

# 机器学习（本科生公选课）GEC6531

## 第5节 k近邻（kNN）k Nearest Neighbors

计算机科学与技术学院

张瑞 教授

邮箱: [ruizhang6@hust.edu.cn](mailto:ruizhang6@hust.edu.cn)

# 签到 & 思考

## ■ 微助教签到（学校要求）

1. 加入课堂：微信扫码或者通过微助教公众号



二维码有效期至: 2024-11-16

课堂名称: GEC6531 机器学习 (公选课)

课堂编号: OA628

---

1、扫码关注公众号: 微助教服务号。

2、点击系统通知: “[点击此处加入【GEC6531 机器学习 \(公选课\)】课堂](#)”, 填写学生资料加入课堂。

---

\*如未成功收到系统通知, 请点击公众号下方“学生” - “全部(A)” - “加入课堂” --- “输入课堂编号”手动加入课堂

2. 微信扫码签到

如何进行多分类, 也就是多于2类的分类问题?

# 今天的目录

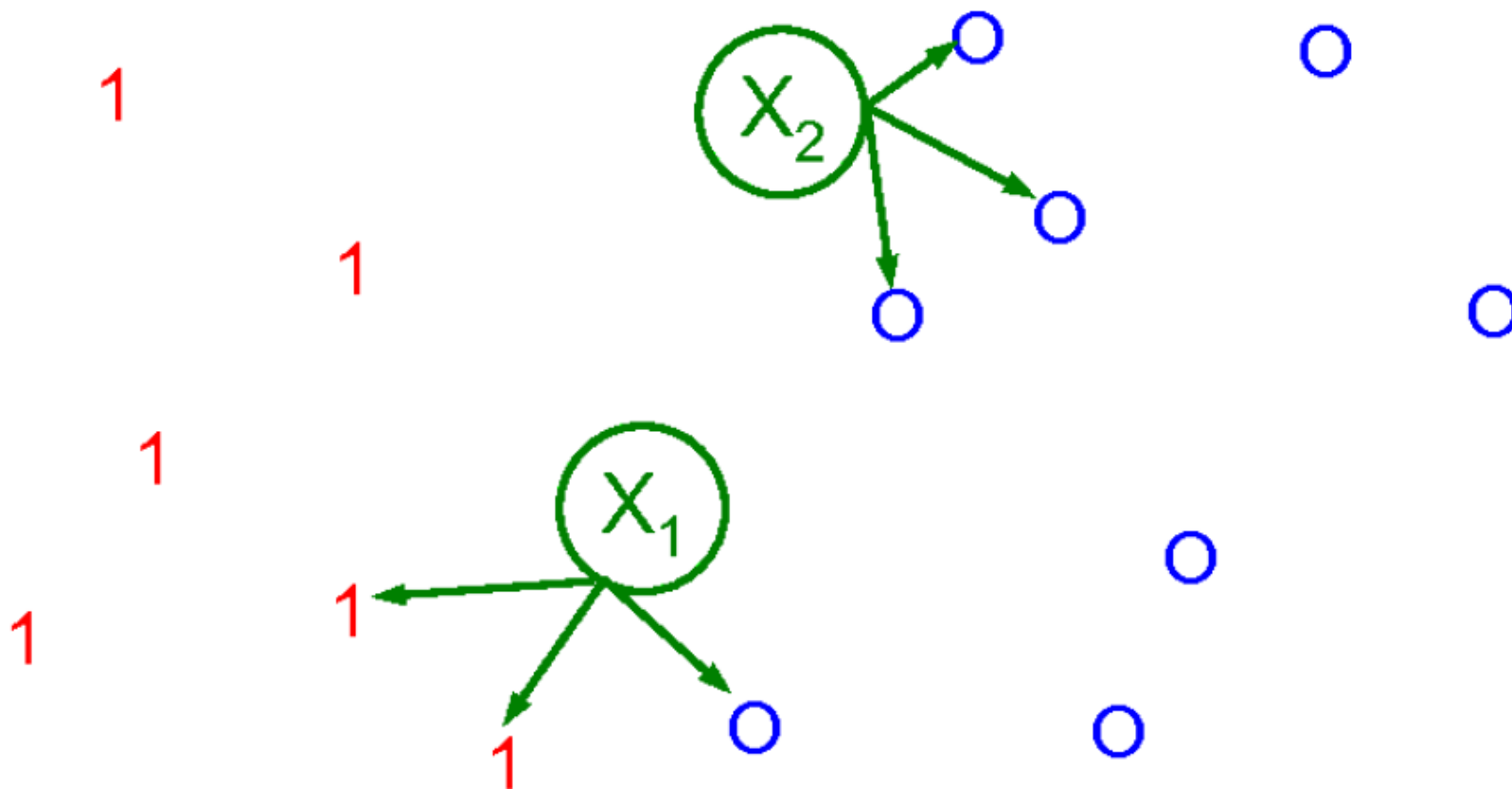
- **k 近邻算法 (KNN)**
  - k-NN 的形式化定义
  - kNN 实例
- **参数选择**
  - 距离函数
  - k 值的选择
- **特殊的 k 近邻分类器**
  - 1-最近邻分类器
  - 加权的最近邻分类器
- **维度灾难**
  - 点对之间的距离
  - 到超平面的距离
  - 低维结构的数据
- **如何解决维数灾难问题**
  - 降维
  - 数据约简

# 今天的目录

- **k 近邻算法 (KNN)**
  - k-NN 的形式化定义
  - kNN 实例
- **参数选择**
  - 距离函数
  - k 值的选择
- **特殊的 k 近邻分类器**
  - 1-最近邻分类器
  - 加权的最近邻分类器
- **维度灾难**
  - 点对之间的距离
  - 到超平面的距离
  - 低维结构的数据
- **如何解决维数灾难问题**
  - 降维
  - 数据约简

# kNN实例, $k = 3$

《战国策·齐策三》：“物以类聚，人以群分”



