Machine Learning in High Dimension
IA317
Locally Sensitive Hashing

Thomas Bonald

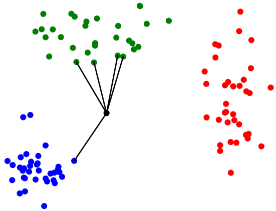
2023 – 2024

TELECOM Paris
IP PARIS

Nearest neighbors

A set of **methods** for

- Classification
- Regression
- Clustering
- Anomaly detection



Advantages

- Simple
- Explainable

Issues

- Choice of distance
- Complexity

Nearest neighbor search

Exact search:

- Exhaustive search $O(n)$
- Tree search $O(\log n) \to O(n)$

Nearest neighbor search

Exact search:

- Exhaustive search $O(n)$
- Tree search $O(\log n) \to O(n)$

Approximate search:

- Locally sensitive hashing $O(1)$

Nearest neighbor search

Exact search:

- Exhaustive search $O(n)$
- Tree search $O(\log n) \to O(n)$

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 $k < K << n$

Hash table

Outline

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

Locally Sensitive Hashing

Locally Sensitive Hashing

Let $\mathcal{H} = \{h : \mathbb{R}^d \to \{1, \dots, m\}\}$ 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 $x, y \in \mathbb{R}^d$:

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

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

Locally Sensitive Hashing

Let $\mathcal{H} = \{h : \mathbb{R}^d \to \{1, \dots, m\}\}$ 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 $x, y \in \mathbb{R}^d$:

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

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

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

Example

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

Example

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

Bit sampling

Hashing $\mathcal{H} = \{h^{(1)}, \ldots, h^{(d)} : \{0,1\}^d \to \{0,1\}\}$ where

$$\forall j = 1, \ldots, d, \quad h^{(j)}(x) = x_j$$

Example

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

Bit sampling

Hashing $\mathcal{H} = \{h^{(1)}, \ldots, h^{(d)} : \{0,1\}^d \to \{0,1\}\}$ where

$$\forall j = 1, \ldots, d, \quad h^{(j)}(x) = x_j$$

Locally sensitive for the **Hamming distance**

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

Concatenation

Concatenation

Property

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

Concatenation

Property

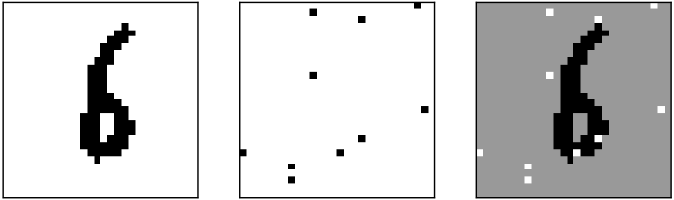
If $\mathcal{H}$ is a locally sensitive hashing, then for any $N < \text{card}(\mathcal{H})$, $\mathcal{H}' = \{(h_1, \ldots, h_N) \in \mathcal{H}^N\}$ is a locally sensitive hashing.

Proof: For any $x, y \in \mathbb{R}^d$ and $(h_1, \ldots, h_N) \in \mathcal{H}'$:

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

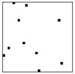
Example

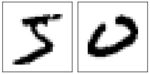
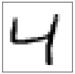
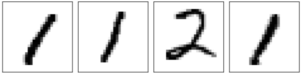
MNIST dataset in black & white (28×28 images, $d = 784$)
Selection of $N = 10$ random pixels



Application

MNIST dataset in black & white (28×28 images, $d = 784$)
Selection of $N = 10$ random pixels



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

Hash table

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

Definition

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

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

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

Hash table

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

Definition

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

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

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

Note: Use a **dictionary** in Python!

Multiple hash tables

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

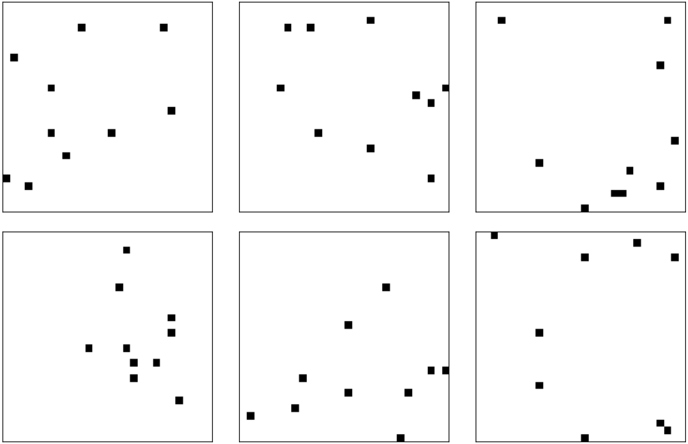
Principle

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

$$\begin{aligned}
H_1 &: j \to \text{set of samples } i \text{ such that } h_1(x_i) = j \\
H_2 &: j \to \text{set of samples } i \text{ such that } h_2(x_i) = j \\
&\vdots \\
H_L &: j \to \text{set of samples } i \text{ such that } h_L(x_i) = j
\end{aligned}$$

Example

MNIST dataset in black & white (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 $x \in \mathbb{R}^d$

