Scikit Learn: Machine Learning in Python

Gianluca Corrado

gianluca.corrado@unitn.it

Machine Learning

Python Scientific Lecture Notes

- Scikit Learn is based on Python
- especially on NumPy, SciPy, and matplotlib
- which are packages for scientific computing in Python

Basics on Python and on scientific computing

- `http://scipy-lectures.github.io/`

Downloading and Installing

Requires:

- Python ($\geq$2.6 or $\geq$3.3)
- NumPy ($\geq$ 1.6.1)
- SciPy ($\geq$ 0.9)

`http://scikit-learn.org/stable/install.html`

Documentation and Reference

Documentation

`http://scikit-learn.org/stable/documentation.html`

Reference Manual with class descriptions

`http://scikit-learn.org/stable/modules/classes.html`

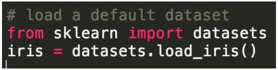
Outline

Today we are going to learn how to:

- Load and generate datasets
- Split a dataset for cross-validation
- Use some learning algorithms
  - Naive Bayes
  - SVM
  - Random forest
- Evalute the performance of the algorithms
  - Accuracy
  - F1-score
  - AUC ROC

Datasets

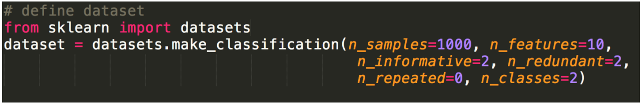
- The `sklearn.datasets` module includes utilities to load datasets
- Load and fetch popular reference datasets (e.g. Iris)



```python
# load a default dataset
from sklearn import datasets
iris = datasets.load_iris()
```

`http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html`

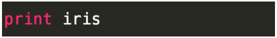
- Artificial data generators (e.g. binary classification)



```python
# define dataset
from sklearn import datasets
dataset = datasets.make_classification(n_samples=1000, n_features=10,
                                       n_informative=2, n_redundant=2,
                                       n_repeated=0, n_classes=2)
```

`http://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html`

Now inspect the data structures



```python
print iris
```

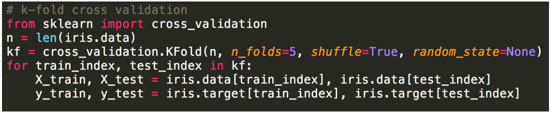
Cross-validation

k-fold cross-validation

- Split the dataset D in k equal sized disjoint subsets D_i
- For $i \in [1, k]$
  - train the predictor on $T_i = D \setminus D_i$
  - compute the score of the predictor on the test set D_i
- Return the average score accross the folds

Cross-validation

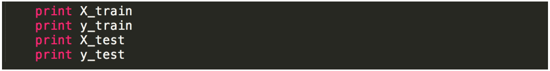
- The sklearn.cross_validation module includes utilities for cross-validation and performance evaluation
- e.g. k-fold cross validation



```
# k-fold cross validation
from sklearn import cross_validation
n = len(iris.data)
kf = cross_validation.KFold(n, n_folds=5, shuffle=True, random_state=None)
for train_index, test_index in kf:
    X_train, X_test = iris.data[train_index], iris.data[test_index]
    y_train, y_test = iris.target[train_index], iris.target[test_index]
```

http://scikit-learn.org/stable/modules/generated/sklearn.cross_validation.KFold.html

Now inspect the data structures



```
print X_train
print y_train
print X_test
print y_test
```

Naive Bayes

Hint

- Attribute values are assumed independent of each other

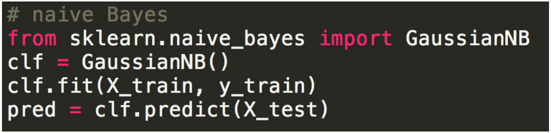
$$P(a_1, \dots, a_m | y_i) = \prod_{j=1}^{m} P(a_j | y_i)$$

- Definition

$$y^* = argmax_{y_i} \prod_{j=1}^{m} P(a_j | y_i) P(y_i)$$

Naive Bayes

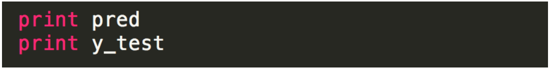
- The `sklearn.naive_bayes` module implements naive Bayes algorithms
- e.g. Gaussian naive Bayes



```python
# naive Bayes
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train, y_train)
pred = clf.predict(X_test)
```

http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html

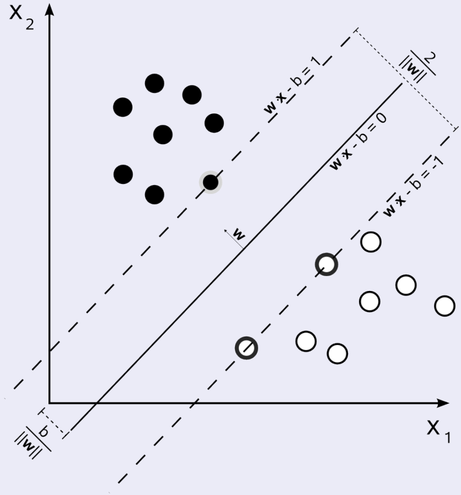
Now inspect the data structures



```python
print pred
print y_test
```

