EXPLAINING TIME SERIES BY COUNTERFACTUALS

Anonymous authors
Paper under double-blind review

ABSTRACT

We propose a method to automatically compute the importance of features at every observation in time series, by simulating counterfactual trajectories given previous observations. We define the importance of each observation as the change in the model output caused by replacing the observation with a generated one. Our method can be applied to arbitrarily complex time series models. We compare the generated feature importance to existing methods like sensitivity analyses, feature occlusion, and other explanation baselines to show that our approach generates more precise explanations and is less sensitive to noise in the input signals.

1 INTRODUCTION

Time series data are ubiquitous in application domains such as healthcare, finance, and others. Explaining the features that drive the output of time series models is a challenging task due to complex non-linear temporal dependencies and cross-correlations in the data. The explainability problem in ML is significantly exacerbated when complex models like LSTM deep networks (Hochreiter and Schmidhuber, 1997) are used. While explanations for time series models can be obtained in various forms, understanding which features drive (not causally) model output remains an understudied technical challenge. Evaluating globally relevant features has been studied for multivariate time series model (Yang et al., 2005; Yoon et al., 2005; Hmamouche et al., 2017). However, for applications like healthcare, where a patient state is being monitored, individualized (per patient) feature importance at every time point (or each observation) is crucial. Yet statistical methods to learn such importance have received little attention. In this work we propose a method to learn importance of every observation in a multivariate time series model, for individual samples. To the best of our knowledge this is the first individualized feature importance benchmark for time series settings.

Our method provides personalized feature relevance for every sample at each time instance by evaluating the expected change in model prediction had the observation at the time step been different. Our explanation is therefore a counterfactual explanation. We characterize appropriate conditional distributions at every time step, which allow to sample better quality counterfactuals and in turn a statistically sound method of deriving feature importance as explanations. We demonstrate that our explanations have less variance compared to ad-hoc choices of distributions considered in literature so far. Large variance in assigned importance is mainly due to unrealistic or out of distribution counterfactual samples. Using the conditional distribution ensures samples are from the data distribution and also realistic under individual dynamics.

2 FEED-FORWARD COUNTERFACTUALS FOR TIME SERIES EXPLANATION

In this section we describe our method, Feed Forward Counterfactual (FFC), for generating explanation for time series models. A feature is considered important if it affects the model output the most. In time series, the dynamics of the features also change over time, which may impact model outcome. As such it is critical to also identify the precise time points of such changes.

In order to find important observations for high-dimensional time series models, we propose to
We assign importance score to each observation $x_{i,t}$ that is defined as the change in model output when the observation is replaced by a generated counterfactual. The counterfactual observation can come from any distribution, however the quality of the counterfactual random variable directly affects the quality of the explanation. We generate the counterfactual sample conditioned on signal history up to time $t$ by sampling from the distribution $p(x_{i,t}|X_{0:t-1})$. Using a conditional generator guarantees that our counterfactuals are sampled not only within domain but also specifically likely under the individual sample $X^{(n)}$, as opposed to having a generator that models data across population. Conditioning on the history also allows us to learn the dynamics of the signal and therefore generate a plausible counterfactual given the past observations.

$p(x_{i,t}|X_{0:t-1})$ represents the distribution at time $t$, if there were no change in the dynamics of the signals. The counterfactual $\hat{x}_{i,t}$ is sampled from the marginal distribution $p(\hat{x}_{i,t}|X_{0:t-1})$, obtained from $p(x_{i,t}|X_{0:t-1})$. Let $\mathcal{F}(X_{0:t-1}, x_{-i,t}, \hat{x}_{i,t})$ be the output of the model at time $T$, when $x_{i,t}$ is replaced by the generated counterfactual $\hat{x}_{i,t}$. We estimate feature importance $Imp(i,t)$ as $E_{p(\hat{x}_{i,t}|X_{0:t-1})}[|\mathcal{F}(X_{0:t}) - \mathcal{F}(X_{0:t-1}, x_{-i,t}, \hat{x}_{i,t})|]$, summarized in figure 2(b).

2.3 Properties

Our proposed method has the following compelling properties in explaining the estimator $\mathcal{F}$:
We approximate the conditional distribution of $p$. We can thus rank time instances in order of importance. That is, time $t$ we model $x$ as a multivariate Gaussian with full covariance to model all existing correlation between $x$.

