Two-Way Incremental Seriation in the Temporal Domain with Three-dimensional Visualization: Making Sense of Evolving High-Dimensional Data Sets

Peter Wittek∗

Abstract

Two-way seriation is a popular technique to analyse groups of similar instances and their features, as well as the connections between the groups themselves. The two-way seriated data may be visualized as a two-dimensional heat map or as a three-dimensional landscape where colour codes or height correspond to the values in the matrix. To achieve a meaningful visualization of high-dimensional data, a compactly supported convolution kernel is introduced, which is similar to filter kernels used in image reconstruction and geostatistics. This filter populates the high-dimensional space with values that interpolate nearby elements, and provides insight into the clustering structure. Ordinary two-way seriation is also extended to deal with updates of both the row and column spaces. Combined with the convolution kernel, a three-dimensional visualization of dynamics is demonstrated on two data sets, a news collection and a set of microarray measurements.

Keywords: Two-way Seriation, Gaussian Filtering, Landscape Visualization, High-dimensional Data, Hamiltonian Path,

1. Introduction

Biclustering and heat maps are visual approaches to demonstrate the clustering structure in a set of data with a given feature set. As the data set is updated, the existing distributional pattern of instances and features changes, their relations and relative importance shift. This leads to a dynamic pattern that we wish to visualize to gain insight into evolving structures.

∗Corresponding address: Swedish School of Library and Information Science, University of Borås, Allegatan 1, Borås, S-501 90, Sweden
Several methods exist that achieve a similar purpose. For instance, self-organizing maps (SOMs) have been used to provide a visual overview of the clusters in expanding document collections [21]. A SOM is a two-dimensional grid of a neural network in which the nodes are adjusted as subsequent document vectors stimulate the network. Eventually clusters of documents will be assigned to nodes and the overall structure can be visualized.

In a neural network, the two dimensional layout is somewhat arbitrary and only relative positions matter. The x and y axes are not meaningful. The axes in a matrix layout, however, are typically easily interpreted. For instance, the row space may refer to features and the column space to instances that the features describe. In a term-document matrix the features are index terms (terms that are semantically charged to be significant for marking up a document), whereas instances are documents (text, a web page, a blog entry, etc.). If we rearrange the rows and columns in the matrix such that related terms will be near one another, and clusters of similar documents will also be nearby column vectors, we arrive at a visual reflection of a statistical model, a heat map of the matrix [29].

The underlying idea of a heat map is a seriation of both the row and the column space. We regard seriation as “sequencing objects along a continuum that rely upon a symmetric proximity measure defined between the objects to be seriated” [19]. Seriation is widely used in the visualization of binary matrices [6, 24], and it is also used in genetic research to understand which genes are activated simultaneously [5, 11].

In the context of text mining, [32] introduced a seriation using a distance function that did not only rely on the distributional patterns of words, but also included an external lexical database encoding word relations. This method translated seriation to a graph problem in which an approximate solution to a minimum-weight Hamiltonian path was sought [27]. The weights in the path corresponded to distances, and the nodes to words. While many other methods exist to find a one-way seriation [23], this was proved to be efficient when applied to text classification in conjunction with compactly supported basis functions, B-spline kernels in particular [34].

The purpose of the present work is to introduce an incremental two-way seriation on a non-binary matrix, that is, seriating both the row and column spaces thereof, and subsequently visualize the resulting matrix. The two-way seriation must be able to accommodate updates of the matrix, as either the feature space or the number of instances or both can expand. Eventually we arrive at a surface-type visualization in three dimensions based on the
seriated matrix that shows the structure of the expanding collection. We demonstrate the algorithm using two data sets. The first one is a large collection of news, which, when processed, results in a sparse vector space with a very high number of dimensions. The second collection is a series of experiments on yeast; the instances refer to experiment types, whereas the features are derived from microarray measurements. The key contributions of the paper are as follows:

- Large-scale two-way seriation of sparse data;
- Incremental updates of both the row and column space seriation;
- Application of filtering for better visualization of sparse data;
- 3D visualization of dynamics.

The rest of the paper is organized as follows. We quickly overview the relevant work on seriation and two-way seriation in Section 2. Section 3 introduces the problem of temporal evolution of data sets, and proposes a solution to extend two-way seriation to such cases. The high-dimensional data might be sparse, or the number of features can be so high that visualizing the result of the two-way seriation can be difficult. Section 4 offers a solution by applying a convolution filter to the output of the two-way seriation. Using high-dimensional sparse data sets, Section 5 explores the parameters of the proposed method, and Section 6 applies the method to another collection. Finally, Section 7 concludes the paper.

