ディープラーニング入門

【画像認識（物体認識とセグメンテーション）】

Ver.1

2019年３月3日

次世代人材開発研究所

ディープラーニング市場の拡大

- 2016年の$655Mから2025年には、$35Bに拡大と予測

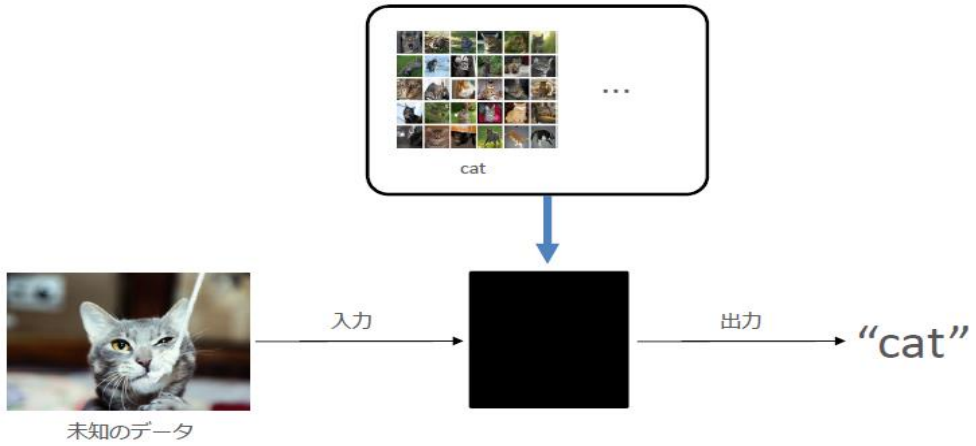
ディープラーニングとは？

- ディープラーニングは機械学習の手法の一つ
- モデルは画像、テキ スト、音声から直接分類方法を学習
  - いわゆる学習によるパターン認識
- ディープラーニングはニューラルネットワークの構造が使われる
  - ディープはネットワーク内の層数を示し、層が多いほどネットワークは深くなる。
  - ニューラルネットワークは2〜3層程度だが、ディープニューラルネットワークは数百の層になる場合もある

機械学習とディープラーニング

機械学習とは？

- 機械学習とは「明示的にプログラミングすることなく，コンピュータに行動させるようにする科学」のこと。
- 一般には過去データやサンプルから学び認識や予測、分類を実現する技術。



学習データに含まれていない未知のデータを、
正しく認識させることが目標

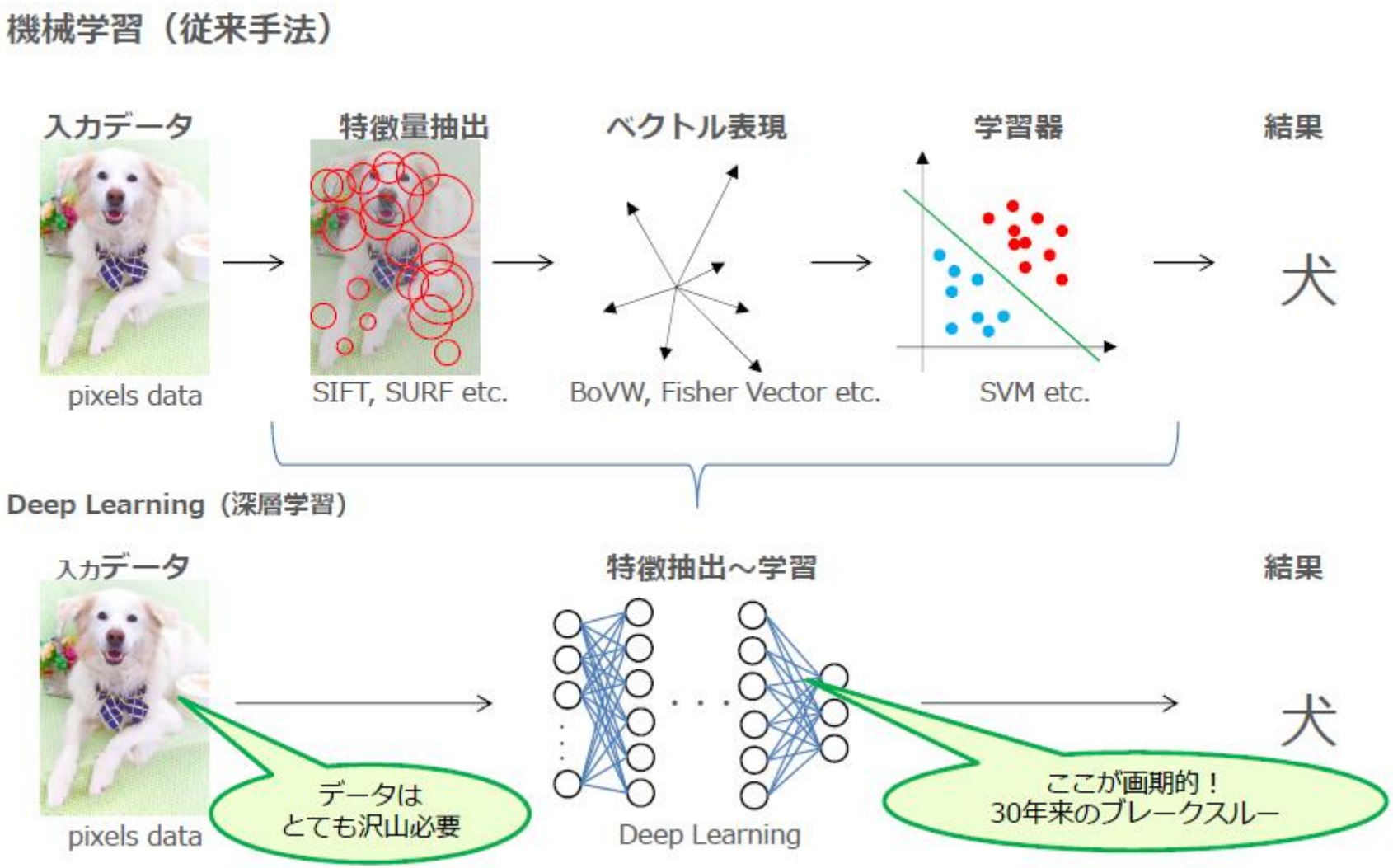
機械学習の分類

- 教師あり学習
  - 既知の情報からパターンを学習
  - 予測対象が数値であれば「回帰」、カテゴリであれば「分類」
- 教師なし学習
  - データのパターンをデータのみから推測
  - クラスタリング
- 他にも、半教師あり学習、強化学習、等々

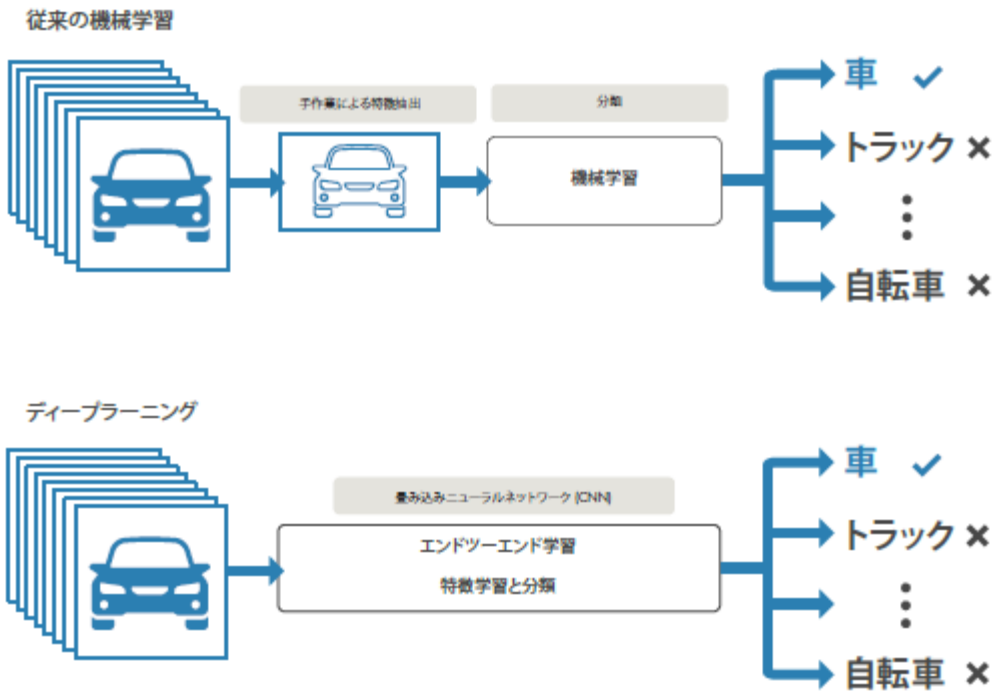
機械学習の種類

種類	概要
ロジスティック回帰	1950年代から存在する確率モデルで２値分類問題を解く手法。 統計的手法とも言えます。
決定木・回帰木	分類や回帰問題を解く予測モデルを決定木の形で学習する。 条件分岐で予測値を求めるため、人の目で見てモデルが分かりやすい。
ブースティング	一連の弱い仮説器をまとめることで強い仮説を生成する集団学習アルゴリズム。教師あり学習に対応する
Random Forest	決定木を弱仮説器とする集団学習アルゴリズムです。
SVM (Support Vector Machine)	1990年代一世を風靡した手法。Deep Learningが登場する以前は最も認識性能が優れ手法の１つだった。現在もベンチマーク的な意味合いを持っている。
Deep Learning	1960年代から存在するニューラルネットワークを用いた手法。注目・幻滅を繰り返しきましたが、近年計算機パワーの増大と過学習を抑える工夫があいまって画像認識、音声認識、動画認識など様々な分野で高い精度を叩き出しています。現在最も認識性能が優れた手法と言われています。

ディープラーニングと機械学習との違い１



ディープラーニングと機械学習との違い２

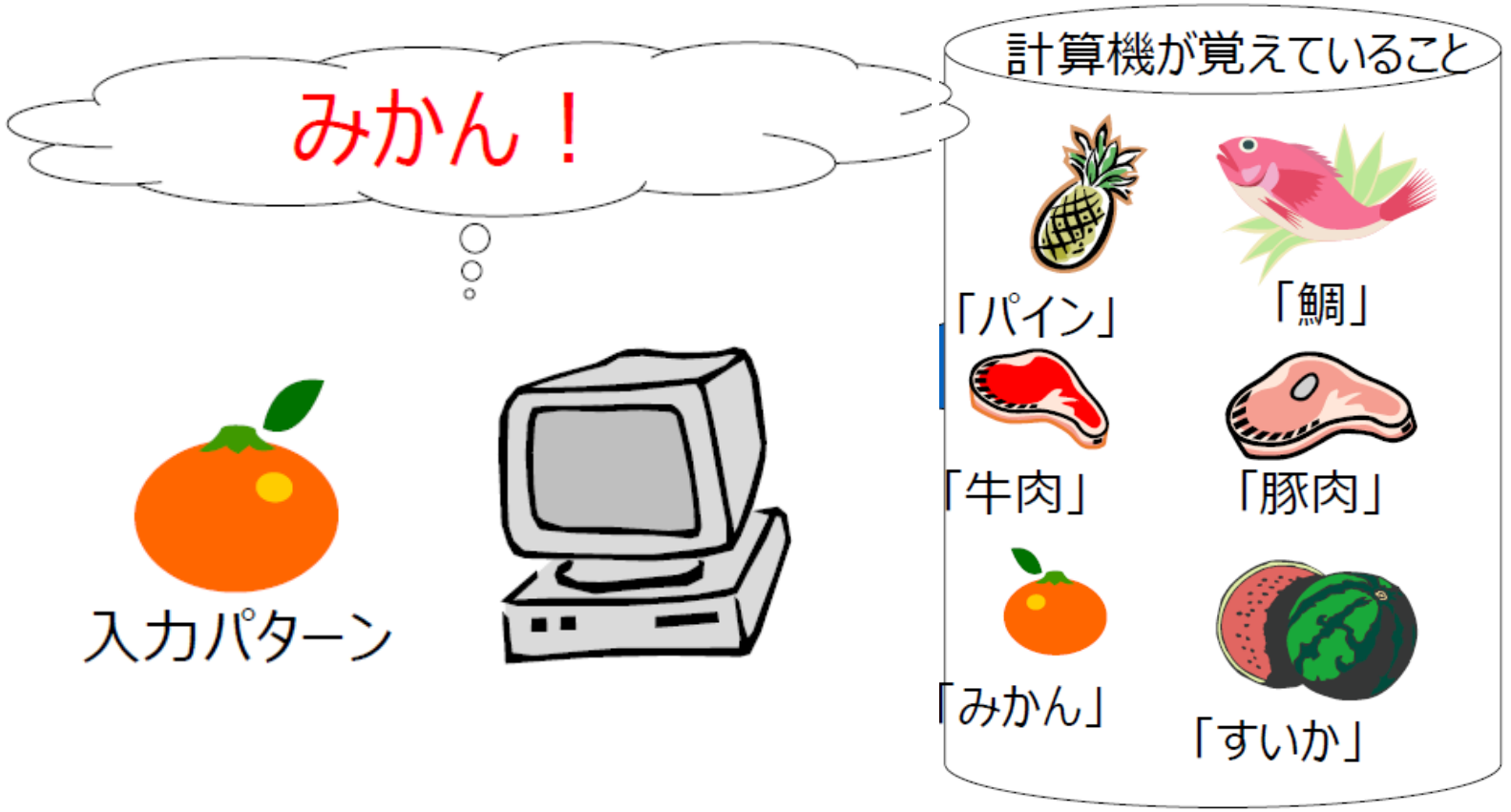


機械学習	ディープラーニング
小さなデータセットでも良い結果が得られる	非常に大規模なデータセットが必要
モデルの学習が高速	計算量が多い
最良の結果を得るには、さまざまな特徴量と分類器を試す必要がある	特徴量と分類器を自動的に学び取る
精度に上限がある	精度に上限がない・・・？

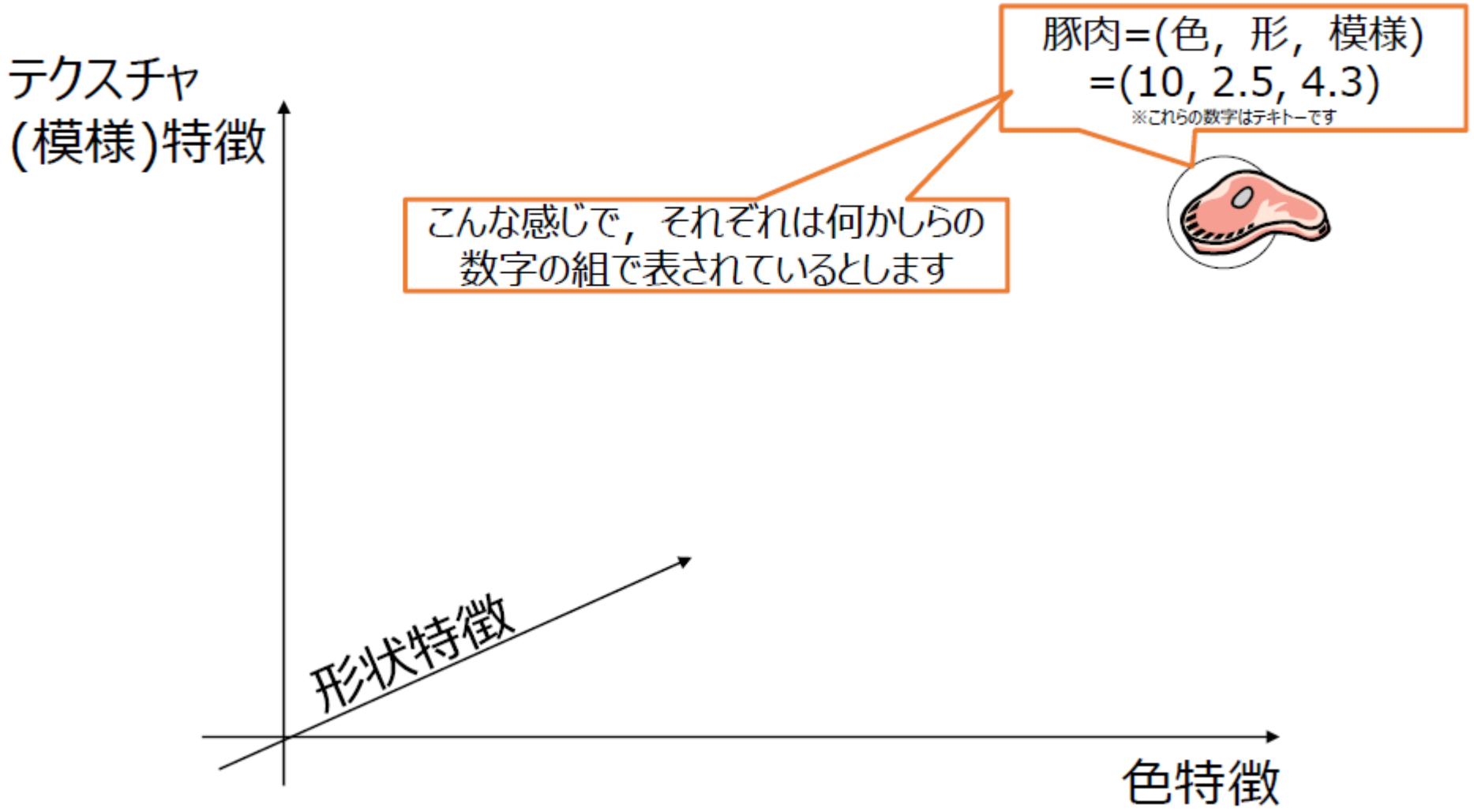
ディープニューラルネットワークの実現技術について・・・

最初にパターン認識について

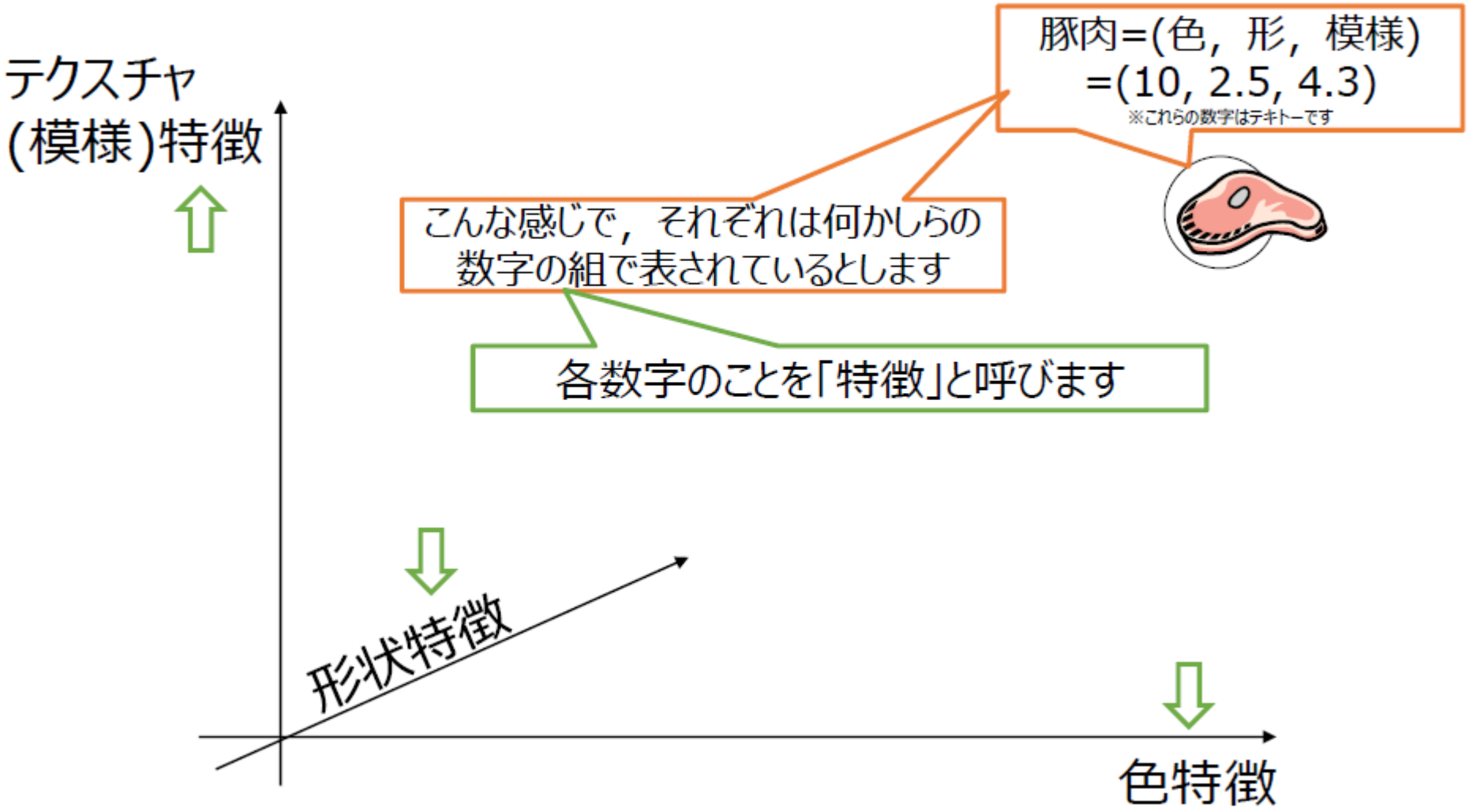
単に「自分が覚えているものの中で入力に最も似てるもの」を答えるだけ!



