Clustering High Dimensional Sparse Casino Player Tracking Datasets

Ross Iaci[1]
Ashok K. Singh

Abstract

In this article, we propose an iterative procedure for clustering sparse high dimensional transaction datasets, specifically two casino player tracking datasets. A common problem in clustering sparse datasets with very large dimensions is that in addition to classical techniques of clustering being unable to provide useful results, latent variable methods used for clustering often do not lead to sufficient data reduction to yield useful and informative results either. Initially, we propose a straightforward resorting of the full dataset and then define an information based sparsity index to subset the sorted data. This new dimension reduced dataset is less sparse, and thus, more likely to produce meaningful results using established techniques for clustering. Using this technique enables the clustering of two secondary datasets from two Las Vegas repeater market casino properties, which consist of the amount of money casino patrons gambled, termed coin-in, on a variety of slot machines.
Key Words and Phrases: Cluster Analysis; Transaction Data; Binary Entropy; Sparse Matrices; Casino Player Tracking.

The volume of customer information collected by businesses is increasing at an astronomical pace.

1. Introduction

The volume of customer information collected by businesses is increasing at an astronomical pace. In 1992 the retail store giant Walmart started with a database of one terabyte for its 3, 600 U.S. stores. This database grew to four petabytes by 2007 as the number of stores increased to a little over 6, 000. The Casino industry has not lagged in this area. Harrah's started the first nationwide player loyalty program named Total Gold in 1997, which transformed into the tiered loyalty program Total Rewards in 2000. Among many of the goals for the loyalty program data is to build customer profiles based, for example, on what games casino patrons are likely to play, how much they play or even their gaming ability (Berman, 2006).

Currently, large retail store chains employ sophisticated forecasting and planning systems based on the clustering, or grouping, of their stores.

Ross Iaci,
Department of Mathematics,
The College of
William and Mary,
Williamsburg, VA, 23187.
Email riaci@wm.edu.
Ashok K. Singh,
William F. Harrah College
of Hotel Administration,
University of Nevada, Las
Vegas, 89154-6021.

[1] Ross Iaci, Department of Mathematics, The College of William and Mary, Williamsburg, VA, 23187. Email riaci@wm.edu. Ashok K. Singh, William F. Harrah of Hotel Administration, University of Nevada, Las Vegas, 89154-6021.

Similarly, large casinos with customer loyalty programs also use customer transaction data for the segmentation of customers in order to improve marketing strategies and increase membership. The groupings of stores, or customer segments, are quite often formed by simplistic methods, such as clustering stores, or customers, in a geographic location by zip code or ranked sales. Typically, these approaches ignore a large amount of the data collected by the business. A better alternative is to use statistical cluster analysis, a data mining tool that uses the correlation between different variables in a database to form store, or customer, clusters. Forecasting, or planning systems, that use this added information obtained from statistical clustering will lead to an increased potential for growth.

The groupings of stores, or customer segments, are quite often formed by simplistic methods. Typically, these approaches ignore a large amount of the data collected by the business. A better alternative is to use statistical cluster analysis.

Customer transaction data is multivariate in nature, with the columns, or variables, that might for example represent the products in the store purchased by a customer. The size of the dataset to be segmented is typically very large, often hundreds of thousands of observations made on hundreds of variables and regularly, the dataset is very sparse, sometimes with over 90% zero entries. With sparse datasets of large dimensions, performing a cluster analysis and obtaining meaningful and interpretable results may not be feasible. In addition, latent variable methods for clustering, such as Principal Component Analysis (PCA) or Factor Analysis, may not work for such datasets as the amount of data reduction is often minimal. In order to use traditional methods for cluster analysis our method first rearranges the full dataset so that there is a less sparse upper left principal matrix i.e., the upper left part of the data matrix does not have a large number of zero entries. To this end, we introduce a sparsity index based on binary entropy to extract a significantly dimensioned reduced, and less sparse, data matrix more suitable for using an appropriate and established clustering technique. Binary entropy is a measure of the variability, or randomness, in a sequence of 0's and 1's. Moreover, since it is to be expected that the column dimensions will still be large, it is likely that it will be necessary to use latent variable methods to characterize, and visualize, the key differences between groups, which are now more likely to provide reasonable results. This procedure can then be iterated by successively removing the extracted row observations in each principal matrix from the original dataset and repeating the method on this new dataset.

Our method is motivated by the need to cluster casino patrons to find groups of players that are similar, or dissimilar, in the type and frequency of the machines that they play. That is, our goal is to perform a cluster analysis on two extremely sparse casino player tracking datasets, where the rows correspond to customers and the columns to slot machines. Specifically, the first dataset consists of $n \cong 200,000$ players $\times$ $p \cong 500$ slot machines, and the second approximately $n \cong 70,000 \times p \cong 50$, where the observations are the percentage of coin-in per player for each machine. Since there are nearly $p \cong 500$ and 50 machines, or columns, the datasets naturally consist mostly of zero entries; there are 96.6% zero entries for the first dataset analyzed in Section 4 and 81.25% for the second dataset studied in Section 5.

Of note, since the datasets are the propriety of the casinos they have been altered to maintain the structure, which is of importance to this article, yet hide the interpretable business information, which is imperative to the casino. This is accomplished by simply randomizing the order of the column labels, *Our method is motivated by the need to cluster casino patrons to find groups of players that are similar, or dissimilar, in the type and frequency of the machines that they play.*

withholding the specific type of slot machine, and randomly interchanging the customer row labels.

There has been a significant amount of research on how to cluster large categorical datasets. In a recent article on this topic Yan, Chen, Liu & Yi (2010) proposed the SCALE technique, which provides a measure of the quality of the clustering results and importantly, does not require the manual setting of the number of clusters. An example of the type of categorical transactional datasets considered is $t_1 = \{milk, bread, beer\}$ and $t_2 = \{milk, bread\}$, which are a three-item and two-item transaction, respectively. Yan et al. (2010) note that, although generic categorical clustering algorithms (Andritsos, Tsaparas, Miller, & Sevcik, 2004; Chen & Liu, 2005; Barbara, Li, & Cuoto, 2002; Gibson, Kleinberg, & Raghavan, 1998; Ganti, Gehrke, & Ramakrishnan, 1999; Guha, Rastogi, & Shim, 1999; Huang, 1998; Li, Ma, & Ogihara, 2004) can be applied by transforming the data into a Boolean dataset, the two key features of high dimensionality and large volume make the existing algorithms inefficient to process the transformed data. Note that our focus here is not on large categorical datasets that we can transform to Boolean data, but rather on quantitative data of very large dimension; the dimensions of the casino datasets used in our examples are significantly larger than those used in Yan et al. (2010). Most importantly, unlike these methods, we do not develop a new clustering procedure to segment the original dataset directly, rather we extract dimension-reduced datasets from the original data that can be mined using the numerous techniques for identifying clusters that have been developed in the literature.

This article is organized as follows: The method developed to find a subset of sparse datasets more suitable for clustering is described in Section 2; the source of the two casino player tracking datasets is discussed in Section 3; analysis of the two casino datasets are presented in Sections 4 and 5, respectively; a general discussion follows in Section 6; and finally, a brief appendix explaining certain steps of the clustering method are given in Appendix 7.

2. Methodology

In this section we detail our method to subset sparse datasets of high dimension. In Section 1 we describe a simple resorting of the original dataset. In Section 2 an algorithm based on information theory is described for finding a non-sparse, or less sparse, subset of the sorted dataset more suitable for performing a cluster analysis.

2.1 Sorting algorithm

Let $\boldsymbol{D}_{n \times p}$ be any data matrix, with entries denoted d_{ij}, and let $\boldsymbol{D}^{(1)}$ be the corresponding indicator matrix of $\mathbf{1}$'s and $\mathbf{0}$'s for each nonzero and zero entry of $\boldsymbol{D}$, respectively. That is, each entry $d_{ij}^{(1)}$ of $\boldsymbol{D}^{(1)}$, is assigned a value of $\mathbf{1}$ if the corresponding entry $d_{ij} > 0$, and $\mathbf{0}$ otherwise, for $i \in \{1, \dots, n\}, j \in \{1, \dots, p\}$. Also, assume that n and p are both large and that most entries of $\boldsymbol{D}$ are zero, i.e., assume that $\boldsymbol{D}$ is a sparse matrix. First, consider the rows of $\boldsymbol{D}^{(1)}$ that have the most nonzero entries and sort the rows of the matrix such that these particular rows are listed first. Next, sort the columns in an analogous manner by sorting the rows of the transpose of the previously row sorted matrix. The resulting sorted indicator matrix, denoted $\boldsymbol{D}^{(1)*}$, has a least sparse principal matrix in an upper left quadrant of dimension $k_1 \times k_2$. Finally, this matrix is used to index the original sorted data matrix, denoted $\boldsymbol{D}^{(*)}$. A method for determining the row and column dimensions (k_1, k_2) of this matrix is given next Section 2. A simple illustrative example of the sort algorithm is given in Section 1 of the Appendix.

