やまぐち
高校生データサイエンティスト育成講座
～ステップアップ編～

1. はじめに

自己紹介

名前：高田 朋貴（Takada Tomoki）
趣味：和太鼓、おわら
所属：SIGNATE Inc. EaaS Product Group VP
専門：情報科学（理学博士）

★お仕事
・AI開発コンサルティング
・AI受託開発
・SIGNATE Cloud責任者
・Udemy講師
・DS協会主催講座講師
・企業等AI研修講師
・最近、本を執筆

本講座について

- **概要**
  - データサイエンスの中級者向け
  - Pythonを使って簡単な分析ができることを前提
  - 画像処理とDeepLearningの基礎を学びます
  - 分析コンペティション※に参加！
  - 学ぶべきことのヒントを提示します

※モデルの予測精度を競う競技会

本講座の目標

① 画像処理の基礎がわかる

② DeepLearningの基礎がわかる

③ Kerasを使った簡単な実装ができるようになる

④ DeepLearningをチューニング方法を知る

今回の課題は２つ！

コンペに挑戦し、精度を上げながら「何をすればモデルの精度が上がるのか？」をたくさん試行錯誤する！

コンペテーマ
「画像ラベリング（20種類）」

自分で画像分類に関するテーマを決めていただき、そのテーマに関するデータを収集し、その結果を発表する！

例）
- お金の硬貨を見分けるAIを作る
- スマホで各硬貨の写真を自分で作る
- 硬貨を見分ける簡単なアプリまで作ってみた

カリキュラム

講座①（7/30(日)）

【内容】NNに関連する用語や概念を座学で学び、単純なNNモデルで精度検証をし、画像における古典的手法と比較する
【課題】NNモデルを使い、学習iterationや学習率などの値を実験的に変更し、精度を検証する

▼Day1
10:00 ~ 10:15：ガイダンス
10:15 ~ 11:15：モデリングのキホン
11:15 ~ 12:00：画像分析のキホン
12:00 ~ 13:00：昼休憩
13:00 ~ 14:00：画像分析チュートリアル
14:00 ~ 14:10：休憩
14:10 ~ 15:30：DLのキホン・チュートリアル
15:30 ~ 15:40：休憩
15:40 ~ 16:50：DLチュートリアル続き
16:50 ~ 17:00：閉講・アンケート

講座②（9/18（月・祝））

【内容】主にDLにおける精度向上のテクニックであるData Augmentation及び転移学習について学ぶ
【課題】
① 画像分類コンペに挑戦いただき、精度が良いモデルを検討いただく
② ご自身で画像分類タスクを考えていただき、データも収集した上でモデル作成まで行い、その結果をレポートにまとめていただく

▼Day2
10:00 ~ 10:10：ガイダンス
10:10 ~ 11:10：Data Augmentation
11:10 ~ 11:15：休憩
11:15 ~ 12:15：チュートリアル
12:15 ~ 13:15：昼休憩
13:15 ~ 14:30：転移学習
14:30 ~ 14:35：休憩
14:35 ~ 15:35：コンペチュートリアル
15:35 ~ 15:40：休憩
15:40 ~ 16:50：ワーク
16:50 ~ 17:00：閉講・アンケート

講座③（12/24（日））

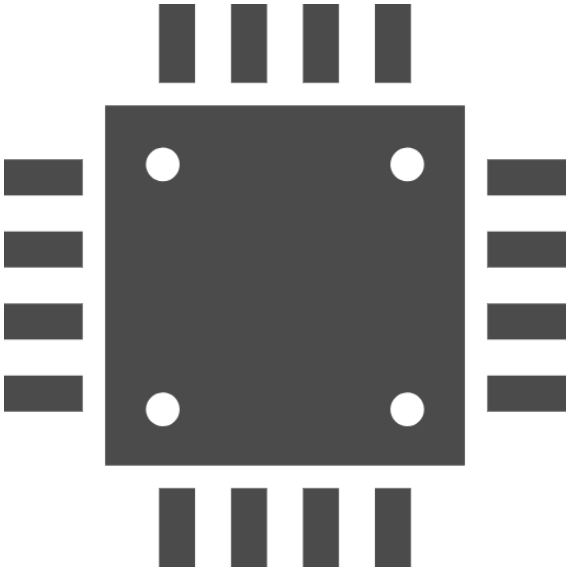
【内容】講座②で出題した課題に関する成果を発表いただき、学んだ内容を他受講生にご共有いただくとともに、結果に対するFBを行う。

▼Day3
10:00 ~ 10:10：ガイダンス
10:10 ~ 12:15：成果発表会①
12:15 ~ 13:15：昼休憩
13:15 ~ 15:35：成果発表会②
15:35 ~ 15:40：休憩
15:40 ~ 16:50：座談会
16:50 ~ 17:00：閉講・アンケート

2. モデリングのキホンの「キ」

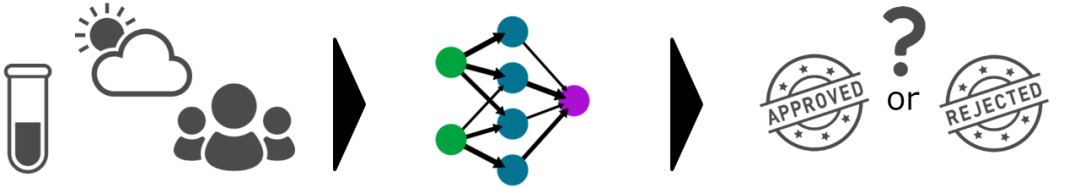
AIと機械学習

- どうやってAIに知的処理をさせるのか？



●機械学習（Machine Learning）

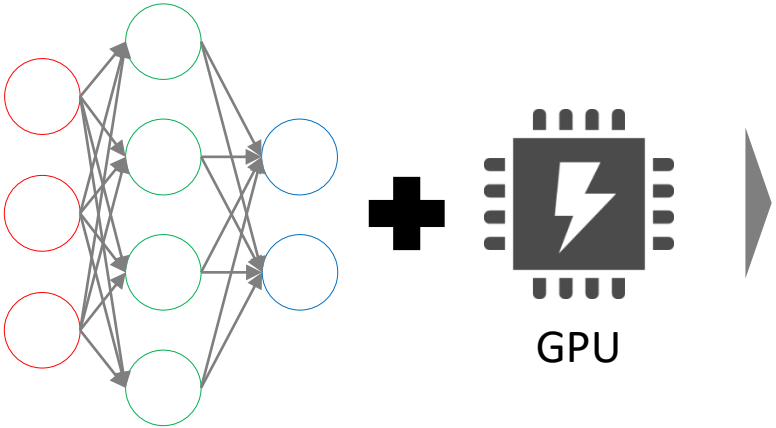
“データから**パターン**を学習し、パターンを発見したり、その結果を利用して将来を予測する”手法のこと。
確率や行列演算等、数学的なアプローチを用いる。

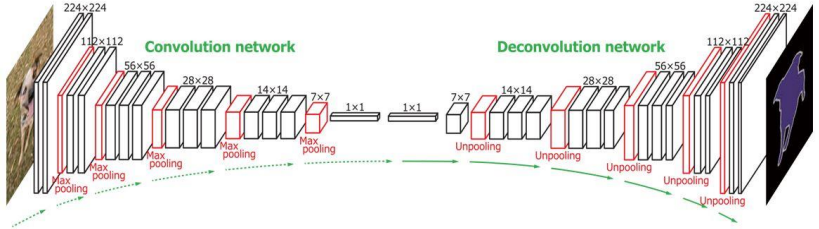


入力データ　　パターン抽出　　過去パターンに基づき予測

機械学習と深層学習

- 深層学習（Deep Learning）
  - Neural Networkと呼ばれる機械学習手法の発展形
    - 昔は2~3層、今は多いもので1000層を超える
  - ハードウェア技術の進化と結びつき実現

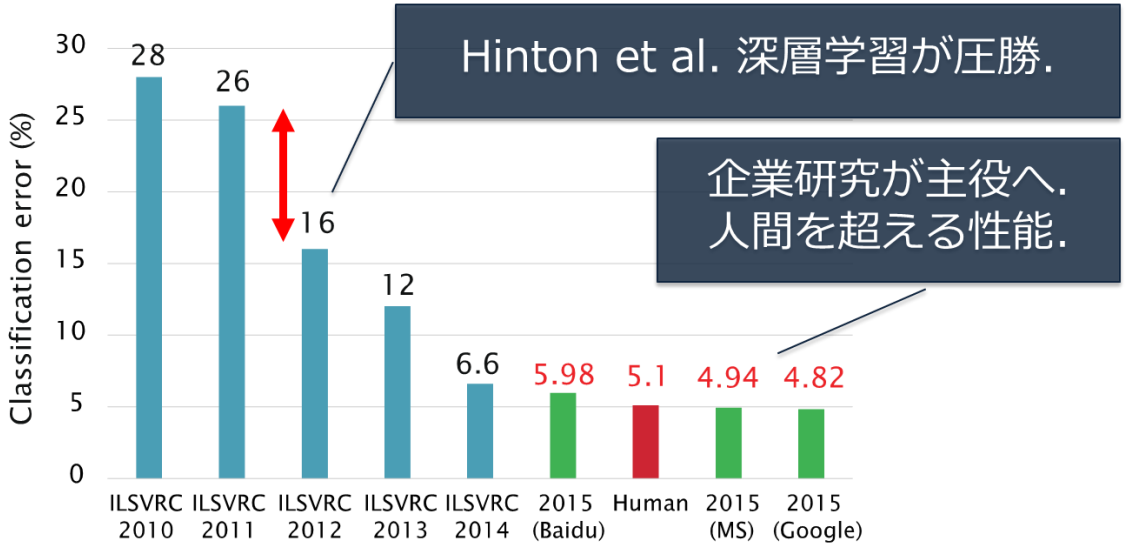




【引用】http://cvlab.postech.ac.kr/research/deconvnet/

機械学習と深層学習

- 深層学習（Deep Learning）
  - 特徴量加工　⇒　構造設計
  - 特に画像/動画データに対して既存手法よりも高い精度を誇る



【引用】中山 英樹, 深層畳み込みニューラルネットワークによる画像特徴抽出と転移学習

特徴量からみる違い

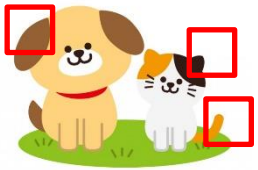
- **犬**と**猫**を見分けたい
  - 特徴量＝予測する際に注目すべき特徴

特徴量からみる違い

- **犬**と**猫**を見分けたい
  - 特徴量＝予測する際に注目すべき特徴

一般的な機械学習

どこを見れば良さそうかを教えてあげる
・耳を見た方がいいよ
・ひげの長さにも違いがあるよ
・しっぽの長さにも特徴あるよ

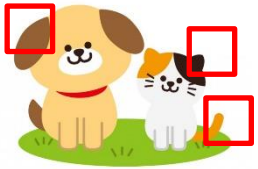


特徴量からみる違い

- **犬**と**猫**を見分けたい
  - 特徴量＝予測する際に注目すべき特徴

一般的な機械学習

どこを見れば良さそうかを教えてあげる
・耳を見た方がいいよ
・ひげの長さにも違いがあるよ
・しっぽの長さにも特徴あるよ

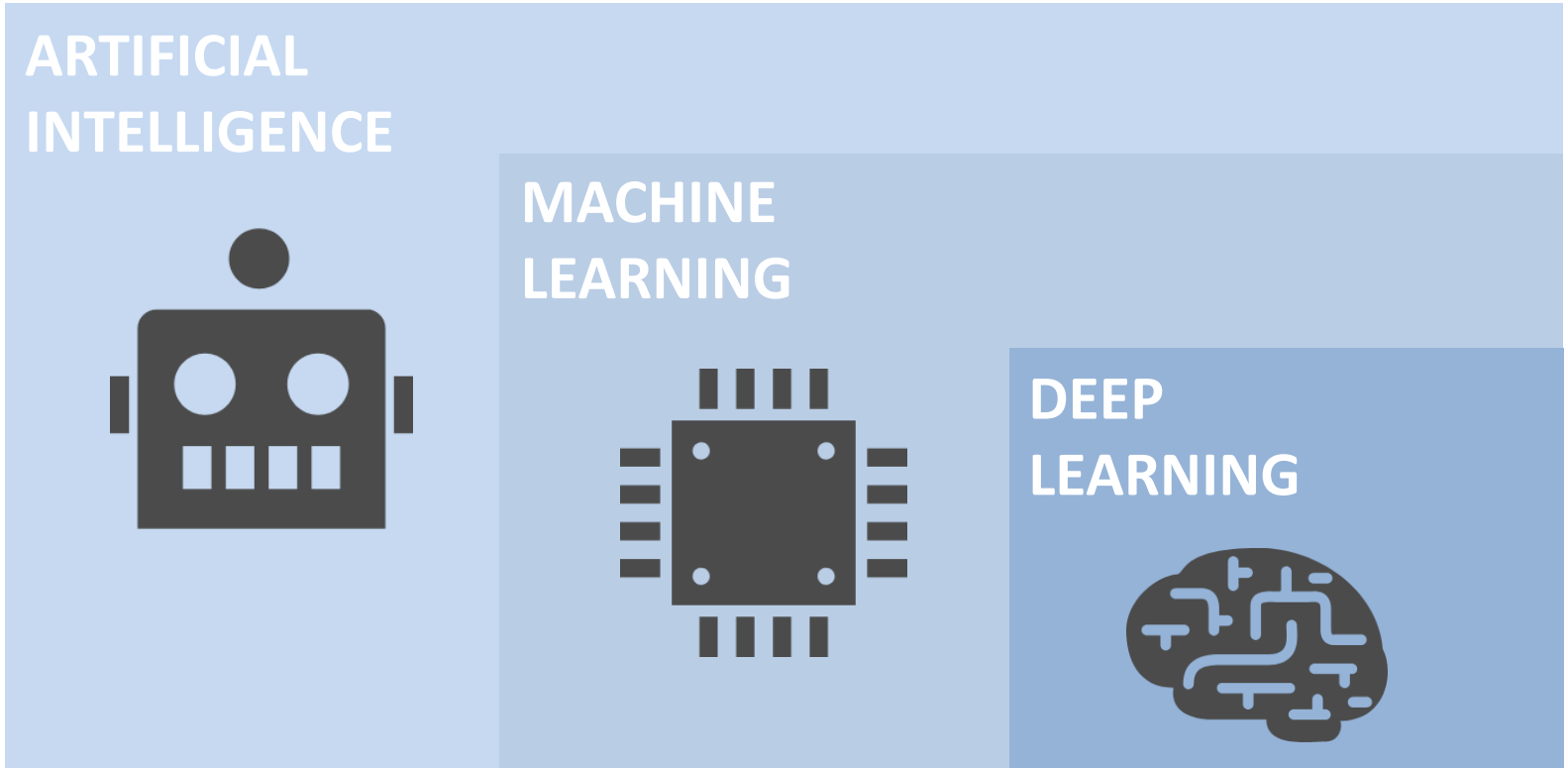


ディープラーニング

データはあげるから自分で見つけて。
その代わり、君のスペックはこっちで定義しておくから。

あ、はい。

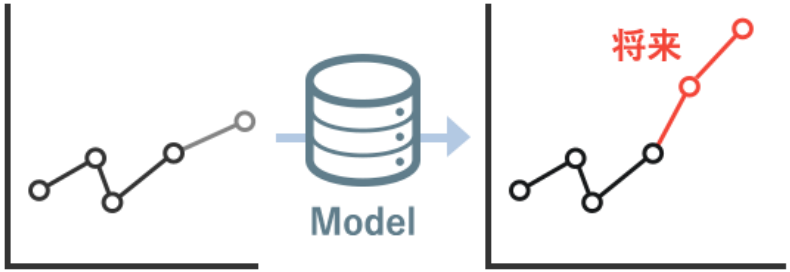
まとめると



代表的な2種類の予測問題

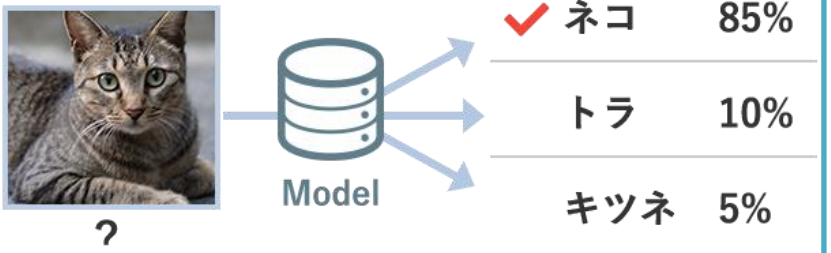
回帰問題

目的変数が**数値**



分類問題

目的変数が**カテゴリ**



それって本当に予測できてる？

①ある日・・・

このデータを使って
良い精度のモデルを作ってくれ

Bさん

Aさん

それって本当に予測できてる？

①ある日・・・

このデータを使って
良い精度のモデルを作ってくれ

わかりました！

Bさん

Aさん

それって本当に予測できてる？

②モデリング前

とりあえずモデリングしてみよう。
もらったデータを全部使って、
このデータで良い精度のモデルを作れ
ばいっか！

Aさん

それって本当に予測できてる？

③モデルの評価

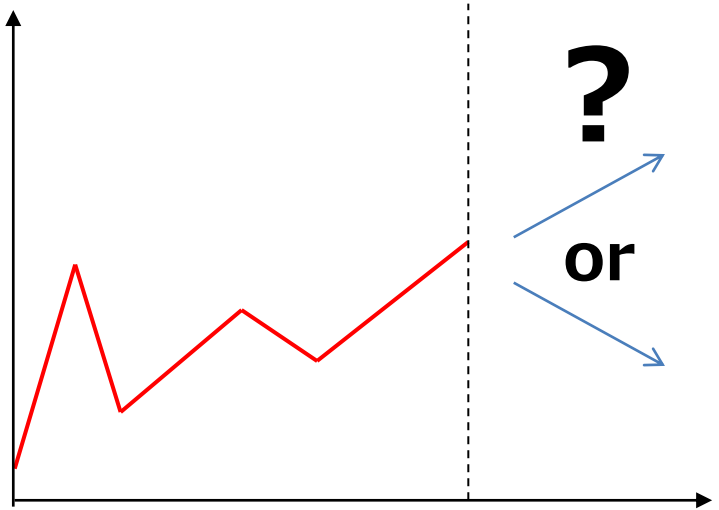
予測精度99%のモデルが出来た！
俺って天才なのかもしれない！

Aさん

それって本当に予測できてる？

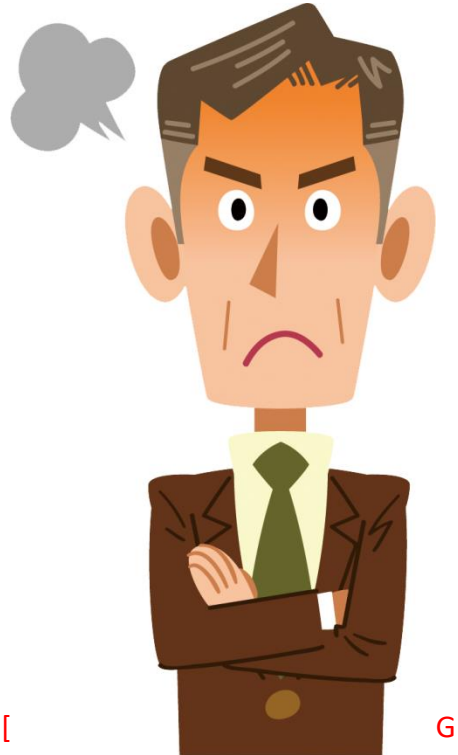
④実際に運用してみたら…

Aさん



それって本当に予測できてる？

④実際に運用してみたら…



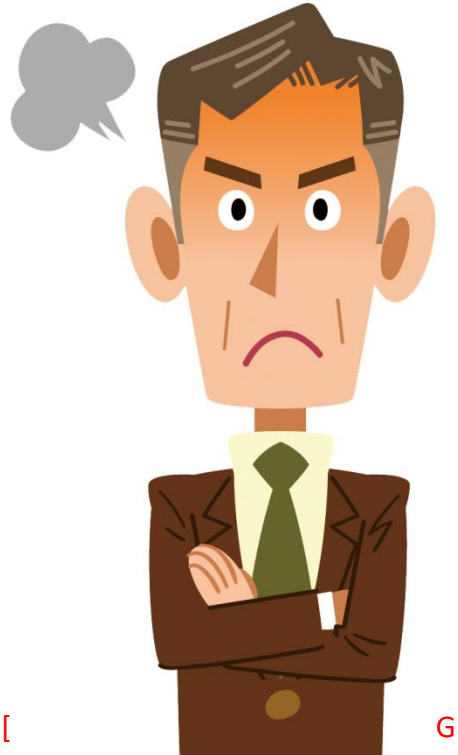
全然予測精度が出ないじゃないか！
どうなってるんだ！！！

Bさん

Aさん

それって本当に予測できてる？

④実際に運用してみたら…



全然予測精度が出ないじゃないか！
どうなってるんだ！！！

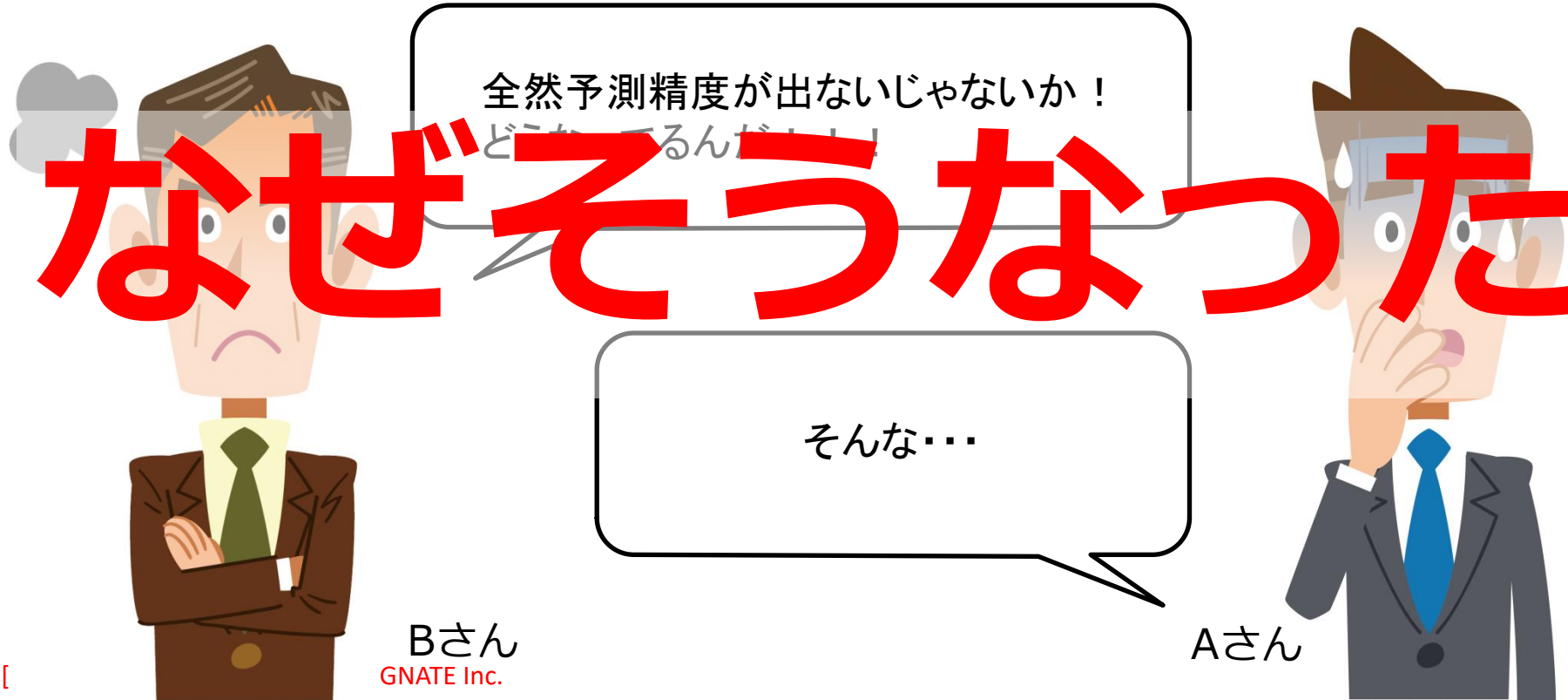
そんな・・・

Bさん

Aさん

それって本当に予測できてる？

④実際に運用してみたら…



何がいけなかったのか？

- いきなりモデルを作り始めた
  - 基礎分析を怠ることなかれ

- モデルを作るときに全てのデータを使ってしまった
  - その結果、モデルが過学習と呼ばれる状態になってしまった

過学習（Overfitting）とは？

- モデル作成に使ったデータだけに特化し過ぎたモデルを作ってしまうこと

例．文系or理系出身かを判定するモデルを作る

【部署Aでは】

部署Aでは、
・理系の男性は全員メガネをしている
・文系の女性は全員メガネをしていない

だから、男性でメガネをしていれば理系、女性でメガネをしていなければ文系と判別すれば、精度100%のモデルを作れるぞ！

ところが…

【部署Bでは】

部署Bでは、
・理系の男性でメガネをしていない男性が多くいた
・文系の女性でメガネをしている女性も多くいた

全体の傾向を考えずに部署Aだけのデータだけに引っ張られた結果、他部署では予測できていないモデルになってしまった…

作ったモデルの精度を検証するには？

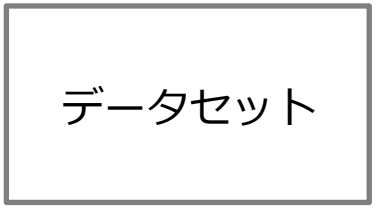
- **予測モデルのゴール**
  - **“未知のデータ”**も予測できるような汎用性あるモデルを作ること

- **どうすればいい？**
  - データを分割して擬似的に未知のデータを作る
    - 片方でモデルを作り、残りを未知のデータする
    - この未知のデータを上手く予測できることを目標とする

