Clustering Pairwise Distances with Missing Data: Maximum Cuts versus Normalized Cuts*

Jan Poland and Thomas Zeugmann

Division of Computer Science
Hokkaido University, Sapporo 060-0814, Japan
{jan,thomas}@ist.hokudai.ac.jp
http://www-alg.ist.hokudai.ac.jp/~{jan,thomas}

Abstract. Clustering algorithms based on a matrix of pairwise similarities (kernel matrix) for the data are widely known and used, a particularly popular class being spectral clustering algorithms. In contrast, algorithms working with the pairwise distance matrix have been studied rarely for clustering. This is surprising, as in many applications, distances are directly given, and computing similarities involves another step that is error-prone, since the kernel has to be chosen appropriately, albeit computationally cheap. This paper proposes a clustering algorithm based on the SDP relaxation of the max-k-cut of the graph of pairwise distances, based on the work of Frieze and Jerrum. We compare the algorithm with Yu and Shi's algorithm based on spectral relaxation of a norm-k-cut. Moreover, we propose a simple heuristic for dealing with missing data, i.e., the case where some of the pairwise distances or similarities are not known. We evaluate the algorithms on the task of clustering natural language terms with the Google distance, a semantic distance recently introduced by Cilibrasi and Vitányi, using relative frequency counts from WWW queries and based on the theory of Kolmogorov complexity.

1 Introduction

Let a set of n objects or data points, $x_1, \ldots, x_n$, be given. We might not know anything about the objects, but assume that their pairwise distances $d_{ij} = d(x_i, x_j)$ are known. Here, $d\colon M \times M \to \mathbb{R}$ is a distance measure over a set M, i.e., $d(x, y) \geq 0$, $d(x, y) = d(y, x)$ for all $x, y \in M$, and $d(x, y) = 0$ iff $x = y$. Then we can *cluster* the data, i.e., assign the x_i to k distinct groups such that the distances within groups are small and the distances between the groups are large. This is done as follows. Construct a graph from the pairwise distances and choose an algorithm from the large class of recently published methods based on graph-theoretic cut criteria. As the cuts are usually $\mathcal{NP}$-hard to optimize, appropriate relaxations have been subject to intensive research. Two types of relaxations are particularly important:

* This work was supported by JSPS 21st century COE program C01. Additional support has been provided by the MEXT Grand-in-Aid for Scientific Research on Priority Areas under Grant No. 18049001

(1) *Spectral* methods, where the top eigenvectors of the graph's adjacency matrix are used to project the data into a lower dimensional space. This gives rise to new theoretical investigations of the popular spectral clustering algorithms.

(2) *Semi-definite programming* (*SDP*), where the discrete constraints of the cut criterion are replaced by continuous counterparts. Then convex solvers can be used for the optimization.

Surprisingly, all of the clustering approaches suggested so far work on a graph of *similarities* rather than distances. This means that, given the distances, we need one additional step to obtain similarities from distances, e.g., by applying a Gaussian kernel. This also involves tuning the kernel width, a quantity which the clustering algorithm is quite sensitive to. Hence, it is natural to avoid this step by using a cut criterion that directly works with the distance graph, e.g., max-cut. We follow Frieze and Jerrum [4] and solve the max-cut problem via an SDP relaxation. We compare this method with a representative of spectral clustering algorithms, namely the spectral relaxation of the normalized cut criterion [12].

As a second contribution of this paper, we propose a simple heuristic for dealing with *missing data*, i.e., the case where some of the pairwise distances d_{ij} are unknown. Then, our aim is to substitute the missing d_{ij} by a value which is most likely to leave the values of the cuts intact. This turns out to be the *mean* of the observed d_{ij}.

One motivation for considering missing data is given by the application we shall use to test the algorithms: Clustering of natural language terms using the Google distance. The Google distance [2] is a means of computing the pairwise distance of any searchable terms by just using the relative frequency count resulting from a web search. The Google API provides a convenient way for automating this process, however with a single key (which is obtained by prior registration) the maximum amount of daily queries is currently limited to 1000. Hence, by querying an incomplete sparse distance matrix rather than a full one, one can speed up considerably the overall process, as we shall demonstrate.

