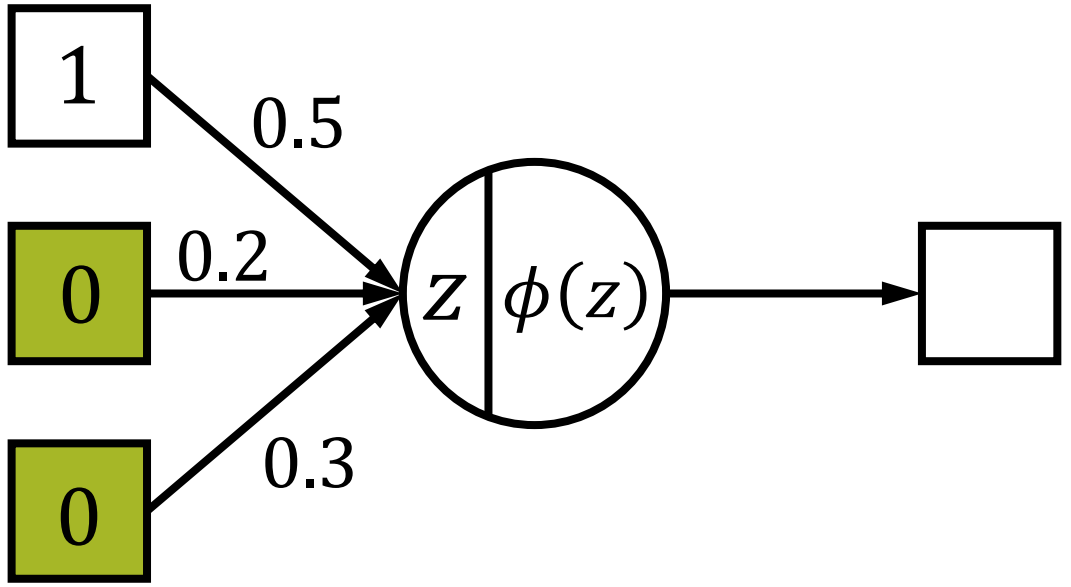
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





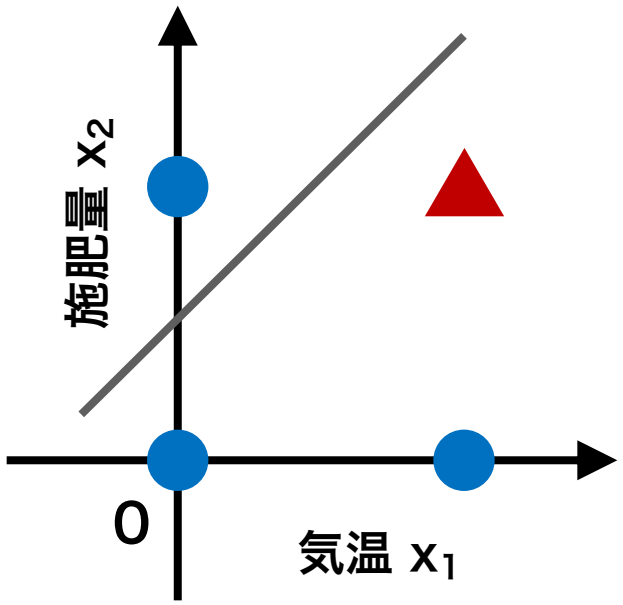
パーセプトロン学習則

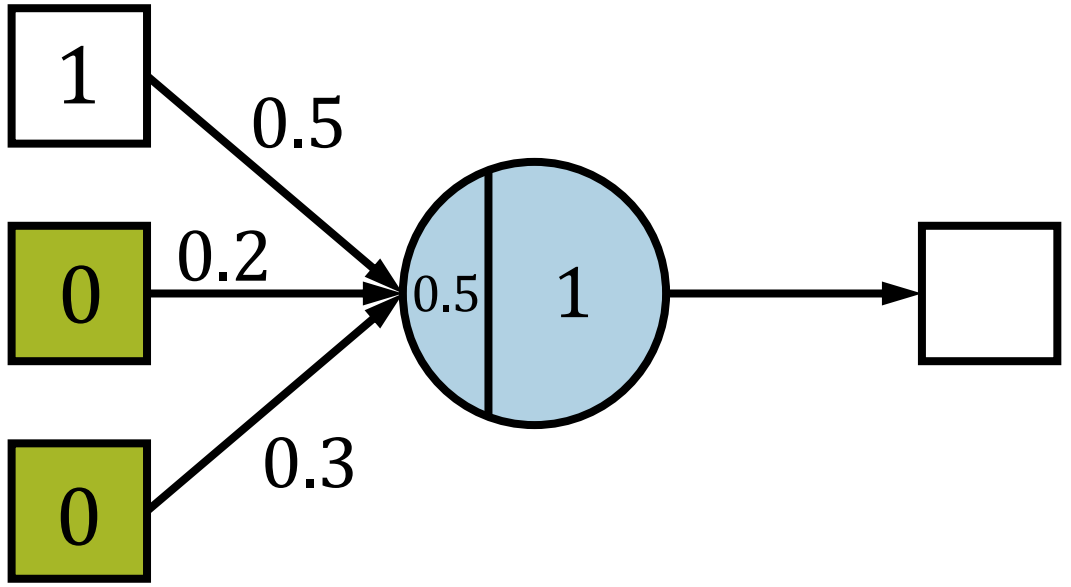
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





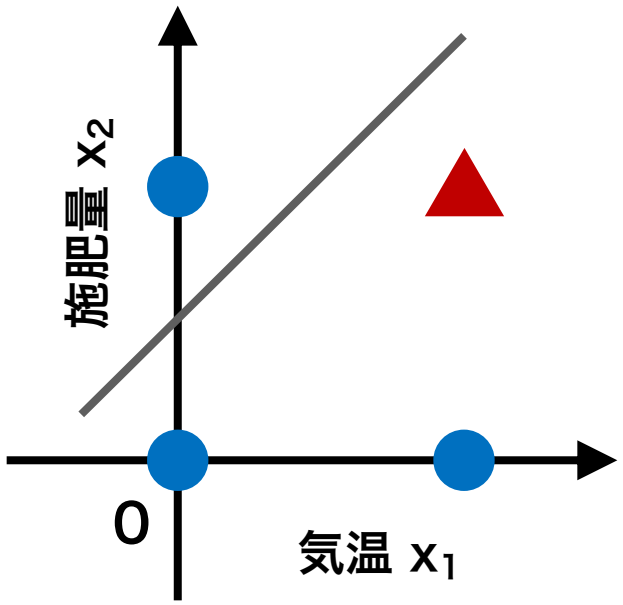
パーセプトロン学習則

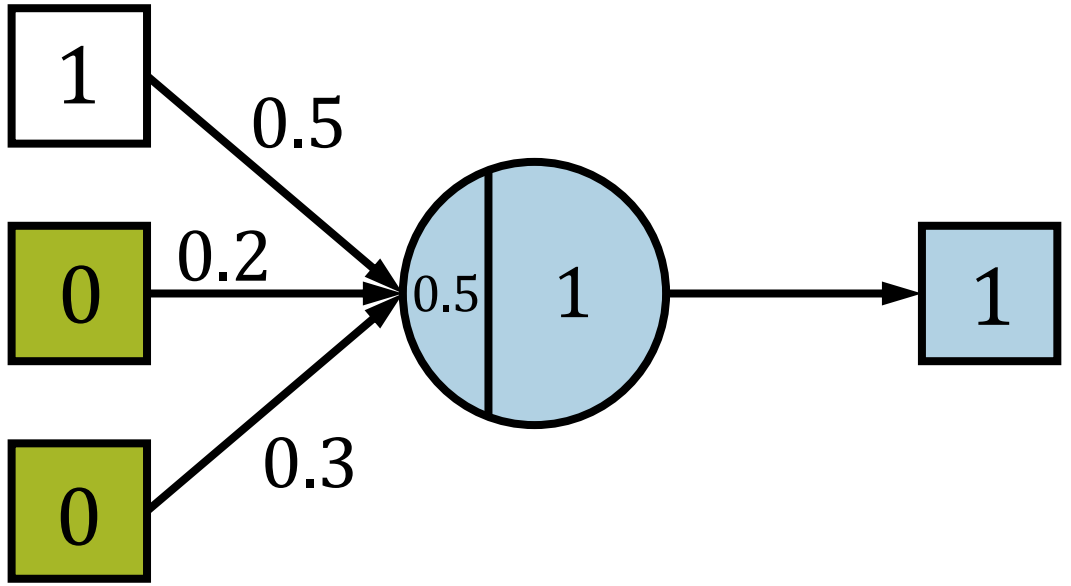
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





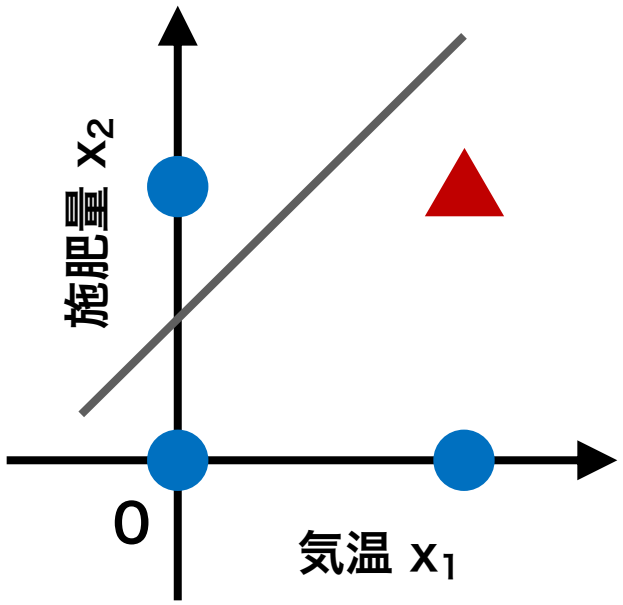
パーセプトロン学習則

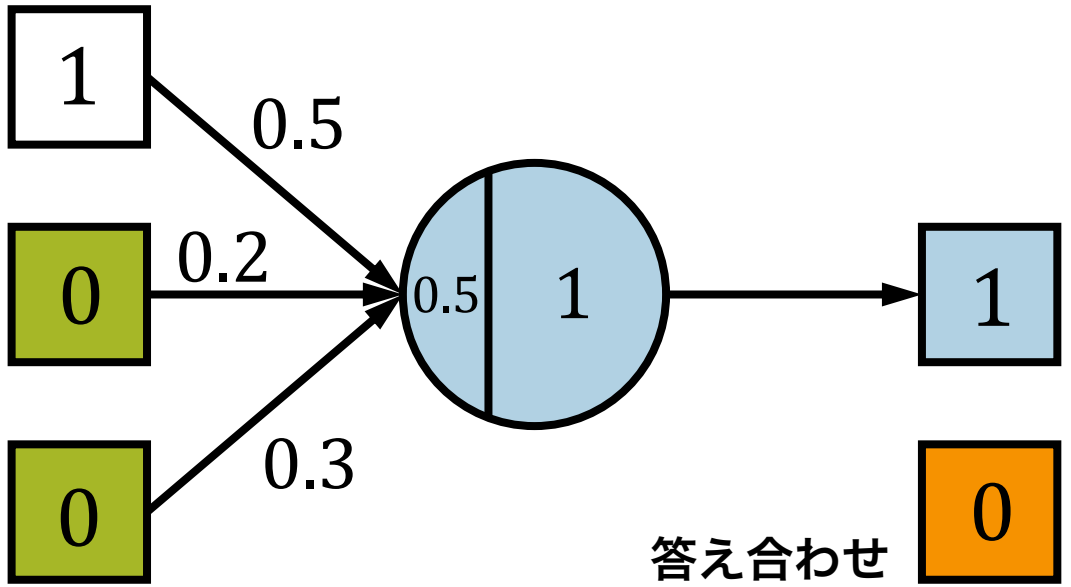
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





損失　1

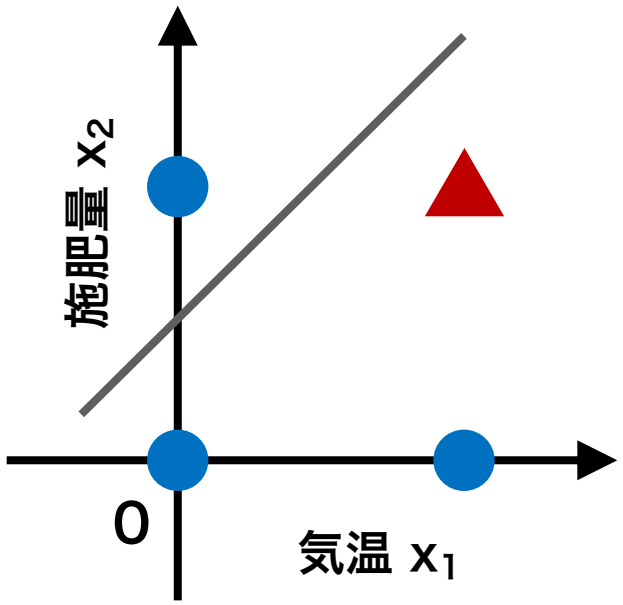
パーセプトロン学習則

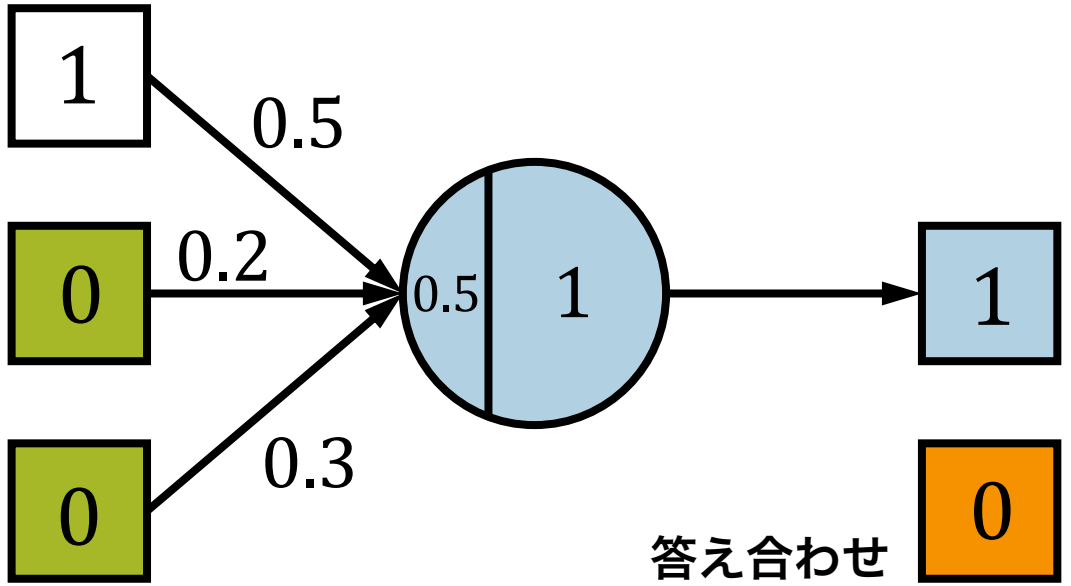
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = 0.5 + 1(0-1)0.1 = 0.4$$

$$w_1^{(new)} = 0.2 + 0(0-1)0.1 = 0.2$$

$$w_2^{(new)} = 0.3 + 0(0-1)0.1 = 0.3$$

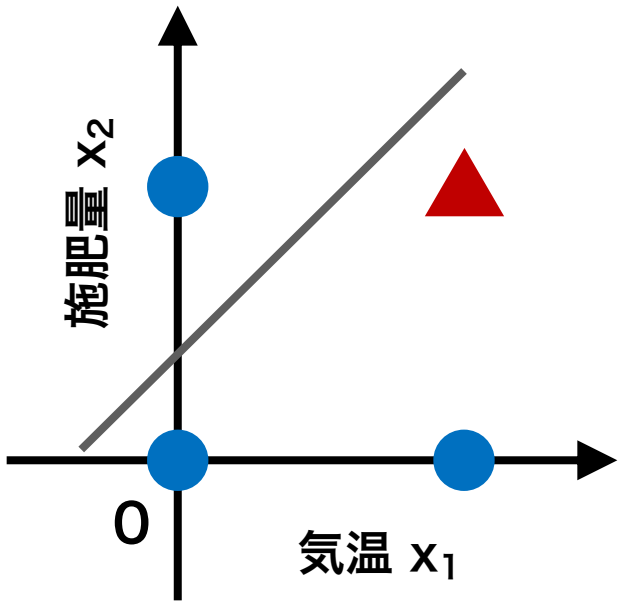
パーセプトロン学習則

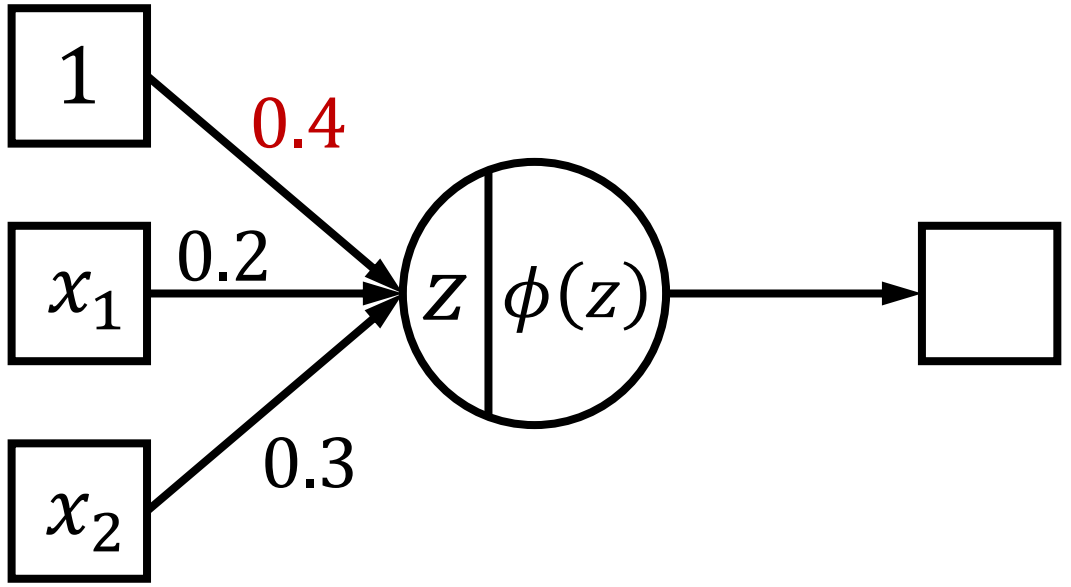
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





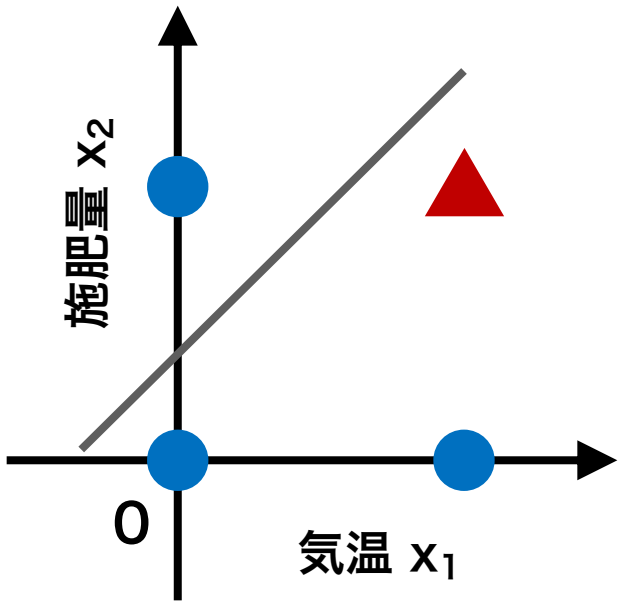
パーセプトロン学習則

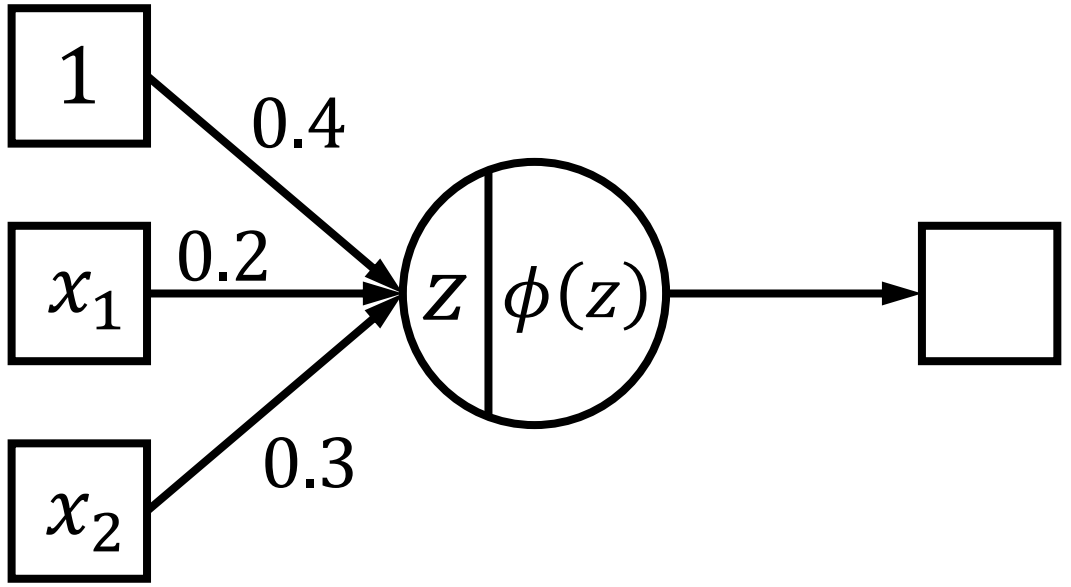
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





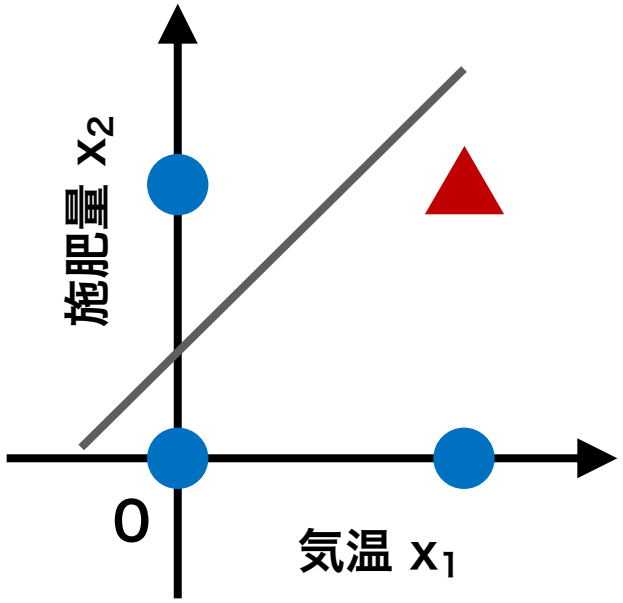
パーセプトロン学習則

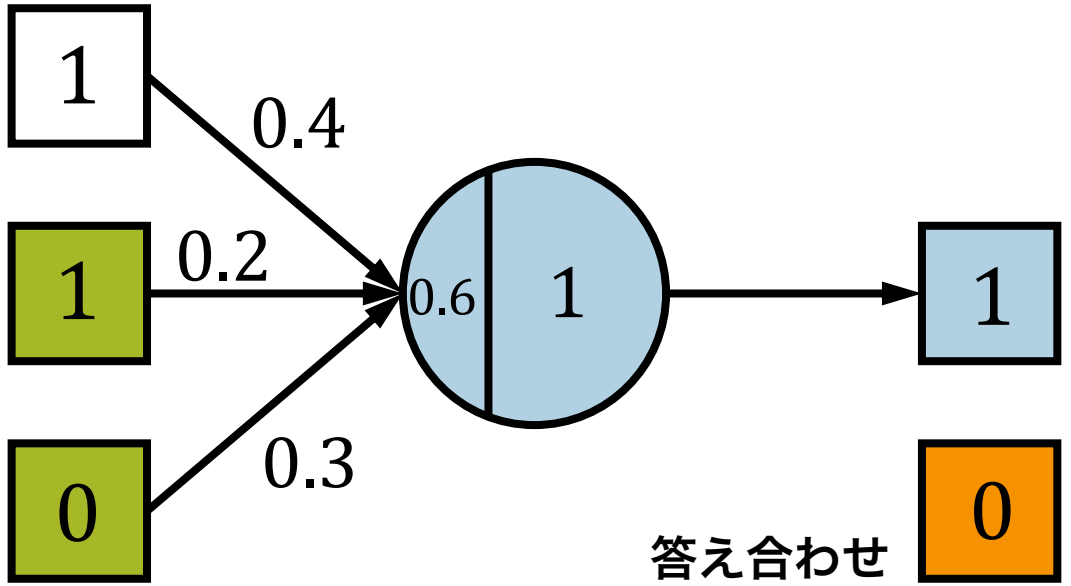
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = 0.4 + 1(0-1)0.1 = 0.3$$

$$w_1^{(new)} = 0.2 + 1(0-1)0.1 = 0.1$$

$$w_2^{(new)} = 0.3 + 0(0-1)0.1 = 0.3$$

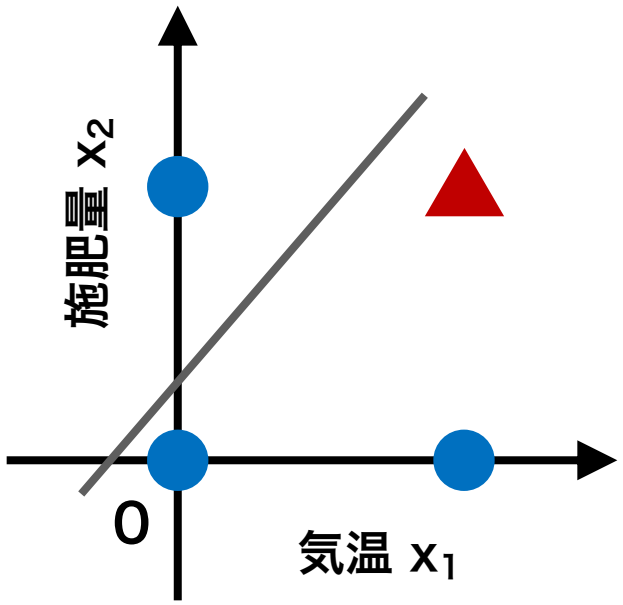
パーセプトロン学習則

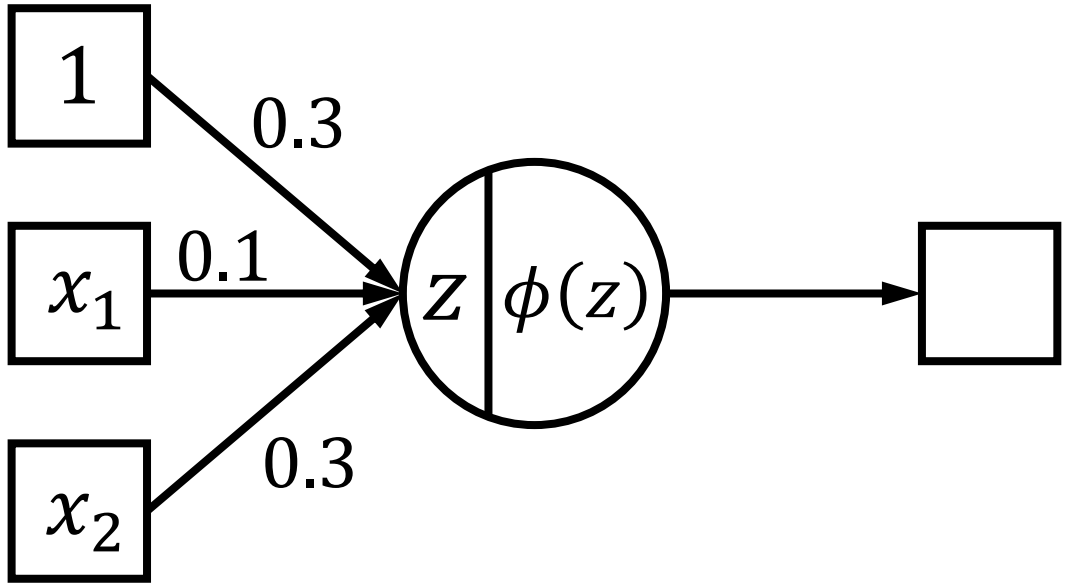
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





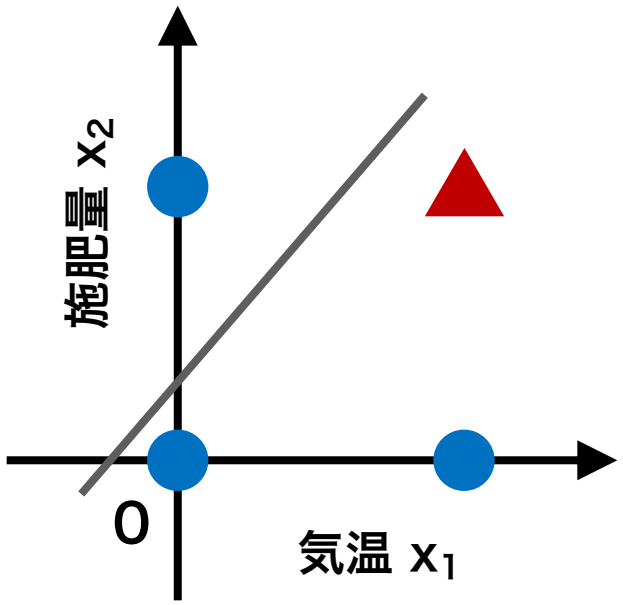
パーセプトロン学習則

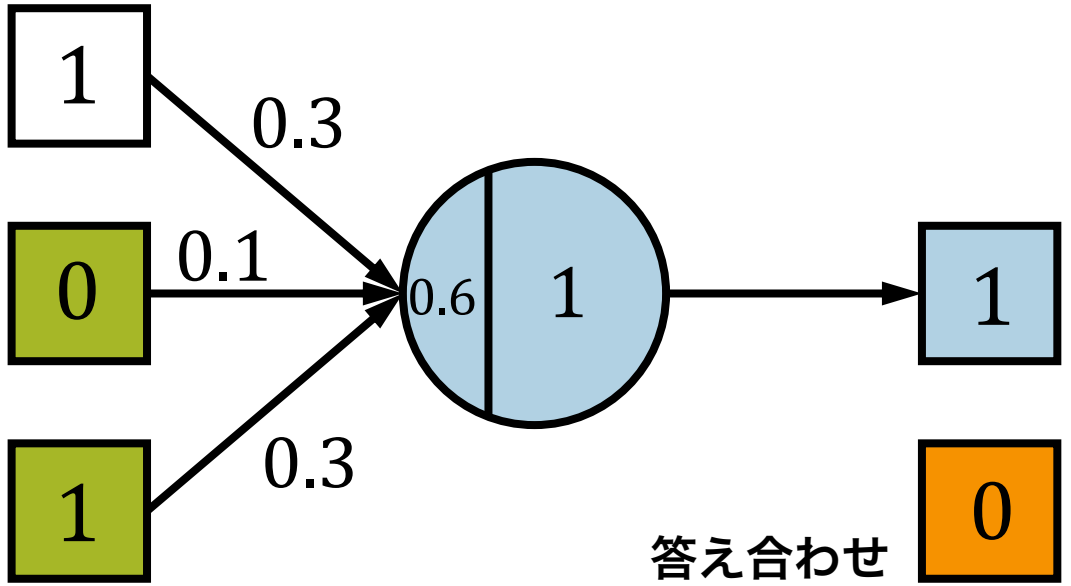
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = 0.3 + 1(0-1)0.1 = 0.2$$

$$w_1^{(new)} = 0.1 + 0(0-1)0.1 = 0.1$$

$$w_2^{(new)} = 0.3 + 1(0-1)0.1 = 0.2$$

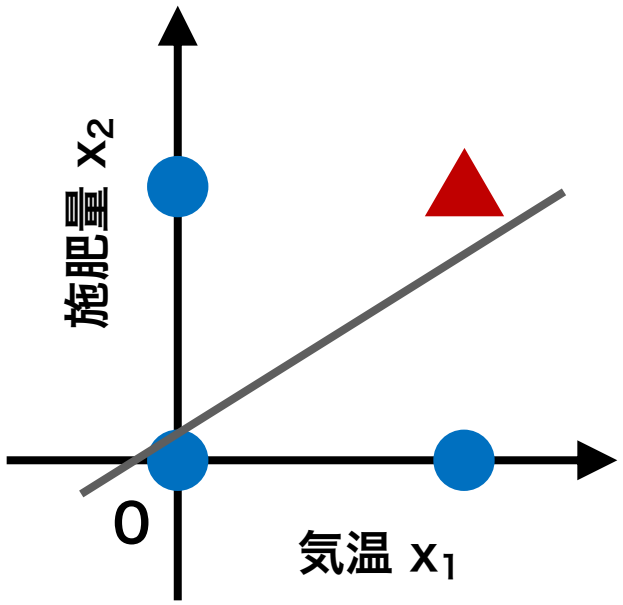
パーセプトロン学習則

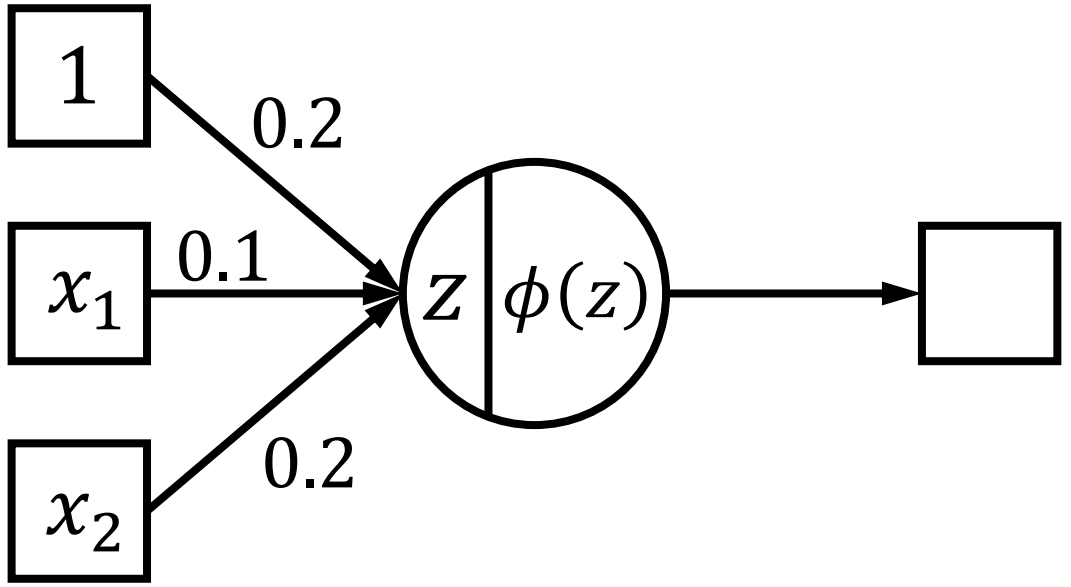
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





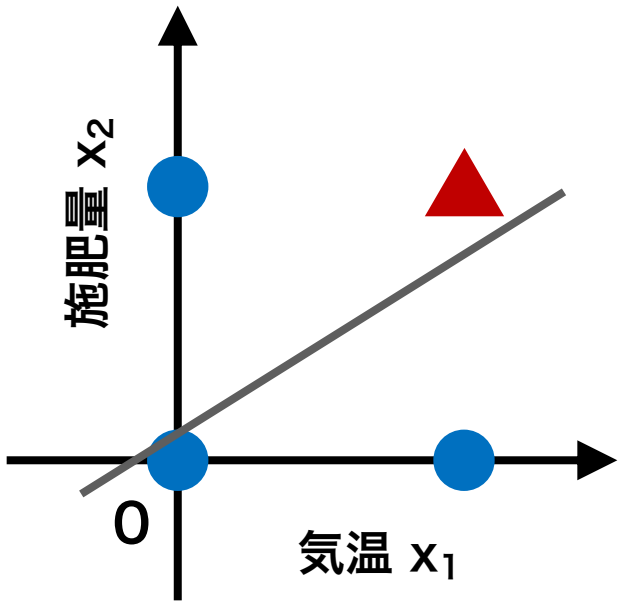
パーセプトロン学習則

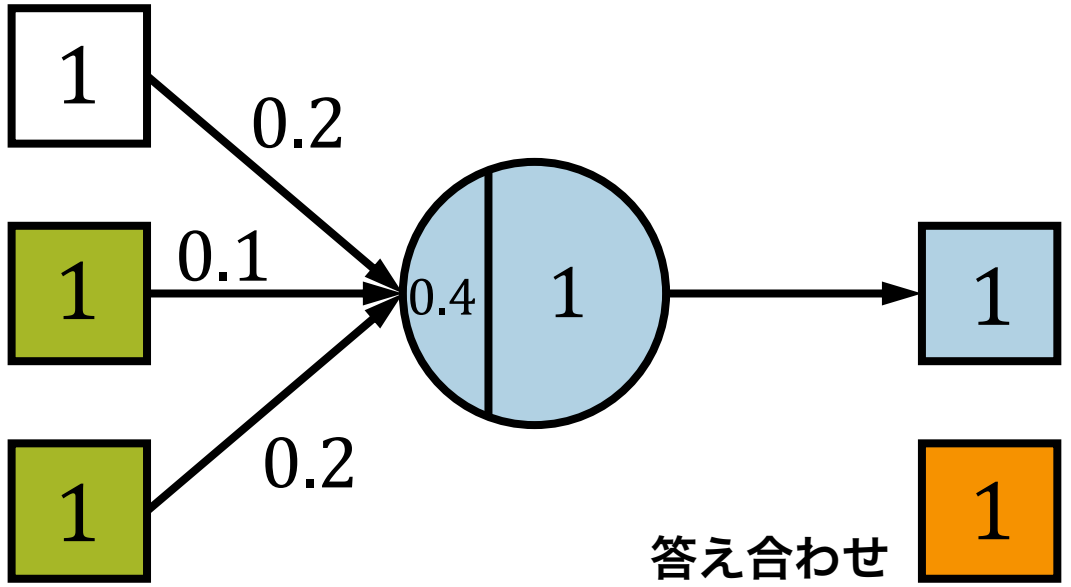
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = 0.2 + 1(1-1)0.1 = 0.2$$

$$w_1^{(new)} = 0.1 + 1(1-1)0.1 = 0.1$$

$$w_2^{(new)} = 0.2 + 1(1-1)0.1 = 0.2$$

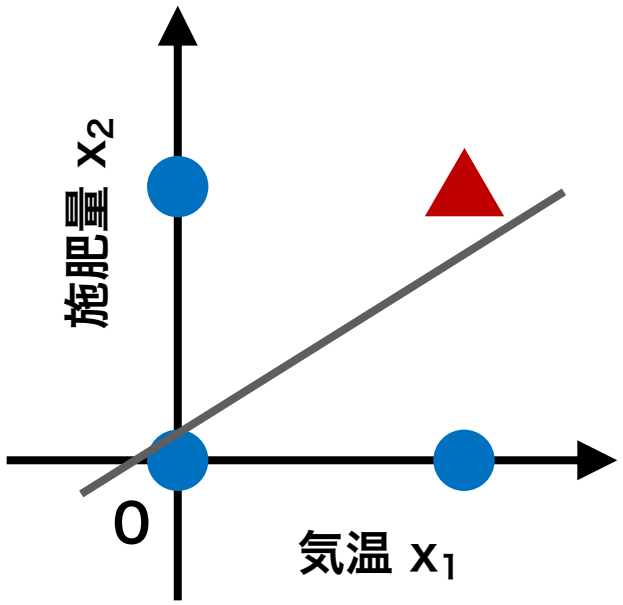
パーセプトロン学習則

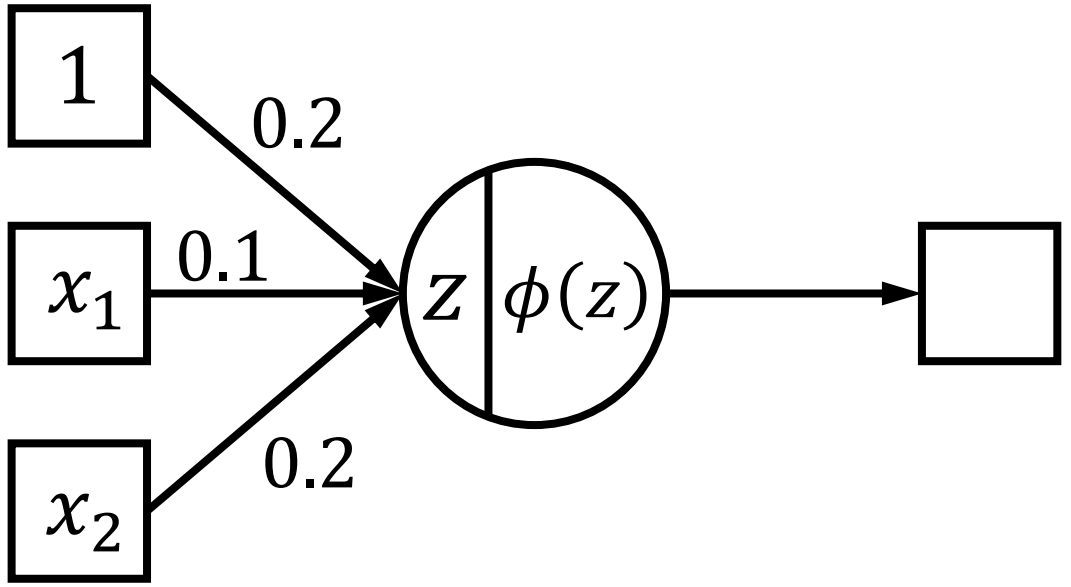
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





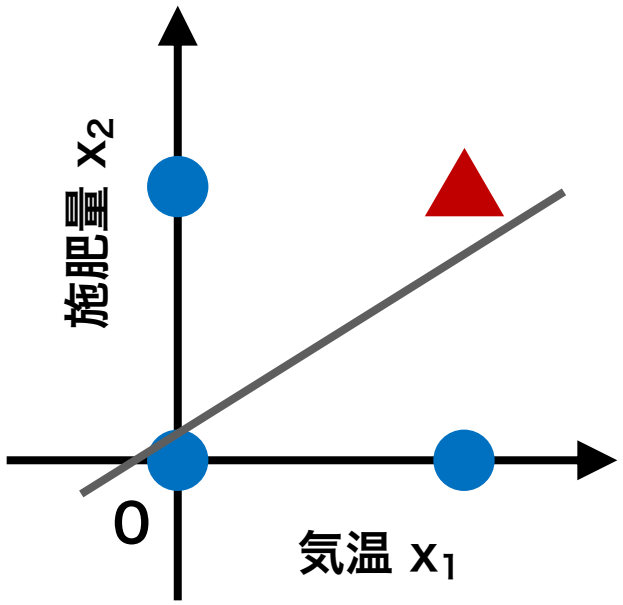
パーセプトロン学習則

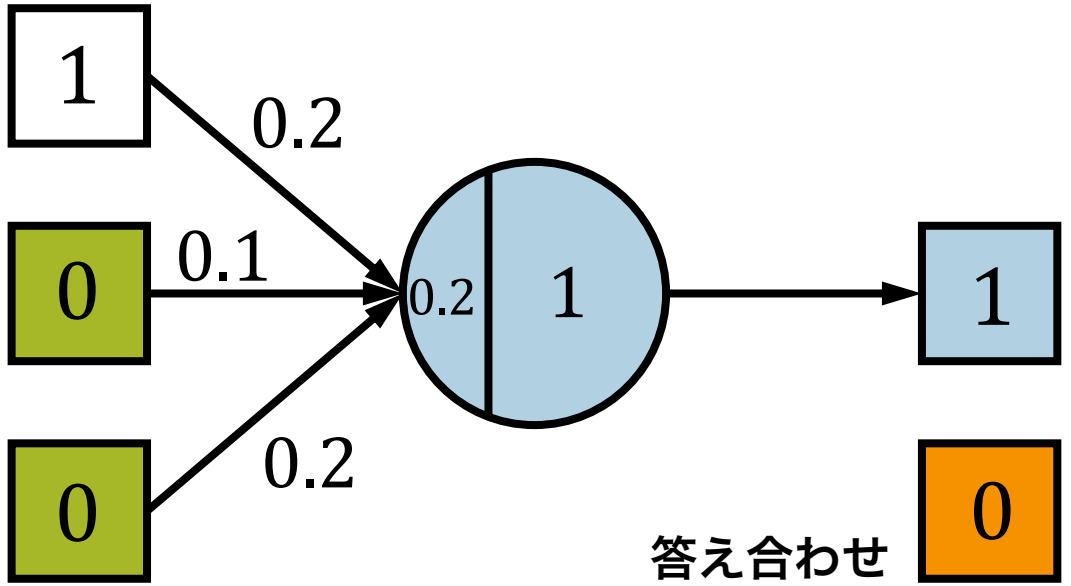
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = 0.2 + 1(0-1)0.1 = 0.1$$

$$w_1^{(new)} = 0.1 + 0(0-1)0.1 = 0.1$$

$$w_2^{(new)} = 0.2 + 0(0-1)0.1 = 0.2$$

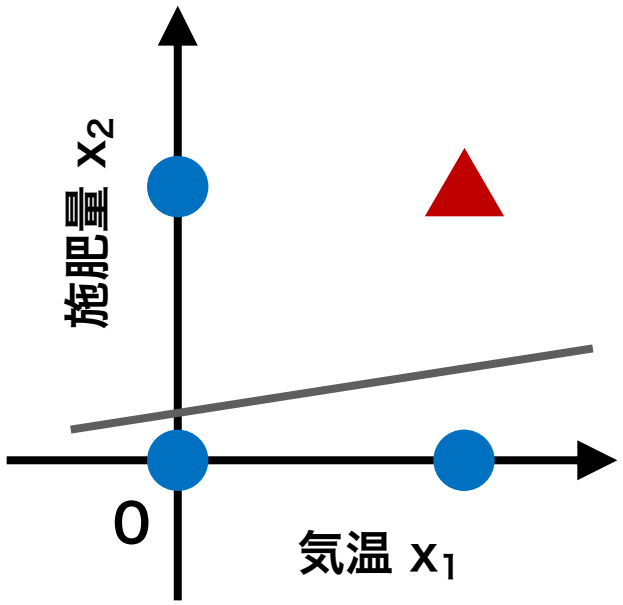
パーセプトロン学習則

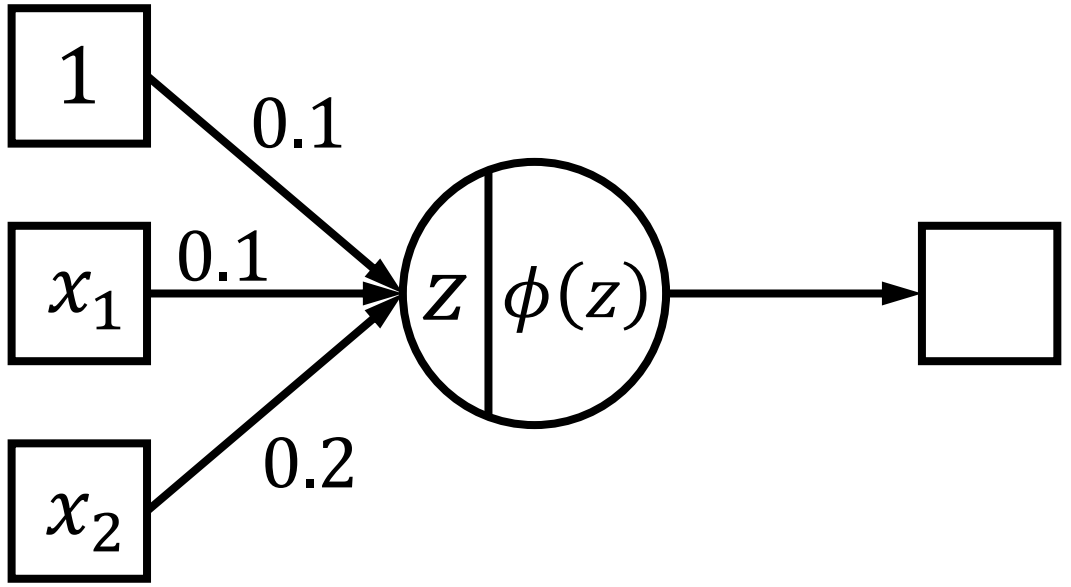
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





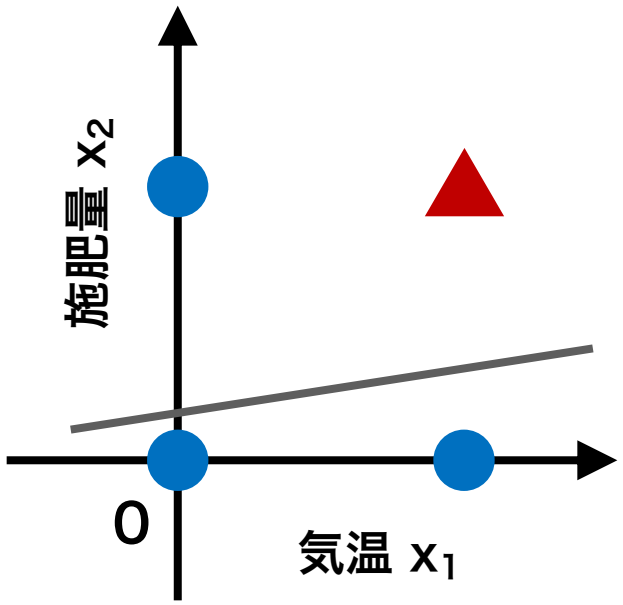
パーセプトロン学習則

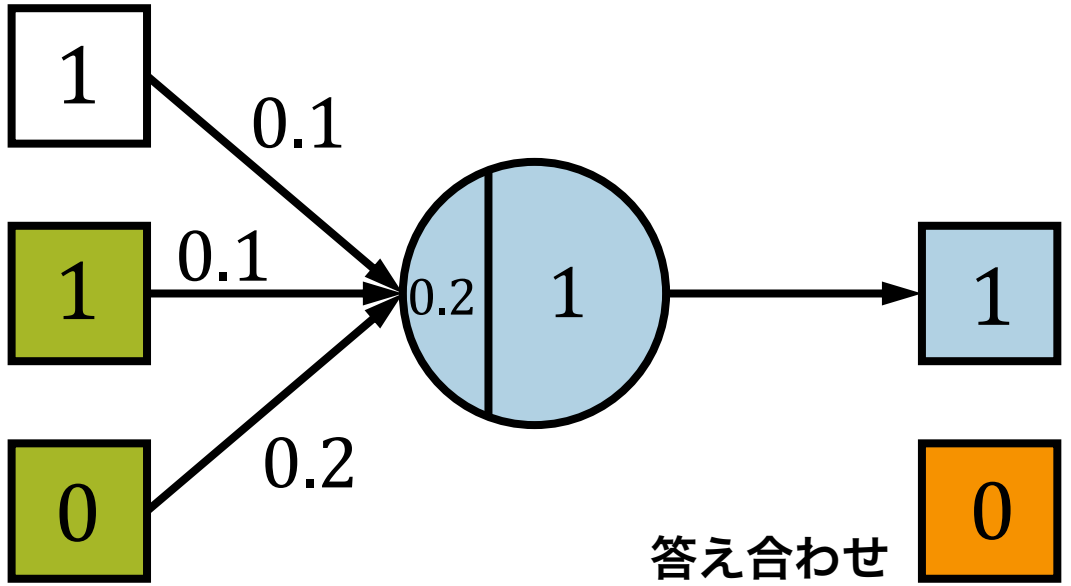
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = 0.1 + 1(0-1)0.1 = 0.0$$

$$w_1^{(new)} = 0.1 + 1(0-1)0.1 = 0.0$$

$$w_2^{(new)} = 0.2 + 0(0-1)0.1 = 0.2$$

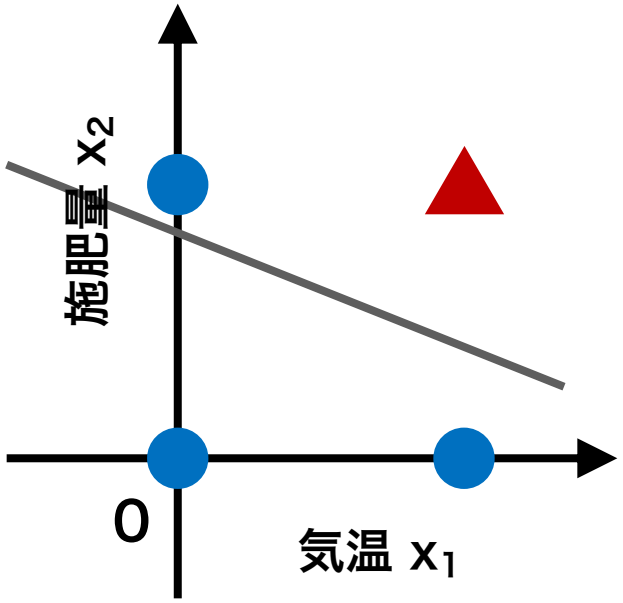
パーセプトロン学習則

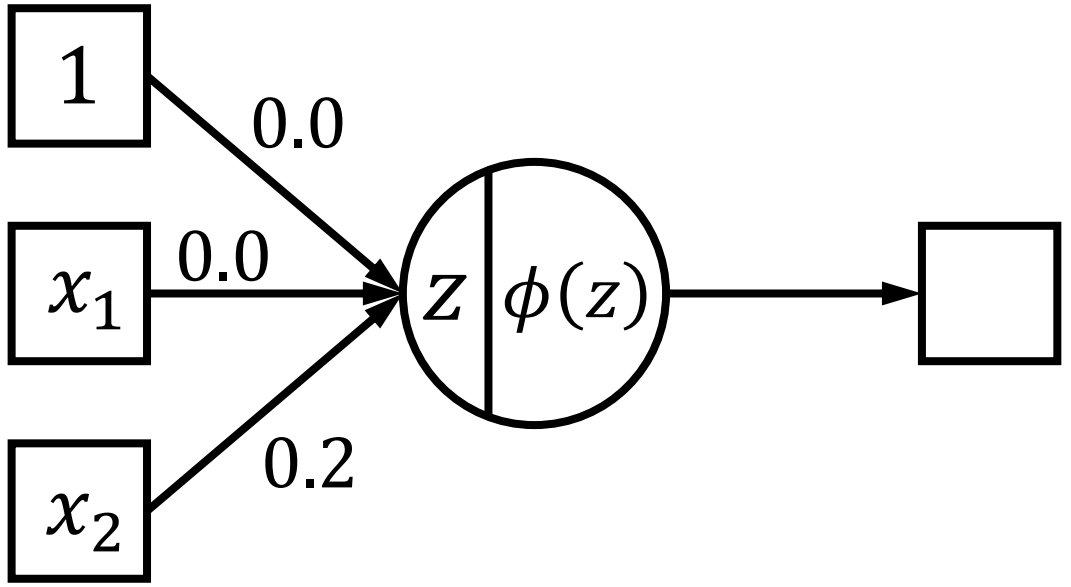
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





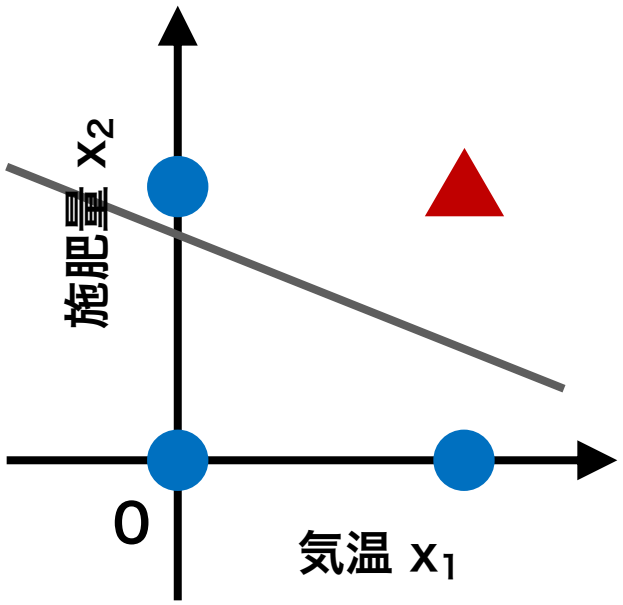
パーセプトロン学習則

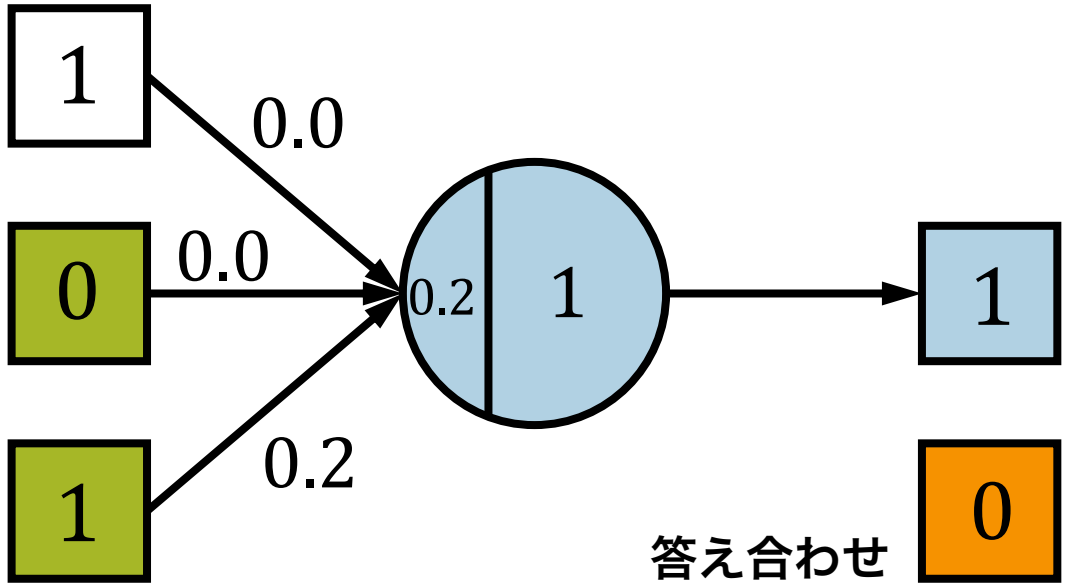
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = 0.0 + 1(0-1)0.1 = -0.1$$

$$w_1^{(new)} = 0.0 + 0(0-1)0.1 = 0.0$$

$$w_2^{(new)} = 0.2 + 1(0-1)0.1 = 0.1$$

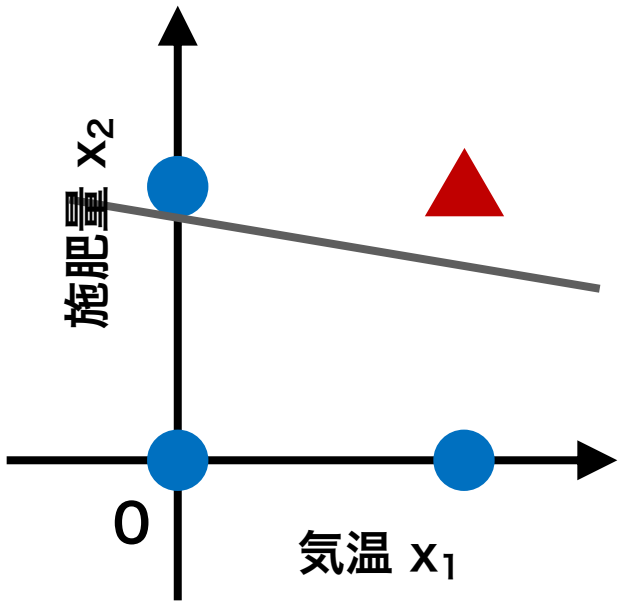
パーセプトロン学習則

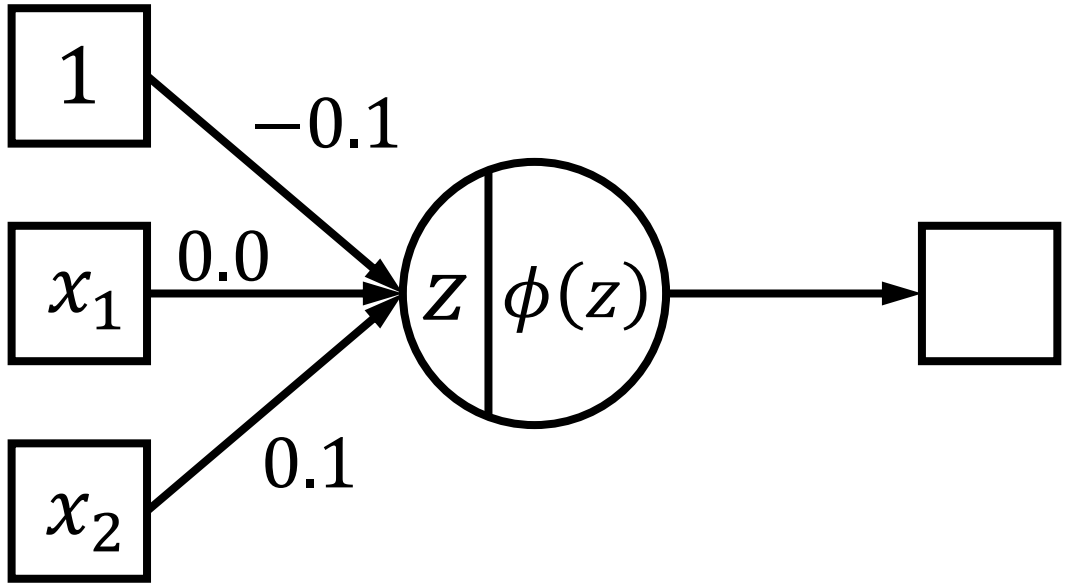
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





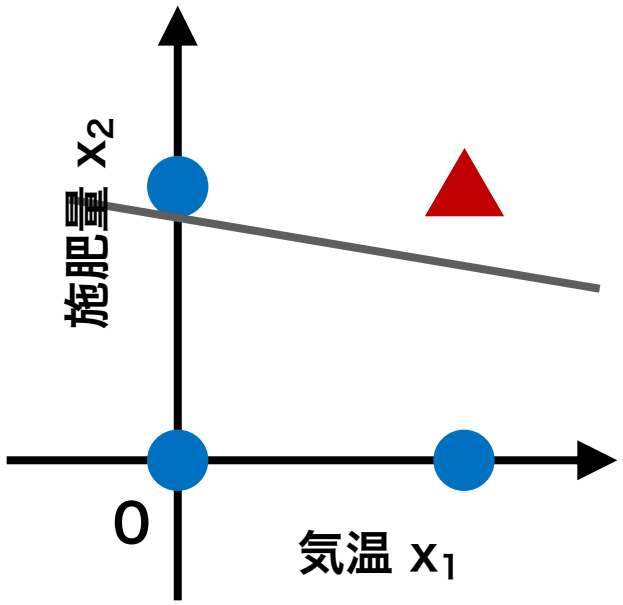
パーセプトロン学習則

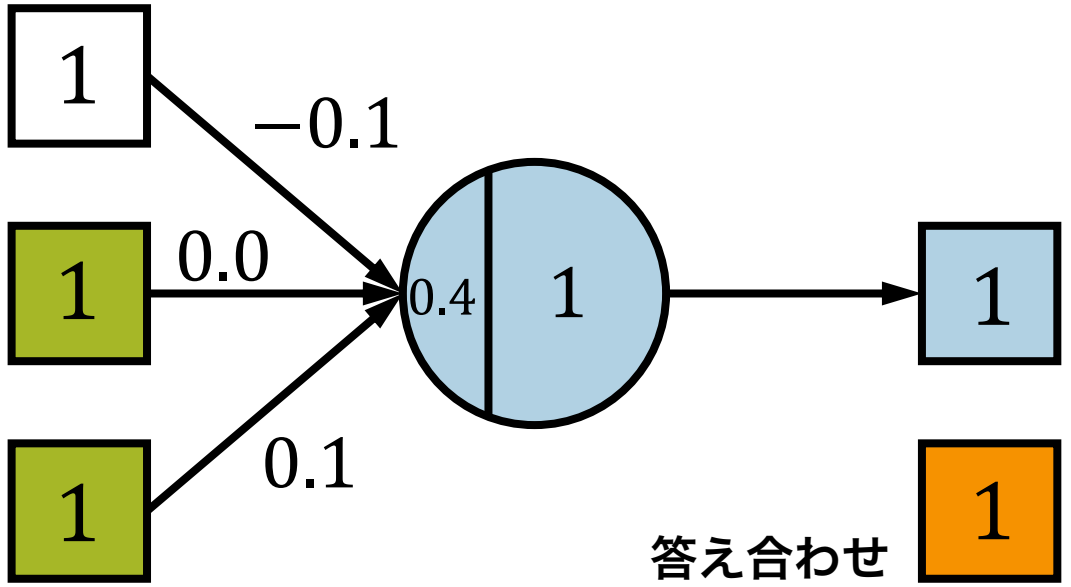
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = -0.1 + 1(1-1)0.1 = -0.1$$

$$w_1^{(new)} = 0.0 + 0(1-1)0.1 = 0.0$$

$$w_2^{(new)} = 0.1 + 1(1-1)0.1 = 0.1$$

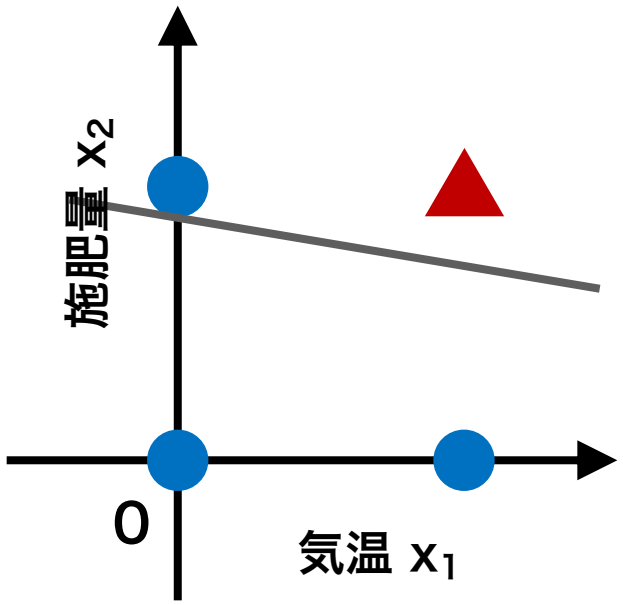
パーセプトロン学習則

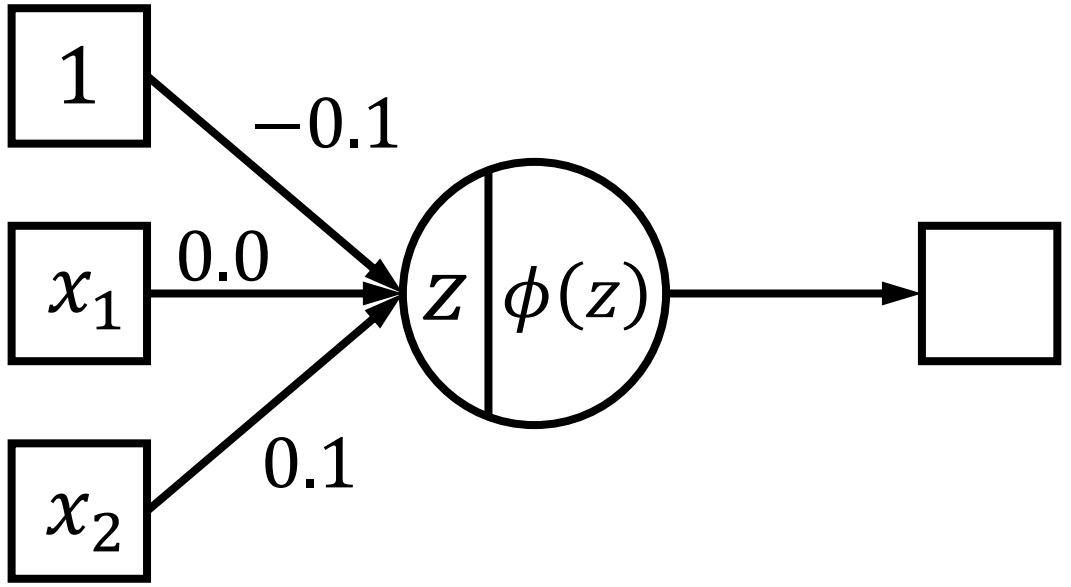
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





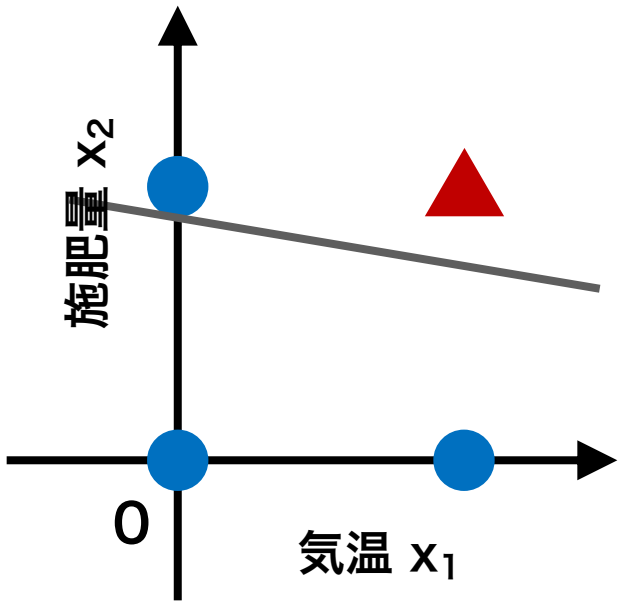
パーセプトロン学習則

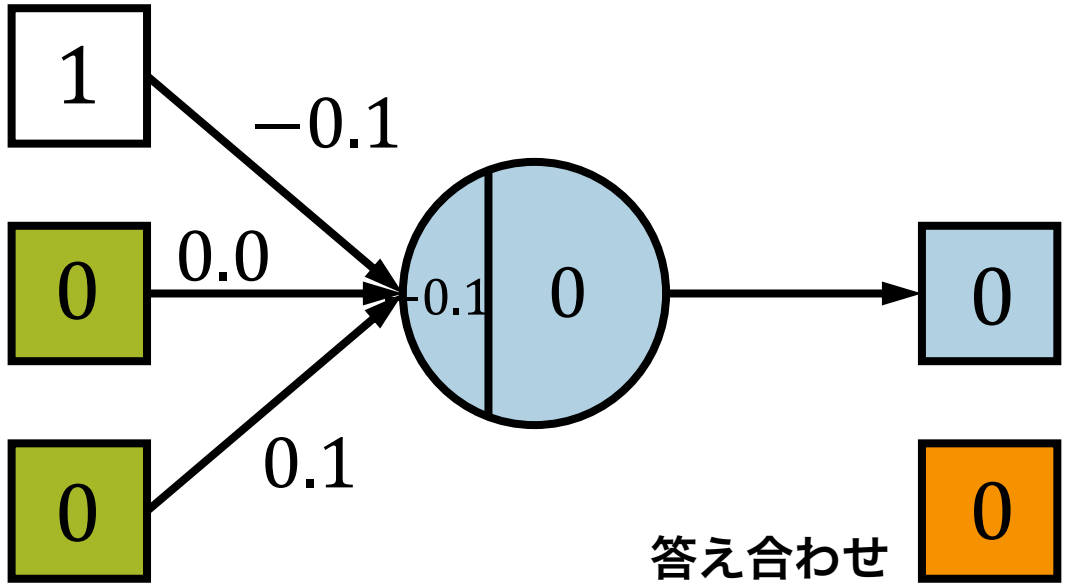
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = -0.1 + 1(0-0)0.1 = -0.1$$

$$w_1^{(new)} = 0.0 + 0(0-0)0.1 = 0.0$$

$$w_2^{(new)} = 0.1 + 0(0-0)0.1 = 0.1$$

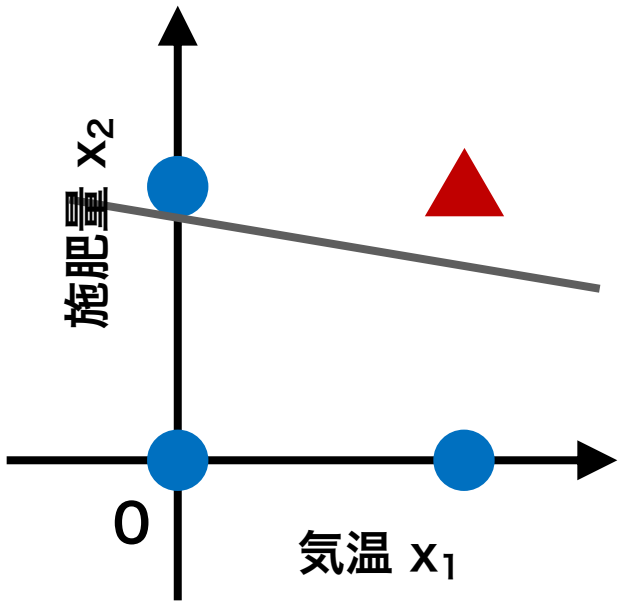
パーセプトロン学習則

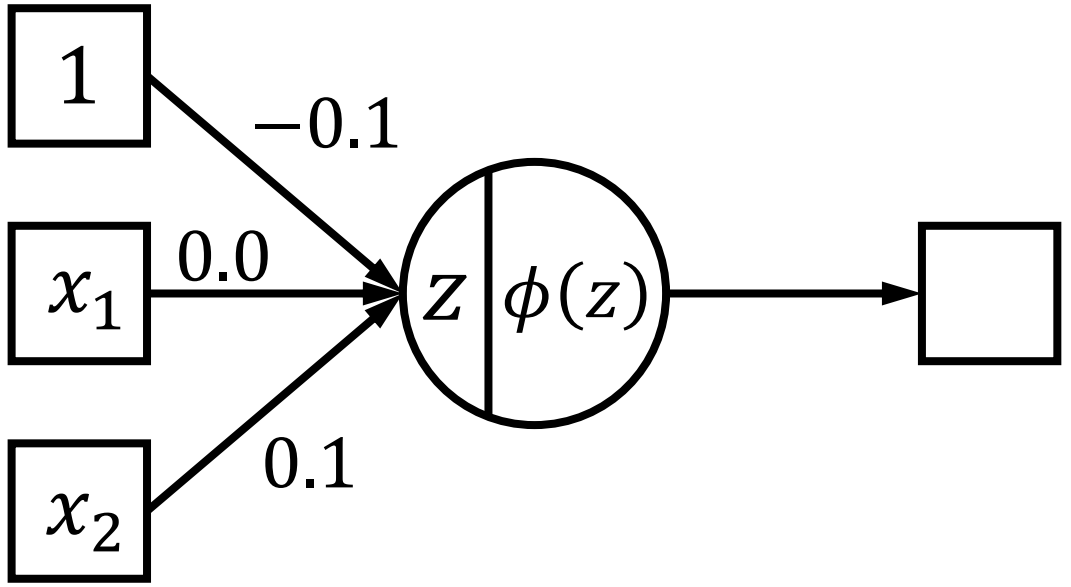
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





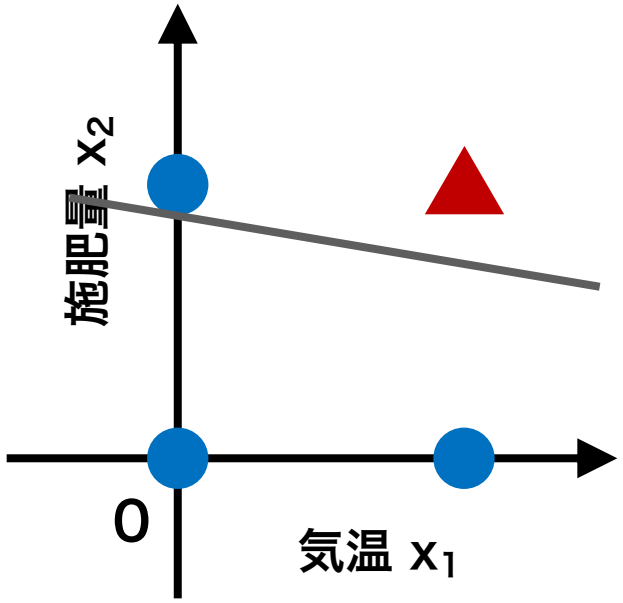
パーセプトロン学習則

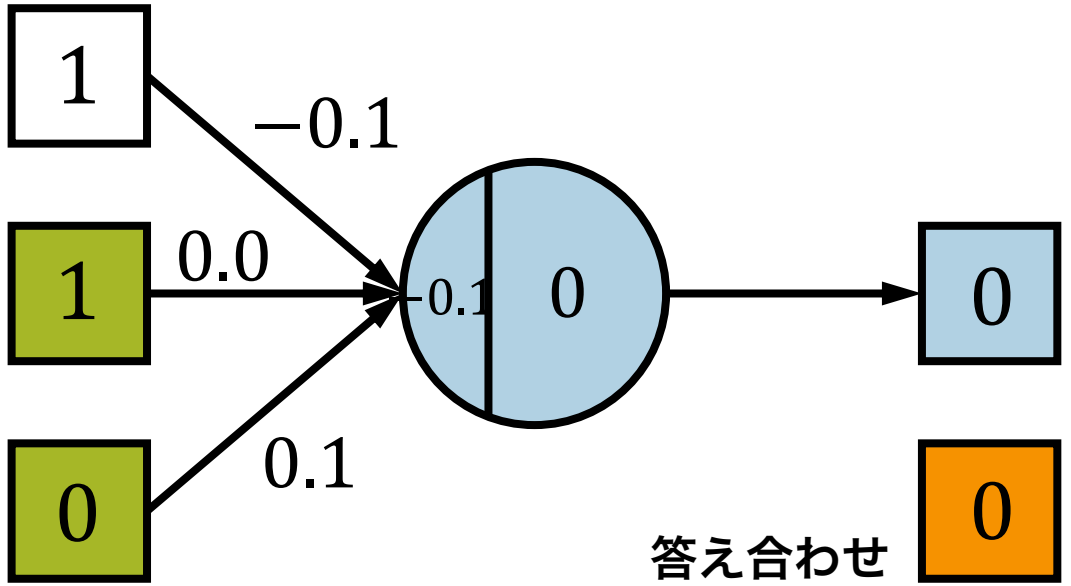
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = -0.1 + 1(0-0)0.1 = -0.1$$

$$w_1^{(new)} = 0.0 + 0(0-0)0.1 = 0.0$$

$$w_2^{(new)} = 0.1 + 0(0-0)0.1 = 0.1$$

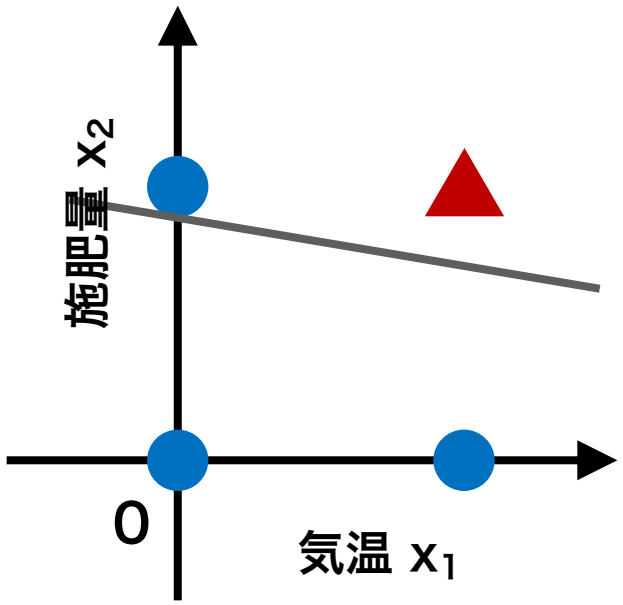
パーセプトロン学習則

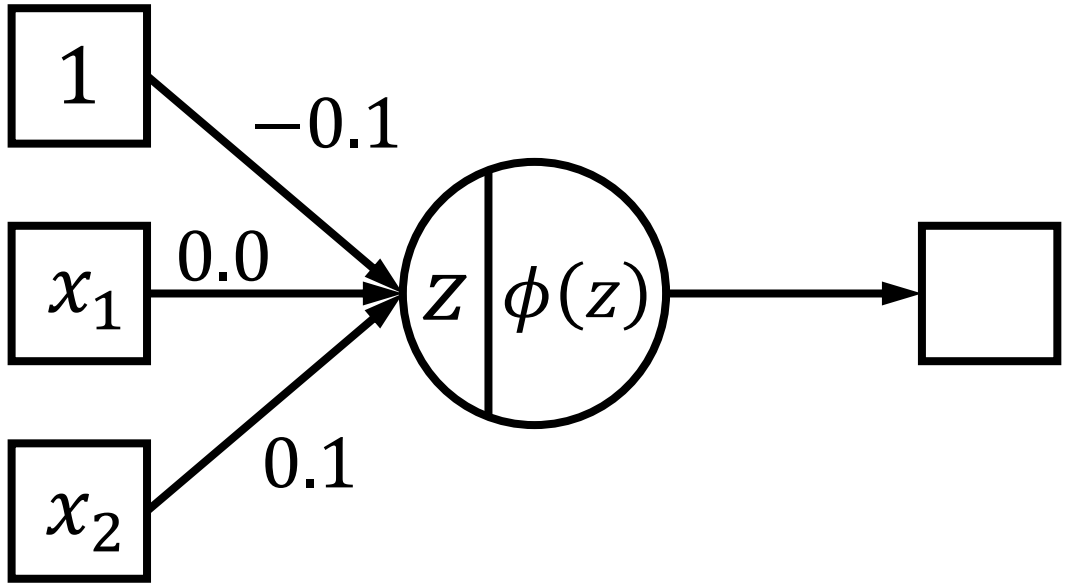
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





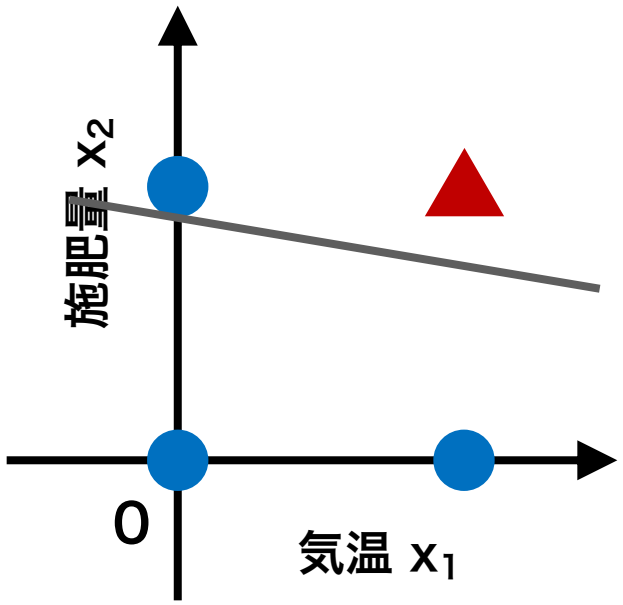
パーセプトロン学習則

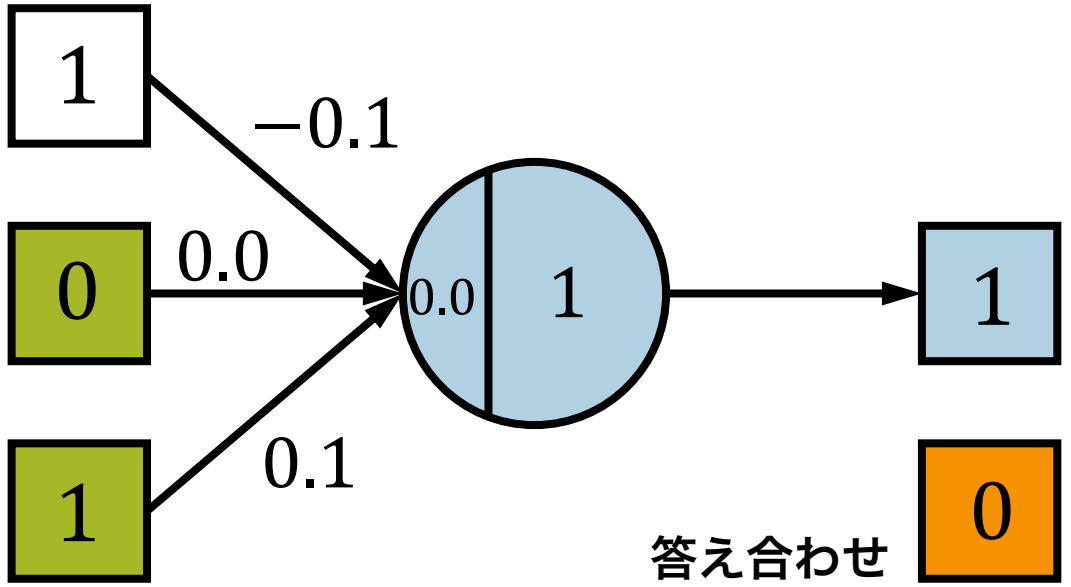
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = -0.1 + 1(0-1)0.1 = -0.2$$

$$w_1^{(new)} = 0.0 + 0(0-1)0.1 = 0.0$$

$$w_2^{(new)} = 0.1 + 1(0-1)0.1 = 0.0$$

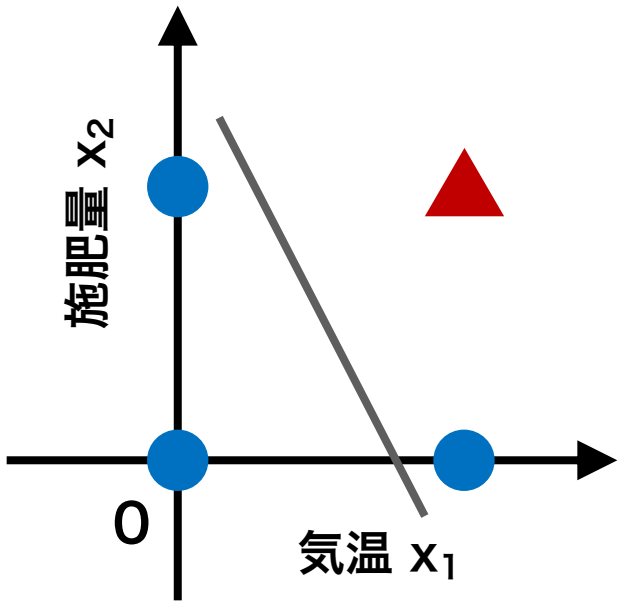
パーセプトロン学習則

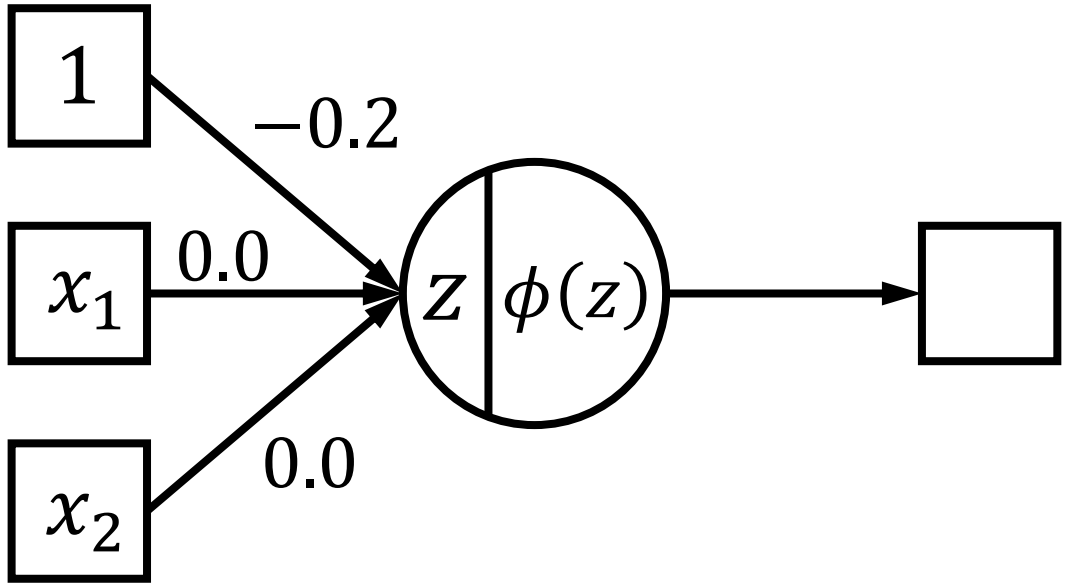
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





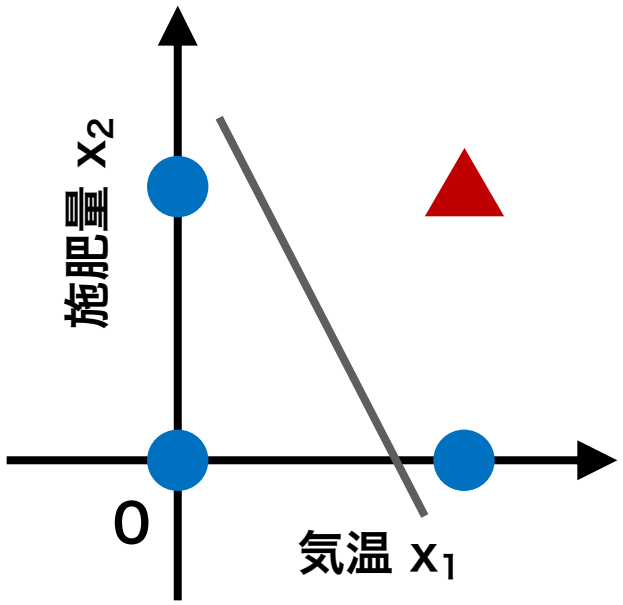
パーセプトロン学習則

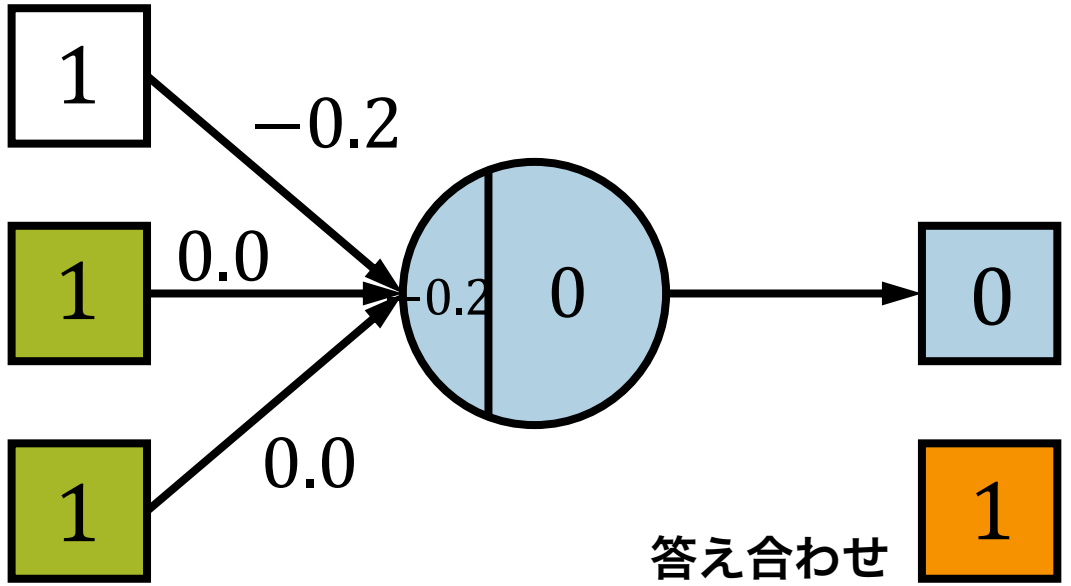
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = -0.2 + 1(1-0)0.1 = -0.1$$

$$w_1^{(new)} = 0.0 + 1(1-0)0.1 = 0.1$$

$$w_2^{(new)} = 0.1 + 1(1-0)0.1 = 0.2$$

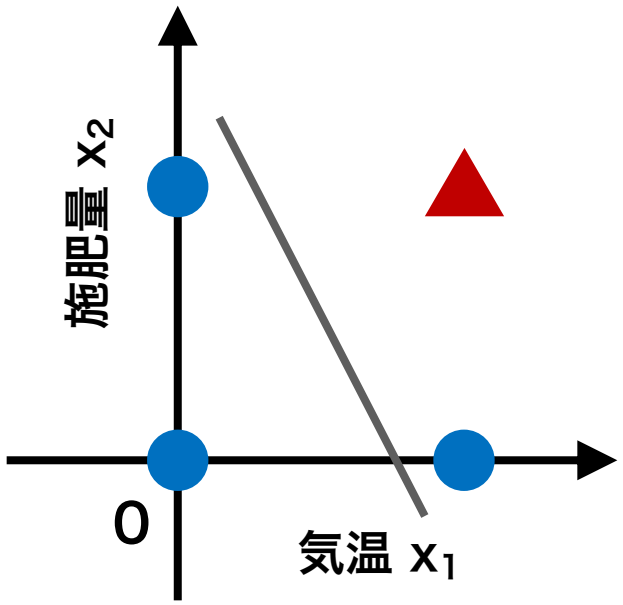
パーセプトロン学習則

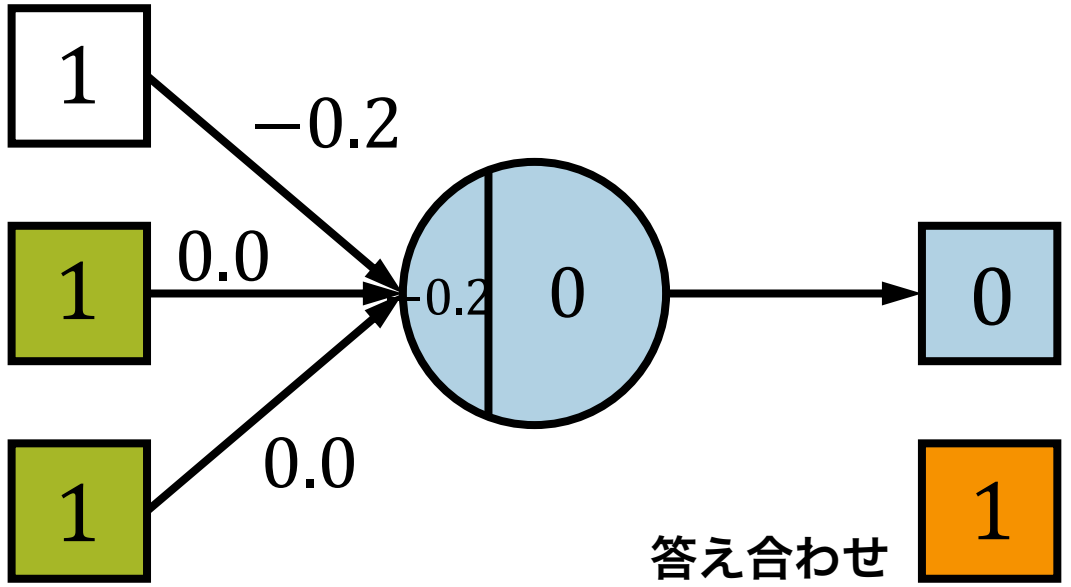
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





$$w_0^{(new)} = -0.2 + 1(1-0)0.1 = -0.1$$

$$w_1^{(new)} = 0.0 + 1(1-0)0.1 = 0.1$$

$$w_2^{(new)} = 0.1 + 1(1-0)0.1 = 0.2$$

パーセプトロン学習則

$$w_j^{new} = w_j^{old} + x_j \underbrace{(y - \hat{y})}_{\text{損失}} \underbrace{\eta}_{\text{学習率}}$$

パーセプトロン学習則

教師ラベルが +1 のときに -1 と間違って予測した場合、この項がプラスになる。このとき w^{new} の値が大きくなり、次にデータを入力したときに $w_j x_j$ の値がより大きくなり、閾値を超えやすくなる。

$$w_j^{new} = w_j^{old} + \overbrace{x_j(y - \hat{y})\eta}$$

パーセプトロン学習則

教師ラベルが -1 のときに +1 と間違って予測した場合、この項がマイナスになる。このとき w^{new} の値が小さくなり、次にデータを入力したときに $w_j x_j$ の値がより小さくなり、閾値を超えにくくなる。

$$w_j^{new} = w_j^{old} + \overbrace{x_j(y - \hat{y})\eta}$$

単層ニューラルネットワーク

- パーセプトロン
- ADALINE
- ニューラルネットワーク

パーセプトロン

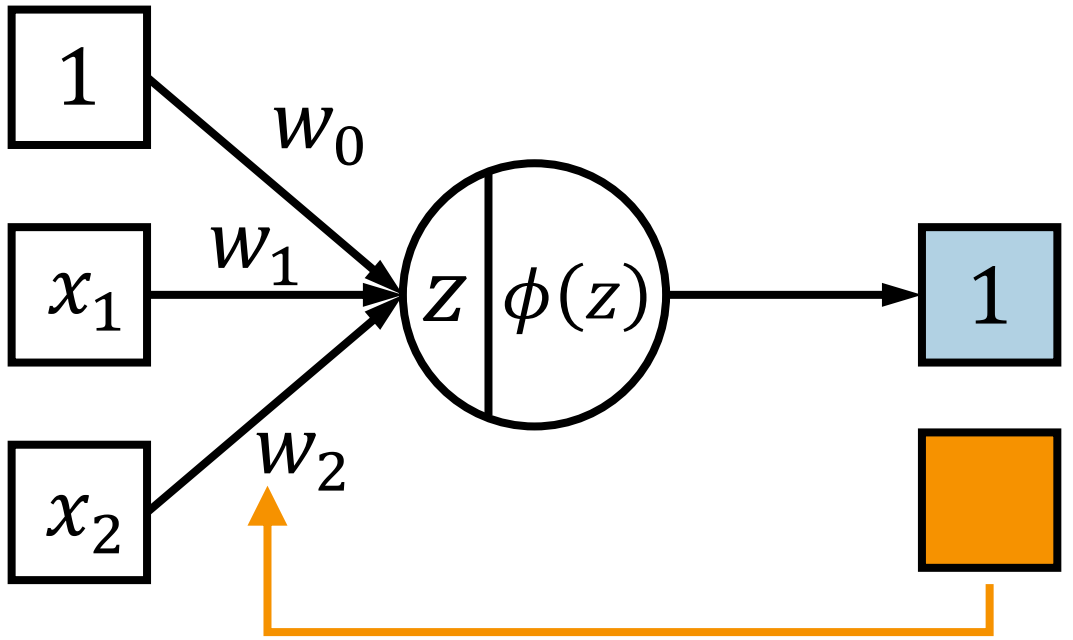
パーセプトロンは、複数の入力値に重みをかけ和を計算し、その和の符号に応じて値を出力するアルゴリズムである。訓練時において、パーセプトロンの出力値（予測値）とラベルの差を使用して、パラメータの更新を行なっている。

入力　x_1　x_2

処理　$z = w_0 + w_1 x_1 + w_2 x_2$

処理　$\phi(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$

出力　1



ADALINE

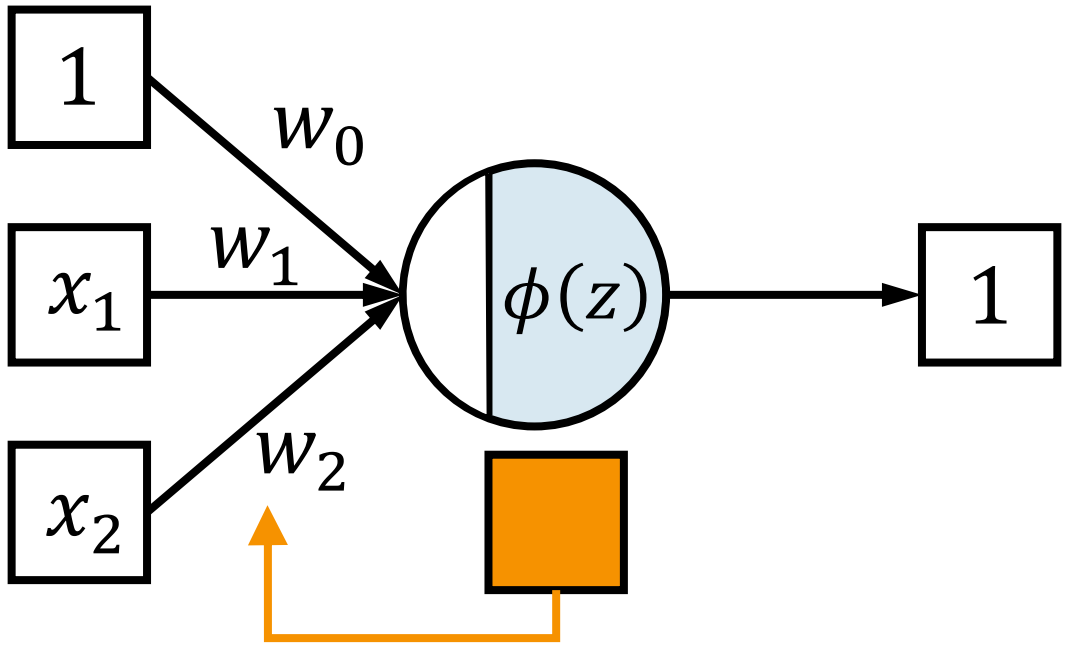
ADALINE はパーセプトロンとほぼ同じ構造であり、単層ニューラルネットワークの一種である。ADALINE における訓練時のパラメータ更新は、ADALINE の出力値（予測値）を使用しない。その代わり、活性化関数の出力値 $\phi(z)$ とラベルの差に基づいてパラメータを更新している。なお、ϕ は恒等関数 $\phi(z)=z$ である。

入力　x_1　x_2

処理　$z = w_0 + w_1 x_1 + w_2 x_2$

処理　$$\phi(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$$

出力　1



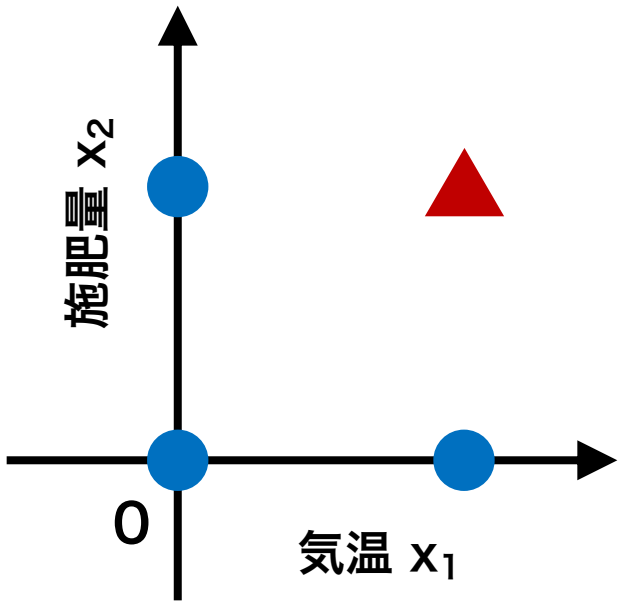
Widrow-Hoff 学習則

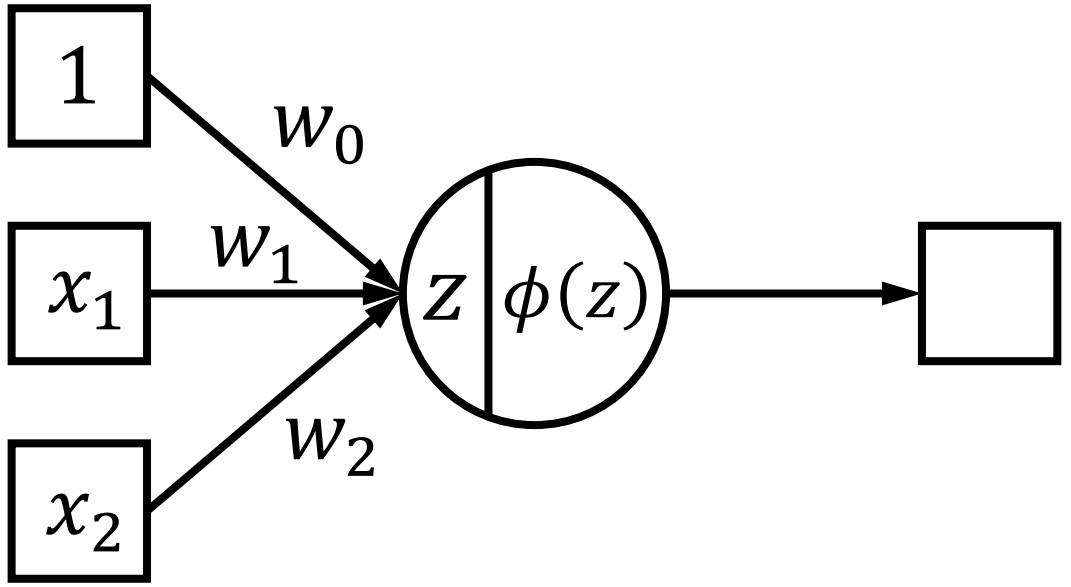
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1





Widrow-Hoff 学習則

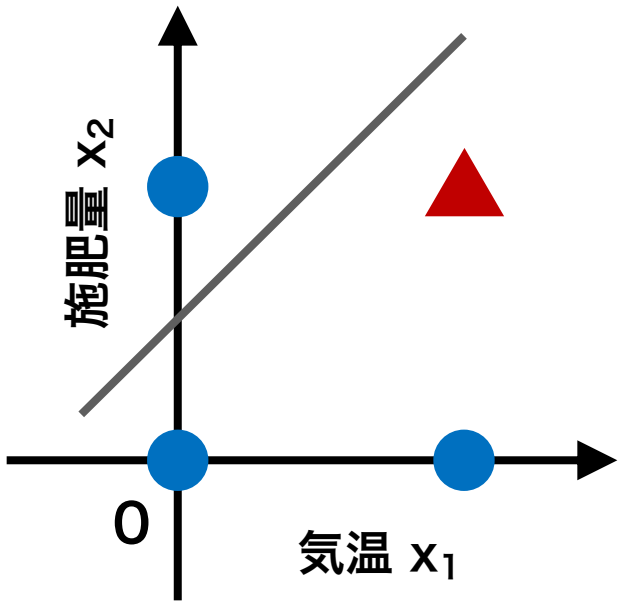
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

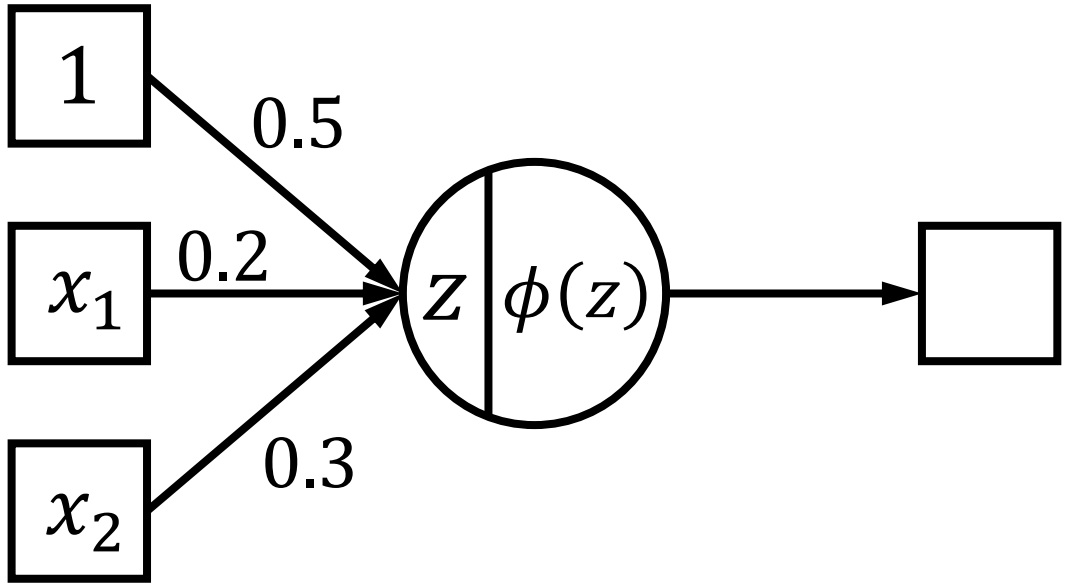
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

※図はイメージである。





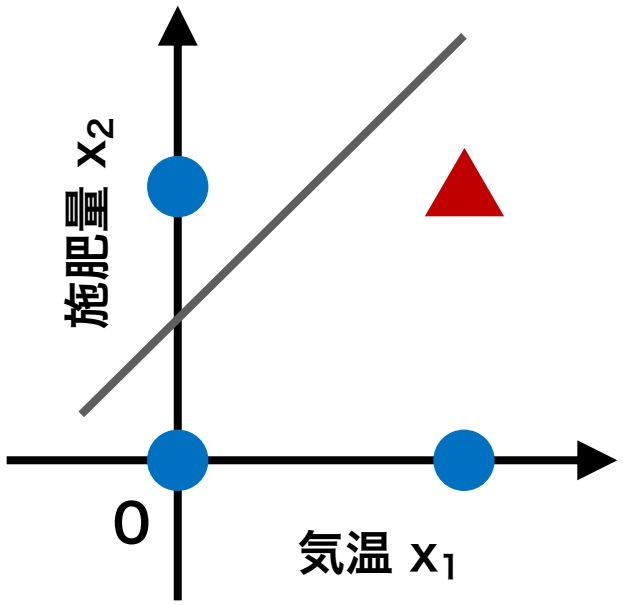
Widrow-Hoff 学習則

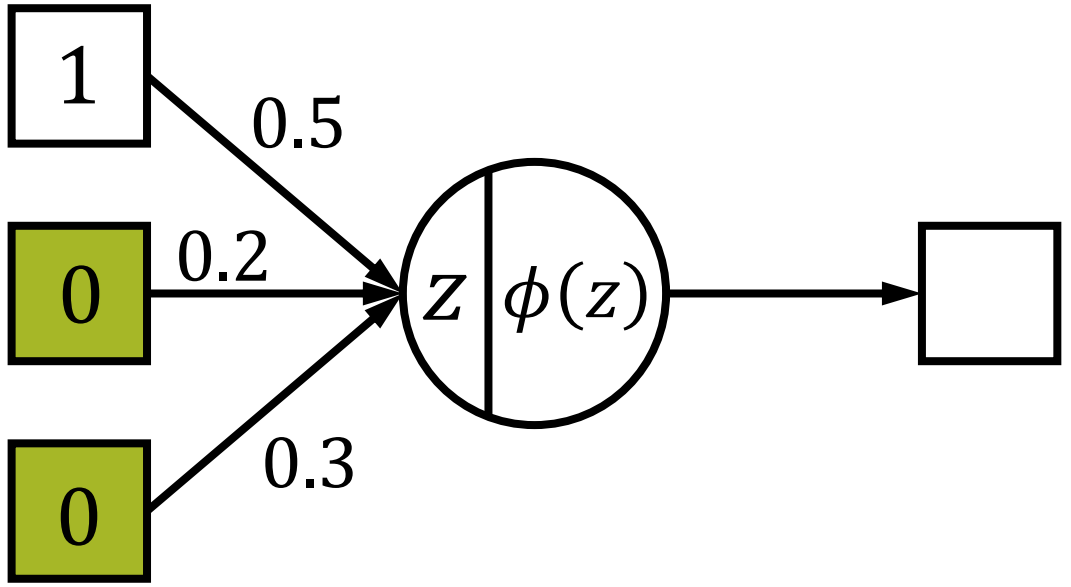
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1





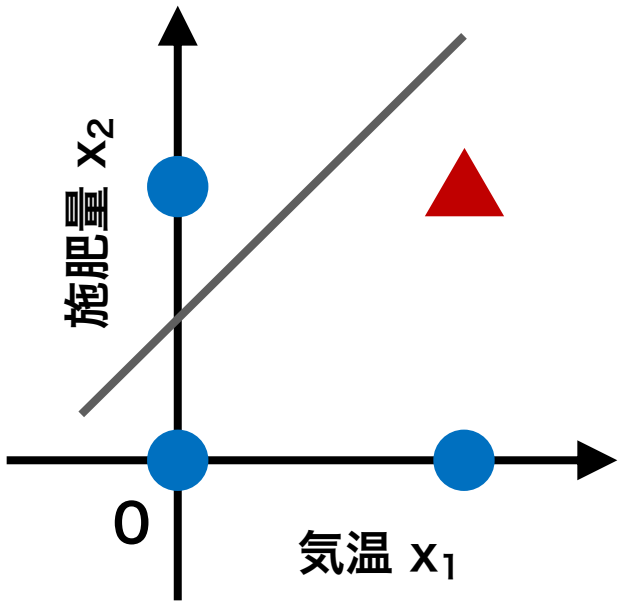
Widrow-Hoff 学習則

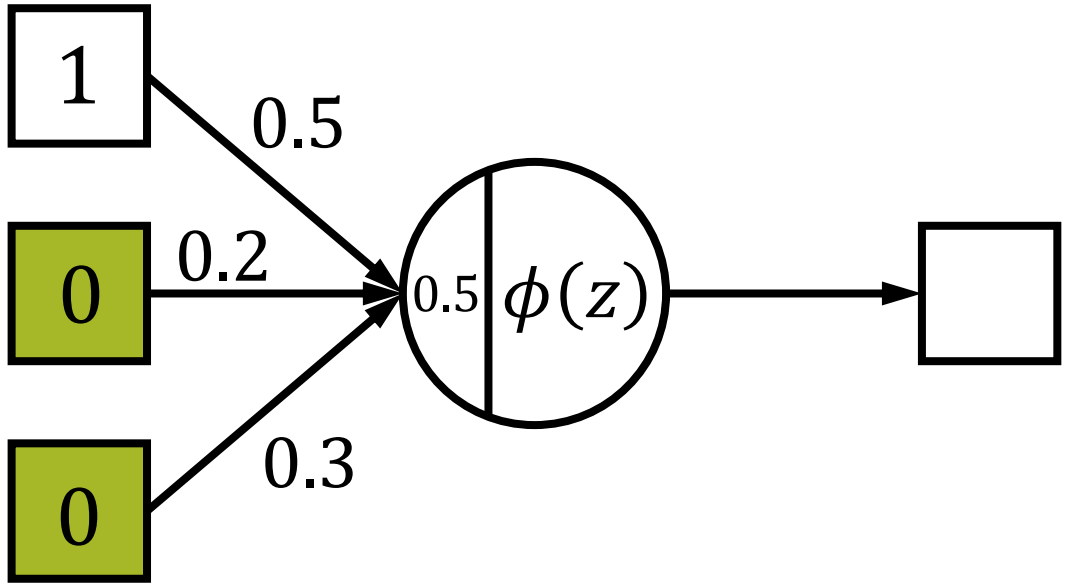
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1





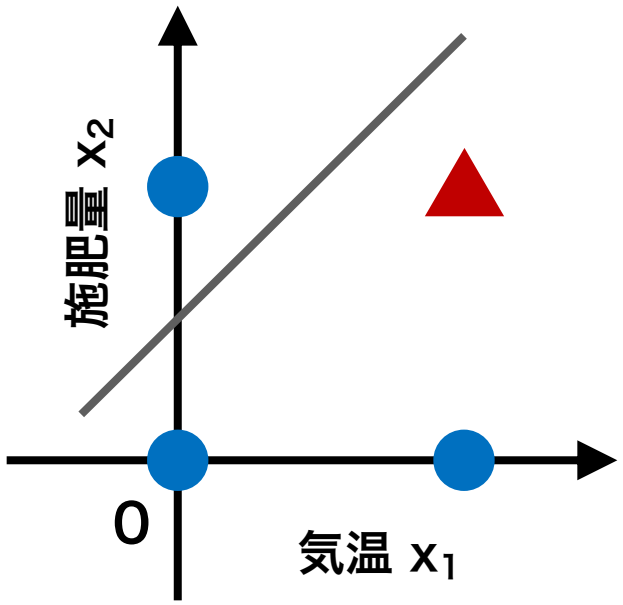
Widrow-Hoff 学習則

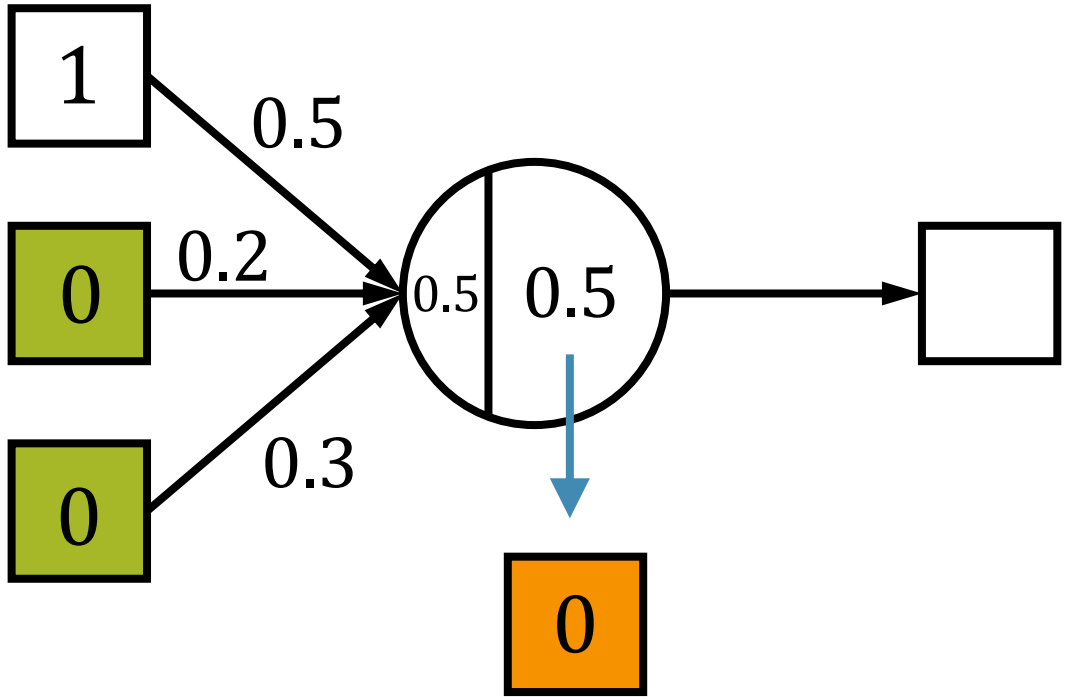
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1





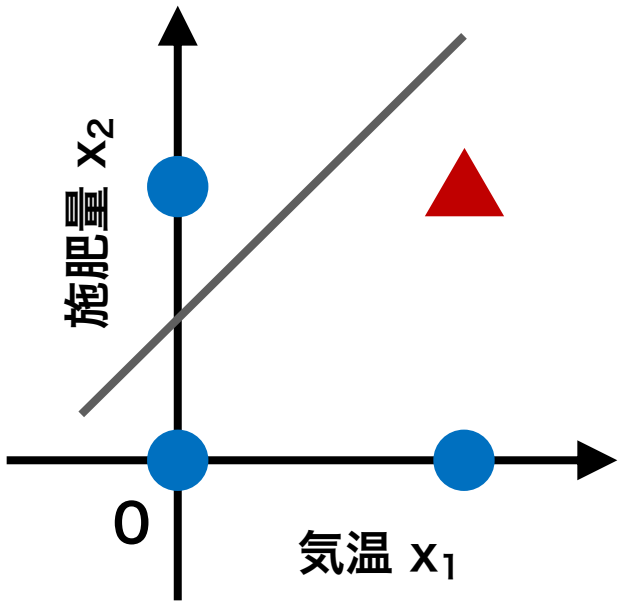
Widrow-Hoff 学習則

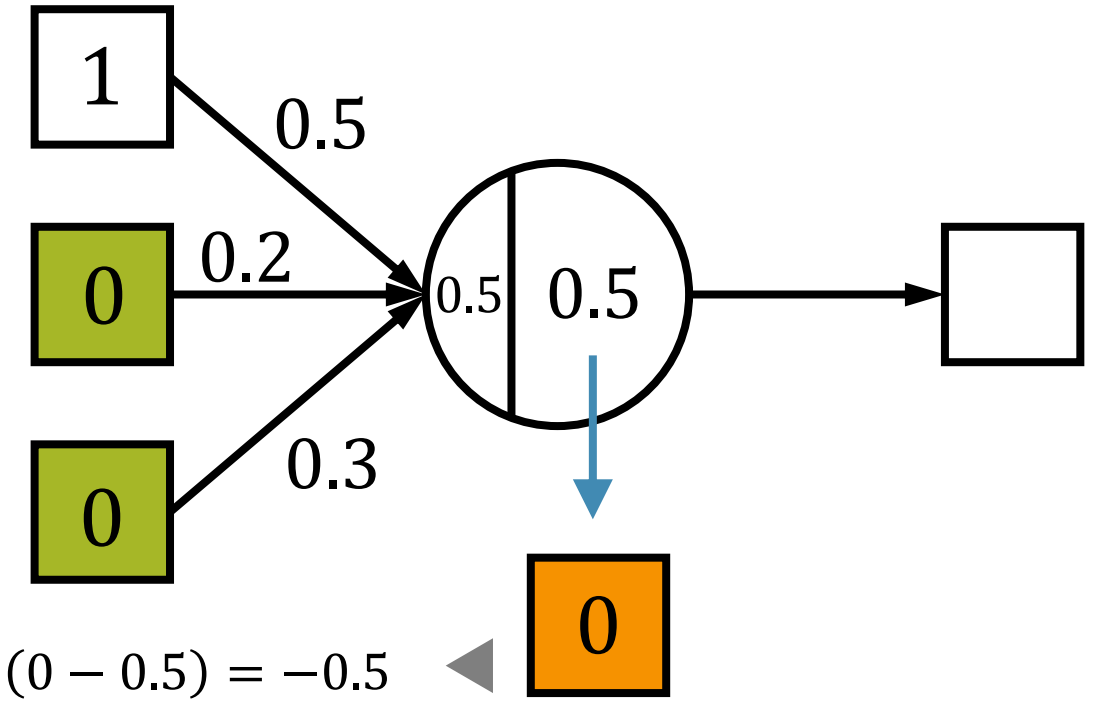
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1





Widrow-Hoff 学習則

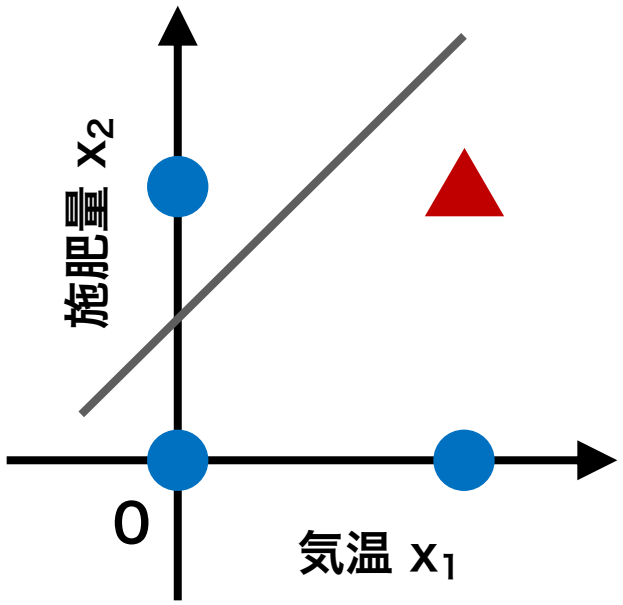
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

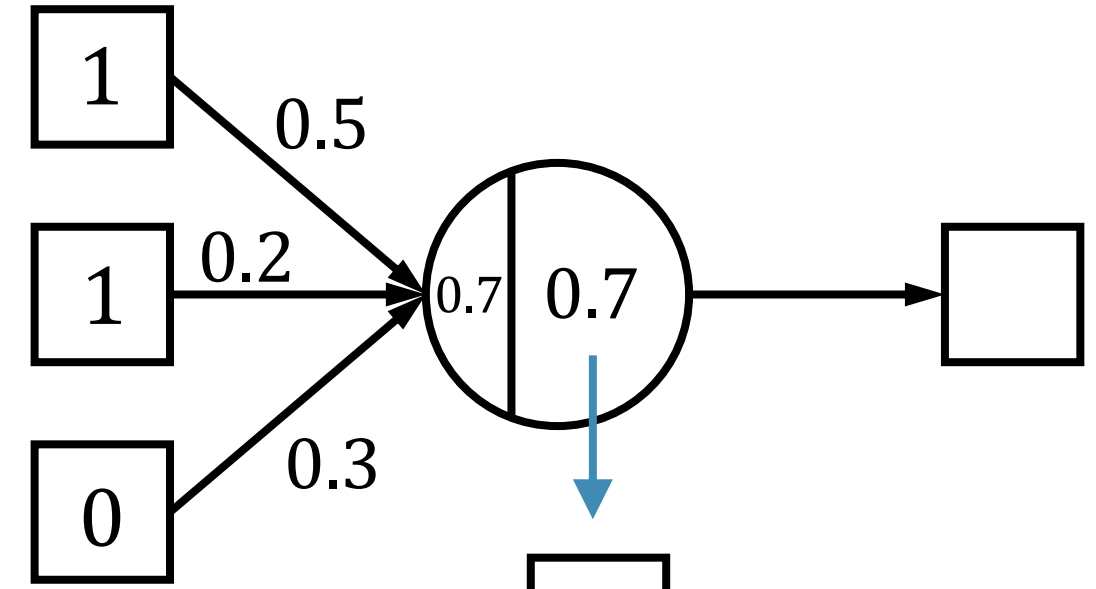
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

ADALINE ではすべての訓練データを用いて損失を計算する。これらの損失の和を使ってパラメータ更新を行う。





$(0 - 0.5) = -0.5$

$(0 - 0.7) = -0.7$

Widrow-Hoff 学習則

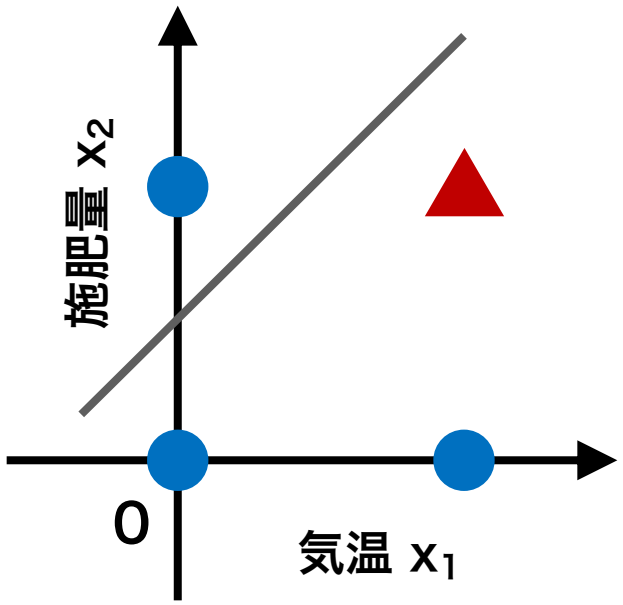
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

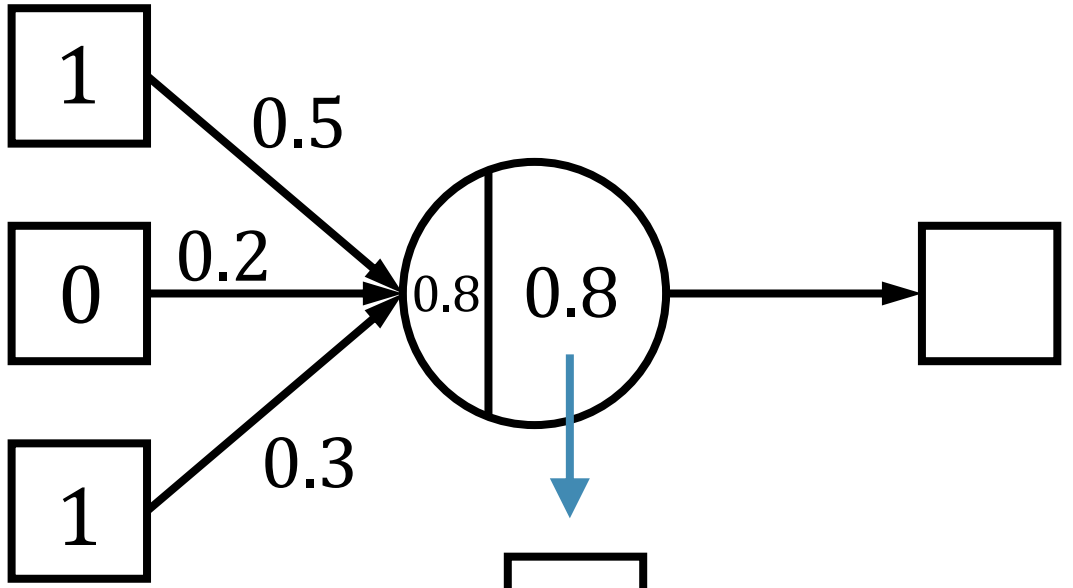
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

ADALINE ではすべての訓練データを用いて損失を計算する。これらの損失の和を使ってパラメータ更新を行う。





$(0 - 0.5) = -0.5$

$(0 - 0.7) = -0.7$

$(0 - 0.5) = -0.5$

Widrow-Hoff 学習則

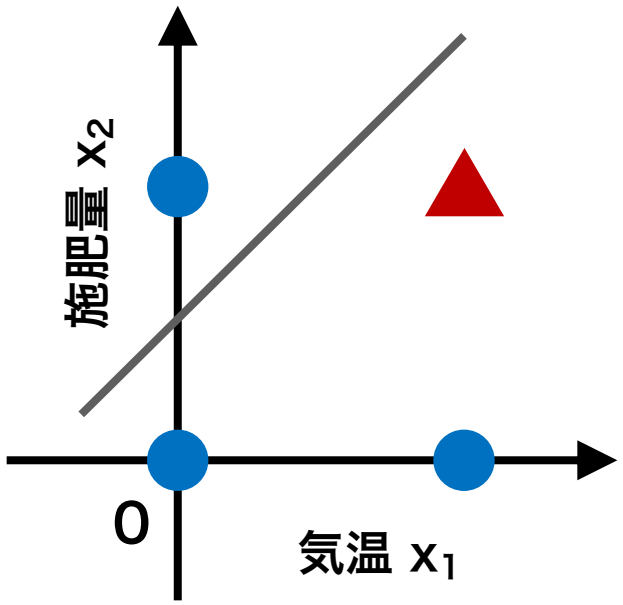
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

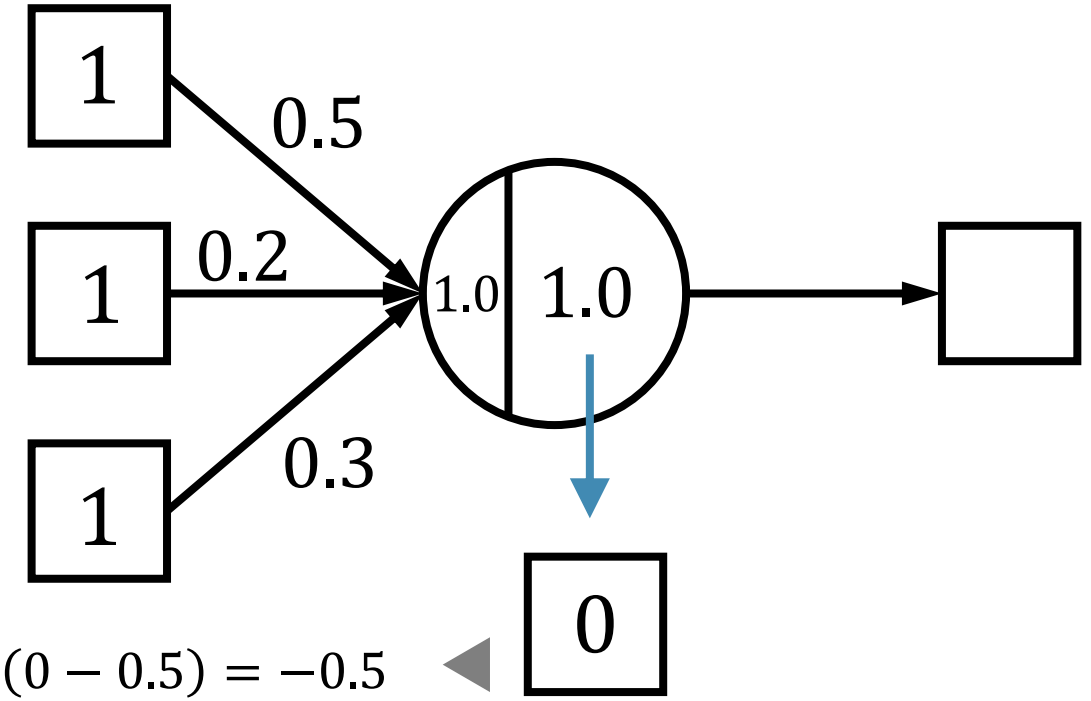
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

ADALINE ではすべての訓練データを用いて損失を計算する。これらの損失の和を使ってパラメータ更新を行う。





$(0 - 0.5) = -0.5$

$(0 - 0.7) = -0.7$

$(0 - 0.5) = -0.5$

$(1 - 1.0) = 0.0$

Widrow-Hoff 学習則

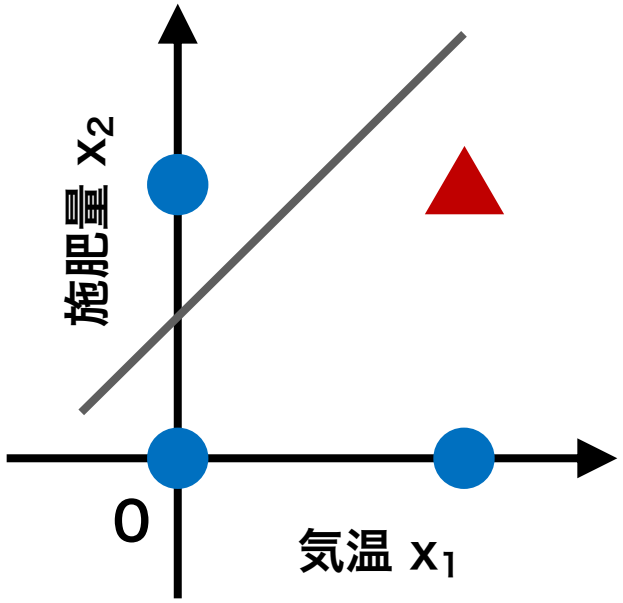
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

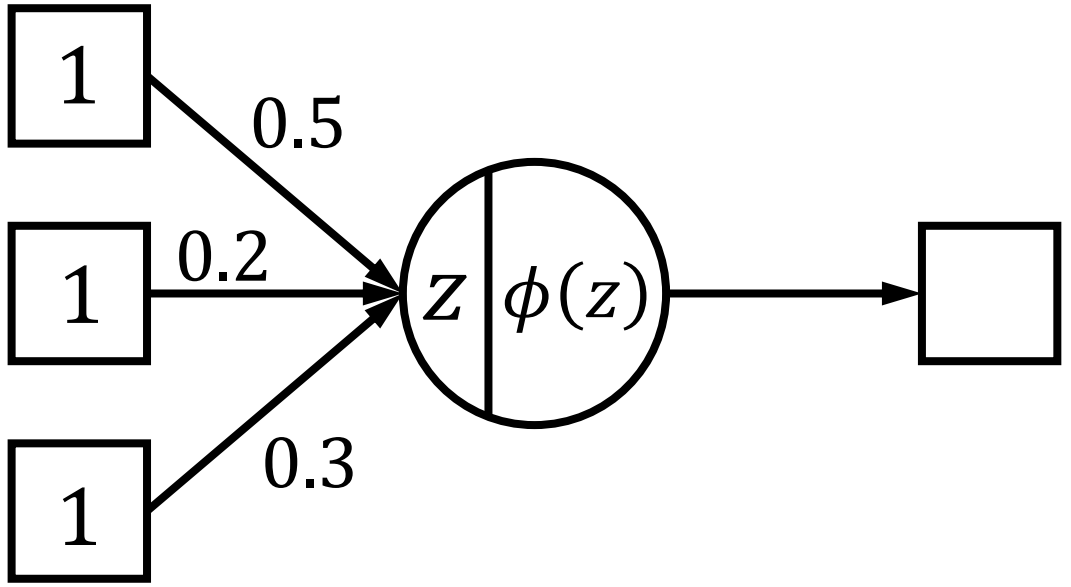
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

ADALINE ではすべての訓練データを用いて損失を計算する。これらの損失の和を使ってパラメータ更新を行う。





$$
\begin{aligned}
w_0^{(new)} &= w_0^{(old)} + \Delta w_0 \\
&= w_0^{(old)} - \eta \frac{\partial J(\boldsymbol{w})}{\partial w_0} \\
&= 0.5 - 0.1(-1.7) \\
&= 0.67
\end{aligned}
$$

$$
\begin{aligned}
(0-0.5) &= -0.5 \\
(0-0.7) &= -0.7 \\
(0-0.5) &= -0.5 \\
(1-1.0) &= 0.0
\end{aligned}
\quad \Rightarrow \quad
\begin{aligned}
-0.5 \times 1 &= -0.5 \\
-0.7 \times 1 &= -0.7 \\
-0.5 \times 1 &= -0.5 \\
0.0 \times 1 &= 0
\end{aligned}
\quad \Rightarrow \quad -1.7
$$

Widrow-Hoff 学習則

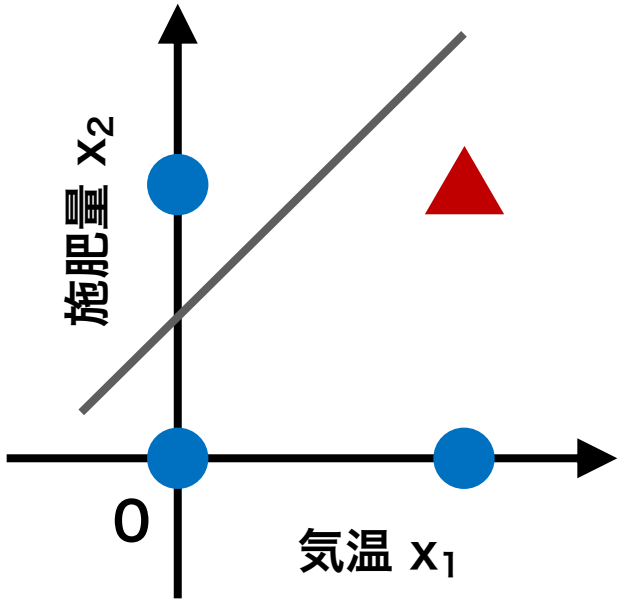
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

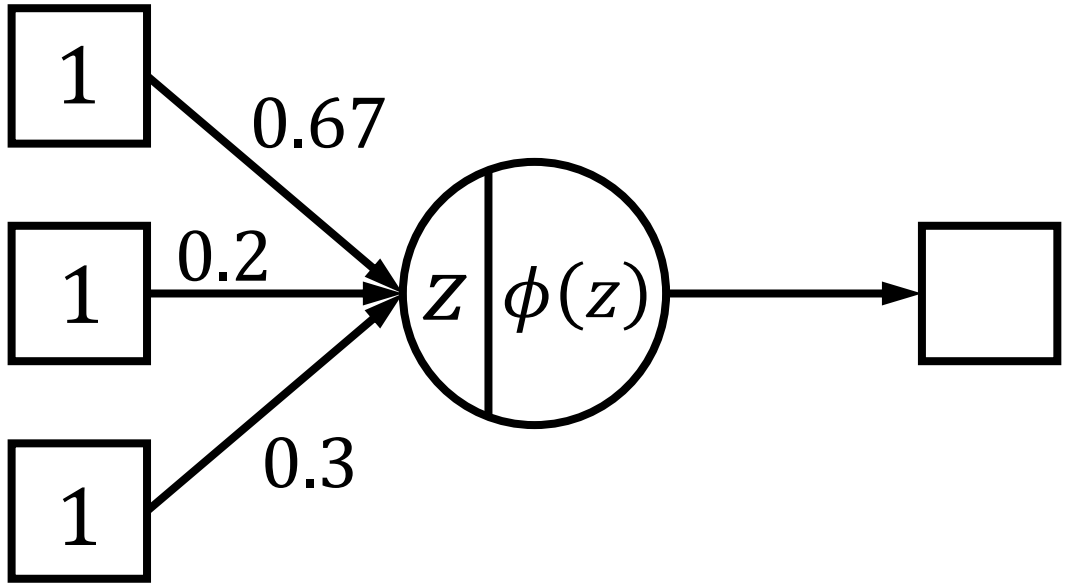
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

ADALINE ではすべての訓練データを用いて損失を計算する。これらの損失の和を使ってパラメータ更新を行う。





$$
\begin{aligned}
w_1^{(new)} &= w_1^{(old)} + \Delta w_1 \\
&= w_1^{(old)} - \eta \frac{\partial J(\boldsymbol{w})}{\partial w_1} \\
&= 0.2 - 0.1(-0.7) \\
&= 0.27
\end{aligned}
$$

$$
\begin{aligned}
(0-0.5) &= -0.5 \\
(0-0.7) &= -0.7 \\
(0-0.5) &= -0.5 \\
(1-1.0) &= 0.0
\end{aligned}
\quad\Rightarrow\quad
\begin{aligned}
-0.5\times 0 &= 0 \\
-0.7\times 1 &= -0.7 \\
-0.5\times 0 &= 0 \\
0.0\times 1 &= 0
\end{aligned}
\quad\Rightarrow\quad -0.7
$$

Widrow-Hoff 学習則

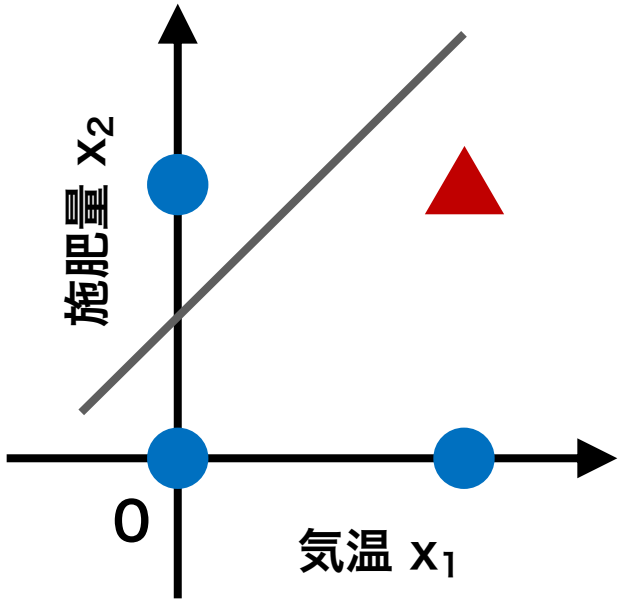
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

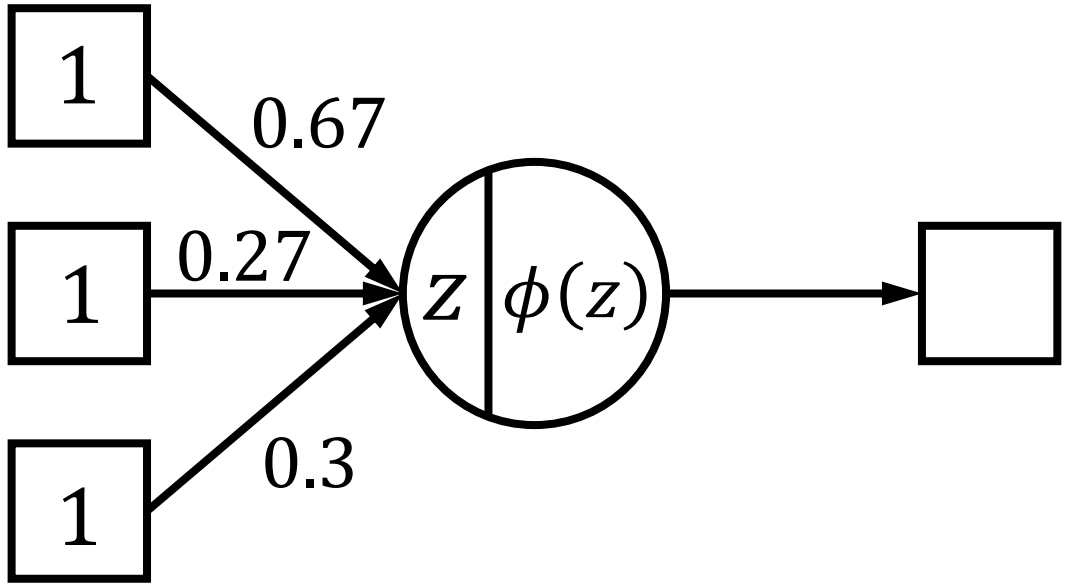
学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1

ADALINE ではすべての訓練データを用いて損失を計算する。これらの損失の和を使ってパラメータ更新を行う。





$$
\begin{aligned}
w_2^{(new)} &= w_2^{(old)} + \Delta w_2 \\
&= w_2^{(old)} - \eta \frac{\partial J(\boldsymbol{w})}{\partial w_2} \\
&= 0.3 - 0.1(-0.5) \\
&= 0.35
\end{aligned}
$$

$$
\begin{aligned}
(0-0.5) &= -0.5 \\
(0-0.7) &= -0.7 \\
(0-0.5) &= -0.5 \\
(1-1.0) &= 0.0
\end{aligned}
\quad\Rightarrow\quad
\begin{aligned}
-0.5\times 0 &= 0 \\
-0.7\times 0 &= 0 \\
-0.5\times 1 &= -0.5 \\
0.0\times 1 &= 0
\end{aligned}
\quad\Rightarrow\quad -0.5
$$

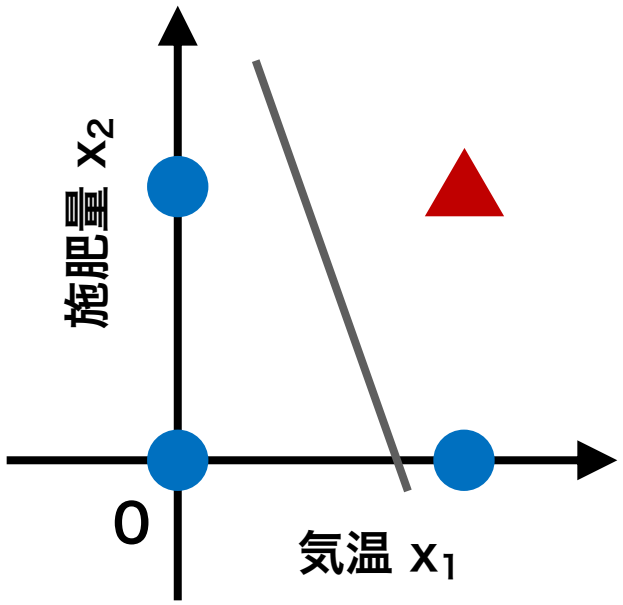
Widrow-Hoff 学習則

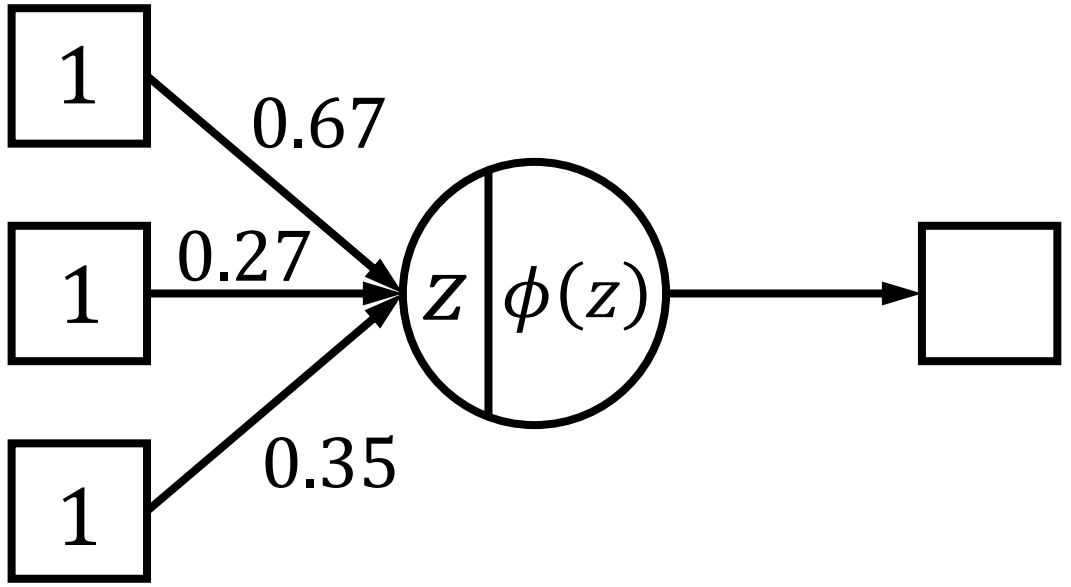
※ Widrow-Hoff 学習則を説明するためのイメージである。実際の計算はもう少し複雑である。

学習データ

気温・施肥量を使って開花予測を行うために、フィールド調査を行い次のようなデータを得た。

気温	施肥量	開花
0	0	0
1	0	0
0	1	0
1	1	1





Widrow-Hoff 学習則

ADALINE の学習において、すべての訓練データに対して、$\phi(z)$ の値とラベルの差の 2 乗の和を損失として定義している。ただし、ϕ は恒等関数であり、$\phi(z) = z = w^t x$ である。最適なモデルは損失を最小にする。そのため、損失を最小にするパラメーター w の値を計算すればよい。

ここで、損失を w の関数とみなし、損失関数を最小にする w を求める。このとき、損失関数は、w についての二次関数であり、下に凸である。この関数の最小値を調べるためには、微分係数に着目する方法がある。

$$loss = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - \phi\left(z^{(i)}\right)\right)^2$$

$$= \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - z^{(i)}\right)^2$$

$$= \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - w^t x^{(i)}\right)^2$$

※ パラメーター w は $w = (X^tX)^{-1}X^ty$ により計算した方が正確である。しかし、(X^tX) は常に逆行列が存在する保証がないため、解がもとまらない場合もある。そのため、データが大量にあるとき、微分係数を利用した方法が一般的に用いられる。

Widrow-Hoff 学習則

$$\boldsymbol{w}^{(new)} = \boldsymbol{w}^{(old)} + \Delta \boldsymbol{w}$$

$$\Delta \boldsymbol{w} = -\eta \nabla loss(\boldsymbol{w})$$

$$\nabla loss(\boldsymbol{w}) = \frac{\partial loss(\boldsymbol{w})}{\partial \boldsymbol{w}}$$

$loss(w) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - w^t x^{(i)}\right)^2$ より、

損失関数 loss を w で偏微分すると、

$$\frac{\partial loss(w)}{\partial w} = \sum_{i}^{n}\left(y^{(i)} - w^t x^{(i)}\right)x^{(i)}$$

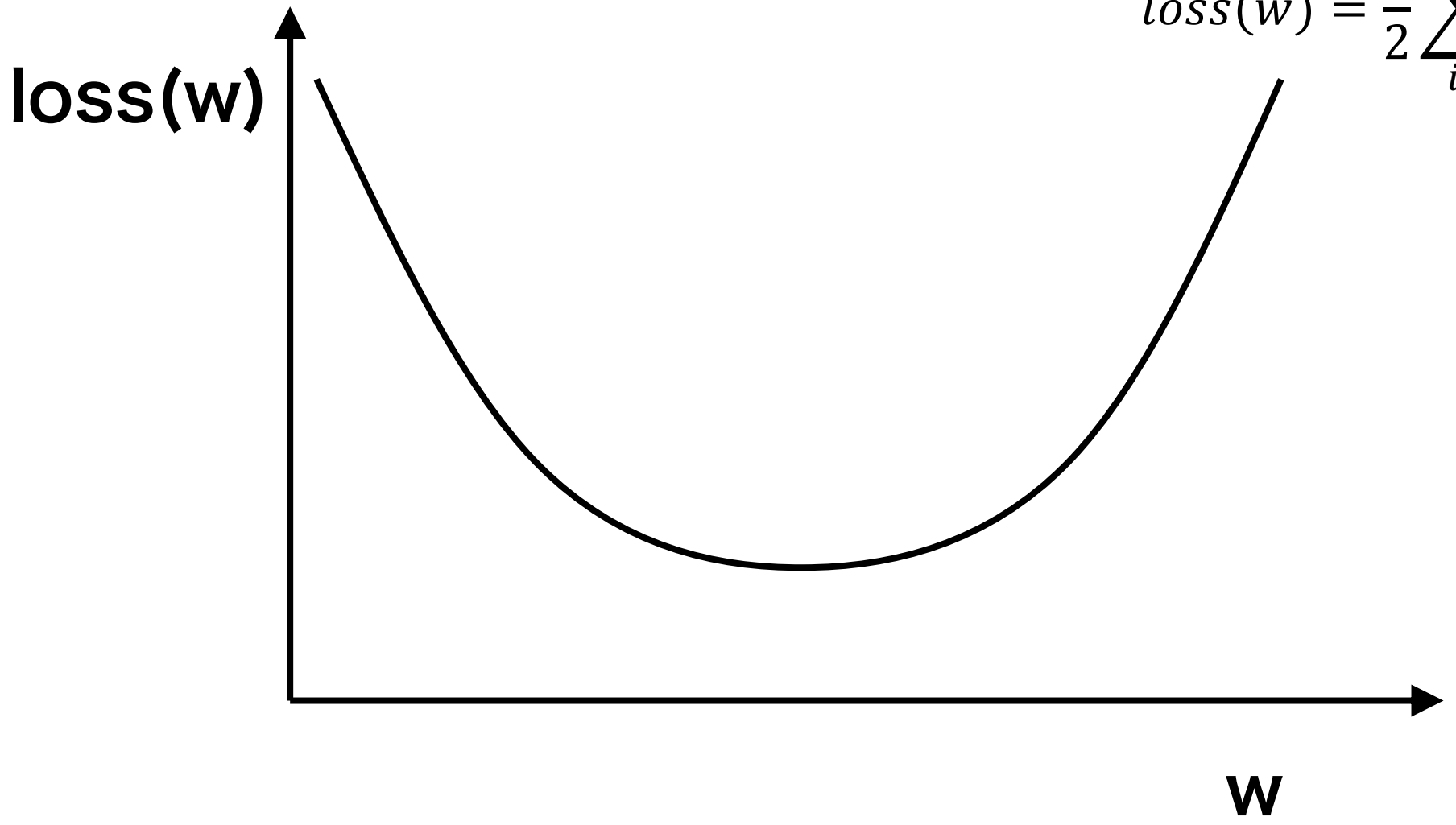
が得られる。よって w_j は次の式に基づいて更新される。

$$w_j^{(new)} = w_j^{(old)} - \eta \frac{\partial loss(\boldsymbol{w})}{\partial w_j}$$

$$= w_j^{(old)} - \eta \sum_{i}^{n}\left(y^{(i)} - w^t x^{(i)}\right)x_j^{(i)}$$

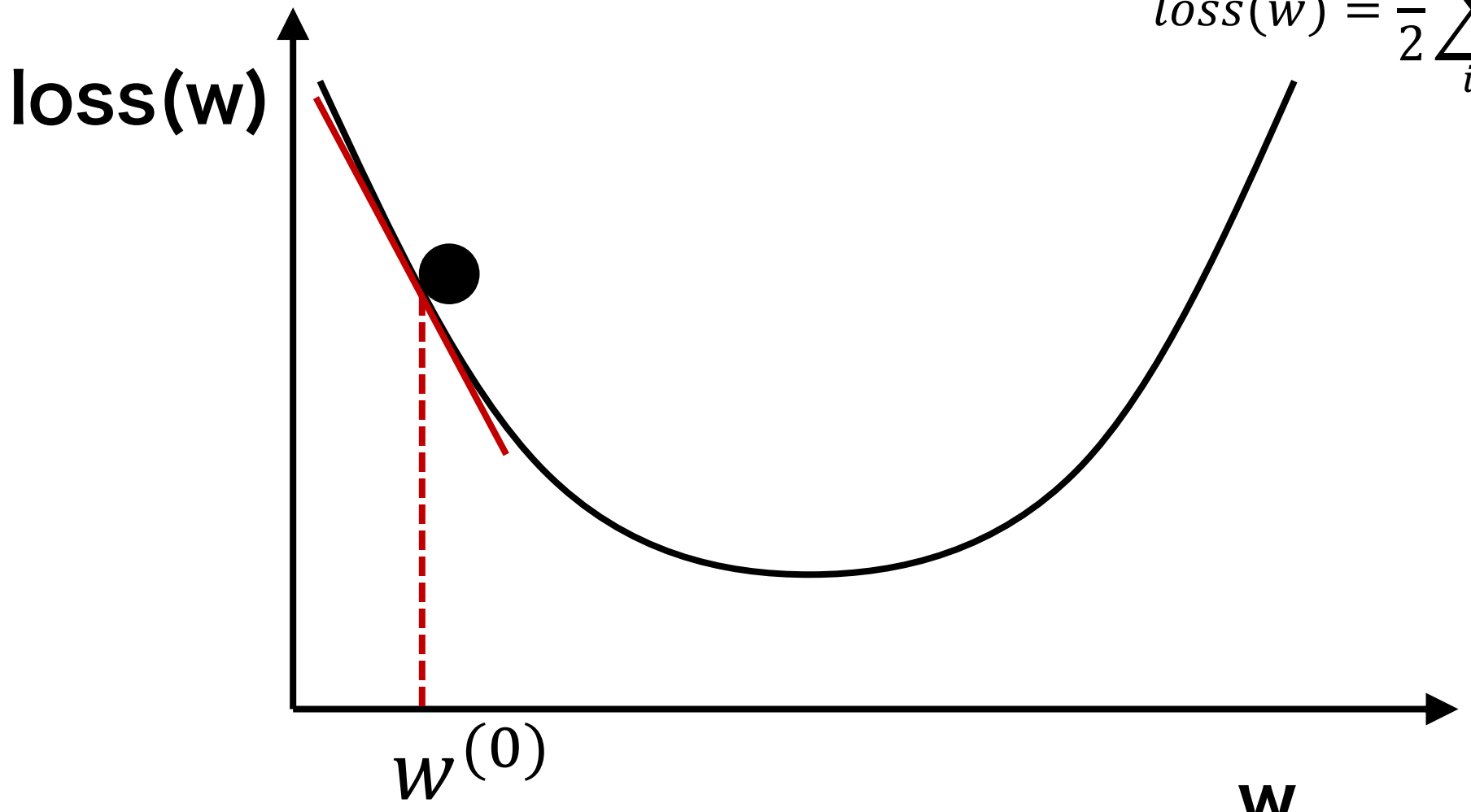
バッチ勾配降下法

$$loss(w) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - w^t x\right)^2$$

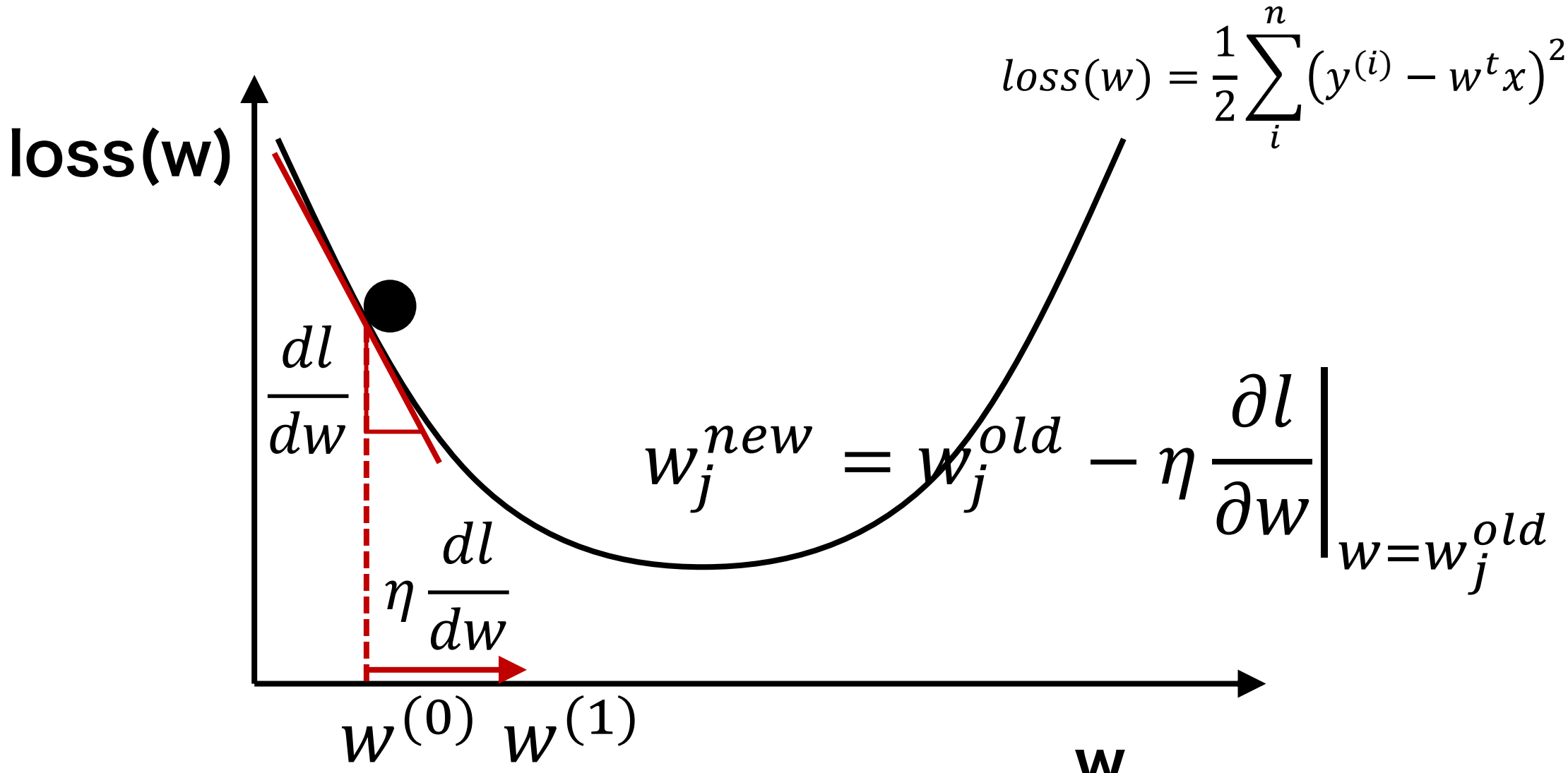


バッチ勾配降下法

$$loss(w) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - w^t x\right)^2$$

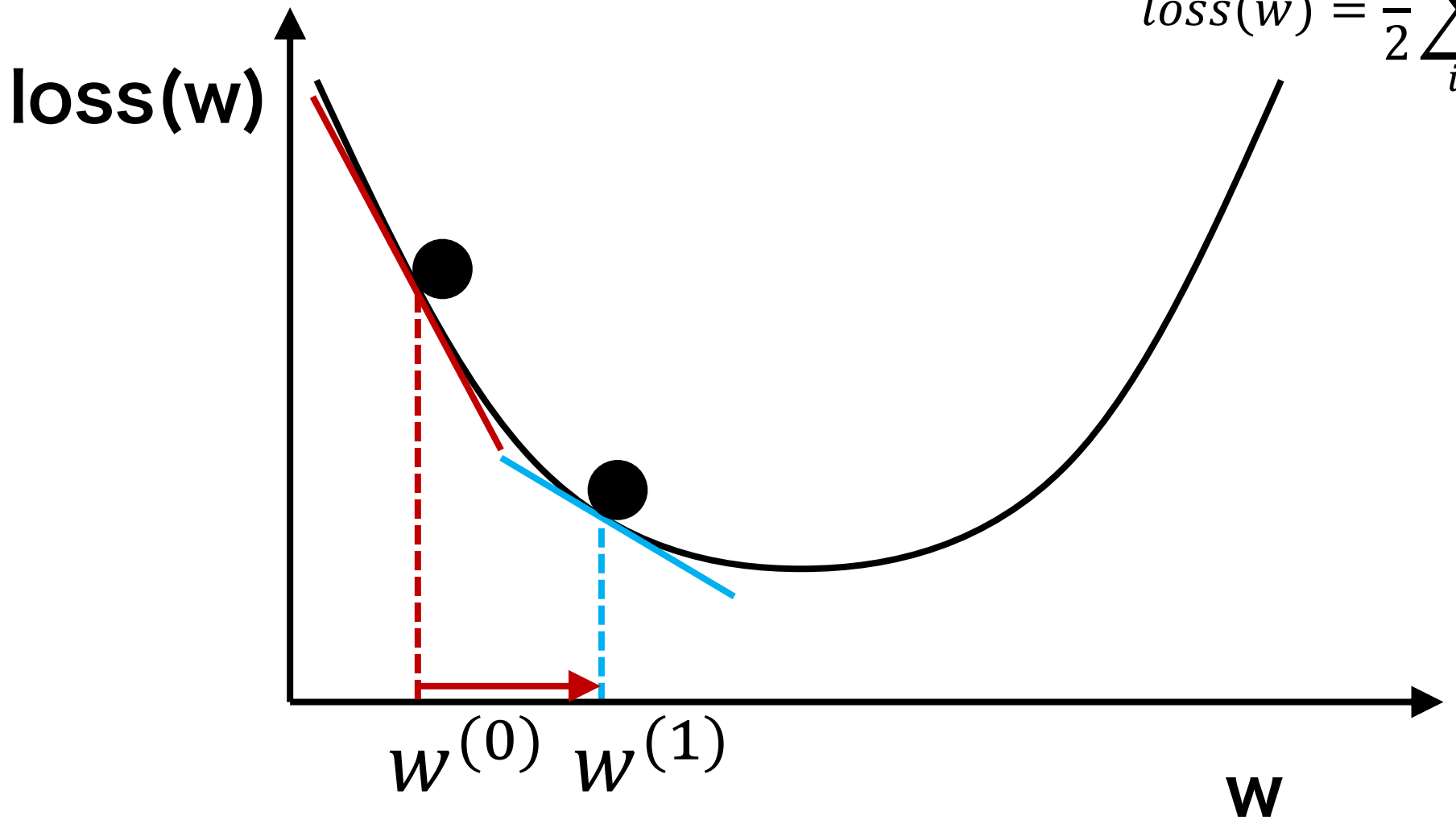


バッチ勾配降下法

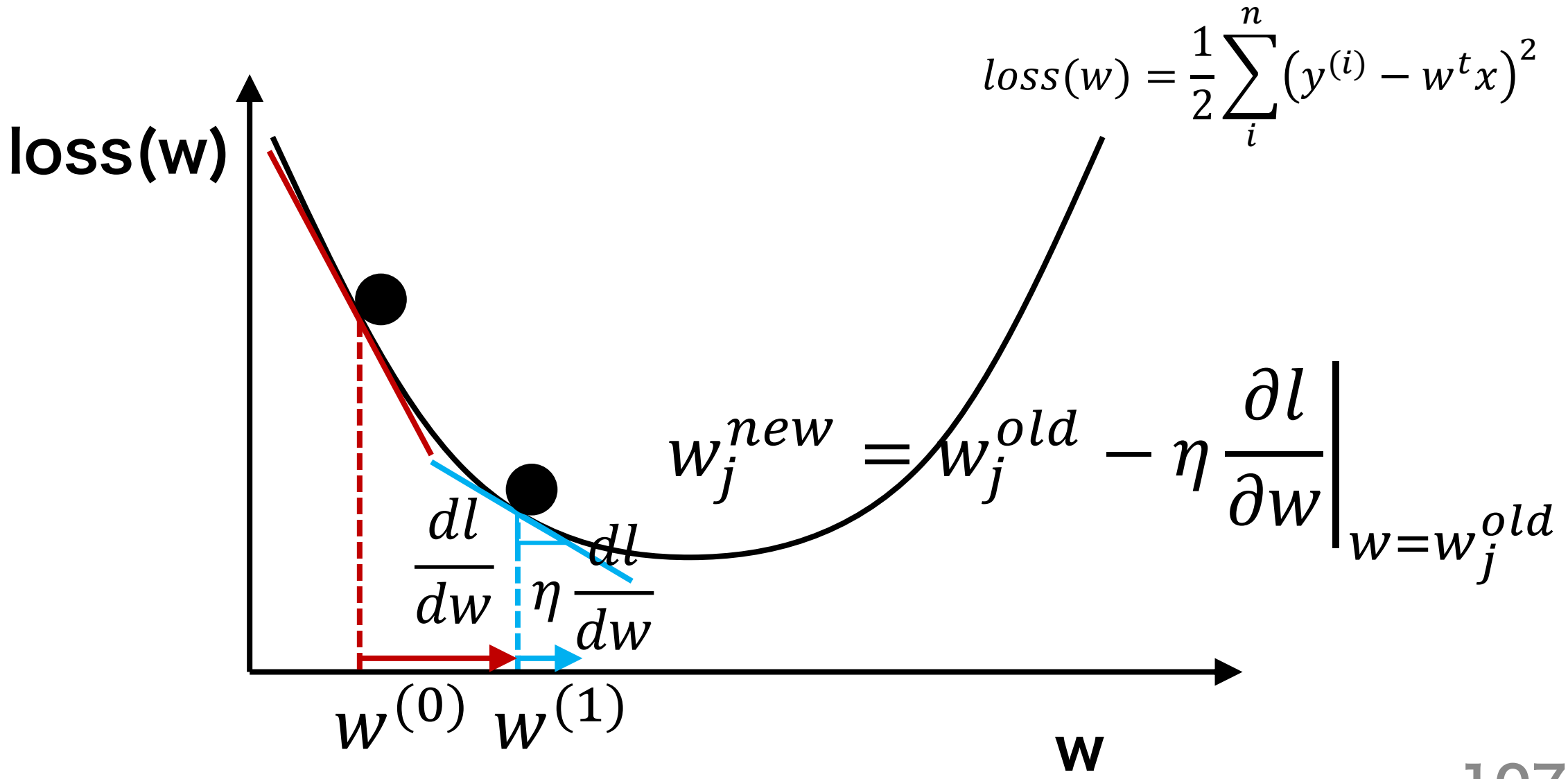

loss(w)
$loss(w) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - w^t x\right)^2$
$\frac{dl}{dw}$
$w_j^{new} = w_j^{old} - \eta \left.\frac{\partial l}{\partial w}\right|_{w=w_j^{old}}$
$\eta \frac{dl}{dw}$
$w^{(0)}$
$w^{(1)}$
w

