Received: 1 November 2020 | Revised: 10 March 2021 | Accepted: 22 March 2021 | Published on: 8 April 2021

DOI: 10.1002/sam.11508

WILEY

RESEARCH ARTICLE

Supervised compression of big data

V. Roshan Joseph[1] | **Simon Mak**[2]

[1]Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA

[2]Department of Statistical Science, Duke University, Durham, North Carolina, USA

Correspondence
V. Roshan Joseph, Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, 755 Ferst Dr NW, Atlanta, GA 30332, USA.
Email: roshan@gatech.edu

Funding information
NSF CSSI Frameworks, Grant/Award Number: 2004571; U.S. National Science Foundation (NSF), Grant/Award Number: CMMI-1921646

Abstract

The phenomenon of big data has become ubiquitous in nearly all disciplines, from science to engineering. A key challenge is the use of such data for fitting statistical and machine learning models, which can incur high computational and storage costs. One solution is to perform model fitting on a carefully selected subset of the data. Various data reduction methods have been proposed in the literature, ranging from random subsampling to optimal experimental design-based methods. However, when the goal is to learn the underlying input–output relationship, such reduction methods may not be ideal, since it does not make use of information contained in the output. To this end, we propose a supervised data compression method called `supercompress`, which integrates output information by sampling data from regions most important for modeling the desired input–output relationship. An advantage of `supercompress` is that it is nonparametric—the compression method does not rely on parametric modeling assumptions between inputs and output. As a result, the proposed method is robust to a wide range of modeling choices. We demonstrate the usefulness of `supercompress` over existing data reduction methods, in both simulations and a taxicab predictive modeling application.

KEYWORDS

clustering, data reduction, experimental design, K-means algorithm, subsampling

1 | INTRODUCTION

The phenomenon of big data has become ubiquitous in nearly all disciplines, from science to engineering. With this wealth of information, however, a key challenge is extracting useful information from the big data in a timely fashion, particularly given limited computational resources. As statistical and machine learning models become more complex, with hundreds to millions of parameters to tune, the size of the training dataset becomes an increasingly significant computational bottleneck. One possible solution is to train the desired model on a *reduced* dataset: by carefully choosing this reduced dataset from the big data, for example, 100,000 out of a billion samples, one can use this for efficient model building and inference. This data reduction approach is crucial for scaling up a wide range of learning methods in big data applications (e.g., [6, 17]).

Much work in the literature on data reduction is on random subsampling, that is, choosing a subsample from the big data uniformly at random. For specific modeling approaches, considerable improvements to random subsampling can be realized by using nonuniform probabilities for sampling, for example, for fitting linear regression models [9, 16]. Recently, Wang et al. [21] developed information-based selection of subsamples using

ideas from optimal experimental design. Extensions of these approaches to classification using logistic regression have also been developed [20, 22]. There has also been an analogous line of work on coresets in the computer science literature [11].

In practical problems, however, model building is an iterative process: the modeler may wish to try out different modeling approaches (e.g., Gaussian processes or neural networks) besides linear regression, then choose the method which fits the data the best. However, the aforementioned methods assume an underlying parametric model (e.g., linear regression) for data reduction. Such methods can therefore give an excellent reduction of the big data when the model is specified correctly, but can yield very poor reduction performance when the model is misspecified. A simple illustration is shown in Figure 1. The left panel shows a large sample generated from a mixture normal distribution (the "big data"), and the next panel shows a reduction of this into 100 points using the information-based optimal subsample (IBOSS; [21]) method, which assumes an underlying linear model. We see that IBOSS selects extreme points on the boundaries of the big data, which are known to have high influence (or leverage) on a linear regression model fit. However, if the modeler wishes to fit more nonlinear or nonparametric models, the IBOSS subsample would clearly not be ideal for fitting such models. Given the iterative nature of model building and the computational burdens of model fitting with big data, a *model-independent* (or *robust*) data reduction strategy is oftentimes preferred—one which provides good model fitting performance over a wide range of modeling choices.

A reasonable model-independent data reduction approach might be to find representative points on the input features via a clustering approach, for example, k-means clustering. The third panel of Figure 1 shows this reduction using k-means cluster centers for the earlier example. We can see that the cluster centers (as reduced data points) provide a good representation of the input features, and can thus be robust to the modeling choices. However, k-means may not result in a *subsample* of the big data, in that the reduced dataset may not be a subset of the larger dataset. Therefore, it can be viewed as a compression method rather than a data reduction method. As we will see later, this will not be an issue for modeling. If the goal is to attain a subsample, then one can take the closest data points to the compressed sample using a nearest neighbor algorithm [13]. Because of this, we might use the terms data reduction and data compression interchangeably.

The above methods, while robust, are "unsupervised" in the sense that they use data only in the input variables (call this $\mathbf{x}$), but does not incorporate information of the response variable (call this y). Figure 1 also shows true input–output relationship as a colored image. We can see that IBOSS has missed the interesting regions of the relationship, whereas k-means clustering seems to oversample from inactive regions of the space. Clearly, we could get a much better subsample using the information on the response variable y from the big data. How such information can be used efficiently to select subsamples seems to be an understudied problem, but could lead to substantial improvements for model building. Developing such a supervised reduction or compression method is the main aim of this article.

The article is organized as follows. Section 2 provides a formulation of the proposed `supercompress` method and discusses a heuristic algorithm for solving this optimization problem. Section 3 investigates the robustness of this method and proposes a modification of `supercompress` which can improve robustness over different modeling choices. Section 4 discusses several numerical examples comparing the proposed `supercompress` methods with existing approaches. Section 5 presents a semiparametric extension of the `supercompress` methods. Section 6 compares our approach to existing methods on a real-world predictive modeling problem for taxicab scheduling. We conclude in Section 7 with remarks and future research directions.

2 | SUPERVISED COMPRESSION VIA CLUSTERING

We first introduce some notation. Let $(\mathbf{X}_j, Y_j), j = 1, \dots, N$ be the big data we wish to reduce, where $\mathbf{X}_j$ is the input variables (or features) and Y_j is its corresponding response variable for the jth data point. Assume that both the input and response variables are either standardized to have zero mean and unit variance. Our aim is to reduce or compress the big data to a smaller set $(\mathbf{x}_i, y_i)$, $i = 1, \dots, n$, where $n \ll N$. Once compression is performed, we will "throw away" the big data to save storage space, and use the compressed dataset for model fitting.