# k-NN 的形式化定义

在模式识别中, k-近邻算法(k-NN) 是一种用于分类和回归的非参数化方法。

假设: 相似的输入有相似的输出

分类规则: 对于一个测试样例即输入 $\mathbf{x}$ , 在它的 $k$  个最相似的训练输入中选择出现最多次数的标签作为输出

回归规则: 输出是对象的一个属性值。这个值是 $k$  个最近邻值的平均值。

# kNN的形式化定义

## 形式化定义

- 测试点:  $x$
- 将  $x$  的  $k$  近邻的集合表示为  $S_x$ 。  $S_x$  正式定义为  $S_x \subseteq D$ , s.t.  $|S_x| = k$  和  $\forall (x', y') \in D/S_x$ ,

$$\text{dist}(x, x') \geq \max_{(x'', y'') \in S_x} \text{dist}(x, x''),$$

(即  $D$  中不在  $S_x$  中的每个点到  $x$  的距离至少与  $S_x$  中的最远点一样远)。  
然后我们可以将分类器  $h()$  定义为一个函数, 返回  $S_x$  中最常见的标签:

$$h(x) = \text{mode}(\{y'' : (x'', y'') \in S_x\}),$$

其中  $\text{mode}(\cdot)$  表示选择出现频率最高的标签。

# kNN二分类示例

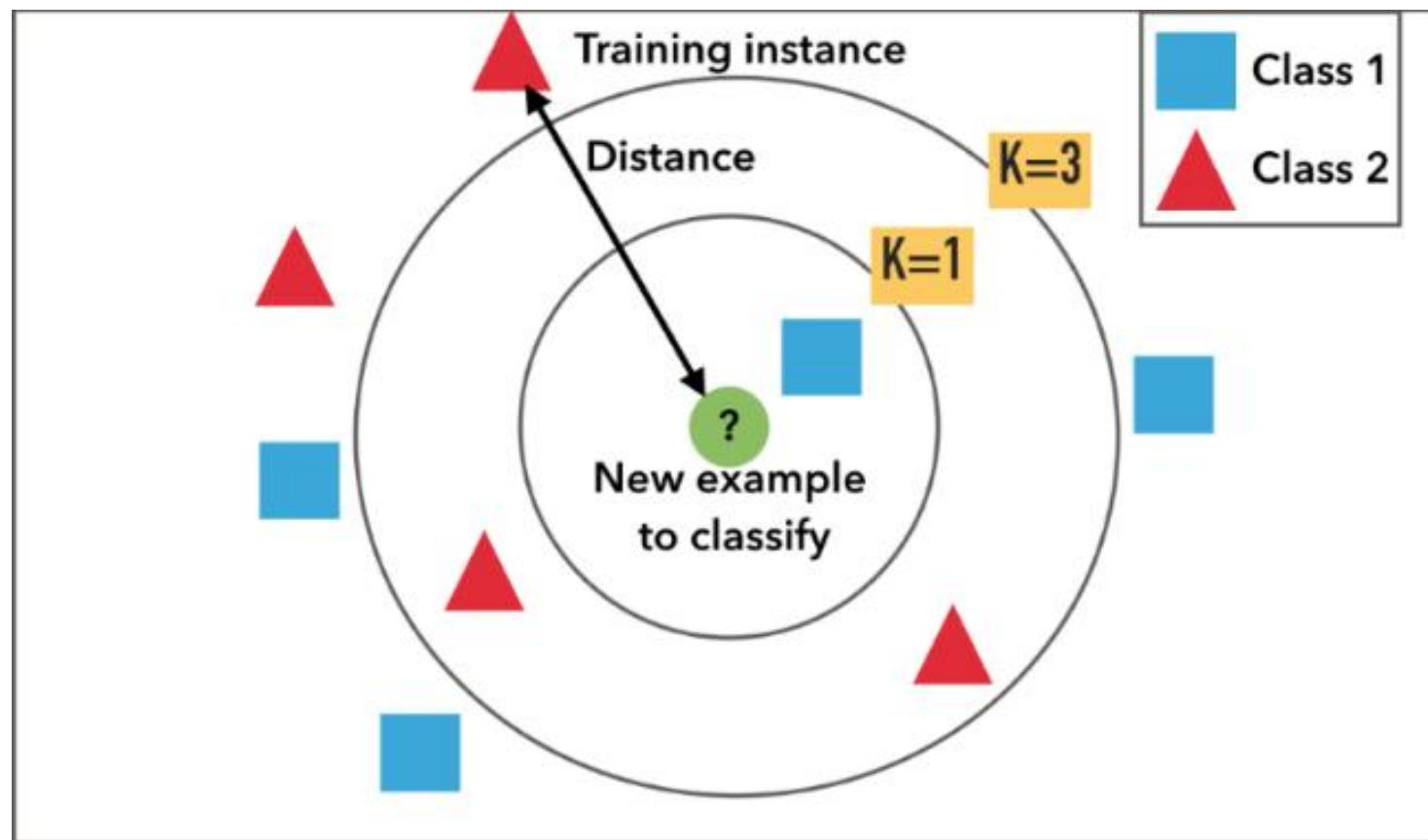


图: k-NN 分类示例。测试样本(内圆) 应该分为第一类蓝色正方形或第二类红色三角形。如果 $k = 3$ (外圆), 它被分配到第二类, 因为有2 个三角形和只有1 个正方形在内圆。如果 $k = 5$ , 它将被分配给第一类(3 个正方形vs. 2 个三角形)。



# 今天的目录

- **k 近邻算法 (KNN)**
  - k-NN 的形式化定义
  - kNN 实例
- **参数选择**
  - 距离函数
  - k 值的选择
- **特殊的 k 近邻分类器**
  - 1-最近邻分类器
  - 加权的最近邻分类器
- **维度灾难**
  - 点对之间的距离
  - 到超平面的距离
  - 低维结构的数据
- **如何解决维数灾难问题**
  - 降维
  - 数据约简

## 参数选择：距离函数

k-近邻分类器基本上依赖于距离度量。该指标反映的标签相似性越好，分类的效果就越好。最常见的选择是 **Minkowski 距离**：

$$\text{dist}(\mathbf{x}, \mathbf{z}) = \left( \sum_{r=1}^d |x_r - z_r|^p \right)^{1/p}.$$