Note that in the analysis of the casino player tracking datasets in Sections 4 and 5 we do not take into account the actual value of the entries of the data matrix $\boldsymbol{D}$. We could do so easily by placing some sort of rank on the each data value. The most basic would be to assign each entry its actual ranked value and then proceed as discussed above; this is demonstrated in Section 7.1 of the Appendix. However, for large sparse datasets this may not be optimal in that a sparse row may contain only a few large values, which would cause this sparse row to be collected among the very first; an undesirable result. This would also likely be true for summing over the actual observed data values. Thus, the resulting matrix would not have the most dense upper left quadrant, as is the goal here. Note that after iterating our method these rows could eventually be collected with similarly typed observations. In the analysis of the casino player tracking datasets, each entry is given a weight of 1 as initially described. This seems appropriate for the datasets considered in this work, since the observations are the percentage of coin in per player on each machine; see Section 3 for a description of the datasets.

Note that our focus here is on quantitative data of very large dimension; the dimensions of the casino datasets used in our examples are significantly larger than those used in Yan et al. (2010).

2.2 Sparsity index

To reduce the dimension of the data matrix both in the number of rows and columns, that is, to find a suitably non sparse dataset more appropriate for applying established clustering methods, we develop a procedure based on information theory. Specifically, we base our method on the well known Shannon binary entropy measure for a Bernoulli random variable X with success probability p,

$$\begin{aligned} \mathrm{H}(X) &= -\sum_{i=0}^{1} (P(X=i))\log_2 P(X=i) \\ &= -[(1-p)\log_2(1-p) \\ &\quad + p\log_2(p)], \quad p \in [0,1], \end{aligned} \tag{1}$$

where $\log_2$ is the base 2 logarithmic function and $\log_2(0) \doteq 0$. $\mathrm{H}(X)$ is a nonnegative function that is maximum, with a value $\mathrm{H}(X) = 1$, when $p = 0.5$, and minimum, with $\mathrm{H}(X) = 0$, when $p = 0$ or 1. $\mathrm{H}(X)$ can essentially be thought of as the ability to predict the event of a 1 or 0. For example, suppose that a fair coin is tossed and a head is denoted as a 1 and a tail as 0, then when $p = 0.5$, the ability to predict the outcome on each toss is minimal and so the entropy is maximum. However, if $p = 1$, then a 1 is certain and the entropy is 0. For more details see, for example, Hyvärinen, Karhunen, & Oja (2001), or Stone (2004). For our purposes we treat any sequence $\{x_n\}$ of 0 and 1's of length n as a vector of observed realizations of a bernoulli random variable with estimated success probability $\hat{p} = \sum_{i=1}^{n} x_i \,/$

n; obviously this is not true for the observations in the datasets, but is not relevant for our method. We then define the binary entropy measure of any vector, $\boldsymbol{a}$ say, consisting only of 1's or 0's, as $\mathrm{H}(\boldsymbol{a}) = -[\,(1-\hat{p})\log_2(1-\hat{p}) + \hat{p}\log_2(\hat{p})]$.

Extending this idea to matrices, consider any matrix $\boldsymbol{A}_{n\times p}$, where each entry $a_{ij} = 0$ or 1, and denote any $k_1 \times k_2$ principal matrix as $\widetilde{\boldsymbol{A}}_{k_1\times k_1}$, $k_1 \in \{1,\dots,n\}$, $k_2 \in \{1,\dots,p$. Next, write the entries of $\widetilde{\boldsymbol{A}}$ as a vector and calculate the binary entropy measure of $\widetilde{\boldsymbol{A}}$ as

$$\mathrm{H}(\widetilde{\boldsymbol{A}}) = -[\,(1-\hat{p})\log_2(1-\hat{p}) + \hat{p}\log_2(\hat{p})],\ \text{where}\ \hat{p} = \sum_{j=1}^{k_2}\sum_{i=1}^{k_1}\frac{a_{ij}}{k_1k_2}. \tag{2}$$

Note that, if $\widetilde{\boldsymbol{A}} = \boldsymbol{0}_{k_1\times k_2} = \boldsymbol{1}_{k_1\times k_2}$, then $\mathrm{H}(\widetilde{\boldsymbol{A}}) = 0$, and thus, values of $\mathrm{H}(\widetilde{\boldsymbol{A}})$ near 0 implies that $\widetilde{\boldsymbol{A}}$ is predominately composed of either 1's or 0's, but not both. Conversely, values of $\mathrm{H}(\widetilde{\boldsymbol{A}})$ near 1 imply that $\widetilde{\boldsymbol{A}}$ has nearly an equal number of 1's and 0's. Therefore, in general the index $\mathrm{H}(\widetilde{\boldsymbol{A}})$ can be used as a measure of the sparsity of the matrix $\widetilde{\boldsymbol{A}}$.

Next, we consider all possible divisions of the sorted indicator matrix $\boldsymbol{D}^{(1)*}_{n\times p}$ into four primary matrices, denoted $\widetilde{\boldsymbol{D}}_j, j \in \{1,\dots,4\}$, and calculate the matrix entropy values $\mathrm{H}(\widetilde{\boldsymbol{D}})$ for each. The matrices containing all possible entropy measures, corresponding to every possible division, are denoted $\boldsymbol{Q}_j$. Given the row and column dimensions, k_1 and k_2, this can be diagrammed as follows,

$$\mathbf{D}^{(1)*} = \begin{bmatrix} \widetilde{\boldsymbol{D}}^{(1)*}_{(1:k_1,1:k_2)} = \widetilde{\boldsymbol{D}}_1 & \widetilde{\boldsymbol{D}}^{(1)*}_{(1:k_1,k_2:p)} = \widetilde{\boldsymbol{D}}_2 \\ \widetilde{\boldsymbol{D}}^{(1)*}_{(k_1:n,1:k_2)} = \widetilde{\boldsymbol{D}}_4 & \widetilde{\boldsymbol{D}}^{(1)*}_{(k_1:n,k_2:p)} = \widetilde{\boldsymbol{D}}_3 \end{bmatrix}_{n\times p} \rightarrow \begin{bmatrix} \boldsymbol{Q}_1 = \mathrm{H}(\widetilde{\boldsymbol{D}}_1) & \boldsymbol{Q}_2 = \mathrm{H}(\widetilde{\boldsymbol{D}}_2) \\ \boldsymbol{Q}_4 = \mathrm{H}(\widetilde{\boldsymbol{D}}_4) & \boldsymbol{Q}_3 = \mathrm{H}(\widetilde{\boldsymbol{D}}_3) \end{bmatrix}_{n\times p.}$$

Here the notation $\widetilde{\boldsymbol{D}}^{(1)*}_{(1:k_1,1:k_2)}$ refers to the matrix determined by the first k_1 rows and k_2 columns, $\widetilde{\boldsymbol{D}}^{(1)*}_{(1:k_1,k_2:p)}$ specifies the matrix determined by the first k_1 rows and k_2 to p columns, etc.. Thus, $\boldsymbol{Q}_1(k_1,k_2)$, $\boldsymbol{Q}_2(k_1,k_2)$, $\boldsymbol{Q}_3(k_1,k_2)$, and $\boldsymbol{Q}_4(k_1,k_2)$ are the matrices with each entry corresponding to the sparsity measure of a corresponding principal matrix determined by k_1 and k_2, $k_1 \in \{1,\dots,n\}$, $k_2 \in \{1,\dots,p\}$. The notation $\boldsymbol{Q}_j(k_1,k_2)$ is used here to emphasize that the size of the matrices depend on the values of k_1 and k_2. Note that, $\boldsymbol{Q}_1(n,p) = \boldsymbol{Q}_2(n,1) = \boldsymbol{Q}_3(1,1) = \boldsymbol{Q}_4(1,p) = \mathrm{H}(\boldsymbol{D}^{(1)*})$, or the total sparsity measure of the matrix $\boldsymbol{D}^{(1)*}$; the shaded region in the far left panel of Figure 1 represents the matrix that would have corresponding entropy measure $\boldsymbol{Q}_2(5,1)$ (or $\boldsymbol{Q}_1(5,p)$). To determine a non sparse upper left matrix for analysis, indexed by $\widetilde{\boldsymbol{D}}_1$, we find estimates of the row and column cutoffs k_1 and k_2 in two steps. In the first step the estimated number of rows $\hat{k}_1$ is taken to be $\hat{k}_1 = \text{argmin}_{k_1} |\boldsymbol{Q}_2(k_1, 1) - c_1|$, where $0 \leq c_1 \leq 1$. This is equivalent to defining $\hat{k}_1$ to be the value of k_1 such that $\boldsymbol{Q}_2(k_1, 1) = c_1$. Note that it is not necessary to fix $k_2 = 1$, but we have done so because the value of c_1 can be chosen correspondingly for any fixed k_2. For example, consider the left panel of Figure 1, the shaded region of a sorted matrix shows a possible matrix for which a value of $\boldsymbol{Q}_2(k_1, 1)$ can be calculated using (2). Further, ignoring the ellipses extending the matrix, if c_1 is near 1, then $|\boldsymbol{Q}_2(k_1, 1) - c_1|$ would be close to 0 for this value of k_1, since there is roughly an equal number of 0's and 1's. The rationale being that the sorted $\boldsymbol{D}^{(1)*}$ matrix should consist of mostly 1's in an upper most left quadrant and transition to mostly 0's in the other three quadrants. In this case, if k_1 is increased, more 0's are added, and the entropy of the corresponding new shaded matrix will approach 0, but not 1. Alternatively, if $c_1 = 0$ then the resulting estimate of the number of rows would approach $\hat{k}_1 = n$. This is because there is only one nonzero entry in the last column, thus, as more zero row entries are added $\boldsymbol{Q}_2(k_1, p) \to 0$, or $|\boldsymbol{Q}_2(k_1, p) - c_1| \to 0$, (up to a point), which implies that $\hat{k}_1$ would get closer to n. So in effect, as the value of c_1 is increased the number of rows determined by k_1 decreases.