2.1 | Mathematical formulation

We will first use a simple nonparametric modeling approach to motivate an optimality criterion for data reduction, which we later modify for improved robustness. Consider the nearest neighbor method for prediction, one of the most basic modeling methods which can be used as a building block in more sophisticated models. To predict at

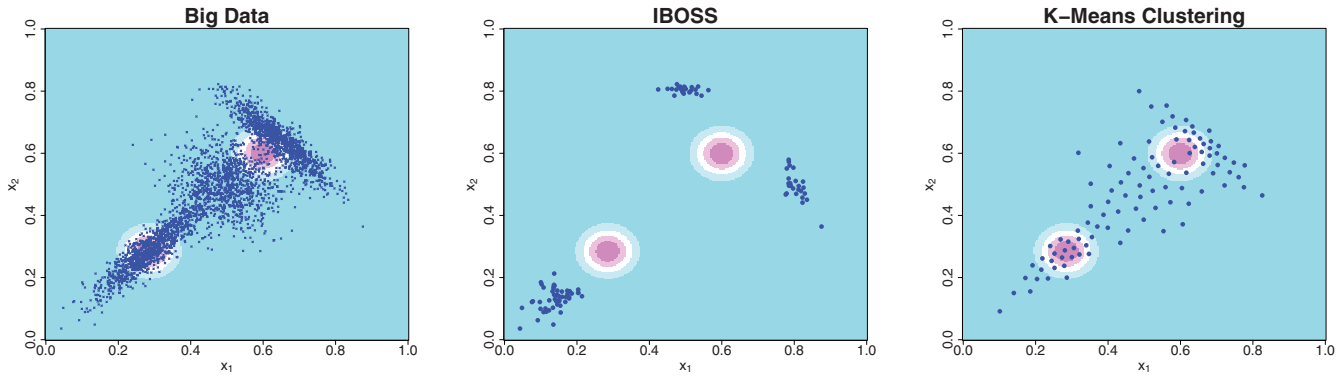

FIGURE 1 From left to right: Big data, a sample of 100 points using information-based optimal subsample (IBOSS), and k-means clustering. The true input–output relationship is shown as an image

a given $\mathbf{x}$, we first need to find the closest point $\mathbf{x}_{i^*}$, where

$$i^* = \underset{i=1:n}{\text{argmin}} \|\mathbf{x} - \mathbf{x}_i\|,$$

where $\|\cdot\|$ denotes the Euclidean distance. Thus, for a given set of points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, we can partition the index set of big data, namely $\{1, \ldots, N\}$, into index sets I_i, $i = 1, \ldots, n$, which collect all indices of points in the big data closest to point $\mathbf{x}_i$:

$$I_i = \{j : \ \|\mathbf{X}_j - \mathbf{x}_i\| < \|\mathbf{X}_j - \mathbf{x}_{i'}\| \ \text{ for all } \ i' \neq i\}, \quad i = 1, \ldots, n. \tag{1}$$

Clearly, the union of these sets recovers the full index set, that is, $\bigcup_{i=1}^n I_i = \{1, \ldots, N\}$, and the partition is also disjoint, that is, $I_i \cap I_j = \emptyset$ for $i \neq j$. The nearest-neighbor regions in (1) are also known as *Voronoi regions* [7].

With this, for any $\mathbf{X}_j$, $j \in I_i$, we will use the prediction y_i within region I_i, where this value can be chosen to minimize a desired loss function. Using the ubiquitous least squares loss, the prediction y_i within region I_i can be chosen to minimize

$$L_i = \sum_{j \in I_i} (Y_j - y_i)^2. \tag{2}$$

This gives the familiar solution of the cluster mean

$$y_i = \frac{\sum_{j \in I_i} Y_j}{N_i},$$

where N_i is the number of points in cluster i. The total loss function is then given by

$$L = \sum_{i=1}^n \sum_{j \in I_i} (Y_j - y_i)^2. \tag{3}$$

The proposed method `supercompress` aims to find the reduced points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, with corresponding partition $I_1, \ldots, I_n$, which minimize the loss function L. This then yields the reduced dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, which can be used for model fitting.

At first glance, the minimization of (3) might look like a standard clustering problem, but it is not! The standard clustering problem (on the y-space) is to minimize the loss function

$$\sum_{i=1}^n \sum_{j \in J_i} (Y_j - y_i)^2 \tag{4}$$

with respect to $\{y_1, \ldots, y_n\}$ and partitions $J_1, \ldots, J_n$, where

$$J_i = \{j : |Y_j - y_i| < |Y_j - y_{i'}| \ \text{ for all } \ i' \neq i\}, \quad i = 1, \ldots, n$$

is the index set of points whose response is closest to y_i. The loss function (4) can be efficiently solved using iterative algorithms such as k-means clustering [15, 18]. Although the objective functions (3) and (4) look similar, the optimization in our problem (i.e., the former) is with respect to the *input* points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and corresponding partition of the input space $I_1, \ldots, I_n$. Because each response variable y can produce many $\mathbf{x}$ values, the input partitions $I_1, \ldots, I_n$ can induce a corresponding partition $J_1, \ldots, J_n$ in the y-space which may be overlapping. This makes our problem much more difficult to solve.

It is also useful to compare the proposed formulation (3) with the traditional k-means clustering formulation in the $\mathbf{x}$-space, which aims to minimize

$$\sum_{i=1}^n \sum_{j \in I_i} \|\mathbf{X}_j - \mathbf{x}_i\|^2 \tag{5}$$

with respect to input points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and its corresponding partition $I_1, \ldots, I_n$. Given a fixed partition, optimizing (5) over $\mathbf{x}_i$'s gives the familiar cluster mean solution $\mathbf{x}_i = \sum_{j \in I_i} \mathbf{X}_j / N_i$, $i = 1, \ldots, n$. Similarly, given fixed

points $\mathbf{x}_1, \ldots, \mathbf{x}_n$, optimizing (5) over partition $I_1, \ldots, I_n$ yields the nearest-neighbor (Voronoi) regions in (1). The well-known k-means clustering algorithm iterates these two conditional optimization steps until convergence. In our formulation, however, the proposed objective function (3) is different, in that the sum-squares term is summed over the y-space instead of the $\mathbf{x}$-space.

One potential drawback of our formulation is that, given a fixed partition, there may not be any unique solution to the reduced points $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. The nonuniqueness is not a serious issue, so we will ignore it for the moment. We will see that this problem will disappear when we regularize our objective function using the k-means criterion, which is discussed later in Section 3.

2.2 | Heuristic algorithm

Clustering in the Euclidean space, that is, finding the *global* minimizer of the objective function in (5), is known to be an NP-hard problem [1]. Finding the globally optimal solution for our data reduction formulation (3) is even harder, because the objective function is related to the x's through an unknown nonlinear mapping. Since the purpose behind data reduction is to save time for fitting computationally expensive models to big data, the data reduction procedure should be computationally efficient. We propose next a simple algorithm which *approximates* the solution of (3) for efficient data reduction.

We illustrate our method using a simple one-dimensional example. Suppose $N = 3000$ data points (the "big data") are generated from the model $Y_i = \phi(X_i; 0.4, 0.01) + \epsilon_i$, where $i = 1, \ldots, N$, X_i's are equally spaced points from 0 to 1, $\phi(x; 0.4, 0.01)$ is the normal density with mean 0.4 and standard deviation 0.01, and $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$ with $\sigma = 0.1$. This is shown in Figure 2 as black circles after standardizing both x and y. Our aim is to reduce this big data to $n = 30$ reduced points. The reduced points from the "unsupervised" k-means clustering procedure (5) are shown in the same figure as red crosses. Here, the response value for each reduced point is obtained by averaging the response values of the big data within its respective cluster. We can see that the $n = 30$ points are almost uniformly sampled on the input domain [0, 1], and therefore misses the important region in the data space around $x = 0.4$.

