NYU | Center for Data Science

Week 08.2: Spatial data structures

DS-GA 1004: Big Data

This week

- Search and MinHash
- **Spatial data structures**

Can we improve the efficiency of MinHash?

Locality sensitive hashing

[Indyk and Motwani, 1998]
[Charikar 2002]

LSH

- Traditional hash functions scatter data "**randomly**"
    - Probability of collision is **independent of similarity** between inputs

- **Locality-sensitive hashes** have high probability of collision for inputs that are ***near each other***

- LSH has a huge literature, this intro will be superficial

LSH + MinHash

- Carve signature matrix into ***b*** blocks of *R* rows each
- Hash each sub-column with a standard (non-local) hash function ***W***
  - Pick ***W*** such that collisions are rare for non-identical inputs
- Candidate set = items that collide in *any* row block

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0
H_3	1	2	0	1
H_4	0	1	0	0
H_5	2	2	0	0
H_6	1	2	1	1

	A	B	C	D
Block 1	0	5	3	0
Block 2	3	0	1	3
Block 3	0	7	2	2

$\boldsymbol{W}([0, 1]) \rightarrow 2$

LSH + MinHash

- Carve signature matrix into ***b*** blocks of *R* rows each
- Hash each su … unction ***W***
  - Pick ***W*** such …
- Candidate set = items that collide in *any* row block

What's the probability that we have at least one block where all rows match?

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0
H_3	1	2	0	1
H_4	0	1	0	0
H_5	2	2	0	0
H_6	1	2	1	1

	A	B	C	D
Block 1	0	5	3	0
Block 2	3	0	1	3
Block 3	0	7	2	2

$W([0, 1]) \rightarrow 2$

LSH vs direct MinHash analysis

- MinHash:
  - P[single row collision] = J(A, B) = j
- LSH:
  - P[all R rows in a block collide] = j^R

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0
H_3	1	2	0	1
H_4	0	1	0	0
H_5	2	2	0	0
H_6	1	2	1	1

	A	B	C	D
Block 1	0	5	3	0
Block 2	3	0	1	3
Block 3	0	7	2	2

LSH vs direct MinHash analysis

- MinHash:
  - P[single row collision] = J(A, B) = j
- LSH:
  - P[all R rows in a block collide] = j^R
  - P[at least one **non-collision** in a block] = 1 - j^R

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0
H_3	1	2	0	1
H_4	0	1	0	0
H_5	2	2	0	0
H_6	1	2	1	1

	A	B	C	D
Block 1	0	5	3	0
Block 2	3	0	1	3
Block 3	0	7	2	2

LSH vs direct MinHash analysis

- MinHash:
  - P[single row collision] = J(A, B) = j
- LSH:
  - P[all R rows in a block collide] = j^R
  - P[at least one **non-collision** in a block] = 1 - j^R
  - P[at least one non-collision in all b blocks] = $(1 - j^R)^b$

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0
H_3	1	2	0	1
H_4	0	1	0	0
H_5	2	2	0	0
H_6	1	2	1	1

	A	B	C	D
Block 1	0	5	3	0
Block 2	3	0	1	3
Block 3	0	7	2	2

LSH vs direct MinHash analysis

- MinHash:
  - P[single row collision] = J(A, B) = j
- LSH:
  - P[all R rows in a block collide] = j^R
  - P[at least one **non-collision** in a block] = 1 - j^R
  - P[at least one non-collision in all b blocks] = $(1 - j^R)^b$
  - P[at least one block **collides on all rows**] = $1 - (1 - j^R)^b$

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0
H_3	1	2	0	1
H_4	0	1	0	0
H_5	2	2	0	0
H_6	1	2	1	1

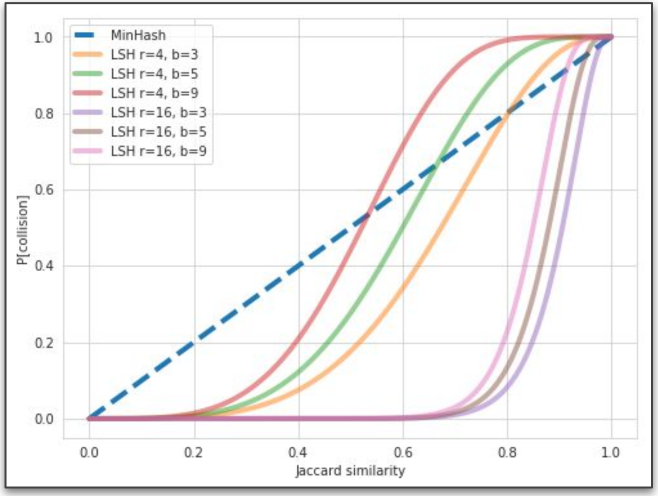
	A	B	C	D
Block 1	0	5	3	0
Block 2	3	0	1	3
Block 3	0	7	2	2