Algorithm 1 FFC

**Input:** $F$: Trained Black-box predictor model, $X_{0:T}$, where $T$ is the max time and $S$: Number of Monte-Carlo samples

1: Train $G$
2: for all $t \in [T]$ and $i \in [d]$ do
3:    $y_T = F(X_{0:t})$
4:    $p(x_i|X_{0:t}) \sim G(X_{0:t-1})$
5: for all $s \in [S]$ do
6:    Sample $\hat{x}_{i,t}^{(s)} \sim p(x_{i,t}|X_{0:t-1})$
7:    $\hat{y}_{i}^{(s)} = F(X_{0:t}, x_{-i,t}, \hat{x}_{i,t}^{(s)})$
8:    $\hat{y}_T = \frac{\sum_{s=0}^{S} \hat{y}_{i}^{(s)}}{S}$
9:    $\text{Importance}_\text{Matrix}(i,t) = |y_T - \hat{y}_T|$
10: Return $\text{Importance}_\text{Matrix}$

**Time Importance (TI)** For every time series, highlighting relevant time events for different features is important for actionability of the explanations. For instance in a clinical setting, just knowing a feature like heart rate is relevant, is not sufficient to intervene - it is also important to know when a deterioration had happened. With FFC, the most eventful time instances can be obtained as:

$$\arg \max_{t \in [T]} \{\text{Imp}(i,t) \forall i \in [d]\}$$

We can thus rank time instances in order of importance. That is, time $t_1 \preceq t_2$, if $\max_{i \in [d]} \{\text{Imp}(i, t_1)\} \geq \max_{i \in [d]} \{\text{Imp}(i, t_2)\}$.

**Feature Importance (FI)** At any time instance $t$, our method assigns importance to every feature of $x_{i,t}$.

The magnitude of our importance function reflects relative importance. Comparing the importance values across features gives the flexibility to report a subset of important features at each time point $t$ and also reflects the correlation between various features of the time series.

2.4 Generator Model for Conditional Distribution $p(x_i|X_{0:t-1})$

We approximate the conditional distribution of $p(x_i|X_{0:t-1})$ using a recurrent latent variable generator model $G$, introduced in Chung et al. (2015). The architecture we use is provided in Figure 2(a). The conditional generator $G$ models $p(x_i|z_{t-1})$ where $z_{t-1} \in \mathbb{R}^k$ is the latent representation of history of the time series up to time $t$. The latent representation is a continuous random variable, modeling the distribution parameters. We only use past information in the time series to reflect temporal dependencies. Using the recurrent structure allows us to model a non-stationary generative model that can also handle varying length of observations. Implementation details of the generator are in the Appendix.

Our counterfactuals are not derived by looking at future values which could be done for reliable imputation. Counterfactuals should represent the past dynamics. Note that our derived feature importance is limited by the quality of imputation that may have been utilized by the black-box risk predictor. For experimental evaluation on the effect of generator specifications on counterfactuals and the quality of explanations, see Section 4.1.

2.5 Feature Importance Assignment Algorithm

The proposed procedure is summarized in Algorithm 1. We assume that we are given a trained block box model $F$, and the data (without labels) it had been trained on. Using the training data, we first train the generator that generates the conditional distribution, (denoted by $G$). In our implementation we model $x$ as a multivariate Gaussian with full covariance to model all existing correlation between features. The counterfactual $\hat{x}_{i,t}$ is then sampled from $G$ and passed to the black-box model to evaluate the effect on the black-box outcome.
3 RELATED WORK

A common method of explaining model performance, in time–series deep learning, is via visualization of activations of latent layers (Strobelt et al., 2018; Siddiqui et al., 2019; Ming et al., 2017) or via sensitivity analysis (Bach et al., 2015; Yang et al., 2018). Understanding latent representations, sensitivity and its relationship to overall model behavior is useful for model debugging. However, these but are too refined to be useful to the end users like clinicians.

Attention models (Vaswani et al., 2017; Vinayavekhin et al., 2018; Xu et al., 2018) are the most commonly known explanation mechanisms for sequential data. However, because of the complex mappings to latent space in recurrent models, attention weights cannot be directly attributed to individual observations of the time series (Guo et al., 2018). To resolve this issue to some extent, Choi et al. (2016) propose an attention model for mortality prediction of ICU patients based on clinical visits. However attention weights may not be consistent as explanations (Jain and Wallace, 2019).