バッチ勾配降下法

$$loss(w) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - w^t x\right)^2$$

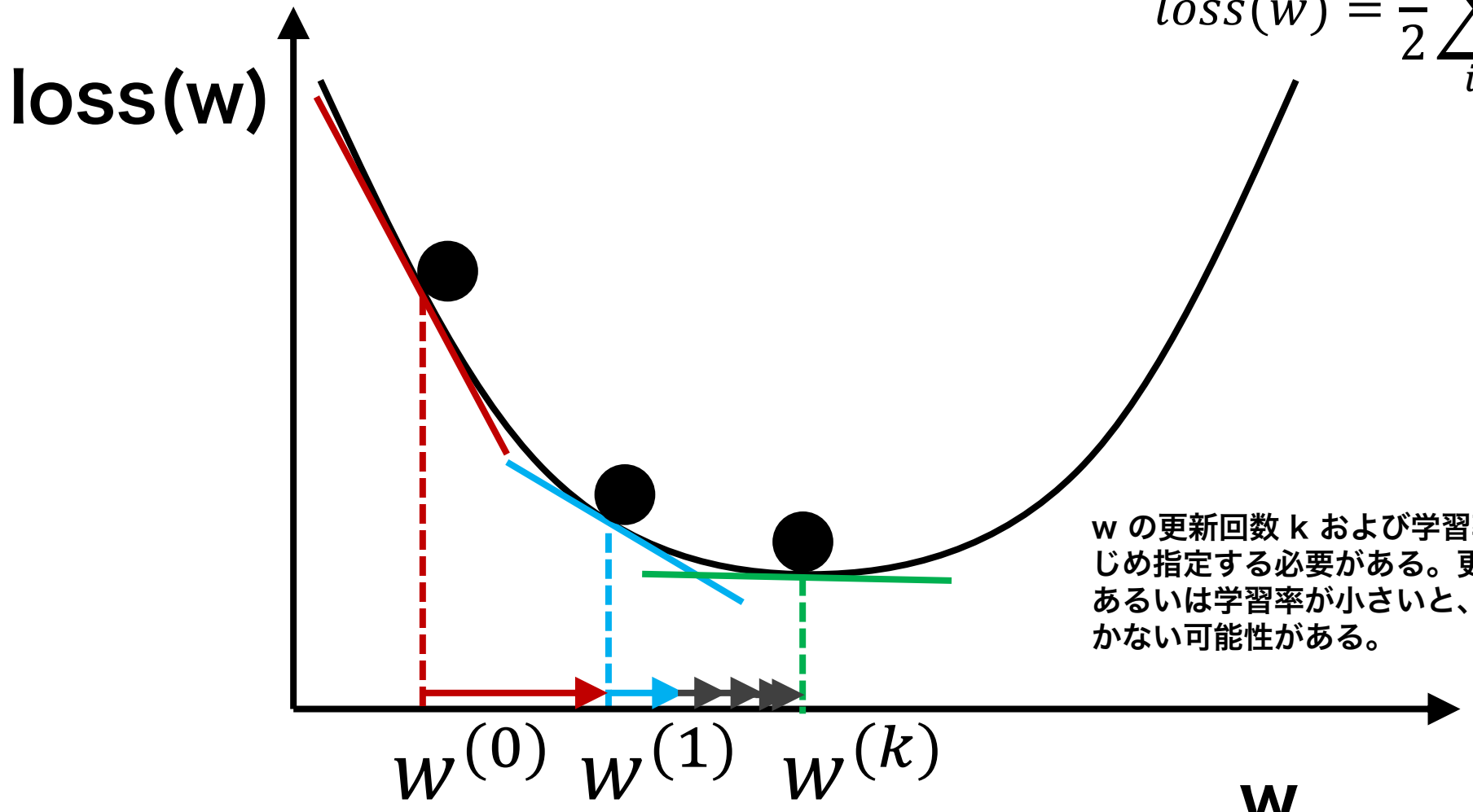


バッチ勾配降下法


loss(w)
$loss(w) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - w^t x\right)^2$
$w_j^{new} = w_j^{old} - \eta \left.\frac{\partial l}{\partial w}\right|_{w=w_j^{old}}$
$\frac{dl}{dw}$
$\eta \frac{dl}{dw}$
$w^{(0)}$
$w^{(1)}$
w

バッチ勾配降下法

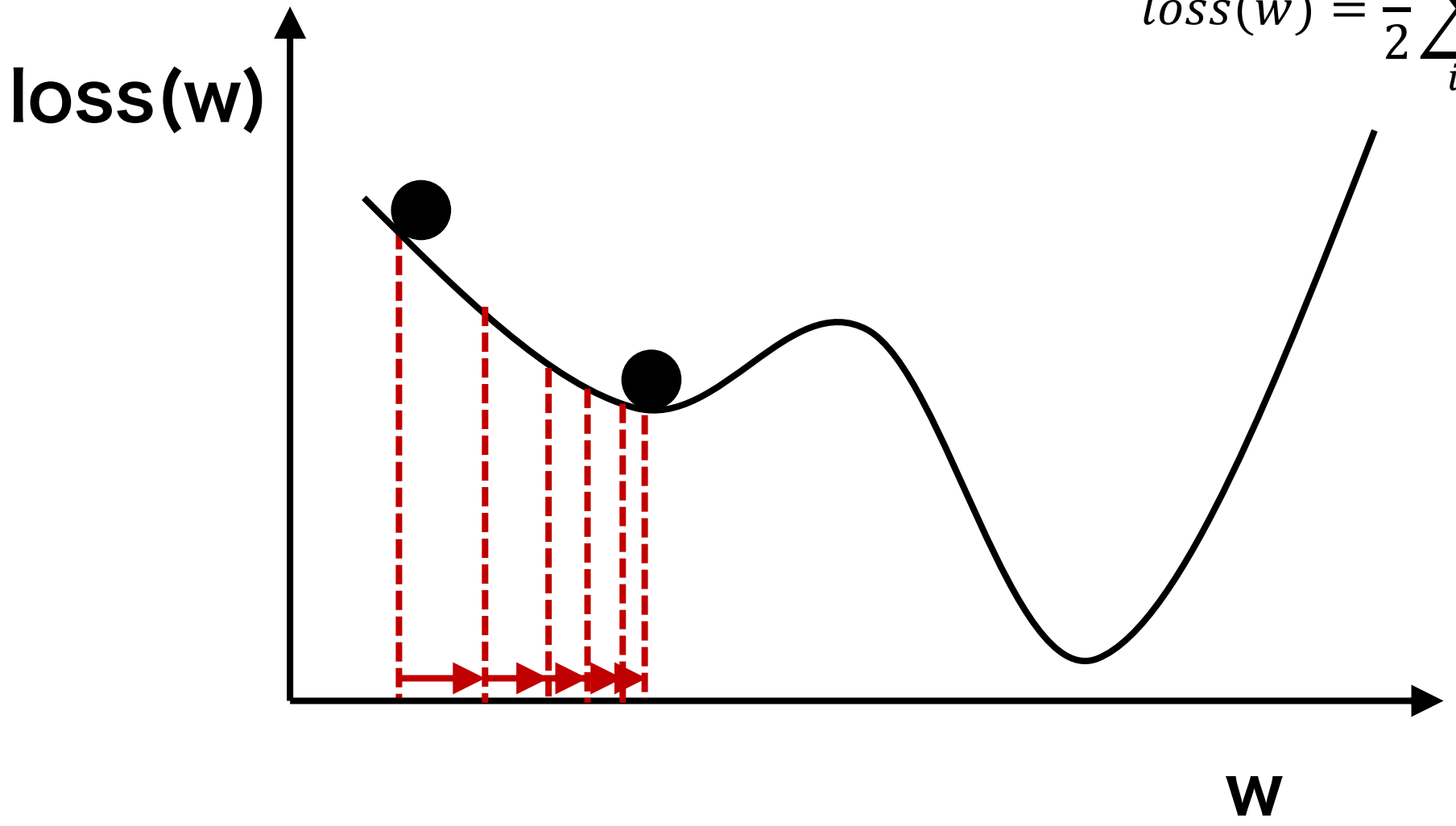
$$loss(w) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - w^t x\right)^2$$



w の更新回数 k および学習率 η は、あらかじめ指定する必要がある。更新回数が少ないあるいは学習率が小さいと、最小値に辿り着かない可能性がある。

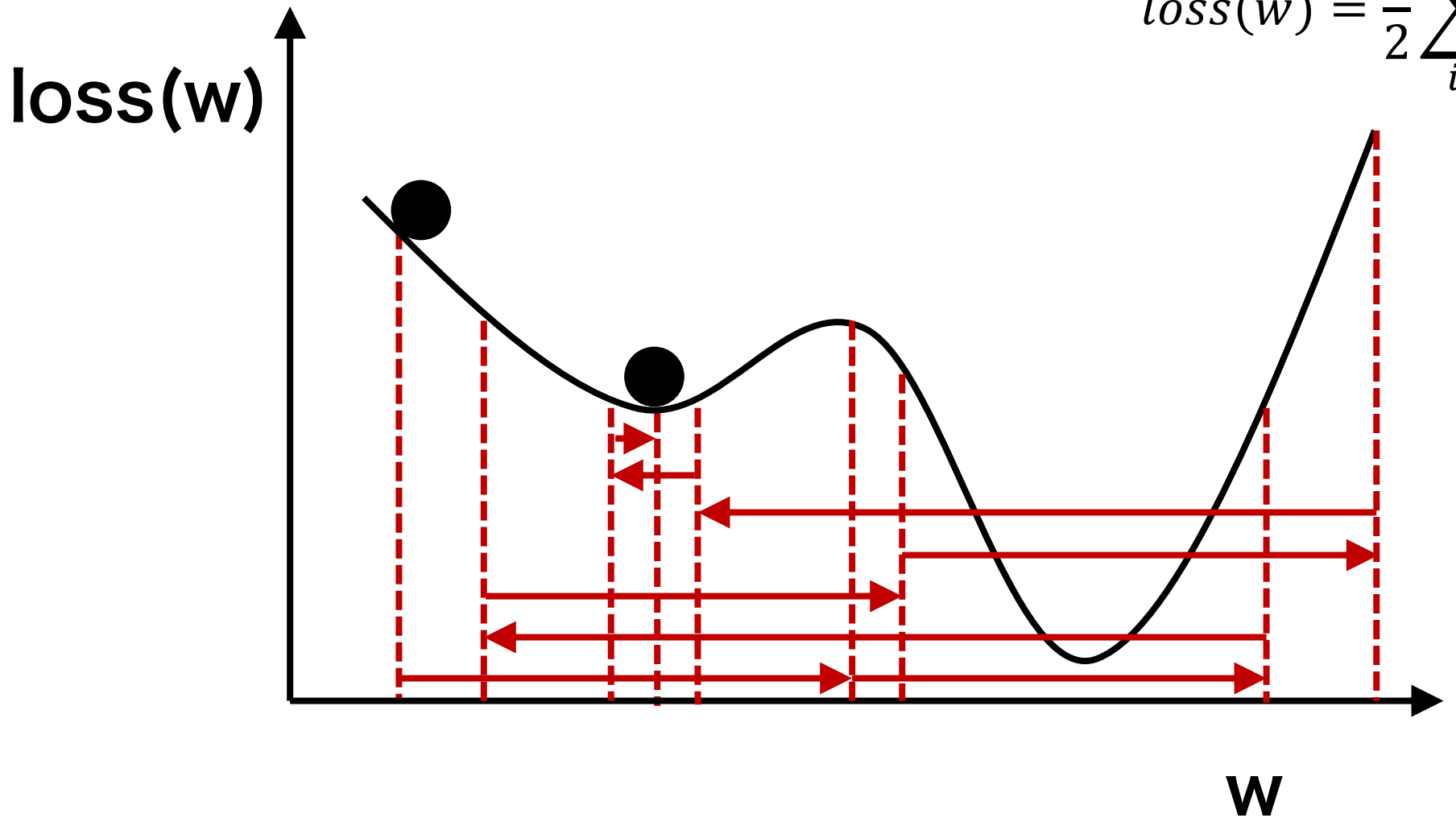
バッチ勾配降下法

$$loss(w) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - w^t x\right)^2$$

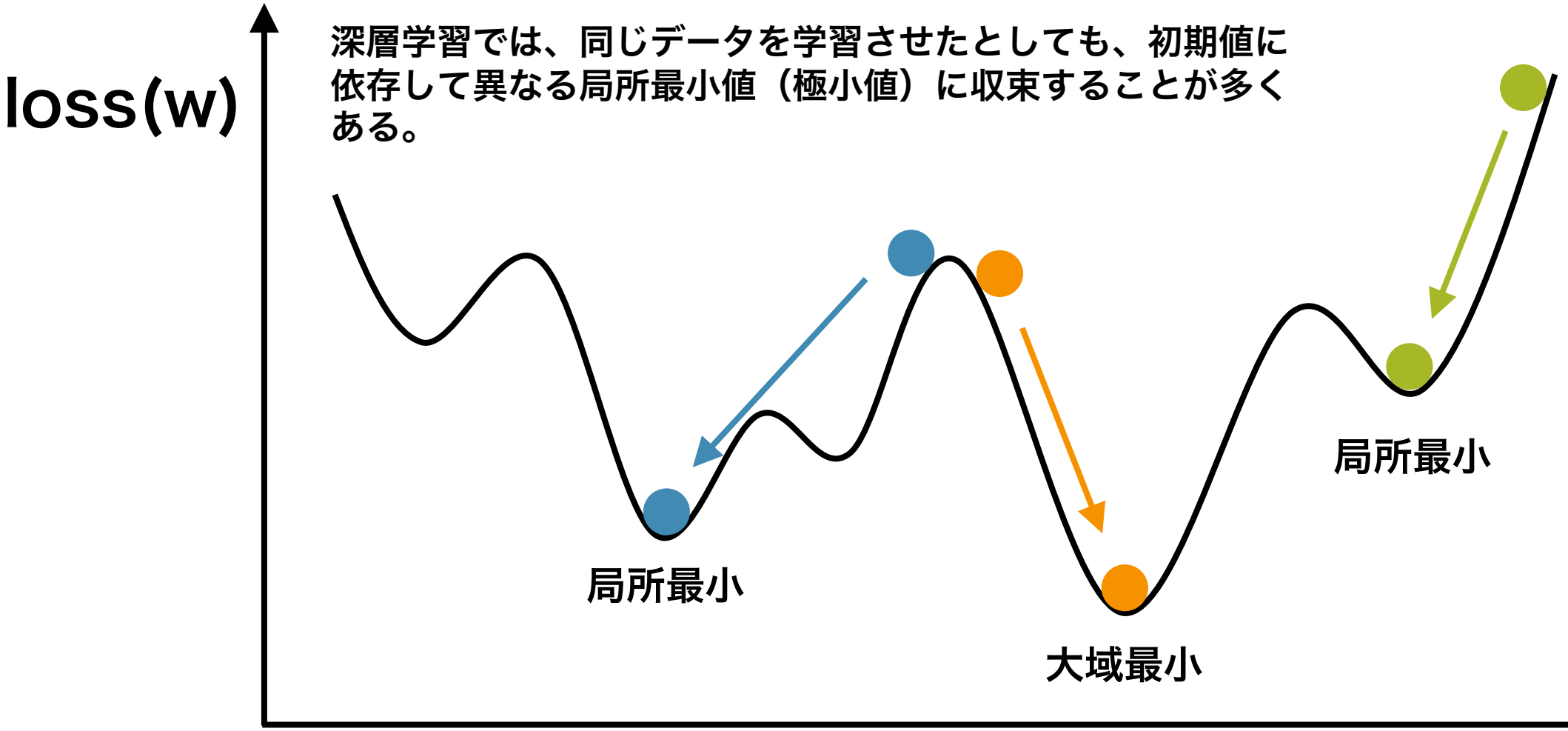


バッチ勾配降下法

$$loss(w) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - w^t x\right)^2$$



バッチ勾配降下法


loss(w)
深層学習では、同じデータを学習させたとしても、初期値に依存して異なる局所最小値（極小値）に収束することが多くある。
局所最小
大域最小
局所最小
w

勾配降下法

バッチ勾配降下法

訓練データセットの全体を使用して損失を計算し、パラメータの更新を行うアルゴリズムである。そのため、すべての訓練データから計算された損失の大局的最小値を正確に求めることができる。しかし、データ量が多い場合、より多くの計算量を必要とする。また、データが増えたときに、全ての訓練データに対して、最初から学習する必要がある。

確率勾配降下法

訓練データセットの全体からランダムに 1 サンプルだけを抽出し、勾配降下法によりパラメータの更新を行うアルゴリズムである。パラメータ更新が頻繁に行われるため、収束が早い。しかし、損失は 1 サンプルだけに基づいて計算されるため、局所的最小値に陥りやすい。

ミニバッチ勾配降下法

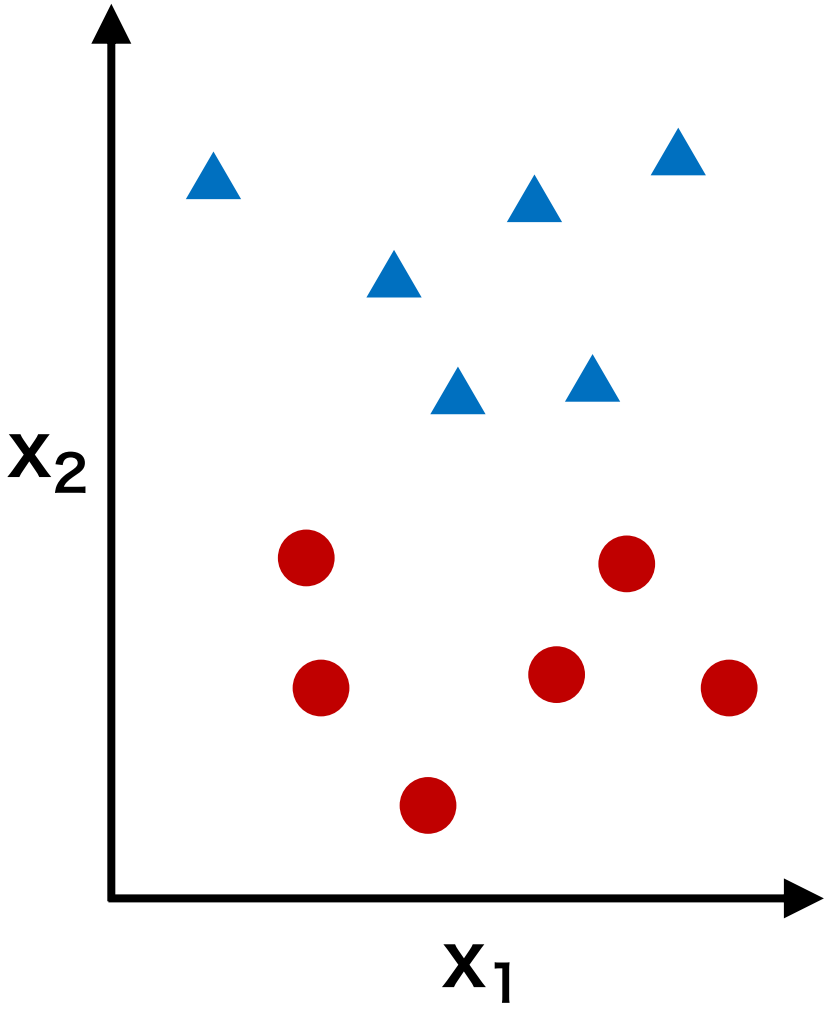
訓練データセットの全体からランダムに b サンプルを抽出してサブセット（ミニバッチ）とする。このミニバッチに対してバッチ勾配降下法を適用する。ミニバッチ勾配降下法は、バッチ勾配降下法と確率勾配降下法の折衷案となるアルゴリズムであり、両者の間の性質を持つ。

単層ニューラルネットワーク

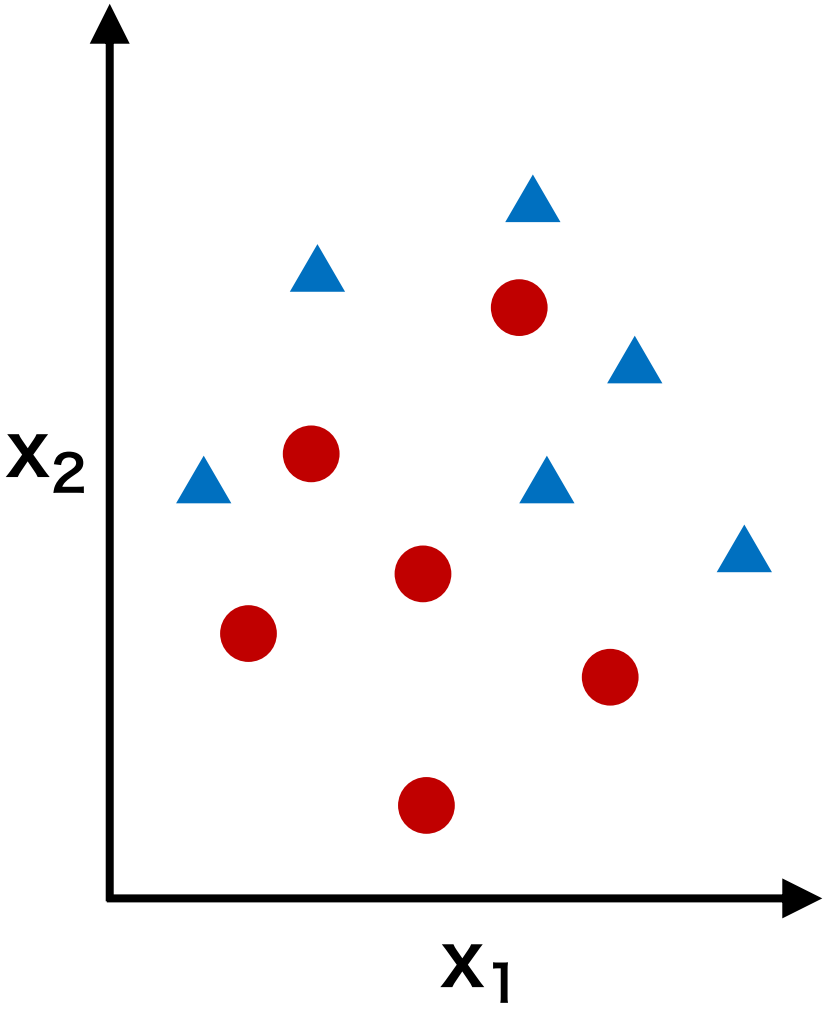
- パーセプトロン
- ADALINE
- ニューラルネットワーク

線形分離

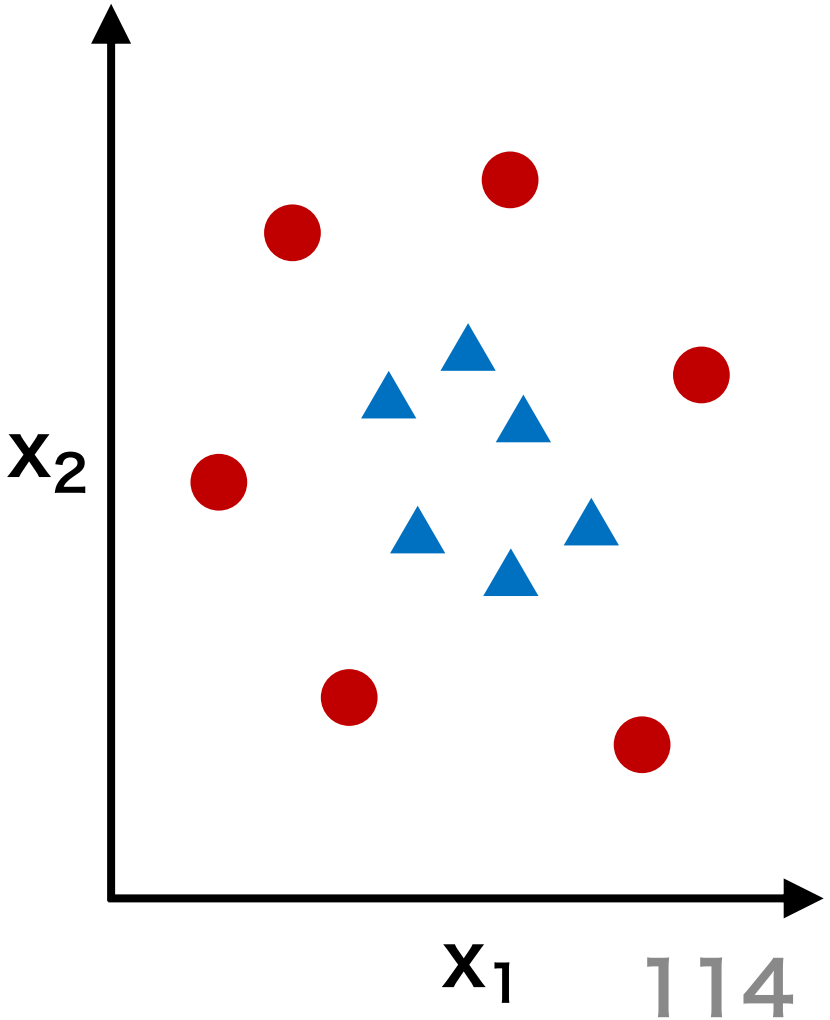
線形分離可能



線形分離不可

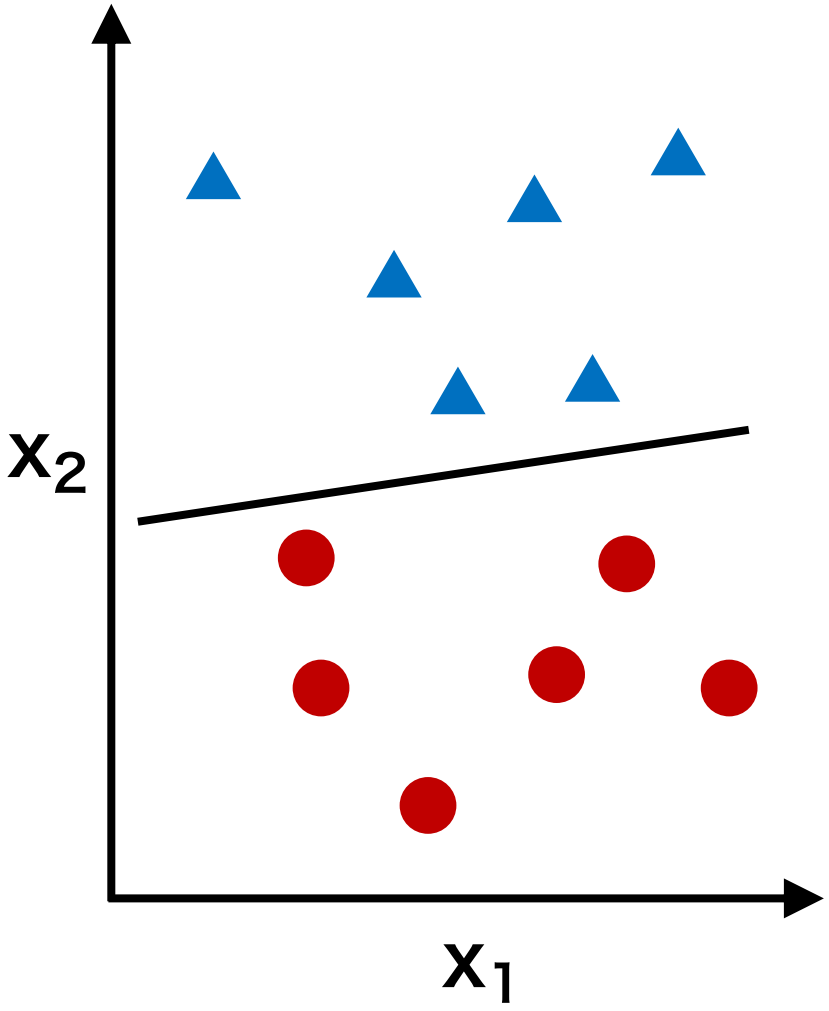


線形分離不可

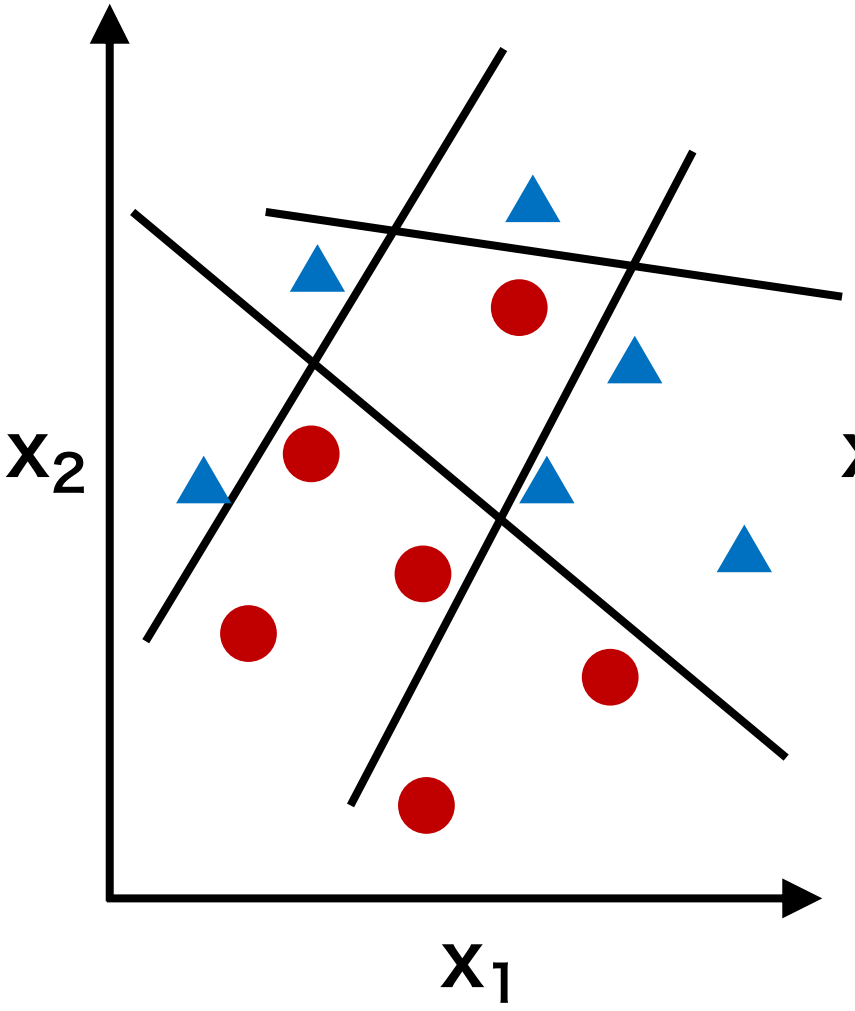


線形分離

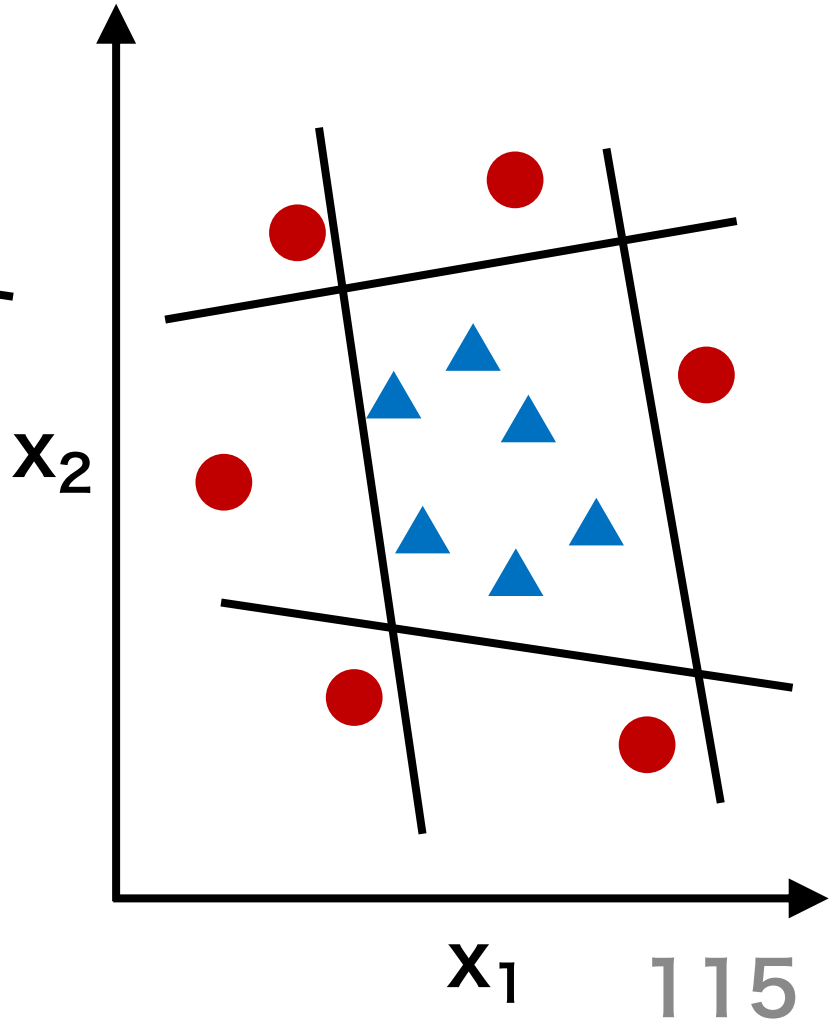
線形分離可能



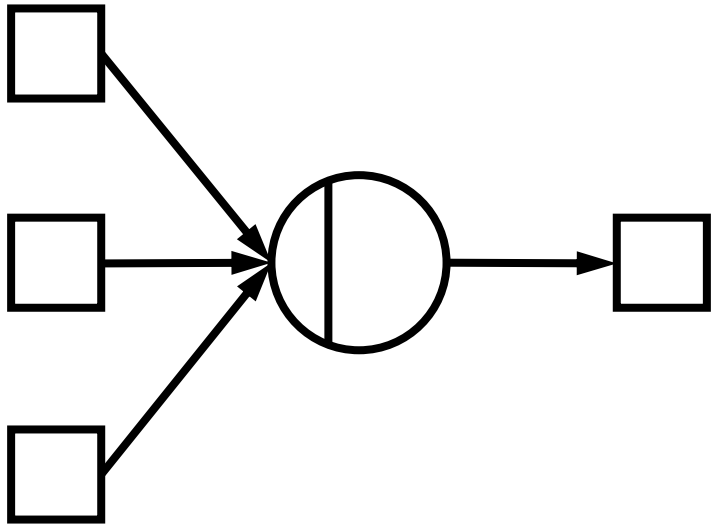
線形分離不可

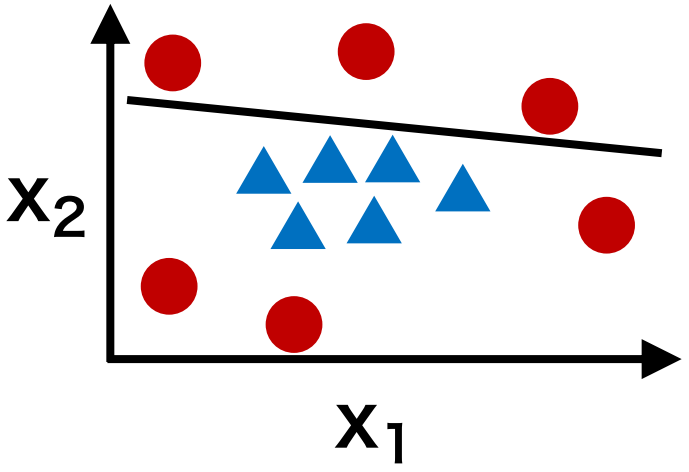


線形分離不可

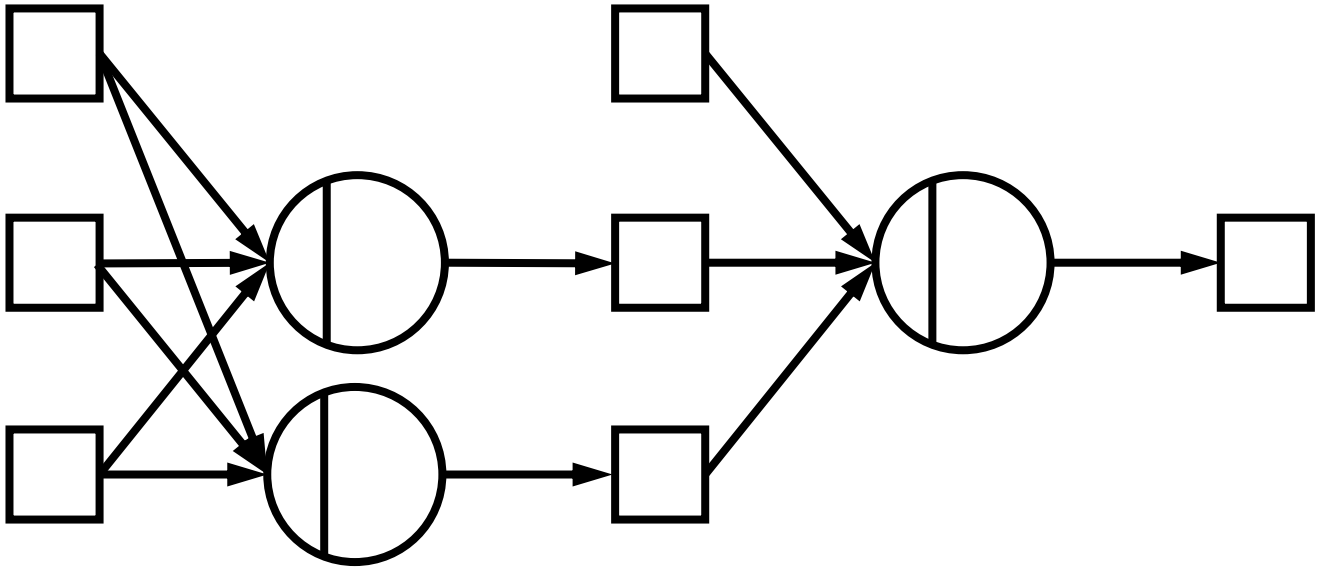


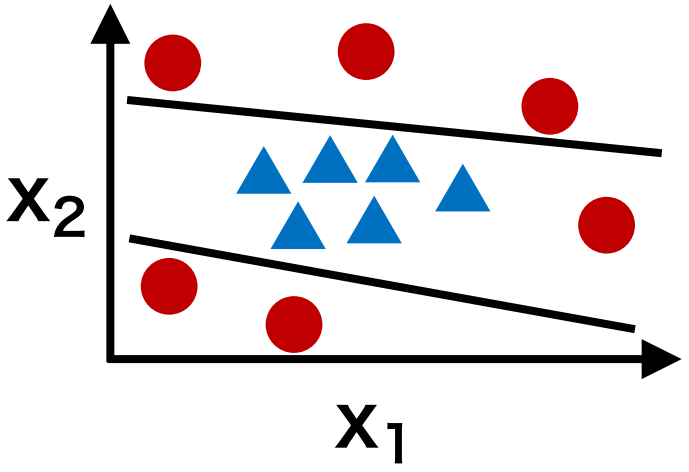
ニューラルネットワーク



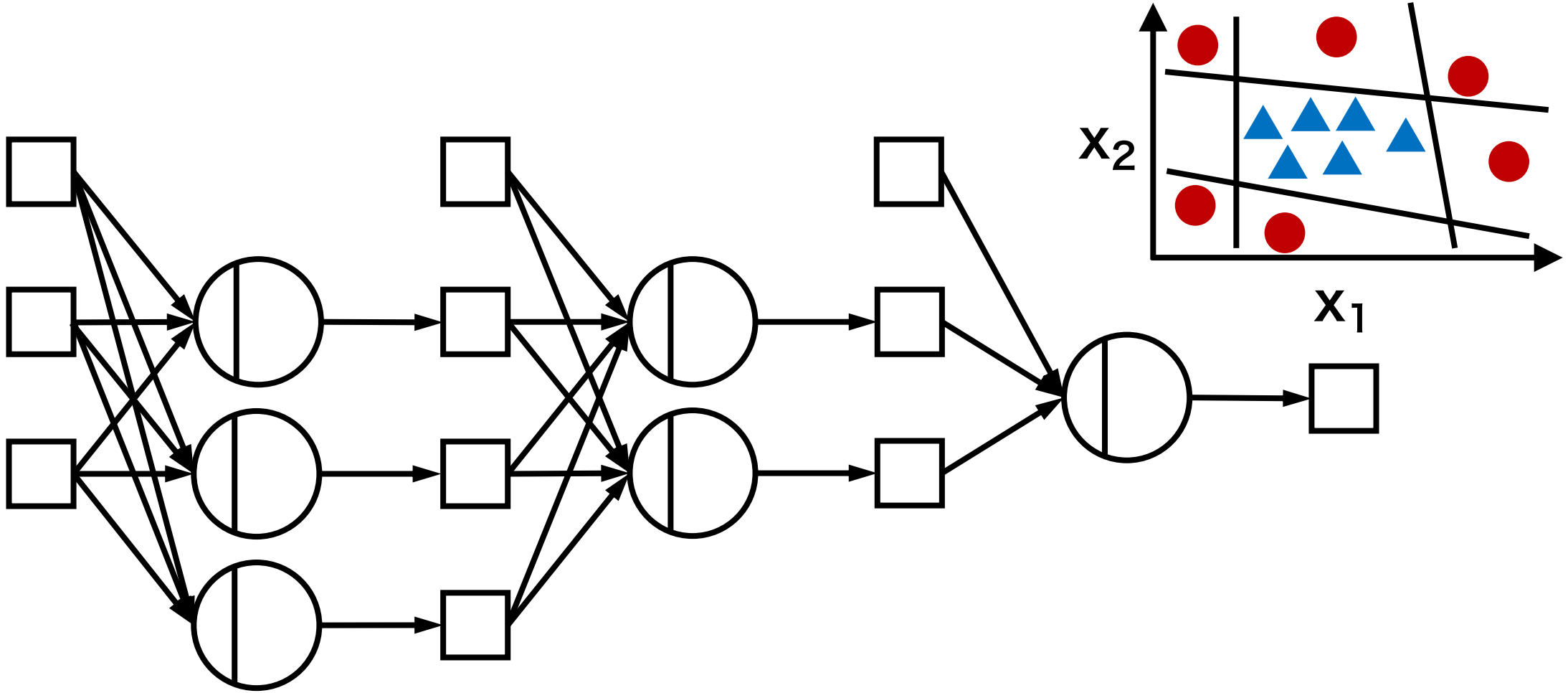


ニューラルネットワーク

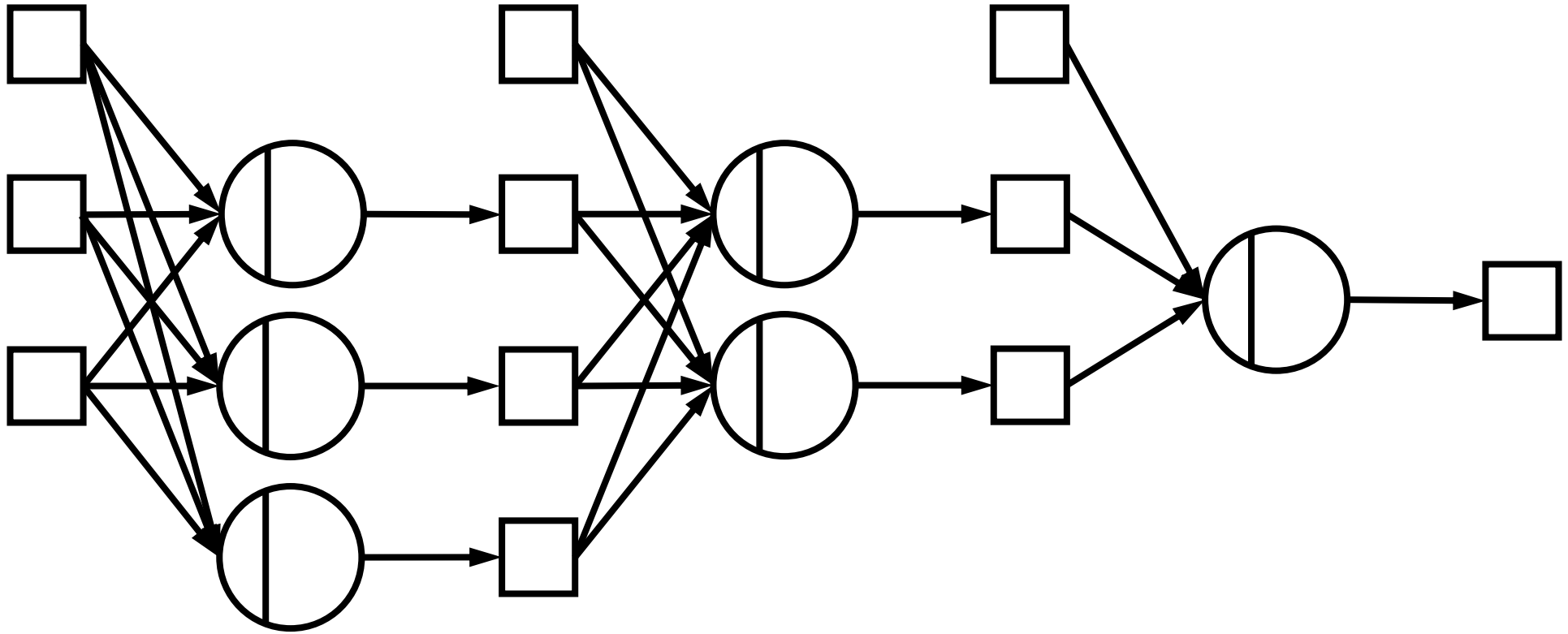




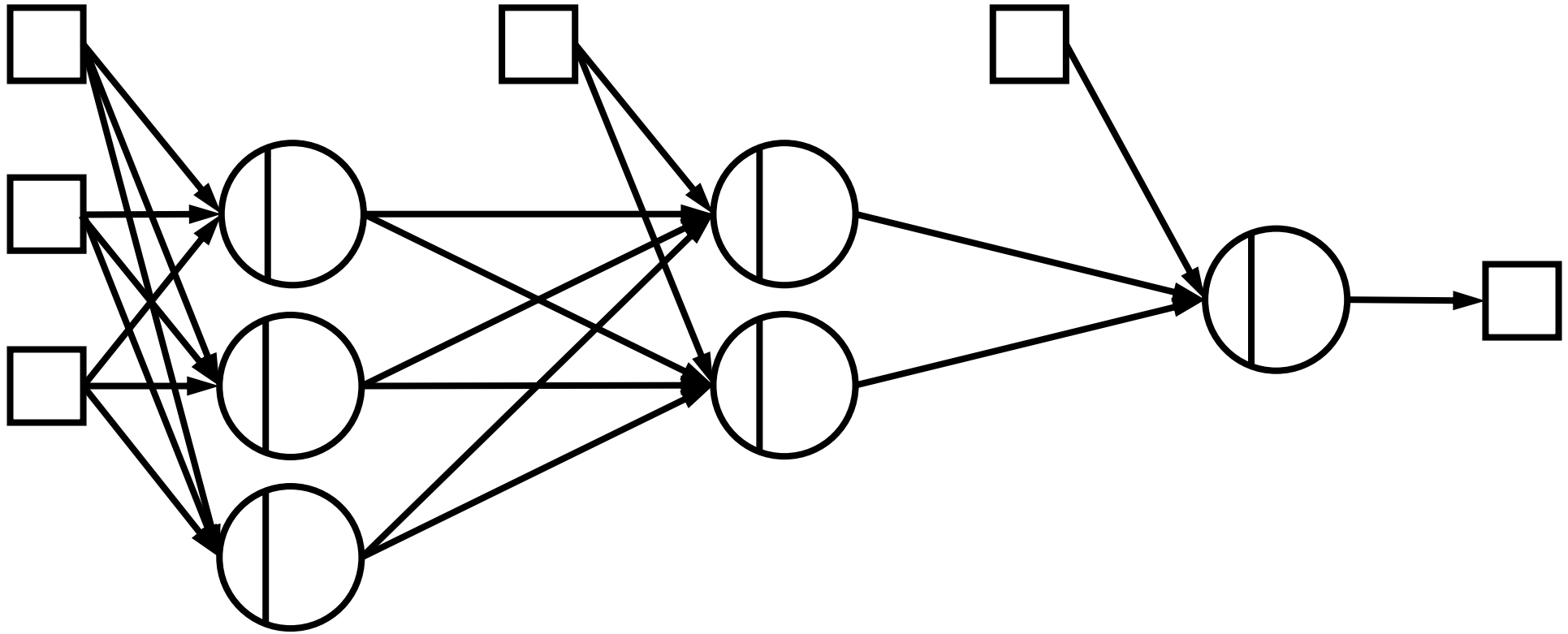
ニューラルネットワーク



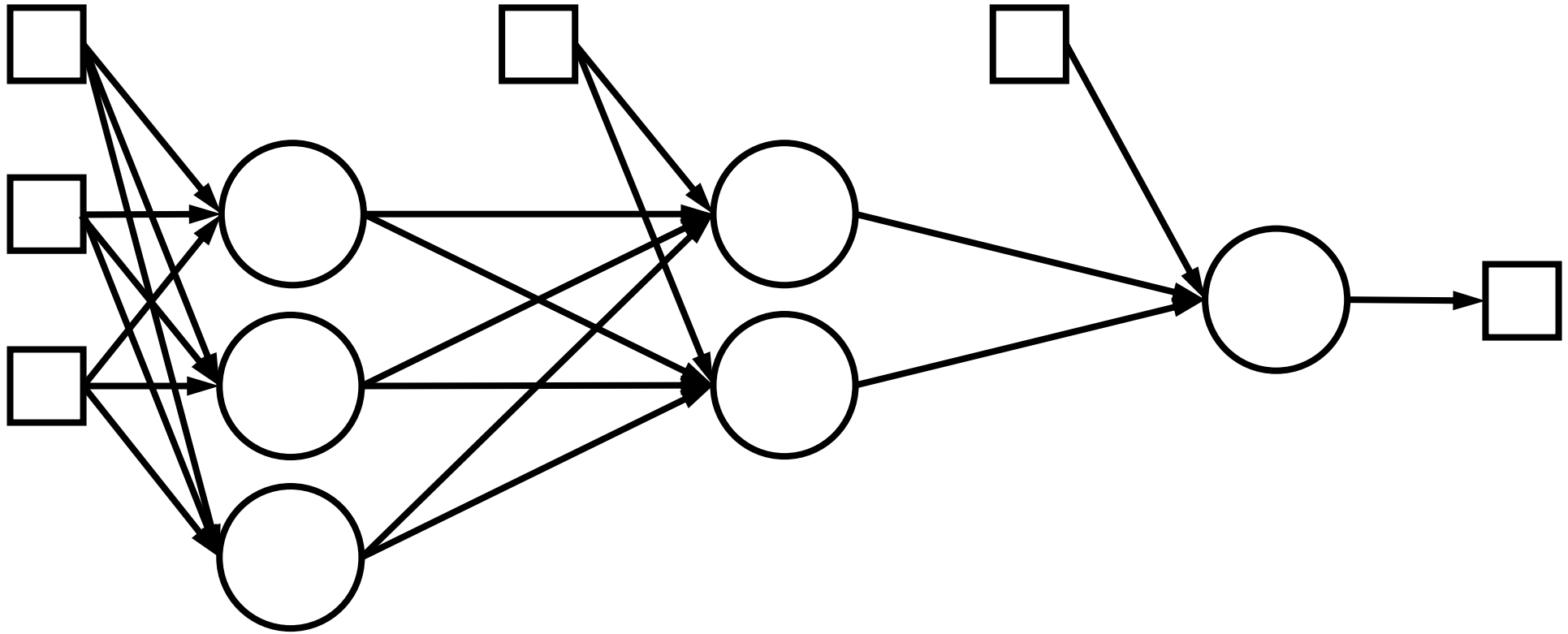
ニューラルネットワーク



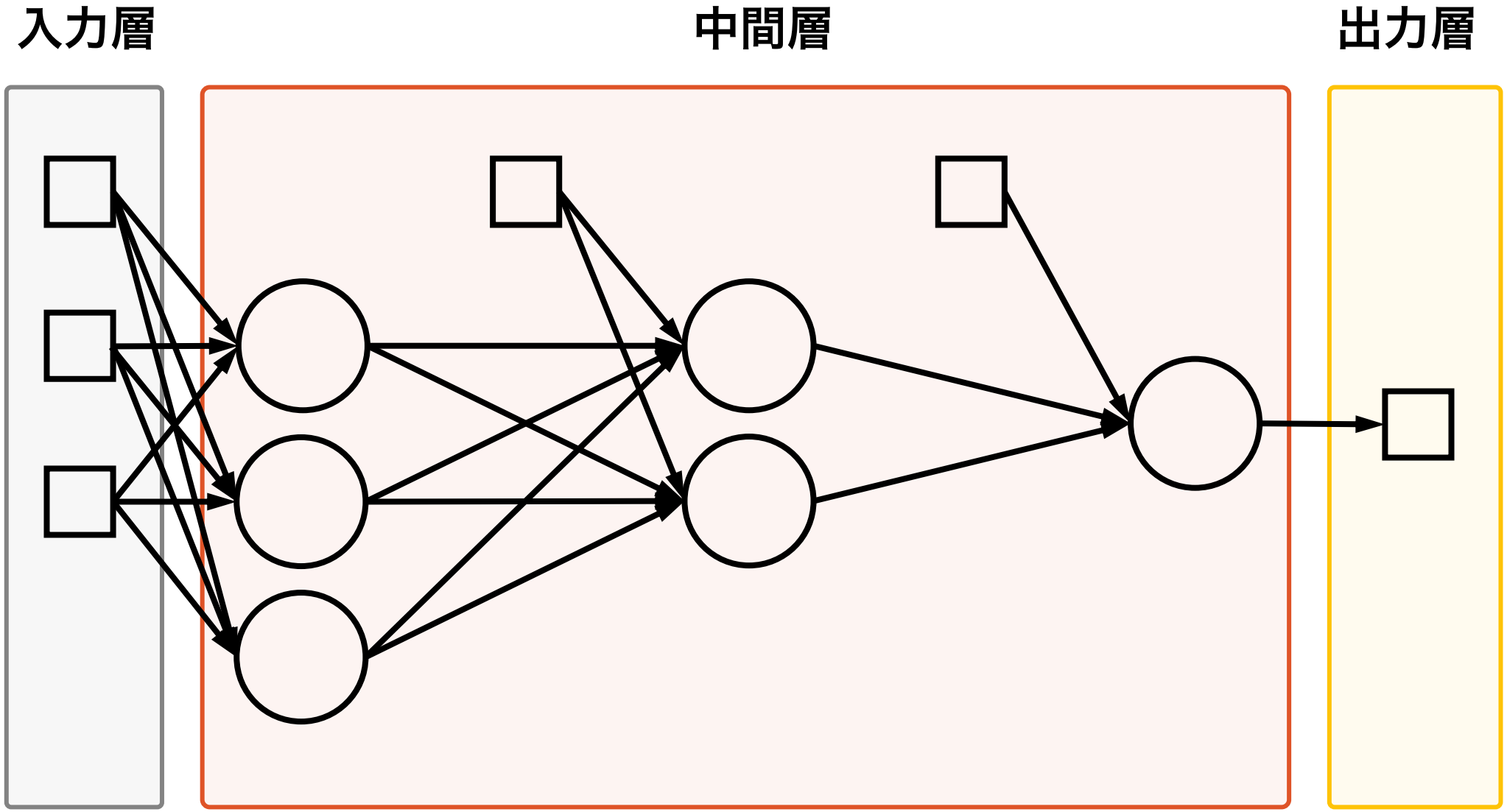
ニューラルネットワーク



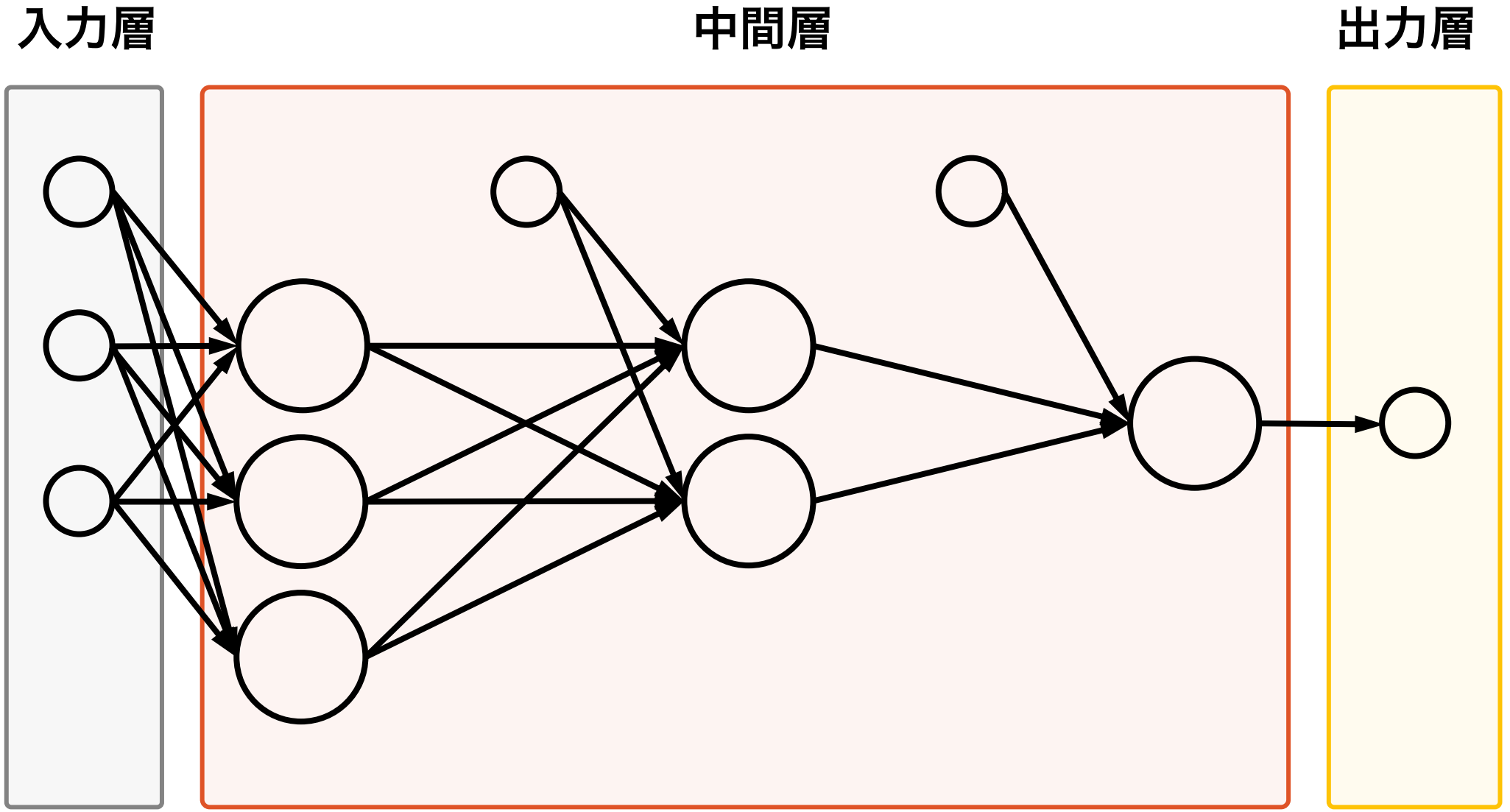
ニューラルネットワーク



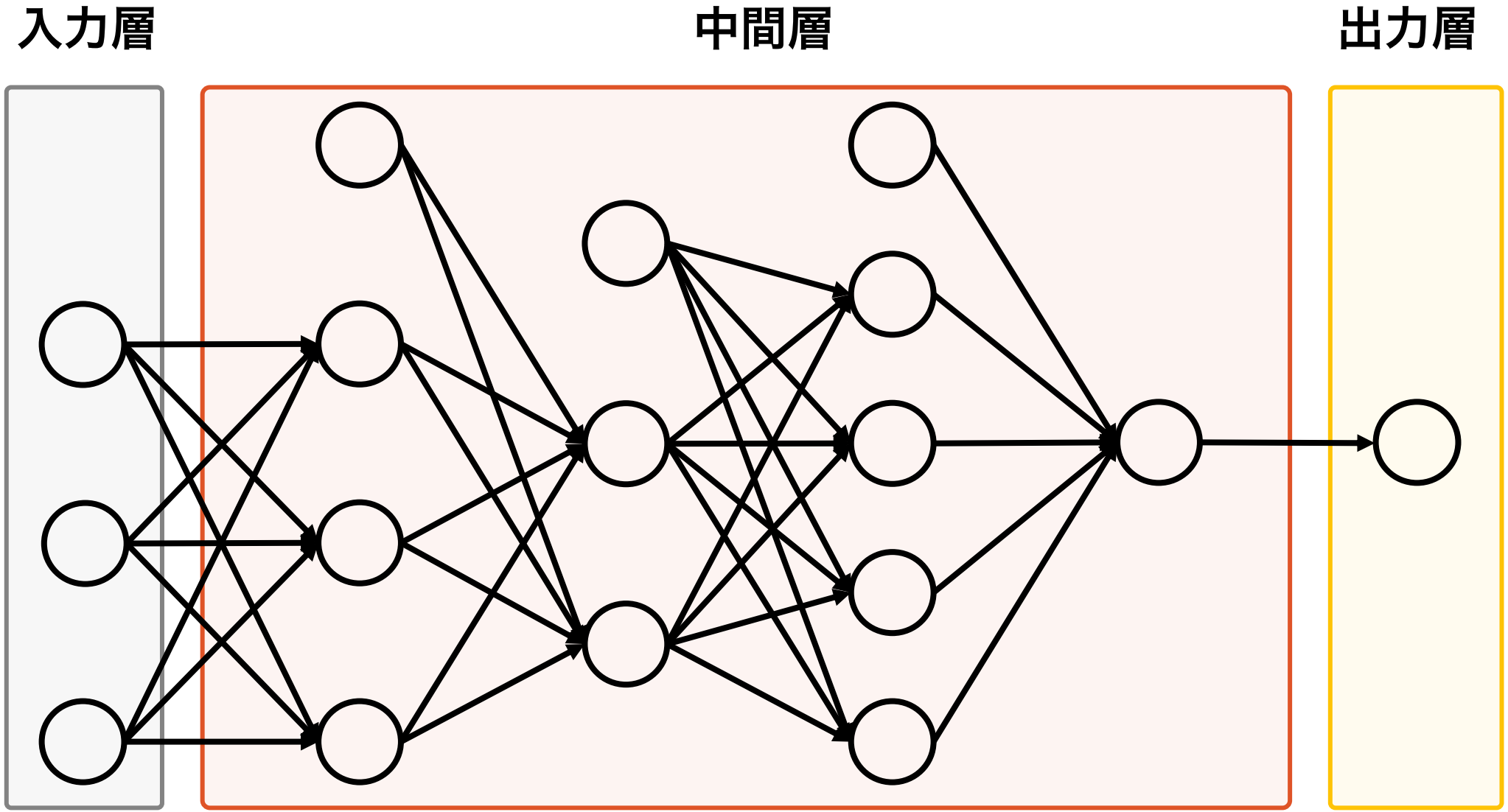
ニューラルネットワーク



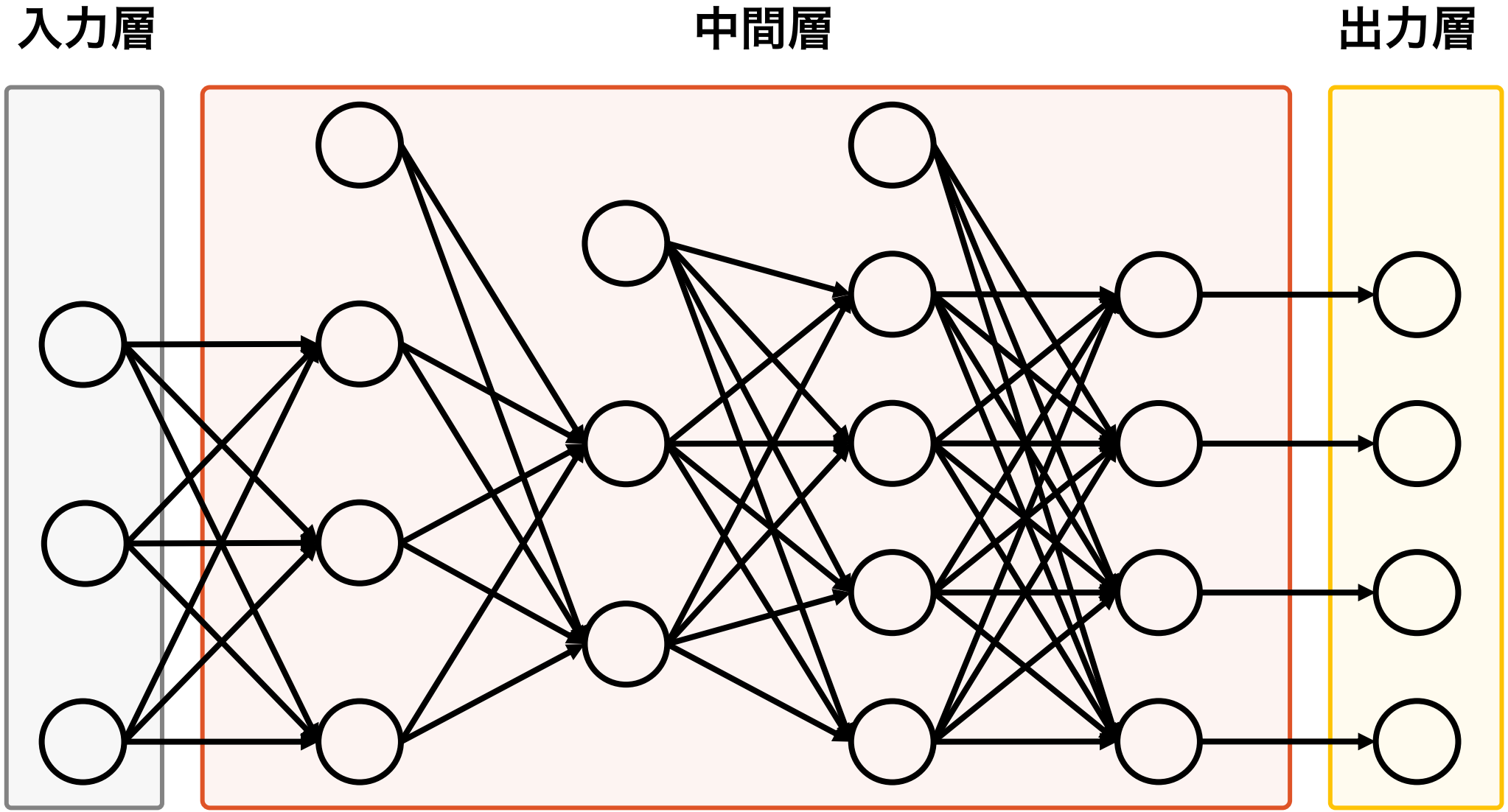
ニューラルネットワーク



ニューラルネットワーク



ニューラルネットワーク



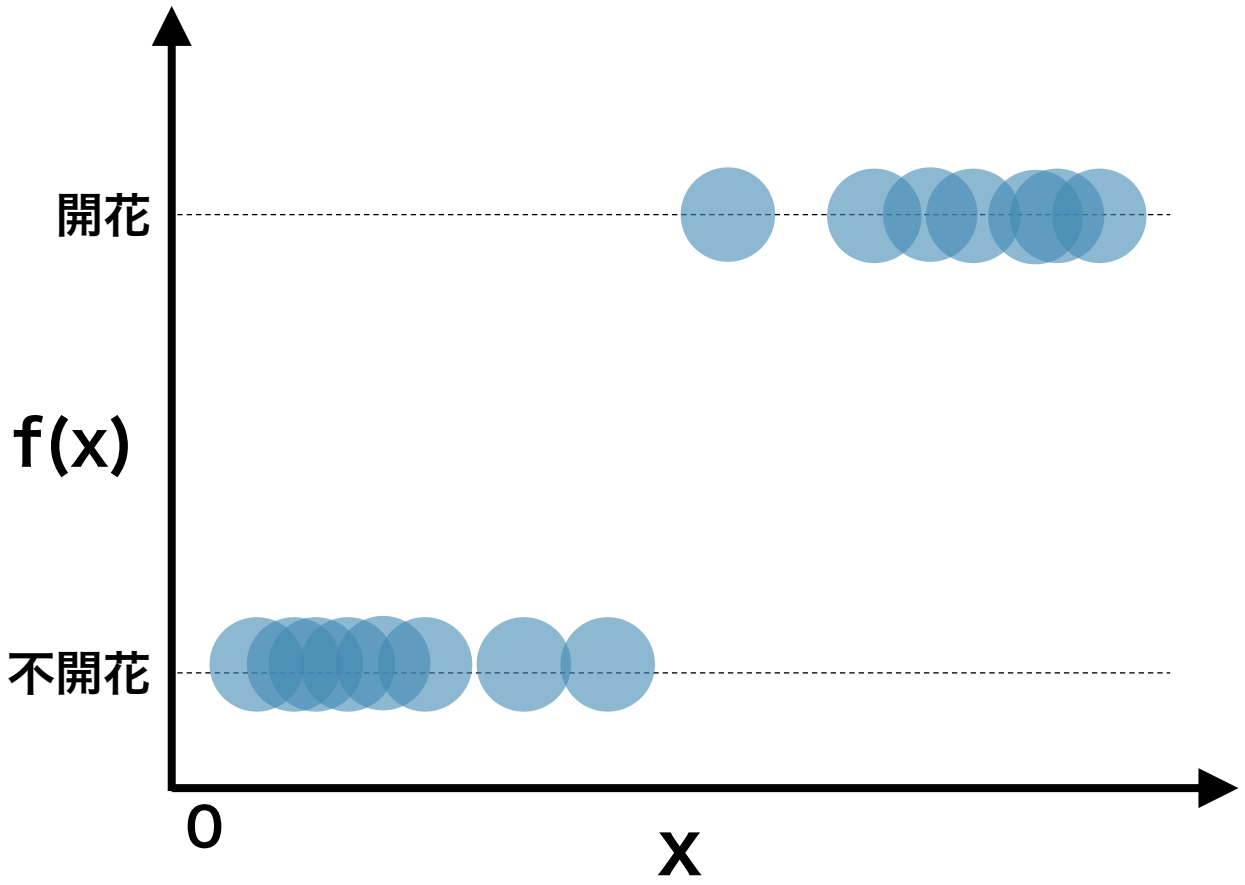
ロジスティック回帰

ロジスティック回帰

ロジスティック回帰は 2 クラスの分類問題を解くアルゴリズムである。例えば、ある日の気温を与えて、その翌日に花が咲くかどうかを分類（予測）する問題に適用できる。

$$f(x) = w_0 + w_1 x$$

▲ 開花　　▲ 気温



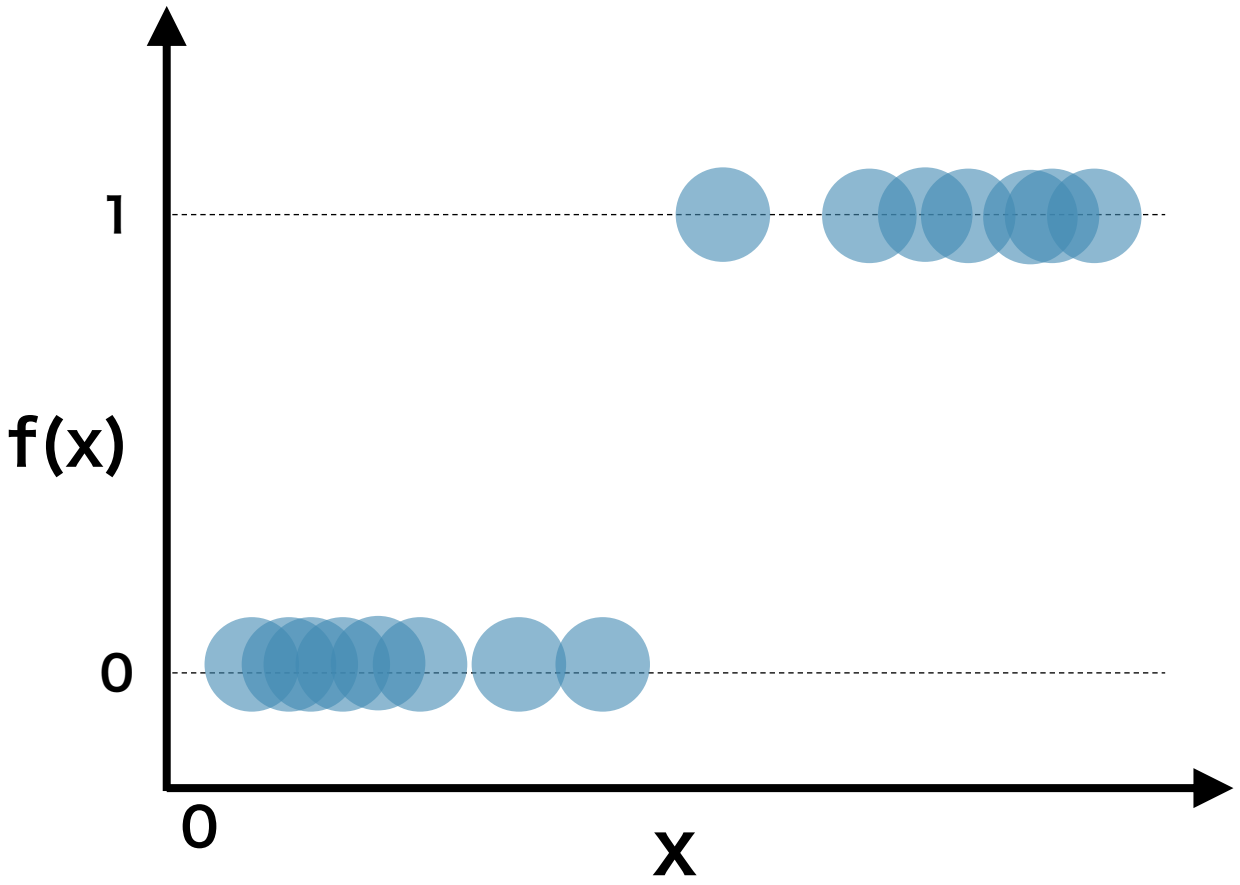
ロジスティック回帰

ロジスティック回帰は 2 クラスの分類問題を解くアルゴリズムである。例えば、ある日の気温を与えて、その翌日に花が咲くかどうかを分類（予測）する問題に適用できる。

$$f(x) = w_0 + w_1 x$$

▲ 開花　　▲ 気温

ラベルは開花・不開花の二通りであるから、開花を 1、不開花を 0 としても一般性を失わない。



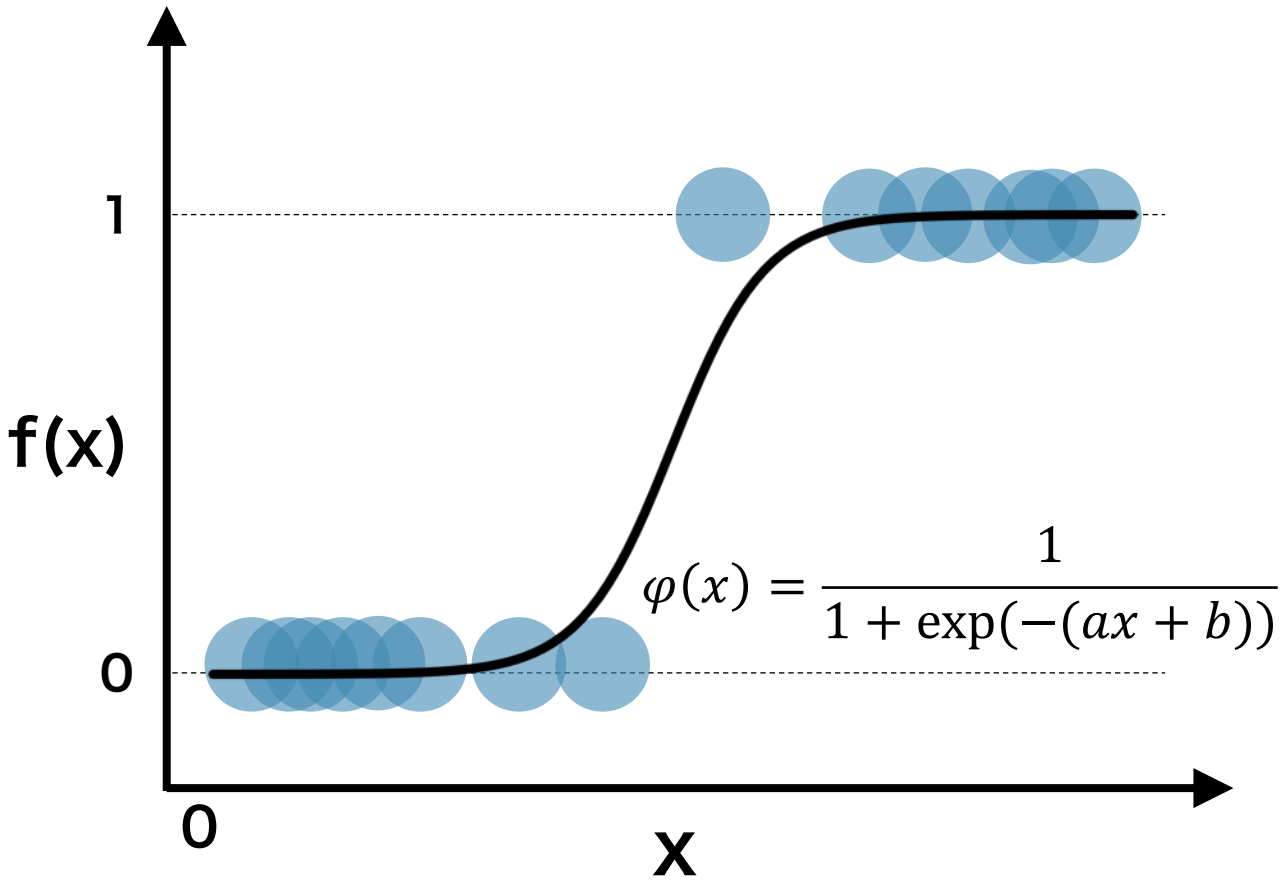
ロジスティック回帰

ロジスティック回帰は 2 クラスの分類問題を解くアルゴリズムである。例えば、ある日の気温を与えて、その翌日に花が咲くかどうかを分類（予測）する問題に適用できる。

$$f(x) = w_0 + w_1 x$$

▲ 開花　　▲ 気温

ラベルは開花・不開花の二通りであるから、開花を 1、不開花を 0 としても一般性を失わない。そこで、気温 x を受け取り、出力値が 0 から 1 までの関数 f を考えれば、関数 f は開花"率"を計算する関数と見なすことができる。



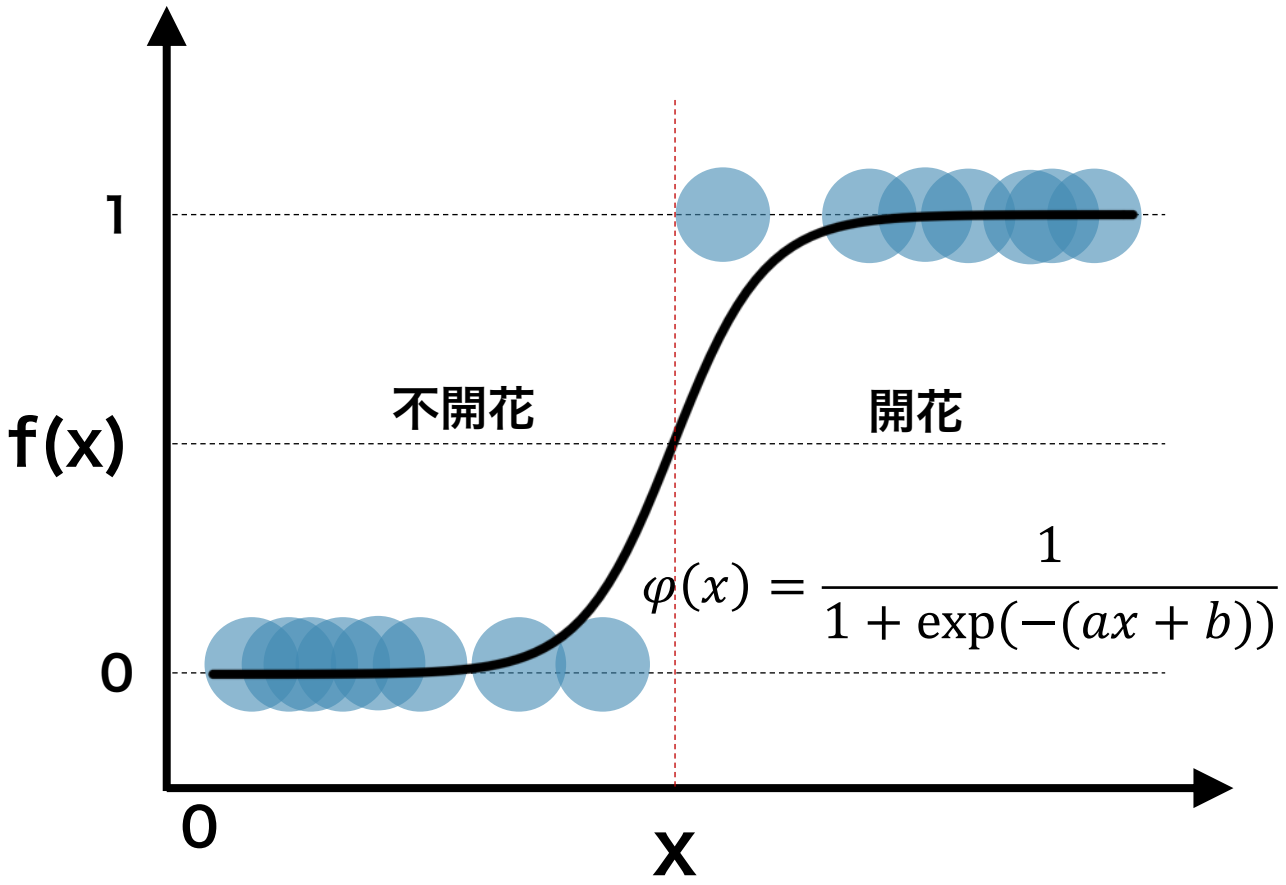
ロジスティック回帰

ロジスティック回帰は 2 クラスの分類問題を解くアルゴリズムである。例えば、ある日の気温を与えて、その翌日に花が咲くかどうかを分類（予測）する問題に適用できる。

$$f(x) = w_0 + w_1 x$$

▲ 開花　　▲ 気温

ラベルは開花・不開花の二通りであるから、開花を 1、不開花を 0 としても一般性を失わない。そこで、気温 x を受け取り、出力値が 0 から 1 までの関数 f を考えれば、関数 f は開花"率"を計算する関数と見なすことができる。そして、開花率が 0.5 以上ならば開花に分類し、そうでなければ不開花に分類することができるようになる。



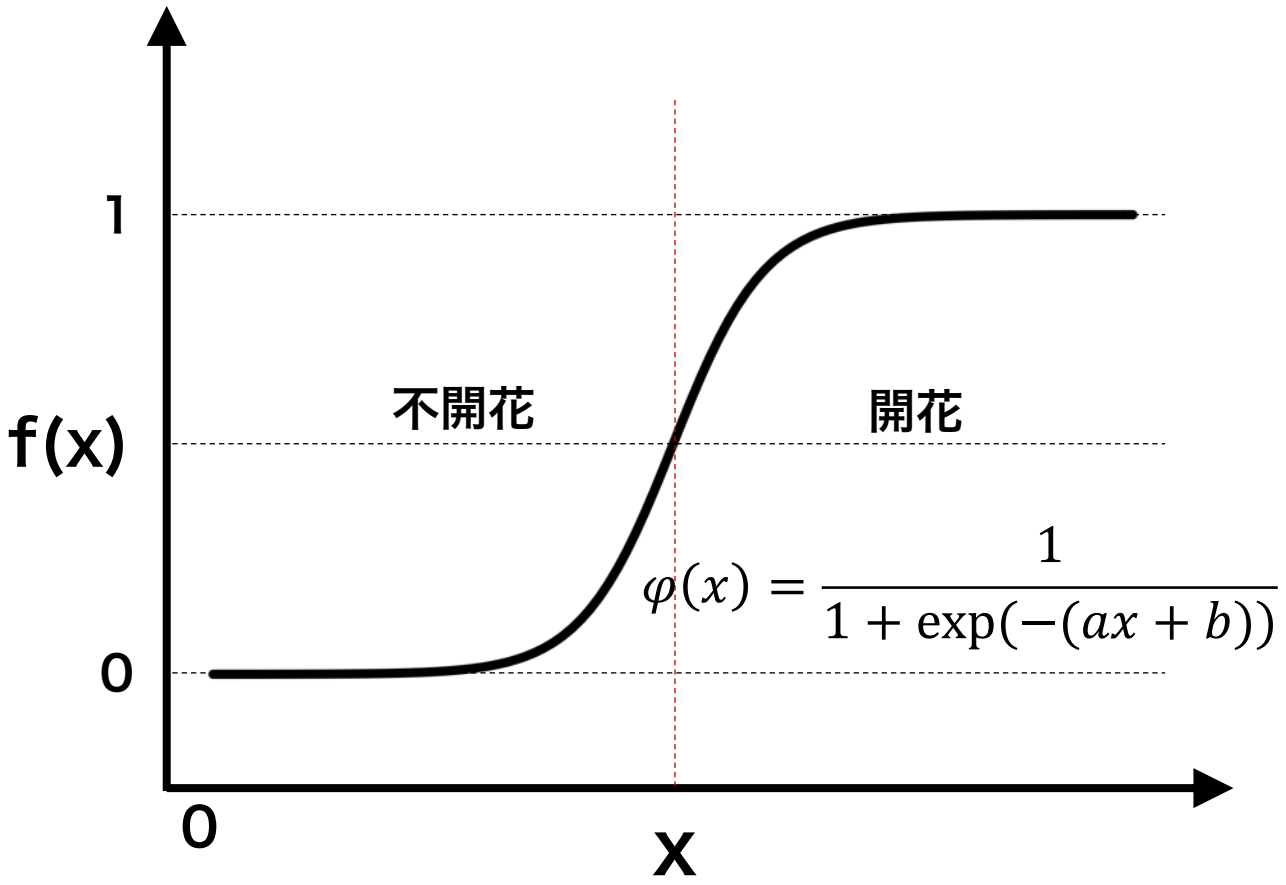
ロジスティック回帰

ロジスティック回帰は 2 クラスの分類問題を解くアルゴリズムである。例えば、ある日の気温を与えて、その翌日に花が咲くかどうかを分類（予測）する問題に適用できる。

$$f(x) = w_0 + w_1 x$$

開花　　気温

ラベルは開花・不開花の二通りであるから、開花を 1、不開花を 0 としても一般性を失わない。そこで、気温 x を受け取り、出力値が 0 から 1 までの関数 f を考えれば、関数 f は開花"率"を計算する関数と見なすことができる。そして、開花率が 0.5 以上ならば開花に分類し、そうでなければ不開花に分類することができるようになる。



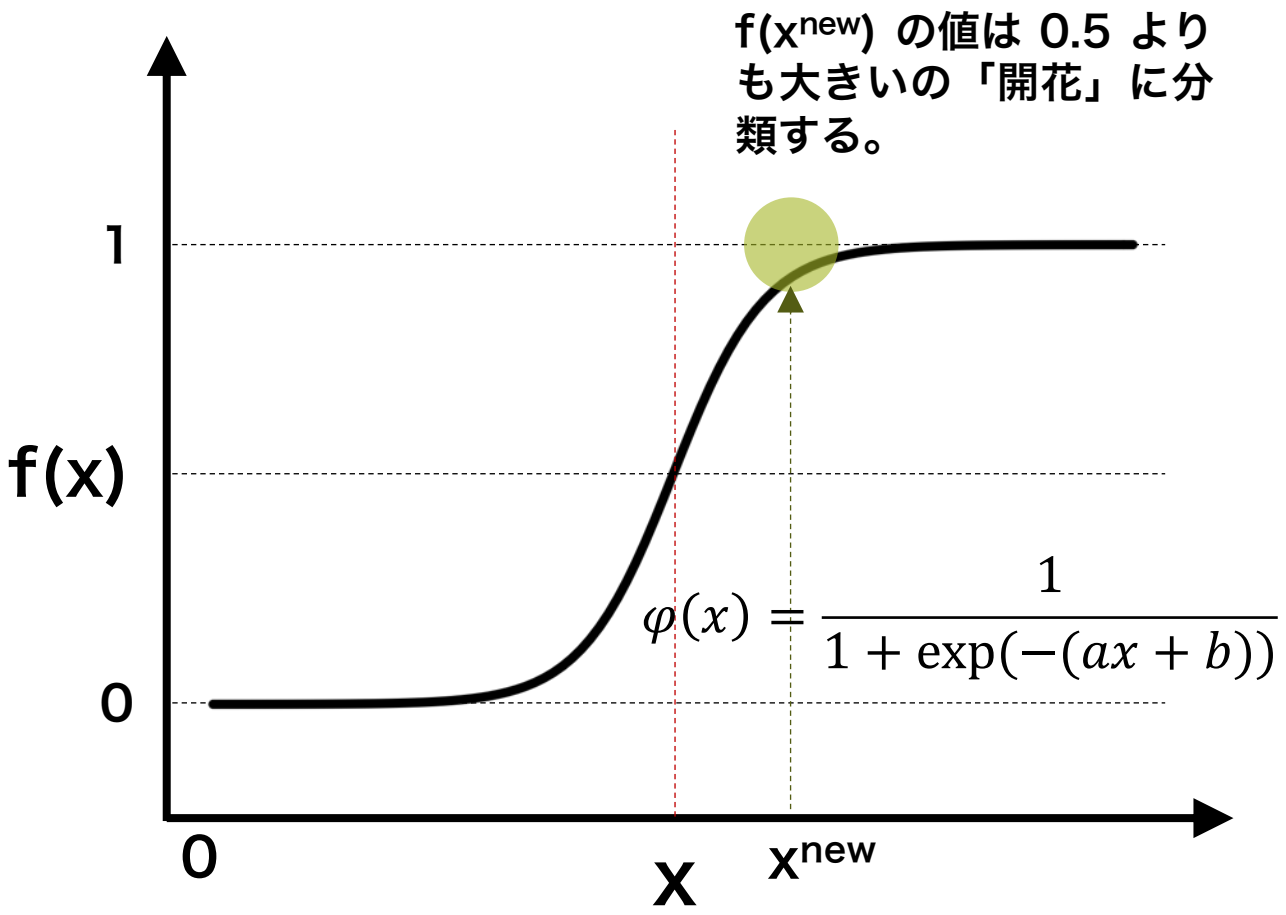
ロジスティック回帰

ロジスティック回帰は 2 クラスの分類問題を解くアルゴリズムである。例えば、ある日の気温を与えて、その翌日に花が咲くかどうかを分類（予測）する問題に適用できる。

$$f(x) = w_0 + w_1 x$$

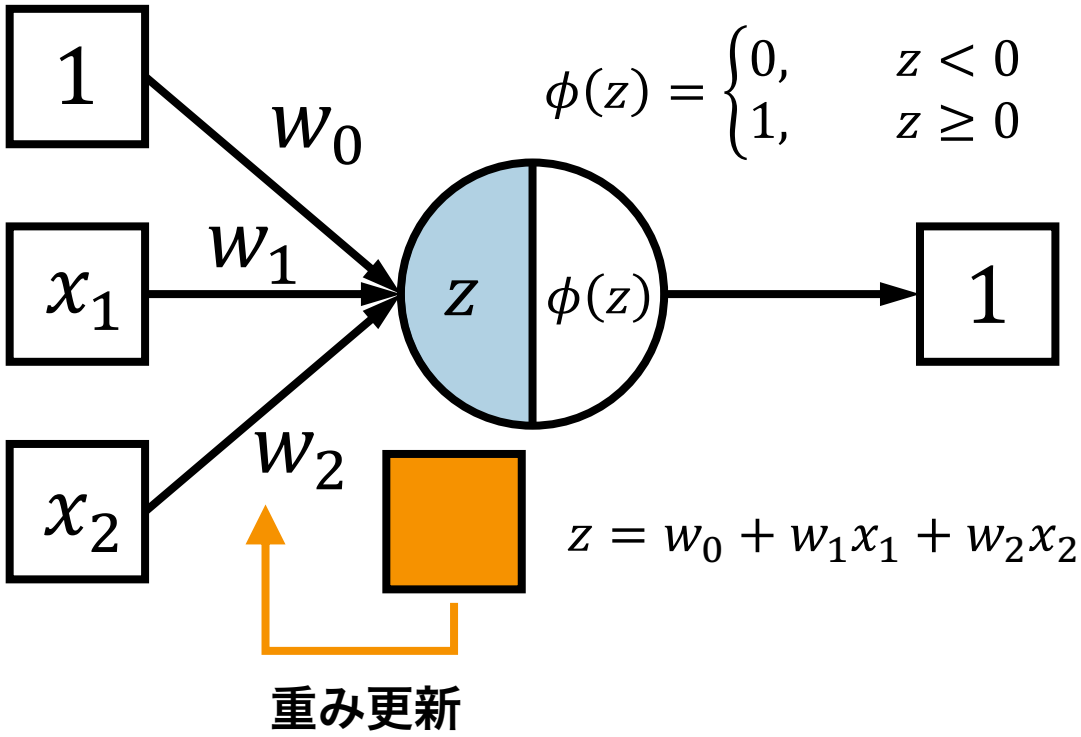
▲ 開花　　▲ 気温

ラベルは開花・不開花の二通りであるから、開花を 1、不開花を 0 としても一般性を失わない。そこで、気温 x を受け取り、出力値が 0 から 1 までの関数 f を考えれば、関数 f は開花"率"を計算する関数と見なすことができる。そして、開花率が 0.5 以上ならば開花に分類し、そうでなければ不開花に分類することができるようになる。



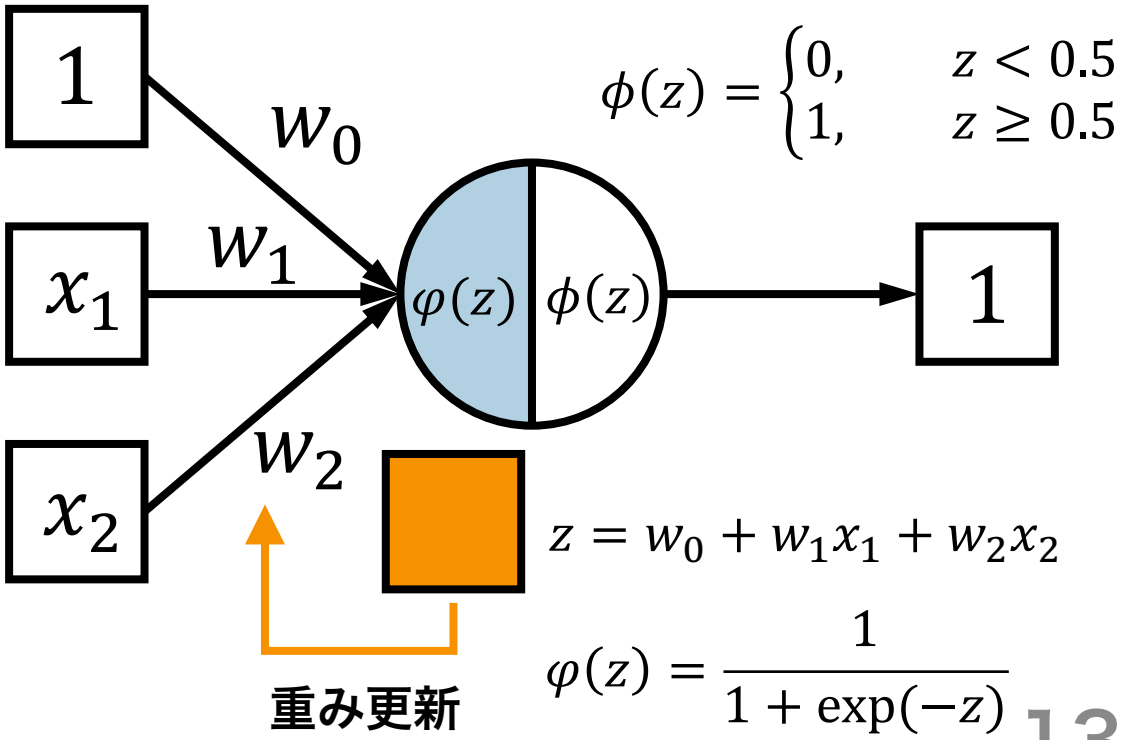
ADALINE

ADALINE は、入力された特徴量それぞれに対して重みをかけて、その総和 z を計算し、総和 z と正解ラベルの差を利用してパラメータの更新を行うアルゴリズムである。



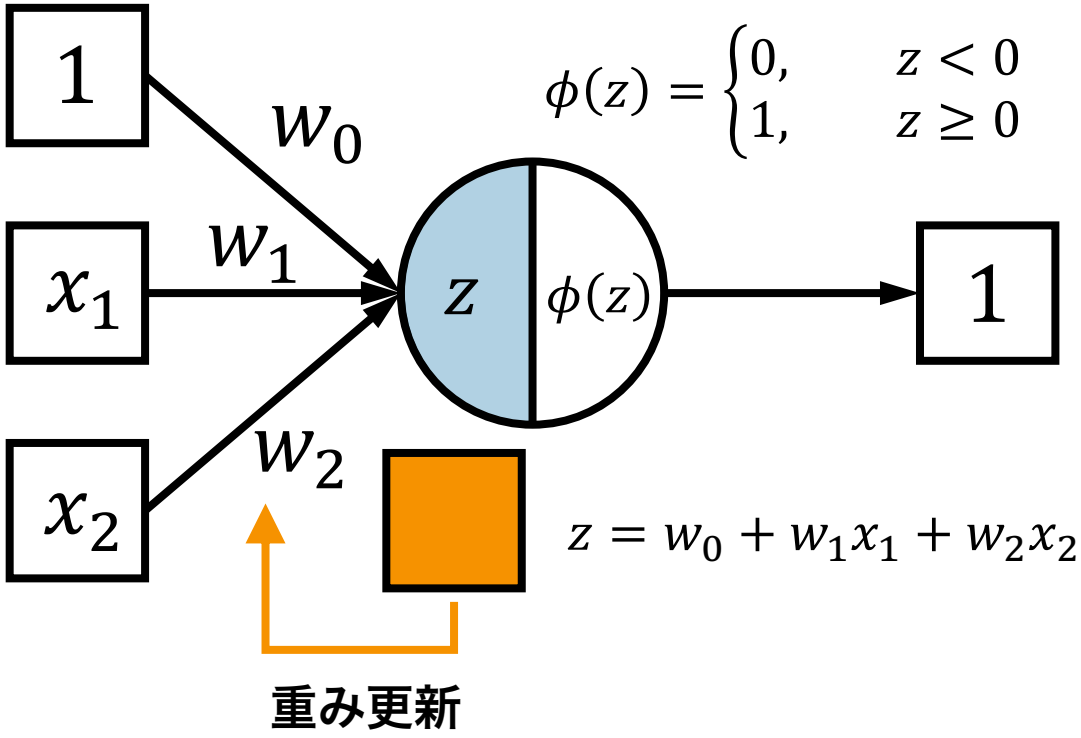
ロジスティック回帰

ロジスティック回帰は、入力された特徴量それぞれに対して重みをかけて、その総和 z を計算する。続いて、総和 z をロジット関数φに代入し、ロジット関数の出力値 $\varphi(z)$ と正解ラベルの差を利用してパラメータの更新を行うアルゴリズムである。



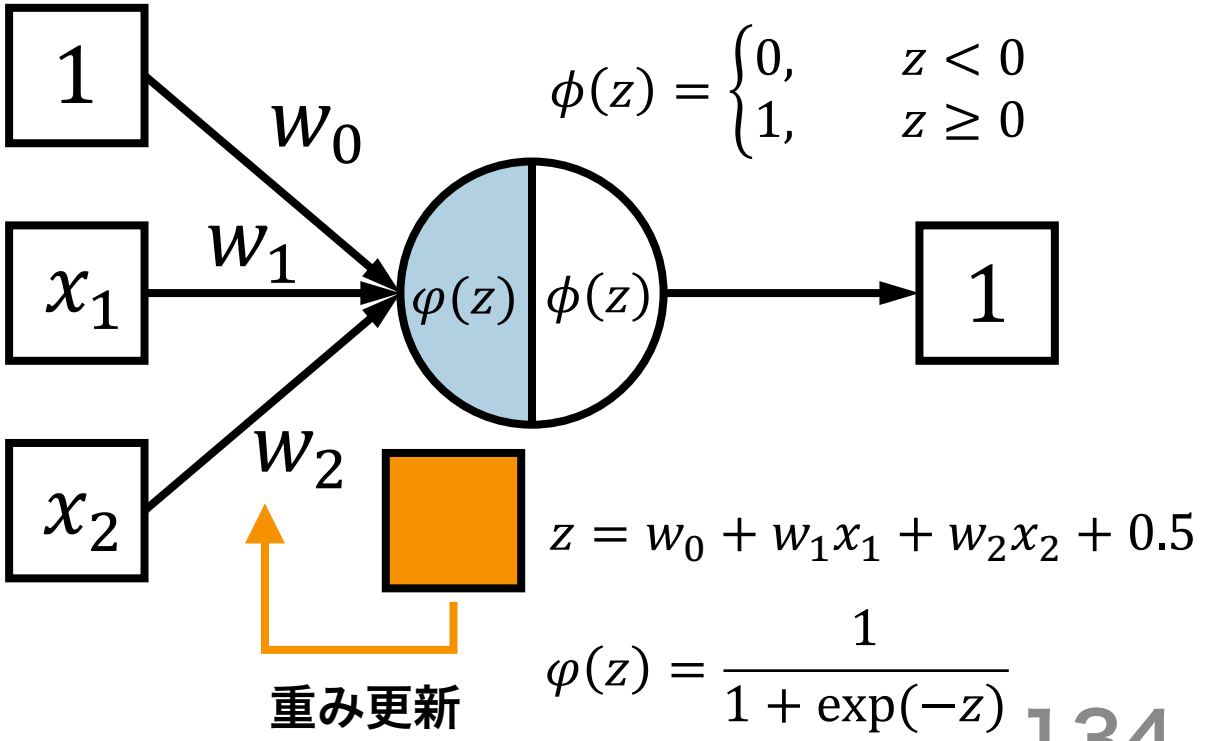
ADALINE

ADALINE は、入力された特徴量それぞれに対して重みをかけて、その総和 z を計算し、総和 z と正解ラベルの差を利用してパラメータの更新を行うアルゴリズムである。



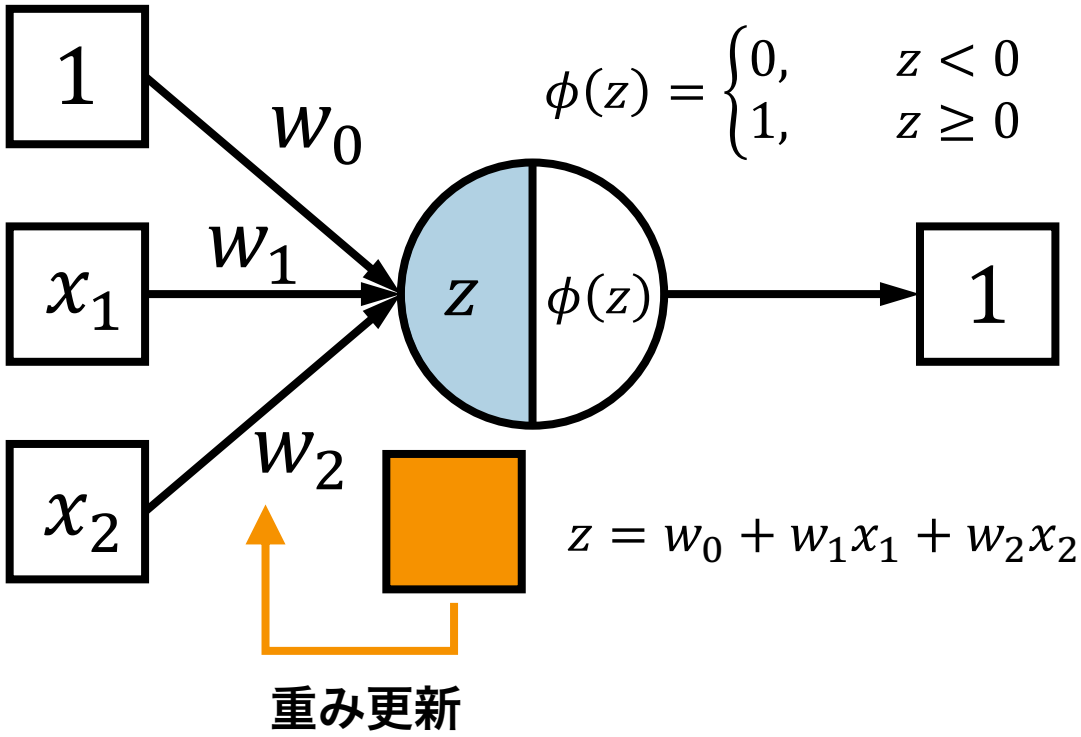
ロジスティック回帰

ロジスティック回帰は、入力された特徴量それぞれに対して重みをかけて、その総和 z を計算する。続いて、総和 z をロジット関数φに代入し、ロジット関数の出力値 $\varphi(z)$ と正解ラベルの差を利用してパラメータの更新を行うアルゴリズムである。



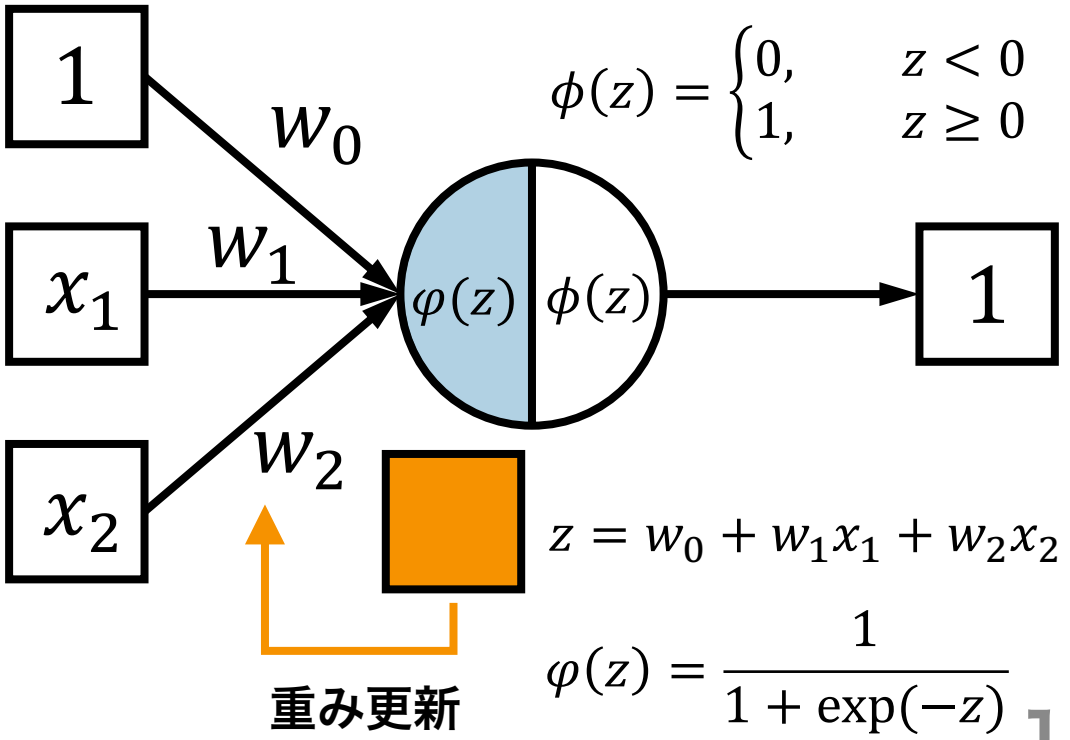
ADALINE

ADALINE は、入力された特徴量それぞれに対して重みをかけて、その総和 z を計算し、総和 z と正解ラベルの差を利用してパラメータの更新を行うアルゴリズムである。



ロジスティック回帰

ロジスティック回帰は、入力された特徴量それぞれに対して重みをかけて、その総和 z を計算する。続いて、総和 z をロジット関数φに代入し、ロジット関数の出力値 $\varphi(z)$ と正解ラベルの差を利用してパラメータの更新を行うアルゴリズムである。



ロジスティック回帰

訓練データのラベルは 0 または 1 である。訓練データが n 個存在し、かつ、互いに独立ならば、その尤度関数は、0 ラベルの確率と 1 ラベルの確率の積として書き表される。

$$lik(w) = \prod_{i=1}^{n} P\left(y^{(i)} \middle| x^{(i)}; \boldsymbol{w}\right) = \prod_{i=1}^{n} \left(\varphi(z^{(i)})\right)^{y^{(i)}} \left(1 - \varphi(z^{(i)})\right)^{1-y^{(i)}}$$

訓練データをできるだけ正確に説明するためには、この尤度関数の値をできるだけ大きくする必要がある。つまり、学習を通して、この尤度関数をもっとも大きくする $z = w^t x$（つまりパラメータ w）の値を求める必要がある。

しかし、確率の積は小数値になる。データ数が大きくなると、尤度関数は限りなく 0 に近い値になり、計算誤差が大きくなる。そこで、計算しやすいように、尤度関数を対数化して、対数尤度関数を求める。

$$logLik(w) = \sum_{i=1}^{n} \left(y^{(i)} \log\left(\varphi(z^{(i)})\right) + \left(1 - y^{(i)}\right) \log\left(1 - \varphi(z^{(i)})\right)\right)$$

尤度関数の最大化問題は、対数尤度関数の最大化問題と同値である。

ロジスティック回帰

対数尤度関数に –1 をかけたものを損失関数とする。このとき、対数尤度関数の最大化問題は、損失関数の最小化問題と同値である。

$$loss(w) = -\sum_{i=1}^{n}\left(y^{(i)}\log\left(\varphi(z^{(i)})\right) + \left(1-y^{(i)}\right)\log\left(1-\varphi(z^{(i)})\right)\right)$$

損失関数が求まったので、この損失関数を最小値を勾配降下法により求めることができるようになる。

ロジスティック回帰

損失関数の最小値を勾配降下法により求めたいので、ここで損失関数をパラメータ w で微分する。

$$\frac{\partial loss(w)}{\partial w} = \left(y\frac{1}{\varphi(z)} - (1-y)\frac{1}{1-\varphi(z)}\right)\frac{\partial \varphi(z)}{\partial w}$$

$$= \left(y\frac{1}{\varphi(z)} - (1-y)\frac{1}{1-\varphi(z)}\right)\frac{\partial \varphi(z)}{\partial z}\frac{\partial z}{\partial w}$$

$$= \left(y\frac{1}{\varphi(z)} - (1-y)\frac{1}{1-\varphi(z)}\right)\varphi(z)(1-\varphi(z))\frac{\partial z}{\partial w}$$

$$= (y(1-\varphi(z)) - (1-y)\varphi(z))x$$

$$= (y-\varphi(z))x$$

関数 $\varphi(z)$ を z で微分するときの計算

$$\frac{\partial \varphi(z)}{\partial z} = \frac{\partial}{\partial z}\frac{1}{1+e^{-z}}$$

$$= \frac{e^{-z}}{(1+e^{-z})^2}$$

$$= \frac{1}{1+e^{-z}}\left(1-\frac{1}{1+e^{-z}}\right)$$

$$= \varphi(z)(1-\varphi(z))$$

ロジスティック回帰

$$\boldsymbol{w}^{(new)} = \boldsymbol{w}^{(old)} + \Delta \boldsymbol{w}$$

$$\Delta \boldsymbol{w} = -\eta \boldsymbol{\nabla} loss(\boldsymbol{w})$$

$$\boldsymbol{\nabla} loss(\boldsymbol{w}) = \frac{\partial loss(\boldsymbol{w})}{\partial \boldsymbol{w}}$$

損失関数 loss を w で偏微分すると、

$$\frac{\partial loss(w)}{\partial w} = (y - \varphi(z))x$$

が得られる。よって w_j は次の式に基づいて更新される。

$$w_j^{(new)} = w_j^{(old)} - \eta \frac{\partial loss(\boldsymbol{w})}{\partial w_j}$$

$$= w_j^{(old)} - \eta \sum_i^n \left(y^{(i)} - \phi\left(z^{(i)}\right) \right) x_j^{(i)}$$

ロジスティック回帰（正則化）

ロジスティック回帰モデルの過学習を抑制し、汎化性能を高めるために、極端なパラメーター（係数）にペナルティを設けることが一般的に行われている。これを正則化という。

$$loss(w) = -\sum_{i=1}^{n}\left(y^{(i)}\log\left(\varphi\left(z^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-\varphi\left(z^{(i)}\right)\right)\right) + \frac{\lambda}{2}\|\boldsymbol{w}\|^2$$

正則化項

※ 正則化には L1 正則化と L2 正則化等がある。ロジスティック回帰において L2 正則化が一般的に使われている。

※ 正則化が正常に機能させるためには、すべての特徴量が比較可能なスケールに変換する必要がある。

※ scikit-learn パッケージの中では、C=1/λ として、次のように正則化の大きさを調整する。

$$loss(w) = -C\sum_{i=1}^{n}\left(y^{(i)}\log\left(\varphi\left(z^{(i)}\right)\right) + \left(1-y^{(i)}\right)\log\left(1-\varphi\left(z^{(i)}\right)\right)\right) + \frac{1}{2}\|\boldsymbol{w}\|^2$$

多クラスの分類

ロジスティック回帰は基本的に 2 つのクラスと 0 と 1 に置き換えて、モデルを立て、パラメーターの推定を行ってきた。分類すべきクラスが 3 つ以上ある場合、いくつかのアプローチを取ることができる。

- One-vs-the-rest

  「クラス i」と「それ以外」を分類するモデルをすべてのクラスに対して作成する。すべての予測モデルの適合度のもっとも高いクラスを予測結果とする。

- One-vs-the-rest

  2 クラス分類モデルを、すべての2 クラスの組み合わせで作成する。そして、これらのモデルの多数決により予測を行う。

全予測モデルの出力値（確率）の合計が 1.0 となるようにソフトマックス関数で調整する。

$$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_c \end{pmatrix} = \frac{1}{\sum_{i=1}^{c} e^{z_i}} \begin{pmatrix} e^{z_1} \\ e^{z_2} \\ \vdots \\ e^{z_c} \end{pmatrix}$$

p_c は c クラスである確率（予測値）

Z_c は、モデル c のパラメータで算出した $w^t x+w_0$ である。

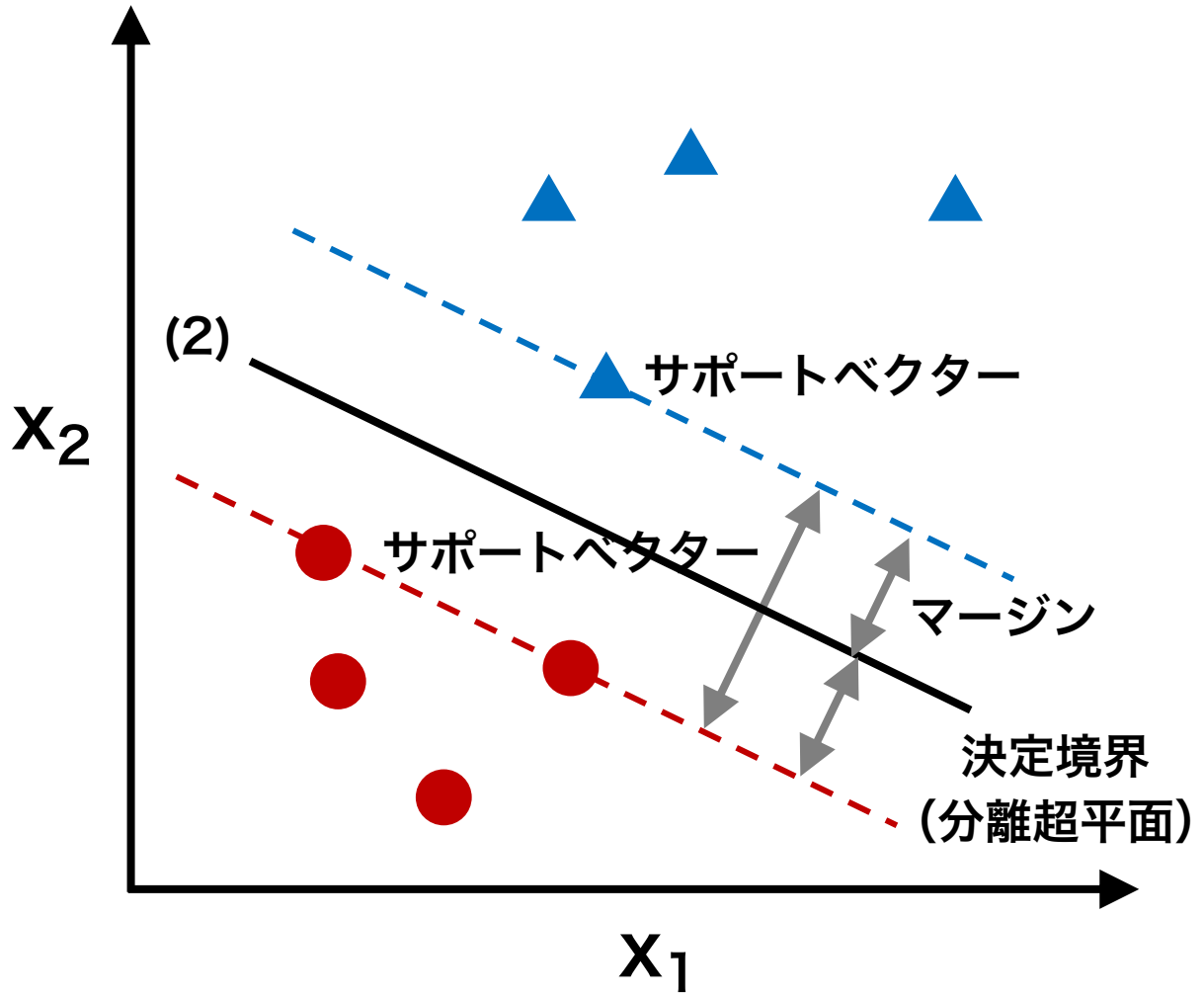
出力値のベクトルに合わせて、損失関数もベクトル演算ができるように、交差エントロピーが使われる。

SVM

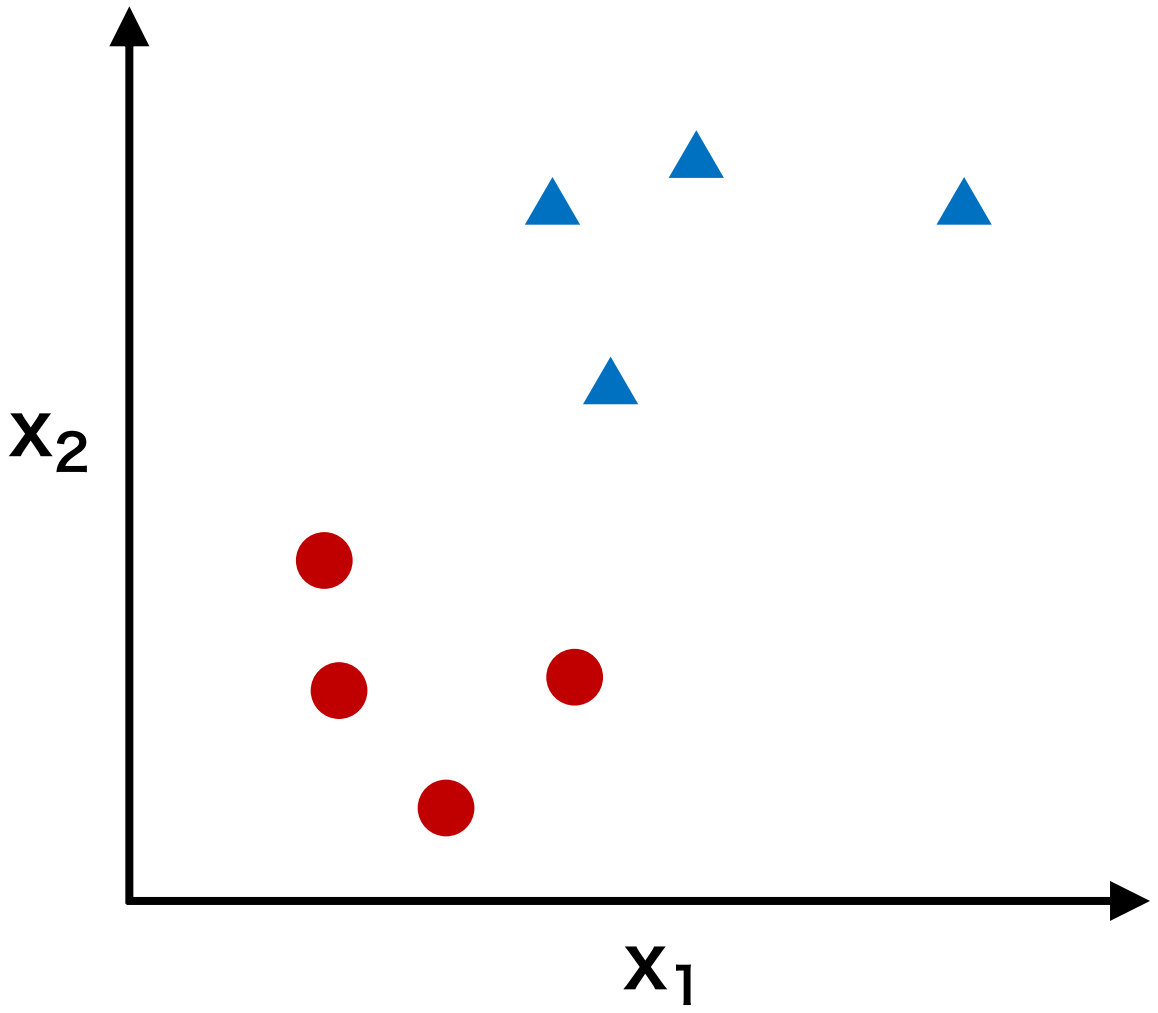
- ハードマージン
- ソフトマージン
- カーネルトリック
- SVR

SVM

SVM (support vector machine) は、2000 年前後から 2010 年前後までに広く使われた分類アルゴリズムである。SVM を用いることで、局所最小解に陥ることなく、大域最小解に到達でき、汎化性能の高い予測モデルを構築できる。

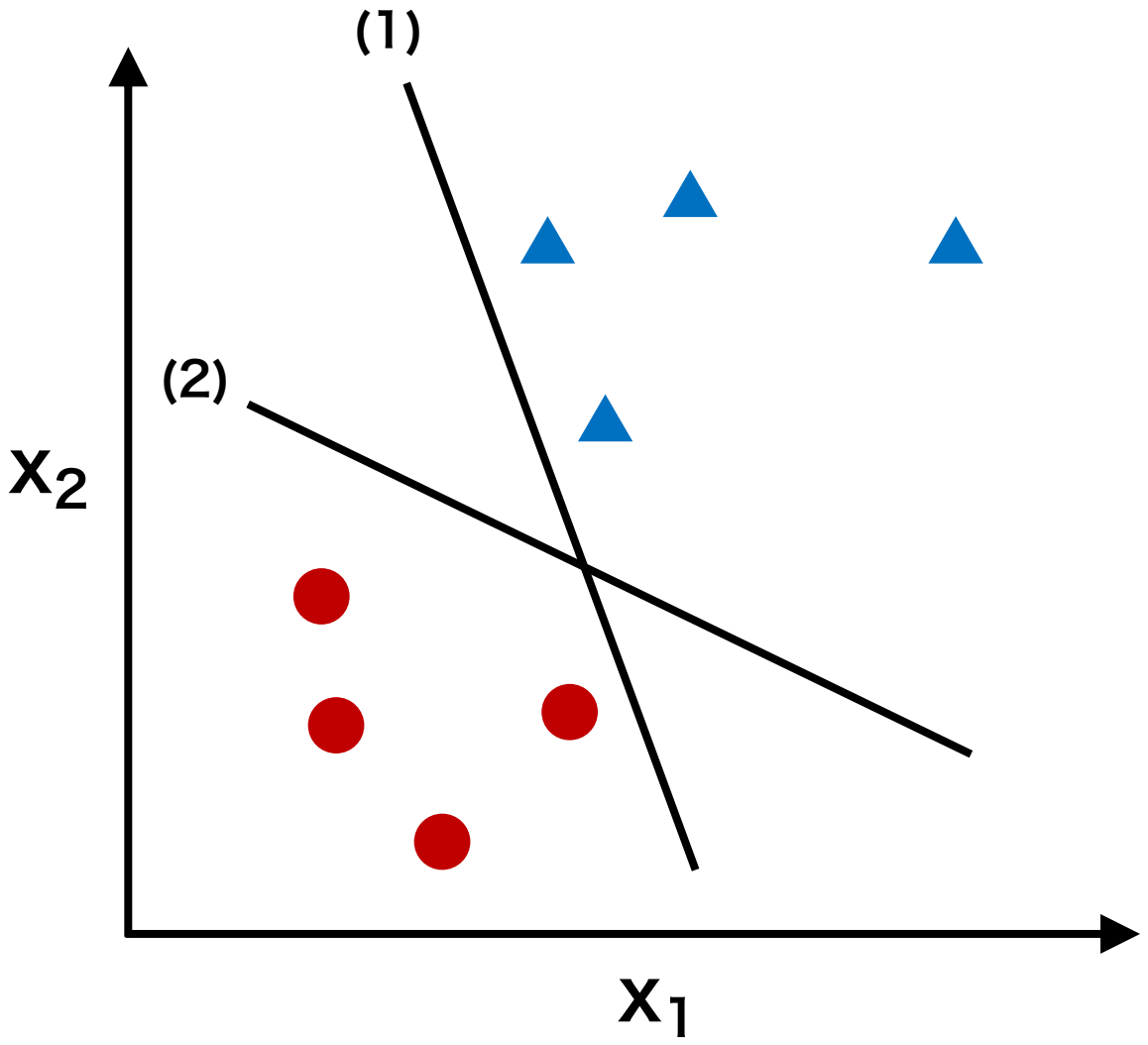


SVM



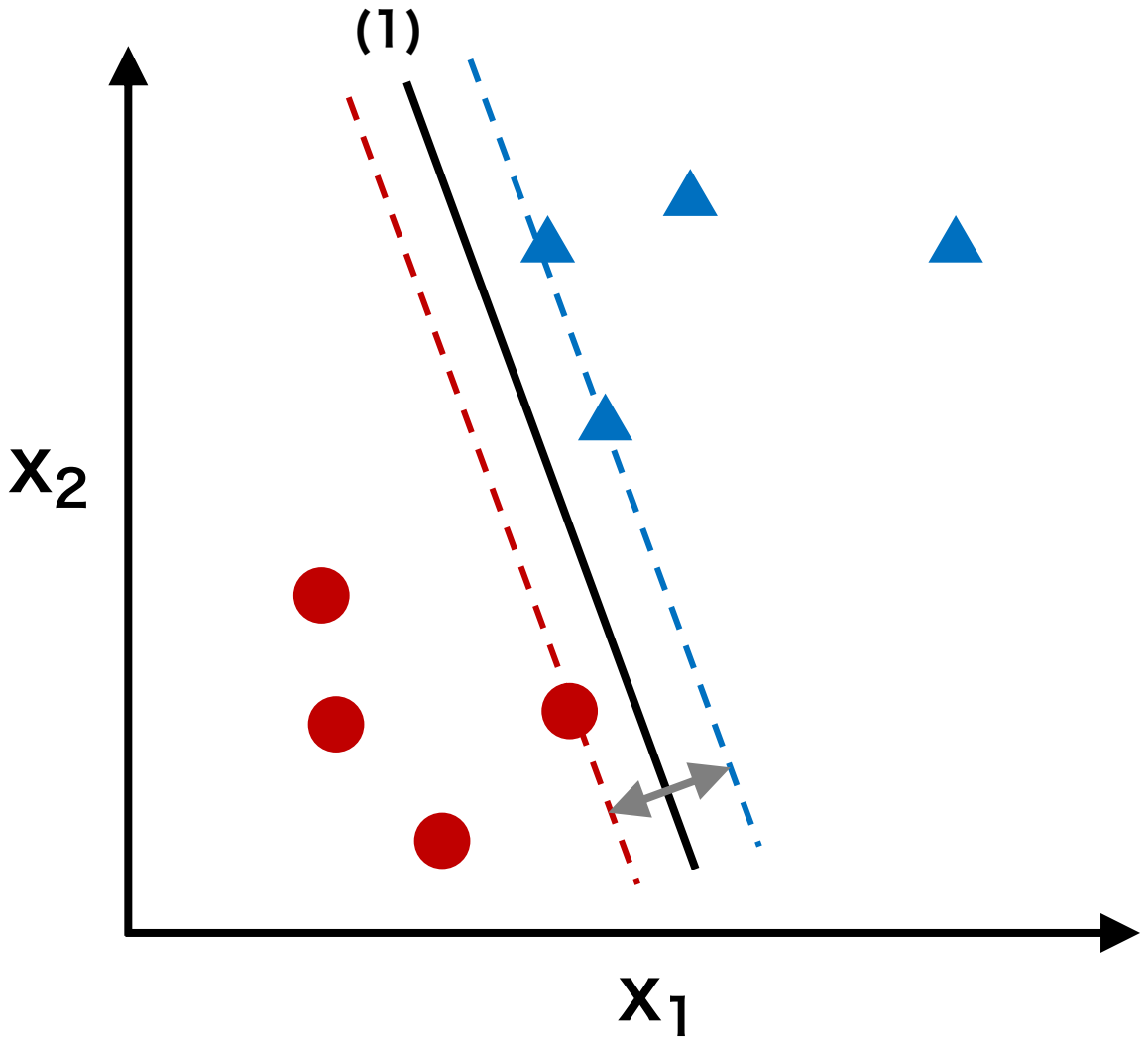
身長と体重を用いてイヌとネコを分けたい。そこで、ネコ４匹、イヌ４匹の身長（x_1）と体重（x_2）を測定した。身長と体重を散布図上に描くと、左図のようになった。ここで、　イヌとネコを分けるためにどのようにすればいいのか。

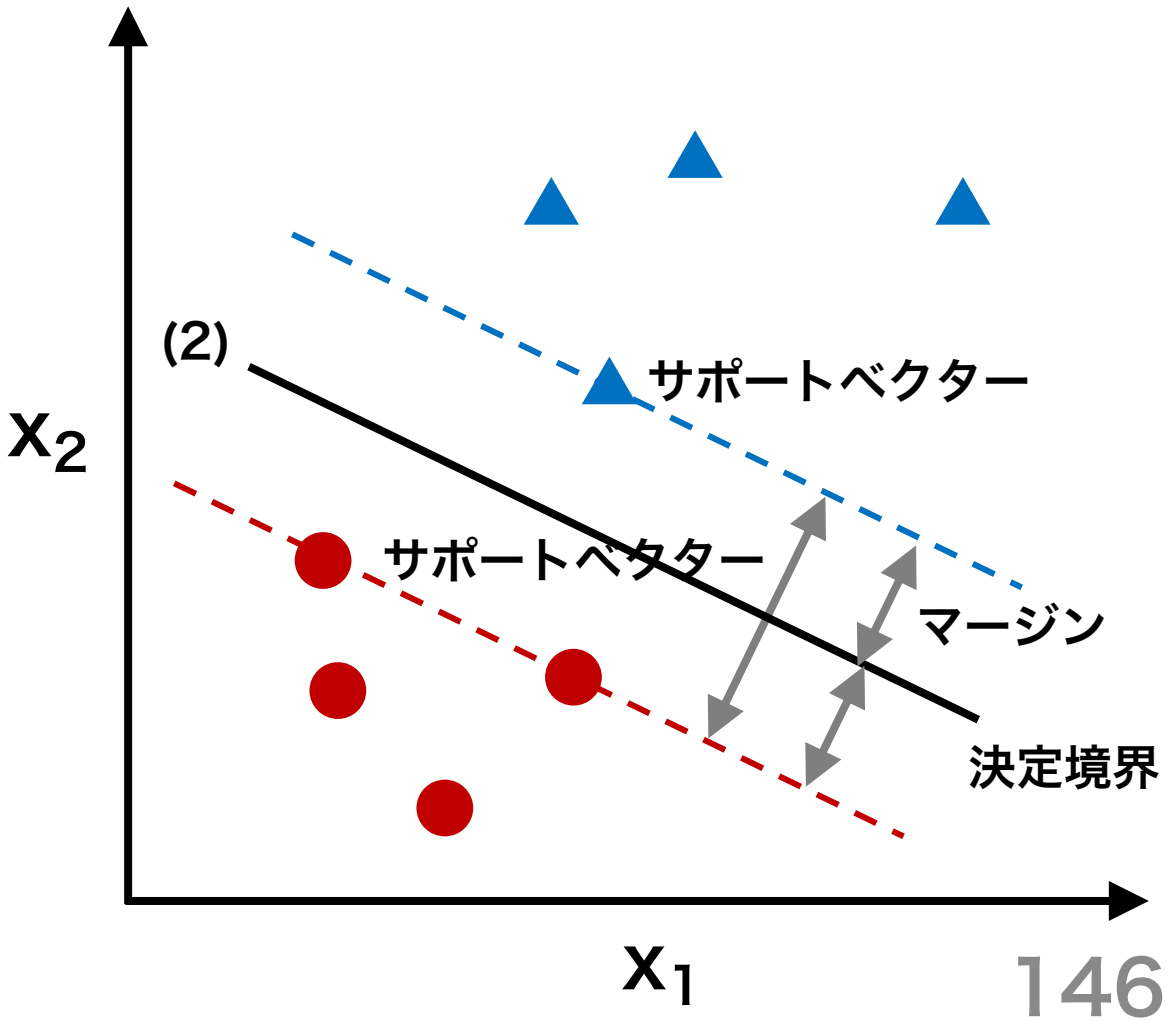
SVM



x_1x_2 平面上に 1 本の直線を描き、直線の右側をイヌで、直線の左側をネコとすることができる。直線を描く方法は複数考えられる。例えば、左図の場合、(1) および (2) の 2 本の直線のどちらも、イヌとネコを分けている。これ以上に学習データが集まらないと仮定した場合、イヌとネコを分けるために、(1) または (2) のどちらが最もよい直線であるか。

SVM

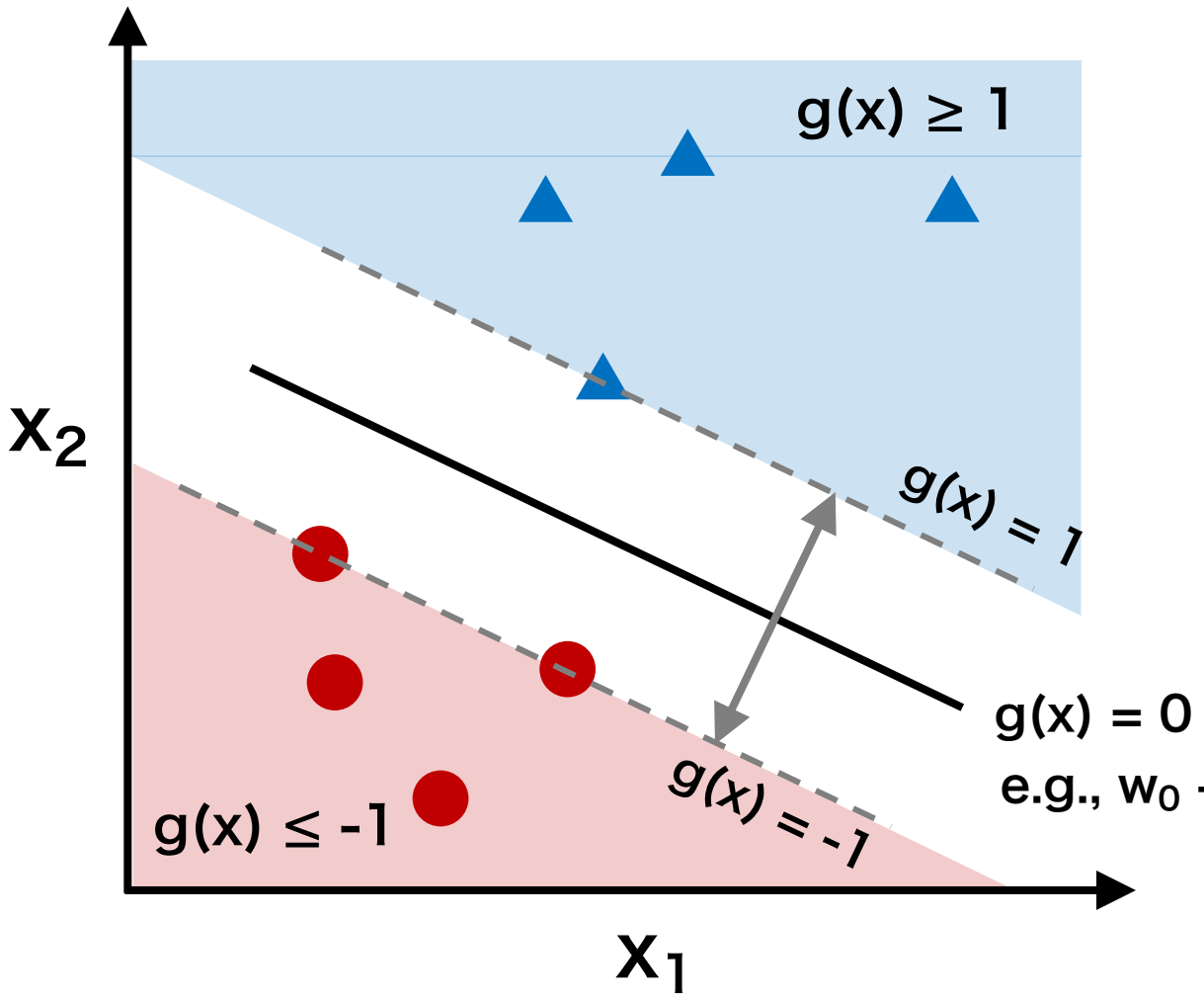




SVM

- ハードマージン
- ソフトマージン
- カーネルトリック
- SVR

SVM / hard-margin

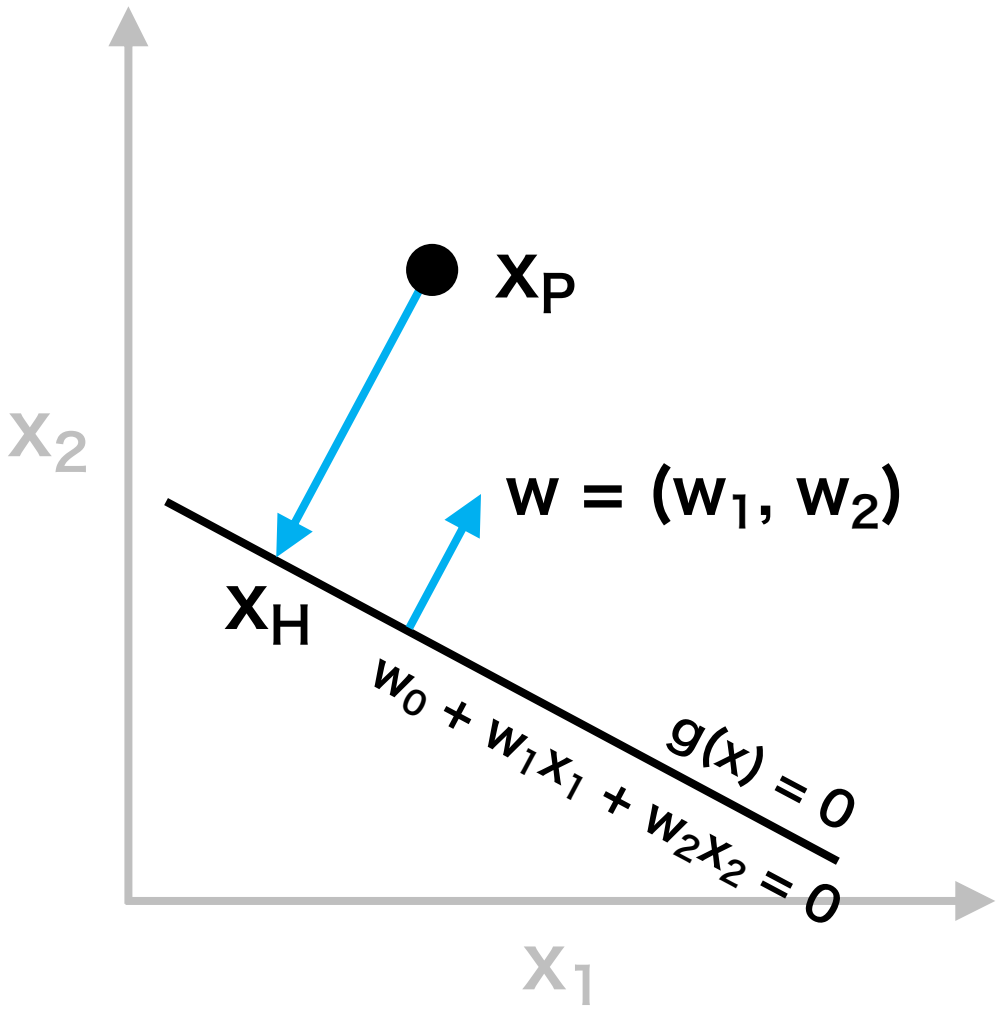


positive データと negative データは線形分離可能であると仮定する。つまり、positive データと negative データを分離できる超平面 g(x) = 0 が存在することを仮定する。例えば、特徴量が 2 次元の場合は $g(x)=w_0+w_1x_1+w_2x_2$ とおくことができる。

g(x) = 0 を定数倍しても不偏であることから、全ての positive データに対して g(x)≥1、全ての negative データに対して g(x)≤-1 としても一般性を失わない。SVM では、g(x)≥1 と g(x)≤-1 の間の距離、すなわち g(x)=1 と g(x)=-1 間の距離を最大にするような g(x) を決めることである。言い換えれば、g(x)=1 と g(x)=-1 間の距離を最大にするような w_0、w_1、および w_2 を決めることである。

e.g., $w_0 + w_1x_1 + w_2x_2 = 0$

点と直線の距離



点と直線の距離

点 $x_P = (p_1, p_2)$ から直線 $g(x) = 0$ に下ろした垂線の足を点 $x_H = (h_1, h_2)$ とする。直線 $g(x) = 0$ の法線ベクトルは $w = (w_1, w_2)$ と書けるので、ある実数 t が存在して、線分 x_Px_H と直線 $g(x) = 0$ の法線ベクトルの間に次の関係が成り立つ。

$$\overrightarrow{\boldsymbol{x}_P\boldsymbol{x}_H} = t\boldsymbol{w}$$

両辺の右から w をかけると

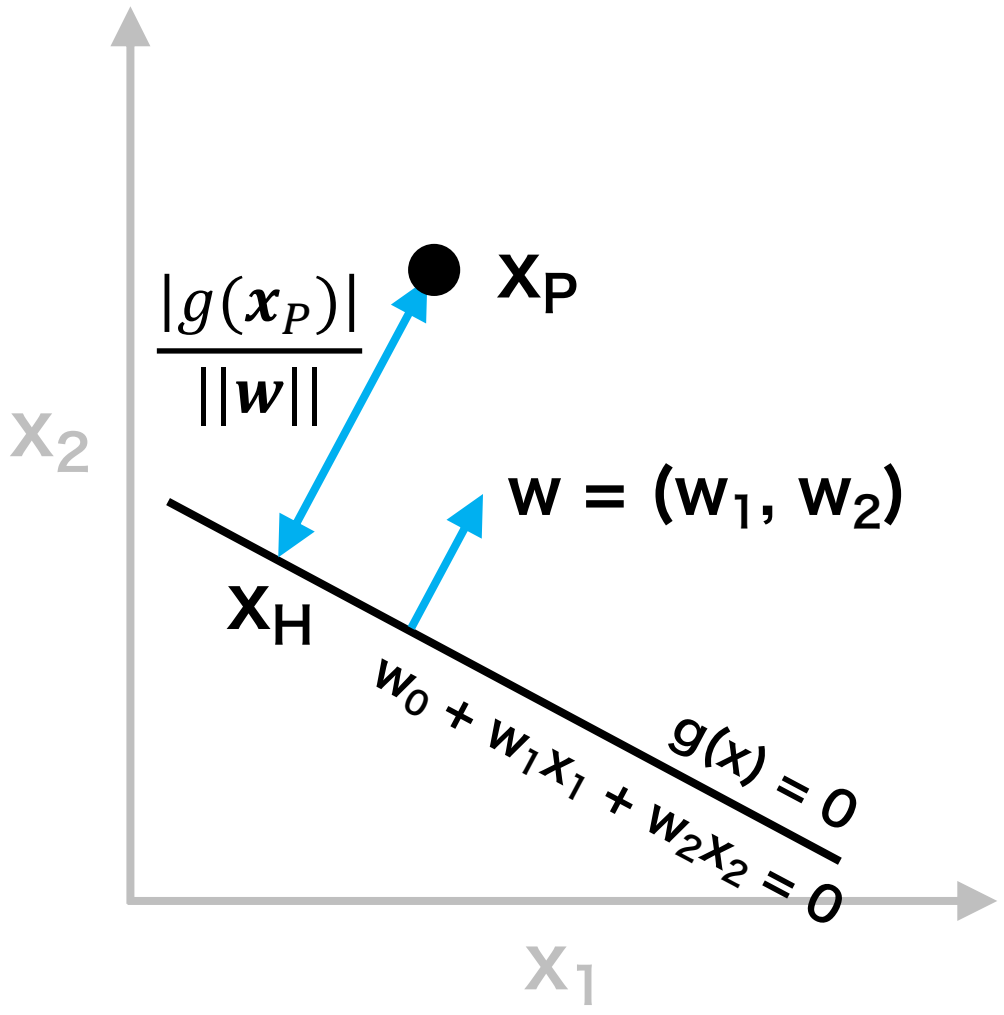
$$\boldsymbol{w}\overrightarrow{\boldsymbol{x}_P\boldsymbol{x}_H} = t\boldsymbol{w}\boldsymbol{w}$$

$$(h_1 - p_1, h_2 - p_2) \cdot (w_1, w_2) = t(w_1, w_2) \cdot (w_1, w_2)$$

$$w_1(h_1 - p_1) + w_2(h_2 - p_2) = t(w_1^2 + w_2^2)$$

$$t = \frac{w_1(h_1 - p_1) + w_2(h_2 - p_2)}{(w_1^2 + w_2^2)}$$

点と直線の距離



点と直線の距離

このとき、線分 x_Px_H の長さは、

$$|\overrightarrow{x_Px_H}| = |t||w|$$

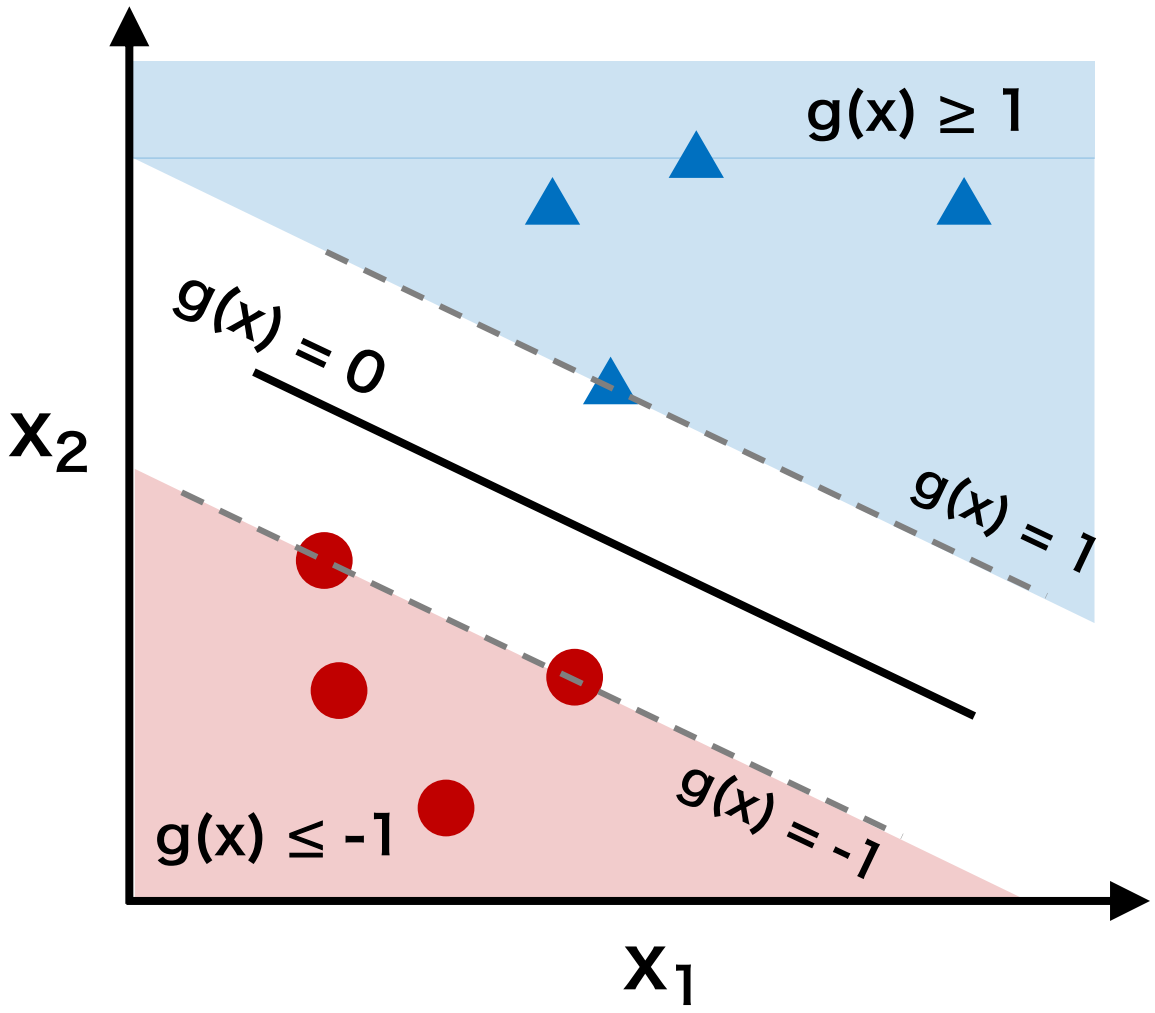
$$= \left|\frac{w_1(h_1 - p_1) + w_2(h_2 - p_2)}{(w_1^2 + w_2^2)}\right| \left|\sqrt{w_1^2 + w_2^2}\right|$$

$$= \frac{|w_1h_1 - w_1p_1 + w_2h_2 - w_2p_2|}{\sqrt{w_1^2 + w_2^2}}$$

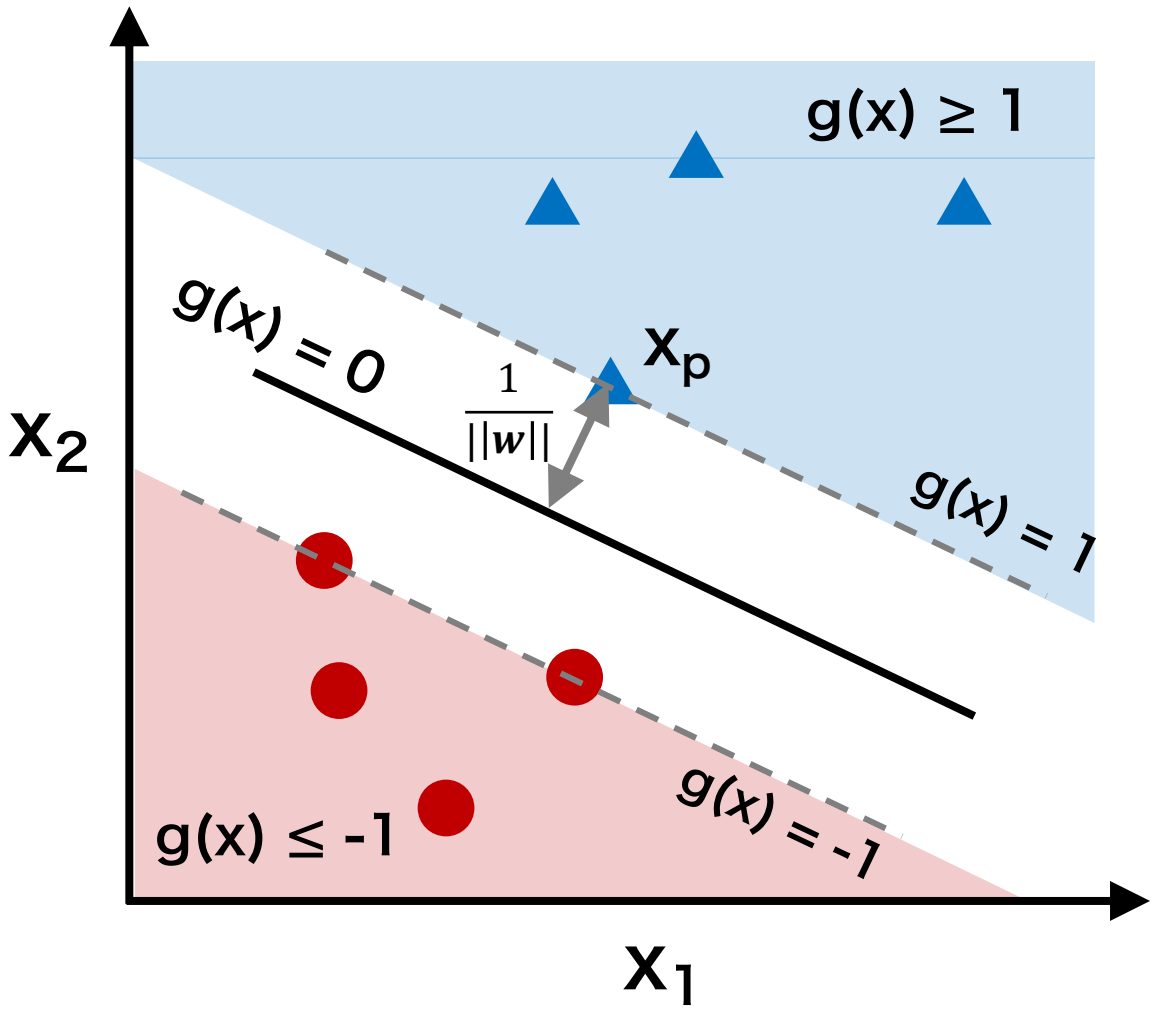
点 x_H は線分 $g(x) = 0$ 上にあるので、$w_0 + w_1h_1 + w_2h_2 = 0$ が成り立つので、上の式は

$$= \frac{|w_0 + w_1p_1 + w_2p_2|}{\sqrt{w_1^2 + w_2^2}} = \frac{|g(x_P)|}{||w||}$$

SVM / hard-margin



SVM / hard-margin



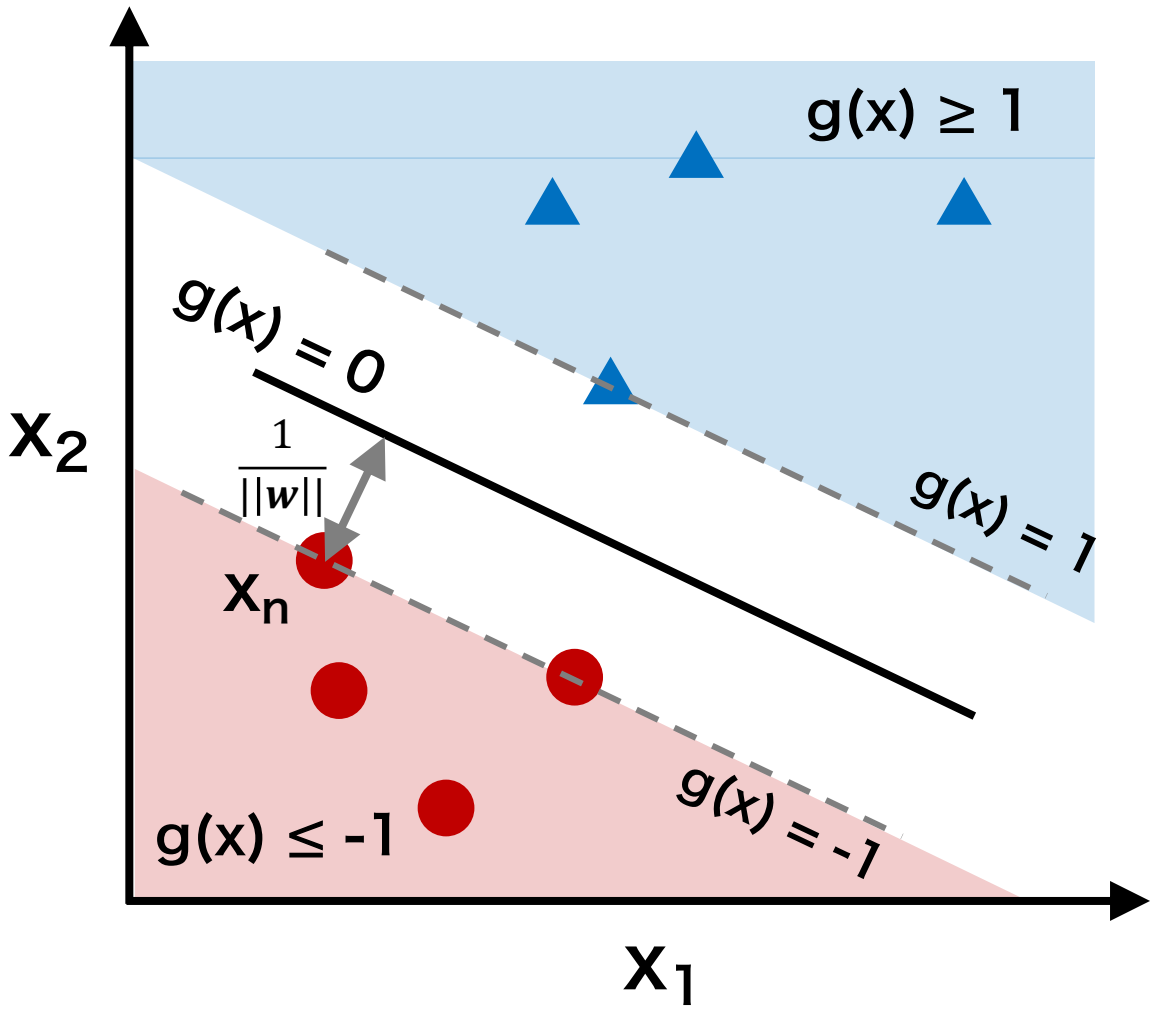
点 x_p から直線 g(x) = 0 までの距離は

$$\frac{|g(x_p)|}{||w||}$$

点 x_p は直線 g(x) = 1 上にあるから、$g(x_p) = 1$ より

$$\frac{|g(x_p)|}{||w||} = \frac{1}{||w||}$$

SVM / hard-margin



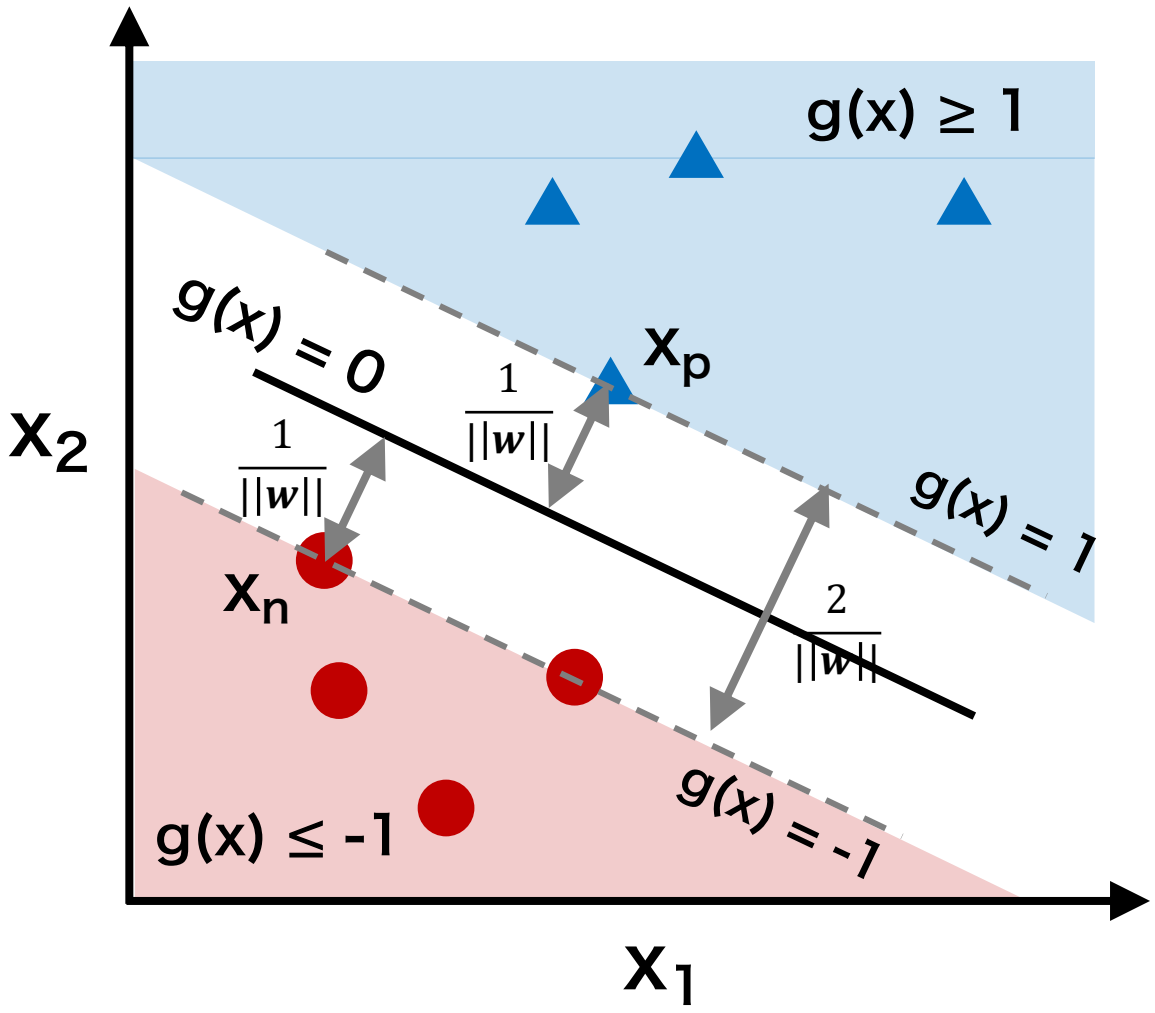
点 x_n から直線 g(x) = 0 までの距離は

$$\frac{|g(\boldsymbol{x}_n)|}{||\boldsymbol{w}||}$$

点 x_n は直線 g(x) = -1 上にあるから、$g(x_n) = -1$ より

$$\frac{|g(\boldsymbol{x}_n)|}{||\boldsymbol{w}||} = \frac{1}{||\boldsymbol{w}||}$$

SVM / hard-margin



点 x_p から直線 g(x) = 0 までの距離は

$$\frac{|g(\boldsymbol{x}_p)|}{||\boldsymbol{w}||} = \frac{1}{||\boldsymbol{w}||}$$

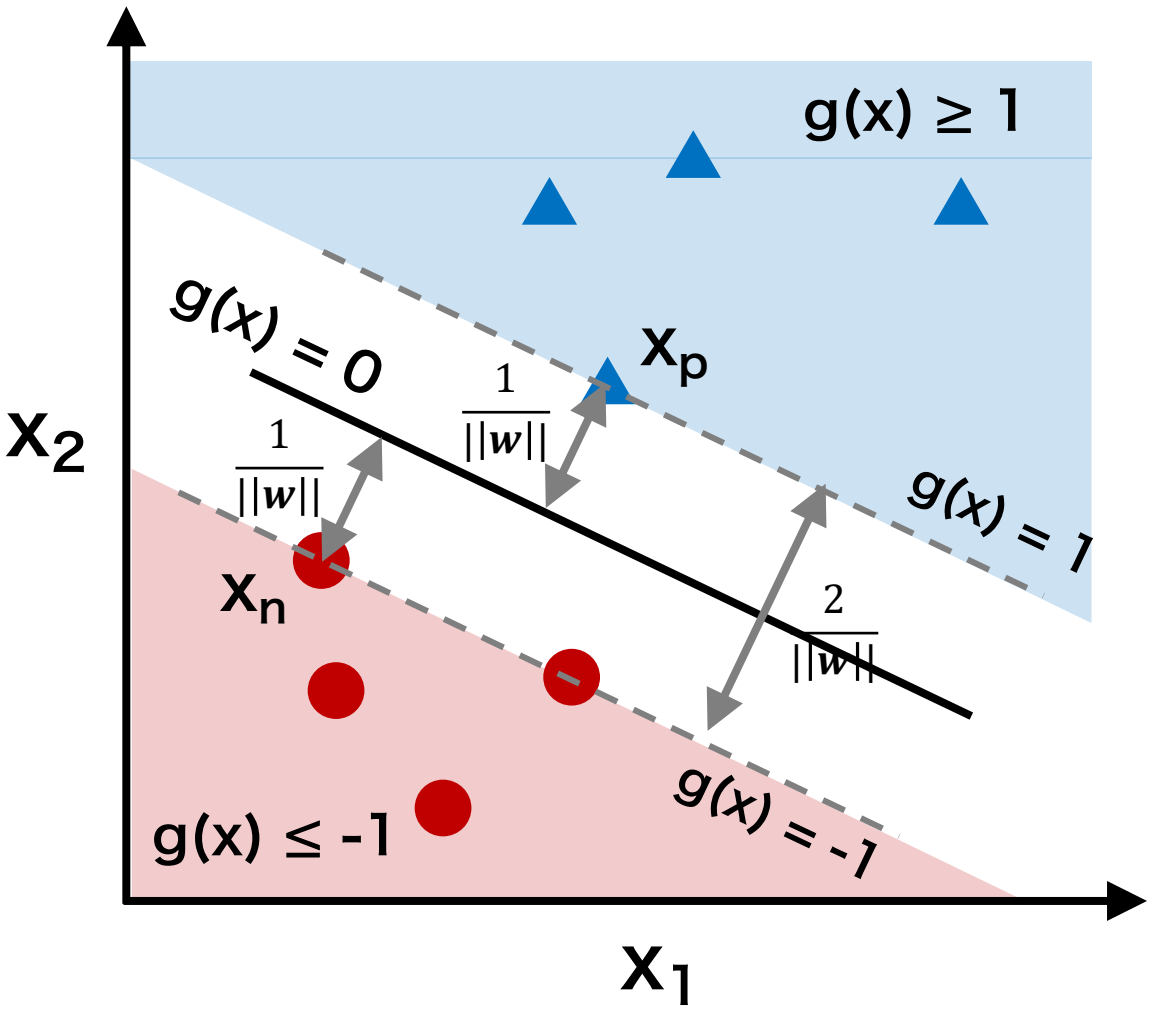
点 x_n から直線 g(x) = 0 までの距離は

$$\frac{|g(\boldsymbol{x}_n)|}{||\boldsymbol{w}||} = \frac{|-1|}{||\boldsymbol{w}||} = \frac{1}{||\boldsymbol{w}||}$$

よって、点 x_p と点 x_n の距離は

$$\frac{1}{||\boldsymbol{w}||} + \frac{1}{||\boldsymbol{w}||} = \frac{2}{||\boldsymbol{w}||}$$

SVM / hard-margin



2 つのクラスのデータを最もよく分離するために、2/||w|| を最大化すればよい。

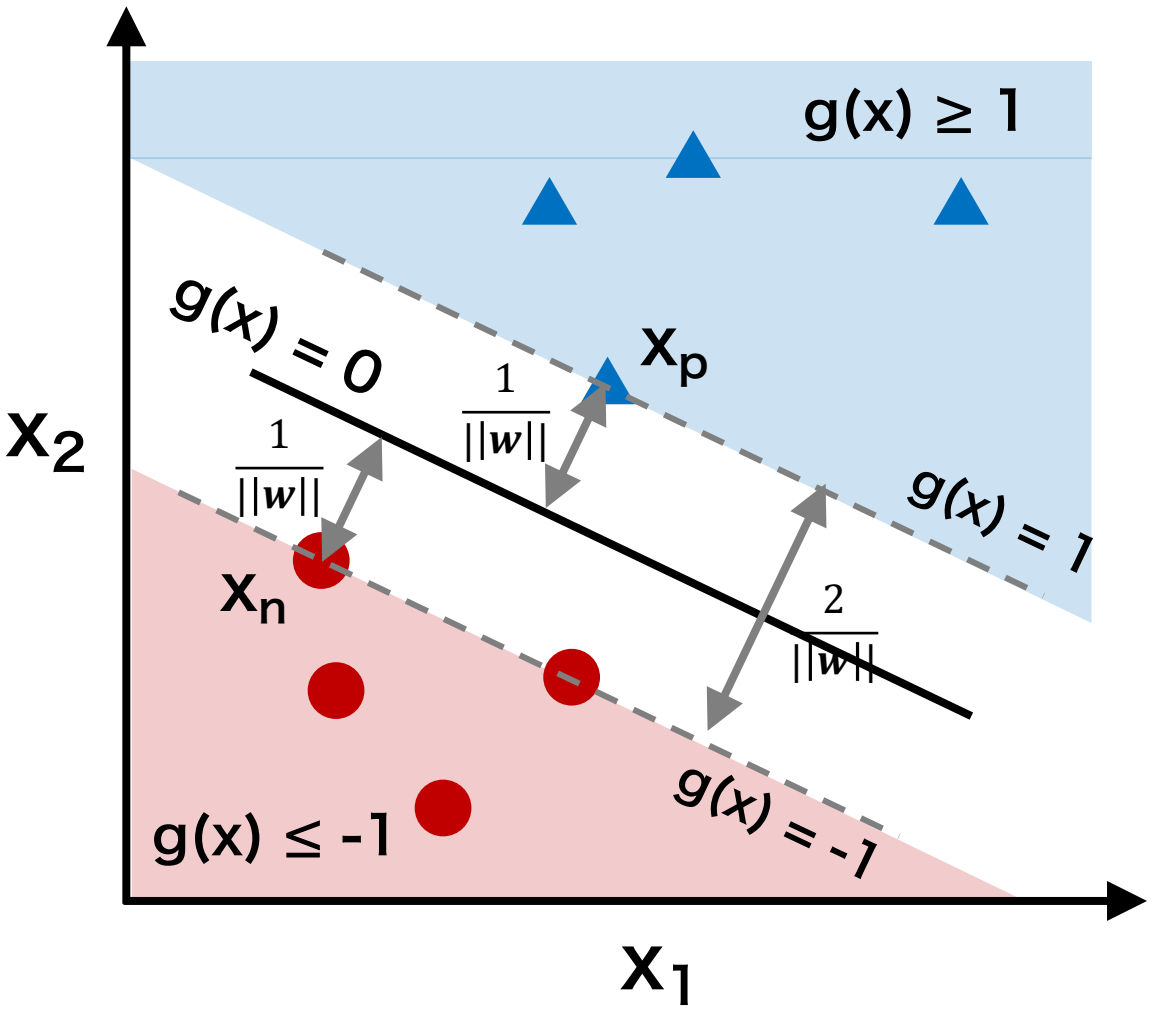
$$\underset{w}{\operatorname{argmax}} \frac{2}{||w||}$$

