

機械学習 第6回 演習1

兵庫県立大学 社会情報科学部

川嶋宏彰

kawashima@sis.u-hyogo.ac.jp

本日の講義内容

2

- 目的：モデルの汎化性能と表現力のトレードオフを解決
 - 過学習を防ぎたい
- 交差検証とハイパーパラメタの調整
 - 交差検証法 (cross validation)
 - グリッドサーチ
- 演習の解説
- レポートの解説

2.1.3節

7.5節

交差検証は
• モデルの評価
• パラメタ調整
どちらにも使う

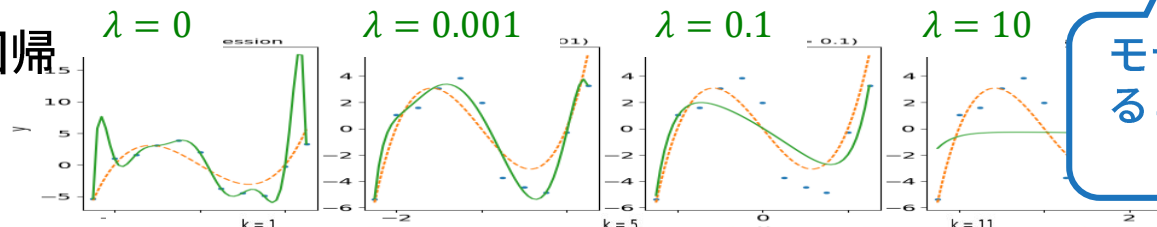
実装では「Pythonではじめる機械学習」
(オライリー) の5章もお勧め
(このスライドのコードはこの本を参考にしています)

bias と variance は一方を下げるともう一方が上がる₃

- λ, α, k, C などをハイパーパラメタと呼ぶ

正則化の度合いを調整

Ridge回帰

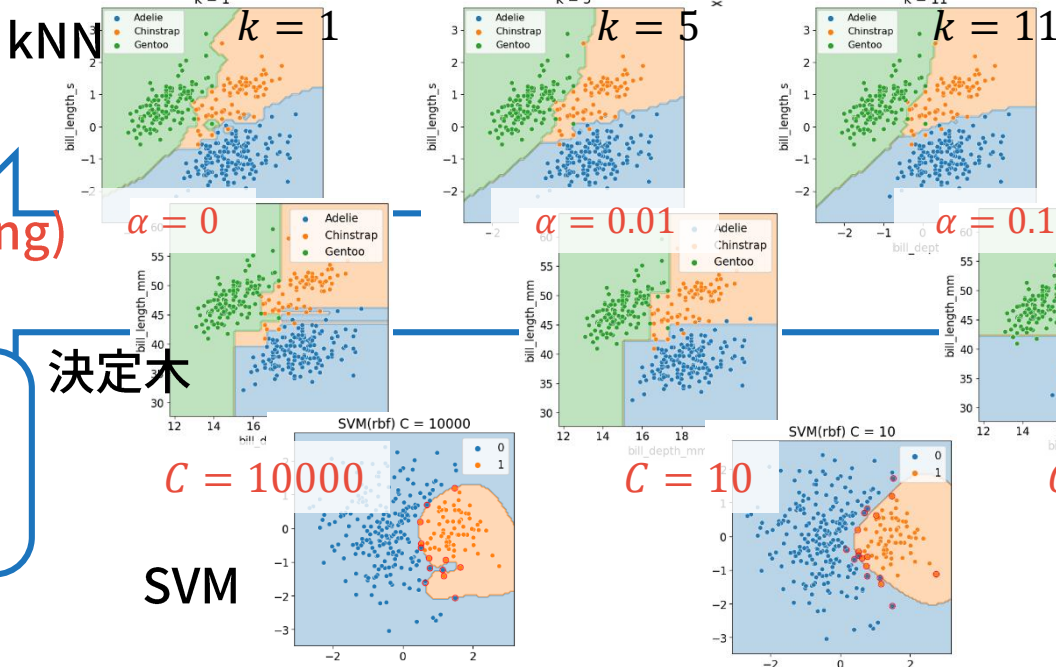


モデルに制限を加えることを「正則化」という

Variance 大
(High variance)

過学習 (overfitting)
汎化能力小

少しの学習データの変化で予測が大きく変化



Bias 大

(High bias)
表現力小
適合不十分

学習データセットの分割

- ハイパーパラメタの調整ではデータセットを3つに分割
 1. 訓練データ (training set): あるハイパーパラメタでモデル推定
 2. 検証データ (validation set) (development set と呼ばれる)
 - 1で使ったハイパーパラメタの評価
 - 汎化性能は高いか (過学習をうまく防いでいるか) を検証
 3. テスト用データ (test set): 調整後の結果を評価



