This paper is © IEEE and appears reformatted in *IEEE Transactions on Visualization and Computer Graphics*, 30(1), 2016, DOI: 10.1109/TVCG.2015.2467552

Use attraction with this dataset.

The Data Context Map: Fusing Data and Attributes into a Unified Display

Shenghui Cheng, *Student Member, IEEE* and Klaus Mueller, *Senior Member, IEEE*

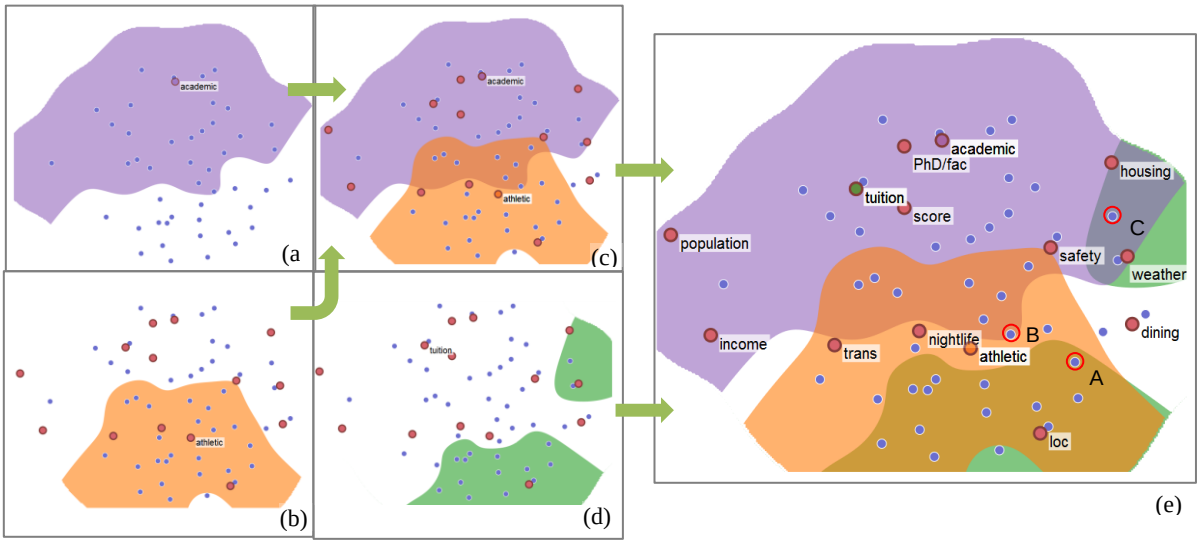


Fig.1. The process to find a dream university. (a) good academics region(>9). (b) good athletic region (>9). (c) combined region by (a) and (b). (d) low tuition region(<$18,000). (e) combined region by region (c) and (d).

Abstract— Numerous methods have been described that allow the visualization of the data matrix. But all suffer from a common problem – observing the data points in the context of the attributes is either impossible or inaccurate. We describe a method that allows these types of comprehensive layouts. We achieve it by combining two similarity matrices typically used in isolation – the matrix encoding the similarity of the attributes and the matrix encoding the similarity of the data points. This combined matrix yields two of the four submatrices needed for a full multi-dimensional scaling type layout. The remaining two submatrices are obtained by creating a fused similarity matrix – one that measures the similarity of the data points with respect to the attributes, and vice versa. The resulting layout places the data objects in direct context of the attributes and hence we call it the data context map. It allows users to simultaneously appreciate (1) the similarity of data objects, (2) the similarity of attributes in the specific scope of the collection of data objects, and (3) the relationships of data objects with attributes and vice versa. The contextual layout also allows data regions to be segmented and labeled based on the locations of the attributes. This enables, for example, the map's application in selection tasks where users seek to identify one or more data objects that best fit a certain configuration of factors, using the map to visually balance the tradeoffs.

Index Terms— High Dimensional Data, Low-Dimensional Embedding, Visual Analytics, Decision Make, Tradeoffs

1 Introduction

The data matrix, *DM,* is one of the most fundamental structures in data analytics. It is the $M \times N$ rectangular array of N variables (often referred to as *attributes* or *labels*) and M samples (also frequently called *cases*, *observations,* or *data items*). The $N \times N$ or $M \times M$ similarity (or *co-occurrence*, *correlation*) matrix S is another often used structure and frequently derived from *DM*. Here it should be noted that the roles of variables and samples can change – there are many practical settings in which we consider the 'variables' as outcomes we wish to predict (or use for prediction) using the set of samples acquired before. To visualize *DM*, current methods either focus on spatially preserving the relations among the samples or on spatially preserving the relations among the variables, but they are typically not capable to do both. This is a severe limitation when one wishes to transform *DM* into a comprehensive map in which the acquired samples are accurately presented in the context of the variables. Our paper describes new data and similarity matrices that overcome these deficiencies.

To illustrate these points, let's consider a parent looking for a university for their child. This is an important decision with many factors to consider – academic score, tuition, athletics, teacher/student ratio and many others. College Prowler [15] is a popular website that allows users to navigate this parameter space by filtering – using slider bars and menu selections for each parameter to narrow down the search. But this is rather tedious and it also makes it difficult to recognize tradeoffs. Conversely, a visualization expert would use interactive parallel coordinates [17]

• *Shenghui Cheng and Klaus Mueller are with the Visual Analytics and Imaging Laboratory, Computer Science Department, Stony Brook University and SUNY Korea. Email: {shecheng, mueller}@cs.sunysb.edu*

Manuscript received 31 Mar. 2015; accepted 1 Aug. 2015; date of publication xx Aug. 2015; date of current version 25 Oct. 2015.
For information on obtaining reprints of this article, please send *e-mail to: tvcg@computer.org.*

plots but it is difficult to imagine that an average parent would engage in such an advanced interface. There are other visualization methods, such as biplots or interior layouts but these are seldom found in the mainstream arena. The wide-spread familiarity of maps, on the other hand, makes these a natural canvas to overview the landscape of universities in the context of the various factors (attributes) to consider. Parents could simply sit back and examine this illustration like an infographic and then decide on a school. They could still use a filter to eliminate some schools from the map but they would never lose sight of the big picture.

Methods like Multidimensional scaling (MDS) [23][2], self-organizing map (SOM) [6][22], t-distributed stochastic neighbor embedding (t-SNE) [28], locally linear embedding (LLE) [31], etc. create 2D map-like data layouts computed from the similarity matrix S of schools (in this case). The entries of S are derived by assessing the distances of pairs of the M schools in the *N*-D space spanned by the N attribute axes. The maps will show similar schools as clusters, and special schools as outliers. This is certainly useful, but parents will not know from the plot alone why some schools are special and others are clustered. What is their ranking, tuition, athletics, etc.?

It is important to note that *S* could just as well hold the similarities of attributes. The maps mentioned above would then allow a visual assessment of the grouping of attributes. So instead of finding that schools A and B are very similar (or dissimilar) in terms of their attributes, one would find that attributes C and D are heavily correlated (or not) in this set of schools. A parent might learn that the higher the academic score, the higher the tuition, and the higher the number of students per faculty (see findings presented in [37] where such a map was presented). And so, if the parent is interested in smaller classes, schools with lower academic scores might be a better choice. Hence, while such a plot is useful in explaining the relationships of the different features of the educational landscape, what it now cannot do is allow anxious parents pick a specific school for their child, which is what they really wish to do.

We propose a framework that overcomes these limitations and combines both of the similarity aspects derived from *DM* into a single comprehensive map which we call the data context map. It requires a non-trivial fusion of the two alternative similarity matrices *S* discussed above. By ways of this fused matrix a mapping can be performed that allows users to faithfully appreciate all three types of relationships in a single display: (1) the patterns of the collection of samples, (2) the patterns of the collection of attributes, and (3) the relationships of samples with the attributes and vice versa. Further, the contextual mapping also provides the information needed to add semantic labelling of the samples as well as the regions they reside in. Iso-contouring these regions then creates decision boundaries by which one can easily recognize trade-offs among different samples which can be helpful in complex decision making. Our paper demonstrates this by ways of a few practical examples.

