Extensions to Self-Taught Hashing: Kernelisation and Supervision

Dell Zhang, Jun Wang, Deng Cai, Jinsong Lu

Birkbeck, University of London
dell.z@ieee.org

The SIGIR 2010 Workshop on
Feature Generation and Selection for Information Retrieval (FGSIR)

23 July 2010, Geneva, Switzerland

Outline

1. Problem
2. Related Work
3. Review of STH
4. Extensions to STH
5. Conclusion

Problem

Similarity Search (aka Nearest Neighbour Search)

— Given a query document, find its most similar documents from a large document collection

- Information Retrieval tasks
  - near-duplicate detection, plagiarism analysis, collaborative filtering, caching, content-based multimedia retrieval, etc.
- k-Nearest-Neighbours (kNN) algorithm
  - text categorisation, scene completion/recognition, etc.

"The unreasonable effectiveness of data"

If a map could include every possible detail of the land, how big would it be?

Problem

A promising way to accelerate similarity search is
Semantic Hashing

- Design compact *binary* codes for a large number of documents so that semantically similar documents are mapped to similar codes (within a short Hamming distance)
  - Each bit can be regarded as a binary **feature**
  - Generating a few most informative binary features to represent the documents
- Then similarity search can done extremely fast by just checking a few nearby codes (memory addresses)
  - For example, 0000 $\Longrightarrow$ 0000, 1000, 0100, 0010, 0001.

Problem

Problem

Outline

1. Problem
2. Related Work
3. Review of STH
4. Extensions to STH
5. Conclusion

Related Work

Fast (Exact) Similarity Search in a *Low*-Dimensional Space

- Space-Partitioning Index
  - KD-tree, etc.
- Data Partitioning Index
  - R-tree, etc.

Related Work

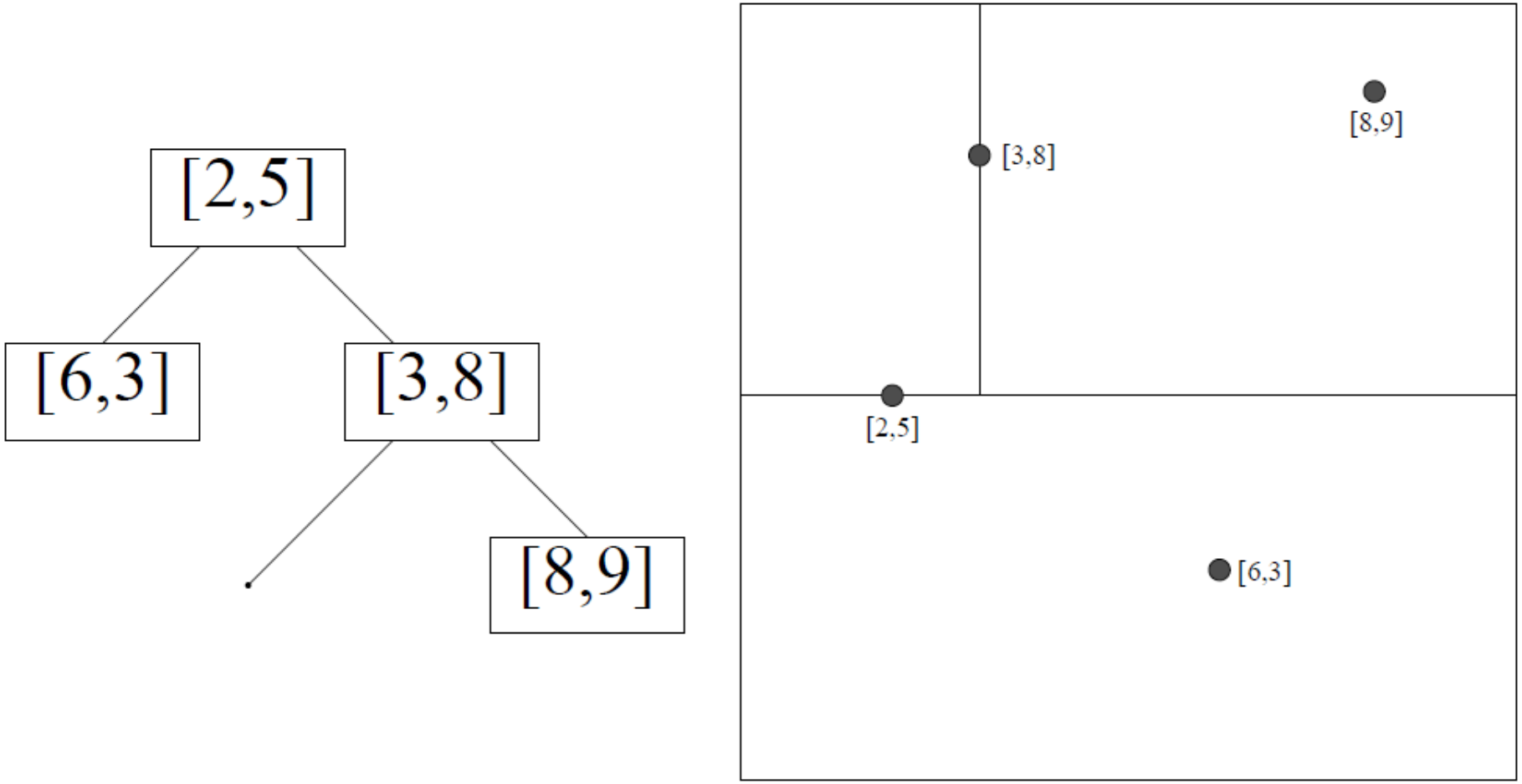


Figure: An example of KD-tree (by Andrew Moore).

