УДК 519.254, 004.048 DOI: 10.14529/cmse240203

ВОССТАНОВЛЕНИЕ МНОГОМЕРНЫХ ВРЕМЕННЫХ РЯДОВ НА ОСНОВЕ ВЫЯВЛЕНИЯ ПОВЕДЕНЧЕСКИХ ШАБЛОНОВ И ПРИМЕНЕНИЯ АВТОЭНКОДЕРОВ

© 2024 А.А. Юртин

Южно-Уральский государственный университет
(454080 Челябинск, пр. им. В.И. Ленина, д. 76)
E-mail: iurtinaa@susu.ru

Поступила в редакцию: 01.05.2024

В настоящее время в широком спектре предметных областей актуальной является задача восстановления пропущенных точек или блоков значений временных рядов. В статье представлен метод SAETI (Snippet-based Autoencoder for Time-series Imputation) для восстановления пропусков в многомерных временных рядах, который основан на совместном применении нейросетевых моделей-автоэнкодеров и аналитического поиска во временном ряде поведенческих шаблонов (сниппетов). Восстановление многомерной подпоследовательности, содержащей пропуски, выполняется посредством двух следующих нейросетевых моделей. Распознаватель получает на вход подпоследовательность, в которой пропуски предварительно заменены на нули, и для каждого измерения определяет соответствующий сниппет. Реконструктор принимает на вход подпоследовательность и набор сниппетов, полученных Распознавателем, и заменяет пропуски на правдоподобные синтетические значения. Реконструктор реализован как совокупность двух следующих моделей: Энкодер, формирующий скрытое состояние для совокупности входной подпоследовательности и распознанных сниппетов; Декодер, получающий на вход скрытое состояние, который восстанавливает исходную подпоследовательность. Представлено детальное описание архитектур вышеперечисленных моделей. Результаты экспериментов над реальными временными рядами из различных предметных областей показывают, что SAETI в среднем опережает передовые аналоги по точности восстановления и показывает лучшие результаты в случае, когда восстанавливаются данные, отражающие активность некоего субъекта.

Ключевые слова: временной ряд, восстановление пропущенных значений, автоэнкодер, поведенческие шаблоны (сниппеты) временного ряда, нейронные сети.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Юртин А.А. Восстановление многомерных временных рядов на основе выявления поведенческих шаблонов и применения автоэнкодеров // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2024. Т. 13, № 2. С. 39–55. DOI: 10.14529/cmse240203.

Введение

В настоящее время в широком спектре приложений возникает задача обработки временных рядов, содержащих пропущенные значения ввиду аппаратно-программных сбоев и человеческого фактора: Интернет вещей [1], управление системами жизнеобеспечения [2], моделирование климата [3] и финансы [4] и др. В подобных приложениях во временных рядах требуется заменить пропуски на синтетические значения, близкие к исходным, чтобы сохранить целостность данных и минимизировать искажения результатов их интеллектуального анализа. Арсенал подходов к решению задачи восстановления пропусков во временных рядах, разработанных научным сообществом, чрезвычайно широк и включает в себя статистические методы [5], аналитические алгоритмы [6, 7] и интенсивно развивающиеся в настоящее время нейросетевые модели [8, 9].

В данной статье представлен новый метод востановления пропущенных значений многомерного временного ряда SAETI (Snippet-based Autoencoder for Time-series Imputation),

основанный на совместном применении поведенческих шаблонов (сниппетов [10]) и нейронных сетей-автоэнкодеров. SAETI включает в себя две последовательно применяемые нейросетевые модели: Распознаватель, определяющий поведенческий шаблон, на который похожа входная подпоследовательность ряда с пропущенными значениями, и Реконструктор, выполняющий восстановление пропусков на основе информации, которая поступила от Распознавателя. Данная работа развивает исследование [11] в следующих аспектах: предложенный метод предназначен для восстановления блоков пропущенных данных временного ряда (а не последнего значения ряда, полученного в режиме реального времени) и применяет автоэнкодеры для реализации Реконструктора, что позволяет эффективно восстанавливать блоки пропущенных данных.

