Tsinghua University

哈希表示学习

Hashing Representation Learning

吴志勇

清华大学深圳国际研究生院

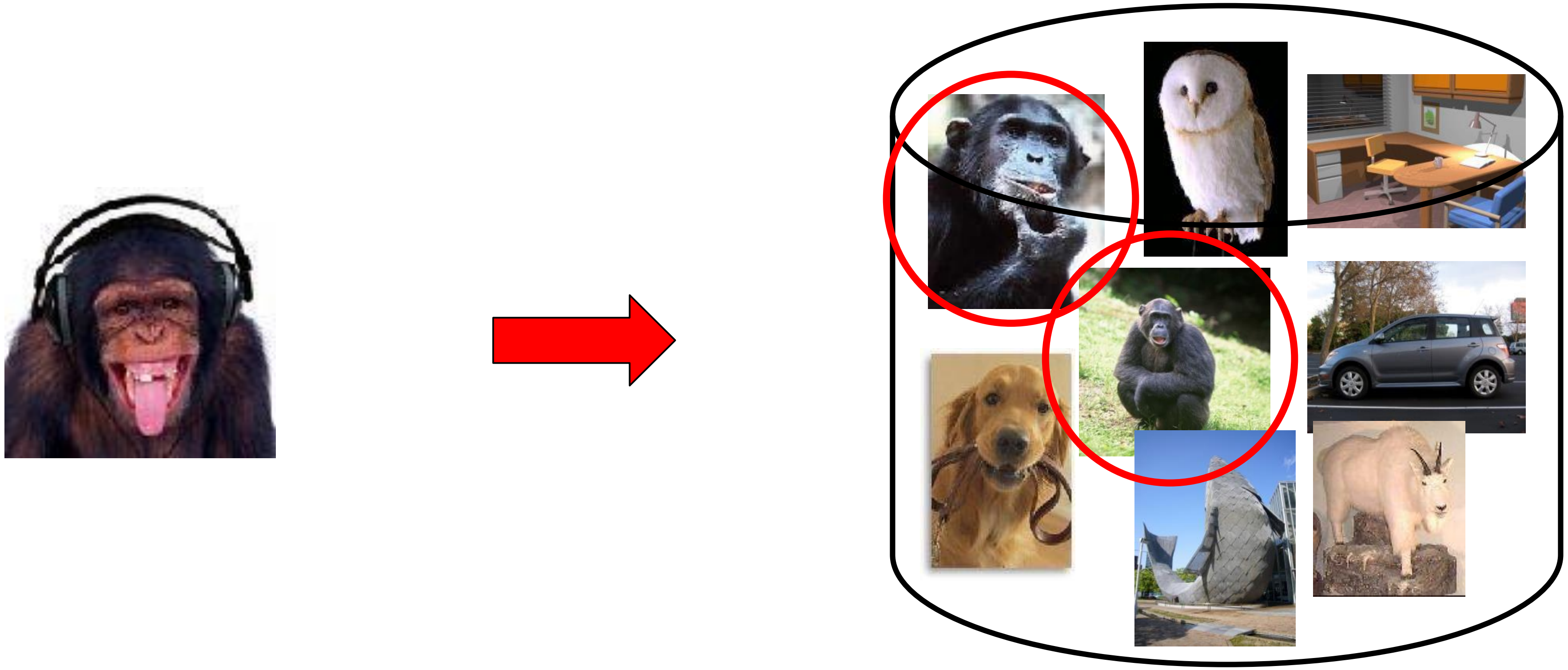
The Problem

- Large scale image search
  - We have a candidate image
  - Want to search a large database to find similar images
  - Search the internet to find similar images

- Fast
- Accurate

Large Scale Search in Database

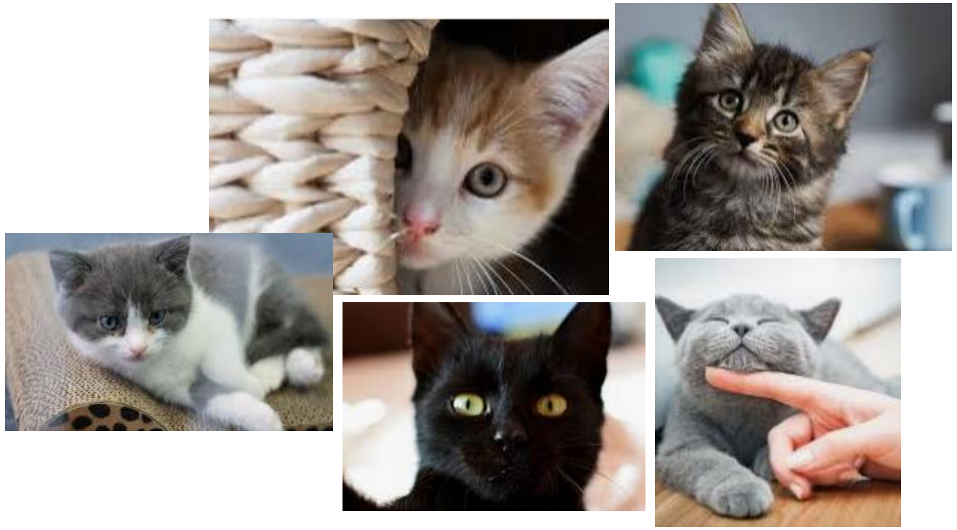
- Find similar images in a large database



Internet Large Scale Search

- Internet contains billions of images

Search the Internet



- The Challenge:
  - Need way of measuring similarity between images (distance metric learning)
  - Needs to scale to Internet (how?)

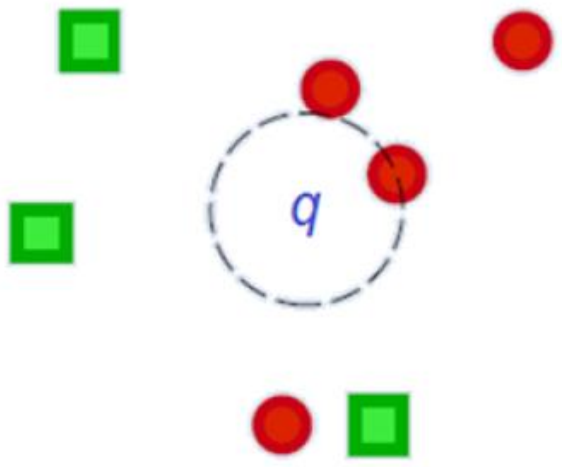
Large Scale Search

- Representation must fit in memory (disk too slow)
- (for 2010) Facebook has ~10 billion images (10^{10})
- PC has ~10 Gbytes of memory (10^{11} bits)

→ Budget of 10^{1} bits/image

Nearest Neighbor Search (Retrieval) for Big Data

- Given a query point q, return the points closest (similar) to q in the dataset (e.g., image database)



- Challenges in big data applications
  - Query speed
  - Storage cost
  - Curse of dimensionality

Requirements for Search

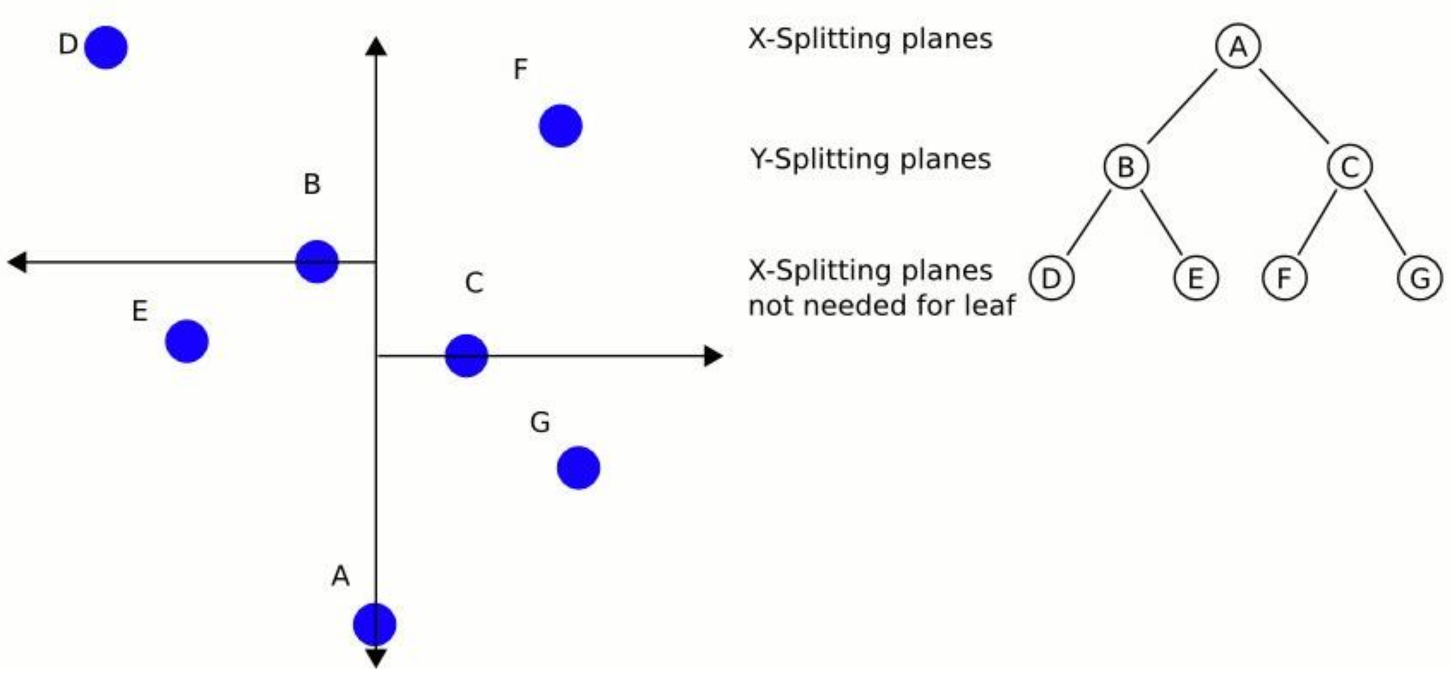
- Search must be both **fast**, **accurate** and **scalable to large data set**
- Fast
  - Kd-trees: tree data structure to improve search speed
  - Locality Sensitive Hashing: hash tables to improve search speed
  - Small code: small binary code (010101101)
- Scalable
  - Require very little memory, enabling their use on standard hardware or even on handheld devices
- Accurate
  - Learned distance metric

Existing Large Scale Search Algorithms: Categorization

- Tree Based Structure
  - Spatial partitions (i.e. **kd-tree**) and recursive hyper plane decomposition provide an efficient means to search low-dimensional vector data exactly
- Hashing
  - **Locality-sensitive hashing** offers sub-linear time search by hashing highly similar examples together
- Small binary Code
  - **Compact binary code**, with a few hundred bits per image

Tree Based Structure

- Kd-tree
  - The kd-tree is a binary tree in which every node is a **k-dimensional** point



Hashing

h(Statue of Liberty)=
10001010

h(Napoleon)=
01100001

h(Napoleon)=
01100101

flipped bit

Should be very different

Should be similar

Hashing

- By using hash-code to construct index, we can achieve constant or sub-linear search time complexity
- Two stages:
  - Projection stage
    - Projected with real-valued projection function
    - Given a point $\mathbf{x}$, each projected dimension $\boldsymbol{i}$ will be associated with a real-valued projection function $f_i(\mathbf{x})$, e.g. $f_i(\mathbf{x})=\mathbf{w}_i^T\mathbf{x}$
  - Quantization stage
    - Turn real into binary

Hashing: Data-Independent Methods

- The hash function family is defined independently of the training dataset
  - Locality Sensitive Hashing (LSH)

1. A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, 1999.
2. A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117-122, 2008.
3. M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *ACM SOCG*, 2004.
4. P. Jain, B. Kulis, and K. Grauman. Fast Image Search for Learned Metrics. In *CVPR*, 2008.
5. B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, 2009.

- Hash function: random projections or manually constructed

Hashing: Data-Dependent Methods (Learning to Hash)

- Hash functions are learned from a given training dataset
  - Compared with data-independent methods, data-dependent methods (also called learning to hash methods) can achieve comparable or even better accuracy with shorter binary codes

- Seminal papers

1. R. Salakhutdinov and G. Hinton. Semantic Hashing. In *SIGIR workshop on Information Retrieval and applications of Graphical Models*, 2007.
2. R. Salakhutdinov and G. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969-978, 2009.
3. A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.
4. Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.

Locality Sensitive Hashing

- Hashing methods to do fast Nearest Neighbor (NN) search
- *Sub-linear time search* by hashing highly similar examples together in a hash table
  - Take random projections of data
  - Quantize each projection with few bits
  - Strong theoretical guarantees

Small Binary Code

1110101010101010

- Binary?
  - 0101010010101010101
  - Only use binary code (0/1)
- Small?
  - A small number of bits to code each image
  - i.e. 32 bits, 256 bits
- How could this kind of small code improve the image search speed?

