Time Series Quantization and Compression subject to Distortion Measures

A.V. Poghosyan[1], A.N. Harutyunyan[1], N.A. Hovhannisyan[1], N.M. Grigoryan[1],
A.J. Han Vinck[2,3], and Y. Chen[2]
[1]VMware Eastern Europe
[2]Institute of Digital Signal Processing, University of Duisburg-Essen
[3]University of Johannesburg

Abstract
Efficiently storing storms of data to preserve their utility is a challenging data science problem in many technology applications such as management of cloud computing infrastructures. The problem is information-theoretic in nature. We consider various compression approaches (which are based on Machine Learning) to time series (metric) data measured from IT or cloud computing resources (virtual machine, host, application server, etc.) while monitoring those distributed systems. The main idea is to apply time series quantization and then compress the data with elimination of sequential duplications of values.

Introduction

Reliable management of modern cloud infrastructures and applications are increasingly relying on monitoring of massive volumes of time series data related to every aspect of the data center and storing them in specific data structures. This leads to a quite large storage consumption problem if considering also a vast number of self-generated time series needed for achieving the management goals. Commonly, it requires terabytes of storage space for archiving months/years of monitored data from IT resources/objects, each with hundreds/thousands of parameters (measured as individual time series metrics). We need a procedure to optimize data storage and information retrieval processes, especially for users/customers who need to store years of data using "realistic" storage space and disk I/O utilization requirements. Direct implementation of well-known compression methods will not solve the problem as some strict practical limitations exist:

a) Maintaining the common "time stamp - data value" structure of time series. Hence, we can't use, for example, vocabulary-based techniques for lossless compression or approximations, interpolations, and transformations for lossy compression;
b) Although, the storage space and I/O minimization have the highest priority, in general, a compromise between those minimizations and time of compression-decompression, CPU, Memory utilizations is a challenge. Hence, archiving approaches and approaches with heavy decompression are unacceptable.

We suggest a generic approach in line with the above-mentioned limitations – quantization of time series (lossy operation) and elimination of data value repetitions (lossless operation). This approach would be effective if the quantized time series is a low-variability data with big number of sequential data value repetitions. Quantization of time series is performed subject to distortion criteria imposed by users and preserving the main information content "enough" for effective management of the system. Distorted time series data can still be used for performing some analytical tasks such as outlier detection of new observations.

Below we discuss three different Machine Learning approaches (see [1,2]), targeting distinct use cases and complexity vs accuracy trade-offs. *Algorithm A* describes the process of quantization that maximizes the compression rate or minimizes the loss function. *Algorithm B* performs quantization with distortion depending on the data point importance – with lower distortion for important patterns and higher for the remaining. *Algorithm C* describes ideas for multi-variate time series with application of clustering methods. Related researches (for instance, [3]) in the domain of time series compression propose spline approximation-based algorithms. Our prior and related works include papers [4]-[8] linking also to relevant information-theoretic concepts.

Algorithm A
Optimal Quantization subject to Fidelity and Compression Rate

Under this scenario, the range of time series data is partitioned according to a set of quantiles. The quantized time series data is obtained from the original metric by substituting its values with the nearest quantiles. The quantized time series data is then compressed by removing sequential duplicate values. More specifically, quantization is performed by partitioning the metric range according to a selected number n of quantiles denoted by $q_1, \dots, q_n$. The set of quantiles $\{q_i\}_{i=1}^{n}$ divides the data range of time series data into $n+1$ groups of data points based on their values. Each group contains almost the same number of data points.

Let $x_k = x(t_k), k = 1, \dots, N$, be data points of a time series. Denote $X = \{x_k\}_{k=1}^{N}$. And let $x_k^q = x^q(t_k), k = 1, \dots, N$, be the nearest quantiles of the metric values corresponding to time stamps t_k. The sequence of quantized time series is represented by $X^q = \{x_k^q\}_{k=1}^{N}$.

Accuracy of this quantization can be measured, for instance, by ℓ_1-error (loss function)

$$l_1 = \frac{1}{N}\sum_{k=1}^{N}\left|x_k - x_k^q\right|.$$

Compression of the quantized metric is then performed by elimination of sequential repetitions, or consecutive duplications, of quantized data points x_k^q from the quantized data set X^q. Hence, we get a loosely compressed metric $x_k^c = x^c(t_k), k = 1, \dots, M$, where $M \leq N$. Denote $X^c = \{x_k^c\}_{k=1}^{M}$.

The compression rate is given by

$$CR = 100\frac{N-M}{N}.$$

The number of quantiles, n, is ideally selected to minimize the loss function and to maximize the compression rate.

However, simultaneous minimization of the loss function and maximization of the compression rate are contradicting operations.

Therefore, we consider two optimization set-ups. In the first set-up, the user requirement on the quality of compressed data is controlled by an upper bound (Δ) on the loss function (LF), while the objective is to maximize the compression rate:

$$\begin{aligned} CR &\rightarrow max \\ LF &\leq \Delta. \end{aligned} \tag{1}$$

According to the second optimization setting, the user is interested in at least r% of storage reduction subject to minimum loss:

$$\begin{aligned} CR &\geq r \\ LF &\rightarrow min. \end{aligned} \tag{2}$$

Optimization problems in (1) and (2) may result in different optimal number of quantiles *n* to work with.

Figures 1 and 2 consider a specific infrastructure health metric and its quantization with four quantiles. Here, $n = 4$, $q_1 = 25$ (0.06-th quantile), $q_2 = 33$ (0.2-th quantile), $q_3 = 73$ (0.38-th quantile), $q_4 = 78$ (0.66-th quantile), $\ell_1/mean(data) = 0.026$ and $CR = 98.6\%$.

