Similarity Estimation

Hu Fu @SHUFE, 2023.

Similarity Between Data Points

Similarity Between Data Points

- Similarity estimation arises in many big data applications

Similarity Between Data Points

- Similarity estimation arises in many big data applications
  - Find near-duplicate documents/webpages to remove redundancy

Similarity Between Data Points

- Similarity estimation arises in many big data applications
  - Find near-duplicate documents/webpages to remove redundancy
  - Detect pirated files

Similarity Between Data Points

- Similarity estimation arises in many big data applications
  - Find near-duplicate documents/webpages to remove redundancy
  - Detect pirated files
- How do we define and compute/estimate similarity?

Jaccard Similarity

Jaccard Similarity

- An article may be represented by the set of words it contains

Jaccard Similarity

- An article may be represented by the set of words it contains
  - Such a representation is high-dimensional but sparse

Jaccard Similarity

- An article may be represented by the set of words it contains
  - Such a representation is high-dimensional but sparse
- **Def.** The Jaccard similarity between two sets S and T is $\frac{|S \cap T|}{|S \cup T|}$, denoted as $\mathrm{SIM}(S, T)$.

Jaccard Similarity

- An article may be represented by the set of words it contains
  - Such a representation is high-dimensional but sparse
- **Def.** The Jaccard similarity between two sets S and T is $\dfrac{|S \cap T|}{|S \cup T|}$, denoted as $\mathrm{SIM}(S, T)$.
- We consider S, T very similar if $\mathrm{SIM}(S, T) \geq \alpha$ for some threshold α.

Jaccard Similarity

- An article may be represented by the set of words it contains
  - Such a representation is high-dimensional but sparse
- **Def.** The Jaccard similarity between two sets S and T is $\frac{|S \cap T|}{|S \cup T|}$, denoted as $\mathrm{SIM}(S, T)$.
- We consider S, T very similar if $\mathrm{SIM}(S, T) \geq \alpha$ for some threshold α.
- Is there a quick way to estimate $\mathrm{SIM}(S, T)$?

Min Hashing

Min Hashing

- Let the vocabulary be $[n] := \{1, \ldots, n\}$.

Min Hashing

- Let the vocabulary be $[n] := \{1,\ldots,n\}$.
- For a permutation σ on $[n]$ and $S \subseteq [n]$, define $\sigma_{\min}(S) := \min_{i \in S} \sigma(i)$

Min Hashing

- Let the vocabulary be $[n] := \{1,\ldots,n\}$.
- For a permutation σ on $[n]$ and $S \subseteq [n]$, define $\sigma_{\min}(S) := \min_{i \in S} \sigma(i)$

Lemma. For $S, T \subseteq [n]$, if σ is a random permutation of $[n]$, then $\mathbb{P}[\sigma_{\min}(S) = \sigma_{\min}(T)] = \dfrac{|S \cap T|}{|S \cup T|}$ = SIM(S, T)

Min Hashing

- Let the vocabulary be $[n] := \{1,\ldots,n\}$.
- For a permutation σ on $[n]$ and $S \subseteq [n]$, define $\sigma_{\min}(S) := \min_{i \in S} \sigma(i)$

Lemma. For $S, T \subseteq [n]$, if σ is a random permutation of $[n]$, then $\mathbb{P}[\sigma_{\min}(S) = \sigma_{\min}(T)] = \dfrac{|S \cap T|}{|S \cup T|} = \mathrm{SIM}(S, T)$

- A data structure that estimates Jaccard similarity:

Min Hashing

- Let the vocabulary be $[n] := \{1, \ldots, n\}$.
- For a permutation σ on $[n]$ and $S \subseteq [n]$, define $\sigma_{\min}(S) := \min_{i \in S} \sigma(i)$

<u>**Lemma.**</u> **For $S, T \subseteq [n]$, if σ is a random permutation of $[n]$, then $\mathbb{P}[\sigma_{\min}(S) = \sigma_{\min}(T)] = \frac{|S \cap T|}{|S \cup T|} =$ SIM(S, T)**

- A data structure that estimates Jaccard similarity:
    - Take ℓ random permutations $\sigma^1, \ldots, \sigma^\ell$ of $[n]$