Next, with the number of rows fixed, we estimate the number of columns as $\hat{k}_2 = \text{argmin}_{k_2} |\boldsymbol{Q}_1(\hat{k}_1, k_2) - c_2|$, where $0 \leq c_2 \leq 1$. Continuing the example, once more ignoring the ellipses, the right panel of Figure 1 shows a possible shaded matrix where $|\boldsymbol{Q}_1(\hat{k}_1, k_2) - c_2|$ would be near 0 for a value of c_2 near 0. Again, as c_2 becomes closer to 1 then more columns would be added, with $\hat{k}_2$ near p if $c_2 = 1$.

$$\left[\begin{array}{ccccccccc}
1 & 1 & 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & \cdots & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & \cdots & 0 & 0 \\
\hline
0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0
\end{array}\right]$$

$$\left[\begin{array}{ccccc|cccc}
1 & 1 & 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 0 & \cdots & 0 & 1 \\
1 & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\
1 & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\
1 & 1 & 1 & 0 & 1 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0
\end{array}\right]$$

Figure 1. Explanatory example. Left panel: $\boldsymbol{Q_2(k_1, 1)}$ and Right panel: $\boldsymbol{Q_1(k_1, k_2)}$ Section 2.

Succinctly, for a sorted indicator matrix $\boldsymbol{D}_{n\times p}^{(1)*}$ we introduce the sparsity index,

$$\mathcal{S}(\boldsymbol{D}^{(1)*}; k_1, k_2) \\ = |\boldsymbol{Q}_1|[\boldsymbol{Q}_2(k_1, 1) - c_1], k_2| \\ - c_2|. \tag{3}$$

In practice, for $0 \leq c_1, c_2 \leq 1$, the estimated row and column dimensions $\hat{k}_1$ and $\hat{k}_2$ are the solutions of

$$(\hat{k}_1, \hat{k}_2) = \underset{k_1,k_2}{\operatorname{argmin}} \mathcal{S}\,(\boldsymbol{D}^{(1)*}; k_1, k_2 \\ = \underset{k_2}{\operatorname{argmin}} \,|\, \boldsymbol{Q}_1([\underset{k_1}{\operatorname{argmin}} \,|\, \boldsymbol{Q}_2(k_1, 1) \\ - c_1|], k_2) - c_2|, \tag{4}$$

where $\hat{k}_1$ is determined first, substituted for k_1, and then the outer minimization determines $\hat{k}_2$. Note that successive minimization is not necessary, but worked well for the analyzed datasets and also reduces computation time, since with k_1 fixed k_2 is determined from a smaller dimensioned indicator matrix. This reduced dimensioned dataset is then clustered. Finally, these observations are removed from the original dataset and the steps repeated until the entire dataset has been clustered.

The algorithm is now shown below pictorially for one iteration for an arbitrary sorted indicator matrix $\boldsymbol{D}^{(1)*}$. Moreover, this is purely illustrative in that no specific values of c_1 and c_2, and hence, $\hat{k}_1$ and $\hat{k}_2$, are specified. In the first step the estimated number of rows $\hat{k}_1$ is the value of k_1 such that $|\boldsymbol{Q}_2(k_1, 1) - c_1| \approx 0$, or $\boldsymbol{Q}_2(k_1, 1) \approx c_1$, for example,

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & \cdots & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \end{bmatrix}_{n\times p}$$

$$\hat{k}_1 \rightarrow \begin{bmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & \cdots & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & \cdots & 0 & 0 \end{pmatrix}_{\hat{k}_1\times p} \\ \begin{matrix} 0 & 0 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \end{matrix} \end{bmatrix}_{n\times p.}$$

Next, the estimated number of rows $\hat{k}_2$ subsets $\boldsymbol{D}^{(1)*}$, and the number of columns is found in similar fashion as above in that $\hat{k}_2$ is the value of k_2 such that $\boldsymbol{Q}_1(\hat{k}_1, k_2) \approx c_2$; for example,

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & \cdots & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & \cdots & 0 & 0 \end{bmatrix}_{\hat{k}_1 \times p} \quad \hat{k}_2$$

$$\rightarrow \begin{bmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}_{\hat{k}_1 \times \hat{k}_2} & \begin{matrix} 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 1 \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \\ 0 & \cdots & 0 & 0 \end{matrix} \end{bmatrix}_{\hat{k}_1 \times p.}$$

Finally, the actual data indexed by the parenthesis in the far right bracketed matrix of the original sorted data matrix $\boldsymbol{D}^{(*)}$ is used for clustering; again we denote this matrix as $\widetilde{\boldsymbol{D}}$. For example, assuming the matrix indexes the same sorted data values in the example used to describe the sorting algorithm in Appendix Section 7.1, the data matrix for clustering is

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}_{\hat{k}_1 \times \hat{k}_2} \rightarrow$$

$$\widetilde{\boldsymbol{D}} = \begin{bmatrix} 200.03 & 111.12 & 160.21 & 21.56 & 105.65 \\ 400.56 & 140.65 & 191.25 & 241.28 & 90.65 \\ 20.17 & 12.65 & 170.26 & 222.87 & 0 \\ 333.65 & 137 & 180.64 & 124.34 & 0 \\ 50.54 & 155.26 & 205.64 & 0 & 100.26 \end{bmatrix}.$$

Naturally, other sparsity indexes involving the $\boldsymbol{Q}_j$ matrices can be constructed to find appropriate row and column cutoff values, which illustrates the usefulness of this technique. As previously discussed, using the sparsity index (3), values of c_1 and c_2 closer to 1 result in the smallest number of rows and largest number of columns being selected. For the casino datasets there are an extremely large amount of rows, or customers, so c_1 closer to 1 results in a larger reduction in the number of customers. However, the columns have less dimension and are composed of the slot machines, which are of strong interest, so including more columns seems reasonable. These values of c_1 and c_2 generate datasets that produce distinctive clusters for the datasets studied below. In practice, and dependent on the particular dataset being analyzed, different values can be used to create various sizes of dimension reduced datasets.

Note that, we are assuming the original data matrices are of high dimension, for example the size of the first casino dataset analyzed in Section 4 is on the order of $n \cong 200,000 \times p \cong 500$. In addition to sorting such a large dataset, we also need to calculate the $\boldsymbol{Q}_j$ matrices for the sparsity index, which is computationally intensive assuming the given high dimensional scenario. To this end, efficient codes have been developed in MATLAB for all calculations.

3. Data source

A Las Vegas hotel casino corporation provided secondary data from two of their repeater market properties. As is the case with almost all Las Vegas repeater market properties, this casino's primary revenue is generated by slot machines. In order to protect the confidentiality of the data source, the name of the property and the exact dimensionality of the datasets are not reported.

Both of the datasets consist of coin-in from p different slot machines for n carded customers, i.e., from players who use their reward cards. The datasets consist of n rows of carded customers, and p columns of slot machines. Both datasets were first normalized by converting the coin-in values to the percentage of coin-in per player. The dimension of the first dataset analyzed in Section 4 is on the order of $n \cong 200,000 \times p \cong 500$. The second dataset was smaller, with dimensions around $n \cong 70,000 \times p \cong 50$.

