Estimação, Deteção e Análise II

05 – Avaliação e seleção de modelos

Validação cruzada

Bootstrap

Curvas ROC

Seleção de características

Regularização

Validação cruzada[1]

Carregar o dataset iris e aplicar SVM:

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn import datasets
from sklearn import svm

X, y = datasets.load_iris(return_X_y=True)
print(X.shape, y.shape) # (150, 4) (150,)
```

Definir conjunto de treino como 60% do dataset e de teste como 40%:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=0)

print(X_train.shape, y_train.shape) # (90, 4) (90,)
print(X_test.shape, y_test.shape) # (60, 4) (60,)

clf = svm.SVC(kernel='linear', C=1).fit(X_train, y_train)
print(clf.score(X_test, y_test)) # 0.9666666666666667
```

Aplicar 5-fold cross-validation:

```
from sklearn.model_selection import cross_val_score
clf = svm.SVC(kernel='linear', C=1, random_state=42)
scores = cross_val_score(clf, X, y, cv=5)
print(scores) # [0.96666667 1. 0.96666667 0.96666667 1.]
print("%0.2f accuracy - stdev of %0.2f" % (scores.mean(), scores.std()))
# 0.98 accuracy with a standard deviation of 0.02
```

Alterar a métrica usada para cálculo do score:

```
from sklearn import metrics
scores = cross_val_score(clf, X, y, cv=5, scoring='f1_macro')
print(scores) # [0.96658312 1. 0.96658312 0.96658312 1.]
```

Alterar o método de cross-validation:

```
from sklearn.model_selection import ShuffleSplit
n_samples = X.shape[0]
cv = ShuffleSplit(n_splits=5, test_size=0.3, random_state=0)
print(cross_val_score(clf, X, y, cv=cv)) # [0.97777778 0.97777778 1. 0.95555556 1.]
```

Usar várias métricas para avaliar cross-validation:

```
from sklearn.model_selection import cross_validate
from sklearn.metrics import recall_score
scoring = ['precision_macro', 'recall_macro']
clf = svm.SVC(kernel='linear', C=1, random_state=0)
scores = cross_validate(clf, X, y, scoring=scoring)

print(scores)
```

[1] Mais exemplos: https://scikit-learn.org/stable/modules/cross_validation.html

```
# {
#     'fit_time': array([0.01562285, 0., 0., 0., 0.]),
#     'score_time': array([0., 0., 0., 0., 0.]),
#     'test_precision_macro': array([0.96969697, 1., 0.96969697, 0.96969697, 1.]),
#     'test_recall_macro': array([0.96666667, 1., 0.96666667, 0.96666667, 1.])
# }
```

Bootstrap

Analisar e executar o seguinte exemplo (apenas para perceber a amostra escolhida):

```
# scikit-learn bootstrap
from sklearn.utils import resample
# data sample
data = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6]
# prepare bootstrap sample
boot = resample(data, replace=True, n_samples=4, random_state=1)
print('Bootstrap Sample: %s' % boot)
# out of bag observations
oob = [x for x in data if x not in boot]
print('OOB Sample: %s' % oob)
```

Curvas ROC[2]

Cálculo de ROC:

```
import numpy as np
import matplotlib.pyplot as plt
from itertools import cycle

from sklearn import svm, datasets
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn.multiclass import OneVsRestClassifier
from sklearn.metrics import roc_auc_score

# Import some data to play with
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Binarize the output
y = label_binarize(y, classes=[0, 1, 2])
n_classes = y.shape[1]

# Add noisy features to make the problem harder
random_state = np.random.RandomState(0)
n_samples, n_features = X.shape
X = np.c_[X, random_state.randn(n_samples, 200 * n_features)]

# shuffle and split training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=0)

# Learn to predict each class against the other
classifier = OneVsRestClassifier(
    svm.SVC(kernel="linear", probability=True, random_state=random_state)
)
y_score = classifier.fit(X_train, y_train).decision_function(X_test)

# Compute ROC curve and ROC area for each class
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
```

[2] Exemplos para multiclass: https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

```
    roc_auc[i] = auc(fpr[i], tpr[i])

# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
```

Plot da ROC curve:

```
plt.figure()
lw = 2
plt.plot(
    fpr[2],
    tpr[2],
    color="darkorange",
    lw=lw,
    label="ROC curve (area = %0.2f)" % roc_auc[2],
)
plt.plot([0, 1], [0, 1], color="navy", lw=lw, linestyle="--")
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title("Receiver operating characteristic example")
plt.legend(loc="lower right")
plt.show()
```

Seleção de características[3]

Remover features com baixa variância (mesmo valor em mais que 80% dos dados):

```
from sklearn.feature_selection import VarianceThreshold
X = [[0, 0, 1], [0, 1, 0], [1, 0, 0], [0, 1, 1], [0, 1, 0], [0, 1, 1]]
sel = VarianceThreshold(threshold=(.8 * (1 - .8)))
print(sel.fit_transform(X))
# [[0 1]
#  [1 0]
#  [0 0]
#  [1 1]
#  [1 0]
#  [1 1]]
```

Univariate feature selection (com teste χ^2):

```
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
X, y = load_iris(return_X_y=True)
print(X[:5])
X_new = SelectKBest(chi2, k=2).fit_transform(X, y)
print(X_new[:5])
```

Regularização

Ridge Regression

```
from sklearn import linear_model
reg = linear_model.Ridge(alpha=.5)
print(reg.fit([[0, 0], [0, 0], [1, 1]], [0, .1, 1])) # Ridge(alpha=0.5)
print(reg.coef_) # [0.34545455 0.34545455]
print(reg.intercept_) # 0.13636363636363638
```

Lasso Regression:

```
from sklearn import linear_model
reg = linear_model.Lasso(alpha=0.1)
```

[3] Mais exemplos: https://scikit-learn.org/stable/modules/feature_selection.html

```
print(reg.fit([[0, 0], [1, 1]], [0, 1])) # Lasso(alpha=0.1)
print(reg.predict([[1, 1]])) # array([0.8])
```