2. Seriation and two-way seriation

Seriation is a combinatorial data analysis method that reorders instances into a sequence along a one-dimensional continuum. The basis of the re-ordering is a pairwise distance between the instances. Seriation algorithms place instances next to each other if the distance between them is small, and the eventual order reveals regularity and patterns in the whole series [23]. Seriation is different from clustering: clustering groups nearby objects together, but it does not necessarily reveal relations among the groups. Yet both approaches are NP-hard problems [30, p.525].

Seriation as described above is uncommon. Two-way seriation is the typical application. Two-way seriation assumes a matrix representation, or a two-dimensional layout of the data, where rows correspond to instances
and columns correspond to features that describe the individual instances (this assignment of rows and columns is arbitrary and can happen the other way). This approach has a history of over a hundred years (see [23] for an overview), and it is commonly used in a wide variety of information visualization methods, including microarray data [5, 11], binary matrices [24], and others [6]. Seriation is similar to clustering, and two-way seriation is similar to biclustering: instances with similar feature subsets are grouped together in regular patterns. The difference is that the overall structure of the seriated two-dimensional array is meaningful, the groups follow each other in an optimized order. In other words the overall structure is not an arbitrary ordering of row and column cluster trees. Certain types of two-way seriation are also called heat maps [29].

Many seriation algorithms have implementations that are available for free, and some are also available under an open source licence [5, 16, 35]. These implementations deal with dense data.

While sparse data asks for similar seriation methods, visualizing the result is harder. [2] tested different methods to improve the browsing experience of hypertext documents with promising results, although the plots of the sparse matrices resulted from document indexing were not immediately useful for visual analysis. Minimizing the Hamiltonian path length results in a seriation optimal with respect to dissimilarities between neighbouring objects [17, 27]. Taking this approach as its basis, [32] dealt with one-way seriation of sparse data to improve the performance of text classification. Potential visual applications were not considered.

We extend a Hamiltonian path-based seriation to deal with two-way sparse data that are dynamically updated, and we explore some potential visualization methods that reveal patterns in the collection.

Figure 1: An overview of the proposed dynamic two-way seriation and visualization method
3. Dynamic seriation

The seriation method described in [32] works well with a wide range of distance functions on sparse data. The key idea of the algorithm is to study the weighted graph that is described by the pairwise distances between feature vectors. Nodes correspond to features, the weight of an edge is the distance between the two features. More formally, let \( V \) denote a set of instances \( \{x_1, x_2, \ldots, x_n\} \), where \( n \) is the number of instances; the instances are in an arbitrary order. Let \( d(x_i, x_j) \) denote the distance between the instances \( x_i \) and \( x_j \). Let \( G = (V, E) \) denote a weighted undirected graph, where the weights on the set \( E \) are defined by the distances between the features. \( G \) is a \( K_n \) complete graph. The optimal seriation would be a minimum-weight Hamiltonian path of the graph; this is similar to the clustering approach based on the travelling salesman problem [8]. Since this is an NP-hard problem, the paper introduced a simple nearest neighbour heuristic that grows the path one by one and greedily adds the next closest item to the order. We will refer to this approach as ‘left-right heuristic’. The algorithm has two immediate problems:

- It is a one-way seriation, only the feature space of the matrix is seriated.
- Updates are not handled well, as new items can only be appended to either end of the seriation.

To solve the first problem, we use the same seriation algorithm on the instance space as well.

The second problem can be tackled by introducing a new heuristic for the updates. Now let \( x_1, x_2, \ldots, x_n \) be an existing seriation of the instances. Let \( x_w \) denote a new instance. We would like to insert the new instance such that the increase in the total length of the path will be minimal. We need to calculate the following:

- The increases of length if the new instance is appended to either end of the existing seriation: \( d(x_w, x_1) \) and \( d(x_n, x_w) \).

- The increase of length if the new instance is inserted somewhere in between: \( d(x_i, x_w) + d(x_w, x_{i+1}) - d(x_i, x_{i+1}) \) for all \( i = 1, \ldots, n - 1 \).

