

推荐系统算法合集，满满都是干货（建议收藏）



推荐引擎算法学习导论：

协同过滤、聚类、分类

引言

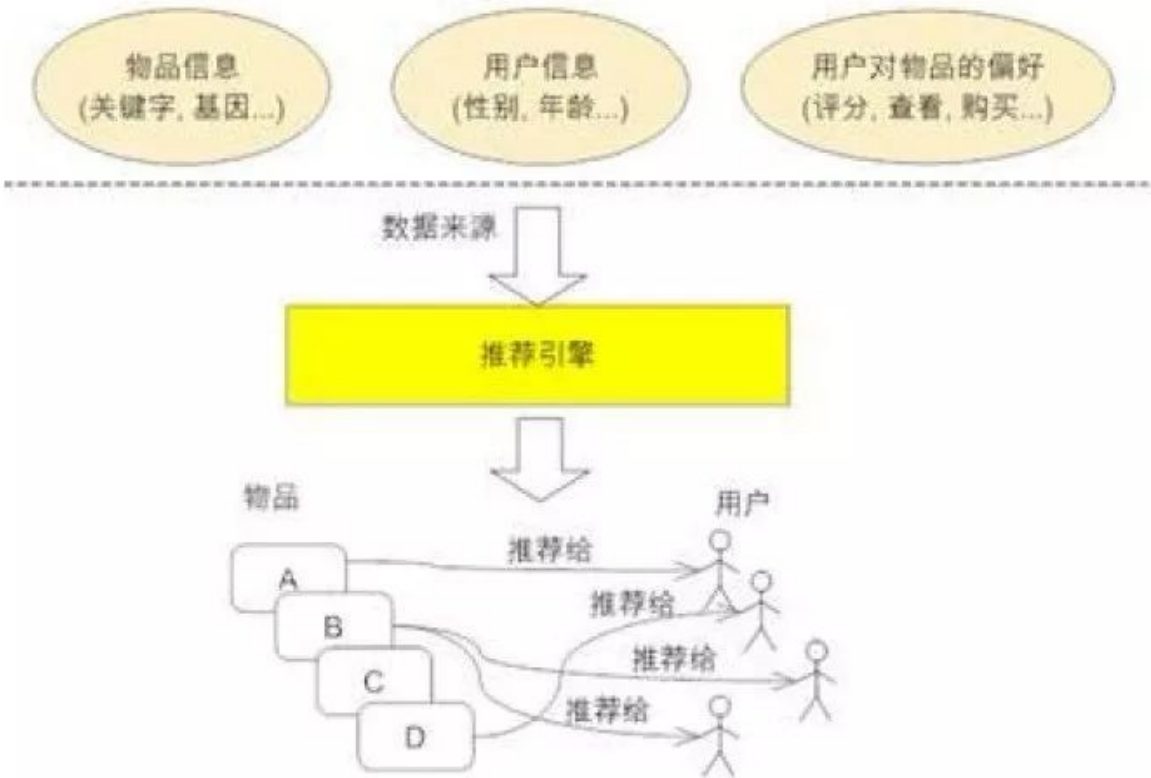
昨日看到几个关键词：语义分析，协同过滤，智能推荐，想着想着便兴奋了。于是昨天下午开始到今天凌晨3点，便研究了一下推荐引擎，做了初步了解。日后，自会慢慢深入仔细研究（日后的工作亦与此相关）。当然，此文也会慢慢补充完善。

本文作为对推荐引擎的初步介绍的一篇导论性的文章，将略去大部分的具体细节，侧重用最简单的语言简要介绍推荐引擎的工作原理以及其相关算法思想，且为了着重浅显易懂有些援引自本人1月7日在微博上发表的文字（特地整理下，方便日后随时翻阅），尽量保证本文的短小。不过，事与愿违的是，文章后续补充完善，越写越长了。

同时，本文所有相关的算法都会在日后的文章一一陆续具体阐述。本文但求微言导论，日后但求具体而论。若有任何问题，欢迎随时不吝赐教或批评指正。谢谢。

推荐引擎原理

推荐引擎尽最大努力的收集尽可能多的用户信息及行为，所谓广撒网，勤捕鱼，然后“特别的爱给特别的你”，最后基于相似性的基础之上持续“给力”，原理如下图所示（图引自本文的参考资料之一：探索推荐引擎内部的秘密）：



推荐引擎的分类

推荐引擎根据不同依据如下分类：

- 1、根据其是不是为不同的用户推荐不同的数据，分为基于大众行为（网站管理员自行推荐，或者基于系统所有用户的反馈统计计算出的当下比较流行的物品）、及个性化推荐引擎（帮你找志同道合，趣味相投的朋友，然后在此基础上实行推荐）；
- 2、根据其数据源，分为基于人口统计学的（用户年龄或性别相同判定为相似用户）、基于内容的（物品具有相同关键词和Tag，没有考虑人为因素），以及基于协同过滤的推荐（发现物品，内容或用户的相关性推荐，分为三个子类，下文阐述）；
- 3、根据其建立方式，分为基于物品和用户本身的（用户-物品二维矩阵描述用户喜好，聚类算法）、基于关联规则的（The Apriori algorithm算法是一种最有影响的挖掘布尔关联规则频繁项集的算法）、以及基于模型的推荐（机器学习，所谓机器学习，即让计算机像人脑一样持续学习，是人工智能领域内的一个子领域）。

关于上述第二个分类(2、根据其数据源)中的基于协同过滤的推荐：随着 Web2.0 的发展，Web 站点更加提倡用户参与和用户贡献，因此基于协同过滤的推荐机制因运而生。它的原理很简单，就是根据用户对物品或者信息的偏好，发现物品或者内容本身的相关性，或者是发现用户的相关性，然后再基于这些关联性进行推荐。

而基于协同过滤的推荐，又分三个子类：

1、基于用户的推荐(通过共同口味与偏好找相似邻居用户，K-邻居算法，你朋友喜欢，你也可能喜欢)，

2、基于项目的推荐(发现物品之间的相似度，推荐类似的物品，你喜欢物品A，C与A相似，可能也喜欢C)，

3、基于模型的推荐(基于样本的用户喜好信息构造一个推荐模型，然后根据实时的用户喜好信息预测推荐)。

我们看到，此协同过滤算法最大限度的利用用户之间，或物品之间的相似相关性，而后基于这些信息的基础上实行推荐。下文还会具体介绍此协同过滤。

不过一般实践中，我们通常还是把推荐引擎分两类：

- 第一类称为协同过滤，即基于相似用户的协同过滤推荐（用户与系统或互联网交互留下的一切信息、蛛丝马迹，或用户与用户之间千丝万缕的联系），以及基于相似项目的协同过滤推荐（尽最大可能发现物品间的相似度）；
- 第二类便是基于内容分析的推荐（调查问卷，电子邮件，或者推荐引擎对本blog内容的分析）。

新浪微博推荐机制

在新浪微博推荐好友的机制中：

1、我与A非好友，但我的好友中有不少人与A是好友，即我和A有不少共同的好友，那么系统便会把A也推荐给我（新浪称之为共同好友）；

2、我关注的人中有不少人关注了B，那么系统推测我也可能会喜欢B，从而亦会把B也推荐给我（新浪称之为间接关注人）。

但新浪实际操作起来，这两种方式会搅在一起，如我关注的人中，有不少人关注了B，但事实上这关注B的不少人中有些也是我的好友。以上推荐方式，统称为基于相似用户的协同过滤推荐（无非就是找到：用户与用户之间千丝万缕的联系，或是从你的好友入手，或是从你关注的人入手）。

当然，还有一类比如人气用户推荐，便是上文所述的基于大众行为的推荐，即人云亦云、跟风。系统推测大家都喜欢的，可能你也会喜欢。如大家都知道姚晨新浪微博粉丝数量排第一，则争相关注，最终粉丝量越推越高。两种推荐方式如下图所示：



The image shows a user interface for recommending people. At the top, there's a header "可能感兴趣的人" (People you might be interested in) with a "换一换" (Refresh) button. Below this, a user profile for "agentzh" is shown, including a profile picture, a "+ 加关注" (Follow) button, and "4个共同好友" (4 common friends). A callout box highlights the common friends: "我的好友中：图灵杨海玲、蒋涛CSDN、淘宝褚霸等4人也与他互相关注" (Among my friends: Turing Yang Hailing, Jiang Tao CSDN, Taobao Chu Ba, etc. 4 people also follow him mutually).

