概率与学习-KNN

高　阳，李文斌

http://cs.nju.edu.cn/rl

2023年09月12日

近朱者赤
近墨者黑

大纲

回顾

k-近邻分类器

最近邻分类器

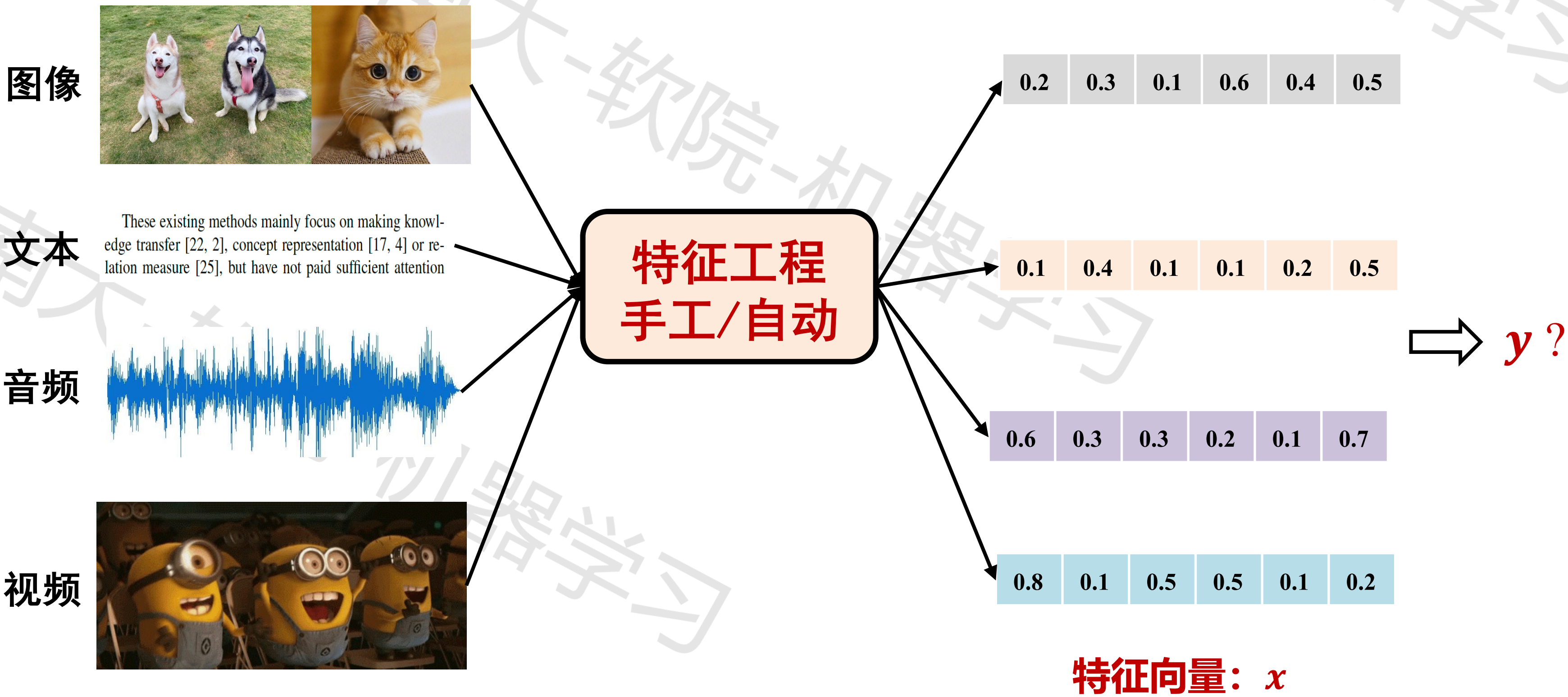
k-近邻回归

降低近邻计算

扩展阅读

回顾

- 统计学角度：机器学习的目的是得到映射 $\mathcal{X} \mapsto \mathcal{Y}$



回顾

- 特征与属性（Feature and Attributes）

图像

特征工程
手工/自动

0.2	0.3	0.1	0.6	0.4	0.5

特征向量：$\boldsymbol{x} \in \mathbb{R}^6$

0.2	0.3	0.1	0.6	0.4	0.5

属性1

属性2

…

属性6

回顾

- 分类问题 Classification
  - 训练集：$D_{train} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_n, y_n)\}$
  - 训练样本：$\boldsymbol{x}_i \in \mathcal{X} \in \mathbb{R}^d$
  - 样本标签：$y_i \in \mathcal{Y} = \{1,2, \dots, C\}$
  - 测试集：$D_{test} = \{\overline{\boldsymbol{x}}_1, \overline{\boldsymbol{x}}_2, \dots, \overline{\boldsymbol{x}}_m\}$