Related Work

Fast (Approximate) Similarity Search in a *High*-Dimensional Space

- Data-Oblivious Hashing
  - Locality-Sensitive Hashing (LSH)
- Data-Aware Hashing
  - binarised Latent Semantic Indexing (LSI), Laplacian Co-Hashing (LCH)
  - stacked Restricted Boltzmann Machine (RBM)
  - boosting based Similarity Sensitive Coding (SSC) and Forgiving Hashing (FgH)
  - **Spectral Hashing (SpH)** — *the state of the art*
    - Restrictive assumption: the data are uniformly distributed in a hyper-rectangle

Related Work

Table: Typical techniques for accelerating similarity search.

low-dimensional space	exact similarity search	data-aware	KD-tree, R-tree
high-dimensional space	approximate similarity search	data-oblivious	LSH
		data-aware	LSI, LCH, RBM, SSC, FgH, SpH, STH

Outline

1. Problem
2. Related Work
3. Review of STH
4. Extensions to STH
5. Conclusion

Review of STH

Input:

- $X = \{\mathbf{x}_i\}_{i=1}^{n} \subset \mathbb{R}^m$

Output:

- $f(\mathbf{x}) \in \{-1, +1\}^l$: hash function
  - -1 = bit off; $+1$ = bit on
  - $l \ll m$

Review of STH

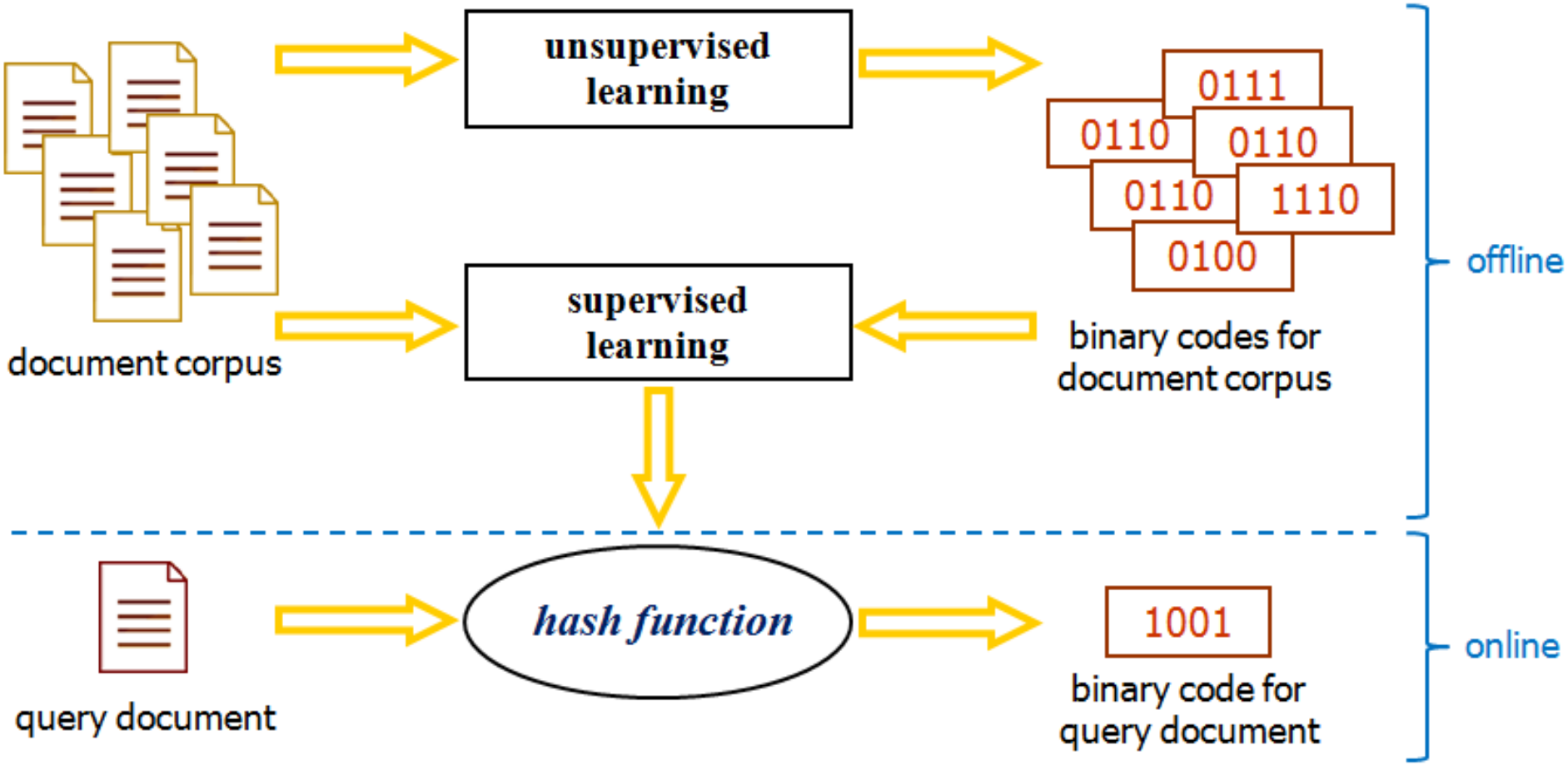


Figure: The proposed STH approach to semantic hashing.

Review of STH

Stage 1: Learning of Binary Codes

- Let $\mathbf{y}_i \in \{-1, +1\}^l$ represent the binary code for document vector $\mathbf{x}_i$
  - -1 = bit off; $+1$ = bit on.
- Let $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^T$