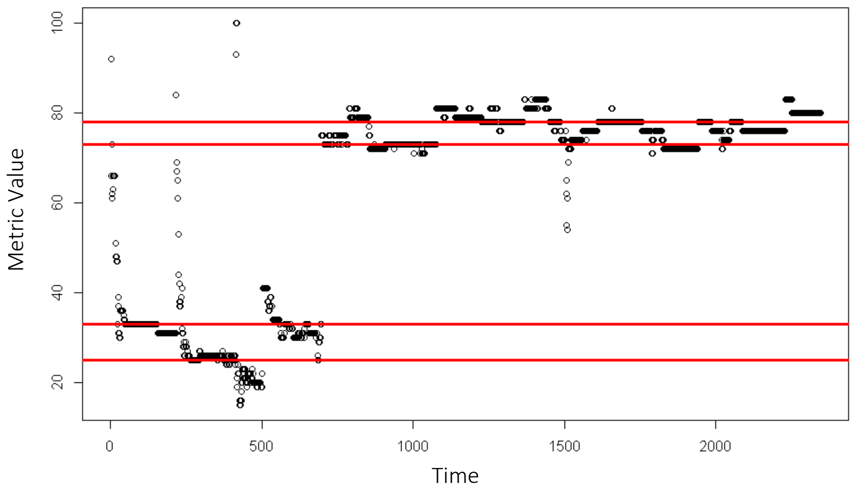

Fig. 1. Red solid lines show the quantiles which minimize the ℓ_1 error.

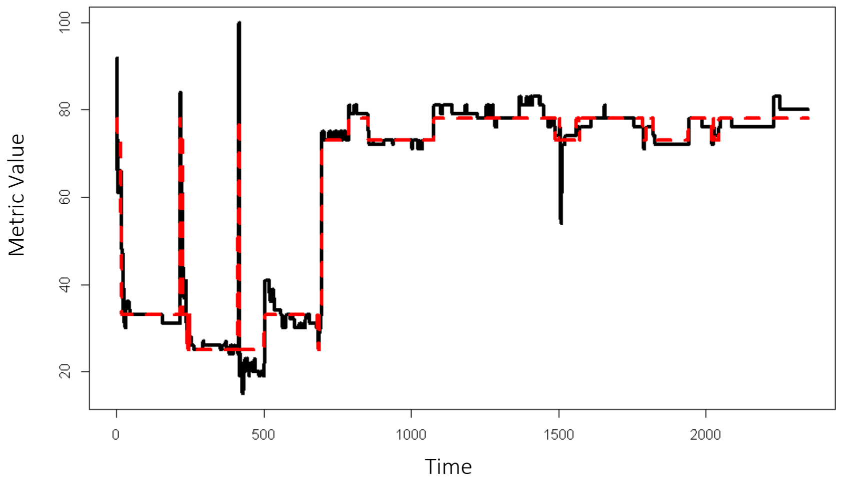

Fig. 2. Original (black solid) and quantized (red dashed) time series for $n = 4$, where the corresponding quantiles minimize the ℓ_1 error.

Algorithm B
Compression based on Data Importance Patterns

Algorithm A quantizes the time series subject to a distortion measure. Assume that the data values in the metric are associated with different degrees of "importance" (can be treated based on a cost function). For example, data points from cluster "A" have "high importance", values from cluster "B" have "moderate importance", and values from cluster "C" have "low importance". Then, the compression is performed linked to the importance cost of the cluster. For example, we can apply lossless compression for the most important cluster "A" and lossy compression or lower accuracy compression for "B" and "C".

In general, importance of data points relates to behavioral patterns of time series data and can be different while working with performance, capacity, or configuration metrics. For example, in case of performance metrics, the importance of a data point relates to its "participation" in an anomaly process. For identifying a performance anomaly, dynamic or hard thresholding (see [9]-[11]) techniques can be applied. Data points that violate thresholds should be more important than data points within thresholds. Hence, while compressing historical data with known anomaly patterns, it is natural to preserve those anomalies (outliers) and largely compress/reduce within-thresholds data points by acceptable low accuracy.

Now, assume that H and L are upper and lower thresholds, respectively. Our compression approach is summarized in the following 5 steps:

Step 1. Determine the points satisfying the condition $x_k > H$ and store them without distortion with their corresponding time stamps.

Step 2. Determine the points satisfying the condition $x_k < L$ and store them without distortion with their corresponding time stamps;

Step 3. According to the required accuracy (say, user defined), determine the value of parameter n indicating the number of slices the interval $[L, H]$ should be divided into by the following straight lines:

$$c_k = L + \frac{H-L}{n}k, k = 0, \dots, n.$$

Step 4. Determine the points in the intervals $I_k = [c_k, c_{k+1}), k = 0, \dots, n-2, I_{n-1} = [c_{n-1}, c_n]$, and calculate the median, the average or any other reasonable statistical measure for each group of data points within I_k. Denote those measures by $m_0, \dots, m_{n-1}$. If a data point belongs to the I_j interval, then change its value to m_j. This results in a quantized time series with distorted in-bound and exact out-bound data points;

Step 5. Eliminate data value duplications with the corresponding time stamps. This results in a compressed time series. More duplications imply higher compression rates.

Let $v_k = x_{k+1} - x_k, k = 1, \dots, N-1$, be the corresponding variability metric and V= $\{v_k\}$. We define variability measure (VarM) of a time series as a percentage of jumps in X

$$VarM = 100\frac{Number\ of\ Non-Zero\ Components\ in\ V + 1}{Number\ of\ All\ Components\ in\ X}$$

$VarM = 0\%$ characterizes constant time series X with $x_k \equiv c$. Conventionally, if $VarM \leq 50\%$, we deal with a low-variability metric, otherwise it is a high-variability metric.