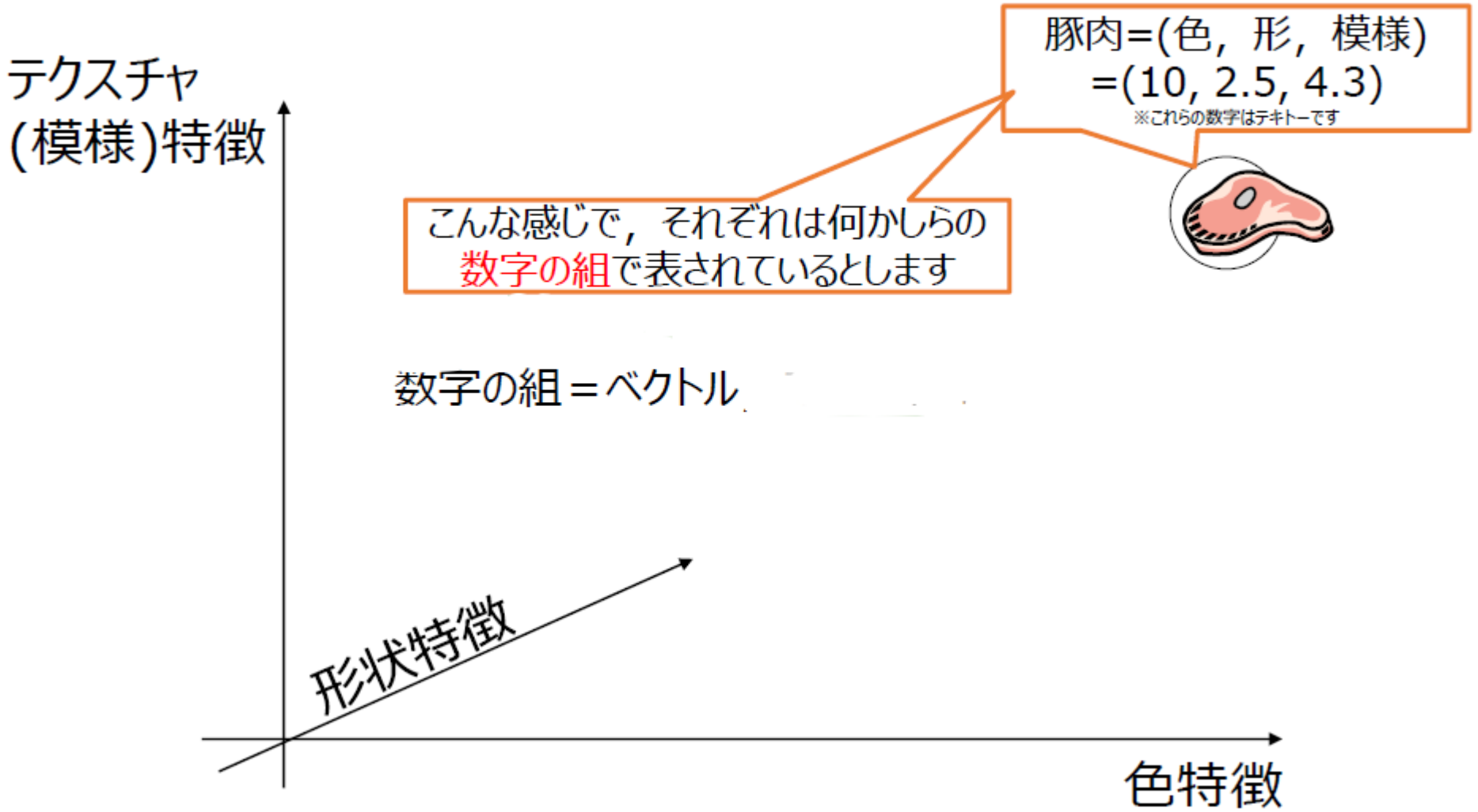
似ているものを座標空間で表すと・・・1



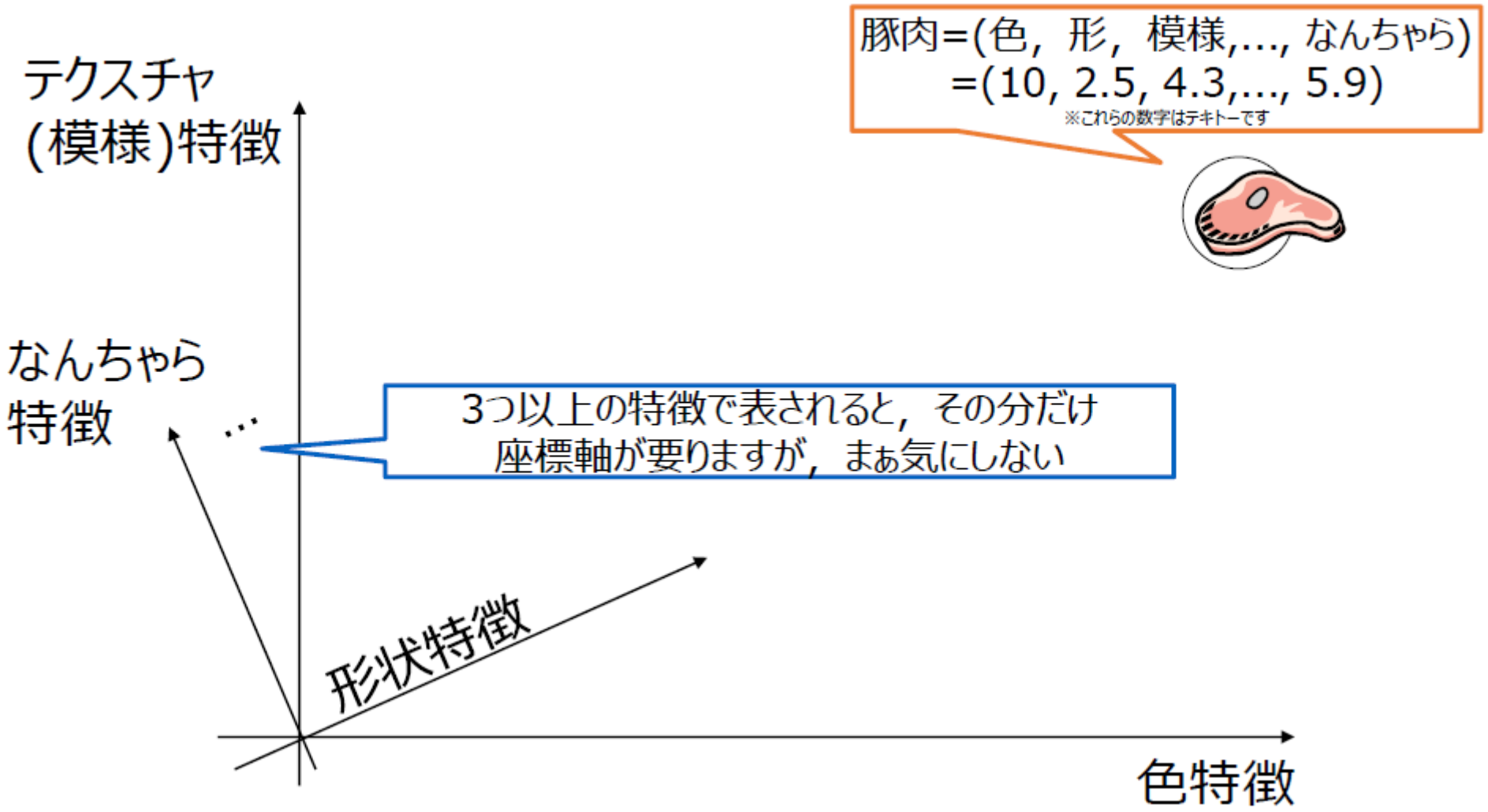
似ているものを座標空間で表すと・・・２



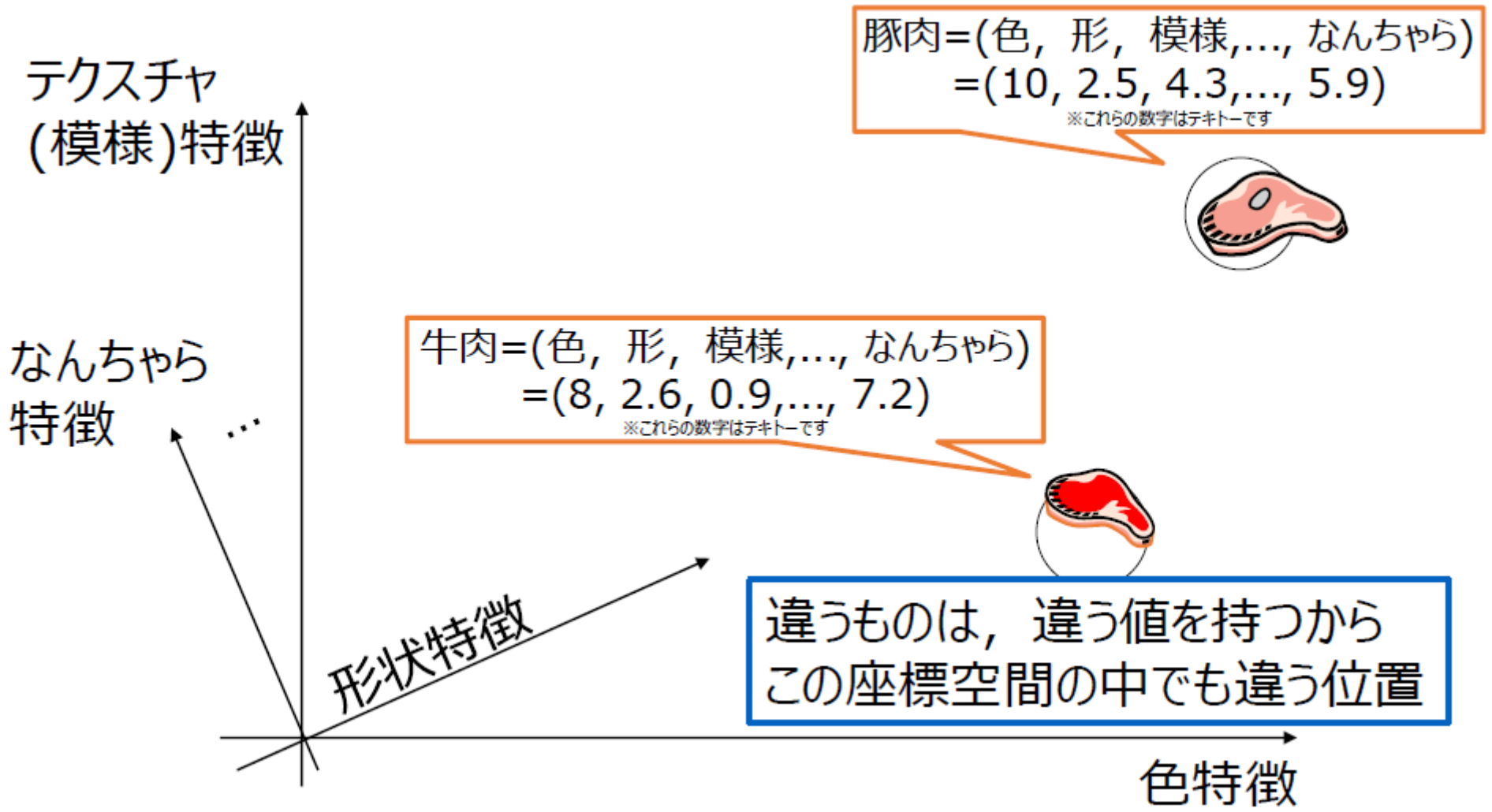
似ているものを座標空間で表すと・・・３



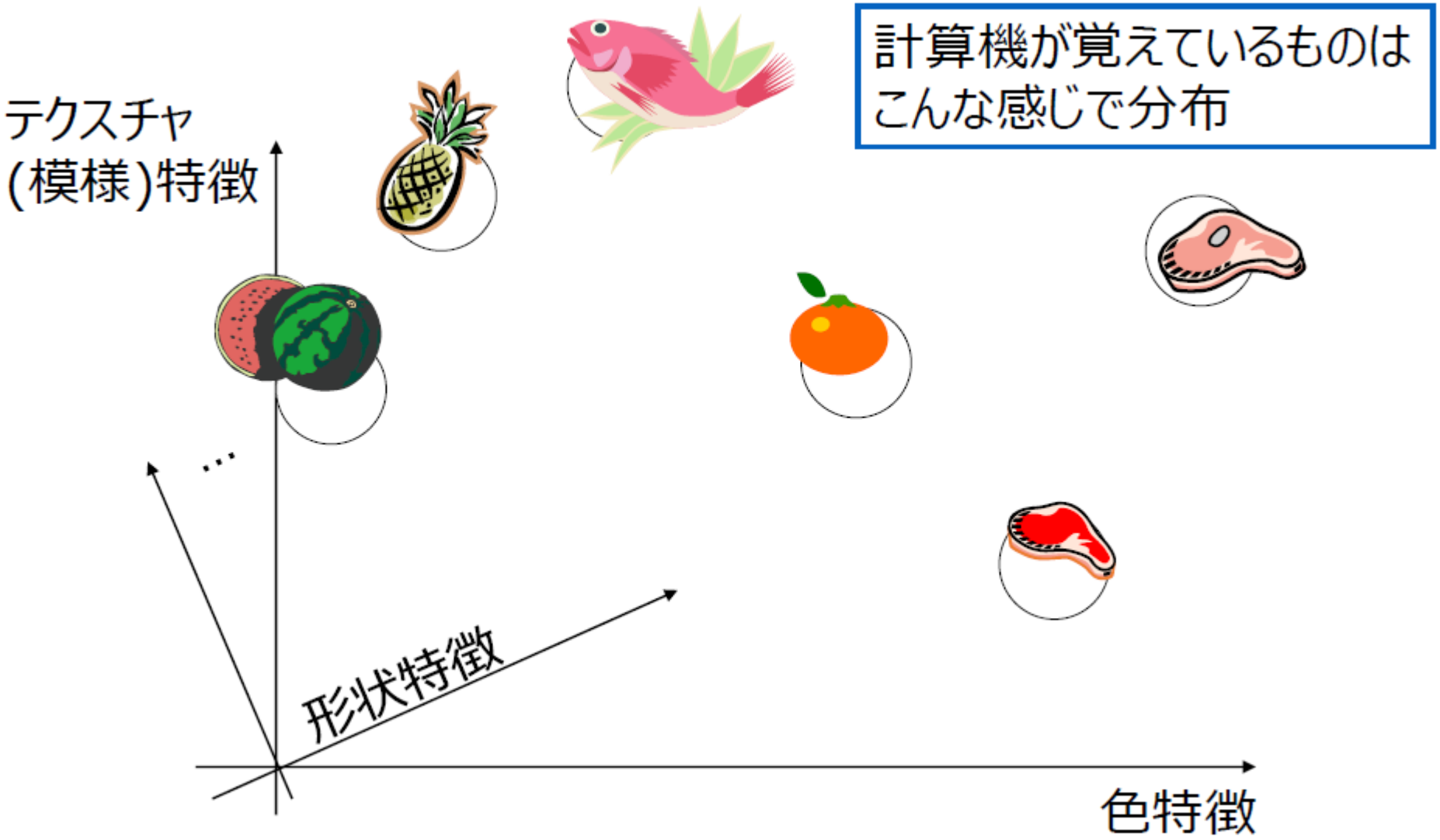
似ているものを座標空間で表すと・・・４



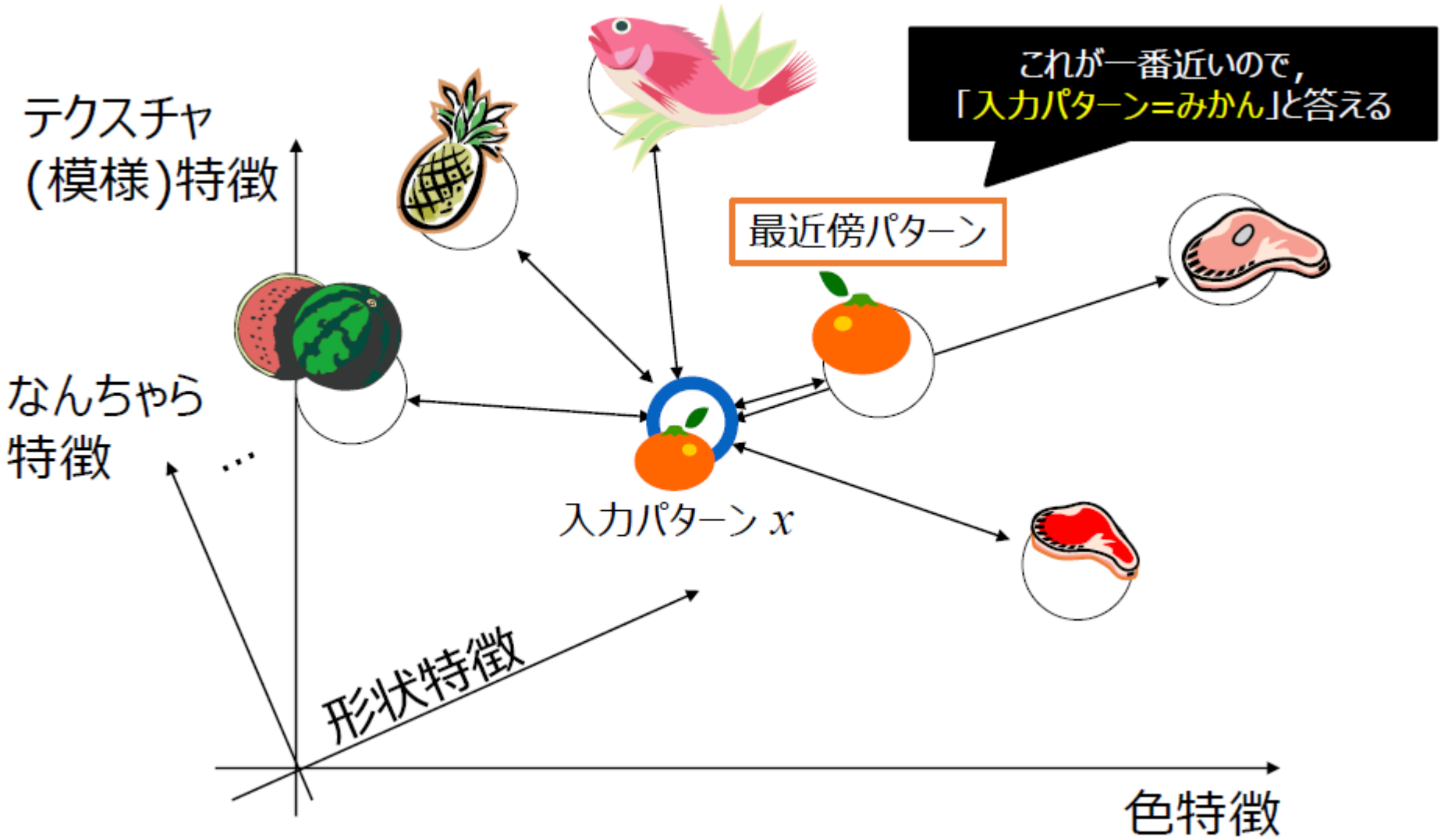
似ているものを座標空間で表すと・・・５



似ているものを座標空間で表すと・・・６



似ているものを座標空間で表すと・・・7



特徴の類似度が重要

パターン自身の性質

- オレンジの画素数→多い
- 青の画素数→小
- 円形度→高い
- 線対称度→高い
- 模様→細かい

….

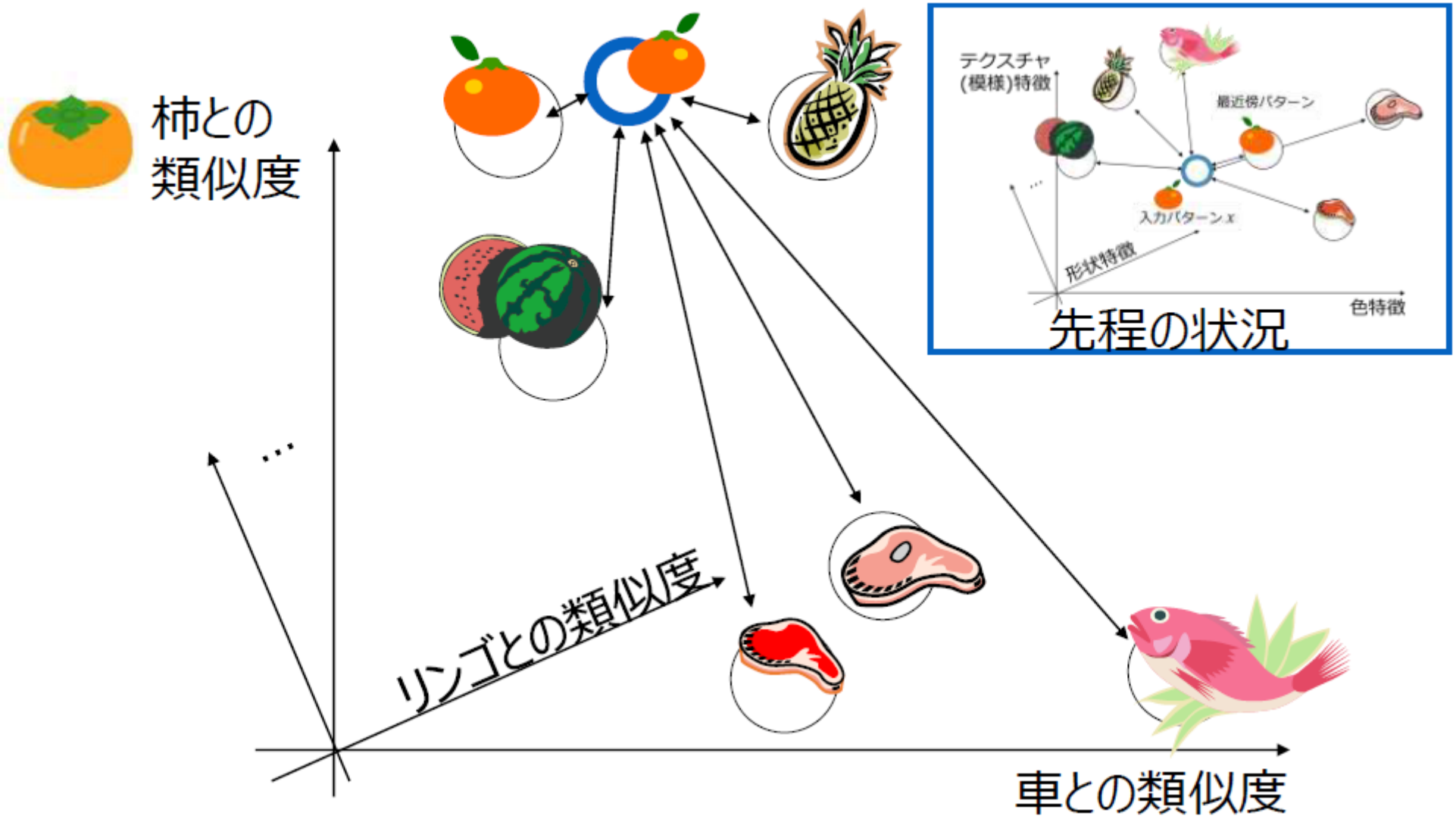
他のパターンとの関係

- 「車」との類似度→低
- 「リンゴ」との類似度→高
- 「猿」との類似度→低
- 「柿」との類似度→高

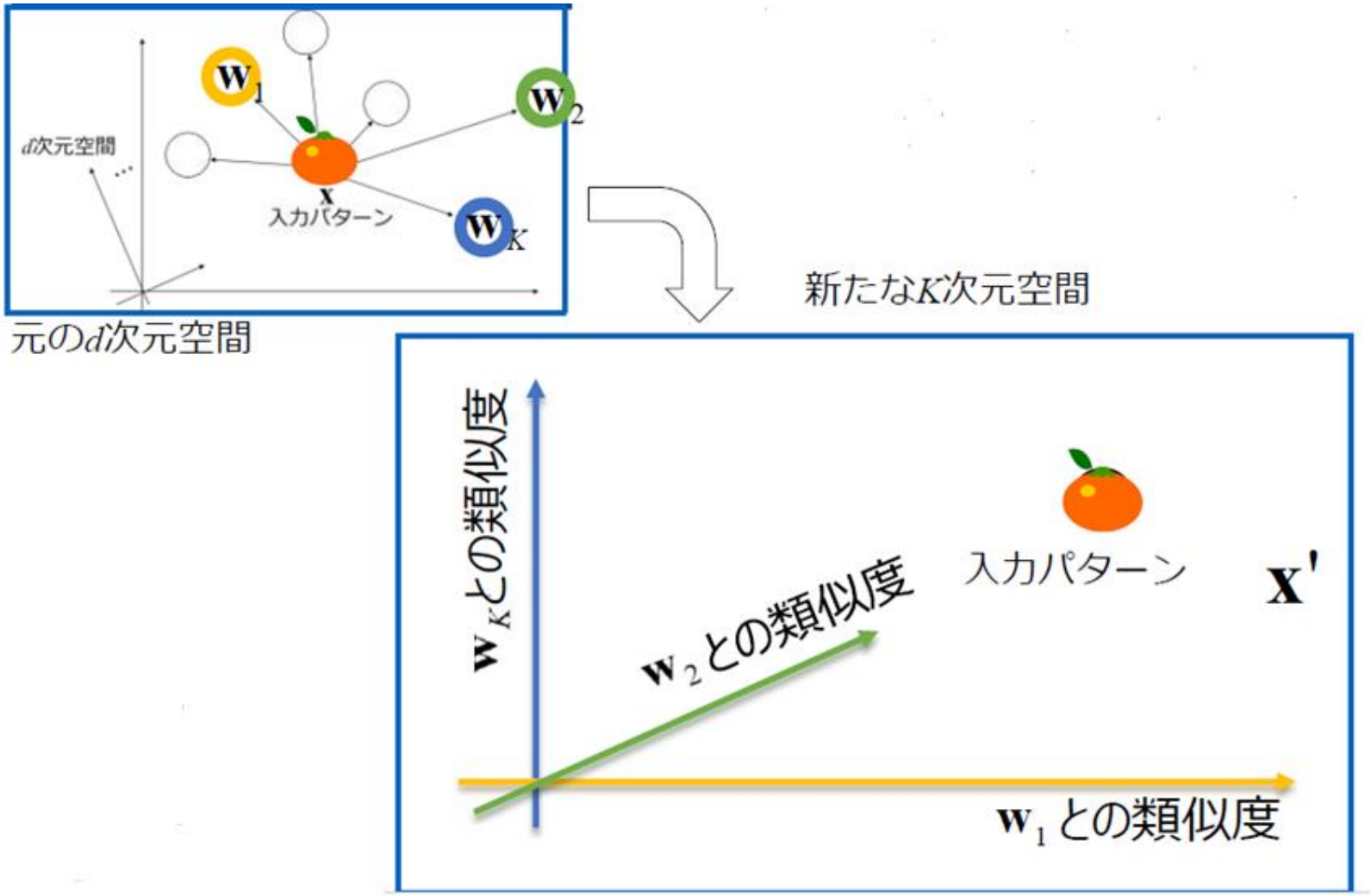
…

他者との類似度も
特徴になる！

他のパターンとの関係（類似度）も識別可能

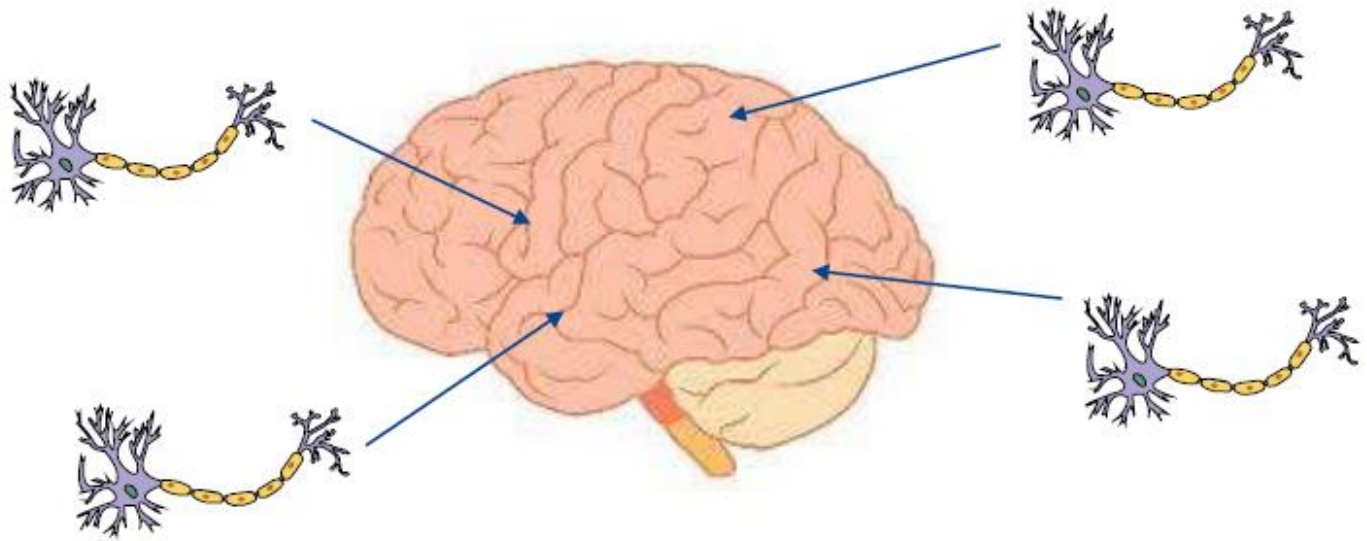


類似への「空間変換」


d次元空間
w_1
w_2
w_K
x
入力パターン
元のd次元空間
新たなK次元空間
w_Kとの類似度
w_2との類似度
入力パターン
x'
w_1との類似度

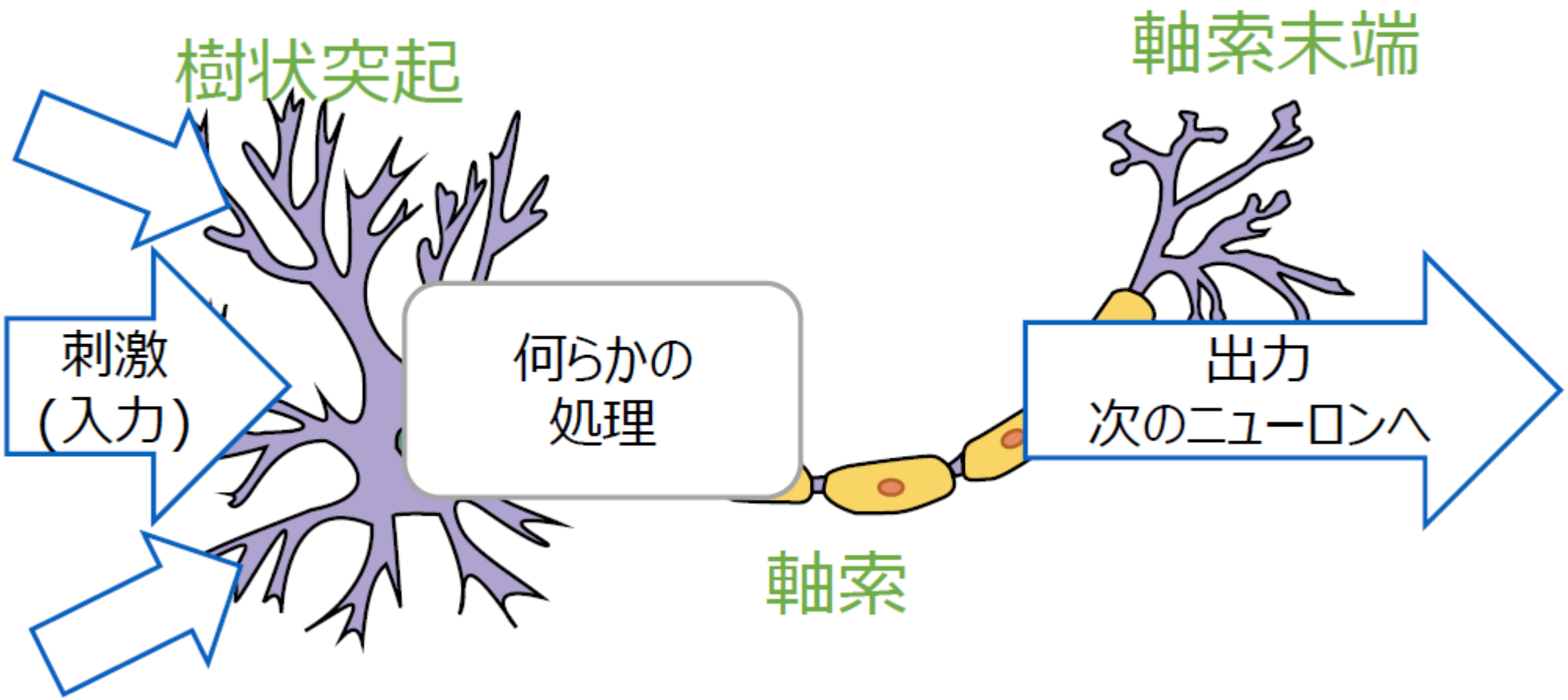
脳に関する素朴な疑問

- 脳はニューロンの集合体
- どこのニューロンを見ても機能は(ほぼ)同じ

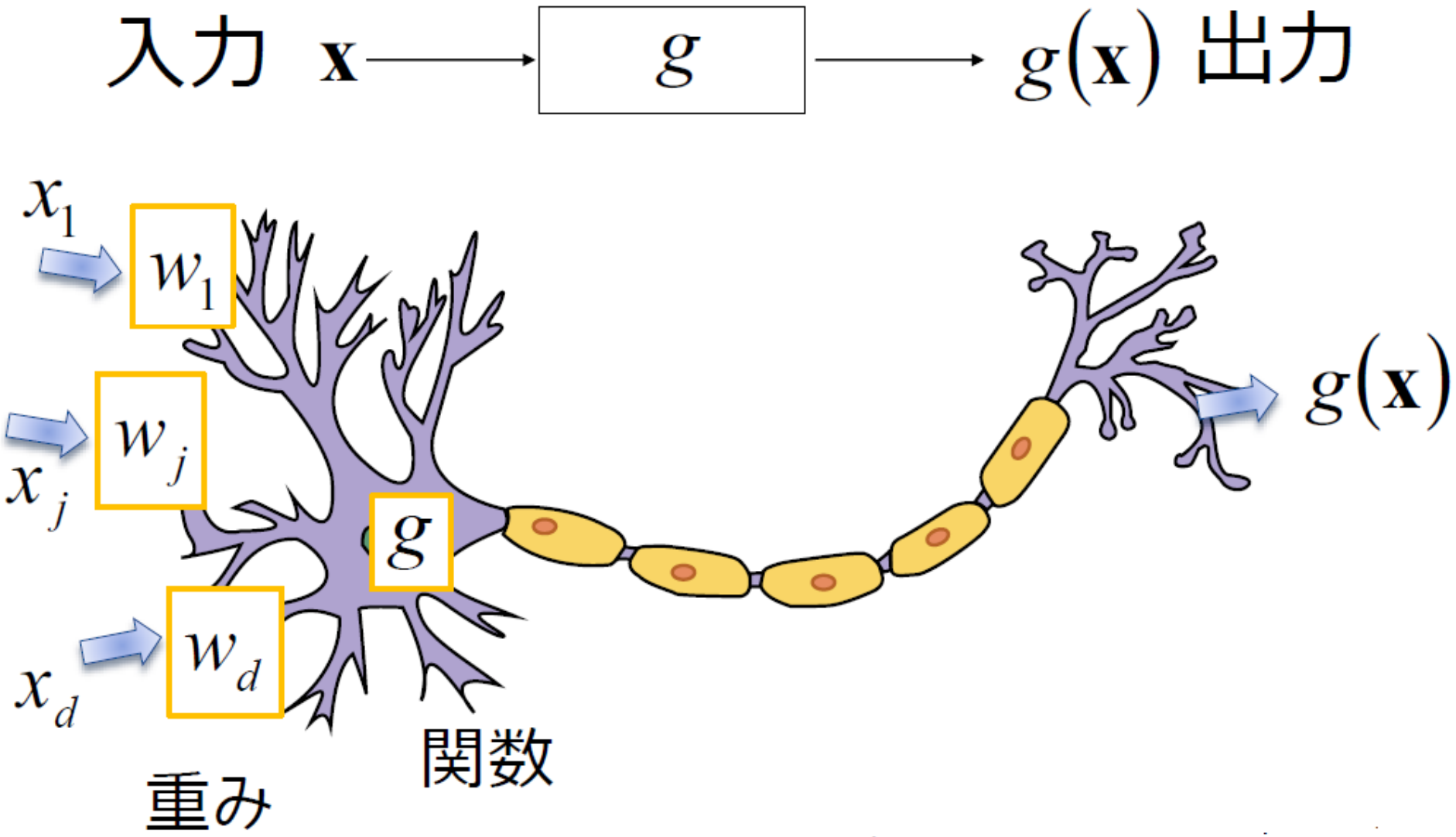


- なんで同じものを寄せ集めているだけなのに，
  脳はこれだけ多様な機能を実現できるのか？
  - 認識機能の実現だけでも，特徴抽出と識別が必要なのに…

神経細胞（ニューロン）

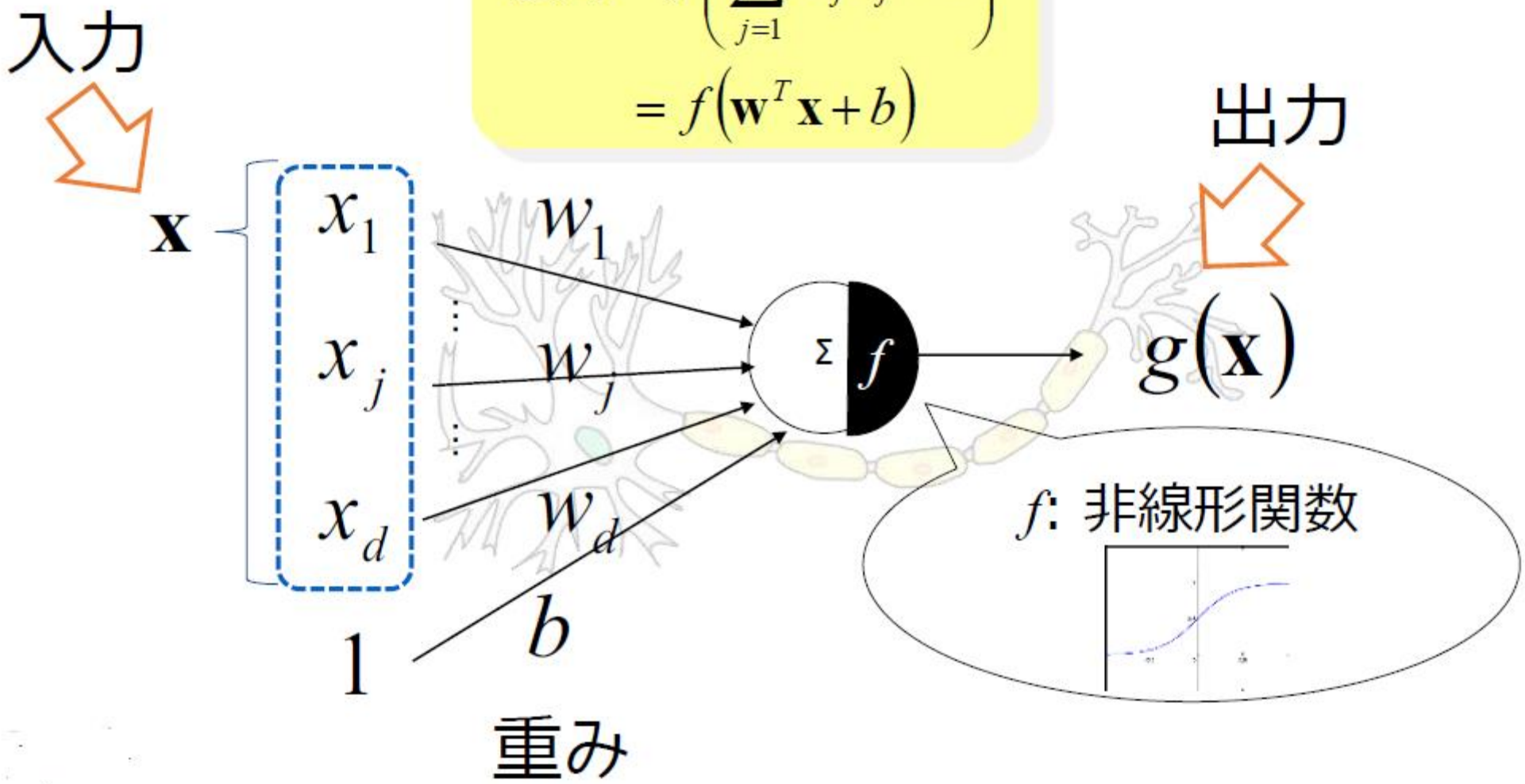


ニューロンの計算モデルの概念



ニューロンの計算モデルをもっと詳しく

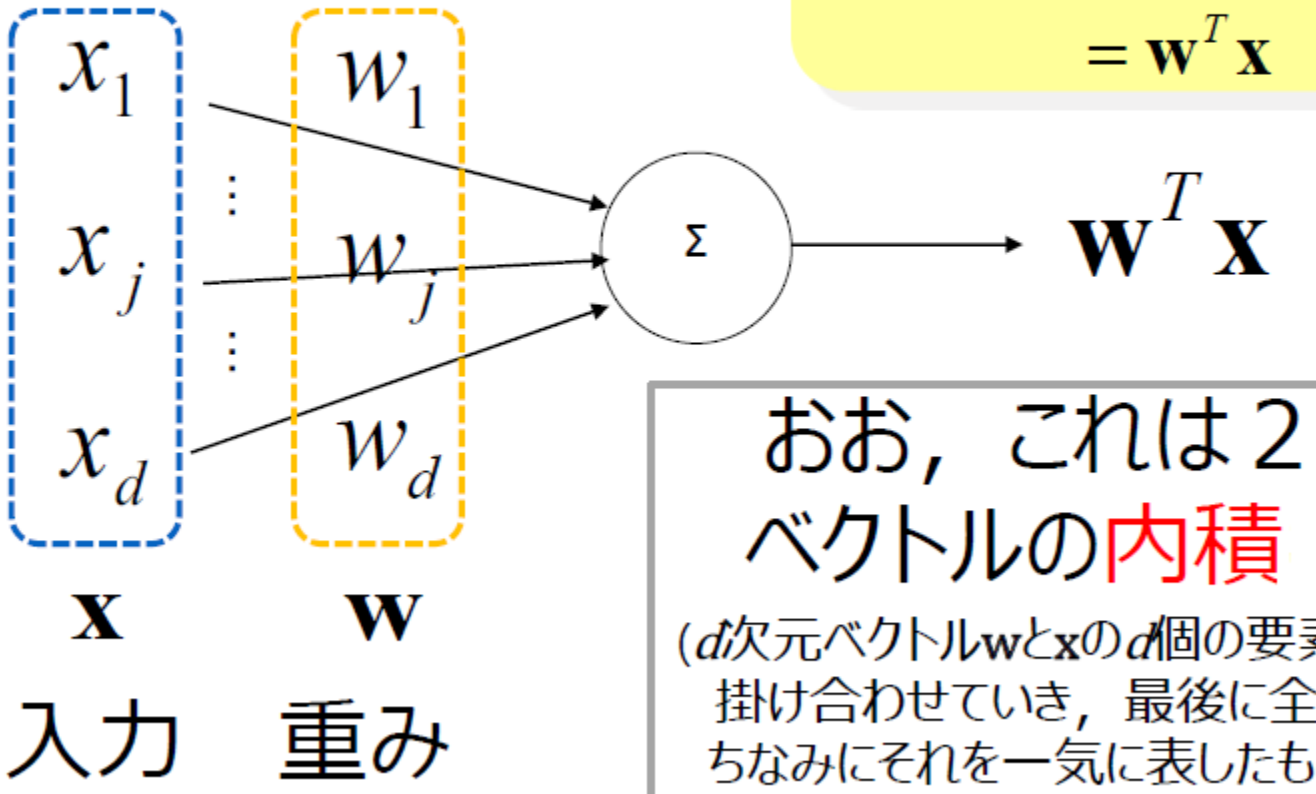
$$g(\mathbf{x}) = f\left(\sum_{j=1}^{d} w_j x_j + b\right) = f\left(\mathbf{w}^T \mathbf{x} + b\right)$$



計算モデルを簡単にすると

$$g(\mathbf{x}) = \sum_{j=1}^{d} w_j x_j$$

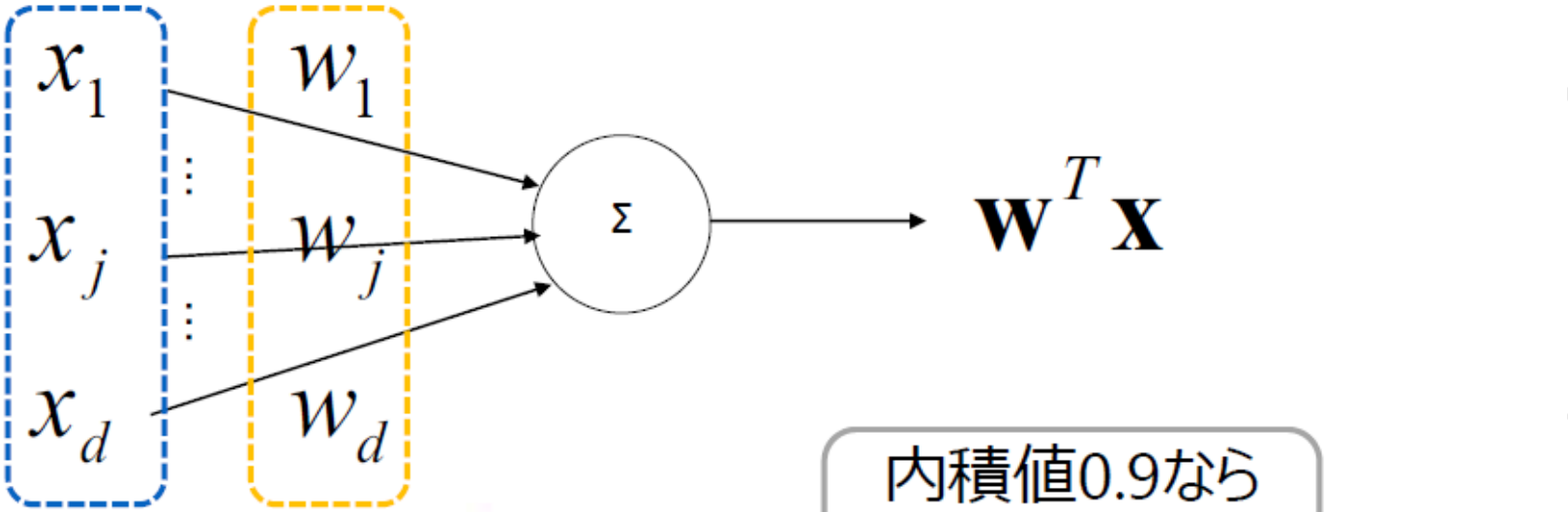
$$= \mathbf{w}^T \mathbf{x}$$



おお，これは２つのベクトルの内積

（d次元ベクトル$\mathbf{w}$と$\mathbf{x}$のd個の要素それぞれを掛け合わせていき，最後に全部足す．
ちなみにそれを一気に表したものが $\mathbf{w}^T\mathbf{x}$）

