Efficient Visualization of Large-scale Data Tables through Reordering and Entropy Minimization

Nemanja Djuric, Slobodan Vucetic
Temple University

Abstract

Visualization of data tables with n examples and m columns using heatmaps provides a holistic view of the original data. As there are $n!$ ways to order rows and $m!$ ways to order columns, and data tables are typically ordered without regard to visual inspection, heatmaps of the original data tables often appear as noisy images. However, if rows and columns of a data table are ordered such that similar rows and similar columns are grouped together, a heatmap may provide a deep insight into the underlying data distribution. We propose an information-theoretic approach to produce a well-ordered data table. In particular, we search for ordering that minimizes entropy of residuals of predictive coding applied on the ordered data table. This formalization leads to a novel ordering procedure, EM-ordering, that can be applied separately on rows and columns. For ordering of rows, EM-ordering repeats until convergence the steps of (1) rescaling columns and (2) solving a Traveling Salesman Problem (TSP) where rows are treated as cities. To allow fast ordering of large data tables, we propose an efficient TSP heuristic with modest $O(n \log(n))$ time complexity.

Data visualization background

Visualization is used for exploratory analysis prior to application of statistical methods, as a confirmatory tool to disprove or confirm hypotheses, while sometimes visual presentation is an ultimate goal.

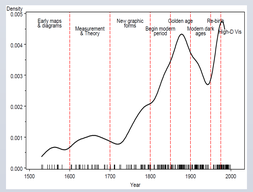

Figure 1. Historical development

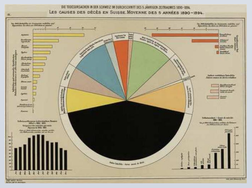

Figure 2. Illustration from 1897 report

However, despite its long history, there still remains a need for further development of visualization methods. This is particularly evident when working with large-scale data, where commonly used visualization tools are either too simplistic to gain a deeper insight (e.g., histograms, scatter plots, pie and bar charts), or are too cumbersome and computationally costly in large-scale setting, such as parallel coordinates, correlation matrix plots, and biplots and star plots. Inadequacy of standard tools has been recognized in a number of recent papers, as summarized by Vempala: *Data in high dimension are difficult to visualize and understand. This has always been the case and is even more apparent now with the availability of large high-dimensional datasets and the need to make sense of them.*

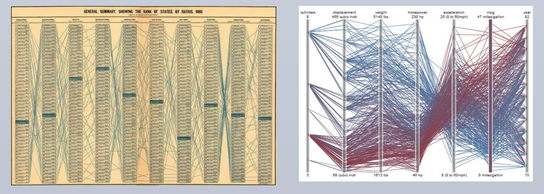

Figure 3. Visualization using parallel coordinates: a) Illustration from 1883; b) Modern use

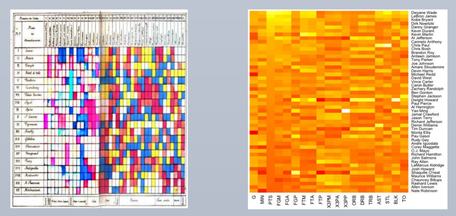

Figure 4. Visualization using heatmaps: a) Illustration from 1873; b) Modern use

Data reordering and the EM-ordering algorithm

Improving heatmap using data reordering

Data reordering methods are based on the following observation: any permutation of table columns or table rows does not lead to loss of information. Therefore, by permuting rows (columns) so that similar rows (columns) are close, we can reveal unknown regularities and patterns in the data without modifying the data, see Figure 5.

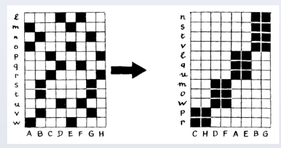

Figure 5. Idea behind data reordering

Algorithm	Time	Space
PCA	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n)$
LLE	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
SC	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
HC	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$
HC-olo	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$
LK	$\mathcal{O}(n^{2.2})$	$\mathcal{O}(n)$
TSP-means	$\mathcal{O}(n \log(n))$	$\mathcal{O}(n)$

Figure 6. Algorithm complexities

Existing approaches

1) One-dimensional projection by either **PCA** or **LLE** effectively induces a linear ordering in the new 1-D space.
2) Spectral clustering (**SC**) finds ordering of examples by computing the second largest eigenvector of the normalized similarity matrix.
3) Hierarchical clustering (**HC**) method finds binary tree representation of the data in a bottom-up fashion, resulting in $O(n^2)$ time complexity. To find ordering of examples, we simply read leaves of the tree from left to right. However, as there are 2^{n-1} linear orderings of tree nodes that obey the obtained tree structure, and HC algorithm finds the optimal leaf ordering (**HC-olo**) for a given tree structure in $O(n^3)$ time.
4) If examples are viewed as cities, the task can be seen as a Traveling Salesman Problem (TSP), where we need to find a path through all the cities such that the traversal cost is minimized. The Lin-Kernighan (**LK**) method has been widely accepted as the best TSP solver providing the best trade-off between tour quality and computational speed, however it is applicable only to moderately-sized data sets.