Our paper is structured as follows. Section 2 summarizes related work. Section 3 provides its theoretical aspects. Section 4 describes the construction of our data context map. Section 5 presents case studies. Section 6 concludes the paper and expands on future work.

2 Related Work

The visualization of high-dimensional data on a 2D canvas essentially follows three major paradigms – projective data displays, interior displays, and space embeddings. However, since the visualization of high-dimensional data in 2D is inherently an ill-posed problem, there is no method without drawbacks. It is simply impossible to preserve all variances of a high-dimensional point cloud in a 2D mapping. Hence the different methods that have been described offer different strengths and weaknesses, but some do better than others.

2.1 Projective and interior displays

These displays typically warp the data in some way to emphasize certain properties, such as locality or similarity. A projective display is the scatterplot matrix [12] which is an extension of the scatterplot. It reserves a scatterplot tile for each pair of variables and projects the data items into it. This distributes the data context into two variables per tile which makes it difficult to appreciate the overall context pertaining to all variables simultaneously. In addition, the mapping operation can lead to ambiguities as points located far way in high-dimensional space may project into similar 2D locations. This adds to the difficulties for recognizing multivariate relationships.

Parallel coordinates and their radial version, the star plot [1], represent the variables as parallel or radial axes, respectively, and map the data as polylines across the axes. However, the clutter of polylines can become a significant problem once the number of dimensions and data points increases. In order to decrease the clutter of lines, star coordinates [20] arrange the attribute axes in a radial fashion but instead of constructing polylines, they plot the data points as a vector sum of the individual axis coordinates. However, since a vector sum is an aggregation, it maps the data to locations that are not unique. In other words, points that map to nearby locations may not be close in high-dimensional space, and vice versa. To help users resolve these ambiguities, at least partially, an interactive interface is often provided that allows them to rotate and scale the data axes and so uncover false neighbors.

In fact, there are number of displays that are similar to star coordinates and share its shortcomings [32]. These are Radviz [13], Generalized Barycentric Coordinates (GBC) plot [29], and PolyViz [14]. We call them *interior displays* since they all lay out the variables as *dimension anchors* [14] around a circle and map the data items as points inside it, given some weighting function that relates to a data point's different attribute strengths. All of these displays are useful in what they have been designed to convey, that is, the relation of data points with respect to the attributes. But since the mapping function does not involve the similarity of the data points, ambiguities result.

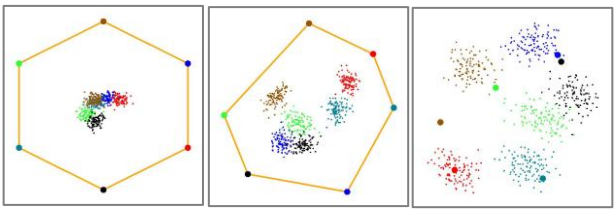
Our research described here has been motivated by recent work presented by the authors [7] which proposes an optimization approach to reduce the data mapping ambiguities in Radviz-type displays. The current framework is radically different in that it maps the attributes not in the periphery along a circle, but intersperses them into the data distribution which reduces all mapping errors significantly. It also enables the region labelling and decision boundaries discussed above.

good for me!

2.2 Comparing the interior displays

We have shown in [7] that the method of GBC [29] can serve as a standard reference framework to describe most interior displays. The GBC plot uses the dimension values of an N-D point as weights in a weighted sum of the anchor 2D locations to determine the point's placement in the 2D polygon.

Using the GBC plots, we conducted a controlled experiment to compare them with the method proposed here. For this, we generated a test dataset comprised of a set of 6 6-D Gaussian distributions. We first randomized the 6 6-D center vectors and then randomized 600 data points following these distributions. Fig. 5d visualizes this dataset using parallel coordinates, assigning each Gaussian a unique color. In addition, we also colored the axes (representing the 6 dimensions) such that each axis color matches that of the cluster with the highest value for that dimension. Fig. 2 shows how (a) standard GBC compares with (b) the optimized GBC plot [7], and (c) the method proposed in this paper which allows the attribute nodes to intersperse with the samples. We can show our method is more flexible and can preserve the pairwise distances well. We will describe more comparison in section 4.1.7.



(a) (b) (c)

Fig.2. Comparing different attribute/sample layout schemes: (a) standard GBC, (b) optimized GBC, and (c) our new method.

2.3 Embedded displays

The ambiguities in the relations of the data points are often overcome by embedding the high-dimensional space into the 2D canvas. Principal component analysis (PCA) [19] finds the two eigenvectors associated with the largest variation in the data (expressed by the largest positive eigenvalues) and then projects the data points into the plane spanned by these vectors. Other methods seek to create a mapping from high-dimensional to 2D space that optimizes for some measure of data point similarity. MDS [23] aims to preserve some distance metric, such as Euclidian distance or pattern distance [24]. Other mappings, such as ISOMAP [34], LLE [31], SOM [6][22], t-SNE [28], LAMP [18], and PLP [30] optimize for geodesic distance, distribution distance, locality, etc.

In these 2D embeddings, the viewer can easily appreciate neighborhood relations and obtain a good overview of the space quickly. However, these methods also have a shortcoming – the mapped data points no longer maintain any context with the attribute space as this information is typically not preserved in the mapping. If users wish to see the relationships of both attributes and data samples then two separate maps need to be created using the two alternative forms of the similarity matrix *S* as presented in the introduction – one for the samples and one for the attributes. But with two separately and independently created maps, it is difficult, if not impossible, to appreciate the mutual relationships of the samples and their attributes – the context. The method we describe in this paper fuses the two alternative similarity matrices and so is able to create an embedding in which the relationships among samples, among attributes, and among the two of them is equally well preserved. We note that in practice we use a dissimilarity (or distance) matrix. Similarity is just the reverse of dissimilarity.

2.4 Fused displays

The work on fused displays is relatively rare. One recent implementation is by Broeksema et al. [3] who similar to us, have also created a fused matrix of samples and attributes and used it for 2D layouts. Our approach is different from theirs in multiple ways:

- Their framework is primarily designed for categorical data. Numerical data are binned into regular intervals which can be inaccurate. Conversely, our approach starts with numerical data by default and could use the approach of Zhang et al. [37] to transform any categorical variables into numerical ones, taking into account the pairwise distribution relationships.
- They use a linear projection approach based on Multiple Correspondence Analysis (MCA) to create the 2D mapping. Our layouts are generated via numerical optimization which can support a variety of constraints and can also better preserve high-dimensional relationships.
- They compute a tiled Voronoi diagram to divide the domain into value regions which only accounts for the relationships among the attributes and their levels. Our approach generates a set of general iso-contours computed from a continuous heat map of the data, using adaptive kernel density estimation. The extended accuracy this affords also affects the accuracy of the decision boundaries computed from these regions [4].

3 Theory and Method

We wish to create a mapping in which all three types of relationships in *DM* are preserved – the relationships among the samples, among the attributes, and mutually among the samples and attributes. Here, the notion of relationship can be a distance, such as Euclidian (across space) or geodesic (across a manifold), or a similarity, such as Pearson's correlation [33], cosine, or pattern, or it can be some measure of significance, such as value or feature. We combine these functions collectively into a *distance metric, F*, and note that, depending on the application, each relationship might be expressed in a different *F*. For example, the similarity of attributes might be measured by correlation, while the proximity of samples might be gauged via the Euclidian distance. We wish for a mapping that preserves this set of simultaneous constraints as well as possible. It calls for an optimization strategy on a fused representation of three types of relationships. The pipeline of this process is shown in Fig.3.

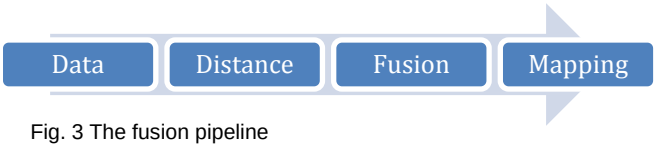


Fig. 3 The fusion pipeline

In the following we outline the various steps of this pipeline in detail. The underlying primitive is a distance matrix, one for each of the three pairs, encoding the respective *F*. The fusion process then merges these three matrices into a single distance matrix emphasizing certain constituents or equalizing them. This is followed by a mapping to 2D using an optimization process. We use an MDS-type strategy because it is well tested for such mapping problems.

