

K-Nearest Neighbors (KNN)

The “hello world” Algorithm

2019年6月26日

李文哲

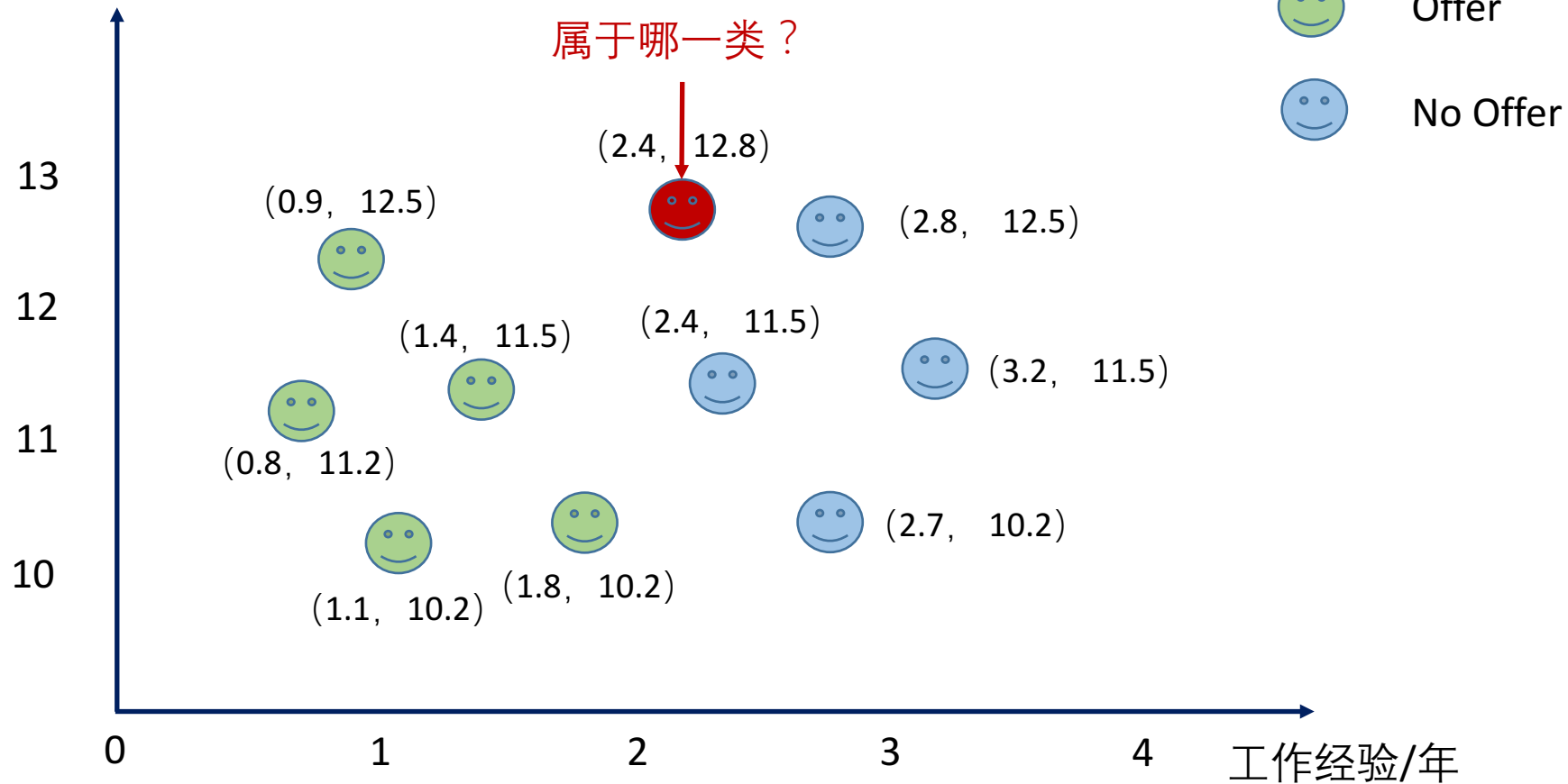
K最近邻 (KNN) 算法

- 最容易**理解**的算法：**理解核心不超过5分钟**
- 最容易**实现**的算法：**从零实现不超过5行代码**

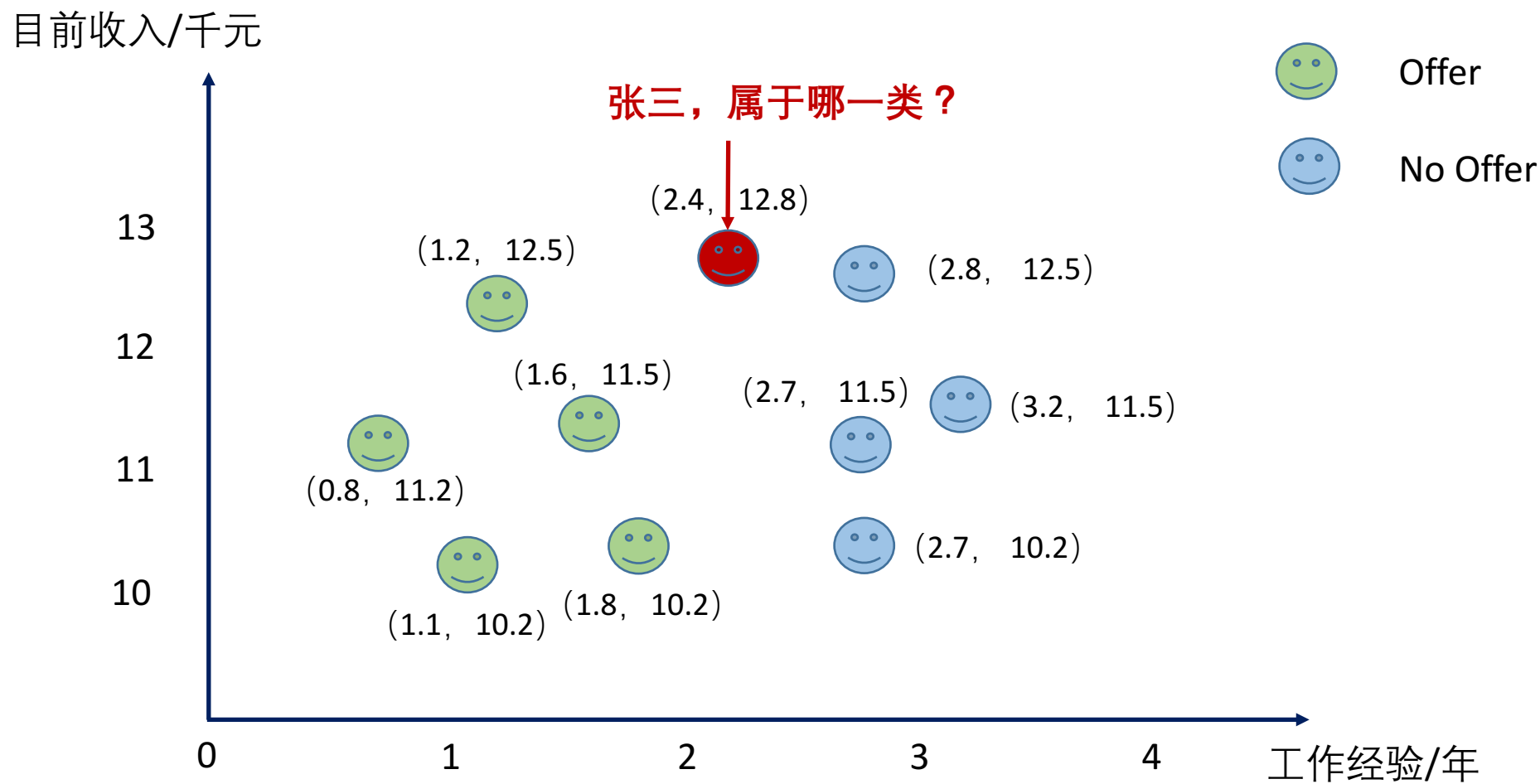
K最近邻 (KNN) 算法

Q: 张三要参加一家公司的面试，但它通过各种渠道得知了拿到offer的人的情况和没有拿到offer人的情况。我们一起来预测一下张三是否会拿到offer?