未知のデータを予測できる？

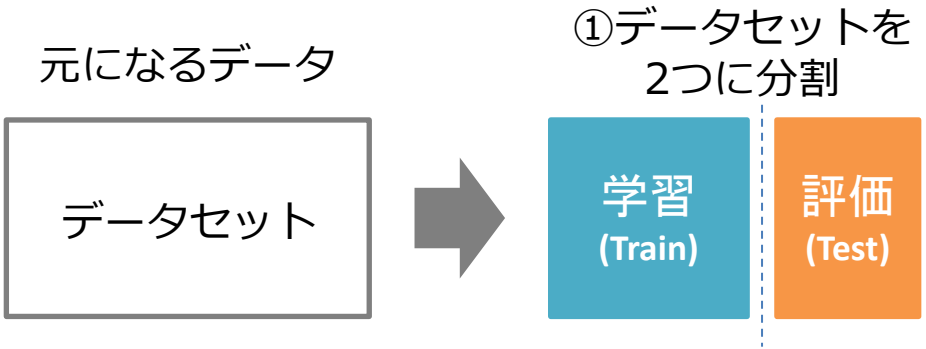
“**未知のデータ**でも予測できるか？（汎用性）”
を評価することが大切

元になるデータ



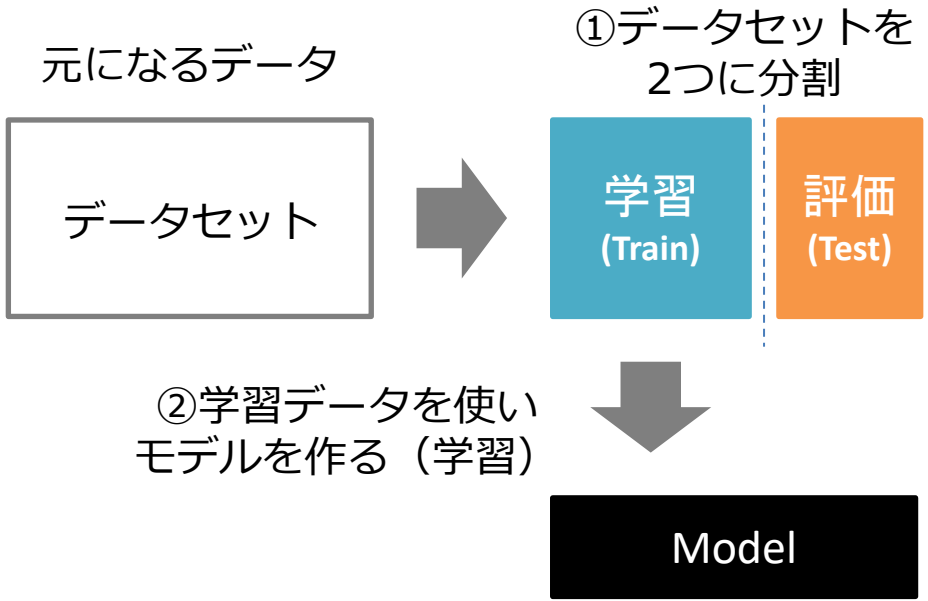
未知のデータを予測できる？

“**未知のデータ**でも予測できるか？（汎用性）”
を評価することが大切



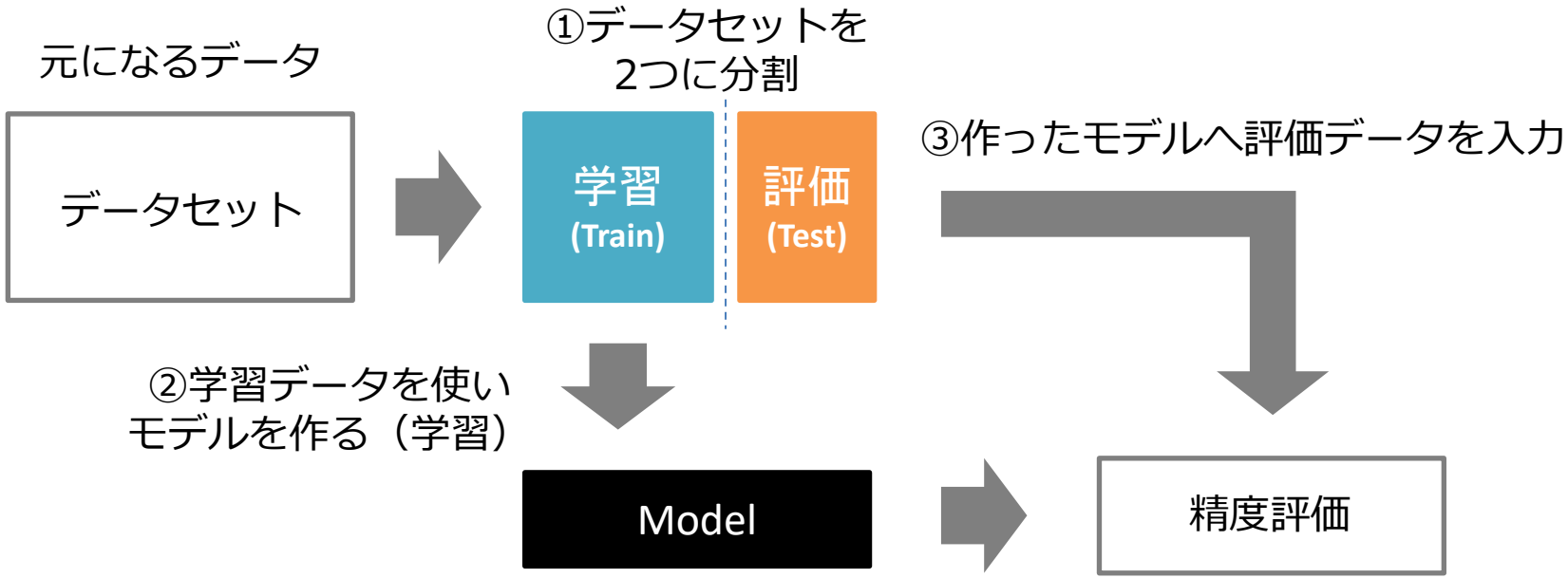
未知のデータを予測できる？

“**未知のデータ**でも予測できるか？（汎用性）”
を評価することが大切



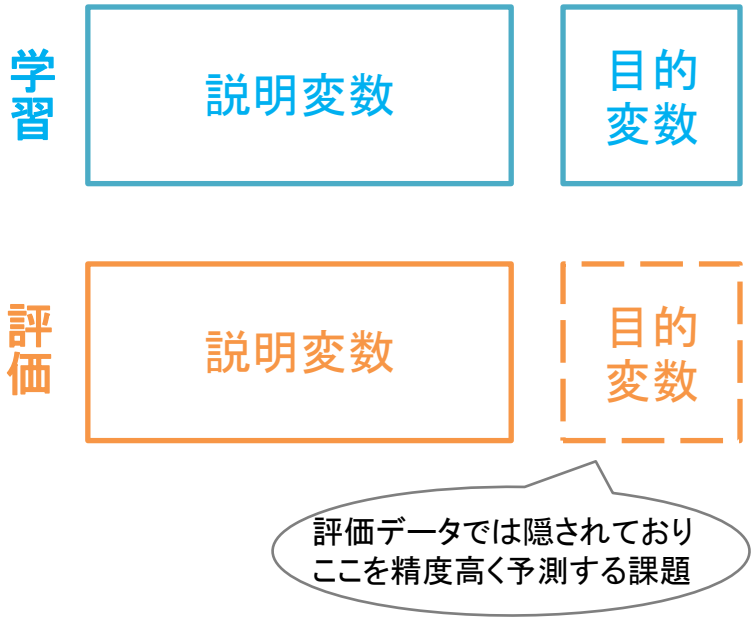
未知のデータを予測できる？

“**未知のデータ**でも予測できるか？（汎用性）”
を評価することが大切



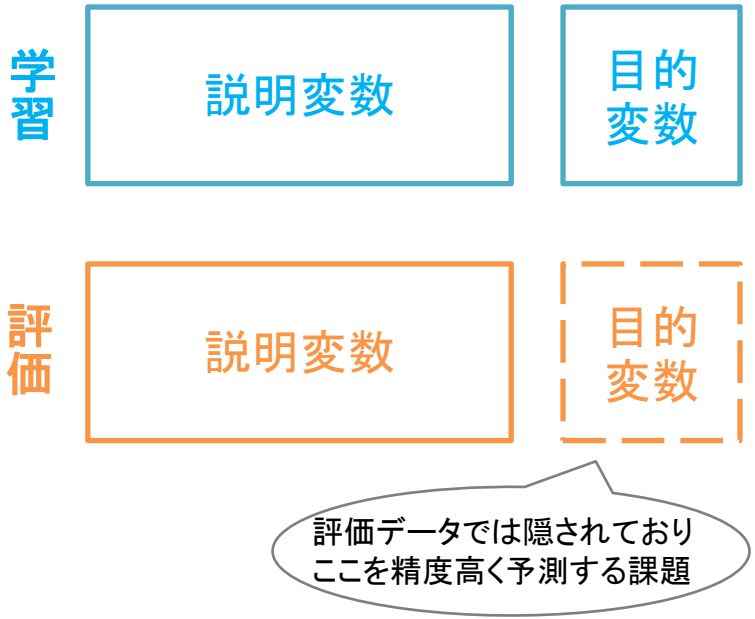
CompeやChallenge Missionでは

既にtrainとtestに分割されて配布されます



CompeやChallenge Missionでは

既にtrainとtestに分割されて配布されます



【分析のステップ】

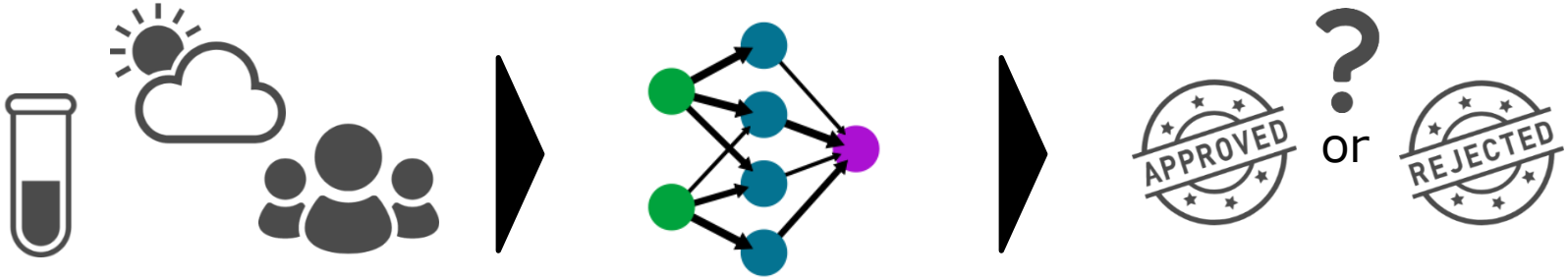
①学習データを使い、モデルを作成する

②学習データを使い、良いモデルが作れているかどうかを確認する

③作成したモデルを使い、評価データを予測し、投稿する

機械学習とは？（Machine Learning）

- “データから**パターン**を学習し、パターンを発見したり、その結果を利用して将来を予測すること”
  - 一般に予測精度が高い
  - 学習結果が複雑であり、人間の理解が難しいものが多い
  - ハイパーパラメータと呼ばれる変数の調整が重要



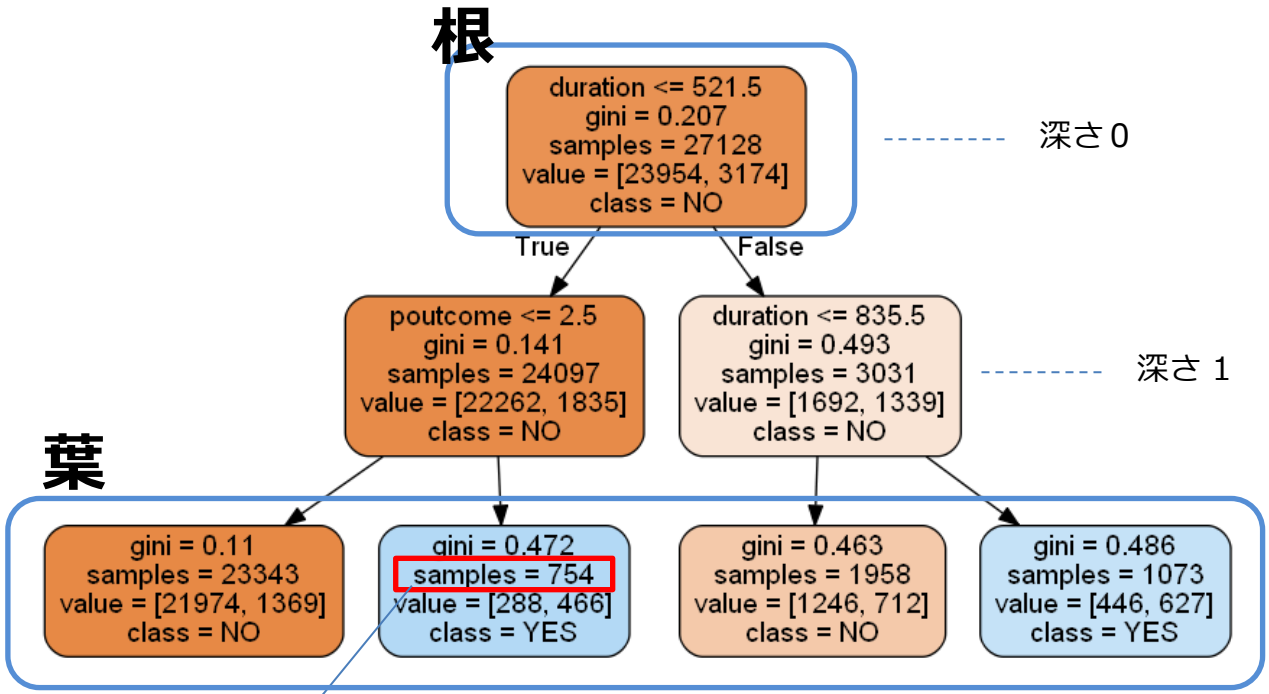
入力データ　　パターン抽出　　過去パターンに基づき予測

パラメータとは？

- 正確に言うとハイパーパラメータといいます
  - データに合わせて、設定しておかなければならない値
  - モデルが複雑になるほどパラメータ数は増加する傾向

<E.g.>

決定木の重要なパラメータはどれ？



①max_depth（決定木の深さ）
・深ければ深いほど、分岐数も増える為、説明力が上がる
・深すぎると意味のない分岐もできやすく、過学習のおそれがある

②min_samples_leaf（葉に属する最小サンプル数）
・葉に所属する最低サンプル数を制御できる
・サンプル数が少ない＝信憑性の低い分岐の可能性

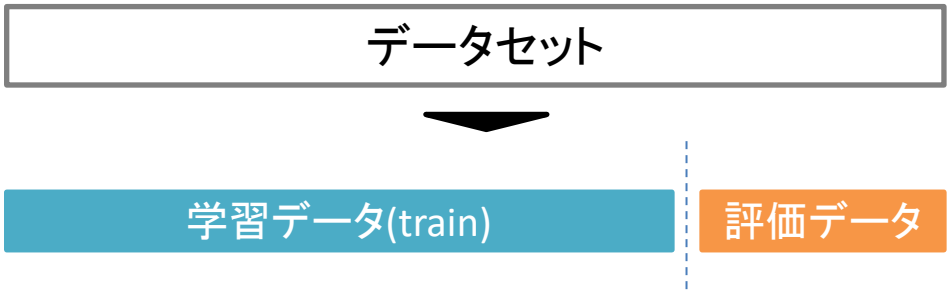
パラメータの検証方法

- パラメータは注意してチューニングしないと過学習してしまう危険がある
- **“未知のデータ”**に対しても予測できるように調整しなければならない
  - 答えがわかっているデータ（学習データ）から仮の未知データを作りだせばいい

パラメータの検証方法

- パラメータは注意してチューニングしないと過学習してしまう危険がある
- **“未知のデータ”**に対しても予測できるように調整しなければならない
  - 答えがわかっているデータ（学習データ）から仮の未知データを作りだせばいい

<基本的な考え方>

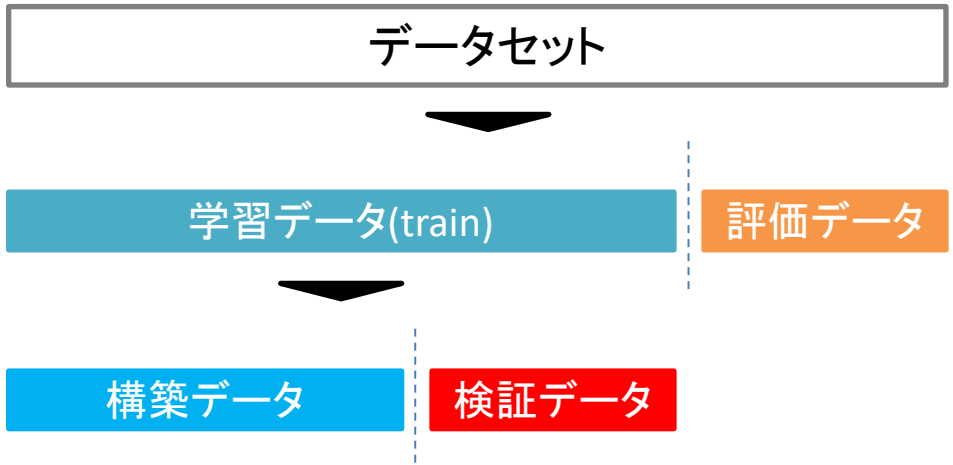


①未知のデータへの評価の為に
学習データと評価データに分ける
※コンテストではここまでしてある

パラメータの検証方法

- パラメータは注意してチューニングしないと過学習してしまう危険がある
- **“未知のデータ”**に対しても予測できるように調整しなければならない
  - 答えがわかっているデータ（学習データ）から仮の未知データを作りだせばいい

<基本的な考え方>



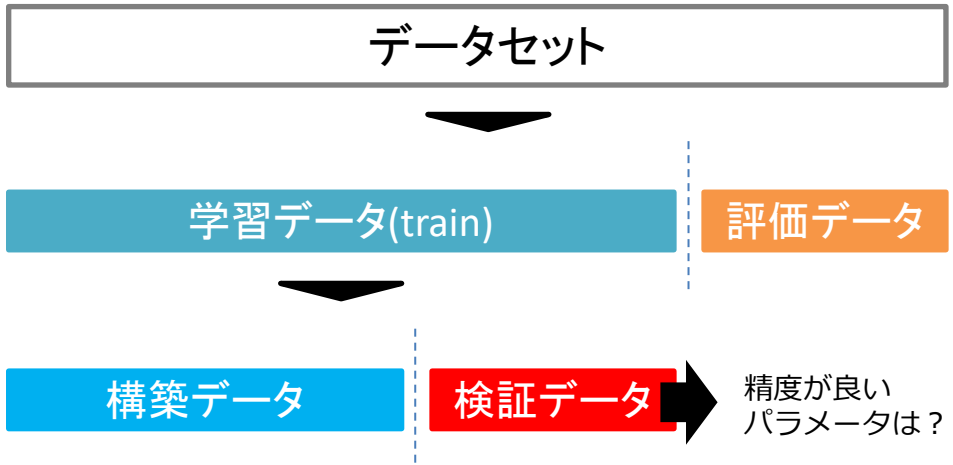
①未知のデータへの評価の為に
学習データと評価データに分ける
※コンテストではここまでしてある

②パラメータをチューニングする為に
学習データを更に分割し、モデル構築用
データとモデル検証用データを作る

パラメータの検証方法

- パラメータは注意してチューニングしないと過学習してしまう危険がある
- **“未知のデータ”**に対しても予測できるように調整しなければならない
  - 答えがわかっているデータ（学習データ）から仮の未知データを作りだせばいい

<基本的な考え方>



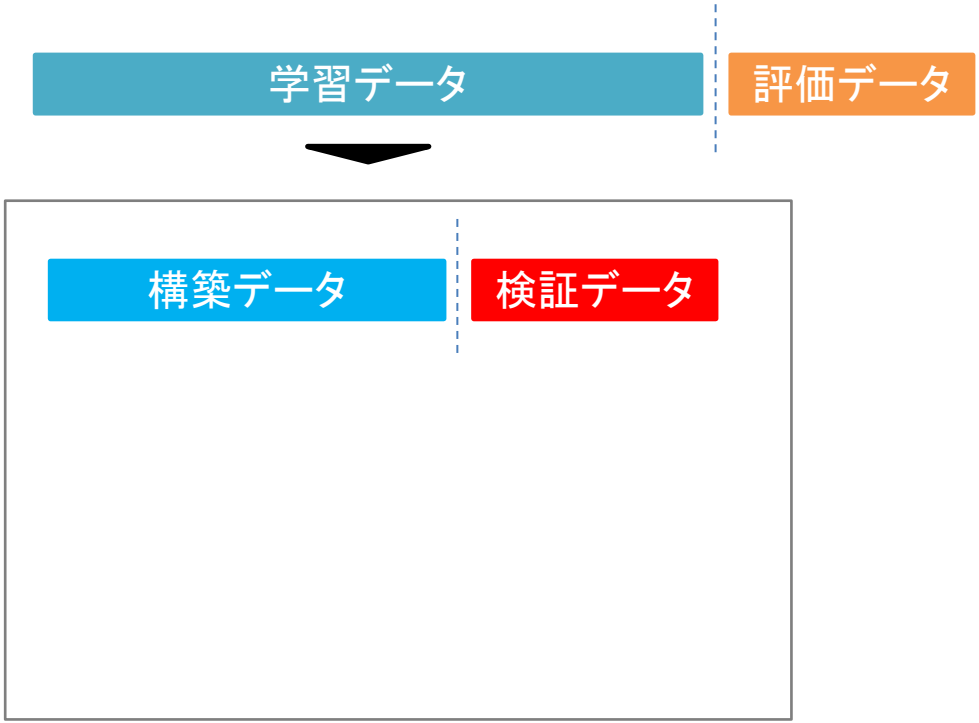
①未知のデータへの評価の為に
学習データと評価データに分ける
※コンテストではここまでしてある