可能感兴趣的人 换一换

 agentzh
+ 加关注
4个共同好友 ▲

我的好友中：图灵杨海玲、
蒋涛CSDN、淘宝褚霸等4人
也与他互相关注



Hadoop中国

+ 加关注

4个共同好友 ▼



汪华 ✓

+ 加关注

9个间接关注人 ▲

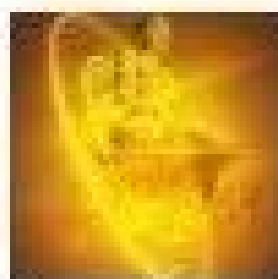
我关注的人中： 十分李鹏、
dirichlet09、李开复 等 9人也
关注了他

推荐 隐私设置

更多 »

人气用户推荐

换一换



谭飞 ✓

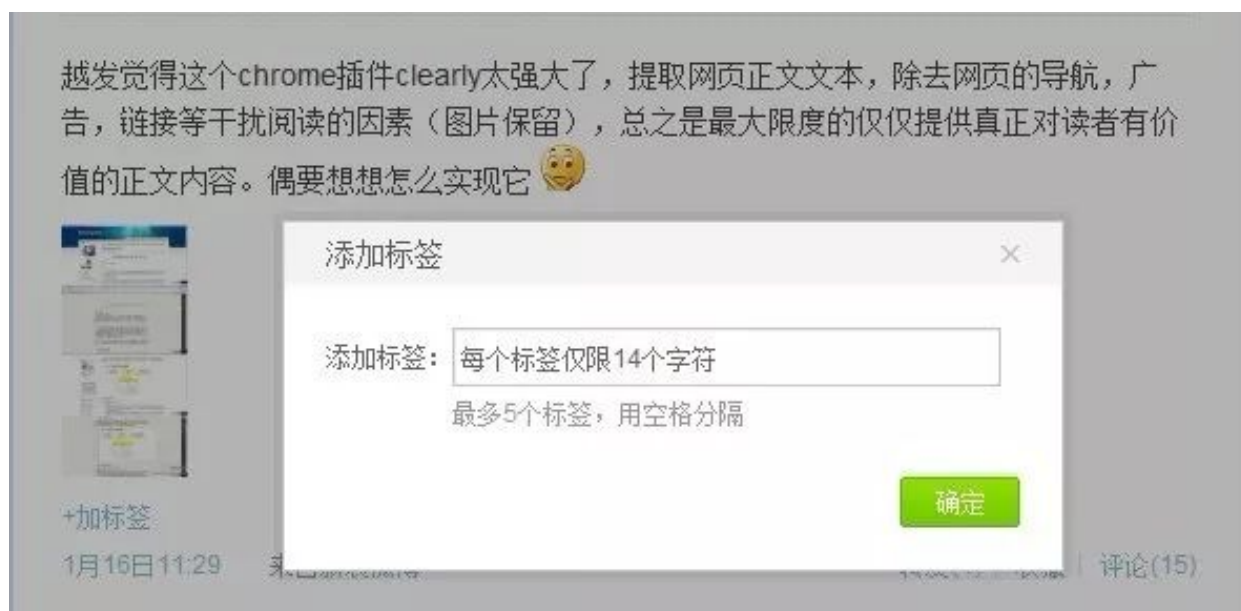
+ 加关注

姚晨等也关注他 ▲

著名时评人影评人电影监制

你关注的人：姚晨、新周刊也

不过，上述不论是基于用户的推荐方式，还是基于大众行为的推荐都并没有真正寻找到用户与用户之间共同的兴趣，偏好和口味，因为很多的时候，朋友的朋友不一定能成为你自己的朋友，且有的人清高于世，你们都追求的，我偏不屑。所以，从分析用户发表的微博的内容相关入手，找到各自共同的关注点、兴趣点才是王道。当然新浪微博最近让用户选择给自己发表的微博内容打上标签，以利于日后寻找微博内容中相关用户共同的标签tag，关键词，此种推荐方式正是基于微博内容分析的推荐。如下图：



只是问题是，谁会不遗余力发完微博后，还去给它添加什么标签呢？所以，新浪微博还得努力，寻找另一种更好地分析微博内容的方式。不然系统全盘扫描海里用户的海量微博内容，则恐怕吃不消也负担不起。

然个人觉得倒是可以从微博关键词（标签tag云）和每个用户为自己打的标签（打着越多的共同标签可定义为相似用户）入手，如下图左右部分所示：



我的标签 (10)

标签推荐

语义分析 机器学习

微软100题 信息检索

算法工程师 推荐系统

CSDN July 结构之法

算法之道

标签搜索 管理

也就是说，通过共同的好友和通过间接关注的人来定义相似用户是不靠谱的，只有通过基于微博内容的分析寻找相似用户才是可行之道，同时，更进一步，通过微博内容分析得到标签tag云后，再从中找到相同或相近的标签tag云寻找相似的用户无疑比已有推荐好友方式（通过共同的好友和通过间接关注的人来定义相似用户）更靠谱。

3.1、多种推荐方式结合

在现行的 Web 站点上的推荐往往都不是单纯只采用了某一种推荐的机制和策略，他们往往是将多个方法混合在一起，从而达到更好的推荐效果。

举个例子如Amazon中除此基于用户的推荐之外，还会用到基于内容的推荐(物品具有相同关键词和Tag)：如新产品的推荐；基于项目的协同过滤推荐(喜欢A，C与A类似，可能也喜欢C)：如捆绑销售and别人购买/浏览的商品。

总之，多种推荐方式结合，加权（用线性公式（linear formula）将几种不同的推荐按照一定权重组合起来，具体权重的值需要在测试数据集上反复实验，从而达到最好的推荐效果。）、切换、分区、分层等混合。但不论是哪种推荐方式，一般也就涵盖在上文所述的推荐方式中。

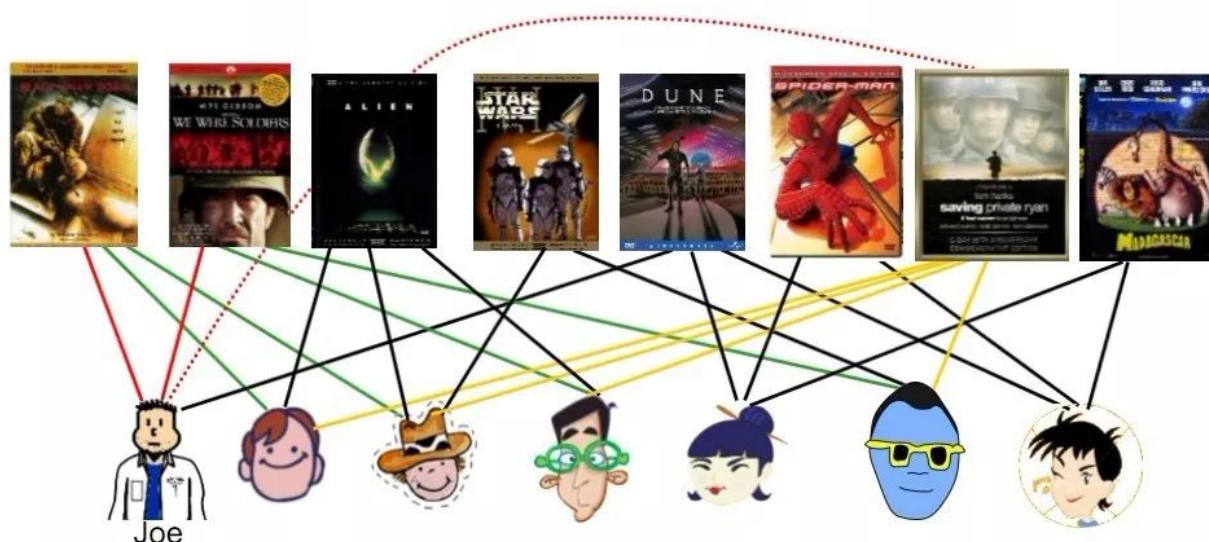
4

协同过滤推荐

协同过滤是利用集体智慧的一个典型方法。要理解什么是协同过滤(Collaborative Filtering, 简称 CF)，首先想一个简单的问题，如果你现在想看个电影，但你不知道具体看哪部，你会怎么做？大部分的人会问问周围的朋友或者