In vision, prior works tackle explainability from the counterfactual perspective, finding regions of the image that affect model prediction the most. Fong and Vedaldi (2017) assumes higher importance for inputs that when replaced by an uninformative reference value, maximally change the classifier output. A criticism to such methods is that they may generate out-of-distribution counterfactuals, leading to unreliable explanations. Chang et al. (2019) address this issue for images using conditional generative models for inpainting regions with realistic counterfactuals.

Evaluating sample based feature importance remains largely unstudied for time series models. While more widely studied for image classification, (Bach et al., 2015; Fong and Vedaldi, 2017) these methods cannot be directly extended to time series models due to complex time-series dynamics. Most efforts in this domain focus on population level feature importance (Tyralis and Papacharalampous, 2017). Suresh et al. (2017) is one of the few methods addressing sample based feature importance and use a method similar to Fong and Vedaldi (2017), called “feature occlusion”. They replace each time series observation by a sample from uniform noise to evaluate its effect on model outcome to attribute feature importance. We argue that carefully choosing the counterfactual selection policy is necessary for derive reliable importances. Specifically, replacing observations with noisy out-of-domain samples can lead to arbitrary changes to model output that are not reflective of systematic behavior in the domain. Even if an observation is sampled from the domain distribution, it does not characterize temporal dynamics and dependencies well enough, potentially highlighting features that only reflect global model behavior, as opposed to sample specific feature importance. We therefore model the data–distribution in order to generate reliable counterfactuals. We demonstrate the implications of the choice of the generator (and hence the counterfactuals) on the quality of explanation.

4 EVALUATION

We evaluate our explainability method for finding important features in time series on 2 simulated datasets and 2 real datasets. Our goal is two-fold a) comparison to existing feature importance baselines in time series and b) evaluating the choice of generators on the quality of counterfactuals and explanations.

We compare to existing feature importance baselines described below:

1. **Feature Occlusion (FO) (Suresh et al., 2017)**: Method introduced in Suresh et al. (2017). This method is an ad-hoc approach for generating counterfactuals. When replacing \( x_{i,t} \) with a random sample from the uniform distribution, the change in model output defines the importance for \( x_{i,t} \).

2. **Augmented feature occlusion (AFO)**: We augment the method introduced in Suresh et al. (2017) by sampling counterfactuals from the bootstrapped distribution over each feature. This avoids generating out-of-distribution samples.

3. **Sensitivity Analysis (SA)**: This method evaluates the sensitivity of the output to every observation, by taking the derivative of \( y_t \) with respect to \( x_{i,t} \), at every time point.

4. **LIME (Ribeiro et al., 2016)**: One of the most commonly used explainability methods that assigns local importance to features. Although LIME does not assign temporal importance, for this baseline, we use LIME at every time point to generate feature importances.
4.1 Simulated Data I

![Figure 3: Two samples showing performance on simulated data. Top row are the original sampled signals (one per subfigure). FFC and AFO assign importance at the time of spike. Additional samples are in Appendix A.2.](image)

Evaluating the quality of explanations is challenging due to the lack of a gold standard/ground truth for the explanations. Additionally, explanations are reflective of model behavior, therefore such evaluations are tightly linked to the reliability of the model itself. Therefore we created the simulated environment in order to test our method.

In this experiment, we simulate a time series data such that only one feature determines the outcome. Specifically, the outcome (label) changes to 1 as soon as a spike is observed in the relevant feature. We keep the task fairly simple for two main reasons: 1) to ensure that the black-box classifier can indeed learn the right relation between the important feature and the outcome, which allows us to focus on evaluating the quality of the explanations without worrying about the quality of the classifier. 2) to have a gold standard for the explanations since the exact important event predictive of the outcome are known. We expect the explanations to assign importance only to the one relevant feature, at the exact time of spike, even in the presence of spikes in other non-relevant features.

To simulate these data, we generate $d = 3$ (independent) sequences as a standard non-linear auto-regressive moving average (NARMA) time series of the form: $x(t + 1) = 0.5x(t) + 0.5x(t)\sum_{l=0}^{t-1} x(t - l) + 1.5u(t - (t - 1))u(t) + 0.5$ for $t \in [80]$, where the order is 2 and $u \sim \text{Normal}(0, 0.01)$. We add linear trends to the features and introduce random spikes over time for every feature. Note that since spikes are not correlated over time, no of the generators (used in FFC, AFO, FO) will learn to predict it. The important feature in this setting is feature 1. The complete procedure is described in Appendix A.3.1. We train an RNN-based black-box model on this data, resulting in $\text{AUC} = 0.99$ on the test set.