②パラメータをチューニングする為に
学習データを更に分割し、モデル構築用
データとモデル検証用データを作る

③分割したデータを使って良さそうな
パラメータを見つける

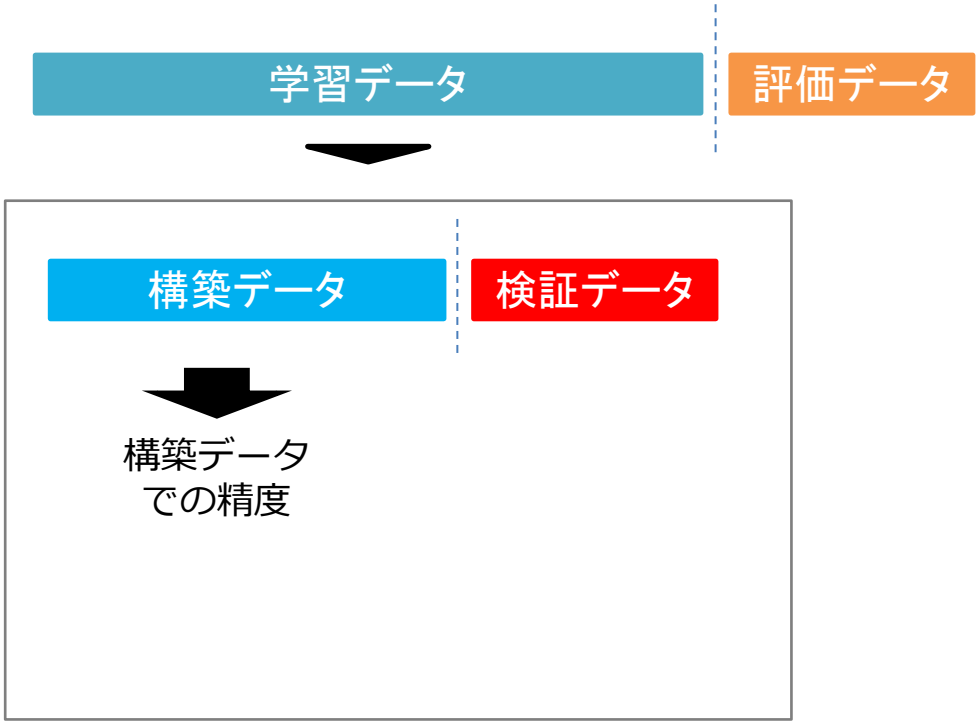
どうやって良いパラメータを見つける？

- 構築データと検証データの精度を比較する



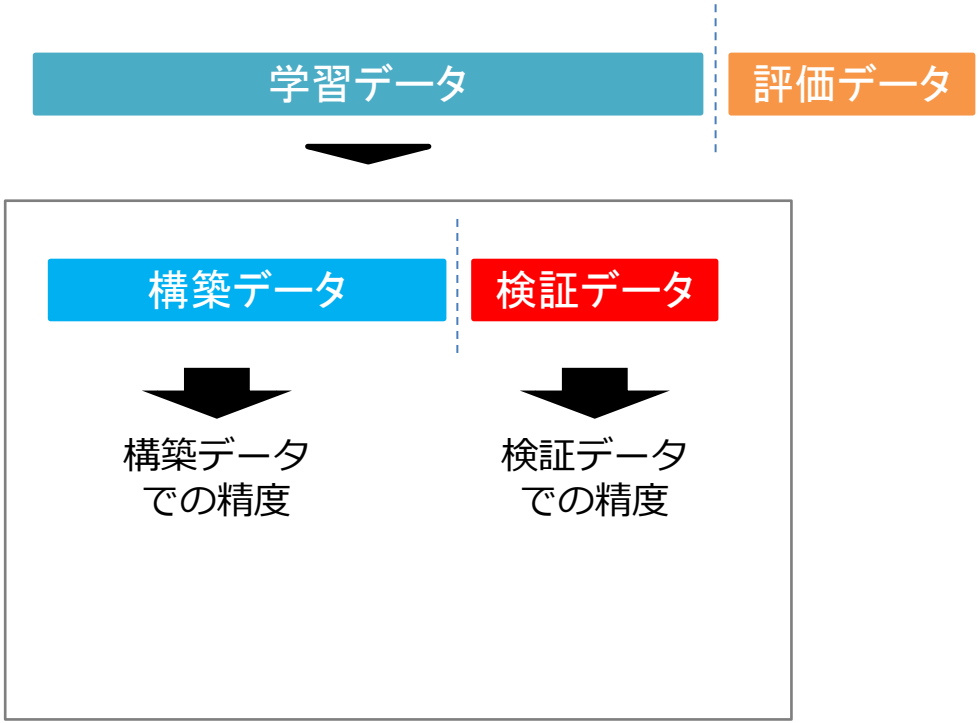
どうやって良いパラメータを見つける？

- 構築データと検証データの精度を比較する



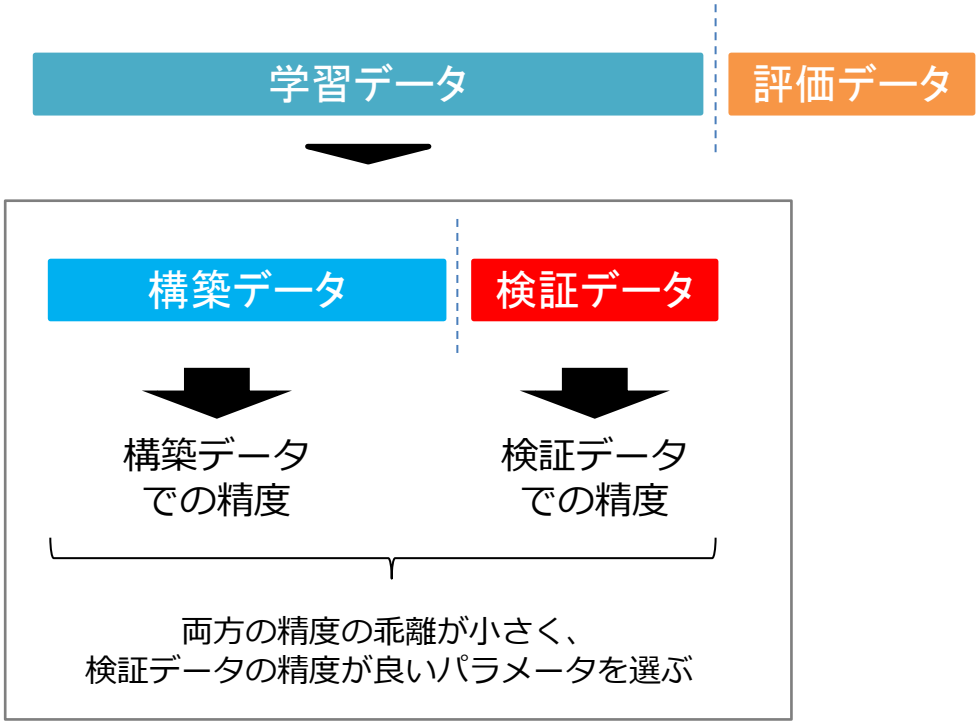
どうやって良いパラメータを見つける？

- 構築データと検証データの精度を比較する



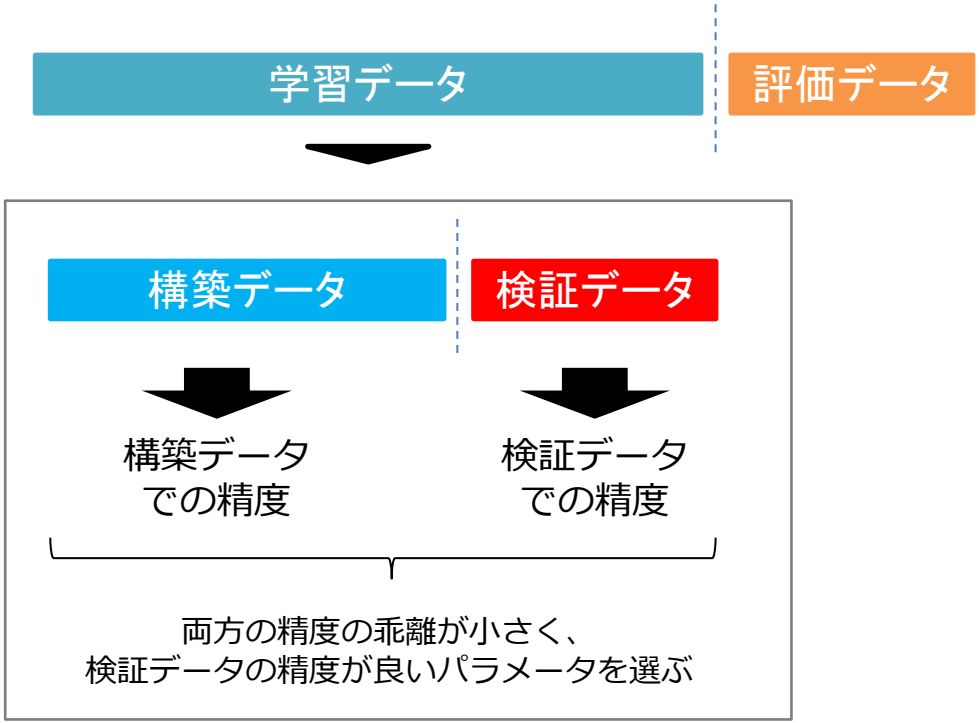
どうやって良いパラメータを見つける？

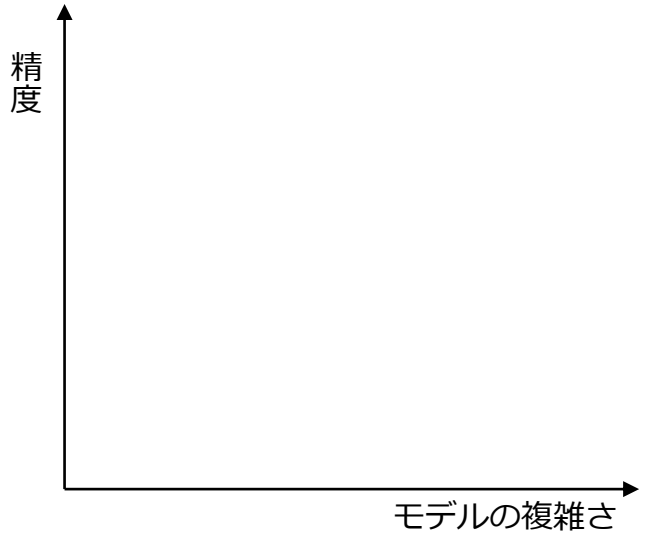
- 構築データと検証データの精度を比較する



どうやって良いパラメータを見つける？

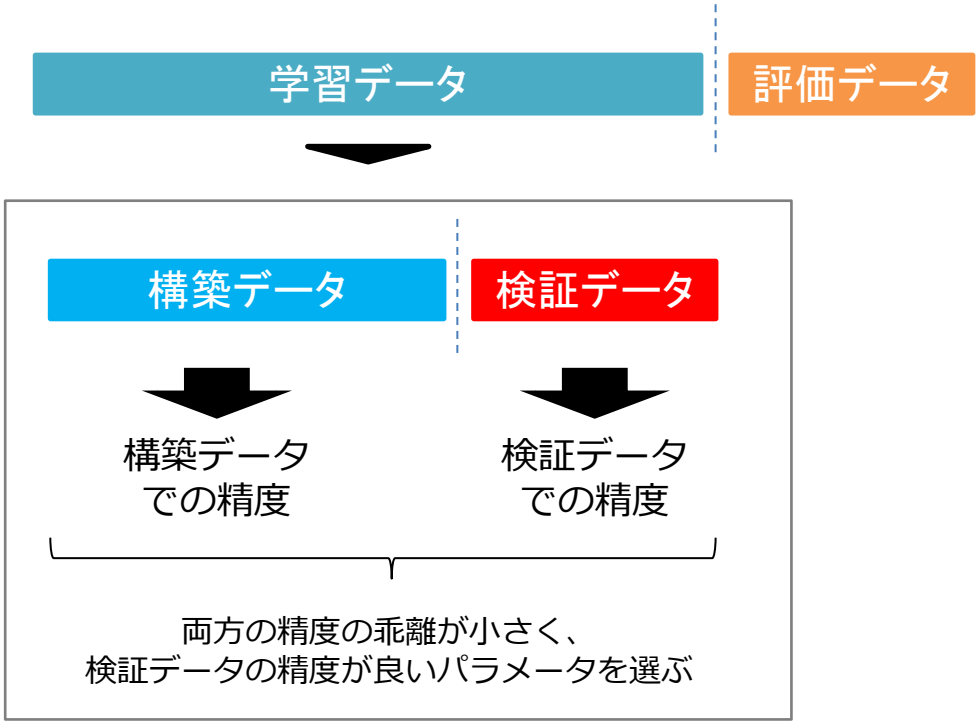
- 構築データと検証データの精度を比較する

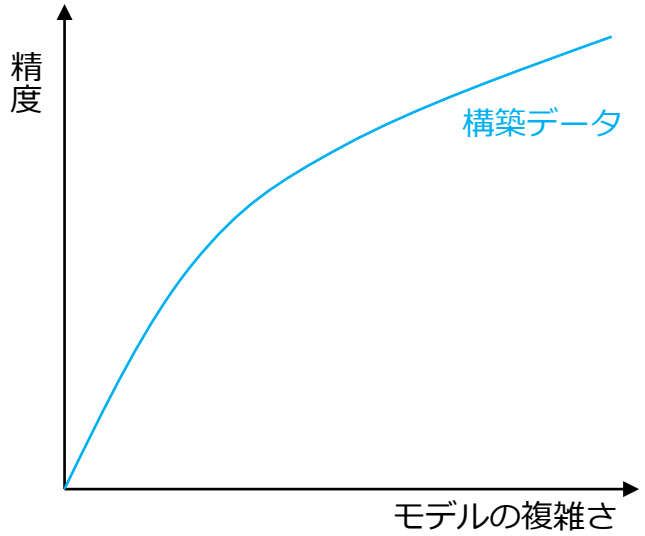




どうやって良いパラメータを見つける？

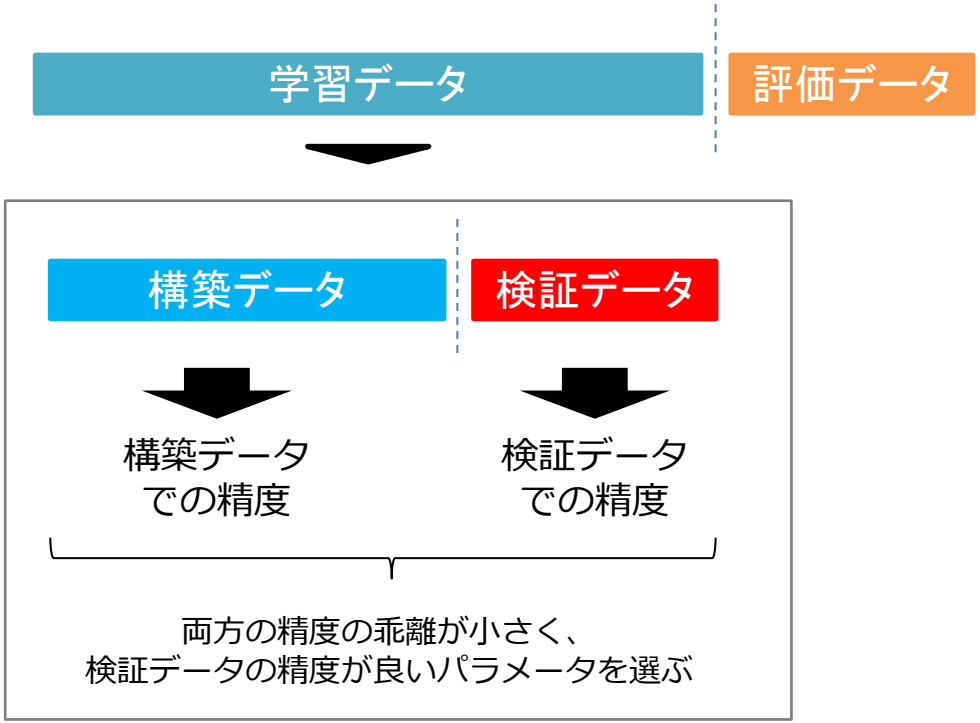
- 構築データと検証データの精度を比較する

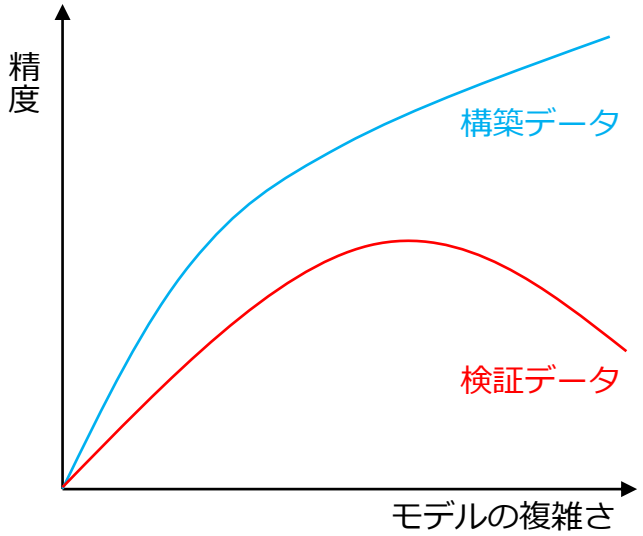




どうやって良いパラメータを見つける？

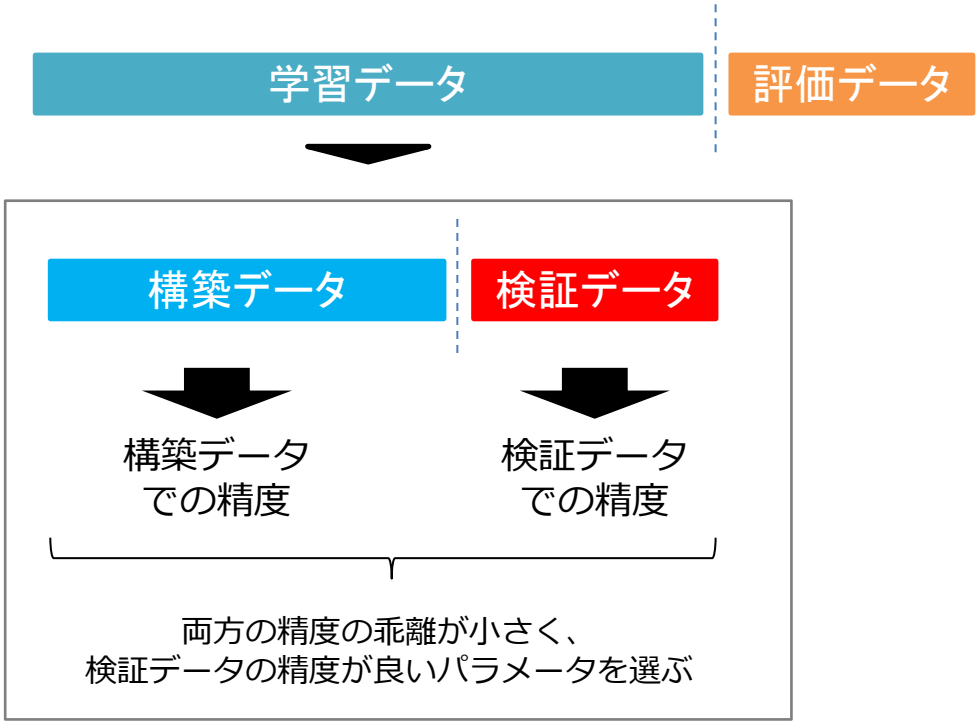
- 構築データと検証データの精度を比較する

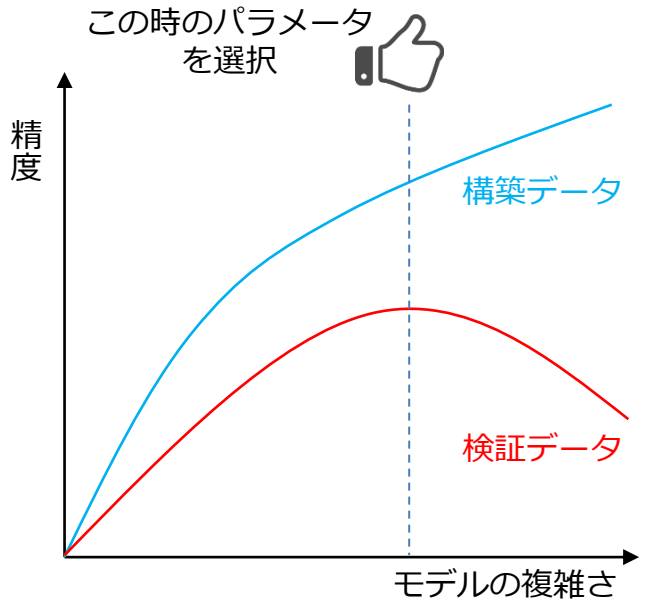




どうやって良いパラメータを見つける？

- 構築データと検証データの精度を比較する





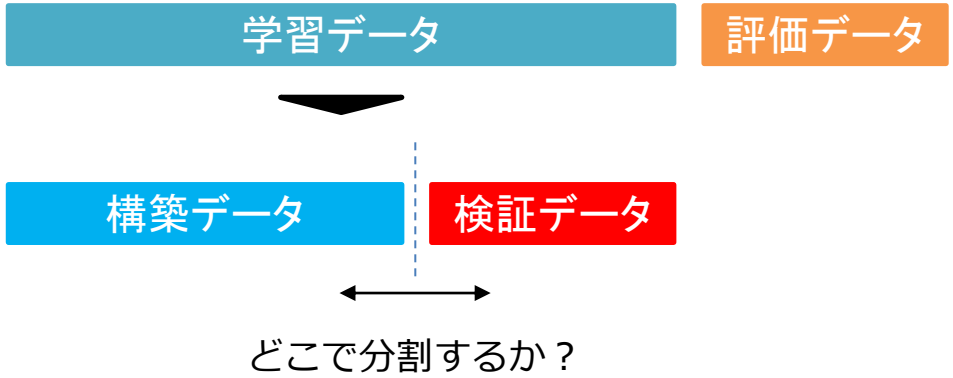
具体的な分割方法

① ホールドアウト（Hold out）
- 単純な方法であるが欠点がある
- データが大量にあれば有効

② クロスバリデーション（cross-validation）
- K分割交差検証とも言われる
- ホールドアウトの欠点を補った手法
- 計算に時間がかかる

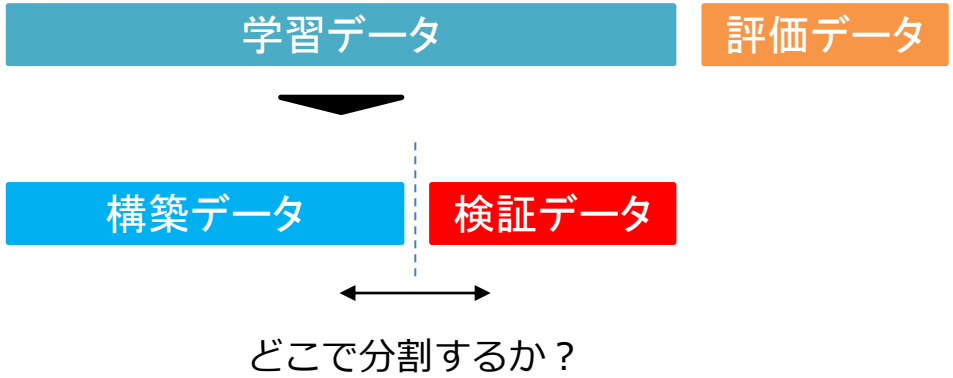
ホールドアウト法

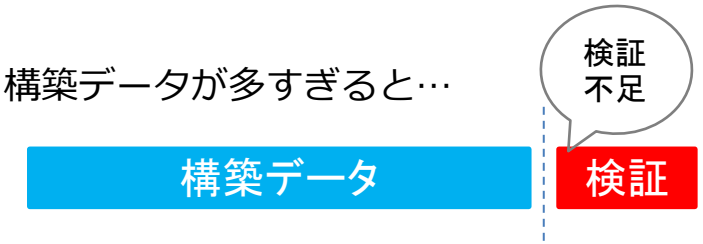
- 学習データを単純に２つに分割する
  - 片方をモデル構築データとし、残りをモデル検証データとする
  - 分割する割合は好きなもので良い（通例、学習:検証=7:3）
  - どこで分割するかで検証精度が大きく変わることもあり、不安定



ホールドアウト法

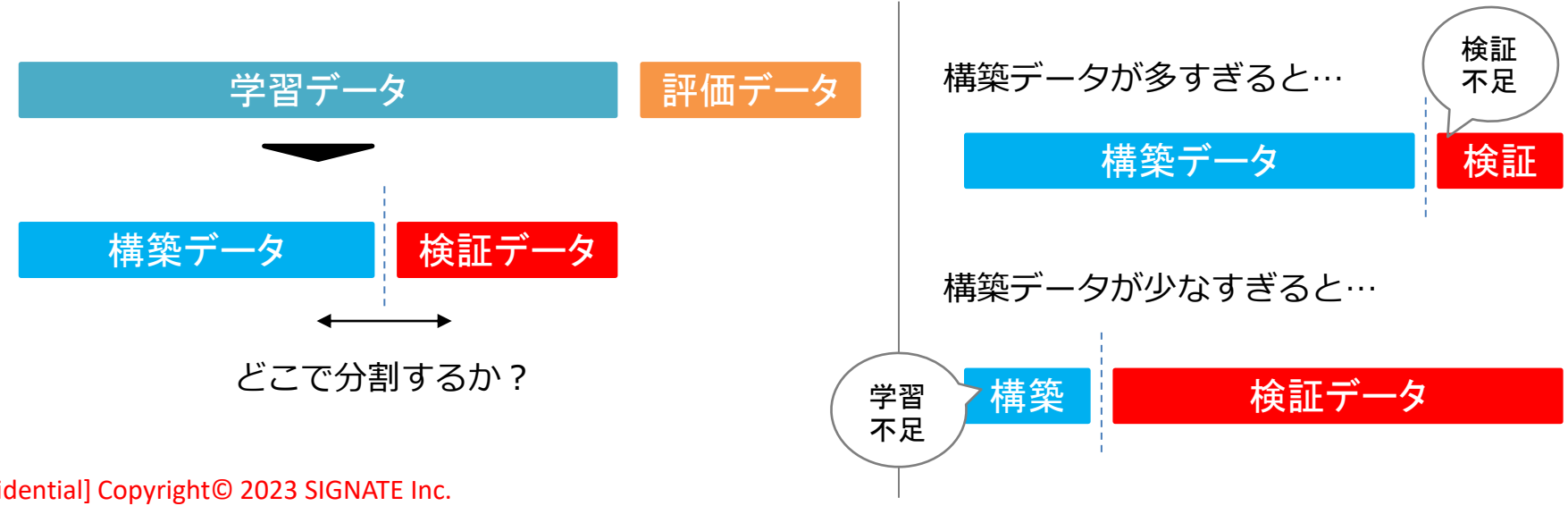
- 学習データを単純に２つに分割する
  - 片方をモデル構築データとし、残りをモデル検証データとする
  - 分割する割合は好きなもので良い（通例、学習:検証=7:3）
  - どこで分割するかで検証精度が大きく変わることもあり、不安定





ホールドアウト法

- 学習データを単純に２つに分割する
  - 片方をモデル構築データとし、残りをモデル検証データとする
  - 分割する割合は好きなもので良い（通例、学習:検証=7:3）
  - どこで分割するかで検証精度が大きく変わることもあり、不安定



クロスバリデーション

- データをK個に分割し、検証をK回繰り返す
  - K-1個をモデル構築データとし、残り１個をモデル検証データとする
  - 上記を他の組み合わせでも行い、K組の平均精度を元にパラメータを決定する

<K=3の場合>

クロスバリデーション

- データをK個に分割し、検証をK回繰り返す
  - K-1個をモデル構築データとし、残り１個をモデル検証データとする
  - 上記を他の組み合わせでも行い、K組の平均精度を元にパラメータを決定する

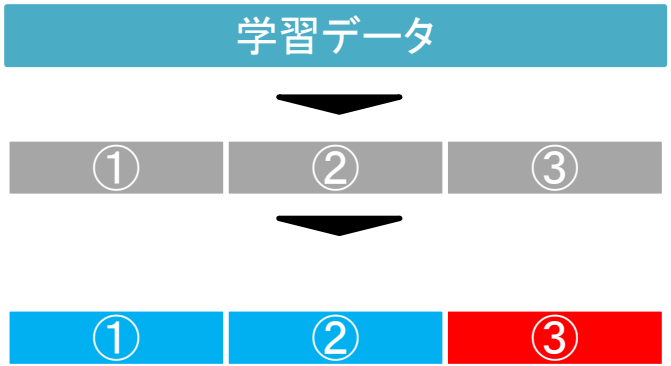
<K=3の場合>

①学習データをK分割する

クロスバリデーション

- データをK個に分割し、検証をK回繰り返す
  - K-1個をモデル構築データとし、残り１個をモデル検証データとする
  - 上記を他の組み合わせでも行い、K組の平均精度を元にパラメータを決定する

<K=3の場合>

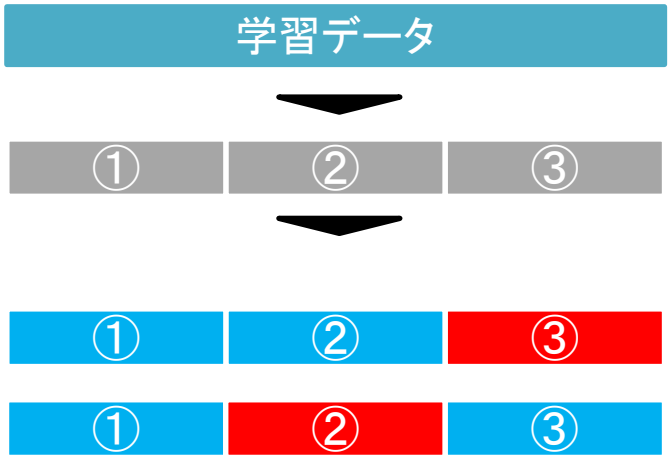


①学習データをK分割する
②K-1個を構築データ、残りを１個を検証データとする

クロスバリデーション

- データをK個に分割し、検証をK回繰り返す
  - K-1個をモデル構築データとし、残り１個をモデル検証データとする
  - 上記を他の組み合わせでも行い、K組の平均精度を元にパラメータを決定する

