Filling Survey Missing Data With Generative Adversarial Imputation Networks

Cesar Y. Villarreal Guzman

22 December 2020

Abstract

In this paper we propose an alternative method for dealing with missing values on survey data sets. The method leverages two known techniques, categorical encoding and generative adversarial imputation networks. We test this approach on the "Kaggle Data Science Survey" and the "American Community Survey", we experiment with different proportions of missing values and sample sizes and find the technique to yield accurate imputations.

Keywords: Generative Adversarial Networks; Imputation Algorithms; Surveys

1. Introduction

Survey response rates have seen a decline in recent years and more often than not we see incomplete survey data sets. Often this is because survey designers give the option to skip a question, or simply a respondent decides to not finish the survey. In the literature this is known as data missing completely at random (MCAR) because in most cases there is no dependency on any of the variables. This pervasive problem has also been the cause for multiple solutions to emerge. An imputation algorithm, for example, can be used to estimate missing values based on data that was observed/measured. A substantial amount of research has been dedicated to developing imputation algorithms for medical data but it is also commonly used in image concealment, data compression, and counterfactual estimation.

Often imputation algorithms work very well when the data is continuous, and or contains a mixture of continuous and categorical responses, however it is common to only observe categorical and text responses in survey data sets. Text data is an almost impossible problem to solve because we can't just simply create an algorithm that will write an opinion on behalf of another person. There are both ethical and technical problems associated. Categorical responses on the other hand are simpler to use because having a finite amount of categories allows us to encoded the data. The most popular encoding technique is known in the statistics literature as dummy variable encoding or in the computer science and machine learning literature as one-hot encoding. This popular technique also comes with its limitations since a substantial amount of information is lost by turning variables into vectors of 0 and 1s.

Moreover this technique requires us to increase the dimensions of our data set which results in a loss of computational efficiency.

Hence, we address the problem of data imputation when the data set consists of only categorical variables, and in particular when the data comes from a survey. In this paper we propose an alternative method for data imputation in survey data which comprises of combining two known methods, categorical encoding and a state of the art imputation algorithm. Specifically, we encode categorical variables with a technique that is based on the weight of evidence (WOE) method and then use the imputation algorithm known as generative adversarial imputation networks (GAIN) to fill missing values.

The paper is divided in the following manner, first in section 2 we elaborate on generative adversarial imputation networks and how they are applied in this context by discussing the proposed encoding technique based on the weight of evidence method. In section 3 we discuss the experiments we conducted on the "*Kaggle Data Science Survey*" (Kaggle 2020)] and the "*American Community Survey*" (Steven Ruggles and Sobek 2020) to test the effectiveness of this method. In this section we comment on the surveys, the experiment and our empirical results. Finally, in the last section we comment on our results and the implications, how this method could be applied in practice, limitations and areas of future work.

2. Survey Generative Adversarial Imputation Networks

2.1 Generative Adversarial Imputation Networks

First to understand generative adversarial imputation networks (GAIN) we comment on the GAN framework. Generative adversarial nets (GANs) define a framework for estimating generative models via an adversarial process in which two models are trained: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G (Goodfellow et al. 2014). Commonly both the generator and discriminator are two separate neural networks.

Generative adversarial imputation networks (GAIN) is an imputation algorithm that generalized the idea of traditional GANs. The generator's goal is to accurately impute the missing data and the discriminator's goal is to distinguish between observed and imputed components. The discriminator is trained to minimize the classification loss (when classifying which components were observed and which have been imputed), and the generator is trained to maximize the discriminator's misclassification rate (Yoon, Jordon, and Schaar 2018). As with regular GANs both networks are trained in an adversarial process. In the following sections we give a brief explanation of how the GAIN framework is applied, we advice the reader to look at Yoon, Jordon, and Schaar (2018) for theoretical details and the training algorithms.

2.1.1 Data Imputation Problem

We beginning by outlining the problem to solve and introducing some notation. Consider the random variable $\mathbf{X} = (X_1, \ldots, X_d)$, called the data vector, which takes values in a

d-dimensional space V^d, and a random variable $\mathbf{M} = (M_1, \ldots, M_d)$, called the mask vector, taking values in $\{0,1\}^d$. For each $i \in \{1, \ldots, d\}$ we define a new space $\tilde{V} = V \cup \{NaN\}$, where the variable NaN represents a point not in any V_i which is an unobserved value. Let $\tilde{V}^d = \tilde{V}_1 \times \cdots \times \tilde{V}_d$ and define a new random variable $\tilde{\mathbf{X}} \in \tilde{V}^d$ in the following way