4. Casino player dataset 1

Compounding the large dimensions in this dataset is that most of the n carded players play at most only a few slot machines, thus, the percentage of 0 entries is 96.6%. The method detailed in Section 2 is used to find non-sparse datasets more appropriate for clustering in Section 4.1. In Section 4.2 a cluster analysis is performed and the final results discussed in Section 4.3.

4.1 Dimension reduced datasets

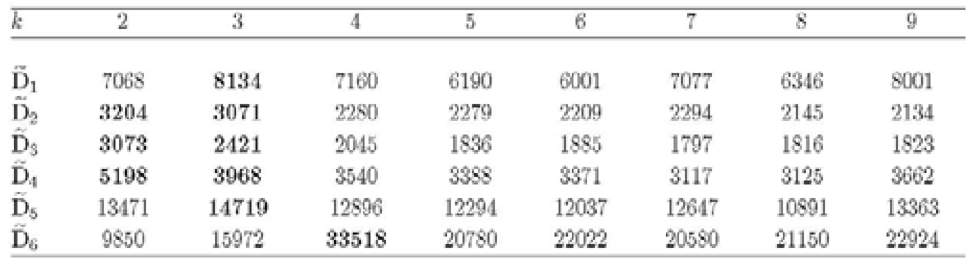
To reduce the dimension of the dataset to find interpretable and informative clusters of casino players and the machines they play, the number of rows and columns $(\hat{k}_1, \hat{k}_2)$ of the resorted dataset is obtained using the method described in Section 2.2 for various values of the pair (c_1, c_2). Specifically, the following values are used: $(c_1, c_2) = [(1, 0.97); (0.99, 0.96); (0.98, 0.95)]$. In particular, for this dataset the non-sparsity, or density of nonzero values, was dispersed across all the $p \cong 500$ columns. Thus, the transition to a large amount of zero entries in the upper right quadrant of the sorted data matrix is more gradual. This in turn also resulted in a larger number of rows being included that showed this trend across increasing columns. So, choosing a value of c_1 very close to 1 seemed the most probable value to produce a significantly row reduced dataset likely to provide meaningful results when clustered. Now, choosing c_2 close to 1 has the effect of including a large amount of columns, or machines, which is naturally deemed important. That is, we want to exclude only the machines that had the most minimal amount of play, and thus, likely to have a negative effect on the clustering results.

These particular values of the parameters (c_1, c_2) yielded the following estimated row and column values: $(\hat{k}_1, \hat{k}_2) = [(403, 346); (604, 316); (812, 268)]$. That is, these values create dimension reduced data matrices, $\widetilde{\boldsymbol{D}}$, of size $(n = 403, p = 346)$, $(n = 604, p = 316)$, and $(n = 812, p = 268)$. In addition, the values $(\hat{k}_1, \hat{k}_2) = [(2000, 350); (10000, 350); (20000, 350)]$ were also included in the analysis for comparison. These row and column values are chosen to exceed the maximum of the estimated values in order to compare the results in analyzing the reduced dimensioned datasets to ones remotely comparable to the full dataset. Note that the last dataset is roughly one quarter the size of the original data matrix. In subsequent discussion of the analysis, each of these datasets are denoted $\widetilde{\boldsymbol{D}}_j, j \in \{1, \dots, 6\}$ in order of introduction, i.e. $\widetilde{\boldsymbol{D}}_1$ is of dimension 403×346, $\widetilde{\boldsymbol{D}}_2$ of dimension 604×316, and so on.

4.2 Cluster analysis

For each of the six dimension reduced datasets of Section 4.1 a k-means cluster analysis, $k = 2, \dots, 9$, is performed. For each dataset the standard deviations of all of the p variables were similar, and thus, the cluster analysis was performed on the raw data values. Specifically, the dissimilarities between each group are calculated using the centroid method. The squared Euclidean distance between the centroids of each group, or the L_1 distance, often termed the “city block" distance, is used in the k-means procedure implemented in MATLAB. The centroid method was chosen since, like the “nearest neighbor" and “group average" methods, this leads to “spherical" clusters exhibiting high internal affinity, Krzanowski (2001). This method is also suggested by MacQueen (1967), with a full description found in the classic multivariate text by Johnson & Wichern (2002).

Letting $\boldsymbol{W}$ represent the matrix of within-cluster variation, and $\boldsymbol{B}$ the matrix of between-cluster variation, a reasonable method to decide between the most appropriate number of clusters is to choose the one such that the between-cluster variability relative to the within-cluster variability is maximum. A relevant measure of this ratio of variability is the trace of $\mathcal{K} = [1/(n - k)\boldsymbol{W}]^{-1}[1/(k-1)\boldsymbol{B}]$. Table 1 gives the values of this sample trace for each of the six datasets and each of the $k = 2, \dots, 9$ clusters.



k	2	3	4	5	6	7	8	9
$\widetilde{\mathrm{D}}_1$	7068	**8134**	7160	6190	6001	7077	6346	8001
$\widetilde{\mathrm{D}}_2$	**3204**	**3071**	2280	2279	2209	2294	2145	2134
$\widetilde{\mathrm{D}}_3$	**3073**	**2421**	2045	1836	1885	1797	1816	1823
$\widetilde{\mathrm{D}}_4$	**5198**	**3968**	3540	3388	3371	3117	3125	3662
$\widetilde{\mathrm{D}}_5$	13471	**14719**	12896	12294	12037	12647	10891	13363
$\widetilde{\mathrm{D}}_6$	9850	15972	**33518**	20780	22022	20580	21150	22924

Table 1. Value of the trace ($\mathcal{K}$) - Example 4

For $\widetilde{\boldsymbol{D}}_1$ and $\widetilde{\boldsymbol{D}}_5$ the trace is maximum when $k = 3$, and excluding $\widetilde{\boldsymbol{D}}_6$, second largest for the remaining datasets. Therefore, we conclude for the first 5 datasets that either a 2 or 3 cluster analysis is the most optimal, and a 4 clusters analysis for the last dataset may be best.

However, for any dataset $\widetilde{\boldsymbol{D}}_j$ there are $p > 3$ variables, and thus, no direct way to visualize the differences between the groups. In order to view the clusters, and draw inferences visually, we must project them into a subspace of dimension less than 3. That is, we need to try and approximate the grouped p-dimensional points in a lower dimension, if possible. To do this, since we have more reasonably dimensioned datasets, we perform a Canonical Variate

Analysis (CVA), obtained from a Multivariate Analysis of Variance (MANOVA), on the scatter matrix $\boldsymbol{W}^{-1}\boldsymbol{B}$. The 1st eigen-vector $\boldsymbol{e}_1$ of $\boldsymbol{W}^{-1}\boldsymbol{B}$ projects the data into a one-dimensional subspace such that the between-group variability is maximum relative to within-group variability. More simply, the data is projected into a one-dimensional subspace that maximizes the separation between groups. The projections, or linear combinations, of the p-dimensional data points are termed the canonical variates hereafter. The associated eigen-value, λ_1, gives the measure of the ratio of the between-group and within group variance. Analogous to PCA, the eigenvectors $\boldsymbol{e}_2, \boldsymbol{e}_3, \dots \boldsymbol{e}_s,\ s \leq min(p,k)$ project the data into subspaces such that the between-group variability is the $2^{\text{nd}}, 3^{\text{rd}}, \dots, s^{\text{th}}$ greatest relative to within-group variability. Mathematical details are summarized in Section 2 of the Appendix; comprehensive details can be found in Krzanowski (2001).

4.3 Results

Scatter plots of the 1^{st} versus the 2^{nd} canonical variates for all datasets indicated that tight and distinctive groupings occur when $k = 3$ or 4; scatter plots for three clusters are given in Figure 2. Based on this observation, and that the maximum values of the traces in Section 4.2 usually occur at or near $k = 3$, the results for three clusters are discussed hereafter.

The upper most left panel of Figure 2 indicates that the most separated, or strongest, clustering occurs for $\widetilde{\boldsymbol{D}}_1$. The plots in the next two upper panels corresponding to $\widetilde{\boldsymbol{D}}_2$ and $\widetilde{\boldsymbol{D}}_3$ show that the clusters become less separated as the number of rows are increased and the number of columns decreased. In fact, as the number of rows increases the projected clusters become less discernible from each other, which indicates that only a subset of this dataset can be grouped into strong distinct clusters; or that only these clusters of p dimensional points can be approximated in two dimensions. In addition, the separation between the groups for each of the datasets is along both the 1^{st} (y) and 2^{nd} (x) canonical axes.