Now consider the following strategy for sequentially optimizing the proposed data reduction criterion (3). We first start off with two points, say obtained using "unsupervised" k-means clustering on the $\mathbf{x}$-space. This is shown on the first top panel of Figure 3. The loss computed for the two corresponding clusters (see (2)) are $L_1 = 2888.4$ and $L_2 = 0.6$. Since the first cluster (left) has larger loss, we divide that cluster into two, by performing k-means

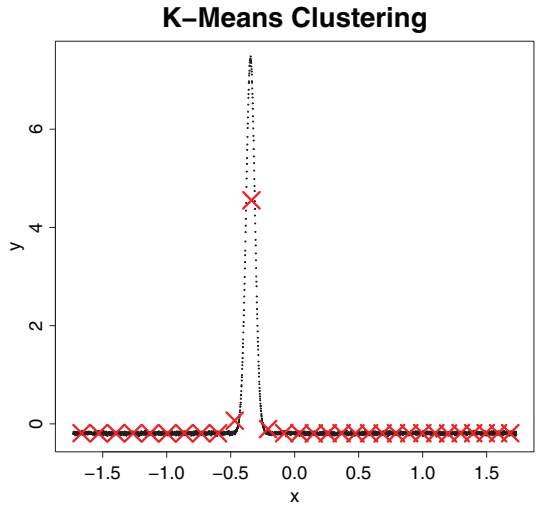

FIGURE 2 One-dimensional example. Big data are shown as black circles and the compressed data using k-means clustering is shown as red crosses

clustering on the points within the first cluster. This then gives two new cluster centers, as shown in the second top panel of Figure 3. Note that the two new points replace the cluster center from the original cluster. Finally, we compute the new Voronoi regions (see (1)) with respect to the three points and continue the procedure to obtain the next reduced point. This last step is shown on the bottom left panel of the figure.

The three steps above can then be iterated sequentially to generate the compressed dataset. In particular, at the mth step, we will identify the worst cluster as the one with highest loss, that is,

$$i^* = \underset{i=1:m}{\text{argmax}} L_i,$$

where L_i is computed using (2) for $i = 1, \ldots, m$. Then we divide the cluster I_{i^*} into two by using k-means clustering. The two new cluster centers will then replace the old cluster center $\mathbf{x}_{i^*}$. Finally, the Voronoi regions (1) with respect to the $m + 1$ points are then computed. The computation of Voronoi regions can be done very fast even for big datasets using kd-tree based algorithms, for example, using the implementation in the R package `FNN` [4]. The procedure is then continued until the desired number of points is selected. The bottom right panel of Figure 3 shows the results of the final step (Step 29). We can see that the $n = 30$ points obtained by this greedy algorithm nicely captures the input–output relationship, especially within the high variability region around $x = 0.4$.

An advantage of this sequential construction algorithm is that we can monitor the performance of data reduction, which can be used as a stopping rule for the procedure. We can compute, say, the root-mean squared prediction error

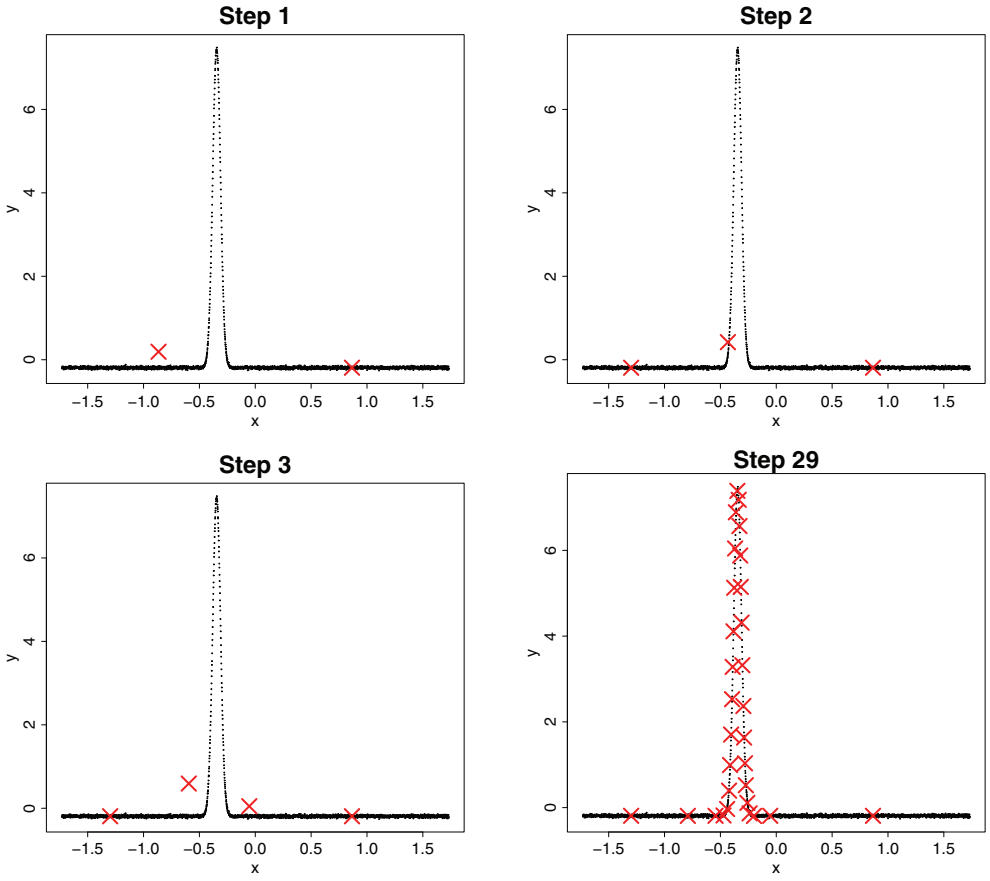

FIGURE 3 Step-by-step illustration of the proposed method in the one-dimensional example

(RMSE):

$$\text{RMSE}_n = \sqrt{\sum_{i=1}^{n} L_i / N},$$

which is shown on the left panel of Figure 4. This measure, of course, decreases to 0 as n increases to N. So we may choose an n where the curve stabilizes. Another option is to use an R^2 or adjusted-R^2 measure as in regression analysis:

$$R_n^2 = 1 - \frac{\sum_{i=1}^{n} L_i}{\sum_{j=1}^{N} (Y_j - \overline{Y})^2} \quad \text{and}$$

$$\text{adj}R_n^2 = 1 - \frac{\sum_{i=1}^{N} L_i/(N-n)}{\sum_{j=1}^{N} (Y_j - \overline{Y})^2/(N-1)},$$

where $\overline{Y}$ is the mean response in the big data. The right panel of Figure 4 shows the adjusted R^2 measure for this example. We see that, while the adjusted R^2 seems to improve (i.e., increase) as n increases, unlike the RMSE, it will start decreasing after a large enough choice of n (here, around $n = 100$). This provides a useful heuristic for stopping the sequential data reduction procedure.

In this example, the final R^2 for the "unsupervised" k-means clustering (i.e., on the $\mathbf{x}$-space) is 81.4%, whereas with the proposed algorithm, the R^2 increases to 99.75%, which is a substantial improvement. Of course, the greedy procedure above can be improved through more careful optimization. For example, at each step, we can try to find the two new cluster centers via nonlinear optimization. However, even though we consider only two points (i.e., a binary split of the worst cluster), it is still an optimization problem in $2p$ dimensions, which can be very time-consuming to solve using standard nonlinear optimization methods. Another approach might be to compute several choices of cluster centers which may be good approximations of the optimal solution, then choose the best centers. In our implementation, we used another solution obtained by clustering in the y-space to get the new points. Then, at each step we can make a choice between the two centers obtained by clustering in the x-space and those obtained using clustering in the y-space. This is still very fast to compute. The details are given in Algorithm 1. Using this modified approach, the R^2 becomes 99.79%, which is a slight improvement to what we got before.