$$\tilde{\mathbf{X}} = \begin{cases} X_i & \text{if } M_i = 1 \\ NaN & \text{otherwise} \end{cases} \tag{1}$$

i.e. the random variable $\mathbf{M}$ indicates which entries of $\mathbf{X}$ are observed in $\tilde{\mathbf{X}}$. Suppose we have n i.i.d copies $\tilde{\mathbf{X}}$ denoted $\tilde{\mathbf{x}}^1, \ldots, \tilde{\mathbf{x}}^n$ then we define the data set $\mathcal{D} = \{(\tilde{\mathbf{x}}^i, \mathbf{m}^i)\}_{i=1}^n$ where each $\mathbf{m_i}$ indicates with a 0 the values missing in $\tilde{\mathbf{x}}^i$. The goal of data imputation is that of modeling $P(\mathbf{X}|\tilde{\mathbf{X}} = \tilde{\mathbf{x}}^i)$.

2.1.2 GAIN Methodology

Given data $\mathcal{D} = \{(\tilde{\mathbf{x}}^i, \mathbf{m}^i)\}_{i=1}^n$ as described above consider the function $G : \tilde{V}^d \times \{0,1\}^d \times [0,1]^d \to V^d$ called the generator which takes as input $\tilde{\mathbf{x}}^i$, $\mathbf{m}^i$ and a noise vector $\mathbf{z} \in [0,1]^d$ of the same dimension as $\tilde{\mathbf{x}}^i$. This noise vector is sampled independently of $\tilde{\mathbf{x}}^i$ and $\mathbf{m}^i$. We denote the vector of imputed values and the completed data vector respectively as

$$\bar{\mathbf{x}}^i = G(\tilde{\mathbf{x}}^i, \mathbf{m}^i, (\mathbf{1} - \mathbf{m}^i) \odot \mathbf{z}) \tag{2}$$

$$\hat{\mathbf{x}}^i = \mathbf{m}^i \odot \tilde{\mathbf{x}}^i + (\mathbf{1} - \mathbf{m}^i) \odot \bar{\mathbf{x}}^i \tag{3}$$

where $\mathbf{1}$ denotes a vector of 1s and $\odot$ represents element wise multiplication. A function $D : V^d \times \mathcal{H} \to [0,1]^d$, called the discriminator, takes as input completed vectors $\hat{\mathbf{x}}^i$ and has the objective of distinguishing which components are real (observed) and which are fake (imputed) - this amounts to predicting the mask vector, $\mathbf{m}^i$. In particular the j-th entry of $D(\hat{\mathbf{x}}^i, \mathbf{h})$ denotes the probability that the j-th entry of $\hat{\mathbf{x}}^i$ was observed. The vector $\mathbf{h}$ is what the authors of GAIN refer to as the hint mechanism which is a matrix that resembles the true mask $\mathbf{m}^i$ but has a number of differences. This hint mechanism as the name should suggest "helps" the discriminator D predict the true mask $\mathbf{m}$. Figure 1 was taken from the original GAIN paper and helps understand what was mentioned in this section more intuitively by displaying in a graphical way the architecture of the GAIN framework.

The objective is as follows: we train D to maximize the probability of correctly predicting $\mathbf{M}$ and G to minimize the probability of D predicting $\mathbf{M}$. Notice the training is adversarial which resembles the original GAN framework. We define the quantity $V(D, G)$ to be

$$V(G, D) = \mathbb{E}_{\tilde{\mathbf{X}}, \mathbf{M}, \mathbf{H}} \left[\mathbf{M}^T \log D(\hat{\mathbf{X}}, \mathbf{H}) + (\mathbf{1} - \mathbf{M})^T \log(\mathbf{1} - D(\hat{\mathbf{X}}, \mathbf{H})) \right] \tag{4}$$

where log is element wise and dependence on G is trough $\hat{\mathbf{X}}$. The objective can then be described in notation as

