GAIN: Missing Data Imputation using Generative Adversarial Nets

Jinsung Yoon [1 *] **James Jordon** [2 *] **Mihaela van der Schaar** [1 2 3]

Abstract

We propose a novel method for imputing missing data by adapting the well-known Generative Adversarial Nets (GAN) framework. Accordingly, we call our method Generative Adversarial Imputation Nets (GAIN). The generator (G) observes some components of a real data vector, imputes the missing components conditioned on what is actually observed, and outputs a completed vector. The discriminator (D) then takes a completed vector and attempts to determine which components were actually observed and which were imputed. To ensure that D forces G to learn the desired distribution, we provide D with some additional information in the form of a *hint* vector. The hint reveals to D *partial* information about the missingness of the original sample, which is used by D to focus its attention on the imputation quality of particular components. This hint ensures that G does in fact learn to generate according to the true data distribution. We tested our method on various datasets and found that GAIN significantly outperforms state-of-the-art imputation methods.

1. Introduction

Missing data is a pervasive problem. Data may be missing because it was never collected, records were lost or for many other reasons. In the medical domain, the respiratory rate of a patient may not have been measured (perhaps because it was deemed unnecessary/unimportant) or accidentally not recorded (Yoon et al., 2017; Alaa et al., 2018). It may also be the case that certain pieces of information are difficult or even dangerous to acquire (such as information gathered from a biopsy), and so these were not gathered for those reasons (Yoon et al., 2018b). An imputation algorithm can be used to estimate missing values based on data that was observed/measured, such as the systolic blood pressure and heart rate of the patient (Yoon et al., 2018c). A substantial amount of research has been dedicated to developing imputation algorithms for medical data (Barnard & Meng, 1999; Mackinnon, 2010; Sterne et al., 2009; Purwar & Singh, 2015). Imputation algorithms are also used in many other applications such as image concealment, data compression, and counterfactual estimation (Rubin, 2004; Kreindler & Lumsden, 2012; Yoon et al., 2018a).

Missing data can be categorized into three types: (1) the data is missing completely at random (MCAR) if the missingness occurs entirely at random (there is no dependency on any of the variables), (2) the data is missing at random (MAR) if the missingness depends only on the *observed* variables[1], (3) the data is missing not at random (MNAR) if the missingness is neither MCAR nor MAR (more specifically, the data is MNAR if the missingness depends on *both observed* variables and the *unobserved* variables; thus, missingness cannot be fully accounted for by the observed variables). In this paper we provide theoretical results for our algorithm under the MCAR assumption, and compare to other state-of-the-art methods in this setting[2].

State-of-the-art imputation methods can be categorized as either discriminative or generative. Discriminative methods include MICE (Buuren & Oudshoorn, 2000; Buuren & Groothuis-Oudshoorn, 2011), MissForest (Stekhoven & Bühlmann, 2011), and matrix completion (Mazumder et al., 2010a; Yu et al., 2016; Schnabel et al., 2016; Mazumder et al., 2010b); generative methods include algorithms based on Expectation Maximization (García-Laencina et al., 2010) and algorithms based on deep learning (e.g. denoising autoencoders (DAE) and generative adversarial nets (GAN)) (Vincent et al., 2008; Gondara & Wang, 2017; Allen & Li, 2016). However, current generative methods for imputation have various drawbacks. For instance, the approach for data imputation based on (García-Laencina et al., 2010) makes assumptions about the underlying distribution and fails to generalize well when datasets contain mixed categorical and continuous variables. In contrast, the approaches based on DAE (Vincent et al., 2008) have been shown to work well in practice but require complete data during training. In

[*]Equal contribution [1]University of California, Los Angeles, CA, USA [2]University of Oxford, UK [3]Alan Turing Institute, UK. Correspondence to: Jinsung Yoon <jsyoon0823@gmail.com>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

[1]A formal definition of MAR can be found in the Supplementary Materials.

[2]Empirical results for the MAR and MNAR settings are shown in the Supplementary Materials.

many circumstances, missing values are part of the inherent structure of the problem so obtaining a complete dataset is impossible. Another approach with DAE (Gondara & Wang, 2017) allows for an incomplete dataset; however, it only utilizes the observed components to learn the representations of the data. (Allen & Li, 2016) uses Deep Convolutional GANs for image completion; however, it also requires complete data for training the discriminator.

In this paper, we propose a novel imputation method, which we call Generative Adversarial Imputation Nets (GAIN), that generalizes the well-known GAN (Goodfellow et al., 2014) and is able to operate successfully even when complete data is unavailable. In GAIN, the generator's goal is to accurately impute 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. Thus, these two networks are trained using an adversarial process. To achieve this goal, GAIN builds on and adapts the standard GAN architecture. To ensure that the result of this adversarial process is the desired target, the GAIN architecture provides the discriminator with additional information in the form of "hints". This hinting ensures that the generator generates samples according to the true underlying data distribution.

2. Problem Formulation

Consider a d-dimensional space $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_d$. Suppose that $\mathbf{X} = (X_1, ..., X_d)$ is a random variable (either continuous or binary) taking values in $\mathcal{X}$, whose distribution we will denote $P(\mathbf{X})$. Suppose that $\mathbf{M} = (M_1, ..., M_d)$ is a random variable taking values in $\{0, 1\}^d$. We will call $\mathbf{X}$ the data vector, and $\mathbf{M}$ the mask vector.

For each $i \in \{1, ..., d\}$, we define a new space $\tilde{\mathcal{X}}_i = \mathcal{X}_i \cup \{*\}$ where $*$ is simply a point not in any $\mathcal{X}_i$, representing an unobserved value. Let $\tilde{\mathcal{X}} = \tilde{\mathcal{X}}_1 \times ... \times \tilde{\mathcal{X}}_d$. We define a new random variable $\tilde{\mathbf{X}} = (\tilde{X}_1, ..., \tilde{X}_d) \in \tilde{\mathcal{X}}$ in the following way:

$$\tilde{X}_i = \begin{cases} X_i, & \text{if } M_i = 1 \\ *, & \text{otherwise} \end{cases} \quad (1)$$

so that $\mathbf{M}$ indicates which components of $\mathbf{X}$ are observed. Note that we can recover $\mathbf{M}$ from $\tilde{\mathbf{X}}$.