Figure 8 demonstrates explanations of each of the compared approaches on simulated data for 2 test samples. As shown in Figure 8(a), Sensitivity analysis does not pick up on the importance of the spike. Feature occlusion gives false positive importance to spikes that happen in non-important signals as well as the important one. Augmented feature occlusion resolves this problem since it samples the counterfactuals from the data distribution, however, it generates noisier results as it samples from the bootstrap distribution. The proposed method (FFC) only assigns importance to the first feature at the time of spike. Hence, FFC generates fewer false relevance scores.

Note that all baseline methods provide an importance for every sample at every time point. The true explanation should highlight feature 1 at time points of spike. Using this ground truth, we evaluate the AUROC and AUPRC of the generated explanations denoted by (exp). Table 1 summarizes the results for simulated data.

4.2 Simulated Data II

The first simulation does not necessarily evaluate feature importance under complex state dynamics as is common in applications. In this simulation, we create a dataset with complex dynamics with
known ground truth explanations. The dataset consists of multivariate time series signals with 3 features. A Hidden Markov Model with 2 latent states, with linear transition matrix and multivariate normal emission probabilities is used to simulate observations. The the outcome $y$ is a random variable, which, in state 1, is only affected by feature and in state 2, only affected by feature 2.

The ground truth explanation for output at time $T$ is the observation $x_{i,t}$ where $i$ is the feature that drives the output in the current state and $t$ indicates when feature $i$ became important. In a time series setting, a full explanation for outcome at $t = T$ should include the most important feature variable as well as the time point of importance (here state change).

Figure 4 demonstrates assigned importance for a time series sample. The shaded regions indicate the top 5 important observations ($x_{i,t}$) for each method, the color indicating the corresponding feature $i$. AFO, FO and FFC are able to learn the state dynamics and are able to find the important feature of each state. However, the top importance values in AFO and FO do not correspond to the important time points. Only in FFC, the top important observations are indicative of state changes. Table 1 shows the performance compared to ground-truth explanations for this data.

Figure 4: Simulated Data II: The top plot shows the time series signals and the output risk of the black-box model. States are shaded in green and yellow. Plots below show the reported importance of all features over time for baseline methods. Shaded regions in these plots represent the top 5 important observations reported by each method.

4.2.1 Effect of generator specification

As mentioned earlier, the quality of explanations rely on the quality of the counterfactual. In this paper, we propose the conditional generator for generating the counterfactual, which improves over existing methods. We demonstrate this by looking evaluating the log probability of the counterfactual under the true generator distribution $p^*(x_t|X_{0:t-1})$. Results are summarized in Table 1, Column 3.

Since we simulate data using an HMM, we can analytically derive the distribution $p^*(x_{i,t}|X_{0:t-1})$. Details of the derivation are included in Appendix A.3.1.
4.3 MIMIC MORTALITY

Explaining models based on feature importance is critical for clinical settings. Most of clinical data come in form of time series therefore, it is important to find critical time points in patients’ trajectories along with important features. We evaluate our method on a benchmark mortality prediction task based on the Intensive Care Unit (ICU) time series data from MIMIC. The MIMIC-III dataset consists of de-identified EHRs for \(\sim 40,000\) ICU patients at the Beth Israel Deaconess Medical Center, Boston, MA. The dataset has time series measurements such as vitals and lab results over patients ICU stay (Johnson et al., 2016). We use an RNN-based mortality predictor model as a black-box for evaluation. The model is trained on 14,712 adult ICU admissions and reaches a classification AUC of 0.7939(0.007), using 8 vital and 20 lab measurements, as well as patient static data. More details on the model and data used are in Appendix A.4.

Following the procedure in Algorithm 1, we train a conditional generators for non-static time series features. We compare results across all four existing methods by visualizing importance scores over time. Figure 5 shows an example trajectory of a patient and the predicted outcome. We plot the importance score over time for top 3 signals, selected by each method. Shaded regions in bottom four rows indicate the most important observations, color representing the feature.