Now let us examine the effect of noise on the `supercompress` procedure. Figure 5 shows the selected points as σ in the noise term increases. We can see that the performance of the proposed method deteriorates as the noise in the data increases, which is not too unexpected.

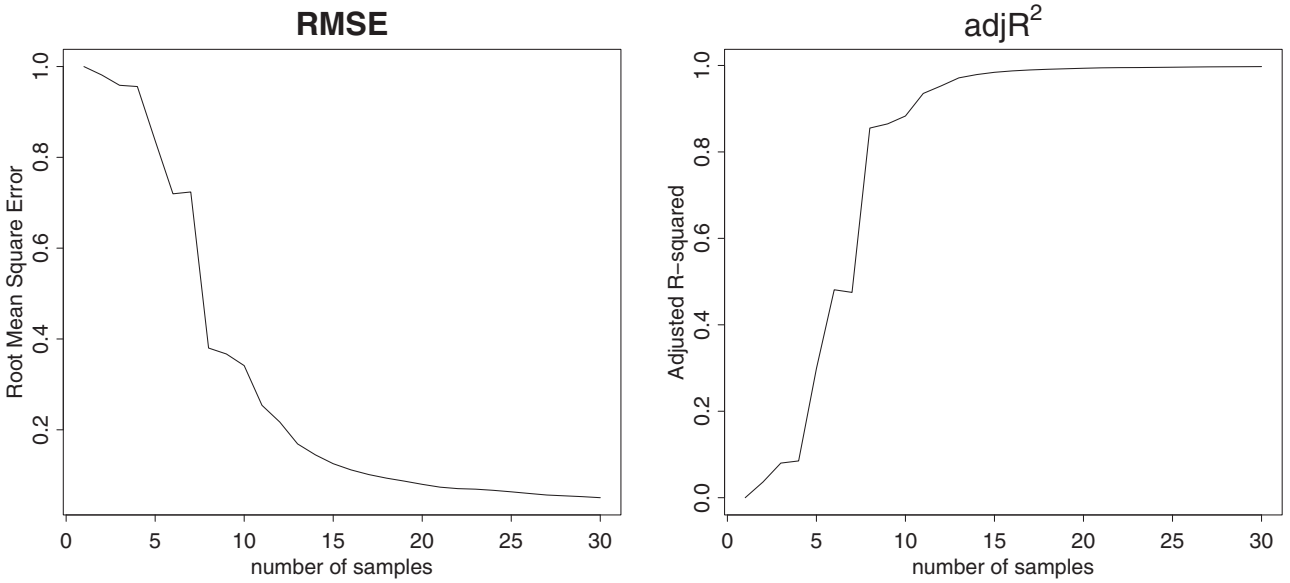

FIGURE 4 Monitoring the performance of the `supercompress` algorithm using root mean squared error and adjusted R^2 in the one-dimensional example

Algorithm 1. `supercompress`$(n, \{(\mathbf{X}_j, Y_j)\}_{j=1}^N)$: Supervised data reduction via clustering.

Return: reduced data points $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$;

- Perform k-means clustering using two clusters on input features of the big data $\{\boldsymbol{X}_j\}_{j=1}^m$, yielding two cluster centers in $\boldsymbol{x}$-space $\{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ and partitions $\{I_1, I_2\}$;
- Initialize cluster centers $\mathcal{D} = \{\boldsymbol{x}_1, \boldsymbol{x}_2\}$ and partitions $\mathcal{P} = \{I_1, I_2\}$;
- Compute the loss (2) for each partition in $\mathcal{P}$;

for $m = 1, \cdots, n-2$ **do**

- Find the cluster with highest loss $i^* = \text{argmax}_{i=1,\cdots,m+1} L_i$;
- **Split 1**: Perform k-means clustering using two clusters on input features in cluster i^*: $\{\boldsymbol{X}_j\}_{j \in I_{i^*}}$, yielding two cluster centers in $\boldsymbol{x}$-space $\{\boldsymbol{x}', \tilde{\boldsymbol{x}}'\}$;
- **Split 2**: Perform k-means clustering using two clusters on the response in cluster i^*: $\{y_j\}_{j \in I_{i^*}}$, yielding a partition in the y-space. Let $\{\boldsymbol{x}^*, \tilde{\boldsymbol{x}}^*\}$ be the cluster means in $\boldsymbol{x}$-space for this partition;
- Compute the loss (2) for the two split choices, and choose the cluster centers $(\boldsymbol{x}, \tilde{\boldsymbol{x}})$ which yield smaller loss;
- Remove from $\mathcal{D}$ the old center $\boldsymbol{x}_{i^*}$ and add new centers $\{\boldsymbol{x}^*, \tilde{\boldsymbol{x}}^*\}$;
- Update the partitions $\mathcal{P} = \{I_i\}_{i=1}^{m+2}$ from centers $\mathcal{D}$;

end

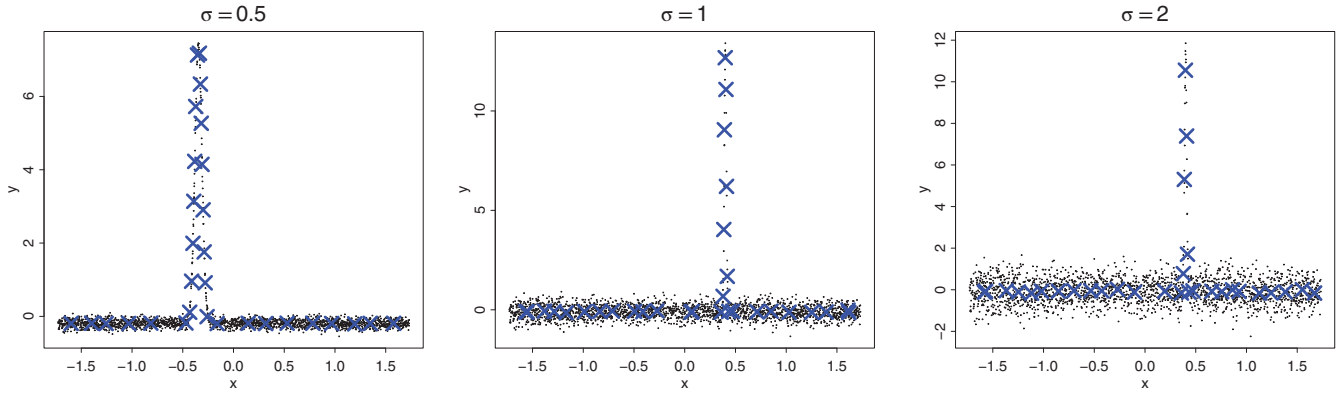

FIGURE 5 Performance of the `supercompress` procedure as the noise in the data increases

