AIエンジニアリング　(5)

教師あり学習まとめ

機械学習とはなんだろう

機械学習とは

- 実は機械学習の定義もあまりはっきりしたものはない
- ただ、一般的に機械学習は次の3種類からなるので、この総称といっていい
  1. 教師あり学習
  2. 教師なし学習
  3. 強化学習

1.教師あり学習(supervised learning)

1. 人間がお手本(教師データ)を用意してあげる
2. 教師データをもとにして、人間と同じように判断できるまで学習する

- 例
  - 迷惑メール(スパムメール)フィルタ
  - 文字認識
  - 画像認識
  - 音声認識

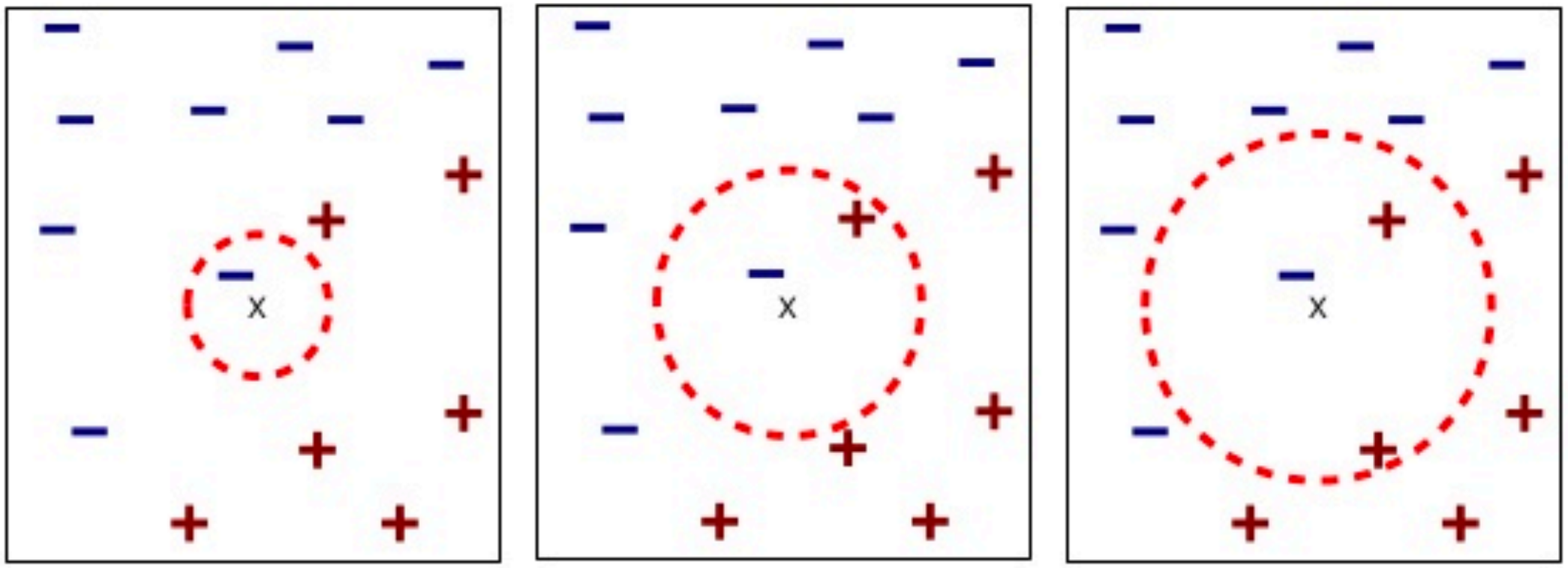
どうやって予想するか？

- ほとんどのアルゴリズムは「重み」の学習をする
- A * CRIM + B * ZN+ … + M * LSTAT + N = 住宅価格
- みたいな式(モデル)を考えたら、A〜Nまでの変数をいろいろ変えることが学習になる。これが重み(weight)。
- どういう式(モデル)を考えるかと、重みをどうやって計算するかでいろいろな手法がある

線形モデル

- A * CRIM + B * ZN+ ... + M * LSTAT + N = 住宅価格
- のように、1変数(1特徴量)あたり1個の係数を掛け算して全部足す（内積を計算）、というようなモデル
- とても簡単だが、簡単がゆえのメリットがある
  - シンプル(実装が簡単)
  - 早い
  - 理論が明快
  - 応用が広い
  - 精度が低いかと言うとそうでもない

k-NN (k=1,2,3)