The paper is structured as follows. In the next two sections, we introduce the two algorithms based on max-k-cut and norm-k-cut relaxations, respectively, and recall some theory. In Section 4 we address the missing data problem. Section 5 confronts the two algorithms, looking on the exact cut criteria rather than the relaxations, and compares the computational resources required. Section 6 describes the Google distance. In Section 7 we present experimental results with the Google distance. Relation to other work is discussed and conclusions are given in Section 8.

2 Max-k-cut

Given a fully connected, weighted graph $G=(V,D)$ with vertices $V=\{x_1,\ldots,x_n\}$ and edge weights $D=\{d_{ij}\geq 0 \mid 1\leq i,j\leq n\}$ which express pairwise distances, a *k-way-cut* is a partition of V into k disjoint subsets $S_1,\ldots,S_k$. Here k is assumed to be given. We define the predicate $A(i,j)=0$ if x_i and x_j happen to be in the same subset, i.e., if $\exists\ell[1\leq\ell\leq k,\ 1\leq i,j\leq n$ and $i,j\in S_\ell]$, and

$A(i,j) = 1$, otherwise. The weight of the cut $(S_1, \ldots, S_k)$ is defined as

$$\sum_{i,j=1}^{n} d_{i,j} A(i,j) \ .$$

The *max-k-cut* problem is the task of finding the partition that maximizes the weight of the cut. It can be stated as follows: Let $a_1, \ldots, a_k \in \mathcal{S}^{k-2}$ be the vertices of a regular simplex, where

$$\mathcal{S}^d = \{x \in \mathbb{R}^{d+1} \mid \|x\|_2 = 1\}$$

is the d-dimensional unit sphere. Then the inner product $a_i \cdot a_j = -\frac{1}{k-1}$ whenever $i \neq j$. Hence, finding the max-k-cut is equivalent to solving the following integer program:

$$\begin{aligned} \text{IP}: \text{maximize } & \tfrac{k-1}{k} \sum_{i<j} d_{ij}(1 - y_i \cdot y_j) \\ \text{subject to } & y_j \in \{a_1, \ldots, a_k\} \text{ for all } 1 \leq j \leq n. \end{aligned}$$

Frieze and Jerrum [4] propose the following semidefinite program (SDP) in order to relax the integer program:

$$\begin{aligned} \text{SDP}: \text{maximize } & \tfrac{k-1}{k} \sum_{i<j} d_{ij}(1 - v_i \cdot v_j) \\ \text{subject to } & v_j \in \mathcal{S}^{n-1} \text{ for all } 1 \leq j \leq n \text{ and} \\ & v_i \cdot v_j \geq -\tfrac{1}{k-1} \text{ for all } i \neq j \text{ (necessary if } k \geq 3). \end{aligned}$$

The constraints $v_i \cdot v_j \geq -\frac{1}{k-1}$ are necessary for $k \geq 3$ because otherwise the SDP would prefer solutions where $v_i \cdot v_j = -1$, resulting in a larger value of the objective. We shall see in the experimental part that this indeed would result in invalid approximations. The SDP finally can be reformulated as a convex program:

$$\begin{aligned} \text{CP}: \text{minimize } & \sum_{i<j} d_{ij} Y_{ij} && \text{(1a)} \\ \text{subject to } & Y_{jj} = 1 \text{ for all } 1 \leq j \leq n \text{ and} && \text{(1b)} \\ & Y_{ij} \geq -\tfrac{1}{k-1} \text{ for all } i \neq j \text{ (necessary if } k \geq 3) \text{ and} && \text{(1c)} \\ & Y = (Y_{ij})_{1 \leq i,j \leq n} \text{ satisfies } Y \succeq 0. && \text{(1d)} \end{aligned}$$

Here, for the matrix $Y \in \mathbb{R}^{n \times n}$ the last condition $Y \succeq 0$ means that Y is positive semidefinite. Efficient solvers are available for this kind of optimization problems, such as CSDP [1] or SeDuMi [10]. In order to implement the constraints $Y_{ij} \geq -\frac{1}{k-1}$ with these solvers, positive slack variables Z_{ij} have to be introduced together with the equality constraints $Y_{ij} - Z_{ij} = -\frac{1}{k-1}$.