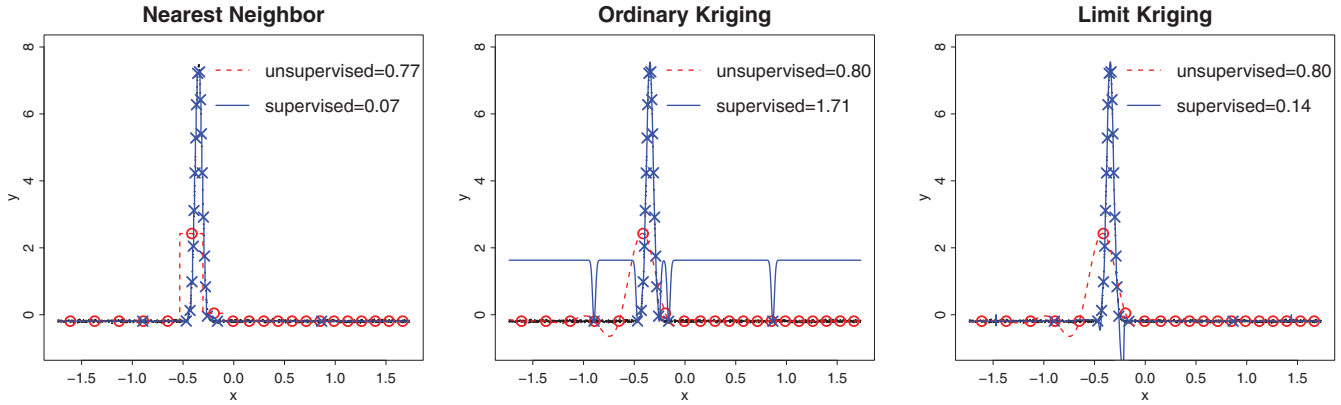

FIGURE 6 Performance of different modeling methods using the reduced dataset created by unsupervised k-means (red-dashed lines) and the proposed `supercompress` procedure (blue-solid lines). The root mean squared prediction errors of the two methods are shown in the legend

However, it seems to still perform better than the traditional unsupervised k-means (see Figure 2).

3 | ROBUSTIFYING THE SUPERCOMPRESS PROCEDURE

To investigate the robustness of the `supercompress` procedure to different modeling choices, consider again the simple one-dimensional example of the previous section. Let $N = 10{,}000$ and that we want to reduce the data to $n = 20$. Consider the following three nonparametric regression models: (i) 1-nearest neighbor, (ii) ordinary kriging [19], and (iii) limit kriging [12]. Two sets of compressed data are generated, the first using the traditional k-means (on only input features), and the second using the proposed `supercompress` procedure. The compressed points and the three fitted models are plotted in Figure 6. The RMSE values of these fits are shown in the legend of each plot. The ordinary kriging and limit kriging were fitted using a Gaussian correlation function with parameters estimated using the R package `mlegp` [8].

We can see that there is more than a 10-fold improvement in the nearest neighbor prediction when using the proposed `supercompress` method. However, the predictions became worse with ordinary kriging because the fitted predictor behaves strangely in the regions where the data are sparse. One reason for this is that the ordinary kriging model assumes stationarity, which makes the predictor "regress" back to the mean in regions where there is no data. Limit kriging, proposed in Joseph [12], is a simple modification to the ordinary kriging model to avoid this mean-regressive feature, and because of this, `supercompress` still provides a noticeable improvement over its unsupervised counterpart. In fact, for limit kriging, the fitted predictor converges to the nearest neighbor predictor, which explains why the proposed method is performing better for such a model.

The foregoing results are a bit disappointing, because our aim was to develop a data reduction method which can perform well on a wide variety of modeling methods. Thus, our hope that the nearest neighbor method would be robust to all modeling choices appears to not be true! We introduce next a modification of `supercompress` which can yield improved robustness.

One possible strategy to robustify the method is to take a convex combination of the objective functions in the unsupervised and supervised procedures. Consider the new loss function:

$$\widetilde{L} = \lambda \sum_{i=1}^{n} \sum_{j \in I_i} \|\mathbf{X}_j - \mathbf{x}_i\|^2 + (1-\lambda) \sum_{i=1}^{n} \sum_{j \in I_i} (Y_j - y_i)^2, \qquad (6)$$

where $\lambda \in [0, 1]$ is a weight parameter quantifying the trade-off the unsupervised and supervised clustering criteria. When $\lambda = 0$, the formulation (6) reduces to the earlier `supercompress` criterion (3), and when $\lambda = 1$, this reduces to traditional (unsupervised) k-means criterion (5). One can view this modified criterion as a trade-off between a fully *supervised* reduction strategy ($\lambda = 0$), which fully incorporates response information for data reduction, and a fully *robust* reduction strategy ($\lambda = 1$), which reduces the data using just input feature information. With a choice of $\lambda \in (0, 1)$, the reduced points $\{\mathbf{x}_i\}_{i=1}^{n}$ minimizing (6) would (conceptually) integrate response information in a model-robust way.

Of course, the choice of parameter λ in (6) plays a critical role in the quality of the reduced points. One option would be to tune λ via cross-validation, but this would incur multiple passes of the big data and would be very

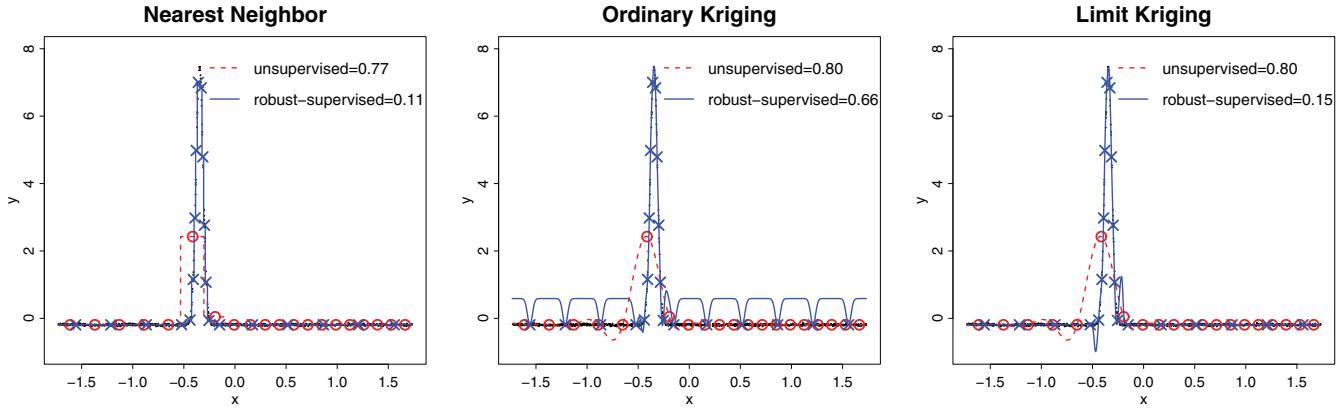

FIGURE 7 Performance of different modeling methods using the reduced dataset created by unsupervised k-means (red-dashed lines) and the `rsupercompress` procedure (blue-solid lines). The root mean square prediction errors of the two methods are shown in the legend

computationally expensive. Instead, we found that the default choice of $\lambda = 1/(1+p)$ (where p is the number of input variables) works well in practice. One reason for this is because, with this choice of λ, the objective function (6) becomes proportional to

$$\sum_{i=1}^{n}\sum_{j\in I_i}\frac{1}{p}\|\mathbf{X}_j - \mathbf{x}_i\|^2 + \sum_{i=1}^{n}\sum_{j\in I_i}(Y_j - y_i)^2.$$

This suggests that, with $\lambda = 1/(1+p)$, equal importance is placed on the clustering problems in $\mathbf{x}$-space and y-space, thus providing a balanced trade-off between supervision and robustness.