Min Hashing

- Let the vocabulary be $[n] := \{1,\ldots,n\}$.
- For a permutation σ on $[n]$ and $S \subseteq [n]$, define $\sigma_{\min}(S) := \min_{i \in S} \sigma(i)$

Lemma. For $S, T \subseteq [n]$, if σ is a random permutation of $[n]$, then $\mathbb{P}[\sigma_{\min}(S) = \sigma_{\min}(T)] = \frac{|S \cap T|}{|S \cup T|} =$ SIM(S, T)

- A data structure that estimates Jaccard similarity:
  - Take ℓ random permutations $\sigma^1, \ldots, \sigma^\ell$ of $[n]$
  - For each record $S \subseteq [n]$, store $(\sigma^1_{\min}(S), \cdots, \sigma^\ell_{\min}(S))$

Min Hashing

- Let the vocabulary be $[n] := \{1,\ldots,n\}$.
- For a permutation σ on $[n]$ and $S \subseteq [n]$, define $\sigma_{\min}(S) := \min_{i \in S} \sigma(i)$

Lemma. For $S, T \subseteq [n]$, if σ is a random permutation of $[n]$, then $\mathbb{P}[\sigma_{\min}(S) = \sigma_{\min}(T)] = \frac{|S \cap T|}{|S \cup T|} =$ SIM(S, T)

- A data structure that estimates Jaccard similarity:
  - Take ℓ random permutations $\sigma^1, \ldots, \sigma^\ell$ of $[n]$
  - For each record $S \subseteq [n]$, store $(\sigma^1_{\min}(S), \cdots, \sigma^\ell_{\min}(S))$

To estimate SIM(S, T), output $\dfrac{|\{i : \sigma^i_{\min}(S) = \sigma^i_{\min}(T)\}|}{\ell}$

Min Hashing

- Let the vocabulary be $[n] := \{1,\ldots,n\}$.
- For a permutation σ on $[n]$ and $S \subseteq [n]$, define $\sigma_{\min}(S) := \min_{i \in S} \sigma(i)$

Lemma. For $S, T \subseteq [n]$, if σ is a random permutation of $[n]$, then $\mathbb{P}[\sigma_{\min}(S) = \sigma_{\min}(T)] = \frac{|S \cap T|}{|S \cup T|} = \mathrm{SIM}(S, T)$

- A data structure that estimates Jaccard similarity:
  - Take ℓ random permutations $\sigma^1, \ldots, \sigma^\ell$ of $[n]$
  - For each record $S \subseteq [n]$, store $(\sigma^1_{\min}(S), \cdots, \sigma^\ell_{\min}(S))$

To estimate $\mathrm{SIM}(S, T)$, output $\frac{|\{i : \sigma^i_{\min}(S) = \sigma^i_{\min}(T)\}|}{\ell}$

How large should ℓ be for $(1 \pm \epsilon)$-approximation w.p. $1 - \delta$?

Minwise Independent Permutations

Minwise Independent Permutations

- Storing random permutations and computing $\sigma_{\min}(S)$ is expensive

Minwise Independent Permutations

- Storing random permutations and computing $\sigma_{\min}(S)$ is expensive
- Recurring idea: use pseudo-randomness (e.g. from hash families)

Minwise Independent Permutations

- Storing random permutations and computing $\sigma_{\min}(S)$ is expensive
- Recurring idea: use pseudo-randomness (e.g. from hash families)
    - Let S_n be the set of permutations of $[n]$.

Minwise Independent Permutations

- Storing random permutations and computing $\sigma_{\min}(S)$ is expensive
- Recurring idea: use pseudo-randomness (e.g. from hash families)
  - Let S_n be the set of permutations of $[n]$.
  - Think of the definition of k-wise independent hash families

Minwise Independent Permutations

- Storing random permutations and computing $\sigma_{\min}(S)$ is expensive
- Recurring idea: use pseudo-randomness (e.g. from hash families)
  - Let S_n be the set of permutations of $[n]$.
  - Think of the definition of k-wise independent hash families

Def. [Broder, Charikar, Frieze, Mitzenmacher] A family $\mathscr{F} \subseteq S_n$ is a *minwise independent family* of permutations if for every $S \subseteq [n]$ and any $a \in S$, for σ sampled uniformly from $\mathscr{F}$, $\mathbb{P}[\sigma_{\min}(S) = \sigma(a)] = \frac{1}{|S|}$.

