数据科学入门

12 K近邻法

最近邻法是最简单的预测模型之一，它没有多少数学上的假设，也不要求任何复杂的处理，它所要求的仅仅是：

• 某种距离的概念

• 一种彼此接近的点具有相似性质的假设

如果一个样本在特征空间中的k个最相似(即特征空间中最邻近)的样本中的大多数属于某一个类别，则该样本也属于这个类别，

算法描述：

1. 计算测试数据与各个训练数据之间的距离；
2. 按照距离的递增关系进行排序；
3. 选取距离最小的K个点；
4. 确定前K个点所在类别的出现频率；
5. 返回前K个点中出现频率最高的类别作为测试数据的预测分类。

> 维数灾难
>
> 高维空间过于巨大。高维空间内的点根本不会表现得彼此邻近
>
> 随着维度数量的增加，点和点之间的平均距离也增加了。但更麻烦的是最近距离和平均距离之间的比例
>
> 在更高的维度上——除非你能以指数规模得到更多的数——大片空白空间代表的是远离你想用在预测中的所有的点的区域

13 朴素贝叶斯算法

贝叶斯方法：使用概率统计的知识对样本数据集进行分类，特点是结合先验概率和后验概率，即避免了只使用先验概率的主观偏见，也避免了单独使用样本信息的过拟合现象。贝叶斯分类算法在数据集较大的情况下表现出较高的准确率，同时算法本身也比较简单

朴素贝叶斯：假定给定目标值时属性之间相互条件独立。也就是说没有哪个属性变量对于决策结果来说占有着较大的比重，也没有哪个属性变量对于决策结果占有着较小的比重

例：垃圾邮件过滤器

给定邮件是或不是垃圾邮件的条件下，其中的每个单词存在与否与其他单词毫不相干。直观地讲，就是知道某封垃圾邮件是否含有单词 viagra 无法帮助我们判断该垃圾邮件是否含有单词 rolex。

$$P(X_1=x_1,\cdots,X_n=x_n|S)=P(X_1=x_1|S)\times\cdots\times P(X_n=x_n|S)$$

> 下溢问题
>
> 通常希望尽量避免出现大量概率相乘的情况，因为计算机不擅长处理非常接近于零的浮点数。

$$\exp(\log(p_1)+\cdots+\log(p_n))$$

> 假如词汇表中的单词 data 仅出现在训练集的非垃圾邮件中，那么 *P*(“data”|*S*)=0。也就是说，对于任何含有单词 data 的邮件，我们的朴素贝叶斯分类器总是认为它是垃圾邮件的概率为 0，
>
> 伪记数；k
>
> $$P(X_i|S) = (k + \text{含有 } w_i \text{ 的垃圾邮件的数量}) / (2k + \text{垃圾邮件数量})$$

14 简单线性回归

假设有常数 *α*（alpha）和 *β*（beta），而 *εi* 是误差项。使

$$y_i=\beta x_i+\alpha+\varepsilon_i$$

需要一个更好的指标来评估模型对数据的拟合效果：决定系数或 R 平方，用来表示纳入模型的自变量引起的变动占总变动的百分比：

15 多重回归分析

$$y_i= \alpha + \beta_1 x_{i1} + \cdots + \beta_k x_{ik} + \varepsilon_i$$

- 第一个假设是 *x* 的各个列是线性无关的，即任何一列绝对不会是其他列的加权和
- 第二个重要的假设是 *x* 的各列与误差 *ε* 无关。

正则化：指给误差项添加一个惩罚项，并且该惩罚项会随着 beta 的增大而增大。然后，我们开始设法将误差项和惩罚项的组合值最小化。因此，惩罚项越大，就越能防止系数过大。