ニューロンは内積（類似度）？



入力　重み

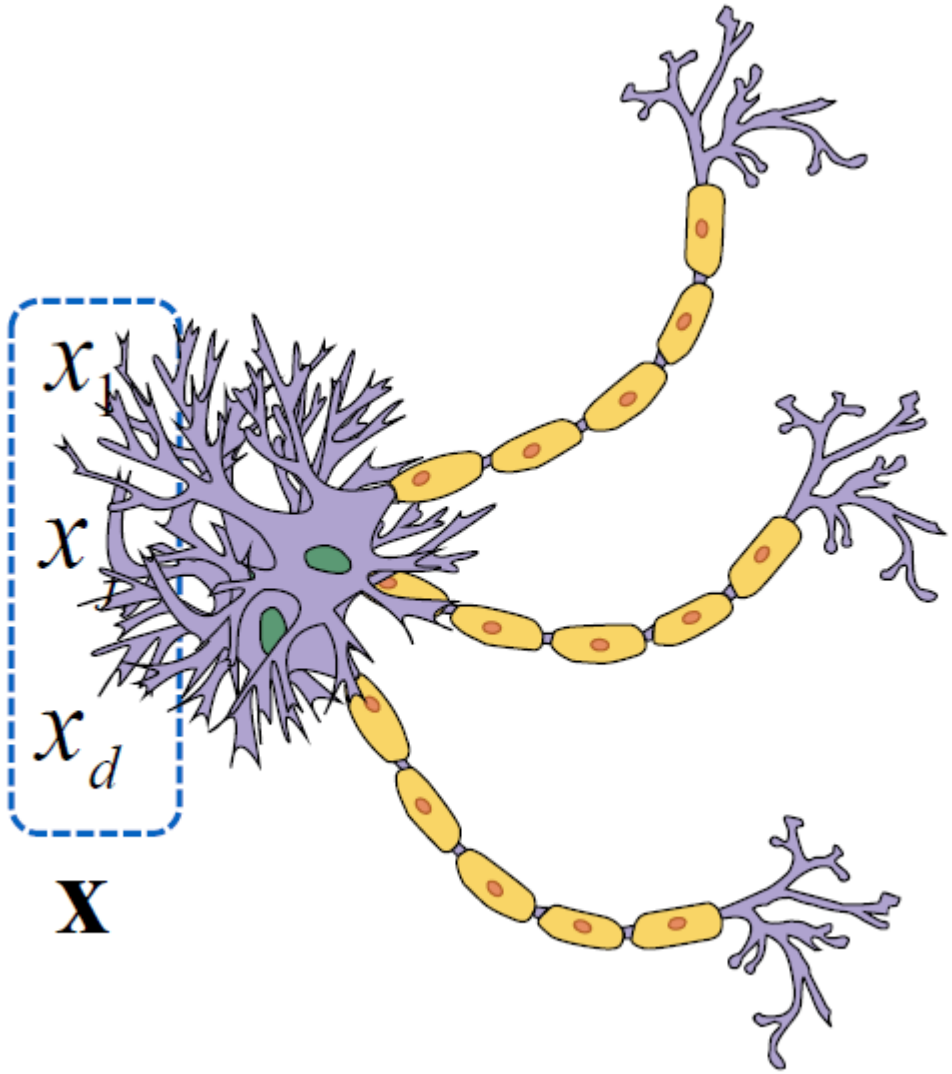
内積値0.9なら
似てる！

内積値0.02なら
似てない

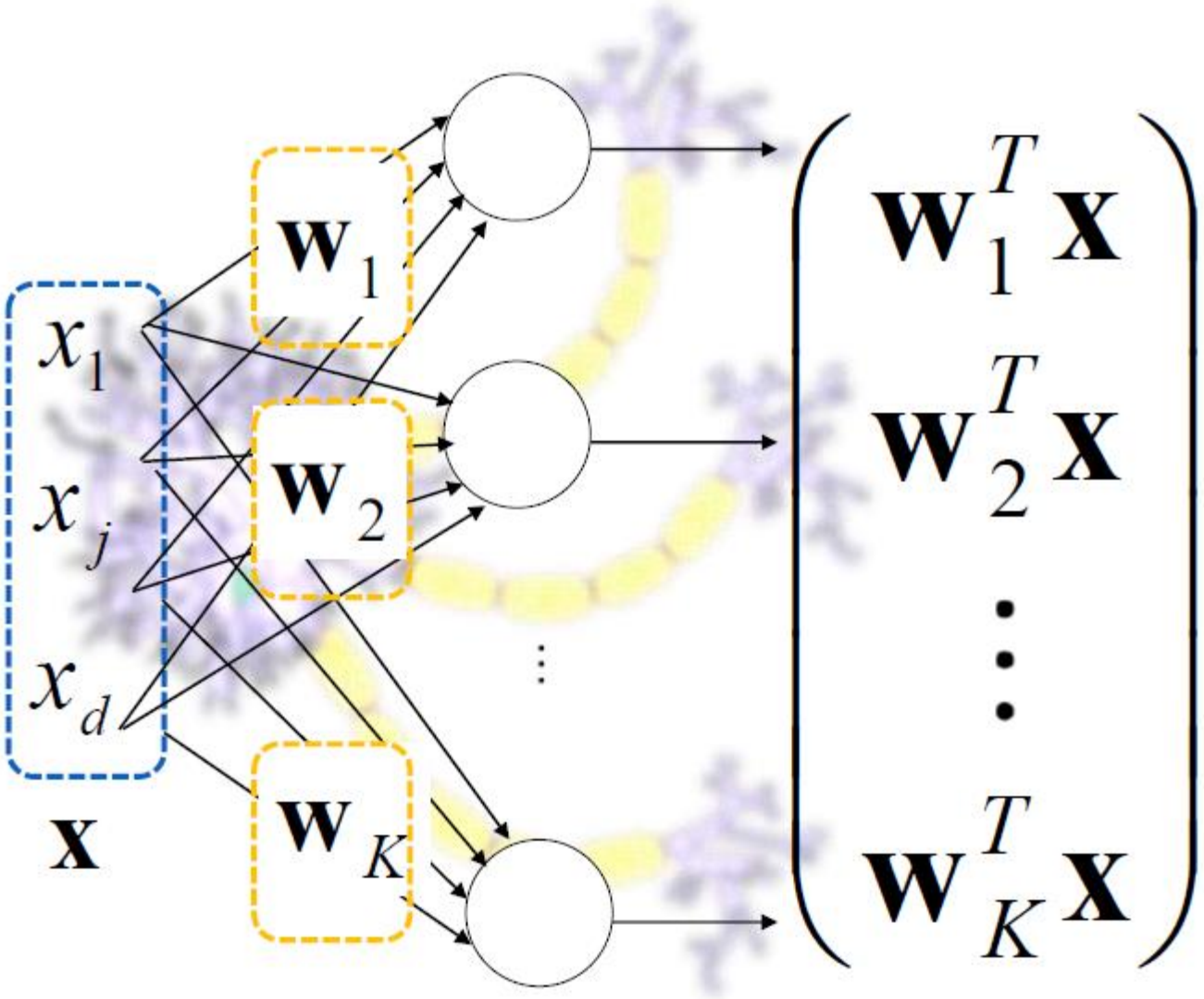
似たベクトルの内積値は大きくなる

→内積は入力$\mathbf{x}$ と重み$\mathbf{w}$の類似度を表現

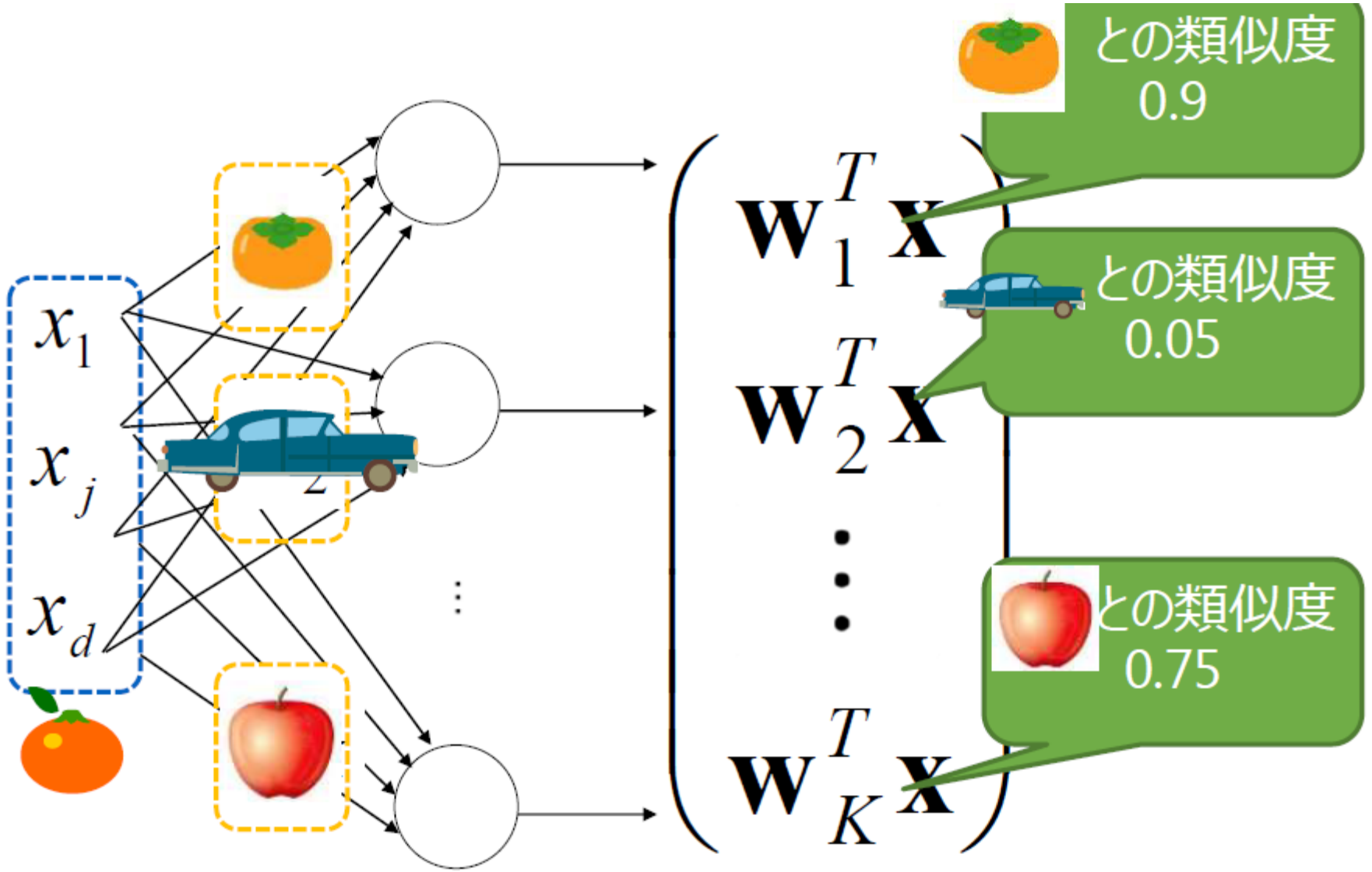
K個のニューロンがあればK個の類似度が…



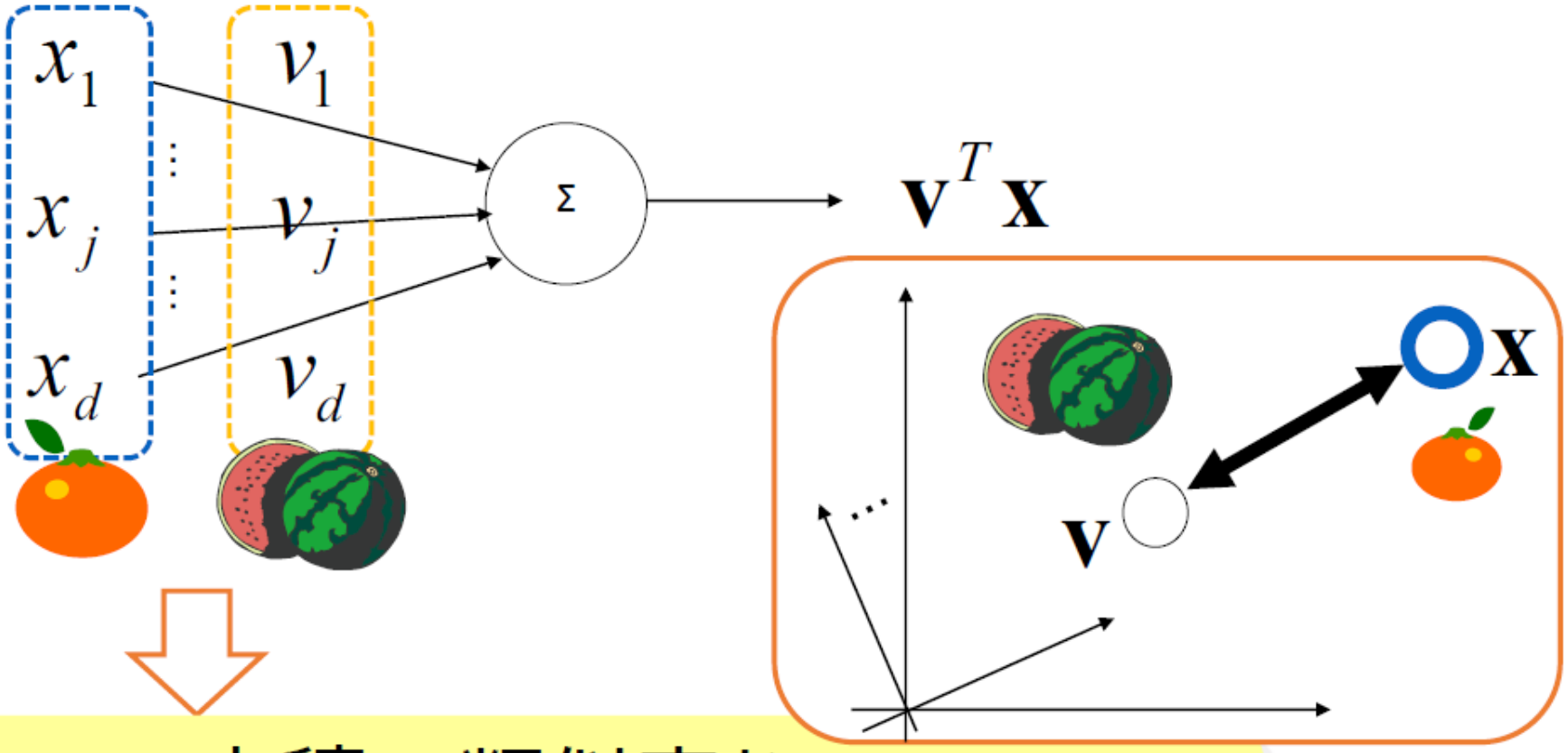
K個のニューロンがあればK個の類似度が…



*K*個のニューロンがあれば*K*次元の特徴（類似度）が出せる…

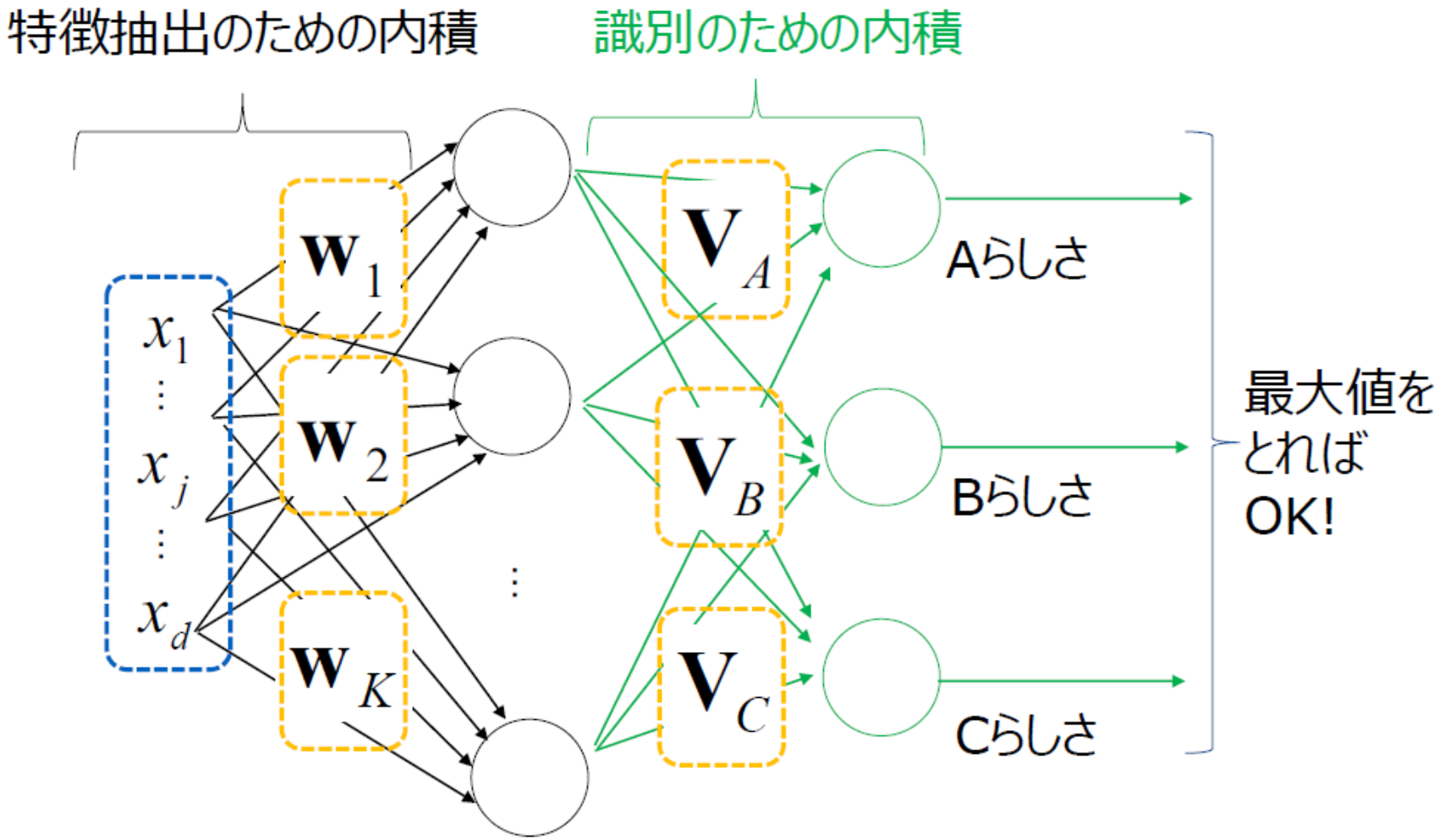


内積は分類にも使用できる・・・



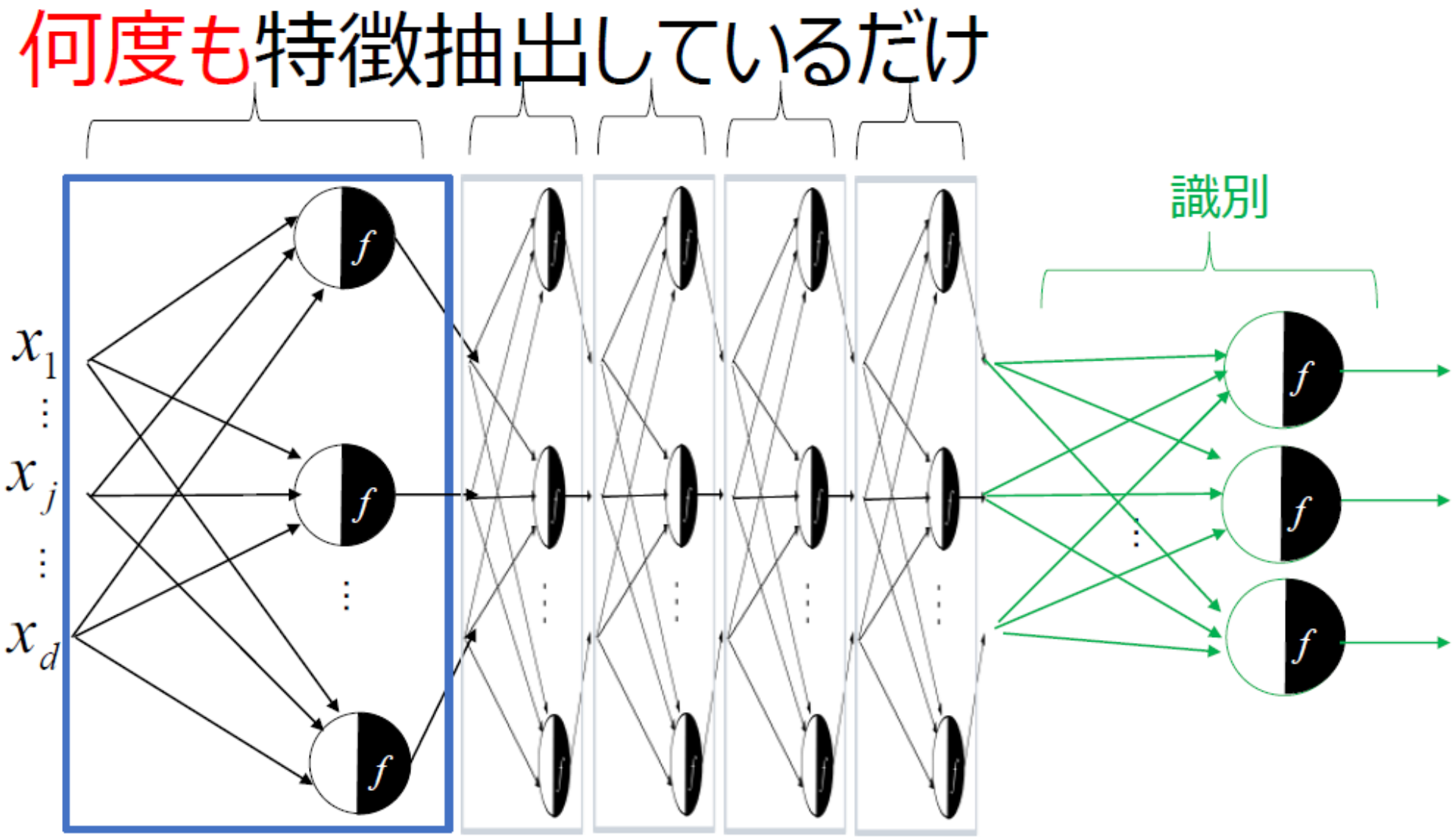
内積＝類似度なので
最近傍識別のための類似度にも使える

ニューラルネットワーク・・・

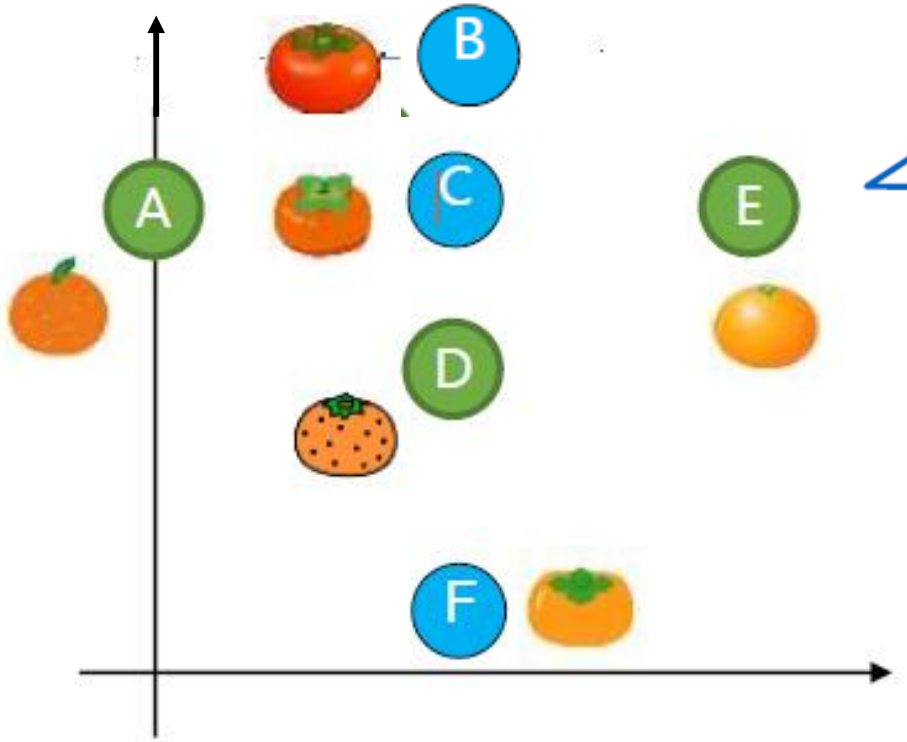


内積だけで特徴抽出と識別(分類)の両方を実現！

次にディープニューラルネット・・・

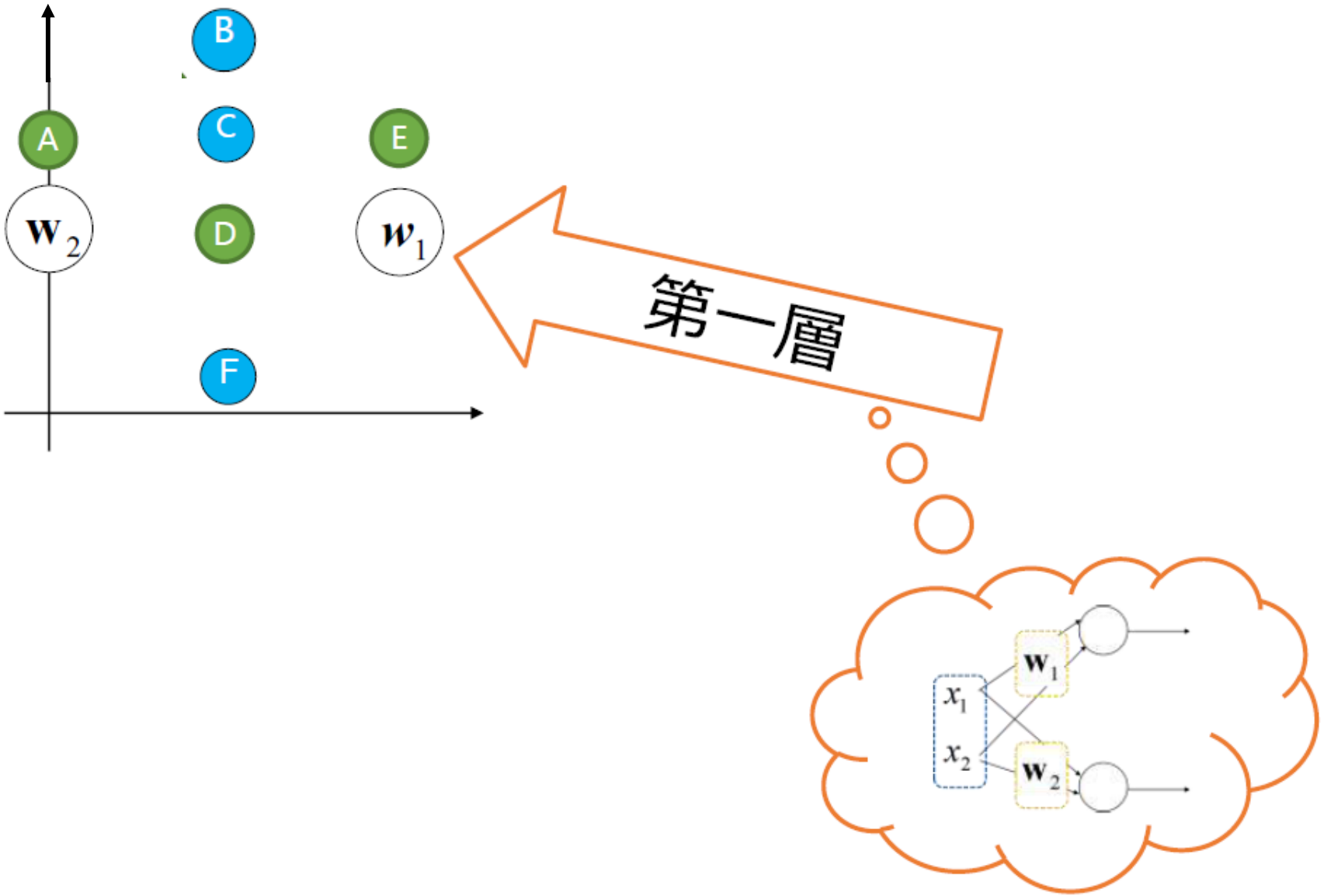


なぜ何度も特徴抽出を行うのか？

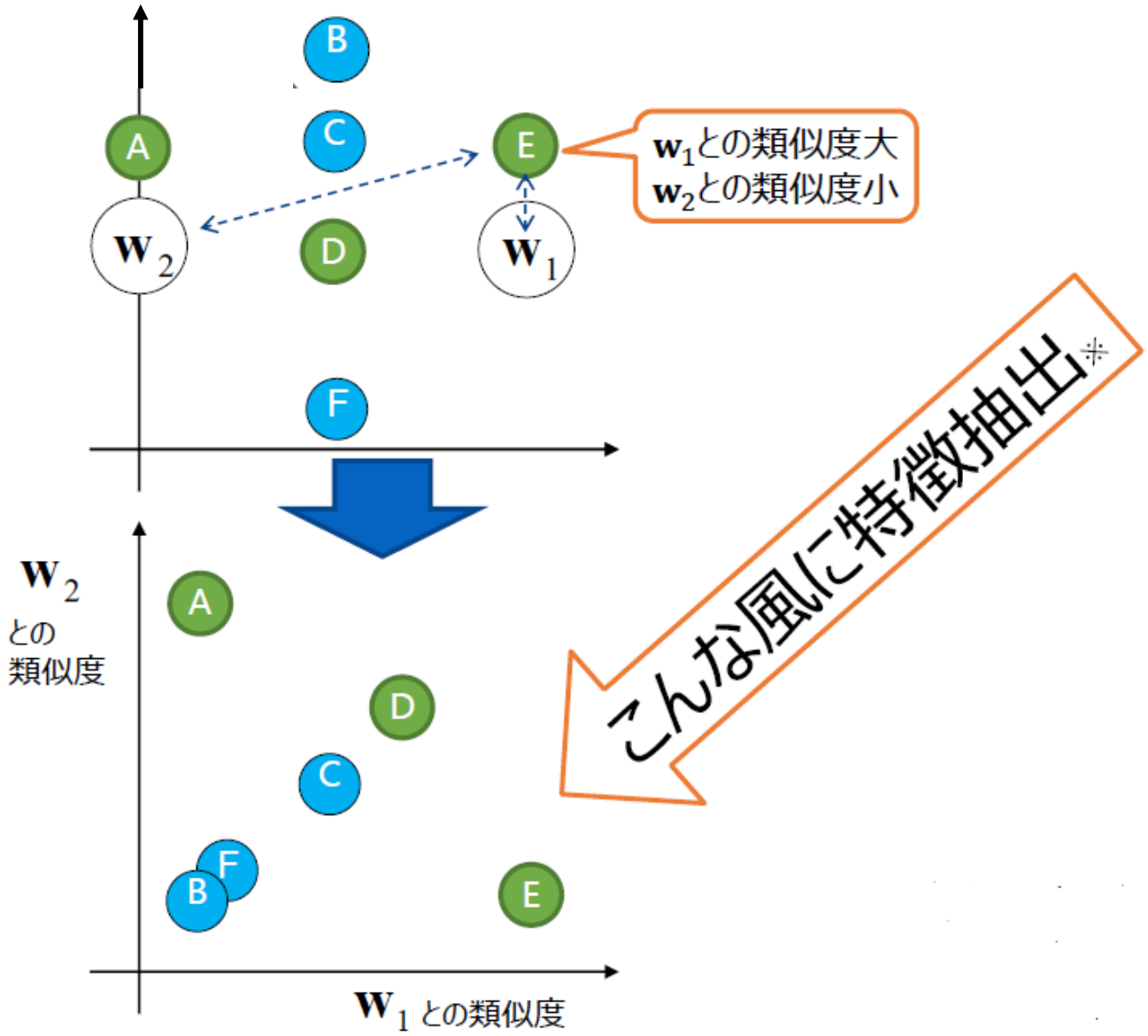


例えば，
みかんと柿の認識問題：
ゴチャゴチャ混ざっていて，
誤認識が起きそう

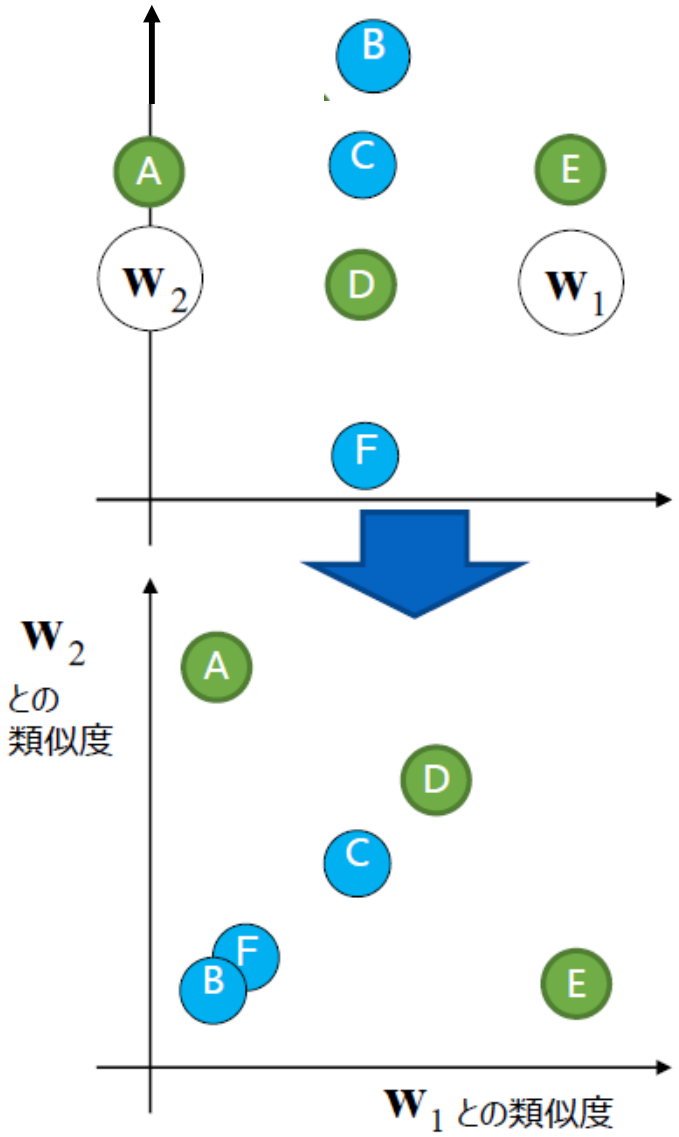
なぜ何度も特徴抽出を行うのか？



なぜ何度も特徴抽出を行うのか？

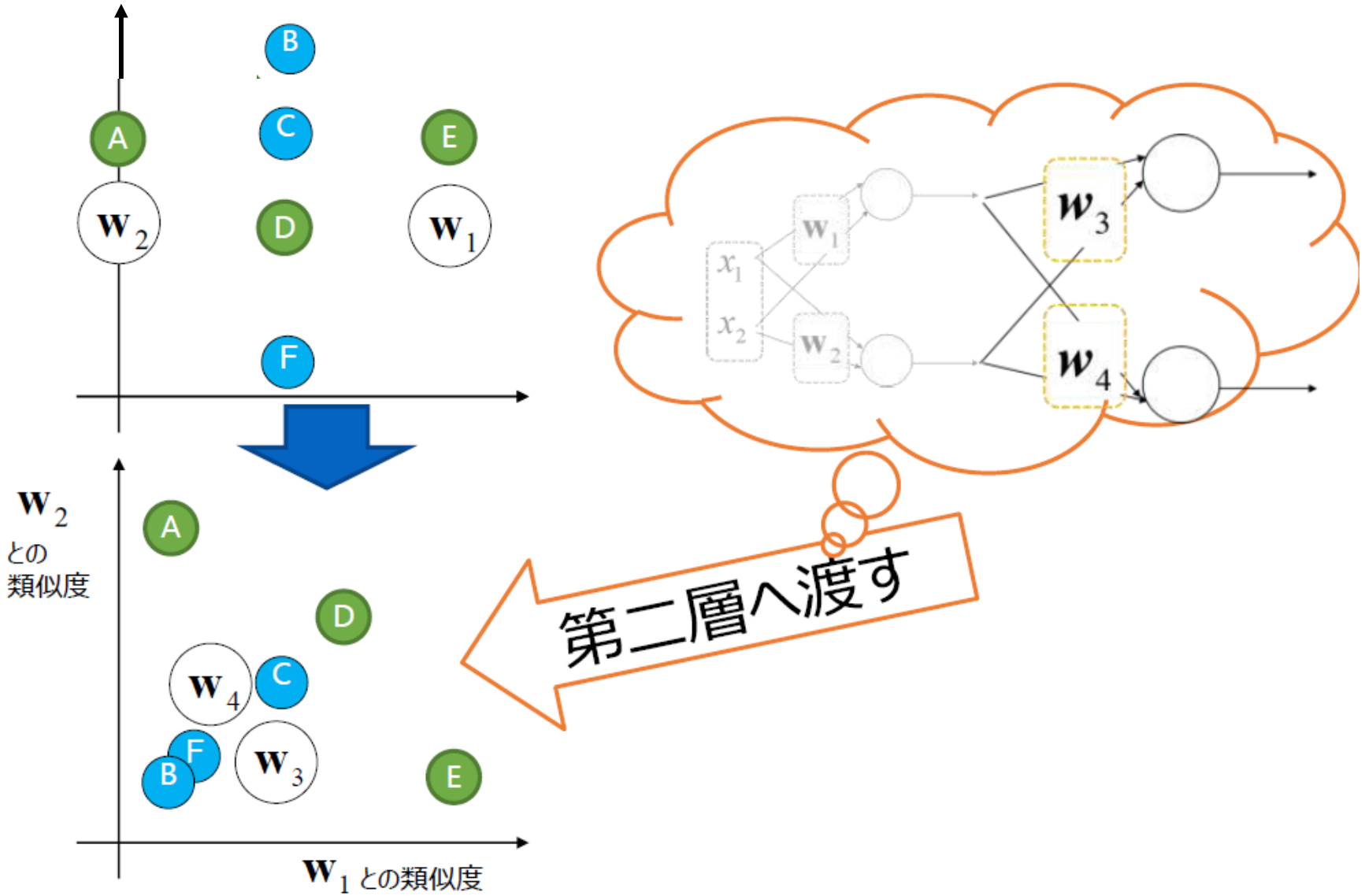


なぜ何度も特徴抽出を行うのか？

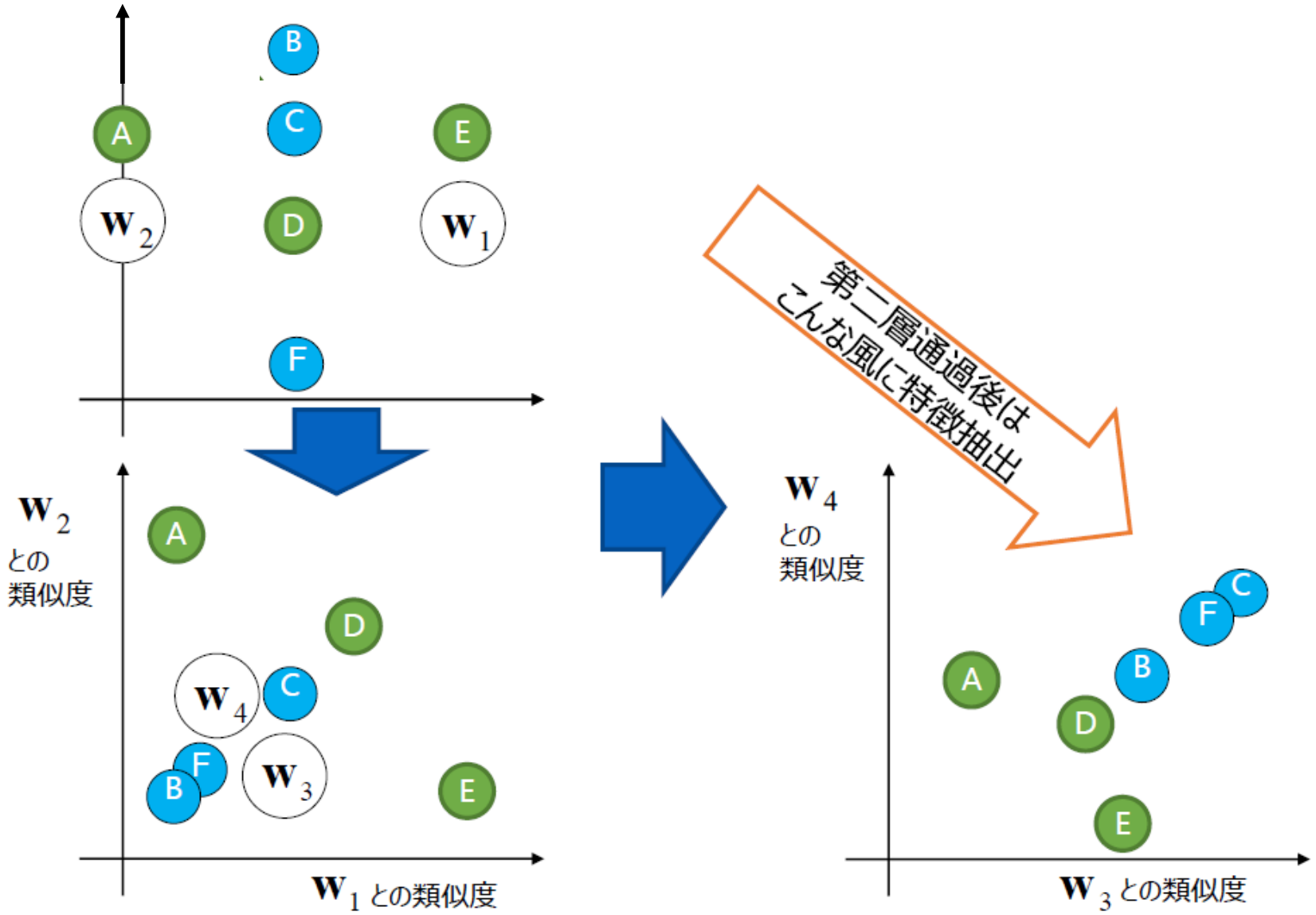


上の状況より，
