Diagnostics of Semantic Word Spaces

A Thesis about a Barcodes

GABRIEL ISHEDEN

Bachelor’s Thesis at CSC
Supervisors: Hedvig Kjellström
Jussi Karlgren
Mikael Vejdmo-Johansson
Examiner: Mårten Olsson

TRITA xxx yyyy-nn
Abstract

In collaboration with Gavagai, a company that develops automated and scalable methods for retrieving actionable intelligence from dynamic data, I have been studying semantic word spaces and topology. In this bachelor’s thesis, with help from computational topology, I introduce new ways to describe properties of these semantic word spaces, so-called barcodes. I develop a measure to describe barcodes of betti number zero, prove its validity and discuss its implications.
Contents

1 Preliminaries ................................................. 1
   1.1 Introduction ............................................. 1
   1.2 Word Space Models ...................................... 1
   1.3 On High Dimensional Vector Spaces .................... 2
   1.4 Topology ................................................ 3
   1.5 Barcodes ................................................ 5
   1.6 Betti-0 Barcodes ...................................... 6
   1.7 Goal ..................................................... 8

2 The Betti-0 Measure ......................................... 9
   2.1 The Betti-0 Generator Algorithm ....................... 9
   2.2 The Betti-0 Norm ....................................... 12
   2.3 The Reverse Triangle Inequality ....................... 13
   2.4 The K-Measure ......................................... 14

3 Conclusions .................................................. 17
   3.1 Discussion .............................................. 17
   3.2 Future Work ............................................ 19
      3.2.1 Optimizing The Betti-0 Measure .................. 19
      3.2.2 Measuring Higher Order Features ................. 19
   3.3 Summary ................................................. 19

Bibliography .................................................... 21
1. Preliminaries

1.1 Introduction

In machine learning and particularly in computational linguistics we want to analyse big data sets. In linguistics these data sets are in most cases texts, with words and sentences having a significant degree of dependence. To analyse these texts most applications use various kinds of vector space models. These vector space models involve, in most cases, very high dimensional vector spaces. The goal of this thesis is to produce a measure that describes the differences and similarities of two different high dimensional vector spaces.

1.2 Word Space Models

In order to analyse a text in a structured manner a word space model is often applied to produce vector spaces from the texts. These vector space models come in many forms depending on the texts one wants to analyse. Factors that affect the choise of vector space model include the amount of text, computational power available and the type of analysis being conducted.

The most naive approach of creating a word space from a number of texts is the search engine model \[7\] also known as a word frequency table. The search engine model can be describes as follows:

Given \(m\) texts \((T_1, T_2, \ldots, T_m)\). Let all the words that exist in the texts \((T_1, T_2, \ldots, T_m)\) at least once be called \((w_1, w_2, \ldots, w_n)\). We create the matrix \(W = \{W_{ij}\}\) by letting \(W_{ij}\) be the number of occurrences of the word \(w_i\) in the text \(T_j\). This produces a matrix where all the rows represent words and all the columns represent texts. Thus every row is a point in the high-dimensional vector space, and every text represents a basis vector of this vector space.

The search engine model leads to a very big matrix with the number of columns increasing for every new text we process, this leads to an increase of the dimension of the vector space for every text we add to the text mass. Another problem with the naive approach is that the matrix becomes sparse, which may indicate an ineffective representation.

Common methods to reduce the dimension of a word space are different random indexing models. In their paper *Random indexing of text samples for latent semantic*
analysis Kanerva et al. [7] introduce a simple random indexing model for texts. Given \( m \) texts \((T_1, T_2, ..., T_m)\). Let all the words that exist in the texts \((T_1, T_2, ..., T_m)\) at least once be called \((w_1, w_2, ..., w_n)\). Now we give each texts an associated index vector:

\[(T_1, T_2, ..., T_m) \rightarrow (I_1, I_2, ..., I_m)\]

To decrease the dimensions of the matrix we create, we let the dimension of \( I_j \) be smaller than \( m \). A standard procedure is letting the \( I \) vectors have dimension 1000. The vectors \( I_j \) is then randomly indexed with 10 indices of ones and 10 indices of minus ones, the rest of the indexes are set to zero. To represent the given texts in a vector space, we start off with a zero matrix \( W = \{W_{ij}\} \) with \( n \) rows and \( m \) columns. The next step is to read every text. For every occurrence of word \( w_i \) in text \( T_j \) we add the index vector \( I_j \) to row \( i \) in \( W \).

This method provides a non-sparse and significantly smaller word matrix. The matrix’ dimensions are also fixed independently of the number of texts we read, with only the number of vectors (new words represented by rows in \( W \)) increasing.

1.3 On High Dimensional Vector Spaces

The main focus of the vector space models is to preserve as much meaning or information as possible from the texts you want to analyse. When doing this, you will often end up in an extremely high dimensional vector space, with a dimension of several hundred, sometimes even reaching the thousands.