目前收入/千元

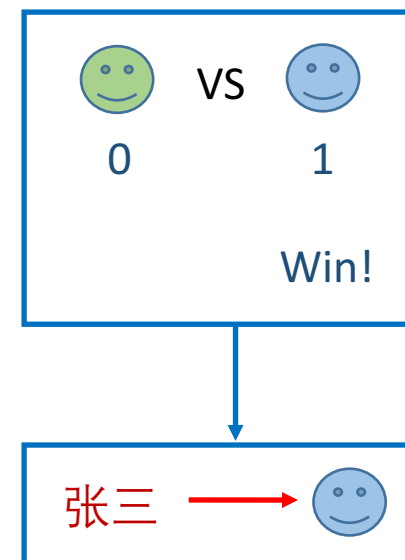
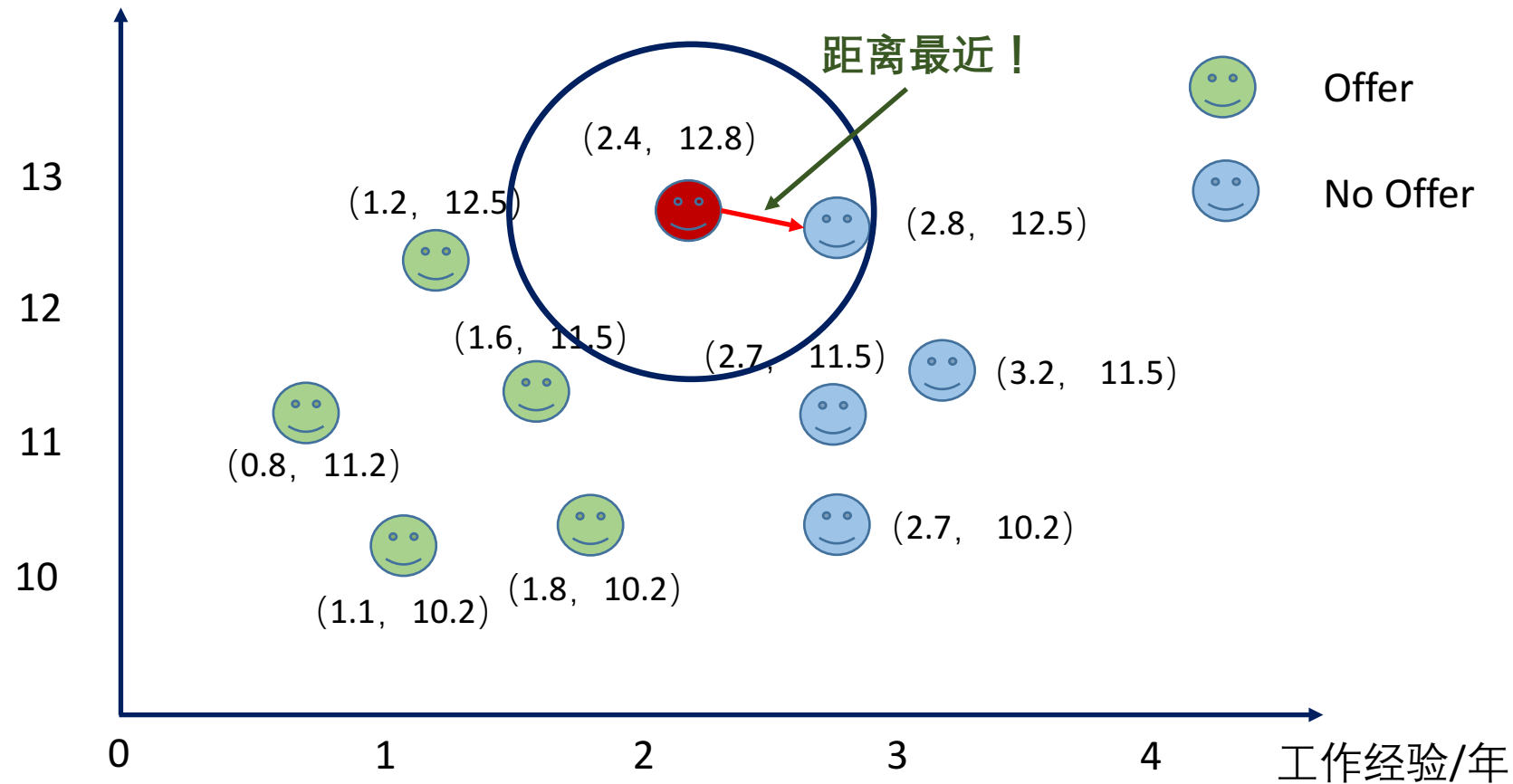


K最近邻 (KNN) 算法

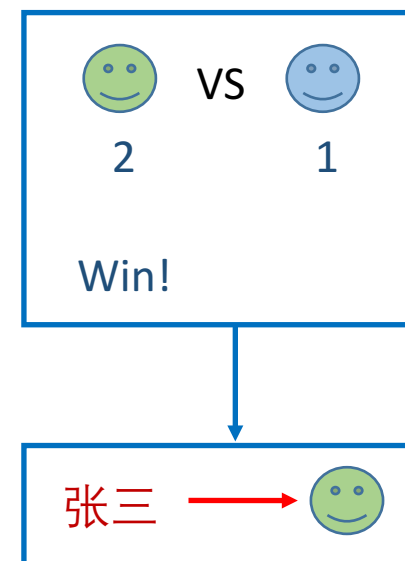
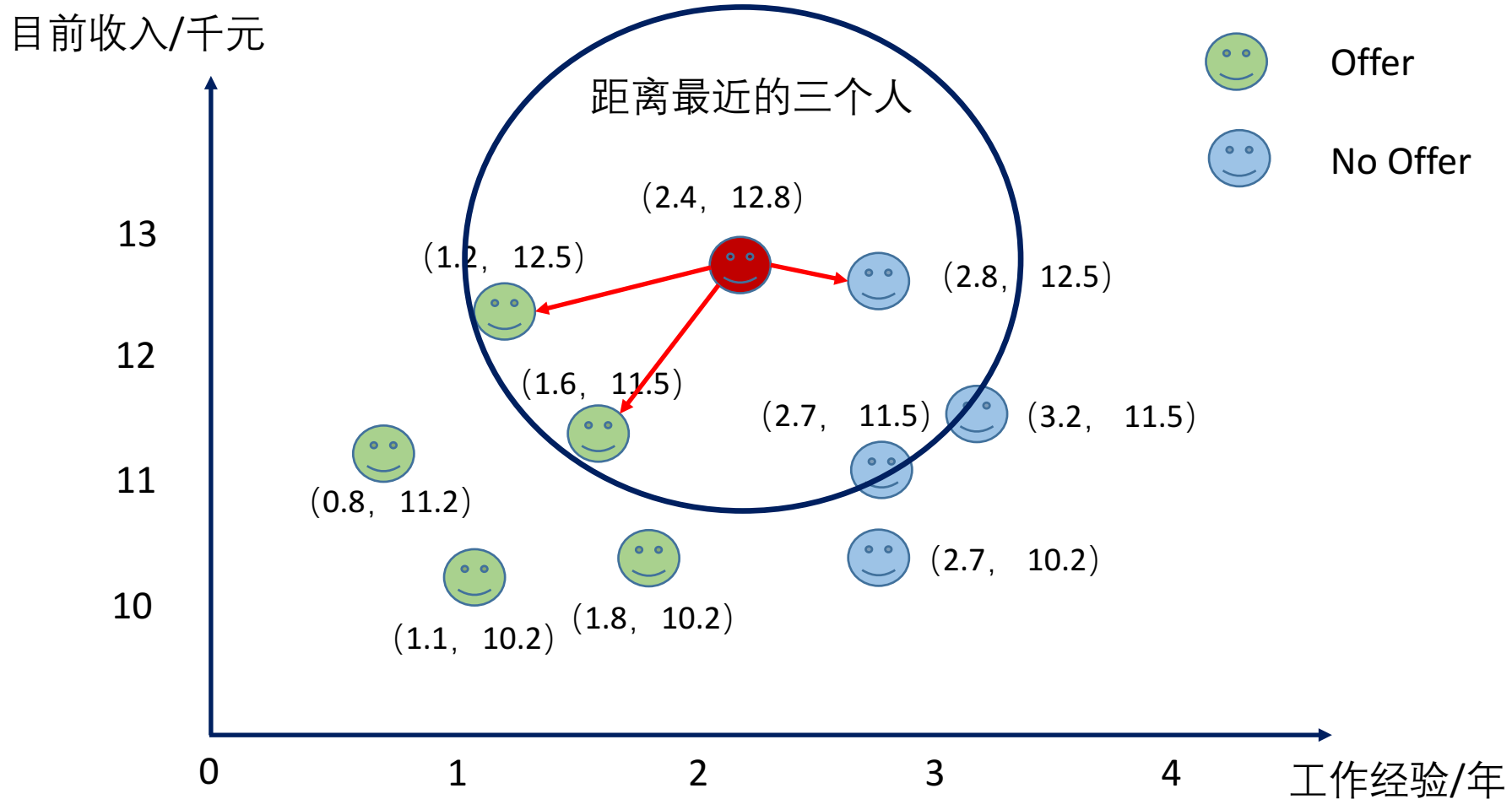