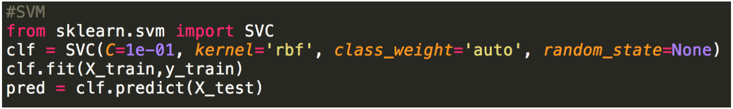
SVM

Hint



Hint

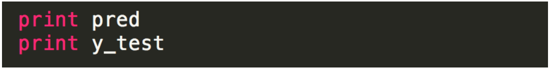
- The `sklearn.svm` module includes Support Vector Machine algorithms
- e.g. Support-C Vector Classification



```
#SVM
from sklearn.svm import SVC
clf = SVC(C=1e-01, kernel='rbf', class_weight='auto', random_state=None)
clf.fit(X_train,y_train)
pred = clf.predict(X_test)
```

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

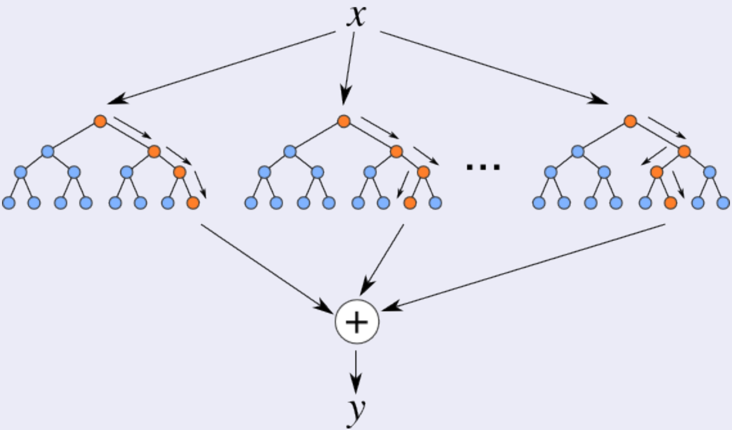
Now inspect the data structures



```
print pred
print y_test
```

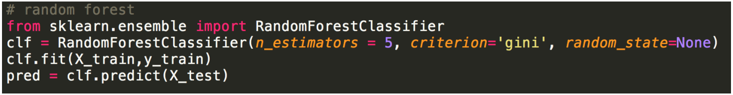
Random Forest

Hint



Random Forest

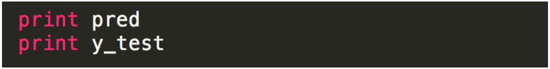
- The `sklearn.ensemble` module includes ensemble-based methods for classification and regression
- e.g. Random Forest Classifier



```
# random forest
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(n_estimators = 5, criterion='gini', random_state=None)
clf.fit(X_train,y_train)
pred = clf.predict(X_test)
```

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Now inspect the data structures



```
print pred
print y_test
```

Performance evaluation

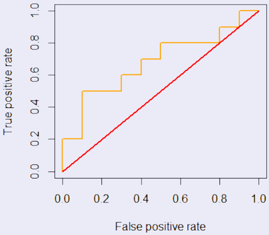
Recap

$$Acc = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Pre = \frac{TP}{TP + FP} \qquad Rec = \frac{TP}{TP + FN}$$

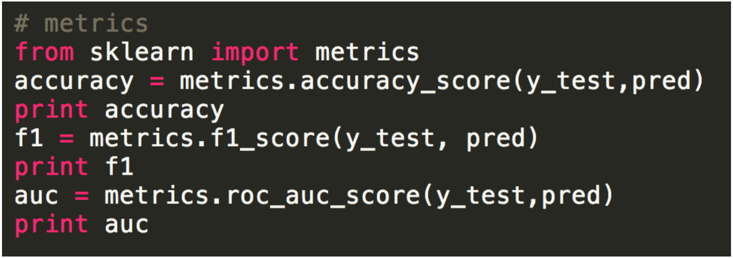
$$F1 = \frac{2(Pre * Rec)}{Pre + Rec}$$

AUC ROC



Performance evaluation

- The `sklearn.metrics` module includes score functions, performance metrics and pairwise metrics and distance computations.
- e.g. accuracy, F1-score, AUC ROC



```
# metrics
from sklearn import metrics
accuracy = metrics.accuracy_score(y_test,pred)
print accuracy
f1 = metrics.f1_score(y_test, pred)
print f1
auc = metrics.roc_auc_score(y_test,pred)
print auc
```

