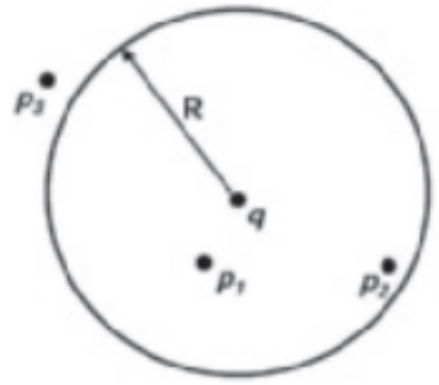
LSH: A Survey of Hashing for Similarity Search

CS 584: Big Data Analytics

LSH Problem Definition

- Randomized c-approximate R-near neighbor or (c,r)-NN: Given a set P of points in a d-dimensional space, and parameters R > 0, δ > 0, construct a data structure such that given any query point q, if there exists an R-near neighbor of q in P, reports some cR neighbor of q in P with probability 1- δ



- Randomized R-near neighbor reporting: Given a set P pf points in a d-dimensional space, and parameters R > 0, δ > 0, construct a data structure such that given any query point q, reports each R-near neighbor of q with a probability 1- δ

LSH Definition

- Suppose we have a metric space S of points with a distance measure d
- An LSH family of hash functions, $\mathcal{H}(r, cr, P_1, P_2)$, has the following properties for any $q, p \in \mathcal{S}$
  - If $d(p, q) \leq r$, then $P_{\mathcal{H}}[h(p) = h(q)] \geq P_1$
  - If $d(p, q) \geq cr$, then $P_{\mathcal{H}}[h(p) = h(q)] \leq P_2$
- For family to be useful, $P_1 > P_2$
- Theory leaves unknown what happens to pairs at distances between r and cr

LSH Gap Amplification

- Choose L functions g_j, j = 1, .., L
  - $g_j(q) = (h_{1,j}(q), \cdots, h_{k,j}(q))$
  - $h_{k,j}$ are chosen at random from LSH family $\mathcal{H}$

- Retain only the nonempty buckets (since total number of buckets may be large) - O(nL) memory cells

- Construct L hash tables, where for each j = 1, .. L, the nth hash table contains the datapoint hashed using the function g_j

LSH Query

- Scan through the L buckets after processing q and retrieve the points stored in them

- Two scanning strategies
  - Interrupt the search after finding the first L’ points
  - Continue the search until all points from all buckets are retrieved

- Both strategies yields different behaviors of the algorithm

LSH Query Strategy 1

Set L' = 3L to yield a solution to the randomized c-approximate R-near neighbor problem

- Let $\rho = \frac{\ln 1/P_1}{\ln 1/P_2}$
- Set L to $\theta(n^\rho)$
- Algorithm runs in time proportional to n^ρ
- Sublinear in n if $P_1 > P_2$

LSH Query Strategy 2

- Solves the randomized R-near neighbor reporting problem
    - Value of failure probability depends on choice of k and L
- Query time is also dependent on k and L and can be as high as $\theta(n)$

Hamming Distance [Indyk & Motwani, 1998]

- Binary vectors: $\{0, 1\}^d$
- LSH family: $h_i(p) = p_i$, where i is a randomly chosen index
- Probability of same bucket:

$$P(h(\mathbf{y}_i) = h(\mathbf{y}_j)) = 1 - \frac{||\mathbf{y}_i - \mathbf{y}_j||_H}{d}$$

- Exponent is $\rho = 1/c$

Jaccard Coefficient: Min-Hash

- Similarity between two sets C_1, C_2

$$\text{sim}(C_1, C_2) = ||C_1 \cap C_2||/||C_1 \cup C_2||$$

- Distance: 1 - sim(C_1, C_2)

- LSH family: pick a random permutation

$$h_\pi(C) = \min_\pi \pi(C)$$

- Probability of same bucket:

$$P[h_\pi(C_1) = h_\pi(C_2)] = \text{sim}(C_1, C_2)$$

Jaccard Coefficient: Other Options

- K-min sketch: generalization of min-wise sketch used for min-hash with smaller variance but cannot be used for ANN using hash tables like min-hash

- Min-max hash: instead of keeping the smallest hash value of each random permutation, keeps both the smallest and largest values of each random permutation and has smaller variance than min-hash

- B-bit minwise hashing: only uses lowest b-bits of the min-hash value and has substantial advantages in terms of storage space

