

[博客专区](#) > [ylliu的博客](#) > [博客详情](#)

Scikit Learn: 在python中机器学习

ylliu 发表于 4年前 阅读 47392 收藏 53 点赞 11 评论 7

[收藏](#)[程序员们的淘宝店！寻找开源技术服务伙伴！>>>](#) HOT

Scikit Learn: 在python中机器学习

Warning

警告：有些没能理解的句子，我以自己的理解意译。

翻译自：Scikit Learn:Machine Learning in Python

作者: Fabian Pedregosa, Gael Varoquaux

先决条件

- Numpy, Scipy
- IPython
- matplotlib
- scikit-learn

目录

- 载入示例数据
 - 一个改变数据集大小的示例：数码数据集(digits datasets)
 - 学习和预测
- 分类
 - K最近邻(KNN)分类器
 - 训练集和测试集
 - 分类支持向量机(SVMs)
 - 线性支持向量机
 - 使用核
- 聚类：将观测值聚合
 - k均值聚类
 - 应用到图像压缩
- 用主成分分析降维
- 将一切放在一起：人脸识别
- 线性模型：从回归到稀疏
 - 稀疏模型
 - 同一问题的不同算法
- 模型选择：选择估计器和它们的参数
 - 格点搜索和交叉验证估计器
 - 格点搜索
 - 交叉验证估计器

- Footnotes

警告：在0.9版中(2011年9月发行)，scikit-learn的导入路径从scikits.learn更改为sklearn

载入示例数据

首先我们载入一些用来玩耍的数据。我们将使用的数据是非常简单的著名的花朵数据——安德森鸢尾花卉数据集。

我们有一百五十个鸢尾花的一些尺寸的观测值：萼片长度、宽度，花瓣长度和宽度。还有它们的亚属：山鸢尾（Iris setosa）、变色鸢尾（Iris versicolor）和维吉尼亚鸢尾（Iris virginica）

向python对象载入数据：

```
In [1]: from sklearn import datasets
In [2]: iris = datasets.load_iris()
```

数据存储在.data项中，是一个(n_samples, n_features)数组。

```
In [3]: iris.data.shape
Out[3]: (150, 4)
```

每个观察对象的种类存储在数据集的.target属性中。这是一个长度为n_samples的整数一维数组：

```
In [5]: iris.target.shape
Out[5]: (150,)

In [6]: import numpy as np

In [7]: np.unique(iris.target)
Out[7]: array([0, 1, 2])
```

一个改变数据集大小的示例：数码数据集(digits datasets)

数码数据集¹包括1797个图像，每一个都是个代表手写数字的8x8像素图像

```
In [8]: digits = datasets.load_digits()

In [9]: digits.images.shape
Out[9]: (1797, 8, 8)

In [10]: import pylab as pl

In [11]: pl.imshow(digits.images[0], cmap=pl.cm.gray_r)
Out[11]: <matplotlib.image.AxesImage at 0x3285b90>

In [13]: pl.show()
```

为了在scikit中使用这个数据集，我们把每个8x8图像转换成长度为64的矢量。(译者注：或者直接用digits.data)

```
In [12]: data = digits.images.reshape((digits.images.shape[0], -1))
```

学习和预测

现在我们已经获得一些数据，我们想要从中学习和预测一个新的数据。在scikit-learn中，我们通过创建一个估计器(estimator)从已经存在的数据学习，并且调用它的fit(X,Y)方法。

```
In [14]: from sklearn import svm

In [15]: clf = svm.LinearSVC()

In [16]: clf.fit(iris.data, iris.target) # learn from the data
Out[16]:
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='l2', multi_class='ovr', penalty='l2',
          tol=0.0001, verbose=0)
```

一旦我们已经从数据学习，我们可以使用我们的模型来预测未观测数据最可能的结果。

```
In [17]: clf.predict([[ 5.0,  3.6,  1.3,  0.25]])
Out[17]: array([0], dtype=int32)
```

