Linear Spectral Hashing

Zalán Bodó and Lehel Csató

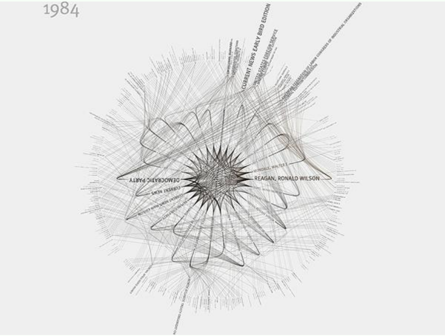
Faculty of Mathematics and Computer Science,
Babeș–Bolyai University, Cluj-Napoca/Kolozsvár

ESANN, Bruges, Belgium, April 2013

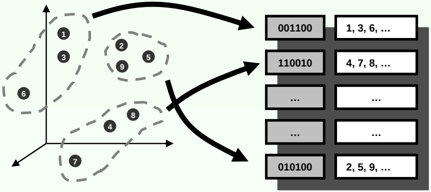
Contents

Introduction

- searching for similar items requires to go through the entire database
- need for a method to examine only a subset of the database, the *closest* items
- binary hash codes offer a solution to this problem
- sub-linear time retrieval w.r.t. the database size



Jer Thorp: This Was 1984



- idea: put the points into a hash table, such that **nearby** points lie in buckets with the *same* or *nearby* hash keys
- hash key/keyword/codeword = binary sequence
- *easy* to compute **Hamming** distance
- if $\boldsymbol{u}, \boldsymbol{v} \in \{0,1\}^n$, then

$$d_{\mathrm{H}}(\boldsymbol{u}, \boldsymbol{v}) = \sum_{i=1}^{n} (\boldsymbol{u} \,\mathrm{XOR}\, \boldsymbol{v})$$

ALG LSH

Generate k normally distributed ($\mathcal{N}(0,1)$) d-dimensional vectors: $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_k\}$

The k hash functions will be

$$h_i(\boldsymbol{u}) = \begin{cases} 1, & \boldsymbol{r}_i'\boldsymbol{u} \geq 0 \\ 0, & \boldsymbol{r}_i'\boldsymbol{u} < 0 \end{cases}, \quad i = 1, 2, \ldots, k$$

The binary keyword assigned to $\boldsymbol{x}$ is

$$[h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_k(\boldsymbol{x})]'$$

- $\boldsymbol{u}, \boldsymbol{v}$ – vectors in $\mathbb{R}^d$
- $\boldsymbol{r}$ – random $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ vector

$$P(h^{\boldsymbol{r}}(u) = h^{\boldsymbol{r}}(v)) = 1 - \frac{\theta(\boldsymbol{u}, \boldsymbol{v})}{\pi}$$

Locality-sensitive hashing

ALG LSH

1: Generate k normally distributed ($\mathcal{N}(0,1)$) d-dimensional vectors: $\{\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_k\}$

2: The k hash functions will be

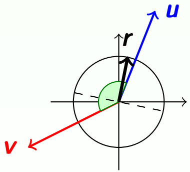
$$h_i(\boldsymbol{u}) = \begin{cases} 1, & \boldsymbol{r}_i'\boldsymbol{u} \geq 0 \\ 0, & \boldsymbol{r}_i'\boldsymbol{u} < 0 \end{cases}, \quad i = 1, 2, \ldots, k$$

3: The binary keyword assigned to $\boldsymbol{x}$ is

$$[h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_k(\boldsymbol{x})]'$$

- $\boldsymbol{u}, \boldsymbol{v}$ – vectors in $\mathbb{R}^d$
- $\boldsymbol{r}$ – random $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ vector

$$P(h^{\boldsymbol{r}}(u) = h^{\boldsymbol{r}}(v)) = 1 - \frac{\theta(\boldsymbol{u}, \boldsymbol{v})}{\pi}$$