2/||w|| の最大化は、$||w||^2/2$ の最小化と置き換えても一般性を失わないので、上式は次式と同じである。

$$\underset{w}{\operatorname{argmin}} \frac{1}{2}||w||^2$$

ただし、g(x) に関して、positive データのとき g(x)≥1、negative データのとき g(x)≤-1 を満たす必要がある。

SVM / hard-margin



positive データに関しては、

$$(w_0 + \boldsymbol{w}^t \boldsymbol{x}) \geq 1$$

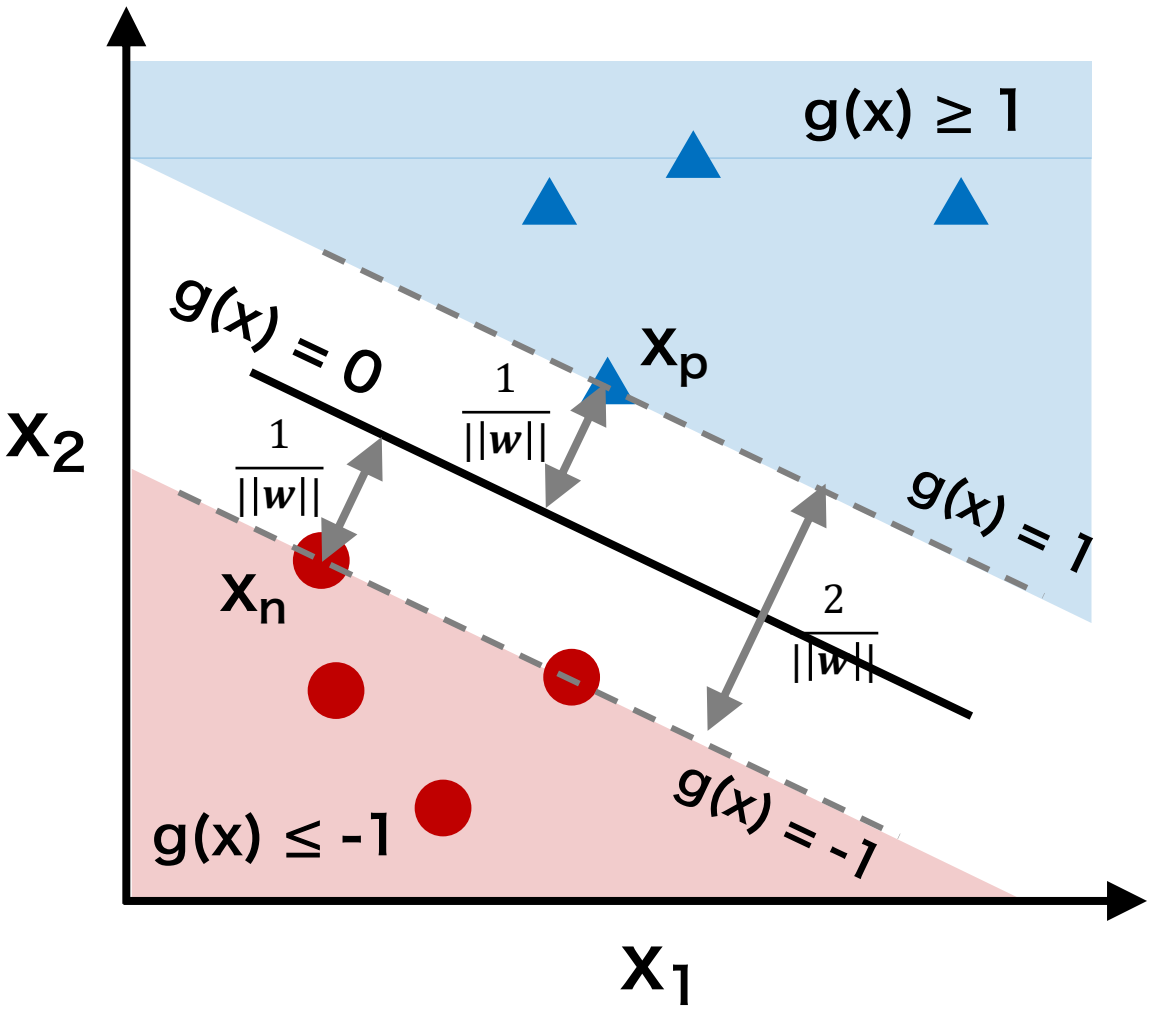
negative データに関しては、

$$(w_0 + \boldsymbol{w}^t \boldsymbol{x}) \leq -1$$

このとき positive のとき y = 1、negative のとき y = -1 となる y を考えると、両式は次のように 1 つの式で表すことができる。

$$y_i(w_0 + \boldsymbol{w}^t \boldsymbol{x}_i) \geq 1$$

SVM / hard-margin



以上により、SVM ハードマージン法において、次の式に基づいて決定境界（超平面）を推定する。

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{w}||^2$$

ただし、次の制約条件のもとで最小化を行う。

$$y_i(w_0 + \boldsymbol{w}^t \boldsymbol{x}_i) \geq 1$$

※ 距離計算を行うので、特徴量同士が比較可能なスケールに変換する必要がある。

SVM / hard-margin

ラグランジュの未定乗数法を利用して最適な w を求める。このとき ||w||²/2 の最小化問題は、次式で示す L_p の最小化問題に帰着できる。

$$L_p(\boldsymbol{w}, w_0, \boldsymbol{\lambda}) = \frac{1}{2}||\boldsymbol{w}||^2 - \sum_{i=1}^{n} \lambda_i (y_i(w_0 + \boldsymbol{w}^t \boldsymbol{x}_i) - 1)$$

$$\text{subject to} \quad \lambda_i \geq 0 \quad (i = 1, 2, 3, \cdots n)$$

L_p 関数は複雑で解くのが困難である。その双対問題を考える。ある制約条件のもとで L_p を最小化する問題は、双対定理により、別の制約条件のもとで L_D を最大化する問題に変換できる。

$$\underset{\boldsymbol{x}}{\operatorname{argmin}}\, \boldsymbol{c}^t \boldsymbol{x} \qquad \text{subject to} \qquad \boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{b}\ ,\ \boldsymbol{x} \geq 0$$

$$\underset{\boldsymbol{y}}{\operatorname{argmin}}\, \boldsymbol{b}^t \boldsymbol{y} \qquad \text{subject to} \qquad \boldsymbol{A}^t \boldsymbol{x} \leq \boldsymbol{c}\ ,\ \boldsymbol{y} \geq 0$$

SVM / hard-margin

L_p の双対問題を求めるために、L_p を w および w_0 で偏微分して 0 とおく。

$$\frac{\partial L_p}{\partial \boldsymbol{w}} = w - \sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x}_i = 0$$

$$\frac{\partial L_p}{\partial w_0} = \sum_{i=1}^{n} \lambda_i y_i = 0$$

偏微分の結果を L_p に代入すると次式が得られる。

$$L_D(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i=j}^{n} \lambda_i \lambda_j t_i t_j \boldsymbol{x}_i^t \boldsymbol{x}_j$$

SVM / hard-margin

以上により、L_p の最小化問題は、次のように L_D の最大化問題に帰着して解くことができる。

$$\underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \left\{ \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i=j}^{n} \lambda_i \lambda_j t_i t_j \boldsymbol{x}_i^t \boldsymbol{x}_j \right\}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \lambda_i y_i = 0 \,, \quad \lambda_i \geq 0 \quad (i = 1, 2, 3, \cdots n)$$

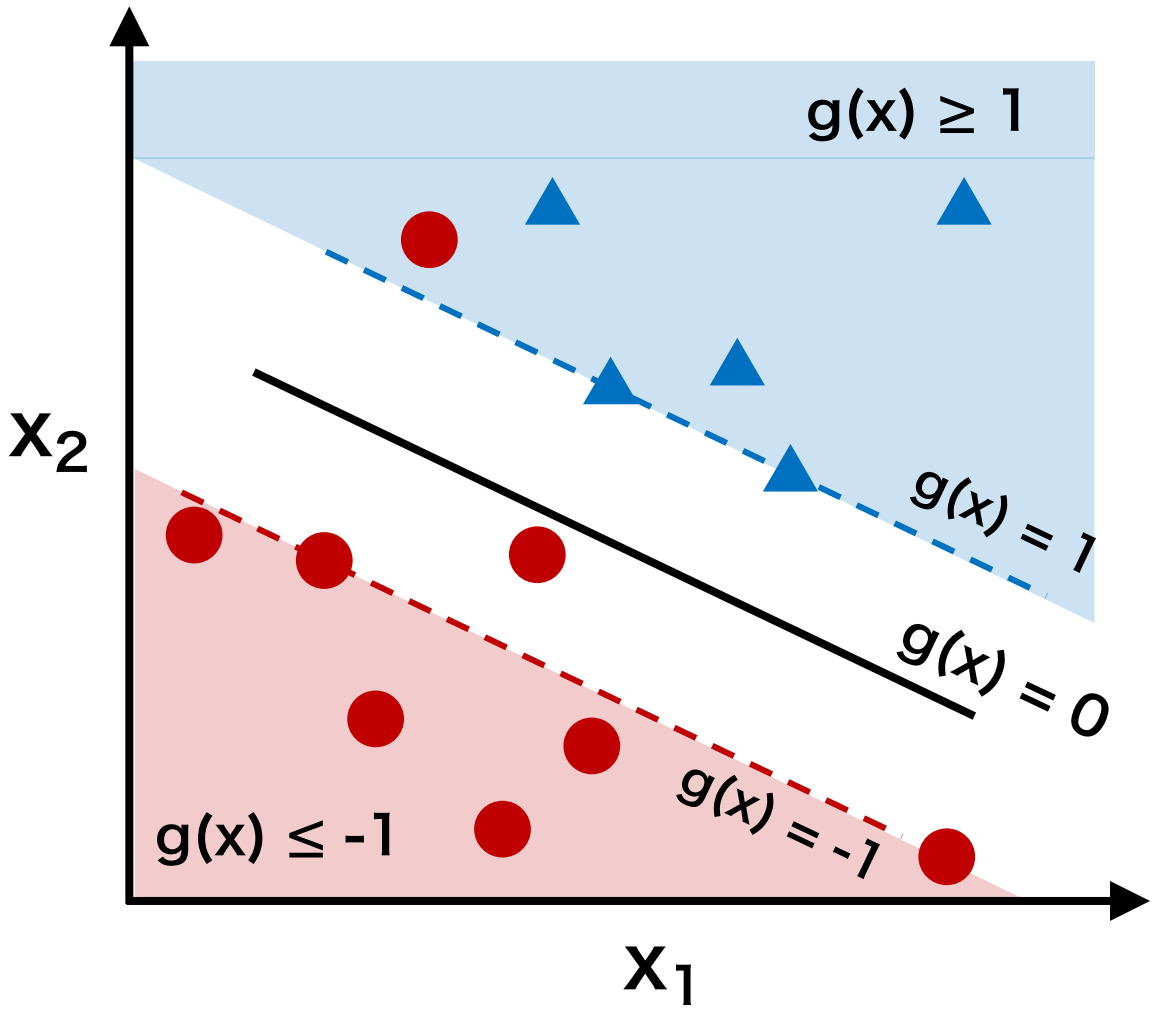
L_D の最大化問題を解くアルゴリズムとして逐次最小問題最適化法（SOM）が考案されている。SOM は、λ_i および λ_j を選び、両者の値を変化させることを繰り返すことで、L_D の最大値を探索していくアルゴリズムである。$\boldsymbol{\lambda}$ が求まると、前式から w も求まる。

$$\boldsymbol{w} = \sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x}_i$$

SVM

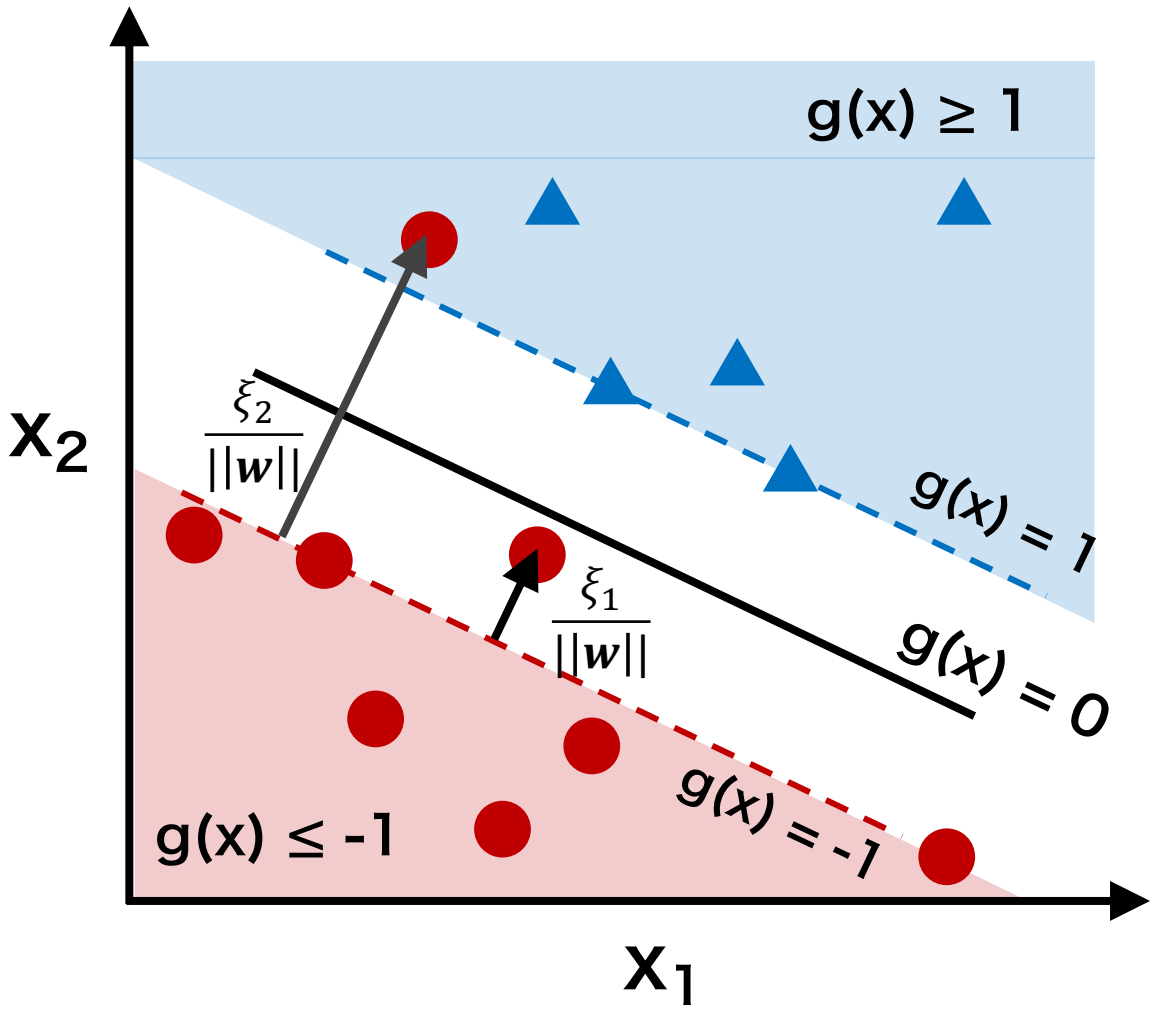
- ハードマージン
- ソフトマージン
- カーネルトリック
- SVR

SVM / soft-margin



マージンを超えたデータに対してペナルティを設けることによって、線形分離でないデータにもSVMを適用することができる。

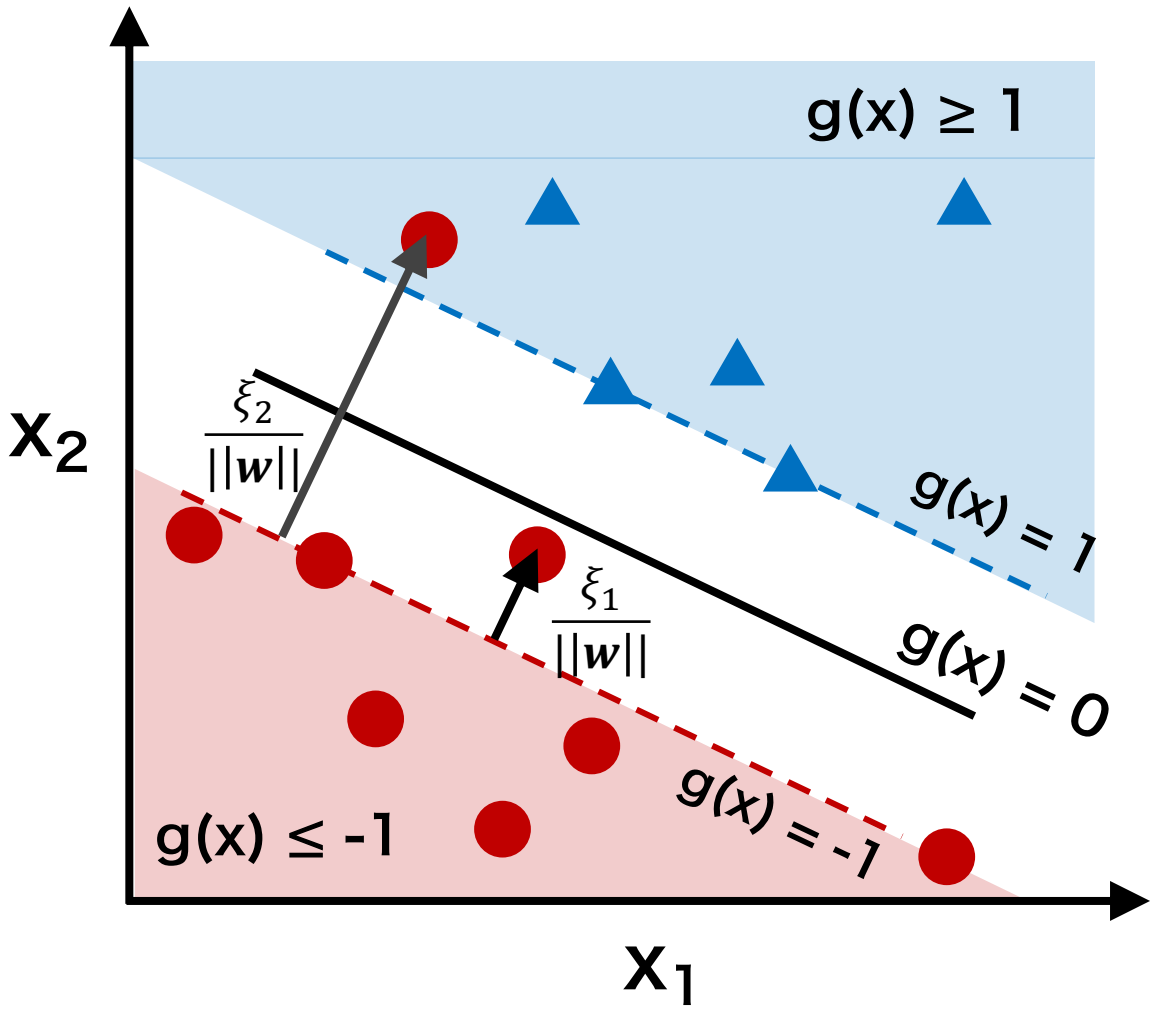
SVM / soft-margin



マージンを超えたデータに対してペナルティを設けることによって、線形分離でないデータにもSVMを適用することができる。

$\frac{\xi_i}{\|\|w\|\|} = 0$	マージンの内側にあり、正しく分類されたサンプル。
$0 < \frac{\xi_i}{\|\|w\|\|} < 1$	マージンと超平面の間にあるサンプル。
$1 < \frac{\xi_i}{\|\|w\|\|}$	超平面を超えて反対側にある誤分類されたサンプル。

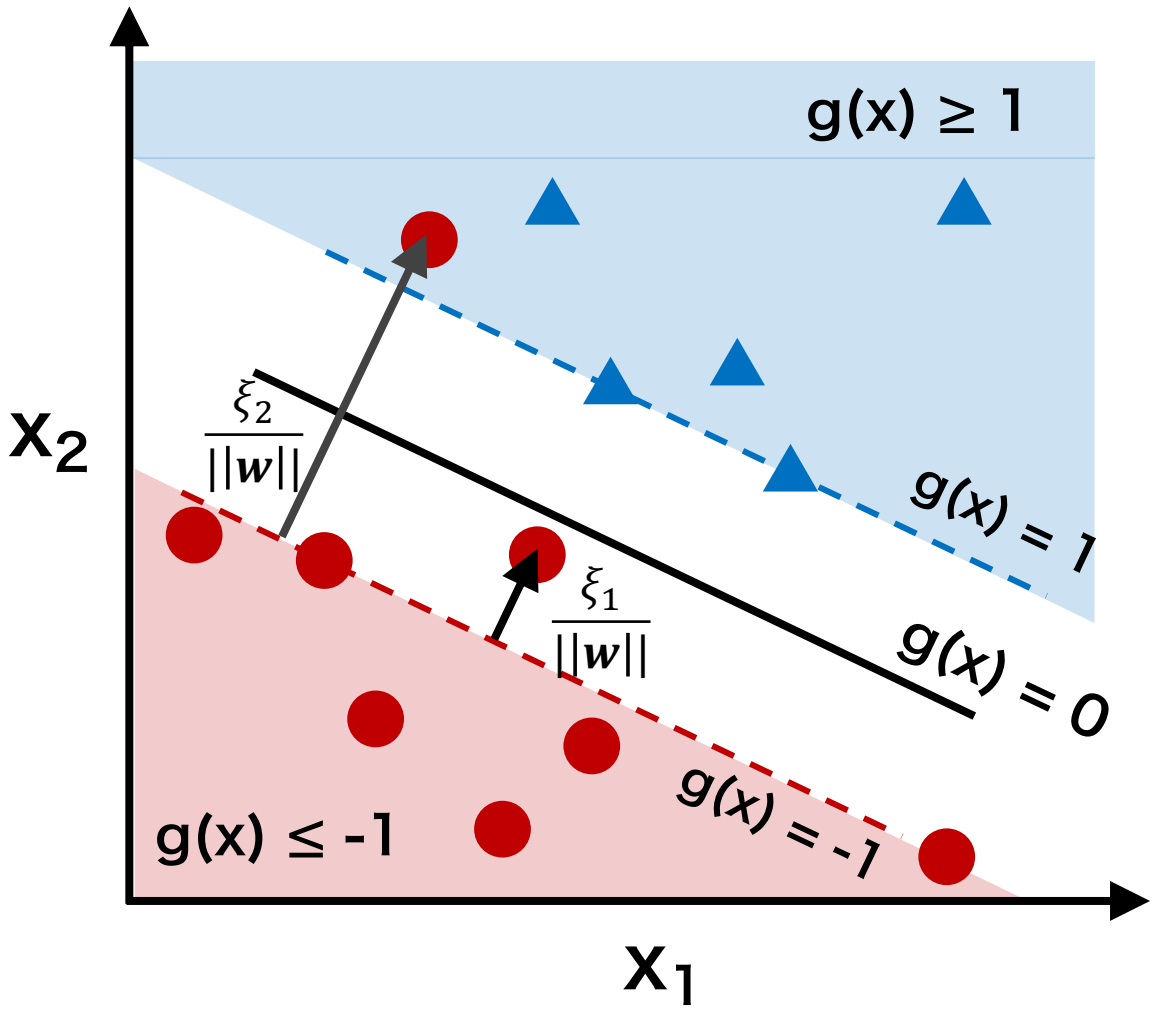
SVM / soft-margin



超平面を超えて反対側に誤分類されるサンプルを、ある程度 K 以下に抑えたい場合は、

$$\sum_{i=1}^{n} \frac{\xi_i}{||\boldsymbol{w}||} \leq K$$

SVM / soft-margin



以上により、SVM ハードマージン法において、次の式に基づいて決定境界（超平面）を推定する。

$$\underset{w}{\operatorname{argmin}} \frac{1}{2}||\boldsymbol{w}||^2$$

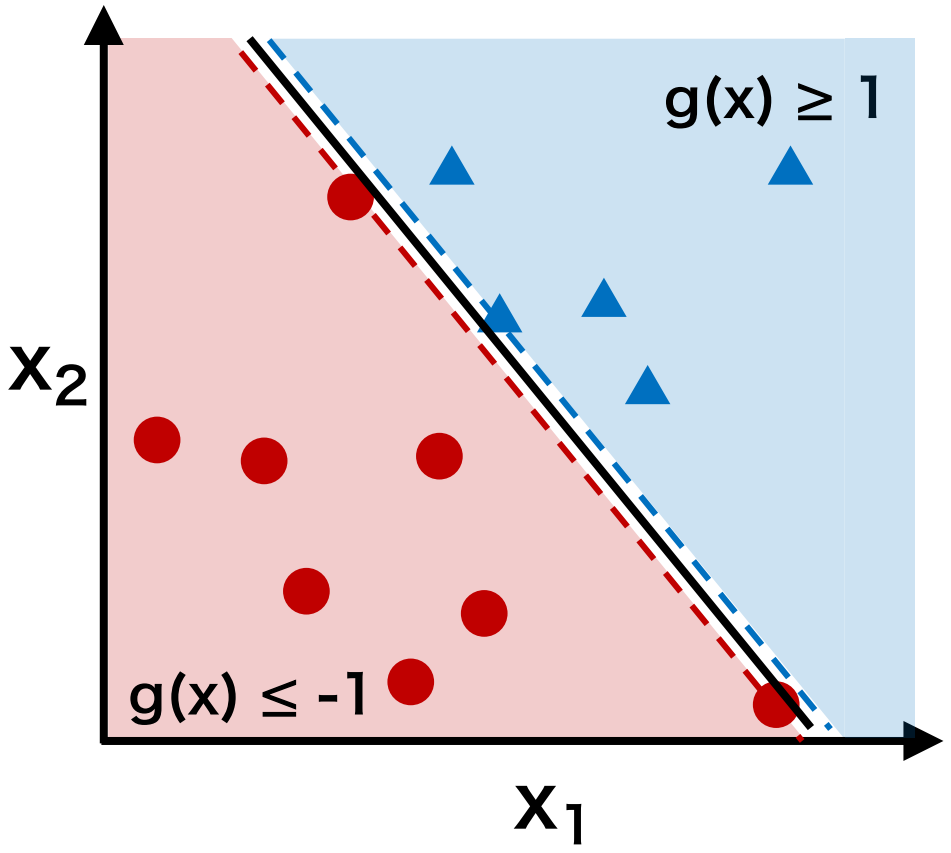
ただし、次の制約条件のもとで最小化を行う。

$$\sum_{i=1}^{n} \frac{\xi_i}{||\boldsymbol{w}||} \leq K$$

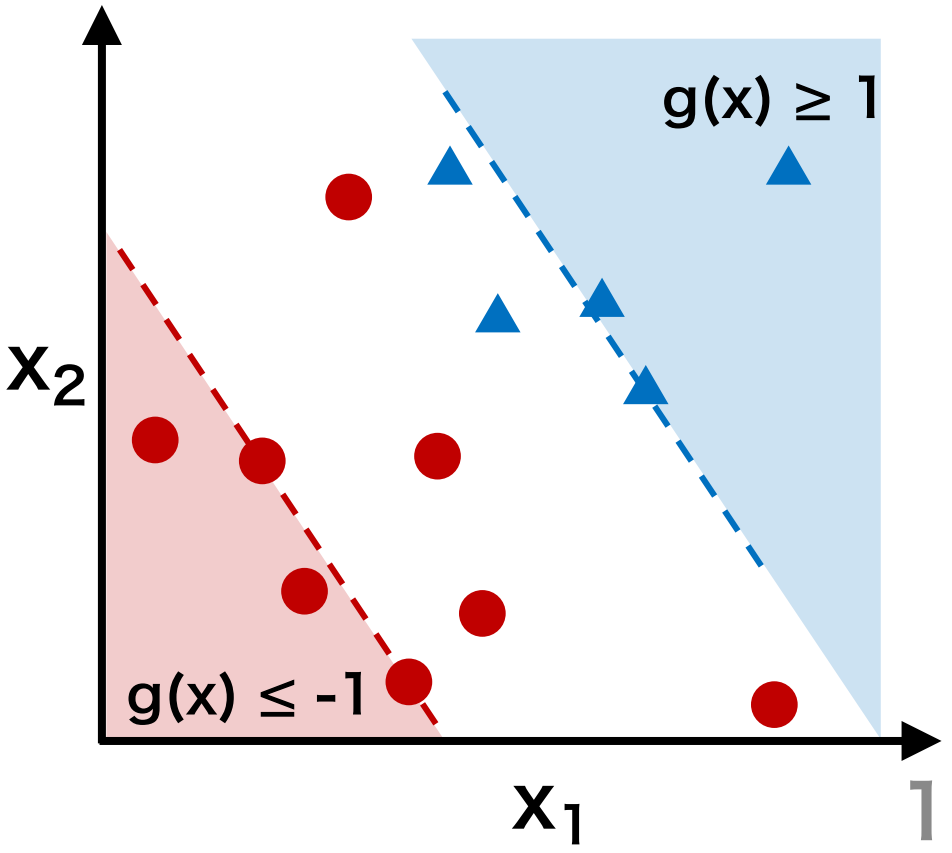
$$\xi_i \geq 0 \qquad i = 1, 2, 3, \cdots, n$$

SVM / soft-margin

K の値を 0 にすることで、誤分類を許容しないハードマージン法と同じになる。下図の場合、訓練データを完全に分離することができるが、未知のデータに対しても同様な性能で予測できる保証はない。



K の値を大きくすることで、より多くの誤分類を許容してしまうモデルが作られる。



SVM / soft-margin

ソフトマージン法の最適化式は次のように書き換えることができる。

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \frac{1}{2} ||\boldsymbol{w}||^2 + \gamma \sum_{i=1}^{n} \xi_i \right\}$$

$$\text{subject to} \quad \xi_i \geq 0 \ , \quad y_i(w_0 + \boldsymbol{w}^t \boldsymbol{x}_i) \geq 1 - \xi_i \qquad (i = 1, 2, 3, \cdots n)$$

ソフトマージン法の最適化式をラグランジュの未定乗数法で解くために定式化を行う。ここでラグランジュ乗数αおよびνを用意すると、ソフトマージン法の最適化式は、次式の最小化問題に帰着できる。

$$L_p(\boldsymbol{w}, w_0, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \frac{1}{2} ||\boldsymbol{w}||^2 + \gamma \sum_{i=1}^{n} \xi_i - \sum_{i=1}^{n} \lambda_i \big(y_i(w_0 + \boldsymbol{w}^t \boldsymbol{x}_i) - (1 - \xi_i) \big) - \sum_{i=1}^{n} \nu_i \xi_i$$

$$\text{subject to} \quad \lambda_i \geq 0 \ , \quad \nu_i \geq 0 \qquad (i = 1, 2, 3, \cdots n)$$

SVM / soft-margin

ハードマージン法と同様に L_p の双対問題 L_D を求めるため、L_p を w、w_0、ξ で偏微分して 0 とおく。

$$\frac{\partial L_p}{\partial \boldsymbol{w}} = \boldsymbol{w} - \sum_{i=1}^{n} \lambda_i y_i \boldsymbol{x}_i = 0$$

$$\frac{\partial L_p}{\partial w_0} = \sum_{i=1}^{n} \lambda_i y_i = 0$$

$$\frac{\partial L_p}{\partial \xi_i} = \gamma - \nu_i - \lambda_i$$

SVM / soft-margin

L_p を w、w_0、ξで偏微分して 0 といて求めた等式を Lp に代入すると次式が得られる。

$$L_D(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i=j}^{n} \lambda_i \lambda_j t_i t_j \boldsymbol{x}_i^t \boldsymbol{x}_j$$

このとき、L_p の最小化問題は次のような最大化問題に置き換えて解くことができる。

$$\underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \left\{ \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i=j}^{n} \lambda_i \lambda_j t_i t_j \boldsymbol{x}_i^t \boldsymbol{x}_j \right\}$$

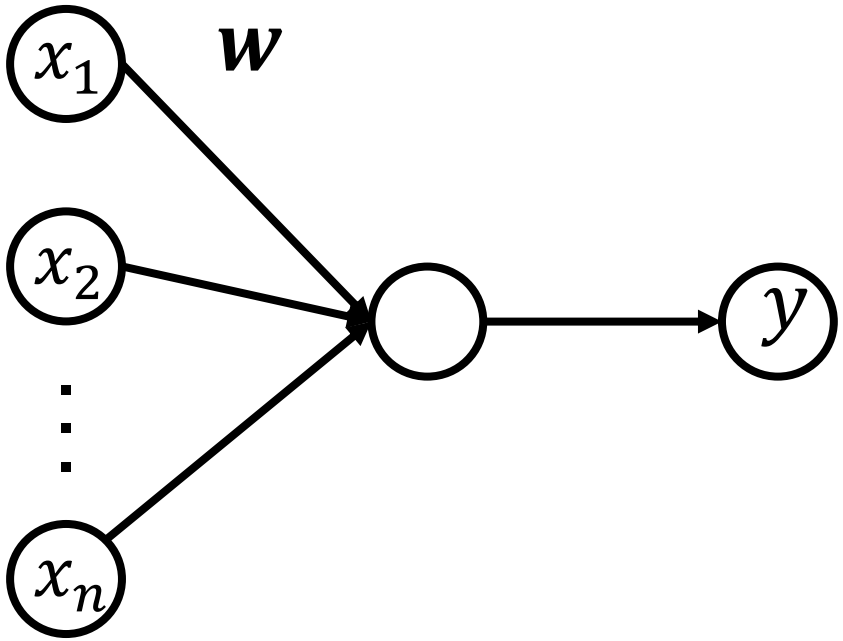
$$\text{subject to} \quad \sum_{i=1}^{n} \lambda_i y_i = 0 \,, \quad \gamma \geq \lambda_i \geq 0 \quad (i = 1, 2, 3, \cdots n)$$

ソフトマージン法で求めた L_D の式は、ハードマージン法で求めた L_D と同じ形をしている。ただし、制約条件に違いが見られる。

SVM

$$y = \mathrm{sign}(\boldsymbol{w}^t \boldsymbol{x} - w_0)$$

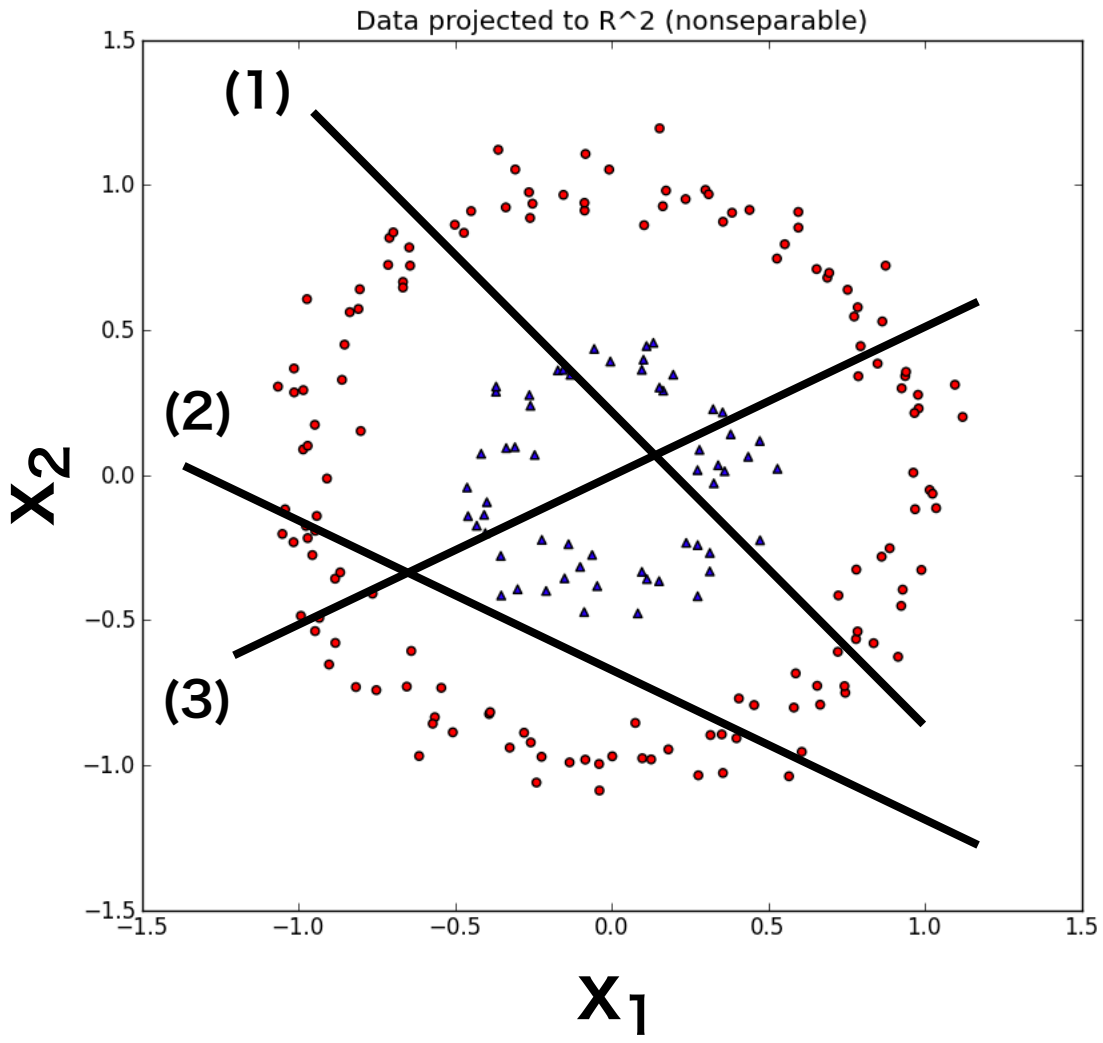
SVM では、特徴量ベクトル x に重みをかけて、さらにバイアス項 w_0 を加えた結果が +1 または -1 として出力している。



SVM

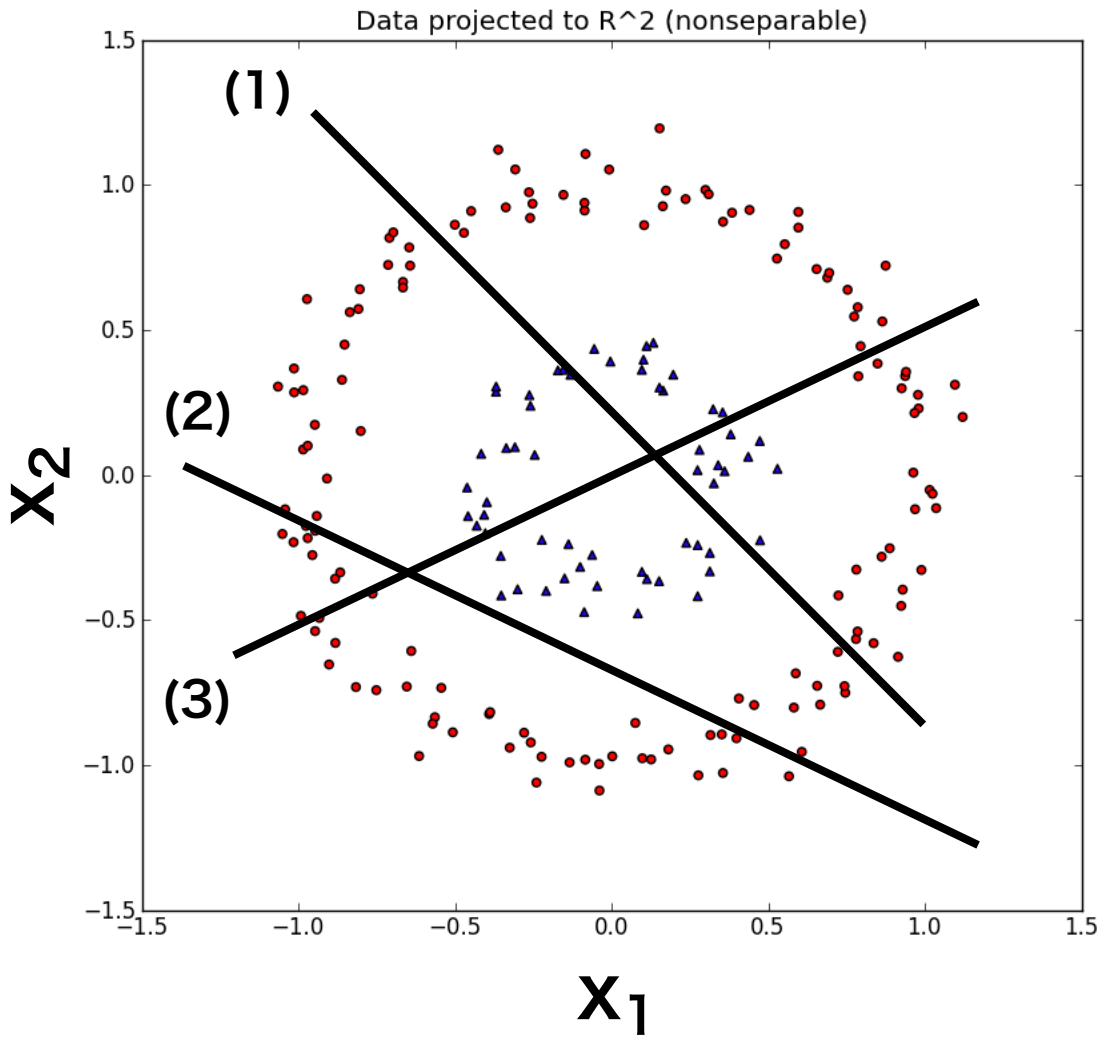
- ハードマージン
- ソフトマージン
- カーネルトリック
- SVR

カーネルトリック



データの分布によってソフトマージン法でも解けない問題もある。例えば、左図のような線型分離不可能なデータをそのまま SVM を適用しても、モデルを構築できない。

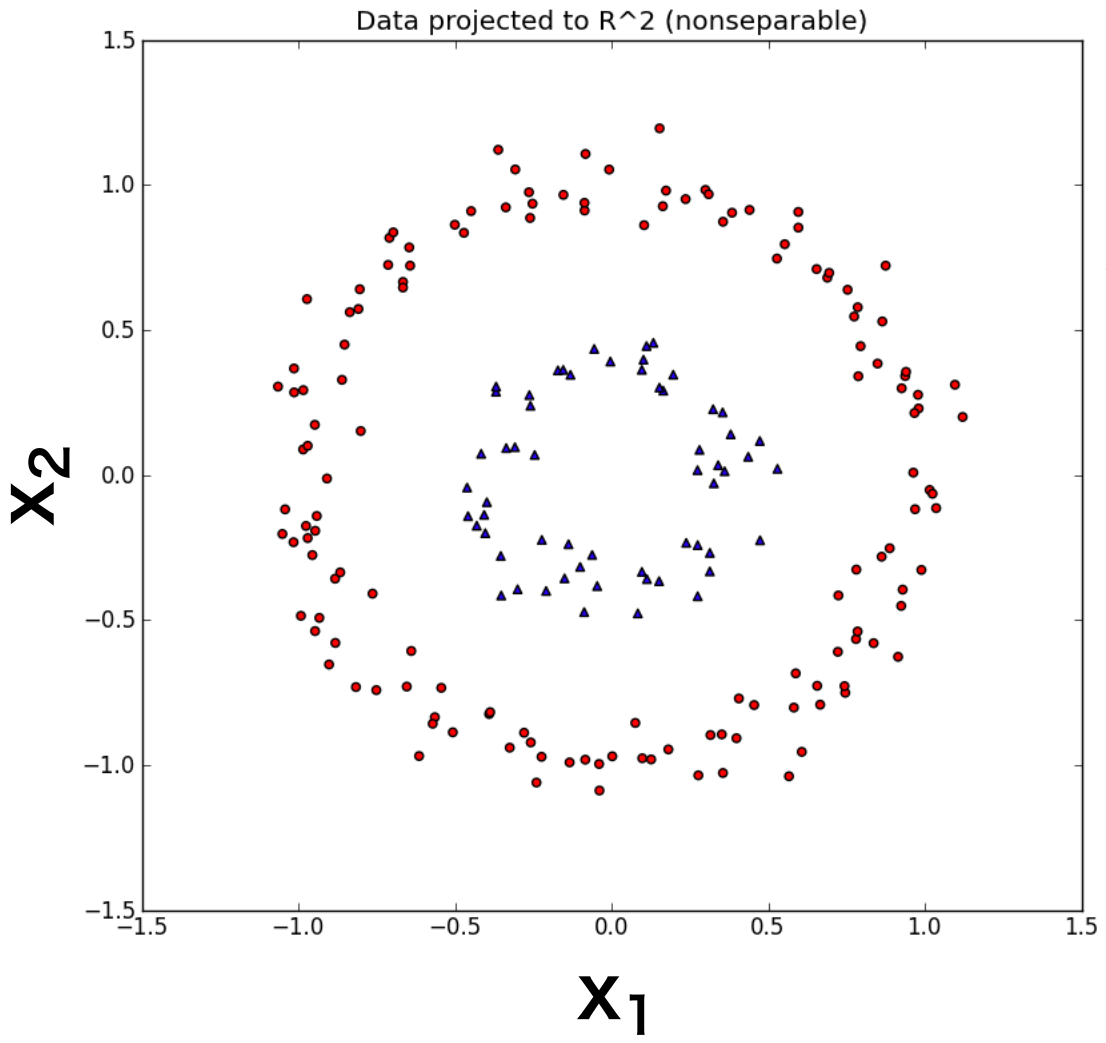
カーネルトリック

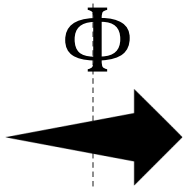


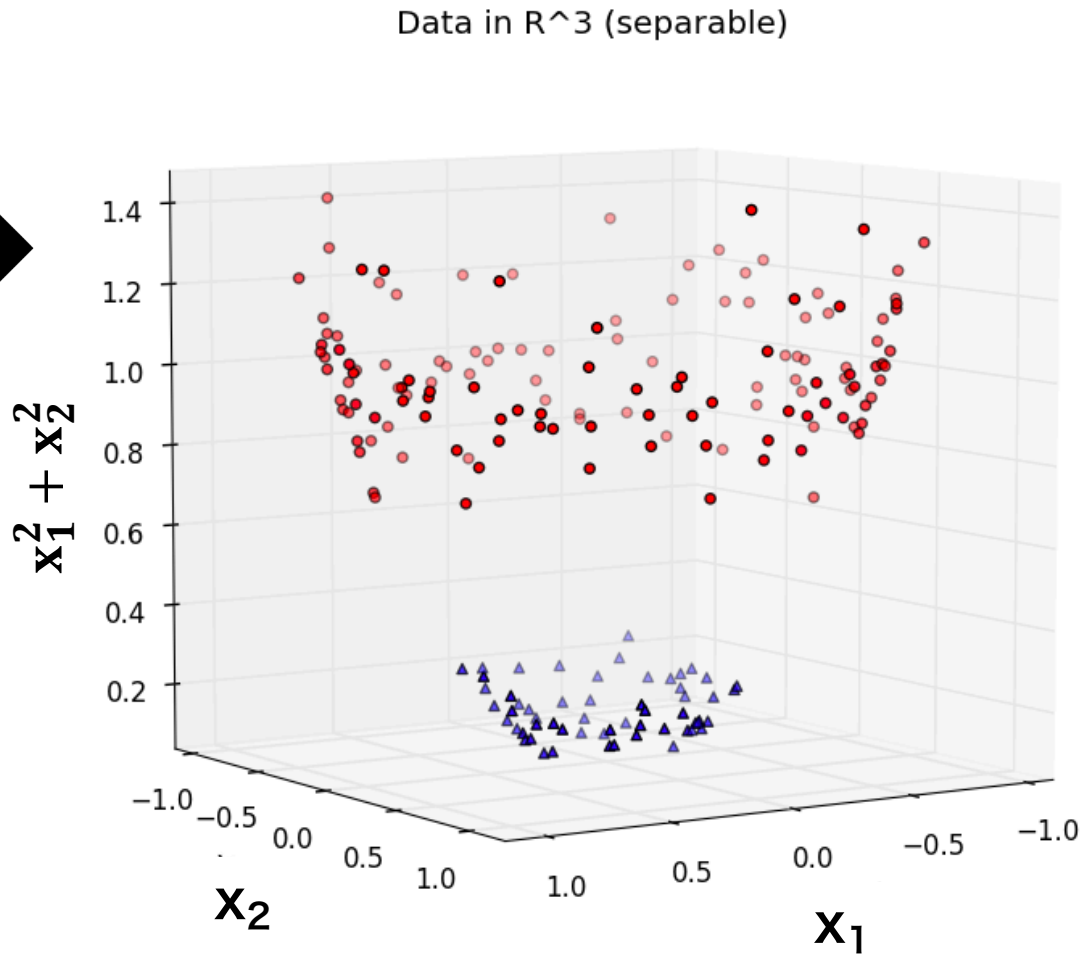
データの分布によってソフトマージン法でも解けない問題もある。例えば、左図のような線型分離不可能なデータをそのまま SVM を適用しても、モデルを構築できない。このような、線型分離不可能なデータを、多次元空間に写像することでによって、線型分離できるようになることがある。

$$(x_1, x_2) \xrightarrow{\Phi} (x_1, x_2, x_1^2 + x_2^2)$$

カーネルトリック


Data projected to R^2 (nonseparable)
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x2
x1


Φ


Data in R^3 (separable)
1.4
1.2
1.0
0.8
0.6
0.4
0.2
x1² + x2²
−1.0
−0.5
0.0
0.5
1.0
x2
1.0
0.5
0.0
−0.5
−1.0
x1

カーネル関数

特徴量を高次元空間に写像して、高次元空間でマージンの最大化を行うことで、難しい問題も解けるようになる。しかし、SVM において最適な w を求めるためには $x^t x$ を計算する必要があるため、高次元空間における計算量が膨大になる。

$$\underset{\boldsymbol{\lambda}}{\operatorname{argmax}} \left\{ \sum_{i=1}^{n} \lambda_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{i=j}^{n} \lambda_i \lambda_j t_i t_j \boldsymbol{x}_i^t \boldsymbol{x}_j \right\}$$

n 個の特徴量を高次元へ射影して、k (>> n) の特徴量を作り出した場合、k 個の特徴量の総当たりで内積を求める必要があり、計算量が莫大になる。

カーネル関数

ある関数 $k(a, b) = (1 + ab)^2$ を考える。この関数に、2 つの特徴量 $x_i = (x_{i1}, x_{i2})$ および $x_j = (x_{j1}, x_{j2})$ を代入して、計算を行う。

$$
\begin{aligned}
& k(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&= \left(1 + \boldsymbol{x}_i \boldsymbol{x}_j\right)^2 \\
&= \left(1 + x_{i1}x_{j1} + x_{i2}x_{j2}\right)^2 \\
&= 1 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2} + x_{i1}^2 x_{j1}^2 + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} \\
&= \left(1 \quad \sqrt{2}x_{i1} \quad \sqrt{2}x_{i2} \quad \sqrt{2}x_{i1}x_{i2} \quad x_{i1}^2 \quad x_{i2}^2\right) \cdot \left(1 \quad \sqrt{2}x_{j1} \quad \sqrt{2}x_{j2} \quad \sqrt{2}x_{j1}x_{j2} \quad x_{j1}^2 \quad x_{j2}^2\right) \\
&= \boldsymbol{z}_i^t \boldsymbol{z}_j
\end{aligned}
$$

← 2 次元の特徴量 x を 6 次元の特徴量空間 z に写像して、その空間で内積を求めたことに等しい。関数 k を使えば、6 次元への写像を行わなくても、内積を求めることが可能。

カーネル関数

カーネル関数を用いることで、より少ない計算量で、高次元化した空間で内積を求めることができる。このようなカーネル関数には、多項式カーネル、RBF カーネル、シグモイドカーネルなどが考案される。

多項式カーネル

$$k(x, x') = (a + xx')^b$$

RBF カーネル

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

シグモイドカーネル

$$k(x, x') = \tanh(axx' + b)$$

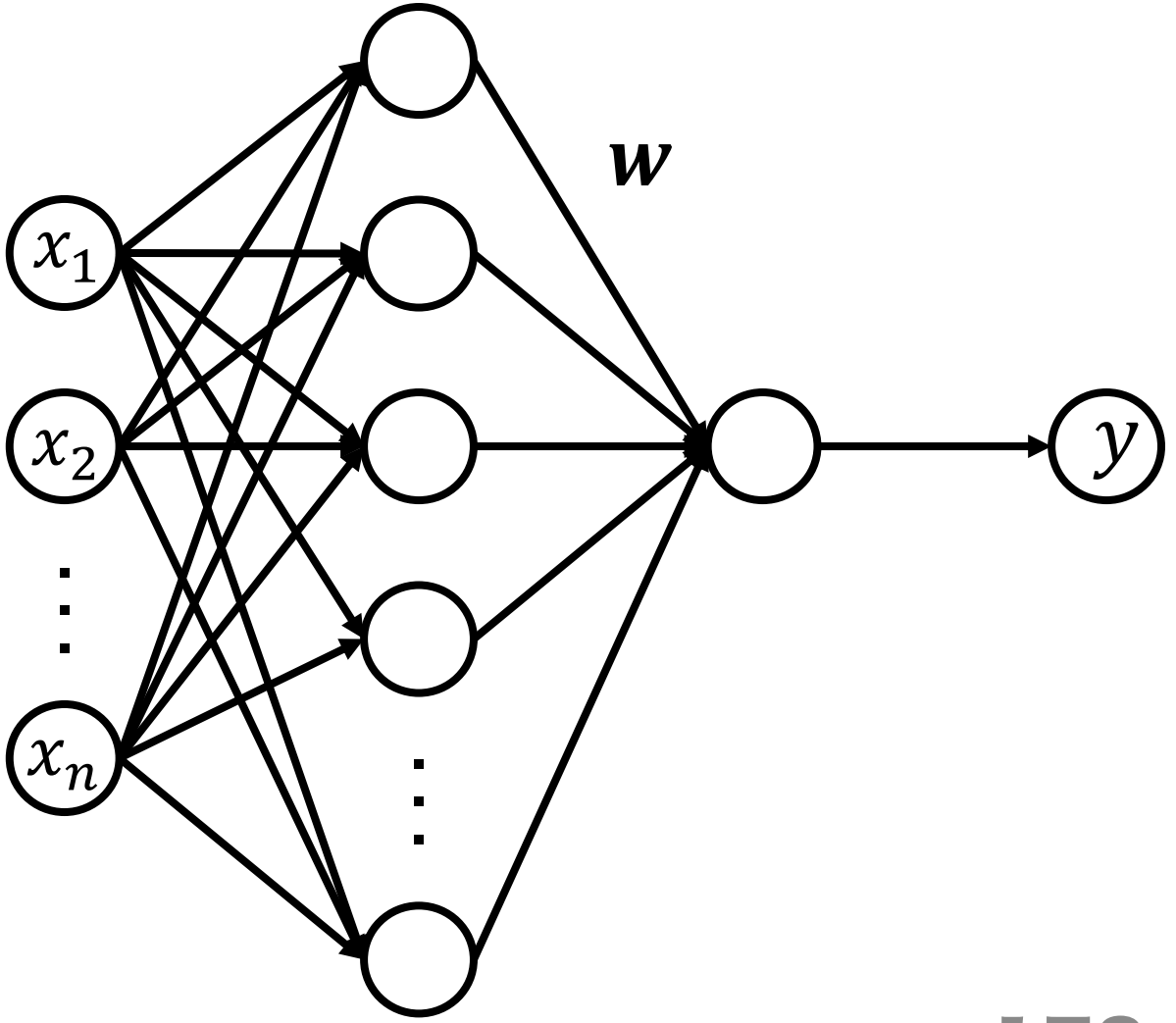
カーネル SVM

$$y = \mathrm{sign}(\boldsymbol{w}^t \phi(\boldsymbol{x}) - w_0)$$

$$= \mathrm{sign}\left(\sum_{i \in S} \lambda_i t_i k(\boldsymbol{x}_i, \boldsymbol{x}) - w_0\right)$$

※ 関数 Lp を w で偏微分した結果 $\boldsymbol{w} = \sum_{i \in S} \lambda_i t_i \boldsymbol{x}_i$ を使用

カーネル SVM では、特徴量ベクトル x を高次元空間に写像し、写像先の特徴量ベクトルに重みをかけて、さらにバイアス項 w_0 を加えた結果を +1 または -1 として出力している。



多クラスの分類

SVM は 2 つのクラスの間にある決定境界を決めて、入力データが決定境界のプラス側にあるのか、それともマイナス側になるのかを分類するアルゴリズムである。分類すべきクラスが 3 つ以上ある場合、次のように対処する方法が挙げられる。

One-vs-the-rest

- 「クラス i」と「それ以外」を分類するモデルを c 個作成する。
  - positive データと negative データの数に偏りが生じる場合があることに注意。
- c 個のモデルの適合度の最も高いクラスを予測結果とする。

One-vs-one

- 「クラス i」と「クラス j」を分類するモデルを作成する。
  - c(c-1)/2 個のモデルが作成される。
  - positive データと negative データの数に偏りが生じる場合があることに注意。
- c(c-1)/2 個の出力から多数決によって予測結果を出力する。

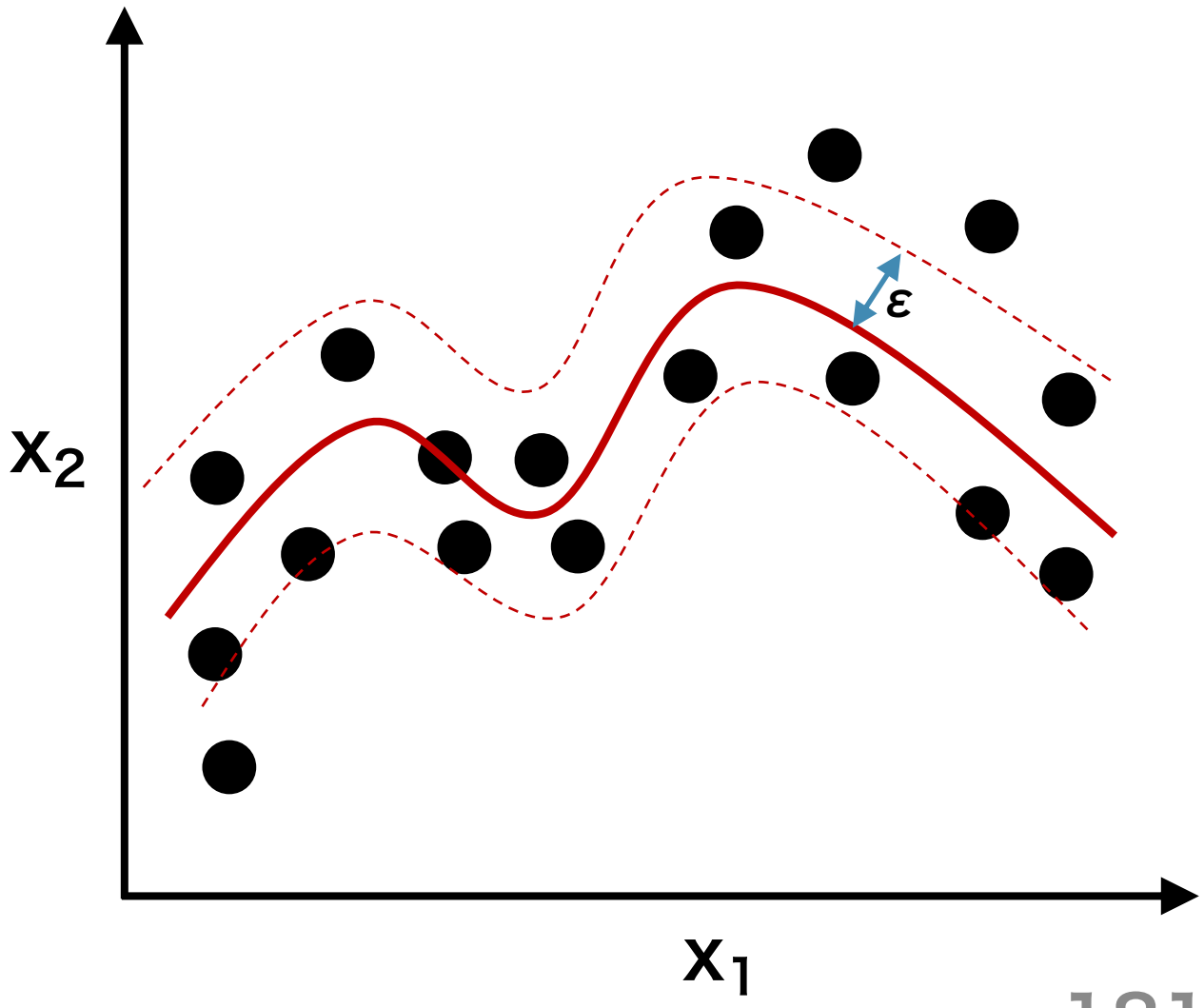
SVM

- ハードマージン
- ソフトマージン
- カーネルトリック
- SVR

SVR

SVR は学習データが回帰曲線の一定範囲内 ε におさまるように回帰曲線を学習する。なお、学習データが回帰曲線から距離 ε を超えた場合、ペナルティを与えて学習を進める。

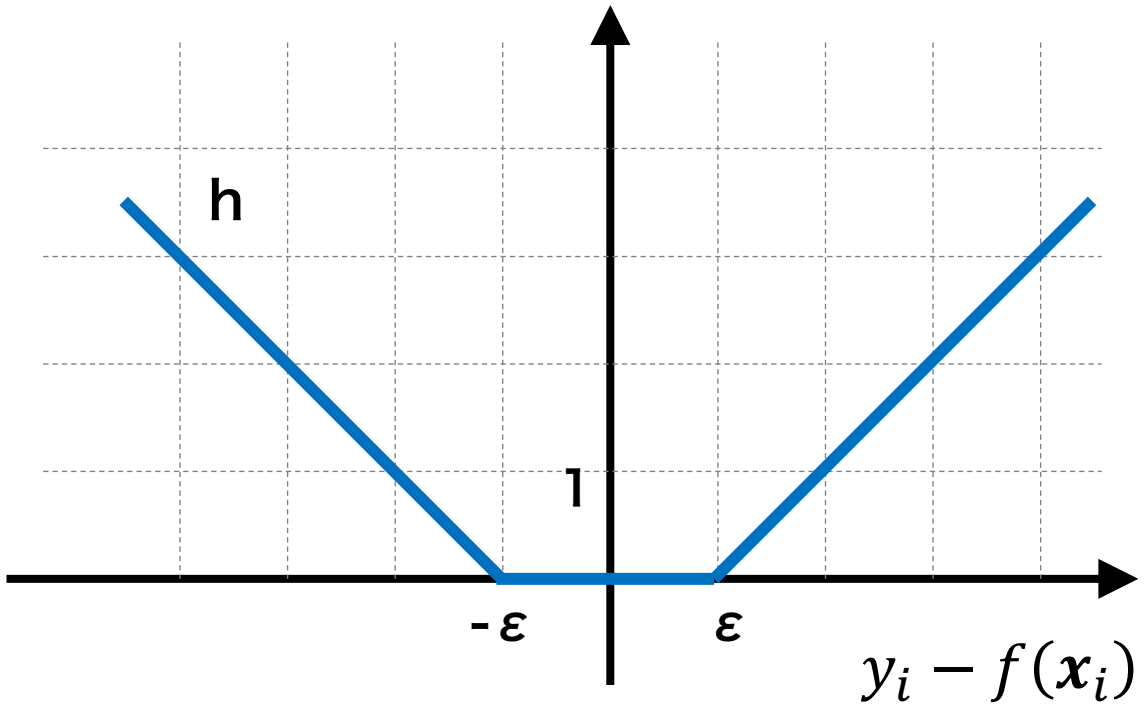
SVR で計算された回帰曲線には数理モデルを仮定していない。このため、学習データの範囲外にある未知データに対する予測は保証されない。



誤差関数

誤差関数は次のように、学習データと回帰曲線の距離がεを超えたら、超えた分だけペナルティを与える。

$$h\left(y_i - f(\boldsymbol{x}_i)\right) = \max\{0, |y_i - f(\boldsymbol{x}_i)| - \varepsilon\}$$



SVR

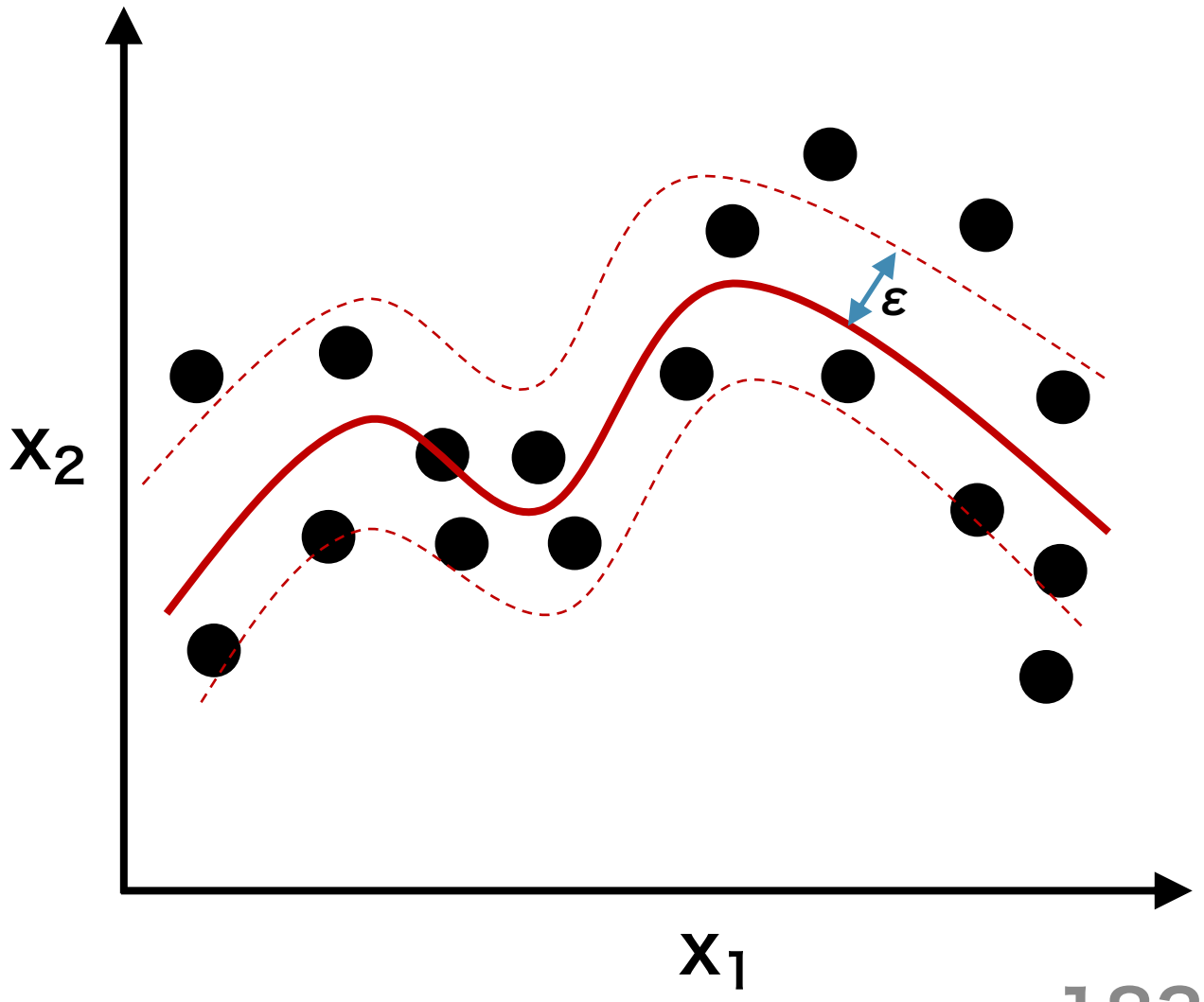
SVR は SVM と同様にマージンの最大化問題に帰着できる。

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \left\{ \frac{1}{2} ||\boldsymbol{w}||^2 + \gamma \sum_{i=1}^{n} h\left(y_i - f(\boldsymbol{x}_i)\right) \right\}$$

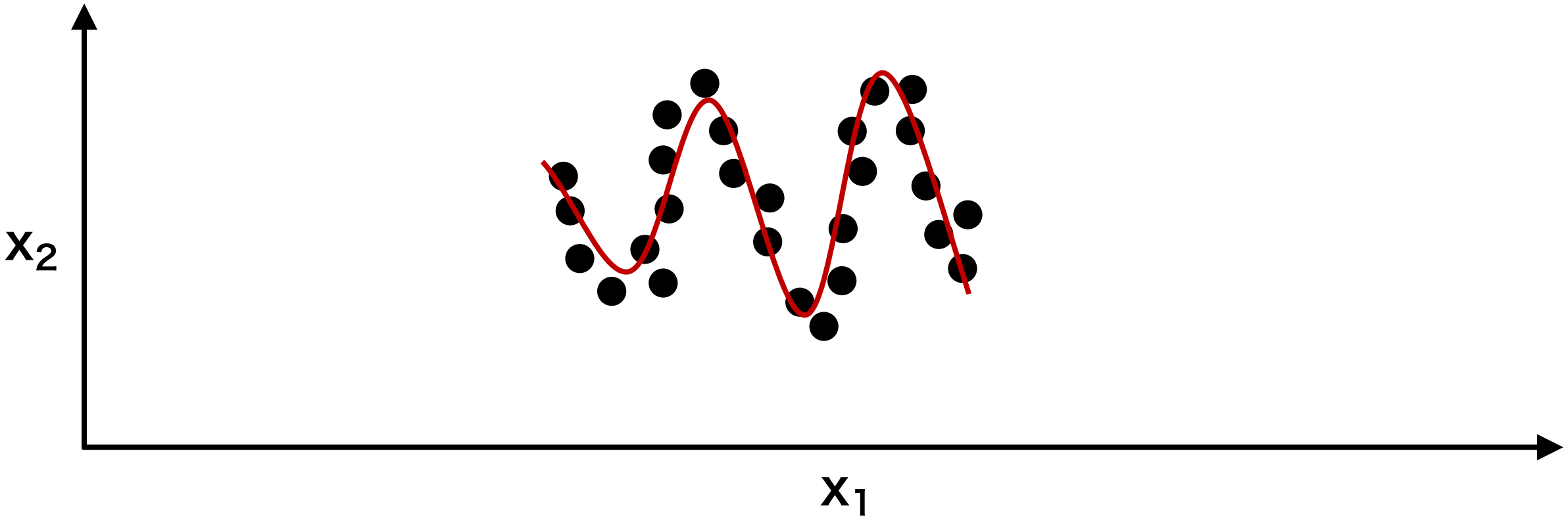
マージンの最大化
（重みの最小化）

誤差の最小化

この項は、正則化項として作用して過学習を防ぐ。

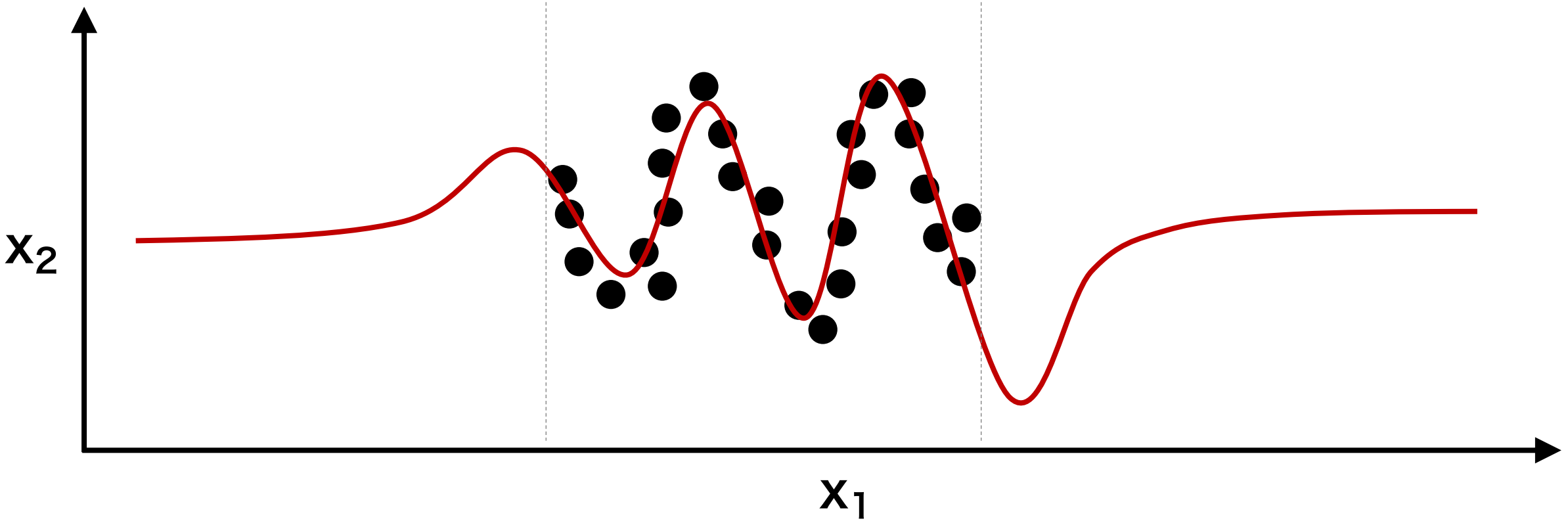


SVR



SVR

学習データの範囲で高精度な予測（フィッティング）結果が得られている。



決定木

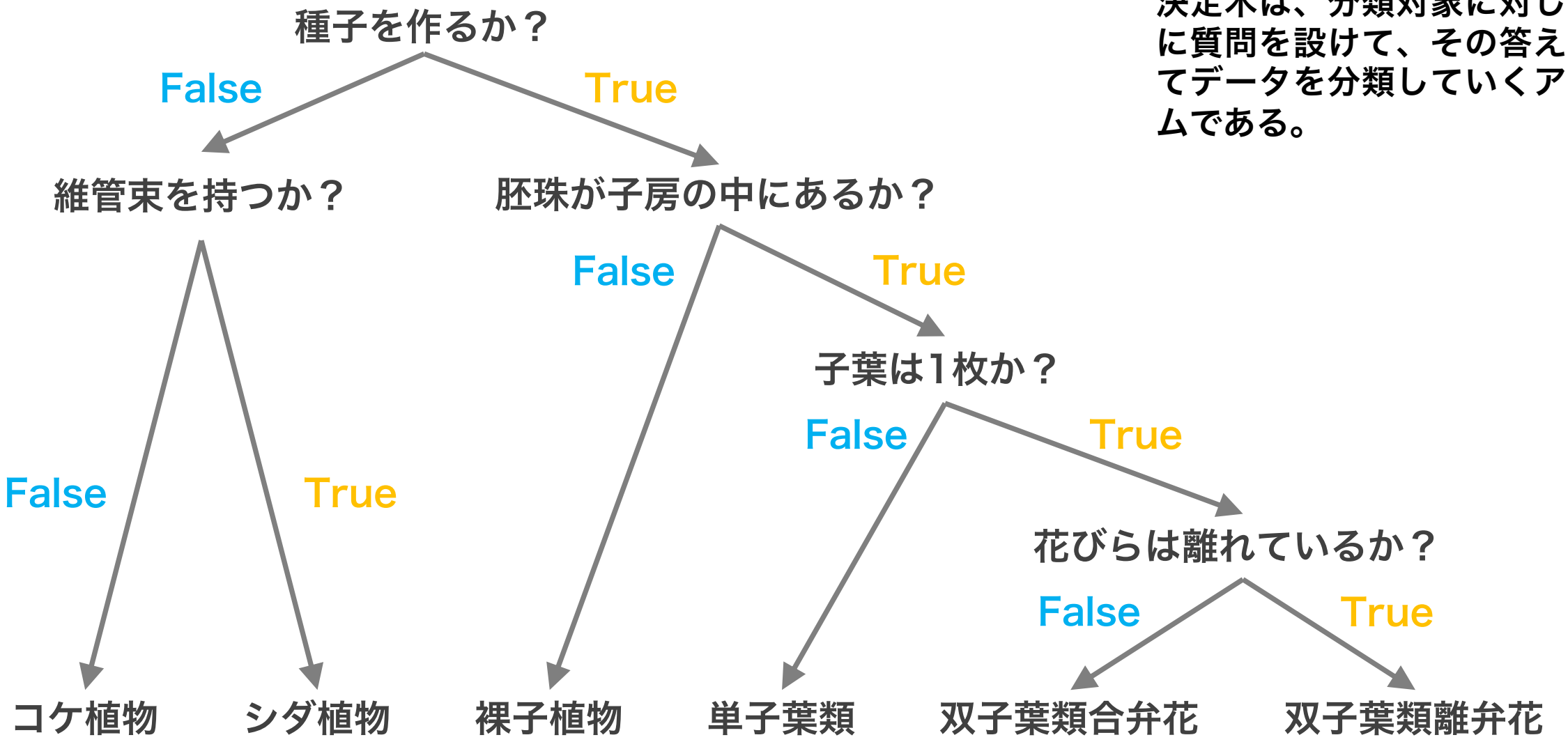
- 決定木
- ランダムフォレスト
- 回帰木

決定木

- 決定木
- ランダムフォレスト
- 回帰木

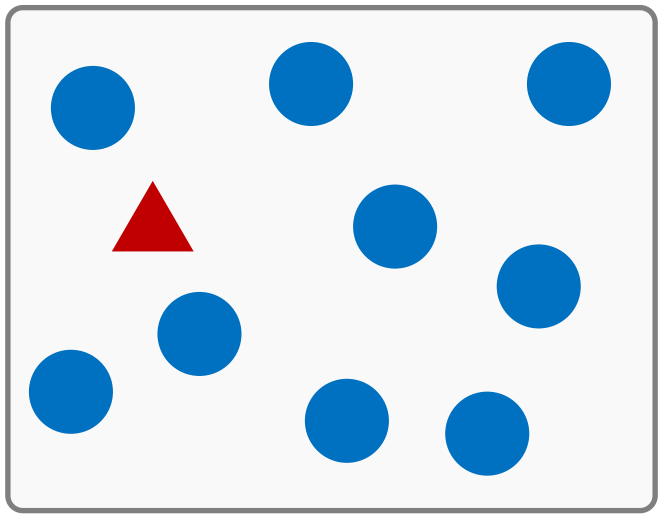
決定木

決定木は、分類対象に対して、次々に質問を設けて、その答えに基づいてデータを分類していくアルゴリズムである。

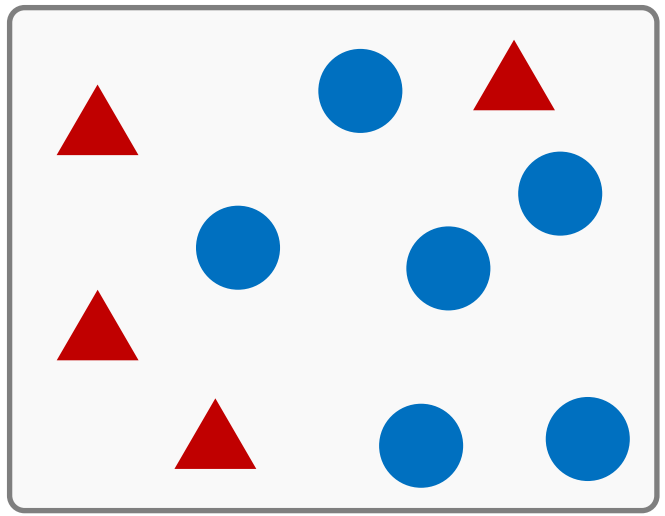


不純度

不純度小



不純度大



不純度は、１つのノードに含まれるデータの種類の乱雑さを表す指標である。その指標として、次のようなものが使われている。

エントロピー

$$I_H(t) = -\sum_{j=1}^{C} p(j|t) \log_2 p(j|t)$$

ジニー不純度

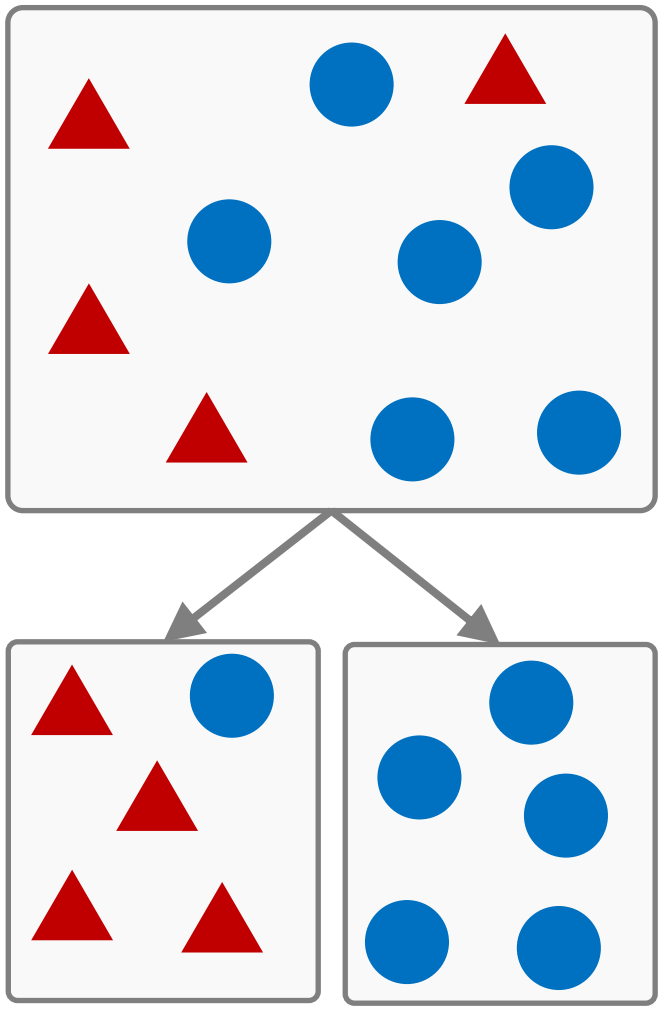
$$I_G(t) = \sum_{j=1}^{C} p(j|t)(1 - p(j|t))$$

分類誤差

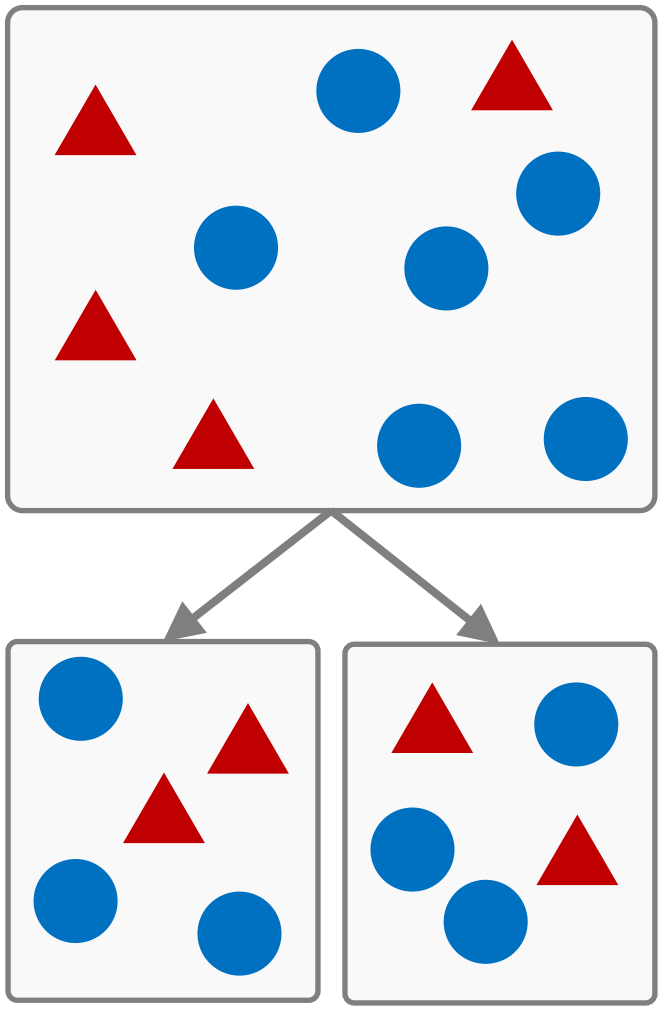
$$I_E(t) = 1 - \max\{p(j|t)\}$$

情報利得

情報利得大



情報利得小

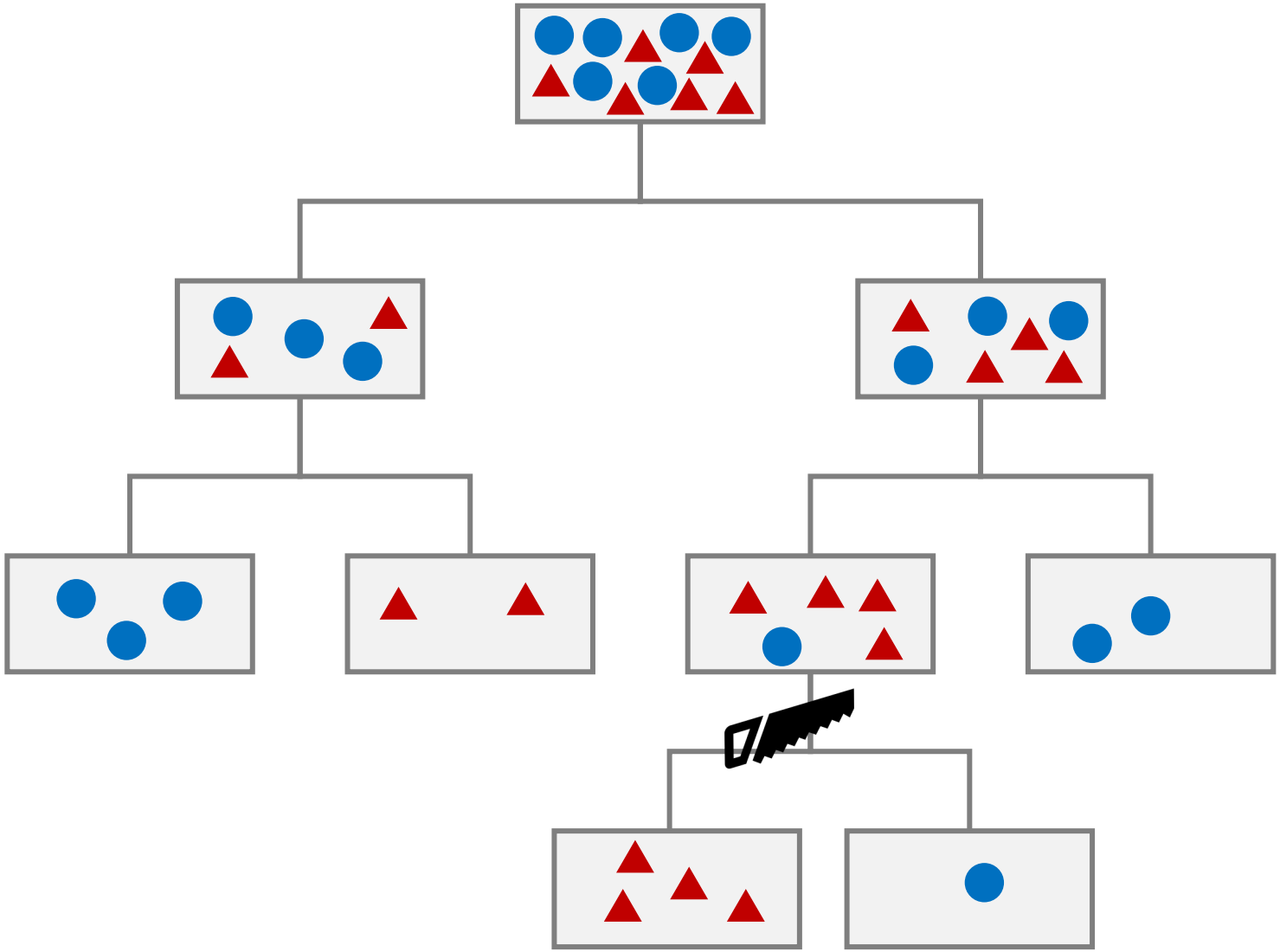


情報利得は、親ノードの不純度と子ノードの不純度の差を表される。あるノード k における不純度を $I(D_k)$ とおくと、情報利得は次のように計算される。ただし、N_k はノード k のサンプルサイズとし、親ノードを p、その子ノードを 1, 2, ..., c とする。

$$IG(D_p) = I(D_p) - \sum_{j=1}^{c} \frac{N_j}{N_p} I(D_j)$$

この式より、親ノードの不純度が高く、子ノードの不純度が小さいとき、情報利得が大きくなる。決定木は、情報利得が最も大きくなるように、データを分類している。

決定木

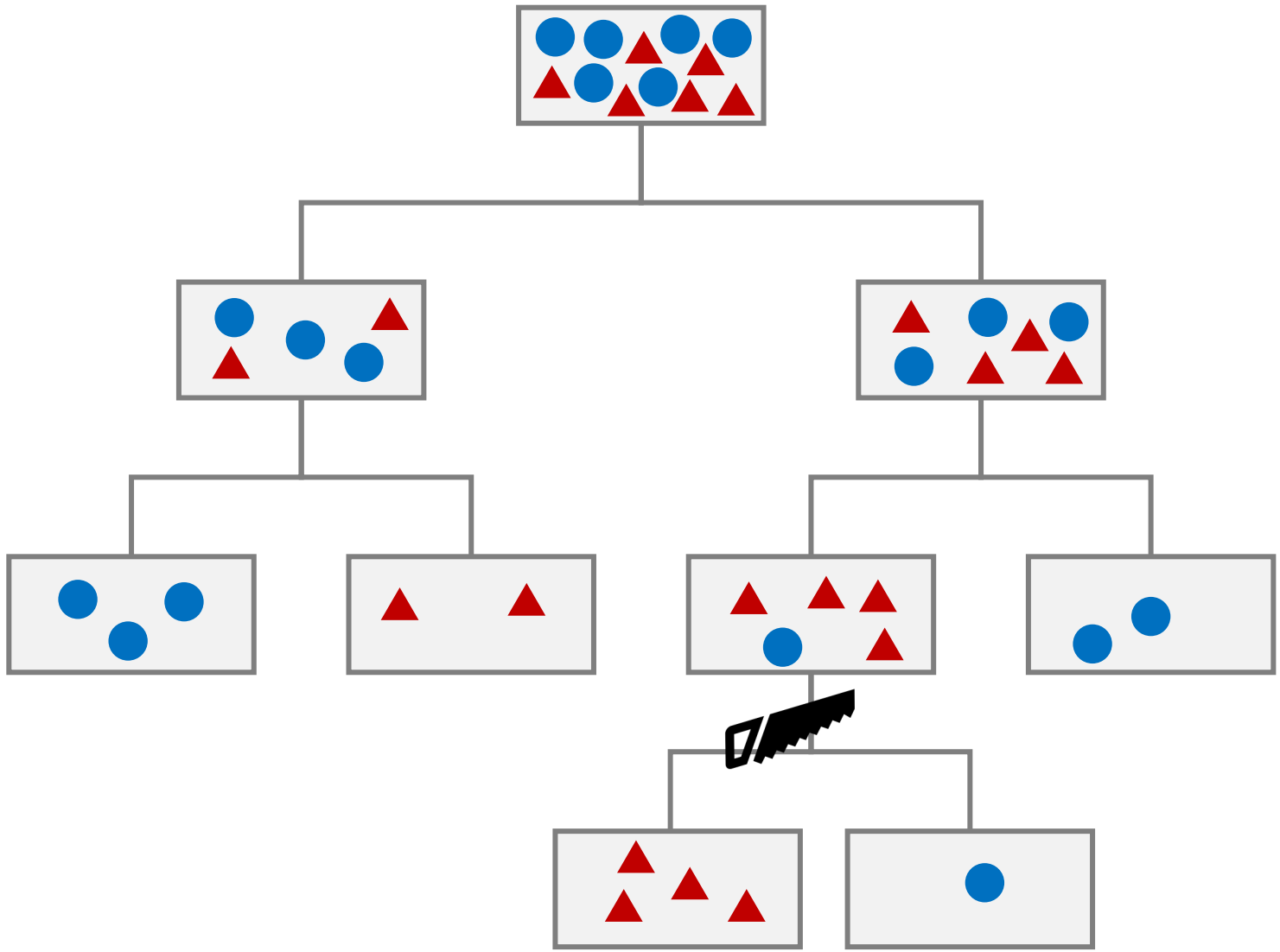


決定木の枝数（層数）を増やことで、すべての訓練データを間違いなく分類できるようになる。訓練データに含まれるノイズまでも分類対象となってしまうことがある。この状態を過学習という。過学習を防ぐために、剪定（枝刈り）を行い、枝数を制限することが行われている。

- 情報利得が閾値以下であれば、そこで剪定する。
- 決定木全体の複雑性が小さくなるように剪定する。

可視化（graphviz, dtreeviz）

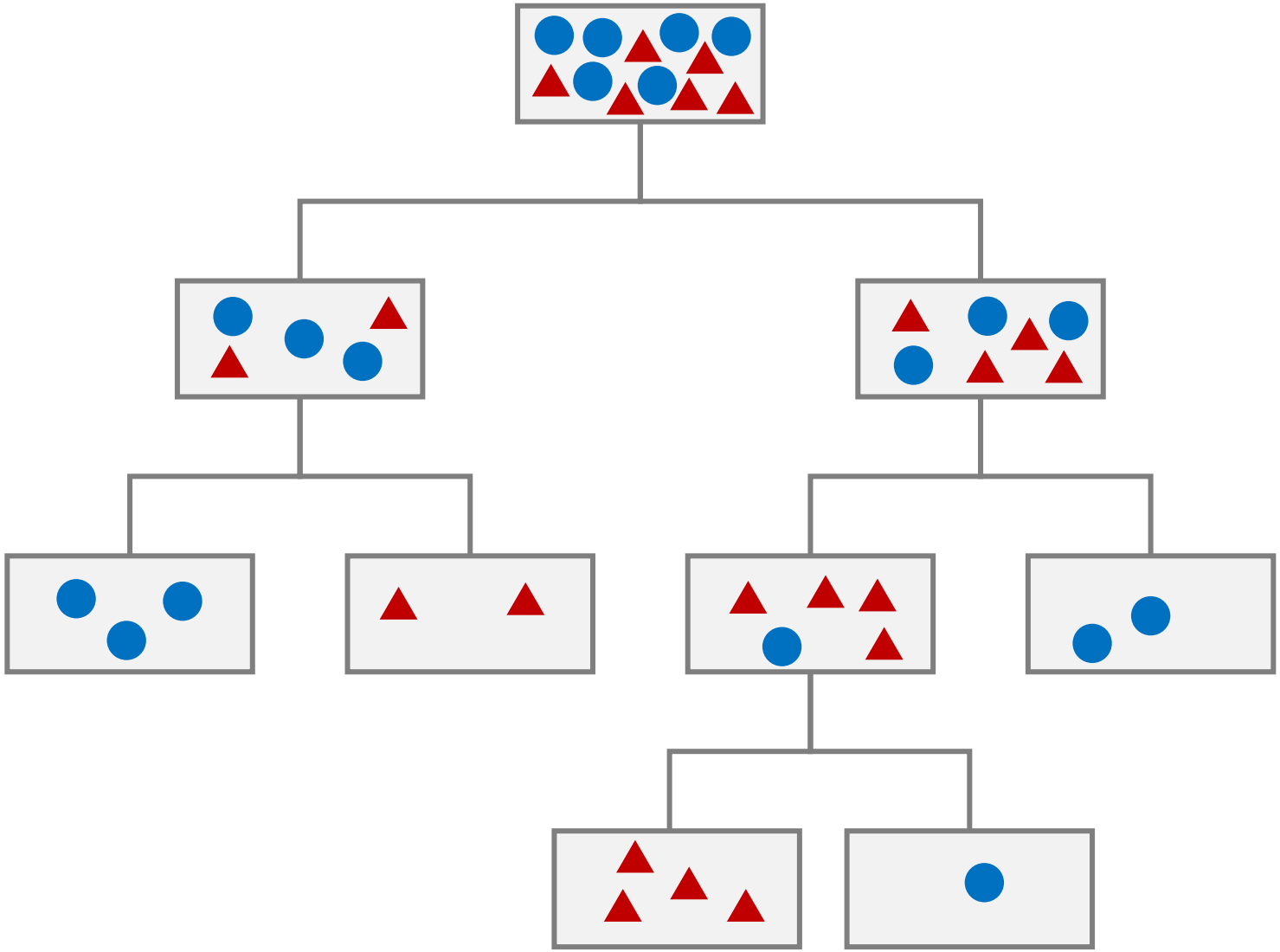
決定木



決定木の枝数（層数）を増やことで、すべての訓練データを間違いなく分類できるようになる。訓練データに含まれるノイズまでも分類対象となってしまうことがある。この状態を過学習という。過学習を防ぐために、剪定（枝刈り）を行い、枝数を制限することが行われている。

- 情報利得が閾値以下であれば、そこで剪定する。
- 決定木全体の複雑性が小さくなるように剪定する。

重要度



重要度

木の根の近くで採用された特徴量が、他の特徴量よりも優れている（分類に有効である）。「どれだけ有効か」がその特徴量の重要度になる。例えば、親子間のジニ係数の減少幅が大きければ、その特徴量の重要度が高いといえる。

- 相関の高い特徴量が複数ある場合、そのうち1特徴量の重要度だけが高く、他は低くなる。
- 重要度はクラス分けのために重要かどうかを示すだけであり、positive あるいは negative という情報を示さない。

重要度

特徴量 x_k の重要度は、特徴量 x_k で分割したノードの改善度の和として、次のように計算できる。

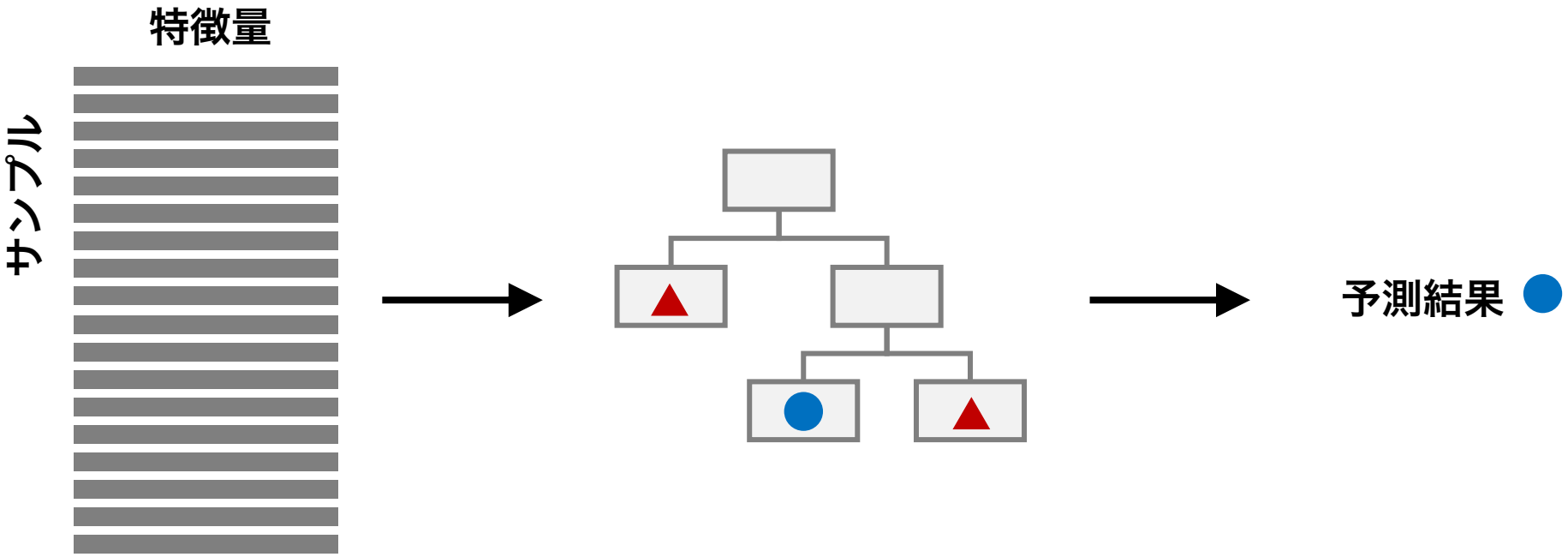
$$importance = \sum_{D_p \in x_k} \Delta I(D_p)$$

$$\Delta I(D_p) = \frac{N_p}{N} I(D_p) - \sum_{j=1}^{c} \frac{N_j}{N} I(D_j)$$

ただし、

D_p	親ノード
D_j	D_pの子ノード
$I(D_i)$	ノード D_i の不純度
N_i	ノード D_i に中のサンプル数
N	全サンプル数

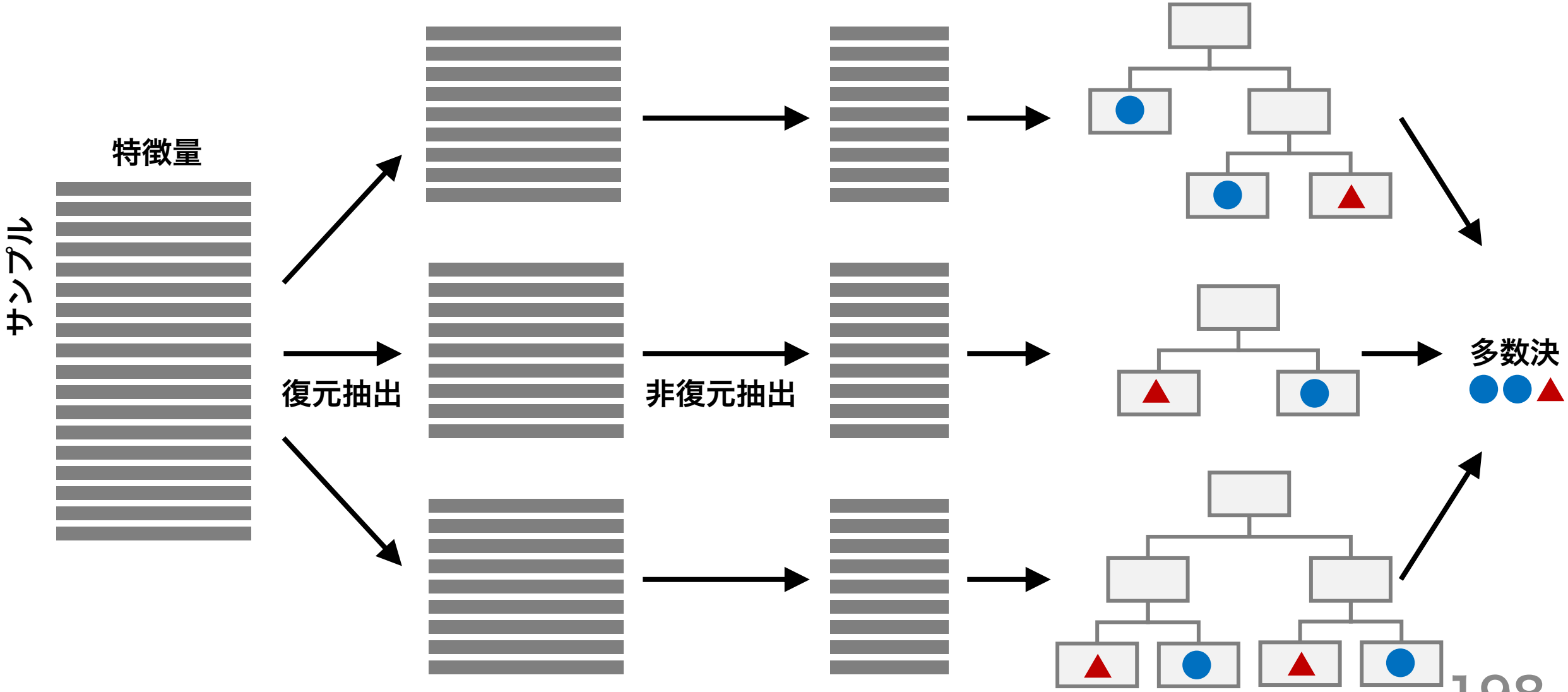
決定木



決定木

- 決定木
- ランダムフォレスト
- 回帰木

ランダムフォレスト



決定木

- 決定木
- ランダムフォレスト
- 回帰木

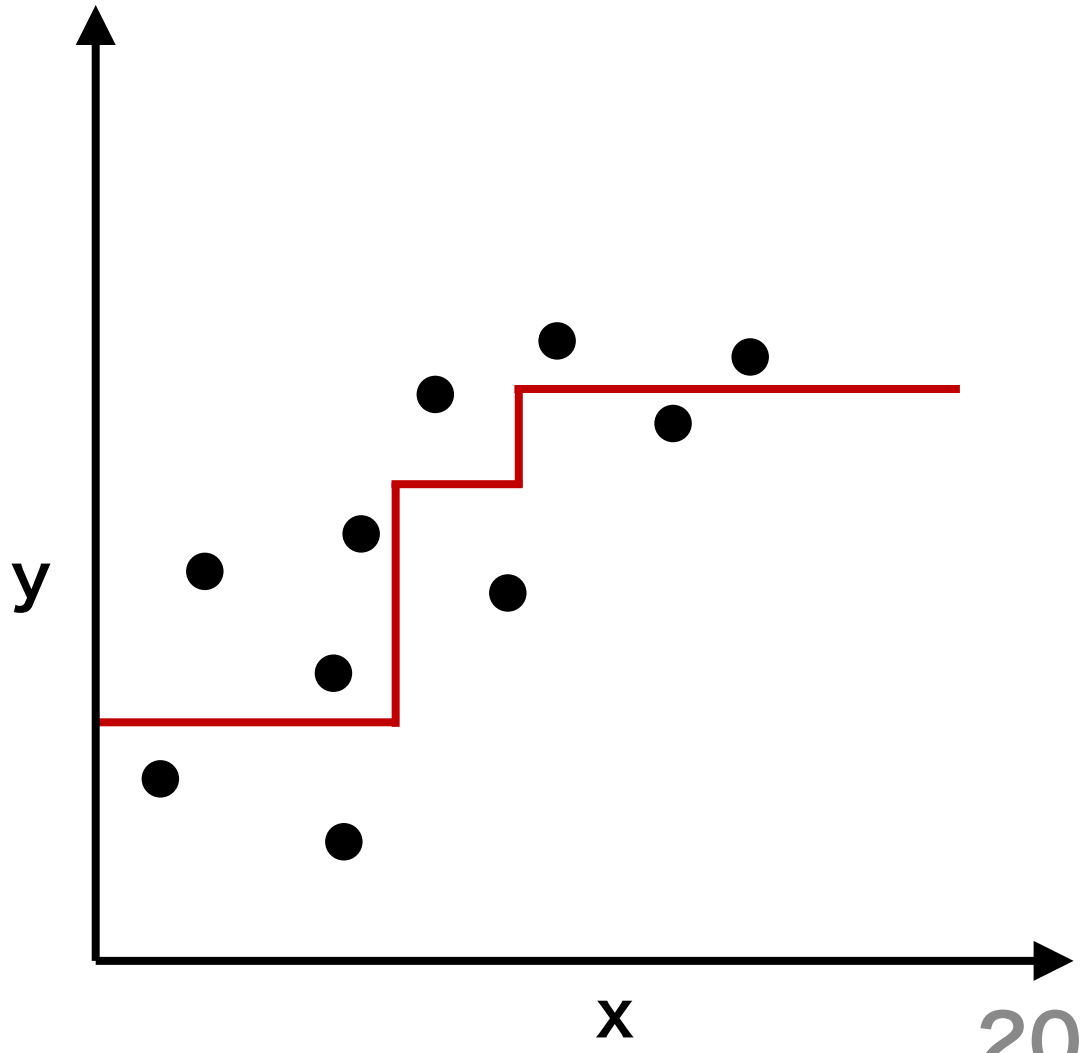
回帰木

回帰木は、決定木のアルゴリズムを回帰に応用したアルゴリズムである。このとき、不純度として、平均二乗誤差が使われる。なお D_t はノード t を表す。

$$I(t) = MSE(t) = \frac{1}{N_t} \sum_{i \in D_t} \left(y^{(i)} - \hat{y}_t \right)^2$$

$$\hat{y}_t = \frac{1}{N_t} \sum_{i \in D_t} y^{(i)}$$

回帰木で構築されたモデルの予測値は、線形回帰とは異なり、連続ではない。



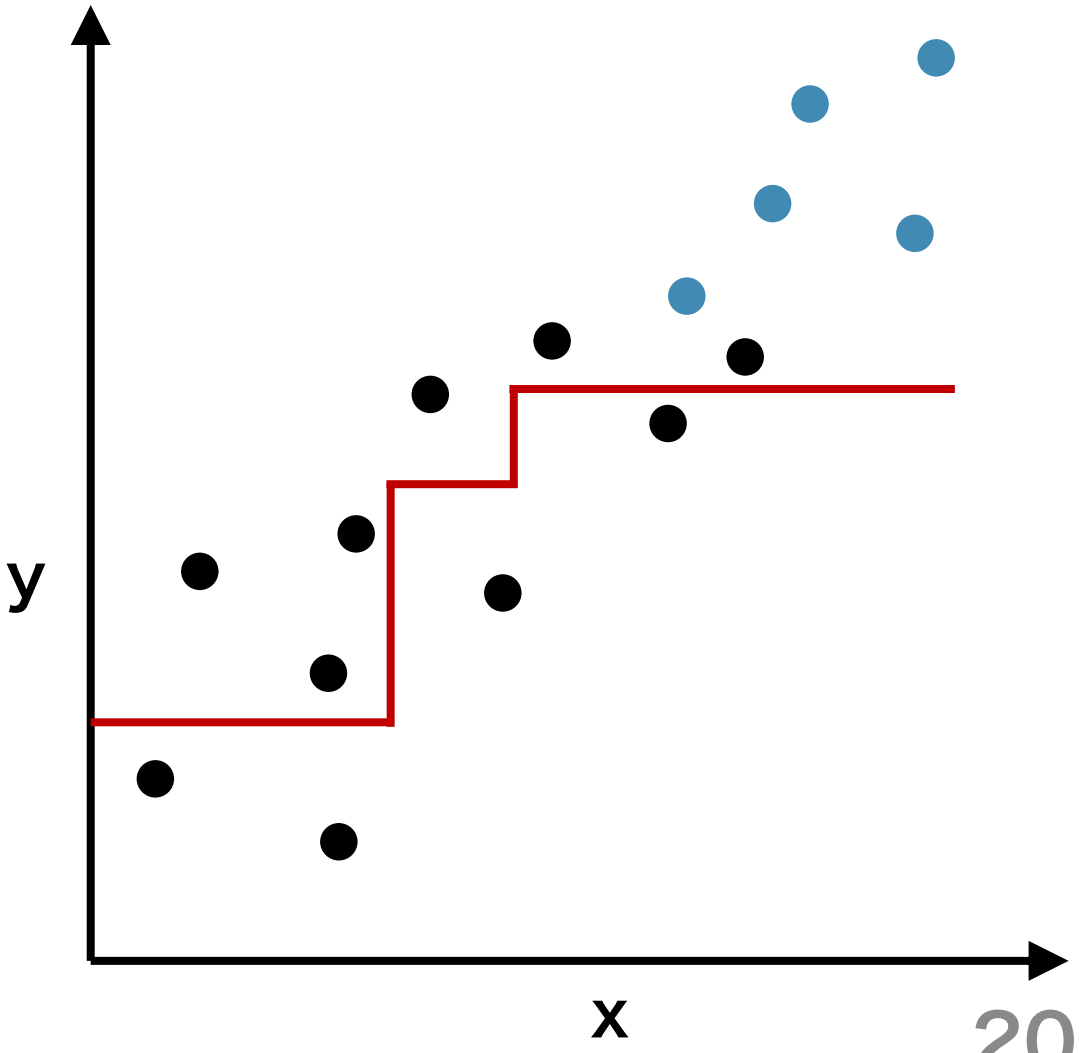
回帰木

回帰木は、決定木のアルゴリズムを回帰に応用したアルゴリズムである。このとき、不純度として、平均二乗誤差が使われる。なお D_t はノード t を表す。

$$I(t) = MSE(t) = \frac{1}{N_t} \sum_{i \in D_t} \left(y^{(i)} - \hat{y}_t \right)^2$$

$$\hat{y}_t = \frac{1}{N_t} \sum_{i \in D_t} y^{(i)}$$

回帰木で構築されたモデルの予測値は、線形回帰とは異なり、連続ではない。また、回帰木で構築されたモデルは、訓練データの範囲外にある未知データが入力された場合に、正確に予測できない場合がある。

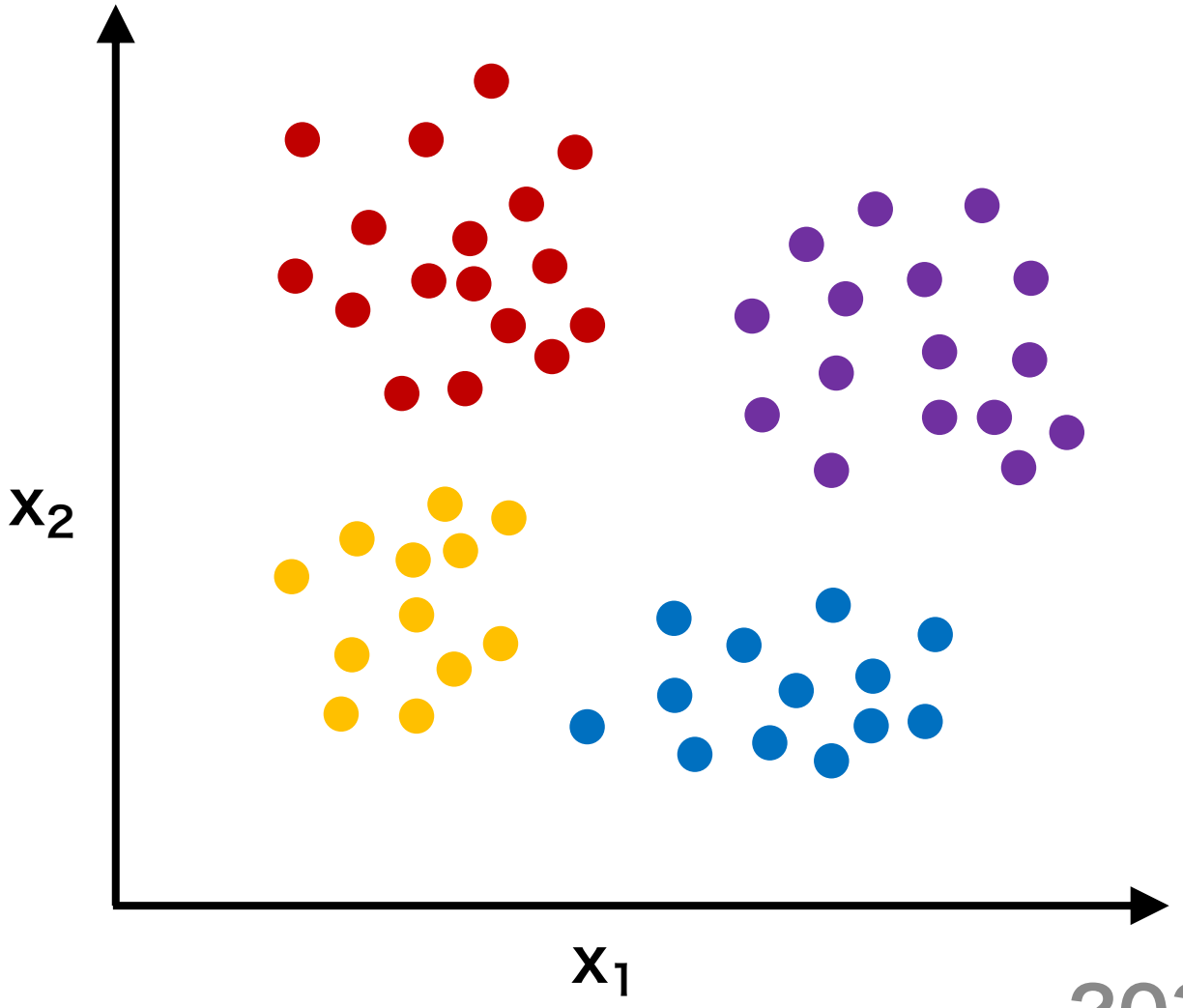


k-近傍法

k-近傍法

k-近傍法は、分類問題を解くための教師あり学習の一つである。k-近傍法では、未知のデータを予測するとき、未知のデータとすべての学習データとの間の距離を計算し、未知データに最も近い k 個の学習データのクラスに基づいて、多数決により分類する。アルゴリズムが簡単で実装しやすいが、ほとんど使われていない。

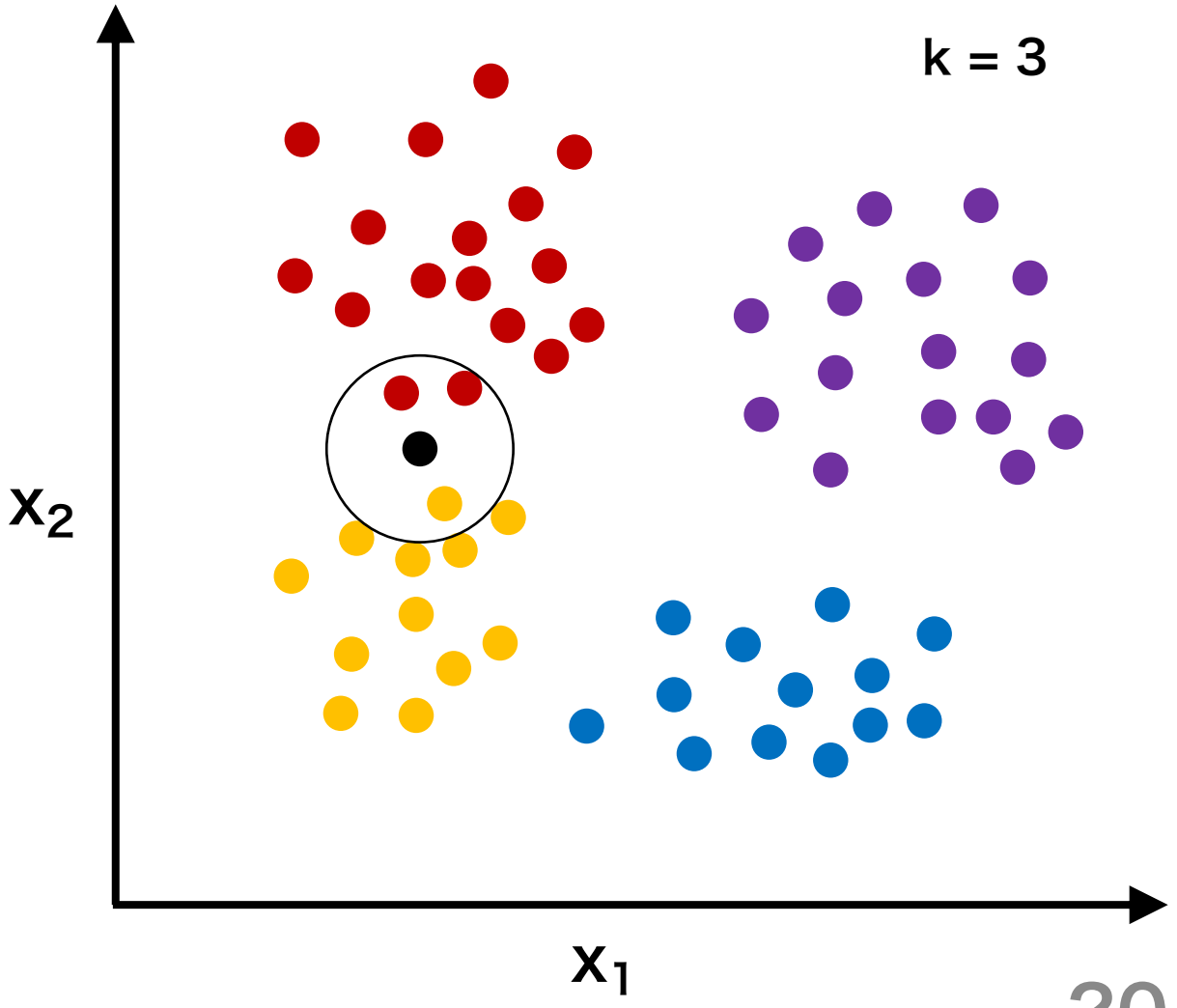
- ハイパーパラメータ k や距離の定義などを調整する必要がある。
- 学習データのサンプル数が多いとき、または特徴量が多いとき、距離計算の計算量が膨大になるため、予測速度が遅くなる。
- 疎なデータセットに対して、予測性能が悪い。



k-近傍法

k-近傍法は、分類問題を解くための教師あり学習の一つである。k-近傍法では、未知のデータを予測するとき、未知のデータとすべての学習データとの間の距離を計算し、未知データに最も近い k 個の学習データのクラスに基づいて、多数決により分類する。アルゴリズムが簡単で実装しやすいが、ほとんど使われていない。

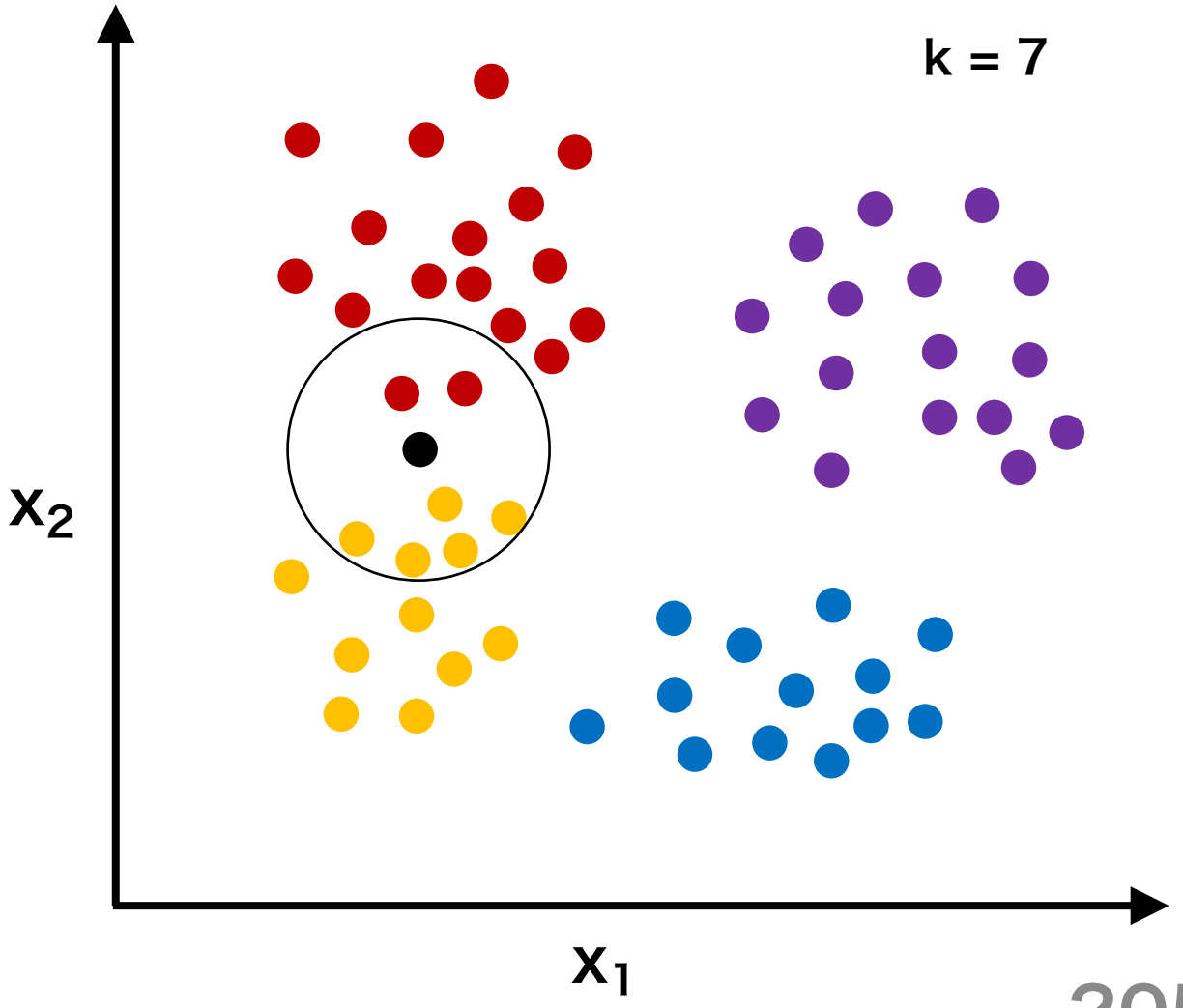
- ハイパーパラメータ k や距離の定義などを調整する必要がある。
- 学習データのサンプル数が多いとき、または特徴量が多いとき、距離計算の計算量が膨大になるため、予測速度が遅くなる。
- 疎なデータセットに対して、予測性能が悪い。



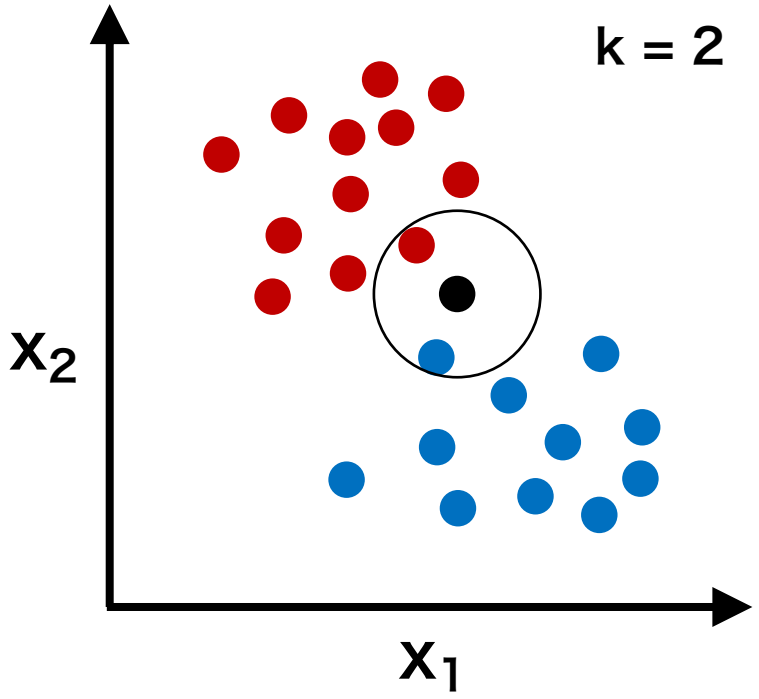
k-近傍法

k-近傍法は、分類問題を解くための教師あり学習の一つである。k-近傍法では、未知のデータを予測するとき、未知のデータとすべての学習データとの間の距離を計算し、未知データに最も近い k 個の学習データのクラスに基づいて、多数決により分類する。アルゴリズムが簡単で実装しやすいが、ほとんど使われていない。

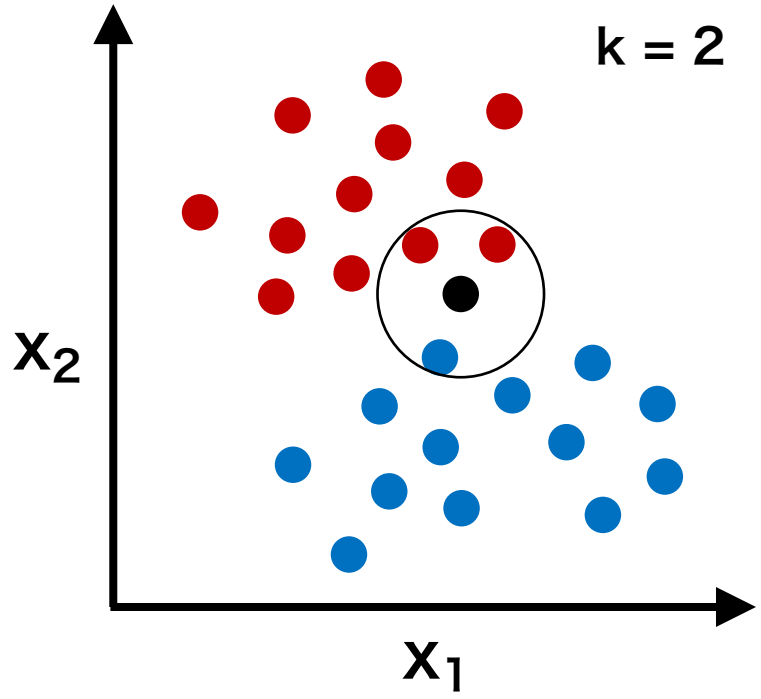
- ハイパーパラメータ k や距離の定義などを調整する必要がある。
- 学習データのサンプル数が多いとき、または特徴量が多いとき、距離計算の計算量が膨大になるため、予測速度が遅くなる。
- 疎なデータセットに対して、予測性能が悪い。



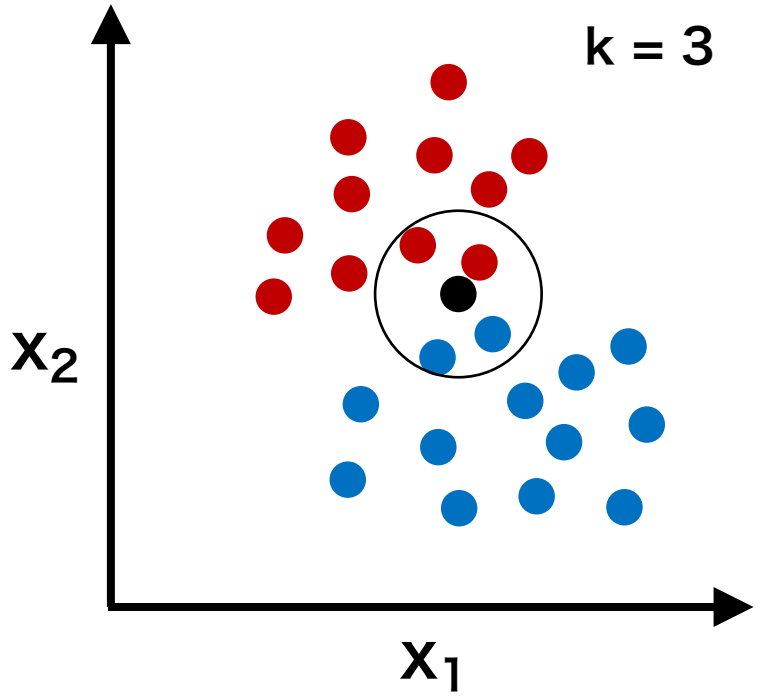
k-近傍法



k の値を変更したり、奇数にしたりすることで、この状況を回避する。



k の値を変更したり、ランダムに 2 つを選んだりすることで、この状況を回避する。



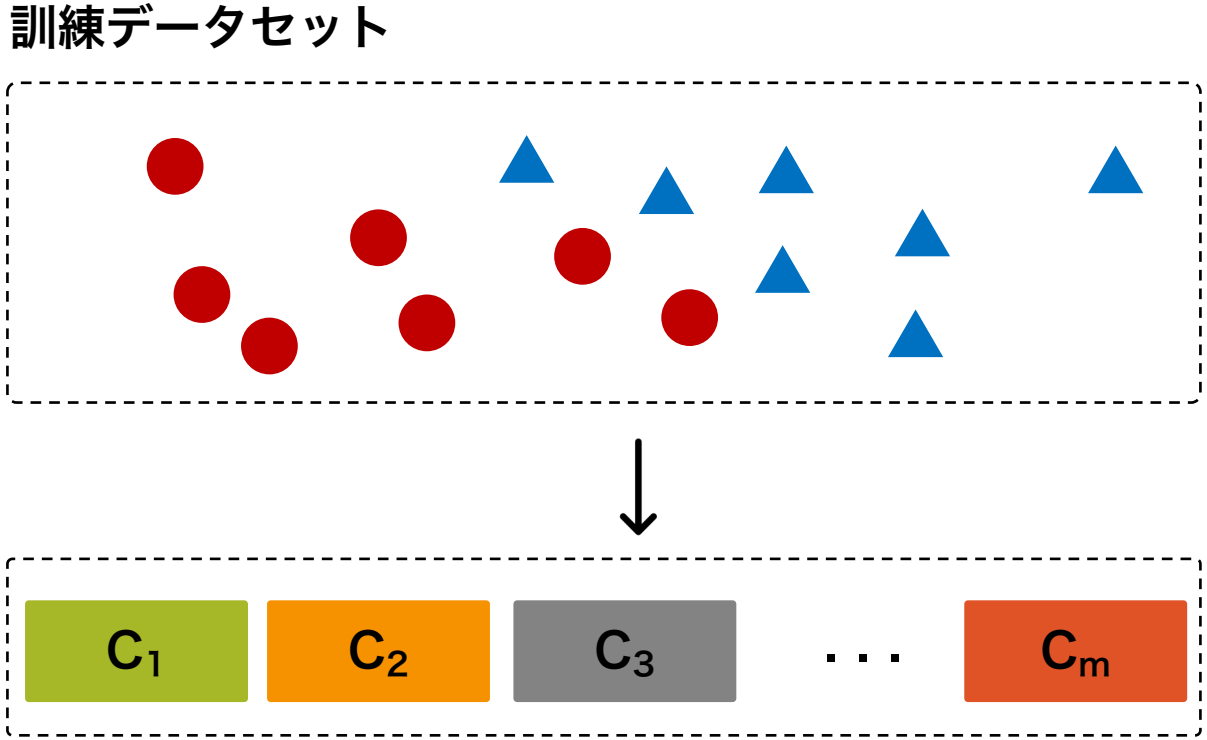
距離を重みとして、重みづけ距離を用いることで、この状況を回避できる。

アンサンブル学習

- バギング
- ブースティング
- スタッキング

アンサンブル学習

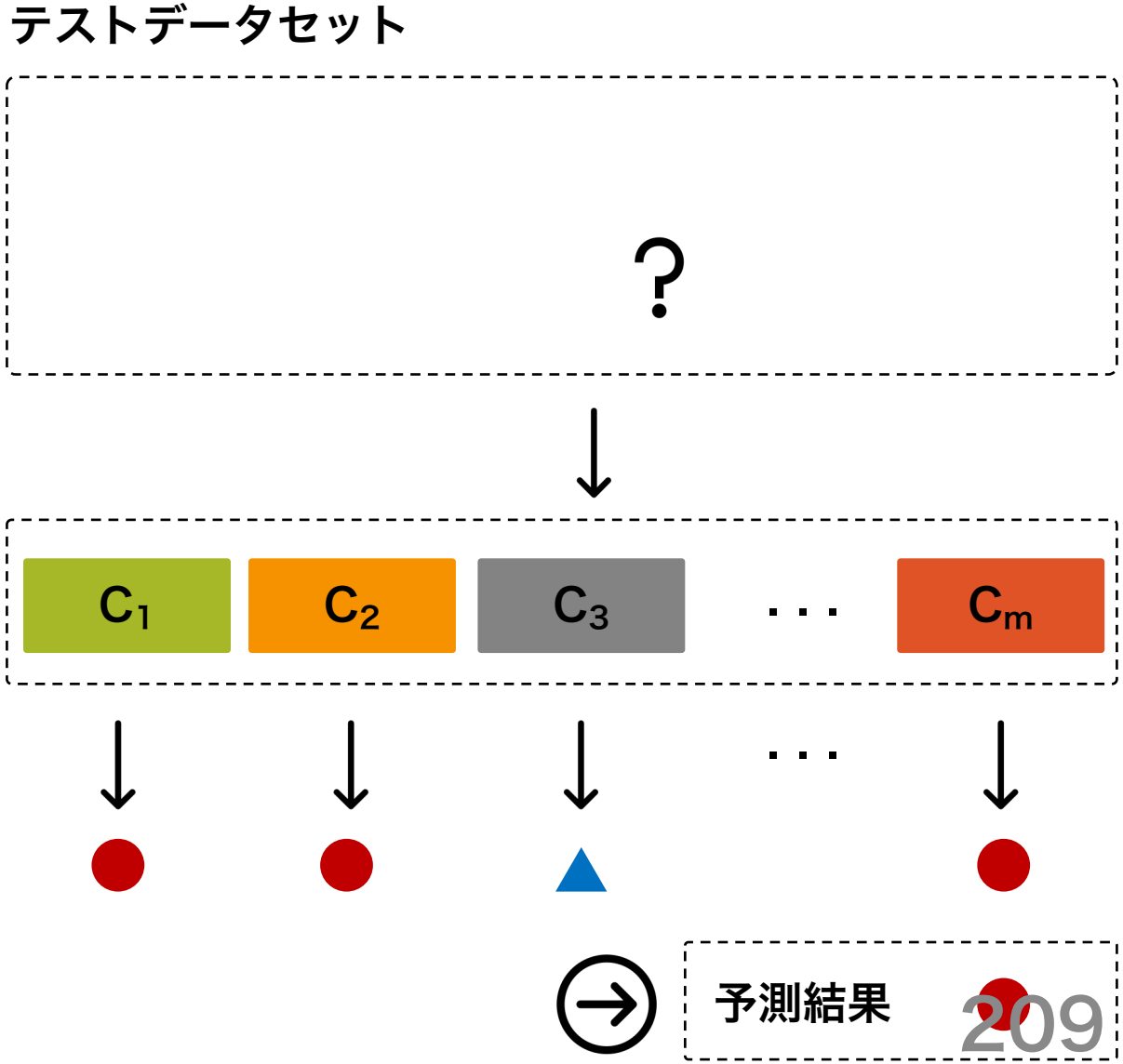
アンサンブル学習とは、様々な学習アルゴリズムで作った予測器を組み合わせて一つの予測器を作る方法である。複数の予測器を組み合わせることで、個々の予測器の弱点が補完され、より高性能な予測器を構築できる。しかし、アンサンブル法により作られる予測器は往々にして複雑であり、技術的な労力に見合わない予測性能の改善が見られない場合もある。



アンサンブル学習

アンサンブル学習とは、様々な学習アルゴリズムで作った予測器を組み合わせて一つの予測器を作る方法である。複数の予測器を組み合わせることで、個々の予測器の弱点が補完され、より高性能な予測器を構築できる。しかし、アンサンブル法により作られる予測器は往々にして複雑であり、技術的な労力に見合わない予測性能の改善が見られない場合もある。

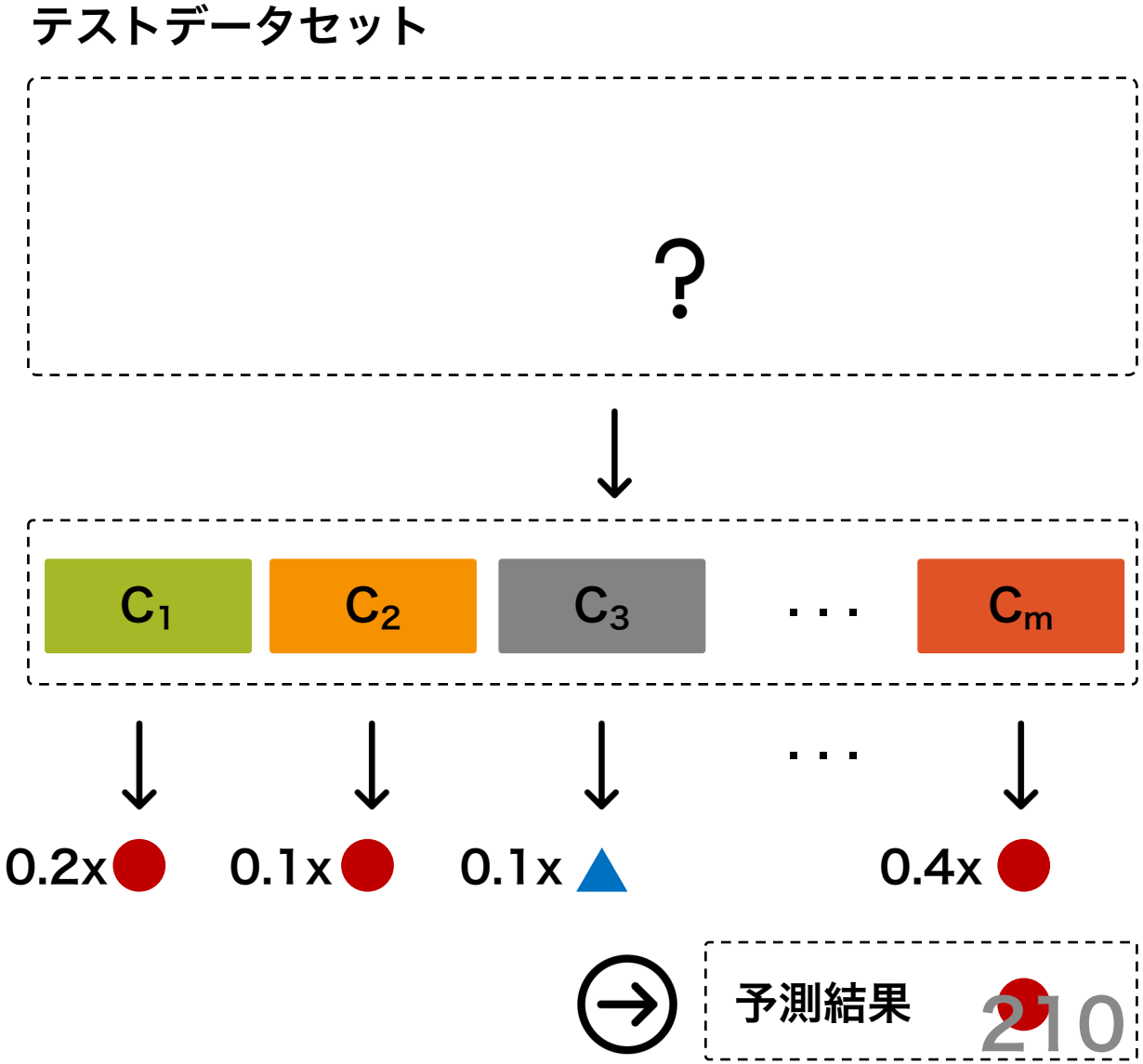
アンサンブル法によって作られる予測器に未知のデータが代入されると、それぞれの予測器が予測結果を出力する。このとき、多数決によって最終的な予測結果を出力する。



アンサンブル学習

アンサンブル学習とは、様々な学習アルゴリズムで作った予測器を組み合わせて一つの予測器を作る方法である。複数の予測器を組み合わせることで、個々の予測器の弱点が補完され、より高性能な予測器を構築できる。しかし、アンサンブル法により作られる予測器は往々にして複雑であり、技術的な労力に見合わない予測性能の改善が見られない場合もある。

アンサンブル法によって作られる予測器に未知のデータが代入されると、それぞれの予測器が予測結果を出力する。このとき、多数決によって最終的な予測結果を出力する。場合によって、これらの予測器の結果に重みをつけて、最終的な予測結果を出力することもある。

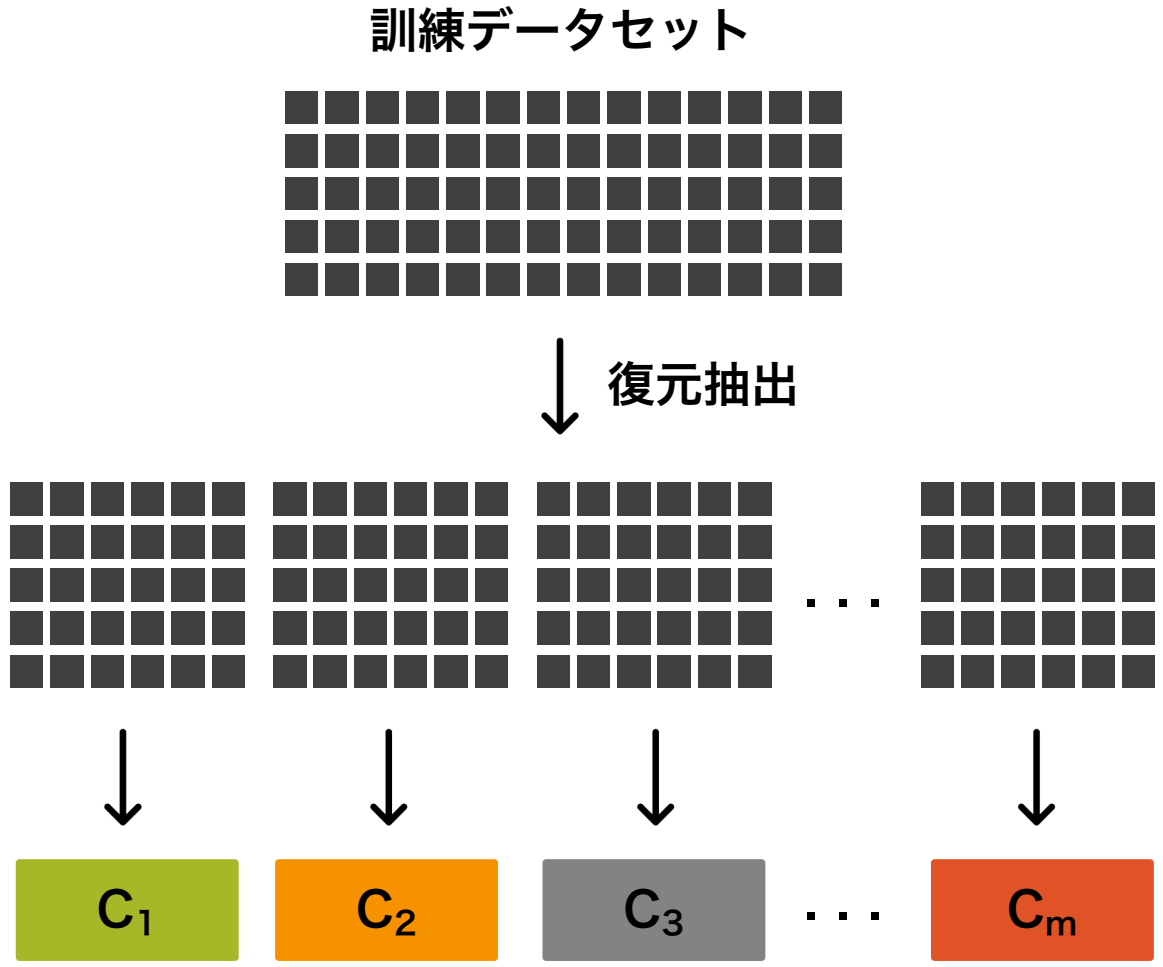


アンサンブル学習

- バギング
- ブースティング
- スタッキング

バギング

同じ訓練データセットを用いて複数の予測器を作成した場合、過学習を起したバリアンスの高い予測モデルが作られる可能性が高い。そこで、訓練データセットから復元抽出したサブセットを用いて、各予測器を学習させるバギングと呼ばれる手法が使われることもある。各予測器は異なる訓練データのサブセットから作られるため、過学習を抑制できる。



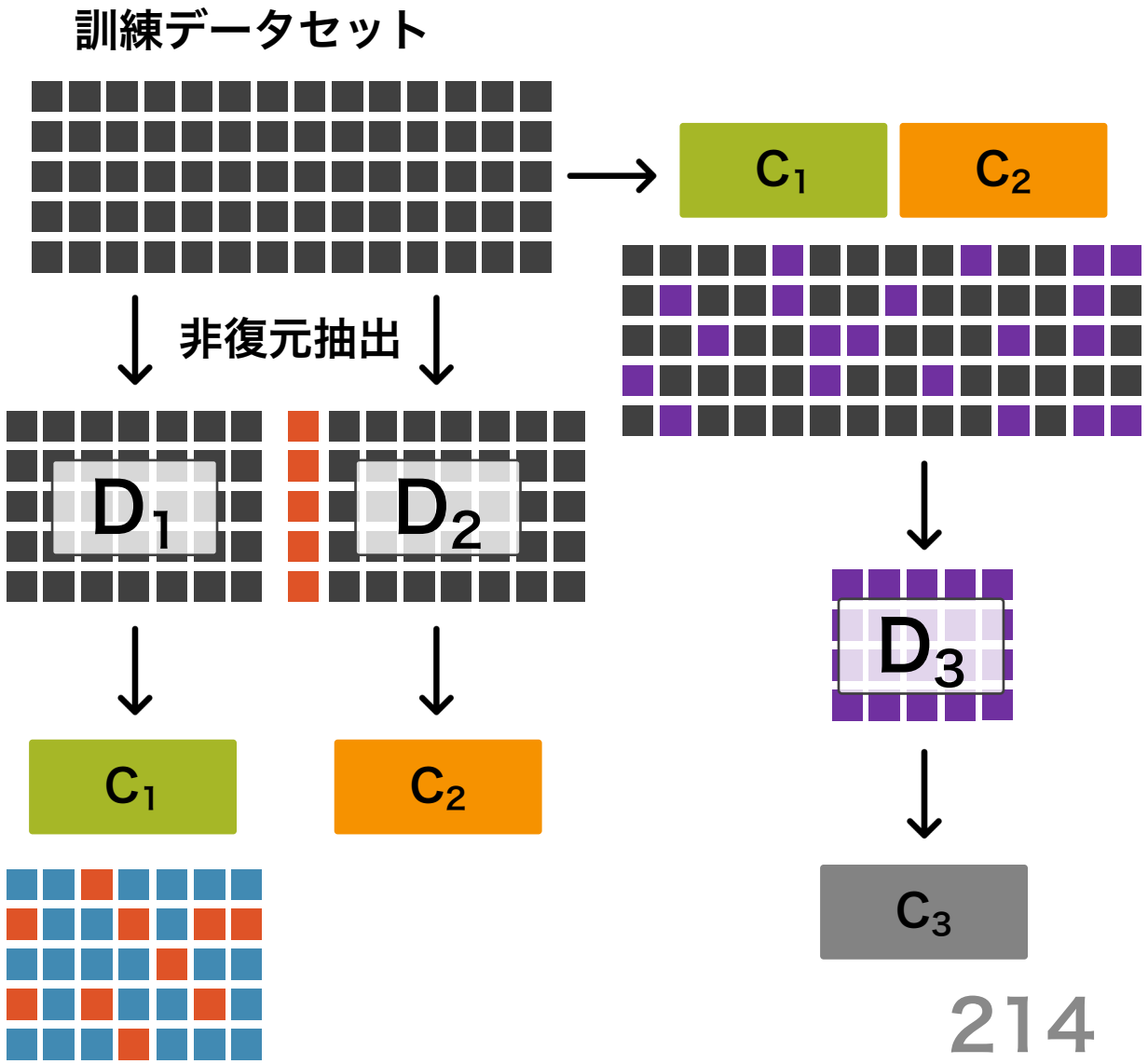
アンサンブル学習

- バギング
- ブースティング
- スタッキング

ブースティング

ブースティングは、アンサンブル学習の一つで、誤分類されやすいサンプルを重点的に再学習を行うことで、モデルの最終的な性能を向上を図っている。比較的に有名なアルゴリズムとして AdaBoost がある。AdaBoost では、概ね次のような手順で学習を行なっている。

1. 訓練データを 2 個のサブセット D_1 と D_2 に分ける。
2. サブセット D_1 を利用して予測器 C_1 を作成する。
3. 手順 2 で作られた予測器を使って、サブセット D_1 を予測する。誤分類したサンプルの 50% をサブセット D_2 に加える。
4. サブセット D_2 を利用して予測器 C_2 を作成する。
5. 訓練データセット全体を予測器 C_1 と C_2 に予測させて、両者の予測結果が異なっているサンプルをサブセット D_3 とする。
6. サブセット D_2 を利用して予測器 C_2 を作成する。
7. 予測器 C_1, C_2, C_3 の多数決により予測を行う。



ブースティング

- AdaBoost, GradientBoosting
  - https://scikit-learn.org/stable/modules/ensemble.html
  - scikit-learn に実装されている。

- XGBoost (Extreme Gradient Boosting, 2014)
  - https://xgboost.readthedocs.io/en/latest/
  - gradient boosting とランダムフォレストを組み合わせたアルゴリズム
  - 性能は高いがハイパーパラメータが多い

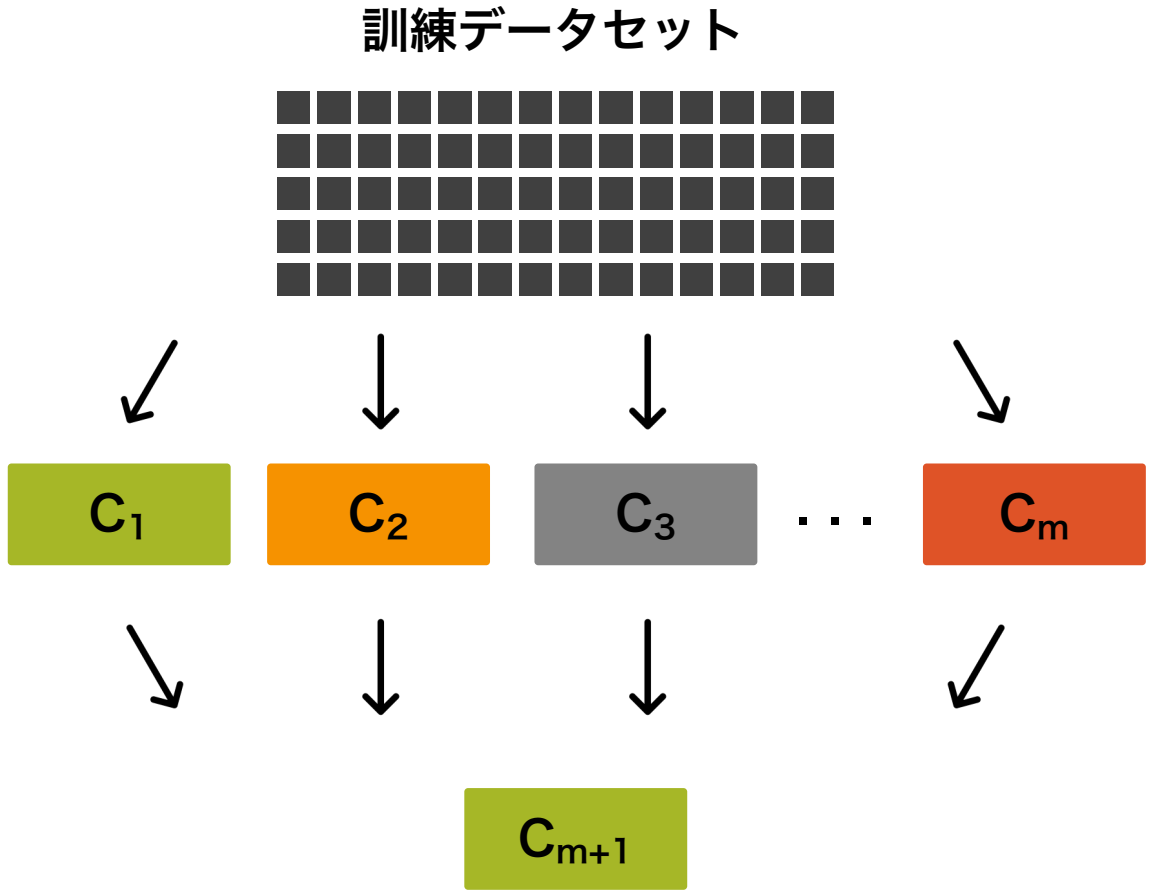
- LightGBM (Microsoft, 2016)
  - https://lightgbm.readthedocs.io/en/latest/
  - gradient boostingとランダムフォレストを組み合わせたアルゴリズム
  - 性能は高いがハイパーパラメータが多い

アンサンブル学習

- バギング
- ブースティング
- スタッキング

スタッキング

独立した学習器（たとえばサポートベクターマシン、ランダムフォレスト、ニューラルネットワークなど）で訓練データを学習、それらの出力をあわせて別の学習器の訓練データとし、学習。これを次々と繰り返す。上位層の出力を下位層でどのように訓練データとして用いるかは、データによってさまざまに工夫する。



スタッキング

GBM　ブースティング
NN　ニューラルネットワーク
FM　Factorization Machine
LR　ロジスティック回帰
KRR　Kernel Ridge Regression
ET　Extremely Random Trees
RF　ランダムフォレスト
KNN　k-近傍法

KDD Cup 2015 Jeong-Yoon Lee
https://www.slideshare.net/jeongyoonlee/winning-data-science-competitions-74391113

データ前処理

- 欠損値処理
- 特徴量エンジニアリング
- 特徴量選択
- 特徴量抽出

データ収集

市場調査を行い、ニーズを確定し、モデル構築に必要なデータを収集する。

ワークフロー構築

収集されたデータに対して欠損値や異常値処理を行い、データを機械学習が使える形に整形し、機械学習アルゴリズムに渡すまでのワークフローを構築する。

モデル構築

入力データを使用して、パラメーターチューニングを行い、最適なモデルを構築する。

実装

機械学習モデルをアプリ化し、スマホや専用機器に実装し、実用化する。

データ前処理

- **欠損値処理**　調査方法や調査データを再確認し、データの確認を行う。未調査による欠損値や測定不備などによる異常値を取り除いたり、補完したりする。
- **データ整形**　データの形式をプログラムが取り扱いやすいように形式変換を行う。例えば、文字列からなるクラスラベルを整数値に置き換えたりする。
- **スケーリング**　特徴量同士のスケールを揃えることで、ほとんどの機械学習アルゴリズムうまく動作できるようになる。
- **特徴量選択**　特徴量の中から重要なものを選択したり、あるいはいくつかの特徴量から新しい特徴量を合成したりして、特徴選択を行う。過学習を抑えることが期待される。

データ前処理

- 欠損値処理
- 特徴量エンジニアリング
- 特徴量選択
- 特徴量抽出

欠損値処理

scikit-learn などの機械学習用のライブラリーのほとんどは、欠損値を含むデータに対応していない。そのため、これらのライブラリーを使用するにあたって、ユーザー自身が欠損値や異常値を除去したり、補正したりする前処理を行う必要がある。欠損値は以下の 3 タイプあり、欠損値処理が有効であるのは MCAR のタイプであることに注意。

- MCAR (Missing Completely At Random)
  完全にランダムで生じる欠損値。
- MAR (Missing At Random)
  観測データに依存してランダムに生じる欠損値。
- MNAR (Missing Not At Random)
  欠損データに依存して生じる欠損値。

label	length	weight	nleaves	nseeds
koshihikari	20.7	NA	79	932
hitomebore	18.4	19.3	86	838
tsuyahime	24.2	20.1	92	840
koshihikari	21.4	29.0	82	871
koshihikari	20.3	28.6	91	925

MAR

scikit-learn などの機械学習用のライブラリーのほとんどは、欠損値を含むデータに対応していない。そのため、これらのライブラリーを使用するにあたって、ユーザー自身が欠損値や異常値を除去したり、補正したりする前処理を行う必要がある。欠損値は以下の 3 タイプあり、欠損値処理が有効であるのは MCAR のタイプであることに注意。

- ○ MCAR (Missing Completely At Random)
  完全にランダムで生じる欠損値。
- ○ MAR (Missing At Random)
  観測データに依存してランダムに生じる欠損値。
- ○ MNAR (Missing Not At Random)
  欠損データに依存して生じる欠損値。

label	length	weight	nleaves	nseeds
koshihikari	20.7	NA	79	932
hitomebore	18.4	19.3	86	838
tsuyahime	24.2	20.1	92	840
koshihikari	21.4	29.0	82	871
koshihikari	20.3	NA	91	925

MAR は、観測データに依存してランダムに生じる欠損値。例えば、koshihikari 品種のデータに限って、weight 特徴量が欠損しやすい。他の特徴量と関係を利用して、ある程度バイアスを抑えて補正することができる。

MNAR

scikit-learn などの機械学習用のライブラリーのほとんどは、欠損値を含むデータに対応していない。そのため、これらのライブラリーを使用するにあたって、ユーザー自身が欠損値や異常値を除去したり、補正したりする前処理を行う必要がある。欠損値は以下の 3 タイプあり、欠損値処理が有効であるのは MCAR のタイプであることに注意。

- ○ MCAR (Missing Completely At Random)
  完全にランダムで生じる欠損値。
- ○ MAR (Missing At Random)
  観測データに依存してランダムに生じる欠損値。
- ○ MNAR (Missing Not At Random)
  欠損データに依存して生じる欠損値。

label	length	weight	nleaves	nseeds
koshihikari	20.7	NA	79	932
hitomebore	18.4	19.3	86	838
tsuyahime	24.2	20.1	92	840
koshihikari	21.4	29.0	82	871
koshihikari	20.3	NA	91	925

MNAR は、欠損データに依存して生じる欠損値。例えば、nanatsuboshi 品種の栽培実験がほとんど行われていないために、その品種の関連するすべてのデータがほぼ欠損している。そもそもデータが少ない（存在しない）ので補正が困難である。

欠損値処理

scikit-learn や keras などの機械学習用のライブラリーのほとんどは、欠損値を含むデータに対応していない。そのため、これらのライブラリーを使用するにあたって、ユーザー自身が欠損値や異常値を除去したり、補正したりする前処理を行う必要がある。

- **サンプルの削除**
  欠損値や異常値を含むサンプルをデータセットから除去する。この操作により、サンプルの数が減るため、サンプル不足によるモデルの信頼度の低下を招くことがある。特定のクラスのサンプル数が著しく減ることもある。

label	length	weight	nleaves	nseeds
~~koshihikari~~	~~20.7~~	~~NA~~	~~79~~	~~932~~
hitomebore	18.4	19.3	86	838
tsuyahime	24.2	20.1	92	840
koshihikari	21.4	29.0	82	871
~~koshihikari~~	~~20.3~~	~~NA~~	~~91~~	~~925~~

label	length	weight	nleaves	nseeds
hitomebore	18.4	19.3	86	838
tsuyahime	24.2	20.1	92	840
koshihikari	21.4	29.0	82	871

※ koshihikari 品種のデータ数が著しく減少したことに注意。

欠損値処理

scikit-learn や keras などの機械学習用のライブラリーのほとんどは、欠損値を含むデータに対応していない。そのため、これらのライブラリーを使用するにあたって、ユーザー自身が欠損値や異常値を除去したり、補正したりする前処理を行う必要がある。

- **サンプルの削除**
  欠損値や異常値を含むサンプルをデータセットから除去する。この操作により、サンプルの数が減るため、サンプル不足によるモデルの信頼度の低下を招くことがある。特定のクラスのサンプル数が著しく減ることもある。

- **特徴量の削除**
  欠損値や異常値を含む特徴量をデータセットから除去する。この操作により、除去された特徴量が予測に使われる重要な特徴量であるとき、モデルの性能を著しく低下させてしまうことがある。

label	length	weight	nleaves	nseeds
koshihikari	20.7	NA	79	932
hitomebore	18.4	19.3	86	838
tsuyahime	24.2	20.1	92	840
koshihikari	21.4	29.0	82	871
koshihikari	20.3	28.6	91	925

label	length	nleaves	nseeds
koshihikari	20.7	79	932
hitomebore	18.4	86	838
tsuyahime	24.2	92	840
koshihikari	21.4	82	871
koshihikari	20.3	91	925

欠損値処理

scikit-learn や keras などの機械学習用のライブラリーのほとんどは、欠損値を含むデータに対応していない。そのため、これらのライブラリーを使用するにあたって、ユーザー自身が欠損値や異常値を除去したり、補正したりする前処理を行う必要がある。

- **サンプルの削除**
  欠損値や異常値を含むサンプルをデータセットから除去する。この操作により、サンプルの数が減るため、サンプル不足によるモデルの信頼度の低下を招くことがある。特定のクラスのサンプル数が著しく減ることもある。

- **特徴量の削除**
  欠損値や異常値を含む特徴量をデータセットから除去する。この操作により、除去された特徴量が予測に使われる重要な特徴量であるとき、モデルの性能を著しく低下させてしまうことがある。

- **補完**
  欠損値や異常値を、その特徴量全体の平均値あるいは最頻値で補完する。サンプル数や特徴量の減少がない。

label	length	weight	nleaves	nseeds
koshihikari	20.7	NA	79	932
hitomebore	18.4	19.3	86	838
tsuyahime	24.2	20.1	92	840
koshihikari	21.4	29.0	82	871
koshihikari	20.3	28.6	91	925

label	length	weight	nleaves	nseeds
koshihikari	20.7	24.3	79	932
hitomebore	18.4	19.3	86	838
tsuyahime	24.2	20.1	92	840
koshihikari	21.4	29.0	82	871
koshihikari	20.3	28.6	91	925

欠損値補完

欠損値補完は、最もシンプルな方法として、欠損値となっている部分を、その特徴量全体の平均値あるいは最頻値に置き換えることで補完する。この他に、線形補間、kNN、MICE などもあり、実用上、平均値で置き換えることよりも、これらの手法がよく使われている。

線形補間

特徴量と特徴量あるいは特徴量と教師ラベル間の回帰分析を行い、その回帰係数等を使用して、特徴量中の欠損値を推定して補完する。

kNN

kNN のアルゴリズムを使用して、特徴空間上で、欠損値を起こしたサンプルが配置している場所を探索し、その近くにある値で補完する。

MICE

複数の欠損値を一端埋めたデータセットを作成する。次に、ある欠損値を 1 つだけ欠損させた状態で、線形補間などを行う。次に、同様な操作により、欠損値を 1 つずつ補完していく方法。

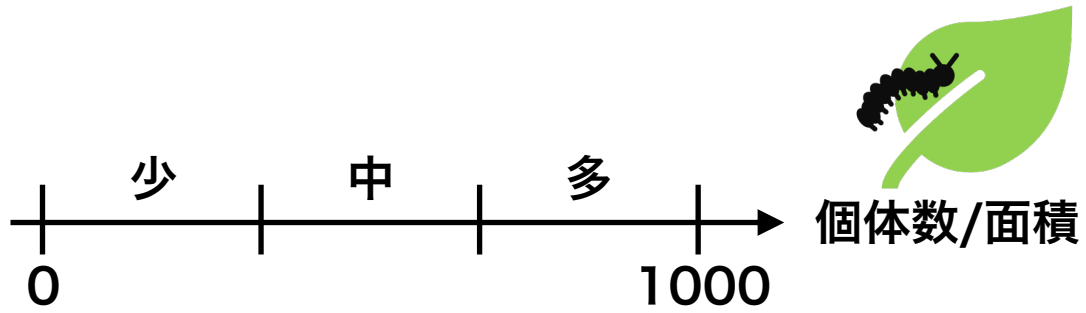
データ前処理

- 欠損値処理
- 特徴量エンジニアリング
- 特徴量選択
- 特徴量抽出

前処理

機械学習の前処理は、学習時の収束速度や予測時の予測性能の向上が目的で行われる。統計における「データを特定の確率分布に近づける」という目的ではない。そのため、解析者は各々のデータの特徴と目的に応じて最適な前処理を決める必要がある。

- 気象データを用いて害虫発生量を予測するとき、害虫の個体数よりも規模を予測するのが真の目的であれば、初めから、害虫の発生量に閾値を設けて多・中・少にしてからモデル構築を行う。



- 予測したいカテゴリ数が多い場合は、重要でないカテゴリを 1 つまとめて、モデルを構築する。

特徴量エンジニアリング

- カテゴリデータ
- 連続量データ

特徴量エンジニアリング

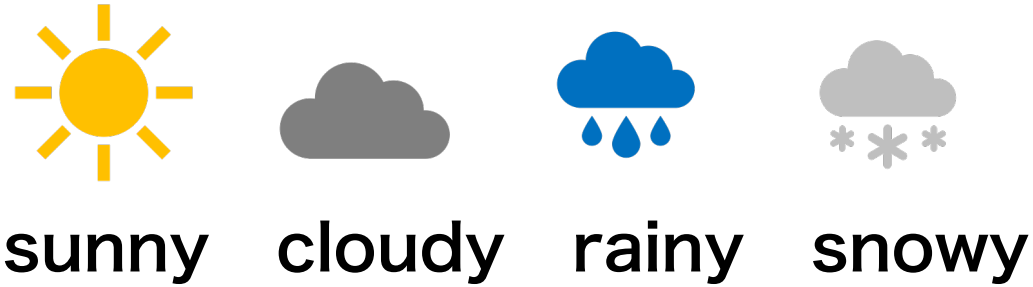
- カテゴリデータ
- 連続量データ

データ整形（カテゴリデータ）

カテゴリ変数は、服のサイズを表す S、M、L や動物名を表す sunny、cloudy、rainy、snowy などのように、通常文字列で表されている。前者のように各文字に順序関係が存在するとき、その特徴量を順序特徴量という。これに対して、後者のように順序のない特徴量を名義特徴量という。ほとんどの機械学習アルゴリズムは、これらのような文字からなる情報をサポートしていない。そのため、ユーザーは自らこれらの文字情報を数値情報に置き換える必要がある。

○ 順序特徴量

○ 名義特徴量



順序特徴量

順序特徴量の場合、その大小関係を保持させて整数値に変換することで対応できる。例えば、S = 1、M = 2、L = 3 のように変換することができる。

Age	Size
26	S
31	M
28	S
34	L
38	L
32	L

→

Age	Size
26	1
31	2
28	1
34	3
38	3
32	3

名義特徴量（one-hot encoding）

名義特徴量の場合は、ダミー変数を制定して、各文字ラベルをダミー変数に置き換えることができる。例えば、下の表のように sunny、cloudy、rainy、snowy の 4 つのラベルに対応するためのダミー変数を 4 つ用意して、sunny であれば sunny 特徴量に 1 を、他の 3 特徴量に 0 として表すことができる。

sunny

cloudy

rainy

snowy

Flower	Weather
1	rainy
1	cloudy
0	sunny
1	rainy
0	snowy
0	sunny

→

Flower	sunny	cloudy	rainy	snowy
1	0	0	1	0
1	0	1	0	0
0	1	0	0	0
1	0	0	1	0
0	0	0	0	1
0	1	0	0	0

名義特徴量（dummy encoding）

名義特徴量の場合は、ダミー変数を制定して、各文字ラベルをダミー変数に置き換えることができる。ラベルが 4 つあるとき、4 つのダミー変数を用意する必要はない。例えば、sunny、cloudy および rainy に対応するダミー変数を用意して、この 3 つのダミー変数がすべて 0 のときは自動的に snowy を意味するようになる。one-hot encoding に比べ、この dummy encoding の方が多重共線性を回避することができる。

sunny　cloudy　rainy　snowy

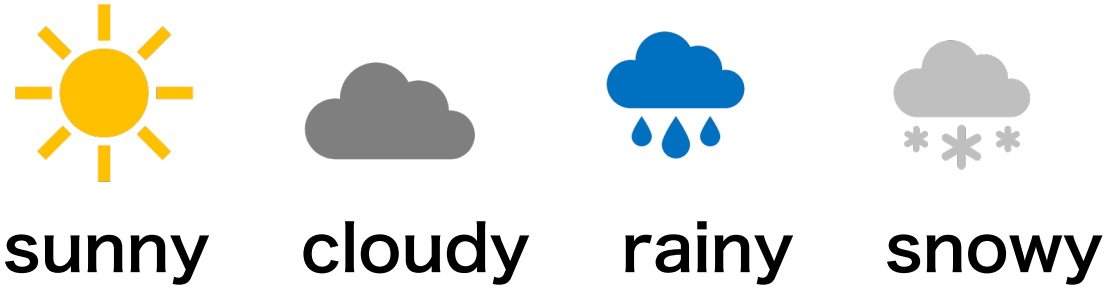
Flower	Weather
1	rainy
1	cloudy
0	sunny
1	rainy
0	snowy
0	sunny

→

Flower	sunny	cloudy	rainy	
1	0	0	1	
1	0	1	0	
0	1	0	0	
1	0	0	1	
0	0	0	0	snowy
0	1	0	0	

名義特徴量（effect encoding）

one-hot encoding の多重共線性を回避するもう一つの方法として、effect encoding がある。effect encoding では、dummy encoding のとき、すべての要素が 0 のベクトルを -1 に置き換えたものに等しい。



Flower	Weather
1	rainy
1	cloudy
0	sunny
1	rainy
0	snowy
0	sunny

→

Flower	sunny	cloudy	rainy
1	0	0	1
1	0	1	0
0	1	0	0
1	0	0	1
0	-1	-1	-1
0	1	0	0

snowy

ビッグデータの encoding

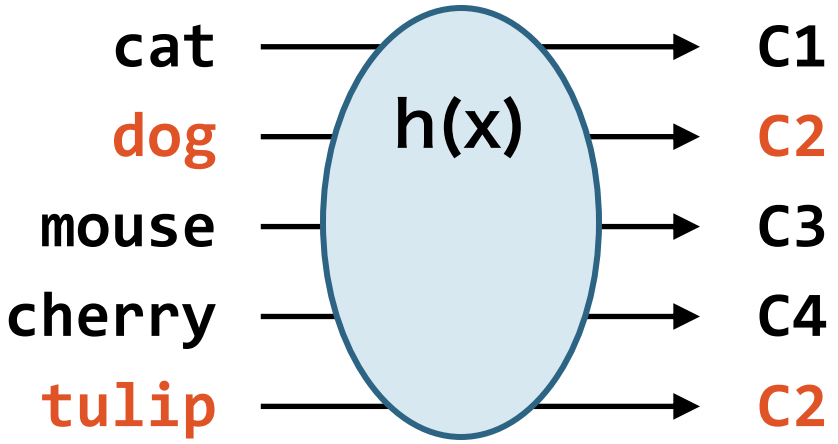
カテゴリ n 個あるとき、少なくとも n -1 個のダミー変数を用意する必要がある。カテゴリ数の多いデータに対して、encoding する場合は、大きな計算メモリを必要とする。この場合、カテゴリ数を減らしたり、あるいはカテゴリを何らかの方法で連続値に変換して用いるなどの対処を行う場合がある。