Details of These Algorithms

- Locality sensitive hashing (LSH)
  - Basic LSH
  - LSH for learned metric

- Small binary code
  - Basic small code idea
  - Spectral hashing

Locality Sensitive Hashing

局部敏感哈希

Locality Sensitive Hashing (LSH)

- The basic idea behind LSH is to project the data into a low-dimensional binary (Hamming) space; that is, each data point is mapped to a b-bit vector, called the *hash key*
- For a group of hash functions H, each hash function h ($h \in H$) must satisfy the *locality sensitive hashing* property:

$$P[h(x_i)=h(x_j)] = sim(x_i, x_j)$$

where $sim(x_i, x_j) \in [0,1]$ is the similarity function of interest

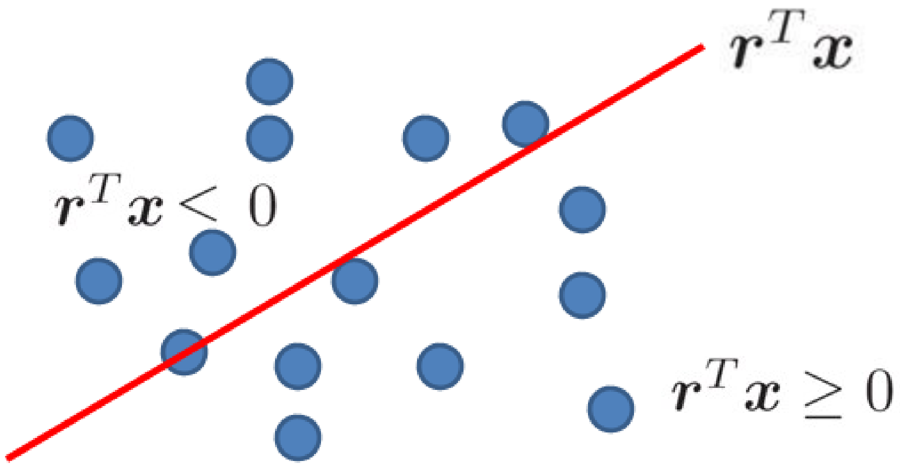
M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *ACM SOCG*, 2004.

LSH Functions for Dot Products

- The hashing function of LSH to produce Hash Code

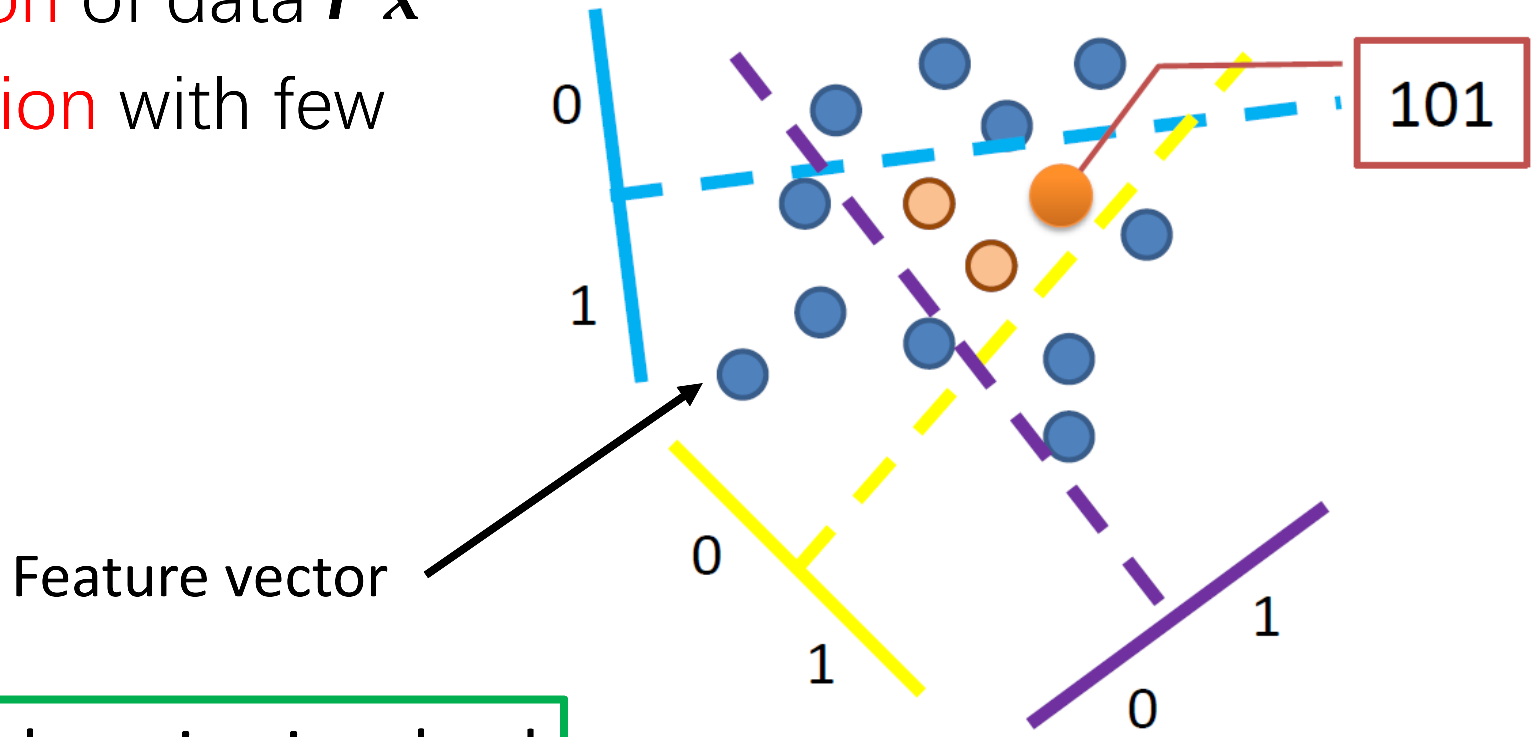
$$h_{\boldsymbol{r}}(\boldsymbol{x}) = \begin{cases} 1, & if\ \boldsymbol{r}^T\boldsymbol{x} \geq 0 \\ 0, & otherwise \end{cases}$$

$\boldsymbol{r}^T\boldsymbol{x} \geq 0$ is a hyper-plane separating the space



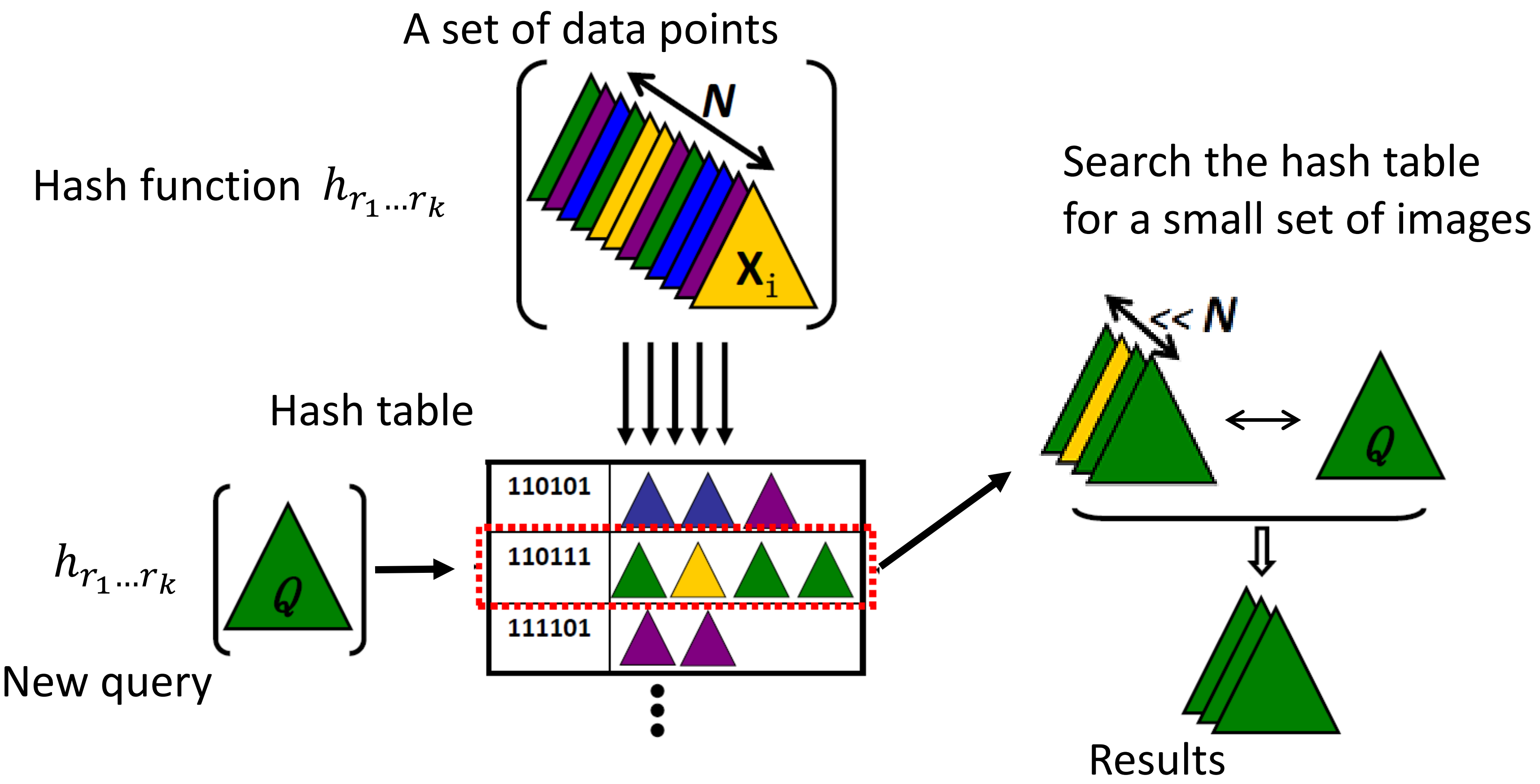
Locality Sensitive Hashing (LSH)

- Take random projection of data $\boldsymbol{r}^T\boldsymbol{x}$
- Quantize each projection with few bits



No learning involved

How to Search from Hash Table?



Could We Improve LSH?

- In LSH, each hash function h must satisfy the *locality sensitive hashing* property:

$$P[h(x_i)=h(x_j)] = sim(x_i, x_j)$$

where $sim(x_i, x_j) \in [0,1]$ is the similarity function of interest

Metric Learning, 度量学习

Could We Improve LSH?

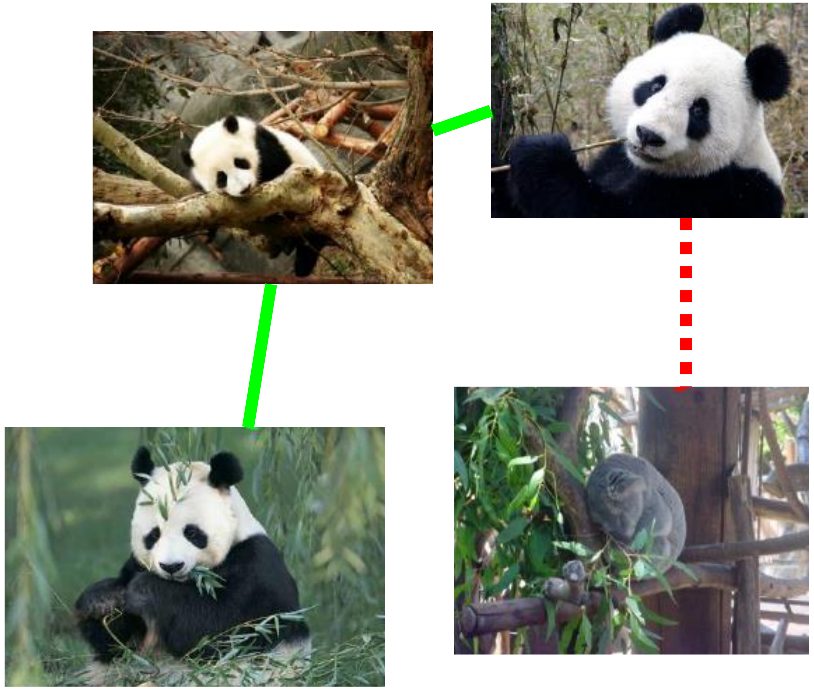
$$P[h(x_i)=h(x_j)] = sim(x_i, x_j)$$

- Could we utilize learned metric to improve LSH?
- How to improve LSH from learned metric?
- Assume we have already learned a distance metric A from domain knowledge
- X^TAX has better quantity than simple metrics such as Euclidean distance