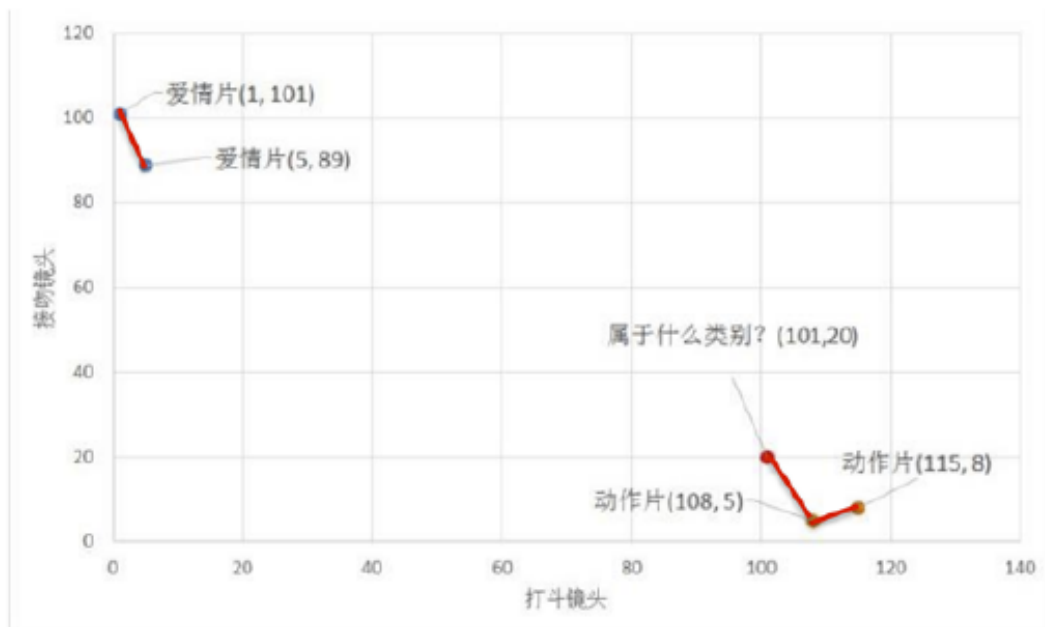
### 测试

这个距离定义是相当通用的，并且包含许多众所周知的距离作为特殊情况。你能确认出以下候选距离吗？

- $p = 1$
- $p = 2$
- $p \rightarrow \infty$

## 欧氏距离(Euclidean distance)

### 电影分类

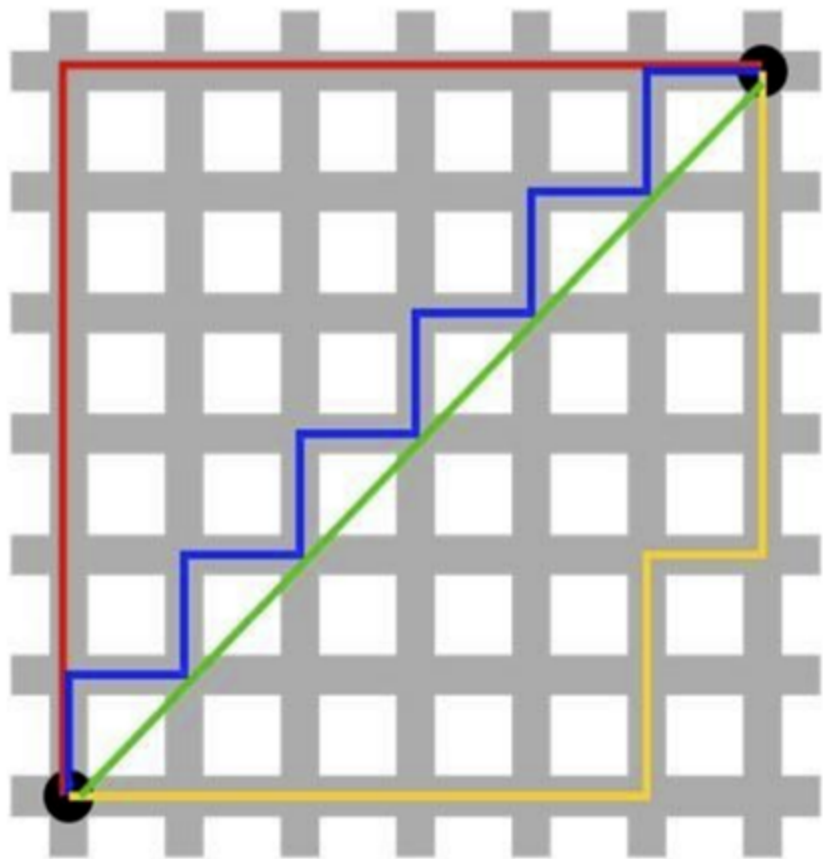


$$d(x, y) = \sqrt{\sum_i (x_i - y_i)^2}$$

欧几里得度量 (Euclidean Metric) (也称欧氏距离) 是一个通常采用的距离定义, 指在  $m$  维空间中两个点之间的真实距离, 或者向量的自然长度 (即该点到原点的距离)。在二维和三维空间中的欧氏距离就是两点之间的实际距离。

$$p = 1$$

## 曼哈顿距离(Manhattan distance)



$$d(x, y) = \sum_i |x_i - y_i|$$

想象你在城市道路里，要从一个十字路口开车到另外一个十字路口，驾驶距离是两点间的直线距离吗？显然不是，除非你能穿越大楼。实际驾驶距离就是这个“曼哈顿距离”。而这也是曼哈顿距离名称的来源，曼哈顿距离也称为城市街区距离(City Block distance)。


$$p = \infty$$

## 切比雪夫距离(Chebyshev distance)

$$d(x, y) = \max_i |x_i - y_i|$$

二个点之间的距离定义是其各坐标数值差绝对值的最大值。

国际象棋棋盘上二个位置间的切比雪夫距离是指王要从一个位子移至另一个位子需要走的步数。由于王可以往斜前或斜后方向移动一格，因此可以较有效率的到达目的的格子。上图是棋盘上所有位置距f6位置的切比雪夫距离。

	a	b	c	d	e	f	g	h	
8	5	4	3	2	2	2	2	2	8
7	5	4	3	2	1	1	1	2	7
6	5	4	3	2	1		1	2	6
5	5	4	3	2	1	1	1	2	5
4	5	4	3	2	2	2	2	2	4
3	5	4	3	3	3	3	3	3	3
2	5	4	4	4	4	4	4	4	2
1	5	5	5	5	5	5	5	5	1
	a	b	c	d	e	f	g	h	

