Approximate Near Neighbor Search: Locality-sensitive Hashing

COMPCSI 753: Algorithms for Massive Data

Ninh Pham

University of Auckland

Auckland, Aug 10, 2020

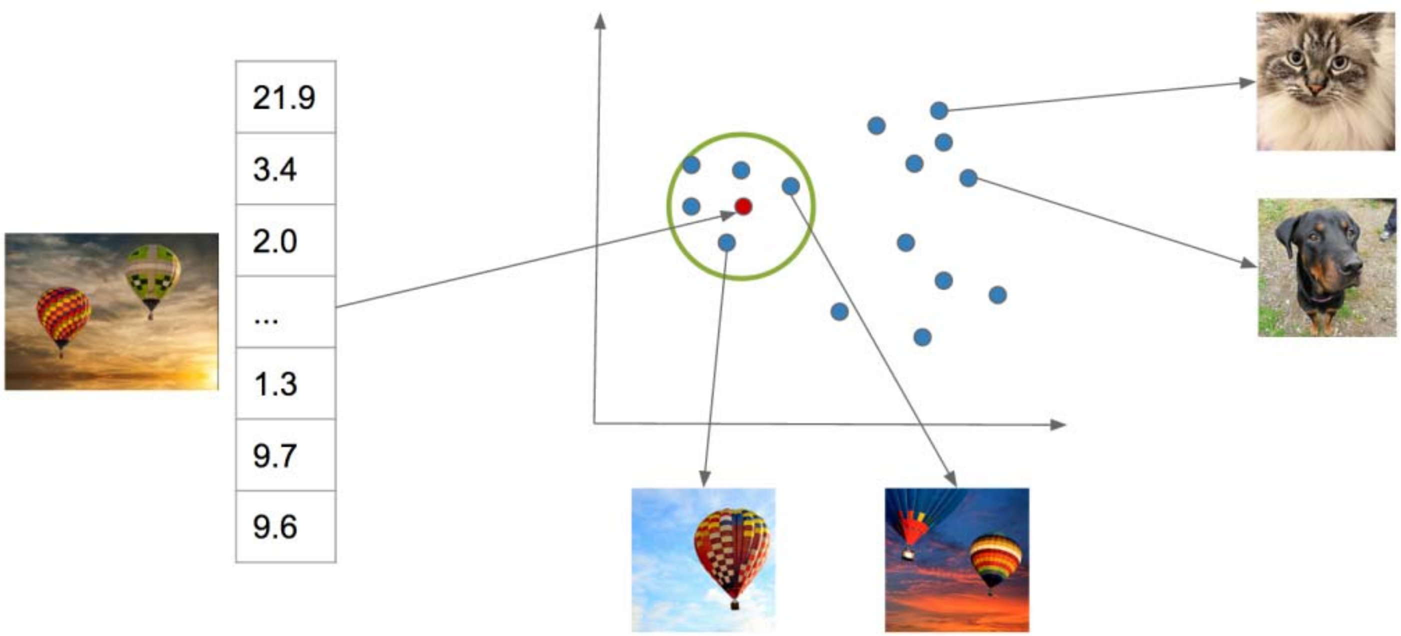
Outline

- Popular similarity/distance measure
  - Definitions
  - Applications
- Locality-sensitive hashing framework for approximate near neighbor search
  - LSH definition
  - LSH framework for near neighbor search

A common metaphor

- We can present many problems as finding "similar" items or finding near-neighbors in high dimensions.

- Near neighbors search:



Source: https://code.flickr.net/2017/03/07/introducing-similarity-search-at-flickr/

Distance/similarity measure

- For each application, we need to define a similarity/distance measure to distinguish between
    - Similar and dissimilar objects.
    - Near and far apart objects.

- Popular similarity/distance measures:
    - Jaccard similarity/distance
    - Cosine similarity/distance
    - Hamming distance
    - Euclidean distance (L2)
    - Manhattan distance (L1)

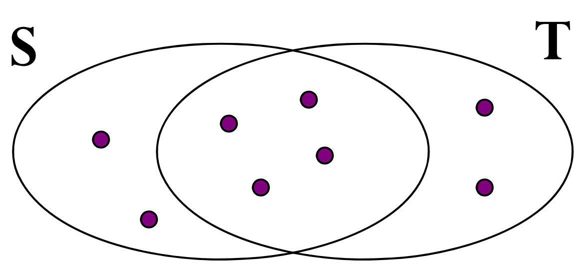
Jaccard similarity/distance

- Jaccard similarity of two sets **S** and **T**:

$$J(S,T) = \frac{|S \cap T|}{|S \cup T|}$$

- Jaccard distance of two sets **S** and **T**: **1 – J(S,T)**

- Example:



4 items in intersection
8 items in union
J(S,T) = 1/2
1 - J(S,T) = 1/2

Applications

- Texts/Documents:
    - Present text/documents as a set of shingles
    - Usage: Detect plagiarism, mirror pages, articles from same source

- Recommender system data:
    - Online-purchases: users as sets, items as set elements
    - Netflix: users as sets, movies rated as set elements
    - Usages: Collaborative filtering recommender systems need to compute the similarity between users-users, items-items for accurate recommendation

Applications

- Market basket data:
  - Items as sets, transaction IDs as set elements.
  - Usage: Association rules $\mathbf{X} \rightarrow \mathbf{Y}$ if $\mathbf{J(X,Y)} \geq \mathbf{s}$.