Figure 3 gives the histogram plots of the coefficients of the eigenvectors, or loadings, corresponding to the 1^{st} canonical variates for all datasets. Most coefficient values are near zero with the exception of a few. Thus, especially for the first two datasets, $\widetilde{\boldsymbol{D}}_1$ and $\widetilde{\boldsymbol{D}}_2$, only a few select machines explain the greatest between group variation. For example, in the histogram plots of the loadings for dataset $\widetilde{\boldsymbol{D}}_1$ in the top left panel of Figure 3 the six largest values greater than 100 correspond to machines(loading); $p_{111(-107.36)}$, $p_{112(-395.66)}$, $p_{124(-100.77)}$, $p_{77(-137.65)}$, and $p_{119(114.02)}$, $p_{120(493.31)}$, $p_{74(355.24)}$, and $p_{116(447.0655)}$. Thus, the 1^{st} canonical variate is a contrast between the slot machines $p_{111}, p_{112}, p_{124}, p_{77}$ and $p_{119}, p_{120}, p_{74}, p_{116}$, with the strongest contrast between p_{112} and p_{120}, p_{74}, p_{116}. For example, the 1^{st} canonical variate might distinguish players of wide area progressive machines from non-players, that is, wide area progressive players and non-players are the most distinguishable. The 2^{nd} canonical variate results are: $p_{116(714.4)}$, $p_{124(212.0)}$, $p_{77(568.9)}$, $p_{120(339.2)}$ and $p_{111(-166.9)}$, $p_{112(-187.5)}$, $p_{74(-1146.2)}$, again, analogous to the 1^{st} canonical variate, is largely a contrast between just a few slot machines.

Although the actual machines that are given a certain weight is important to the casino, for the purpose of illustrating our method this is less relevant. This example shows the usefulness of our technique in determining a dataset more suitable for clustering. Performing a CVA is necessary since the dimensions are high, and allows us to make inferences as to what are the characteristics, or types of machines, that separate casino slot players the most. Also, further iterations of our method did not yield strong results and thus have been omitted for brevity.

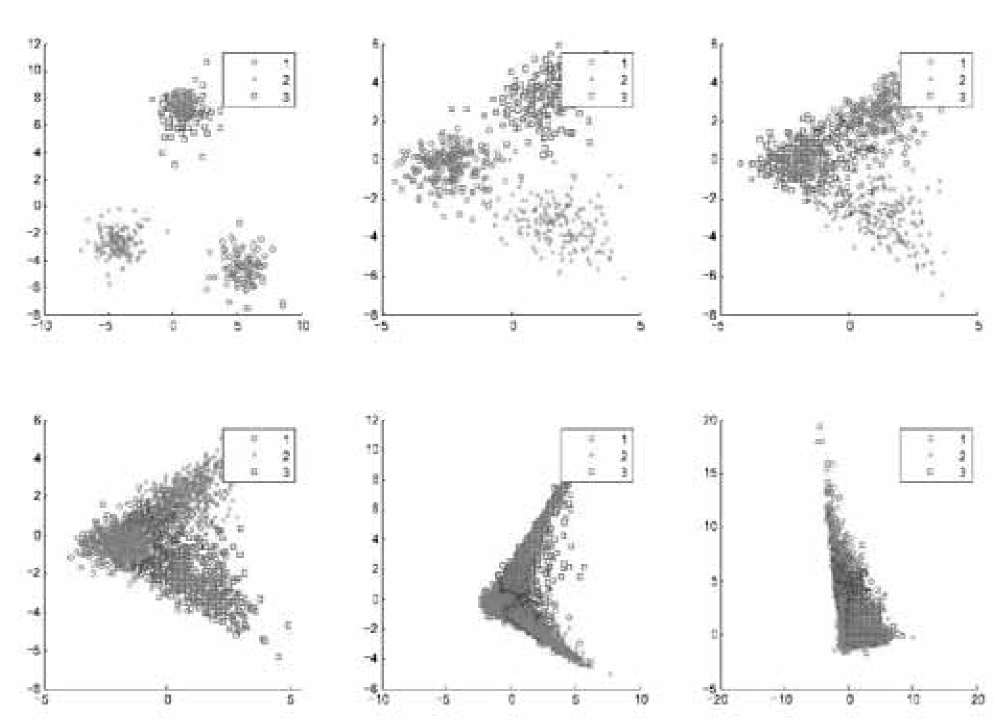


Figure 2. 1st vs. 2nd canonical variate plots for $k = 3$ clusters - Casino data Example 4

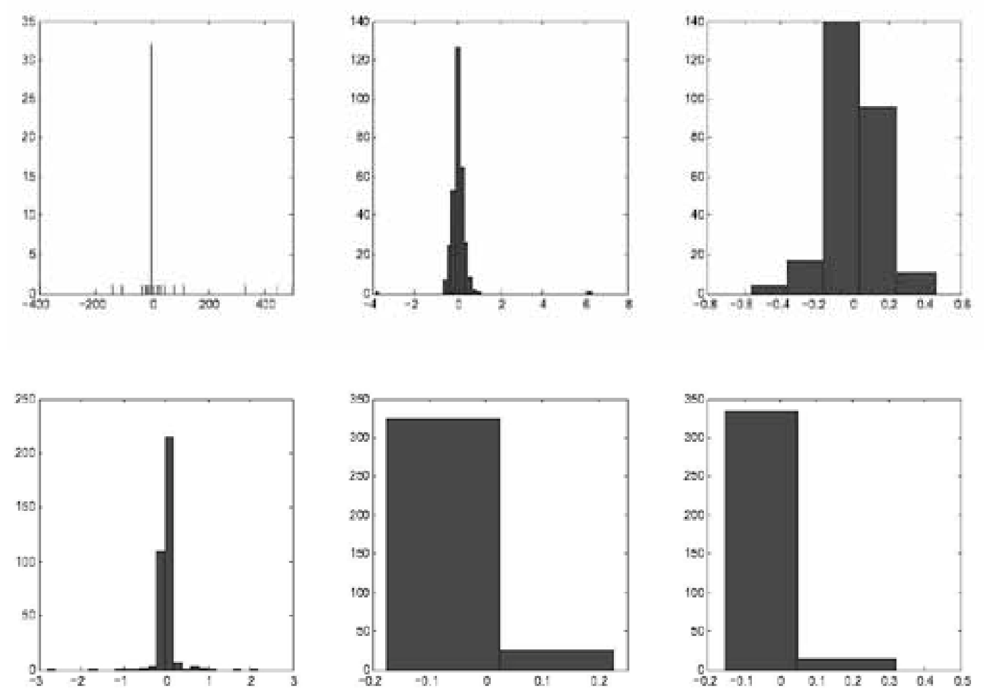


Figure 3. 1st canonical coefficients (loadings) for $k = 3$ clusters - Casino data Example 4

5. Casino Player Dataset 2

In this dataset, 10% of the carded players played only one slot machine, so these players were put in one cluster that we will refer to as "loyal customers" and then removed from the dataset. The steps used to analyze the previous casino player tracking dataset are then repeated on the dataset consisting of the remaining players.

5.1 Dimension reduced dataset

For the same reasons discussed in the previous example the following sequence of values to subset the dataset are used: $(c_1, c_2) = [(1, 0.97); (0.99, 0.96); (0.98, 0.95)]$. These values create dimension reduced datasets, $\widetilde{\boldsymbol{D}}$, of size $(n = 10348, p = 47)$, $(n = 15048, p = 33)$, and $(n = 17466, p = 16)$. In addition, the values $(\hat{k}_1, \hat{k}_2) = [(2000, 58); (5000, 47)]$ were also included in the analysis for comparison. These row and column values are chosen to exceed the maximum number of columns and the minimum number of players in order to contrast the analysis results from the dimension reduced datasets. In subsequent discussion each of the datasets are denoted $\widetilde{\boldsymbol{D}}_j, j \in \{1, \dots, 5\}$ in the order of introduction.