注意：我们可以通过它以下划线结束的属性存取模型的参数：

```
In [18]: clf.coef_
Out[18]:
array([[ 0.18424352,  0.45122644, -0.8079467 , -0.45071302],
       [ 0.05190619, -0.89423619,  0.40519245, -0.93781587],
       [-0.85087844, -0.98667529,  1.38088883,  1.86538111]])
```

分类

K最近邻(KNN)分类器

最简单的可能的分类器是最近邻：给定一个新的观测值，将n维空间中最靠近它的训练样本标签给它。其中n是每个样本中特性(features)数。

k最近邻²分类器内部使用基于球树(ball tree)³来代表它训练的样本。

KNN分类示例：

```
In [19]: # Create and fit a nearest-neighbor classifier

In [20]: from sklearn import neighbors

In [21]: knn = neighbors.KNeighborsClassifier()

In [22]: knn.fit(iris.data, iris.target)
Out[22]:
KNeighborsClassifier(algorithm='auto', leaf_size=30, n_neighbors=5, p=2,
                    warn_on_equidistant=True, weights='uniform')

In [23]: knn.predict([[0.1, 0.2, 0.3, 0.4]])
Out[23]: array([0])
```

训练集和测试集

当验证学习算法时，不要用一个用来拟合估计器的数据来验证估计器的预测非常重要。确实，通过kNN估计器，我们将总是获得关于训练集完美的预测。

```
In [24]: perm = np.random.permutation(iris.target.size)

In [25]: iris.data = iris.data[perm]

In [26]: iris.target = iris.target[perm]

In [27]: knn.fit(iris.data[:100], iris.target[:100])
Out[27]:
KNeighborsClassifier(algorithm='auto', leaf_size=30, n_neighbors=5, p=2,
                    warn_on_equidistant=True, weights='uniform')

In [28]: knn.score(iris.data[100:], iris.target[100:])
/usr/lib/python2.7/site-packages/sklearn/neighbors/classification.py:129: NeighborsWarning: kneighbors: neighb
    neigh_dist, neigh_ind = self.kneighbors(X)
Out[28]: 0.95999999999999996
```

Bonus的问题：为什么我们使用随机的排列？

分类支持向量机(SVMs)

线性支持向量机

SVMs⁴尝试构建一个两个类别的最大间隔超平面。它选择输入的子集，调用支持向量即离分离的超平面最近的样本点。

```
In [60]: from sklearn import svm

In [61]: svc = svm.SVC(kernel='linear')
```

```
In [62]: svc.fit(iris.data, iris.target)
Out[62]:
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, degree=3, gamma=0.0,
    kernel='linear', probability=False, shrinking=True, tol=0.001,
    verbose=False)
```

scikit-learn中有好几种支持向量机实现。最普遍使用的是svm.SVC, svm.NuSVC和svm.LinearSVC;“SVC”代表支持向量分类器(Support Vector Classifier)(也存在回归SVMs, 在scikit-learn中叫作“SVR”)。

练习

训练一个数字数据集的svm.SVC。省略最后10%并且检验观测值的预测表现。

使用核

类别不总是可以用超平面分离, 所以人们指望有些可能是多项式或指数实例的非线性决策函数:

- 线性核

```
svc = svm.SVC(kernel='linear')
```

- 多项式核

```
svc = svm.SVC(kernel='poly', ... degree=3) # degree: polynomial degree
```

- RBF核(径向基函数)⁵

```
svc = svm.SVC(kernel='rbf') # gamma: inverse of size of # radial kernel
```

练习

以上提到的哪些核对数字数据集有更好的预测性能?(译者: 前两个)

