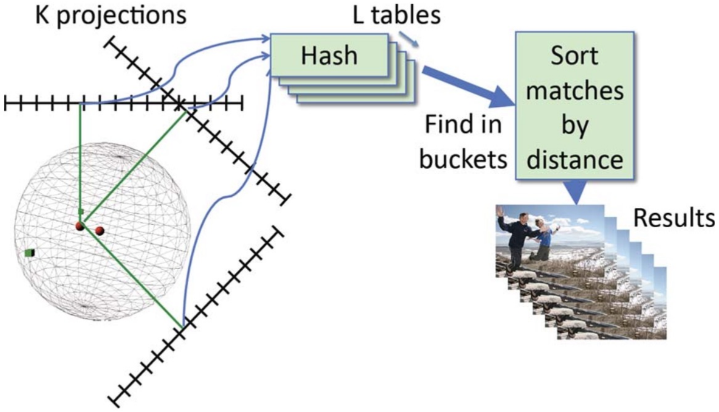
CS60021: Scalable Data Mining

Similarity Search and Hashing

Sourangshu Bhattacharya

LEARNING TO HASH

Locality Sensitive Hashing



Picture courtesy Slaney et al.

Given input data, radius r, approx factor c and confident δ

Output: if there is any point at distance $\leq r$ then w.p. $1-\delta$ return one at distance $\leq cr$

Algo: Choose (k, L).

do L times

iid hash functions : $\{\mathrm{h}_{\mathrm{i}1} \dots . \mathrm{h}_{\mathrm{ik}}\}$

Create hash table H_i by putting each x in bucket $H_i(x) = \left(h_{i1}(x), \dots h_{ik}(x)\right)$

Store non-empty buckets in normal hash table

Issues

- Parameters k, L need to be tuned for each domain
- Random directions are meant to create a random partitioning of the dataset
- While useful to guard against "worst case datasets", we do not exploit the dataset structure

Hashing as binary codes

- Assume points are in Euclidean space

- How can we get binary vectors so that Hamming distance approximates Euclidean distance

Properties of a binary code

- Should be easily computable
- Should preserve distances approximately
- Should have small number of bits
  - the bits should be independent and unbiased

Optimization

- W_{ij} = similarity between i and j
  - Say $W_{ij} = \exp\left(-\frac{|x_i - x_j|^2}{s}\right)$

- y_i = codeword for point i
- $|y_i - y_j|^2$ also equals Hamming(i, j)

Learning codes

- Average hamming distance = $\sum_{ij} W_{ij} \left| y_i - y_j \right|^2$

- $y_i \in \{-1, +1\}^k$
- Each bit should be unbiased: $\sum_i y_i = 0$

- Bits should be uncorrelated $\sum_i y_i y_i^t = I$

Casting as optimization problem

[Waiss et al.]

- Can we solve : minimize $\sum_{ij} W_{ij} \left| y_i - y_j \right|^2$
- subject to
  - $y_i \in \{-1, +1\}^k$
  - $\sum_i y_i = 0$
  - $\sum_i y_i y_i^t = I$

Hardness

- Unfortunately, no!, even for single bit

- Graph partitioning problem: For graph G partition V(G) into two sets A and B such that $|A| = |B|$ and

$$minimize \sum_{i \in A, j \in B} W_{ij}$$

Spectral Relaxation

- $Y = n \times k$ code matrix
- Diagonal $D,\ D_{ii} = \sum_j W_{ij}$
- minimize $\sum_{ij} W_{ij} \left| y_i - y_j \right|^2$ = $trace(Y^t (D\ - W) Y)$
  - $Y^t \cdot 1 = 0$
  - $Y^t\, Y = I$
  - Drop the constraint that Y are in $\{-1, +1\}$

Spectral codes

- The above problem is solved by $Y =$ smallest – k eigenvectors of $D - W$
  - After dropping the one with value 0

- To get codes,
  - We could threshold eigenvectors, but then hard to extend it for query

Eigenvectors

- Assume that the data is coming from some distribution in R^d
  - But estimating this distribution is hard also
  - We could try to interpolate the eigenvectors to query points, under above assumptions, but is computationally expensive (Nystrom extension)

