WDS

DataScience@Web

机器学习讨论班

2018年暑期

2. K近邻

王驰

介绍内容

- 基本概念
- 距离的度量、k 值的选取
- 特征归一化
- KD 树的构建与搜索
- 优缺点
- 现场实验
- 课后练习

基本概念

- K 近邻算法是一种**基本分类和回归方法**。本 ppt 只讨论分类问题的 k 近邻法。
- 给定一个训练数据集，对新的输入实例，在训练数据集中找到与该实例**最邻近**的 k 个实例，
- **这 k 个实例的多数属于某个类**，就把该输入实例分类到这个类中。（**类似于现实生活中少数服从多数的思想**）

距离的度量

最常见的**欧氏距离**：

$$\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$$

曼哈顿距离：

$$\sum_{i=1}^{k}\left|x_i - y_i\right|$$

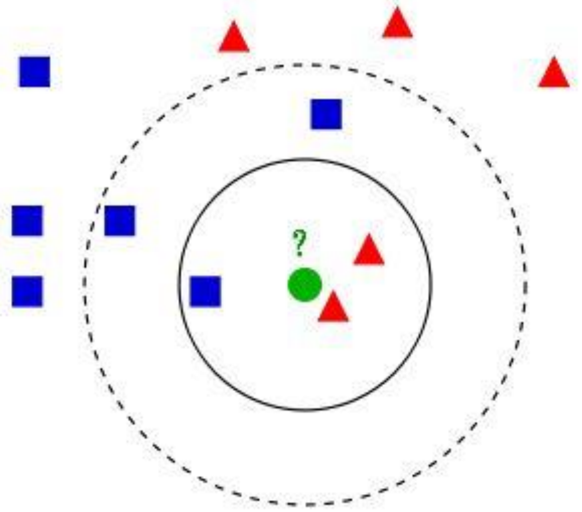
更一般的 Minkowski 距离：