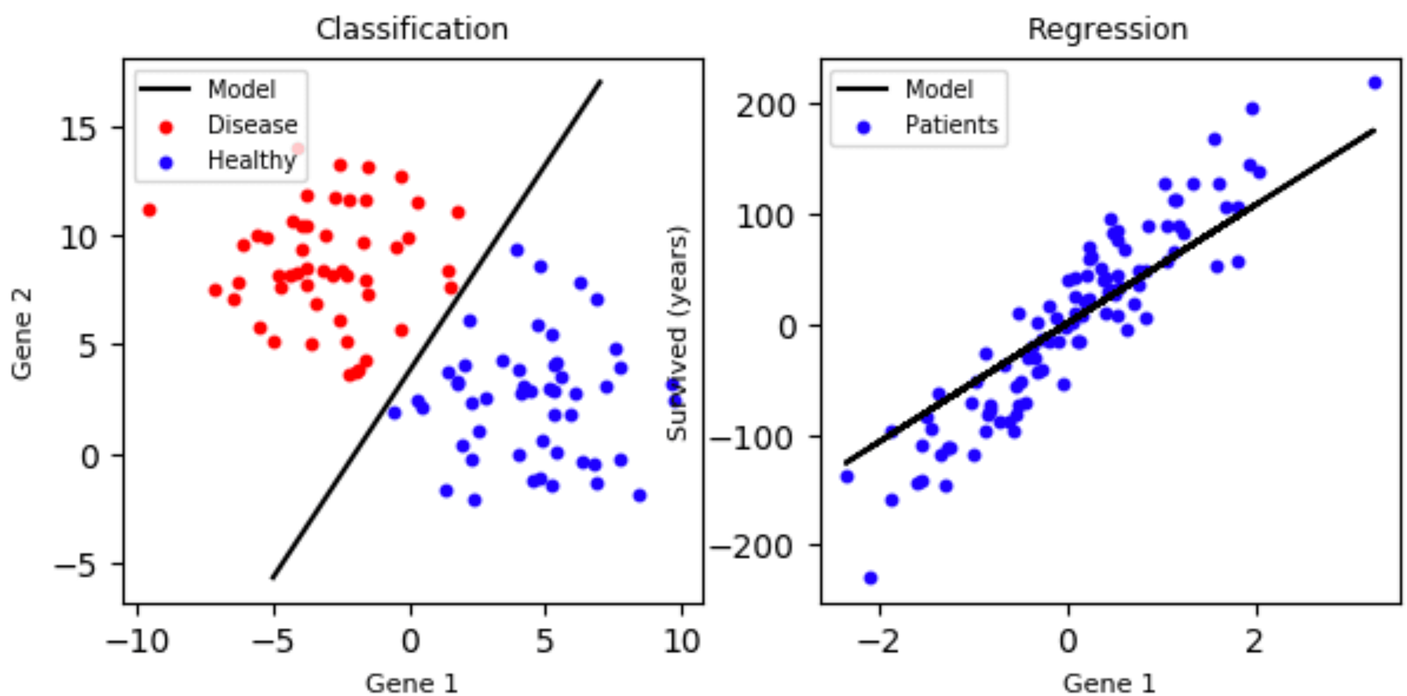
回顾

- 回归问题 Regression
  - 训练集：$D_{train} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_n, y_n)\}$
  - 训练样本：$\boldsymbol{x}_i \in \mathcal{X} \in \mathbb{R}^d$
  - 样本标签：$y_i \in \mathbb{R}$

  - 测试集：$D_{test} = \{\overline{\boldsymbol{x}}_1, \overline{\boldsymbol{x}}_2, \dots, \overline{\boldsymbol{x}}_m\}$

回顾

- 分类VS回归



输出为离散值

输出为连续值

回顾

- 距离度量 ($\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^d$)

欧氏距离 $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T(\boldsymbol{x}_i - \boldsymbol{x}_j)}$

余弦相似性 $\mathrm{s}(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\boldsymbol{x}_i{}^T \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_\mathrm{j}\|}$

曼哈顿距离 $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_{\boldsymbol{j}}\|_1$

切比雪夫距离 $d(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_{\boldsymbol{j}}\|_\infty$

马氏距离 $\mathrm{d}_M(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^T \boldsymbol{M} (\boldsymbol{x}_i - \boldsymbol{x}_j)}$

k-近邻分类器

- k-近邻分类器 (k-Nearest Neighbor Classifier, k-NN)
  - ✓ 算法流程
    - 计算测试样本$\bar{\boldsymbol{x}}$和D_{train}中所有训练样本$\boldsymbol{x}_i$之间的距离$d(\bar{\boldsymbol{x}}, \boldsymbol{x}_i)$
    - 对所有距离值（相似度值）进行升序（降序）排列
    - 选择k个最近（距离最小/相似度最大）的训练样本
    - 采用投票法，将近邻中样本数最多的类别标签分配给$\bar{\boldsymbol{x}}$

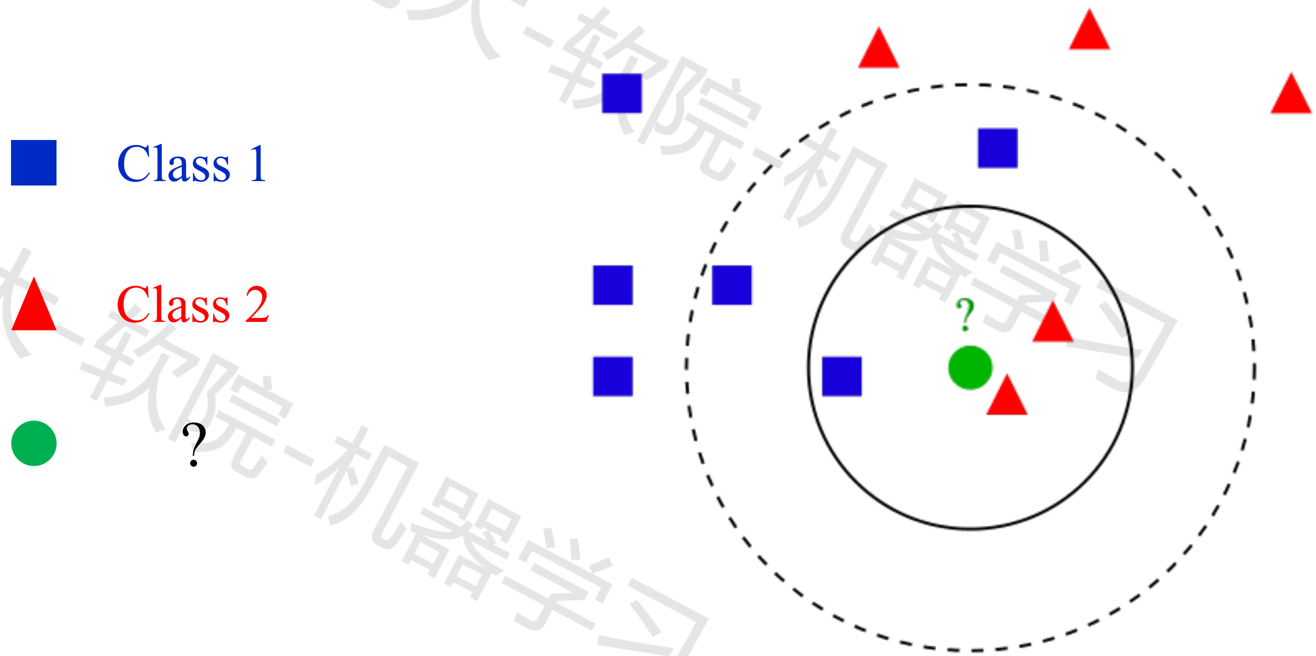
k-近邻分类器

- k的取值的影响

  - k 一般取奇数值，避免平局
  - k 取不同的值，分类结果可能不同
  - k 值较小时，对噪声敏感，整体模型变得复杂，容易过拟合
  - k 值较大时，对噪声不敏感，整体模型变得简单，容易欠拟合