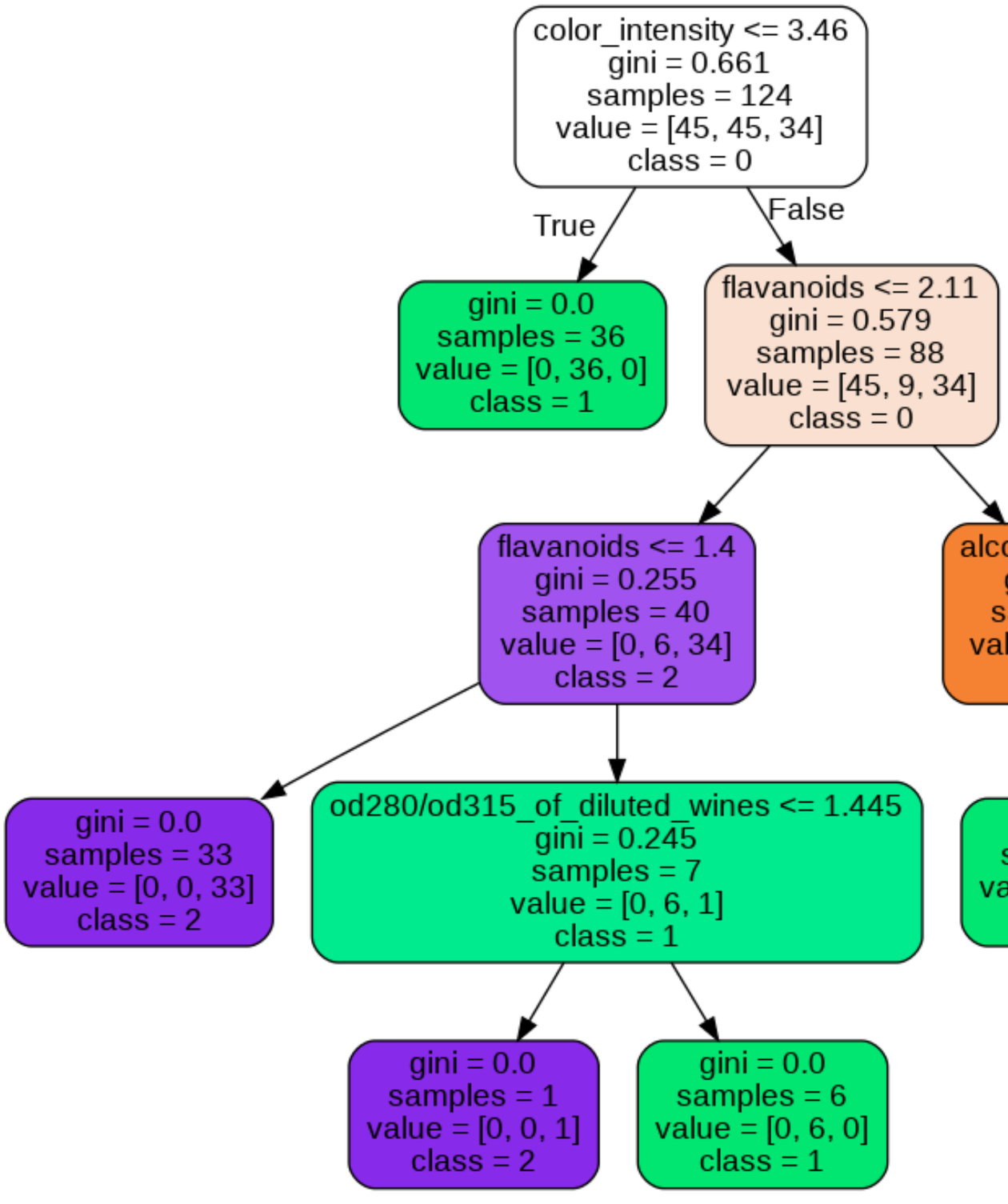
(a) 1-nearest neighbor　(b) 2-nearest neighbor　(c) 3-nearest neighbor

判定したいもの(X)に一番近いデータをk個とり、

多数決で決める。

よって、明らかにkは奇数のほうが良い。

決定木(decision tree)