K最近邻 (KNN) 算法 (K=1)

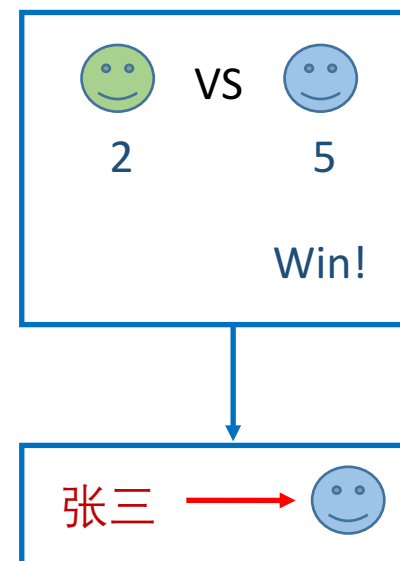
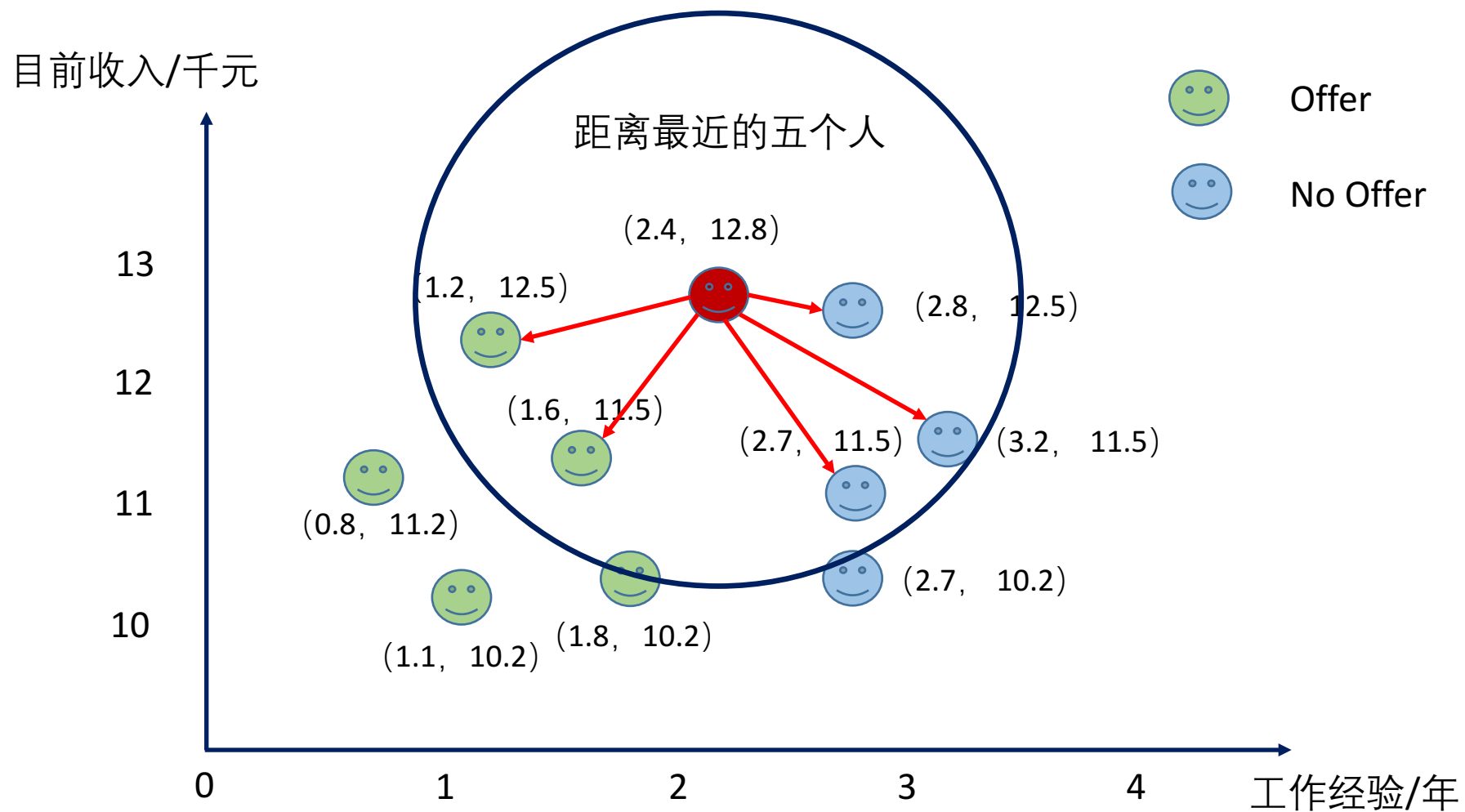
目前收入/千元



K最近邻 (KNN) 算法 (K=3)



K最近邻 (KNN) 算法 (K=5)




K最近邻 (KNN) 算法

问题： 在使用KNN算法的时候，我们一般会选择**奇数 (odd number)** 的K, 为什么？

Coding Time-1

有几个需要考虑的问题

1. 把一个物体表示成向量

 (2.8, 12.5)

2. 标记号每个物体的标签 (i.e., offer/no offer)

 Offer  No Offer

3. 计算两个物体之间的距离/相似度

(2.4, 12.8)  —————  (2.8, 12.5)

4. 选择合适的K

K=1, 3, 5, 7, 9

1. 把一个物体表示成向量

- 这也叫做“特征工程” 英文叫 **Feature Engineering**
- 模型的**输入一定是数量化的信息**，我们需要把现实生活中的物体表示成**向量/矩阵/张量**形式。