「テスト」と「調整」のための分割

5

- 学習に用いていないデータで評価することが基本！

- 学習と学習結果の評価に使うデータを分ける

`train_test_split()`

Training set (訓練データ)

Test set

- ハイパーパラメタの「調整」ではハイパーパラメタの良さを評価

あるハイパーパラメタ
でモデルを当てはめ

ハイパーパラメタ
の検証

学習用: 次スライド以降

調整後モデルの評価用

Training set

Validation
set

Test set

いろいろなハイパーパラメタで検証

まずこの中でも `train_test_split()` してみる

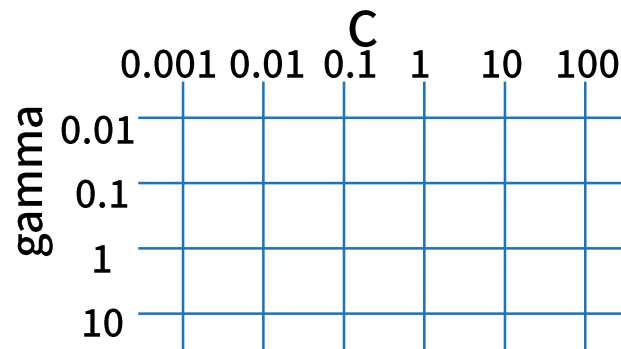
グリッドサーチでハイパーパラメタ調整

6

・グリッドサーチは各ハイパーパラメタの組合せを比較

```
X_trainval, X_test, y_trainval, y_test = train_test_split(X, y, test_size=0.2)
X_train, X_valid, y_train, y_valid = train_test_split(
    X_trainval, y_trainval, test_size=0.2)
```

```
best_score = 0 # 最大値の記録用
for gamma in [0.01, 0.1, 1, 10]: # RBFのパラメタ
    for C in [0.001, 0.01, 0.1, 1, 10, 100]:
        svm = SVC(gamma=gamma, C=C)
        svm.fit(X_train, y_train)
        score = svm.score(X_valid, y_valid)
        if score > best_score:
            best_score = score
            best_parameters = {'C': C, 'gamma': gamma}
```



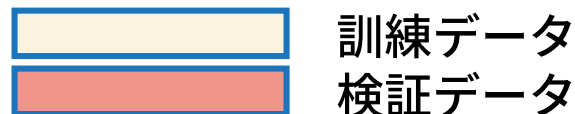
これだと一つの訓練-検証ペアでのスコアなので不十分
(たまたまそのペアで良かった／悪かったかもしれない)

交差検証 (CV)
に置き換える

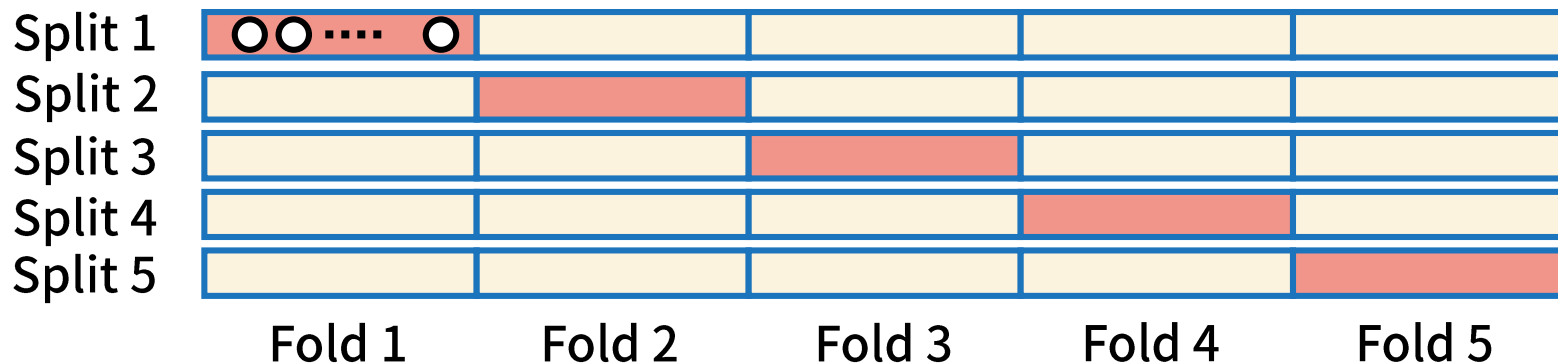
交差検証法 (cross-validation)

7

- 交差検証法



- 5-fold の例: 5通りの分割結果(split)が得られる



- 各splitで訓練 (training) と検証 (validation) を行う

分割

- Leave-one-out (LOO)

