LSH----局部敏感哈希

2016年5月15日 星期日　　下午10:15

一、简介

局部敏感哈希算法（LSH，Locality Sensitive Hashing）。

是一种近似的最近邻搜索算法，可以快速检索相似样本。

思想是把各样本通过固定的哈希函数映射到不同的桶中，期望在相似的样本尽量在一个桶中；我们在检索阶段，给定一个待检索的样本，先根据hash函数定位所属的桶，再在该桶中遍历找出近邻的样本。达到缩小查找范围，提高检索效果的目的。

二、相关研究与名词定义

1、主流的索引技术

基于树的索引技术（tree-based index）

基于哈希的索引技术（hashing-based index）

基于词的倒排索引技术（visual words based inverted index）

2、哈希函数

下称hash function，或h（x）。哈希函数作用是根据样本生成键值，键值可以对应到桶中。

3、哈希表

下称hash table。样本与key的映射关系。

4、桶

bucket，键值相同的样本在一个桶中。即第i个桶中，$\forall h(x) = K_i$

三、思想

1、借助哈希思想

如何实现快速查询？O（1）的方法是哈希算法。就是将样本映射为一个键值，方便快速查找（复杂度O（1））。利用哈希函数进行映射，不同的样本键值可能一样，键值一样的不同样本构成桶（bucket）。样本与键值的映射关系成为哈希表。

如何实现快速索引？我们考虑是否能设计一种算法，利用哈希算法的效率，实现快速索引。
简单的说，快速索引其实就是，快速查找相似样本，快速查找最近邻或近似最近邻的样本。
传统的哈希技术有一个缺点：哈希函数构造的映射关系是XX的。
键值相同的样本的相关性可能并不大。

比如，1~10000中的数字，我们想给他映射到10个bucket中，我们认为数字接近代表样本相似，如98与99。我们采用的hash function是%10。我们能看出98和99相近但不在一个bucket中，99与1999不相近但是在一个桶中。这不是我们想要的。

我们希望找到一种哈希函数，使得相近的样本尽量在一个桶中，不相邻的不在。也就是希望哈希函数有“局部敏感”的性质，说白了就是对于98和99这样实际相似的样本是敏感的。

2、局部敏感哈希

思想是把各样本通过固定的哈希函数映射到不同的桶中，期望在相似的样本尽量在一个桶中；我们在检索阶段，给定一个待检索的样本，先根据hash函数定位所属的桶，再在该桶中遍历找出近邻的样本。达到缩小查找范围，提高检索效果的目的。

算法设计的关键点在于如何选定哈希函数。效果提升的关键点在于如何在false positive、false negtive之间折中。

四、算法（哈希函数选取与使用）

1、哈希函数的度量

什么样的哈希函数算作具有“局部敏感”，可以相似度。“相似”可以用连续值去衡量，“是否在一个bucket中”是一个0/1问题，我们得在连续和离散之间找到一个桥梁，那就是概率。简单说就是越相似越有可能在一个bucket中。表示成数学形式就是：

1）如果d(x,y) ≤ d1，则h(x) = h(y)的概率至少为p1；

2）如果d(x,y) ≥ d2，则h(x) = h(y)的概率至多为p2；

其中d(x,y)表示样本x和样本y之间的距离，d1 < d2，h(x)和h(y)分别表示对x和y进行hash变换（hash function），

h（x）、h（y）的返回值就是hash的键值，键值相同的在一个bucket中。

满足以上两个条件的hash functions称为(d1,d2,p1,p2)-sensitive。

而通过一个或多个(d1,d2,p1,p2)-sensitive的hash function对原始数据集合进行hashing生成一个或多个hash table的过程称为Locality-sensitive Hashing。

2、哈希函数与相似度衡量方式

从三，1、三，2中我们能看出，我们关心的具体是什么样的相似度，是从哪个角度衡量这个至为关键。如果我们认为98和99是相近的是一种哈希函数，我们认为1234与2234是相近的又是一种函数。根据不同的相似度，我们有不同的hash function，针对常见的相似度，常用的hash function如下：

A 杰卡德距离（Jaccard distance）

距离含义：

杰卡德相似度（Jaccard similarity）：两个集合（向

杰卡德相似度（Jaccard similarity）：两个集合（向量）交集元素个数，与并集元素个数只比。两个向量相同元素个数与所有元素个数只比。

Jaccard distance = 1 - Jaccard similarity

hash function：

min-hashing，其是(d1,d2,1-d1,1-d2)-sensitive的。

min-hashing定义：

n个m维的样本，构成一个m*n的特征矩阵（有m行、n列，值是0/1的），其中每个样本是一个列向量。

如图：s_i是样本

元素	S1	S2	S3	S4
他	0	0	1	0
成功	0	0	1	1
我	1	0	0	0
减肥	1	0	1	1
要	0	1	0	1

Min-hashing定义为：特征矩阵按行进行一个随机的排列后，第一个列值为1的行的行号。

假设上图已经是随机排序后的，那么最小哈希值：h(S1)=3，h(S2)=5，h(S3)=1，h(S4)=2。

理解：

为什么要这样用？

其实，min-hashing相同的概率和Jaccard相似度非常相关，有这样的关系：

P(h(Si)=h(Sj)) = sim(Si,Sj)

证明P(h(Si)=h(Sj)) = sim(Si,Sj)的过程，详见博客

http://www.cnblogs.com/maybe2030/p/4953039.html

什么会相等？我们考虑Si和Sj这两列，它们所在的行的所有可能结果可以分成如下三类：

（1）A类：两列的值都为1；

（2）B类：其中一列的值为0，另一列的值为1；

（3）C类：两列的值都为0.