- ○ 特徴量ハッシング
- ○ ビンカウンティング

n0	n1	n2	n3	n4	n5	n6	n7	n8	n0
0	0	1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	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	1	0	0	0	0
0	0	1	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	0	0
0	0	0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0

特徴量ハッシング

特徴量ハッシングは、カテゴリ名（文字）をハッシュ関数に代入し、その出力を新しいカテゴリ名として、採用するような方法である。このとき、ハッシュ関数として n 種類のカテゴリを入力として受け取り、m 種類のカテゴリを出力するような関数を用いれば、カテゴリ名を減らすことが可能である。ただし、ハッシュ関数の制御が困難で、例えば右のように dog と tulip がおなじハッシュ名なったりするため、解釈が難しくなる。従って、解釈を重視する場合には、特徴量ハッシングは不適である。



- ハッシュ関数
  - md5
  - sha1
  - murmurhash3

文字を整数に変換してから、それを整数 k で割ると、その余は k 種類になる。このことを応用して、n 個のカテゴリを k 個のカテゴリに変換できるようになる。

species
cat
dog
mouse
cherry
tulip

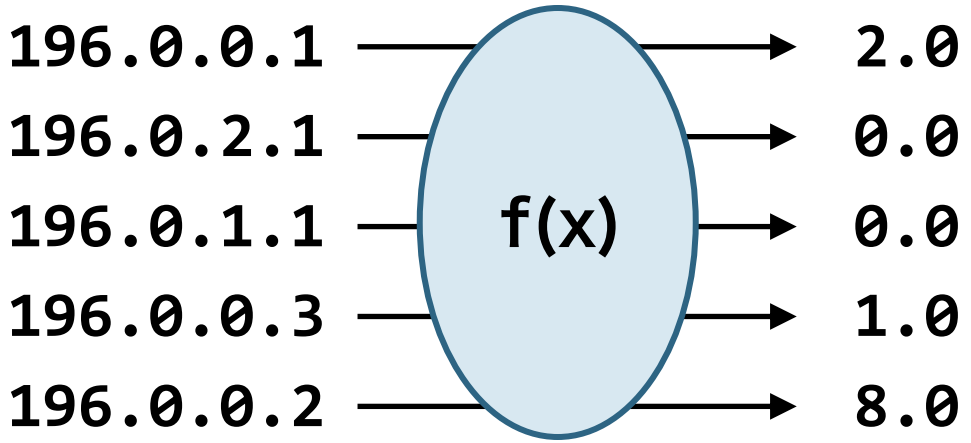
→

C1	C2	C3
1	0	0
0	1	0
0	0	1
0	0	0
0	1	0

Shi et al., http://proceedings.mlr.press/v5/shi09a.html
Weiberger et al., https://arxiv.org/abs/0902.2206

ビンカウンティング

カテゴリに対して、何らかの値を集計した統計量を計算して、カテゴリ名をその統計量で置き換える。例えば、IP アドレスを変換する場合は、DDoS 攻撃の攻撃頻度を統計量とみなすことができる。ただし、変換方法により、教師ラベルの情報が、特徴量として織り込まれることに注意する必要がある。



IP	stats
196.0.0.1	2.0
196.0.2.1	0.0
196.0.1.1	0.0
196.0.0.3	1.0
196.0.0.2	8.0

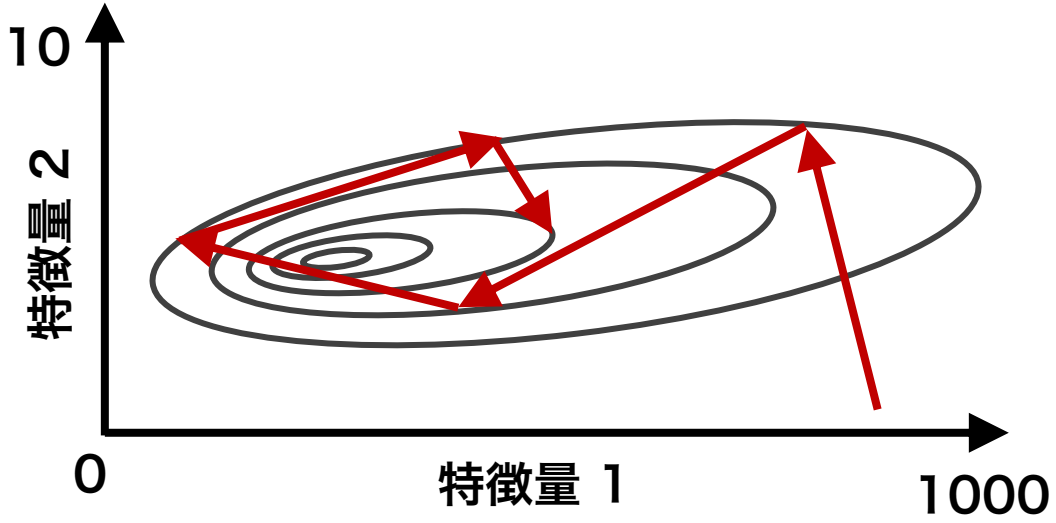
特徴量エンジニアリング

- カテゴリデータ
- 連続量データ

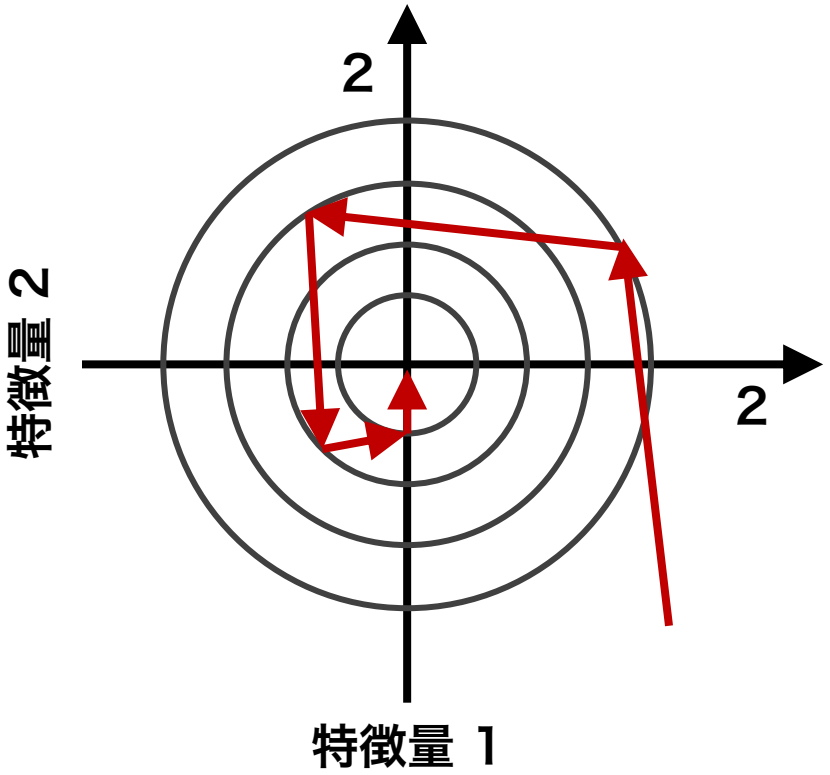
連続値の前処理

スケールの異なる特徴量を係数を推定するとき、推定結果がスケールの大きい特徴量の係数に影響されやすく、収束が遅くなる。このようなスケールの大きいデータに対して、階級化、二値化、対数化などを行うことで、モデルの予測性能を向上させる場合が多い。

- 離散化
  - 二値化
  - 階級化
- 対数化

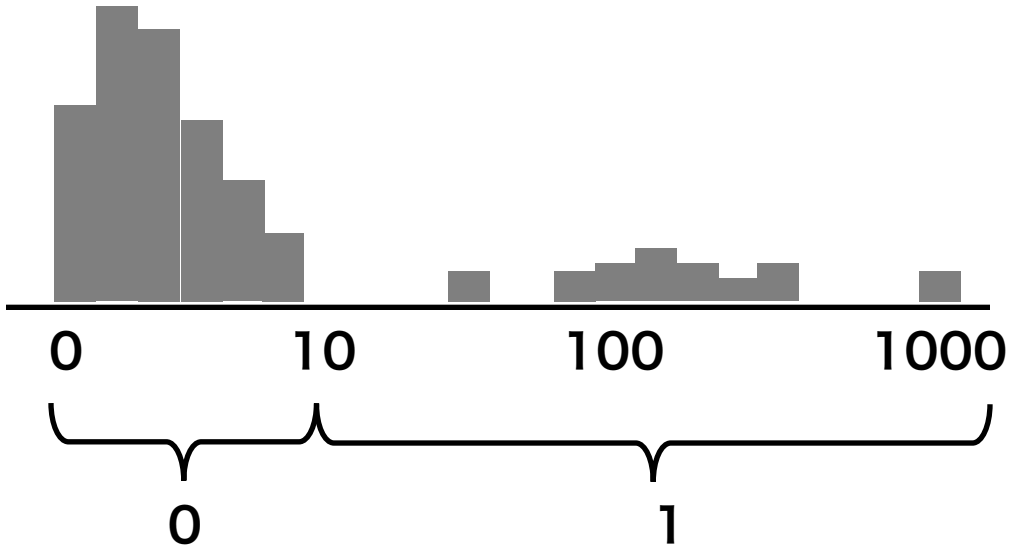


スケーリング



離散化（二値化）

特徴量のスケールが大きく、その分布が広いとき、二値化を行うことで予測性能を向上させることがある。例えば、右の例ように、データの 90% が 10 以下の値を取り、残りの 10% が 10 よりも大きい値をとるような場合は、10 を閾値としてデータを 0 または 1 に変換すると、スケールの影響を大きく抑えることができるようになる。閾値の設定は、データの分布やドメイン知識などを考慮して決める。



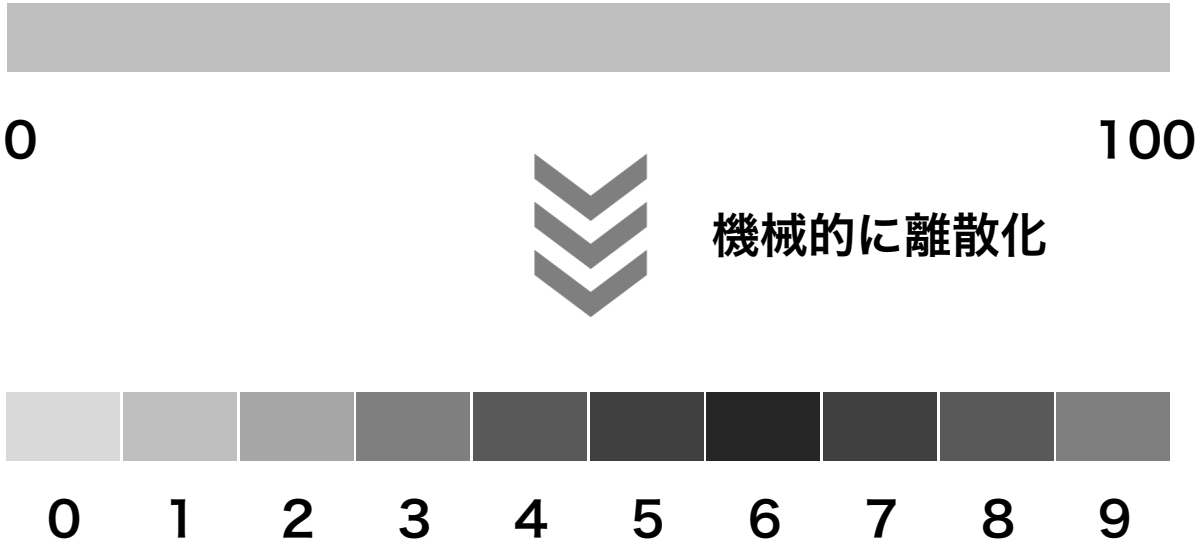
weight
19
203
28
4
430

→

k
0
1
1
0
1

離散化（階級化）

スケールの大きい特徴量を、いくつかの階級に分けることで、スケールを小さくすることができる。例えば、年齢データを 10 歳ごとに階級を決めて割り当てることで、0〜100 までのスケールをわずか 0〜9 までに縮小させることができる。



Age
21
24
32
35
28

→

Age
2
2
3
3
2

離散化（階級化）

スケールの大きい特徴量を、いくつかの階級に分けることで、スケールを小さくすることができる。例えば、年齢データを 10 歳ごとに階級を決めて割り当てることで、0〜100 までのスケールをわずか 0〜9 までに縮小させることができる。また、階級化は固定間隔で決める代わりに、ドメイン知識を使用して決めることもある。

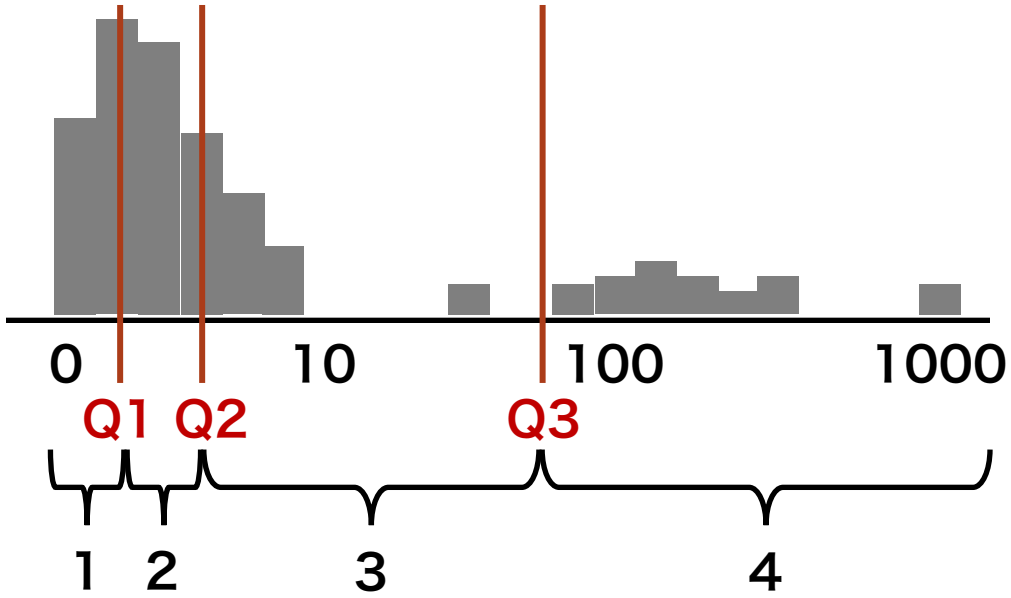
Age
21
24
32
35
28

→

Age
2
3
3
3
2

離散化（分位数）

q-分位数を利用した離散化も取られている。例えば、四分位数を使用して、データ全体を 1, 2, 3, 4 の 4 つの階級に分けることができる。



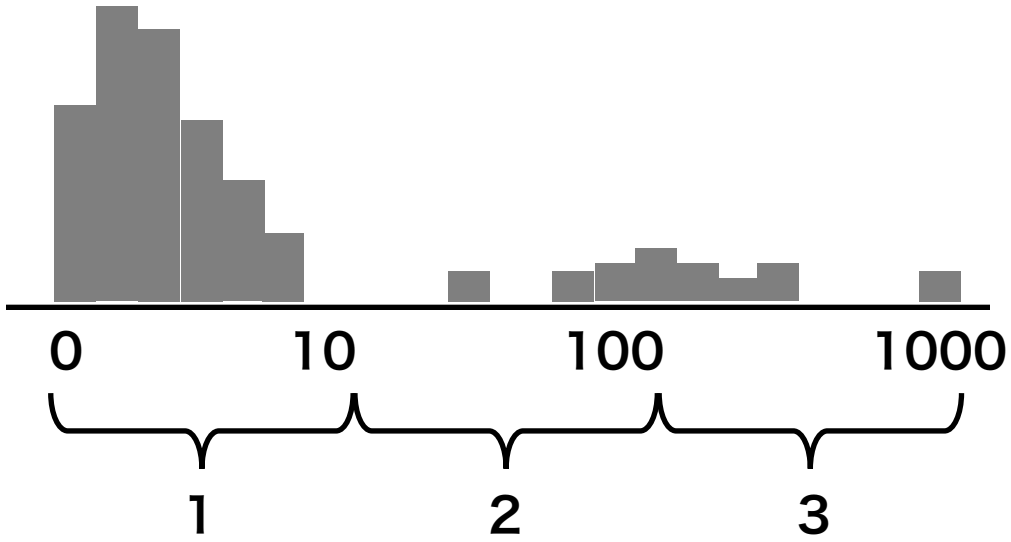
weight
19
203
28
4
430

→

k
3
4
3
1
4

離散化（累乗階級）

スケールの大きい特徴量に対して、10 の累乗で階級化することもある。



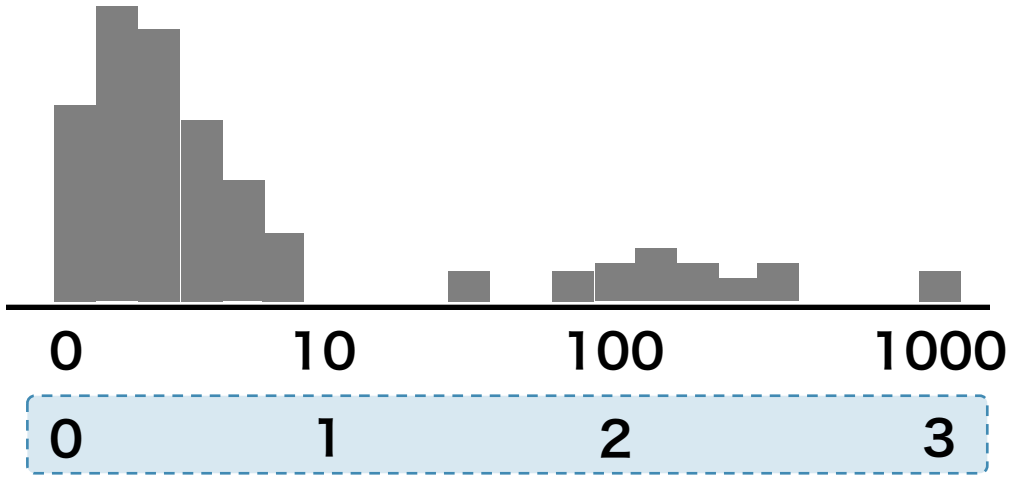
weight
19
203
28
4
430

→

k
2
3
2
1
3

対数化

スケールの大きい特徴量、とくに分布の裾の広い特徴量に対して、対数化することも、予測性能を上げるための有効な前処理法の一つである。対数化は、データに負の値が含まれると変換できないので、負の値が存在する場合は、定数を足してすべて正の値となるようにする。



weight	k
19	2.94
203	5.31
28	3.33
4	1.38
430	6.06

Box-Cox 変換

分散安定化変換は、分散が平均に依存しないように変数の分布を変換する操作である。その代表的な変換として Box-Cox 変換が一般的に知られている。対数変換や平方根変換なども Box-Cox 変換の特別な場合である。λは変換後のデータが最も正規分布に近くなるように決定される。

$$z_i = \begin{cases} \dfrac{x^{\lambda}}{\lambda}, & \lambda \neq 0 \\ \ln \lambda, & \lambda = 0 \end{cases}$$

スケーリング

機械学習を行うときは、基本的に特徴量をスケーリングした方が、学習が収束しやすいと言われている。スケーリングは、複数の特徴量のスケールを同じスケールに揃える作業である。スケーリングには、正規化と標準化の 2 種類が存在する。

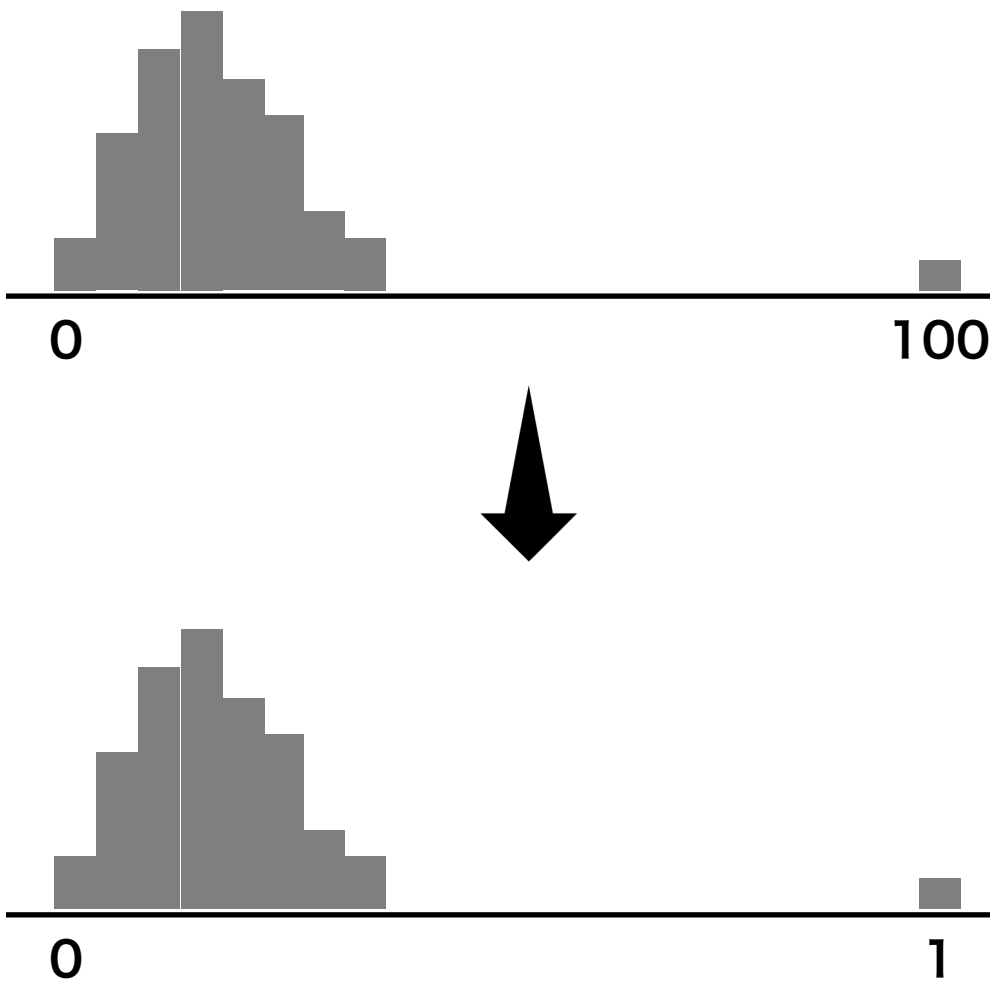
機械学習における正規化および標準化は、学習時の収束速度や予測時の予測性能の向上が目的として、行われている。統計における「データを正規分布に近づける」という目的ではないことに注意。

正規化

機械学習における正規化とは、特徴量をある値の範囲内に収めることである。例えば、特徴量の範囲を0≤x≤1 の範囲に変換する操作（min-max スケーリング）がなどが正規化にあたる。この変換により、データの最小値は 0、最大値は 1 となるので、上限と下限が存在しているデータに対して有効である。min-max スケーリングを行うことでデータの分布の形は変化しない。また、min-max スケーリングは、右図のように外れ値に弱いと言われている。

$$z_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

○ min-max スケーリング



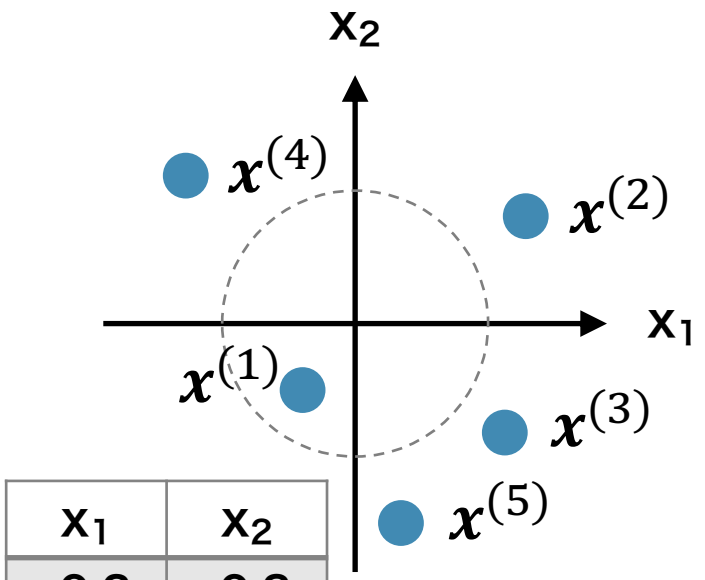
L2正規化

特徴量を特徴量空間上において、その距離を 1 となるような L2 正規化法もある。L2 正規化法は、各サンプルの全特徴量に対して、正規化を行っている。そのため、正規化後の各サンプルの分布は変化することに注意。

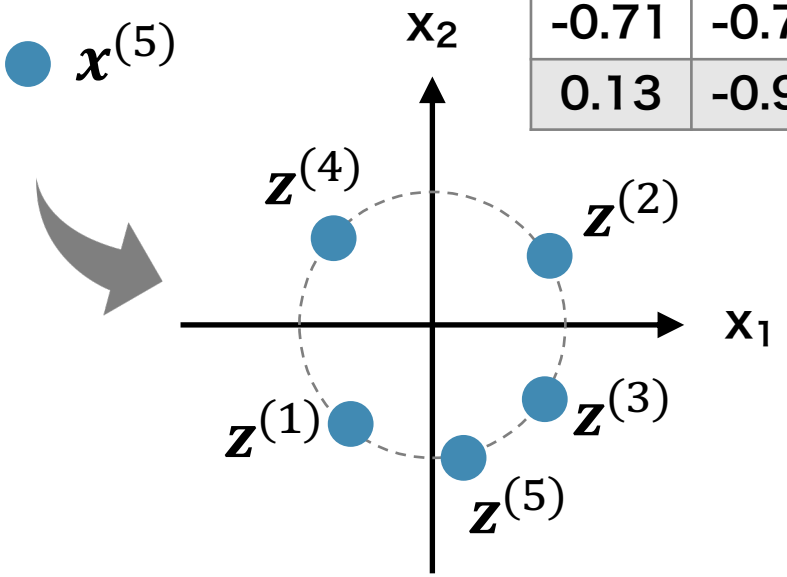
$$z^{(i)} = \frac{x^{(i)}}{\sqrt{\left(x_1^{(i)}\right)^2 + \left(x_2^{(i)}\right)^2 + \cdots + \left(x_p^{(i)}\right)^2}}$$

パラメーター推定時に、パラメーターの数が増えないように罰則をかけるときの L2 正則化と間違えないようすること。

L2 正規化



x_1	x_2
-0.2	-0.3
1.4	0.8
1.2	-0.7
-1.2	-1.2
0.2	-1.5

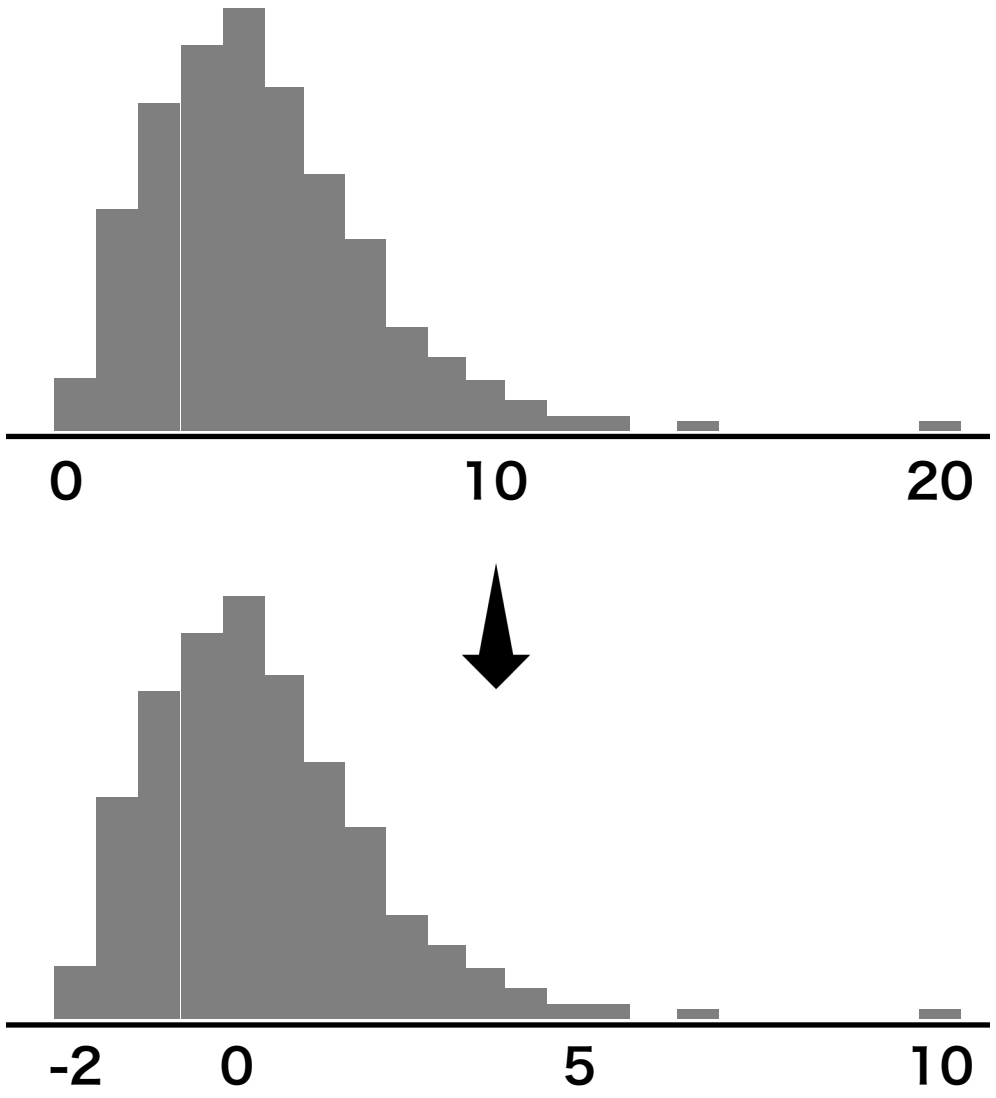


x'_1	x'_2
-0.54	-0.83
0.87	0.50
0.86	-0.50
-0.71	-0.71
0.13	-0.99

標準化

標準化は、データを平均 0、分散 1 のデータに変換している。標準化は、正規化に比べて、外れ値の影響をやや受けにくい。機械学習では、正規化に比べて、標準化がよく使われている。

$$z_i = \frac{x_i - \mu_x}{\sigma_x}$$

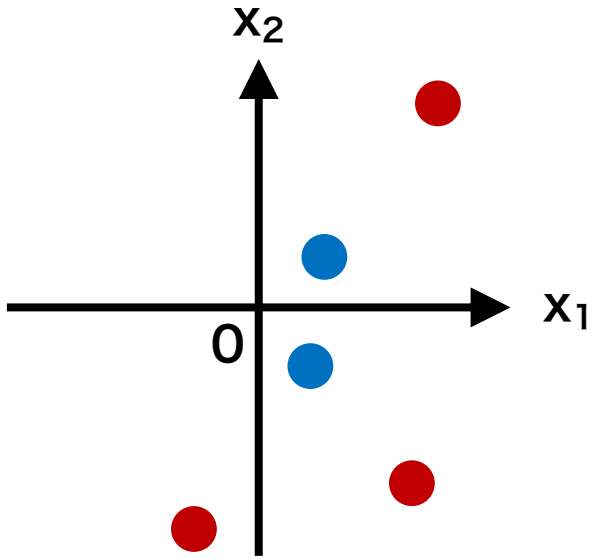


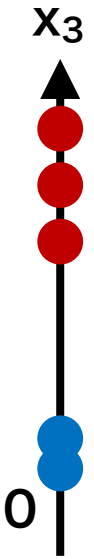
データ前処理

- 欠損値処理
- 特徴量エンジニアリング
- 特徴量選択
- 特徴量抽出

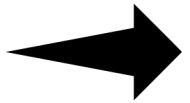
交互作用特徴量

特徴間の交互作用を考慮して、それらの組み合わせで新しい特徴量を作ることができる。新しい特徴量を作成するときは、機械的に、例えば、x_1 および x_2 の 2 つの特徴量が存在するとき、両者の積 x_1x_2 を新しい特徴量となるように作ることもあれば、ドメイン知識を利用して、例えば「$x_1 > 10$」かつ「$x_2 > 0$」のとき x_1+x_2、そうでなければ x_1x_2 となるような特徴量を作ることもある。機械的に特徴量を作成する場合、手間が少ないが、予測性能の向上に貢献しない特徴量も含まれる。これに対して、ドメイン知識を使用して特徴量を作成する場合、手間がかかるが、有効な特徴量である可能性が高い。





y	x_1	x_2
●	0.5	-0.6
●	0.6	0.7
●	0.3	0.2
●	-0.8	0.2
●	0.1	-0.3



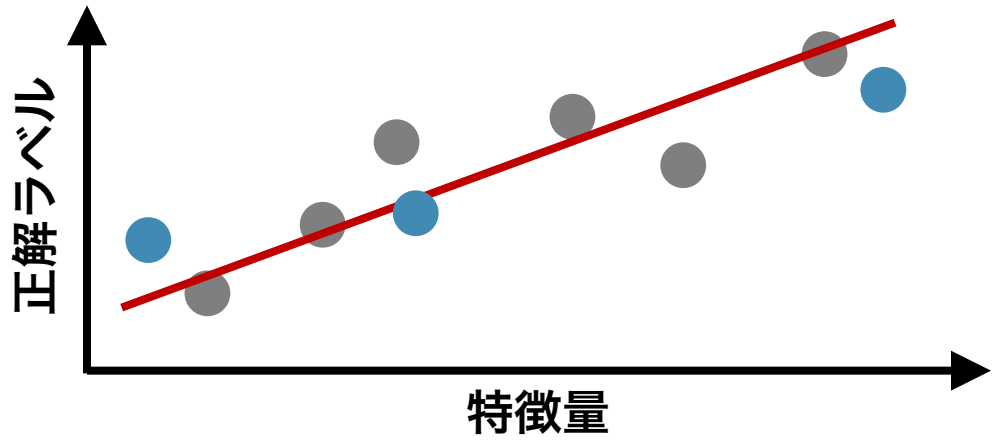
y	x_1	x_2	x_3
●	0.5	-0.6	0.61
●	0.6	0.7	0.85
●	0.3	0.2	0.13
●	-0.8	0.2	0.68
●	0.1	-0.3	0.10

特徴量選択

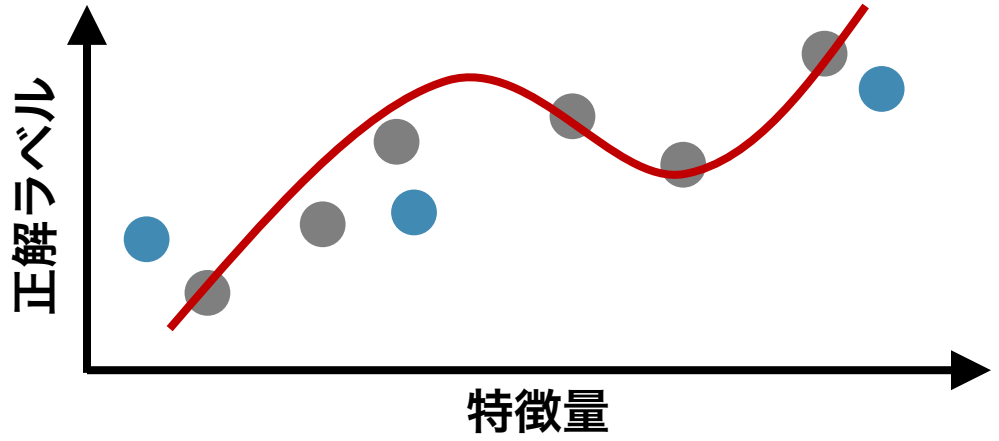
機械学習モデルが過度に複雑すぎる（パラメーター数が多すぎる）と、モデルが訓練データに過度にフィッティングしてしまい、汎用性を失うことがある。このことを過学習という。過学習を抑えるためには、データを増やしたり、パラメーターの数を減らしたりする必要がある。とくに、データが増やせない場合に、重要な特徴量だけを選択してモデルを構築することが、予測性能を高くする上で非常に役立つ。

- **フィルタ法**　特徴量と正解ラベルの相関や交互情報量等を計算して、閾値よりも小さければ削除する方法。
- **ラッパー法**　特徴量の一部だけを使用してモデルを構築し評価する。これを繰り返すことで最適な特徴量を選択する方法。
- **組み込み法**　学習アルゴリズムに組み込まれている特徴量選択法。決定木や L1 正則化などが組み込み法である。

- **単純なモデル**



- **複雑なモデル**



特徴量選択（フィルタ法）

フィルタ法では、各特徴量と正解ラベルの相関を調べ、その特徴量が正解ラベルを予測する上で有効かどうかを判定し、選択する手法である。ｐ 個の特徴量がある場合、ｐ 回の判定を行えばよく、計算コストが小さい。しかし、判定は、各特徴量と正解ラベルだけで判断され、特徴量同士の交互作用は考慮されず、そのうえ、予測結果も考慮されていないために、予測するために重要な特徴量が選択されない可能性がある。

- ○ 相関係数
- ○ 交互情報量
- ○ カイ２乗検定
- ○ フィッシャースコア
- ○ 回帰分析
- ○ 分散
- ○ 分散分析

特徴量選択（ラッパー法）

ラッパー法では、総当たりで特徴量を組み合わせて、予測モデルを作成し、評価することを繰り返して、特徴量の最適な組み合わせを選択する手法である。ラッパー法は、特徴量同士の交互作用や予測性能を考慮しているため有効な特徴量選択と言える。しかし、特徴量の数が多いとき、膨大な計算コストがかかる。

○ 逐次前進法

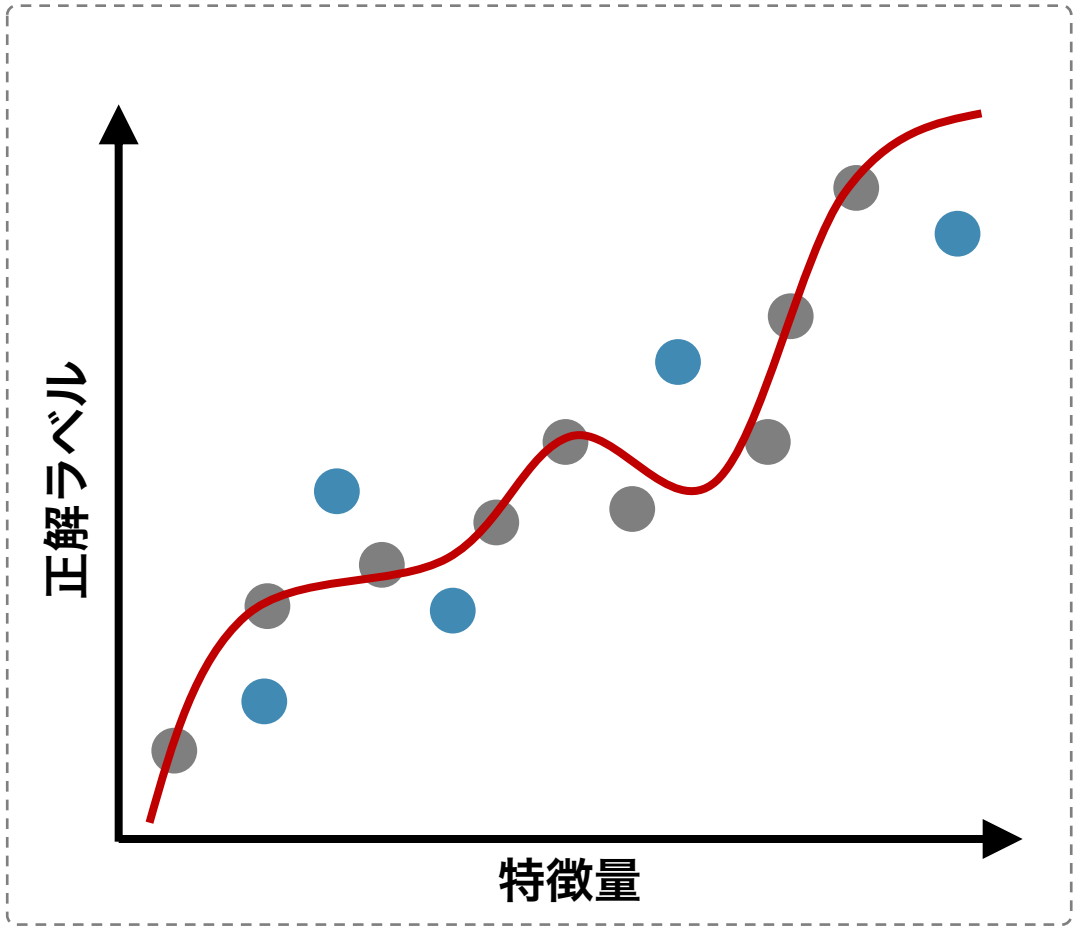
特徴量 1 つだけモデルの作成と評価を行い、最適な特徴量を決める。次に、選択された特徴量に加えて、もう 1 つの特徴量を足してモデルの作成と評価を行う。この操作を、予測性能が改善されなくなるまで繰り返す。

○ 逐次後退法

特徴量をすべて含むモデルから特徴量を 1 つだけ除去したモデルに対して評価を行い、最も不要な特徴量を取り除く。次に、そのモデルからさらにもう 1 つの特徴量を除去したモデルに対して評価を行い、最も不要な特徴量を取り除く。この操作を、予測性能が改善されなくなるまで繰り返す。

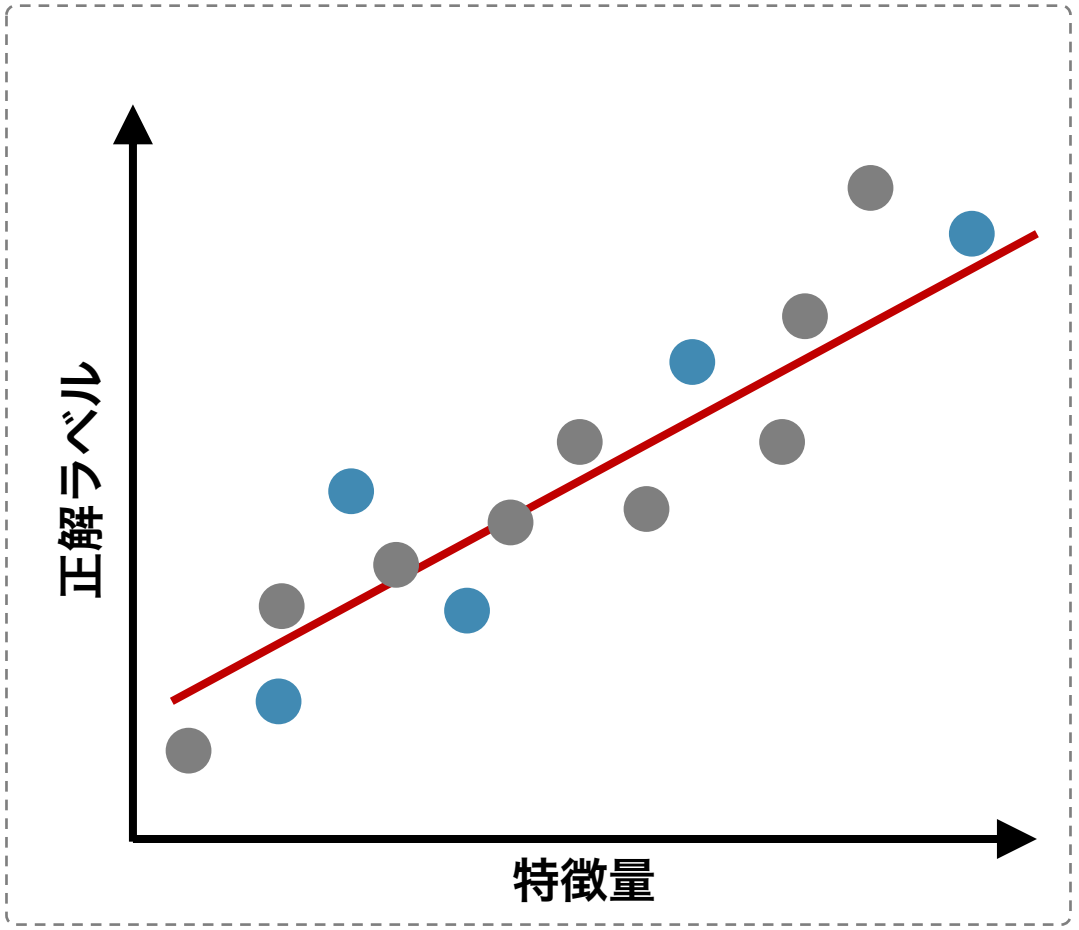
特徴量選択（正則化）

$y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$



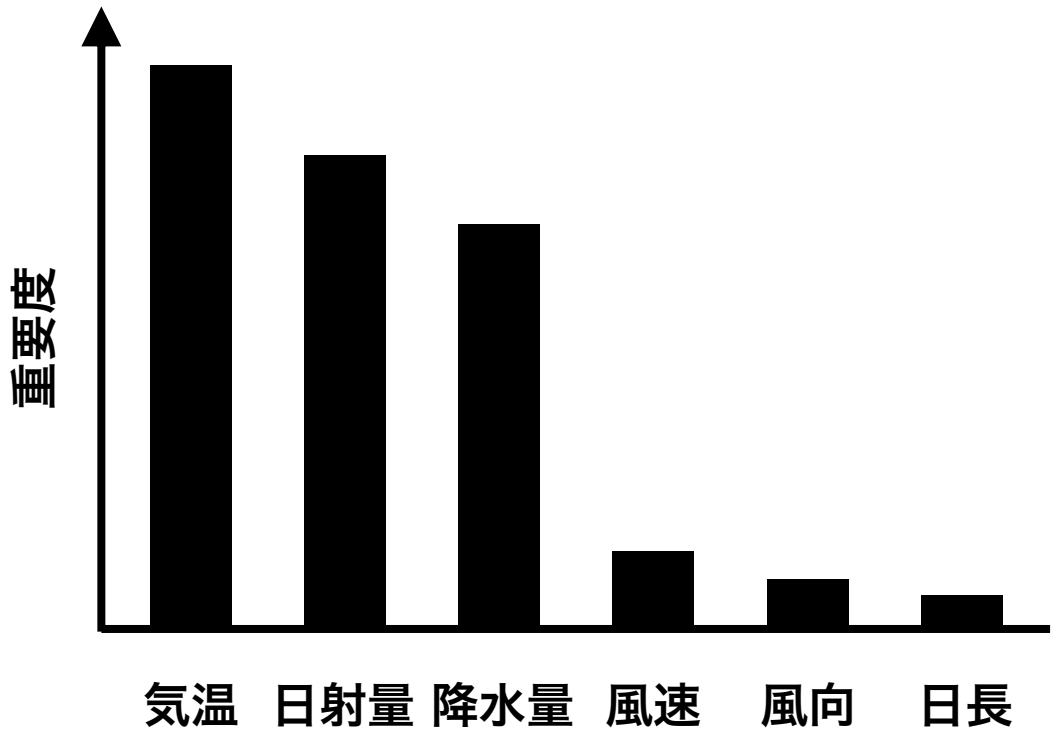
正則化

$y = w_0 + w_2x_2$



特徴量選択（ランダムフォレスト）

ランダムフォレストでモデルを構築すると、モデルに含まれる各特徴量の重要度が計算される。この重要度に基づいて特徴量を選択することもできる。なお、ランダムフォレストを利用した特徴量選択において、重要と思われる複数の特徴量が互いに強く相関している場合、ランダムフォレストはそのうち1特徴量だけの重要度を高く評価し、それ以外の特徴量を低く評価するので、注意が必要である。



データ前処理

- 欠損値処理
- 特徴量エンジニアリング
- 特徴量選択
- 特徴量抽出

特徴量抽出

- 主成分分析
- カーネル主成分分析
- 線形判別分析

特徴量抽出

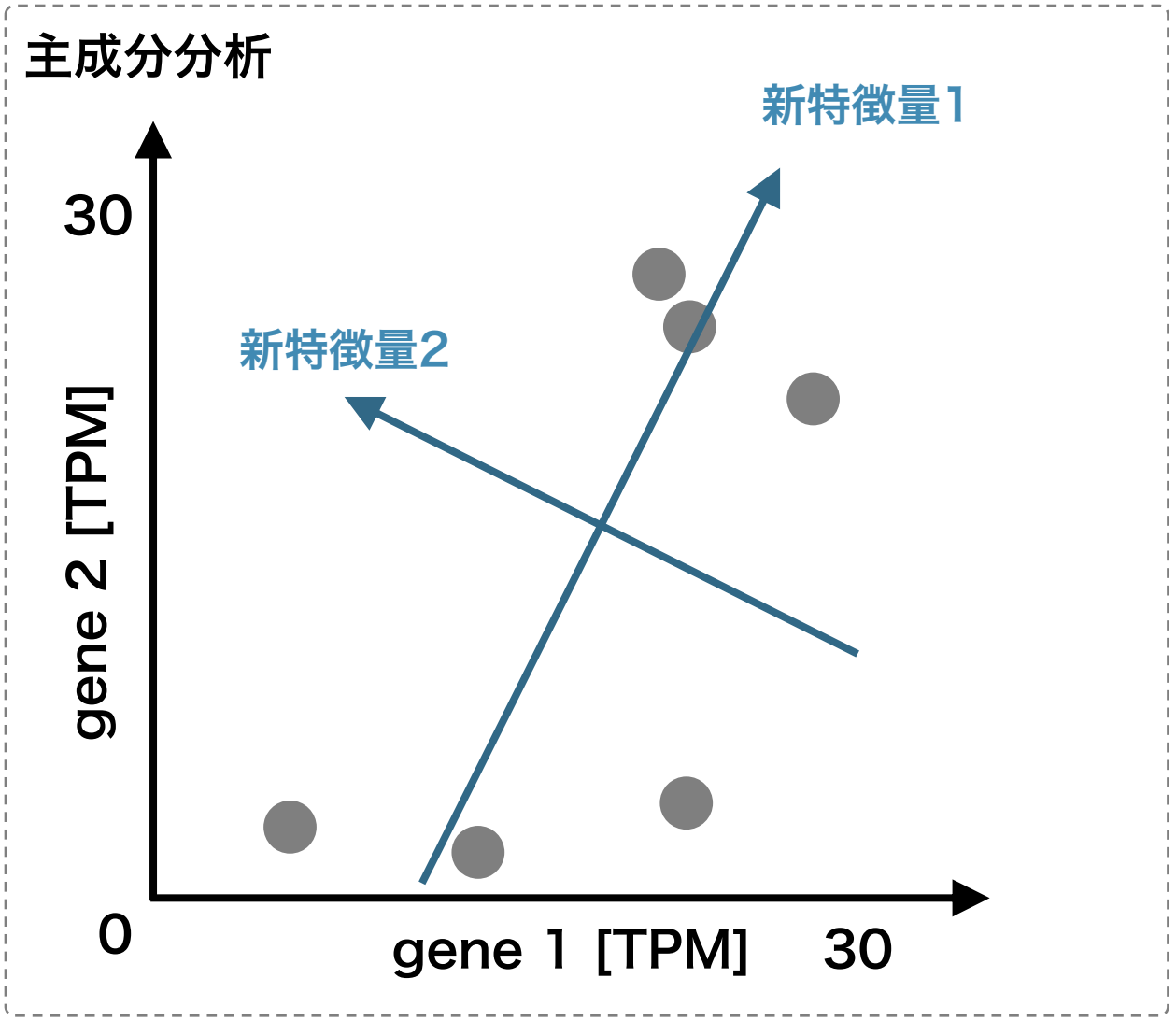
- 主成分分析
- カーネル主成分分析
- 線形判別分析

主成分分析

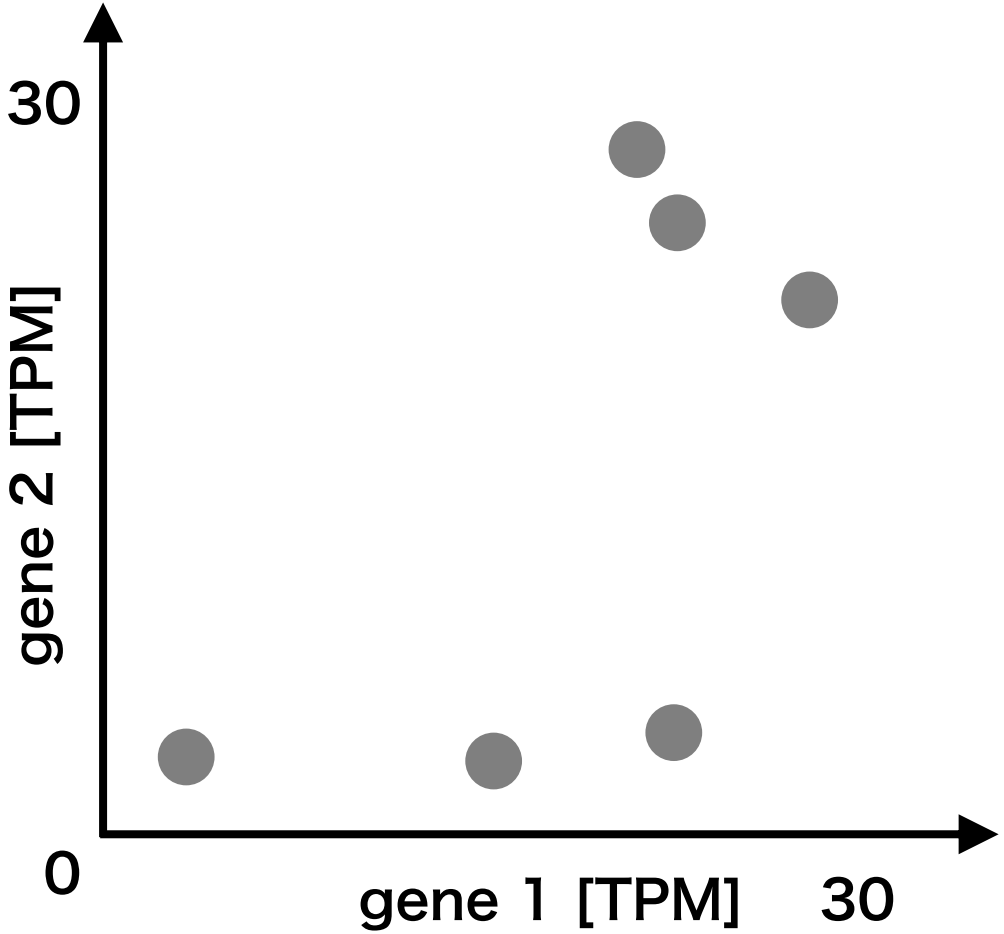
主成分分析は、次元削減の目的で使われている。具体的に、観測された p 次元 の 特徴量から、すべてのサンプルを最もよく分離できる 新 し い 特徴量を探している。

データ

id	gene 1	gene 2
CS1	20	26
CS2	27	20
CS3	23	21
CS4	5	3
CS5	18	4
CS6	22	4



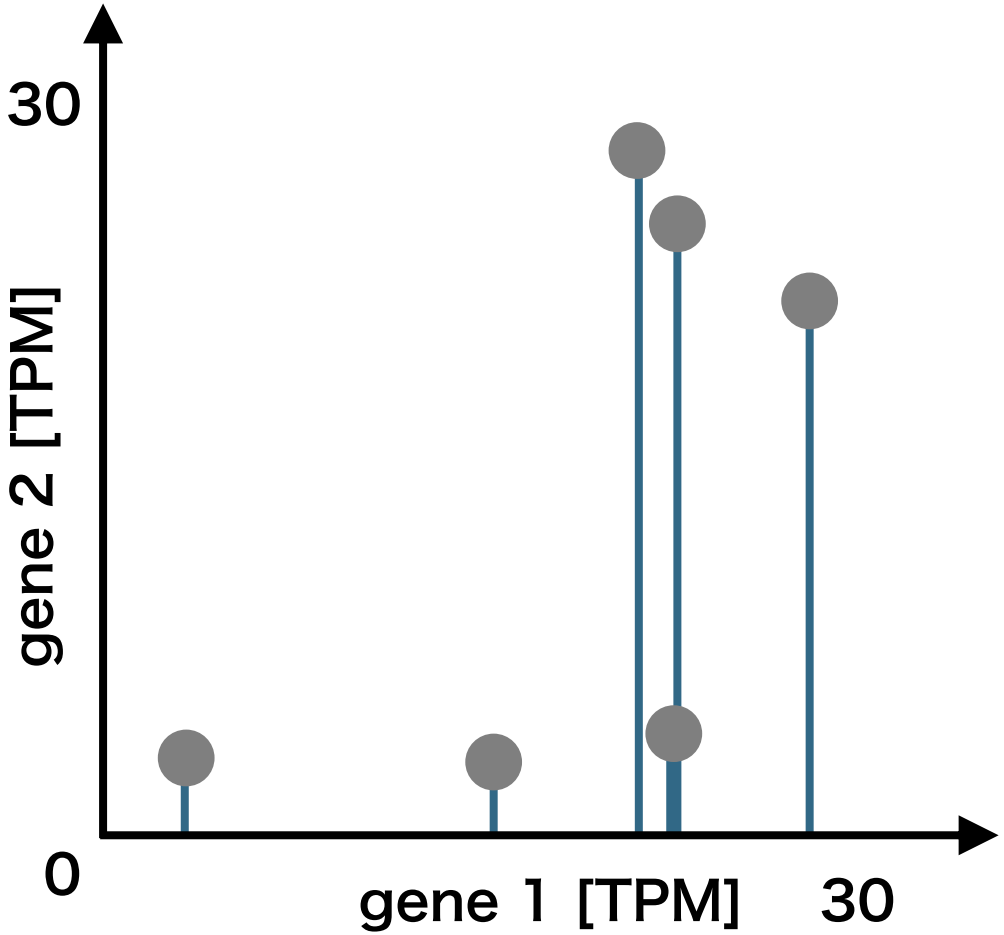
主成分分析



データの散らばり具合を測る指標として、一般的に分散が用いられる。データ同士の距離が近いと分散が小さく、距離が遠くなると分散も大きくなる。ある軸方向で見たとき、全データから分散を計算することで、その軸における全データがどれほど分離しているのかを把握できる。

$$Var[x] = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

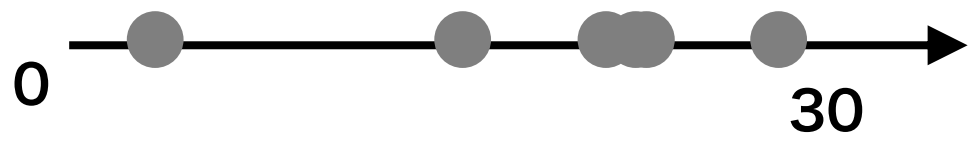
主成分分析



x 軸（gene 1 発現量）における分散

$$Var[x] = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

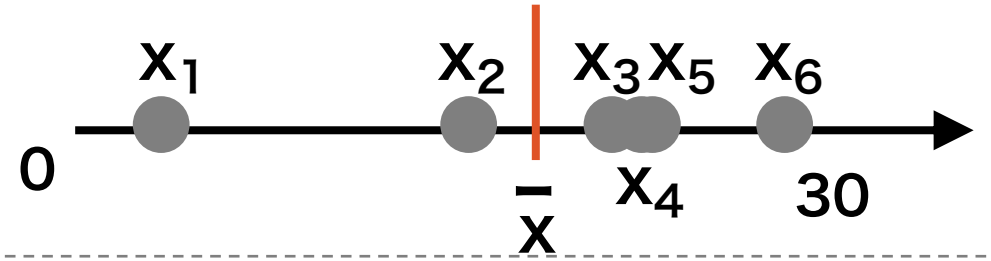
主成分分析



x 軸（gene 1 発現量）における分散

$$Var[x] = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

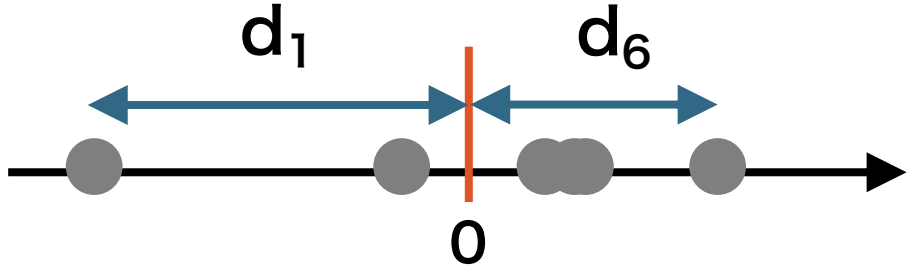
主成分分析



x 軸（gene 1 発現量）における分散

$$Var[x] = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

主成分分析



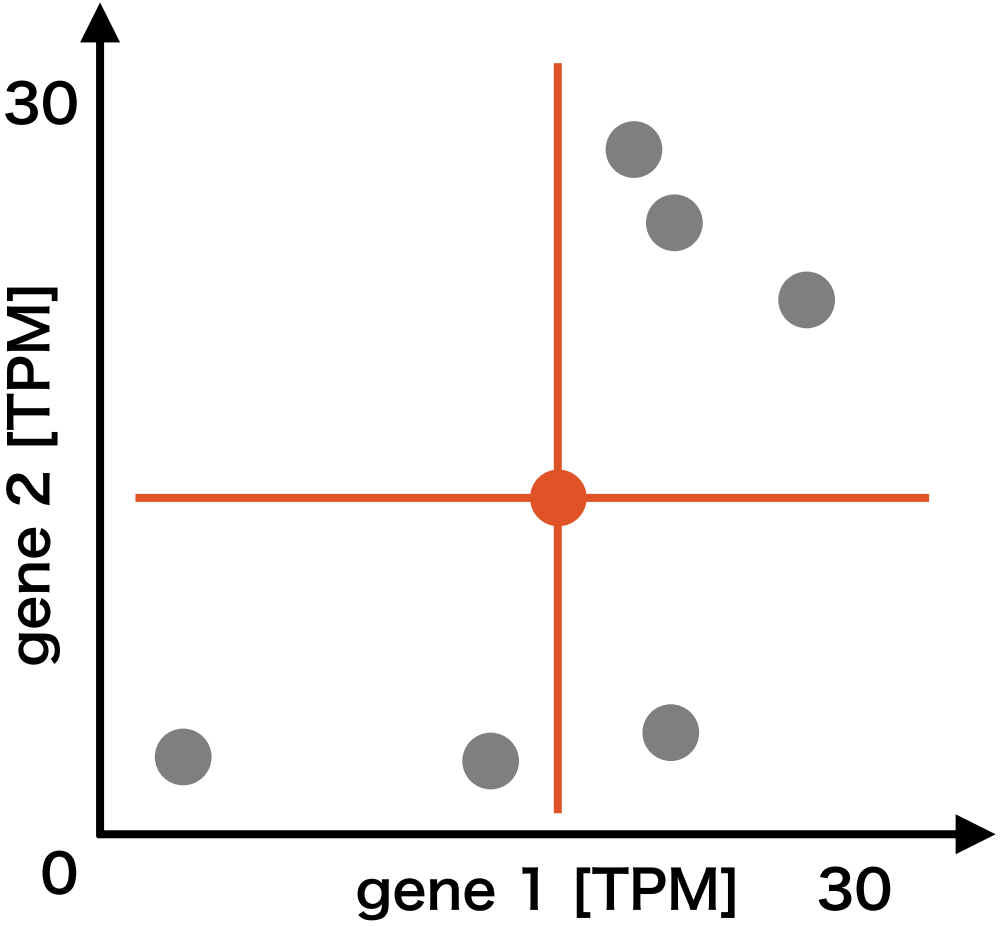
x 軸（gene 1 発現量）における分散

$$Var[x] = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

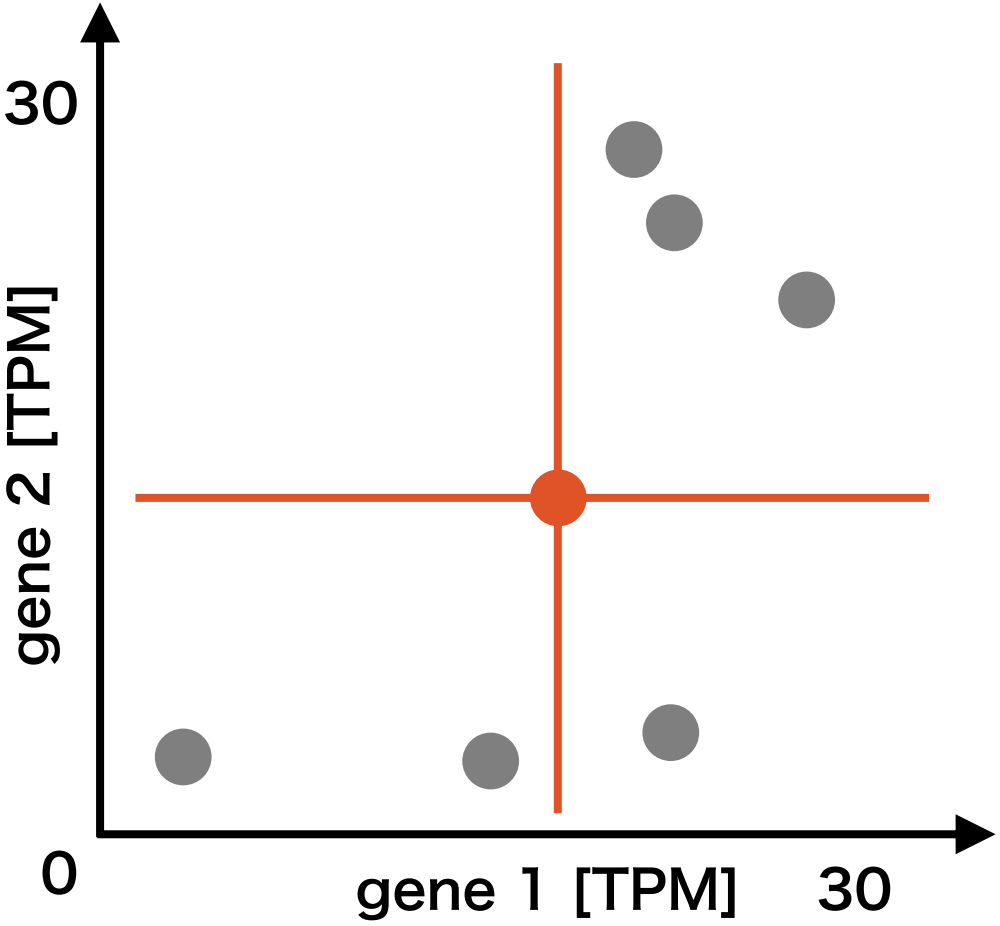
平均が 0 となるように平行移動しても一般性を失われない。

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

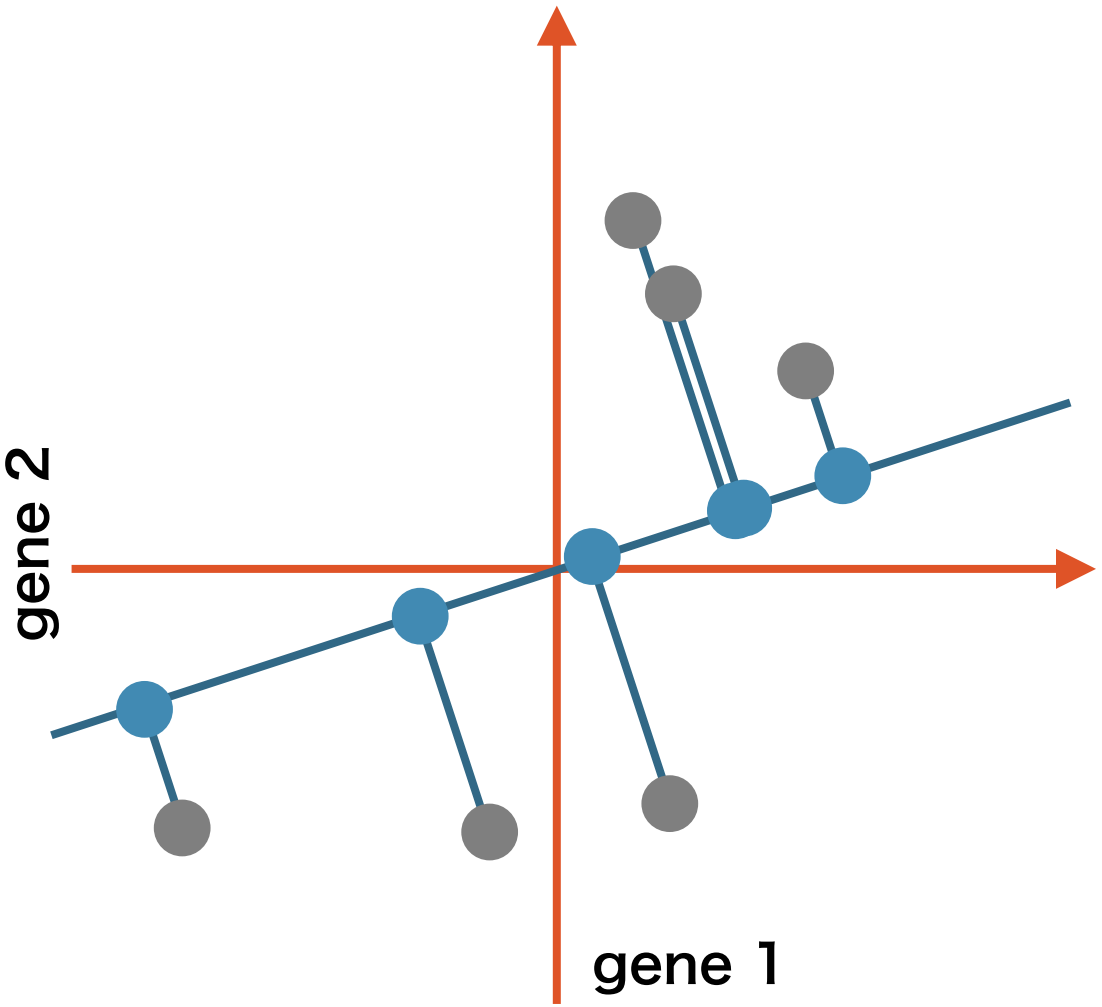
主成分分析



主成分分析



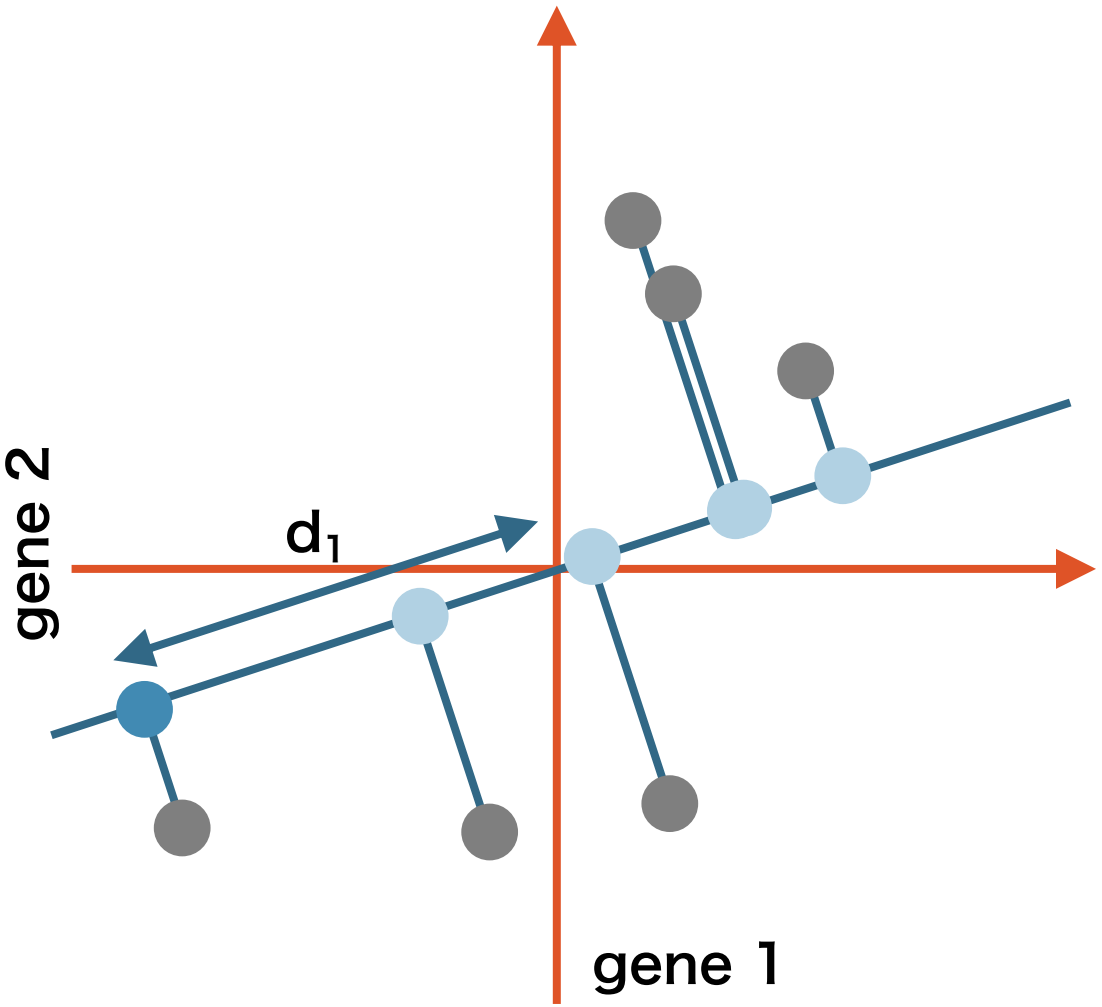
主成分分析



新しい軸における分散

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

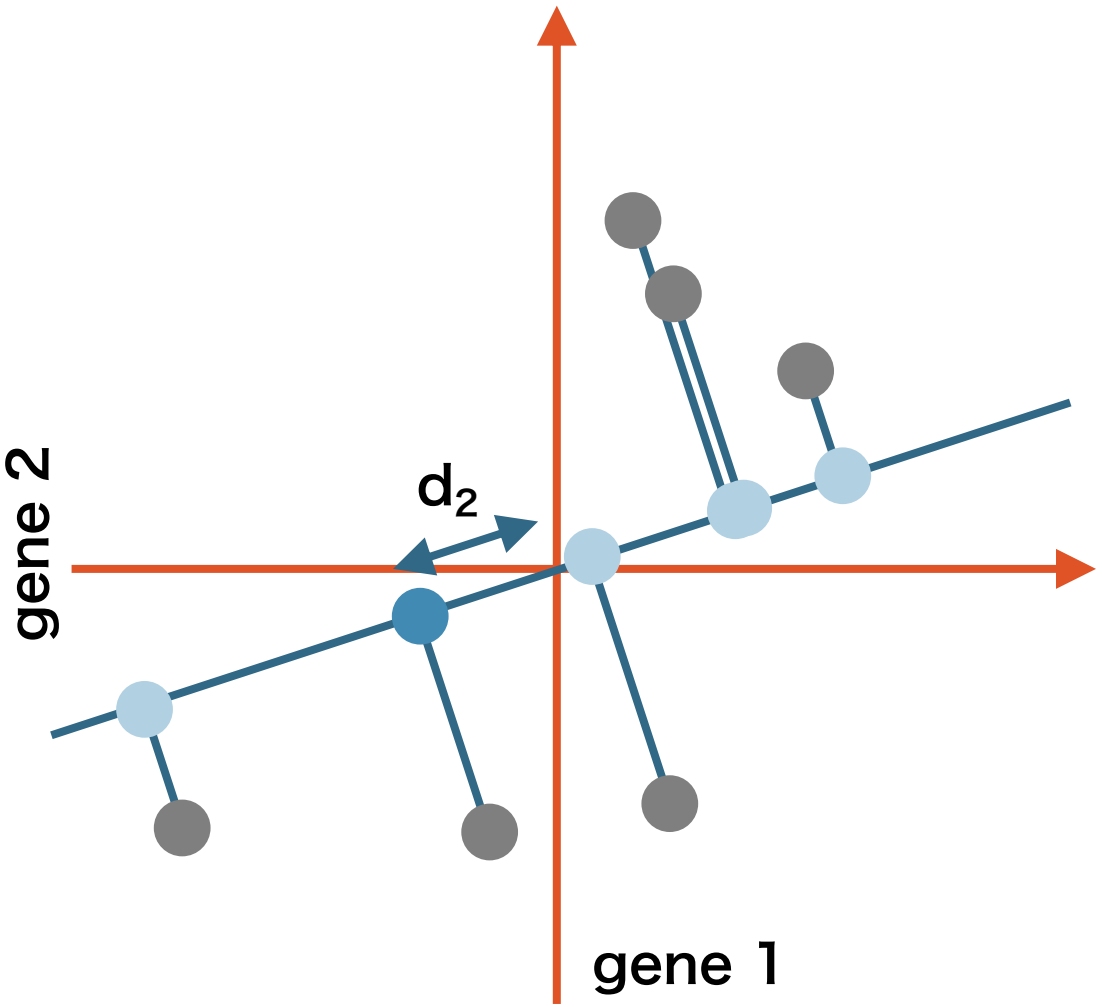
主成分分析



新しい軸における分散

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

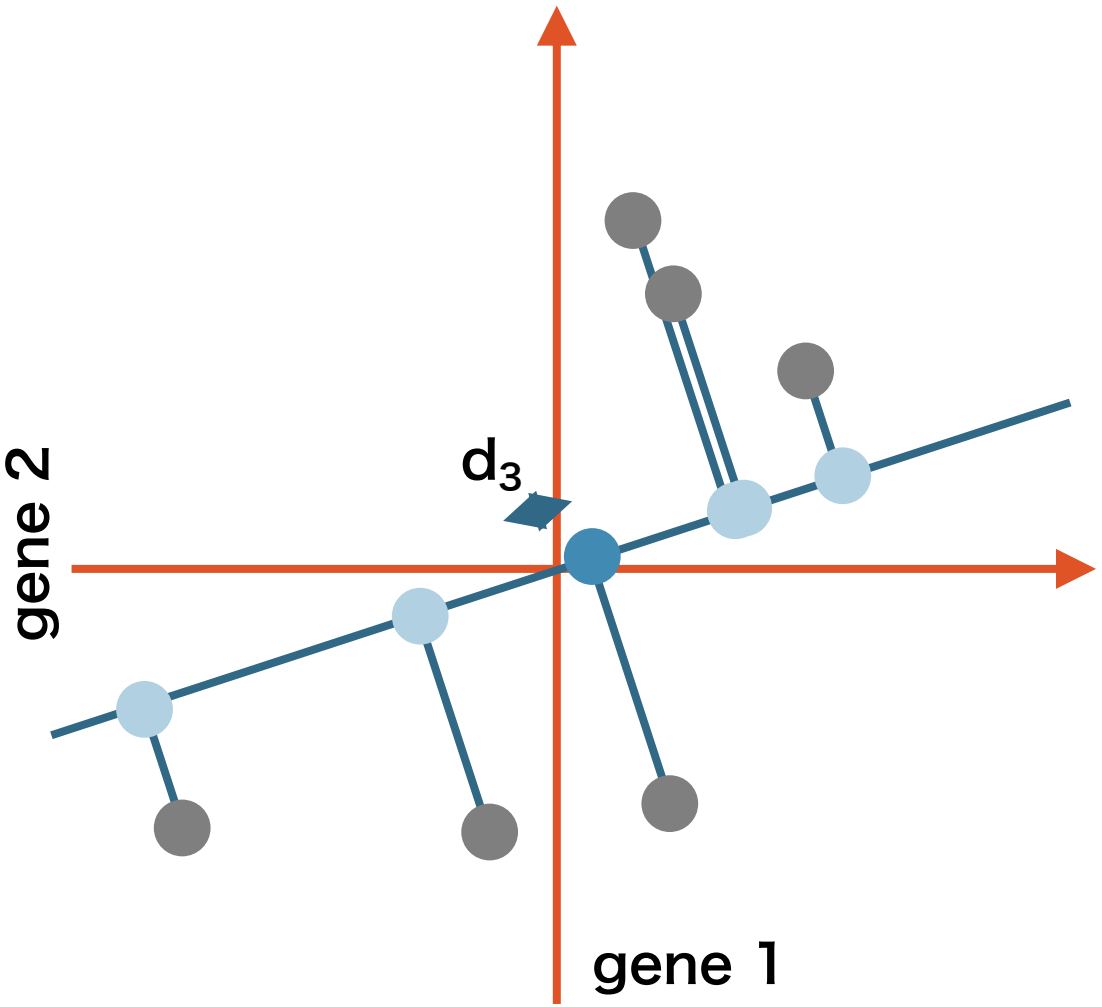
主成分分析



新しい軸における分散

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

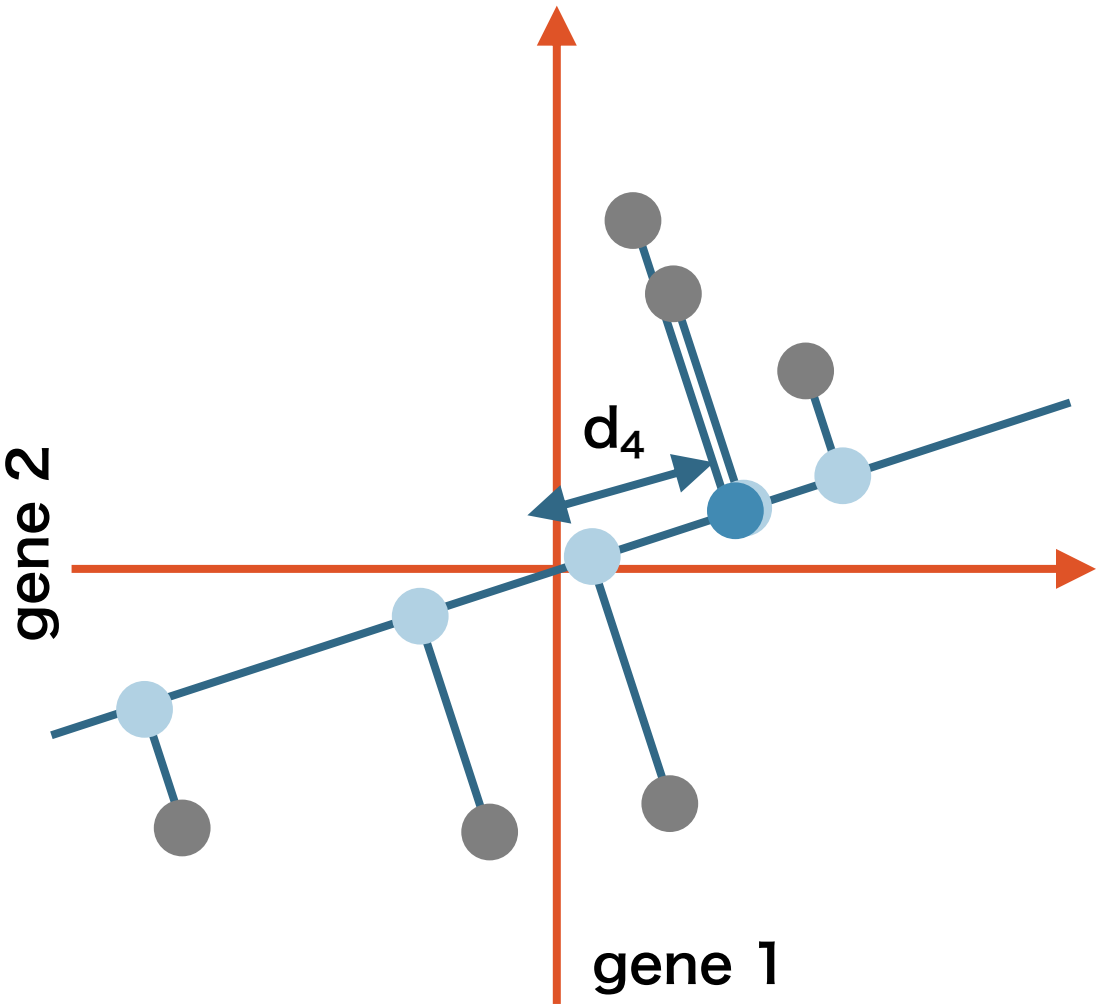
主成分分析



新しい軸における分散

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

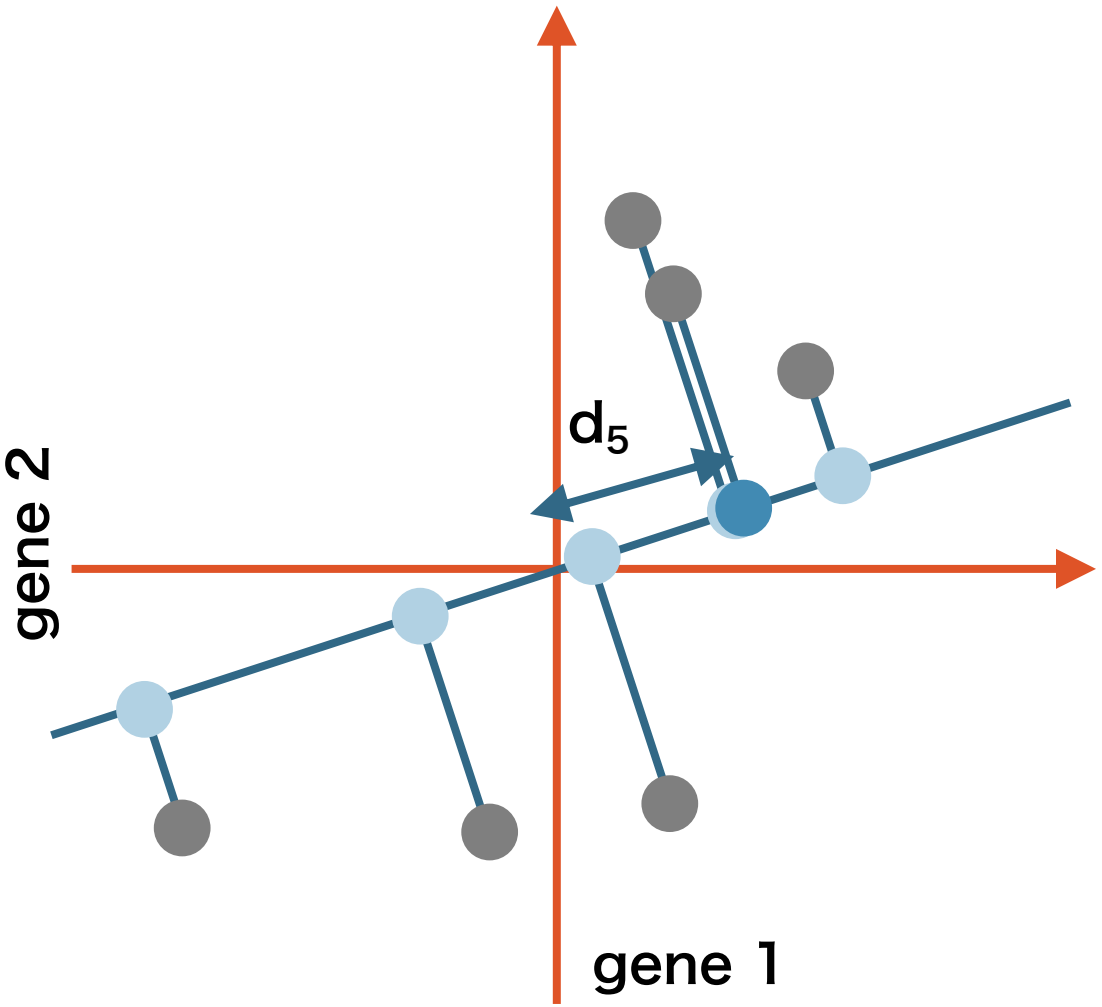
主成分分析



新しい軸における分散

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

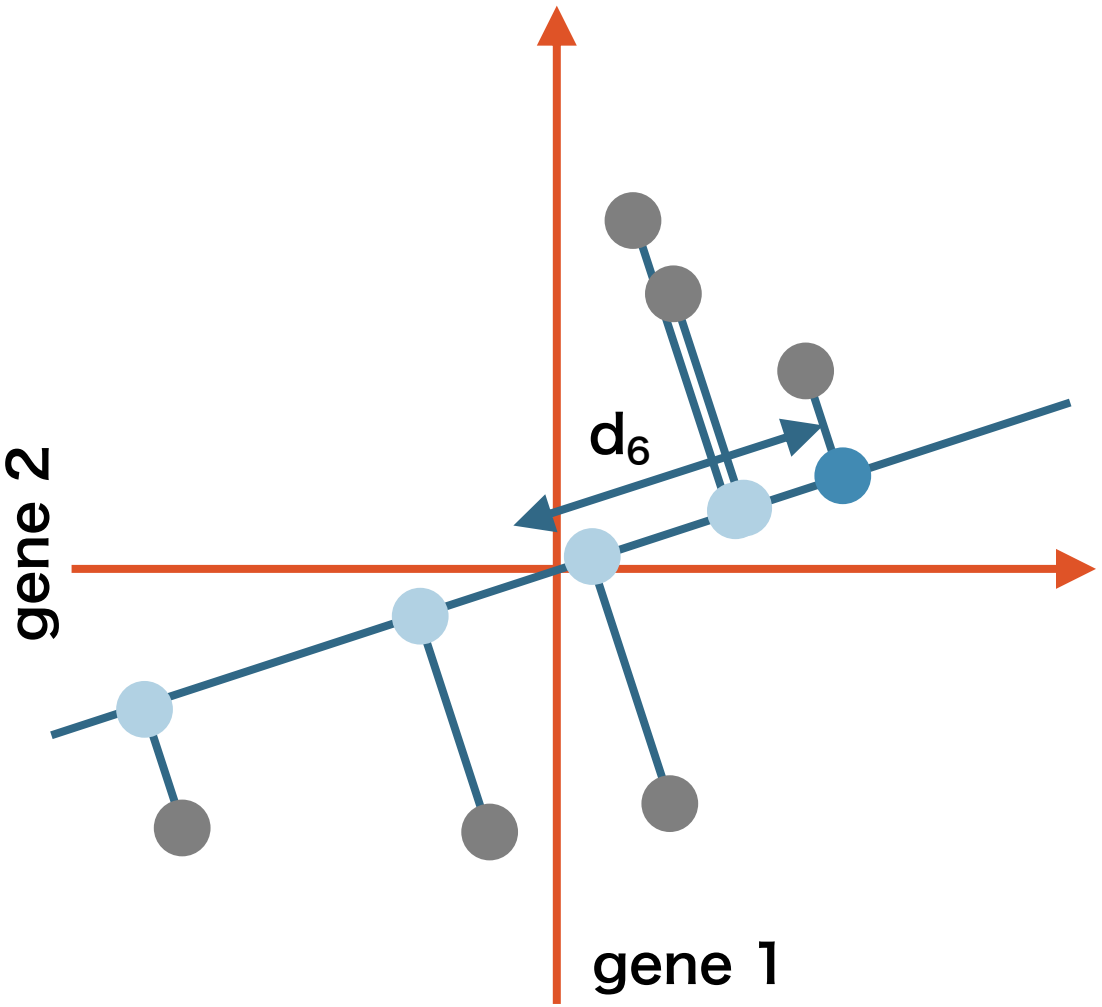
主成分分析



新しい軸における分散

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

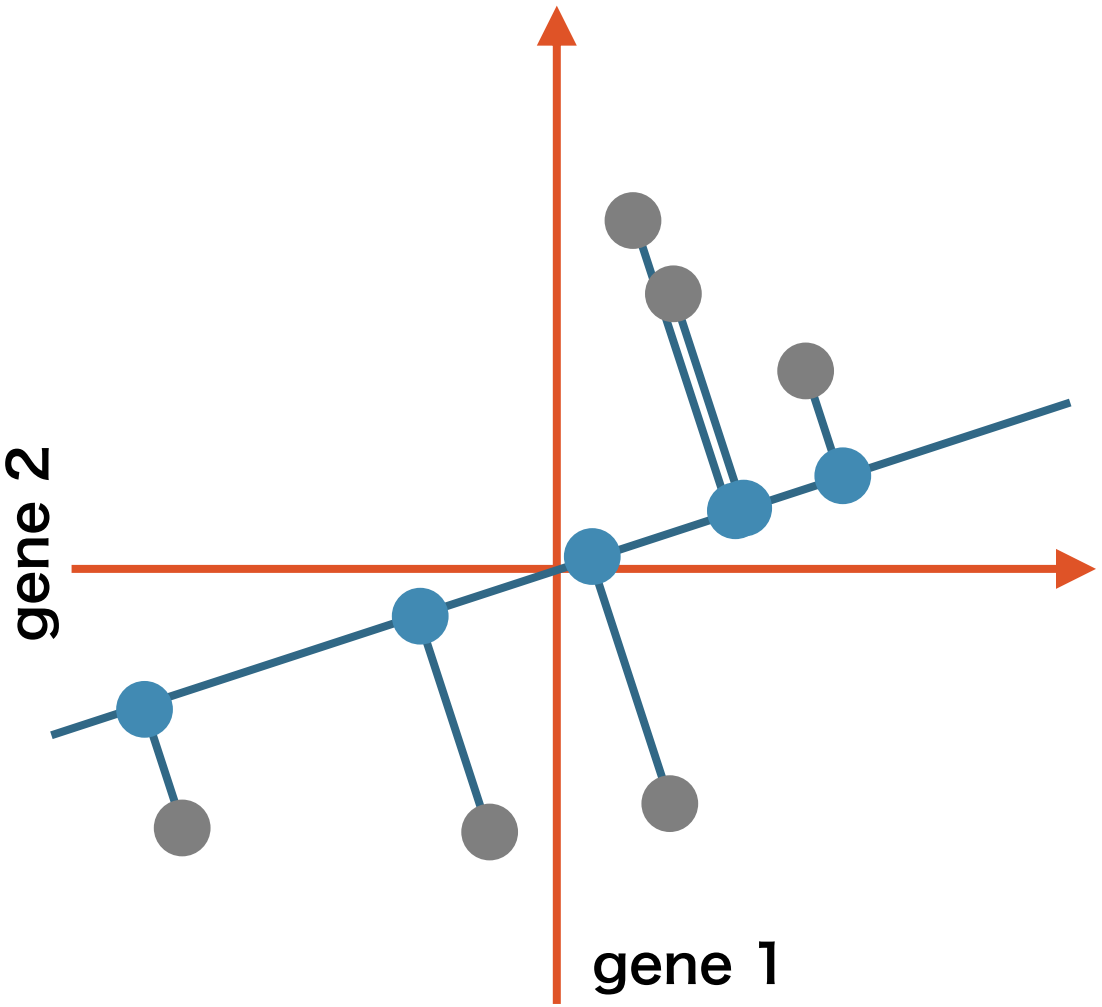
主成分分析



新しい軸における分散

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

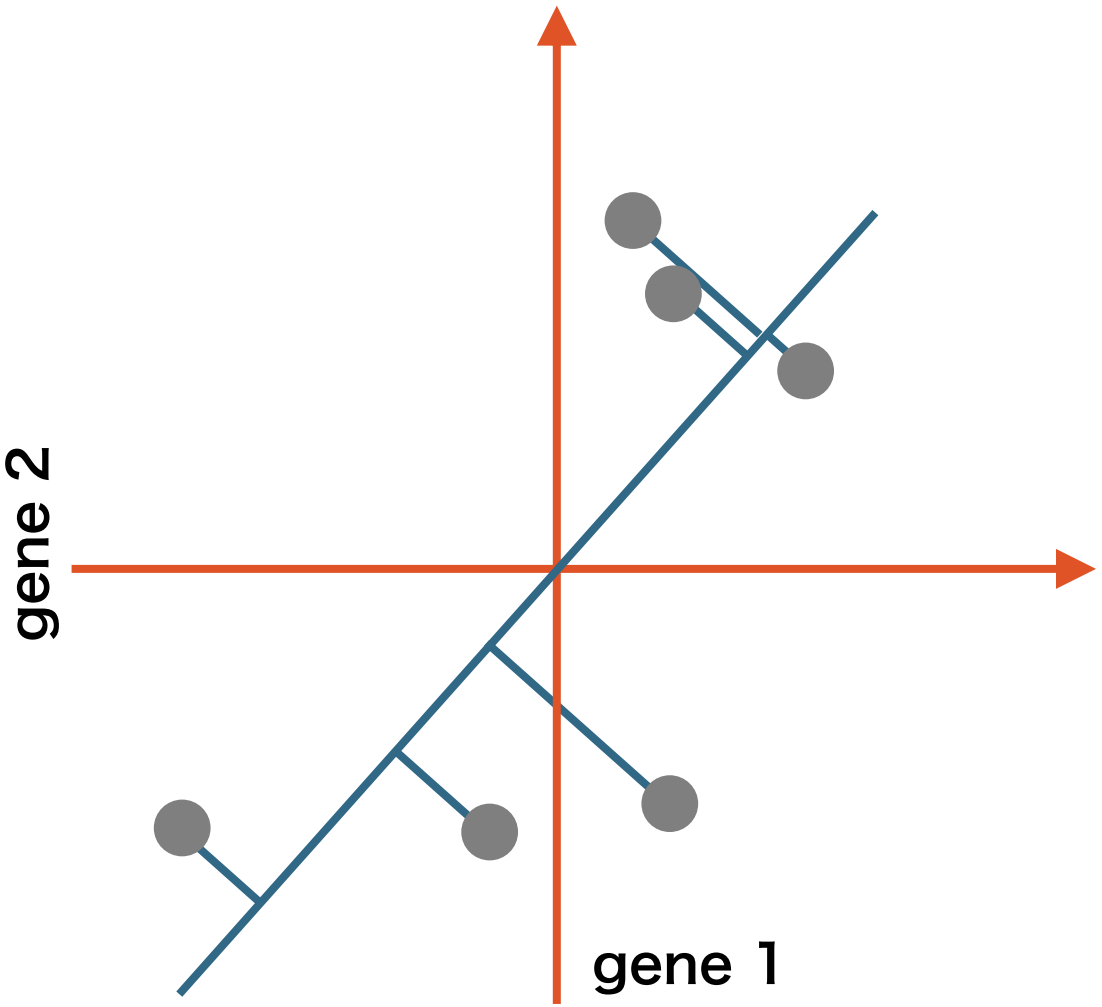
主成分分析



新しい軸における分散

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

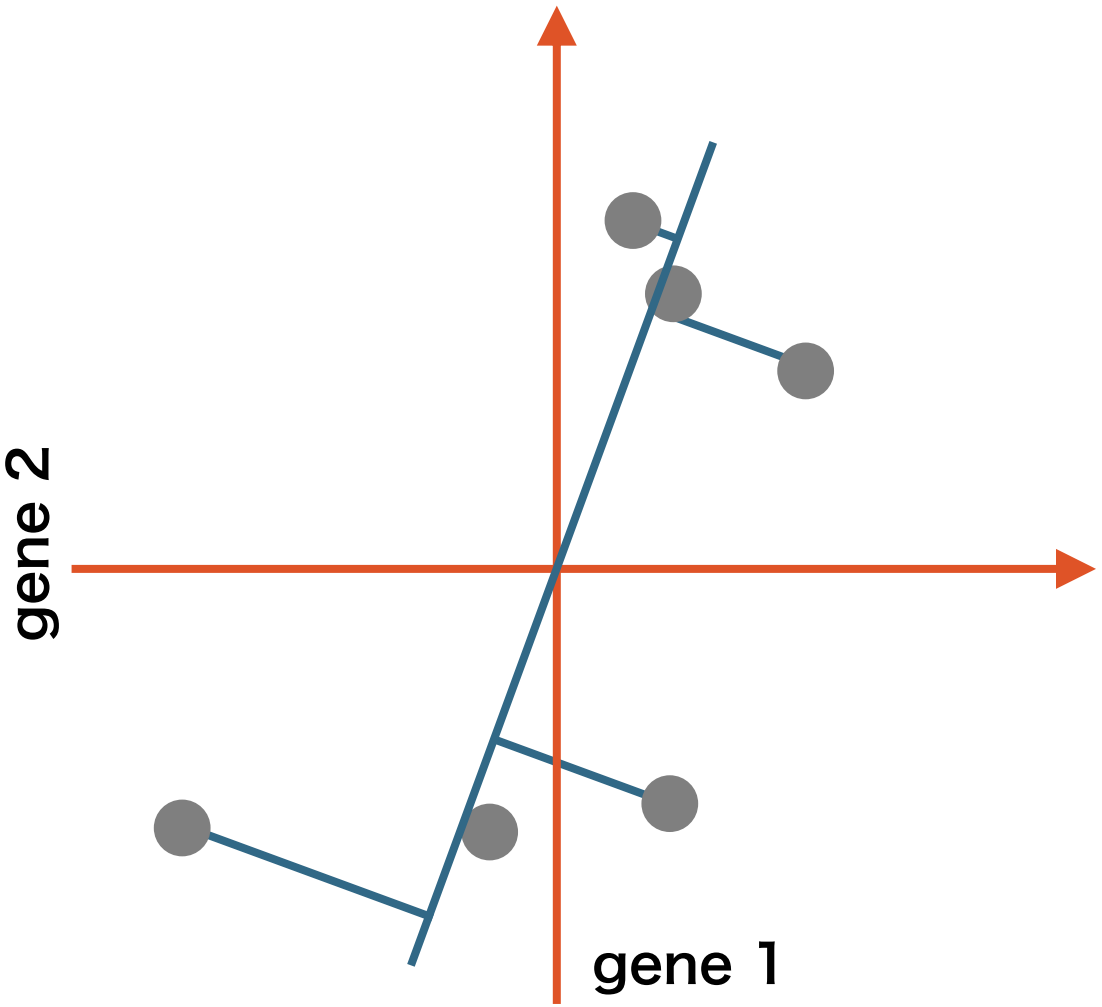
主成分分析



新しい軸における分散

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

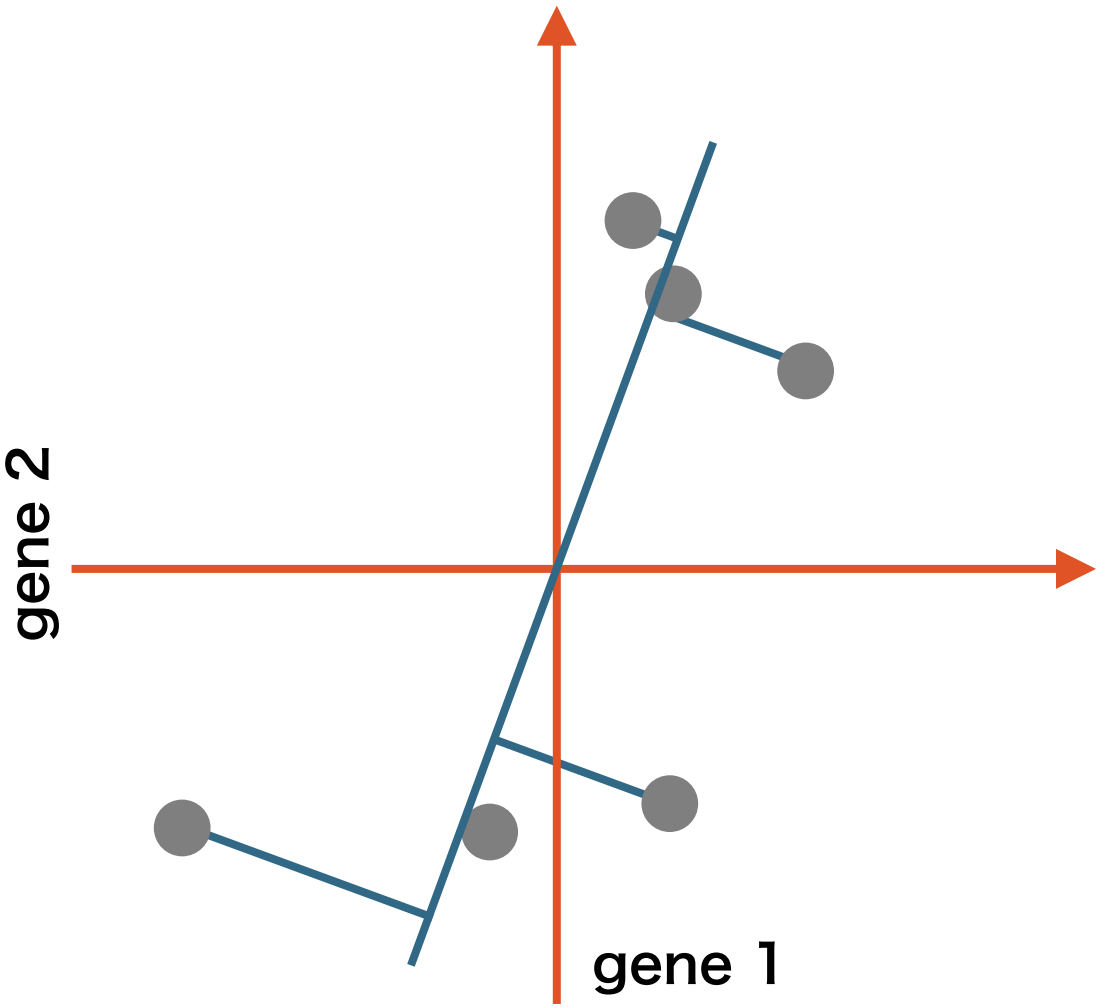
主成分分析



新しい軸における分散

$$Var[d] = \frac{1}{n-1}\sum_{i=1}^{n} d_i^2$$

主成分分析



1. 全データの重心を計算し、その重心が原点となるように平行移動する。この平行移動により、データ同士の距離や方向は変化しないことに注意。

2. 原点を通る直線（2次元データの場合）を 1 本描く。この直線を新しい座標軸 x' とみなしたとき、すべての点の x' 軸上の座標を求める。x' 軸における座標を使用して、データの分散を計算する。

3. 原点を通る直線を原点を中心に回転しながら、データの分散を求め、分散が最大となるような直線を求める。

4. 分散が最大となるような軸が見つかれば、次に、手順2〜3を繰り返して、その軸と垂直なるような新しい軸を見つける。この作業を繰り返して、入力データと同じ数だけの新しい軸を見つける。

分散共分散行列

$$\Sigma_X = \begin{bmatrix} Cov(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & Cov(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ Cov(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & Cov(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix} = \frac{1}{n} \boldsymbol{X'}^t \boldsymbol{X'}$$

ただし、 $Cov(\boldsymbol{x}_k, \boldsymbol{x}_l) = E[(\boldsymbol{x}_k - E[\boldsymbol{x}_k])(\boldsymbol{x}_l - E[\boldsymbol{x}_l])]$

$$= \frac{1}{n} \sum_{j=1}^{n} (x_{jk} - \bar{x}_k)(x_{jl} - \bar{x}_l)$$

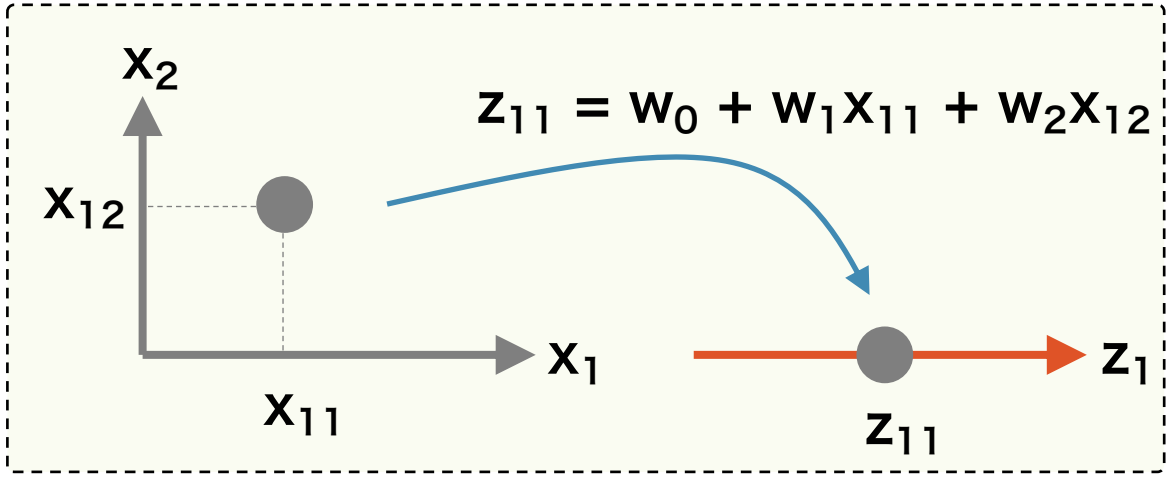
$$= \frac{1}{n} \boldsymbol{x'}_k^t \boldsymbol{x'}_l$$

$$\boldsymbol{x'}_k = (x_{1k} - \bar{x}_k \quad \cdots \quad x_{nk} - \bar{x}_k)^t$$

- 各特徴間の分散を表している。$Cov(x_k, x_l)$ は特徴 k と特徴 l の共分散を表す。対角線上の $Cov(x_k, x_k)$ は特徴 k の分散を表す。

分散

$$z = Xw$$



$$\Sigma_z = \frac{1}{n} z'^t z' = \frac{1}{n} (X'w)^t (X'w)$$

$$= \frac{1}{n} w^t X'^t X' w$$

$$= w^t \Sigma_X w$$

- 観測データ行列 X があり、行列 X に射影ベクトル w（n行1列）をかけて座標変換を行う。座標変換後のデータを z とする。この操作により n 個の特徴量が 1 つの点に射影される。
- ベクトル Z の分散Σ_zは、次のようにして、行列 X の分散共分散行列を用いて書き表すことができる。

第 1 主成分

$$\underset{\boldsymbol{w}}{\operatorname{argmax}}\, \Sigma_{\boldsymbol{z}}$$

$$= \underset{\boldsymbol{w}}{\operatorname{argmax}}\, \boldsymbol{w}^t \Sigma_{\boldsymbol{X}} \boldsymbol{w} \qquad \text{ただし、} \quad \|\boldsymbol{w}_k\| = 1$$

- 射影後の座標である z の分散 Σ_z が最も大きくなるような射影を考える。
- Σ_z を最大にする w を求める。ただし、w は回転のみを行う単位ベクトルであるから、制限条件として ||w|| = 1 である。

第 1 主成分

$$L(\boldsymbol{w}) = \boldsymbol{w}^t \Sigma_X \boldsymbol{w} - \lambda(\boldsymbol{w}^t \boldsymbol{w} - 1)$$

w で偏微分し 0 と置くと、

$$\frac{\partial L(\boldsymbol{w})}{\partial \boldsymbol{w}} = 2\Sigma_X \boldsymbol{w} - 2\lambda \boldsymbol{w} = \boldsymbol{0}$$

$$\Sigma_X \boldsymbol{w} = \lambda \boldsymbol{w}$$

固有値　固有ベクトル

- ラグランジュの未定乗数法で求めると、行列の固有値問題に帰着できる。

固有値問題

$$\Sigma_X \boldsymbol{w} = \lambda \boldsymbol{w}$$

この固有値を問題を解くために、$|\Sigma_X - \lambda I| = 0$ を満たす λ をまず計算する。ただし、I は単位行列を表す。

$$|\Sigma_X - \lambda \mathbf{I}| = \mathbf{0}$$

これにより、p 個の解 λ が得られる。これらの λ を大きい順に $\lambda_1 > \lambda_2 > \ldots > \lambda_p$ とする。

- 行列 X が n 行 p 列（p 個の特徴量を含む n 個のデータ）のとき、Σ_X は p 行 p 列になる。
- そのため、固有値問題を解く際に、λ も p 個計算される。

固有値問題

固有値　$\boldsymbol{\lambda} = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_p \end{pmatrix}$　$\lambda_1 > \lambda_2 > \cdots > \lambda_p$

すると、それぞれのλ_jに対して、$\Sigma_X w = \lambda_j w$ を満たす固有ベクトル w_j が計算される。

固有ベクトル　$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{w}_1 \\ \vdots \\ \boldsymbol{w}_p \end{pmatrix}$

- 行列 X が n 行 p 列（p 個の特徴量を含む n 個のデータ）のとき、Σ_X は p 行 p 列になる。
- そのため、固有値問題を解く際に、λ も p 個計算される。

固有値問題と主成分分析

λ 固有値：第１主成分の分散

$$\Sigma_z = \boldsymbol{w}^t \Sigma_X \boldsymbol{w} = \boldsymbol{w}^t \lambda \boldsymbol{w} = \lambda \boldsymbol{w}^t \boldsymbol{w} = \lambda$$

$\boldsymbol{w}$ 固有ベクトル：座標変換を行う回転行列

$$\boldsymbol{z} = \boldsymbol{X}\boldsymbol{w}$$

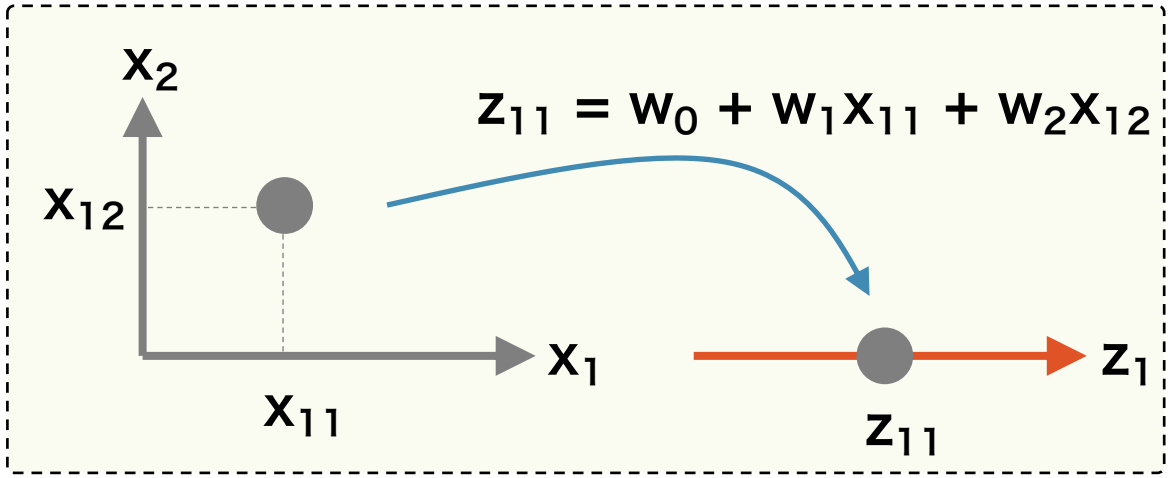
$\boldsymbol{z}$ 主成分得点

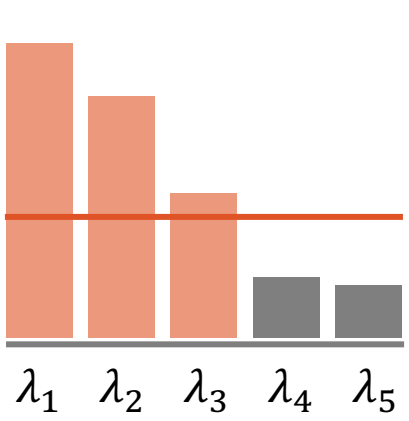
$$\boldsymbol{z} = \boldsymbol{X}\boldsymbol{w}$$

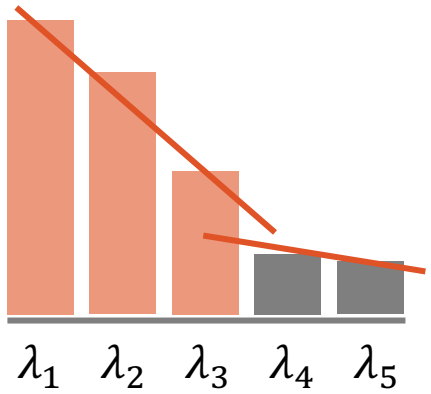
- 固有値λは、主成分の寄与率の計算に使われる。

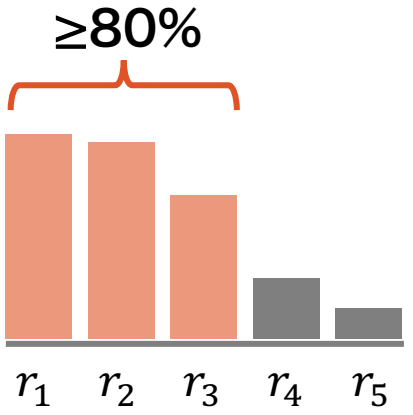
主成分得点

$$z = Xw$$









- 主成分分析は p 次元のデータを低次元に射影する次元削減の目的で使われる。
- 何次元に射影するか（主成分をいくつ選ぶか）は、研究分野目的によって異なる。
  - 固有値がある閾値以上のとき
  - 固有値の変化が緩やかになったとき
  - 累積寄与率が 80~90% を超えたとき
  - 寄与率が 1/p を越えるとき

寄与率

各主成分の分散λ_1、 λ_2、・・・、 λ_pが与えられたとき、第 k 主成分の寄与率は、次のように計算される。

$$r_k = \frac{\lambda_k}{\sum_{i=1}^{p} \lambda_i}$$

また、第 k 主成分の累積寄与率は、次のように計算される。

$$c_k = \frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{p} \lambda_i}$$

- 主成分分析にあたり、分析結果から何個目の主成分を取るのかを寄与率で判断する場合がある

主成分負荷量

観測データの各特徴量と各主成分の相関を主成分負荷量（因子負荷量）という。主成分負荷量は次のようにして計算できる。

$$Cov(x_i, z_j) = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

特徴量抽出

- 主成分分析
- カーネル主成分分析
- 線形判別分析

カーネル主成分分析

写像関数ϕを考え、特徴量を高次元空間に写像してから主成分分析を行う方法がある。

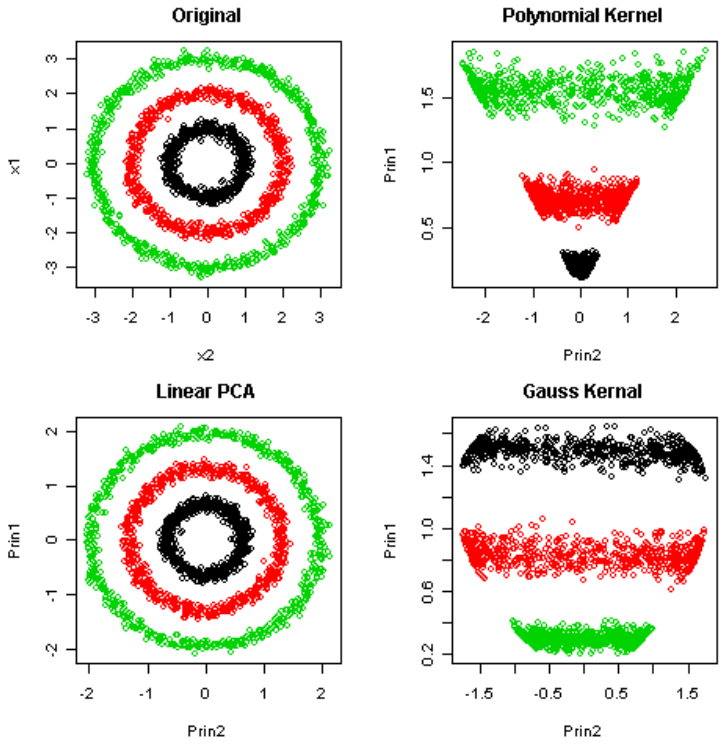
$$\phi: \mathbb{R}^d \to \mathbb{R}^k$$

写像後の特徴量空間において、分散共分散行列は次のように書ける。

$$\Sigma_{\phi(X)} = \frac{1}{n}\phi(\boldsymbol{X})^T\phi(\boldsymbol{X})$$

主成分分析と同様に固有値問題に帰着できる。

$$\Sigma_{\phi(X)}\boldsymbol{w} = \lambda\,\boldsymbol{w}$$



カーネル主成分分析

ベクトル a を次のようにおくと、

$$\boldsymbol{a} = \frac{\phi(\boldsymbol{X})\boldsymbol{w}}{\lambda^2 n}$$

固有値問題が次のように置き換えられる。

$$\frac{1}{n}\boldsymbol{K}\boldsymbol{a} = \lambda\,\boldsymbol{a} \qquad \boldsymbol{K} = \phi(\boldsymbol{X})\phi(\boldsymbol{X})^T$$

これを解くと n 個の固有値および固有ベクトルが得られるが、最初の k 個を選ぶ。

- Kはカーネル行列と呼ばれている。カーネル行列の各成分は、カーネル関数による写像後の特徴量同士の内積を表している。
- カーネル関数は多項式カーネル、RBFカーネル、双曲線正接カーネルがよく使われている。

https://axa.biopapyrus.jp/machine-learning/preprocessing/kernel-pca.html

カーネル主成分分析

新しいデータ x' を射影したい場合、$\phi(X)^T w$ を計算する必要がある。

$$\phi(X)^T w = \sum_i a^{(i)} \phi(x')^T \phi(x^{(i)})$$
$$= \sum_i a^{(i)} k(x', x^{(i)})$$

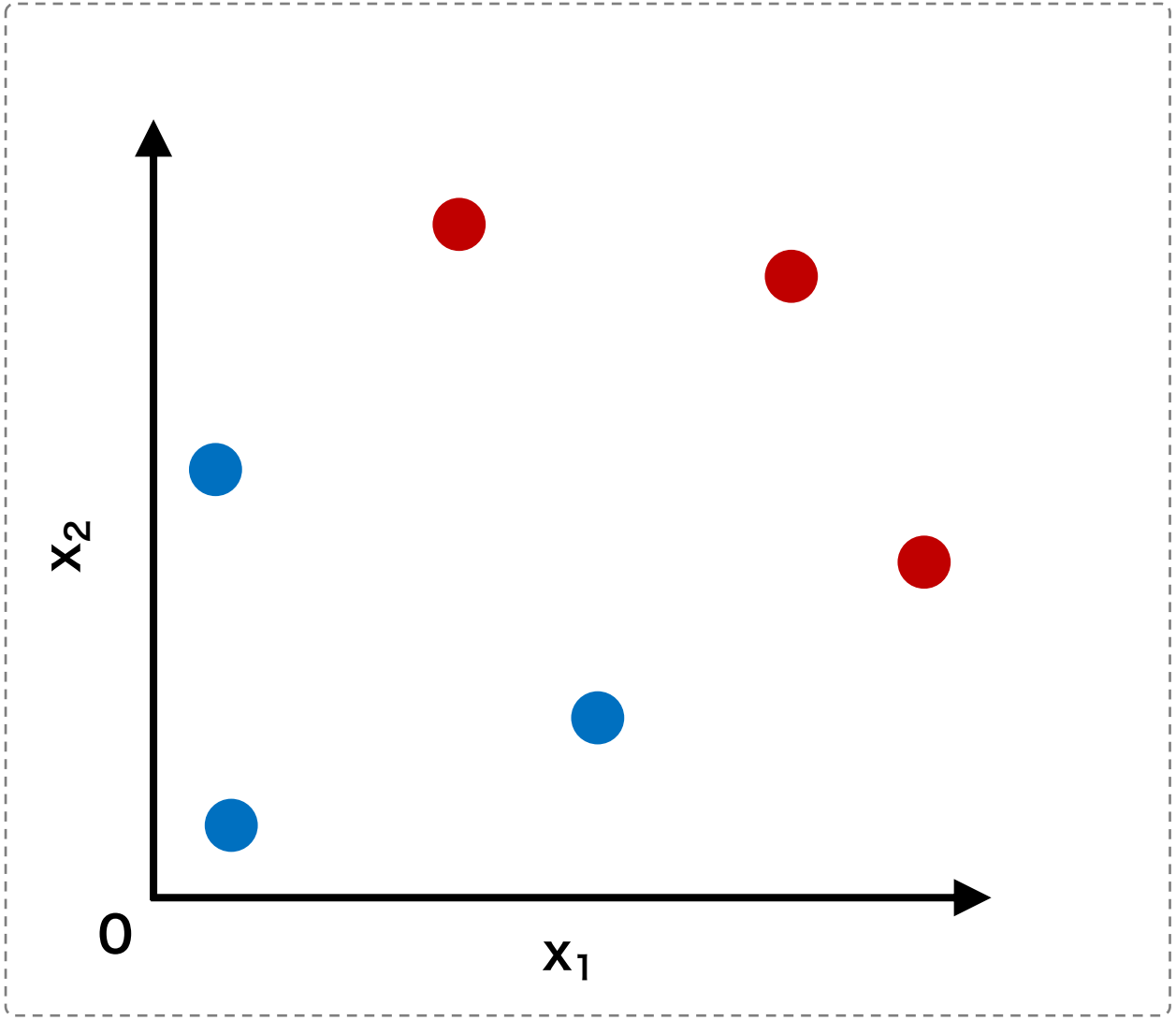
これを解くと n 個の固有値および固有ベクトルが得られるが、最初の k 個を選ぶ。

- 新しいデータ x' を既存の特徴量空間に射影するとき、x' と訓練データ x との類似度を計算する必要がある。
- カーネル主成分分析を使用するとき、訓練データをすべて記憶する必要がある。

特徴量抽出

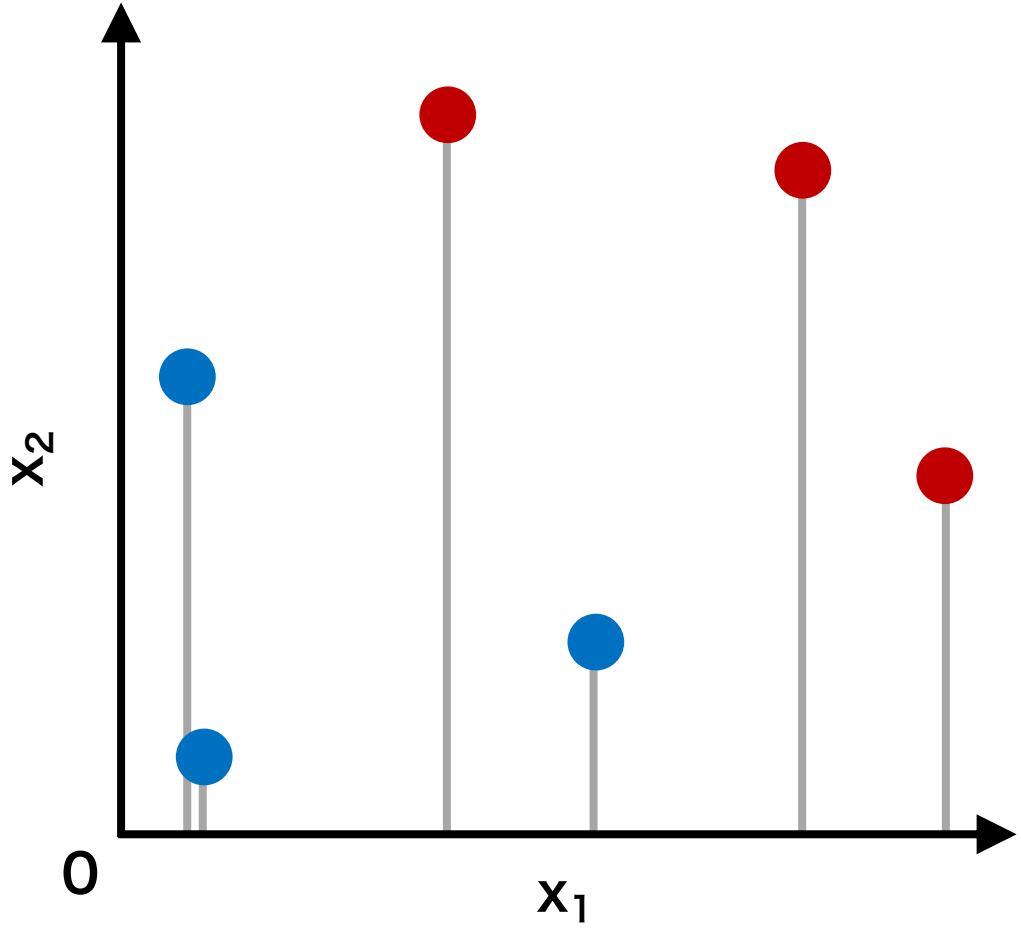
- 主成分分析
- カーネル主成分分析
- 線形判別分析

線形判別分析



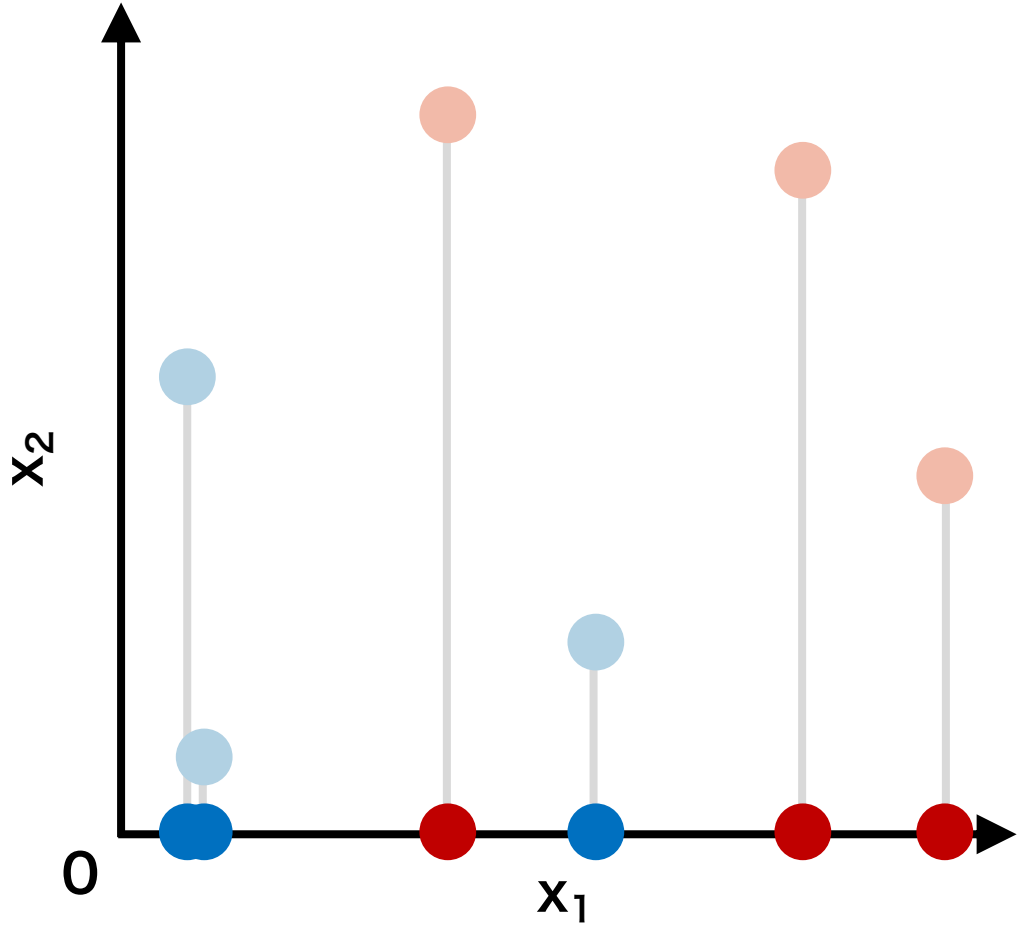
線形判別分析は特徴抽出の目的で使われる。主成分分析と異なり、データの教師ラベルを使用して、異なるクラスを最もよく分離できる空間にデータを写像する。

線形判別分析



線形判別分析では、射影後の座標において、２ つのクラスを最もよく分離できるような軸を見つける。

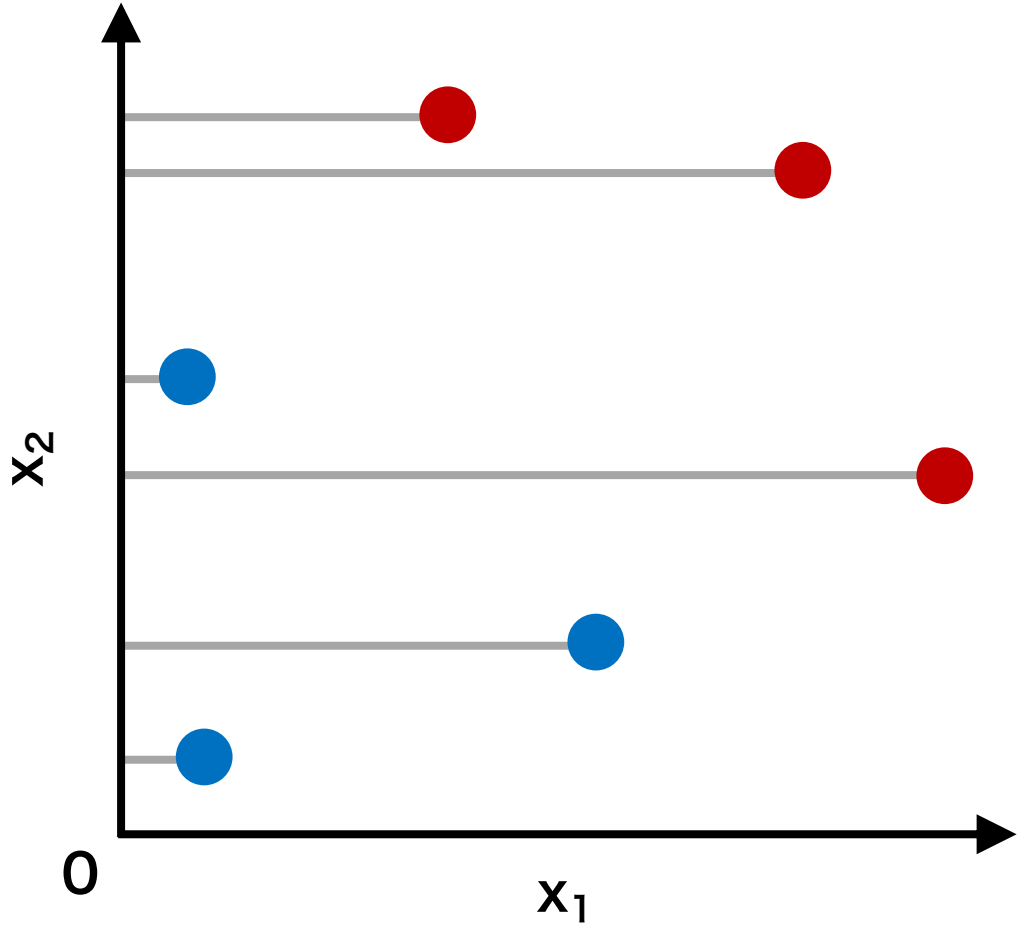
線形判別分析



線形判別分析では、射影後の座標において、２ つのクラスを最もよく分離できるような軸を見つける。

- x_1 軸では、青点と赤点を分離することができない。

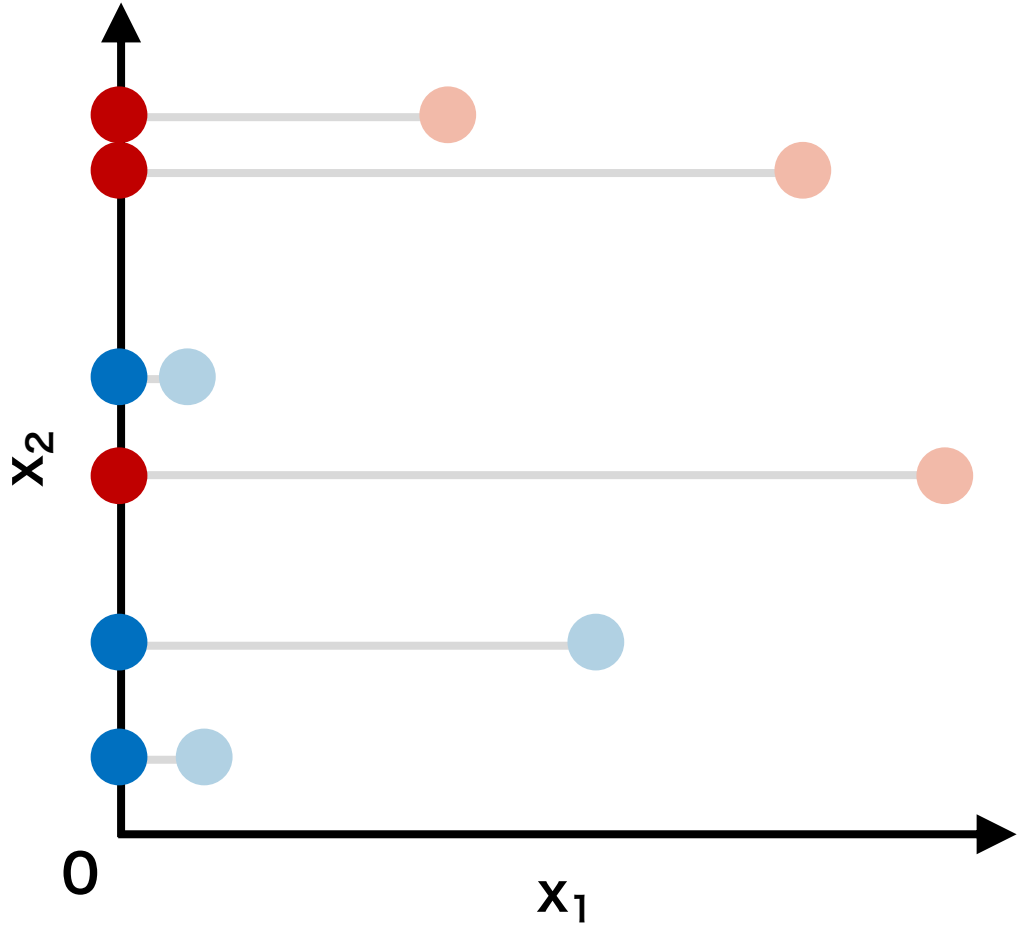
線形判別分析



線形判別分析では、射影後の座標において、２ つのクラスを最もよく分離できるような軸を見つける。

- x_1 軸では、青点と赤点を分離することができない。

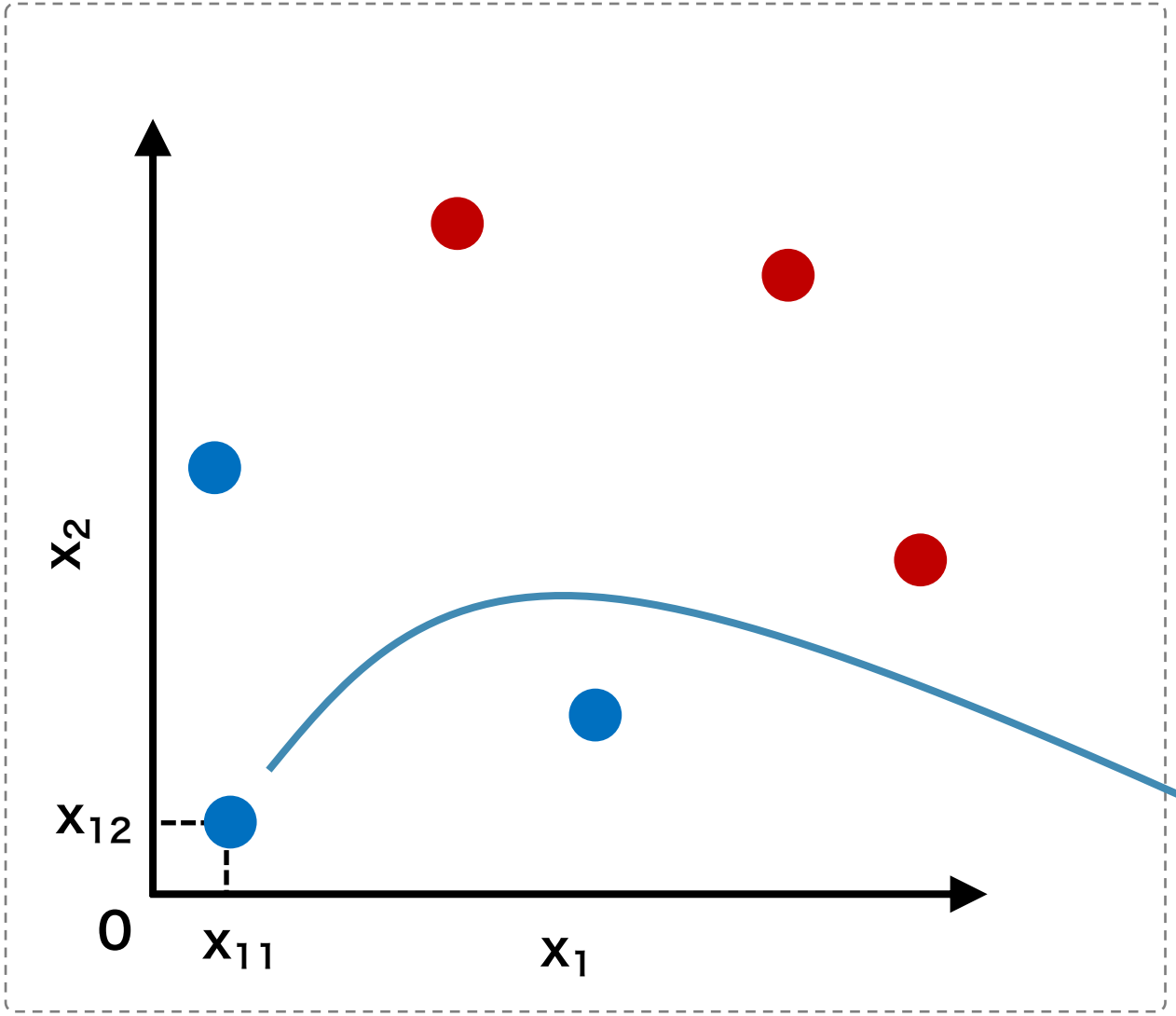
線形判別分析



線形判別分析では、射影後の座標において、２ つのクラスを最もよく分離できるような軸を見つける。

- x_1 軸では、青点と赤点を分離することができない。
- x_2 軸では、青点と赤点を分離することができない。

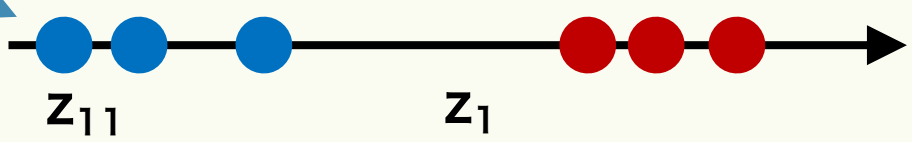
線形判別分析



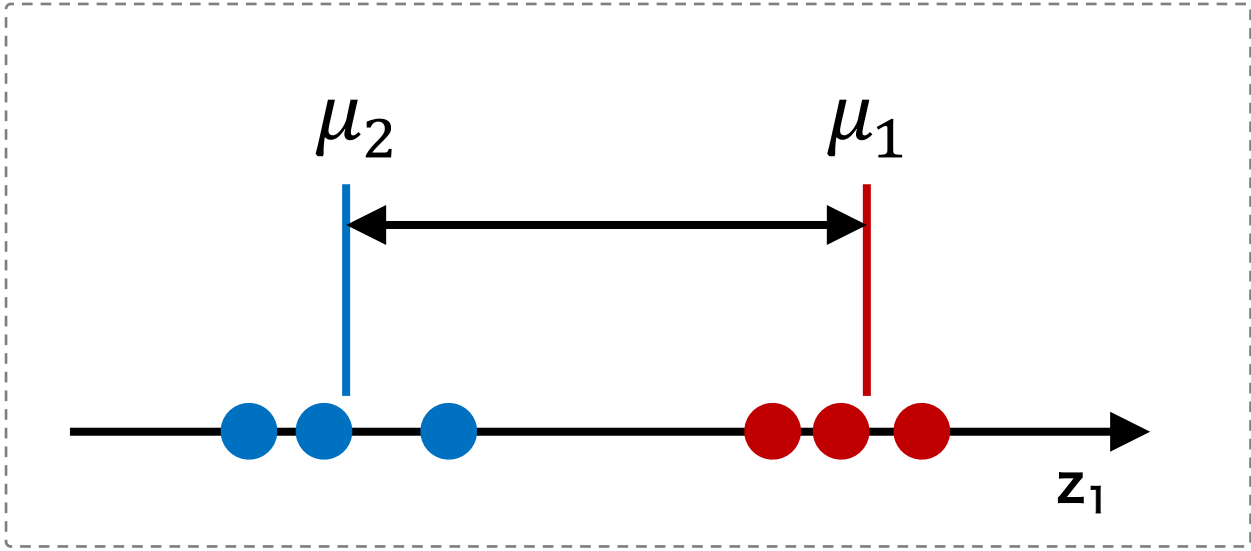
線形判別分析では、射影後の座標において、2 つのクラスを最もよく分離できるような軸を見つける。

- x_1 軸では、青点と赤点を分離することができない。
- x_2 軸では、青点と赤点を分離することができない。
- 青点と赤点をもっともよく分離できる線形写像 w を考えて、w で写像する。

$$z_{11} = w_0 + w_1 x_{11} + w_2 x_{12}$$



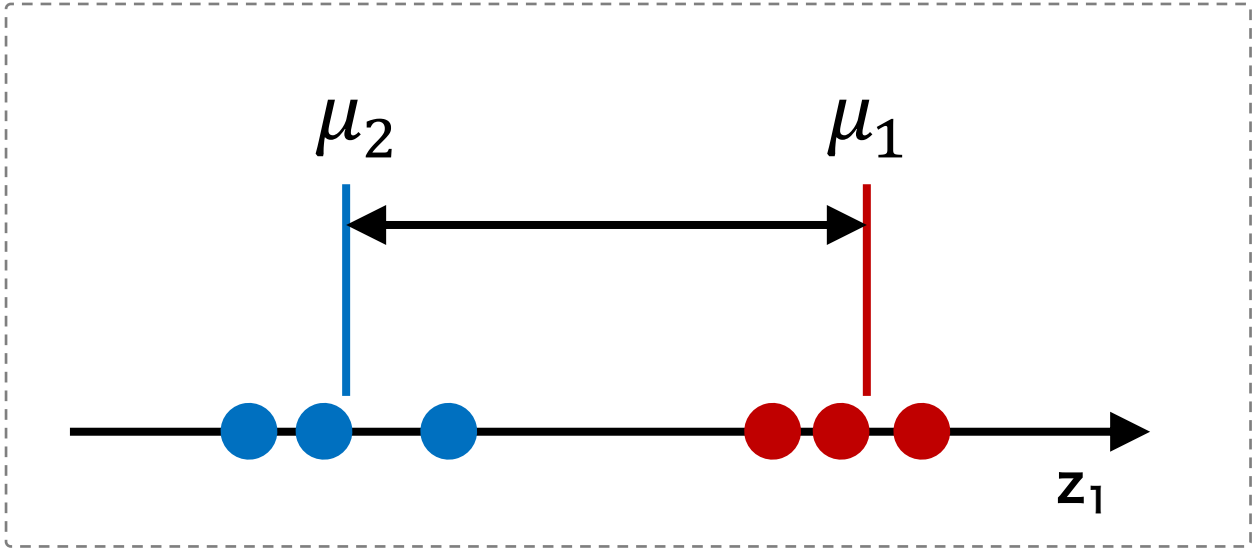
線形判別分析

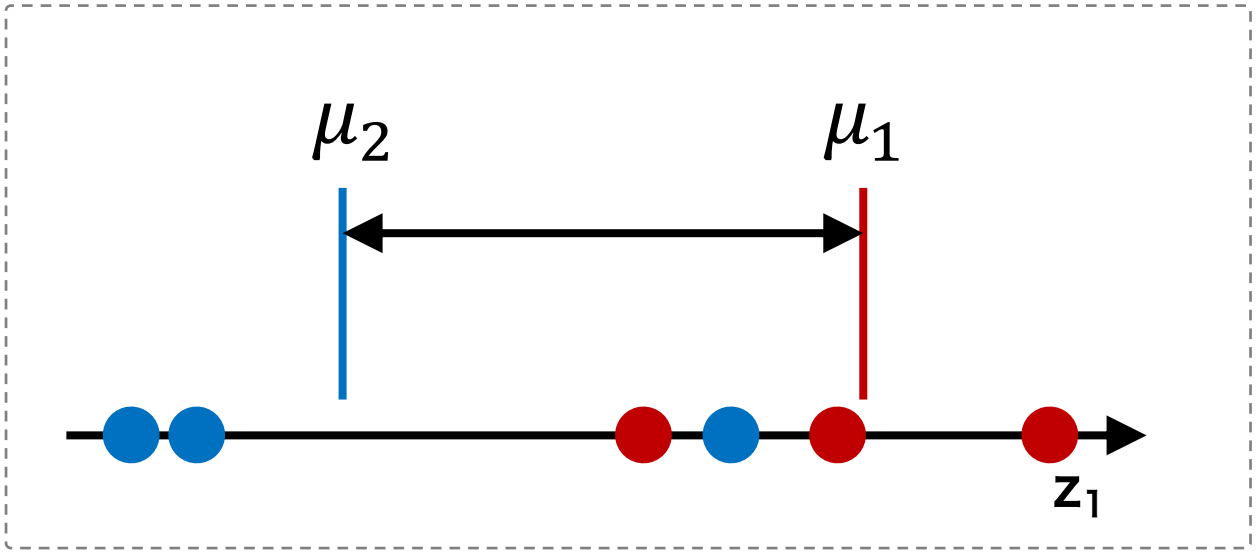


線形判別分析では、射影後の座標において、２ つのクラスを最もよく分離できるような軸を見つける。例えば、ある軸における、２ クラスのデータの平均が最も離れていれば、それは ２ つのクラスをもっともよく分離できる軸とみなすことができる。

$$J(\boldsymbol{w}) = |\mu_1 - \mu_2|^2$$

線形判別分析



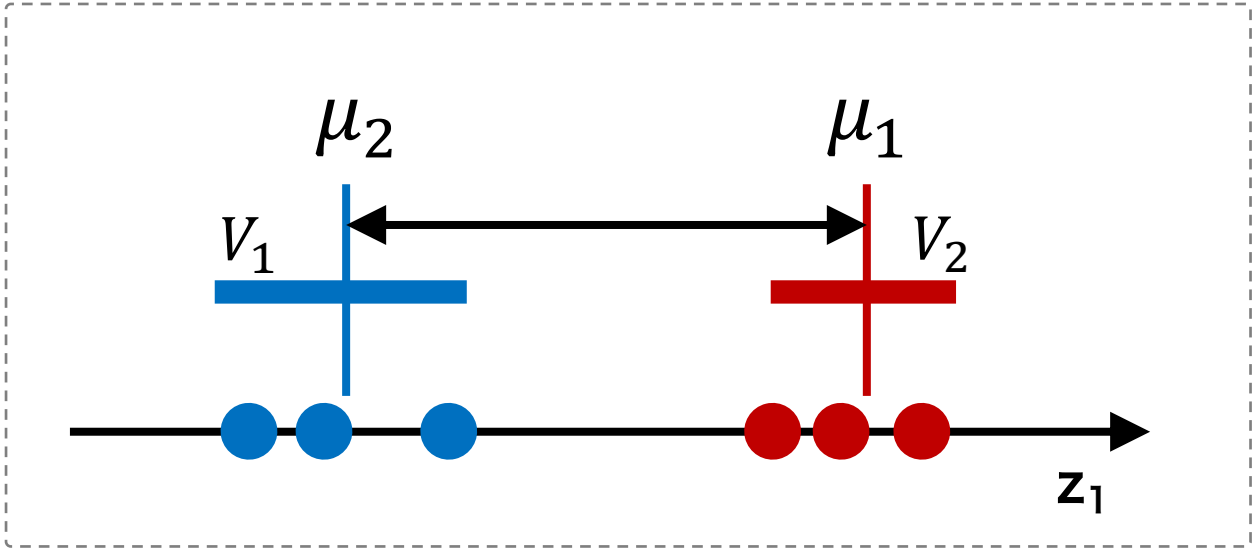


線形判別分析では、射影後の座標において、２ つのクラスを最もよく分離できるような軸を見つける。例えば、ある軸における、２ クラスのデータの平均が最も離れていれば、それは ２ つのクラスをもっともよく分離できる軸とみなすことができる。

$$J(\boldsymbol{w}) = |\mu_1 - \mu_2|^2$$

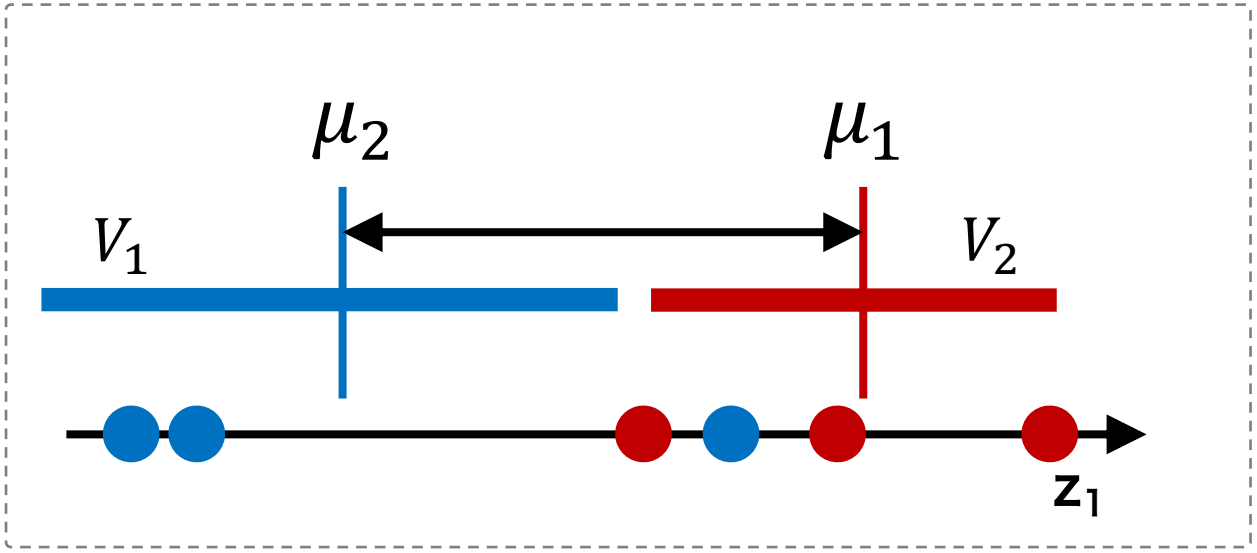
ただし、クラス内分散が大きいときは、各クラス平均同士の距離が大きくても、分離できない場合がある。

線形判別分析



線形判別分析では、射影後の座標において、２ つのクラスを最もよく分離できるような軸を見つける。例えば、ある軸における、２ クラスのデータの平均が最も離れていれば、それは ２ つのクラスをもっともよく分離できる軸とみなすことができる。

$$J(\boldsymbol{w}) = |\mu_1 - \mu_2|^2$$



ただし、クラス内分散が大きいときは、各クラス平均同士の距離が大きくても、分離できない場合がある。そのため、分散の効果も取り入れる必要がある。

$$J(\boldsymbol{w}) = \frac{|\mu_1 - \mu_2|^2}{V_1 + V_2}$$

線形判別分析

$$J(\boldsymbol{w}) = \frac{|\mu_1 - \mu_2|^2}{V_1 + V_2}$$

最大化

最小化

- クラス 1 とクラス 2 を分離するために、次のような最適化を行う必要がある。
  - 写像後の座標軸において、クラス 1 平均とクラス 2 平均の距離をできるだけ長くする。
  - クラス 1 の分散をできるだけ小さくする。
  - クラス 2 の分散をできるだけ小さくする。

線形判別分析

$$J(\boldsymbol{w}) = \frac{|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2|^2}{\boldsymbol{V}_1 + \boldsymbol{V}_2}$$

最大化

最小化

- クラス 1 とクラス 2 を分離するために、次のような最適化を行う必要がある。
  - 写像後の座標軸において、クラス 1 平均とクラス 2 平均の距離をできるだけ長くする。
  - クラス 1 の分散をできるだけ小さくする。
  - クラス 2 の分散をできるだけ小さくする。
- 空間上でもこの等式は成り立つ。

平均間の距離

$$|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2|^2 = |\bar{\boldsymbol{z}}_1 - \bar{\boldsymbol{z}}_2|^2$$

$$= |\boldsymbol{w}^t \bar{\boldsymbol{x}}_1 - \boldsymbol{w}^t \bar{\boldsymbol{x}}_2|^2$$

$$= |\boldsymbol{w}^t (\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)|^2$$

$$= \boldsymbol{w}^t \boxed{(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)(\bar{\boldsymbol{x}}_1 - \bar{\boldsymbol{x}}_2)^t} \boldsymbol{w}$$

クラス間共分散行列

- クラス 1 とクラス 2 を分離するために、次のような最適化を行う必要がある。
  - 写像後の座標軸において、クラス 1 平均とクラス 2 平均の距離をできるだけ長くする。
  - クラス 1 の分散をできるだけ小さくする。
  - クラス 2 の分散をできるだけ小さくする。
- 空間上でもこの等式は成り立つ。

クラス内分散

クラス j の分散は、分散共分散行列として表される。

$$\Sigma_{z_j} = \frac{1}{n} z_j'^t z_j' = \frac{1}{n} \left(X_j' w\right)^t \left(X_j' w\right)$$
$$= w^t \Sigma_{X_j} w$$

よって、クラス 1 の分散とクラス 2 の分散は次のように計算される。

$$V_1 + V_2 = w^t \Sigma_{X_1} w + w^t \Sigma_{X_2} w$$
$$= w^t \left(\Sigma_{X_1} + \Sigma_{X_2}\right) w$$

クラス内共分散行列和

- クラス 1 とクラス 2 を分離するために、次のような最適化を行う必要がある。
  - 写像後の座標軸において、クラス 1 平均とクラス 2 平均の距離をできるだけ長くする。
  - クラス 1 の分散をできるだけ小さくする。
  - クラス 2 の分散をできるだけ小さくする。
- 空間上でもこの等式は成り立つ。

分散共分散行列

参考

$$\Sigma_X = \begin{bmatrix} Cov(\boldsymbol{x}_1, \boldsymbol{x}_1) & \cdots & Cov(\boldsymbol{x}_1, \boldsymbol{x}_n) \\ \vdots & \ddots & \vdots \\ Cov(\boldsymbol{x}_n, \boldsymbol{x}_1) & \cdots & Cov(\boldsymbol{x}_n, \boldsymbol{x}_n) \end{bmatrix} = \frac{1}{n} \boldsymbol{X'}^t \boldsymbol{X'}$$

- 各特徴間の分散を表している。$Cov(x_k, x_l)$ は特徴 k と特徴 l の共分散を表す。対角線上の $Cov(x_k, x_k)$ は特徴 k の分散を表す。

ただし、 $Cov(\boldsymbol{x}_k, \boldsymbol{x}_l) = E[(\boldsymbol{x}_k - E[\boldsymbol{x}_k])(\boldsymbol{x}_l - E[\boldsymbol{x}_l])]$

$$= \frac{1}{n} \sum_{j=1}^{n} (x_{jk} - \bar{x}_k)(x_{jl} - \bar{x}_l)$$

$$= \frac{1}{n} \boldsymbol{x'}_k^t \boldsymbol{x'}_l$$

$$\boldsymbol{x'}_k = (x_{1k} - \bar{x}_k \quad \cdots \quad x_{nk} - \bar{x}_k)^t$$

線形判別分析

$$J(\boldsymbol{w}) = \frac{|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2|^2}{\boldsymbol{V}_1 + \boldsymbol{V}_2}$$

$$= \frac{\boldsymbol{w}^t(\overline{\boldsymbol{x}}_1 - \overline{\boldsymbol{x}}_2)(\overline{\boldsymbol{x}}_1 - \overline{\boldsymbol{x}}_2)^t\boldsymbol{w}}{\boldsymbol{w}^t(\Sigma_{\boldsymbol{X}_1} + \Sigma_{\boldsymbol{X}_2})\boldsymbol{w}}$$

$$= \frac{\boldsymbol{w}^t\boldsymbol{S}_B\boldsymbol{w}}{\boldsymbol{w}^t\boldsymbol{S}_W\boldsymbol{w}}$$

J(w) の最大化は次の式に置き換えることができる

$$\underset{\boldsymbol{w}}{\operatorname{argmax}}\, \boldsymbol{w}^t\boldsymbol{S}_B\boldsymbol{w} \quad \text{ただし} \quad \mathbf{w}^{\mathbf{t}}\mathbf{S}_W\mathbf{w} = \mathbf{1}$$

- 線形判別分析は左の式に帰着できる。
- J(w) を最大化にする w が w' であるとき、ある定数 k が存在し、kw' も J(w) を最大にする。そこで、w に対して制約条件を設ける。

線形判別分析

ラグランジュの未定乗数法を利用して最大化問題を解く。

$$L(\boldsymbol{w}, \lambda) = \boldsymbol{w}^t \boldsymbol{S}_B \boldsymbol{w} - \lambda(\boldsymbol{w}^t \boldsymbol{S}_w \boldsymbol{w} - 1)$$

$\boldsymbol{\lambda}$ で偏微分して 0 とおくと、

$$\boldsymbol{w}^t \boldsymbol{S}_w \boldsymbol{w} - 1 = \boldsymbol{0}$$

$$\boldsymbol{w}^t \boldsymbol{S}_w \boldsymbol{w} = 1$$

- 線形判別分析は左の式に帰着できる。
- J(w) を最大化にする w が w' であるとき、ある定数 k が存在し、kw' も J(w) を最大にする。そこで、w に対して制約条件を設ける。

線形判別分析

w で偏微分して 0 とおくと、

$$\boldsymbol{S}_B\boldsymbol{w} - \lambda\boldsymbol{S}_w\boldsymbol{w} = \boldsymbol{0}$$

$$\boldsymbol{S}_B\boldsymbol{w} = \lambda\boldsymbol{S}_w\boldsymbol{w}$$

$\boldsymbol{w}^t\boldsymbol{S}_w\boldsymbol{w} = 1$ だから、左辺から S_w^{-1} をかけると、

$$\boldsymbol{S}_w^{-1}\boldsymbol{S}_B\boldsymbol{w} = \boldsymbol{S}_w^{-1}\lambda\boldsymbol{S}_w\boldsymbol{w} = \lambda\boldsymbol{w}$$

これにより線形判別分析は固有値問題に帰着することができた。

- 線形判別分析は左の式に帰着できる。
- J(w) を最大化にする w が w' であるとき、ある定数 k が存在し、kw' も J(w) を最大にする。そこで、w に対して制約条件を設ける。

モデル評価

- パラメーターチューニング
- 評価データセット
- バイアスとバリアンス
- 評価指標

モデル評価

- パラメーターチューニング
- 評価データセット
- バイアスとバリアンス
- 評価指標

パラメータ

機械学習には 2 種類のパラメーターがある。一つは、訓練データから学習されるパラメータ（重み）である。もう一つは、アルゴリズムに依存するパラメータ（ハイパーパラメータ）で、学習を行う前に人があらかじめ与えておく必要がある。例えば、回帰問題では、入力に使用する変数の数がハイパーパラメータであり、各変数にかかる係数がパラメータである。

モデル 1

$$開花率 = b_0 + b_1気温 + b_2降水量 + b_3湿度 + b_4日射量$$

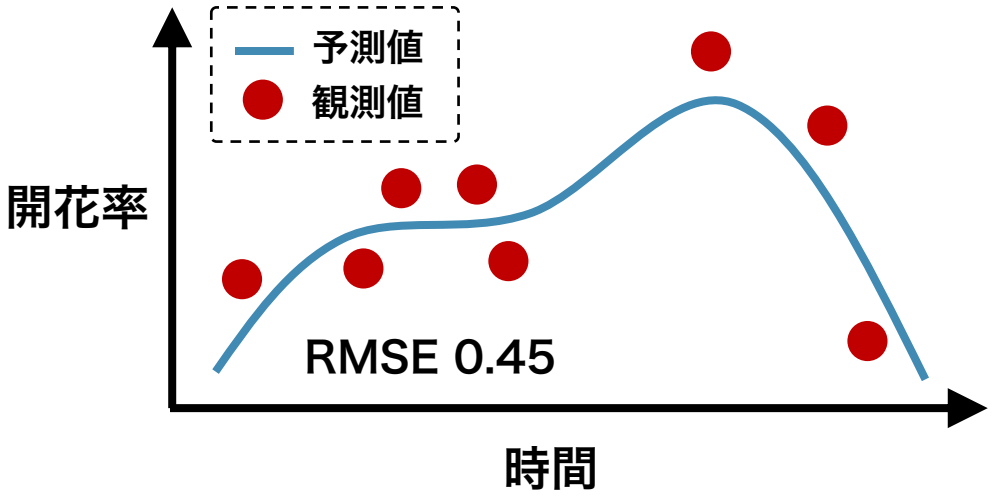
ハイパーパラメータ	気温，降水量，湿度，日射量
パラメータ	b_0, b_1, b_2, b_3, b_4

モデル 2

$$開花率 = b_0 + b_1降水量 + b_2日射量$$

ハイパーパラメータ	気温，降水量，日射量
パラメータ	b_0, b_1, b_2

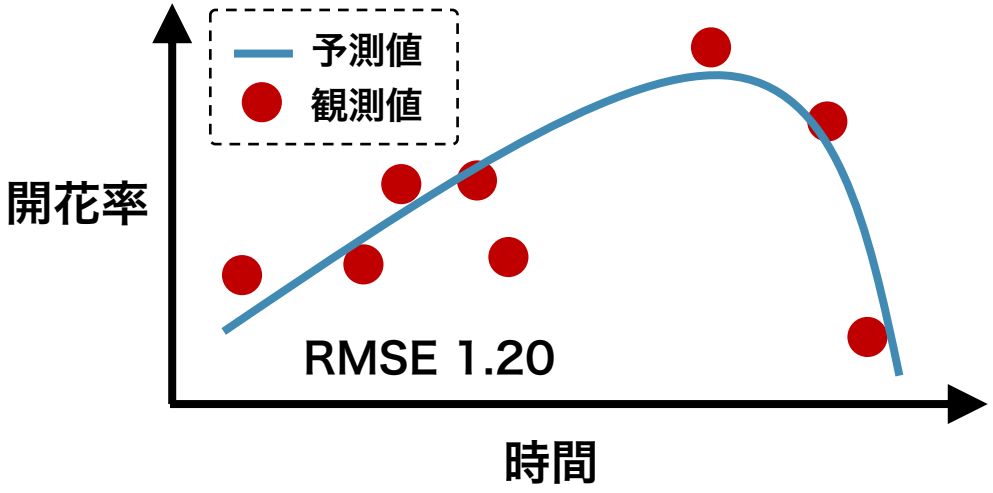
ハイパーパラメータ（離散値）



モデル 1

$$開花率 = b_0 + b_1気温 + b_2降水量 + b_3湿度 + b_4日射量$$

ハイパーパラメータ	気温，降水量，湿度，日射量
パラメータ	b_0, b_1, b_2, b_3, b_4



モデル 2

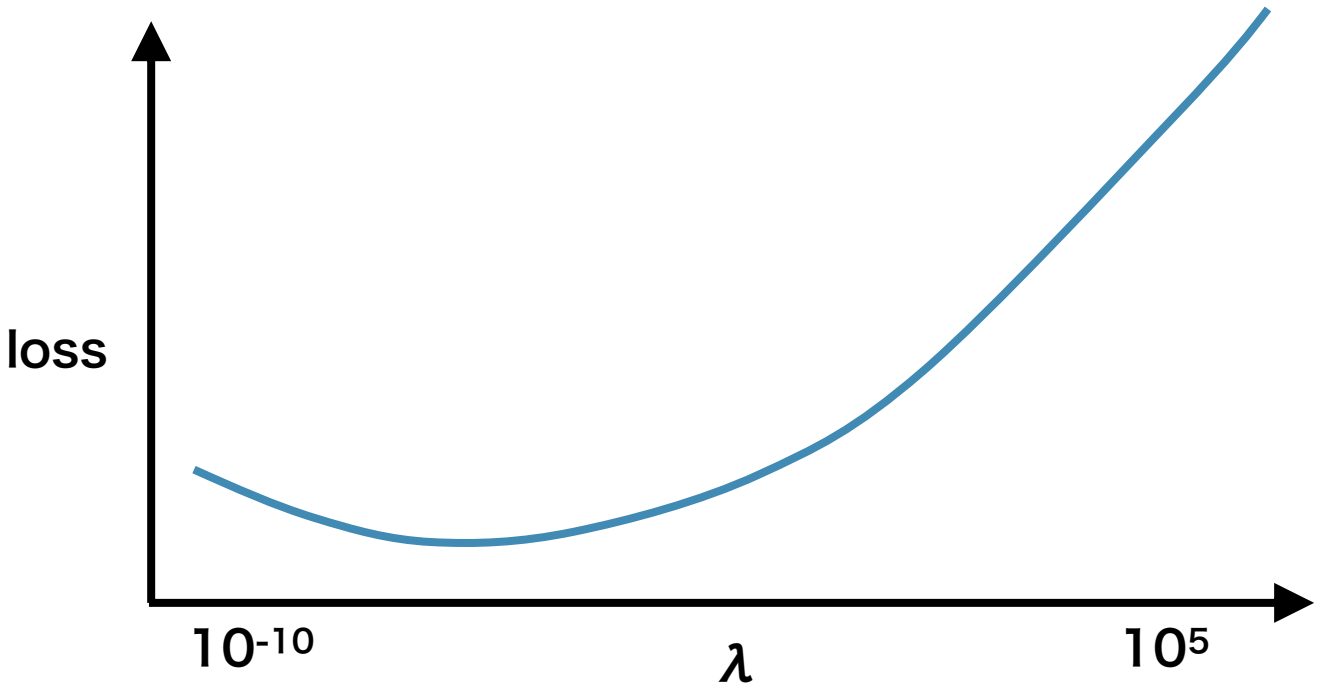
$$開花率 = b_0 + b_1降水量 + b_2日射量$$

ハイパーパラメータ	気温，降水量，日射量
パラメータ	b_0, b_1, b_2

ハイパーパラメータ（連続値）

$$loss(w) = -\boldsymbol{w}^T\boldsymbol{x} + \lambda\frac{\|\boldsymbol{w}\|^2}{2}$$