As shown in Figure 5, counterfactual based methods mostly agree and pick the same signals as important features. We further evaluate this by looking into accordance scores among methods, indicating the percentage of agreement. This analysis is provided in the Appendix A.4, and the heat map in Figure 11 demonstrates the average score across test samples. However, the methods don’t agree on the exact time importance. As we observe in Figure 5 and other patient trajectories, FFC assigns importance to observations at the precise times of signal change. This is exactly as expected from the method. The FFC counterfactual is conditioned on patient history and thus the counterfactual represents an observation assuming a patient had not change state.

![Figure 5: The top row shows the output risk of the prediction model (black dashed line), and normalized time series signals. Shaded regions in the top row represent specific clinical interventions. The four bottom rows correspond to importance scores generated by each of the methods. The error bars indicate the standard deviation of the importance value. Hatched regions in bottom four rows indicate the top most important observations.](figure5.png)

4.3.1 EVALUATION USING CLINICAL ANNOTATIONS

Since evaluation of explanations can be subjective, we also use intervention information present in patient records to evaluate clinical applicability across baselines. Clinicians intervene with a medication or procedure when there is a notable, adverse change in patient trajectory. Therefore, picking up the most relevant features before an intervention onset is indicative of clinical validity of the method. While we cannot directly associate an intervention with a specific cause (observation),
we look at the overall distribution of signals considered important by each of the methods, prior to intervention onset. Figure 6 shows these histograms for a number of interventions. We see consistent assignment of importance across all methods. This means they associate the same influential signals to the same medical intervention.

Figure 6: Top four features assigned to be important for each intervention across compared methods.

4.4 GREENHOUSE GAS (GHG) OBSERVING NETWORK DATA SET

This experiment evaluates the utility of our method for attributing importance to GHG tracers across different locations in California. The GHG data consists of 15 time series signals from 2921 grid cells. A target time series is a synthetic signal of GHG concentrations. We use an RNN model to estimate GHG concentrations using all tracers. Evaluating which tracers are most useful in reconstructing this synthetic signal can be posed as a feature importance problem for weather modeling over time.

<table>
<thead>
<tr>
<th>Method</th>
<th>MIMIC AUROC -model w/o top 10 (lower is better)</th>
<th>MIMIC AUROC -model w/ top 3 only (higher is better)</th>
<th>GHG MSE -model w/o top 10 (higher is better)</th>
<th>GHG MSE -model w/ top 3 only (lower is better)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FFC</td>
<td>0.7807 (0.0025)</td>
<td>0.7213 (0.0014)</td>
<td>4546.460</td>
<td>4537.188</td>
</tr>
<tr>
<td>AFO</td>
<td>0.7861 (0.0055)</td>
<td>0.7208 (0.002)</td>
<td>4544.163</td>
<td>4532.027</td>
</tr>
<tr>
<td>FO</td>
<td>0.7861 (0.0026)</td>
<td>0.72 (0.0021)</td>
<td>4538.940</td>
<td>4537.044</td>
</tr>
<tr>
<td>SA</td>
<td>0.7788 (0.004)</td>
<td>0.6939 (0.0011)</td>
<td>4532.440</td>
<td>4532.386</td>
</tr>
<tr>
<td>LIME</td>
<td>0.7798 (0.0031)</td>
<td>0.6795 (0.0004)</td>
<td>4536.328</td>
<td>4551.874</td>
</tr>
</tbody>
</table>

Table 2: Real Data - Global Importance.

In order to quantitatively evaluate the proposed method on real data, we evaluate how well the method performs at selecting globally relevant methods as a proxy. We aggregate the importance of all features over time (and training samples) and retrain the black–box by i) removing top 10 relevant features as indicated by each method ii) using top 3 relevant features only. The performance summary is provided in Table 2 suggesting that among methods that derive instance wise feature importance over time, FFC also generates reasonable global relevance of features. Results for both MIMIC-III and GHG datasets are summarized in Table 2.

5 DISCUSSION

We propose a new definition for obtaining sample-based feature importance for high-dimensional time series data. We evaluate the importance of each feature at every time point, to locate highly important observations. We define important observations as those that cause the biggest change in model output had they been different from the actual observation. This counterfactual observation is generated by modeling the conditional distribution of the underlying data dynamics. We propose a generative model to sample such counterfactuals. We evaluate and compare the proposed definition and algorithm to several existing approaches. We show that our method is better at localizing important observations over time. This is one of the first methods that provides individual feature importance over time. Future extension to this work will include analysis on real datasets annotated with feature importance explanations. The method will also be extended to evaluate change in risk based on most relevant subsets of observations.
REFERENCES