5.2 Cluster analysis

For each of the five datasets, $\widetilde{\boldsymbol{D}}_j$, a k-means cluster analysis, $k = 2, \dots, 6$, is performed. For each dataset the standard deviations of all of the p variables were similar, and thus, the cluster analysis was performed on theraw data values. As before, differences between groups are calculated using the centroid method implementing the “city block" distance. Once more, to aid in deciding the most appropriate numbers of clusters, Table 2 gives the values of the sample trace of $\mathcal{K} = [1/(n-k)\boldsymbol{W}]^{-1}[1/(k-1)\boldsymbol{B}]$ for each of the five datasets and $k = 2, \dots, 6$ clusters. For $\widetilde{\boldsymbol{D}}_2$ the trace is maximum when $k = 3$, while $k = 2$ produces the largest trace for the remaining datasets.

k	2	3	4	5	6
$\widetilde{\mathrm{D}}_1$	**82970**	70440	6541	5346.0	4823.1
$\widetilde{\mathrm{D}}_2$	10327	**12479**	8865	7155.6	6777.0
$\widetilde{\mathrm{D}}_3$	**12499**	11662	11924	9321.7	7583.9
$\widetilde{\mathrm{D}}_4$	**15660**	14320	1514	1302.6	1268.3
$\widetilde{\mathrm{D}}_5$	**28820**	17290	1894	2069.5	2533.1

Table 2. Value of the trace ($\mathcal{K}$) - Example 5

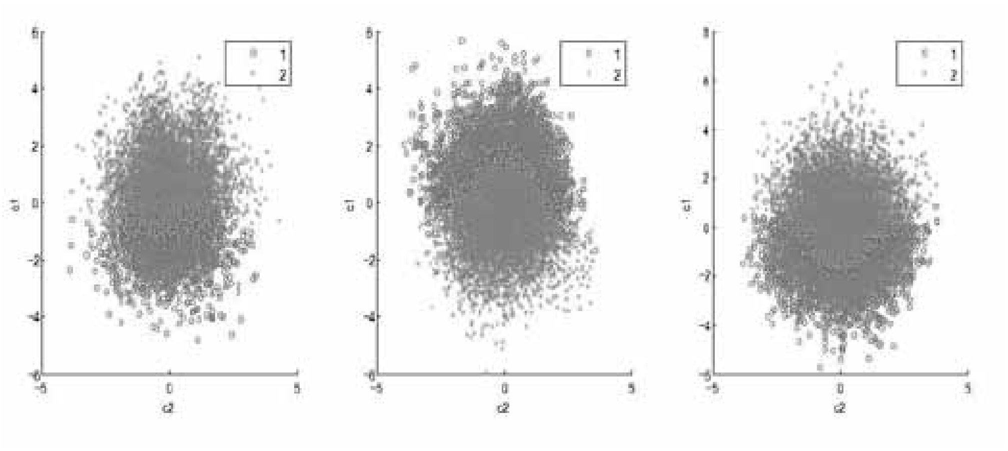
5.3 Results

Again, in order to draw inferences from the k-means cluster analysis we perform a CVA on $\boldsymbol{W}^{-1}\boldsymbol{B}$ obtained from a MANOVA on each dataset. For all datasets plots of the 1st versus the 2nd canonical variates indicated that the tightest and most distinct groupings occur for $k = 2$ and 3 clusters. The scatter plots for two and three clusters are given Figures 4 and 6, respectively. Recalling that the maximum value of the traces in Section 5.2 occur almost exclusively when $k = 2$, the results for two and three clusters are discussed hereafter.

For $k = 2$ histogram plots of the loadings associated with the 1st canonical variates for all datasets are given in Figure 5 and show that two groups are separated along the 1st canonical (y) axis, but not the2nd (x) axis. The most interesting groupings occur for datasets $\widetilde{\boldsymbol{D}}_2$ and $\widetilde{\boldsymbol{D}}_3$, where the loadings of the coefficients contrast half of the $p = 58$ machines. That is, looking at the middle and right upper panels of Figure 5 we see that positive and negative loadings are close in value and nearly evenly dispersed across the slot machines. Thus, the two clusters of casino players for this dataset are most separated by whether they play a machine associated with a negative or positive loading in the 1st coefficient vector. This grouping of positive and negative weights that separate the two groups also seems to apply to the dimension reduced dataset $\widetilde{\boldsymbol{D}}_1$, with the exception that many of the machines are given weights near zero. Note that, $\widetilde{\boldsymbol{D}}_1$ consists of $p = 47$ machines, while $\widetilde{\boldsymbol{D}}_2$ and $\widetilde{\boldsymbol{D}}_3$ are comprised of $p = 33$ and 16 machines, respectively.

For $k = 3$ clusters a similar result to that of the $k = 2$ clusters for $\widetilde{\boldsymbol{D}}_1$ occurs for datasets $\widetilde{\boldsymbol{D}}_1$, $\widetilde{\boldsymbol{D}}_2$ and $\widetilde{\boldsymbol{D}}_3$. The canonical variate plots are given in Figure 6 and the loadings of the 1st canonical variates in Figure 7. The top upper panel of Figure 7 corresponding to the first 3 reduced datasets shows that there are three groups of machines that separate the three clusters the most; machines with a large negative weight, machines with a large positive weight, and those weighted less (close to zero).

To complete the clustering of the this Casino Player Tracking data, the observations in the six dimension reduced datasets are again removed from the original data and the above steps carried out on the resulting dataset; the results did not show any strong groupings and thus, have been omitted for brevity.



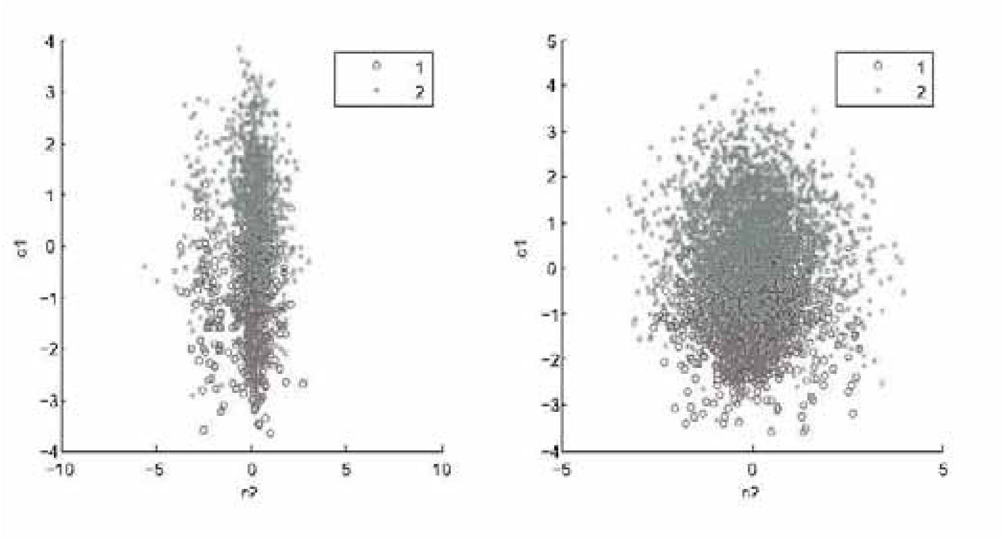


Figure 4. 1st vs. 2nd canonical variate plots for $k = 2$ clusters - Casino data Example 5

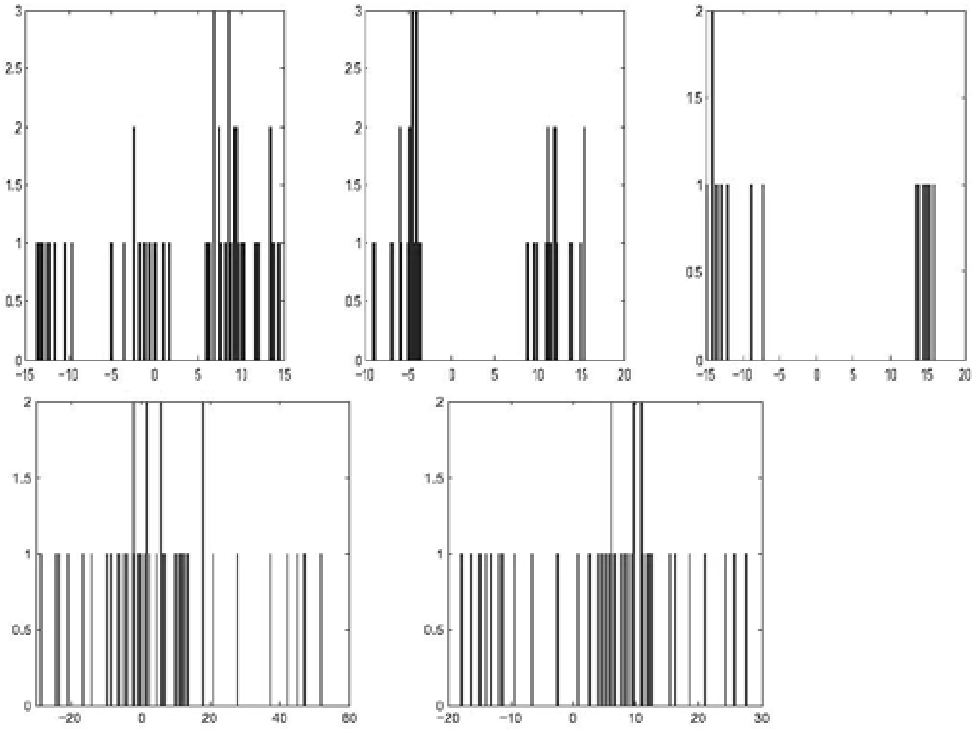
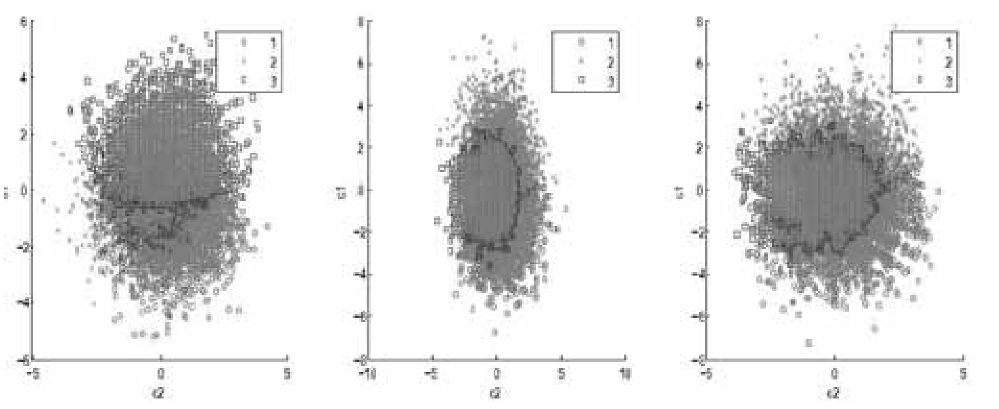