Distance Metric Learning

- Distance Metric
  - "*Generic*" distances or low-dimensional representations are amenable to fast search, but may be inaccurate for a given problem
  - *Learned* task-specific distance functions are more accurate, but current methods cannot guarantee fast search for them
- Goal:
  - Develop approximate similarity search method for learned metrics
  - Encode side-information into randomized locality-sensitive hash functions
  - Applicable for a variety of image search tasks

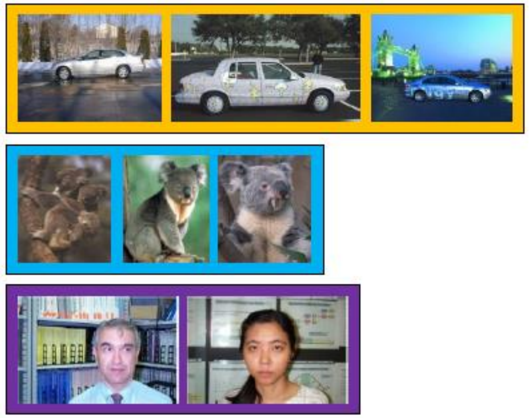
Metric Learning



- There are various ways to judge appearance / shape similarity …
- But often we know more about (some) data than just their appearance
- Exploit partially labeled data and/or (dis)similarity constraints to construct more useful distance function

Example Sources of Similarity Constraints

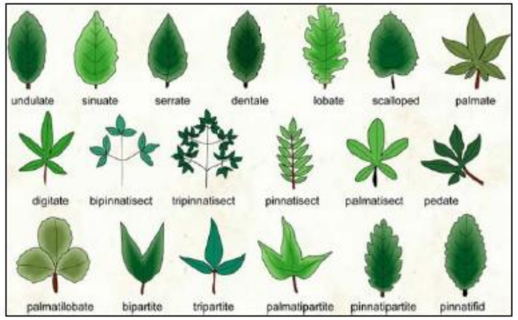
Partially labeled image databases



Fully labeled image databases

User feedback

Detected video shots, tracked objects



Problem-specific knowledge

Mahalanobis Distance

- Distance parameterized by probability distribution $d \times d$ matrix $\mathbf{A}$:

$$d_{\mathbf{A}}(x_i, x_j) = (x_i - x_j)^{\mathrm{T}} \mathbf{A} \, (x_i - x_j)$$

- Similarity measure is associated with generalized inner product (kernel)

$$s_{\mathbf{A}}(x_i, x_j) = x_i^{\mathrm{T}} \mathbf{A} \, x_j$$

- Then how to learn the distance metric?

Information-Theoretic Metric Learning (ITML)

- Formulation (Log-Det Divergence):

$$min_A D_{ld}(A, A_0)$$

$$\text{s.t.} \begin{array}{l} (x_i - x_j)^T A(x_i - x_j) \le u \quad if\ (i,j) \in S \ [\text{similarity constraints}] \\ (x_i - x_j)^T A(x_i - x_j) \ge l \quad if\ (i,j) \in D \ [dis\text{similarity constraints}] \end{array}$$

- Advantages:
  - Simple, efficient algorithm
  - Can be applied in kernel space

JK Davis, B Kulis, P Jain, S Sra, and IS Dhillon, Information-theoretic metric learning. In *ICML*, 2007.

How to Use Learned Distance Metric?

$$
\begin{aligned}
&d(x, y) \\
&= (x-y)^T \mathbf{A} (x-y) \\
&= (x-y)^T \mathbf{A}^{\frac{1}{2}^T} \mathbf{A}^{\frac{1}{2}} (x-y) \\
&= (\mathbf{A}^{\frac{1}{2}} x - \mathbf{A}^{\frac{1}{2}} y)^T (\mathbf{A}^{\frac{1}{2}} x - \mathbf{A}^{\frac{1}{2}} y)
\end{aligned}
$$

- $\mathbf{A}^{\frac{1}{2}}$ is a linear embedding function that embeds the data into a low-dimensional binary space
- Define $\mathbf{G} = \mathbf{A}^{\frac{1}{2}}$

LSH Functions for Learned Metrics

- Given learned metric with $\mathbf{A}=\mathbf{G}^T\mathbf{G}$
- $\mathbf{G}$ should be viewed a linear parametric function or a linear embedding function for data $\boldsymbol{x}$
- Thus the LSH function could be:

Data embedding

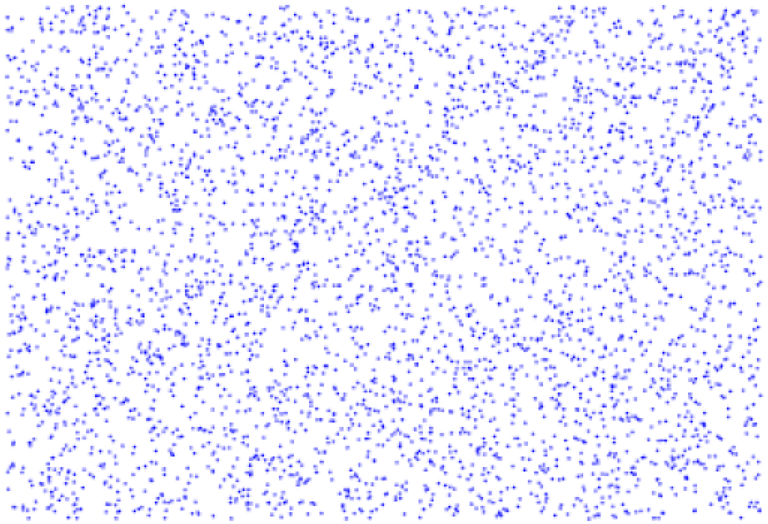
$$h_{r,\mathbf{A}}(\boldsymbol{x}) = \begin{cases} 1, & \text{if } r^T \boxed{\mathbf{G}\boldsymbol{x}} \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

- The key idea is first embed the data into a lower-dimensional binary space by $\mathbf{G}$ and then do LSH in the lower dimensional space

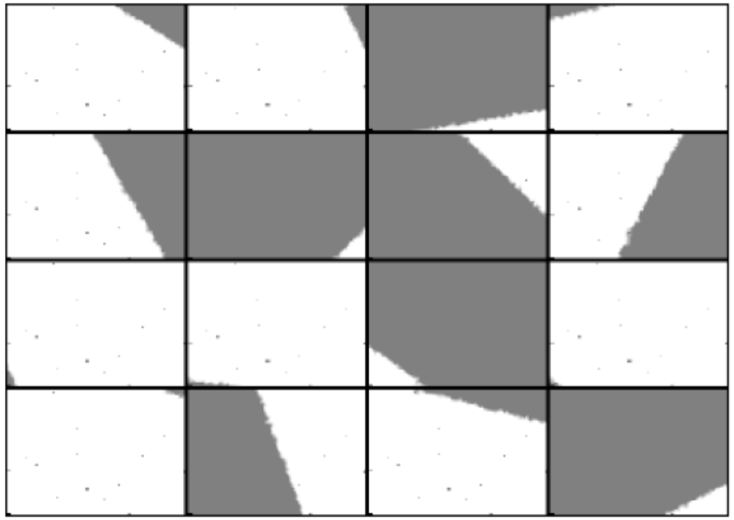
P. Jain, B. Kulis, and K. Grauman. Fast Image Search for Learned Metrics. In *CVPR*, 2008.

Toy Example

- 2D uniform distribution



Training Samples



LSH

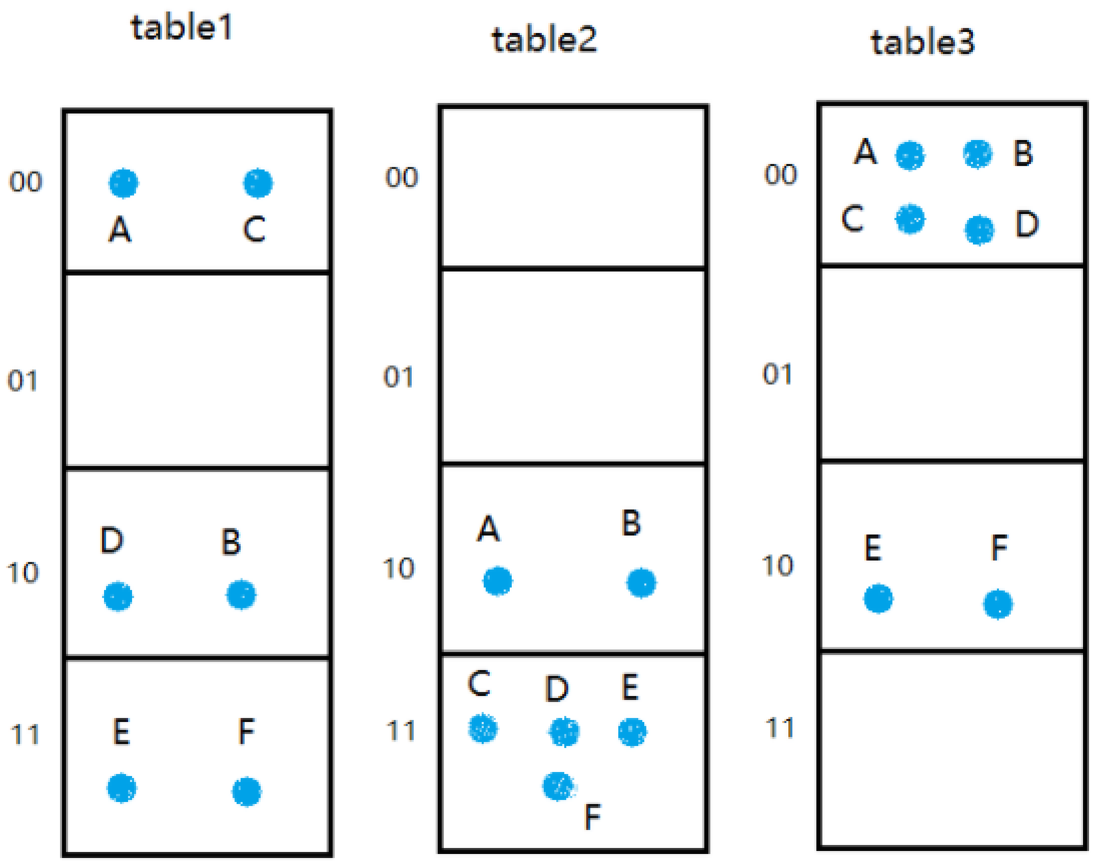
An Example

$$A = (1,1) \quad B = (2,1) \quad C = (1,2)$$
$$D = (2,2) \quad E = (4,2) \quad F = (4,3)$$

data samples

$$v(A) = 10001000$$
$$v(B) = 11001000$$
$$v(C) = 10001100$$
$$v(D) = 11001100$$
$$v(E) = 11111100$$
$$v(F) = 11111110$$