$$p = 1, 2, \dots, \infty$$

## 闵可夫斯基距离(Minkowski distance)

$p$ 取1或2时的闵氏距离是最为常用的

$p = 2$ 即为欧氏距离,

$p = 1$ 时则为曼哈顿距离。

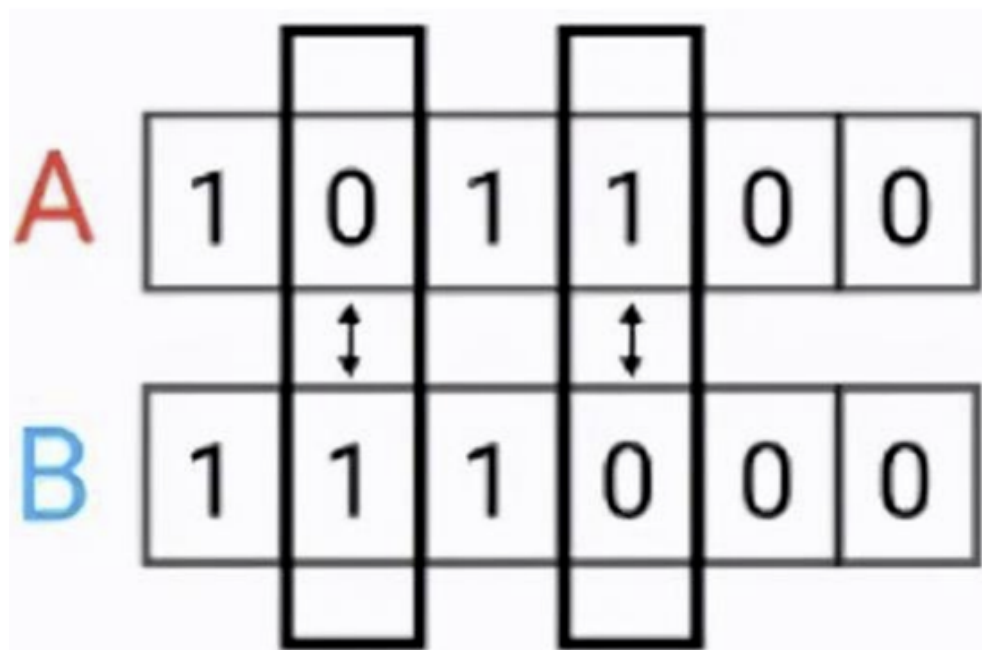
当 $p$ 取无穷时的极限情况下, 可以得到切比雪夫距离

$$d(x, y) = \left( \sum_i |x_i - y_i|^p \right)^{\frac{1}{p}}$$



# 其他距离

## 汉明距离(Hamming distance)

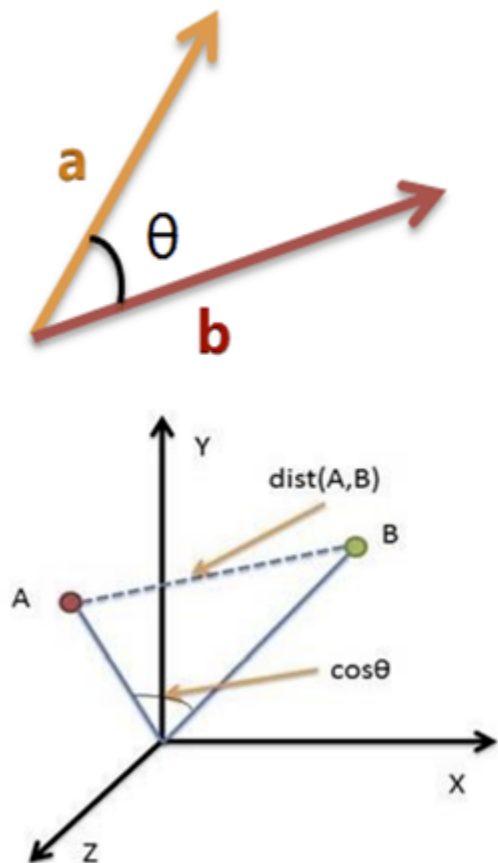


$$d(x, y) = \frac{1}{N} \sum_i 1_{x_i \neq y_i}$$

汉明距离是使用在数据传输差错控制编码里面的，汉明距离是一个概念，它表示两个（相同长度）字对应位不同的数量，我们以表示两个字之间的汉明距离。对两个字符串进行异或运算，并统计结果为1的个数，那么这个数就是汉明距离。

# 其他距离

## 余弦相似度



两个向量有相同的指向时，余弦相似度的值为1；两个向量夹角为 $90^\circ$ 时，余弦相似度的值为0；两个向量指向完全相反的方向时，余弦相似度的值为-1。

假定 $A$ 和 $B$ 是两个 $n$ 维向量， $A$ 是 $[A_1, A_2, \dots, A_n]$ ， $B$ 是 $[B_1, B_2, \dots, B_n]$ ，则 $A$ 和 $B$ 的夹角的余弦等于：

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$



# K 值的影响

## 如何选择最合适的 K 值

更小的 K 值:

- 减少学习的近似误差 [训练数据]
- 放大学习的估计误差 [测试数据]
- 更复杂的模型, 容易过拟合

更大的 K 值:

- 放大学习的近似误差 [训练数据]
- 减少学习的估计误差 [测试数据]
- 更简单的模型

# 最好的K 值

## 如何选择一个最合适的 K 值

k 的最佳选择取决于数据：

一般来说，较大的 k 值会降低噪声对分类的影响，但会使类之间的边界不那么明显。  
一个好的 k 可通过各种启发式方法进行选择（参见超参数优化）。  
在二分类问题中，选择 k 为奇数是有帮助的，以避免出现平局。

常用的选择“经验最优”k 的方法包括：

- 自助法 (Bootstrap method)
- 交叉验证法 (cross validation)
- 贝叶斯方法 (Bayes method)

# 今天的目录

- **k 近邻算法 (KNN)**
  - k-NN 的形式化定义
  - kNN 实例
- **参数选择**
  - 距离函数
  - k 值的选择
- **特殊的 k 近邻分类器**
  - 1-最近邻分类器
  - 加权的最近邻分类器
- **维度灾难**
  - 点对之间的距离
  - 到超平面的距离
  - 低维结构的数据
- **如何解决维数灾难问题**
  - 降维
  - 数据约简