The most common problems you encounter when dealing with high dimensional vector spaces are big calculations and certain phenomenons usually called the curse of dimensionality.

The curse of dimensionality states that when the dimension of a vector is large, a proximity query becomes unstable. The intuitive way of understanding this concept is by considering two random points. Since the dimensionality is big, most randomly chosen pair of points will be equally far away from one another with the angle between them and the origo being 90 degrees.

The curse of dimensionality was described in a mathematical way by Beyer et al. in *When is Nearest Neighbors Meaningful?* with the following theorem theorem [6]:

**Theorem 1.3.1 (Beyer et al.)** (Simplified for euclidean distance metrix \(||·||\))

\[
\lim_{d \rightarrow \infty} \text{var} \frac{||X_d||}{E[||X_d||]} = 0 \Rightarrow \frac{D_{\text{max}} - D_{\text{min}}}{D_{\text{min}}} \rightarrow 0
\]

This means that if the variance of \( n \) \( d \)-dimensional stochastic variable is 0 as \( d \rightarrow \infty \), the difference between the furthest and nearest point tends increases slowly.

Similar results have been shown when working with word spaces. In *Filaments of Meaning in Word Space* Karlgren et al. [8] shows that semantic high dimensional
word spaces behave in a similar fashion. If we look at word spaces with three dimensions, the distribution of the angle from the origin will be even. If we increase this dimension further, the distribution will take on the appearance of a peak to a higher degree.

![Figure 1.1. (From Filaments of Meaning in Word Space, by Karlgren et al. [8]. This figure shows the behavior of the angle to points with respect to the origin for randomly chosen points. A similar behavior can be observed for word spaces.)](image)

### 1.4 Topology

The data being analysed is represented as point cloud data - an unordered sequence of points or vectors in an n-dimensional vector space. Standard methodology to analyse low dimensional point cloud data with machine learning algorithms involve, among others, support vector machines (SVM) and principal component analysis (PCA). When the point cloud data is high dimensional however, standard machine learning algorithm fall short. To address some of these problems a new field of machine learning has evolved, topological data analysis.

The foundation behind topological data analysis was first introduced by Edelsbrunner et al. [4]. This work and the work of other contributors have recently been reviewed and further developed by Carlsson [3].

At its most basic, the idea behind topological data analysis is to make a graph out of the point cloud data. The common approach to make a graph out of the point cloud data is to divide the point cloud data into simplicial complexes. The definition of a simplicial complex is cited from [9].

**Definition 1.4.1** A simplicial complex is a finite collection of simplices $K$ such that $\sigma \in K$ and $\tau \leq \sigma$ implies $\tau \in K$, and $\sigma_i, \sigma_j \in K$ implies $\sigma_i \cap \sigma_j$ is either empty or a face of both.

A practical method of dividing the point cloud data into simplicial complexes is to use so called the Čech complex or the Vietoris-Rips complex, also called the Rips complex. The most commonly used of these simplicial complexes is the Rips
complex [5]. The definition of the Čech complex and Vietoris-Rips complex is cited from [5].

**Definition 1.4.2** Given a collection of points \( \{x_\alpha\} \) in Euclidean space \( E^n \), the Čech complex, \( C_\epsilon \), is the abstract simplicial complex whose k-simplices are determined by unordered \((k + 1)\)-tuples of points \( \{x_\alpha\}_0^k \) whose closed \( \epsilon/2 \)-ball neighborhoods have a point of common intersection.

**Definition 1.4.3** Given a collection of points \( \{x_\alpha\} \) in Euclidean space \( E^n \), the Rips complex, \( R_\epsilon \), is the abstract simplicial complex whose k-simplices correspond unordered \((k + 1)\)-tuples of points \( \{x_\alpha\}_0^k \) which are pairwise within distance \( \epsilon \).

For ease of computation, the Rips complex is normally used when producing simplicial complexes.

![Figure 1.2.](From Barcodes: the persistent topology of data, by Robert Ghrist [5]. The picture is used with his permission.) From a fixed set of points, the Čech complex \( C_\epsilon \) and Rips complex \( R_\epsilon \) can be completed for a given value of \( \epsilon \). The difference between the two is depicted to the bottom left (Čech) and bottom right (Rips).

When dealing with point cloud data represented as a graph, another question arises, what should our choice of parameter \( \epsilon \) be? For smaller values of \( \epsilon \) the graph will be a discrete set of points. On the other hand if \( \epsilon \) is large enough the graph will become
1.5. BARCODES

a single high dimensional simplex containing all points. It would seem practical to find a values of $\epsilon$ that reveal as many features of the point cloud data as possible.

This problem was addressed by Edelsbrunner et al. [4] with the introduction of persistence homology. Instead of choosing a single value of $\epsilon$, we let $\epsilon$ start from zero and increase. In persistent homology we look at topological features that persists over significant values of $\epsilon$ and let those features that only show for a short time be considered noise.