人  = (

图片 **6** = (

2. 标记号每个物体的标签

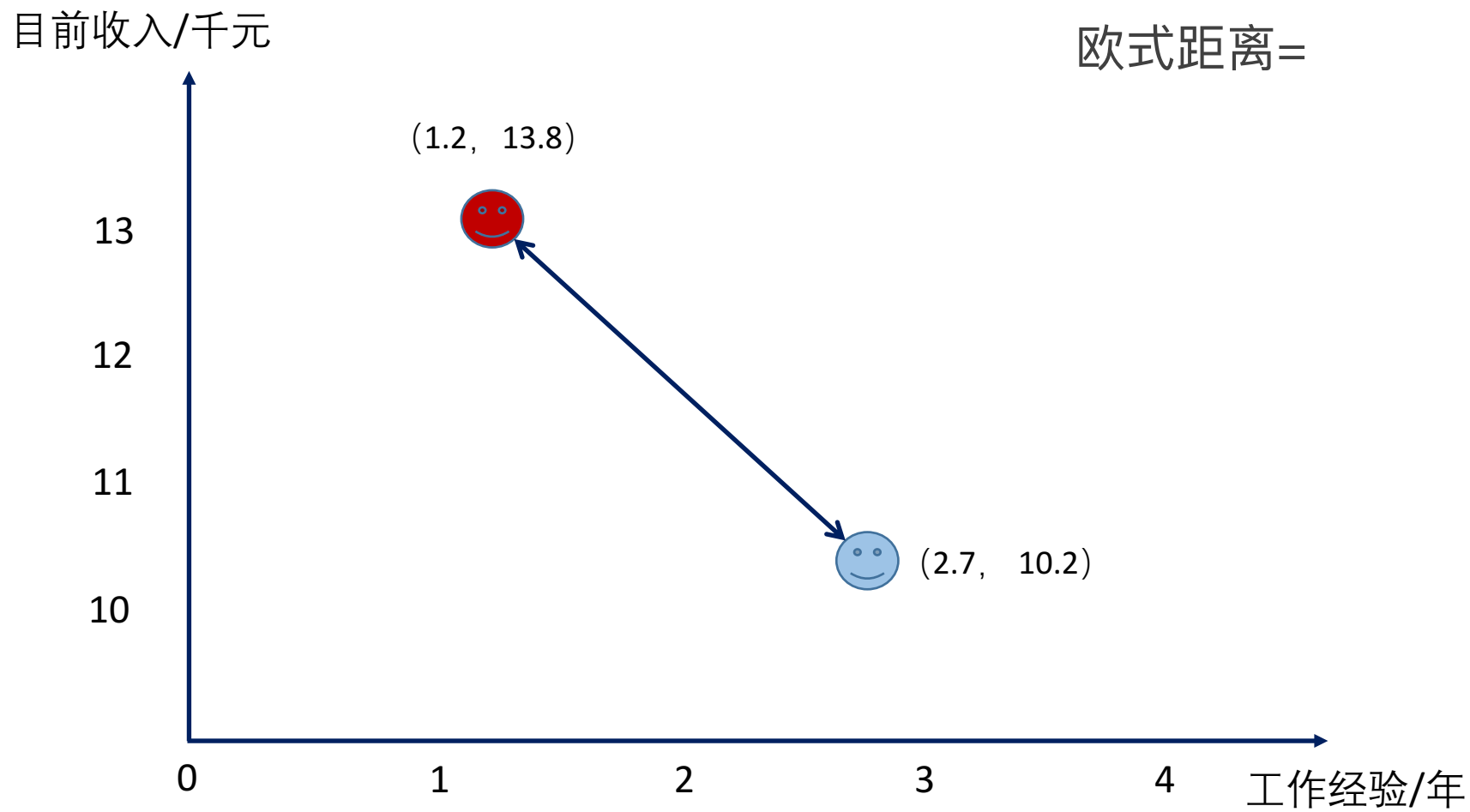


好人/坏人识别



水果种类识别

3. 计算两个物体之间的距离/相似度



Coding Time-2

4. 选择合适的K

为了选择合理的K, 首先需要去理解**K对算法的影响**

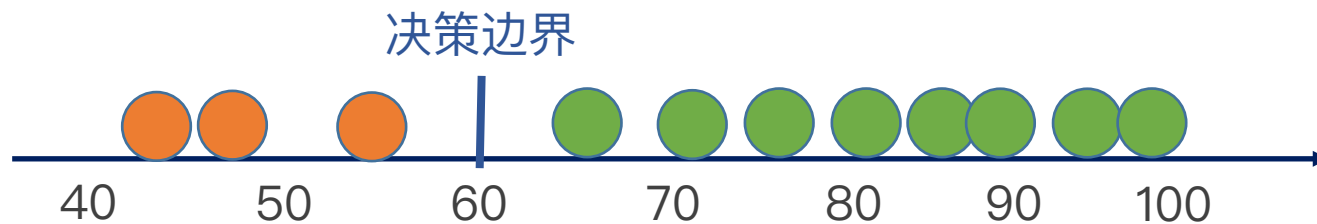


```
graph TD; A[为了选择合理的K, 首先需要去理解K对算法的影响] --> B[为了理解K对算法的影响, 需要先理解什么叫算法的决策边界]
```