LSH vs direct MinHash analysis

- MinHash:
  - P[single row collision] = J(A, B) = j
- LSH:
  - P[all R rows in a block collide] = j^R
  - P[at least one **non-collision** in a block] = 1 - j^R
  - P[at least one non-collision in all b blocks] = $(1 - j^R)^b$
  - P[at least one block **collides on all rows**] = $1 - (1 - j^R)^b$

	A	B	C	D
H_1	0	0	1	0
H_2	0	1	2	0
H_3	1	2	0	1
H_4	0	1	0	0
H_5	2	2	0	0
H_6	1	2	1	1

	A	B	C	D
Block 1	0	5	3	0
Block 2	3	0	1	3
Block 3	0	7	2	2



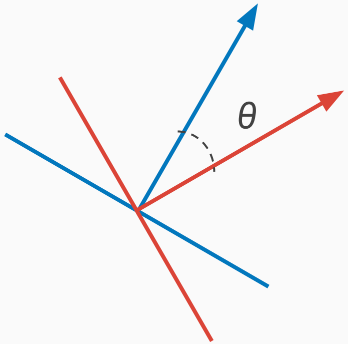
Result:
Collisions are **more likely** for high Jaccard similarity and **less likely** for low Jaccard similarity

LSH for cosine similarity

[Charikar 2002]

- What if we want to compare vectors u, $v \in \mathbf{R}^d$ by cosine similarity?

$$\mathrm{sim}(u, v) = \cos(\theta)$$

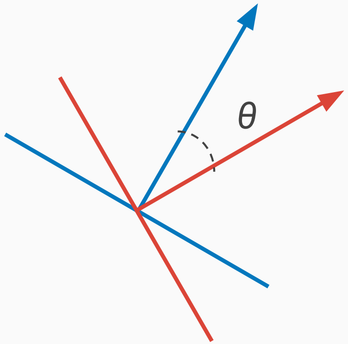


LSH for cosine similarity
[Charikar 2002]

- What if we want to compare vectors $u, v \in \mathbf{R}^d$ by cosine similarity?

$$\mathrm{sim}(u, v) = \cos(\theta)$$

- Pick a vector w uniformly from the sphere in $\mathbf{R}^d$
  - $h_w(x) = 1$ if $w^T x >= 0$
    $= 0$ if $w^T x < 0$



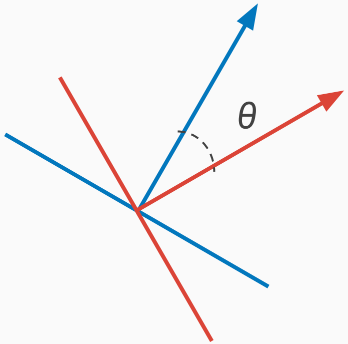
LSH for cosine similarity

[Charikar 2002]

- What if we want to compare vectors $u, v \in \mathbf{R}^d$ by cosine similarity?

  $$\mathrm{sim}(u, v) = \cos(\theta)$$

- Pick a vector w uniformly from the sphere in $\mathbf{R}^d$
  - $h_w(x) = 1$ if $w^T x >= 0$
    $= 0$ if $w^T x < 0$



- What's the probability of collision?

  - $\mathbf{P}[h_w(u) = h_w(v)] \quad = 1 - \mathbf{P}[h_w(u) \neq h_w(v)]$

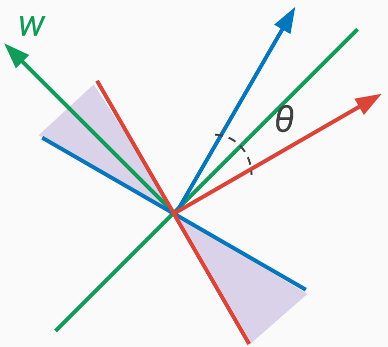
LSH for cosine similarity

[Charikar 2002]