As an example, cited from Ghrist [5]: Let $R = (R_i)_{i=1}^N$ be a sequence of Rips complexes associated to a fixed point cloud for an increasing sequence of $\epsilon$ values $(\epsilon_i)_{i=1}^N$. There are natural inclusion maps such that:

$R_1 \xrightarrow{\iota} R_2 \xrightarrow{\iota} \ldots \xrightarrow{\iota} R_N$

Instead of looking at individual Rips complexes $R_i$, we consider at the iterated inclusion maps $\iota : H_\ast R_i \rightarrow H_\ast R_j$ for $i < j$. These inclusion maps reveal which features persist.

1.5 Barcodes

In order to capture the the topological features found in the point cloud data, a compact way of describing them is required. The features usually considered are the so called Betti numbers and we describe them with barcodes. Definition, both formal and formal, from wikipedia [1]:

**Definition 1.5.1** For a non-negative integer $k$, the $k$:th Betti number $b_k(X)$ of the space $X$ is defined as the rank of the abelian group $H_k(X)$, the $k$:th homology group of $X$.

**Informal Definition 1.5.2** Informally, the $k$:th Betti number refers to the number of unconnected $k$:dimensional surfaces. The first few Betti numbers have the following intuitive meaning:
- $b_0$ is the number of connected components
- $b_1$ is the number of one-dimensional or circular holes
- $b_2$ is the number of two-dimensional holes or voids

To produce barcodes we look at the Rips complex $R_\epsilon$ where epsilon starts from zero. We then make a kind of a bar graph with a bar for each component of each betti number on the y-axis and the value of $\epsilon$ on the x-axis. The starting point of every bar is the value of $\epsilon$ when the component first appears and the end point is value of $\epsilon$ when the component stops to appear.
The calculate the three first betty numbers of this torus, we count the number of connected components, the number of circular holes and the number of voids. This yealds $b_0 = 1$, $b_1 = 2$, $b_2 = 1$, since there is one big component, two circular holes (one in the middle and one around the 'tube'), and a void (inside the 'tube')

1.6 Betti-0 Barcodes

To understand how the barcodes with betti number zero behaves we have to look at them in a more structured manner. How do we produce betti-0 barcodes?

Consider a point cloud $P$ with $n$ different points and the Rips complex $R_\epsilon$. We let all the points in $P$ be labeled $(P_1, P_2, ..., P_n)$.

When we produce betti-0 barcodes we start off with $\epsilon = 0$, this menas that the betti-0 barcodes will start with $n$ bars, labeled from 1 to $n$. We then start increasing $\epsilon$. As $\epsilon$ increases edges will start appearing in the graph and components will start to connect with one another. This means that when there are no connecting components, we will start off with $n$ barcodes. When $\epsilon$ increases and the edges start appearing, two components will merge into one component. When two components merge one of the bars will end. By convention we let the bar with the highest index end. $\epsilon$ will then increase till there is one bar left. This bar will continue for infinitely big values of $\epsilon$. 
Another way to look at $P$ is with the following notation:

**Notation 1.6.1** Consider the point cloud $P$ with $n$ different points labeled $(P_1, P_2, ..., P_n)$ and a parameter $\epsilon$. Consider the complete graph of $P$ and remove all the edges that are greater than $\epsilon$.

We denote by $S_\epsilon = \{s_i\}$ the family of all the spanning trees after the edges greater than $\epsilon$ have been removed.

We let $s_i$ denote a spanning tree and let the number $s_i$ be the $P_i$ with the maximum index in that tree. See figure 1.4 for an example of $S_\epsilon$.

When we have defined $S_\epsilon$ we can look at the spanning trees in $S_\epsilon$ as $\epsilon$ increases. Every spanning tree in $S_\epsilon$ represents one connecting component each, thus as $\epsilon$ increases the numbers $s_i$ still in $S_\epsilon$ are the bars still growing and the numbers from 1 to $n$ that are not represented in $S_\epsilon$ are bars that have ended.
Figure 1.5. In this example let $P$ be the 6 points above. Here $S_{c=0} = \{1, 2, 3, 4, 5, 6\}$, $S_{c=1} = \{1, 2, 5, 6\}$ and $S_{c=2} = \{1\}$

1.7 Goal

The goal of this project is to address questions about the properties of a word space $P$.

Can I say something about the properties of $P$ by looking at its betti-0 barcodes?

In my bachelor’s thesis I will develop tools to find and point out properties of the word space $P$. My focus will be on trying to find ways to describe betti-0 barcodes by defining a norm and a measure to measure properties of $P$. 
2. The *Betti-0* Measure

In this section I will merely tell what definitions I make and what theorems I prove. For discussion about why I chose the definitions, norms and measures I did see chapter 3, Discussion and Summary

2.1 The *Betti-0 Generator* Algorithm

**Definition 2.1.1** Given a point cloud \(P = \{P_i\}\) with \(n\) points and a parameter \(\epsilon \geq 0\). Let the spanning trees of \(S_\epsilon\) be labeled \(S_\epsilon = \{s_i\}_{i=1}^{k}\) (see notation 1.6.1).