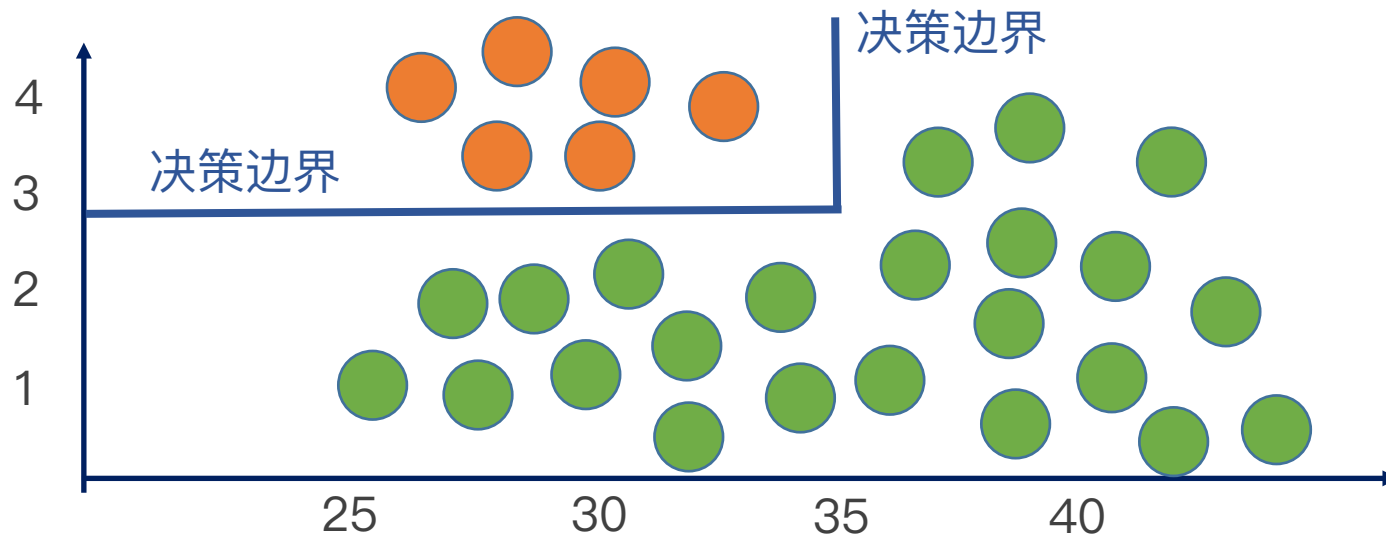
为了理解K对算法的影响, 需要先理解什么叫算法的**决策边界**

决策边界

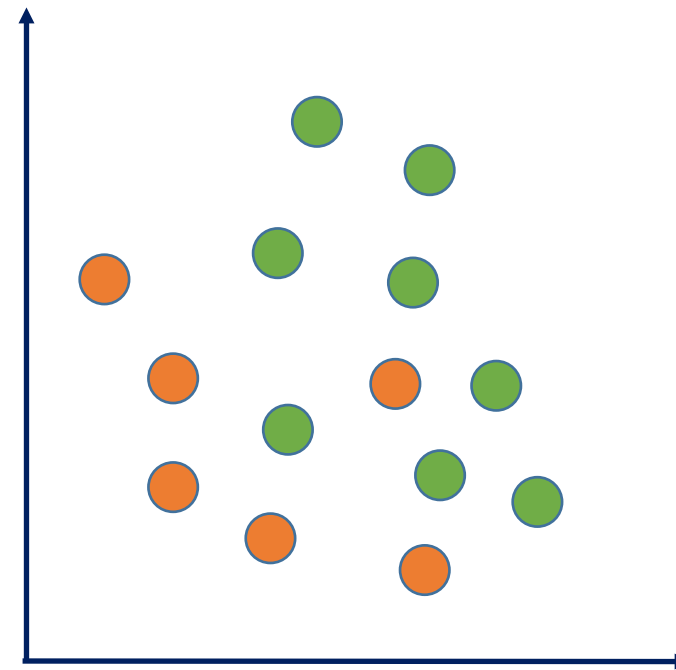
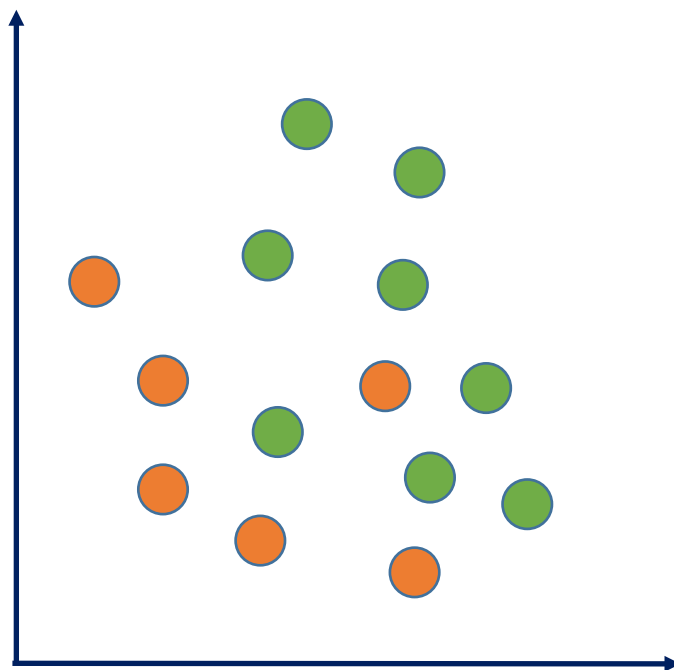
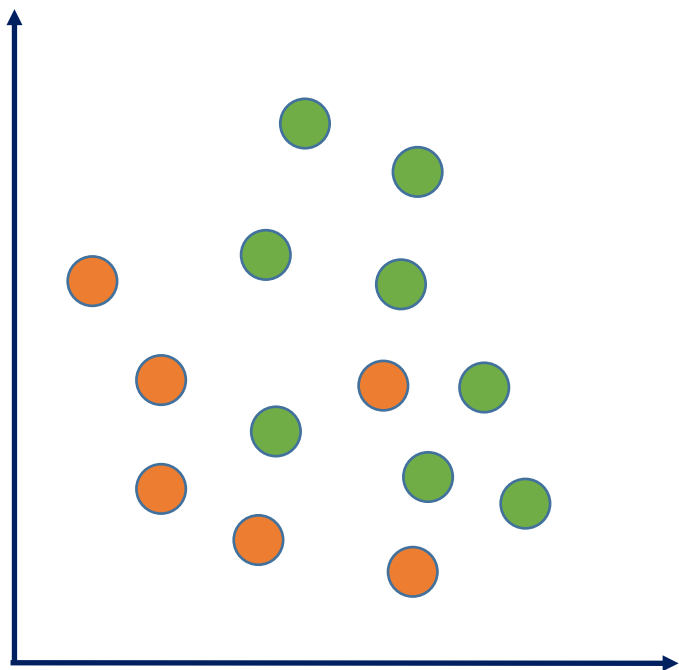
例1： 大学里60分以上作为及格， 60分以下作为不及格