Review of STH

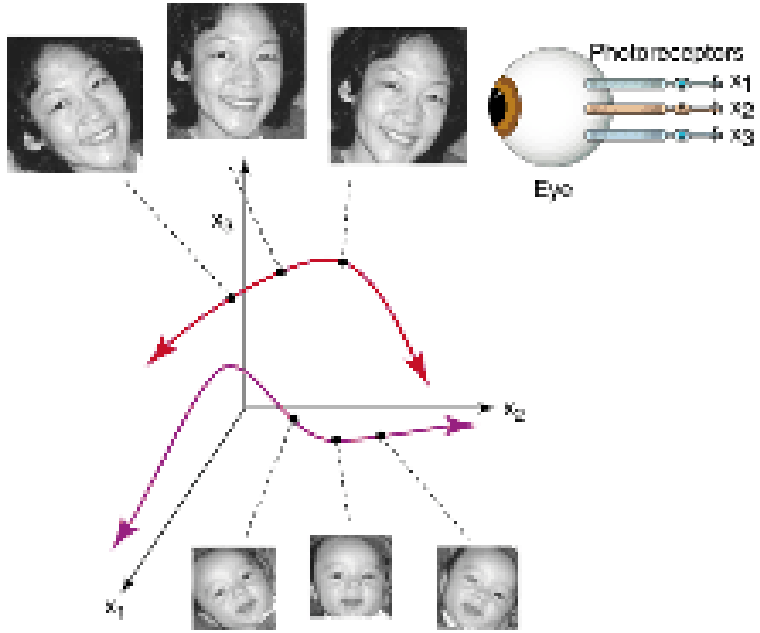
Criterion 1a: Similarity Preserving

- We focus on the *local* structure of data
- $N_k(\mathbf{x})$: the set of k-nearest-neighbours of document $\mathbf{x}$
- The local similarity matrix $\mathbf{W}$
  - i.e., the adjacency matrix of the k-nearest-neighbours graph
  - symmetric and sparse

$$W_{ij} = \begin{cases} \left(\frac{\mathbf{x}_i^T}{\|\mathbf{x}_i\|}\right) \cdot \left(\frac{\mathbf{x}_j}{\|\mathbf{x}_j\|}\right) & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

$$W_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or } \mathbf{x}_j \in N_k(\mathbf{x}_i) \\ 0 & \text{otherwise} \end{cases}$$

Review of STH



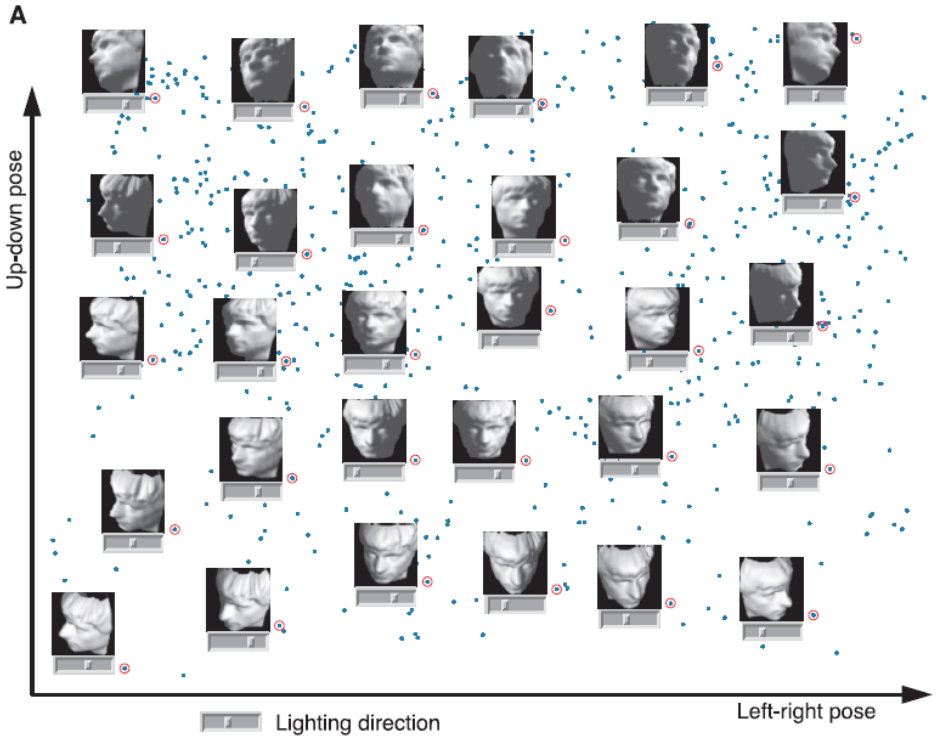


Figure: The local structure of data in a high-dimensional space.

Review of STH

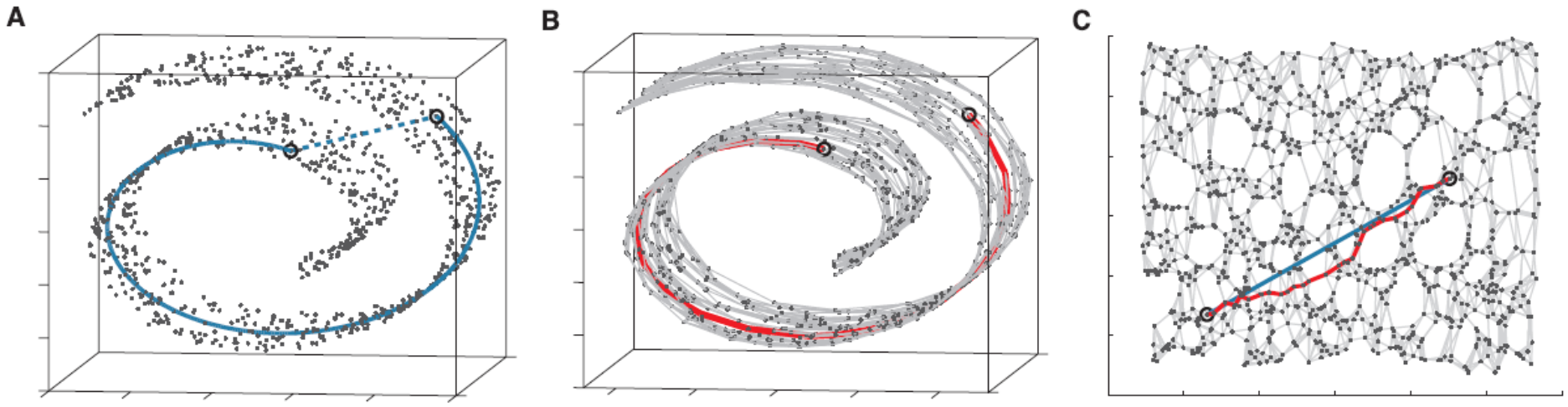


Figure: Manifold analysis: exploiting the local structure of data.

