The Normalized Information Distance and its Applications Using Graph Cuts

Thomas Zeugmann

Division of Computer Science
Graduate School of Information Science and Technology
Hokkaido University

ALG-LAB Seminar, May 17, 2018

Motivation I

The similarity between objects is a fundamental notion in everyday life. It is also fundamental to many data mining and machine learning algorithms, especially clustering algorithms. Most often the similarity between objects is measured by a domain-specific distance measure based on features of the objects.

Motivation I

The similarity between objects is a fundamental notion in everyday life. It is also fundamental to many data mining and machine learning algorithms, especially clustering algorithms. Most often the similarity between objects is measured by a domain-specific distance measure based on features of the objects.

For example, the distance between pieces of music can be measured using features like rhythm, pitch, or melody, that is, features that do not make sense in any other domain. Such methods need special knowledge about the application domain for extracting the relevant features beforehand.

Motivation I

The similarity between objects is a fundamental notion in everyday life. It is also fundamental to many data mining and machine learning algorithms, especially clustering algorithms. Most often the similarity between objects is measured by a domain-specific distance measure based on features of the objects.

For example, the distance between pieces of music can be measured using features like rhythm, pitch, or melody, that is, features that do not make sense in any other domain. Such methods need special knowledge about the application domain for extracting the relevant features beforehand.

This is not only difficult, it also runs the risk of being *biased*. Another consequence is that data mining algorithms tend to have many parameters by which the algorithm's sensitivity to certain features can be controlled, or, put differently, must be *tuned*.

Motivation II

As a radically different approach, the paradigm of parameter-free data mining has emerged.

Motivation II

As a radically different approach, the paradigm of parameter-free data mining has emerged.

Its main characteristic is that the algorithms have no parameters and that essentially the same algorithm can be applied in all areas. The most promising approach to this paradigm uses Kolmogorov complexity theory as its basis. In the past decade, various researchers developed the so-called normalized information distance (NID) in a series of steps. We follow Vitányi, Balbach, Cilibrasi and Li to explain these steps.

The Normalized Information Distance I

In the following we consider binary strings $x \in \{0,1\}^*$ and use λ for the empty string. We totally order all binary strings according to their length and within the same length lexicographically.
We write $\ell(x)$ to denote the *length* of x, thus $\ell(x) = \lfloor \log(x+1) \rfloor$.

For any string x we denote by $\overline{x}$ the string $x = 1^{\ell(x)}0x$, called the *self-delimiting encoding* of x. The set $\{\overline{x} \mid x \in \{0,1\}^*\}$ then is a *prefix set*, that is, no element of it is prefix of another element. Prefix sets have an important property, namely they satisfy the Kraft inequality, i.e.,

$$\sum_{x \in S} 2^{-\ell(x)} \leqslant 1 \,, \qquad \text{for any prefix set } S \subseteq \{0,1\}^* \,.$$

The Normalized Information Distance II

Partial functions whose domain is a prefix set are called *prefix functions*. They are important for defining Kolmogorov complexity. Defining a pairing function $\langle x, y \rangle = \bar{x}y$ and extending it in the usual way to n-tuples, it suffices to consider functions of one argument.

The Normalized Information Distance II

Partial functions whose domain is a prefix set are called *prefix functions*. They are important for defining Kolmogorov complexity. Defining a pairing function $\langle x, y \rangle = \bar{x}y$ and extending it in the usual way to n-tuples, it suffices to consider functions of one argument.

Consider a function φ and two strings p and x such that $\varphi(p) = x$. We interpret p as a description of x by means of the *description language* φ. The length of a *shortest* description p is called the *complexity* of the string x with respect to φ.

The Normalized Information Distance II

Partial functions whose domain is a prefix set are called *prefix functions*. They are important for defining Kolmogorov complexity. Defining a pairing function $\langle x, y \rangle = \bar{x}y$ and extending it in the usual way to n-tuples, it suffices to consider functions of one argument.

Consider a function φ and two strings p and x such that $\varphi(p) = x$. We interpret p as a description of x by means of the *description language* φ. The length of a *shortest* description p is called the *complexity* of the string x with respect to φ.

Definition

Let φ be a partial recursive function. The *conditional complexity* (with respect to φ) of x given y is defined by

$$C_\varphi(x|y) = \min\{\ell(p) \mid \varphi(y, p) = x\},$$

and the *unconditional complexity* of x by $C_\varphi(x) = C_\varphi(x|\lambda)$.

The Normalized Information Distance III

Theorem

There is a partial recursive function φ_0 such that for all partial recursive functions φ there is a constant c with

$$C_{\varphi_0}(x|y) \leqslant C_{\varphi}(x|y) + c \quad \textit{for all } x,\ y\ .$$

The Normalized Information Distance III

Theorem

There is a partial recursive function φ_0 *such that for all partial recursive functions* φ *there is a constant* c *with*

$$C_{\varphi_0}(x|y) \leqslant C_{\varphi}(x|y) + c \quad \textit{for all } x,\ y\ .$$

While simple and elegant, the notion of C_{φ_0} has some oddities, e.g., $C_{\varphi_0}(xy)$ is in general not upper-bounded by the sum of $C_{\varphi_0}(x)$ and $C_{\varphi_0}(y)$. To overcome these oddities it is beneficial not to consider *all* partial recursive functions but only the prefix functions.

The Normalized Information Distance III

Theorem

There is a partial recursive function φ_0 *such that for all partial recursive functions* φ *there is a constant* c *with*

$$C_{\varphi_0}(x|y) \leqslant C_{\varphi}(x|y) + c \quad \textit{for all } x,\ y\ .$$

While simple and elegant, the notion of C_{φ_0} has some oddities, e.g., $C_{\varphi_0}(xy)$ is in general not upper-bounded by the sum of $C_{\varphi_0}(x)$ and $C_{\varphi_0}(y)$. To overcome these oddities it is beneficial not to consider *all* partial recursive functions but only the prefix functions.

Theorem

There is a partial recursive prefix function ψ_0 *such that for all partial recursive prefix functions* ψ *there is a constant* c *with*

$$C_{\psi_0}(x|y) \leqslant C_{\psi}(x|y) + c \quad \textit{for all } x,\ y\ .$$

The Normalized Information Distance IV

We write $K(x|y)$ and $K(x)$ instead of $C_{\psi_0}(x|y)$ and $C_{\psi_0}(x|\lambda)$, respectively and refer to K as the *Kolmogorov complexity*.

Note that in contrast to C_{φ_0}, all shortest programs that "occur" in K constitute a prefix set. This implies that we can concatenate any such two programs and still recognize them as two distinct programs. This allows the construction of a program that simulates two other programs and combines their output while being only a constant number of bits larger than the concatenation of the original two programs.
The following theorem summarizes major properties of K.

The Normalized Information Distance V

Theorem

(1) K *is not partial recursive.*

(2) $\mathrm{K}(x) \leqslant \ell(x) + 2\log \ell(x) + \mathrm{O}(1)$ *for all* x.

(3) $\mathrm{K}(x,y) \leqslant \mathrm{K}(x) + \mathrm{K}(y|x) + \mathrm{O}(1)$ *for all* x, y.

(4) *Up to an additional term of* $\mathrm{O}(1)$*: for all* x, y

$$\mathrm{K}(x,y) = \mathrm{K}(x) + \mathrm{K}(y|\langle x, \mathrm{K}(x)\rangle) = \mathrm{K}(y) + \mathrm{K}(x|\langle y, \mathrm{K}(y)\rangle)$$

(5) *Up to an additional term of* $\mathrm{O}(\log \mathrm{K}(xy))$*:*

$$\mathrm{K}(x,y) = \mathrm{K}(x) + \mathrm{K}(y|x) = \mathrm{K}(y) + \mathrm{K}(x|y) \quad \textit{for all } x,\ y\ .$$

The Normalized Information Distance VI

All programs that can be identified as shortest ones by K form a prefix set. By the Kraft inequality we then have

$$\sum_x 2^{-K(x)} \leqslant 1 \, .$$

So, we can regard the quantities $2^{-K(x)}$ as quantities very similar to probabilities.

The Normalized Information Distance VI

All programs that can be identified as shortest ones by K form a prefix set. By the Kraft inequality we then have

$$\sum_x 2^{-K(x)} \leqslant 1 \, .$$

So, we can regard the quantities $2^{-K(x)}$ as quantities very similar to probabilities.

Let us come back now to our goal to define minimal information distance. This should be the length of the shortest program for a universal computer to transform x into y and y into x. Note that we *cannot* use $K(y|x)$ itself, since $K(\lambda|x)$ is small for all x but intuitively, a random string is far from the empty string.