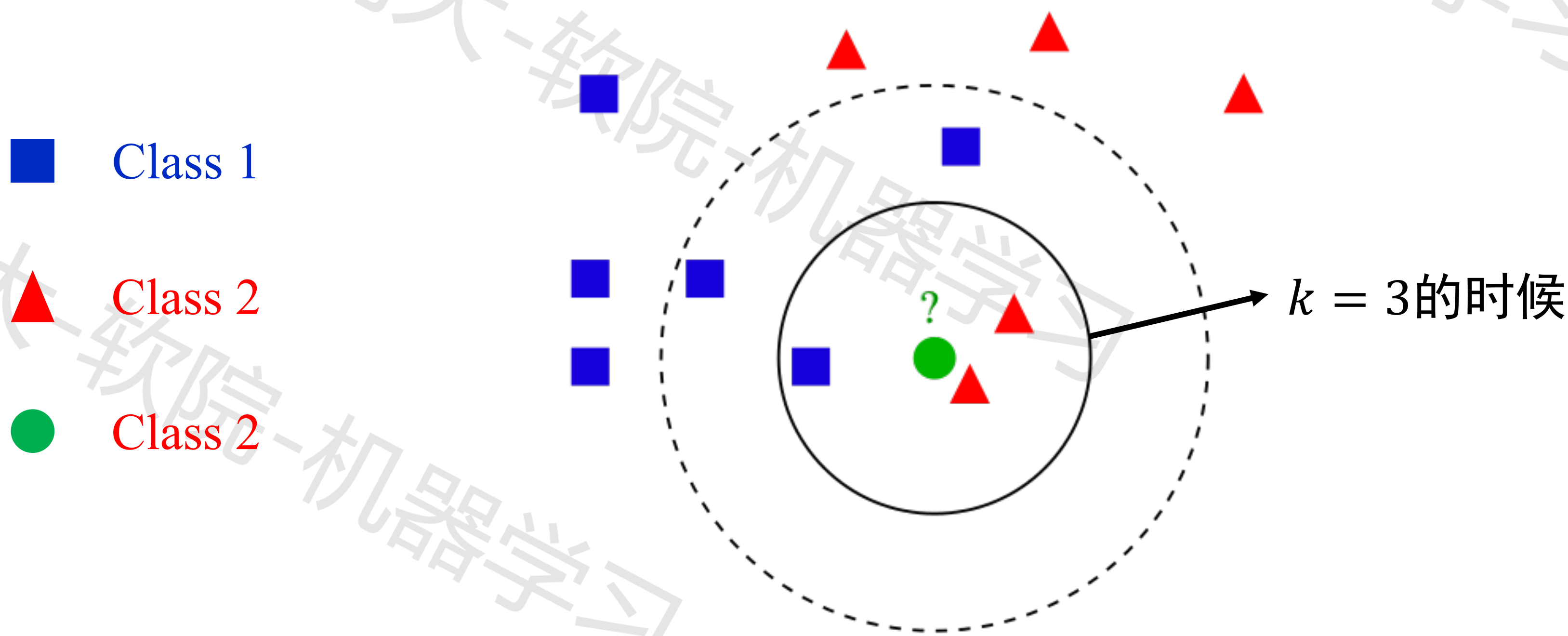
k-近邻分类器

- k的取值的影响



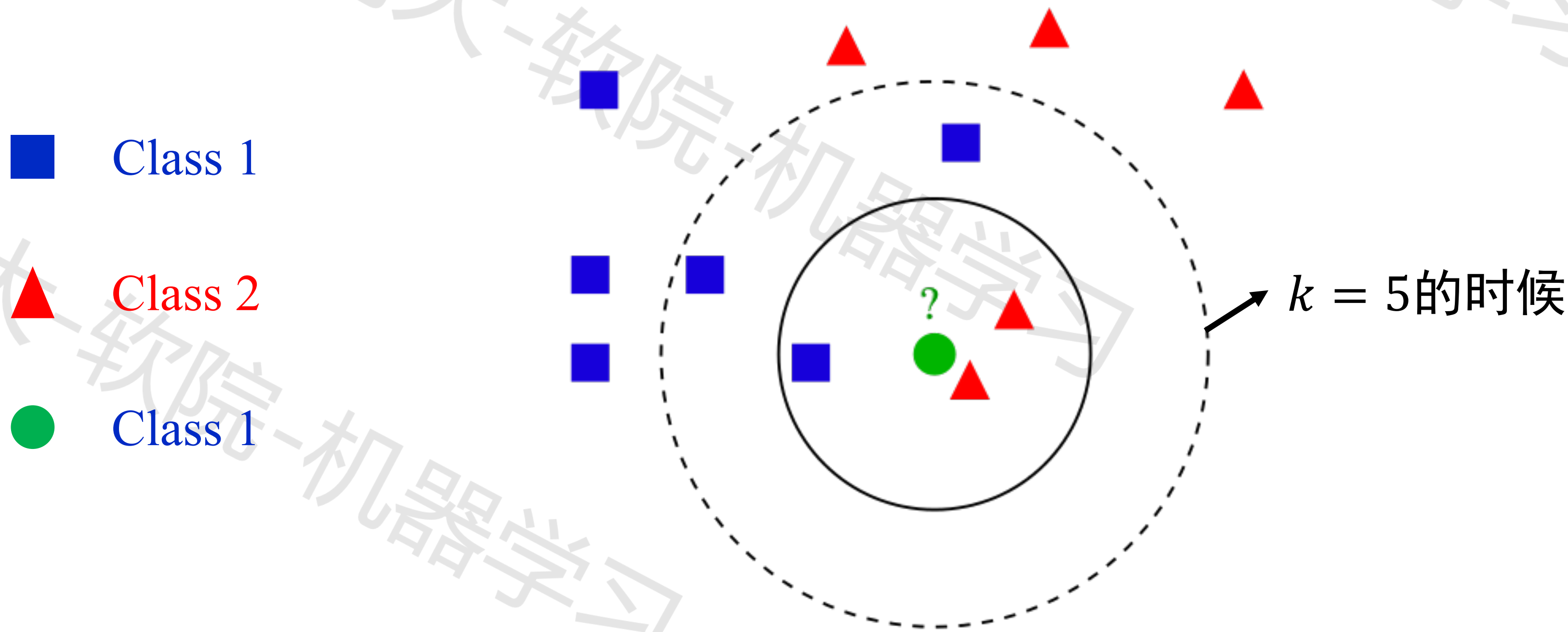
k-近邻分类器

- k的取值的影响



k-近邻分类器

- k的取值的影响



最近邻分类器

- 最近邻分类器 (*1*-Nearest Neighbor Classifier, *1*-NN)
  - ✓ 算法流程
    - 计算测试样本$\bar{\boldsymbol{x}}$和D_{train}中所有训练样本$\boldsymbol{x}_i$之间的距离$d(\bar{\boldsymbol{x}}, \boldsymbol{x}_i)$
    - 对所有距离值进行进行升序排列
    - 选择最近的那个训练样本$i^* = \underset{i}{\operatorname{argmin}}\, d(\bar{\boldsymbol{x}}, \boldsymbol{x}_i)$
    - 将$\boldsymbol{x}_{i^*}$的类别标签分配给$\bar{\boldsymbol{x}}$

最近邻分类器

- 泛化错误率

  ✓ 最近邻分类器的错误率（测试样本为$\boldsymbol{x}$，其最近邻为$\boldsymbol{z}$）：

$$P(err) = 1 - \sum_{c \in \mathcal{Y}} P(c|\boldsymbol{x})P(c|\boldsymbol{z})$$

  ✓ 假设样本i.i.d, 令$c^* = \underset{c \in \mathcal{Y}}{\mathrm{argmax}} P(c|\boldsymbol{x})$表示贝叶斯最优分类器的结果：

$$\begin{aligned} P(err) &\simeq 1 - \sum_{c \in \mathcal{Y}} P^2(c|\boldsymbol{x}) \\ &\le 1 - P^2(c^*|\boldsymbol{x}) \\ &= \big(1 + P(c^*|\boldsymbol{x})\big)\big(1 - P(c^*|\boldsymbol{x})\big) \\ &\le 2 \times \big(1 - P(c^*|\boldsymbol{x})\big) \end{aligned}$$

最近邻分类器

- 泛化错误率
  - ✓ 最近邻分类器的错误率（测试样本为$\boldsymbol{x}$，其最近邻为$\boldsymbol{z}$）：

$$P(err) = 1 - \sum_{c \in \mathcal{Y}} P(c|\boldsymbol{x})P(c|\boldsymbol{z})$$

  - ✓ 假设样本i.i.d, 令$c^* = \underset{c \in \mathcal{Y}}{\operatorname{argmax}} P(c|\boldsymbol{x})$表示贝叶斯最优分类器的结果：

$$P(err) \simeq 1 - \sum_{c \in \mathcal{Y}} P^2(c|\boldsymbol{x})$$

