

Machine Learning in High Dimension

IA317

Locally Sensitive Hashing

Thomas Bonald

2022 – 2023



Nearest neighbors

A set of **methods** for

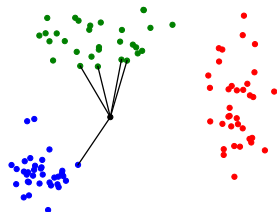
- ▶ Classification
- ▶ Regression
- ▶ Clustering
- ▶ Anomaly detection

Advantages

- ▶ Simple
- ▶ Explainable

Issues

- ▶ Parameter setting
- ▶ Complexity



Nearest neighbor search

Exact search:

- ▶ Exhaustive search $O(n)$
- ▶ Tree search $O(\log n) \rightarrow O(n)$

Approximate search:

- ▶ Locally sensitive hashing $O(1)$

Note: Both can be combined.

For instance, **exact** search of k nearest neighbors among K **approximate** nearest neighbors, with $k < K \ll n$

Outline

1. **Locally sensitive hashing**
2. Nearest neighbor search
3. Hash functions
4. Parameter setting

Locally Sensitive Hashing

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

Definition

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

$$d(x, y) \leq d_1 \implies \mathbb{P}(h(x) = h(y)) \geq p_1$$

$$d(x, y) \geq d_2 \implies \mathbb{P}(h(x) = h(y)) \leq p_2$$

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

Note: It is sufficient that $\mathbb{P}(h(x) = h(y))$ **decreases** with $d(x, y)$.

Example

For **binary** features, $x \in \{0, 1\}^d$.

Bit sampling

Hashing $\mathcal{H} = \{h^{(1)}, \dots, h^{(d)} : \mathbb{R}^d \rightarrow \{0, 1\}\}$ where

$$\forall i, \quad h^{(i)}(x) = x_i$$

Locally sensitive for the **Hamming distance**

$$\forall x, y \in \{0, 1\}^d, \quad P(h(x) = h(y)) = 1 - \frac{d(x, y)}{d}$$

Concatenation

Property

If \mathcal{H} is a locally sensitive hashing, then for any $N \geq 1$, \mathcal{H}^N is a locally sensitive hashing.

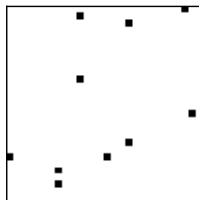
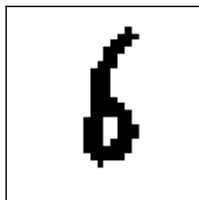
Proof: For any $x, y \in \mathbb{R}^d$:

$$\begin{aligned} P((h_1, \dots, h_N)(x) = (h_1, \dots, h_N)(y)) \\ &= P(h_1(x) = h_1(y)) \dots P(h_N(x) = h_N(y)) \\ &= P(h(x) = h(y))^N. \end{aligned}$$

Example

MNIST dataset (28×28 images, $d = 784$)

Selection of $N = 10$ random pixels



Hash table

Data $x_1, \dots, x_n \in \mathbb{R}^d$

Definition

Given some **hash function** $h : \mathbb{R}^d \rightarrow \{1, \dots, m\}$, table H with

$$H : j \rightarrow \text{set of samples } i \text{ such that } h(x_i) = j$$








for each entry $j \in \{1, \dots, m\}$.

Example

MNIST dataset (28×28 images, $d = 784$)

Selection of $N = 10$ random pixels



Signature	Bucket	Count
0001001000	 	2
0001100000		1
0000001000	   	4
...

Multiple hash tables

Data $x_1, \dots, x_n \in \mathbb{R}^d$

Principle

Given some hash functions h_1, \dots, h_L chosen **uniformly at random**, we build L hash tables:

$H_1 : j \rightarrow$ set of samples i such that $h_1(x_i) = j$

$H_2 : j \rightarrow$ set of samples i such that $h_2(x_i) = j$

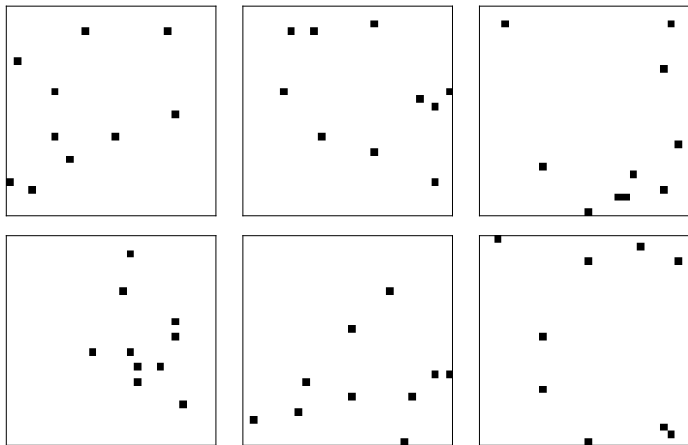
\vdots

$H_L : j \rightarrow$ set of samples i such that $h_L(x_i) = j$

Example

MNIST dataset (28×28 images, $d = 784$)

$L = 6$ hash tables, each based on $N = 10$ random pixels



Outline

1. Locally sensitive hashing
2. **Nearest neighbor search**
3. Hash functions
4. Parameter setting

Nearest neighbor search

Query $x \in \mathbb{R}^d$

Algorithm

- ▶ $j_1, \dots, j_L \leftarrow$ entry of x in each table H_1, \dots, H_L
- ▶ $S \leftarrow H_1(j_1) \cup \dots \cup H_L(j_L)$
- ▶ Return the k nearest neighbors of x in S

Example

MNIST dataset (28×28 images, $d = 784$)