$VarM$ can be determined both for original and compressed time series, $VarM(Original)$ and $VarM(Quantized)$, respectively. $VarM(Quantized)$ characterizes the compression rate (CR) of the above described algorithm defined as $CR = 100 - VarM(Quantized)$.

We see that the efficiency of our approach is strictly relates to the variability of the quantized time series. Of course, this approach can be applied immediately to a time series without quantization, but it will lead to poor results for high-variability metrics.

Figures 3 and 4 show the compression process of a high-variability time series ($VarM = 99.4\%$) while dividing within Dynamic Thresholds (DT – the typical time-varying ranges of time series, see [10]) area into $n = 2$ equal parts. In this case, the quantized time series is a low-variability data with $VarM = 33.2\%$. After elimination of duplications we get compression rate $CR = 66.8\%$ and $\ell_1 = 0.08$.

Now we describe the compression process for an entire database containing 86725 time series (metrics) from 309 different objects (VMs). Each of those time series have 1000-9300 data points. We found that 63.3% of those metrics are low-variability and 36.6% are high-variability (see Fig. 5).

Fig. 6 shows compression rates of high-variability data when $n = 10$. Compression rates of low-variability metrics range from 50% to 100% and are mostly concentarted around 100%. Fig. 7 demonstrates relative (divided over the averages) ℓ_1-errors of quantization of high-variability time series. They are concentrated around 0.02. Similar errors of low-variability metrics are ranging from 0 to 0.04 but mostly are concentrated around 0.

Fig. 8 indicates average compression rates of different metric categories vs different number of within-DT segments. We see that compression rates of high-variability metrics are very sensitive to parameter n and relatively acceptable values of n are $n = 1,2,3$.

Fig. 9 graphs the average errors for different metric categories vs different number of within-DT segments. Overall, the trade-off between compression rate and error of quantization is achieved when $n = 1,2,3$.

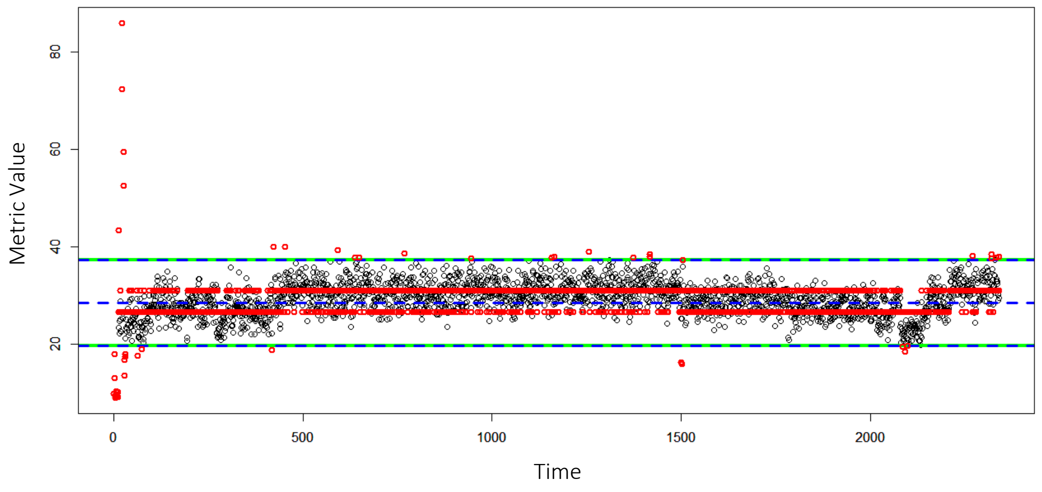

Fig. 3. Original high-variability time series with upper and lower dynamic thresholds (green solid lines). Red points show the values of the quantized time series.

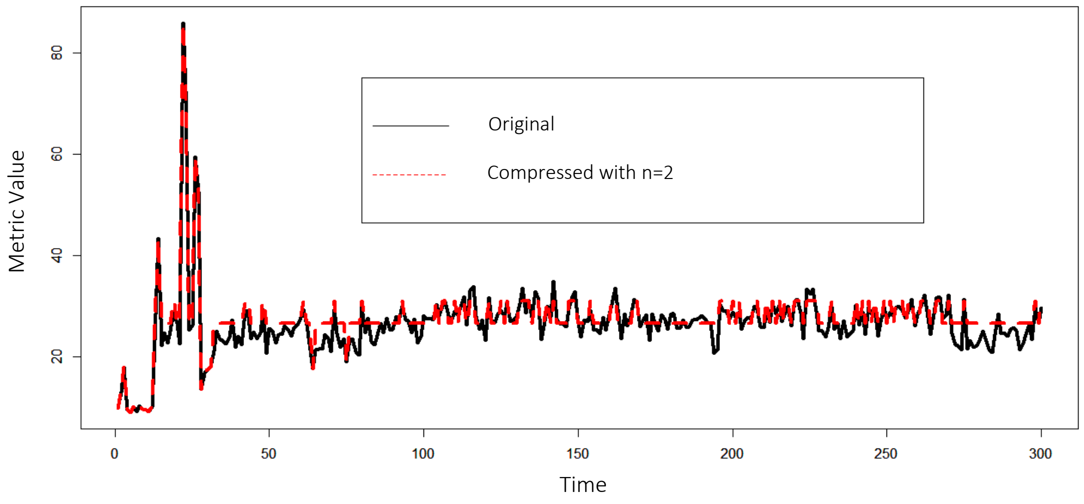

Fig. 4. Original high-variability time series (black solid) and quantized low-variability time series (red dashed).

