Approximative distance computation by random hashing

Selim Mimaroglu · Murat Yagci · Dan A. Simovici

Published online: 26 May 2011
© Springer Science+Business Media, LLC 2011

Abstract We propose an approximate computation technique for inter-object distances of binary data sets. Our approach is based on locality sensitive hashing. We randomly select a number of projections of the data set and group objects into buckets based on the hash values of these projections. For each pair of objects, occurrences in the same bucket are counted and the exact Hamming distance is approximated based on the number of co-occurrences in all buckets. We parallelize the computation using mainly two schemes. The first assigns each random subspace to a processor for calculating the local co-occurrence matrix, where all the local co-occurrence matrices are combined into the final co-occurrence matrix. The second method provides the same distance approximation in longer runtimes by limiting the total message size in a parallel computing environment, which is especially useful for very large data sets generating immense message traffic. Our methods produce very accurate results, scale up well with the number of objects, and tolerate processor failures. Experimental evaluations on supercomputers and workstations with several processors demonstrate the usefulness of our methods.

1 Introduction

Locality Sensitive Hashing (LSH), introduced in [1] and [2], can be used for an approximate calculation of distances between the tuples of a data set by using random-
ized hash functions. A close variant of LSH which works best with the Hamming distance is described in [3]. LSH is used for clustering the Web in [4]. In [5], it is used to enhance the agglomerative hierarchical clustering of the single link method [6]. Both of these techniques rely on the same idea provided by LSH: *Close objects are likely to collide under a high number of randomly chosen hashing functions.* Both of these techniques compute the real distances between objects residing in the same blocks. The clustering algorithms proposed in [4] and [5] focus on finding the approximate set of near neighbors $\text{ANN}(u)$ of an object $u$, followed by finding real near neighbors of $u$ by computing the actual distances $d(u, v)$ for all $v \in \text{ANN}(u)$. Note that some of the real neighbors of $u$ may be missed because LSH does not guarantee to put all the close objects in the same blocks.

We propose a method for approximating the distance matrix for data sets of bit vectors. The core idea is to randomly choose $mk$-dimensional subspaces and consider a bucket for each possible bit vector in this subspace. Then the vectors are hashed into the matching buckets and, for each pair of tuples, the occurrences in the same bucket are counted. The exact Hamming distance is approximated based on the portion of co-occurrences in the $m$ subspaces. Next, we parallelize the computation using two schemes. The first assigns each subspace to a single processor calculating its parts of the co-occurrence matrix and afterward adds up the complete co-occurrence matrix over all subspaces. The second method exchanges results between each processor during computation.

Our data set is a binary table $D$, having $N$ distinct tuples and a set $I$ that consists of $n$ distinct attributes. A set $K \subseteq I$ with $k$ attributes, designated as a *probe* and chosen randomly, defines a random hashing function $h_K$ by assigning to a tuple $t_j$ the numerical binary equivalent of the projection of $t_j$ on the set $K$, $t_j[K]$. Each hashing function produces a partition of the set of tuples; each block of this partition consists of tuples that collide under that hashing function.

Parallel and distributed computing techniques are able to solve big and complicated problems by using a variety of divide-and-conquer techniques. In this paper, we introduce several parallel data mining programming methodologies that are applicable in two widely used architectures: shared disk cluster environment, and shared memory architectures [7].

Preliminary results are presented in [8], and the paper is structured as follows. Section 2 examines the relation between randomly generated hash function collisions and distances. In Sect. 3, we present the algorithms and implementation guidelines. Experimental setup and test results are presented in Sect. 4. A final section contains our conclusions and future scope.

## 2 Collisions and distances

In this section, we examine the relation between randomly generated hash function collisions and inter-object distances.

### 2.1 Representation of hash function

*A binary data collection* is a sequence $D = (t_1, \ldots, t_N)$ of tuples, where $t_j \in \{0, 1\}^n$ and $n = |I|$, the cardinality of set of attributes.