よりゴチャゴチャ度は減ったが
もうちょっとキレイに分けたい
(専門的には「まだ線形分離不能」)

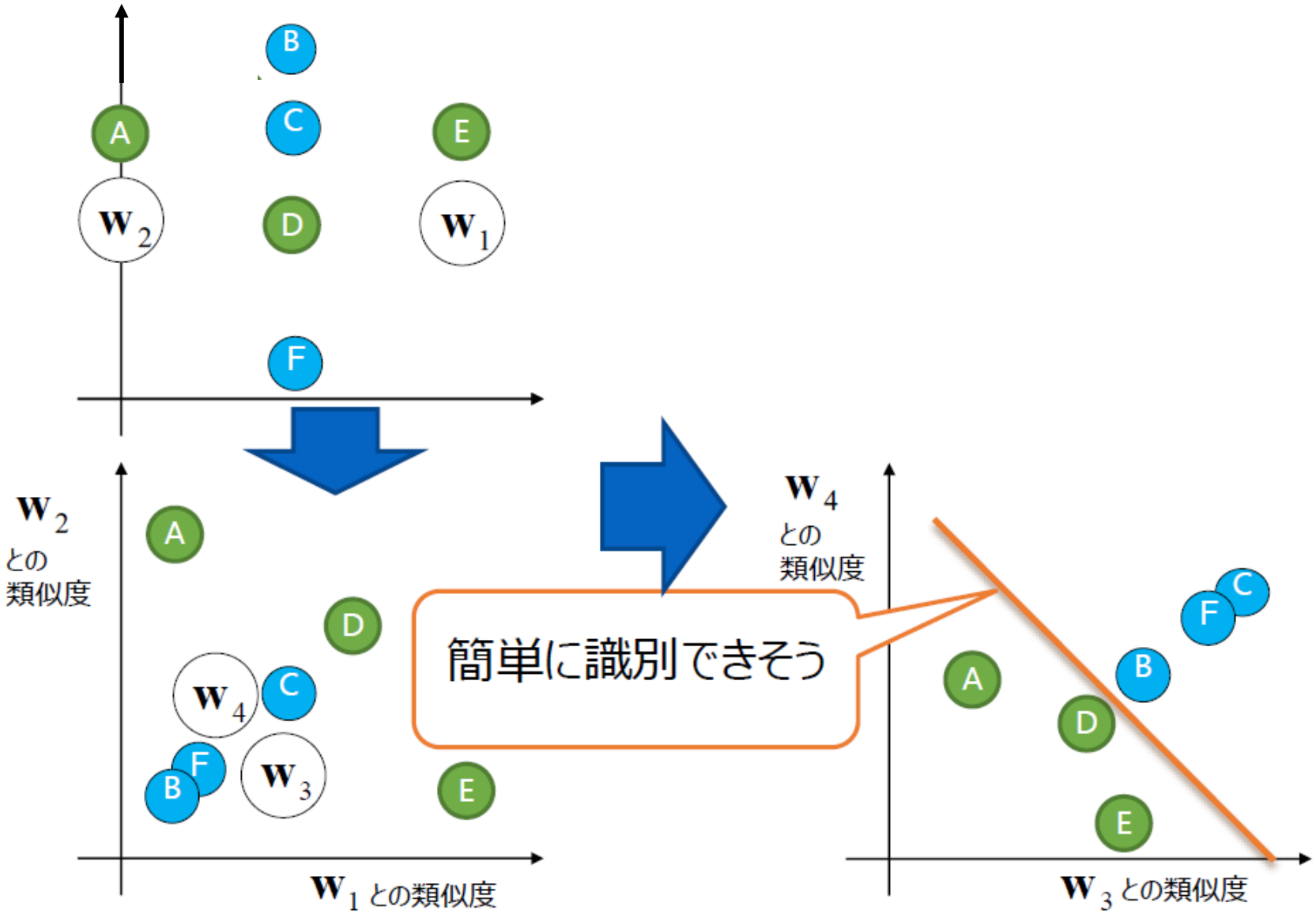
なぜ何度も特徴抽出を行うのか？



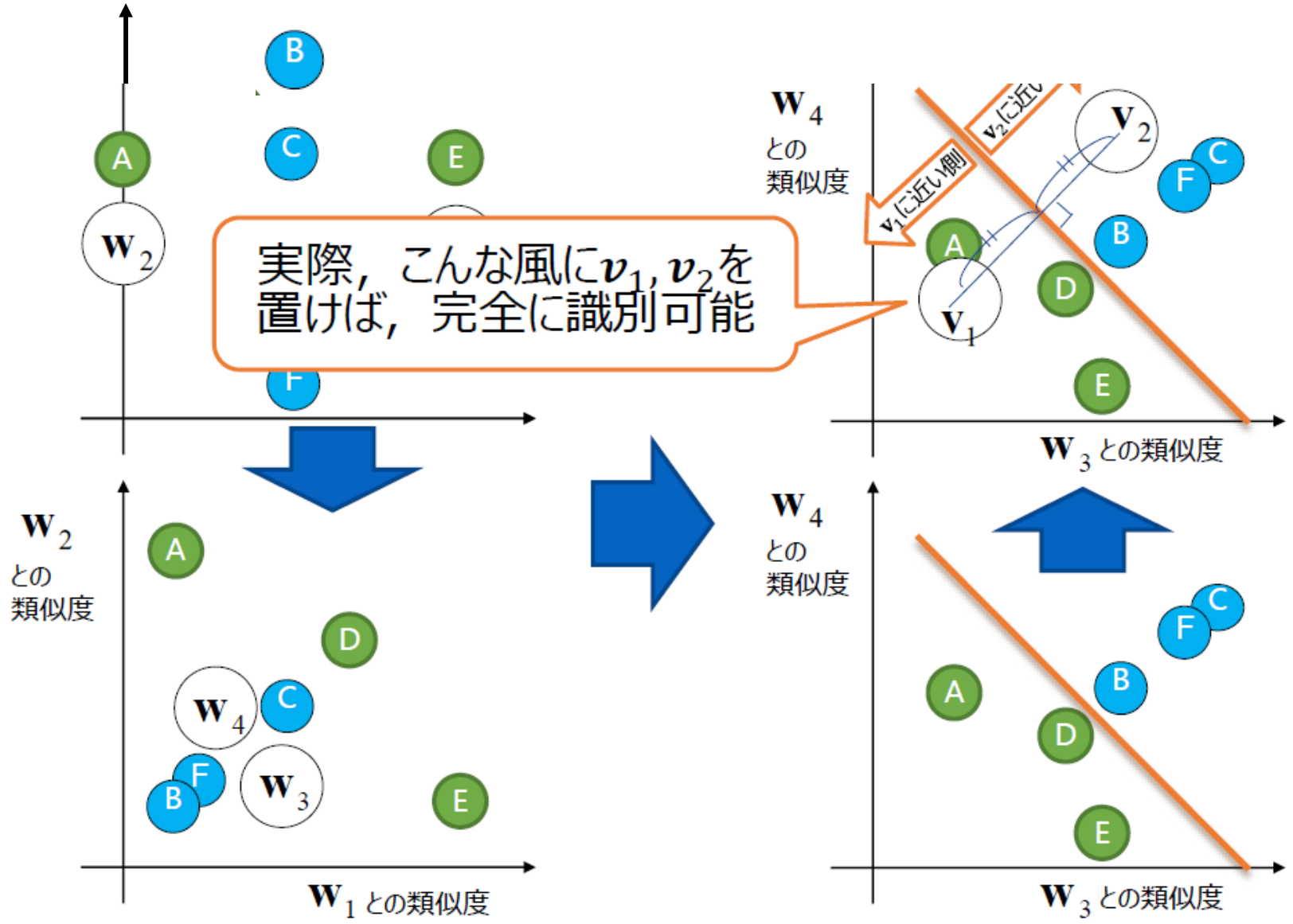
なぜ何度も特徴抽出を行うのか？



なぜ何度も特徴抽出を行うのか？

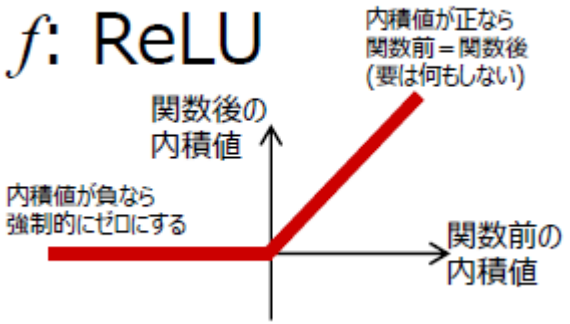


なぜ何度も特徴抽出を行うのか？



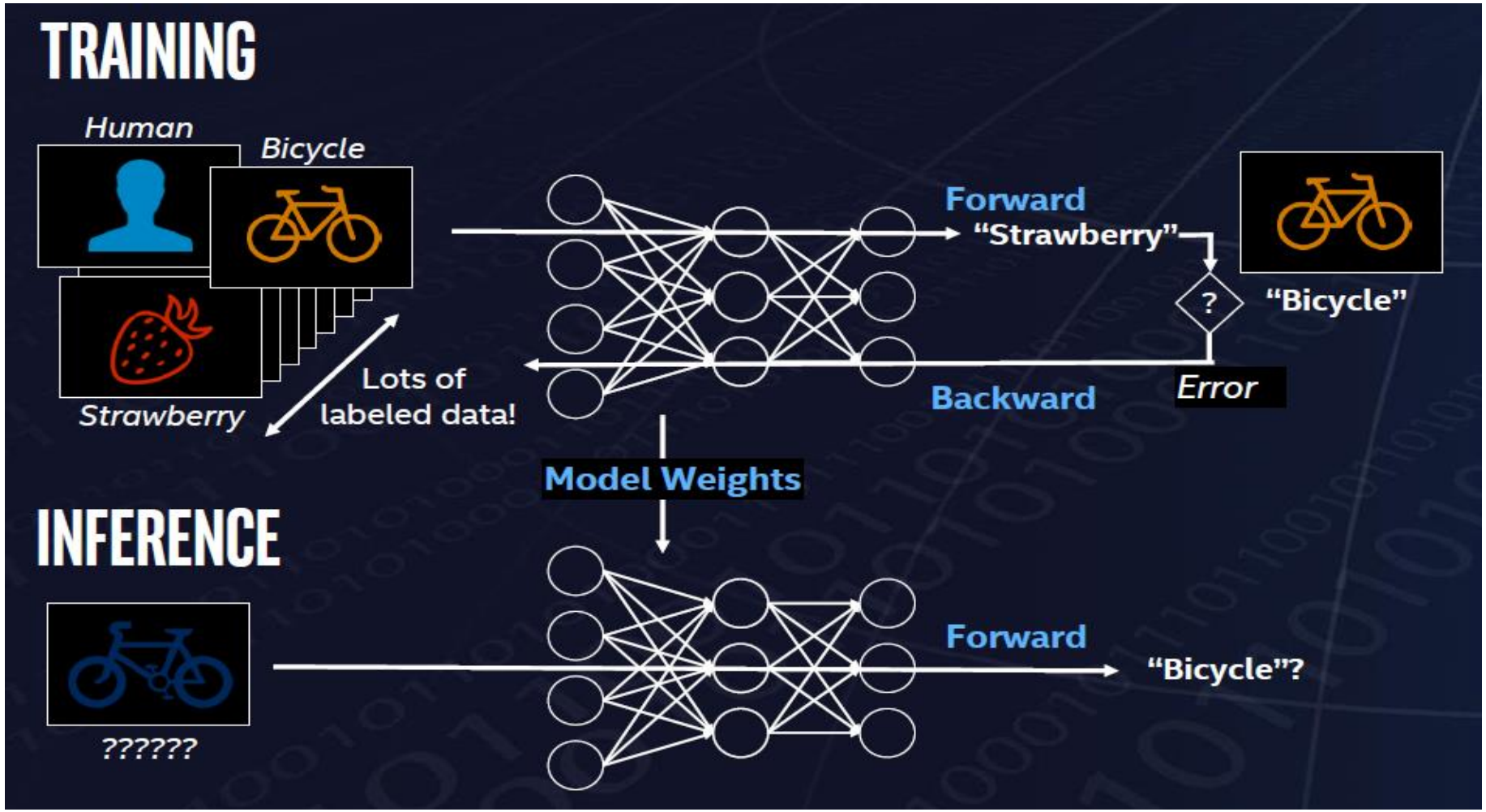
なぜ何度も特徴抽出を行うのか？

- 要するに、各層で空間変換(線形変換)をやって、なるべく分離識別しやすいようにしている
- それを何度も繰り返す(=多層を通す）ことで、どんどん分離識別しやすくなるはず・・・
- ただし、空間変換時に、ちょっとだけ非線形処理が入る



ディープラーニングにおける学習のしかた

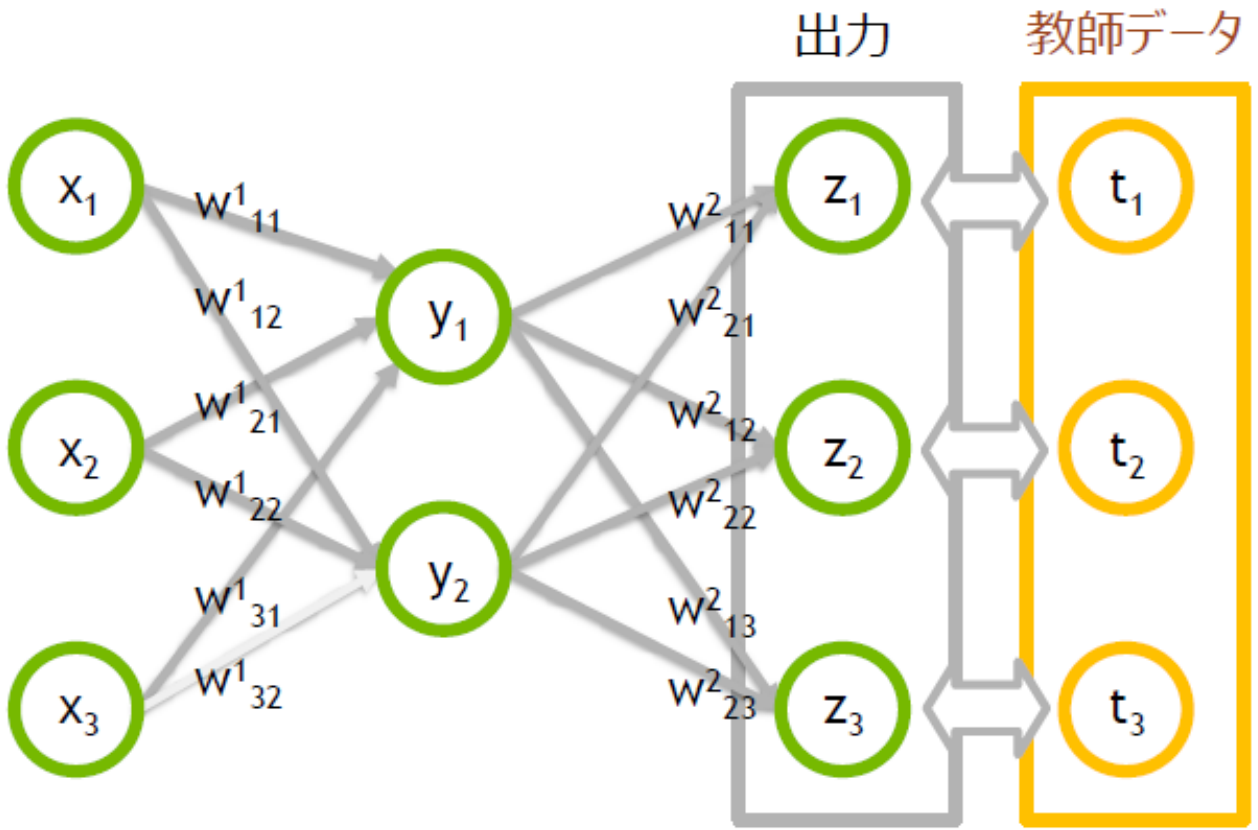
ディープラーニングの学習と推論



高精度な学習には大量の学習データと非常に深い層が必要

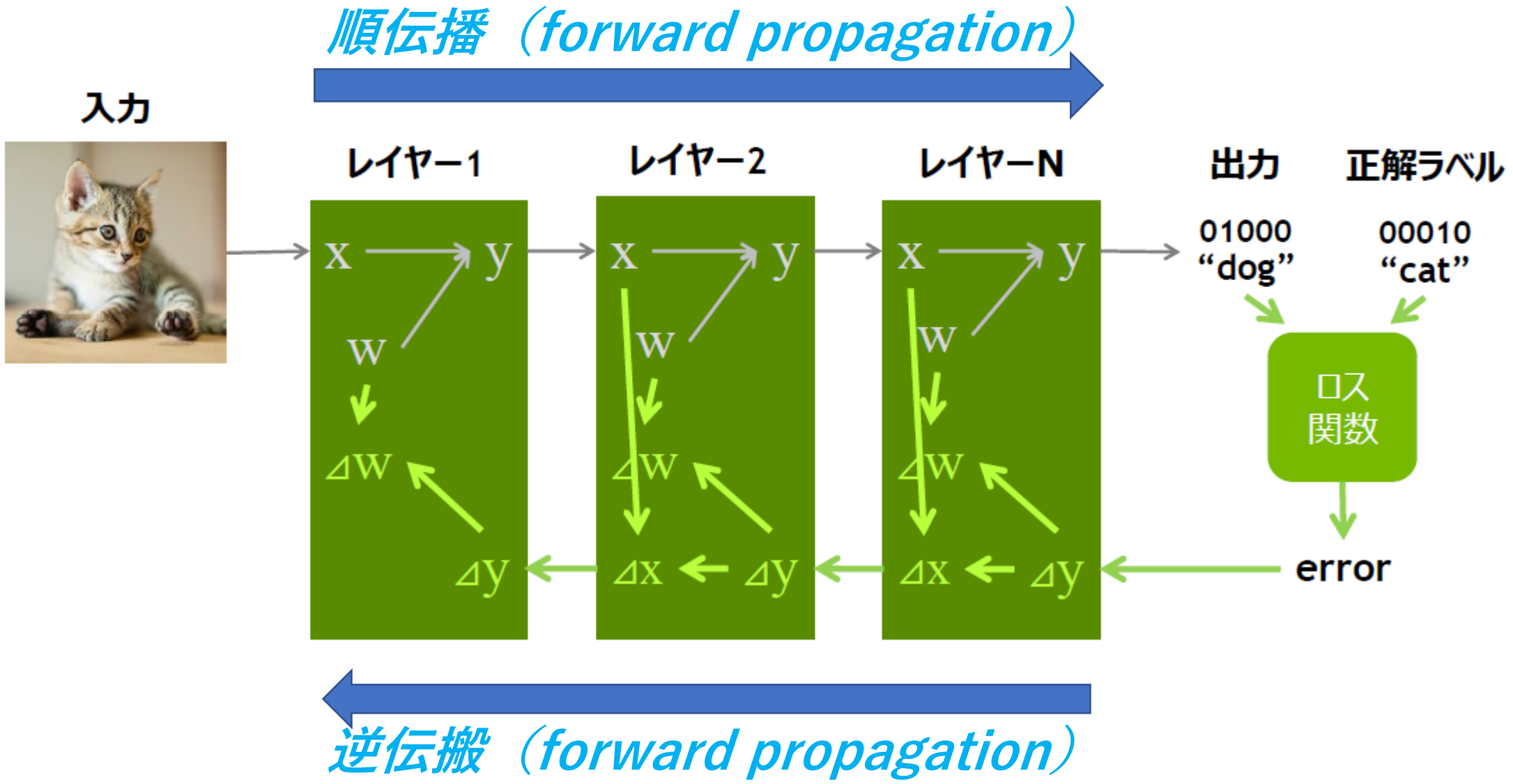
ディープラーニングの学習とは・・・

- ニューラルネットワークの構造の重みの調整

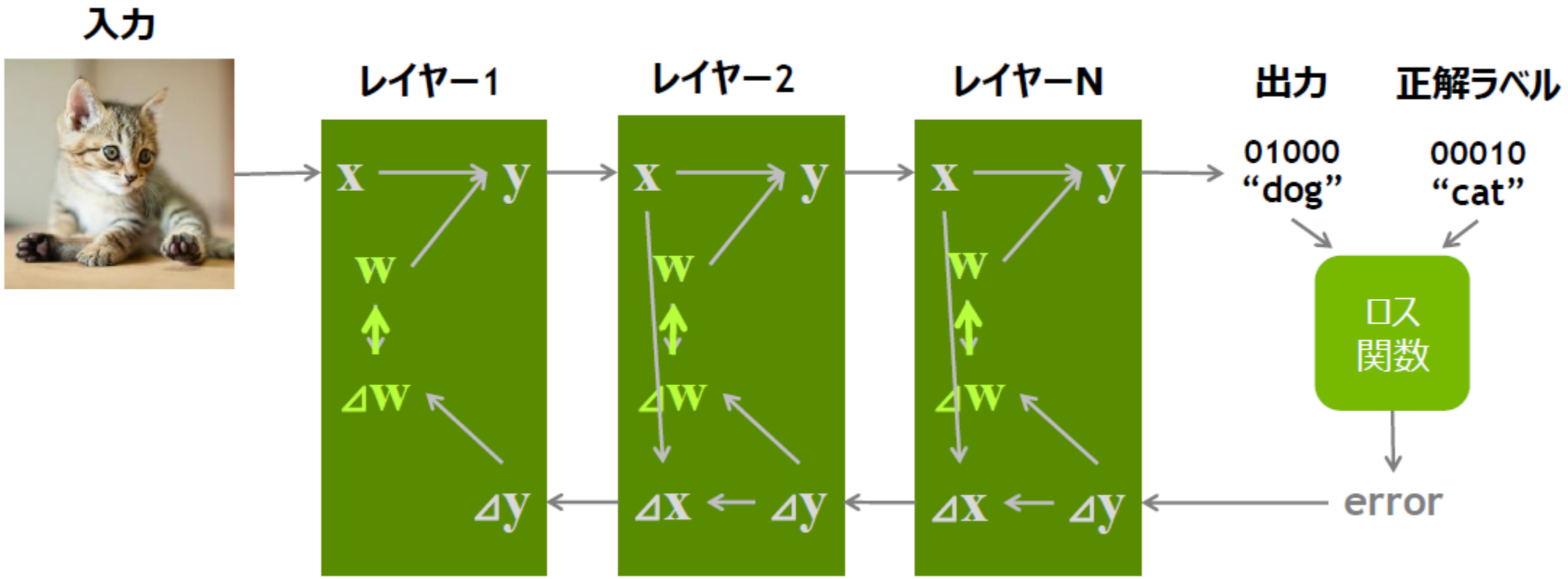


出力z と正解ラベルt の誤差を計算し、出力が正解ラベルに近づくように重みW1とW2を調整

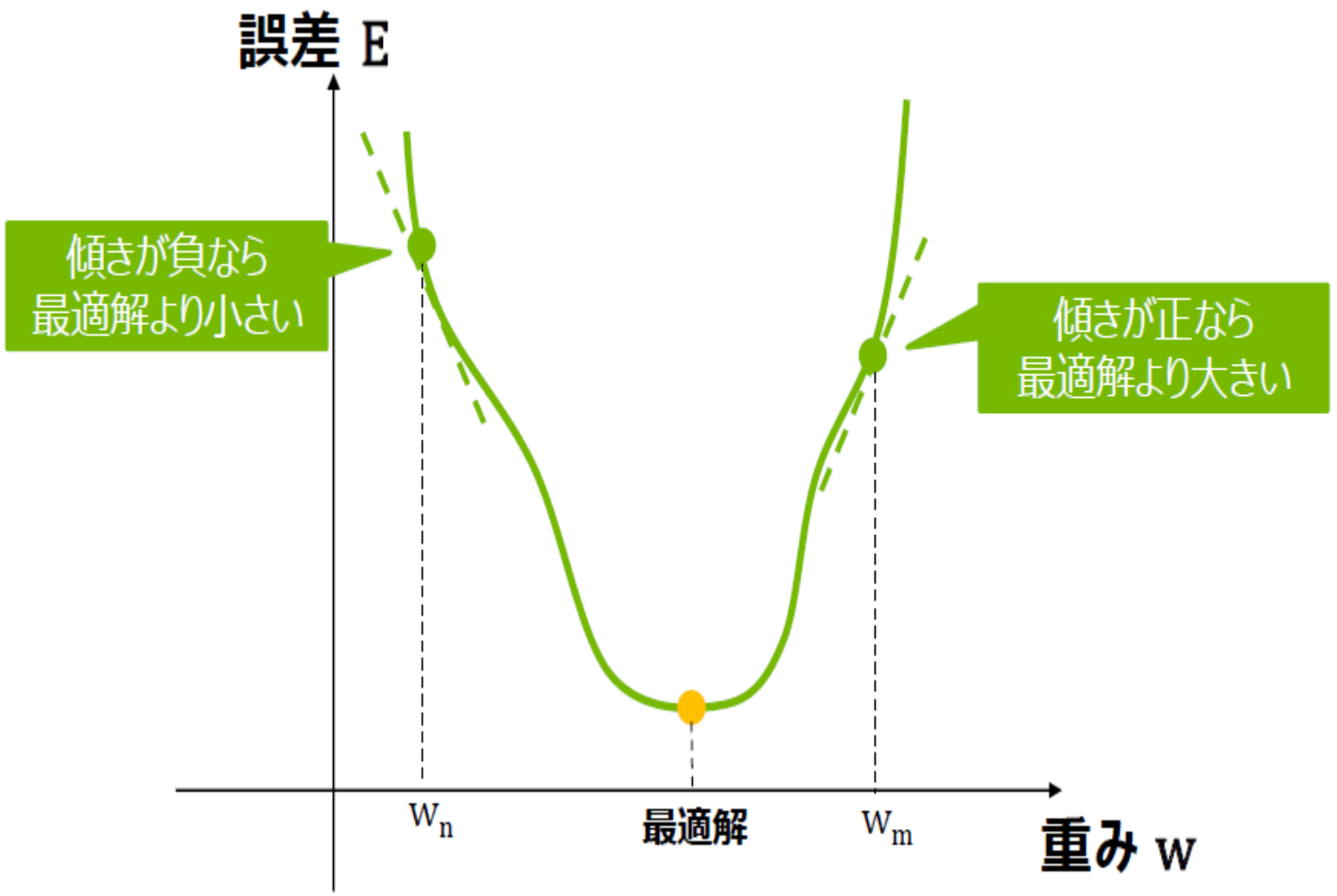
誤差逆伝播法

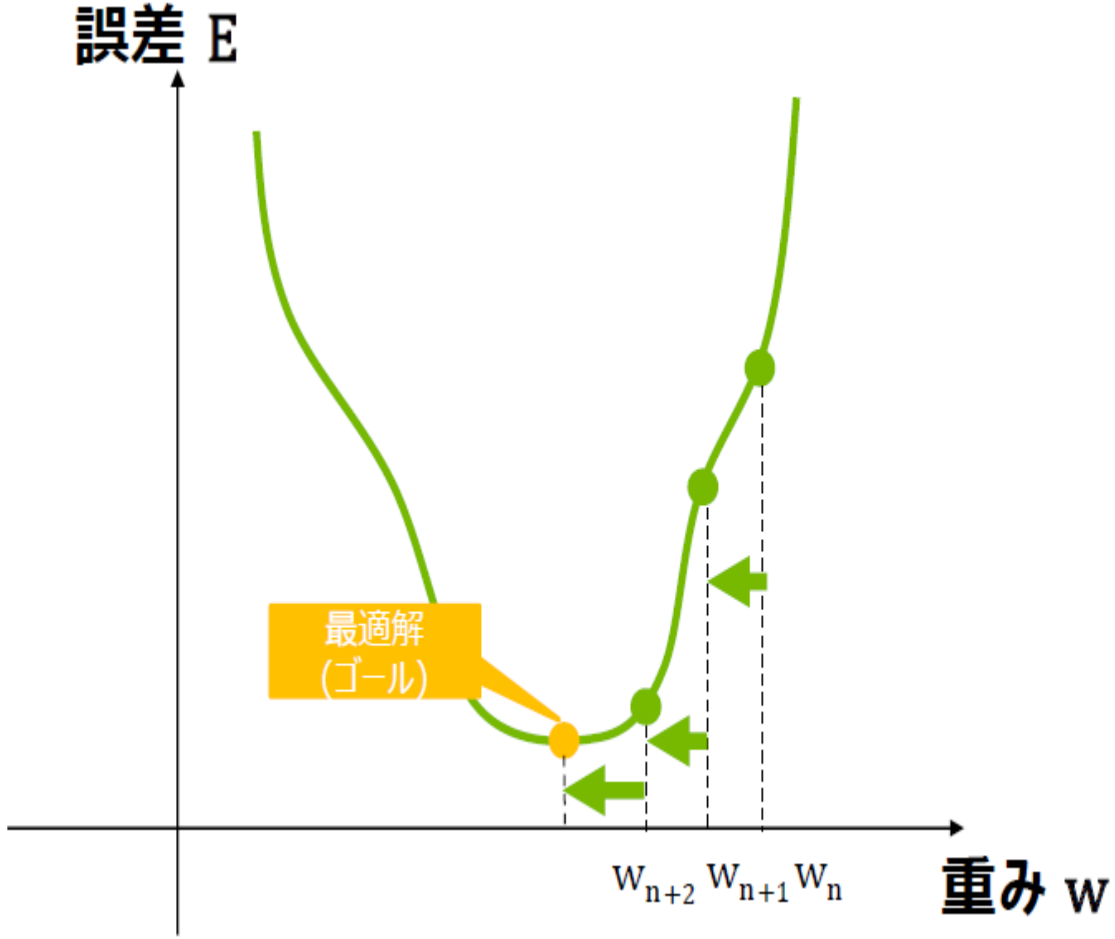

順伝播（forward propagation）
入力
レイヤー1
レイヤー2
レイヤーN
出力
正解ラベル
01000
“dog”
00010
“cat”
x
y
w
⊿w
⊿y
⊿x
ロス
関数
error
逆伝搬（forward propagation）

