AIエンジニアリング　(2)

分類問題を解く

前回のおさらい

前回のおさらい

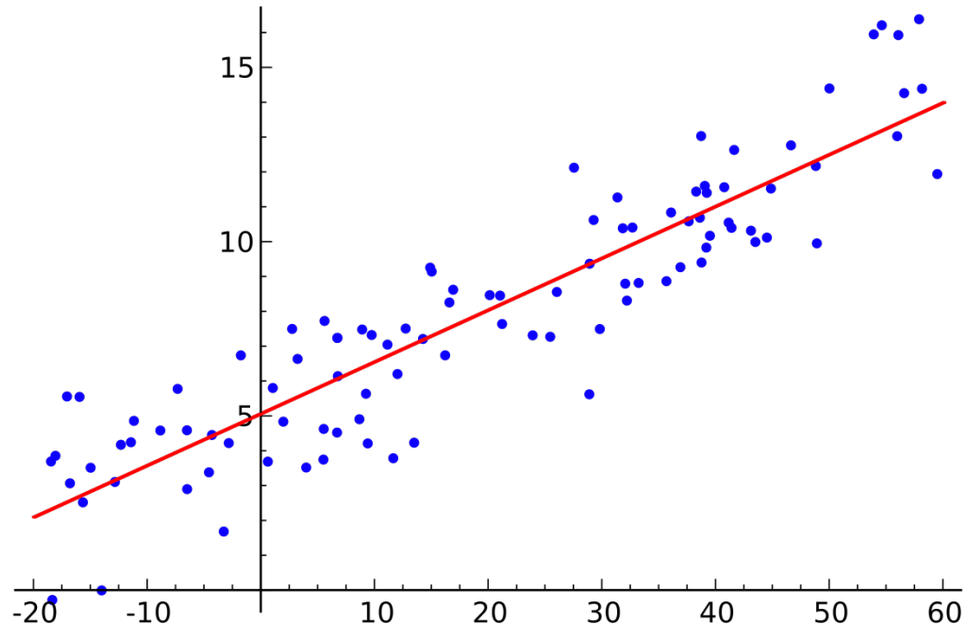
- 機械学習のうち、回帰問題を解いた
- 前回やった範囲では、回帰問題とは...

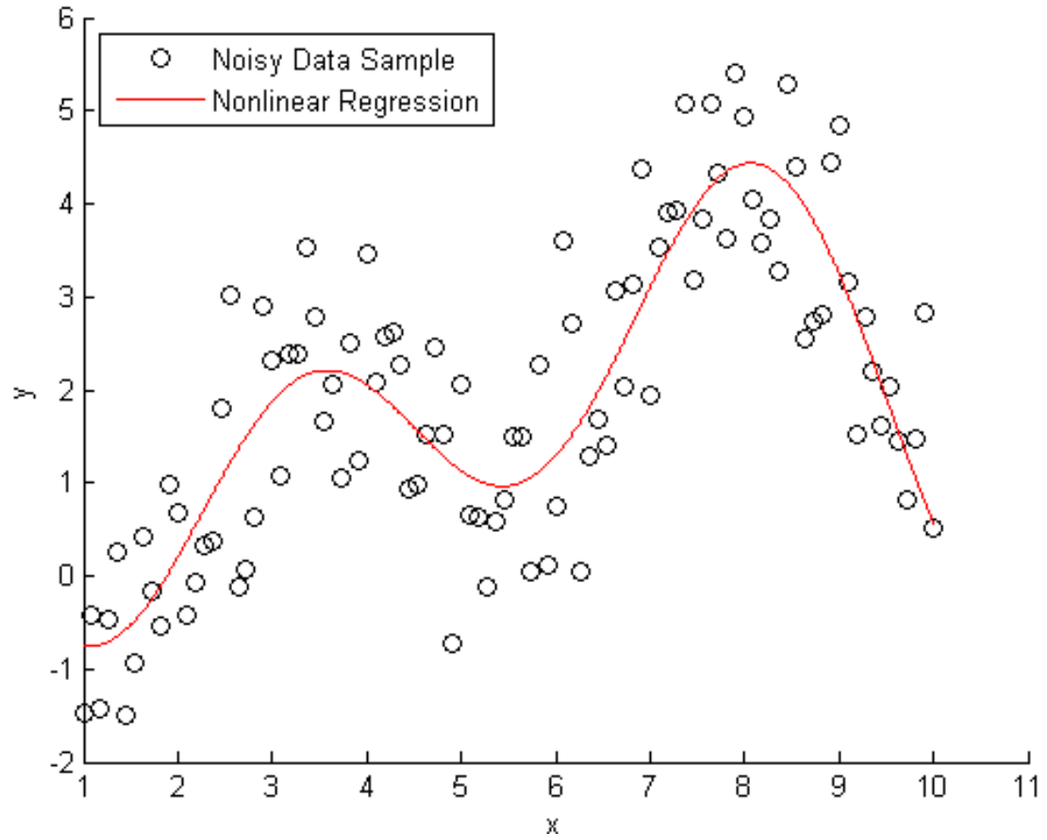
1. $y = ax_1 + bx_2 + cx_3 + \ldots$
   のような式（モデル）を考えて、
2. a, b, c, ... に当てはまる数を調整する (学習)
3. 学習データによく当てはまる係数を見つければOK

- ということをすればよかった

まだやってないこと

- 直線で近似できないような複雑なデータはどうする？
- 「よく当てはまる」かどうかをきちんと測定するにはどうすればよいか
- うまく学習するコツのようなものはあるか？