A APPENDIX

A.1 NOTATION

The notation used for exposition work is briefly summarized in Table 3.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[K]$ for integer $K$</td>
<td>Set of indices $[K] = {1, 2, \ldots, K}$</td>
</tr>
<tr>
<td>$i, t, n$</td>
<td>Index for feature $i$ in $[d]$, time step $t$ and sample $n$ in $[N]$ respectively</td>
</tr>
<tr>
<td>$-i$</td>
<td>Set $[d] \setminus i$</td>
</tr>
<tr>
<td>Observations and Outcomes</td>
<td></td>
</tr>
<tr>
<td>$x_{i,t}$</td>
<td>Observation $i$ at time $t$.</td>
</tr>
<tr>
<td>$\mathbf{x}_{0:t} \in \mathbb{R}^{d \times t}$</td>
<td>Vector $[x_{1,t}, x_{2,t}, \ldots, x_{d,t}]$.</td>
</tr>
<tr>
<td>$y_t \triangleq \mathcal{F}(\mathbf{x}_{0:t})$</td>
<td>Observed outcome of the black-box model $\mathcal{F}$, at time $t$.</td>
</tr>
<tr>
<td>Generative Model and Estimator</td>
<td></td>
</tr>
<tr>
<td>$\mathcal{G}_i : \mathbb{R}^{d-1 \times m} \rightarrow \mathbb{R}$</td>
<td>Conditional Generative Model sampling for feature $i$.</td>
</tr>
<tr>
<td>$z_t \in \mathbb{R}^m$</td>
<td>Latent encoding of history up to $t$.</td>
</tr>
</tbody>
</table>

Table 3: Notation used in the paper.

A.2 SIMULATED DATA I

To simulate these data, we generate $d = 3$ (independent) sequences as a standard non–linear auto-regressive moving average (NARMA) time series. Note also that we add linear trends to features 1 and 2 of the form: $x(t + 1) = 0.5x(t) + 0.5x(t) \sum_{l=0}^{l-1} x(t - l) + 1.5u(t - (l - 1))u(t) + 0.5 + \alpha_dt$ for $t \in [80]$, $\alpha > 0$ (0.065 for feature 2 and 0.003 for feature 1), and where the order $l = 2$, $u \sim \text{Normal}(0, 0.03)$. We additionally add linear trends to features. We add spikes to each sample (uniformly at random over time) and for every feature $d$ following the procedure below:

$$y_d \sim \text{Bernoulli}(0.5);$$
$$\eta_d = \begin{cases} 
\text{Poisson}(\lambda = 2) & \text{if } 1(y_d == 1) \\
0 & \text{otherwise} 
\end{cases}$$
$$g_d \sim \text{Sample}([T], \eta_d); \ x_{d,t} = x_{d,t} + \kappa \forall t \in g_d$$

where $\kappa > 0$ indicates the additive spike. The label $y_t = 1 \forall t > t_1$, where $t_1 = \min g_d$, i.e. the label changes to 1 when a spike is encountered in the first feature and is 0 otherwise.

We sample our time series using the python TimeSynth\(^1\) package. Number of samples generated: 10000 (80%,20% split).

\(^1\)https://github.com/TimeSynth/TimeSynth
A.2.1 Additional Samples

Figure 7: Two samples showing performance on simulated data I. Top row are the original sampled signals (one per subfigure).

Figure 8: Two samples showing performance on simulated data I. Top row are the original sampled signals (one per subfigure).

A.3 Simulated Data II

This simulated data is a two state HMM (2 states) with initial state \( \pi = [0.5, 0.5] \). Transition probability \( T \) being:

\[
T = \begin{bmatrix}
0.1 & 0.9 \\
0.1 & 0.9 \\
\end{bmatrix}
\]

The emission probability in each state is a multivariate Gaussian: \( \mathcal{N}(\mu_1, \Sigma_1) \) and \( \mathcal{N}(\mu_2, \Sigma_1) \) where \( \mu_1 = [1.2, 1.5, 0.8] \) and \( \mu_2 = [-1.2, -0.8, -1.5] \). Marginal variance for all features in each state is 0.8 with only features 1 and 2 being correlated (\( \Sigma_{12} = \Sigma_{21} = 0.01 \)) in state 1 and only 0 and 2 in state 2 (\( \Sigma_{12} = \Sigma_{20} = 0.01 \)). In state 1, the label \( y \) only depends on feature 1 and in state 2, label depends only on feature 2.