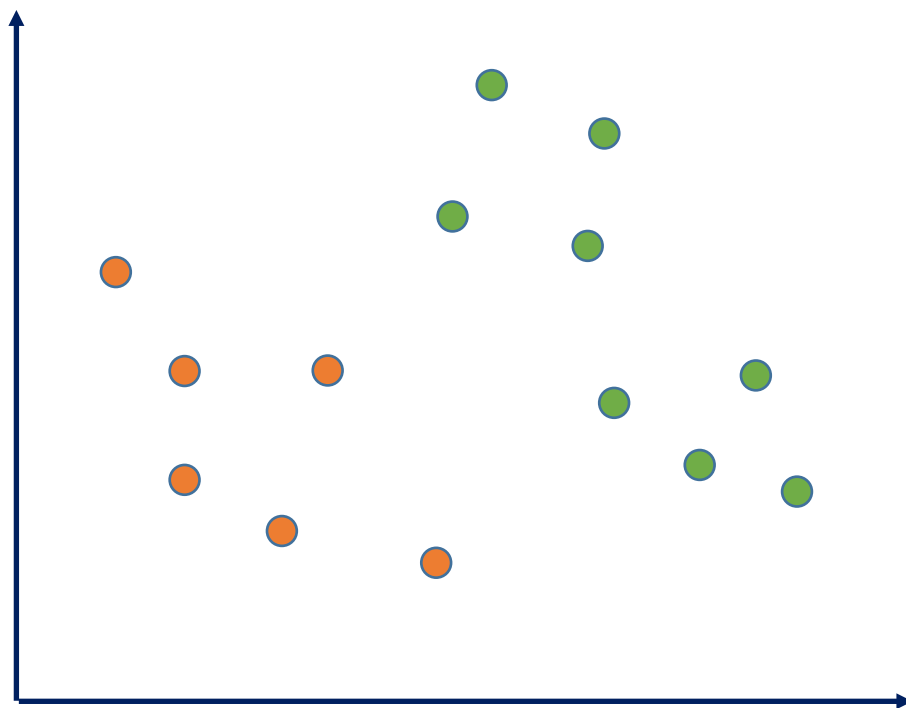
例2： 今年某高校要计划引入35岁以下， 具有海外研究经验3年以上的学者。



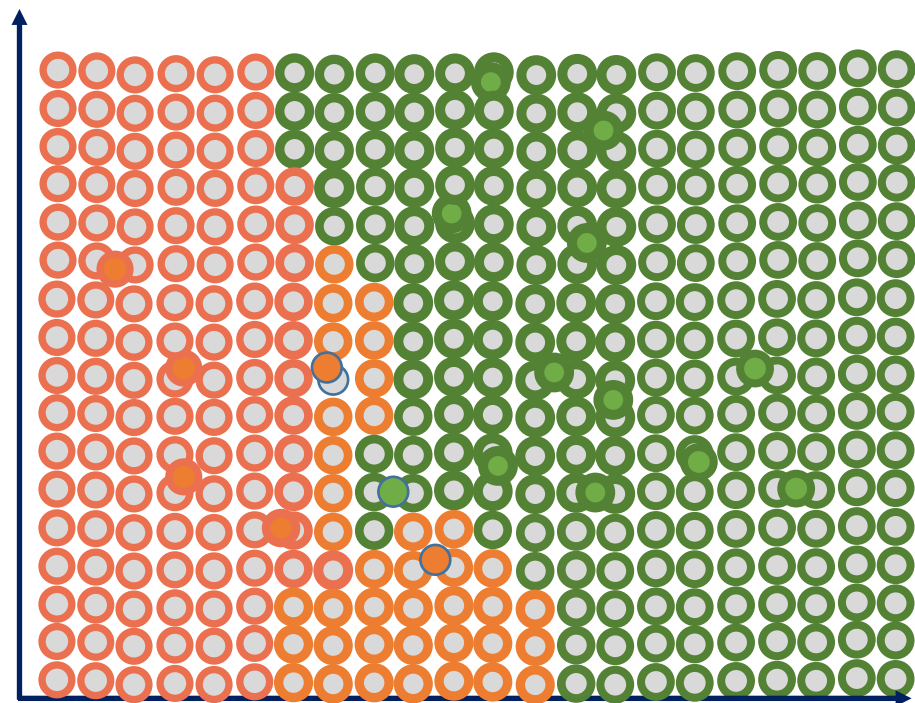
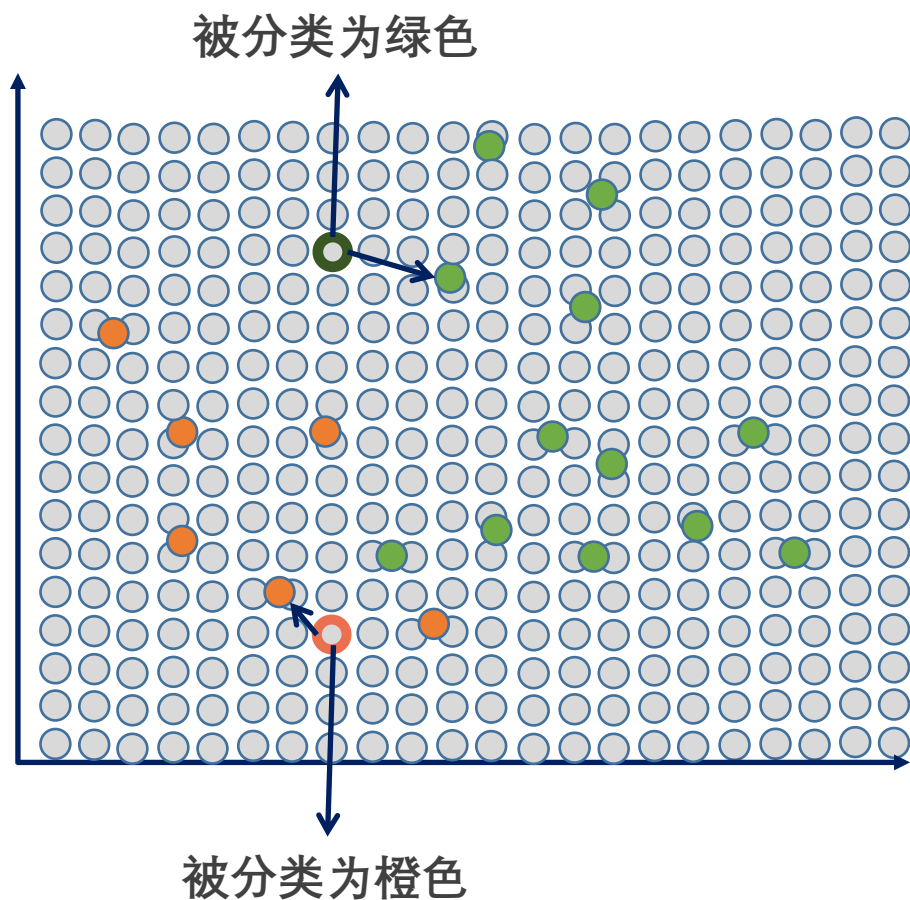
决策边界决定“线性分类器”或者“非线性分类器”



KNN的决策边界：怎么寻找 (i.e., $K=1$) ?



KNN的决策边界：怎么寻找（i.e., $K=1$ ）？



KNN的决策边界：K值的影响

问题：随着K的增加，会怎么变化？

Coding Time-3

怎么去选择合适的K?

答案是交叉验证 (Cross Validation)

通常，我们也把它归类为“调参”



但什么是交叉验证? What? How?

交叉验证

Intuition

把训练数据进一步分成训练数据 (Training Data) 和验证集 (Validation Data)。选择在验证数据里最好的超参数组合。

训练数据

测试数据

训练数据

验证数据

测试数据

5折交叉验证(5-fold Cross Validation)

K=1 时:

测试数据

训练数据

验证数据

训练数据

验证数据

训练数据

验证数据

训练数据

训练数据

验证数据

训练数据

验证数据

训练数据

5折交叉验证(5-fold Cross Validation)

K=3 时:

测试数据

训练数据

验证数据

训练数据

验证数据

训练数据

验证数据

训练数据

训练数据

验证数据

训练数据

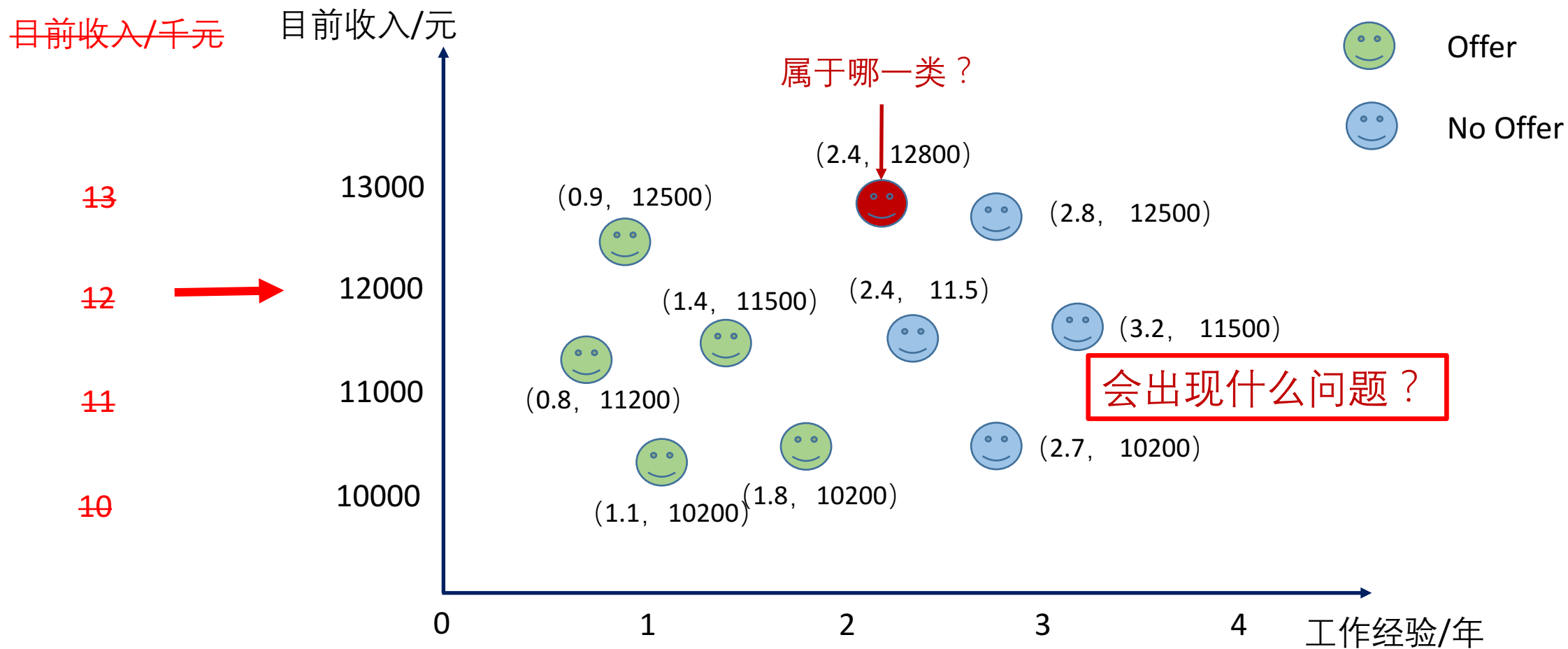
验证数据

训练数据

交叉验证中需要注意的点

1. 千万不能用测试数据来调参
2. 千万不能用测试数据来调参
3. 千万不能用测试数据来调参
4. 数据量越少，可以适当增加折数，why?