Throughout the remainder of the paper, we will often use lower-case letters to denote realizations of a random variable and use the notation $\mathbf{1}$ to denote a vector of 1s, whose dimension will be clear from the context (most often, d).

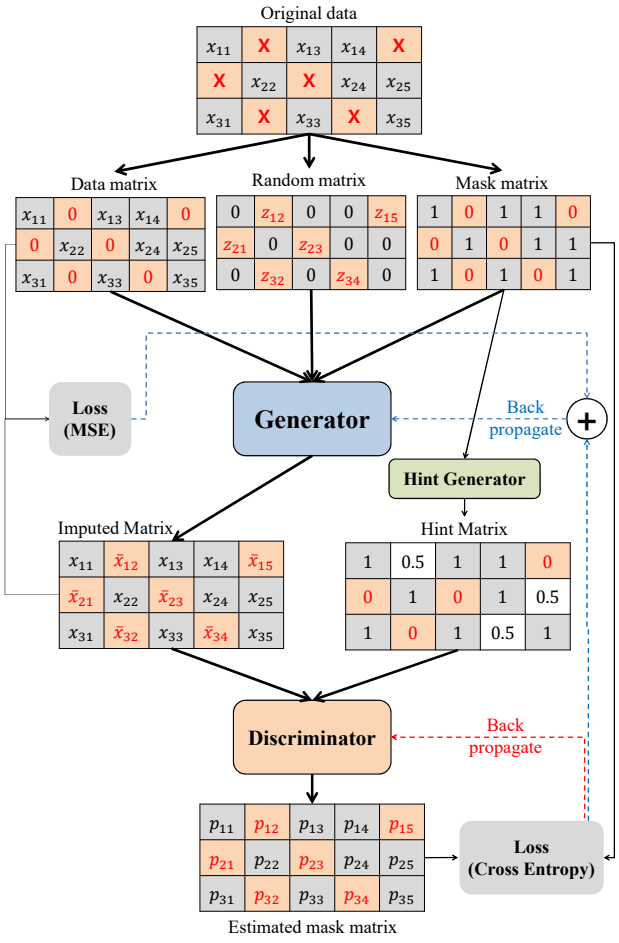

Figure 1. The architecture of GAIN

2.1. Imputation

In the imputation setting, n i.i.d. copies of $\tilde{\mathbf{X}}$ are realized, denoted $\tilde{\mathbf{x}}^1, ..., \tilde{\mathbf{x}}^n$ and we define the dataset $\mathcal{D} = \{(\tilde{\mathbf{x}}^i, \mathbf{m}^i)\}_{i=1}^n$, where $\mathbf{m}^i$ is simply the recovered realization of $\mathbf{M}$ corresponding to $\tilde{\mathbf{x}}^i$.

Our goal is to *impute* the unobserved values in each $\tilde{\mathbf{x}}_i$. Formally, we want to generate samples according to $P(\mathbf{X}|\tilde{\mathbf{X}} = \tilde{\mathbf{x}}^i)$, the conditional distribution of $\mathbf{X}$ given $\tilde{\mathbf{X}} = \tilde{\mathbf{x}}^i$, for each i, to fill in the missing data points in $\mathcal{D}$. By attempting to model the *distribution* of the data rather than just the expectation, we are able to make multiple draws and therefore make *multiple imputations* allowing us to capture the uncertainty of the imputed values (Buuren & Oudshoorn, 2000; Buuren & Groothuis-Oudshoorn, 2011; Rubin, 2004).

3. Generative Adversarial Imputation Nets

In this section we describe our approach for simulating $P(\mathbf{X}|\tilde{\mathbf{X}} = \tilde{\mathbf{x}}^i)$ which is motivated by GANs. We highlight key similarities and differences to a standard (conditional)

GAN throughout. Fig. 1 depicts the overall architecture.

3.1. Generator

The generator, G, takes (realizations of) $\tilde{\mathbf{X}}$, $\mathbf{M}$ and a noise variable, $\mathbf{Z}$, as input and outputs $\bar{\mathbf{X}}$, a vector of imputations. Let $G : \tilde{\mathcal{X}} \times \{0,1\}^d \times [0,1]^d \to \mathcal{X}$ be a function, and $\mathbf{Z} = (Z_1, ..., Z_d)$ be d-dimensional noise (independent of all other variables).

Then we define the random variables $\bar{\mathbf{X}}, \hat{\mathbf{X}} \in \mathcal{X}$ by

$$\bar{\mathbf{X}} = G(\tilde{\mathbf{X}}, \mathbf{M}, (\mathbf{1} - \mathbf{M}) \odot \mathbf{Z}) \tag{2}$$

$$\hat{\mathbf{X}} = \mathbf{M} \odot \tilde{\mathbf{X}} + (\mathbf{1} - \mathbf{M}) \odot \bar{\mathbf{X}} \tag{3}$$

where $\odot$ denotes element-wise multiplication. $\bar{\mathbf{X}}$ corresponds to the vector of *imputed* values (note that G outputs a value for every component, even if its value was observed) and $\hat{\mathbf{X}}$ corresponds to the completed data vector, that is, the vector obtained by taking the partial observation $\tilde{\mathbf{X}}$ and replacing each $*$ with the corresponding value of $\bar{\mathbf{X}}$.

This setup is very similar to a standard GAN, with $\mathbf{Z}$ being analogous to the noise variables introduced in that framework. Note, though, that in this framework, the target distribution, $P(\mathbf{X}|\tilde{\mathbf{X}})$, is essentially $||\mathbf{1} - \mathbf{M}||_1$-dimensional and so the noise we pass into the generator is $(\mathbf{1} - \mathbf{M}) \odot \mathbf{Z}$, rather than simply $\mathbf{Z}$, so that its dimension matches that of the targeted distribution.

3.2. Discriminator

As in the GAN framework, we introduce a discriminator, D, that will be used as an adversary to train G. However, unlike in a standard GAN where the output of the generator is either *completely* real or *completely* fake, in this setting the output is comprised of some components that are real and some that are fake. Rather than identifying that an entire vector is real or fake, the discriminator attempts to distinguish which *components* are real (observed) or fake (imputed) - this amounts to predicting the mask vector, $\mathbf{m}$. Note that the mask vector $\mathbf{M}$ is pre-determined by the dataset.