The Normalized Information Distance VII

The asymmetry of the conditional complexity $K(y|x)$ can be overcome by using $K(y|x) + K(x|y)$ but the resulting metric will overestimate the information required to translate between x and y in case there is some redundancy between the information required to get x from y and from y to x. Thus, we define:

Definition

The *max distance* between x and y is $E(x, y) = \max\{K(x|y), K(y|x)\}$.

The Normalized Information Distance VIII

Note that E satisfies the *density conditions*, i.e.,

$$\sum_{y\,:\,y\neq x} 2^{-\mathsf{E}(x,y)} \leqslant 1 \quad \text{and} \quad \sum_{x\,:\,x\neq y} 2^{-\mathsf{E}(x,y)} \leqslant 1\,;$$

(that means that there is at most a certain, finite number of objects y at a distance d from x).
Moreover, E is upper semicomputable.
Furthermore, we define:

Definition

An *admissible information distance* is a total, possibly asymmetric, nonnegative function on the pairs x, y of binary strings that is 0 if and only if $x = y$, is upper semicomputable, and satisfies the density requirement (1).

The Normalized Information Distance IX

Theorem

The function E *with* $E(x,y) = \max\{K(x|y), K(y|x)\}$ *is an admissible information distance and a metric. It is minimal in the sense that for every admissible information distance* D *we have* $E(x,y) \leqslant D(x,y) + O(1)$.

The Normalized Information Distance IX

Theorem

The function E *with* $\mathsf{E}(x,y) = \max\{\mathsf{K}(x|y), \mathsf{K}(y|x)\}$ *is an admissible information distance and a metric. It is minimal in the sense that for every admissible information distance* D *we have* $\mathsf{E}(x,y) \leqslant \mathsf{D}(x,y) + O(1)$.

The latter theorem shows that the information distance E is is universal in that among all admissible distances it is always least. That is, it accounts for the *dominant* feature in which two objects are alike. For that reason E is also called the *universal information distance*.

The Normalized Information Distance IX

Theorem

The function E *with* $\mathsf{E}(x,y) = \max\{\mathsf{K}(x|y), \mathsf{K}(y|x)\}$ *is an admissible information distance and a metric. It is minimal in the sense that for every admissible information distance* D *we have* $\mathsf{E}(x,y) \leqslant \mathsf{D}(x,y) + O(1)$.

The latter theorem shows that the information distance E is is universal in that among all admissible distances it is always least. That is, it accounts for the *dominant* feature in which two objects are alike. For that reason E is also called the *universal information distance*. Many admissible distances are absolute, but if we want to express similarity, then we are more interested in relative ones. Thus, we have to normalize the universal information distance $\mathsf{E}(x,y)$ to obtain a *universal similarity distance*. It should give a value of 0 when objects are maximally similar and distance 1 when they are maximally dissimilar.

The Normalized Information Distance X

Definition

The *normalized information distance* between two strings x and y is defined as

$$NID(x,y) = \frac{\max\{K(x|y),\, K(y|x)\}}{\max\{K(x),\, K(y)\}}\,.$$

The Normalized Information Distance X

Definition

The *normalized information distance* between two strings x and y is defined as

$$NID(x,y) = \frac{\max\{K(x|y),\, K(y|x)\}}{\max\{K(x),\, K(y)\}}\,.$$

Remarks: Dividing by $\max\{K(x),\, K(y)\}$ is not the most obvious idea for normalizing E, but the more obvious ideas fail:

The Normalized Information Distance X

Definition

The *normalized information distance* between two strings x and y is defined as

$$NID(x,y) = \frac{\max\{K(x|y),\, K(y|x)\}}{\max\{K(x),\, K(y)\}}\,.$$

Remarks: Dividing by $\max\{K(x),\, K(y)\}$ is not the most obvious idea for normalizing E, but the more obvious ideas fail:

- Divide by the length. Then no matter whether we divide by the sum or maximum of the length, the triangle inequality is not satisfied.

The Normalized Information Distance X

Definition

The *normalized information distance* between two strings x and y is defined as

$$NID(x,y) = \frac{\max\{K(x|y),\, K(y|x)\}}{\max\{K(x),\, K(y)\}}\,.$$

Remarks: Dividing by $\max\{K(x),\, K(y)\}$ is not the most obvious idea for normalizing E, but the more obvious ideas fail:

- Divide by the length. Then no matter whether we divide by the sum or maximum of the length, the triangle inequality is not satisfied.
- Divide by $K(x,y)$. Then the distances will be 1/2 whenever x and y satisfy $K(x) \approx K(y) \approx K(x|y) \approx K(y|x)$. But in this situation, x and y are completely dissimilar, and we would expect distance values of about 1.

The NID and the NCD

Theorem (M. Li *et al.*)

The normalized information distance $NID(x, y)$ *takes values in the range* $[0, 1]$ *and is a metric, up to ignorable discrepancies.*

The NID and the NCD

Theorem (M. Li *et al.*)

The normalized information distance NID(x, y) *takes values in the range* $[0, 1]$ *and is a metric, up to ignorable discrepancies.*

Since the *NID* is defined via K, it is not computable. To apply this idea to real-world data mining tasks, standard compression algorithms, such as gzip, bzip2, or PPMZ, have been used as approximations of K. This yields the *normalized compression distance* (*NCD*).

The NID and the NCD

Theorem (M. Li *et al.*)

The normalized information distance $NID(x, y)$ *takes values in the range* $[0, 1]$ *and is a metric, up to ignorable discrepancies.*

Since the *NID* is defined via K, it is not computable. To apply this idea to real-world data mining tasks, standard compression algorithms, such as gzip, bzip2, or PPMZ, have been used as approximations of K. This yields the *normalized compression distance* (*NCD*).

Definition

The *normalized compression distance* between two strings x and y is defined as

$$NCD(x, y) = \frac{C(xy) - \min\{C(x),\ C(y)\}}{\max\{C(x),\ C(y)\}},$$

where C is any given data compressor.

The NCD

Note that the compressor C has to be computable and *normal* in order to make the NCD a useful approximation.

Definition

A compressor C is said to be *normal* if it satisfies the following axioms for all strings x, y, z and the empty string λ.

(1) $C(xx) = C(x)$ and $C(\lambda) = 0$; (identity)

(2) $C(xy) \geqslant C(x)$; (monotonicity)

(3) $C(xy) = C(yx)$; (symmetry)

(4) $C(xy) + C(z) \leqslant C(xz) + C(yz)$; (distributivity)

up to an additive $O(\log n)$ term, with n the maximal binary length of a string involved in the (in)equality concerned.

Google Distance I

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

$$p^{\text{www}}(X) = p^{\text{www}}(x_1, x_2, \ldots, x_n) = \tag{1}$$

$$\frac{\#\text{ web pages containing all } x_1, x_2, \ldots, x_n}{\#\text{ relevant web pages}}.$$

Google Distance I

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

$$p^{\text{www}}(X) = p^{\text{www}}(x_1, x_2, \ldots, x_n) = \tag{1}$$

$$\frac{\#\text{ web pages containing all } x_1, x_2, \ldots, x_n}{\#\text{ relevant web pages}}.$$

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.

Google Distance II (Example)

Let x be a string, e.g. $x =$ "abnormal".

Google Distance II (Example)

Let x be a string, e.g. $x =$ "abnormal".

$$p(x) = \frac{\#\,\text{web pages containing } x}{\#\,\text{relevant web pages}}$$

Google Distance II (Example)

Let x be a string, e.g. $x =$ "abnormal".

$$p(x) = \frac{\#\,\text{web pages containing } x}{\#\,\text{relevant web pages}}$$

What are the relevant web pages?