Spectral hashing

- find $\boldsymbol{y}_i \in \{-1, 1\}^r$, $i = 1, \ldots, N$ codewords, such that for nearby points get nearby codewords:

$$\begin{aligned} \boldsymbol{Y}^* &= \operatorname*{argmin}_{\boldsymbol{Y} \in \mathbb{B}^{N \times r}} \sum_{i,j=1}^{N} W_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2 \\ \text{s.t.} & \quad \sum_{i=1}^{N} \boldsymbol{y}_i = \boldsymbol{0}, \quad \frac{1}{N} \sum_{i=1}^{N} y_i y_i' = \boldsymbol{I} \end{aligned}$$

  - 1. condition: bits to be balanced
  - 2. condition: bits to be uncorrelated

- relaxation:

$$\boldsymbol{Y}^* = \operatorname*{argmin}_{\boldsymbol{Y} \in \mathbb{R}^{N \times r}} \operatorname{tr}(\boldsymbol{Y}' \boldsymbol{L} \boldsymbol{Y}) \qquad \text{s.t.} \quad \boldsymbol{Y}'\boldsymbol{1} = \boldsymbol{0}, \quad \boldsymbol{Y}'\boldsymbol{Y} = \boldsymbol{I}$$

Spectral hashing

- find $\boldsymbol{y}_i \in \{-1, 1\}^r$, $i = 1, \ldots, N$ codewords, such that for nearby points get nearby codewords:

$$\boldsymbol{Y}^* = \operatorname*{argmin}_{\boldsymbol{Y} \in \mathbb{B}^{N \times r}} \sum_{i,j=1}^{N} W_{ij} \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \boldsymbol{y}_i = \boldsymbol{0}, \quad \frac{1}{N} \sum_{i=1}^{N} y_i y_i' = \boldsymbol{I}$$

  - 1. condition: bits to be balanced
  - 2. condition: bits to be uncorrelated
- relaxation:

$$\boldsymbol{Y}^* = \operatorname*{argmin}_{\boldsymbol{Y} \in \mathbb{R}^{N \times r}} \operatorname{tr}(\boldsymbol{Y}'\boldsymbol{L}\boldsymbol{Y}) \quad \text{s.t.} \quad \boldsymbol{Y}'\boldsymbol{1} = \boldsymbol{0}, \quad \boldsymbol{Y}'\boldsymbol{Y} = \boldsymbol{I}$$

- solution is given by the first r eigenvectors of $\boldsymbol{L}$, $\boldsymbol{Y}^* = [\boldsymbol{v}_2,\ \boldsymbol{v}_3,\ \ldots,\ \boldsymbol{v}_{r+1}]$ with nonzero eigenvalues
- **generalization**: using the *Gaussian* kernel – eigenfunctions of the weighted Laplace–Beltrami operators, assuming a multidimensional uniform distribution

Spectral clustering and max-margin hyperplanes

- normalized spectral clustering (relaxed):

$$\mathbf{z}^* = \operatorname*{argmax}_{\mathbf{z} \in \mathbb{R}^N} \frac{\mathbf{z}' \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2} \mathbf{z}}{\mathbf{z}'\mathbf{z}} \qquad \text{s.t.} \quad \mathbf{z}' \mathbf{D}^{1/2} \mathbf{1} = 0$$

- equivalent **inductive** problem, using a max-margin separating hyperplane with normal $\mathbf{w}$:

$$\mathbf{w}^* = \operatorname*{argmax}_{\mathbf{w} \in \mathcal{H}} \frac{\mathbf{w}' \Phi \mathbf{D}^{-1} \Phi' \mathbf{w}}{\mathbf{w}'\mathbf{w}} \qquad \text{s.t.} \quad \sum_{i=1}^{N} \mathbf{w}' \phi(\mathbf{x}_i) = 0$$

