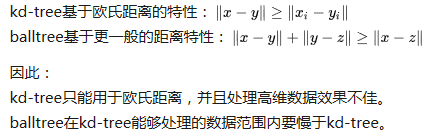
K最近邻分类算法



## KNN算法基本介绍

K-Nearest Neighbor（k最邻近分类算法），简称KNN，是最简单的一种有监督的机器学习算法。也是一种懒惰学习算法，即开始训练仅仅是保存所有样本集的信息，直到测试样本到达才开始进行分类决策。

KNN算法的核心思想是，要想确定测试样本属于哪一类，就先寻找所有训练样本中与该测试样本“距离”最近的前K个样本，然后判断这K个样本中大部分所属的类型，就认为是该测试样本的类型。也就是所谓的“近朱者赤近墨者黑”，根据与其最近的k个样本的类型决定其自身的类型。因此K的确定和测算距离的方式是影响样本最终分类准确率的重要因素。

常用的测算距离的方法是多维空间的欧式距离法。

其优点为：易于理解，实现简单，无需估计参数，无需训练。

不足之处为：需要保存所有的训练数据，内存开销大，而且训练数据较多时会导致很高的算法复杂度，训练数据类型不均匀可能会导致预测准确率下降。

## KNN算法数据说明

我们将采用scikit-learn库中自带的鸢尾花数据集进行测试。可以在D:\anaconda python\pkgs\scikit-learn-0.19.0-np113py36\_0\Lib\site-packages\sklearn\datasets\data路径下查看元数据，部分数据实例如下图（共计150条数据）。



鸢尾花数据集采集的是鸢尾花的测量数据（特征属性）以及其所属的类别。

测量数据特征包括: 萼片长度、萼片宽度、花瓣长度、花瓣宽度

所属类别有三类: Iris Setosa，Iris Versicolour，Iris Virginica ，用数字0,1,2表示。

通过python加载该数据集

from sklearn import datasets

iris = datasets.load\_iris()

# 属性数据并查看数据特征

iris\_X = iris.data

print(iris.data.shape)

# 类别数据

iris\_y = iris.target

并且下面两种方式将该数据集拆分为训练数据和测试数据：

# 方法一：使用python的train\_test\_split库进行数据集拆分

iris\_train\_X , iris\_test\_X, iris\_train\_y ,iris\_test\_y = train\_test\_split(iris\_X, iris\_y, test\_size=0.2,random\_state=0)

# 方法二：随机选择部分数据作为测试集或训练集（适用于少量数据）

np.random.seed(0)

# permutation随机生成鸢尾花总条数的(150)序列集

select = np.random.permutation(len(iris\_y))

iris\_X\_train = iris\_X[select[:-30]]

iris\_y\_train = iris\_y[indices[:-30]]

iris\_X\_test = iris\_X[indices[-30:]]

iris\_y\_test = iris\_y[indices[-30:]]

# 方法一：使用python的train\_test\_split库进行数据集拆分

## KNN算法python实现

在此我们将直接使用python的scikit-learn 库中的 neighbors.KNeighborsClassifier类，通过KNN算法对鸢尾花进行分类。

**# 首先进行类的初始化**

knn =KNeighborsClassifier(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=1, n\_neighbors=5, p=2,

weights='uniform')

参数介绍

* n\_neighbors=5，就是KNN中的k，默认为5。
* weights='uniform'，是距离计算中使用的权重，默认为'uniform' 是等权加权，也可以选'distance'是按照距离的倒数进行加权，也可以自己设置其他加权方式。（给距离增加权重，如果越近的距离权重越高，能在一定程度上避免样本分布不平均的问题）
* metric='minkowski'、p=2，表示采用的是欧氏距离的计算。