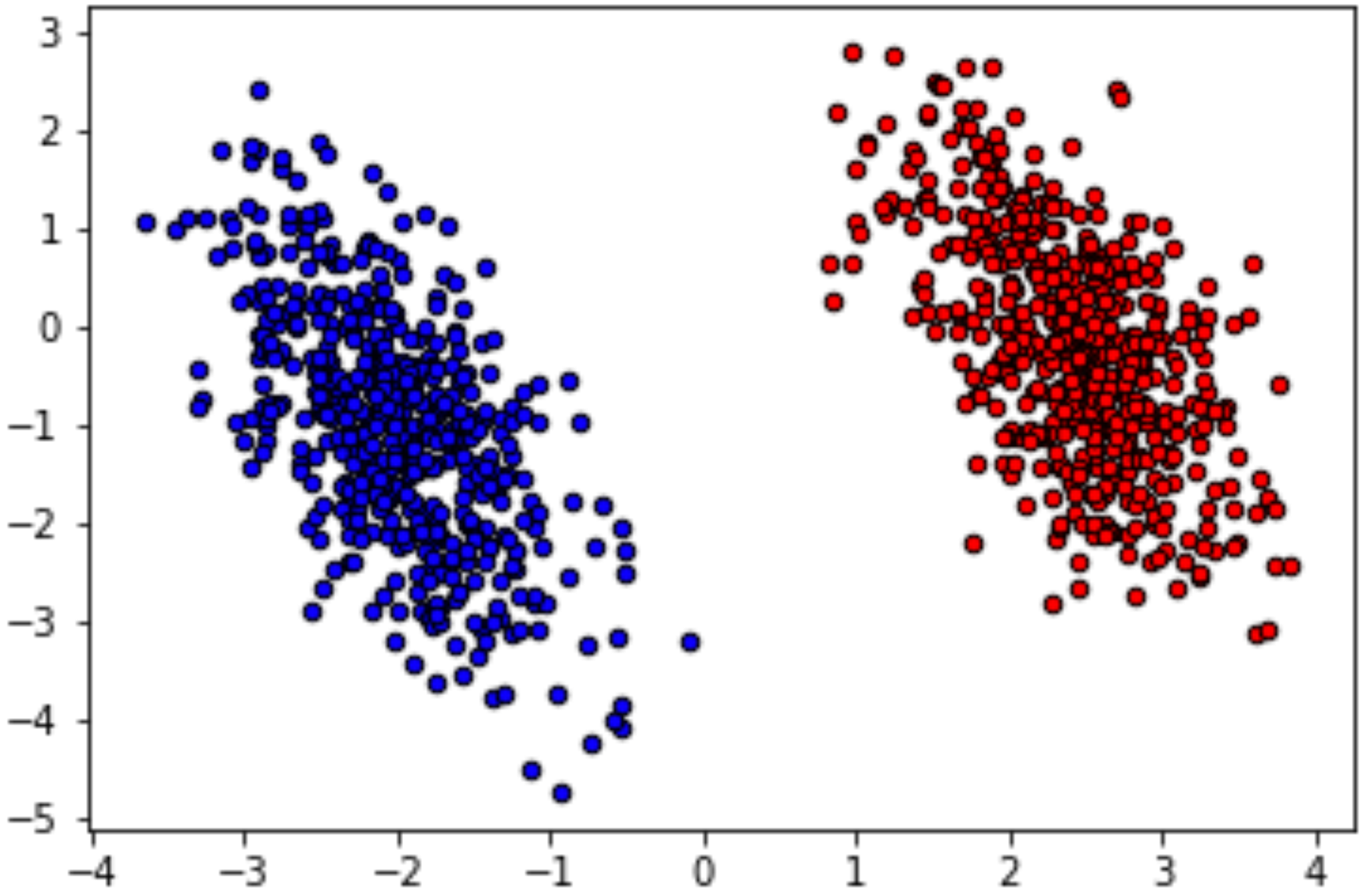
まだやってないこと

- それはさておき、まずは分類問題を解いてみる
- 回帰：
  - 数字を出力する
  - 例) Boston Dataset
- 分類：
  - YESかNOか？AかBか？のような出力

