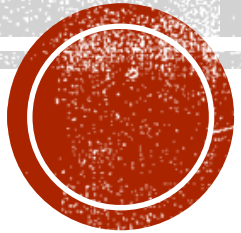
Pythonではじめる機械学習



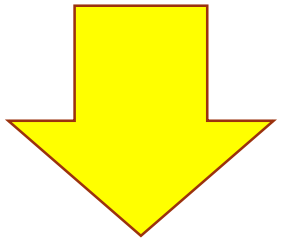
5章　モデルの評価と改良
　5.2　グリッドサーチ

　　5.2.1　単純なグリッドサーチ
　　5.2.2　パラメータの過剰適合の危険性と検証セット
　　5.2.3　交差検証を用いたグリッドサーチ

15T4061L　南濱　篤

5.2 グリッドサーチ

- すべてのモデル、すべてのデータセットで、
  モデルの重要なパラメータに対し、
  最良の汎化性能を与える設定を見つけなければならない。



最もよく用いられる方法

- **グリッドサーチ**
  基本的にはパラメータのすべての組合せを試す方法。

（例：RBFを用いたSVMの場合、gamma と C はそれぞれ 6 つの値を
とるとすると、すべての組合せ36通りを試す。）

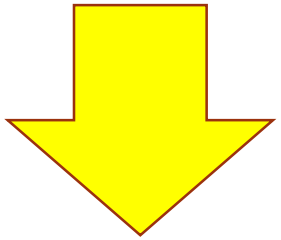
5.2.1 単純なグリッドサーチ

単純なグリッドサーチ

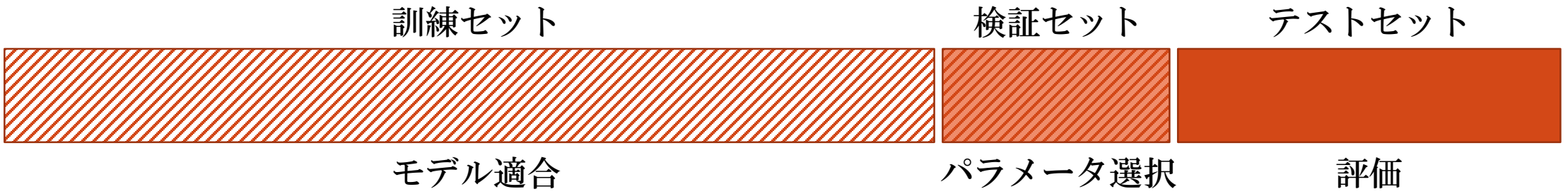
- パラメータに対するforループによる実装
- ループの中で、それぞれのパラメータの組合せに対して
  クラス分類器を訓練して評価する

5.2.2 パラメータの過剰適合の危険性と検証セット

- テストデータをチューニングに使ってしまうと、
  モデルの精度評価に使えなくなる。
- 新しいデータに対して必ずしも当てはまらない。



- 検証セットを用意する。



5.2.2 パラメータの過剰適合の危険性と検証セット

- パラメータのチューニングにテストセットを使用

In:

```
for gamma in [0.001, 0.01, 0.1, 1, 10, 100]:
 for C in [0.001, 0.01, 0.1, 1, 10, 100]:
  # それぞれのパラメータの組合せに対してSVCを訓練
  svm = SVC(gamma=gamma, C=C)
  svm.fit(X_train, y_train)
  # SVCをテストセットで評価
  score = svm.score(X_test, y_test)
  # 良いスコアだったらスコアとパラメータを保存
  if score > best_score:
   best_score = score
    best_parameters = {'C': C, 'gamma': gamma}
```

Out:

```
Size of training set: 112   size of test set: 38
Best score: 0.97
Best parameters: {'C': 100, 'gamma': 0.001}
```

5.2.2 パラメータの過剰適合の危険性と検証セット

- パラメータのチューニングに検証セットを使用

In:

```
for gamma in [0.001, 0.01, 0.1, 1, 10, 100]:
 for C in [0.001, 0.01, 0.1, 1, 10, 100]:
  # それぞれのパラメータの組合せに対してSVCを訓練
  svm = SVC(gamma=gamma, C=C)
  svm.fit(X_train, y_train)
  # SVCを検証セットで評価
  score = svm.score(X_valid, y_valid)
  # 良いスコアだったらスコアとパラメータを保存
  if score > best_score:
   best_score = score
    best_parameters = {'C': C, 'gamma': gamma}
```

Out:

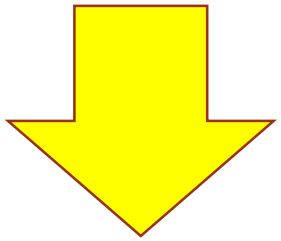
```
Size of training set: 84   size of validation set: 28   size of test set: 38


Best score on validation set: 0.96
Best parameters:  {'C': 10, 'gamma': 0.001}
Test set score with best parameters: 0.92
```