Finally, for obtaining the partitioning from the vectors v_j or the matrix Y, Frieze and Jerrum [4] propose to sample k points $z_1, \ldots, z_k$ randomly on $\mathcal{S}^{n-1}$,

representing the groups, and assign each v_j to the closest group, i.e., the closest z_j. They show approximation guarantees generalizing those of Goemans and Williamson [5]. In practice however, the approximation guarantee does not necessarily imply a good clustering, and applying the k-means algorithm for clustering the v_j gives better results here. We use the kernel k-means (probably introduced for the first time by [9]) which directly works on the scalar products $Y_{ij} = v_i \cdot v_j$, without need of recovering the v_j. We recapitulate the complete algorithm:

Algorithm. Clustering as an SDP relaxation of max-k-cut
Input: Distance matrix $D = (d_{ij})$.
1. Solve the SDP via the CP (1a) through (1d).
2. Cluster the resulting matrix Y using kernel k-means.

3 Normalized k-cut

The normalized cut criterion has emerged as one of the most widely accepted cut criteria for clustering. It is defined on a graph $G = (V, W)$ of pairwise similarities rather than distances: $W = \{w_{ij} \mid w_{ij} \in [0,1], 1 \le i, j \le n\}$. Here, we identify the edges of G with their weights given by the similarities. For a k-way-cut, i.e., a partition of V into k disjoint subsets $S_1, \ldots, S_k$, the norm-k-cut criterion is defined as (cf. Yu and Shi [12])

$$\frac{1}{k}\sum_{\ell=1}^{k}\frac{\sum_{i\in S_\ell, j\notin S_\ell} w_{ij}}{\sum_{i\in S_\ell, j\in V} w_{ij}} = 1 - \frac{1}{k}\sum_{\ell=1}^{k}\frac{\sum_{i\in S_\ell, j\in S_\ell} w_{ij}}{\sum_{i\in S_\ell, j\in V} w_{ij}} =: 1 - \text{knassoc}(S_1, \ldots, S_k), \quad (2)$$

where $\text{knassoc}(S_1, \ldots, S_k)$ is called the *k-way normalized associations criterion*. Therefore, minimizing the norm-k-cut value is equivalent to maximizing the norm-knassoc value. Following [12], this is the task we consider. For a vector $v = (v_1, \ldots, v_n)$ we write $\text{Diag}(v)$ to denote the matrix $M = (m_{ij})_{1\le i,j\le n}$ with $m_{ii} = v_i$ and $m_{ij} = 0$ for all $i \ne j$. Furthermore, for a matrix $M = (m_{ij})_{1\le i,j\le n}$ we write $\text{diag}(M)$ to denote the vector $(m_{11}, \ldots, m_{nn})$.

Optimizing (2) can be restated as solving the following integer program

$$\begin{aligned} \text{IP}: \text{maximize } & \tfrac{1}{k}\text{tr}(Z^T W Z) \\ \text{subject to } & Z = X(X^T \Sigma X)^{-\frac{1}{2}}, \\ & \Sigma = \text{Diag}\Big(\big(\sum\nolimits_{i=1}^{n} w_{ij}\big)_{1\le j\le n}\Big), \text{ and} \\ & X \in \{0,1\}^{n\times k} \text{ such that } \sum_{\ell=1}^{k} X(j,\ell) = 1 \text{ for all } 1 \le j \le n. \end{aligned} \quad (3)$$

Relaxing the constraints on Z and passing to continuous domain, we need to solve the following continuous program

$$\begin{aligned} \text{ContP}: \text{maximize } & \tfrac{1}{k}\text{tr}(Z^T W Z) \\ \text{subject to } & Z^T \Sigma Z = I_k, \end{aligned} \quad (4)$$

where I_k is the k-dimensional identity matrix and Σ is defined as above. The space of all optima of (4) can be described with the help of a spectral decomposition of the Laplacian $L = \Sigma^{-\frac{1}{2}} W \Sigma^{-\frac{1}{2}}$

$$L = \Sigma^{-\frac{1}{2}} W \Sigma^{-\frac{1}{2}} = U \Lambda U^T, \tag{5}$$

where U is orthogonal and Λ is diagonal. Let $\Lambda^* \in \mathbb{R}^{n \times k}$ denote the part of Λ containing the largest k eigenvalues (such that $\Lambda^*(i,j) = 0$ unless $i = j$), and $U^* \in \mathbb{R}^{n \times k}$ be the corresponding eigenvectors, then

$$\mathcal{Z} = \{\Sigma^{-\frac{1}{2}} U^* R \mid R^T R = I_k\} \tag{6}$$

describes the space of solutions of (4).