Google Distance II (Example)

Let x be a string, e.g. $x =$ "abnormal".

$$p(x) = \frac{\#\,\text{web pages containing } x \text{ and "の"}}{\#\,\text{web pages containing "の"}}$$

web pages containing "の" = the relevant web pages for Japanese

Google Distance II (Example)

Let x be a string, e.g. $x =$ "abnormal".

$$p(x) = \frac{\#\,\text{web pages containing } x \text{ and "the"}}{\#\,\text{web pages containing "the"}}$$

web pages containing "the" = the relevant web pages for English

Google Distance II (Example)

Let x be a string, e.g. $x =$ "abnormal".

$$p(x) = \frac{\#\,\text{web pages containing } x \text{ and "the"}}{\#\,\text{web pages containing "the"}}$$

web pages containing "the" = the relevant web pages for English

Conditional probabilities:

$$p(\text{"axiom"}|\text{"abnormal"}) = \frac{\#\text{ "axiom" and "abnormal" and "the"}}{\#\text{ "abnormal" and "the"}}$$

Google Distance III

Although the probabilities defined in this way do not satisfy the Kraft inequality, we may still define *complexities*

$$\begin{aligned} K^{www}(X) &= -\log\left(p^{www}(X)\right) \text{ and} \\ K^{www}(Y \mid X) &= K^{www}(Y \diamond X) - K^{www}(X)\,. \end{aligned}$$

Google Distance III

Although the probabilities defined in this way do not satisfy the Kraft inequality, we may still define *complexities*

$$\begin{aligned} K^{www}(X) &= -\log\left(p^{www}(X)\right) \text{ and} \\ K^{www}(Y \mid X) &= K^{www}(Y \diamond X) - K^{www}(X)\,. \end{aligned}$$

Then we use the *NID* in order to define the *web distance* of two ASCII strings x and y,

$$d^{www}(x,y) = \frac{K^{www}(x \diamond y) - \min\left\{K^{www}(x), K^{www}(y)\right\}}{\max\left\{K^{www}(x), K^{www}(y)\right\}} \tag{2}$$

We call d^{www} the *Google distance*.

Clustering I

Using the *NCD* or the Google distance results in computing a *distance matrix* $d(x,y)_{x,y\in X}$, where $X = (x_1,\ldots,x_n)$ is the relevant data list. However, many widely used clustering programs work with similarities, e.g., for spectral clustering one has to perform the following steps:

$$\text{distance} \rightsquigarrow \text{similarity} \rightsquigarrow \text{Laplacian} \rightsquigarrow \text{spectrum} \rightsquigarrow \text{clustering}$$

For performing the step distance $\rightsquigarrow$ similarity we used a Gauss kernel

$$s(x,y) = e^{-\frac{1}{2\sigma^2}\cdot d(x,y)^2}$$

The kernel has to be chosen appropriately and the clustering is quite sensitive to this choice.

Questions

It is therefore natural to ask:

Questions

Can we work directly with distances instead of using similarities?

Questions

It is therefore natural to ask:

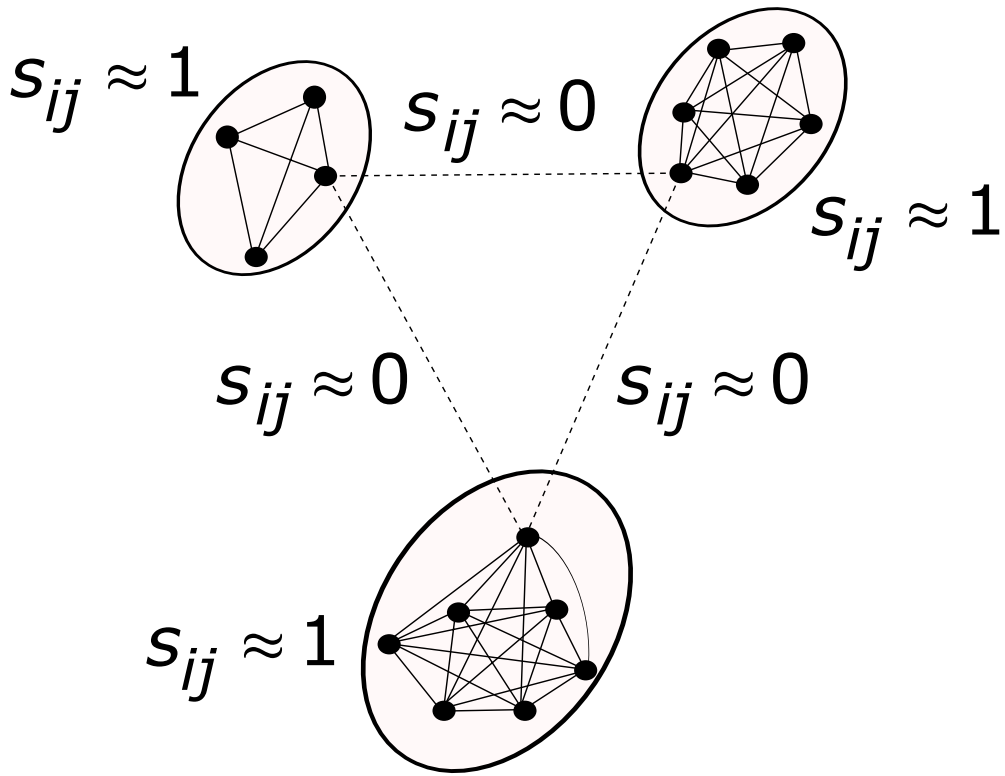
Questions

Can we work directly with distances instead of using similarities?

What has to be changed to achieve this goal?

Recalling What We have Done

Graph defined by adjacency matrix of pairwise similarities



Normalized Min-cut

- The resulting problem is called *normalized min-cut* and known to be $\mathcal{NP}$-hard.

Normalized Min-cut

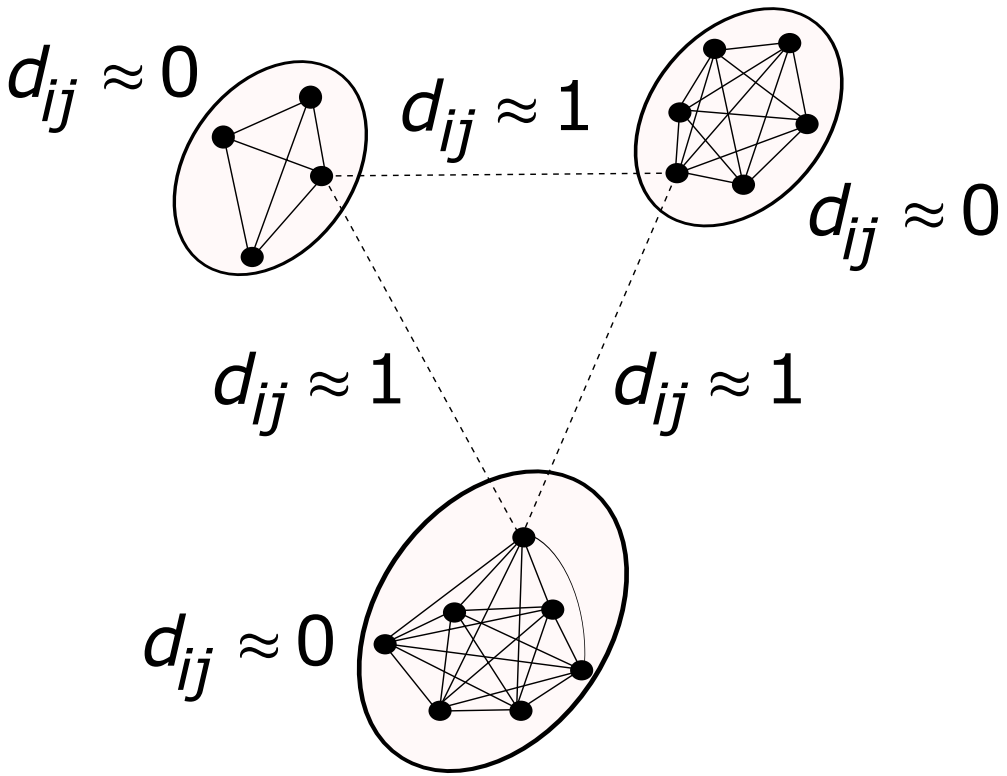
- The resulting problem is called *normalized min-cut* and known to be $\mathcal{NP}$-hard.
- Spectral clustering can be related to approximating the normalized min-cut (cf. Yu and Shi (2003)).

Normalized Min-cut

- The resulting problem is called *normalized min-cut* and known to be $\mathcal{NP}$-hard.
- Spectral clustering can be related to approximating the normalized min-cut (cf. Yu and Shi (2003)).
- Spectral clustering needs time $O(n^3)$.

Looking at Distances

Graph defined by adjacency matrix of pairwise distances



Maximum-cut

- The resulting problem is called *maximum-cut* and known to be $\mathcal{NP}$-hard.

Maximum-cut

- The resulting problem is called *maximum-cut* and known to be $\mathcal{NP}$-hard.
- We need a new approximation method (this is done below via maximum-cut $\rightsquigarrow$ IP $\rightsquigarrow$ SDP.)

Maximum-cut

- The resulting problem is called *maximum-cut* and known to be $\mathcal{NP}$-hard.
- We need a new approximation method (this is done below via maximum-cut $\rightsquigarrow$ IP $\rightsquigarrow$ SDP.)
- Semidefinite programming (abbr. SDP) needs time $O(n^3)$ if there are two clusters, and $O(n^6)$ if there are $k > 2$ clusters.

Spectral Clustering I

Observation: Spectral decomposition of a similarity matrix gives a clustering!