结论：最近邻分类器虽然简单，但它的泛化错误率不超过贝叶斯分类器的错误率两倍！

k-近邻回归

- k-近邻回归 (k-Nearest Neighbor Regression)
  - ✓ 算法流程
    - 计算测试样本$\overline{\boldsymbol{x}}$和$D_{train}$中所有训练样本$\boldsymbol{x}_i$之间的距离$d(\overline{\boldsymbol{x}}, \boldsymbol{x}_i)$
    - 对所有距离值（相似度值）进行进行升序（降序）排列
    - 选择k个最近（距离最小/相似度最大）的训练样本
    - 将距离值的倒数作为权重，然后将k个近邻的标签值加权平均，作为$\overline{\boldsymbol{x}}$的预测值

k-近邻回归

- 近邻平滑
  - ✓ 核平滑法（kernel smoother ）
    - 二次核（Epanechnikov quadratic kernel）

$$K_{E,\lambda}(x_0,x)=\begin{cases}0.75\left(1-\dfrac{(x_0-x)^2}{\lambda^2}\right) & if\ |x-x_0|<\lambda\\ 0 & otherwise\end{cases}$$

    - 次方核（tricube kernel）

$$K_{T,\lambda}(x_0,x)=\begin{cases}\left(1-\left|\dfrac{x-x_0}{\lambda}\right|^3\right)^3 & if\ |x-x_0|<\lambda\\ 0 & otherwise\end{cases}$$

    - 高斯核（Gaussian kernel）

$$K_{\mathrm{G},\lambda}(x_0,x)=\frac{1}{\lambda\sqrt{2\pi}}e^{-\frac{(x-x_0)^2}{2\lambda^2}}$$

k-近邻分类器

- 讨论
  - ✓ k-NN是典型的“懒惰学习”（lazy learning）
    - 训练阶段仅仅是把样本保存起来，训练时间开销为零，待收到测试样本后再进行处理
  - ✓ SVM、CNN等是“急切学习”（eager learning）
    - 在训练阶段就对样本进行学习处理的方法，这类方法尝试在训练期间构造一个通用的，与输入无关的目标函数

k-近邻分类器

- 讨论
  - ✓ 优点
    - ○ 精度高
    - ○ 对异常值不敏感
    - ○ 无数据输入假定
  - ✓ 缺点
    - ○ 计算复杂度高
    - ○ 空间复杂度高

k-近邻分类器

- 讨论
  - ✓ 时间复杂度
    - 假设$d(\boldsymbol{x}_i, \boldsymbol{x}_j)$是欧式距离，时间复杂度为$O(d)$
    - 训练阶段：0
    - 测试阶段：$O(nd + n\log k)$
  - ✓ 思考
    - 从n个数（距离）中选择k个最小的，时间复杂度是？
    - 空间复杂度？