The location of the new instance will be where the above calculation is minimal. The update of the feature space is similar. We will refer to this approach as ‘insert heuristic’. This approach is very similar to the bond-energy
algorithm [25], but instead of optimizing for ‘clumps’, that is, submatrices containing elements of similar values, we optimize only in one dimension at a time. This allows more flexibility with regards to the distance functions, potentially enabling different distance types for the column and the row space.

4. Filtering and visualization

Two-way seriation of a dense matrix will result in a heat map, that is, a two-dimensional grid of coloured squares where the hot spots will correspond to closely related features and instances.

Sparse data is more difficult. Since the vast majority of the values are zero, a heat map or a three-dimensional view of it would not be meaningful. A similar problem is present in information retrieval and text classification: similar documents or documents corresponding to queries are not necessarily found if they do not share the exact same index terms. To deal with the problem, non-zero elements are introduced in the space to fill in the gap between the original non-zeroes. This is similar to smoothing in image processing where a neighbourhood of points is used with varying weights to produce a new image. If the image was damaged, or parts of it are missing, this smoothing would fill in the gaps. In a vector space of documents, non-zeroes are inserted at location if it is relevant in the corresponding document or query. Common methods include dimensionality reduction techniques such as singular value decomposition [10] and random projection [20]. The benefits of enriching a sparse space with non-zeroes are well-known in language technology applications, often achieving results comparable to humans in certain applications [22].

While the problem is more apparent with sparse data, the same problem exists with very high dimensional dense data. There are too many points to visualize, it is not easy to understand or overview the clustering structure that emerges after the two-way seriation.

A different technique took the one-way seriation of feature space as its starting point and introduced compactly supported wavelets over the feature space [34]. The distance function between the instances measured the overlap between the wavelets. This is similar to image filtering in two dimensions, where convolution is performed on the pixels with a kernel. It is a commonly used operation in blurring, sharpening, edge detection, and other image processing steps. The Gaussian filter (also known as Gaussian blur or the two-dimensional Weierstrass transform) convolves the data with the
two-dimensional Gaussian function $G(x, y) = \frac{1}{2\pi(\sigma_x^2 + \sigma_y^2)} \exp(-\frac{x^2 + y^2}{2(\sigma_x^2 + \sigma_y^2)})$. This transform can be viewed as a ‘smoothed’ version of the data: a value of the transform at the point $(x, y)$ is obtained by averaging the values of the data, weighted with a Gaussian centred at $(x, y)$. To perform the convolution on discrete data, a convolution matrix is generated, typically with dimensions $6\sigma_x \times 6\sigma_y$, eventually leading to a compactly supported kernel. Gaussian filters have been used in language modelling to address data sparseness [7]. We introduce this compactly supported two-dimensional Gaussian filter over the two-way seriated high-dimensional data. This approach technically embeds the data into the $L_2(\mathbb{R}^2)$ space while ‘blurring’ nearby items together. This process is also similar to normalized convolution used in sparse image reconstruction when the kernel is a Gaussian [14, 26]. If the data is assumed to be Gaussian, ordinary kriging also yields similar results [3].

The colour values of a heat map can be interpreted as a height coordinate, and, using two-dimensional wavelets as Parzen window estimators over the points, we may derive a three-dimensional surface visualization of the matrix. Such methods have been used in the past to visualize the topic structure of document collections, also utilizing seriation [9, 31]. Extensions of these methods have also been developed to analyse gene expressions [28]. The ability to capture the temporal changes of a collection is somewhat limited in these methods, whereas a different, two-dimensional visualization method focuses exclusively on dynamics [18].

Note that in our visualisation, points on the x-y axes can have labels: the name of the features and the label of the instances. The landscape that emerges in three dimensions does not represent explicit information of the concepts that underlie the nearby features or instances.

5. Discussion

This section explores the impact of different parameter settings using a high-dimensional, sparse collection that has an evolving temporal pattern.
5.1. Collection and preprocessing

The collection consists of 21,578 news reports that appeared on the Reuters wire between February and October 1987. We filtered words that appear less than three times and also the most common ones. Stop words were removed and stemming was performed. Indexing was performed with Lucene 3.6.0. The resulting index had 10,074 index terms. We used tf-idf weighting to smooth the space.

Figure 2: The consecutive distances of the results of two different heuristic algorithms to find a seriation based on the approximation of a minimum-weight Hamiltonian path. Reuters data set.