To optimize the new loss function (6), we can in fact use the same heuristic algorithm from Section 2 with a slight modification. The only change needed is that the Y_j's in the previous algorithm are changed to $\widetilde{Y}_j = \sqrt{(1-\lambda)/\lambda}Y_j$, and the loss functions L_i in (2) are changed to

$$\widetilde{L}_i = \sum_{j\in I_i}\{\|\mathbf{X}_j - \mathbf{x}_i\|^2 + (\widetilde{Y}_j - y_i)^2\}.$$

Furthermore, when $\lambda = 1$, all the points can be generated using the traditional k-means algorithm, and we do not need to perform any of the iterative steps of Algorithm 1. Therefore, we initialize the algorithm with $\lceil \lambda n \rceil$ points obtained using k-means. We will call this robust modification of supervised compression `rsupercompress` from here on.

The reduced dataset constructed by the new `rsupercompress` method with $\lambda = 1/(1+p)$ and the three fitted models are shown in Figure 7. We can see that there are now more points in the left and right tails of the function, so the fitted models are uniformly performing better than those fitted using the unsupervised k-means dataset. Hence, for this problem, the proposed modified supervised approach indeed provides effective and robust data compression, which is as desired. We can still see the undesirable "reversion to the mean" issue in the ordinary kriging predictor, but the deviation from the true function is reduced due to the presence of more points in the tails.

4 | EXAMPLES

4.1 | A test example

Consider the Michaelwicz function in p dimensions given by

$$f(\mathbf{x}) = -\sum_{k=1}^{p}\sin(\pi x_k)\sin^{20}(k\pi x_k^2),$$

where $\mathbf{x}\in[0,1]^p$. First consider two dimensions, that is, $p = 2$. We generate $N = 20{,}000$ points randomly from $U(0,1)^2$ and $Y_i = f(\mathbf{X}_i) + \epsilon_i$, where $\epsilon_i \overset{\text{iid}}{\sim} N(0, 0.0001^2)$. After standardizing, this dataset is compressed to $n = 200$ points using the traditional unsupervised k-means, the proposed supervised compression `supercompress` ($\lambda = 0$) and robust supervised compression `rsupercompress` ($\lambda = 1/(p+1)$). The compressed data points are shown in Figure 8, along with the image of the function and Voronoi regions (1).

We can see that the unsupervised k-means spreads the points uniformly in the space, whereas the proposed `supercompress` places more points in the regions where the function varies a lot. On the other hand, `rsupercompress` is a compromise between the two, which as demonstrated in the previous section will be more robust to different modeling choices. We can also see that the Voronoi regions are roughly the same size in the unsupervised k-means, whereas they are quite different with the supervised procedures. Bigger Voronoi regions appear where the function varies less and smaller Voronoi regions appear where the function varies more. This makes

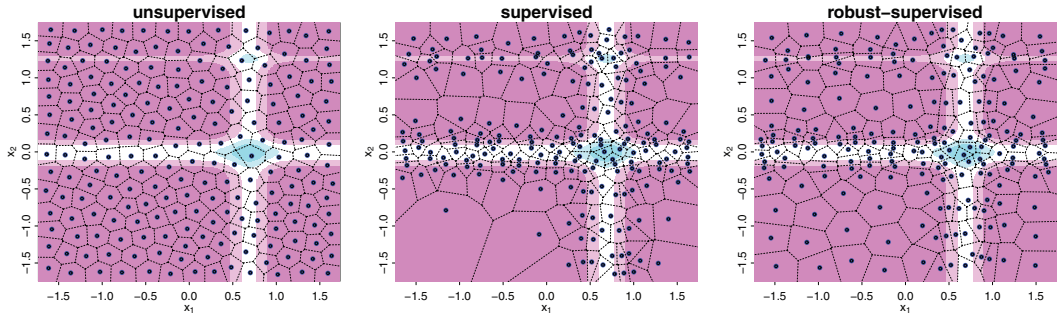

FIGURE 8 Two-dimensional example using Michaelwicz function

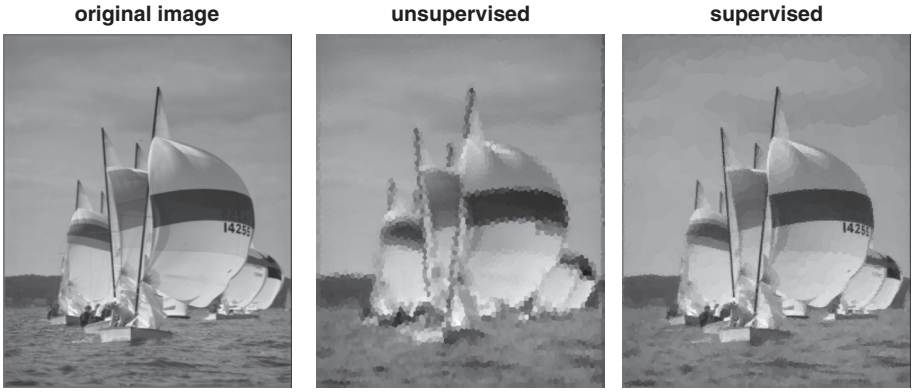

FIGURE 9 Comparison of unsupervised (middle) and supervised (right) procedures on the 5% compression of the original boat image (left)

sense because the function is approximated by a constant within each Voronoi region.

Now consider a higher dimensional example. Let $p = 5$ in the Michaelwicz function. We generate $N = 50{,}000$ points as before. Suppose we want to reduce this to $n = 500$ points. Both the unsupervised and supervised procedures did not perform well here. For comparison, the R^2 value of the unsupervised k-means is about 19%, whereas for the supervised version it is about 25%. This indicates that both requires much larger n in high dimensions to accurately represent the big data. If we increase the n to 5000, the R^2 for the unsupervised and supervised procedures becomes 43% and 57%, respectively. We can see that the gain in R^2 with the supervised version is bigger because the data points are selected more carefully from regions where it matters the most. On the other hand, the `supercompress` procedure took 645 s to generate the 5000 points as opposed to only 12 s for the k-means on a 3.2 GHz computer. This is the price we need to pay for the improved performance of the `supercompress` procedure.

4.2 | An image compression example

We will use an image compression example to compare the performance of supervised and unsupervised k-means. The left panel of Figure 9 shows the gray scale image of a boat taken from the `R` package `imager` [2], which is a compressed version of a picture taken from http://r0k.us/graphics/kodak/kodim09.html. This image has 256×384 pixels and thus, Y is a vector of length 98,304. Suppose we wish to compress it to 5% of its original size, that is, $n = 4915$.

The middle panel of Figure 9 shows the compressed image based on unsupervised k-means and the last panel shows the compressed image based on `supercompress`. The supervised procedure clearly does a better job in the image compression than the unsupervised k-means. This is because the supervised procedure picks out the regions with large or sudden changes, which is helpful in reconstructing the image. It is quite striking to see that the number "14255" in the sail of the boat is readable even with a 5% compression. However, the clouds in the picture are not well captured, but this may not be the interesting part of the picture. Overall, the `supercompress` procedure seems to automatically pick up the salient features of the picture that seem to agree with our intuition.