Review of STH

Criterion 1a: Similarity Preserving

- The Hamming distance between two codes y_i and y_j is

$$\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{4}$$

- We minimise the weighted total Hamming distance, as it incurs a heavy penalty if two similar documents are mapped far apart

$$\sum_{i=1}^{n}\sum_{j=1}^{n} W_{ij}\frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{4}$$

  - The squared error of distance would lead to a non-convex optimisation problem

Review of STH

Spectral Methods for Manifold Analysis
— Minimising Cut-Size

For single-bit codes $\mathbf{f} = (y_1, \ldots, y_n)^T$:

$$S = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \frac{(y_i - y_j)^2}{4} = \frac{1}{4} \mathbf{f}^T \mathbf{L} \mathbf{f}$$

- Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W}$
  - $\mathbf{D} = \mathrm{diag}(k_1, \ldots, k_n)$ where $k_i = \sum_j W_{ij}$

Review of STH

Spectral Methods for Manifold Analysis
— Minimising Cut-Size

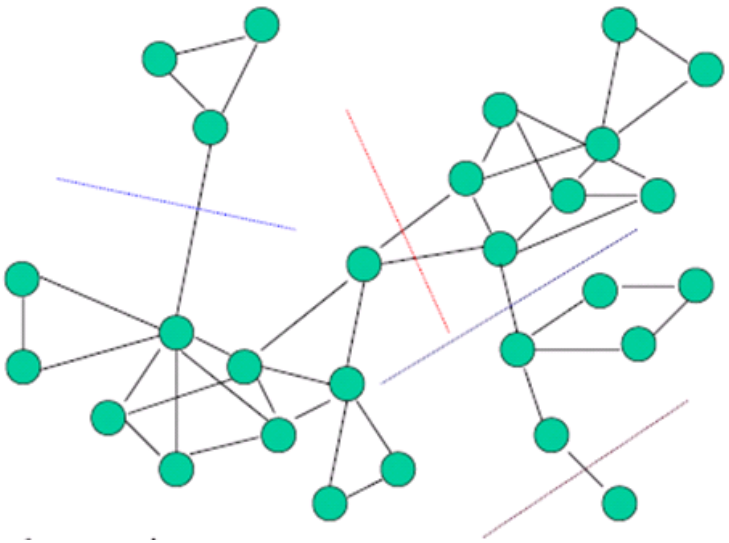


Figure: Spectral graph partitioning through *Normalised Cut*.

Review of STH

Spectral Methods for Manifold Analysis
— Minimising Cut-Size

- Real relaxation
  - Requiring $y_i \in \{-1, +1\}$ makes the problem NP hard
  - Substitute $\tilde{y}_i \in \mathbb{R}$ for y_i
- $\mathbf{L}$ is positive semi-definite
  - eigenvalues: $0 = \lambda_1 = \ldots = \lambda_z < \lambda_{z+1} \leq \ldots \leq \lambda_n$
  - eigenvectors: $\mathbf{u}_1, \ldots, \mathbf{u}_z, \mathbf{u}_{z+1}, \ldots, \mathbf{u}_n$
- Optimal non-trivial division: $\mathbf{f} = \mathbf{u}_{z+1}$
  - The number of edges across clusters is small

Review of STH

Spectral Methods for Manifold Analysis
— Minimising Cut-Size

For l-bit codes $\mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_n]^T$:

$$S = \sum_{i=1}^{n} \sum_{j=1}^{n} W_{ij} \frac{\|\mathbf{y}_i - \mathbf{y}_j\|^2}{4} = \frac{1}{4} \mathrm{Tr}(\mathbf{Y}^T \mathbf{L} \mathbf{Y})$$

- Let $\tilde{\mathbf{Y}}$ be the real relaxation of $\mathbf{Y}$

Review of STH

Spectral Methods for Manifold Analysis
— Minimising Cut-Size

- Laplacian Eigenmap (LapEig)

$$\begin{aligned} \arg\min_{\tilde{\mathbf{Y}}} \quad & \mathrm{Tr}(\tilde{\mathbf{Y}}^T \mathbf{L} \tilde{\mathbf{Y}}) \\ \text{subject to} \quad & \tilde{\mathbf{Y}}^T \mathbf{D} \tilde{\mathbf{Y}} = \mathbf{I} \\ & \tilde{\mathbf{Y}}^T \mathbf{D} \mathbf{1} = \mathbf{0} \end{aligned}$$

- Generalised Eigenvalue Problem

$$\mathbf{L}\mathbf{v} = \lambda \mathbf{D}\mathbf{v} \tag{1}$$

$$\tilde{\mathbf{Y}} = [\mathbf{v}_1, \ldots, \mathbf{v}_l]$$

Review of STH

Criterion 1b: Entropy Maximising

Best utilisation of the hash table
= Maximum entropy of the codes
= Uniform distribution of the codes (each code has equal probability)

- The p-th bit is on for half of the corpus and off for the other half

$$y_i^{(p)} = \begin{cases} +1 & \tilde{y}_i^{(p)} \geq \text{median}(\mathbf{v}_p) \\ -1 & \text{otherwise} \end{cases}$$

- The bits at different positions are almost mutually uncorrelated, as the eigenvectors given by LapEig are orthogonal to each other

Review of STH

Stage 2: Learning of Hash Function

How to get the codes for new documents previously unseen?
— Out-of-Sample Extension

- High computational complexity
  - Nystrom method
  - Linear approximation (e.g., LPI)
- Restrictive assumption about data distribution
  - Eigenfunction approximation (e.g., SpH)

Review of STH

Stage 2: Learning of Hash Function

- We reduce it to a supervised learning problem
  - Think of each bit $y_i^{(p)} \in \{+1, -1\}$ in the binary code for document $\mathbf{x}_i$ as a binary class label (class-"on" or class-"off") for that document
  - Train a binary classifier $y^{(p)} = f^{(p)}(\mathbf{x})$ on the given corpus that has already been "labelled" by the 1st stage
  - Then we can use the learned binary classifiers $f^{(1)}, \ldots, f^{(l)}$ to predict the l-bit binary code $y^{(1)}, \ldots, y^{(l)}$ for any query document $\mathbf{x}$