3.1 Data Matrix

We begin with DM, the *data matrix,* with m rows and n columns,

$$DM = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{m1} & \cdots & x_{mn} \end{bmatrix}$$

Here, the rows denote the data samples, the columns denote the variables and x_{ij} is the data value in the ith row and jth column. Without loss of generality, we assume DM is normalized to [0, 1].

Depending on how we look at *DM*, row-wise or column-wise, we have two types of spaces – the data space *D* and the variable space *V*, respectively. The data space *D* contains all m data items (samples):

$$D_i = [x_{i1}, x_{i2}, \ldots, x_{in}] \qquad (i = 1,2, \ldots, m)$$

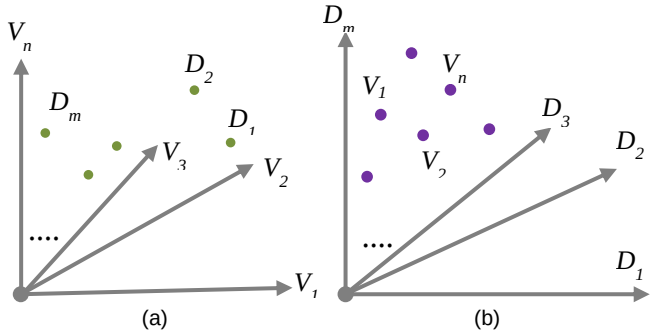


Fig. 4. The two spaces: (a) data space *D* and (b) variable space *V*.

and is spanned by the n orthogonal attribute (or variable) axes (see Fig, 4a) Conversely, the variable space V contains all n data attributes:

$$V_j = [x_{1j}, x_{2j}, \dots, x_{mj}]' \qquad (j = 1,2, \dots, n)$$

and is spanned by a set of m orthogonal data item axes (Fig. 4b).

The data space D is the more familiar of the two but there are many applications, in which samples can turn into attributes and vice versa depending on the focus of the analytics. For example, for a data matrix storing the results of a DNA microarray experiment for multiple specimens, one research objective might consider the genes expressed in the microarray to be the samples and the specimens to be the attributes, or vice versa.

3.2 The Composite Distance Matrix (CM)

The next step is to define the desired distance or similarity metric for each relationship. Mapping more similar items into closer proximity, we need to use (1-correlation), and (1-attribute value) etc, while the spatial distance metrics, such as Euclidian can be used as is. We have four different distance matrices:

- *DD* to store the pairwise distance of data items
- *VV* to store the pairwise distance of attributes (variables)
- *VD* to store the pairwise distance of attributes to data items
- *DV* to store the pairwise distance of data items to attributes

DD is an $n \times n$ matrix with elements *DDij=F(Di, Dj)* and *VV* is a $m \times m$ matrix with elements *VVij=F(Vi,Vj)*. Fig. 6a and Fig. 6 b shows an MDS layout of *DD* and *VV* respectively for the 6 test Gaussians described in Section 2.2.

3.2.1 The Data to Variables Distance Matrices (*DV*, *VD*)

The *DV* and *VD* matrices are new types of matrices. They are required to enforce the distance/similarity constraints in the relation of the data samples with the attribute (dimension) anchors and vice versa. In the following, let us first consider *DV* – similar arguments also hold for *VD*.

Referring to Fig. 4a which shows the data space *D*, one can make the argument that an attribute axis is essentially just another data sample – a (fictional) data point with unit length, n dimensions, and a single non-zero component, namely a value of 1 for the attribute's dimension j. So essentially, the attribute vector serves a dual role: (1) as a dimension axis and (2) as a data point. With this in mind we can then impose any distance metric that links the m data samples with the n attribute axes to fill the $m \times n$ matrix *DV*.

The derivation of the matrix *VD* follows a similar line of thought. Just now we consider the variable space *V* depicted in Fig. 4b where the axes are m-dimensional unit vectors each with exactly one dimension component set to 1. A point in that space is defined by the values a certain variable has for all of the data samples – one column of *DM*. For example, for a car dataset, if V_1 is horsepower (hp) and V_2 is miles per gallons (mpg) and we have two cars – a VW and a Ford – then the coordinates for V_1 would be [hp(VW), hp(Ford)] and the coordinates for V_2 would be [mpg(VW), mpg(Ford)]. We can again impose any distance metric between the n V-points and the m points constituted by the *D*-vectors to fill the $n \times m$ matrix *VD*.

We note that in order for CM to be a proper distance matrix, *VD* should be a transpose of *DV*. This, however, is not necessarily the case, even when normalizing the vectors in *V* and *D* which would place all distance relationships on the surface of a hypersphere. It occurs because *V* and *D* have different dimensionalities (and different hyperspheres) and are also not related by a simple scale factor. The only similarity metric we know that fulfils this matrix identity is (1-*value*), where '*value*' is the value a space point *SP* has for a space dimension vector *SD*'s coordinate. The (1-*value*) distance can be thought of as a *significance distance*. It is small for a given data point when the value of a point's attribute is large, encoding a notion of affinity that *SP* has for *SD*. We have used this distance for all examples shown in this paper. For the case when *VD* is not a transpose of *DV*, like Euclidian or correlation distance, we typically select one of *DV* or *VD* – using the one with the larger matrix norm – and computing the other by transposing it. In this case, *DV* and *VD* become symmetric.

3.2.2 Assembling the composite distance matrix (*CM*)

With all four constraint matrices in place, we can now assemble the composite distance matrix *CM* from them. The fused space composed of *D* and *V* and the composite distance matrix *CM* are shown in Fig. 5. We can now use it within an MDS-like optimization framework to achieve the 2D mapping into the joint sample/attribute display. But first we need to make some adjustments as is described in the following section.

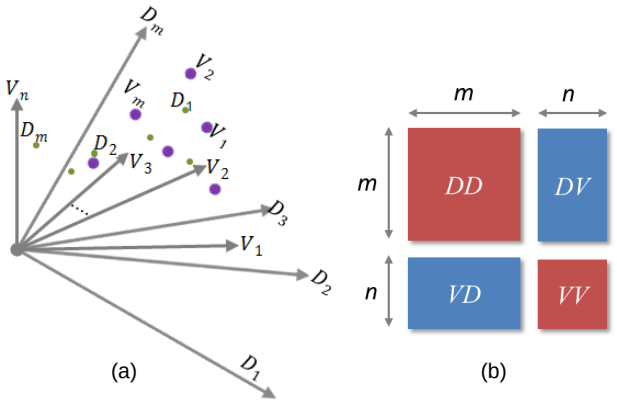


Fig. 5: (a) The fused space composed of *D* and *V* and (b) the composite distance matrix *CM* and the extents of its submatrices *DD*, *DV*, *VD*, and *VV*.

3.3 Fusion

In order to merge or fuse the two spaces, *V* and *D*, in consideration of the four distance constraint matrices, *VV*, *DD*, *DV* and *VD*, defined on them we require a set of transformations – scale, rotation, translation. For the time being we have only implemented scaling.

The four matrices *VV*, *DD*, *DV* and *VD* that make up *CM* were not created equally. They have been calculated from vectors with different lengths – n or m – and they may also have used different distance metrics *F*. We have observed that this inequality, if not compensated for, can lead to cases in which data samples and attributes may not mix well. That is, points due to the data samples and those due to the attributes may clump together into separate and disjoint communities.

Thus, transformations are necessary to enlarge or shrink the data or variable spaces. Suppose, we have the transformation θ:

$$D_\theta = \theta_D(D) \qquad V_\theta = \theta_V(V) \qquad (1)$$

where D_θ and V_θ are the transformed D and V, respectively.