$$\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$$

在实际应用中，距离函数的选择应该根据数据的特性和分析的需要而定

K 值的选取

- 右图中， 有**两类**不同的样本数据：蓝色和红色
- 绿色的圆是**待分类的数据**
- 问题：这个绿色的圆属于蓝色的分类还是红色的分类？



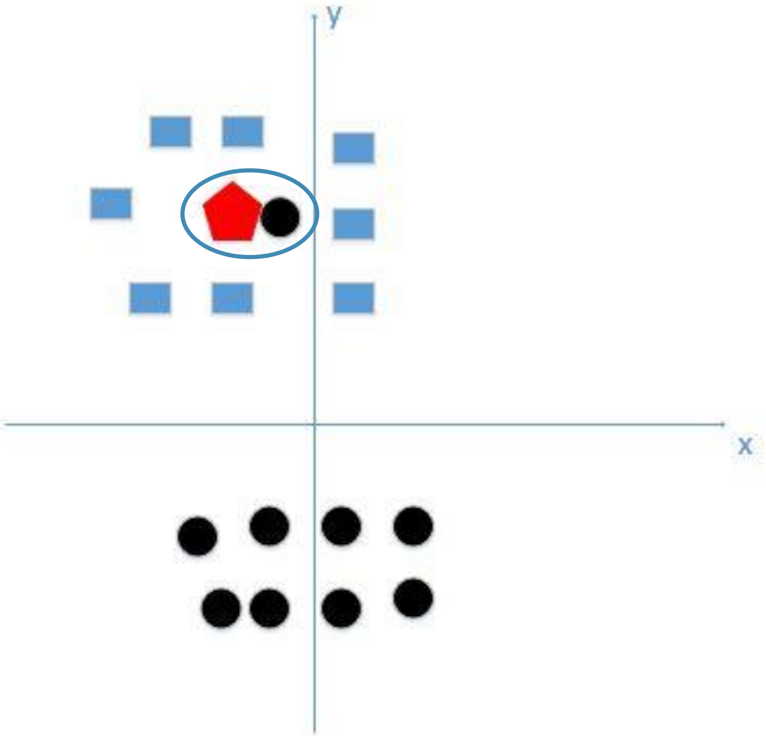
- 如果 k=3，答案是属于红色
- 如果 k=5，答案是属于蓝色

K 值的选取

如果 k 值过小，

- **整体模型会变得复杂**
- **很容易学习到噪声**
- **容易发生过拟合**

右图中，k = 1 时，红色块被分类到**黑色圆点**的类别中

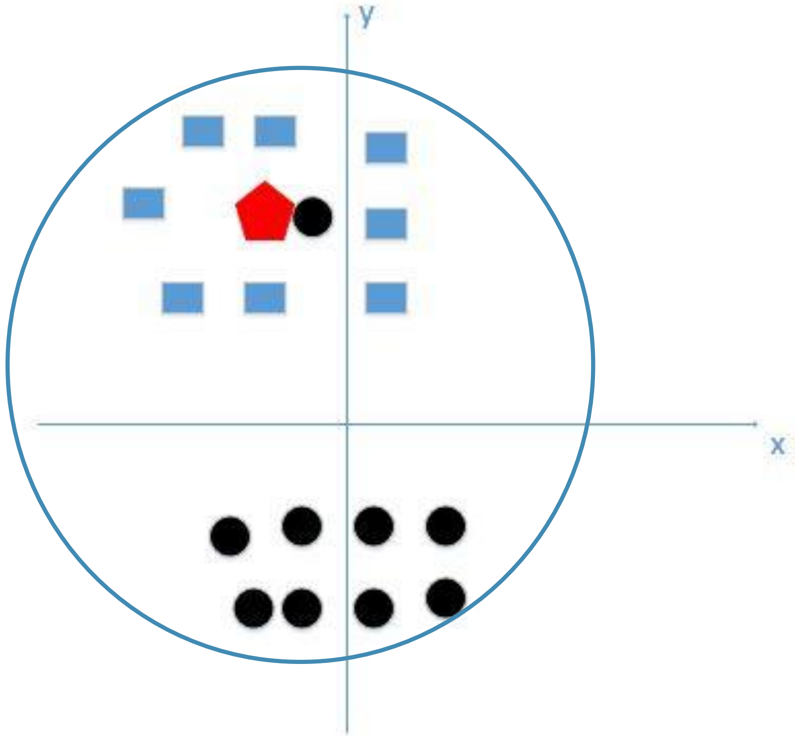


K 值的选取

如果 k 值过大，

- **整体模型变得简单**
- **容易忽略训练数据中有用的信息**

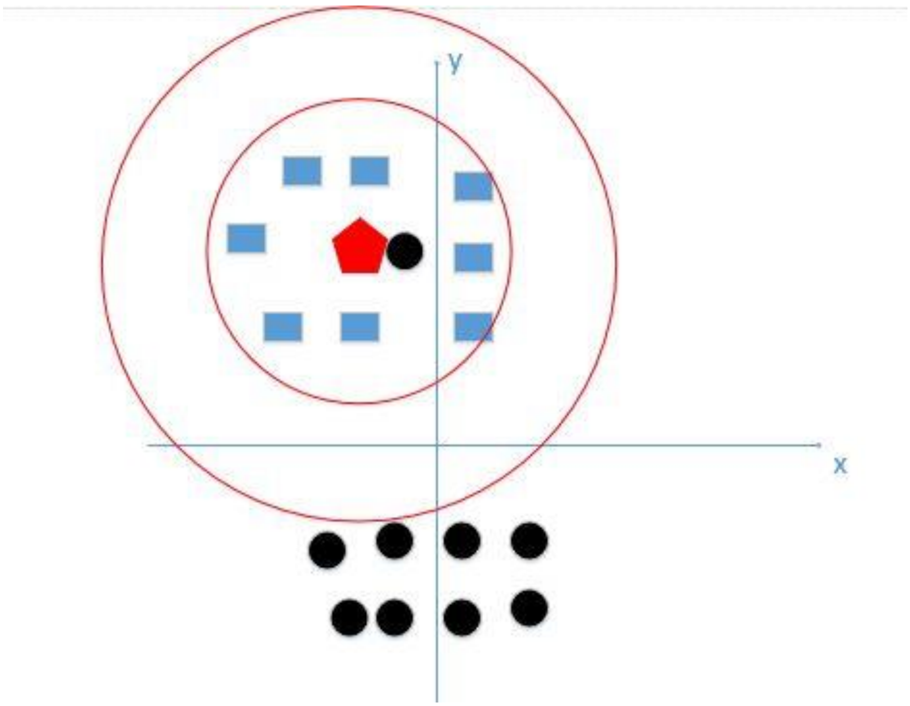
右图中，选取**过大的 k 值**，也会使得红色块被分类到**黑色圆点**的类别中



