Locally Sensitive Hashing

Thomas Bonald
Institut Polytechnique de Paris

January 2024

Locally sensitive hashing (LSH) is an approach to searching **approximate nearest neighbors** in high dimension. The reader can consult the chapter 3 of the textbook[1] of Stanford course on Mining of Massive Datasets [2] for more details on LSH.

1 Principle

Let $\mathcal{H} = \{h : \mathbb{R}^d \to \{1, \dots, m\}\}$ be a set of hash functions.

The hash scheme $\mathcal{H}$ is said to be **locally sensitive** if there exist distances $d_1 < d_2$ and probabilities $p_1 > p_2$ such that for all $x, y \in \mathbb{R}^d$:

$$\begin{aligned} d(x,y) \le d_1 \quad &\Longrightarrow \quad \mathrm{P}(h(x) = h(y)) \ge p_1 \\ d(x,y) \ge d_2 \quad &\Longrightarrow \quad \mathrm{P}(h(x) = h(y)) \le p_2 \end{aligned}$$

where h is chosen **uniformly at random** in $\mathcal{H}$.

The idea is that close samples have likely the same hash value (or signature). Note that this property is satisfied whenever $\mathrm{P}(h(x) = h(y))$ decreases with $d(x, y)$.

The **concatenation** of locally sensitive hash functions provide locally sensitive hash functions.

If $\mathcal{H}$ is a locally sensitive hash scheme, then for any $N < \mathrm{card}(\mathcal{H})$, the hash scheme $\mathcal{H}' = \{(h_1, \dots, h_N) \in \mathcal{H}^N\}$ is locally sensitive.

To prove this result, consider $x, y \in \mathbb{R}^d$ and $(h_1, \dots, h_N) \in \mathcal{H}'$:

$$\mathrm{P}((h_1, \dots, h_N)(x) = (h_1, \dots, h_N)(y)) = \mathrm{P}(h_1(x) = h_1(y)) \dots \mathrm{P}(h_N(x) = h_N(y)) = \mathrm{P}(h(x) = h(y))^N.$$

2 Hash tables

For any hash function $h : \mathbb{R}^d \to \{1, \dots, m\}$, a hash table can be built to index a dataset $x_1, \dots, x_n \in \mathbb{R}^d$.

The hash table associated with the dataset $x_1, \dots, x_n \in \mathbb{R}^d$ is indexed by $j \in \{1, \dots, m\}$.
The bucket j contains all data x_i (or corresponding indices i) such that $h(x_i) = j$.

[1] The book is available online at `http://www.mmds.org`.

For searching the nearest neighbors of a target x, one looks at all data samples in bucket $j = h(x)$. If several hash tables are built (for different hash functions), the corresponding buckets can be considered in increasing order of size (the smaller buckets, the more specific the corresponding data samples).

3 Hash functions

Finally, we present some usual locally sensitive hash functions.

Bit sampling. For binary features, the simplest LSH scheme consists in looking at a single (random) bit.

> The Bit sampling scheme is $\mathcal{H} = \{h^{(1)}, \ldots, h^{(d)}\}$ with $h^{(j)}(x) = x_j \in \{0, 1\}$ for all $j = 1, \ldots, d$.

This hash scheme is locally sensitive for the **Hamming distance**, because for any $x, y \in \{0, 1\}^d$ and any hash function h chosen uniformly at random in $\mathcal{H}$,

$$P(h(x) = h(y)) = \frac{1}{d} \sum_{j=1}^{d} 1_{\{x_j = y_j\}} = 1 - \frac{d(x, y)}{d},$$

where $d(x, y)$ is the Hamming distance between x and y (number of distinct bits). Thus $P(h(x) = h(y))$ decreases with $d(x, y)$. By concatenation, we get a rich family of locally sensitive hash schemes.

MinHash. A popular LSH scheme is MinHash. For any permutation σ of the d features, we define:

$$h_\sigma(x) = \min_{j : x_j = 1} \sigma(j).$$

This is the rank of the first bit equal to 1 when the components of x are read in the order σ. Let S_d be the set of all permutations of $\{1, \ldots, d\}$.

> The MinHash scheme is $\mathcal{H} = \{h_\sigma, \sigma \in S_d\}$.