- ちなみに分類も回帰も「教師あり学習」の範囲内
  - つまり、学習データ(お手本）が与えられている

分類問題を解く

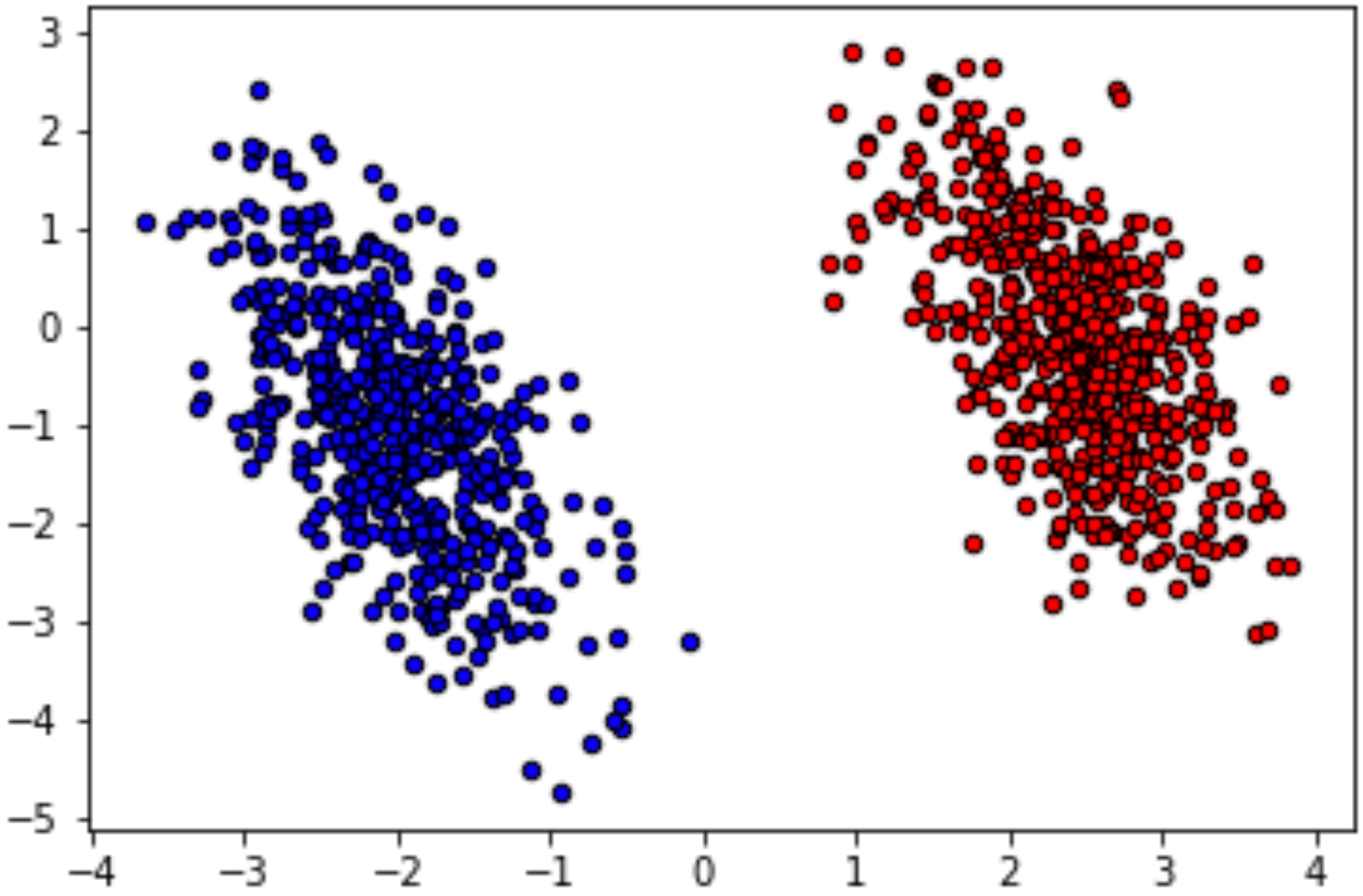
例1



線形分類器

- 実は分類も回帰と結構似ている方法で解ける
- 一番簡単な手法は、前回と同じく「直線を引く」
  - 線形分類器(linear classifier)と呼ぶ
- これは簡単だが奥が深く、実は性能も悪くない