For some relaxed solution $Z \in \mathcal{Z}$, the corresponding $X \in \mathbb{R}^{n \times k}$ can be obtained by inverting (3):

$$\tilde{X} = \mathrm{Diag}\big(\mathrm{diag}(ZZ^t)\big) Z.$$

Then one can reconstruct an integer solution X by applying an EM procedure and alternatingly optimizing the rotation matrix R and the discretization X:

1. For given R and resulting Z and $\tilde{X}$, let $X(i,\ell) = 1$, if $\tilde{X}(i,\ell) \geq \tilde{X}(i,m)$ for all $1 \leq m \leq k$, and let $X(i,\ell) = 0$, otherwise.
2. For given X and $\tilde{X}$, compute a singular value decomposition $X^T \tilde{X} = \tilde{U} \tilde{\Lambda} \bar{U}^T$ and let $R = \tilde{U} \bar{U}^t$.

We recapitulate the algorithm:
Algorithm. Clustering as a spectral relaxation of norm-k-cut [12]
Input: Similarity matrix $W = (w_{ij})$.
1. Solve the eigenvalue problem (5).
2. Use the described EM procedure to optimize the discretization.

Note that the EM procedure used in this algorithm is different from, but nevertheless closely related to the k-means algorithm used in the algorithm based on max-k-cut and many other spectral clustering algorithms.

4 Missing data

Assume that either the distance matrix D or the similarity matrix W is not fully specified, but a portion of the off-diagonal entries is missing. One motivation for considering this case could be the desire to save resources by computing only part of the entries (e.g., for the Google distance discussed below, normal user registration permits only a limited amount of queries a day). Suppose that $M \in \{0,1\}^{n \times n}$ is a matrix such that $\mathrm{diag}(M) = 0$ and $M(i,j) = 1$ if and only if $D(i,j)$ (or $W(i,j)$, respectively) is not missing. Assume that the diagonal of D is zero and that of W is one, and denote the ith column of a matrix X by $X[i]$. Define the mean of the observed values,

$$\bar{D} = \frac{1}{\sum_{i,j} M(i,j)} \sum_{i=1}^{n} D[i]^T M[i] \quad \text{or} \quad \bar{W} = \frac{1}{\sum_{i,j} M(i,j)} \sum_{i=1}^{n} W[i]^T M[i].$$

Then, replacing the missing entries in D with the value $\bar{D}$, the resulting distance matrix D is an unbiased estimate for the original full matrix, if the positions of the missing values are sampled from a uniform distribution. Hence, the resulting max-k-cut criterion for each partition is an unbiased estimate for the criterion respective to the original matrix, and this is the best we can do to achieve our goal that the optimal k-way-cuts of the original and the completed matrix are the same.

Also in the case of a similarity matrix W, the missing values should be replaced by the mean of the observed values. Asymptotically for $n \to \infty$, this also yields an unbiased estimate for the norm-k-cut criterion. However, the reasoning is more difficult here, since the norm-k-cut criterion is a sum of quotients, and for two random variables X and Y, we have $E[X/Y] \neq E[X]/E[Y]$. Still, the actual values of numerator and denominator are close to their expectations, as one can verify using concentration inequalities, e.g., Hoeffding's inequality. Then, for large n, with high probability the quotient is close to the corresponding quantity for the original (full) similarity matrix.

5 Max-k-cut versus norm-k-cut

In this section, we compare the max-cut and norm-cut criteria on distance and similarity matrices that are small enough to allow for a brute-force computation of the exact criteria. We start from 10×10 matrices D_0 and W_0 consisting of two blocks of each size 5,

$$D_0 = \begin{pmatrix} 0 & \cdots & 0 & 1 & \cdots & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & \cdots & 1 \\ 1 & \cdots & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & 1 & 0 & \cdots & 0 \end{pmatrix} \quad \text{and} \quad W_0 = \begin{pmatrix} 1 & \cdots & 1 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 1 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & \cdots & 1 \end{pmatrix}.$$

From these matrices, distance matrices D and similarity matrices W are obtained by (1) perturbing the value by Gaussian noise of varying amplitude, (2) making the matrices symmetric and rescaling them to the interval $[0, 1]$, (3) removing a fraction of the off-diagonal values and replacing them by the mean of the remaining values. Another matrix we use for the norm-cut criterion is a kernel matrix obtained from the distance matrix using a Gaussian kernel, $W^D = \exp(-\frac{1}{2\sigma^2} D^2)$ (all operations are meant in the pointwise sense here). Since the values of the distance matrix are normalized to $[0, 1]$, we use a fixed $\sigma = \frac{1}{3}$. The missing values of W^D are replaced by the mean of the observed values in W^D.