λ を連続的に変化させて、loss 関数に最小値を与える λ を探索する。



グリッドサーチ

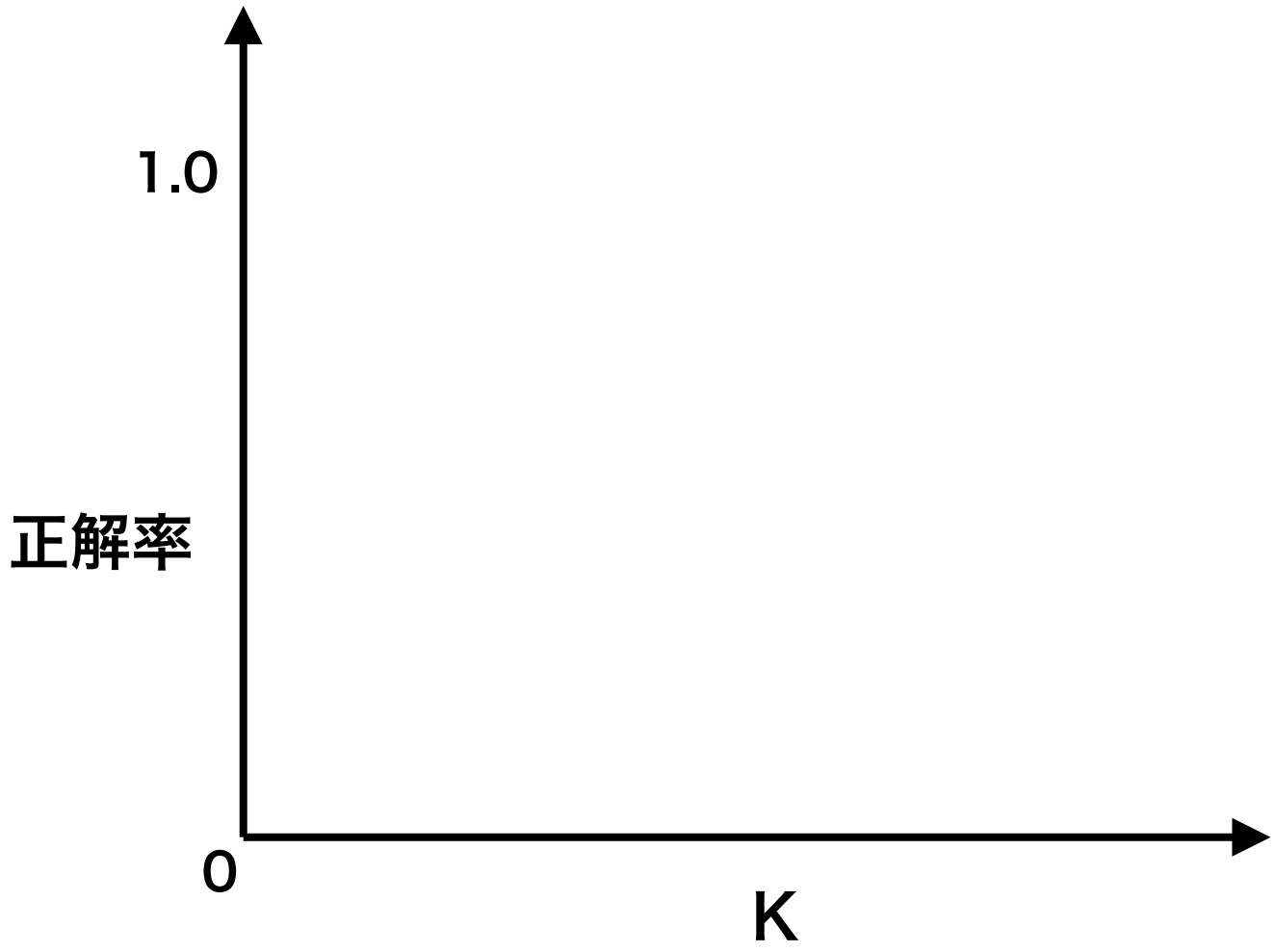
すべてのデータセットに共通する最適な K は存在しない。そのため、いくつかの K を試して、そのデータセットに最適な K を選ぶ必要がある。最適な K を選ぶ方法の 1 つとして、グリッドサーチがある。

1. K の候補となる値を複数個挙げる。

2. すべての K に対して、学習と検証を行い、その正解率を記録する。

3. 最も大きな正解率を出した K の近辺の値を候補として、再度、学習と検証を行う。

4. 最適な K の候補域を徐々に縮小させながら最適な K を探索する。



グリッドサーチ

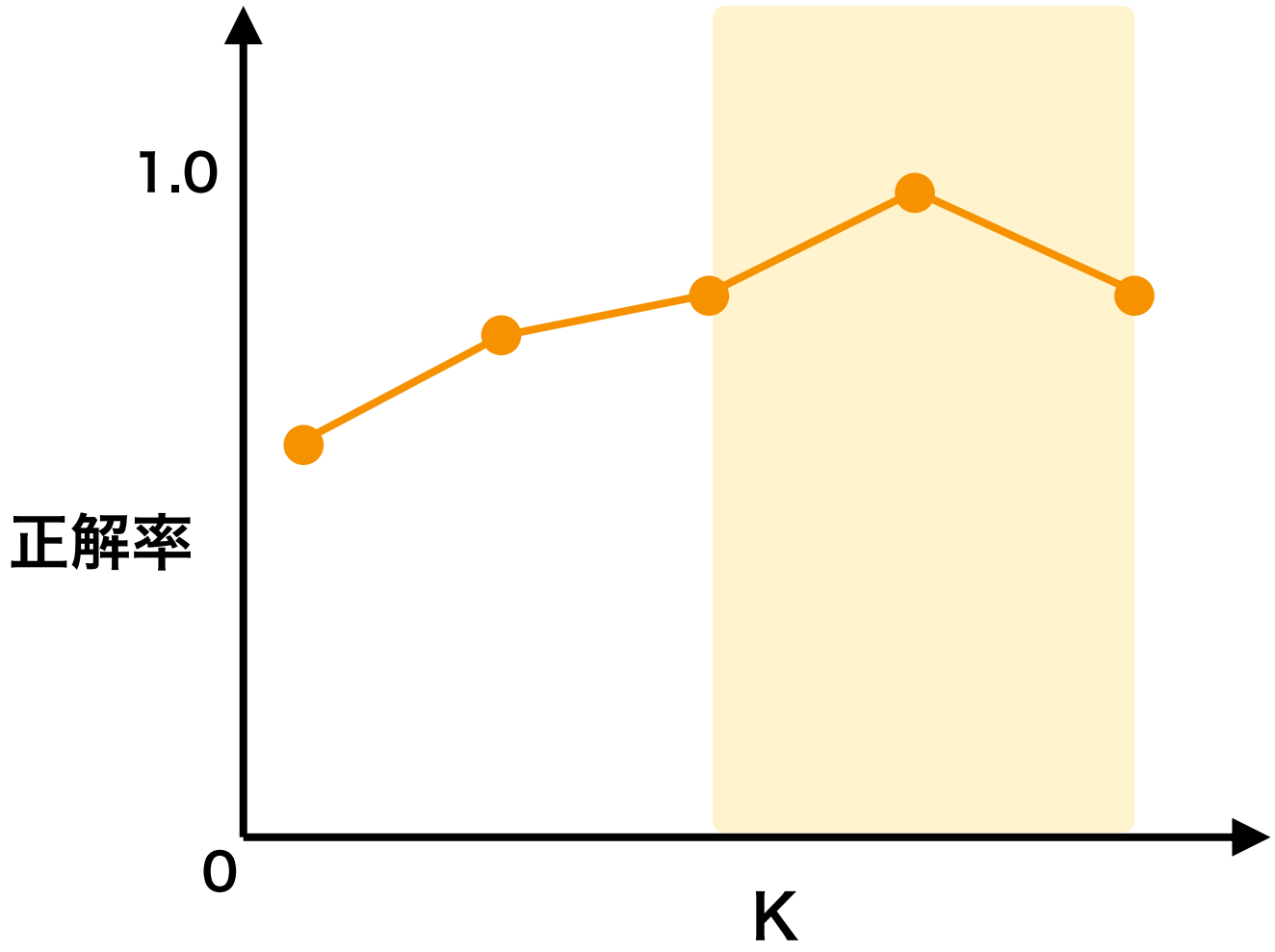
すべてのデータセットに共通する最適な K は存在しない。そのため、いくつかの K を試して、そのデータセットに最適な K を選ぶ必要がある。最適な K を選ぶ方法の 1 つとして、グリッドサーチがある。

1. K の候補となる値を複数個挙げる。

2. すべての K に対して、学習と検証を行い、その正解率を記録する。

3. 最も大きな正解率を出した K の近辺の値を候補として、再度、学習と検証を行う。

4. 最適な K の候補域を徐々に縮小させながら最適な K を探索する。



グリッドサーチ

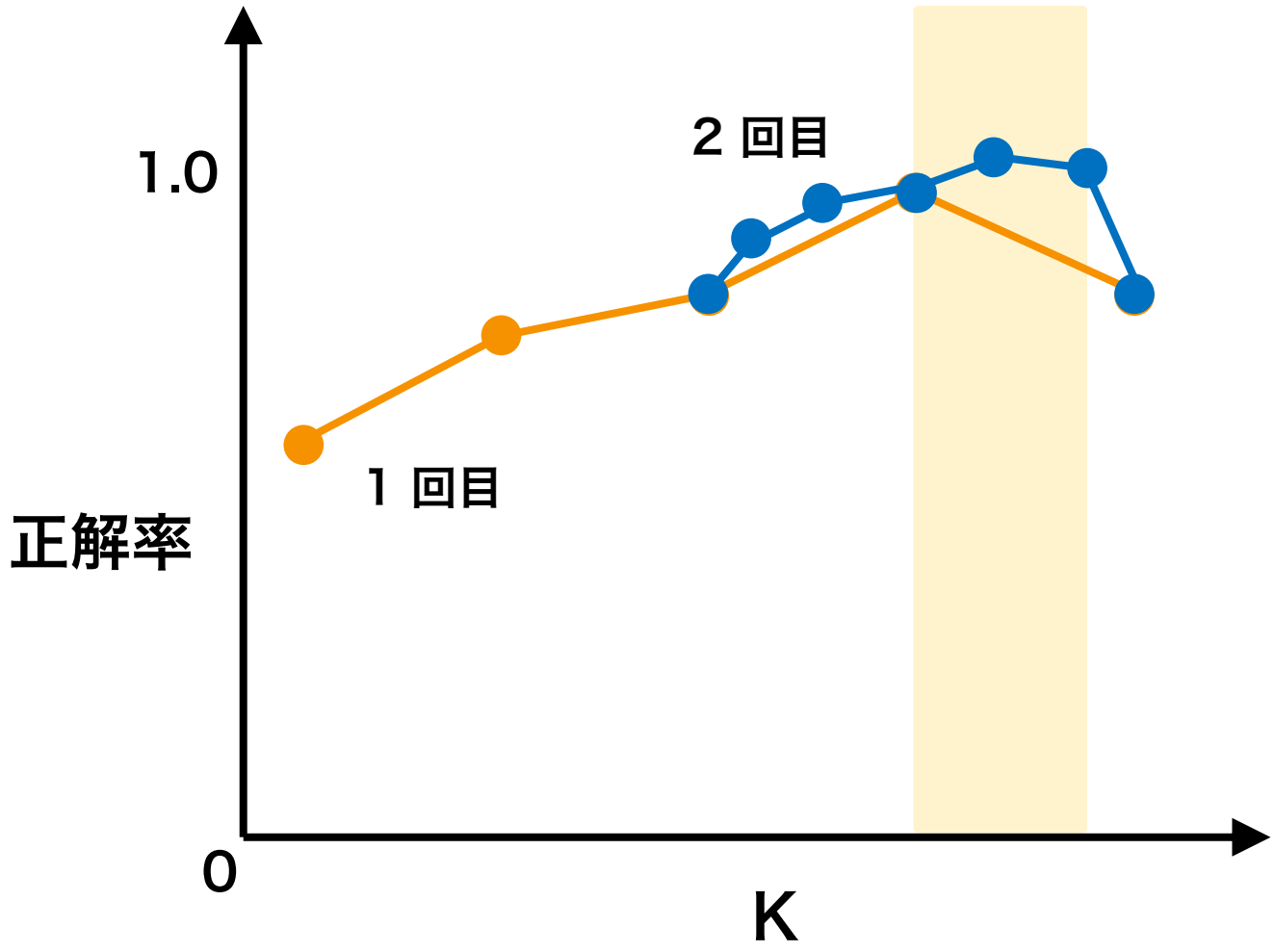
すべてのデータセットに共通する最適な K は存在しない。そのため、いくつかの K を試して、そのデータセットに最適な K を選ぶ必要がある。最適な K を選ぶ方法の 1 つとして、グリッドサーチがある。

1. K の候補となる値を複数個挙げる。

2. すべての K に対して、学習と検証を行い、その正解率を記録する。

3. 最も大きな正解率を出した K の近辺の値を候補として、再度、学習と検証を行う。

4. 最適な K の候補域を徐々に縮小させながら最適な K を探索する。



グリッドサーチ

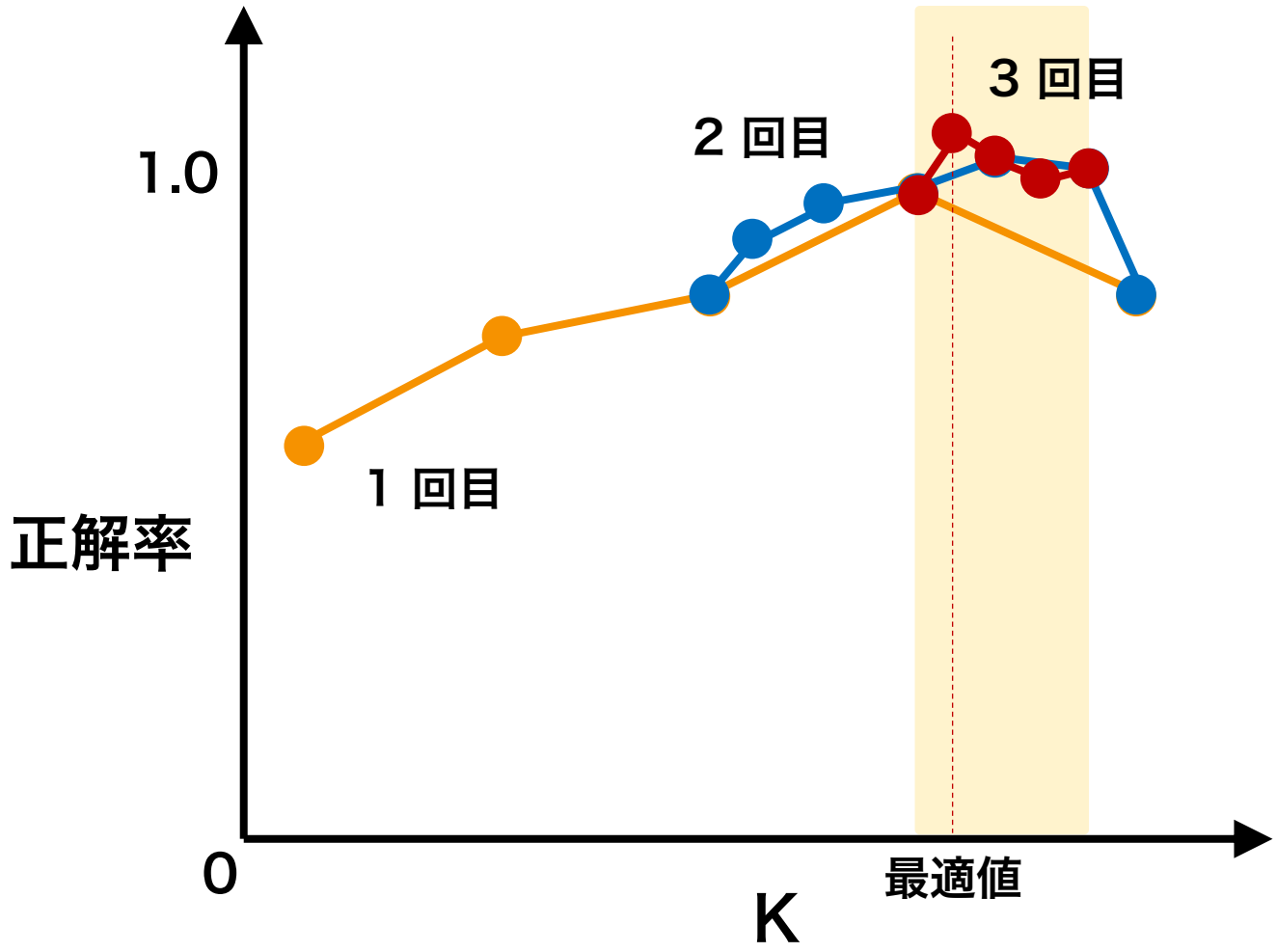
すべてのデータセットに共通する最適な K は存在しない。そのため、いくつかの K を試して、そのデータセットに最適な K を選ぶ必要がある。最適な K を選ぶ方法の 1 つとして、グリッドサーチがある。

1. K の候補となる値を複数個挙げる。

2. すべての K に対して、学習と検証を行い、その正解率を記録する。

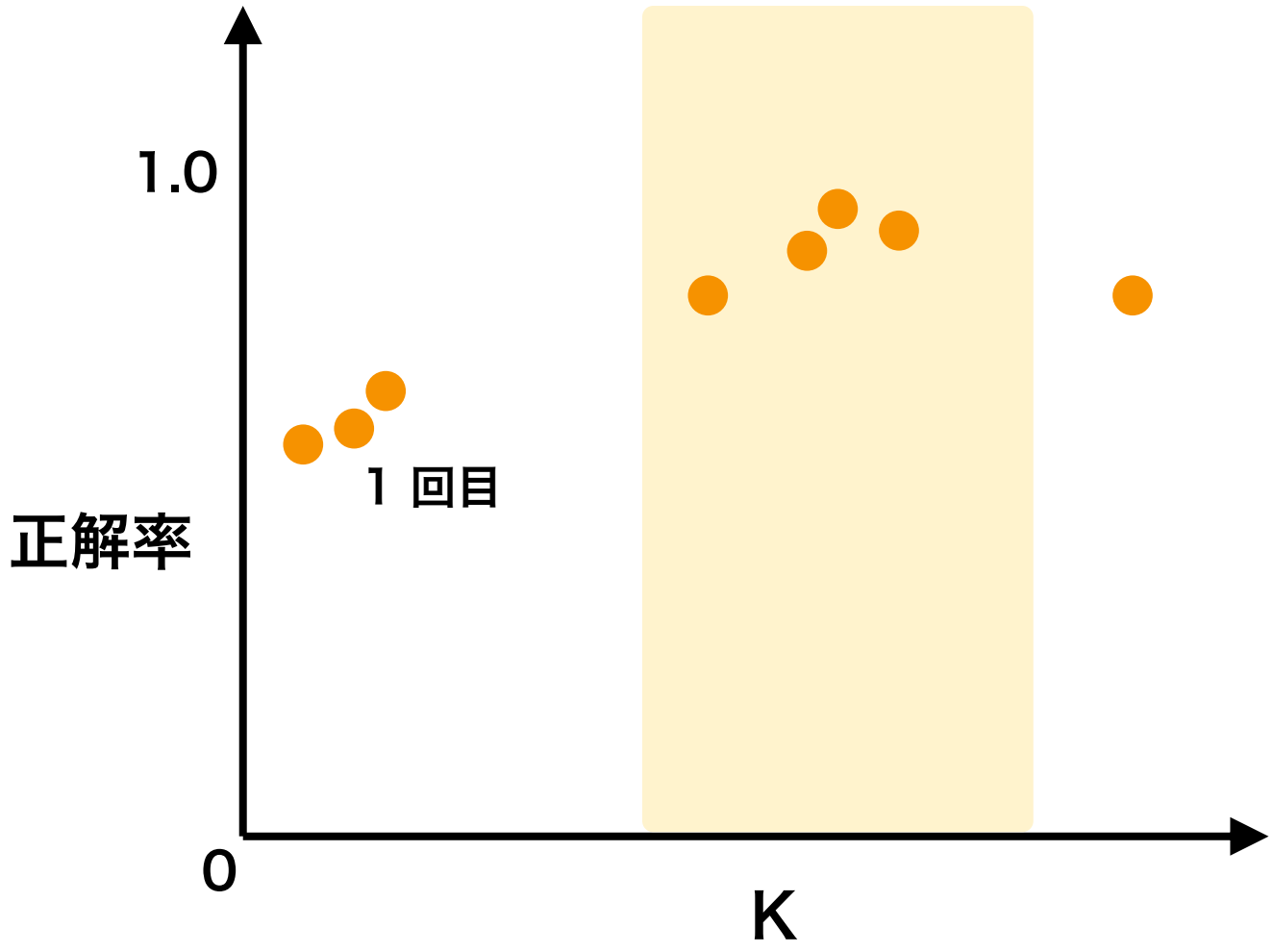
3. 最も大きな正解率を出した K の近辺の値を候補として、再度、学習と検証を行う。

4. 最適な K の候補域を徐々に縮小させながら最適な K を探索する。



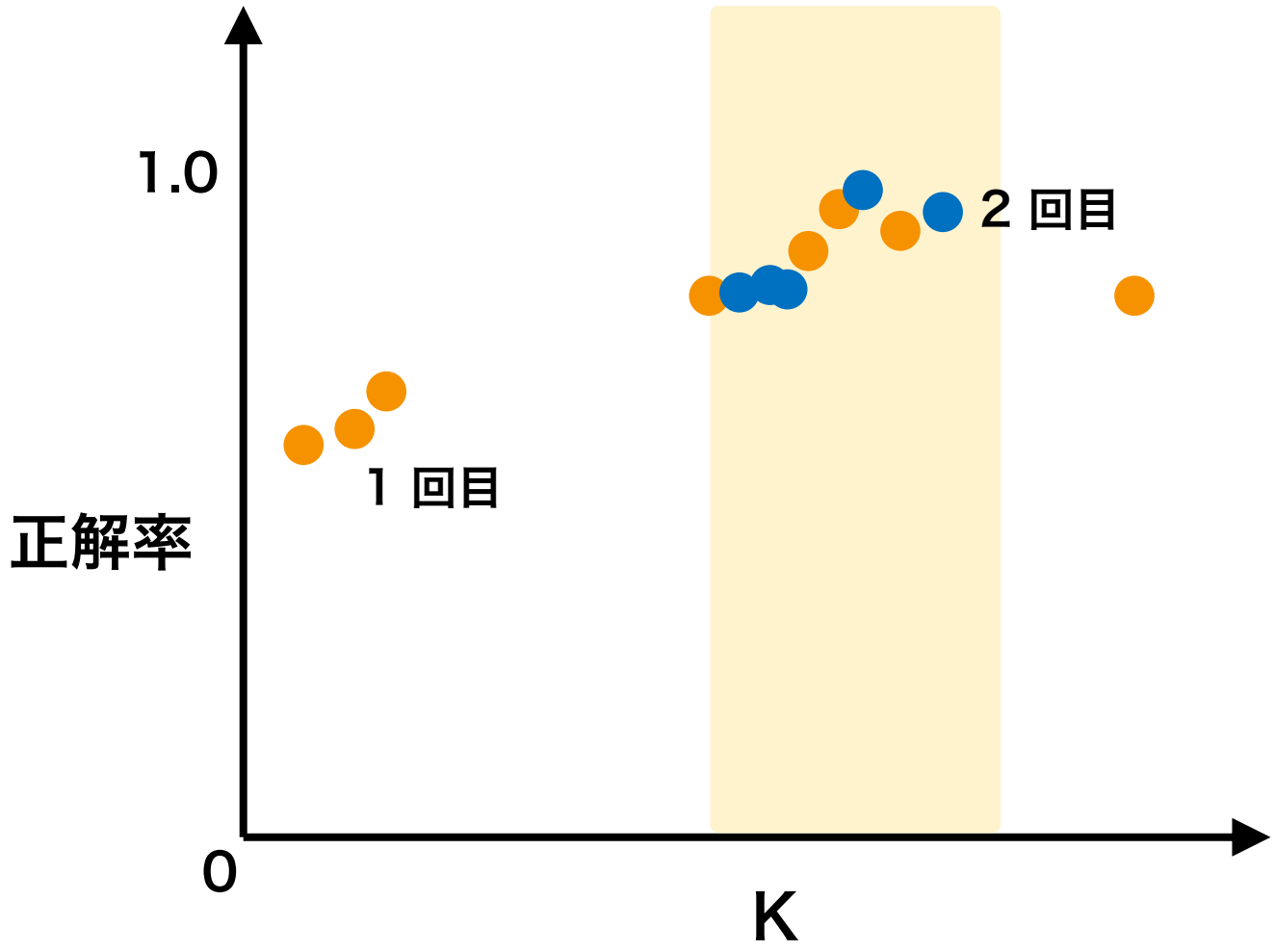
ランダムサーチ

ランダムサーチは、指定された範囲からランダムに K の候補を生成して検証を行う方法である。ランダムサーチも、グリッドサーチと同様に、複数回繰り返して、最適な K を決めることもできる。ただし、K はランダムに生成されるため、最適解あるいはその近傍にたどり着けない場合がある。



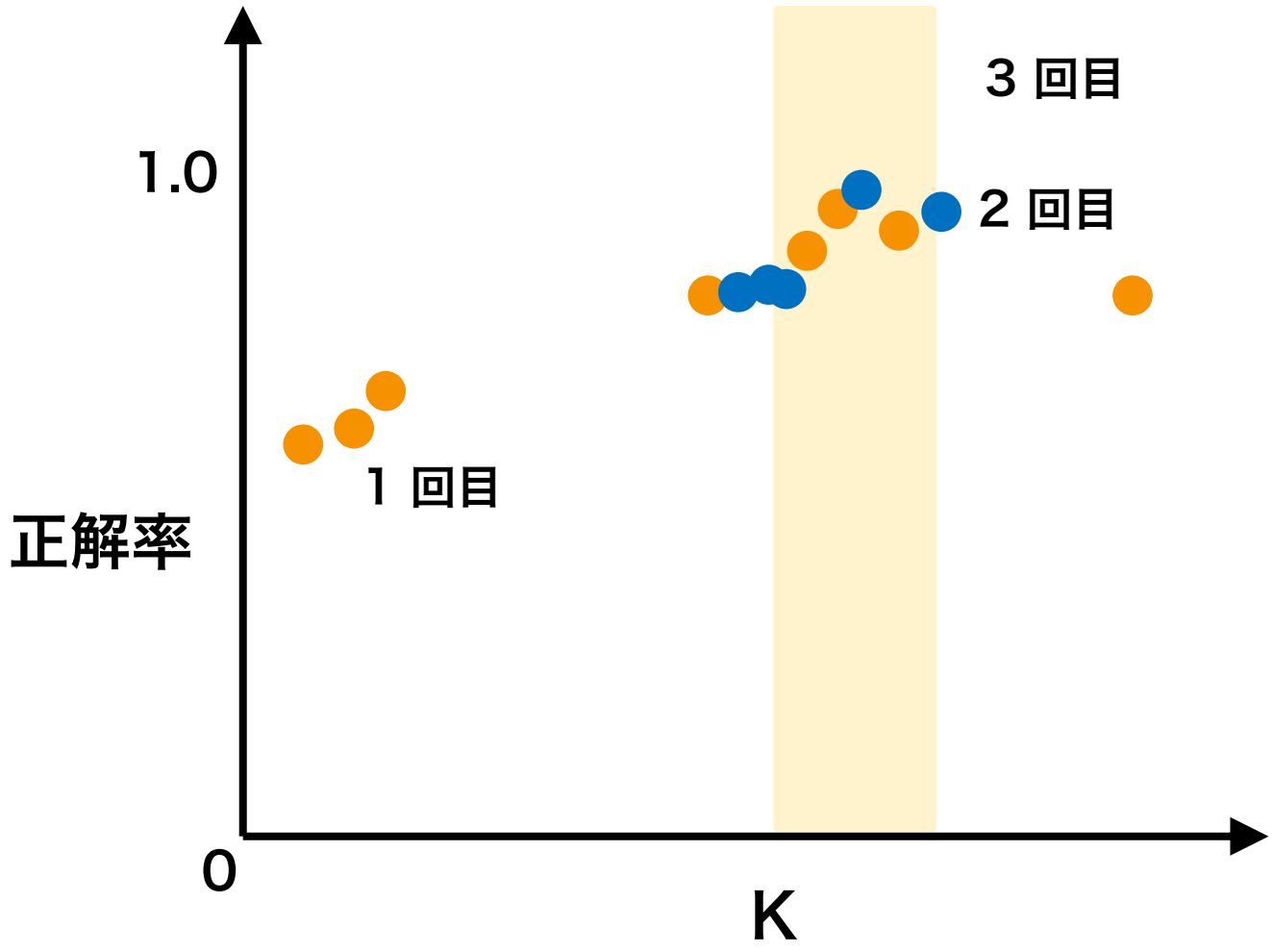
ランダムサーチ

ランダムサーチは、指定された範囲からランダムに K の候補を生成して検証を行う方法である。ランダムサーチも、グリッドサーチと同様に、複数回繰り返して、最適な K を決めることもできる。ただし、K はランダムに生成されるため、最適解あるいはその近傍にたどり着けない場合がある。



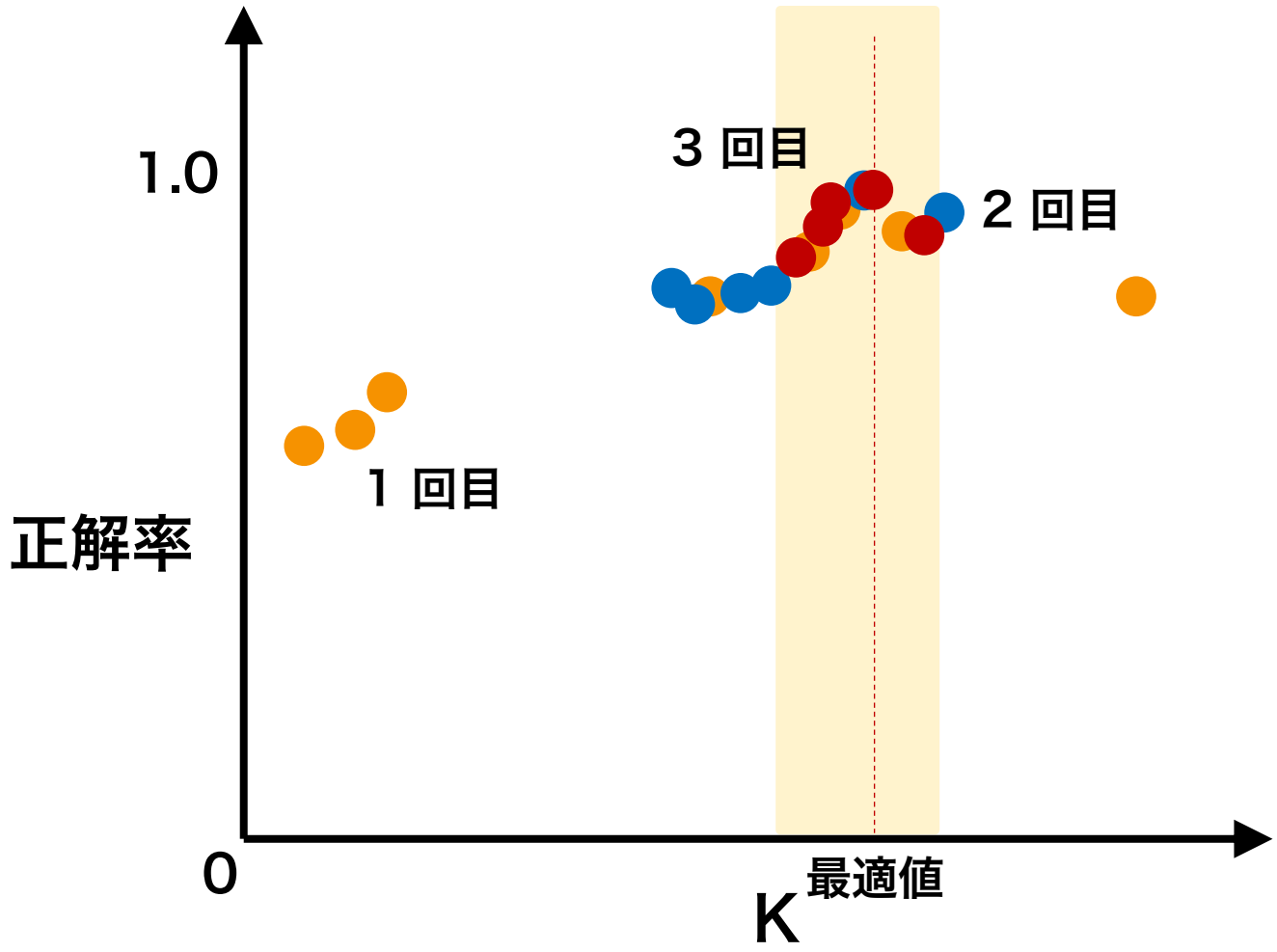
ランダムサーチ

ランダムサーチは、指定された範囲からランダムに K の候補を生成して検証を行う方法である。ランダムサーチも、グリッドサーチと同様に、複数回繰り返して、最適な K を決めることもできる。ただし、K はランダムに生成されるため、最適解あるいはその近傍にたどり着けない場合がある。



ランダムサーチ

ランダムサーチは、指定された範囲からランダムに K の候補を生成して検証を行う方法である。ランダムサーチも、グリッドサーチと同様に、複数回繰り返して、最適な K を決めることもできる。ただし、K はランダムに生成されるため、最適解あるいはその近傍にたどり着けない場合がある。



モデル評価

- パラメーターチューニング
- 評価データセット
- バイアスとバリアンス
- 評価指標

統計モデル

統計学では、過去の観測データを説明するために、モデルを作成する傾向が強い。そのため、統計学におけるモデリングは、モデル作成とモデル評価の両方において同じデータセットを使用する。

モデル作成時に使用したデータと同じデータセットで評価を行うと、評価結果は「モデル作成時に使用したデータセットに対する評価結果」になる。つまり、この評価結果は、このモデルが過去のデータをどのぐらい正確に説明できたのかを示している。

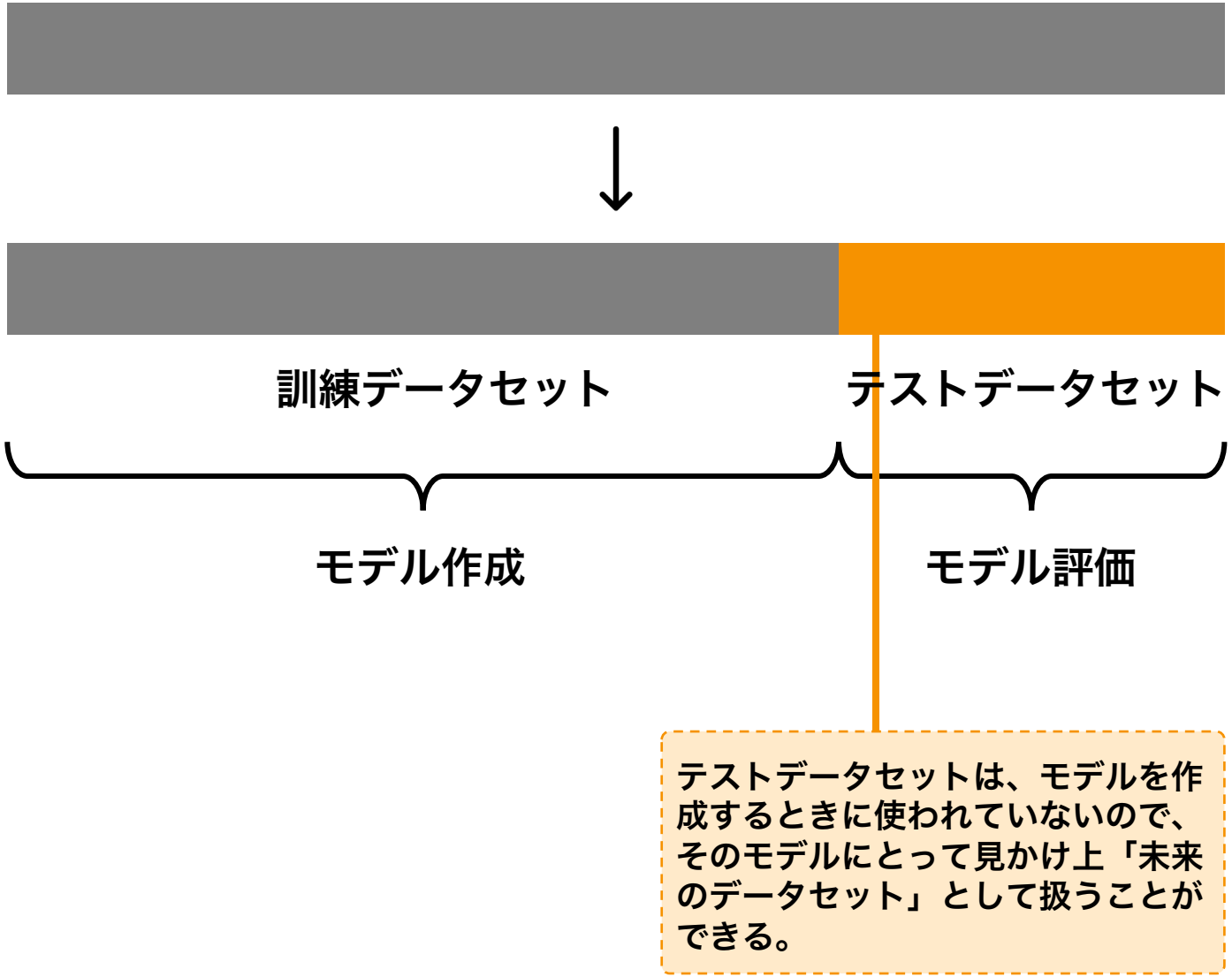
モデル作成

モデル評価

機械学習モデル

機械学習の分野では、過去の観測データから規則性を見出し、その規則性を用いて未来予測に重みを置いてある。未来の観測データがない場合は、過去のデータセットを分割し、見かけ上未来の観測データを作ることで対応する。

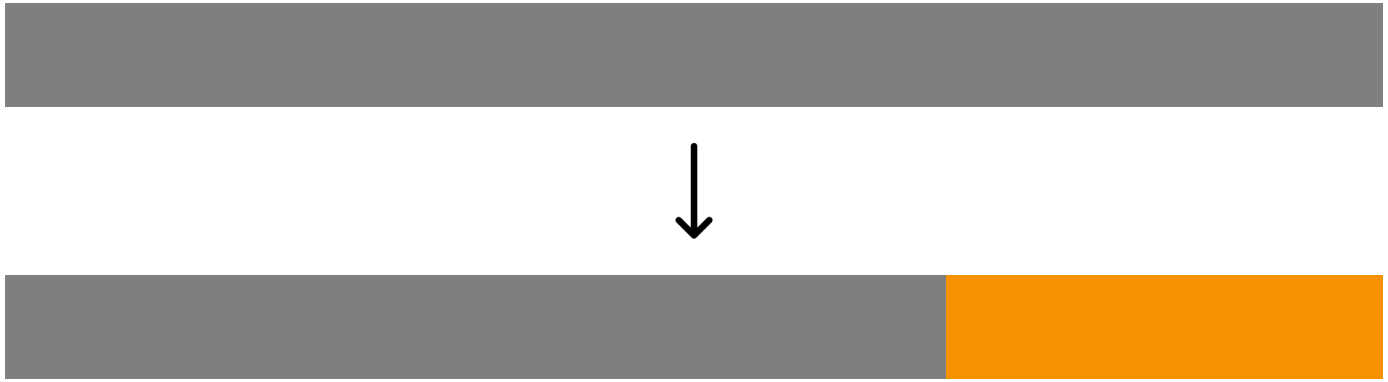
モデル作成時に使用したデータと異なる未知のデータセットで評価を行うと、評価結果は「未知のデータセットに対するモデルの性能評価」になる。つまり、この評価結果は、このモデルが未知のデータをどのぐらい正確に説明できる（予測できる）のかを示している。



テストデータセットと過学習

機械学習の分野では、過去の観測データから規則性を見出し、その規則性を用いて未来予測に重みを置いてある。未来の観測データがない場合は、過去のデータセットを分割し、見かけ上未来の観測データを作ることで対応する。

しかし、テストデータセットを繰り返しモデル検証に使用したとき、テストデータセットに過剰適合するモデルが作られてしまう。つまり、テストデータセットも訓練データセットとして使われてしまったことになる。



訓練データセット　　テストデータセット

1. $\lambda = 10^{-9}$ として、訓練データセットで学習し、テストセットで検証して、正解率が 0.8 となった。
2. $\lambda = 10^{-8}$ として、訓練データセットで学習し、テストセットで検証して、正解率が 0.9 となった。
3. $\lambda = 10^{-7}$ として、訓練データセットで学習し、テストセットで検証して、正解率が 0.7 となった。
4. ...

同じテストデータセットを使用して検証を繰り返すことで、「テストデータセット」に最適なモデルが構築される。

モデル評価

訓練データの複雑さに対して、パラメーターの少ない単純なモデルを適用すると学習不足（under fitting）になり、パラメーターの多い複雑なモデルを適用すると過学習（over fitting）に陥る。学習不足・過学習のない汎用性のあるモデルを作成するには、モデルの評価を正確に行う必要がある。

ホールドアウト法

機械学習モデルを構築するための全データセットを訓練データセット、検証データセット、テストデータセットに分割して、モデルの構築と評価を行う。

k 分割交差検証

機械学習モデルを構築するための全データセットを k 個のサブセットに分割して、k - 1 個のサブセットを使ってモデルを構築し、残りの 1 サブセットを使ってモデルの検証を行う。

ホールドアウト法

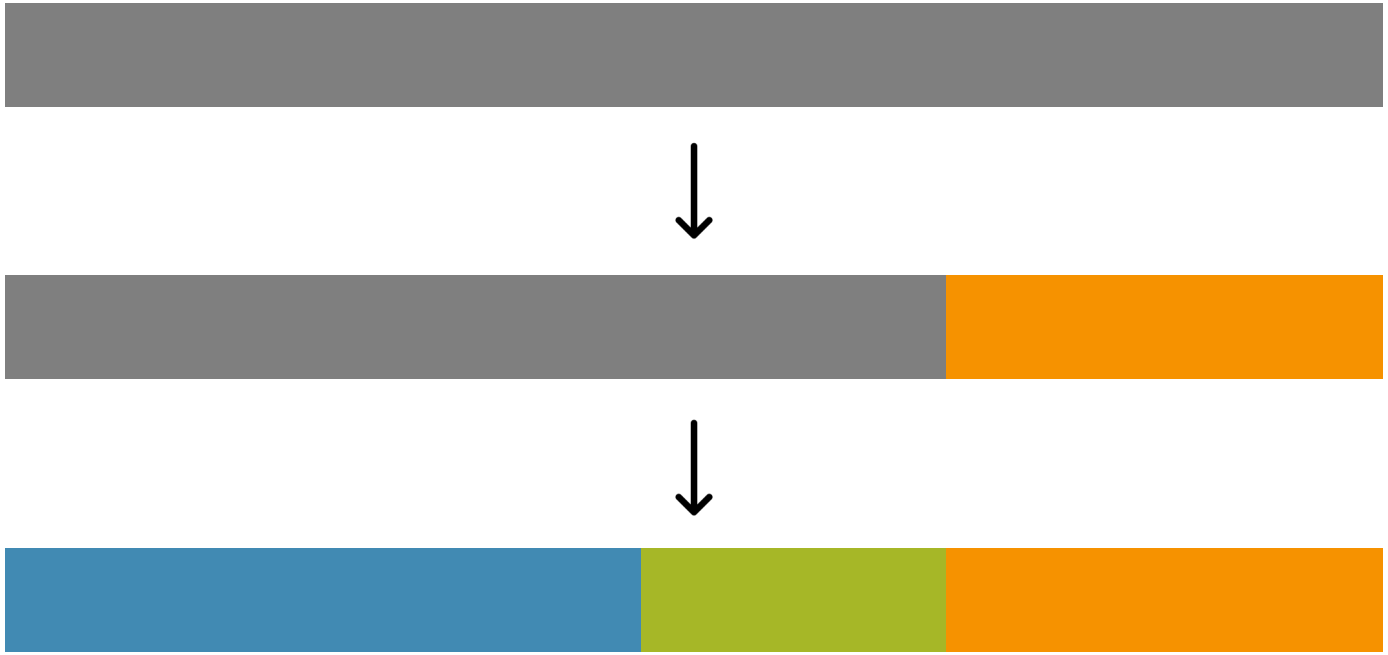
ホールドアウト法は、機械学習モデルを構築するための全データセットを訓練データセット、検証データセット、テストデータセットに分割して、モデルの構築と評価を行う方法である。

1. ハイパーパラメータをある値に固定して、訓練データセットを用いてモデルの学習を行い、パラメータを推定する。

2. 検証データセットを用いて、学習済みのモデルを評価する。

3. ハイパーパラメータの値を変えて、手順1-2を繰り返し、最適なパラメータとハイパーパラメータの組み合わせを求める。

4. 最適な組み合わせに対して、テストデータセットを用いてモデルの最終評価を行う。



訓練データセット

モデルのパラメータを推定するために使われる。

検証データセット

モデル構築時のハイパーパラメータをチューニングするために使われる。

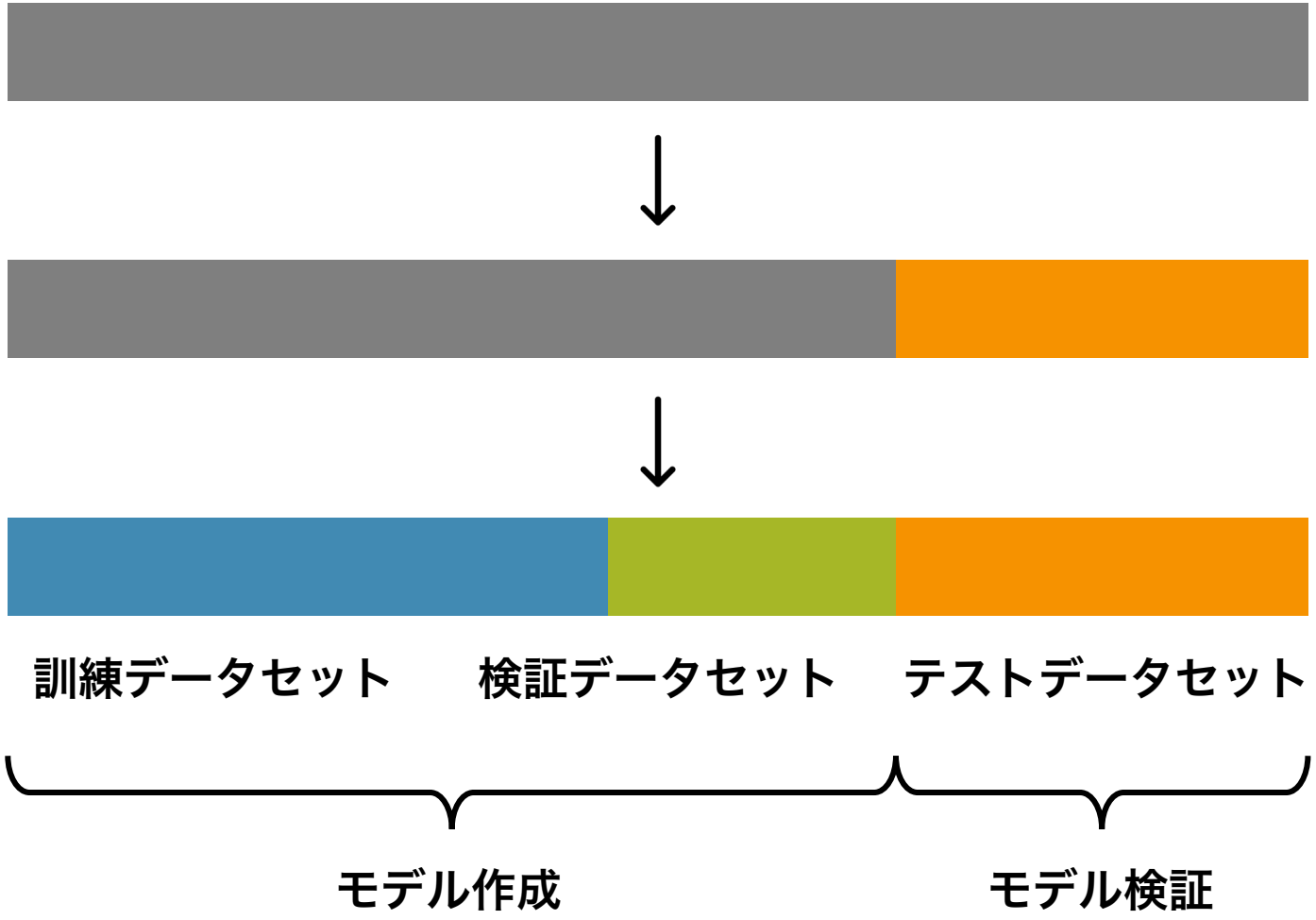
テストデータセット

最適なパラメータとハイパーパラメータで構築されたモデルの性能を評価するために使われる。

ホールドアウト法

ホールドアウト法は、機械学習モデルを構築するための全データセットを訓練データセット、検証データセット、テストデータセットに分割して、モデルの構築と評価を行う方法である。

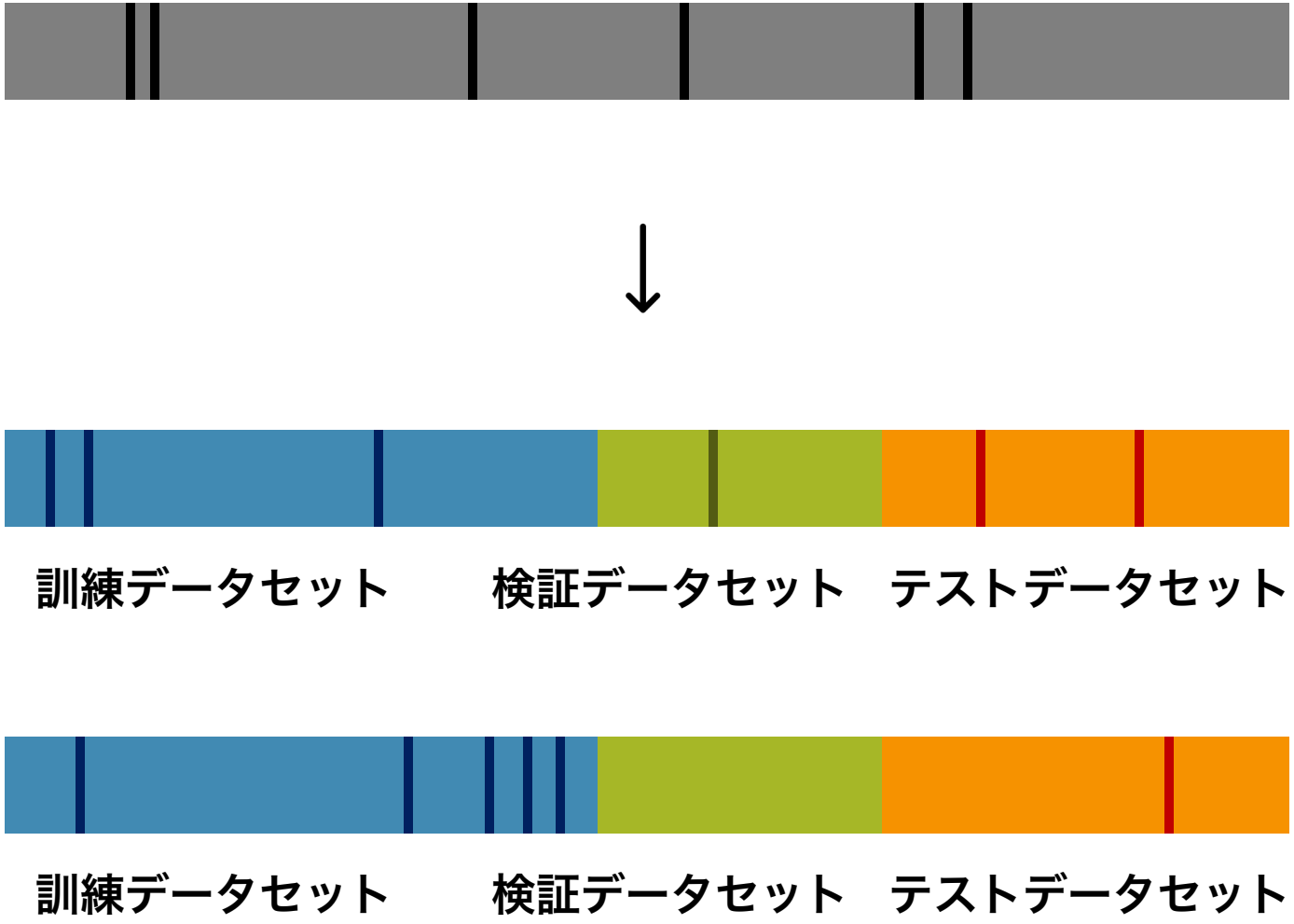
- モデルの学習と評価に使われるデータセットが分かれている。モデルの性能を評価するときは"未知"のデータセットを使用しているので、高性能と評価されたモデルは、未知のデータに対して高性能を発揮できる。



ホールドアウト法

ホールドアウト法は、機械学習モデルを構築するための全データセットを訓練データセット、検証データセット、テストデータセットに分割して、モデルの構築と評価を行う方法である。

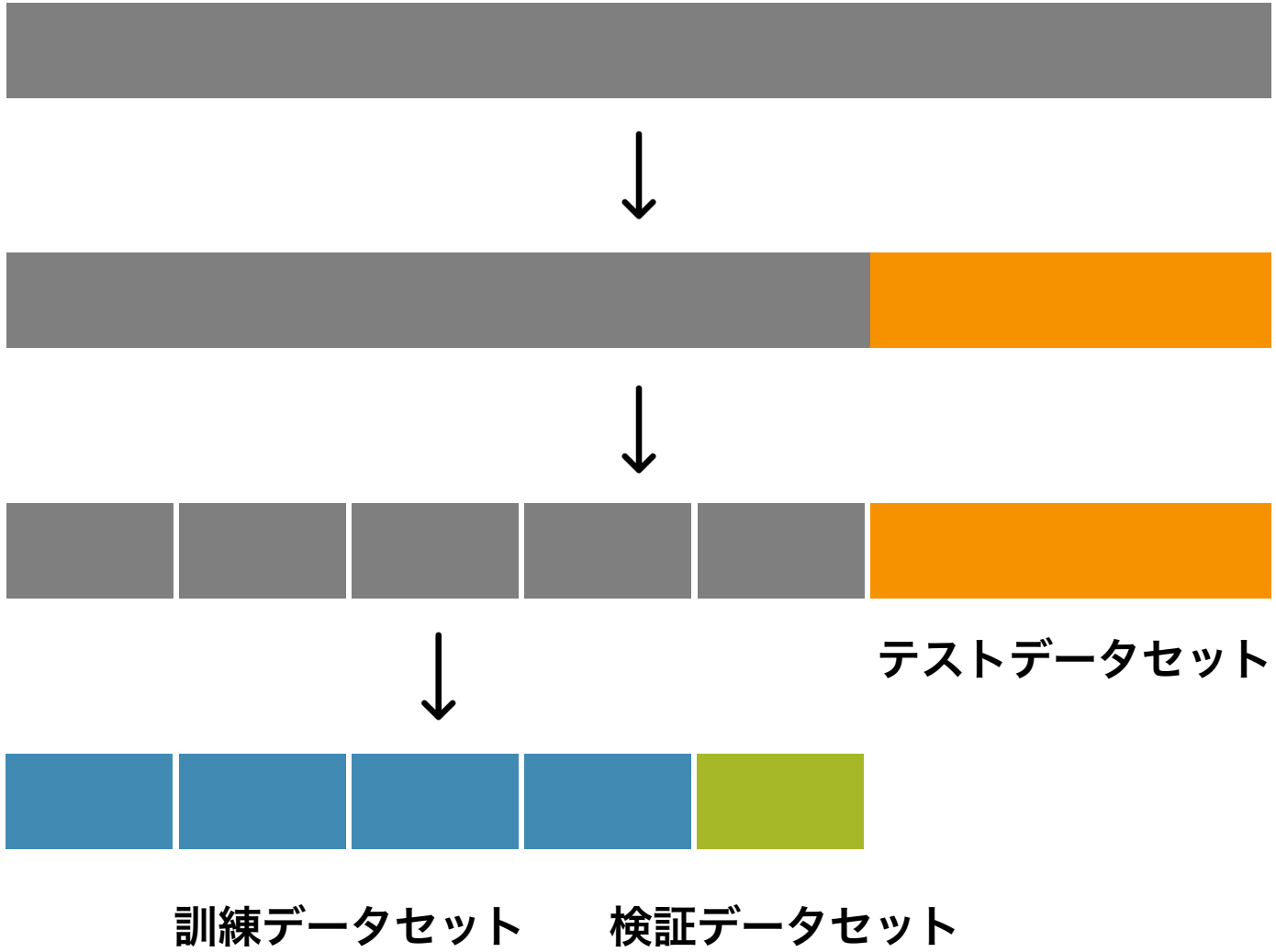
- モデルの学習と評価に使われるデータセットが分かれている。モデルの性能を評価するときは"未知"のデータセットを使用しているので、高性能と評価されたモデルは、未知のデータに対して高性能を発揮できる。
- データセットを分割するとき、分割方法によってモデルの性能評価に影響を与える場合がある。



k 分割交差検証

k 分割交差検証は、機械学習モデルを構築するための全データセットを k 個のサブセットに分割して、k - 1 個のサブセットを使ってモデルを構築し、残りの 1 サブセットを使ってモデルの検証を行う方法である。

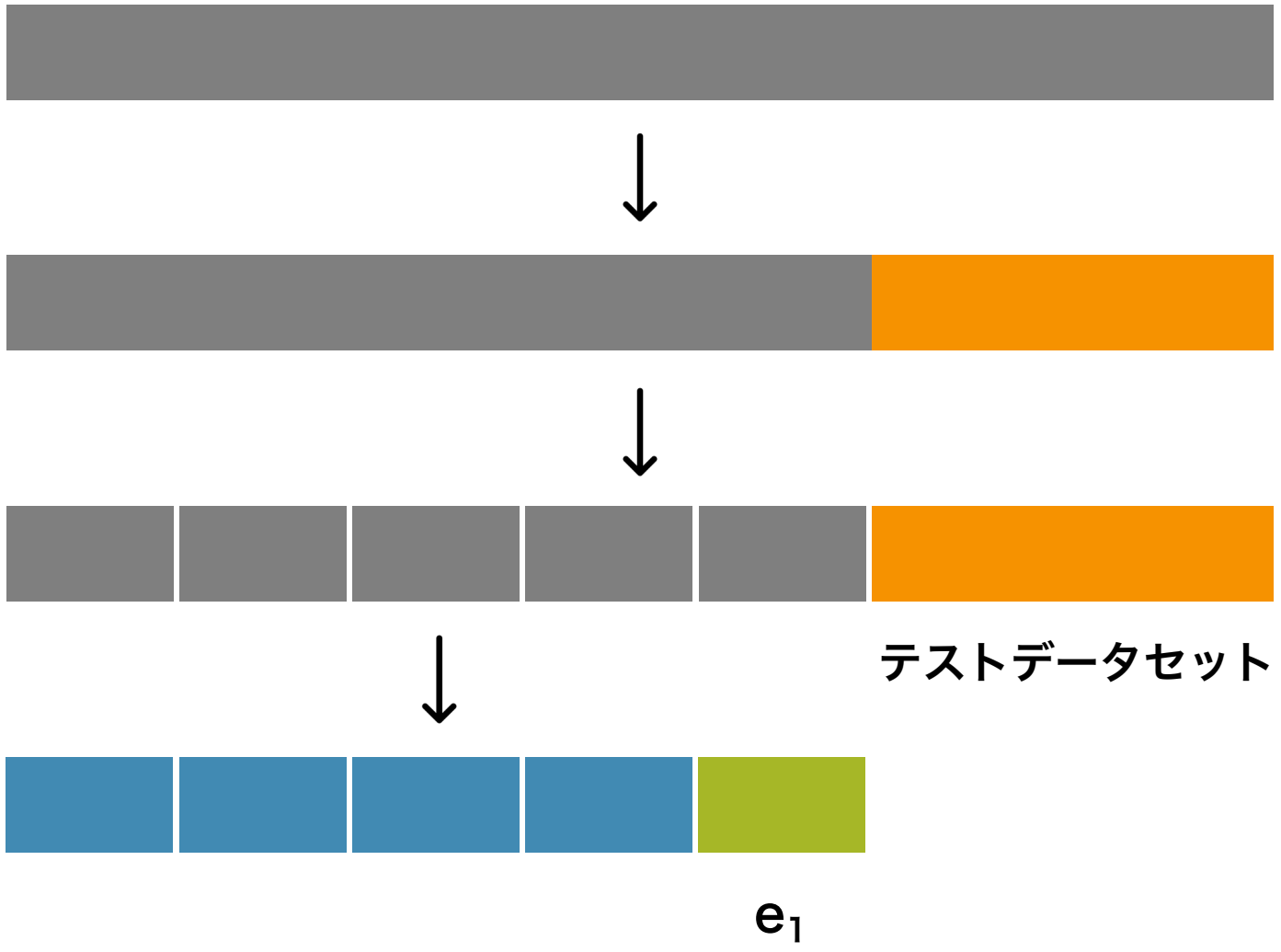
1. ハイパーパラメータをある値に固定して、訓練データセットを用いてモデルの学習を行い、パラメータを推定する。
2. 検証データセットを用いて、学習済みのモデルを評価する。
3. 訓練データセットと検証データセットを順に入れ替えて、k 回の学習と検証を行い、その平均値を検証結果とする。
4. ハイパーパラメータの値を変えて、手順1-3を繰り返し、最適なパラメータとハイパーパラメータの組み合わせを求める。
5. 最適な組み合わせに対して、全訓練データセットでモデルを構築し、テストデータセットを用いてそのモデルの最終評価を行う。



k 分割交差検証

k 分割交差検証は、機械学習モデルを構築するための全データセットを k 個のサブセットに分割して、k - 1 個のサブセットを使ってモデルを構築し、残りの 1 サブセットを使ってモデルの検証を行う方法である。

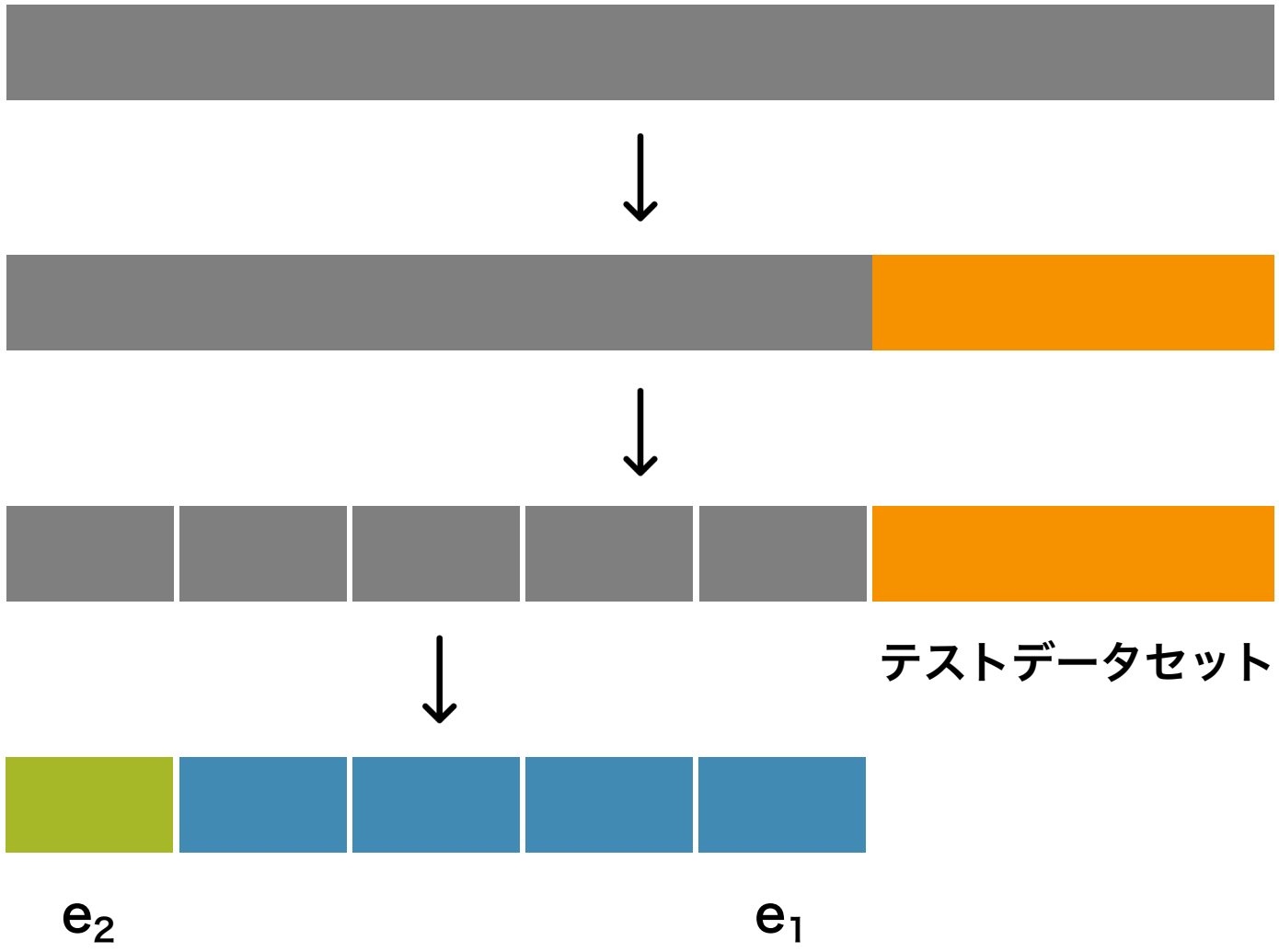
1. ハイパーパラメータをある値に固定して、訓練データセットを用いてモデルの学習を行い、パラメータを推定する。
2. 検証データセットを用いて、学習済みのモデルを評価する。
3. 訓練データセットと検証データセットを順に入れ替えて、k 回の学習と検証を行い、その平均値を検証結果とする。
4. ハイパーパラメータの値を変えて、手順1-3を繰り返し、最適なパラメータとハイパーパラメータの組み合わせを求める。
5. 最適な組み合わせに対して、全訓練データセットでモデルを構築し、テストデータセットを用いてそのモデルの最終評価を行う。



k 分割交差検証

k 分割交差検証は、機械学習モデルを構築するための全データセットを k 個のサブセットに分割して、k - 1 個のサブセットを使ってモデルを構築し、残りの 1 サブセットを使ってモデルの検証を行う方法である。

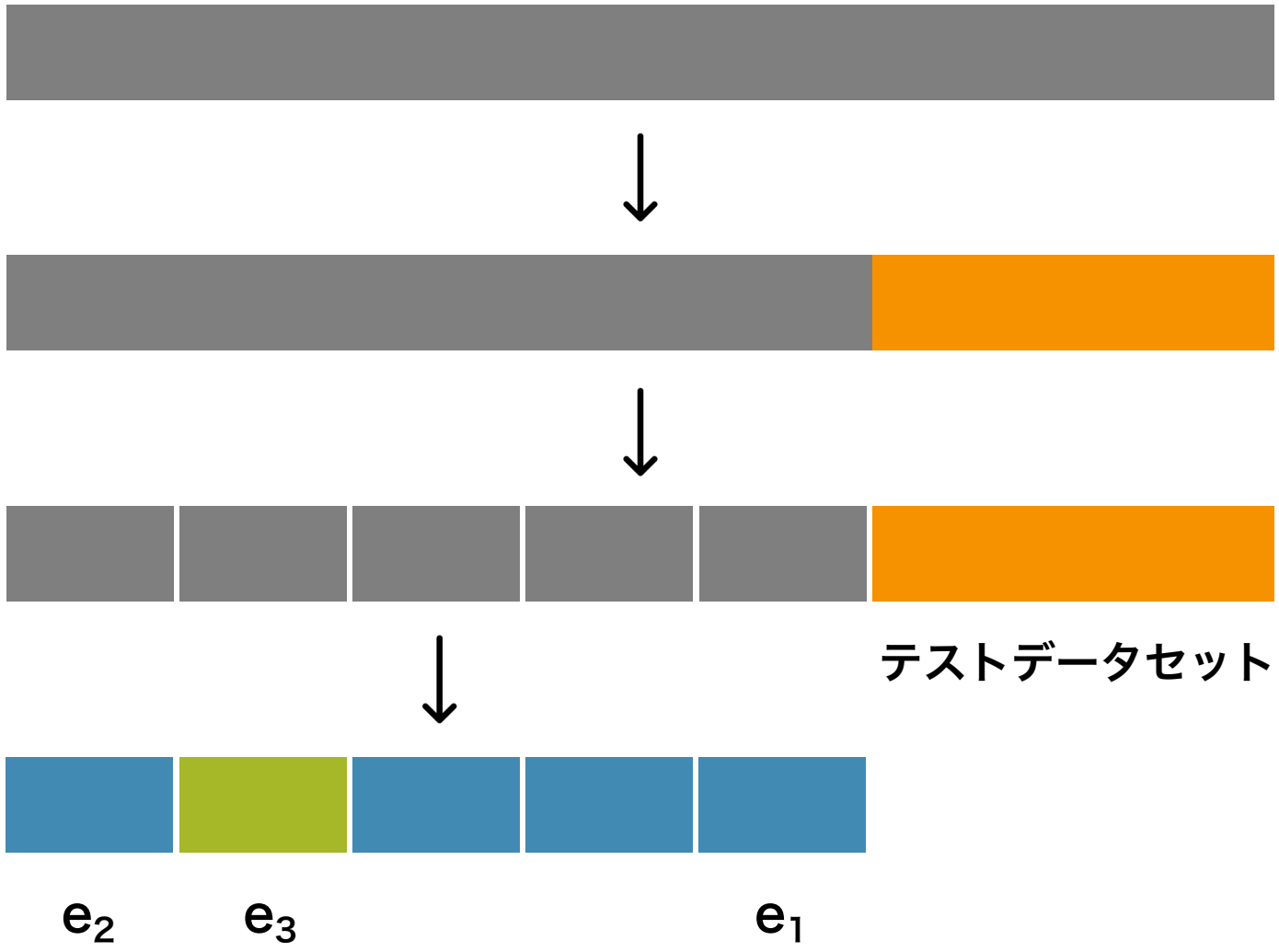
1. ハイパーパラメータをある値に固定して、訓練データセットを用いてモデルの学習を行い、パラメータを推定する。
2. 検証データセットを用いて、学習済みのモデルを評価する。
3. 訓練データセットと検証データセットを順に入れ替えて、k 回の学習と検証を行い、その平均値を検証結果とする。
4. ハイパーパラメータの値を変えて、手順1-3を繰り返し、最適なパラメータとハイパーパラメータの組み合わせを求める。
5. 最適な組み合わせに対して、全訓練データセットでモデルを構築し、テストデータセットを用いてそのモデルの最終評価を行う。



k 分割交差検証

k 分割交差検証は、機械学習モデルを構築するための全データセットを k 個のサブセットに分割して、k - 1 個のサブセットを使ってモデルを構築し、残りの 1 サブセットを使ってモデルの検証を行う方法である。

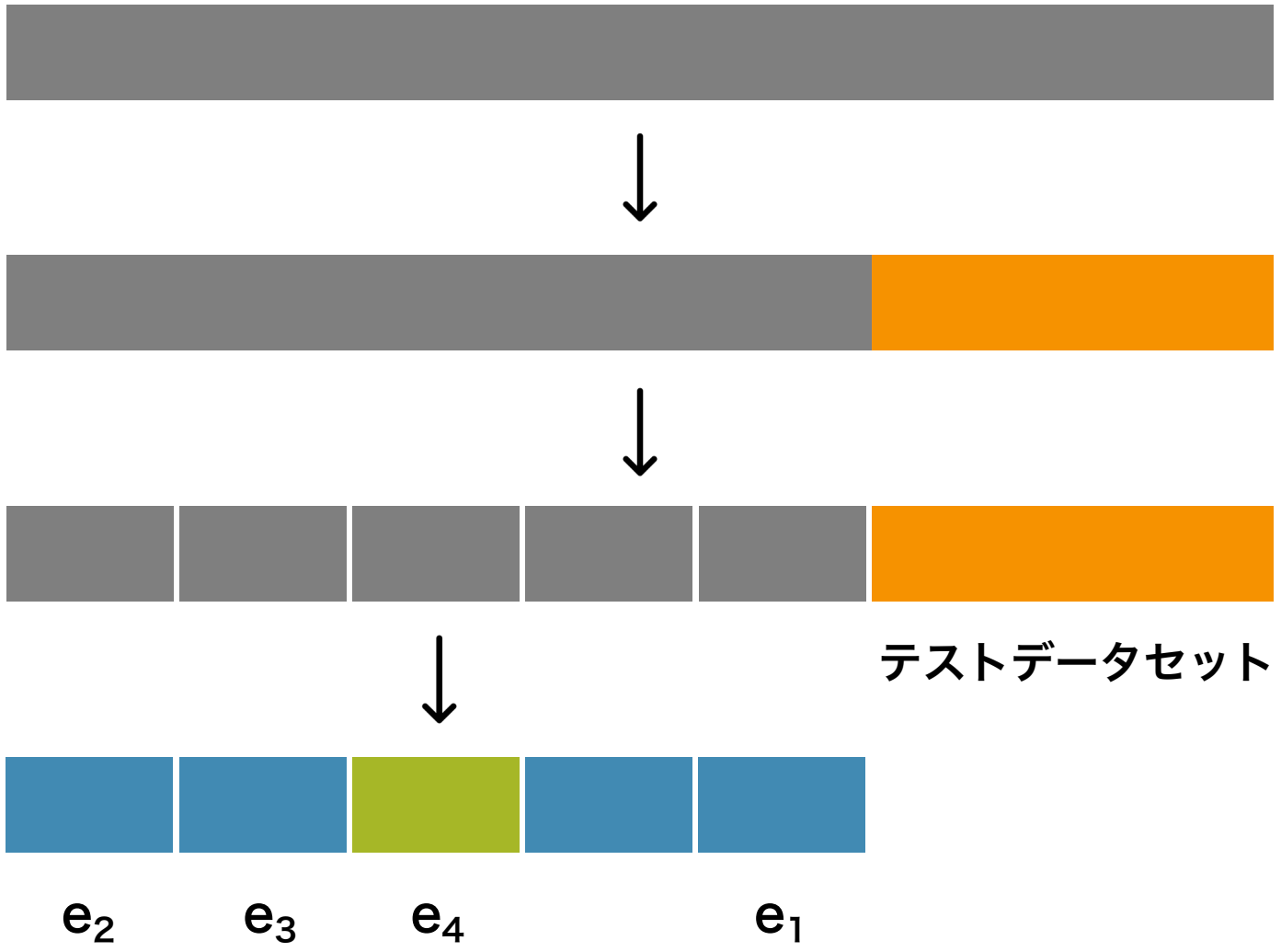
1. ハイパーパラメータをある値に固定して、訓練データセットを用いてモデルの学習を行い、パラメータを推定する。
2. 検証データセットを用いて、学習済みのモデルを評価する。
3. 訓練データセットと検証データセットを順に入れ替えて、k 回の学習と検証を行い、その平均値を検証結果とする。
4. ハイパーパラメータの値を変えて、手順1-3を繰り返し、最適なパラメータとハイパーパラメータの組み合わせを求める。
5. 最適な組み合わせに対して、全訓練データセットでモデルを構築し、テストデータセットを用いてそのモデルの最終評価を行う。



k 分割交差検証

k 分割交差検証は、機械学習モデルを構築するための全データセットを k 個のサブセットに分割して、k - 1 個のサブセットを使ってモデルを構築し、残りの 1 サブセットを使ってモデルの検証を行う方法である。

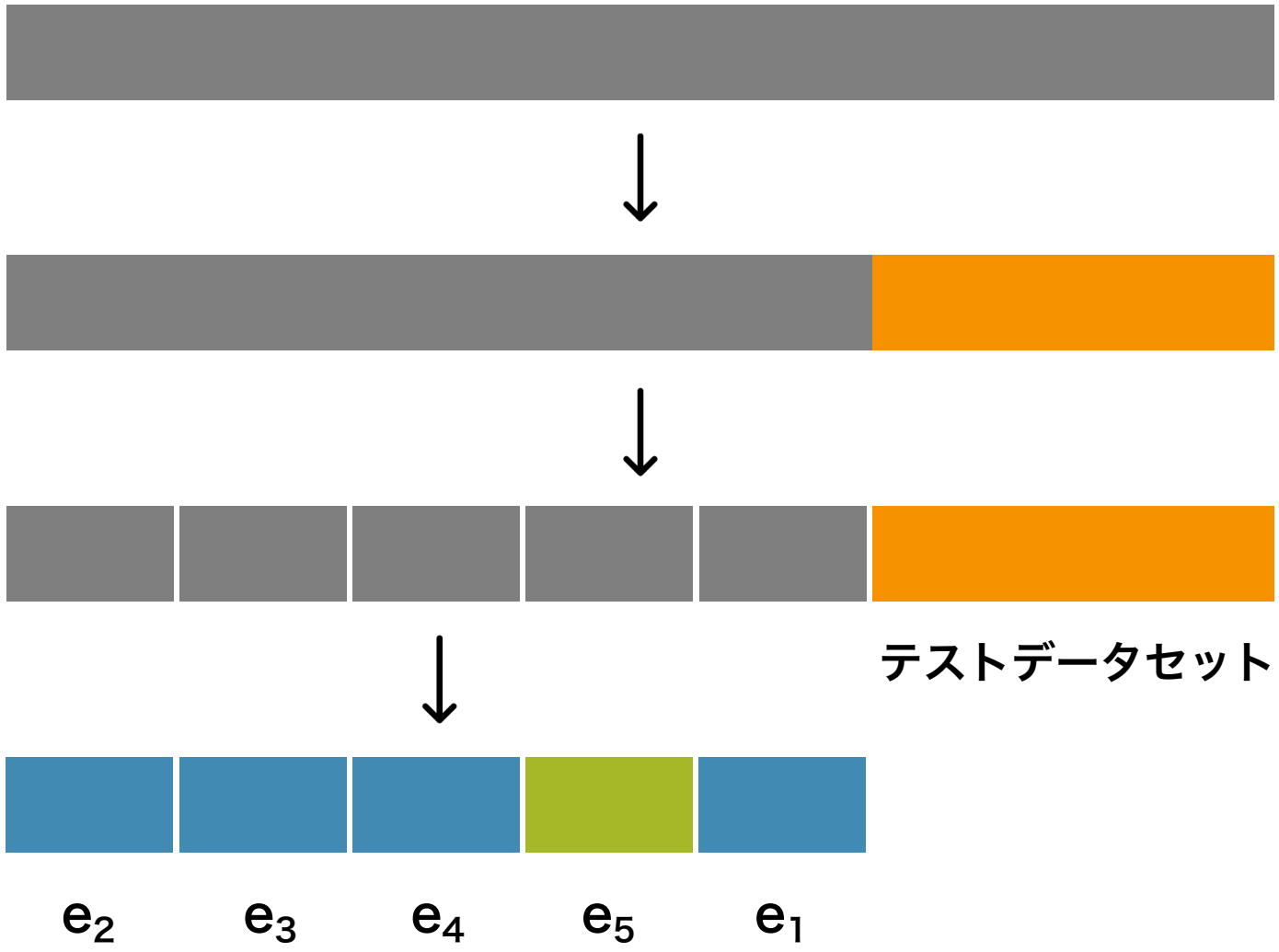
1. ハイパーパラメータをある値に固定して、訓練データセットを用いてモデルの学習を行い、パラメータを推定する。
2. 検証データセットを用いて、学習済みのモデルを評価する。
3. 訓練データセットと検証データセットを順に入れ替えて、k 回の学習と検証を行い、その平均値を検証結果とする。
4. ハイパーパラメータの値を変えて、手順1-3を繰り返し、最適なパラメータとハイパーパラメータの組み合わせを求める。
5. 最適な組み合わせに対して、全訓練データセットでモデルを構築し、テストデータセットを用いてそのモデルの最終評価を行う。



k 分割交差検証

k 分割交差検証は、機械学習モデルを構築するための全データセットを k 個のサブセットに分割して、k - 1 個のサブセットを使ってモデルを構築し、残りの 1 サブセットを使ってモデルの検証を行う方法である。

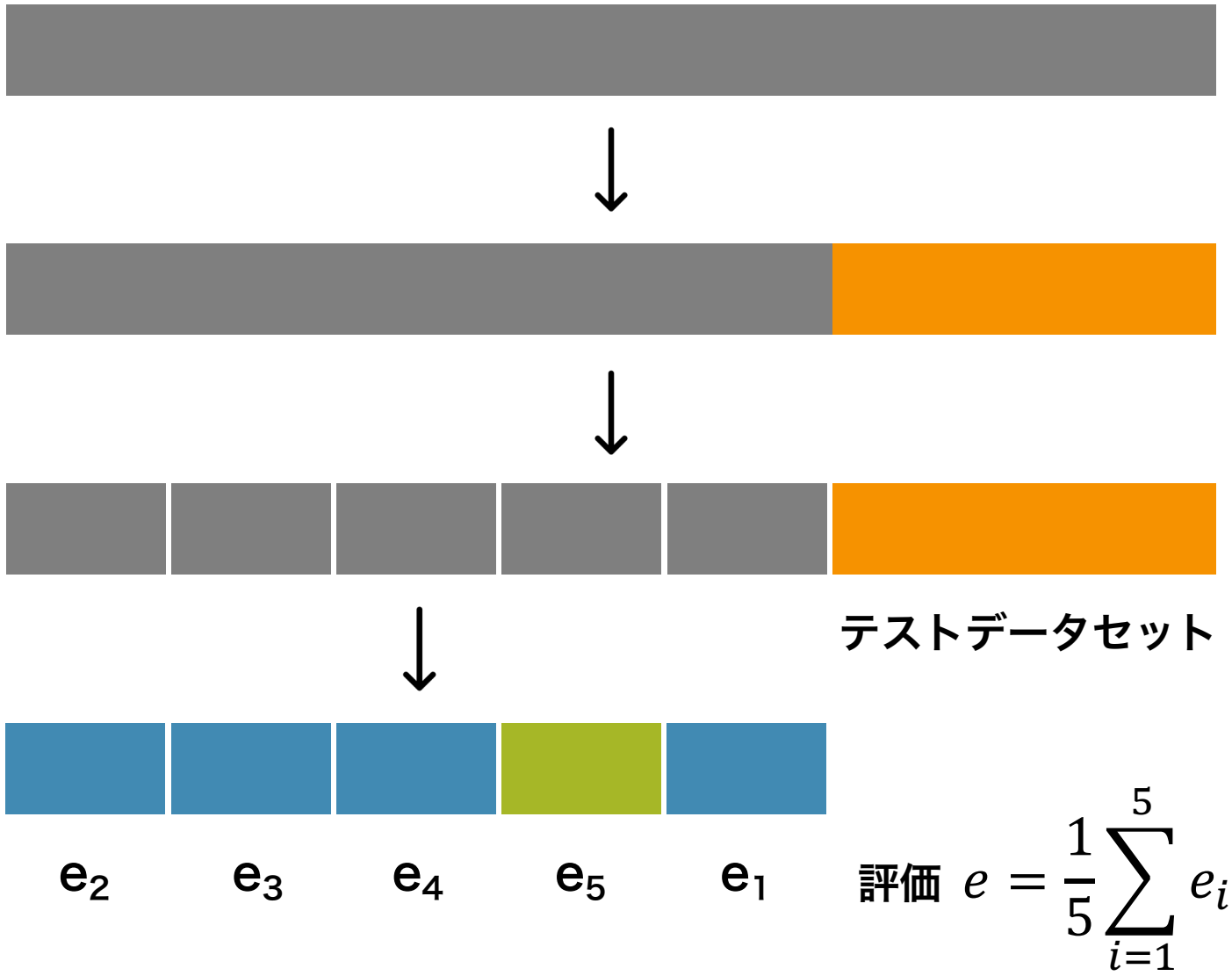
1. ハイパーパラメータをある値に固定して、訓練データセットを用いてモデルの学習を行い、パラメータを推定する。
2. 検証データセットを用いて、学習済みのモデルを評価する。
3. 訓練データセットと検証データセットを順に入れ替えて、k 回の学習と検証を行い、その平均値を検証結果とする。
4. ハイパーパラメータの値を変えて、手順1-3を繰り返し、最適なパラメータとハイパーパラメータの組み合わせを求める。
5. 最適な組み合わせに対して、全訓練データセットでモデルを構築し、テストデータセットを用いてそのモデルの最終評価を行う。



k 分割交差検証

k 分割交差検証は、機械学習モデルを構築するための全データセットを k 個のサブセットに分割して、k - 1 個のサブセットを使ってモデルを構築し、残りの 1 サブセットを使ってモデルの検証を行う方法である。

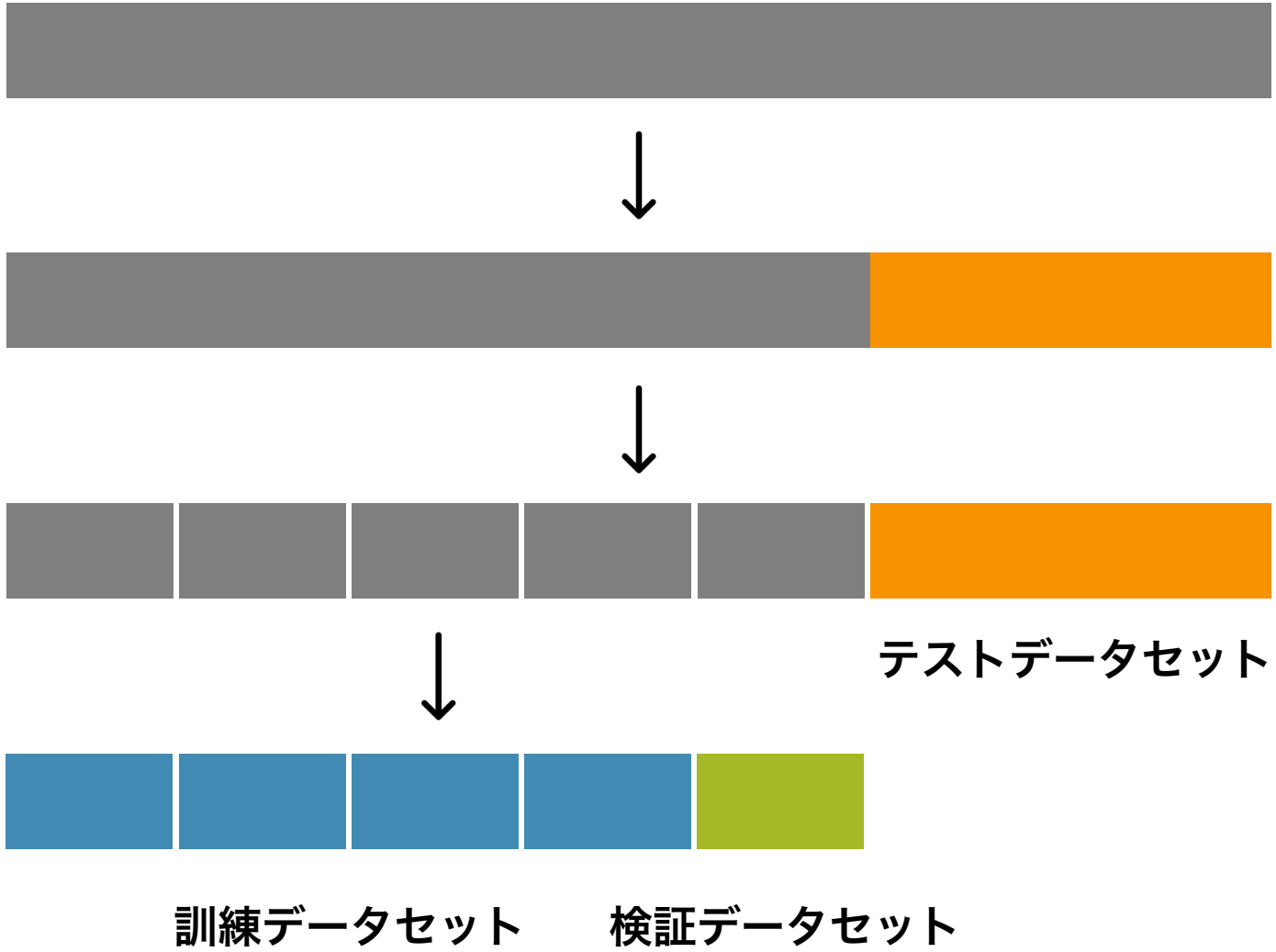
1. ハイパーパラメータをある値に固定して、訓練データセットを用いてモデルの学習を行い、パラメータを推定する。
2. 検証データセットを用いて、学習済みのモデルを評価する。
3. 訓練データセットと検証データセットを順に入れ替えて、k 回の学習と検証を行い、その平均値を検証結果とする。
4. ハイパーパラメータの値を変えて、手順1-3を繰り返し、最適なパラメータとハイパーパラメータの組み合わせを求める。
5. 最適な組み合わせに対して、全訓練データセットでモデルを構築し、テストデータセットを用いてそのモデルの最終評価を行う。



k 分割交差検証

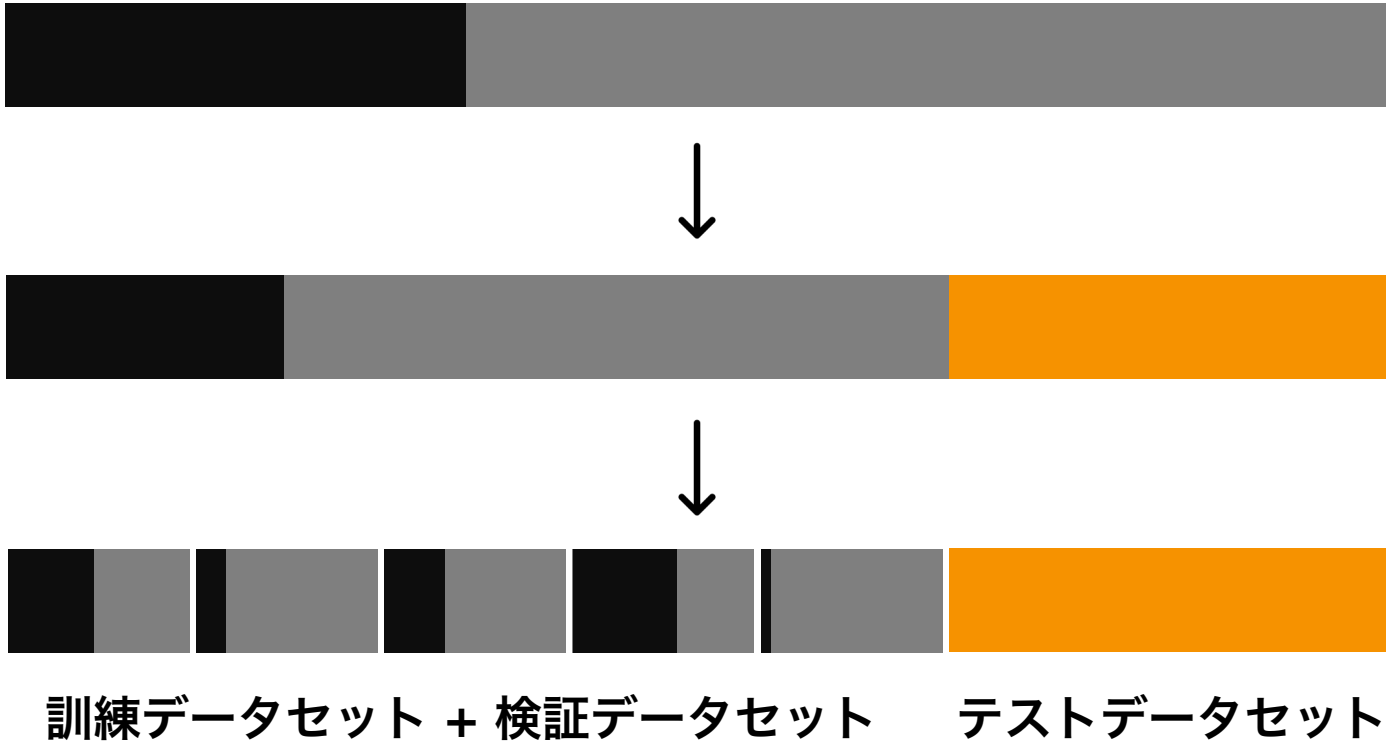
k 分割交差検証における分割数 k は、データセットの大きさに応じて決める。一般に、k が大きいほど、各回の学習に使われるデータが多いので、学習不足が防げられる。しかし、各回の学習で使われる訓練データが互いに似たものになり、過学習に陥りやすい。

- 一般的に k = 10 が推奨される。
- 訓練データセットが大きい場合は、k = 5 でも同等な性能評価の効果が得られる。
- 訓練データセットが非常に小さい場合は、k = n （Leave-One-Out; LOO）が推奨される。
- 訓練データが特定のクラスに偏っている場合は、層化k分割交差検証も検討すること。



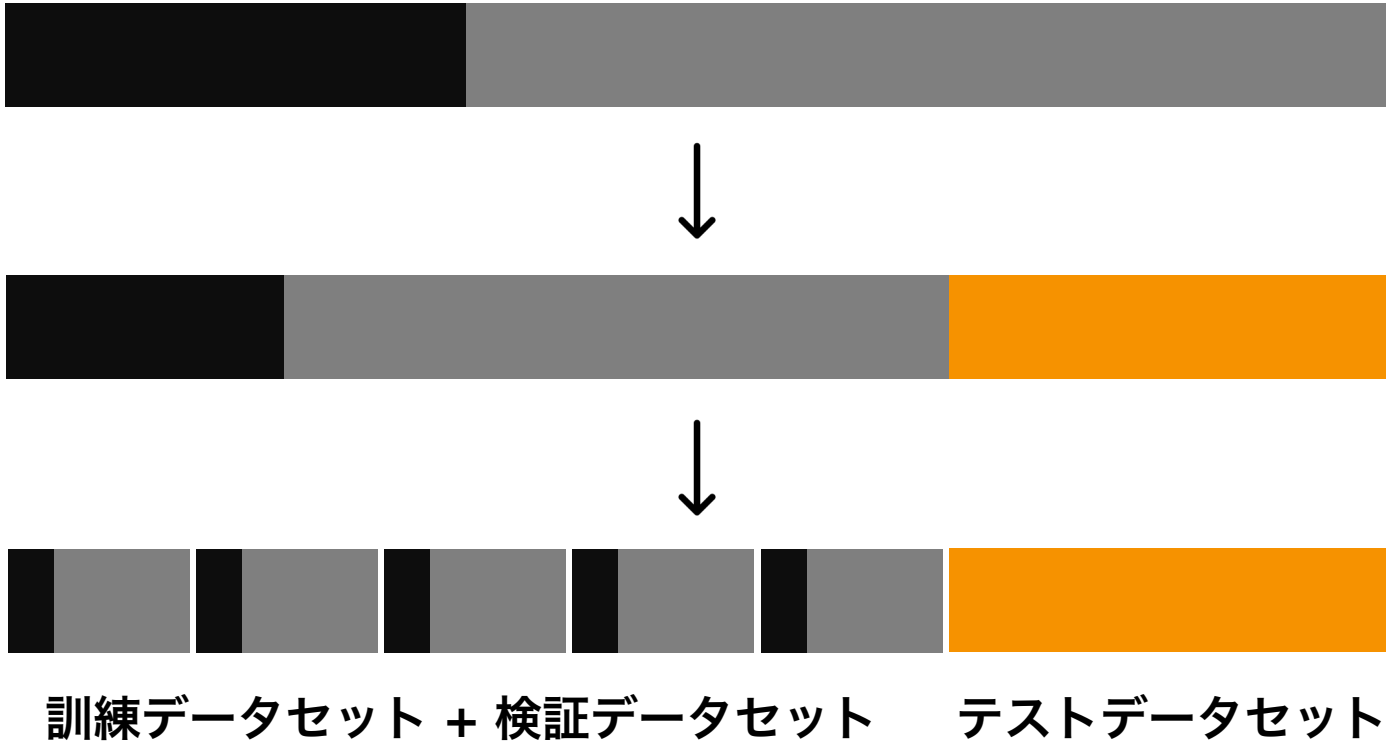
k 分割交差検証

データセットのラベルが不均衡のとき、k 分割交差検証を行うことで、特定のサブセットに特定のラベルが多く含まれるような状況が起こる可能性がある。このようなサブセットは性能評価に悪影響を与える可能性がある。



層化 k 分割交差検証

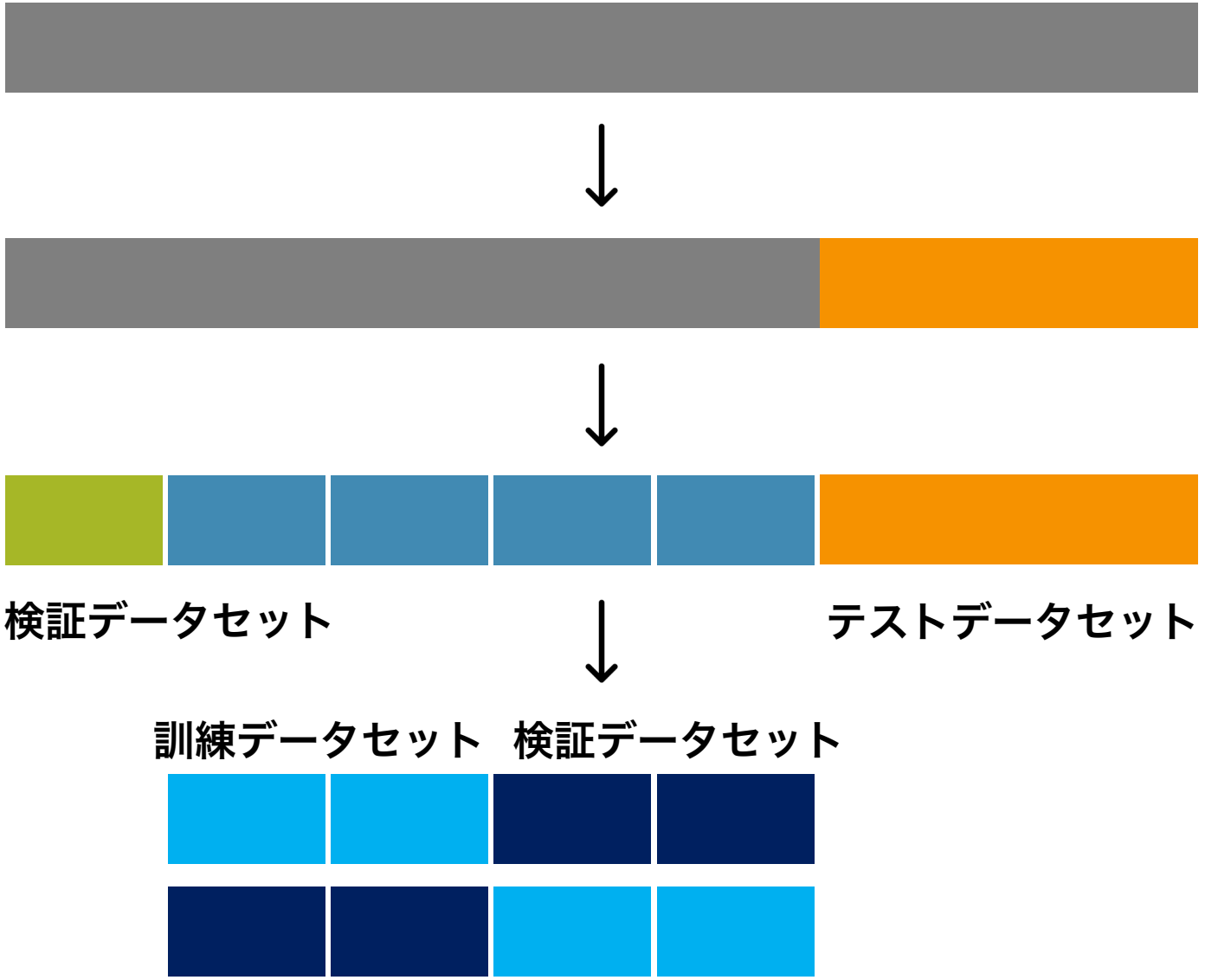
層化 k 分割交差検証では、データセットをサブセットに分割するとき、各サブセット中の各ラベルの比率が同じとなるように分割が行われる。



入れ子式交差検証

複数のアルゴリズム間（例えば SVM とランダムフォレストなど）の比較を行う場合は、入れ子式交差検証が有効な場合もある。

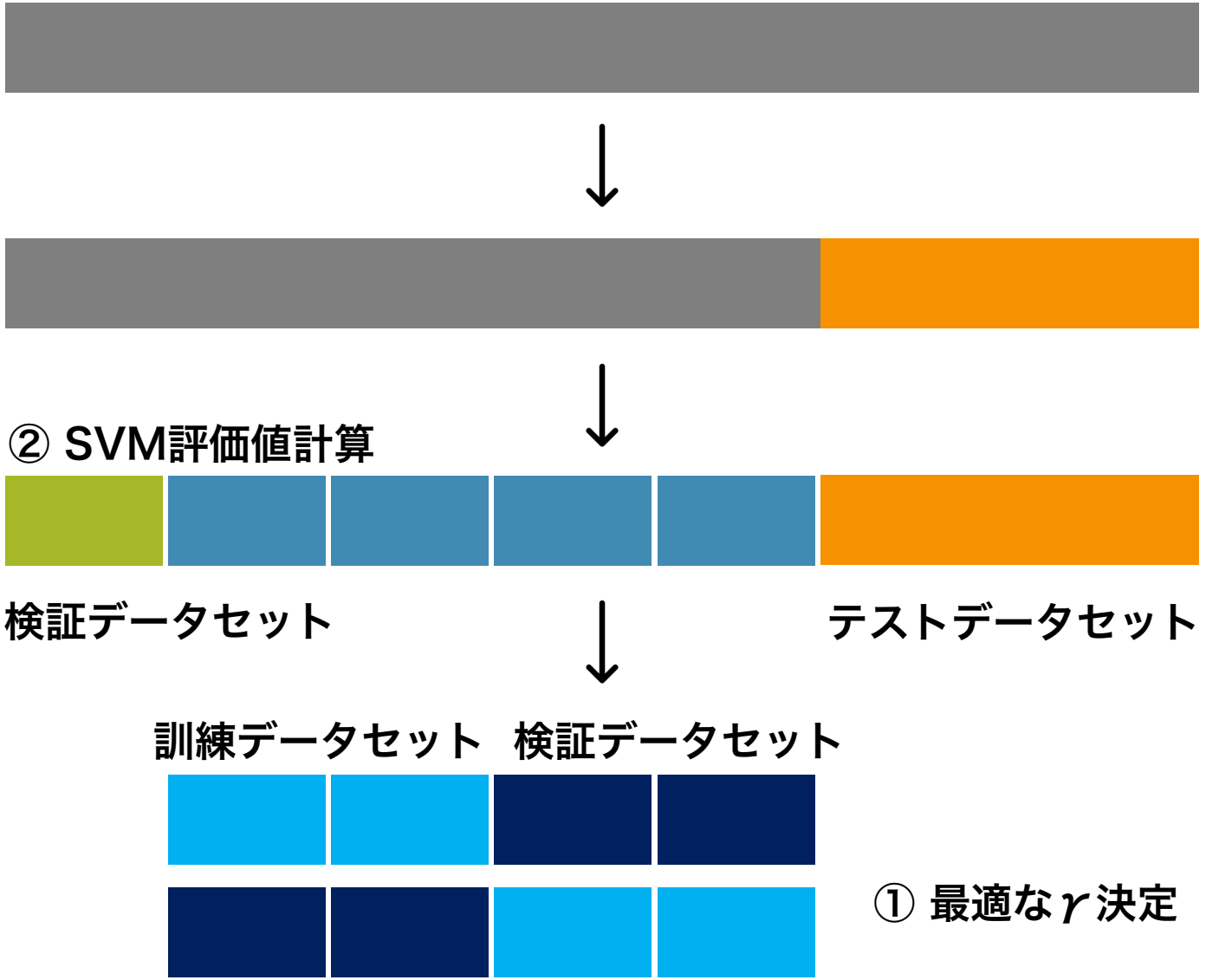
1. アルゴリズムを固定し（例えば SVM）、訓練データセットと検証データセットを用いて 2 分割交差検証を行い、最適なハイパーパラメータを決める。
2. 最適なパラメータで作成したモデルを検証データセットで 5 分割交差検証で選択したアルゴリズム（SVM）の性能評価する。
3. アルゴリズムを変えて（例えば決定木）、手順1-2を繰り返す。これにより、各アルゴリズムで作られたモデルに対する 5 分割交差検証で性能評価による評価結果が得られる。この評価結果をもとに、最適なアルゴリズム（例えばSVM・決定木の中からSVM）を決める。
4. 手順3で得られた最適なモデルに対して、テストデータで最終評価を行う。



入れ子式交差検証

複数のアルゴリズム間（例えば SVM とランダムフォレストなど）の比較を行う場合は、入れ子式交差検証が有効な場合もある。

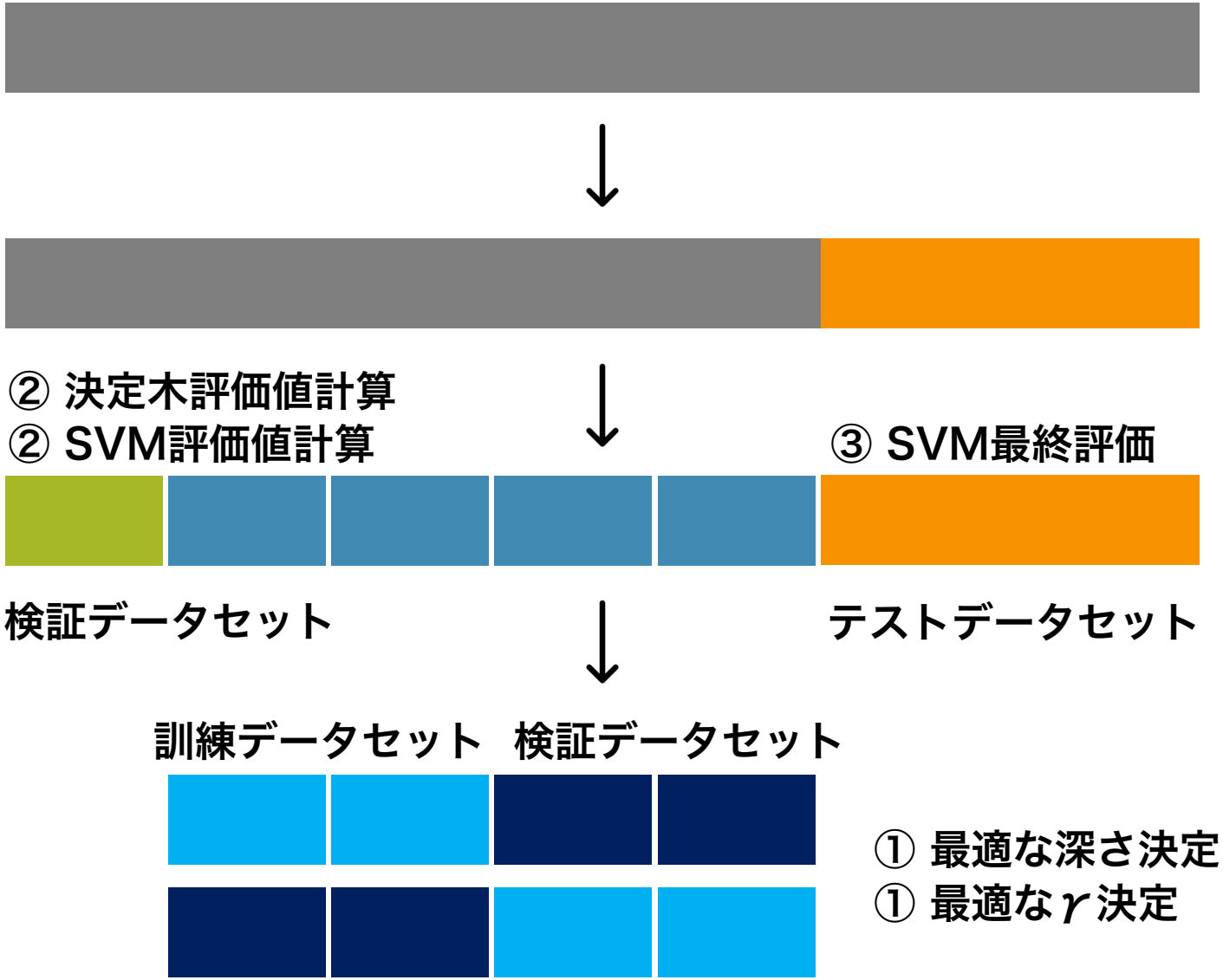
1. アルゴリズムを固定し（例えば SVM）、訓練データセットと検証データセットを用いて 2 分割交差検証を行い、最適なハイパーパラメータを決める。
2. 最適なパラメータで作成したモデルを検証データセットで 5 分割交差検証で選択したアルゴリズム（SVM）の性能評価する。
3. アルゴリズムを変えて（例えば決定木）、手順1-2を繰り返す。これにより、各アルゴリズムで作られたモデルに対する 5 分割交差検証で性能評価による評価結果が得られる。この評価結果をもとに、最適なアルゴリズム（例えばSVM・決定木の中からSVM）を決める。
4. 手順3で得られた最適なモデルに対して、テストデータで最終評価を行う。



入れ子式交差検証

複数のアルゴリズム間（例えば SVM とランダムフォレストなど）の比較を行う場合は、入れ子式交差検証が有効な場合もある。

1. アルゴリズムを固定し（例えば SVM）、訓練データセットと検証データセットを用いて 2 分割交差検証を行い、最適なハイパーパラメータを決める。
2. 最適なパラメータで作成したモデルを検証データセットで 5 分割交差検証で選択したアルゴリズム（SVM）の性能評価する。
3. アルゴリズムを変えて（例えば決定木）、手順1-2を繰り返す。これにより、各アルゴリズムで作られたモデルに対する 5 分割交差検証で性能評価による評価結果が得られる。この評価結果をもとに、最適なアルゴリズム（例えばSVM・決定木の中からSVM）を決める。
4. 手順3で得られた最適なモデルに対して、テストデータで最終評価を行う。



モデル評価

- パラメーターチューニング
- 評価データセット
- バイアスとバリアンス
- 評価指標

バイアスとバリアンス

- バイアス・バリアンスのトレードオフ
- 学習曲線
- 検証曲線

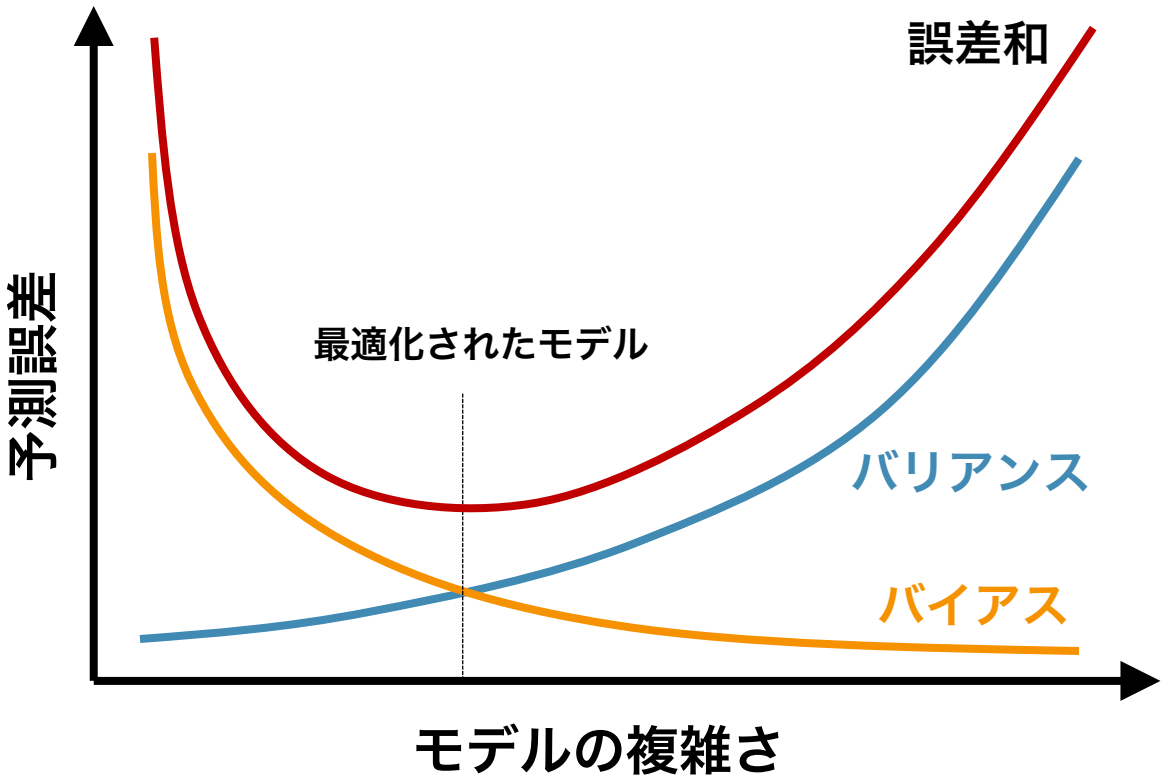
バイアスとバリアンス

- バイアス・バリアンスのトレードオフ
- 学習曲線
- 検証曲線

Bias-Variance Tradeoff

モデルの予測誤差（真の値と予測値の差）は、ノイズ、バリアンス、およびバイアスに分解できる。ノイズはデータそのものに含まれるばらつきであり、そのデータセットを使用する以上、取り除くことができない。ノイズに対して、バリアンスおよびバイアスはモデルの複雑さに由来するばらつきで、調整することができる。一般に、バリアンスが小さく、かつバイアスが小さいモデルの汎化性能が高いと言われている。しかし、バリアンスとバイアスはトレードオフの関係にあり、一方を小さくすれば、他方が大きくなる。この関係をバイアス-バリアンス（偏りと分散）のトレードオフという。

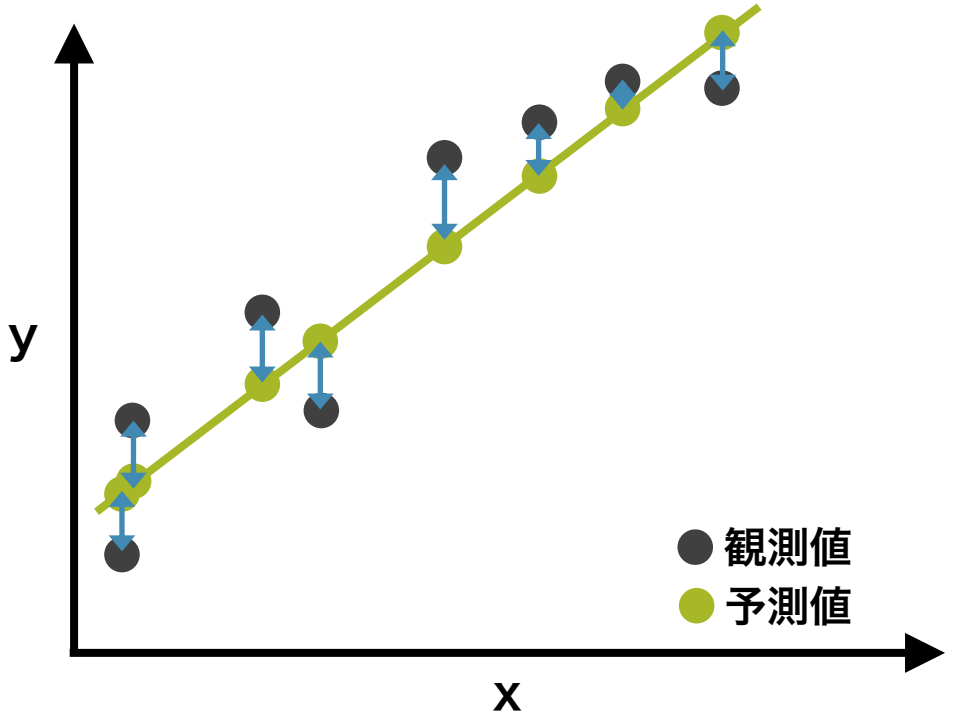
予測誤差 ＝ ノイズ ＋ バリアンス ＋ バイアス



Bias-Variance Tradeoff

$$
\begin{aligned}
& E[(y-\hat{y})^2] \\
&= E[y^2+\hat{y}^2-2y\hat{y}] \\
&= E[y^2]+E[\hat{y}^2]-2E[y\hat{y}] \\
&= \{Var[y]+(E[y^2])^2\}+\{Var[\hat{y}]+(E[\hat{y}^2])^2\}-2E[y\hat{y}] \\
&= Var[y]+Var[\hat{y}]+\{(E[y^2])^2-2E[y\hat{y}]+(E[\hat{y}^2])^2\} \\
&= Var[y]+Var[\hat{y}]+\{(E[y^2])^2-2E[y]E[\hat{y}]+(E[\hat{y}^2])^2\} \\
&= Var[y]+Var[\hat{y}]+(E[y]-E[\hat{y}])^2 \\
&= noise+variance+bias^2
\end{aligned}
$$

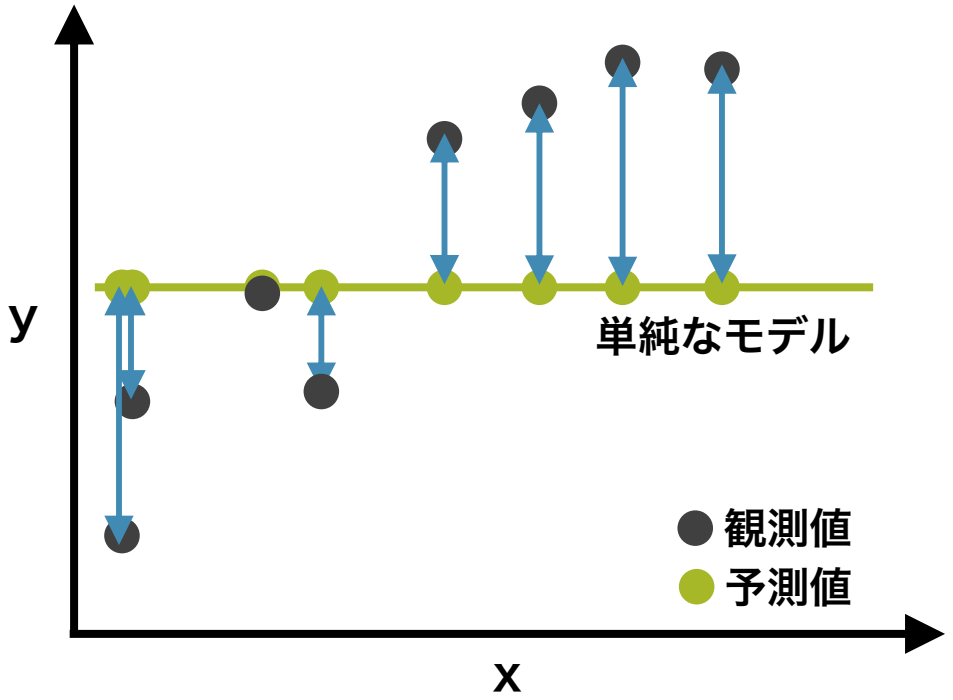
バリアンスを小さくすれば $E[(y-\hat{y})^2]$ が小さくなる。極端な例を挙げれば、モデルのパラメータをなくし単純化させて、 y = 1.0 とすれば、バリアンス $Var[\hat{y}]$ が 0 になる。しかし、このときバイアス $(E[y]-E[\hat{y}])^2$ が大きくなる。



Bias-Variance Tradeoff

$$
\begin{aligned}
& E[(y-\hat{y})^2] \\
&= E[y^2+\hat{y}^2-2y\hat{y}] \\
&= E[y^2]+E[\hat{y}^2]-2E[y\hat{y}] \\
&= \{Var[y]+(E[y^2])^2\}+\{Var[\hat{y}]+(E[\hat{y}^2])^2\}-2E[y\hat{y}] \\
&= Var[y]+Var[\hat{y}]+\{(E[y^2])^2-2E[y\hat{y}]+(E[\hat{y}^2])^2\} \\
&= Var[y]+Var[\hat{y}]+\{(E[y^2])^2-2E[y]E[\hat{y}]+(E[\hat{y}^2])^2\} \\
&= Var[y]+Var[\hat{y}]+(E[y]-E[\hat{y}])^2 \\
&= noise+variance+bias^2
\end{aligned}
$$

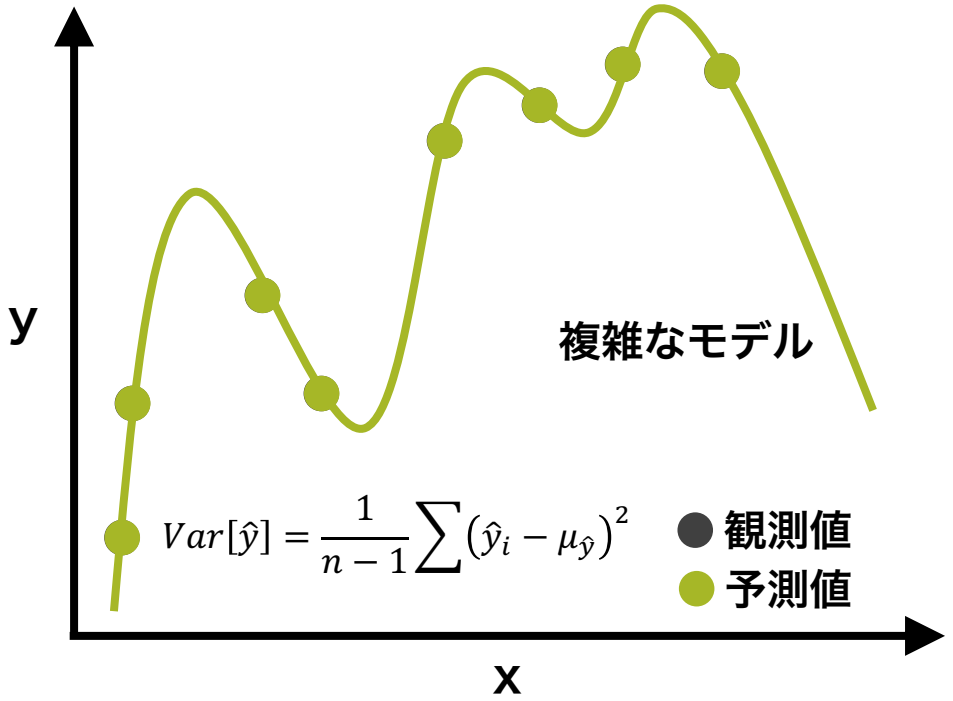
バリアンスを小さくすれば $E[(y-\hat{y})^2]$ が小さくなる。極端な例を挙げれば、モデルのパラメータをなくし単純化させて、 y = 1.0 とすれば、バリアンス $Var[\hat{y}]$ が 0 になる。しかし、このときバイアス $(E[y]-E[\hat{y}])^2$ が大きくなる。



Bias-Variance Tradeoff

$$
\begin{aligned}
& E[(y-\hat{y})^2] \\
&= E[y^2+\hat{y}^2-2y\hat{y}] \\
&= E[y^2]+E[\hat{y}^2]-2E[y\hat{y}] \\
&= \{Var[y]+(E[y^2])^2\}+\{Var[\hat{y}]+(E[\hat{y}^2])^2\}-2E[y\hat{y}] \\
&= Var[y]+Var[\hat{y}]+\{(E[y^2])^2-2E[y\hat{y}]+(E[\hat{y}^2])^2\} \\
&= Var[y]+Var[\hat{y}]+\{(E[y^2])^2-2E[y]E[\hat{y}]+(E[\hat{y}^2])^2\} \\
&= Var[y]+Var[\hat{y}]+(E[y]-E[\hat{y}])^2 \\
&= noise+variance+bias^2
\end{aligned}
$$

バイアスを小さくすれば $E[(y-\hat{y})^2]$ が小さくなる。極端な例を挙げれば、モデルのパラメータを増やし複雑かさせれば過学習になり、バイアスが 0 になる。しかし、このとき、バリアンス $Var[\hat{y}]$ が大きくなる。



バイアスとバリアンス

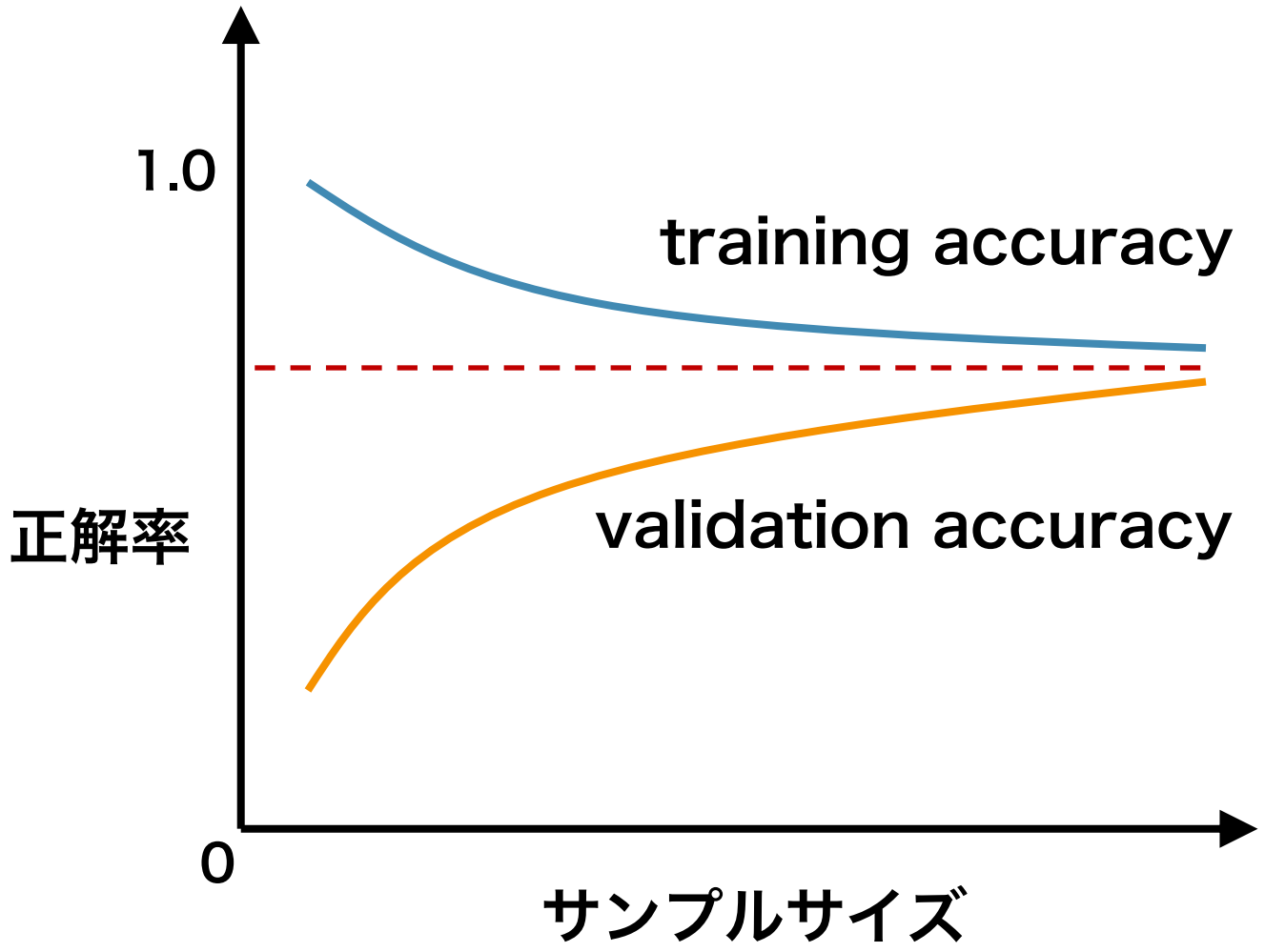
- バイアス・バリアンスのトレードオフ
- 学習曲線
- 検証曲線

学習曲線

訓練データの複雑さに対して、パラメーターの少ない単純なモデルを適用すると学習不足になり、パラメーターの多い複雑なモデルを適用すると過学習に陥る。学習不足・過学習のない汎用性のあるモデルを作成するには、モデルの評価を正確に行う必要がある。

構築したモデルが、学習不足あるいは過学習の問題を持つかどうかを調べるために、学習曲線と検証曲線が使われる。

学習曲線は、横軸にサンプルサイズ、縦軸に正解率をプロットしたグラフである。全データセットに対してダウンサンプリングを行い、各サンプルサイズにおける予測時の正解率を図示したグラフである。

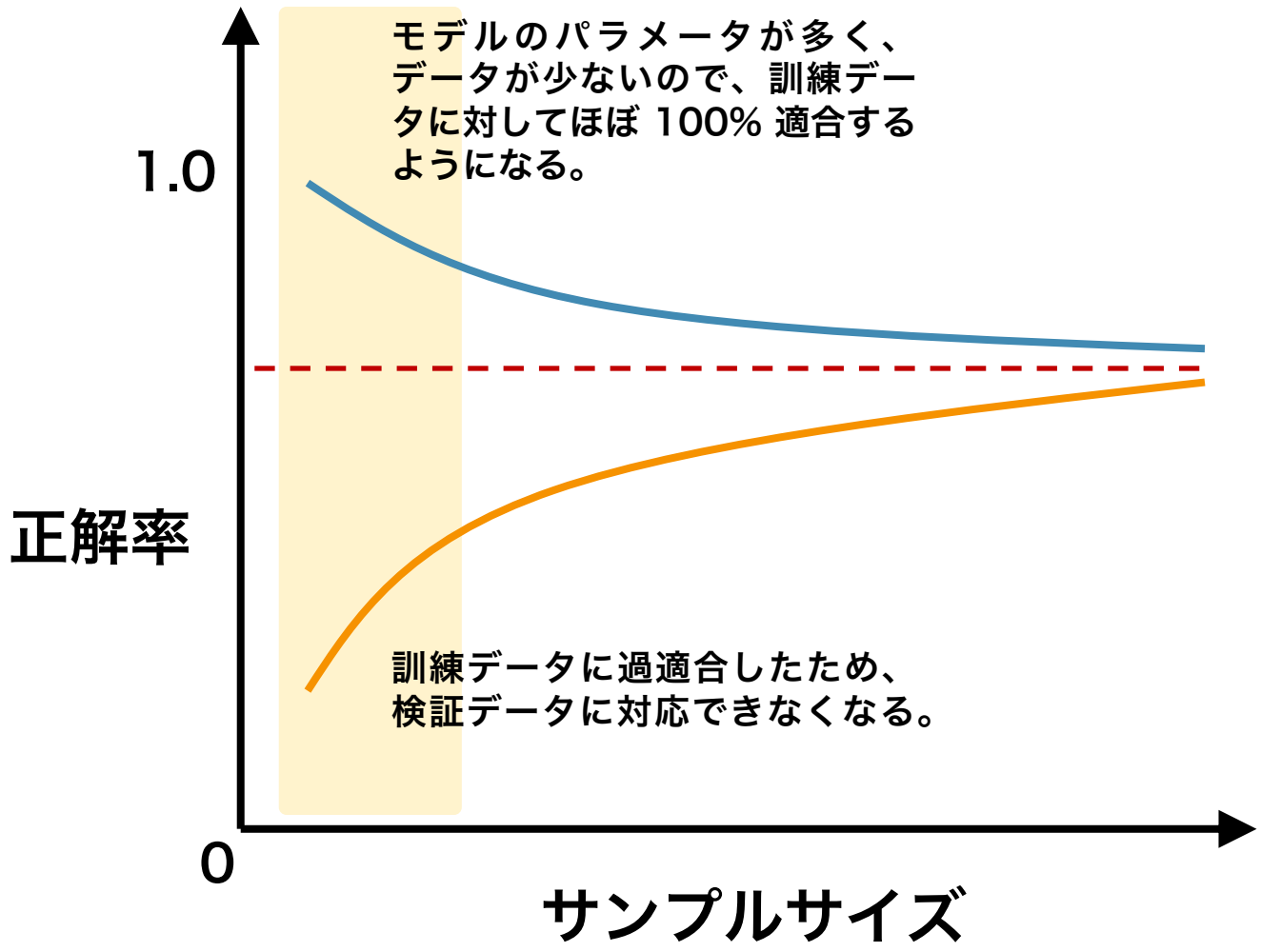


学習曲線

訓練データの複雑さに対して、パラメーターの少ない単純なモデルを適用すると学習不足になり、パラメーターの多い複雑なモデルを適用すると過学習に陥る。学習不足・過学習のない汎用性のあるモデルを作成するには、モデルの評価を正確に行う必要がある。

構築したモデルが、学習不足あるいは過学習の問題を持つかどうかを調べるために、学習曲線と検証曲線が使われる。

学習曲線は、横軸にサンプルサイズ、縦軸に正解率をプロットしたグラフである。全データセットに対してダウンサンプリングを行い、各サンプルサイズにおける予測時の正解率を図示したグラフである。

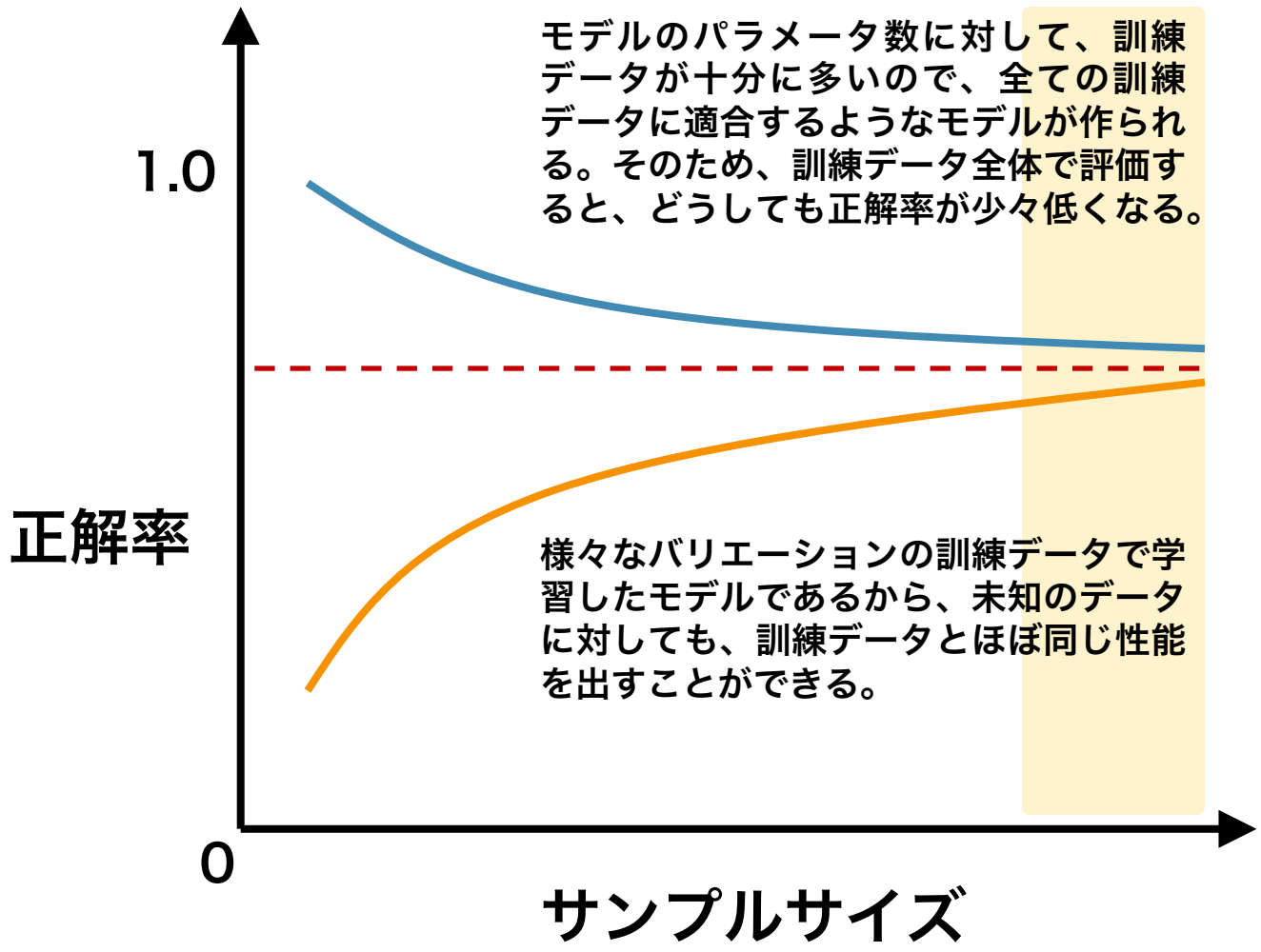


学習曲線

訓練データの複雑さに対して、パラメーターの少ない単純なモデルを適用すると学習不足になり、パラメーターの多い複雑なモデルを適用すると過学習に陥る。学習不足・過学習のない汎用性のあるモデルを作成するには、モデルの評価を正確に行う必要がある。

構築したモデルが、学習不足あるいは過学習の問題を持つかどうかを調べるために、学習曲線と検証曲線が使われる。

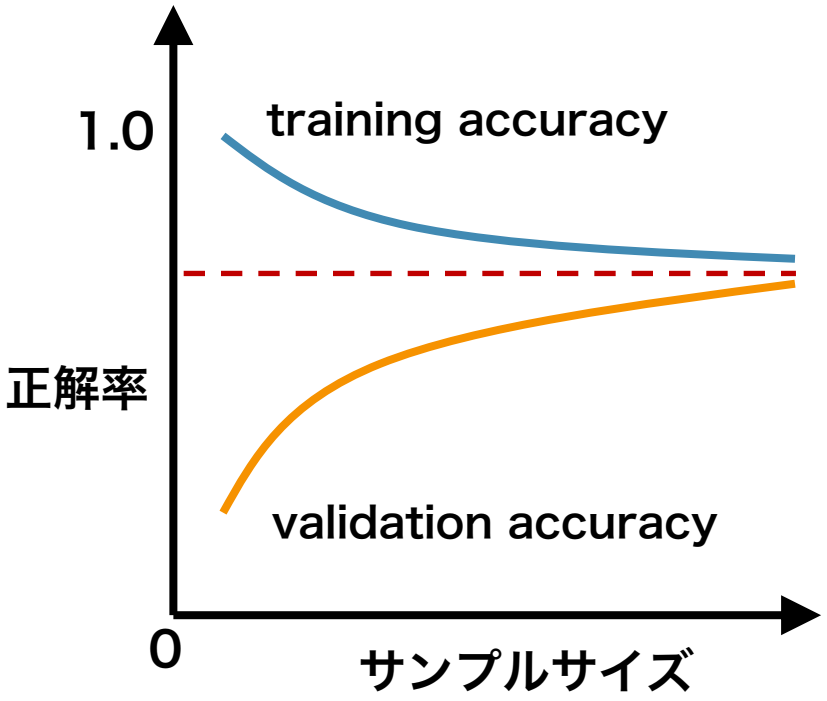
学習曲線は、横軸にサンプルサイズ、縦軸に正解率をプロットしたグラフである。全データセットに対してダウンサンプリングを行い、各サンプルサイズにおける予測時の正解率を図示したグラフである。



学習曲線

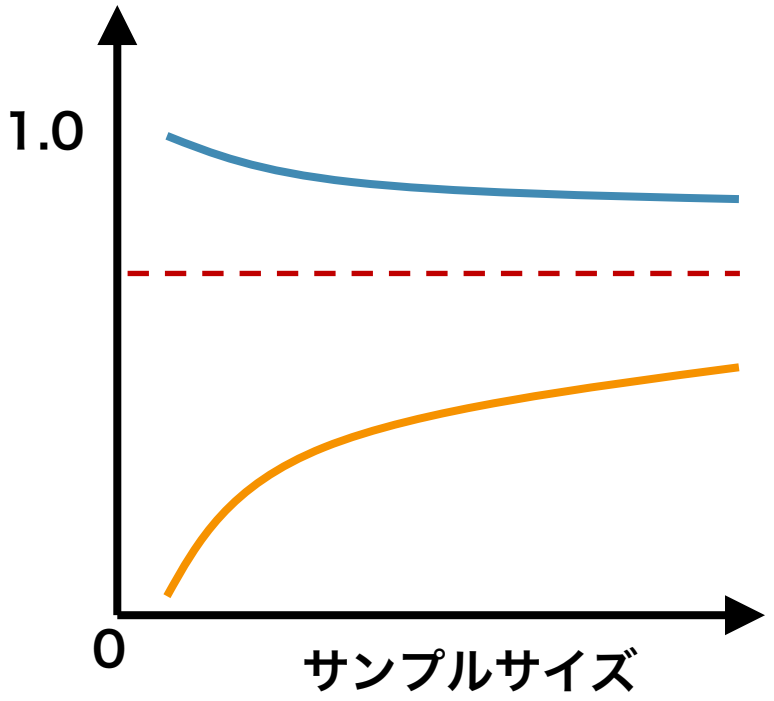
最適なモデル

訓練データと未知のデータの両方に対してほぼ同等な性能を発揮できるバランスのとれたモデル。



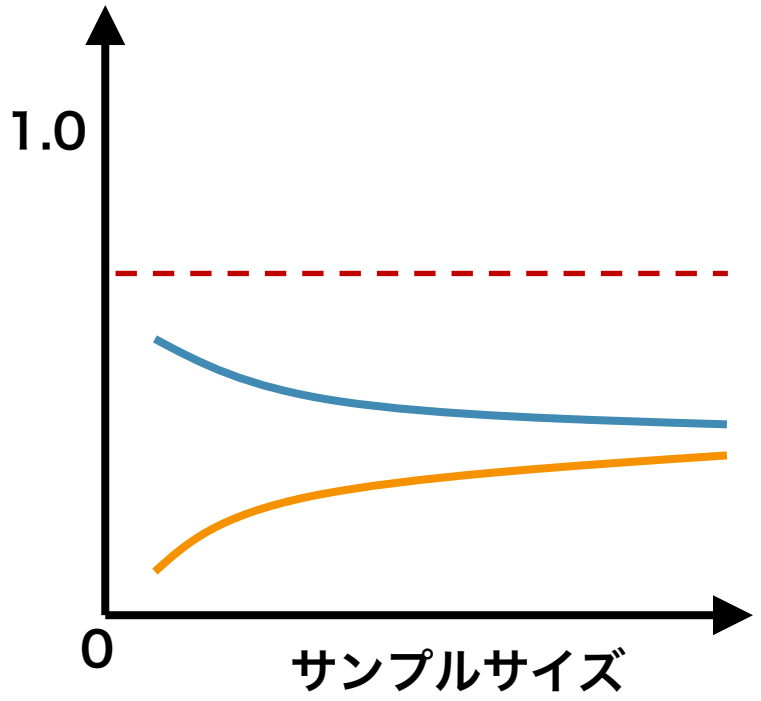
バリアンスの高いモデル

訓練データに対して過適合を起こし、未知のデータに対応できないような過学習を起こしているモデル。



バイアスの高いモデル

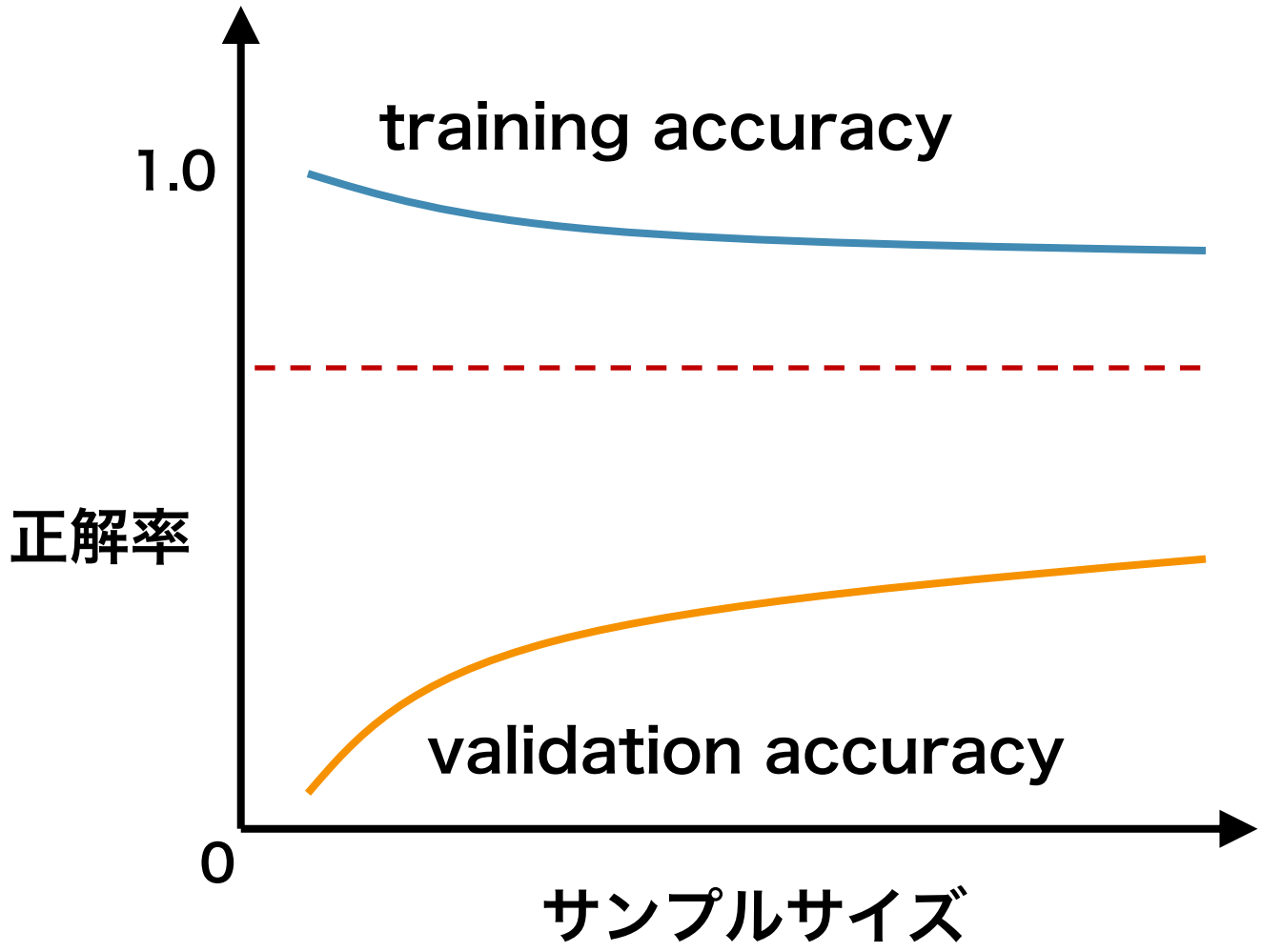
訓練データと未知のデータの両方に対応できないような学習不足を起こしているモデル。



学習曲線

バリアンスの大きいモデルは、訓練データに過剰適合したためと考えられる。そのため、バリアンスを小さくするためには、次のような対処方法がある。

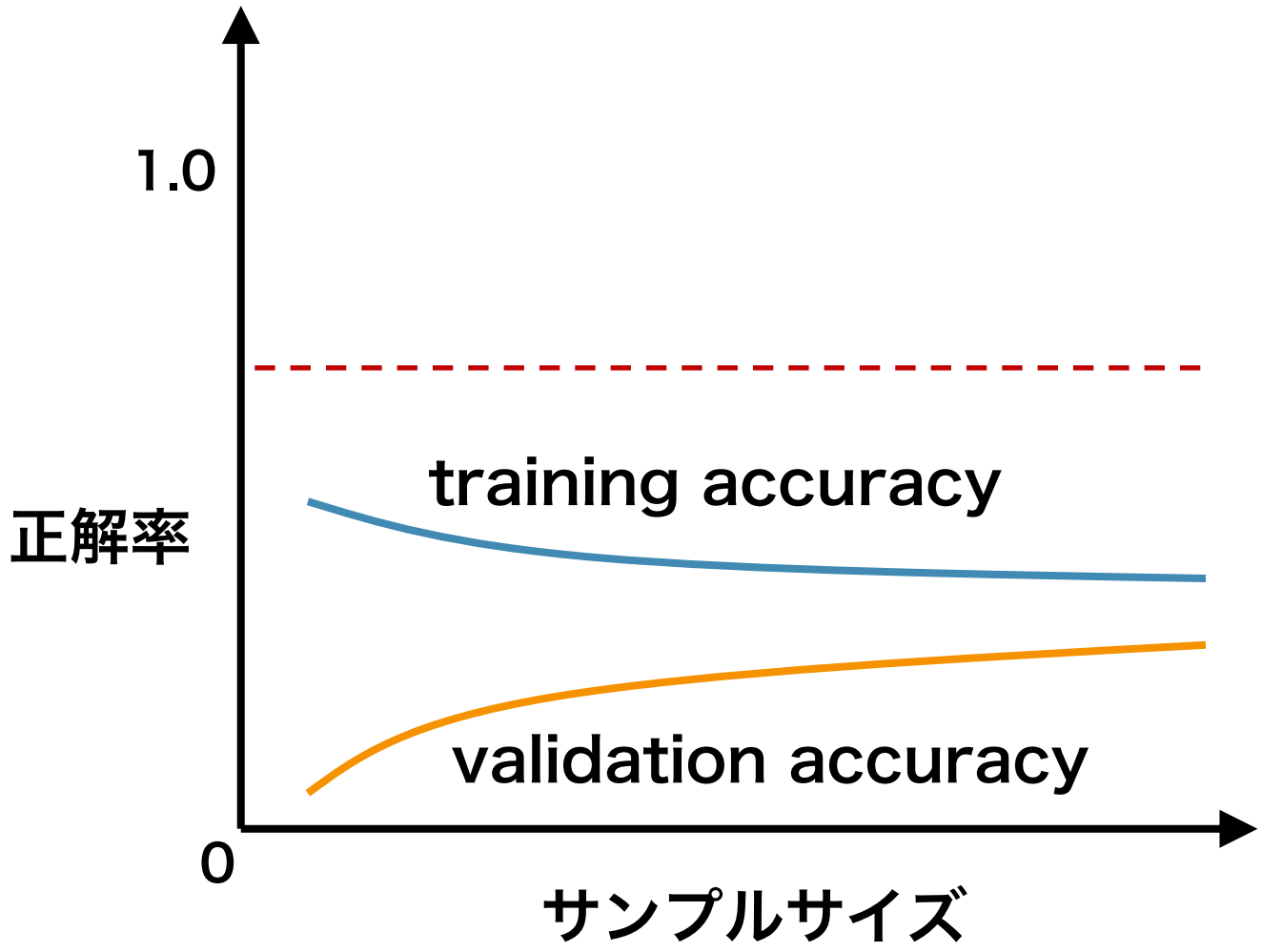
- モデルの複雑さを減らす。
    - 次元圧縮・特徴抽出などにより特徴量の次元数を減らす。
    - 正則化の強さを強くし、パラメータ数を減らす。
- 訓練データがすべての場合をカバーできていない場合は、訓練データを増やす。



学習曲線

バイアスの大きいモデルは、訓練データに対しても性能が低いことから、学習不足と考えられる。そのため、バイアスを小さくするためには、次のような対処方法がある。

- モデルの複雑さを増やす。
    - 新たな特徴量を作成または収集し、特徴量の数を増やす。
    - 正則化の強さを弱くし、パラメータ数の削減を抑える。



バイアスとバリアンス

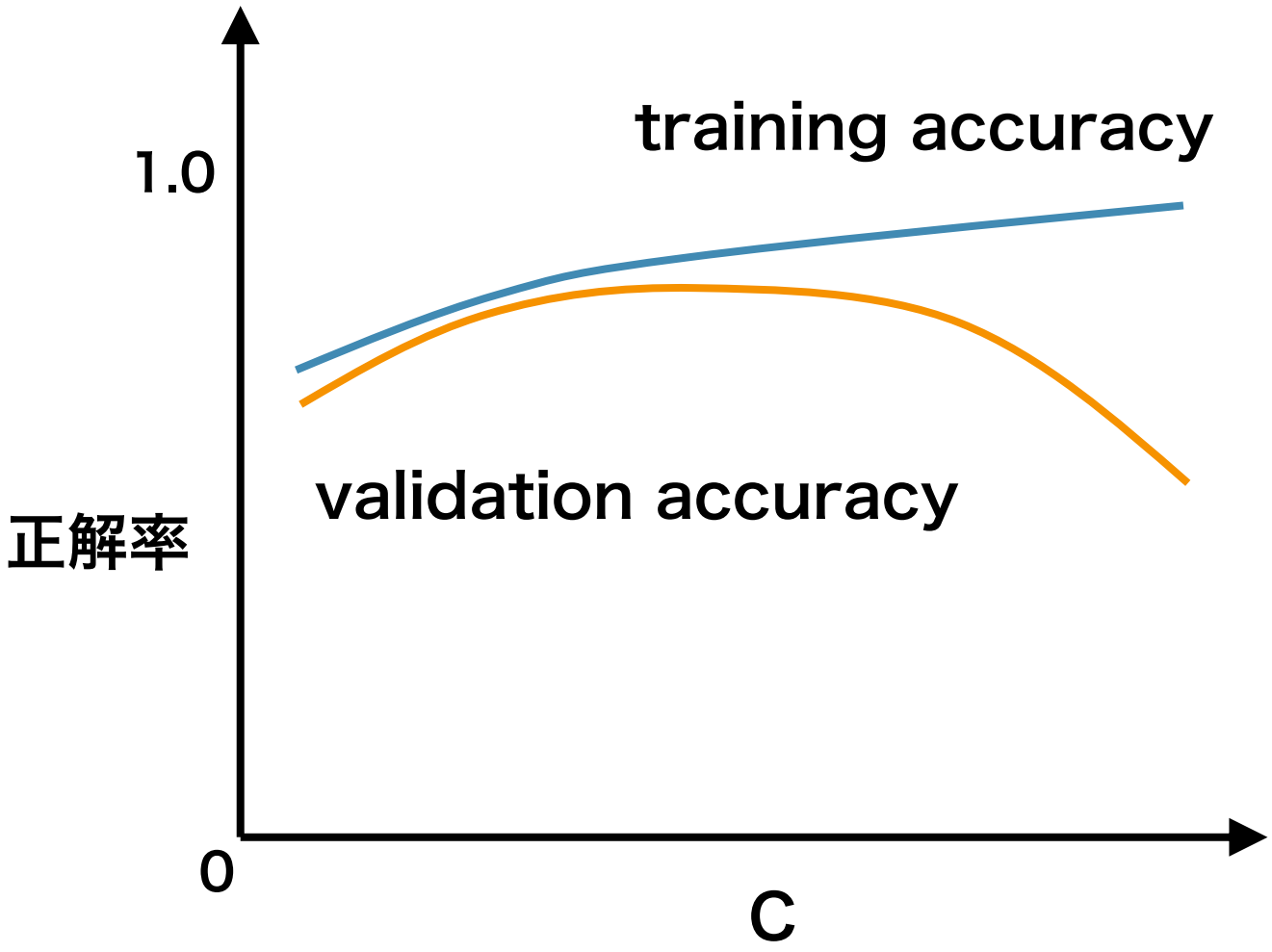
- バイアス・バリアンスのトレードオフ
- 学習曲線
- 検証曲線

検証曲線

訓練データの複雑さに対して、パラメーターの少ない単純なモデルを適用すると学習不足になり、パラメーターの多い複雑なモデルを適用すると過学習に陥る。学習不足・過学習のない汎用性のあるモデルを作成するには、モデルの評価を正確に行う必要がある。

構築したモデルが、学習不足あるいは過学習の問題を持つかどうかを調べるために、学習曲線と検証曲線が使われる。

検証曲線は、横軸にハイパーパラメータの値を表し、縦軸は、そのときの予測正解率を表した曲線である。

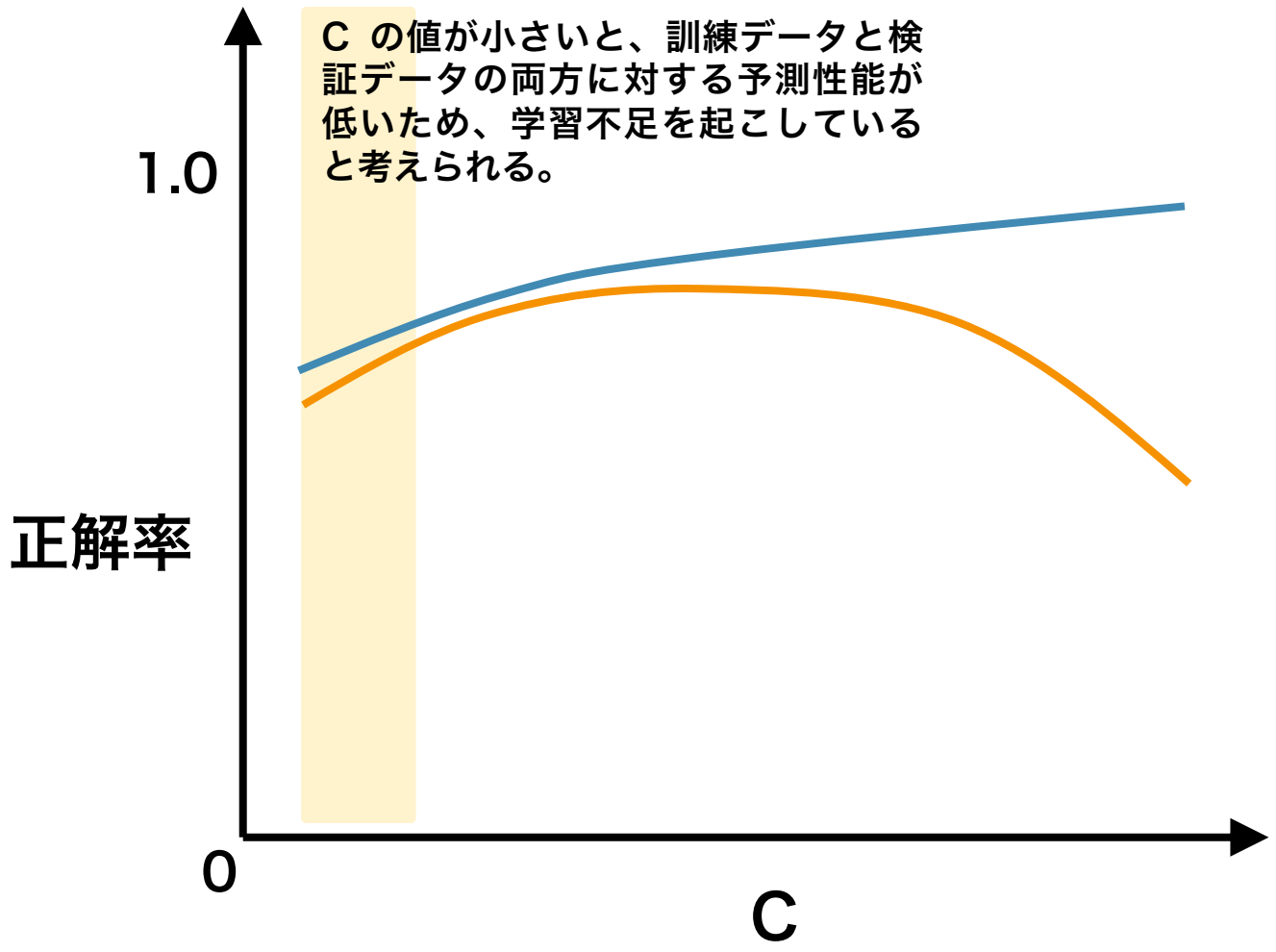


検証曲線

訓練データの複雑さに対して、パラメーターの少ない単純なモデルを適用すると学習不足になり、パラメーターの多い複雑なモデルを適用すると過学習に陥る。学習不足・過学習のない汎用性のあるモデルを作成するには、モデルの評価を正確に行う必要がある。

構築したモデルが、学習不足あるいは過学習の問題を持つかどうかを調べるために、学習曲線と検証曲線が使われる。

検証曲線は、横軸にハイパーパラメータの値を表し、縦軸は、そのときの予測正解率を表した曲線である。

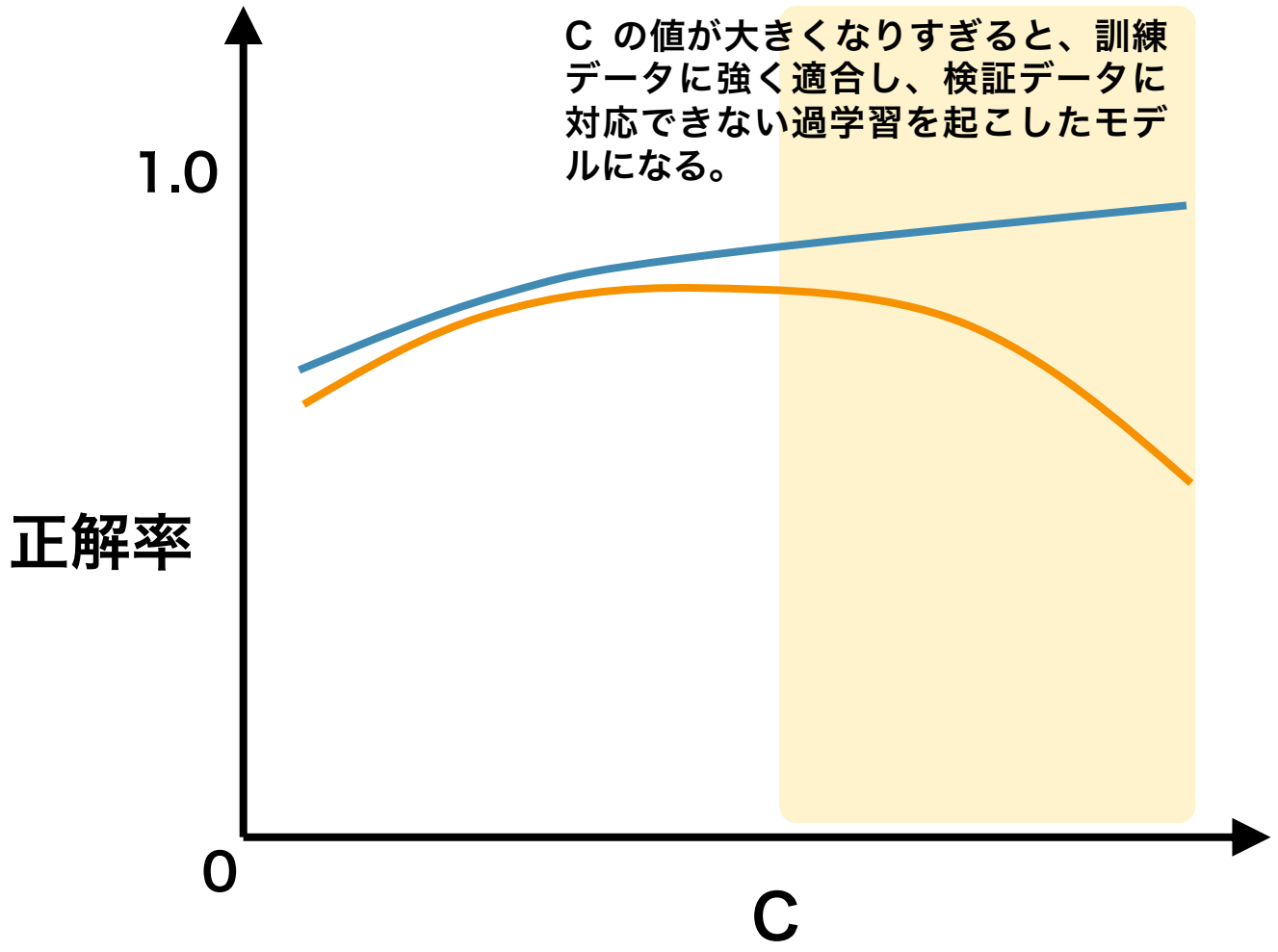


検証曲線

訓練データの複雑さに対して、パラメーターの少ない単純なモデルを適用すると学習不足になり、パラメーターの多い複雑なモデルを適用すると過学習に陥る。学習不足・過学習のない汎用性のあるモデルを作成するには、モデルの評価を正確に行う必要がある。

構築したモデルが、学習不足あるいは過学習の問題を持つかどうかを調べるために、学習曲線と検証曲線が使われる。

検証曲線は、横軸にハイパーパラメータの値を表し、縦軸は、そのときの予測正解率を表した曲線である。



モデル評価

- パラメーターチューニング
- 評価データセット
- バイアスとバリアンス
- 評価指標

評価指標

- 分類問題
- 回帰問題

評価指標

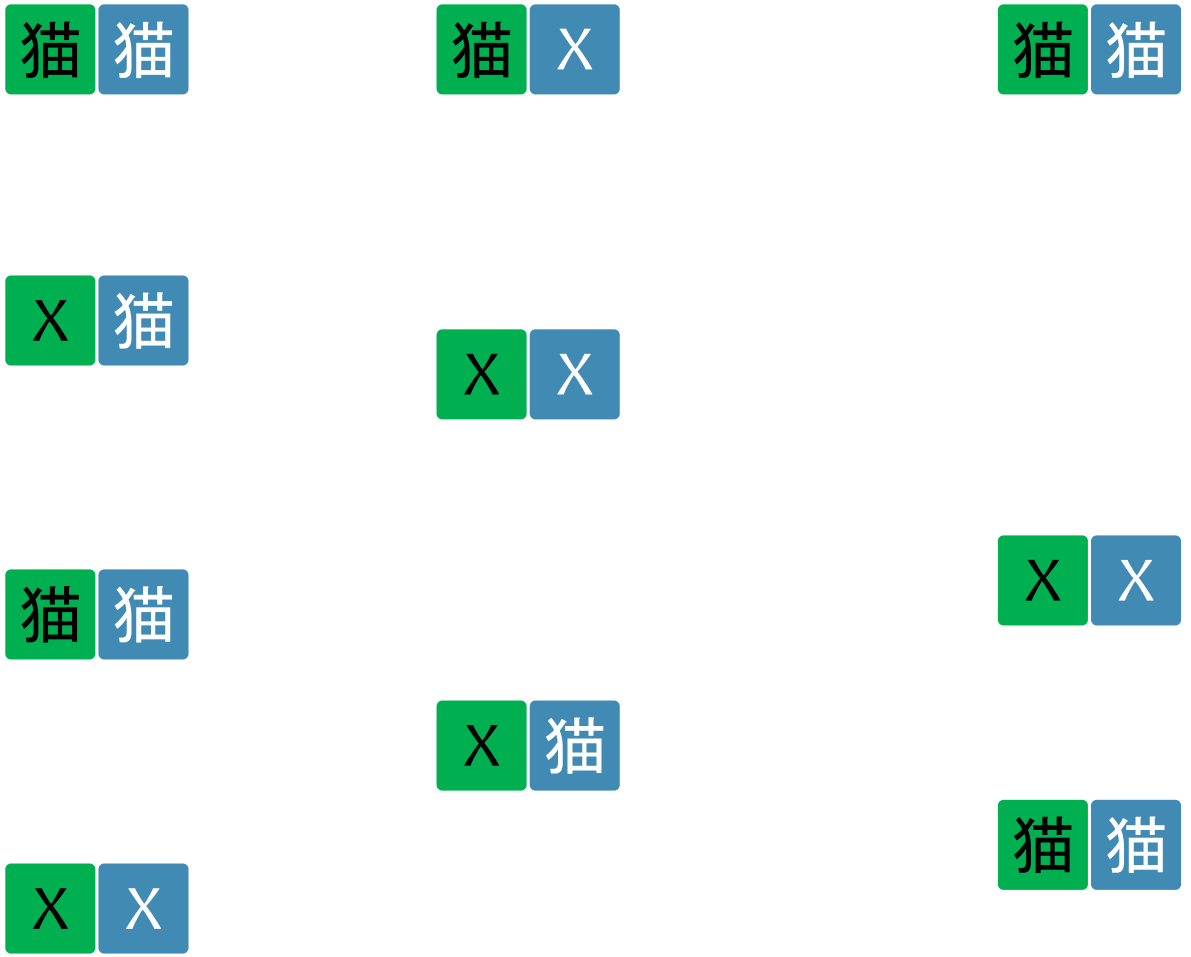
- 分類問題
- 回帰問題

混同行列

分類問題において、真のラベルと予測ラベルは、次の混同行列に示した、TP、FN、FP、およびTN の 4 通りの組み合わせを取ることができる。分類の性能を評価するために、この混同行列から適合率、再現率などを計算して利用する。

混同行列

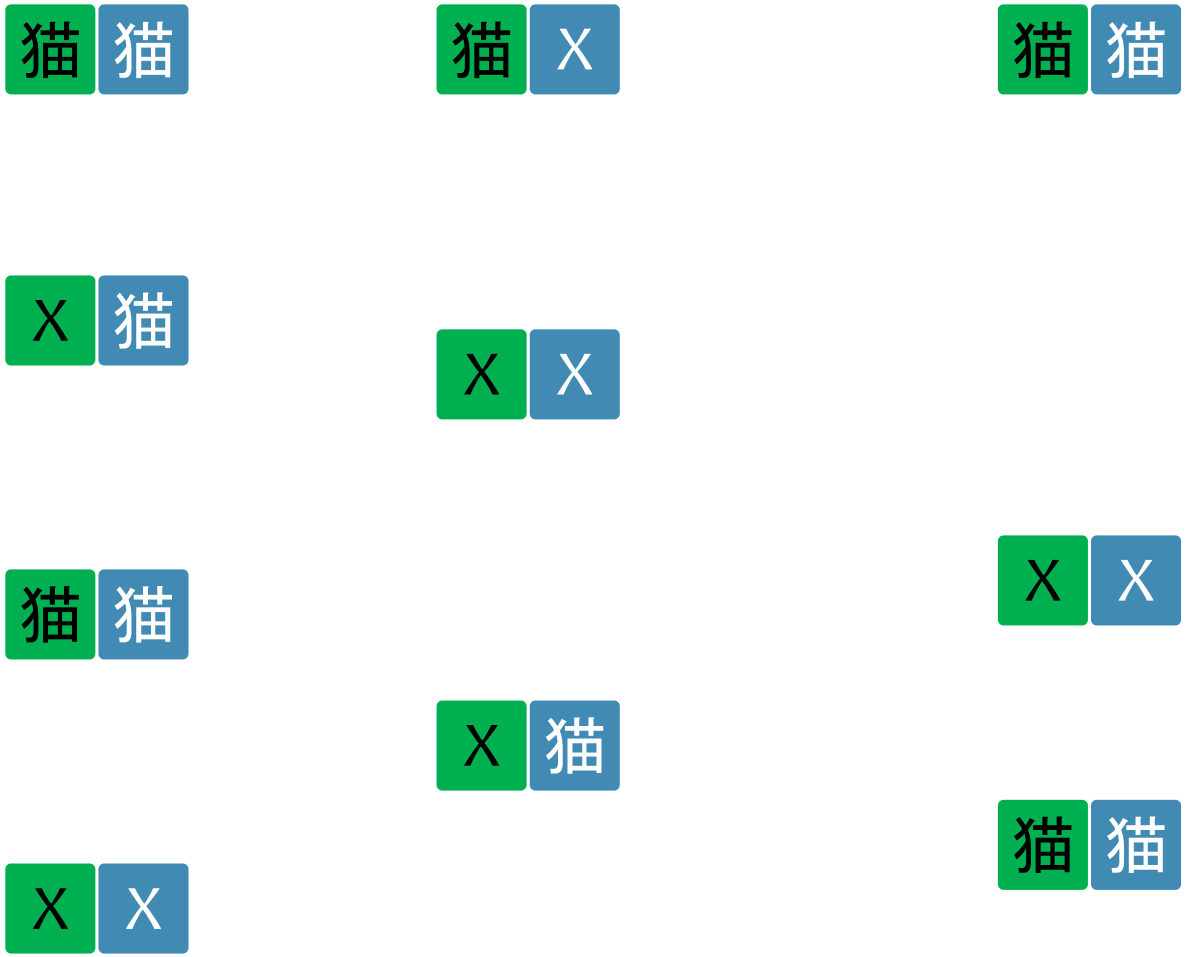
		予測ラベル	
		positive	negative
ラベル	positive	TP	FN
	negative	FP	TN



混同行列

分類問題において、真のラベルと予測ラベルは、次の混同行列に示した、TP、FN、FP、および TN の 4 通りの組み合わせを取ることができる。分類の性能を評価するために、この混同行列から適合率、再現率などを計算して利用する。