<K=3の場合>

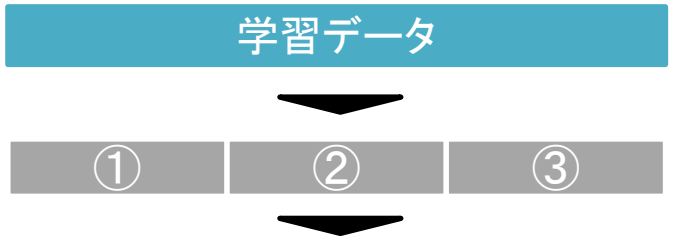


①学習データをK分割する
②K-1個を構築データ、残りを１個を検証データとする

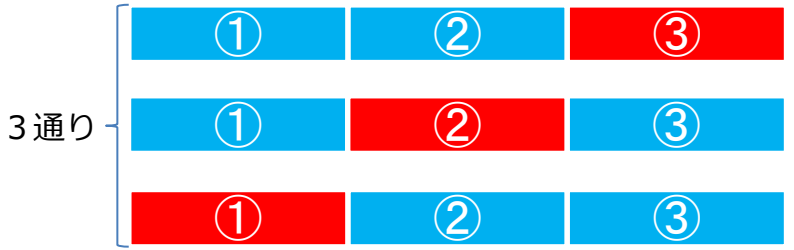
クロスバリデーション

- データをK個に分割し、検証をK回繰り返す
  - K-1個をモデル構築データとし、残り１個をモデル検証データとする
  - 上記を他の組み合わせでも行い、K組の平均精度を元にパラメータを決定する

<K=3の場合>



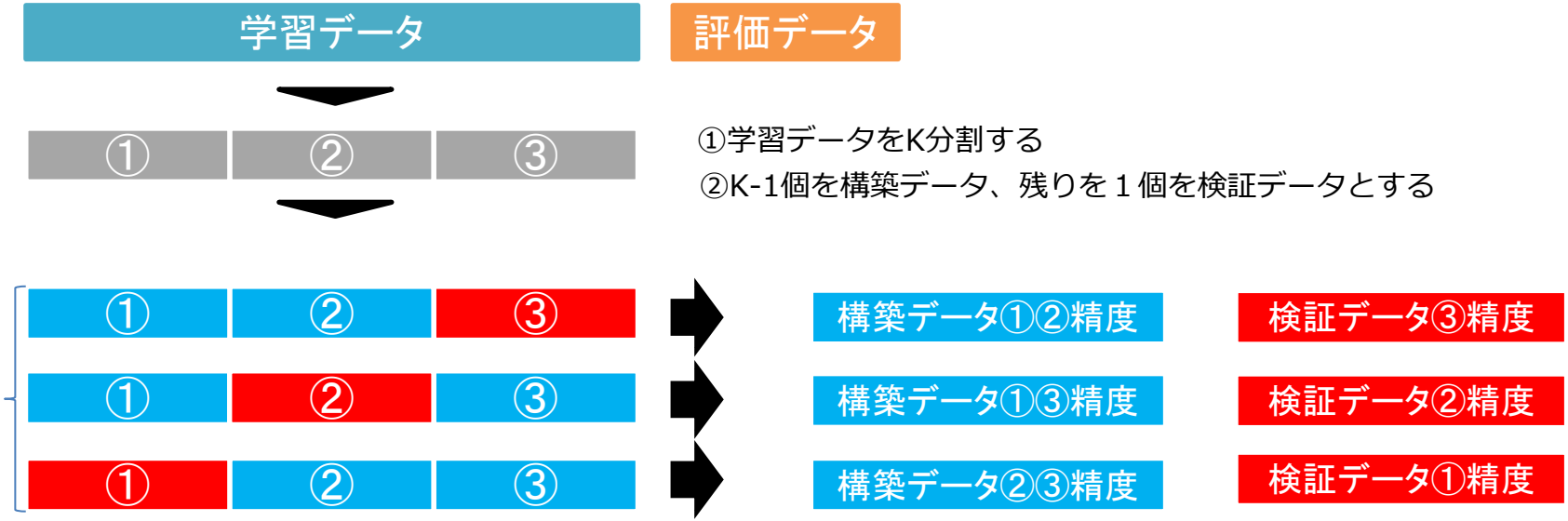
①学習データをK分割する
②K-1個を構築データ、残りを１個を検証データとする



クロスバリデーション

- データをK個に分割し、検証をK回繰り返す
  - K-1個をモデル構築データとし、残り１個をモデル検証データとする
  - 上記を他の組み合わせでも行い、K組の平均精度を元にパラメータを決定する

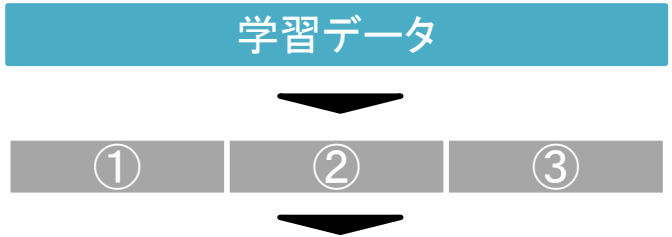
<K=3の場合>



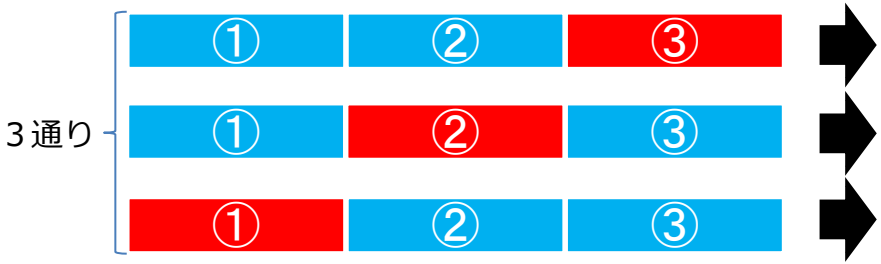
クロスバリデーション

- データをK個に分割し、検証をK回繰り返す
  - K-1個をモデル構築データとし、残り１個をモデル検証データとする
  - 上記を他の組み合わせでも行い、K組の平均精度を元にパラメータを決定する

<K=3の場合>



①学習データをK分割する
②K-1個を構築データ、残りを１個を検証データとする

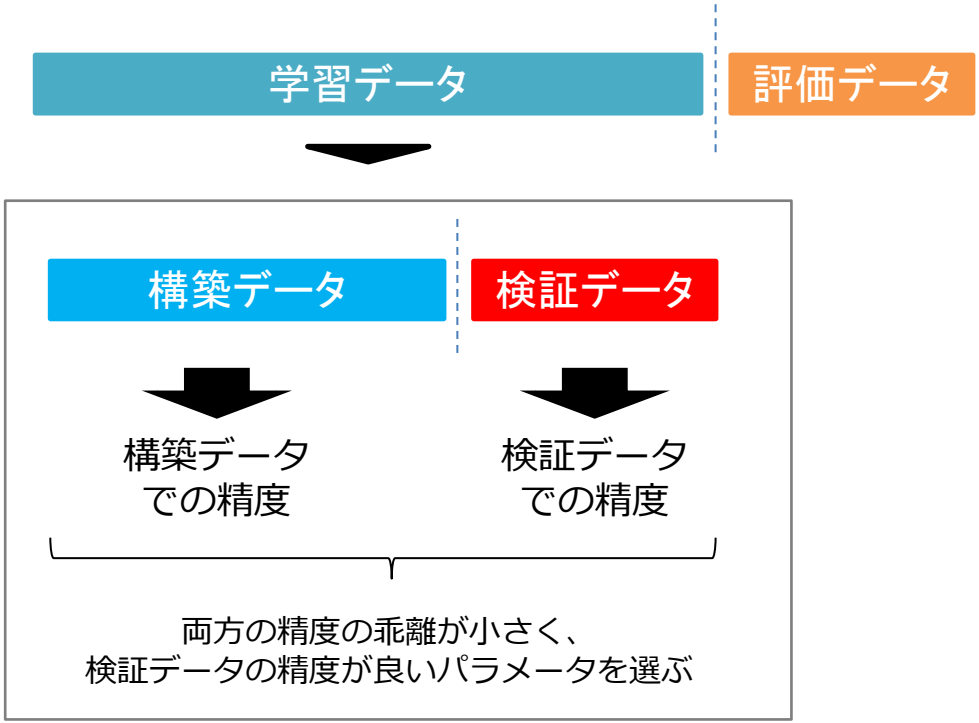


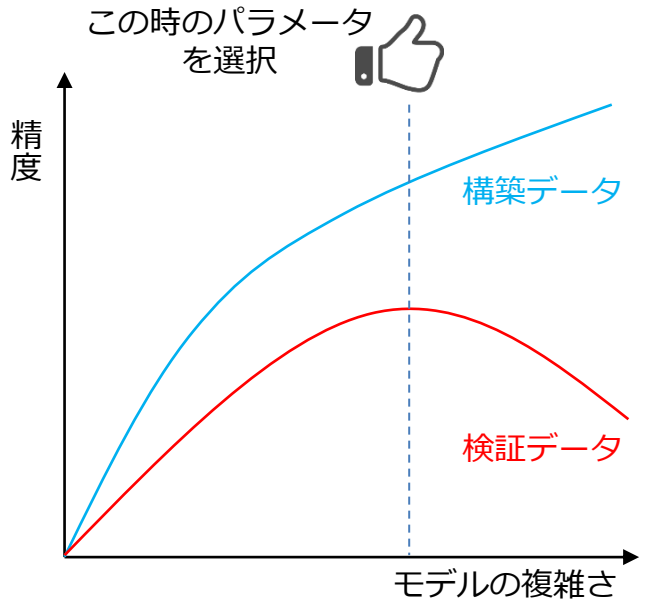
平均をとり、
構築データへの
精度とする

平均をとり、
検証データへの
精度とする

どうやって良いパラメータを見つける？

- 構築データと検証データの精度を比較する





プログラムの書き方

▼ホールドアウト

```
import numpy as np
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)

from sklearn.linear_model import LinearRegression as LR
model = LR()
model.fit(X_train, y_train)
pred = model.predict(X_test)

from sklearn.metrics import mean_squared_error as MSE
print(np.sqrt(MSE(y_test, pred)))
```

▼クロスバリデーション

```
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression as LR

model = LR()
cv = cross_val_score(model, X, y, cv=3, scoring='neg_mean_squared_error')
cv = np.fabs(cv) ** 0.5

print(cv, np.mean(cv))
```

Scoringの引数
・MSE：‘neg_mean_squared_error’
・AUC：‘roc_auc’

下記で選択できる評価関数一覧を出力可

```
import sklearn
print(sklearn.metrics.SCORERS.keys())
```

※重回帰モデルを利用し、評価関数はRMSEを想定した場合

2. Google Colaboratory

Google Colaboratoryとは？

Googleが提供するnotebook

・環境設定不要でインターネットがあれば使える

・**GPU**が無料で利用できる

・共有が簡単にできる

Google Colaboratoryとは？

▼チュートリアルページ

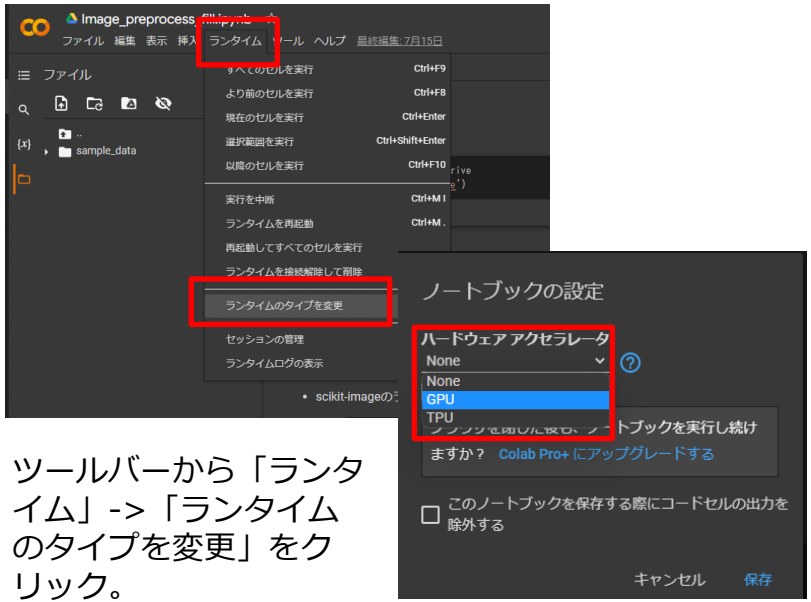
https://colab.research.google.com/notebooks/welcome.ipynb?hl=ja

▼基本的な手順

1	2	3	4
GoogleDriveにデータを配置	**ランタイムの設定**	**GoogleDriveをマウント**	**環境設定、分析を開始する**
ご自身のGoogleアカウントにログインの上、Drive上にデータを配置。 学習データファイル数が多い場合にはzipのままアップすることを推奨。	Colabを使う意義として一番重要なところ。 デフォルトではCPUとなっているので、GPUもしくはTPUに最初に設定しておくこと。	マウントすることで、自分のGoogle driveのファイルを読み書きすることができるようになる。 ファイルパスがやや癖があるので注意。	そのままでも利用可能だが、ライブラリのバージョンの調整などが必要であれば、先に実施しておく。 問題なければそのまま分析を開始。

Google Colaboratoryとは？

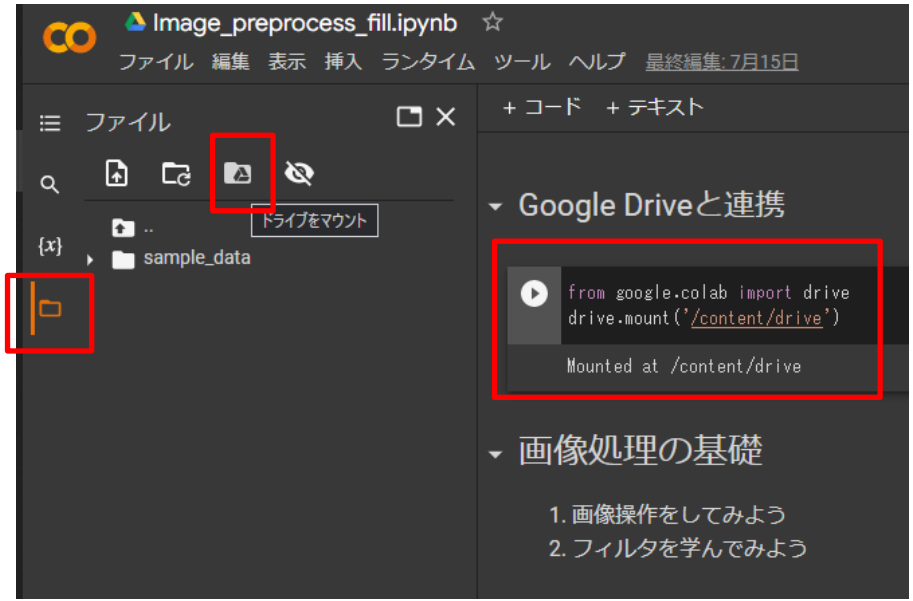
ランタイムの設定



ツールバーから「ランタイム」->「ランタイムのタイプを変更」をクリック。

ハードウェアアクセラレータをGPUなどに変更。

Driveのマウント



左のメニューバーの中の一番したのファイルを選び、左から3番目のアイコンをクリック。

次のセルが自動挿入されるので、こちらを実行すればマウントされる。

Tensorflowのサンプルコード

https://www.tensorflow.org/tutorials/quickstart/beginner?hl=ja

Google Colaboratoryの制約

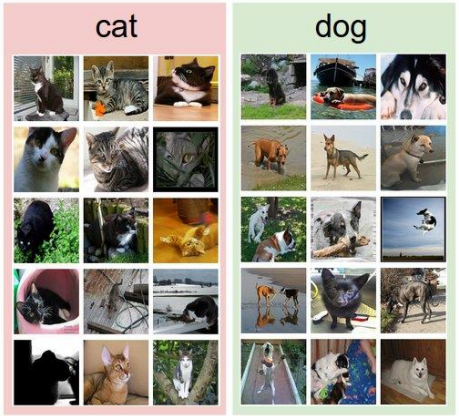
- **90分ルール**
  - アイドル状態が90分続くと停止
    - アイドル＝ブラウザ閉じる、PCスリープ等
  - PCスリープでも停止になるので注意

- **12時間ルール**
  - 連続使用は12時間まで
    - セッションの有無にかかわらず、起動から１２時間経過すると、リセットされる。
  - 学習等、実行が長い処理は定期的に中間ファイルを保存するといったリスクヘッジをとる必要がある

3. 画像分析のキホンの「キ」

画像認識の一般的なタスク

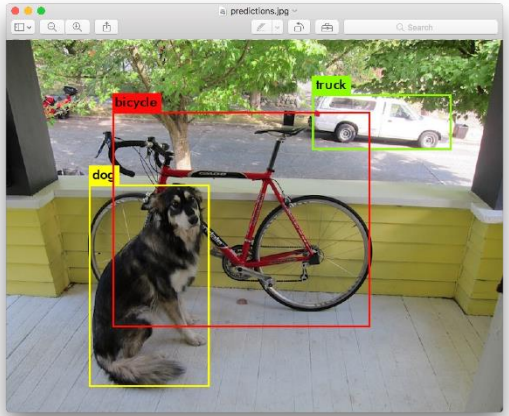
▼画像分類



【引用】http://cs231n.github.io/classification

画像に
何が写っているか？

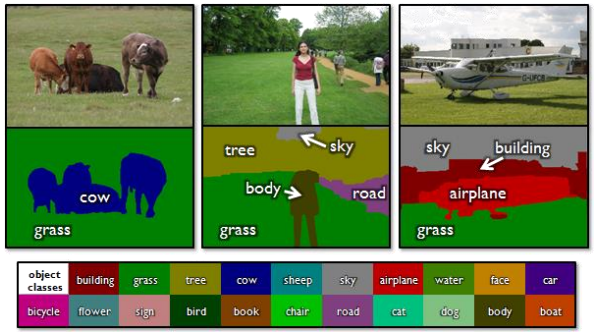
▼画像検出



【引用】https://pjreddie.com/darknet/yolo/

画像に“どこに”
何が写っているか？

▼画像セグメンテーション



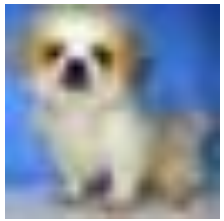
【引用】http://jamie.shotton.org/work/research.html

画像にどこに
何が写っていて”その領域”
はどこまでがそうか？

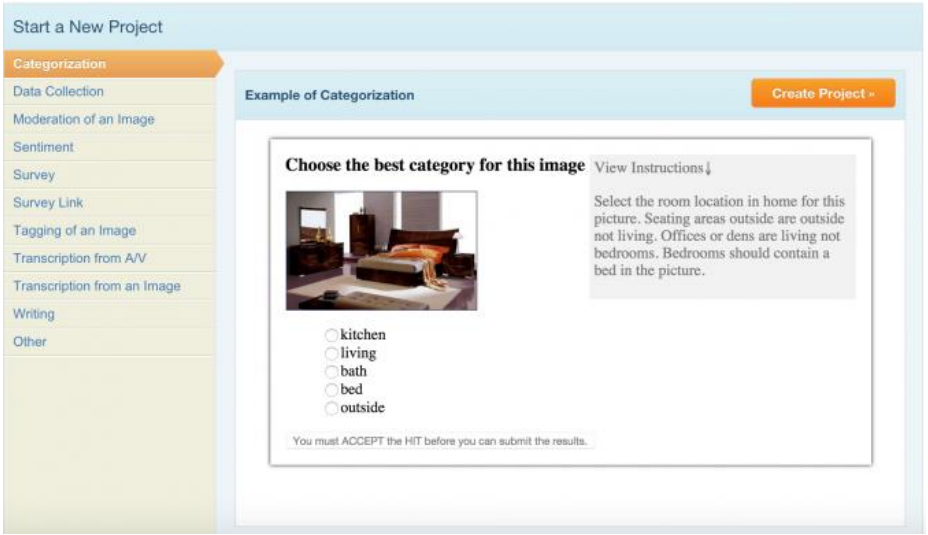
画像分類（classification）

- 画像に付与されるラベルを推定する
- 教師データ：**ラベル**
- ImageNetと呼ばれるデータセットでは約1,400万画像

“cat”



“dog”

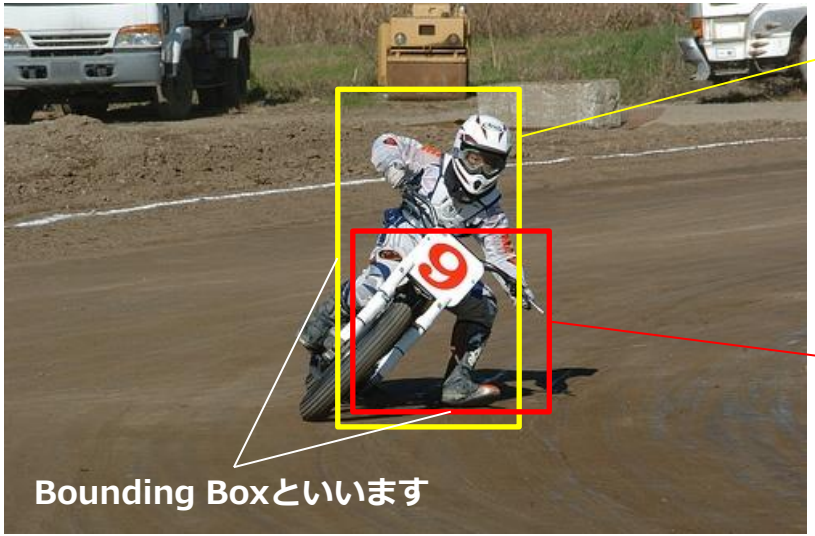


【引用】https://cloudacademy.com/blog/machine-learning-datasets-mechanical-turk/

画像検出（detection）

- 物体の場所とその物体名を推定する
- 教師データ：**矩形（4点座標）とラベル**
- 例）Pascal Vocデータ

★画像データとアノテーションデータがセットで与えられる



アノテーションデータ

```
<object>
  <name>person</name>                 物体名
  <pose>Frontal</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>                            矩形
    <xmin>183</xmin>                  座標
    <ymin>70</ymin>
    <xmax>330</xmax>
    <ymax>253</ymax>
  </bndbox>
</object>
<object>
  <name>motorbike</name>              物体名
  <pose>Frontal</pose>
  <truncated>0</truncated>
  <difficult>0</difficult>
  <bndbox>                            矩形
    <xmin>185</xmin>                  座標
    <ymin>110</ymin>
    <xmax>336</xmax>
    <ymax>251</ymax>
  </bndbox>
</object>
```

画像セグメンテーション（segmentation）

- 物体の領域まで推定する
- 教師データ：**領域塗布済みの画像**
- 例）Pascal Vocデータ

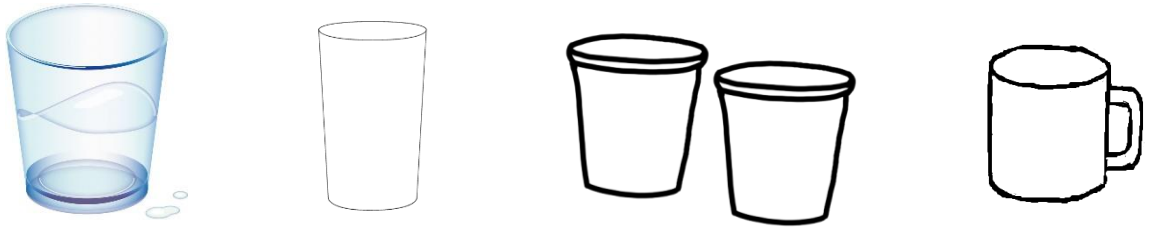
Input Image

Object Segmentation（物体に色付け）

Class Segmentation（属性毎に色分け）

ところで・・・

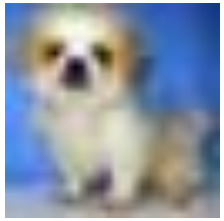
- ヒトはどのように物体を認識するか？
  - 観測された物体の**“形の特徴”**
  - その周辺にあるコンテキスト



機械に認識させる為の手法も基本的には同じ

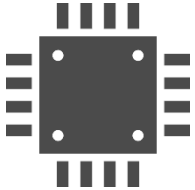
手法による特徴量抽出の違い

一般的な機械学習



input

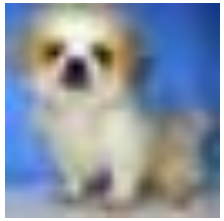
人手で設計した
特徴量抽出



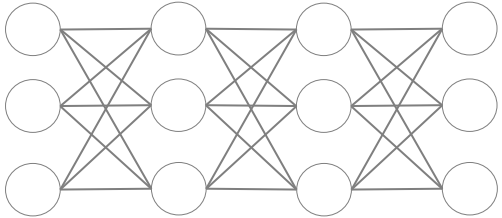
分類器

分類結果の出力

ディープラーニング



input



特徴量抽出+分類器
“構造のみ”を決定し、大量のinput&outputで
重要な特徴量を学習／分類

分類結果の出力

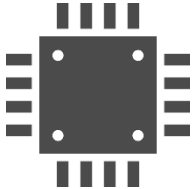
手法による特徴量抽出の違い

入力画像に対する
ケアが大変重要

一般的な機械学習

input

人手で設計した
特徴量抽出

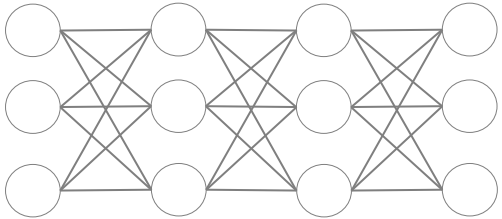


分類器

分類結果の出力

ディープラーニング

input



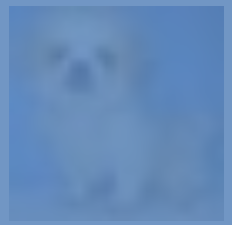
特徴量抽出+分類器
“構造のみ”を決定し、大量のinput&outputで
重要な特徴量を学習／分類