Data reordering through data compression

Assume that data set D is given, with $D = \{\mathbf{x}_i, i = 1,...,n\}$, where $\mathbf{x}_i = [x_{i1}, x_{i2}, ..., x_{im}]$. Then, the task is to reorder the data so that it is maximally compressible. We use Differential Predictive Coding (DPC) scheme to code the data, which transforms the data set D into D_{DPC} as follows,

$$D = \{\mathbf{x}_i, i = 1,...,n\} \rightarrow D_{DPC} = \{\mathbf{x}_1, \varepsilon_2, ..., \varepsilon_n\}$$
$$\text{where } \varepsilon_i = (\mathbf{x}_i - \mathbf{x}_{i-1}),\ i = 2,..., n.$$

We measure data compressibility using the entropy of prediction errors $\boldsymbol{\varepsilon}_i$ that are assumed Gaussian, which can be written as

$$H(\varepsilon) = \frac{n}{2}(m \cdot \log(2\pi) + \sum_{j=1}^{m} \log(\sigma_j(\varepsilon))) + 0.5 \sum_{i=2}^{n} \sum_{j=1}^{m} \frac{(\mathbf{x}_{\pi(i),j} - \mathbf{x}_{\pi(i-1),j})^2}{\sigma_j^2}.$$

where π is the current ordering of rows. The optimization problem becomes $(\pi^*, \{\sigma_1^*, ..., \sigma_m^*\}) = \arg \min_{\pi, \{\sigma_1, ..., \sigma_m\}} H(\varepsilon)$, solved using the EM-ordering algorithm:

Algorithm 1 EM-ordering

Inputs: data set D; initial guess for $\{\sigma_j\}_{j=1,...,m}$
Output: ordered set D; learned $\{\sigma_j\}_{j=1,...,m}$

1. **repeat** until convergence
2. **run** TSP solver for current σ_j to find π
3. **calculate** σ_j for current ordering of D

However, the best TSP solvers have super-quadratic time complexity, and we propose an $O(n \log(n))$ method, called TSP-means:

1. Create a 2^l-ary tree through recursive runs of k-means ($k = 2$)
2. Traverse the tree breath-first, solve TSP defined on each node's children

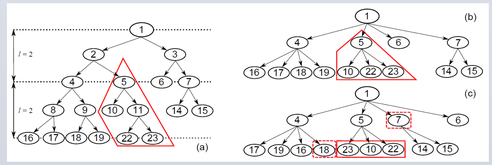

Figure 7. Steps of TSP-means, an $O(n \log(n))$ TSP solver

Experiments

Validation of TSP-means

We generated 2-dimensional data set called *circles*. We uniformly at random sampled half of the examples from a circle of radius 3, and the second half from the circle of radius 4. We also added small noise to all examples, and show the results in Figure 8.

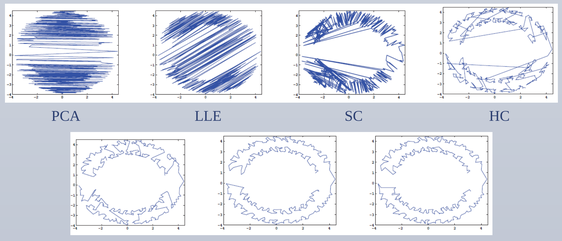

Figure 8. Results of reordering algorithms on *circles* data

To compare the execution times for HC-olo, LK and TSP-means, we uniformly sampled 2-D and 3-D examples from a square and a cube of width 1, and increased data size from 500 to 1,000,000 (see Figure 9).

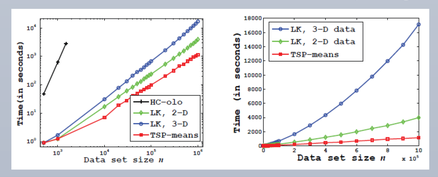

Figure 9. Execution times on uniform data set: (a) logarithmic scale; (b) linear scale

Validation of EM-ordering

When labeled examples are available, as a quality measure we used Figure of Merit (FOM). Denoting label of the i^{th} example as $y(i)$, and using binary indicator function $I()$, FOM score of ordering π is computed as

$$\text{FOM}(\pi) = \frac{1}{n-1} \sum_{i=1}^{n-1} I(y(\pi(i)) \neq y(\pi(i+1))).$$

In Table 1 we report FOM of ordering methods on 11 classification sets.