K 值的选取

既**不能过大**，也**不能过小**。

右图中 k 值的选择，**红色圆边界之间**这个范围是最好的。



如何选取？

实验调参。比如选取一个较小的数值，采取**交叉验证法**来选取最优的 k 值

特征归一化

举例：

- 用一个人**身高(cm)**与**脚码**（尺码）大小来作为**特征值**
- 类别为**男性**或者**女性**
- 有如下 5 个**训练样本**

身高	179	178	165	177	160
尺码	42	43	35	42	35
性别分类	男	男	女	男	女

特征归一化

身高	179	178	165	177	160
尺码	42	43	35	42	35
性别分类	男	男	女	男	女

第一维**身高特征**是第二维**脚码特征**的 **4倍** 左右

距离度量的时**会偏向于第一维特征**

这样造成俩个特征并**不是等价重要**的，可能会导致**距离计算错误**

从而导致**预测错误**

特征归一化

身高	179	178	165	177	160	167
尺码	42	43	35	42	35	43
性别分类	男	男	女	男	女	女?
到测试样本的距离	$\sqrt{145}$	$\sqrt{121}$	$\underline{\boldsymbol{\sqrt{53}}}$	$\underline{\boldsymbol{\sqrt{101}}}$	$\underline{\boldsymbol{\sqrt{103}}}$	

一个女性的脚 43 码的可能性，**远远小于**男性脚 43 码的可能性

但由于各个**特征量纲的不同**，导致了**身高**的重要性已经**远大于脚码**了

归一化的目的： 让每个特征**同等重要**

特征归一化

一般来说，假设样本特征是 $\{(x_{i1},\ x_{i2}, \ldots ,\ x_{in})\}_{i=1}^{m}$，取每个维度的**最大值减最小值：**

$$\boldsymbol{M_j} = \max_{\boldsymbol{i=1,\ldots,m}} \boldsymbol{x_{ij}} - \min_{\boldsymbol{i=1,\ldots,m}} \boldsymbol{x_{ij}}$$

在计算距离时将**每个坐标轴除以相应的 $\boldsymbol{M_j}$** 以实现归一化，即：

$$d\big((y_1, \ldots, y_n), (z_1, \ldots, z_n)\big) = \sqrt{\sum_{j=1}^{n} (\frac{y_j}{M_j} - \frac{z_j}{M_j})^2}$$

KD 树

如何搜索最近的 k 个样本？

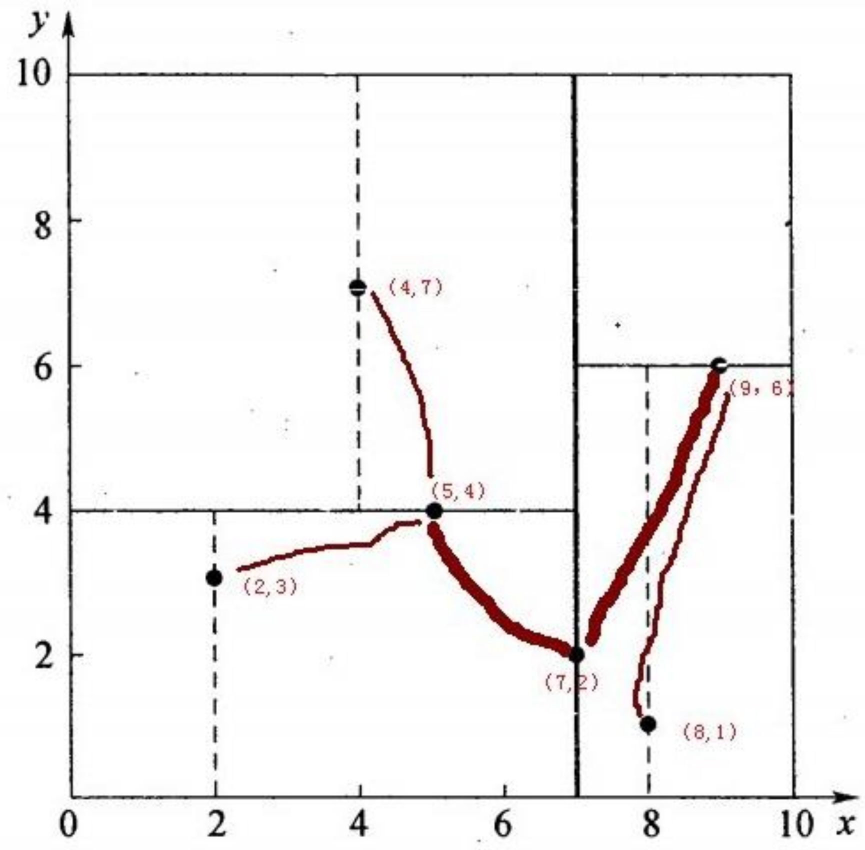
- 线性扫描： 需要计算**每个样本**到输入实例点的距离
- 构建数据索引，如 KD 树：
  - K-dimension tree的缩写
  - 把整个空间划分为特定的几个部分
  - 应用于多维空间关键数据的搜索