Minwise Independent Permutations

- Storing random permutations and computing $\sigma_{\min}(S)$ is expensive
- Recurring idea: use pseudo-randomness (e.g. from hash families)
  - Let S_n be the set of permutations of $[n]$.
  - Think of the definition of k-wise independent hash families

Def. [Broder, Charikar, Frieze, Mitzenmacher] A family $\mathcal{F} \subseteq S_n$ is a *minwise independent family* of permutations if for every $S \subseteq [n]$ and any $a \in S$, for σ sampled uniformly from $\mathcal{F}$, $\mathbb{P}[\sigma_{\min}(S) = \sigma(a)] = \dfrac{1}{|S|}$.

Minwise independent families suffice for estimation of Jaccard similarity.

Minwise Independent Permutations

- Storing random permutations and computing $\sigma_{\min}(S)$ is expensive
- Recurring idea: use pseudo-randomness (e.g. from hash families)
  - Let S_n be the set of permutations of $[n]$.
  - Think of the definition of k-wise independent hash families

<u>Def.</u> **[Broder, Charikar, Frieze, Mitzenmacher] A family $\mathscr{F} \subseteq S_n$ is a *minwise independent family* of permutations if for every $S \subseteq [n]$ and any $a \in S$, for σ sampled uniformly from $\mathscr{F}$, $\mathbb{P}[\sigma_{\min}(S) = \sigma(a)] = \dfrac{1}{|S|}$.**

Minwise independent families suffice for estimation of Jaccard similarity.

Minwise independent families of size 4^n exist ($\ll |S_n| = n!$)

Minwise Independent Permutations

- Storing random permutations and computing $\sigma_{\min}(S)$ is expensive
- Recurring idea: use pseudo-randomness (e.g. from hash families)
  - Let S_n be the set of permutations of $[n]$.
  - Think of the definition of k-wise independent hash families

<u>Def.</u> [Broder, Charikar, Frieze, Mitzenmacher] A family $\mathscr{F} \subseteq S_n$ is a *minwise independent family* of permutations if for every $S \subseteq [n]$ and any $a \in S$, for σ sampled uniformly from $\mathscr{F}$, $\mathbb{P}[\sigma_{\min}(S) = \sigma(a)] = \dfrac{1}{|S|}$.

Minwise independent families suffice for estimation of Jaccard similarity.

Minwise independent families of size 4^n exist ($\ll |S_n| = n!$)

Any minwise independent family has size $e^{(1-o(1))n}$

Relaxation

Def. [Broder, Charikar, Frieze, Mitzenmacher] A family $\mathscr{F} \subseteq S_n$ is a *minwise independent family* of permutations if for every $S \subseteq [n]$ and any $a \in S$, for σ sampled uniformly from $\mathscr{F}$, $\mathbb{P}[\sigma_{\min}(S) = \sigma(a)] = \frac{1}{|S|}$.

Def. [Broder, Charikar, Frieze, Mitzenmacher] A family $\mathscr{F} \subseteq S_n$ is a (ϵ, k) *minwise independent family* of permutations if for every $S \subseteq [n]$ with $|S| \leq k$ and any $a \in S$, for σ sampled uniformly from $\mathscr{F}$, $\frac{1-\epsilon}{|S|} \leq \mathbb{P}[\sigma_{\min}(S) = \sigma(a)] \leq \frac{1+\epsilon}{|S|}$.

Relaxation

<u>Def.</u> [Broder, Charikar, Frieze, Mitzenmacher] A family $\mathscr{F} \subseteq S_n$ is a *minwise independent family* of permutations if for every $S \subseteq [n]$ and any $a \in S$, for σ sampled uniformly from $\mathscr{F}$, $\mathbb{P}[\sigma_{\min}(S) = \sigma(a)] = \frac{1}{|S|}$.

<u>Def.</u> [Broder, Charikar, Frieze, Mitzenmacher] A family $\mathscr{F} \subseteq S_n$ is a (ϵ, k) *minwise independent family* of permutations if for every $S \subseteq [n]$ with $|S| \leq k$ and any $a \in S$, for σ sampled uniformly from $\mathscr{F}$, $\frac{1-\epsilon}{|S|} \leq \mathbb{P}[\sigma_{\min}(S) = \sigma(a)] \leq \frac{1+\epsilon}{|S|}$.

