Interactive Exploratory Data Analysis

Sergey Malinchik, Belinda Orme, Joseph A. Rothermich, Eric Bonabeau
Icosystem Corporation
10 Fawcett St.
Cambridge, MA 02138
Email: {sergey, belinda, jr, eric}@icosystem.com

Abstract- We illustrate with two simple examples how Interactive Evolutionary Computation (IEC) can be applied to Exploratory Data Analysis (EDA). IEC is particularly valuable in an EDA context because the objective function is by definition either unknown a priori or difficult to formalize. The first example IEC is used to evolve the “true” metric of attribute space. Indeed, the assumed distance function in attribute space strongly conditions the information content of a two-dimensional display of the data, regardless of the dimension reduction approach. The goal here is to evolve the attribute space distance function until “interesting” features of the data are revealed when a clustering algorithm is applied. In a second example, we show how a user can interactively evolve an auditory display of cluster data. In this example, we use IEC with Genetic Programming to evolve a mapping of data to sound functions in order to sonify qualities of data clusters.

I. INTRODUCTION

At a time when the amount of data that is potentially available for analysis and exploitation is increasing exponentially in all fields and industries, the ability to explore quickly gigantic databases for valuable patterns is becoming a crucial competitive advantage. The problem is: what should one look for? Exploratory Data Analysis (EDA) [1,2] is the art of exploring data without any clear a priori ideas of what to look for. Most techniques used in EDA are interactive and visual. There are a number of display techniques that take care of the visualization aspect. As the dimensionality of the data (that is, the number of attributes) increases, informative low-dimensional projections of the data are necessary, most often in two dimensions. Families of techniques such as multi-dimensional scaling (MDS) [3] are used to generate such projections. While there exist powerful visualization techniques, the interactivity of EDA is often ad hoc and best characterized as tinkering. That is because in EDA the notion of “interestingness” of a particular display or pattern is, by definition, difficult to formalize. Thus, designing data mining patterns, filters, projection algorithms, clustering algorithms and other search models that will produce “interesting” results when applied to a dataset is a difficult problem [2].

If one assumes that interesting patterns will be recognized when they are discovered even though they could not be formulated ahead of time, one can use a technique originally developed to generate “interesting” images and pieces of art [4-7], known as Interactive Evolutionary Computation (IEC). Although IEC has been applied to some data mining problems in the past (see [8], for a review), it has never been described as a canonical tool to perform EDA. This paper is an attempt to show how pervasive the technique might become. IEC is a directed search evolutionary algorithm which requires human input to evaluate the fitness of a pattern (here, the fitness might be how interesting a detected pattern is) and uses common evolutionary operators such as mutation and crossover [9,10] to breed the individual-level patterns that produce the fittest collective-level patterns. IEC combines computational search with human evaluation [8].

As a first example, we show IEC applied to evolving parameters for clustering analysis. A secondary example of IEC applies to the sonification of data. Sonification is often used to represent events or alarms and has recently been the focus of data analysis [11,12]. We show the use of IEC to evolve the mapping of data to sound so that the user can guide feature extraction, sound shaping and other steps in the process of creating auditory displays of data.

II. THE PROBLEMS

We illustrate the use of IEC in EDA with two simple examples using five-dimensional real-valued datasets described in section II.A. In the first example, IEC is used to evolve the distance function in attribute space to produce the most compelling clusters using a simple parametric clustering algorithm. In the second example, we use IEC to evolve a basic example of data sonification for cluster analysis.

A. The dataset

The experimental results in example 1 were obtained using a synthetic, five-dimensional real-valued dataset described here. Example 2, interactive sonification, uses a similarly structured but simpler data set. The five dimensions are represented as (X1, X2, X3, X4, X5). The dataset contains 24000 points in five separate clusters.
The clusters are generated in the following way: each of the five coordinates \((x_1, x_2, x_3, x_4, x_5)\) of each data point is generated independently from a Gaussian distribution whose mean is equal to the corresponding coordinate of the center of the cluster that point belongs to, and whose standard deviations, listed below, reflect the cluster’s elongation or compression along certain dimensions. Some of the planar projections of some of the clusters are then rotated by a certain angle. The five clusters are defined below:

- **Cluster #1:** 3000 points, center located at: (650, 255, 540, 500, 300), standard deviations for each coordinate: (50, 40, 20, 200, 60); \((X_3, X_4)\) coordinates rotated by 20 degrees in the \((X_3, X_4)\) plane around point \((x_3=0, x_4=0)\).
- **Cluster #2:** 6000 points, center located at: (450, 200, 400, 500, 300), standard deviations for each coordinate: (50, 70, 150, 40, 60); \((X_3, X_4)\) coordinates rotated by 30 degrees in the \((X_3, X_4)\) plane around point \((x_3=0, x_4=0)\).
- **Cluster #3:** 7000 points, center located at: (400, 400, 500, 200, 300), standard deviations for each coordinate: (90, 50, 160, 40, 20); \((X_3, X_5)\) coordinates rotated by 90 degrees in the \((X_3, X_5)\) plane around point \((x_3=0, x_5=0)\).
- **Cluster #4:** 4000 points, center located at: (600, 550, 350, 300, 540), standard deviations for each coordinate: (80, 70, 20, 40, 80).
- **Cluster #5:** 4000 points, center located at: (350, 580, 600, 340, 600), standard deviations for each coordinate: (60, 70, 40, 40, 60).

Although the dataset is simple, discovering its cluster properties (# of clusters, locations, shapes) without **a priori** knowledge is a non-trivial task.

### B. Evolving distance functions in attribute space

Clustering is a useful and commonly used technique in EDA. The goal of clustering is to compress the amount of data by categorizing or grouping similar items together [13]. There are many different clustering algorithms. However, the clusters resulting from the application of one specific clustering algorithm to a data set are heavily dependent on the distance function assumed in attribute space. That distance function is rarely questioned and even more rarely an object of study. For example, when dealing with real-valued data, it is often implicitly assumed without further examination that the relevant distance function is the Euclidian distance or \(L_2\) (\(L_1 = \) city-block distance; \(L_2 = \) Euclidian distance; \(L_{\infty} = \) max norm). However, that might not appropriately reflect the potentially complex structure of attribute space. For example, the use of the Euclidian distance in attribute space for the dataset described in Section II. A does not lead to the extraction of the data’s clusters because of the elongation and compression of the clusters along certain dimensions. The problem here is to discover a distance function in attribute space that contains some of the fundamental properties of that space. To do so we apply a simple parametric clustering algorithm (K-means, well suited to the globular clusters of our dataset) to the data using a variety of distance functions and we then examine the resulting clusters; the distance function is evolved until the clusters look “right” or provide valuable information.

A commonly used clustering method is K-means clustering, which is a least squares partitioning method allowing users to divide a collection of objects directly into \(K\) disjoint clusters. The energy function \(E_k\) that is minimized in K-means is the sum of the squared distances between each data item \(x_m\) and the nearest cluster centroid:

\[
E_k = \sum_m \| x_m - c(x_m) \|^2
\]

where \(c(x_m)\) is the centroid that is closest to \(x_m\). The algorithm starts by initializing a set of \(K\) cluster centroids denoted by \(c_i, \ i=1,\ldots,K\). Data points are examined sequentially in a random order and each point is assigned to the nearest centroid. The positions of the centroids are then adjusted iteratively by re-computing the centroids to move them closer to the set of points that belong to the corresponding cluster. Several passes through data are performed until every data point is consistently assigned to one cluster, that is, until the assignment of points to clusters is stable.

K-means is characterized by simplicity and computational efficiency but it is sensitive to cluster shapes. In particular, it fails when clusters are too close to one another. When clusters are strongly anisotropic (for example, elongated or compressed along certain dimensions), it is helpful to define the distance function in such a way that it counterbalances cluster asymmetry, thereby revealing the structure of attribute space. The family of weighted Euclidean distance functions in \(R^N\) is explored using IEC. A distance function in that family is given by:

\[
d_m(x^p, x^q) = \sqrt{\sum_{i=1}^N w_i (x^p_i - x^q_i)^2}
\]

where \(x^p\) and \(x^q\) are two points in \(R^N\), and \(w = (w_1, w_2,\ldots, w_N)\) is the weight vector that characterizes that distance function, with \(0 \leq w_i \leq 1, \ i=1,\ldots, N, \ \text{and} \ \sum_{i=1}^N w_i = 1\).
The object of the IEC-based search is to evolve $w$ to extract valuable information from the clusters.