The extended distance matrix \(M_P^{(\epsilon)}\) is a matrix \(M_P^{(\epsilon)} = \{M_{ij}\}\) where \(0 \leq i, j \leq k\) and \(M_{ij}\) has the following properties:

1. If \(i = j\) then \(M_{ij} = 0\)
2. If \(i \neq j\) and \(\min(i, j) = 0\) then \(M_{ij} = s_{\max(i,j)}\)
3. If \(i \neq j\) and \(\min(i, j) > 0\) then \(M_{ij} = \min_{x_1, x_2} d(x_1, x_2)\) \(x_1 \in s_i, x_2 \in s_j\)

**Notation 2.1.2** Given a point cloud with \(n\) points \(P = \{P_i\}\) where \(P_i\) are vectors. Let \(B_P = (b_1, b_2, \ldots, b_n)\) denote the length of each bar from the Betti-0 barcodes generated from the point cloud \(P\).

Example:
Consider a \(P\) as in figure 1.4.

\[
M_P^{(\epsilon=0)} = \begin{pmatrix}
0 & 1 & 2 & 3 & 4 & 5 & 6 \\
1 & 0 & \sqrt{17} & 1 & \sqrt{2} & \sqrt{41} & 5 \\
2 & \sqrt{17} & 0 & 4 & 5 & 4 & 4\sqrt{2} \\
3 & 1 & 4 & 0 & 1 & 4\sqrt{2} & 4 \\
4 & \sqrt{2} & 5 & 1 & 0 & \sqrt{41} & \sqrt{17} \\
5 & \sqrt{41} & 4 & 4\sqrt{2} & \sqrt{41} & 0 & 4 \\
6 & 5 & 4\sqrt{2} & 4 & \sqrt{17} & 4 & 0
\end{pmatrix}
\]

\[
M_P^{(\epsilon=1)} = \begin{pmatrix}
0 & 1 & 2 & 5 & 6 \\
1 & 0 & 4 & 4\sqrt{2} & 4 \\
2 & 4 & 0 & 4 & 4\sqrt{2} \\
5 & 4\sqrt{2} & 4 & 0 & 4 \\
6 & 4 & 4\sqrt{2} & 4 & 0
\end{pmatrix}
\]
And $M_P^{(\epsilon=5)} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$

To simplify several steps in my proof of the reverse triangle inequality I have developed an algorithm to generate Betti-0 barcodes. I have chosen to call this algorithm Betti-0 Generator. Betti-0 Generator will determine the value of each of the scalars $b_i$ in $B_P$. Betti-0 Generator takes as input the extended distance matrix $M_P^{(\epsilon)}$ with $k$ spanning trees and gives as output an extended distance matrix of $k-1$ spanning trees and the length of a specified bar. Betti-0 Generator ends when all the points in $P$ have entered the same spanning tree, that is when the extended distance matrix $M_P^{(\epsilon)}$ has only one spanning tree.

**Step one:**

The goal of the first step is to determine which spanning tree $s_\alpha$ will connect to another spanning tree $s_\beta$ first as $\epsilon$ increases. To do this we find the smallest element $M_{\alpha \beta}$ where $\alpha \neq \beta$ and $1 \leq \alpha, \beta$. Such an element always exists when the number of spanning trees is greater than one. We also note that $M_{\alpha \beta} = M_{\beta \alpha}$. Now we let $b_{\max(s_\alpha, s_\beta)} = M_{\alpha \beta}$.

Note: if there are other elements $M_{\gamma \delta}$ such that $M_{\gamma \delta} = M_{\alpha \beta}$ we can choose any of the smallest elements arbitrarily.

**Step two:**

The goal of the second step of the algorithm is to reduce $M_P^{(\epsilon)}$ to $M_P^{(\epsilon+1)}$ where $\epsilon + 1 = M_{\alpha \beta}$. To do this we consider the rows $\{M_{\alpha i}\}$, $\{M_{\beta i}\}$ and the columns $\{M_{i \alpha}\}$ and $\{M_{i \beta}\}$. To preserve properties (1) – (3) in definition 2.1.1 as $M_P^{(\epsilon)} \rightarrow M_P^{(\epsilon+1)}$, we need to handle $M_P^{(\epsilon)}$ in a proper fashion. Without loss of generality I will assume that $\beta > \alpha$ throughout this step.

To preserve property (3) of an extended distance matrix, we will do the following: For every $1 \leq i \leq k$ we let $M_{\alpha i} = \min(M_{\alpha i}, M_{\beta i})$ and $M_{i \alpha} = \min(M_{i \alpha}, M_{i \beta})$. Since the spanning trees $s_\alpha$ and $s_\beta$ connect when $\epsilon + 1 = M_{\alpha \beta}$ we label the new spanning tree $s_{\alpha \beta}$ and let the distances to every other spanning tree be the minimum of the distances from $s_\alpha$ and $s_\beta$ to every other spanning tree.