5.2.3 交差検証を用いたグリッドサーチ

- 5.2.3.1　交差検証の結果と解析
- 5.2.3.2　グリッドサーチでないサーチ空間
- 5.2.3.3　異なる交差検証手法を用いたグリッドサーチ
- 5.2.3.4　ネストした交差検証
- 5.2.3.5　交差検証とグリッドサーチの並列化

5.2.3 交差検証を用いたグリッドサーチ

- データの分割され方により、性能に大きな差が生じる。
  （例：テストセットと検証セットでの得られたパラメータの値）



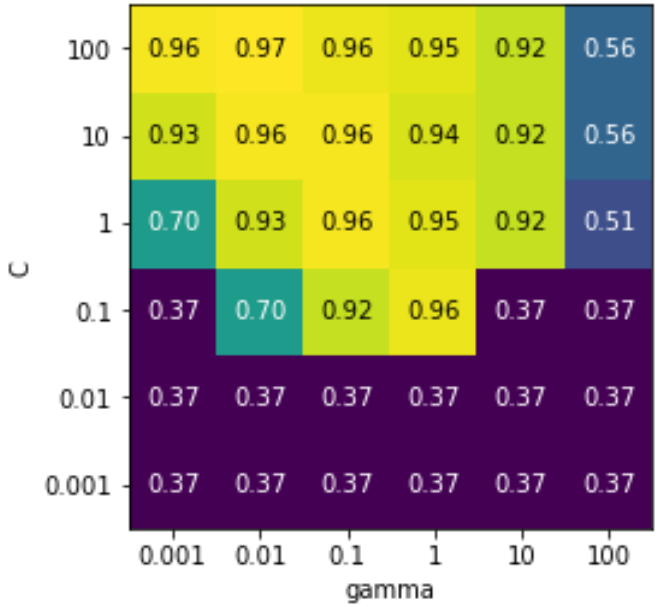
- それぞれのパラメータの組合せに対して交差検証を行う。
  （scikit-learnではGridSearchCVクラスを提供）
- 5分割交差検証の場合、パラメータの組合せが36通りのとき
  36×5通りの計算を行うことになる。

非常に時間が掛かる

5.2.3.1 交差検証の結果と解析

- 交差検証結果を可視化することにより、モデルの汎化性能がサーチパラメータに依存する様子を理解する手助けになる。

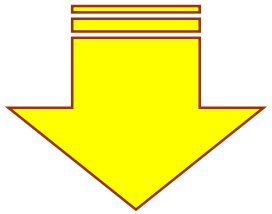
パラメータの設定によって
大きく変動することが分かる



5.2.3.2　グリッドサーチでないサーチ空間

- GridSearchCVは、すべてのパラメータのすべての組合せに対して試行を行うが適さない場合がある。

  例として、SVCのkernelが'linear'であればCしか使われず、gammaを変化させて試行する必要がない。



- 「条件付き」パラメータを扱うために、GridSearchCVは、param_gridとして辞書のリストを受け付けるようになっている。

5.2.3.2 グリッドサーチでないサーチ空間

In:
```
param_grid = [{'kernel': ['rbf'],
               'C': [0.001, 0.01, 0.1, 1, 10, 100],
               'gamma': [0.001, 0.01, 0.1, 1, 10, 100]},
              {'kernel': ['linear'],
               'C': [0.001, 0.01, 0.1, 1, 10, 100]}]
```

- 最初のグリッドは、パラメータkernelは常に'rbf'となり、Cとgammaを変化させる。

  次のグリッドは、パラメータkernelは常に'linear'となり、Cだけを変化させる。

5.2.3.3　異なる交差検証手法を用いたグリッドサーチ

- GridSearchCVは、デフォルトで

  クラス分類　：　層化k分割交差検証

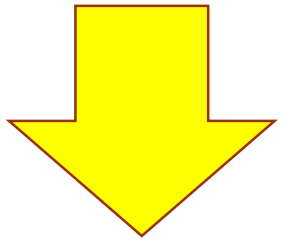
  回帰　：　k分割交差検証

  になっている。

  しかし、交差検証の分割器をcvパラメータとして渡すことが可能。

5.2.3.4　ネストした交差検証

- GridSearchCVを用いる例では、訓練セットとテストセットを一度だけ分割しているため、その一度の分割に依存してしまい、不安定になる。



- もとのデータを一度だけ分けるのではなく、交差検証で何度も分割する手法をネストした交差検証。

5.2.3.5　交差検証とグリッドサーチの並列化

- Scikit-learnではネストした並列実行はサポートされていない。
- グリッドサーチと交差検証をクラスタの複数のマシンを用いて並列化することも可能だがscikit-learnでは実装されていない。