Balls, Bins and Random Graphs: Bloom Filters

1 Introduction

Let's say there was an Indian professor whose was very forgetful. He never remember about the student in his class. So when a student visit him he always remain in doubt whether she is in his class or not. So he thought of technique which will help him to get to know whether the student is in his class or not. He now keep an array of size of number of cities in the India. Now he asked the students of his class to give him the name of the 4 cities that they have first visited in their life. Then he uses a hash function that map the name of the cities to one of the boxes of the array. So he put one in all the boxes of array where the first 4 cities visited by his students are mapped by the hash function and zero in all the other boxes. So now he has an array of zeroes and ones. Now when a student visit him again after some years he can just ask him the first four cities that she has visited and hash them. If all the boxes of arrays where the cities are hashed are one then he will decide that she is his student otherwise not.

But there is some problem in this algorithm. There is possibility that even when a person come who is not his student the algorithm can say that she is his student. We will call such cases to be false positive. Next we will see what is probability of getting a false positive . Before that let's first see what is probability of a particular box being zero.

W.L.G we will see the probability of first box containing zero. Let's pick a student. Assume that number of cities in India is 360(also the size of the array) and the number of students in his class is 24. Now the probability that the first city that the student will tell will not hash to the first box will be $(1 - 1/360)$. Now for the sake of simplicity we will allow students to repeat the same cities i.e. a student can tell that he visited Mumbai, Chandigarh, Mumbai and Mumbai. So now the probability that none of those cities will hash to first box will be $(1 - 1/360)^4$. There are 24 students in his class. So the probability that none of the cities told by his student hashed to first box will be $(1 - 1/360)^{4*24}$. Let us denote by p. So expected number of zeroes will be p*360.

Next is to see the probability of false positive. We know that when we will hash a particular city it will assign him a value uniformly at random. Therefore when a person who is not his student tell the professor a name of the city the probability that it will contain 1 will be (1-p). Now he will tell 4 cities. So the probability that all the four cities will map to the box containing one will be $(1 - p)^4$ which is the probability of getting a false positive.

Next question that arises is how many cities should the professor ask for. It seems that more the number of cities more it is difficult to masquerade as a student. But as we if ask more cities array will also be more likely to be filled which makes its easier to masquerade as a student. So there is trade off here. Before going into detail of what could be the ideal number of cities let us first define the problem formally.

2 Formal Definition

Let S be a set such that $S = \{a_1, a_2, a_3, ..., a_m\}$where $a_1, a_2, a_3, ...a_m$ are the points in the S (you can see them as students). The problem statement is given an x, we have to tell whether $x \in S$ or not. Also we have k hash functions as $h_1, h_2, ..., h_k$ which will map a given number to a number from 1 to n. The algorithm will be as follows,

- Take an array A of size n.
- Assign $A[h_j(a_i)] = 1, \forall 1 \leq i \leq m and 1 \leq j \leq k$. (Here you can visualize $h_1(a), h_2(a), ...h_k(a)$ as hashed cities).
- Now for a new point x if $A[h_1(x)] = A[h_2(x)].... = A[h_k(x)] = 1$ then we will say x belongs to S otherwise not.

Here it can happen that even when x does not belong to S our algorithm will say that x belongs to S ie even when $x \ni S$, $A[h_1(x)], A[h_2(x)], ..., A[h_k(x)]$ are all equal to 1. Such cases are called false positives. The probability of such cases(let it call f) will be,

$f = (1-p)^k$, where p is the probability of a cell being zero

but $p = (1 - 1/n)^{mk} \approx \exp(-mk/n)$. Therefore f will become,

$f = (1 - \exp(-mk/n))^k$

Now let us see try to calculate the ideal k such that f is minimized. For this we will differentiate f with respect to k equate it to zero. But before that it is better to perform this differentiation on $\log(f)$, let say g, as the differentitation become easier that way and the value of k for minimum f will be same g. Now g is equal to $k \log(1 - \exp(-mk/n)$. Putting its differentiation w.r.t. k to be equal to zero we will get,

$$\frac{dg}{dk} = 0$$

$$\log(1 - \exp(-mk/n)) + k \frac{\exp(-mk/n}{1+\exp(-mk/n)} \frac{-m}{n} = 0$$

Solving above you will get k to be equal to $\ln(2) \frac{n}{m}$. Plugging this value into f it comes out to be $(0.618)^{\frac{n}{m}}$. The value of f will depend upon n and m. If n increases or m decreases the value f decreases. But this will result in increase k too which increase the number of hash functions which results in increasing the complexity of the algorithm.