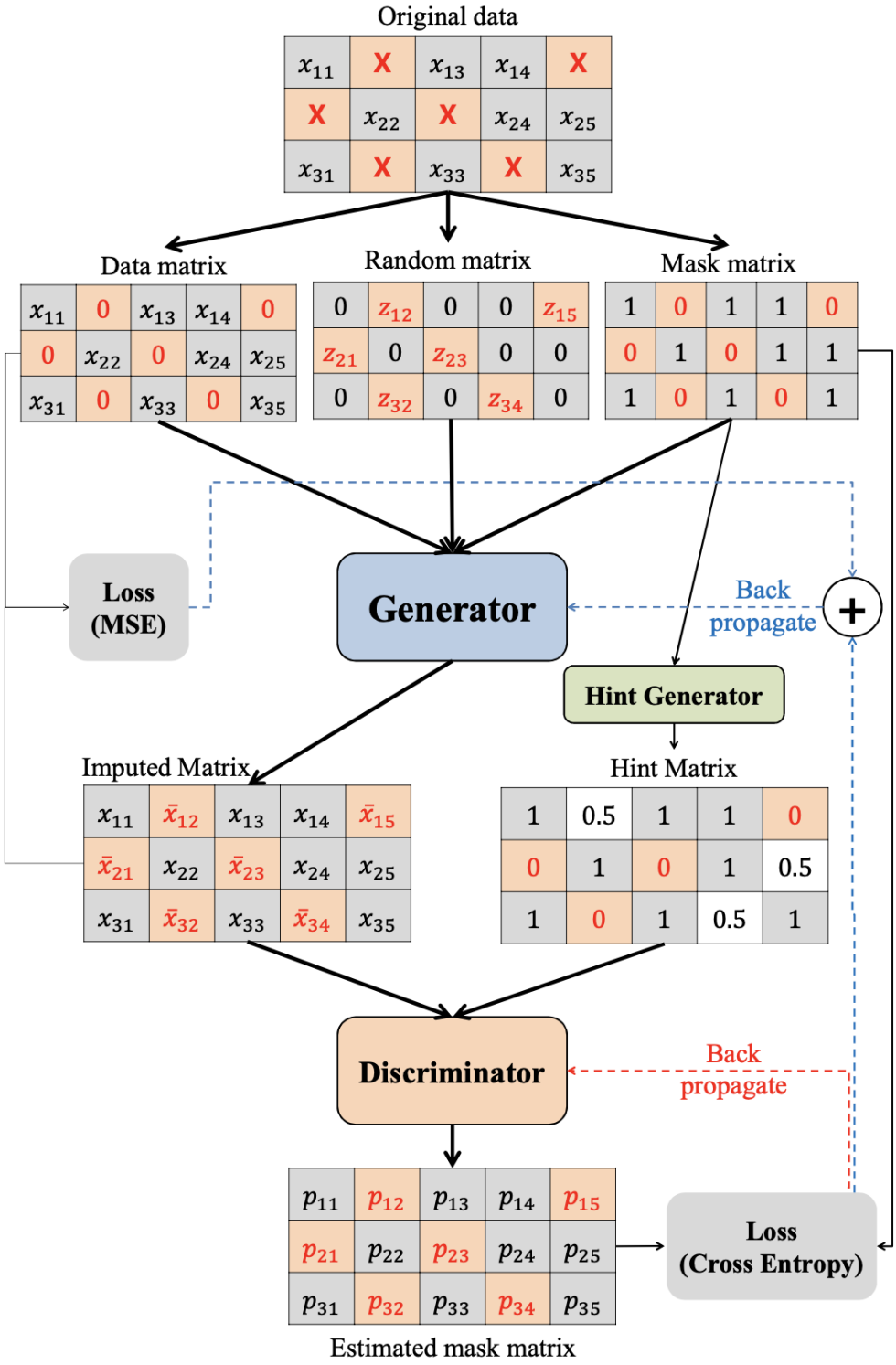

Figure 1: GAIN Architecture

$$\min_G \max_D V(D, G) \tag{5}$$

which is the known min-max problem introduced by the GAN framework. In practice the optimization problem we solve is as follows: let $\hat{\mathbf{M}} = D(\hat{\mathbf{X}}, \mathbf{H})$ then the optimization problem can be re-written to

$$\mathcal{L}(\mathbf{M}, \hat{\mathbf{M}}) = \sum_{j=1}^{d} M_j \log(\hat{M}_j) + (1 - M_j) \log(1 - \hat{M}_j) \tag{6}$$

$$\min_G \max_D \mathbb{E}\left[\mathcal{L}(\mathbf{M}, \hat{\mathbf{M}})\right] \tag{7}$$

As mentioned other theoretical details and the training algorithm should be consulted in the original paper by Yoon, Jordon, and Schaar (2018).

2.2 Variable Encoding

In practice there are several techniques to encode categorical variables into a continuous or numerical variable. The weight of evidence is one such technique which evolved from the logistic regression framework and has been used in the credit scoring world for decades (Bhalla 2015). Since it evolved from credit scoring world, it is generally described as a measure of the separation of good and bad customers. This technique is great when used to calculate the information value (IV) of a variable, which quantifies the predictive capacity. However in the context of generative networks we do not have a target variable as with a typical machine learning setting. Instead using the weigh of evidence as inspiration we define an encoding which we will refer to as the *weight of a category c*. The proposed encoding is as follows:

$$W(c) = \log\left(\frac{\#\text{ of non-}c\text{'s}}{\#\text{ of }c\text{'s}}\right) \tag{8}$$

Here log denotes the natural logarithm. In the event that two categories result in the same count then vary the count of one of these variables by 1. That is, if possible then either add or subtract 1 from the count of one of the variables to avoid a collision.