- Example:
  - Bread = $\{1, 2, 4, 5\}$
  - Milk = $\{1, 3, 4, 5\}$
  - Diapers = $\{2, 3, 4, 5\}$
  - Beer = $\{2, 3, 4\}$
  - Eggs = $\{2\}$
  - Cola = $\{3, 5\}$
  - J(Diapers, Beer) = $3/4$

ID	Items
1	{Bread, Milk}
2	{Bread, Diapers, Beer, Eggs}
3	{Milk, Diapers, Beer, Cola}
4	{Bread, Milk, Diapers, Beer}
5	{Bread, Milk, Diapers, Cola}
…	…

market basket transactions

{Diapers, Beer} Example of a frequent itemset

{Diapers} → {Beer} Example of an association rule

MinHash [Broder'97]

- Permute the Boolean matrix with a random permutation $\boldsymbol{\pi}$

- MinHash function:

$\mathbf{h_\pi(S)}$ = the index of the first row with value **1** of **S** (in the permuted order π)

$$\mathbf{h_\pi(S) = min_\pi\ \pi(S)}$$

- $\mathbf{Pr_\pi[h_\pi(S) = h_\pi(T)] = J(S,T)}$

	A	B	C	D
1	1	1	1	0
2	1	1	0	1
3	0	1	0	1
4	0	0	0	1
5	1	0	0	1
6	1	1	1	0
7	1	0	1	0

Documents

Shingles

Cosine similarity/distance

- Given two points $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_d)$ and $\mathbf{y} = (\mathbf{y}_1, \ldots, \mathbf{y}_d)$ in high-dimensional sphere, i.e. $\|\boldsymbol{x}\|_2^2 = \|\boldsymbol{y}\|_2^2 = \mathbf{1}$

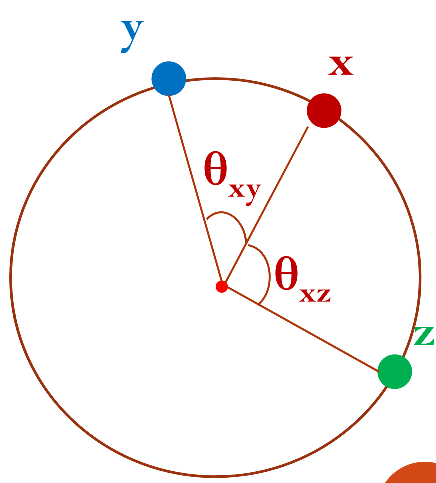
$$\|\boldsymbol{x}\|_2^2 = \sqrt{\boldsymbol{x}_1^2 + \boldsymbol{x}_2^2 + \cdots + \boldsymbol{x}_d^2} = \mathbf{1}$$

- Cosine similarity:

$$\cos(\theta_{xy}) = \langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i=1}^{d} \boldsymbol{x}_i \boldsymbol{y}_i$$

where $0 \leq \theta_{xy} \leq \pi$

- Cosine distance: $1 - \cos(\theta_{xy})$



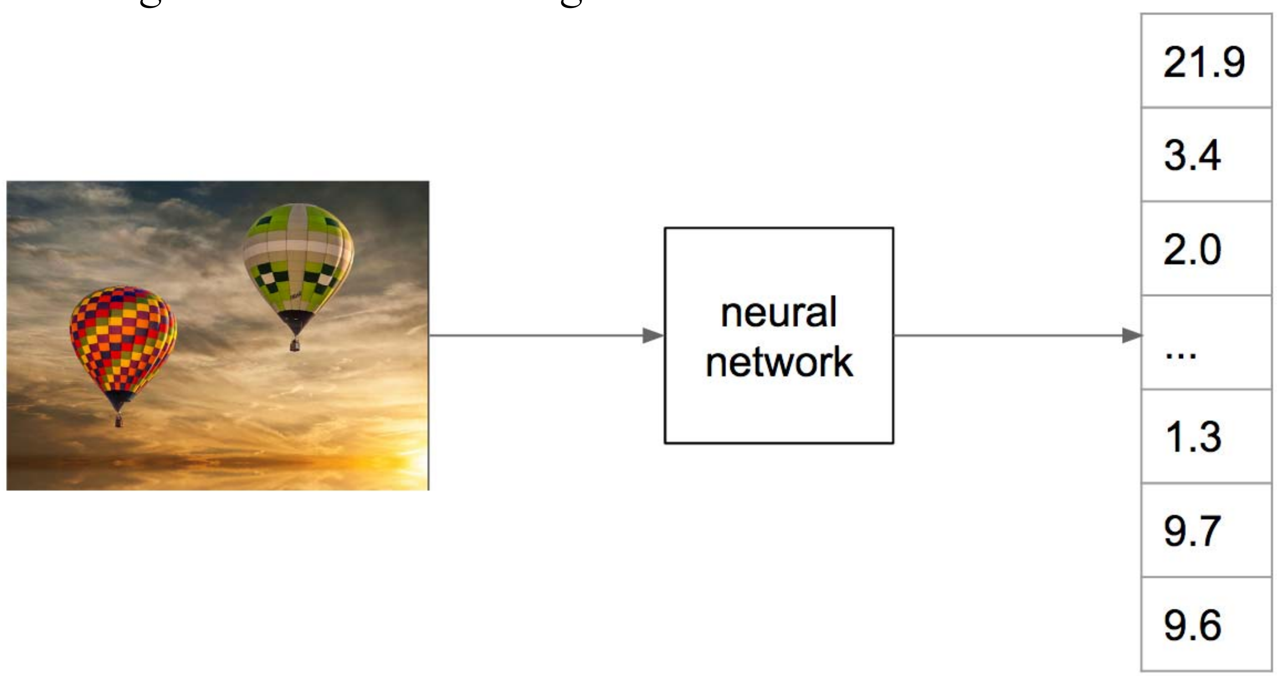
Applications

- Texts/Documents:
  - Present texts/documents as a high-dimensional points using term frequency–inverse document frequency (tf-idf).
  - L2 normalization, i.e. normalizing $\mathbf{x}$ by $(\mathbf{x}_1, \dots, \mathbf{x}_d)/\|\boldsymbol{x}\|_2^2$.
  - Usage: Information retrieval (text search, topic modeling) and most of MinHash applications.