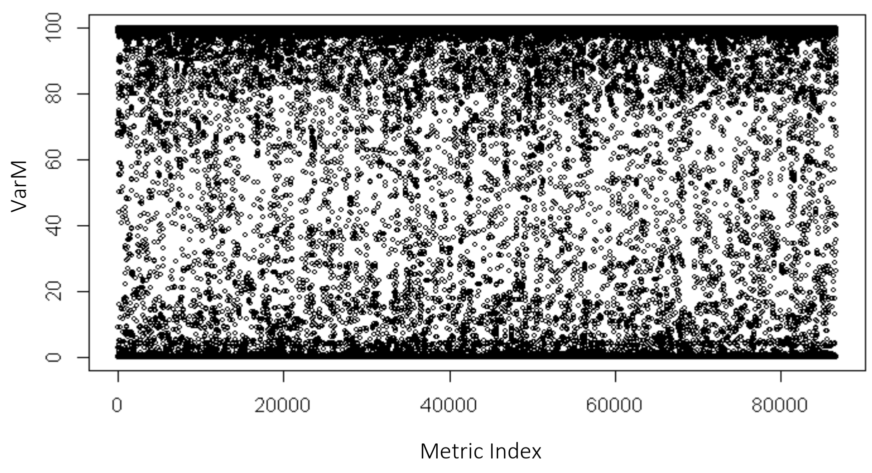

Fig. 5. Variability measures for different time series.

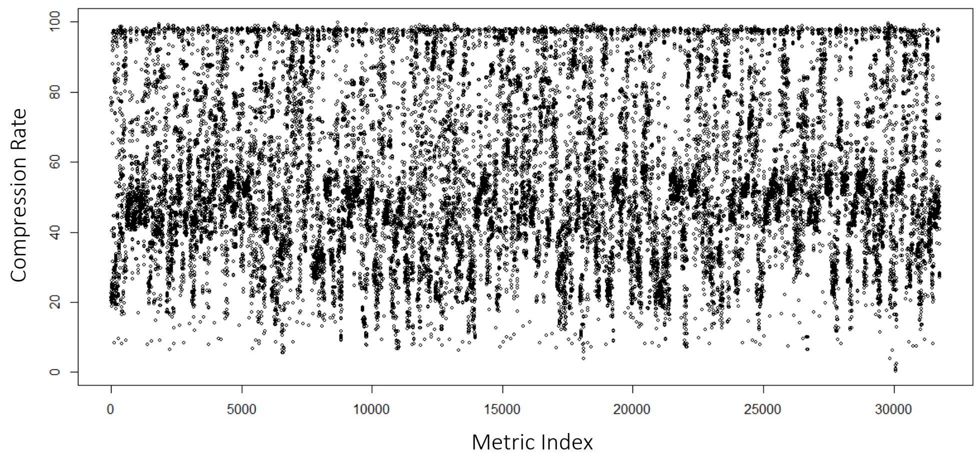

Fig. 6. Compression rates for high-variability metrics when $n = 10$.

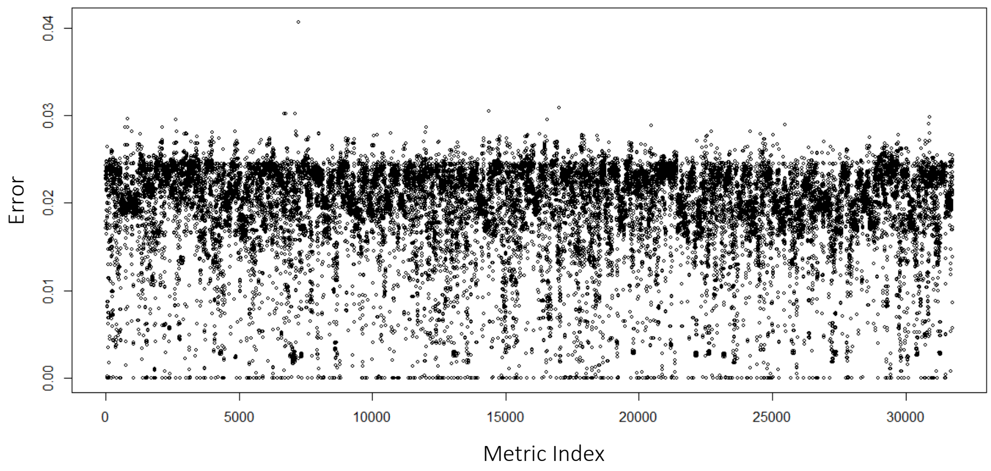

Fig. 7. Relative ℓ_1-errors after quantization for high-variability metrics when $n = 10$.

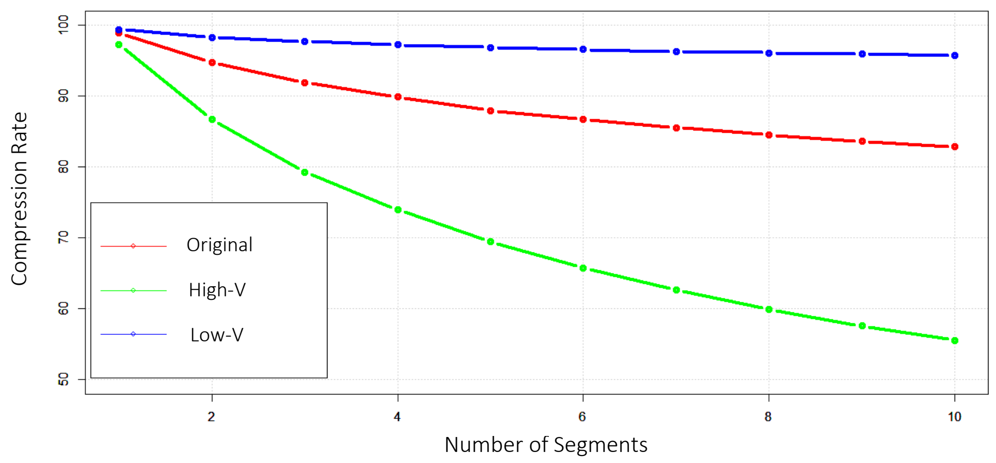

Fig. 8. Average compression rates for different metrics categories vs different number of within-DT segments.

