COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2019.

Lecture 7

LOGISTICS

- Problem Set 1 is due Thursday in Gradescope.
- My office hours today are 1:15pm-2:15pm.

Lecture Pace: Piazza poll results for last class:

- 18%: too fast
- 48%: a bit too fast
- 26%: perfect
- 8%: (a bit) too slow

So will try to slow down a bit.

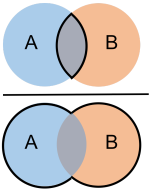
Last Class: Hashing for Jaccard Similarity

- MinHash for estimating the Jaccard similarity.
- Application to fast similarity search.
- Locality sensitive hashing (LSH).

This Class:

- Finish up MinHash and LSH.
- The Frequent Elements (heavy-hitters) problem.
- Misra-Gries summaries.

Jaccard Similarity: $J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{\text{\# shared elements}}{\text{\# total elements}}$.



Two Common Use Cases:

- **Near Neighbor Search:** Have a database of n sets/bit strings and given a set A, want to find if it has high similarity to anything in the database. Naively $O(n)$ time.
- **All-pairs Similarity Search:** Have n different sets/bit strings. Want to find all pairs with high similarity. Naively $O(n^2)$ time.

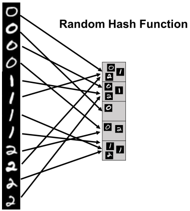
$MinHash(A) = \min_{a \in A} \mathbf{h}(a)$ where $\mathbf{h} : U \to [0,1]$ is a random hash.

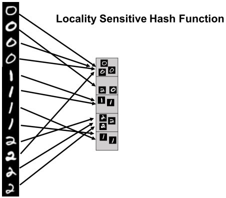
Locality Sensitivity: $\Pr(MinHash(A) = MinHash(B)) = J(A,B)$.

Represents a set with a single number that captures Jaccard similarity information!

Given a collision free hash function $\mathbf{g} : [0,1] \to [m]$,

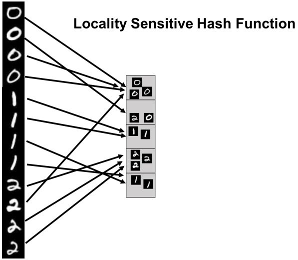
$$\Pr\left[\mathbf{g}(MinHash(A)) = \mathbf{g}(MinHash(B))\right] = J(A,B).$$





What happens to $\Pr\left[\mathbf{g}(MinHash(A)) = \mathbf{g}(MinHash(B))\right]$ if $\mathbf{g}$ is not collision free? Collision probability will be larger than $J(A,B)$.

When searching for similar items only search for matches that land in the same hash bucket.



- **False Negative:** A similar pair doesn't appear in the same bucket.
- **False Positive:** A dissimilar pair is hashed to the same bucket.

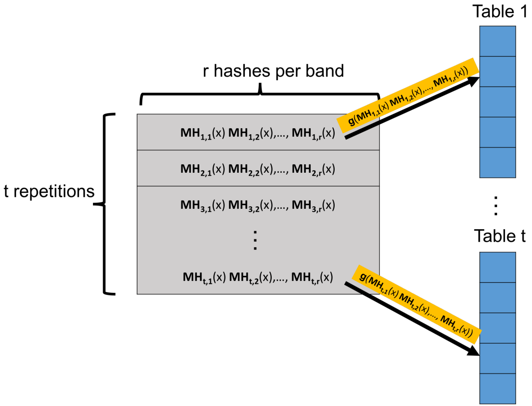
Need to balance a small probability of false negatives (a high hit rate) with a small probability of false positives (a small query time.)

Consider a pairwise independent random hash function $\mathbf{h} : U \to [m]$. Is this locality sensitive?

$$\Pr(\mathbf{h}(x) = \mathbf{h}(y)) = \frac{1}{m} \text{ for all } x, y \in U.$$ Not locality sensitive!

- Random hash functions (for load balancing, fast hash table look ups, bloom filters, distinct element counting, etc.) aim to evenly distribute elements across the hash range.
- Locality sensitive hash functions (for similarity search) aim to distribute elements in a way that reflects their similarities.

Balancing False Negatives/Positives with MinHash via repetition.



