Approximate Query Processing

- Space Partitioning-based
  - Tree
  - Encoding
  - Locality Sensitive Hashing
- Graph-based Methods

Notes:
- Recent works mainly in the Database area
- Prefer ease of exposition over rigor
- Categorization is not fixed/unique

Locality Sensitive Hashing (LSH)

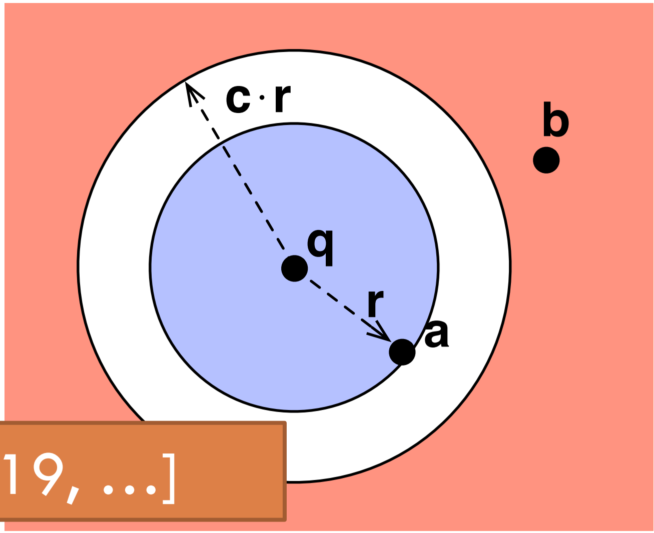
- From the persective of collision probability
  - (Ordinary) hash function h:
    - Pr[h(x) = h(y)] = ε, if x ≠ y
  - LSH
    - Pr[h(x) = h(y)] increases with locality
    - Randomness comes from r.v. h ∈ H

c.f., Cryptographic hash functions
Pr [h(x) = h(y)] = 2^{-m}, if
Hamming(x, y) = 1

(r_1, r_2, p_1, p_2)-sensitive [IM98]

- Pr[h(x) = h(y)] ≥ p_1 , if dist(x, y) ≤ r_1
- Pr[h(x) = h(y)] ≤ p_2 , if dist(x, y) ≥ r_2

Pr[h(x) = h(y)] = sim(x, y) [C02] too narrow



Can be generalized [SWQZ+14, ACPS18, CKPT19, …]

LSH is the **best** approach using **sub-quadratic** space

Locality Sensitive Hashing (LSH)

- Equality search
  - Index: store o into bucket h(o)
  - Query:
    - retrieve every o_i in the bucket h(q)
    - verify if o_i = q

Pr[h(q) = h(o)] = 1/B

- LSH
  - ∀h ∈ *LSH-family*, **Pr**[Q(h(q)) = Q(h(o))] = f(Dist(q, o))
    - Q( ): quantization (not essential)
    - "Near-by" points have more chance of colliding with q than "far-away" points
  - Similar index & query procedures, with a weak probabilistic guarantee

c.f., [PIM12] for the rigorous QP procedure

➔ Repeat to boost the guarantee

LSH Families

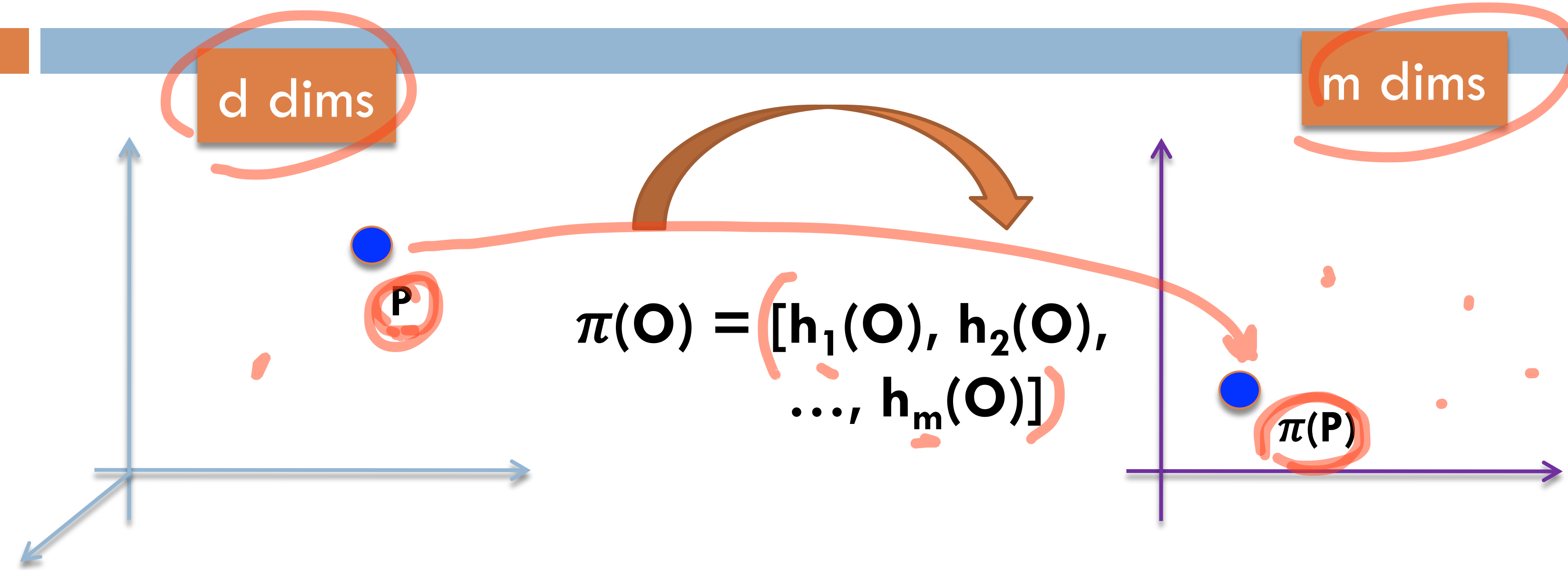
- Many are known
  - L_p $(0 < p \leq 2)$: use p-stable distribution to generate the projection vector
    - For L_2, just use random Gaussian vector
    - Other families exists, e.g., sparse random projection
  - Angular distance (arccos): SimHash
  - Jaccard: minhash (based on random permutation)
  - Hamming:
    - random projection
    - covering LSH

Comments

- New queries can be reduced to known LSH cases
  - Maximum inner product search (MIPS)
  - Set containment
  - Group aggregated query
- Related to various distortion-bounded embedding
  - Edit distance: CGK-embedding to Hamming with O(K) distortion

Probabilistic Mapping

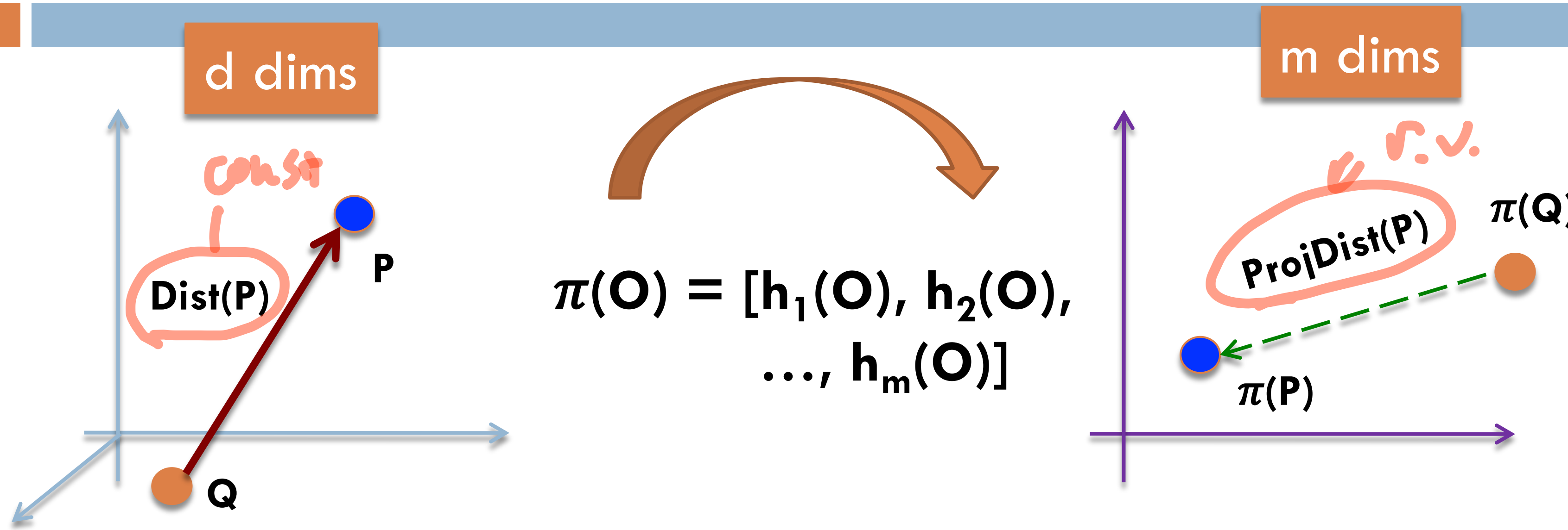
$d >> m$



$$\pi(O) = [h_1(O), h_2(O), \ldots, h_m(O)]$$

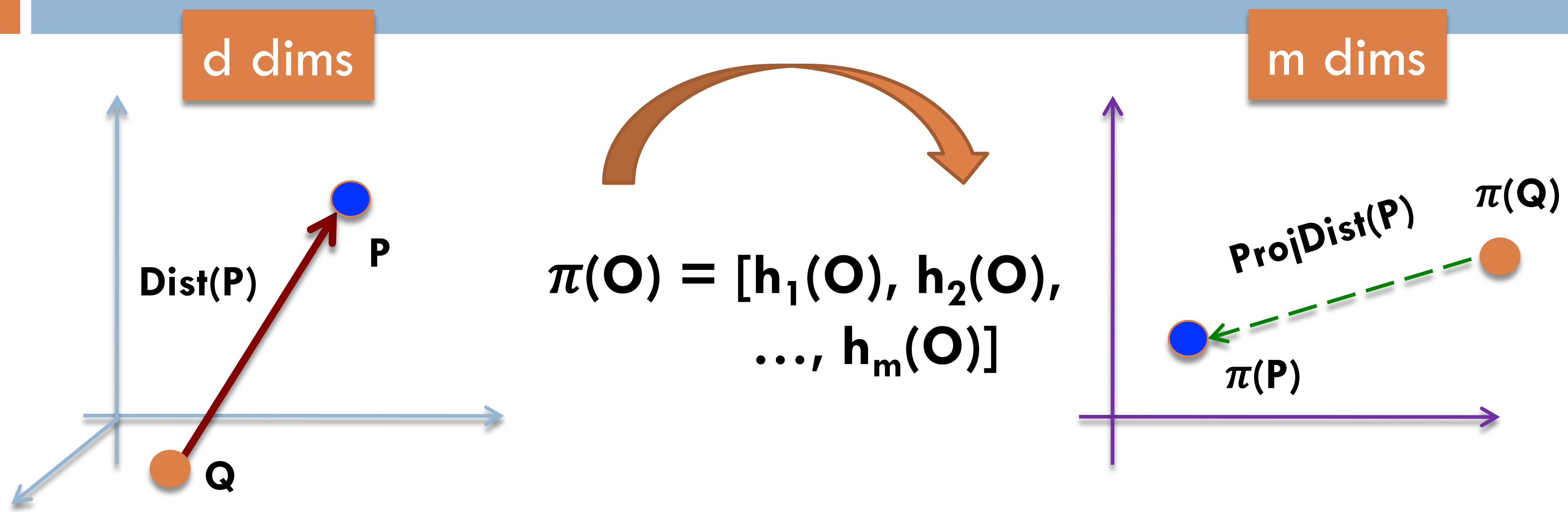
- Probabilistic, linear mapping from the original space to the projected space