- A tf-idf definition for a corpus of **n** documents:
  - $\mathbf{tf(t, d)} = \dfrac{\text{number of times term } \mathbf{t} \text{ occur in document } \mathbf{d}}{\text{total number of terms in document } \mathbf{d}}$
  - $\mathbf{idf(t)} = \mathbf{log}_2\left(\dfrac{\mathbf{n}}{\mathbf{1} + \text{number of documents where } \mathbf{t} \text{ appears}}\right)$
  - $\mathbf{tf\text{-}idf(t, d) = tf(t, d) * idf(t)}$

Applications

- Images:
  - Present an image as a high-dimensional vector, e.g. scale-invariant feature transform (SIFT), deep learning ("image2vec"),…
  - Usage: content-based image retrieval



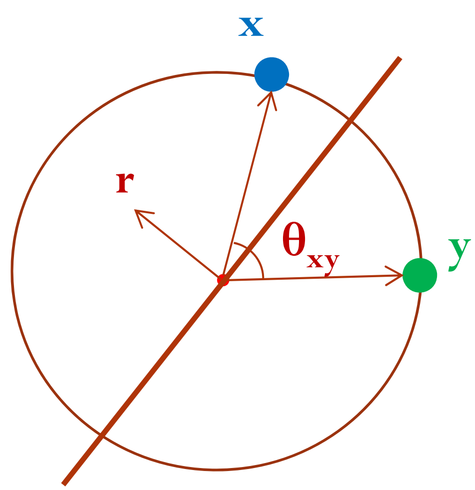
Source: https://code.flickr.net/2017/03/07/introducing-similarity-search-at-flickr/

SimHash [Charikar'02]

- Generate a random Gaussian vector $\mathbf{r} = (\mathbf{r}_1, \ldots, \mathbf{r}_d)$ where $\mathbf{r}_i$ is drawn from $\mathbf{N(0, 1)}$

- SimHash function:

$$\mathbf{h_r(x)} = \begin{cases} \mathbf{1 \text{ if } \langle \boldsymbol{x}, \boldsymbol{r} \rangle \geq 0} \\ \mathbf{0 \text{ if } \langle \boldsymbol{x}, \boldsymbol{r} \rangle < 0} \end{cases}$$



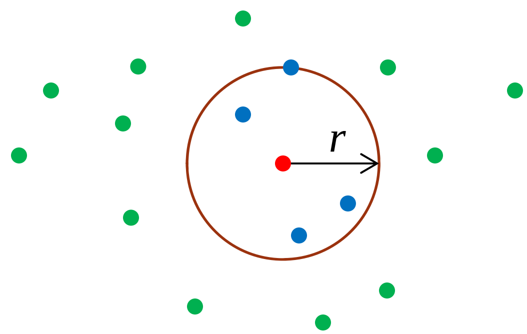
- $\mathbf{Pr_r\,[h_r(x) = h_r(y)] = 1 - \theta_{xy} / \pi}$

Outline

- Popular similarity/distance measure
- Locality-sensitive hashing framework for approximate near neighbor search

r-near neighbor search

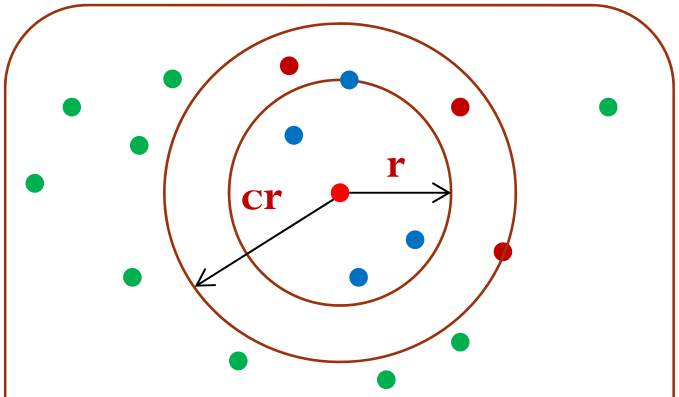
- **r**-near neighbor search (**r**NNS) problem:
  - Given a data set $\mathbf{S} \subset \mathbf{R}^{\mathbf{d}}$, a distance function $\mathbf{d(x, y)}$, a distance threshold $\mathbf{r}$, and a query $\mathbf{q} \in \mathbf{R}^{\mathbf{d}}$, return some point $\mathbf{x} \in \mathbf{S}$ where $\mathbf{d(x, q)} \leq \mathbf{r}$.



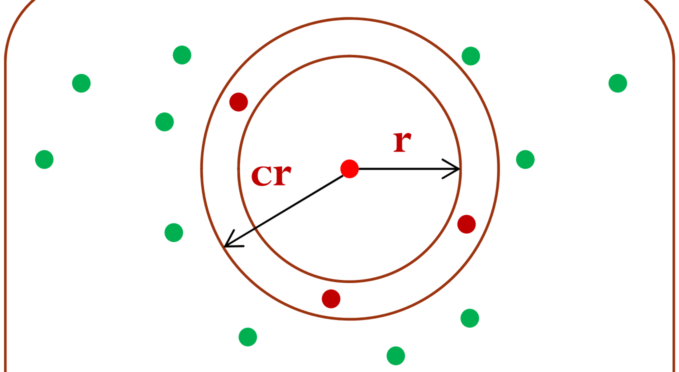
- Other variants:
  - **k-**nearest neighbors search
  - **r**-near neighbor reporting

Problem relaxation

- Randomized **c**-approximation **r**NNS:
  - Given a data set $\mathbf{S} \subset \mathbf{R}^d$, a distance function $\mathbf{d(x, y)}$, parameters $\mathbf{r > 0}$, $\mathbf{c > 1}$, $\boldsymbol{\delta} > \mathbf{0}$, and a query $\mathbf{q} \in \mathbf{R}^d$, with probability $\mathbf{1 - \delta}$
    - If exists $\mathbf{x} \in \mathbf{S}$ and $\mathbf{d(x, q)} \leq \mathbf{r}$, return some point $\mathbf{x'} \in \mathbf{S}$ where $\mathbf{d(x', q)} \leq \mathbf{cr}$.
    - Else, return nothing.