Spectral Clustering I

Observation: Spectral decomposition of a similarity matrix gives a clustering!

Example: The matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$

has eigenvectors

$$\begin{pmatrix} 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & -2 \end{pmatrix}$$

and eigenvalues

$$\begin{pmatrix} 3 & 2 & 0 & 0 & 0 \end{pmatrix}$$

Spectral Clustering I

Observation: Spectral decomposition of a similarity matrix gives a clustering!

Example: The matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$$ has eigenvectors $$\begin{pmatrix} 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & -2 \end{pmatrix}$$

and eigenvalues $$\begin{pmatrix} 3 & 2 & 0 & 0 & 0 \end{pmatrix}$$

Theory indicates that this works also with noise.

Spectral Clustering I

Observation: Spectral decomposition of a similarity matrix gives a clustering!

Example: The matrix

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \text{ has eigenvectors } \begin{pmatrix} 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & -2 \end{pmatrix}$$

$$\text{and eigenvalues } \begin{pmatrix} 3 & 2 & 0 & 0 & 0 \end{pmatrix}$$

Theory indicates that this works also with noise.

Theory indicates that one should normalize the matrix before spectral analysis $\Rightarrow$ Laplacian

Spectral Clustering II

Algorithm Spectral clustering of a word list
Input: word list $X = (x_1, x_2, \ldots, x_n)$, number of clusters k
Output: clustering $c \in \{1, \ldots, k\}^n$

1. for $x, y \in X$, compute Google relative frequencies $p^{www}(x)$, $p^{www}(x, y)$
2. for $x, y \in X$, compute complexities $K^{www}(x)$, $K^{www}(x, y)$
3. compute distance matrix $D = \left(d^{www}(x, y)\right)_{x,y \in X}$
4. compute $\sigma = \mathit{mean}(D)/\sqrt{2}$
5. compute similarity matrix $A = \left(\exp(-\frac{1}{2} d^{www}(x, y)^2/(2\sigma^2))\right)$
6. compute Laplacian $L = S^{-\frac{1}{2}} A S^{-\frac{1}{2}}$, where $S_{ii} = \sum_j A_{ij}$ and $S_{ij} = 0$ for $i \neq j$
7. compute top k eigenvectors $V \in \mathbb{R}^{n \times k}$
8. cluster V using kLines (Fischer and Poland (2004))

Semidefinite Programming I

- Given a weighted graph $G = (V, D)$ with vertices $V = \{x_1, \ldots, x_n\}$ and edge weights $D = \{d_{ij} \geqslant 0 \mid 1 \leqslant i, j \leqslant n\}$ 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 $A(i, j) = 0$ if $\exists \ell [1 \leqslant \ell \leqslant k,\ 1 \leqslant i, j \leqslant n$ 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

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

Semidefinite Programming II

The *max-k-cut* problem is the task of finding the partition that maximizes the weight of the cut.
Let

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

be the d-dimensional unit sphere, and let $a_1, \ldots, a_k \in \mathcal{S}^{k-2}$ be the vertices of a regular simplex.
Then the inner product $a_i \cdot a_j = -\frac{1}{k-1}$ whenever $i \neq j$.

Semidefinite Programming II

The *max-k-cut* problem is the task of finding the partition that maximizes the weight of the cut.
Let

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

be the d-dimensional unit sphere, and let $a_1, \ldots, a_k \in S^{k-2}$ be the vertices of a regular simplex.
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:

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

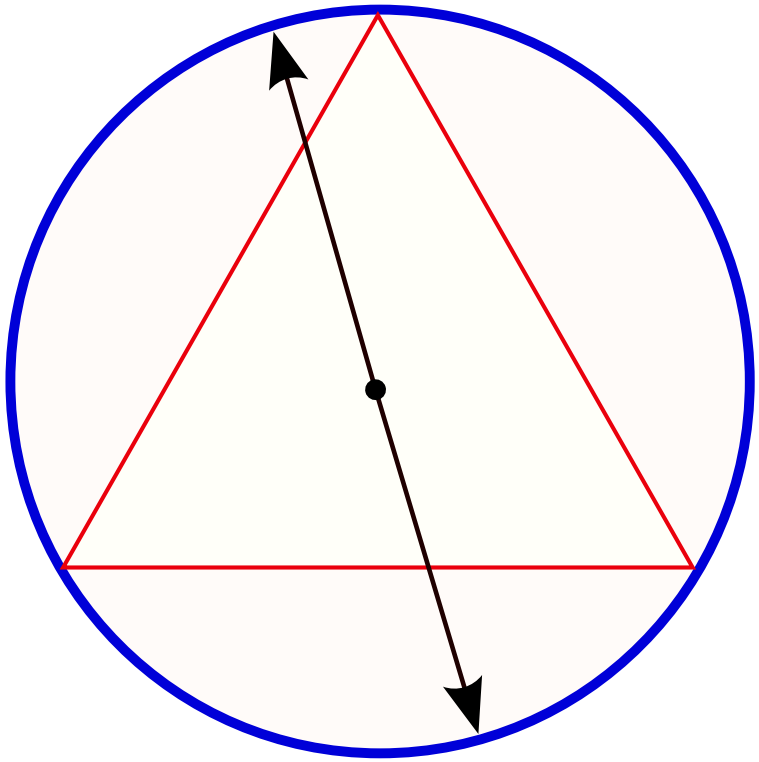
Semidefinite Programming III

Frieze and Jerrum (1997) propose the following semidefinite program (SDP) in order to relax the integer program:

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

The constraints $v_i \cdot v_j \geqslant -\frac{1}{k-1}$ are necessary for $k \geqslant 3$ because otherwise the SDP would prefer solutions where $v_i \cdot v_j = -1$, resulting in a larger value of the objective.

