ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Linear Spectral Hashing

Zalán Bodó and Lehel Csató [*]

Babeş–Bolyai University - Faculty of Mathematics and Computer Science
Kogălniceanu 1., 400084 Cluj-Napoca - Romania

Abstract. Spectral hashing assigns binary hash keys to data points. This is accomplished via thresholding the eigenvectors of the graph Laplacian and obtaining binary codewords. While calculation for inputs in the training set is straightforward, an intriguing and difficult problem is how to compute the hash codewords for unseen data. A second problem we address is the computational difficulties when using the Gaussian similarity measure in spectral hashing: for specific problems – mainly the processing of large text databases – we propose linear scalar products as similarity measures and analyze the performance of the algorithm. We implement the linear algorithm and provide an inductive – generative – formula that leads to a prediction method similar to *locality-sensitive hashing* for a new data point. Experiments on document retrieval show promising results.

1 Introduction

In recent years several algorithms were proposed for fast approximate nearest neighbor search, providing sub-linear search times for a query. One of the most popular such methods is locality-sensitive hashing [2] (LSH), that partitions the dataset into clusters where similar data points are grouped together, and the grouping is made *without* comparing data points. Partitioning is accomplished by generating random hyperplanes, i.e. random vectors in the input or in a feature space [3]. The hash function is the sign of the dot product between the data point and the random vectors. Using carefully chosen random vectors, similar data point will likely be in *nearby* buckets, where proximity is defined via Hamming distance.

One can label a hashing algorithm as unsupervised, semi-supervised or supervised approach [1]. Whilst *supervised methods* are the most promising, for large datasets these are unfeasible as the labeling by humans would be too costly. Therefore we use unsupervised methods, namely the spectral hashing algorithm [9] that is able to *learn* relatively short codewords. In this paper we build a linear algorithm derived from spectral hashing with simple, LSH-like prediction and validate the results for a document retrieval problem.

The structure of the paper is as follows: Section 2 presents the spectral clustering in general. It describes the presentation of normalized spectral clustering as a maximum-margin hyperplane separation. After a brief description of spectral hashing in Section 3, the maximum-margin formulation is used in Section 4 to derive a new algorithm for computing the codewords for unseen points. Section 5 describes the experiments with text corpora, and discusses the results.

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

2 Spectral clustering and max-margin hyperplanes

In *spectral clustering* [8] – one of the most successful clustering algorithms – partitioning is done using the minimum cut in the neighborhood graph, where minimum cut is defined as removing edges such that the graph is partitioned into two – or more – disconnected subgraphs with minimal sum of edges.

The sum of edges alone as objective function favors unbalanced and small clusters. To overcome this problem, normalized cut is used. Since labeling implies negative or positive units, $z_i \in \mathbb{B} = \{-1, +1\}$, the label assigning problem is combinatorial. To avoid the combinatorial explosion, the original problem is relaxed to the following optimization on a continuous domain [7, 5]:

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

where $\mathbf{z}$ is the vector of – continuous – labels, $\mathbf{W}$ is a similarity matrix, and $\mathbf{D} = \operatorname{diag}(\mathbf{W1})$ is the diagonal degree matrix. Similarity can be defined using any positive definite kernel, and often it is defined using the Gaussian one: $W_{ij} = k_G(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{1}{2\sigma^2}\|\mathbf{x}_i - \mathbf{x}_j\|^2\right)$. Equation (1) is maximized by the eigenvector with the second largest (< 1) eigenvalue of the so-called normalized graph Laplacian $\mathbf{L}_{\text{sym}} = \mathbf{D}^{-1/2}(\mathbf{D} - \mathbf{W})\mathbf{D}^{-1/2}$. Having the solution, the cluster indicator is the thresholded vector of $\mathbf{z}^*$.

An equivalent view of the optimization from eq. (1) is that of separating the points by a maximum-margin hyperplane in the feature space defined by the same proximity matrix $\mathbf{W}$ [5], that we detail in the following.

Let us consider the matrix $\mathbf{\Phi} = [\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \ldots, \phi(\mathbf{x}_N)]$ as the projection of the training dataset in a feature space defined by the mapping $\phi : X \to \mathcal{H}$. We are looking for a *separating hyperplane* in that feature space, defined by the normal $\mathbf{w} \in \mathcal{H}$. To ensure that the clusters induced by the hyperplane are approximately the same size, the constraint $\sum_{i=1}^{N} \mathbf{w}'\phi(\mathbf{x}_i) = 0$ is introduced; and we also restrict $\|\mathbf{w}\| = 1$. Then we rewrite the optimization problem as

$$\mathbf{w}^* = \operatorname*{argmax}_{\mathbf{w}\in\mathcal{H}} \frac{\mathbf{w}'\mathbf{\Phi}\mathbf{D}^{-1}\mathbf{\Phi}'\mathbf{w}}{\mathbf{w}'\mathbf{w}} \qquad \text{s.t.} \quad \sum_{i=1}^{N} \mathbf{w}'\phi(\mathbf{x}_i) = 0 \tag{2}$$

where $\mathbf{w}'$ denotes the transpose of $\mathbf{w}$.