2.3 GAINs on Survey Data

The proposed method is then to combine both ideas to complete[1] the dataset. First, given the survey data set one must use the encoding technique described in 2.2 to encode all categories in each column, and filling missing values with 0s. This should output a "new" dataset with only numerical variables. We then train the proposed imputation networks. To accomplish this we split the data into a train and test split and depending on the loss adjust hyperparameters. Once the networks are trained then we apply the generator to the entire data set. Then to

[1]Here complete means to fill the missing values in the data set using the proposed imputation algorithm.

"translate" back to categorical we use the nearest neighbor approach. Suppose the original set of categories $C = \{c_1, \ldots, c_k\}$ gets encoded into $C_{weights} = \{c'_1, \ldots, c'_k\}$ where each $c'_i \in \mathbb{R}$. Then for an imputed prediction $\hat{x}$ we replace $\hat{x}$ with

$$\hat{c}' = \text{argmin}_{c' \in C_{weights}} ||\hat{x} - c'||$$

where $||a - b||$ denotes the usual euclidean norm. Then clearly $\hat{c}' \in C_{weights}$ and we can decode back onto a categorical variable.

3. Experiment

3.1 Data

To quantify the efficacy of the proposed alternative we conducted two experiments, first on a traditional web survey, the *Kaggle Data Science Survey*, and the *American Community Survey* (ACS), a census sample, to test the method on a large scale survey.

Before talking about the specific data sets we must comment on the differences between a probability and a non-probability survey. At its core a non-probability survey is sample obtained by non-random selection, or a selection without the use of probability. This contrasts with probability samples, where, a full list of the target population is available from which respondents are selected uniformly at random, although the exact method may vary depending on the study.

Most contemporary research involving surveys use non-probability surveys due to the simplicity in terms of logistics and financial costs. However, it is important to point out that non-probability surveys trade complexity for risk of bias. A big portion of research has been devoted to adjusting biased survey samples or non-representative samples. An example of a popular method is multilevel regression with poststratification (Little 1993; Park, Gelman, and Bafumi 2004).

The two surveys used are examples of non-probability and probability surveys respectively. In particular we chose an online survey (Kaggle Data Science Survey) as our non-probability survey because most non-probability surveys in the past years have been online convenience samples. Moreover we used the ACS data as our probability survey because as online non-probability surveys become more prevalent, census data and census samples by definition have the requirement of being non-biased. It is because of this that probability surveys remain the most convenient tool to get census data with no biases.

The 2020 Kaggle Machine Learning & Data Science survey (Kaggle 2020) was an online survey conducted from October 7 to 30 or approximately 3.5 weeks. As with most contemporary surveys, the survey is not a random sample from the population of interest. Instead, an invitation to participate in the survey was sent to anyone who opted-in to the Kaggle email list. Cleaned data was released under a CC 2.0 license as part of an ongoing data science competition hosted on Kaggle[2].

[2]Kaggle Data and Challenge: https://www.kaggle.com/c/kaggle-survey-2020/overview

A subset of questions were selected for this experiment. We limited the number of questions because the original survey contained logic that would display different questions to respondents depending on their previous answers. Therefore, we selected 8 questions that where asked to all respondents. The resulting subset contained rows with missing values which where removed for this experiment. Therefore the data used was a Kaggle subset containing 8 columns and $16,374$ rows.

It is important to point out that most online surveys do not achieve such a high response rate. Hence, to test how sample size affects the proposed method we extracted two smaller subsets from the original Kaggle survey. One of size of 1000 and the other of size 100. The selection was done at random to avoid bias.

The American Community Surveys (ACS) (Steven Ruggles and Sobek 2020) is a project of the U.S. Census Bureau that has replaced the decennial census as the key source of information about the American population and its housing characteristics. This survey has been conducted since 2000 and the most recent sample released is from 2018. An important distinction is that the ACS is a sample and not a full census data set. More information on how to access the ACS data[3] can be found in the Appendix.

Moreover, the ACS survey is sent to a sample of addresses (about 3.5 million) in the 50 states, District of Columbia, and Puerto Rico and it is conducted every month, every year. The Master Address File (MAF) is the Census Bureau's official inventory of known housing units in the United States and Puerto Rico. It serves as the source of addresses and hence sampling frame for the ACS. Their sampling process is a complicated 2 phase process but in summary first they assign addresses to sixteen sampling strata, then determine base rate and calculate stratum sampling rates and finally systematically selecting samples. Hence, we can classify the ACS as a probability sample.