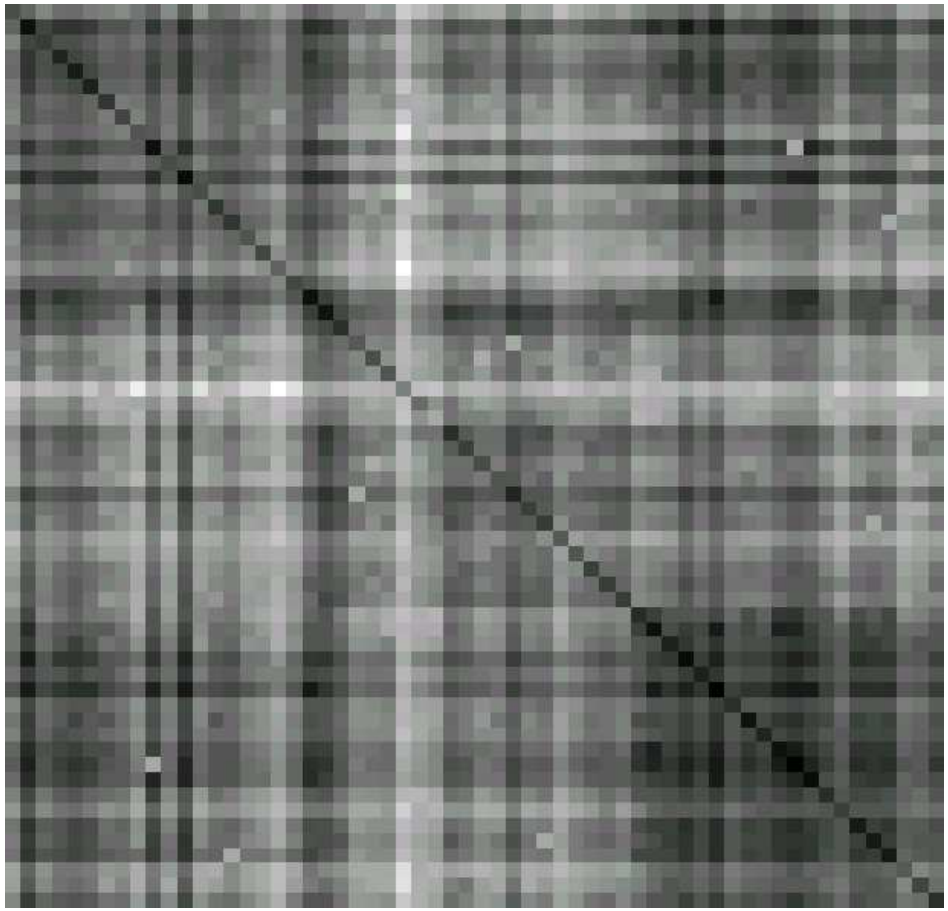
Illustration



Math-Med-Finance: an Example Data Set

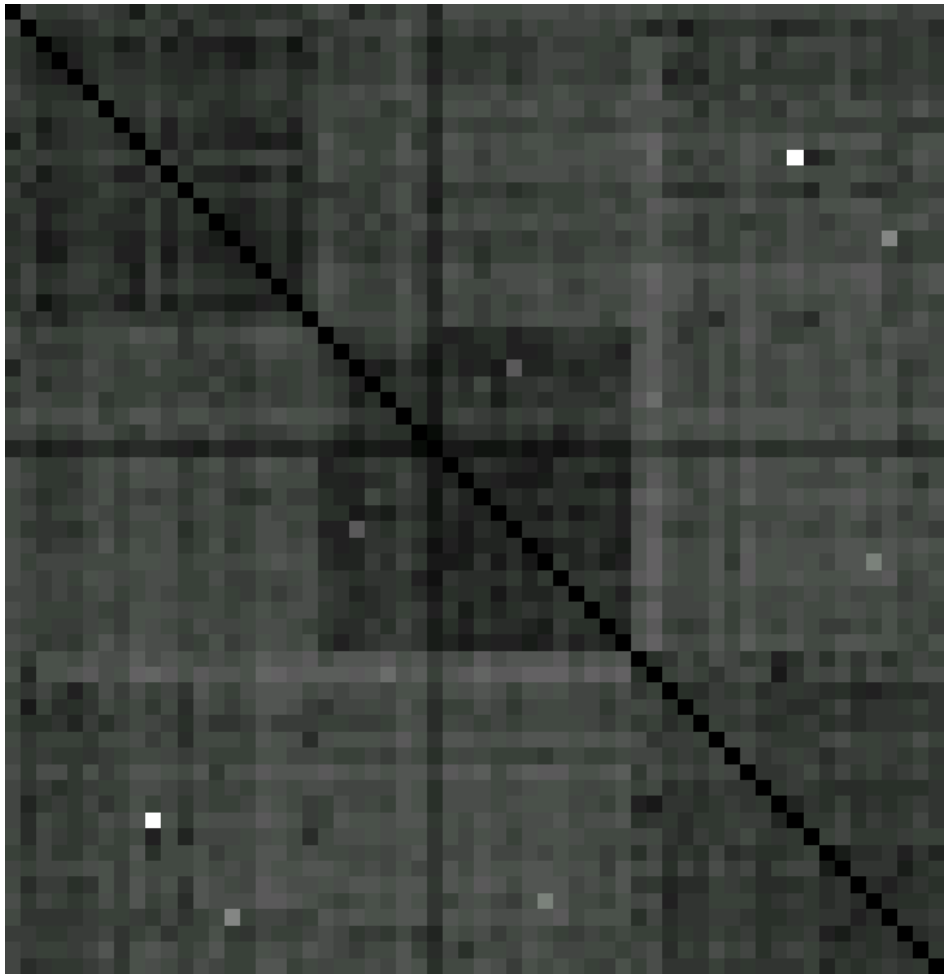
axiom, average, coefficient, probability, continuous, coordinate, cube, denominator, disjoint, domain, exponent, function, histogram, infinity, inverse, logarithm, permutation, polyhedra, quadratic, random, cancer, abnormal, abscess, bacillus, delirium, betablocker, vasomotor, hypothalamic, cardiovascular, chemotherapy, chromosomal, dermatitis, diagnosis, endocrine, epilepsy, oestrogen, ophthalmic, vaccination, traumatic, transplantation, nasdaq, investor, obligation, benefit, bond, account, clearing, currency, deposit, stock, market, option, bankruptcy, creditor, assets, liability, transactions, insolvent, accrual, unemployment

Complexities for the Math-Med-Finance Data



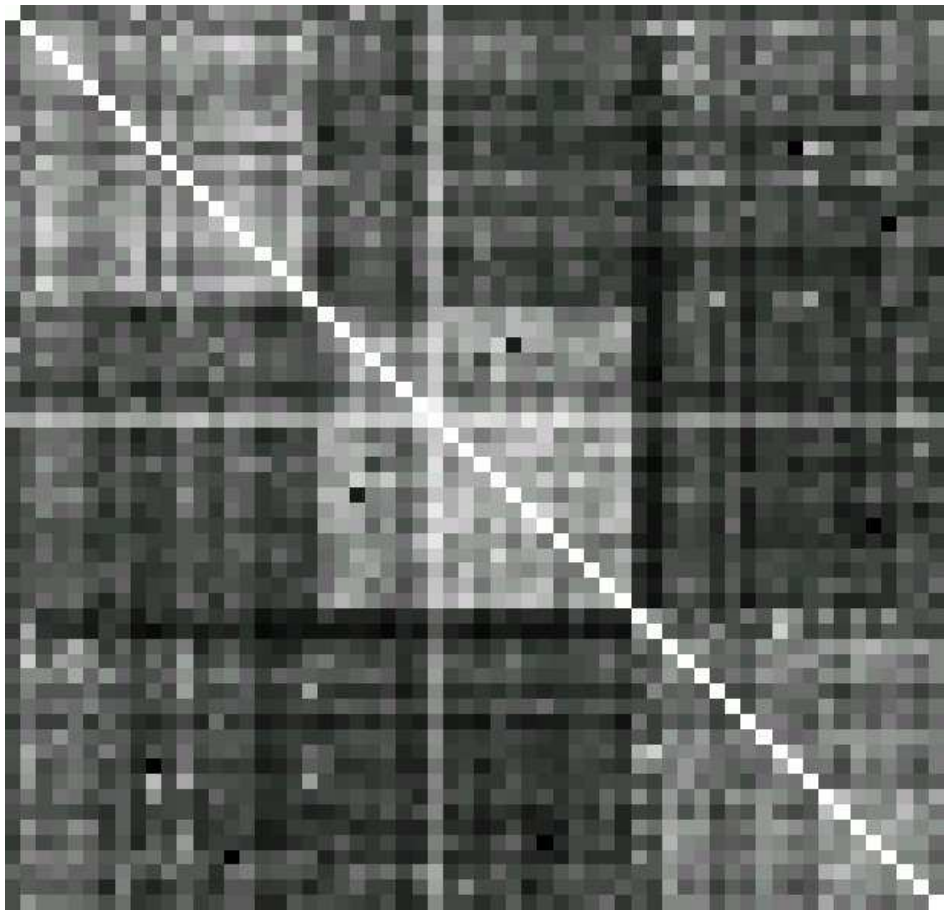
Large complexities are white, small complexities black

Distances for the Math-Med-Finance Data



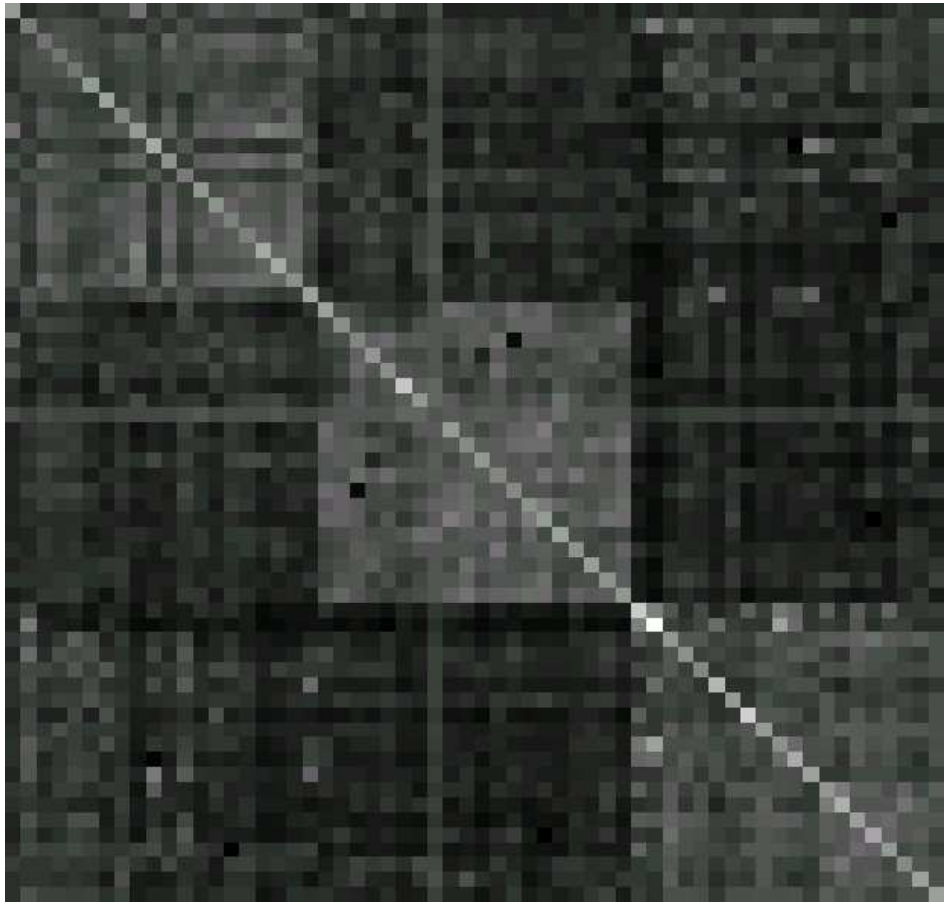
The block structure is already visible.