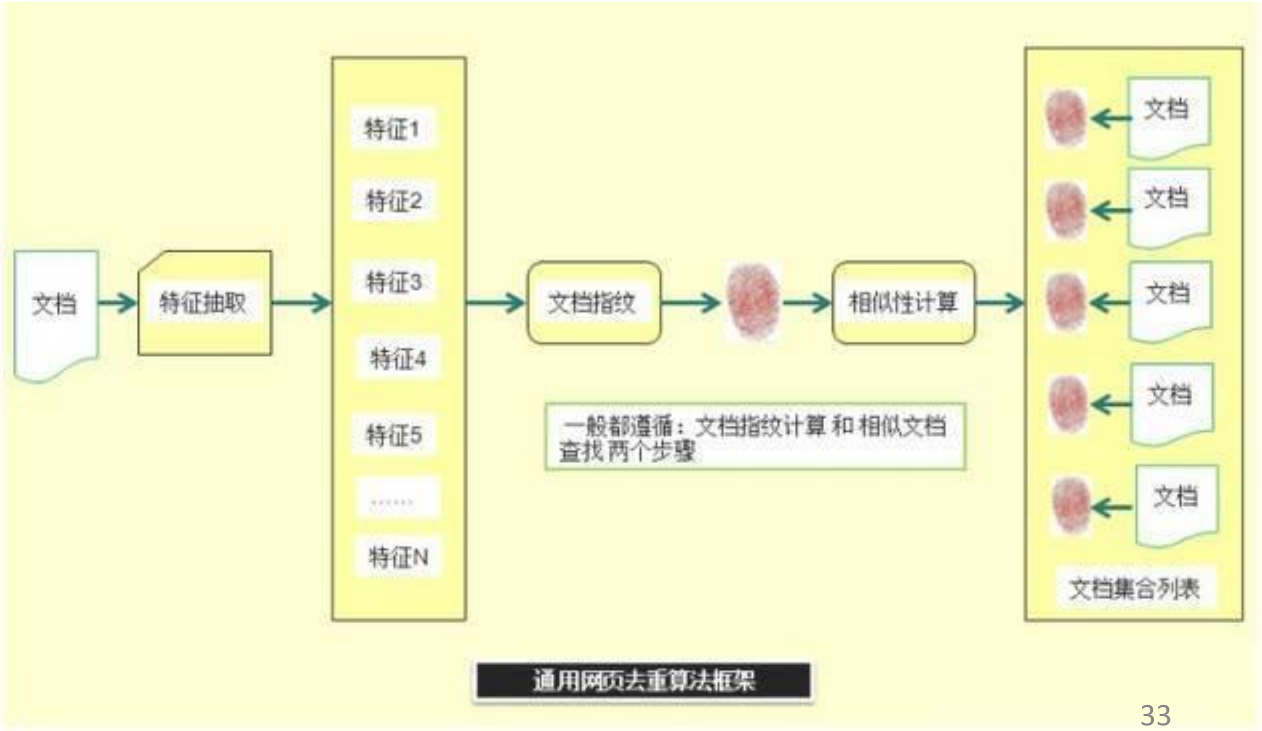
embed to low-dimensional binary space



do LSH in the lower dimensional space

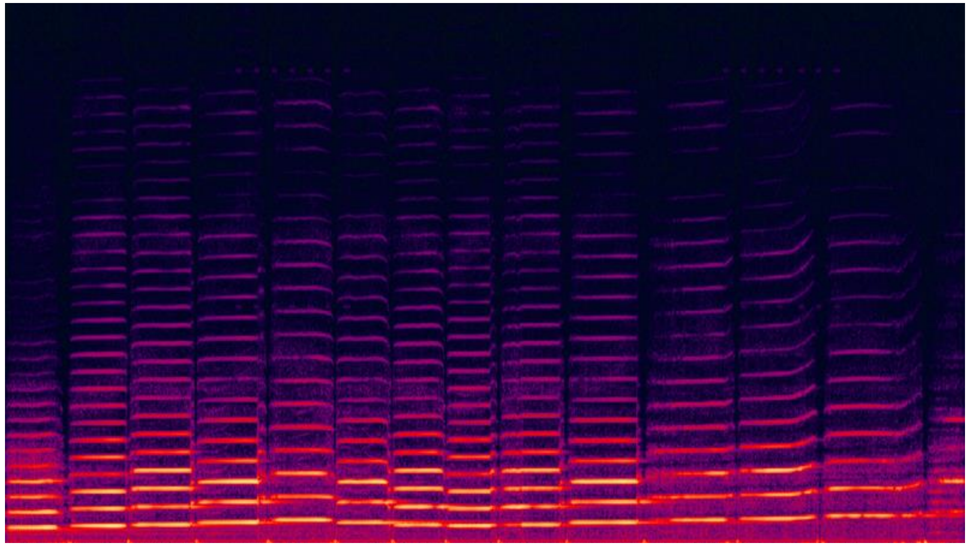
Applications

- Near-duplicate detection (近似检测)
  - 通常运用在网页去重方面
  - 在搜索中往往会遇到内容相似的重复页面，它们中大多是由于网站之间转载造成的
  - 可以对页面计算LSH，通过查找相等或相近的LSH值找到Near-duplicate



Applications

- Content based audio retrieval (基于内容的音频检索)
  - 基于傅立叶变换提取音频指纹 (audio fingerprint)
    - 频率带宽、频谱中心、谐波成分、音调
  - 针对音频指纹数据库数据量大、数据维数高的特点，采用局部敏感哈希LSH作为近似最近邻的高维数据索引算法，用于音频指纹检索



Spectrogram of violin playing.

Questions?

- Is Hashing fast enough?
- Is sub-linear search time fast enough?
- Is it scalable enough? (adapt to the memory of a PC?)

NO!

- Small binary code could do better
- Cast an image to a compact binary code, with a few **hundred bits per image**
- Small code is possible to perform real-time searches with millions from the Internet using a single large PC
- Fast: Within 1 second! (for 80 million data → 0.146 sec)
- Scalable: 80 million data (~300G) → 120M

Small Binary Code

小二值编码

Small Binary Code

- First introduced in text search/retrieval
- [Salakhutdinov and Hintion, 2007, 2009] introduced it for text documents retrieval

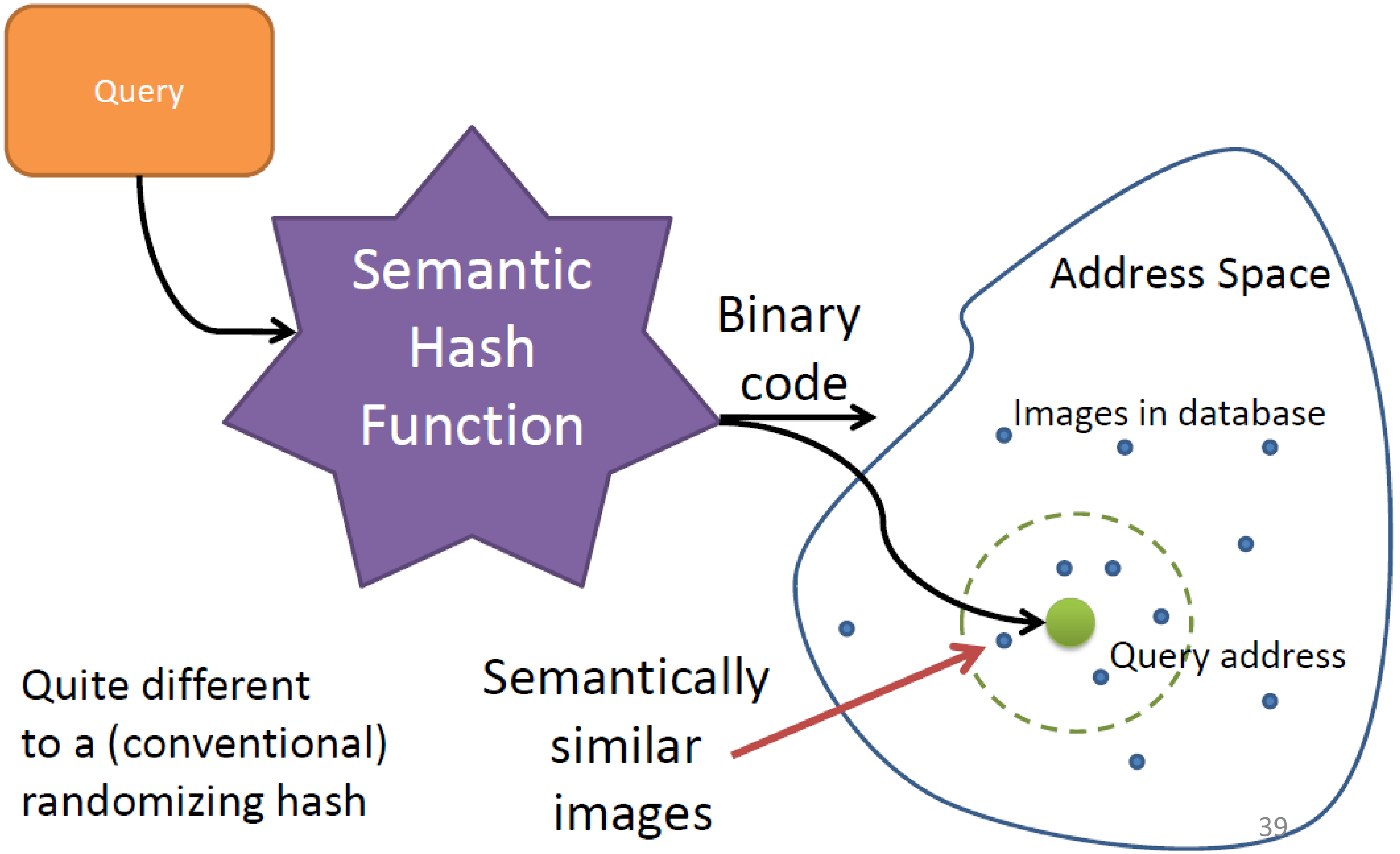
R. Salakhutdinov and G. Hinton. Semantic Hashing. In *SIGIR workshop on Information Retrieval and applications of Graphical Models*, 2007.

R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969-978, 2009.

- [Torralba et al, 2008] Introduced to computer vision

A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *CVPR*, 2008.

Semantic Hashing



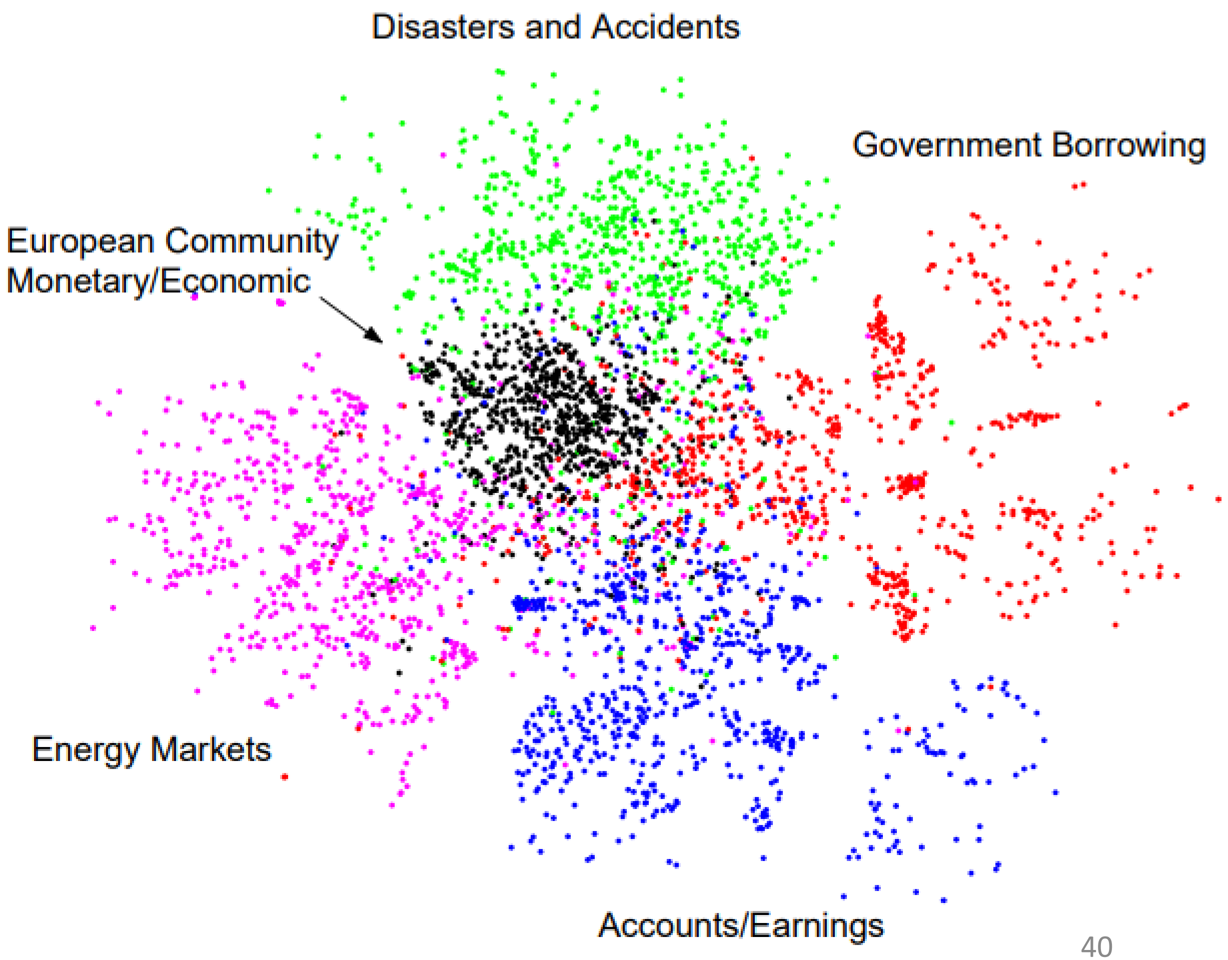
R. Salakhutdinov and G. Hinton. Semantic Hashing. In *SIGIR workshop on Information Retrieval and applications of Graphical Models*, 2007.