To access the ACS surveys an account from IPUMS USA website is required[ˆ2]. The database allows for the creation of a customized data set. In particular we chose the 2018 ACS survey and selected the following variables: sex, age, race, and Hispanic origin. Automatically, other variables are appended to the selection, and we removed them for the purpose of the experiment. There were no missing values in the data set. The ACS data set contained 4 columns and $2,499,621$ rows.

3.3 Results

In total we used 4 data sets. The fully cleaned ACS survey, the cleaned Kaggle survey and the smaller subsets of sizes 1000 and 100. In each test we randomly removed cells to model missing data, specifically we tested removing 10, 20, 30 and 40% of the total cell count. Thirty trials were performed for each percentage to avoid possible biases, i.e. thirty trials removing 10% then thirty removing 20% and so on.

[3]ACS Data: https://usa.ipums.org/usa/

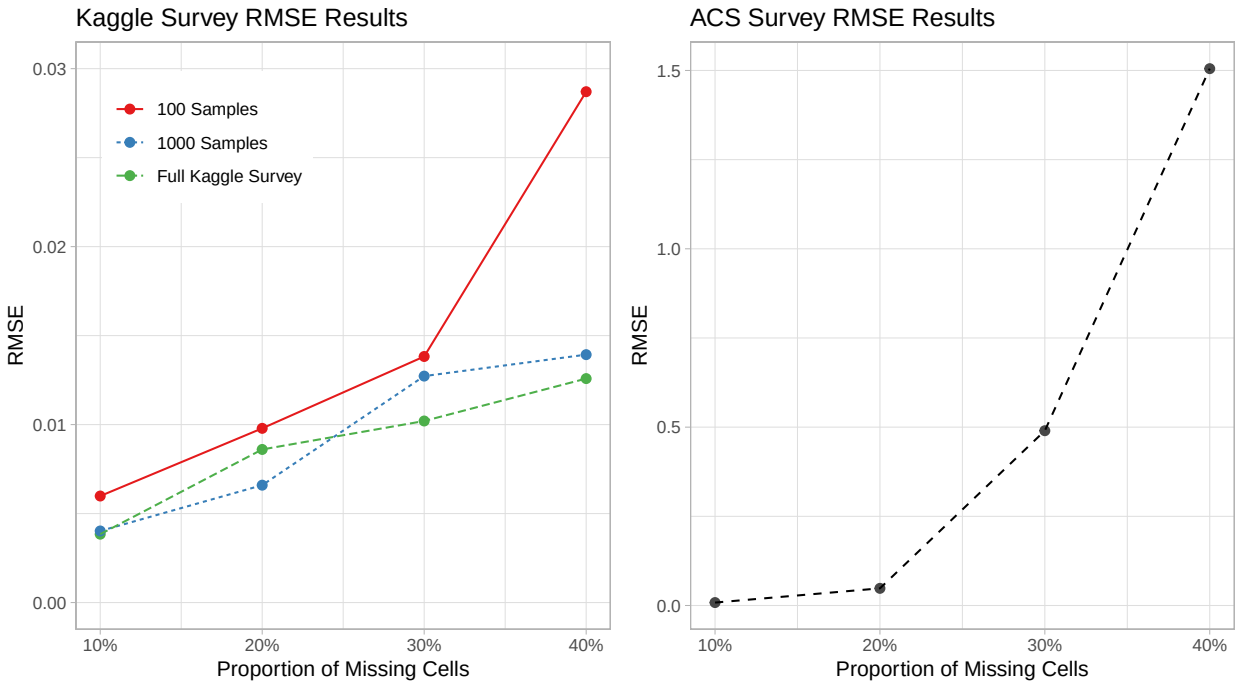

Figure 2: Average RMSE Scores

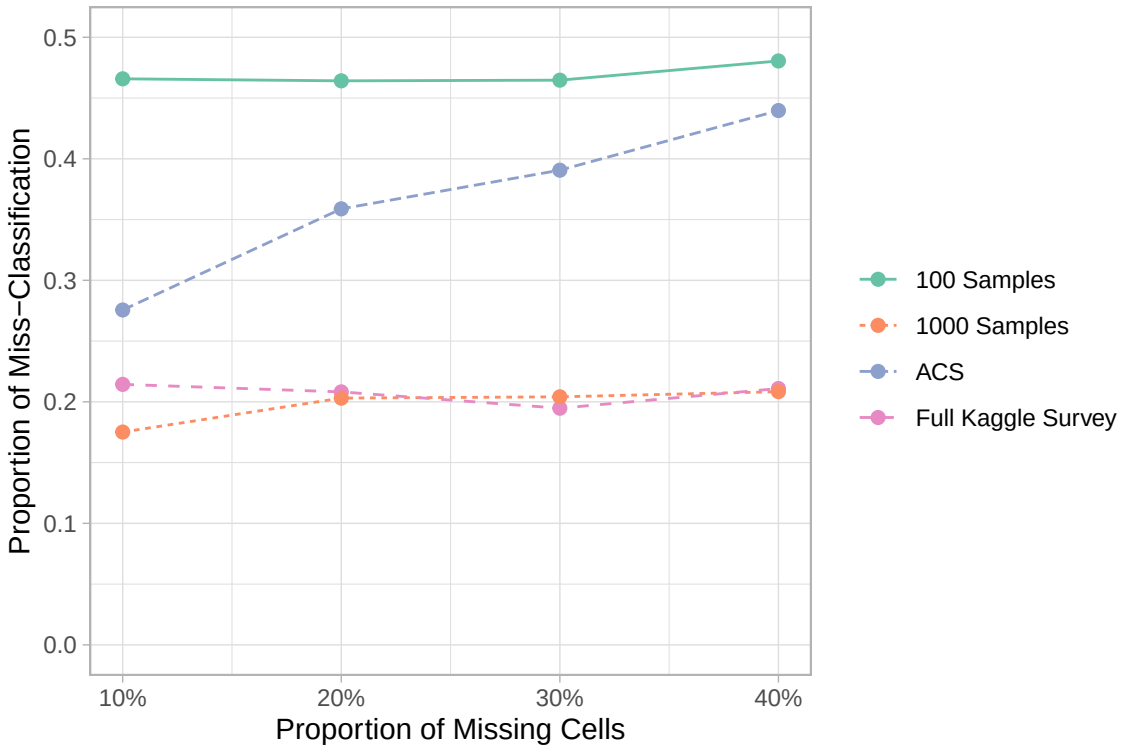

Figure 3: Average Miss-Classification Scores

Table 1: Percentage of Data Recovered.

% Missing Cells	Full Kaggle	1000 Kaggle	100 Kaggle	ACS
10	7.86	8.25	5.34	7.24
20	15.84	15.94	10.72	12.82
30	24.16	23.88	16.06	18.28
40	31.56	31.67	20.78	22.41

We used two metrics for our experiment. Root mean squared error (RMSE), a metric that quantifies the root average distance (error) between the ground truth and the model predictions, was used to evaluate the continuous predictions made by the imputation model. Then, when we converted predictions back to categorical variables and evaluated how many of the imputed values where miss-classified, this lead to calculating the proportion of miss-classification. Figure 2 displays the average RMSE scores for each set of tests and Figure 3 the average proportion of miss-classification.

As expected, in Figure 2 we can see a positive linear trend between the proportion of missing values and RMSE. This is also true for proportion of miss-classification but it is surprising that there is significant difference in the slopes between the plots in Figure 2 and Figure 3. Moreover it is clear that the method performs better on the non-probability survey than on the census sample however we elaborate more on why this is the case in the discussion section.