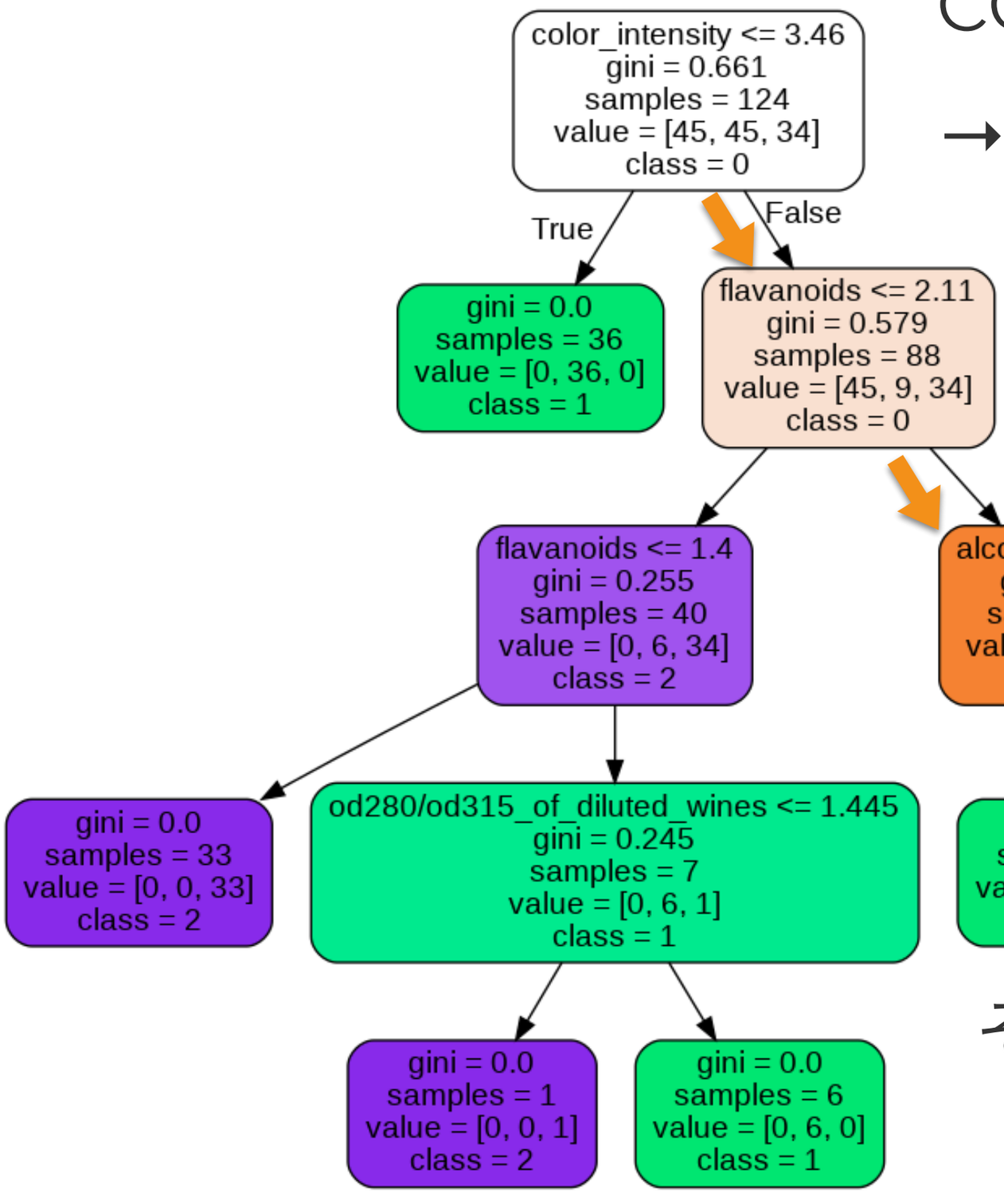


1個の特徴を選択して、

YES/NOで分岐する

…というのを繰り返す。

決定木(decision tree)