- What if we want to compare vectors u, $v \in \mathbf{R}^d$ by cosine similarity?

$$\mathrm{sim}(u, v) = \cos(\theta)$$

- Pick a vector w uniformly from the sphere in $\mathbf{R}^d$
  - $h_w(x) = 1$ if $w^T x >= 0$
    $= 0$ if $w^T x < 0$



- What's the probability of collision?
  - $\mathbf{P}[h_w(u) = h_w(v)] = 1 - \mathbf{P}[h_w(u) \neq h_w(v)]$
    $= 1 - \mathbf{P}[w$ in **shaded region**$]$

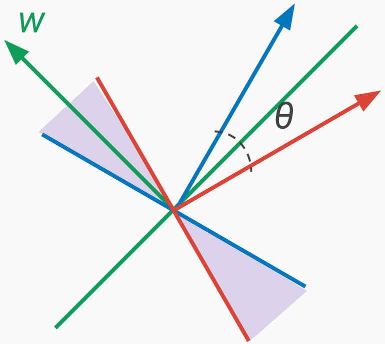
LSH for cosine similarity

[Charikar 2002]

- What if we want to compare vectors $u, v \in \mathbf{R}^d$ by cosine similarity?

$$\text{sim}(u, v) = \cos(\theta)$$

- Pick a vector w uniformly from the sphere in $\mathbf{R}^d$
  - $h_w(x) = 1$ if $w^T x >= 0$
    $= 0$ if $w^T x < 0$



- What's the probability of collision?
  - $\mathbf{P}[h_w(u) = h_w(v)] = 1 - \mathbf{P}[h_w(u) \neq h_w(v)]$
    $= 1 - \mathbf{P}[w$ in **shaded region**$]$
    $= 1 - 2 \cdot |\theta| / 2\pi$
    $= 1 - |\theta|/\pi$

Not exactly cos(θ), but monotonically decreasing with |θ| ⇒ same rank-ordering

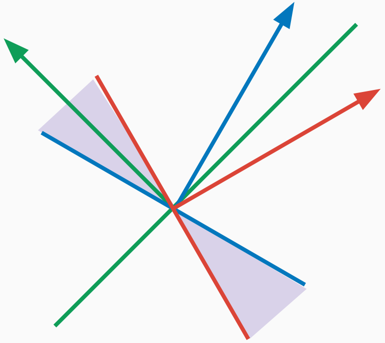
Multiple projections

- P[**No collision** | single projection] = θ/π
- P[**All projections do not collide** | m projections] = $(\theta/\pi)^m$
- P[At least one **Collision** | m projections] = $1 - (\theta/\pi)^m$

Multi-probe LSH

[Lv et al., 2007]

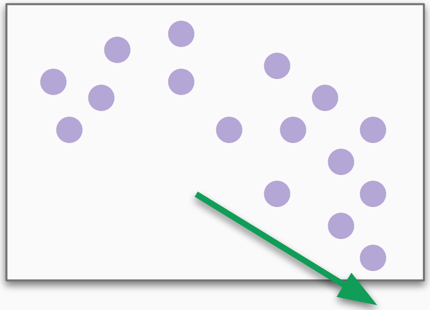
- Random projections can isolate neighbors from each other
  - LSH uses multiple projections to minimize the chance of this happening
  - But it might take a lot of projections!
- ***Multi-probe LSH*** explores neighboring hash buckets
  - Did the query land close to a threshold?
  - If so, check both buckets
- End result: better recall with fewer hash tables



Spatial trees

[Bentley, 1975]
[Uhlmann, 1991]

Recursive partitioning



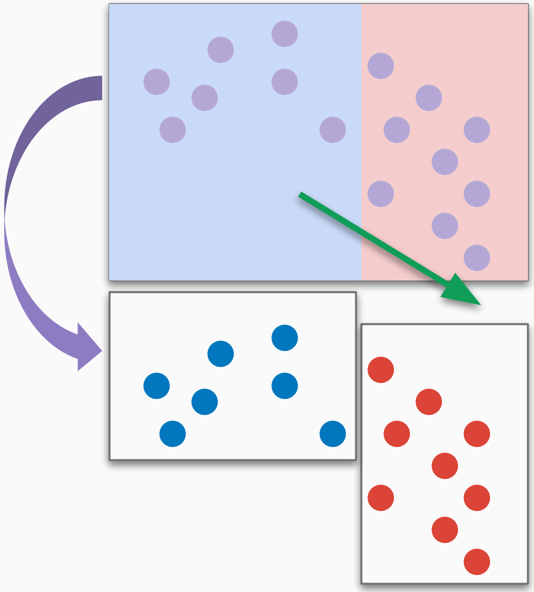
- Spatial trees recursively partition data into subsets
    - Pick a direction w
    - Split data $\{x_i\}$ at median $\{w^T x_i\}$
    - Recurse on left and right subsets
    - Stop when input set is sufficiently small
- Each split cuts data in half

