Locality-sensitive hashing without false negatives in ℓ_1

Negev Shekel-Nosatzki, Chen-Yu Yang, Parthiban Loganathan
Department of Computer Science
Columbia University

COMS 6998: Algorithmic Techniques for Massive Data

December 2015

Outline

1. Locality Sensitive Hashing Introduction
2. Our LSH construction
3. Conjecture
4. Future work

Locality Sensitive Hashing

Goal: A data structure to distinguish between close and far points with near-linear space and sub-linear query time. Can be used in Nearest Neighbor Search for example.

Definition (informal)

A *locality sensitive hash function, LSH* is a random hash function $h : \mathbb{R}^d \to U$ (h drawn from a family $\mathcal{H}$, U some finite set) such that

1. $d(q,p) \leq r \implies \Pr[h(q) = h(p)] = P_1$ is "not-so-small"
2. $d(q,p) > cr \implies \Pr[h(q) = h(p)] = P_2$ is "small"

Generally, $P_1 < P_2$ and we use n^ρ hash tables where:

$$\rho = \frac{\log(1/P_1)}{\log(1/P_2)}.$$

Locality Sensitive Hashing in Hamming

From class, LSH for Hamming Space $\{0,1\}^d$ with distance metric $(x, y) = |\{x_i \neq y_i\}|$.
The hash family, $\mathcal{H} : \{g : \{0,1\}^d \to \{0,1\}^k\}$, is defined as:

$$g(p) := (h_1(p), h_2(p), \ldots, h_k(p)),$$

where

$$h_i(p) := p_j \text{ for random } j \leftarrow [d].$$

For NNS, we maintain $L = n^\rho$ such Hamming-LSH tables g.

LSH from Pagh'16

- It's **covering** - guarantees collision for $\|x - y\| \leq r$. ie. Probability of false negatives is 0.
- Far points **collide with low probability**.
- Chooses **correlated hash functions** instead of selecting them independently to cover all ways in which two points can be r apart.
- **Performance is almost as good** as what we saw in class where we had constant probability of false negatives. The algorithm is efficient if $r = \Theta(\log n)$.

LSH from Pagh'16

- It's **covering** - guarantees collision for $\|x - y\| \leq r$. ie. Probability of false negatives is 0.
- Far points **collide with low probability**.
- Chooses **correlated hash functions** instead of selecting them independently to cover all ways in which two points can be r apart.
- **Performance is almost as good** as what we saw in class where we had constant probability of false negatives. The algorithm is efficient if $r = \Theta(\log n)$.

LSH from Pagh'16

- It's **covering** - guarantees collision for $\|x - y\| \leq r$. ie. Probability of false negatives is 0.
- Far points **collide with low probability**.
- Chooses **correlated hash functions** instead of selecting them independently to cover all ways in which two points can be r apart.
- **Performance is almost as good** as what we saw in class where we had constant probability of false negatives. The algorithm is efficient if $r = \Theta(\log n)$.

LSH from Pagh'16

- It's **covering** - guarantees collision for $\|x - y\| \leq r$. ie. Probability of false negatives is 0.
- Far points **collide with low probability**.
- Chooses **correlated hash functions** instead of selecting them independently to cover all ways in which two points can be r apart.
- **Performance is almost as good** as what we saw in class where we had constant probability of false negatives. The algorithm is efficient if $r = \Theta(\log n)$.

Pagh's LSH Definition

[Pagh'16] Given $S \subseteq \{0,1\}^d, c > 1$ and $r \in \mathbf{N}$ there exists a data structure with parameter ρ such that:

- On query $y \in \{0,1\}^d$ the data structure us guaranteed to return $x \in S$ with $\|x - y\| < cr$ if there exists $x' \in S$ with $\|x' - y\| \leq r$.
- The size is $O(dn^{1+\rho})$ bits where $n = |S|$.
- The expected query times is $O(n^\rho(1 + d/w))$, where w is the word length.
- Parameter ρ is a function of n, r and c bounded by $\rho \leq \min\left(\ln 4/c + \frac{\log r}{\log n}, 1/c + \frac{r}{\log n}\right)$. If $\log(n)/cr \in \mathbf{N}$, then $\rho = 1/c$.

Pagh LSH Construction

$\mathcal{H}_\mathcal{A} = \{x \mapsto x \wedge a | a \in \mathcal{A}\}$ where $\mathcal{A} \subseteq \{0,1\}^d$ is a set of bit masks. We go through all $h \in \mathcal{H}_\mathcal{A}$ and check for collisions.
$\mathcal{A}(m) = \{a(v) | v \in \{0,1\}^{r+1} \setminus \{\mathbf{0}\}\}$ where $a(v)_i = \langle m(i), v \rangle \; \forall$ $v \in \{0,1\}^{r+1}$ and $m : \{1, \cdots, d\} \to \{0,1\}^{r+1}$.
Pick random function m uniformly.
Example: this procedure produces the following covering set when $m(i)$ is the binary representation of i:

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

1 samples the coordinate. More efficient than having $\binom{d}{r}$ possible values for a.