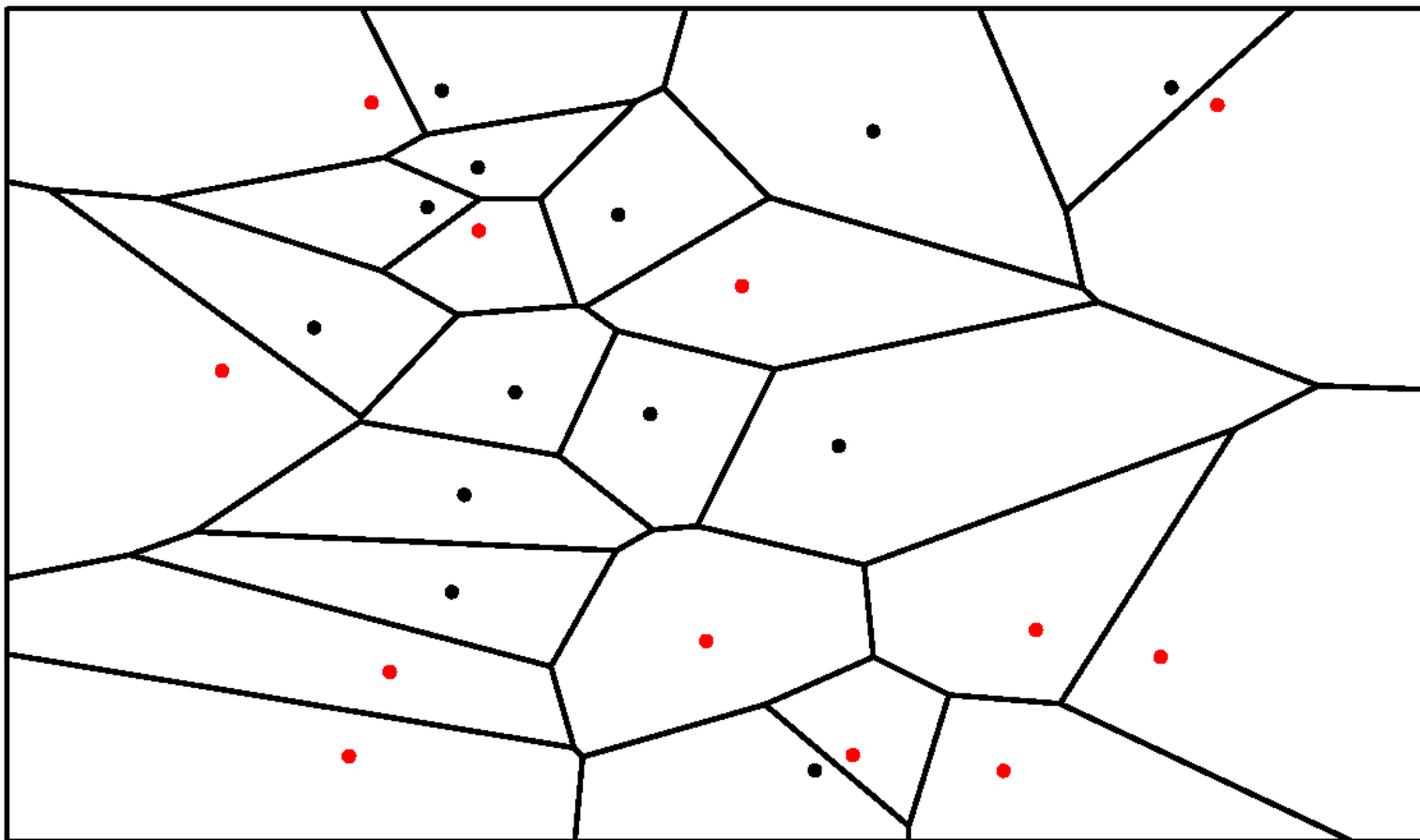
# 特殊的 $k$ 近邻分类器：1-NN 分类器

## 1-NN 分类器

最直观的最近邻分类器是一个 1 近邻分类器 ( $k=1$ )，它将一个点  $x$  分配给特征空间中最近邻点对应的类。

性能保证：当训练数据集的大小接近无穷大时，1 近邻分类器可保证错误率不超过贝叶斯错误率（给定数据分布的最小可实现错误率）的两倍。

# kNN 示例, $k = 1$



Voronoi Diagram

# 贝叶斯最优分类器

举例：假设（但是大多数时候几乎是不会发生这个情况的）你知道  $P(y|x)$ ，然后你可以简单的预测最可能的标签。

$$y^* = h_{opt}(x) = \underset{y}{\operatorname{argmax}} P(y|x)$$

虽然贝叶斯优化器可以得到最优值，但是它有时候也会出错。如果一个样本点并没有它最可能的标签，那么它就会出错。我们可以准确计算这个情况发生的可能性（也就是准确的错误率）：

$$\epsilon_{BayesOpt} = 1 - P(h_{opt}(x)|x) = 1 - P(y^*|x)$$

假设对于一个样本邮件  $x$  要么被分类成为普通邮件 (+1)，要么分类成为一个垃圾邮件 (-1)。对于相同的邮件  $x$  的条件概率为：

$$P(+1|x) = 0.8$$

$$P(-1|x) = 0.2$$

在这种情况下，贝叶斯最优分类器将会预测标签为  $y^* = +1$ ，作为最有可能的标签，因此这个错误率为：

$$\epsilon_{BayesOpt} = 0.2$$

# 1-NN 的收敛性

[Cover, Hart, 1967]

当  $n \rightarrow \infty$ , 1-NN 的错误率不会超过贝叶斯最优分类器错误率的两倍。(类似的保证适用于  $k > 1$ 。)