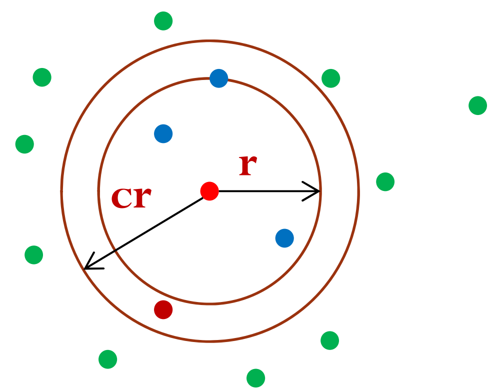
Return one of red or blue points



Return nothing

LSH definition

- Definition [Indyk & Motwani'98]:
  - Given a distance function $\mathbf{d}: \mathbf{U} \times \mathbf{U} \to \mathbf{R}$ and positive values $\mathbf{r}, \mathbf{c}, \mathbf{p_1}, \mathbf{p_2}$ where $\mathbf{p_1} > \mathbf{p_2}$, $\mathbf{c} > \mathbf{1}$. A family of functions $\mathbf{H}$ is called ($\mathbf{r}, \mathbf{c}, \mathbf{p_1}, \mathbf{p_2}$)-sensitive if for uniformly chosen $\mathbf{h} \in \mathbf{H}$ and all $\mathbf{x}, \mathbf{y} \in \mathbf{U}$:
  - If $\mathbf{d(x, y) \leq r}$ then $\mathbf{Pr\ [h(x) = h(y)] \geq p_1}$; (similar/near points)
  - If $\mathbf{d(x, y) \geq cr}$ then $\mathbf{Pr\ [h(x) = h(y)] \leq p_2}$. (dissimilar/far away points)



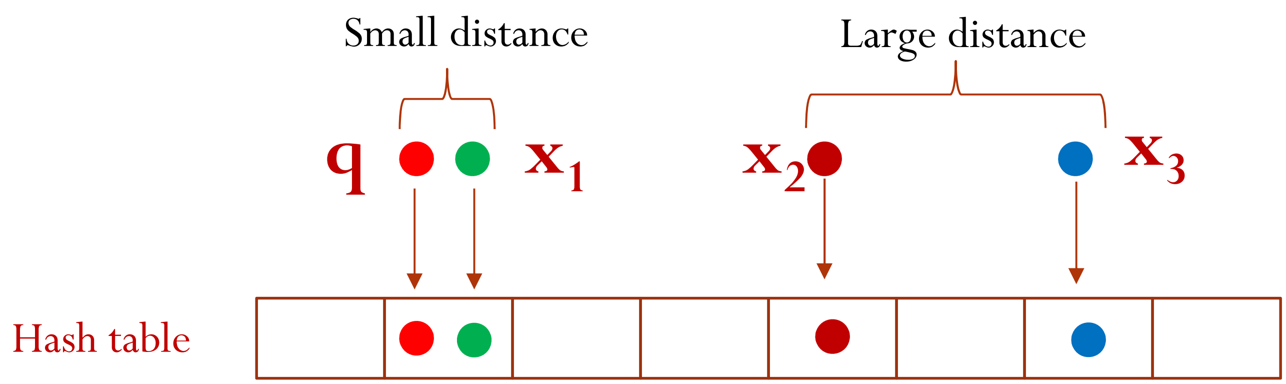
Exercise

- MinHash is locality-sensitive hashing
  - $\mathbf{Pr_{\pi}[h_{\pi}(S) = h_{\pi}(T)] = J(S,T)}$
  - What are the values of $\mathbf{p_1}$, $\mathbf{p_2}$ given **r** and **cr**?
- SimHash is locality-sensitive hashing
  - $\mathbf{Pr_{r}\ [h_{r}(x) = h_{r}(y)] = 1 - \theta_{xy} / \pi}$
  - What are the values of $\mathbf{p_1}$, $\mathbf{p_2}$ given **r** and **cr**?

Locality-sensitive hashing framework

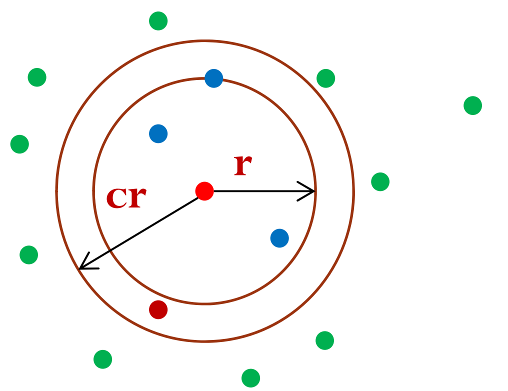
- Claim:
  - We can solve the randomized **c**-approximation **r**-near neighbor search in sublinear time using locality-sensitive hashing.
  - Sublinear time: $\mathbf{O(n^{\rho}), 0 < \rho < 1}$ for any query and data set of size **n**.

- Intuition: Candidate points are points within the same bucket of the query point.



Recap: LSH definition

- Definition [Indyk & Motwani'98]:
  - Given a distance function $\mathbf{d: U \times U \rightarrow R}$ and positive values $\mathbf{r, c, p_1, p_2}$ where $\mathbf{p_1 > p_2}$, $\mathbf{c > 1}$. A family of functions $\mathbf{H}$ is called $(\mathbf{r, c, p_1, p_2})$-sensitive if for uniformly chosen $\mathbf{h \in H}$ and all $\mathbf{x, y \in U}$:
  - If $\mathbf{d(x, y) \leq r}$ then $\mathbf{Pr\ [h(x) = h(y)] \geq p_1}$; (similar/near points)
  - If $\mathbf{d(x, y) \geq cr}$ then $\mathbf{Pr\ [h(x) = h(y)] \leq p_2}$. (dissimilar/far away points)



How to make it as small as possible, i.e. $\mathbf{p_2 = 1/n}$.

Concatenate several hash functions

- Define a hash function $\mathbf{g(x) = (h_1(x), h_2(x), \ldots , h_k(x))}$ where $\mathbf{h_i}$, $\mathbf{1 \le i \le k}$, are chosen independently and randomly from a LSH family $\mathbf{H}$.