Thm. [Indyk] Let $\mathscr{H}$ be a t-wise independent hash family from $[n]$ to $[n]$, with $t = \Omega(\log \frac{1}{\epsilon})$, then $\mathscr{H}$ is a (ϵ, k) minwise independent family of permutations for $k = O(\epsilon n)$.

Relaxation

<u>Def.</u> [Broder, Charikar, Frieze, Mitzenmacher] A family $\mathscr{F} \subseteq S_n$ is a *minwise independent family* of permutations if for every $S \subseteq [n]$ and any $a \in S$, for σ sampled uniformly from $\mathscr{F}$, $\mathbb{P}[\sigma_{\min}(S) = \sigma(a)] = \frac{1}{|S|}$.

<u>Def.</u> [Broder, Charikar, Frieze, Mitzenmacher] A family $\mathscr{F} \subseteq S_n$ is a (ϵ, k) *minwise independent family* of permutations if for every $S \subseteq [n]$ with $|S| \leq k$ and any $a \in S$, for σ sampled uniformly from $\mathscr{F}$, $\frac{1-\epsilon}{|S|} \leq \mathbb{P}[\sigma_{\min}(S) = \sigma(a)] \leq \frac{1+\epsilon}{|S|}$.

Thm. [Indyk] Let $\mathscr{H}$ be a t-wise independent hash family from $[n]$ to $[n]$, with $t = \Omega(\log \frac{1}{\epsilon})$, then $\mathscr{H}$ is a (ϵ, k) minwise independent family of permutations for $k = O(\epsilon n)$.

How do you use minwise independent family to sample near-uniformly from the distinct elements in a streaming input?

Angular Distance

Angular Distance

- Two vectors in $\mathbb{R}^d$ may be considered similar if they point roughly the same direction

Angular Distance

- Two vectors in $\mathbb{R}^d$ may be considered similar if they point roughly the same direction
- For $u, v \in \mathbb{R}^d$, let $\mathrm{dist}(u, v) := \dfrac{\theta(u, v)}{\pi}$, where $\theta(u, v)$ is the angle between u and v

Angular Distance

- Two vectors in $\mathbb{R}^d$ may be considered similar if they point roughly the same direction
- For $u, v \in \mathbb{R}^d$, let $\text{dist}(u, v) := \dfrac{\theta(u, v)}{\pi}$, where $\theta(u, v)$ is the angle between u and v
  - If u, v are unit vectors, then $\cos(\theta(u, v)) = u \cdot v$

Angular Distance

- Two vectors in $\mathbb{R}^d$ may be considered similar if they point roughly the same direction
- For $u, v \in \mathbb{R}^d$, let $\mathrm{dist}(u, v) := \dfrac{\theta(u, v)}{\pi}$, where $\theta(u, v)$ is the angle between u and v
  - If u, v are unit vectors, then $\cos(\theta(u, v)) = u \cdot v$
- Similarity between u, v is $1 - \mathrm{dist}(u, v)$

Sim Hash

Sim Hash

- Can we estimate the similarity without computing the full inner product?

Sim Hash

- Can we estimate the similarity without computing the full inner product?
  - Say we have a data set $v_1, \ldots., v_n \in \mathbb{R}^d$

Sim Hash

- Can we estimate the similarity without computing the full inner product?
  - Say we have a data set $v_1, \ldots., v_n \in \mathbb{R}^d$
- Idea [Charikar]: use random projection!

Sim Hash

- Can we estimate the similarity without computing the full inner product?
  - Say we have a data set $v_1, \ldots, v_n \in \mathbb{R}^d$
- Idea [Charikar]: use random projection!
  - Pick random direction (unit vector) $r \in \mathbb{R}^d$

Sim Hash

- Can we estimate the similarity without computing the full inner product?
  - Say we have a data set $v_1, \ldots, v_n \in \mathbb{R}^d$
- Idea [Charikar]: use random projection!
  - Pick random direction (unit vector) $r \in \mathbb{R}^d$
  - For each v_i, let $h_r(v_i) := \text{sign}(r \cdot v_i)$

Sim Hash

- Can we estimate the similarity without computing the full inner product?
  - Say we have a data set $v_1, \ldots, v_n \in \mathbb{R}^d$
- Idea [Charikar]: use random projection!
  - Pick random direction (unit vector) $r \in \mathbb{R}^d$
  - For each v_i, let $h_r(v_i) := \mathrm{sign}(r \cdot v_i)$

Recall: how do we sample a random direction in $\mathbb{R}^d$?