称之为广义上的邻居(neighborhood)，看看最近有什么好看的电影推荐，而我们一般更倾向于从口味比较类似的朋友那里得到推荐。这就是协同过滤的核心思想。如下图，你能从图中看到多少信息？

Neighborhood based collaborative filtering



4.1、协同过滤推荐步骤

做协同过滤推荐，一般要做好以下几个步骤：

1) 若要做协同过滤，那么收集用户偏好则成了关键。可以通过用户的行为诸如评分（如不同的用户对不同的作品有不同的评分，而评分接近则意味着喜好口味相近，便可判定为相似用户），投票，转发，保存，书签，标记，评论，点击流，页面停留时间，是否购买等获得。如下面第2点所述：所有这些信息都可以数字化，如一个二维矩阵表示出来。

2) 收集了用户行为数据之后，我们接下来便要对数据进行减噪与归一化操作(得到一个用户偏好的二维矩阵，一维是用户列表，另一维是物品列表，值是用户对物品的偏好，一般是 $[0, 1]$ 或者 $[-1, 1]$ 的浮点数值)。下面再简单介绍下减噪和归一化操作：

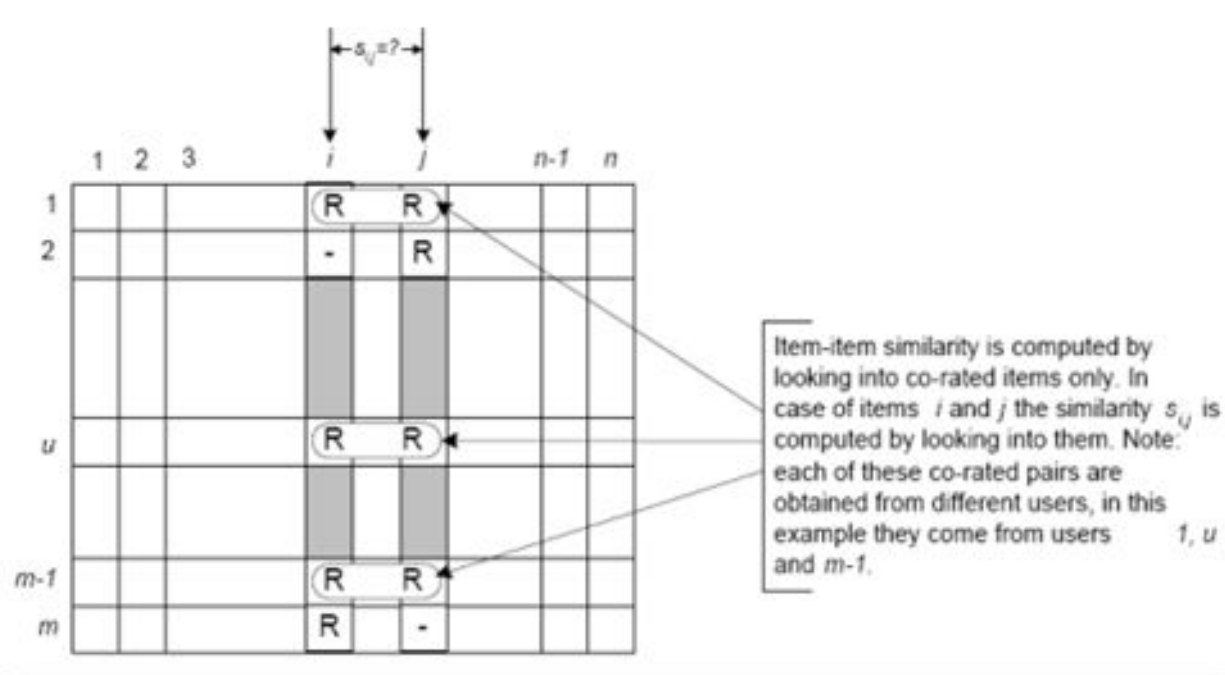
所谓减噪：用户行为数据是用户在使用应用过程中产生的，它可能存在大量的噪音和用户的误操作，我们可以通过经典的数据挖掘算法过滤掉行为数据中的噪音，这样可以是我们的分析更加精确（类似于网页的去噪处理）。

所谓归一化：将各个行为的数据统一在一个相同的取值范围中，从而使得加权求和得到的总体喜好更加精确。最简单的归一化处理，便是将各类数据除此类中的最大值，以保证归一化后的数据取值在 $[0, 1]$ 范围中。至于所谓的加权，很好理解，因为每个人占的权值不同，类似于一场唱歌比赛中对某几个选手进行投票决定其是否晋级，观众的投票抵1分，专家评委的投票抵5分，最后得分最多的选手直接晋级。

3) 找到相似的用户和物品，通过什么途径找到呢？便是计算相似用户或相似物品的相似度。

4) 相似度的计算有多种方法，不过都是基于向量Vector的，其实也就是计算两个向量的距离，距离越近相似度越大。在推荐中，用户-物品偏好的二维矩阵下，我们将某个或某几个用户对莫两个物品的偏好作为一个向量来计算两个物品之间的相似度，或者将两个用户对某个或某几个物品的偏好作为一个向量来计算两个用户之间的相似度。

相似度计算算法可以用于计算用户或者项目相似度。以项目相似度计算 (Item Similarity Computation) 为例，通性在于都是从评分矩阵中，为两个项目 i, j 挑选出共同的评分用户，然对这个共同用户的评分向量，进行计算相似度 s_{ij} ，如下图所示，行代表用户，列代表项目 (注意到是从 i, j 向量中抽出共有的评论，组成的一对向量，进行相似度计算)：



所以说，很简单，找物品间的相似度，用户不变，找多个用户对物品的评分；找用户间的相似度，物品不变，找用户对某些个物品的评分。

5) 而计算出来的这两个相似度则将作为基于用户、项目的两项协同过滤的推荐。

常见的计算相似度的方法有：欧几里德距离，皮尔逊相关系数（如两个用户对多个电影的评分，采取皮尔逊相关系数等相关计算方法，可以抉择出他们的口味和偏好是否一致），Cosine相似度，Tanimoto系数。

下面，简单介绍其中的欧几里得距离与皮尔逊相关系数：

- 欧几里德距离（Euclidean Distance）是最初用于计算欧几里德空间中两个点的距离，假设 x, y 是 n 维空间的两个点，它们之间的欧几里德距离是：

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

可以看出，当 $n=2$ 时，欧几里德距离就是平面上两个点的距离。当用欧几里德距离表示相似度，一般采用以下公式进行转换：距离越小，相似度越大（同时，避免除数为0）：

$$sim(x, y) = \frac{1}{1 + d(x, y)}$$

- 余弦相似度Cosine-based Similarity

两个项目 i, j 视作为两个 m 维用户空间向量，相似度计算通过计算两个向量的余弦夹角，那么，对于 $m \times n$ 的评分矩阵， i, j 的相似度 $sim(i, j)$ 计算公式：