重みの更新（誤差逆伝播法）

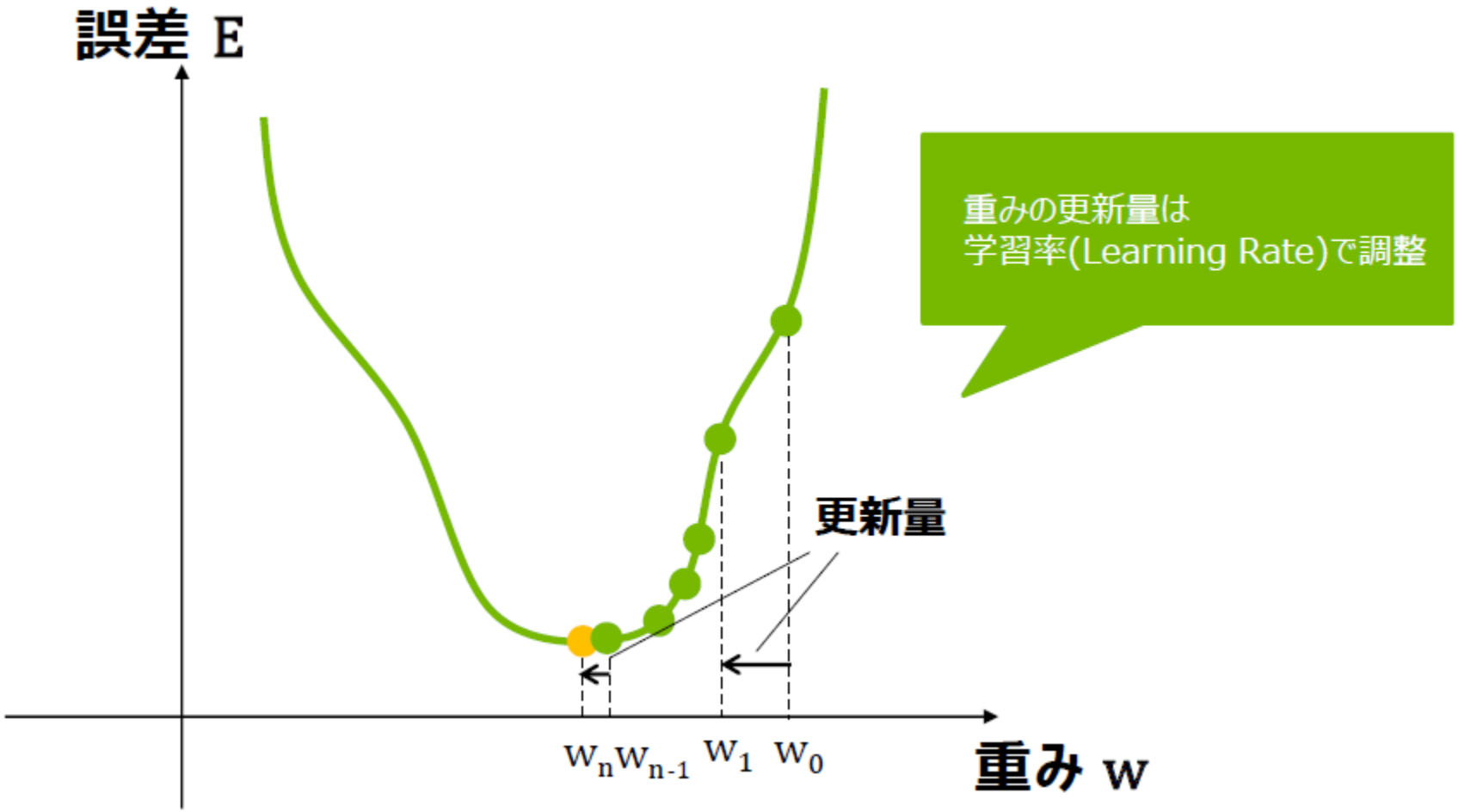


勾配降下法（誤差逆伝播法）



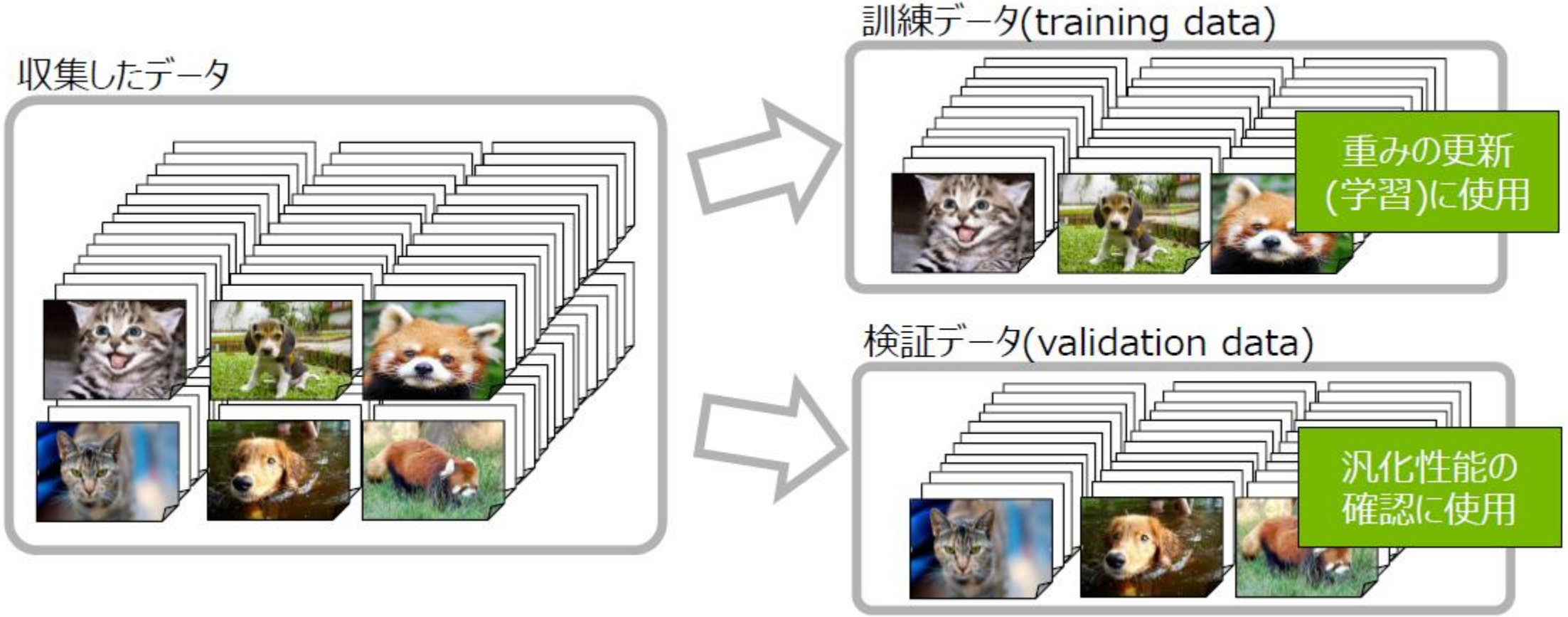


学習率（誤差逆伝播法）



学習データと検証データ

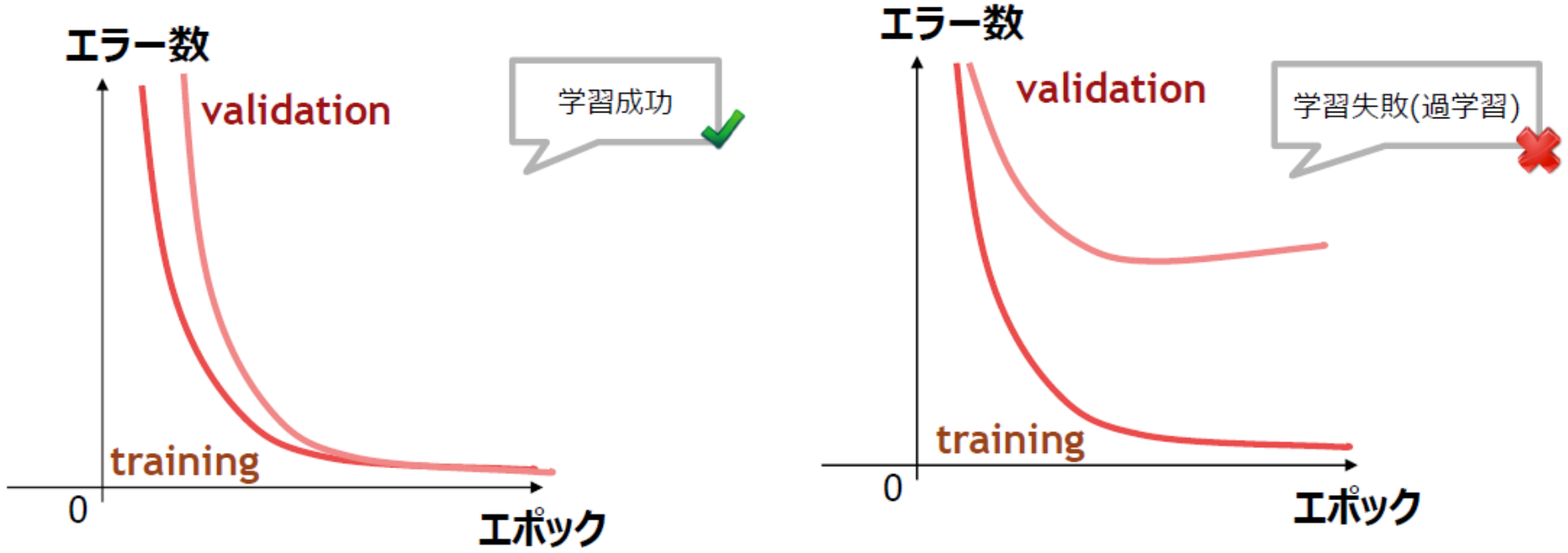
- データを**訓練データ(training data)**と**検証データ(validation data)**に分割する



学習時の性能の確認

すべての訓練データを用いて重み更新を行う+ すべての検証データで汎化性能を確認⇒**1 エポック(epoch)**

各エポックで訓練データをニューラルネットワークに与えた際の間違い率と検証データを与えた際の間違い率を確認しながら学習を進める必要がある



ディープニューラルネットワークによる画像認識

ディープラーニングと従来の画像処理の違い

ディープラーニングによる画像処理

・画像の特徴を抽出するために多数のフィルタを使用。
・オブジェクトの特徴はオブジェクトのタイプついて各入力画像を出力ノードに関連付けるという目的で解析される。
・画像が出力ノードに関連付けられたオブジェクトである確率を表す値が出力ノードに割り当てられる。

従来の画像処理

・重要な特徴を抽象化し、それらをオブジェクトに関連付けるための演算フィルタ選択と接続に基づく。
・明確に定義されたオブジェクトや制御されたシーンのみで機能する。
・多数のオブジェクトやさまざまなシーンで重要な機能を予測するのが困難。

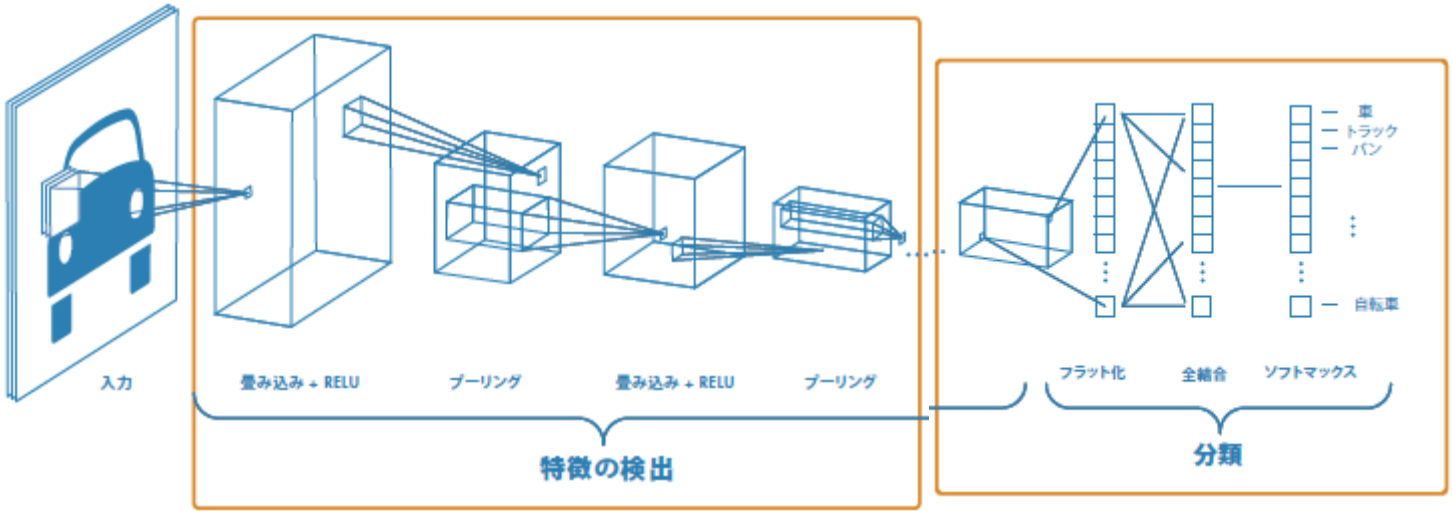
多層ニューラルネットワークにおける画像処理上の弱点

- データの形状が壊れてしまう
  - 入力画像は２次元であるのに対して、多層NNの入力は１次元配列で、そのため２次元配列を１次元配列に変換する必要がある。
  - このように次元の変換はデータの形状を壊してしまう。
- 学習するパラメータの数が多い
  - 各層への入力と各層のユニット同士がすべて結合している全結合層で、一つの層の重み数は、入力数 × ユニット数 必要。
    - 例えばMINSTデータセット（０～９の数値データ）の文字画像は28×28 の小さなもので、1層だけで784 × ユニット数 個の重みが必要。

同一クラスの画像のバリエーションの対応が困難（パラメータ数が爆発）
例えば、犬が右を向いている、左を向いている

畳み込みニューラルネットワークとは？

- 静止画や動画向けのディープラーニングで最も一般的なアルゴリズムです。（CNNまたはConvNet）
  - 特徴抽出関連の層
  - 分類関連の層



・どのような層をどう積み上げるかについての確実な公式はない。
・いくつか試してその精度を確認する。
・事前学習済みのネットワークを使用するのもよい。

- **特徴抽出のための層**
  - 畳み込み
    - 入力画像を複数の畳み込みフィルターに通します。それぞれのフィルターが画像の持っている特定の特徴に対して活性化する。
  - プーリング
    - 非線形のダウンサンプリングを実行して出力を単純化していきます。同時にネットワークの学習に必要なパラメーター数も削減する。
  - 活性化関数 (ReLU)
    - 負の値をゼロにマッピングし、 正の値を保持するように動作することで、より高速かつ効率的な学習を実現する。
- これら3つ層は数十層または数百層にわたって繰り返され、それぞれの層が異なる特徴量を検出するように学習が行われる。

- **分類のための層**
  - 最後から2番目の層は、K次元のベクトルを出力する**全結合層** (FC) である。Kはネットワークで予測可能なクラスの数を示す。このベクトルには、分類されるすべての画像の各クラスに対する確率が含まれる。
  - 最終層は、**ソフトマックス**関数を使用して分類結果を出力する。

畳み込みニューラルネットワークの特徴

- 畳み込み層は、元の画像にフィルタをかけて特徴マップを出力する
  - （構成性）。
- 特徴マップのサイズは元の画像より少し小さくなる
  - （元画像とフィルタのサイズによってサイズが変わる）。
- 画像全体をフィルタがスライドする為、特徴がどこにあっても抽出
  - （移動不変性または位置不変性）。
- フィルタは自動作成され、学習により変わってゆく
  - （誤差逆伝搬）。
- フィルタの数だけ特徴マップが出力される。

一般的なCNNの構造・・・

多クラス分類のネットワークの例

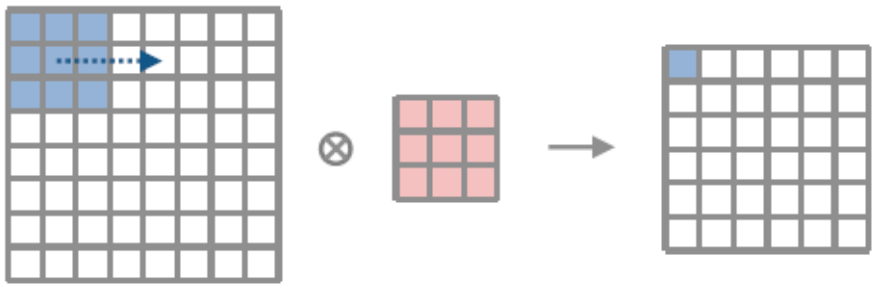
CNNの重要な２つの演算

Convolution Layer（畳み込み層）

→ 画像のフィルタ処理に相当する処理

→ 特徴抽出器としての役割

→ ストライドと呼ばれる間引きを行うこともある

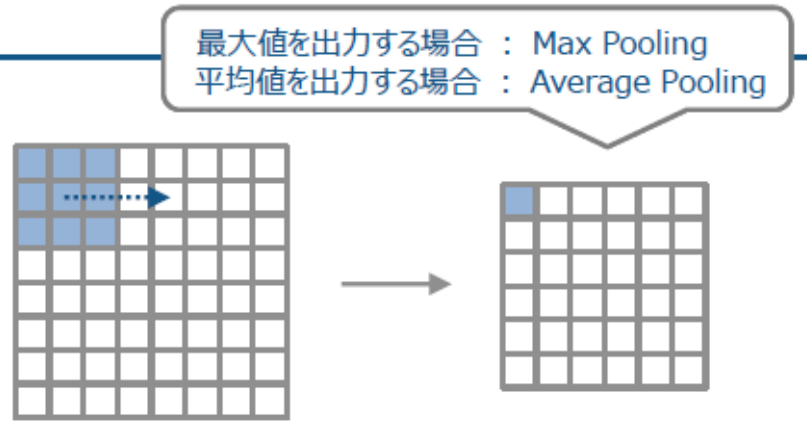


Pooling Layer（プーリング層）

→ 領域内の最大値または平均値を出力

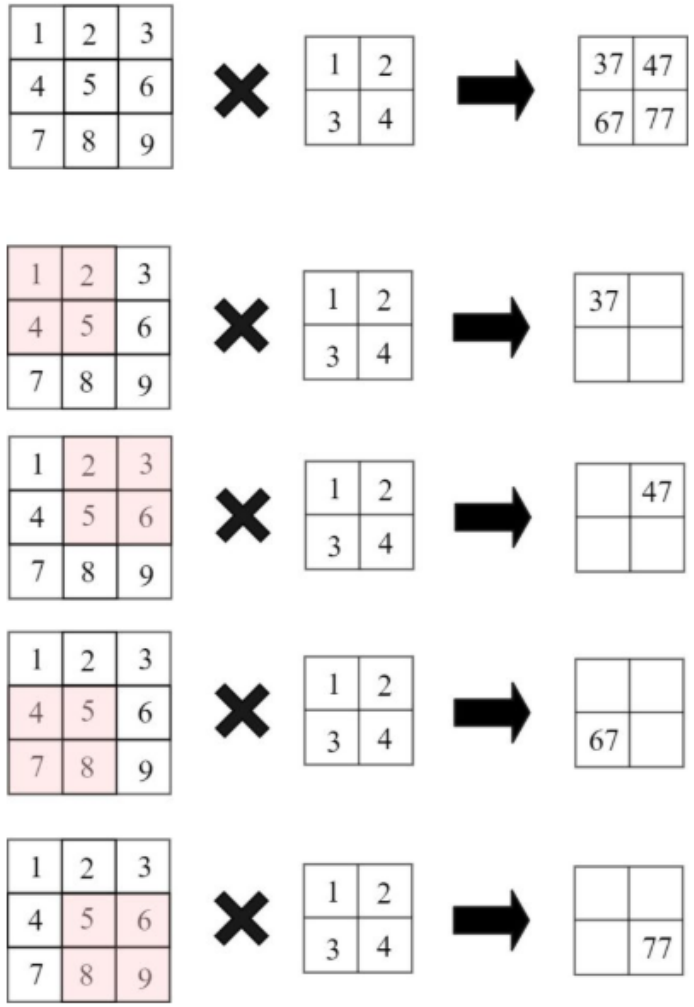
→ 平行移動等に対するロバスト性に関係

→ ストライドと呼ばれる間引きを行うこともある

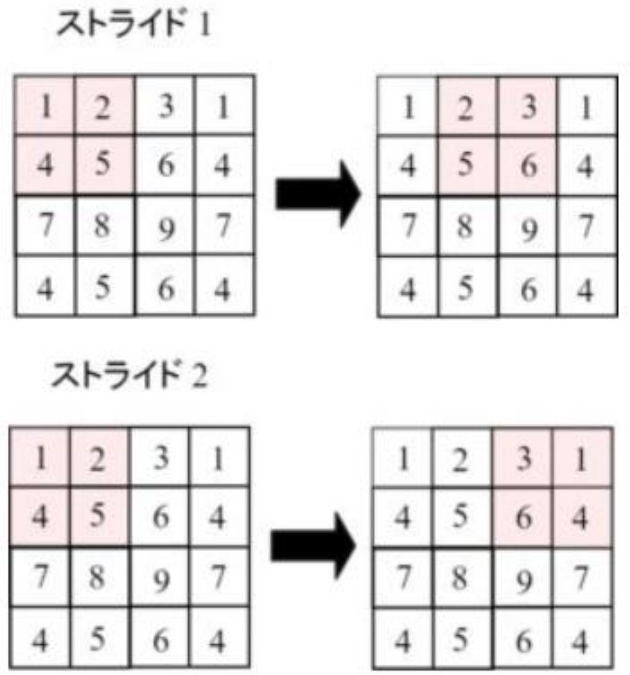


層と層の間を一部のみ連結して、重みを共有すると、ニューラルネットで畳み込みが表現できる

畳み込み演算の具体例

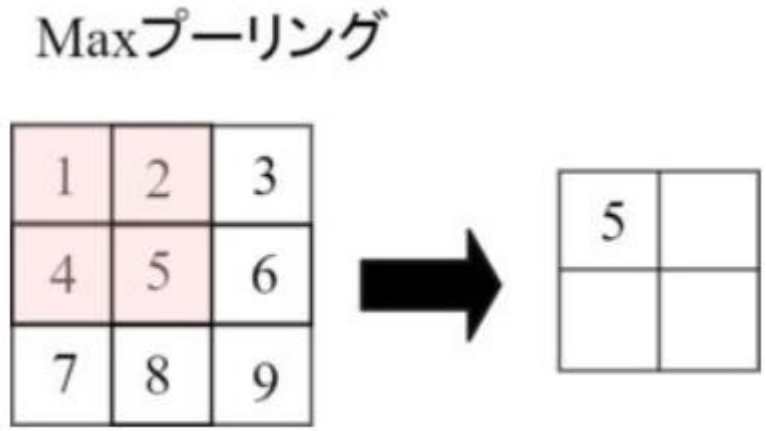


フィルタを入力に対して少しずつ動かし、入力とフィルタの対応している部分を掛け合わせたものの和。



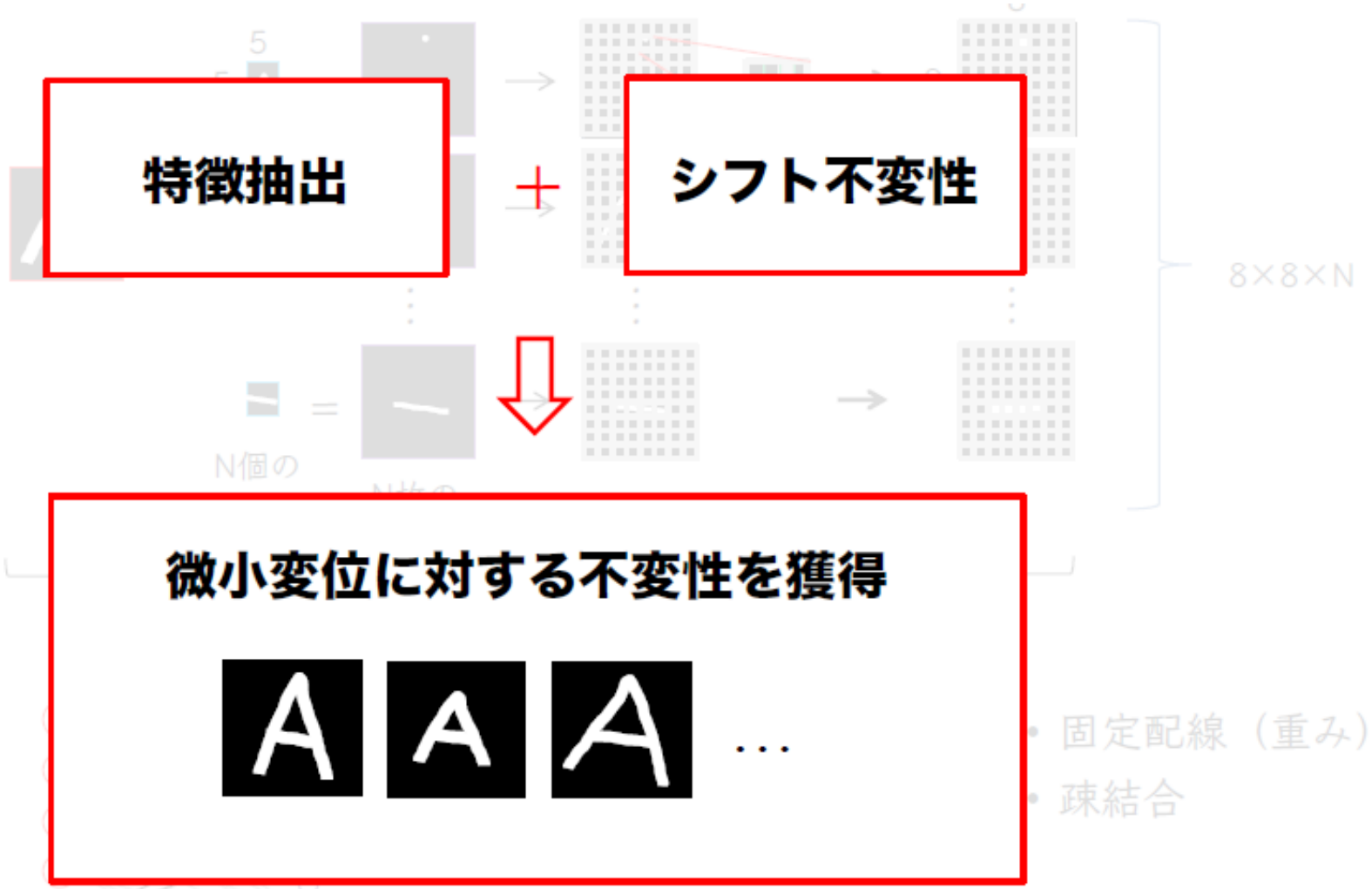
演算時にフィルタがずれる量のことを「ストライド」といい、ストライドを大きくすればするほど演算結果の大きさは小さくなる。

プーリング演算の具体例



- プーリングは指定した範囲の要素をまとめ、入力の大きさを小さくする。
- プーリングは指定範囲何にある特徴的なデータを抜き出す。
  - データの位置のズレやノイズがあったとしてもそれらを吸収する。
- Maxプーリング以外にも、指定範囲内の平均をとるAverageプーリングなどがある。

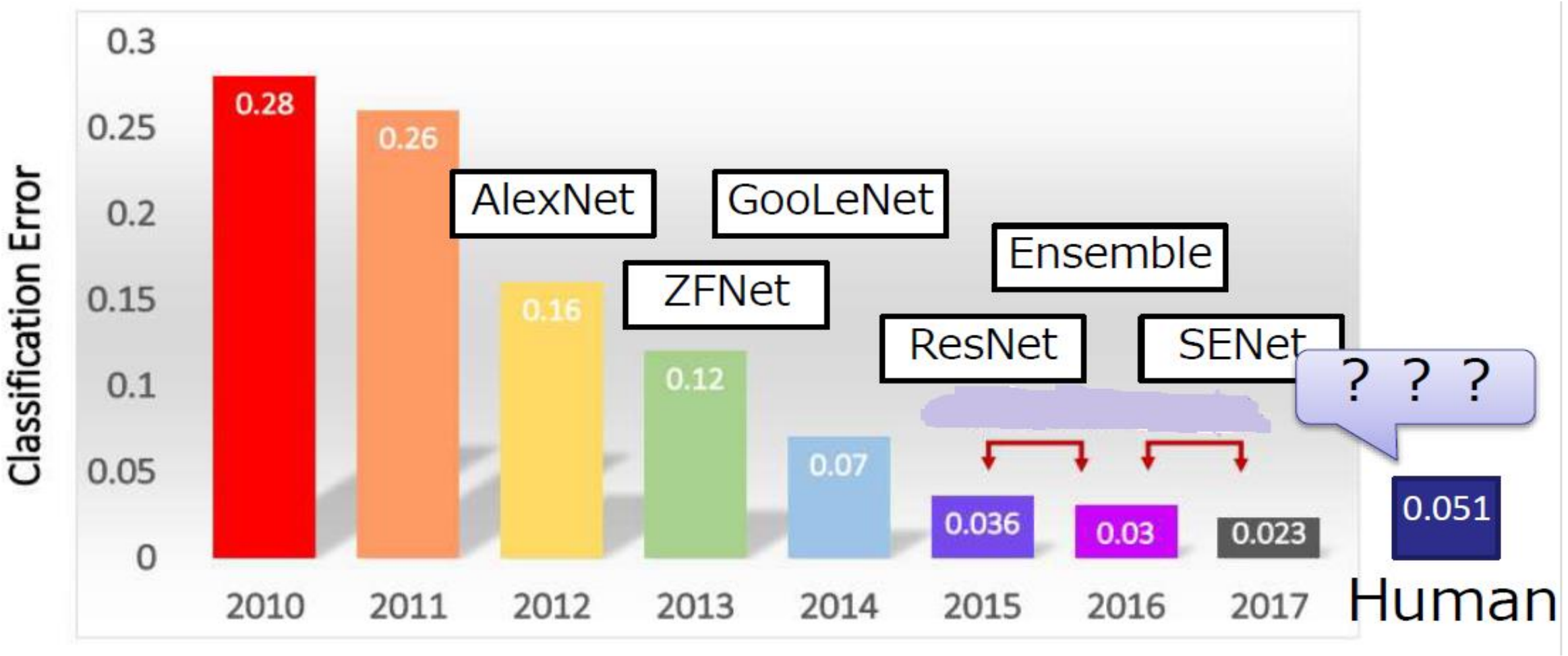
CNNの重要な２つの演算の効果



CNNにおけるその他、特徴的な技術

- Batch Normalization
  - バッチごとに各層の特徴マップを正規化
    - 平均を0、分散を1
    - 大規模なモデルの学習に非常に重要
  - 効果
    - 大きな学習係数が使える➡収束性が向上
    - 初期値にそれほど依存しない
    - その他
- Skip-connection
  - プーリング回数の少ない前半の特徴マップを中間層をスキップし、後半の識別層に接続
  - 前半の特徴マップがもつ低レベルの特徴も後半の識別層に伝播可能
    - エッジや色情報など
- Depthwise convolution
  - 特徴マップのチャネル毎にそれぞれ空間方向の畳込みを行う。

クラス分類タスクのエラー率（上位5つ エラー）の推移



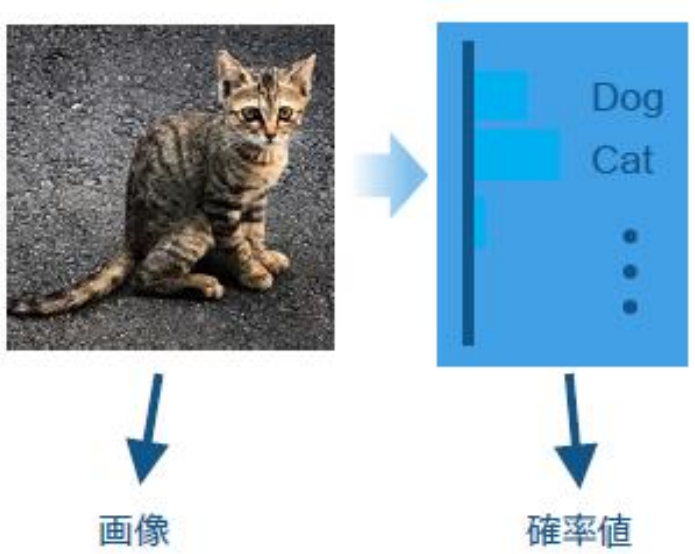
http://image-net.org/challenges/talks_2017/ILSVRC2017_overview.pdf