聚类: 将观测值聚合

给定鸢尾花数据集, 如果我们知道有三种鸢尾花, 但是无法得到它们的标签, 我们可以尝试非监督学习: 我们可以通过某些标准聚类观测值到几个组别里。

k均值聚类

最简答的聚类算法是k均值算法。这将一个数据分成k个集群, 以最小化观测值(n维空间中)到聚类中心的均值来分配每个观测点到集群;然后均值重新被计算。这个操作递归运行直到聚类收敛, 在max_iter回合内到最大值。⁶

(一个替代的k均值算法实现在scipy中的cluster包中。这个scikit-learn实现与之不同, 通过提供对象API和几个额外的特性, 包括智能初始化。)

```
In [82]: from sklearn import cluster, datasets

In [83]: iris = datasets.load_iris()

In [84]: k_means = cluster.KMeans(k=3)

In [85]: k_means.fit(iris.data)
Out[85]:
KMeans(copy_x=True, init='k-means++', k=3, max_iter=300, n_init=10, n_jobs=1,
    precompute_distances=True,
    random_state=<mtrand.RandomState object at 0x7f4d860642d0>, tol=0.0001,
    verbose=0)

In [86]: print k_means.labels_[:10]
[1 1 1 1 1 2 2 2 2 0 0 0 0]

In [87]: print iris.target[:10]
[0 0 0 0 0 1 1 1 1 2 2 2 2]
```

应用到图像压缩

译者注: Lena是经典的图像处理实例图像, 8位灰度色深, 尺寸512 x 512

聚类可以被看作是一种从信息中选择一小部分观测值。例如，这个可以被用来海报化一个图像(将连续变化的色调转换成更少几个色调)：

```
In [95]: from scipy import misc

In [96]: lena = misc.lena().astype(np.float32)

In [97]: X = lena.reshape((-1, 1)) # We need an (n_sample, n_feature) array

In [98]: k_means = cluster.KMeans(5)

In [99]: k_means.fit(X)
Out[99]:
KMeans(copy_x=True, init='k-means++', k=5, max_iter=300, n_init=10, n_jobs=1,
       precompute_distances=True,
       random_state=<mtrand.RandomState object at 0x7f4d860642d0>, tol=0.0001,
       verbose=0)

In [100]: values = k_means.cluster_centers_.squeeze()

In [101]: labels = k_means.labels_

In [102]: lena_compressed = np.choose(labels, values)

In [103]: lena_compressed.shape = lena.shape
```

译者注：想看效果？

```
In [31]: import matplotlib.pyplot as plt

In [32]: plt.gray()

In [33]: plt.imshow(lena_compressed)
Out[33]: <matplotlib.image.AxesImage at 0x4b2c510>

In [34]: plt.show()
```

原图类似。

![Image]

用主成分分析降维

以上根据观测值标记的点云在一个方向非常平坦，所以一个特性几乎可以用其它两个确切地计算。PCA发现哪个方向的数据不是平的并且它可以通过在一个子空间投影来降维。

警告：PCA将在模块decomposition或pca中，这取决于你scikit-learn的版本。

```
In [75]: from sklearn import decomposition

In [76]: pca = decomposition.PCA(n_components=2)

In [77]: pca.fit(iris.data)
Out[77]: PCA(copy=True, n_components=2, whiten=False)

In [78]: X = pca.transform(iris.data)
```

现在我们可以可视化(降维过的)鸢尾花数据集：

```
In [79]: import pylab as pl

In [80]: pl.scatter(X[:, 0], X[:, 1], c=iris.target)
Out[80]: <matplotlib.collections.PathCollection at 0x4104310>
```