Formally, the discriminator is a function $D : \mathcal{X} \to [0,1]^d$ with the i-th component of $D(\hat{\mathbf{x}})$ corresponding to the probability that the i-th component of $\hat{\mathbf{x}}$ was observed.

3.3. Hint

As will be seen in the theoretical results that follow, it is necessary to introduce what we call a hint mechanism. A hint mechanism is a random variable, $\mathbf{H}$, taking values in a space $\mathcal{H}$, both of which *we define*. We allow $\mathbf{H}$ to depend on $\mathbf{M}$ and for each (imputed) sample $(\hat{\mathbf{x}}, \mathbf{m})$, we draw $\mathbf{h}$ according to the distribution $\mathbf{H}|\mathbf{M} = \mathbf{m}$. We pass $\mathbf{h}$ as an additional input to the discriminator and so it becomes a function $D : \mathcal{X} \times \mathcal{H} \to [0,1]^d$, where now the i-th component of $D(\hat{\mathbf{x}}, \mathbf{h})$ corresponds to the probability that the i-th component of $\hat{\mathbf{x}}$ was observed conditional on $\hat{\mathbf{X}} = \hat{\mathbf{x}}$ *and* $\mathbf{H} = \mathbf{h}$.

By defining $\mathbf{H}$ in different ways, we control the amount of information contained in $\mathbf{H}$ about $\mathbf{M}$ and in particular we show (in Proposition 1) that if we do not provide "enough" information about $\mathbf{M}$ to D (such as if we simply did not have a hinting mechanism), then there are several distributions that G could reproduce that would all be optimal with respect to D.

3.4. Objective

We train D to *maximize* the probability of correctly predicting $\mathbf{M}$. We train G to *minimize* the probability of D predicting $\mathbf{M}$. We define the quantity $V(D, G)$ to be

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

where $\log$ is element-wise logarithm and dependence on G is through $\hat{\mathbf{X}}$.

Then, as with the standard GAN, we define the objective of GAIN to be the minimax problem given by

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

We define the loss function $\mathcal{L} : \{0,1\}^d \times [0,1]^d \to \mathbb{R}$ by

$$\mathcal{L}(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{d} \Big[a_i \log(b_i) + (1 - a_i) \log(1 - b_i)\Big]. \tag{6}$$

Writing $\hat{\mathbf{M}} = D(\hat{\mathbf{X}}, \mathbf{H})$, we can then rewrite (5) as

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

4. Theoretical Analysis

In this section we provide a theoretical analysis of (5). Given a d-dimensional space $\mathcal{Z} = \mathcal{Z}_1 \times ... \times \mathcal{Z}_d$, a (probability) density[3] p over $\mathcal{Z}$ corresponding to a random variable Z, and a vector $\mathbf{b} \in \{0,1\}^d$ we define the set $A_\mathbf{b} = \{i : b_i = 1\}$, the projection $\phi_\mathbf{b} : \mathcal{Z} \to \Pi_{i \in A_\mathbf{b}} \mathcal{Z}_i$ by $\phi_\mathbf{b}(z) = (z_i)_{i \in A}$ and the density $p^\mathbf{b}$ to be the density of $\phi_\mathbf{b}(Z)$.

Throughout this section, we make the assumption that $\mathbf{M}$ is independent of $\mathbf{X}$, i.e. that the data is MCAR.

We will write $p(\mathbf{x}, \mathbf{m}, \mathbf{h})$ to denote the density of the random variable $(\hat{\mathbf{X}}, \mathbf{M}, \mathbf{H})$ and we will write $\hat{p}$, p_m and p_h to

[3] For ease of exposition, we use the term density even when referring to a probability mass function.

denote the marginal densities (of p) corresponding to $\hat{\mathbf{X}}$, $\mathbf{M}$ and $\mathbf{H}$, respectively. When referring to the joint density of two of the three variables (potentially conditioned on the third), we will simply use p, abusing notation slightly.

It is more intuitive to think of this density through its decomposition into densities corresponding to the true data generating process, and to the generator defined by (2),

$$p(\mathbf{x}, \mathbf{m}, \mathbf{h}) = p_m(\mathbf{m})\hat{p}^{\mathbf{m}}(\phi_{\mathbf{m}}(\mathbf{x}|\mathbf{m})) \times \hat{p}^{\mathbf{1}-\mathbf{m}}(\phi_{\mathbf{1}-\mathbf{m}}(\mathbf{x})|\mathbf{m}, \phi_{\mathbf{m}}(\mathbf{x}))p_h(\mathbf{h}|\mathbf{m}). \tag{8}$$

The first two terms in (8) are both defined by the data, where $\hat{p}^{\mathbf{m}}(\phi_{\mathbf{m}}(\mathbf{x})|\mathbf{m})$ is the density of $\phi_{\mathbf{m}}(\hat{\mathbf{X}})|\mathbf{M} = \mathbf{m}$ which corresponds to the density of $\phi_{\mathbf{m}}(\mathbf{X})$ (i.e. the true data distribution), since conditional on $\mathbf{M} = \mathbf{m}$, $\phi_{\mathbf{m}}(\hat{\mathbf{X}}) = \phi_{\mathbf{m}}(\mathbf{X})$ (see equations 1 and 3). The third term, $\hat{p}^{\mathbf{1}-\mathbf{m}}(\phi_{\mathbf{1}-\mathbf{m}}(\mathbf{x})|\mathbf{m}, \phi_{\mathbf{m}}(\mathbf{x}))$, is determined by the generator, G, and is the density of the random variable $\phi_{\mathbf{1}-\mathbf{m}}(G(\tilde{\mathbf{x}}, \mathbf{m}, \mathbf{Z})) = \phi_{\mathbf{1}-\mathbf{m}}(\bar{\mathbf{X}})|\tilde{\mathbf{X}} = \tilde{\mathbf{x}}, \mathbf{M} = \mathbf{m}$ where $\tilde{\mathbf{x}}$ is determined by $\mathbf{m}$ and $\phi_{\mathbf{m}}(\mathbf{x})$. The final term is the conditional density of the hint, which we are free to define (its selection will be motivated by the following analysis).