- 1-fold あたり1個のデータ
- データ数が少ない場合など

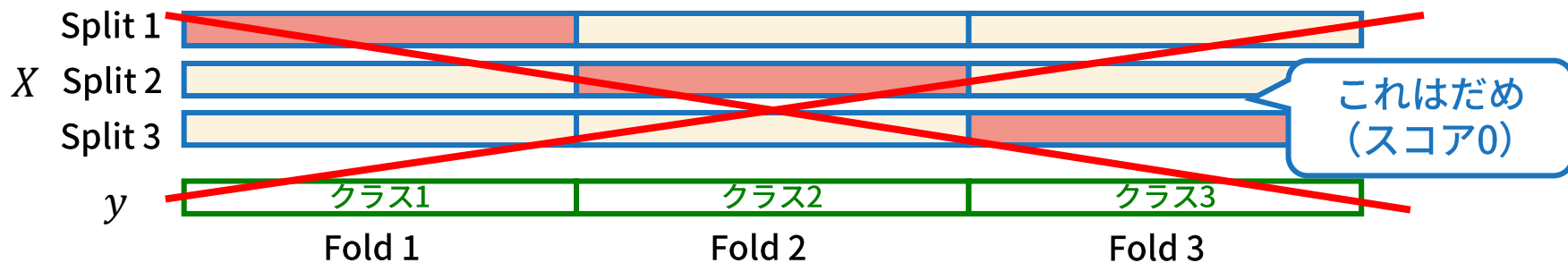
```
from sklearn.model_selection
import cross_val_score,
Kfold, LeaveOneOut
```

階層化されたKFold

8

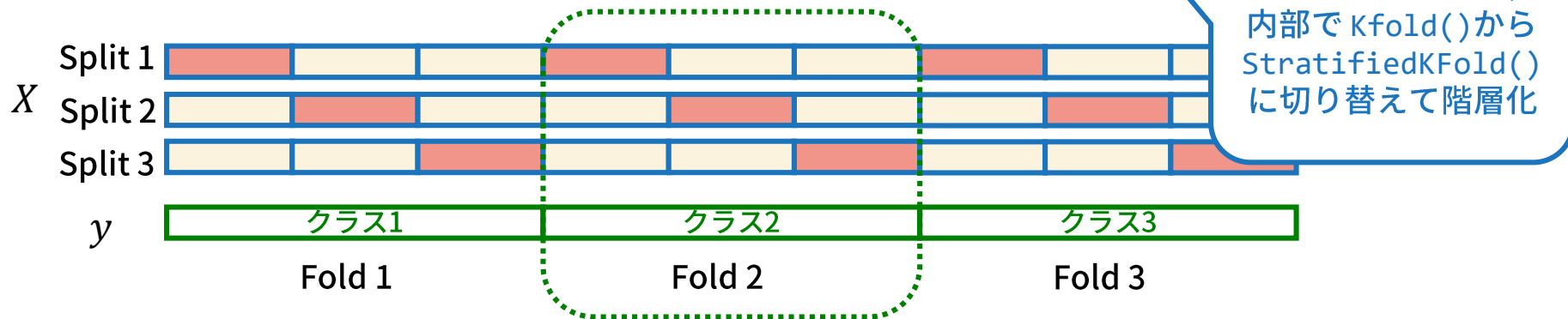
- 各クラスでバランスよく訓練-検証データを作るために階層化する

`Kfold()` はデフォルトで前から順に分割 → クラスの並びによってはまずい



階層化 (stratified) された3-foldデータ `StratifiedKfold()`

`cross_val_score(clf, X_trainval, y_trainval, cv=3)`



グリッドサーチでハイパーパラメタ調整

9

- 交差検証によるグリッドサーチが標準的

```
X_trainval, X_test, y_trainval, y_test = train_test_split(X, y,  
    test_size=0.2) # まずテストデータを切り離しておく
```

```
for gamma in [0.01, 0.1, 1, 10]: # 4通り  
    for C in [0.001, 0.01, 0.1, 1, 10, 100]: # 6通り  
        svm = SVC(gamma=gamma, C=C)  
        cv_scores = cross_val_score(svm, X_trainval, y_trainval, cv=5)  
        score = np.mean(cv_scores)  
        if score > best_score:  
            best_score = score  
            best_parameters = {'C': C, 'gamma': gamma}
```

scoringで"roc_auc",
"f1" なども指定可能

何回学習されるか？

scikit-learn ではこれを簡単に行うクラスが用意されている

交差検証でのグリッドサーチを一気に行う GridSearchCV()

10

```
X_train, X_test, y_train, y_test = train_test_split(X, y,  
                                                    test_size=0.2) # まずテストデータを切り離しておく
```

```
# SVM(RBF)のハイパーパラメタ候補
```

```
param_grid = {  
    'gamma': [0.01, 0.1, 1, 10], # 4通り  
    'C': [0.001, 0.01, 0.1, 1, 10, 100] # 6通り  
}
```