Finally, Table 1 displays the amount of "recovered" data using our method. What we mean by "recovered" is the proportion of correctly predicted cells, for example: suppose we removed 20% of the cells. Then, we say we recovered 15% if we correctly predicted 15% of the missing cells.

4. Discussion

Let us begin our discussion by commenting on the results obtained from the experiments. Overall we can see the effectiveness of the method for filling missing values, in the worst case we recovered a bit more than half of the missing values and on the better cases we recovered more than 80%. The RMSE scores obtained seem to be unexpectedly small however, it is important to not forget that they were expected to be. This is because the majority of the data remained untouched.

In the results section we comment on the significant difference between the slopes of the RMSE plots and the proportion of miss-classification. One possible explanation could be that as we increase the number of untouched variables, the average distance between our predictions and the ground truth increases. This is a common theme when dealing with averages. Moreover, the nearest neighbors approach is effective because we effectively remove this "error" and decode to a categorical variable. It is an interesting question for the future to investigate how these slopes change as more categories are introduced to the set of possible responses.

Figure 2 and more specifically the ACS results may be a bit of a red herring because it appears in plain sight as if the results are much worst than the Kaggle survey. It is important to recall that the RMSE is an average. We mention this because there is considerable difference in dimensions between the Kaggle survey and the ACS. Recall that the ACS data set contains more than 2 million observations in comparison to $16,000$ for the Kaggle survey. In other words the difference in RMSE scores was to be expected.