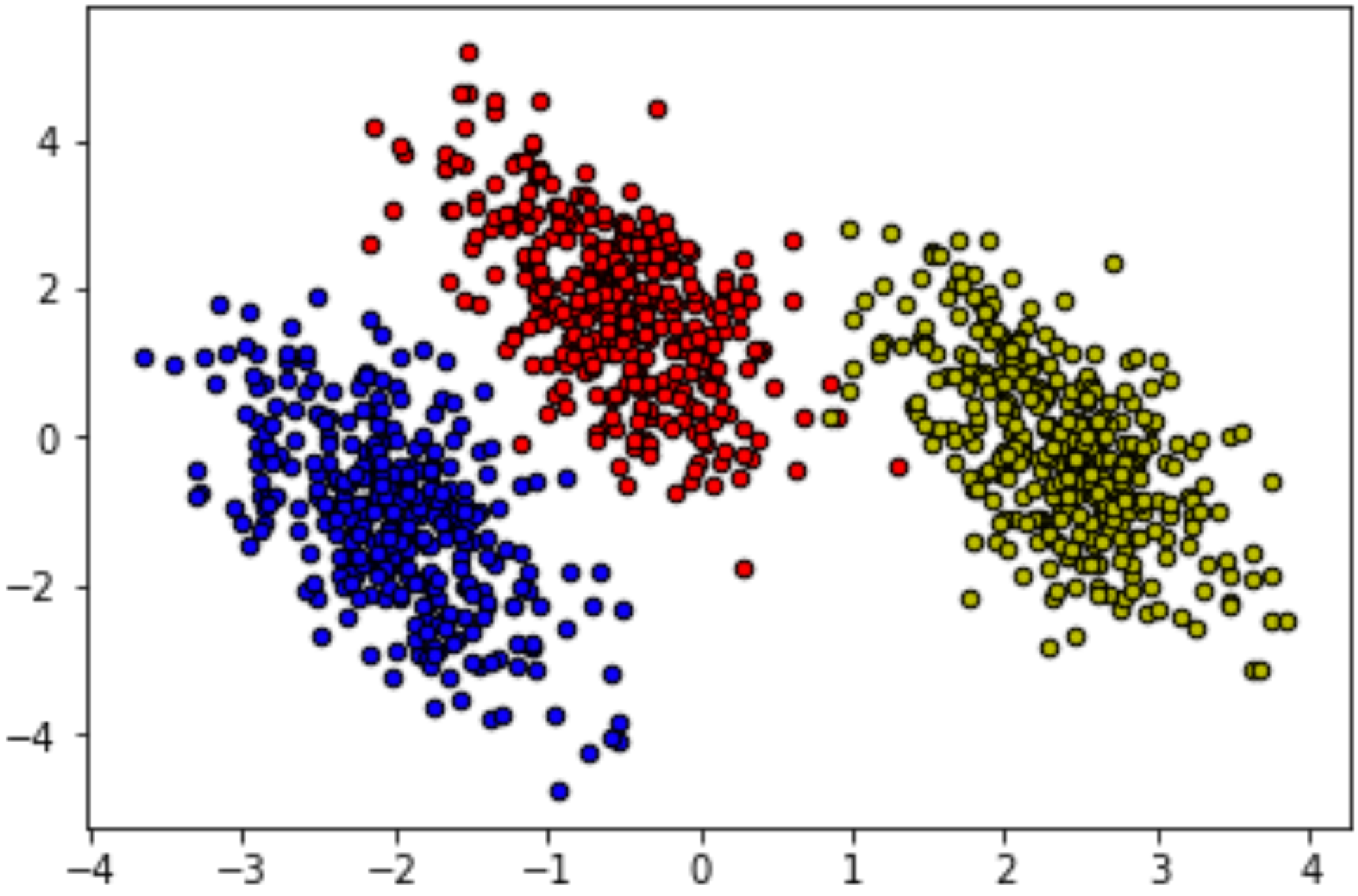
どのように直線を引けばよいだろうか



2クラス分類

- 分類する種類が2個しかない場合は、2クラス分類または**2値分類**と呼ぶ(2値分類のほうがよく言う)
- これが分類問題の基本となる形

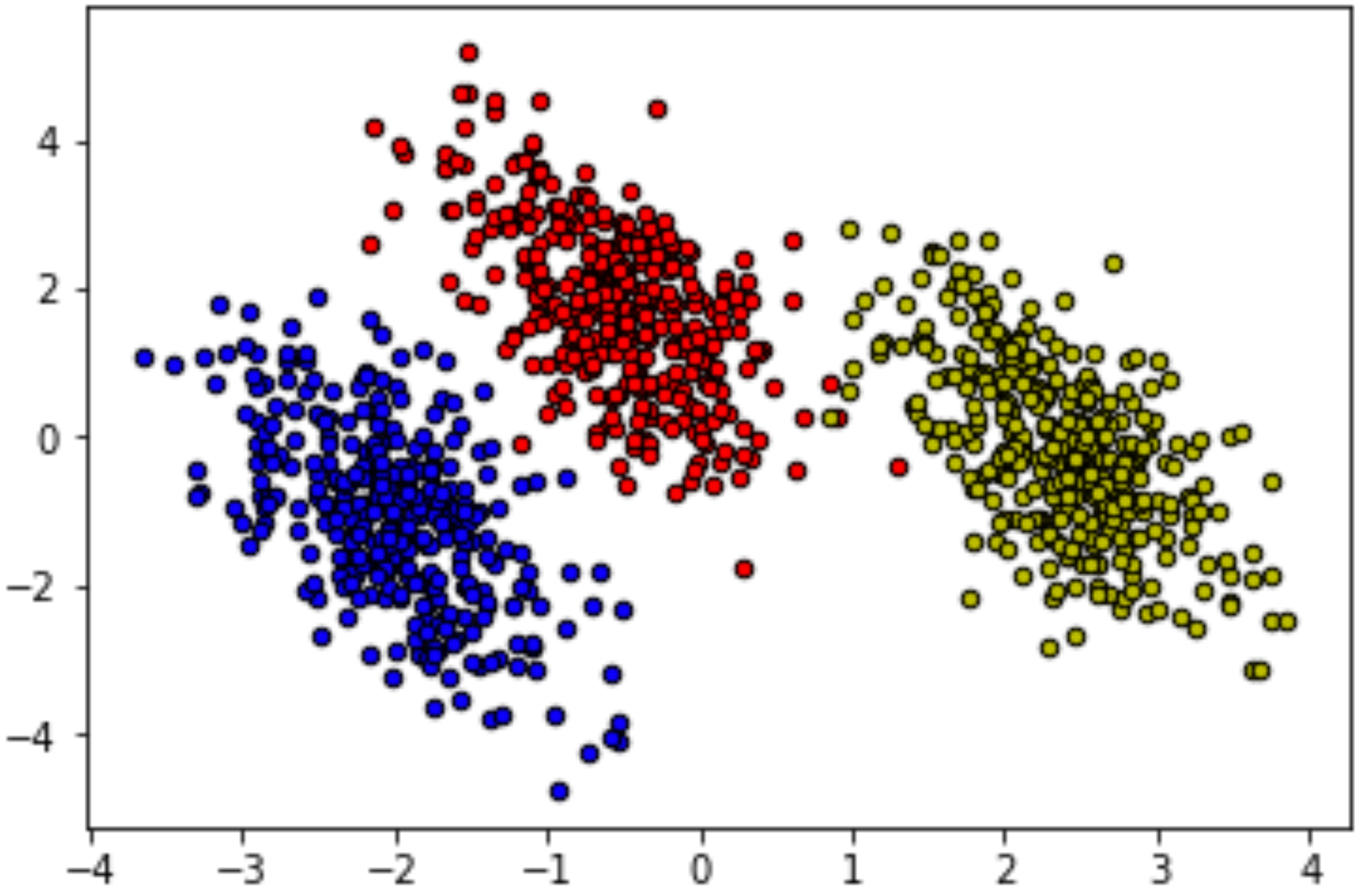
例2



多クラス分類

- 分類する種類が3つ以上の場合は多クラス分類(multi-class)とよぶ
- 多クラスになると考えるのが結構面倒くさい

どのように、何本の直線を引けばよいだろうか



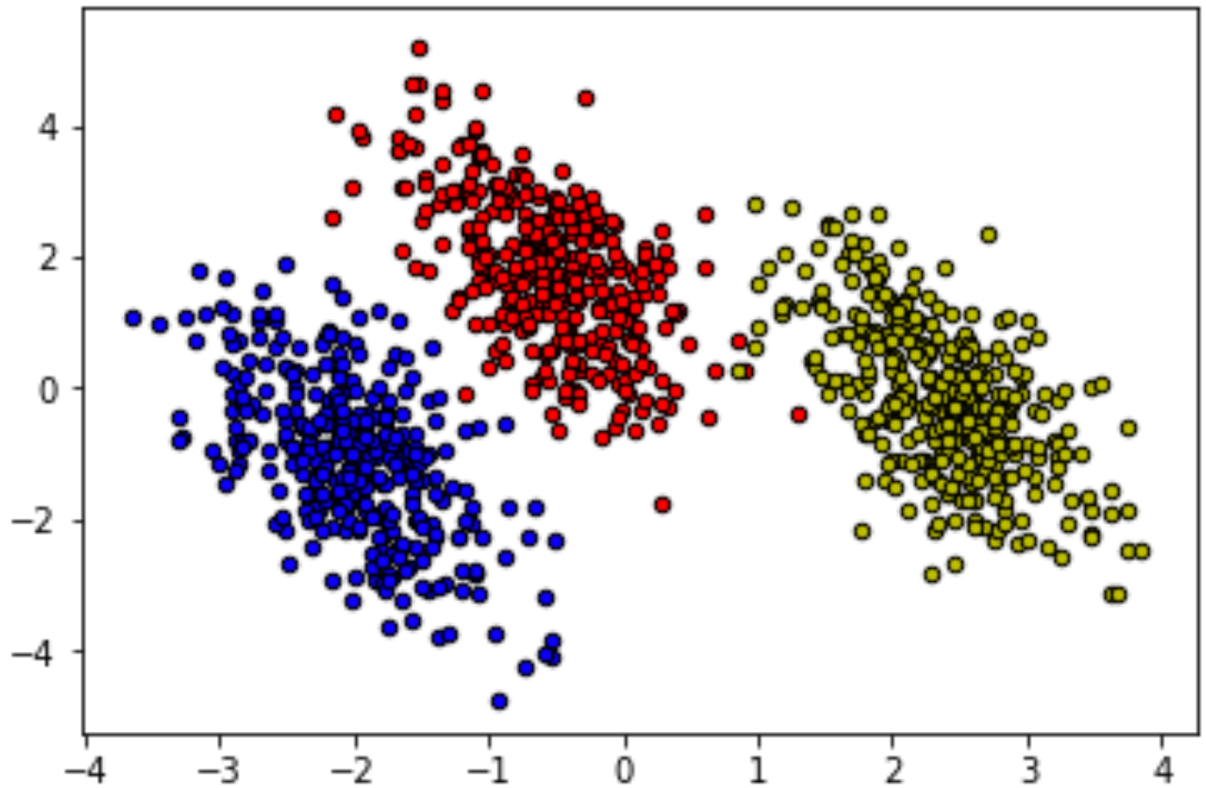
多クラス分類

- 2値分類の手法を使って多クラス分類したいとき、大雑把に分けると次の2種類の考え方がある