Review of STH

Kernel Methods for *Pseudo*-Supervised Learning
— Support Vector Machine (SVM)
$y^{(p)} = f^{(p)}(\mathbf{x}) = \text{sgn}(\mathbf{w}^T\mathbf{x})$

$$\begin{aligned} \underset{\mathbf{w},\xi_i \geq 0}{\arg\min} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + \frac{C}{n}\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & \forall_{i=1}^{n} : y_i^{(p)}\mathbf{w}^T\mathbf{x}_i \geq 1 - \xi_i \end{aligned} \tag{2}$$

- large-margin classification $\longrightarrow$ good generalisation
- linear/non-linear kernels $\longrightarrow$ linear/non-linear mapping
- convex optimisation $\longrightarrow$ global optimum

Review of STH

Self-Taught Hashing (STH): The **Learning** Process

1. Unsupervised Learning of Binary Codes
   - Construct the k-nearest-neighbours graph for the given corpus
   - Embed the documents in an l-dimensional space through LapEig (1) to get an l-dimensional real-valued vector for each document
   - Obtain an l-bit binary code for each document via thresholding the above vectors at their median point, and then take each bit as a binary class label for that document
2. Supervised Learning of Hash Function
   - Train l SVM classifiers (2) based on the given corpus that has been "labelled" as above

Review of STH

Self-Taught Hashing (STH): The **Prediction** Process

1. Classify the query document using those l learned classifiers
2. Assemble the output l binary labels into an l-bit binary code

Outline

1. Problem
2. Related Work
3. Review of STH
4. Extensions to STH
5. Conclusion

Extension I: Kernelisation

In the second stage of STH, we rewrite the SVM quadratic optimisation problem (2) into its dual form

$$\begin{aligned} \arg\min_{\alpha} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i^{(p)} y_j^{(p)} \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n \\ & \sum_{i=1}^{n} \alpha_i y_i^{(p)} = 0 \end{aligned} \tag{3}$$

and replace the inner product between $\mathbf{x}_i$ and $\mathbf{x}_j$ by a nonlinear kernel such as the Gaussian kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \tag{4}$$

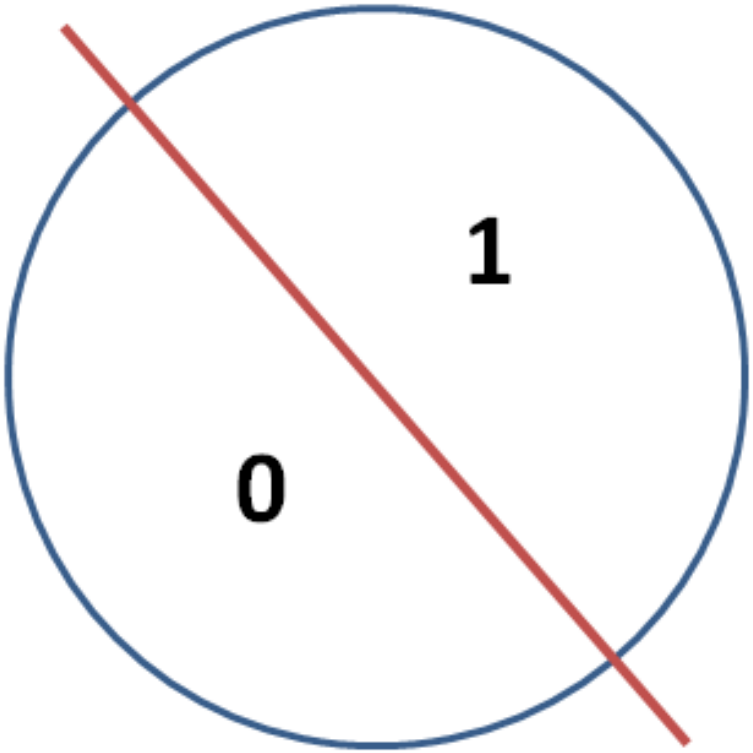
Extension I: Kernelisation

Then the p-th bit (i.e., binary feature) of the binary code for a query document $\mathbf{x}$ would be given by

$$f^{(p)}(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^{n} \alpha_i y_i^{(p)} K(\mathbf{x}, \mathbf{x}_i)\right) \tag{5}$$

which is a nonlinear mapping.

Extension I: Kernelisation



- For example, using 16-bit binary codes,
  - linear hashing: $2l = 2 \times 16 = 32$ sectors
  - nonlinear hashing: $2^l = 2^{16} = 65536$ pieces