COLOR_INTENSITYが3.46以下？

→いいえ

FLAVANOIDSが2.11以下？

→いいえ

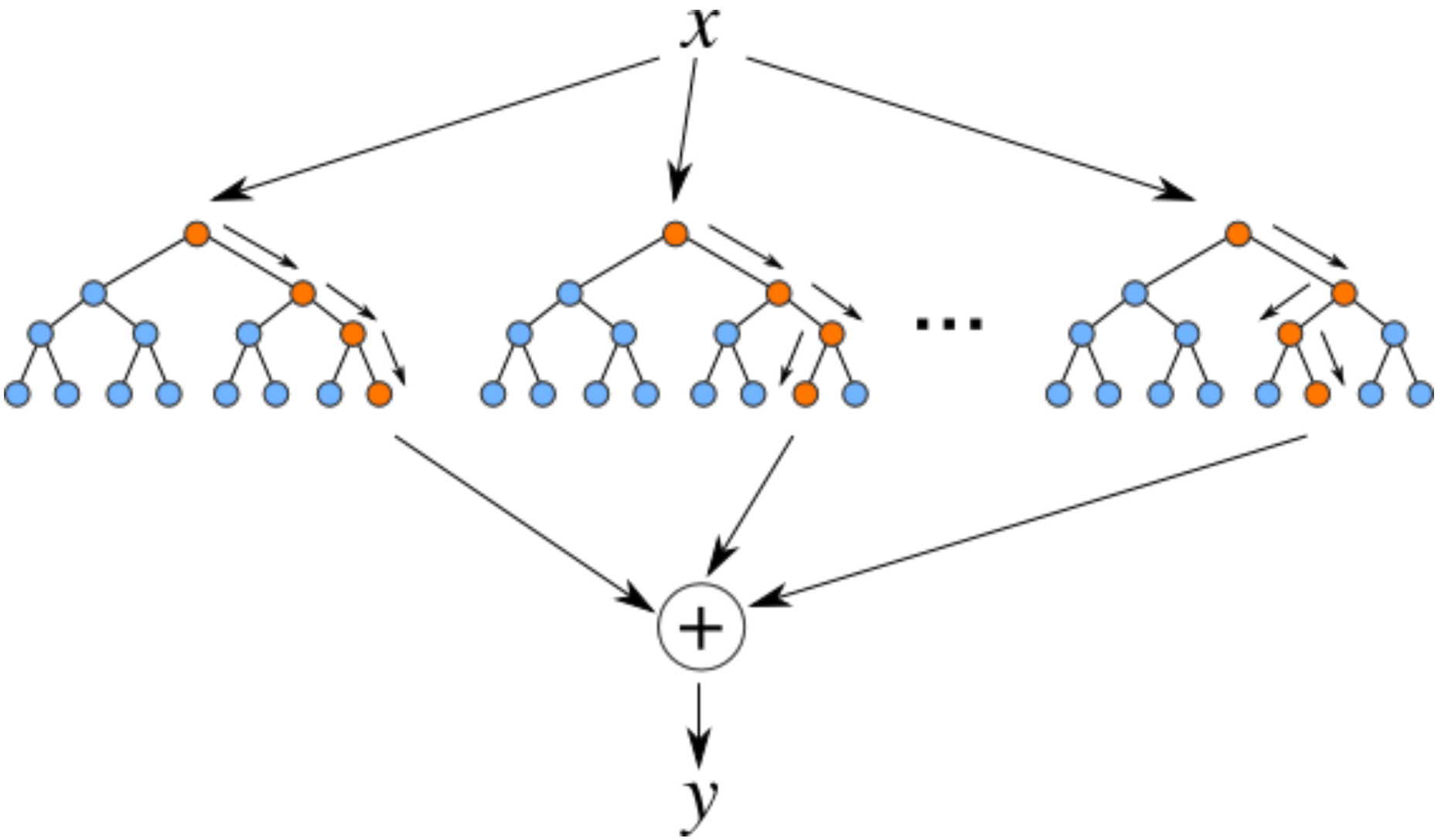
ALCOHOLが12.785以下？→はい

それはCLASS=1です

Random Forest 系

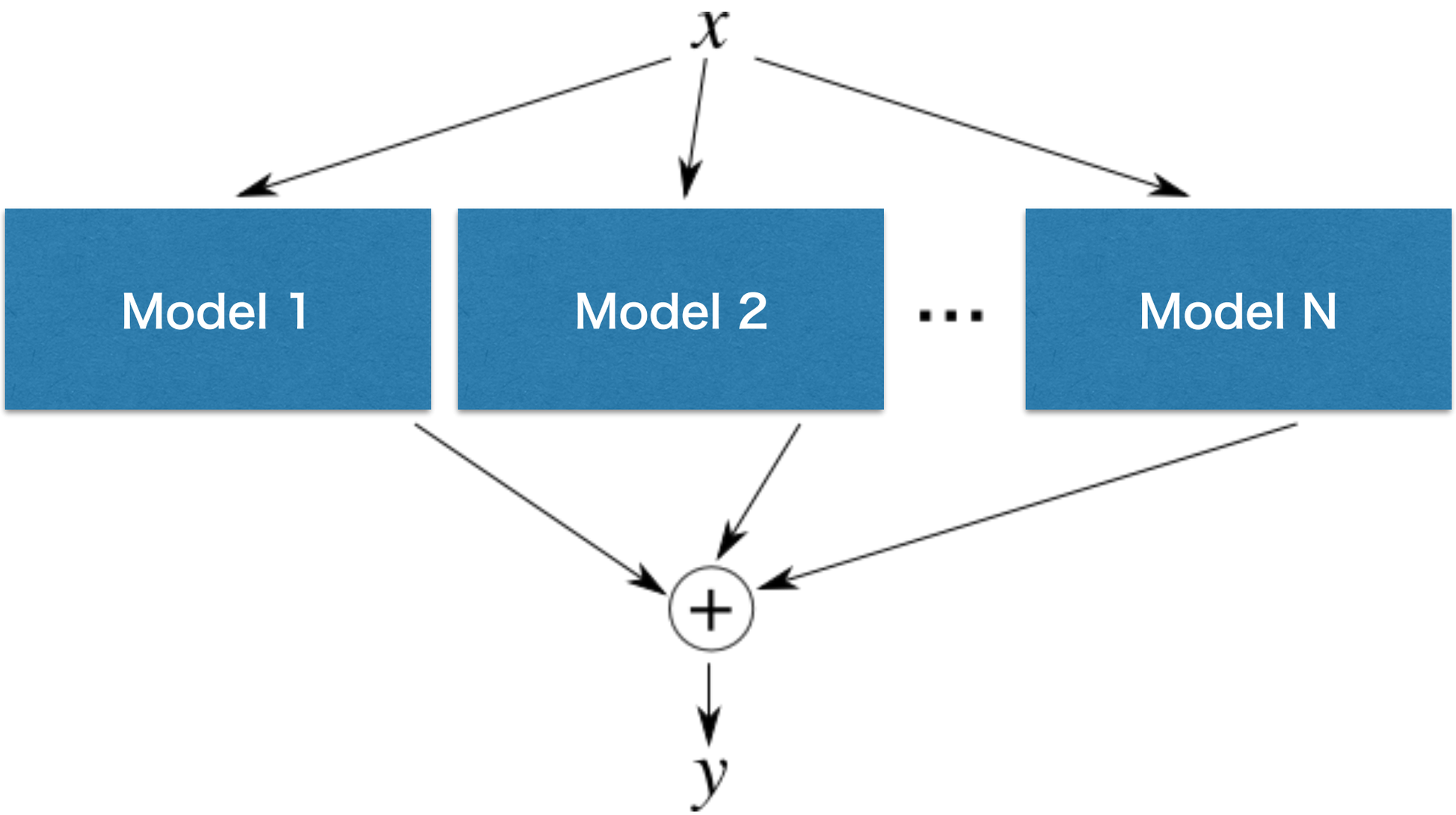
- 2001年登場→最近の主流
- scikit-learn に実装済みかつ有名なものは3つ
  - Random Forest (元祖)
  - Extremely Randomized Trees (Extra Trees)
  - Gradient Boosting Decision Trees (GBDT)
- その他色々