Figure 5. 1^{st} canonical coefficients (loadings) for $k = 2$ clusters - Casino data Example 5



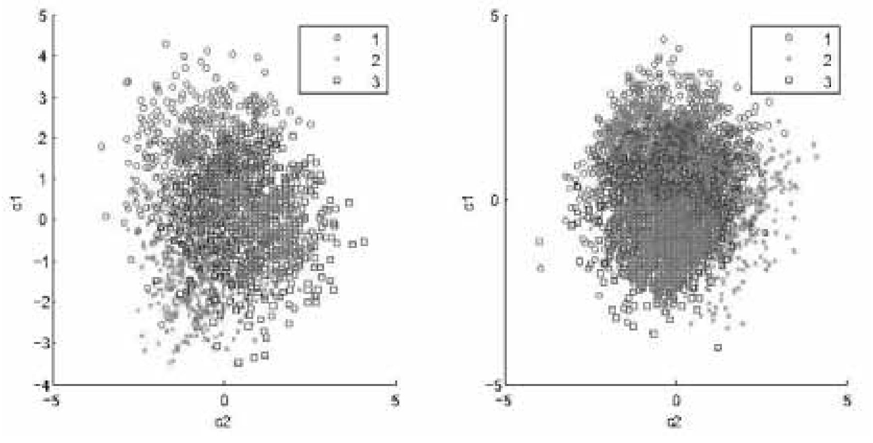
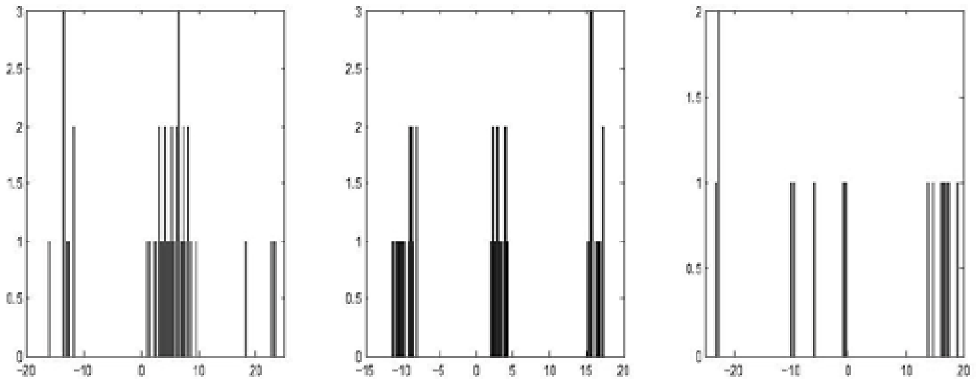


Figure 6. 1st vs. 2nd canonical variate plots for $k = 3$ clusters - Casino data Example 5



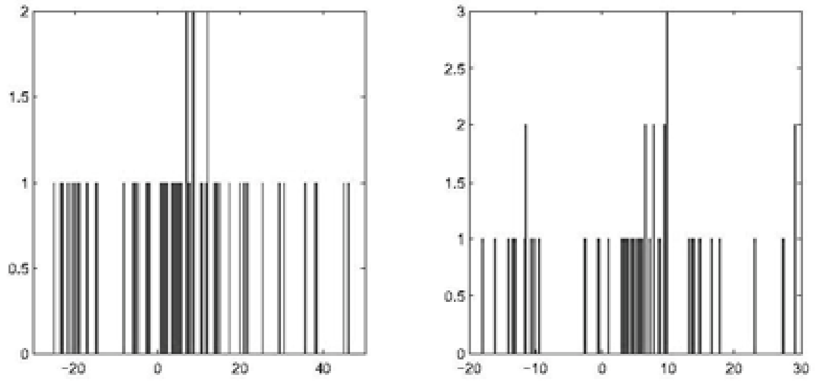


Figure 7. 1st canonical coefficients (loadings) for $k = 3$ clusters - Casino data Example 5

6. Discussion

In this article, we introduce a technique for clustering large and sparse casino player tracking datasets. One common problem in clustering such datasets is that data reduction methods on the full dataset, for example PCA, typically do not result in a noticeable dimension reduction and hence, in interpretable clusters. However, our clustering method for such datasets finds subsets of the original data with a large percentage of non-zero entries which are more suitable for the implementation of standard clustering techniques. That is, our method extracts dimension reduced datasets from the original dataset so that appropriate clustering techniques can be selected from the vast amount already contained in the literature.

Both casino datasets were of substantially different dimensions, but for each, our method was able to extract non-sparse subsets of the data of suitable dimensions to be clustered. To subset both datasets we used values of (c_1, c_2) so that a larger proportion of slot machines (columns) would be selected at each iteration. The initial reduced datasets produced under this parameterization for the first casino transaction data example enabled the classical k-means clustering procedure to recover three strong clusters. A Canonical Variate Analysis was then able to identify a contrast of a few slot machines that maximized the separation between these groups. The second less sparse casino dataset produced somewhat less distinct clusters, but segmented a larger number of customers. In contrast to the first dataset, the CVA analysis determined that a contrast involving most of the machines separated the groups.

Although the main goal was to segment casino player transaction datasets, the procedure is adaptable to sparse datasets of different types with large dimension that need to be segmented using classical clustering techniques that might otherwise be inappropriate. Our sparsity index is also flexible in that it can be parameterized to select more or less columns, or rows, dependent on the problem. Thus, these reasons, and the intuitive nature of our method, makes our procedure for clustering sparse high dimensional datasets attractive to general practitioners and gaming operators. For example, this approach can be used to cluster slot machine games based upon features such as free spin and bonus games, slot denomination, maximum number of lines, scatter and wild symbols, with a view to determine what features are associated with high coin-in values. Most of the above features are binary in nature, and therefore a large dataset of coin-in values of various slot games with a binary column for each of the above features would be a sparse data matrix. The clustering of slot games based on such a dataset would identify the features of popular slot machines.

Thus, these reasons, and the intuitive nature of our method, makes our procedure for clustering sparse high dimensional datasets attractive to general practitioners and gaming operators. For example, this approach can be used to cluster slot machine games based upon features such as free spin and bonus games, slot denomination, maximum number of lines, scatter and wild symbols, with a view to determine what features are associated with high coin-in values.

References

Andritsos, P., Tsaparas, P., Miller, RJ. & Sevcik, KC. (2004). Limbo: scalable clustering of categorical data. *Proceedings of international conference on extending database technology,* 123 146.

Barbara, D., Li, Y. & Couto J. (2002). Coolcat: an entropy-based algorithm for categorical clustering. *Proceedings of ACM conference on information and knowledge management,* 582 589.

Berman, B. (2006). Developing an Effective Customer Loyalty Program. *California Management Review,* 49, 123.

Chen K, & Liu L (2005) The "best k" for entropy-based categorical clustering. *Proceedings of international conference on scientific and statistical database management,* 253 262.