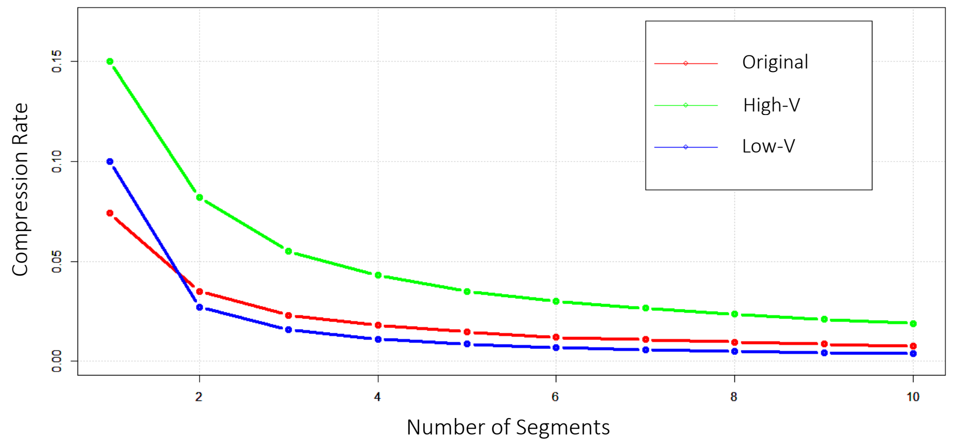

Fig. 9. Average errors for different metric categories vs different number of within DT segments.

Algorithm C
Multidimensional Data

Time series metrics can be aggregated into a multi-dimensional representation to analyze their trade-off behavior using machine learning/clustering. In other words, the values of n different metrics at time stamp t: $m(t) \equiv (m_1(t), m_2(t), \dots, m_n(t)) \in R^{n+1}$, make a point $m(t)$ in $(n+1)$-dimensional space, where one of the axes is the time.

In many cases, the time can be excluded from the multi-dimensional representation making a point m_k in n-dimensional space $m = \{m_k\}_{k=1}^{N}$, $m_k = \left(m_{1,k}, m_{2,k}, \dots, m_{n,k}\right) \in R^n$. We want to evaluate the historically typical trade-off between those metrics by analyzing the above-mentioned vectors. This leads to application of machine learning techniques of clustering ([12,13]) to identify typical spaces $m(t)$ or m stays within.

How to store the high volumes of historical multidimensional data in a reduced way that it still contains underlying utility or “important patterns”? In particular, how to optimally compress the above-mentioned multi-variate time series data subject to a required fidelity measure d?

Such a goal statement implies an optimization problem of finding the best "Δ-coverage” of the data set according to the fidelity or distortion measure d. This coverage is defined by a subset of vectors

$$m_k = \{m_{1,k}, m_{2,k}, \dots, m_{n,k}\}, k = 1, \dots, K$$

such that for every other m_k from the data set there is a point $m_{i_0}, 1 \le i_0 \le K$ satisfying the condition $d(m_k, m_{i_0}) \le \Delta$.

The Δ-coverage with size K is called optimal if there is no other coverage of lesser size. In an implementation, d can be the Euclidean distance together with the K-means clustering approach:

1) to find the K centroids of the data set;
2) verify if the distortion requirement Δ is met for all clustered points;
3) if it is not met, increase K by 1 and repeat the procedure until it is satisfied.

As soon as the final centroids are found and the optimal coverage is achieved, our data reduction coding procedure collapses all within-cluster data points into corresponding centroids, so we store only those centroids and thus preserve the required fidelity in the data representation.

This reduced data set can be still used to perform streaming compression of newly arrived observations into one of the closest centroids (k-nearest neighbor logic) subject to distortion level. If that is not possible, then compose a new centroid and proceed further.

Above described procedure might lead to an unacceptable big value for K in case of some outliers in multi-dimensional data. We consider an outlier-Δ-coverage assuming to preserve outliers exactly and apply optimal Δ-coverage for the remaining data points. This will help to achieve the same distortion with lesser number of clusters or better accuracy with the same coverage.

We selected two metrics from our private environment. One is “Badge-Health-Classic” taken as the first component of m_k and the other is “CPU-0-Ready-Summation” taken as the second component (see Fig. 10). Then we performed K-means clustering with different $K \ge 1$ number of clusters. Assume that $c_j = (c_j^1, c_j^2)$ is the j-th cluster centroid. If a point (x_k, y_k) belongs to the j-th cluster, we encode its value as (c_j^1, c_j^2). Later, for each K, we calculate ℓ_{max} -error (see Fig. 11). Formal definitions are as follows. Let $m_k = (x_k, y_k), k = 1, \dots, N$, and distances are measured as follows:

$$d_1\left(m^{(1)}, m^{(2)}\right) = \max_k \left(\left(x_k^{(1)} - x_k^{(2)}\right)^2 + \left(y_k^{(1)} - y_k^{(2)}\right)^2 \right)^{1/2}$$

or $d_2\left(m^{(1)}, m^{(2)}\right) = \operatorname{mean}_k \left(\left(x_k^{(1)} - x_k^{(2)}\right)^2 + \left(y_k^{(1)} - y_k^{(2)}\right)^2 \right)^{1/2}$, where $m_k^{(s)} = \left(x_k^{(s)}, y_k^{(s)}\right), s = 1,2$. It is natural to use relative measures, the results for different metrics can be compared easily. If $m_k^{(1)}$ is the original data and $m_k^{(2)}$ is its quantized version, then

$$\ell_{max} = \frac{d_1\left(m^{(1)}, m^{(2)}\right)}{\left\| m_k^{(1)} \right\|_{max}} = d_1\left(m^{(1)}, m^{(2)}\right) / \max_k \left(\left(x_k^{(1)}\right)^2 + \left(y_k^{(1)}\right) \right)^{1/2}$$

or

$$\ell_2 = \frac{d_1\left(m^{(1)}, m^{(2)}\right)}{\left\| m_k^{(1)} \right\|_{\ell_2}} = d_1\left(m^{(1)}, m^{(2)}\right) / \operatorname{mean}_k \left(\left(x_k^{(1)}\right)^2 + \left(y_k^{(1)}\right) \right)^{1/2}.$$