Extension I: Kernelisation

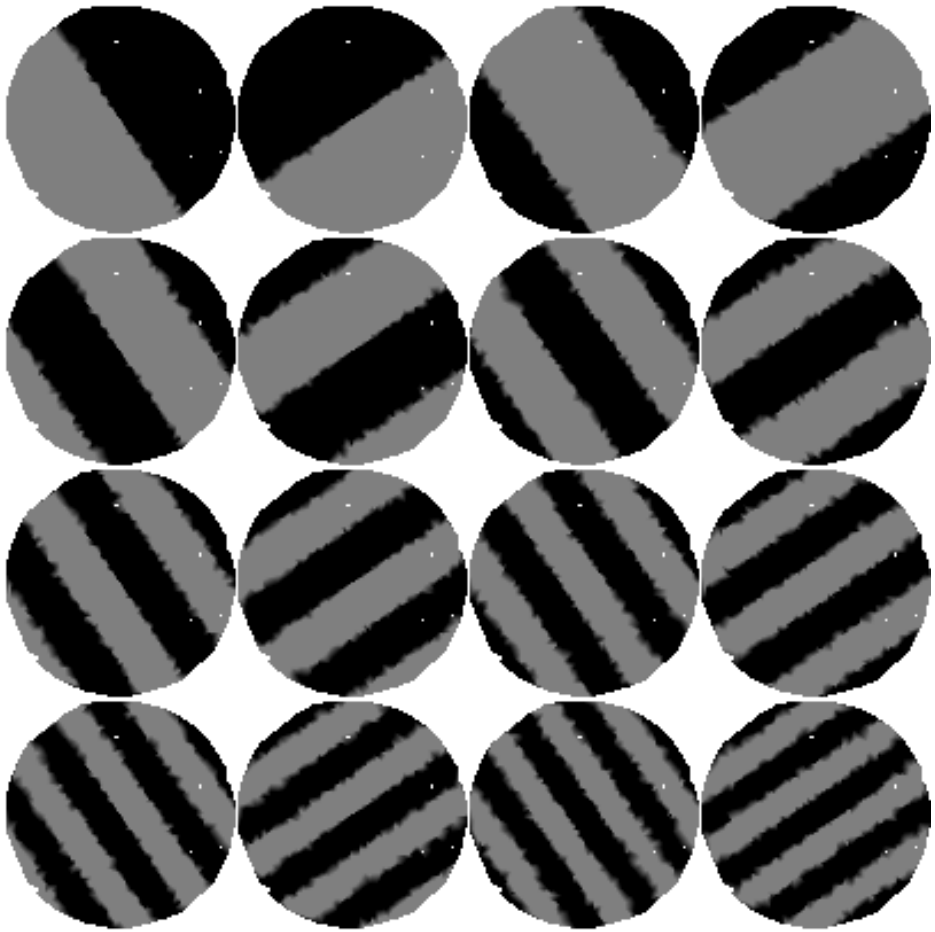


Figure: The 16-bit hash function for the pie dataset using SpH.

Extension I: Kernelisation

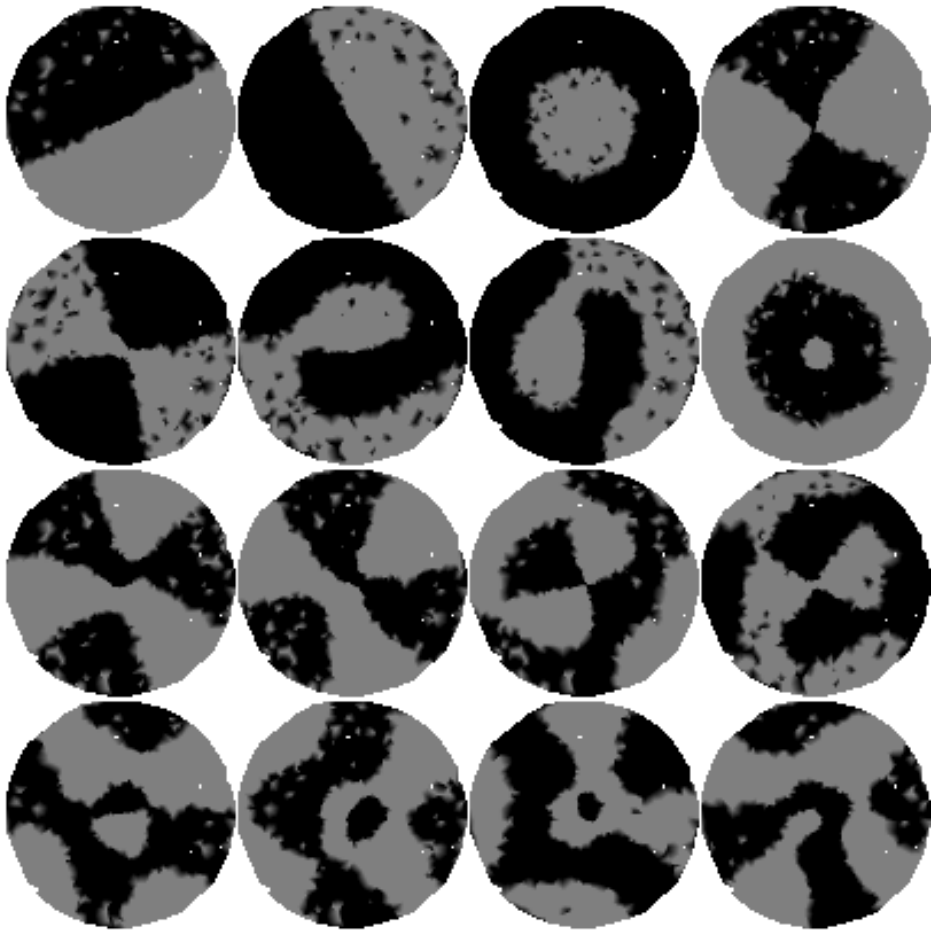


Figure: The 16-bit hash function for thepie dataset using STH.