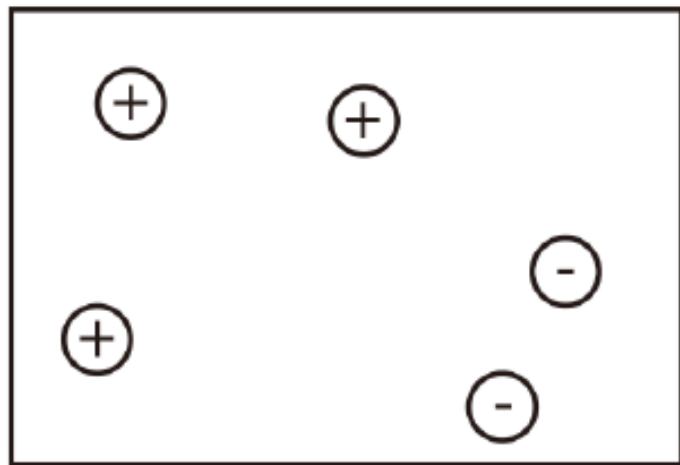


图:  $n$  很小时

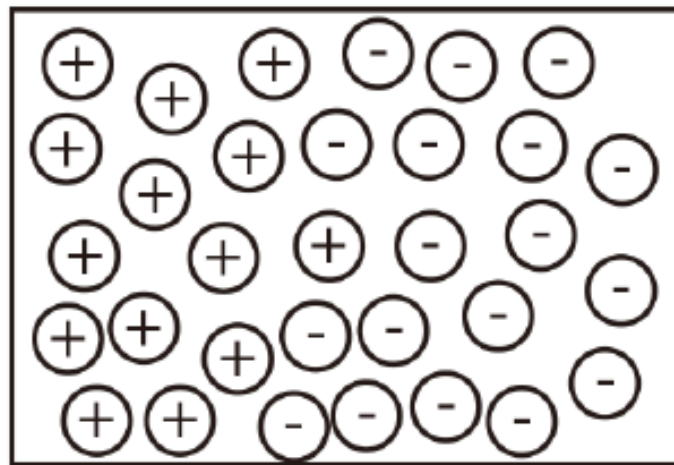


图:  $n$  很大时

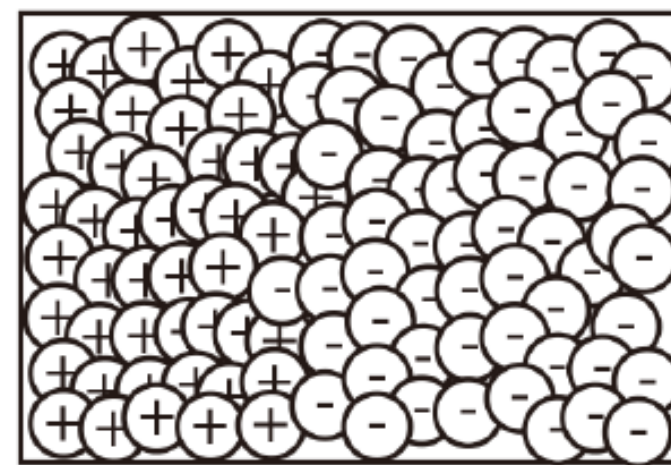
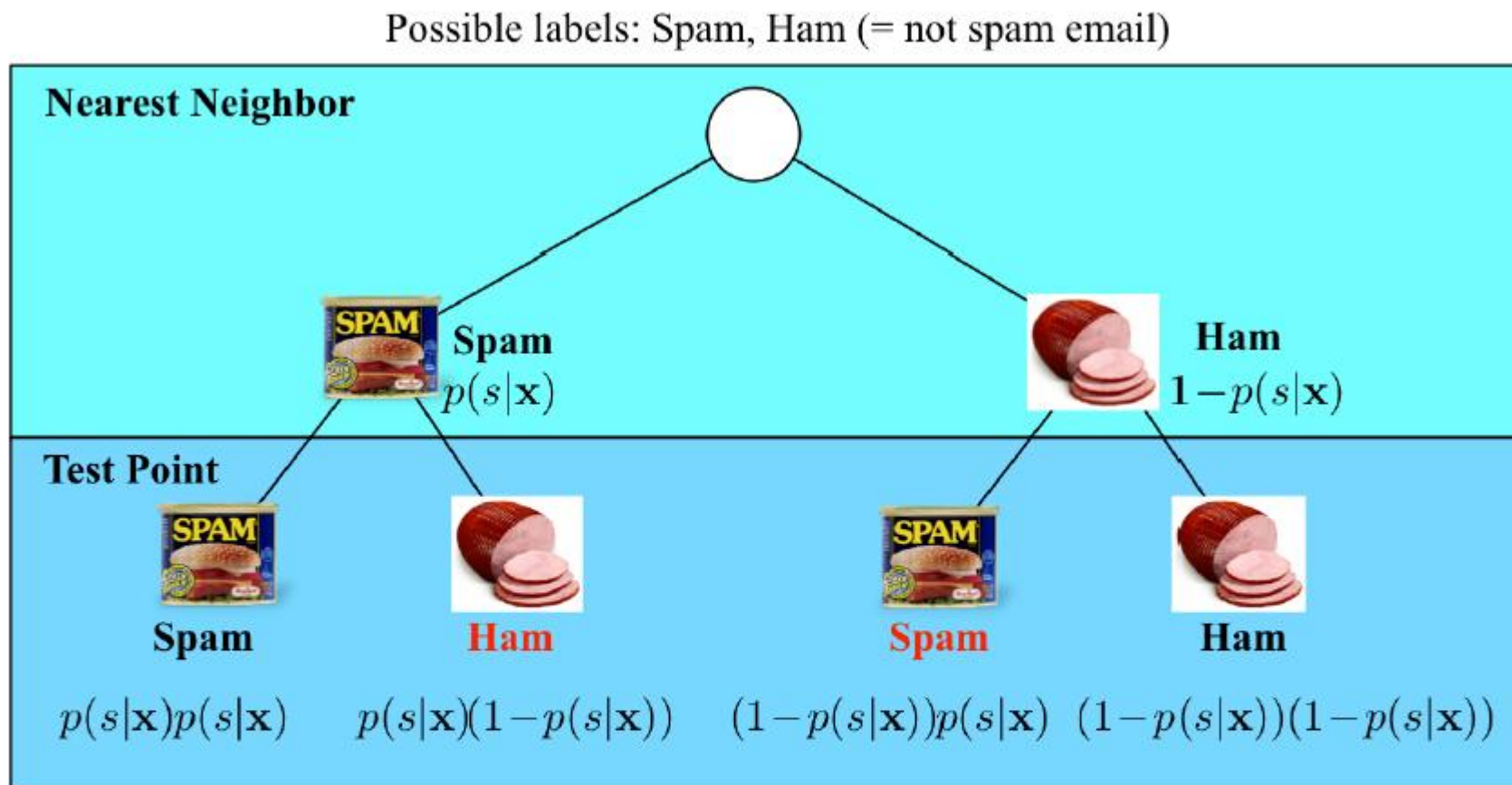


图:  $n \rightarrow \infty$

[Cover, Hart, 1967] Cover, Thomas, and, Hart, Peter. Nearest neighbor pattern classification. IEEE Transactions on Information Theory, 1967, 13(1): 21-27.



# 1-NN 的收敛性证明



图：在极限情况下，测试点与其最近的邻居是重合的。有两种情况会发生错误分类，即测试点和它最近的邻居有不同的标签。此事件发生的概率是两个红色事件的概率之和：

$$(1-p(s|x))p(s|x) + p(s|x)(1-p(s|x)) = 2p(s|x)(1-p(s|x)) .$$



# 特殊的 k 近邻分类器：加权的最近邻分类器

k 近邻分类器可以被看作是给 k 个最近邻分配权重  $1/k$ ，而其他所有实例的权重为 0。

这可以推广到加权的最近邻分类器。

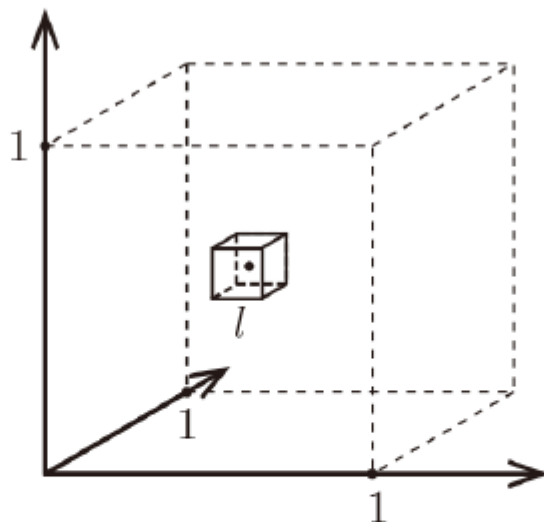
也就是说，第  $i$  个最近的邻居被赋予权重  $w_{ni}$ ，其  $\sum_{i=1}^n w_{ni} = 1$ 。