Using this decomposition, one can think of drawing a sample from $\hat{p}$ as first sampling $\mathbf{m}$ according to $p_m(\cdot)$, then sampling the "observed" components, $\mathbf{x}_{obs}$, according to $\hat{p}^{\mathbf{m}}(\cdot)$ (we can then construct $\tilde{\mathbf{x}}$ from $\mathbf{x}_{obs}$ and $\mathbf{m}$), then *generating* the imputed values, $\mathbf{x}_{imp}$, from the generator according to $\hat{p}^{\mathbf{1}-\mathbf{m}}(\cdot|\mathbf{m}, \mathbf{x}_{obs})$ and finally sampling the hint according to $p_h(\cdot|\mathbf{m})$.

Lemma 1. *Let $\mathbf{x} \in \mathcal{X}$. Let p_h be a fixed density over the hint space $\mathcal{H}$ and let $\mathbf{h} \in \mathcal{H}$ be such that $p(\mathbf{x}, \mathbf{h}) > 0$. Then for a fixed generator, G, the i-th component of the optimal discriminator, $D^*(\mathbf{x}, \mathbf{h})$ is given by*

$$D^*(\mathbf{x}, \mathbf{h})_i = \frac{p(\mathbf{x}, \mathbf{h}, m_i = 1)}{p(\mathbf{x}, \mathbf{h}, m_i = 1) + p(\mathbf{x}, \mathbf{h}, m_i = 0)} \tag{9}$$

$$= p_m(m_i = 1|\mathbf{x}, \mathbf{h}) \tag{10}$$

for each $i \in \{1, ..., d\}$.

Proof. All proofs are provided in Supplementary Materials. □

We now rewrite (4), substituting for D^*, to obtain the following minimization criterion for G:

$$C(G) = \mathbb{E}_{\hat{\mathbf{X}}, \mathbf{M}, \mathbf{H}}\Big(\sum_{i:M_i=1} \log p_m(m_i = 1|\hat{\mathbf{X}}, \mathbf{H}) + \sum_{i:M_i=0} \log p_m(m_i = 0|\hat{\mathbf{X}}, \mathbf{H})\Big), \tag{11}$$

where dependence on G is through $p_m(\cdot|\hat{\mathbf{X}})$.

Theorem 1. *A global minimum for $C(G)$ is achieved if and only if the density $\hat{p}$ satisfies*

$$\hat{p}(\mathbf{x}|\mathbf{h}, m_i = t) = \hat{p}(\mathbf{x}|\mathbf{h}) \tag{12}$$

for each $i \in \{1, ..., d\}$, $\mathbf{x} \in \mathcal{X}$ and $\mathbf{h} \in \mathcal{H}$ such that $p_h(\mathbf{h}|m_i = t) > 0$.

The following proposition asserts that if $\mathbf{H}$ does not contain "enough" information about $\mathbf{M}$, we cannot guarantee that G learns the desired distribution (the one uniquely defined by the (underlying) data).

Proposition 1. *There exist distributions of $\mathbf{X}$, $\mathbf{M}$ and $\mathbf{H}$ for which solutions to (12) are not unique. In fact, if $\mathbf{H}$ is independent of $\mathbf{M}$, then (12) does not define a unique density, in general.*

Let the random variable $\mathbf{B} = (B_1, ..., B_d) \in \{0, 1\}^d$ be defined by first sampling k from $\{1, ..., d\}$ uniformly at random and then setting

$$B_j = \begin{cases} 1 \text{ if } j \neq k \\ 0 \text{ if } j = k. \end{cases} \tag{13}$$

Let $\mathcal{H} = \{0, 0.5, 1\}^d$ and, given $\mathbf{M}$, define

$$\mathbf{H} = \mathbf{B} \odot \mathbf{M} + 0.5(\mathbf{1} - \mathbf{B}). \tag{14}$$

Observe first that $\mathbf{H}$ is such that $H_i = t \implies M_i = t$ for $t \in \{0, 1\}$ but that $H_i = 0.5$ implies nothing about M_i. In other words, $\mathbf{H}$ reveals all but one of the components of $\mathbf{M}$ to D. Note, however, that $\mathbf{H}$ does contain some information about M_i since M_i is not assumed to be independent of the other components of $\mathbf{M}$.

The following lemma confirms that the discriminator behaves as we expect with respect to this hint mechanism.

Lemma 2. *Suppose $\mathbf{H}$ is defined as above. Then for $\mathbf{h}$ such that $h_i = 0$ we have $D^*(\mathbf{x}, \mathbf{h})_i = 0$ and for $\mathbf{h}$ such that $h_i = 1$ we have $D^*(\mathbf{x}, \mathbf{h})_i = 1$, for all $\mathbf{x} \in \mathcal{X}$, $i \in \{1, ..., d\}$.*

The final proposition we state tells us that $\mathbf{H}$ as specified above ensures the generator learns to replicate the desired distribution.

Proposition 2. *Suppose $\mathbf{H}$ is defined as above. Then the solution to (12) is unique and satisfies*

$$\hat{p}(\mathbf{x}|\mathbf{m}_1) = \hat{p}(\mathbf{x}|\mathbf{m}_2) \tag{15}$$

for all $\mathbf{m}_1, \mathbf{m}_2 \in \{0, 1\}^d$. In particular, $\hat{p}(\mathbf{x}|\mathbf{m}) = \hat{p}(\mathbf{x}|\mathbf{1})$ and since $\mathbf{M}$ is independent of $\mathbf{X}$, $\hat{p}(\mathbf{x}|\mathbf{1})$ is the density of $\mathbf{X}$. The distribution of $\hat{\mathbf{X}}$ is therefore the same as the distribution of $\mathbf{X}$.

For the remainder of the paper, $\mathbf{B}$ and $\mathbf{H}$ will be defined as in equations (13) and (14).

5. GAIN Algorithm

Using an approach similar to that in (Goodfellow et al., 2014), we solve the minimax optimization problem (5) in an iterative manner. Both G and D are modeled as fully connected neural nets.