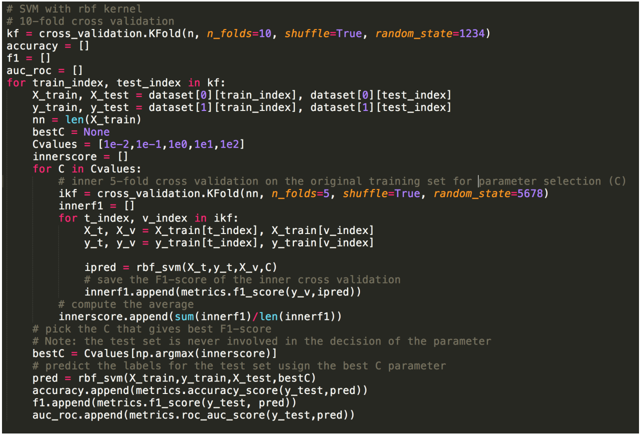
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.auc.html

Choosing parameters

- Some algorithms have parameters
- e.g. parameter C for SVM, number of trees for Random Forest
- Performance can significantly vary according to the chosen parameters
- It is important to choose wisely
- **train, VALIDATION, test**

Choosing parameters e.g. SVM

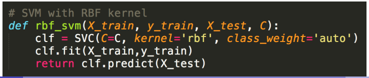
np.argmax requires to add import numpy as np



```
# SVM with rbf kernel
# 10-fold cross validation
kf = cross_validation.KFold(n, n_folds=10, shuffle=True, random_state=1234)
accuracy = []
f1 = []
auc_roc = []
for train_index, test_index in kf:
    X_train, X_test = dataset[0][train_index], dataset[0][test_index]
    y_train, y_test = dataset[1][train_index], dataset[1][test_index]
    nn = len(X_train)
    bestC = None
    Cvalues = [1e-2,1e-1,1e0,1e1,1e2]
    innerscore = []
    for C in Cvalues:
        # inner 5-fold cross validation on the original training set for parameter selection (C)
        ikf = cross_validation.KFold(nn, n_folds=5, shuffle=True, random_state=5678)
        innerf1 = []
        for t_index, v_index in ikf:
            X_t, X_v = X_train[t_index], X_train[v_index]
            y_t, y_v = y_train[t_index], y_train[v_index]

            ipred = rbf_svm(X_t,y_t,X_v,C)
            # save the F1-score of the inner cross validation
            innerf1.append(metrics.f1_score(y_v,ipred))
        # compute the average
        innerscore.append(sum(innerf1)/len(innerf1))
    # pick the C that gives best F1-score
    # Note: the test set is never involved in the decision of the parameter
    bestC = Cvalues[np.argmax(innerscore)]
    # predict the labels for the test set usign the best C parameter
    pred = rbf_svm(X_train,y_train,X_test,bestC)
    accuracy.append(metrics.accuracy_score(y_test,pred))
    f1.append(metrics.f1_score(y_test, pred))
    auc_roc.append(metrics.roc_auc_score(y_test,pred))
```

where



```
# SVM with RBF kernel
def rbf_svm(X_train, y_train, X_test, C):
    clf = SVC(C=C, kernel='rbf', class_weight='auto')
    clf.fit(X_train,y_train)
    return clf.predict(X_test)
```

Summary

sklearn allows to:

- load and generate datasets
- split them to perform cross-validation
- easily apply learning algorithms
- evaluate the performace of such algorithms

Assignment

The second ML assignment is to compare the performance of three different classification algorithms, namely Naive Bayes, SVM, and Random Forest.
For this assignment you need to generate a random binary classification problem, and train (using 10-fold cross validation) the three different algorithms. For some algorithms inner cross validation (5-fold) for choosing the parameters is needed. Then, show the classification performace (per-fold and averaged) in the report, briefly discussing the results.

Note

The report has to contain also a short description of the methodology used to obtain the results.

Assignment

Steps

1. Create a classification dataset (n_samples $\geq$ 1000, n_features $\geq$ 10)
2. Split the dataset using 10-fold cross validation
3. Train the algorithms
    - GaussianNB
    - SVC (possible C values [1e-02, 1e-01, 1e00, 1e01, 1e02], and RBF kernel)
    - RandomForestClassifier (possible n_estimators values [10, 100, 1000], and Gini purity)
4. Evaluate the cross-validated performance
    - accuracy
    - F1-score
    - AUC ROC
5. Write a short report summarizing the methodology and the results

Assignment

- After completing the assignment submit it via email
- Send an email to gianluca.corrado@unitn.it (cc: passerini@disi.unitn.it)
- Subject: sklearnSubmit
- Attachment: id_name_surname.zip containing:
    - the Python code
    - the report (PDF format)

NOTE

- No group work
- This assignment is mandatory in order to enroll to the oral exam