We then define $M_P^{(\epsilon+1)} = \{M'_{ij}\}$ by letting:

- $\{M'_{ij}\} = \{M_{ij}\}$ for $0 \leq i, j \leq (\beta - 1)$
- $\{M'_{ij}\} = \{M_{i(\beta+1)}\}$ for $\beta \leq i \leq (k-1)$ and $0 \leq j \leq (\beta - 1)$
- $\{M'_{ij}\} = \{M_{(\beta+1)j}\}$ for $0 \leq i \leq (\beta - 1)$ and $\beta \leq j \leq (k-1)$
- $\{M'_{ij}\} = \{M_{(\beta+1)(j+1)}\}$ for $\beta \leq i \leq (k-1)$ and $\beta \leq j \leq (k-1)$
2.1. THE BETTI-0 GENERATOR ALGORITHM

Now since (1) and (2) holds for $M^t$ and we have only removed row and column number $\beta$, (1) and (2) holds for $M^{t+1}$.

Now, if the number of spanning trees in $S_\epsilon$ is greater then one, repeat from step one.
If there is only one spanning tree in $S_\epsilon$ we move on to step three in the algorithm.
We note that the algorithm will move on to step three after $n - 1$ steps.

Step three:

When $\epsilon$ reaches a value such that $M_\epsilon$ consists of only one spanning tree, an increase $\epsilon$ will not change $M_\epsilon$ no matter how big $\epsilon$ gets. This means that the final bar $b_1$ should be infinitely long, $b_1 = \infty$. After this step we have produced all $n$ barcodes for $B_P$.

Something to notice is that if there are several values in $M_\epsilon$ that are equal, the $\epsilon$ will not increase when step two in the process is redone.

This algorithm also gives us a quite nice lemma.

**Lemma 2.1.3** If the spanning trees $s_\alpha$ and $s_\beta$ in $M^{t_k}_P = \{M_{ij}\}$ and $s_{\alpha'}$ and $s_{\beta'}$ in $M^{t_{k+1}}_P = \{M'_{ij}\}$ are such that $s_\alpha = s_{\alpha'}$ and $s_\beta = s_{\beta'}$ then:

$$M_{\alpha'\beta'}' \leq M_{\alpha\beta}$$

Proof:

If the spanning trees named $s_\alpha$ and $s_\beta$ in $M^{t_k}_P$ still exist in $M^{t_{k+1}}_P$ then the smallest distance in $M^{t_k}_P$ is not a distance between $s_\alpha$ or $s_\beta$ and any other point ($\ast$). Let the distance between two spanning trees in $M^{t_k}_P$ be called $M_{m,n}$.

If $m \neq \alpha$ and $n \neq \beta$ or $n \neq \alpha$ and $m \neq \beta$ then $M_{m,n}$ is not affected when two spanning trees connect, which means that $M_{\alpha\beta} = M_{\alpha'\beta'}'$.

Now assume that one of the indexes is equal to another. Lets assume $m = \alpha$, $n = \beta$.
This menas that:

$$M_{\alpha'\beta'} = \min(M_{\alpha\beta}, M_{m\beta}) = \min(M_{m\beta}, M_{n\beta}) \leq M_{m\beta} = M_{\alpha\beta}$$

This shows the lemma for $m = \alpha$, $n \neq \beta$. If one of the indexes are equal, the argument is similar. Since ($\ast$) is true, both $m = \alpha$ and $n = \beta$ will not be true.

**Proposition 2.1.4** If the spanning trees $s_\alpha$ and $s_\beta$ in $M^{t_1}_P = \{M_{ij}\}$ and $s_{\alpha'}$ and $s_{\beta'}$ in $M^{t_2}_P = \{M'_{ij}\}$ are such that $s_\alpha = s_{\alpha'}$, $s_\beta = s_{\beta'}$ and $\epsilon_1 < \epsilon_2$ then:

$$M_{\alpha'\beta'}' \leq M_{\alpha\beta}$$
Proof:

By induction, proposition 2.1.4 follows from lemma 2.1.3.

To illustrate how the algorithm works we will look at an example of a step in the process:

Assume that

\[ M^{(\epsilon=1)}_P = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 3 & 5 \\ 2 & 3 & 0 & 4 \\ 3 & 5 & 4 & 0 \end{pmatrix} \]

The minimum of \( M^{(\epsilon=1)}_P \) is \( M_{23} = 3 \). This means that \( b_2 = 3 \). Now as \( \epsilon \to 3 \) we compare row 2 and 3 and replace \( M_{2i} \) with \( \min(M_{2i}, M_{3i}) \), the same thing is done with column 2 and 3, \( M_{i2} \) is replaced with \( \min(M_{i2}, M_{i3}) \). We then remove row and column number 3:

\[ M^{(\epsilon=1)}_P \rightarrow \begin{pmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 0 & 4 \\ 2 & 0 & 0 & 4 \\ 3 & 4 & 4 & 0 \end{pmatrix} \rightarrow M^{(\epsilon=3)}_P = \begin{pmatrix} 0 & 1 & 3 \\ 1 & 0 & 4 \\ 3 & 4 & 0 \end{pmatrix} \]

We also note that in line with lemma 2.1.3 the distance between spanning tree 1 and 3 in \( M^{(\epsilon=3)}_P \) is less than or equal the distance between the trees in \( M^{(\epsilon=1)}_P \).