特征矩阵相当稀疏，导致大部分的行都属于C类，但只有A、B类行的决定sim(Si,Sj)，假定A类行有a个，B类行有b个，那么sim(si,sj)=a/(a+b)。现在我们只需要证明对矩阵行进行随机排列，两个的最小hash值相等的概率P(h(Si)=h(Sj))=a/(a+b)，如果我们把C类行都删掉，那么第一行不是A类行就是B类行，如果第一行是A类行那么h(Si)=h(Sj)，因此P(h(Si)=h(Sj))=P(删掉C类行后，第一行为A类)=A类行的数目/所有行的数目=a/(a+b)，这就是最小hash的神奇之处。

B 汉明距离（hamming distance）

距离含义：两个相同长度向量中，有多少个位置的编码是不一样的。

hash function:

H_i(V) = V在第i位上的值，i是一个随机数。

理解：

由于汉明距离的计算方式就是根据编码相同/不同，编码中的一维也能从一定程度上说明各个向量V的相似性。

是(d1,d2,1-d1/d,1-d2/d)-sensitive的

C cos距离（cosine distance）

距离含义：向量之间的cos夹角

hash function：

H_i（V）= sign（V · R_i）R_i是一个随机向量，·是向量点乘（dot-product）

理解：

可以看做是把各个向量V，做同一个投影变换R_i，投影到一个一维的空间（也就是一条直线）上，成为了一个点。根据这个点在坐标原点0的左边还是右边分成了两份。

如果做多个H_i的话，相当于通过多种投影，投影到了

多个方向上再切分；相当于在空间中切了很多刀，每一刀是从一个随机的方向把空间切成了两份。

为什么适用于cos

cos是一种用dot-product加归一化的方式计算夹角（夹角，其实就是不考虑长度），sign（V·R_i）是由于sign()只关注正负，是不考虑V长度的（证明：V扩大n倍之后H_i（V）不变）。所以sign（V·R_i）也是一种基于dot-product的不考虑长度的衡量方式。可以理解为cos相近的两个向量，sign（V·R_i）也相近（也许能证明？？）

D 欧式距离（euclidean distance）

距离含义：向量各维度差值平方和的开方，各维度差值的二范数

hash function：

$H_i(V) = |V \cdot R_i + b_i| / a$，R是一个随机向量，a是桶宽，b是一个在[0,a]之间均匀分布的随机变量。其是(a/2,2a,1/2,1/3)-sensitive的。

理解：

把各个向量V投影之后，V·R可以看做是将V向R上进行投影操作，相当于把一条直线分为了多个长度为a的段（对应多个桶）

为什么适用于Euc-Dis

他是一种考虑长度的划分方式。如果把待衡量的两个向量看做是空间中的点（起点为原点，终点为改点的向量），使用欧氏距离的时候，这两个点应该在空间位置上是很接近的。在空间中两个接近的点如此投影之后应

该也是接近的。其实欧式距离的hash function比cos的好找。

3、使用方式

根据相似度衡量方式的要求，选定hash function family，也就是使用哪种hash function。

对一种hash function，选定k个具体的hash function，（由于选定hash function的过程需要随机的做选取），k个具体的hash function组合形成一条样本最终的hash function。

比如，hamming距离的相似度衡量中，k=3，分别是第5、1、8位的编码。那么最终的hash function就是“如果第5位、1位、8位的编码一直，则在一个桶中”。这就是最终的hash function。

这样就得到一个hash table，每个样本对应一个hash的key，这样的key就是满足局部敏感性的。

4、算法执行过程

1. 离线建立索引

（1）选取满足(d1,d2,p1,p2)-sensitive的LSH hash functions；

（2）根据对查找结果的准确率（即相邻的数据被查找到的概率）确定hash table的个数L，每个table内的hash functions的个数K，以及跟LSH hash function自身有关的参数；

（3）将所有数据经过LSH hash function哈希到相应的桶内，构成了一个或多个hash table；

2. 在线查找

（1）将查询数据经过LSH hash function哈希得到相应的桶号；

（2）将桶号中对应的数据取出；（为了保证查找速度，通常只需要取出前2L个数据即可）；

（3）计算查询数据与这2L个数据之间的相似度或距离，返回最近邻的数据；

五、LSH效果优化

同一个hash function family中，根据不同组随机变量，生成多个具体的hash function，组成一个hash table。在实际使用的过程中，为了增强LSH的效果，通常采用两种方式：1、在一个hash table中选择更多的hash function，2、选择多个hash table并进行组合。

hash function组合的方式如下：

A AND操作

几个hash function必须都满足键值相等，才可以

$\forall i, h\_i(x) = h\_i(y)$

B OR 操作

几个hash function只要有一个满足键值相等，就可以

$\exists i, h\_i(x) = h\_i(y)$

C AND OR 级联

六、参考资料

维基：https://en.wikipedia.org/wiki/Locality-sensitive_hashing

博客1：http://blog.csdn.net/icvpr/article/details/12342159

博客2：http://www.cnblogs.com/maybe2030/p/4953039.html