  - If $\mathbf{d(x, y) \le r}$ then $\mathbf{Pr\ [g(x) = g(y)] \ge (p_1)^k}$ (similar/near points)

  - If $\mathbf{d(x, y) \ge cr}$ then $\mathbf{Pr\ [g(x) = g(y)] \le (p_2)^k}$ (dissimilar/far away points)

- Set $\mathbf{k = \log(n) / \log(1/p_2)}$, we have
  - $\mathbf{(p_2)^k = (p_2)^{\log(n) / \log(1/p2)} = 1/n}$
  - $\mathbf{(p_1)^k = (p_1)^{\log(n)/\log(1/p2)} = 1/n^\rho}$ where $\mathbf{0 < \rho = \dfrac{\log(1/p1)}{\log(1/p2)} < 1}$

LSH framework intuition

- Similar points:
  - If $\mathbf{d(x, y)} \leq \mathbf{r}$ then $\mathbf{Pr}\ [\mathbf{g(x)} = \mathbf{g(y)}] \geq \mathbf{1/n^{\rho}}$
  - If there is a similar point, i.e. $\mathbf{d(x, q)} \leq \mathbf{r}$, in expectation, we need to check $\mathbf{n^{\rho}}$ points to find a similar point in our bucket.

- Dissimilar points:
  - If $\mathbf{d(x, y)} \geq \mathbf{cr}$ then $\mathbf{Pr}\ [\mathbf{g(x)} = \mathbf{g(y)}] \leq \mathbf{1/n}$
  - Since we have **n** points, in expectation, only **1** dissimilar point is in the same bucket of query.

LSH framework intuition

- Similar points:
  - If $\mathbf{d(x, y)} \leq \mathbf{r}$ then $\mathbf{Pr}\ [\mathbf{g(x)} = \mathbf{g(y)}] \geq \mathbf{1/n^{\rho}}$
  - If there is a similar point, i.e. $\mathbf{d(x, q)} \leq \mathbf{r}$, in expectation, we need to check $\mathbf{n^{\rho}}$ points to find a similar point in our bucket.

How to get $\mathbf{n^{\rho}}$ points?

- Dissimilar points:
  - If $\mathbf{d(x, y)} \geq \mathbf{cr}$ then $\mathbf{Pr}\ [\mathbf{g(x)} = \mathbf{g(y)}] \leq \mathbf{1/n}$
  - Since we have $\mathbf{n}$ points, in expectation, only **1** dissimilar point is in the same bucket of query.

No worries on dissimilar points

LSH framework intuition

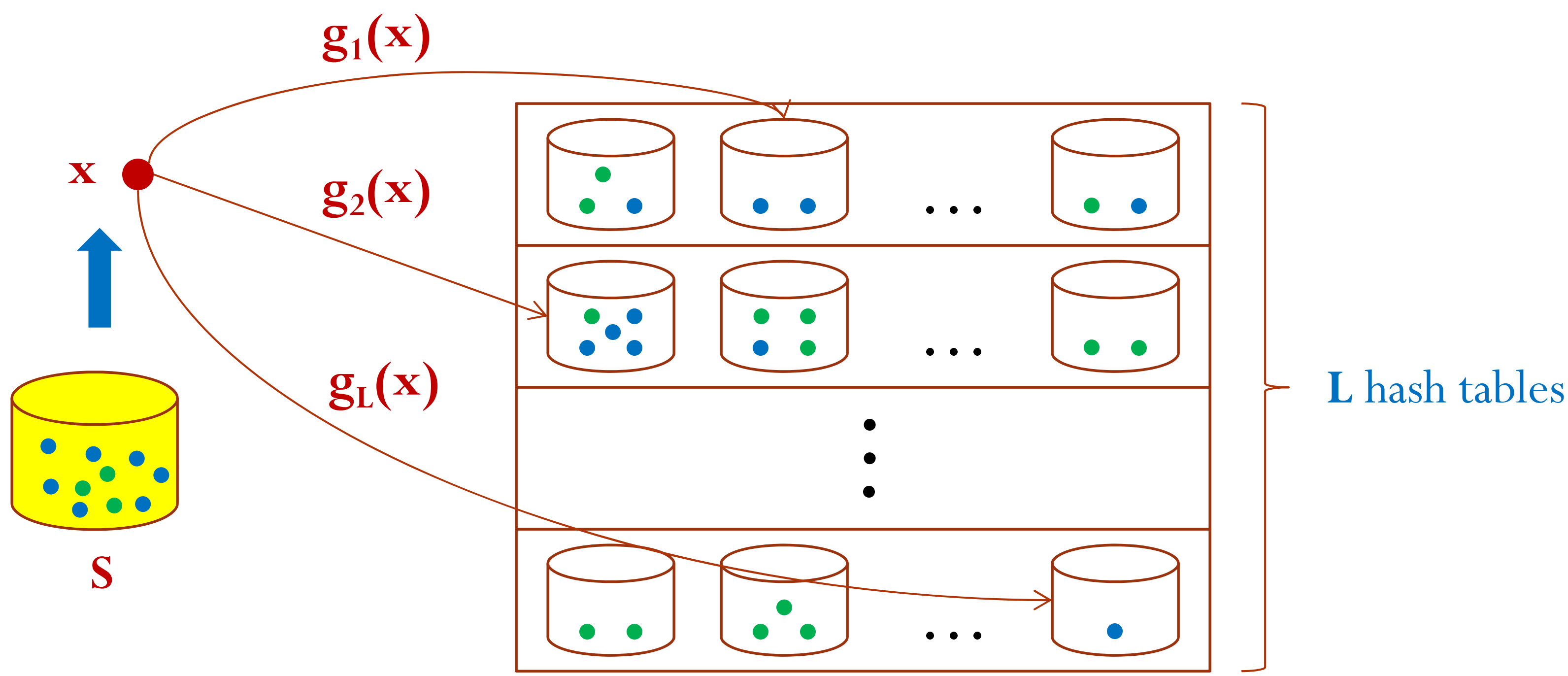
- Similar points:
  - If $\mathbf{d(x, y)} \leq \mathbf{r}$ then $\mathbf{Pr\ [g(x) = g(y)]} \geq \mathbf{1/n^{\rho}}$
  - If there is a similar point, i.e. $\mathbf{d(x, q)} \leq \mathbf{r}$, in expectation, we need to check $\mathbf{n^{\rho}}$ points to find a similar point in our bucket.