There are different ways to define the θ. In order to mix the data and variables spaces well, we should balance the difference of each of the four matrices. One simple way to define θ or achieve this is to make the four sub-matrices (the entities in each submatrix) have equal mean. In this way, the two spaces have equal scale. In addition, in order to keep the distance matrixes unite, we make the *DV* and *VD* also have these equal scales

$$\overline{D_\theta D_\theta} = \overline{D_\theta V_\theta} = \overline{V_\theta D_\theta} = \overline{V_\theta V_\theta} \qquad (2)$$

where the $\bar{}$ operator denotes the mean of the distance matrix. There are different options to make these four distance matrices have the same mean (or L_1 norm) – we can use a linear, polynomial, or kennel function. A linear function has the advantage that it preserves the distribution, topology, etc. and thus, for this paper we apply a linear weight adjustment for each submatrix. In this way,

the transform is a simple weight adjustment for each submatrix. The weights are obtained as:

$$W_{DD}:W_{DV}:W_{VD}:W_{VV} = \frac{M_{max}}{D_\theta D_\theta}:\frac{M_{max}}{D_\theta V_\theta}:\frac{M_{max}}{V_\theta D_\theta}:\frac{M_{max}}{V_\theta V_\theta} \quad (3)$$

where W is the weight for the submatrix and M_{max} is the maximum mean of all the submatrices.

3.4 Mapping

With the composite distance matrix *CM* in hand, the final step is to create the joint map of samples and attribute points. We have opted to use an optimization approach for the map layout, as opposed to a linear projection with PCA or biplots since it gives us more freedom in choosing the constraints governing the layout, such as mixed distance functions, layout schedules, and mapping criteria. There are a number of distance-preserving optimization algorithms applicable for our purposes. LLE produces locally optimal layouts, while MDS-type schemes create globally optimal layouts which have become more popular in recent years since they provide a consistent overview of the data. Finally, t-SNE or linear discriminant analysis (LDA) [9] excel in their ability to isolate individual clusters, but they have a reduced ability to preserve the statistical appearance of the clusters which we feel is important for visualization. We have therefore chosen a metric MDS approach. Particularly useful here is the iterative and progressive point insertion schedule of Glimmer MDS [16]. We have adopted this multi-level scheme for our framework since it allows us to implement a variety of strategies for controlling the layout.

One of these strategies makes use of the weighting scheme for handling the submatrices of *CM*, as proposed in the previous section. It results in a rather general framework and offers much freedom to design a visualization that fits current criteria of interest. Users can simply assign the default weights that give equal emphasis to all submatrices or they can increase the weight for one of more submatrices that influence those aspects they would like to focus on. For example, a user might want to have an accurate representation of the relationships among the samples and of the samples to the variables but is less interested in an accurate representation of the relationships the attributes have with one another. So he/she would increase W_{DD}, W_{DV} and at the same amount W_{VD}, but reduce W_{VV}. Reducing one or more constraints will enable the mapping algorithm to trade the precision losses incurred for these unimportant relations in favor of those that are less desirable. Essentially, it serves as a buffer of the errors that are incurred with the necessarily imperfect space embedding. Section 4.1.1 describes another mechanism by which users can express their emphases – the scheduling of the data/variable primitives in the MDS-like layout.

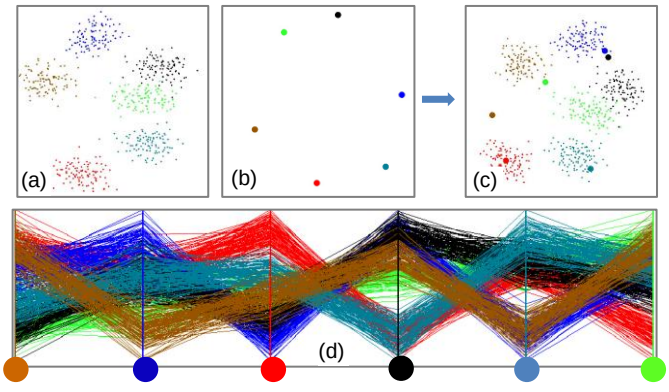


Fig. 6: Layout experiment for the 6 Gaussian test dataset. (a) MDS layout of the data samples (Euclidian distance); (b) MDS layout of the attributes (correlation distance); (c) parallel coordinate display with node colors marked; (d) MDS layout of samples and attributes using the CM matrix (samples: Euclidian, attributes: correlation).

3.5 A First Example

Fig. 6 shows a first result achieved with this mapping using the 6 test Gaussians introduced in Section 2.2. Fig. 6a is the MDS layout for just the data samples using the Euclidian distance metric; Fig. 6b is a MDS layout for the attributes using Pearson's correlation distance [33]; Fig. 6c is the layout created with MDS using the entire *CM* matrix and weights set to not give emphasis to any *CM* submatrix, and Fig. 6d is the parallel coordinate display for this dataset with the axes marked with the colors used for the attribute nodes in Fig. 6b and c. We used the (1-value) distance for the *DV* and *VD* submatrices.

We first observe that the layout of the clusters in the sample-only MDS plots has been well preserved in the *CM*-based MDS layout. On the other hand, the locations of the attributes, while still largely isolated to account for the correlation differences, have changed and better match the associations they have with the data clusters. This shows that the fusion of the two spaces *D* and *V* is not just a trivial superposition of the two plots.

Some more specific observations we make are: (1) the red cluster has a clear dominance in the red attribute and indeed its dimension node gets mapped right into the red cluster's center, (2) the green and the brown cluster both have high values in the green attribute and so the green attribute's node gets mapped between these two clusters, (3) similar is true for the brown attribute and the red and brown data clusters; (4) the dark blue and black attributes have somewhat similar (but switched) relationships with respect to high values of the black and dark blue clusters and so they get mapped more closely to each other right between these two clusters.

On closer inspection of Fig. 6 it appears that lower levels in the attributes are being taken into lesser or no account in *CM*'s layout. This can be explained by the distance metrics we chose for this particular case. The preference of the algorithm in picking attribute locations with respect to high values of the data clusters is due to the (1-value) distance we selected for the *DV* and *VD* submatrices. The behavior would change had we chosen a different distance. This and other choices, as well as their effects, largely depend on the aspects in the data the analyst would like to emphasize. Here in this example the emphasis was on extreme values.

4 Constructing the Data Context Map (DCM)

In this section we provide more details on the map construction and its segmentation into regions of similar properties.

4.1.1 Populating the map

The submatrices of *CM* can not only be weighted differently during the MDS layout, we can also impose different MDS schedules for the samples and the attribute points. We take advantage of this concept to achieve layouts with different priorities.

We require an iterative MDS algorithm to achieve this goal. Iterative MDS algorithms often do not update all points simultaneously at each step. Rather, they select a subset of points that is allowed to move, while another stays put, either indefinitely after an initial layout or the point sets alternate. The point sets can also be transient and can change over time. A particularly convenient algorithm in this regards is the Glimmer MDS (G-MDS) algorithm [16]. It has a stochastic force algorithm which iteratively moves each point until a stable state is reached. The forces acting on a point are based on a *Near Set* of points and a *Random Set* of points. The Near Set contains those points that are nearest to the point being updated. The Random Set contains points that are randomly chosen from the set of available points. It ensures some global control in the update process. We have altered the standard Glimmer MDS framework in two ways. First, we manipulate which types of points – variables or data – are allowed to be chosen for the Random Set. Second, we manipulate which types of points are allowed to be updated. Both change the local minimum of G-MDS as it is a metric MDS scheme using non-convex optimization.

Using this flexible update scheme we currently provide four MDS schedules: (1) Update the variables and the data points simultaneously (M-MDS); (2) Map the variables first, then fix them and only map the data (VF-MDS); (3) Map the data first, then fix them, and only map the variables (DF-MDS); (4) the user defined order (U-MDS). We describe each of these in turn in more detail.

(a) Update all types of points simultaneously (M-MDS)

This first schedule is the most general. It only runs G-MDS once and both types of points can be in the Random Set. See Fig. 7a.