C. Evolving Sonification for Exploratory Data Analysis

Hermann [11] defines sonification as “the transformation of data relations into perceived relations in an acoustic signal for the purposes of facilitating communication or interpretation”. Sonification has been long been used for feedback and process monitoring, e.g., alarms, ECG, computer beeps, etc. Recently it has been used to create auditory displays of data [11,12]. Kramer [14] describes some of the advantages of sound as an analysis tool: It allows for eyes-free monitoring, rapid detection in high-stress environments, and parallel listening of high-dimensional data. The ears are very good at hearing trends and have acute temporal resolution allowing understanding patterns in time sequences. Having an affective response gives humans the ability to learn about and connect to sounds quickly. There are also some disadvantages of sounds such as a difficulty of users to identify precise frequencies, users perceive sound differently, no absolute references, and sound can be a tiring input for users.

Sound is created by a wave that can be described by a simple continuous sine function: $\alpha \sin(ft)$ where $\alpha$ is the amplitude, $f$ the frequency and $t$ the time dependence. A more complex sound can be created by additive synthesis, which is the addition of multiple sound waves:

$$\text{tone}(t) = \alpha, \sin(f_1t) + \beta, \sin(f_2t) + \gamma, \sin(g_1t)$$

Sampling a continuous sound function at a sampling rate creates a digitized sound, which can then be stored or heard by a user. Some work has been done in mapping data to frequencies in a sound function as a means to create an auditory display of the data. Hermann [12] has studied the use of principal curves to serve as the time domain for sound creation on a data set. Time-series data has been mapped to frequencies to a create sequences of sounds [15]. However, to our knowledge, the use of a user-guided evolution of sound functions to map data to sound is so far unexplored.

We propose the use of Interactive Evolution for the design of data sonification. Evolution can assist in two steps: (1) in feature extraction including the basic selection and combinations of dimensions to include in the target sound, (2) in filtering, convoluting, and reshaping of the data in the processing of the sound. This can be especially useful in situations such as EDA when the user may not know what they are looking for “until they hear it”.

Since sound is a time-dependent function, the application of sonification to time-series analysis comes naturally. However, in this example we study the use of sonification of cluster data as we previously explored visually. Clusters can be analyzed for density, size, shape, and variance and each of these attributes can be mapped to a quality of sound. The objective of using Interactive Evolution in this process is letting the user guide how data is used to create an auditory display and which quality of the data is emphasized.

III. Example 1: Interactive Evolution of Clustering Parameters

A. Overview of the search mechanism

The IEC search method works as follows. A small initial population of solutions is generated, where a solution is a distance function. The resulting two-dimensional representations are computed, generated and displayed to a human observer. The observer selects the displays that are the most interesting – the fittest individuals in the population according to whatever set of objective and subjective criteria the observer may be using, or assigns a fitness value to each display. A new population (new generation) of solutions is generated by applying mutation and crossover operators to the solutions that correspond to the previous generation’s fittest displays. In addition to the offspring and mutated versions of those solutions, randomly generated solutions are injected into the population to ensure diversity. The new population’s two-dimensional representations are then calculated and the results displayed to the observer, and so forth. This procedure is iterated until interesting displays emerge from the search, pointing toward corresponding linear projections or distance functions.

The user interface is a critical component of the method. The observer evaluates solutions based on visual inspection of their two-dimensional representations. The visual interface is shown in Figure 3. Obviously, this method can only work if the population size is kept small and if interesting solutions emerge after a reasonably small number of generations.