Random Forest 系



決定木をたくさん用意して、多数決で答えを決める

アンサンブル(Ensemble)学習



性能が低い学習器(弱学習器)をたくさん用意して、

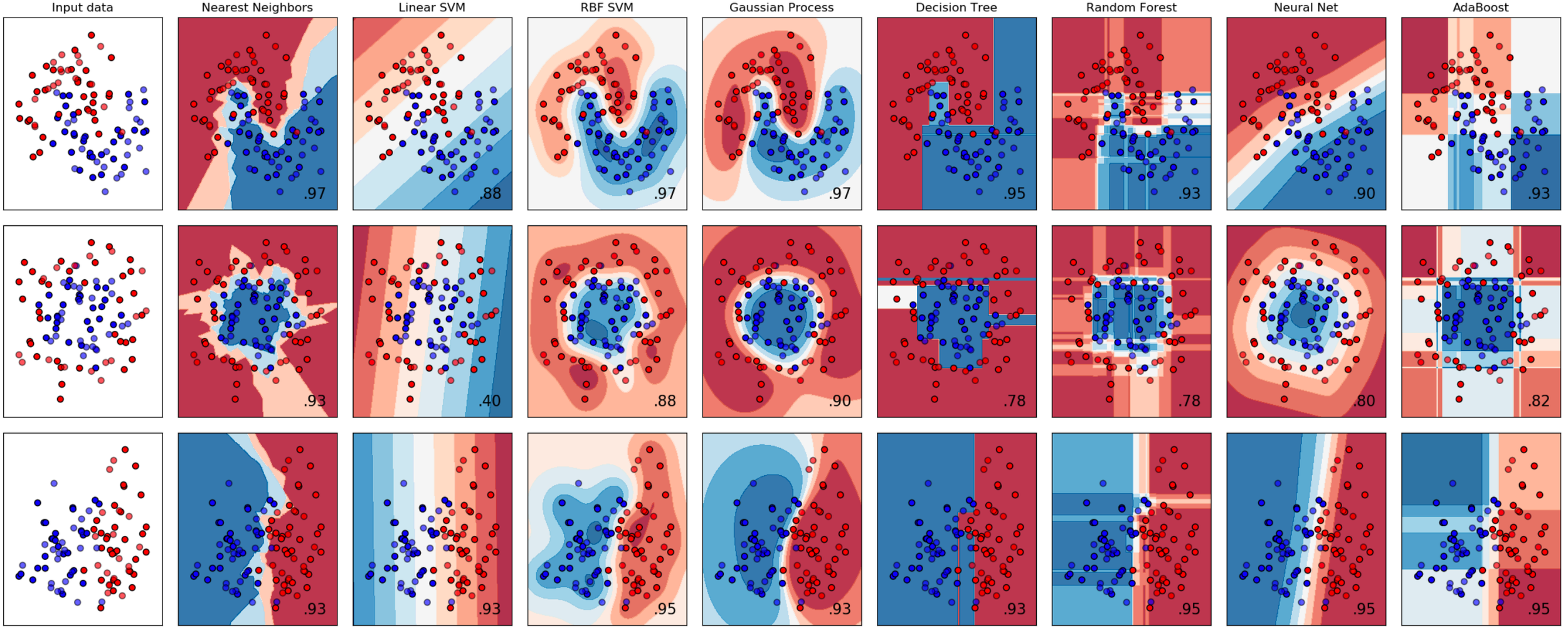
それらを合体させて性能の高い学習器を作る手法の総称

アンサンブル学習

- アンサンブル学習の種類
  - バギング(Bagging) - Random Forest など
    - たくさん作って合わせるだけ(多数決 or 平均を取る)
  - ブースティング(Boosting) - GBDT, AdaBoost など
    - 1個ずつモデルを作って、より難しい問題に適応させていく方法
  - スタッキング(Stacking) - 特定の手法というよりはテクニック
    - あるモデルの出力を次のモデルの入力にして、多段階に学習させていく方法

