Recommendation Engine using MinHash

Lorenzo Fundaro

May 12, 2014

1 PROBLEM MODELING

1.1 DATA STRUCTURES

The problem of finding similarity between sets can be tackled using a technique called MinHash. For that purpose, it is common to model all the sets and the elements they have in a *characteristic matrix* shown as follows:

	S_1	S_2	...	S_n
e_1	1	0	0	1
e_2	0	0	1	1
...				
e_n	1	0	1	1

where $(e_i,S_i) = 1$ if S_i contains the element e_i and $(e_i,S_i) = 0$ otherwise. This matrix is naturally very *sparse*, so for this particular implementation I decided to use a map where the keys represent a *productId* and the correspondent value is a list of integers containing several *userId*'s. This decision would allow for easier traversal when calculating the signature matrix that MinHash requires to build.

1.2 HASHING FUNCTIONS

For the set of hashing functions I created a class called *Hash* under the package *utils*. This class is useful for generating a list of functions that are applied to every element of the *signature matrix*. The following function is used for hashing:

$$h = (a_i * x + b_i) \bmod 2147483647 \tag{1.1}$$

where a_i, b_i are coefficients generated randomly and the large number is a prime number.

2 Problems encountered during development

- **Complex traversal on data structure:** when applying the MinHash algorithm one has to make calculations with the signature matrix. For this calculations, it is necessary to traverse the list of products and for each product grab the list of users that have bought it and for each user hash its column in the signature matrix and save the changes for the next traversal. In functional programming side effects are not allowed, but in this case, to make things easier I decided to make the signature matrix a variable and not a value.
- **Hash function:** in order to get recommendations, the amount of hashing functions should be between 4 and 10, any number beyond that range would not allow for any hash collisions, thus giving no output. I still don't understand very well why I cannot use, e.g.: 100 functions. Choosing the right hashing function is still an open question for me.

3 Possible Improvements

Once we have calculated the signature matrix we can proceed on the suggestion of products for a given user. This calculation involves comparing the signature column of the user with the rest of the users and obtaining the similarity index. The majority of these calculations don't throw any results and a lot of CPU time is wasted for bigger datasets. By using a technique called Locality Sensitive Hashing (LSH) we can reduce the candidate sets considering only those signatures columns that when compared will give a positive result.

4 Resources

- Mining of Massive Datasets by Jure Leskovec, Anand Rajaraman and Jeff Ullman.
- Google News Personalization: Scalable Online Collaborative Filtering.
- Introduction to the MinHash algorithm