It is interesting that in case of ℓ_{max}-error, the impact of number of clusters is significant only for $K = 6$. This is due to some outliers that we see in the data which are far from 2 dominating dense regions. Figures 12 and 13 show the results of K-means clustering when $K = 2$ and $K = 6$. Comparison of those figures explains the jump in ℓ_{max} for $K = 6$ where one of the clusters contains only 4 points (visually perceived as outliers). This can be a useful procedure for outlier detection – determine the clusters with small percentage of points and define those points as outliers. If the outliers are known, then we can preserve them exactly and encode the remaining normal points by the given Δ-accuracy thus concluding the idea of outlier-Δ-coverage.

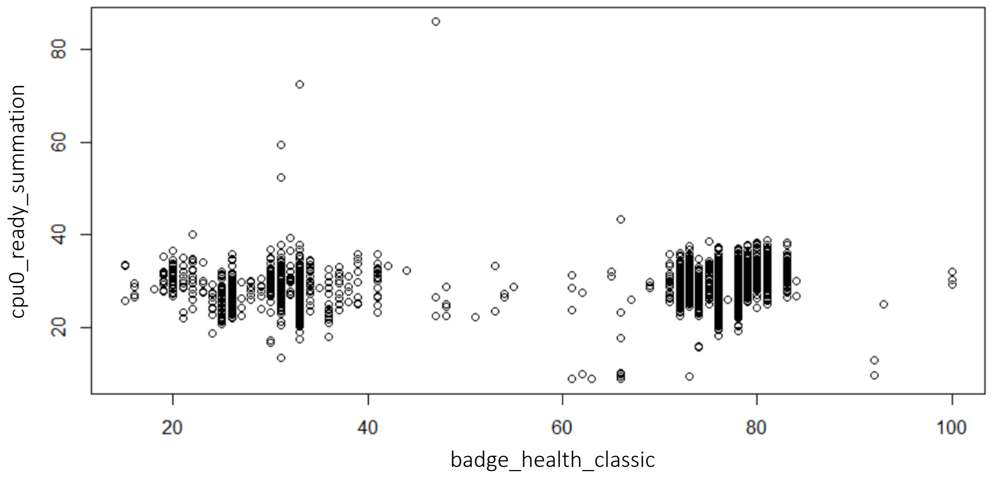

Fig. 10. Original 2D-data that we use for experiments.

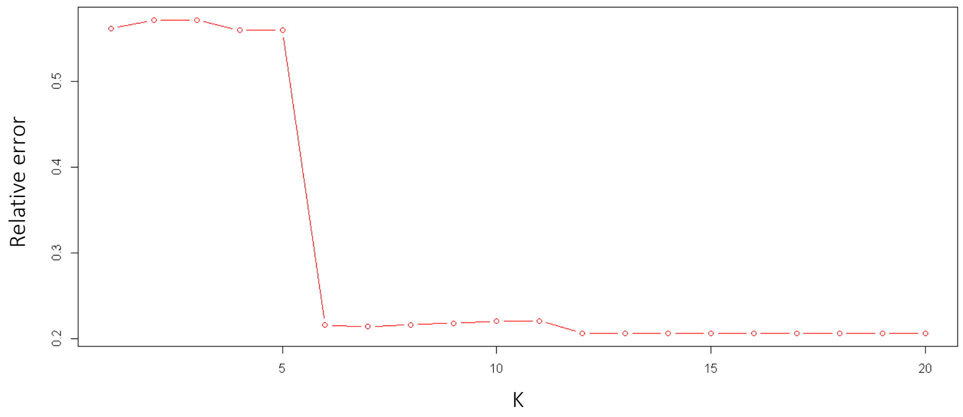

Fig. 11. $\ell_{\max}$ error for different number of clusters while applying K-means clustering for optimal coverage.

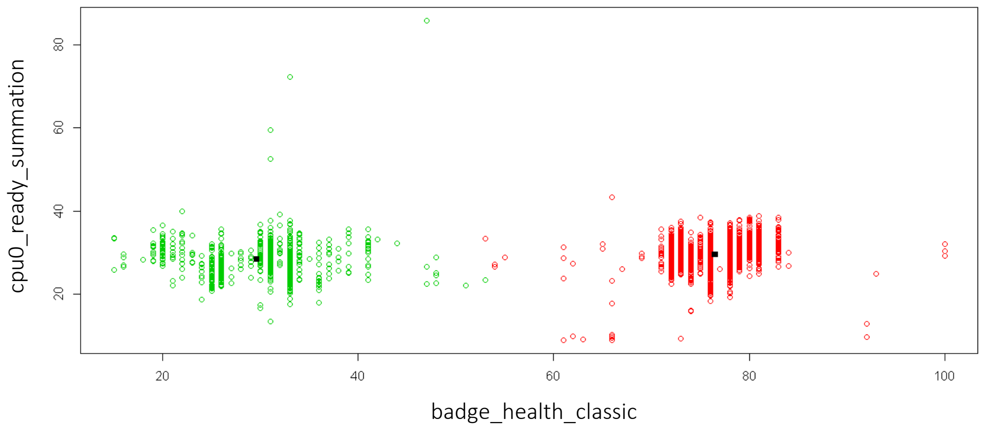

Fig. 12. Clustering of data in Fig. 10 by means of K-means when $\mathrm{K} = 2$.