Use $\mathbf{L = n^{\rho}}$ hash tables

- Dissimilar points:
  - If $\mathbf{d(x, y)} \geq \mathbf{cr}$ then $\mathbf{Pr\ [g(x) = g(y)]} \leq \mathbf{1/n}$
  - Since we have **n** points, in expectation, only **1** dissimilar point is in the same bucket of query.

In expectation, less than $\mathbf{L = n^{\rho}}$ points

LSH framework: Hash tables construction



Space usage: $O(nL + dn)$

$g(x) = (h_1(x), h_2(x), \ldots, h_k(x))$ where $h_i \in H$

Prob. collision of near points: $1/n^{\rho}$

Prob. collision of far away points: $1/n$

LSH framework: Hash tables construction

- Preprocessing:

1) Choose L hash function g_j, $1 \le j \le L$, by setting $g_j = (h_{1,j}, h_{2,j}, \dots, h_{k,j})$ where $h_{i,j}$, $1 \le i \le k$, are chosen independently and randomly from the LSH family H.

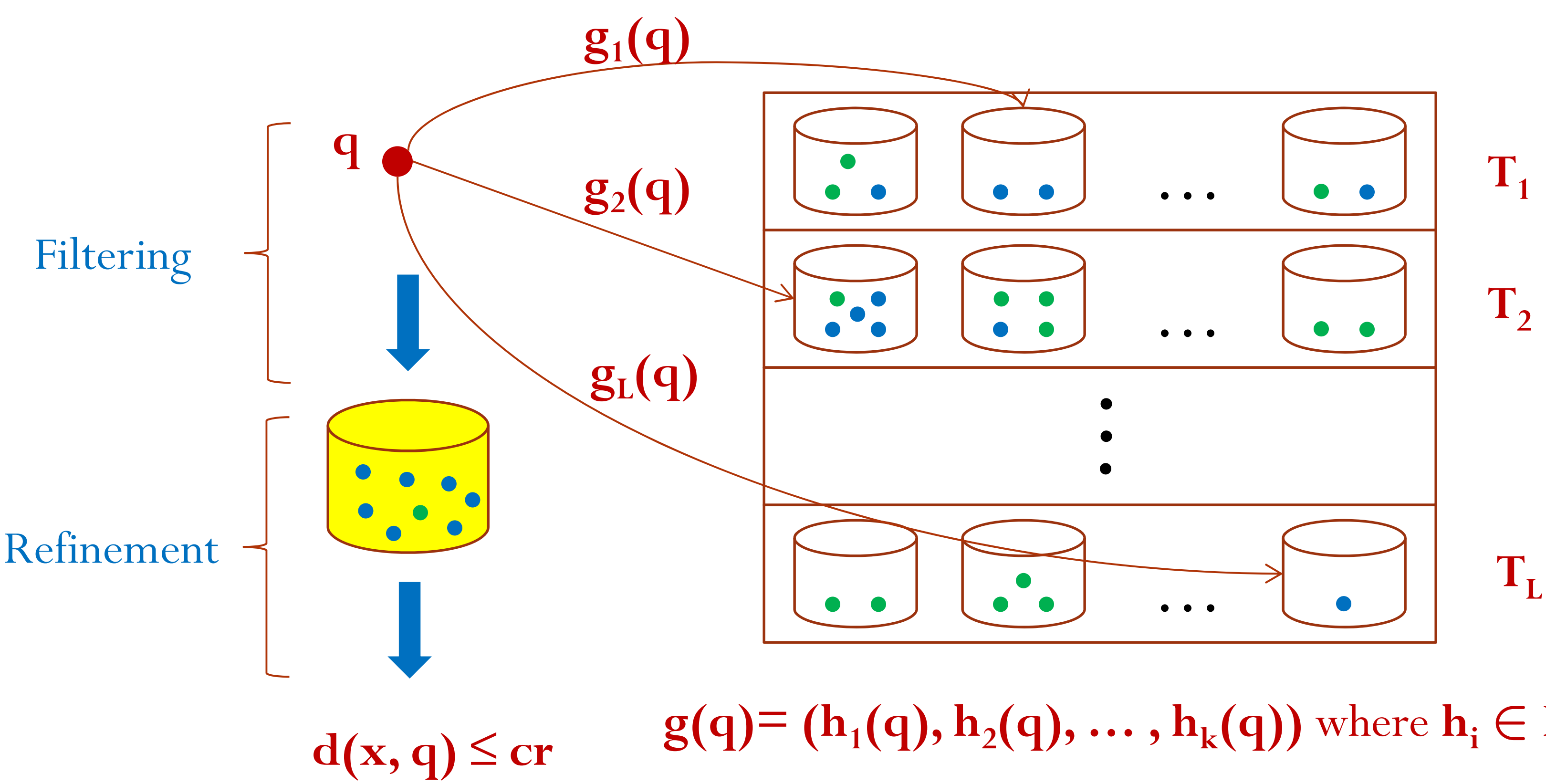
2) Construct L hash tables, where each hash table T_j, $1 \le j \le L$, contains the data set S hashed using the hash function g_j.

- Complexity:
  - Space usage: $\mathbf{O(nL + dn)}$
  - Time: $\mathbf{O(dknL)}$ with the assumption that the evaluation of $\mathbf{h(x)}$ takes $\mathbf{O(d)}$ time.

LSH framework: Hash tables lookup

Important:

We interrupt search after finding first **$L' = 2n^{\rho}$** points, including duplicates.