PCA不仅在可视化高维数据集时非常有用。它可以用来作为帮助加速对高维数据不那么有效率的监督方法⁷的预处理步骤。

将一切放在一起：人脸识别

一个实例使用主成分分析来降维和支持向量机来分类进行人脸识别。

译者注：让程序自动下载(确保联网，文件较大，要等待很久)或者手动下载数据并放到./scikit_learn_data/lfw_home/下。

```

"""
Stripped-down version of the face recognition example by Olivier Grisel

http://scikit-learn.org/dev/auto_examples/applications/face_recognition.html

## original shape of images: 50, 37
"""
import numpy as np
import pylab as pl
from sklearn import cross_val, datasets, decomposition, svm

# ..
# .. load data ..
lfw_people = datasets.fetch_lfw_people(min_faces_per_person=70, resize=0.4)
perm = np.random.permutation(lfw_people.target.size)
lfw_people.data = lfw_people.data[perm]
lfw_people.target = lfw_people.target[perm]
faces = np.reshape(lfw_people.data, (lfw_people.target.shape[0], -1))
train, test = iter(cross_val.StratifiedKFold(lfw_people.target, k=4)).next()
X_train, X_test = faces[train], faces[test]
y_train, y_test = lfw_people.target[train], lfw_people.target[test]

# ..
# .. dimension reduction ..
pca = decomposition.RandomizedPCA(n_components=150, whiten=True)
pca.fit(X_train)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)

# ..
# .. classification ..
clf = svm.SVC(C=5., gamma=0.001)
clf.fit(X_train_pca, y_train)

# ..
# .. predict on new images ..
for i in range(10):
    print lfw_people.target_names[clf.predict(X_test_pca[i])[0]]
    _ = pl.imshow(X_test[i].reshape(50, 37), cmap=pl.cm.gray)
    _ = raw_input()

```

全部代码：face.py

线性模型：从回归到稀疏

糖尿病数据集

糖尿病数据集包含442个病人的测量而得的10项生理指标(年龄，性别，体重，血压)，和一年后疾病进展的指示：

```

In [104]: diabetes = datasets.load_diabetes()

In [105]: diabetes_X_train = diabetes.data[:-20]

In [106]: diabetes_X_test = diabetes.data[-20:]

In [107]: diabetes_y_train = diabetes.target[:-20]

In [108]: diabetes_y_test = diabetes.target[-20:]

```

这个手头的任务是用来从生理指标预测疾病。

稀疏模型

为了改善问题的条件(无信息变量，减少维度的不利影响，作为一个特性(feature)选择的预处理，等等)，我们只关注有信息的特性将没有信息的特性设置为0.这个罚则函数法⁸,叫作套索(Lasso)⁹，可以将一些系数设置为0.这些方法叫作稀疏方法(*sparse method*)，稀疏化可以被视作奥卡姆剃刀：相对于复杂模型更倾向于简单的。

```

In [109]: from sklearn import linear_model

In [110]: regr = linear_model.Lasso(alpha=.3)

In [111]: regr.fit(diabetes_X_train, diabetes_y_train)
Out[111]:
Lasso(alpha=0.3, copy_X=True, fit_intercept=True, max_iter=1000,
      normalize=False, positive=False, precompute='auto', tol=0.0001,
      warm_start=False)

In [112]: regr.coef_ # very sparse coefficients

```

○

Scikit Learn: 7

载入示例数据

一个改变数

学习和预测

分类

K最近邻(KNN)

训练集和测

分类支持向量

线性支持向

使用核

聚类：将观测值

k均值聚类

应用到图像

用主成分分析降

将一切放在一个

线性模型：从正

稀疏模型

同一问题的

```
Out[112]:
array([[ 0.          , -0.          , 497.34075682, 199.17441034,
        -0.          , -0.          , -118.89291545,  0.          ,
        430.9379595 ,  0.          ]])

In [113]: regr.score(diabetes_X_test, diabetes_y_test)
Out[113]: 0.55108354530029791
```

这个分数和线性回归(最小二乘法)非常相似：

```
In [114]: lin = linear_model.LinearRegression()

In [115]: lin.fit(diabetes_X_train, diabetes_y_train)
Out[115]: LinearRegression(copy_X=True, fit_intercept=True, normalize=False)

In [116]: lin.score(diabetes_X_test, diabetes_y_test)
Out[116]: 0.58507530226905713
```