KNN中需要考虑的点 — 回顾之前的例子



特征的缩放

1. 线性归一化 (Min-max Normalization)
2. 标准差标准化 (Z-score Normalization)

特征缩放 — 线性归一化 (Min-max Normalization)

| 特征 | 特征 |
|-----|------|
| 1.5 | 0 |
| 2 | 0.2 |
| 3 | 0.6 |
| 4 | 1.0 |
| 1.5 | 0 |
| 2.5 | 0.4 |
| 2 | 0.2 |
| 2.3 | 0.33 |
| 3 | 0.6 |
| 1.5 | 0 |

数值范围 = [1.5, 4]

数值范围 = [0, 1]

$$X_{new} = \frac{X - \min(X)}{\max(X) - \min(X)}$$

↑
X的最小值
↓
X的最大值

特征缩放 — 标准差标准化 (Z-score Normalization)

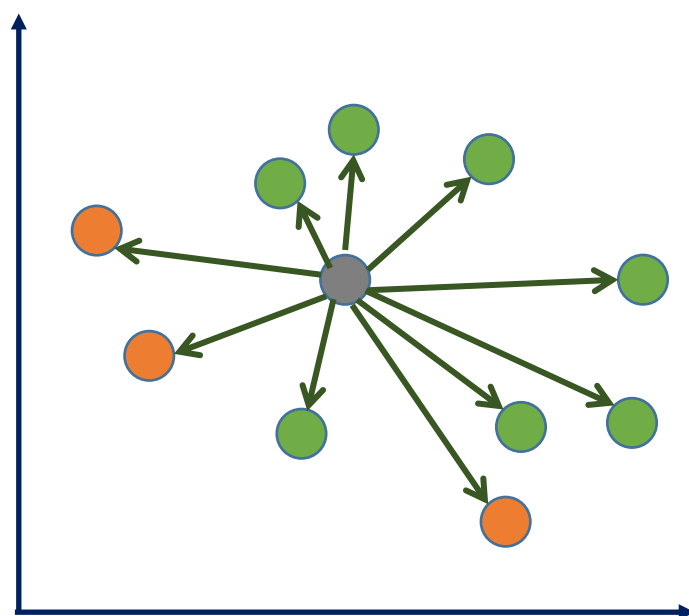
| 特征 | | 特征 |
|-----|---|------|
| 1.5 | | 0 |
| 2 | | 0.2 |
| 3 | | 0.6 |
| 4 | | 1.0 |
| 1.5 | → | 0 |
| 2.5 | | 0.4 |
| 2 | | 0.2 |
| 2.3 | | 0.33 |
| 3 | | 0.6 |
| 1.5 | | 0 |

$$X_{new} = \frac{X - \overset{\substack{\uparrow \\ \text{X的平均值}}}{mean(X)}}{std(X)}$$

\downarrow
X的标准差

Coding Time-4

KNN 的延伸内容(1) — 如何处理大数据量？

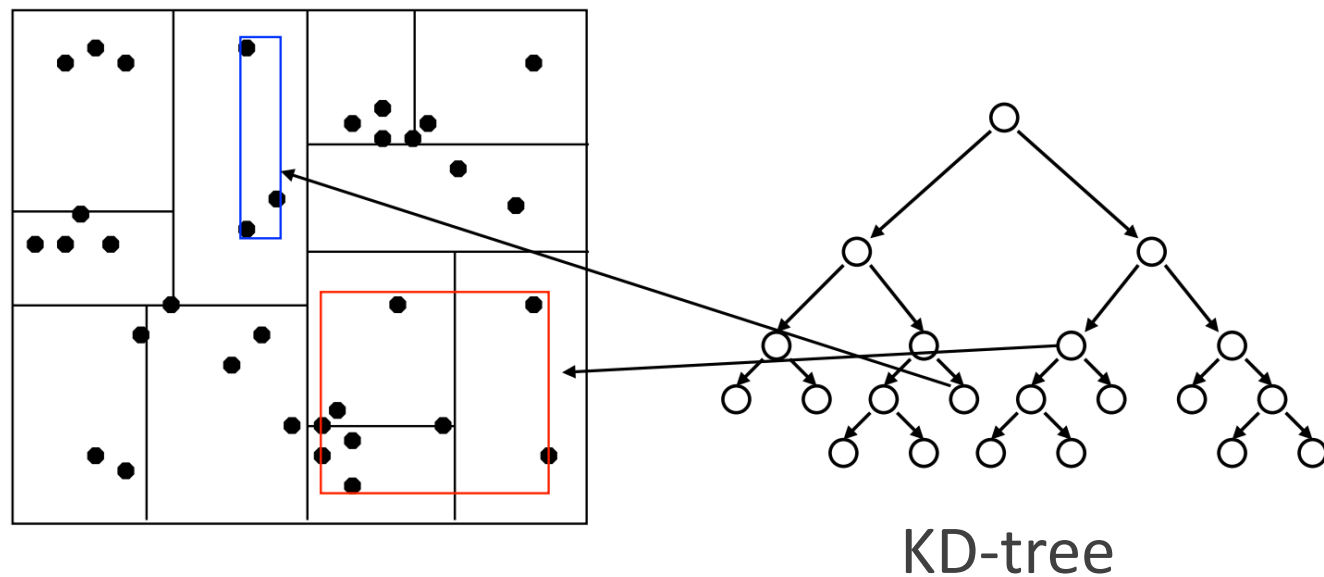


对新样本的分类

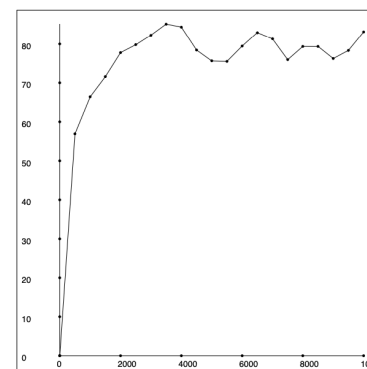
- 预测阶段：需要**计算被预测样本和每一个训练样本之间的距离**
- 时间复杂度： $O(N)$ ， N 是样本总数
- 当 N 很大的时候，这个复杂度显然不能作为实时预测

需要怎么办？

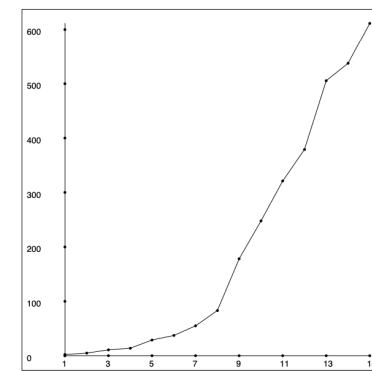
KNN 的延伸内容(1) — 如何处理大数据量?



- 对于样本个数 N 来说 $O(\log N)$
- 对于数据维度 d 来说, 指数级复杂度。



N



d

KNN 的延伸内容(1) — 如何处理大数据量?

近似算法：不再寻求完全准确的解，可以适当损失精确率



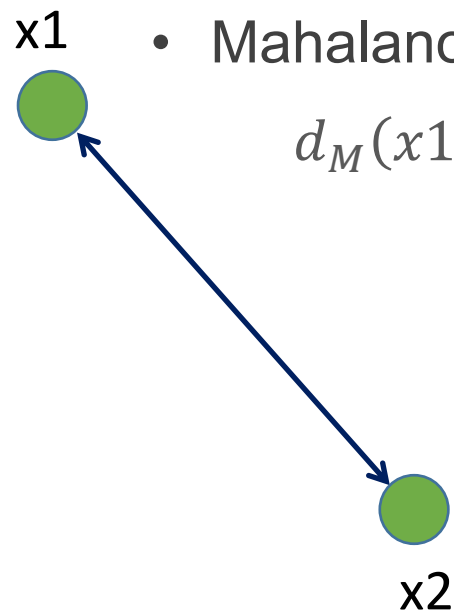
利用类似**哈希**算法 — Locality Sensitivity Hashing (LSH)

核心思想：把样本分在不同的bucket, 使得距离比较近的样本较大概率在同一个bucket里

LSH 函数 h 对于给定的相似度量函数 d , 满足：

- $d(x_1, x_2) \leq r$ 时 $p(h(x_1)=h(x_2))$ 的概率高
- $d(x_1, x_2) > \alpha r$ 时, $p(h(x_1)=h(x_2))$ 的概率低

KNN 的延伸内容(2) — 如何处理特征之间的相关性?

