Set similarity with k-permutation Minwise Hashing - NOTES

Simon Wehrli

11.5.2013

1 Introduction

Many today's applications are faced with very large datasets. A common task is to find *similarity* between two or several such sets.

This notes are intended as a help for the talk and explains only the core concepts of [1] and [2].

1.1 Similarity

We denote by Ω the set of all possible items of the sets $S_n \subseteq \Omega$, $n = 1$ to N. $|\Omega| = D$ is always large (e.g. $D = 2^{64}$). Often we consider only two sets $S_1 = X$, $S_2 = Y$. Let $a = |X \cap Y|$, $b = |X| - a$ and $c = |Y| - a$. Figure 1 summarizes these definitions graphically.

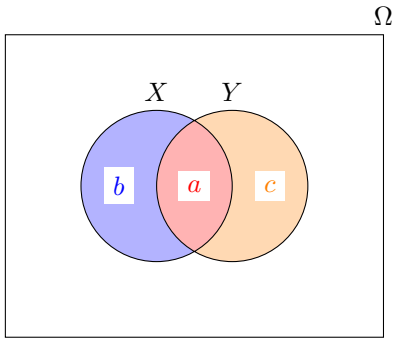

Figure 1: Two example sets in Ω and the notation a, b, c for the sizes of important subsets.

Definition 1. *The normalized similarity between two sets X and Y, known as* resemblance *or* Jaccard similarity*, denoted by R, is*

$$R = \frac{|X \cap Y|}{|X \cup Y|} = \frac{a}{|X| + |Y| - a} = \frac{a}{a + b + c} \tag{1}$$

Other Notations Later we often look at some important event of a problem where we use $\Pr[\cdot]$ as a shorthand for the *Probability* of that event. To get an estimator for R out of a probabilistic argument, we always use some Bernoulli experiment. This is a process, where we repeatedly flip a possibly biased coin, but the bias does not change. We can see it as a sequence of binary random variables, because only two outcomes are possible, 0 or 1. An event is described by an equation. We use the notation $1\{equation\}$ which is one if and only if the *equation* in curly braces is true:

$$1\{equation\} = \begin{cases} 1 & \text{if } equation \text{ evaluates to } true \\ 0 & \text{otherwise} \end{cases}$$

2 Original Minwise Hashing

Suppose a random permutation π is performed on Ω[1],

$$\pi : \Omega \longrightarrow \Omega, \qquad \text{where } \Omega = \{0, 1, \ldots, D-1\}.$$

To simplify notation, we overload the definition of π to work also for subsets of Ω. Thus with $\pi(S_n)$ we denote the application of the permutation π to every element of the set S_n. More precisely,

$$\pi : 2^{\Omega} \longrightarrow 2^{\Omega},$$
$$\pi : S_n \longmapsto \pi\left(S_n\right) = \left\{\pi\left(i\right) | i \in S_n\right\}.$$

We define

$$h_{S_n,\pi} = min(\pi(S_n))$$

to be the smallest element of set S_n permuted with π_j. Note that the smallest element changes under the permutations because π is a permutation on Ω and S_n is only a subset of Ω in general.

An elementary probability argument shows that for two sets $X, Y \subseteq \Omega$

$$\Pr[h_{X,\pi} = h_{Y,\pi}] = \Pr[\min(\pi(X)) = \min(\pi(Y))] = \frac{|X \cap Y|}{|X \cup Y|} = R. \tag{2}$$

We can now build an unbiased estimator $\hat{R}_M$ of R with k minwise independent permutations, $\pi_1, \pi_2, \ldots, \pi_k$:

$$\hat{R}_M = \frac{1}{k} \sum_{j=1}^{k} 1\left\{h_{X,\pi_j} = h_{Y,\pi_j}\right\}, \tag{3}$$

$$Var(\hat{R}_M) = \frac{1}{k} R(1-R). \tag{4}$$

[1]We assume there is a perfect hash function applied to the elements of the original domain which always gives us $\Omega = \{0, 1, \ldots, D-1\}$. Note that in the paper [1], the hash function is applied after the permutation and the minimum-function, but it is simpler to understand this way.

2.1 The Algorithm

Based on the theoretical results, Algorithm 1 presents the procedure of (k-permutation) Minwise Hashing.

Algorithm 1 Original Minwise Hashing algorithm, applied to estimating pairwise resemblances in a collection of N sets.

Input: Sets $S_n \subseteq \Omega = \{0, 1, \ldots, D-1\}, n = 1$ to N. ▷ $D = |\Omega|$
Output: Estimated resemblance $\hat{R}_M$

```
// Pre-processing
Generate k random permutations π_j : Ω ⟶ Ω, j = 1 to k
for all n = 1 to N, j = 1 to k do
    Store min(π_j(S_n)), denoted by h_{S_n,π_j}.
end for

// Estimation (Use two sets X, Y as an example)
Estimate the resemblance by R̂_M = (1/k) Σ_{j=1}^{k} 1{h_{X,π_j} = h_{Y,π_j}}
```

3 One Permutation Hashing

This algorithm is directly motivated by the optimization potential of the standard Minwise Hashing method: intuitively, it ought to be "wasteful" in that all elements in a set are permuted, scanned but only the minimum will be used. As the name already suggests, we reduce the preprocessing step to only one permutation.