分類器いろいろ

kNN　SVM　SVM　DT　RF　NN　Boost

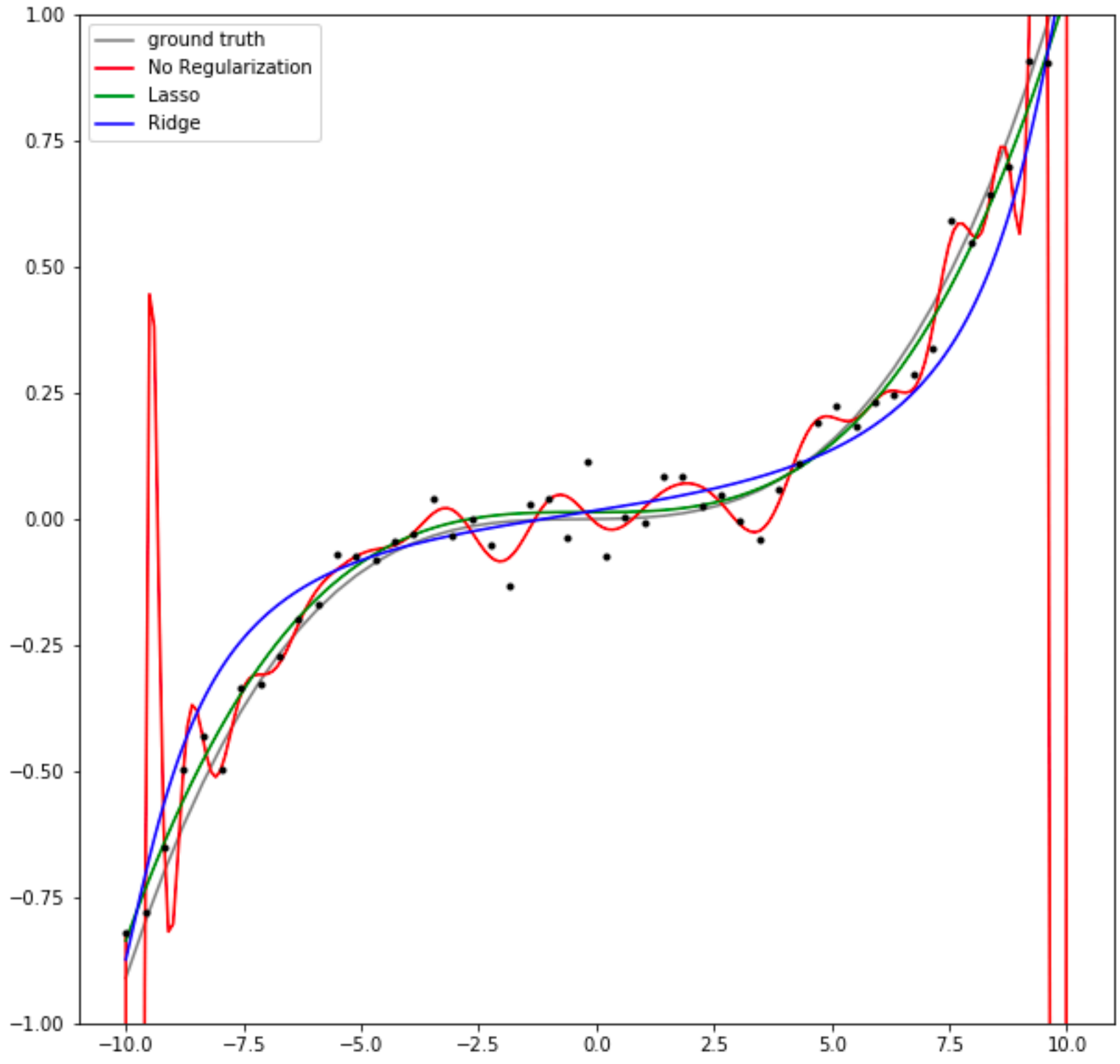


※ここでのSVMは線形モデルと同じようなものだと思ってよい

気をつけること

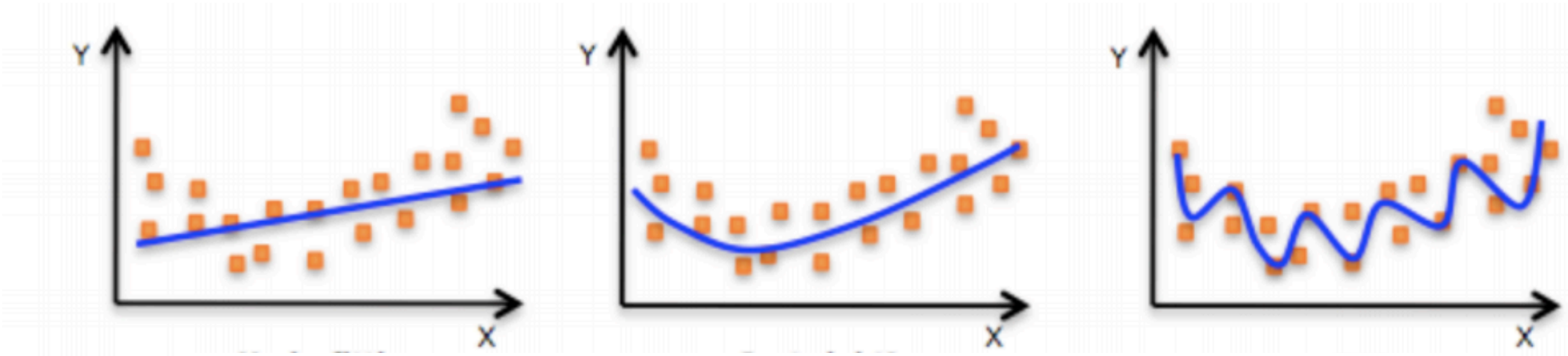
- 過学習

  (overfitting)