(b) Update variables first, then the data (VF-MDS)

Here the goal is to achieve a layout that prioritizes the fidelity of the variable-variable (V-V) distances. It runs G-MDS two times. In the first run only the V-points are entered into the G-MDS point set. This results in an accurate V-layout. Then we run G-MDS the second time with the V-points frozen. Essentially, we add a statement that disallows the selection of a V-point for update, that is, only the data points (D-points) are allowed to move. Since this has the tendency to drive the D-points away from the V-points we only allow V-points in the Random Set. This preserves the influence the V-points have on the layout of the D-points. See Fig. 7b.

(c) Update data first, then the variables (DF-MDS)

This is essentially the reverse of the VF-MDS scheduling scheme and prioritizes the fidelity of the data-data (D-D) distances. This schedule also has two stages. First G-MDS is run in the D-points only. Next, G-MDS is run on the V-points with the D-points frozen and the D-points are only allowed in the Random Set. See Fig. 7c.

(d) User-defined iteration schedule (U-MDS)

The three schemes just presented are very basic update schedules and maybe there are better ones. For this purpose we allow users to draw a customized schedule via a timing (iteration) diagram editor. It first runs the VF-MDS schedule for a few iterations, then the DF-MDS schedule, and finally the M-MDS schedule. See Fig. 7d.

(e) Comparing the schedules

Comparing the layouts achieved with the different schedules we observe that for VF-MDS the variable to variable error is lowest and for the DF-MDS the data to data error is lowest. It also appears that M-MDS and U-MDS are good compromises. It depends on the user's priorities which method to choose. Considering the accuracy and complexity, we normally choose the first schedule, but the others are also useful for different preferences.

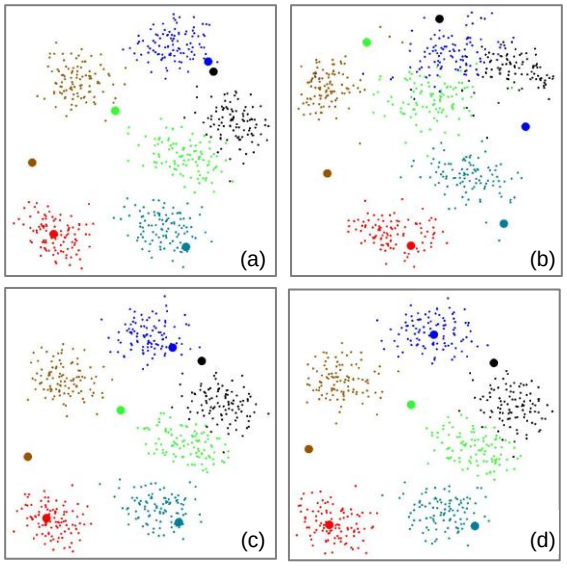


Fig.7. MDS layout schedules. (a) M-MDS layout. (b) VF-MDS. (c) DF-MDS (d) U-MDS.

4.1.2 First use case for the car dataset

We use the UCI Auto MPG dataset for our first non-toy example. This dataset has 392 cars built 1983 or older with 7 attributes – MPG, #cylinders (CYL), horsepower, weight, acceleration, year, and origin (US, Japan, Europe). Note that acceleration is the time a car requires to reach 60 mph and so slower cars have higher values. Fig. 8 shows a data context map generated via M-MDS. In this map, the large red points represent the attributes while the small blue points represent the cars. Cars that locate close to a given attribute node have high values for this attribute. On the other hand, cars that locate far away from a certain attribute node have a low value for it.

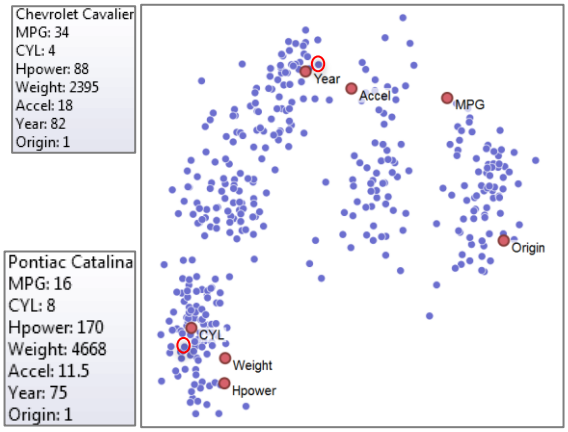


Fig.8. The data context map for the car data.

We observe that there are two main populations of correlated attributes. On one side there are horsepower, weight, and CYL, and on the other there are acceleration, mpg, and year. Origin is somewhat separate. We can also observe four distinct clusters of cars (with some sub-clusters) which are all heavily elongated in the vertical direction. Their relation with the attributes reveals that each cluster has a fairly large diversity in car attributes. Using the attribute nodes as landmarks we can now gauge the types of cars these clusters contain. For example, the cluster in the lower left contains the large high-performance cars with high horsepower and weight. The other clusters are more difficult to judge since they are so elongated and span a large attribute interval.

The map can be readily used for informed selection tasks. The user would simply look for features he is most interested in (or not at all), observe how many cars are actually available that have the desired feature constellation, and then select cars near these attributes (or far away depending on preference). For example, the user may be interested in a full-sized car, clicks on a node in that region on the map, and uncovers a 1975 Pontiac Catalina which is an entry-level full-size car (red-circled sample node in bottom cluster). Or he may be interested in a newer economic car and so selects a node close to the year attribute and fairly close to the mph and acceleration attribute. He correctly finds a newer (for the dataset) 1982 Chevy Cavalier which is an economy-grade compact car (red-circled sample node in top left cluster).

4.1.3 Error evaluation

Since our data context map is a 2D optimized layout, there is necessarily an error. As in every layout scheme we can estimate the error by comparing the distance in the matrix *CM* with the corresponding Euclidian distances in the 2D layout. We can use $\widetilde{DD}$, $\widetilde{DV}$, $\widetilde{VD}$ and $\widetilde{VV}$ to store the 2D layout distances, respectively. A popular metric to summarize the layout error is *stress* [23]. The error E in each sub-matrix is:

$$E_{IJ} = \sqrt{\frac{\sum_{(i\in I, j\in J)}(IJ_{ij} - \widetilde{IJ}_{ij})^2}{\sum_{(i\in I, j\in J)} IJ_{ij}^2}} \quad I,J \in \{D,V\} \qquad (4)$$

The overall error E_A is also weighted based on different blocks,

$$E_A = \sum_{I,J\in\{D,V\}} \beta_{IJ} E_{IJ} \quad (5)$$

where β_{IJ} is the weight. Typically, we set the β_{IJ} to:

$$\beta_{DD} : (\beta_{DV} + \beta_{VD}) : \beta_{VV} = 1:2:4 \quad (6)$$

As mentioned in Section 2.2, we have used the GBC plot as the standard formulation to describe the set of interior displays, and we also presented an optimized plot to improve the GBC plot error [7], called DIFGBC. In this section we compare this error with the one we can now reach with the data context map (DCM).Table 1 below compares the error for three datasets we studied.

We find that E_{VV} improves greatly – this is because the interior layouts map the variables to the 1-dimensional space (the boundary of the enclosing shape) but the DCM maps them into 2-dimensional space which naturally incurs less error. The E_{DD} error also greatly improved, but the E_{DV} error did not or even grew slightly for these examples, but this is also dependent on the update schedule, the distance metric, and the weighting. Yet, the overall error improved greatly and this quantitatively shows our data context map is more accurate than the competing interior layouts even when optimized.

Table 1. Comparing the error of the optimized GBC plot and the DCM

DataSet	Layout	E_{VV}	E_{DV}	E_{DD}	E_A	Up%
Car	DIFGBC	0.34	0.25	0.23	0.3	36.7%
	DCM	0.16	0.27	0.17	0.19	
University	DIFGBC	2.07	0.32	0.49	1.35	71.1%
	DCM	0.38	0.41	0.36	0.39	
Campaign	DIFGBC	0.33	0.26	0.31	0.31	25.8%
	DCM	0.22	0.3	0.16	0.23	

4.2 Segmenting the Map

The data context map as presented so far already allows attribute-informed selection of data objects, as we have demonstrated in Section 4.1.6. But it was somewhat difficult to judge the different value regions for combinations of attributes. This would be easy if the map could be somehow colored into distinct spatial areas which then could each be tagged by the respective attribute value combinations. To achieve this goal we require a continuous representation of the map. We have used adaptive kernel density estimation (AKDE) for this purpose.