B. Genetic algorithm

The evolutionary mechanism for example 1 is as follows:

- A simple real-valued representation is used to encode the genotype. The genotype is a juxtaposition of 5 real-valued genes ($w_1, \ldots, w_N$ with $N=5$).
- Population size is equal to nine.
- Initial gene values are generated randomly from a uniform distribution over the allowed range.
- The user assigns fitness values to the various displays. The default value for all displays before user intervention is 0. The user can play with different views (two-dimensional projections onto
coordinate planes) of the data before assigning a fitness value. After visual inspection of the dataset, the user sets the number of clusters (K) and the tool randomly generates the starting positions of the cluster centroids in the original 5-dimensional space. Each of the nine solutions displayed represents the results of the application of the K-means clustering algorithm with a different distance function in attribute space, characterized by the weight vector w.

- The results are displayed simultaneously to the user. The user can move back and forth and reassign fitness values.
- The user can select one or more displays to be part of an elite pool. Unless otherwise specified by the user, only one elite individual is selected by the algorithm, which is the one with the highest fitness.
- By clicking the mouse on the corresponding image, the user indicates his choice and the tool automatically sets the fitness function to the value of 5. Double-clicking means the selection of the elite individual, which will assign a fitness value equal to 10. The user’s choice is indicated on the screen by a green or yellow frame around ordinary selected individuals or elite ones, respectively.
- The mating pool is comprised of solutions with fitness larger than zero.
- The next generation is created in the following way: one elite solution remains unchanged. Four solutions are created from the ranked mating pool. Two offspring are created from two randomly selected parents in the mating pool by applying a random single point crossover. The last four solutions are created from single parents from the mating pool after application of a single point mutation. The new value of the mutated point is drawn from a normal distribution function whose mean is equal to the current gene value and whose standard deviation equal to one third of the gene value range.

IV. EXAMPLE 2: INTERACTIVE EVOLUTION FOR SONIFICATION

A. Representation and Evolutionary Algorithm

In this example, we use Genetic Programming (GP) [16] to serve as the genetic representation for our sound functions. The concept of genetic programming fits well with creation of sounds by evolution, especially when we use functions such as [sin, *, +, -] which will easily create the functions we have illustrated in Section 0 as the building blocks of sound.

GP uses a Lisp-like structure to represent a parse tree that contains terminals and operators and is decoded to create a function. A sample GP function “\(\text{add}(\sin(\text{mul}(a,b)), \sin(\text{mul}(a,c)))\)” is decoded to the parse tree shown in Figure 1. This function is simply additive synthesis, here the sum of two sine functions, each containing the product of time and a data column. In this example, time and each data column are mapped directly to GP terminals. Using this mapping, GP can create a wide range of sounds and non-sounds.

Further, we can also consider the terminals in GP to be populated with data, which biases the search in the direction of creating audible sounds. For example, data could be preprocessed into time varying sine waves as shown in Figure 2 before being passed to the GP. A GP function would then simply have to use the addition function to replicate additive synthesis. The method in Figure 2 is used in the results shown below.

Each generation involves the creation of N functions, which are evaluated using preprocessed data and presented to the user as sounds. The user judges fitness of individuals by selecting which sound or sounds should be included in creating the next generation. Mutation and Crossover are performed based on the user’s selection as in standard parse-tree GP. Since each successive generation is created to contain traits similar to sounds the user selects, the search is guided towards meaningful auditory representations of the clusters.

B. Pre-processing and post-processing of sound data

Some amount of preprocessing and normalization of data is required for sonification. For example, specifics about the magnitude and range of the data should not cause the amplitude and frequency range of a sound to be out of audible range. A preprocessing algorithm can be
used which maps the mean of the data to an arbitrary
frequency, and then distributes data around this point.
For a time series, data can be mapped directly to a
frequency after scaling. Other techniques for time series
include mapping of data to a moving average,
differencing, and percentage change.

The second stage of preprocessing pertains to clustered
data. To extract information from clustered data, we
would like to know features of the cluster such as its
shape, spatial location, etc. Euclidean distance and
absolute distance across dimensions can be used as input
to sound functions in GP. The first method, using
Euclidean distance of each element from the cluster
centroid, would result in a single data column as input
to the GP. The second method, the one used for the
experiments discussed here, uses the absolute distance of
each dimension. This provides the ability to evolve
different mappings of dimensions to sound, and makes it
possible to evolve a more powerful feature extraction
function.