物体検出とは？

画像認識のタイプ

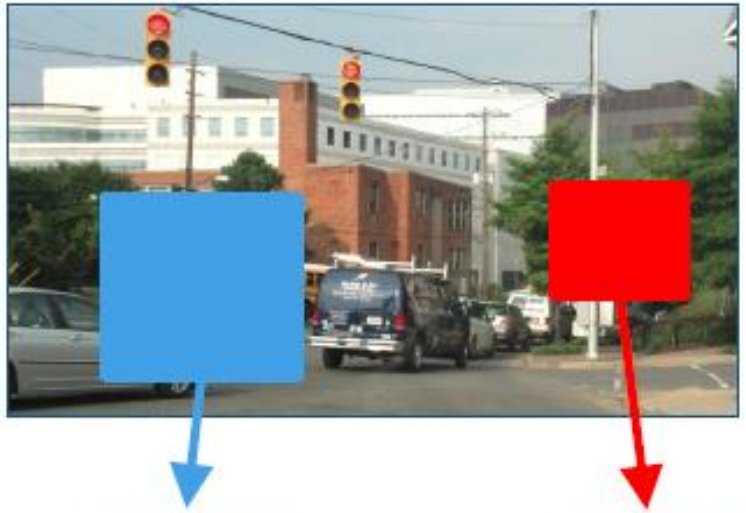
物体認識（画像全体）

CNN（畳み込みニューラルネットワーク）



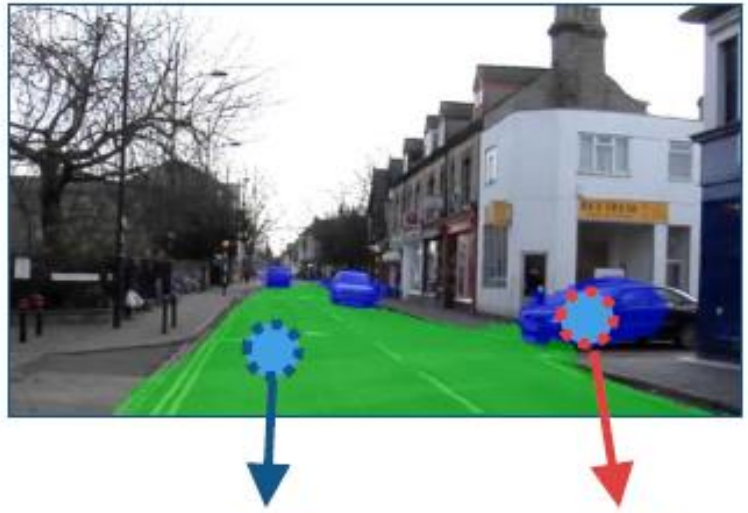
物体の検出と認識

R-CNN、Fast R-CNN、Faster R-CNN
YOLO3、SDD



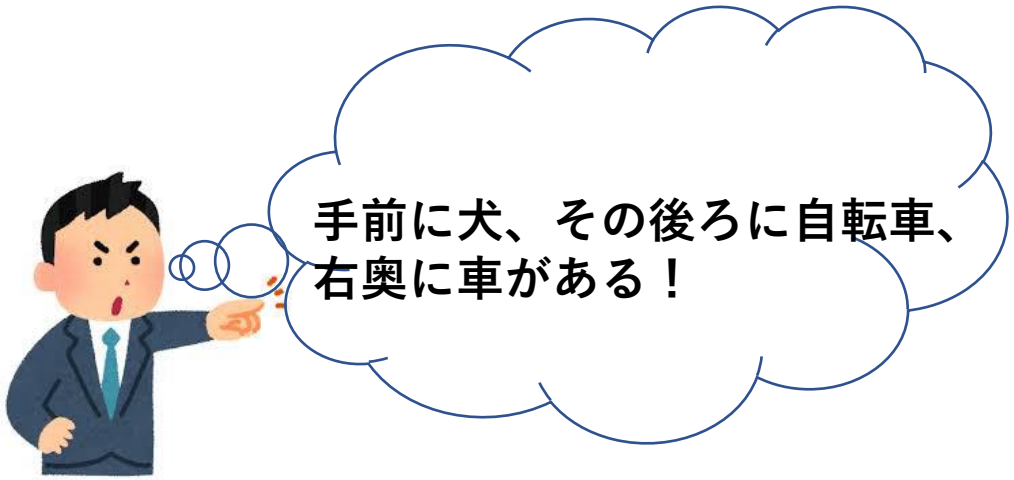
物体認識（ピクセル単位）

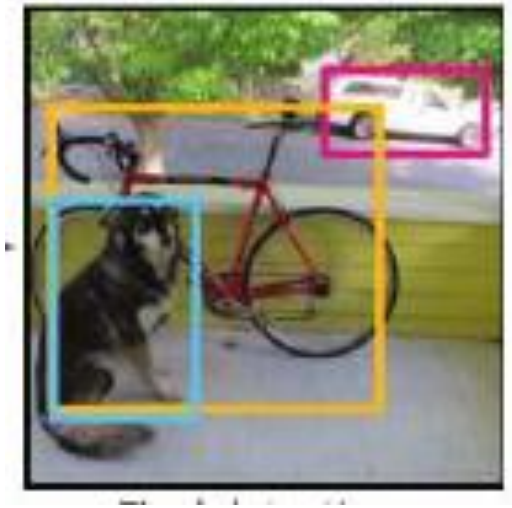
FCN、enet、U-Net、Mask R-CNN



そもそも物体検出とは？

人間は一目見ただけで物体の分類と位置の推定を行っている。





このような仕組みができないか・・・？

そもそも物体検出とは？

- CV(Computer Vision)タスクの一つ
  - 与えられた画像の中から
  - ➡ 物体の位置とカテゴリ（クラス）を当てる

- 物体検出の処理ステージ
  - two-stage detector (Faster R-CNN系、Mask R-CNNなど)
  - one-stage detector (YOLO3、SSD、M2Detなど)

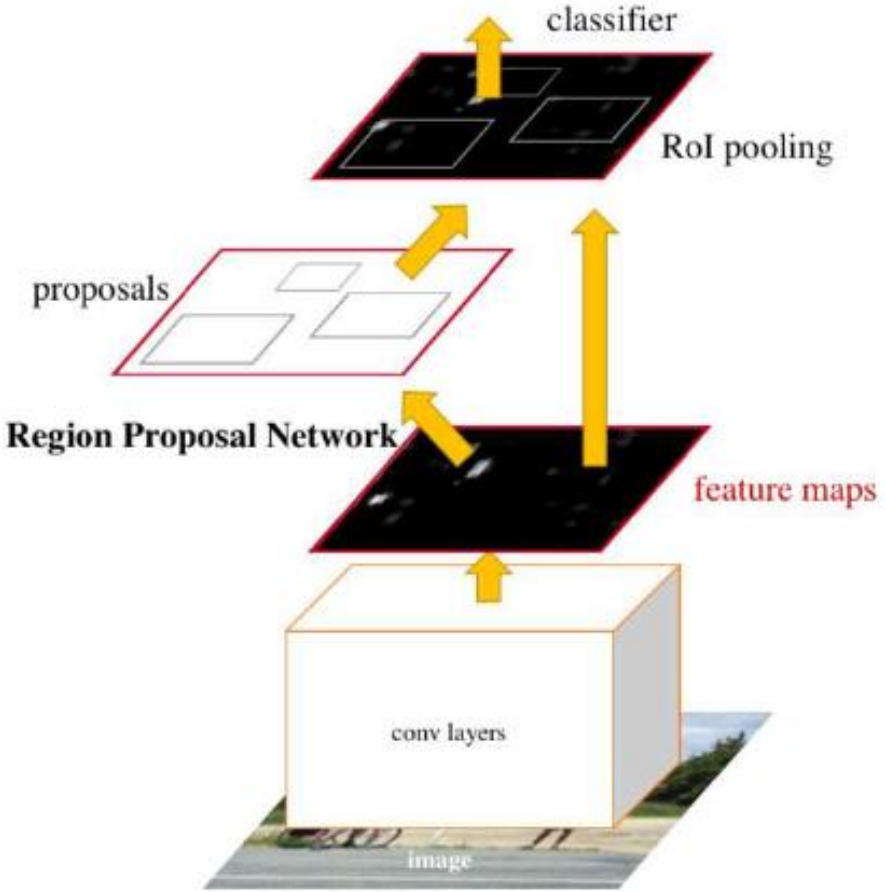
- 基本的な流れ
  1. 画像から物体領域の候補抽出(Region Proposal)
  - ➡ 枠、Bounding Boxとも呼ばれる
  2. 各枠で画像認識
  - ➡ 他クラス分類問題

Two-stage detector

- 以下の2段階のプロセスで物体検出をする手法
  ①候補領域の抽出
  ②クラス、bounding boxの座標の推定

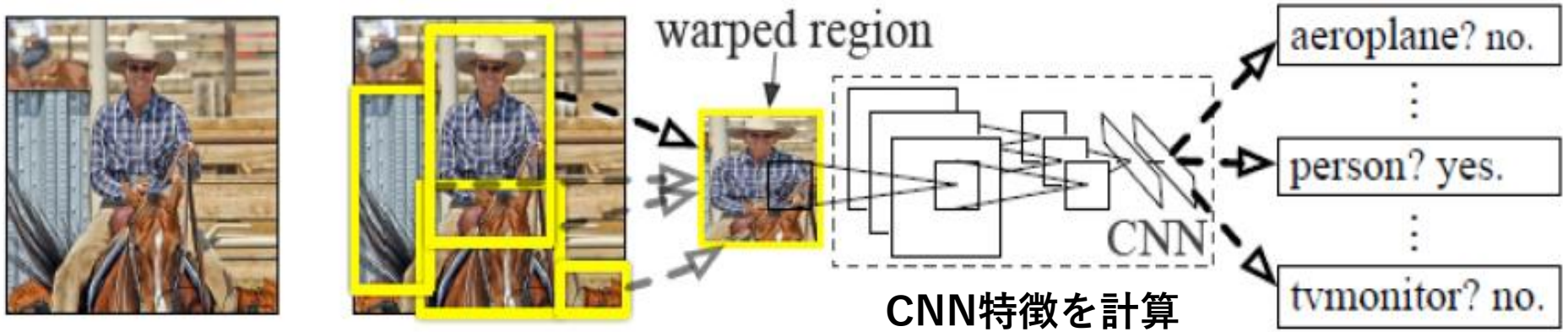
- 代表的なモデルはFaster R-CNN

- 検出精度が高いが、速度がやや遅い



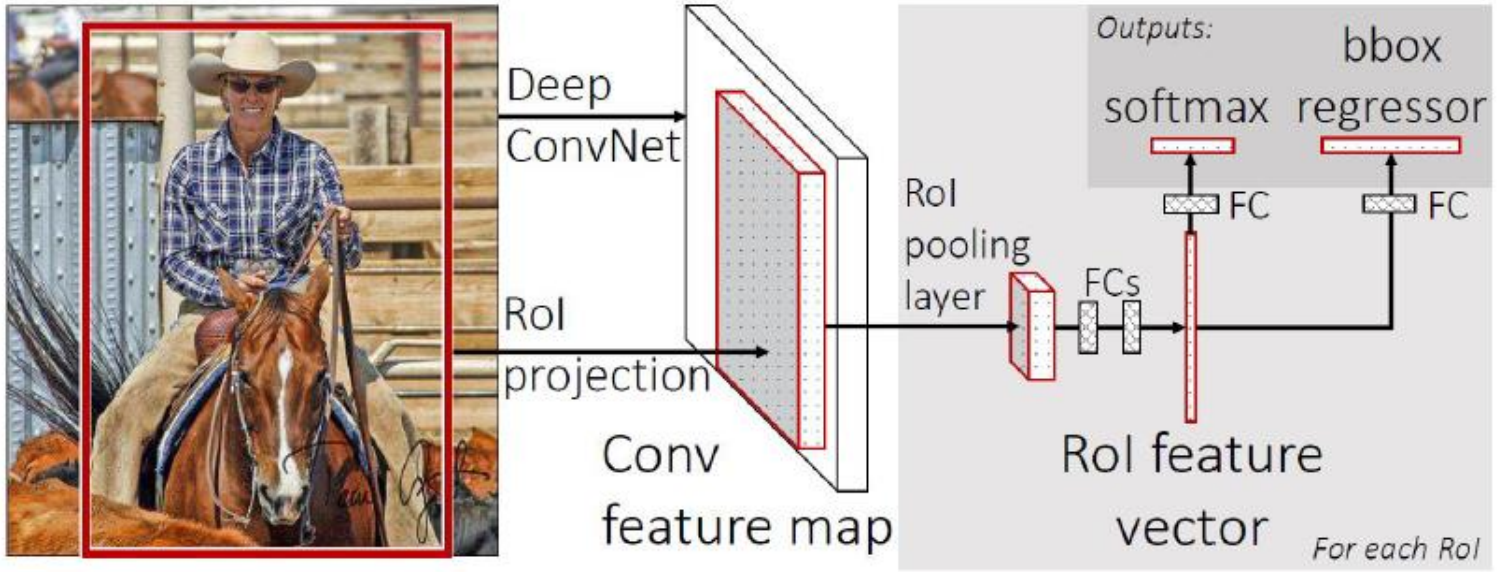
R-CNN(Region with CNN feature)

- 畳み込みニューラルネットワーク（CNN）は計算量が高いので、探索窓(Sliding Window)による検出は更に計算量高い
- Selective Searchという手法を用いて物体候補領域を検出し、候補領域上のみ処理することで計算量削減



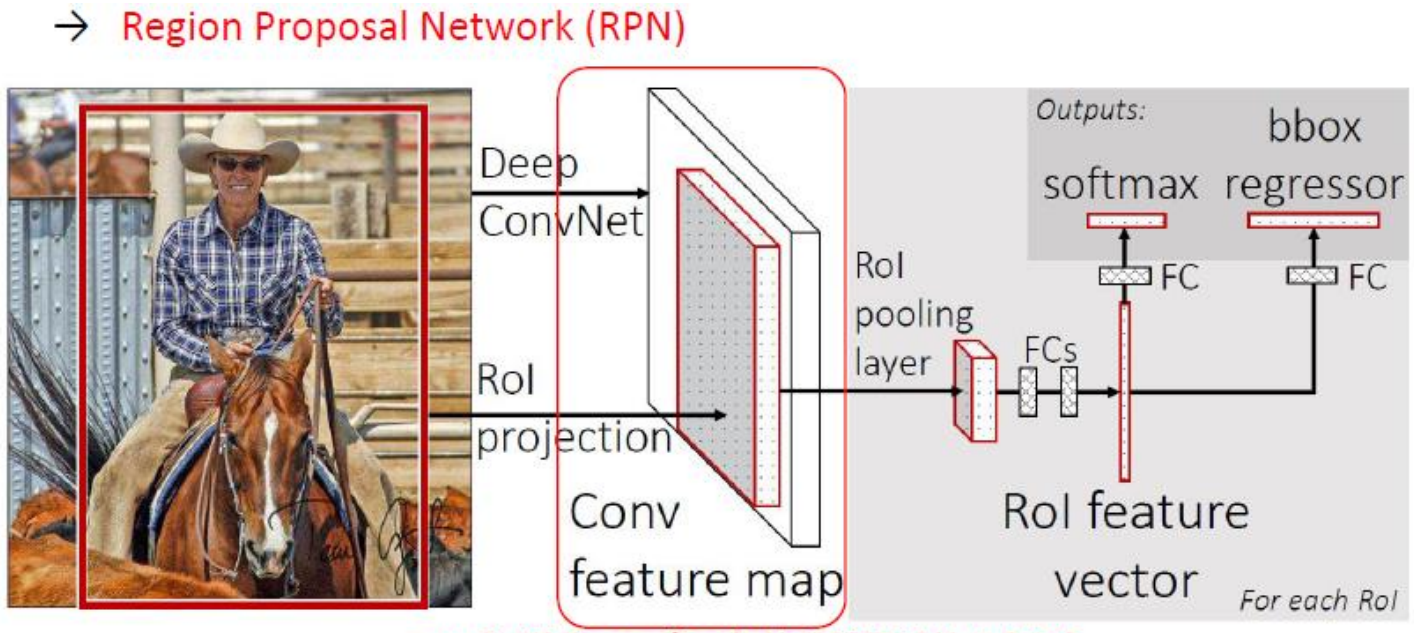
Fast R-CNN

- R-CNNでは物体候補領域を１つ１つCNNで判別していたため遅い（VGGを使用した場合、GPU込みで１枚47
- 画像全体にCNNをかけて特徴マップを生成し、最後のプーリング層を物体候補領域(ROI)に合わせて切り出す
- R-CNNを検出時約213倍高速化



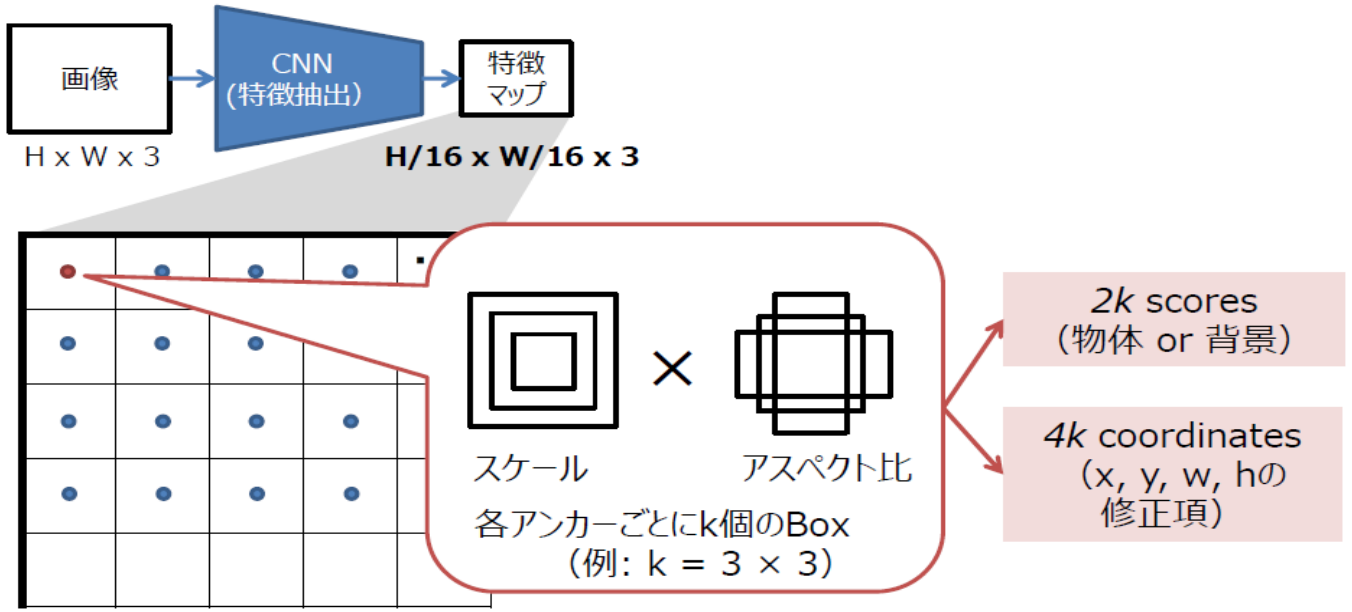
Faster R-CNN

- R-CNNおよびFast R-CNNではSelective Searchを用いて物体候補領域を事前に求めておく必要。
- Fast R-CNNのSelective Search部分をfeature map上で行うことで、余計な処理を省き、高精度化/高速化
  - １枚当たり約200msec



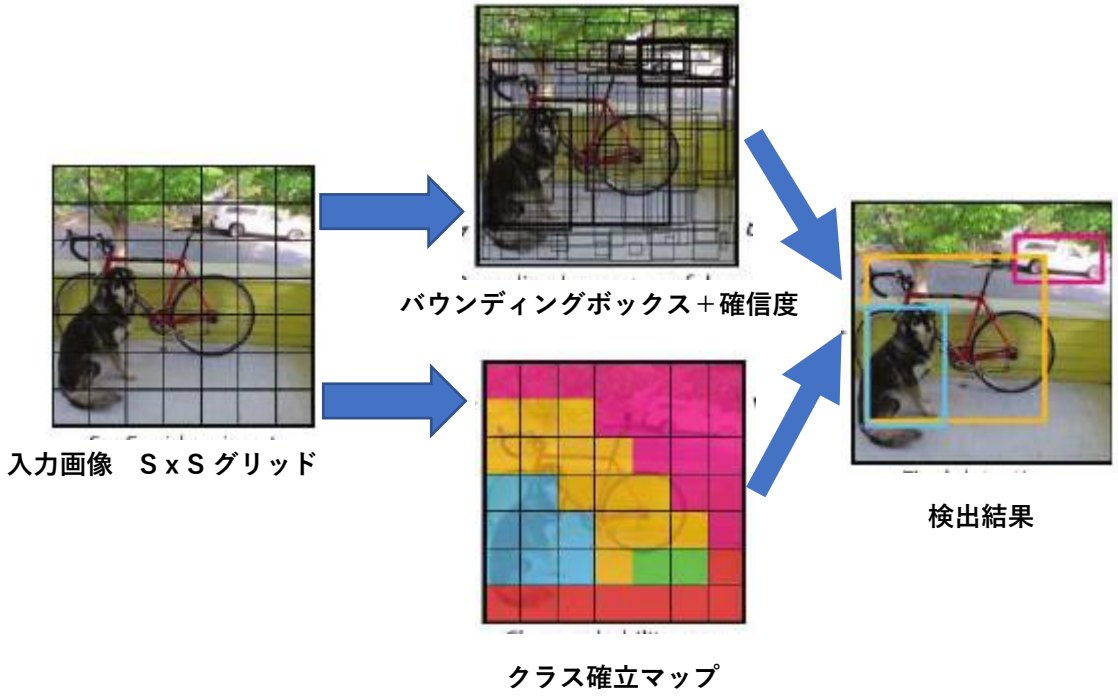
Region Proposal Network

- 特徴マップ上にAnchorを定義
  - 方眼紙に見立てて、各マスの中心のイメージ
- 各Anchor毎にk個のAnchor Boxを定義
  - スケールとアスペクト比の組み合わせ
- 各Anchor Box毎に、物体らしさのスコアと位置・サイズの修正項を予測するように訓練する



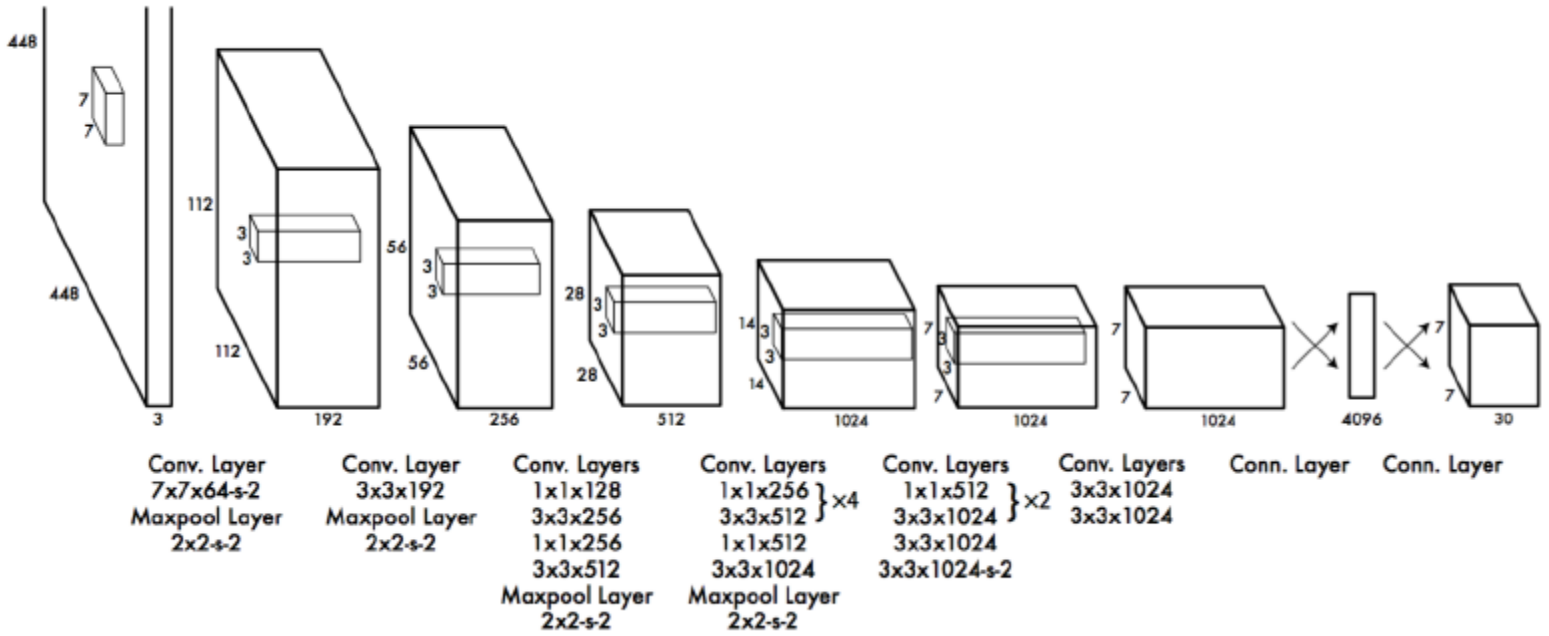
One-stage detectors

- 一段のCNNで直接bounding boxの座標とクラスを推定する
- 代表的なモデルはYOLO3、SSD、M2Det
- two-stage detectorに比べると、検出速度が速く，精度は劣る

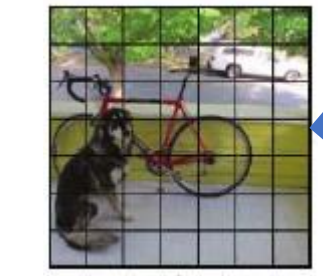


yoloモデルのアーキテクチャー

２４層の畳み込みのあと、2層の全結合



yoloの推論の流れ



入力画像　S x S グリッド

B(x,y,h,w,F)

バウンディングボックス＋確信度

・7x7個のセルがそれぞれ
2個の矩形候補領域を出力
・確信度、Fを太さで表現

閾値で確信度、Fの
高いものだけ採用

C=Pr(Class|Object)

クラス確立マップ

検出結果

対応セルにおける確立の高
いクラスと照らし合わせて、
矩形領域のクラスを決定

yolo3の機能

- single shotの物体検出手法の一つ
  - 似たような手法にSSD、M2Detがある
- 53層のNeural Networkモデル
  - v2と比較して検出速度は若干低下したが、検出精度は向上
  - ResNet構造とFeature Pyramid Network構造
  - Max poolingは廃止
- 3つの異なるスケールでボックスを予測
  - 3つの異なるスケールから特徴量を抽出し、Feature mapを作成
- アップサンプリングの使用
  - 直前の2つのレイヤーからFeature mapを取得し、2倍にアップサンプリング
  - Object Boxの意味のある情報と細かい情報の取得に有効
- 分類にはロジスティック回帰を使用

SSD: Single Shot Multibox Detector

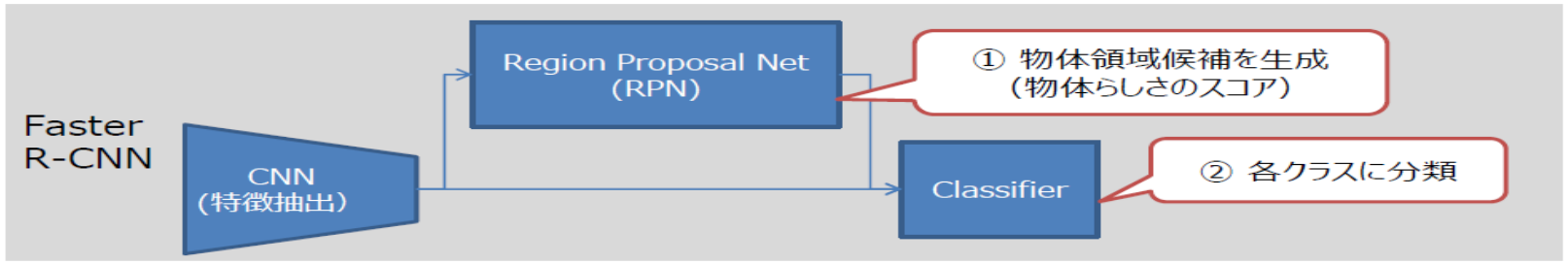
- Faster RCNNよりも高速で精度も良いモデル
  - 入力画像サイズの小さいモデル（精度はそこそこ）では58FPSを達成
- Fasterにおいて

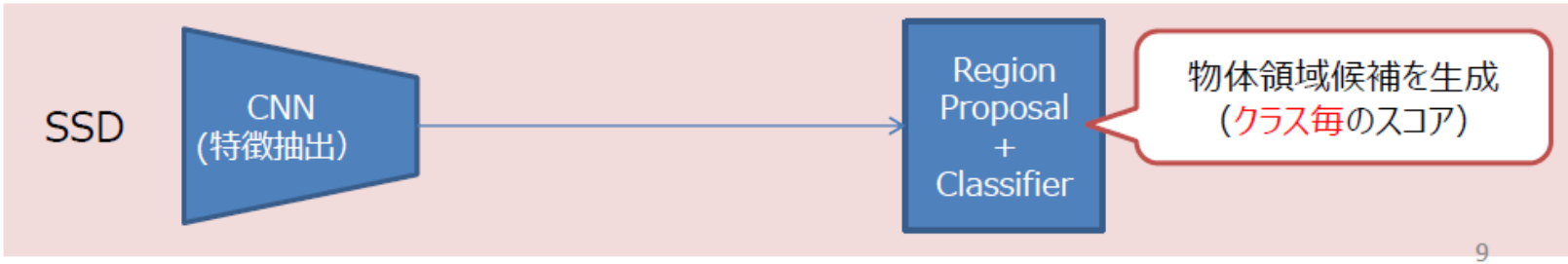
  ①領域候補生成、

  ②各領域の特徴ベクトルを切り出して分類、

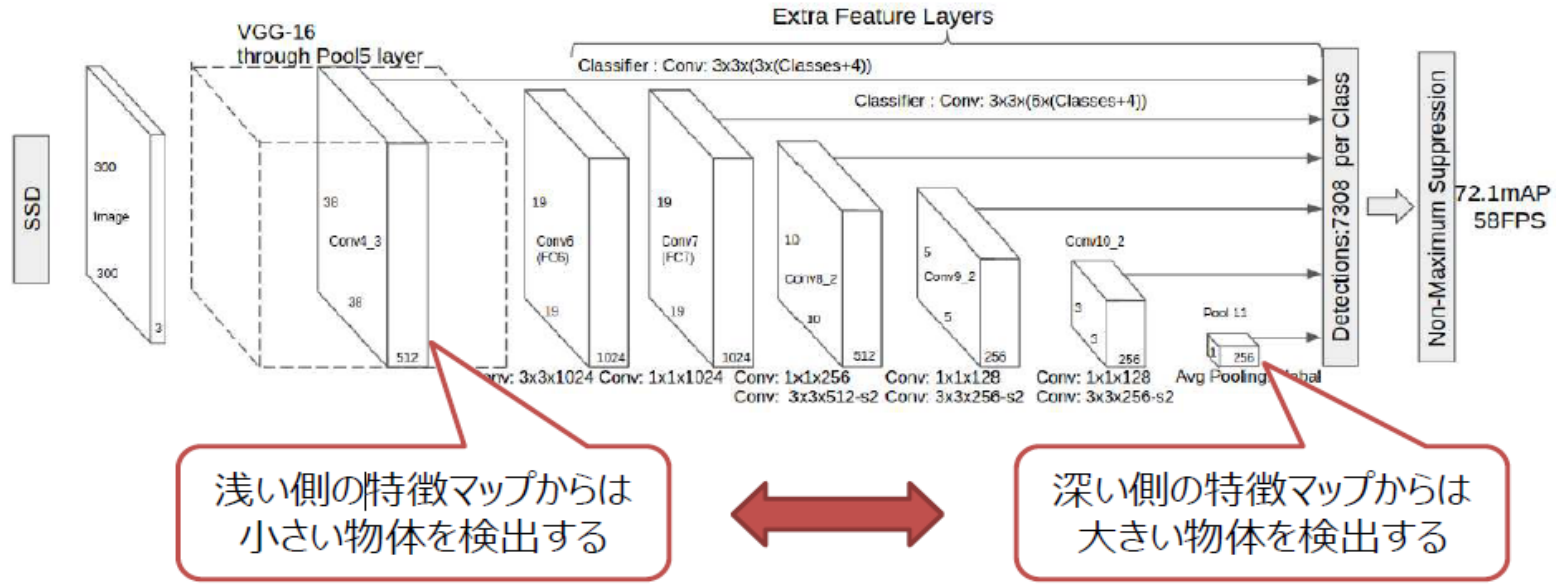
  と2段階で行っていた処理を一気に行う。
- 深さの異なる複数の特徴マップを使い、
  - 浅い側は小さい物体、
  - 深い側は大きい物体を検出。
  - 深さにより、デフォルトのBoxサイズを変えている

