CS-7260 Internet Architecture and Protocols
Association Rule mining problem

Nov 19th 2012 ***Scribe by- Arvinder S Saini***

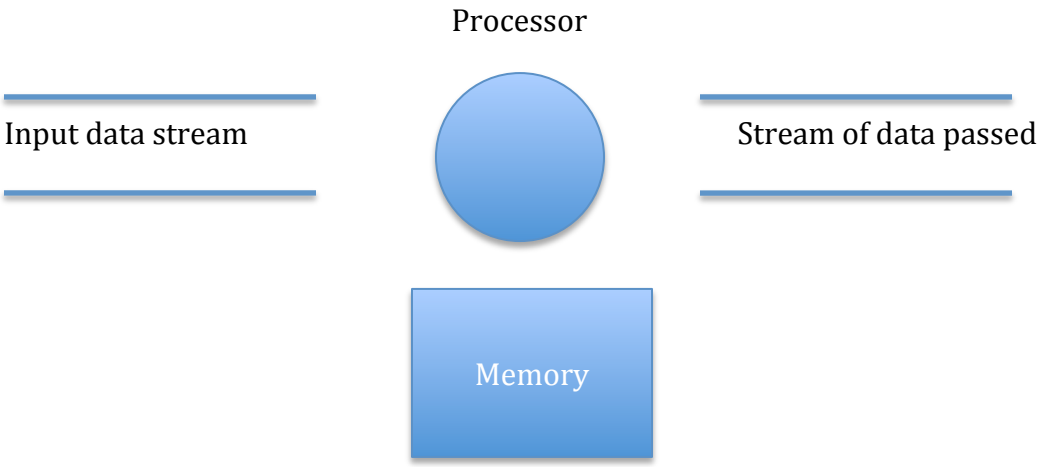
In this lecture we talked about the Association Rule mining problem. Suppose there is table that contains 100,000 columns and a billion rows with value either 0 or 1. The Association Rule problem corresponds to when we need to find the co-relation between any two columns in this table. This is the number of 1s in the bitwise AND of any two columns. There is a naïve brute force algorithm to do that as explained below. For instance consider this table.

–––––––100,000––––>

Transaction Id	Cereals	Milk	Cola		
312313oo	1	0	1	- - -	
321213ll	0	1	1	- - -	
3123123op	0	1	0		
.	.				
.	.				
.	.				
.	.				

The table contains a 100,000 columns and each column has one billion entries to it. Now we need to find a co-relation between two columns, Cereals and Milk. In order to find this, we need to perform operation AND on these two columns. With this naïve algorithm, imagine if we are required to find a co-relation between all the pairs of columns that are there. Considering that bitwise AND operation on 2 columns take 1 sec. The total time required for all the columns will be equal to $10C_2$ * 1 sec which is approximately equal to $5x10^9$ seconds.
There are $30x10^7$ seconds in a year which means that our total time required in our calculation would be $(5x10^9)/ 30x10^7$ seconds. Which is approximately equal to 160 years.

Now there is a classical problem in data streaming which is to find the number of distinct elements in the data stream. In network applications, we need to count this number very fast hence even the Hash table in the DRAM might not be enough to do this. However we can use a hash table in SRAM but this will turn out to be very expensive. Hence in this scenario the computational complexity is very important. So we need to design a smart algorithm that allows us to do that with the limited sized memory.



Consider the figure above. Imagine a stream of input data and with a small amount of memory and we are required to calculate the number of different packet passed. Each packet in the data stream is passed through the processor but the data can be accessed only once as we have only small amount of memory. (We are considering this case because usually we have data stream that is very large as compared to the size of memory used). We can process every packet and use the output to store some information about the packet. And after the entire stream has passed hopefully there will be enough information inside the memory that can allow us to find the number of distinct packets.

In order to solve this problem, consider a uniform has function 'h'. So basically in our example, 'h' is a function that maps the ids into a uniform random number between 0 and 1.

h: [ids] → uniform (0, 1)

We will try to use this hash function to define our algorithm for counting number of distinct elements. Lets define a variable called the minimum register (MR), which records the minimum value and it start with infinity. Now for each small data called datum we do,

```
foreach(datum){
        MR ← min{MR, h(datum)}
}
```

So, for every piece of data the algorithm hashes it and takes the minimum. So at the end of the stream, the expectation of MR is given by

Theorem: E[MR] = 1/ (1 + number of distinct elements)

Which means that if you have 7 distinct elements then there will be 7 distinct points between 0 and 1 and the value of expectation of each element will be equal to 1/8 as our hash function is uniform. If we use only one register then it will be very noisy. So suppose we can use 100 hash functions and 100 registers to store the min values. Then our algorithm will change to

```
foreach(datum){
        for(i = 1 to 100){
        MR_i ← min{MR_i, h_i(datum)}
        }
}
```

So, instead of getting one minimum value, we will get 100 minimum values and we can use these values with different theories (like mean of median, median of mean etc.) to get the number of distinct elements. By using this result we can actually solve the Association Rule mining problem.

In our example, instead of data stream, we have sets of data. What we need to find in our example is

$|S_1 \cap S_2| = |S_1| + |S_2| - |S_1 \cup S_2|$ --(equation 1)

by applying the above algorithm to each set we will get a vector containing 100 register values for each set S_1 (MR_1^1 , MR_2^1 … MR_{100}^1) and S_2(MR_1^2 , MR_2^2 … MR_{100}^2). The value of $|S_1 \cup S_2|$ in the equation above can be calculated by minimum values from two register vectors.

$\min\{MR_1^1, MR_1^2\}$
$\min\{MR_2^1, MR_2^2\}$
.
.
.
$\min\{MR_{100}^1, MR_{100}^2\}$

In our example, we process every column(Cereals, Milk etc.) and reduce it to 100 numbers like we just did and these 100 values corresponds to the number of 1s in each set. The result of bitwise AND of these two columns will be then given by equation 1. The time now required to compute the AND operation will be much less.