KD 树的构建

6 个二维数据点

{（2, 3)，(5, 4)，(9, 6)，(4, 7)，(8, 1)，(7, 2) }

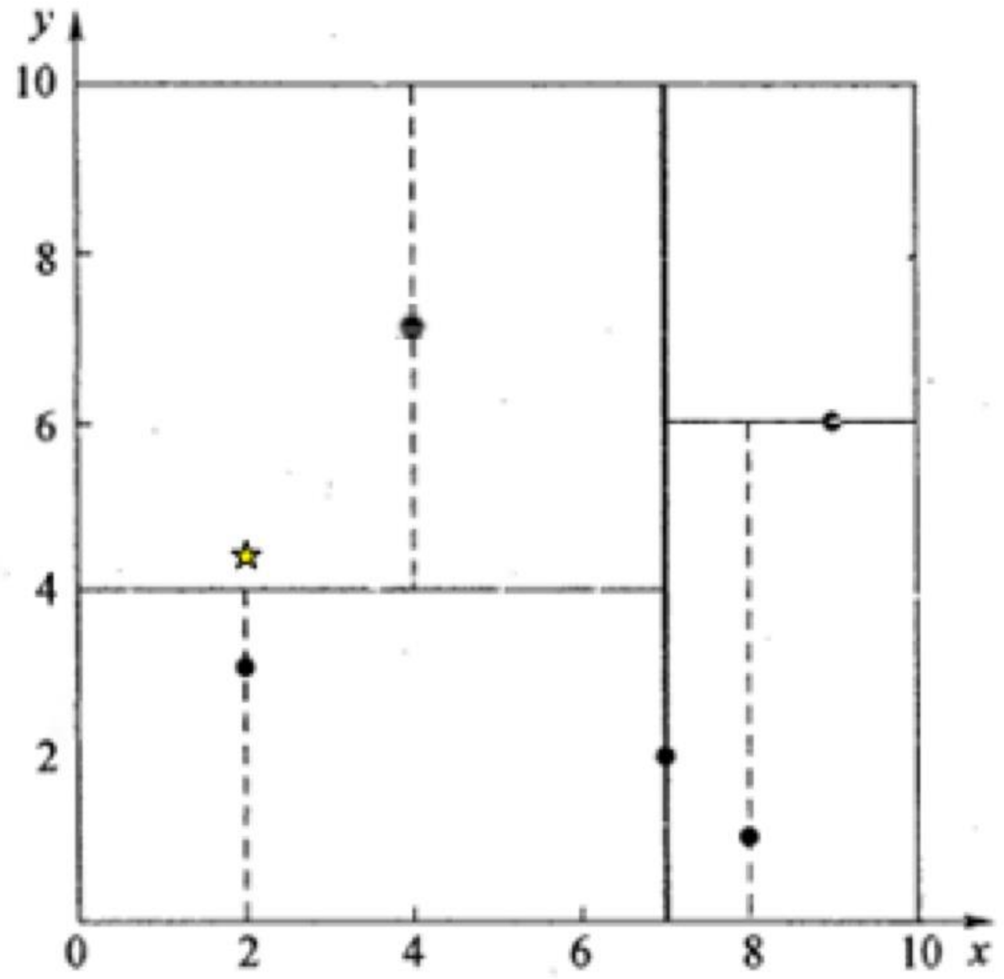
- 确定 split域 = x
  x，y维度上的数据方差分别为39，28.63，所以在**x轴上方差更大**，故split域值为x
- 确定 Node-data = (7, 2)
  7 为 x 维上的中位数
- 分割平面
  x = 7将整个空间左子空间和右子空间
- 递归地分割左子空间和右子空间