20 自然语言处理

词云：使单词及其数量可视化，它不仅能够以艺术化的形式展示单词，而且还能使单词的大小与其数量呈正比。

例：

图 20-1：由热门术语组成的词云

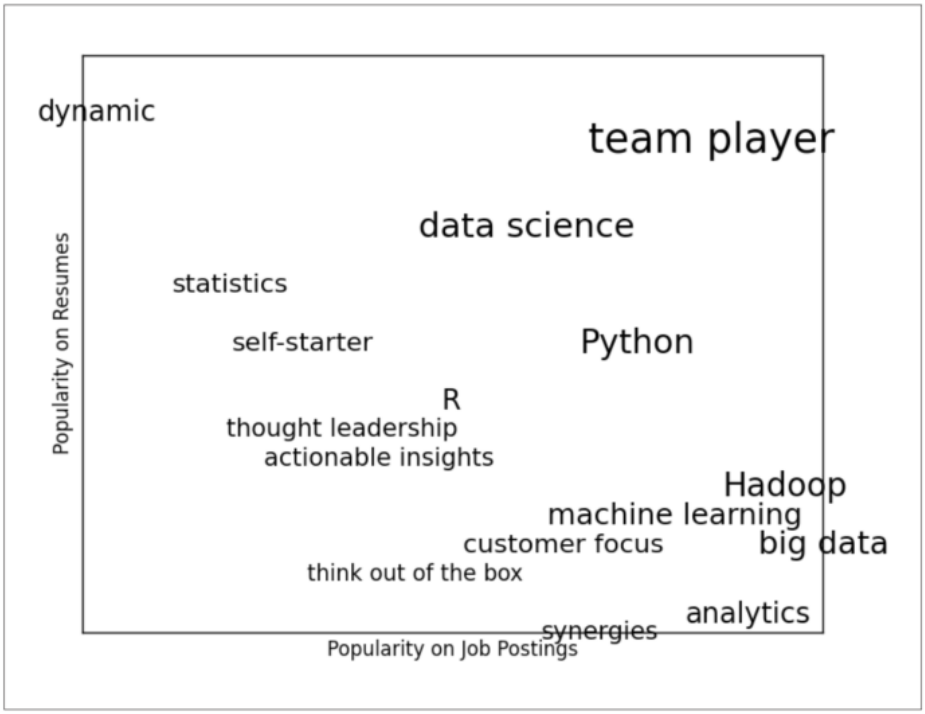

图 20-2：一个更有意义（尽管不如先前那么美观）的词云

22 网络分析

中介中心度：它可以用来找出经常位于其他节点对之间的最短路径中的人。中介中心度可以通过累加节点 *j* 和节点 *k* 之间经过节点 *i* 的最短路径所占比例，以及节点 *j* 和 *k* 之外所有的节点对中相应的比例来求出。

接近中心度：首先，为每个用户计算其疏远度（farness），即该用户到所有其他用户的最短路径的长度总和。

特征向量中心度：认为一个节点的重要性即取决于其邻居节点的数量（即该节点的度），也取决于每个邻居节点的重要性。

$$EC(i) = x_i = c\sum_{j=1}^{n} a_{ij} x_j,$$

23 推荐系统

基于用户的协同过滤方法：一种利用用户兴趣的方法是根据这些兴趣找到有类似爱好的人，然后再根据这些人的爱好来向你推荐你可能感兴趣的东西。

基于物品的协同过滤算法：直接计算两种兴趣之间的相似度，然后将与用户当前兴趣相似的兴趣放到一起，并从中为用户推荐感兴趣的东西。

24 数据库与SQL

- CREATE TABLE：创建表

```
users = [[0, "Hero", 0],
 [1, "Dunn", 2],
 [2, "Sue", 3],
 [3, "Chi", 3]]

 CREATE TABLE users (
 user_id INT NOT NULL,
 name VARCHAR(200),
 num_friends INT);
```

- INSERT：插入行

```
INSERT INTO users (user_id, name, num_friends) VALUES (0, 'Hero', 0);
```

- UPDATE：需要更新已经存在于数据库中的数据

```
UPDATE users
SET num_friends = 3
WHERE user_id = 1;
```

- DELETE：可以在表中删除行

```
DELETE FROM users;      #一个比较有风险的方法是直接删掉表的每行：

DELETE FROM users WHERE user_id = 1;  #稍微安全一点的方法是增加 WHERE 子句，
                                      #再删掉匹配某种条件的行
```

- SELECT：查询表

```
SELECT * FROM users; -- 得到所有内容
SELECT * FROM users LIMIT 2; -- 得到前两行
SELECT user_id FROM users; -- 只得到特定列
SELECT user_id FROM users WHERE name = 'Dunn'; -- 只得到特定行
```

- GROUP BY：可以将在特定列有相同值的行进行分组，并求出特定的汇总值，如 MIN、MAX、COUNT 或 SUM。

```
#你需要对每个可能的名字长度找出相应的用户数目和最小 user_id：
SELECT LENGTH(name) as name_length,
 MIN(user_id) AS min_user_id,
 COUNT(*) AS num_users
FROM users
GROUP BY LENGTH(name);
```

- ORDER BY：需要对结果排序

```
SELECT * FROM users
ORDER BY name
LIMIT 2;
```

- JOIN：关系型数据库的表通常是正则化的，意味着依照冗余最小化的原则进行组织。

```
CREATE TABLE user_interests (
 user_id INT NOT NULL,
 interest VARCHAR(100) NOT NULL
);
```

-