R. Salakhutdinov and G. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969-978, 2009.

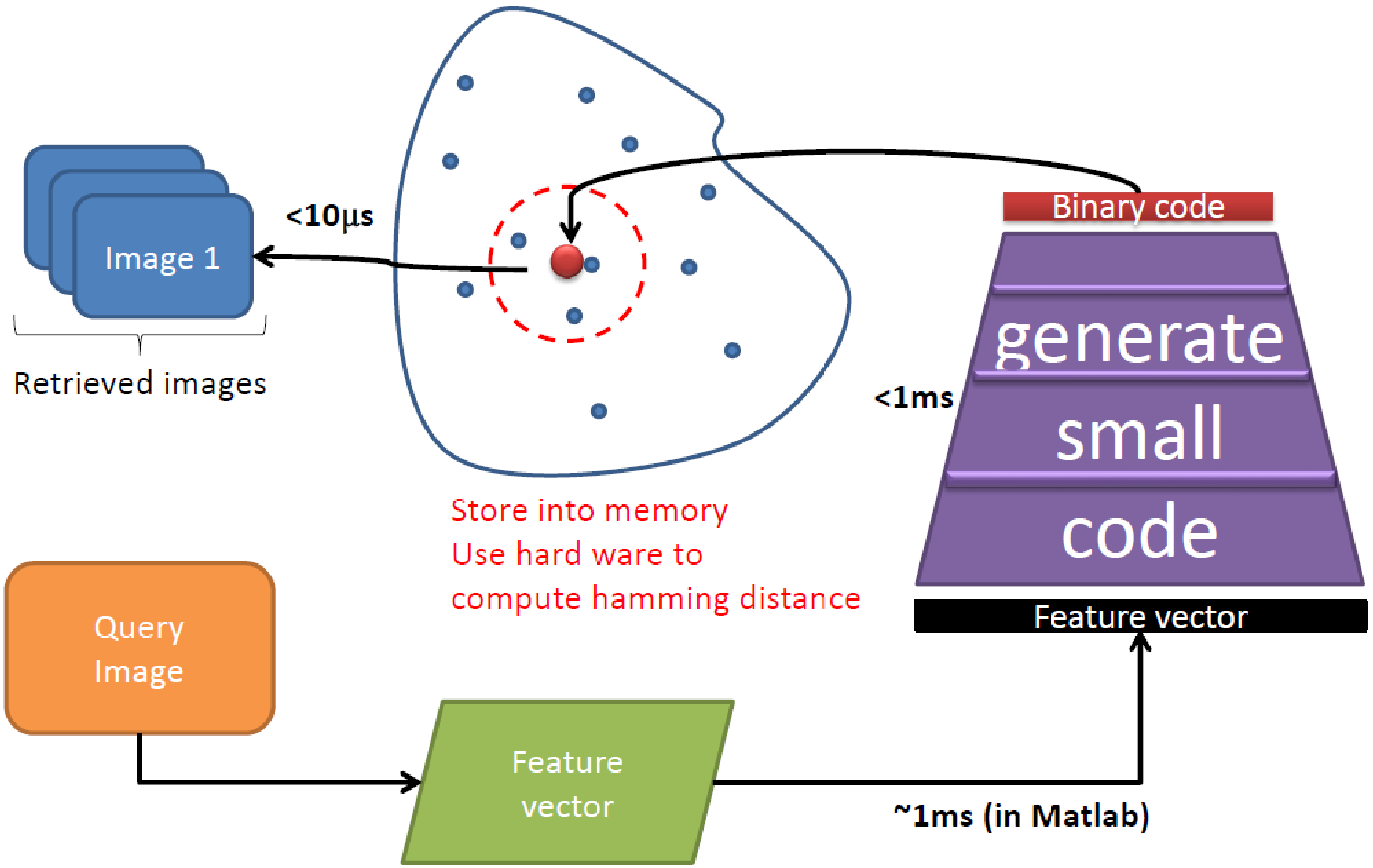
Semantic Hashing

- Similar points are mapped into a similar small code

- Then store these codes into memory and compute Hamming distance (very fast, carried out by hardware)



Overall Query Scheme



Searching Framework

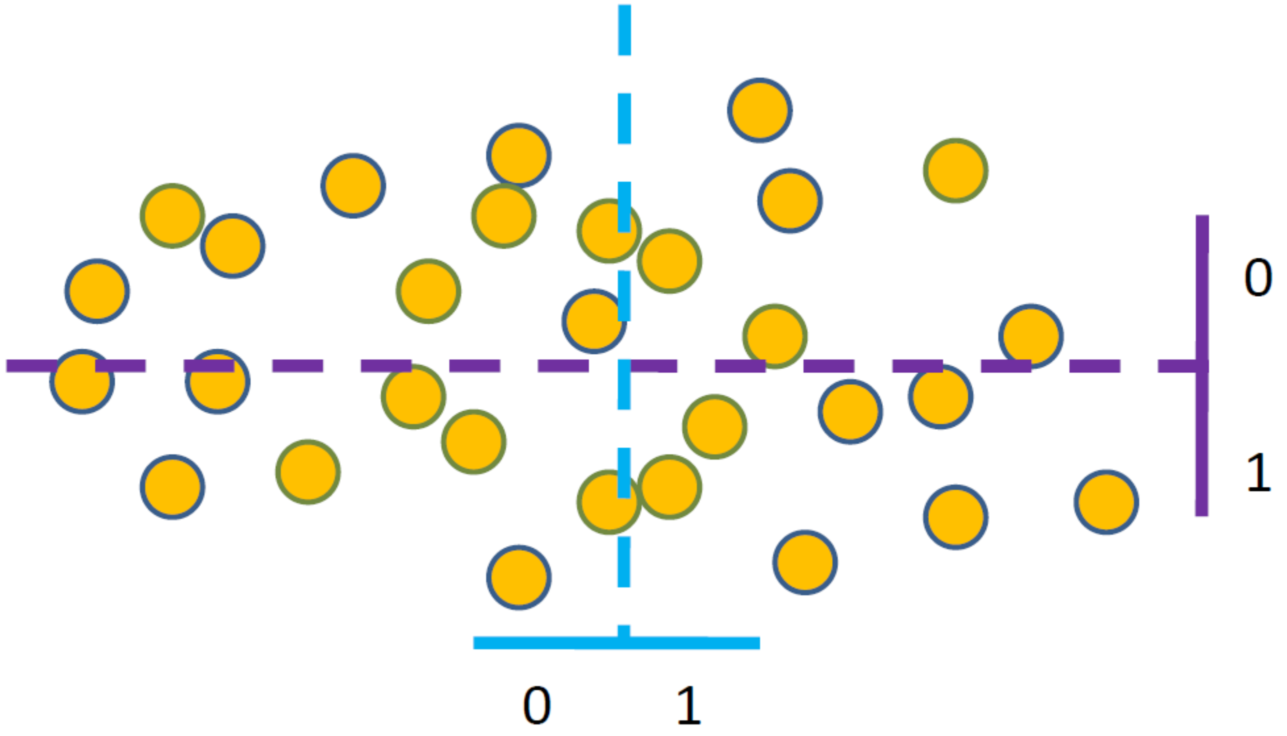
- Produce binary code (01010011010)
- Store these binary code into the memory
- Use hardware to compute the Hamming distance (very fast)
- Sort the Hamming distances and get final ranking results

How to Learn Small Binary Code?

- Simplest method (use median)
- LSH are already able to produce binary code
- Restricted Boltzmann Machines (RBM)
- Optimal small binary code by spectral hashing

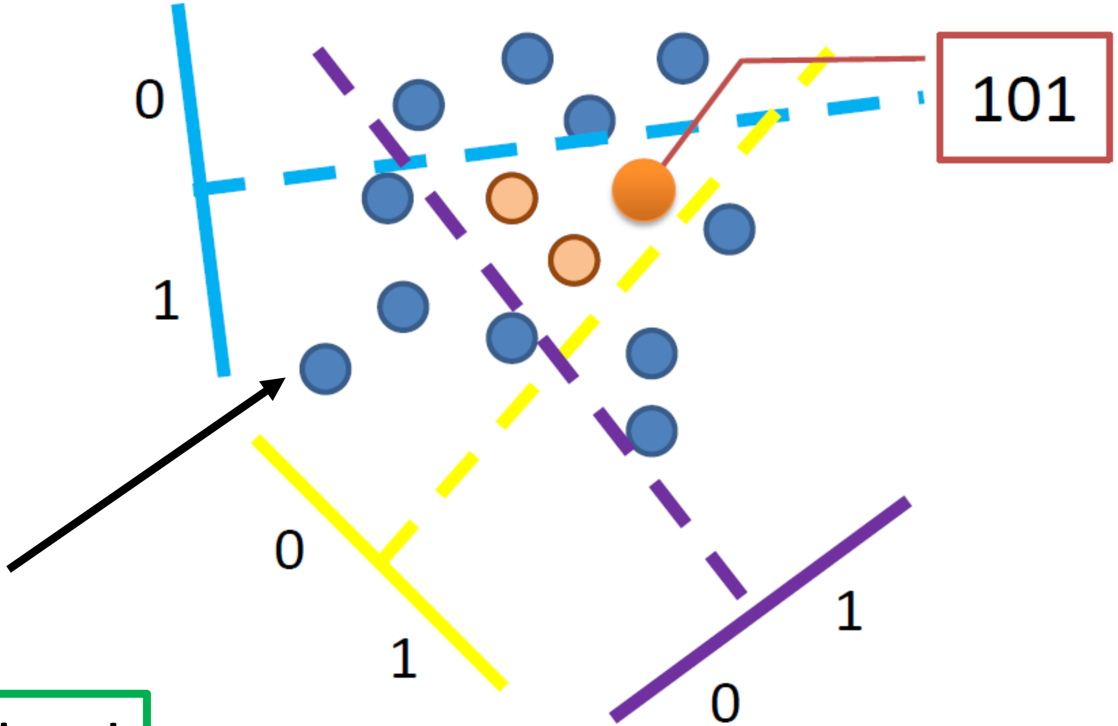
1. Simple Binarization Strategy

- Set threshold (unsupervised), e.g. use median



2. Locality Sensitive Hashing

- LSH is ready to generate binary code (unsupervised)
- Talk random projections of data
- Quantize each projection with few bits

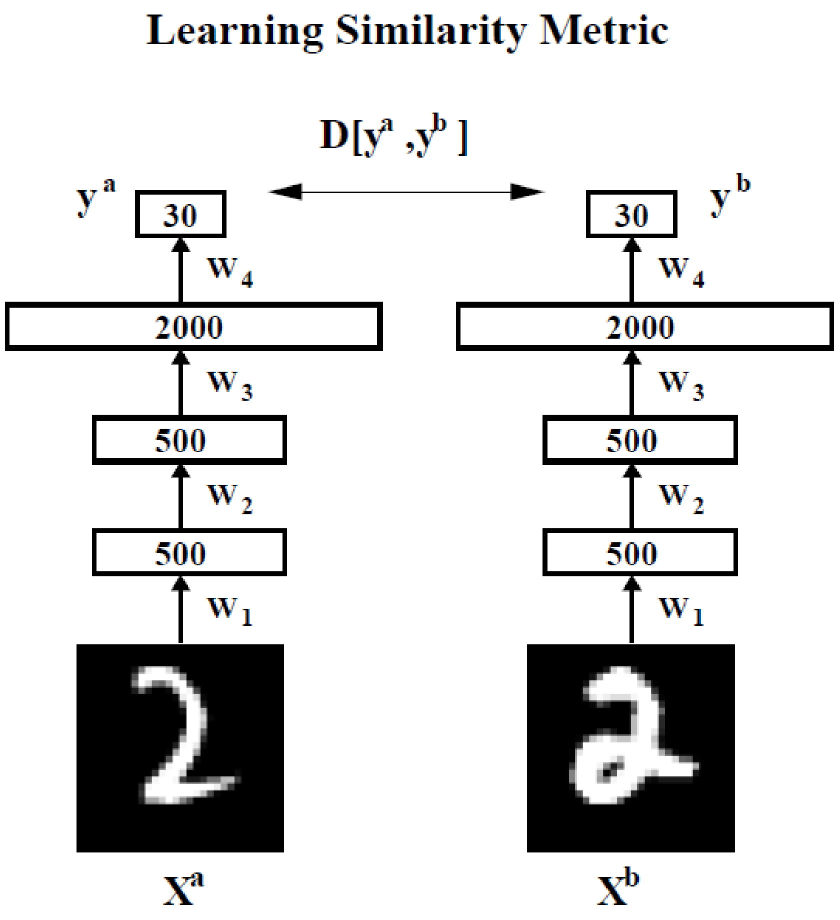


Feature vector

No learning involved

3. RBM to Generate Code

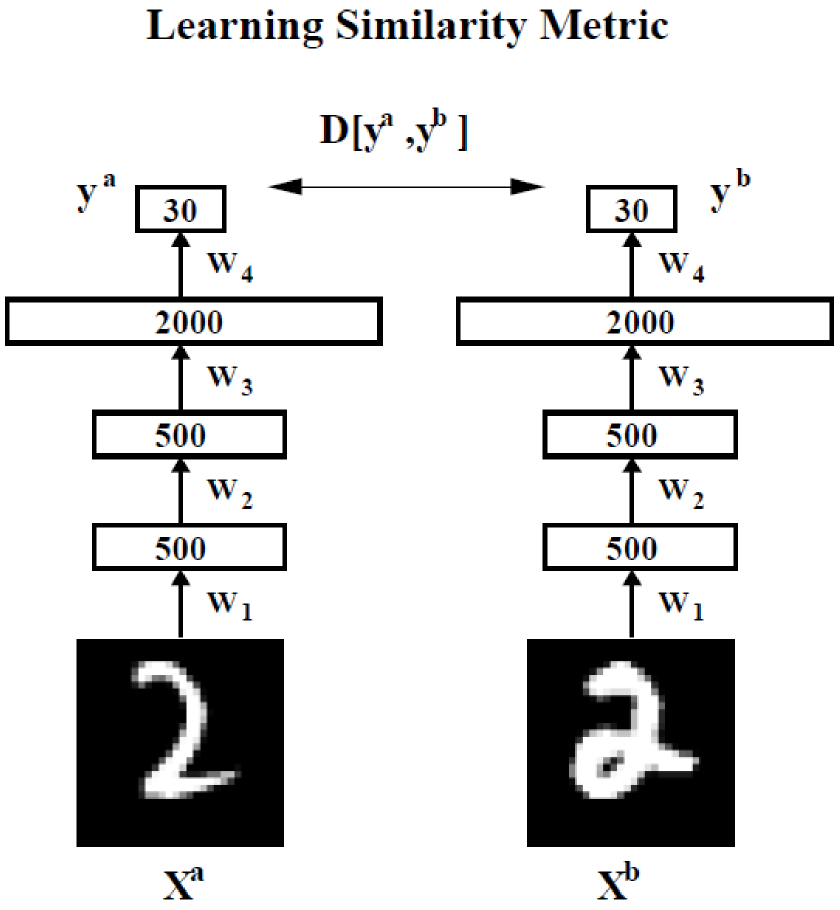
- Use a deep neural network to train small code
  - Learn the nonlinear transformation from the input MNIST image to a low-dimensional feature space in which K-nearest neighbor classification performs will
- Supervised method



R. Salakhutdinov and G. E. Hinton. Learning a Nonlinear Embedding by Preserving Class Neighbourhood Structure. In AISTATS, 2007.