k-近邻分类器

- 讨论-排序算法

排序算法	平均时间复杂度	最坏时间复杂度	最好时间复杂度	空间复杂度	稳定性
冒泡排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定
直接选择排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	不稳定
直接插入排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定
快速排序	O(nlogn)	$O(n^2)$	O(nlogn)	O(nlogn)	不稳定
堆排序	O(nlogn)	O(nlogn)	O(nlogn)	$O(1)$	不稳定
希尔排序	O(nlogn)	O(ns)	$O(n)$	$O(1)$	不稳定
归并排序	O(nlogn)	O(nlogn)	O(nlogn)	$O(n)$	稳定
计数排序	O(n+k)	O(n+k)	O(n+k)	O(n+k)	稳定
基数排序	O(N*M)	O(N*M)	O(N*M)	O(M)	稳定

k-近邻分类器

- 降低计算

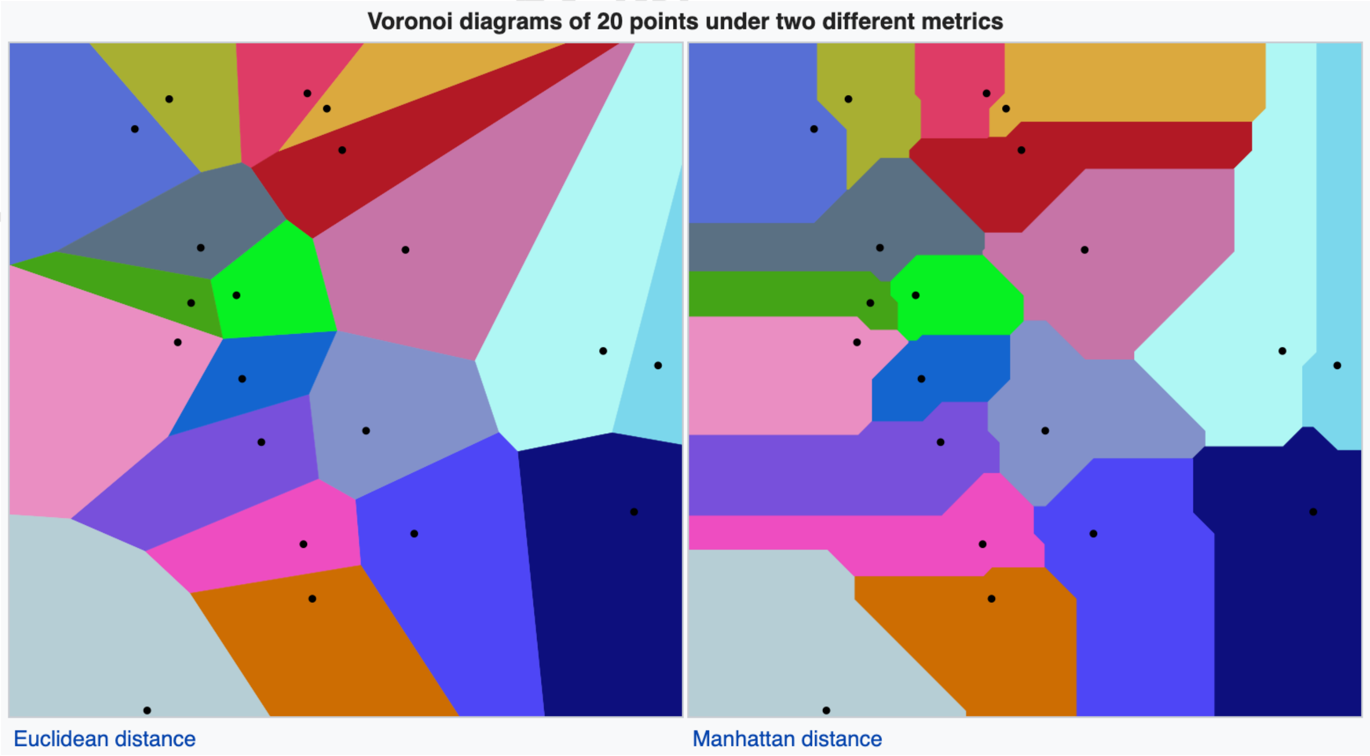
  ✓ 特征维度2-5: 维诺图 Voronoi diagrams

  ✓ 特征维度6-30: KD-Tree

  ✓ 高维特征：
    - 降维算法，例如PCA
    - 近似最近邻（approximate nearest neighbor, ANN）
    - 哈希（hashing）

降低近邻计算

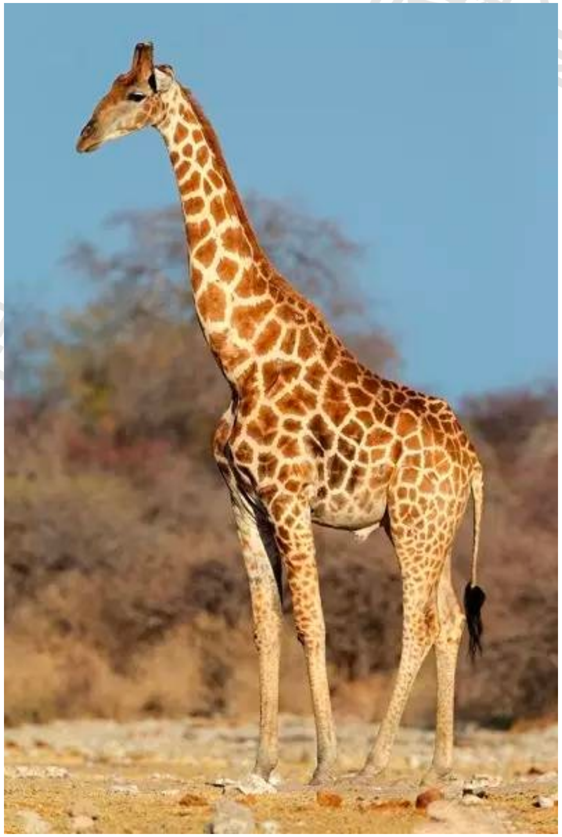
- 维诺图(Voronoi diagram)
  - ✓ 定义：根据一组给定的目标，将一个平面划分成靠近每一个目标的多个区块