  - $\phi : X \to \mathcal{H}$, $\Phi = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)]$
  - $\mathbf{W} = \Phi'\Phi$
  - if $\mathbf{v}_2$ is the second eigenvector of $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, then $\mathbf{u}_2 = \Phi \mathbf{D}^{-1/2} \mathbf{v}_2 e_2^{-1/2}$ is the second eigenvector of $\Phi \mathbf{D}^{-1} \Phi'$
  - $f_2(\mathbf{x}) = \phi(\mathbf{x})'\mathbf{u}_2 = \phi(\mathbf{x})' \Phi \mathbf{D}^{-1/2} \mathbf{v}_2 e_2^{-1/2}$
  - assigns the same labels:
    $\operatorname{sgn}(\Phi'\mathbf{w}^*) = \operatorname{sgn}\left(\Phi'\Phi\mathbf{D}^{-1/2}\mathbf{v}_2 e_2^{-1/2}\right) =$
    $\operatorname{sgn}\left(\mathbf{D}^{-1/2}\Phi'\Phi\mathbf{D}^{-1/2}\mathbf{v}_2 e_2^{-1/2}\right) = \operatorname{sgn}\left(\mathbf{v}_2 e_2^{1/2}\right)$

Spectral clustering and max-margin hyperplanes

- normalized spectral clustering (relaxed):

$$\mathbf{z}^* = \operatorname*{argmax}_{\mathbf{z}\in\mathbb{R}^N} \frac{\mathbf{z}'\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{z}}{\mathbf{z}'\mathbf{z}} \qquad \text{s.t.} \quad \mathbf{z}'\mathbf{D}^{1/2}\mathbf{1} = 0$$

- equivalent **inductive** problem, using a max-margin separating hyperplane with normal $\mathbf{w}$:

$$\mathbf{w}^* = \operatorname*{argmax}_{\mathbf{w}\in\mathcal{H}} \frac{\mathbf{w}'\Phi\mathbf{D}^{-1}\Phi'\mathbf{w}}{\mathbf{w}'\mathbf{w}} \qquad \text{s.t.} \quad \sum_{i=1}^{N}\mathbf{w}'\phi(\mathbf{x}_i) = 0$$

  - $\phi : X \to \mathcal{H}$, $\Phi = [\phi(\mathbf{x}_1),\ \phi(\mathbf{x}_2),\ \ldots,\ \phi(\mathbf{x}_N)]$
  - $\mathbf{W} = \Phi'\Phi$
  - if $\mathbf{v}_2$ is the second eigenvector of $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$, then $\mathbf{u}_2 = \Phi\mathbf{D}^{-1/2}\mathbf{v}_2 e_2^{-1/2}$ is the second eigenvector of $\Phi\mathbf{D}^{-1}\Phi'$
  - $f_2(\mathbf{x}) = \phi(\mathbf{x})'\mathbf{u}_2 = \phi(\mathbf{x})'\Phi\mathbf{D}^{-1/2}\mathbf{v}_2 e_2^{-1/2}$
  - assigns the same labels:

$$\operatorname{sgn}(\Phi'\mathbf{w}^*) = \operatorname{sgn}\left(\Phi'\Phi\mathbf{D}^{-1/2}\mathbf{v}_2 e_2^{-1/2}\right) =$$

$$\operatorname{sgn}\left(\mathbf{D}^{-1/2}\Phi'\Phi\mathbf{D}^{-1/2}\mathbf{v}_2 e_2^{-1/2}\right) = \operatorname{sgn}\left(\mathbf{v}_2 e_2^{1/2}\right)$$

Spectral clustering and max-margin hyperplanes

- normalized spectral clustering (relaxed):

$$\boldsymbol{z}^* = \underset{\boldsymbol{z}\in\mathbb{R}^N}{\operatorname{argmax}} \frac{\boldsymbol{z}'\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}\boldsymbol{z}}{\boldsymbol{z}'\boldsymbol{z}} \qquad \text{s.t.} \quad \boldsymbol{z}'\boldsymbol{D}^{1/2}\mathbf{1} = 0$$