计算距离的方式默认为闵可夫斯基距离，是一组距离的定义。

两个n维变量a(x11,x12,…,x1n)与 b(x21,x22,…,x2n)间的闵可夫斯基距离：

。其中当p=1时，为曼哈顿距离；当p=2时，为欧氏距离；当p→∞时，为切比雪夫距离。根据变参数p的不同，闵氏距离可以表示一类的距离。

* algorithm='auto'，是分类时采取的算法，有'brute'、'kd\_tree'和'ball\_tree'，三种默认按照数据特征从这三种中选择最合适的。其中kd-tree基于欧氏距离的特性可以快速处理20维以内的数据集，balltree基于更一般的距离特性，适合处理高维数据。（三种算法的具体实现之后会进行详细介绍）
* leaf\_size=30，是kd\_tree或ball\_tree生成的树的树叶（二叉树中未分枝的节点）的大小。
* n\_job=1，是并行计算的线程数量，默认是1，输入-1则设为CPU的内核数。

**# 提供数据集进行训练**

knn.fit(iris\_X\_train, iris\_y\_train)

**# 预测测试集数据鸢尾花类型**

predict\_result = knn.predict(iris\_X\_test)

print(predict\_result)

**# 计算预测的准确率**

print(knn.score(iris\_X\_test, iris\_y\_test))

完整代码

*#!/usr/bin/python  
# -\*- coding: utf-8 -\*-*\_\_author\_\_ **= 'ziyi'***# KNN调用***import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
**from** sklearn **import** datasets  
**from** sklearn.neighbors **import** KNeighborsClassifier  
**from** sklearn.model\_selection **import** train\_test\_split  
  
**from** sklearn **import** datasets  
iris **=** datasets.**load\_iris**()  
print(iris.data.shape)  
*# 拆分属性数据*iris\_X **=** iris.data  
*# 拆分类别数据*iris\_y **=** iris.target  
  
print(len(iris\_y))  
*# plt.plot(iris\_X)  
# plt.show()*a **=** np.**unique**(iris\_y)  
print(np.**unique**(iris\_y))  
print(np.**zeros**(a.shape))*#按照原矩阵格式以0填充所有数据  
# Split iris data in train and test data  
# A random permutation, to split the data randomly  
  
# 方法一：拆分测试集和训练集,并进行预测*iris\_train\_X , iris\_test\_X, iris\_train\_y ,iris\_test\_y **= train\_test\_split**(iris\_X, iris\_y, test\_size**=**0.2,random\_state**=**0)  
knn1 **= KNeighborsClassifier**(n\_neighbors**=**3)  
knn1.**fit**(iris\_train\_X, iris\_train\_y)  
knn1.**predict**(iris\_test\_X)  
print (iris\_test\_y)  
  
  
*# 方法二：拆分测试集和训练集*np.random.**seed**(0)  
*# permutation随机生成0-150的系列*indices **=** np.random.**permutation**(len(iris\_y))  
iris\_X\_train **=** iris\_X[indices[**:-**30]]  
iris\_y\_train **=** iris\_y[indices[**:-**30]]  
iris\_X\_test **=** iris\_X[indices[**-**30**:**]]  
iris\_y\_test **=** iris\_y[indices[**-**30**:**]]  
knn **= KNeighborsClassifier**()  
*# 提供训练集进行顺利*knn.**fit**(iris\_X\_train, iris\_y\_train)  
*# 预测测试集数据鸢尾花类型*predict\_result **=** knn.**predict**(iris\_X\_test)  
print(predict\_result)  
print(knn.**score**(iris\_X\_test, iris\_y\_test))  
*# precision, recall, thresholds = precision\_recall\_curve(iris\_y\_test, knn.predict(iris\_X\_test))  
# answer = knn.predict\_proba(iris\_X\_test)[:,1]  
# print(classification\_report(iris\_y\_test, answer ))***KNeighborsClassifier**(algorithm**='auto'**, leaf\_size**=**30, metric**='minkowski'**,  
 metric\_params**=None**, n\_jobs**=**1, n\_neighbors**=**5, p**=**2,  
 weights**='uniform'**)

本次小结