混同行列		予測ラベル	
		positive	negative
ラベル	positive	TP = 4	FN
	negative	FP	TN

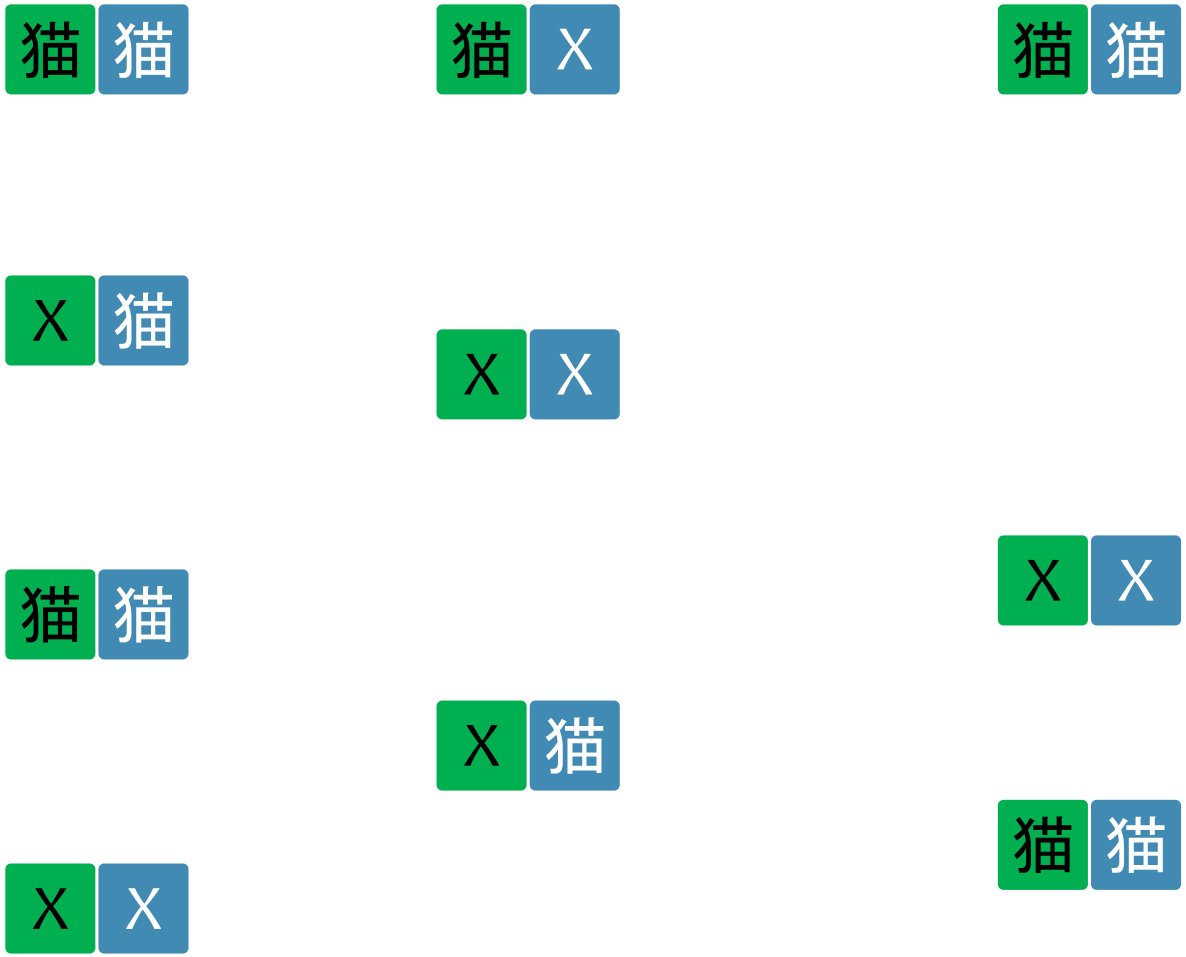


混同行列

分類問題において、真のラベルと予測ラベルは、次の混同行列に示した、TP、FN、FP、および TN の 4 通りの組み合わせを取ることができる。分類の性能を評価するために、この混同行列から適合率、再現率などを計算して利用する。

混同行列

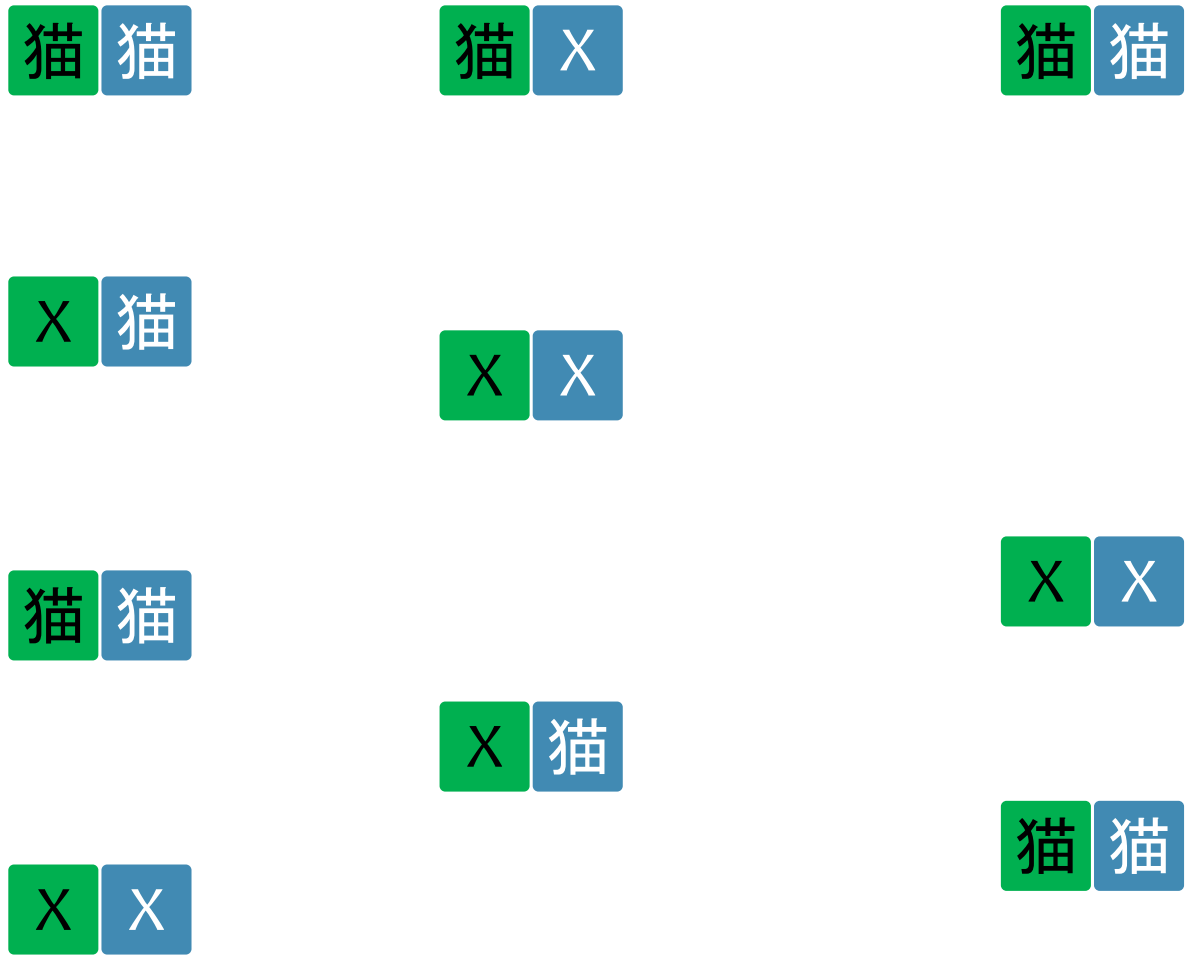
		予測ラベル	
		positive	negative
ラベル	positive	TP = 4	FN = 1
	negative	FP	TN



混同行列

分類問題において、真のラベルと予測ラベルは、次の混同行列に示した、TP、FN、FP、および TN の 4 通りの組み合わせを取ることができる。分類の性能を評価するために、この混同行列から適合率、再現率などを計算して利用する。

混同行列		予測ラベル	
		positive	negative
ラベル	positive	TP = 4	FN = 1
	negative	FP = 2	TN

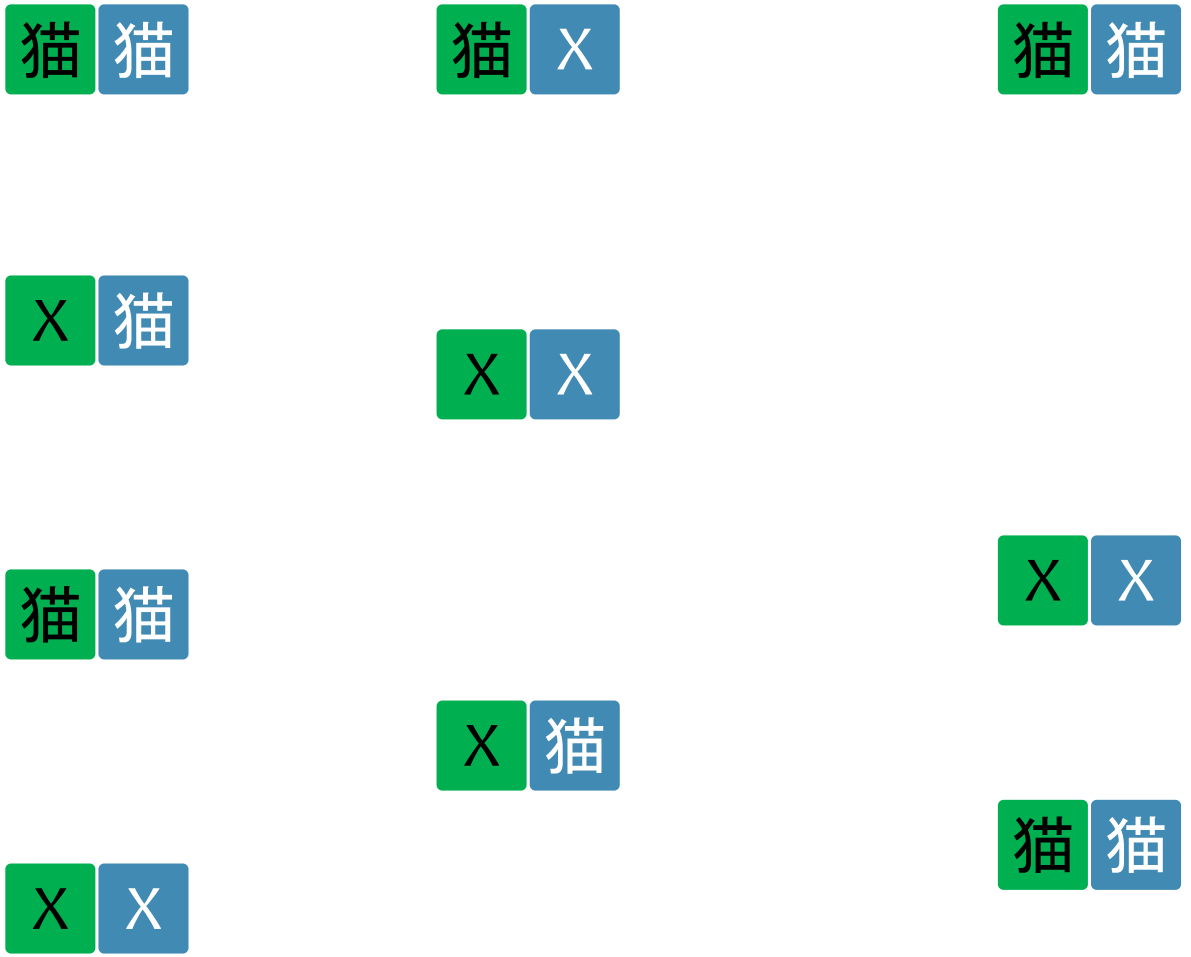


混同行列

分類問題において、真のラベルと予測ラベルは、次の混同行列に示した、TP、FN、FP、および TN の 4 通りの組み合わせを取ることができる。分類の性能を評価するために、この混同行列から適合率、再現率などを計算して利用する。

混同行列

		予測ラベル	
		positive	negative
ラベル	positive	TP = 4	FN = 1
	negative	FP = 2	TN = 3



評価指標

分類問題において、真のラベルと予測ラベルは、次の混同行列に示した、TP、FN、FP、およびTN の 4 通りの組み合わせを取ることができる。分類の性能を評価するために、この混同行列から適合率、再現率などを計算して利用する。

混同行列

		予測ラベル	
		positive	negative
ラベル	positive	TP	FN
	negative	FP	TN

誤分類率

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

適合率

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

正解率

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

再現率（真陽性率）

$$REC = TPR = \frac{TP}{TP + FN}$$

偽陽性率

$$FPR = \frac{FP}{FP + TN}$$

F1 値

$$F1 = 2\frac{PRE \cdot REC}{PRE + REC}$$

確認問題

- 癌の早期発見は生存率の向上に役立つ。癌検査において、血液成分から癌になっているか否かをできるだけ正確に予測したい。この予測モデルを構築するためには、どの指標が適しているのか。

- スパムメールは仕事に支障をきたす。メール本文のキーワードを使って、そのメールがスパムメールかどうかを予測するモデルを構築するためには、どの指標が適しているのか。

混同行列

		予測ラベル	
		positive	negative
ラベル	positive	TP	FN
	negative	FP	TN

誤分類率

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

適合率

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

正解率

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

再現率（真陽性率）

$$REC = TPR = \frac{TP}{TP + FN}$$

偽陽性率

$$FPR = \frac{FP}{FP + TN}$$

F1 値

$$F1 = 2\frac{PRE \cdot REC}{PRE + REC}$$

確認問題 – 解答

- 癌の早期発見は生存率の向上に役立つ。癌検査において、血液成分から癌になっているか否かをできるだけ正確に予測したい。この予測モデルを構築するためには、どの指標が適しているのか。
  - 年齢が20代前後のデータの場合、癌患者が健常者に比べて著しく少ない。そのため、FP および TN が分母となっている指標はモデルを正確に評価できない可能性がある。よって、この場合は、再現率が適している。
  - がん患者を見落としたくないので、誤検出（偽陽性）が多くてもいい。よって、この場合も再現率を利用する。

再現率（真陽性率）

$$REC = TPR = \frac{TP}{TP + FN}$$

適合率

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

正解率

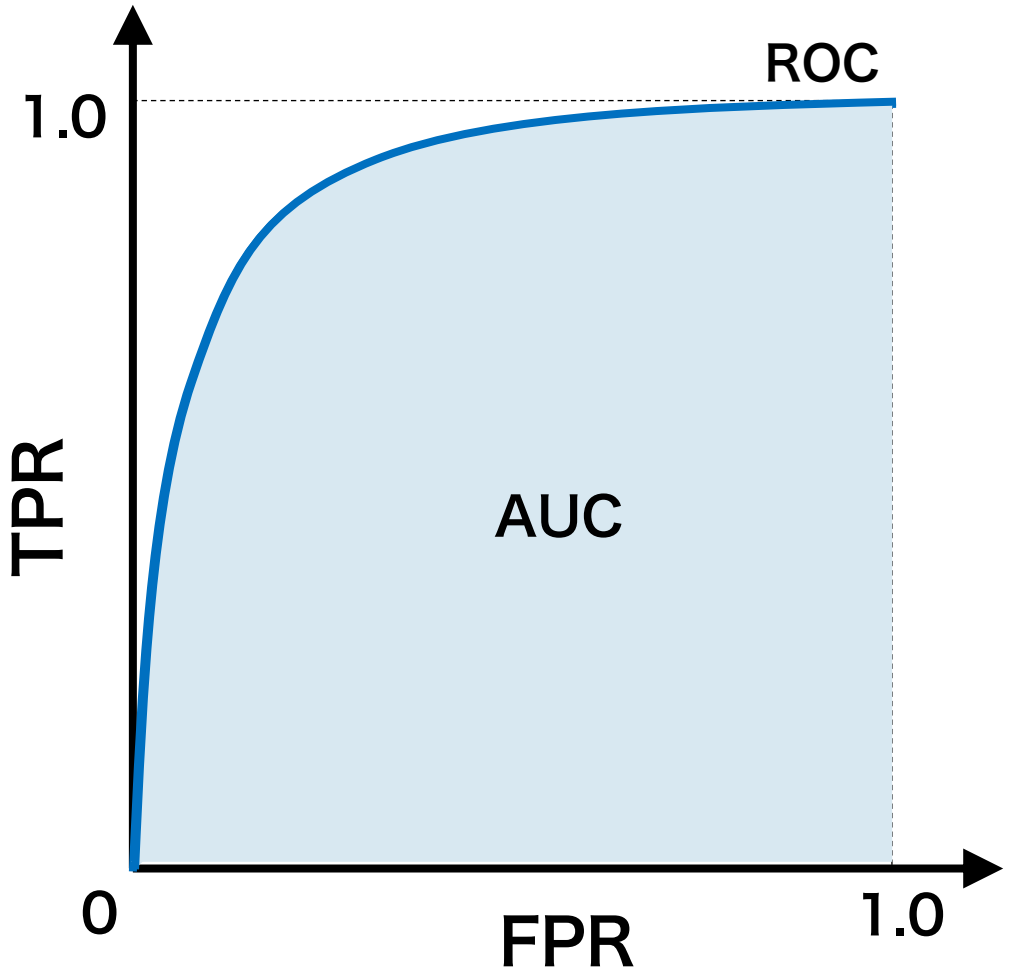
$$ACC = \frac{TP + TN}{TP + FN + FP + TN}$$

混同行列

		予測ラベル	
		positive	negative
ラベル	positive	TP	FN
	negative	FP	TN

評価指標（ROC 曲線）

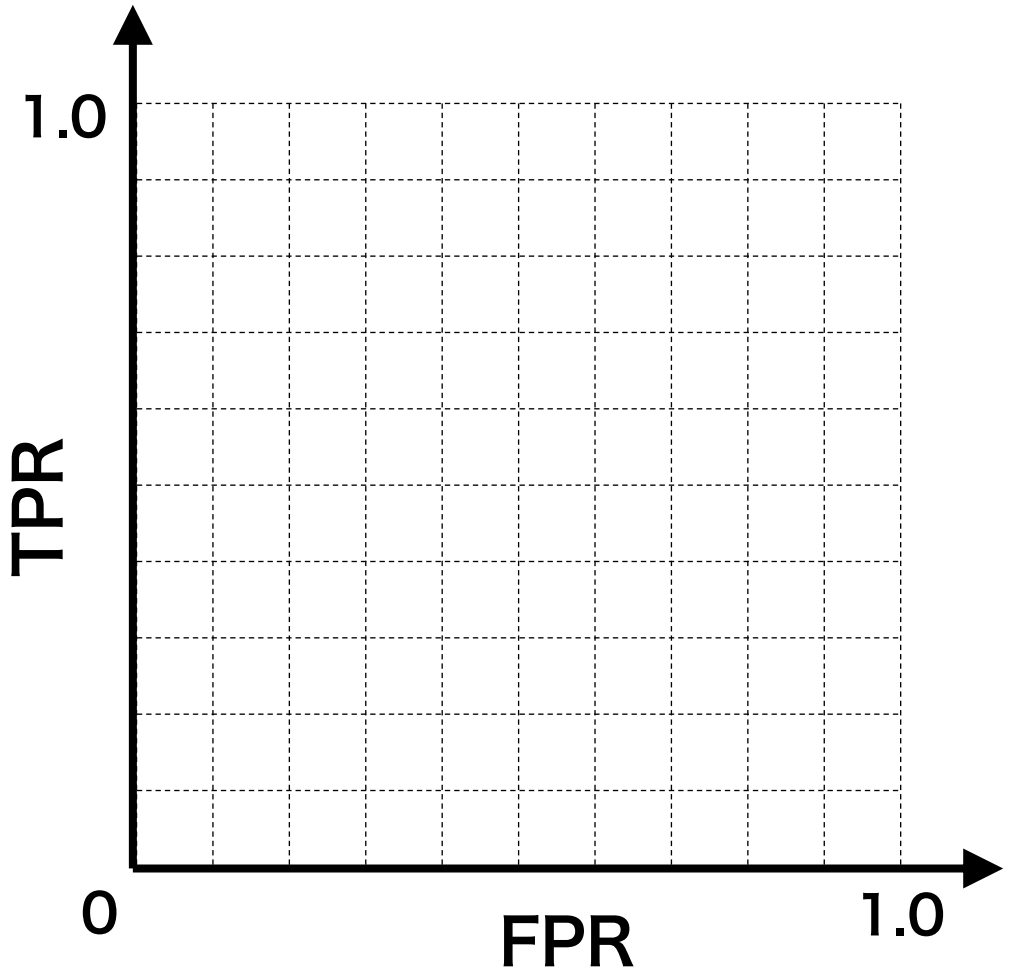
ROC 曲線は、positive と判断するスコアの閾値を動かして、TPR と FPR を計算し、縦軸に TPR、横軸に FPR を取ったときに描かれる線グラフである。ROC 曲線の下の部分の面積を AUC として計算することができ、AUC が 1 に近づくほどモデルの性能が良いとされる。



評価指標（ROC 曲線）

スコア	ラベル	判定	評価
0.984	猫		
0.933	猫		
0.870	X		
0.867	猫		
0.702	猫		
0.540	X		
0.438	X		
0.419	猫		
0.331	X		
0.252	X		

予測モデルが出力するスコアに基づいて、スコアの高い順にデータを並べ替える。次に、スコアに対して閾値を設けて、閾値を超えた場合に positive、閾値以下の場合に negative と判定する。判定結果と教師ラベルを比較し TPR および FPR を計算する。

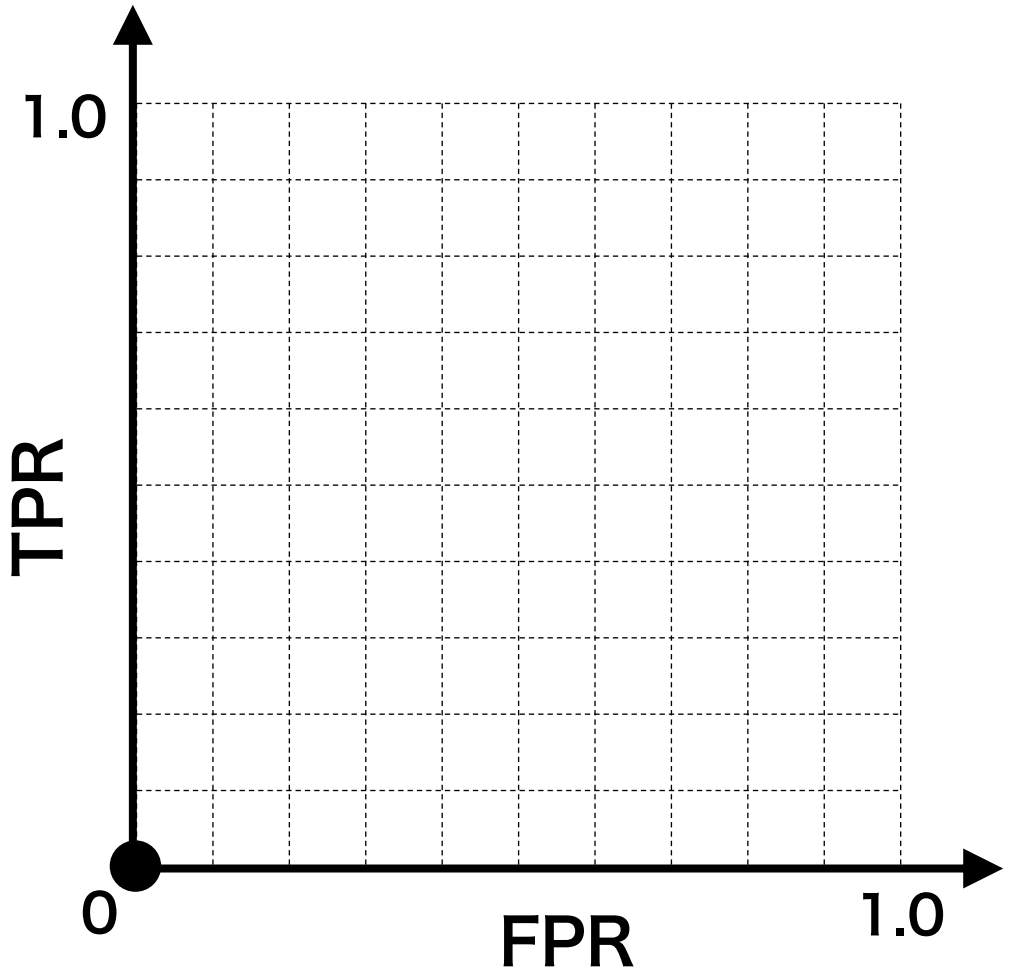


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	X	FN
0.933	猫	X	FN
0.870	X	X	TN
0.867	猫	X	FN
0.702	猫	X	FN
0.540	X	X	TN
0.438	X	X	TN
0.419	猫	X	FN
0.331	X	X	TN
0.252	X	X	TN

閾値

$$TPR = \frac{TP}{TP + FN} = 0 \qquad FPR = \frac{FP}{FP + TN} = 0$$

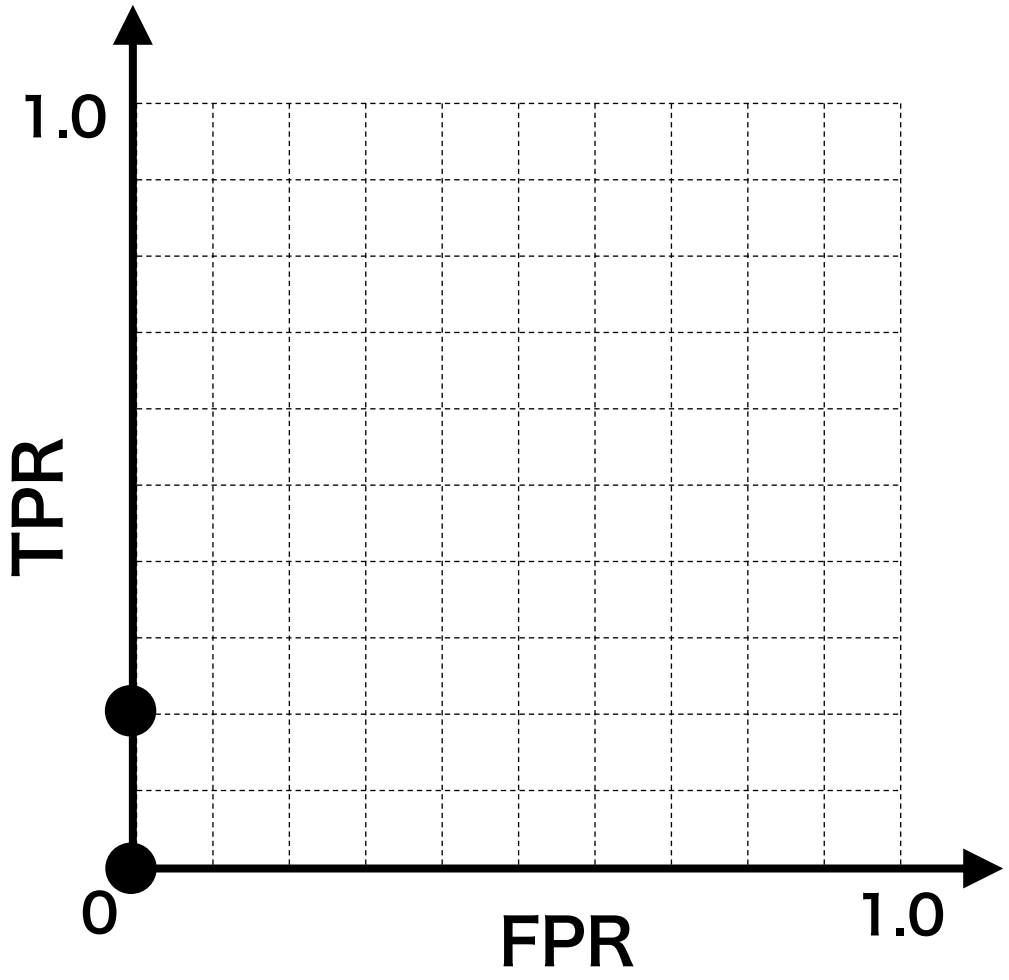


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	X	FN
0.870	X	X	TN
0.867	猫	X	FN
0.702	猫	X	FN
0.540	X	X	TN
0.438	X	X	TN
0.419	猫	X	FN
0.331	X	X	TN
0.252	X	X	TN

閾値

$$TPR = \frac{TP}{TP + FN} = \frac{1}{5} \qquad FPR = \frac{FP}{FP + TN} = 0$$

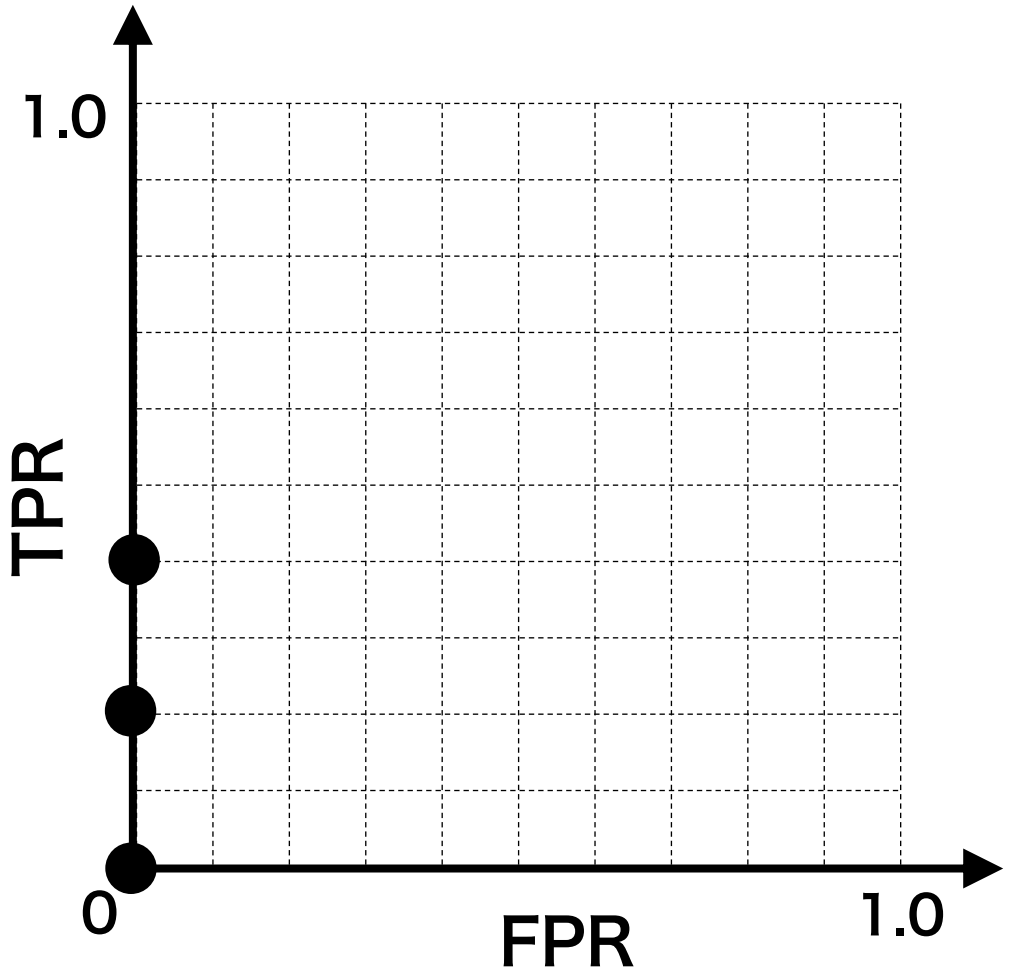


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	X	TN
0.867	猫	X	FN
0.702	猫	X	FN
0.540	X	X	TN
0.438	X	X	TN
0.419	猫	X	FN
0.331	X	X	TN
0.252	X	X	TN

閾値

$$TPR = \frac{TP}{TP + FN} = \frac{2}{5} \qquad FPR = \frac{FP}{FP + TN} = 0$$

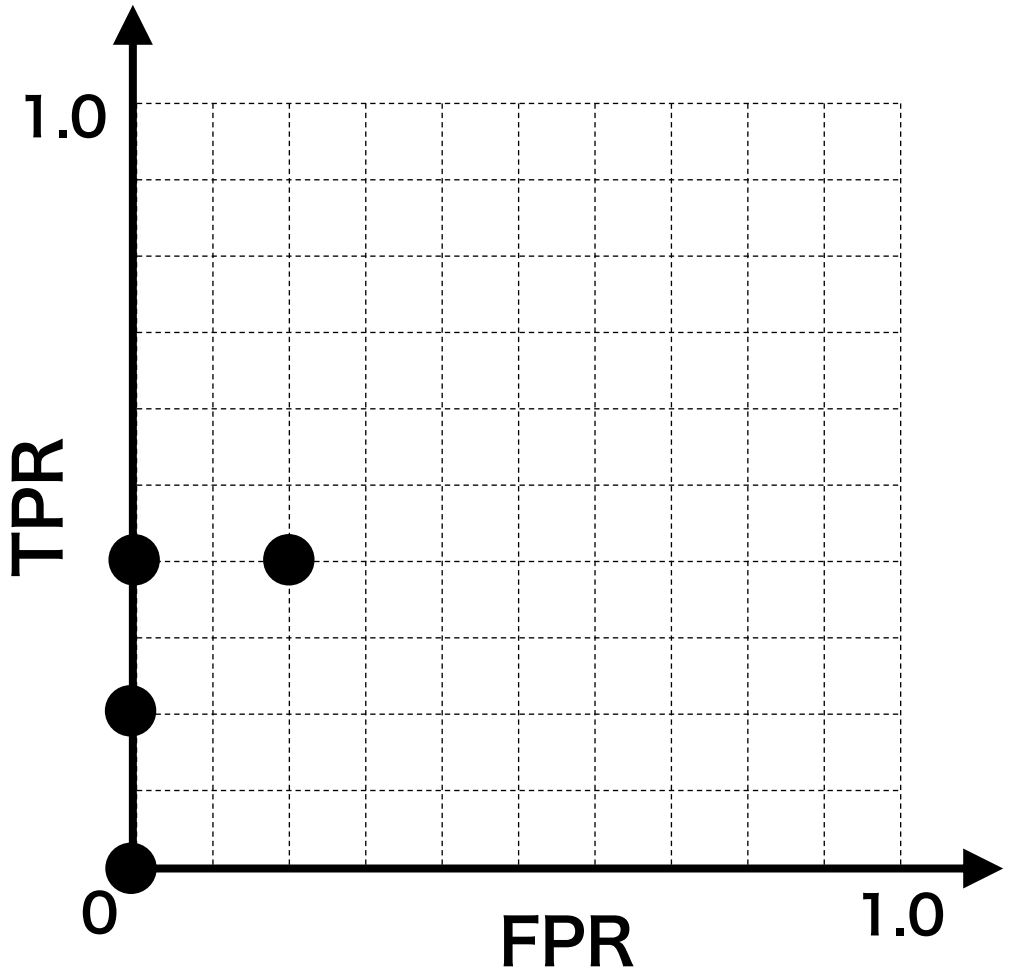


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	X	FN
0.702	猫	X	FN
0.540	X	X	TN
0.438	X	X	TN
0.419	猫	X	FN
0.331	X	X	TN
0.252	X	X	TN

閾値

$$TPR = \frac{TP}{TP + FN} = \frac{2}{5} \qquad FPR = \frac{FP}{FP + TN} = \frac{1}{5}$$

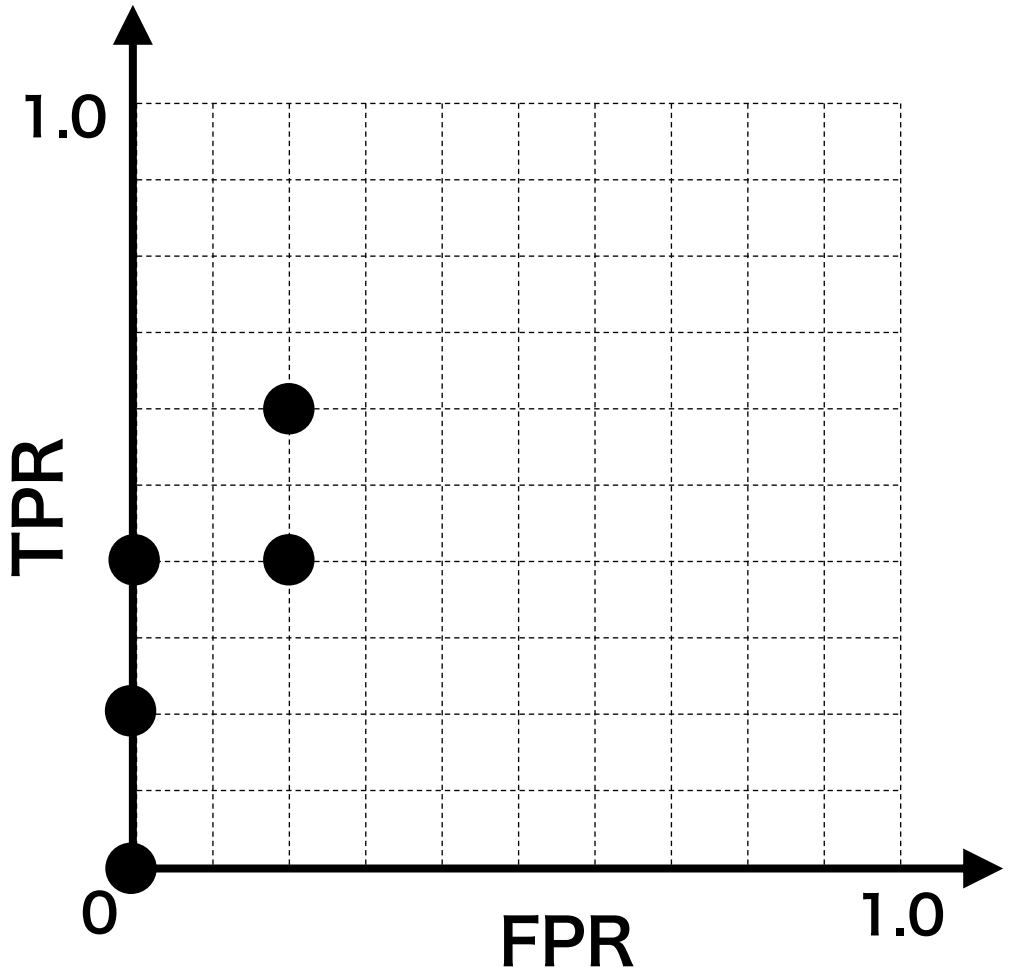


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	猫	TP
0.702	猫	X	FN
0.540	X	X	TN
0.438	X	X	TN
0.419	猫	X	FN
0.331	X	X	TN
0.252	X	X	TN

閾値

$$TPR = \frac{TP}{TP + FN} = \frac{3}{5} \qquad FPR = \frac{FP}{FP + TN} = \frac{1}{5}$$

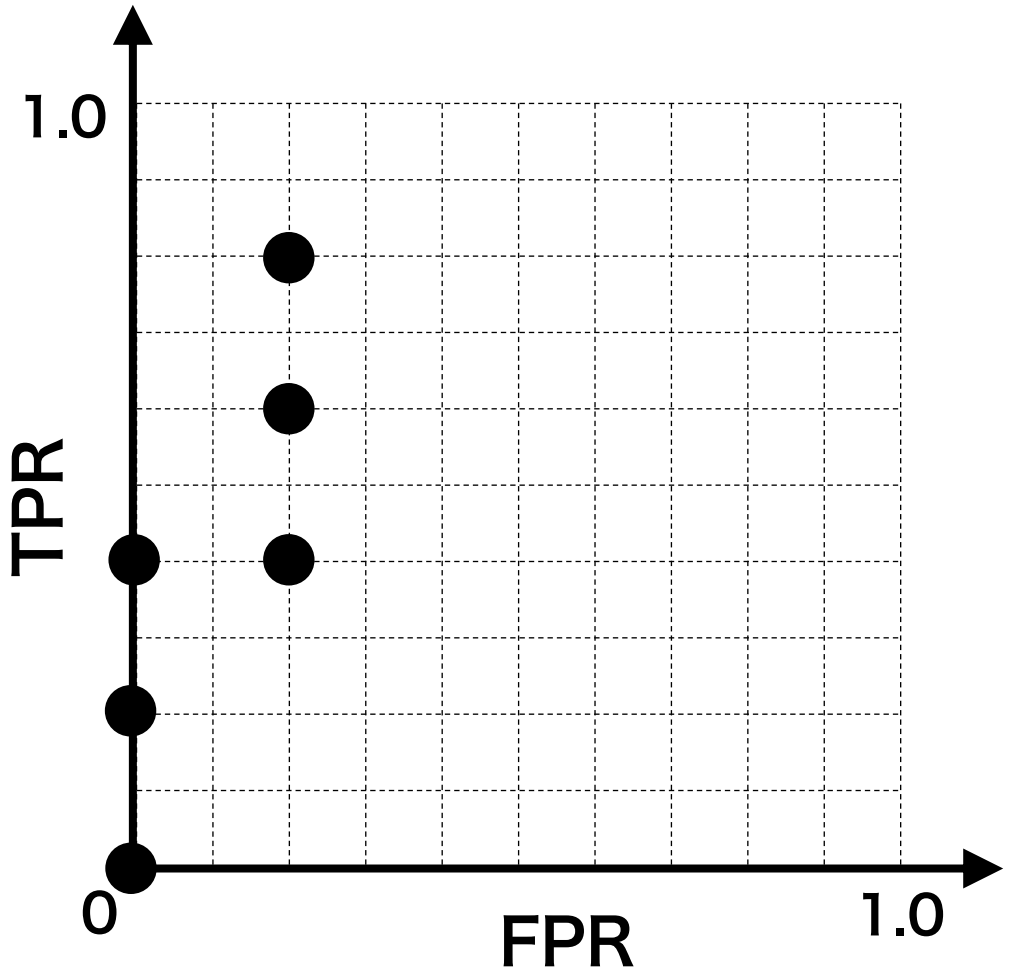


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	猫	TP
0.702	猫	猫	TP
0.540	X	X	TN
0.438	X	X	TN
0.419	猫	X	FN
0.331	X	X	TN
0.252	X	X	TN

閾値

$$TPR = \frac{TP}{TP + FN} = \frac{4}{5} \qquad FPR = \frac{FP}{FP + TN} = \frac{1}{5}$$

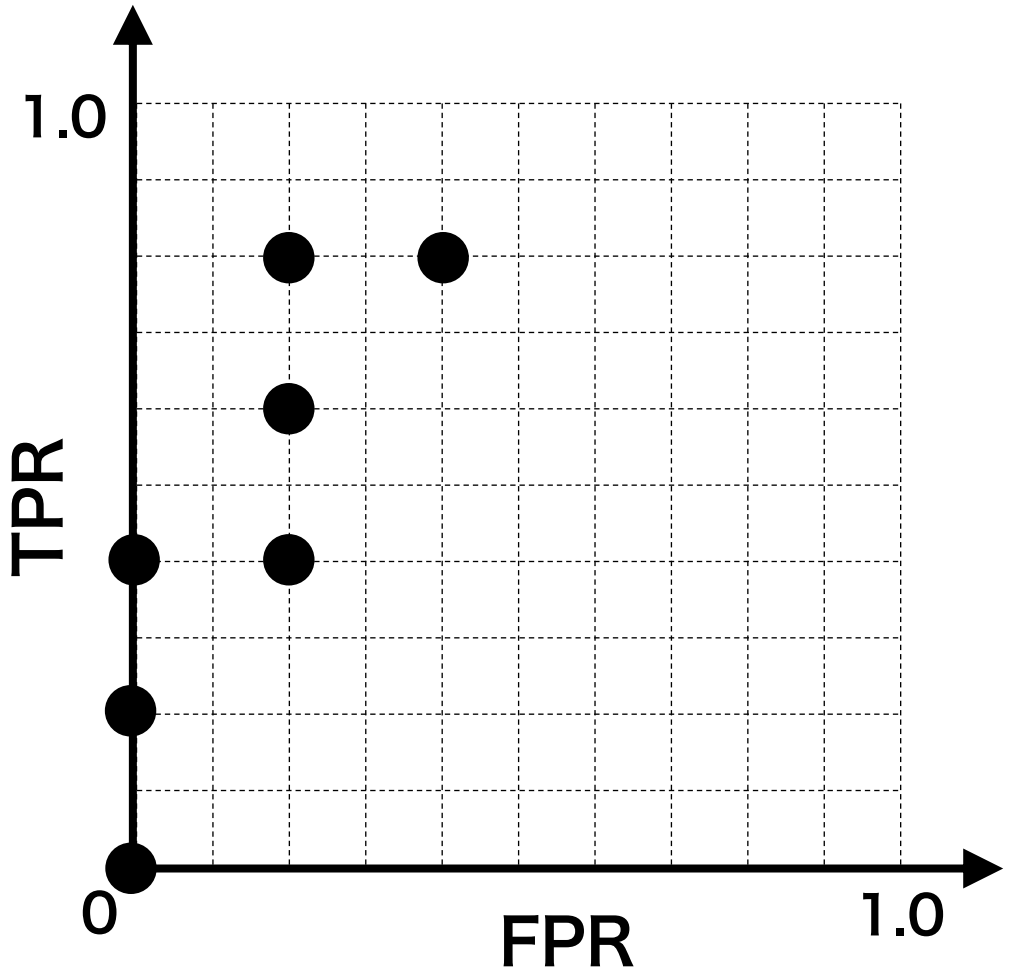


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	猫	TP
0.702	猫	猫	TP
0.540	X	猫	FP
0.438	X	X	TN
0.419	猫	X	FN
0.331	X	X	TN
0.252	X	X	TN

閾値

$$TPR = \frac{TP}{TP + FN} = \frac{4}{5} \qquad FPR = \frac{FP}{FP + TN} = \frac{2}{5}$$

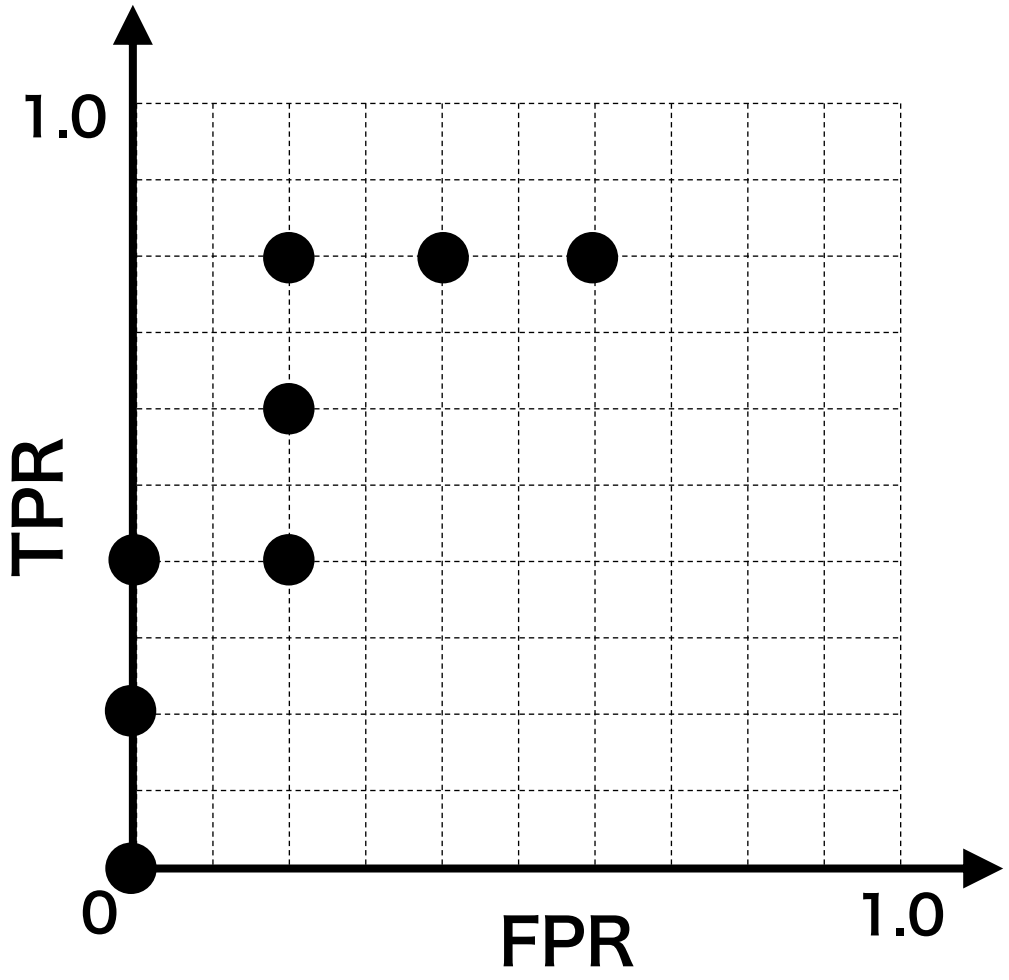


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	猫	TP
0.702	猫	猫	TP
0.540	X	猫	FP
0.438	X	猫	FP
0.419	猫	X	FN
0.331	X	X	TN
0.252	X	X	TN

閾値

$$TPR = \frac{TP}{TP + FN} = \frac{4}{5} \qquad FPR = \frac{FP}{FP + TN} = \frac{3}{5}$$

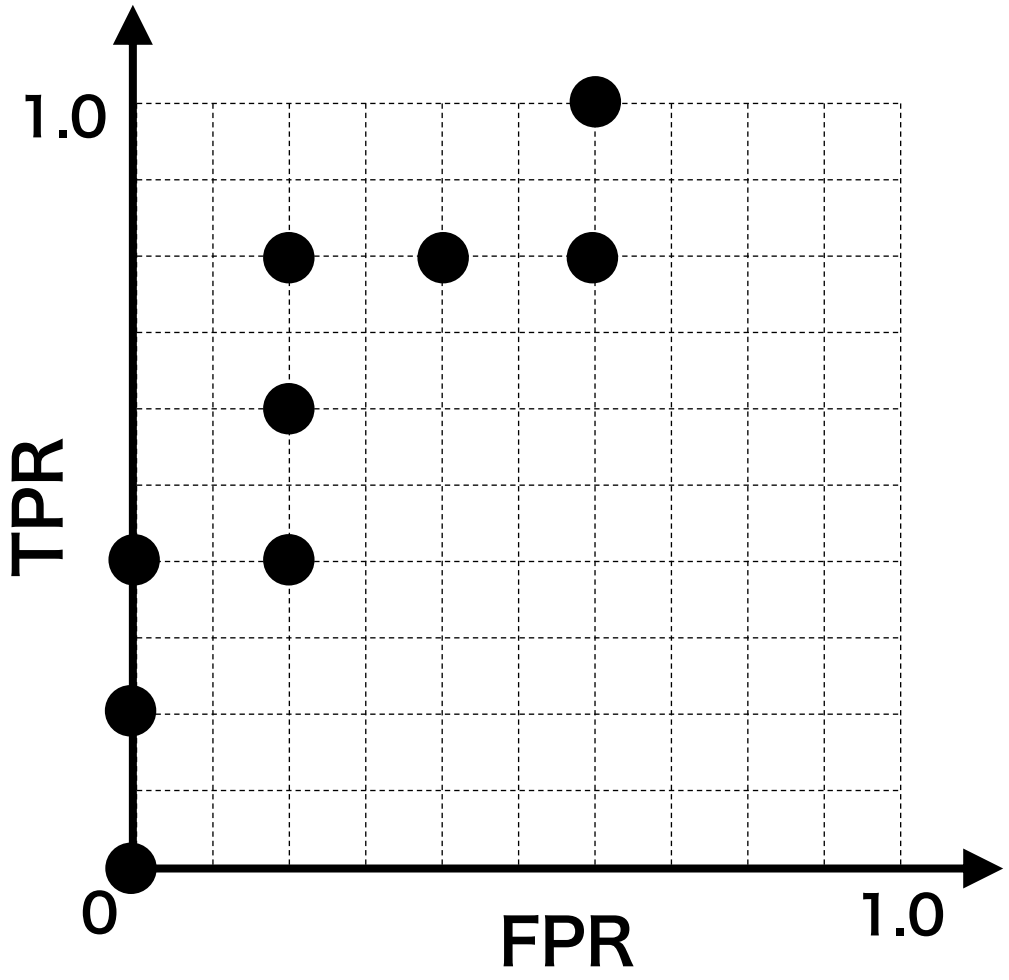


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	猫	TP
0.702	猫	猫	TP
0.540	X	猫	FP
0.438	X	猫	FP
0.419	猫	猫	TP
0.331	X	X	TN
0.252	X	X	TN

閾値

$$TPR = \frac{TP}{TP + FN} = 1 \qquad FPR = \frac{FP}{FP + TN} = \frac{3}{5}$$

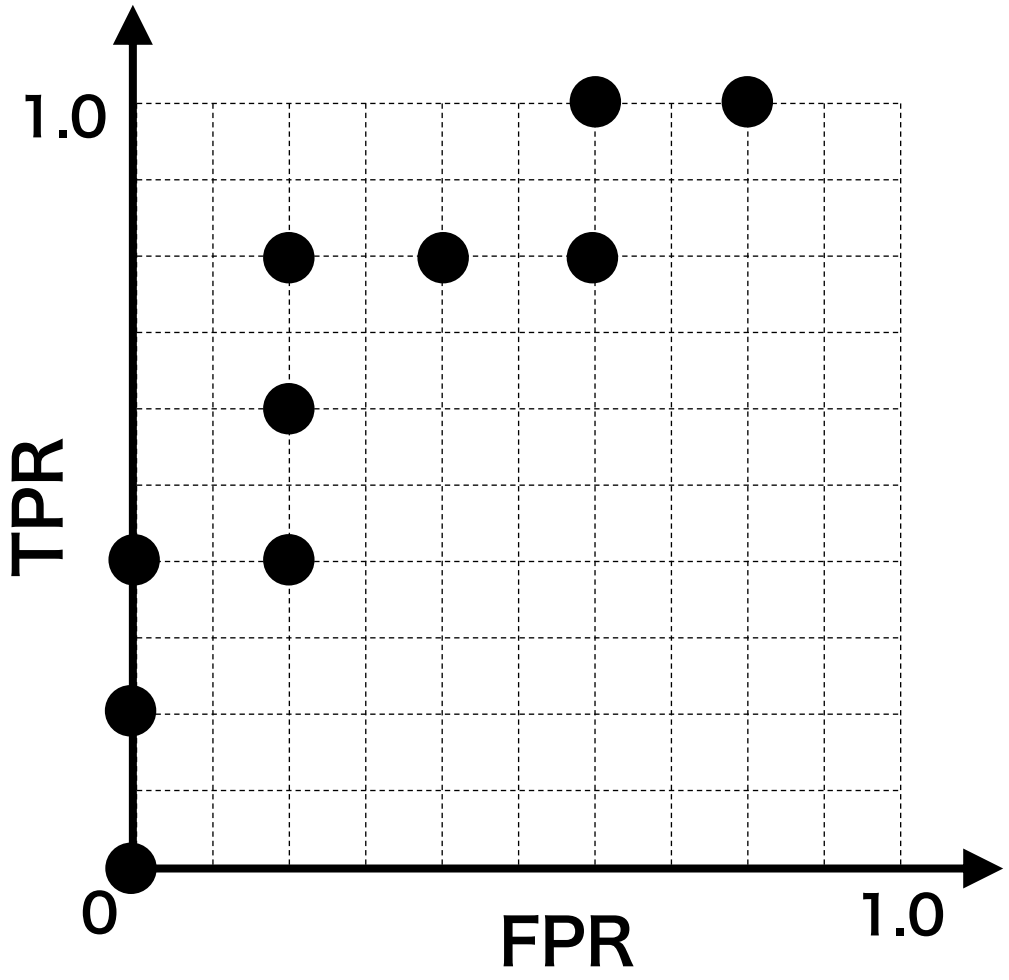


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	猫	TP
0.702	猫	猫	TP
0.540	X	猫	FP
0.438	X	猫	FP
0.419	猫	猫	TP
0.331	X	猫	FP
0.252	X	X	TN

閾値

$$TPR = \frac{TP}{TP + FN} = 1 \qquad FPR = \frac{FP}{FP + TN} = \frac{4}{5}$$

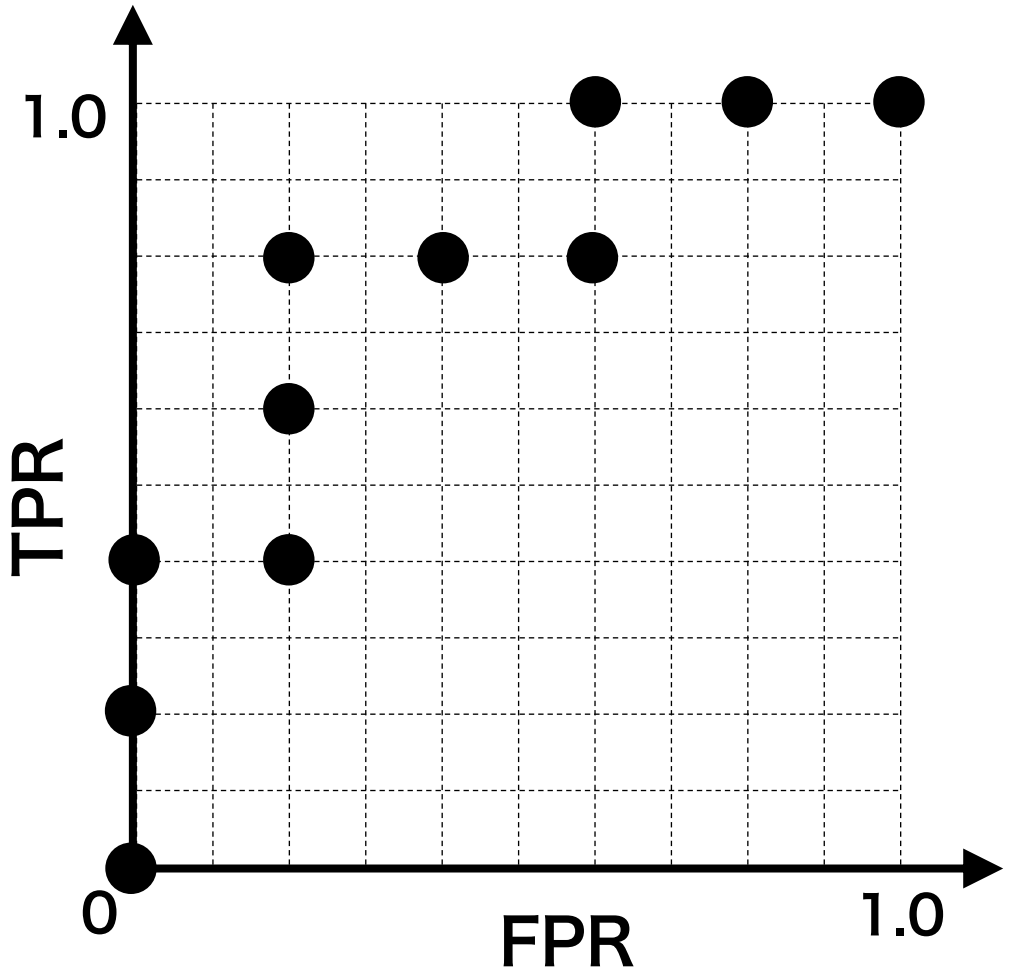


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	猫	TP
0.702	猫	猫	TP
0.540	X	猫	FP
0.438	X	猫	FP
0.419	猫	猫	TP
0.331	X	猫	FP
0.252	X	猫	FP

閾値

$$TPR = \frac{TP}{TP + FN} = 1 \qquad FPR = \frac{FP}{FP + TN} = 1$$

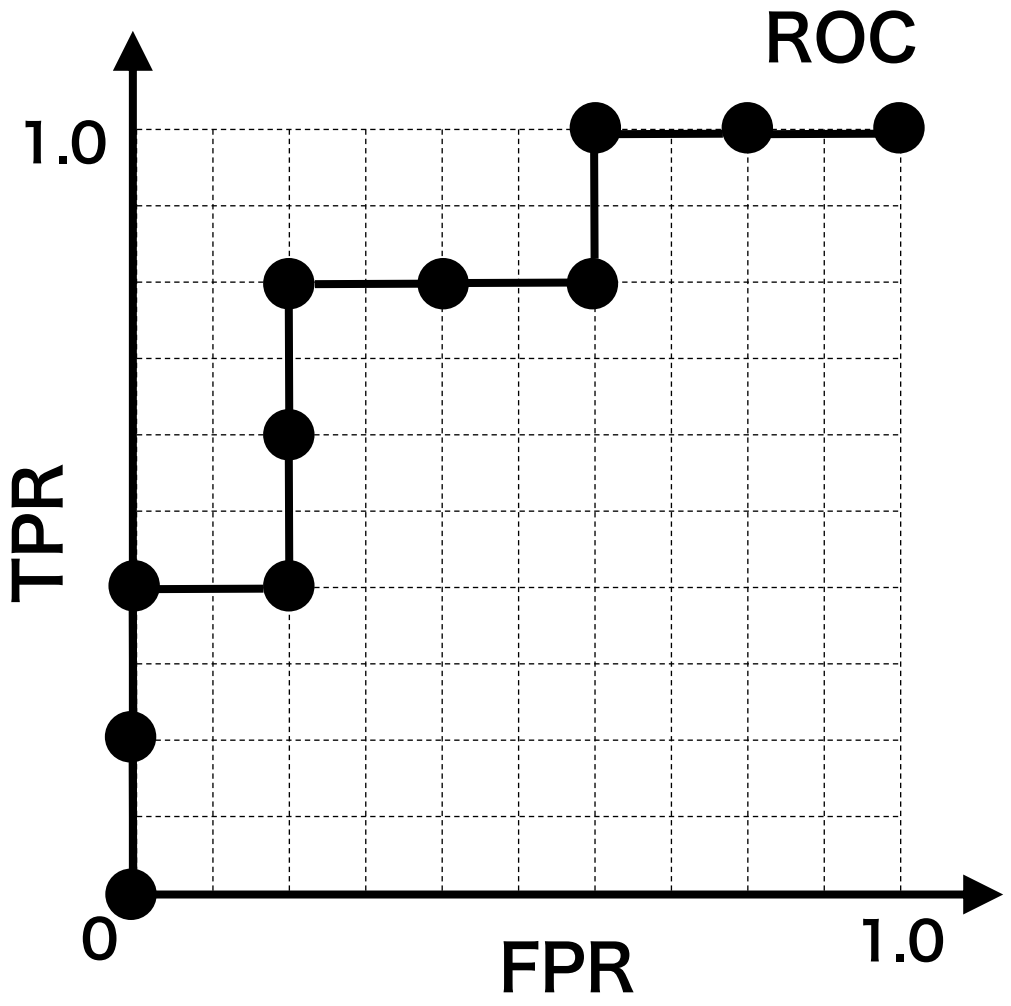


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	猫	TP
0.702	猫	猫	TP
0.540	X	猫	FP
0.438	X	猫	FP
0.419	猫	猫	TP
0.331	X	猫	FP
0.252	X	猫	FP

閾値

$$TPR = \frac{TP}{TP + FN} = 1 \qquad FPR = \frac{FP}{FP + TN} = 1$$

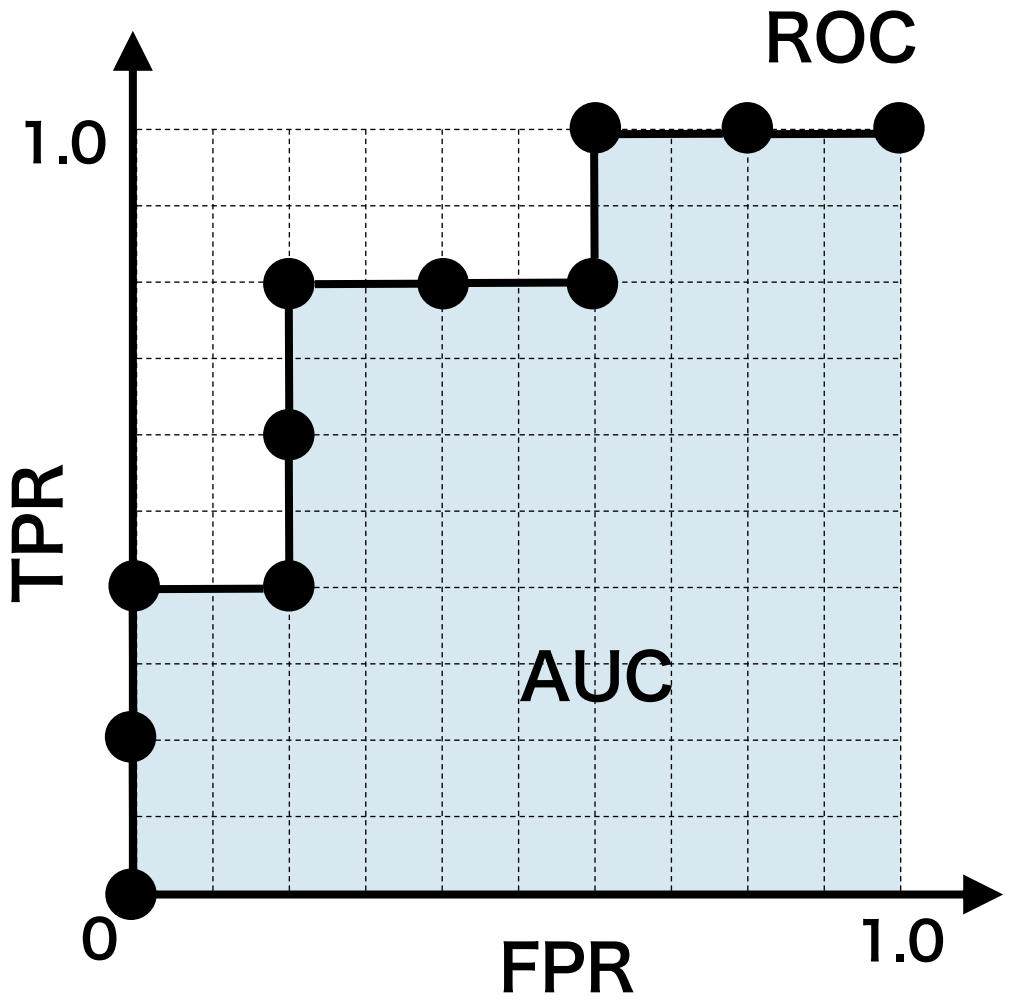


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	猫	TP
0.702	猫	猫	TP
0.540	X	猫	FP
0.438	X	猫	FP
0.419	猫	猫	TP
0.331	X	猫	FP
0.252	X	猫	FP

閾値

$$TPR = \frac{TP}{TP + FN} = 1 \qquad FPR = \frac{FP}{FP + TN} = 1$$

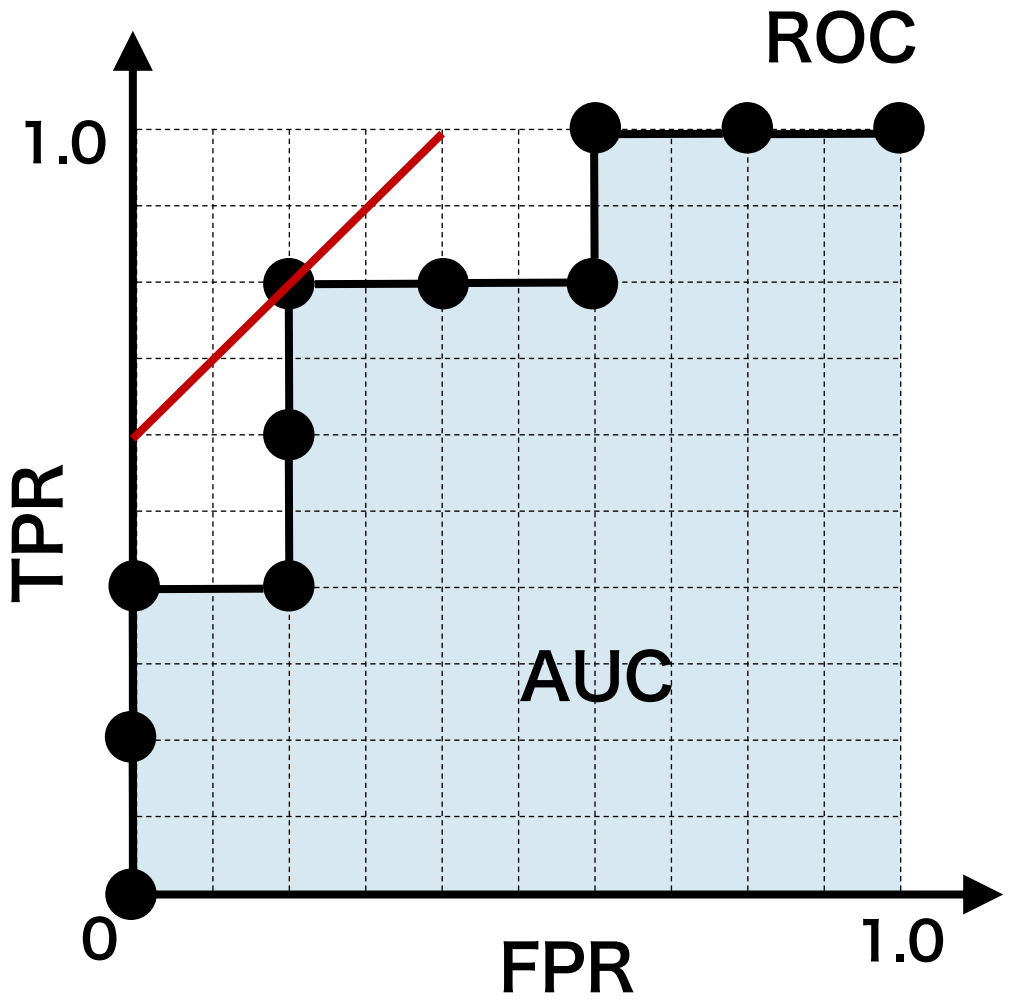


評価指標（ROC 曲線）

スコア	ラベル	予測	評価
0.984	猫	猫	TP
0.933	猫	猫	TP
0.870	X	猫	FP
0.867	猫	猫	TP
0.702	猫	猫	TP
0.540	X	X	TN
0.438	X	X	TN
0.419	猫	X	FN
0.331	X	X	TN
0.252	X	X	TN

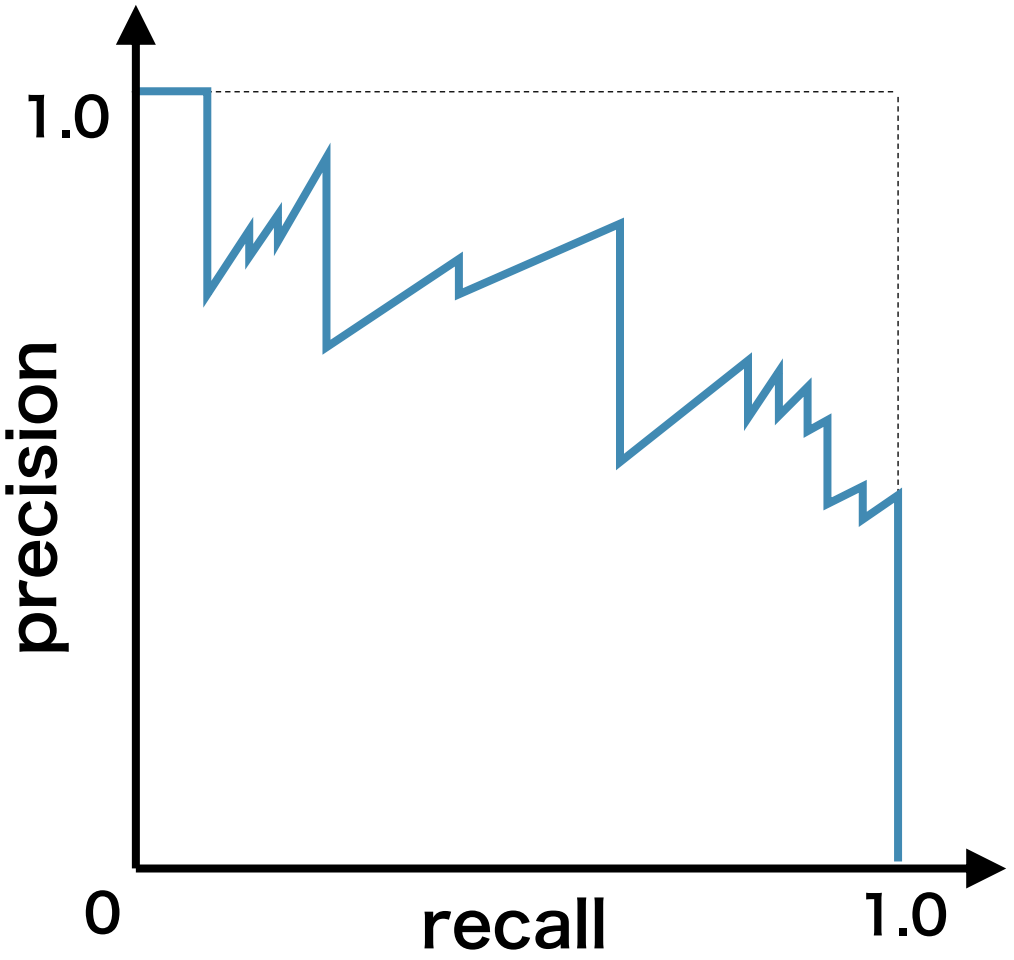
閾値

点 (0.0, 1.0) から ROC に結ぶ線分のうち、線分の長さが最短となるときの閾値が、最適な閾値として利用されることがある。



評価指標（PR 曲線）

PR 曲線は、ROC 曲線と同様に、positive と判断するスコアの閾値を動かして、precision と Recall を計算し、縦軸に precision、横軸に recall を取ったときに描かれる線グラフである。PR 曲線の下の部分の面積を AUC として計算することができ、AUC が 1 に近づくほどモデルの性能が良いとされる。PR 曲線は、positive データと negative データのサンプル数に偏りがある場合に有効である。



評価指標

多クラス分類問題の場合の評価指標は、２ 値分類問題の評価指標を One-versus-All に拡張した評価指標が使われる。この拡張方法には、マクロ平均法とマイクロ平均法がある。例えば、PRE のマクロ平均とマイクロ平均を求める場合は、右のようにする。

- マクロ平均
  - 各クラスの評価指標を平等に評価している。
  - ラベルが一部のクラスに偏っている場合でも正確に評価できる。
- マイクロ平均
  - 個々の予測結果を平等に評価している。
  - ラベルが一部のクラスに偏っている場合は、評価指標に正確にならない場合がある。

適合率

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

マクロ平均法

$$PRE^{macro} = \frac{PRE_1 + PRE_2 + \cdots + PRE_k}{k}$$

マイクロ平均法

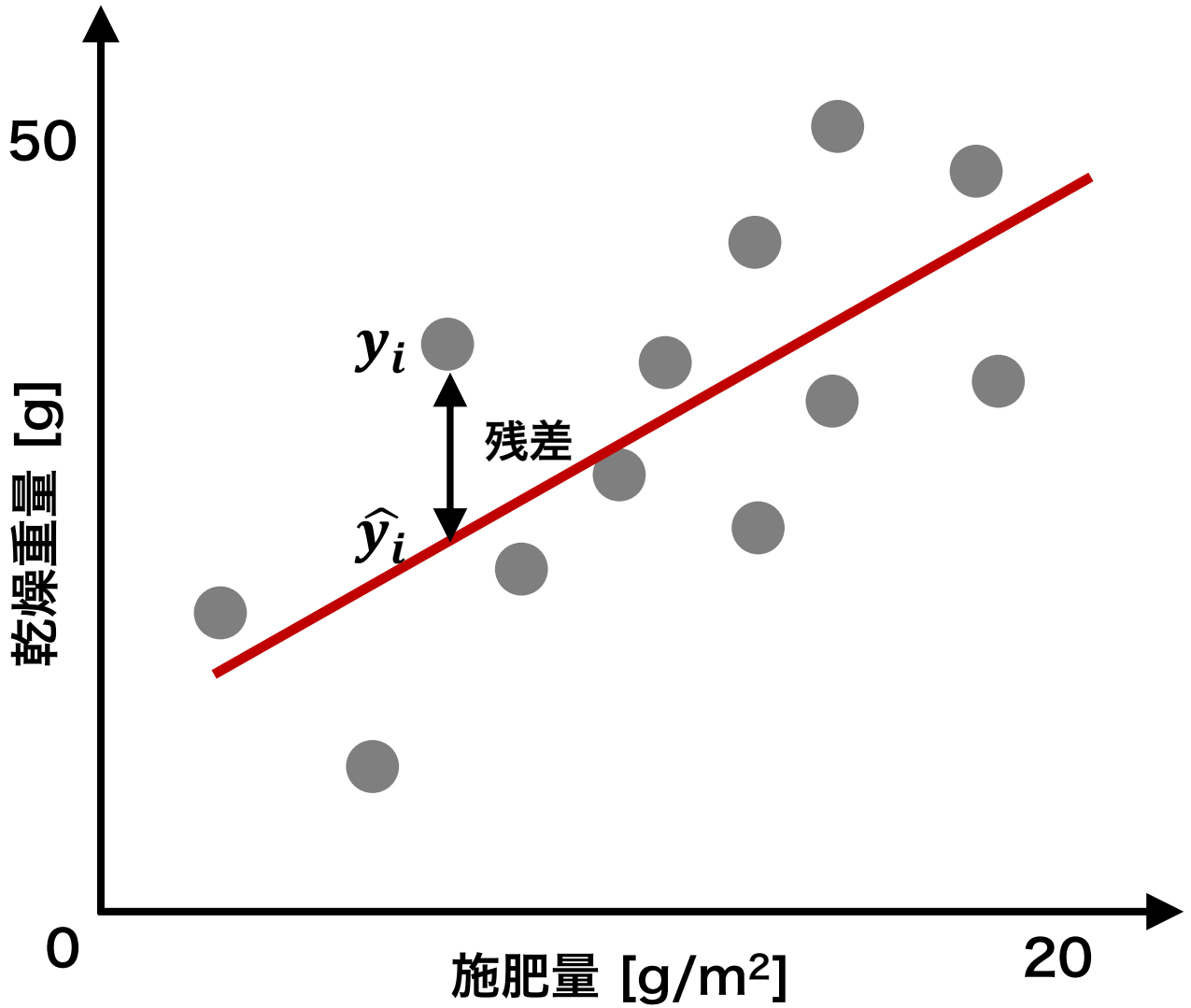
$$PRE^{micro} = \frac{TP_1 + TP_2 + \cdots + TP_k}{TP_1 + \cdots + TP_k + FP_1 + \cdots + FP_k}$$

評価指標

- 分類問題
- 回帰問題

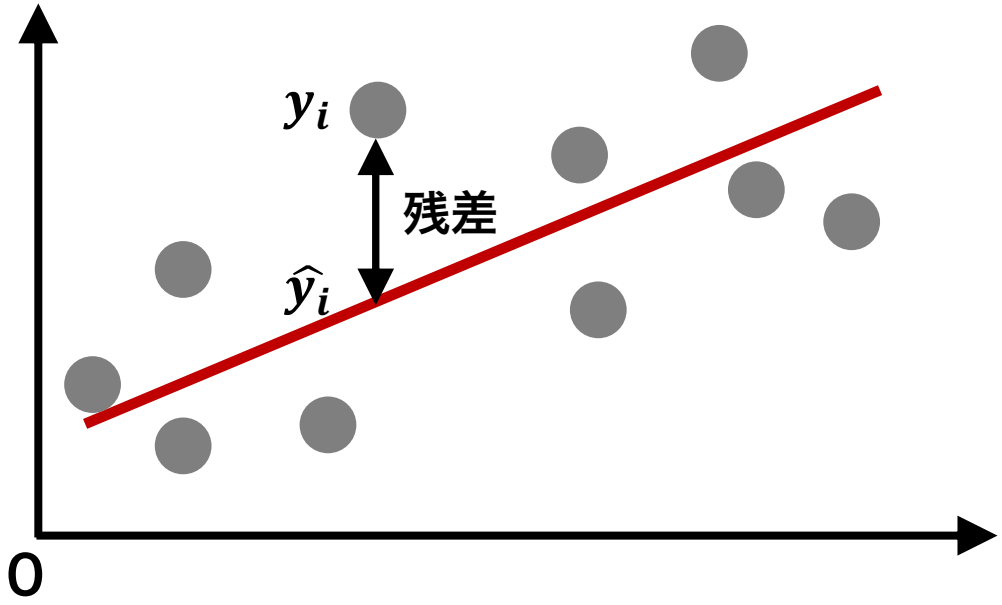
回帰分析

回帰分析は、特徴量とその特徴量に応答する連続値の関係を学習するのが目的である。回帰モデルの予測値は連続値であるため、その性能を評価するためにはアタリ・ハズレという概念が不適である。そのため、回帰モデルの性能を評価するためには、予測値が真の値に比べて、どれぐらい離れているのかを指標としている。



評価指標

回帰分析は、特徴量とその特徴量に応答する連続値の関係を学習するのが目的である。回帰モデルの予測値は連続値であるため、その性能を評価するためにはアタリ・ハズレという概念が不適である。そのため、回帰モデルの性能を評価するためには、予測値が真の値に比べて、どれぐらい離れているのかを指標としている。



R^2

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

MAE

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

MSE

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

RMSE

$$RMSE = \sqrt{MSE}$$

回帰分析

- 線形回帰
- スパース回帰
- ロバスト推定

回帰分析

- 線形回帰
- スパース回帰
- ロバスト推定

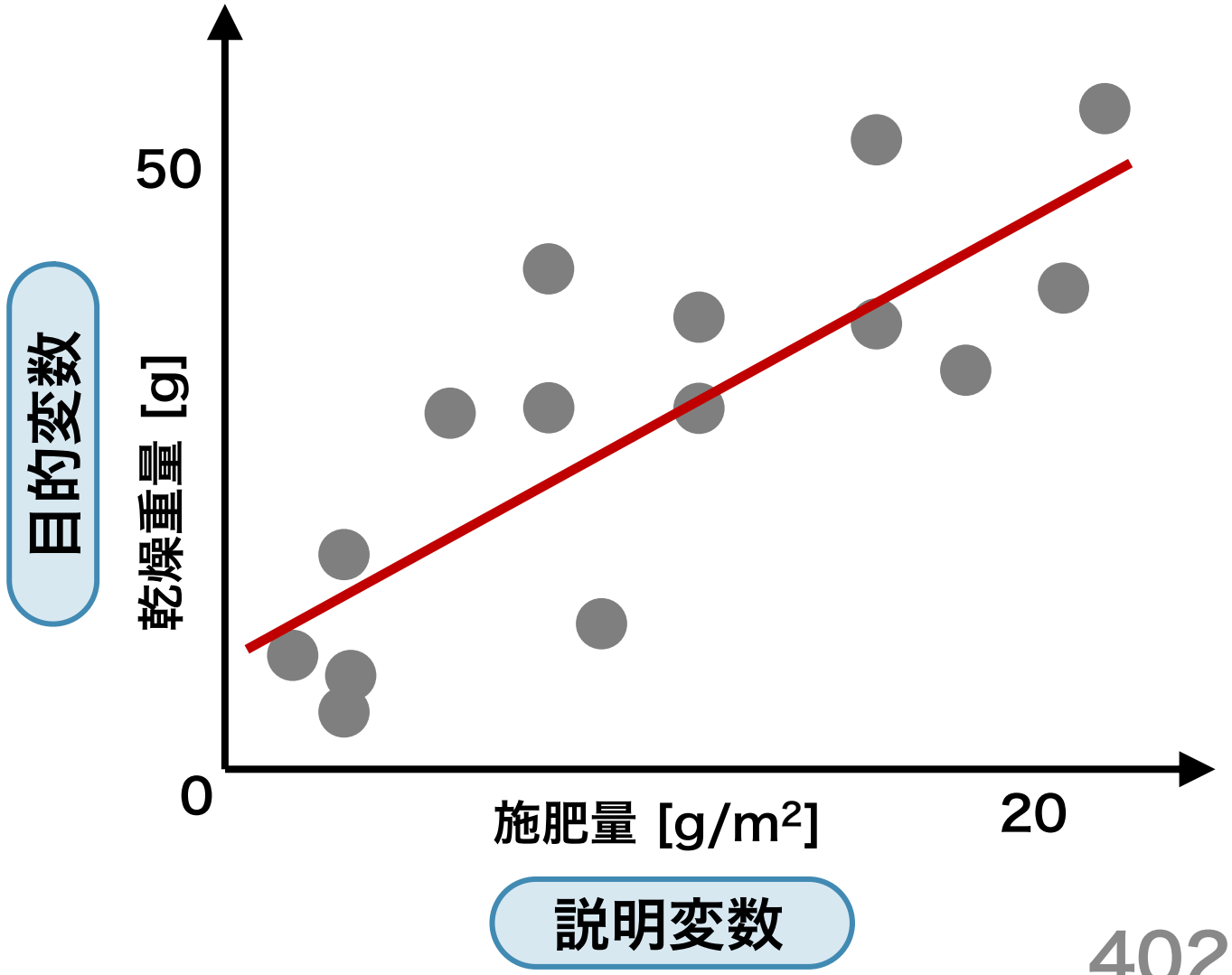
回帰分析

回帰分析は、連続値をとる目的変数を予測するために使われる教師あり学習の一つである。具体的に、特徴量とその特徴量に応答する連続値の関係を学習するのが目的である。回帰分析では、特徴量のことを説明変数、予測したい値を目的変数という。

$$y = w_0 + w_1 x$$

▲ 乾燥重量 — 目的変数

▲ 施肥量 — 説明変数

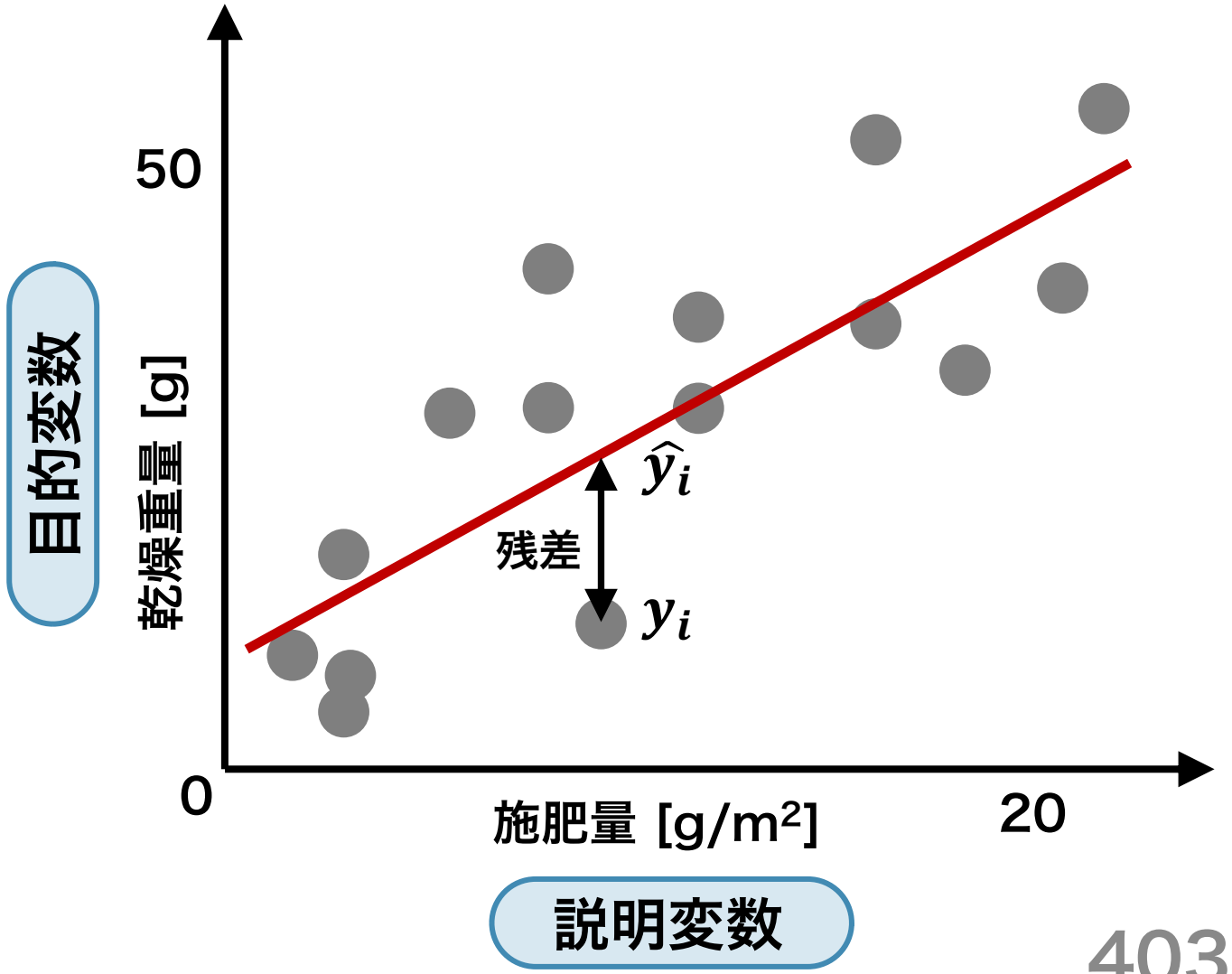


回帰分析

「特徴量とその特徴量に応答する連続値の関係」を表す最適な直線とは、観測値とその直線で予測した値の差が最も小さいということである。全サンプルの観測値と予測値の残差に着目することで、損失関数を次のように置くことができる。

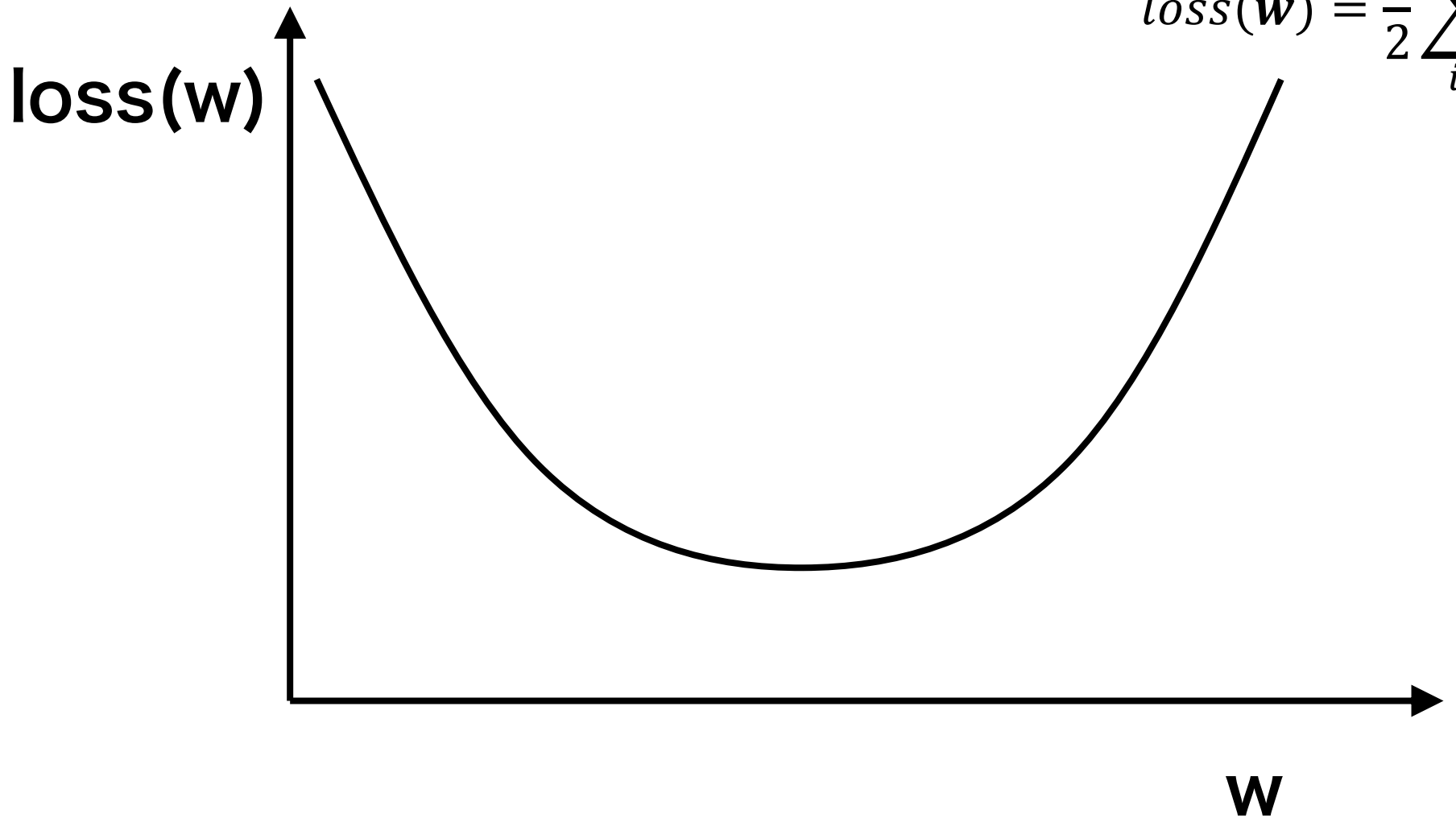
$$loss(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

$$= \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^t x_i)^2$$



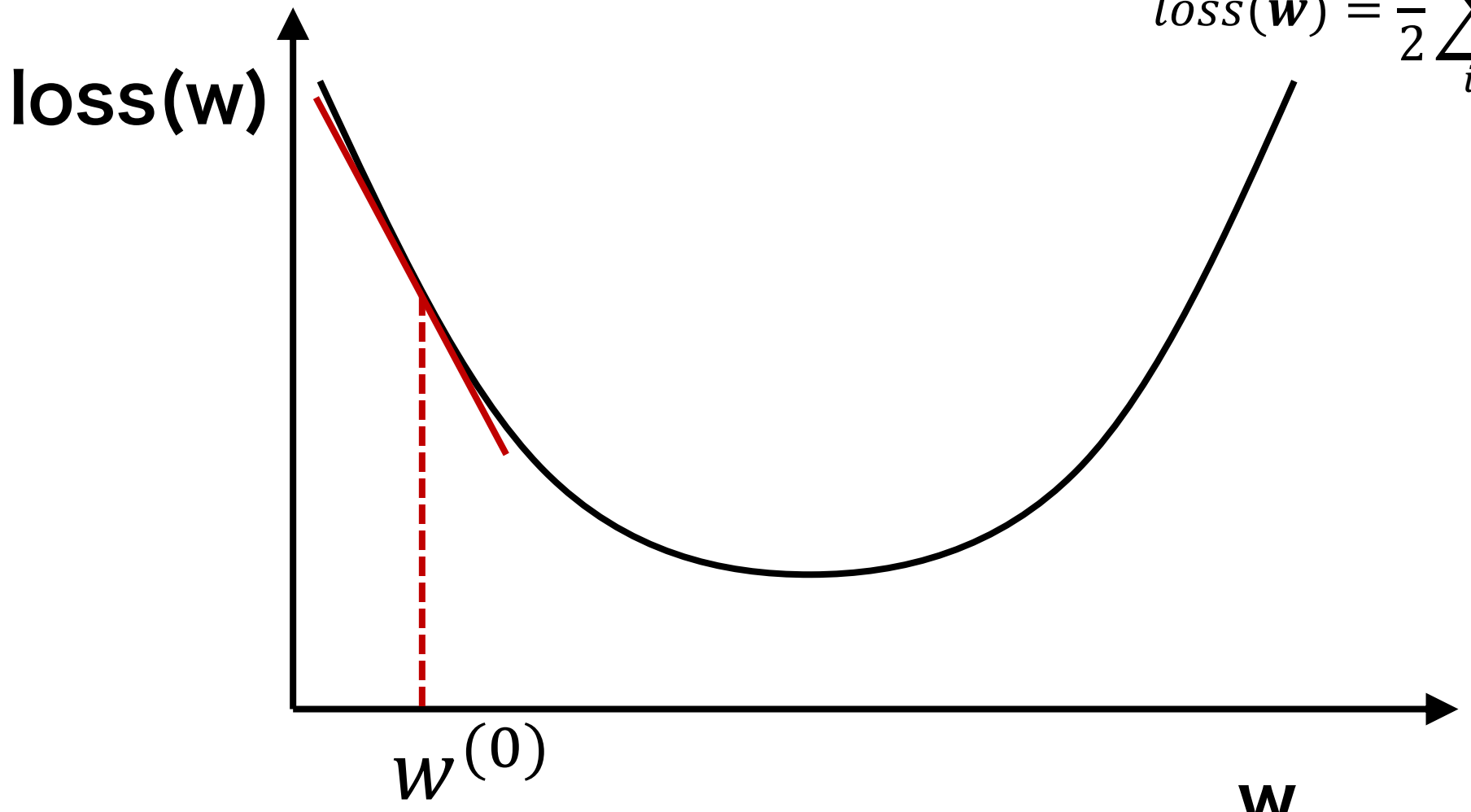
勾配降下法

$$loss(\boldsymbol{w}) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - \boldsymbol{w}^{t}x\right)^{2}$$

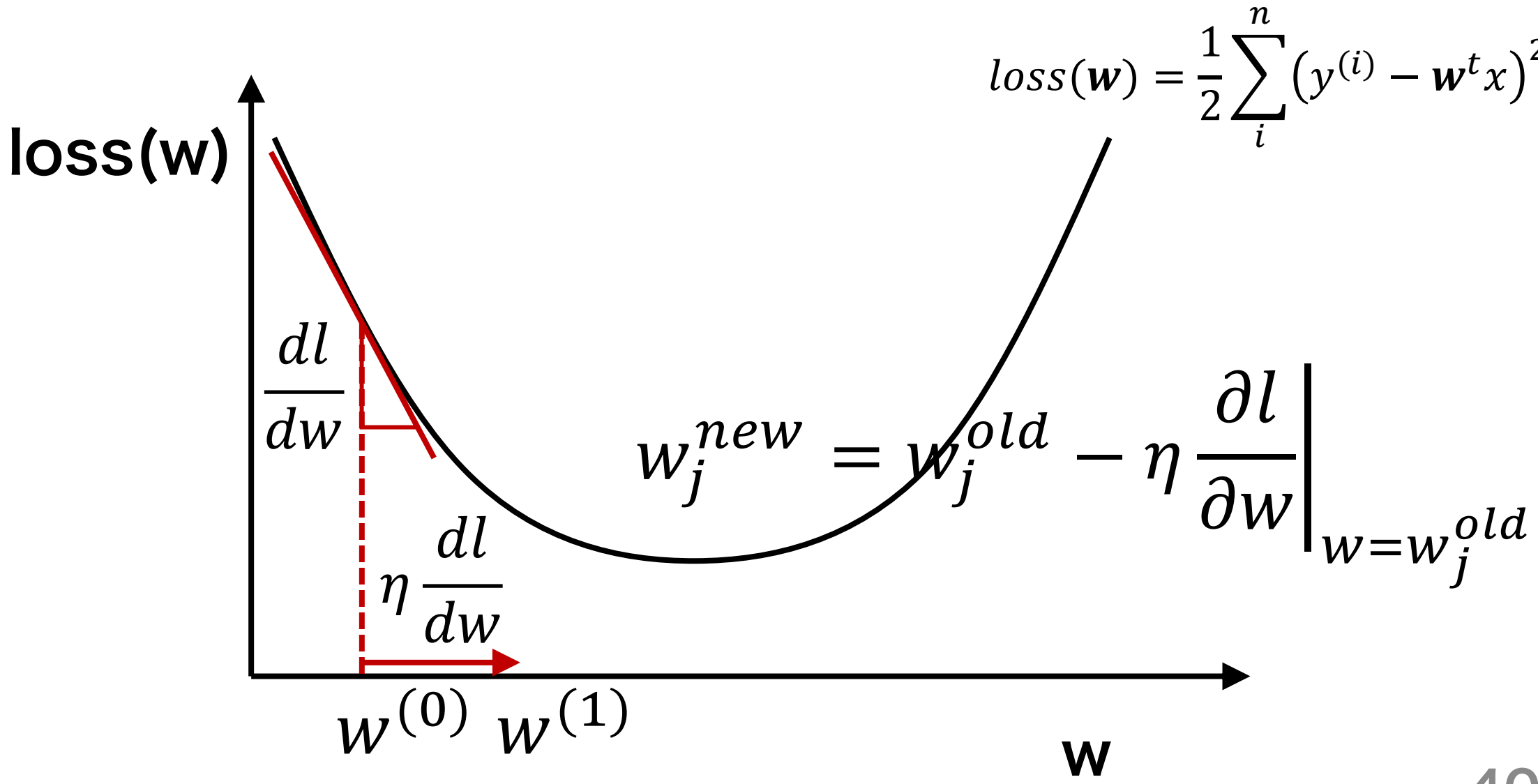


勾配降下法

$$loss(\boldsymbol{w}) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - \boldsymbol{w}^{t}x\right)^{2}$$

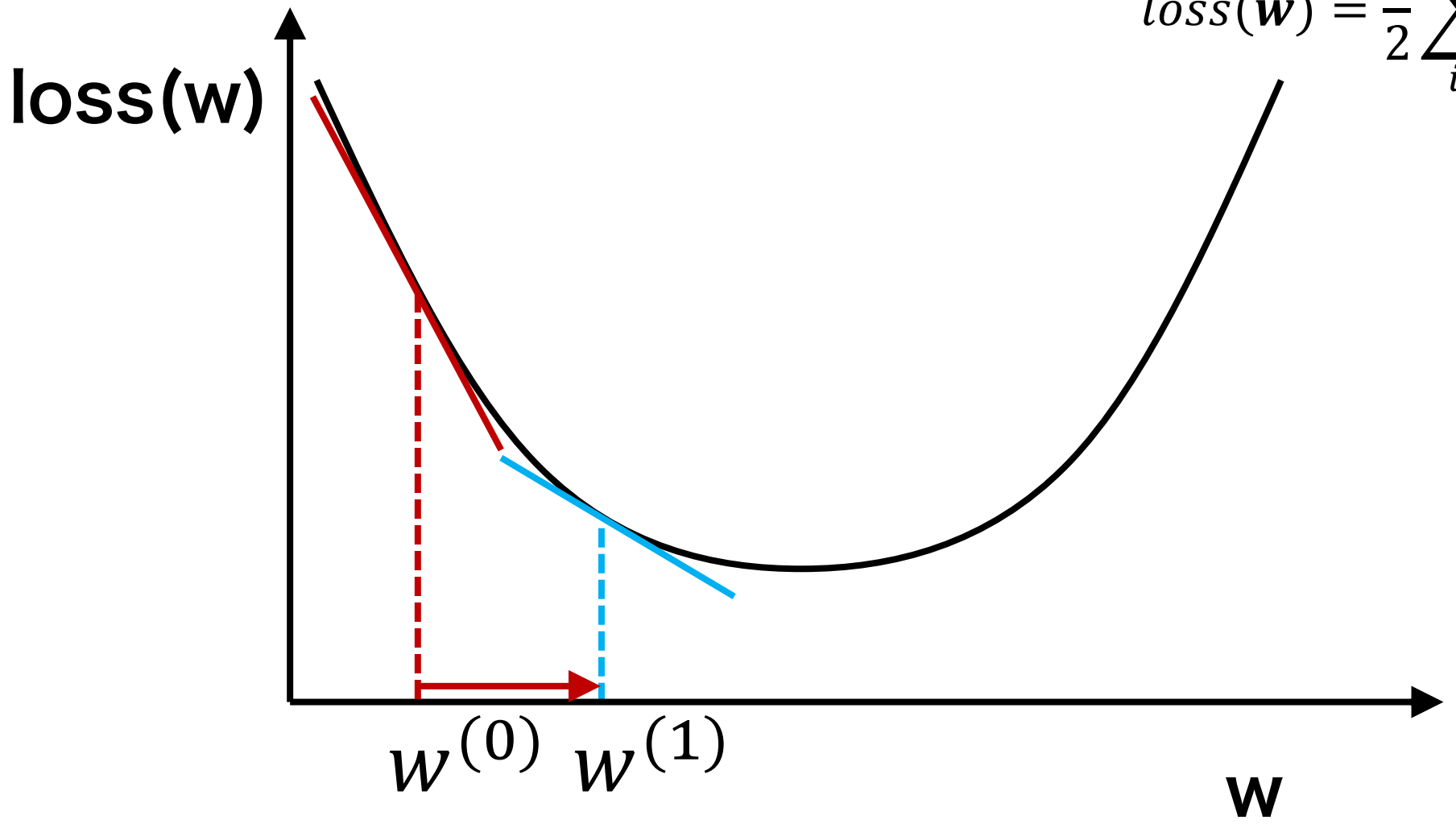


勾配降下法

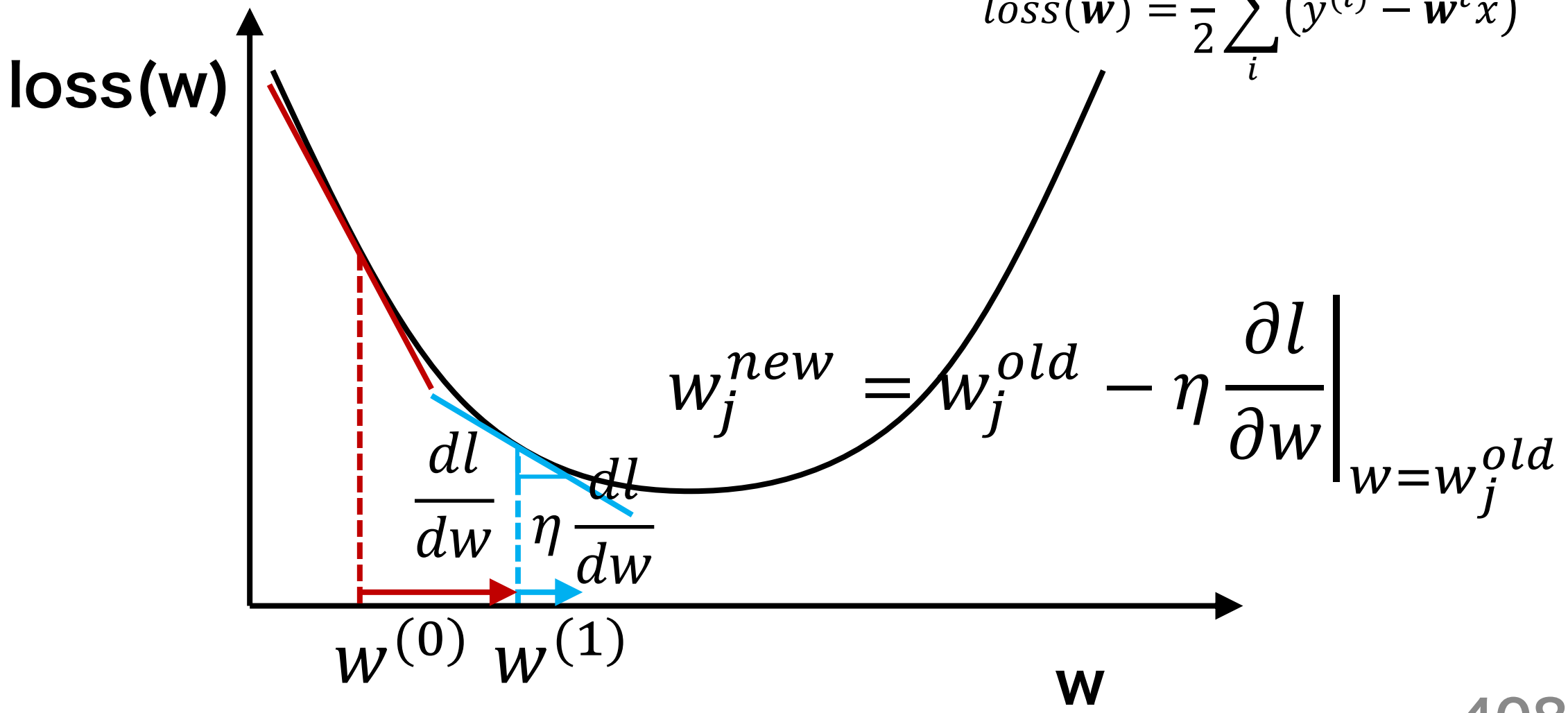

loss(w)
$loss(\boldsymbol{w}) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - \boldsymbol{w}^t x\right)^2$
$\frac{dl}{dw}$
$w_j^{new} = w_j^{old} - \eta \left.\frac{\partial l}{\partial w}\right|_{w=w_j^{old}}$
$\eta \frac{dl}{dw}$
$w^{(0)}$
$w^{(1)}$
w

勾配降下法

$$loss(\boldsymbol{w}) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - \boldsymbol{w}^{t}x\right)^{2}$$

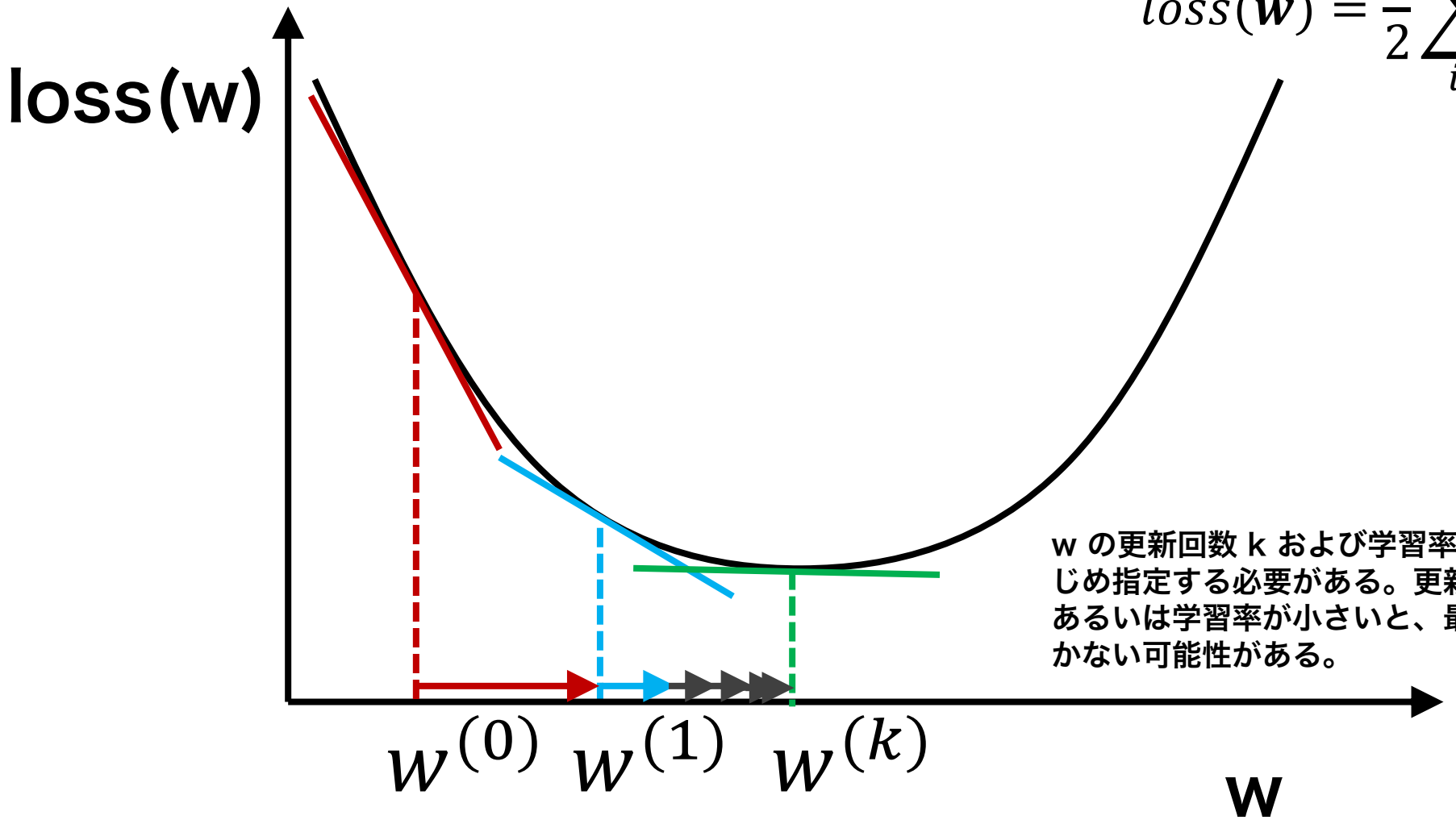


勾配降下法


loss(w)
$loss(\boldsymbol{w}) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - \boldsymbol{w}^t x\right)^2$
$w_j^{new} = w_j^{old} - \eta \left.\frac{\partial l}{\partial w}\right|_{w=w_j^{old}}$
$\frac{dl}{dw}$
$\eta \frac{dl}{dw}$
$w^{(0)}$
$w^{(1)}$
w

勾配降下法

$$loss(\boldsymbol{w}) = \frac{1}{2}\sum_{i}^{n}\left(y^{(i)} - \boldsymbol{w}^t x\right)^2$$



w の更新回数 k および学習率 η は、あらかじめ指定する必要がある。更新回数が少ないあるいは学習率が小さいと、最小値に辿り着かない可能性がある。

変数選択

回帰モデルを構築するとき、ほとんどの場合、説明変数が複数個存在する。目的変数を予測するのにほとんど寄与しない説明変数や非常に相関の強い説明変数のペアを回帰モデルに組み込むことで、モデルの複雑さが増えて、バイアスが高くなる。そのため、バイアスの低い頑健なモデルを作成するには、不要な説明変数を減らし、重要な説明変数だけを選択してモデルに組み込む必要がある。

○ 複雑なモデル

乾燥重量 ▼ 気温 ▼ 降水量 ▼ 湿度 ▼ 日射量 ▼

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4$$

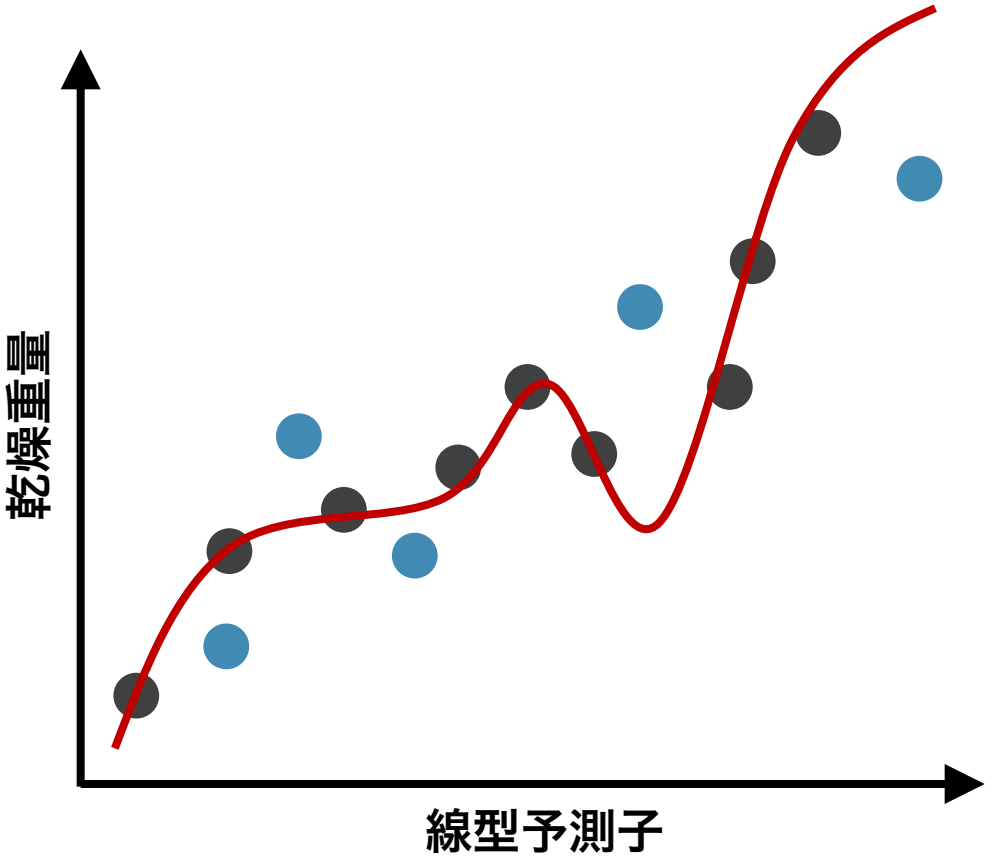
○ 単純なモデル

乾燥重量 ▼ 気温 ▼ 降水量 ▼

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

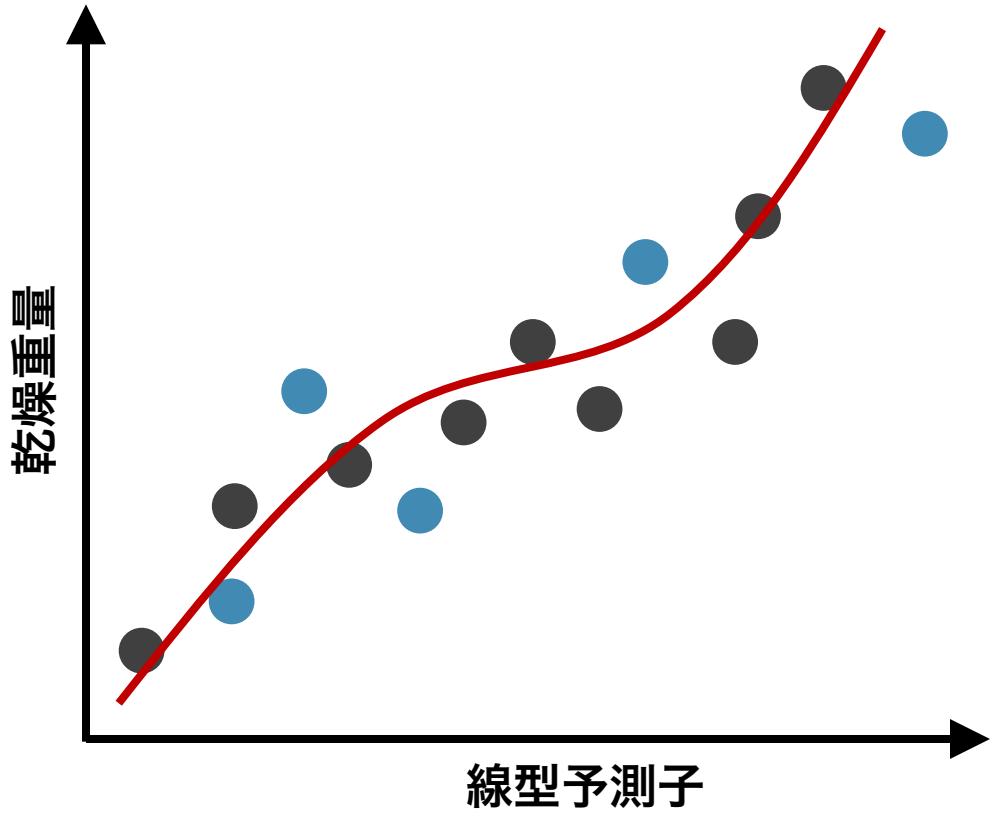
変数選択

○ $y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$



変数選択

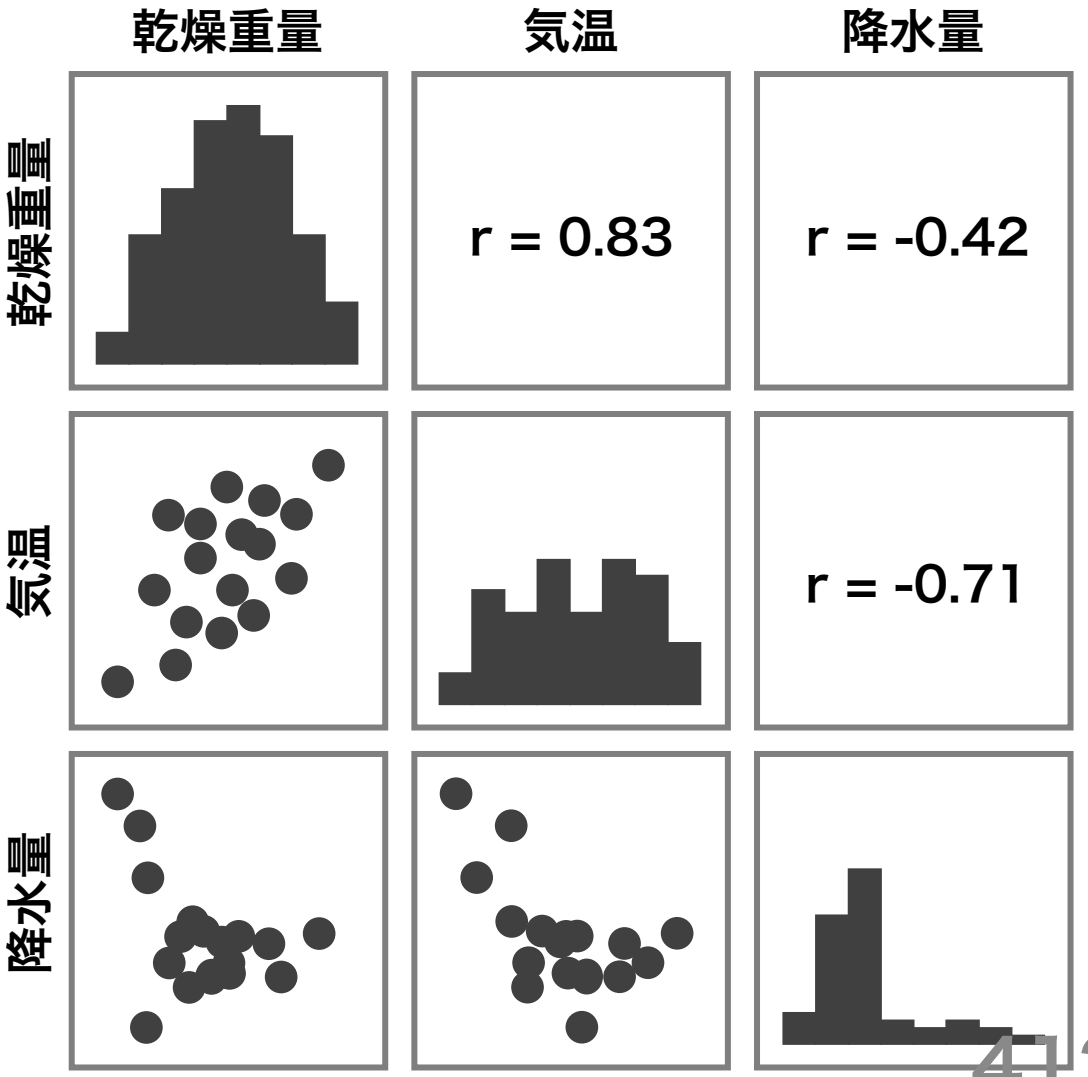
○ $y = w_0 + w_1x_1 + w_2x_2$



変数選択

重要な目的変数を選択するには、各変数の分布や変数間の相関等を調べる必要がある。そのため、モデルを構築する前に、例えば右図のようなペアプロットを描いて、変数の分布や特徴などを確認することが推奨される。

各変数の分布について、統計学における回帰分析では、変数が正規分布に従うことが必要とされる。これに対して、機械学習の分野では、説明変数および目的変数が正規分布に従う必要はないとされる。これは、機械学習において、第１種のエラーおよび第２種のエラーを評価する必要がないため、エラーを調べるための分布を仮定しなくてもよいからである。



変数選択

機械学習モデルが過度に複雑すぎる（パラメーター数が多すぎる）と、モデルが訓練データに過度にフィッティングしてしまい、汎用性を失うことがある。このことを過学習という。過学習を抑えるためには、データを増やしたり、パラメーターの数を減らしたりする必要がある。とくに、データが増やせない場合に、重要な特徴量だけを選択してモデルを構築することが、予測性能を高くする上で非常に役立つ。

- **フィルタ法**　特徴量と正解ラベルの相関や交互情報量等を計算して、閾値よりも小さければ削除する方法。
- **ラッパー法**　特徴量の一部だけを使用してモデルを構築し評価する。これを繰り返すことで最適な特徴量を選択する方法。
- **組み込み法**　学習アルゴリムズに組み込まれている特徴量選択法。決定木や L1 正則化などが組み込み法である。

変数選択

機械学習モデルが過度に複雑すぎる（パラメーター数が多すぎる）と、モデルが訓練データに過度にフィッティングしてしまい、汎用性を失うことがある。このことを過学習という。過学習を抑えるためには、データを増やしたり、パラメーターの数を減らしたりする必要がある。とくに、データが増やせない場合に、重要な特徴量だけを選択してモデルを構築することが、予測性能を高くする上で非常に役立つ。

- ○ フィルタ法　特徴量と正解ラベルの相関や交互情報量等を計算して、閾値よりも小さければ削除する方法。
- ○ ラッパー法　特徴量の一部だけを使用してモデルを構築し評価する。これを繰り返すことで最適な特徴量を選択する方法。
- ○ 組み込み法　学習アルゴリムズに組み込まれている特徴量選択法。決定木や L1 正則化などが組み込み法である。

- ○ 相関係数
- ○ 交互情報量
- ○ カイ２乗検定
- ○ フィッシャースコア
- ○ 回帰分析
- ○ 分散
- ○ 分散分析

変数選択

機械学習モデルが過度に複雑すぎる（パラメーター数が多すぎる）と、モデルが訓練データに過度にフィッティングしてしまい、汎用性を失うことがある。このことを過学習という。過学習を抑えるためには、データを増やしたり、パラメーターの数を減らしたりする必要がある。とくに、データが増やせない場合に、重要な特徴量だけを選択してモデルを構築することが、予測性能を高くする上で非常に役立つ。

- ○ フィルタ法　特徴量と正解ラベルの相関や交互情報量等を計算して、閾値よりも小さければ削除する方法。
- ○ ラッパー法　特徴量の一部だけを使用してモデルを構築し評価する。これを繰り返すことで最適な特徴量を選択する方法。
- ○ 組み込み法　学習アルゴリムズに組み込まれている特徴量選択法。決定木や L1 正則化などが組み込み法である。

- ○ 逐次前進法

  特徴量 1 つだけモデルの作成と評価を行い、最適な特徴量を決める。次に、選択された特徴量に加えて、もう 1 つの特徴量を足してモデルの作成と評価を行う。この操作を、予測性能が改善されなくなるまで繰り返す。

- ○ 逐次後退法

  特徴量をすべて含むモデルから特徴量を 1 つだけ除去したモデルに対して評価を行い、最も不要な特徴量を取り除く。次に、そのモデルからさらにもう 1 つの特徴量を除去したモデルに対して評価を行い、最も不要な特徴量を取り除く。この操作を、予測性能が改善されなくなるまで繰り返す。

逐次前進法

特徴量 1 つだけモデルの作成と評価を行い、最適な特徴量を決める。次に、選択された特徴量に加えて、もう 1 つの特徴量を足してモデルの作成と評価を行う。この操作を、予測性能が改善されなくなるまで繰り返す。

$$y = w_0 + w_1 x_1 \quad RMSE = 0.41$$

$$y = w_0 + w_2 x_2 \quad RMSE = 0.68$$

$$y = w_0 + w_3 x_3 \quad RMSE = 0.83$$

$$y = w_0 + w_4 x_4 \quad RMSE = 0.54$$

y 乾燥重量

x_1 気温

x_2 降水量

x_3 湿度

x_4 日射量

逐次前進法

特徴量 1 つだけモデルの作成と評価を行い、最適な特徴量を決める。次に、選択された特徴量に加えて、もう 1 つの特徴量を足してモデルの作成と評価を行う。この操作を、予測性能が改善されなくなるまで繰り返す。

$$y = w_0 + w_1 x_1 \quad RMSE = 0.41$$

$$y = w_0 + w_1 x_1 + w_2 x_2 \quad RMSE = 0.32$$

$$y = w_0 + w_1 x_1 + w_3 x_3 \quad RMSE = 0.38$$

$$y = w_0 + w_1 x_1 + w_4 x_4 \quad RMSE = 0.41$$

y 乾燥重量

x_1 気温

x_2 降水量

x_3 湿度

x_4 日射量

逐次前進法

特徴量 1 つだけモデルの作成と評価を行い、最適な特徴量を決める。次に、選択された特徴量に加えて、もう 1 つの特徴量を足してモデルの作成と評価を行う。この操作を、予測性能が改善されなくなるまで繰り返す。

$$y = w_0 + w_1 x_1 \quad RMSE = 0.41$$

$$y = w_0 + w_1 x_1 + w_2 x_2 \quad RMSE = 0.32$$

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3 \quad RMSE = 0.34$$

$$y = w_0 + w_1 x_1 + w_2 x_2 + w_4 x_4 \quad RMSE = 0.36$$

y 乾燥重量

x_1 気温

x_2 降水量

x_3 湿度

x_4 日射量

変数選択

機械学習モデルが過度に複雑すぎる（パラメーター数が多すぎる）と、モデルが訓練データに過度にフィッティングしてしまい、汎用性を失うことがある。このことを過学習という。過学習を抑えるためには、データを増やしたり、パラメーターの数を減らしたりする必要がある。とくに、データが増やせない場合に、重要な特徴量だけを選択してモデルを構築することが、予測性能を高くする上で非常に役立つ。

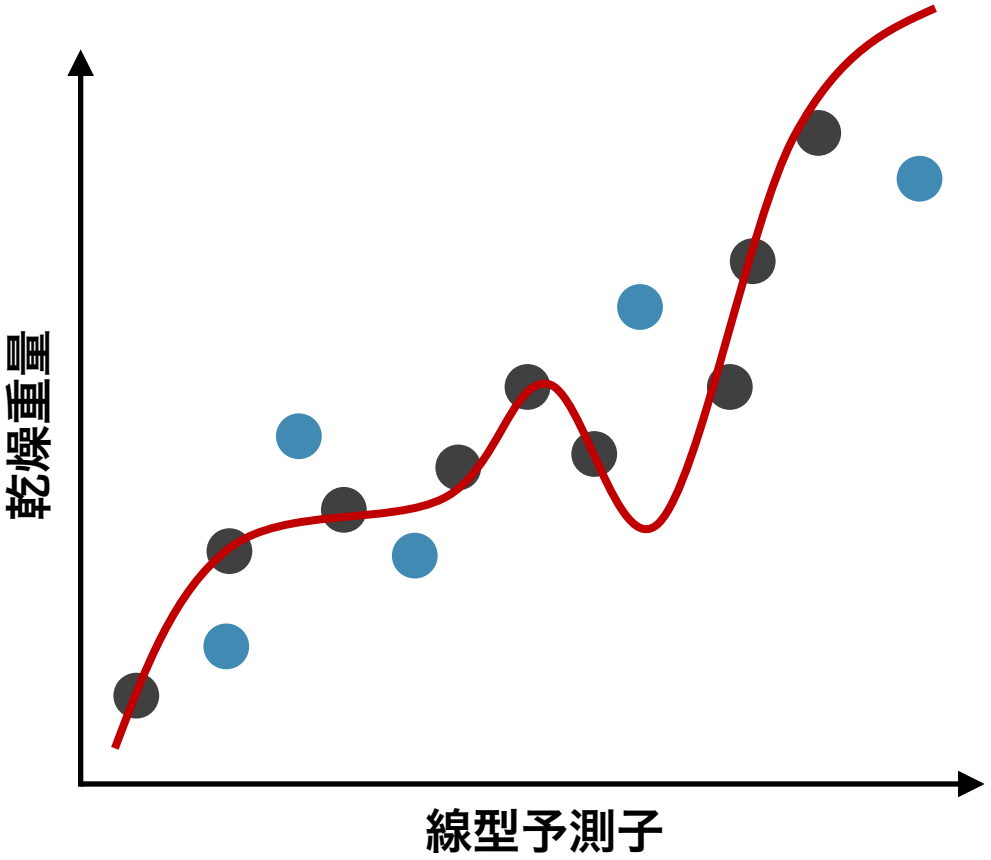
- ○ フィルタ法　特徴量と正解ラベルの相関や交互情報量等を計算して、閾値よりも小さければ削除する方法。
- ○ ラッパー法　特徴量の一部だけを使用してモデルを構築し評価する。これを繰り返すことで最適な特徴量を選択する方法。
- ◎ 組み込み法　学習アルゴリムズに組み込まれている特徴量選択法。決定木や L1 正則化などが組み込み法である。
- ○ 決定木
- ○ 正則化

回帰分析

- 線形回帰
- スパース回帰
- ロバスト推定

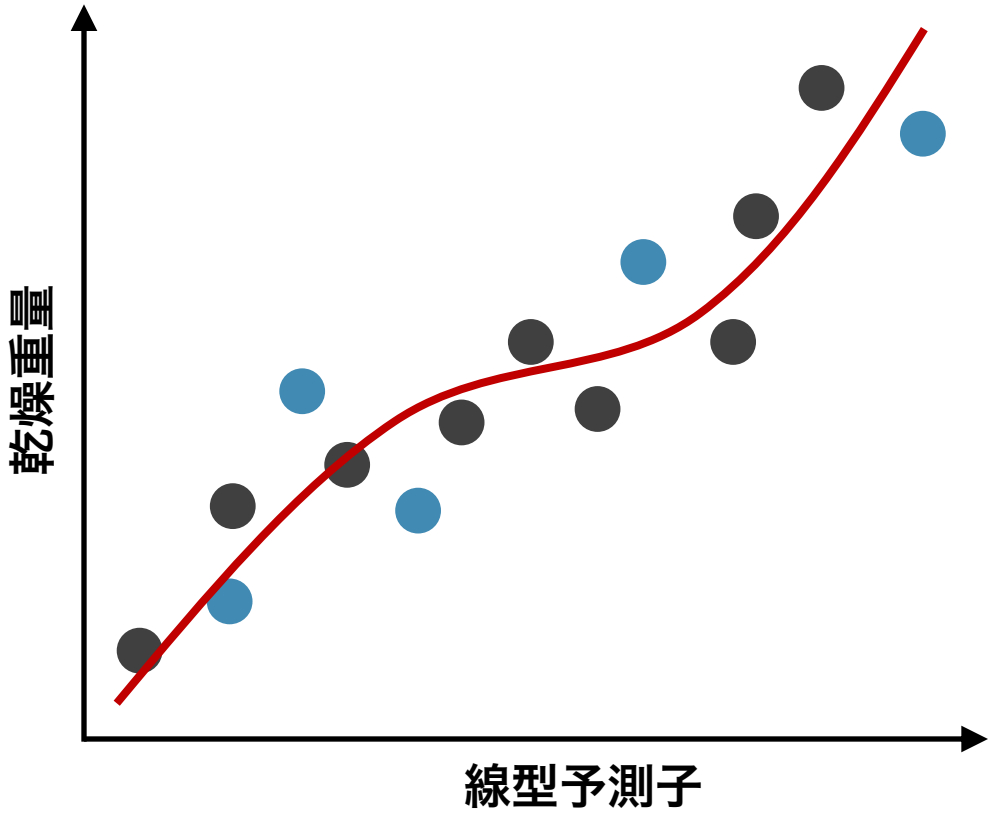
正則化

○ $y = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4$



正則化

○ $y = w_0 + w_1x_1 + w_2x_2$



正則化

回帰分析の損失関数は、全データに対する予測値と実測値の残差の二乗和である。回帰分析では、この残差二乗和が最小となるようなパラメータ w を勾配降下法により推定している。この推定において、w_i がゼロになる確率がほとんどない（i 番目の特徴量がすべてゼロあるいは共相関の特徴量でない限り、ゼロにならない）。そのため、このままで推定を行うと、モデル中のパラメータ数は、特徴量と同じ数になる。

$$loss(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^t x_i)^2$$

モデルの複雑度を減らすためには、パラメータ数を減らせば良い。そこで、例えば、右の式のように、損失関数中の残差二乗和の後にパラメータ数に対するペナルティ項を加える。このとき、勾配降下法では、残差二乗和と罰則項の両方を考慮した上で、式全体の最小化を行う。

$$loss(\boldsymbol{w}) = \underbrace{\frac{1}{2}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^t x_i)^2}_{\text{すべての特徴量の係数 w がゼロでないときに、最小値をとる。}} + \underbrace{\lambda\sum_{j=1}^{m}|w_j|}_{\text{すべての w がゼロのときに、最小値をとる。}}$$

正則化

LASSO

Sparse !

$$loss(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{w}^t\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{w}\|_1^1$$

- 罰則項が各パラメータの絶対値の和である
- パラメータがゼロになりやすい
- 相関の強い変数が複数存在すると、ランダムに 1 つだけ選ばれる、他の変数のパラメータがゼロになる

Ridge

$$loss(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{w}^t\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{w}\|_2^2$$

- 罰則項がパラメータの 2 乗和である
- パラメータがほとんどの場合ゼロにならない
- 罰則項は特徴量にノイズを加えることに等しいので、相関の強い変数が複数存在しても、ノイズが加わることで相関が弱くなり、解が求まるようになる

正則化

ElasticNet

$$loss(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{w}^t\boldsymbol{x}\|_2^2 + \lambda(\alpha\|\boldsymbol{w}\|_1^1 + (1-\alpha)\|\boldsymbol{w}\|_2^2)$$

- 罰則項が各パラメータの絶対値の和と 2 乗和を 1:(1-α) の重みをかけたあとの和である
- α の値が大きいと、LASSO の性質が強くなるため、パラメータがゼロになりやすくなる
- 相関の強い変数が複数個存在すると、該当する複数のパラメータが同じ値になりやすくなる

最小二乗解

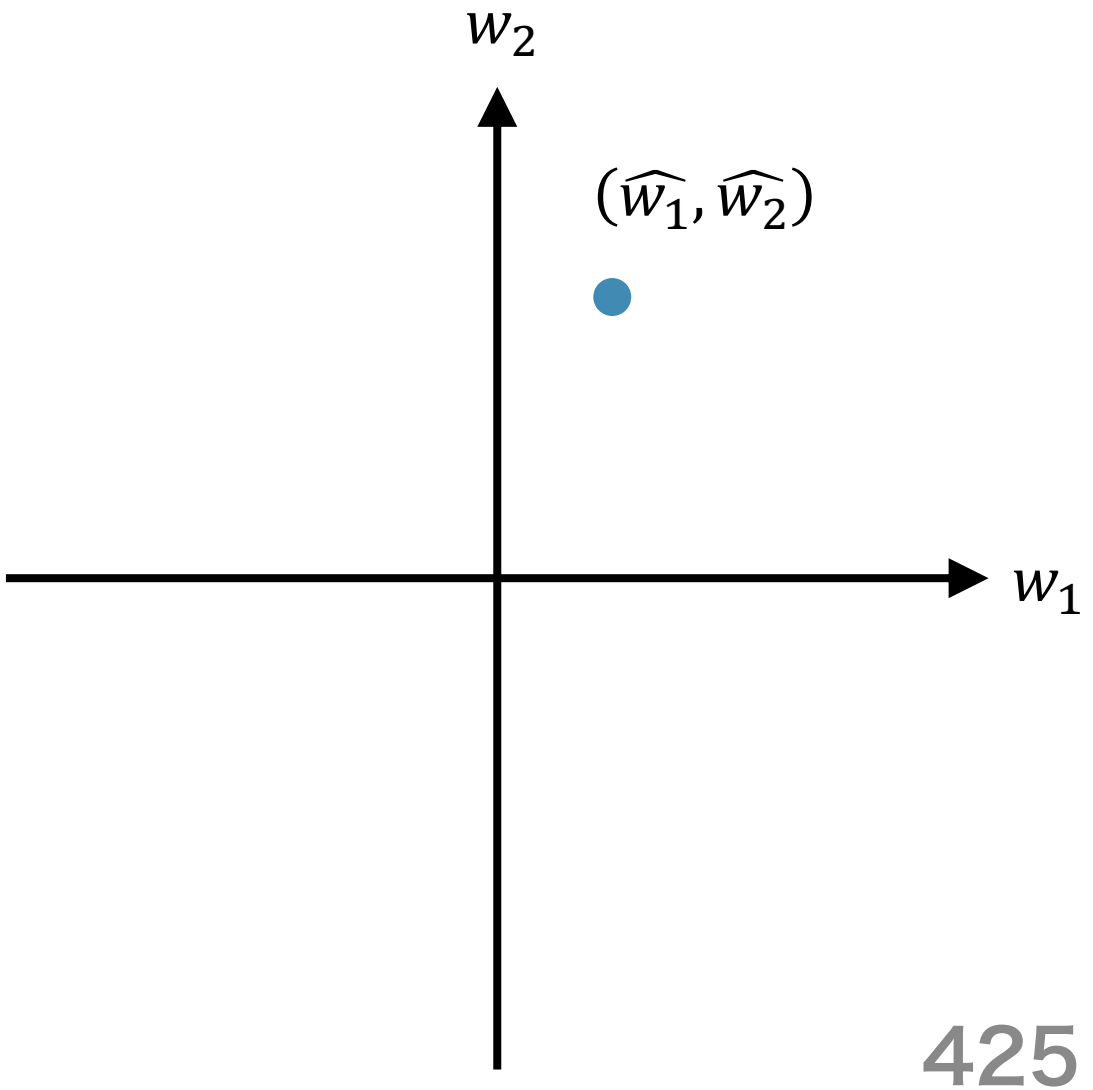
LASSO はスパース推定の手法として知られている。LASSO を用いて回帰モデルを作成すると、重要な変数だけが自動的に選択されてモデルに組み込まれる。重要でない変数のパラメータがゼロとなり、予測に貢献しなくなる。

LASSO 仕組みを説明するために、説明変数が 2 つの普通の回帰モデルを考える。このとき、回帰モデルのパラメータを推定するときの損失関数は次のようになる。

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

$$loss(\boldsymbol{w}) = \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{w}^t \boldsymbol{x}\|_2^2$$

パラメータ推定の結果、最適解が $\widehat{w_1}$ と $\widehat{w_2}$ と推定されたと仮定する。この最適解の組み合わせをパラメータ空間上にプロットすると右図のようになる。



LASSO

Least absolute shrinkage and selection operator

LASSO 回帰では、普通の回帰式にパラメータの絶対値の和を罰則項として加えている。

$$loss(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{w}^t\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{w}\|_1^1$$

上式は次式と同じことを表している。

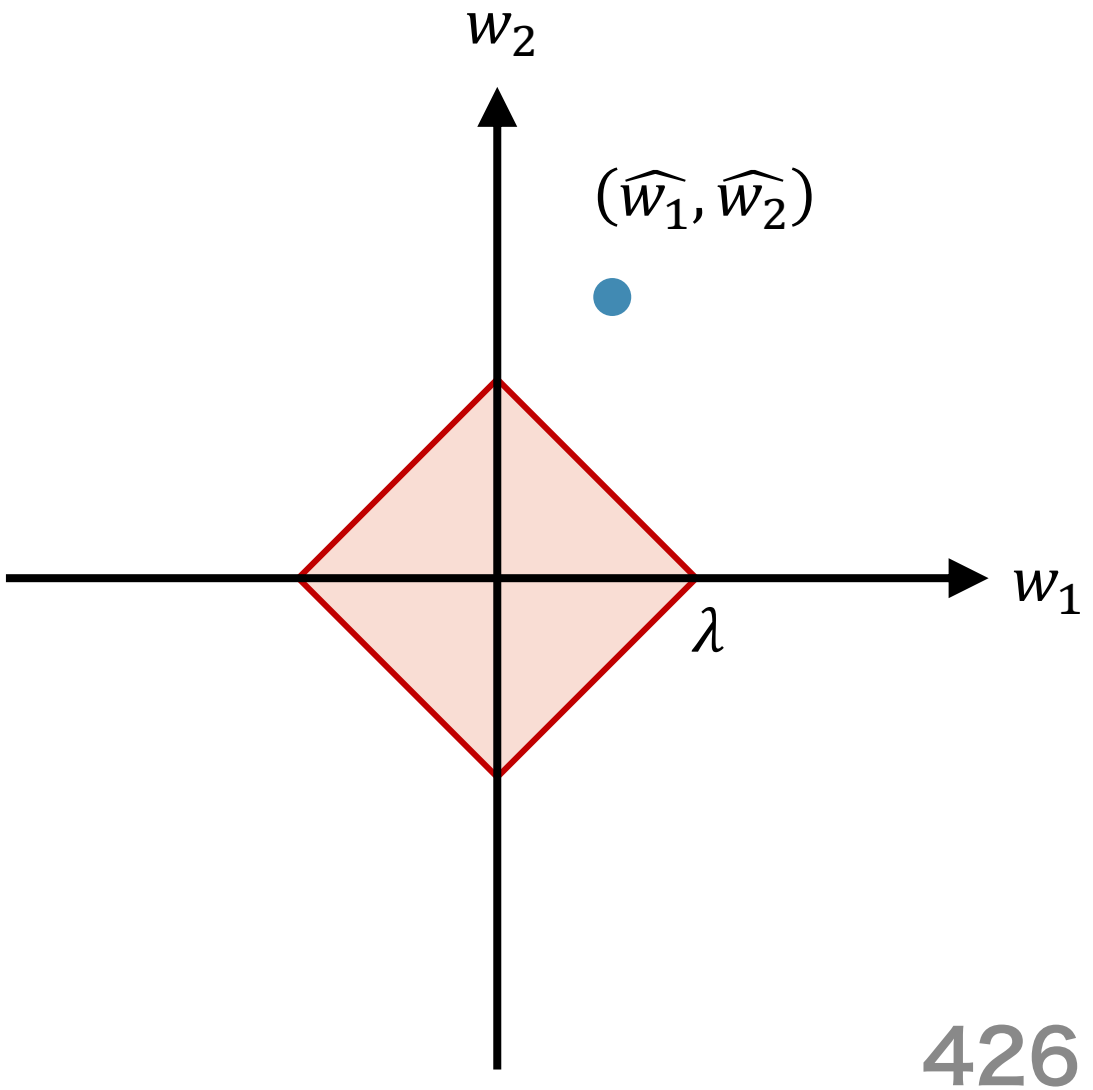
$$loss(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{w}^t\boldsymbol{x}\|_2^2$$

$$\text{subject to } \|\boldsymbol{w}\|_1^1 \leq \lambda$$

制約条件（罰則項）を展開すると次のようになる。

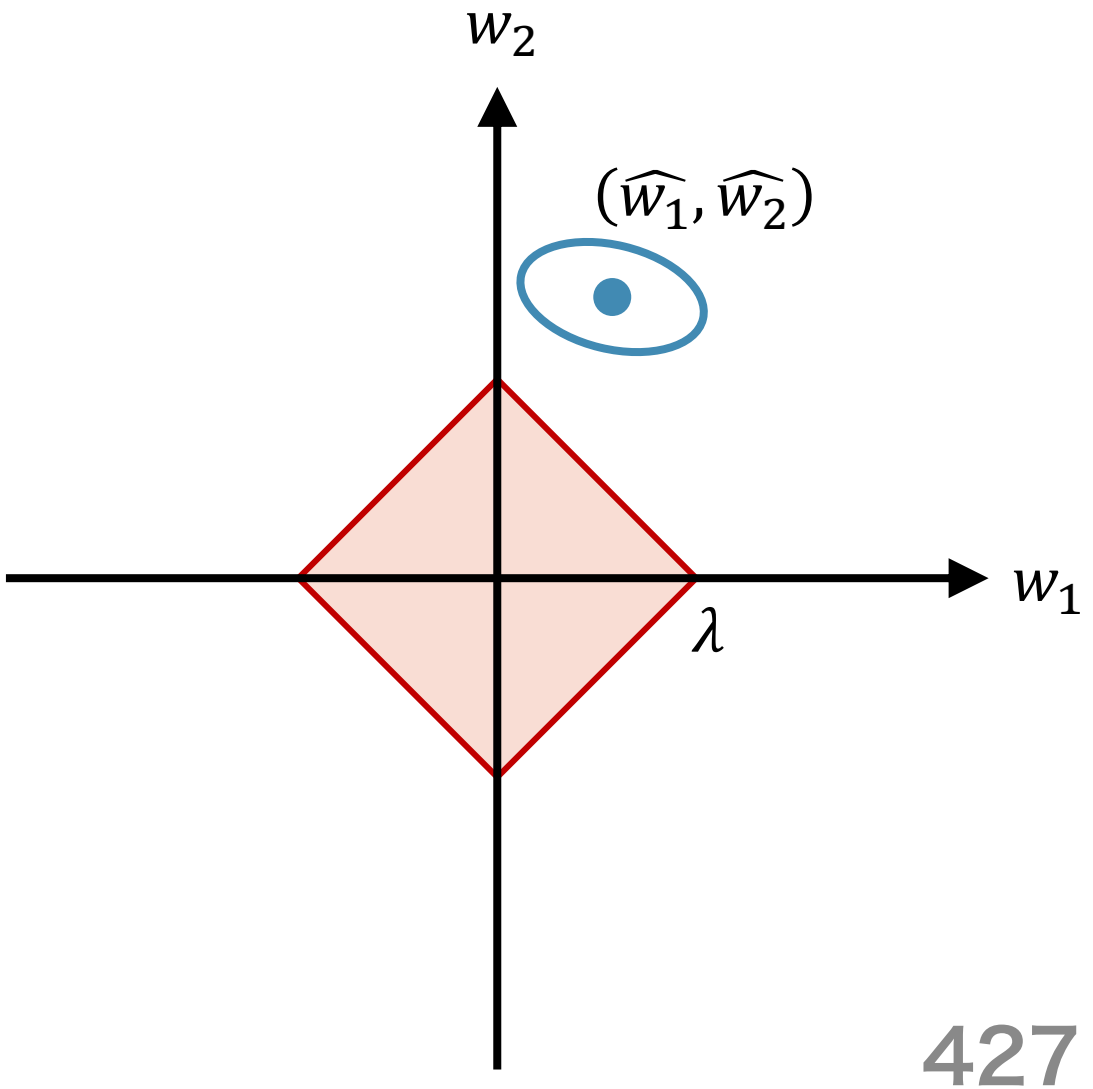
$$\|\boldsymbol{w}\|_1^1 = |w_1| + |w_2| \leq \lambda$$

つまり、解となる $(\widehat{w_1}, \widehat{w_2})$ の組み合わせは、赤枠または赤枠の内部に存在しなければならない。



LASSO

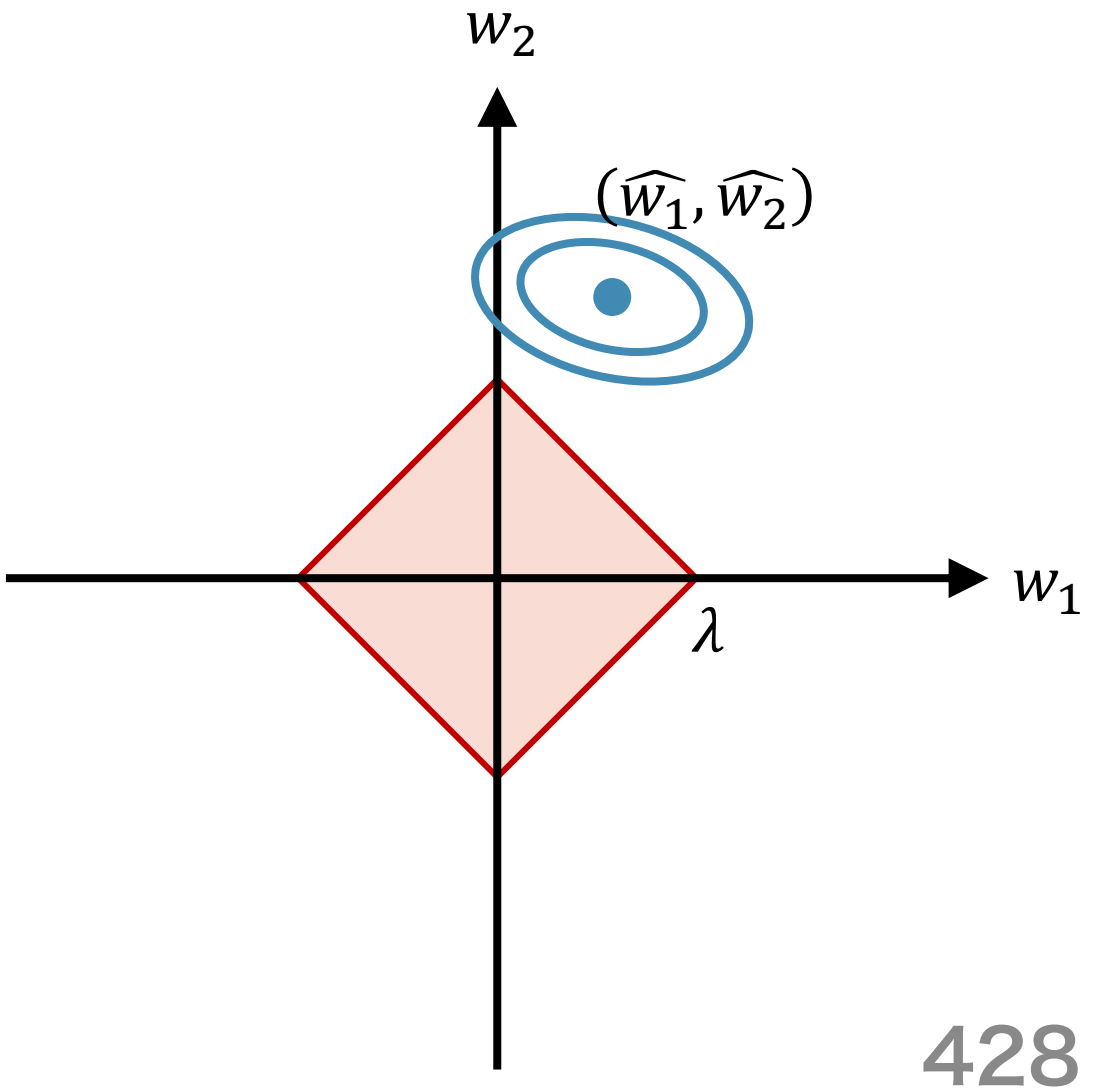
パラメータの組み合わせが赤枠を含む内部のなかに存在するようにするためには、残差二乗和を少し大きく許容する必要がある。そこで、残差二乗和を少しだけ大きく許容する。このとき、パラメータ $(\widehat{w_1}, \widehat{w_2})$ の組み合わせ複数存在し、例えば右図のように、最適解の周りに楕円状で分布している。



LASSO

パラメータの組み合わせが赤枠を含む内部のなかに存在するようにするためには、残差二乗和を少し大きく許容する必要がある。そこで、残差二乗和を少しだけ大きく許容する。このとき、パラメータ $(\widehat{w_1}, \widehat{w_2})$ の組み合わせ複数存在し、例えば右図のように、最適解の周りに楕円状で分布している。

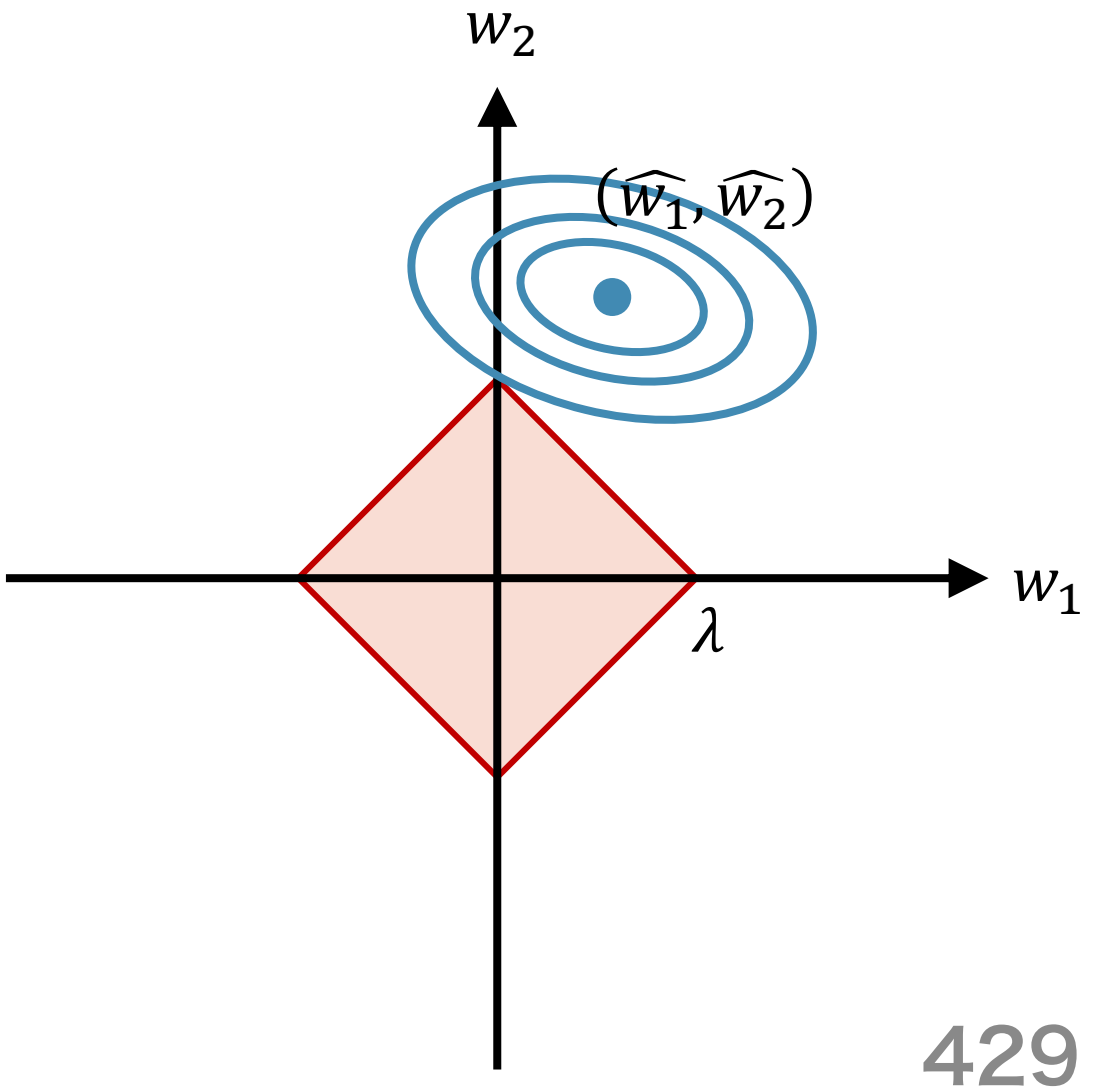
残差二乗和をもう少し大きくしてみる。



LASSO

パラメータの組み合わせが赤枠を含む内部のなかに存在するようにするためには、残差二乗和を少し大きく許容する必要がある。そこで、残差二乗和を少しだけ大きく許容する。このとき、パラメータ $(\widehat{w_1}, \widehat{w_2})$ の組み合わせ複数存在し、例えば右図のように、最適解の周りに楕円状で分布している。

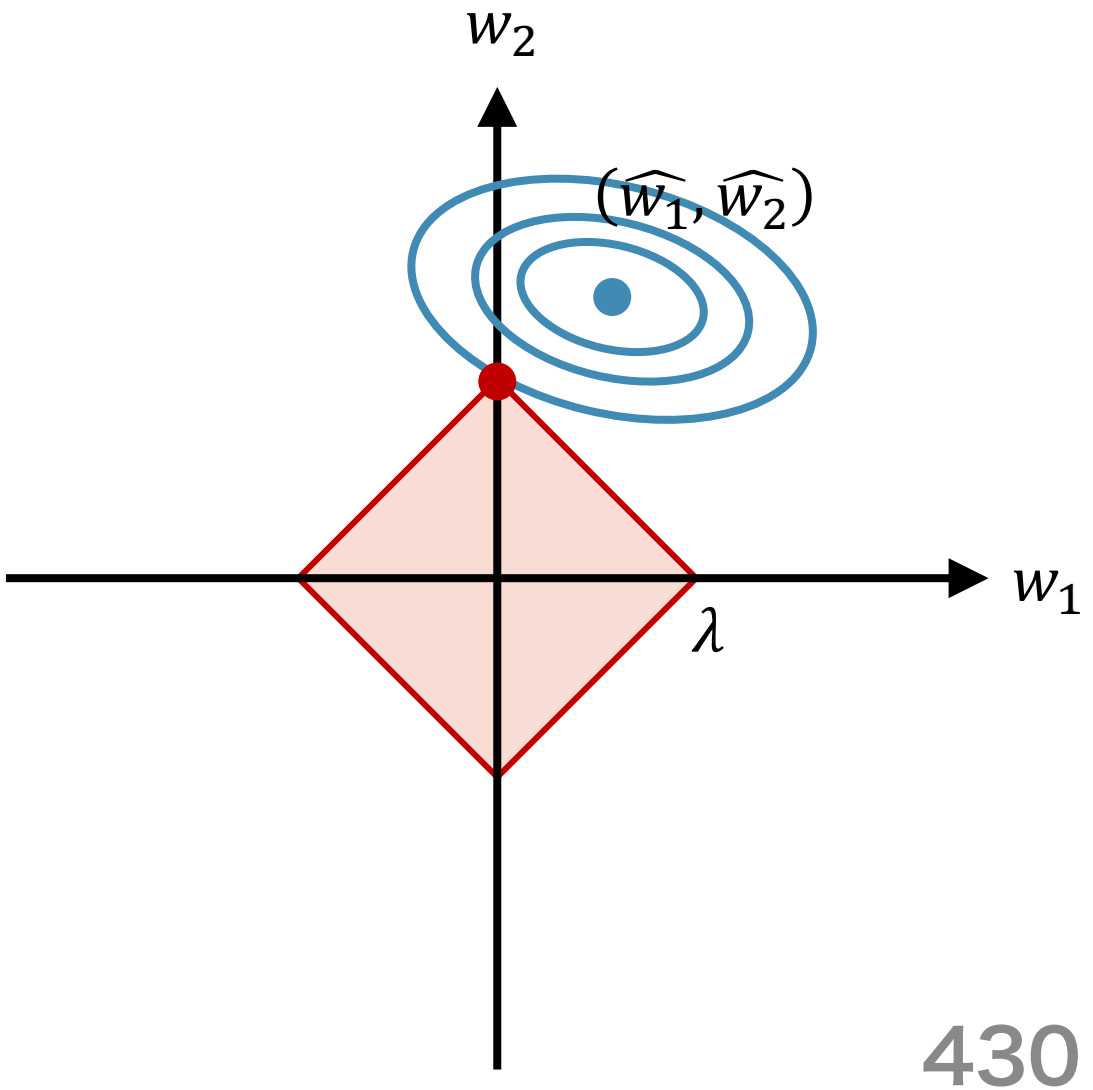
残差二乗和をもう少し大きくしてみる。さらに大きくしてみる。



LASSO

パラメータの組み合わせが赤枠を含む内部のなかに存在するようにするためには、残差二乗和を少し大きく許容する必要がある。そこで、残差二乗和を少しだけ大きく許容する。このとき、パラメータ $(\widehat{w_1}, \widehat{w_2})$ の組み合わせ複数存在し、例えば右図のように、最適解の周りに楕円状で分布している。

残差二乗和をもう少し大きくしてみる。さらに大きくしてみる。すると、$(\widehat{w_1}, \widehat{w_2})$ のとりうる値域と赤枠が点 $(\widehat{w_1} = 0, \widehat{w_2} = 1)$ で接するようになる。この時点でパラメータ推定を終了させれば、$\widehat{w_1} = 0$ および $\widehat{w_2} \neq 0$ となる解が得られる。LASSO では、このように残差二乗和を少しずつ増やしていき、最終的に制約条件を満たすパラメータの組み合わせを見つけている。

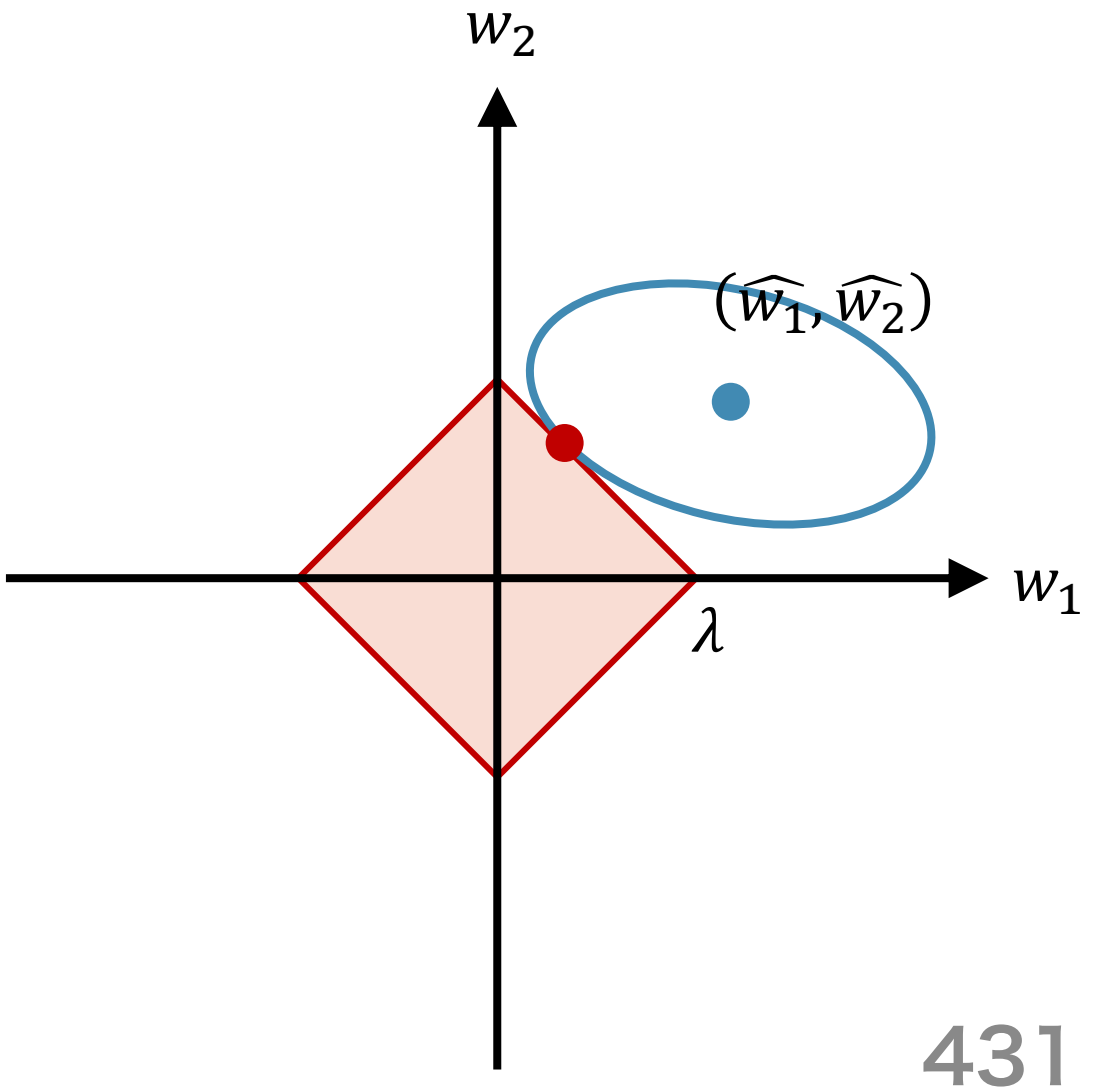


LASSO

パラメータの組み合わせが赤枠を含む内部のなかに存在するようにするためには、残差二乗和を少し大きく許容する必要がある。そこで、残差二乗和を少しだけ大きく許容する。このとき、パラメータ $(\widehat{w_1}, \widehat{w_2})$ の組み合わせ複数存在し、例えば右図のように、最適解の周りに楕円状で分布している。

残差二乗和をもう少し大きくしてみる。さらに大きくしてみる。すると、$(\widehat{w_1}, \widehat{w_2})$ のとりうる値域と赤枠が点 $(\widehat{w_1} = 0, \widehat{w_2} = 1)$ で接するようになる。この時点でパラメータ推定を終了させれば、$\widehat{w_1} = 0$ および $\widehat{w_2} \neq 0$ となる解が得られる。LASSO では、このように残差二乗和を少しずつ増やしていき、最終的に制約条件を満たすパラメータの組み合わせを見つけている。

ただし、λを固定したとき、LASSO ではいつでもスパースな解が得られるとは限らない。

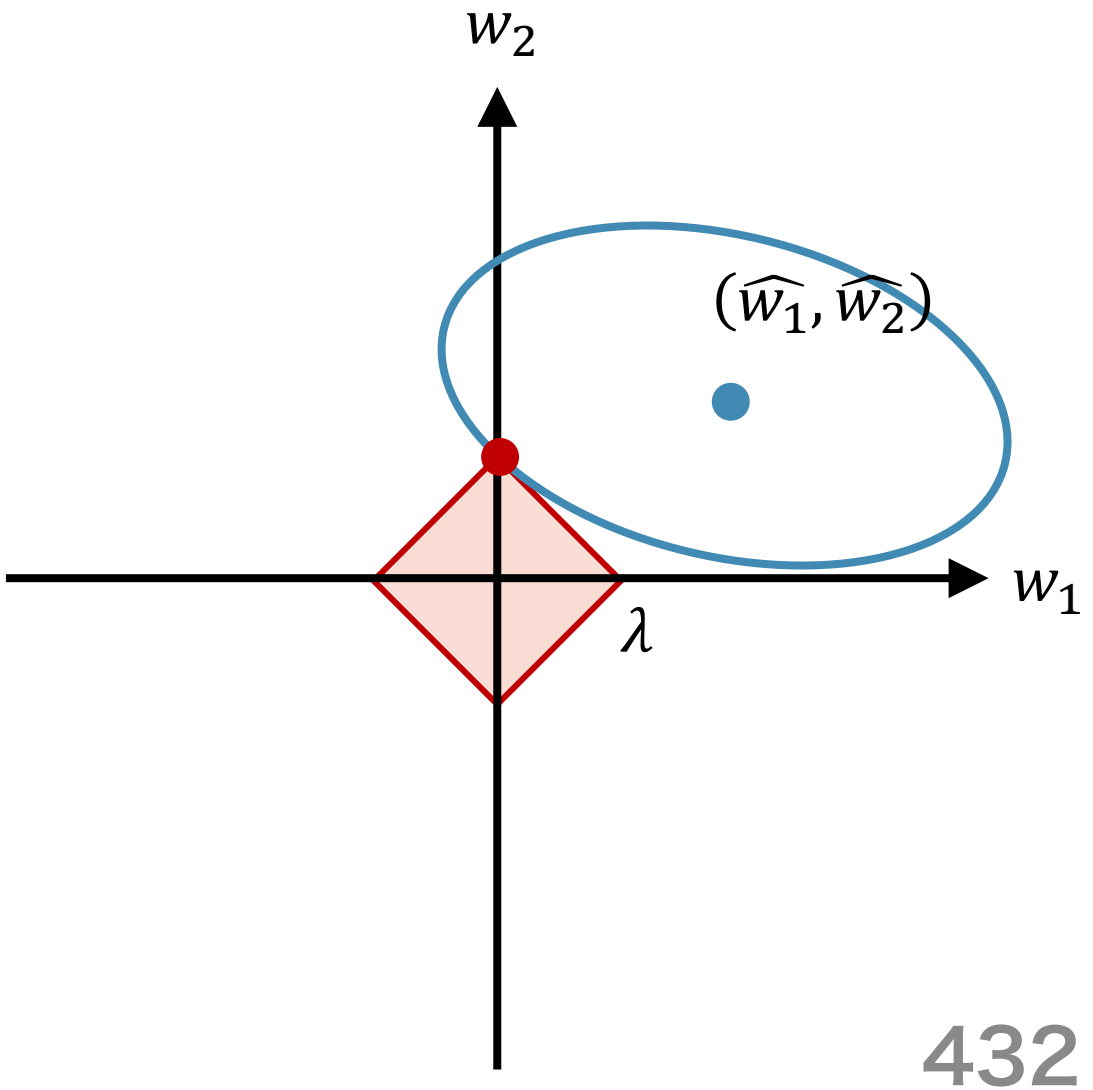


LASSO

パラメータの組み合わせが赤枠を含む内部のなかに存在するようにするためには、残差二乗和を少し大きく許容する必要がある。そこで、残差二乗和を少しだけ大きく許容する。このとき、パラメータ $(\widehat{w_1}, \widehat{w_2})$ の組み合わせ複数存在し、例えば右図のように、最適解の周りに楕円状で分布している。