分類結果の出力

手法による特徴量抽出の違い

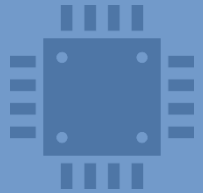
入力画像に対する
ケアが大変重要

一般的な機械学習



input

人手で設計した
特徴量抽出

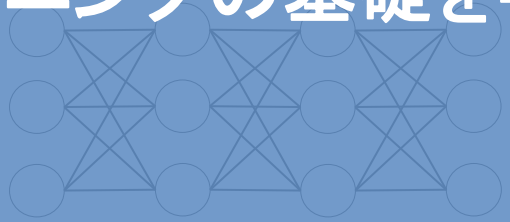


分類器

分類結果の出力

ディープラーニング

input



特徴量抽出+分類器
“構造のみ”を決定し、大量のinput&outputで
重要な特徴量を学習／分類

分類結果の出力

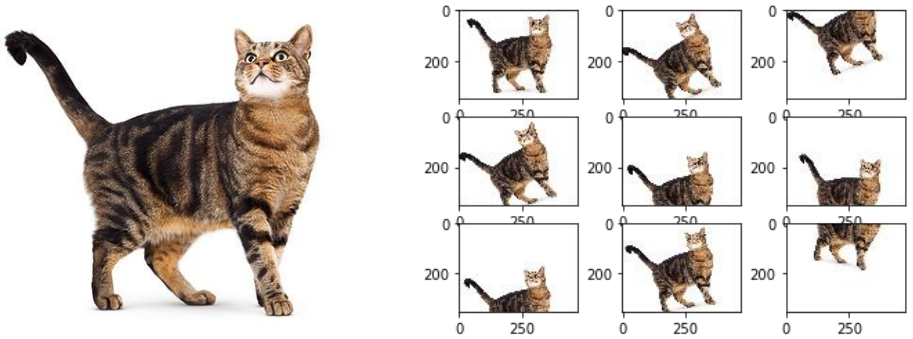
まずは画像処理の基本から学びましょう
続いて、ディープラーニングの基礎を学んでいきましょう

画像処理とは？

- **画像処理の目的**
  - 画像を操作・分類し、画像から何らかの情報を得ること
  - E.g) 画像分類、画像検索、画像認識・・・

- **機械学習では…**
  - 画像の特徴を目立たせる処理
  - 画像のバラエティを増やす

▼バラエティを増やす（Data Argumentation）



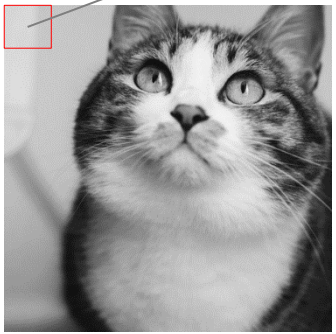
【引用】https://towardsdatascience.com/image-augmentation-for-deep-learning-histogram-equalization-a71387f609b2

画像データってなんだ？

- 行列（多次元配列）データで表現される

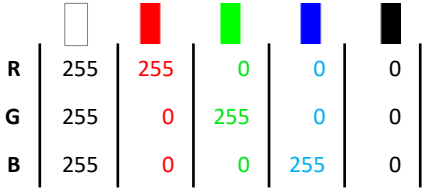
▼白黒画像（グレースケール）

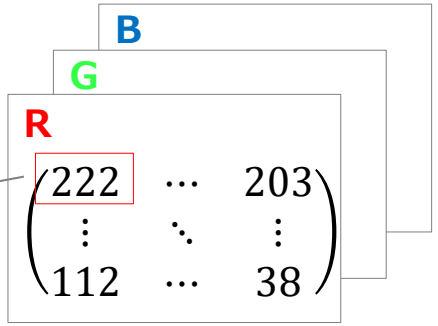
$$\begin{pmatrix} 216 & 216 & \cdots & 188 \\ 216 & 216 & & 188 \\ \vdots & & \ddots & \vdots \\ 111 & 112 & \cdots & 32 \end{pmatrix}$$



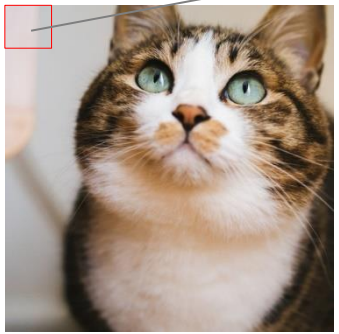
グレースケールの
1 ピクセルは
1 つの値で表現される

▼カラー画像（RGB）





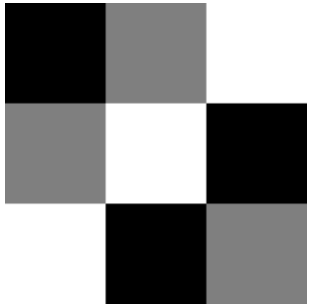
$$\begin{pmatrix} 222 & \cdots & 203 \\ \vdots & \ddots & \vdots \\ 112 & \cdots & 38 \end{pmatrix}$$



カラー画像の1ピクセルは
R,G,Bの3つの値で表現される

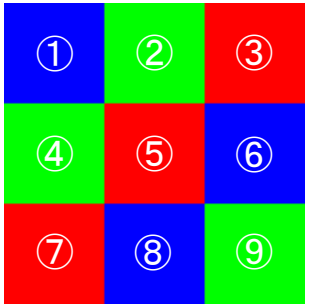
配列表現で表現すると

▼白黒画像（グレースケール）



```
array([[  0, 127, 255],    ←1行目
       [127, 255,   0],    ←2行目
       [255,   0, 127]])   ←3行目
```

▼カラー画像（RGB）




```
                R   G   B
                ↓   ↓   ↓
array([[[  0,   0, 255],    ←①
        [  0, 255,   0],    ←②
        [255,   0,   0]],   ←③

       [[  0, 255,   0],    ←④
        [255,   0,   0],    ←⑤
        [  0,   0, 255]],   ←⑥

       [[255,   0,   0],    ←⑦
        [  0,   0, 255],    ←⑧
        [  0, 255,   0]]])  ←⑨
```

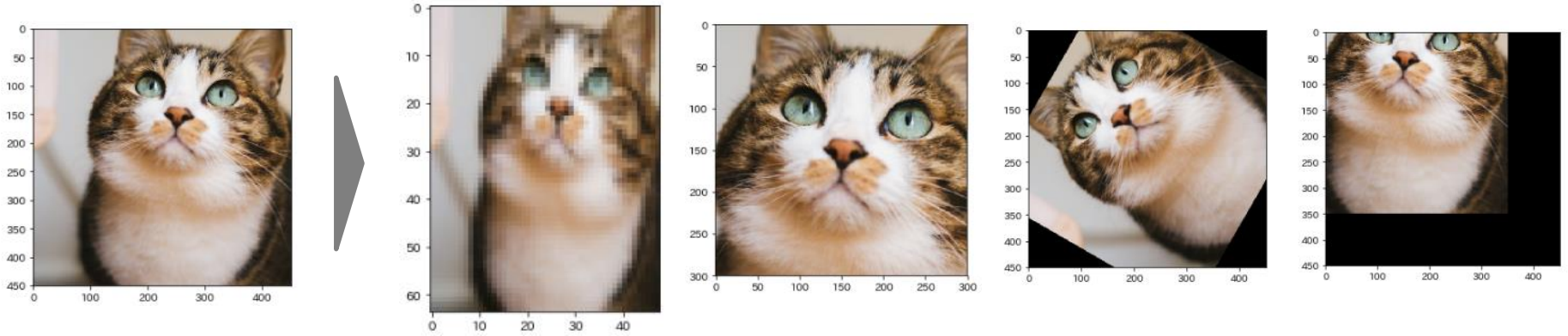
画像処理をpythonでやる場合にはnumpyという
数値ライブラリを使いこなす必要があります。

Pythonにおける画像操作

- **便利なライブラリ**
  - Numpy, Pillow, OpenCV…
  - 今回はNumpyとScikit-imageを使います

- **画像操作**

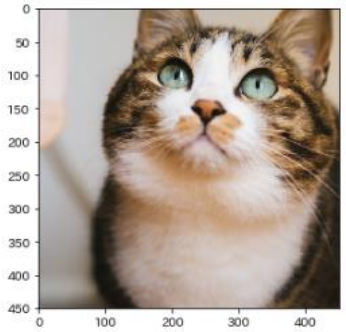
オリジナル　リサイズ　クロッピング　回転　平行移動



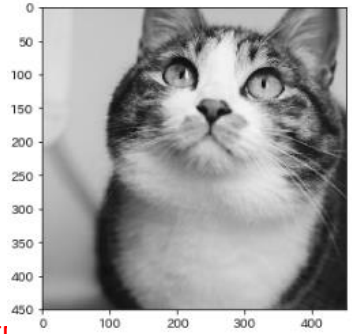
基本的な画像操作（１）

▼画像を読み込む

```
img = io.imread("image.png")
io.imshow(img)
```

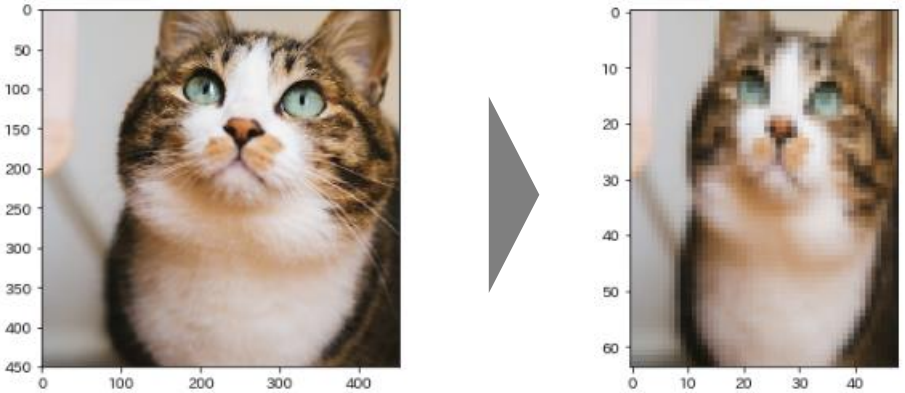


```
img = io.imread("image.png",as_gray=True)
io.imshow(img)
```



▼画像のサイズを変える

```
im_resized = transform.resize(img, output_shape=(64,48),
                              mode="reflect",anti_aliasing=True)
io.imshow(im_resized)
```



基本的な画像操作（２）

▼画像からRGB情報を取得

```
img.shape
```

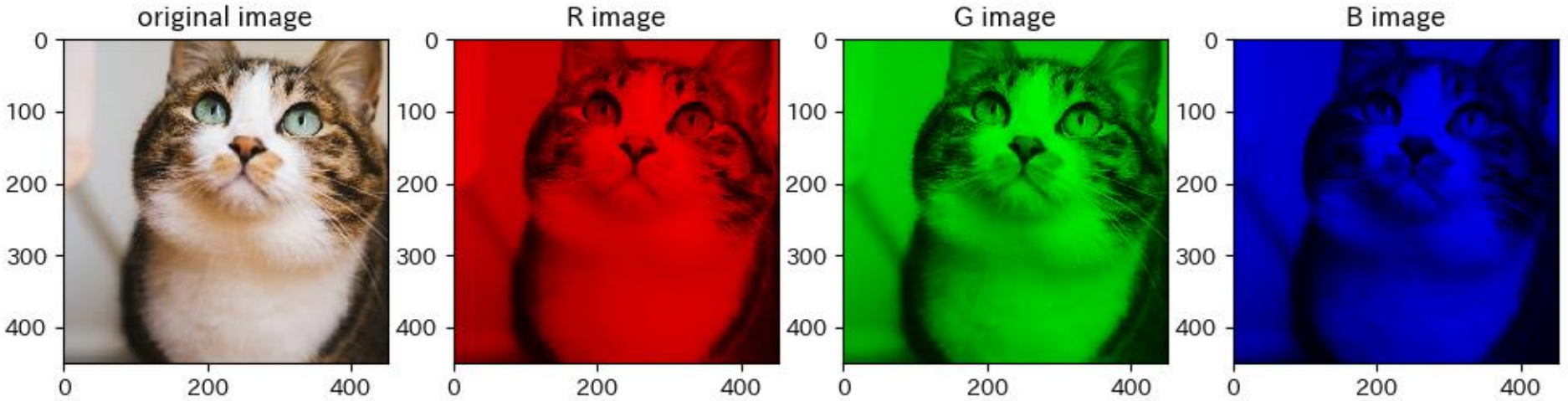
```
(450, 452, 3)
```

縦軸 横軸 RGB

```
io.imshow(img[:,:,0])
```

```
io.imshow(img[:,:,1])
```

```
io.imshow(img[:,:,2])
```

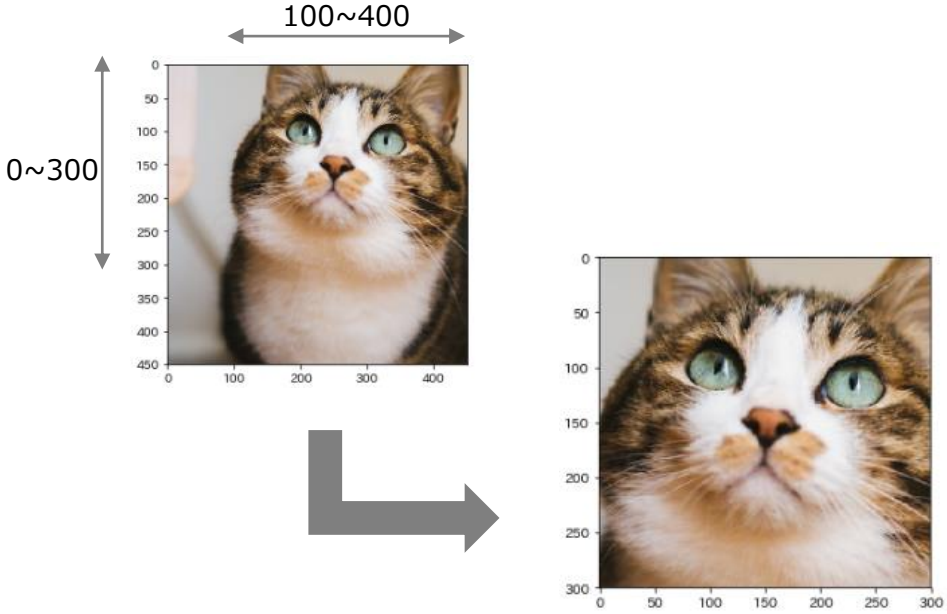


※プログラムでは上記コマンド実行時にはグレースケールで表示されます

基本的な画像操作（３）

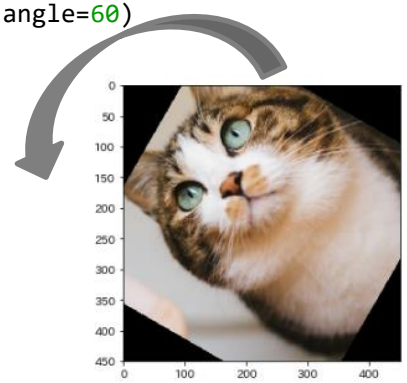
▼画像をクロッピング

```
                縦軸     横軸   RGB
img_crop = img[0:300,100:400,:]
io.imshow(img_crop)
```

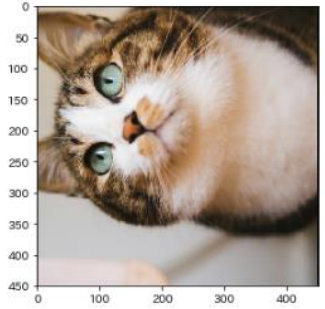


▼画像を回転

```
img_rot = transform.rotate(img,angle=60)
io.imshow(img_rot)
```



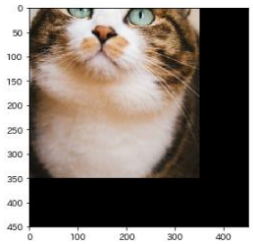
```
img_rot = transform.rotate(img,angle=90)
io.imshow(img_rot)
```



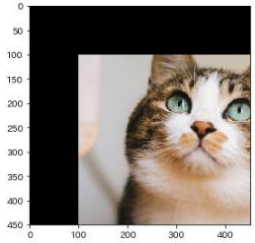
基本的な画像操作（４）

▼画像を平行移動

```
trans = transform.AffineTransform(translation=(100,100))
img_shift = transform.warp(img, inverse_map=trans)
io.imshow(img_shift)
```



```
trans = transform.AffineTransform(translation=(-100,-100))
img_shift = transform.warp(img, inverse_map=trans)
io.imshow(img_shift)
```



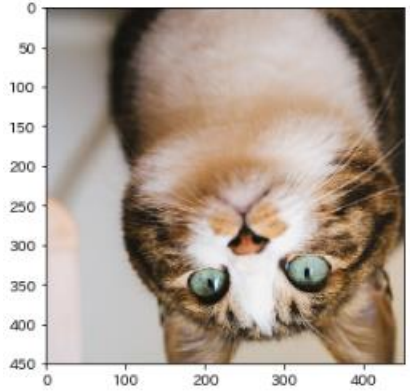
▼画像を反転

```
a = np.array([1,2,3,4])
a[::-1]
```

```
array([4, 3, 2, 1])
```

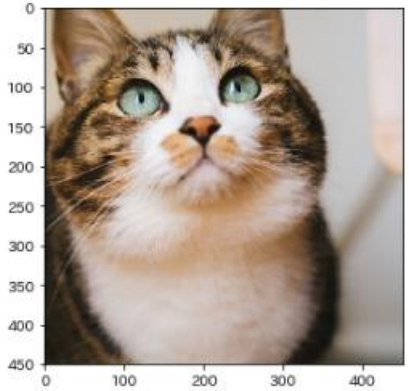
縦軸を反転

```
io.imshow(img[::-1,:,:])
```



横軸を反転

```
io.imshow(img[:,::-1,:])
```

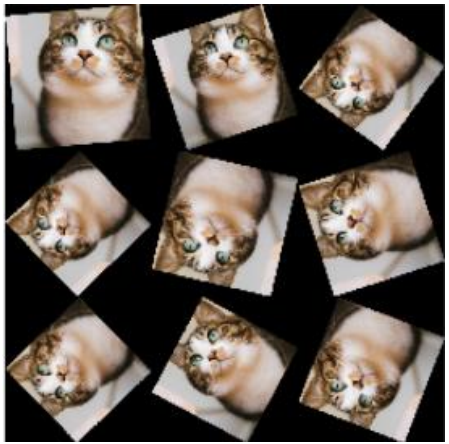


Data Augmentation

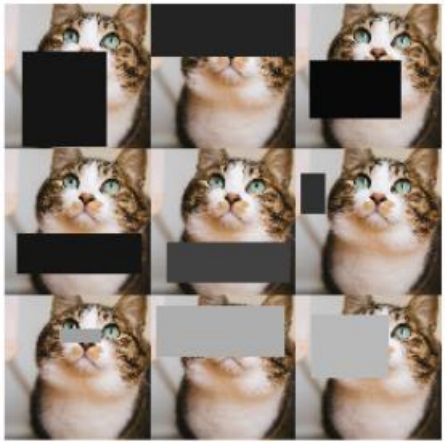
- 人工的にオリジナル画像を加工し、複数枚の画像を生成
  - 前処理の１つ
  - DeepLearningではデータ量（多様性）が精度に影響