Probabilistic Mapping



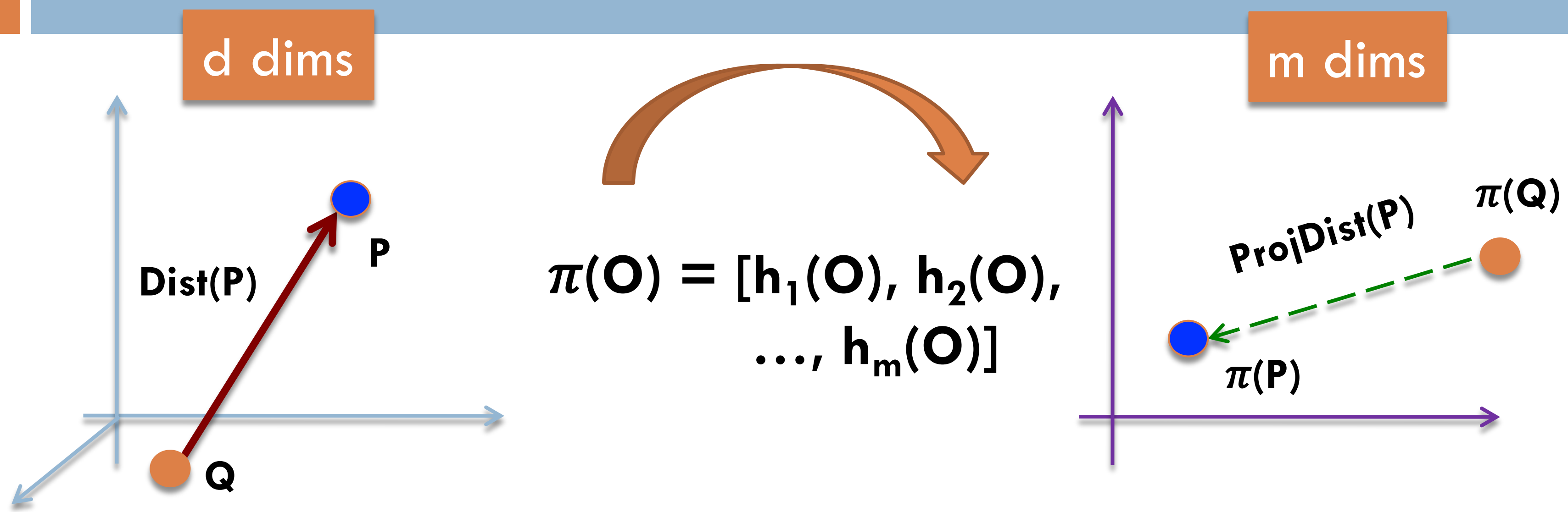
- Probabilistic, linear mapping from the original space to the projected space
- What about the distances (wrt Q or π(Q)) in these two spaces?

Probabilistic Distance Tracking Property of the Mapping



- ProjDist(P)^2 ~ Dist(P)^2 * χ^2_m [SWQZ+14]
  - ProjDist(P)^2 can be computed (incrementally) from h_i(P) and h_i(Q) due to the linearity of the hash function
  - Can be generalized to other p-stable LSH functions

Probabilistic Distance Tracking Property of the Mapping



$$\pi(O) = [h_1(O), h_2(O), \ldots, h_m(O)]$$

LSH provides a probabilistic distance-preserving mapping between the two spaces

Johnson & Lindenstrauss Lemma [JL84] only works for L2 and induces a method that requires more space than LSH [AIR18]

Roadmap

- Roadmap
  - Practical LSH methods (i.e., linear index complexity)
  - Data-dependent LSH methods

$O(n^{1+\rho})$

New Perspectives

- Inference Method
- Access method
- Stopping condition

Inference

- Problem 1:
  - Given that ProjDist(P) $\leq$ r, what can we infer about Dist(P)?

$\text{ProjDist}(P)^2 \sim \text{Dist}(P)^2 * \chi^2_m$

- SRS:

Similar to the usual (r_1, r_2, p_1, p_2) definition of LSH

E_1
E_2

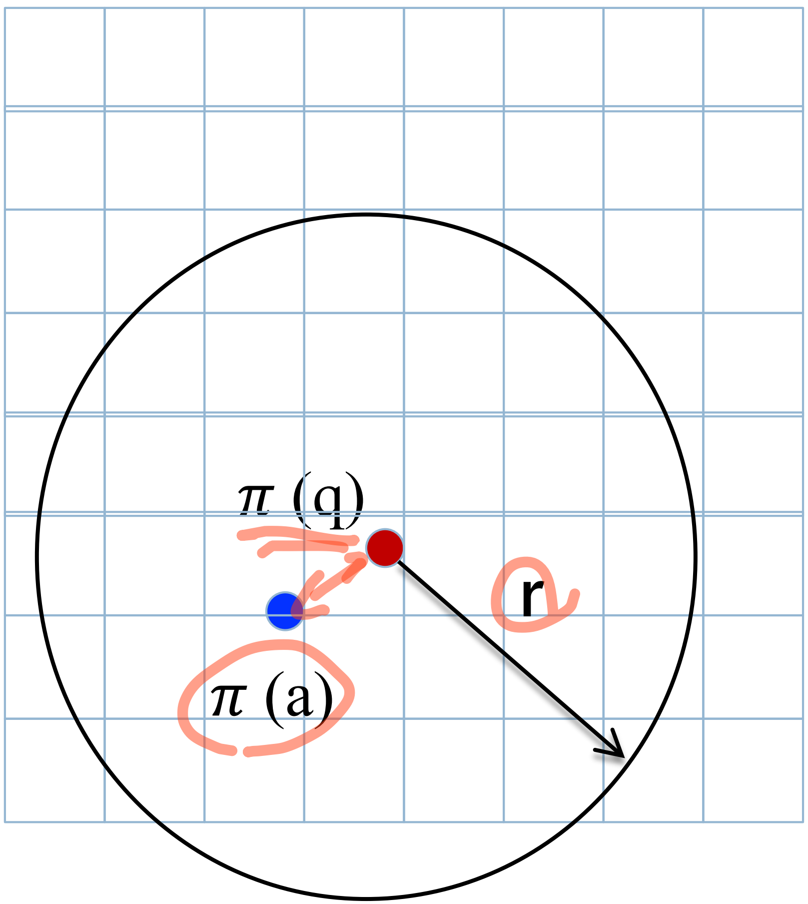
  - If Dist(P) $\leq$ R, then Pr[ProjDist(P) $\leq$ r] $\geq \Psi_m((r/R)^2)$
  - If Dist(P) > cR, then Pr[ProjDist(P) $\leq$ r] $\leq \Psi_m((r/cR)^2) = t$
  - (some probability) at most O(tn) points with ProjDist $\leq$ R
  - (constant probability) one of the O(tn) points has Dist $\leq$ R

- This solves the so-called (R, c)-NN queries → returns a c^2 ANN
- Using another algorithm & proof → returns a c-ANN

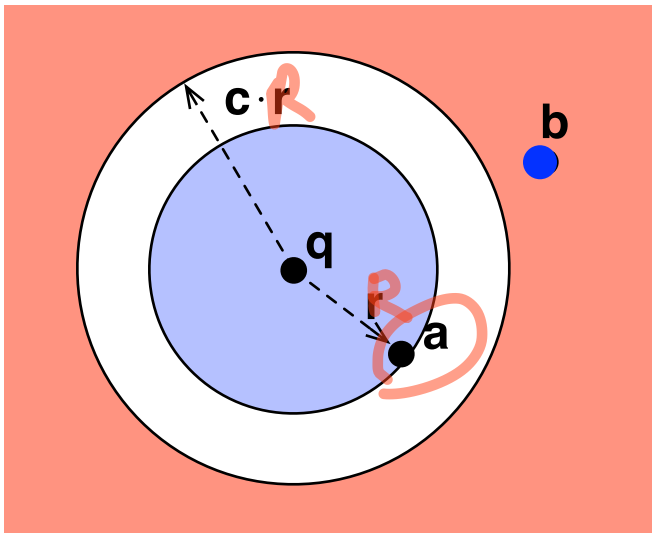
Inference requires <u>precise & complete</u> information of the projections

Near Points

Likely ($\geq p_1$)



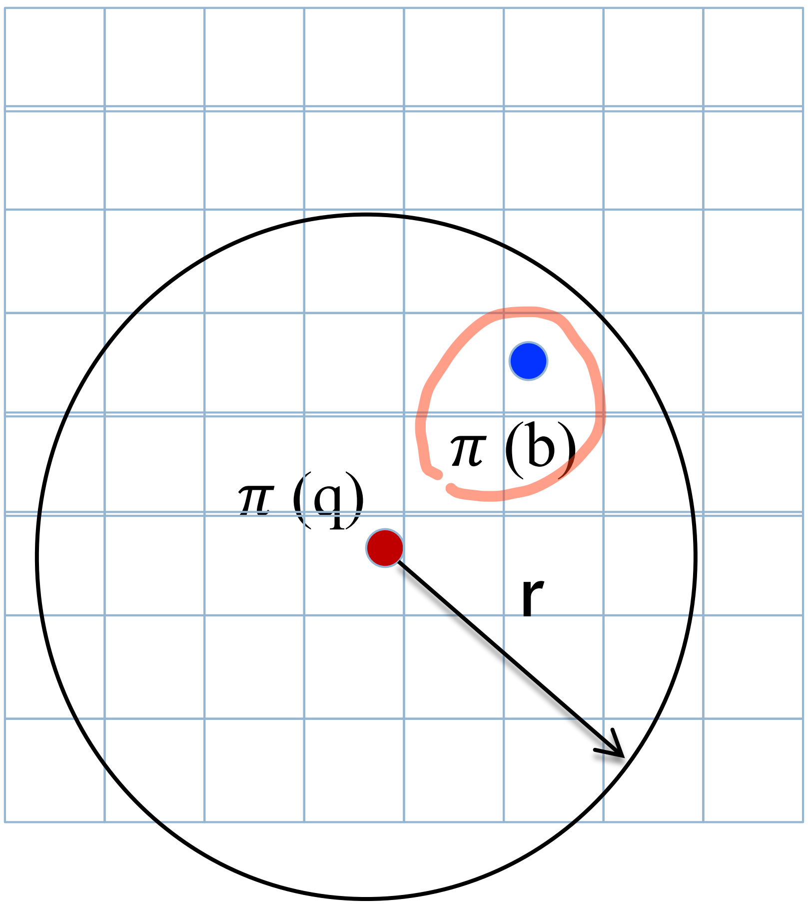
Dist(a) ≤ R



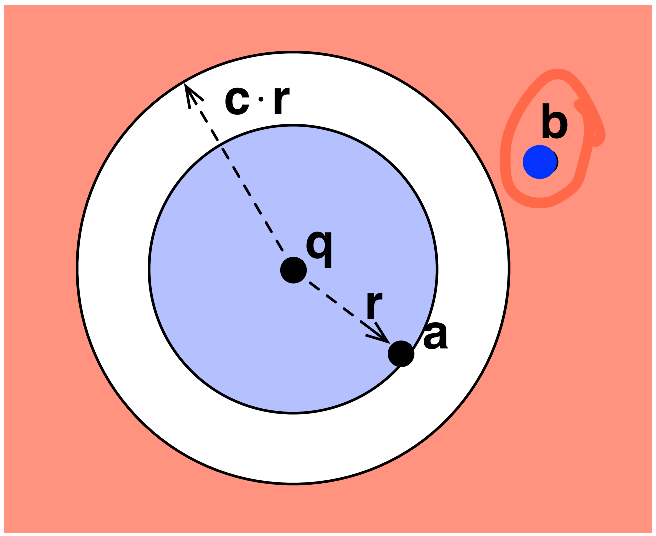
d-dimensional space

Faraway Points

Unlikely ($\leq p_2$)