降低近邻计算

- 维诺图(Voronoi diagram)

盐滩



长颈鹿

降低近邻计算

- 维诺图(Voronoi diagram)

  ✓ 维诺图由一系列维诺单元（Voronoi cells）组成

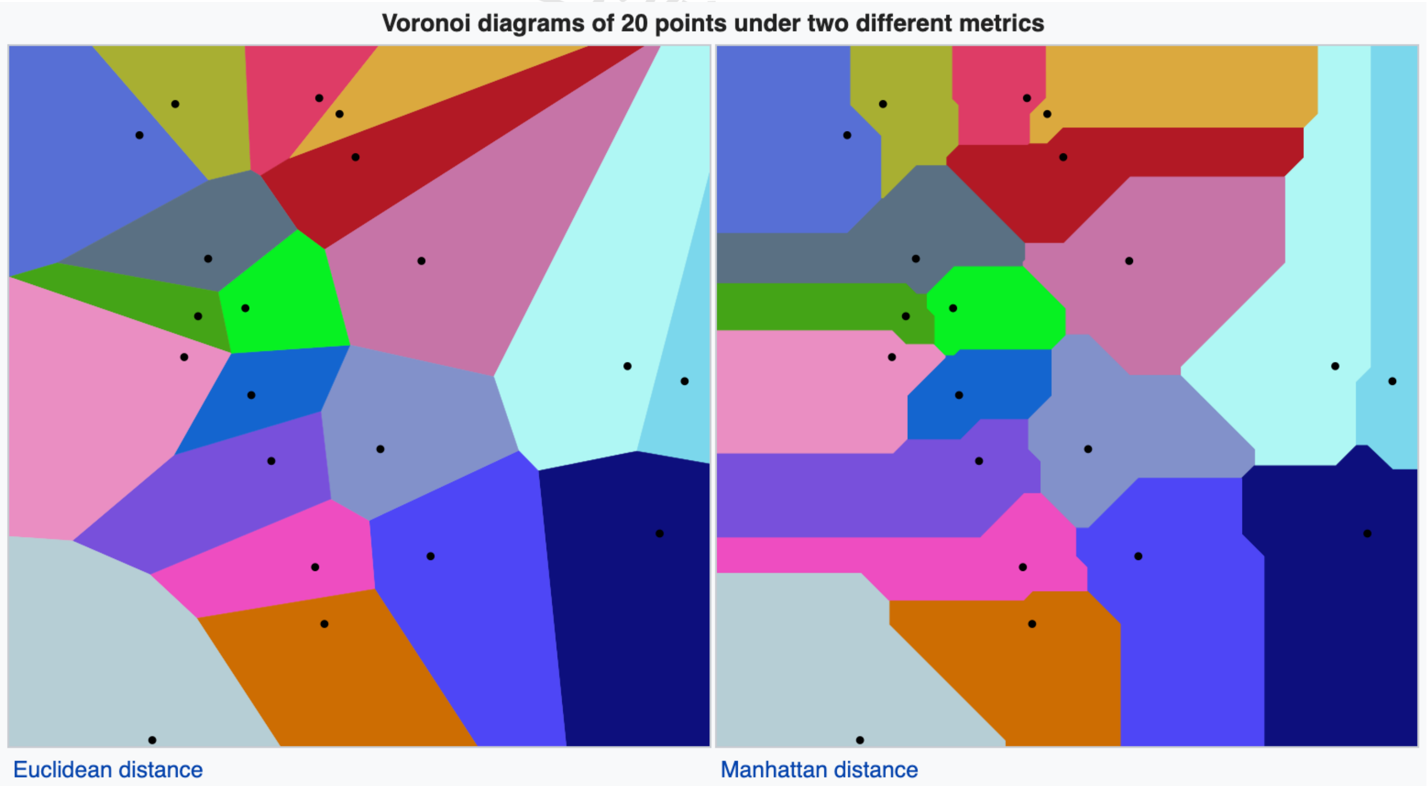
  假设X是一个点集，包含K个基点$(P_k)_{k\in K}$，那么维诺单元R_k定义为：

$$R_k = \{x \in X \mid d(x, P_k) \le d(x, P_j) \text{ for all } j \neq k\}$$

  ✓ 每个维诺单元永远都是一个凸多面体

降低近邻计算

- 维诺图(Voronoi diagram)
  - ✓ 在两种度量下，包含20个点的维诺图：



降低近邻计算

- 维诺图(Voronoi diagram)
  - ✓ 查询或测试：给定一个查询$q \in \mathbb{R}^d$，找到$P_k \in X$，使得$q \in R_k$
  - ✓ 时间复杂度
    - ○ 2维数据： $O(n\log n)$ 用来计算维诺图和梯形图；查询（测试）时间为$O(\log n)$，其中使用梯形图进行点的定位
    - ○ d维数据：得使用二叉空间分割树（binary space partition tree , BSP tree）进行点的定位，但是时间估计比较难，难以量化
  - ✓ 适用范围：1-NN
  - ✓ 适合特征维度：2-3维，可能4维或5维

降低近邻计算

- KD树（KD-Tree）

✓ KD树是一种对K维空间中的实例点进行存储以便对其进行快速检索的树形数据结构。

✓ KD树是二叉树，表示对K维空间的一个划分（partition)。构造KD树相当于不断地用垂直于坐标轴的超平面将K维空间切分，构成一系列的K维超矩形区域。KD树的每个结点对应于一个K维超矩形区域。