Random Cropping

Random Rotation

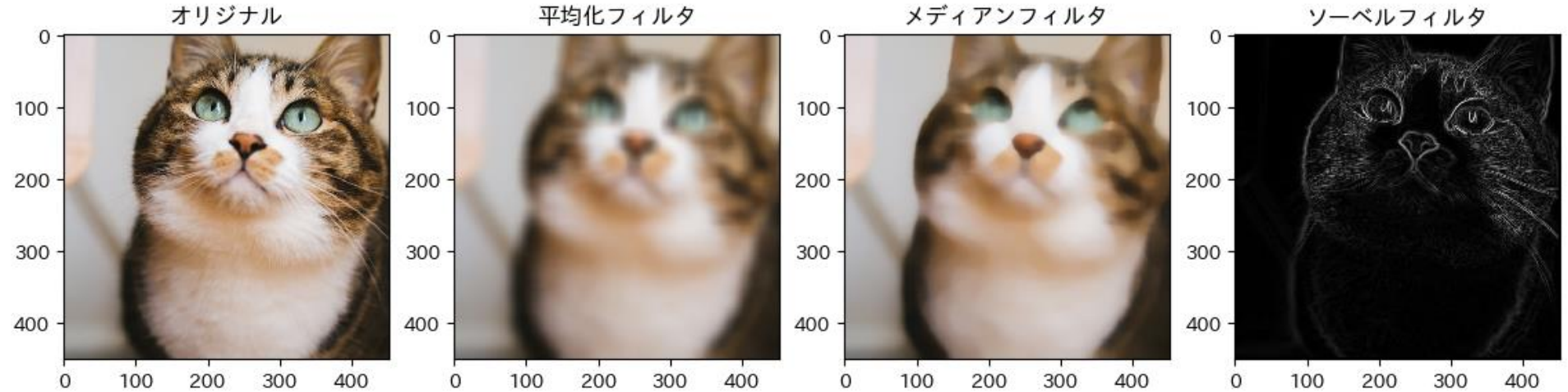


Random Erasing



空間フィルタリングとは？

- 画像内の注目画素及びその周囲の画素を含めて何らかの計算を行う画像処理のこと
- 画像をぼかしたり、輪郭を浮き彫りにすることができる

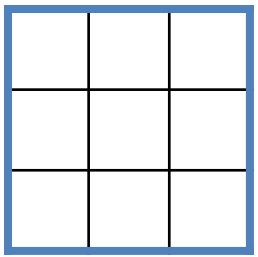


フィルタリングの仕組み

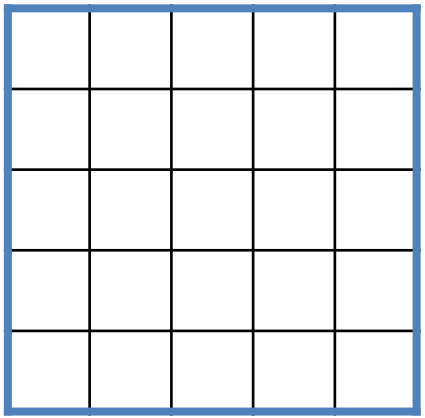
基本①：フィルタを使う

フィルタリングをする際には
フィルタ（カーネル）と呼ばれる
ものを使う。

3×3フィルタ



5×5フィルタ



フィルタ内の値で画像に対する効果が変わる

平均化フィルタ

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

ソーベルフィルタ

-1	0	1
-2	0	2
-1	0	1

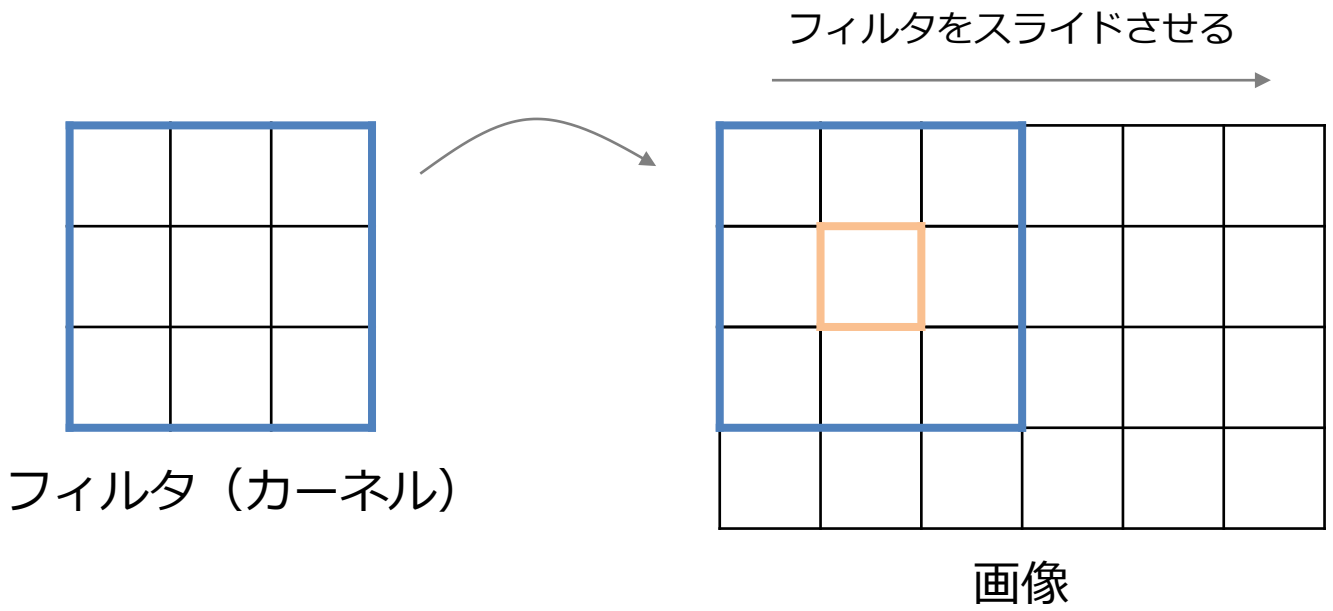
x方向

-1	-2	-1
0	0	0
1	2	1

y方向

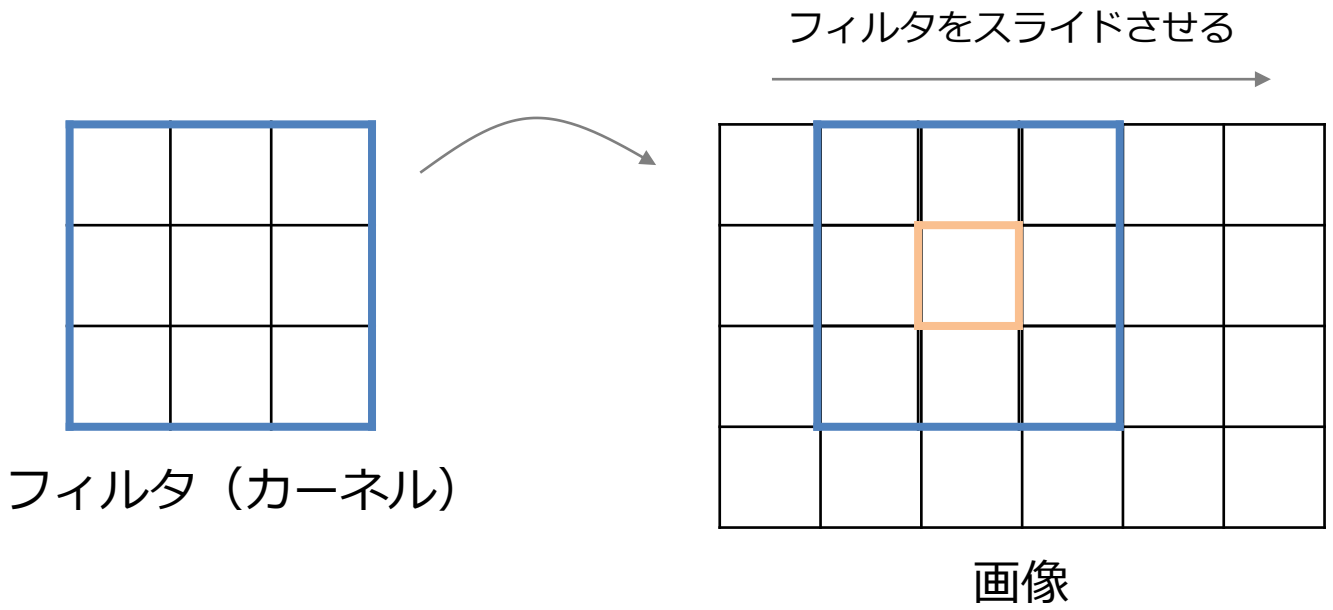
フィルタリングの仕組み

基本②：フィルタをスライドさせる



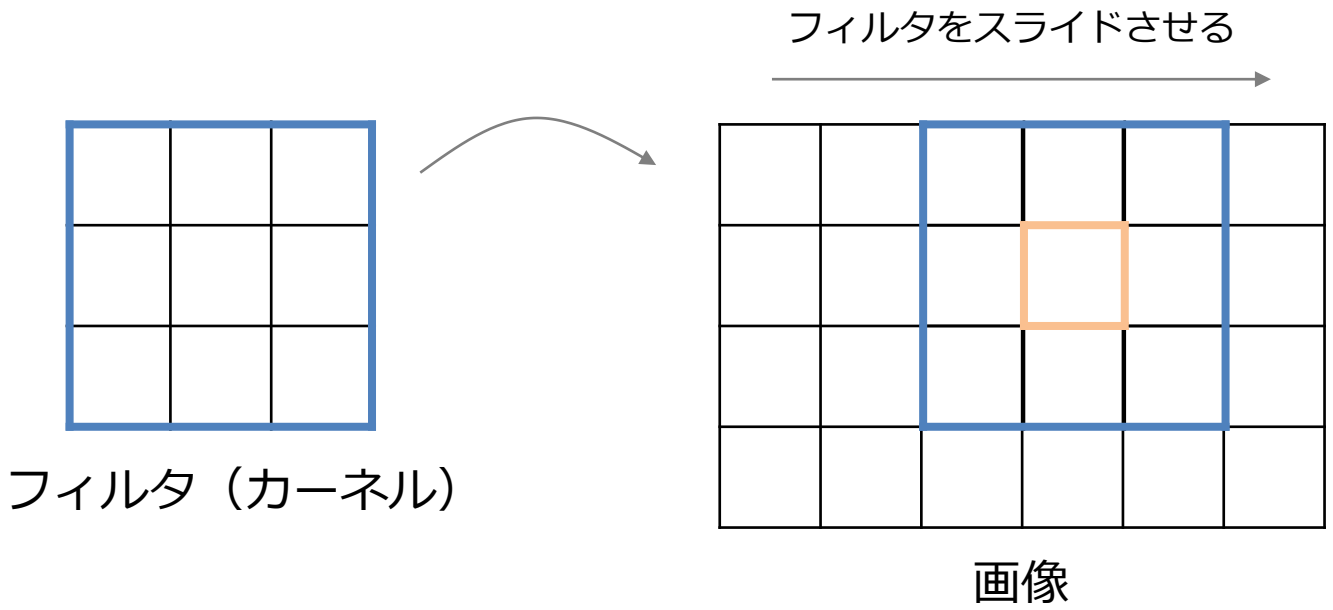
フィルタリングの仕組み

基本②：フィルタをスライドさせる



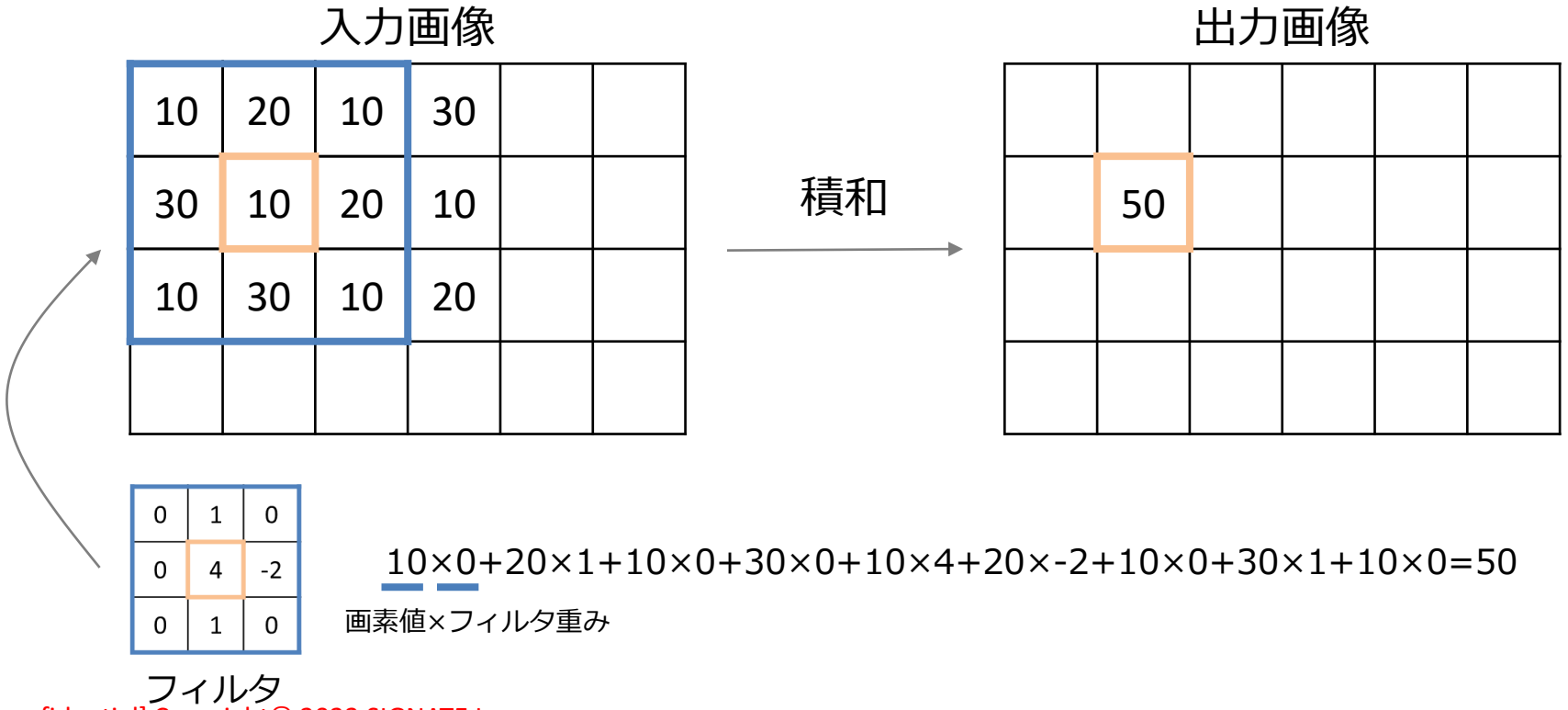
フィルタリングの仕組み

基本②：フィルタをスライドさせる



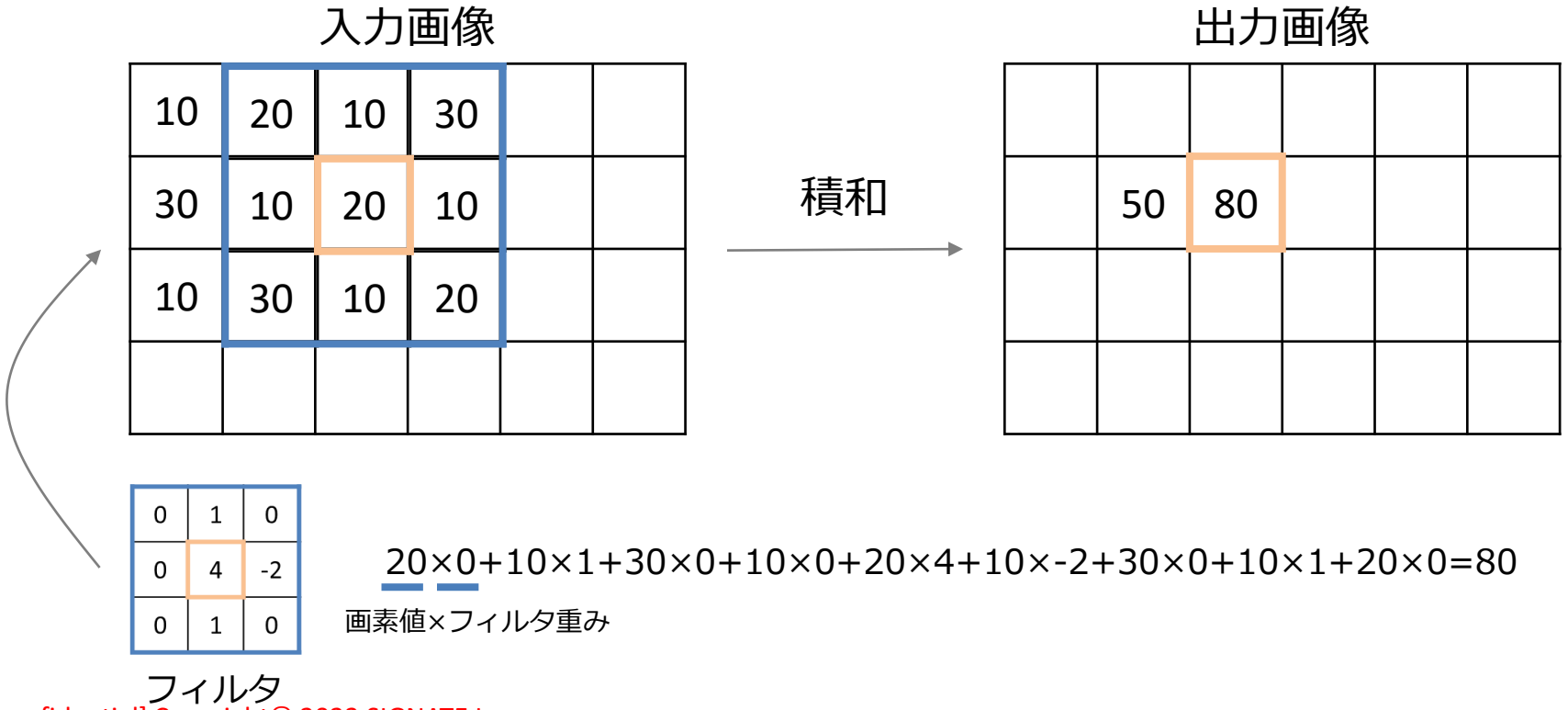
フィルタリングの仕組み

基本③：フィルタの重みを元に計算を行う



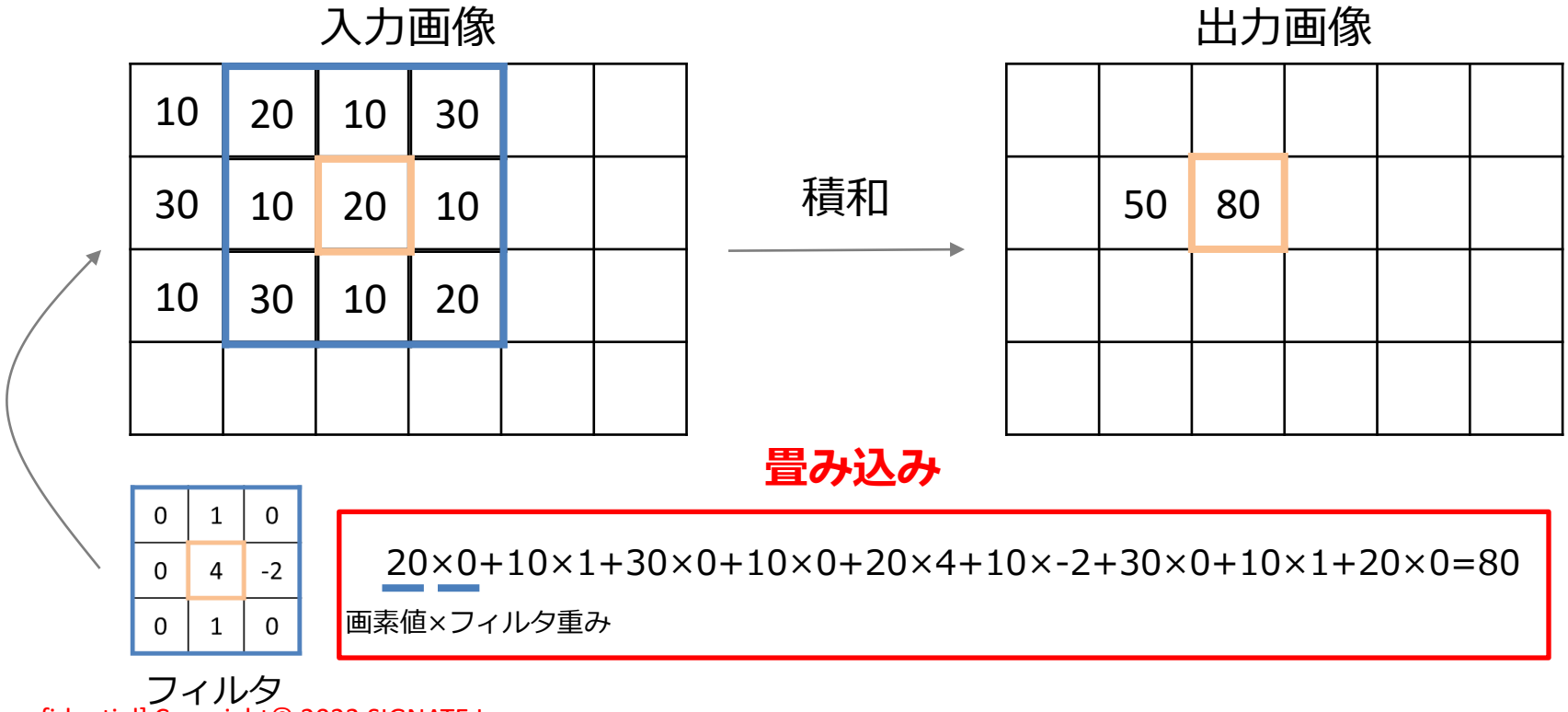
フィルタリングの仕組み

基本③：フィルタの重みを元に計算を行う



フィルタリングの仕組み

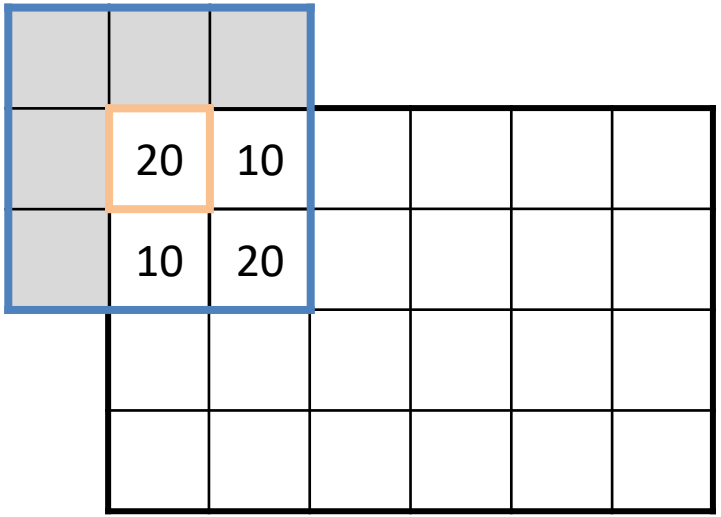
基本③：フィルタの重みを元に計算を行う



フィルタリングの仕組み

基本④：端は何かしらの値で補間する

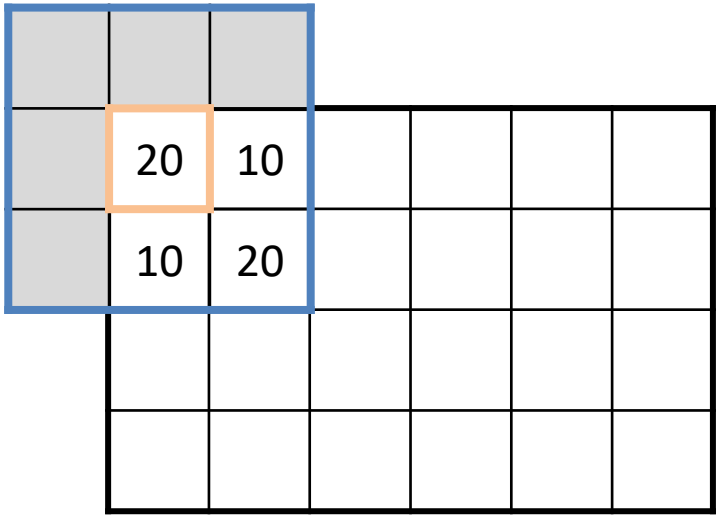
端を計算することができない



フィルタリングの仕組み

基本④：端は何かしらの値で補間する

端を計算することができない



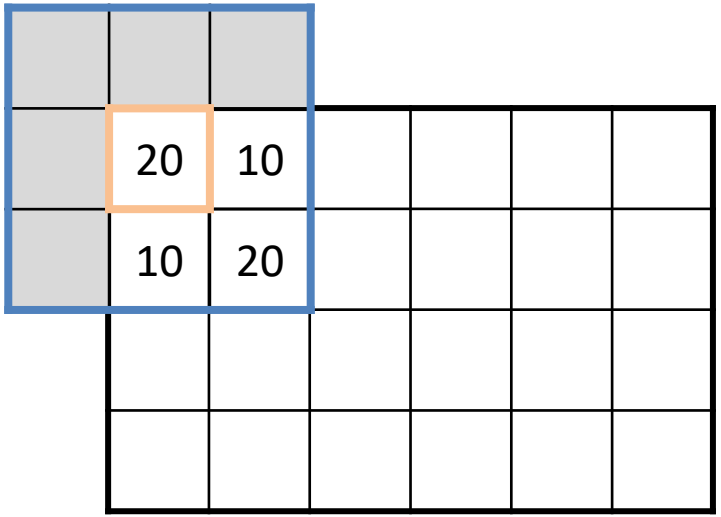
e.g) 外側を0で埋める

0	0	0	0	0	0	0
0	20	10				
0	10	20				
0						
0						

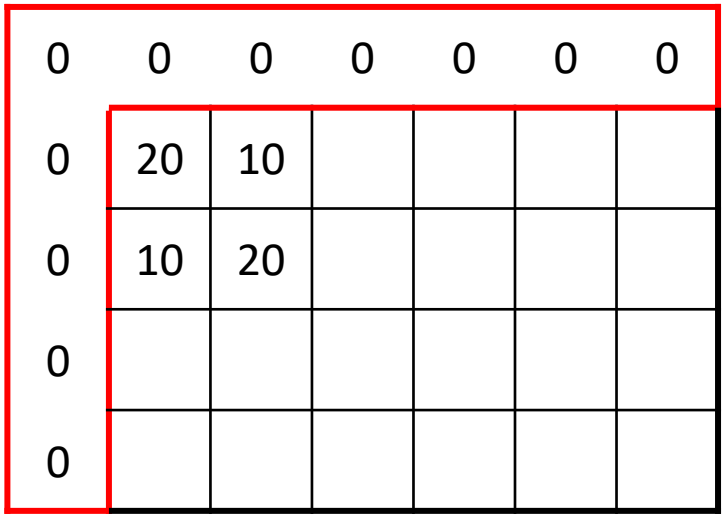
フィルタリングの仕組み

基本④：端は何かしらの値で補間する

端を計算することができない

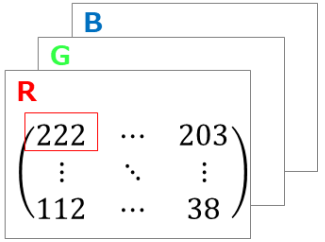


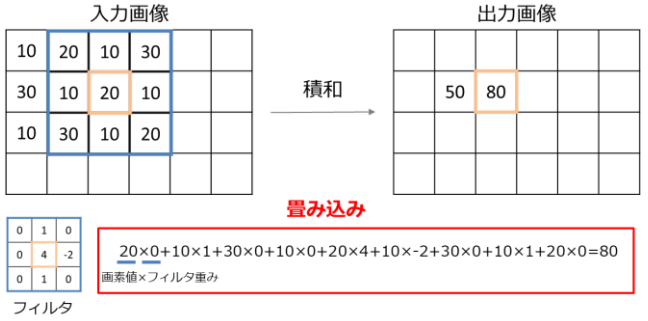
Zero-padding



ここまでのまとめ

- **画像データ**
  - 行列として表される
  - RGBなら3行列
- **画像操作**
  - 回転、平行移動、クロッピング、リサイズ
  - Data Augmentation
- **フィルタリング**
  - フィルタ（カーネル）
  - 畳み込み演算
  - Zero-padding

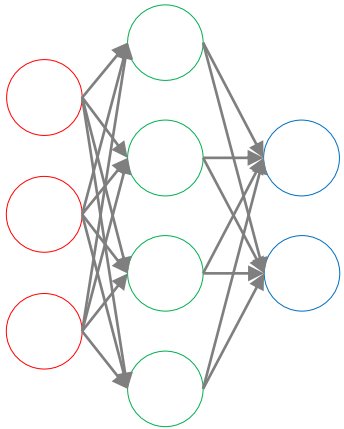


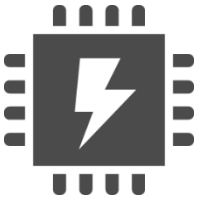


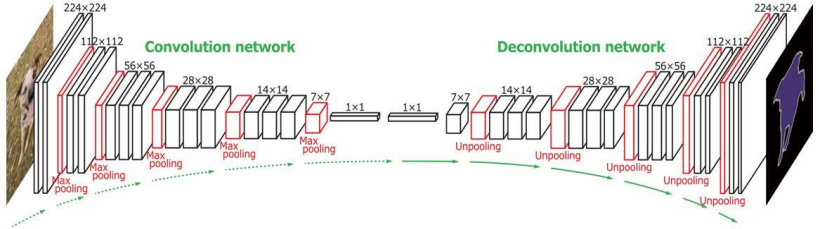
4. ディープラーニング入門

（復習）深層学習

- 深層学習（Deep Learning）
  - Neural Networkと呼ばれる機械学習手法の発展形
    - 昔は2~3層、今は多いもので1000層を超える
  - ハードウェア技術の進化と結びつき実現



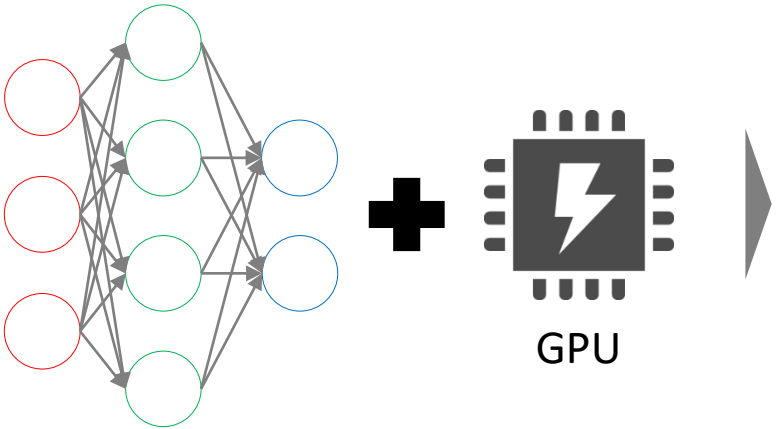


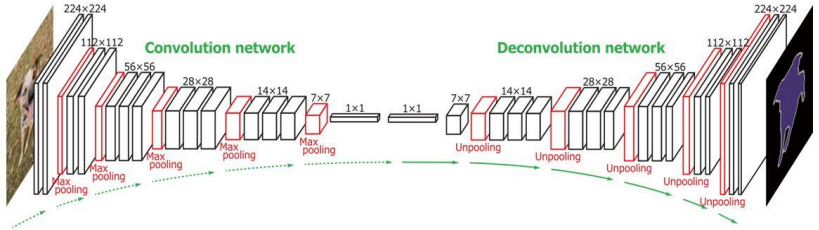


【引用】http://cvlab.postech.ac.kr/research/deconvnet/

（復習）深層学習

- 深層学習（Deep Learning）
  - **Neural Network**と呼ばれる機械学習手法の発展形
    - 昔は2~3層、今は多いもので1000層を超える
  - ハードウェア技術の進化と結びつき実現

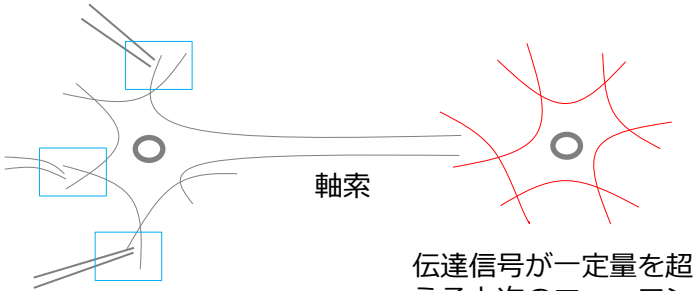




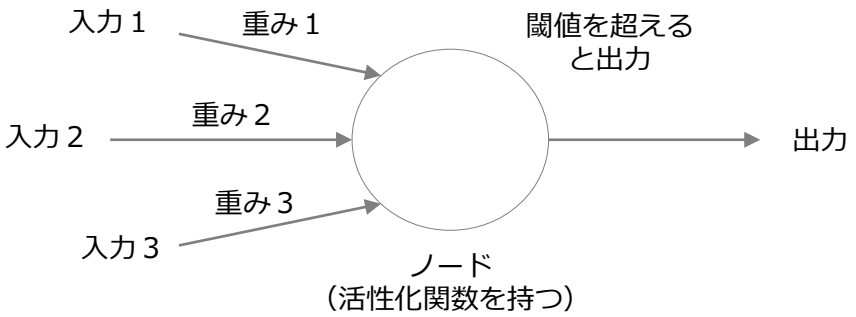
【引用】http://cvlab.postech.ac.kr/research/deconvnet/