4.2.1 Adaptive Kernel Density Estimation (AKDE)

The AKDE [21] is a method for estimating the density of a point cloud. It first estimates the local density of each sample and then shrinks or enlarges the sample's bandwidth. Suppose we have N points and each point is marked as P_i with a fixed bandwidth H. For any point P, its local density f is obtained by:

$$f(P) = \frac{1}{N}\sum_{i=1}^{N} K_H(\|P - P_i\|) \quad (7)$$

where $\|.\|$ is the L_2 distance. We can then estimate the local smoothing parameter λ_i and from it, the new bandwidth H_i for adaptive smoothing:

$$\lambda_i = (G/f(P_i))^2 \qquad H_i = H \times \lambda_i \quad (8)$$

where G is the geometric mean of all the samples local density.

The adaptive bandwidth of the AKDE kernels makes sure that small dense regions are preserved and not over-smoothed while less dense regions are properly fused. Fig. 9 compares fixed kernel density estimation (KDE) (Fig. 9a) with AKDE (Fig. 9b) for the car dataset. In this figure the brighter values correspond to lower values and vice versa. Consider the regions pointed to by the yellow arrow where we can see two separate regions for the AKDE, while these

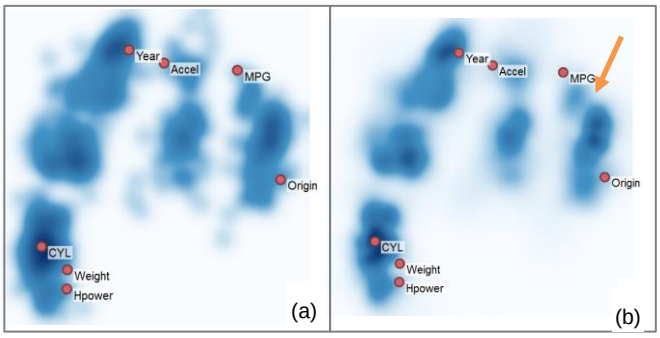


Fig.9.The KDE (a) and AKDE (b) show the density of the data points respectively.

regions appear mixed together for the KDE. There are also other examples in the map where AKDE gives a more accurate estimate of the local density.

4.2.2 Creating the attribute distance field using AKDE

We estimate the values in the continuous map based on adaptive kernels – when the point has a higher density, it would have lower bandwidth to shrink its effect area, and vice versa. Then, based on the adaptive kernel distance, we use Nadaraya-Watson kernel regression [27][35] to obtain the estimated value. Suppose the value at P_i is x_i, then the value x at the estimated point P is

$$x = \sum_{i=1}^{N} \frac{K_H(\|P - P_i\|)\cdot x_i}{\sum_{j=1}^{N} K_H(\|P - P_j\|)} \quad (9)$$

where K_H is the kennel function. Here we choose Gaussian function. However, some areas on the 2D canvas are far away from the samples and are therefore undefined. Thus it is important to control the border of the map and remove these undefined areas. We set the threshold ε for the sum distance – if the estimated point is far away from all the samples, we ignore it.

$$\sum_{j=1}^{N} K_H(\|P - P_j\|) \geq \epsilon \quad (10)$$

Fig. 10 shows how we convert the point map into a distance heatmap using AKDE. Here we first color the data points based on their values (here, of the horsepower attribute, Fig. 10a) and then generate the heatmap based on AKDE-based interpolation (Fig. 10b). We can see that the AKDE can estimate the values well and the border of this heatmap is also well defined.

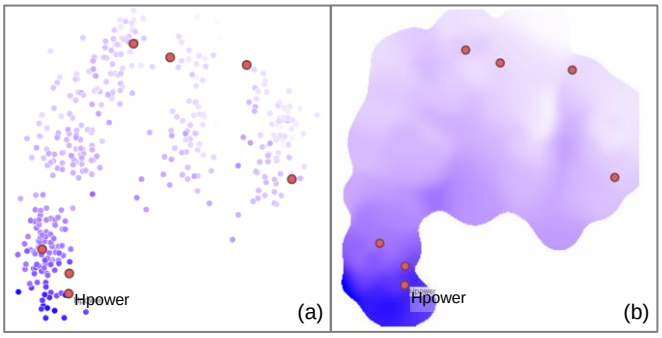


Fig.10. The samples' (a) "Horsepower" values and (b) its heatmap.

4.2.3 Creating the contour fields

Just by using the distance heatmap alone it is difficult to make out actual values. A common technique to visualize distance fields is via topographic maps. Then if a point is within a certain pair of iso-contours we can easily read off its value. We generate these contours via the conrec algorithm [1]. Fig. 11a shows the contour field of Fig. 10a, for the horsepower attribute. We observe that the contour region's value decreases level by level as we move away from the attribute node.

The contour field can also compare the layouts generated with standard MDS and our DCM. Fig. 11b shows the contour field generated from a distance heatmap based on a standard MDS layout (also using Glimmer MDS but without using attribute points). We find that the contour field has a rather ragged appearance with many more islands than the one generated from the DCM. In the data context map, on the other hand, the attribute nodes attract high-valued points and push low-valued points away. This magnetic force organizes the samples and so a smooth distance field can be created.

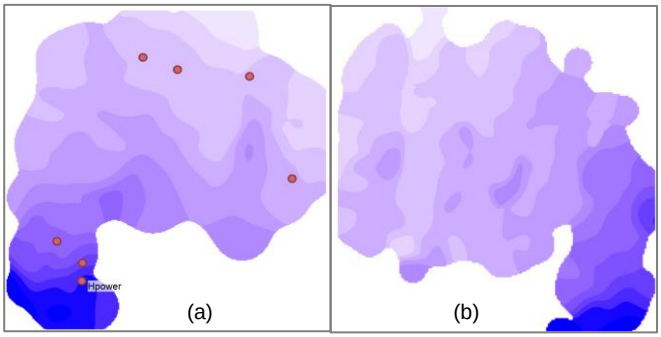


Fig.11. The data context map contour (a) and MDS plot contour (b).

4.2.4 Creating the decision regions

Each attribute gives rise to a set of contours, and a closed range of attribute values gives rise to a filled region between the two corresponding contours. Fig. 12a shows such a region for the horsepower range (120~230). We emphasize that this region has been computed from the value field generated by the actual data samples and so any sample selection that is based on it will be accurate. As such any of the cars that get mapped into the salmon-colored region in Fig. 12a indeed has a horsepower value in it.

Fig. 12b shows the iso-region for the (15-46) mpg value range which we can obtain in a similar fashion. This purple region contains all cars that have a mpg rating in that range. Next we can superimpose these regions to create the joint map shown in Fig. 12c. This joint map has three regions. The first is due to the original horsepower range only, the second is due to the original mpg range, and the third, overlapping region blending into a darker salmon color, contains cars that fit both value ranges. So if we wanted a car that fits both criteria we would pick a car from this overlap region.