Статья организована следующим образом. Раздел 1 содержит краткий обзор работ по тематике исследования. В разделе 2 приводятся формальные определения понятий и нотация, используемые в статье. Предложенный метод описан в разделе 3. Результаты вычислительных экспериментов, исследующих эффективность предложенного метода, приведены в разделе 4. Заключение содержит сводку полученных результатов и направления будущих исследований.

1. Обзор связанных работ

Разработка моделей, методов и алгоритмов, обеспечивающих как можно более точное восстановление временных рядов из различных предметных областей, является одной из наиболее актуальных задач обработки данных [7]. Для решения указанной задачи на сегодня разработано большое количество как аналитических алгоритмов, так и методов на основе нейросетевых моделей. В настоящее время наиболее эффективными аналитическими алгоритмами признаются [6, 7] следующие разработки: ORBITS [7], DynaMMo [12], CDRec [13], SoftImpute [14] и др. Нейросетевые методы восстановления временных рядов используют широкий спектр современных архитектур нейронных сетей (см., например, обзоры [8, 9]): рекуррентные нейронные сети (например, BRITS [15], M-RNN [16] и др.), генеративно-состязательные сети (например, E^{2}GAN [17], BRNN-GAN [18] и др.), трансформеры с механизмом самовнимания (self-attention; например, SAITS [19], STING [20]), автоэнкодеры и др. Далее кратко рассмотрены методы NAOMI [21] и GP-VAE [22], которые являются типичными представителями класса методов восстановления временных рядов на основе автоэнкодеров и потому наиболее близки по тематике к исследованию, описанному в данной статье.

Автоэнкодеры представляют собой класс нейронных сетей, используемый для нелинейного снижения размерности входных данных. Структуру автоэнкодера можно разделить на две основные части: энкодер (encoder) и декодер (decoder). Энкодер выполняет процесс сжатия входных данных в скрытое состояние (hidden state) меньшей размерности. Скрытое состояние является точкой в скрытом пространстве (hidden space), содержащей важные характеристики входных данных. Скрытое пространство представляет собой множество всех возможных значений скрытого состояния, которые могут быть сгенерированы энкодером. Каждая точка в таком пространстве соответствует определенному входному образцу, преобразованному в скрытое состояние. Декодер выполняет процесс восстановления исходных данных из скрытого состояния. Во время обучения главной задачей модели является минимизация различия между входными данными и данными, восстановленными из скрытого состояния.

При восстановлении пропущенных значений временных рядов типичной является следующая схема применения автоэнкодеров. Во время предварительной обработки пропуски и часть существующих точек заменяются специальными значениями (например, нулями). Автоэнкодер изучает скрытые закономерности в данных, необходимые для качественного сжатия временного ряда, игнорируя шум, сформированный инициализацией пропущенных значений. В процессе обучения ошибка модели вычисляется между восстановленными декодером значениями и существующими точками, искусственно помеченными как пропущенные. Такой подход позволяет модели обучаться кодировать и декодировать входные данные, игнорируя пропуски.

NAOMI (Non-Autoregressive Multiresolution Sequence Imputation) состоит из двух ключевых компонентов: двунаправленного кодировщика (forward-backward encoder), преобразующего существующие точки временного ряда в скрытые состояния, и многоуровневого декодера, осуществляющего восстановление пропущенных значений на основе доступных скрытых состояний других точек. Декодер проводит процесс восстановления рекурсивно, переходя от максимально удаленных друг от друга точек подпоследовательности к соседним.