Dist(b) $\geq$ cR

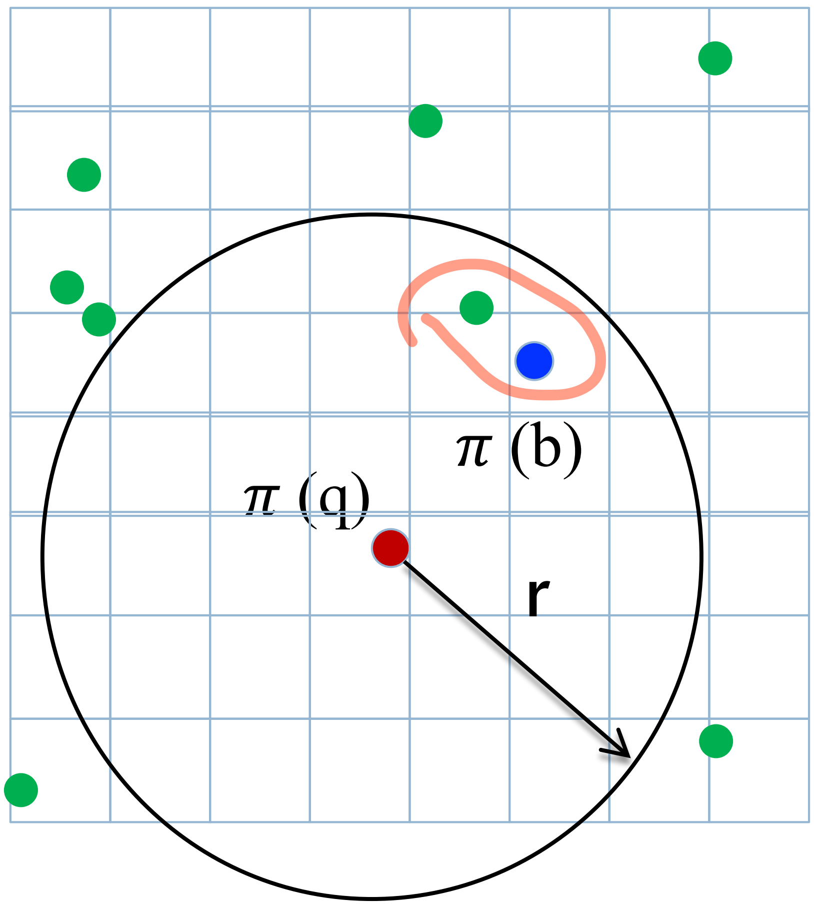


d-dimensional space

Consider all faraway Points

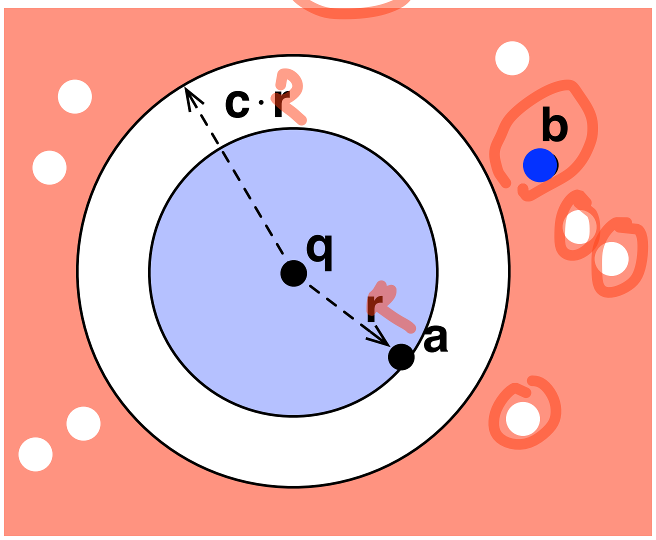
Expected to see few of them

$E \leq P_2 \cdot (n-1)$



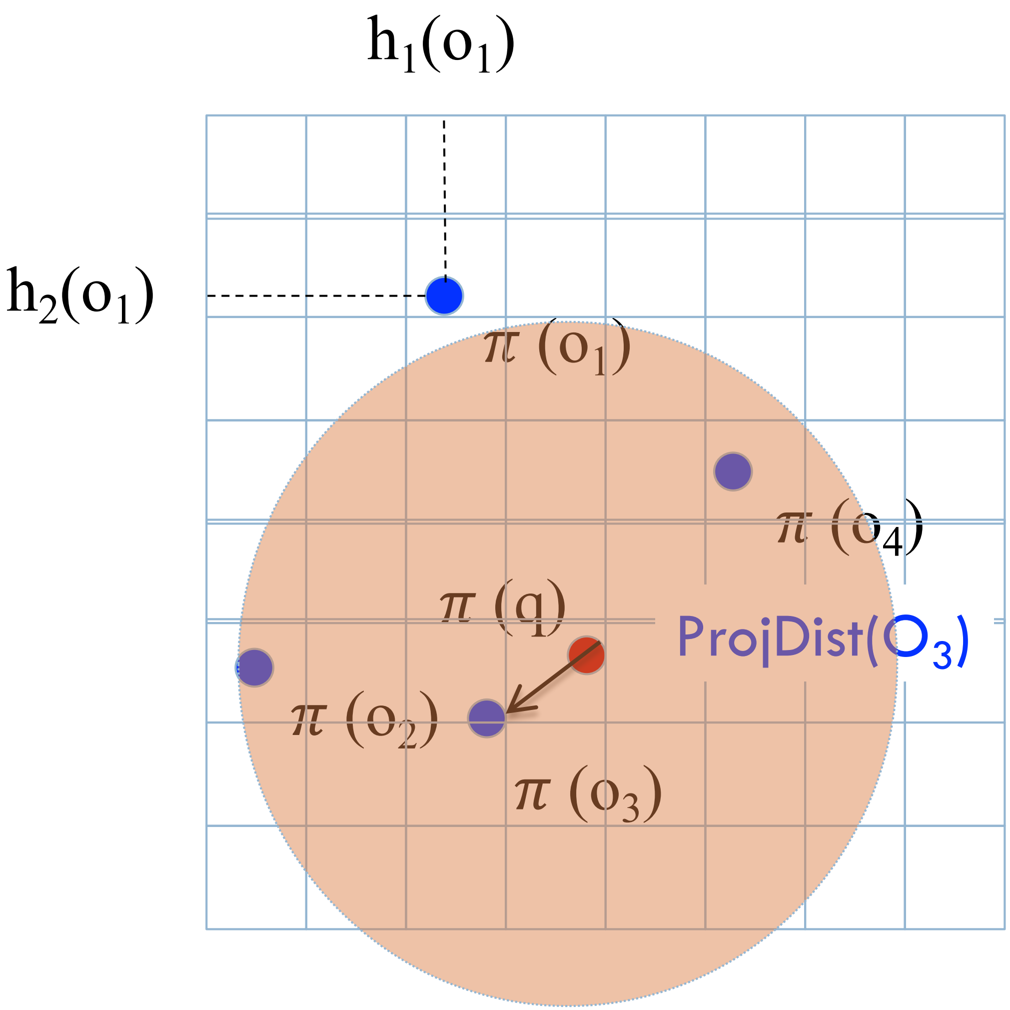
Dist(b) $\geq$ cR

n-1



d-dimensional space

Exact t*n-NN Query in m-dim Space



if tn = 2, then one of the top-3 NNs in the projected space around π(Q) is a c-ANN with constant probability

ProjDist(o_3) is the mininum among the 4 points

Inference

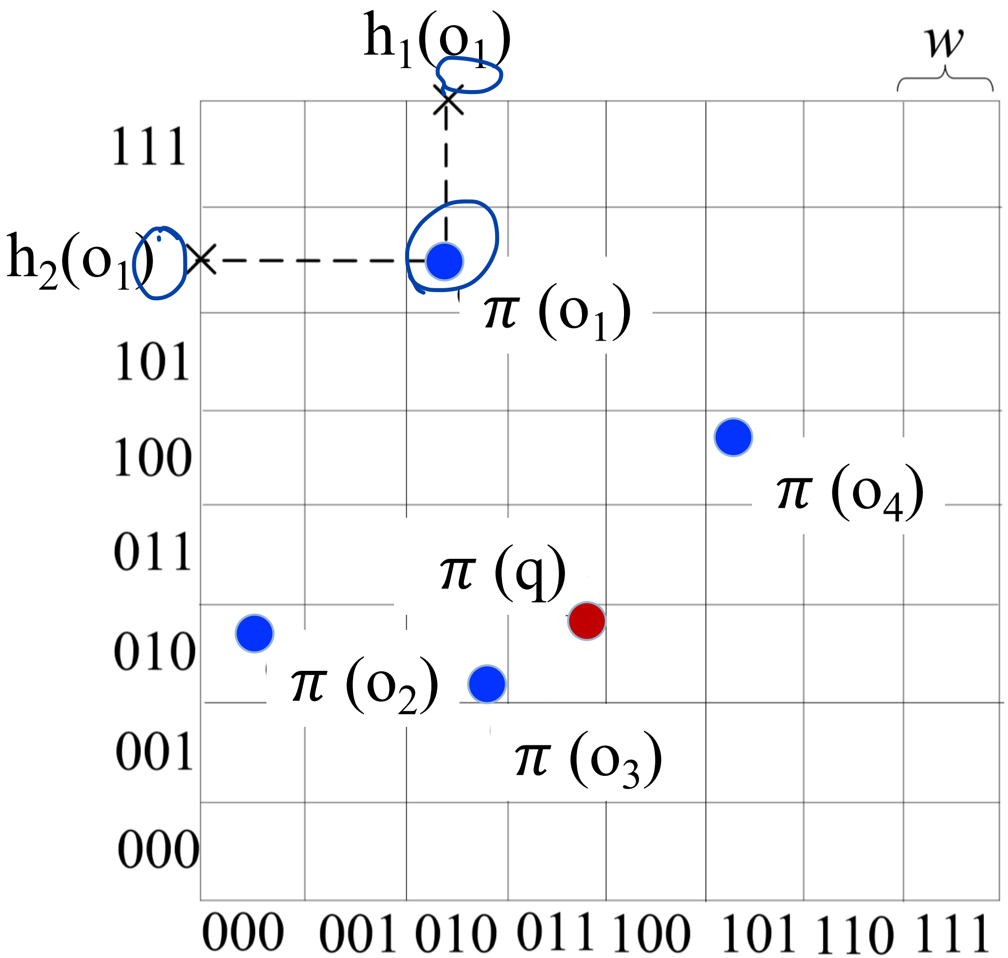
- Problem 2:
  - Given that $z(\pi(P))$ is similar to $z((\pi(Q))$, what can we infer about Dist(P)?