ℓ_1 LSH w/o false negatives

Goal: Create an efficient Hashing that always collide close points and rarely collide far points

Plan:
- Embed $\ell 1$ into Hamming
- Hash Hamming

Embed: Assuming $x \in [M]^d$, we can embed x to $\{0,1\}^{Md}$ using unary embedding (as we saw in PS1) and hash that.

Eff.:
- $T = \mathcal{O}(M \cdot d \cdot n^\rho)$
- $S = \mathcal{O}(M \cdot d \cdot n^{1+\rho})$

Can we do better?

ℓ_1 LSH w/o false negatives

Goal: Create an efficient Hashing that always collide close points and rarely collide far points

Plan:
- Embed $\ell 1$ into Hamming
- Hash Hamming

Embed: Assuming $x \in [M]^d$, we can embed x to $\{0, 1\}^{Md}$ using unary embedding (as we saw in PS4) and hash that.

Eff.:
- $T = \mathcal{O}(M \cdot d \cdot n^\rho)$
- $S = \mathcal{O}(M \cdot d \cdot n^{1+\rho})$

Can we do better?

ℓ_1 LSH w/o false negatives

Goal: Create an efficient Hashing that always collide close points and rarely collide far points

Plan:
- Embed $\ell 1$ into Hamming
- Hash Hamming

Embed: Assuming $x \in [M]^d$, we can embed x to $\{0,1\}^{Md}$ using unary embedding (as we saw in PS4) and hash that.

Eff.:
- $T = \mathcal{O}(M \cdot d \cdot n^\rho)$
- $S = \mathcal{O}(M \cdot d \cdot n^{1+\rho})$

Can we do better?

ℓ_1 LSH w/o false negatives

Goal: Create an efficient Hashing that always collide close points and rarely collide far points

Plan:
- Embed $\ell 1$ into Hamming
- Hash Hamming

Embed: Assuming $x \in [M]^d$, we can embed x to $\{0,1\}^{Md}$ using unary embedding (as we saw in PS4) and hash that.

Eff.:
- $T = \mathcal{O}(M \cdot d \cdot n^\rho)$
- $S = \mathcal{O}(M \cdot d \cdot n^{1+\rho})$

Can we do better?

ℓ_1 LSH w/o false negatives

Goal: Create an efficient Hashing that always collide close points and rarely collide far points

Plan:
- Embed $\ell 1$ into Hamming
- Hash Hamming

Embed: Assuming $x \in [M]^d$, we can embed x to $\{0,1\}^{Md}$ using unary embedding (as we saw in PS4) and hash that.

Eff.:
- $T = \mathcal{O}(M \cdot d \cdot n^{\rho})$
- $S = \mathcal{O}(M \cdot d \cdot n^{1+\rho})$

Can we do better?

ℓ_1 LSH w/o false negatives

Goal: Create an efficient Hashing that always collide close points and rarely collide far points

Plan:
- Embed $\ell 1$ into Hamming
- Hash Hamming

Embed: Assuming $x \in [M]^d$, we can embed x to $\{0,1\}^{Md}$ using unary embedding (as we saw in PS4) and hash that.

Eff.:
- $T = \mathcal{O}(M \cdot d \cdot n^{\rho})$
- $S = \mathcal{O}(M \cdot d \cdot n^{1+\rho})$

Can we do better?

ℓ_1 Embedding Setup

- Can we use Sampling ?
  - Random Sampling will miss far points too often
  - Grid shifting can cut close points - exponential (in r or d) cost to avoid that prob.
- **Our Solution - Modulo Embedding**
  - Keep close points close
  - Contract distance of far points - but keep them far enough ($> cr$) with constant probability
  - Amplify probability by iterating and concatenating

ℓ_1 Embedding Setup

- Can we use Sampling ?
  - Random Sampling will miss far points too often
  - Grid shifting can cut close points - exponential (in r or d) cost to avoid that prob.
- **Our Solution - Modulo Embedding**
  - Keep close points close
  - Contract distance of far points - but keep them far enough ($> cr$) with constant probability
  - Amplify probability by iterating and concatenating

ℓ_1 Embedding Setup

- Can we use Sampling ?
  - Random Sampling will miss far points too often
  - Grid shifting can cut close points - exponential (in r or d) cost to avoid that prob.
- **Our Solution - Modulo Embedding**
  - Keep close points close
  - Contract distance of far points - but keep them far enough ($> cr$) with constant probability
  - Amplify probability by iterating and concatenating

Modulo Mapping

- Define $f_\alpha : \mathbb{N} \to \{0,1\}^\alpha$ to be the following mapping with parameter α:

$$f_\alpha(x) = \begin{cases} 0^{\alpha-\beta}1^\beta, & \text{if } \beta < \alpha \\ 1^{2\alpha-\beta}0^{\beta-\alpha}, & \text{otherwise} \end{cases}$$

  Where $\beta = x \pmod{2\alpha}$.
- Example with $\alpha = 5$:

$$\begin{array}{ll} \beta = 0 \to 00000 & \beta = 5 \to 11111 \\ \beta = 1 \to 00001 & \beta = 6 \to 11110 \\ \beta = 2 \to 00011 & \beta = 7 \to 11100 \\ \beta = 3 \to 00111 & \beta = 8 \to 11000 \\ \beta = 4 \to 01111 & \beta = 9 \to 10000 \end{array}$$