- 1対多 (one-versus-rest)
- 1対1 (one-versus-one)

One-versus-rest classifier

- クラスがk個あるとき、k個の分類器を作る
- 分類器kはクラスkと判定したら1を出力して、そうでないなら0を出力するようなモデルと考える
- 複数の分類器が1を出力したときにどうするかを考えないといけない

- 分類器1:
  - 青かそれ以外を判別
- 分類器2:
  - 赤かそれ以外を判別
- 分類器3:
  - 黄かそれ以外を判別



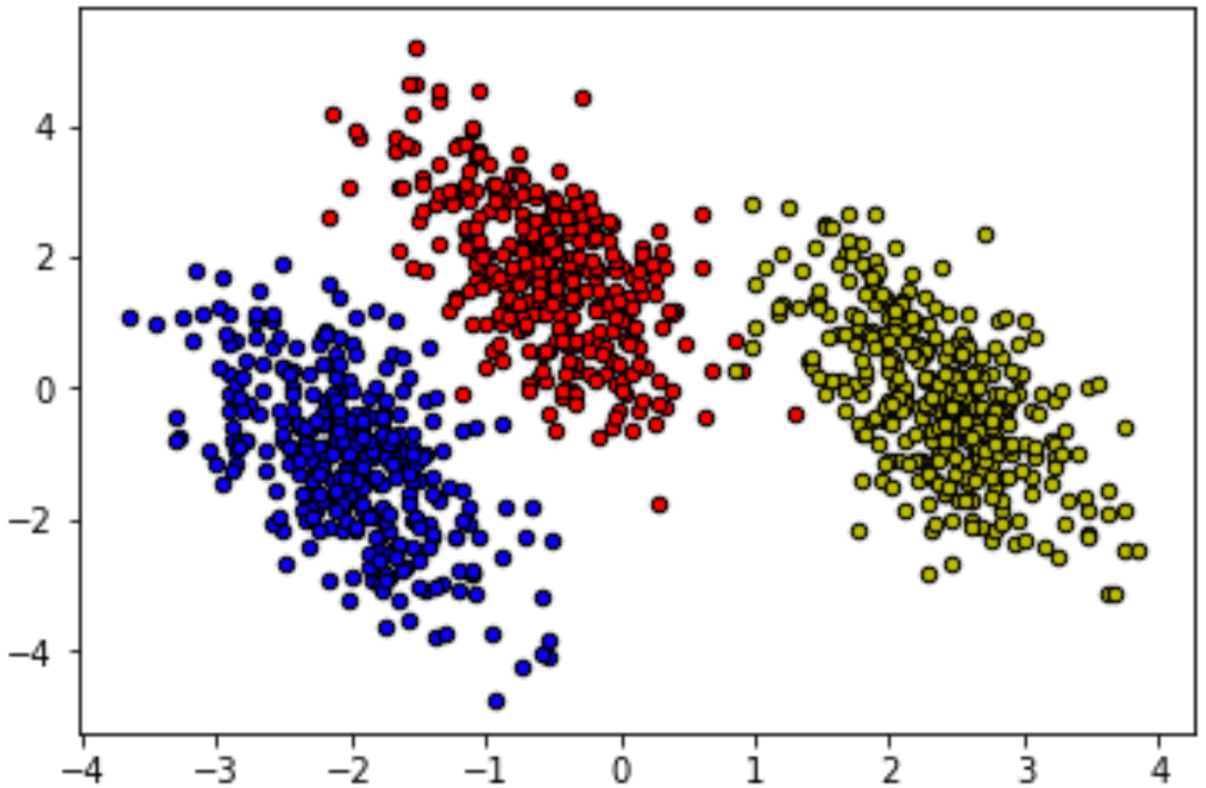
Q. 分類器1が「青」、分類器2が「赤」、分類器3が「黄色以外」と出力したらどうする？