Measured by LLCP($z(\pi(P))$, $z((\pi(Q))$)

- LSB:
  - E_1 If Dist(P) $\le$ R, then Pr[LLCP(P, Q) $\ge \delta$] $\ge p_1^m$
  - E_2 If Dist(P) $>$ 2R, then Pr[LLCP(P, Q) $\ge \delta$] $\le p_2^m$
  - (some probability) at most $O(p_2^m\, n)$ points with ProjDist $\le$ R
  - (constant probability) one of the $O(p_2^m\, n)$ points has Dist $\le$ R

Inference requires precise & complete information of about z() values

z-order



$z(\pi(o_1)) = 01\ 11\ 00$

$z(\pi(Q)) = 00\ 11\ 10$

$z(\pi(o_3)) = 00\ 11\ 00$

$LLCP(o_1, Q) = 1$

$LLCP(o_3, Q) = 4$

Inference

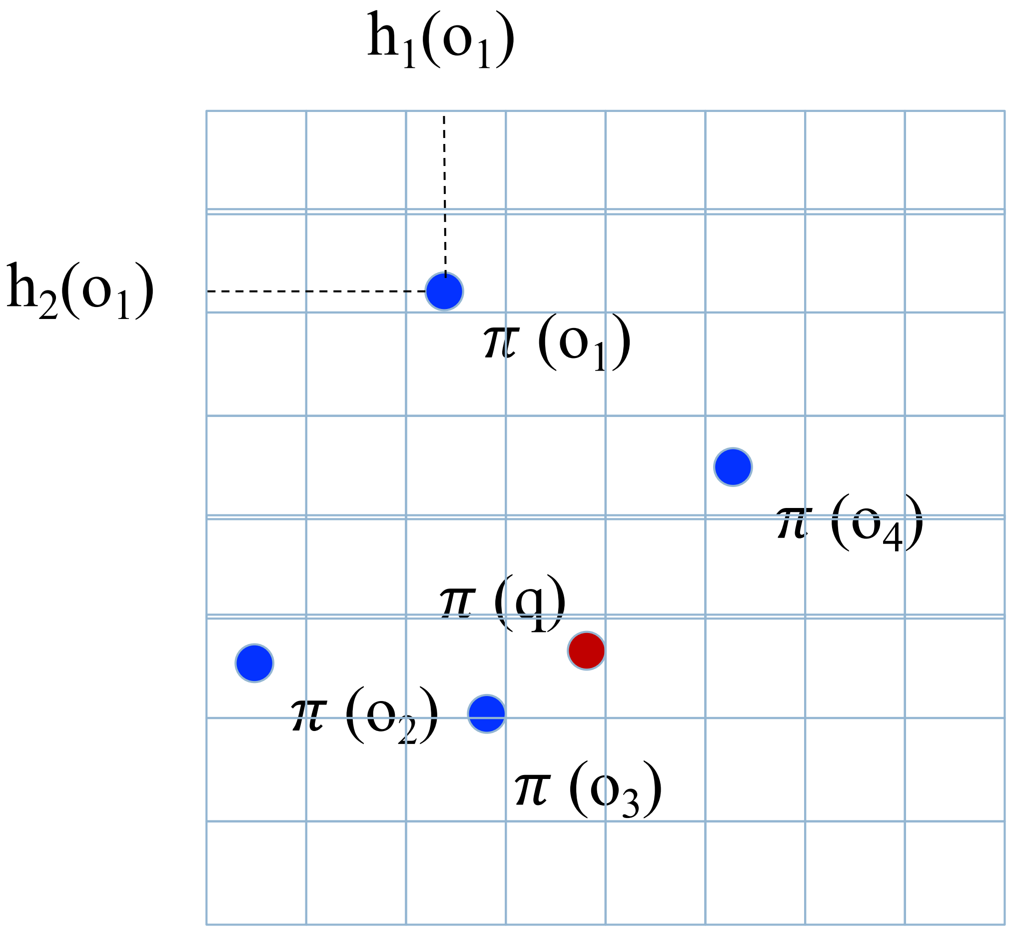
- Problem 3:

  Collision wrt w: if $|h_i(P) - h_i(Q)| \leq w$

  - Given that P's #collision $\geq \alpha m$, what can we infer about Dist(P)?
- C2LSH/QALSH:
  - E_1 If Dist(P) $\leq$ R, then Pr[#collision $\geq \alpha m$] $\geq \gamma_1$
  - E_2 If Dist(P) $>$ cR, then Pr[#collision $\geq \alpha m$] $\leq \gamma_2$
  - (some probability) at most $O(\gamma_2 * n)$ points with #collision $\geq \alpha m$
  - (constant probability) one of the $O(\gamma_2 * n)$ points has #collision $\geq \alpha m$

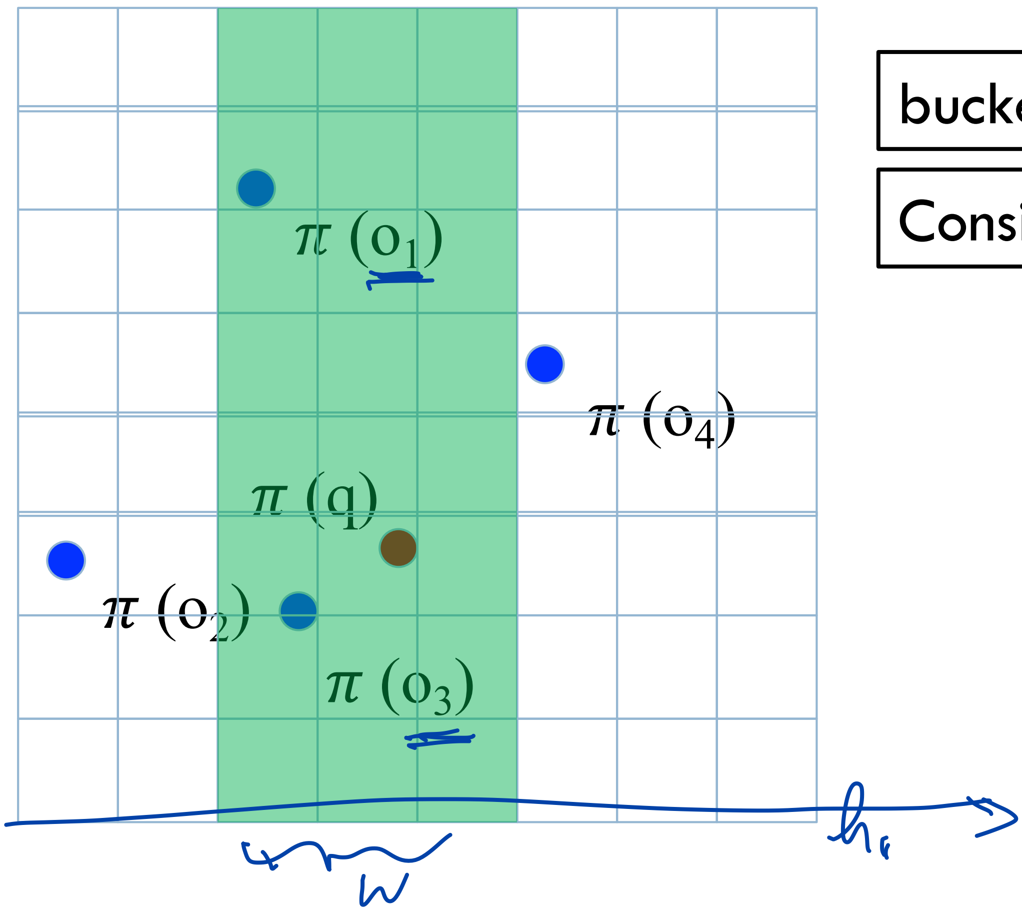
Inference requires rough & incomplete information of the projections

Collision count



bucket width = 4

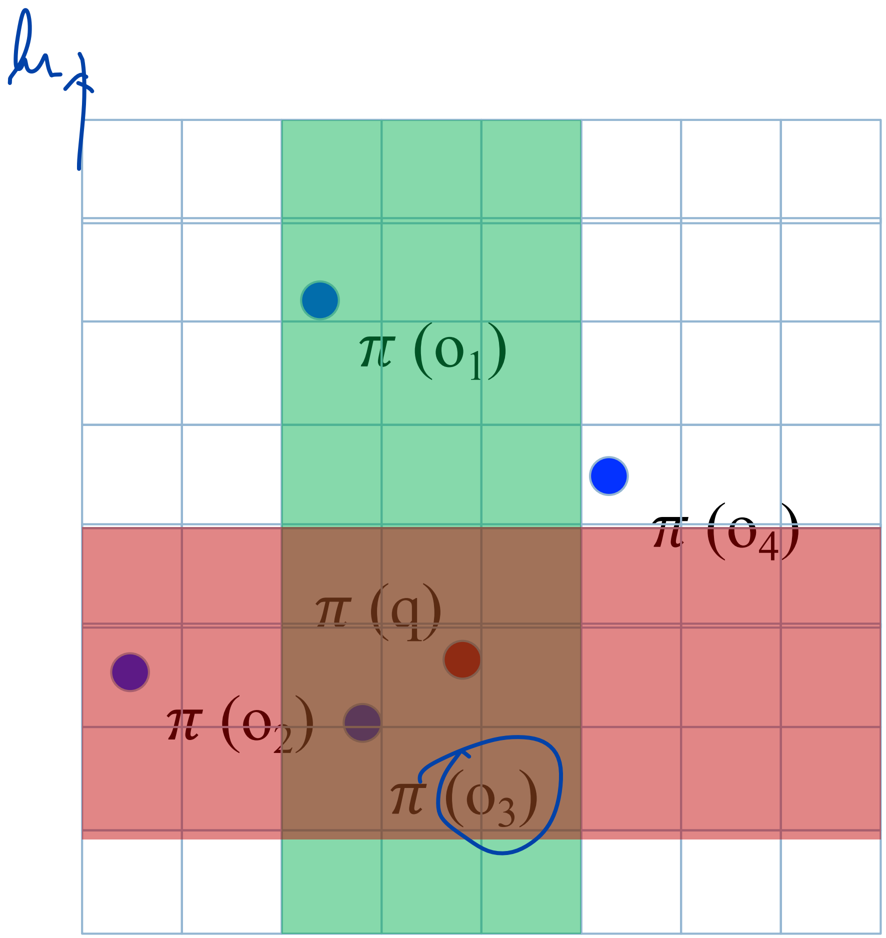
Collision count



bucket width = 3

Consider $h_1()$ axis

Collision count



bucket width = 3

Consider $h_1()$ axis

Consider $h_2()$ axis

#Collision(o_1) = 1
#Collision(o_2) = 1
#Collision(o_3) = 2
#Collision(o_4) = 0

Some Variants

- Looseness in C2LSH
  - γ_1 and γ_2 computed using tail bounds
  - Constant probability obtained using the union bound
  - Contrast between γ_1 and γ_2 is low by using QZ(h_i(P)) = QZ(h_i(Q)) as the collision
- QALSH
  - Use the right collision definition (virtual bucketing)
- PDA-LSH [YDSS20]
  - Computes γ_1 and γ_2 using Gaussian as an approximation
  - Approximately compute $\Pr[E_1 \wedge E_2]$

Inference

E

Collision wrt w: if $|h_i(P) - h_i(Q)| \le w$

- Problem 4:
  - Given that P's #collision $\geq \alpha m$, what can we infer about x $\triangleq$ Dist(P) ?