- **f_α preserves modulo distance !**
  $\|f_\alpha(x) - f_\alpha(y)\|_{Ham} = \|f_\alpha(|x-y|)\|_1$
  $= \min(x - y \pmod{2\alpha}, y - x \pmod{2\alpha})$

Modulo Mapping

- Define $f_\alpha : \mathbb{N} \to \{0,1\}^\alpha$ to be the following mapping with parameter α:

$$f_\alpha(x) = \begin{cases} 0^{\alpha-\beta}1^\beta, & \text{if } \beta < \alpha \\ 1^{2\alpha-\beta}0^{\beta-\alpha}, & \text{otherwise} \end{cases}$$

  Where $\beta = x \pmod{2\alpha}$.
- Example with $\alpha = 5$:

  $\beta = 0 \to 00000$ $\beta = 5 \to 11111$
  $\beta = 1 \to 00001$ $\beta = 6 \to 11110$
  $\beta = 2 \to 00011$ $\beta = 7 \to 11100$
  $\beta = 3 \to 00111$ $\beta = 8 \to 11000$
  $\beta = 4 \to 01111$ $\beta = 9 \to 10000$
- f_α **preserves modulo distance !**
  $\|f_\alpha(x) - f_\alpha(y)\|_{Ham} = \|f_\alpha(|x-y|)\|_1$
  $= \min(x - y \pmod{2\alpha}, y - x \pmod{2\alpha})$

$\ell 1 \rightarrow$ Hamming Modulo Embedding - Construction

- Define $T : [M]^d \rightarrow \{0,1\}^*$ as follows:
  - Randomly pick $k = log(n)$ integers $\alpha_1..\alpha_k$ from $[A, 2A]$, where $A = \max(5cr, \log(M))$
  - Run $f_{\alpha_j}(x_i)$ for each $i \in [d], j \in [k]$
  - Concatenate all strings
- Given input $x \in [M]^d$, we will hash $T(x)$ using Pagh with $r' = k \cdot r$

$\ell 1 \rightarrow$ Hamming Modulo Embedding - Analysis

- Final Dimension: $\mathcal{O}(d \cdot \log(n) \cdot (r + \log(M)))$ (but also now we are in bit-space, so representation and computation are $\log(M)$ faster).
  - $T = \mathcal{O}((r + \log(M)) \cdot \log(n) \cdot d \cdot n^{\rho})$
  - $S = \mathcal{O}((r + \log(M)) \cdot \log(n) \cdot d \cdot n^{1+\rho})$
- When $r = \Theta(log(n))$ (this is where Pagh focus on and when his algo is optimal and $\rho = 1/c$) - our expected query time is $T = \mathcal{O}(\text{polylog}(n, M) \cdot d \cdot n^{1/c})$ which is much faster than any existing deterministic $\ell 1$ NNS algorithm with near-linear space.

Conjecture behind Construction

- Conjecture: The modulo function keeps close points close all the time, and any far point will stay far on average. Specifically:

$$\forall x > 1, \forall A \geq \log(x), \alpha \sim \mathcal{U}[A, 2A]:$$

$$\mathbb{E}_\alpha[\min(x \mod 2\alpha, -x \mod 2\alpha)] \geq \min(x, \epsilon\alpha)$$

  For some constant ϵ
- As long as we sample $\alpha_i \geq 2cr/\epsilon$, then expectation for far coordinates is larger than $2cr$.
- Variance is bounded by range
- After $\log(n)$ times $O(1)$ far points will become close
- Cannot be 100% correct for any A (think about $x = LCM(\alpha_1 \dots \alpha_k) + 1$)
  - It's okay since $LCM(\alpha_1 \dots \alpha_k) > M$

Test of conjecture

- We run the following experiments
- **Completeness Test**
  - Test all $[2^{26}], K = 20, cr \in [4, 50]$.
  - No far points becomes close in $T = 10$ rounds for each cr.
- **Big Numbers Test**
  - Each round randomly sample 2^{20} numbers in $[2^{60}], K = 20, cr \in \{4, 10, 50, 100\}$.
  - No far points becomes close in $T = 100$ rounds for each cr.

What if r is big?

- If we need to hash with big r, we can replace r factor with d and pay constant growth in exponent and denominator (trade-off)
- Divide each coordinate by $\frac{r\Delta}{d}$ (for some constant $\Delta < c - 1$) and round them to the nearest integer.
- far point might contract to $\frac{cr}{1+\Delta}$, so we hash (using Pagh) with $c' = \frac{c}{1+\Delta}$.
- Final dimension becomes $\mathcal{O}(\log(n)(d^2 + d\log(M))/\Delta)$ and the time becomes $\approx n^{\rho+\Delta}$.

Acknowledgements

- Alex for helping us with our project direction.
- Pagh, Rasmus. "Locality-sensitive hashing without false negatives." arXiv preprint arXiv:1507.03225 (2015).

End

Thank you

Questions?