  ⇒ O(log N) splits to get small candidate sets

Recursive partitioning

- Spatial trees recursively partition data into subsets
    - Pick a direction w
    - Split data $\{x_i\}$ at median $\{w^T x_i\}$
    - Recurse on left and right subsets
    - Stop when input set is sufficiently small
- Each split cuts data in half

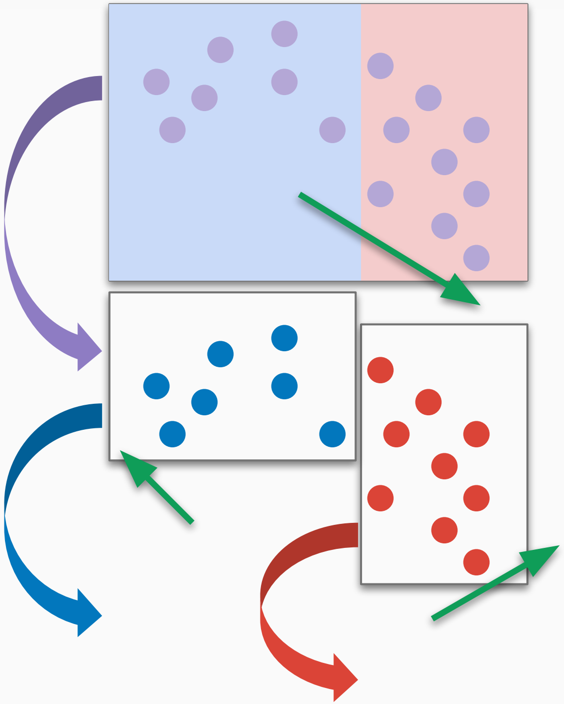
  $\Rightarrow$ O(log N) splits to get small candidate sets



Recursive partitioning

- Spatial trees recursively partition data into subsets
  - Pick a direction w
  - Split data $\{x_i\}$ at median $\{w^T x_i\}$
  - Recurse on left and right subsets
  - Stop when input set is sufficiently small
- Each split cuts data in half

  ⇒ O(log N) splits to get small candidate sets



KD-Trees

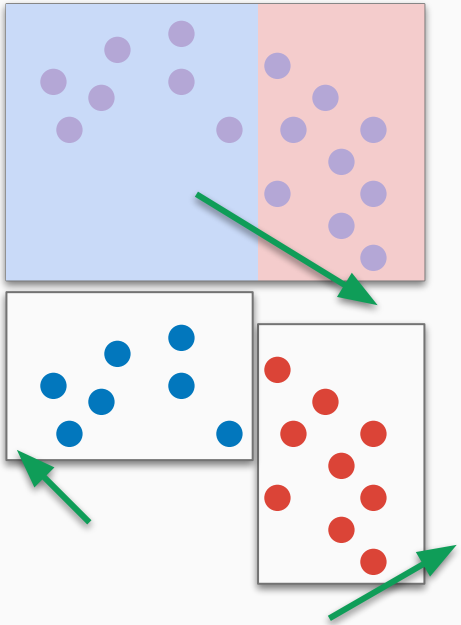
[Bentley, 1975]

- Splitting direction cycles through coordinates
  - $w_i \leftarrow \mathbf{e}_i$ (i^{th} standard basis)
- This works in low dimensions, but is inefficient in high dimensions
- Better alternative:
  - $w_i \leftarrow$ coordinate of maximum variance

Alternative splitting rules

[Verma, Kpotufe, & Dasgupta, 2009]

- Principal direction:
  - $w \leftarrow$ direction of maximum variance (PCA) estimated from input
  - **Idea**: variance ~= diameter of subset, and we want that to be **small**
- 2-means:
  - $w \leftarrow$ direction spanned by *k*-means solution for (k=2)
  - **Idea**: minimize variance (diameter) *after* splitting
- Random projection:
  - $w \leftarrow$ random on the unit sphere
  - **Idea**: ¯_(ツ)_/¯ randomness is robust!
    (and adaptive to intrinsic dimension of data)
  - Like LSH, takes multiple random projections (ie a forest) to work well