- equivalent **inductive** problem, using a max-margin separating hyperplane with normal $\boldsymbol{w}$:

$$\boldsymbol{w}^* = \underset{\boldsymbol{w}\in\mathcal{H}}{\operatorname{argmax}} \frac{\boldsymbol{w}'\boldsymbol{\Phi}\boldsymbol{D}^{-1}\boldsymbol{\Phi}'\boldsymbol{w}}{\boldsymbol{w}'\boldsymbol{w}} \qquad \text{s.t.} \quad \sum_{i=1}^{N} \boldsymbol{w}'\phi(\boldsymbol{x}_i) = 0$$

  - $\phi : X \to \mathcal{H}$, $\boldsymbol{\Phi} = [\phi(\boldsymbol{x}_1), \phi(\boldsymbol{x}_2), \ldots, \phi(\boldsymbol{x}_N)]$
  - $\boldsymbol{W} = \boldsymbol{\Phi}'\boldsymbol{\Phi}$
  - if $\boldsymbol{v}_2$ is the second eigenvector of $\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}$, then $\boldsymbol{u}_2 = \boldsymbol{\Phi}\boldsymbol{D}^{-1/2}\boldsymbol{v}_2 e_2^{-1/2}$ is the second eigenvector of $\boldsymbol{\Phi}\boldsymbol{D}^{-1}\boldsymbol{\Phi}'$
  - $f_2(\boldsymbol{x}) = \phi(\boldsymbol{x})'\boldsymbol{u}_2 = \phi(\boldsymbol{x})'\boldsymbol{\Phi}\boldsymbol{D}^{-1/2}\boldsymbol{v}_2 e_2^{-1/2}$
  - assigns the same labels:
    $\operatorname{sgn}(\boldsymbol{\Phi}'\boldsymbol{w}^*) = \operatorname{sgn}\left(\boldsymbol{\Phi}'\boldsymbol{\Phi}\boldsymbol{D}^{-1/2}\boldsymbol{v}_2 e_2^{-1/2}\right) =$
    $\operatorname{sgn}\left(\boldsymbol{D}^{-1/2}\boldsymbol{\Phi}'\boldsymbol{\Phi}\boldsymbol{D}^{-1/2}\boldsymbol{v}_2 e_2^{-1/2}\right) = \operatorname{sgn}\left(\boldsymbol{v}_2 e_2^{1/2}\right)$

Linear spectral hashing

- problem: generalization
- idea: use the spectral clustering results
- instead of $\boldsymbol{L}$ use the normalized Laplacian $\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$
- using the results from spectral clustering:

$$f(\boldsymbol{x}) = [f_2(\boldsymbol{x}),\ f_3(\boldsymbol{x}),\ \ldots,\ f_{r+1}(\boldsymbol{x})]'$$

- problem: computing $\phi(\boldsymbol{x})'\Phi$ requires N kernel computations
- *solution*: use dot products in the *input space*
- training: compute the r vectors $\boldsymbol{g}_k := \boldsymbol{u}_k$ or $\boldsymbol{g}_k := \boldsymbol{X}\boldsymbol{D}^{-1/2}\boldsymbol{v}_k$, $k = 2, \ldots, r+1$
- **prediction** (codeword for new point):

$$f(\boldsymbol{x}) = \left[\boldsymbol{x}'\boldsymbol{g}_2, \boldsymbol{x}'\boldsymbol{g}_3,\ \ldots,\ \boldsymbol{x}'\boldsymbol{g}_{r+1}\right]'$$

Linear spectral hashing

- problem: generalization
- idea: use the spectral clustering results
- instead of $\boldsymbol{L}$ use the normalized Laplacian $\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$
- using the results from spectral clustering:

$$f(\boldsymbol{x}) = [f_2(\boldsymbol{x}),\ f_3(\boldsymbol{x}),\ \ldots,\ f_{r+1}(\boldsymbol{x})]'$$