Angle-based Distance: Random Projection

- Consider angle between two vectors:

$$\arccos\left(\frac{p \cdot q}{||p||_2||q||_2}\right)$$

- LSH family: pick a random vector w, which follows the standard Gaussian distribution

$$h_w(p) = \text{sign}(w \cdot p)$$

- Probability of collision

$$P(h(p) = h(q)) = 1 - \frac{\theta(p, q)}{\pi}$$

Angle-Based Distance: Other Families

- Super-bit LSH: divide random projections into G groups and orthogonalized B random projections for each group to yield GB random projections and G B-super bits

- Kernel LSH: build LSH functions with angle defined in kernel space $$\theta(p,q) = \arccos \frac{\phi(p)^\top \phi(q)}{||\phi(p)||_2 ||\phi(q)||_2}$$

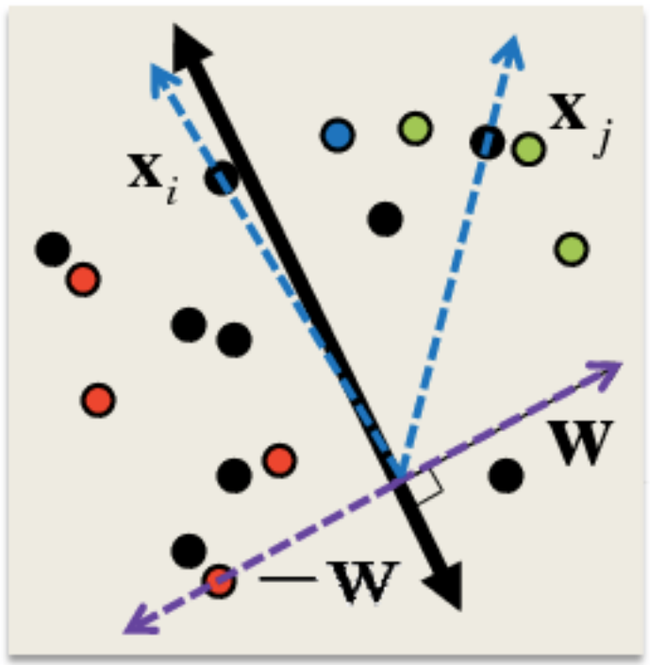
- LSH with learnt metric: first learn Mahalanobis metric from semi-supervised information before forming hash function

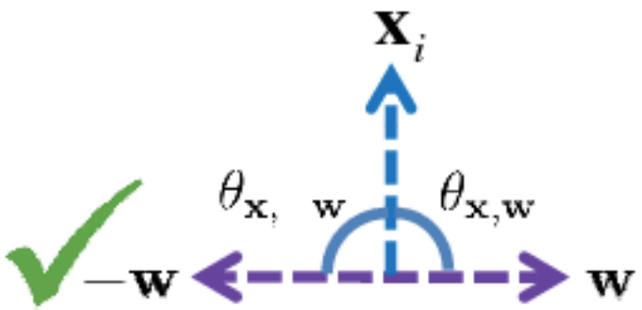
$$\theta(p,q) = \arccos \frac{p^\top A q}{||Gp||_2 ||Gq||_2}, \; G^\top G = A$$

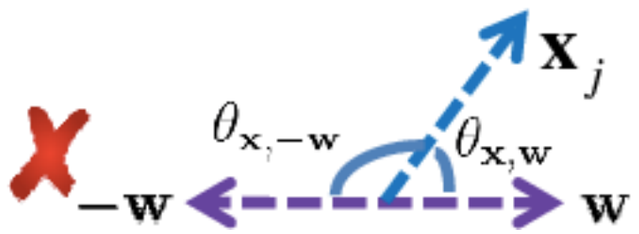
Angle-Based Distance: Other Families (2)

- Concomitant LSH: uses concomitant (induced order statistics) rank order statistics to form the hash functions for cosine similarity

- Hyperplane hashing: retrieve points closest to a query hyperplane







http://vision.cs.utexas.edu/projects/activehash/

ℓ_p Distance: Norms

- Norms usually computed over vector differences
- Common examples:
  - Manhattan (p = 1) on telephone vectors capture symmetric set difference between two customers
  - Euclidean (p = 2)
  - Small values of p (p = 0.005) capture Hamming norms, distinct values

