Filling Survey Missing Data With Generative Adversarial Imputation Networks

Cesar Y. Villarreal Guzman

19 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 "Stack Overflow Annual Developer Survey", we experiment with different proportions of missing values and sample sizes and find the technique to yield high quality data 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 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*" and the "*Stack Overflow Annual Developer Survey*" to test the effectiveness of this method. In this section we comment on the surveys, network architectures and hyperparameters 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 the theoretical details.

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 $i \in \{1, \dots, d\}$ 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 $\tilde{\mathbf{x}}^1, \dots, \tilde{\mathbf{x}}^n$ 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

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

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

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

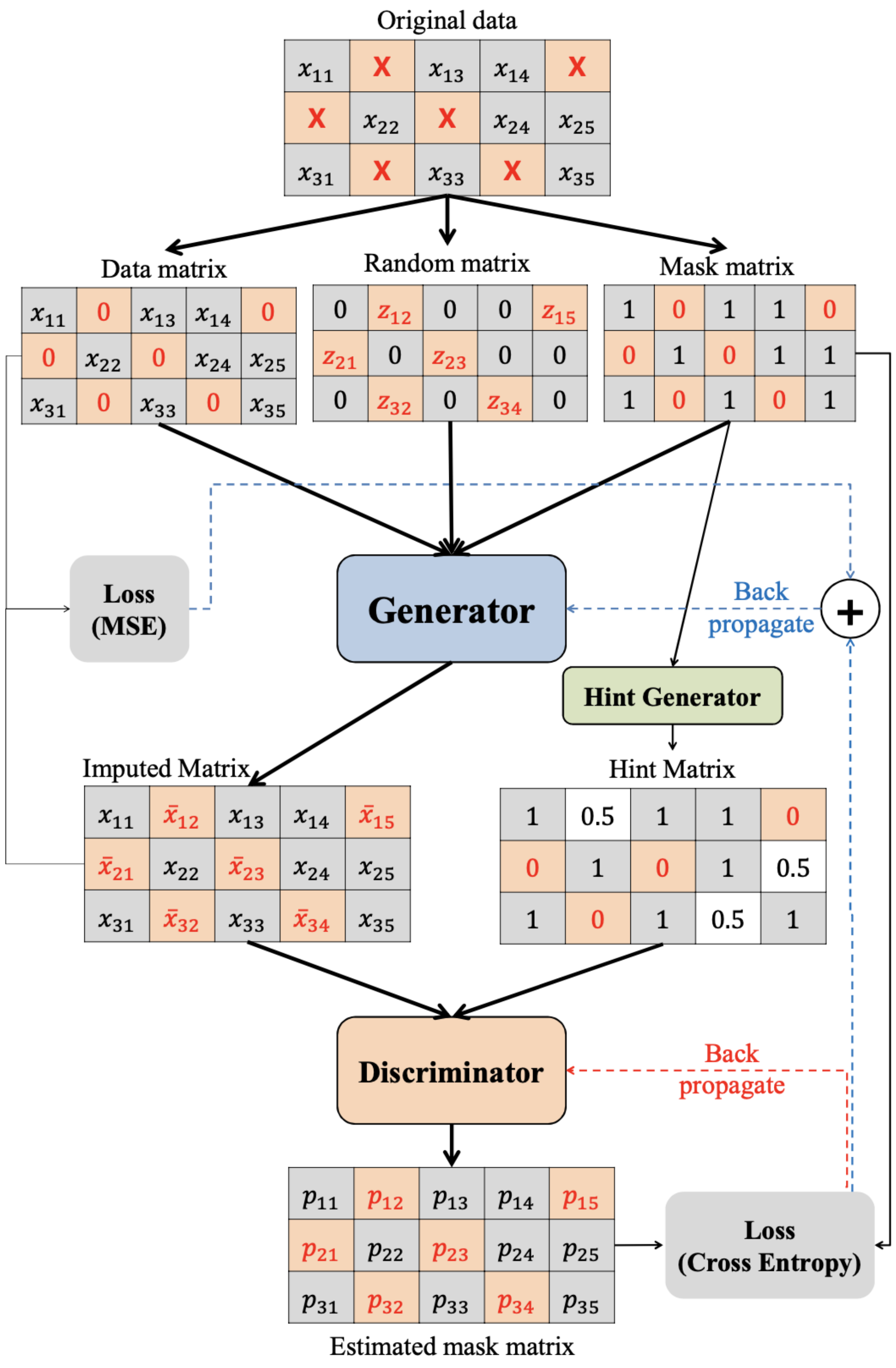
Original data
x_{11} X x_{13} x_{14} X
X x_{22} X x_{24} x_{25}
x_{31} X x_{33} X x_{35}
Data matrix
x_{11} 0 x_{13} x_{14} 0
0 x_{22} 0 x_{24} x_{25}
x_{31} 0 x_{33} 0 x_{35}
Random matrix
0 z_{12} 0 0 z_{15}
z_{21} 0 z_{23} 0 0
0 z_{32} 0 z_{34} 0
Mask matrix
1 0 1 1 0
0 1 0 1 1
1 0 1 0 1
Loss (MSE)
Generator
Back propagate
+
Hint Generator
Imputed Matrix
x_{11} $\bar{x}_{12}$ x_{13} x_{14} $\bar{x}_{15}$
$\bar{x}_{21}$ x_{22} $\bar{x}_{23}$ x_{24} x_{25}
x_{31} $\bar{x}_{32}$ x_{33} $\bar{x}_{34}$ x_{35}
Hint Matrix
1 0.5 1 1 0
0 1 0 1 0.5
1 0 1 0.5 1
Discriminator
Back propagate
p_{11} p_{12} p_{13} p_{14} p_{15}
p_{21} p_{22} p_{23} p_{24} p_{25}
p_{31} p_{32} p_{33} p_{34} p_{35}
Estimated mask matrix
Loss (Cross Entropy)