Requires assumption or tolerance of a prior

- Bayesian LSH:
  - $\boxed{Pr[\ x \mid E\ ]} = Pr[\ E \mid x\ ] * \boxed{Pr[x]}\ /\ Pr[E]$

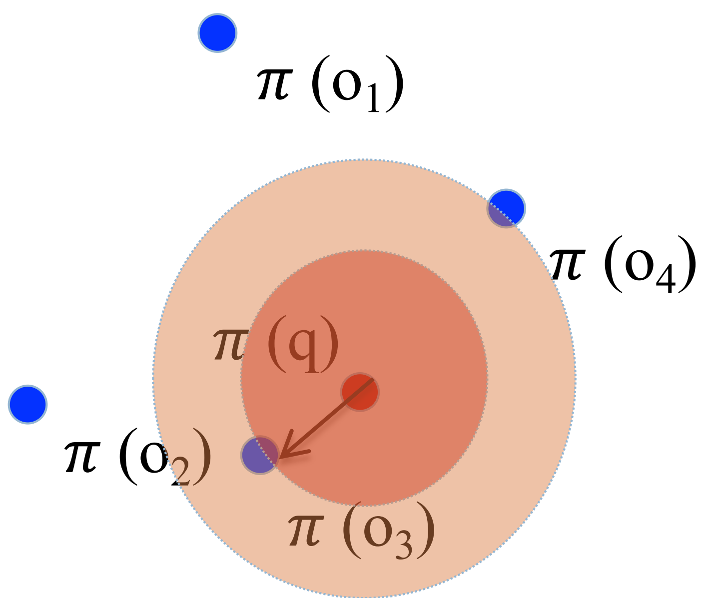
Posterior distribution

  - Then, one can calculate many things
    - $Pr[\ x \geq R \mid E]$
    - MAP estimate $x^\wedge = \mathrm{argmax}_x\ Pr[\ x \mid E]$
    - Bayesian Tail probabilities: $Pr[\ |x^* - x^\wedge| > \varepsilon \mid E]$

Inference requires rough & incomplete information of the projections

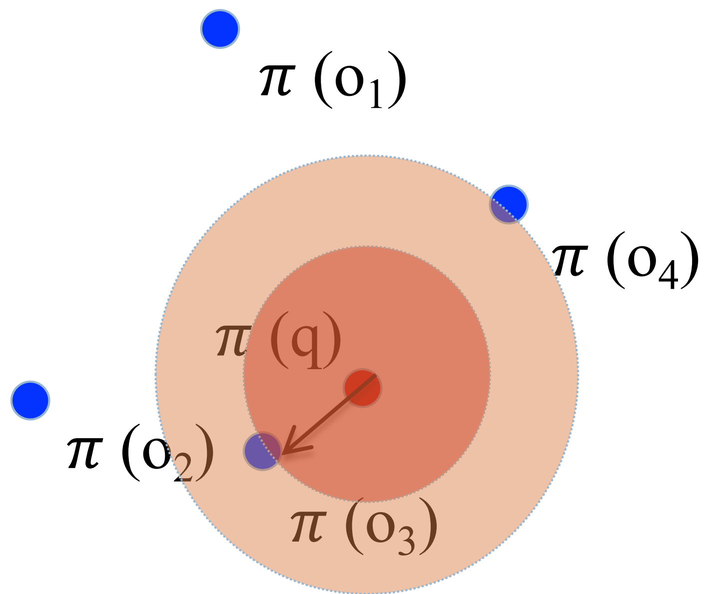
Access Method

- Inference method ➔ Access method
- SRS requires accessing projected points according to increasing ProjDist ➔ R-tree (on disk) or Cover Tree (in memory)
  - m cannot be too large (e.g., m in [6, 8]) for R-tree



Access Method

- Inference method ➔ Access method
- Replace R-tree in SRS by a variant of the M-tree → PM-LSH
  - Allow m to be reasonably large (e.g., 15)
  - Uses distortion-based inference (partially)

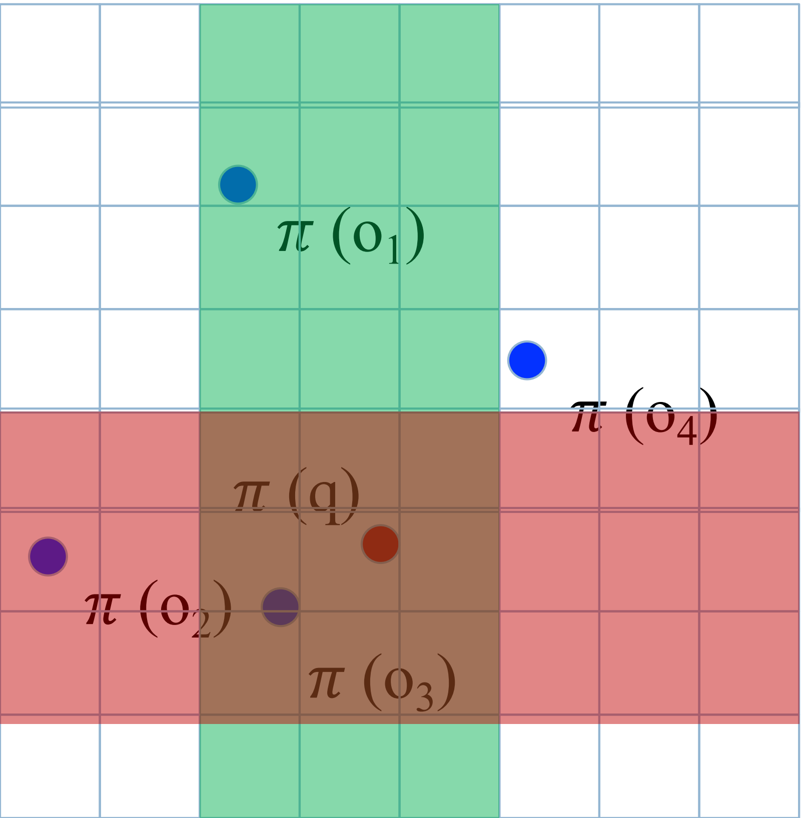


Access Method

- Inference method 🡐 Access method
- Use a new definition of bucket and collision ➔ R2LSH
  - Bucket in 2d subspace ≜ Ball of radius w centered at $\pi_i(Q)$, i = 1, …, m/2
  - Use the SRS-2 style stopping condition
  - Use a new index based on polar coordinates

Access Method

- Inference method ← Access method
- C2LSH requires accessing each projection with a bucket width constraint → B-tree

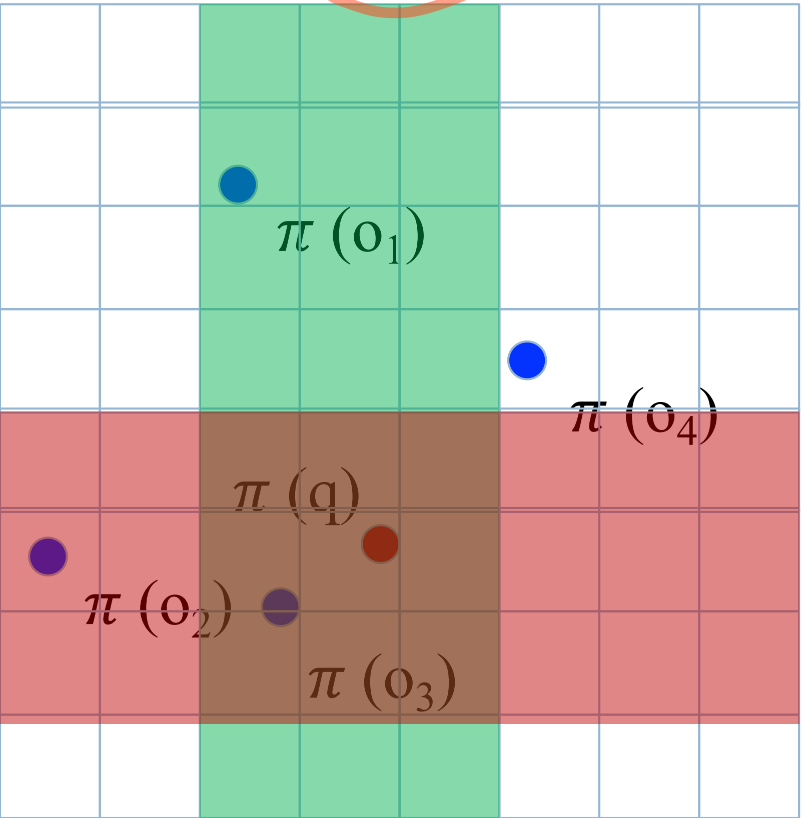


bucket width = 3
#Collision threshold = 2
Only o_3 is a candidate

However, we do have partial information about o_1 and o_2

Access Method

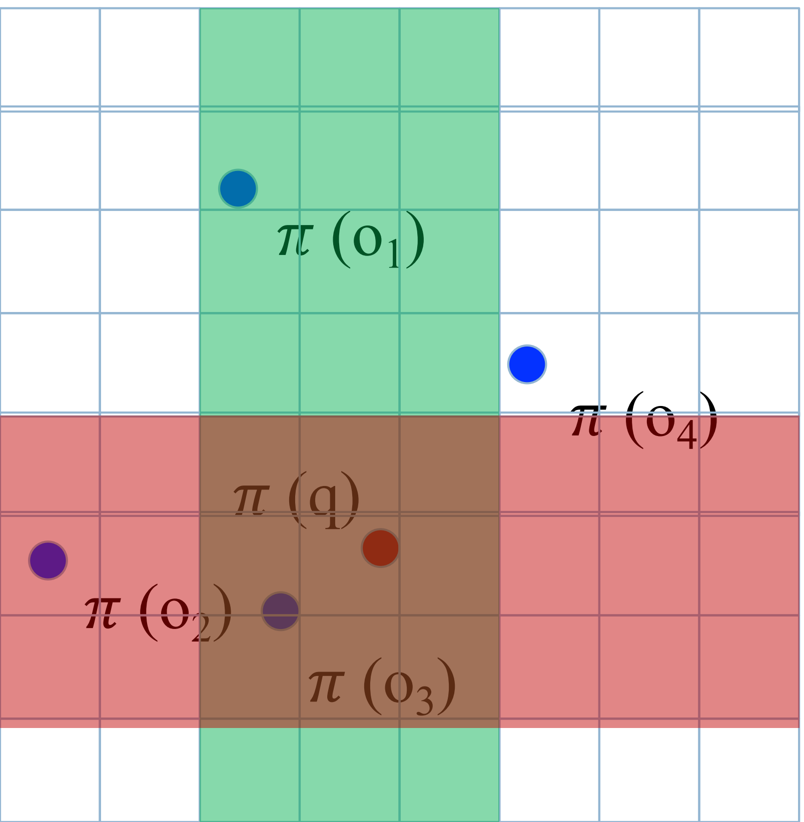
- Inference method ← Access method
- Make use of almost ALL accessed points in QALSH
  → VHP