We will setup a running example with $X, Y, Z \subseteq \Omega = \{0, 1, \cdots, 15\}$. Let be π some random permutation on Ω and the already permuted sets be

$$\pi(X) = \{2, 4, 7, 13\}, \quad \pi(Y) = \{0, 3, 6, 13\}, \quad \pi(Z) = \{0, 1, 10, 12\}.$$

Now again we build up a data matrix where the rows are equal to the vector representations of the permuted sets:

$$\begin{array}{r} \\ \text{Sets:} \\ \pi(X): \\ \pi(Y): \\ \pi(Z): \end{array} \overbrace{\begin{array}{cccccccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ \hline 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{array}}^{\Omega} \tag{5}$$

The idea is to divide the columns evenly into t (here $t = 4$) bins (parts), take the minimum in each bin. Because later we only compare minima within one bin, we can re-index the elements to use the smallest possible representation:

$$
\begin{array}{lc}
\text{Sets:} & \overbrace{0\ 1\ 2\ 3}^{\Omega'}\ \overbrace{0\ 1\ 2\ 3}^{\Omega'}\ \overbrace{0\ 1\ 2\ 3}^{\Omega'}\ \overbrace{0\ 1\ 2\ 3}^{\Omega'} \\
\pi(X): & \left(\begin{array}{cccc|cccc|cccc|cccc}
0 & 0 & \textcircled{1} & 0 & \textcircled{1} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 0 & 0 \\
\pi(Y): & \textcircled{1} & 0 & 0 & 1 & 0 & 0 & \textcircled{1} & 0 & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 0 & 0 \\
\pi(Z): & \textcircled{1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \textcircled{1} & 0 & \textcircled{1} & 0 & 0 & 0
\end{array}\right) \\
 & \text{bin 1} \quad\quad \text{bin 2} \quad\quad \text{bin 3} \quad\quad \text{bin 4}
\end{array}
\tag{6}
$$

We get the minima-vectors

$$
\begin{aligned}
v_X &= [2, 0, *, 1], \\
v_Y &= [0, 2, *, 1], \\
v_Z &= [0, *, 2, 0],
\end{aligned}
\tag{7}
$$

where '$*$' denotes an empty bin.

To derive the *resemblance* between two sets, e.g. X, Y, we introduce two definitions:

$$
\begin{aligned}
\text{number of "jointly empty bins":} \quad & N_{emp} = \sum_{j=1}^{t} I_{emp,j}, \\
\text{number of "matched bins":} \quad & N_{mat} = \sum_{j=1}^{t} I_{mat,j},
\end{aligned}
\tag{8}
$$

where $I_{emp,j}$ and $I_{mat,j}$ are defined for the j-th bin, as

$$
\begin{aligned}
I_{emp,j} &= \begin{cases} 1 & \text{if both } \pi(X) \text{ and } \pi(Y) \text{ are empty in the } j\text{-th bin} \\ 0 & \text{otherwise} \end{cases} \\
I_{mat,j} &= \begin{cases} 1 & \text{if both } \pi(X) \text{ and } \pi(Y) \text{ are not empty and the smallest} \\ & \text{ elements in the } j\text{-th bin matches, i.e. } (v_X)_j = (v_Y)_j \\ 0 & \text{otherwise} \end{cases}
\end{aligned}
\tag{9}
$$

3.1 The Estimator

We formulate the estimator as

Lemma 1. *The* resemblance *is estimated by*

$$
\hat{R}_{mat} = \frac{N_{mat}}{k - N_{emp}} \tag{10}
$$

is unbiased, i.e.

$$
\mathrm{E}\left[\hat{R}_{mat}\right] = R. \tag{11}
$$

In our example we have $N_{emp} = 1$ and $N_{mat} = 1$. Thus $\hat{R}_{mat} = 1/3$.

Empirical results show that the One Permutation Hashing scheme performs as well or even slightly better than the b-bit k-permutation hashing scheme.

Space used for ...	storing permutation(s)	storing pre-processing data p
Naïve approach (store whole sets)	—	$O(D)$
original Minwise Hashing	$O(kD\log(D))$	$O(k\log(D))$
b-bit Minwise Hashing	$O(kD\log(D))$	$O(kb)$
One Permutation Hashing	$O(D\log(D))$	$O(t\log(D/t))$

Table 1: Algorithm comparison. We only list the space complexity because the processing time boundaries are given by the time needed to read the stored data at least once and hence is dominated by the space complexity.

References

[1] Ping Li 0001 and Arnd Christian König. Theory and applications of b -bit minwise hashing. *Commun. ACM*, 54(8):101–109, 2011.

[2] Ping Li 0001, Art B. Owen, and Cun-Hui Zhang. One permutation hashing for efficient search and learning. *CoRR*, abs/1208.1259, 2012.