変数名はこれまでの `X_trainval` などを
ここでは `X_train` としているので注意

scoringで
"roc_auc" など
も指定可能

```
grid_search = GridSearchCV(SVC(), param_grid, cv=5)
```

```
# 各グリッドで交差検証
```

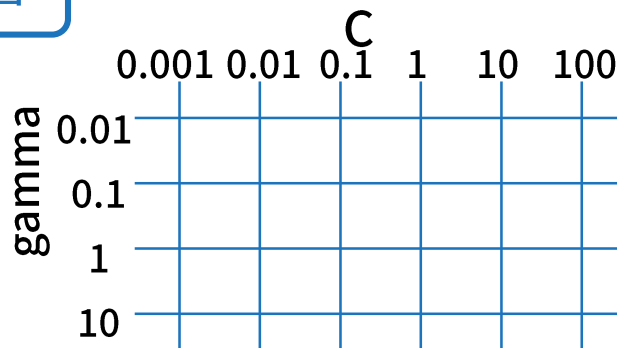
内部で 4 x 6 x 5 回学習

```
grid_search.fit(X_train, y_train)
```

```
# サーチ中のベストパラメタと交差検証スコア
```

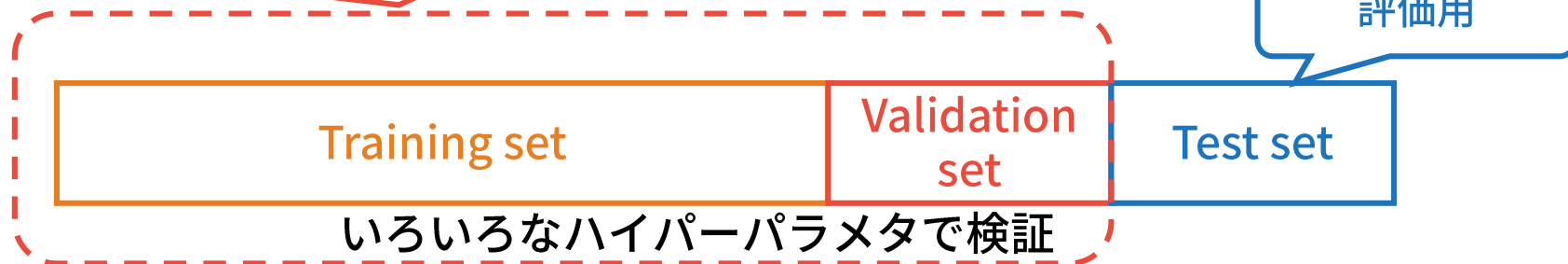
```
print(grid_search.best_params_)
```


```
print(grid_search.best_score_)
```



- ベストなハイパーパラメタが見つかったあとどうするか？
 - 検証データを切り分けずにテストデータ以外全部で再度学習
 - `GridSearchCV()` は再学習までやってくれる
(もちろん自前で分類器を学習し直してもよい)
 - ベストパラメタで再学習したモデルでスコアや予測を計算するなら
`grid_search.score()` や `grid_search.predict()`
(注意) `grid_search.best_score_` は再学習前なので検証データが学習に利用されていない

調整後のパラメタを使ってこのデータ全部で再学習



- テストデータの分割は一通りでよいのか？
テストも交差検証で行う必要があるのではないか？
 - 二重化（入れ子）の交差検証を行うこともある (nested CV)
 - 外側のループ：学習用データとテスト用データの分割
 - 内側のループ：訓練・検証データの分割（ハイパーパラメタ調整用）
 - 学習回数：(グリッド数) x (外側の交差検証のK) x (内側の交差検証のK)
時間がかかることが多い（今日の演習の「発展」）
- 研究によってはある程度簡略化する場合も
 - まだ十分なデータ数がない段階での評価  LOOもよく用いられる
 - テストは実際にアプリケーションを展開して行う場合など、いろいろ

- 演習用コード

- <https://colab.research.google.com/drive/1-apPY7lto-T4ElH97nuOmhBzaMwRDfhB?usp=sharing>

1. 上のURLから ipynb をダウンロード

- ファイル>ダウンロード>.ipynb をダウンロード

2. 自分のJupyter Notebook で実行

- 内容

1. 様々な評価指標

- 第4回スライドの宿題2（混同行列やAUCなど）に対応

2. グリッドサーチによるハイパーパラメタの調整

- ✕切：5/26（金）ユニパで提出（詳細は ipynb を確認）

googleアカウントでログインし、
ファイル>ドライブにコピーを保存
した後に、Colaboratoryで実行し、
ipynb をダウンロード・提出するのも可