All values displayed in Figures 1 through 3 below are means of 1500 independent samples. Figure 1 shows that, when using the max-cut criterion, the relative number of experiments that result in a different clustering than the originally intended one, grows if either the noise amplitude or the fraction of missing values increases. Of course this was expected. The max-cut criterion even yields always the correct clustering if both noise amplitude and missing data fraction are sufficiently low.

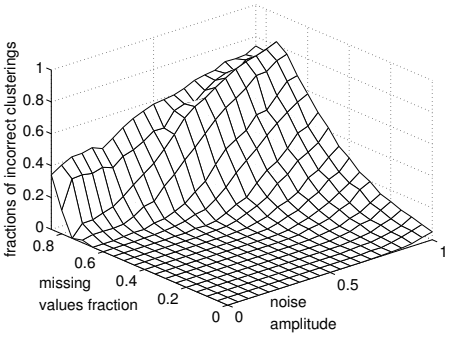

Fig. 1. Average fraction of incorrect clusterings by max-k-cut on a noisy distance matrix with missing data.

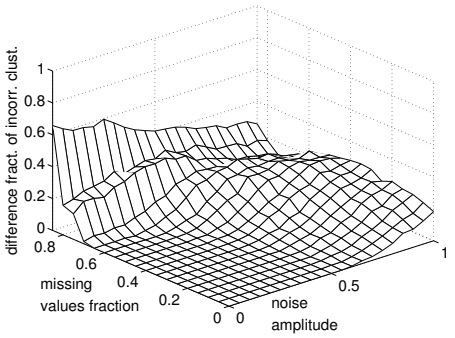

Fig. 2. Difference of the average fraction of incorrect clusterings by norm-k-cut relative to max-k-cut, where the similarity matrix W^D was obtained from the distance matrix D as $W^D = \exp(-\frac{1}{2\sigma^2}D^2)$.

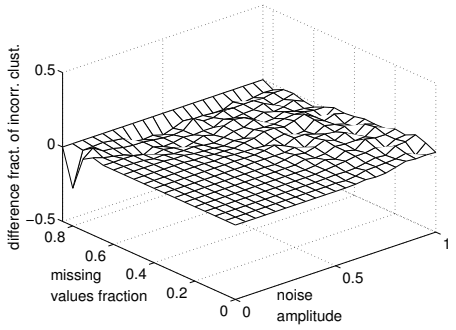

Fig. 3. Average fraction of incorrect clusterings by norm-k-cut: Difference of a $W^D = \exp(-\frac{1}{2\sigma^2}D^2)$ matrix to a directly generated similarity matrix W.

The same holds in principle for the norm-cut criterion, both for the directly generated similarity matrices W and for those matrices W^D derived from the distance matrix by means of the Gaussian kernel. However, in Figure 2, where the average difference of the error rates of the norm-cut clustering of W^D to the max-cut clustering of D is displayed, we can see: The norm-cut clustering always produces a higher error rate. The error rate is even more significantly higher for large fractions of missing values.

Did we introduce this increased error artificially by the additional transformation with the Gaussian kernel? Figure 3 indicates that this is not the case, as it shows nowhere a significantly positive value. Precisely, Figure 3 displays the difference of the error rates of the norm-cut clusterings of W^D relative to the directly generated matrices W.

Next, we turn to the computational resources required by the algorithms. Both max-cut and norm-cut are $\mathcal{NP}$-hard to optimize, so let us look at the relaxations. The spectral decomposition of a $n \times n$ matrix can be done in time $O(n^3)$, and if only the top k eigenvectors are desired, the effort can be even reduced to $O(kn^2)$ by an appropriate Lanczos method. Therefore the norm-cut/spectral algorithm has quadratic or (depending on the implementation) at most cubic complexity.