Extension I: Kernelisation

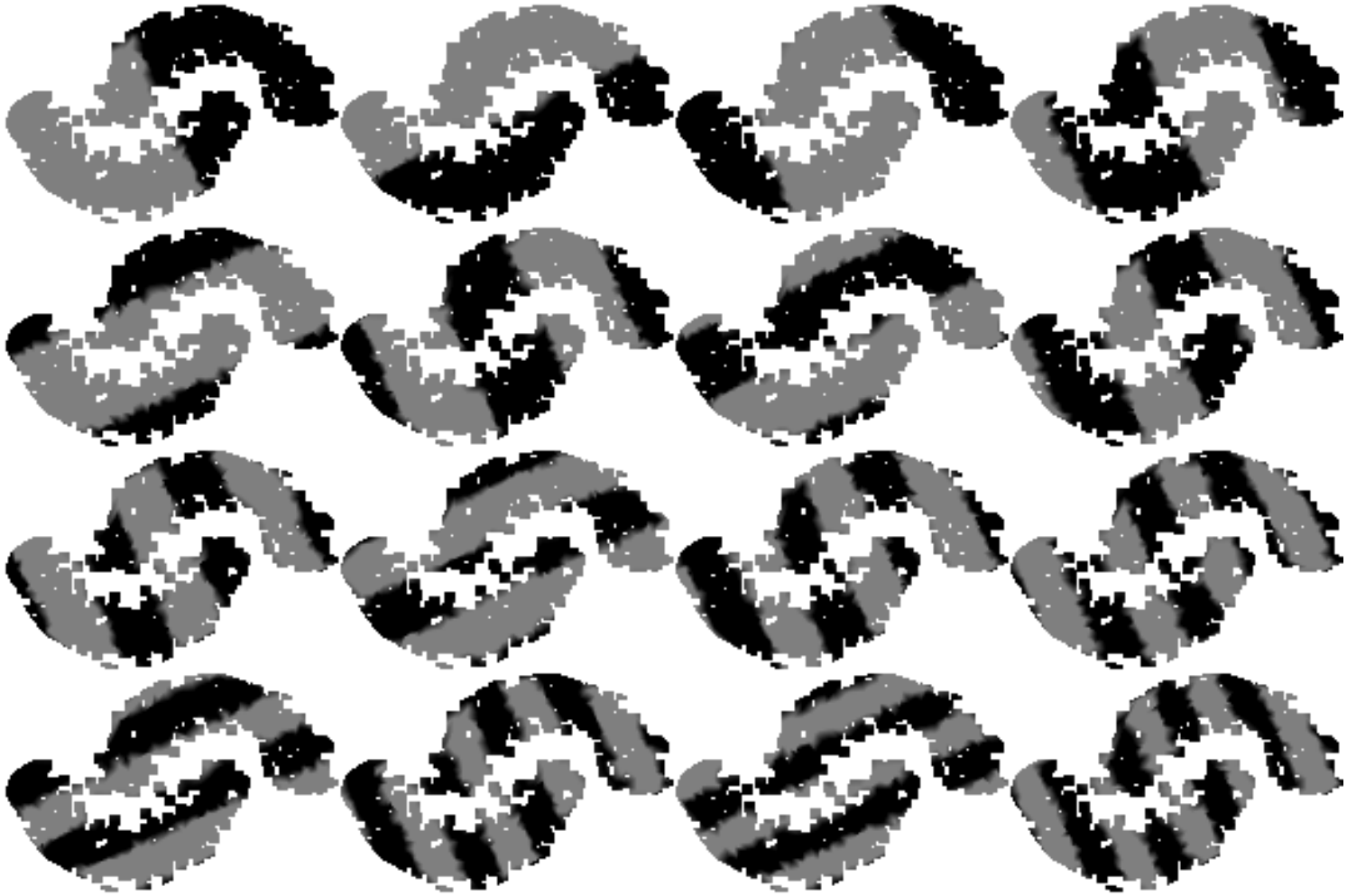


Figure: The 16-bit hash function for the two-moon dataset using SpH.

Extension I: Kernelisation

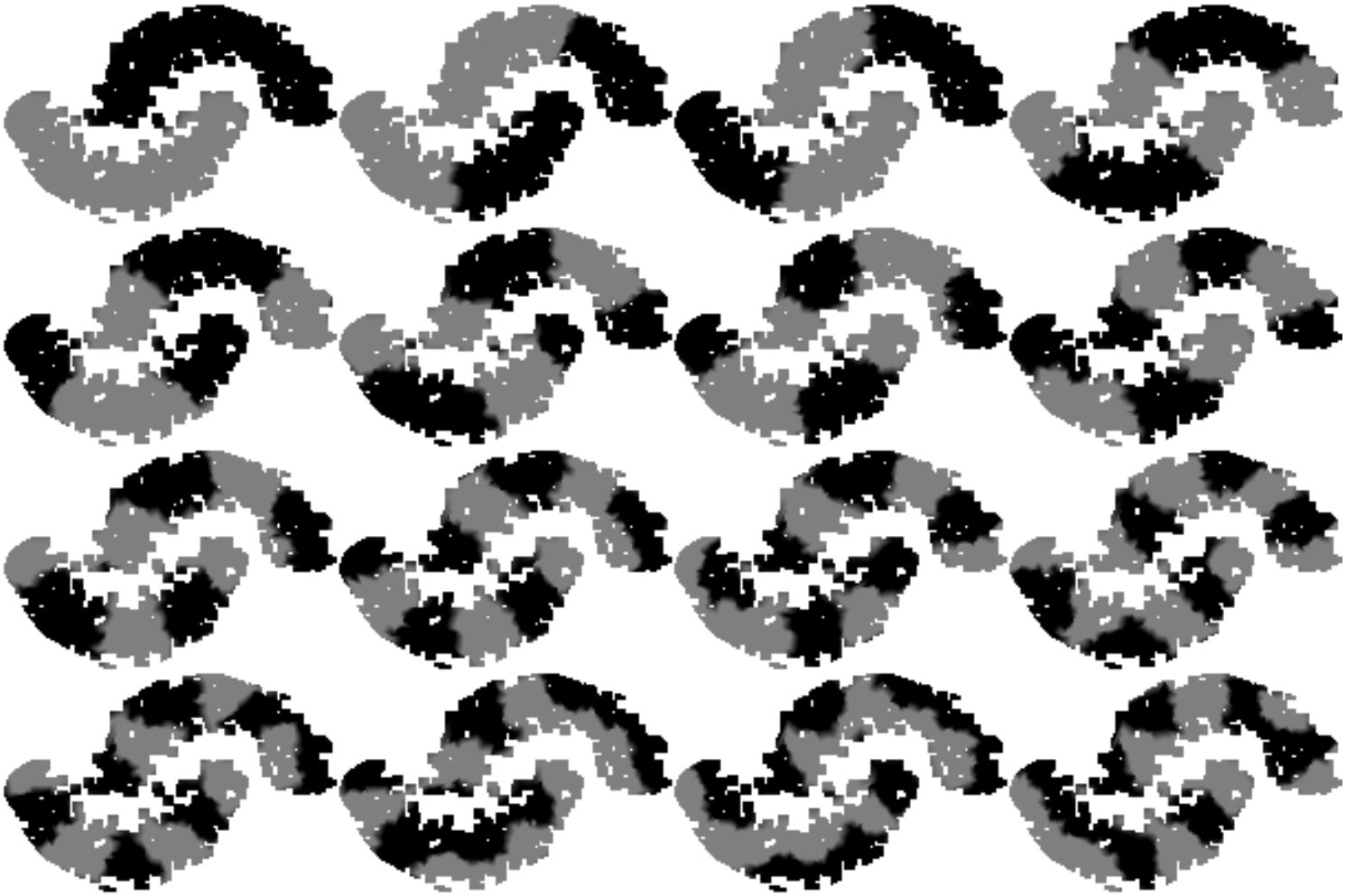


Figure: The 16-bit hash function for the two-moon dataset using STH.