同一问题的不同算法

同一数学问题可以用不同算法解决。例如,sklearn中的Lasso对象使用坐标下降(coordinate descent)方法¹⁰解决套索回归,这在大数据集时非常有效率。然而,sklearn也提供了LassoLARS对象,使用LARS这种在解决权重向量估计非常稀疏,观测值很少的问题很有效率的方法。

模型选择：选择估计器和它们的参数

格点搜索和交叉验证估计器

格点搜索

scikit-learn提供了一个对象,该对象给定数据,在拟合一个参数网格的估计器时计算分数,并且选择参数最大化交叉验证分数。这个对象在构建时采用一个估计器并且暴露一个估计器API：

```
In [117]: from sklearn import svm, grid_search

In [118]: gammas = np.logspace(-6, -1, 10)

In [119]: svc = svm.SVC()

In [120]: clf = grid_search.GridSearchCV(estimator=svc, param_grid=dict(gamma=gammas),n_jobs=-1)

In [121]: clf.fit(digits.data[:1000], digits.target[:1000])
Out[121]:
GridSearchCV(cv=None,
             estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0, degree=3, gamma=0.0,
                           kernel='rbf', probability=False, shrinking=True, tol=0.001,
                           verbose=False),
             fit_params={}, iid=True, loss_func=None, n_jobs=-1,
             param_grid={'gamma': array([ 1.00000e-06,  3.59381e-06,  1.29155e-05,  4.64159e-05,
                                           1.66810e-04,  5.99484e-04,  2.15443e-03,  7.74264e-03,
                                           2.78256e-02,  1.00000e-01])},
             pre_dispatch='2*n_jobs', refit=True, score_func=None, verbose=0)

In [122]: clf.best_score
/usr/lib/python2.7/site-packages/sklearn/utils/__init__.py:79: DeprecationWarning: Function best_score is deprecated
warnings.warn(msg, category=DeprecationWarning)
Out[122]: 0.98600097103091122

In [123]: clf.best_estimator.gamma
/usr/lib/python2.7/site-packages/sklearn/utils/__init__.py:79: DeprecationWarning: Function best_estimator is deprecated
warnings.warn(msg, category=DeprecationWarning)
Out[123]: 0.0021544346900318843
```

默认GridSearchCV使用三次(3-fold)交叉验证。然而,如果它探测到一个分类器被传递,而不是一个回归量,它使用分层的3次。

交叉验证估计器

交叉验证在一个algorithm by algorithm基础上可以更有效地设定参数。这就是为何,对给定的估计器,scikit-learn使用“CV”估计器,通过交叉验证自动设定参数。

```
In [125]: from sklearn import linear_model, datasets
```

```
In [126]: lasso = linear_model.LassoCV()

In [127]: diabetes = datasets.load_diabetes()

In [128]: X_diabetes = diabetes.data

In [129]: y_diabetes = diabetes.target

In [130]: lasso.fit(X_diabetes, y_diabetes)
Out[130]:
LassoCV(alphas=array([ 2.14804,  2.00327, ...,  0.0023 ,  0.00215]),
        copy_X=True, cv=None, eps=0.001, fit_intercept=True, max_iter=1000,
        n_alphas=100, normalize=False, precompute='auto', tol=0.0001,
        verbose=False)

In [131]: # The estimator chose automatically its lambda:

In [132]: lasso.alpha
Out[132]: 0.013180196198701137
```

这些估计器是相似的，以‘CV’为它们名字的后缀。

练习

对糖尿病数据集，找到最优的正则化参数alpha。(0.016249161908773888)

Footnotes

1. 什么手写数字数据集↩
2. KNN分类算法↩
3. Ball tree数据结构↩
4. 支持向量机↩
5. 径向基函数↩
6. 看看wikipedia吧↩
7. 监督学习↩
8. Penalty methods↩
9. LASSO method↩
10. Coordinate descent↩