人間のニューロンと人工ニューロン

人間のニューロン
1000億個くらいあると言われている



人工ニューロン

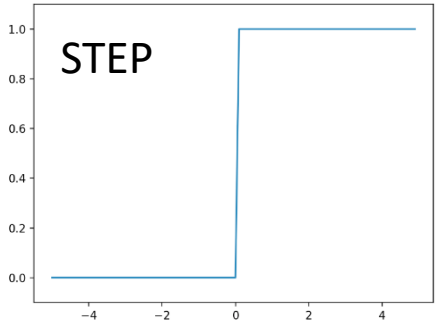


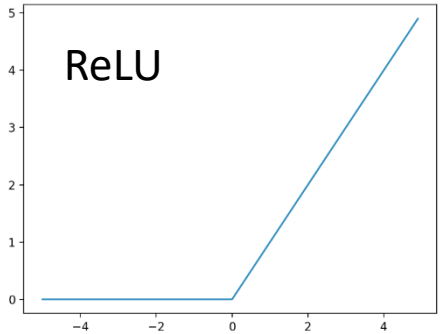
入力信号の総和がある一定量（閾値）を超えると発火し、次のニューロンに伝達する

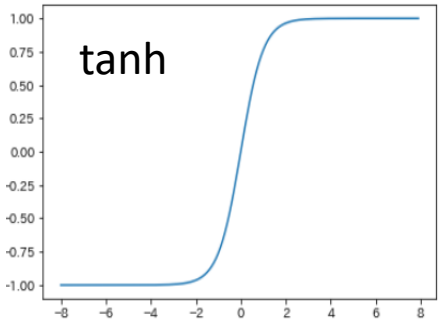
活性化関数が重要

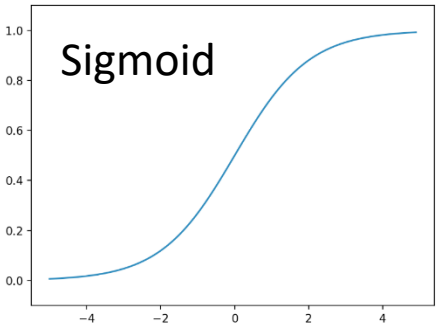
いろいろな活性化関数

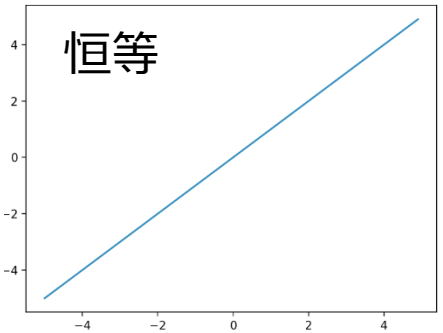
- 入力信号の総和に対してどのように活性化させるかを決める関数
  - 学習の効率化や、タスクに応じた出力を得る為に選択が必要

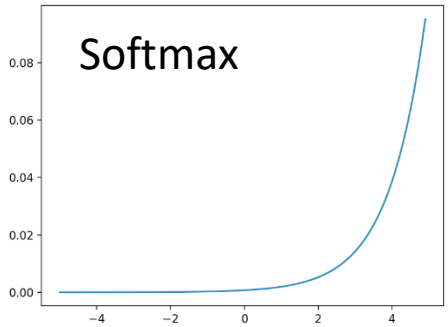






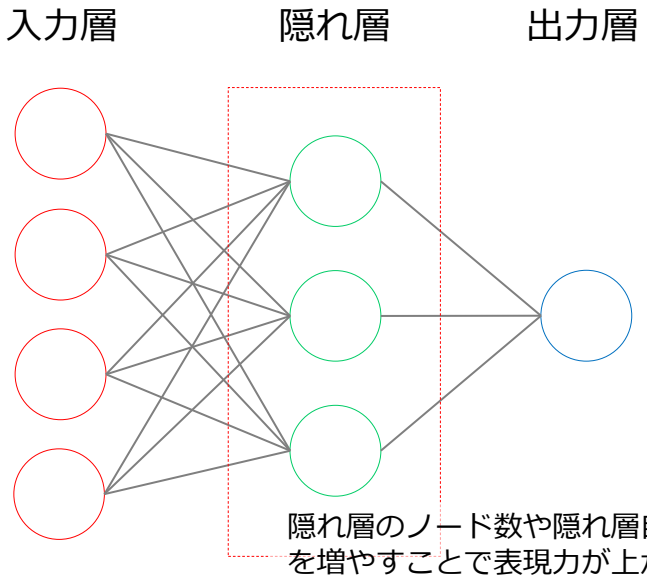


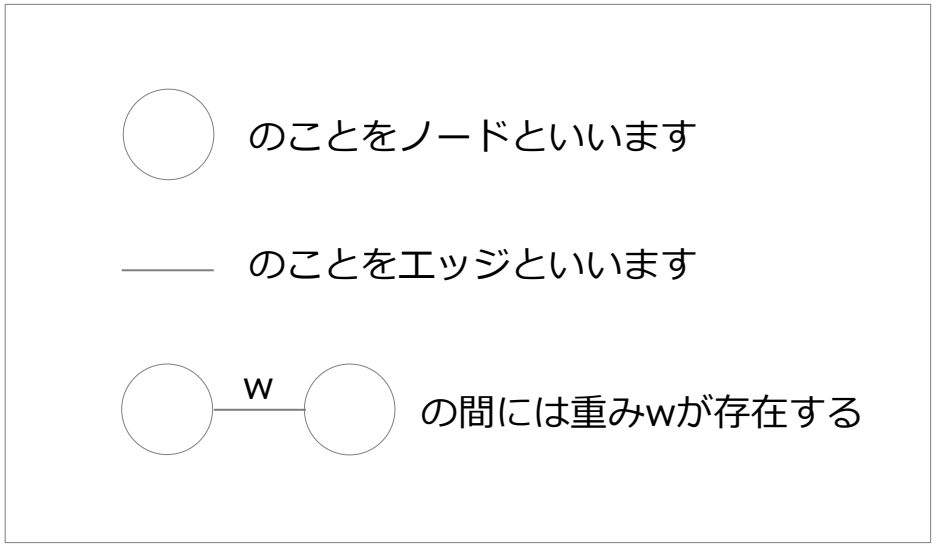




ニューラルネットワーク

- 人工ニューロンを組み合わせたもの
  - 1943年にマカロックとピッツがモデル化したのが始まり
  - 近年よく取り上げられているものは「階層型ネットワーク」という型

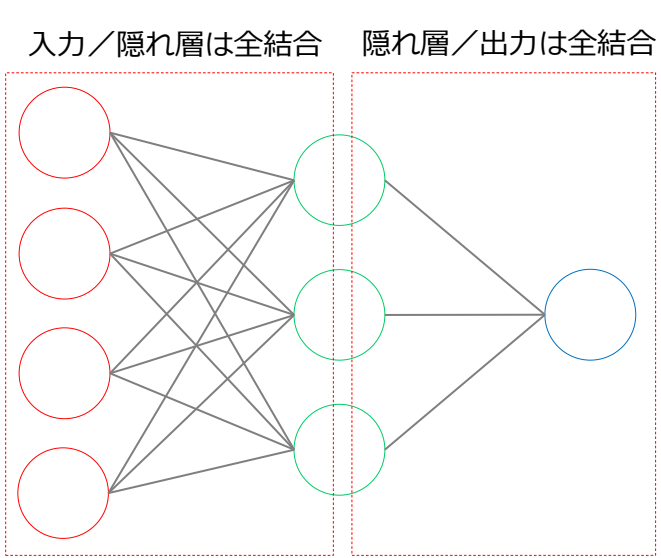


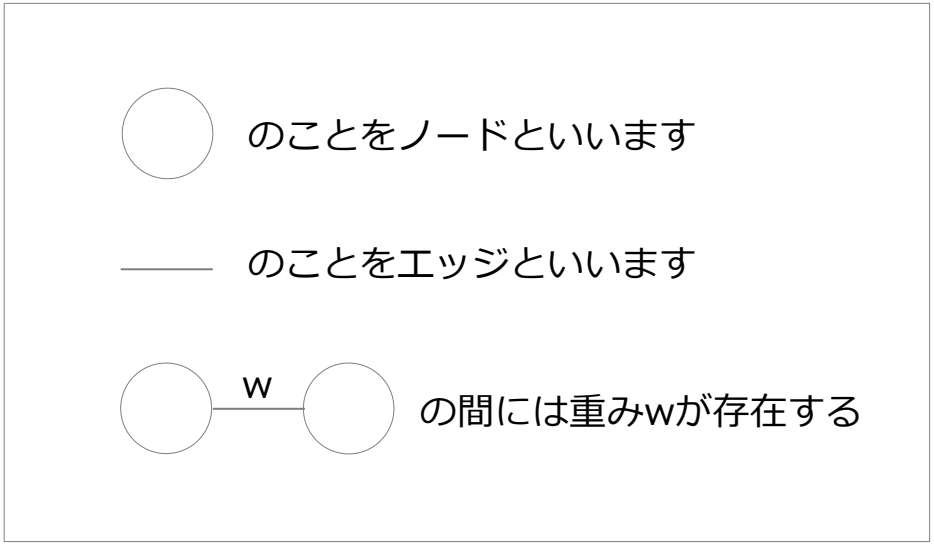


ニューラルネットワーク

- 人工ニューロンを組み合わせたもの
  - 1943年にマカロックとピッツがモデル化したのが始まり
  - 近年よく取り上げられているものは「階層型ネットワーク」という型

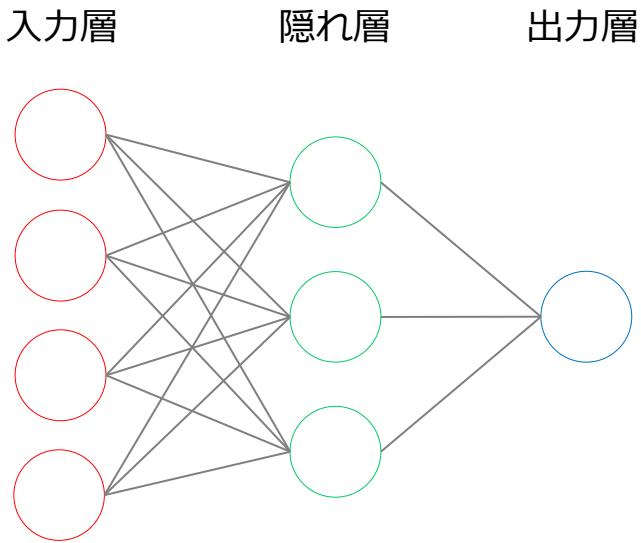
“全結合”＝前層の全ノードが次層の全ノードと結合

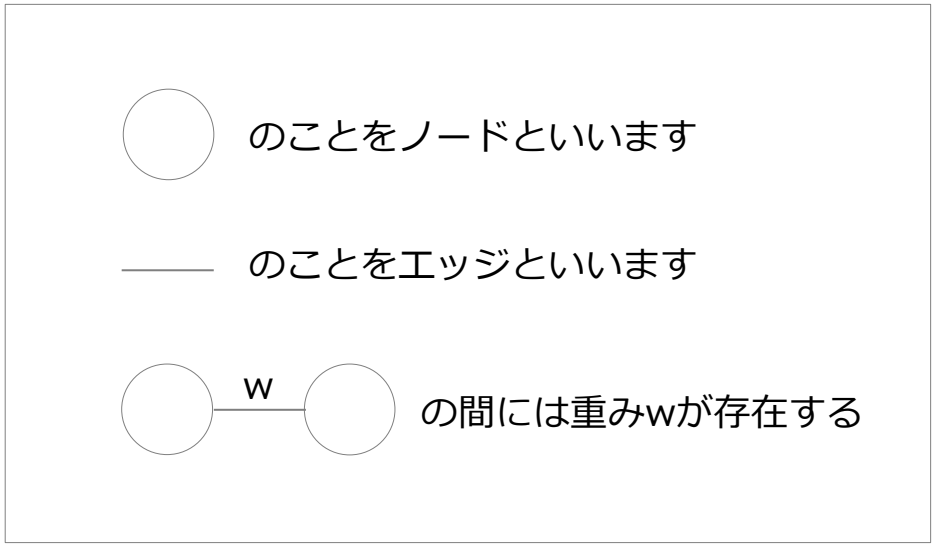




ニューラルネットワーク

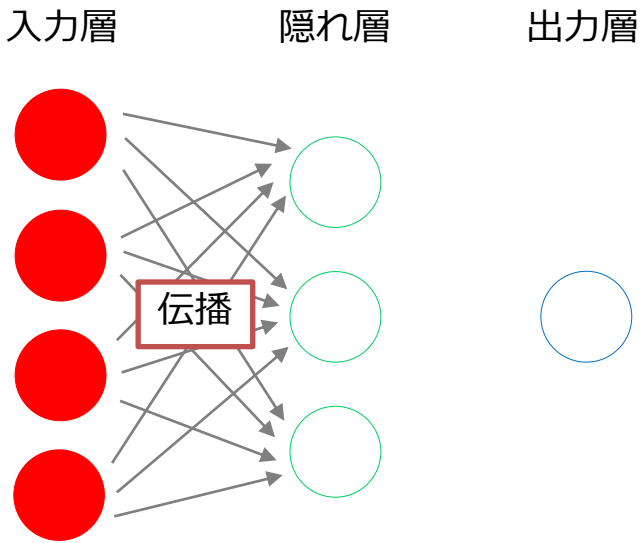
- 人工ニューロンを組み合わせたもの
  - 1943年にマカロックとピッツがモデル化したのが始まり
  - 近年よく取り上げられているものは「階層型ネットワーク」という型

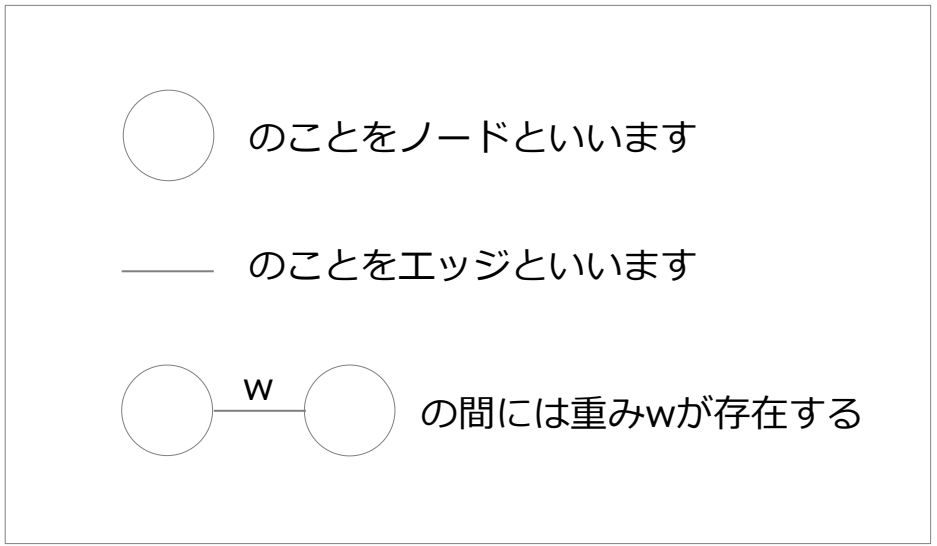




ニューラルネットワーク

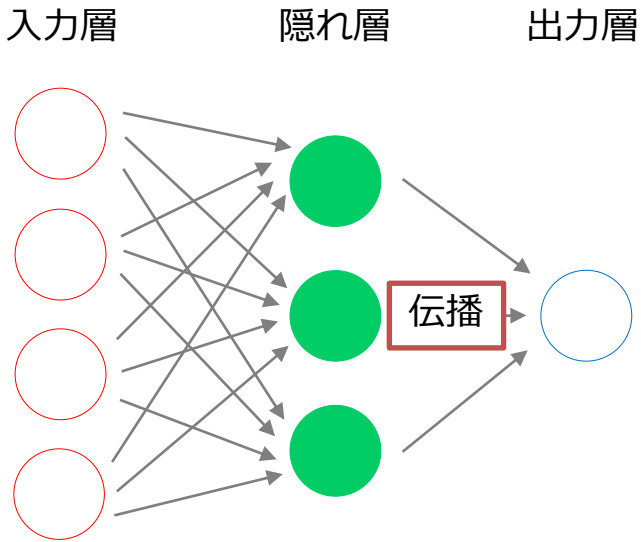
- 人工ニューロンを組み合わせたもの
  - 1943年にマカロックとピッツがモデル化したのが始まり
  - 近年よく取り上げられているものは「階層型ネットワーク」という型

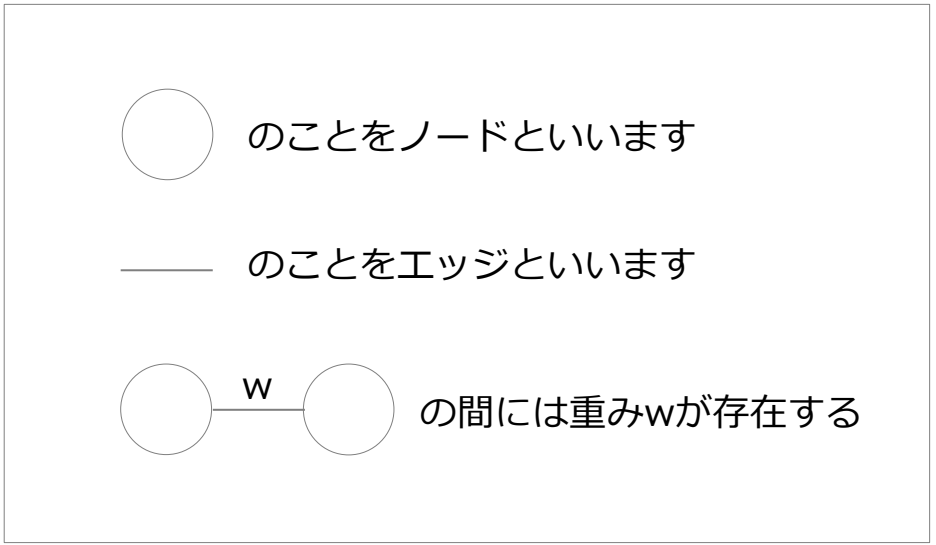




ニューラルネットワーク

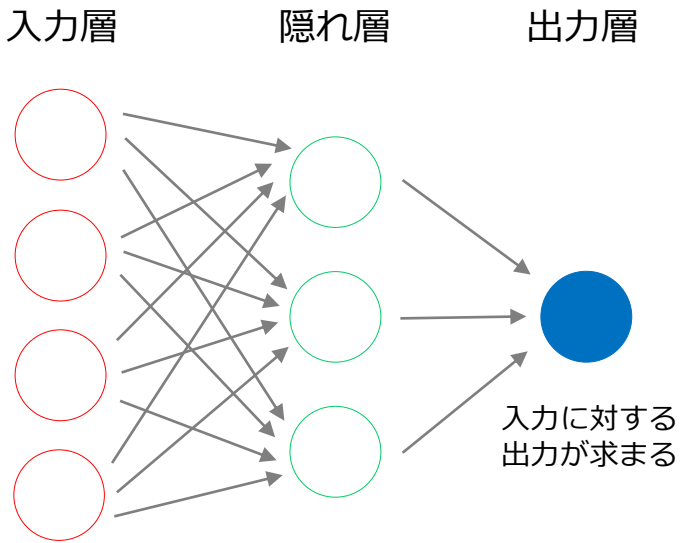
- 人工ニューロンを組み合わせたもの
  - 1943年にマカロックとピッツがモデル化したのが始まり
  - 近年よく取り上げられているものは「階層型ネットワーク」という型





ニューラルネットワーク

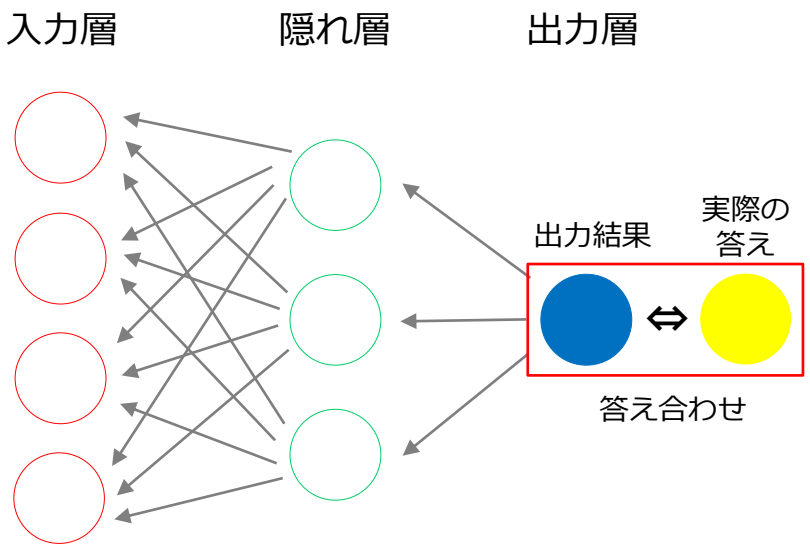
- 人工ニューロンを組み合わせたもの
  - 1943年にマカロックとピッツがモデル化したのが始まり
  - 近年よく取り上げられているものは「階層型ネットワーク」という型



この過程を“順伝播”と言います。
モデルが入力に対して予測をする際には順伝播により、出力の値が定まります。
※機械学習で言う「予測」の部分となります

ニューラルネットワーク

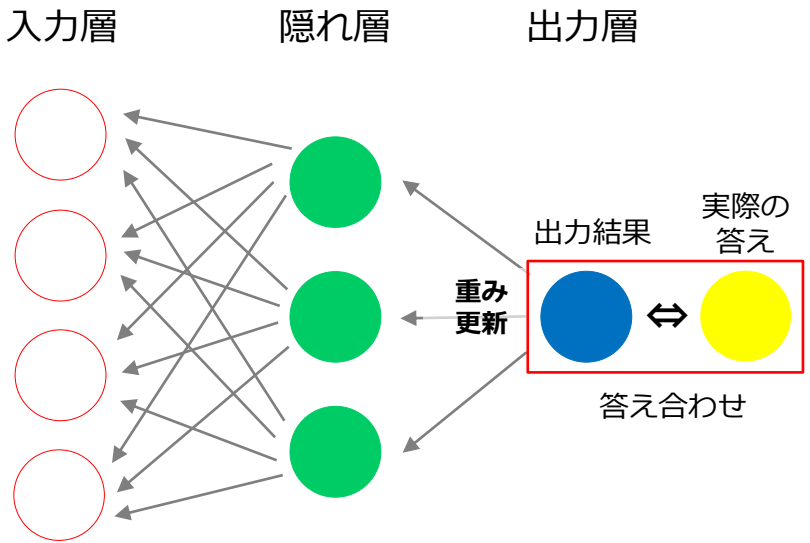
- 人工ニューロンを組み合わせたもの
  - 1943年にマカロックとピッツがモデル化したのが始まり
  - 近年よく取り上げられているものは「階層型ネットワーク」という型



モデルの学習は“逆伝播”となります。
出力値と実際の答えとの答え合わせをし、この誤差を小さくするように重みが更新されます。

ニューラルネットワーク

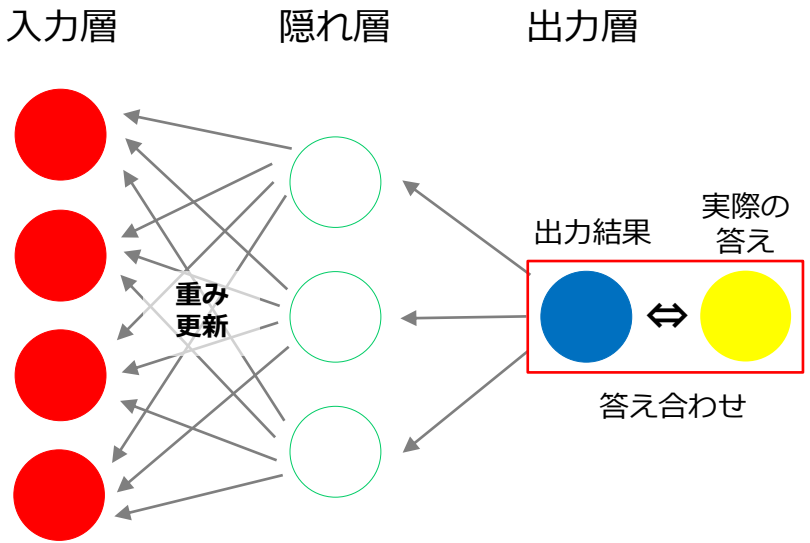
- 人工ニューロンを組み合わせたもの
  - 1943年にマカロックとピッツがモデル化したのが始まり
  - 近年よく取り上げられているものは「階層型ネットワーク」という型



モデルの学習は“逆伝播”となります。
出力値と実際の答えとの答え合わせをし、この誤差を小さくするように重みが更新されます。

ニューラルネットワーク

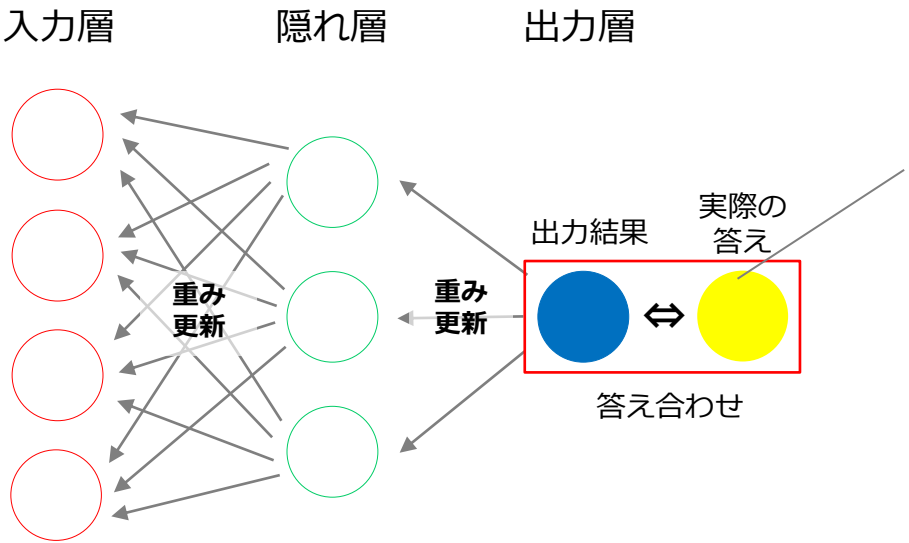
- 人工ニューロンを組み合わせたもの
  - 1943年にマカロックとピッツがモデル化したのが始まり
  - 近年よく取り上げられているものは「階層型ネットワーク」という型