Similarly to spectral clustering, the solution of the above optimization is the eigenvector of the second largest eigenvalue of the matrix $\mathbf{\Phi}\mathbf{D}^{-1}\mathbf{\Phi}'$. As this is a feature space entity, its dimensionality is often infinite, therefore we aim at expressing the result using a smaller dimensional entity. We can use the spectral decomposition of the matrices $\mathbf{AA}' = \mathbf{D}^{-1/2}\mathbf{\Phi}'\mathbf{\Phi}\mathbf{D}^{-1/2} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ and $\mathbf{A}'\mathbf{A} = \mathbf{\Phi}\mathbf{D}^{-1}\mathbf{\Phi}'$ and establish that if $\mathbf{v}_2$ is the second eigenvector of $\mathbf{AA}'$, then $\mathbf{u}_2 = \mathbf{\Phi}\mathbf{D}^{-1/2}\mathbf{v}_2 e_2^{-1/2}$ is the second eigenvector of $\mathbf{A}'\mathbf{A}$, with e_2 the second largest eigenvalue.

The result is important: we obtain an *inductive clusterizer*, a hyperplane in the feature space $\mathcal{H}$ with normal $\mathbf{w}^* = \mathbf{u}_2 = \mathbf{\Phi}\mathbf{D}^{-1/2}\mathbf{v}_2 e_2^{-1/2}$. It is important the inductive nature of the algorithm: no eigenvector computations are needed, but the cluster label of the data point $\mathbf{x}$ is calculated by

$$f_2(\mathbf{x}) = \phi(\mathbf{x})'\mathbf{u}_2 = \phi(\mathbf{x})'\mathbf{\Phi}\mathbf{D}^{-1/2}\mathbf{v}_2 e_2^{-1/2} \tag{3}$$

where $f_2(\mathbf{x})$ refers to the fact that the scalar product with the second largest eigenvalue is computed. Furthermore, the separator from above assigns the same cluster labels to the points as the normalized spectral clustering, since

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

which is the same as $\operatorname{sgn}(\mathbf{v}_2)$, completing the proof.

Furthermore, the above formula can be applied to every eigenvector. Omitting the eigenvalue from the formula does not change our clusterizer, since $\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ is positive semi-definite. We use this result in spectral hashing for computing the hash codeword for an unseen point.

3 Spectral hashing

Spectral hashing [9] is a novel method for hashing-based nearest neighbor search. It is based on minimizing the weighted distances between binary codewords, and the weighting is based on the similarity between data points. As in LSH, this binary sequence clusters the points, but in contrast to the classical LSH output, the length of the sequence can be significantly smaller.

We want thus to find a mapping that assigns codewords to data points such that these codewords are close to each other for nearby data points, where we can consider the similarity matrix $\mathbf{W}$ from Section 2. If $\mathbf{y}_i$, $i = 1, \ldots, N$ denotes the codewords for the training points, and the vectors $\mathbf{y}_i'$ are organized into the rows of the matrix $\mathbf{Y}$, then the following optimization problem has to be solved:

$$\mathbf{Y}^* = \operatorname*{argmin}_{\mathbf{Y}\in\mathbb{B}^{N\times r}} \sum_{i,j=1}^{N} W_{ij}\|\mathbf{y}_i - \mathbf{y}_j\|^2 \qquad \text{s.t.} \quad \sum_{i=1}^{N}\mathbf{y}_i = \mathbf{0}, \quad \frac{1}{N}\sum_{i=1}^{N} y_i y_i' = \mathbf{I} \tag{4}$$

where the first condition is for maintaining the balance between the bits – bits are distributed evenly over $\{-1, 1\}$ – and the last condition makes the bits uncorrelated. Since the optimization problem above is NP-hard [9], analogously to spectral clustering, using the graph Laplacian $\mathbf{L}$, it is relaxed into

$$\mathbf{Y}^* = \operatorname*{argmin}_{\mathbf{Y}\in\mathbb{R}^{N\times r}} \operatorname{tr}(\mathbf{Y}'\mathbf{L}\mathbf{Y}) \qquad \text{s.t.} \quad \mathbf{Y}'\mathbf{1} = \mathbf{0}, \quad \mathbf{Y}'\mathbf{Y} = \mathbf{I} \tag{5}$$