One of the reasons why we did not test the method on smaller samples of the ACS data set is because we hypothesize that there is no drop in performance due to the type of survey, i.e. there is no difference in performance between probability and non-probability surveys. We make this claim because there is enough research on neural networks having the capacity to represent a wide variety of distributions or more generally continuous functions between euclidean and non-euclidean spaces (Pinkus 1999; Lu et al. 2017; Hanin and Sellke 2018; Kratsios 2020).

Having said that, Figure 3 provides evidence that the network did not learn the underlying distribution to the extent that the networks for the Kaggle models did. This is likely because the Kaggle data is a highly biased sample and the distribution that generates it is easier to learn. During the training process we noticed that there is always a "sweet spot" with the number of training epochs that majorly impacts the performance of the networks. It also a possibility that we failed to find the most optimal set of hyperparameters. This is also very common in the deep learning literature due to the volatility of neural networks.

Moreover, the variable that we suspected would have a significant impact in performance is size of available data and both Figures 2 and 3 provide evidence of this claim. This is a common theme in the deep learning literature, the more data available the better the performance of neural networks. In practice, the method should be applied on samples with at least 1000 observations, although this is just a recommendation more than an empirical fact. An area of future work could be developing a more appropriate method for survey response imputation when the available sample size is small, i.e. in the hundreds or less.

To comment on some of the limitations of the method, we first must address what is perhaps the biggest limitation of all imputation algorithms and that is that at the end of the day we replace missing values with mere predictions. In a controlled setting like the one in this experiment we have a ground truth to compare our predictions but in a practical setting this is not the case. The argument that this paper, and most imputation algorithm papers make is that the predictions can be very "close" to the ground truth.

In addition, the biggest drawback of the Survey GAIN method, specifically, is the complexity that comes associated with it. The GAIN framework is perhaps one of the best imputation algorithms at the moment however, the requirement to train two neural networks is a huge limitation. This is because survey researchers need to find the best set of hyperparameters like network architecture, activation functions, learning rates, etc.

What's more is that the method is not even an option for researchers whose main discipline is not statistics or computer science. Even in the scenario where the researcher's main discipline is statistics or computer science, there is an another prerequisite namely, a minimum prior understanding of generative models and deep learning.

As a counterpoint we would like to argue that the proposed method is in its best state and why a change to the main imputation method may be harmful to performance. Moreover we will briefly comment on how the previously stated drawbacks could be solved.

As mentioned the GAIN framework has shown to be among the best performing imputation algorithms however, it is not the best proven method. A more recent method MisGAN by Li, Jiang, and Marlin (2019) displayed better results than GAINs. The problem is that the method involves training five networks instead of two, making it an even less practical solution to the imputation problem than GAINs. On the contrary other imputation algorithms, which may be simpler, such as expectation–maximization (EM), multiple imputation by chained equations (MICE), matrix factorization or auto-encoders may perform worse and suffer significant performance differences depending on the type of survey.

The development of automated imputation pipelines for surveys which leverage the proposed method, could potentially solve the complexity problems that are associated. The best set of hyperparameters can be found in an algorithmic way with methods such as grid search. The code used for this experiment follows some of the basic software engineering principles and can easily be extended into a more robust framework, a link to the repository hosting the code used in this project can be found in the Appendix. Making such a software package is by no means easy but it certainly proposes a solution if the method is to be widely adopted.

Finally, the most noticeable limitation of our method is the inefficiency of the nearest neighbors algorithm. This popular machine learning method is well known to be a very inefficient algorithm when it comes to time and space complexity mainly because at all times we have to keep a set of values in memory and test each one of them one by one. As a result there is a noticeable increase in training and decoding time when using the ACS data set and should be taken into consideration when dealing with very large data sets.

Data imputation will continue to be an area that can be improved, however there is an interesting opportunity for future work on developing better imputation algorithms for niche data sets like surveys. Similarly other algorithms like MisGANs could be used to test how much better they are against GAINS. Finally, the encoding methodology is what could be improved the most, nearest neighbors is known to be a simple and effective algorithm that comes at a high computational cost both in time and memory. A better encoding technique would introduce a much more efficient encoding and decoding method.

In conclusion we have proposed a new data imputation alternative when dealing with survey data or when the data set is comprised of only categorical variables. The proposed method leverages the already existing imputation algorithm known as generative adversarial imputation networks and a new encoding/decoding methodology for transforming categorical responses to numerical observations. We conducted experiments on a non-probability and probability survey and found that the method accurately predicts the missing values, in the best cases we accurately predicted more than 80% of the missing values. Some suggestions were given in order to adopt this method in a more practical setting as well as multiple suggestions for future work.

