一道倾城

基于scikit-learn包实现机器学习之KNN(K近邻)-完整示例

基于scikit-learn包实现机器学习之KNN(K近邻)

scikit-learn（简称sklearn）是目前最受欢迎，也是功能最强大的一个用于机器学习的Python库件。它广泛地支持各种分类、聚类以及回归分析方法比如支持向量机、随机森林、DBSCAN等等，由于其强大的功能、优异的拓展性以及易用性，目前受到了很多数据科学从业者的欢迎，也是业界相当著名的一个开源项目之一。

基于上一篇的k近邻原理讲解，我们这一片主要是利用相应的工具包实现机器学习，为了逐步掌握这样成功的工具包，我们从简单的KNN开始入手，其中scikit-learn就是一个不错的选择。

sklearn的安装

首先我们需要安装sklearn这个库（由于您可以在我们的网页上进行所有的实验，所以您可以在之后再尝试在您的电脑上安装sklearn）。sklearn是Python的扩展库，因此我们必须首先设置好Python运行环境。同时，由于sklearn基于Numpy和Scipy这两个库，我们也要首先安装好它们。然后，我们便可以使用pip或conda来自动安装sklearn，具体方法如下：

```
# 安装sklearn之前必须先安装较新版本的Scipy与Numpy
# 使用pip安装sklearn:
pip install -U scikit-learn
# 使用conda安装sklearn:
conda install scikit-learn
```

安装好sklearn之后，我们便可以在Python脚本中使用来自sklearn中的各种数据、功能函数等等。

sklearn内置数据集

数据是机器学习的关键，在机器学习工作中我们需要花费大量的时间来采集和整理数据，合理且科学的数据是得到良好机器学习效果的关键。一般而言，一个分类问题的机器学习过程需要用到四块数据内容，分别是：

- 训练数据，一般用train来表示
- 训练数据的分类属性，一般用target来表示
- 测试数据，一般用test来表示
- 测试数据的真实分类属性，用于评估分类器性能，一般用expected来表示

为了方便学习和测试机器学习中的各种内容，sklearn内置了各种有用的数据集，文本处理、图像识别等具有代表性的问题的数据在sklearn中均有收集(对于初学者来说，不得不说很人性化)。

昵称：一道倾城
园龄：1年9个月
粉丝：1
关注：3
+加关注

< 2018年7月 >

日	一	二	三	四	五	六
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

搜索

找找看
谷歌搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

SVM(3)
Python(3)
sklearn(3)
支持向量机(2)
KNN(2)
K近邻(1)
Machine Learning in Action(1)
scikit-learn(1)
SVC参数分析(1)
代码(1)
更多

随笔分类

机器学习(4)
算法分析与设计(3)

随笔档案

2017年4月 (6)
2017年1月 (1)

最新评论

1. Re:0-1背包问题算法详解及实现
您好，我们这支持数学公式，详见
--博客园团队

本文中用的鸢尾花数据集同样可以在sklearn中的datasets模块中找到。

KNN算法实现

不多说，直接先上代码，后面再进行详解。

```
#-*-coding:utf-8 -*-
from sklearn import datasets
#导入内置数据集模块
from sklearn.neighbors import KNeighborsClassifier
#导入sklearn.neighbors模块中KNN类
import numpy as np
np.random.seed(0)
#设置随机种子，不设置的话默认是按系统时间作为参数，因此每次调用随机模块时产生的随机数都不一样设置后每次产生的一样
iris=datasets.load_iris()
#导入鸢尾花的数据集，iris是一个类似于结构体的东西，内部有样本数据，如果是监督学习还有标签数据
iris_x=iris.data
 #样本数据150*4二维数据，代表150个样本，每个样本4个属性分别为花瓣和花萼的长、宽
iris_y=iris.target
#长150的以为数组，样本数据的标签
indices = np.random.permutation(len(iris_x))
#permutation接收一个数作为参数(150),产生一个0-149一维数组，只不过是随机打乱的，当然她也可以接收一个一维数组作为参数，结果是直接对这个数组打乱
iris_x_train = iris_x[indices[:-10]]
 #随机选取140个样本作为训练数据集
iris_y_train = iris_y[indices[:-10]]
#并且选取这140个样本的标签作为训练数据集的标签
iris_x_test  = iris_x[indices[-10:]]
 #剩下的10个样本作为测试数据集
iris_y_test  = iris_y[indices[-10:]]
#并且把剩下10个样本对应标签作为测试数据及的标签

knn = KNeighborsClassifier()
#定义一个knn分类器对象
knn.fit(iris_x_train, iris_y_train)
#调用该对象的训练方法，主要接收两个参数：训练数据集及其样本标签

iris_y_predict = knn.predict(iris_x_test)
 #调用该对象的测试方法，主要接收一个参数：测试数据集
probility=knn.predict_proba(iris_x_test)
 #计算各测试样本基于概率的预测
neighborpoint=knn.kneighbors(iris_x_test[-1],5,False)
#计算与最后一个测试样本距离在最近的5个点，返回的是这些样本的序号组成的数组
score=knn.score(iris_x_test,iris_y_test,sample_weight=None)
#调用该对象的打分方法，计算出准确率

print('iris_y_predict = ')
print(iris_y_predict)
#输出测试的结果

print('iris_y_test = ')
print(iris_y_test)
#输出原始测试数据集的正确标签，以方便对比
print 'Accuracy:',score
#输出准确率计算结果
print 'neighborpoint of last test sample:',neighborpoint
```