Eigenvectors

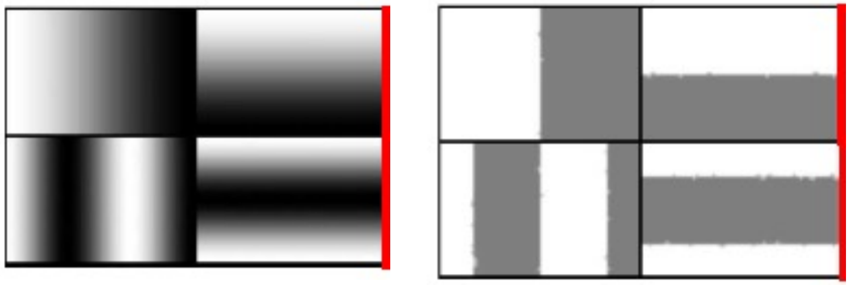
- Assume that the data is coming from some distribution in R^d
  - But estimating this distribution is hard also
  - We could try to interpolate the eigenvectors to query points, under above assumptions, but is computationally expensive (Nystrom extension)

- Assume data distribution is product of uniform distributions
  - Use PCA to find the axes

Eigenfunctions

- Take limit of eigenvectors as $n \to \infty$, and consider the "normalized" similarity matrix (Laplacian)
- Analytical form of Eigenfunctions exists for certain distributions (uniform, Gaussian)
- For uniform

$$\Phi_k(x) = \sin\left(\frac{\pi}{2} + \frac{k\pi}{b-a}x\right)$$
$$\lambda_k = 1 - e^{-\frac{\epsilon^2}{2}\left|\frac{k\pi}{b-a}\right|^2}$$

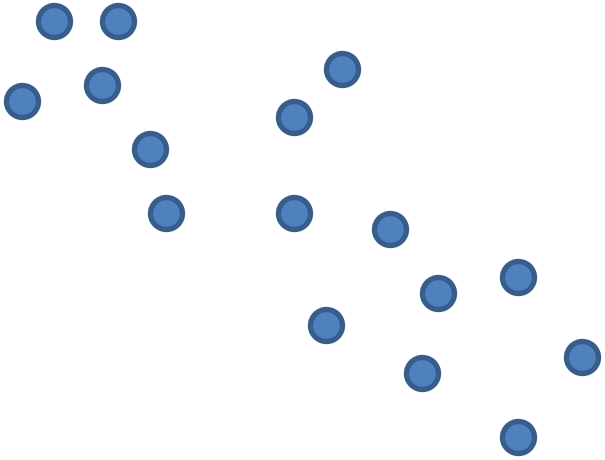


- Constant time calculation for any new point

[Image from Waiss et al]