Create t hash tables. Each is indexed into not with a single MinHash value, but with r values, appended together. A length r signature:

$$\text{MH}_{i,1}(x), \text{MH}_{i,2}(x), \ldots, \text{MH}_{i,r}(x).$$

For A, B with Jaccard similarity $J(A, B) = s$, probability their length r MinHash signatures collide:

$$\Pr\left([\mathrm{MH}_{i,1}(A), \ldots, \mathrm{MH}_{i,r}(A)] = [\mathrm{MH}_{i,1}(B), \ldots, \mathrm{MH}_{i,r}(B)]\right) = s^r.$$

Probability the signatures don't collide:

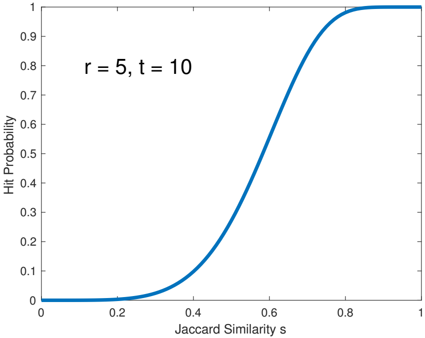
$$\Pr\left([\mathrm{MH}_{i,1}(A), \ldots, \mathrm{MH}_{i,r}(A)] \neq [\mathrm{MH}_{i,1}(B), \ldots, \mathrm{MH}_{i,r}(B)]\right) = 1 - s^r.$$

Probability there is at least one collision in the t hash tables:

$$\Pr\left(\exists i : [\mathrm{MH}_{i,1}(A), \ldots, \mathrm{MH}_{i,r}(A)] = [\mathrm{MH}_{i,1}(B), \ldots, \mathrm{MH}_{i,r}(B)]\right) = 1 - (1 - s^r)^t.$$

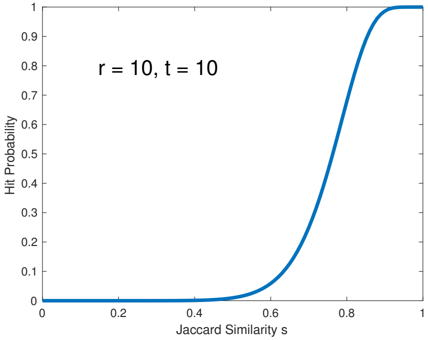
$\mathrm{MH}_{i,j}$: $(i, j)^{th}$ independent instantiation of MinHash. t repetitions ($i = 1, \ldots t$), each with r hash functions ($j = 1, \ldots r$) to make a length r signature.

Using t repetitions each with a signature of r MinHash values, the probability that x and y with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.



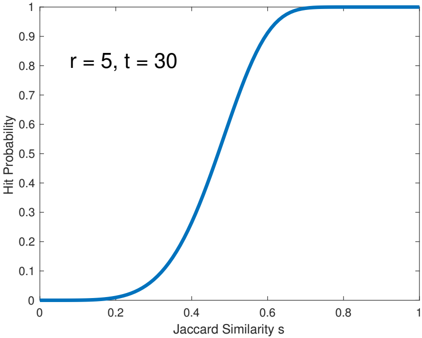
THE s-CURVE

Using t repetitions each with a signature of r MinHash values, the probability that x and y with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.



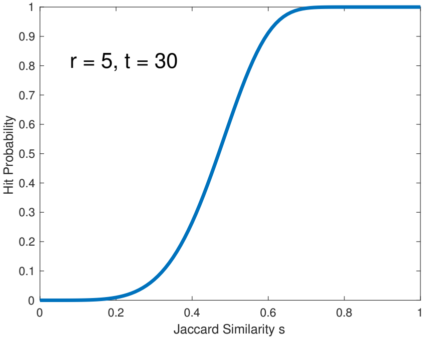
THE s-CURVE

Using t repetitions each with a signature of r MinHash values, the probability that x and y with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.



THE s-CURVE

Using t repetitions each with a signature of r MinHash values, the probability that x and y with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.



r and t are tuned depending on application. 'Threshold' when hit probability is 1/2 is $\approx (1/t)^{1/r}$. E.g., $\approx (1/30)^{1/5} = .51$ in this case.

For example: Consider a database with $10,000,000$ audio clips. You are given a clip x and want to find any y in the database with $J(x,y) \geq .9$.

- There are 10 true matches in the database with $J(x,y) \geq .9$.
- There are 1000 near matches with $J(x,y) \in [.7,.9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x,y) = s$ is $1-(1-s^{25})^{50}$.