On the other hand, solving the SDP in order to approximate max-cut is more expensive. The respective complexity is $O(n^3 + m^3)$ (see [1]), where m is the number of constraints. If $k = 2$, then $m = n$ and the overall complexity is cubic. However, for $k \geq 3$, we need $m = O(n^2)$ constraints, resulting in an overall computational complexity of $O(n^6)$.

Finally, we remark that the analysis of the eigenvalues of the similarity matrix can yield a quite useful criterion to automatically determine the number k of clusters, in case that k is not known. We do not know of a corresponding method based on the distance matrix. We shall not further discuss this issue here and assume in the following that k is known.

6 The Google distance

Our sample application for the subsequent simulations will be clustering of natural language terms using the Google distance. This distance has been suggested by Cilibrasi and Vitányi [2] as a semantical distance function on pairs of words or terms. For instance, for most of today's people, the terms "Claude Debussy" and "Béla Bartók" are much tighter related than "Béla Bartók" and "Michael Schumacher". The World Wide Web represents parts of the world we live in as a huge collection of documents, mostly written in natural language. We briefly describe the derivation of the Google distance, starting with concepts from Kolmogorov complexity (Algorithmic Information Theory).

Let us fix a universal Turing machine (which one we fix is not relevant, since each universal machine can interpret each other by using a "compiler" program of constant length). For concreteness, we assume that its program tape is binary, such that all subsequent logarithms referring to program lengths are w.r.t. base 2 (which is also not relevant for our algorithms). The output alphabet is ASCII. Then, the (prefix) Kolmogorov complexity of a character string x is defined as

$$K(x) = \text{length of the shortest self-delimiting program generating } x,$$

where by the requirement "self-delimiting" we make sure that the programs form a prefix-free set and therefore the Kraft inequality holds:

$$\sum_x 2^{-K(x)} \leq 1 \ , \text{ where } x \text{ ranges over all ASCII strings}$$

The Kolmogorov complexity is a well-defined quantity regardless of the choice of the universal Turing machine, up to an additive constant.

If x and y are ASCII strings and x^* and y^* are their shortest (binary) programs, respectively, we can define $K(y|x^*)$, which is the length of the shortest self-delimiting program generating y where x^*, the program for x, is given. $K(x|y^*)$ is computed analogously. Thus, we may follow [8] and define the *universal similarity metric* as

$$d(x,y) = \frac{\max\{K(y|x^*), K(x|y^*)\}}{\max\{K(x), K(y)\}} \tag{7}$$

This can be interpreted as (approximately) the ratio by which the complexity of the more complex string decreases, if we already know how to generate the less complex string. The similarity metric is almost a metric according to the usual definition, as it satisfies the metric (in)equalities up to order $1/\max\{K(x), K(y)\}$.

Given a collection of documents like the World Wide Web, we can define the probability of a term or a tuple of terms just by counting relative frequencies. That is, for a tuple of terms $X = (x_1, x_2, \ldots, x_n)$, where each term x_i is an ASCII string, we set

$$p^{\text{www}}(X) = p^{\text{www}}(x_1, x_2, \ldots, x_n) = \frac{\#\text{ web pages cont. all } x_1, x_2, \ldots, x_n}{\#\text{ relevant web pages}}. \tag{8}$$

Conditional probabilities can be defined likewise as

$$p^{\text{www}}(Y|X) = p^{\text{www}}(Y \diamond X)/p^{\text{www}}(X)\ ,$$

where X and Y are tuples of terms and $\diamond$ denotes the concatenation. Although the probabilities defined in this way do not satisfy the Kraft inequality, we may still define complexities

$$K^{\text{www}}(X) = -\log\left(p^{\text{www}}(X)\right) \text{ and } K^{\text{www}}(Y|X) = K^{\text{www}}(Y \diamond X) - K^{\text{www}}(X). \tag{9}$$

Then we use (7) in order to define the *web distance* of two ASCII strings x and y, following Cilibrasi and Vitányi [2], as

$$d^{\text{www}}(x,y) = \frac{K^{\text{www}}(x \diamond y) - \min\{K^{\text{www}}(x), K^{\text{www}}(y)\}}{\max\{K^{\text{www}}(x), K^{\text{www}}(y)\}} \tag{10}$$

We query the page counts of the pages by using the Google API, so we call d^{www} the Google distance. Since the Kraft inequality does not hold, the Google distance is quite far from being a metric, unlike the universal similarity metric above.