$$sim(i, j) = \cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

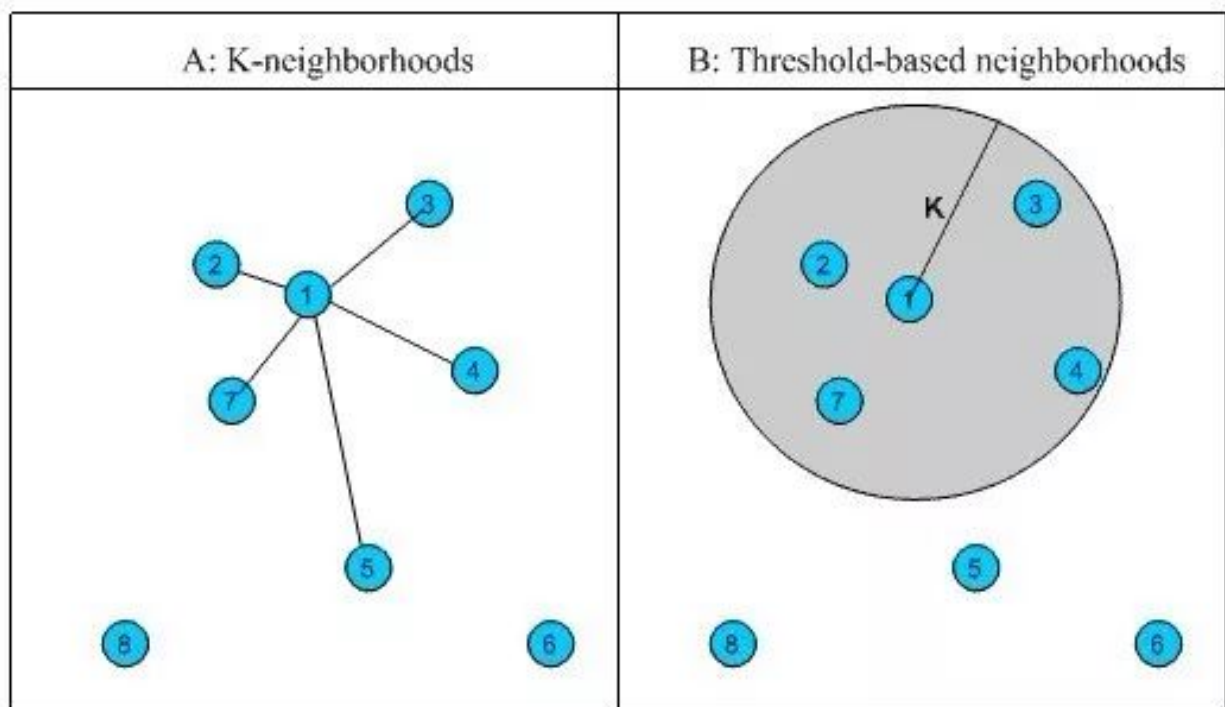
（其中 “ \cdot ” 记做两个向量的内积）

- 皮尔逊相关系数一般用于计算两个定距变量间联系的紧密程度，为了使计算结果精确，需要找出共同评分的用户。记用户集U为既评论了 i 又评论了 j 的用户集，那么对应的皮尔森相关系数计算公式为：

$$sim(i, j) = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_i)(R_{u,j} - \bar{R}_j)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_i)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_j)^2}}$$

其中 $R_{u,i}$ 为用户u 对项目 i 的评分，对应带横杠的为这个用户集U对项目i的评分评分。

6) 相似邻居计算。邻居分为两类：1、固定数量的邻居K-neighborhoods（或Fix-size neighborhoods），不论邻居的“远近”，只取最近的 K 个，作为其邻居，如下图A部分所示；2、基于相似度门槛的邻居，落在以当前点为中心，距离为 K 的区域中的所有点都作为当前点的邻居，如下图B部分所示。



再介绍一下K最近邻 (k-Nearest Neighbor, KNN) 分类算法：这是一个理论上比较成熟的方法，也是最简单的机器学习算法之一。该方法的思路是：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。

7) 经过4) 计算出来的基于用户的CF(基于用户推荐之用：通过共同口味与偏好找相似邻居用户，K-邻居算法，你朋友喜欢，你也可能喜欢)，基于物品的CF(基于项目推荐之用：发现物品之间的相似度，推荐类似的物品，你喜欢物品A，C与A相似，那么你可能也喜欢C)。

4. 2、基于基于用户相似度与项目相似度

上述3. 1节中三个相似度公式是基于项目相似度场景下的，而实际上，基于用户相似度与基于项目相似度计算的一个基本的区别是，基于用户相似度是基于评分矩阵中的行向量相似度求解，基于项目相似度计算式基于评分矩阵中列向量相似度求解，然后三个公式分别都可以适用，如下图：

	Item 1	Item 2	Item 3	Item 4	Item 5	Item 6
User 1	4	0	1	3	2	3
User 2	3	0	2	3	3	4
User 3	4	5	6	0	2	0
User 4	0	3	1	0	0	2
User 5	5	0	2	0	0	2
User 6	4	2	2	1	2	4

（其中，为0的表示未评分）

- 基于项目相似度计算式计算如Item3，Item4两列向量相似度；
- 基于用户相似度计算式计算如User3，User4量行向量相似度。

千言万语不如举个例子。我们来看一个具体的基于用户相似度计算的例子。

假设我们有一组用户，他们表现出了对一组图书的喜好。用户对一本图书的喜好程度越高，就会给其更高的评分。我们来通过一个矩阵来展示它，行代表用户，列代表图书。

如下图所示，所有的评分范围从1到5，5代表喜欢程度最高。第一个用户（行1）对第一本图书（列1）的评分是4，空的单元格表示用户未给图书评分。



						
	4	3			5	
	5		4		4	
	4		5	3	4	
		3				5
		4				4
			2	4		5

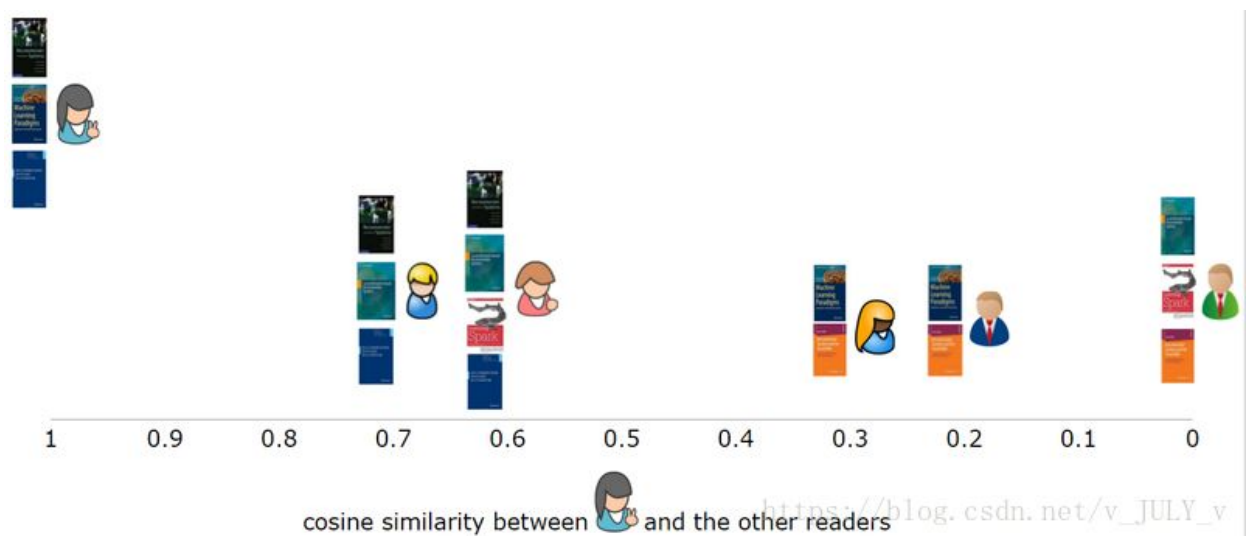
使用基于用户的协同过滤方法，我们首先要做的是基于用户给图书做出的评价，计算用户之间的相似度。

让我们从一个单一用户的角度考虑这个问题，看图1中的第一行，要做到这一点，常见的做法是将使用包含了用户喜好项的向量（或数组）代表每一个用户。相较于使用多样化的相似度量这种做法，更直接。













在这个例子中，我们将使用余弦相似性去计算用户间的相似度。

当我们把第一个用户和其他五个用户进行比较时，就能直观的看到他和其他用户的相似程度。

对于大多数相似度量，向量之间相似度越高，代表彼此更相似。本例中，第一个用户第二、第三个用户非常相似，有两本共同书籍，与第四、第五个用户的相似度低一些，只有一本共同书籍，而与最后一名用户完全不相似，因为没有一本共同书籍。