Sim Hash

- Can we estimate the similarity without computing the full inner product?
    - Say we have a data set $v_1, \ldots., v_n \in \mathbb{R}^d$
- Idea [Charikar]: use random projection!
    - Pick random direction (unit vector) $r \in \mathbb{R}^d$
    - For each v_i, let $h_r(v_i) := \text{sign}(r \cdot v_i)$

Recall: how do we sample a random direction in $\mathbb{R}^d$?

Lemma. **For** $u, v \in \mathbb{R}^d$, $\mathbb{P}[h_r(u) = h_r(v)] = \dfrac{\theta(u,v)}{\pi}$

Sim Hash

- Can we estimate the similarity without computing the full inner product?
  - Say we have a data set $v_1, \ldots, v_n \in \mathbb{R}^d$
- Idea [Charikar]: use random projection!
  - Pick random direction (unit vector) $r \in \mathbb{R}^d$
  - For each v_i, let $h_r(v_i) := \text{sign}(r \cdot v_i)$

Recall: how do we sample a random direction in $\mathbb{R}^d$?

<u>Lemma.</u> For $u, v \in \mathbb{R}^d$, $\mathbb{P}[h_r(u) = h_r(v)] = \dfrac{\theta(u, v)}{\pi}$

Proof: The projection of r onto the plane spanned by u, v is again a random direction.

Sim Hash

- Can we estimate the similarity without computing the full inner product?
  - Say we have a data set $v_1, \ldots, v_n \in \mathbb{R}^d$
- Idea [Charikar]: use random projection!
  - Pick random direction (unit vector) $r \in \mathbb{R}^d$
  - For each v_i, let $h_r(v_i) := \text{sign}(r \cdot v_i)$

Recall: how do we sample a random direction in $\mathbb{R}^d$?

<u>Lemma.</u> For $u, v \in \mathbb{R}^d$, $\mathbb{P}[h_r(u) = h_r(v)] = \dfrac{\theta(u, v)}{\pi}$

Proof: The projection of r onto the plane spanned by u, v is again a random direction.

- SimHash: a data structure that estimates angular similarity:

Sim Hash

- Can we estimate the similarity without computing the full inner product?
  - Say we have a data set $v_1, \ldots., v_n \in \mathbb{R}^d$
- Idea [Charikar]: use random projection!
  - Pick random direction (unit vector) $r \in \mathbb{R}^d$
  - For each v_i, let $h_r(v_i) := \text{sign}(r \cdot v_i)$

Recall: how do we sample a random direction in $\mathbb{R}^d$?

<u>Lemma.</u> For $u, v \in \mathbb{R}^d$, $\mathbb{P}[h_r(u) = h_r(v)] = \dfrac{\theta(u,v)}{\pi}$

Proof: The projection of r onto the plane spanned by u, v is again a random direction.

- SimHash: a data structure that estimates angular similarity:
  - Take ℓ random directions $r^1, \ldots, r^\ell \in \mathbb{R}^d$

Sim Hash

- Can we estimate the similarity without computing the full inner product?
  - Say we have a data set $v_1, \ldots, v_n \in \mathbb{R}^d$
- Idea [Charikar]: use random projection!
  - Pick random direction (unit vector) $r \in \mathbb{R}^d$
  - For each v_i, let $h_r(v_i) := \text{sign}(r \cdot v_i)$

Recall: how do we sample a random direction in $\mathbb{R}^d$?

Lemma. **For** $u, v \in \mathbb{R}^d$, $\mathbb{P}[h_r(u) = h_r(v)] = \dfrac{\theta(u, v)}{\pi}$

Proof: The projection of r onto the plane spanned by u, v is again a random direction.

- SimHash: a data structure that estimates angular similarity:
  - Take ℓ random directions $r^1, \ldots, r^\ell \in \mathbb{R}^d$
  - For each record v_i, store the ℓ-tuple $(h_{r_1}(v_i), \cdots, h_{r_\ell}(v_i))$

SimHash was showcased in this popular book by Jun Wu