ℓ_p Distance: p-stable Distributions

- Let v in R^d and suppose Z, X_1, …, X_d are drawn iid from a distribution D. Then D is p-stable if:

$$< v, X >= ||v||_p Z$$

- Known that p-stable distributions exist for $p \in (0, 2]$

- Examples:
  - Cauchy distribution is 1-stable
  - The standard Gaussian distribution is 2-stable
- For 0 < p < 2, there is a way to sample from a p-stable distribution given two uniform random variables over [0, 1]

http://dimacs.rutgers.edu/Workshops/StreamingII/datar-slides

ℓ_p Distance: p-stable Distributions (2)

- Consider a vector, where each Xi is drawn from a p-stable distribution

- For any pair of vectors, a, b:
  aX - bX = (a - b) X (by linearity)

- Thus aX - bX is distributed as (l_p(a-b))X' where X' is a p-stable distribution random variable

- Using multiple independent X's we can use a X - b X to estimate l_p(a - b)

http://dimacs.rutgers.edu/Workshops/StreamingII/datar-slides

ℓ_p Distance: p-stable Distributions (3)

- For a vector a, the dot product a X projects onto the real line
- For any pair of vectors a, b, these projections are "close" (with respect to p) if l_p(a-b) is "small" and "far" otherwise
- Divide the real line into segments of width w
- Each segment defines a hash bucket: vectors that project to the same segment belong to the same bucket

http://dimacs.rutgers.edu/Workshops/StreamingII/datar-slides

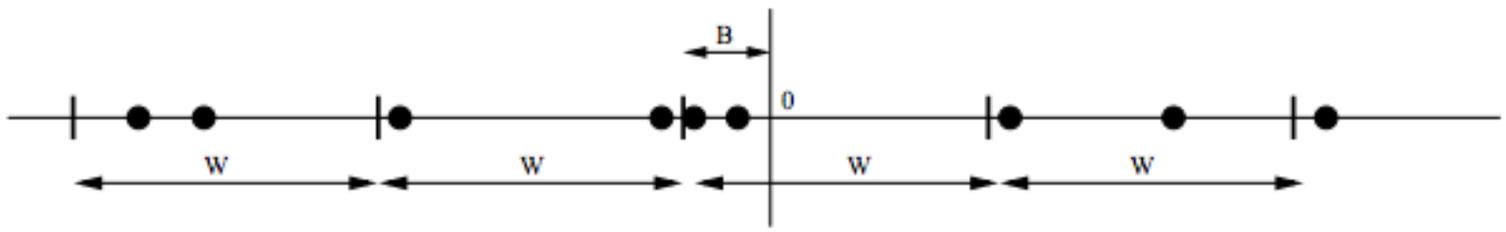
ℓ_p Distance: Hashing family

- Hash function:

$$h_{a,b}(v) = \left\lfloor \frac{a \cdot v + b}{w} \right\rfloor$$

- a is a d dimensional random vector where each entry is drawn from p-stable distribution

- b is a random real number chosen uniformly from [0, w] (random shift)



http://dimacs.rutgers.edu/Workshops/StreamingII/datar-slides

ℓ_p Distance: Collision probabilities

- pdf of the absolute value of p-stable distribution: $f_p(t)$
- Simplify notation: c = ||x - q||$_p$
- Probability of collision:

$$P(c) = \int_{t=0}^{w} \frac{1}{c} f(\frac{t}{c})(1 - \frac{t}{w})dt$$

- Probability only depends on the distance c and is monotonically decreasing

http://dimacs.rutgers.edu/Workshops/StreamingII/datar-slides

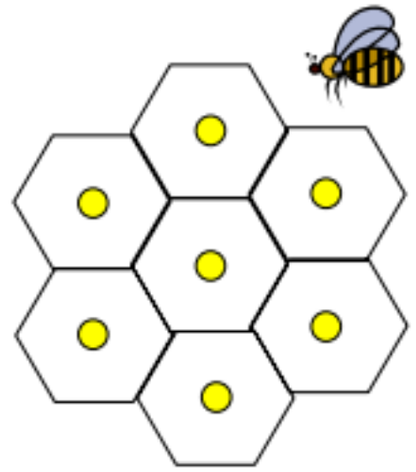
ℓ_p Distance: Comparison

- Previous hashing scheme for p = 1, 2
  - Reduction to hamming distance
  - Achieved $\rho = 1/c$
- New scheme achieves smaller exponent for p = 2
  - Large constants and log factors in query time besides n^{ρ}
  - Achieves the same for p = 1

http://dimacs.rutgers.edu/Workshops/StreamingII/datar-slides

ℓ_p Distance: Other Families

- Leech lattice LSH: multi-dimensional version of the previous hash family
  - Very fast decoder (about 519 operations)
  - Fairly good performance for exponent with c = 2 as the value is less than 0.37