One-versus-one classifier

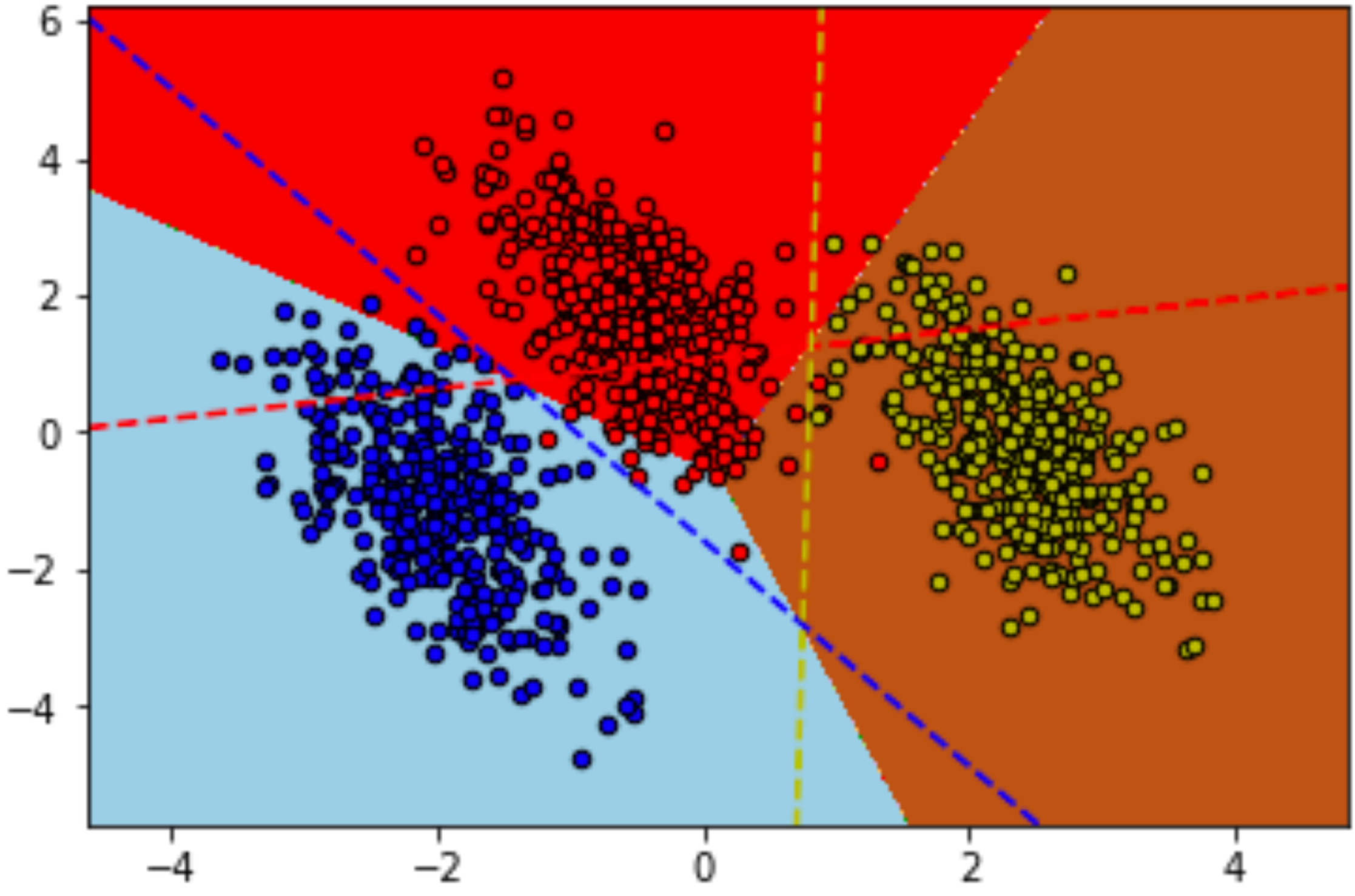
- クラスがk個あるとき、k(k-1)/2個の分類器を作る
- クラスaとクラスbを識別する分類器を作る
- 最終的なクラスは多数決で決める
  - 赤と青のどっち？→赤
  - 青と黄のどっち？→黄　　→「赤」と判定
  - 黄と赤のどっち？→赤
- ただこの方法も循環ができて決定できない場合がある

この場合はどうする？

- 分類器1(赤or青):
  - 「赤」と出力
- 分類器2(青or黄):
  - 「青」と出力
- 分類器3(黄or赤):
  - 「黄」と出力



循環が発生して決定できない


6
4
2
0
−2
−4
−4
−2
0
2
4

二値分類器を使うのは面倒臭そう

最初から多値分類できるような手法はないか？

- ある
- 二値分類器はそれはそれで優秀だが、多値分類に応用したい場合はone-versus-restでもone-versus-oneでも問題が起きがち
- 最初から多値分類できる手法もあるので、それを使うと良い

じゃあ最初から多値分類だけ使えばよいのでは？

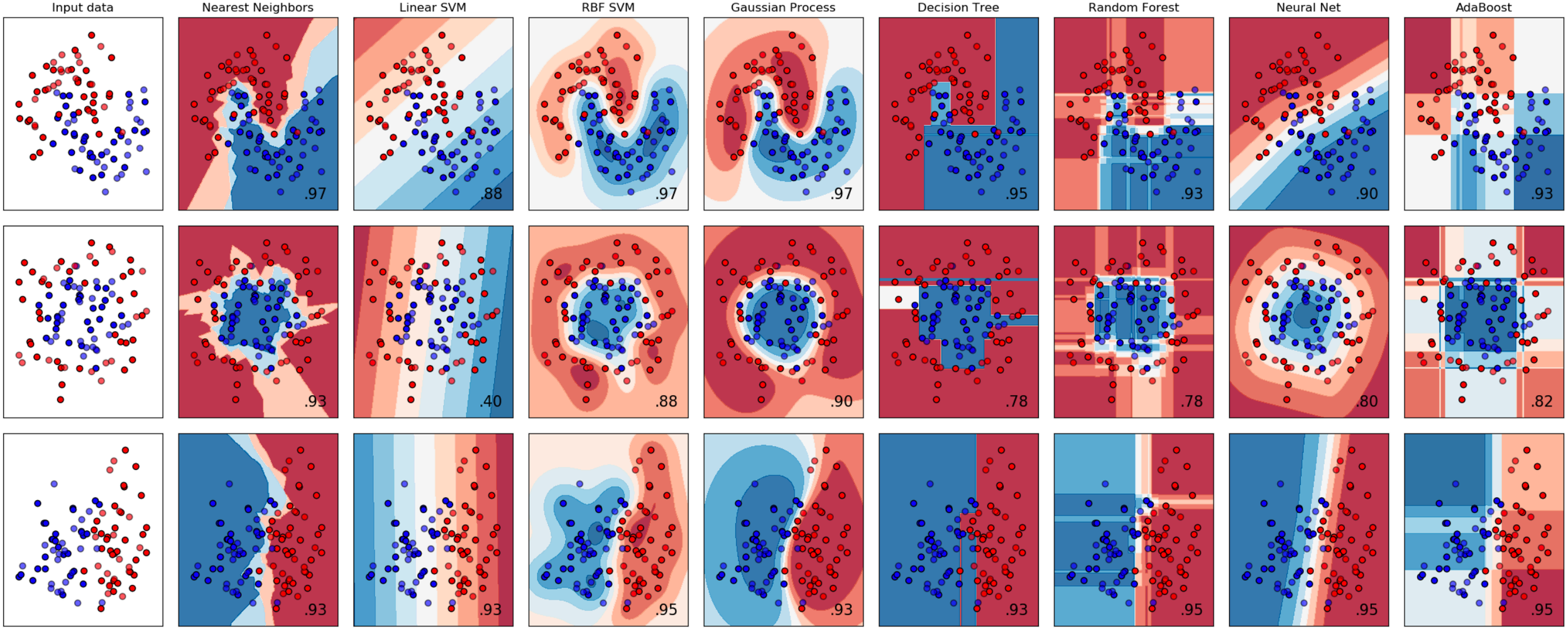
- そうでもない
- 機械学習の場合、シンプルな手法のほうが強い
- つまり、シンプルな手法ではどうしようもないときだけ複雑な手法を選ぶほうがよい
- このあたりはまた後日の講義で詳しく見ていく
  - ノーフリーランチ定理
  - 過学習(overfitting)