Finally, we add a third constraint – origin. Origin is a discrete variable and we select the value 2 – the European cars. This gives rise to the green region in Fig. 12d. Blending it with the horsepower-mpg joint map creates the triple-attribute joint map shown in Fig. 12e. Now if we wanted to buy a car that is European and fits the other two range constraints we would look into the olive green region on the lower right. There are still some choices. We could pick a car on the upper boundary of that region which would be a more efficient car but with less horsepower. There is a car that fits the bill, which has been circled in the figure. Alternatively, we could pick a car from the left region boundary which would be a less efficient car but with a bit more muscle. There is also a car that fits this preference. However, if we sought to find a car that represents a compromise of mpg and horsepower – one that falls right into center of the region – we learn from the map that there is no such car in the database. There are obviously many more

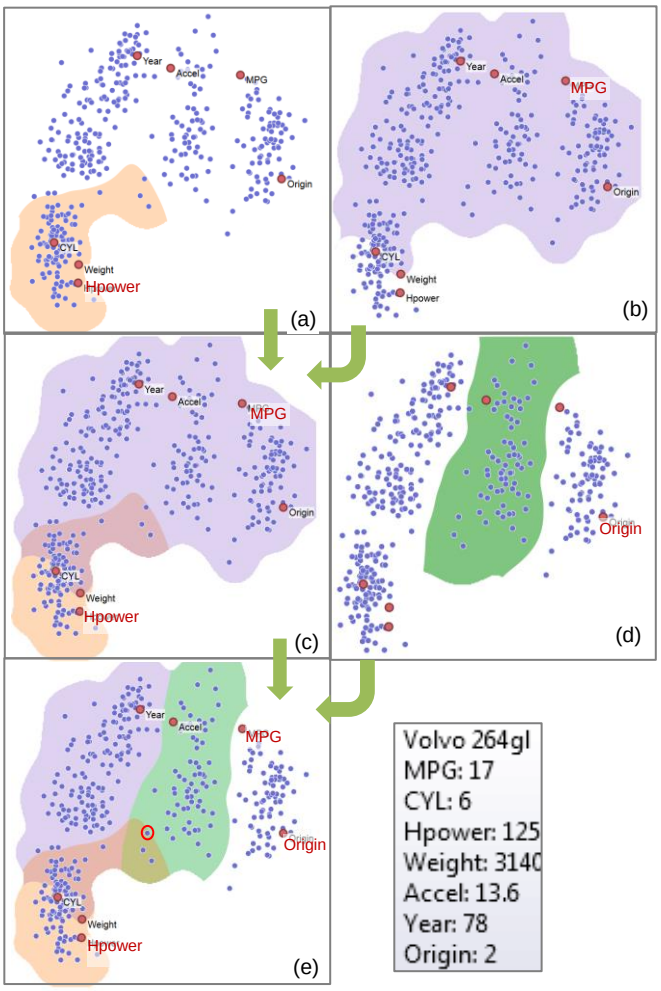


Fig.12. The decision of (a) "Hpower" (120~230), (b) "MPG" (15~46) and (d) "Origin" ("European") and (c) (e) their merge process.

explorations we can do with this map in hand. Since the interface is fully interactive, the user is free to modify his preferences in real time and fit the map to these preferences.

4.2.5 Creating a fully segmented and self-labeled map

Now suppose we have k attributes and each attribute can be divided into l_k levels based on users' preference. For example, these levels can be high level, middle level, low level etc. Then we can encode the entire area and see the combination of these attributes. Each region can then be encoded as $[R_1, R_2, \dots, R_k]$, where R_i represents the level in the each factor i and $R_i \in [0, \dots, l_i]$. We can divide the domain based on these codes and color the regions. However, it is important to maintain the color connection such that users can read the combination of different colors. We first assign the color for each attribute and let distances between colors are as big as possible. We set the intensity of each color based on the contour range level. Finally, when a region is composited we blend these colors.

Fig.13 shows an example for three attributes horsepower, mpg, and origin. We choose two levels - low or high (we set 40% as the threshold). For origin we split the set between Euro-Japanese cars and US cars. We give each attribute the color shown inside the attributes' symbols and then color the entire domain via color blending. We can now color each of the regions depending on levels of the participating attributes. The legend below the figure lists a human-created annotation for the regions. However, such a labelling could also be done automatically, using the levels of the attributes in each region to support natural language generation.

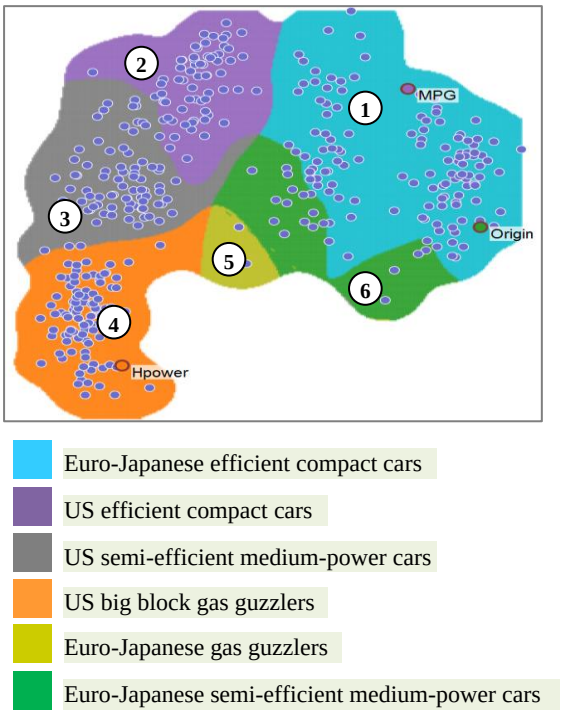


Fig.13. The fully segmented and self-labeled map based on "Horsepower", "MPG" and "Origin".

5 Case Studies

5.1 Selecting a College

Let us now return to the scenario we mentioned in the introduction – selecting a college. Our database has 46 universities distinguished by 14 attributes of interest: academics, athletics, housing, location, nightlife, safety, transportation, weather, score, tuition, dining, PhD/faculty, population, and income. Now suppose there is a prospective student, Tom, who is looking for a university. He aims for a school that has high athletics (>9), high academics (>9), but low tuition (<$18,000). He searches the universities with a traditional browser, but sadly he cannot find one which can meet all three requirements at the same time. He knows that he needs to make a compromise, trading off a few factors, and find the school that offers the right balance. This, however, he finds hard to do because he does not even know what his personal good balance really is. He wants to see "what's out there" and get inspired. So he calls up the data context map to immersive himself into the landscape of schools to find the elusive balance.

He begins by generating the decision boundaries based on his three criteria. This is shown in the teaser image, Fig.1a-d. Then he merges them and gets Fig. 1e. He (once more) recognizes that there is no university that can satisfy all three criteria at the same time – for example, the green tuition region does not overlap with the two other regions simultaneously. But now he sees in one view what his options are. He notices a few schools that meet two of his conditions – those schools that fall into two-layer overlap areas. He picks a few that are closest to the third layer at "just the right distance" as he describes it – the schools labelled A, B, and C in Fig. 1e He says that he likes A "because it has good athletics and low tuition, while the academics is not stellar but alright". Similarly, B is good and he'd be "OK with paying a bit more tuition for the great value." Finally, school C has good academics and low tuition which is great because he "could just use the savings to buy a big screen TV to watch the games of other schools". Nevertheless, he picks A and lives happily ever after.

5.2 Analyzing the business priorities

Our second case study demonstrates the utility of the fused display of samples and attributes. We peek into an analysis session of a group of top level managers of a multinational company with many subsidiaries in different countries. The topic is to determine the different priorities these companies have when it comes to sales strategy and long term goals. They have 600 samples of sales team data with 10 attributes: #Leads (generated), #Leads Won (LW), #Opportunities (generated), Pipeline Revenue (Rev), Expected Return on Investment (EROI), Actual Cost (Cost), Cost/WonLead (Cost/LW), Planned Revenue (Rev), and Planned ROI (PROI). The highly paid visual analytics consultant the firm has hired pulls out the data context map and with a few mouse clicks produces the visualization shown in Fig. 14. It is quickly seen that there are three clusters, call them red group, blue group, and green group. It turns out these three groups have rather different strategies and priorities.

The red group's focus is dominated by #Opp, PROI, Cost/WL, where they have high values and achievements. At the same time, however, they score very low in #Leads, LW, Rev, etc. The members of this group tend to focus on the individual leads and invest a lot in these, and as a result they have usually a high number of opportunities. The blue group, in contrast, are possibly larger companies – they have high revenue and they can generate a large amount of Leads. The green group is dominated by PRev and %Comp. Since they have high expected revenue, their %completed is high. But clearly all groups have one thing in common – cost. This factor has equal distance for all of them. It means they all care about the cost with similar weights.