GP-VAE (Gaussian Process Variational AutoEncoder) [22] представляет собой модель, использующую для восстановления вариационный автоэнкодер (VAE) [23] и моделирование гауссовского процесса (GP) [24]. В отличие от автоэнкодера, VAE основан на вероятностной модели скрытых переменных. VAE кодирует входные данные в скрытое состояние, представленное в виде параметров многомерного распределения (среднее и дисперсия), определенных в скрытом пространстве. Разработчики данной модели предполагают, что эволюцию скрытых переменных входных подпоследовательностей во времени можно представить как гауссовский процесс (вероятностный процесс, в котором каждая конечная комбинация случайных переменных распределена нормально). Результатом моделирования гауссовского процесса является аппроксимация скрытого состояния входных данных. Аппроксимированное скрытое состояние поступает на вход декодера для формирования выхода модели.

2. Основные определения и нотация

Ниже приводятся обозначения и определения терминов, используемых в данной статье, в соответствии с работой [10].

2.1. Временной ряд и подпоследовательность

Одномерный временной ряд представляет собой хронологически упорядоченную последовательность вещественных значений:

$$T = \{t_i\}_{i=1}^{n}, \quad t_i \in \mathbb{R}. \tag{1}$$

Длина временного ряда, n, обозначается как $|T|$.

Подпоследовательность $T_{i,\,m}$ одномерного временного ряда T — это непрерывный промежуток из m элементов ряда, начиная с i-го элемента:

$$T_{i,\,m} = \{t_k\}_{k=i}^{i+m-1}, \quad 1 \leqslant i \leqslant n-m+1, \quad 3 \leqslant m \ll n. \tag{2}$$

Многомерный временной ряд — это набор семантически связанных одномерных временных рядов одинаковой длины, которые синхронизированы во времени. Пусть d обозначает *размерность* многомерного ряда ($d > 1$), количество *измерений* — одномерных рядов в нем.

Подобно одномерному случаю, многомерный временной ряд, его подпоследовательность и отдельные точки обозначим как $\boldsymbol{T}$, $\boldsymbol{T}_{i,m}$ и $\boldsymbol{t}_i$ соответственно, и определим их следующим образом:

$$\boldsymbol{T} = [\{T^{(k)}\}_{k=1}^{d}]^{\intercal}, \tag{3}$$

$$\boldsymbol{T}_{i,m} = [\{T_{i,m}^{(k)}\}_{k=1}^{d}]^{\intercal}, \tag{4}$$

$$\boldsymbol{t}_i = [\{t_i^{(k)}\}_{k=1}^{d}]^{\intercal}. \tag{5}$$

Множество подпоследовательностей многомерного временного ряда будем обозначать как $\boldsymbol{S}_{\boldsymbol{T}}^{m}$:

$$\boldsymbol{S}_{\boldsymbol{T}}^{m} = \{\boldsymbol{T}_{i,m}\}_{i=1}^{n-m+1}. \tag{6}$$

2.2. Сниппеты (поведенческие шаблоны) временного ряда

Сниппеты представляют собой подпоследовательности временного ряда, выражающие типичные активности субъекта, деятельность которого описывает данный ряд, и определяются следующим образом [10].

Для заданной длины подпоследовательности m представим временной ряд T как набор не перекрывающихся подпоследовательностей, *сегментов*. Поскольку $m \ll n$, то без ограничения общности полагаем, что n кратно m, и набор сегментов Seg_T^m определяется следующим образом:

$$Seg_T^m = \{Seg_i\}_{i=1}^{n/m}, \quad Seg_i = T_{m\cdot(i-1)+1,\,m}. \tag{7}$$

Сниппеты T выбираются из Seg_T^m. Введем положительное целое число K ($1 \leqslant K \leqslant n/m$), которое представляет собой ожидаемое количество сниппетов (типичных активностей исследуемого субъекта). Определим, C_T^m, набор сниппетов длины m следующим образом:

$$C_T^m = \{C_i\}_{i=1}^{K}, \quad C_i \in Seg_T^m. \tag{8}$$

Сниппет имеет следующие атрибуты: индекс, набор ближайших соседей и покрытие. Для сниппета $C_i \in C_T^m$ указанные параметры обозначаются как $C_i.index$, $C_i.NN$ и $C_i.frac$ соответственно.