### 2.2 The Betti-0 Norm

**Definition 2.2.1** Given a point cloud \( A \) with \( n \) points and its Betti-0 barcode \( B_A = (b_1, b_2, \ldots, b_n) \). Let \( \| A \|_{B_0} \) be defined by:

\[ \| A \|_{B_0} = \sum_{i=1}^{n} \left( \frac{1}{b_i} \right) \]

Throughout this paper we will write \( \| \cdot \| \) instead of \( \| \cdot \|_{B_0} \)

**Lemma 2.2.2 (Positivity)** Given a point cloud \( A \) consisting of \( n \) different points, then \( \| A \| \geq 0 \) with \( \| A \| = 0 \) if and only if \( A \) consists of only one random point.

Proof:

If \( A \) consists of only one point we get that as \( \epsilon \) increases \( M^{(\epsilon)}_A \) will contain only one spanning tree independent of the value of \( \epsilon \) implying that \( B_A \) will consist of only one bar with \( b_1 = \infty \). From definition 2.2.1 we get \( \| A \| = 0 \).

Now assume that \( A \) consists of the two different points \( a_1 \) and \( a_2 \). Now as \( \epsilon \to \)
d(a_1, a_2) the spanning tree a_1 will connect to the spanning tree a_2 producing a bar with length d(a_1, a_2). This gives us B_A = (\infty, d(a_1, a_2)).

Definition 2.2.1 implies:

\[ ||A|| = (\frac{1}{\infty}) + \frac{1}{d(a_1, a_2)} = \frac{1}{d(a_1, a_2)} > 0 \]

Now by induction we have that if A consists of n different points then ||A|| > 0

2.3 The Reverse Triangle Inequality

**Theorem 2.3.1** Given two disjoint point clouds, K with m points and L with n points, then:

\[ ||K|| + ||L|| \leq ||K \cup L|| \]

To prove the theorem above we will choose a proper labeling for the points in K and L and then produce the barcodes \( B_K = (k_1, k_2, .., k_m) \), \( B_L = (l_1, l_2, .., l_n) \) and \( B_{K \cup L} = (b_1, b_2, .., b_{m+n}) \). We will then show that:

\[ (*) \quad b_i \leq k_i \text{ for } 1 \leq i \leq m \text{ and } b_{m+i} \leq l_i \text{ for } 1 \leq i \leq n \]

If we show (*) we see from definition 2.2.1 that we are done with the proof.

**Proof:**

We start this proof by giving labels to all the points in K, L and K \( \cup \) L. The points in K we call \( (s_{k_1}, s_{k_2}, .., s_{k_m}) \), the points in L we call \( (s_{l_1}, s_{l_2}, .., s_{l_n}) \) and the points in K \( \cup \) L we call \( (d_1, d_2, .., d_{m+n}) \). We label the points in an order such that:

The point labeled \( s_{k_i} \) in K is the same point as the point labeled \( d_i \) in K \( \cup \) L.

The point labeled \( s_{l_i} \) in L is the same point as the point labeled \( d_{m+i} \) in K \( \cup \) L.

We will now start creating the barcodes \( B_K = (k_1, k_2, .., k_m) \), \( B_L = (l_1, l_2, .., l_n) \) and \( B_{K \cup L} = (b_1, b_2, .., b_{m+n}) \) simultaneously using Betti-0 Generator with all three algorithms using the same \( \epsilon \).

We will now show (*) by assuming, without loss of generality, that i is the index where \( k_i \) is the first bar created such that such that:

\[ (1) \quad b_i > k_i \]

This happens when \( \epsilon \) goes from \( \epsilon_0 \rightarrow \epsilon_1 \) (\( = k_i \)). We will assume that the bar \( k_i \) is created when the spanning tree \( s_i \) connects to the spanning tree \( s_\alpha \), which means that \( s_\alpha < s_i \). Since the bar \( k_i \) is created as \( \epsilon_0 \rightarrow \epsilon_1 \), \( k_i \) is the smallest element in \( M^{\epsilon_0}_K \) in 'Step one' of Betti-0 Generator.