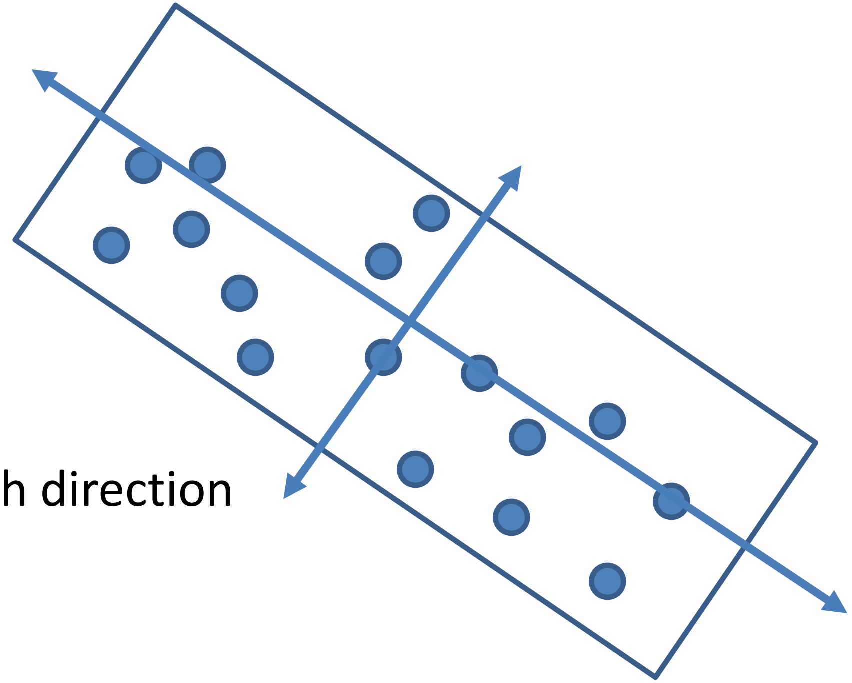
Algorithm

<u>Input:</u> Data $\{x_i\}$, target dimensionality k



Algorithm

Create top k PCA of $D - W$



Gives us top k axes

Find the $[a_i, b_i]$ for each axes

and create $\phi_1(x) \ldots \phi_k(x)$ for each direction

Algorithm

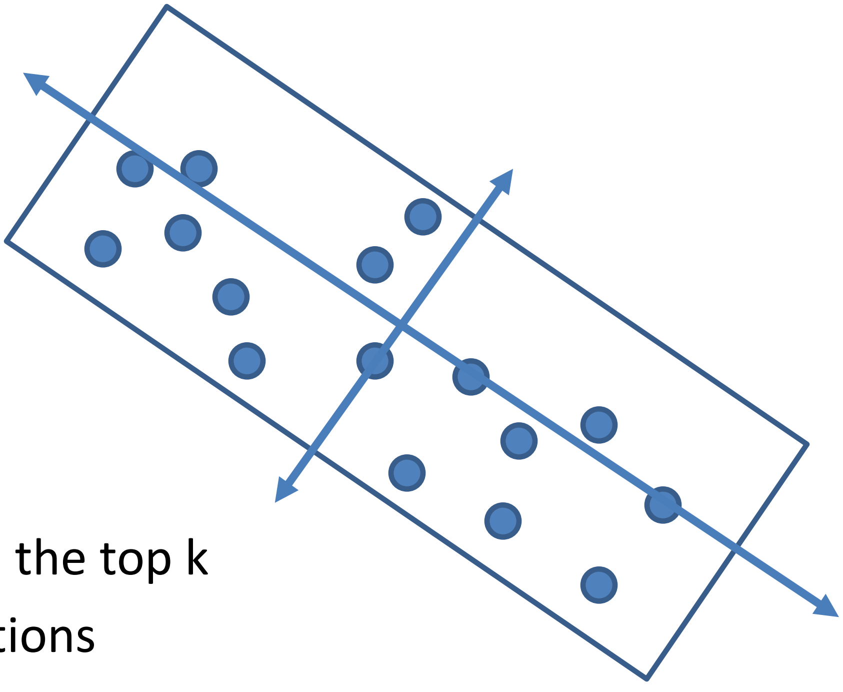
Create top k PCA of $D - W$

Gives us top k axes

Find the $[a_i, b_i]$ for each axes

and create $\phi_{i1}(x) \ldots \phi_{ik}(x)$

and $\lambda_{i1} \ldots \lambda_{ik}$ for each direction

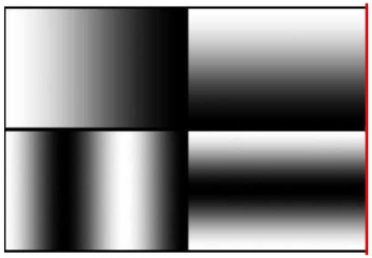


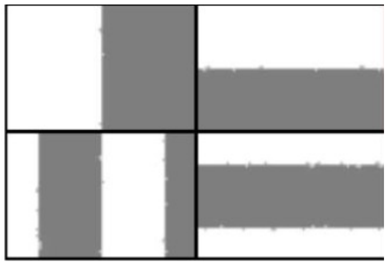
Total dk eigenvalues→ sort and take the top k eigenvalues and corresponding functions