Индекс сниппета представляет собой номер сегмента, которому соответствует сниппет:

$$C_i.index = j \Leftrightarrow Seg_j = T_{m\cdot(j-1)+1,\,m}. \tag{9}$$

Ближайшие соседи сниппета — это набор подпоследовательностей ряда, наименее удаленных от данного сниппета:

$$C_i.NN = \{T_{j,m} \mid Seg_{C_i.index} = \arg\min_{1\leqslant s\leqslant n/m} \mathrm{MPdist}(T_{j,m}, Seg_s), 1 \leqslant j \leqslant n-m+1\}, \tag{10}$$

где $\mathrm{MPdist}(\cdot,\cdot)$ — это функция расстояния, основанная на евклидовой нормированной метрике, предложенная в работе [25]. Покрытие сниппета — это отношение мощности множества его ближайших соседей к общему числу подпоследовательностей ряда соответствующей длины:

$$C_i.frac = \frac{|C_i.NN|}{n-m+1}. \tag{11}$$

В наборе C_T^m сниппеты упорядочиваются в порядке убывания их покрытия:

$$\forall C_i, C_j \in C_T^m : \quad i < j \Leftrightarrow C_i.frac \geqslant C_j.frac. \tag{12}$$

Для многомерного временного ряда $\boldsymbol{T}$ назовем *словарем сниппетов* множество $\boldsymbol{C_T^m}$, объединяющее в себе наборы сниппетов по всем измерениям ряда:

$$\boldsymbol{C_T^m} = \bigcup_{k=1}^{d} C_{T^{(k)}}^m. \tag{13}$$

3. Метод восстановления многомерного временного ряда

3.1. Архитектура

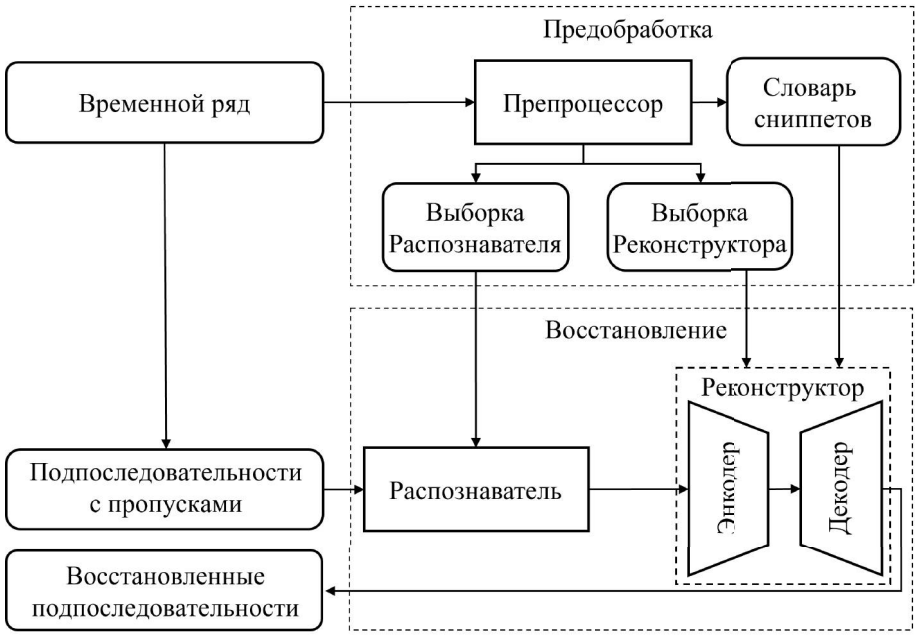


Рис. 1. Метод восстановления многомерного временного ряда

Архитектура предложенного метода представлена на рис. 1 и предполагает подготовку данных и восстановление. На вход поступает многомерный временной ряд, содержащий пропущенные значения. Препроцессор формирует обучающие выборки для нейросетевых моделей метода, Распознавателя и Реконструктора, и словарь сниппетов. Для восстановления подпоследовательности, содержащие пропуски, последовательно обрабатываются Распознавателем и Реконструктором. Распознаватель определяет принадлежность каждого измерения входной многомерной подпоследовательности множеству ближайших соседей сниппетов по данному измерению. Реконструктор, анализируя данные входных подпоследовательностей и сниппеты, заменяет пропуски на синтезированные значения.

3.2. Препроцессор

Препроцессор формирует обучающие выборки Распознавателя и Реконструктора и словарь сниппетов в предположении, что в обрабатываемом временном ряде доля подпоследовательностей, содержащих пропуски, не превышает наперед заданного экспертом в предметной области параметра α, где $0 < \alpha < 1$ и $\alpha = 0.5$ является типичным значением:

$$|\{\boldsymbol{T}_{i,m} \in \boldsymbol{S}_{\boldsymbol{T}}^{m} \mid \exists\, t_j \in T_{i,m}^{(k)},\ t_j = \texttt{NaN}\}| \leqslant \alpha \cdot (n - m + 1). \tag{14}$$

На первом шаге предобработки выполняется минимаксная нормализация каждого измерения временного ряда, приводящая значения к диапазону $[0, 1]$:

$$\hat{t}_i = \frac{t_i - \min\limits_{1 \leqslant k \leqslant n} t_k}{\max\limits_{1 \leqslant k \leqslant n} t_k - \min\limits_{1 \leqslant k \leqslant n} t_k}. \tag{15}$$

Затем Препроцессор выполняет поиск сниппетов в каждом измерении ряда $\boldsymbol{T}$, ограничиваясь при этом подпоследовательностями, которые не содержат пропущенные значения. Указанный поиск отличается от оригинального алгоритма поиска сниппетов SnippetFinder [10] и реализован посредством модификации соответствующей функции библиотеки Matrix Profile API [26], что позволило выполнять вычисление профилей расстояний параллельно. Результатом предварительной обработки является словарь сниппетов $\boldsymbol{C}_{\boldsymbol{T}}^{m}$.

Для формального описания формирования обучающих выборок введем оператор, который заменяет на ноль элементы входной многомерной подпоследовательности следующим образом: сначала обнуляются пропущенные значения, затем обнуляется заданная доля оставшихся значений, выбираемых случайным образом. В формальной записи имеем оператор $\texttt{Zero}_\beta \colon \boldsymbol{S}_{\boldsymbol{T}}^{m} \to \boldsymbol{S}_{\boldsymbol{T}}^{m}$, $0 \leqslant \beta \leqslant 1$:

$$\begin{aligned} &\texttt{Zero}_\beta(\boldsymbol{T}_{i,m}) = \hat{\boldsymbol{T}}_{i,m}, \\ &\forall t_j^{(k)} = \texttt{NaN} \quad \hat{t}_j^{(k)} = 0, \\ &\forall t_j^{(k)} \neq \texttt{NaN} \quad \Pr(\hat{t}_j^{(k)} = 0) = \beta. \end{aligned} \tag{16}$$

Применение введенного оператора при формировании обучающих выборок Распознавателя и Реконструктора позволит указанным моделям в процессе обучения адаптироваться и научиться выделять признаки, необходимые для различения случаев, когда данные содержат нули вследствие нормализации, и случаев, когда пропущенные значения заменены на ноль оператором $\texttt{Zero}_\beta$.