Similarities for the Math-Med-Finance Data



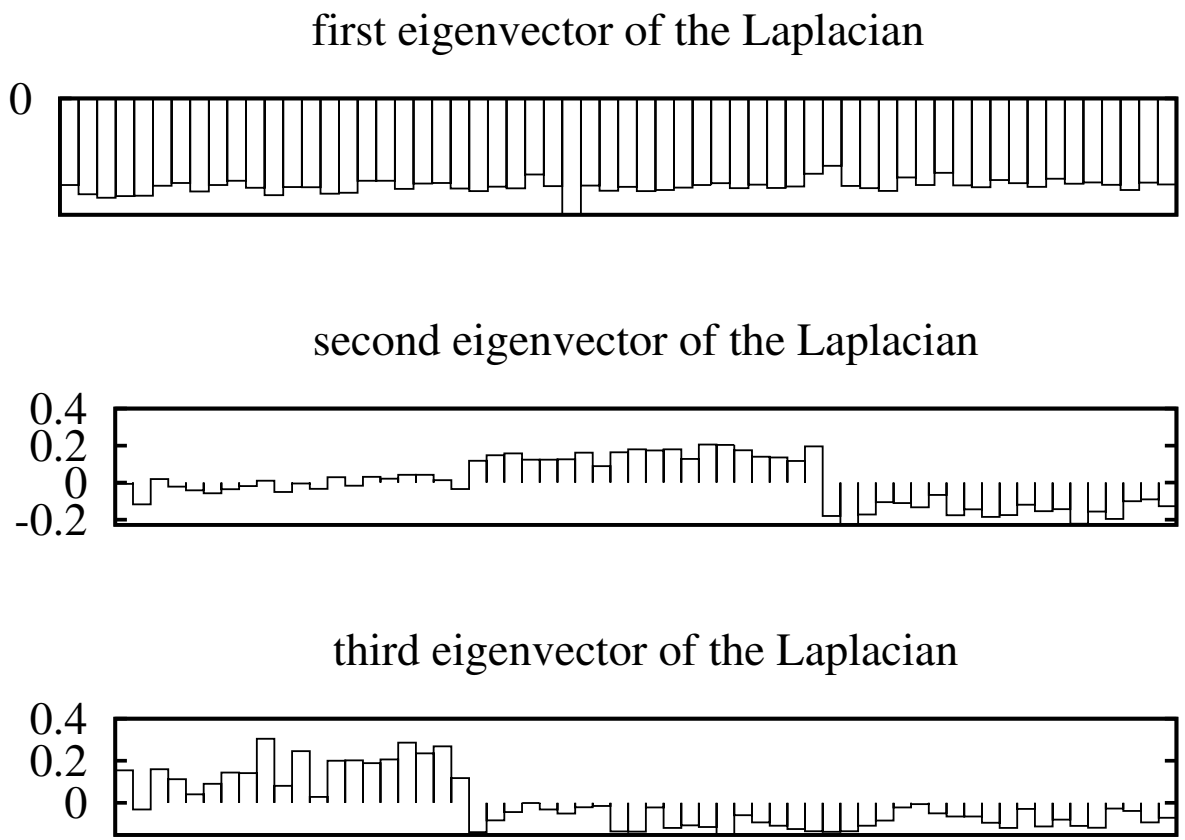
The block structure is again visible.

Laplacian for the Math-Med-Finance Data



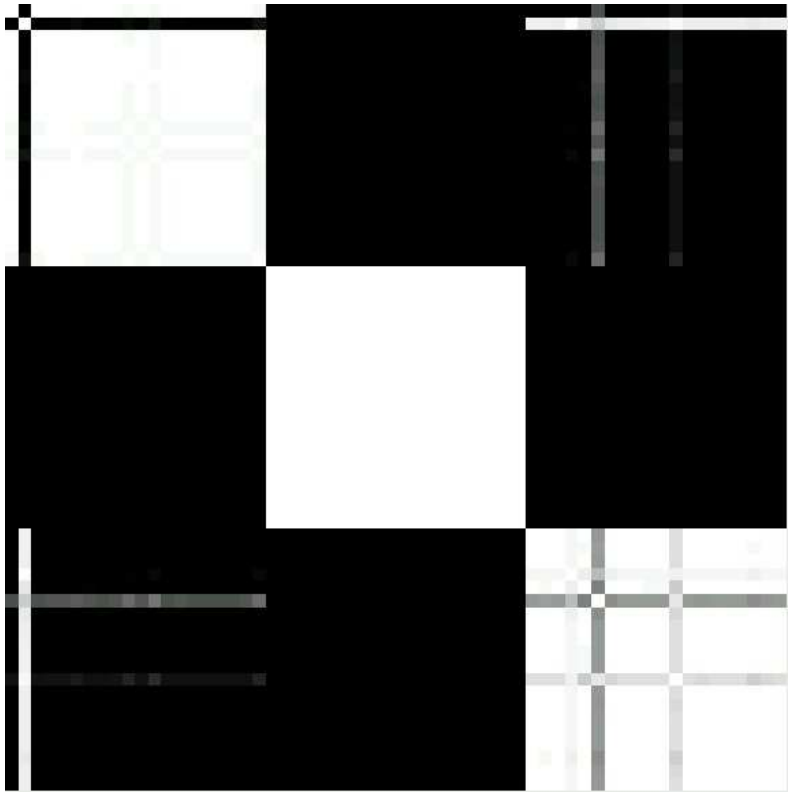
The block structure is clearly visible.

Eigenvector Plot for the Math-Med-Finance Data



So, there is one error, i.e., "`average`" which assigned to the financial terms instead of the mathematical terms.

Kernel matrix after SDP for the Math-Med-Finance Data



The block structure is clearly visible, again "`average`" is in the financial terms.

Colors-Nums data set

purple, three, blue, red, eight,
transparent, black, white, small, six,
yellow, seven, fortytwo, five,
chartreuse, two, green, one, zero, orange, four

Colors-Nums data set

`purple, three, blue, red, eight,`
`transparent, black, white, small, six,`
`yellow, seven, fortytwo, five,`
`chartreuse, two, green, one, zero, orange, four`

This data set is from Cilibrasi and Vitányi. "`small`" is supposed to be a number and "`transparent`" a color.

Colors-Nums data set

`purple, three, blue, red, eight,`
`transparent, black, white, small, six,`
`yellow, seven, fortytwo, five,`
`chartreuse, two, green, one, zero, orange, four`

This data set is from Cilibrasi and Vitányi. "`small`" is supposed to be a number and "`transparent`" a color.

Spectral clustering indicates that there are three clusters, "`fortytwo`" forms a singleton group, and "`white`" and "`transparent`" are misclustered as numbers.

Colors-Nums with SDP

Clustering with SDP gives a slightly different result: Here, best results are obtained with $k = 2$ clusters, in which case only "`fortytwo`" is wrongly assigned to the colors.

So, SDP is clearly better here.

People data set

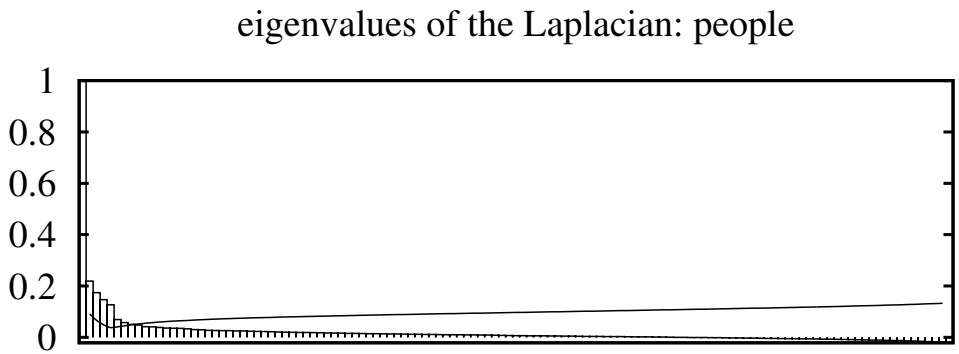
Takemitsu, Stockhausen, Kagel, Xenakis,
Ligeti, Kurtag, Martinu, Berg, Britten,
Crumb, Penderecki, Bartok, Beethoven,
Mozart, Debussy, Hindemith, Ravel,
Schoenberg, Sibelius, Villa-Lobos, Cage,
Boulez, Kodaly, Prokofiev, Schubert,
Rembrandt, Rubens, Beckmann, Botero, Braque,
Chagall, Duchamp, Escher, Frankenthaler,
Giacometti, Hotere, Kirchner, Kandinsky,
Kollwitz, Klimt, Malevich, Modigliani,
Munch, Picasso, Rodin, Schlemmer, Tinguely,
Villafuerte, Vasarely, Warhol, Rowling,
Brown, Frey, Hosseini, McCullough, Friedman,
Warren, Paolini, Oz, Grisham, Osteen,
Gladwell, Trudeau, Levitt, Kidd, Haddon,
Brashares, Guiliano, Maguire, Sparks,

People data set cont.