- Spherical LSH: designed for points that are on unit hypersphere in Euclidean space

χ^2 Distance (Used in Computer Vision)

- Distance over two vectors p, q:

$$\chi^2(p,q) = \sqrt{\sum_{i=1}^{d} \frac{(p_i - q_i)^2}{p_i - q_i}}$$

- Hash family:

$$h_{w,b}(p) = \lfloor g_r(w^\top x) + b \rfloor,\ g_r(p) = \frac{1}{2}(\sqrt{\frac{8p}{r^2} + 1} - 1)$$

- Probability of collision:

$$P(h_{w,b}(p) = h_{w,b}(q)) = \int_0^{(n+1)r^2} \frac{1}{c} f(\frac{t}{c})(1 - \frac{t}{(n+1)r^2})dt$$

pdf of the absolute value of the 2-stable distribution

Learning to Hash

Task of learning a compound hash function to map an input item x to a compact code y

- Hash function
- Similarity measure in the coding space
- Optimization criterion

Learning to Hash: Common Functions

- Linear hash function

$$y = \text{sign}(w^\top x)$$

- Nearest vector assignment computed by some algorithm, e.g., K-means

$$y = \text{argmin}_{k \in \{1, \cdots, K\}} ||x - c_k||_2$$

- Family of hash functions influences efficient of computing hash codes and the flexibility of partitioning the space

Learning to Hash: Similarity Measure

- Hamming distance and its variances
  - Weighted Hamming distnace
  - Distance table lookup
  - …
- Euclidean distance
  - Asymmetric Euclidean didstance

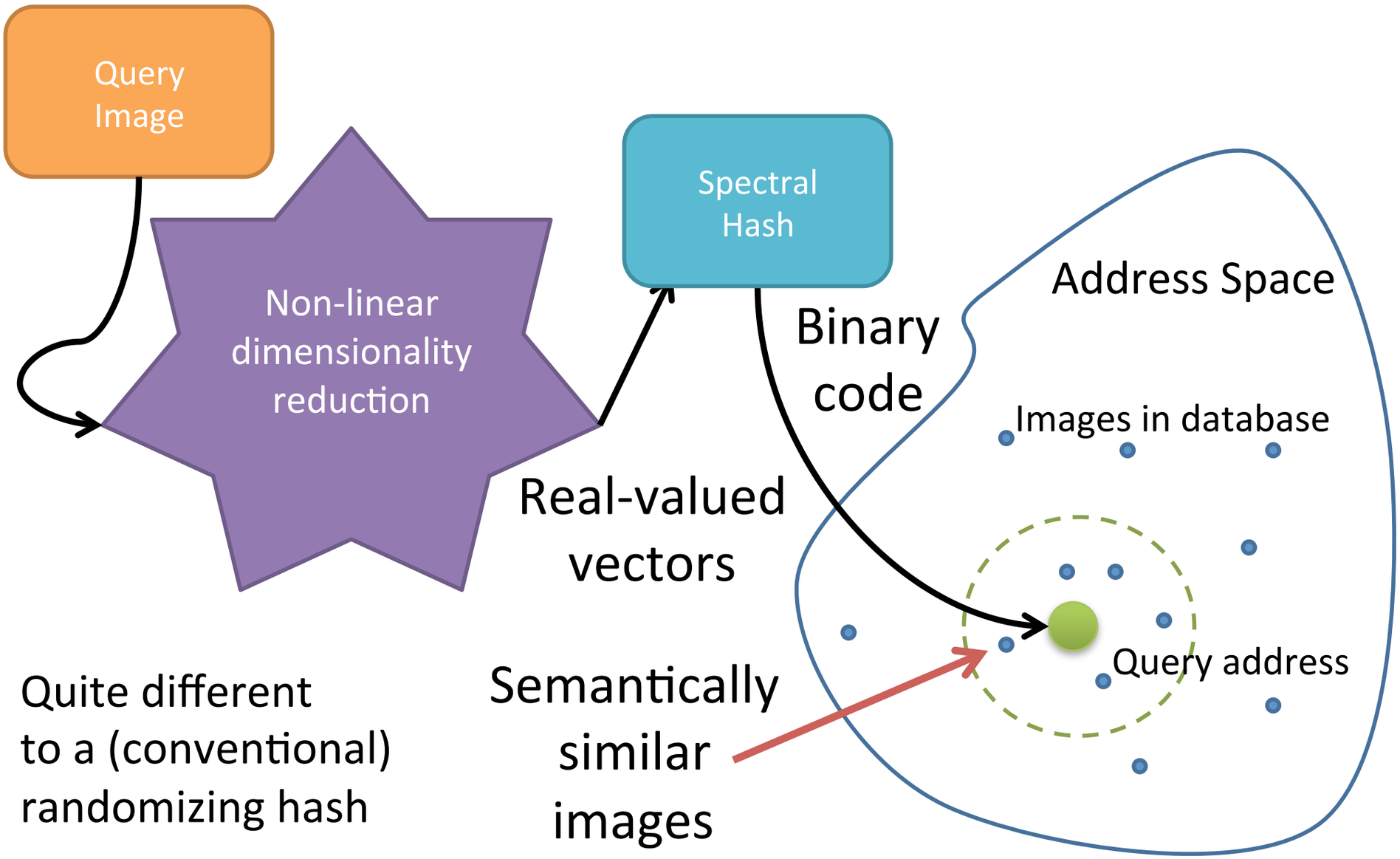
Learning to Hash: Optimization Criterion