© 著作权归作者所有

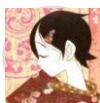
分类：工作日志 字数：3328

打赏

点赞

收藏

分享



yyliu

无锡


+ 关注

粉丝 29 | 博文 14 | 码字总数 20045




相关博客

Python机器学习库scikit-learn实践

 幽幽幽幽古溪


920

scikit-learn使用笔记与sign prediction简单小结

 rxpert

360


scikit-learn多分类probability输出

 AC-carrot

220

评论 (7)

Ctrl+Enter 发表评论

 **OldPanda**

1楼 2013/01/23 10:18

您好！请教一个问题，我在用scikit learn进行SVM的训练时，发现最后得出的svc类的大小和训练样本的数量呈线性关系，这正常么？

 **yyliu**

2楼 2013/01/23 18:09

引用来自“OldPanda”的评论

您好！请教一个问题，我在用scikit learn进行SVM的训练时，发现最后得出的svc类的大小和训练样本的数量呈线性关系，这正常么？

不清楚.....svc类大小是什么.....

 **OldPanda**

3楼 2013/01/25 12:06


引用来自“yyliu”的评论

引用来自“OldPanda”的评论

您好！请教一个问题，我在用scikit learn进行SVM的训练时，发现最后得出的svc类的大小和训练样本的数量呈线性关系，这正常么？

不清楚.....svc类大小是什么.....

我把最后的训练结果存成二进制文件了，指文件的大小

 **yyliu**

4楼 2013/01/25 13:33

引用来自“OldPanda”的评论

引用来自“yyliu”的评论


引用来自“OldPanda”的评论

您好！请教一个问题，我在用scikit learn进行SVM的训练时，发现最后得出的svc类的大小和训练样本的数量呈线性关系，这正常么？

不清楚.....svc类大小是什么.....

我把最后的训练结果存成二进制文件了，指文件的大小

从来没存储过.....不知道

 **OldPanda**

5楼 2013/01/25 20:07

引用来自“yyliu”的评论

引用来自“OldPanda”的评论

引用来自“yyliu”的评论

引用来自“OldPanda”的评论

您好！请教一个问题，我在用scikit learn进行SVM的训练时，发现最后得出的svc类的大小和训练样本的数量呈线性关系，这正常么？

不清楚.....svc类大小是什么.....

我把最后的训练结果存成二进制文件了，指文件的大小

https://my.oschina.net/u/175377/blog/84420

9/10

从来没存储过.....不知道

额，不过还是谢谢你了：)



yyliu

6楼 2013/01/26 21:55

引用来自“OldPanda”的评论

引用来自“yyliu”的评论

引用来自“OldPanda”的评论

引用来自“yyliu”的评论

引用来自“OldPanda”的评论

您好！请教一个问题，我在用scikit learn进行SVM的训练时，发现最后得出的svc类的大小和训练样本的数量呈线性关系，这正常么？

不清楚.....svc类大小是什么.....

我把最后的训练结果存成二进制文件了，指文件的大小

从来没存储过.....不知道

额，不过还是谢谢你了：)

抱歉，帮不上忙了



yyliu

7楼 2015/10/12 12:49

引用来自“OldPanda”的评论

引用来自“yyliu”的评论

引用来自“OldPanda”的评论

引用来自“yyliu”的评论

引用来自“OldPanda”的评论

您好！请教一个问题，我在用scikit learn进行SVM的训练时，发现最后得出的svc类的大小和训练样本的数量呈线性关系，这正常么？

不清楚.....svc类大小是什么.....

我把最后的训练结果存成二进制文件了，指文件的大小

从来没存储过.....不知道

额，不过还是谢谢你了：)

两年后忽然想起这个问题，猜想和svm关系应该不大，而是特征抽取过程造成了更多的新特性。你是在做文本分类？