集成算法(Ensemble Method)

作者：刘伟杰 日期：2015-12-01
参考：
[1]《机器学习实战》 Peter Harrington
[2] scikit-learn官方手册

1. 集成算法：

将多个分类器集成起来而形成的新的分类算法。这类算法又称元算法(meta-algorithm)。最常见的集成思想有两种bagging和boosting。

2. 集成思想：

1. boosting：

   基于错误提升分类器性能，通过集中关注被已有分类器分类错误的样本，构建新分类器并集成。

2. bagging：

   基于数据随机重抽样的分类器构建方法。

3. 算法示例：

1. 随机森林（Random Forest: bagging ＋ 决策树）：

   将训练集按照横（随机抽样本）、列（随机抽特征）进行有放回的随机抽取，获得n个新的训练集，训练出n个决策树，通过这n个树投票决定分类结果。主要的parameters 有 n_estimators 和 max_features。

   ```
   >>> from sklearn.ensemble import RandomForestClassifier
   >>> X = [[0, 0], [1, 1]]
   >>> Y = [0, 1]
   >>> clf = RandomForestClassifier(n_estimators=10)
   >>> clf = clf.fit(X, Y)

   >>> # 扩展： Extremely Randomized Trees 比随机森林还牛逼的分类算法，见(http://scikit-
   ```

2. Adaboost (adaptive boosting: boosting + 单层决策树):

训练数据中的每个样本，并赋予其一个权重，这些权重构成了向量D。一开始，这些权重都初始化成相等值。首先在训练数据上训练出一个弱分类器并计算该分类器的错误率，然后在统一数据集上再训练分类器。在第二次训练中，会调高那些前一个分类器分类错误的样本的权重。如此反复，训练出许多分类器来进行加权投票，每个分类器的权重是基于该分类器的错误率计算出来的。

我的实现：

```
https://github.com/autoliuweijie/MachineLearning/tree/master/adaboost
```

scikit-learn:

```
>>> from sklearn.cross_validation import cross_val_score
>>> from sklearn.datasets import load_iris
>>> from sklearn.ensemble import AdaBoostClassifier

>>> iris = load_iris()
>>> clf = AdaBoostClassifier(n_estimators=100)
>>> scores = cross_val_score(clf, iris.data, iris.target)
>>> scores.mean()
0.9...
```

3. GBDT (Gradient Boosting Decision Tree: boosting + 决策树）：

   GBDT与Adaboost类似，反复训练出多个决策树，每次更新训练集的权重是按照损失函数负梯度的方向。过程比较复杂，这里不赘述。对参数做一下说明：n_estimators是弱分类器个数；max_depth或max_leaf_nodes可以用来控制每棵树的规模；learning_rate是hyper-parameter，取值范围为(0, 1.0],用来控制过拟合与欠拟合。

   ```
   >>> from sklearn.datasets import make_hastie_10_2
   >>> from sklearn.ensemble import GradientBoostingClassifier

   >>> X, y = make_hastie_10_2(random_state=0)
   >>> X_train, X_test = X[:2000], X[2000:]
   >>> y_train, y_test = y[:2000], y[2000:]

   >>> clf = GradientBoostingClassifier(n_estimators=100, learning_rate=1.0,
   ...     max_depth=1, random_state=0).fit(X_train, y_train)
   >>> clf.score(X_test, y_test)
   0.913...
   ```

   决策树还有个兄弟回归树，GBDT也有个兄弟GBRT用来做回归：

```
>>> import numpy as np
>>> from sklearn.metrics import mean_squared_error
>>> from sklearn.datasets import make_friedman1
>>> from sklearn.ensemble import GradientBoostingRegressor

>>> X, y = make_friedman1(n_samples=1200, random_state=0, noise=1.0)
>>> X_train, X_test = X[:200], X[200:]
>>> y_train, y_test = y[:200], y[200:]
>>> est = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,
...     max_depth=1, random_state=0, loss='ls').fit(X_train, y_train)
>>> mean_squared_error(y_test, est.predict(X_test))
5.00...
```

4. 自己设计:

   根据bagging或者boosting思想，自己选择弱分类器来集成:

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