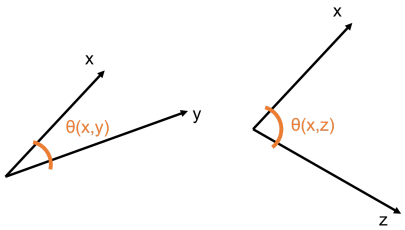
- Hit probability for $J(x,y) \geq .9$ is $\geq 1-(1-.9^{25})^{50} \approx .98$ and ≤ 1.
- Hit probability for $J(x,y) \in [.7,.9]$ is $\leq 1-(1-.9^{25})^{50} \approx .98$
- Hit probability for $J(x,y) \leq .7$ is $\leq 1-(1-.7^{25})^{50} \approx .007$

Expected Number of Items Scanned: (proportional to query time)

$$1 * 10 + .98 * 1000 + .007 * 9,998,990 \approx 80,000 \ll 10,000,000.$$

Repetition and s-curve tuning can be used for search with any similarity metric, given a locality sensitive hash function for that metric.

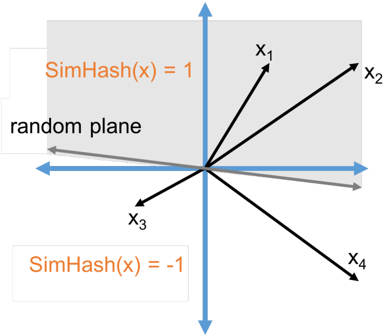
- LSH schemes exist for many similarity/distance measures: hamming distance, cosine similarity, etc.



Cosine Similarity: $\cos(\theta(x,y)) = \frac{\langle x,y \rangle}{\|x\|_2 \cdot \|y\|_2}$.

- $\cos(\theta(x,y)) = 1$ when $\theta(x,y) = 0°$ and $\cos(\theta(x,y)) = 0$ when $\theta(x,y) = 90°$, and $\cos(\theta(x,y)) = -1$ when $\theta(x,y) = 180°$

SimHash Algorithm: LSH for cosine similarity.

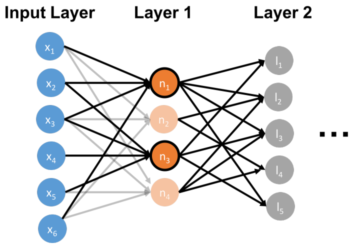


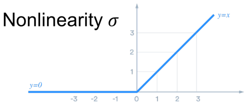
$$SimHash(x) = \mathrm{sign}(\langle x, t \rangle) \text{ for a random vector } t.$$

$$\Pr\left[SimHash(x) = SimHash(y)\right] = 1 - \frac{\theta(x,y)}{\pi} \approx \frac{\cos(\theta(x,y)) + 1}{2}.$$

HASHING FOR NEURAL NETWORKS

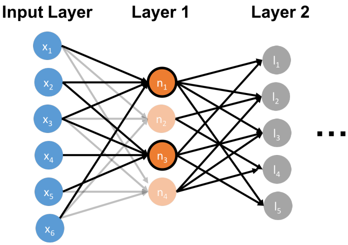
Many applications outside traditional similarity search. E.g., approximate neural net computation (Anshumali Shrivastava).

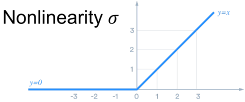




$$n_i = \sigma\left(\sum_{j=1}^{m} w(x_j, n_i) \cdot x_j\right) = \sigma(\langle w_i, x\rangle)$$

- Evaluating $\mathcal{N}(x)$ requires $|x| \cdot |\text{layer 1}| + |\text{layer 1}| \cdot |\text{layer 2}| + \ldots$ multiplications if fully connected.
- Can be expensive, especially on constrained devices like cellphones, cameras, etc.
- For approximate evaluation, suffices to identify the neurons in each layer with high activation when x is presented.





$$n_i = \sigma\left(\sum_{j=1}^{m} w(x_j, n_i) \cdot x_j\right) = \sigma(\langle w_i, x\rangle)$$

- Important neurons have high activation $\sigma(\langle w_i, x\rangle)$.
- Since σ is typically monotonic, this means large $\langle w_i, x\rangle$.
- $\cos(\theta(w_i, x)) = \frac{\langle w_i, x\rangle}{\|w_i\|\|x\|}$. Thus these neurons can be found very quickly using LSH for cosine similarity search.

	Bloom Filters	Hash Table	MinHash	Distinct Elements
Goal	Check if x is a duplicate of y in database.	Check if x is a duplicate of any y in database and return y.	Check if x is a duplicate of any y in database and return y.	Count # of items, excluding duplicates.
Approximate Duplicates?	✗	✗	✓	✗

All different variants of detecting duplicates/finding matches in large datasets. An important problem in many contexts!

MinHash(*A*) is a single number sketch, that can be used both to estimate the number of items in *A* and the Jaccard similarity between *A* and other sets.

Questions on MinHash and Locality Sensitive Hashing?