有名な多値分類のアルゴリズム

- k近傍法 (k-Nearest Neighbor)
- 決定木 (Decision Tree)
    - Random Forest
    - Gradient Boosting
- ニューラルネットワーク(Neural Network)
    - 多層パーセプトロン (Multilayer Perceptron)
        - ディープラーニング (Deep Learning)

分類器いろいろ

kNN SVM SVM DT RF NN Boost



k近傍法(k-Nearest Neighbor, kNN)

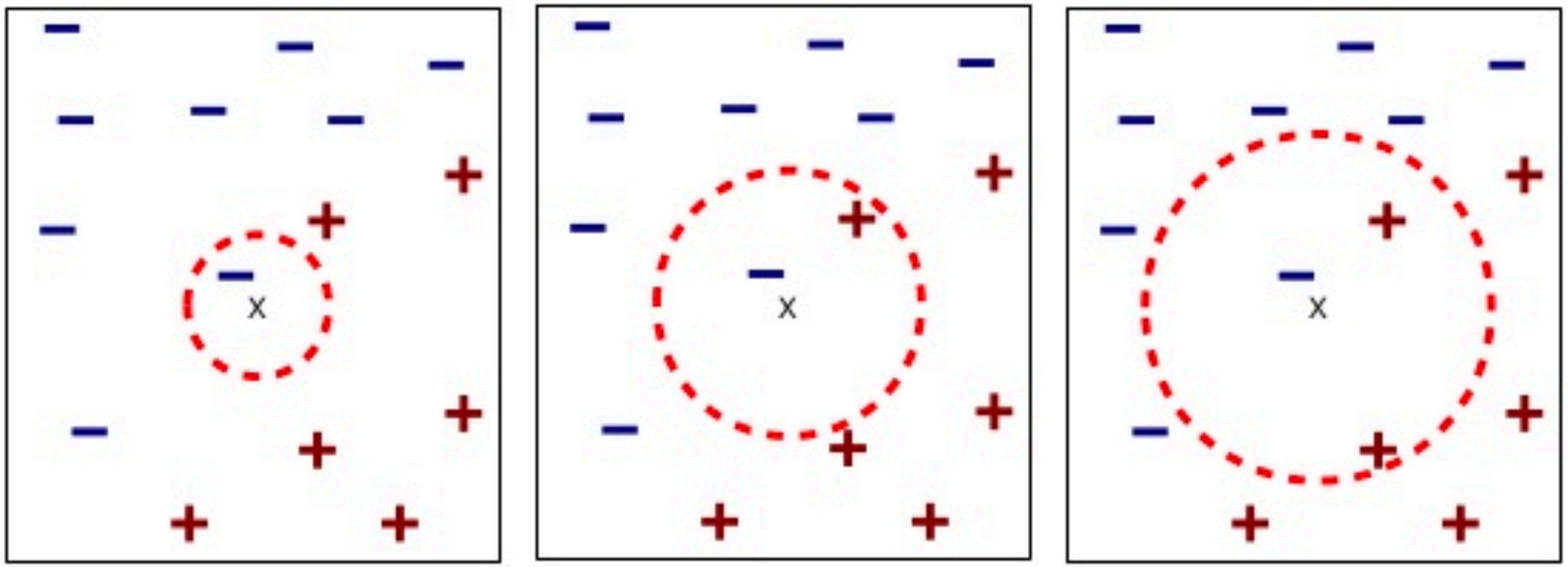
メリット

- 非線形分類可能
- 理論的に最小に近い誤り率に漸近
- 理論、実装が楽
- オンライン学習可
- 学習が不要
- 回帰もできる

デメリット

- どんどん遅くなる
- メモリもどんどん使う
- ノイズに弱い

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



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

普通はkは奇数にする

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

Wine dataset

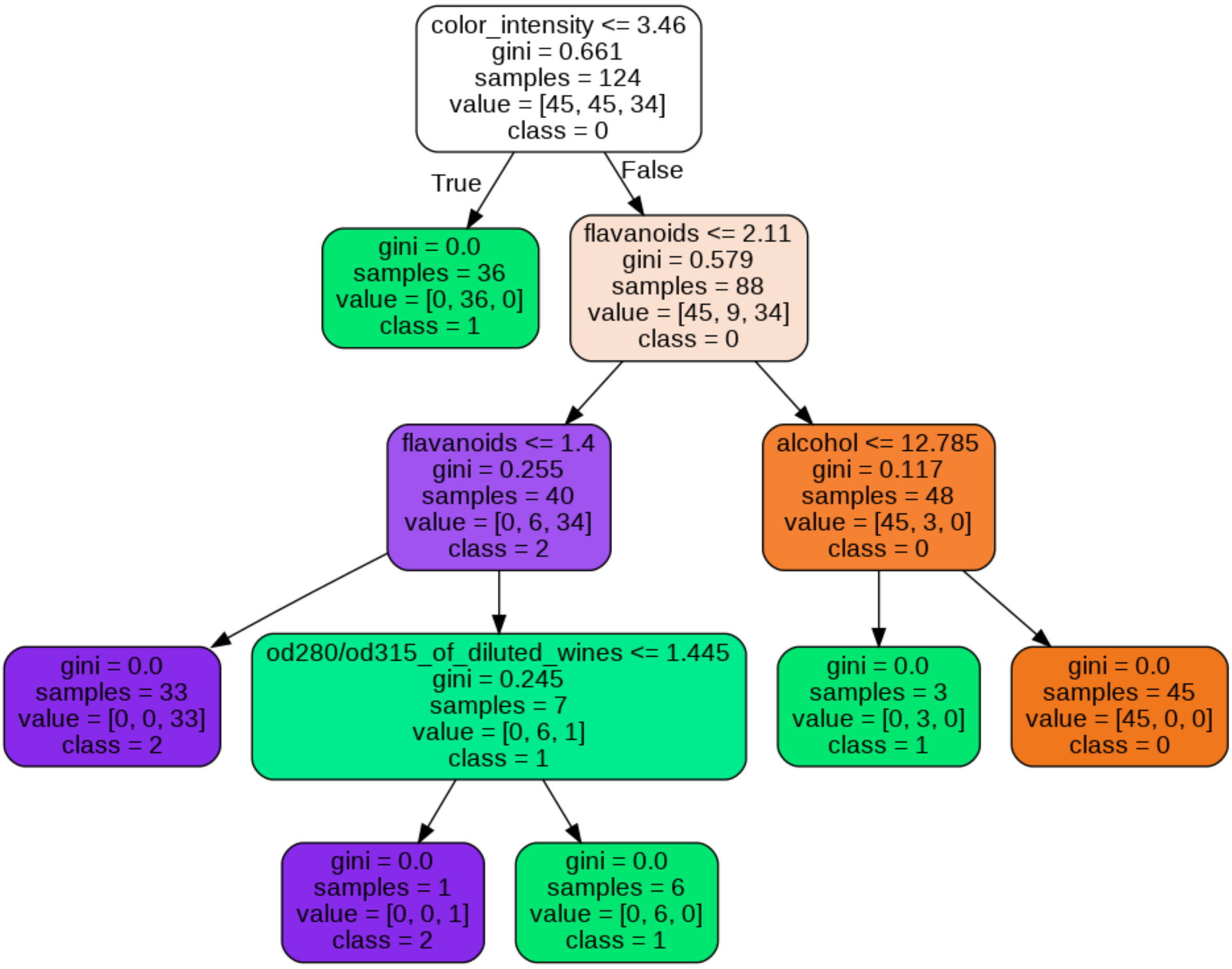
説明変数

1. alcohol アルコール濃度
2. malic_acid リンゴ酸
3. ash 灰（？
4. alcalinity_of_ash 灰のアルカリ成分（？
5. magnesium マグネシウム
6. total_phenols 総フェノール類量
7. flavanoids フラボノイド（ポリフェノールらしい）
8. nonflavanoid_phenols 非フラボノイドフェノール類
9. proanthocyanins プロアントシアニジン（ポリフェノールの一種らしい）
10. color_intensity 色の強さ
11. hue 色合い
12. od280/od315_of_diluted_wines ワインの希釈度合い
13. proline プロリン（アミノ酸の一種らしい）

目的変数

14. ワインの品種

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	od280/od315
0	14.23	1.71	2.43	15.6	127.0	2.80	3.06	0.28	2.29	5.64	1.04	
1	13.20	1.78	2.14	11.2	100.0	2.65	2.76	0.26	1.28	4.38	1.05	