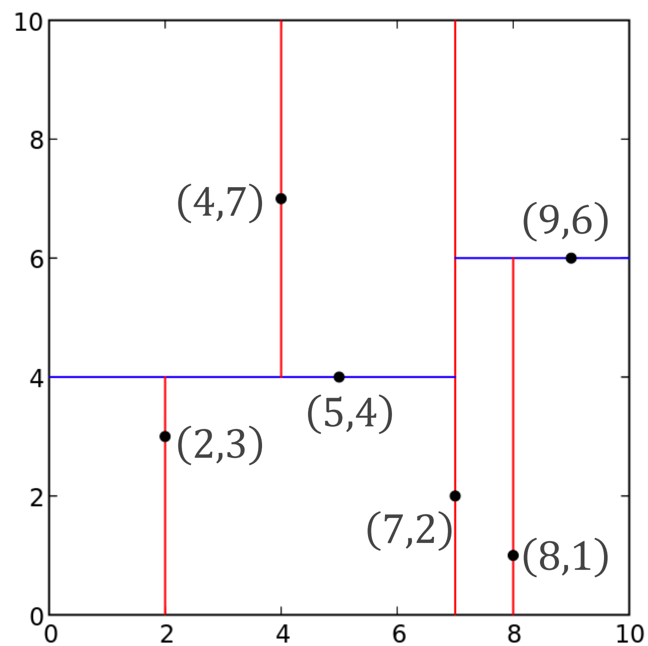
降低近邻计算

- KD树构造
  - ✓ 构造流程:
    - ○ 确定split域。计算每个特征维度的方差，方差最大的维度即为split域的值
    - ○ 确定Node-data域。数据集点集按其第split域的值排序，中位数的数据点即被选为Node-data
    - ○ 对剩下的数据点进行划分，确定左右子空间
    - ○ 递归。在每个子空间继续进行空间划分，直到空间中只包含一个数据点

降低近邻计算

- KD树构造

假设有6个二维数据点{(2,3), (5,4), (9,6), (4,7), (8,1), (7,2)}，数据点位于二维空间内，如下图黑点所示。

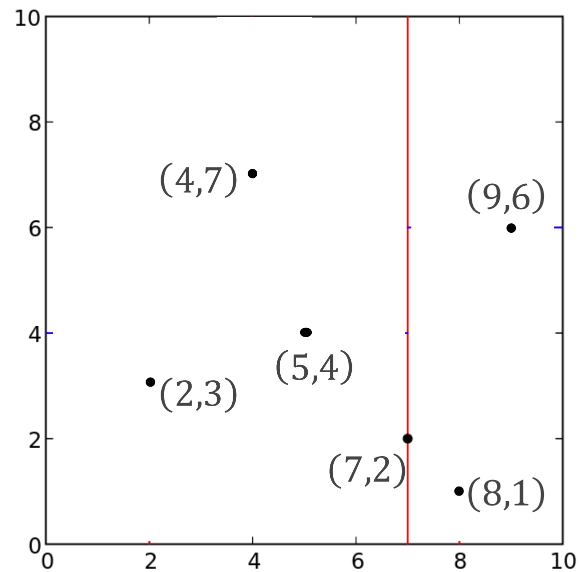


二维数据KD树空间划分示意图

降低近邻计算

- KD树构造

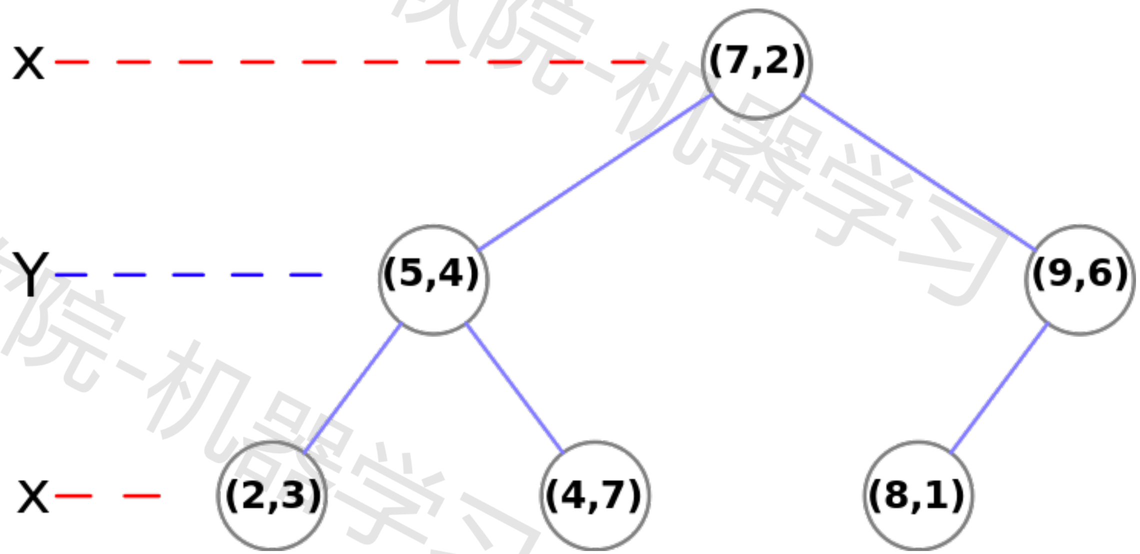
1. 确定split域的值；分别计算x，y方向上的方差，可知x方向方差更大，则split=0（将x，y方向分别编码为0和1）
2. 确定Node-data的值；根据 x 轴方向的值 2,5,9,4,8,7 得到中位数7，则选定$\text{Node}-\text{data}=(7,2)$对空间进行切分，即$x=7$
3. 递归。。。



$x=7$将空间划分成左右两个子空间

降低近邻计算

- KD树构造
  - ✓ 得到KD树：



降低近邻计算

- KD树搜索

1. 二叉搜索：从根节点开始，以递归的方式从树的顶端向下移动

2. 当到达一个叶子节点，即得到最邻近的近似点，判断其是否为最优，并保存为“当前最优”

