Machine Learning

Indah Agustien Siradjuddin

Ensemble Learning

Semester Gasal 2019-2020

Agregasi sejumlah *predictor* terkadang memberikan akurasi prediksi yang lebih baik dibandingkan *individual predictor*, baik itu *classifier* ataupun *regressor*
Grup sejumlah *predictor* : *ensemble*
Algoritma pembelajaran : *ensemble learning*
Setiap predictor dilatih, dan prediksi akhir dihitung dengan menggunakan suatu metode agregasi.

Tipe *Ensemble Learning*:

1. voting
2. bagging/pasting
3. random forest
4. boosting
5. stacking
6. etc

Voting

Sumber :

Prediksi akhir : 'hard voting' dari berbagai jenis *classifier* yang berbeda(logistic, tree, nn, etc), yaitu prediksi mayoritas

Sumber :

Voting dengan Scikit

In [1]:

```
from sklearn import datasets
from sklearn. tree import DecisionTreeClassifier
import numpy as np
from sklearn.linear_model.logistic import LogisticRegression
from sklearn.ensemble import VotingClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
```

In [2]:

```
iris=datasets.load_iris()
iris=datasets.load_iris()
featIris=iris.data[:,:3]
targetIris=iris.target
X_train, X_test, y_train, y_test = train_test_split(featIris, targetIris, test_size=0.7
, random_state=42)
```

In [4]:

```
treeClf1=DecisionTreeClassifier(max_depth=1)
logClf = LogisticRegression()
treeClf2=DecisionTreeClassifier(max_depth=1,criterion='entropy')
votingClf = VotingClassifier(estimators=[('lr', logClf), ('tr1', treeClf1),('tr2',treeC
lf2)],voting='hard')
for clf in (treeClf1,treeClf2,logClf,votingClf):
    clf.fit(X_train, y_train)
    y_pred = clf.predict(X_test)
    print(y_pred)
    print(clf.__class__.__name__, accuracy_score(y_test, y_pred))
```

```
[1 1 2 1 2 1 1 2 1 1 2 1 1 1 1 1 2 1 1 2 1 2 1 2 2 2 2 2 1 1 1 1 1 1 2 1
 1 1 1 2 1 1 1 1 1 2 2 1 2 1 2 1 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 1 2 1 2 2
 1 1 2 1 1 1 2 1 1 1 2 1 2 1 1 2 1 2 2 2 2 1 1 1 2 2 1 1 1 2 2]
DecisionTreeClassifier 0.5714285714285714
[2 0 2 2 2 0 2 2 2 2 2 0 0 0 0 2 2 2 2 2 0 2 0 2 2 2 2 2 0 0 0 0 2 0 0 2 2
 0 0 0 2 2 2 0 0 2 2 2 2 2 2 2 2 0 2 2 0 0 0 2 2 0 0 0 2 0 2 2 0 2 2 0 2 2
 2 2 2 2 0 2 2 0 0 2 2 0 2 0 0 2 2 2 2 2 2 2 0 0 2 2 0 0 0 2 2]
DecisionTreeClassifier 0.6857142857142857
[2 0 2 2 1 0 1 1 1 1 2 0 0 0 0 2 2 1 2 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
 0 0 0 2 2 1 0 0 1 2 2 1 2 1 2 2 0 2 1 0 0 0 1 2 0 0 0 2 0 1 2 0 1 2 0 2 2
 1 2 2 1 0 1 2 0 0 1 2 0 2 0 0 2 1 2 2 2 2 1 0 0 2 2 0 0 0 2 2]
LogisticRegression 0.8761904761904762
[2 0 2 2 2 0 1 2 1 1 2 0 0 0 0 2 2 1 2 2 0 2 0 2 2 2 2 2 0 0 0 0 1 0 0 2 1
 0 0 0 2 2 1 0 0 1 2 2 1 2 1 2 2 0 2 1 0 0 0 1 2 0 0 0 2 0 1 2 0 1 2 0 2 2
 1 2 2 1 0 1 2 0 0 1 2 0 2 0 0 2 1 2 2 2 2 1 0 0 2 2 0 0 0 2 2]
VotingClassifier 0.8761904761904762

C:\Users\Indah Agustin\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\Indah Agustin\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:469: FutureWarning: Default multi_class will be changed to 'aut
o' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
C:\Users\Indah Agustin\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:432: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
C:\Users\Indah Agustin\Anaconda3\lib\site-packages\sklearn\linear_model\lo
gistic.py:469: FutureWarning: Default multi_class will be changed to 'aut
o' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

Bagging / Pasting

- Voting : terdiri dari berbagai jenis *classifier*
- Bagging/Pasting : Sekelompok *predictor* dari algoritma Machine Learning yang sama, akan tetapi dilakukan proses *training* pada urutan dataset yang berbeda.
- Bagging (Bootstrap aggregating)/Pasting : data dapat di*sampling* pada predictor yang berbeda
- Bagging : data dapat di*sampling* beberapa kali pada predictor yang sama
- Prediksi akhir : voting (untuk *classifier*) dan rata-rata (untuk *regressor*)

Sumber :

Bagging dengan Scikit

In [5]:

```
from sklearn.ensemble import BaggingClassifier
bag_clf = BaggingClassifier(DecisionTreeClassifier(), n_estimators=500,max_samples=40,
bootstrap=True, n_jobs=-1)
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
```

```
0.9619047619047619
```

Out of Bag

OOB : Sekelompok data yang tidak pernah di*sampling* untuk proses pelatihan pada Bagging/pasting

In [6]:

```
from sklearn.ensemble import BaggingClassifier
bag_clf = BaggingClassifier(LogisticRegression(), n_estimators=500,max_samples=40, boot
strap=True, n_jobs=-1,oob_score=True)
bag_clf.fit(X_train, y_train)
y_pred = bag_clf.predict(X_test)
print(bag_clf.oob_score_)
print(accuracy_score(y_test, y_pred))
```

```
0.8888888888888888
0.8857142857142857
```

Random Forest

Ensemble dari sejumlah classifier *decision tree* dengan menggunakan metode *bagging*/*pasting*

In [7]:

```
from sklearn.ensemble import RandomForestClassifier
rnd_clf = RandomForestClassifier(n_estimators=500, max_leaf_nodes=16, n_jobs=-1)
rnd_clf.fit(X_train, y_train)
y_pred_rf = rnd_clf.predict(X_test)
print(accuracy_score(y_test, y_pred_rf))
```

```
0.9523809523809523
```

Adaboost

- Mengkombinasikan *weak learners* menjadi *strong learner*
- Pada voting/bagging/pasting, setiap *predictor* dilakukan proses training secara terpisah (independent)
- Adaboost (adaptive boosting), setiap *predictor* di*train* secara berurutan
- Kesalahan predictor sebelumnya akan diperbaiki oleh *current predictor* dengan cara fokus pada data pelatihan yang tidak dapat diklasifikasikan secara benar oleh predictor sebelumnya
- Bobot data tersebut dinaikkan, sehingga memiliki probabalitas *sampling* lebih tinggi

Sumber :

Inisialisasi

$w^{(i)} = \frac{1}{m}$; m jumlah data pada dataset

weighted error dari Predictor ke-j

$$r_j = \frac{\sum_{i=1}^{m} w^i; \hat{y}_j^i \neq y^i}{\sum_{i=1}^{m} w^i}$$

Bobot Predictor

$$\alpha_j = \eta ln \frac{1 - r_j}{r_j}$$

Semakin besar *weighted error* akan membuat bobot *predictor* semakin rendah, dan sebaliknya
Bobot ini digunakan untuk *update* bobot tiap data pada dataset, dan prediksi akhir (*final aggregation*)

update weight

$$w^i = w^i \times e^{\alpha_j \times err}$$

if $\hat{y}_j^i == y^i$ then err=0 else err=1

Prediksi Akhir

$$\hat{y}(x) = \arg\max_k \sum_{j=1}^{N} \alpha_j; \hat{y}_j(x) == k$$

In [8]:

```
from sklearn.ensemble import AdaBoostClassifier
ada_clf = AdaBoostClassifier(DecisionTreeClassifier(max_depth=1), n_estimators=200,algo
rithm="SAMME.R", learning_rate=0.5)
ada_clf.fit(X_train, y_train)
y_pred=ada_clf.predict(X_test)
print(accuracy_score(y_test, y_pred))
```

```
0.8666666666666667
```

Referensi