The collection is fairly large, enabling us to see how the parameters influence the visual landscape.
5.2. Difference in the reordering heuristic

The computational complexity of the two heuristics is identical, the computational cost of the algorithms is $O(n^2)$, or $O(n \log n)$, if a tree-based spatial index is used [34]. Yet, there is a difference in actual running time (Table 1). The implementation is not parallelized, the running time was measured on a Intel i7 CPU clocked at 2.67GHz. The calculation of the distance function is invoked far more frequently in the insert heuristic and this leads to a slow down, the left-right heuristic is at least twice as fast.

Figure 2 plots the consecutive distances of the results of the two heuristics using Euclidean distance. The sum of these distances gives a single metric to evaluate the relative quality of different seriation methods (Table 2). The left-right heuristic starts off with small distances between the initial neighbours. Then, as it runs out of choices, the two sides of the seriation show big distances between consecutive elements. The insert heuristic is more even. The totals indicate a marginal overall advantage to the insert heuristic. Given that the left-right heuristic is faster to run, it is a good idea to use this to obtain the initial seriation, then use the insert heuristic to update the seriation with the new elements that are added to the collection.

5.3. The impact of Gaussian filtering

Figure 3 demonstrates the impact of filtering the two-dimensional seriated data with a Gaussian kernel. It is obvious from the plain density plot that sparse data do not show a clear heat map structure: showing an almost uniform colour, individual data points are lost in the flood of zero elements. As the convolution with the compact Gaussian kernel is performed, clustering structure is much easier to see and three-dimensional visualization is more viable.

The length of the support of the Gaussian kernel that filters the data has a fundamental impact on the three-dimensional landscape (Figure 4). Given that the number of instances is relatively low, a short support that does not distort the landscape is reasonable. The number of features, however,
Figure 3: Four views of the same data: density plots after two-way seriation. (a) Ordinary density plot shows little clustering structure due to data sparsity. Showing an almost uniform colour, individual data points are lost in the flood of zero elements. (b) A three-dimensional view from above of the density plot is a better summary of the clusters. By adding height to the data points, they visually rise above the zero elements. (c) The density plot after Gaussian filtering clearly shows the clusters. (d) Plotting the density plot in three dimensions after Gaussian filtering shows a landscape-like view of the clustering structure.

is large. With a shorter support of about one hundred features length, the landscape is still jagged (Subfigure 4(a)). On the other hand, a support that
(a) Support length: (300; 100)
(b) Support length: (300; 300)
(c) Support length: (300; 500)
(d) Support length: (300; 700)

Figure 4: The same data with different support lengths in the Gaussian kernel

is too long blurs too much of the details (Subfigure 4(d)). We find that on this data set, a support of five hundred features is a reasonable compromise.

There is no explicit way of finding the correct length of the support. The landscape we are looking at depends on many factors, including the quality of the seriation, the range and spread of matrix elements, and the total number of rows and columns. It is largely a qualitative judgement, and most probably the only way to select the best support length is to generate a sequence with varying supports.
5.4. Distance functions

The distance functions have a deep impact on the overall structure of the outcome. We tested two distance types, Euclidean and cosine. The former takes the length of the vector into consideration, and longer vectors are typically further away from the rest of the data. The cosine dissimilarity is agnostic to vector lengths. Due to these characteristics, a seriation with Euclidean distance penalizes large vectors and pushes them to the side (Figure 5). The Gaussian filter cannot efficiently blend in these outliers with the rest of the data, resulting in undesirable peaks on the sides. The cosine distance (Subfigure 5(b)) results in a smoother landscape.

Apart from distance functions that consider distributional patterns alone, there are ones that are able to incorporate resources external to the data at hand. For instance, lexical resources organize index term relations in a graph hierarchy, yielding a family of graph-based distance functions [4, 13]. The impact of such distance functions to the quality of seriation was studied elsewhere [32].

If the data contain class information, that is, the elements belong to categories, it is possible to quantitatively evaluate the quality of seriations obtained with distance functions. One may build support vector classifiers that also consider the order of elements. The distance function that yields the highest classification accuracy will correspond to the best seriation [33, 34].
5.5. Dynamics

Figure 6 demonstrates the dynamic updates of two-way seriation. Starting with one hundred documents as the basis of the left-right seriation (Sub-figure 6(a)), eventually the insert heuristic expands the seriation to the full
This animated visualization of evolving text corpora displays the underlying dynamics of semantic content. New topics, new themes enter, others become less prevalent. These changes are captured by the appearance and disappearance of landscape features. Hot spots are identified by fast growing areas, where many new documents or index terms are inserted. Later certain peaks may split, indicating the formation of subtopics, as more and more details are added to the document collection.