The "number of relevant web pages" arising in (8) will be estimated by hand for all of the subsequent simulations. Actually, the clustering is not very sensitive to this value. On the other hand, Google no longer publishes its database size, and the full database size would not be the most appropriate value anyway for many applications. E.g., if we cluster words in a not so common language, then the index size relative to this language might be more appropriate.

7 Experimental results with the Google distance

We evaluate both clustering algorithms from Sections 2 and 3 on a set of natural language terms clustering tasks. We used the following datasets, which are all available at `http://www-alg.ist.hokudai.ac.jp/datasets.html` .

data set information				clustering errors and comp. time			
name	size	#clusters	missing data	max-cut/SDP		norm-cut/spectral	
`people2`	50	2	0%	0	(4 sec)	0	(0.07 sec)
`people3`	75	3	0%	0	($\sim$ 90 sec)	2	(0.13 sec)
`people4`	100	4	0%	1	($\sim$ 876 sec)	5	(0.2 sec)
`people5`	125	5	0%	4	($\sim$ 2544 sec)	8	(0.35 sec)
`alt-ds`	64	2	0%	1	(3 sec)	1	(0.1 sec)
`math-med-fin`	60	3	0%	1	($\sim$ 36 sec)	1	(0.1 sec)
`finance-cs-j`	30	2	0%	4	(1.8 sec)	1	(0.05 sec)
`phil-avi-d`	198	2	50%	5	(12 sec)	6	(2 sec)
`math-cuisine`	600	2	70%	23	(137 sec)	22	(16.6 sec)

Table 1. Empirical comparison of the algorithms on the basic data sets (without removing additional data).

The dataset `people2` contains the names of 25 famous classical composers and 25 artists (i.e., two intended clusters), `people3` contains all names from `people2` plus 25 bestseller authors, `people4` is extended by 25 mathematicians, and `people5` additionally contains 25 classical composers. The dataset `alt-ds` contains not terms in natural language, but rather titles and authors' last names from (almost all of) the papers from the ALT 2004 and DS 2004 conferences. Furthermore we use the datasets `math-med-fin` containing 20 terms each from the mathematical, medical, and financial terminology, `finance-cs-j` contains 20 financial and 10 computer science terms in Japanese, `phil-avi-d` has 98 terms from philately and 100 terms from aviation in German, and `math-cuisine` has 254 mathematical and 346 cuisine-related terms (in English). The distance matrices of the last two data sets are not fully given: in `phil-avi` only 50% of the entries are known, in `math-cuisine` it is only 30%.

For the norm-cut based algorithm, we need to convert the distance matrix to a similarity matrix. We do this by using a Gaussian kernel $W^D = \exp(-\frac{1}{2\sigma^2}D^2)$ and set the width parameter $\sigma = \bar{D}/\sqrt{2}$, which gives good results in practice. Another almost equally good choice is $\sigma = \frac{1}{3}$, which can be justified by the fact that the Google distance is scale invariant and mostly in $[0, 1]$.

Table 1 shows the number of clustering errors, i.e., the number of data points that are sorted to a different group than the intended one, respectively, on the data sets just described. One can see that both algorithms perform well in principle, in fact many of the "errors" displayed are in reality ambiguities of the data, e.g., the only misclustering in the `math-med-fin` data set concerns the

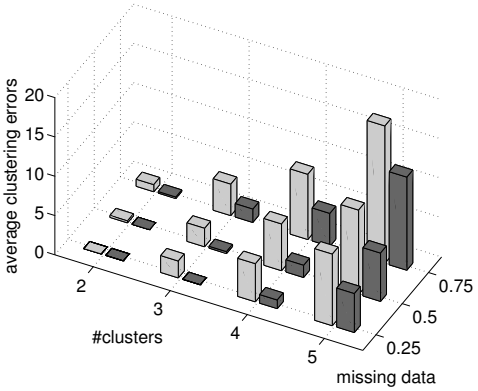

Fig. 4. Comparison of the max-cut/SDP (dark bars) and the norm-cut/spectral algorithm (light bars) with variable fraction of missing data and variable number of clusters and data set size, on the data sets `people2-people5`.