Figure 1: GAIN Architecture

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

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

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

Theoretical details and outline of the proposed 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:

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

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. A collision means two values which are encoded onto the same number. In practice on should keep track of the weights to convert back and forth from weight to category.

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

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

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

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

3. Experiment

To quantify the efficacy of the proposed alternative we conducted two experiments, first on a traditional web survey, the *Kaggle Data Science Survey*. With it we created smaller samples of the original survey to test the performance when the number of observations decreases. We also used census data from the *American Community Survey*(ACS) to test the efficiency of the method on a large scale survey. In both tests we randomly removed cells to model missing data, specifically we tested removing 10, 20, 30 40 and 50% of the total cell count. Moreover since we removed at random, thirty trials were performed for each percentage, i.e. thirty trials removing 10% then thirty removing 20% with and so on. On the coming sections we give more context on the data used for the experiments and comment on the experiment itself and the results obtained.

3.1 Data

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. his 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 trade complexity for risk of a biased sample. A big portion of research has been devoted to adjusting biased survey samples or non-representative samples. Perhaps the most popular method used in the social sciences is the use multilevel regression with poststratification (Little 1993; Park, Gelman, and Bafumi 2004) and a new method using the same ideas like stacked regression with poststratification (Buttice and Highton 2017).

The two surveys used are examples of non-probability and probability surveys. 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 a non-biased. It is because of this that probability surveys remain the most convenient tool to achieve this.

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 Kaggle 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].

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.

3.2 Data Processing and Network Architectures

To access the ACS surveys an account from IPUMS USA website is required[^2]. 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.

A subset of columns from the Kaggle survey was used for this experiment. We limited the number of columns because the original survey contained logic and depending on some of the answers more questions would be asked. Therefore, we selected 8 questions that where asked to all respondents. This columns contained rows with missing values which where removed for the purposes of this experiment. The remaining Kaggle data set contained 8 columns and $16,374$ rows.

3.3 Results

4. Discussion

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

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

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.

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

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.

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.

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.

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.