KD 树的搜索

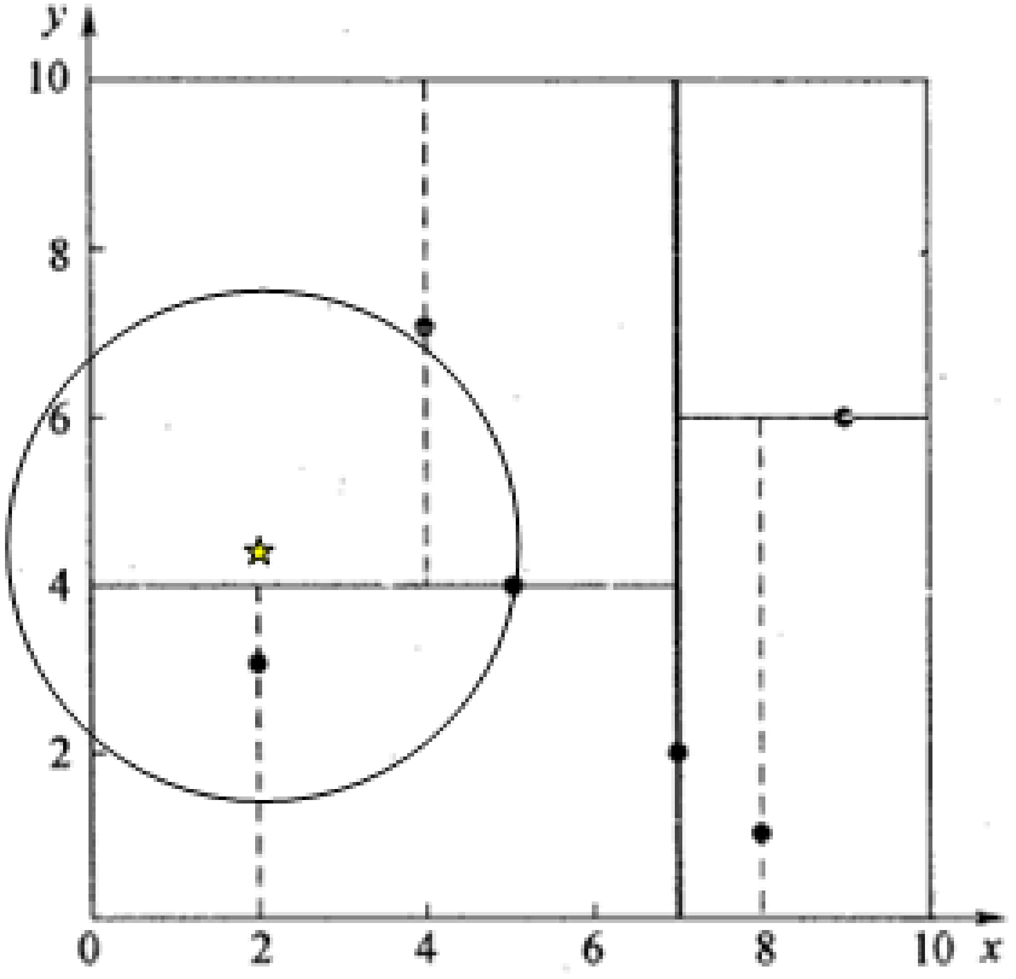
查找（2，4.5）的最近邻

- 二分查找（2, 4.5）所属的区域（即查找 KD 树的叶节点）
  （7, 2）->（5, 4）->（4, 7）
- 取(4, 7)为当前**近似最近邻点**， 计算其与目标查找点的距离：3.202