3. RBM to Generate Code

- For any given distance metric D, measure similarity between two input vectors $\mathbf{x}_a, \mathbf{x}_b \in X$ by computing $D[f(\mathbf{x}_a|W), f(\mathbf{x}_b|W)]$
- Where $f(\mathbf{x}|W)$ is a function $f: X \rightarrow Y$ mapping the input vectors in X into a feature space Y and is parameterized by W

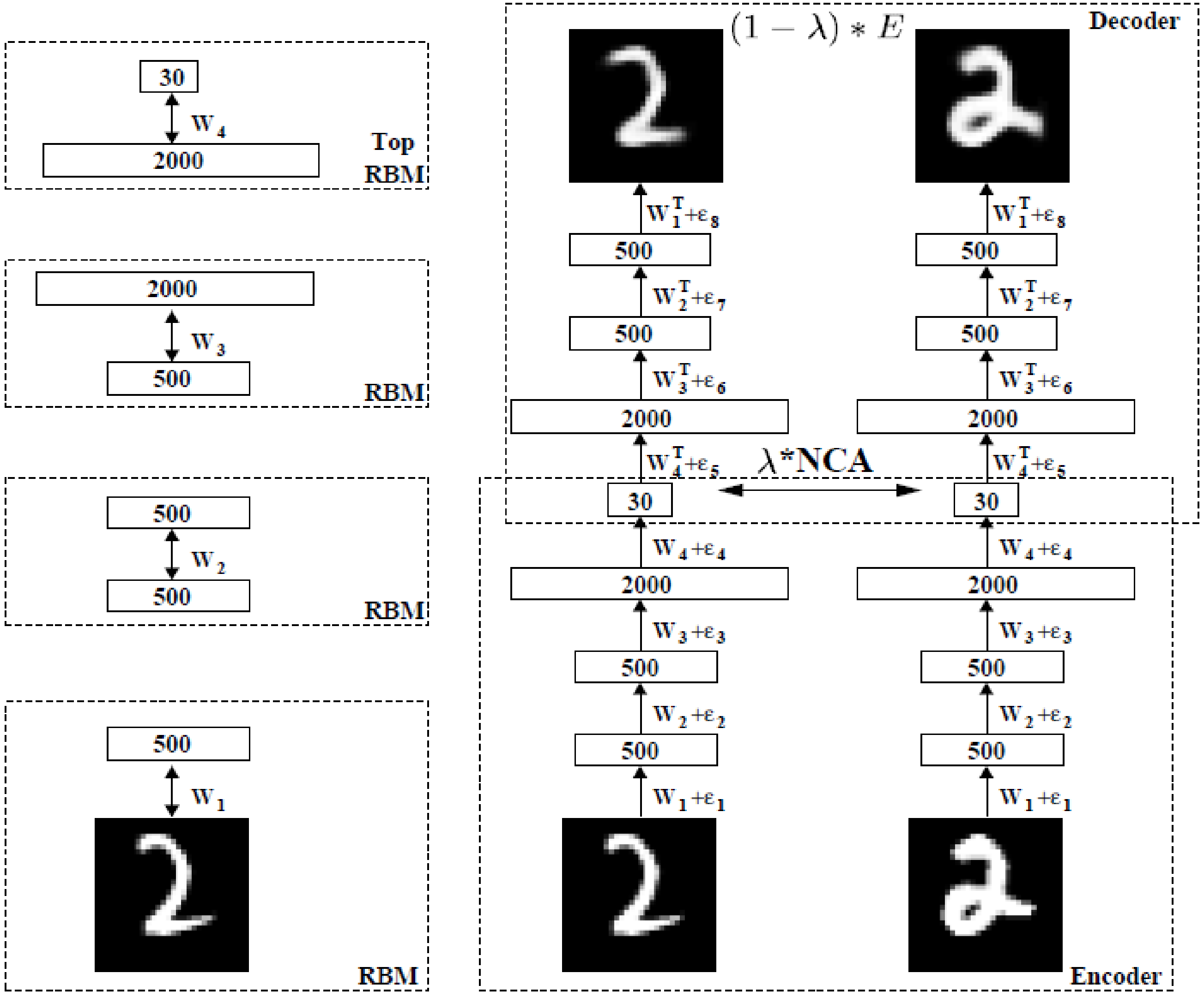


- Pretraining with a stack of RMBs
- Introduce Neighbourhood Component Analysis (NCA) as the classification error

$$O_{NCA} = \sum_{a=1}^{N} \sum_{b:c^a=c^b} p_{ab}$$

$$p_{ab} = \frac{\exp(-d_{ab})}{\sum_{z \neq a} \exp(-d_{az})}, \qquad p_{aa} = 0$$

$$d_{ab} = \| f(\mathbf{x}^a | W) - f(\mathbf{x}^b | W) \|^2$$

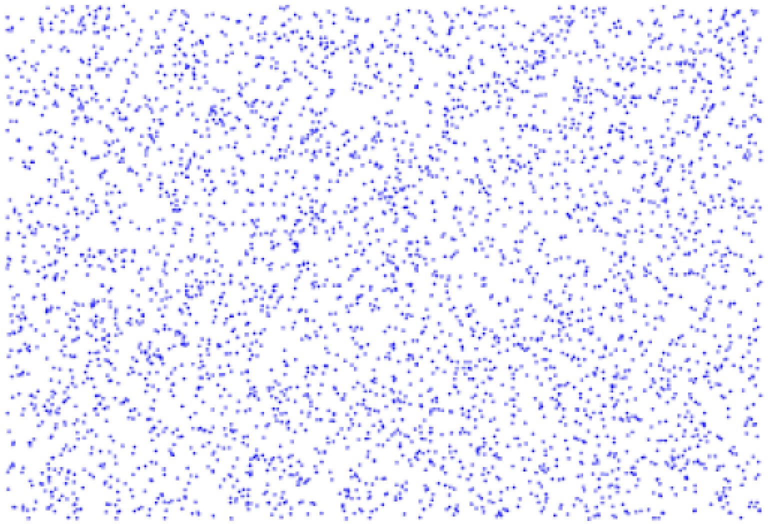


Pretraining

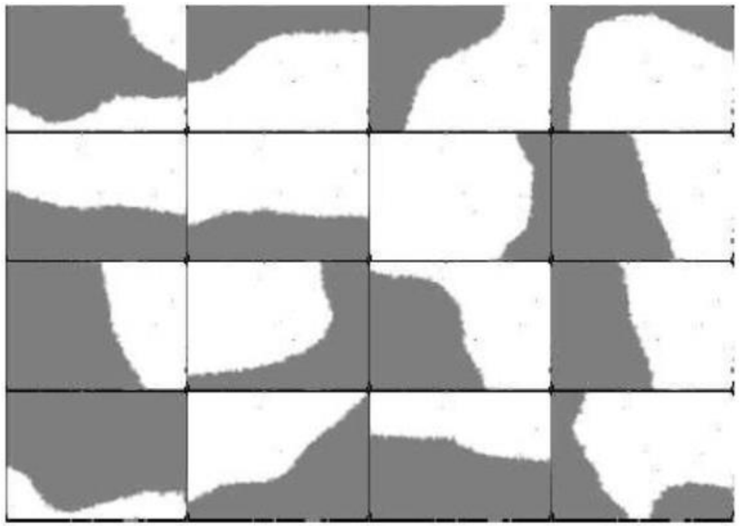
Fine−tuning

Toy Example

- 2D uniform distribution



Training Samples



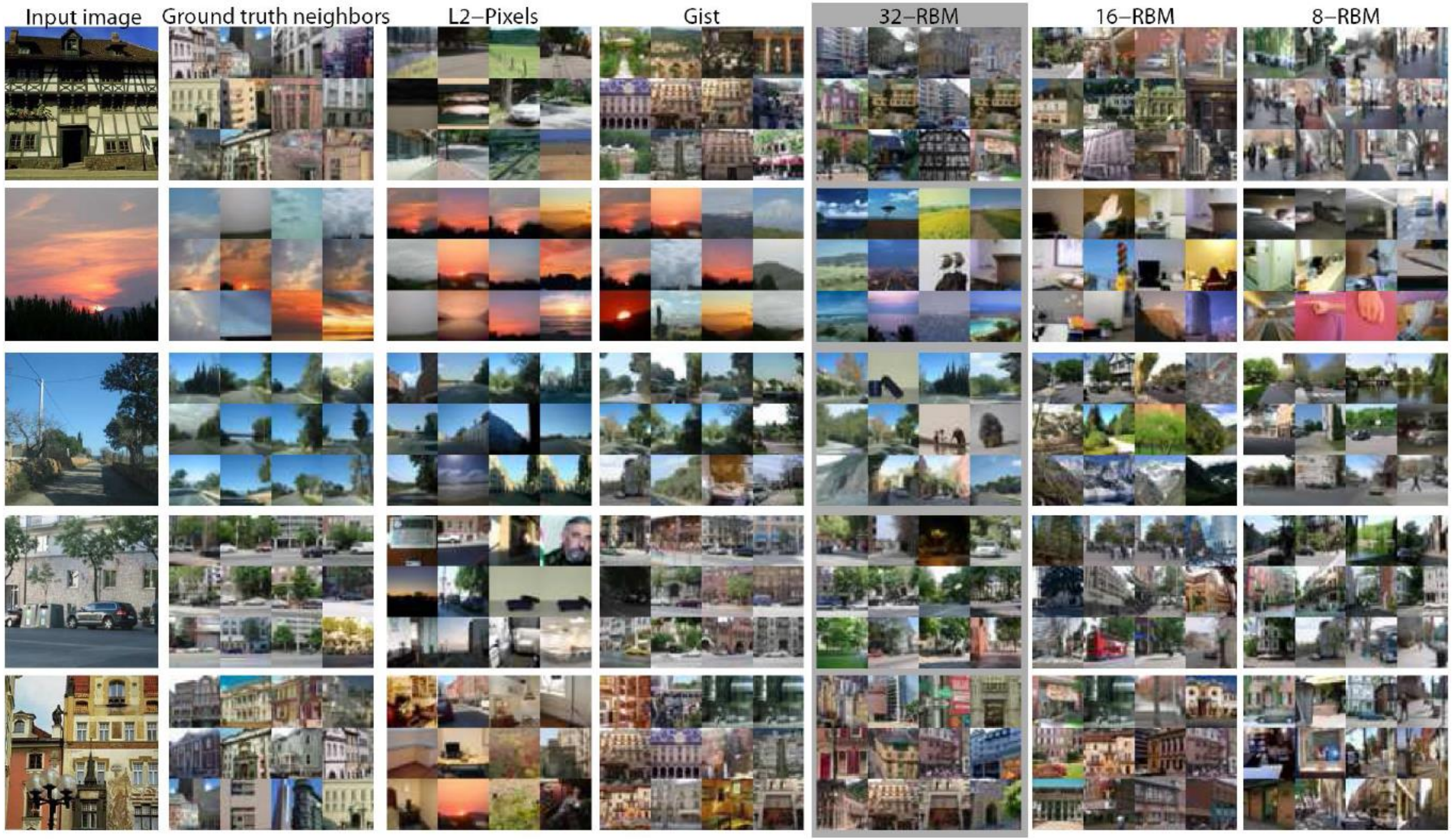
RBM (two hidden layers)