5 | SEMIPARAMETRIC DATA REDUCTION

Fitting a parametric model to data can be thought of as a way to compress data. For example, suppose the big data were indeed generated from a linear regression model.

Then, by fitting a perfectly specified linear regression model to this data, we can estimate the $p+1$ regression parameters and throw away the big data. The $p+1$ parameter estimates can then be viewed as the compressed data. The catch here is, of course, when the data are *not* generated from a linear regression model, such a "parametric" compression may result in a poor summary of the big data. What we have proposed so far is a "nonparametric" way to compress data, one which is robust to a wide range of data generating mechanisms. We explore next a compromise between these two extremes—a *semiparametric* approach—which aims to combine the best of both worlds.

Suppose the data is generated from the following model

$$Y_i = \mu(\mathbf{X}_i; \boldsymbol{\beta}) + \epsilon_i, \quad i = 1, \ldots, n, \tag{7}$$

where $\mu(\mathbf{X}; \boldsymbol{\beta})$ is a model parameterized by the unknown parameters $\boldsymbol{\beta}$. In the absence of any prior information about the underlying relationship, we could use the following parametric model

$$\mu(\mathbf{X}; \beta) = \beta_0 + \sum_{k=1}^{p} \beta_k f_k(X_k),$$

where $f_k(\cdot)$'s are some known data-independent basis functions. The unknown parameters β can be estimated from the big data using, for example, maximum likelihood estimation. Let $\widehat{\beta}_k$ denote the estimate of parameter β_k, $k = 1, \ldots, p$. We can then compute the residuals

$$Z_i = Y_i - \mu(\mathbf{X}_i; \widehat{\boldsymbol{\beta}}), \quad i = 1, \ldots, n.$$

The data compression procedure of the previous sections can now be applied to $\{(\mathbf{X}_i, Z_i)\}_{i=1}^{N}$ to obtain the reduced data $\{(\mathbf{x}_i, z_i)\}_{i=1}^{n}$. However, in addition to this compressed data, we also need to store the estimates of $\boldsymbol{\beta}$ and the basis functions $f_1(\cdot), \ldots, f_p(\cdot)$. Note that it is important to choose *data-independent* basis functions, otherwise the data (or "knots") that are needed to define the basis functions will also need to stored, and will become part of the compressed data.

Consider again the one-dimensional example in Section 3. Suppose we take $\mu(X; \beta) = \beta_0$. From the big data, we obtain the estimate $\widehat{\beta}_0 = 0$ (because the data have been standardized). The reduced data using the unsupervised k-means and `supercompress` procedures are shown in Figure 10, which is the same as before. Now, instead of fitting an ordinary kriging model to the reduced data, we fit a zero-mean simple kriging model to the residuals. The final predictions can be obtained by adding back the $\widehat{\beta}_0$ to the predictions from the simple kriging model. These are also plotted in Figure 10. We can see that this semiparametric approach yields a substantial improvement in predictive performance even with a simple constant parametric model, compared to the earlier nonparametric approaches (see the middle panels of Figures 6 and 7). This is because the predictions now regress back to the "true" mean of the data, and not to the estimated mean from the compressed data.

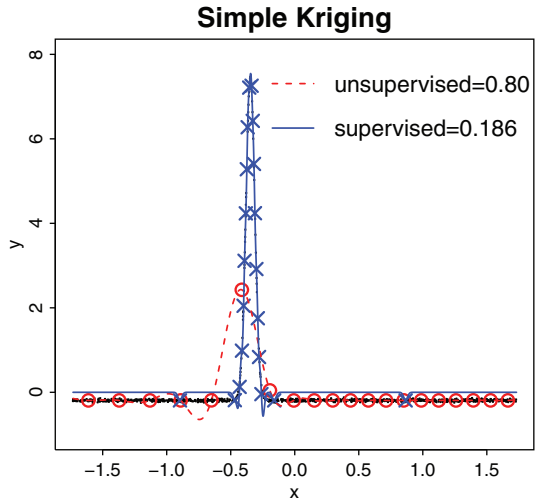

FIGURE 10 Comparison of simple kriging model fitted to the compressed data from unsupervised k-means and `supercompress` procedures with mean estimated from the big data

6 | APPLICATION TO TAXICAB SCHEDULING

Finally, we illustrate the usefulness of the proposed reduction method in a real-world application on predicting the ride duration of taxi trips in New York City. A common challenge faced by taxi companies is real-time *taxi scheduling*: the efficient assignment of taxis to passengers to ensure smooth and timely service. One key part of this scheduling involves predicting the duration of a current trip, which allows one to anticipate when the cab will be available for future trips. We consider here the predictive modeling problem for ride duration of a taxi trip from individual trip attributes.

For model training, we adopt a dataset released by the New York Taxi and Limosine Commission in 2016, which was made publicly available for a popular Kaggle predictive modeling competition in 2017 [14], with over 11,000 submission entries. Since then, there has been much work on augmenting this dataset with additional input features from real-time routing engines. In particular, we use the augmented dataset provided in Benmeida [3], which includes additional features extracted from the Open Source Routing System [10], a high-performance

TABLE 1 Input features (units) for training the taxicab predictive modeling problem

Input features	
Trip distance (m)	Secondary proportion (% of total distance)
Motorway proportion (% of total distance)	Tertiary proportion (% of total distance)
Trunk proportion (% of total distance)	Unclassified proportion (% of total distance)
Primary proportion (% of total distance)	Residential proportion (% of total distance)

Note: The response variable to predict is trip duration (in seconds).

TABLE 2 Test RMSE of a 1-NN predictor trained using n reduced points from random subsampling, k-means, `supercompress`, and `rsupercompress`

1-NN predictor			
	$n = 250$	$n = 500$	$n = 2000$
K-means (NP)	81.60	66.00	71.37
K-means (SP)	32.52	31.73	30.46
`supercompress` (NP)	40.85	37.73	34.05
`supercompress` (SP)	29.56	29.50	28.29
`rsupercompress` (NP)	41.11	38.30	33.98
`rsupercompress` (SP)	29.73	29.47	28.83

Note: All reduction methods are performed both nonparametrically (NP) and semiparametrically (SP) via an l_1-regularized linear model with fitted Box–Cox transformation (BoxCox-l_1). The test RMSE for BoxCox-l_1 is 37.92.

routing engine for shortest paths in road networks. There are a total of $p = 8$ input features in this dataset (details found in Table 1), with the response variable for prediction being the trip duration (in seconds). This augmented dataset is then randomly split into $N = 1{,}458{,}643$ entries for training, and the remaining $N_{\text{te}} = 625{,}134$ entries for testing.

Clearly, with over 1,000,000 training data points, fitting an efficient predictive model for real-time scheduling is a daunting task. One solution (among many) is to train the predictive model on a smaller subset of the data. To this end, we compare the proposed supervised procedures (`supercompress` and `rsupercompress`) with k-means clustering. Each method is performed with and without the initial parametric reduction outlined in Section 5, via an l_1-regularized linear regression model with a fitted Box–Cox transformation on the response variable (we refer to this as BoxCox-l_1 from here on). Prediction performance is then evaluated on the holdout test set.