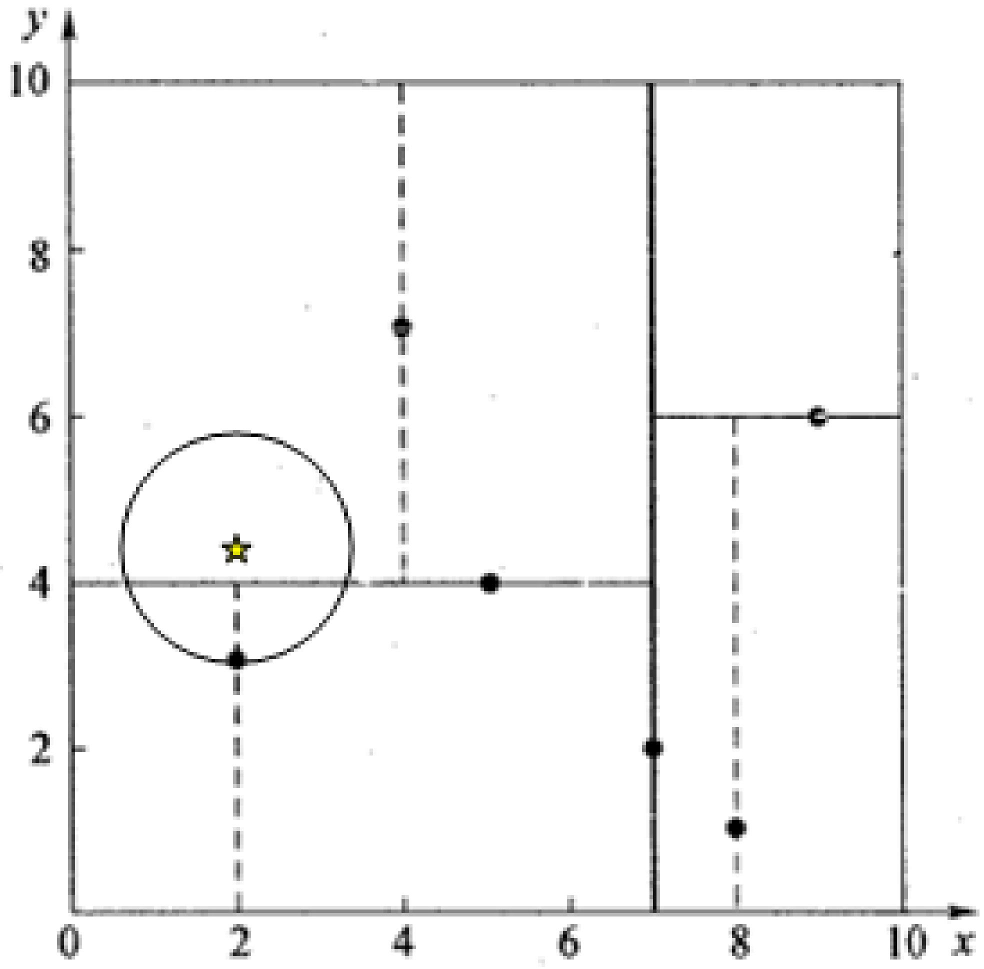
KD 树的搜索

- 回溯到（5, 4），计算其与查找点之间的距离为 3.041（**遇到了更近的点**）
- 以 3.041 为半径作圆，该圆和 y = 4 交割，**说明可能在 (5, 4) 的左子空间有更近的点。**



KD 树的搜索

- 进入 (5, 4) 左子空间进行查找，**在该空间内找到离 (2, 4.5) 最近的叶子节点** (2, 3)
- (2, 3) 距 (2, 4.5) 比 (5, 4) 要近，所以**最近邻点更新为 (2, 3)**
- 回溯至 (7, 2)，以 (2, 4.5) 为圆心作圆，**不和 x = 7 分割超平面交割**。
- 至此，搜索结束，返回最近邻点 (2, 3)



KD 树的搜索

- 二分查找出包含目标点 x 的叶结点
- 以此叶结点为**当前最近点**
- 向上回退，在每个结点进行以下操作
  - 如果该结点保存的实例点比当前最近点距离目标点更近，则以该实例点为**当前最近点**
  - 以**目标点为球心**、以**目标点与当前最近点间的距离**为半径画圆，检查是否和另一子结点对应的区域**相交。**
  - 如果相交，递归地**在另一端进行最近邻搜索**。否则向上回退。
  - 当回退到根结点时，搜索结束。最后的**当前最近点即为最近邻点。**

优缺点

- 优点
  - 简单直观，易于实现
  - **没有显式的学习过程**

    新数据可以直接加入数据集而**不必进行重新训练**

优缺点

- **缺点**
  - 当样本不平衡时，**比如一个类的样本容量很大，其他类的样本容量很小**，输入一个样本的时候，k 个邻近值大多数都是大样本容量的那个类，这时可能会导致分类错误。
  - 计算量较大。当维数较大时，直接利用 KD 树快速检索的性能急剧下降。
    N个节点的 k 维 KD 树搜索过程时间复杂度为 $O(kN^{1-1/k})$

现场实验

- 数据集: MNIST
- 语言：Python 3
- 库：scikit-learn

课后练习

推广在 KD 树中寻找最近邻的算法，思考如何搜索 k 近邻。

谢 谢