Object	#Collisions	Partial ProjDist	
o_1	1	3.2	
o_2	1	3.5	
o_3	2	1.3	

Access Method

- Inference method ← Access method
- Make use of almost ALL accessed points in QALSH → VHP



Object	#Collisions	Partial ProjDist	PPDist Threshold
o_1	1	3.2	3.4
o_2	1	3.5	3.4
o_3	2	1.3	1.9

Assuming the Partial ProjDist thresholds for different #Collisions, both o_1 and o_3 are candidates

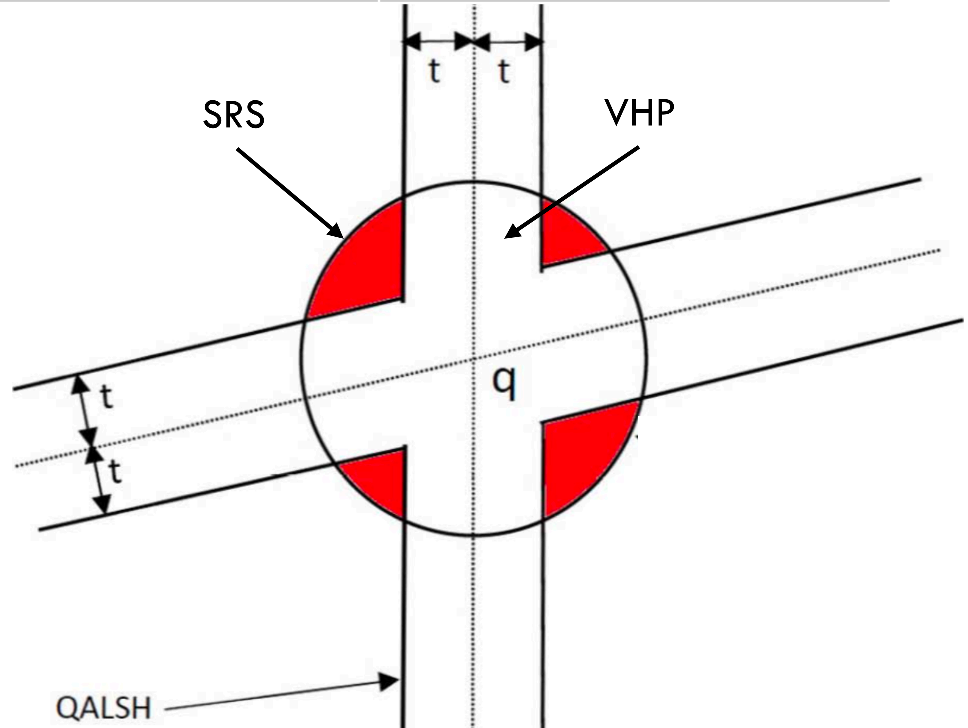
Some Comparisons

- Candidate Conditions

Method	Collision Count	(Observed) Distance	Max Candidates
SRS	= m	$\leq$ r	T
QALSH	$\geq \alpha$m	n/a	βn
VHP	$\geq$ i (i = 1, 2, …, m)	$\leq l_i$	βn

- Candidate Regions

VHP = SRS ∩ QALSH



Stopping Condition

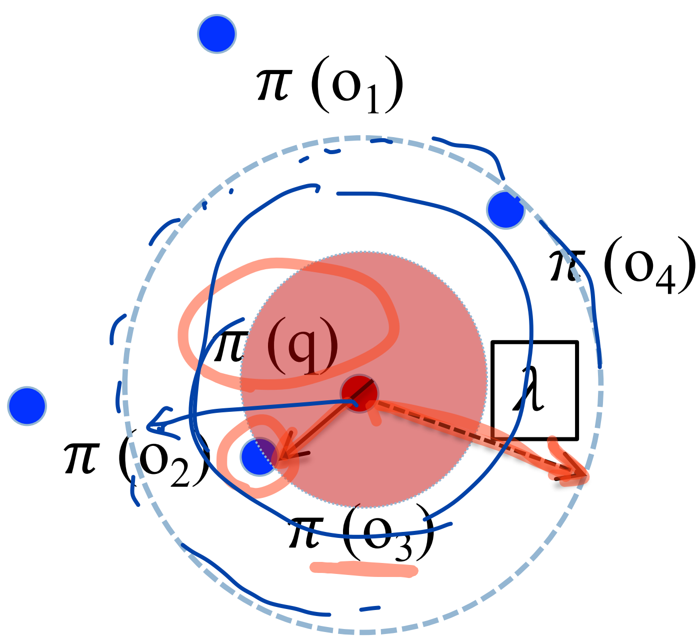
- Traditional LSH, C2LSH, SRS-1
  - Solve $(c^k r, c)$-NN queries, $k = 0, 1, \ldots$
    - limitations:
      - only c^2 approximate ratio
      - cannot support $c = 1$
  - Stopping on either condition:
    - E_1 a candidate has distance $\leq c^k R$
    - E_2 there are more than "enough" candidates found

Stopping Condition

- SRS-2 and R2LSH
  - Accessing objects by increasing order of their ProjDist
  - Keep the o_{min} which has the smallest Dist so far
  - Stop when ProjDist ≥ λ Dist(O_{min})

$$\lambda = \frac{1}{c}\sqrt{\Psi_m^{-1}(p_\tau)}$$

Works even for c = 1 !!



Assume Dist(o_3) = 1, then SRS-2 at most scans a hypersphere of radius λ

This hypersphere is monotonically shrinking (as we found better o_{min})

Stopping Condition

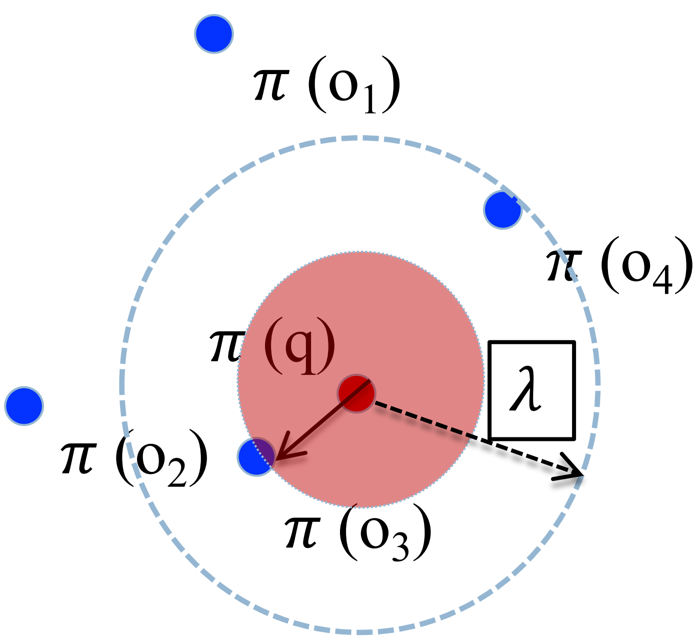
Distance Distortion → LSH

Probabilistic Mapping

- SRS-2 and R2LSH
  - Accessing objects by increasing order of their ProjDist
  - Keep the o_{min} which has the smallest Dist so far
  - Stop when ProjDist ≥ λ Dist(O_{min})

$$\lambda = \frac{1}{c}\sqrt{\Psi_m^{-1}(p_\tau)}$$

Works even for c = 1 !!



Assume Dist(o_3) = 1, then SRS-2 at most scans a hypersphere of radius λ

This hypersphere is monotonically shrinking (as we found better o_{min})

Stopping Condition

- I-LSH (upon QALSH)
  - Solve (r_k, c)-NN queries, where the r_k sequence is obtained according to the data near π(Q)
    - obtains c-ANN
  - Stopping on either condition:
    - Dist(o_{min}) $\leq \lambda$ r, where 2r is the "current" virtual bucket width
    - there are more than "enough" candidates found

Comment

- Easy to relax the LSH method in practice at the cost of no worst-case guarantees
  - E2LSH: use fewer number of random projections
  - Multiprobe LSH (entropyLSH and other variants): space-time tradeoff
  - LSH in practice: use empirically tuned parameters (k, l)
  - HD-index: space filling curves as pseudo-LSH functions
  - SK-LSH: Replace LSB-tree/forest by a dimension-wise linear mapping
  - …

Data-sensitive Hashing

- LSH is data-insensitive
  - Indexing hyper-parameters determined by the shape of the data only
  - Indexing parameters are randomly generated
- Efforts to make data-sensitive, LSH-like methods
  - [AR15]
    - Aim: break the lower bounds of ρ
  - DSH
  - OPFA / NeOPFA
- Learning-to-hash methods
  - NSH [PCM15]
  - [LYZX+18] and many in the ML/CV communities

c.f., [AIR18]

c.f., https://learning2hash.github.io and https://cs.nju.edu.cn/lwj/L2H.html

DSH [GJLO14]

- Learn a family of (hash) functions, H, that preserves kNN of queries

1. Training data:

$$\mathbf{W}_{ij} = \begin{cases} 1 & \text{, if } o_j \in kNN(q_i) \\ -1 & \text{, if } o_j \notin kNN(q_i) \wedge o_j \text{ is sampled} \\ 0 & \text{otherwise.} \end{cases}$$

   - sampled queries
   - their k-NN objects (+ve)
   - samples non-c*k-NN objects (-ve)

2. Function family:
   - Thresholded linear functions $h(\mathbf{x}; \mathbf{a}) = \mathrm{sgn}(\mathbf{a}^\top \mathbf{x})$

3. Learn one hash function

$$\arg\min_h \sum_i \sum_j \ell(q_i, o_j)\, \mathbf{W}_{ij} \quad \text{, where } \ell(q, o) = (h(q) - h(o))^2$$

DSH [GJLO14]

