Fast Information-Theoretic Agglomerative Co-Clustering

Tiantian Gao Leman Akoglu

Stony Brook University
Department of Computer Science
{tiagao,leman}@cs.stonybrook.edu

Preliminaries

Our algorithm iteratively merges those clusters whose merge yields a lower objective cost. However, operations such as finding nearest neighbors or closest pair of clusters are expensive, especially in high dimensions. To quickly find highly similar clusters to be merged, we exploit the Locality-Sensitive Hashing (LSH) technique, which we briefly describe in this section.

Simply put, LSH [2] is a randomized algorithm for similarity search among a given set of data points D. It uses a hash function h which ensures that similar points are hashed to the same entry (hash bucket) with high probability, while non-similar points are hashed to the same bucket with low probability. As a result, it quickens the similarity search by narrowing down the search to points that are hashed to the same bucket. In order to further reduce the probability of highly similar points hashing to different buckets, it uses multiple hash functions.

LSH uses different, suitable hash functions for similarity search with respect to different similarity functions. In this paper, we will use two: min-hashing [1] for Jaccard similarity and random-projection-based hashing [3] for cosine similarity. We next explain each one in more detail.

Min-hashing for Jaccard similarity.

Let data points in D consist of subsets of some global set of items I and the similarity function of interest be the Jaccard similarity $J(D_i, D_j) = \frac{|D_i \cap D_j|}{|D_i \cup D_j|}$. Let π be a random permutation on the indices of I. For a data point $D_i \subseteq I$, the min-hash value is defined as $v_\pi(D_i) = \min_{d_i \in D_i}\{\pi(d_i)\}$. Each random permutation π gives a hash value v_π, which maps every data point to an integer hash value. Given two data points $D_i, D_j \subseteq I$, it can be shown that the probability that their hash values agree is equal to their Jaccard similarity, i.e., $\Pr[v_\pi(D_i) = v_\pi(D_j)] = J(D_i, D_j)$.

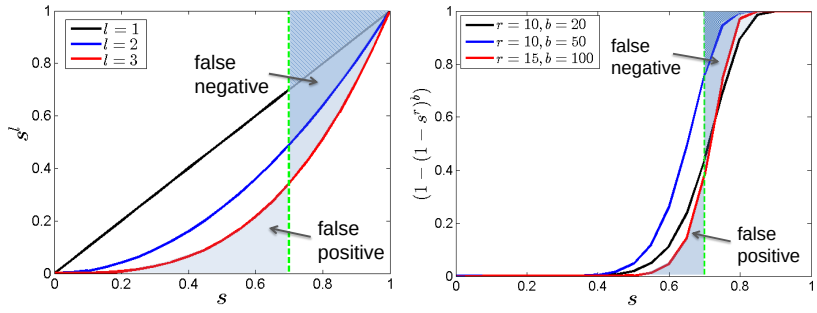
For a single hash value, the probability of two points hashing to the same bucket increases linearly with their similarity. In LSH, often a *signature* consisting of l min-hashes are generated for every data point, using l random permutations $\pi_1, \ldots, \pi_l$. This way, the probability two points matching on the same hash bucket becomes $J(D_i, D_j)^l$.

Random projections for cosine similarity.

Similarly, let data points in D be d dimensional vectors and the similarity function of interest be the cosine similarity $cos(D_i, D_j) = 1 - \theta(D_i, D_j)$, where θ denotes the angle between the two vectors. Let rnd be a random hyperplane. For a data vector $D_i \in \mathbb{R}^d$, the hash value is defined as $v_{rnd}(D_i) = sign(D_i \cdot rnd) = \pm 1$, depending on which side of the hyperplane D_i lies. Given two data points $D_i, D_j \in \mathbb{R}^d$, it can be shown that the probability that their hash values agree is closely related to their cosine similarity, in particular, $\Pr[v_{rnd}(D_i) = v_{rnd}(D_j)] = 1 - \frac{\theta(D_i, D_j)}{\pi}$. Using a number of random vectors, one can generate a signature of length l as before.

LSH parameters.

Let s denote the similarity (Jaccard or cosine) between two data points. With a signature of length l, probability that these two points will hash to the same bucket is proportional to s^l. This probability may be very small even for large $s \in [0, 1]$, i.e. two points may hash to different buckets even if they are sufficiently similar, which yields many false negatives. Even for small l, e.g. $l = 1$, the amount of false negatives might be undesirable (see Figure 1(a)).



(a) s vs. s^l (b) s vs. $1 - (1 - s^r)^b$

Fig. 1. Probability of two data points with similarity s hashing to the same bucket, using (a) single hash function, and (b) multiple hash functions. Shaded regions depict false negatives and false positives for threshold similarity $s = 0.7$ and for the parameters depicting the red curve.

To circumvent this issue, one can hash the signature of a data point *multiple* times. Specifically, $r < l$ elements from the original signature are randomly sampled to generate a sub-signature which is used for hashing. Sub-signature hashing is repeated b times. This results in b hash tables. This way the probability of two points to be hashed to the same bucket in *at least* one of the hash tables becomes $1 - (1 - s^r)^b$. Intuitively, this transforms the probability curve into an S-shape curve for which parameters r and b can be tuned to catch the most similar points and only a few non-similar points (see Figure 1(b)).

Number of sampled elements r:

The size of a (sub-)signature r determines how far (or non-similar) points can take the same hash value. For large r, only highly similar points are likely to hash to the same value (more false negatives), whereas for small r farther points can hash to the same value (more false positives).

Number of hash tables b:

By increasing the number of hash tables (or repetitions) b, sufficiently similar points could enter the same bucket for at least one hash table. This reduces the false negatives. On the other hand, for large b, farther points can also be contained in the same buckets (more false positives). Often, false positives are eliminated by measuring the real similarities of points in the same buckets. However, this increases the running time.

Overall, the LSH parameters r and b are used to adjust the trade-off between false positives (larger running time) and false negatives (smaller coverage).

References

1. A. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, pages 21–29, 1997.
2. A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.
3. W. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. volume 26 of *Contemporary Mathematics*, pages 189–206. 1984.