# 今天的目录

- **k 近邻算法 (KNN)**
  - k-NN 的形式化定义
  - kNN 实例
- **参数选择**
  - 距离函数
  - k 值的选择
- **特殊的 k 近邻分类器**
  - 1-最近邻分类器
  - 加权的最近邻分类器
- **维度灾难**
  - 点对之间的距离
  - 到超平面的距离
  - 低维结构的数据
- **如何解决维数灾难问题**
  - 降维
  - 数据约简

# 点对之间的距离

形式上，想象一个单位立方体  $[0, 1]^d$ 。所有训练数据都是从这个立方体中均匀采样，即  $\forall i, x_i \in [0, 1]^d$ ，我们考虑这样如下测试点的  $k = 10$  最近的邻居。



设  $\ell$  为包含测试点的所有  $k$ -最近邻的最小超立方体的边长。

Q:  $\ell$  约等于多少？

A:  $\ell^d \approx \frac{k}{n}$ ,  $\ell \approx \left(\frac{k}{n}\right)^{1/d}$ .

# 点对之间的距离

对于  $k = 10, n = 1000$ :

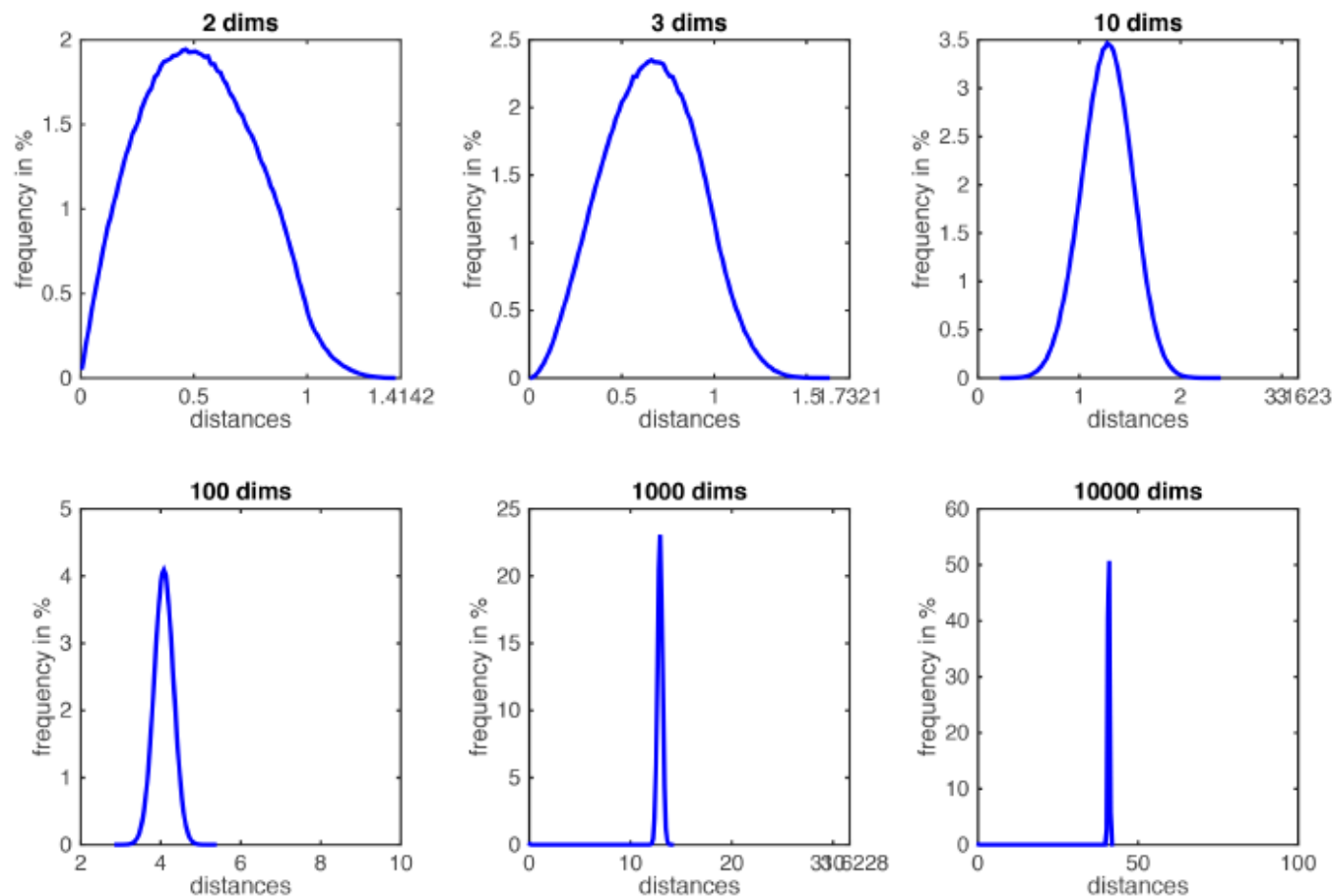
$d$	$\ell$
2	0.1
10	0.63
100	0.955
1000	0.9954

因此, 当  $d \gg 0$  时, 几乎需要整个空间来找到10-NN。

这打破了  $k$ -NN 假设, 因为  $k$  个近邻并不比训练集中的其他任何数据点更接近(因此更相似)。

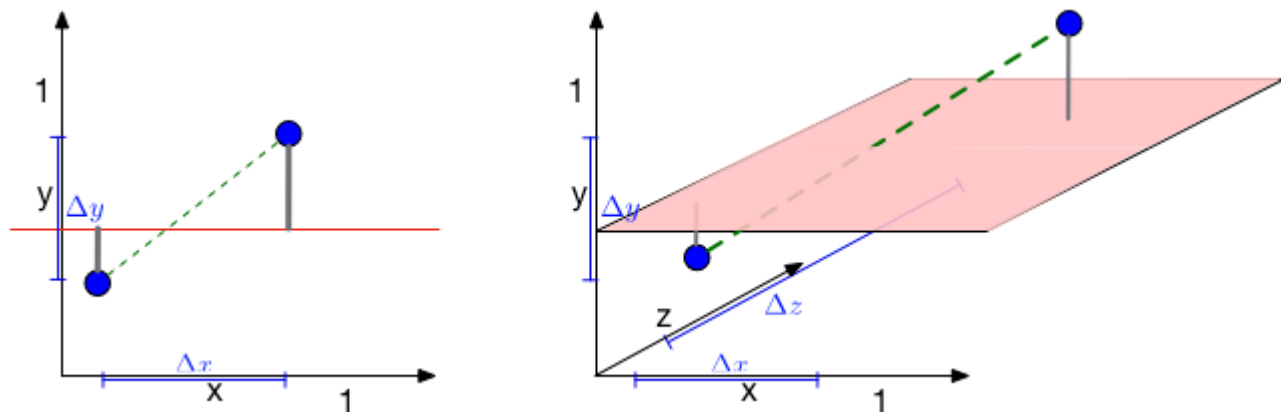
如果它们实际上并不相似的话, 为什么测试点要与那些  $k$  近邻共享标签?