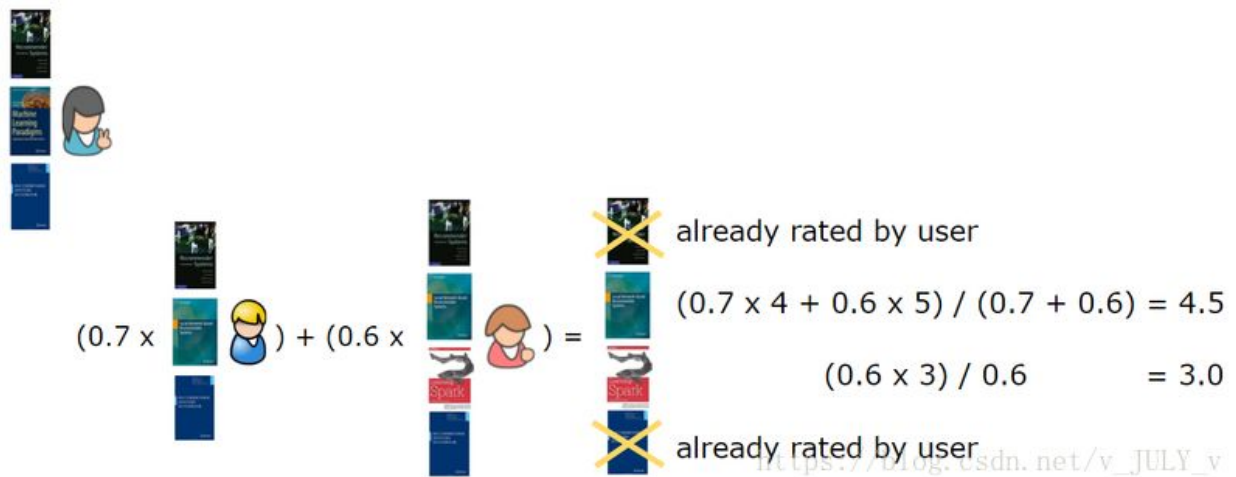


更一般的，我们可以计算出每个用户的相似性，并且在相似矩阵中表示它们。这是一个对称矩阵，单元格的背景颜色表明用户相似度的高低，更深的红色表示它们之间更相似。

						
	1.00	0.75	0.63	0.22	0.30	0.00
	0.75	1.00	0.91	0.00	0.00	0.16
	0.63	0.91	1.00	0.00	0.00	0.40
	0.22	0.00	0.00	1.00	0.97	0.64
	0.30	0.00	0.00	0.97	1.00	0.53
	0.00	0.16	0.40	0.64	0.53	1.00

所以，我们找到了与第一个用户最相似的第二个用户，删除用户已经评价过的书籍，给最相似用户正在阅读的书籍加权，然后计算出总和。

在这种情况下，我们计算出 $n=2$ ，表示为了产生推荐，需要找出与目标用户最相似的两个用户，这两个用户分别是第二个和第三个用户，然后第一个用户已经评价了第一和第五本书，故产生的推荐书是第三本（4.5分），和第四本（3分）。



此外，什么时候用 item-base，什么时候用 user-base 呢：
<http://weibo.com/1580904460/zhZ9AiIkZ?mod=weibotime?>

一般说来，如果item数目不多，比如不超过十万，而且不显著增长的话，就用item-based好了。为何？如@wuzh670所说，如果item数目不多+不显著增长，说明item之间的关系在一段时间内相对稳定(对比user之间关系)，对于实时更新item-similarity需求就降低很多，推荐系统效率提高很多，故用item-based会明智些。

反之，当item数目很多，建议用user-base。当然，实践中具体情况具体分析。如下图所示（摘自项亮的《推荐系统实践》一书）：

表2-11 UserCF和ItemCF优缺点的对比

	UserCF	ItemCF
性能	适用于用户较少的场合，如果用户很多，计算用户相似度矩阵代价很大	适用于物品数明显小于用户数的场合，如果物品很多（网页），计算物品相似度矩阵代价很大
领域	时效性较强，用户个性化兴趣不太明显的领域	长尾物品丰富，用户个性化需求强烈的领域
实时性	用户有新行为，不一定造成推荐结果的立即变化	用户有新行为，一定会导致推荐结果的实时变化
冷启动	在新用户对很少的物品产生行为后，不能立即对他进行个性化推荐，因为用户相似度表是每隔一段时间离线计算的	新用户只要对一个物品产生行为，就可以给他推荐和该物品相关的其他物品
	新物品上线后一段时间，一旦有用户对物品产生行为，就可以将新物品推荐给对它产生行为的用户兴趣相似的其他用户	但没有办法在不离线更新物品相似度表的情况下将新物品推荐给用户
推荐理由	很难提供令用户信服的推荐解释	利用用户的历史行为给用户做推荐解释，可以令用户比较信服

5

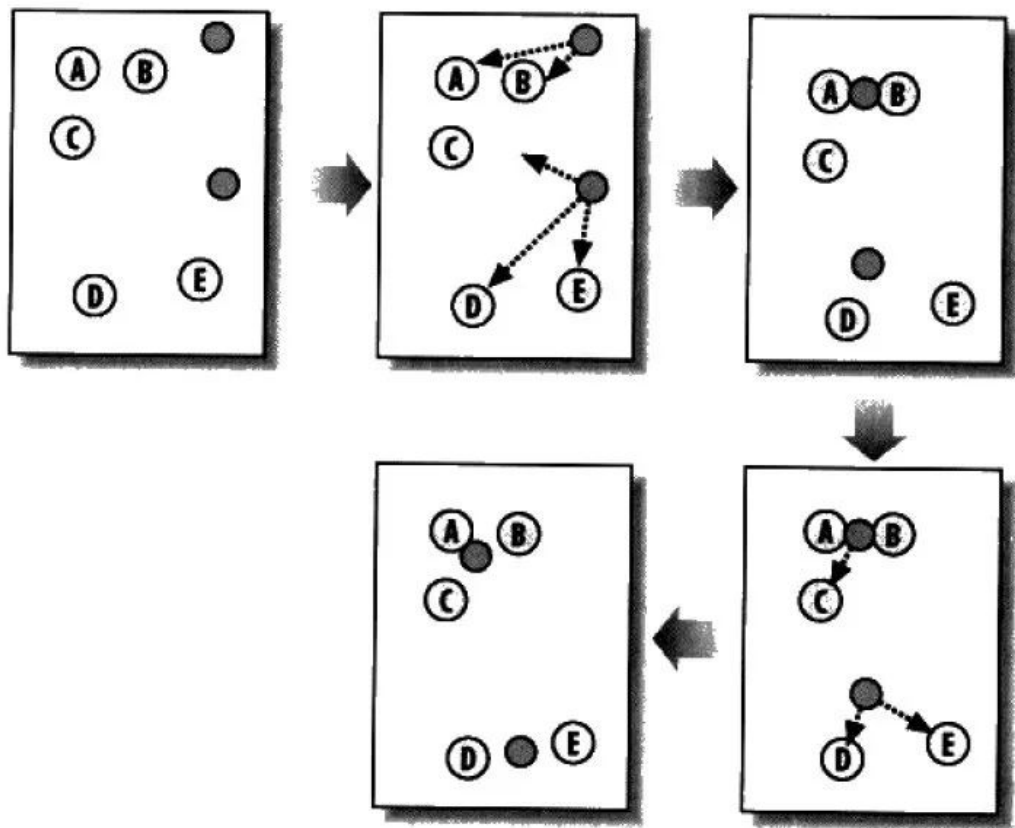
聚类算法

聚类聚类，通俗的讲，即所谓“物以类聚，人以群分”。聚类（Clustering）是一个数据挖掘的经典问题，它的目的是将数据分为多个簇（Cluster），在同一个簇中的对象之间有较强的相似度，而不同簇的对象差别较大。

5.1、K 均值聚类算法

K-均值（K-Means）聚类算法与处理混合正态分布的最大期望算法很相似，因为他们都试图找到数据中自然聚类的中心。此算法假设对象属性来自于空间向量，目标是使各个群组内部的均方误差总和最小。

K均值聚类算法首先会随机确定K个中心位置（位于空间中代表聚类中心的点），然后将各个数据项分配给最临近的中心点。待分配完成之后，聚类中心就会移到分配给该聚类的所有节点的平均位置处，然后整个分配过程重新开始。这一过程会一直重复下去，直到分配过程不再产生变化为止。下图是包含两个聚类的K-均值聚类过程：