LabelMe Retrieval

- LabelMe is a large database with human annotated images
  - First generate small code
  - Use hamming distance to search for similar images
  - Sort the results to produce final ranking

LabelMe Retrieval

- 12 closest neighbors under different distance metrics



4. Spectral Hashing

Y. Weiss, A. Torralba, and R. Fergus. Spectral Hashing. In *NIPS, 2008.*

- Closely related to the problem of spectral graph partitioning
- What makes a good code?
  - easily computed for a novel input
  - requires a small number of bits to code the full dataset
  - maps similar items to similar binary code words

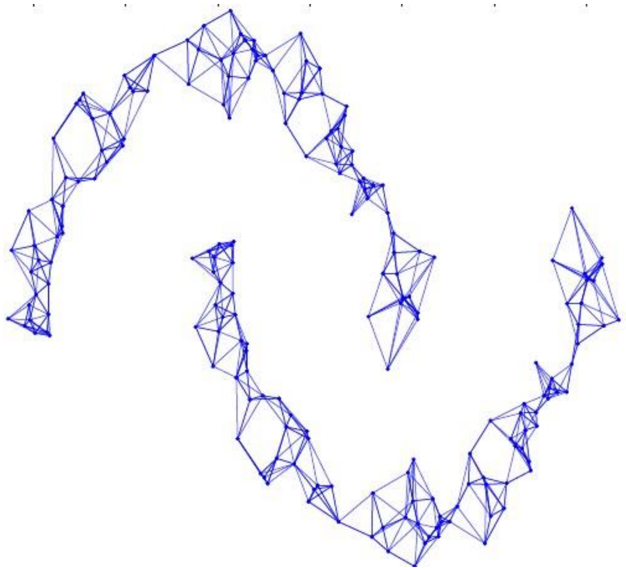
Y. Weiss, A. Torralba, and R. Fergus. Spectral Hashing.
In *NIPS*, 2008.

Spectral Hashing

- To simplify the problem, first assume that the items have already been embedded in a Euclidean space
- Try to embed the data into a Hamming space
- Hamming space is binary space 010101001⋯

Some Definitions

- Let $\{y_i\}_{i=1}^{n}$ be the list of code words (binary vector of length k) for n data points
- $W(i,j) = \exp\left(-\left\|x_i - x_j\right\|^2/\epsilon^2\right)$ is the affinity matrix characterizing similarities between data points



Objective Function

- The average Hamming distance between similar points is minimal

$$
\begin{aligned}
minimize: &\sum_{ij} W_{ij} \|y_i - y_j\|^2 \\
subject\ to: &\ y_i \in \{-1, 1\}^k \\
&\sum_i y_i = 0 \\
&\frac{1}{n} \sum_i y_i y_i^T = I
\end{aligned}
$$

- What does this objective function mean?

Objective of Spectral Hashing

$$minimize: \sum_{ij} W_{ij} \|y_i - y_j\|^2$$

the average Hamming distance between similar neighbors in the Euclidean space

$$subject\ to: y_i \in \{-1, 1\}^k$$

the code is binary: (-1, 1) → (0, 1)

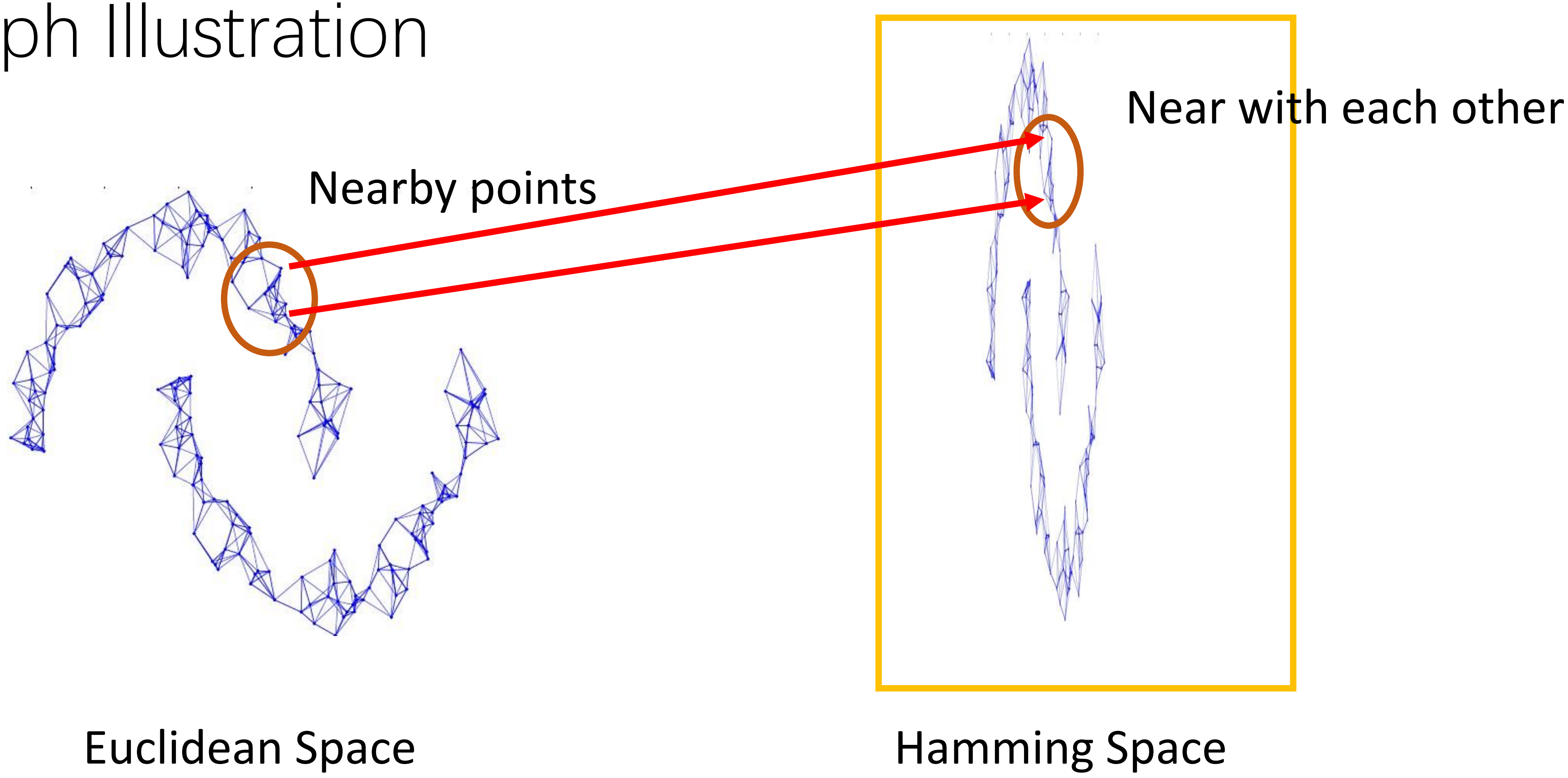
$$\sum_i y_i = 0$$

each bit has 50% to be -1 or 1

$$\frac{1}{n} \sum_i y_i y_i^T = I$$

the bits to be uncorrelated
(bounding condition for the objective)

Graph Illustration

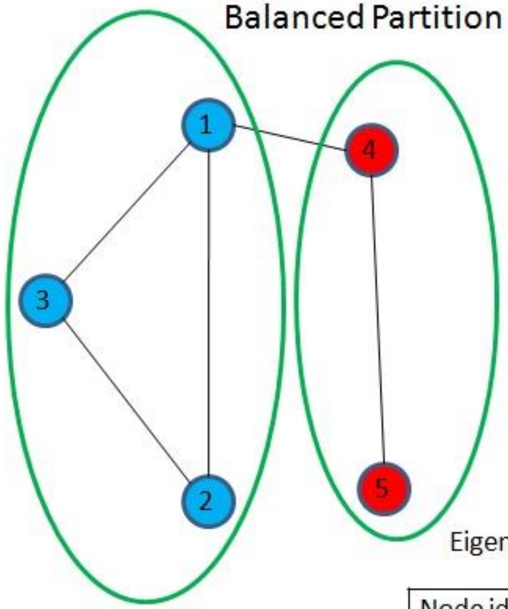


Spectral Graph Partitioning

- Given a graph $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E})$ with adjacency matrix $\boldsymbol{A}$, where an entry $\boldsymbol{A}_{ij}$ implies an edge between node $\boldsymbol{i}$ and $\boldsymbol{j}$, and degree matrix $\boldsymbol{D}$, which is a diagonal matrix, where each diagonal entry of a row $\boldsymbol{i}$, $\boldsymbol{D}_{ii}$, represents the node degree of node $\boldsymbol{i}$. The Laplacian matrix is defined as $\boldsymbol{L}=\boldsymbol{D}-\boldsymbol{A}$. Now, a ratio-cut partition for graph $\boldsymbol{G}=(\boldsymbol{V},\boldsymbol{E})$ is defined as a partition of $\boldsymbol{V}$ into disjoint $\boldsymbol{U}$ and $\boldsymbol{W}$, such that cost of $\mathrm{cut}(\boldsymbol{U},\boldsymbol{W})/(|\boldsymbol{U}|\,|\boldsymbol{W}|)$ is minimized.
- In such a scenario, the second smallest eigenvalue (λ_2) of $\boldsymbol{L}$, yields a lower bound on the optimal cost (c) of ratio-cut partition with $c \geq \lambda_2/n$. The eigenvector ($\boldsymbol{V}_2$) corresponding to λ_2, called the Fiedler vector, bisects the graph into only two communities based on the ***sign*** of the corresponding vector entry.

Spectral Graph Partitioning

Balanced Partition



L=D-A

Node id	1	2	3	4	5
1	3	-1	-1	-1	0
2	-1	2	-1	0	0
3	-1	-1	2	0	0
4	-1	0	0	2	-1
5	0	0	0	-1	1

Eigen value decomposition of L: (V)

Node id	1	2	3	4	5
1	-0.44721	0.201774	-0.317515	0	0.8114622
2	-0.44721	0.41931	0.242173	-0.707106	-0.255974
3	-0.44721	0.41931	0.24217	0.7071067	-0.2559747
4	-0.44721	-0.3379	-0.7030	0	-0.4375313
5	-0.447958	-0.70246	0.5362	0	0.13801875
	E=[0,	0.5188,	2.3111,	3.0000,	4.1701]

Spectral Relaxation

- We obtain an easy problem whose solutions are simply the k eigenvectors of D-W with minimal eigenvalues

$$\begin{aligned} minimize &: \sum_{ij} W_{ij} \|y_i - y_j\|^2 \\ subject\ to &: y_i \in \{-1, 1\}^k \\ &\sum_i y_i = 0 \\ &\frac{1}{n} \sum_i y_i y_i^T = I \end{aligned}$$

$$\Rightarrow$$

$$\begin{aligned} minimize &: trace(Y^T (D - W) Y) \\ subject\ to &: Y(i,j) \in \{-1, 1\} \\ &Y^T 1 = 0 \\ &Y^T Y = I \end{aligned}$$

$$D(i,i) = \sum_j W(i,j)$$

Spectral Relaxation

- We obtain an easy problem whose solutions are simply the k eigenvectors of D-W with minimal eigenvalues

$$\begin{aligned} minimize: trace(Y^T(D-W)Y) \\ subject\ to: Y(i,j) \in \{-1, 1\} \\ Y^T 1 = 0 \\ Y^T Y = I \end{aligned}$$

- Observation: Similar with spectral graph partition
- Could be solved by computing generalized eigenvalue problem