- problem: computing $\phi(\boldsymbol{x})'\Phi$ requires N kernel computations
- *solution*: use dot products in the *input space*
- training: compute the r vectors $\boldsymbol{g}_k := \boldsymbol{u}_k$ or $\boldsymbol{g}_k := \boldsymbol{X}\boldsymbol{D}^{-1/2}\boldsymbol{v}_k$, $k = 2, \ldots, r+1$
- **prediction** (codeword for new point):

$$f(\boldsymbol{x}) = \left[\boldsymbol{x}'\boldsymbol{g}_2, \boldsymbol{x}'\boldsymbol{g}_3,\ \ldots,\ \boldsymbol{x}'\boldsymbol{g}_{r+1}\right]'$$

Linear spectral hashing

- problem: generalization
- idea: use the spectral clustering results
- instead of $\boldsymbol{L}$ use the normalized Laplacian $\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$
- using the results from spectral clustering:

$$f(\boldsymbol{x}) = [f_2(\boldsymbol{x}),\ f_3(\boldsymbol{x}),\ \ldots,\ f_{r+1}(\boldsymbol{x})]'$$

- problem: computing $\phi(\boldsymbol{x})'\boldsymbol{\Phi}$ requires N kernel computations
- *solution*: use dot products in the *input space*
- training: compute the r vectors $\boldsymbol{g}_k := \boldsymbol{u}_k$ or $\boldsymbol{g}_k := \boldsymbol{X}\boldsymbol{D}^{-1/2}\boldsymbol{v}_k$, $k = 2, \ldots, r+1$
- **prediction** (codeword for new point):

$$f(\boldsymbol{x}) = \left[\boldsymbol{x}'\boldsymbol{g}_2, \boldsymbol{x}'\boldsymbol{g}_3,\ \ldots,\ \boldsymbol{x}'\boldsymbol{g}_{r+1}\right]'$$

Linear spectral hashing

- problem: generalization
- idea: use the spectral clustering results
- instead of $\boldsymbol{L}$ use the normalized Laplacian $\boldsymbol{D}^{-1/2}\boldsymbol{L}\boldsymbol{D}^{-1/2}$
- using the results from spectral clustering:

$$f(\boldsymbol{x}) = [f_2(\boldsymbol{x}),\ f_3(\boldsymbol{x}),\ \ldots,\ f_{r+1}(\boldsymbol{x})]'$$

- problem: computing $\phi(\boldsymbol{x})'\boldsymbol{\Phi}$ requires N kernel computations
- *solution*: use dot products in the *input space*
- training: compute the r vectors $\boldsymbol{g}_k := \boldsymbol{u}_k$ or $\boldsymbol{g}_k := \boldsymbol{X}\boldsymbol{D}^{-1/2}\boldsymbol{v}_k$, $k = 2, \ldots, r+1$
- **prediction** (codeword for new point):

$$f(\boldsymbol{x}) = \left[\boldsymbol{x}'\boldsymbol{g}_2, \boldsymbol{x}'\boldsymbol{g}_3,\ \ldots,\ \boldsymbol{x}'\boldsymbol{g}_{r+1}\right]'$$

Experiments

- experiments on Reuters-21578[1] and 20Newsgroups[2] corpora
- processing the documents:
  - removing stopwords (199)[3]
  - document frequency thresholding: using the 5000 most frequent words as dimensions
  - **bag-of-words** representation + **tf–idf** weighting
  - normalization to unit length
- labels were not used
- for every document of the test set we searched for the 50 nearest in the training set
- we measured to which extent the neighbors were found using **precision** and **recall** (*area under the recall–precision curve, precision for Hamming distance* < 2)

[1] http://disi.unitn.it/moschitti/corpora.htm
[2] http://qwone.com/~jason/20Newsgroups/20news-bydate.tar.gz
[3] WordNet::Similarity 2.05, http://wn-similarity.sourceforge.net/

Experiments

- experiments on Reuters-21578[1] and 20Newsgroups[2] corpora
- processing the documents:
  - removing stopwords (199)[3]
  - document frequency thresholding: using the 5000 most frequent words as dimensions
  - **bag-of-words** representation + **tf–idf** weighting
  - normalization to unit length
- labels were not used
- for every document of the test set we searched for the 50 nearest in the training set
- we measured to which extent the neighbors were found using **precision** and **recall** (*area under the recall–precision curve, precision for Hamming distance* < 2)