以下代码所示即是此K-均值聚类算法的python实现：

```

1 //K-均值聚类算法
2 import random
3
4 def kcluster(rows,distance=pearson,k=4):
5     # 确定每个点的最小值和最大值
6     ranges=[(min([row[i] for row in rows]),max([row[i] for row in rows]))
7             for i in range(len(rows[0]))]
8
9     # 随机创建k个中心点
10    clusters=[[random.random()*(ranges[i][1]-ranges[i][0])+ranges[i][0]
11              for i in range(len(rows[0]))] for j in range(k)]
12
13    lastmatches=None
14    for t in range(100):
15        print 'Iteration %d' % t
16        bestmatches=[] for i in range(k)]
17
18        # 在每一行中寻找距离最近的中心点
19        for j in range(len(rows)):
20            row=rows[j]
21            bestmatch=0
22            for i in range(k):
23                d=distance(clusters[i],row)
24                if d<distance(clusters[bestmatch],row): bestmatch=i
25            bestmatches[bestmatch].append(j)
26
27        # 如果结果与上一次相同，则整个过程结束
28        if bestmatches==lastmatches: break
29        lastmatches=bestmatches
30
31        # 把中心点移到其所有成员的平均位置
32        for i in range(k):
33            avgs=[0.0]*len(rows[0])
34            if len(bestmatches[i])>0:
35                for rowid in bestmatches[i]:
36                    for m in range(len(rows[rowid])):
37                        avgs[m]+=rows[rowid][m]
38                for j in range(len(avgs)):
39                    avgs[j]/=len(bestmatches[i])
40                clusters[i]=avgs
41
42    # 返回k组序列，其中每个序列代表一个聚类
43    return bestmatches

```

k-Means是一种机器学习领域中的一种非监督学习。下面，简要介绍下监督学习与无监督学习：

- 监督学习的任务是学习带标签的训练数据的功能，以便预测任何有效输入的值。监督学习的常见例子包括将电子邮件消息分类为垃圾邮件，根据类别标记网页，以及识别手写输入。创建监督学习程序需要使用许多算法，最常见的包括神经网络、Support Vector Machines (SVMs) 和 Naive Bayes 分类程序。

- 无监督学习的任务是发挥数据的意义，而不管数据的正确与否。它最常应用于将类似的输入集成到逻辑分组中。它还可以用于减少数据集中的维度数据，以便只专注于最有用的属性，或者用于探明趋势。无监督学习的常见方法包括K-Means，分层集群和自组织地图。

5.2、Canopy 聚类算法

Canopy 聚类算法的基本原则是：首先应用成本低的近似的距离计算方法高效的将数据分为多个组，这里称为一个 Canopy，我们姑且将它翻译为“华盖”，Canopy 之间可以有重叠的部分；然后采用严格的距离计算方式准确的计算在同一 Canopy 中的点，将他们分配与最合适的簇中。Canopy 聚类算法经常用于 K 均值聚类算法的预处理，用来找合适的 k 值和簇中心。

5.3、模糊 K 均值聚类算法

模糊 K 均值聚类算法是 K 均值聚类的扩展，它的基本原理和 K 均值一样，只是它的聚类结果允许存在对象属于多个簇，也就是说：它属于我们前面介绍过的可重叠聚类算法。为了深入理解模糊 K 均值和 K 均值的区别，这里我们得花些时间了解一个概念：模糊参数（Fuzziness Factor）。

与 K 均值聚类原理类似，模糊 K 均值也是在待聚类对象向量集合上循环，但是它并不是将向量分配给距离最近的簇，而是计算向量与各个簇的相关性（Association）。假设有一个向量 v ，有 k 个簇， v 到 k 个簇中心的距离分别是 $d_1, d_2 \dots d_k$ ，那么 v 到第一个簇的相关性 u_1 可以通过下面的算式计算：

$$u_1 = \frac{1}{\left(\frac{d_1}{d_1}\right)^{\frac{2}{m-1}} + \left(\frac{d_1}{d_2}\right)^{\frac{2}{m-1}} + \dots + \left(\frac{d_1}{d_k}\right)^{\frac{2}{m-1}}}$$

计算 v 到其他簇的相关性只需将 d_1 替换为对应的距离。从上面的算式，我们看出，当 m 近似 2 时，相关性近似 1；当 m 近似 1 时，相关性近似于到该簇的距离，所以 m 的取值在 $(1, 2)$ 区间内，当 m 越大，模糊程度越大， m 就是我们刚刚提到的模糊参数。