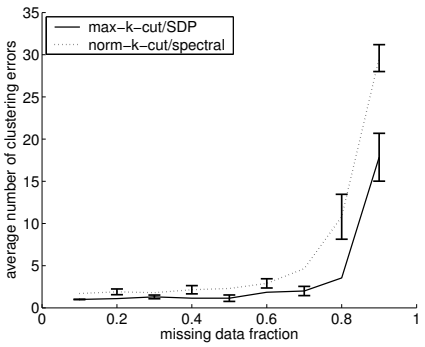

Fig. 5. Comparison with variable fraction of missing data on the `math-med-fin` set.

term "`average`" which was intended to belong to the mathematical terms but ended up in the financial group.

We remark that the constraints (1c) and the resulting huge SDP size were really necessary in order to get reasonable results: Without these constraints, e.g., clustering the `people5` data set with the SDP algorithm, the resulting average number of errors is 36.

Looking on the computation times in Table 1 (measured on a 3 Ghz Pentium IV), the spectral method is clearly much faster than the SDP, in particular for $k = 3$ or more clusters. Here the quadratic number of constraints in the SDP and the resulting 6th order computation time are really expensive. Actually, the available SDP software (CSDP, SeDuMi) do not even work at all with much larger problems if $k \geq 3$.

Next we consider a situation with varying fraction of missing data, shown in Figure 4 for the data sets `people2-people5`. Here the max-cut/SDP algorithm consistently outperforms the norm-cut/spectral algorithm, in particular if the number of clusters or the fraction of missing data grows. The same can be observed on the `math-med-fin` as shown in Figure 5. Note that both algorithms work quite well until about 70% missing data, after that the error increases sharply. Both figures are based on 20 independent samples of missing data each, where the missing data locations were sampled in a balanced way such that each row and column of the distance matrix has the same fraction of missing values. Figure 5 also displays 95%-confidence bars based on the standard normal assumption.

8 Relations to other work and Conclusions

There are many papers on clustering based on similarity matrices, in particular spectral clustering. It seems that norm-(k-)cut is quite established as an ideal criterion here, but there are different, such as min-max cut [3]. But also SDP has been used in connection with spectral clustering and kernels: [11] propose a SDP relaxation for norm-k-cut clustering based on a similarity matrix, while [6] and [7] use SDP for completion and learning of kernel matrices, respectively.

To our knowledge, the present work is the first one to use a distance matrix and a max-(k-)cut criterion for similar clustering tasks, which is natural in many applications where distances are given instead of similarities. We have seen that a SDP relaxation works quite well and yields results which tend to be superior to the spectral clustering results, in particular if the fraction of missing values grows. However, the SDP relaxation is expensive for $k = 3$ or more clusters. Thus we conclude with the open question of how to obtain a more efficient relaxation of max-k-cut, for instance a spectral one.

References

[1] B. Borchers and J. G. Young. Implementation of a primal-dual method for sdp on a shared memory parallel architecture. March 27, 2006.
[2] R. Cilibrasi and P. M. B. Vitányi. Automatic meaning discovery using Google. Manuscript, CWI, Amsterdam, 2006.
[3] C. H. Q. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *ICDM '01: Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 107–114. IEEE Computer Society, 2001.
[4] A. Frieze and M. Jerrum. Improved algorithms for MAX k-CUT and MAX BISECTION. *Algorithmica*, 18(1):67–81, 1997.
[5] M. X. Goemans and D. P. Williamson. .879-approximation algorithms for MAX CUT and MAX 2SAT. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 422–431. ACM Press, 1994.
[6] T. Graepel. Kernel matrix completion by semidefinite programming. In *ICANN '02: Proceedings of the International Conference on Artificial Neural Networks*, pages 694–699. Springer-Verlag, 2002.
[7] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan. Learning the kernel matrix with semidefinite programming. *JMLR*, 5:27–72, 2004.
[8] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.
[9] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.
[10] J. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(12):625–653, 1999.
[11] E. P. Xing and M. I. Jordan. On semidefinite relaxation for normalized k-cut and connections to spectral clustering. Technical Report UCB/CSD-03-1265, EECS Department, University of California, Berkeley, 2003.
[12] S. X. Yu and J. Shi. Multiclass spectral clustering. In *ICCV '03: Proceedings of the Ninth IEEE International Conference on Computer Vision*, pages 313–319. IEEE Computer Society, 2003.