Excluding the first constraint, the solution of the problem is given by the first r eigenvectors of $\mathbf{L}$, $\mathbf{Y}^* = [\mathbf{v}_1,\ \mathbf{v}_2, \ldots,\ \mathbf{v}_r]$ [4]. Thus $\mathbf{y}_i^{*\prime} = [v_{1i},\ v_{2i},\ \ldots,\ v_{ri}]$. The first constraint is satisfied by all the eigenvectors, except the first one, the constant one eigenvector with eigenvalue 0, because $\mathbf{Y}^{*\prime}\mathbf{1} = \mathbf{0}$ is equivalent to requiring the dot product of each eigenvector with the constant one vector to equal zero. Therefore, the final solution is given by $\mathbf{Y}^* = [\mathbf{v}_2,\ \mathbf{v}_3,\ \ldots,\ \mathbf{v}_{r+1}]$ where $\mathbf{v}_2$ denotes the eigenvector corresponding to the second smallest positive eigenvalue.

In [9], using the Gaussian kernel, generalization was done using the eigenfunctions of the weighted Laplace–Beltrami operators, assuming a multidimensional uniform distribution. In this paper we use the results from Section 2, providing a more simple and elegant approach for the linear case.

4 Linear spectral hashing

The crucial problem is to generalize spectral hashing for unseen points. That could be solved by including the point in the dataset and recalculating the first r eigenvectors. It is however obvious that this is not admissible, since turns the problem into a more complex one than the initial search was.

In this section we connect spectral hashing with spectral clustering and use the generalization results from Section 2. For this, in problem (5) we simply change the Laplacian to the normalized Laplacian $\mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, thus the solution is now given by the first r eigenvectors of the normalized Laplacian with strictly positive eigenvalues. Subsequently, we can use eq. (3) to compute the *cluster labels* of a point $\mathbf{x}$ by substituting the first r eigenvectors – starting from the second largest eigenvalue – into $\mathbf{u}_k$ or $\mathbf{v}_k$:

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

and we assumed that the new $\mathbf{x}$ comes from the same distribution as the training points, thus using (3) we approximate the assigned labels.

However, computing $\phi(\mathbf{x})'\mathbf{\Phi}$ requires N kernel computations, which is the same as comparing the new point $\mathbf{x}$ to every training example, as in the original nearest neighbor search. Thus, instead of simplification we actually increased the number of comparisons, i.e. similarity or distance computations. Nevertheless, if we use dot products, that is we handle the points in the *input space*, we can compute the r vectors $\mathbf{g}_k := \mathbf{u}_k$ or $\mathbf{g}_k := \mathbf{X}\mathbf{D}^{-1/2}\mathbf{v}_k,\ k = 2, \ldots, r+1$, and when a new point $\mathbf{x}$ arrives calculate the dot products $\mathbf{x}'\mathbf{g}_k$ and threshold the resulting values to obtain the binary codeword. Thus, the computation of a new codeword requires only r dot products of input space vectors. Whether to calculate $\mathbf{u}_k$ or $\mathbf{v}_k$ depends on the size and dimensionality of the dataset. For large, but relatively low-dimensional data – which we assume is the case – the eigendecomposition of $\mathbf{X}\mathbf{D}^{-1}\mathbf{X}'$ is performed. The codewords assigned to training points are composed by the first r eigenvectors with strictly positive eigenvalues of the normalized Laplacian, computed similarly.

Our method has the limitation that the similarity matrix $\mathbf{W} = \mathbf{X}\mathbf{X}'$ must be positive, but in turn it offers a simple and fast codeword generation process for unseen points.