We first optimize the discriminator D with a fixed generator G using mini-batches of size k_D[4]. For each sample in the mini-batch, $(\tilde{\mathbf{x}}(j), \mathbf{m}(j))$[5], we draw k_D independent samples, $\mathbf{z}(j)$ and $\mathbf{b}(j)$, of $\mathbf{Z}$ and $\mathbf{B}$ and compute $\hat{\mathbf{x}}(j)$ and $\mathbf{h}(j)$ accordingly. Lemma 2 then tells us that the only outputs of D that depend on G are the ones corresponding to $b_i = 0$ for each sample. We therefore only train D to give us these outputs (if we also trained D to match the outputs specified in Lemma 2 we would gain no information about G, but D would overfit to the hint vector). We define $\mathcal{L}_D : \{0,1\}^d \times [0,1]^d \times \{0,1\}^d \to \mathbb{R}$ by

$$\mathcal{L}_D(\mathbf{m}, \hat{\mathbf{m}}, \mathbf{b}) = \sum_{i:b_i=0} \Big[m_i \log(\hat{m}_i) + (1-m_i)\log(1-\hat{m}_i) \Big]. \tag{16}$$

D is then trained according to

$$\min_D - \sum_{j=1}^{k_D} \mathcal{L}_D(\mathbf{m}(j), \hat{\mathbf{m}}(j), \mathbf{b}(j)) \tag{17}$$

recalling that $\hat{\mathbf{m}}(j) = D(\hat{\mathbf{x}}(j), \mathbf{m}(j))$.

Second, we optimize the generator G using the newly updated discriminator D with mini-batches of size k_G. We first note that G in fact outputs a value for the *entire* data vector (including values for the components we observed). Therefore, in training G, we not only ensure that the imputed values for missing components ($m_j = 0$) successfully fool the discriminator (as defined by the minimax game), we also ensure that the values outputted by G for observed components ($m_j = 1$) are close to those actually observed. This is justified by noting that the conditional distribution of $\mathbf{X}$ given $\tilde{\mathbf{X}} = \tilde{\mathbf{x}}$ obviously fixes the components of $\mathbf{X}$ corresponding to $M_i = 1$ to be $\tilde{X}_i$. This also ensures that the representations learned in the hidden layers of $\tilde{\mathbf{X}}$ suitably capture the information contained in $\tilde{\mathbf{X}}$ (as in an auto-encoder).

To achieve this, we define two different loss functions. The first, $\mathcal{L}_G : \{0,1\}^d \times [0,1]^d \times \{0,1\}^d \to \mathbb{R}$, is given by

$$\mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}}, \mathbf{b}) = -\sum_{i:b_i=0} (1-m_i)\log(\hat{m}_i), \tag{18}$$

Algorithm 1 Pseudo-code of GAIN

while training loss has not converged **do**

(1) Discriminator optimization

Draw k_D samples from the dataset $\{(\tilde{\mathbf{x}}(j), \mathbf{m}(j))\}_{j=1}^{k_D}$

Draw k_D i.i.d. samples, $\{\mathbf{z}(j)\}_{j=1}^{k_D}$, of $\mathbf{Z}$

Draw k_D i.i.d. samples, $\{\mathbf{b}(j)\}_{j=1}^{k_D}$, of $\mathbf{B}$

for $j = 1, ..., k_D$ **do**

$\bar{\mathbf{x}}(j) \leftarrow G(\tilde{\mathbf{x}}(j), \mathbf{m}(j), \mathbf{z}(j))$

$\hat{\mathbf{x}}(j) \leftarrow \mathbf{m}(j) \odot \tilde{\mathbf{x}}(j) + (\mathbf{1} - \mathbf{m}(j)) \odot \bar{\mathbf{x}}(j)$

$\mathbf{h}(j) = \mathbf{b}(j) \odot \mathbf{m}(j) + 0.5(\mathbf{1} - \mathbf{b}(j))$

end for

Update D using stochastic gradient descent (SGD)

$$\nabla_D - \sum_{j=1}^{k_D} \mathcal{L}_D(\mathbf{m}(j), D(\hat{\mathbf{x}}(j), \mathbf{h}(j)), \mathbf{b}(j))$$

(2) Generator optimization

Draw k_G samples from the dataset $\{(\tilde{\mathbf{x}}(j), \mathbf{m}(j))\}_{j=1}^{k_G}$

Draw k_G i.i.d. samples, $\{\mathbf{z}(j)\}_{j=1}^{k_G}$ of $\mathbf{Z}$

Draw k_G i.i.d. samples, $\{\mathbf{b}(j)\}_{j=1}$ of $\mathbf{B}$

for $j = 1, ..., k_G$ **do**

$\mathbf{h}(j) = \mathbf{b}(j) \odot \mathbf{m}(j) + 0.5(\mathbf{1} - \mathbf{b}(j))$

end for

Update G using SGD (for fixed D)

$$\nabla_G \sum_{j=1}^{k_G} \mathcal{L}_G(\mathbf{m}(j), \hat{\mathbf{m}}(j), \mathbf{b}(j)) + \alpha \mathcal{L}_M(\mathbf{x}(j), \tilde{\mathbf{x}}(j))$$

end while

and the second, $\mathcal{L}_M : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, by

$$\mathcal{L}_M(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} m_i L_M(x_i, x_i'), \tag{19}$$

where

$$L_M(x_i, x_i') = \begin{cases} (x_i' - x_i)^2, & \text{if } x_i \text{ is continuous,} \\ -x_i \log(x_i'), & \text{if } x_i \text{ is binary.} \end{cases}$$

As can be seen from their definitions, $\mathcal{L}_G$ will apply to the missing components ($m_i = 0$) and $\mathcal{L}_M$ will apply to the observed components ($m_i = 1$).

$\mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}})$ is smaller when $\hat{m}_i$ is closer to 1 for i such that $m_i = 0$. That is, $\mathcal{L}_G(\mathbf{m}, \hat{\mathbf{m}})$ is smaller when D is less able to identify the imputed values as being imputed (it falsely categorizes them as observed). $\mathcal{L}_M(\mathbf{x}, \tilde{\mathbf{x}})$ is minimized when the reconstructed features (i.e. the values G outputs for features that were observed) are close to the actually observed features.