[1] http://disi.unitn.it/moschitti/corpora.htm
[2] http://qwone.com/~jason/20Newsgroups/20news-bydate.tar.gz
[3] WordNet::Similarity 2.05, http://wn-similarity.sourceforge.net/

Experiments

- experiments on Reuters-21578[1] and 20Newsgroups[2] corpora
- processing the documents:
  - removing stopwords (199)[3]
  - document frequency thresholding: using the 5000 most frequent words as dimensions
  - **bag-of-words** representation + **tf–idf** weighting
  - normalization to unit length
- labels were not used
- for every document of the test set we searched for the 50 nearest in the training set
- we measured to which extent the neighbors were found using **precision** and **recall** (*area under the recall–precision curve, precision for Hamming distance* < 2)

[1] http://disi.unitn.it/moschitti/corpora.htm
[2] http://qwone.com/~jason/20Newsgroups/20news-bydate.tar.gz
[3] WordNet::Similarity 2.05, http://wn-similarity.sourceforge.net/

Experiments

- experiments on Reuters-21578[1] and 20Newsgroups[2] corpora
- processing the documents:
  - removing stopwords (199)[3]
  - document frequency thresholding: using the 5000 most frequent words as dimensions
  - **bag-of-words** representation + **tf–idf** weighting
  - normalization to unit length
- labels were not used
- for every document of the test set we searched for the 50 nearest in the training set
- we measured to which extent the neighbors were found using **precision** and **recall** (*area under the recall–precision curve, precision for Hamming distance* < 2)

[1] http://disi.unitn.it/moschitti/corpora.htm
[2] http://qwone.com/~jason/20Newsgroups/20news-bydate.tar.gz
[3] WordNet::Similarity 2.05, http://wn-similarity.sourceforge.net/

Experiments

- experiments on Reuters-21578[1] and 20Newsgroups[2] corpora
- processing the documents:
  - removing stopwords (199)[3]
  - document frequency thresholding: using the 5000 most frequent words as dimensions
  - **bag-of-words** representation + **tf–idf** weighting
  - normalization to unit length
- labels were not used
- for every document of the test set we searched for the 50 nearest in the training set
- we measured to which extent the neighbors were found using **precision** and **recall** (*area under the recall–precision curve*, *precision for Hamming distance* < 2)

[1] http://disi.unitn.it/moschitti/corpora.htm
[2] http://qwone.com/~jason/20Newsgroups/20news-bydate.tar.gz
[3] WordNet::Similarity 2.05, http://wn-similarity.sourceforge.net/

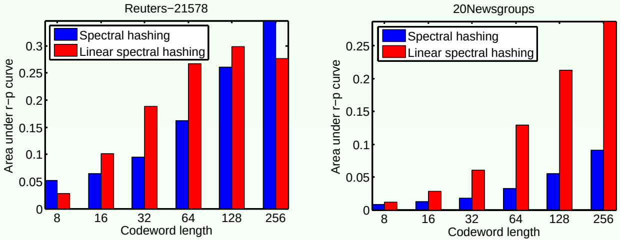


Figure: Area under the recall–precision curve as a function of the codeword length.

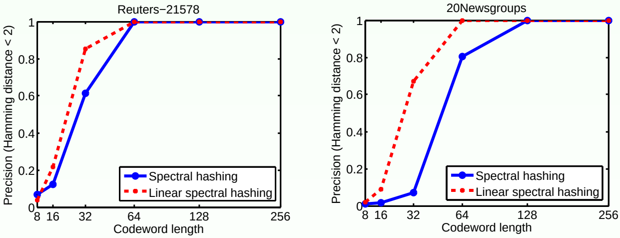


Figure: Precision results for Hamming distance < 2.

Thank you!

Questions?

The authors acknowledge the support of the Romanian Ministry of Education and Research via grant PN-II-RU-TE-2011-3-0278.