SSD: Single Shot Multibox Detector



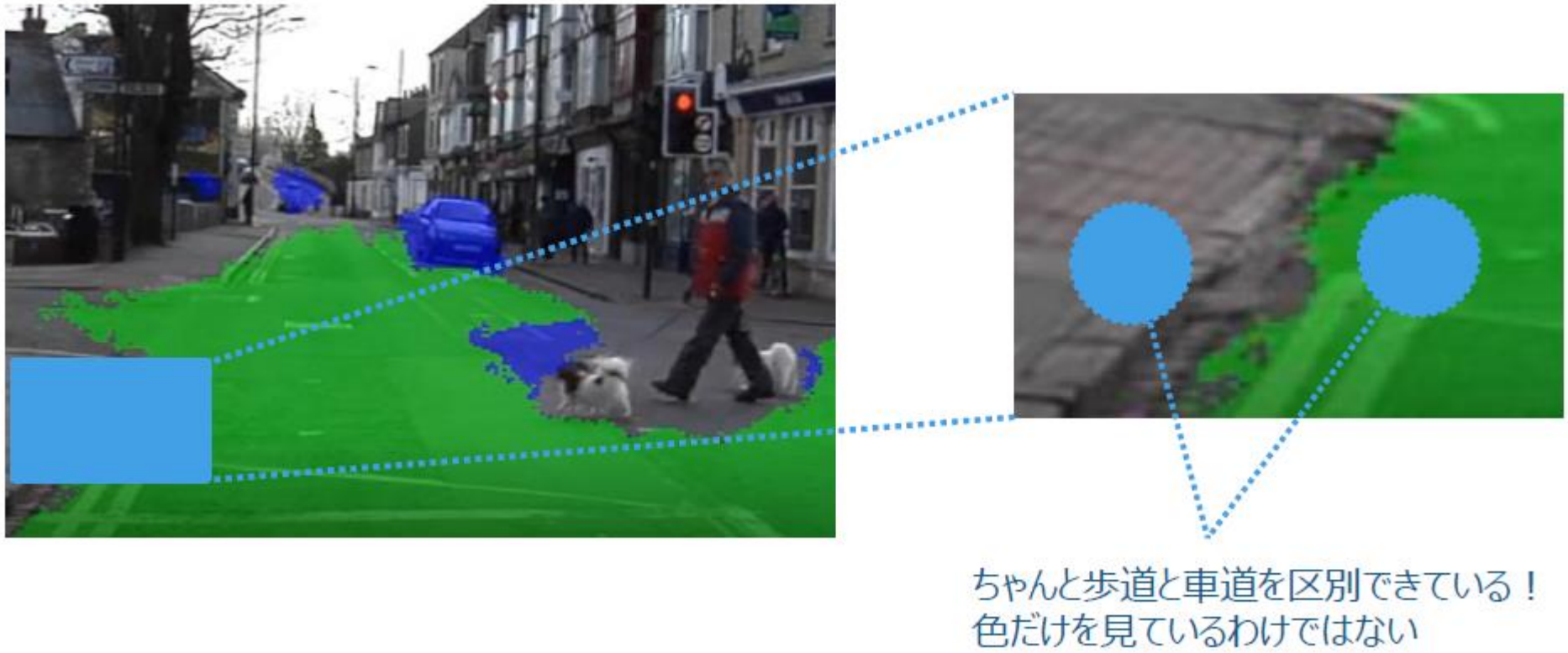


9



セグメンテーションとは？

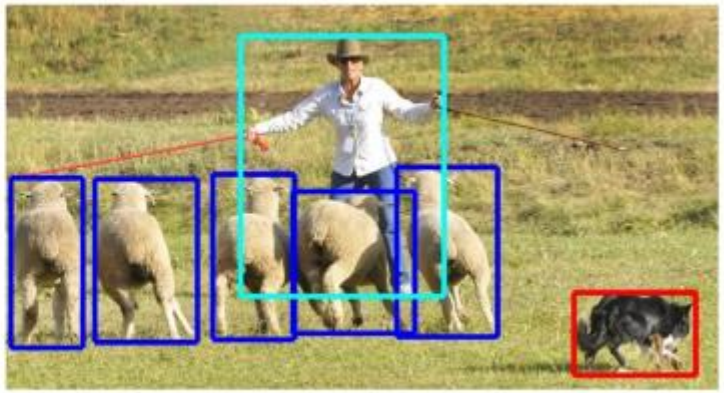
畳み込みニューラルネットによるセグメンテーション



各ピクセルをその意味（周辺のピクセルの情報）に基づいて、カテゴリ分類する手法

シーンの認識の種類

画像の分類



オブジェクトの検出

セマンティックセグメンテーション

インスタンスセグメンテーション

セグメンテーションの応用分野

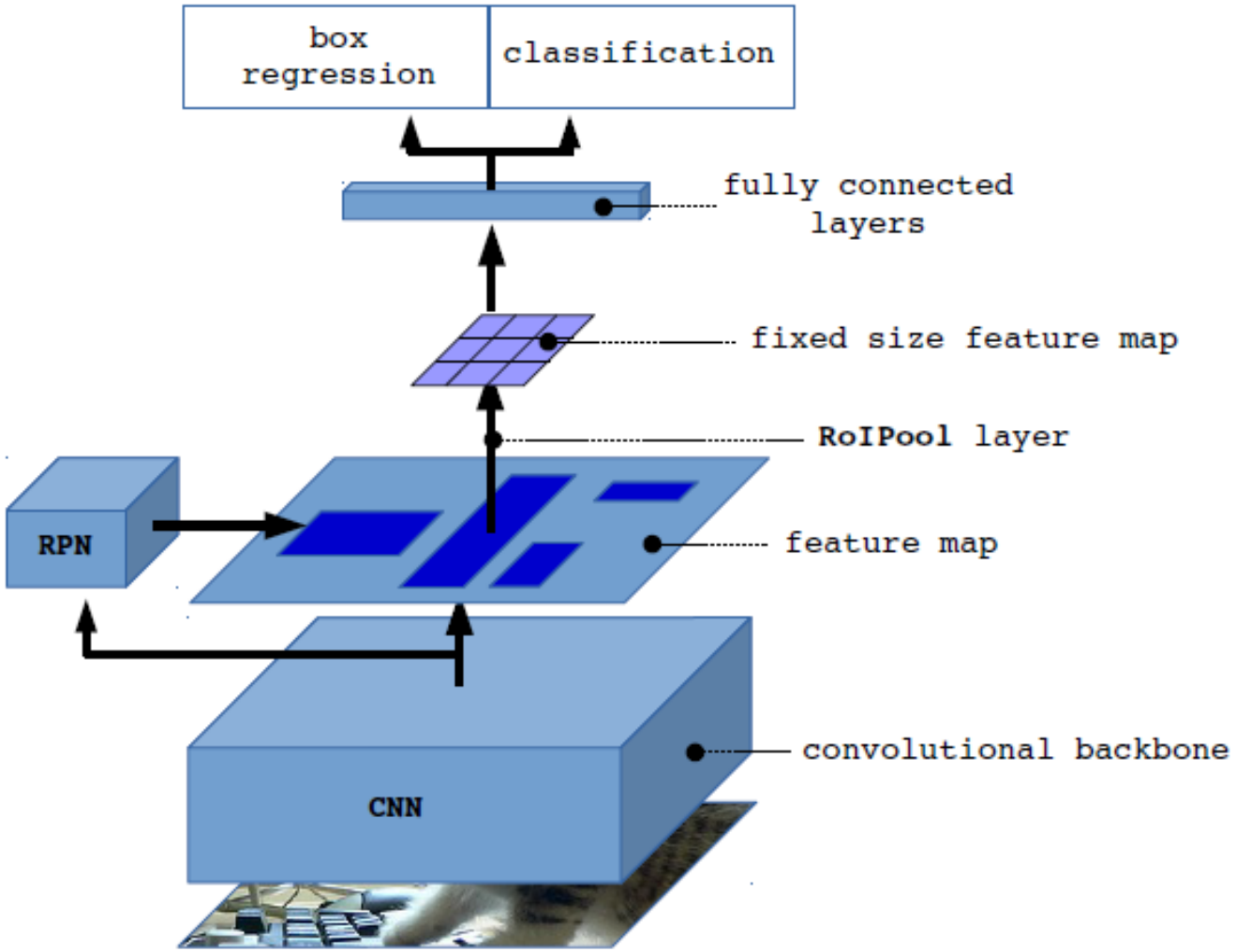
- 医療：CT、MRIなどから腫瘍箇所の形状で指示
- ファッション：着ている服装の識別と置き換え
- 検査／検品：故障箇所や液漏れなど箇所を明確に分類
- 自動運転：自律走行時の各物体を形状で明確に指示

インスタンスセグメンテーション
Mask R-CNN

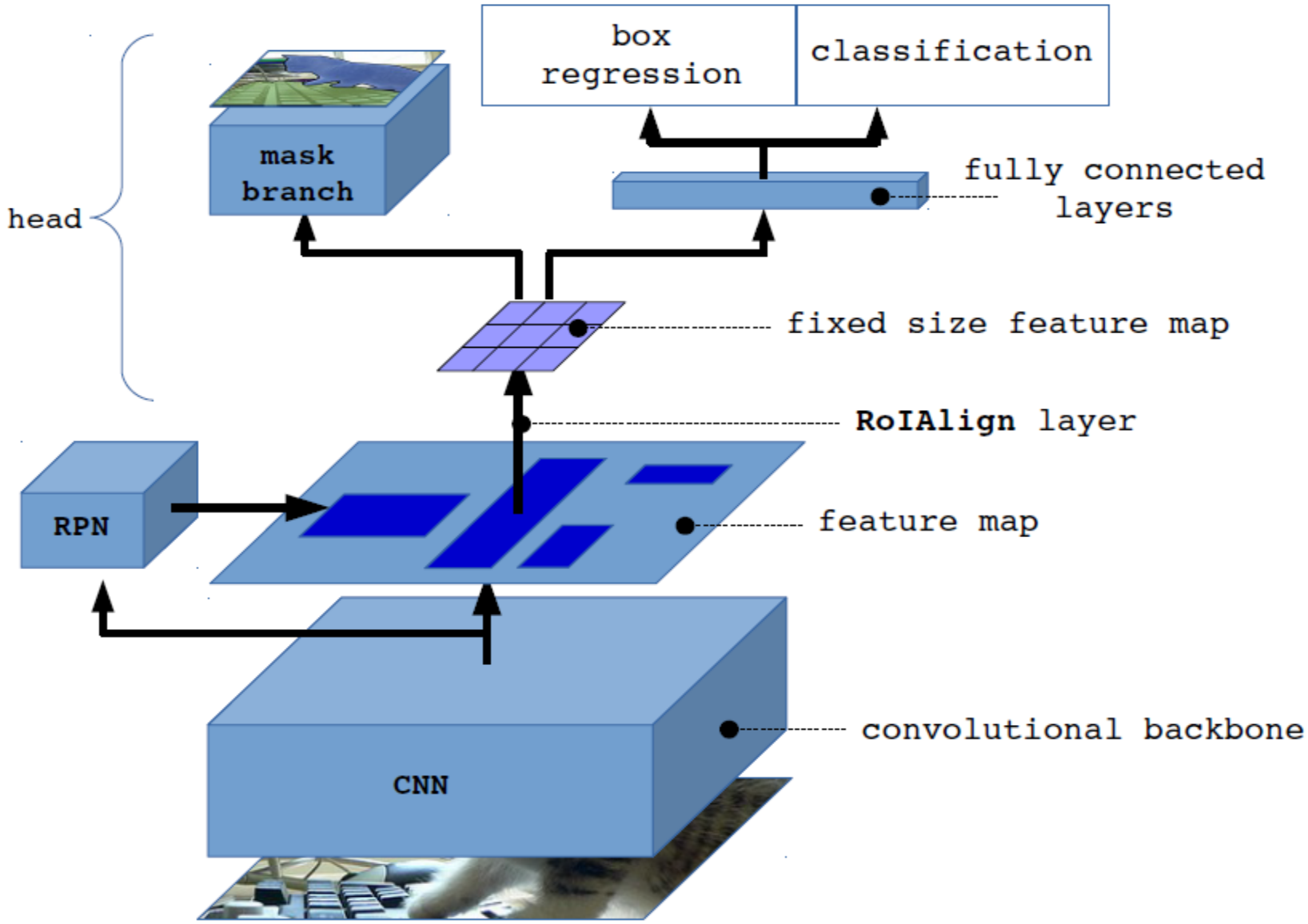
Mask R-CNNの概要

- 最新の複数のアルゴリズムによる複数の物体を認識
  - オブジェクト検出
  - オブジェクトの分類
  - インスタンスセグメンテーション
- 結果
  - バウンディングボックスの検出とキーポイント検出を行う
- 洗練されたモジュール化、学習が容易
- 拡張性
  - マイナーの変更にポーズ推定(姿勢認識)に拡張

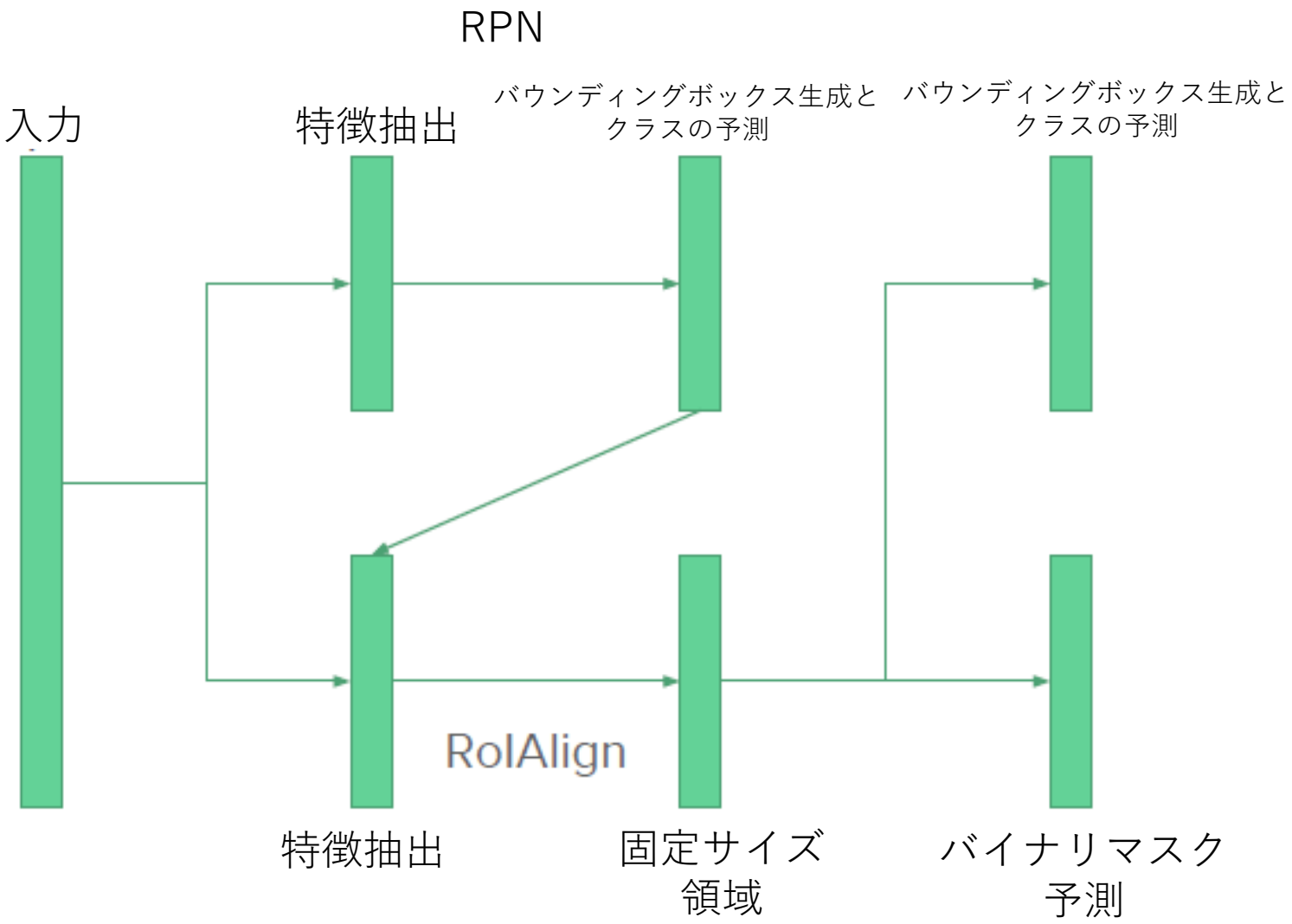
バックグラウンド: Faster R-CNN



Mask R-CNNの構成



Mask R-CNNアーキテクチャー

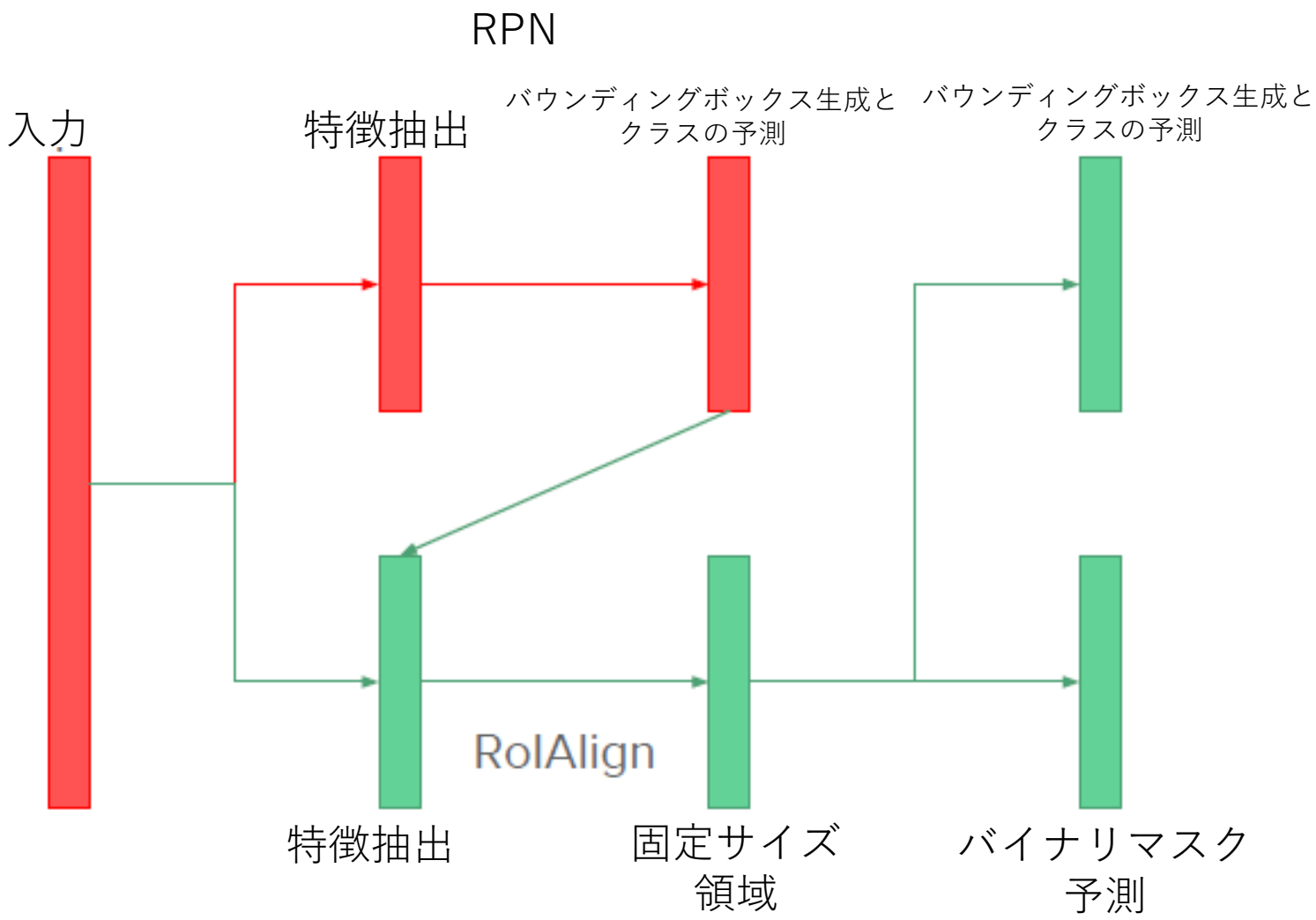


Mask R-CNNアーキテクチャー

- ステージ１
  - RPN (物体候補領域)の生成

- ステージ２
  - 物体のバウンディングボックスの生成
  - 物体のクラス予測
  - バイナリマスク生成

物体候補領域 (RPN)の生成

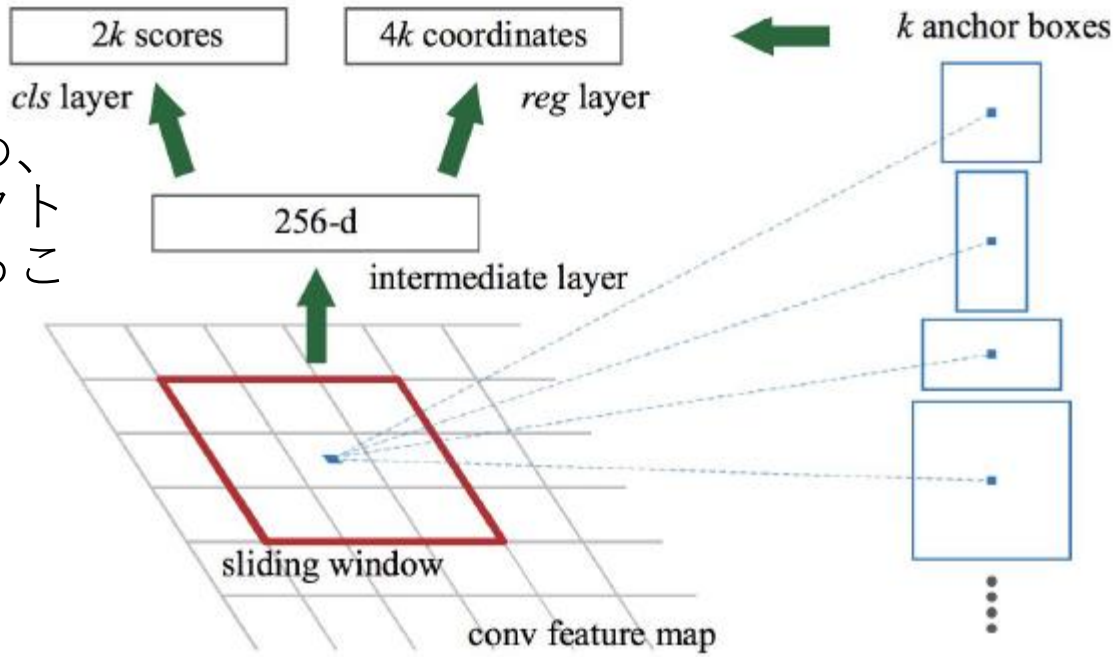


物体候補領域 (RPN)の生成

- 特徴抽出
  - 入力画像から高い特徴を抽出する
  - MxNxC
    - MとNは、画像のサイズに関連する
    - Cは使用されるカーネル数である
    - MとNは奇数である
- 領域候補
  - 特徴抽出器の最後の層は、3×3のスライディングウィンドウを使って画像全体を移動
  - 移動中に、kアンカーボックスを試す
    - アンカーボックスは異なる比率と面積を持つ
    - バウンディングボックスの生成サイズは4kで、クラス予測のサイズは2kになる

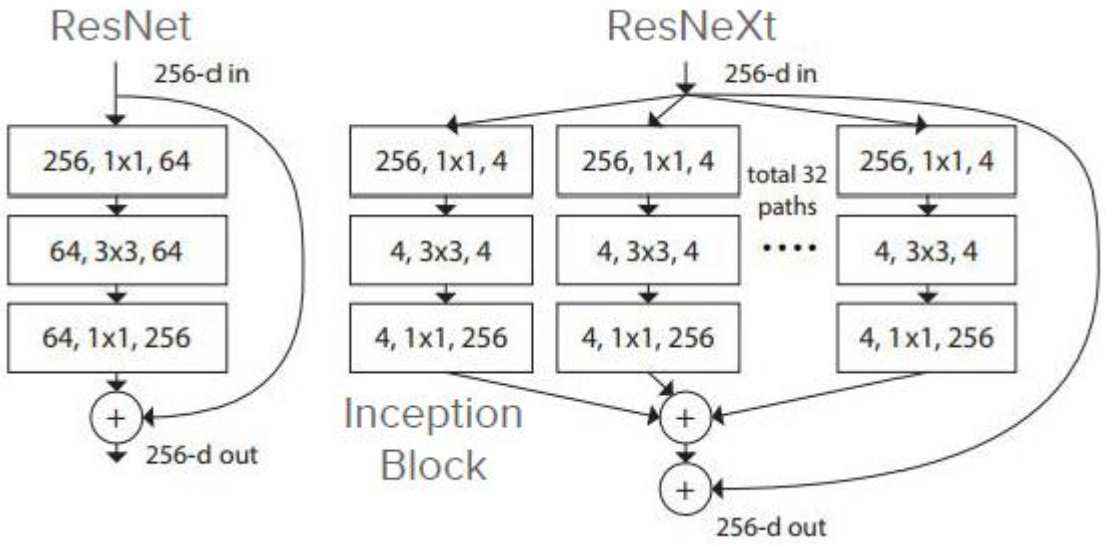
物体候補領域 (RPN)の生成

- 3×3と奇数×Mの理由
  - 中心からの情報を見逃したくない
- なぜK個のアンカーボックスか？
  - オブジェクトは様々な形状にすることができるため、様々なアンカー設定を使用することで、オブジェクトに対するバウンディングボックスの一致度を高めることができる。
- 2kのスコアと4kの座標が必要な理由
  - 2は背景かどうかを表します。
  - 4は（x、y、w、h）を表します
  - アンカーごとに、スコアとオフセットを予測。



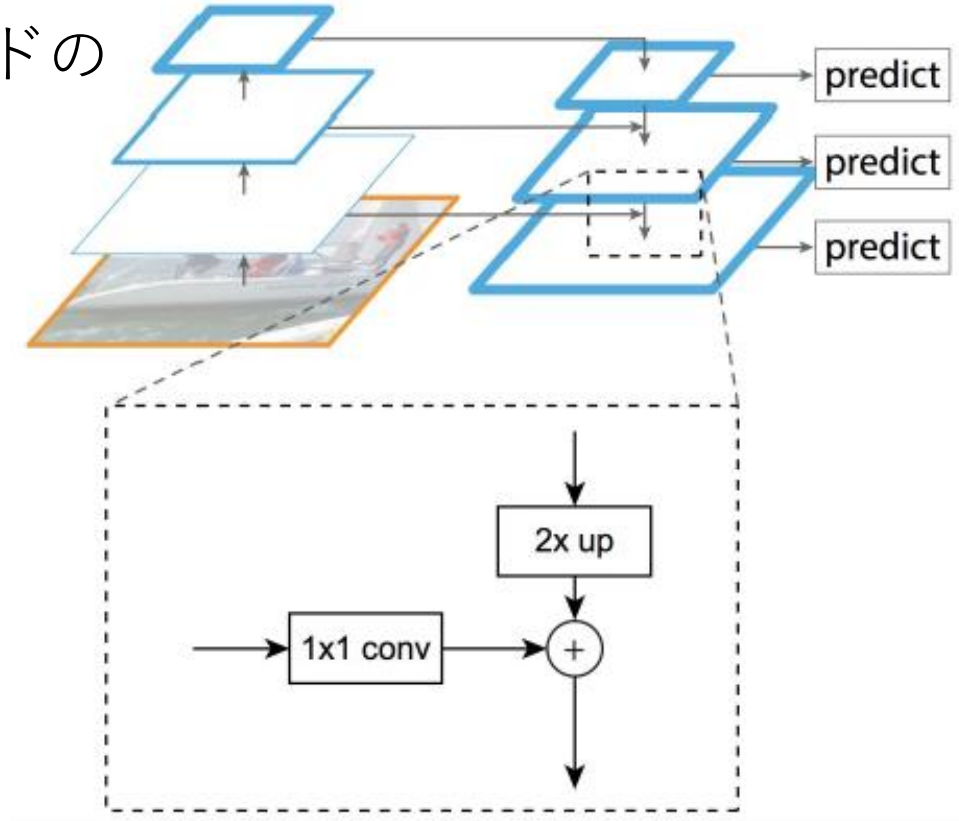
特徴抽出

- ステージIIに進む前に、特徴抽出ついて説明する。
- Mask R-CNNは、2つのネットワーク設定が可能（バックボーン）
  - ResNet
  - ResNet とResNeXt の特徴ピラミッドネットワーク



特徴ピラミッドネットワーク

- なぜピラミッドなのか？
  - スケールが異なるオブジェクトは、ピラミッドのレベルの1つに分類される。
- 接続をスキップする理由
  - すべてのレベルは特徴的意味が強い
- 解決策
  - 1x1 convolution
- なぜ各スケールで予測が行われるのか？
  - マルチスケールでオブジェクトを処理する

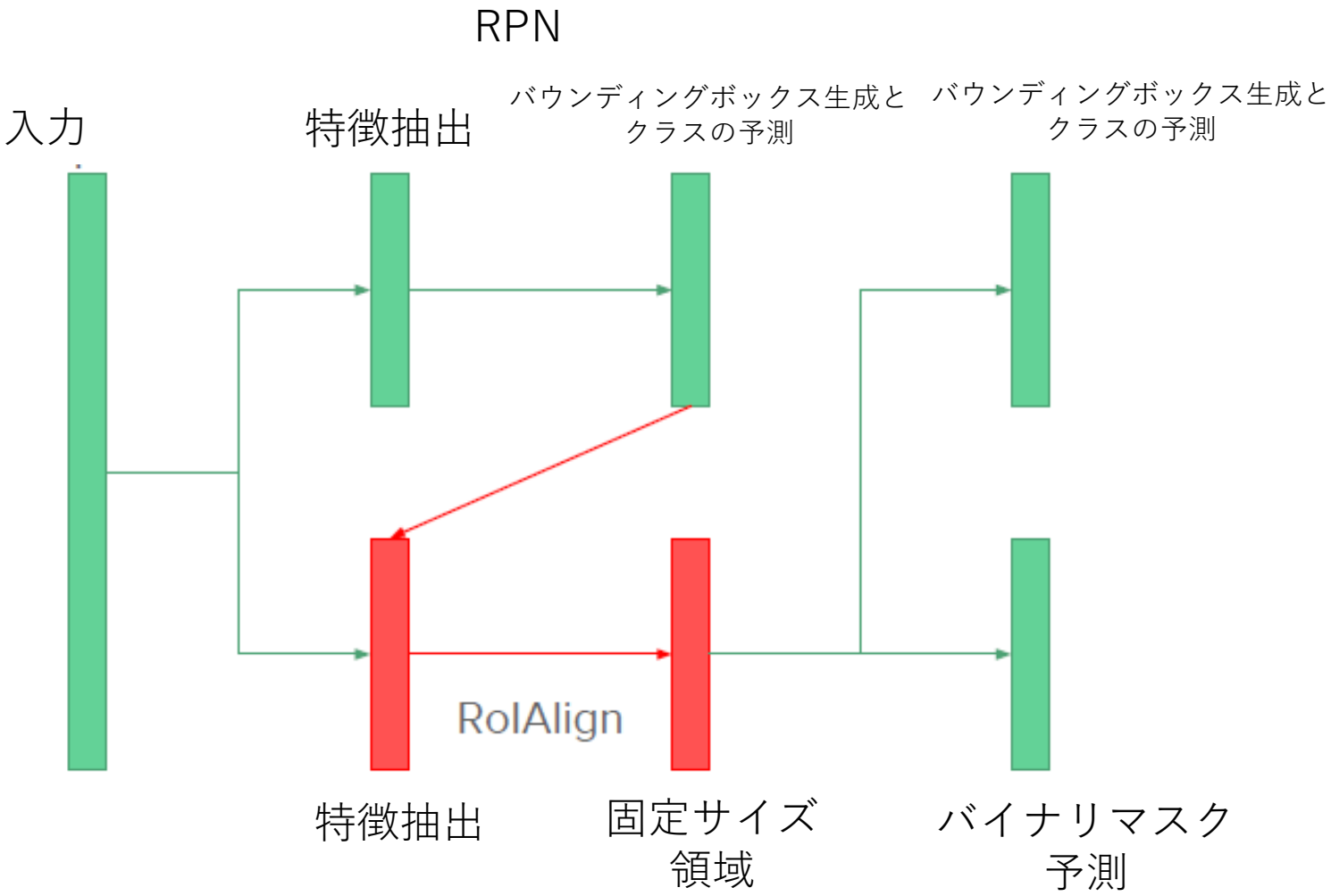


Mask R-CNNアーキテクチャー

- ステージ１
  - RPN (物体候補領域)の生成

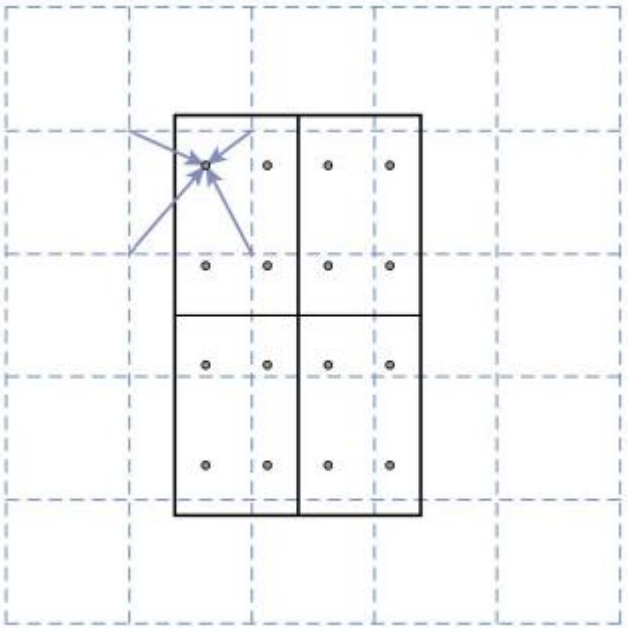
- ステージ２
  - 物体のバウンディングボックスの生成
  - 物体のクラス予測
  - バイナリマスク生成

RoIAlign


RPN
入力
特徴抽出
バウンディングボックス生成と
クラスの予測
バウンディングボックス生成と
クラスの予測
RoIAlign
特徴抽出
固定サイズ
領域
バイナリマスク
予測

RoIAlign

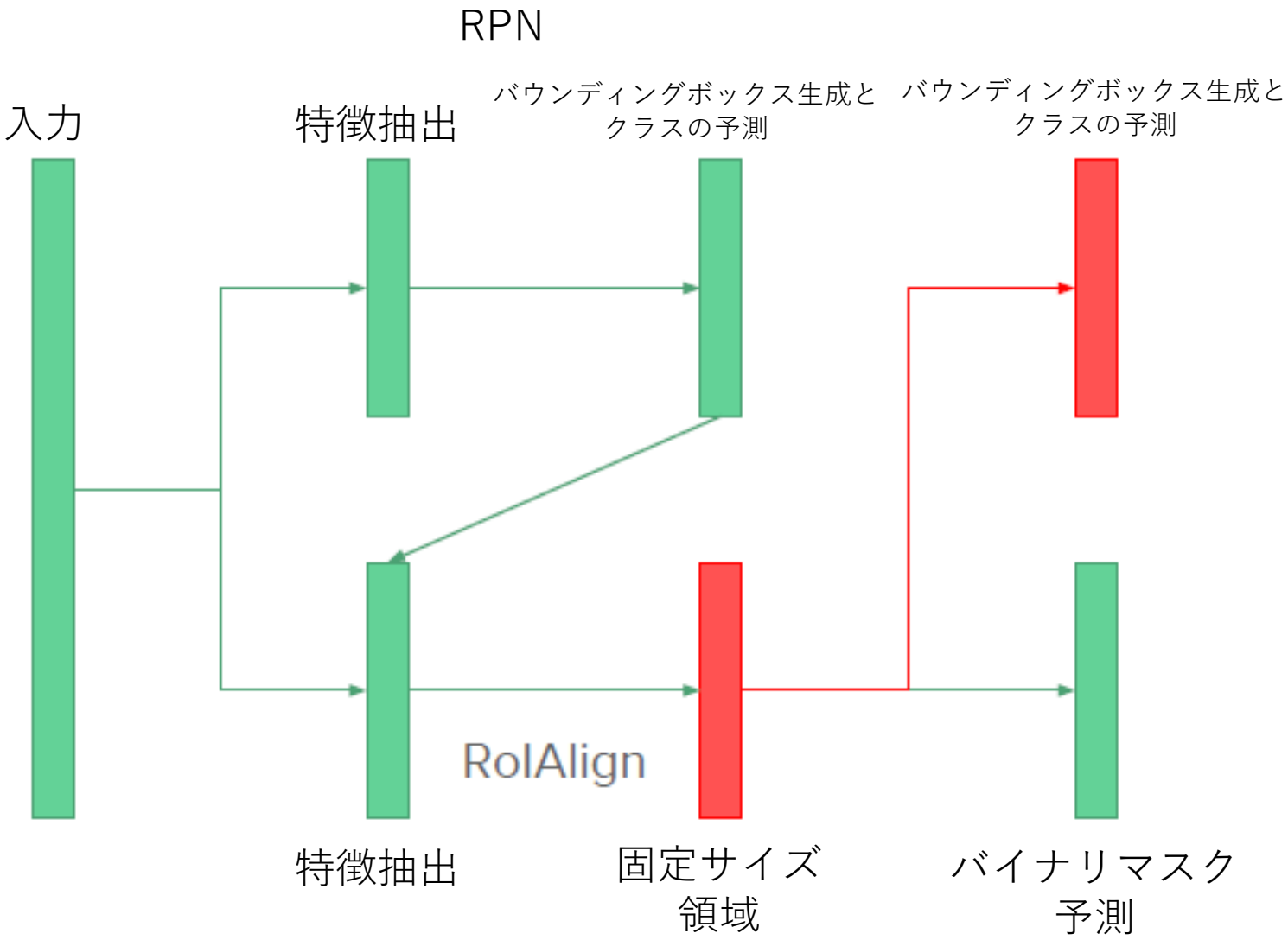
- ステージ 2 の最初のステップ
- RoIAlignを使う理由
  - 同じサブネットワークを使用してクラス、マスク、およびバウンディングボックスを予測できるように、特徴マップのサイズを同じにする。
  - 変換の差異に焦点を当てる - オブジェクトの位置が重要
  - 位置合わせ不良の原因となる量子化（丸め）を避ける
- RoIAlign
  - 1ビン内のサンプル点にバイリニア法(双一次補間)を使用する
    - 右の画像では、ビンごとに4点がサンプリングされている
  - 次に、最大プールを実行してそのビンを表すポイントを選択する。