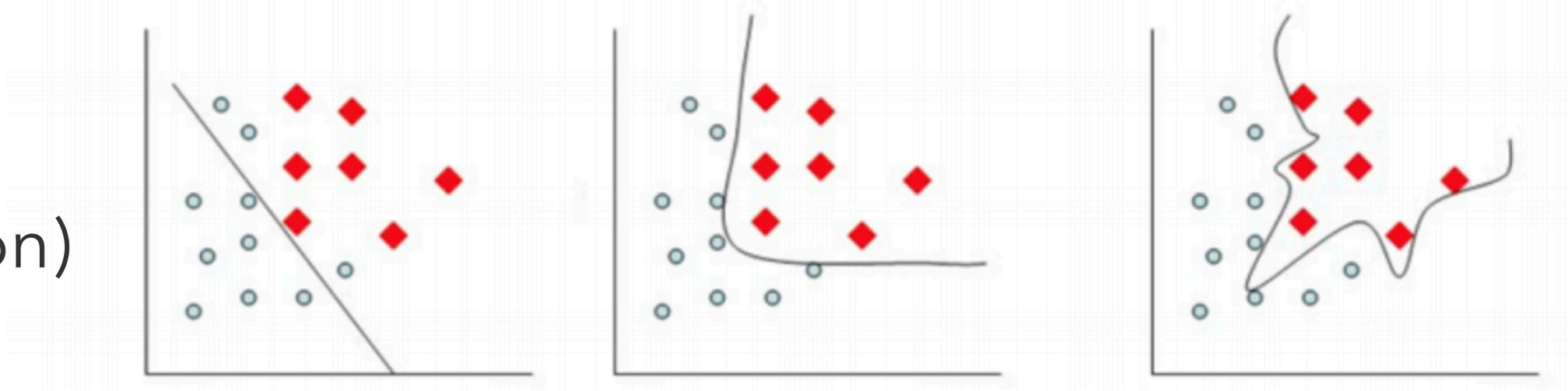


過学習(overfitting)

回帰
(regression)



分類
(classification)



Underfitting ✨Good✨ Overfitting

なぜ過学習？

- 学習データだけを正確に分類・回帰させようとする
  と、それ自体はうまくいくが、未知のデータに対しては
  性能が悪化する
- 機械学習の目的を考えると、「与えられた学習データ
  をきちんと分類(or回帰)できる」こと**ではなく**、「学習
  したあと、未知のデータが入力されてもきちんと推論
  できる」ことが重要
- ゆえに過学習したモデルは一見うまくいっているよう
  で、**何の役にも立たない**

過学習対策

- バギングなどで平均をとるとモデルの得手不得手が慣らされるので多少良くなる（ただし過学習したモデルをたくさん組み合わせてもダメ）
- 線形モデルの場合、L1正則化、L2正則化を使ってパラメータが小さくなるようにする(Lasso, Ridge)
- 学習用データ(train), 検証用データ(validation), テスト用データ(test)にデータを分けて、過学習していないかチェックする

演習

- 以下のデータセットを使って分類・回帰をしてください
  - Iris Dataset
  - Digits Dataset
  - Boston Dataset
  - Diabetes Dataset
  - Breast cancer Wisconsin dataset

演習

- Iris Dataset

```
from sklearn.datasets import load_iris
iris = load_iris()
```

- Digits Dataset

```
from sklearn.datasets import load_digits
digits = load_digits(n_class=10)
```

- Boston Dataset

```
from sklearn.datasets import load_boston
boston = load_boston()
```

- Diabetes Dataset

```
from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
```

- Breast cancer Wisconsin dataset

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
```

演習の流れ

- データを読み込んで表示してみる(printなどで)
- 分類なのか回帰なのか調べる
- 分かれていなければ、train, validation, testデータに分ける（今回はtrainとそれ以外でも良いです）
- 適当なアルゴリズムを使って検証してみる
- アルゴリズムのチューニングをする

演習の流れ - 注意点

- 精度が低い

  →まぁ最悪仕方ない　そういうこともある

- 一見精度が高いように見えるけど過学習してる

  → 絶対ダメ！　　試験なら0点