Consider first the 1-nearest-neighbor (1-NN) predictor, on which the proposed compression method is based. Table 2 shows the test RMSE of the considered data compression methods using the 1-NN predictor, for reduced sample sizes of $n = 250$, 500, and 2000. The test RMSE for a parametric BoxCox-l_1 fit is also provided as a benchmark. We see that the two proposed supervised methods provide a noticeable lower test errors compared to the unsupervised k-means clustering, which shows that information from the output variable can indeed improve predictive performance under 1-NN. For each method, the semiparametric approach of performing compression after an initial BoxCox-l_1 fit yields much lower test errors than its nonparametric counterpart, which suggests a linear trend in the data.

Consider next the performance of these methods for two other widely used predictive models in the literature: the Gaussian process (GP) model [19] and the neural network (NNet) model [5]. The GP model is fit with an anisotropic Matérn correlation function. Two types of NNets are implemented here: a simple NNet using one hidden layer with one node, and a more complex NNet using two hidden layers with four and one nodes, respectively. Both NNet models use a tanh activation function. Table 3 shows the test RMSE of the considered methods using $n = 500$ reduced points. We see that `rsupercompress` (with semiparametric compression) provides uniformly low test errors over all model choices, whereas all other methods yield high test errors for at least one of the considered models. This shows that the proposed supervised data compression approach, with the robustness and semiparametric modifications, provides both better and more robust predictive performance over the unsupervised compression method.

	1-NN	GP	1-Layer NNet	2-Layer NNet
K-means (NP)	66.00	73.29	924.27	921.57
K-means (SP)	31.73	453.86	490.38	918.70
`supercompress` (NP)	37.73	627.03	687.77	687.72
`supercompress` (SP)	29.50	232.52	37.90	37.90
`rsupercompress` (NP)	38.30	92.62	507.18	711.55
`rsupercompress` (SP)	29.47	58.35	37.90	37.92

TABLE 3 Test RMSE of different predictors trained using $n = 500$ reduced points from k-means, `supercompress`, and `rsupercompress`

Note: All reduction methods are performed both non-parametrically (NP) and semi-parametrically (SP) via an l_1-regularized linear model with fitted Box–Cox transformation (BoxCox-l_1). The test RMSE for BoxCox-l_1 is 37.92.

7 | CONCLUSION

This article discusses the problem of reducing/compressing the size of big data using the information in the output. We first proposed a solution to this problem by choosing a subset of the data or more accurately some representative points that will give a good fit to the big data if we were to use a 1-nearest neighbor prediction approach. The distinguishing feature of this approach is that the representative points are chosen from regions where it matters the most for modeling purposes. We also proposed a fast supervised k-means-based algorithm for finding the solution. Although the proposed approach seems to work well, the solution is found to be not robust to all possible input–output modeling choices. We then discussed on how the approach can be modified to improve robustness against possible modeling choices. We also discussed further improvements to the method by using a semiparametric data compression approach. We demonstrated through many examples that the proposed supervised data compression method can outperform the other data compression methods that do not use information in the output.

In our discussion, we have focused on continuous inputs and outputs. Extending the proposed method to deal with categorical inputs and outputs is important for regression as well as for classification problems. One simple idea for incorporating categorical inputs is to partition the data with respect to the levels of the categorical input and then perform data compression independently within each partition. However, doing the same thing for categorical outputs may not be the best approach, as evidenced by the optimal design-based data reduction works on logistic regression [20, 22]. This seems like a promising topic for future research.

ACKNOWLEDGMENTS

This research is supported by a U.S. National Science Foundation (NSF) grant CMMI-1921646, and by an NSF CSSI Frameworks grant 2004571.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available.

ORCID

V. Roshan Joseph https://orcid.org/0000-0002-9430-5301

Simon Mak https://orcid.org/0000-0002-5693-7076

REFERENCES

1. D. Aloise, A. Deshpande, P. Hansen, and P. Popat, *NP-hardness of Euclidean sum-of-squares clustering*, Mach. Learn. 75 (2009), 245–248.
2. S. Barthelme, D. Tschumperle, J. Wijffels, H. E. Assemlal, and S. Ochi, *imager: Image processing library based on "CImg"*, R 0.42.3, 2020, available at https://CRAN.R-project.org/package=imager.
3. M. Benmeida, *NYC Taxi Trip Durations: Data augmentation using OSRM*, 2017, accessed October 17, 2020.
4. A. Beygelzimer, S. Kakadet, J. Langford, S. Arya, D. Mount, and S. Li, *FNN: Fast nearest neighbor search algorithms and applications*, R 1.1.3, 2019, available at https://CRAN.R-project.org/package=FNN.
5. C. M. Bishop, *Neural networks for pattern recognition*, New York, NY, Oxford University Press, 1995.
6. L. Bottou, *Large-scale machine learning with stochastic gradient descent*, in *Proceedings of COMPSTAT 2010*, Heidelberg, Germany, Springer, 2010, 177–186.
7. J. Conway and N. Sloane, *Voronoi regions of lattices, second moments of polytopes, and quantization*, IEEE Trans. Inf. Theory 28 (1982), 211–226.
8. G. M. Dancik, *mlegp: Maximum likelihood estimates of Gaussian processes*, R 3.1.7, 2018, available at https://CRAN.R-project.org/package=mlegp.
9. P. Drineas, M. Magdon-Ismail, M. Mahoney, and D. Woodruff, *Faster approximation of matrix coherence and statistical leverage*, J. Mach. Learn. Res. 13 (2012), 3475–3506.
10. S. Huber and C. Rust, *Calculate travel time and distance with OpenStreetMap data using the open source routing machine (OSRM)*, Stata J. 16 (2016), 416–423.
11. J. Huggins, T. Campbell, and T. Broderick, *Coresets for scalable Bayesian logistic regression*, in *Advances in neural information*

processing systems, Red Hook, NY, Curran Associates, 2016, 4080–4088.
12. V. R. Joseph, *Limit kriging*, Technometrics 48 (2006), 458–466.
13. V. R. Joseph and A. Vakayil, *Split: An optimal method for data splitting*, arXiv preprint arXiv:2012.10945, 2020.
14. Kaggle, *New York City Taxi Trip Duration*, 2017, accessed October 17, 2020.
15. S. P. Llyod, *Least squares quantization in PCM's*, Bell Telephone Laboratories Paper, 1957.
16. P. Ma, M. Mahoney, and B. Yu, *A statistical perspective on algorithmic leveraging*, J. Mach. Learn. Res. 16 (2015), 861–911.
17. S. Mak and V. R. Joseph, *Support points*, Ann. Statist. 46 (2018), 2562–2592.
18. J. Max, *Quantizing for minimum distortion*, IRE Trans. Inf. Theory IT-6 (1960), 7–12.
19. T. J. Santner, B. J. Williams, and W. I. Notz, *The design and analysis of computer experiments*, Springer, New York, 2003.
20. H. Wang, *More efficient estimation for logistic regression with optimal subsamples*, J. Mach. Learn. Res. 20 (2019), 1–59.
21. H. Wang, M. Yang, and J. Stufken, *Information-based optimal subdata selection for big data linear regression*, J. Amer. Statist. Assoc. 114 (2019), 393–405.
22. H. Wang, R. Zhu, and P. Ma, *Optimal subsampling for large sample logistic regression*, J. Amer. Statist. Assoc. 113 (2018), 829–844.

How to cite this article: Joseph VR, Mak S. Supervised compression of big data. *Stat Anal Data Min: The ASA Data Sci Journal*. 2021;14:217–229. https://doi.org/10.1002/sam.11508