2	13.16	2.36	2.67	18.6	101.0	2.80	3.24	0.30	2.81	5.68	1.03	
3	14.37	1.95	2.50	16.8	113.0	3.85	3.49	0.24	2.18	7.80	0.86	
4	13.24	2.59	2.87	21.0	118.0	2.80	2.69	0.39	1.82	4.32	1.04	


color_intensity <= 3.46
gini = 0.661
samples = 124
value = [45, 45, 34]
class = 0
True
False
gini = 0.0
samples = 36
value = [0, 36, 0]
class = 1
flavanoids <= 2.11
gini = 0.579
samples = 88
value = [45, 9, 34]
class = 0
flavanoids <= 1.4
gini = 0.255
samples = 40
value = [0, 6, 34]
class = 2
alcohol <= 12.785
gini = 0.117
samples = 48
value = [45, 3, 0]
class = 0
gini = 0.0
samples = 33
value = [0, 0, 33]
class = 2
od280/od315_of_diluted_wines <= 1.445
gini = 0.245
samples = 7
value = [0, 6, 1]
class = 1
gini = 0.0
samples = 3
value = [0, 3, 0]
class = 1
gini = 0.0
samples = 45
value = [45, 0, 0]
class = 0
gini = 0.0
samples = 1
value = [0, 0, 1]
class = 2
gini = 0.0
samples = 6
value = [0, 6, 0]
class = 1

Random Forest 系

メリット

- 重要なハイパパラメータが少ない
- 過学習しづらい
- 非線形分類できる（回帰もできる）
- 解釈容易性
- 精度が高い
- 変数の重要度がわかる
- RF, ERT は分散学習が可能
- オンライン学習はできなくもない

デメリット

- 特にない
- DLには精度で負ける
- データや変数が少ないときは変な結果が出ることがある

著名なライブラリ

- 今は scikit-learn を使っているけど、RF系、特にGBDTは他のライブラリを使うことも多い
  - XGBoost(Distributed Machine Learning Community)
  - LightGBM(Microsoft)
- これらはライブラリ（実装名）であってアルゴリズムの名前ではない
- アルゴリズムはGradientBoosting(GBDT)