阅读排行榜

1. 基于scikit-learn包实现机器学习之KNN(K近邻)-完整示例(8417)
2. 从核函数到SVM原理--sklearn-SVM实现(7233)
3. sklearn-SVC实现与类参数(4386)
4. 0-1背包问题优化算法详解(795)
5. 0-1背包问题算法详解及实现(741)

评论排行榜

1. 0-1背包问题算法详解及实现(1)

```
print 'probility:',probility
************************************
结果输出：
iris_y_predict =
[1 2 1 0 0 0 2 1 2 0]
iris_y_test =
[1 1 1 0 0 0 2 1 2 0]
Accuracy: 0.9
neighborpoint of last test sample: [[ 75  41  96  78 123]]
probility: [[ 0.   1.   0. ]
            [ 0.   0.4  0.6]
            [ 0.   1.   0. ]
            [ 1.   0.   0. ]
            [ 1.   0.   0. ]
            [ 1.   0.   0. ]
            [ 0.   0.   1. ]
            [ 0.   1.   0. ]
            [ 0.   0.   1. ]
            [ 1.   0.   0. ]]
```

基于此，KNN算法实现已经彻底完成了，当然，我们还是需要稍微再解读一下，上面所说的主要两个对象的方法，一个是fit()方法，一个是predict()方法，我说的主要接受两个参数，并不是说只接受两个参数或者一个参数，而是说其他参数都有默认值，且是内部参数，这里我们详细解读。KNeighborsClassifier是一个类，它集成了其他的NeighborsBase, KNeighborsMixin,SupervisedIntegerMixin, ClassifierMixin。这里我们暂时不管它。主要看它的几个主要的方法。当然有的方法是其从父类那里集成过来的。

__init__()　初始化函数(构造函数) 它主要有一下几个参数：

n_neighbors=5

int 型参数　knn算法中指定以最近的几个最近邻样本具有投票权，默认参数为5

weights='uniform'

str参数　　即每个拥有投票权的样本是按什么比重投票，'uniform'表示等比重投票，'distance'表示按距离反比投票，[callable]表示自己定义的一个函数，这个函数接收一个

距离数组，返回一个权值数组。默认参数为'uniform'

algrithm='auto'

str参数　　即内部采用什么算法实现。有以下几种选择参数：'ball_tree':球树、'kd_tree':kd树、'brute':暴力搜索、'auto':自动根据数据的类型和结构选择合适的算法。默认情况下是'auto'。暴力搜索就不用说了大家都知道。具体前两种树型数据结构哪种好视情况而定。KD树是对依次对K维坐标轴，以中值切分构造的树,每一个节点是一个超矩形，在维数小于20时效率最高--可以参看《统计学习方法》第二章。ball tree 是为了克服KD树高维失效而发明的，其构造过程是以质心C和半径r分割样本空间，每一个节点是一个超球体。一般低维数据用kd_tree速度快，用ball_tree相对较慢。超过20维之后的高维数据用kd_tree效果反而不佳，而ball_tree效果要好，具体构造过程及优劣势的理论大家有兴趣可以去具体学习。

leaf_size=30

int参数　　基于以上介绍的算法，此参数给出了kd_tree或者ball_tree叶节点规模，叶节点的不同规模会影响数的构造和搜索速度，同样会影响储树的内存的大小。具体最优规模是多少视情况而定。

matric='minkowski'

str或者距离度量对象　即怎样度量距离。默认是闵氏距离，闵氏距离不是一种具体的距离度量方法，它可以说包括了其他距离度量方式，是其他距离度量的推广，具体各种距离度量只是参数p的取值不同或者是否去极限的不同情况，具体大家可以参考这里，讲的非常详细

$$dist(x,y)=(\sum_{i=1}^{n}|x_i-y_i|^p)^{\frac{1}{p}}$$

p=2

int参数　　就是以上闵氏距离各种不同的距离参数，默认为2，即欧氏距离。p=1代表曼哈顿距离等等

metric_params=None

距离度量函数的额外关键字参数，一般不用管，默认为None

n_jobs=1

int参数　　指并行计算的线程数量，默认为1表示一个线程，为-1的话表示为CPU的内核数，也可以指定为其他数量的线程，这里不是很追求速度的话不用管，需要用到的话去看看多线程。

fit()