モデルの学習は“逆伝播”となります。
出力値と実際の答えとの答え合わせをし、この誤差を小さくするように重みが更新されます。

ニューラルネットワーク

- 人工ニューロンを組み合わせたもの
  - 1943年にマカロックとピッツがモデル化したのが始まり
  - 近年よく取り上げられているものは「階層型ネットワーク」という型



「誤差逆伝播法」という
⇒誤差が小さくなる方向に重みを調整する

タスクに応じた“**誤差（損失）関数**”を利用する
- 回帰：MSE等（二乗誤差）
- 2値分類：Binary Cross Entropy
- 3つ以上の分類：Categorical Cross Entropy

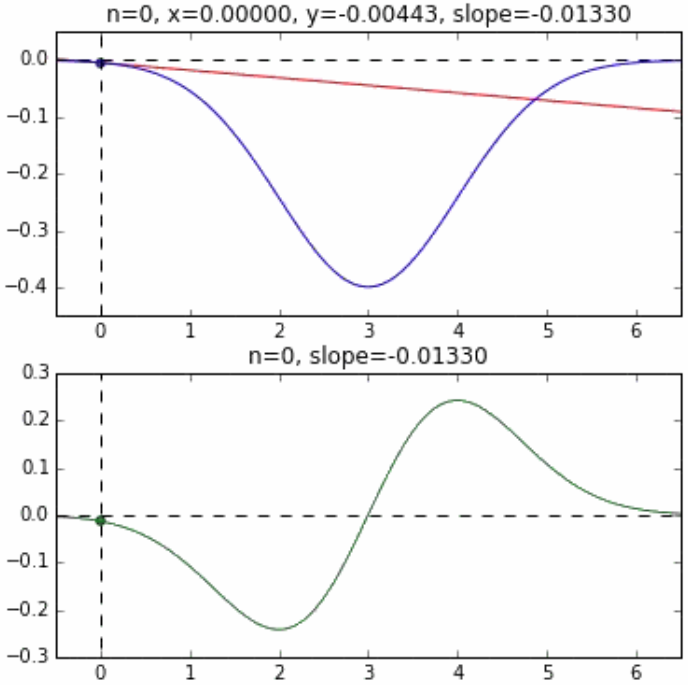
誤差を小さくするためには…

Point1 勾配降下法
誤差が小さくなるようにする最適化手法

Point2 学習率
どのくらい大きく動かすかは「学習率」で決まる

勾配降下法（Gradient Descent）

- 目的となる関数を最小化（最大化）するパラメータを探索する手法



【引用】
https://qiita.com/kenmatsu4/items/d282054ddedbd68fecb0

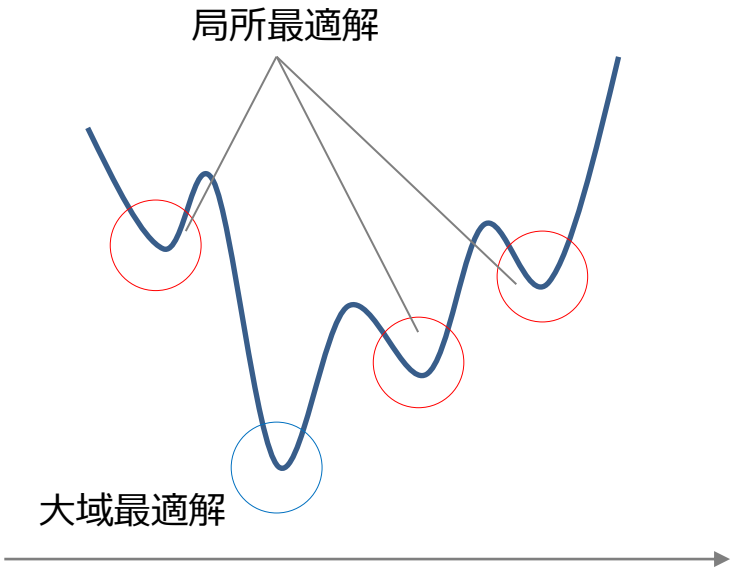
試行を繰り返すことで徐々に最適値を探索
⇒パラメータが多いと計算に時間がかかる

▼勾配降下法の種類
- 最急降下法（Gradient Descent, GD）
- 確率的勾配降下法（Stochastic GD, SGD）
- ミニバッチ確率的勾配降下法

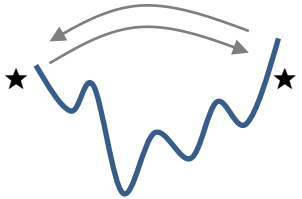
★ミニバッチ確率的勾配降下法
- ニューラルネットの学習に良く使われる
- データを小さく分割（ミニバッチ）
- 各ミニバッチ毎で探索方向を決定する

学習率

★誤差を最小とする地点を探すことが目的



学習率が大きすぎると…



通り過ぎてしまい、
いつまでも収束しない

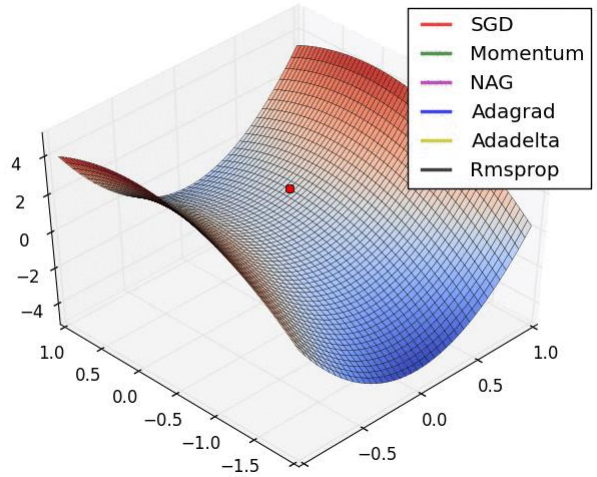
学習率が小さすぎると…

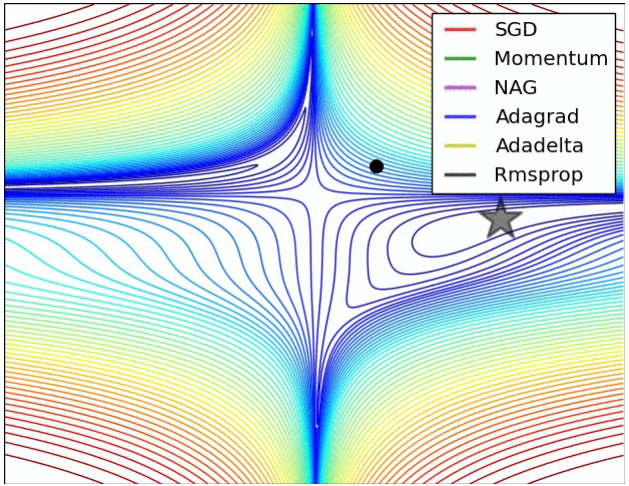
学習に時間がかかる

局所から抜け出せない

オプティマイザー

- 勾配降下法の最適化アルゴリズム
  - 局所から抜け出す為の工夫手段
  - どれが良いかは実験的に試すしかない



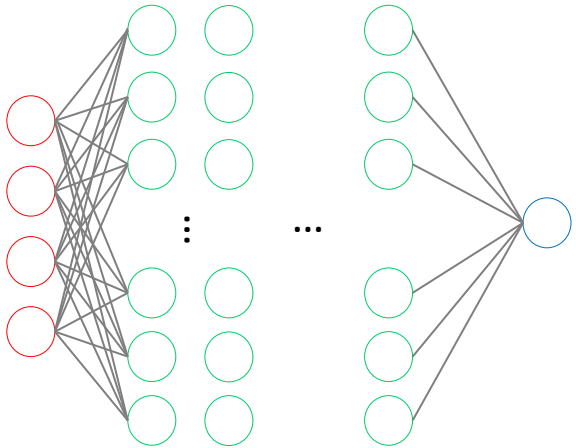


【引用】http://imgur.com/a/Hqolp

ここまでのまとめ

- **ニューラルネットワーク**
  - 人工ニューロンが複数つながったもの
  - 各エッジの重みを更新することで学習がなされる
  - 入力層と出力層はタスクに応じて設計が必要

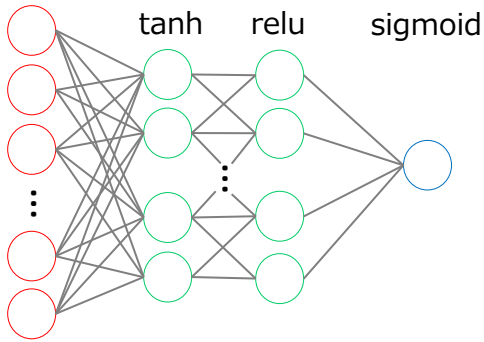
- **精度を上げる為には…**
  - 隠れ層のノード数及び隠れ層の層数
  - 活性化関数×オプティマイザー×学習率



Kerasによる実装

Keras

e.g) 銀行顧客ターゲティング
口座を開設するか否かの確率を求める



入力層
51dim

隠れ2層
各10dim

出力層
1dim

※エポック=100,minibatch=128
オプティマイザー=SGD

```
import keras
from keras.models import Sequential
from keras.layers import Dense
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

dat = pd.read_csv("train.csv")
dat2 = pd.get_dummies(dat2.iloc[:,1:-1])
dat2 = dat2.apply(lambda x : (x-np.mean(x))/np.std(x))
trainX, testX, y_train, y_test = train_test_split(dat2,dat["y"],test_size=0.3)
```

読み込み
前処理

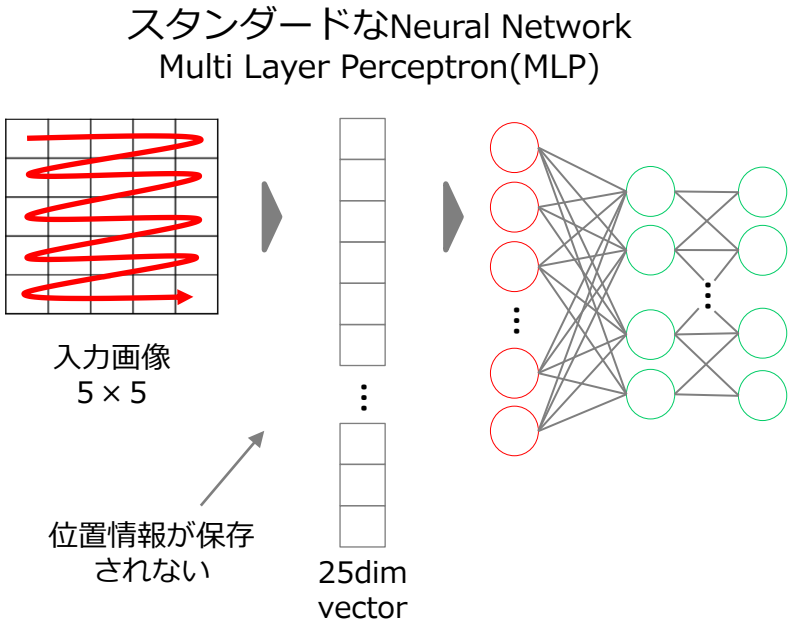
```
model = Sequential()
model.add(Dense(10,input_shape=(51,),activation="tanh"))
model.add(Dense(10,activation="relu"))
model.add(Dense(1,activation="sigmoid"))
model.compile(loss='binary_crossentropy',
              optimizer=keras.optimizers.SGD(lr=0.1, momentum=0.9, decay=0.0, nesterov=True),
              metrics=['accuracy'])
```

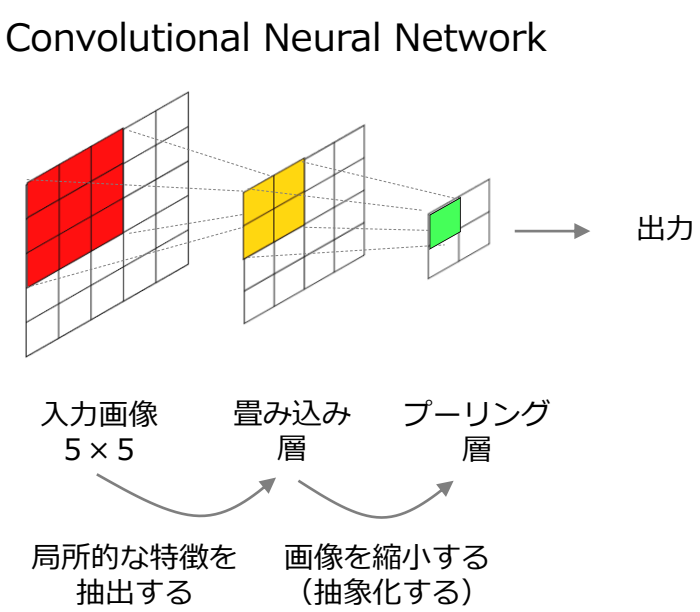
```
history = model.fit(trainX, y_train,
                    batch_size=128,
                    epochs=100,
                    verbose=1,
                    validation_data=(testX, y_test))
```

← validation（ホールドアウト法）

Convolutional Neural Network(CNN)

- Neural networkにConvolution（畳み込み）を追加したもの
  - 2012年における画像認識コンペにて圧倒的な精度を出し、有名に。(AlexNet)
  - その後に続くGoogLeNetやResNet, SENetの礎となる構造

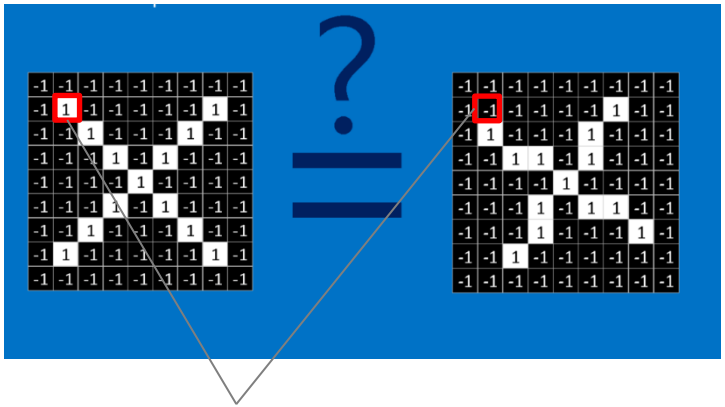




なぜCNNが良いのか？

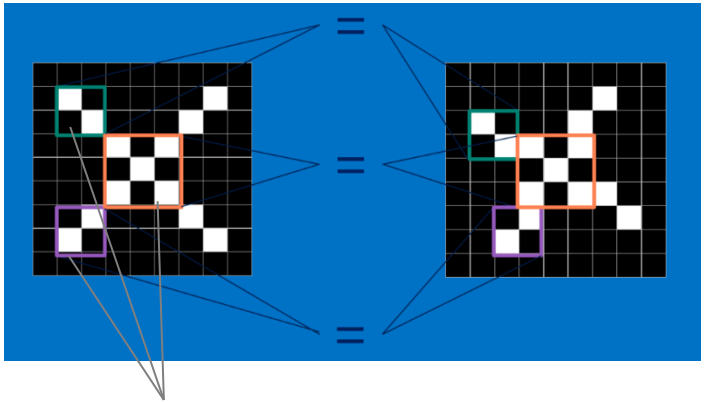
- 画像認識の仕方に違いがある

▼MLPは全体的に見てパターンマッチを行う
- 位置ズレの影響が大きい
- 典型から外れるとマッチングが困難



MLPでは少しでも位置がズレると、入力値が変わってしまう為、判定に大きな影響が出る

▼CNNは細かくパターンマッチを行う
- 合成性：断片を組み合わせて判定
- 移動不変性：位置ズレに強い

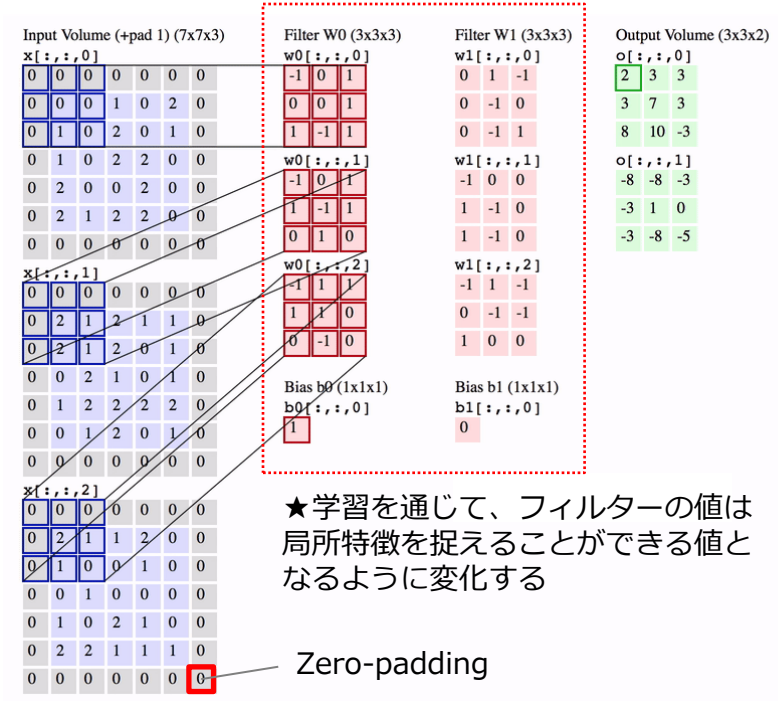


断片的な局所的情報（特徴マップという）の基づいたマッチングをし、そのマッチング度合をトータルに考慮して判定を行う

【引用】https://brohrer.github.io/how_convolutional_neural_networks_work.html

CNNの特徴的な要素

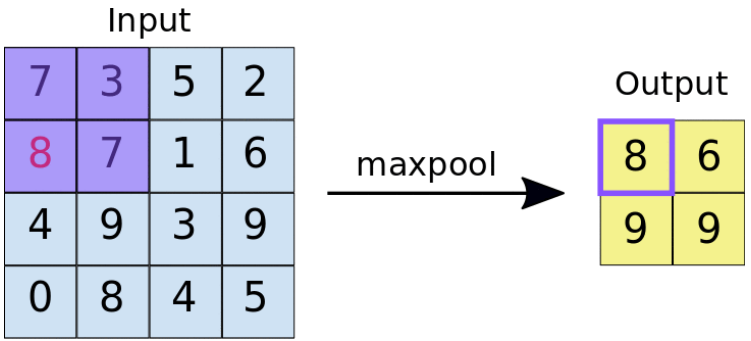
畳み込み層（Convolution Layer）



★学習を通じて、フィルターの値は局所特徴を捉えることができる値となるように変化する

【引用】http://cs231n.github.io/convolutional-networks/

プーリング層（Pooling Layer）

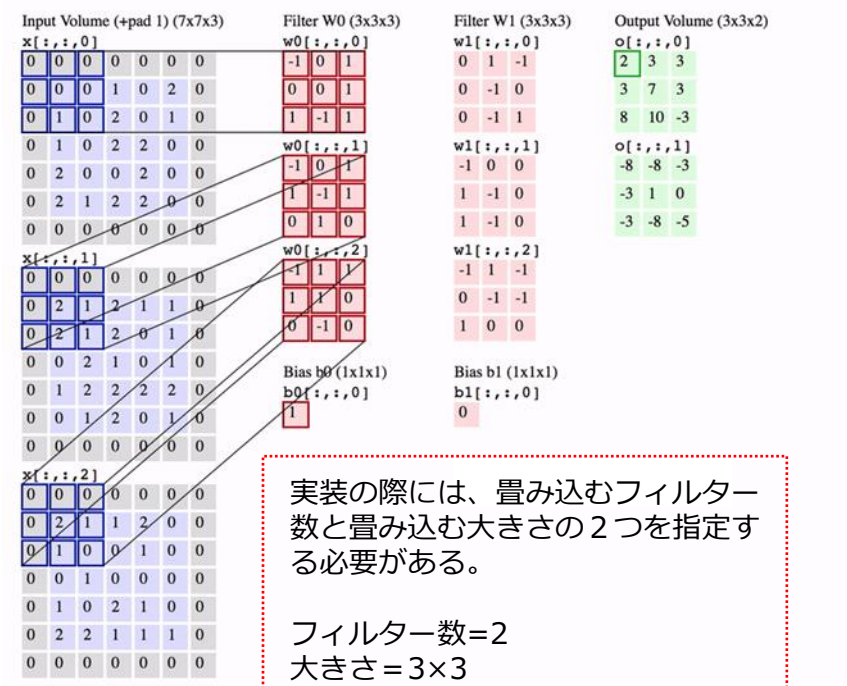


単純に各領域における最大値を出力する。
これにより、特徴的なピクセル値を残すことができ、
微妙な位置のずれや歪みなどを吸収することができる。
※畳み込み層のように学習するパラメータがない

【引用】https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

CNNの特徴的な要素

畳み込み層（Convolution Layer）

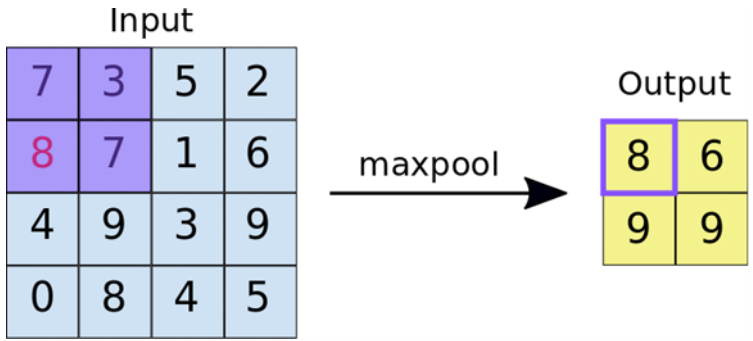


実装の際には、畳み込むフィルター数と畳み込む大きさの２つを指定する必要がある。

フィルター数=2
大きさ＝3×3

【引用】http://cs231n.github.io/convolutional-networks/

プーリング層（Pooling Layer）



実装の際には、プーリングするサイズを指定する必要がある。

大きさ＝2×2

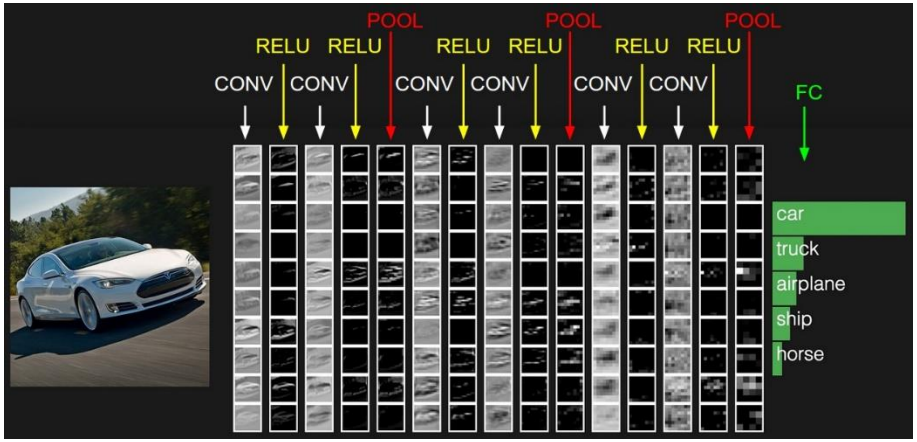
【引用】https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks

CNNの例

AlexNet

★基本的には「畳み込み⇒プーリング」を1セットとして何回か繰り返すことでCNNを多層化している構造が多い

VGGNet（簡易）の例



【引用】http://cs231n.github.io/convolutional-networks/

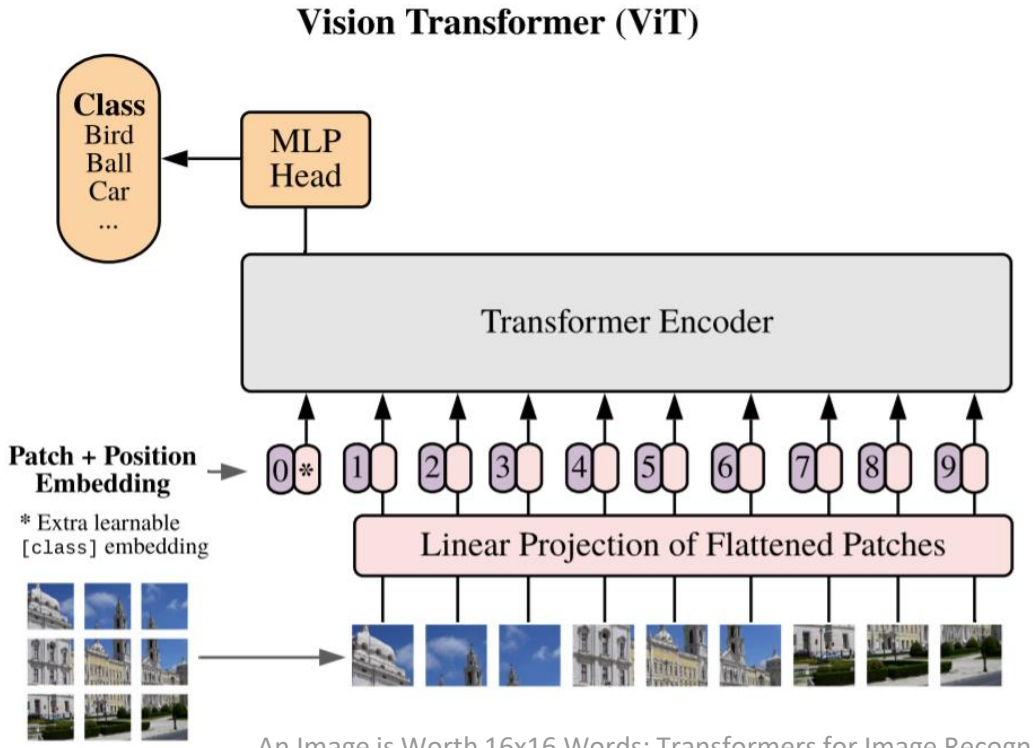
VGGNet

最近はCNNじゃない？

元々は言語系で主流なモデル

E.g) 翻訳モデル
- 入力：日本語
- 出力：英語

これを画像に転用。
- 入力：分割した画像
- 出力：ラベル



An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, Alexey Dosovitskiy, et al. (2020)

おしまい

Let's Enjoy DataScience!!!