残差二乗和をもう少し大きくしてみる。さらに大きくしてみる。すると、$(\widehat{w_1}, \widehat{w_2})$ のとりうる値域と赤枠が点 $(\widehat{w_1} = 0, \widehat{w_2} = 1)$ で接するようになる。この時点でパラメータ推定を終了させれば、$\widehat{w_1} = 0$ および $\widehat{w_2} \neq 0$ となる解が得られる。LASSO では、このように残差二乗和を少しずつ増やしていき、最終的に制約条件を満たすパラメータの組み合わせを見つけている。

ただし、λ を固定したとき、LASSO ではいつでもスパースな解が得られるとは限らない。λ の値を調整することでスパースな解が得られる。



Ridge

Ridge 回帰におけるパラメータ推定は、次のように書ける。

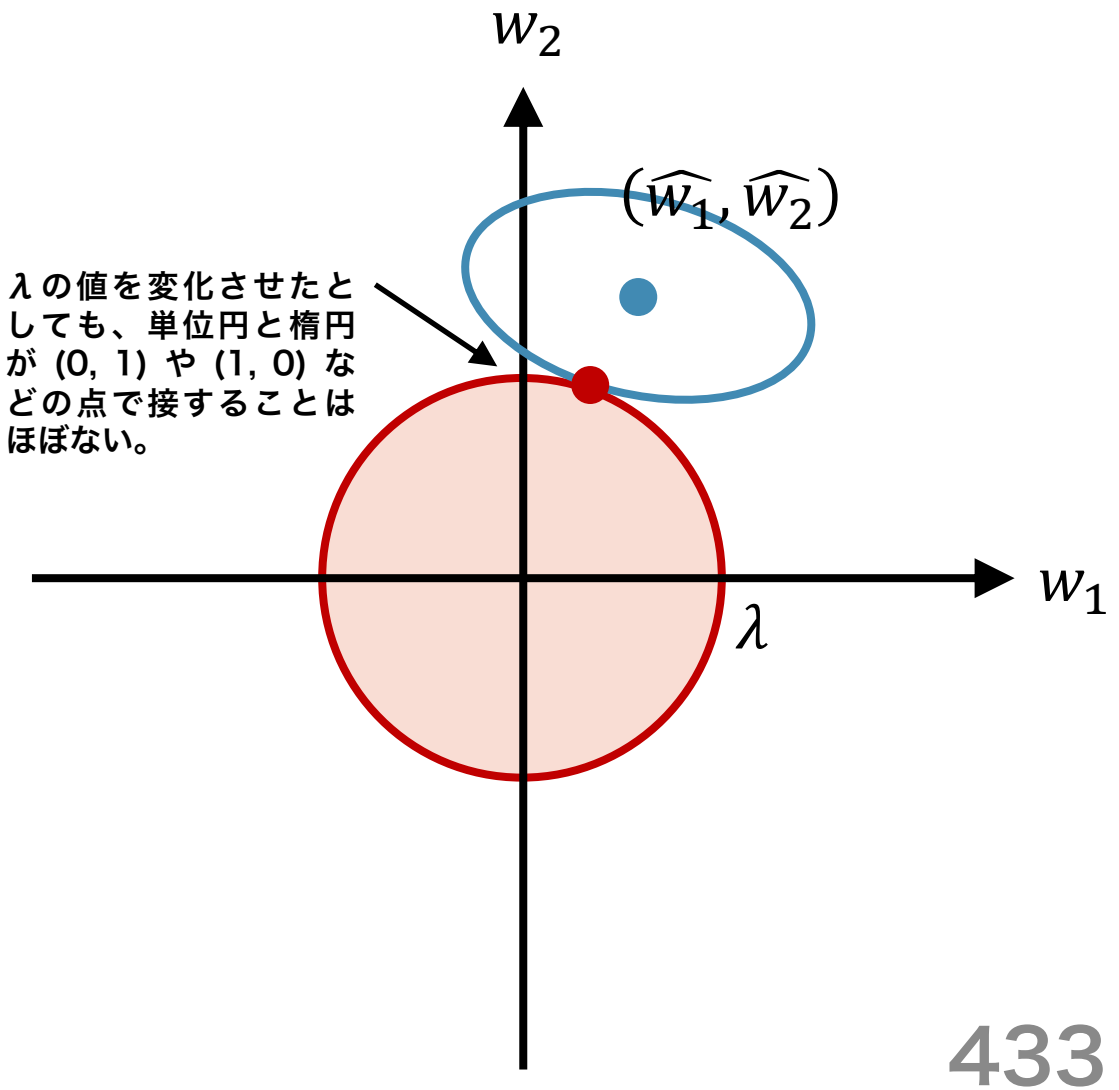
$$loss(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{w}^t\boldsymbol{x}\|_2^2$$

$$\text{subject to } \|\boldsymbol{w}\|_2^2 \leq \lambda$$

制約条件（罰則項）を展開すると次のようになる。

$$\|\boldsymbol{w}\|_2^2 = w_1^2 + w_2^2 \leq \lambda$$

右図のように、Ridge の場合は、LASSO とは異なりスパースとなる解を得るのが非常に難しいことがわかる。



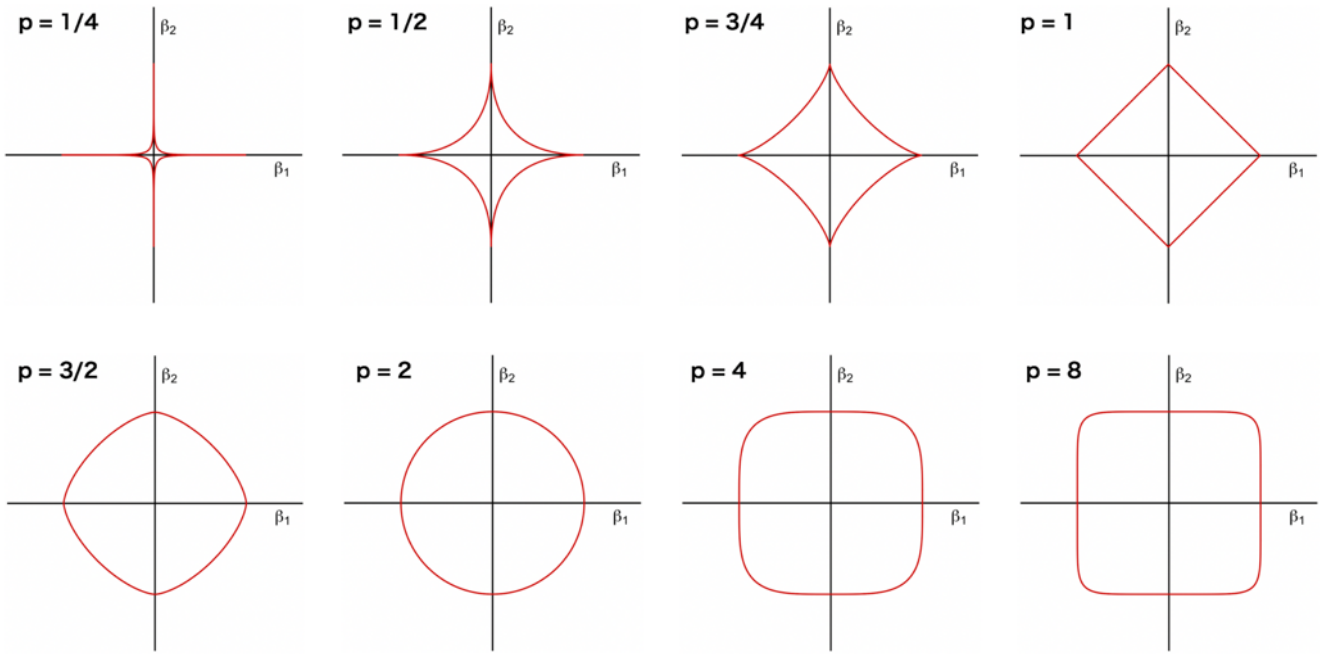
Bridge 回帰

回帰分析において、罰則項として p 次のノルムを加えてパラメータ推定を行うう場合を Bridge 回帰とよぶ。LASSO 回帰および Ridge 回帰は、Bridge 回帰の特別な場合として捉えることができる。

- p > 1 のとき、値の小さいパラメータをさらに小さくするよりも、値の大きいパラメータを小さくした方が罰則が小さいので、個々のパラメータがゼロになりにくい
- 0 < p < 1 のとき、どのパラメータを小さくしても、罰則の減少は同じなので、個々のパラメータがゼロになりやすくなる
  - ただし、罰則項が非凸関数になるため計算が困難である
- スパース推定として p = 1 が一般的に使われている

Bridge

$$loss(\boldsymbol{w}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{w}^t\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{w}\|_p^p$$



LASSO

LASSO は単純で実装しやすいアルゴリズムである。そのため、様々な解析目的に応じて LASSO が改良されている。

- Fused LASSO
- Group LASSO
- Overlapping group LASSO
- Cluster LASSO
- OSCAR

Fused LASSO

LASSO は単純で実装しやすいアルゴリズムである。そのため、様々な解析目的に応じて LASSO が改良されている。

- ○ Fused LASSO
- ○ Group LASSO
- ○ Overlapping group LASSO
- ○ Cluster LASSO
- ○ OSCAR

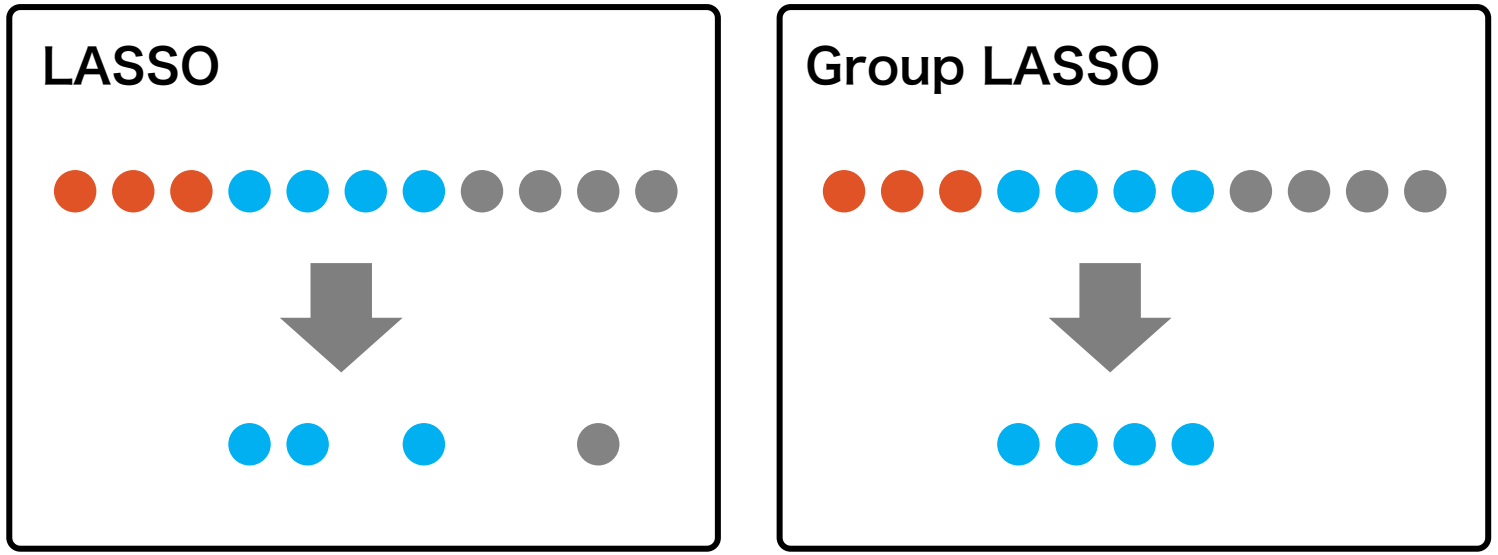
Fused LASSO は、隣接した説明変数同士の係数を同じ値になるように束縛項を加えた LASSO である。説明変数に順序があり、そして、ある説明変数に着目したとき、その 1 つ前および 1 つ後の説明変数も同じ値にしたい場合に fused LASSO を利用する。時系列データや画像データなのような、変数に順序がある系列データに適用できる。

Group LASSO

LASSO は単純で実装しやすいアルゴリズムである。そのため、様々な解析目的に応じて LASSO が改良されている。

- ○ Fused LASSO
- ◎ Group LASSO
- ○ Overlapping group LASSO
- ○ Cluster LASSO
- ○ OSCAR

説明変数の属性に基づいていくつかのグループに分けられるとき、変数選択においてグループ毎に選択したい場合に group LASSO を使う。

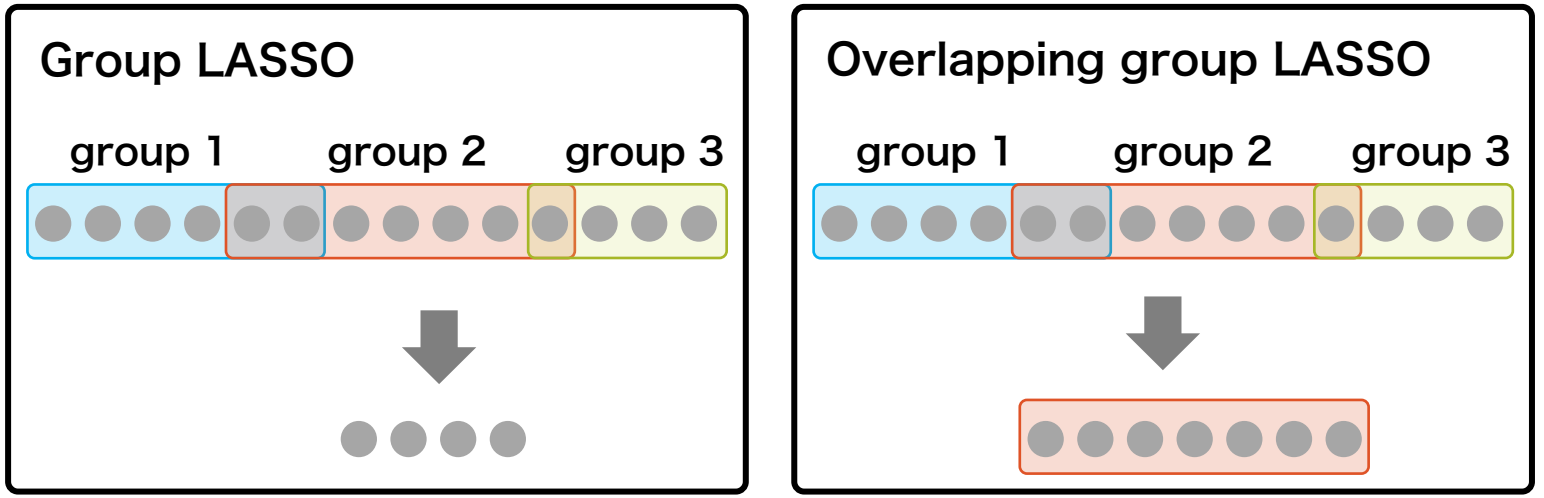


Overlapping group LASSO

LASSO は単純で実装しやすいアルゴリズムである。そのため、様々な解析目的に応じて LASSO が改良されている。

- ○ Fused LASSO
- ○ Group LASSO
- ◎ Overlapping group LASSO
- ○ Cluster LASSO
- ○ OSCAR

説明変数の属性に基づいていくつかのグループに分けられ、変数選択においてグループ毎に選択したい場合にグルーピング効果を取り入れた LASSO を使用する。とくに、1 つの変数が複数のグループに分かれる時は、overlapping group LASSO が有効である。



Cluster LASSO

LASSO は単純で実装しやすいアルゴリズムである。そのため、様々な解析目的に応じて LASSO が改良されている。

- ○ Fused LASSO
- ○ Group LASSO
- ○ Overlapping group LASSO
- ◯ Cluster LASSO
- ○ OSCAR

fused LASSO

- 隣り合う 2 個の説明変数の係数の差を罰則項

cluster LASSO

- 2 個の説明変数の係数の差をすべての組み合わせに対して求めたものを罰則項
- 係数に基づくクラスタリングが可能

OSCAR

octagonal shrinkage and clustering algorithm for regression

LASSO は単純で実装しやすいアルゴリズムである。そのため、様々な解析目的に応じて LASSO が改良されている。

- ○ Fused LASSO
- ○ Group LASSO
- ○ Overlapping group LASSO
- ○ Cluster LASSO
- ◯ OSCAR

fused LASSO

- 隣り合う 2 個の説明変数の係数の差を罰則項

cluster LASSO

- 2 個の説明変数の係数の差をすべての組み合わせに対して求めたものを罰則項
- 係数に基づくクラスタリングが可能

cluster LASSO

- 2 個の説明変数の係数の差の絶対値をすべての組み合わせに対して求めたものを罰則項
- 係数に基づくクラスタリングが可能
- 負の相関でも同じ係数になる。

回帰分析

- 線形回帰
- スパース回帰
- ロバスト推定

RANSAC

random sampling consensus

RANSAC は、訓練データから「正常値」をサンプリングし、その正常値で回帰モデルを構築するアルゴリズムである。RANSAC は外れ値の影響を受けにくいが、未知のデータ予測にプラスに働く保証はない。

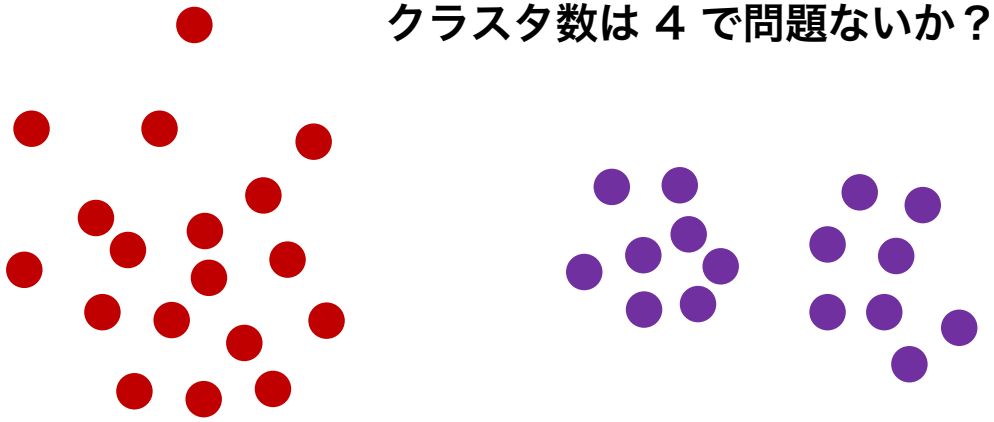
1. 訓練データからランダムに n 個のサンプルを抽出し、「正常値」とする。正常値で学習を進める。
2. 訓練データのうち手順 1 で抽出されなかったデータすべてを学習済みモデルに入力し、予測を行う。それらの予測値が、許容範囲内に入っているサンプルを正常値に追加する。
3. 正常値で再学習を行う。
4. モデル評価を行う。
5. 性能が一定以上に達したとき、あるいは一定回数を繰り返したとき終了する。そうでなければ、手順 1 に戻る。

クラスタリング

- プロトタイプベースクラスタリング
- 階層的クラスタリング
- 密度ベースクラスタリング

クラスタリング

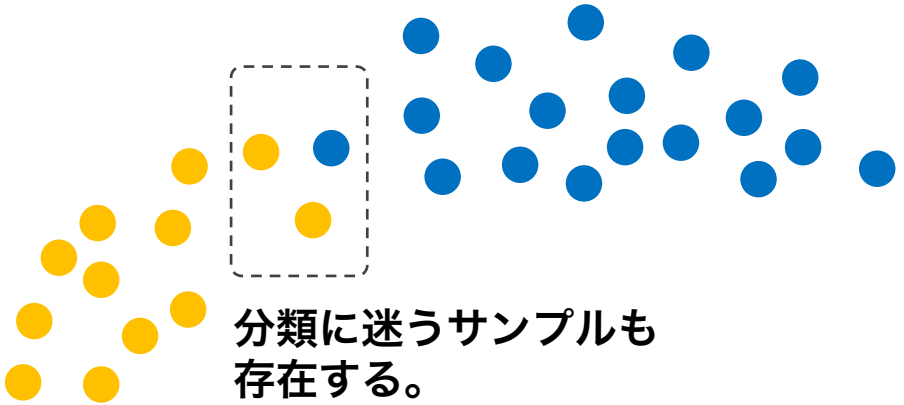
クラスタリングは、データの類似性を利用して、データ全体をいくつかのクラスタに分割する教師なし学習の一つである。クラスタリングはプロトタイプベースな手法、階層的な手法や密度ベースな手法などが使われているが、それぞれが得意としているデータ構造が異なっている。そのため、データを正確にクラスタリングするためには、様々なアルゴリズムを試した上で、そのデータ構造に適したクラスタリングアルゴリズムを使用する必要がある。



正確なクラスタリングの用件

均質性　同一クラスタに属するサンプル同士の性質が互いに似ていること。

分離性　異なるクラスタに属するサンプル同士の性質が互いに異なる。

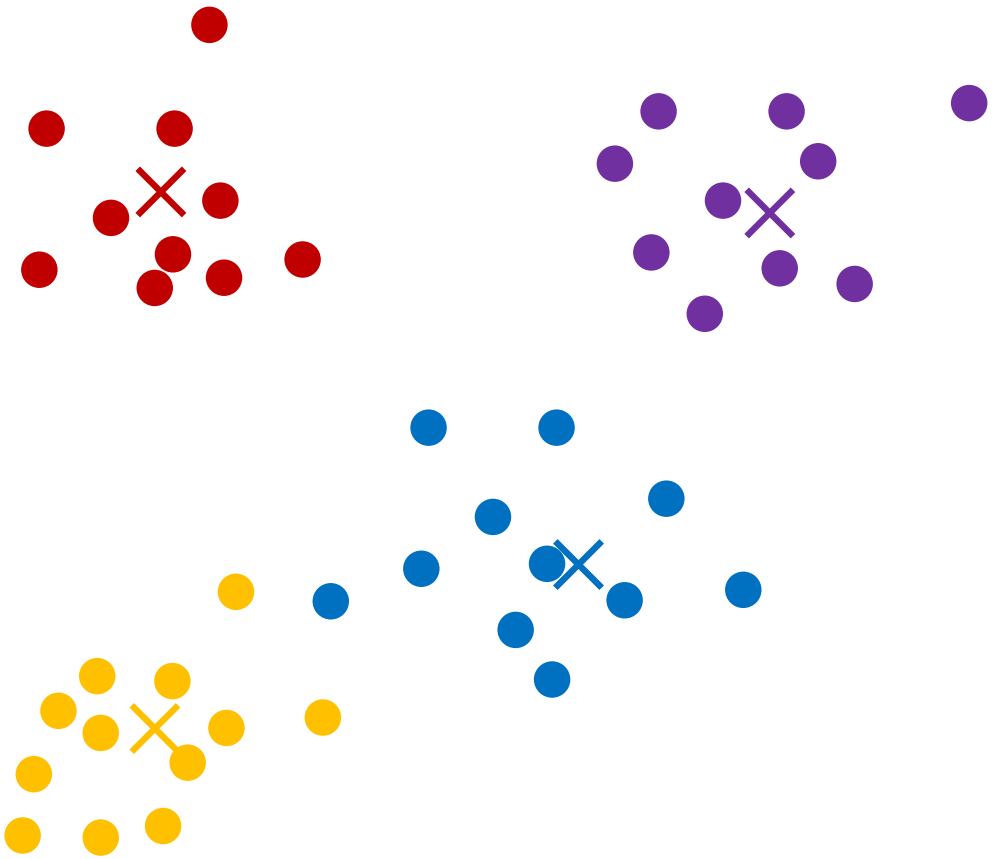


クラスタリング

- プロトタイプベースクラスタリング
- 階層的クラスタリング
- 密度ベースクラスタリング

プロトタイプベースクラスタリング

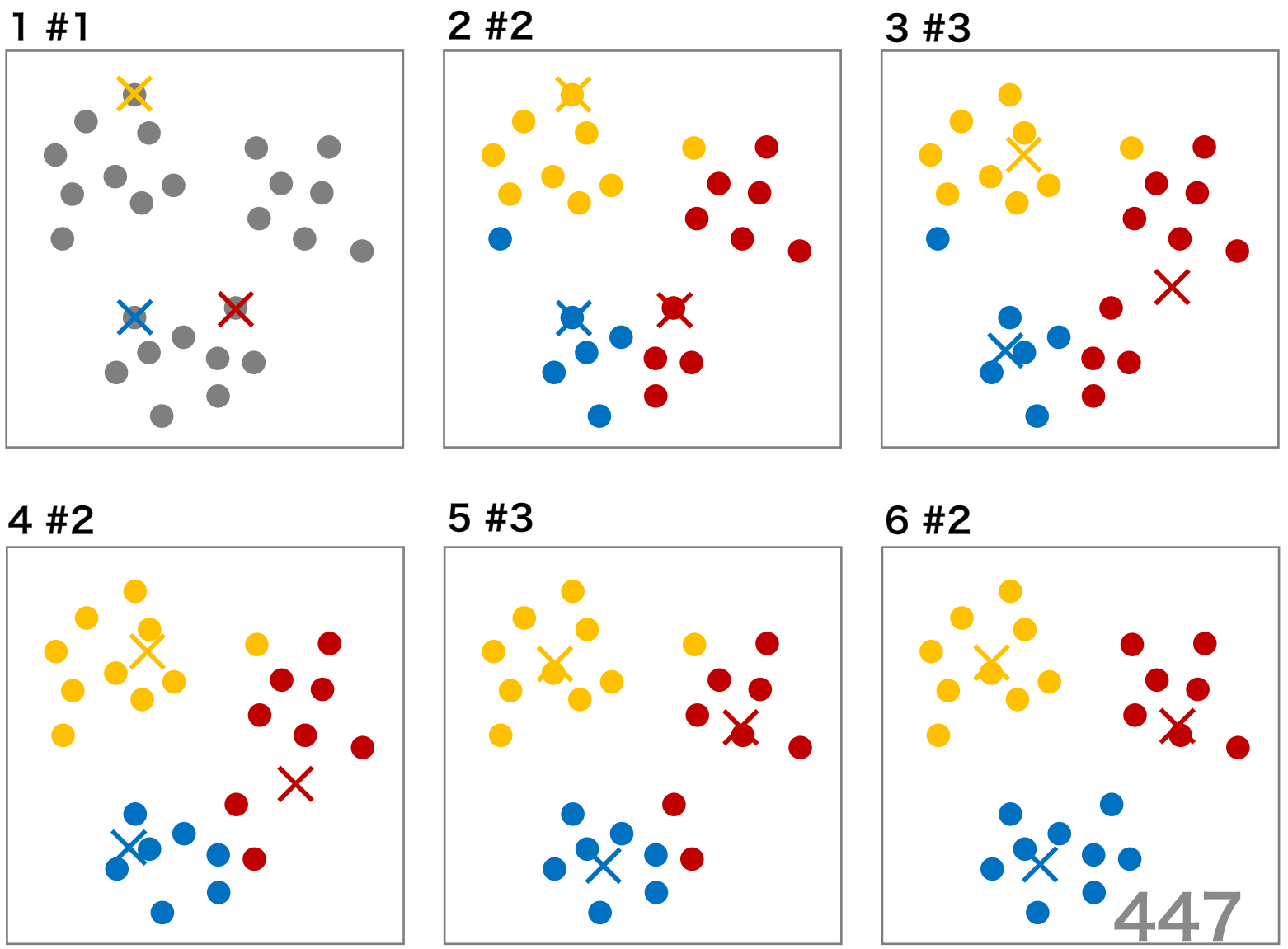
プロトタイプベースのクラスタリングは、特徴空間上に k 個のセントロイドを指定して、サンプル点とセントロイドの距離に基づいてクラスタの割り当てを行う方法である。クラスタリングにあたり、あらかじめ k を与える必要があり、クラスタリング結果が k の値に影響されれる。代表的な手法には、k-means などがある。



k-means

k-means は広く使われているクラスタリングアルゴリズムの一つである。アルゴリズムが簡単で、計算が高速であるため、ビッグデータのクラスタリングに広く使われている。

1. サンプルからランダムに k 個を選び、これらをセントロイドとする。
2. サンプルとセントロイドの距離を計算し、各サンプルを最も近いセントロイドに割り当てる。
3. 各々のセントロイドに割り当てられたサンプルの重心をそのセントロイドにする。
4. 割り当てが変化しなくなったとき、あるいは繰り返し回数に達したとき、終了する。そうでなければ、手順 1-3 を繰り返す。

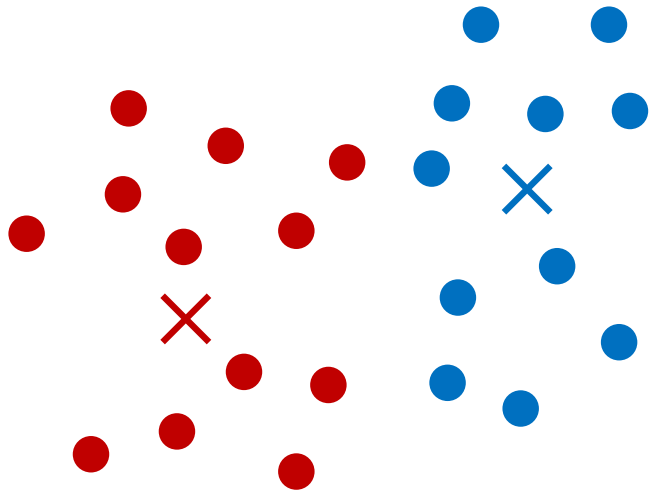


k-means

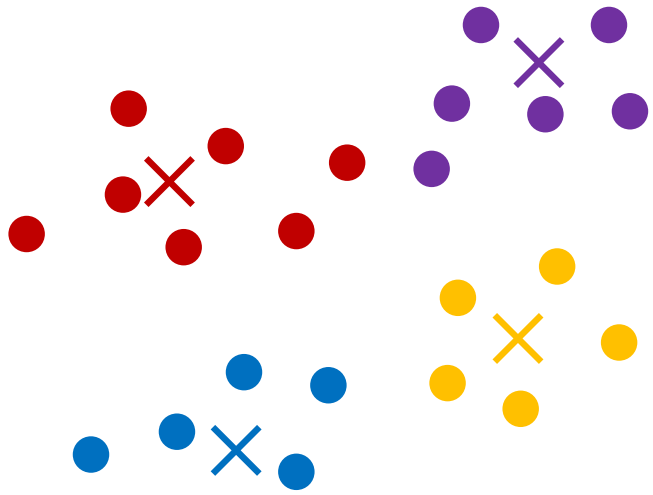
k-means は実装が簡単で、計算が高速である。しかし、クラスタリング前にクラスタ数を決めなければならなかったり、データの構造が球状構造を仮定したりするなどの制限も多い。

- クラスタリングを行う前にクラスタ数 k を指定する必要がある。クラスタリング結果は k の値に影響される。
- k 個のセントロイドを選ぶとき、セントロイドの選び方によってクラスタリング結果が異なる。
- データの構造が球状構造のときに最も効果的である。
- 各クラスタに含まれるサンプル数がほぼ同じ時に最も効果的である。

k = 2



k = 4

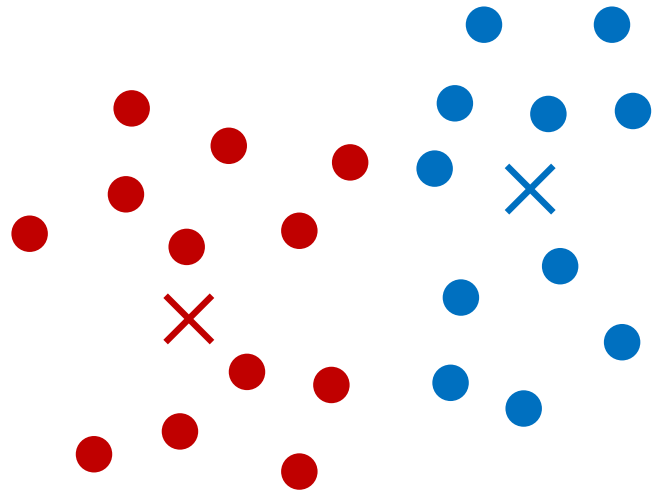


k-means

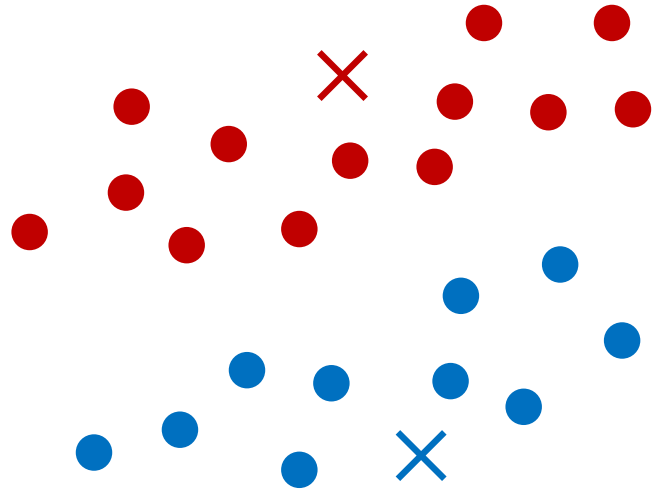
k-means は実装が簡単で、計算が高速である。しかし、クラスタリング前にクラスタ数を決めなければならなかったり、データの構造が球状構造を仮定したりするなどの制限も多い。

- ○ クラスタリングを行う前にクラスタ数 k を指定する必要がある。クラスタリング結果は k の値に影響される。
- ◯ k 個のセントロイドを選ぶとき、セントロイドの選び方によってクラスタリング結果が異なる。
- ○ データの構造が球状構造のときに最も効果的である。
- ○ 各クラスタに含まれるサンプル数がほぼ同じ時に最も効果的である。

k = 2



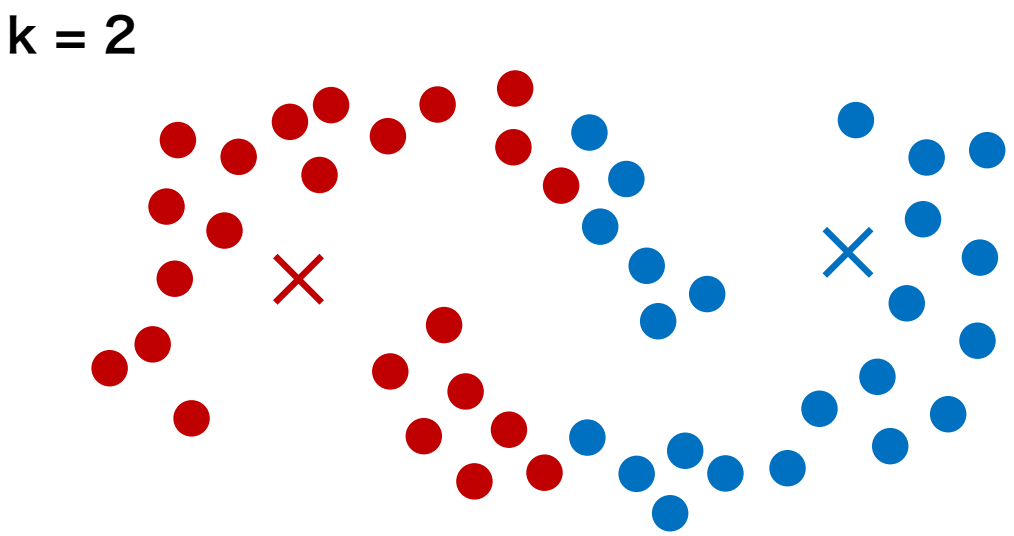
k = 2

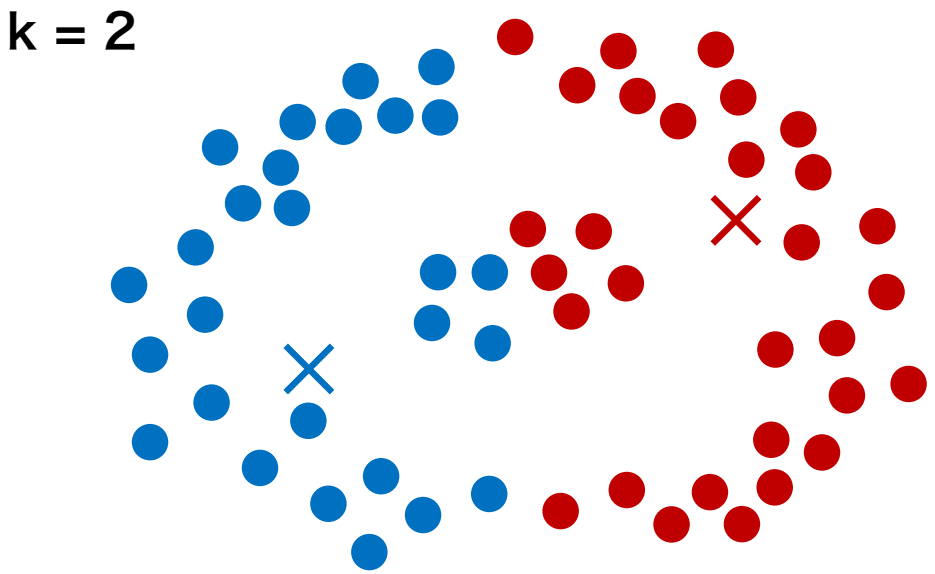


k-means

k-means は実装が簡単で、計算が高速である。しかし、クラスタリング前にクラスタ数を決めなければならなかったり、データの構造が球状構造を仮定したりするなどの制限も多い。

- ○ クラスタリングを行う前にクラスタ数 k を指定する必要がある。クラスタリング結果は k の値に影響される。
- ○ k 個のセントロイドを選ぶとき、セントロイドの選び方によってクラスタリング結果が異なる。
- ● データの構造が球状構造のときに最も効果的である。
- ○ 各クラスタに含まれるサンプル数がほぼ同じ時に最も効果的である。

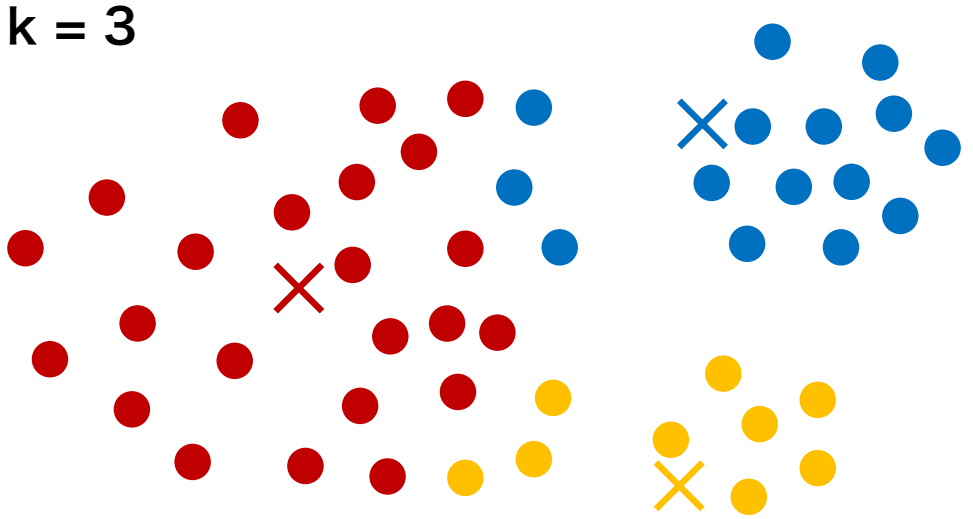


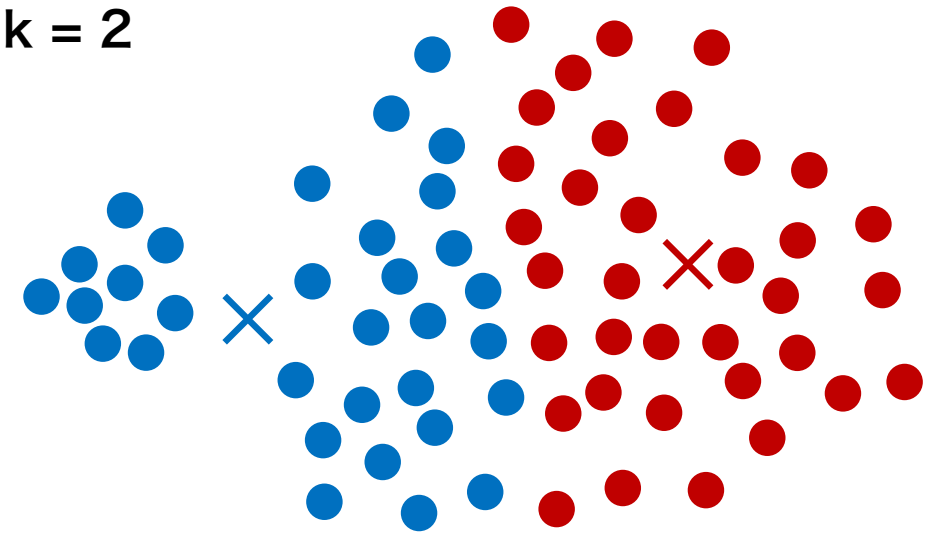


k-means

k-means は実装が簡単で、計算が高速である。しかし、クラスタリング前にクラスタ数を決めなければならなかったり、データの構造が球状構造を仮定したりするなどの制限も多い。

- ○ クラスタリングを行う前にクラスタ数 k を指定する必要がある。クラスタリング結果は k の値に影響される。
- ○ k 個のセントロイドを選ぶとき、セントロイドの選び方によってクラスタリング結果が異なる。
- ○ データの構造が球状構造のときに最も効果的である。
- ○ 各クラスタに含まれるサンプル数がほぼ同じ時に最も効果的である。





k-means++

k-means は実装が簡単で、計算が高速だが、セントロイドの初期値に依存するという問題点がある。セントロイドの選び方によってクラスタリング性能が大きく異なってくる。このセントロイドの初期値の選び方を改善したのが、k-means++ である。kmeans++ では複数のセントロイドを、互いにできるだけ離れるようにして選ぶアプローチをとっている。

1. セントロイドを格納するための空のデータセット M を用意する。
2. 入力サンプルからランダムに 1 つを選び $\mu^{(i)}$ とし、M に加える。
3. M に含まれていない $x^{(j)}$ ごとに、データセット M のすべての点（セントロイド）と距離を計算し、そのうち最小値を $d(x^{(j)}, M)$ とおく。
4. 確率 $p(\mu^{(i+1)})$ で $\mu^{(i+1)}$ を次のセントロイドとして選ぶ。

$$p(\boldsymbol{\mu}^{(i+1)}) = \frac{d\left(\boldsymbol{\mu}^{(i+1)}, \boldsymbol{M}\right)^2}{\sum_j d(\boldsymbol{x}^{(j)}, \boldsymbol{M})^2}$$

5. セントロイドが k 個になるまで手順 3-4 を繰り返す。
6. k-means アルゴリズムでクラスタリングを行う。

Fuzzy C-means (soft k-means)

k-means は 1 つのサンプルを 1 つのクラスタに割り当てるハードクラスタリングの一つである。これに対して、1 つのサンプルを 2 つ以上のクラスタに割り当てるクラスタリングをソフトクラスタリングという。代表的なソフトクラスタリングアルゴリズムは Fuzzy C-means (FCM, soft k-means) などがある。

k-means において、各サンプルのセントロイドをベクトルで表現すると、下のように 0 と 1 からなるベクトルで表現している。これに対して、FCM では、各セントロイドに属する確率として表現している。

k-means

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \vdots \\ \mu_p \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

FCM

$$\begin{pmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \\ \vdots \\ \mu_p \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.77 \\ 0.02 \\ \vdots \\ 0.06 \end{pmatrix}$$

Fuzzy C-means (soft k-means)

1. k のセントロイドをランダムに選ぶ。
2. 各サンプル点に対して、各セントロイドに所属確率を示すベクトル（クラスメンバーシップ確率）をランダムな値で初期化する。以下、サンプル $x_{(j)}$ がセントロイド $\mu^{(i)}$ に属する確率を $w^{(j,i)}$ とおく。
3. k 個のクラスタのセントロイドを次式に基づいて計算する。ただし、m はファジー係数である。

$$\boldsymbol{\mu}^{(i)} = \frac{\sum_{j=1}^{n} w^{(j,i)^m} \boldsymbol{x}^{(j)}}{\sum_{j=1}^{n} w^{(j,i)^m}}$$

4. 各サンプル点のクラスタメンバーショップ確率 $w^{(j,i)}$ を更新する。

$$w^{(j,i)} = \left[\sum_{l=1}^{k} \left(\frac{\left\| \boldsymbol{x}^{(j)} - \boldsymbol{\mu}^{(i)} \right\|^2}{\left\| \boldsymbol{x}^{(j)} - \boldsymbol{\mu}^{(l)} \right\|^2} \right)^{\frac{1}{m-1}} \right]^{-1}$$

5. クラスタメンバーショップ確率が変化しなくなるまで、あるいは最大繰り返し回数に達するまで、手順 3〜4 を繰り返す。

クラスタ数 k

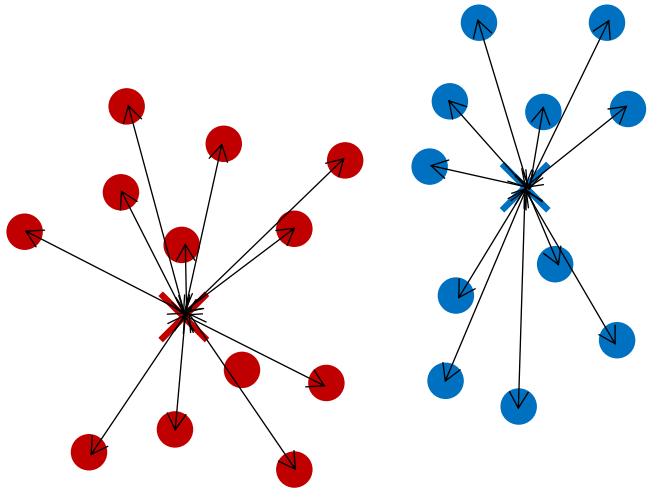
エルボー法

クラスタリングの性能を測る指標として、クラスタ内誤差平方和 SSE が使われる。k が小さいとき、少ないクラスタ内の多くのサンプルが入り、サンプル同士のばらつきが大きくなりがちであるから、SSE も大きくなる。逆に、k が大きくなると、クラスタ内のサンプル数が少なくなり、各サンプルがセントロイドの近づくため、計算される SSE も小さくなる。

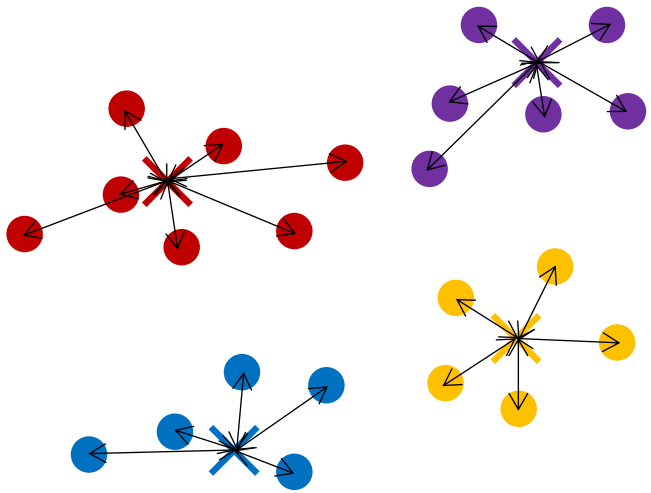
$$J = \sum_{j=1}^{n} \sum_{i=1}^{k} w^{(j,i)} \left\| x^{(j)} - \mu^{(i)} \right\|^2$$

ただし、サンプル j がセントロイド i に属するとき、w(j,i) = 1 であり、そうでなければ w(j,i) = 0 である。

k = 2



k = 4



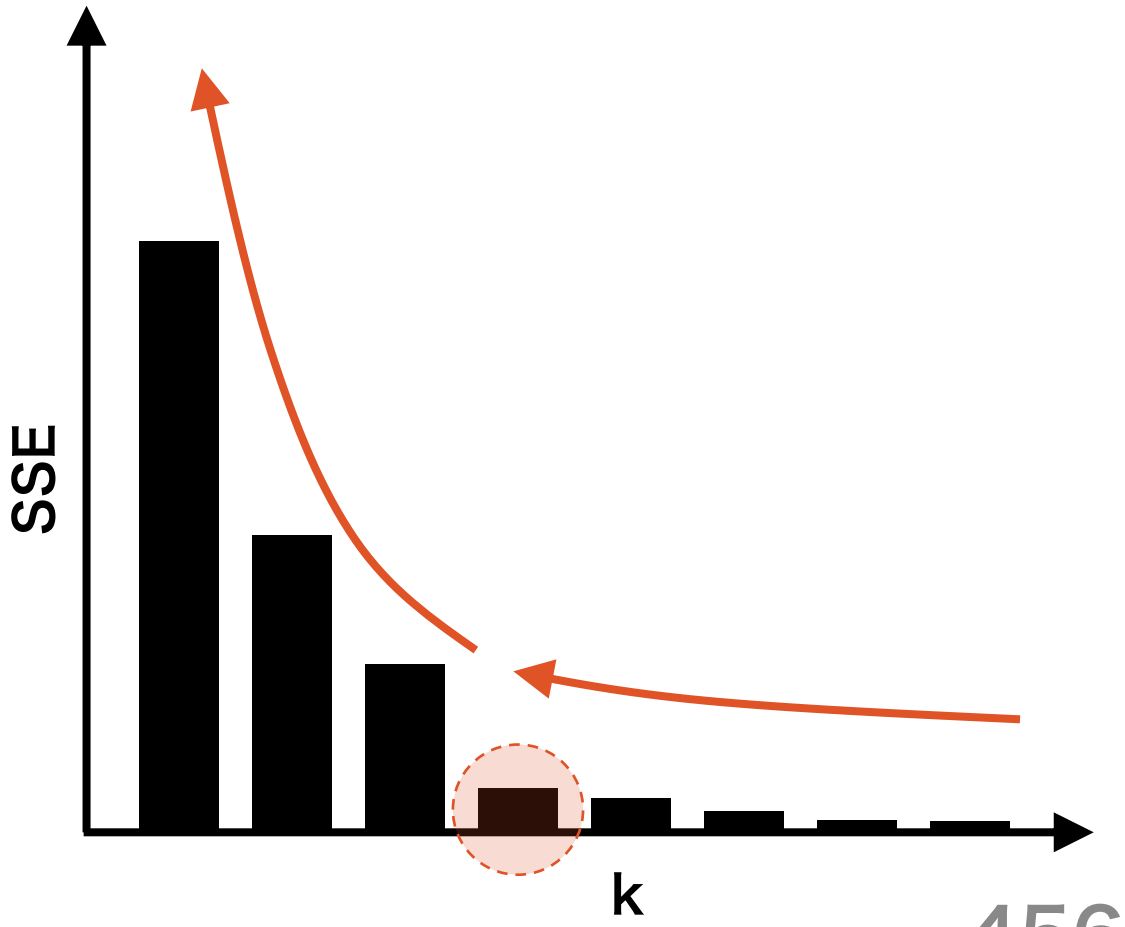
クラスタ数 k

エルボー法

クラスタリングの性能を測る指標として、クラスタ内誤差平方和 SSE が使われる。k が小さいとき、少ないクラスタ内の多くのサンプルが入り、サンプル同士のばらつきが大きくなりがちであるから、SSE も大きくなる。逆に、k が大きくなると、クラスタ内のサンプル数が少なくなり、各サンプルがセントロイドの近づくため、計算される SSE も小さくなる。

$$SSE = \sum_{j=1}^{n} \sum_{i=1}^{k} w^{(j,i)} \left\| x^{(j)} - \mu^{(i)} \right\|^2$$

ただし、サンプル j がセントロイド i に属するとき、w(j,i) = 1 であり、そうでなければ w(j,i) = 0 である。



クラスタ数 k

シルエット図

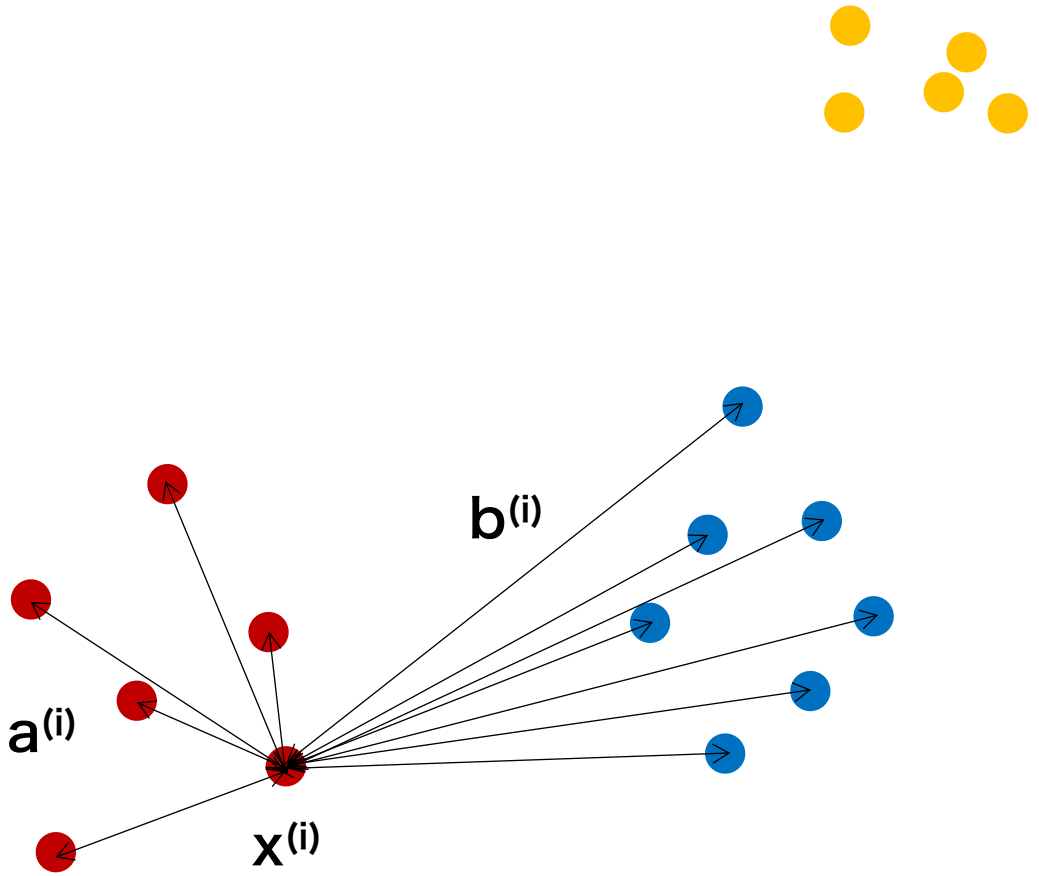
クラスタ内のサンプルがどれほど凝集されているかを測る指標として、シルエット係数がある。シルエット係数を指標として、クラスタ数 k を決めることができる。シルエット係数は次の手順で計算する。

1. サンプル $x^{(i)}$ と同一クラスタ内の他のサンプルとの平均距離を凝集度 $a^{(i)}$ とする。

2. サンプル $x^{(i)}$ と最も近くにある他クラスタの全サンプル との平均距離を乖離度 $b^{(i)}$ とする。

3. サンプル $x^{(i)}$ のシルエット係数 $s^{(i)}$ を次式に基づいて計算する。

$$s^{(i)} = \frac{b^{(i)} - a^{(i)}}{\max\{b^{(i)}, a^{(i)}\}}$$



クラスタ数 k

シルエット図

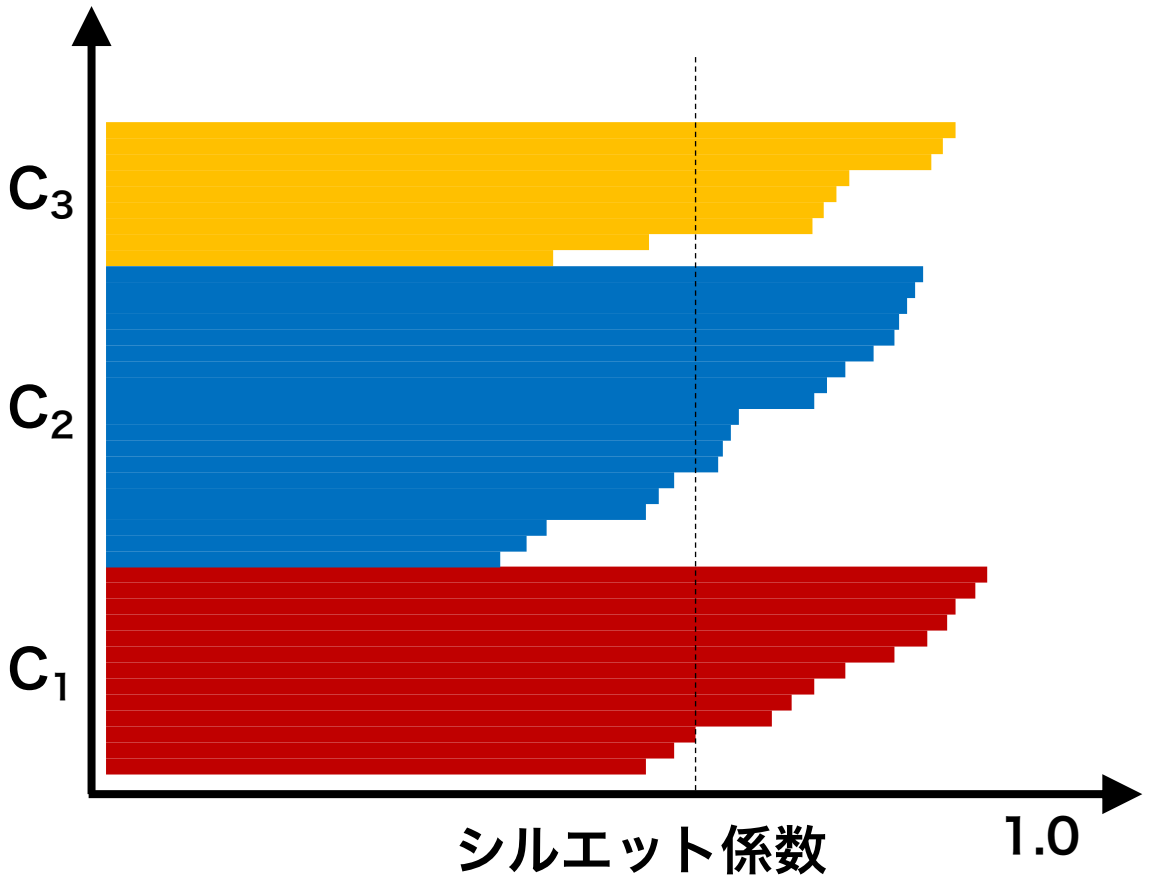
クラスタ内のサンプルがどれほど凝集されているかを測る指標として、シルエット係数がある。シルエット係数を指標として、クラスタ数 k を決めることができる。シルエット係数は次の手順で計算する。

1. サンプル $x^{(i)}$ と同一クラスタ内の他のサンプルとの平均距離を凝集度 $a^{(i)}$ とする。

2. サンプル $x^{(i)}$ と最も近くにある他クラスタの全サンプル との平均距離を乖離度 $b^{(i)}$ とする。

3. サンプル $x^{(i)}$ のシルエット係数 $s^{(i)}$ を次式に基づいて計算する。

$$s^{(i)} = \frac{b^{(i)} - a^{(i)}}{\max\{b^{(i)}, a^{(i)}\}}$$



クラスタ数 k

シルエット図

クラスタ内のサンプルがどれほど凝集されているかを測る指標として、シルエット係数がある。シルエット係数を指標として、クラスタ数 k を決めることができる。シルエット係数は次の手順で計算する。

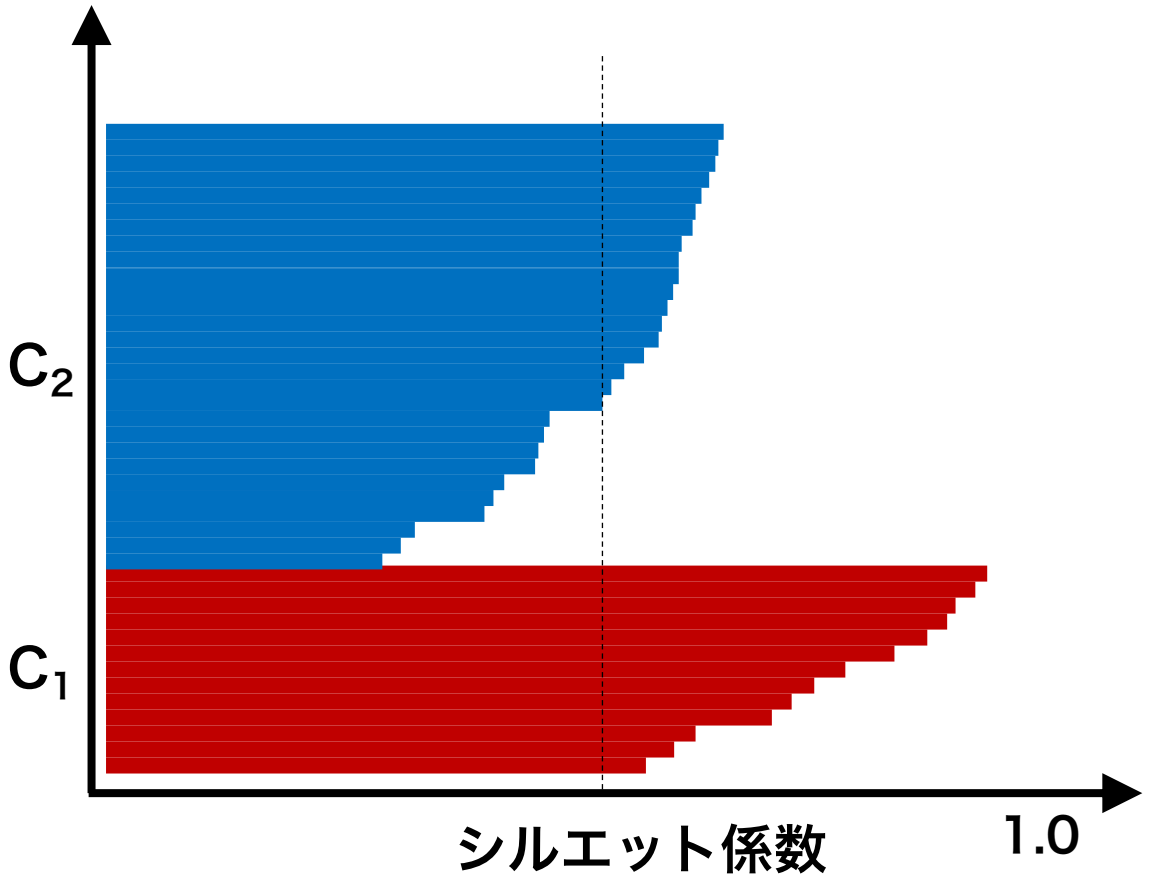
1. サンプル $x^{(i)}$ と同一クラスタ内の他のサンプルとの平均距離を凝集度 $a^{(i)}$ とする。

2. サンプル $x^{(i)}$ と最も近くにある他クラスタの全サンプル との平均距離を乖離度 $b^{(i)}$ とする。

3. サンプル $x^{(i)}$ のシルエット係数 $s^{(i)}$ を次式に基づいて計算する。

$$s^{(i)} = \frac{b^{(i)} - a^{(i)}}{\max\{b^{(i)}, a^{(i)}\}}$$

不適な k の例　各クラスタのシルエット図の幅と長さがばらつく。



クラスタリング

- プロトタイプベースクラスタリング
- 階層的クラスタリング
- 密度ベースクラスタリング

階層的クラスタリング

階層的クラスタリングは、似たもの同士を逐次的に加えていくアルゴリズムである。クラスタリング前にクラス多数を指定する必要がない。その代わりに、クラスタリング結果を見て、クラスタ数を決めることができるため、意味のある分割を行うことができ、結果の解釈に役立つ。

凝集型　似たもの同士を次々に加えることで階層的なクラスタを作成する。

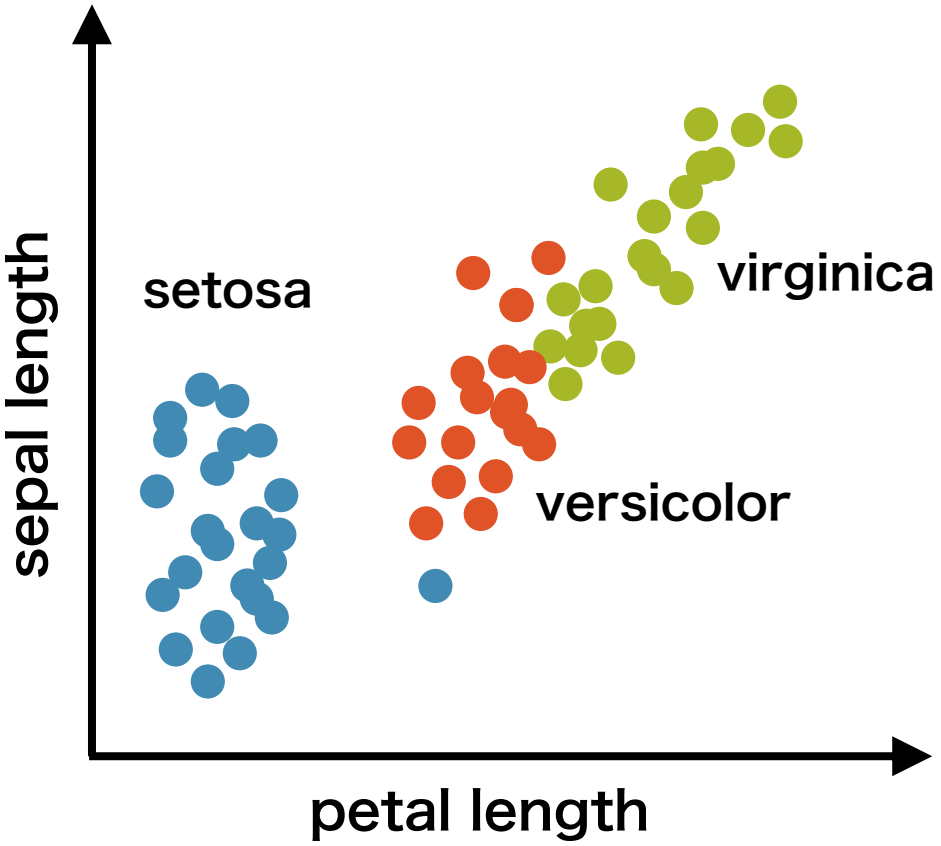
- 単連結法（最短距離法）
- 完全連結法（最長距離法）
- 平均連結法（群平均法）
- ウォード法

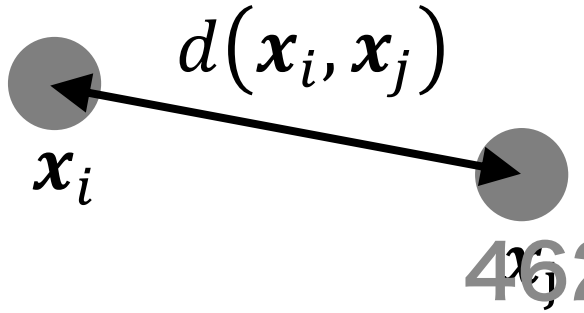
分割型　データ全体を一つのまとまりとみなして、全体を次々と分割していくことで階層的なクラスタを作成する。

- Diana 法

距離

二つのサンプルの似ているということは、二つのサンプルに共通した特徴が多い。そのため、相似性を測る指標として、サンプル同士の特徴ベクトルの距離を利用できる。





距離

ユーグリッド距離

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{\sum (x_{ik} - x_{jk})^2}$$

マハノビス距離

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \overline{\boldsymbol{x}})^t \Sigma^{-1} (\boldsymbol{x}_j - \overline{\boldsymbol{x}})}$$

マンハッタン距離

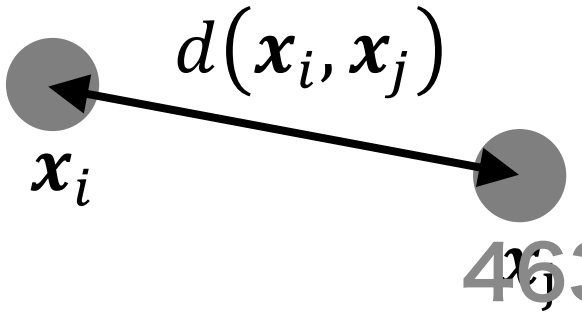
$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sum |x_{ik} - x_{jk}|$$

コサイン距離

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|}$$

ミンコフスキー距離

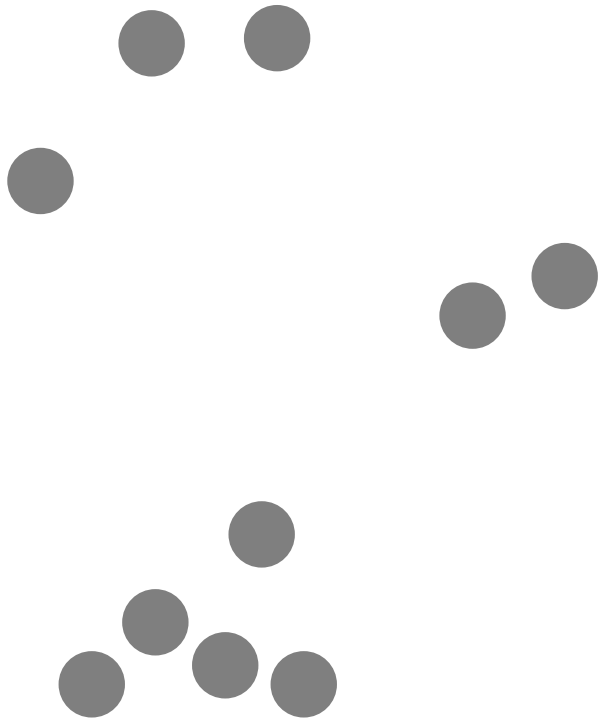
$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt[p]{\sum |x_{ik} - x_{jk}|^p}$$



階層的クラスタリング（凝集型）

階層的クラスタを構築するために、距離の計算方法を定義した上で、クラスタ間距離の計算方法も定義する必要がある。

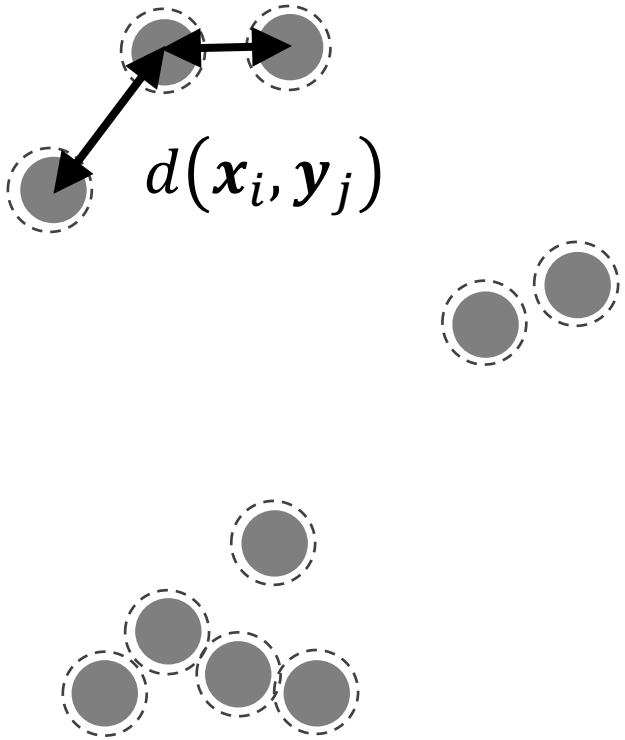
$$d(\boldsymbol{x}_i, \boldsymbol{y}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$$



階層的クラスタリング（凝集型）

階層的クラスタを構築するために、距離の計算方法を定義した上で、クラスタ間距離の計算方法も定義する必要がある。

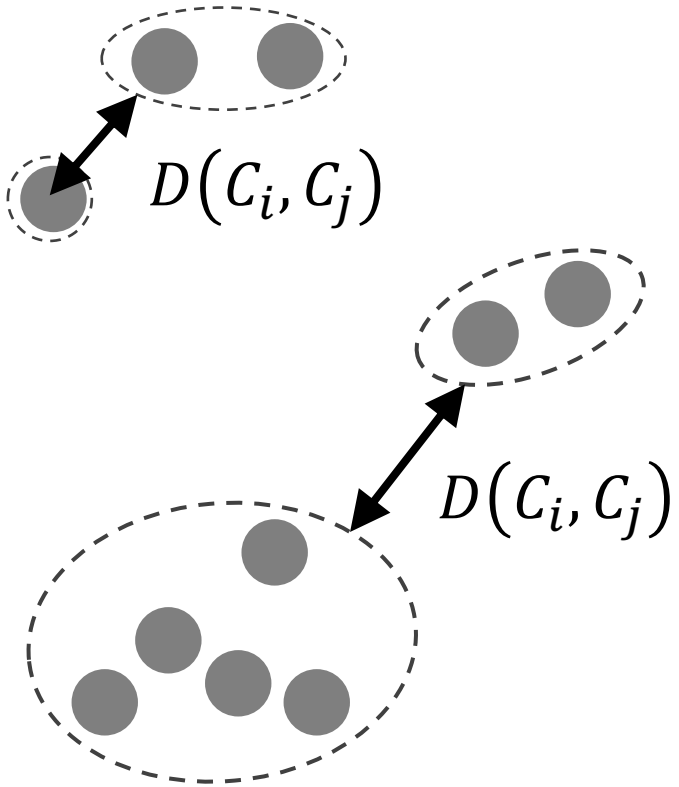
$$d(\boldsymbol{x}_i, \boldsymbol{y}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$$



階層的クラスタリング（凝集型）

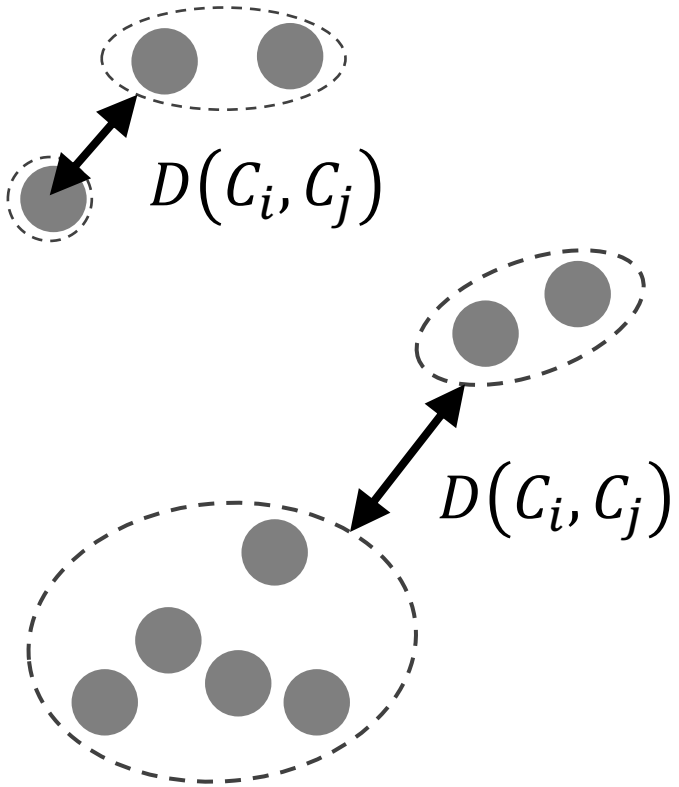
階層的クラスタを構築するために、距離の計算方法を定義した上で、クラスタ間距離の計算方法も定義する必要がある。

$$D(C_i, C_j) = ??$$



階層的クラスタリング（凝集型）

階層的クラスタを構築するために、距離の計算方法を定義した上で、クラスタ間距離の計算方法も定義する必要がある。



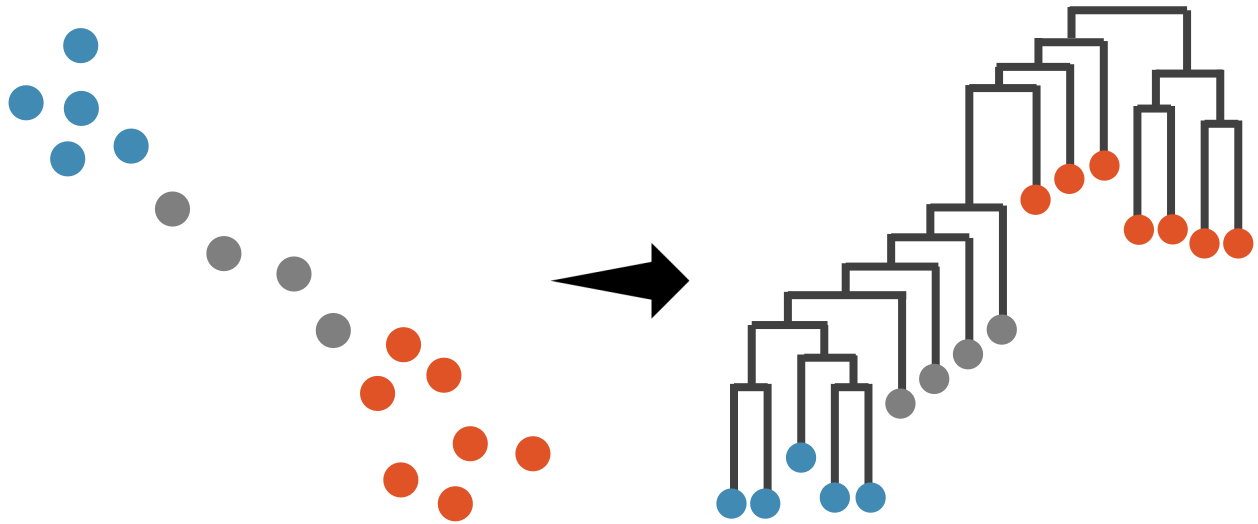
$$D(C_i, C_j) = ??$$

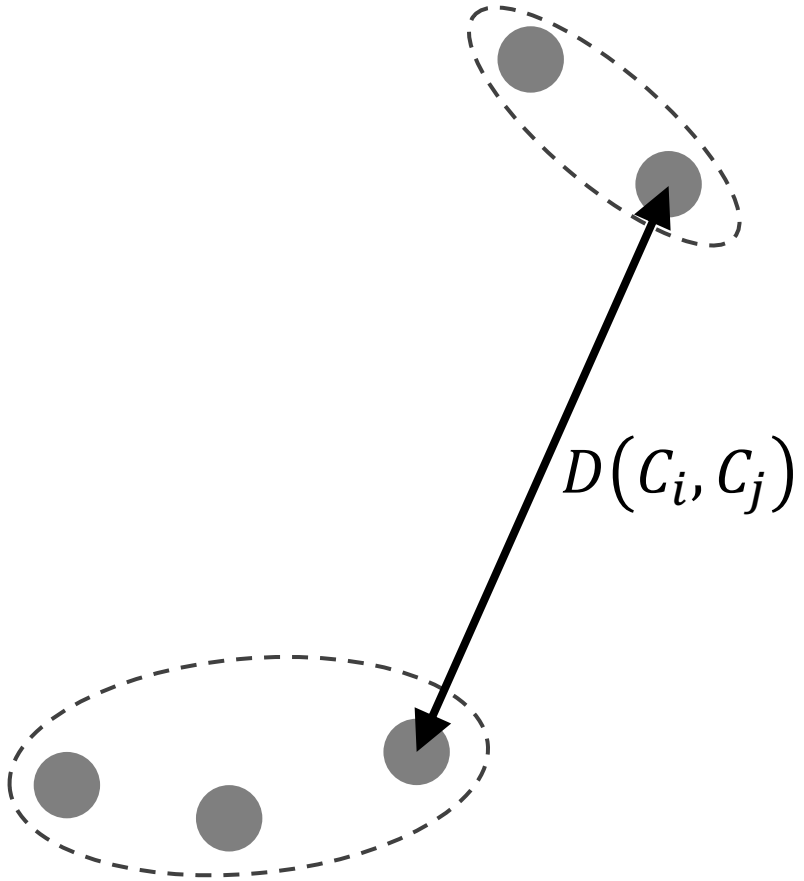
- 単連結法（最短距離法）
- 完全連結法（最長距離法）
- 平均連結法（群平均法）
- ウォード法

単連結法（最短距離法）

単連結法は、対象クラスタに含まれるサンプルのうち、距離が最も近いサンプル同士の距離をクラスタ間距離としている。計算が簡単で、ドーナツ構造のデータも正しくクラスタリングできる。しかし、外れ値に弱く、横に伸びる枝の多い樹形図が得られやすい。

$$D(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} \{d(x_i, x_j)\}$$

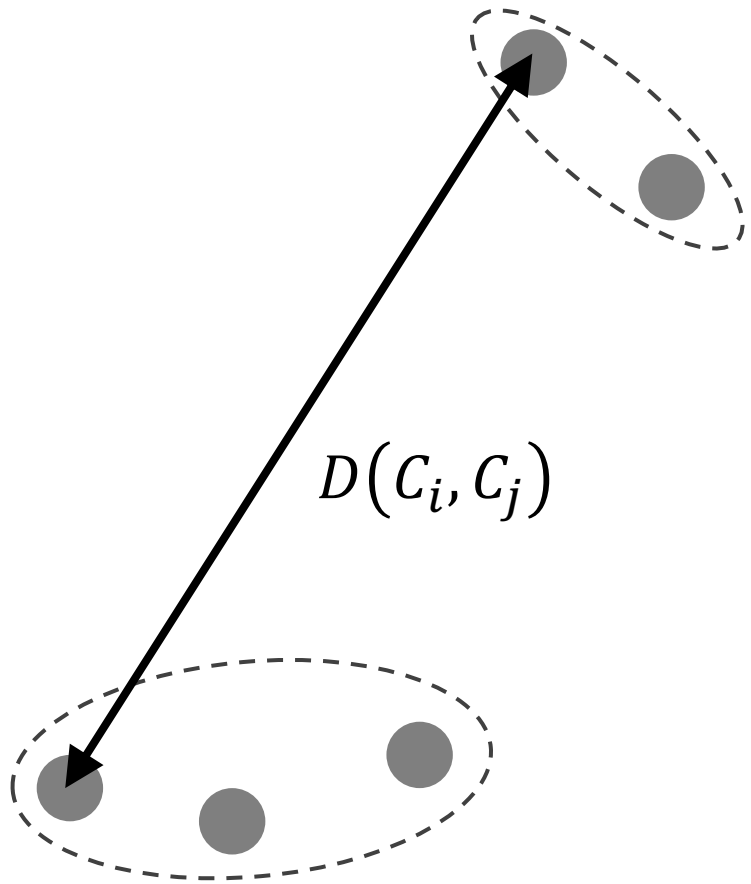




完全連結法（最長距離法）

完全連結法は、対象クラスタに含まれるサンプルのうち、もっとも遠いサンプル同士の距離をクラスタ間距離としている。比較する二つのクラスタ内のサンプル同士の距離ががすべて、クラスタ間距離以内であることが保証される。計算が簡単で、最短距離法より性能が良いとされる。

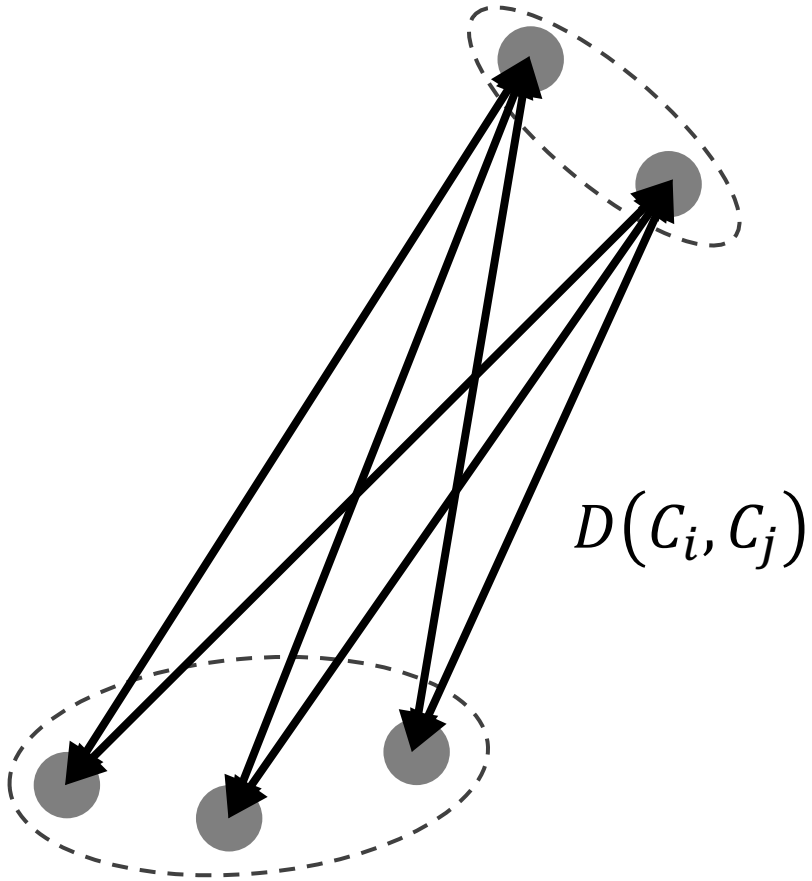
$$D(C_i, C_j) = \max_{\boldsymbol{x}_i \in C_i, \boldsymbol{x}_j \in C_j} \{d(\boldsymbol{x}_i, \boldsymbol{x}_j)\}$$



平均連結法（群平均法）

平均連結法は、対象クラスタに含まれるサンプル同士の総当たりの距離の平均をクラスタ間距離としている。計算が簡単で、外れ値に影響しにくく、安定した性能が得られる。

$$D(C_i, C_j) = \frac{1}{|C_i||C_j|} \sum_{\boldsymbol{x}_i \in C_i} \sum_{x_j \in C_j} \{d(\boldsymbol{x}_i, \boldsymbol{x}_j)\}$$



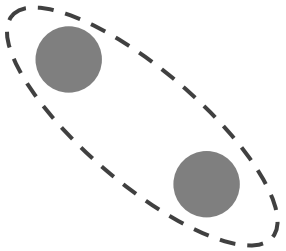
ウォード法

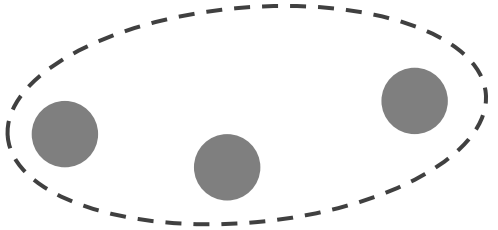
ウォード法は、２ つのクラスタが独立しているときの「各サンプルとクラスタ中心の距離の和」と ２ つのクラスタをまとめたときの「各サンプルとクラスタ中心の距離の和」の差をクラスタ間距離としている。他の方法に比べて高い性能が得られると知られている。

$$D(C_i, C_j) = E(C_i \cup C_j) - E(C_i) - E(C_j)$$

ただし、 $$E(C_i) = \sum_{\boldsymbol{x}_i \in C_i} \left(d(\boldsymbol{x}_i, \boldsymbol{c}_i)\right)^2$$

$$\boldsymbol{c}_i = \sum_{\boldsymbol{x}_i \in C_i} \frac{\boldsymbol{x}_i}{|C_i|}$$





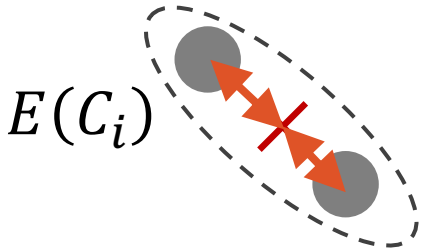
ウォード法

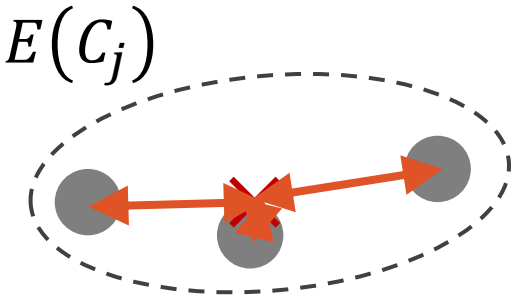
ウォード法は、２ つのクラスタが独立しているときの「各サンプルとクラスタ中心の距離の和」と ２ つのクラスタをまとめたときの「各サンプルとクラスタ中心の距離の和」の差をクラスタ間距離としている。他の方法に比べて高い性能が得られると知られている。

$$D(C_i, C_j) = E(C_i \cup C_j) - E(C_i) - E(C_j)$$

ただし、 $$E(C_i) = \sum_{\boldsymbol{x}_i \in C_i} \left(d(\boldsymbol{x}_i, \boldsymbol{c}_i)\right)^2$$

$$\boldsymbol{c}_i = \sum_{\boldsymbol{x}_i \in C_i} \frac{\boldsymbol{x}_i}{|C_i|}$$





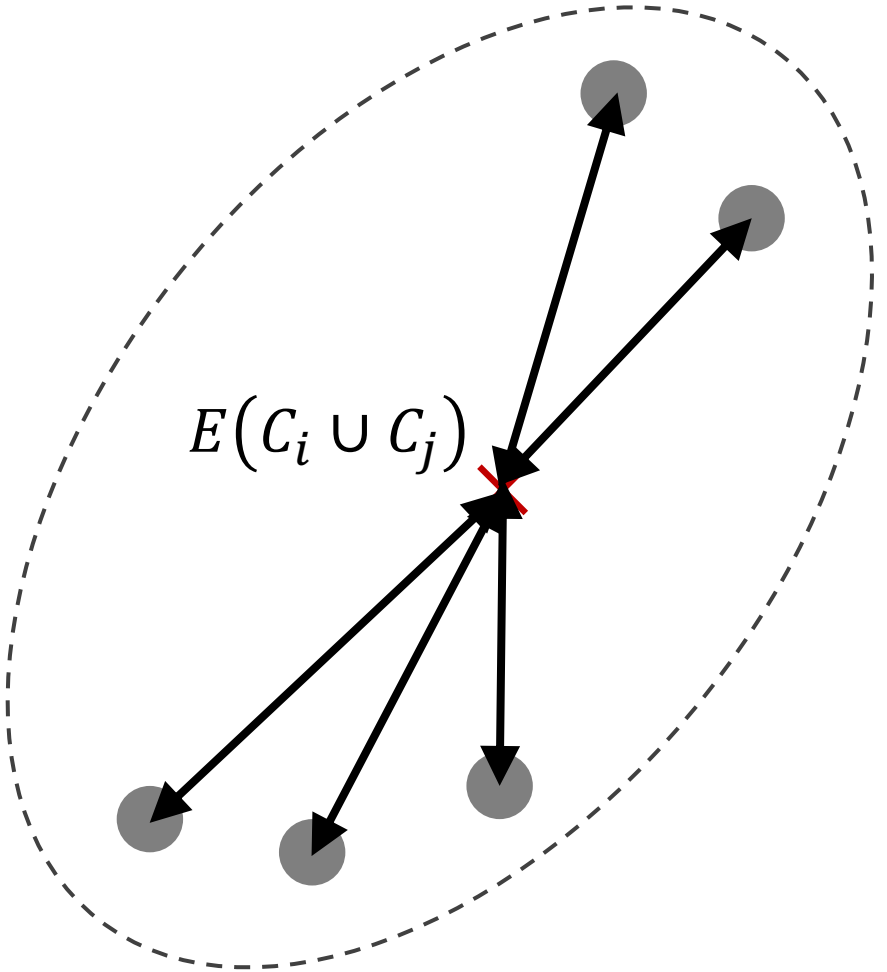
ウォード法

ウォード法は、２ つのクラスタが独立しているときの「各サンプルとクラスタ中心の距離の和」と ２ つのクラスタをまとめたときの「各サンプルとクラスタ中心の距離の和」の差をクラスタ間距離としている。他の方法に比べて高い性能が得られると知られている。

$$D(C_i, C_j) = E(C_i \cup C_j) - E(C_i) - E(C_j)$$

ただし、 $$E(C_i) = \sum_{\boldsymbol{x}_i \in C_i} \left(d(\boldsymbol{x}_i, \boldsymbol{c}_i)\right)^2$$

$$\boldsymbol{c}_i = \sum_{\boldsymbol{x}_i \in C_i} \frac{\boldsymbol{x}_i}{|C_i|}$$



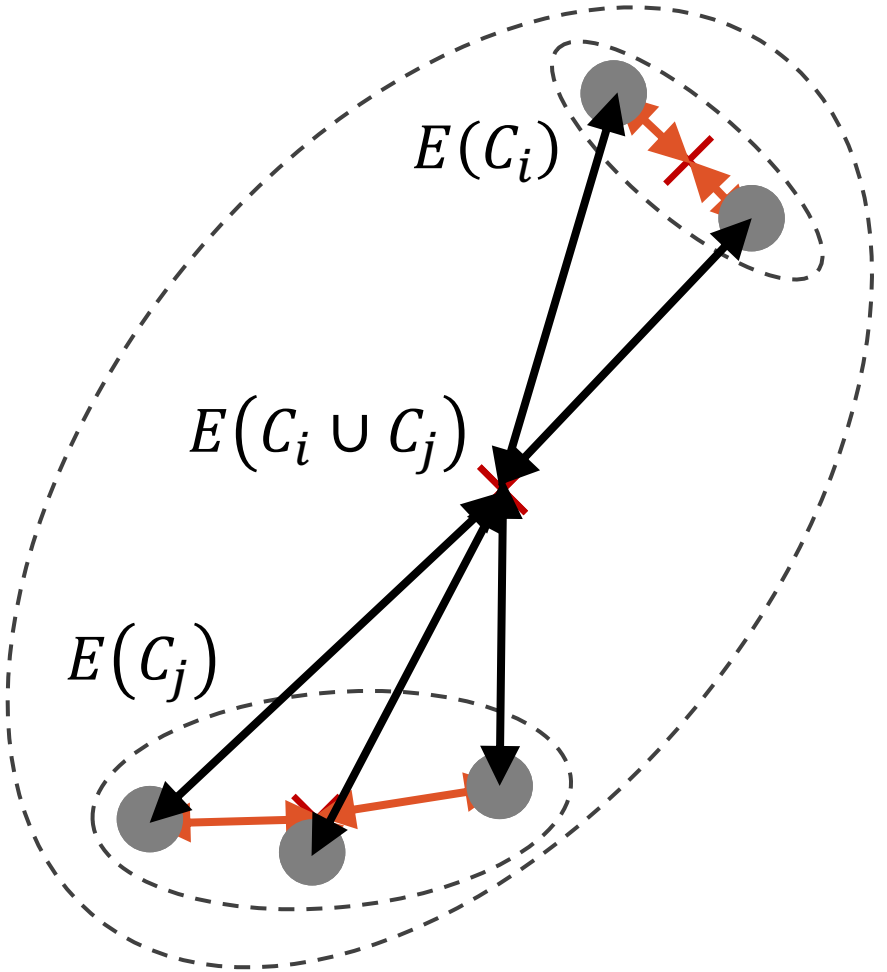
ウォード法

ウォード法は、２ つのクラスタが独立しているときの「各サンプルとクラスタ中心の距離の和」と ２ つのクラスタをまとめたときの「各サンプルとクラスタ中心の距離の和」の差をクラスタ間距離としている。他の方法に比べて高い性能が得られると知られている。

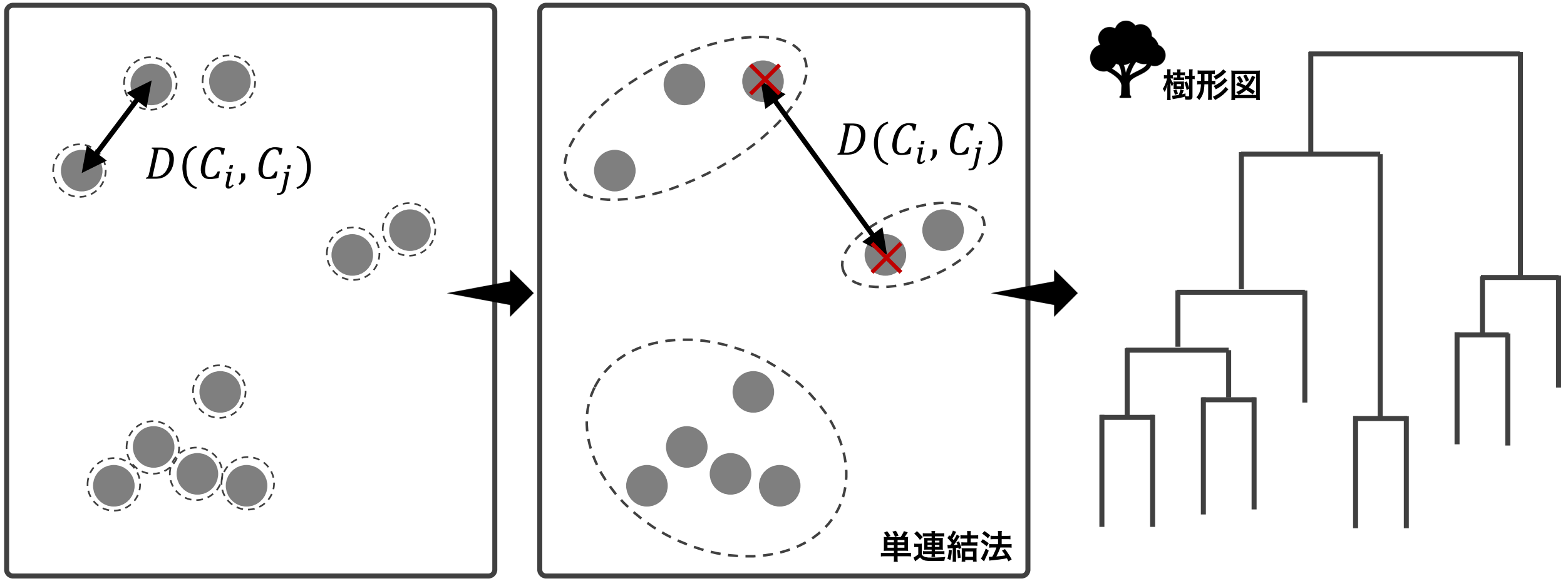
$$D(C_i, C_j) = E(C_i \cup C_j) - E(C_i) - E(C_j)$$

ただし、 $$E(C_i) = \sum_{\boldsymbol{x}_i \in C_i} \left(d(\boldsymbol{x}_i, \boldsymbol{c}_i)\right)^2$$

$$\boldsymbol{c}_i = \sum_{\boldsymbol{x}_i \in C_i} \frac{\boldsymbol{x}_i}{|C_i|}$$

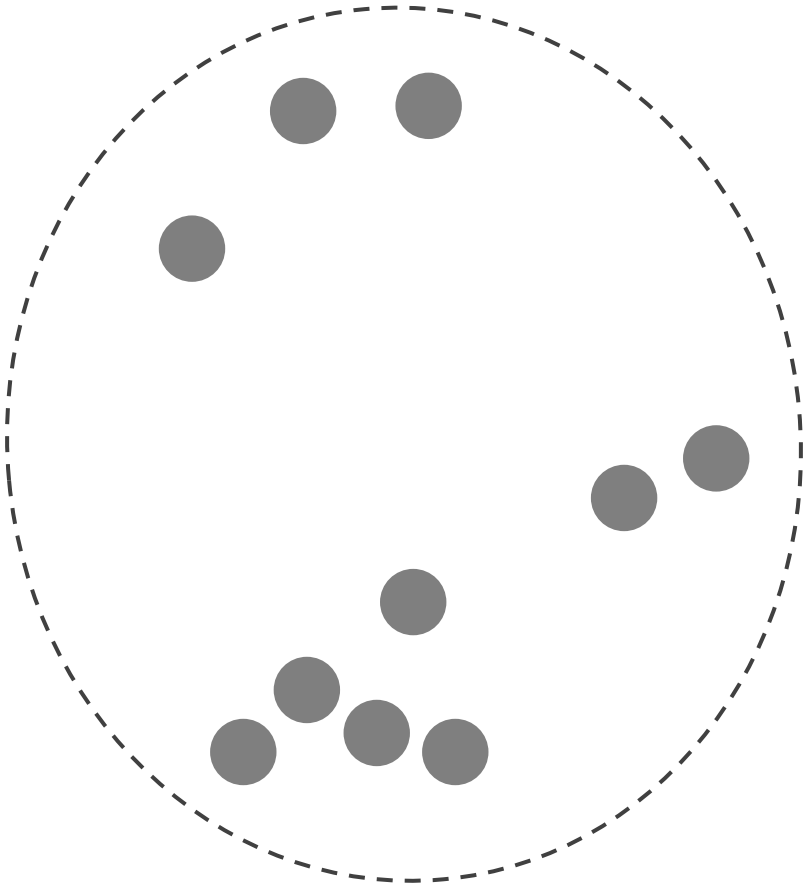


階層的クラスタリング（凝集型）



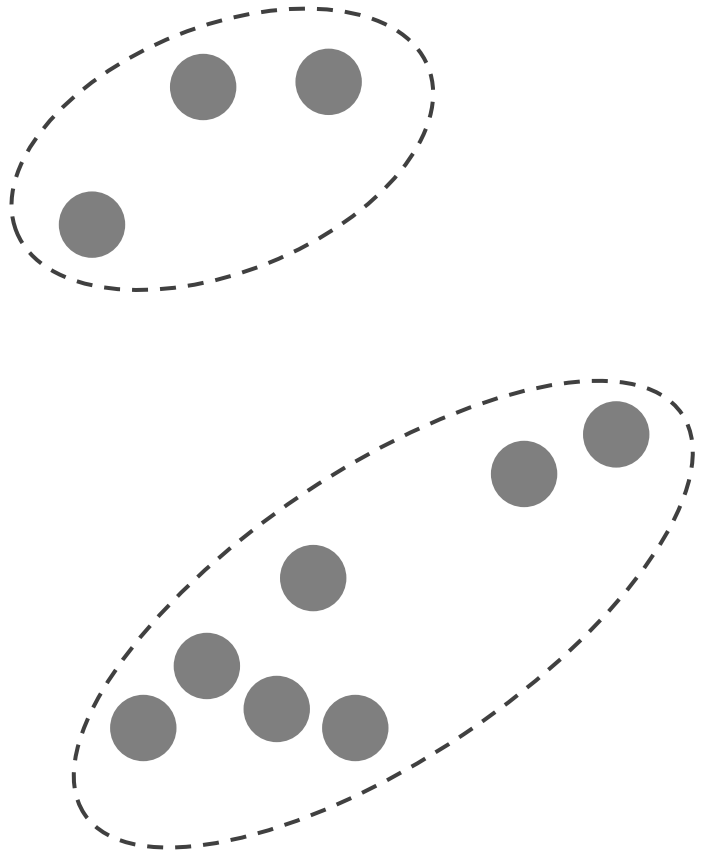
階層的クラスタリング（分割型）

分割型クラスタリングでは、すべてのサンプルが 1 つのクラスタに存在する状態からはじめ、これらのサンプルを 2 つのサブクラスのいずれかに分割する方法である。そして、1 つのサブクラスタに 1 つのサンプルが存在するまで、この操作を再帰的に繰り返す。n 個のサンプルを 2 つのサブクラスタに分けるときの場合の数は、2^n 通りの分け方が考えられる。この操作を再帰的に繰り返して行ったときに、計算量が莫大になる。



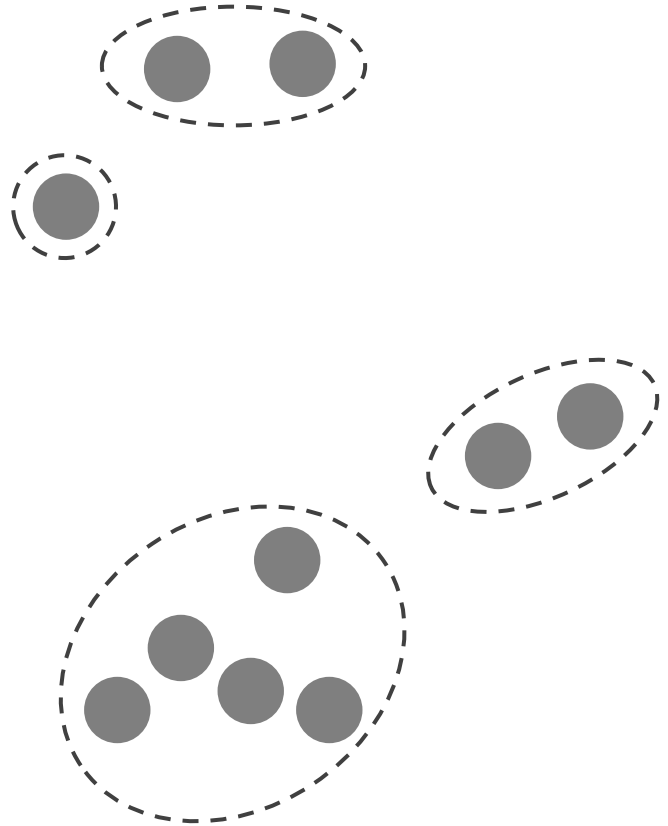
階層的クラスタリング（分割型）

分割型クラスタリングでは、すべてのサンプルが 1 つのクラスタに存在する状態からはじめ、これらのサンプルを 2 つのサブクラスのいずれかに分割する方法である。そして、1 つのサブクラスタに 1 つのサンプルが存在するまで、この操作を再帰的に繰り返す。n 個のサンプルを 2 つのサブクラスタに分けるときの場合の数は、2^n 通りの分け方が考えられる。この操作を再帰的に繰り返して行ったときに、計算量が莫大になる。



階層的クラスタリング（分割型）

分割型クラスタリングでは、すべてのサンプルが 1 つのクラスタに存在する状態からはじめ、これらのサンプルを 2 つのサブクラスのいずれかに分割する方法である。そして、1 つのサブクラスタに 1 つのサンプルが存在するまで、この操作を再帰的に繰り返す。n 個のサンプルを 2 つのサブクラスタに分けるときの場合の数は、2^n 通りの分け方が考えられる。この操作を再帰的に繰り返して行ったときに、計算量が莫大になる。



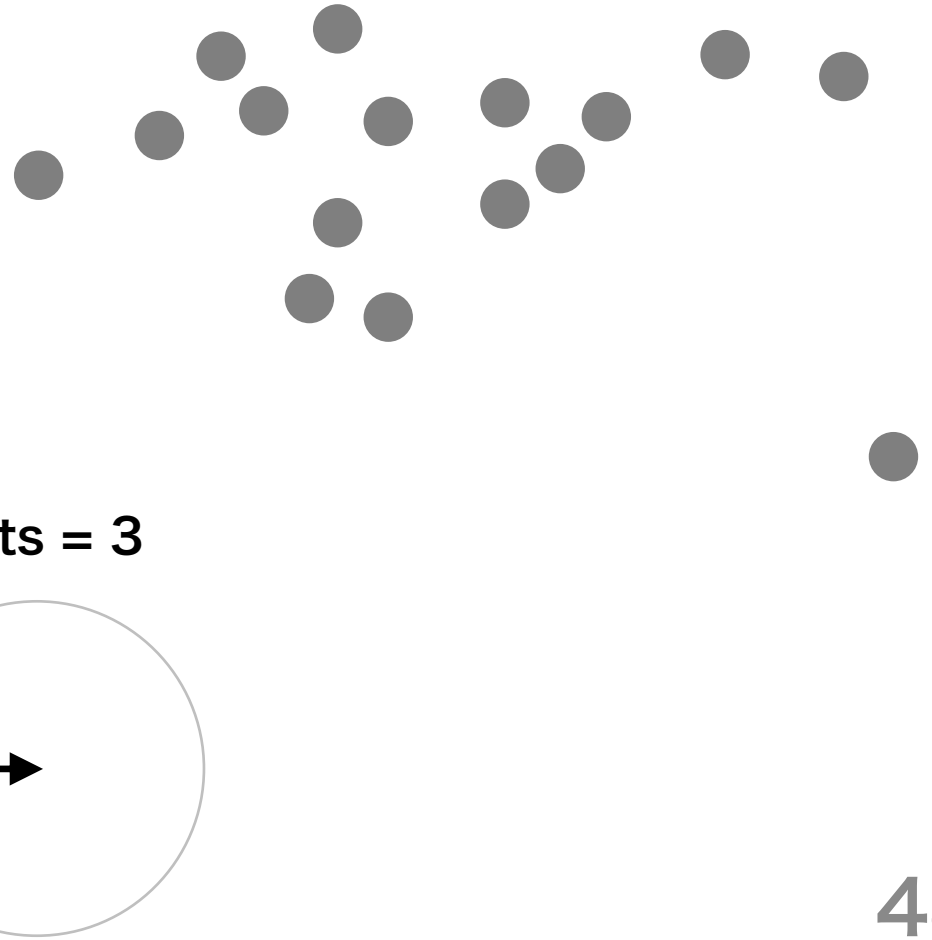
クラスタリング

- プロトタイプベースクラスタリング
- 階層的クラスタリング
- 密度ベースクラスタリング

DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。

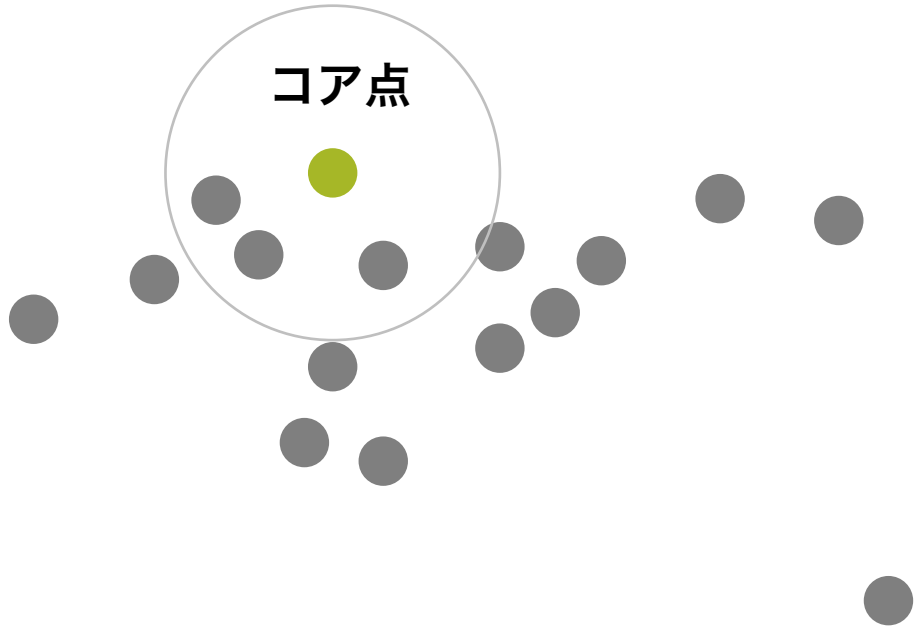
1. データ点を 3 種類に分類する。
   - 半径 ε 内に少なくとも minPts 個の点を含む点をコア点とする。
   - 半径 ε 内に minPts 個未満の点をボーダー点とする。
   - 半径 ε 内に他の点が含まれていない点をノイズとする。
2. コア点のまとまりからクラスタを作成し、ボーダー点を該当するクラスタに割り当てる。



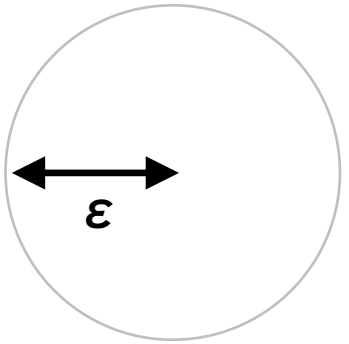
DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。

1. データ点を 3 種類に分類する。
   - 半径 ε 内に少なくとも minPts 個の点を含む点をコア点とする。
   - 半径 ε 内に minPts 個未満の点をボーダー点とする。
   - 半径 ε 内に他の点が含まれていない点をノイズとする。
2. コア点のまとまりからクラスタを作成し、ボーダー点を該当するクラスタに割り当てる。



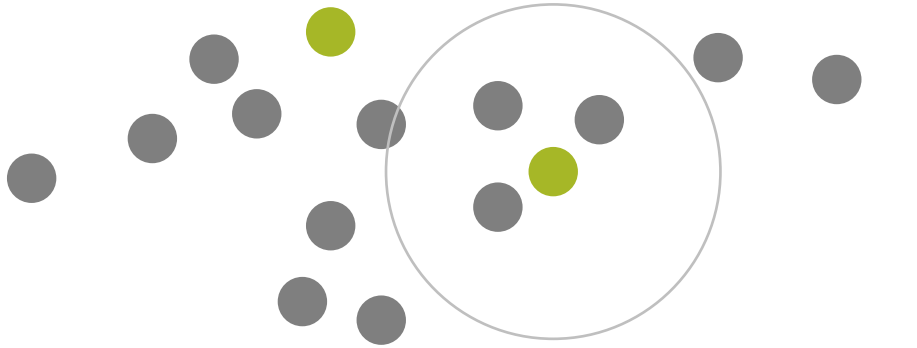
minPts = 3



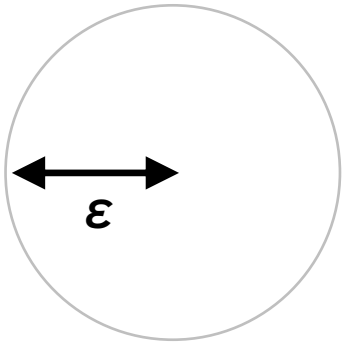
DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。

1. データ点を 3 種類に分類する。
   - 半径 ε 内に少なくとも minPts 個の点を含む点をコア点とする。
   - 半径 ε 内に minPts 個未満の点をボーダー点とする。
   - 半径 ε 内に他の点が含まれていない点をノイズとする。
2. コア点のまとまりからクラスタを作成し、ボーダー点を該当するクラスタに割り当てる。



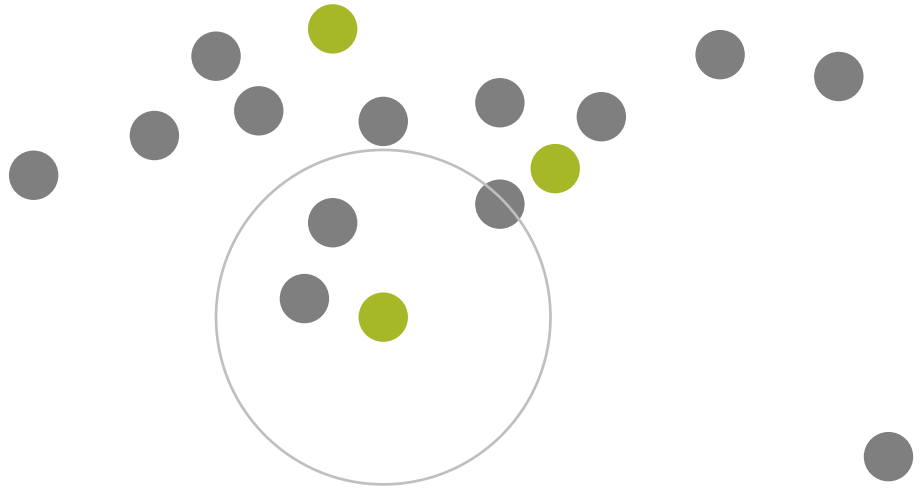
minPts = 3



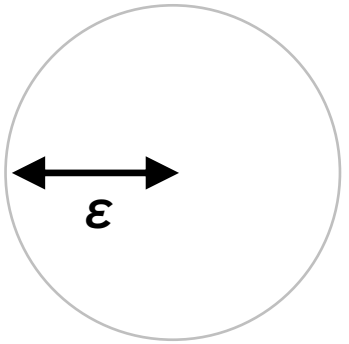
DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。

1. データ点を 3 種類に分類する。
   - 半径 ε 内に少なくとも minPts 個の点を含む点をコア点とする。
   - 半径 ε 内に minPts 個未満の点をボーダー点とする。
   - 半径 ε 内に他の点が含まれていない点をノイズとする。
2. コア点のまとまりからクラスタを作成し、ボーダー点を該当するクラスタに割り当てる。



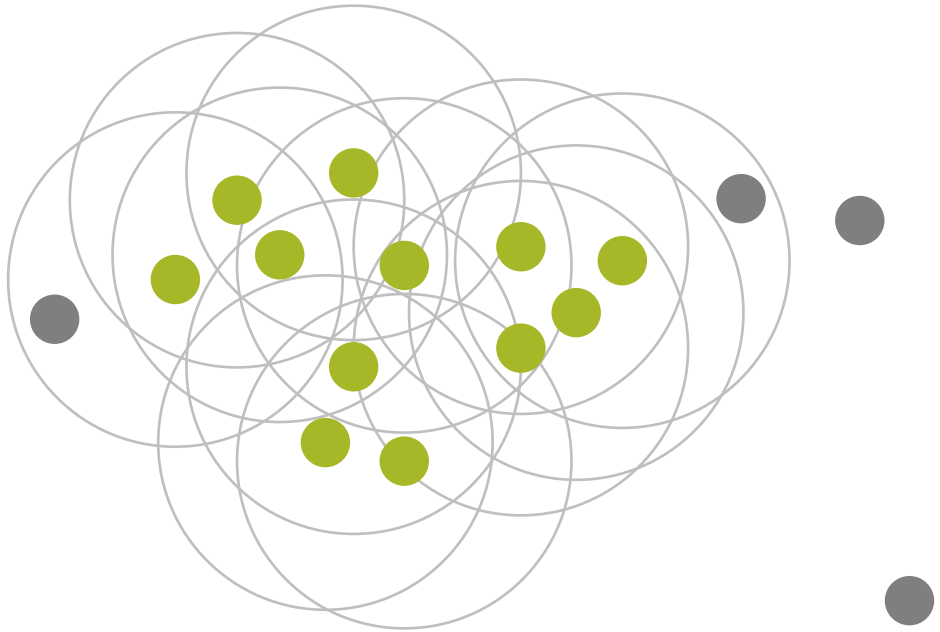
minPts = 3



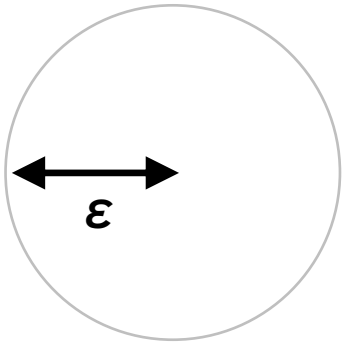
DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。

1. データ点を 3 種類に分類する。
   - 半径 ε 内に少なくとも minPts 個の点を含む点をコア点とする。
   - 半径 ε 内に minPts 個未満の点をボーダー点とする。
   - 半径 ε 内に他の点が含まれていない点をノイズとする。
2. コア点のまとまりからクラスタを作成し、ボーダー点を該当するクラスタに割り当てる。



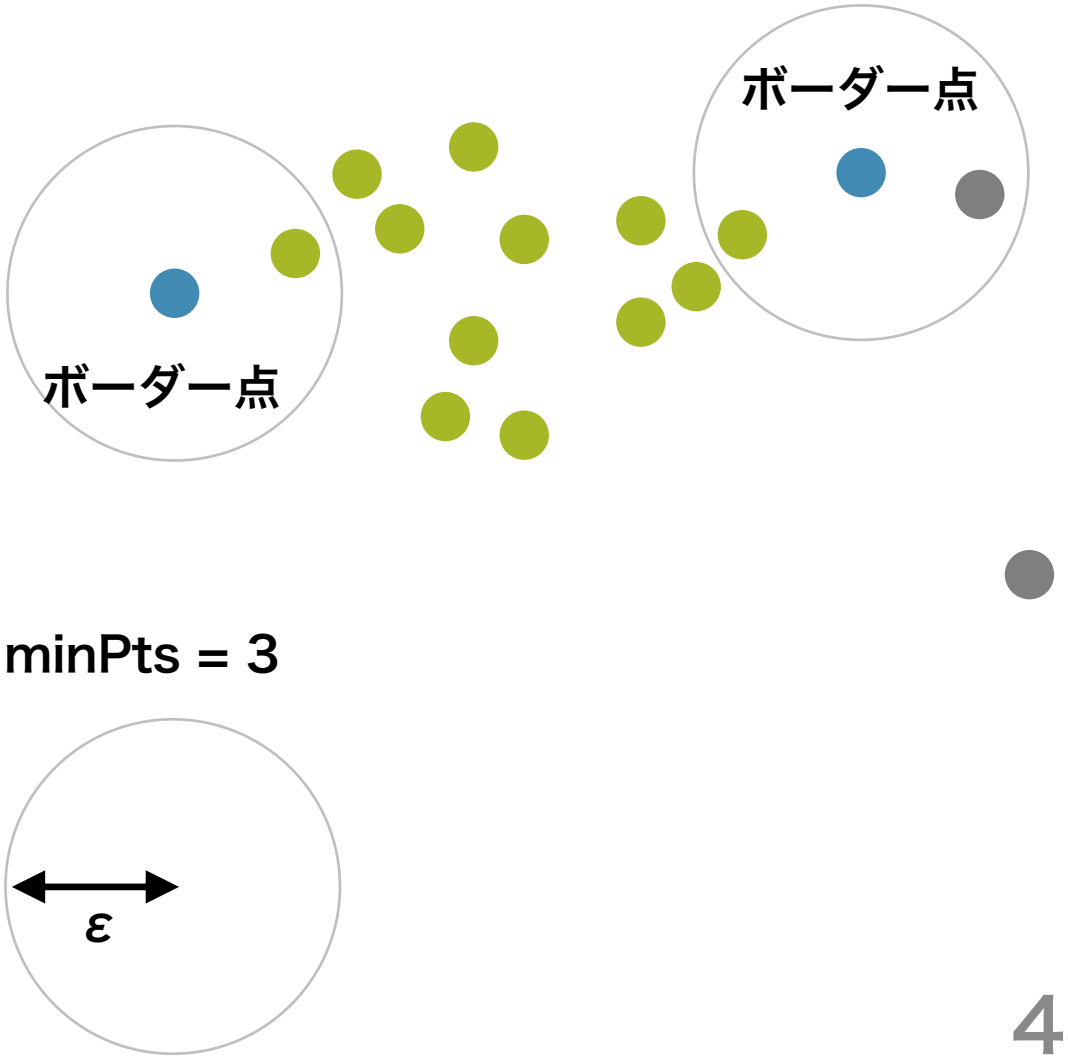
minPts = 3



DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。

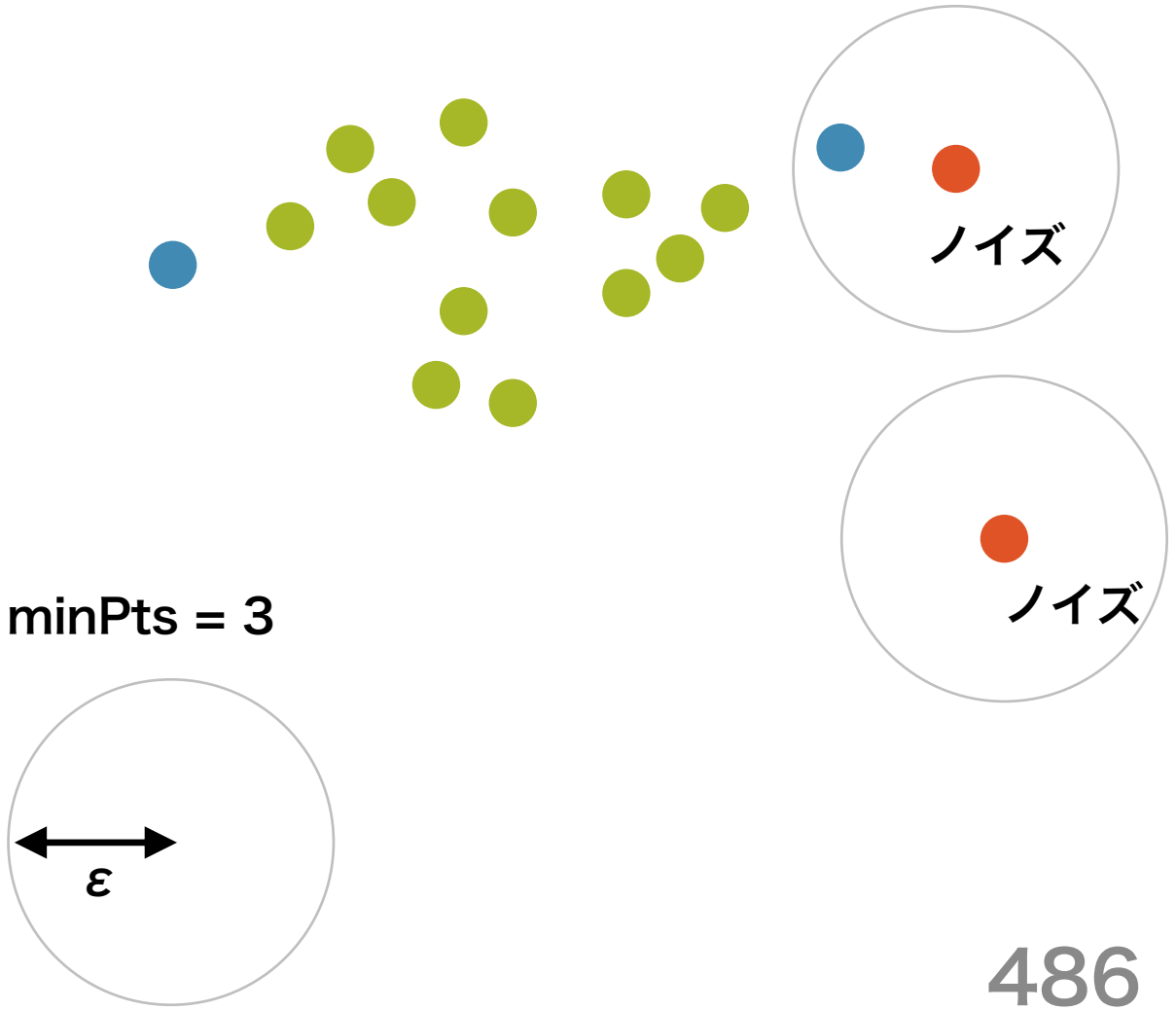
1. データ点を 3 種類に分類する。
   - 半径 ε 内に少なくとも minPts 個の点を含む点をコア点とする。
   - 半径 ε 内に minPts 個未満の点をボーダー点とする。
   - 半径 ε 内に他の点が含まれていない点をノイズとする。
2. コア点のまとまりからクラスタを作成し、ボーダー点を該当するクラスタに割り当てる。



DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。

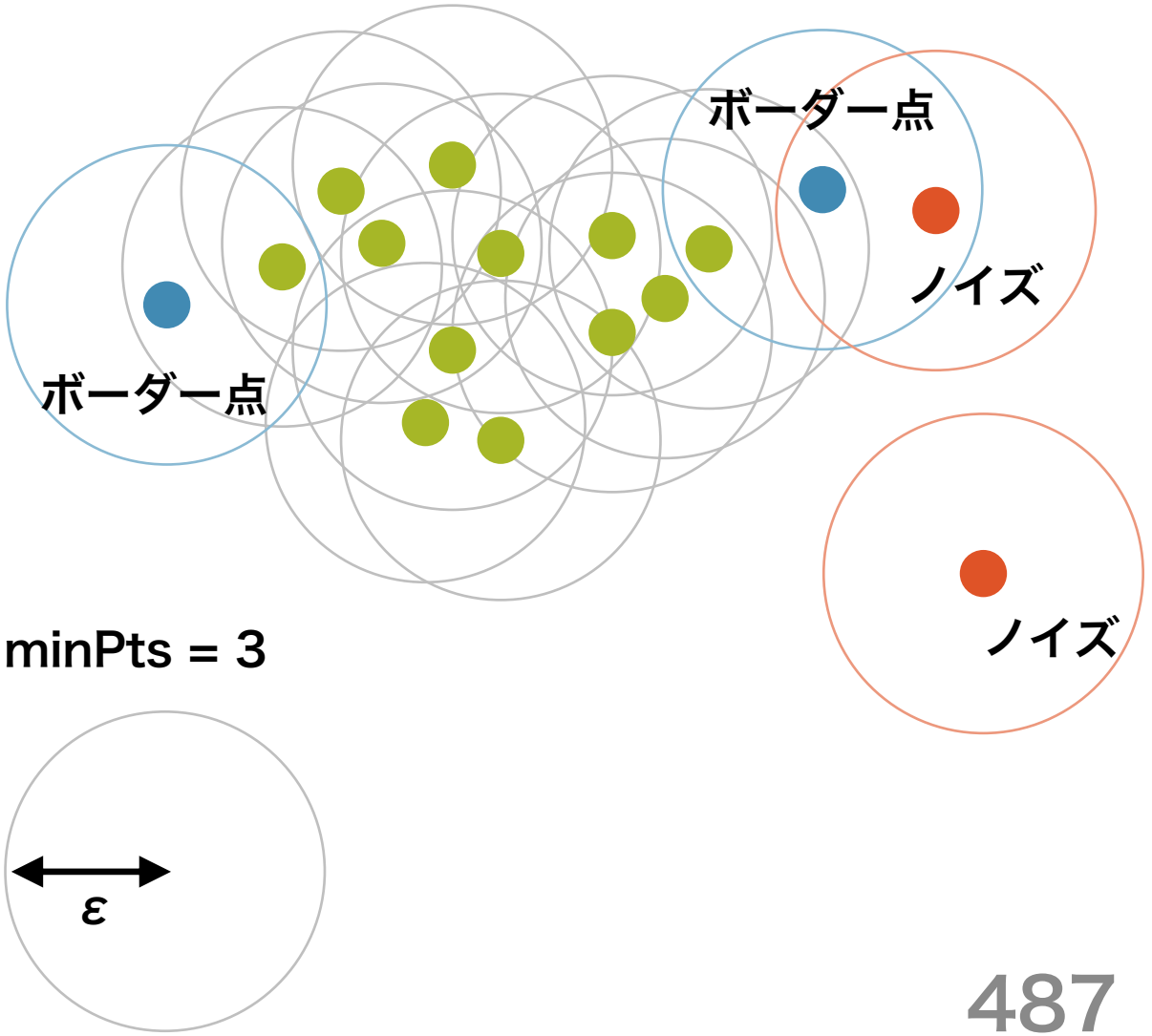
1. データ点を 3 種類に分類する。
   - 半径 ε 内に少なくとも minPts 個の点を含む点をコア点とする。
   - 半径 ε 内に minPts 個未満の点をボーダー点とする。
   - 半径 ε 内に他の点が含まれていない点をノイズとする。
2. コア点のまとまりからクラスタを作成し、ボーダー点を該当するクラスタに割り当てる。



DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。

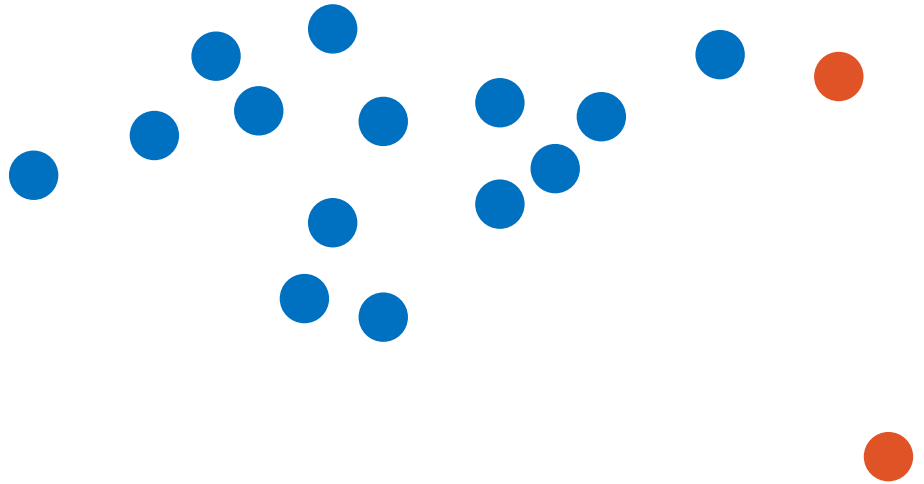
1. データ点を 3 種類に分類する。
   - 半径 ε 内に少なくとも minPts 個の点を含む点をコア点とする。
   - 半径 ε 内に minPts 個未満の点をボーダー点とする。
   - 半径 ε 内に他の点が含まれていない点をノイズとする。
2. コア点のまとまりからクラスタを作成し、ボーダー点を該当するクラスタに割り当てる。



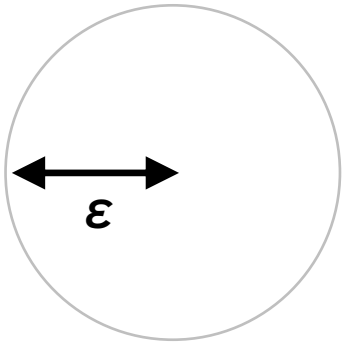
DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。

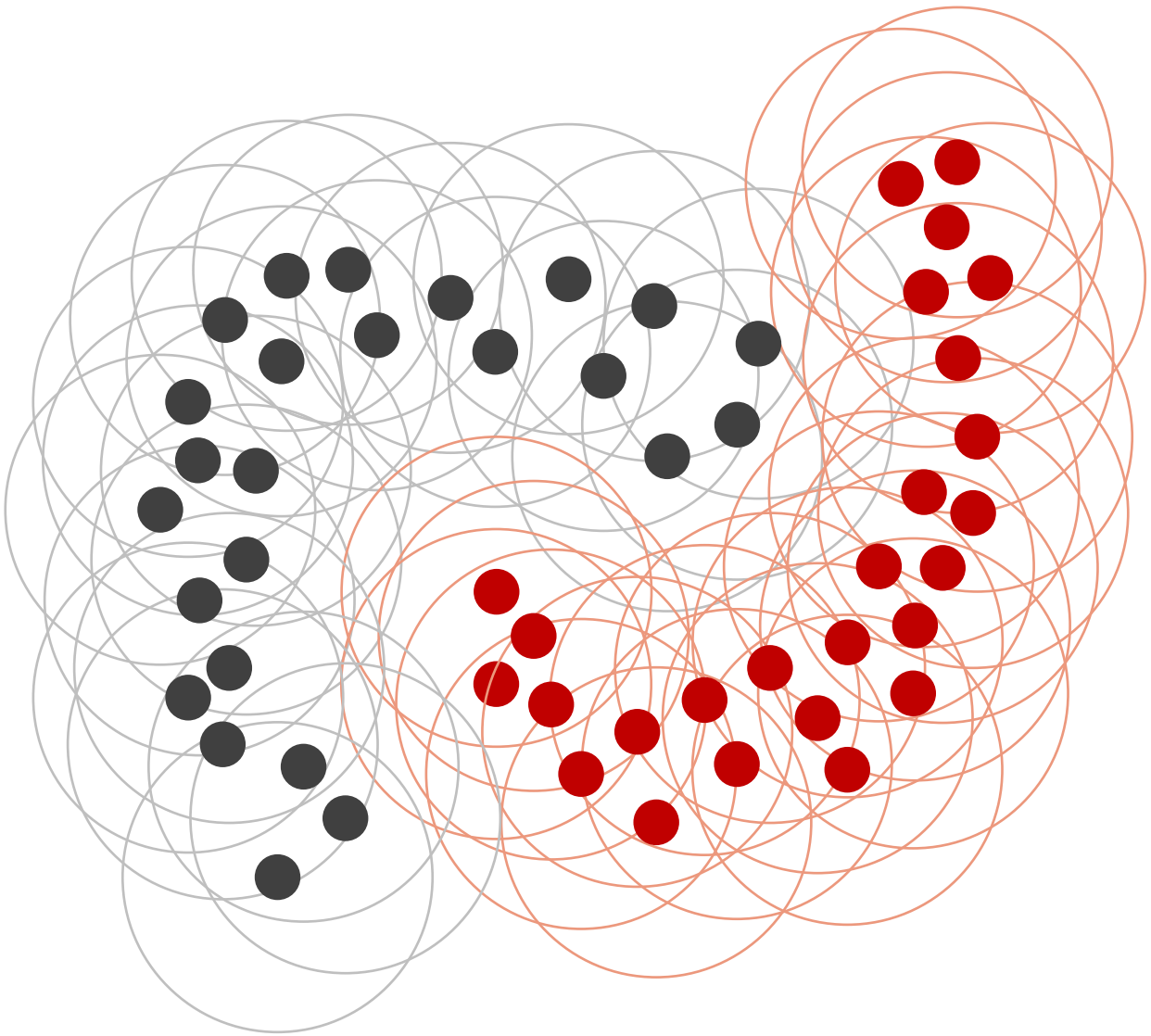
1. データ点を 3 種類に分類する。
   - 半径 ε 内に少なくとも minPts 個の点を含む点をコア点とする。
   - 半径 ε 内に minPts 個未満の点をボーダー点とする。
   - 半径 ε 内に他の点が含まれていない点をノイズとする。
2. コア点のまとまりからクラスタを作成し、ボーダー点を該当するクラスタに割り当てる。

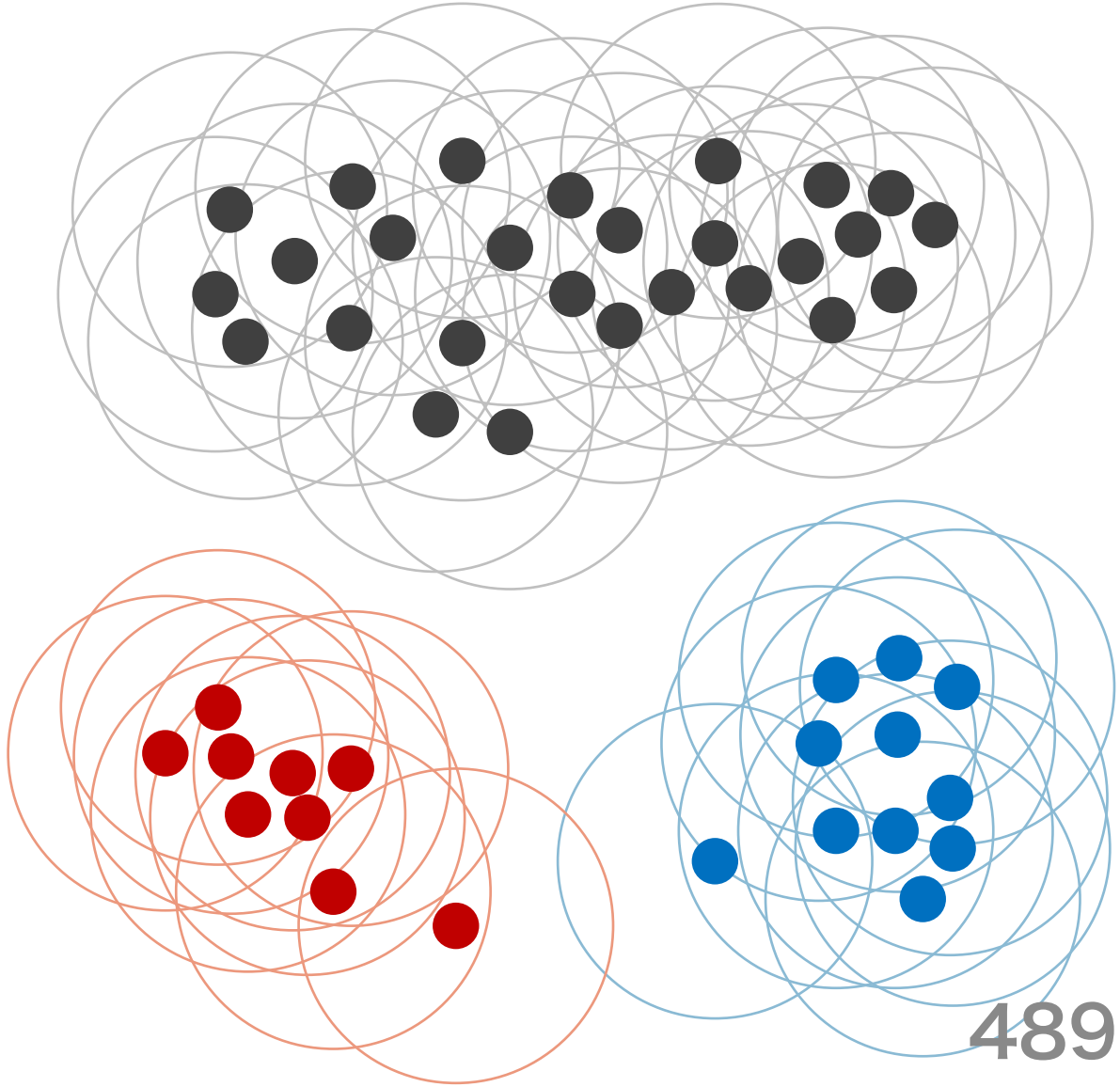


minPts = 3



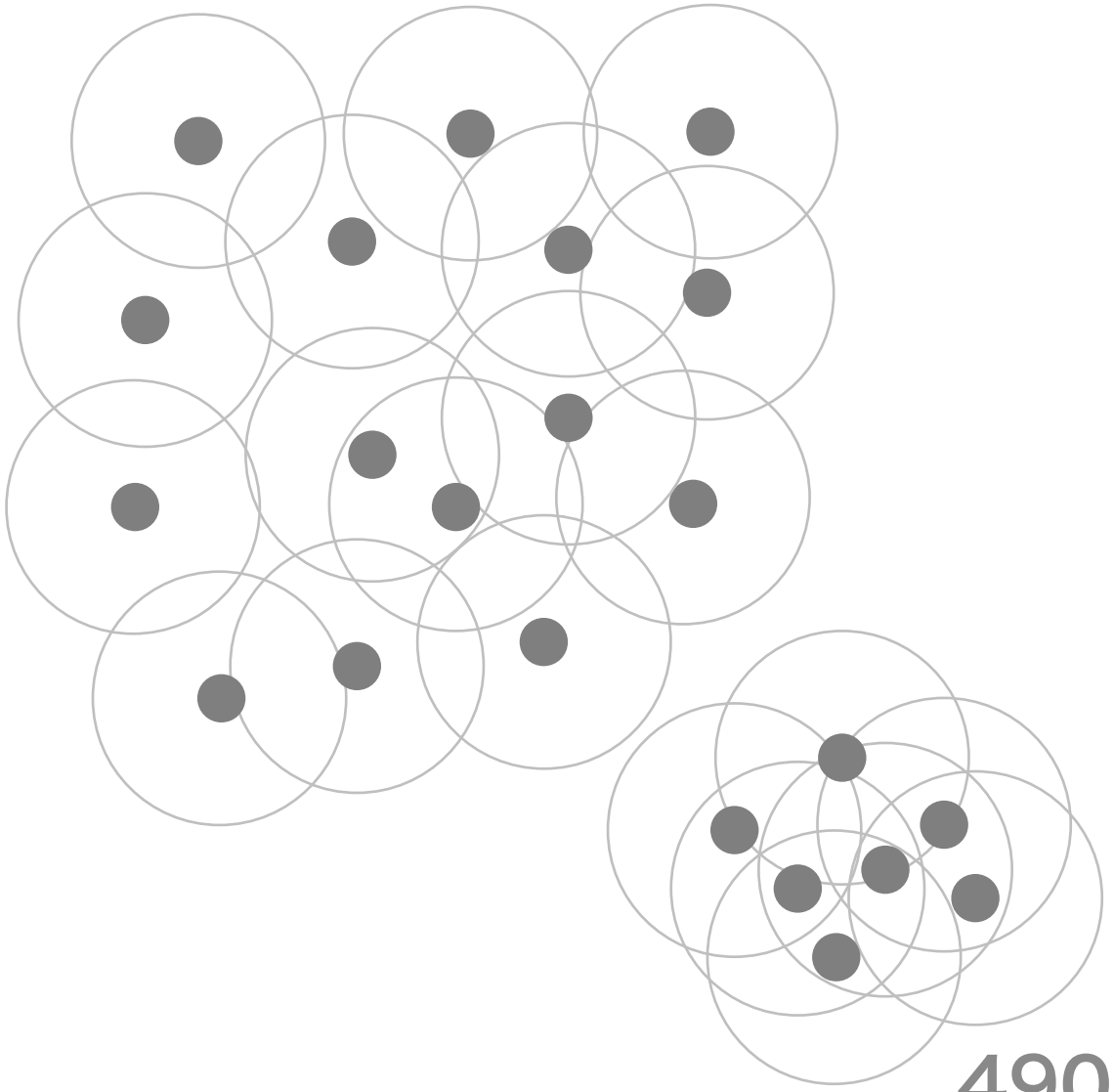
DBSCAN





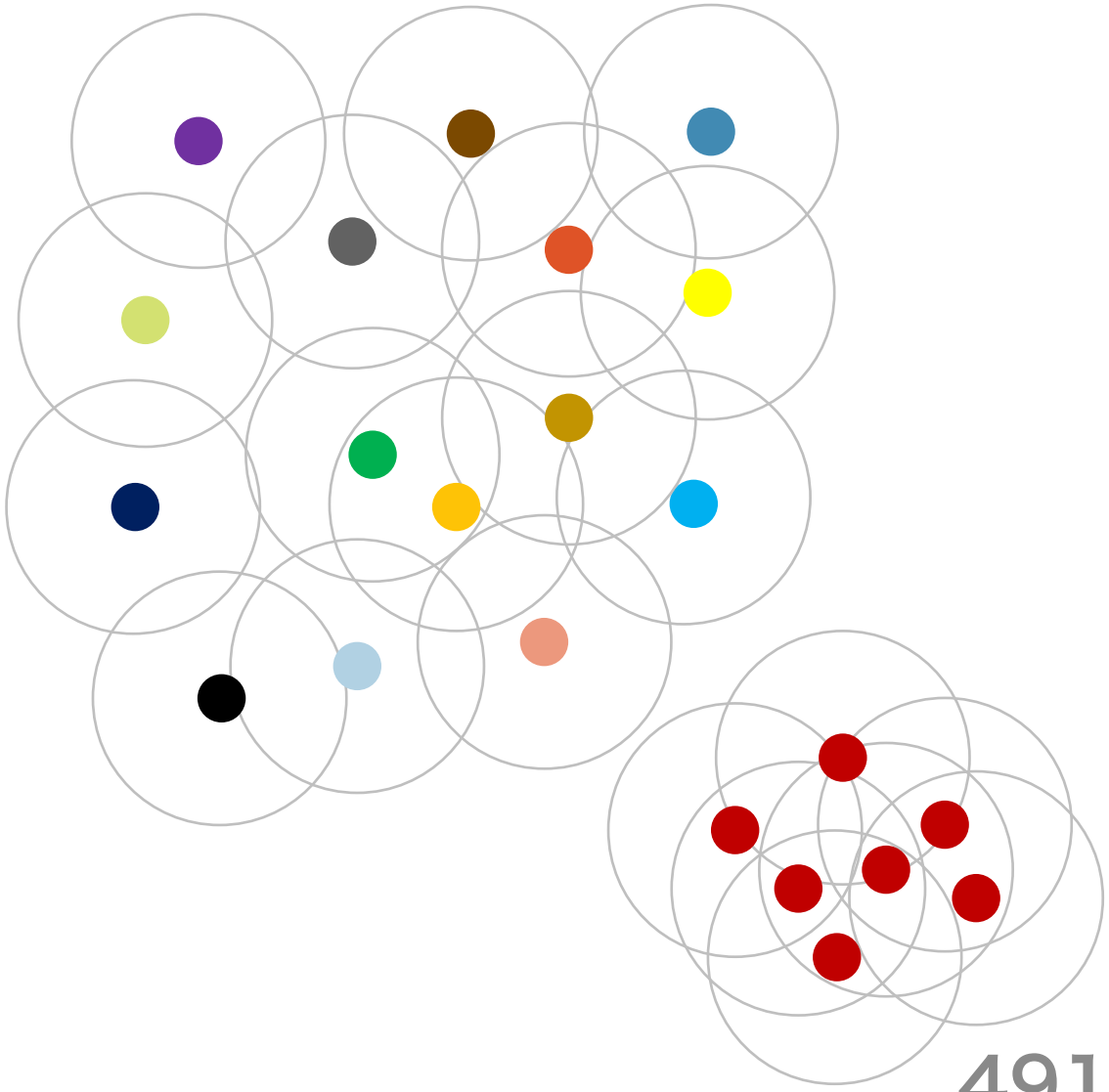
DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。ただし、密度をもとにクラスタリングを行なっているので、クラスタ内の密度が異なるとき、正確にクラスタリングできないことがある。



DBSCAN

DBSCAN は密度に基づくクラスタリングアルゴリズムであり、クラスタリングを行う前にクラスタ数を決めなくてもよい。また、データが球状構造でなくても有効である。ただし、密度をもとにクラスタリングを行なっているので、クラスタ内の密度が異なるとき、正確にクラスタリングできないことがある。



次元削減

- 主成分分析
- 行列分解
- t-SNE
- UMAP
- 自己組織化マップ

次元削減

- 主成分分析
- 行列分解
- t-SNE
- UMAP
- 自己組織化マップ

主成分分析

主成分分析は、オリジナルの特徴量空間を回転や平行移動し、データの情報量（分散）が最も大きい方向をPC1軸（第一主成分）とした座標空間を線形写像によって求める手法である。これにより、最初の数個の主成分を選ぶだけで、データ全体の情報量の 80% を説明できるようになったりする。

http://www.nlpca.org/fig_pca_principal_component_analysis.png

次元削減

- 主成分分析
- 行列分解
- t-SNE
- UMAP
- 自己組織化マップ

NMF

非負値行列因子分解（Non-negative Matrix Factorization, NMF）は、ランクの大きい非負値の行列 X を、ランクの小さい 2つの行列 W と H に分解することである。

基底行列　係数行列

$$X = WH$$

$$X \in \mathbb{R}^{m \times n},\ W \in \mathbb{R}^{m \times k},\ H \in \mathbb{R}^{k \times n}$$

基底行列　　係数行列

k は一般に k << min(m, n) となるように選ぶのが一般的である。また、W および H をランダムな値で初期化を行なったのちに、次式に基づいて、W と H を交互に最適化する。

$$\min_{W,H} \left\{ \frac{1}{2} \|X - WH\|^2 \right\}$$

ただし、W≥0 かつ H≥0 の条件の元で最適化を行うこと。

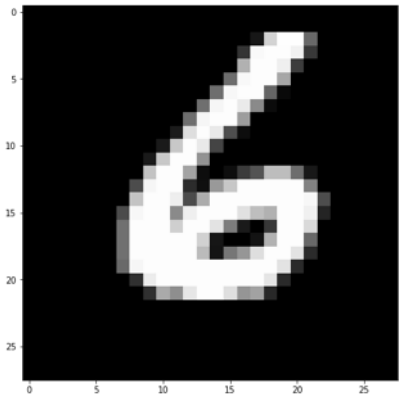
https://doi.org/10.1186/1751-0473-8-10

次元削減

- 主成分分析
- 行列分解
- t-SNE
- UMAP
- 自己組織化マップ

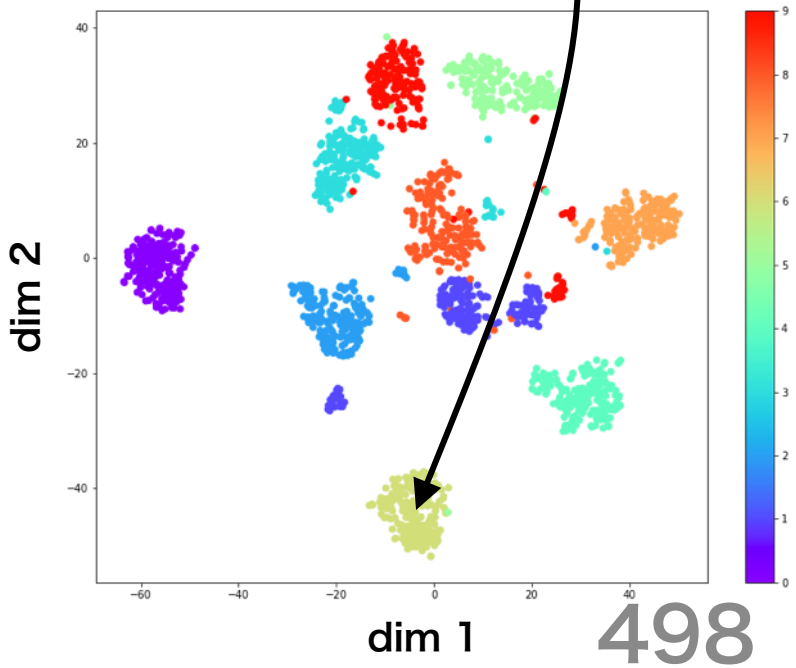
t-SNE

t-SNE (t-distributed Stochastic Neighbor Embedding ,t 分布型確率的近傍埋め込み) は、高次元空間における点の類似度を、低次元空間に埋め込む非線形変換である。



$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_2 \\ \vdots \\ x_{64} \end{pmatrix}$$

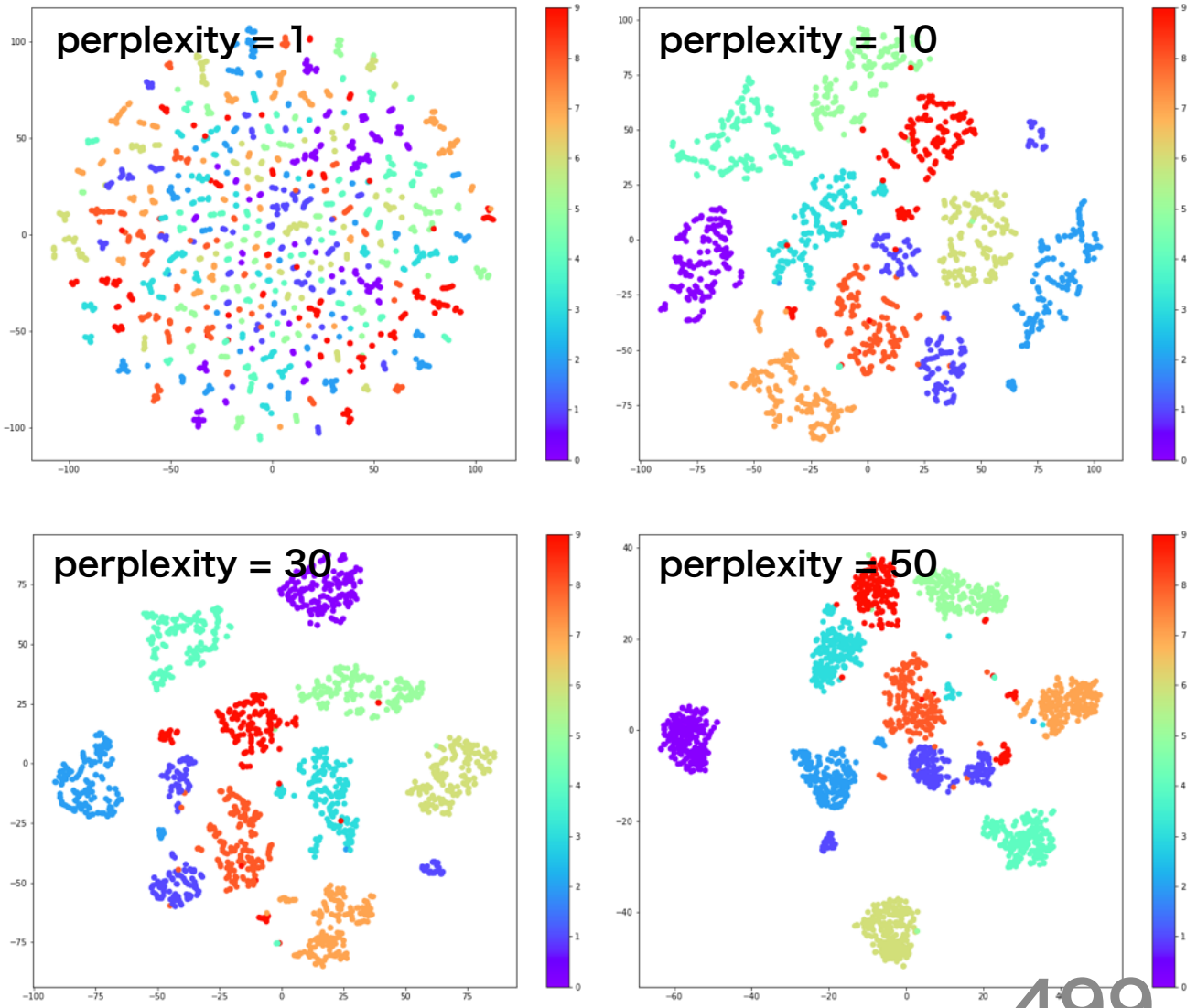
- n 次元のデータを 2 次元で可視化する目的で使われることが多い。
- t-SNE は線形変換でないため、クラスタ内のばらつきやクラスタ間距離は、ハイパーパラメータよって変化し、特別な意味を持たない。また、ハイパーパラメータ perplexity は 5 ～50 の範囲に指定するのが一般的である。



t-SNE

t-SNE (t-distributed Stochastic Neighbor Embedding ,t 分布型確率的近傍埋め込み) は、高次元空間における点の類似度を、低次元空間に埋め込む非線形変換である。

- n 次元のデータを 2 次元で可視化する目的で使われることが多い。
- t-SNE は線形変換でないため、クラスタ内のばらつきやクラスタ間距離は、ハイパーパラメータよって変化し、特別な意味を持たない。また、ハイパーパラメータ perplexity は 5 ～50 の範囲に指定するのが一般的である。



t-SNE

t-SNE は高次元空間 x における点の類似度を、低次元空間 y に埋め込む非線形変換である。高次元空間において、x_i と x_j の類似度を条件付き確率として表している。

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

$$p_{j|i} = \frac{\exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma_i^2}\right)}{\sum_{k \neq i} \exp\left(\frac{-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2}{2\sigma_i^2}\right)}$$

低次元空間において、y_i と y_j の類似度を自由度 1 の t 分布を利用した確率で表している。

$$q_{ij} = \frac{\left(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2\right)^{-1}}{\sum_{k \neq l}(1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}}$$

- 高次元空間における類似度の確率に現れる σ はハイパーパラメーターであり、次式に基づいて計算される。

$$perplexity = 2^{\sum_j p_{j|i} \log_2 p_{j|i}}$$

t-SNE

次に、高次元空間におけるサンプル同士の類似度 p_{ij} と低次元空間 q_{ij} における類似度が同じとなるような y を調べる。両者の差をカルバック・ライブラー情報量で表し、これを損失関数としてこうか勾配法により最適化を行う。

$$\min_{\boldsymbol{y}}\{D_{KL}(P||Q)\}$$

$$D_{KL}(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

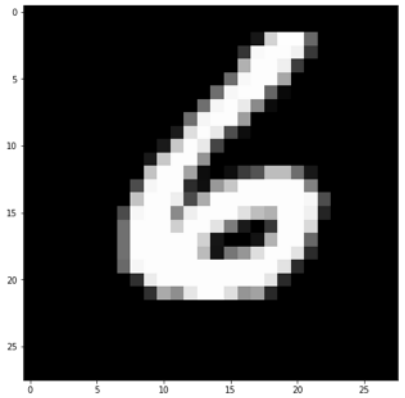
- カルバック・ライブラー情報量は、２つの確率分布の差を測る指標である。

次元削減

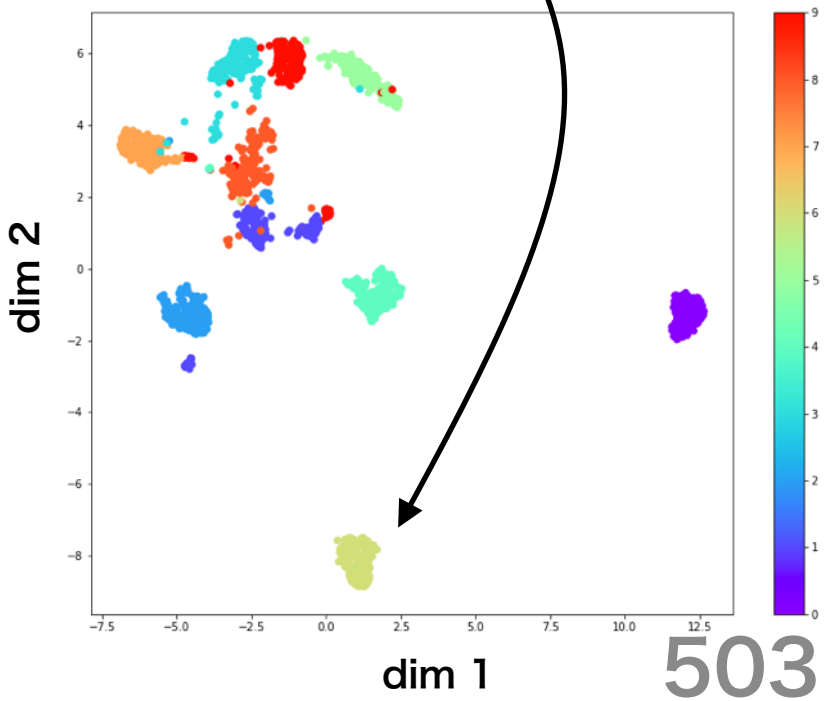
- 主成分分析
- 行列分解
- t-SNE
- UMAP
- 自己組織化マップ

UMAP

UMAP (Uniform Manifold Approximation and Projection) は次元削減や高次元データの可視化などの目的で使われている。

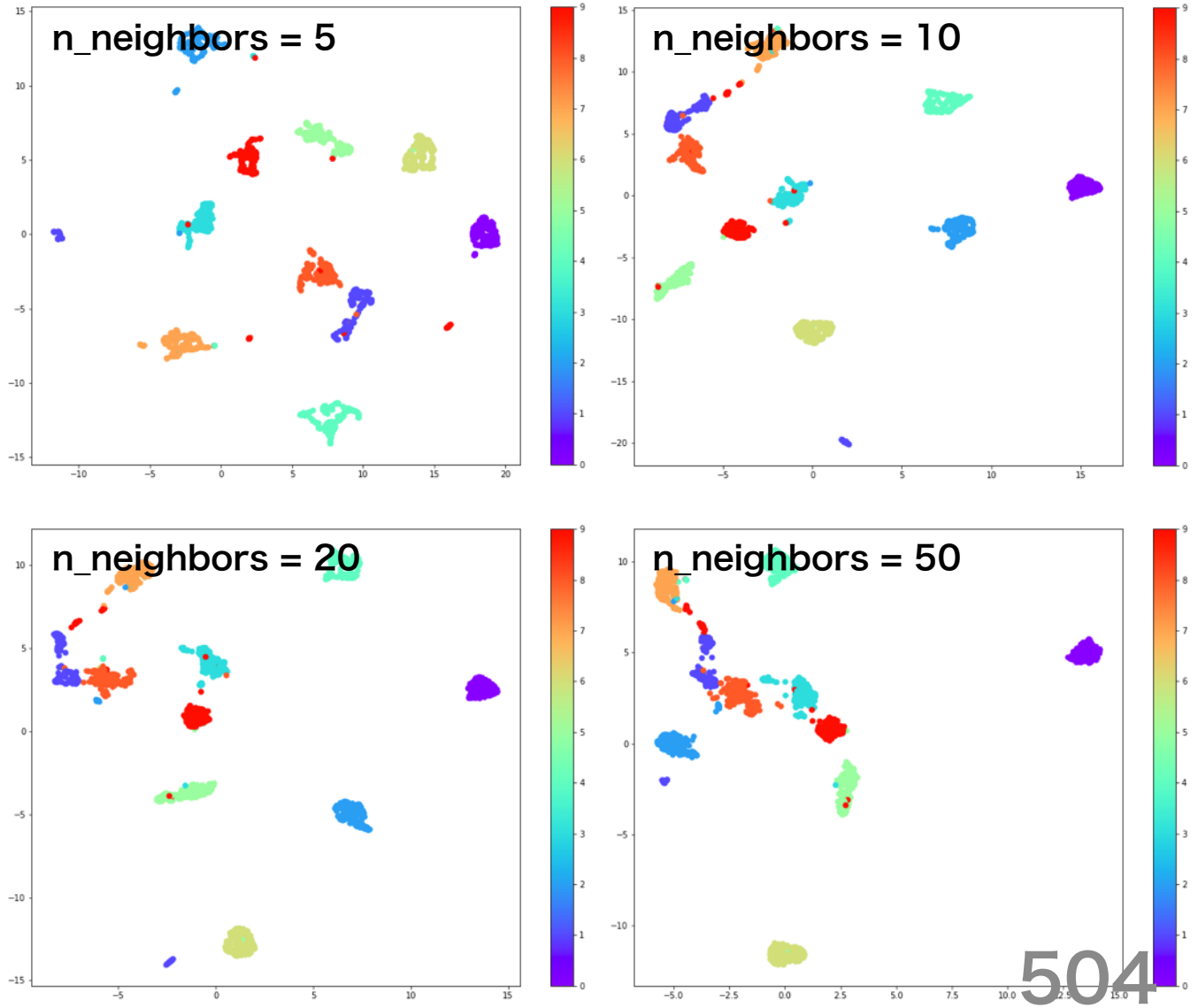


$$\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_2 \\ \vdots \\ x_{64} \end{pmatrix}$$



UMAP

UMAP (Uniform Manifold Approximation and Projection) は次元削減や高次元データの可視化などの目的で使われている。

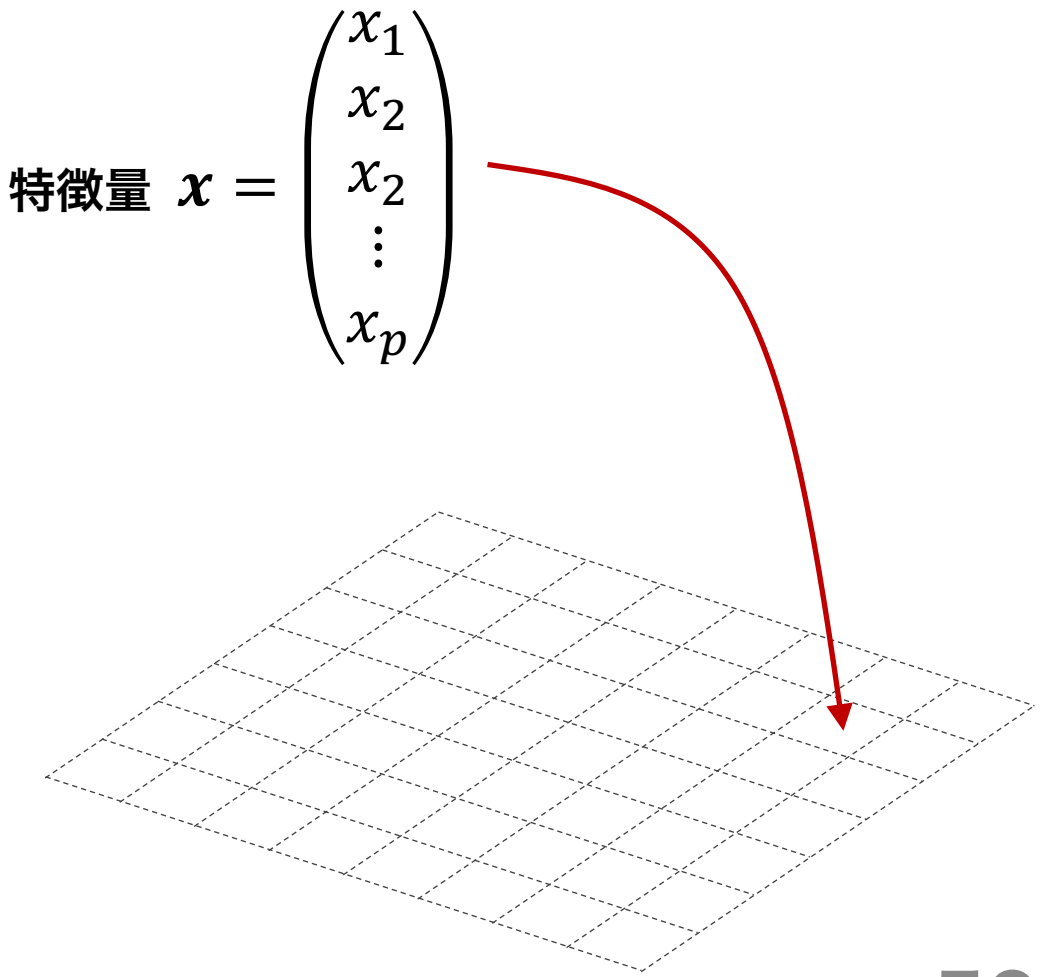


次元削減

- 主成分分析
- 行列分解
- t-SNE
- UMAP
- 自己組織化マップ

SOM

自己組織化マップ（self-organizing map / SOM）は、入力層と出力層からなる教師なし学習ニューラルネットワークの一つである。SOM は、クラスタリングや次元削減の目的で使われる。SOM の入力層は特徴量と同じ次元数であり、出力層は n×m の平面であるのが特徴的である。n および m はハイパーパラメータとして予め与える必要がある。出力層となる平面は正方形の格子状の他に正六角形が用いられることも多い。



SOM

SOM は次の手順に従って学習を行う。ここで、訓練サンプルの数を N 個、特徴量の数を p 個、出力層のサイズ（ユニット数）を mxn 個とする。

1. mxn 個のユニットそれぞれに長さ p 個の参照ベクトルを持たせる。各参照ベクトル $M_{m,n}$ の初期値は乱数で埋める。
2. j = 1 とする。
3. サンプル x_j とすべてのユニットの参照ベクトル $M_{m,n}$ との距離を計算する。最短距離を与えるユニット best matching unit (BMU) を決める。
4. 次式に基づいて、すべてのユニットの参照ベクトルを更新する。

$$\boldsymbol{M}_{m,n}^{(new)} = \boldsymbol{M}_{m,n}^{(old)} + h_{ck}\left(\boldsymbol{x}_j - \boldsymbol{M}_{m,n}^{(old)}\right)$$

   ただし、h_{ck} は近傍関数で、BTU の隣接ユニットであれば 1 を返し、そうでなければ 0 を返す。

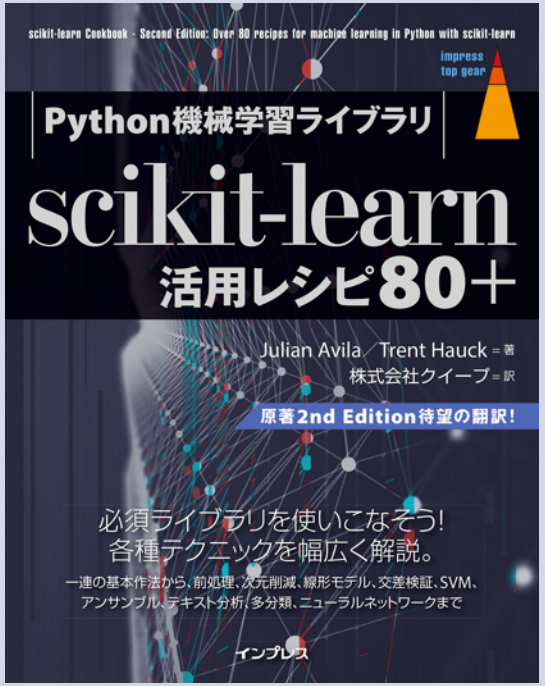
5. j を 1 増やして、j = n になるまで手順 3〜4 を繰り返す。
6. 手順 2〜5 を指定された回数だけ繰り返す。