训练函数，它是最主要的函数。接收参数只有1个，就是训练数据集，每一行是一个样本，每一列是一个属性。它返回对象本身，即只是修改对象内部属性，因此直接调用就可以了，后面用该对象的预测函数取预测自然及用到了这个训练的结果。其实该函数并不是KNeighborsClassifier这个类的方法，而是它的父类SupervisedIntegerMixin继承下来的方法。

predict()

预测函数　接收输入的数组类型测试样本，一般是二维数组，每一行是一个样本，每一列是一个属性返回数组类型的预测结果，如果每个样本只有一个输出，则输出为一个一维数组。如果每个样本的输出是多维的，则输出二维数组，每一行是一个样本，每一列是一维输出。

predict_prob()

基于概率的软判决，也是预测函数，只是并不是给出某一个样本的输出是哪一个值，而是给出该输出是各种可能值的概率各是多少接收参数和上面一样返回参数和上面类似，只是上面该是值的地方全部替换成概率，比如说输出结果又两种选择0或者1，上面的预测函数给出的是长为n的一维数组，代表各样本一次的输出是0还是1.而如果用概率预测函数的话，返回的是n*2的二维数组，每一行代表一个样本，每一行有两个数，分别是该样本输出为0的概率为多少，输出1的概率为多少。而各种可能的顺序是按字典顺序排列，比如先0后1，或者其他情况等等都是按字典顺序排列。

score()

计算准确率的函数，接受参数有3个。 X:接收输入的数组类型测试样本，一般是二维数组，每一行是一个样本，每一列是一个属性。y:X这些预测样本的真实标签，一维数组或者二维数组。sample_weight=None,是一个和X第一位一样长的各样本对准确率影响的权重，一般默认为None.输出为一个float型数，表示准确率。内部计算是按照predict()函数计算的结果记性计算的。其实该函数并不是KNeighborsClassifier这个类的方法，而是它的父类KNeighborsMixin继承下来的方法。

kneighbors()

计算某些测试样本的最近的几个近邻训练样本。接收3个参数。X=None：需要寻找最近邻的目标样本。n_neighbors=None,表示需要寻找目标样本最近的几个最近邻样本，默认为None,需要调用时给出。return_distance=True:是否需要同时返回具体的距离值。返回最近邻的样本在训练样本中的序号。其实该函数并不是KNeighborsClassifier这个类的方法，而是它的父类KNeighborsMixin

继承下来的方法。

下面举一个简单例子让大家稍微看看，（其实上面的例子已经很具体形象了）：

```
Examples
--------
>>> X = [[0], [1], [2], [3]]
>>> y = [0, 0, 1, 1]
>>> from sklearn.neighbors import KNeighborsClassifier
>>> neigh = KNeighborsClassifier(n_neighbors=3)
>>> neigh.fit(X, y)
>>> print(neigh.predict([[1.1]]))
[0]
>>> print(neigh.predict_proba([[0.9]]))
[[ 0.66666667  0.33333333]]
```

以上讲解关于sklearn中的knn算法基本讲解完了，不知道大家是否学会了呢？以上部分除了sklearn背景介绍，后面的部分为原创，希望大家共同学习共同进步，后续会继续更新sklearn的其他方法。

分类: 机器学习

标签: 机器学习, KNN, K近邻, Python, sklearn, scikit-learn

好文要顶 关注我 收藏该文

一道倾城
关注 - 3
粉丝 - 1
+加关注

0 0

« 上一篇：动态规划思想详解及示例实现
» 下一篇：从核函数到SVM原理--sklearn-SVM实现

posted @ 2017-04-09 10:56 一道倾城 阅读(8416) 评论(0) 编辑 收藏

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

【推荐】超50万VC++源码: 大型组态工控、电力仿真CAD与GIS源码库！
【推荐】如何快速搭建人工智能应用？
【大赛】2018首届"顶天立地"AI开发者大赛

最新IT新闻:
· 雷军：我不喜欢把稻草卖成金条的人
· Mozilla正在寻找翻译志愿者以推进Firefox优秀扩展本地化
· 新东方在线赴港递交招股书 启动上市计划
· 国际计算语言学协会成立亚太区分会 王海峰任创始主席
· "微信转账转错人"收钱方已还钱，诚信原则应坚守
» 更多新闻...

最新知识库文章:
· 危害程序员职业生涯的三大观念
· 断点单步跟踪是一种低效的调试方法
· 测试 | 让每一粒尘埃有的放矢
· 从Excel到微服务
· 如何提升你的能力？给年轻程序员的几条建议
» 更多知识库文章...

Copyright ©2018 一道倾城