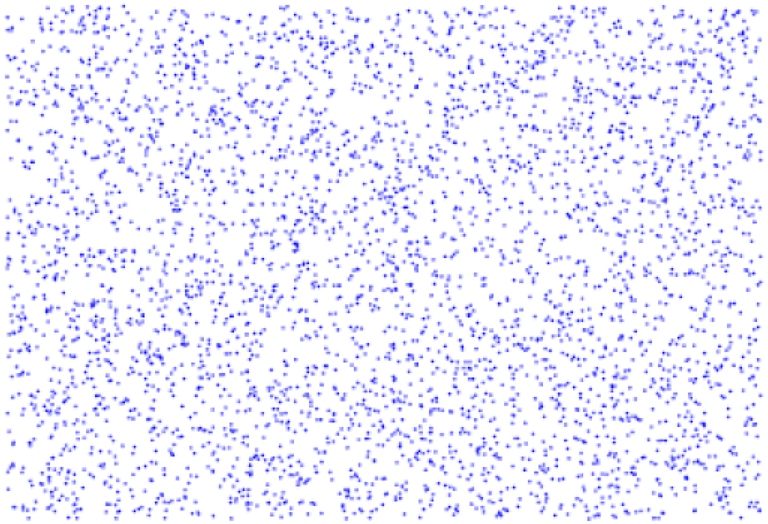
Results for Spectral Hashing

- Synthetic results on uniform distribution

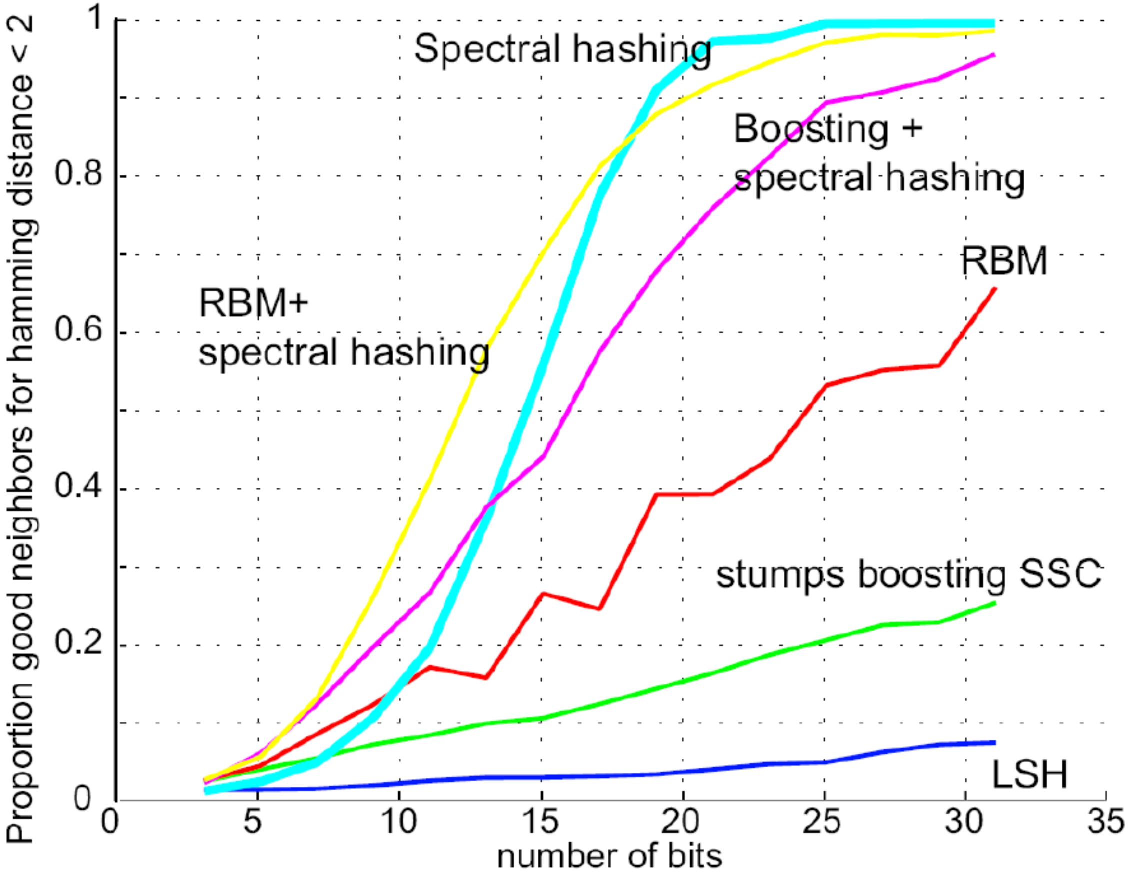
- LabelMe retrieval results using spectral hashing to produce small binary code

Toy Example Comparison

- 2D uniform distribution

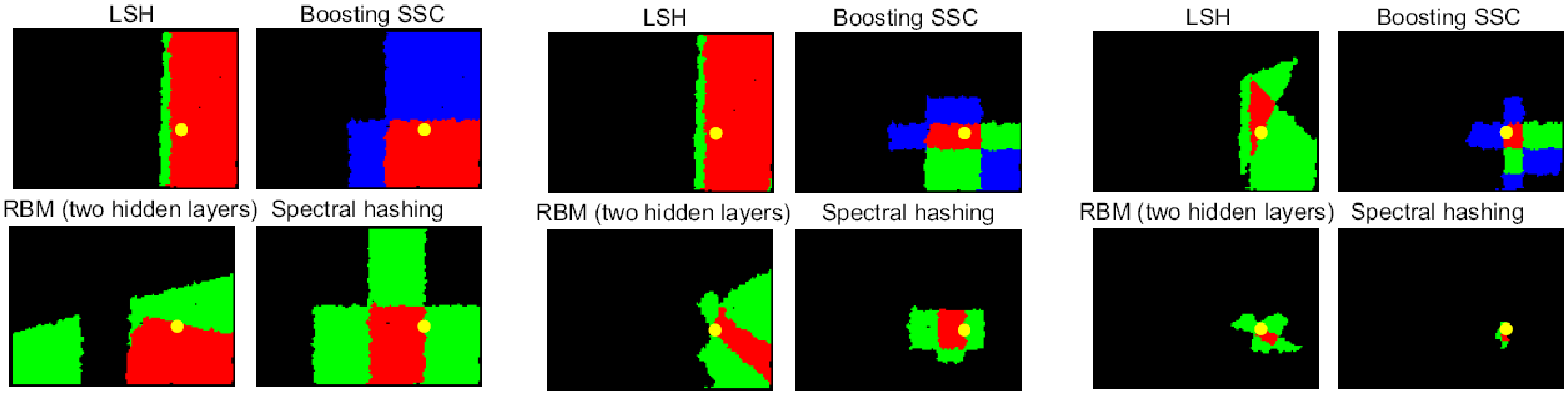


Training Samples



Toy Example Comparison

3 bits | 7 bits | 15 bits



Query Point

Distance from query point

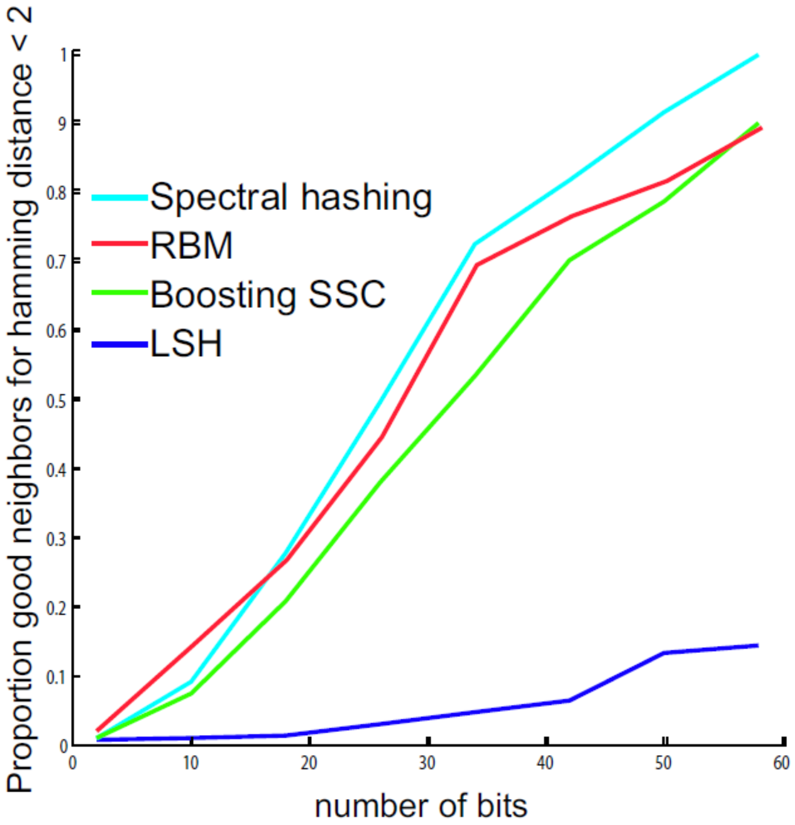
Red – 0 bit

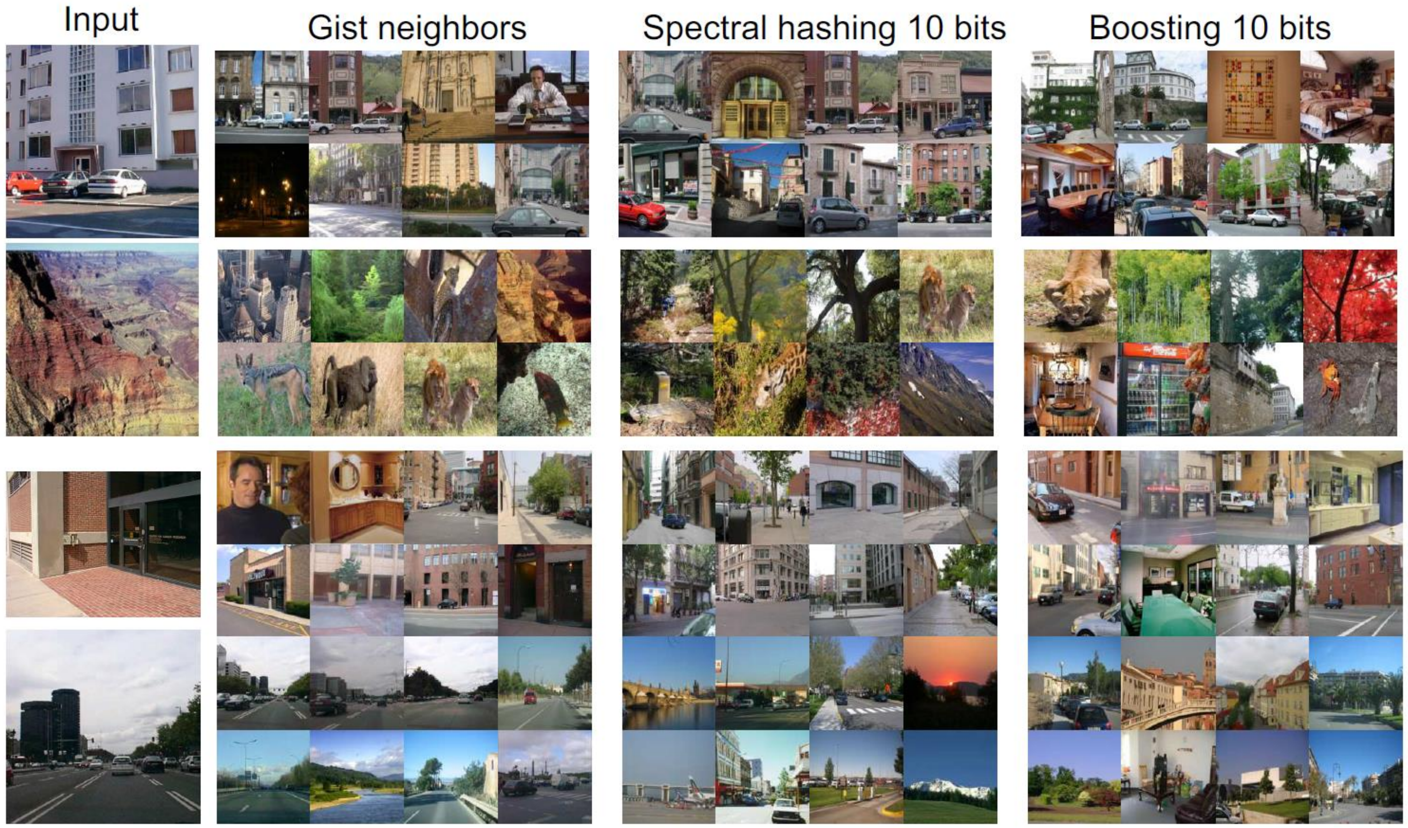
Green – 1 bit

Blue – 2 bits

Black – >2 bits

Some Results on LabelMe





Observation: spectral hashing gets the best performance

Summary

- Image search should be
  - Fast
  - Accurate
  - Scalable
- Tree based methods
- Locality Sensitive Hashing (LSH)
- Small Binary Code (state-of-the-art)

References

1. M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-Sensitive Hashing Scheme Based on p-Stable Distributions. In *ACM SOCG*, 2004.
2. P. Jain, B. Kulis, and K. Grauman. Fast Image Search for Learned Metrics. In *CVPR*, 2008.
3. R. Salakhutdinov and G. Hinton. Semantic Hashing. In *SIGIR workshop on Information Retrieval and applications of Graphical Models*, 2007.
4. R. Salakhutdinov and G. Hinton. Semantic Hashing. *International Journal of Approximate Reasoning*, 2009.
5. A. Torralba, R. Fergus, and Y. Weiss. Small Codes and Large Image Databases for Recognition. In *CVPR*, 2008.
6. Y. Weiss, A. Torralba, and R. Fergus. Spectral Hashing. In *NIPS*, 2008.

Q&A?