G is then trained to minimize the weighted sum of the two

[4]Details of hyper-parameter selection can be found in the Supplementary Materials.

[5]The index j now corresponds to the j-th sample of the mini-batch, rather than the j-th sample of the entire dataset.

Table 1. Source of gains in GAIN algorithm (Mean ± Std of RMSE (Gain (%)))

Algorithm	Breast	Spam	Letter	Credit	News
GAIN	**.0546 ± .0006**	**.0513± .0016**	**.1198± .0005**	**.1858 ± .0010**	**.1441 ± .0007**
GAIN w/o $\mathcal{L}_G$	.0701 ± .0021 **(22.1%)**	.0676 ± .0029 **(24.1%)**	.1344 ± .0012 **(10.9%)**	.2436 ± .0012 **(23.7%)**	.1612 ± .0024 **(10.6%)**
GAIN w/o $\mathcal{L}_M$	.0767 ± .0015 **(28.9%)**	.0672 ± .0036 **(23.7%)**	.1586 ± .0024 **(24.4%)**	.2533 ± .0048 **(26.7%)**	.2522 ± .0042 **(42.9%)**
GAIN w/o Hint	.0639 ± .0018 **(14.6%)**	.0582 ± .0008 **(11.9%)**	.1249 ± .0011 **(4.1%)**	.2173 ± .0052 **(14.5%)**	.1521 ± .0008 **(5.3%)**
GAIN w/o Hint & $\mathcal{L}_M$	.0782 ± .0016 **(30.1%)**	.0700 ± .0064 **(26.7%)**	.1671 ± .0052 **(28.3%)**	.2789 ± .0071 **(33.4%)**	.2527 ± .0052 **(43.0%)**

losses as follows:

$$\min_G \sum_{j=1}^{k_G} \mathcal{L}_G(\mathbf{m}(j), \hat{\mathbf{m}}(j), \mathbf{b}(j)) + \alpha \mathcal{L}_M(\tilde{\mathbf{x}}(j), \hat{\mathbf{x}}(j)),$$

where α is a hyper-parameter.

The pseudo-code is presented in Algorithm 1.

6. Experiments

In this section, we validate the performance of GAIN using multiple real-world datasets. In the first set of experiments we qualitatively analyze the properties of GAIN. In the second we quantitatively evaluate the imputation performance of GAIN using various UCI datasets (Lichman, 2013), giving comparisons with state-of-the-art imputation methods. In the third we evaluate the performance of GAIN in various settings (such as on datasets with different missing rates). In the final set of experiments we evaluate GAIN against other imputation algorithms when the goal is to perform prediction on the imputed dataset.

We conduct each experiment 10 times and within each experiment we use 5-cross validations. We report either RMSE or AUROC as the performance metric along with their standard deviations across the 10 experiments. Unless otherwise stated, missingness is applied to the datasets by randomly removing 20% of all data points (MCAR).

6.1. Source of gain

The potential sources of gain for the GAIN framework are: the use of a GAN-like architecture (through $\mathcal{L}_G$), the use of reconstruction error in the loss ($\mathcal{L}_M$), and the use of the hint ($\mathbf{H}$). In order to understand how each of these affects the performance of GAIN, we exclude one or two of them and compare the performances of the resulting architectures against the full GAIN architecture.

Table 1 shows that the performance of GAIN is improved when all three components are included. More specifically, the full GAIN framework has a 15% improvement over the simple auto-encoder model (i.e. GAIN w/o $\mathcal{L}_G$). Furthermore, utilizing the hint vector additionally gives improvements of 10%.

6.2. Quantitative analysis of GAIN

We use five real-world datasets from UCI Machine Learning Repository (Lichman, 2013) (Breast, Spam, Letter, Credit, and News) to quantitatively evaluate the imputation performance of GAIN. Details of each dataset can be found in the Supplementary Materials.

In table 2 we report the RMSE (and its standard deviation) for GAIN and 5 other state-of-the-art imputation methods: MICE (Buuren & Oudshoorn, 2000; Buuren & Groothuis-Oudshoorn, 2011), MissForest (Stekhoven & Bühlmann, 2011), Matrix completion (Matrix) (Mazumder et al., 2010a), Auto-encoder (Gondara & Wang, 2017) and Expectation-maximization (EM) (García-Laencina et al., 2010). As can be seen from the table, GAIN significantly outperforms each benchmark. Results for the imputation quality of categorical variables in this experiment are given in the Supplementary Materials.

6.3. GAIN in different settings

To better understand GAIN, we conduct several experiments in which we vary the missing rate, the number of samples, and the number of dimensions using Credit dataset. Fig. 2 shows the performance (RMSE) of GAIN within these different settings in comparison to the two most competi-

Table 2. Imputation performance in terms of RMSE (Average ± Std of RMSE)

Algorithm	Breast	Spam	Letter	Credit	News
GAIN	**.0546 ± .0006**	**.0513± .0016**	**.1198± .0005**	**.1858 ± .0010**	**.1441 ± .0007**
MICE	.0646 ± .0028	.0699 ± .0010	.1537 ± .0006	.2585 ± .0011	.1763 ± .0007
MissForest	.0608 ± .0013	.0553 ± .0013	.1605 ± .0004	.1976 ± .0015	.1623 ± 0.012
Matrix	.0946 ± .0020	.0542 ± .0006	.1442 ± .0006	.2602 ± .0073	.2282 ± .0005
Auto-encoder	.0697 ± .0018	.0670 ± .0030	.1351 ± .0009	.2388 ± .0005	.1667 ± .0014
EM	.0634 ± .0021	.0712 ± .0012	.1563 ± .0012	.2604 ± .0015	.1912 ± .0011