Algorithm

Threshold chosen

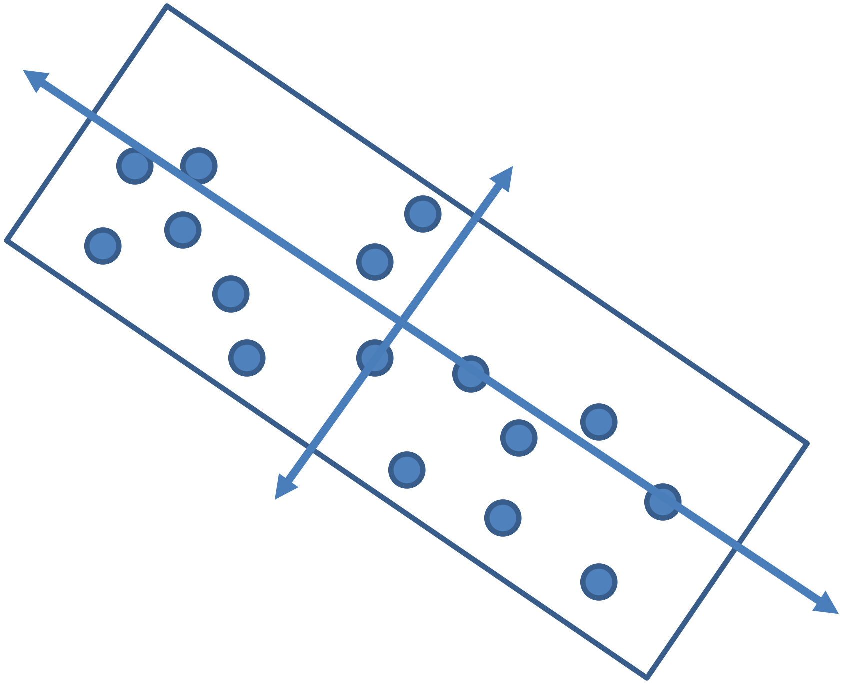
Eigenfunctions





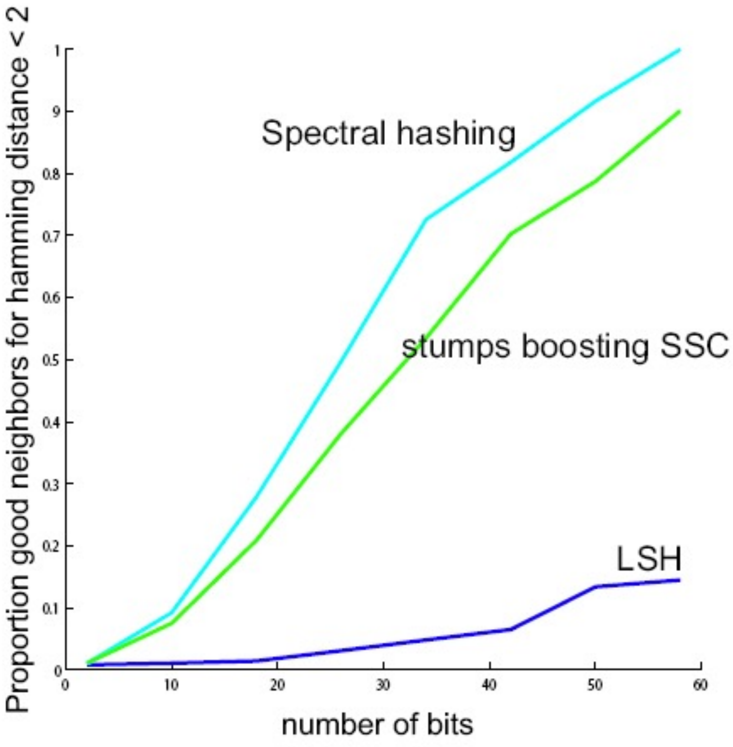
Empirical observation:
 bit codes

seem robust to the
uniform assumption



Results

- Shown to have better properties than naïve LSH on large datasets

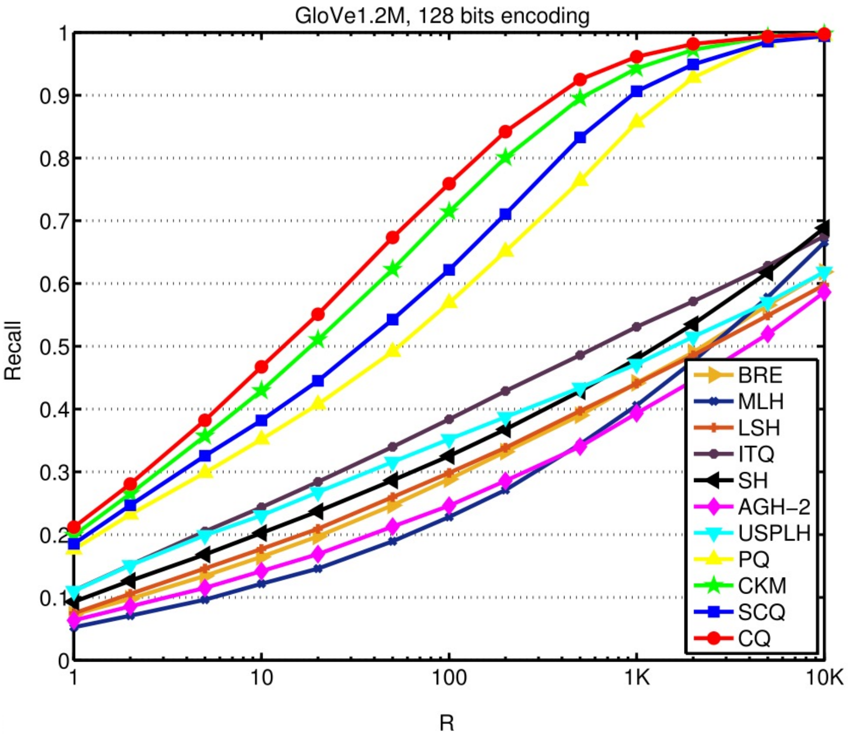


[Image from Waiss et al]

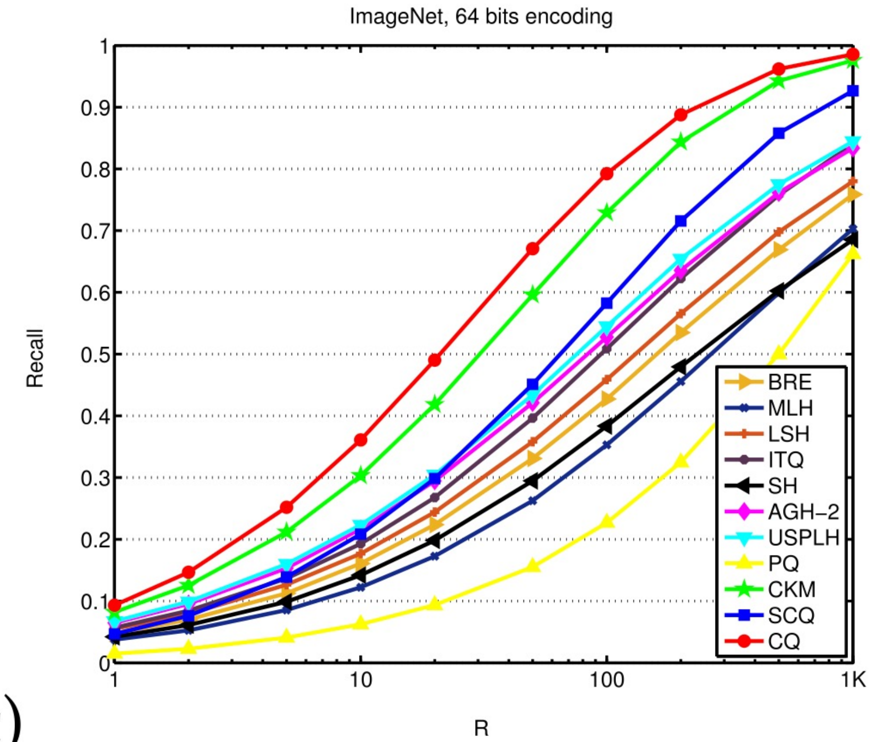
Conclusion

- Large literature on learning the hash codes rather than use random projection
  - Wang, Jingdong, Ting Zhang, Nicu Sebe, and Heng Tao Shen. "**A survey on learning to hash.**" IEEE TPAMI (2017): 769-790.
  - Jegou, Herve, Matthijs Douze, and Cordelia Schmid. "**Product quantization for nearest neighbor search**." IEEE TPAMI (2010): 117-128.

Conclusion



(c)



Conclusion

- Large literature on learning the hash codes rather than use random projection
  - Wang, Jingdong, Ting Zhang, Nicu Sebe, and Heng Tao Shen. "**A survey on learning to hash.**" IEEE TPAMI (2017): 769-790.
  - Jegou, Herve, Matthijs Douze, and Cordelia Schmid. "**Product quantization for nearest neighbor search**." IEEE TPAMI (2010): 117-128.

- Unfortunately, theoretical guarantees are not available for such data-dependent version
  - time to calculate projections might also be higher.

References:

- Primary references for this lecture
  - Spectral Hashing**,** Yair Weiss, Antonio Torralba and Rob Fergus. [*NIPS*], 2008