The output \( y_t \) at every step is assigned using the logit in 3. Depending on the hidden state at time \( t \), only one of the features contribute to the output and is deemed influential to the output.

\[
p_t = \begin{cases} 
\frac{1}{1 + e^{-s_t}} & s_t = 0 \\
\frac{1}{1 + e^{-s_t}} & s_t = 1 
\end{cases}
\]

\( y_t \sim \text{Bernoulli}(p_t) \)
A.3.1 Derivation of $p^*(x_{i,t}|X_{0:t-1})$

The true conditional distribution can be derived using the forward algorithm (Bishop, 2006) as follows:

$$p^*(x_{i,t}|X_{0:t-1}) = \sum_{s_t \in \{0,1\}} p(x_{i,t}|s_t)p(s_t|X_{0:t-1})$$ (4)

where,

$$p(s_t|X_{0:t-1}) = \sum_{s_{t-1} \in \{0,1\}} p(s_t|s_{t-1})p(s_{t-1}|X_{0:t-1})$$ (5)

where $p(s_{t-1}|X_{0:t-1})$ is estimated using the forward algorithm.

Joint Conditional Generative Model

Our generator $G_i$ is trained using an RNN (GRU). We model the latent state $z_t$ with a multivariate Gaussian with diagonal covariance and observations with a multivariate Gaussian with full covariance. Parameter setting of the generator for each of the experiments are shown in tables below.

<table>
<thead>
<tr>
<th>Setting</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN cell</td>
<td>GRU</td>
</tr>
<tr>
<td>Loss</td>
<td>MSE</td>
</tr>
<tr>
<td>Optimizer</td>
<td>Adam</td>
</tr>
</tbody>
</table>

Table 4: General generator Setting

Software used: Python 3.7.3, Pytorch 1.0.1.post2

GPU Info: Quadro 400

CPU Info: Intel(R) Xeon(R) CPU E5-1620 v4 @ 3.50GHz

The counterfactual for observation $i$ at time $t$ can now be sampled by marginalizing over other features at time $t$, i.e., $x_{i,t} \sim \sum_{x_{-i}} p(x|X_{0:t-1})$.

A.4 MIMIC-III Mortality Experiment

Feature selection and data processing: For this experiment, we select adult ICU admission data from the MIMIC dataset. We use static patients’ static, vital measurements and lab result for the analysis. The task is to predict 48 hour mortality based on 48 hours of clinical data, therefore we remove samples with less than 48 hours of data. Table 5 presents a full list of clinical measurements used in this experiment.

The predictor model takes in new measurements every hour, and updates the mortality risk. We quantize the time series data to hour blocks by averaging existing measurements within each hour block. We use 2 approaches for imputing missing values: 1) Mean imputation for vital signals using the sklearn SimpleImputer \(^2\), 2) forward imputation for lab results, where we keep the value of the last lab measurement until a new value is evaluated. We also removed patients who had all 48 quantized measurements missing for a specific feature. Overall, 22,988 ICU admissions were extracted and training process was on a 65%, 15%, 20% train, validation, test set respectively.

Parameter Settings for mortality risk predictor model: The risk predictor model is a recurrent network with GRU cells. All features are scaled to 0 mean, unit variance and the target is a probability score ranging $[0, 1]$. The model achieves 0.7939(0.007) AUC on test set classification task. Detailed specification of the model are presented in Table 6.

### Data class

<table>
<thead>
<tr>
<th>Static measurements</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lab measurements</td>
<td>LACTATE, MAGNESIUM, PHOSPHATE, PLATELET, POTASSIUM, PTT, INR, PT, SODIUM, BUN, WBC</td>
</tr>
<tr>
<td>Vital measurements</td>
<td>HeartRate, DiasBP, SysBP, MeanBP, RespRate, SpO2, Glucose, Temp</td>
</tr>
</tbody>
</table>

Table 5: List of clinical features for the risk predictor model

<table>
<thead>
<tr>
<th>Setting</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>epochs</td>
<td>80</td>
</tr>
<tr>
<td>Model</td>
<td>GRU</td>
</tr>
<tr>
<td>batch size</td>
<td>100</td>
</tr>
<tr>
<td>Encoding size ((m))</td>
<td>150</td>
</tr>
<tr>
<td>Loss</td>
<td>MSE</td>
</tr>
<tr>
<td>Regressor Activation</td>
<td>Sigmoid</td>
</tr>
<tr>
<td>Batch Normalization</td>
<td>True</td>
</tr>
<tr>
<td>Dropout</td>
<td>True</td>
</tr>
<tr>
<td>Gradient Algorithm</td>
<td>Adam (learning rate = 0.001, (\beta_1 = 0.9, \beta_2 = 0.999), weight decay = 0)</td>
</tr>
</tbody>
</table>