The MinHash scheme is locally sensitive for the **Jaccard distance**, because for any $x, y \in \{0, 1\}^d$,

$$P(h_\sigma(x) = h_\sigma(y)) = P(\min_{j : x_j = 1} \sigma(j) = \min_{j : y_j = 1} \sigma(j)) = \frac{\sum_{j=1}^{d} 1_{\{x_j = 1 \text{ and } y_j = 1\}}}{\sum_{j=1}^{d} 1_{\{x_j = 1 \text{ or } y_j = 1\}}} = s(x, y),$$

where $s(x, y)$ is the Jaccard similarity between x and y (fraction of equal features among expressed features).

A variant of MinHash is 1-bit MinHash, defined by the hash functions $h_\sigma \bmod 2$. This hash scheme is also locally sensitive for the Jaccard distance, since:

$$P(h_\sigma(x) = h_\sigma(y) \bmod 2) = P(h_\sigma(x) = h_\sigma(y)) + \frac{1}{2} P(h_\sigma(x) \neq h_\sigma(y)) = \frac{1 + s(x, y)}{2}.$$

Like bit sampling, these hash functions must be concatenated to form interesting hash schemes.

Sign random projection. For any vector $z \in \mathbb{R}^d$, let:

$$h_z(x) = 1_{\{z^T x > 0\}}.$$

If z is a **standard Gaussian** vector, we get a LSH scheme.

The Sign Random Projection scheme is $\mathcal{H} = \{h_z \text{ with } z \sim \mathcal{N}(0, I_d)\}$.

The Sign Random Projection scheme is locally sensitive for the **cosine similarity**, because for any $x, y \in \mathbb{R}^d$,

$$\mathrm{P}(h_z(x) = h_z(y)) = 1 - \frac{\widehat{xy}}{\pi},$$

where $\widehat{xy} \in [0, \pi]$ is the angle between x and y. In particular, $\mathrm{P}(h_z(x) = h_z(y))$ increases with the cosine similarity.

Concatenating N such hash functions gives efficient LSH schemes. This can be considered as the 2^N discretization of the vector space spanned by the N random vectors $z_1, \dots, z_N$. The fact that this random projection preserves the relative Euclidean distances between data samples for sufficiently large N (of order $\log n$ for n data samples) is known as the Johnson–Lindenstrauss lemma (see the Appendix).

Appendix

The Johnson–Lindenstrauss lemma

Let $z \sim \mathcal{N}(0, I_d)$ be some standard Gaussian vector. The projection over z tends to preserve Euclidean distances, in the sense that for any $x, y \in \mathbb{R}^d$,

$$(z^T x - z^T y)^2 = (x - y)^T z z^T (x - y).$$

Taking the expectation, we get:

$$\mathrm{E}((z^T x - z^T y)^2) = (x - y)^T \mathrm{E}(z z^T)(x - y) = ||x - y||^2,$$

showing that $(z^T x - z^T y)^2$ is an unbiased estimator of the square Euclidean distance between x and y.

Now consider N i.i.d. random vectors $z_1, \dots, z_N \sim \mathcal{N}(0, I_d)$. Then the projection over the vector space spanned by $z_1, \dots, z_N$ also preserves the relative Euclidean distances. Denoting by $Z = (z_1, \dots, z_N)$ the matrix formed by these vectors, we get:

$$||Z^T x - Z^T y||^2 = (x - y)^T Z Z^T (x - y) = (x - y)^T \left(\sum_{i=1}^N z_i z_i^T \right) (x - y),$$

so that

$$\mathrm{E}(||Z^T x - Z^T y||^2) = (x - y)^T \left(\sum_{i=1}^N \mathrm{E}(z_i z_i^T) \right) (x - y) = N ||x - y||^2.$$

The square Euclidean distances are preserved in expectation, up to the multiplicative constant N. The Johnson–Lindenstrauss lemma consist in bounding the deviation with respect to this expected value by concentration inequalities [3, 1]. In particular, it is shown that the relative Euclidean distances between n data samples $x_1, \dots, x_n$ are preserved up to some relative error ϵ provided N is of order $O\left(\frac{\log n}{\epsilon^2}\right)$.

References

[1] S. Dasgupta and A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 2003.

[2] J. Leskovec, A. Rajaraman, and J. D. Ullman. *Mining of massive data sets.* Cambridge University Press, 2020.

[3] W. Lindenstrauss and J. Johnson. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 1984.