Extension II: Supervision

In the first stage of STH, we make use of the class label information in the construction of k-nearest-neighbour graph for LapEig: a training document $\mathbf{x}$'s k-nearest-neighbourhood $N_k(\mathbf{x})$ would only contain k documents in the same class as $\mathbf{x}$ that are most similar to $\mathbf{x}$.

Let **STHs** denote such a supervised version of STH to distinguish it from the standard unsupervised version of STH.

Extension II: Supervision

Why not use SVMs directly?

kNN still has its advantages over SVMs in some aspects.

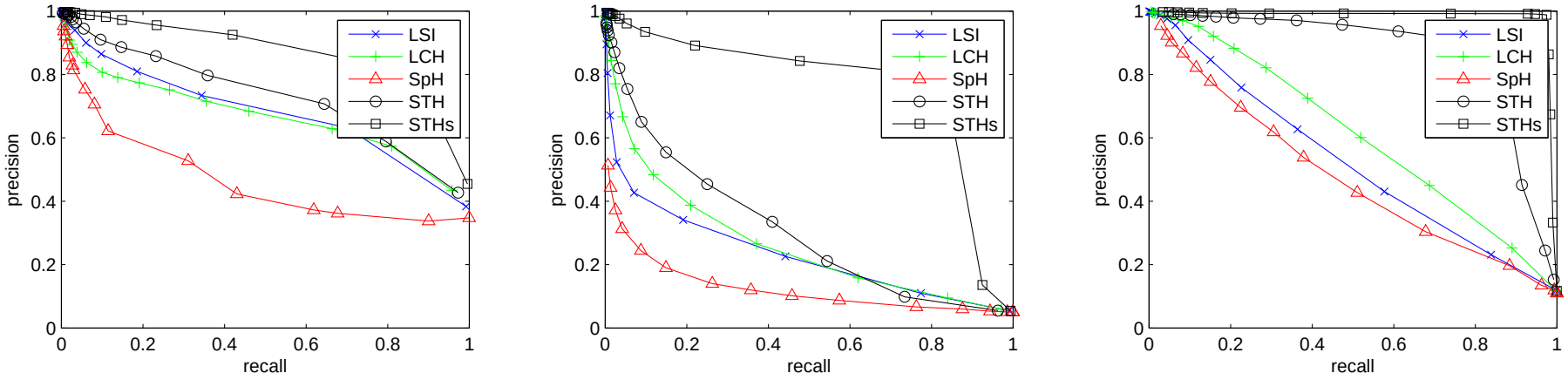
- For example, if there are 1000 classes,
  - the multi-class SVM approach may need 1000 binary SVM classifiers using the one-vs-rest ensemble scheme
  - the kNN (on top of STH) approach using 16-bit binary codes would only require 16 binary SVM classifiers

Extension II: Supervision

Text Datasets

- Reuters21578
  - Top 10 categories
  - 7285 documents
  - ModeApt split: 5228 (75%) training, 2057 (28%) testing
- 20Newsgroups
  - All 20 categories
  - 18846 documents
  - 'bydate' split: 11314 (60%) training, 7532 (40%) testing
- TDT2 (NIST Topic Detection and Tracking)
  - Top 30 categories
  - 9394 documents
  - random split (x10): 5597 (60%) training, 3797 (40%) testing

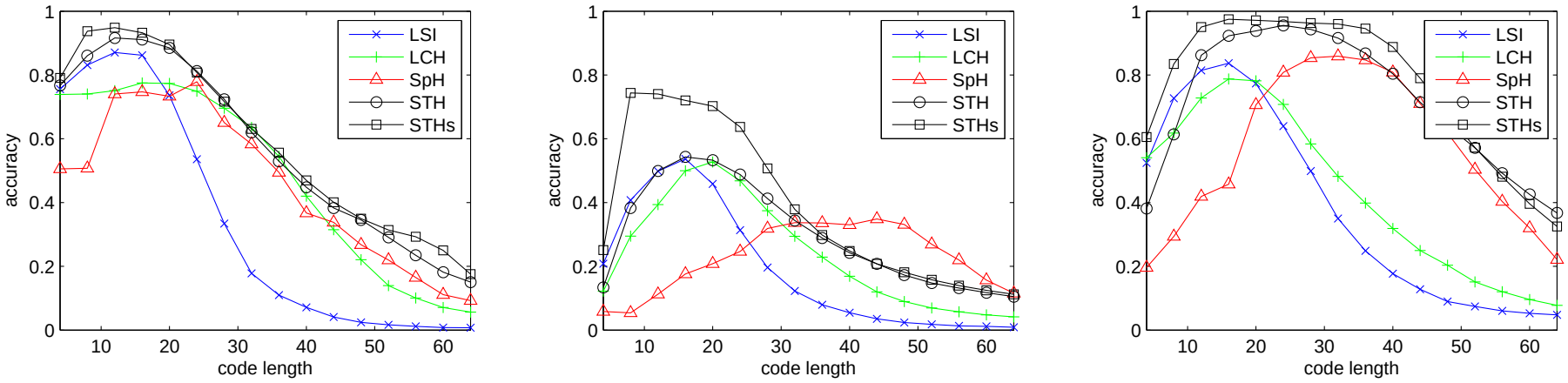
Extension II: Supervision



(a) Reuters21578 (b) 20Newsgroups (c) TDT2

Figure: The precision-recall curve for retrieving same-topic documents.

Extension II: Supervision



(a) Reuters21578 (b) 20Newsgroups (c) TDT2

Figure: The accuracy of approximate kNN classification (via hashing).

Outline

1. Problem
2. Related Work
3. Review of STH
4. Extensions to STH
5. Conclusion

Conclusion

- Major Contribution: Self-Taught Hashing
  - Unsupervised Learning + Supervised Learning
  - Spectral Method + Kernel Method
- Extensions (in the FGSIR Workshop on 23 Jul 2010)
  - Kernelisation
  - Supervision
- Future Work
  - Implementation using MapReduce
  - Applications in Multimedia IR

Question Time

Thanks!

8-)