$L = 3$ hash tables, each based on $N = 10$ random pixels














$x =$ 

Table	Signature of x	Bucket (samples)	Count
1	0000001110	   	2178
2	0001010000	   	42
3	0001001000	   	300

Opportunistic nearest neighbor search

Query $x \in \mathbb{R}^d$

Algorithm

Parameter = K , target search size

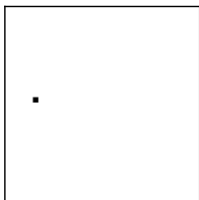
- ▶ $j_1, \dots, j_L \leftarrow$ entry of x in each table H_1, \dots, H_L
- ▶ $S \leftarrow \emptyset$
- ▶ Add buckets $H_1(j_1), \dots, H_L(j_L)$ to S
in **increasing order of size** until $|S| > K$
- ▶ Return the k nearest neighbors of x in S

Outline

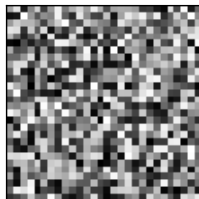
1. Locally sensitive hashing
2. Nearest neighbor search
3. **Hash functions**
4. Parameter setting

Hash functions

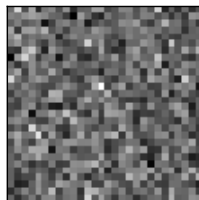
Bit sampling



MinHash



Random projection



MinHash (binary features)

Sample $x \in \{0, 1\}^d$

MinHash

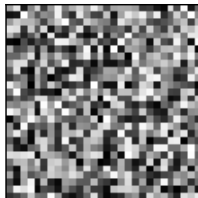
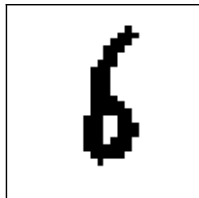
- ▶ Let σ be a **permutation** of $\{1, \dots, d\}$
- ▶ Hash function $h_\sigma : \{0, 1\}^d \rightarrow \{1, \dots, d\}$ defined by:

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

- ▶ Viewing x as a **set**, this is the rank of the **first element** of x , when the d items are read in the order given by σ .

Example

MNIST dataset (28×28 images, $d = 784$)



Analysis of MinHash

Proposition

For all $x, y \in \{0, 1\}^d$,

$$P(h_\sigma(x) = h_\sigma(y)) = s(x, y)$$

where $s(x, y)$ is **Jaccard similarity** between x and y

Locally sensitive for the **Jaccard distance**

1-bit MinHash

Parity bit of $h_\sigma(x) \in \{0, 1\}$

Proposition

For all $x, y \in \{0, 1\}^d$,

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

where $s(x, y)$ is **Jaccard similarity** between x and y

Random projection (numerical features)

Sample $x \in \mathbb{R}^d$

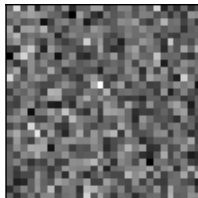
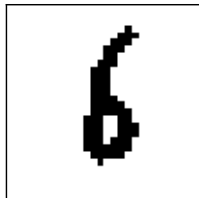
Random Projection

- ▶ Let z be some **standard gaussian** vector of dimension d
- ▶ Hash function $h_z : \mathbb{R}^d \rightarrow \{0, 1\}$ defined by:

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

Example

MNIST dataset (28×28 images, $d = 784$)



Analysis of random projection

Proposition

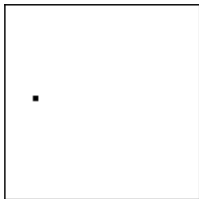
For all $x, y \in \mathbb{R}^d$,

$$P(h_z(x) = h_z(y)) = s(x, y)$$

where $s(x, y) = 1 - \frac{\widehat{xy}}{\pi}$ is the **angular similarity** between x and y , with $\widehat{xy} \in [0, \pi]$ the angle between x and y

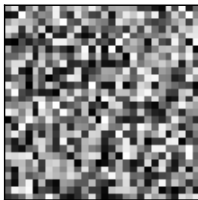
Summary

Random bit



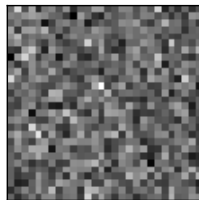
Binary features
Hamming distance

(1-bit) MinHash



Binary features
Jaccard similarity

Random projection



Numerical features
Angular similarity

Outline

1. Locally sensitive hashing
2. Nearest neighbor search
3. Hash functions
4. **Parameter setting**

Concatenation

MNIST dataset (28×28 images, $d = 784$)
10,000 samples

	1	2	5	10	20	100
Bit sampling	5400	3100	700	125	11	1
1-bit MinHash	5000	2500	315	17	2	1
Random projection	5000	2500	317	18	2	1

Table: Average bucket size for different values of N (= key length)

Number of hash tables

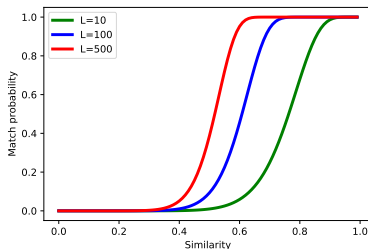
Consider two samples x, y with $P(h(x) = h(y)) = s$

Proposition

The matching probability in **at least one** Hash table H_1, \dots, H_L is

$$1 - (1 - s^N)^L$$

Example: Match probability w.r.t. s for $N = 10$



Summary

Locally sensitive hashing

- ▶ An approach to **approximate** nearest neighbor search through **hash tables**
Construction in $O(n)$
Search in $O(1)$
- ▶ Bit sampling / (1-bit) MinHash / Random projection
- ▶ Key **parameters**: key length N and number of tables L