- Similarity preserving
  - Similarity alignment criterion directly compares the order of ANN search result to true result (order-perserving criterion)
  - Coding consistent hashing encourages the smaller distances in the coding space but with smaller distances in the input space
- Coding balance uniformly distributes the codes amongst each bucket
  - Bit balance, bit independence, search efficiency, etc.

Coding Consistent Hashing: Spectral Hashing

- Pioneering coding consistent hashing algorithms
  - Similar items are mapped to similar hash codes based on the Hamming distance
- Small number of hash bits are required
  - Bit balance and bit correlation

Spectral Hashing



http://cs.nyu.edu/~fergus/drafts/Spectral%20Hashing.ppt

Spectral Hashing: Algorithm

- Use PCA of the N dimensional reference data items to find principal components

- Compute the M 1D Laplacian eigenfunctions with the smallest eigenvalues along each PCA direction

- Pick the M eigenfunctions with the smallest eigenvalues among Md eigenfunctions

- Threshold the eigenfunction at zero, obtaining the binary codes

Coding Consistent Hashing: Other Functions

- Kernelized spectral hashing: extension of spectral hashing to allow hash functions to be defined using kernels

- Hypergraph spectral hashing: extension of spectral hashing from ordinary (pair-wise) graph to a hypergraph (multi-wise graph)

- ICA hashing: achieves coding balance (average number of data items mapped to each hash code is the same) by minimizing mutual information

Similarity Alignment Hashing: Binary Reconstructive Embedding

- Learn hash codes to minimize Euclidean distance in the input space and the Hamming distance in the hash code values

$$\min \sum_{(i,j)\in N} \left(\frac{1}{2}||x_i - x_j||_F^2 - \frac{1}{m}||y_i - y_j||_2^2 \right)^2$$

- Sample data items to form the hashing function using a kernel function and learn the weights

Order Preserving Hashing: Minimal Loss Hashing

- Hinge-like loss function to assign penalties for similar points when they are too far apart

$$\min \sum_{(i,j) \in L} I[s_{ij} = 1] \max(||y_i - y_j||_1 - \rho + 1, 0) +$$
$$I[s_{ij} = 0] \lambda \max(\rho - ||y_i - y_j||_1 + 1, 0)$$

- Optimize using a perceptron-like learning procedure

Learning to Hash: Other Topics

- Many other hash learning algorithms (different objectives associated with different domains)

- Moving beyond Hamming distances in the coding space (e.g., Manhattan, asymmetric distances)

- Quantization (how to partition the projection values of the reference data items along the direction into multiple parts)

- Active and online hashing (using small sets of pairs with labeled information)

- Fast search in Hamming space

Future Hashing Trends

- Scalable hash function learning: existing algorithms are too slow and even infeasible when handling large data

- Hash code computation speedup: improving the cost of encoding a data item

- Distance table computation speedup: product quantization and its variants need to precompute distance table between query and elements of dictionary

- Multiple and cross modality hashing: dealing with the variant of data types and data sources