其余聚类算法本文不再介绍。关于冷启动、数据稀疏、可扩展性、可移植性、可解释性、多样性、推荐信息的价值等问题则待后续阐述。

6

分类算法

接下来，分类算法有很多，本文介绍决策树学习，与贝叶斯定理。

6.1、决策树学习

咱们直接切入正题。所谓决策树，顾名思义，是一种树，一种依托于策略抉择而建立起来的树。

机器学习中，决策树是一个预测模型；他代表的是对象属性与对象值之间的一种映射关系。树中每个节点表示某个对象，而每个分叉路径则代表的某个可能的属性值，而每

个叶结点则对应从根节点到该叶节点所经历的路径所表示的对象的值。决策树仅有单一输出，若欲有复数输出，可以建立独立的决策树以处理不同输出。

从数据产生决策树的机器学习技术叫做决策树学习，通俗说就是决策树。

来理论的太过抽象，下面举两个浅显易懂的例子：

第一个例子：通俗来说，决策树分类的思想类似于找对象。现想象一个女孩的母亲要给这个女孩介绍男朋友，于是有了下面的对话：

女儿：多大年纪了？

母亲：26。

女儿：长的帅不帅？

母亲：挺帅的。

女儿：收入高不？

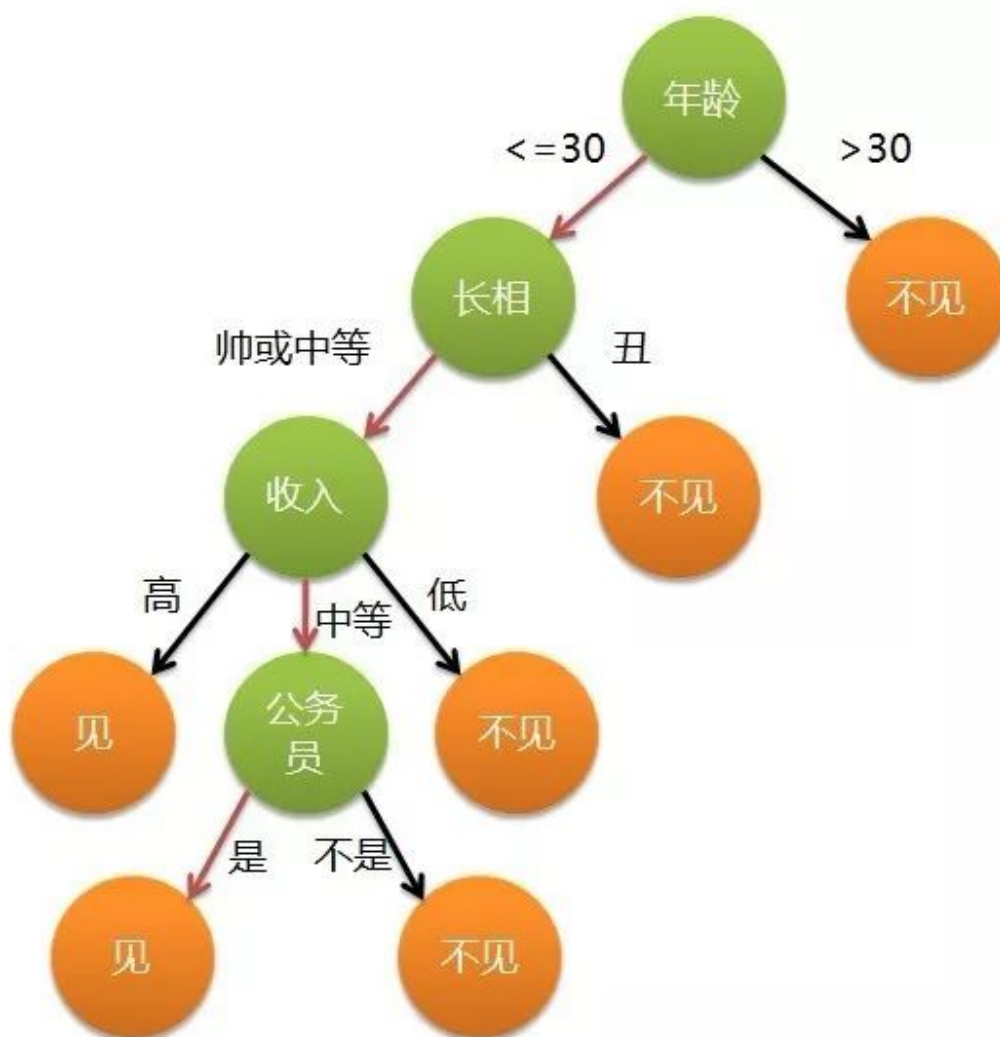
母亲：不算很高，中等情况。

女儿：是公务员不？

母亲：是，在税务局上班呢。

女儿：那好，我去见见。

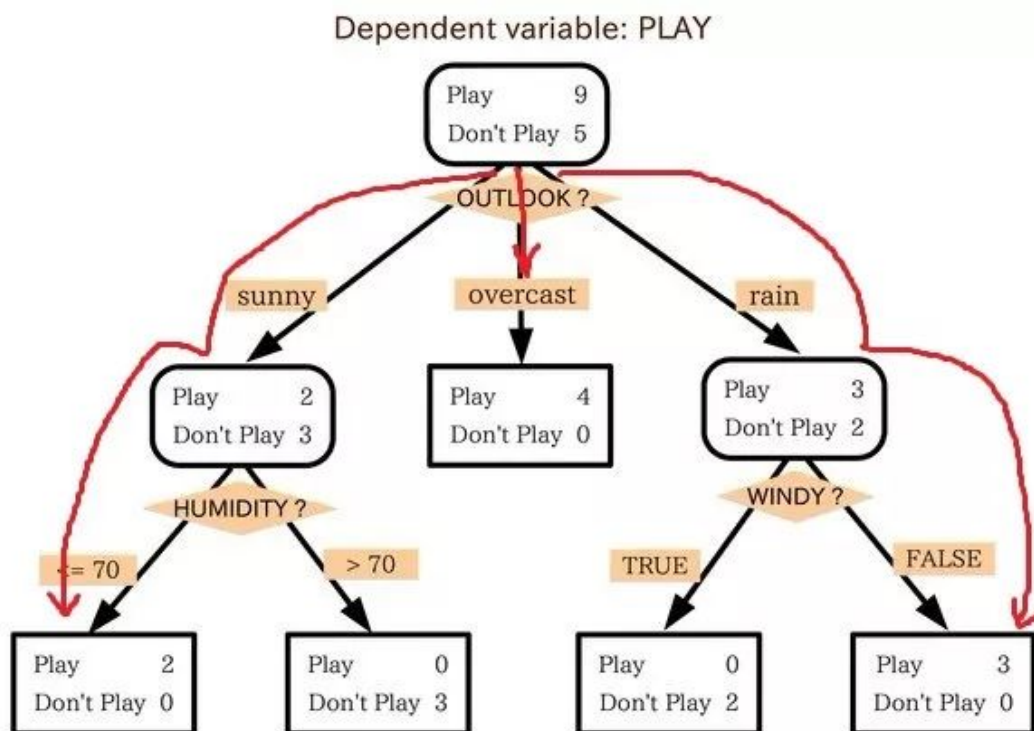
这个女孩的决策过程就是典型的分类树决策。相当于通过年龄、长相、收入和是否公务员对将男人分为两个类别：见和不见。假设这个女孩对男人的要求是：30岁以下、长相中等以上并且是高收入者或中等以上收入的公务员，那么这个可以用下图表示女孩的决策逻辑：



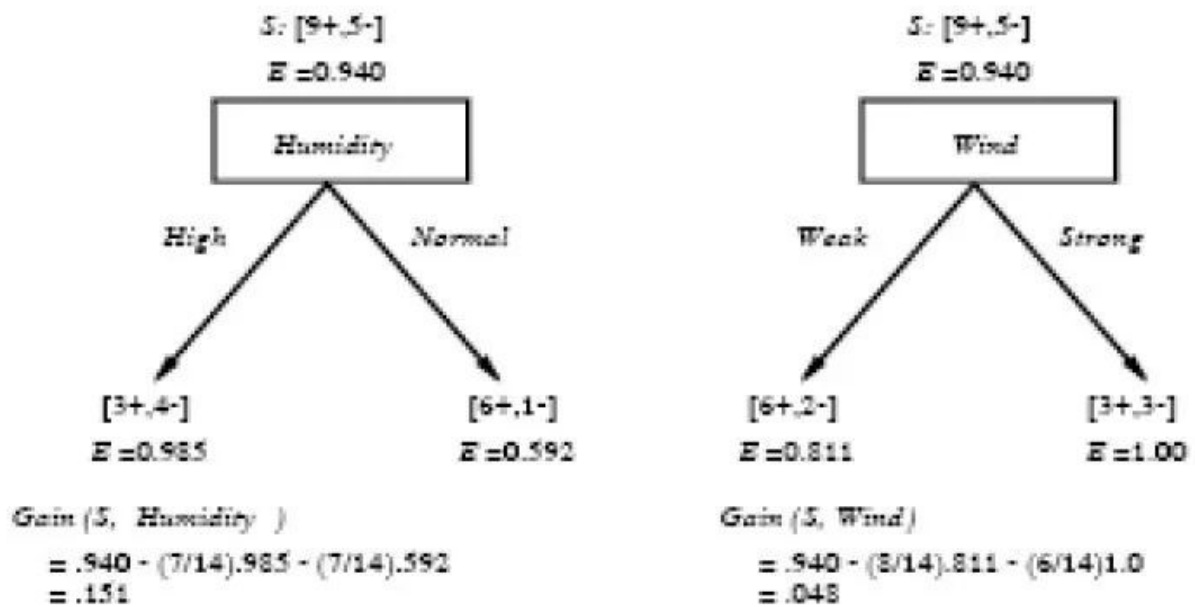
也就是说，决策树的简单策略就是，好比公司招聘面试过程中筛选一个人的简历，如果你的条件相当好比如说清华博士毕业，那么二话不说，直接叫过来面试，如果非重点大学毕业，但实际项目经验丰富，那么也要考虑叫过来面试一下，即所谓具体情况具体分析、决策。

第二个例子来自Tom M. Mitchell著的机器学习一书：

小王的目的是通过下周天气预报寻找什么时候人们会打高尔夫，他了解人们决定是否打球的原因最主要取决于天气情况。而天气状况有晴，云和雨；气温用华氏温度表示；相对湿度用百分比；还有有无风。如此，我们便可以构造一棵决策树，如下（根据天气这个分类决策这天是否合适打网球）：



上述决策树对应于以下表达式： $(\text{Outlook}=\text{Sunny} \wedge \text{Humidity} \leq 70) \vee (\text{Outlook} = \text{Overcast}) \vee (\text{Outlook}=\text{Rain} \wedge \text{Wind}=\text{Weak})$ 。得到的最佳分类属性如下图所示：



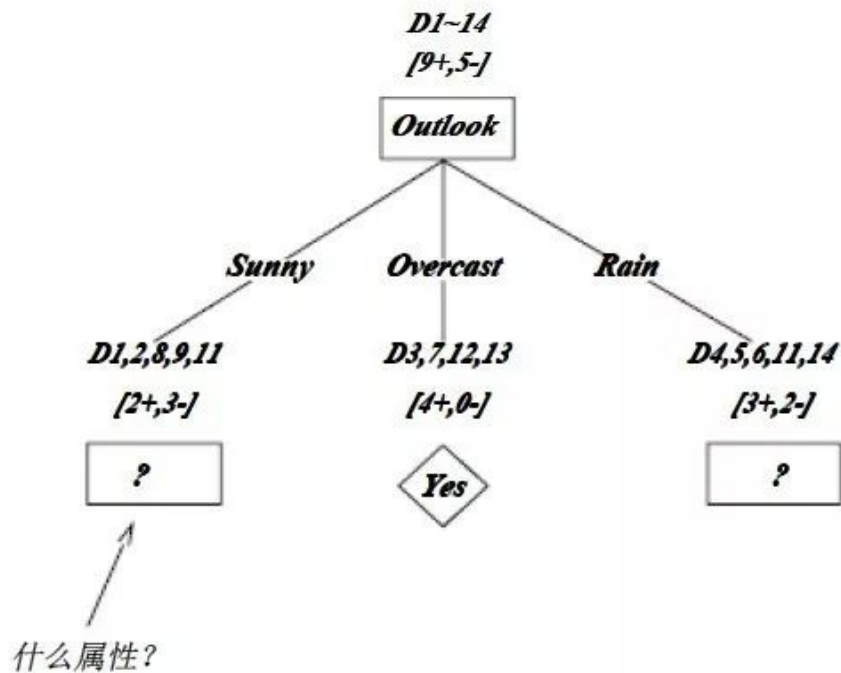
在上图中，计算了两个不同属性：湿度 (humidity) 和风力 (wind) 的信息增益，最终 humidity 这种分类的信息增益 $0.151 > \text{wind}$ 增益的 0.048 。说白了，就是在星期六上午是否适合打网球的问题决策中，采取 humidity 较 wind 作为分类属性更佳，决策树由此而来。