Now consider the spanning tree \( d_\alpha \) in \( M^{\epsilon_0}_{K \cup L} \), containing the spanning tree called \( s_\alpha \).
in $M^0_K$. The spanning tree $s_i$ in $M^0_K$ is called $d_i$ in $M^0_{K∪L}$, from the labeling we did initially. The spanning tree $d_i$ exists in $M^0_{K∪L}$, because if it had already connected to another spanning tree with label smaller than $d_i$ (1) would not be true. We have that the labels $d_{α'} < d_i$, since $s_α < c_i$

Since $s_i$ is the first spanning tree that produces a barcode such that $b_i > k_i$, at least all the points in the spanning tree $s_i$ in $M^0_K$ are contained within the spanning tree called $d_i$ in $M^0_{K∪L}$. With the same reasoning we get that at least all points contained within $s_α$ in $K$ are contained within the spanning tree $d_{α'}$ in $M^0_{K∪L}$.

Let us assume that distance between the spanning tree called $s_i$ and $s_α$ in $M^0_K$ is the distance between the two points initially called $s_k$, and $s_k$. Since the point initially called $s_k$ is contained within $s_i$ and the point initially called $s_k$ is contained within $s_α$, the point initially called $d_k$ is contained within $d_i$ and the point initially called $d_k$ is contained within $d_{α'}$.

But the distance between the points $s_k$ and $s_k'$ is the same as the distance between the points $d_k$ and $d_k'$, which means that the distance between spanning trees $d_{α'}$ and $d_i$ is at most the distance between the points $d_k$ and $d_k$. Since the bar $k_i$ is created when $ε_0 → ε_1$, the distance between the points $c_k$ and $c_k'$ in $M^0_K$ is $ε_1$ which implies that that the distance between the spanning trees $d_{α'}$ and $d_i$ is at most $ε_1$.

But since the distance between the spanning trees $d_{α'}$ and $d_i$ is at most $ε_1$, the bar $b_i$ is created at latest when $ε_0 → ε_1$, which means that $b_i ≤ ε_1$. Now since $k_i = ε_1$ and (1) $b_i > k_i$ we have:

$$b_i ≤ ε_1 ⇒ ε_1 ≥ b_i > k_i = ε_1 ⇒ ε_1 > ε_1$$

Which is a contradiction.

Since we have only assumed (1) and got a contradiction, we get that $b_i ≤ k_i$ for every $i$. We have now shown (*), which proves the reverse triangle inequality for the Betti-0 Norm.

### 2.4 The $K$-Measure

We have now proven the main result of this thesis, theorem 2.3.1. The reverse triangle inequality gives us a chance to create a sort of measure. We start off by noting that since

$$||A|| + ||B|| ≤ ||A ∪ B||$$

is true, we have by induction that $A_i ∩ A_j = ∅ ∀i, j ∈ \{1, 2, .., n\}$ implies that

$$\sum_{k=1}^{n} ||A_k|| ≤ ||\bigcup_{k=1}^{n} A_k||$$

By dividing the extension above by its right hand side, we end up in:

$$\frac{\sum_{k=1}^{n} ||A_k||}{||\bigcup_{k=1}^{n} A_k||} ≤ 1$$
2.4. THE K-MEASURE

We note that lemma 2.2.2 implies that:

\[ 0 \leq \frac{\sum_{k=1}^{n} ||A_k||}{||\bigcup_{k=1}^{n} A_k||} \leq 1 \]

We can now make the following definition:

**Definition 2.4.1** The K-measure \( K(A_1, A_2, \ldots, A_n) \) of \( n \) disjoint point clouds \( (A_1, A_2, \ldots, A_n) \) is defined as:

\[ K(A_1, A_2, \ldots, A_n) = \frac{\sum_{k=1}^{n} ||A_k||}{||\bigcup_{k=1}^{n} A_k||} \]

From the result above we also get

**Theorem 2.4.2** Given \( n \) disjoint point clouds \( (A_1, A_2, \ldots, A_n) \), then:

\[ 0 \leq K(A_1, A_2, \ldots, A_n) \leq 1 \]

Another important property is that since addition of scalars and the union operation are commutative, the k-measure is also unchanged under permutation of \( A_i \):

\[ K(A_1, A_2, \ldots, A_i, \ldots, A_j, \ldots, A_n) = K(A_1, A_2, \ldots, A_j, \ldots, A_i, \ldots, A_n) \]
3. Conclusions

3.1 Discussion

The goal of the K-measure is to be able to ask the following question. Assume that $P$ is a point cloud consisting of two separate clusters. We think that $A$ and $B$ are these two clusters of $P$, such that $A \cup B = P$ and $A \cap B = \emptyset$, but how can we be sure that $A$ and $B$ really are the clusters in $P$?

I will now argue, with two examples, that we might be able to measure this with the K-measure.

Example one:

Assume that $P$ is a point cloud consisting of two separate clusters $X$ and $Y$.

Assume that the pairwise distance between every point in $X$ is less than $\rho$ and that the pairwise distance between every point in two points in $Y$ is less than $\rho$. Assume that the minimal distance between two points in $X$ and $Y$ is $\delta$ such that $\delta > \rho$.

