Scikit-learn ゼミ

1.9.1 ～ 1.9.2

Ensemble methods

bagging

random forest

新納 浩幸

Ensemble method

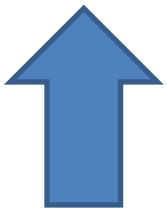
複数の分類器を組み合わせて利用して、
単一の分類器よりも精度を高める

（１）Average Method
　　複数の分類器の平均を取る

（２）Boosting Method
　　単一の分類器を系列的に変化させ、
　　それらを統合する

Bagging

訓練データのサブセットをランダムに取り出し、
それぞれから分類器を学習し、それらを統合して
利用する



簡単だけど強力

様々な Bagging

Pasting

訓練データのサブセットをランダムに取り出す

Bagging

訓練データを入れ替えることで、ランダムなサブセットを構築

Random Subspace

素性をランダムに選んでサブセットを構築

Random Patch

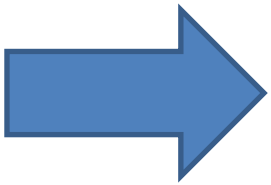
訓練データのサブセットのランダムと、素性のランダムを組み合わせる

適用例

```
>>> from sklearn.ensemble import BaggingClassifier
>>> from sklearn.neighbors import KNeighborsClassifier
>>> bagging = BaggingClassifier(KNeighborsClassifier(),max_samples=0.5,max_features=0
.5)
>>>
```

実装

Scikit-learn では決定木ベースの2つの
average method の手法が実装されている

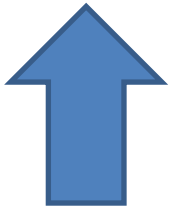


RandomForest

Extra-Trees

Random Forest

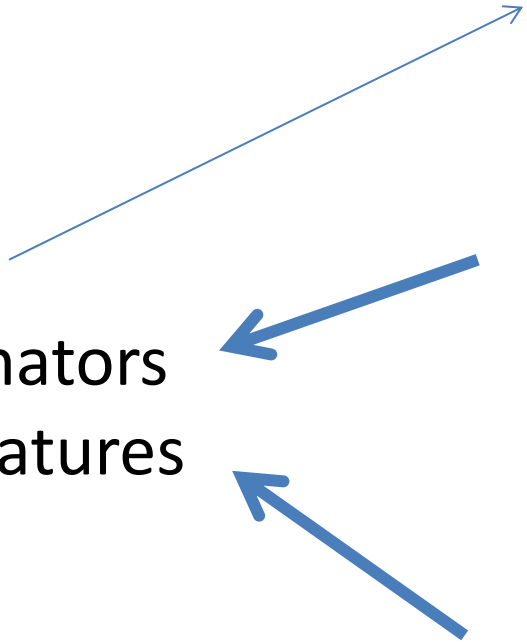
ランダムサンプリングされたトレーニングデータと
ランダムに選択された素性を用いて、複数の決定木を
作成して統合



かなり強力、、、SVM と同等

重要なパラメータ

RandomForestClassifier(・・・)



森の中の木の数、
大きいほどいいけど、
計算時間がかかる

n_estimators
max_features

ランダムな素性のサイズ、
小さいほど分散の減少が大、

分類問題では

max_features = sqrt(n_features)

その他

並列処理も可能

素性の重要度の評価もできる

利用例

```
>>> from sklearn.datasets import load_iris
>>> from sklearn.ensemble import RandomForestClassifier
>>> iris = load_iris()
>>> a = iris.data
>>> b = iris.target
>>> a1 = a[range(0,10)+range(50,60)+range(100,110)]
>>> b1 = b[range(0,10)+range(50,60)+range(100,110)]
>>> clf1 = RandomForestClassifier(n_estimators=10)
>>> clf1.fit(a1,b1)
RandomForestClassifier(bootstrap=True, compute_importances=None,
            criterion='gini', max_depth=None, max_features='auto',
            max_leaf_nodes=None, min_density=None, min_samples_leaf=1,
            min_samples_split=2, n_estimators=10, n_jobs=1,
            oob_score=False, random_state=None, verbose=0)
>>> ans1 = clf1.predict(a)
>>> ans1
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 1, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```