ID3 算法决策树的形成

OK，下图为 ID3 算法第一步后形成的部分决策树。这样综合起来看，就容易理解多了。

1、overcast 样例必为正，所以为叶子结点，总为 yes；

2、ID3 无回溯，局部最优，而非全局最优，还有另一种树后修剪决策树。下图是 ID3 算法第一步后形成的部分决策树：



6.2、贝叶斯分类的基础：贝叶斯定理

贝叶斯定理：已知某条件概率，如何得到两个事件交换后的概率，也就是在已知 $P(A|B)$ 的情况下如何求得 $P(B|A)$ 。这里先解释什么是条件概率：

表示事件B已经发生的前提下，事件A发生的概率，叫做事件B发生下事件A的条件概率。其基本求解公式为：

$$P(A|B) = \frac{P(AB)}{P(B)}$$

贝叶斯定理之所以有用，是因为我们在生活中经常遇到这种情况：我们可以很容易直接得出 $P(A|B)$ ， $P(B|A)$ 则很难直接得出，但我们更关心 $P(B|A)$ ，贝叶斯定理就为我们打通从 $P(A|B)$ 获得 $P(B|A)$ 的道路。

下面不加证明地直接给出贝叶斯定理（公式被网友指出有问题，待后续验证改正）：

$$\frac{(1, 0, 0) \cdot (1, 1, 0)}{\|(1, 0, 0)\| \|(1, 1, 0)\|} = \frac{1}{\sqrt{2}}$$

7

推荐实例扩展

7.1、阅读推荐

先来看一段文字（摘自36kr）：

”北京十分科技也非常看好阅读推荐类的应用，他们花了非常大的精力（一年60人团队），才在今天推出了iPhone 版“酷云阅读”。

为什么要投入这么多人去做这个阅读应用？CEO 李鹏告诉我，这个团队超过一半的人都在做后台相关的东西，包括语义分析、机器学习等算法。他们的目的是将互联网“语义化”以后，把人的兴趣明确，最后把每个人感兴趣的内容推荐给相关的人。在iPhone上，酷云的大致做法和Zite iPad 版类似，用户的行为也是有“喜欢”、“不喜欢”，以及点击相应的媒体来源或者相关的标签来告诉酷云你希望以后看到更多这些内容。

这个目的是大部分阅读推荐应用都有的，但是酷云的做法似乎更加变态。他们除了每天要抓取来自互联网的超过10万篇文章之外，还对全国200个的电视台播出的视频内容进行了索引，以便用户也可以通过文字搜索出视频、以及对视频内容进行一样的推荐。大致做法是先把这些节目都录制下来，然后把声音转文字，最后建立摘要和索引。“

一般的推荐系统应用的算法是有上文所述的什么协同过滤那般复杂呢？以下是援引自本人1月21日所发在微博上的文字：

1、大多数推荐阅读应用一般会给文章根据内容打上标签：算法，iphone（点击相当于为此标签加分加权重），并邀请对文章作出评价：喜欢，或不喜欢。每一次点击都被推荐系统记录了下来，最终渐渐形成用户的标签tag云（与此同时，还可基于相同或相似的标签tag寻找相似用户，从而基于用户推荐），而后系统每检索一篇新的文章，提取出文章的关键字，匹配用户的标签取向，进行推送。

2、目前手机上的新闻阅读做到了分类，如科技，教育，但一般不会采取如网页那般评分表态，所以也就无法记录用户的行为特征，也就不会有新的文章出来后后续的推荐阅读服务，于是造就了一批手机推荐阅读的问世，如 @酷云阅读 ，指阅等。

3、但一般用户的习惯是看完一段新闻便完事了，择日要看则择日看。例如有几个用户愿意为了评价一篇文章而特地去注册一个帐号呢？如何尽量让用户付出额外代价去使用这类阅读器，改变用户习惯，个人认为，是关键。

然后我还对上面的那句：先把这些视频节目都录制下来，然后把声音转文字有点疑问。我们已经知道如果是音乐的话像豆瓣FM可能是如下的做法：

1、你喜欢一些歌曲，而我也喜欢一些歌曲，如果你我喜欢的歌曲中有很多是重复类似的，则系统会把你我定义为好友，即相似用户，基于用户的协同过滤推荐：朋友喜欢，你也可能喜欢；

2、还有一个就是针对歌曲的推荐，你喜欢一首歌曲A，而另一首歌曲B与歌曲A类似（如都是有关爱情、感伤一类的），所以系统猜测你也可能喜欢B，而把B推荐给你。这就是基于项目（物品）的协同过滤推荐。

根据所听歌曲的重复类似判定为好友从而基于用户的协同过滤进行推荐，通过某些歌曲是差不多类似的来基于项目的协同过滤进行推荐，但问题出来了，重复的好说，同一首歌曲同一个歌手嘛，可那些相似音乐歌曲又如何定义判定呢？通过系统去分析歌曲的频谱？区别各个歌曲节奏的快慢，音频？此举虽然看起来有效，但实际实行起来不太现实。

我觉得应该是为那些音乐打上标签tag（估计视频也是这么做的，便于日后查找索引。全视频的实录目前觉得还是不靠谱），如打上“爱情”“感伤”一类的tag，而后tag相同的则可判定为相似歌曲。但关键是怎么打？语音识别？

7.2、标签tag怎么打

初期可以人肉，爬虫，买数据库，等流量上来了，可以考虑ugc。所谓ugc，用户产生内容。但是用户一般不太可能自己给音乐打标签，太繁琐了（如最近的新浪微博的每条微博内容下多了一个“加标签”的提示，但有多少用户愿去理它呢？），当然有的系统也会为你自动产生一些标签tag（当然，你也可以自行加上一些标签），如新浪博客：



如何做到的呢？我的想法是，

应该是系统在背后扫描你的文章一遍，然后提取一些关键词作为tag，供你选择。取哪些关键词呢？当然是取高频词。扫描整篇文章，统计每个单词出现的频率。

然后取其前TOP K，如上面截图中的“算法”在那篇文章中出现了4次，“博客”出现了3次，所以系统为你自动匹配这些标签。

至于采取何种数据结构或方法来统计这些关键词的频率呢。一般的应用hash+堆（十一、从头到尾彻底解析Hash表算法），或trie树（从Trie树谈到后缀树）均可。但当trie树面对的是汉字中文的时候，就比较麻烦了。所以hash+堆是比较理想的选择。

同样，针对视频的话，应该也是类似的：

1、通过系统或机器读取视频内容，把视频转换为文字，然后提取其中频率出现高的关键词（如何提取关键词呢，这就涉及到一个关键问题了：分词。本blog日后阐述），把提取出来的这些关键词作为此视频的标签tag；

2、然后针对这些tag建立索引摘要（什么样的索引？倒排索引。至于什么是倒排索引，参考编程艺术第二十四章：第二十三、四章：杨氏矩阵查找，倒排索引关键词Hash不重复编码实践），最终方便于日后用户或系统的查找（此节系与编程艺术内的朋友讨论整理总结而出）。

具体细节后续阐述。

8、参考文献

1、本人1月7日，1月21日的发表的微博（挂在本blog左侧边栏）；

2、探索推荐引擎内部的秘密，作者：赵晨婷，马春娥；

3、集体智慧编程，TobySeganra著。

4、推荐系统之协同过滤概述。

5、<http://www.cnblogs.com/leoo2sk/>。

6、Mitchell, Tom M. Machine Learning. McGraw-Hill, 1997（机器学习领域的开山之作）。

7 <http://zh.wikipedia.org/wiki/%E5%86%B3%E7%AD%96%E6%A0%91>。

8、<http://www.36kr.com/p/75415.html>。

9、智能web算法，第三章推荐系统（实现了用户及项目的相似度的计算，值得一看）

10 、 协 同 过 滤 推 荐 算 法 和 基 于 内 容 的 过 滤 算 法 ：
<https://time.geekbang.org/article/1947>

想要了解更多资讯，请扫描下方二维码，关注机器学习研究会



机器学习研究云订阅号

转自： 七月在线实验室