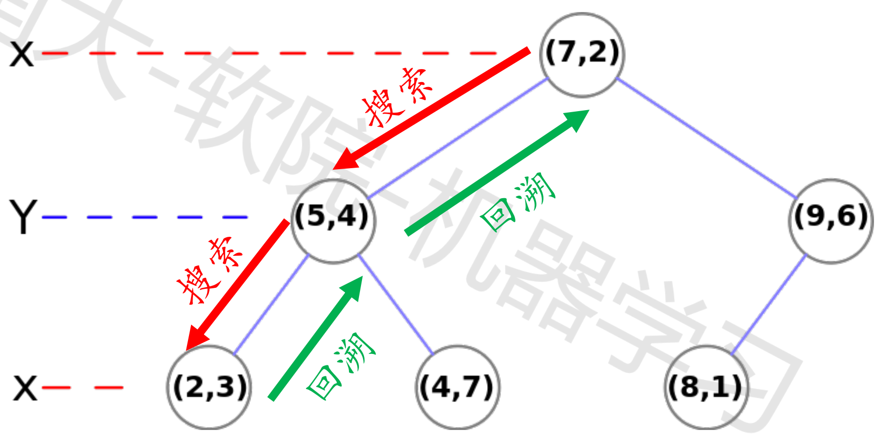
3. 回溯：对整棵树进行递归，并对每个节点执行以下操作
   - 如果当前节点比“当前最优”更近，替换为新的“当前最优”
   - 判断分割平面的另一侧是否存在比“当前最优”更优的点。构造一个超球面，球心为查询点，半径为与当前最优的距离
     - 如果超球面跟超平面相交，则有可能存在更优的点；按照相同的搜索过程，从当前节点向下移动到树的另一个分支以寻找更近的点
     - 如果超球面跟超平面不相交，则沿着树继续往上走，当前节点的另一个分支则被排除

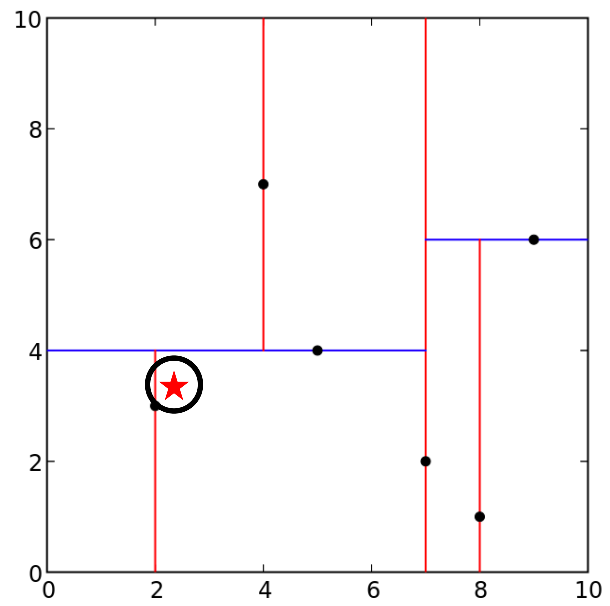
4. 当算法为根节点完成整个过程时，算法结束

降低近邻计算

- KD树搜索

1. 查找查询点(2.1, 3.1)
2. 二叉搜索，找到叶子节点(2, 3)，并且当前最优欧式距离为0.1414
3. 回溯，构造一个超球面（圆）
4. 分别对节点(5, 4)和(7, 2)进行回溯分析
5. 结束整个搜索，返回最近邻(2, 3)





查询点为(2.1, 3.1)，图中星号所示

降低近邻计算

- KD树搜索
  - ✓ 时间复杂度
    - ○ $O(n\log^2 n)$，如果选用复杂度为$O(n\log n)$的排序算法，如快速排序、堆排序、归并排序来找到中位数
    - ○ 为$O(n\log n)$，如果选用复杂度为$O(\mathrm{n})$ median of medians 算法来找到中位数
    - ○ 。。。
    - ○ 寻找最近邻的时间复杂度为： $O(\log n)$

降低近邻计算

- 降维（Dimension reduction）
  - ✓ **核心思想**：通过某种数学变换将原始高维属性空间转变为一个低维子空间，来缓解维数灾难问题。
    - 多维缩放方法（Multiple Dimensional Scaling, MDS）
    - 主成分分析（Principal Component Analysis, PCA）
    - 局部线性潜入（Locally Linear Embedding, LLE）
    - ISOMAP
    - ….
  - ✓ **推荐阅读**：降维算法相关的书籍和论文

降低近邻计算

- 近似最近邻（approximate nearest neighbor, ANN）
  - ✓ **核心思想**：搜索可能是近邻的数据项而不再只局限于返回最可能的数据项，在牺牲可接受范围内的精度的情况下提高检索效率
    - ○ 不要求一定是距离最短的k个
    - ○ 如第k个最近邻，其距离是d_k，则ANN要求其选取的所有k个样例的距离$\hat{d} \leq (1+\varepsilon)d_k$
    - ○ 可以将k-NN搜索速度提高几个数量级
  - ✓ **推荐阅读**：FLANN：https://github.com/mariusmuja/flann
    - ○ ANN相关的软件和论文

降低近邻计算

- 哈希（Hashing）

✓ **核心思想**：利用哈希函数把任意长度的输入映射为固定长度的输出

  - Hash函数f_i: 将$\mathbb{R}^d$分为两部分，分别用$f_i = 0, 1$表示
  - 设计m个hash函数$f_1, \dots, f_m$，每个样本x表示为 m个bit
  - $m \ll d$，计算和存储大幅简化，需要设计好的hash

✓ **推荐阅读**：

  - LSH：https://en.wikipedia.org/wiki/Locality-sensitive_hashing
  - Hash function: https://en.wikipedia.org/wiki/Hash_function

扩展阅读

- Probabilistic *k*-NN
  - *k*-NN算法的一个缺点是它不是建立在任何概率框架上
  - 无法得到关于类别的后验概率
  - 没办法概率化地推断近邻的个数，以及度量的参数
  - 可以通过定义似然函数来构造概率化的*k*-NN

✓ **推荐阅读：**

- https://www.cc.gatech.edu/~afb/classes/CS7616-Spring2014/slides/CS7616-13a-PKNN.pdf

扩展阅读

- Boiman O, Shechtman E, Irani M. **In defense of nearest-neighbor based image classification**[C]. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2008: 1-8.

- Li W, Wang L, Xu J, et al. **Revisiting local descriptor based image-to-class measure for few-shot learning**[C]. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2019: 7260-7268.

谢 谢！