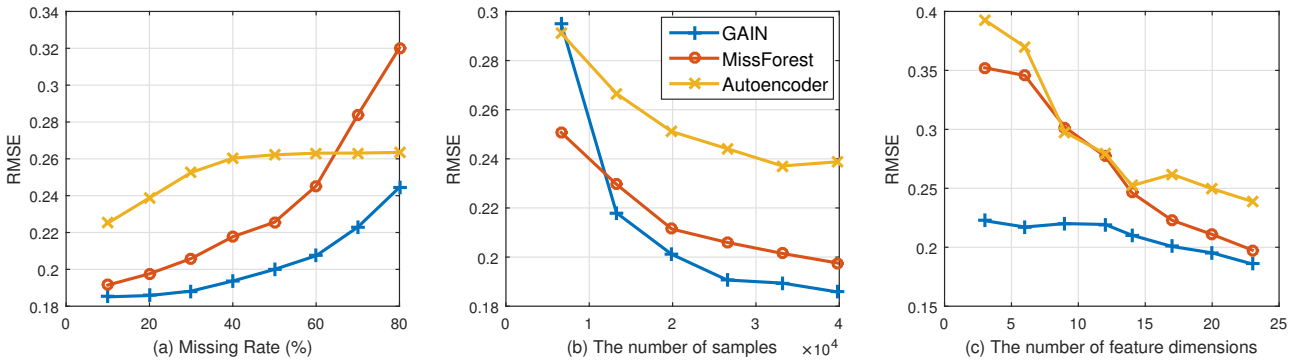

Figure 2. RMSE performance in different settings: (a) Various missing rates, (b) Various number of samples, (c) Various feature dimensions

tive benchmarks (MissForest and Auto-encoder). Fig. 2 (a) shows that, even though the performance of each algorithm decreases as missing rates increase, GAIN consistently outperforms the benchmarks across the entire range of missing rates.

Fig. 2 (b) shows that as the number of samples increases, the performance improvements of GAIN over the benchmarks also increases. This is due to the large number of parameters in GAIN that need to be optimized, however, as demonstrated on the Breast dataset (in Table 2), GAIN is still able to outperform the benchmarks even when the number of samples is relatively small.

Fig. 2 (c) shows that GAIN is also robust to the number of feature dimensions. On the other hand, the discriminative model (MissForest) cannot as easily cope when the number of feature dimensions is small.

6.4. Prediction Performance

We now compare GAIN against the same benchmarks with respect to the accuracy of post-imputation prediction. For this purpose, we use Area Under the Receiver Operating Characteristic Curve (AUROC) as the measure of performance. To be fair to all methods, we use the same predictive model (logistic regression) in all cases.

Comparisons are made on all datasets except Letter (as it has multi-class labels) and the results are reported in Table 3.

As Table 3 shows, GAIN, which we have already shown to achieve the best imputation accuracy (in Table 2), yields the best post-imputation prediction accuracy. However, even in cases where the improvement in imputation accuracy is large, the improvements in prediction accuracy are not always significant. This is probably due to the fact that there is sufficient information in the (80%) observed data to predict the label.

Prediction accuracy with various missing rates: In this experiment, we evaluate the post-imputation prediction performance when the missing rate of the dataset is varied. Note that every dataset (except Letter) has their own binary label.

The results of this experiment (for GAIN and the two most competitive benchmarks) are shown in Fig. 3. In particular, the performance of GAIN is significantly better than the other two for higher missing rates, this is due to the fact that as the information contained in the observed data decreases (due to more values being missing), the imputation quality becomes more important, and GAIN has already been shown to provide (significantly) better quality imputations.

Table 3. Prediction performance comparison

Algorithm	AUROC (Average ± Std)			
	Breast	Spam	Credit	News
GAIN	**.9930 ± .0073**	**.9529 ± .0023**	**.7527 ± .0031**	**.9711 ± .0027**
MICE	.9914 ± .0034	.9495 ± .0031	.7427 ± .0026	.9451 ± .0037
MissForest	.9860 ± .0112	.9520 ± .0061	.7498 ± .0047	.9597 ± .0043
Matrix	.9897 ± .0042	.8639 ± .0055	.7059 ± .0150	.8578 ± .0125
Auto-encoder	.9916 ± .0059	.9403 ± .0051	.7485 ± .0031	.9321 ± .0058
EM	.9899 ± .0147	.9217 ± .0093	.7390 ± .0079	.8987 ± .0157

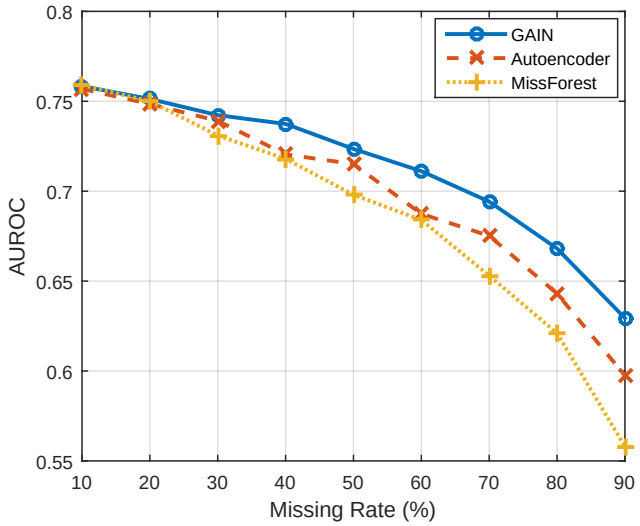

Figure 3. The AUROC performance with various missing rates with Credit dataset

6.5. Congeniality of GAIN

The congeniality of an imputation model is its ability to impute values that respect the feature-label relationship (Meng, 1994; Burgess et al., 2013; Deng et al., 2016). The congeniality of an imputation model can be evaluated by measuring the effects on the feature-label relationships after the imputation. We compare the logistic regression parameters, $\mathbf{w}$, learned from the complete Credit dataset with the parameters, $\hat{\mathbf{w}}$, learned from an incomplete Credit dataset by first imputing and then performing logistic regression.

We report the mean and standard deviation of both the mean bias ($||\mathbf{w} - \hat{\mathbf{w}}||_1$) and the mean square error ($||\mathbf{w} - \hat{\mathbf{w}}||_2$) for each method in Table 4. These quantities being lower indicates that the imputation algorithm better respects the relationship between feature and label. As can be seen in the table, GAIN achieves significantly lower mean bias and mean square error than other state-of-the-art imputation algorithms (from 8.9% to 79.2% performance improvements).