The GP function is evaluated with the preprocessed
data over a series of time steps. Each evaluation results
in a numeric value for the sound’s amplitude at a certain
time. The series of amplitude values over the sampling
time is the actual sound file that can be processed
presented to the listener. Before generating a final sound
file, some post-processing needs to be performed such as
scaling and shifting the wave to ensure that it remains in
an audible range.

V. RESULTS

A. Evolution of K-means Parameters

The user performs the interactive search for distance
functions by looking for valuable and consistent
information. In Figure 3 we present nine displays
resulting from the application of K-means clustering for
randomly generated values of the weight vector \( w \) (Figure
3, Top), and nine displays obtained after five generations
with interactively evolved \( w \). In both cases, we hold the
specific projection of the dataset constant. A criterion
that emerged as the main basis for the user’s selection of
the best distance functions is the presence of
homogeneously colored clusters. If a cluster is not
homogeneously colored or if it appears to be cut in the
middle, the user easily detects this spatial inconsistency
and assigns a low fitness value to such a solution.
Although there is a range of values of \( w \) that yield a good
clustering, all of the best solutions typically share the
same properties: \( w_3 \), \( w_4 \) and \( w_5 \) have significantly lower
values than \( w_1 \) and \( w_2 \), thereby indicating that the
attribute space has anisotropic properties. A good
solution found with IEC is: \( w_1 = 0.33 \), \( w_2 = 0.35 \), \( w_3 =
0.12 \), \( w_4 = 0.06 \) and \( w_5 = 0.12 \).

B. Cluster Sonification

The sonification of cluster data provides the user with
the ability to hear multiple dimensions in parallel and to
shape the auditory display being synthesized. The K-
means clustering algorithm described above was used in
the testing of interactive sonification. In this process, the
K-means parameters and cluster assignments were held
constant, while the auditory display of the information
was evolved. For this experiment, a five-dimensional
cluster was created with standard deviations of (5, 200,
0.05, 0, 10).

A sample generation of evolved sound functions for
these clusters is shown in Figure 4. Since it is not
possible to listen to the sounds here, each sound is shown as a spectrogram. The dimension with no standard deviation can be seen in the spectrograms that contain a solid line. This shows that the data was mapped to a pure tone. The dimension with a small standard deviation of 0.05 creates a pulse in the same way an instrument being tuned pulsates when it gets close to being in tune. It is possible to see the pulse in all spectrograms except 1, 4 and 8 (going top to bottom, left to right). There are also fuzzy areas in the spectrograms that include dimensions that had a high standard deviation. A Fuzzy image in the spectrogram means that the output sound will be noisy. After some practice, it became easy to recognize the amount of noise in a sound as a function of the cluster’s standard deviation. It was also fairly easy to pick out the presence or absence of each dimension in a sound.

Figure 4: Spectrograms of Evolved Sound Functions

The sound functions evolved in Figure 4 show that it is possible to evolve functions that combine and select certain data dimensions for the listener. For example, the bottom left spectrogram was created by an evolved function, which was simply $\text{add}(X_3, X_4)$. This function generated a combination of two dimensions, dimension $X_3$, which had a standard deviation of 0.05, and dimension $X_4$, which had zero standard deviation. Even though this is a very simple evolved function, it demonstrates that features of cluster data can be sonified and a mapping of data to sound can be evolved to aid in data exploration.

VI. DISCUSSION

We have illustrated with two simple examples how IEC can assist in EDA. We believe that IEC has the potential of becoming a canonical EDA tool, helping to structure the user’s intuition and insights in situations where the value of the results from a data-mining algorithm is not known ahead of time and/or is difficult to formalize. There are obviously limitations to what can be achieved with IEC since it relies on small populations and a small number of generations (Takagi, 2001). Sonification of data is also difficult in that it requires a careful processing of data and can result in sound outside of the audible range. One can however increase the potential of an IEC search by combining it with an automated pre-processing or filtering of the solutions when objective, formalizable criteria and constraints are known, so that only pre-processed solutions are presented to the user. Another useful extension would be to give the user the ability to explicitly select sub-structures or sub-modules of the display that seem promising.

REFERENCES


Display, 81--86, Eds.: P. R. Cook, Int. Community for Auditory Display.