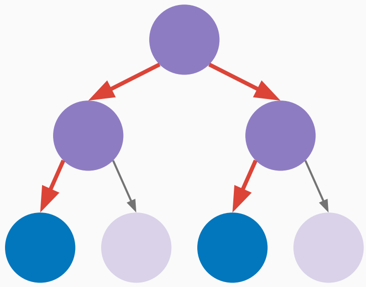
Spill trees

[Liu et al., 2005]

- Partition trees can isolate data near decision boundaries, just like LSH

- Fix is similar to multi-probe LSH:
  - Instead of splitting at median (0 overlap)
  - Make ***overlapping subsets*** with >50% of data

- Query now lands in multiple leaves
  - **Candidate set** = union of leaf sets



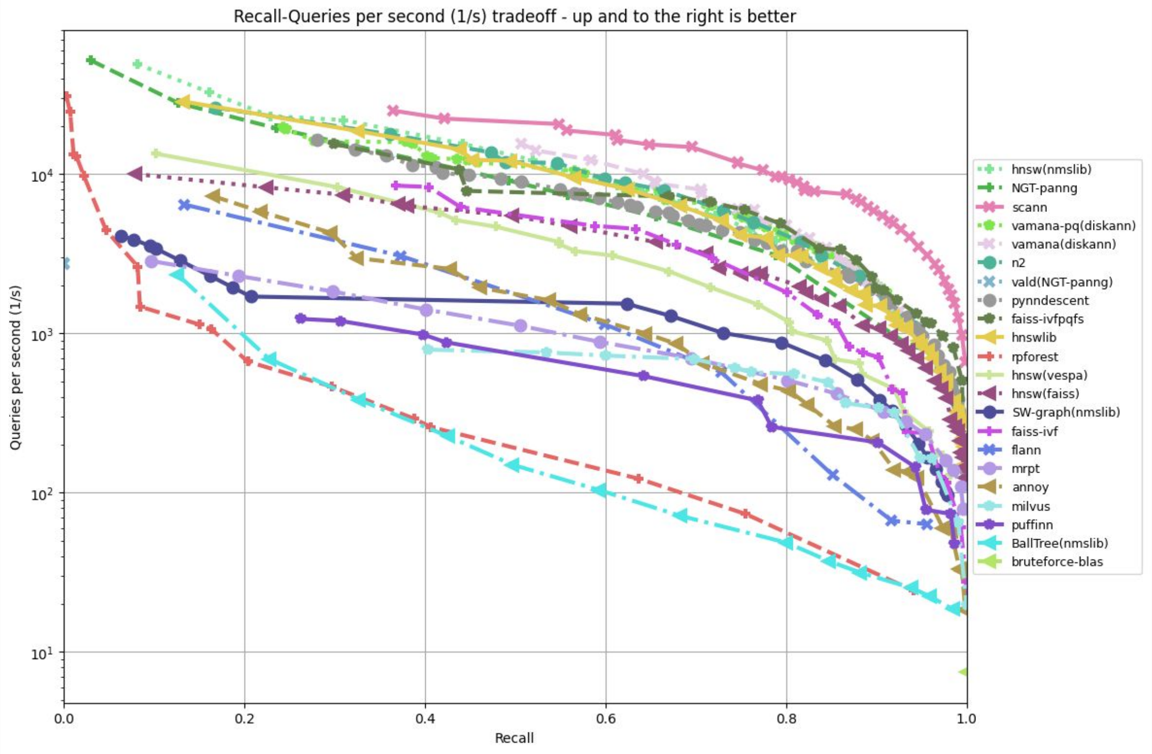
What should I use?

http://ann-benchmarks.com/

benchmarks for approximate nearest neighbor methods

Some points of reference (→)

- **bruteforce**
- **MP-lsh** (multiprobe)
- **KD**(-tree)
- **rpforest**
- **annoy** = approx PD forest
- **hnsw** = hierarchical non-metric small-world network



Wrap-up

- Similarity search benefits from spatial data structures
- MinHash is simple, but low-precision
- LSH can improve precision, many variants available
- Trees and other structures often work better in practice