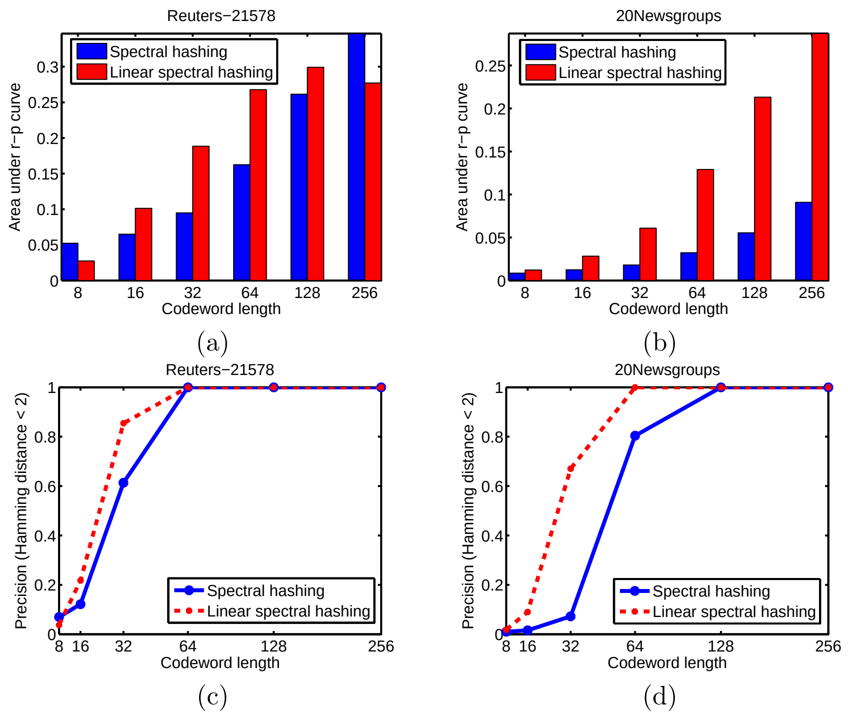

Fig. 1: (a) and (b) – area under the recall-precision curve as a function of the codeword length. (c) and (d) – precision results for Hamming distance < 2.

5 Experimental results and discussion

The experiments were run on two corpora frequently used for evaluating text categorization systems, since linear methods perform well on bag-of-words document vectors. The superiority of spectral hashing to other methods has already been shown [9], therefore we only compare our algorithm to spectral hashing.

In the experiments we used the Reuters-21578 and 20Newsgroups[1] datasets. For both corpora, a stopword list with 199 elements[2] was used to get rid of the probably irrelevant words, after which the remaining most frequent 5000 words were selected as the dimensions of the representational space. In the final step the *tf-idf* transformation [6] was applied and the documents were normalized to unit length. No stemming was applied for the selected features.

For evaluating the performance of the studied methods we did not use the labels of the documents, but for each test document we determined its 50 nearest neighbors from the training set, and measured to which extent the neighbors were

[1]The datasets can be downloaded from `http://disi.unitn.it/moschitti/corpora.htm` and `http://qwone.com/~jason/20Newsgroups/20news-bydate.tar.gz`.

[2]The stoplist was taken from WordNet::Similarity 2.05, `http://wn-similarity.sourceforge.net`.

found by the algorithms. The evaluation measures used were precision and recall [6].

In Figure 1.(a,b) the areas under recall-precision curves are shown for spectral hashing and linear spectral hashing for the Reuters-21578 and 20Newsgroups dataset by varying the codeword bits from 8 to 256, every time doubling the length.[3] Precision and recall was calculated as the function of the Hamming neighborhood, and the number of evaluations was determined by the length of the codeword. For example, for 64 bits the precisions and recalls were calculated for Hamming distances of $\{0, 1, 2, \ldots, 64\}$. Figure 1.(c,d) shows the results obtained with spectral hashing and linear spectral hashing respectively, where the precision was plotted for Hamming distance < 2, increasing the number of codeword bits from 8 to 256.

We presented a method based on spectral hashing but using a different generalization for unseen points. This method outputs r input space vectors, which are used to compute the binary hash sequence of an unseen point by calculating dot products as in LSH. The proposed method can be used for large and relatively low-dimensional datasets, where dot products offer a good similarity measure. Experiments on bag-of-words data show that the proposed method outperforms spectral hashing in the $[16 \mathinner{..} 128]$ interval. We consider this a promising result, since using hash sequence lengths above 128 may be impractical in many situations. Applying arbitrary kernels – for example by using low-dimensional approximations – remains the objective of a future research.

References

[1] Kristen Grauman and Rob Fergus. Learning binary hash codes for large-scale image search. *Machine Learning for Computer Vision*, 411:49–87, 2013.

[2] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.

[3] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137. IEEE, 2009.

[4] Helmut Lütkepohl. *Handbook of matrices.* John Wiley & Sons Ltd., Chichester, 1996.

[5] Ali Rahimi and Ben Recht. Clustering with normalized cuts is clustering with a hyperplane. In *Statistical Learning in Computer Vision*, 2004.

[6] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, March 2002.

[7] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. In *CVPR*, pages 731–737. IEEE Computer Society, 1997.

[8] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[9] Yair Weiss, Antonio B. Torralba, and Robert Fergus. Spectral hashing. In *NIPS*, pages 1753–1760. MIT Press, 2008.

[3]For spectral hashing we used the MATLAB code from `http://www.cs.huji.ac.il/~yweiss/SpectralHashing`.