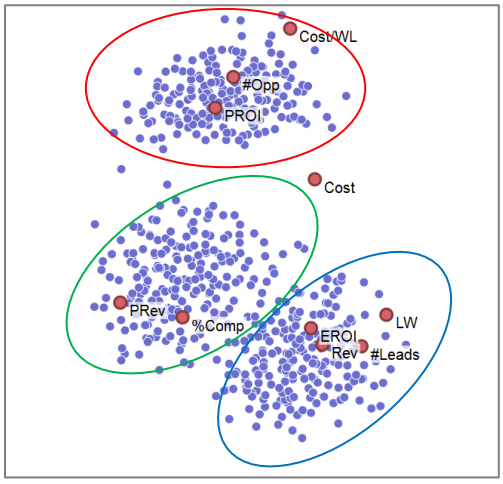


Fig.14. The data context map for the business priority case study.

6 Conclusion

We have described the data context map – a framework and visual interface that enables a comprehensive layout of both data points and variables. We achieve it by fusing two distance matrices – the data and the attribute distance matrix. We create an optimized layout that can be used for in data-driven decision selection and decision problems that require a mindful balancing of trade-offs.

While we provide several parameters for experts to guide the layout for their goals, they are not essential to produce usable results. Casual users can just use the pre-set weights, upload the data, generate the initial map, and interact with the value sliders. Future work will explore how casual users actually do this.

Acknowledgements

This research was supported by NSF grant IIS 1117132 and by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the "ICT Consilience Creative Program" supervised by the IITP (Institute for Information & Communications Techn. Promotion)".

REFERENCES

[1] P. Bourke, "CONREC: A contouring subroutine," *Byte: The Small Systems Journal* 12 (6): 143–150, 1987.

[2] I. Borg and P. Groenen (2005). *Modern multidimensional scaling theory and applications*. 2nd edition. New York: Springer.

[3] B. Broeksema, A. Telea, T. Baudel, "Visual Analysis of Multi-Dimensional Categorical Data Sets," *Computer Graphics Forum*, 32(8): 158-169, 2012.

[4] B. Broeksema, T. Baudel,, A. Telea, P. Crisafulli, "Decision exploration lab: A visual analytics solution for decision management," *IEEE Trans. on Visualization and Computer Graphics*, 19(12), 1972-1981, 2013.

[5] J. Chambers, W. Cleveland, P. Tukey, *Graphical Methods for Data Analysis,* Duxbury Press, 1983.

[6] S. Chen, D. Amid, O. Shir, L. Limonad,D. Boaz, A. Anaby-Tavor, T. Schreck, "Self-organizing maps for multi-objective Pareto frontiers," Proc. *IEEE Pacific Visualization Symposium*, pp.153-160, 2013.

[7] S. Cheng, K. Mueller, "Improving the Fidelity of Contextual Data Layouts Using a Generalized Barycentric Coordinates Framework," *Proc. Pacific Vis*, pp. 295-302, 2015.

[8] J. Choo, S. Bohn, H. Park. "Two-stage framework for visualization of clustered high dimensional data," *Proc. IEEE Visual Analytics Science and Technology Conference (VAST)*, pp. 67-74, 2009.

[9] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, 2000.

[10] K. Gabriel, "The Biplot Graphic Display of Matrices with Application to Principal Component Analysis," *Biometrika*, 58(3): 453-467, 1997.

[11] G. Grinstein, M. Trutschl, U, Cvek, "High-dimensional visualizations," *Proc. Visual Data Mining Workshop, KDD*, 2001.

[12] J. Hartigan, "Printer graphics for clustering," *Journal of Statistical Computation and Simulation*,4(3):187-213, 1975.

[13] P. Hoffman, G. Grinstein, K. Marx, I. Grosse, E. Stanley, "DNA Visual and Analytic Data Mining", *Proc. IEEE Visualization*, pp. 437-441, 1997.

[14] P. Hoffman, G. Grinstein, D. Pinkney, "Dimensional anchors: a graphic primitive for multidimensional multivariate information visualizations," *Proc. Workshop on New Paradigms in Information Visualization and Manipulation*, pp. 9-16, 1999.

[15] https://colleges.niche.com/

[16] S. Ingram, T. Munzner, M. Olano, "Glimmer: Multilevel MDS on the GPU," *IEEE Trans. Visualization and Computer Graphics*, 15(2): 249–261, 2009.

[17] A. Inselberg, B. Dimsdale, "Parallel Coordinates: A Tool for Visualizing Multi-Dimensional Geometry," *Proc. IEEE Visualization*, pp. 361-378, 1990.

[18] P. Joia, D. Coimbra, J. Cuminato, F. Paulovich, L. Nonato, "Local Affine Multidimensional Projection", *IEEE Trans. Vis. Comput. Graph.* 17(12): 2563-2571 (2011).

[19] I. Jolliffe, "Principal Component Analysis," *Series: Springer Series in Statistics*, 2nd ed., Springer, NY, 2002, XXIX, 487 pp.28 illus. ISBN 978-0-387-95442-4.

[20] E. Kandogan, "Star Coordinates: A Multi-Dimensional Visualization Technique with Uniform Treatment of Dimensions," *Proc. IEEE Information Visualization, Late Breaking Topics*, pp. 9-12, 2000.

[21] P. Kerm, "Adaptive Kernel Density Estimation," *The Stata Journal*, 2:148–156, 2002.

[22] T. Kohonen. *Self-Organizing Maps*. Springer, 3rd edition, 2001.

[23] J. Kruskal. M. Wish, Multidimensional Scaling. *Sage Publications*, 1977.

[24] J. Lee, K. McDonnell, A. Zelenyuk, D. Imre, K. Mueller, "A Structure-Based Distance Metric for High-Dimensional Space Exploration with Multi-Dimensional Scaling," *IEEE Trans. on Visualization and Computer Graphics*, 20(3): 351-364, 2014.

[25] G. McLachlan. Discriminant Analysis and Statistical Pattern Recognition, *John Wiley & Sons*, Aug 4, 2004.

[26] J. Nam, K. Mueller, "TripAdvisorN-D: A Tourism-Inspired High-Dimensional Space Exploration Framework with Overview and Detail," *IEEE Trans. Visualization and Computer Graphics*, 19(2):291-305, 2013.

[27] A. Nadaraya, "On Estimating Regression". *Theory of Probability and its Applications* 9 (1): 141–2. doi:10.1137/1109020, 1964.

[28] L. van der Maaten, G. Hinton, "Visualizing data using t-SNE," *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[29] M. Meyer, A. Barr, H. Lee, M. Desbrun, "Generalized Barycentric Coordinates on Irregular Polygons," *J. Graphics Tools*, 7(1):13-22, 2002.

[30] F. Paulovich, C. Silva, L. Nonato, "Two-Phase Mapping for Projecting Massive Data Sets", *IEEE Trans. Vis. Comput. Graph*. 16(6): 1281-1290 (2010)

[31] L. Saul , S. Roweis, "An Introduction to Locally Linear Embedding" *IJPRAI* 01/2009; 23:1739-1752. DOI: 10.1142/S0218001409007752.

[32] R. Spence, *Information Visualization*, Addison-Wesley 2000 ISBN: 0-201-59626-1.

[33] B. Tabachnick and L. Fidell, *Using Multivariate Statistics*, New York: Harper & Row, 2001.

[34] J. Tenenbaum, V. de Silva, J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science 290*, 2319–2323, 2000.

[35] G.Watson. "Smooth regression analysis". Sankhyā: *The Indian Journal of Statistics*, Series A 26 (4): 359–372. JSTOR 25049340, 1964.

[36] J. Yi, R. Melton, J. Stasko, J. Jacko, "Dust & Magnet: Multivariate Information Visualization using a Magnet Metaphor," *Information Visualization,* 4(4) : 239-256, 2005.

[37] Z. Zhang, K. T. McDonnell, E. Zadok, K. Mueller, "Visual Correlation Analysis of Numerical and Categorical Data on the Correlation Map," *IEEE Trans. on Visualization and Computer Graphics*, 21(2): 289-303, 2015