- Learn a family of (hash) functions, H, that preserves kNN of queries
  4. Learn multiple hash functions
     - Multiplicative updates on W_{ij}
       - Increase W_{ij} if incorrectly classified
       - Decrease W_{ij} if correctly classified
     - (Under some assumptions) obtain H that satisifies the (k, ck, p_1, p_2)-sensitive property for the training data
       - if $o \in NN(q, k)$, then collision probability from H $\geq p_1$
       - if $o \notin NN(q, ck)$, then collision probability from H $\leq p_2$

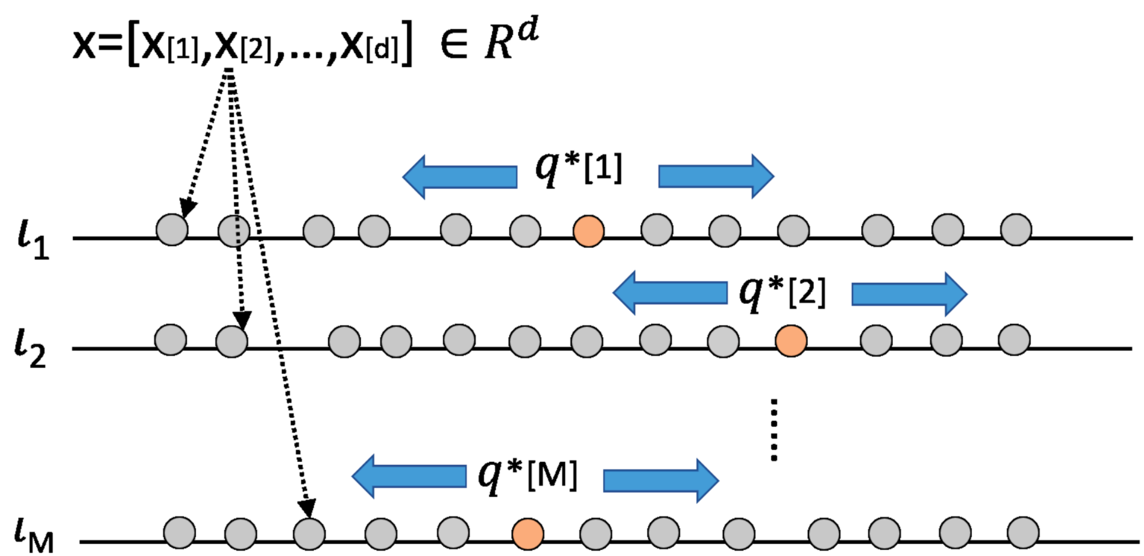
Key difference with AdaBoost: High recall and low precision

Learned ANN Index [LZSW+20]

- Focus on external I/O
  - Use B-trees and maximize the use of sequential I/Os
- Scheme:
  - H: $R^d \rightarrow R^M$
  - Index each dimension of H(X) in a clustered B-tree
- Query processing
  - Collect candidates on each of the M projected dimensions
  - When T candidates are seen on all M lists, rerank them and return top-k

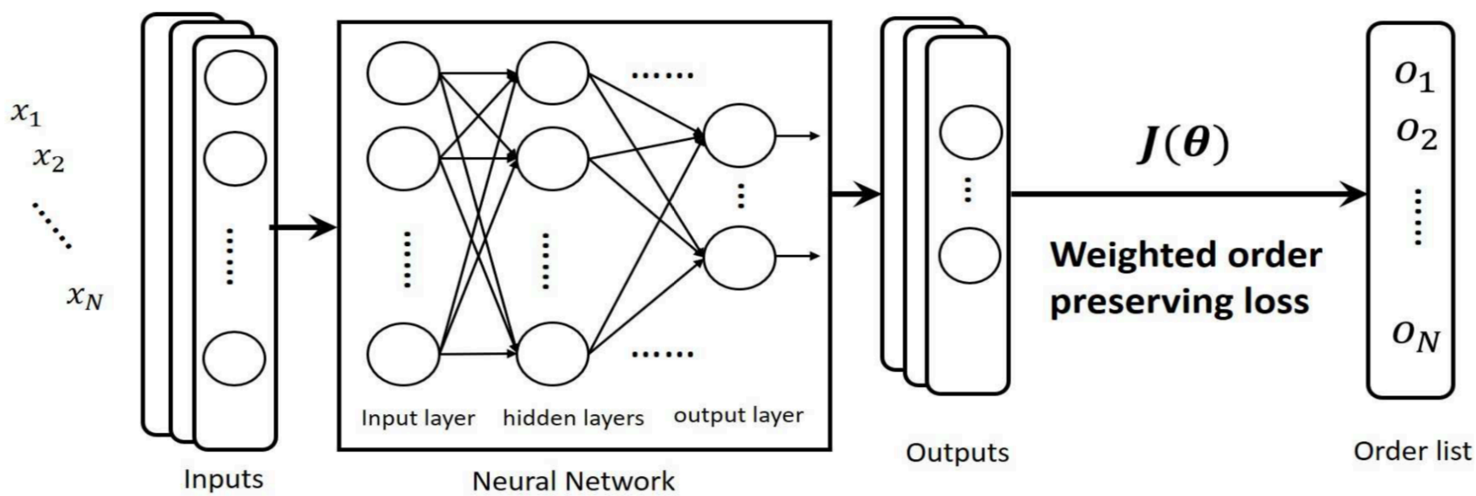
≈ MedRank (minfreq = 1) [FKS03]

must c.f., [AFKPS08]



Function family

- Consider
  - linear functions
    - $H(x)[m] = w_m^T x$
  - non-linear functions



w

How to learn the parameters?

Consider the linear functions:
$H(x)[m] = w_m^T x$

- Goal:
  - Encourage segment-order preserving mappings

Part of the Loss function

$$J^*(\mathbf{w}_m) = \sum_{i=1}^{L} \sum_{\tilde{x} \in l_i^o} \mathbf{1}_{r(\tilde{x}) \in [t\cdot(i-1), t\cdot i)}$$