data set	n	m	c	baseline	PCA	LLE	SC	HC	HC-olo	LK	EM-1	EM-5
iris	150	3	3	0.637	0.188	0.161	0.187	0.181	0.134	**0.114**	0.121	**0.114**
wine	178	13	3	0.649	0.225	0.531	0.056	0.062	0.056	0.045	0.047	**0.032**
breast	277	9	2	0.413	0.337	**0.324**	0.330	0.344	0.340	0.344	0.339	0.338
adult	1,500	123	2	0.374	0.267	n/a	0.235	0.276	0.267	0.252	0.238	**0.232**
banana	1,500	2	2	0.514	0.348	0.356	0.279	0.151	0.152	**0.147**	**0.147**	**0.148**
covertype	1,500	54	7	0.630	0.620	0.612	0.583	0.430	0.395	**0.382**	0.424	0.438
gauss	1,500	2	2	0.491	0.316	0.286	0.286	0.263	0.269	**0.260**	0.266	0.267
madelon	1,500	500	2	0.506	0.447	0.494	0.489	0.464	0.449	0.444	0.439	**0.399**
magic	1,500	10	2	0.464	0.416	0.451	0.300	0.258	**0.235**	0.243	0.244	0.259
waveform	1,500	21	3	0.680	0.462	0.461	0.461	0.266	0.250	0.249	0.239	**0.238**
wave noisy	1,500	42	3	0.680	0.472	0.493	0.466	0.344	0.309	0.310	0.282	**0.237**

Below, in Figure 10, we show heatmaps of reordered *waveform* data set.

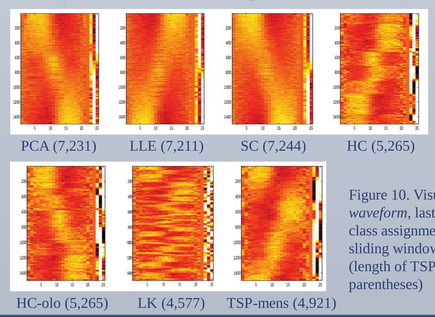

Figure 10. Visualization of *waveform*, last 3 columns are class assignments averaged over sliding window of length 20 (length of TSP tour given in the parentheses)

Real-world applications

Traffic application

We used data set of traffic volumes (no. of cars per min), reported every 10 min by 3,265 sensors on roads in Minneapolis, MN, on Dec. 23th, 2005.

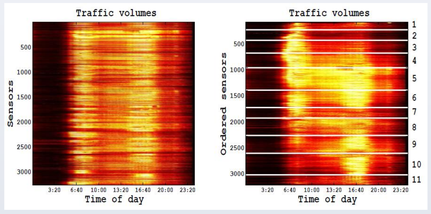

Figure 11. Traffic data: a) original data set; b) data set after reordering using EM-ordering

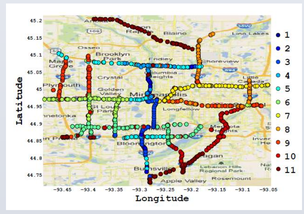

Figure 12. Color-coded sensor locations in Minneapolis road network

From the original data set little can be seen (Fig. 11a). After reordering we see several types of traffic patterns (Fig. 11b; heavy traffic during morning or afternoon only, light or heavy traffic during most of the day, etc.). Upon visual inspection we split the ordered sensors manually into 11 clusters, and show the locations of sensors, colored according to their labels (Figure 12). We see that the sensors along the same road segments were clustered together, and nearby road segments were assigned to similar clusters.

Stock market application

We ran EM-ordering on *stocks* data (see Figure 13a; 252 daily stock returns of 89 companies from 9 sectors of industry).

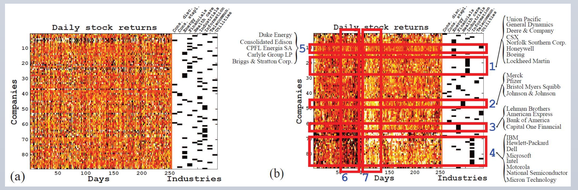

Figure 13. Stocks data: a) original data set (dark pixels encode negative returns, bright pixels positive returns); b) data set after reordering using EM-ordering

After reordering (see Figure 13b), we can observe several clusters of companies operating in industrials, health care, financials, and information technologies sectors (clusters 1 - 4, respectively), having specific patterns of daily returns. We also detected companies from energy and utilities sectors in cluster 5, whose daily returns, unlike returns of the companies from other sectors, did not fluctuate much. Lastly, after reordering the transposed data matrix (i.e., ordering days instead of companies), bear and bull trading days can be easily detected (clusters 6 and 7, respectively).

Conclusions

We proposed EM-ordering, an efficient reordering algorithm for data visualization, naturally emerging from entropy-minimization framework. In addition to finding a near-optimal ordering of the examples, the algorithm can automatically detect noisy features and decrease their influence on the final ordering. Moreover, our algorithm has very favorable time and space complexity, allowing efficient visualization of data tables with millions of examples. Experiments showed that the algorithm outperformed existing methods while being faster.