# 维度灾难



图：“维度灾难”示例图。直方图展示了在  $d$ -维单位立方体内随机分布点的点对距离分布。随着维数  $d$  的增长，所有距离都集中在一个非常小的范围内。

# 到超平面的距离



在 $d$  维空间， $d - 1$  维空间是一个超平面，它的法线垂直于该超平面。在这些超平面上的移动不会增加或减少到超平面的距离，即点只是移动并保持相同的距离。

当在高维空间中点对之间的距离变得非常大时，到超平面的距离就显得很小。

这个观察与机器学习算法是高度相关的。我们后面会看到，许多分类器(例如**Perceptron** 或**SVM**) 将超平面放置在不同类别的集合之间。

维数灾难的一个后果是，大多数数据点会非常接近这些超平面。

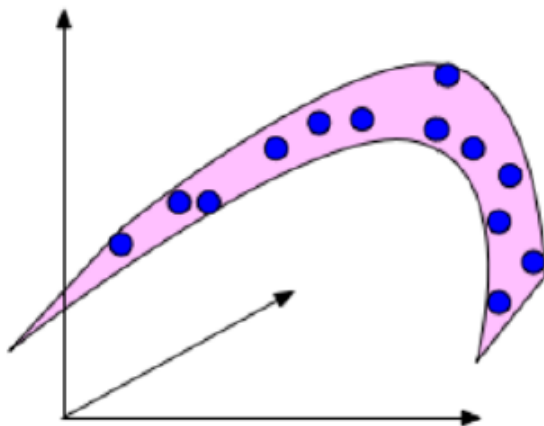
通过对输入添加微量扰动(通常是不可察觉地)，就可以改变分类结果。这种做法在近年来的研究中被称作构建**对抗样本**。

# 低维结构的数据

高维空间中的数据，是否就没有办法处理了？

实际上，大多数具有语义信息的数据位于低维子空间或子流形（Manifold）上。例如：自然图像(数字、人脸)。

虽然人脸图像可能需要1800万像素，但我们可以用少于50个特征(例如男性/女性，头发颜色，头发长短，眼睛大小，……)来描述和识别出不同人的脸。



图：此三维空间中的数据集可以用一个二维的流形来描述。蓝色的点被限制在粉红色曲面上，即嵌入在高维空间中的二维流形

# 今天的目录

- **k 近邻算法 (KNN)**
  - k-NN 的形式化定义
  - kNN 实例
- **参数选择**
  - 距离函数
  - k 值的选择
- **特殊的 k 近邻分类器**
  - 1-最近邻分类器
  - 加权的最近邻分类器
- **维度灾难**
  - 点对之间的距离
  - 到超平面的距离
  - 低维结构的数据
- **如何解决维数灾难问题**
  - 降维
  - 数据约简



# 如何解决维数灾难问题：降维

## 降维

对于高维数据（如维数超过 10），通常会在应用 k-NN 算法之前进行降维，以避免维数灾难的影响。

特征提取和降维可以通过：

- 主成分分析 (Principal component analysis, **PCA**)
- 线性判别分析 (Linear discriminant analysis, **LDA**)
- 典型相关分析 (Canonical correlation analysis, **CCA**)

等技术进行预处理，然后对降维后空间中的特征向量进行 k-NN 聚类。

在机器学习中，该过程也被称为“低维嵌入” (low-dimensional embedding)。

## 数据约简

数据约简是处理庞大数据集的一个重要的问题。

通常，只需要部分数据点就可以进行准确的分类。这些数据被称为**原型 (prototypes)**，可以通过如下方法找出：

- ① 选出**类离群点 (class outliers)**，即  $k$ -NN 分类 ( $k$  给定) 错误的训练数据；
- ② 将其余数据分为两组：
  - (i) 用于分类决策的原型 (prototypes)
  - (ii) 利用原型， $k$ -NN 可以正确分类的吸收点 (absorbed points)
- ③ 将吸收点从训练集中移除。

# 类离群点(class outliers) 的处理

被其他类的示例包围的训练示例，被称为 **class outlier**。class outlier 出现的原因包括：

- 随机噪声、测量误差等造成
- 该类的训练示例不足（出现一个孤立的样本而不是一个集群）
- 输入的属性中缺少重要的特性（即类可在其他维度中被分离）
- 太多其他类的训练例子（不平衡的类），会给一些小类造成一个“敌对”的环境

在存在类离群点 (class outliers) 时使用 k-NN 可能会引入噪音，从而降低 KNN 模型的泛化能力。给定两个正整数  $k > r > 0$ ，如果一个训练样本的  $k$  近邻中包含多于  $r$  个其他类样本，则称该训练样本为  $(k,r)$ -NN 类离群点。

# 总结

- 当距离能可靠地反映语义上有意义的差异时，k-NN 是一个简单、有效的分类器。(通过度量学习 (metric learning)，它变得更加具有竞争力)。
- 当  $n \rightarrow \infty$  时，可以证明 k-NN 是非常准确的，但运行也非常缓慢。
- 当  $d \gg 0$  时，从概率分布中采样的点不再彼此相似，k-NN 假设就失效了。
- k-NN 存储整个数据集来做为训练集。
- k-NN 不学习任何模型。
- k-NN 通过即时计算输入样本和每个训练实例之间的相似性来作出预测。

# 总结：k-NN 的优缺点

## 优点

- 对数据不作假设——对非线性数据很有用
- 简单的算法——解释和理解
- 高准确度（相对）——与更好的监督学习模型相比，准确度是足够高的，但并不具有竞争力
- 多功能——用于分类或回归

## 缺点

- 计算成本很高——因为算法存储了所有的训练数据
- 内存要求高
- 存储所有（或几乎所有）的训练数据
- 预测阶段较慢 ( $O(N)$ )
- 对不相关的特征敏感，对数据规模敏感