$g(q) = (h_1(q), h_2(q), \ldots, h_k(q))$ where $h_i \in H$

LSH framework: Hash tables lookup

- Query for a point **q**:

For each $j = 1, 2, \ldots, L$
1) Retrieve the points from the bucket $g_j(q)$ in the T_j hash table.
2) For each of the reported point x
   2.1) Compute the distance $d(x, q)$.
   2.2) If $d(x, q) \le cr$, return x and stop.
3) Stop as soon as the number of reported points is more than L'.

- Complexity:
  - Time: $\mathbf{O(dkL + dL')}$ with the assumption that the evaluation of $\mathbf{h(q)}$ takes $\mathbf{O(d)}$ time.

Compute hash values

Compute actual distance with **L'** points

Analysis

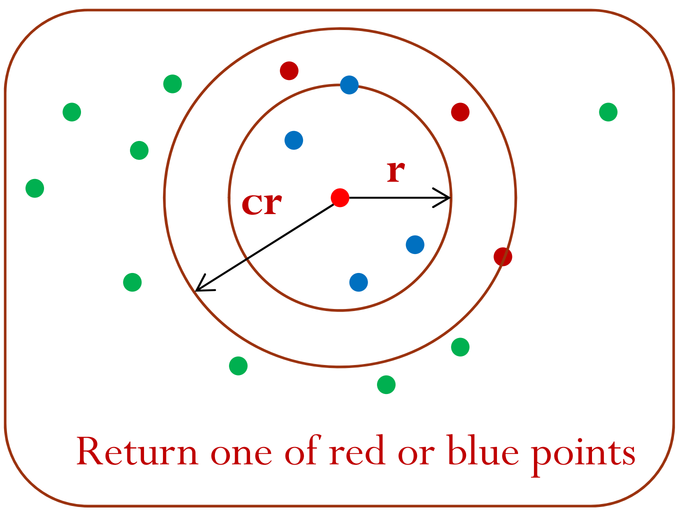
- Parameter settings:
  - $k = \log(n) / \log(1/p_2)$
  - $0 < \rho = \dfrac{\log(1/p1)}{\log(1/p2)} < 1$
  - $L = n^{\rho}$ hash tables
  - Check $L' = 2L = 2n^{\rho}$ points for computing actual distance

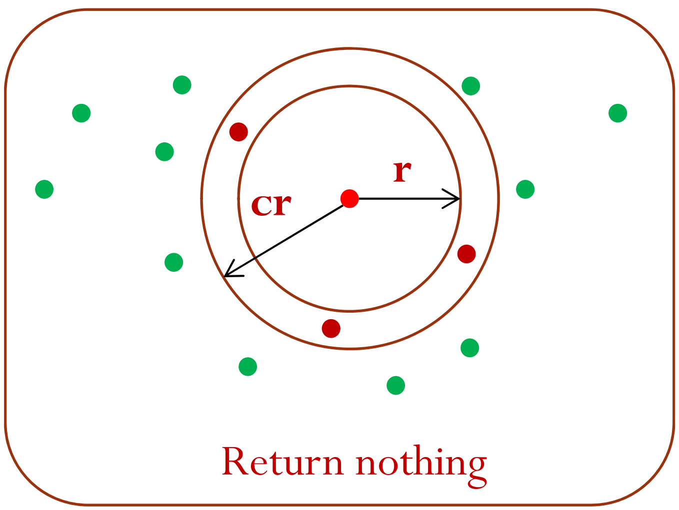
Prob. collision of near points: $1/n^{\rho}$
Prob. collision of far away points: $1/n$

- Complexity:
  - Preprocessing:
    - Space usage: $O(nL + dn)$
    - Time: $O(dkn^{1+\rho})$ with the assumption that the evaluation of $h(x)$ takes $O(d)$ time.
  - Query:
    - Sublinear time: $O(dkn^{\rho} + dn^{\rho})$ with the assumption that the evaluation of $h(q)$ takes $O(d)$ time.

Recap: Problem relaxation

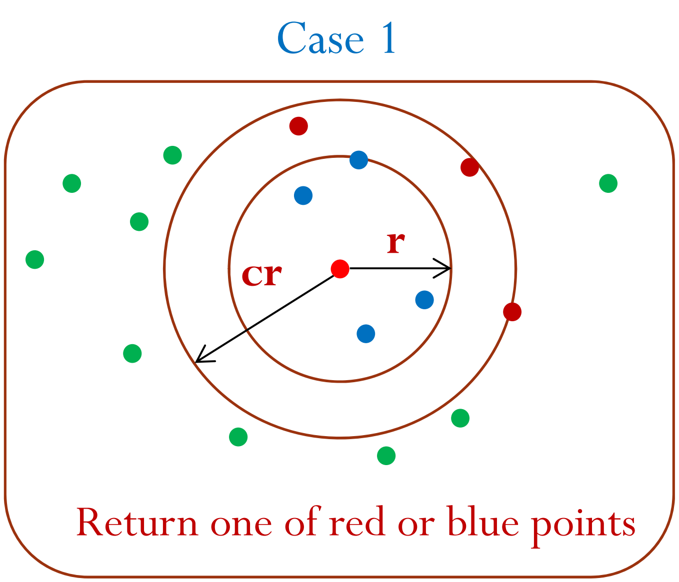
- Randomized **c**-approximation **r**NNS:
  - Given a data set $\mathbf{S} \subset \mathbf{R}^d$, a distance function $\mathbf{d(x, y)}$, parameters $\mathbf{r} > \mathbf{0}$, $\mathbf{c} > \mathbf{1}$, $\boldsymbol{\delta} > \mathbf{0}$, and a query $\mathbf{q} \in \mathbf{R}^d$, with probability $\mathbf{1 - \delta}$
    - If exists $\mathbf{x} \in \mathbf{S}$ and $\mathbf{d(x, q)} \leq \mathbf{r}$, return some point $\mathbf{x'} \in \mathbf{S}$ where $\mathbf{d(x', q)} \leq \mathbf{cr}$.
    - Else, return nothing.