Roberts, Snicket, Lewis, Patterson, Kostova,
Pythagoras, Archimedes, Euclid, Thales,
Descartes, Pascal, Newton, Lagrange, Laplace,
Leibniz, Euler, Gauss, Hilbert, Galois,
Cauchy, Dedekind, Kantor, Poincare, Godel,
Ramanujan, Wiles, Riemann, Erdos, Thomas Zeugmann, Jan
Poland, Rolling, Stones,
Madonna, Elvis, Depeche, Mode, Pink, Floyd,
Elton, John, Beatles, Phil, Collins,
Toten, Hosen, McLachan, Prinzen, Aguilera,
Queen, Britney, Spears, Scorpions, Metallica, Blackmore,
Mercy

We have five groups of each 25 (more or less) famous people: composers, artists, last year's bestseller authors, mathematicians (including the authors of the present paper), and pop music performers.

Eigenvalue Plot for People data set



The eigenvalue plot shows clearly five clusters.

From the 125 names, 9 were not clustered into the intended groups using the spectral method. The highest number of incorrectly clustered names (4 misclusterings) occurred in the least popular group of the mathematicians (but our two names were correctly assigned).

Spectral Clustering of People data set

We also observed that the spectral clustering gets disproportionately harder when the number of clusters increases: Clustering only the first 50, 75, and 100 names gives 0, 2, and 5 clustering errors, respectively.

We also tried clustering the same data set w.r.t. the Japanese web sites in the Google index, this gave 0, 1, 4, and 16 clustering errors for the first 50, 75, 100, and 125 names, respectively.

SDP Clustering of People data set

Clustering with SDP gives better results here: 0, 0, 1, and 4 clustering errors for the first 50, 75, 100, and 125 names, respectively.

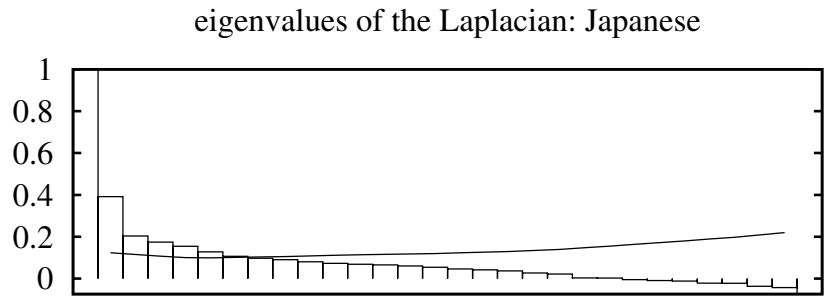
(:)

Japanese Terms

The last data set consists of 20 Japanese terms from finance and 10 Japanese terms from computer science (taken from glossaries):

依頼, 為替, 営業, 円高, 株価, 環境, 金利, 景気, 雇用, 購入, 財政, 株価, 落札, 輸出, 税金, 売上高, 破綻, 流動性, 有価証券, 販路, 回路, 画像処理, 画像圧縮, 関数, 近似, 係数, 形式, 論理, 実験, 算術.

Eigenvalue Plot for Japanese Terms



The eigenvalue plot does not clearly indicate the correct number of $k = 2$ clusters.

Results for Japanese Terms

When using $k = 2$, only the term “環境" (which means “environment") is non-intendedly grouped with the computer science words by the spectral clustering.

Results for Japanese Terms

When using $k = 2$, only the term “環境" (which means “environment") is non-intendedly grouped with the computer science words by the spectral clustering.

But look into GJITEN
Matches in English-main:
環境(かんきょう) (n) environment; circumstance; (P);
Matches in Computer-dict: 環境(かんきょう) environment;
So, maybe our clustering is not wrong either!!

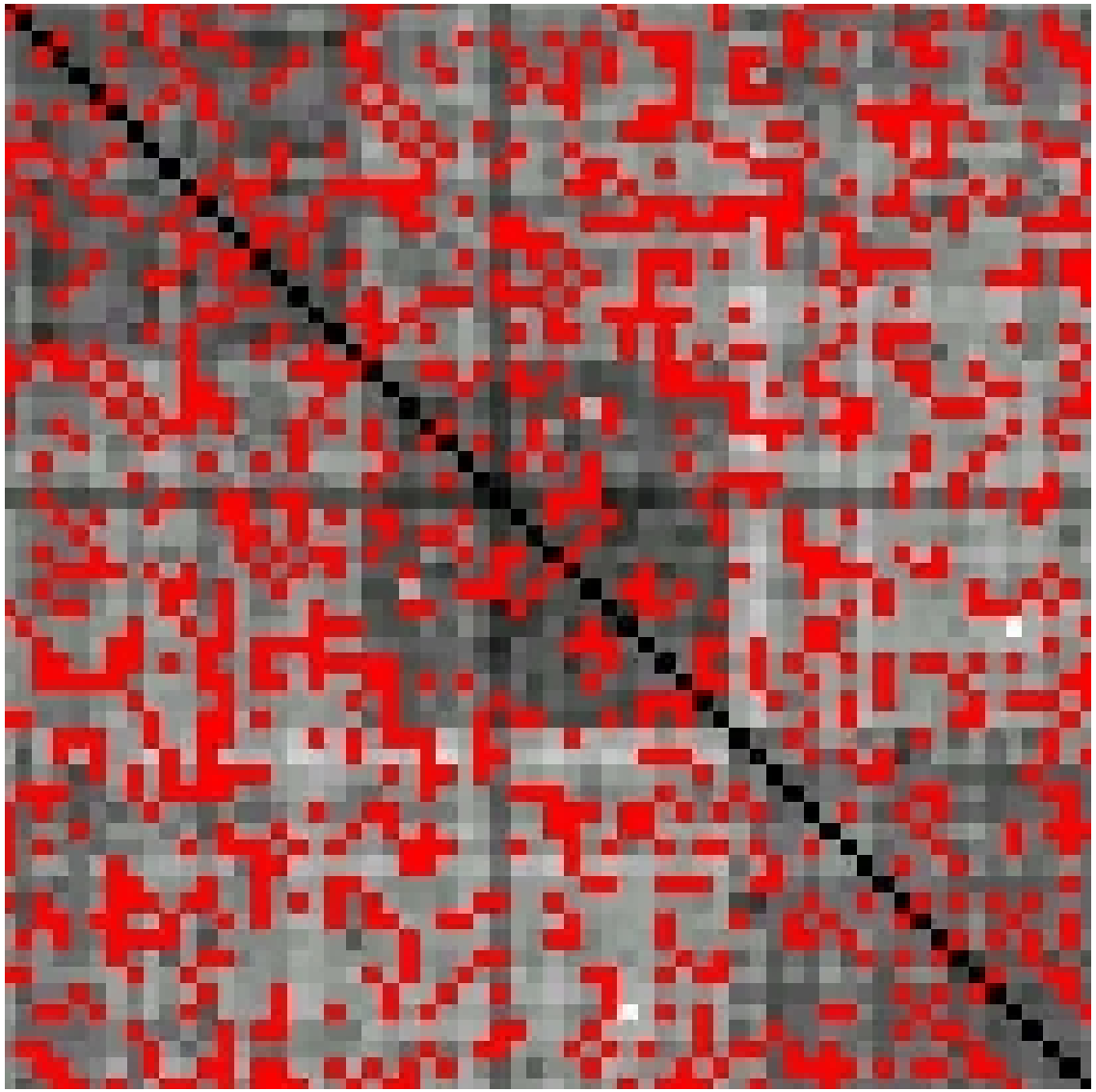
Results for Japanese Terms

When using $k = 2$, only the term “環境" (which means “environment") is non-intendedly grouped with the computer science words by the spectral clustering.