RoIAlign

- RoIAlignはStage IIの特徴抽出で実行さる
- この特徴抽出はステージ１のものとどのように異なるか？
  - それらはまったく同じ構造を持っている
  - それらは学習時間を減らすために重みを共有することができる
  - 重みを共有するモデルと共有しないモデルの間でパフォーマンスの差。
    - 特徴を共有すると、わずかな差で精度が向上します。
    - 特徴を共有によりテスト時間も短縮

バウンディングボックスの生成とクラスの予測



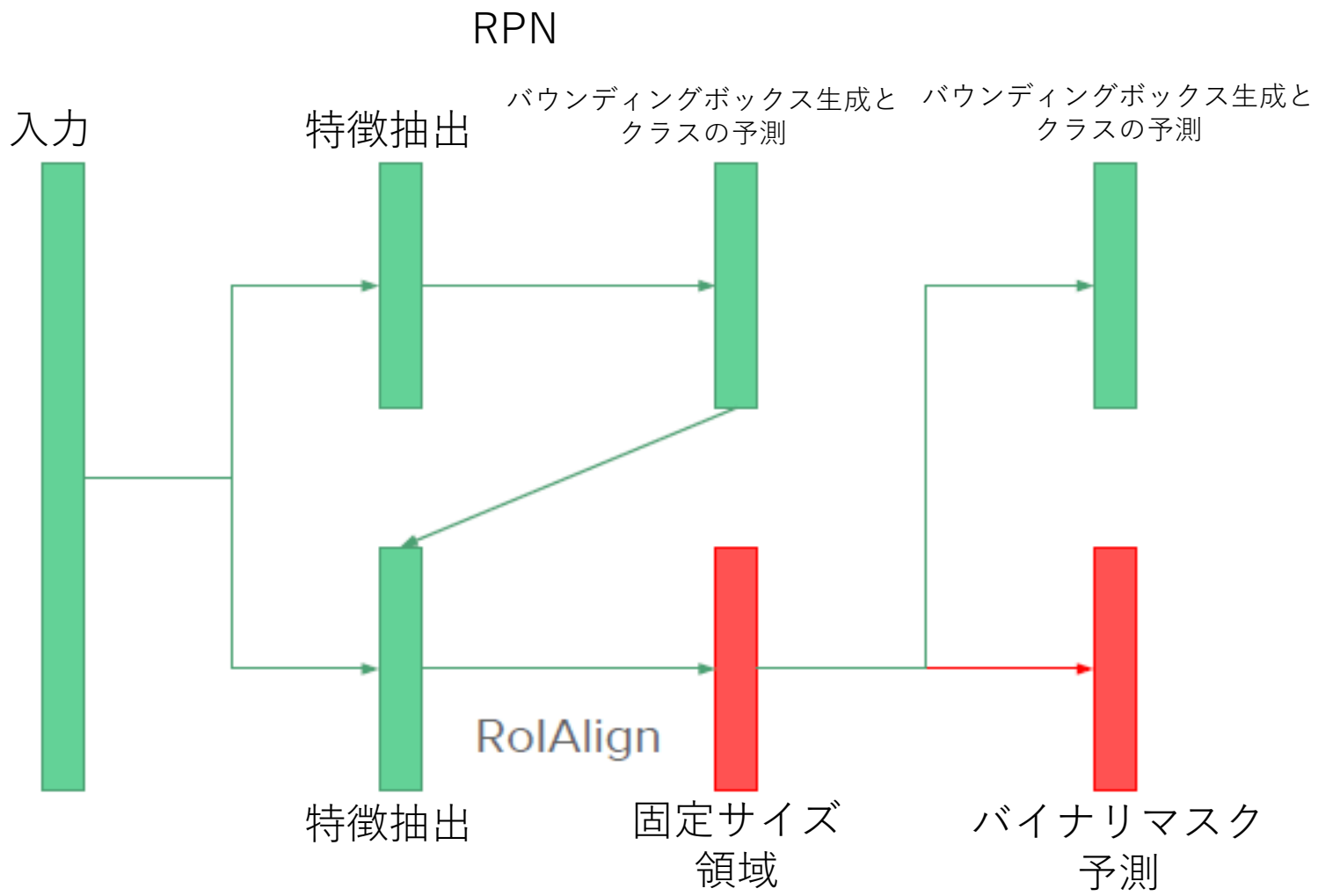
バウンディングボックスの生成

- より正確なバウンディングボックスを得るためにバウンディングボックスオフセットをさらに洗練させる
- 4つの値を生成
  - 左上X座標
  - 左上Y座標
  - バウンディングボックスの幅
  - バウンディングボックスの高さ

クラスの生成

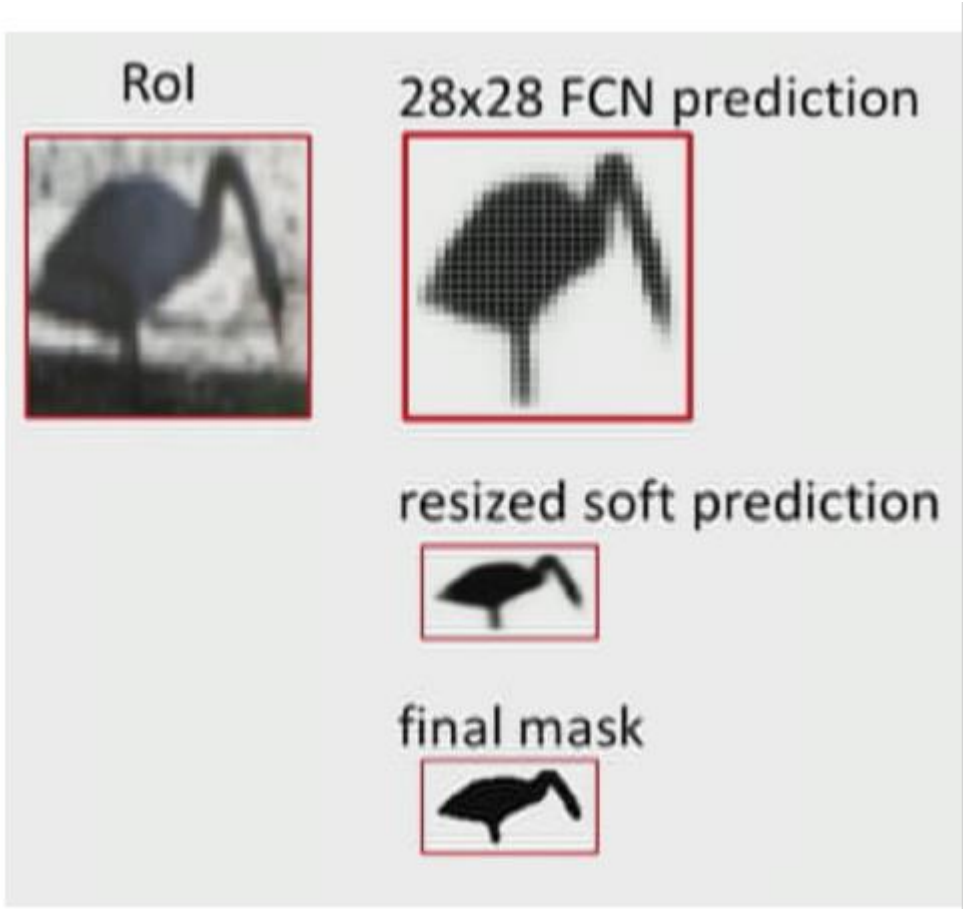
- ラベルセットからラベルを予測

バイナリマスクの予測


RPN
入力
特徴抽出
バウンディングボックス生成と
クラスの予測
バウンディングボックス生成と
クラスの予測
RoIAlign
特徴抽出
固定サイズ
領域
バイナリマスク
予測

バイナリマスクの予測

- 次元：Km^2
  - Kはクラス番号を表す
  - m^2は空間分解能を表す
  - マスクは元のオブジェクトの形状に合うようにサイズは変更さる



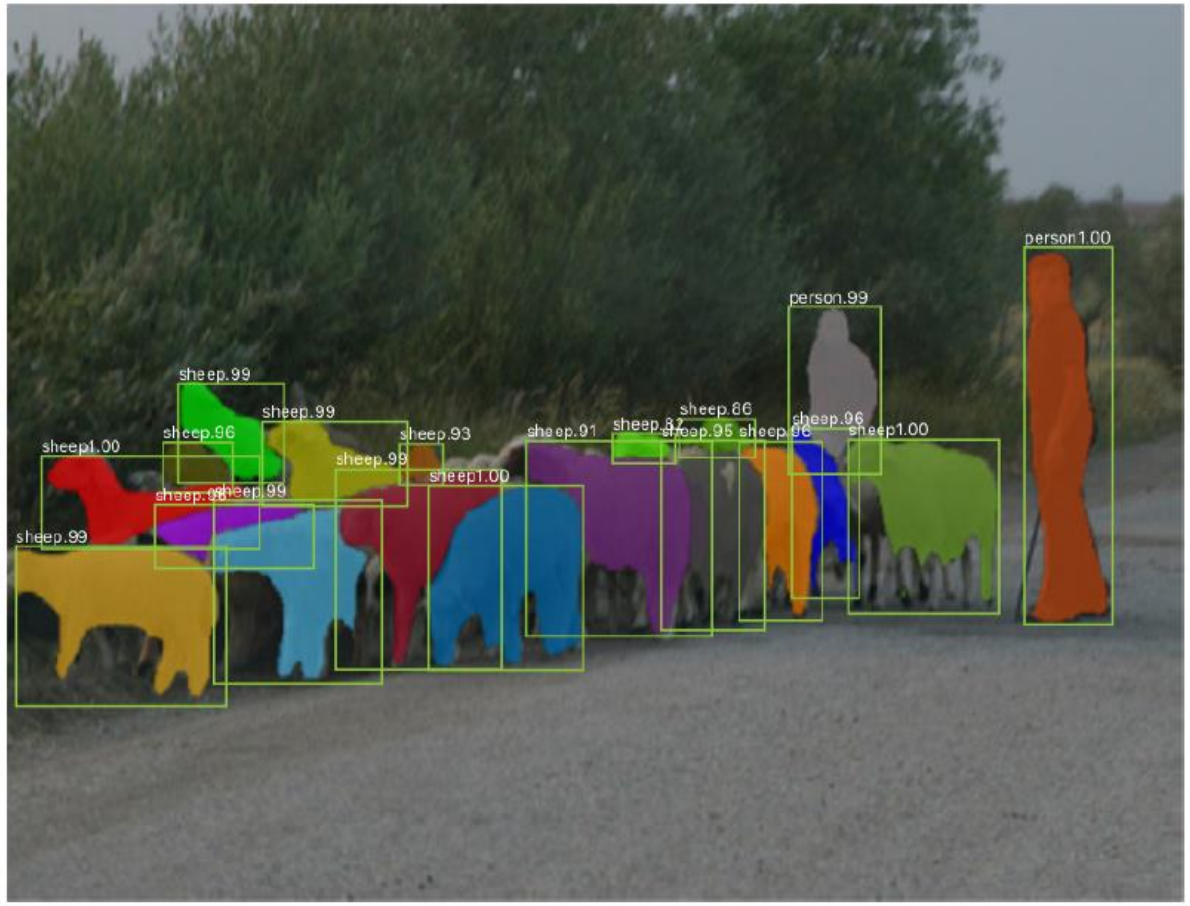
損失関数

- $L = L_{cls} + L_{box} + L_{mask}$
- 分類のロス
  - マルチクラスのクロスエントロピー
- バウンディングボックスのロス
  - 正解バウンディングボックスと予測バウンディングボックスの間のsmooth L1損失
  - smooth L1損失は、L2損失よりも外れ値の影響を受けにくい。
- マスク損失
  - K番目のクラスでのみ定義され、Kは正解ラベル付きのクラス。
  - 平均バイナリクロスエントロピー損失として定義
  - したがってクラスを越えたマスクは競合しない。
    - インスタンスセグメンテーションに最適

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i),$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases}$$

実例：インスタンスセグメンテーション



実例：ポーズ推定

実行と学習環境の構築

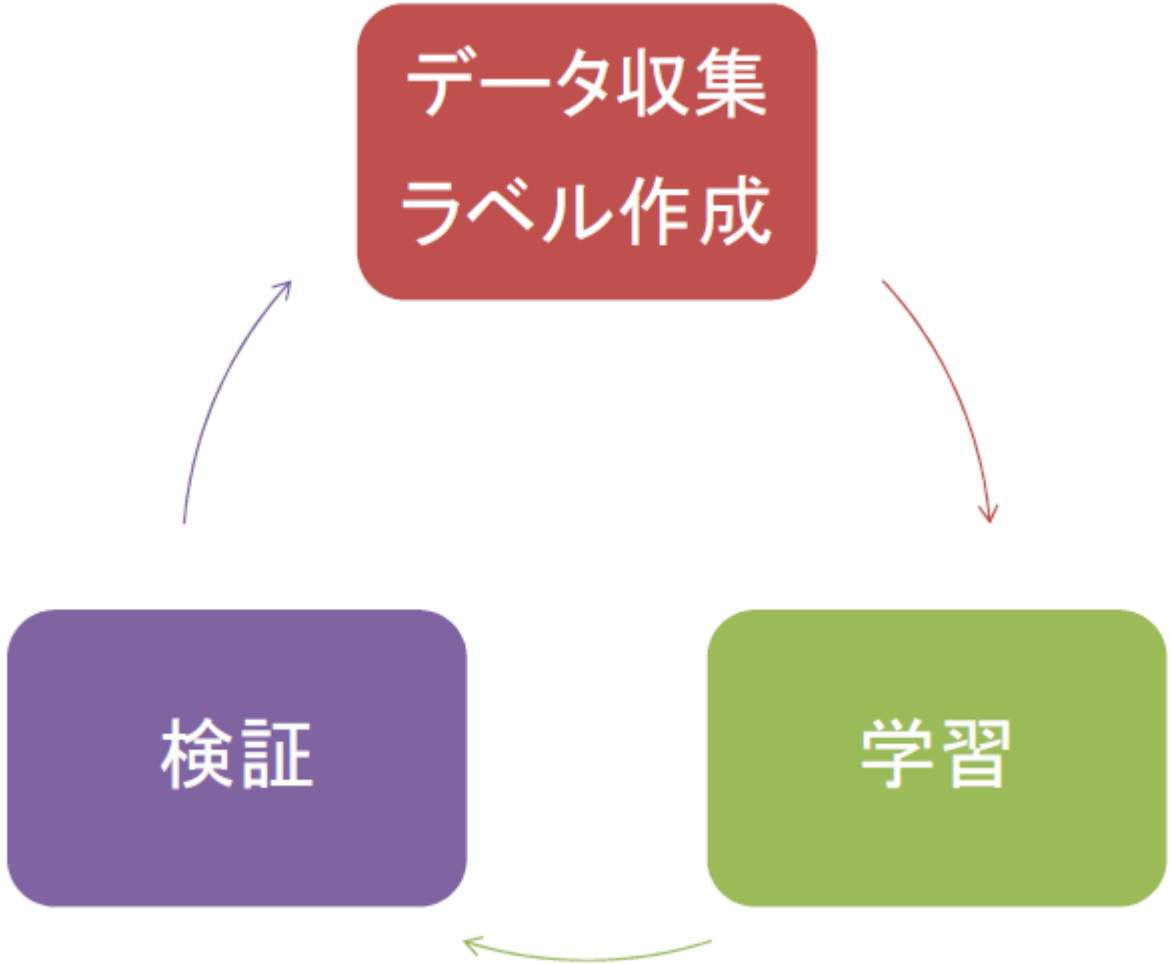
実行と学習環境の構築

- **ディープラーニングにおける学習の概要**
- 基本パッケージ
- GPU環境のインストール（CUDA9.0）
- Mask R-CNNのインストール

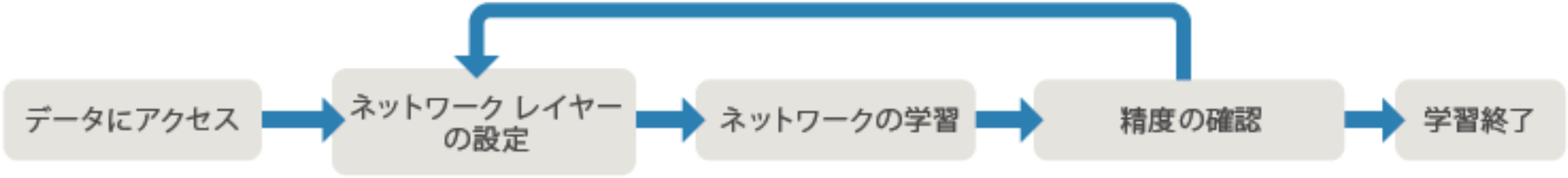
ディープラーニングの学習アプローチ

- ゼロからネットワークを学習させる
- 転移学習を使用して既存ネットワークを再学習させる
- 既存ネットワークを学習させて、セグメンテーションに適用する

学習の作業フロー



ゼロからモデルを学習



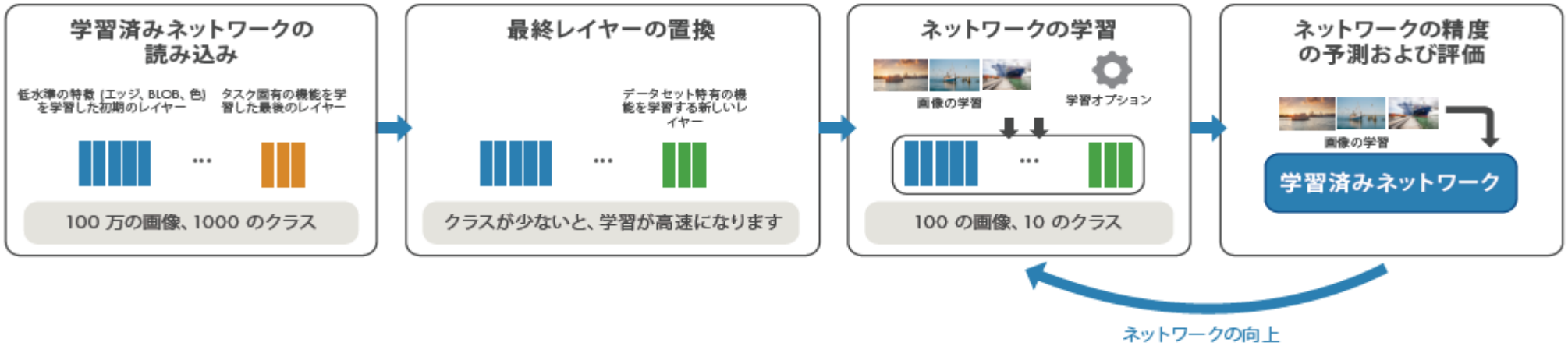
- シンプルなデータセットで品質の良いデータを使う
  - 実際にはあり得ないデータや人間が見ても判断できないデータでないこと
  - 本番データを意識するべき
  - 特定の訓練データばかりで学習し過ぎないこと
- 水増し
  - 元の学習データに変換を加えてデータ量を増やすテクニック
  - 水増し項目
    - ノイズを増やす(ガウシアンノイズやインパルスノイズ)・コントラストの調整・明るさを調整(ガンマ変換)・平滑化(平均化フィルタ)・拡大縮小・反転(左右/上下)・回転・シフト(水平/垂直)・部分マスク(CutoutやRandom Erasing)・トリミング(Random Crop)・変形・変色・背景の差し替えなど

主要な学習オプション

学習オプション	定義	効果
K分割（交差検証）	データセットをK個にわけてそれぞれで学習した結果を総合評価する。 一種のアンサンブル学習	例）学習データが12000ある時、K=5とし交差検証は、K=5なので学習データのうち訓練に使うデータ(10,000）を5分割。1回目は最初の4つを学習データ、残り1つを評価データに使う。2回目以降も同様に行う。全部で 5 回の学習結果の平均を取ったものが、訓練における認識率となる。
最大エポック	エポックは学習セット全体の学習アルゴリズムの全パスです。	指定するエポックの数が多いほどネットワークの学習時間が長くなりますが、それぞれのエポックごとに精度が向上する場合がある。
ミニバッチサイズ	ミニバッチは、GPU で同時に処理される学習データセットのサブセットです。	ミニバッチが大きいほど、学習の速度が向上しますが、最小サイズは GPU メモリによって決まります。学習中にメモリエラーが表示された場合、ミニバッチサイズを小さくする。
学習率	これは学習の速度を制御する主要なパラメーターです。	学習率が低いとより正確な結果が得られますが、ネットワークの学習に時間がかかる場合がある。

転移学習

- 学習済みネットワークを修正し、ネットワークの学習に転移学習を使用し実行する。（学習時間の大幅な削減）
- 手順
  - 学習済みネットワークをインポートする
  - 新しい認識タスクを実行するように最後の 3 つのレイヤーを設定する
  - ネットワークに新しいデータを学習させる
  - 結果をテストする



学習評価指標

- 指標
  - 正解率(accuracy), 適合率(precision), 再現率(recall)
- 結果を4つのパターンに分類
  - 正解が 1 のときに正しく 1 と予測 真陽性(true positive)
  - 正解が 1 のときに誤って 0 と予測 偽陰性(false negative)
  - 正解が 0 のときに誤って 1 と予測 偽陽性(false positive)
  - 正解が 0 のときに正しく 0 と予測 真陰性(true negative)

例として「ある病気の場合に真になる検査」の場合の分類

	検査で陽性	検査で陰性
実は病気	真陽性	偽陰性
実は健康	偽陽性	真陰性

- 適合率(precision)
  - 真陽性 / (真陽性 + 偽陽性) で求められる。
  - 例、「検査で陽性の場合に本当に病気だった確率」
  - 病気であると予測した結果がいかに正確であるかに焦点をあてた指標
- 再現率(recall)
  - 真陽性 / (真陽性 + 偽陰性)
  - 例、「実際に病気だった人が検査で正しく陽性になる確率」
  - 実際に真となる状態（病気であること）を漏れずに網羅出来ているかどうかに焦点をあてた指標
- 正解率 (accuracy)
  - (真陽性 + 真陰性) / (真陽性 + 偽陰性 + 偽陽性 + 真陰性)
  - 全てのパターンを考慮して「正しかった割合」を計算

①病気と検査の関係の場合、敏感に陽性になるような検査をつくれば再現率は上がるが、適合率や正解率は下がる。
② 一方、群衆の中から特定の人を見つけ出すなど、 予測や分類の結果の正しさが何より重視されるケースでは、再現率や正解率より適合率が重視されます。

実行と学習環境の構築

- ディープラーニングにおける学習の概要
- **基本パッケージ**
- GPU環境のインストール（CUDA9.0）
- Mask R-CNNのインストール

基本パッケージ

- Python3.6
- numpy
- scipy
- Pillow
- cython
- matplotlib
- scikit-image
- tensorflow>=1.3.0
- keras>=2.0.8
- opencv-python
- h5py
- lmgaug
- lmutils
- argparse
- IPython[all]

実行と学習環境の構築

- ディープラーニングにおける学習の概要
- 基本パッケージ
- **GPU環境のインストール（CUDA9.0）**
- Mask R-CNNのインストール

- ダウンロードサイト
  - https://developer.nvidia.com/cuda-90-download-archive?target_os=Linux&target_arch=x86_64&target_distro=Ubuntu&target_version=1604&target_type=debnetwork

CUDA Toolkit 9.0 Downloads

Select Target Platform

Click on the green buttons that describe your target platform. Only supported platforms will be shown.

Operating System	Windows, **Linux**, Mac OSX
Architecture	**x86_64**, ppc64le
Distribution	Fedora, OpenSUSE, RHEL, CentOS, SLES, **Ubuntu**
Version	17.04, **16.04**
Installer Type	runfile (local), **deb (local)**, deb (network), cluster (local)

Download Installers for Linux Ubuntu 16.04 x86_64

The base installer is available for download below.
There are 4 patches available. These patches require the base installer to be installed first.

> Base Installer — Download (1.2 GB)

以下を選択する：
Operating System：Linux
Architecture：x86_64
Distribution：Ubuntu
Version：16.04　※18.04とも互換がある
Installer Type：deb(local)

Download(1.2GB) をクリックしてダウンロード開始する。

CUDA9.0以降は他のソフトがまだサポートしないものがあるので注意が必要

CUDA Toolkit 9.0のインストール

```
sudo dpkg -i cuda-repo-ubuntu1604_9.0.*_amd64.deb
sudo apt-key adv --fetch-keys ¥
    http://developer.download.nvidia.com/compute/cuda/repos/ubuntu1604/x86_64/7fa2af80.pub
sudo apt update
sudo apt install cuda-9-0
```

CUDA パスのインストール (.bashrc)

```
echo -e "¥n## CUDA and cuDNN paths" >> ~/.bashrc
echo 'export PATH=/usr/local/cuda-9.0/bin:${PATH}' >> ~/.bashrc
echo 'export LD_LIBRARY_PATH=/usr/local/cuda-9.0/lib64:${LD_LIBRARY_PATH}' >> ~/.bashrc
source ~/.bashrc # CUDAのパスが書き込まれた~/.bashrcを読み込む。
```

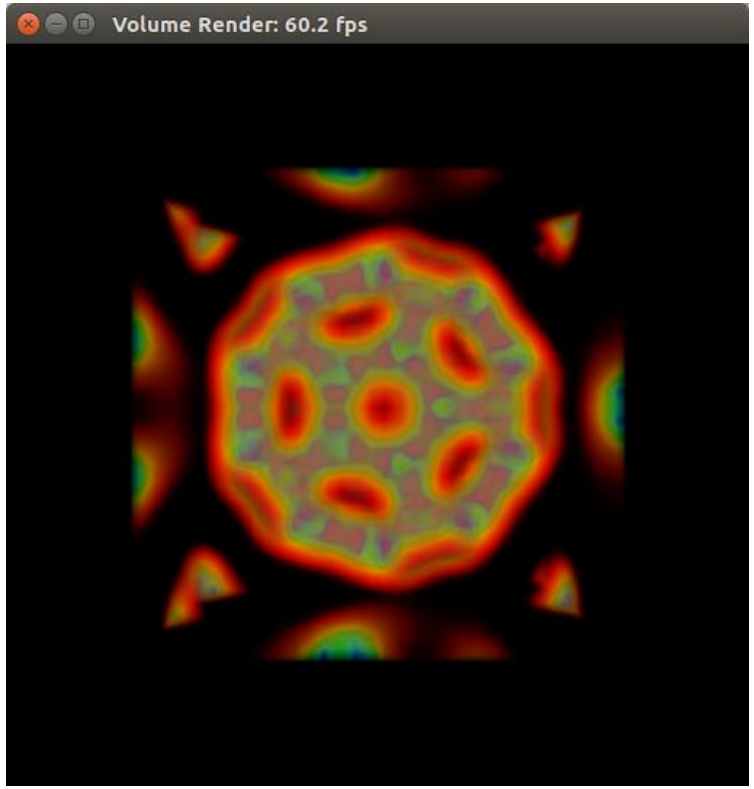
CUDAのパスが通っているか確認。

```
echo $PATH                  # 出力に"/usr/local/cuda-9.0/bin"が含まれているか？
echo $LD_LIBRARY_PATH  # 出力に"/usr/local/cuda-9.0/lib64"が含まれているか？
which nvcc                  # 出力が"/usr/local/cuda-9.0/bin/nvcc"になっているか？
nvidia-smi                  # nvidiaのGPUの情報が表示されているか？
```

cuDNN 7.0のダウンロードとインストール

- ダウンロードサイト
  - https://developer.nvidia.com/rdp/cudnn-download
  - cuDNN 7.0 for CUDA 9.1のdebパッケージをダウンロード
  - アクセスするにはメンバー登録が必要
- 以下の３つのファイルをダウンロード
  - cuDNN v7.0.* Runtime Library for Ubuntu16.04 (Deb)
  - cuDNN v7.0.* Developer Library for Ubuntu16.04 (Deb)
  - cuDNN v7.0.* Code Samples and User Guide for Ubuntu16.04 (Deb)
- インストール
  - dpkg -i libcudnn7-doc_7.0*+cuda9.0_amd64# Install Runtime library
  - dpkg -i libcudnn7_7.0*+cuda9.0_amd64.deb # Install developer library
  - dpkg -i libcudnn7-dev_7.0*+cuda9.0_amd64.deb # Install code samples and userguide.deb
- 動作確認
  - cuda-install-samples-9.0.sh ~ # ホームディレクトリにサンプルコードをコピー。
  - cd ~/NVIDIA_CUDA-9.0_Samples/
  - make 2_Graphics/volumeRender # サンプルの実行ファイルがあるディレクトリに移動。

サンプル画像：



実行と学習環境の構築

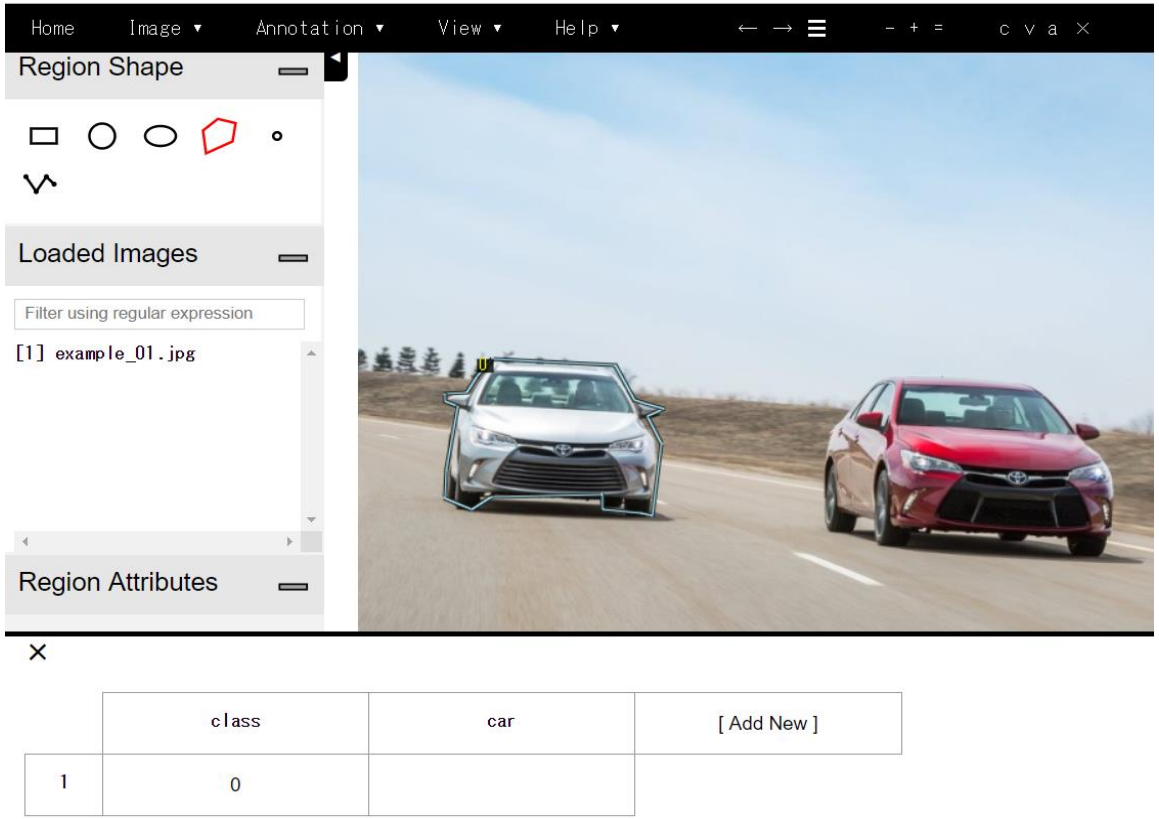
- ディープラーニングにおける学習の概要
- 基本パッケージ
- GPU環境のインストール（CUDA9.0）
- **Mask R-CNNのインストール**

Mask R-CNNのインストール

- Mask R-CNNレポジトリをクローン
  - git clone https://github.com/matterport/Mask_RCNN
- Jupyter Notebookのインストール
  - pip install jupyter
  - 但し、Mask_R_CNNはPyhton3でしか動作しない。
- 必要なライブラリインストール
  - pip install –r requirements.txt
  - python setup.py install
- COCO APIのインストール
  - git clone https://github.com/cocodataset/cocoapi.git
  - cd coco/PythonAPI
  - python setup.py install
- 動作確認
  - jupyter notebook
  - samples > demo.ipynb"をクリック
  - runボタンをクリックで実行

Mask R-CNNのアノテーションファイル生成ツールのインストール

- セグメンテーションのため多角形で切り出しが必要
- VIAツールのダウンロードサイト
  - http://www.robots.ox.ac.uk/~vgg/software/via/
  - Version1.6.xをダウンロード
    - Mask R-CNNは、2.0.xもサポートしている
- 起動
  - Via.htmlをクリックし、ブラウザより起動
  - Region Attributesにクラスのラベルを設定。
    - Attibute : class , value : 0 (例えば、0がcar)
  - 出力は、jsonファイルである。



インスタンスセグメンテーションのデモ

- 物体認識
- 背景ぼかし
- 風船の学習と認識

Thank you