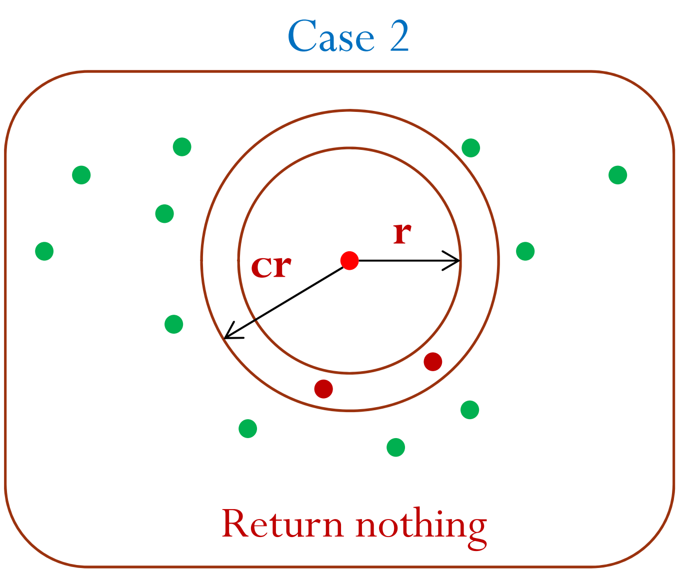




Correctness

- We need to show that, with probability $1 - \delta$:
  - Case 1: If exists $\mathbf{x} \in \mathbf{S}$ and $\mathbf{d(x, q)} \leq \mathbf{r}$, there exists a hash table $\mathbf{T_j}$ such that $\mathbf{g_j(x) = g_j(q)}$ for some $\mathbf{1 \leq j \leq L}$. This means that we can find $\mathbf{x}$.
  - Case 2: The total number of collisions from far away points is at most **2L**. This means that after checking **2L** points, we are certain that all points are far away from $\mathbf{q}$ more than $\mathbf{cr}$.





Correctness: Case 1

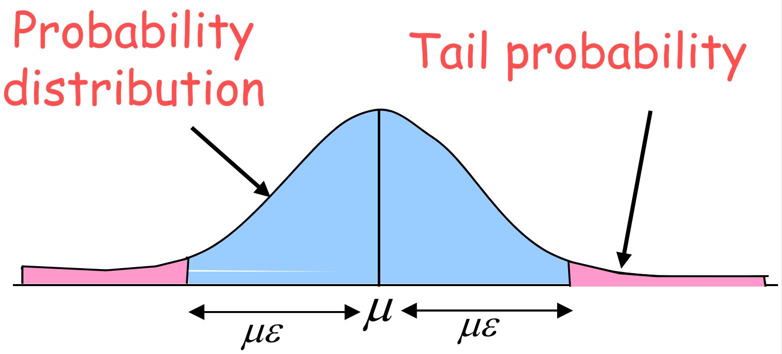
- Observation: If $\mathbf{d(x, q) \le r}$, **then**
  - Prob. $\mathbf{x}$ colliding with $\mathbf{q}$ in any hash table: $\ge \mathbf{1/n^{\rho}}$
  - Prob. $\mathbf{x}$ not colliding with $\mathbf{q}$ in any hash table: $\le \mathbf{1 - 1/n^{\rho}}$
  - Prob. $\mathbf{x}$ not colliding with $\mathbf{q}$ in all $\mathbf{L}$ hash tables: $\le \mathbf{(1 - 1/n^{\rho})^{L}}$
  - Prob. $\mathbf{x}$ not colliding with $\mathbf{q}$ in all $\mathbf{L}$ hash tables: $\mathbf{1/e}$ with $\mathbf{L = n^{\rho}}$
  - Prob. $\mathbf{x}$ colliding with $\mathbf{q}$ in at least $\mathbf{1}$ hash tables: $\mathbf{1 - 1/e}$ with $\mathbf{L = n^{\rho}}$

$(1 - 1/x)^{x} \approx 1/e$

- With prob. $\mathbf{1\text{-}1/e}$, any near point $\mathbf{x}$ will collide with $\mathbf{q}$ in at least $\mathbf{1}$ hash table. This means that we can find $\mathbf{x}$.

Correctness: Case 2

- Observation:
  - Maximum number of far away points: **n**
  - If **d(x, q)** ≥ **cr**, then **Pr [h(x) = h(q)]** ≤ **1/n**. Hence, in expectation, we have at most **1** far away point collide with **q** in each hash table.
  - **E [number of far away points] ≤ L.**



$$\Pr[X \geq a] \leq \frac{E[X]}{a} \text{ for any } a > 0.$$

- Using Markov's inequality:
  - Let **X** be the number of far away points collided with **q** in **L** tables.
  - $\Pr[X \geq 2L] \leq \frac{E[X]}{2L} \leq 1/2$

- With prob. **1/2**, after checking all first **L' = 2L** points far away from **q**, we are certain that all points are far away. Hence, return nothing.

Correctness

- Probability both case 1 and case 2 hold is at least:

$$\mathbf{1 - ((1 - \underset{\uparrow \text{ Prob. case 2 fails}}{1/2}) + (1 - \underset{\uparrow \text{ Prob. case 1 fails}}{(1 - 1/e)})) = 1/2 - 1/e}$$

- Theorem: There exists a sublinear algorithm to answer the **c**-approximate **r**NNS with probability **1/2 – 1/e**.

- Boosting the accuracy: By repeating the LSH algorithm **O(1/δ)** times, we can amplify the probability of success in at least **1** trial to **1 - δ** for any **δ > 0**.

Practical algorithm

- Query for a point **q**:

For each $j = 1, 2, \ldots, L$
1) Retrieve the points from the bucket $g_j(q)$ in the T_j hash table.
2) For each of the reported point x
    2.1) Compute the distance $d(x, q)$.
    2.2) If $d(x, q) \leq r$, return x.
    2.3) Update the k-nearest neighbors with the returned x.

- We cannot have sublinear theoretical guarantee but the accuracy is much higher. In practice, it can run in sublinear time in many of data sets.