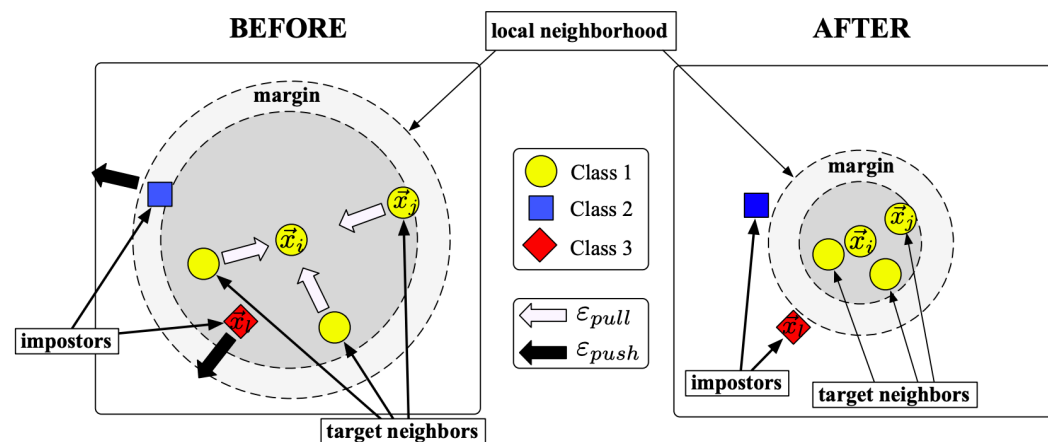


- Mahalanobis Distance

$$d_M(x_1, x_2) = (x_1 - x_2)^T \mathbf{M} (x_1 - x_2)$$

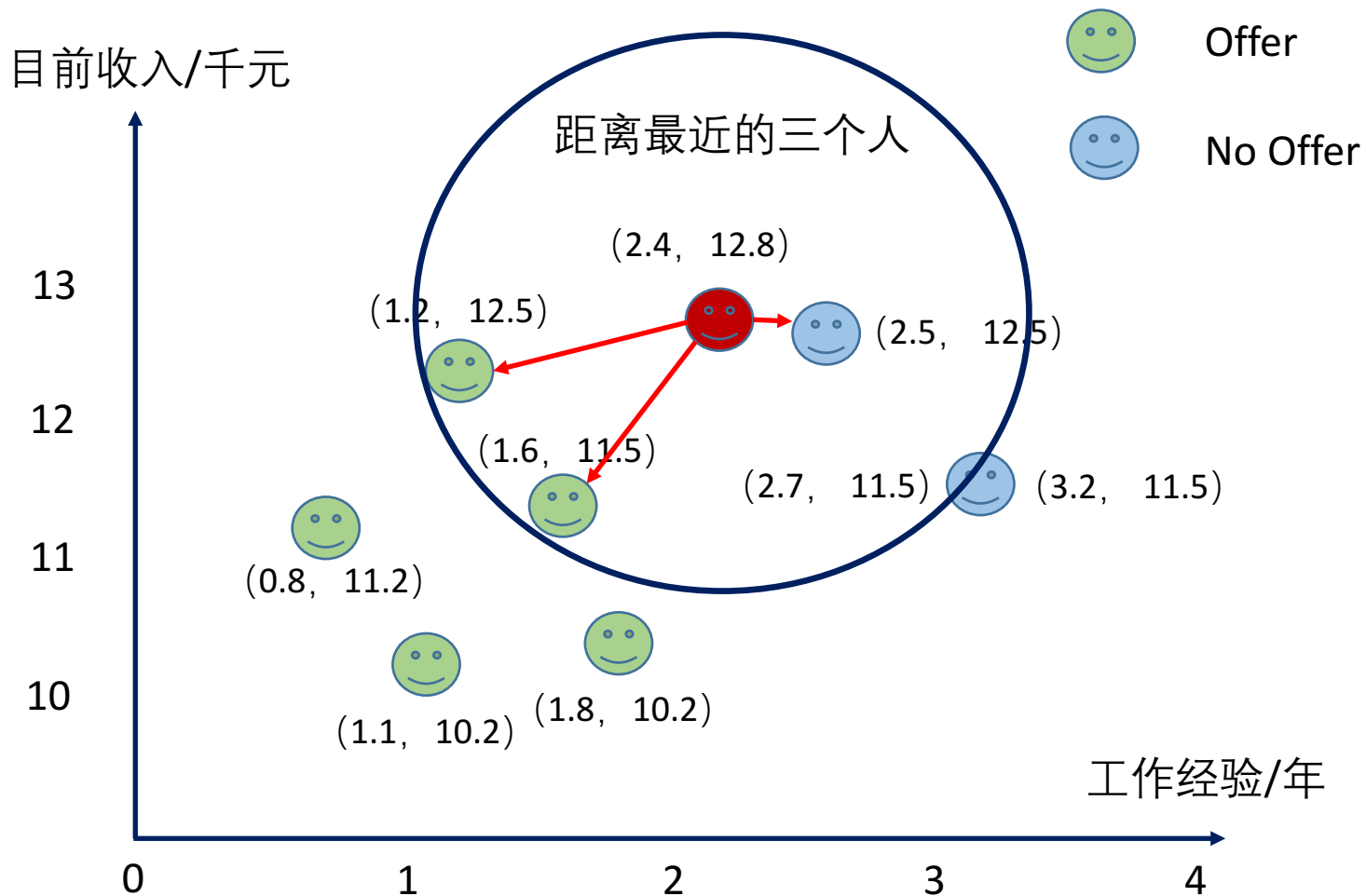
参数矩阵M怎么去学习?

Large Margin Loss



参考: Kilian et al. 2009

KNN 的延伸内容(3) — 怎么处理样本的重要性?



Weighted KNN

权重计算

$$w(\mathbf{x}, \mathbf{x}_i) = \exp(-\lambda \|\mathbf{x} - \mathbf{x}_i\|_2^2)$$

预测

$$\Pr(y|\mathbf{x}) = \frac{\sum_{i=1}^n w(\mathbf{x}, \mathbf{x}_i) \delta(y, y_i)}{\sum_{i=1}^n w(\mathbf{x}, \mathbf{x}_i)}$$

$$\delta(y, y_i) = \begin{cases} 1 & y = y_i \\ 0 & y \neq y_i \end{cases}$$

KNN 的延伸内容(4) — 能不能利用Kernel Trick?

Kernelized KNN

Epanechnikov quadratic kernel

$$K_{\lambda}(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right) \quad D(t) = \begin{cases} \frac{3}{4}(1-t^2) & \text{if } |t| \leq 1; \\ 0 & \text{otherwise.} \end{cases} \quad \lambda = \text{bandwidth}$$

tri-cube kernel

$$K_{\lambda}(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right) \quad D(t) = \begin{cases} (1-|t|^3)^3 & \text{if } |t| \leq 1; \\ 0 & \text{otherw} \end{cases}$$

Gaussian kernel

$$K_{\lambda}(x_0, x) = \frac{1}{\sqrt{2\pi\lambda}} \exp\left(-\frac{(x - x_0)^2}{2\lambda^2}\right)$$

总结

1. KNN是一个非常简单的算法
2. 比较适合应用在低维空间
3. 预测时候的复杂度高，对于大数据需要一定的处理