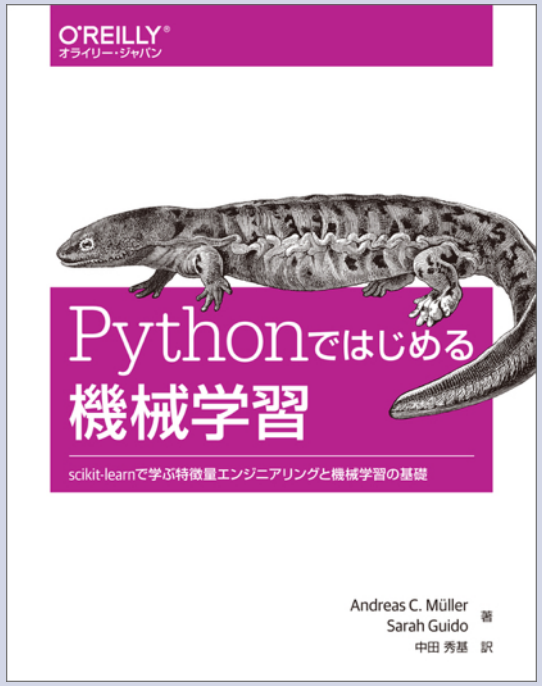
BMU

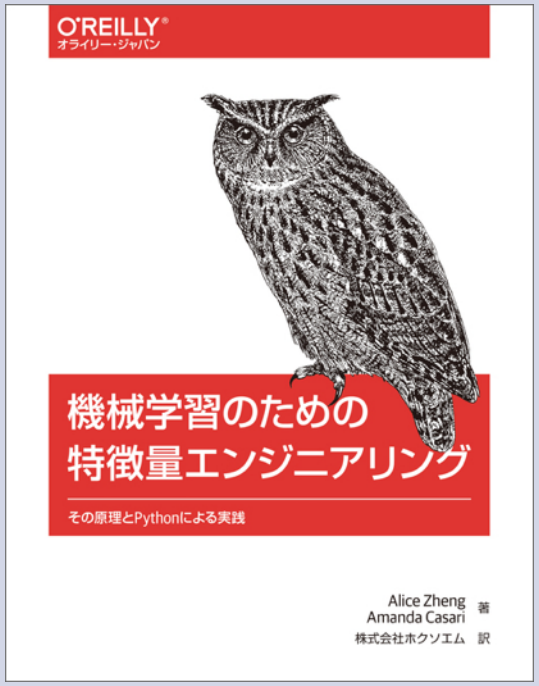
$\boldsymbol{x}$ $\boldsymbol{M}_{1,1}$

参考図書

Python Machine Learning, 2nd Edition
impress top gear
［第2版］
達人データサイエンティストによる理論と実践
Python
機械学習プログラミング
Sebastian Raschka / Vahid Mirjalili
株式会社クイープ
ML本ベストセラーの第2版!
機械学習の全般をカバー
◎各手法の考え方と数学的背景を解説
◎scikit-learnやTensorFlowで実践コーディング
◎CNNやRNNについても新たに解説
インプレス


scikit-learn Cookbook - Second Edition: Over 80 recipes for machine learning in Python with scikit-learn
impress top gear
Python機械学習ライブラリ
scikit-learn
活用レシピ80+
Julian Avila / Trent Hauck
株式会社クイープ
原著2nd Edition待望の翻訳!
必須ライブラリを使いこなそう!
各種テクニックを幅広く解説。
一連の基本作法から、前処理、次元削減、線形モデル、交差検証、SVM、
アンサンブル、テキスト分析、多分類、ニューラルネットワークまで
インプレス


O'REILLY
オライリー・ジャパン
Pythonではじめる
機械学習
scikit-learnで学ぶ特徴量エンジニアリングと機械学習の基礎
Andreas C. Müller
Sarah Guido
中田 秀基 訳

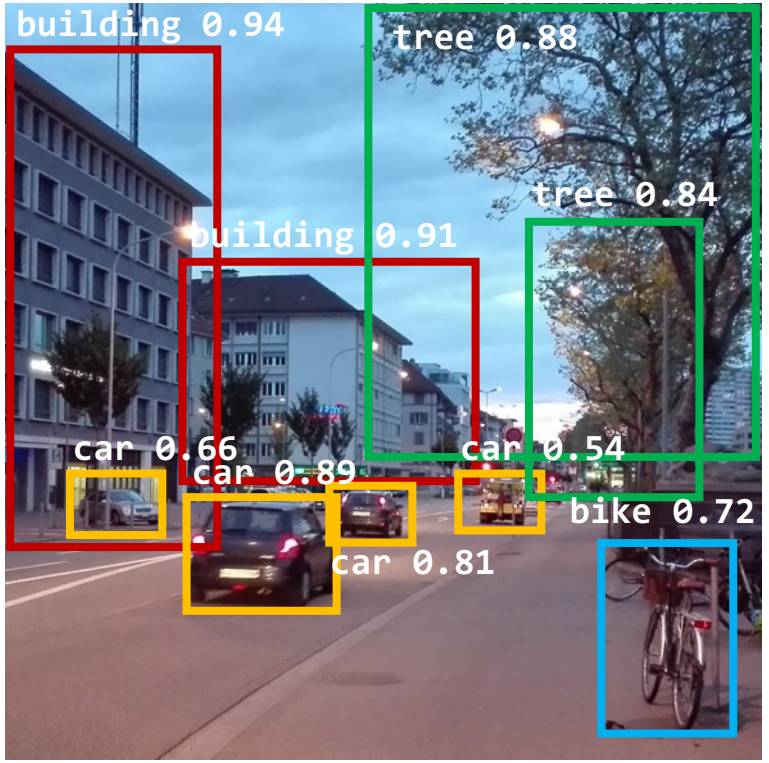

O'REILLY
オライリー・ジャパン
機械学習のための
特徴量エンジニアリング
その原理とPythonによる実践
Alice Zheng
Amanda Casari
株式会社ホクソエム 訳

画像解析概略

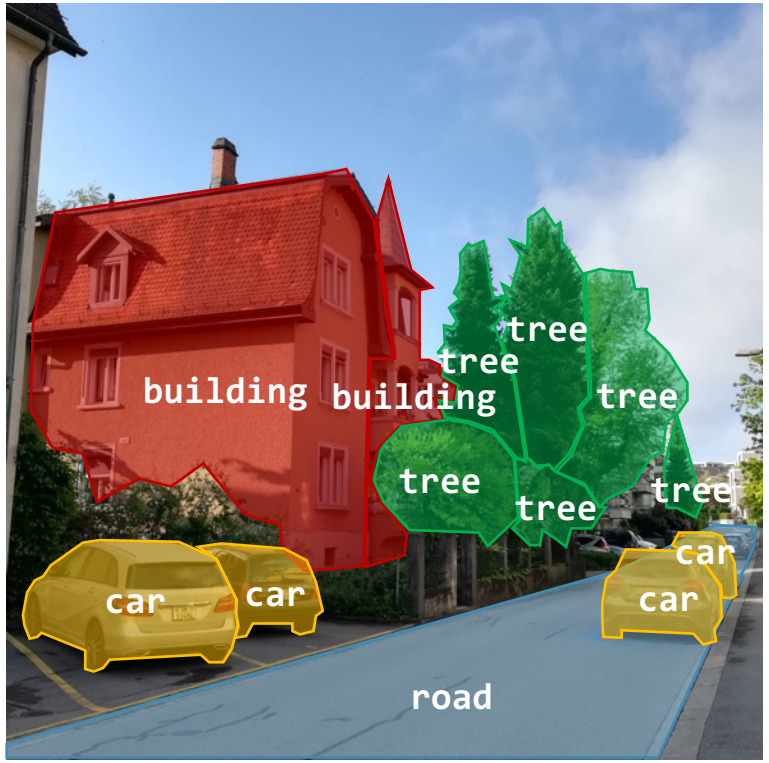
- 物体認識
- 畳み込みニューラルネットワーク
- 転移学習
- 物体検出

物体認識

物体検出



セグメンテーション



物体認識

物体検出

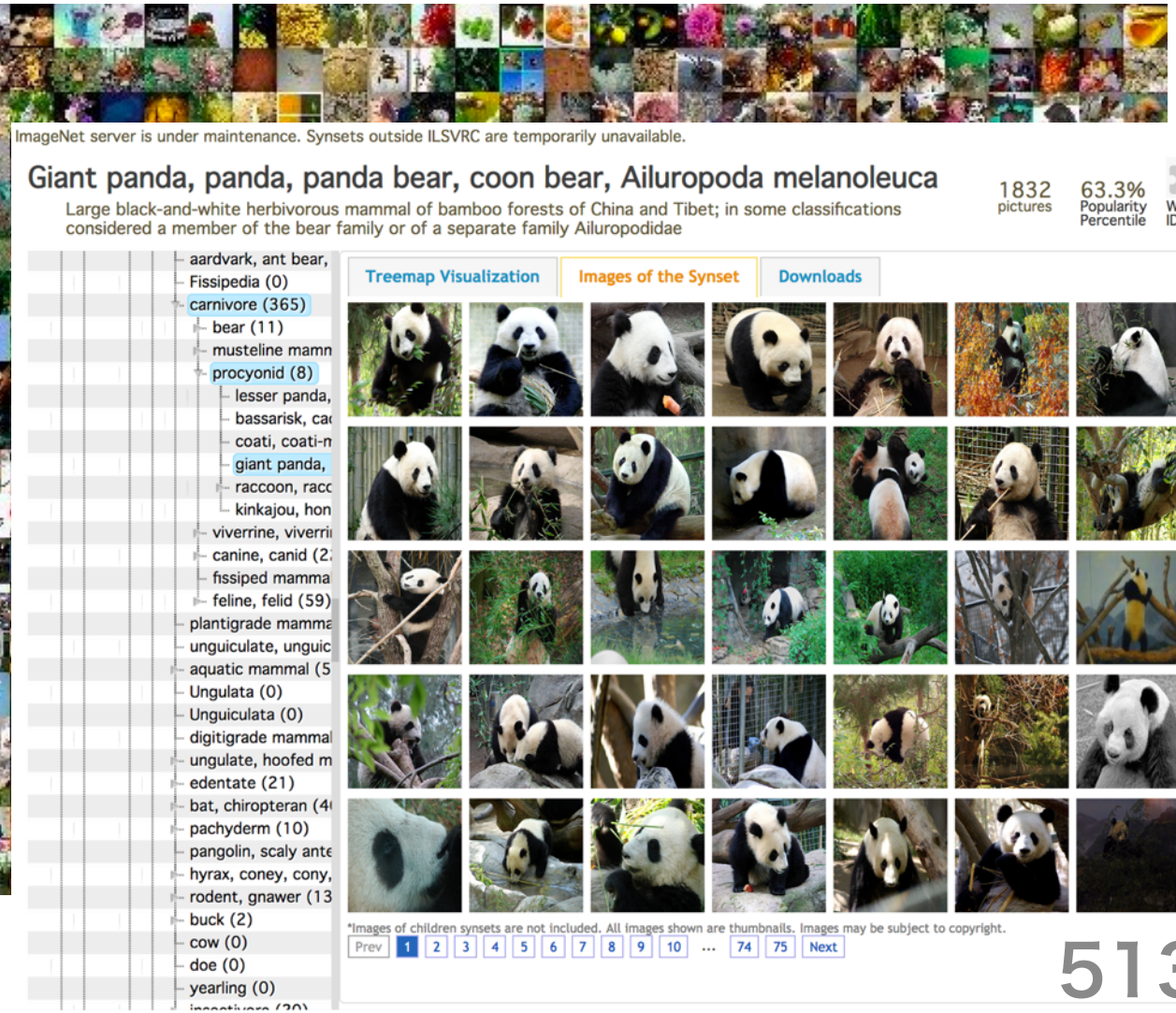
セグメンテーション

画像解析概略

- 物体認識
- 畳み込みニューラルネットワーク
- 転移学習
- 物体検出

画像認識コンテスト ILSVRC

ImageNet Large Scale Visual Recognition Challenge

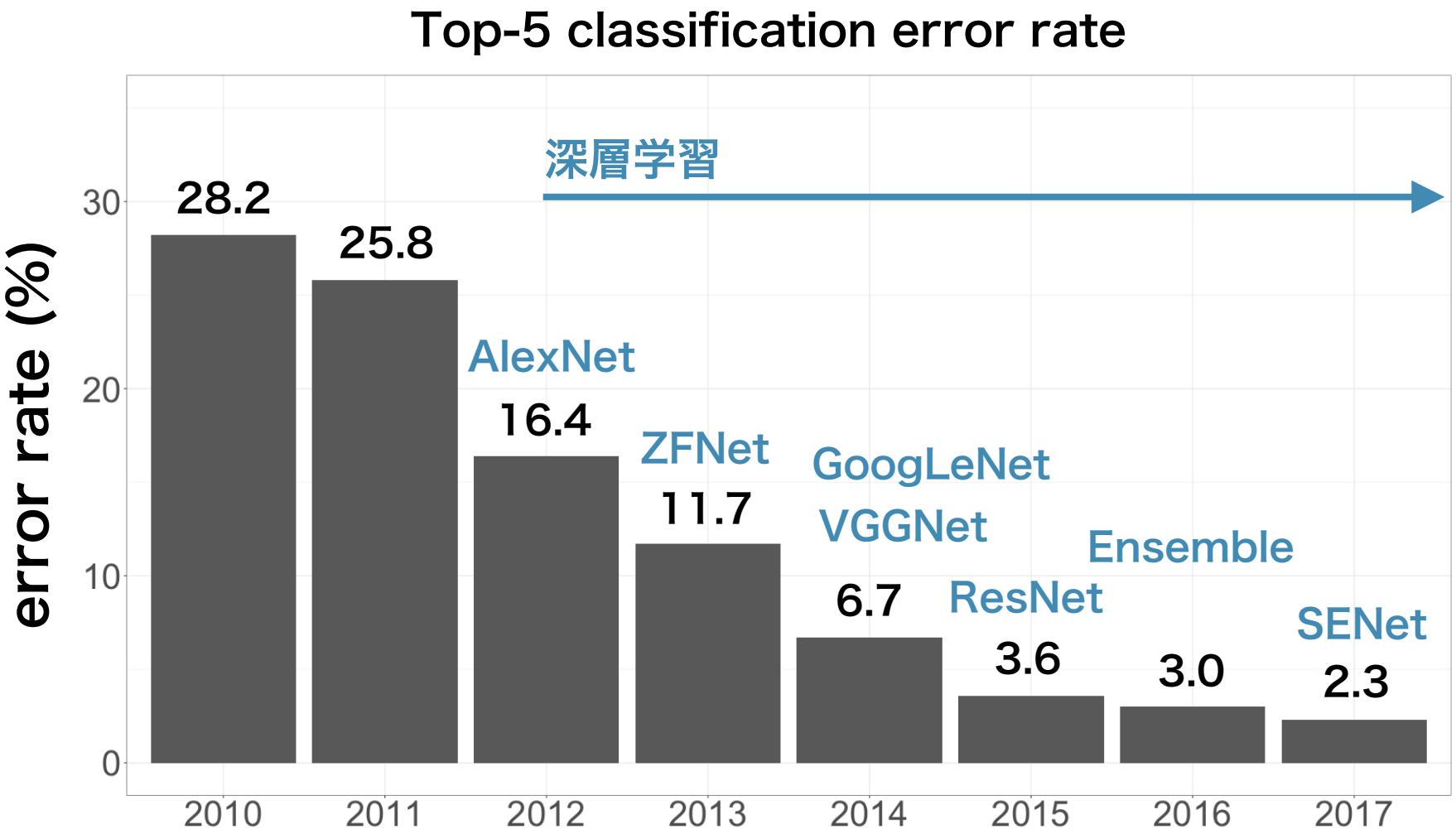


513

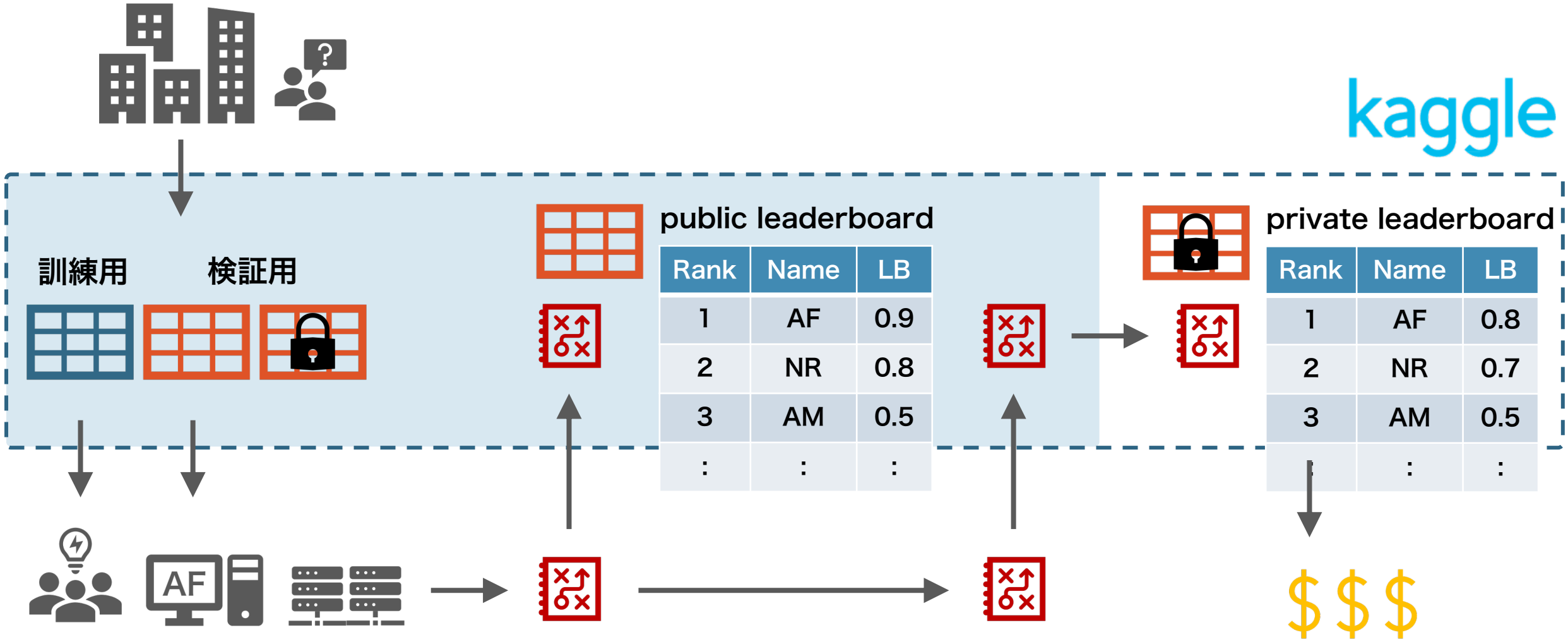
14,197,122 images / 21,841 classes

画像認識コンテスト ILSVRC

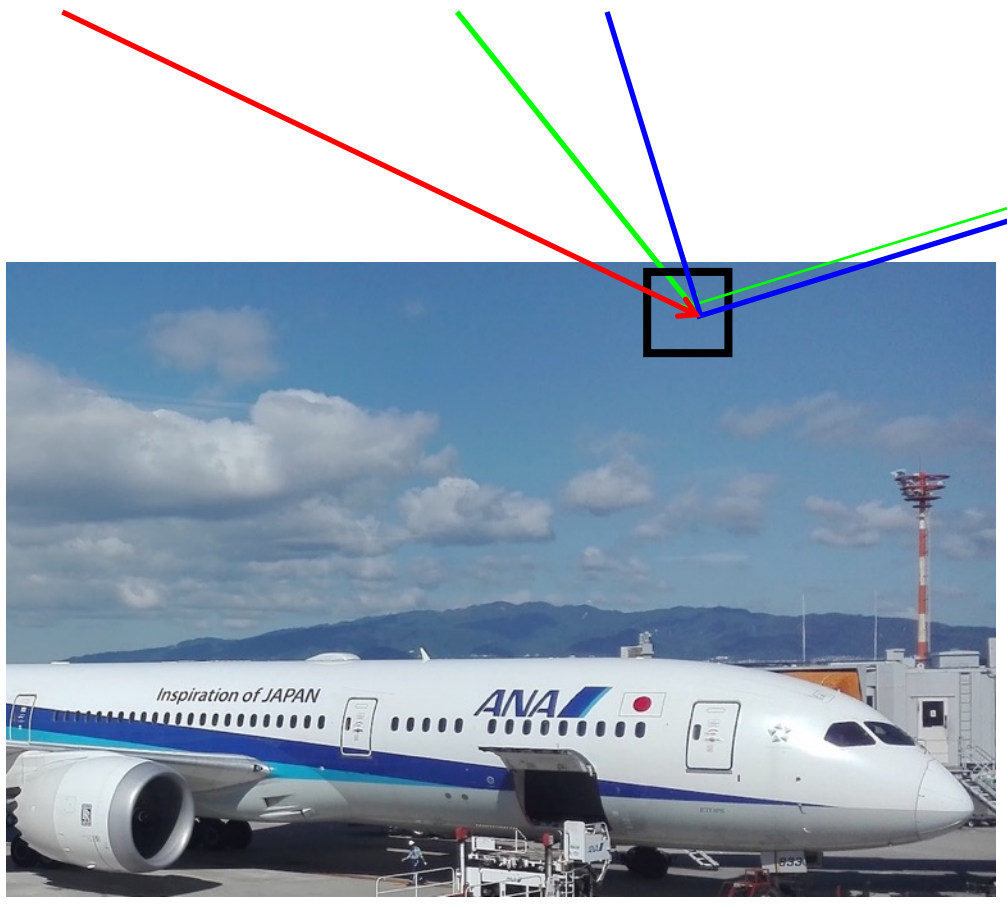
ImageNet Large Scale Visual Recognition Challenge



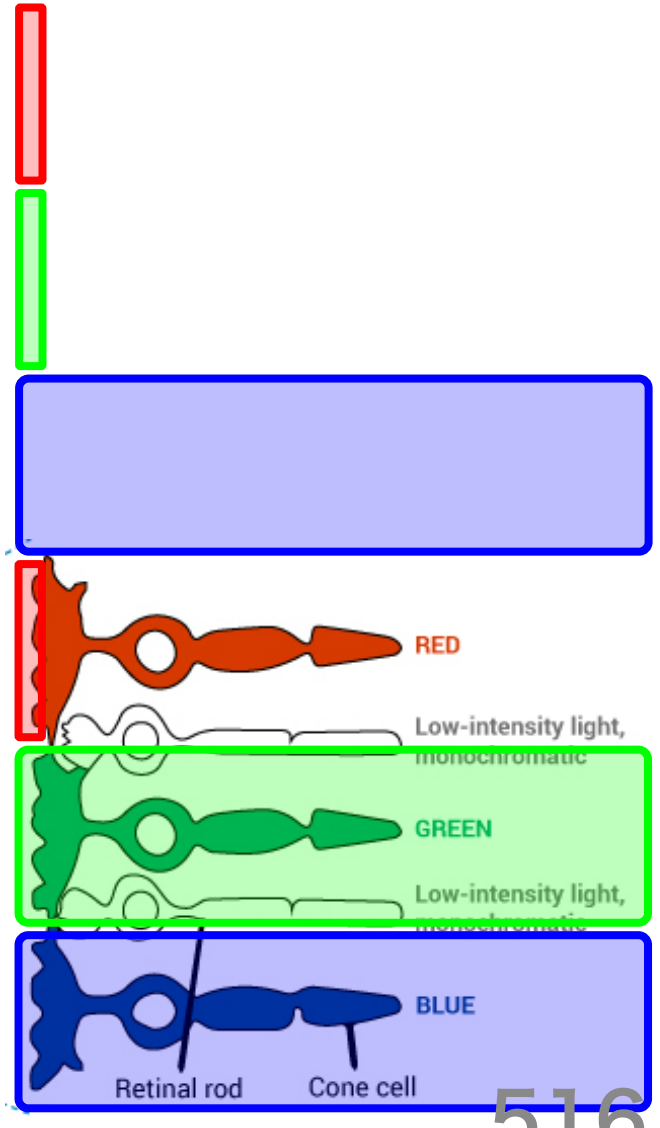
機械学習コンペティション



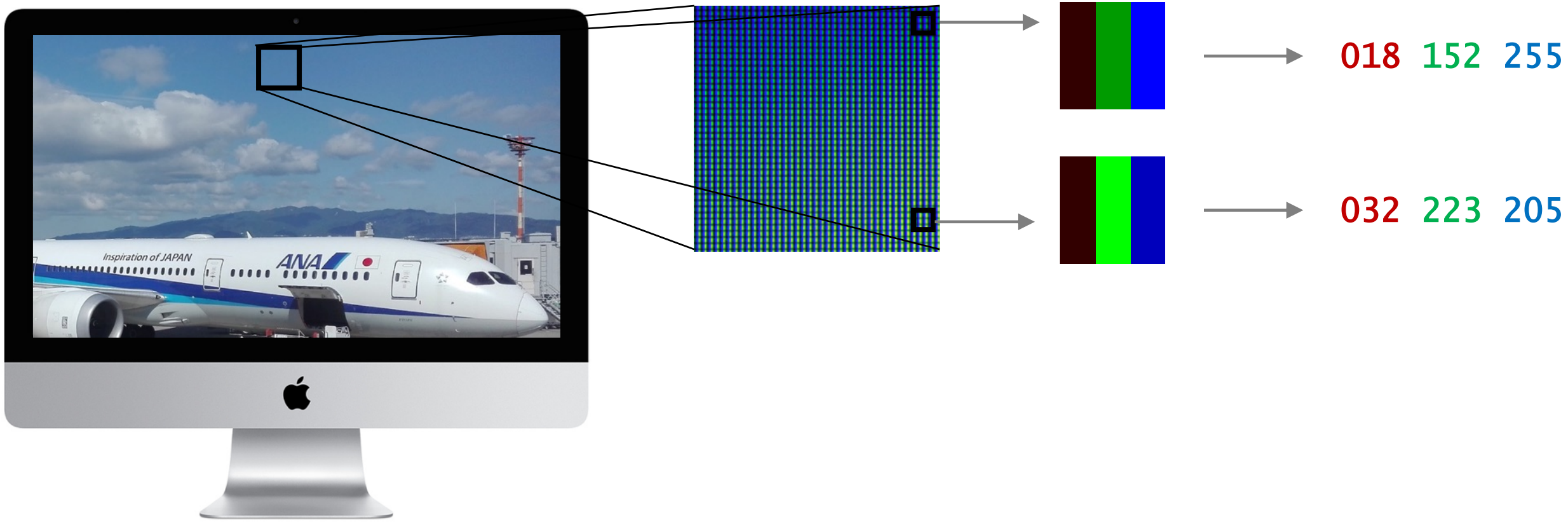
物体認識



青色反射光が強いので、ほぼすべての青色桿体細胞の神経が励起する。緑色反射光がそれほど強くないので、一部だけの緑色桿体細胞が励起する。赤色反射光がほとんどないので、赤色反対細胞がほとんど励起しない。脳は、励起する桿体細胞の割合を総合判断し、色として認識する。



物体認識



物体認識

021	031	022	063	042
041	065	034	044	033
082	024	056	072	038
046	046	033	049	029
025	061	012	103	052
031	051	053	081	041
093	049	058	092	036
054	101	061	024	041
109	023	020	052	022
111	042	024	021	024

162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

221	220	224	230	224
223	209	230	231	243
223	229	240	220	217
220	221	217	230	236
244	250	251	221	216
252	251	250	212	220
245	251	251	247	231
253	249	249	254	221
251	254	255	245	253
254	253	250	249	251

物体認識

物体認識

特徴量

特徴抽出

102
074
144
221
002
140
211
120
104
⋮
⋮

分類

飛行機

深層学習

CNN

飛行機

物体認識（特徴量）

021	031	022	063	042
041	065	034	044	033
082	024	056	072	038
046	046	033	049	029
025	061	012	103	052
031	051	053	081	041
093	049	058	092	036
054	101	061	024	041
109	023	020	052	022
111	042	024	021	024

162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

221	220	224	230	224
223	209	230	231	243
223	229	240	220	217
220	221	217	230	236
244	250	251	221	216
252	251	250	212	220
245	251	251	247	231
253	249	249	254	221
251	254	255	245	253
254	253	250	249	251

特徴抽出

SIFT　　HoG　　textons　　RIFT　　GLOH

物体認識（特徴量）

021	031	022	063	042
041	065	034	044	033
082	024	056	072	038
046	046	033	049	029
025	061	012	103	052
031	051	053	081	041
093	049	058	092	036
054	101	061	024	041
109	023	020	052	022
111	042	024	021	024

162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

221	220	224	230	224
223	209	230	231	243
223	229	240	220	217
220	221	217	230	236
244	250	251	221	216
252	251	250	212	220
245	251	251	247	231
253	249	249	254	221
251	254	255	245	253
254	253	250	249	251

特徴抽出

102
074
144
221
002
211
104
085
101
113
032
111
046
210
142
228
107
123
192
126
114
125
135
171
201

物体認識（特徴量）

021	031	022	063	042
041	065	034	044	033
082	024	056	072	038
046	046	033	049	029
025	061	012	103	052
031	051	053	081	041
093	049	058	092	036
054	101	061	024	041
109	023	020	052	022
111	042	024	021	024

162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

221	220	224	230	224
223	209	230	231	243
223	229	240	220	217
220	221	217	230	236
244	250	251	221	216
252	251	250	212	220
245	251	251	247	231
253	249	249	254	221
251	254	255	245	253
254	253	250	249	251

特徴抽出

102	002
074	103
144	211
221	104
002	085
211	101
104	113
085	032
101	144
113	221
032	052
111	111
046	146
210	104
142	142
228	228
107	107
123	123
192	015
126	135
114	211
125	201
135	192
171	162
201	141

物体認識（特徴量）

021	031	022	063	042
041	065	034	044	033
082	024	056	072	038
046	046	033	049	029
025	061	012	103	052
031	051	053	081	041
093	049	058	092	036
054	101	061	024	041
109	023	020	052	022
111	042	024	021	024

162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

221	220	224	230	224
223	209	230	231	243
223	229	240	220	217
220	221	217	230	236
244	250	251	221	216
252	251	250	212	220
245	251	251	247	231
253	249	249	254	221
251	254	255	245	253
254	253	250	249	251

特徴抽出

102	002	141
074	103	146
144	211	104
221	104	142
002	085	002
211	101	103
104	113	211
085	032	144
101	144	221
113	221	052
032	052	228
111	111	107
046	146	123
210	104	015
142	142	135
228	228	211
107	107	201
123	123	192
192	015	162
126	135	141
114	211	104
125	201	085
135	192	101
171	162	113
201	141	032

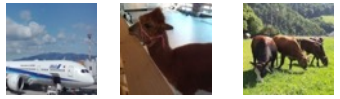
物体認識（特徴量）

021	031	022	063	042
041	065	034	044	033
082	024	056	072	038
046	046	033	049	029
025	061	012	103	052
031	051	053	081	041
093	049	058	092	036
054	101	061	024	041
109	023	020	052	022
111	042	024	021	024

162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

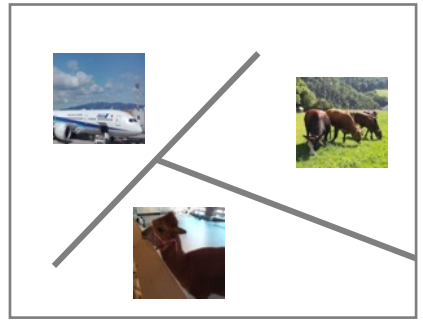
221	220	224	230	224
223	209	230	231	243
223	229	240	220	217
220	221	217	230	236
244	250	251	221	216
252	251	250	212	220
245	251	251	247	231
253	249	249	254	221
251	254	255	245	253
254	253	250	249	251

特徴抽出



102	002	141
074	103	146
144	211	104
221	104	142
002	085	002
211	101	103
104	113	211
085	032	144
101	144	221
113	221	052
032	052	228
111	111	107
046	146	123
210	104	015
142	142	135
228	228	211
107	107	201
123	123	192
192	015	162
126	135	141
114	211	104
125	201	085
135	192	101
171	162	113
201	141	032

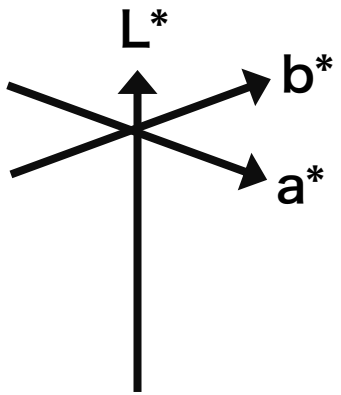
機械学習

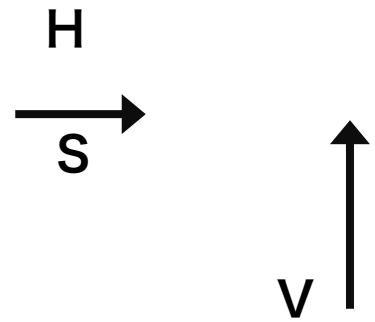


色空間

参考

RGB ⟷ L*a*b* ⟷ HSV





色の種類や明るさを変更するには、R, G, B の座標をすべて変更する必要がある。

色の種類と明るさを独立に扱える色空間。明るさを無視して、色だけに着目して解析するときに有効である。

色の種類、鮮やかさ、明るさを独立に扱える色空間。色だけに着目して解析するときに有効である。

物体認識

物体認識

特徴量

特徴抽出

102
074
144
221
002
140
211
120
104
⋮
⋮

分類

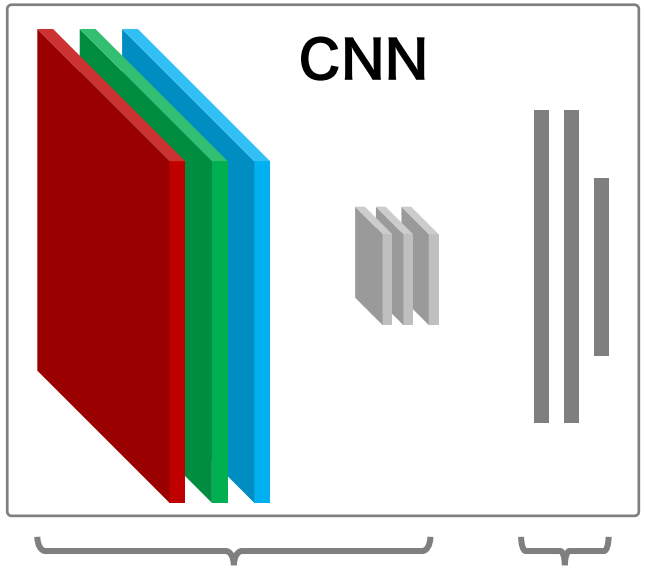
飛行機

深層学習

CNN

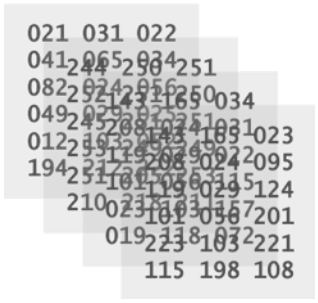
飛行機

物体認識（深層学習）

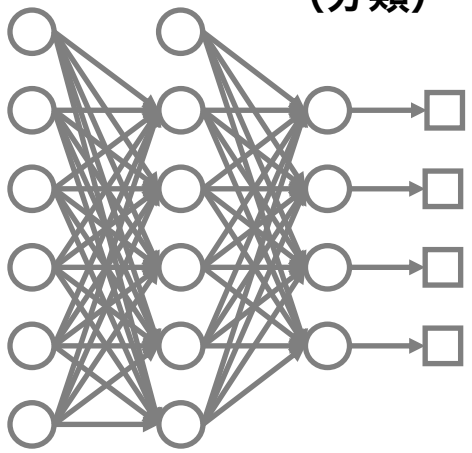



```
airplane 56.1%
sky      25.0%
cloud    12.4%
mountain  6.5%
```

畳み込み
（特徴抽出）



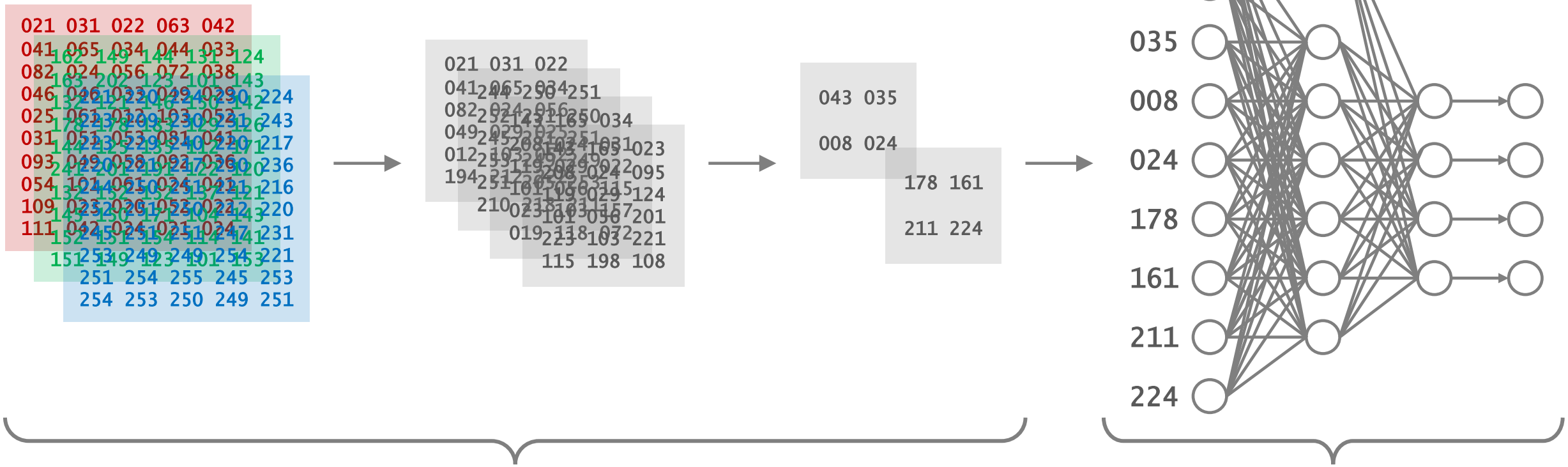
ニューラルネットワーク
（分類）



画像解析概略

- 物体認識
- 畳み込みニューラルネットワーク
- 転移学習
- 物体検出

畳み込みニューラルネットワーク

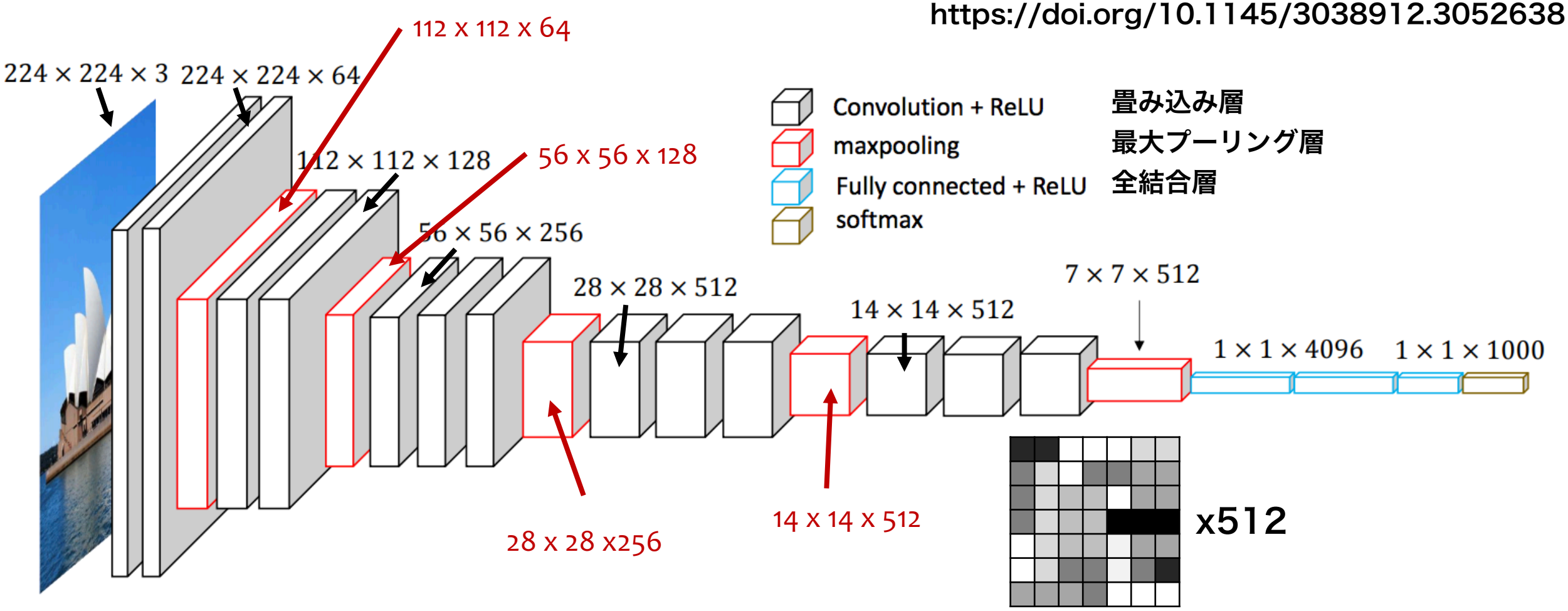


畳み込み
（特徴抽出）

ニューラルネットワーク
（分類）

畳み込みニューラルネットワーク

https://doi.org/10.1145/3038912.3052638



畳み込みニューラルネットワーク

畳み込み層

- 画像から特徴量を抽出する層
- 入力層に近い方の畳み込み層は、縦線、横線、斜線などの抽象的な特徴を抽出
- 出力層に近い方の畳み込み層は、顔の形や車の形などの具体的な特徴を抽出

プーリング層

- 重要な特徴量を残しながら画像のサイズを削減する層
- 後に続く処理が少数のパラメーターで計算できるようになり、過学習を抑制

全結合層

- 畳み込み層およびプーリング層から抽出された特徴量をニューラルネットワークで分類する層

Grayscale Photo

162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

1 channel

Color Photo

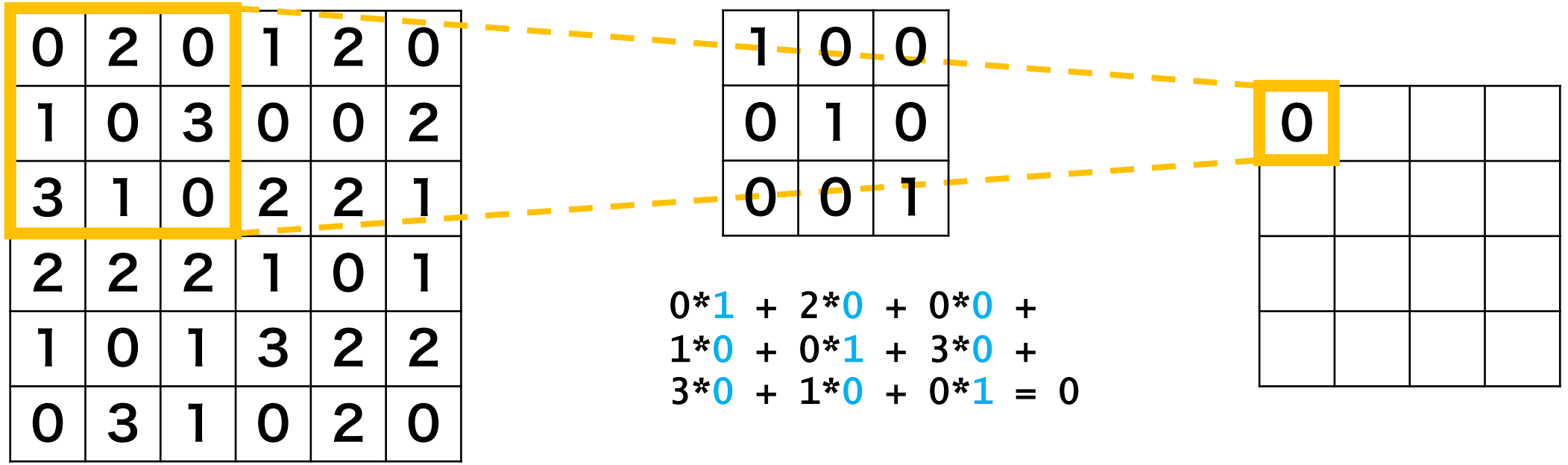
021	031	022	063	042
041	065	034	044	033
082	024	056	072	038
046	046	033	049	029
025	061	013	103	053
031	051	053	081	041
093	049	058	093	036
054	101	065	024	041
109	023	026	053	023
111	042	024	021	024

162	149	144	131	124
163	202	123	101	143
132	121	146	150	142
178	178	183	129	126
144	125	135	112	171
241	201	191	122	120
132	152	152	137	121
145	150	171	104	143
152	151	154	114	141
151	149	123	101	153

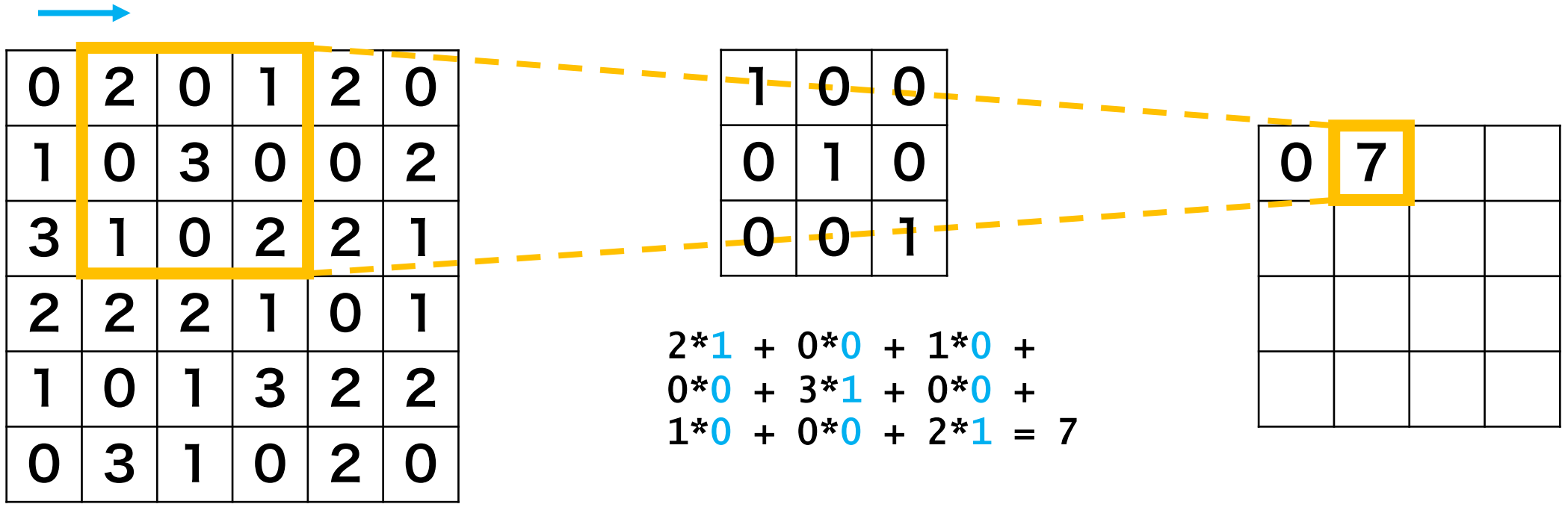
224	243	217	236	216
220	231	221	251	254
255	245	253	254	253
250	249	251		

3 channels

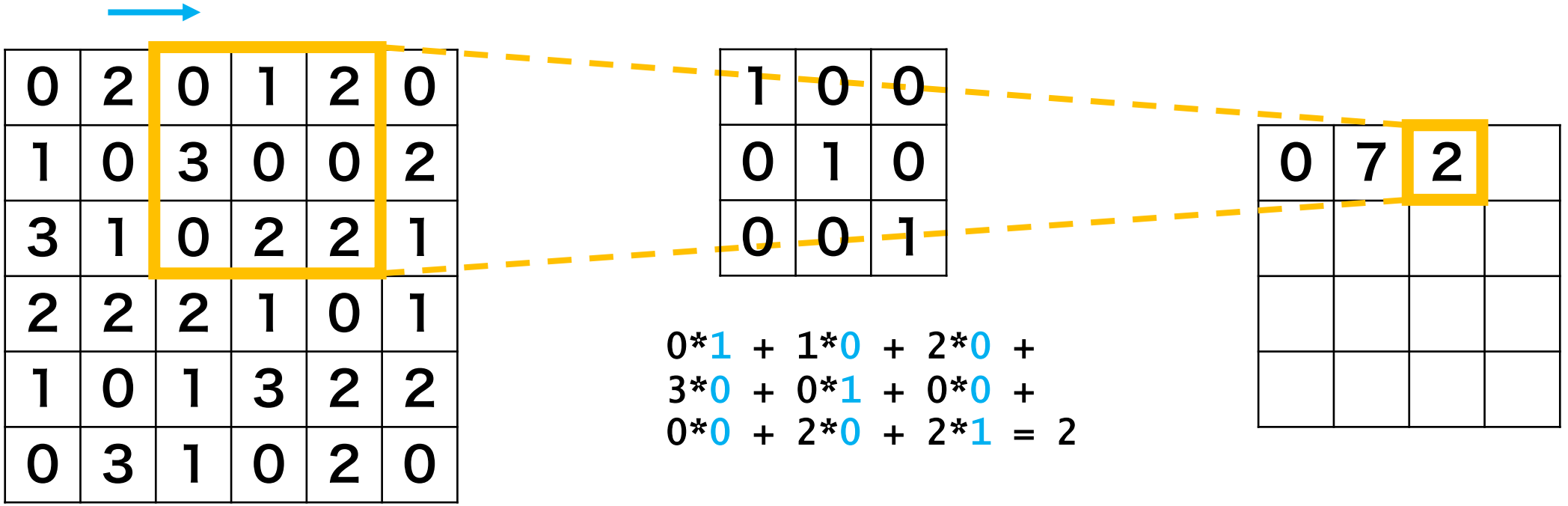
畳み込み演算


0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0*1 + 2*0 + 0*0 +
1*0 + 0*1 + 3*0 +
3*0 + 1*0 + 0*1 = 0
0

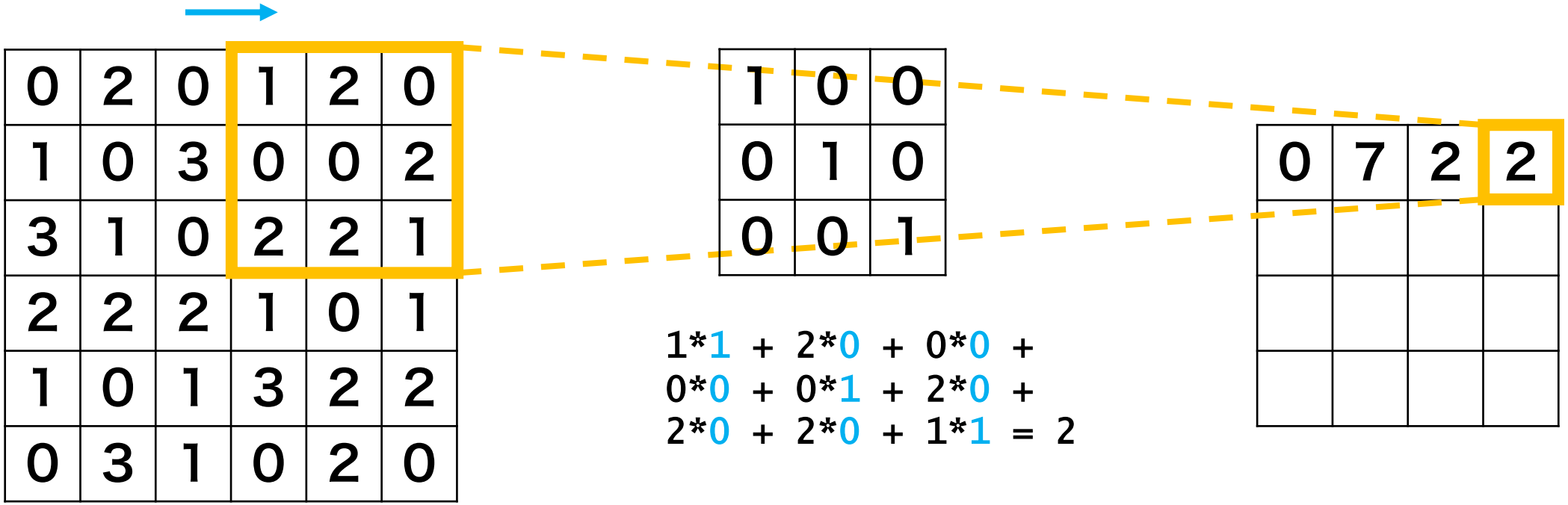
畳み込み演算


0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
2*1 + 0*0 + 1*0 +
0*0 + 3*1 + 0*0 +
1*0 + 0*0 + 2*1 = 7
0 7

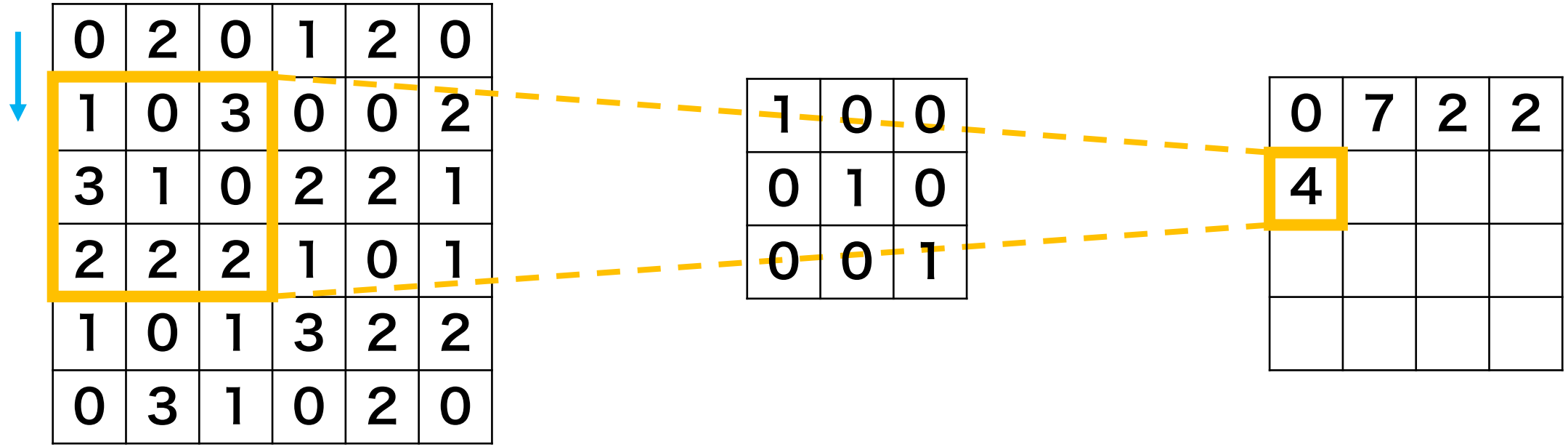
畳み込み演算


0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0*1 + 1*0 + 2*0 +
3*0 + 0*1 + 0*0 +
0*0 + 2*0 + 2*1 = 2
0 7 2

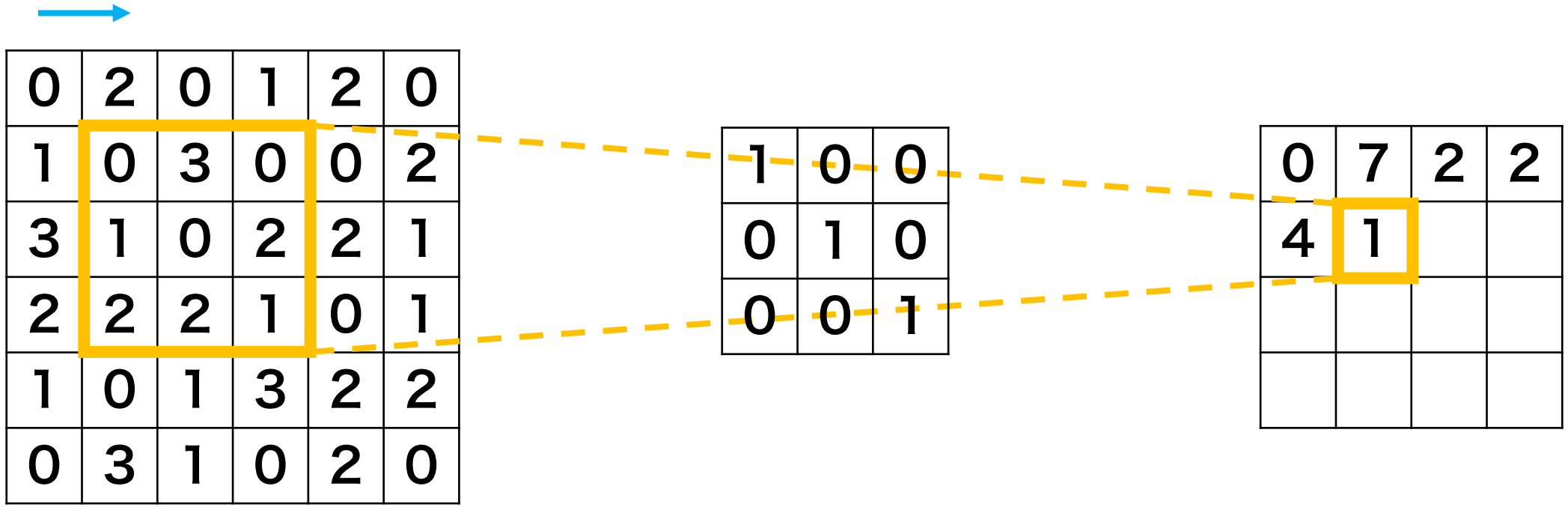
畳み込み演算


0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
1*1 + 2*0 + 0*0 +
0*0 + 0*1 + 2*0 +
2*0 + 2*0 + 1*1 = 2
0 7 2 2

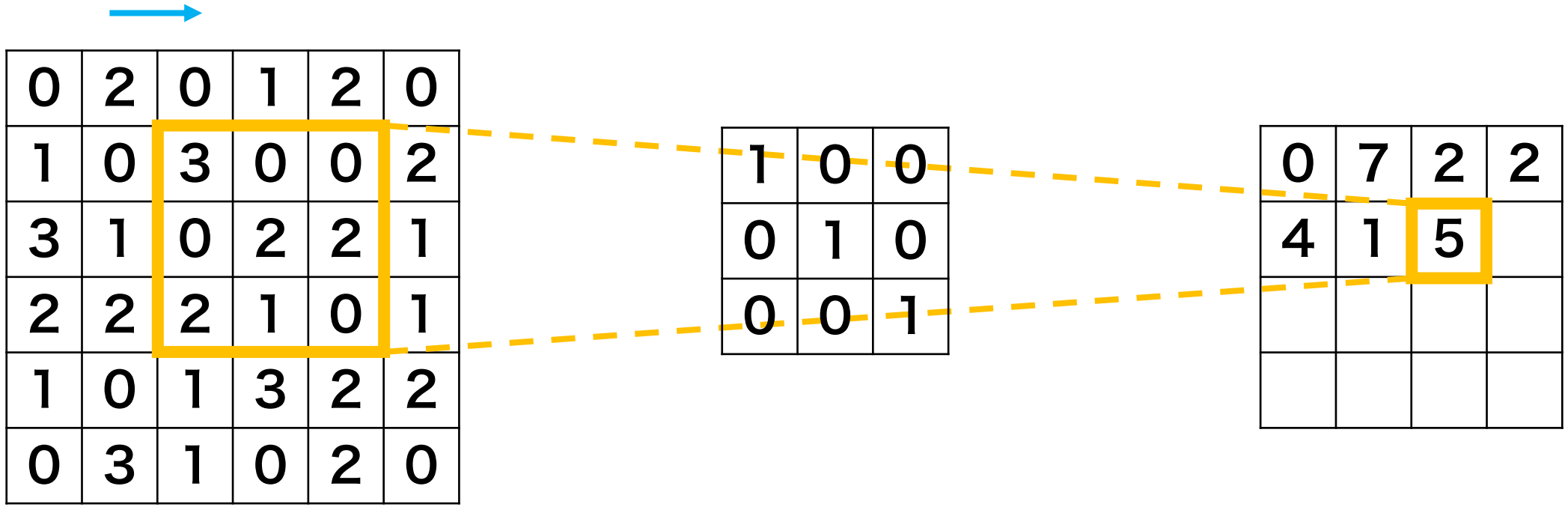
畳み込み演算


0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0 7 2 2
4

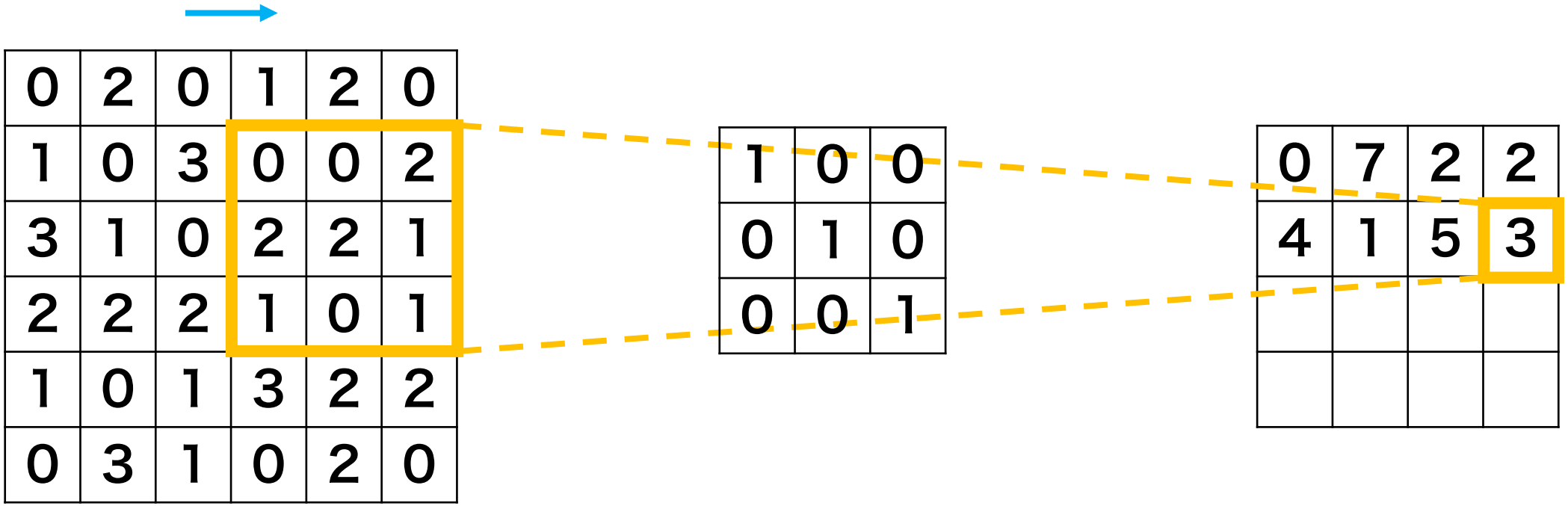
畳み込み演算


0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0 7 2 2
4 1

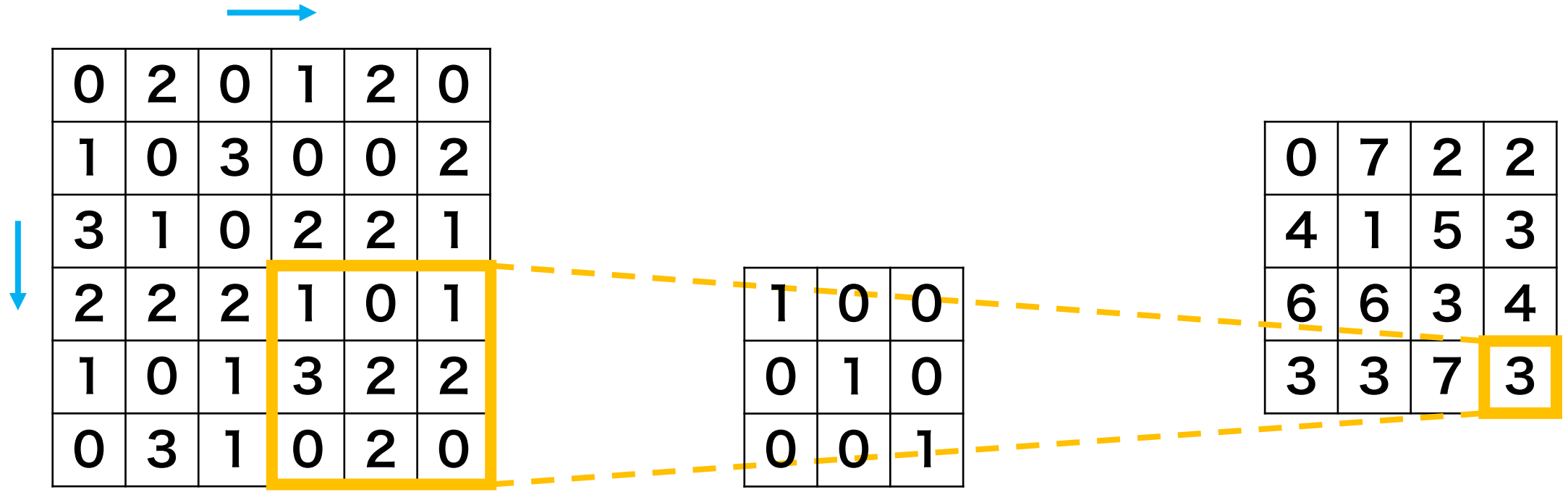
畳み込み演算


0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0 7 2 2
4 1 5

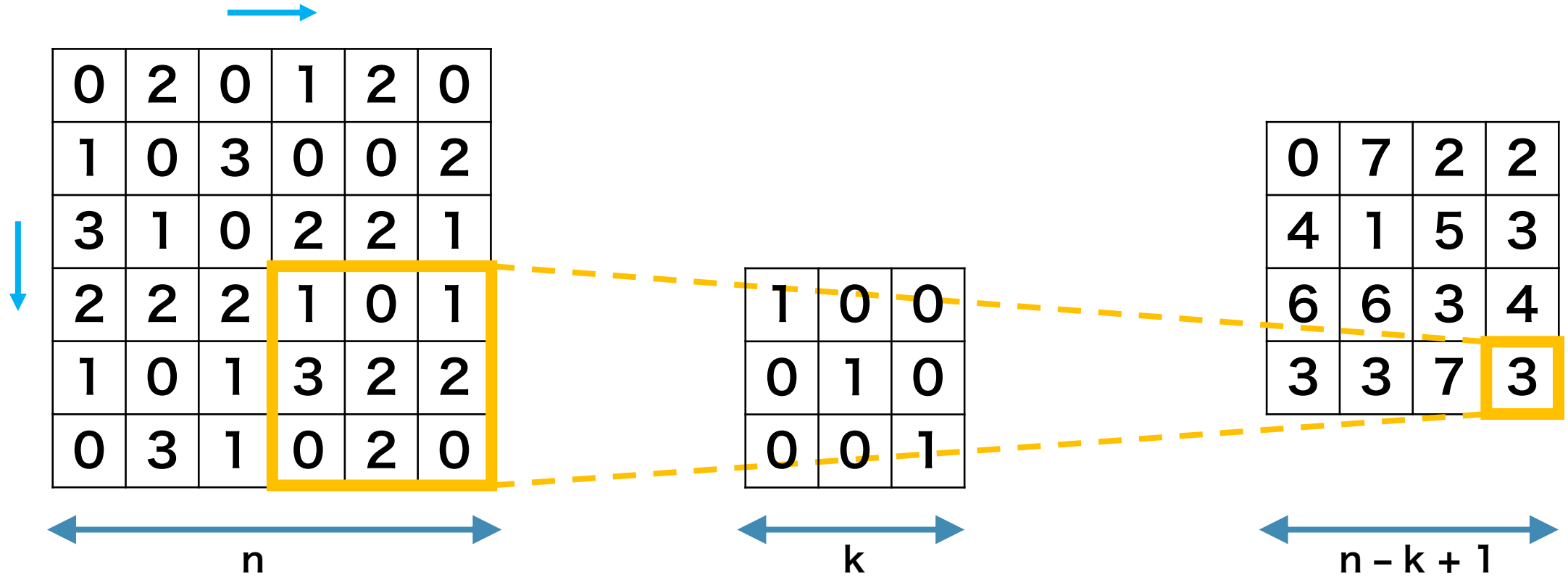
畳み込み演算


0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0 7 2 2
4 1 5 3

畳み込み演算


0 2 0 1 2 0
1 0 3 0 0 2
3 1 0 2 2 1
2 2 2 1 0 1
1 0 1 3 2 2
0 3 1 0 2 0
1 0 0
0 1 0
0 0 1
0 7 2 2
4 1 5 3
6 6 3 4
3 3 7 3

畳み込み演算



畳み込み演算を行うことによって、6x6 の入力行列が 4x4 の行列に縮小される。そのため、畳み込みを複数回行うことによって、画像サイズが大幅に小さくなるだけでなく、画像中央にある特徴が極端に強調され、画像のエッジにある情報が徐々に消えてしまうリスクが生じる。

畳み込み演算

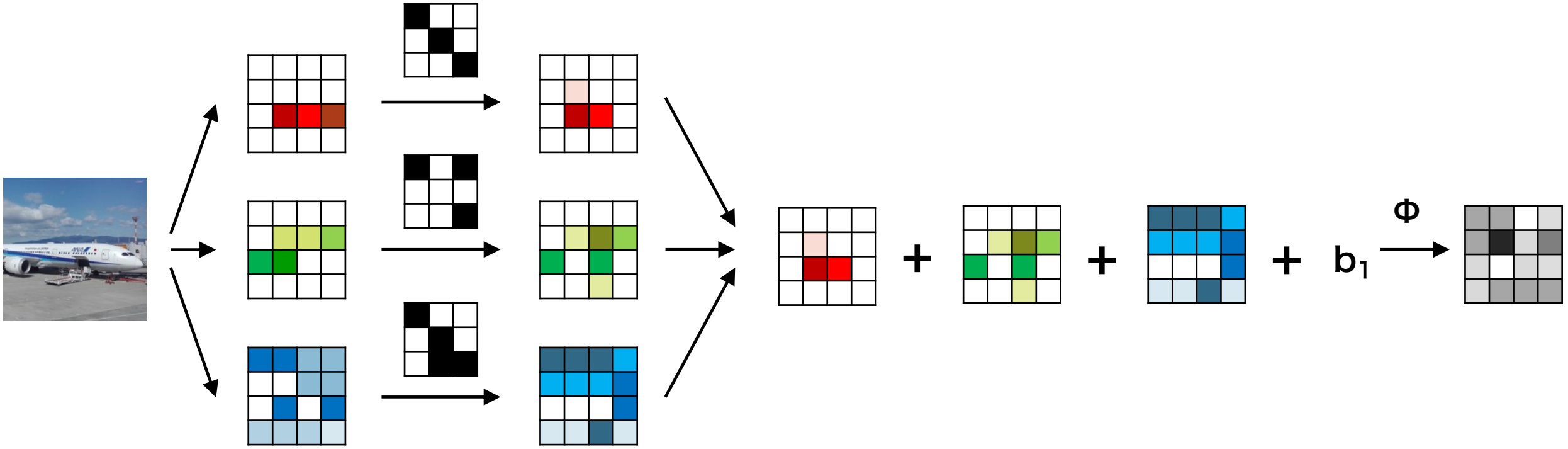
0	0	0	0	0	0	0	0
0	0	2	0	1	2	0	0
0	1	0	3	0	0	2	0
0	3	1	0	2	2	1	0
0	2	2	2	1	0	1	0
0	1	0	1	3	2	2	0
0	0	3	1	0	2	0	0
0	0	0	0	0	0	0	0

1	0	0
0	1	0
0	0	1

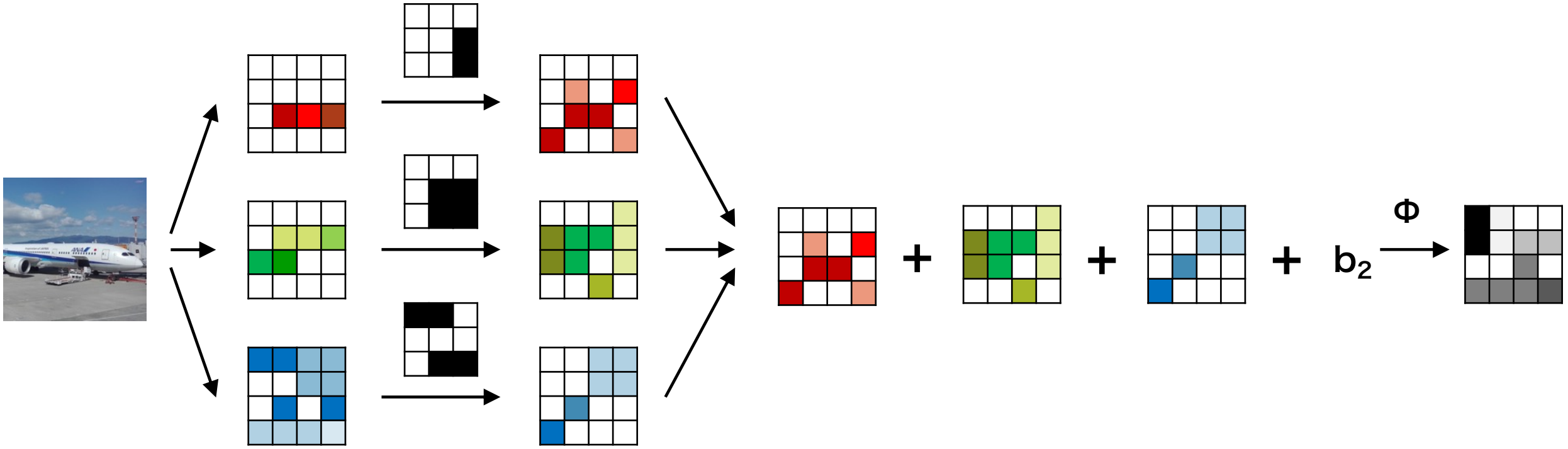
0	5	0	1	2	0
2	0	7	2	2	4
5	4	1	5	3	1
2	6	6	3	4	3
4	3	3	7	3	2
0	4	1	1	5	0

入力行列の周りにゼロを埋めることで、畳み込み後の行列のサイズは変化しない。

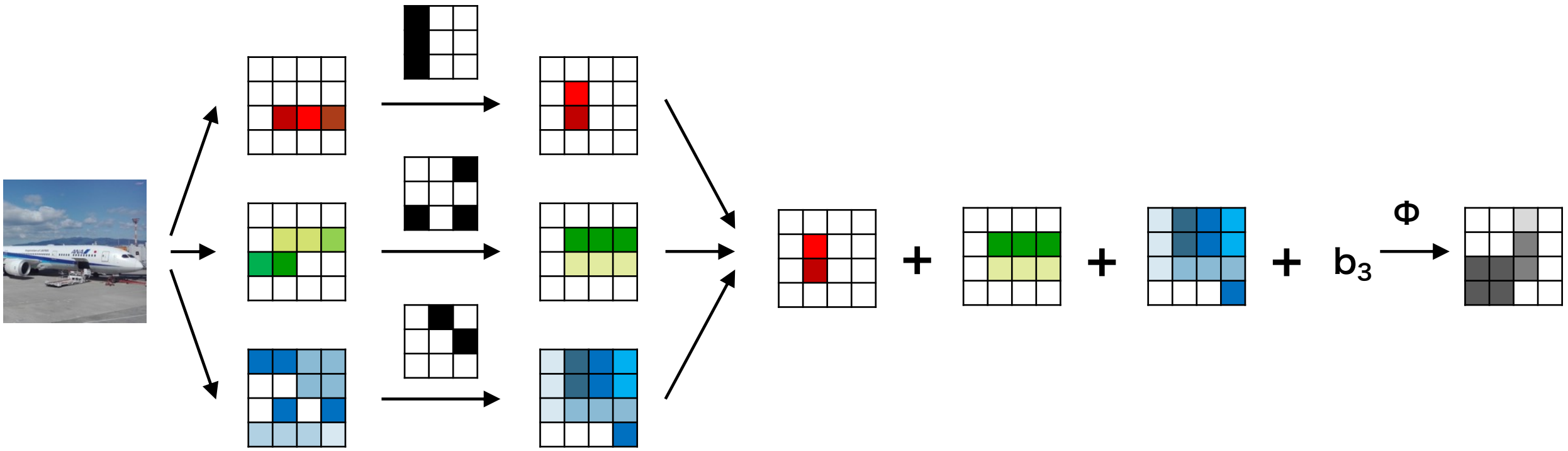
畳み込み演算



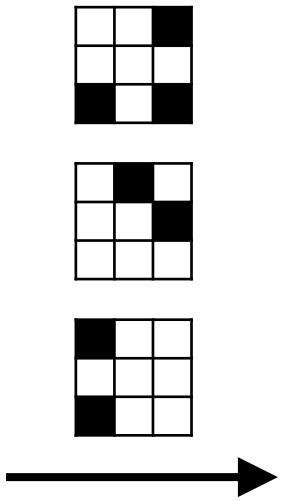
畳み込み演算



畳み込み演算

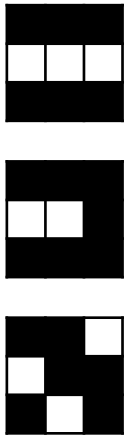


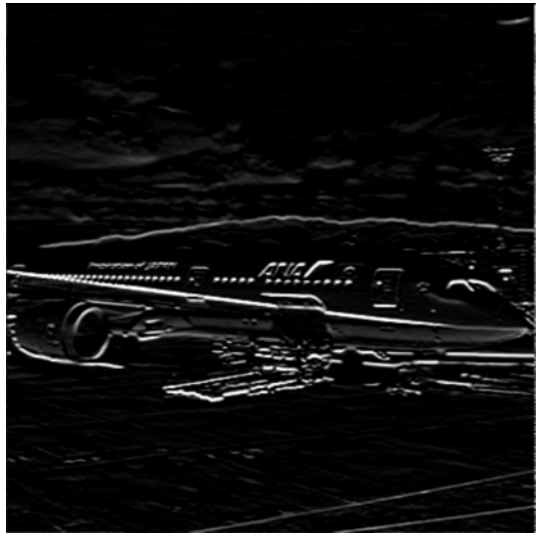
畳み込み演算



カーネル行列を変えることで、1枚の画像から様々な特徴を持つ画像を生成できる。

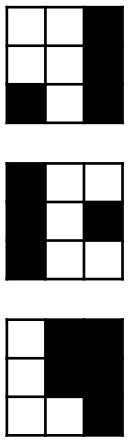
畳み込み演算

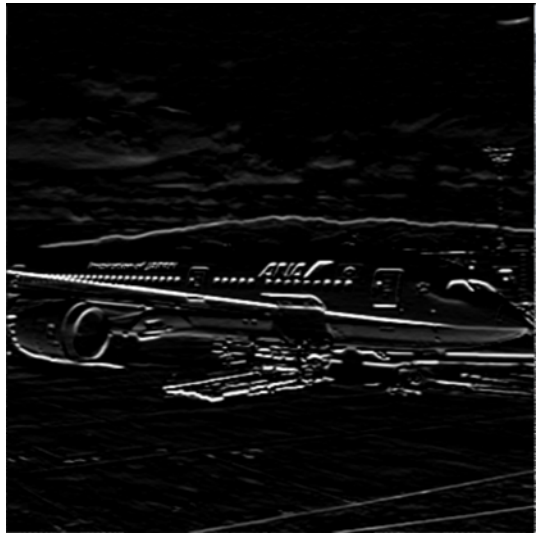




カーネル行列を変えることで、1枚の画像から様々な特徴を持つ画像を生成できる。

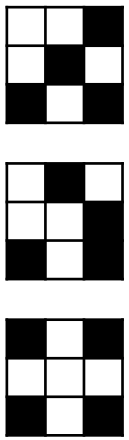
畳み込み演算

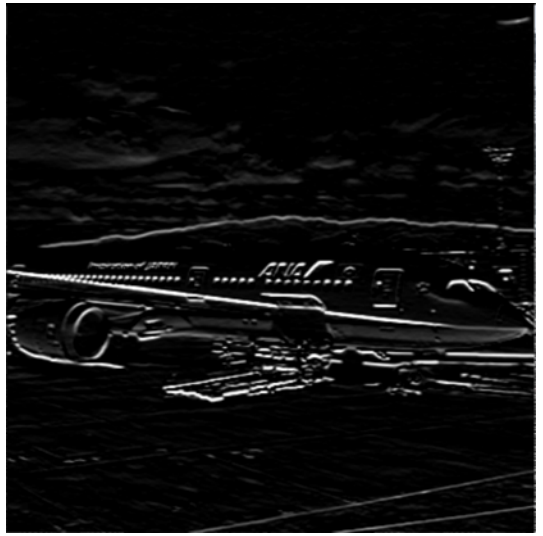




カーネル行列を変えることで、1枚の画像から様々な特徴を持つ画像を生成できる。

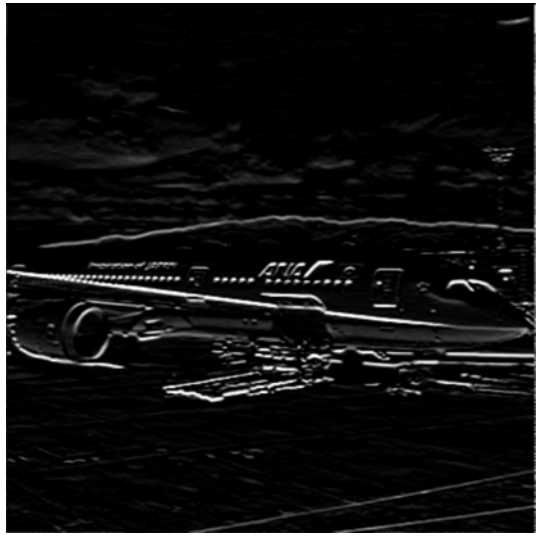
畳み込み演算





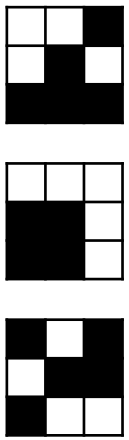
カーネル行列を変えることで、1枚の画像から様々な特徴を持つ画像を生成できる。

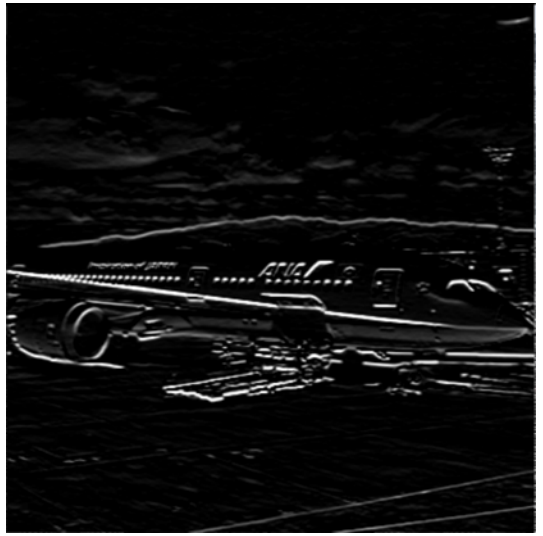
畳み込み演算



カーネル行列を変えることで、1枚の画像から様々な特徴を持つ画像を生成できる。

畳み込み演算

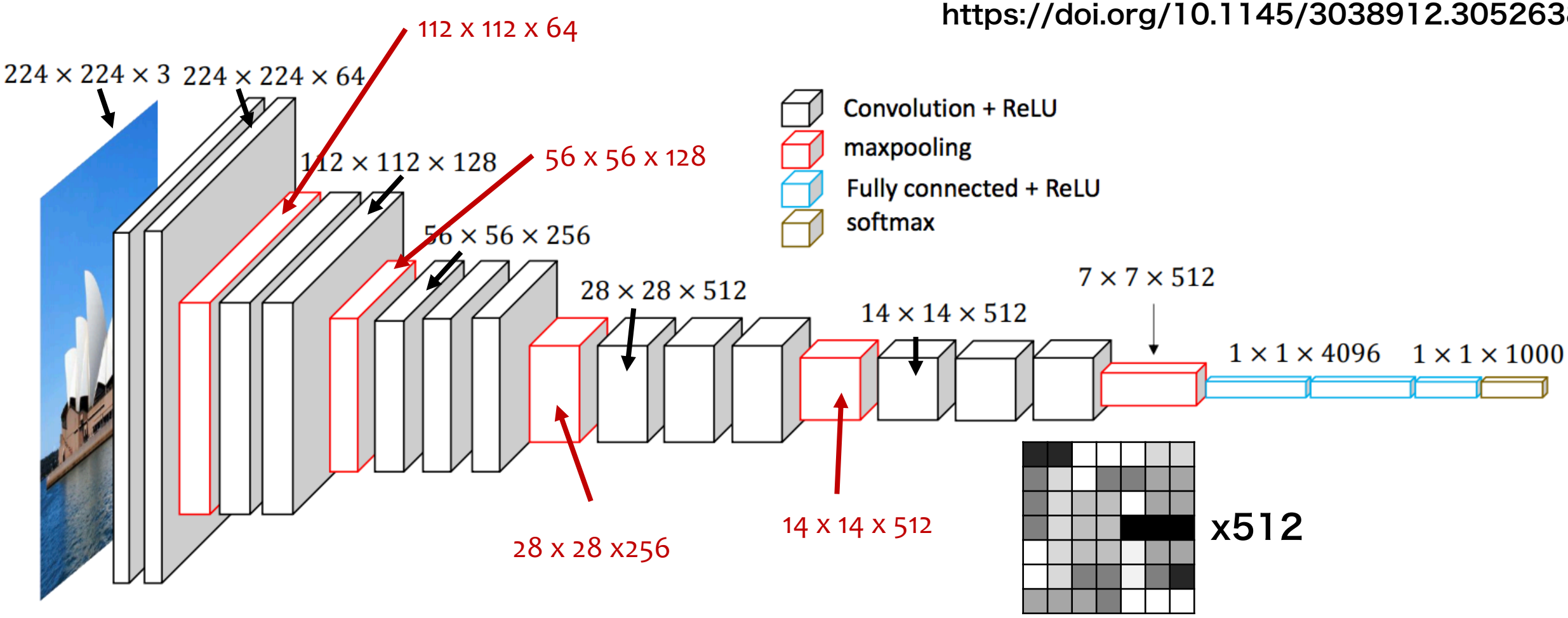




カーネル行列を変えることで、1枚の画像から様々な特徴を持つ画像を生成できる。

畳み込みニューラルネットワーク

https://doi.org/10.1145/3038912.3052638

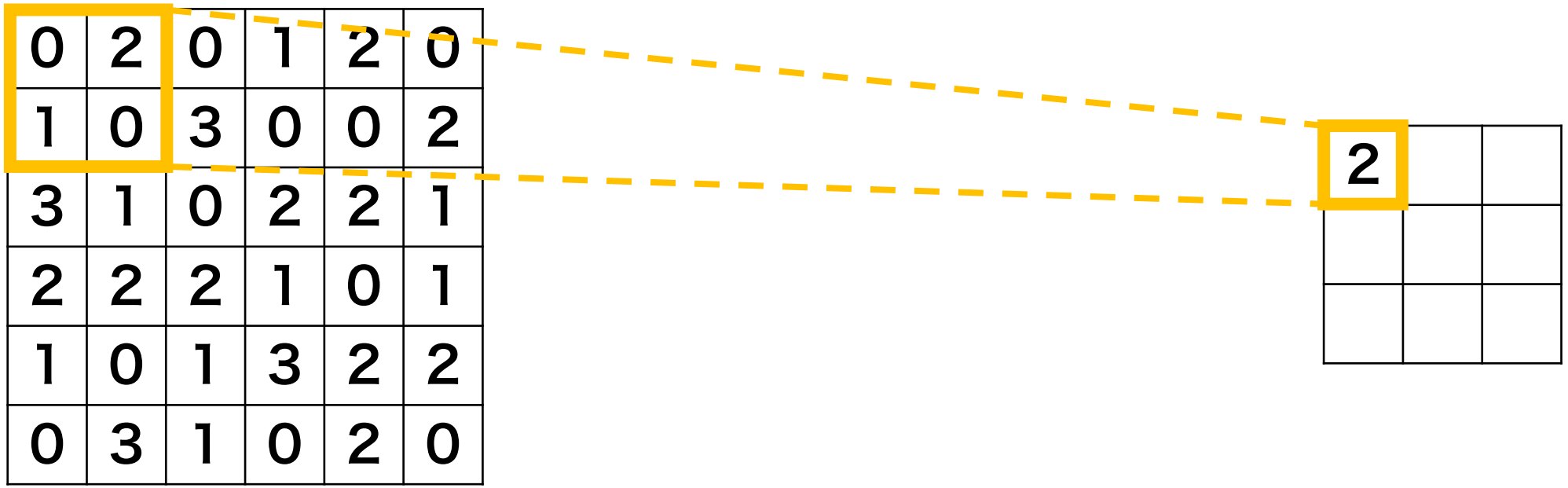


特徴抽出

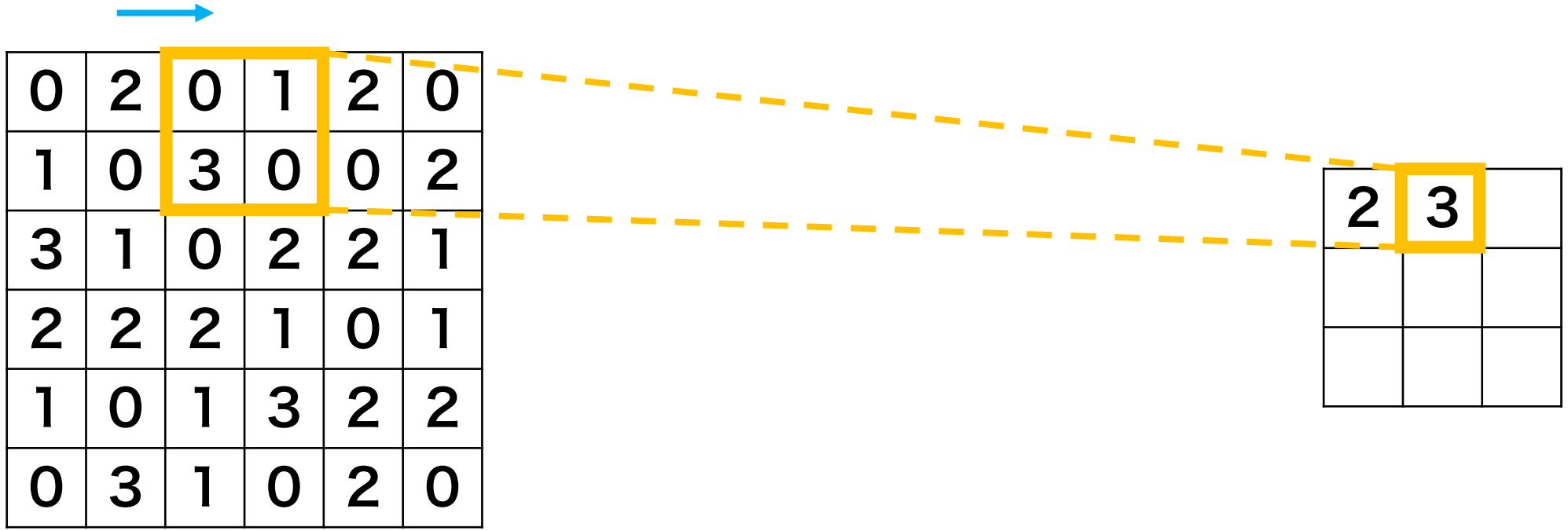
分類

プーリング演算

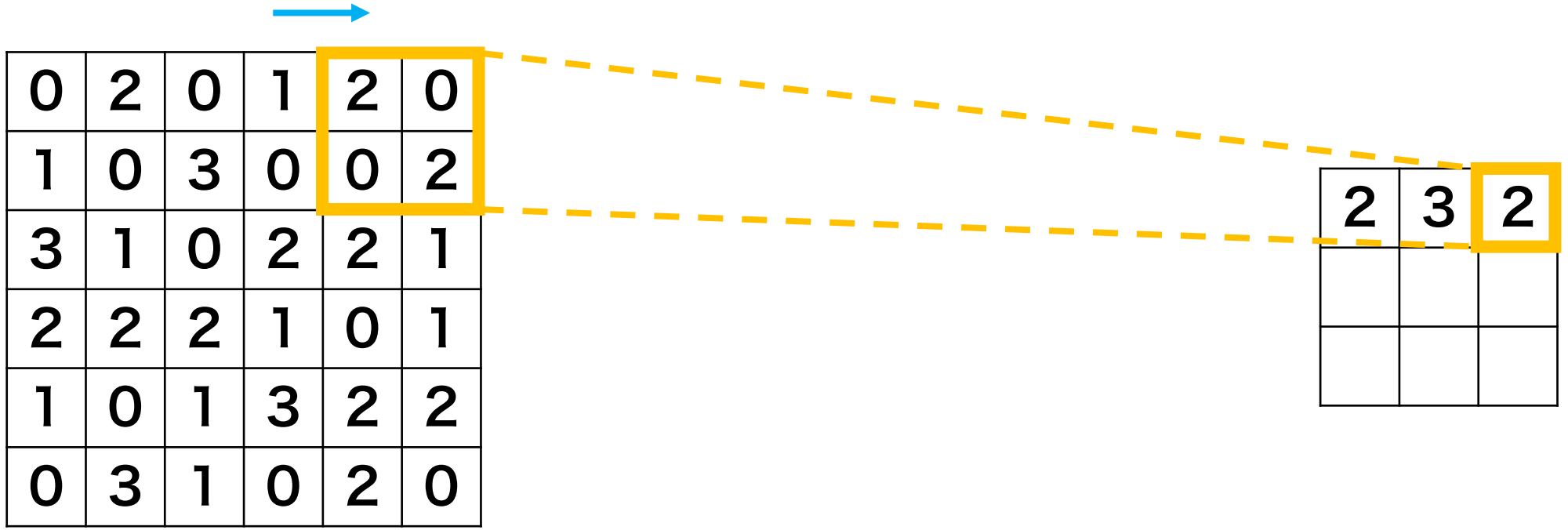
画像の局所的な部分が少し変化しても、その最大値プーリング結果が変化しないため、最大値プーリングは入力データのノイズに対してより頑健な特徴量を生成するのに役立つ。



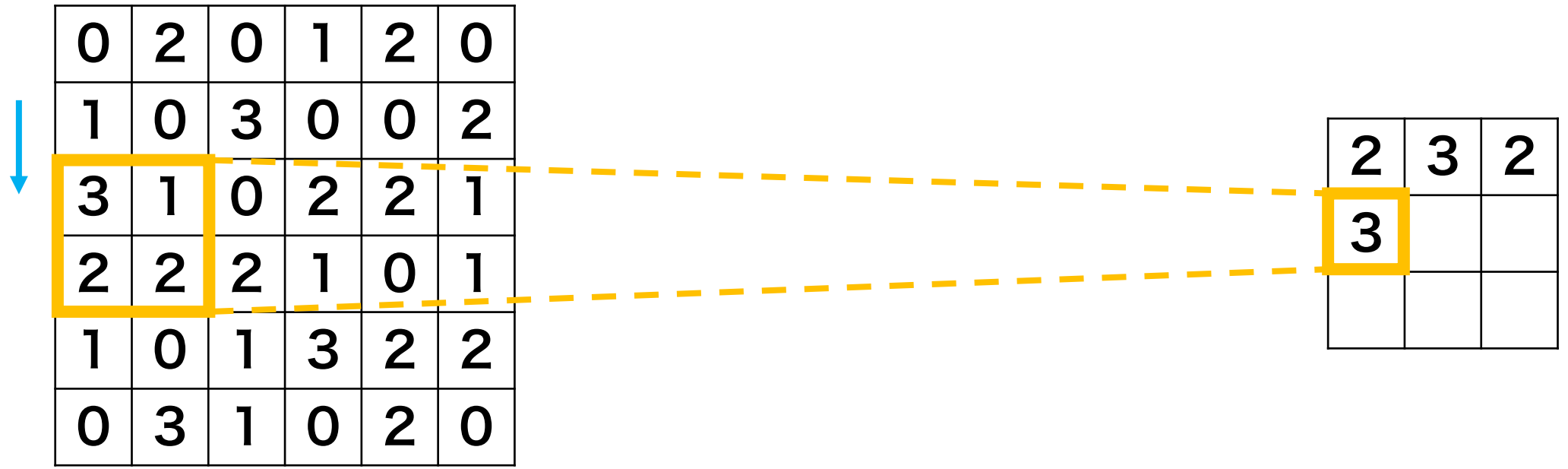
プーリング演算



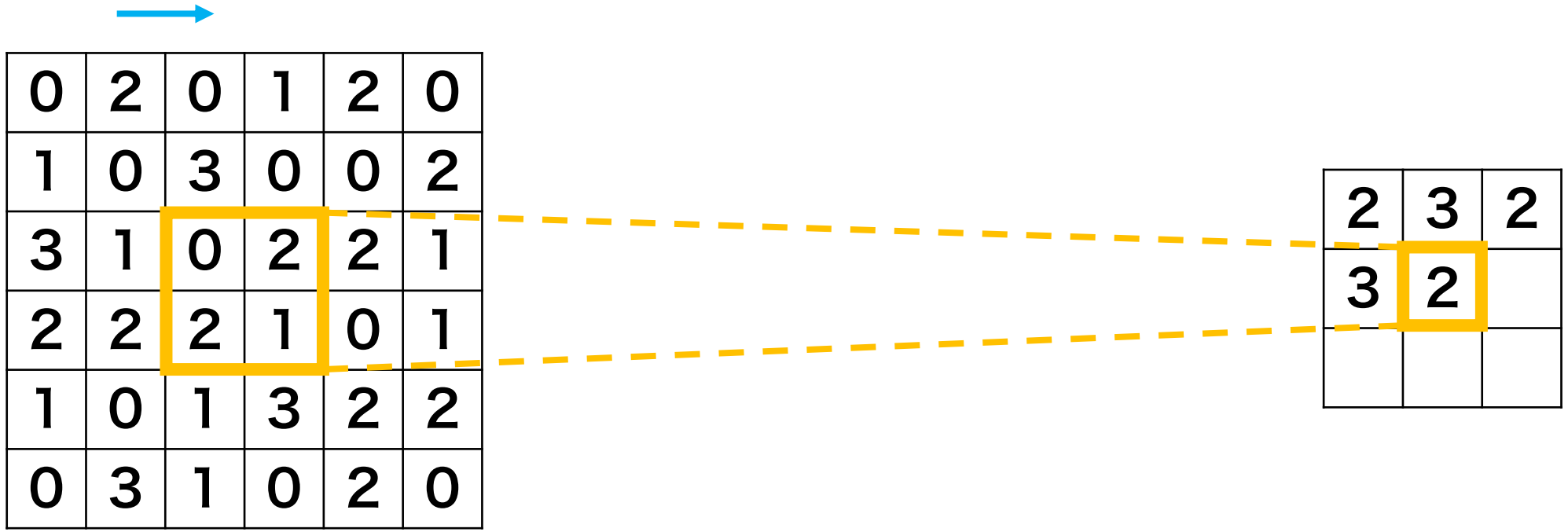
プーリング演算



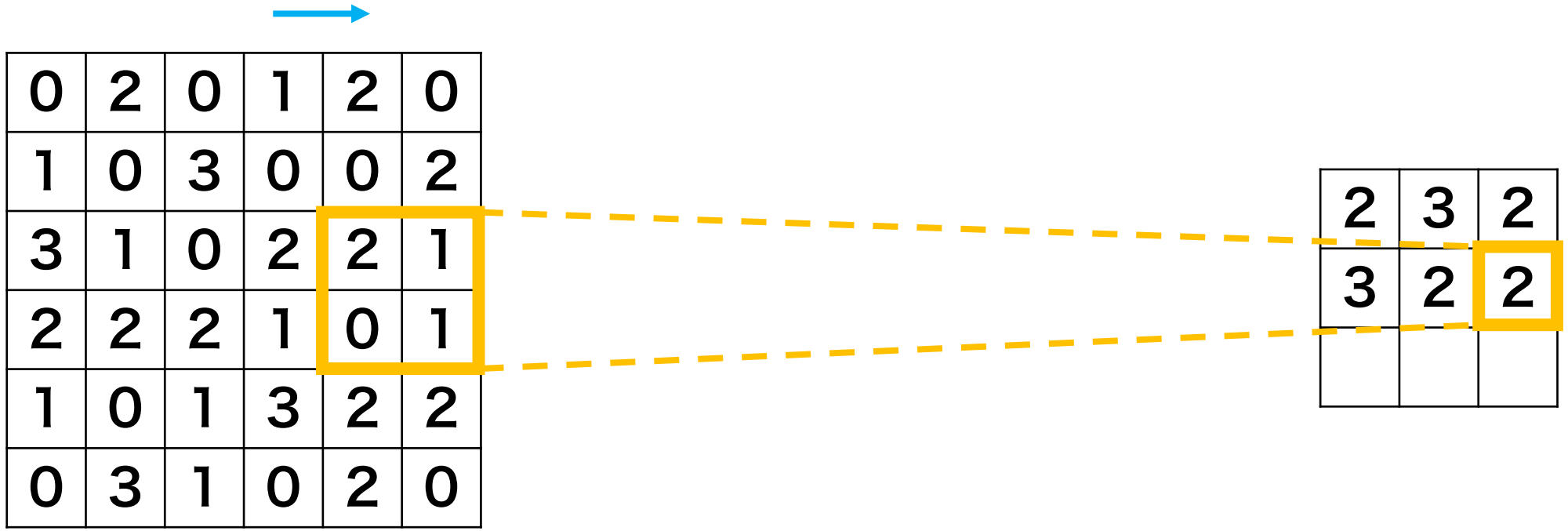
プーリング演算



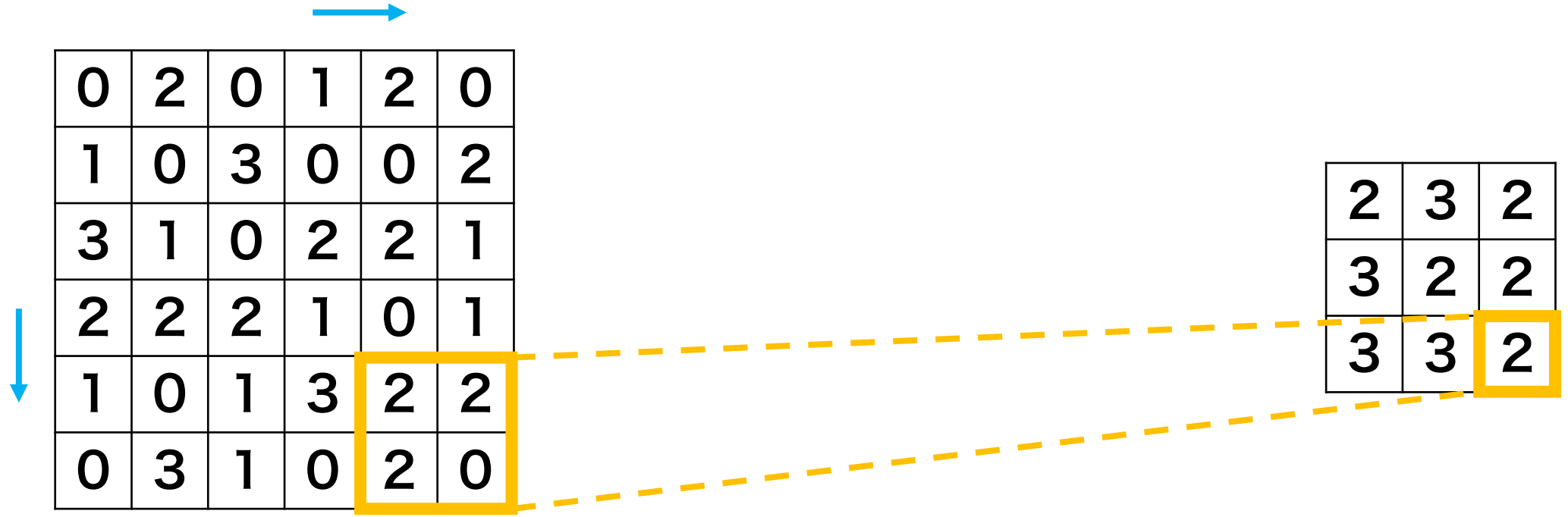
プーリング演算



プーリング演算

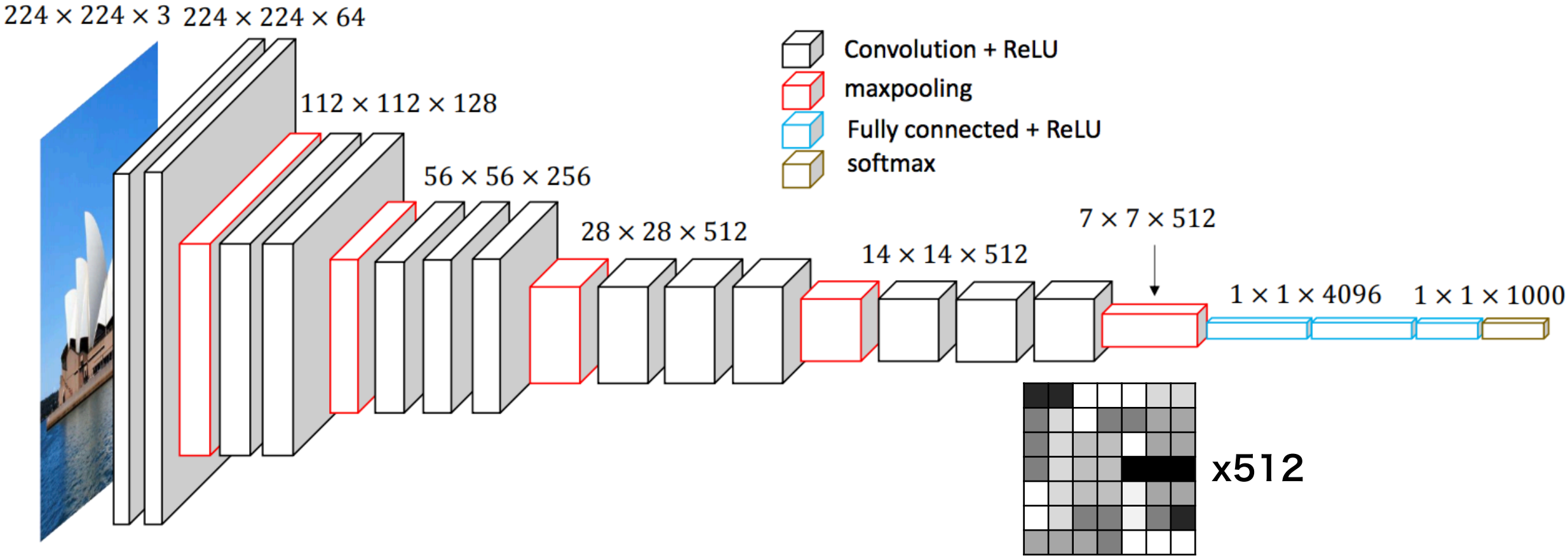


プーリング演算



畳み込みニューラルネットワーク

https://doi.org/10.1145/3038912.3052638

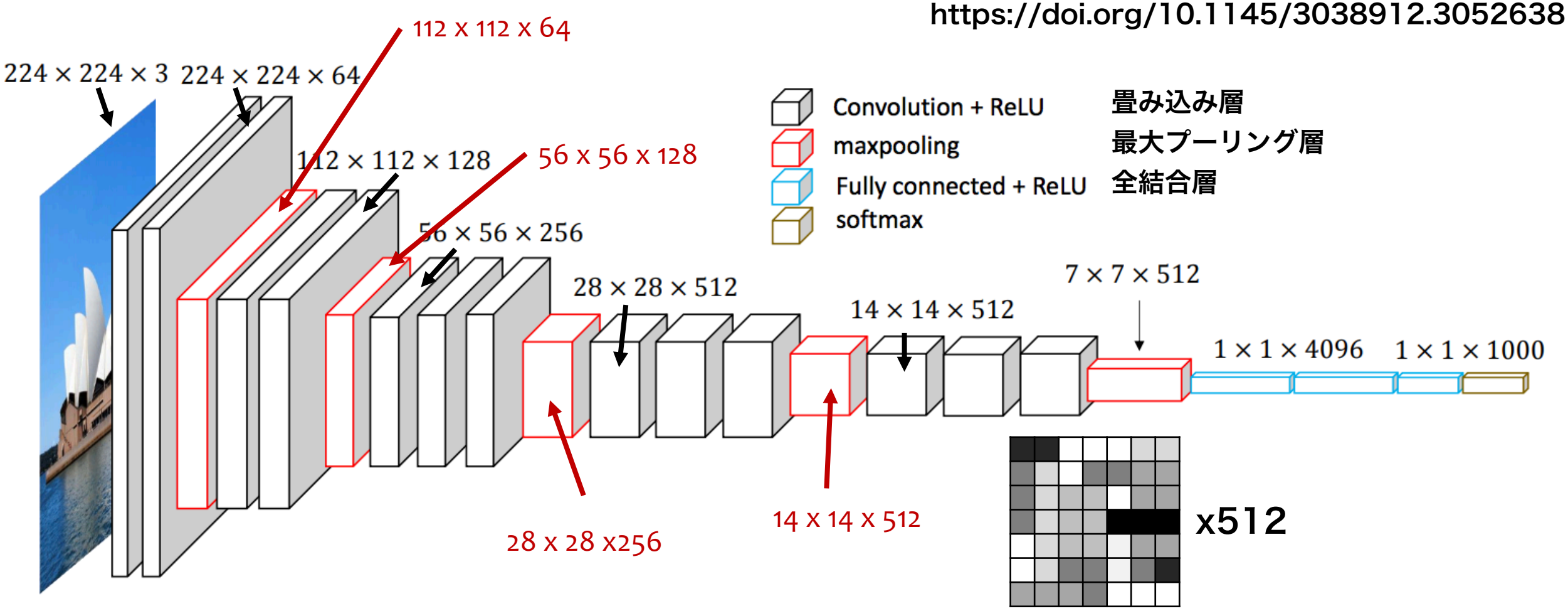


特徴抽出

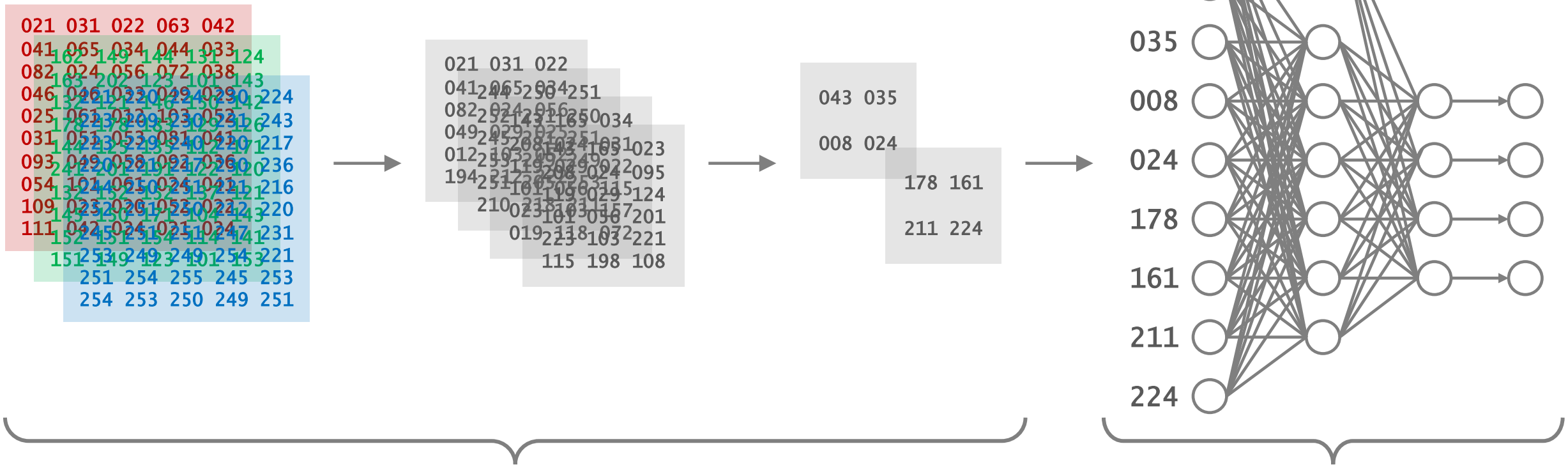
分類

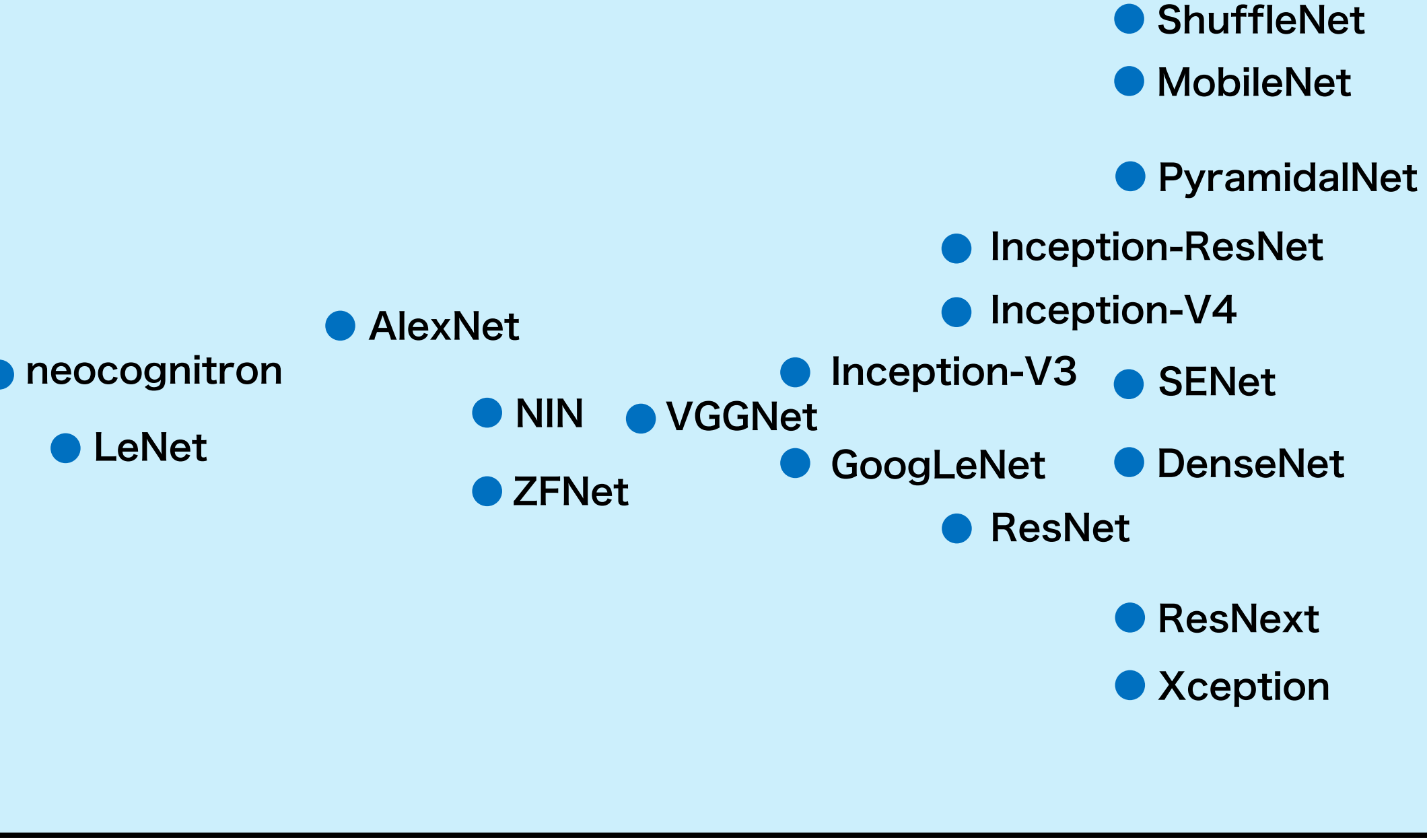
畳み込みニューラルネットワーク

https://doi.org/10.1145/3038912.3052638



畳み込みニューラルネットワーク




ShuffleNet
MobileNet
PyramidalNet
Inception-ResNet
Inception-V4
AlexNet
neocognitron
Inception-V3
SENet
NIN
VGGNet
LeNet
GoogLeNet
DenseNet
ZFNet
ResNet
ResNext
Xception
1998
2012
2014
2016
2018
2020

LeNet

https://doi.org/10.1109/5.726791

- 5 層からなる畳み込みニューラルネットワーク
- backpropagation neural network learning algorithm を拡張した gradient-based learning を適用
- 手書き数字の判別

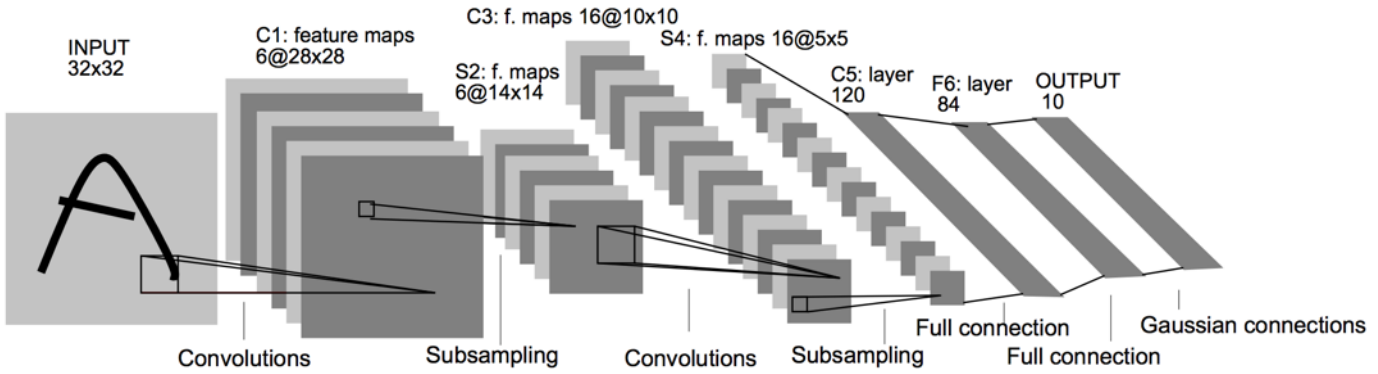


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

AlexNet

https://doi.org/10.1145/3065386

- 3 つの畳み込み層、2 つのプーリング層、および 3 つの全結合層からなる 8 層の深層学習アーキテクチュア

- 画像解析における様々な技術を導入
  - ReLU 関数
  - augmentation
  - dropout
  - local response normalization

- 224x224 の正方形画像を、上半分と下半分に分けて、それぞれを独立に 2 つの GPU で並列学習

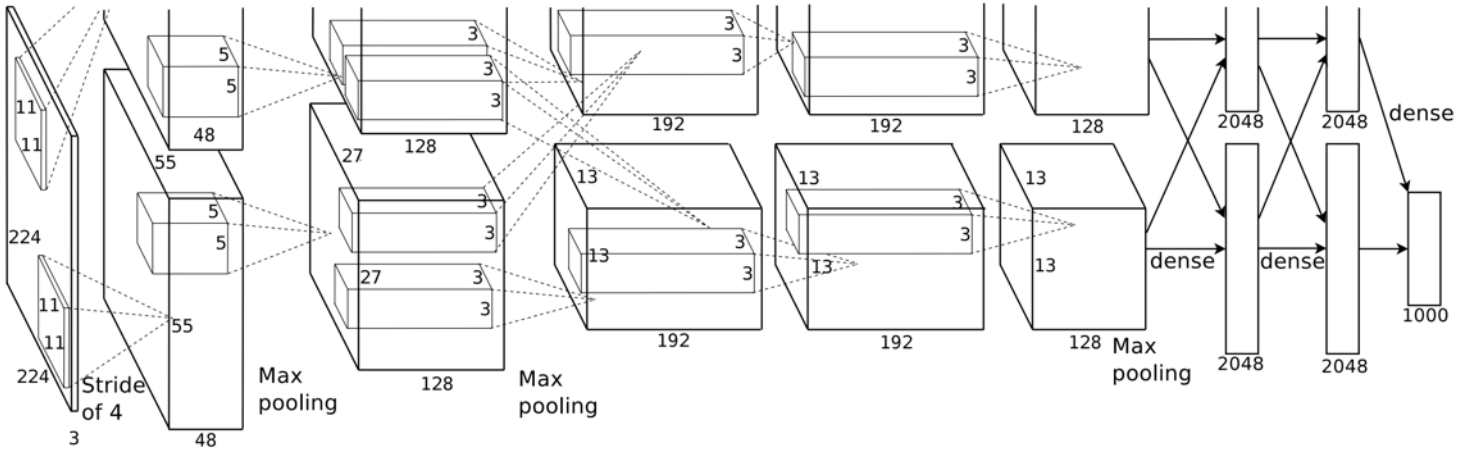


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

ZFNet

https://doi.org/10.1007/978-3-319-10590-1_53

- AlexNet を改良した CNN アーキテクチュア
  - AlexNet の 1 層目の畳み込みカーネルサイズは 11x11 を、7x7 に変更したことにより、画像上の細かい情報を取得できるようになった
  - AlexNet の 1 層目のスライディング距離を 4px から 2px に変更したことにより、エイリアシングの発生を抑えた

- 転移学習
  - Caltech 101 で学習済みのモデルを ImageNet で再学習させたら、ImageNet のみで学習させたモデルよりも性能が高かった

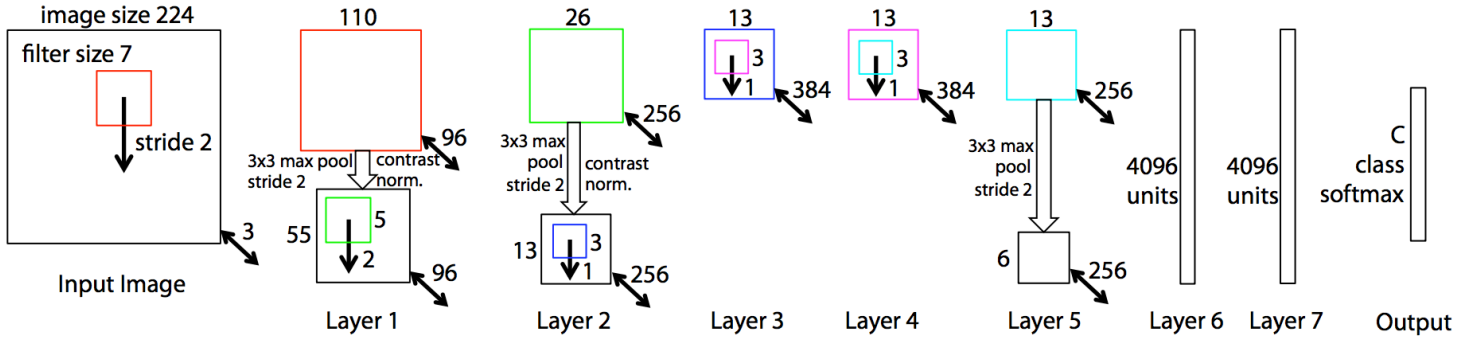


Fig. 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

VGGNet

https://arxiv.org/abs/1409.1556

- 従来のアーキテクチュアを大きく改造し、16 層または 19 層からなる深層アーキテクチュアを実現した
- 16 層および 19 層のアーキテクチュアを比較することで、層が深くなると性能が上がることを示した
- local response normalization は予測性能の向上に寄与しないことを示した
- 1x1 convolution を導入したが、予測性能の向上に寄与しないことを示した

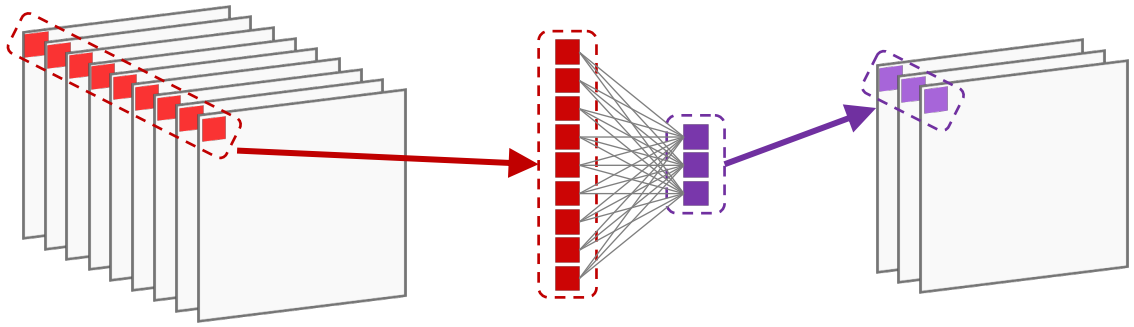
Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

1x1 Convolution

https://arxiv.org/abs/1409.1556

1x1 Convolution は入力画像のサイズを変えずに、チャンネルの数を変更する畳み込み操作である。この操作を行ってから畳み込み演算を行うことでパラメーター数を削減し、過学習を抑えることができる。



9 チャンネル

3 チャンネル

14x14x480 サイズの画像を直接 14x14x48 サイズに畳み込むのに必要なパラメーター数

(14×14×480)×(5×5)×480 = 112,896,000

14x14x480 サイズの画像を 1x1 Convolution によりチャンネル数を 16 に減らしてから、14x14x48 サイズに畳み込むのに必要なパラメーター数

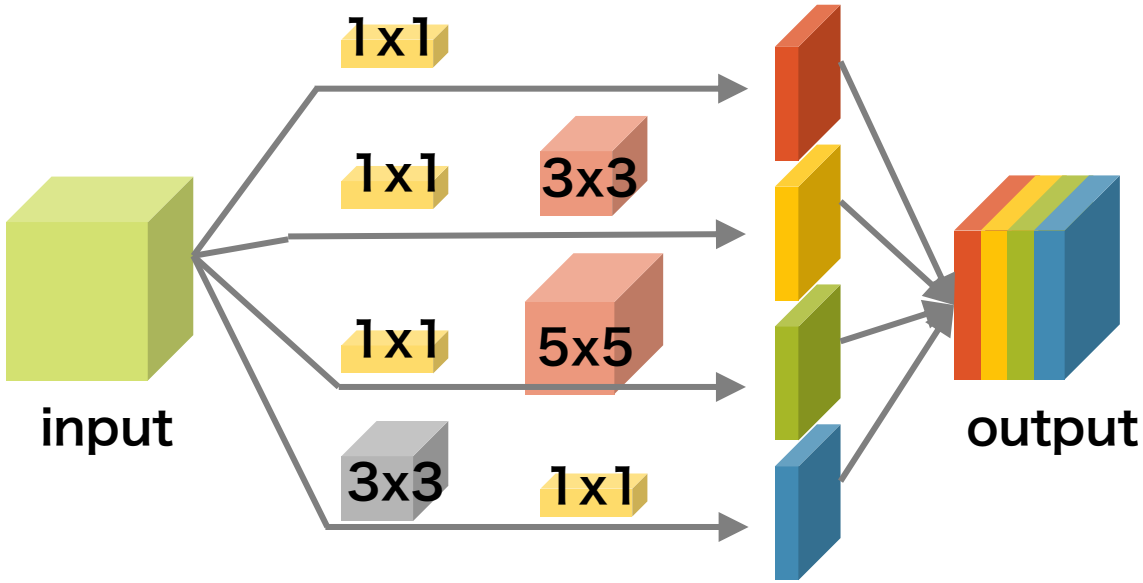
(14×14×480)×(1×1)×16 = 1,505,280

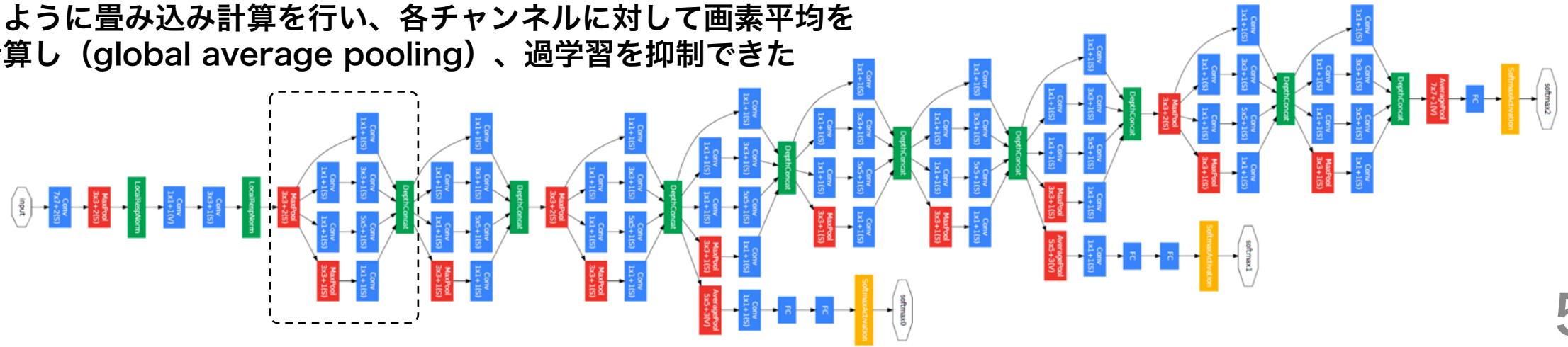
(14×14×16)×(5×5)×48 = 3,763,200

GoogleNet / Inception

https://arxiv.org/abs/1409.4842

- 1x1 Convolution を導入し、パラメーター数を大幅に減らすことができた
- Inception モジュールを導入し、様々なサイズの畳み込みカーネル行列による演算で、画像の細部から抽象的な情報を取得できるようになり、過学習を抑えながら層を深くする（22層）ことができた
- アーキテクチャ途中に分類を行う分岐を設け、auxilary loss を計算することで、勾配消失を抑制できた
- 最後の畳み込み層において、チャンネル数をクラス数と同じになるように畳み込み計算を行い、各チャンネルに対して画素平均を計算し（global average pooling）、過学習を抑制できた

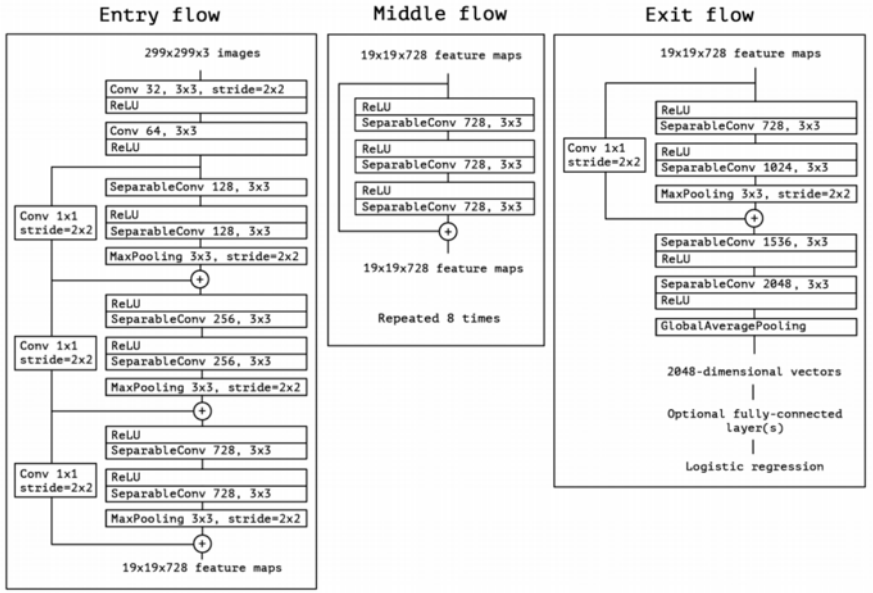


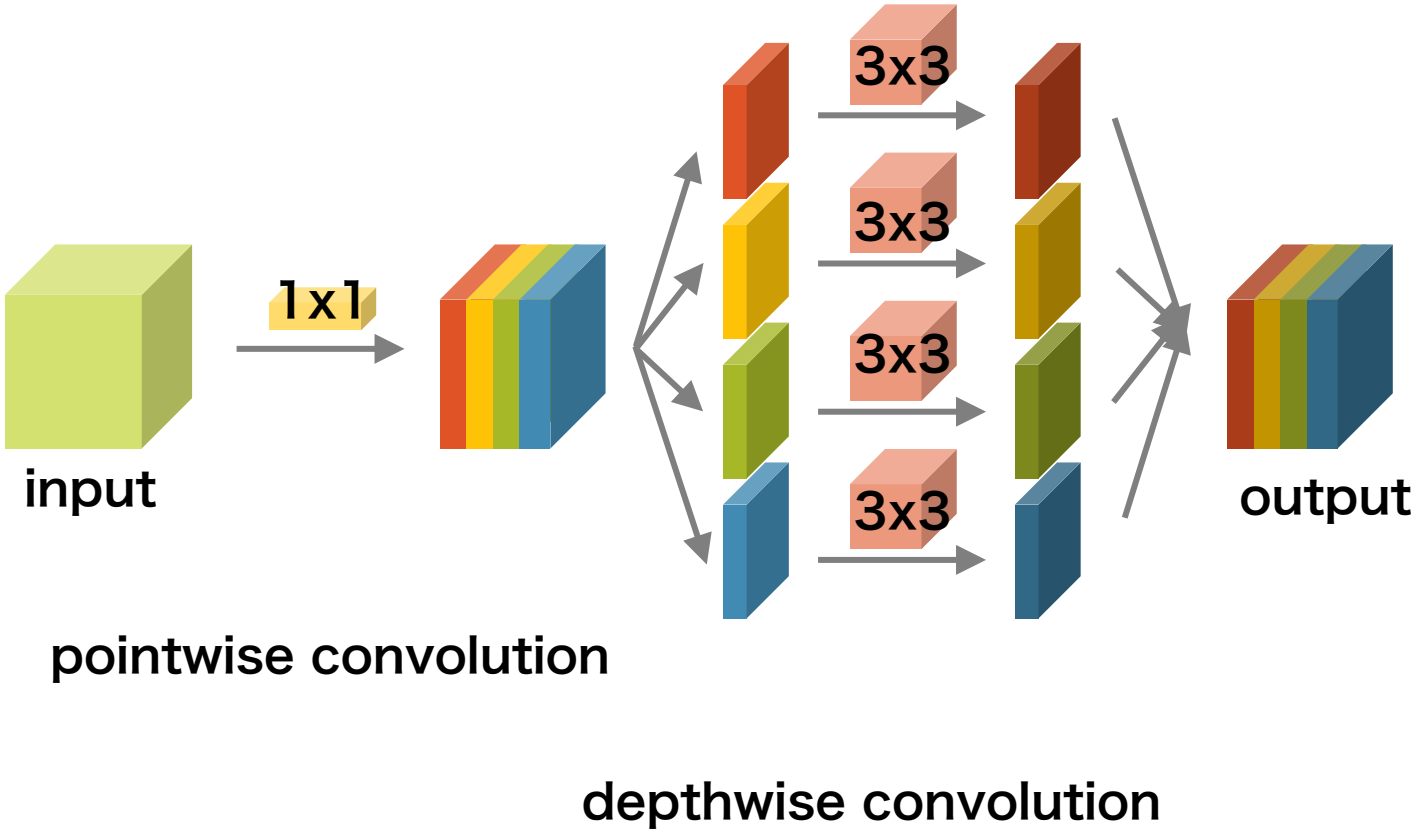


Xception

https://arxiv.org/abs/1610.02357

- Inception モジュール内部の計算順序を改変したアーキテクチュア
  - 1x1 Convolution のあとに 3x3 Convolution 演算を行う
  - パラメータ数および計算量を数分の１に削減できた





MobileNet

https://arxiv.org/abs/1704.04861

- depthwise convolution を導入し、計算量およびパラメーター数を削減し、アーキテクチュアを軽量化した
- SSD に組み込み、スマホなどで物体検出
- MobileNetV2 もリリースされた

Figure 1. MobileNet models can be applied to various recognition tasks for efficient on device intelligence.

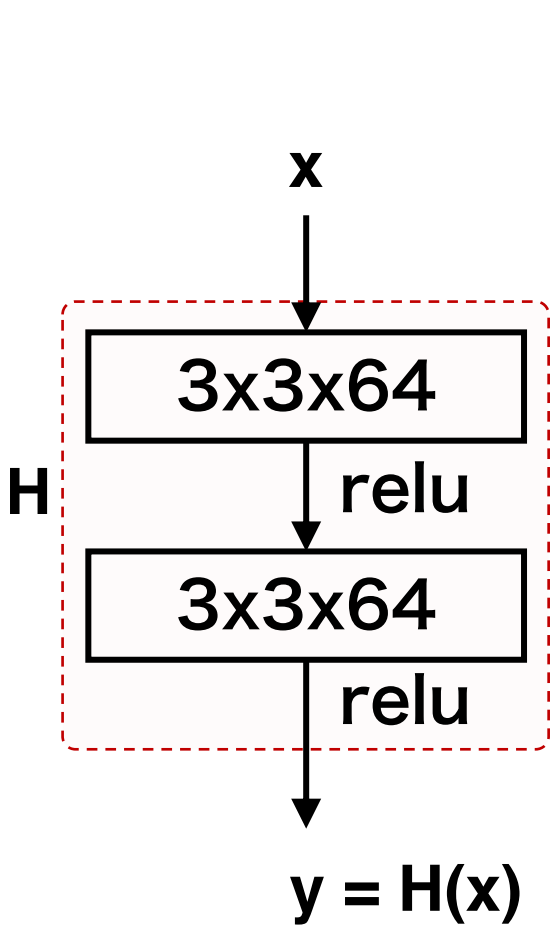
Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

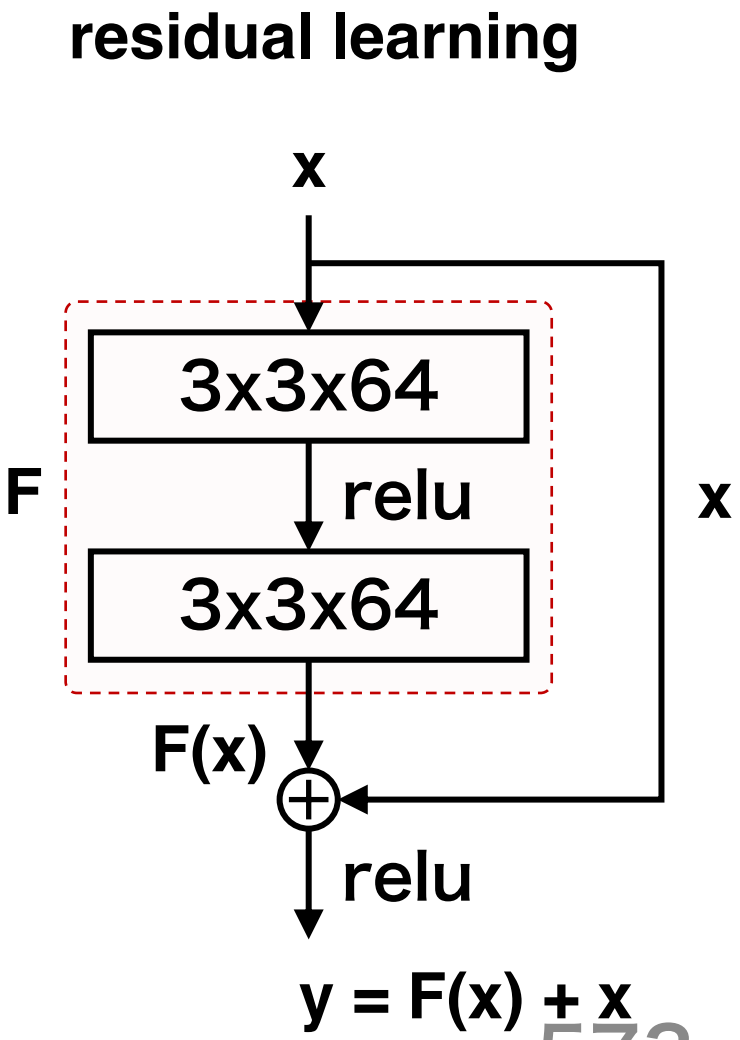
ResNet

https://arxiv.org/abs/1512.03385

- 1x1 Convolution を導入し、パラメーター数を大幅に減らすことができた
- residue network を導入することにより、勾配消失問題を解決し、層を深くする（152層）ことができた
- 一般的なニューラルネットワークは入力 x を受け取り、x の写像である H(x) を出力する。これに対して residue network では、F(x) = H(x) - x と定義し、y = H(x) = F(x) + x を出力している。両者どちらも H(x) を出力するが、前者は学習を通じて写像 H を決定し、後者は写像 F を決定している



residual learning



DenseNet

https://arxiv.org/abs/1608.06993

- residue block を改良した DenseBlock を導入し、
  パラメーター数を削減し、勾配消失問題にも対応可能

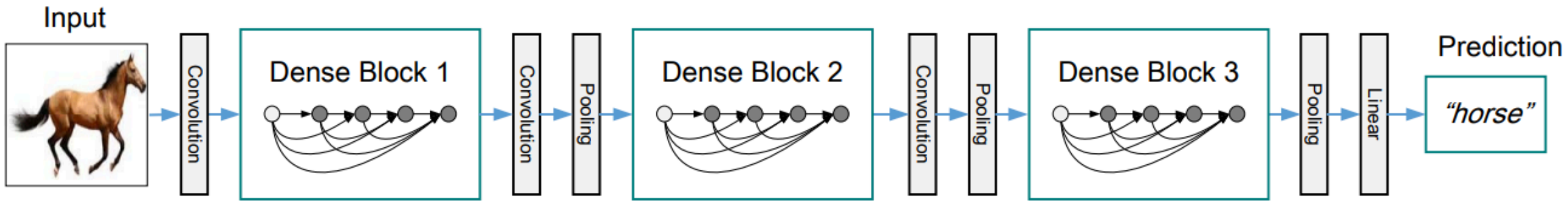
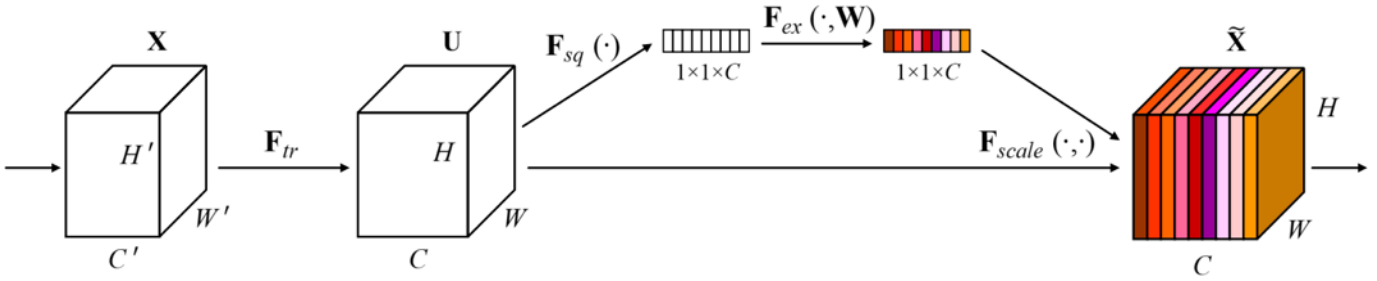


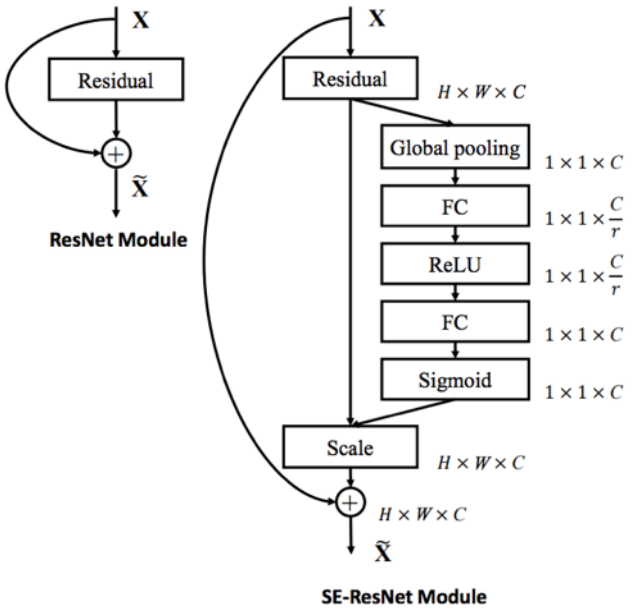
Figure 2: A deep DenseNet with three dense blocks. The layers between two adjacent blocks are referred to as transition layers and change feature-map sizes via convolution and pooling.

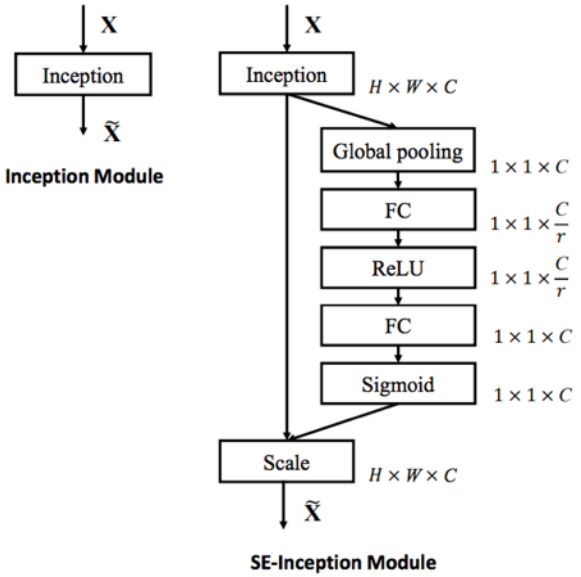
SENet

https://arxiv.org/abs/1709.01507

- Squeeze-and-Excitation block を導入し、チャンネル間の関連性を学習
- 既存の ResNet または Inception モジュールに組み込むことができる



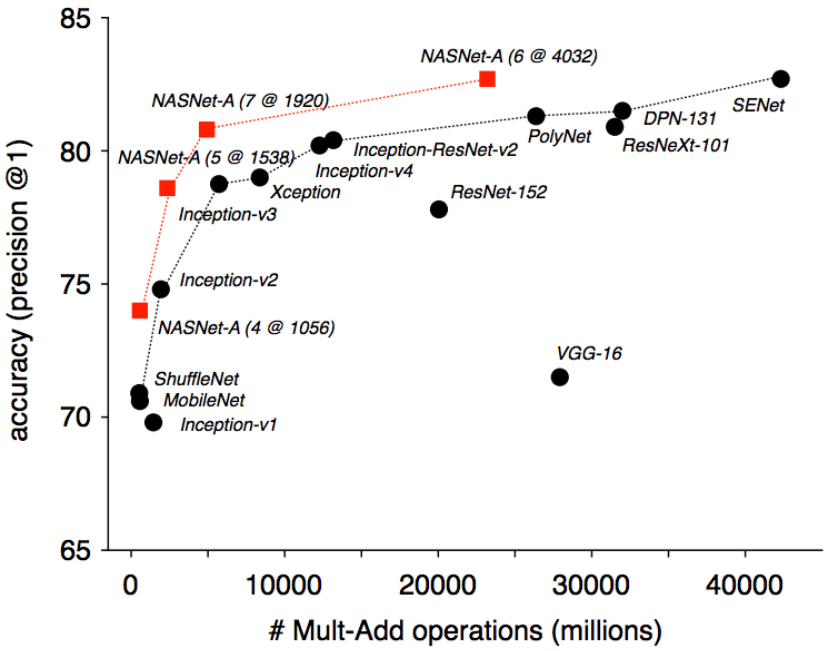


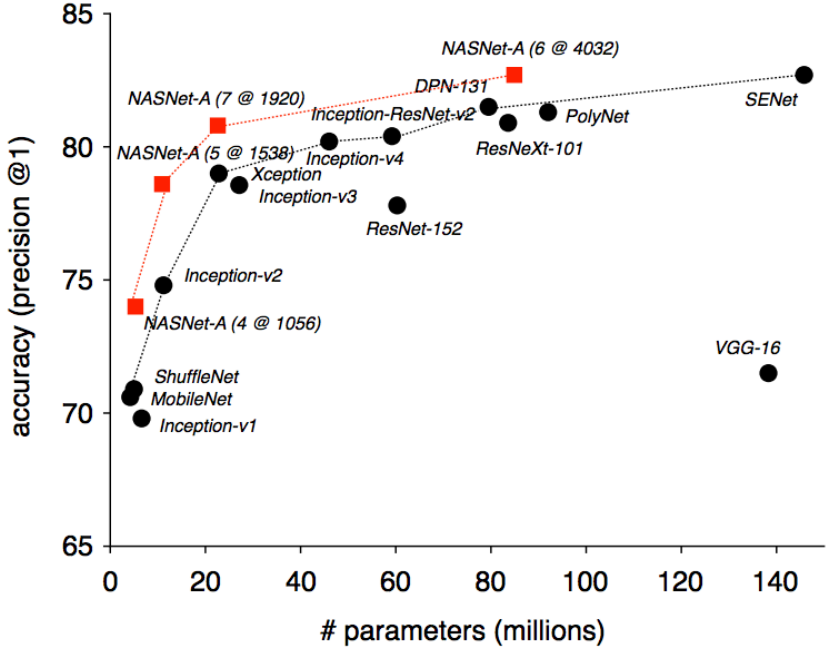


NASNet

https://arxiv.org/abs/1707.07012

- AutoML（機械学習の専門家でなくても高性能な画像分類モデルを生成できる）
- Neural Architecture Search アルゴリズムにより、ニューラルネットワークの最適なアーキテクチュアが自動的に探索され作られる
- ImageNet の画像を AutoML / NAS により学習させることでできたアーキテクチュアが NASNet

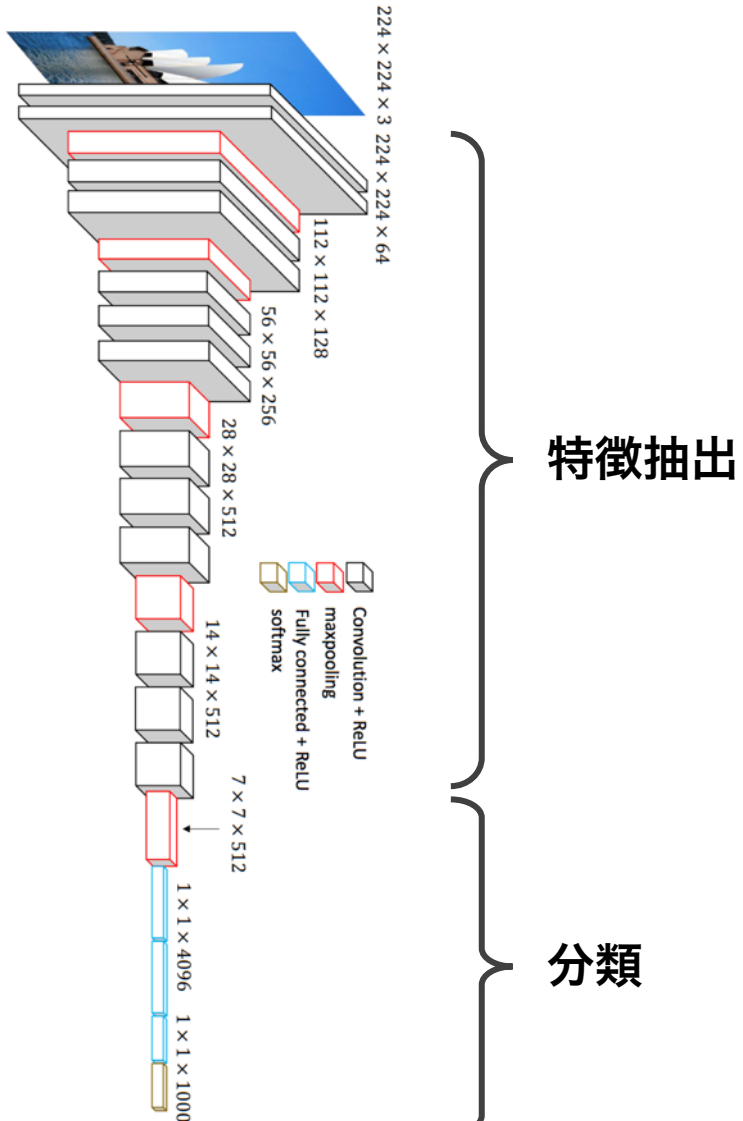




画像解析概略

- 物体認識
- 畳み込みニューラルネットワーク
- 転移学習
- 物体検出

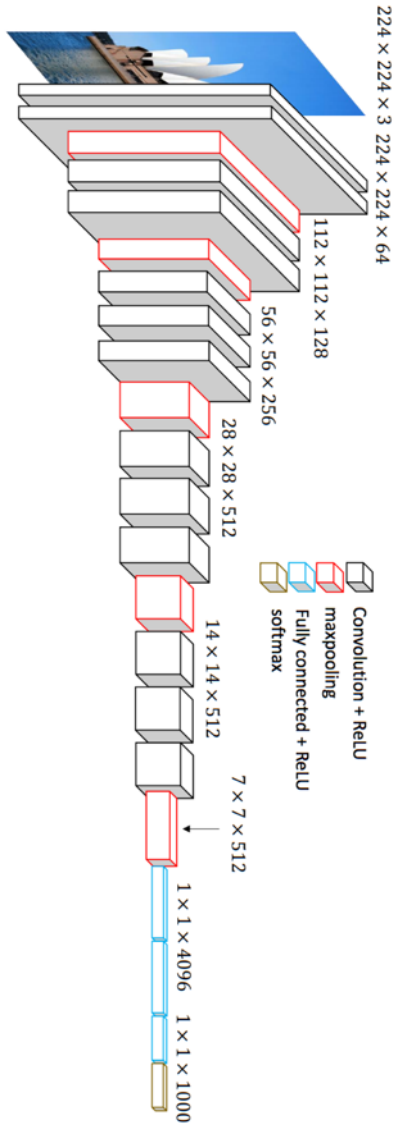
CNN


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
Convolution + ReLU
maxpooling
Fully connected + ReLU
softmax

特徴抽出

分類

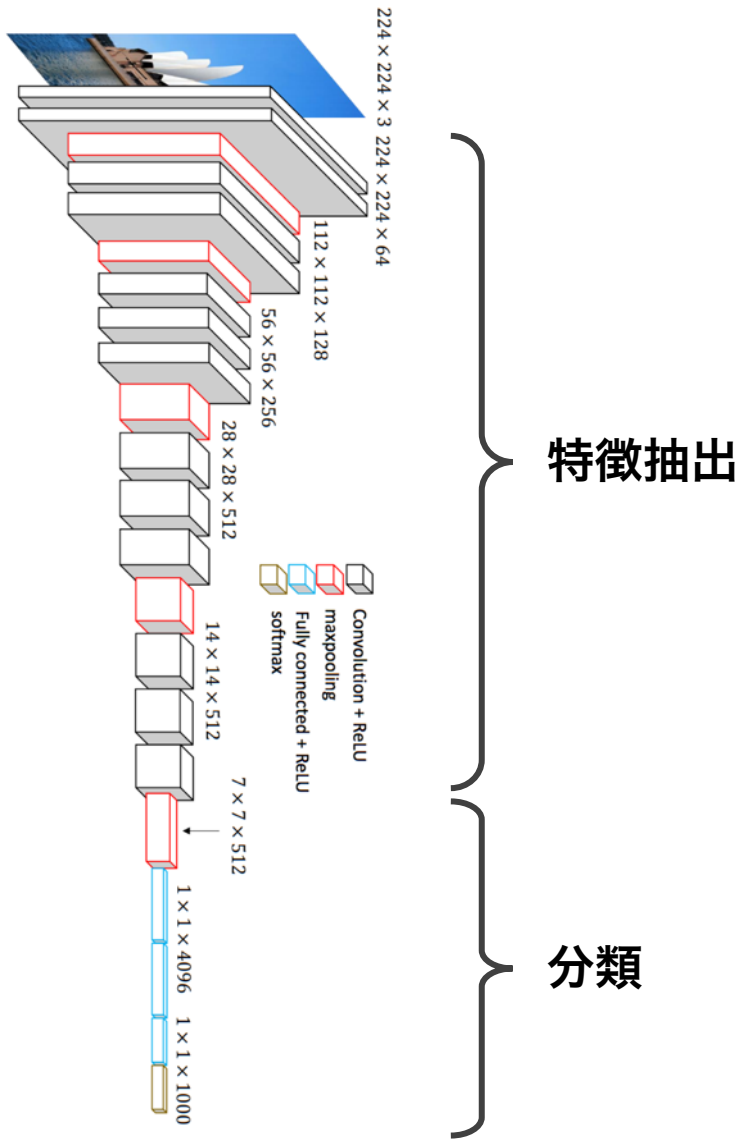
転移学習



特徴抽出　パラメーター固定

分類　　パラメーター再学習

転移学習



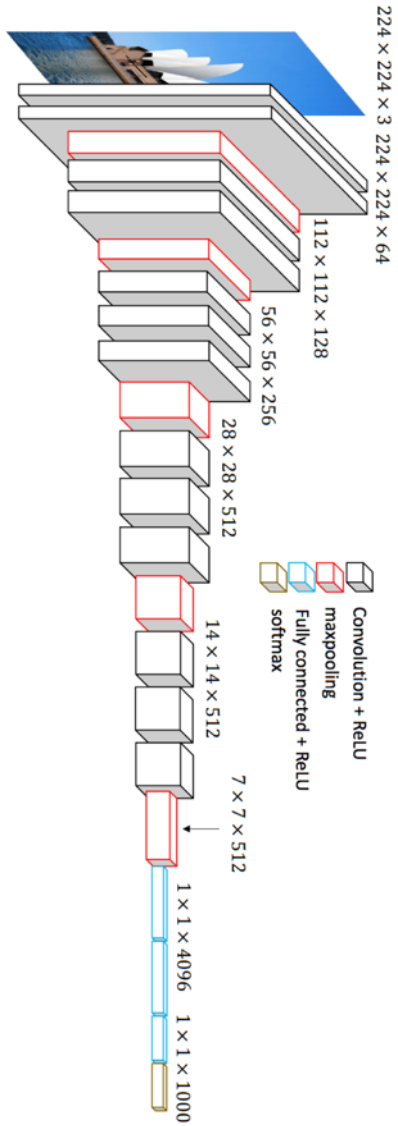
Layer 1

Layer 8

特徴抽出

分類

転移学習

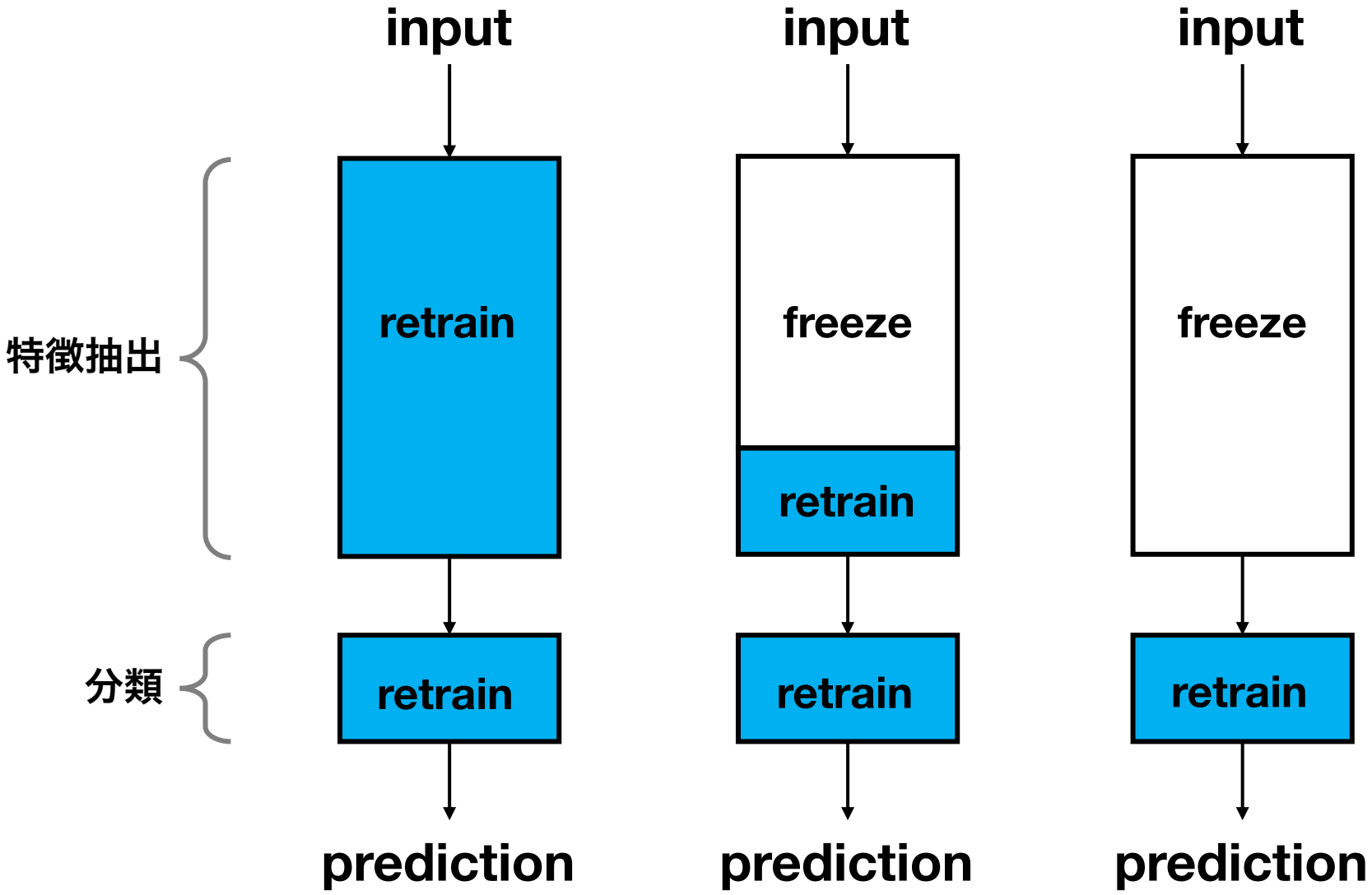


特徴抽出　パラメーター固定

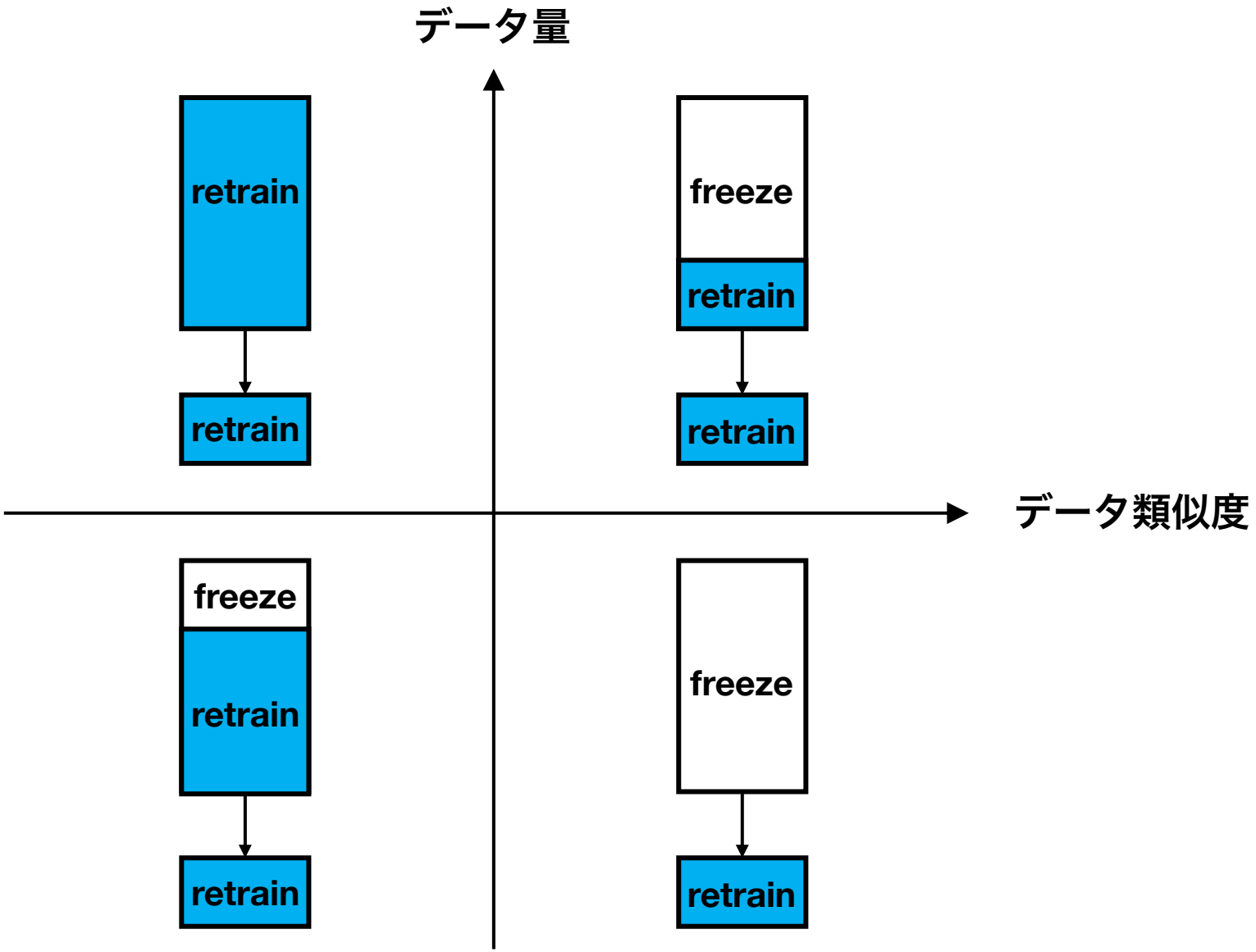
特徴抽出　パラメーター再学習

分類　　パラメーター再学習

転移学習


input
input
input
特徴抽出
分類
retrain
freeze
freeze
retrain
retrain
retrain
retrain
prediction
prediction
prediction

転移学習

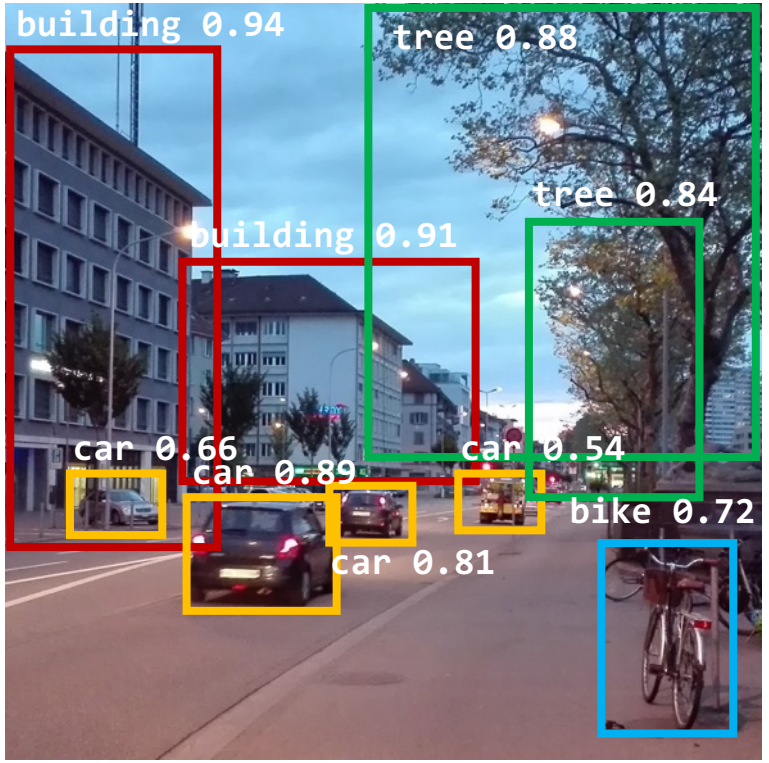

データ量
retrain
retrain
freeze
retrain
retrain
データ類似度
freeze
retrain
retrain
freeze
retrain

画像解析概略

- 物体認識
- 畳み込みニューラルネットワーク
- 転移学習
- 物体検出

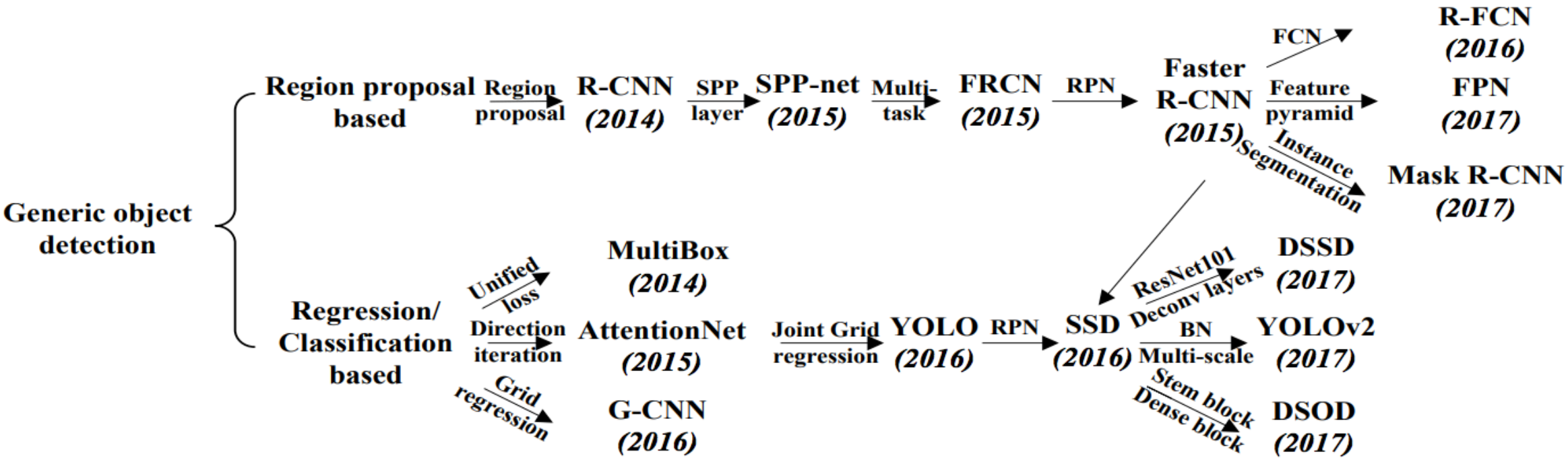
物体認識

物体検出



セグメンテーション

物体検出


Generic object detection
Region proposal based
Region proposal
R-CNN (2014)
SPP layer
SPP-net (2015)
Multi-task
FRCN (2015)
RPN
Faster R-CNN (2015)
FCN
R-FCN (2016)
Feature pyramid
FPN (2017)
Instance Segmentation
Mask R-CNN (2017)
Regression/ Classification based
Unified loss
MultiBox (2014)
Direction iteration
AttentionNet (2015)
Grid regression
G-CNN (2016)
Joint Grid regression
YOLO (2016)
RPN
SSD (2016)
ResNet101 Deconv layers
DSSD (2017)
BN Multi-scale
YOLOv2 (2017)
Stem block Dense block
DSOD (2017)