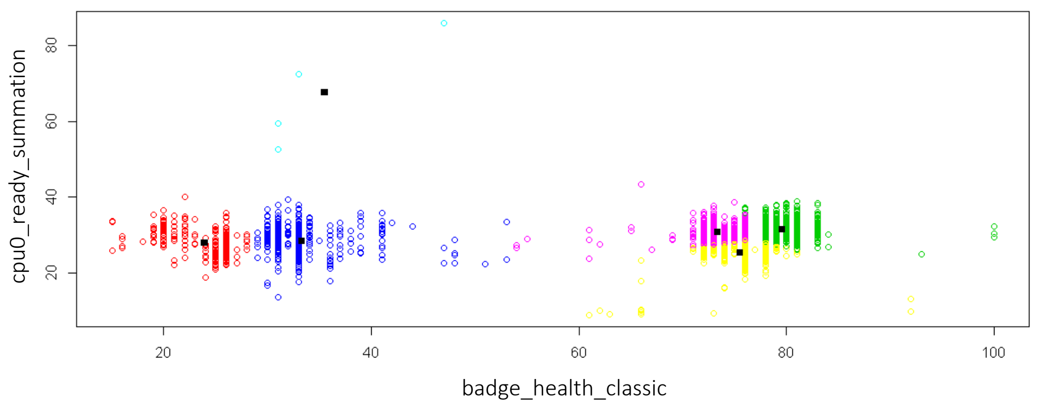

Fig. 13. Clustering of data in Fig. 10 by means of K-means when $\mathrm{K} = 6$.

In Figures 14 and 15, we consider another approach for outlier detection by exploring a within-cluster variability and by defining a normal shift from the cluster centroids. For each j-th cluster we define a radius R_j of normality from the centroid $c_j = (c_j^1, c_j^2)$ as follows:

$$R_j = 3 * \max_k \left(\left(x_k - c_j^1\right)^2 + \left(y_k - c_j^2\right)^2 \right)^{1/2}$$

where the number 3 is a parameter value and the point (x_k, y_k) belongs to the j-th cluster. If a point m_k belongs to the j-th cluster and its distance from the cluster centroid c_j is less than or equal to R_j then we encode it as c_j. If a point lays outside of all normalcy circles, we preserve its exact value without encoding. By changing the number of clusters, we can try to find a minimal (optimal) coverage satisfying the required accuracy.

Fig. 16 shows the values of ℓ_{max} for different values of K while constructing the outlier-optimal coverages by the described procedure. Although for small values of K we managed to decrease the errors, for larger values the impact of outliers is still present. Our final recommendation will be combination of both approaches – first, elimination of clusters with small number of points as outliers, and second, calculation of within-cluster variability only for the big ones.

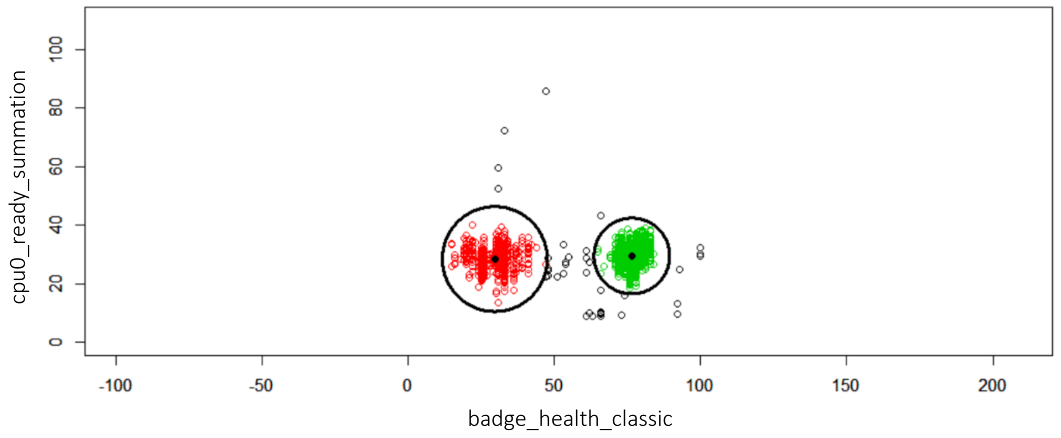

Fig. 14. Result of the K-means clustering with $K = 2$ with the corresponding normalcy circles.

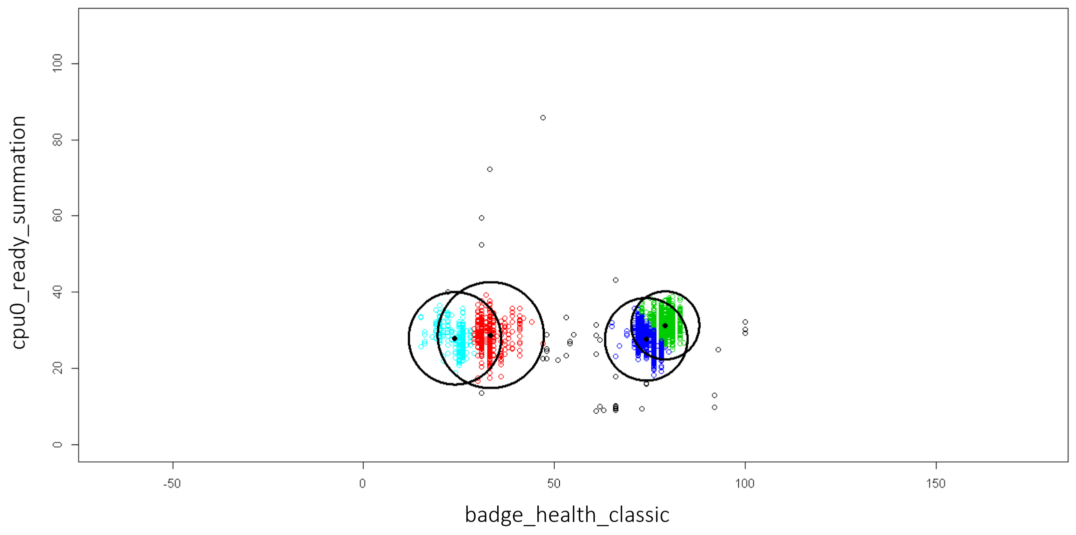

Fig. 15. Result of the K-means clustering with $K = 4$ with the corresponding normalcy circles.