Table 4. Congeniality of imputation models

Algorithm	Mean Bias ($||\mathbf{w} - \hat{\mathbf{w}}||_1$)	MSE ($||\mathbf{w} - \hat{\mathbf{w}}||_2$)
GAIN	**0.3163± 0.0887**	**0.5078± 0.1137**
MICE	0.8315 ± 0.2293	0.9467 ± 0.2083
MissForest	0.6730 ± 0.1937	0.7081 ± 0.1625
Matrix	1.5321 ± 0.0017	1.6660 ± 0.0015
Auto-encoder	0.3500 ± 0.1503	0.5608 ±0.1697
EM	0.8418 ± 0.2675	0.9369 ± 0.2296

7. Conclusion

We propose a generative model for missing data imputation, GAIN. This novel architecture generalizes the well-known GAN such that it can deal with the unique characteristics of the imputation problem. Various experiments with real-world datasets show that GAIN significantly outperforms state-of-the-art imputation techniques. The development of a new, state-of-the-art technique for imputation can have transformative impacts; most datasets in medicine as well as in other domains have missing data. Future work will investigate the performance of GAIN in recommender systems, error concealment as well as in active sensing (Yu et al., 2009). Preliminary results in error concealment using the MNIST dataset (LeCun & Cortes, 2010) can be found in the Supplementary Materials - see Fig. 4 and 5.

Acknowledgement

The authors would like to thank the reviewers for their helpful comments. The research presented in this paper was supported by the Office of Naval Research (ONR) and the NSF (Grant number: ECCS1462245, ECCS1533983, and ECCS1407712).

References

Alaa, A. M., Yoon, J., Hu, S., and van der Schaar, M. Personalized risk scoring for critical care prognosis using mixtures of gaussian processes. *IEEE Transactions on Biomedical Engineering*, 65(1):207–218, 2018.

Allen, A. and Li, W. *Generative Adversarial Denoising Autoencoder for Face Completion*, 2016. URL https://www.cc.gatech.edu/~hays/7476/projects/Avery_Wenchen/.

Barnard, J. and Meng, X.-L. Applications of multiple imputation in medical studies: from aids to nhanes. *Statistical methods in medical research*, 8(1):17–36, 1999.

Burgess, S., White, I. R., Resche-Rigon, M., and Wood, A. M. Combining multiple imputation and meta-analysis with individual participant data. *Statistics in medicine*, 32(26):4499–4514, 2013.

Buuren, S. and Groothuis-Oudshoorn, K. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45(3), 2011.

Buuren, S. v. and Oudshoorn, C. Multivariate imputation by chained equations: Mice v1. 0 user's manual. Technical report, TNO, 2000.

Deng, Y., Chang, C., Ido, M. S., and Long, Q. Multiple imputation for general missing data patterns in the presence of high-dimensional data. *Scientific reports*, 6:21689, 2016.

García-Laencina, P. J., Sancho-Gómez, J.-L., and Figueiras-Vidal, A. R. Pattern classification with missing data: a review. *Neural Computing and Applications*, 19(2): 263–282, 2010.

Gondara, L. and Wang, K. Multiple imputation using deep denoising autoencoders. *arXiv preprint arXiv:1705.02737*, 2017.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural information processing systems*, pp. 2672–2680, 2014.

Kreindler, D. M. and Lumsden, C. J. The effects of the irregular sample and missing data in time series analysis. *Nonlinear Dynamical Systems Analysis for the Behavioral Sciences Using Real Data*, pp. 135, 2012.

LeCun, Y. and Cortes, C. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Lichman, M. UCI machine learning repository, 2013. URL http://archive.ics.uci.edu/ml.

Mackinnon, A. The use and reporting of multiple imputation in medical research–a review. *Journal of internal medicine*, 268(6):586–593, 2010.

Mazumder, R., Hastie, T., and Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug): 2287–2322, 2010a.

Mazumder, R., Hastie, T., and Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research*, 11(Aug): 2287–2322, 2010b.

Meng, X.-L. Multiple-imputation inferences with uncongenial sources of input. *Statistical Science*, pp. 538–558, 1994.

Purwar, A. and Singh, S. K. Hybrid prediction model with missing value imputation for medical data. *Expert Systems with Applications*, 42(13):5621–5631, 2015.

Rubin, D. B. *Multiple imputation for nonresponse in surveys*, volume 81. John Wiley & Sons, 2004.

Schnabel, T., Swaminatan, A., Singh, A., Chandak, N., and Joachims, T. Recommendations as treatments: debiasing learning and evolution. *ICML*, 2016.

Stekhoven, D. J. and Bühlmann, P. Missforestnonparametric missing value imputation for mixed-type data. *Bioinformatics*, 28(1):112–118, 2011.

Sterne, J. A., White, I. R., Carlin, J. B., Spratt, M., Royston, P., Kenward, M. G., Wood, A. M., and Carpenter, J. R. Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls. *BMJ*, 338: b2393, 2009.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International conference on Machine learning*, pp. 1096–1103. ACM, 2008.

Yoon, J., Davtyan, C., and van der Schaar, M. Discovery and clinical decision support for personalized healthcare. *IEEE journal of biomedical and health informatics*, 21 (4):1133–1145, 2017.

Yoon, J., Jordon, J., and van der Schaar, M. GANITE: Estimation of individualized treatment effects using generative adversarial nets. In *International Conference on Learning Representations*, 2018a. URL https://openreview.net/forum?id=ByKWUeWA-.

Yoon, J., Zame, W. R., Banerjee, A., Cadeiras, M., Alaa, A. M., and van der Schaar, M. Personalized survival predictions via trees of predictors: An application to cardiac transplantation. *PloS one*, 13(3):e0194985, 2018b.

Yoon, J., Zame, W. R., and van der Schaar, M. Deep sensing: Active sensing using multi-directional recurrent neural networks. In *International Conference on Learning Representations*, 2018c. URL https://openreview.net/forum?id=r1SnX5xCb.

Yu, H.-F., Rao, H., and Dhillon, I. S. Temporal regularized matrix factorization for high-dimensional time series prediction. *NIPS*, 2016.

Yu, S., Krishnapuram, B., Rosales, R., and Rao, R. B. Active sensing. In *Artificial Intelligence and Statistics*, pp. 639–646, 2009.