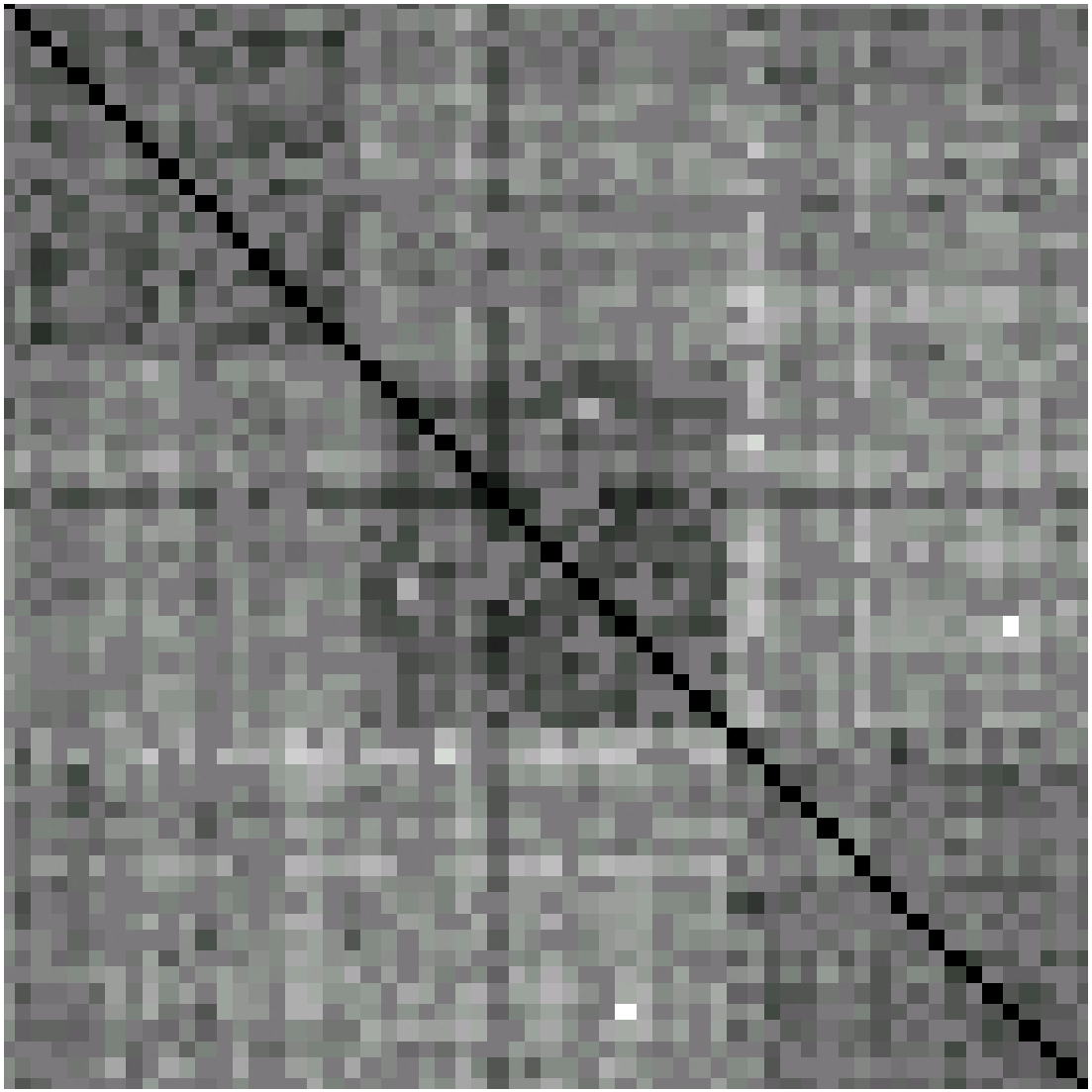
But look into GJITEN
Matches in English-main:
環境(かんきょう) (n) environment; circumstance; (P);
Matches in Computer-dict: 環境(かんきょう) environment;
So, maybe our clustering is not wrong either!!

SDP clustering gives the same result here.

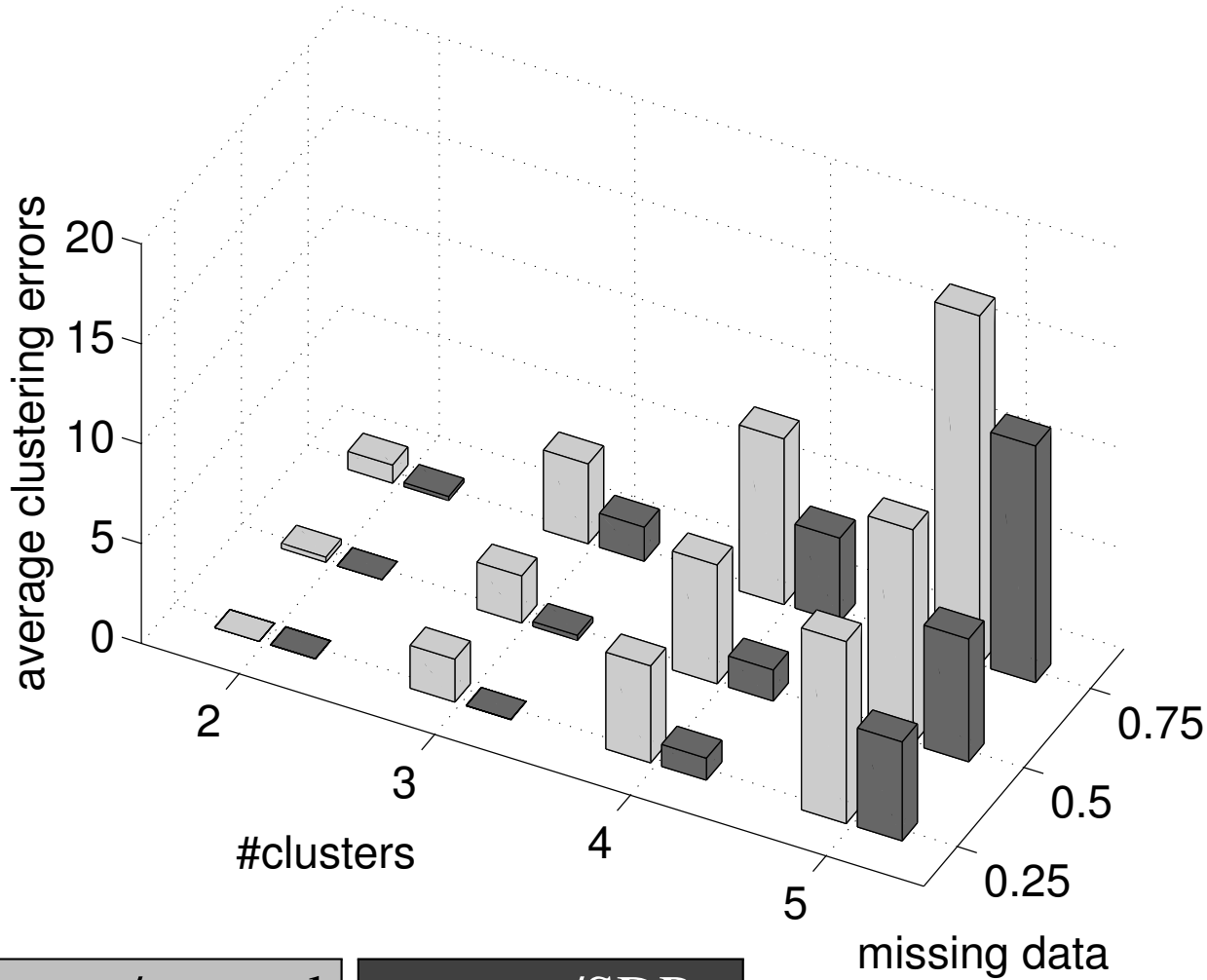
Missing Distances (Math-Med-Finance Data)



Missing Data: Mean Values Substituted



Missing Data: Some Experiments



norm-min-cut/spectral | max-cut/SDP

Conclusions

- Using the Google distance as approximation of the *NID* and some state-of-the-art methods in machine learning, it was possible to automatically separate lists of terms into clusters which make sense. We obtained similar results for the *NCD*.
- Our methods are theoretically quite well founded, basing on the theories of Kolmogorov complexity on the one hand and graph cut criteria and spectral clustering or semidefinite programming on the other hand.
- The SDP clustering is the more direct approach, as it needs less steps. Also, it yields slightly better results. However, it is computationally more expensive than spectral clustering. $\Rightarrow$ Bad if $k \geqslant 4$.
- Both methods are much faster than the computationally expensive phylogenetic trees used by Cilibrasi and Vitányi (2006).

Thank you!