Table 6: Mortality risk predictor model features

![Risk predictor MSE loss](image1)

(a) Risk predictor MSE loss

![Generator Loss](image2)

(b) Generator Loss

**Parameter Settings for conditional Generator:** The recurrent network with specifications show in 7 learns a hidden latent vector \(h_t\), representing the history. \(h_t\) is then concatenated with \(x_{-i,t}\) and fed into a non-linear 1-layer MLP to model the conditional distribution \(p(x_i, t | X_{0:t-1})\).

<table>
<thead>
<tr>
<th>Setting</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>epochs</td>
<td>150</td>
</tr>
<tr>
<td>RNN cell</td>
<td>GRU</td>
</tr>
<tr>
<td>batch normalization</td>
<td>True</td>
</tr>
<tr>
<td>batch size</td>
<td>100</td>
</tr>
<tr>
<td>RNN encoding size ((m))</td>
<td>80</td>
</tr>
<tr>
<td>Regressor encoding size ((m))</td>
<td>300</td>
</tr>
<tr>
<td>Loss</td>
<td>MSE</td>
</tr>
<tr>
<td>Gradient Algorithm</td>
<td>Adam (learning rate = 0.0001, (\beta_1 = 0.9, \beta_2 = 0.999), weight decay = 0)</td>
</tr>
</tbody>
</table>

Table 7: Training Settings for Feature Generators for MIMIC-III Data

Additional importance plots are provided in Figure 10.
Accordance testing: For this test we look into how much different baselines agree on important feature assignment. As we observed from the experiments, counterfactual methods mostly agree on the most important features for individual samples. We define accordance score between 2 methods as the percentage of top $n$ signals both identified as important. A score of 80 means on average over the test data, 80 of the assignments were similar. This is depicted in Figure 11.

A.5 GHG Network Data

Parameter Settings for Generator: The settings are provided in Table 8.

<table>
<thead>
<tr>
<th>Setting</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>epochs</td>
<td>200</td>
</tr>
<tr>
<td>RNN cell</td>
<td>GRU</td>
</tr>
<tr>
<td>batch size</td>
<td>100</td>
</tr>
<tr>
<td>Encoding size ($m$)</td>
<td>100</td>
</tr>
<tr>
<td>Loss</td>
<td>MSE</td>
</tr>
<tr>
<td>Gradient Algorithm</td>
<td>Adam (learning rate = 0.0001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay = 0)</td>
</tr>
</tbody>
</table>

Table 8: Training Settings for Feature Generators for GHG Data
Figure 11: Heat map showing the accordance score between pairs of methods for the most important 6 signals out of 31 clinical features

Figure 12: Left: Generator Loss for ghg data. Right: (Scaled) Regressor MSE loss

Parameter Settings for Black-Box: This black box regresses $d = 15$ tracer time signals to the target synthetic GHG time series for $t = 327$ time points. This model is trained using a 65%, 15%, 20% train, validation, test set respectively. All features are scaled to 0 mean unit variance and the target is scaled time series is scaled in the range $[-1, 1]$. The regressor is an RNN model with the parameter settings given in Table 9. Figure 12 (a) shows the generator loss for all trained conditional setting value

<table>
<thead>
<tr>
<th>Setting</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>epochs</td>
<td>200</td>
</tr>
<tr>
<td>Model</td>
<td>RNN</td>
</tr>
<tr>
<td>batch size</td>
<td>100</td>
</tr>
<tr>
<td>Encoding size ($m$)</td>
<td>100</td>
</tr>
<tr>
<td>Loss</td>
<td>MSE</td>
</tr>
<tr>
<td>Regressor Activation</td>
<td>Linear</td>
</tr>
<tr>
<td>Gradient Algorithm</td>
<td>Adam (learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay = 0)</td>
</tr>
</tbody>
</table>

Table 9: Training Settings for Regressor for GHG Data

(counterfactual) generators, while Figure 12 shows the training loss of black-box that was used to present feature important results in Section 4.4.