Ganti, V., Gehrke, J., & Ramakrishnan, R. (1999). Cactus: clustering categorical data using summaries. *Proceedings of ACM SIGKDD international conference on knowledge discovery and data mining,* 73 83.

Gibson D., Kleinberg J. , & Raghavan, P. (1998). Clustering categorical data: an approach based on dynamical systems. *Proceedings of the 24th international conference on very large data bases,* 311 322.

Guha S., Rastogi R., & Shim, K. (1999). Rock: a robust clustering algorithm for categorical attributes. *Proceedings of IEEE international conference on data engineering,* pp 512 521.

Huang Z (1998) . Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery* 2(3), 283 304.

Hyvärinen, A., Karhunen, J., & Oja, E. (2001). Independent Component Analysis. New York: John Wiley & Sons, Inc.

Johnson, R.A., & Wichern, D.W. (2002). Applied Multivariate Statistical Analysis. Upper Saddle River, New Jersey: Prentice Hall.

Krzanowski, W. J. (2001). Principles of Multivariate Analysis: A User's Perspective. Oxford Statistical Science Series, No. 23.

Li T, , Ma S. & Ogihara, M. (2004). Entropy-based criterion in categorical clustering. *In: Proceedings of international conference on machine learning,* 68 75.

MacQueen, J.B. (1967). Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability,* 1, Berkley, CA: University of California Press, 281 97.

Stone, J.V. (2004). Independent Component Analysis: A Tutorial Introduction. Cambridge, MA:. A Bradford Book. MIT Press.

Yan, H., Chen, K., Liu, L. & Yi, Z. (2010). SCALE: a scalable framework for efficiently clustering transactional data. *Data Mining and Knowledge Discovery,* 20, 1-27.

Appendix A

Data matrix sort example

A simple example of the algorithm for the sort described in Section 2.1 is illustrated for a small dimensioned, non-sparse, data matrix. The less sparse example is used for ease in exposition and highlights more clearly the results of the sort. Row and column interchanges are recorded in parenthesis to the far right.

First, create the indicator matrix $\boldsymbol{D}^{(1)}$ of the data matrix $\boldsymbol{D}$,

$$\boldsymbol{D} = \begin{bmatrix} 200.03 & 105.65 & 111.12 & 160.21 & 21.56 \\ 20.17 & 0 & 12.65 & 170.26 & 222.87 \\ 333.65 & 0 & 137 & 180.64 & 124.34 \\ 400.56 & 90.65 & 140.65 & 191.25 & 241.28 \\ 50.54 & 100.26 & 155.26 & 205.64 & 0 \end{bmatrix} \rightarrow$$

$$\mathbf{D}^{(1)} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{pmatrix} \text{row 1} \\ \text{row 2} \\ \text{row 3} \\ \text{row 4} \\ \text{row 5} \end{pmatrix}$$

Then, counting cumulatively by column the nonzero entries in each row, and sorting by the final column (in bold), yields

$$\begin{bmatrix} 1 & 2 & 3 & 4 & \mathbf{5} \\ 1 & 1 & 2 & 3 & \mathbf{4} \\ 1 & 1 & 2 & 3 & \mathbf{4} \\ 1 & 2 & 3 & 4 & \mathbf{5} \\ 1 & 2 & 3 & 4 & \mathbf{4} \end{bmatrix}$$

$$\rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 & \mathbf{5} \\ 1 & 2 & 3 & 4 & \mathbf{5} \\ 1 & 1 & 2 & 3 & \mathbf{4} \\ 1 & 1 & 2 & 3 & \mathbf{4} \\ 1 & 2 & 3 & 4 & \mathbf{4} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{pmatrix} \text{row 1} \\ \text{row 4} \\ \text{row 2} \\ \text{row 3} \\ \text{row 5} \end{pmatrix}.$$

So far, row 4 has been moved to row position 2, with the remaining rows following in chronological order. Next, we repeat the above sort based on the columns using the transpose of the resultant row sorted indicator matrix above (far right matrix)

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 2 & 3 & 4 & \mathbf{5} \\ 1 & 2 & 3 & 4 & \mathbf{5} \\ 1 & 2 & 3 & 4 & \mathbf{5} \\ 1 & 2 & 3 & 4 & \mathbf{4} \\ 1 & 2 & 2 & 2 & \mathbf{3} \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \text{col 1} \\ \text{col 3} \\ \text{col 4} \\ \text{col 5} \\ \text{col 2} \end{pmatrix}.$$

Here, column (row) 2 is moved to the last position, and column (row) 5 to the next to last position; the remaining columns follow in chronological order in the top three positions. Transposing and translating back to the original values of the data matrix gives the final sorted data matrix $\boldsymbol{D}^{(*)}$,

$$\mathbf{D}^{(1)*} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

$$\rightarrow \mathbf{D}^{(*)} = \begin{bmatrix} 200.03 & 111.12 & 160.21 & 21.56 & 105.65 \\ 400.56 & 140.65 & 191.25 & 241.28 & 90.65 \\ 20.17 & 12.65 & 170.26 & 222.87 & 0 \\ 333.65 & 137 & 180.64 & 124.34 & 0 \\ 50.54 & 155.26 & 205.64 & 0 & 100.26 \end{bmatrix}.$$

Note that, we are not taking into account the actual value of the entries of the data matrix **D**. We could however do so easily by placing some sort of rank on each value. The most basic of which would be to assign each entry its actual ranked value. That is, for **D** above we could write $\boldsymbol{D}^{(1)}$ as,

$$\mathbf{D} = \begin{bmatrix} 200.03 & 105.65 & 111.12 & 160.21 & 21.56 \\ 20.17 & 0 & 12.65 & 170.26 & 222.87 \\ 333.65 & 0 & 137 & 180.64 & 124.34 \\ 400.56 & 90.65 & 140.65 & 191.25 & 241.28 \\ 50.54 & 100.26 & 155.26 & 205.64 & 0 \end{bmatrix} \rightarrow \mathbf{D}^{(1)} =$$

$$\begin{bmatrix} 17 & 7 & 8 & 13 & 3 \\ 2 & 0 & 1 & 14 & 18 \\ 21 & 0 & 10 & 15 & 9 \\ 22 & 5 & 11 & 16 & 20 \\ 4 & 6 & 12 & 18 & 0 \end{bmatrix},$$

and proceed as before by summing over the rows and sorting by the last column.

Appendix B

Canonical Variate Analysis (CVA)

The following is a summary of CVA on grouped, or clustered data, described in Krzanowski [13]. Suppose $\{\boldsymbol{x}_i, i = 1, \dots, n\}$ denotes a sample from the random vector $\boldsymbol{X} = (X_1, \dots, X_p)^T$. Let k denote the number of groups within the sample and $n_i, i \in \{1, \dots, k\}$, the number of observations in each group. The within group covariance matrix $\boldsymbol{W}$, the multivariate analogue of the univariate within sum of squares divided by the degrees of freedom, and similarly, the between group covariance matrix $\boldsymbol{B}$ are defined as

$$\boldsymbol{W} = \frac{1}{(n-k)} \sum_{i=1}^{k} \sum_{j=1}^{n_i} (\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}})(\boldsymbol{x}_{ij} - \bar{\boldsymbol{x}})^T \quad \text{and} \quad \boldsymbol{B} = \frac{1}{(k-1)} \sum_{i=1}^{k} n_i \, (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})(\bar{\boldsymbol{x}} - \bar{\boldsymbol{x}})^T.$$

Maximizing $\lambda = \boldsymbol{a}^T\boldsymbol{B}\boldsymbol{a}/\boldsymbol{a}^T\boldsymbol{W}\boldsymbol{a}$, with respect to the vector $\boldsymbol{a}_{p\times 1}$, is after differentiation, equivalent to solving for the vector $\boldsymbol{a}$ such that $\boldsymbol{B}\boldsymbol{a} - \lambda\boldsymbol{W}\boldsymbol{a} = 0$. This is equivalent to solving the following, $(\boldsymbol{B} - a\boldsymbol{W})\boldsymbol{a} = 0$, $(\boldsymbol{W}^{-1}\boldsymbol{B} - \lambda\boldsymbol{I})\boldsymbol{a} = 0$, or $(\boldsymbol{W}^{-1}\boldsymbol{B})\boldsymbol{a} = \lambda\boldsymbol{a}$. That is, $(\lambda, \boldsymbol{a} = \boldsymbol{e})$ are an eigen-pair for the matrix $\boldsymbol{W}^{-1}\boldsymbol{B}$. Since λ measures the ratio between group and within variation, larger values correspond to a larger separation between clusters. Hence, after the appropriate change in interpretation, this is directly comparable to performing PCA on $\boldsymbol{W}^{-1}\boldsymbol{B}$ instead of the covariance matrix.