Appendix

All the code used for this project can be found here:

https://github.com/cesar-yoab/Survey-GAINs

Network Hyperparameters

As explained in the method section, GAINs are made up of two networks, a generator and a discriminator. The original paper calls for both networks to be made up of fully connected layers. In the experiment we used the same network architectures for the generator and discriminator. The architectures are as follows:

Kaggel Network Architecture

$$\text{input}(16) \rightarrow FC(16, 64) \underset{GELU}{\rightarrow} FC(64, 128) \underset{GELU}{\rightarrow} FC(128, 64) \underset{GELU}{\rightarrow} FC(64, 8) \rightarrow \text{output}(8)$$

ACS Network Architecture

$$\text{input}(8) \rightarrow FC(8, 64) \underset{GELU}{\rightarrow} FC(64, 128) \underset{GELU}{\rightarrow} FC(128, 64) \underset{GELU}{\rightarrow} FC(64, 4) \rightarrow \text{output}(4)$$

Here FC denotes a fully connected layer, and $GELU$ the activation function (Hendrycks and Gimpel 2020) used for that layer. We used the Adam optimizer with learning rate set to 0.001 for all experiments. While training the networks we noticed that there is always a "sweet spot" on the number of epochs in which the best performance is achieved. In Tables x and y we provide the number of epochs used and the α value used in the GAIN training algorithm.

Table 2: Number of training epochs used.

Full Kaggle	1000 Kaggle	100 Kaggle	ACS
40	30	30	1

Table 3: Alpha value used in GAIN training.

Full Kaggle	1000 Kaggle	100 Kaggle	ACS
1000	100	100	1000

References

Bhalla, Deepanshu. 2015. "Weight of Evidence (Woe) and Information Value (Iv) Explained." https://www.listendata.com/2015/03/weight-of-evidence-woe-and-information.html.

Buttice, Matthew K., and Benjamin Highton. 2017. "How Does Multilevel Regression and Poststratification Perform with Conventional National Surveys?" *Political Analysis* 21 (4): 449–67. https://doi.org/10.1093/pan/mpt017.

Goodfellow, Ian J., Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. "Generative Adversarial Networks." http://arxiv.org/abs/1406.2661.

Hanin, Boris, and Mark Sellke. 2018. "Approximating Continuous Functions by Relu Nets of Minimal Width." http://arxiv.org/abs/1710.11278.

Hendrycks, Dan, and Kevin Gimpel. 2020. "Gaussian Error Linear Units (Gelus)." http://arxiv.org/abs/1606.08415.

Kaggle. 2020. "2020 Kaggle Machine Learning & Data Science Survey." https://www.kaggle.com/c/kaggle-survey-2020/overview.

Kratsios, Anastasis. 2020. "The Universal Approximation Property." http://arxiv.org/abs/1910.03344.

Li, Steven Cheng-Xian, Bo Jiang, and Benjamin Marlin. 2019. "MisGAN: Learning from Incomplete Data with Generative Adversarial Networks." http://arxiv.org/abs/1902.09599.

Little, R. J. A. 1993. "Post-Stratification: A Modeler's Perspective." *Journal of the American Statistical Association* 88 (423): 1001–12. http://www.jstor.org/stable/2290792.

Lu, Zhou, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. 2017. "The Expressive Power of Neural Networks: A View from the Width." http://arxiv.org/abs/1709.02540.

Park, David K., Andrew Gelman, and Joseph Bafumi. 2004. "Bayesian Multilevel Estimation with Poststratification: State-Level Estimates from National Polls." *Political Analysis* 12 (4): 375–85. https://doi.org/10.1093/pan/mph024.

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, et al. 2019. "PyTorch: An Imperative Style, High-Performance Deep Learning Library." In *Advances in Neural Information Processing Systems 32*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, 8024–35. Curran Associates, Inc. http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pinkus, Allan. 1999. "Approximation Theory of the Mlp Model in Neural Networks." *Acta Numerica* 8: 143–95. https://doi.org/10.1017/S0962492900002919.

R Core Team. 2020. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. https://www.R-project.org/.

Steven Ruggles, Ronald Goeken, Sarah Flood, and Matthew Sobek. 2020. "IPUMS Usa: Version 10.0 [American Community Surveys 2018]." https://doi.org/0.18128/D010.V10.0.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wilke, Claus O. 2019. *Cowplot: Streamlined Plot Theme and Plot Annotations for 'Ggplot2'*. https://CRAN.R-project.org/package=cowplot.

Xie, Yihui. 2020. *Knitr: A General-Purpose Package for Dynamic Report Generation in R*. https://yihui.org/knitr/.

Yoon, Jinsung, James Jordon, and Mihaela van der Schaar. 2018. "GAIN: Missing Data Imputation Using Generative Adversarial Nets." http://arxiv.org/abs/1806.02920.