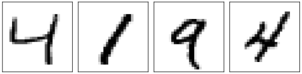
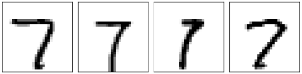
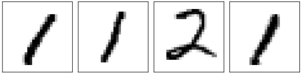
Algorithm

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

Example

MNIST dataset in black & white (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

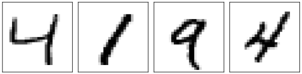
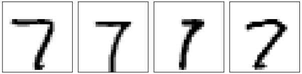
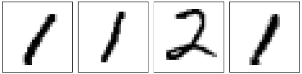
Parameter $= K$, target search size

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

Example

MNIST dataset in black & white (28×28 images, $d = 784$)
$L = 3$ hash tables, each based on $N = 10$ random pixels
5-nearest neighbor search, target $K = 20$

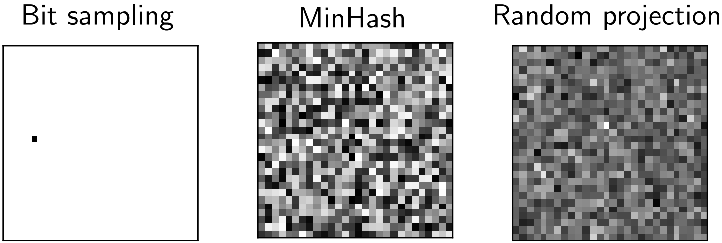
$x =$ 

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

Outline

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

Hash functions



MinHash (binary features)

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

MinHash (binary features)

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

MinHash

- Let σ be a **permutation** of $\{1,\dots,d\}$
- Hash function $h_\sigma : \{0,1\}^d \to \{1,\dots,d\}$ 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.

MinHash (binary features)

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

MinHash

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

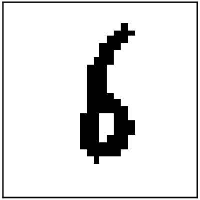
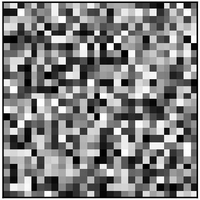
$$h_\sigma(x) = \min_{j : x_j = 1} \sigma(j)$$

- 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: For $d = 5$ and $\sigma : \{1,2,3,4,5\} \mapsto \{3,5,1,2,4\}$
For $x = (0,1,0,1,0)$, then $h_\sigma(x) = \min\{5,2\} = 2$.
Bits are read in the following order: 3, 4, 1, 5, 2.

Example

MNIST dataset in black & white (28×28 images, $d = 784$)

Analysis of MinHash

Proposition

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

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

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

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

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

Random projection (numerical features)

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

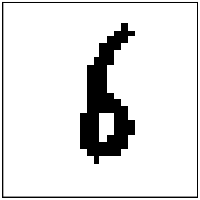
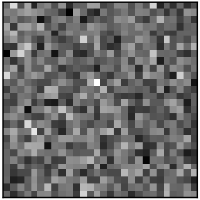
Sign Random Projection

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

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

Example

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

Analysis of sign random projection

Proposition

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

$$\mathrm{P}(h_z(x) = h_z(y)) = s(x, y)$$

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

Analysis of sign random projection

Proposition

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

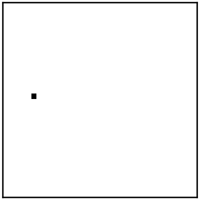
$$\mathrm{P}(h_z(x) = h_z(y)) = s(x, y)$$

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

Note: Locally sensitive for **cosine similarity**

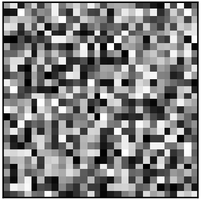
Summary

Bit sampling



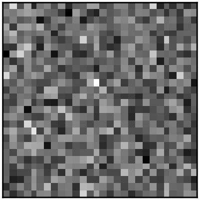
Binary features
Hamming distance

(1-bit) MinHash



Binary features
Jaccard similarity

Sign RandProj



Numerical features
Cosine similarity

Outline

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

Signature length

MNIST dataset (28×28 images, $d = 784$)
10,000 samples
Concatenation of N binary hash functions

N	1	2	5	10	20	100
Bit sampling	5400	3100	700	125	11	1
1-bit MinHash	5000	2500	315	17	2	1
Sign RandProj	5000	2500	317	18	2	1
Perfect split	5000	2500	312	10	1	1

Table: Average size of non-empty buckets with respect to N

Number of hash tables

Consider two samples x, y with $\mathrm{P}(h(x) = h(y)) = s$

Proposition

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

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

Number of hash tables

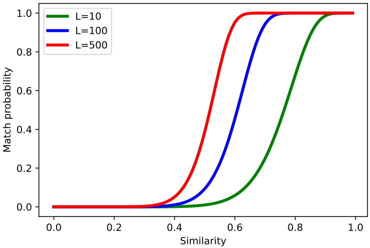
Consider two samples x, y with $\mathrm{P}(h(x) = h(y)) = s$

Proposition

The matching probability in **at least one** Hash table $H_1, \ldots, H_L$ 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 / Sign RandProj
- **Parameters**: signature length N and number of tables L