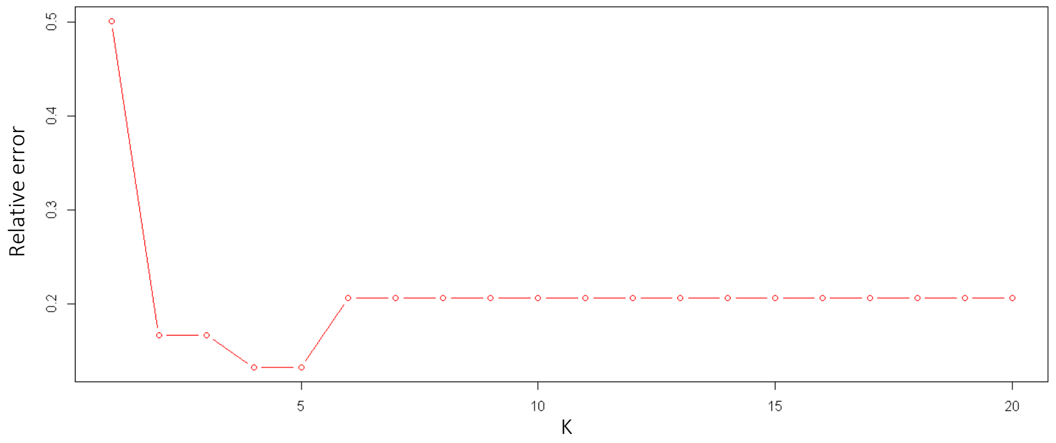

Fig. 16. $\ell_{\max}$ -errors for different values of K while constructing the outlier-optimal-coverage by means of K-means clustering.

References

[1] A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, Methods and Systems to Quantize and Compress Time Series Data. Patent application US 2018/0365301 A1. Filed: June 20, 2017. Published: Dec 20, 2018. Application No: 15/627,925, 2018.

[2] A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, Methods and Systems to Reduce Time Series and Detect Outliers. Patent application US 2018/0365298 A1. Filed: June 20, 2017. Published: Dec 20, 2018. Application No: 15/627,987, 2018.

[3] D. Li, W. Huangfu, and K. Long, Spline approximation-based data compression for sensor arrays in the wireless hydrologic monitoring system, Int. Journal of Distributed Sensor Networks 13(2), Feb. 2017.

[4] A.N. Harutyunyan, A.V. Poghosyan, and N.M. Grigoryan, Experiences in building an enterprise data analytics, Proc. Int. Workshop on Information Theory and Data Science: From Information Age to Big Data Era, pp. 89-94, Oct 3-5, Yerevan, Armenia, 2016.

[5] A.J. Han Vinck, Information theory and big data: Typical or not typical, that is the question, Proc. Workshop on Information Theory and Data Science: From Information Age to Big Data Era, pp. 5-8, Oct. 3-5, Yerevan, Armenia, 2016.

[6] A.V, Poghosyan, A.N. Harutyunyan, N.M. Grigoryan, Compression for time series databases using principal and independent component analysis, Proc. IEEE International Conference on Autonomic Computing (ICAC), July 17-21, Columbus, Ohio, US, 2017.

[7] A.N. Harutyunyan, A.V. Poghosyan, and N.M. Grigoryan, On compression of time series databases, Proc. 10th Asia-Europe Workshop on Concepts in Information Theory and Communications, June 21-23, Boppard, Germany, 2017.

[8] A.N. Harutyunyan, Y. Chen, and A.J. Han Vinck, Thoughts on information-theoretic aspects of several problems in data science, Int. Workshop on Collaborative Technologies and Data Science in Smart City Applications (CODASSCA), Sep. 12-15, Yerevan, Armenia, 2018.

[9] M.A. Marvasti, A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, "An enterprise dynamic thresholding system", IEEE Int. Conf. on Autonomic Computing, June 18-20, Philadelphia, PA, pp. 129-135, 2014.

[10] A.V. Poghosyan, A.N. Harutyunyan, N.M. Grigoryan, and M.A. Marvasti, Data-Agnostic Anomaly Detection. US Patent 10,241,887 B2, Filed: Mar 29, 2013. Published: Oct 2, 2014. Application No: 13/853,321, 2019.

[11] M.A. Marvasti, A.V. Poghosyan, A.N. Harutyunyan, and N.M. Grigoryan, Automated Methods and Systems for Calculating Hard Thresholds. Patent US 9,996,444 B2. Granted: Jun 12, 2018. Filed: Jun 25, 2014, 2018.

[12] M.M. Breunig, H.-P. Kriegel, R.T. Ng, J. Sander, LOF: Identifying density-based local outliers, Proc. 2000 ACM SIGMOD Int. Conf. on Management of Data, pp. 93–104, 2000.

[13] M. Ester, H.P. Kriegel, J. Sander, and X. Xu, A density-based algorithm for discovering clusters in large spatial databases with noise, In KDD*, v*ol. 96, no. 34, pp. 226–231, 1996.