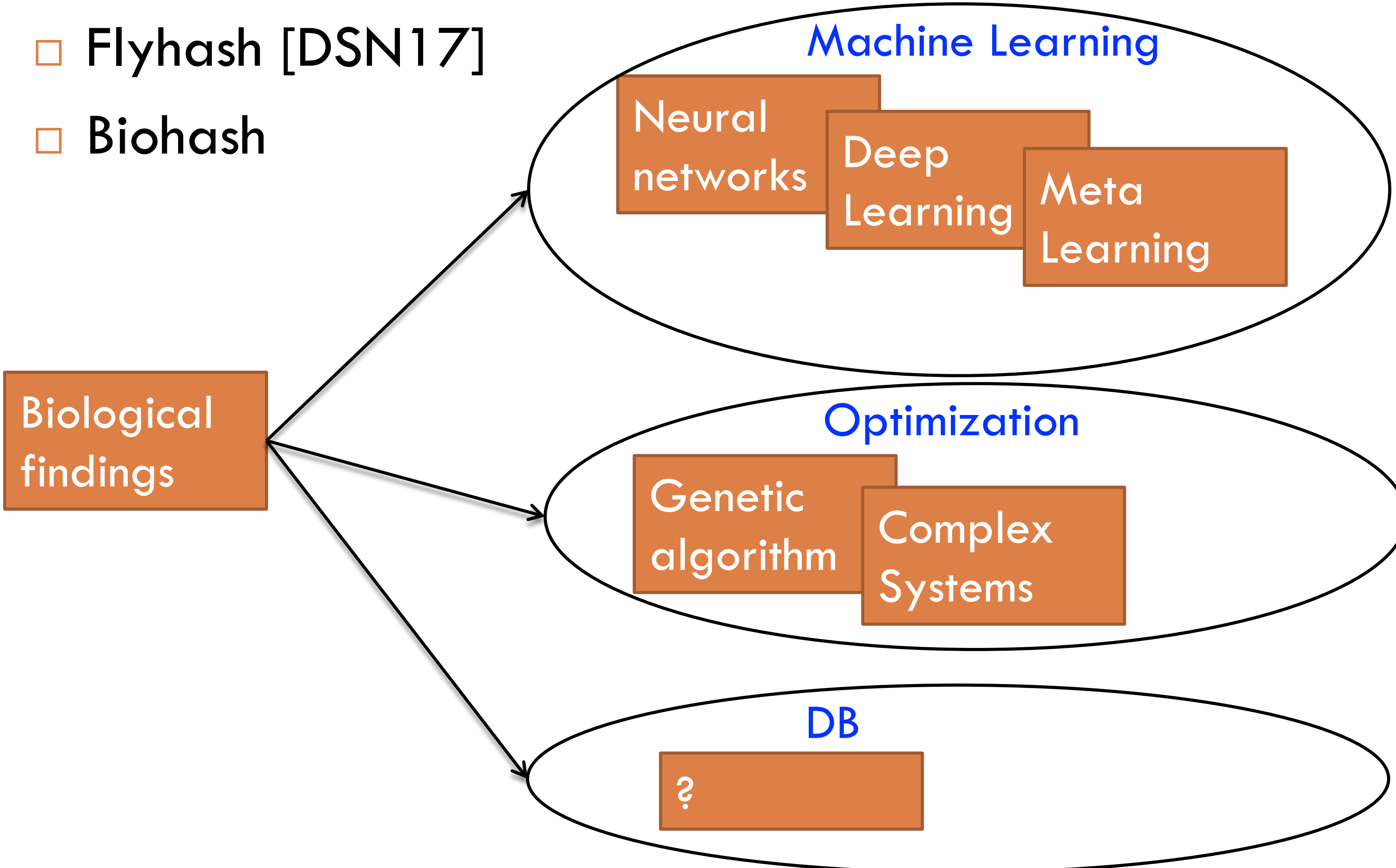
Continuous Relaxation

mapped x in the i-th segment

x in the i-th segment

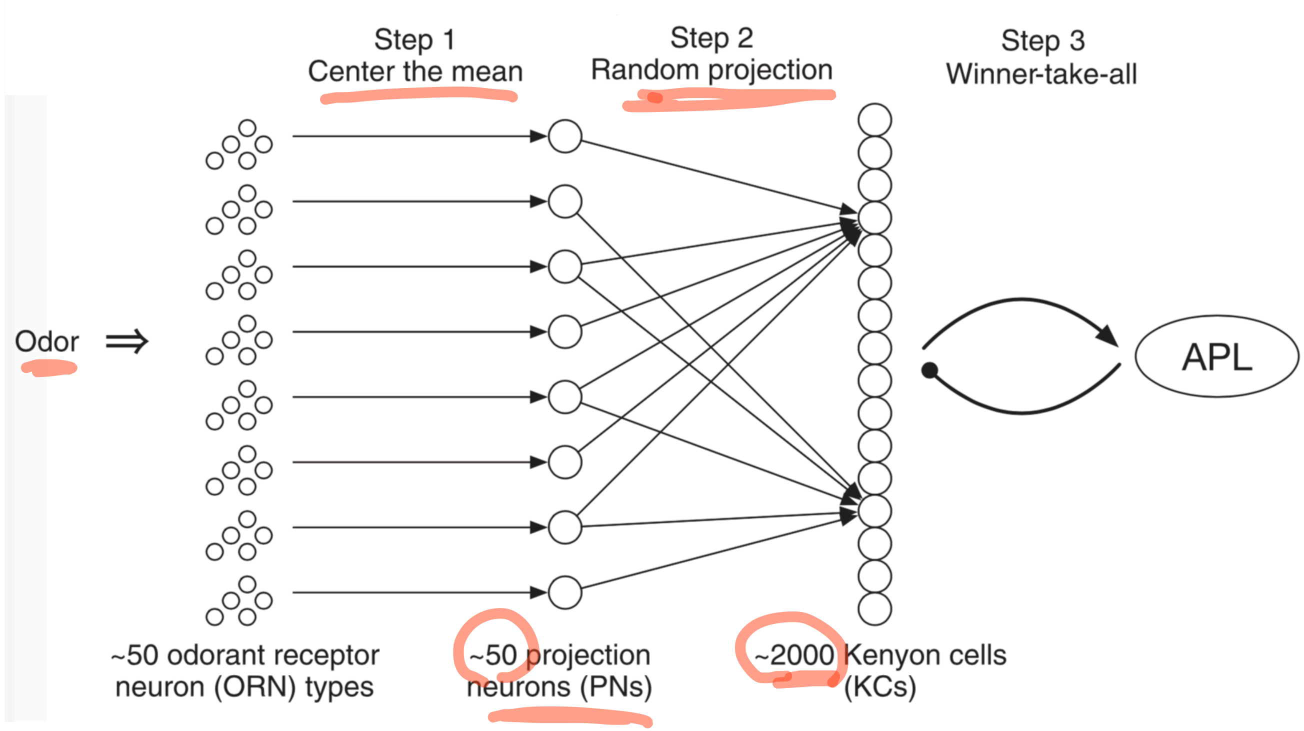
Biology Inspired Hashing

- Flyhash [DSN17]
- Biohash



FlyHash

- The fly olfactory circuit generates a low-overlapping, sparse neuron activation pattern when an odor is presented

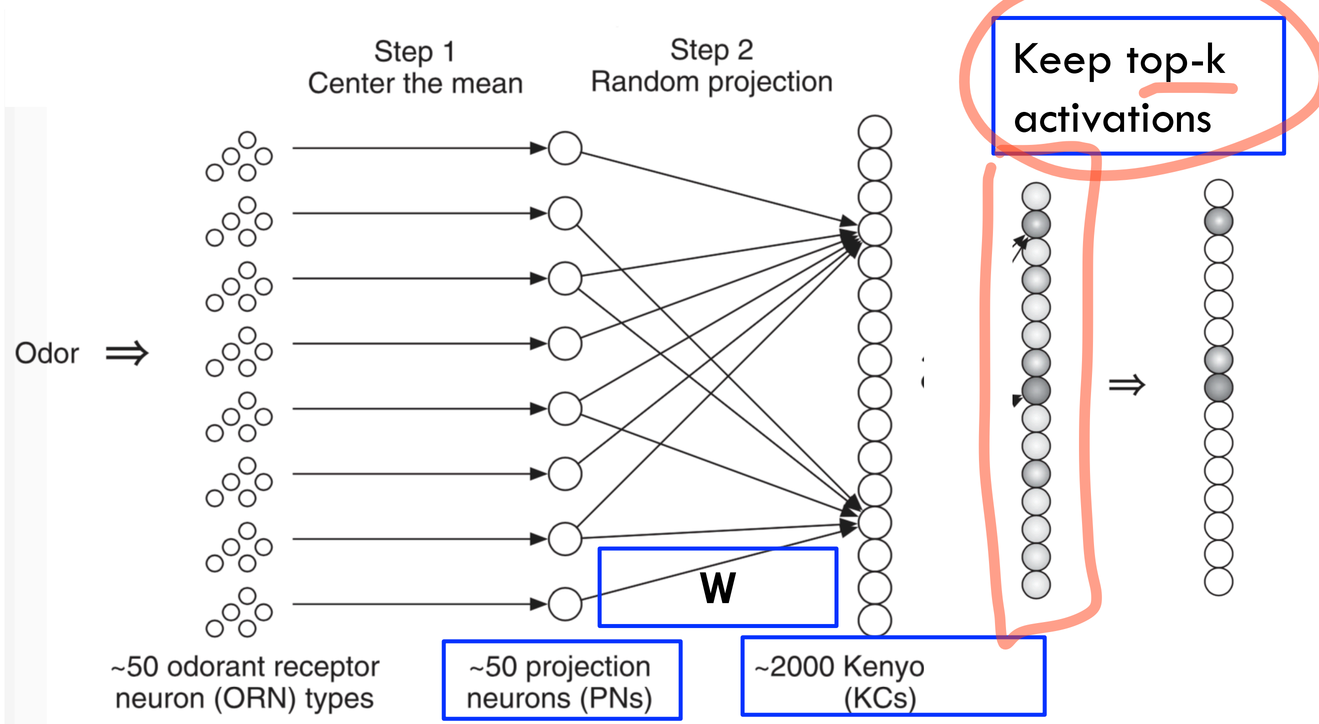


FlyHash

- Difference with LSH
  - W is a sparse binary random matrix
  - Dimensionality expansion !!
  - Sparsification
- L2 distance approximately preserved in expectation

Enable

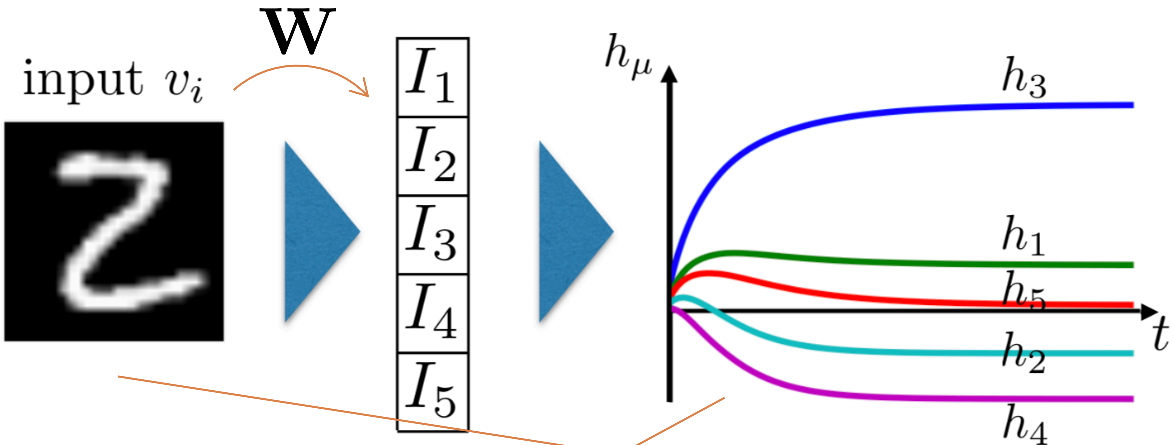
$$\mathbf{z} = \sigma(\mathbf{W}\mathbf{x})$$



[KH19]

- Unsupervised learning inspired by biological synaptic plasticity rules
- Overview
  - (Given W) Stabilizing the hidden competing neurons
  - Learning the projection matrix W

Learning h



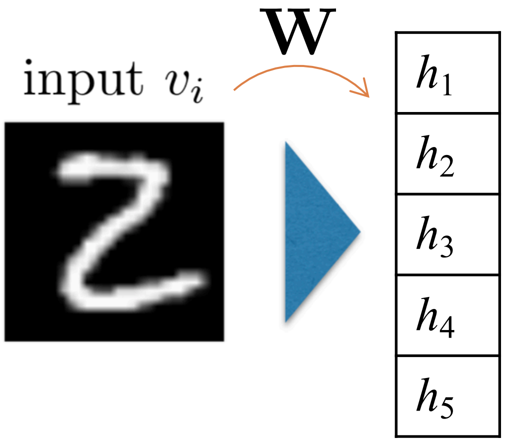
$$\tau \frac{\mathrm{d}}{\mathrm{d}\,t}\left(h_\mu\right) = I_\mu \boxed{-\gamma \sum_{\nu\neq\mu} \mathrm{RELU}\left(h_\nu\right)} - h_\mu, \quad \text{where} \quad I_\mu = \langle \mathbf{W}_\mu, \mathbf{v} \rangle$$

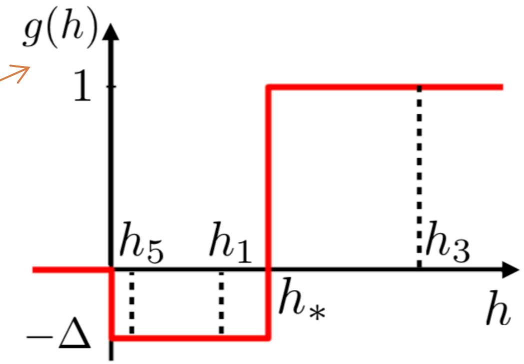
Competing for activation

- Fixing the W, the dynamical equation will converge to a stable hidden vector h

Learning W

h* is the activation threshold





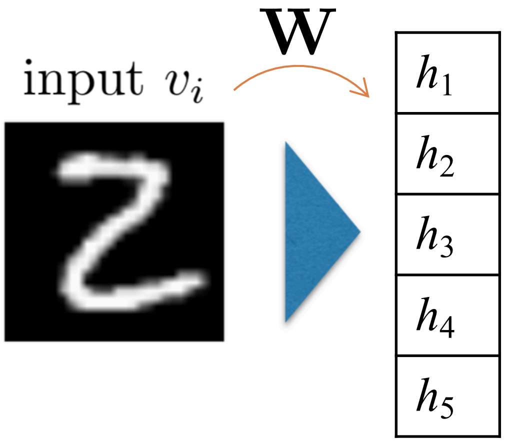
$$\tau_L \frac{\mathrm{d}}{\mathrm{d}\,t}(\mathbf{W}_i) = g(Q)\left(\mathbf{v}_i - \langle \mathbf{W}, \mathbf{v}\rangle \mathbf{W}_i\right), \qquad \text{where } Q = \frac{\langle \mathbf{W}, \mathbf{v}\rangle}{\langle \mathbf{W}, \mathbf{W}\rangle^{\frac{1}{2}}}$$

p = 2, for any one hidden neuron
W is its corresponding weight vector
R = 1

Force $|W|_p$ to converge to R^p

- Fixing the h, the dynamical equation will converge to a final weight matrix W

BioHash – Learning W



sort the projections and rank them

$$g[z] = \begin{cases} 1, & \text{rank}(z) = 1 \\ -\Delta, & \text{rank}(z) = r \\ 0, & \text{otherwise} \end{cases}$$

$$\tau_L \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{W}_i) = g(\boxed{\{ Q \}})\left(\mathbf{v}_i - \langle \mathbf{W}, \mathbf{v} \rangle \mathbf{W}_i\right), \qquad \text{where } Q = \frac{\langle \mathbf{W}, \mathbf{v} \rangle}{\langle \mathbf{W}, \mathbf{W} \rangle^{\frac{1}{2}}}$$

sorting among different normalized inner products (each produced by a different $\mathbf{W}_i$)

- The rest is the same as FlyHash (i.e., WTA sparsification)