Assume that the labels on the points in $X$ are $(X_1, X_2, ..., X_m)$ and the labels on the points in $Y$ are $(Y_1, Y_2, ..., Y_n)$. This labeling is done such that the distance between

\[ Figure 3.1. \] Two point clouds $X$ [to the left] and $X$ [to the right] separated by a distance $\delta$. 

$X_1$ and $Y_1$ is the minimal distance between $X$ and $Y$. Let the barcodes produced from $X$ and $Y$ be $B_X = (x_1, x_2, ..., x_m)$ and $B_Y = (y_1, y_2, ..., y_n)$. Also let the points in $P = (P_1, P_2, ..., P_n, P_{n+1}, ..., P_{m+n})$ be labeled such that $P_i = X_i$ for $1 \leq i \leq n$ and $P_{n+j} = Y_j$ for $1 \leq j \leq m$. We also have $B_P = (p_1, p_2, ..., p_n, p_{n+1}, ..., p_{m+n})$. With this labeling we get for $B_X$ that $x_1 = \infty$ and $x_i = p_i$ for $2 \leq i \leq m$ and for $B_Y$ that $y_1 = \infty$ and $x_j = p_{j+m}$ for $2 \leq j \leq n$. We also have $p_1 = \infty$ and $p_{n+1} = \delta$.

Now that happens with the K-measure as $\delta$ goes to infinity?

\[
K(X, Y) = \frac{\sum_{i=1}^{n} \frac{1}{x_i^2} + \sum_{j=1}^{m} \frac{1}{y_j^2}}{\sum_{k=1}^{m+n} \frac{1}{p_k}} = \{ \sum_{i=2}^{n} \frac{1}{x_i^2} + \sum_{j=2}^{m} \frac{1}{y_j^2} = \sum_{k=2}^{m} \frac{1}{p_k} + \sum_{k=(m+2)}^{m+n} \frac{1}{p_k} = S \} = \frac{1}{\infty} + \frac{1}{\infty} + S = S
\]

Thus

\[
\delta \to \infty \Rightarrow K(X, Y) \to 1
\]

This result implies that the more differentiated the clusters are, the closer the K-measure will tend to 1.

Example two:

Assume that $P = (P_1, P_2, ..., P_n, P_{n+1})$ is a random point cloud with $n + 1$ points. How do we divide $P$ into $n$ subsets $(A_1, A_2, ..., A_n)$ so that every subset consists of at least one point to maximize or minimize $K(A_1, A_2, ..., A_n)$?

From the pigeon hole principle at least one subset will consist of at least two points. Since all the other subsets have to be non-empty all the subsets will be of one point each, except for one subset consisting of two points. With these subsets we get

\[
K(A_1, A_2, ..., A_n) = \frac{1}{|A_1 \cup A_2 \cup \ldots \cup A_n|}
\]

Where $b$ is the length of the bar in the subset with two points that is not infinite. How do we maximize this $K(A_1, A_2, ..., A_n)$? The answer is by letting the subset consisting of two points be the two points in $P$ that are closest to one another. On the other hand $K(A_1, A_2, ..., A_n)$ is minimized when we choose the subset with two points as the two points in $P$ with the largest distance.

These two examples may indicate a property of the K-measure that I have not been able to show mathematically. The property I have been trying investigate is illustrated with the following problem.
3.2 FUTURE WORK

Problem 3.1.1 Assume that \( P = (P_1, P_2, \ldots, P_n) \) is a point cloud with \( n \) points. Assume that we divide \( P \) into subsets \( A = (A_1, A_2, \ldots, A_k) \) where \( \bigcup_{i=1}^{k} A_i = P \). What division into subsets maximize and minimize \( K(A_1, A_2, \ldots, A_k) \)?

I propose that the K-measure is maximized when \( A_i \) are the clusters of \( P \). Even though this may be wrong, the reversed triangle inequality for the betti-0 norm and defined a measure still hold, and they may in themself contain some information about the properties of \( P \).

3.2 Future Work

3.2.1 Optimizing The Betti-0 Measure

Given a point cloud \( P \) and a division of \( P: P = A_1 \cup A_2 \cup \ldots \cup A_k \) such that \( A_i \cap A_j \) is empty.

How is this division into subsets done so that \( K(A_1, A_2, \ldots, A_k) \) is maximized or minimized? I have argued in this thesis that the K-measure may be maximized if the division of \( P \) is done with \( A_i \) as clusters. I have not proven this mathematically and it may not be true mathematically however. The question remains open.

What division into subsets minimize the k-measure? What properties does the division that maximize or minimize the K-measure have?

3.2.2 Measuring Higher Order Features

The question of finding meaning from or differentiating between barcodes of higher betti numbers still remain.

3.3 Summary

In this bachelor thesis, I have developed a measure with the aim to differentiate between clustered and non-clustered subsets. I have proven the reversed triangle inequality for the norm I have defined and proven that my measure gives values between 0 and 1 as output. The true meaning of the measure I have created is not clear in a mathematical sense, however initial examples indicate that this measure may measure clustering.
Bibliography