Figure 7: Three snapshots of the dynamic yeast genome data with five new instances in each step.

The primary reason we believe that a three-dimensional landscape view is superior to a two-dimensional contour plot is because changing height is easier to notice than changing shade. Thus dynamics are easier to inspect in a three dimensional view. In a related effort, we used self-organizing maps \cite{21} to study the dynamics of a corpus of comparable size. The overall shifts were difficult to track when animated \cite{2}.

6. An application to microarray data

Microarrays are commonly used in massive gene expression data analysis. They measure the expression levels of large number of genes simultaneously. The measurement results have a very high number of dimensions, and genomic data is often sparse \cite{1}.

Such data are often taken in the temporal domain. For instance, yeast cells are exposed to an environmental shock, such as a sudden change of

\footnote{1 http://www.youtube.com/watch?v=ANPyp4t2Csw}

\footnote{2 http://www.youtube.com/watch?v=sTR6yX0Ki9U}
temperature, then samples are taken at time intervals to generate the corresponding microarray data. By looking at the dynamics, genes that are activated together in response to the shock help to understand how organisms deal with environmental conditions.

To reveal which genes are activated together, two-way seriation and heat maps are frequently used to analyse the microarray data sequences. The general structure of two-way seriation input data is the numerical matrix of gene expression measurements coming from the microarray. Measurements of the same gene are supposed to be in the same row while measurements under the same experimental condition (at the same time point, for instance) are assumed to be in the same column\(^3\). The microarray data is typically short in the temporal domain [12], which is not surprising, since obtaining a sample is a tedious and expensive procedure.

Heatmaps work well for visualizing one temporal sequence. However, if multiple experiments are carried out, for instance, different types of environmental shocks are tested, heat maps must be generated for each, and they will not be directly comparable. This is where our proposed method helps.

We suggest to map the different kinds of measurements and the corresponding microarray data in one space. For instance, given two types of environmental shocks, temperature change and a chemical one, the instance space will consists of these two at time step zero, and the feature space is the corresponding microarray data. Then subsequent temporal samples would add new instances two by two. Plotting the landscape would reveal further information about the mutual gene activation across experiment types.

Using a common data set of yeast microarray data as a case study [15], we included the following environmental changes that were measured by the authors in different sets of experiments:

- Mild heat shock at variable osmolarity;
- Mild heat shock at variable osmolarity, cells were grown supplemented with 1 M sorbitol through the experiment;
- Mild heat shock at variable osmolarity, cells were grown supplemented with 1 M sorbitol until the end of the heat shock;
- DTT exposure;

\(^3\)http://bioinfo.cipf.es/babelomicstutorial/biclustering
• Hypo-osmotic shock.

Only three time steps were considered, sample collection after five minutes, fifteen minutes, and thirty minutes.

Since the number of instances was very low, we set the corresponding Gaussian support to two. The microarray data had 6152 features, we set the support to two hundred. Initial two-way seriation was more efficient with the left-right heuristic. We used the cosine distance function.

Figure 7 plots the temporal evolution of the data set. The changes between the time steps are dramatic, indicating that not just the temporal clustering structure, but also the information across the experiments is valuable. By labelling the genes, the temporal dynamics reveal which genes act in the same way in different environmental conditions, thus allowing the identification of the most crucial ones. By the third step, two clusters are clearly indicated by the highest peaks in the landscape, which are plotted in yellow. Deeper regions are also visible, showing genes which are not involved in any kind of response to the environmental shocks.

7. Conclusions

Understanding the dynamics of a high-dimensional data set is not feasible by plotting trends and analysing individual features. We presented a method that adds new updates to collection while keeping its previous conceptual order. Similar instances and related features are grouped together in an initial iteration. The updates are inserted to align with this existing clustering structure, but they also change the landscape as they introduce new features and instances. A three-dimensional view of the data set reveals the groups of related concepts and instances, and also offers insights into the dynamics to identify hot spots of changing structure.

Future work shall include a more interactive view, in which, by means of a graphical user interface, the features and instances can be explored, and further useful information can be gained about the dynamics.