Обозначим обучающую выборку нейронной сети как множество пар $D = \langle X, Y \rangle$, где X представляет собой входные данные, а Y — соответствующие им выходные данные. В качестве входных данных обучающей выборки Распознавателю подаются подпоследовательности ряда $\boldsymbol{T}$, для которых на этапе поиска сниппетов была получена разметка. Подпоследовательности не содержат пропущенных значений, однако в них суммарно $\lfloor \beta \cdot d \cdot m \rfloor$ случайных элементов заменены на ноль. Выходными данными полагаются вектора целочисленных значений, где каждый элемент вектора соответствует номеру сниппета, для которого соответствующая координата подпоследовательности входит в множество ближайших соседей сниппета. В итоге формальное определение обучающей выборки Распознавателя выглядит следующим образом:

$$\begin{aligned} D_{\text{Recognizer}} = \{\langle \boldsymbol{X}, \boldsymbol{Y} \rangle \ \mid\ & \boldsymbol{X} = \texttt{Zero}_\beta(\boldsymbol{T}_{i,m}),\ \neg\exists\, t_j^{(k)} = \texttt{NaN} \\ & \boldsymbol{Y} = [\{s^{(k)}\}_{k=1}^{d}]^{\intercal},\ T_{i,m}^{(k)} \in C_{s^{(k)}}^{(k)}.NN, \\ & 1 \leqslant i \leqslant n - m + 1,\ 1 \leqslant k \leqslant d,\ 1 \leqslant s \leqslant K\}. \end{aligned} \tag{17}$$

Входными данными обучающей выборки Реконструктора полагаются все подпоследовательности ряда, в которых имеют место пропуски и суммарно $\lfloor \beta \cdot d \cdot m \rfloor$ случайных эле-

ментов заменены на ноль. Выходными данными полагаются подпоследовательности ряда, в которых пропуски были заменены на ноль. В итоге обучающая выборка Реконструктора определяется следующим образом:

$$D_{\text{Reconstructor}} = \{\langle \boldsymbol{X}, \boldsymbol{Y}\rangle \mid \boldsymbol{X} = \texttt{Zero}_{\beta}(\boldsymbol{T}_{i,m}), \quad \boldsymbol{Y} = \texttt{Zero}_{\beta\equiv 0}(\boldsymbol{T}_{i,m}), \quad 1 \leqslant i \leqslant n-m+1,\ 1 \leqslant k \leqslant d\}. \tag{18}$$

3.3. Распознаватель

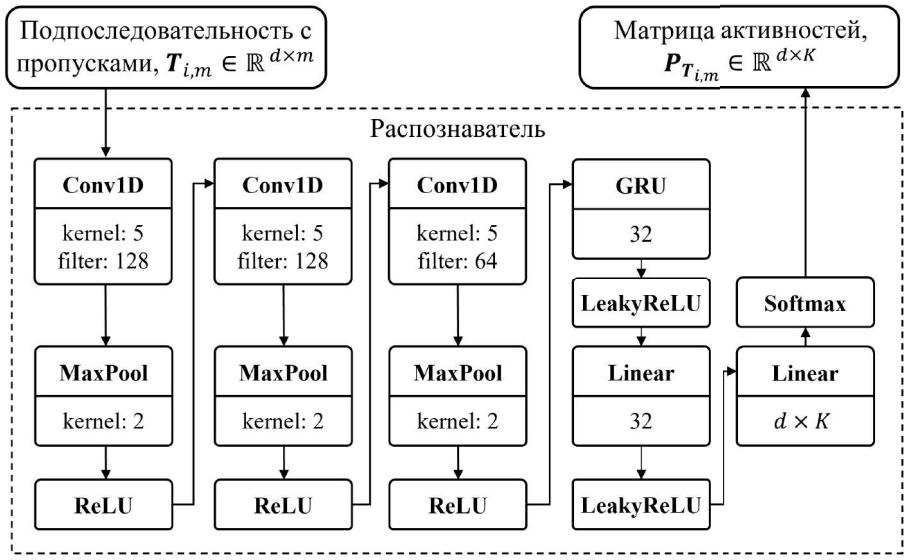


Рис. 2. Структура нейронной сети Распознавателя

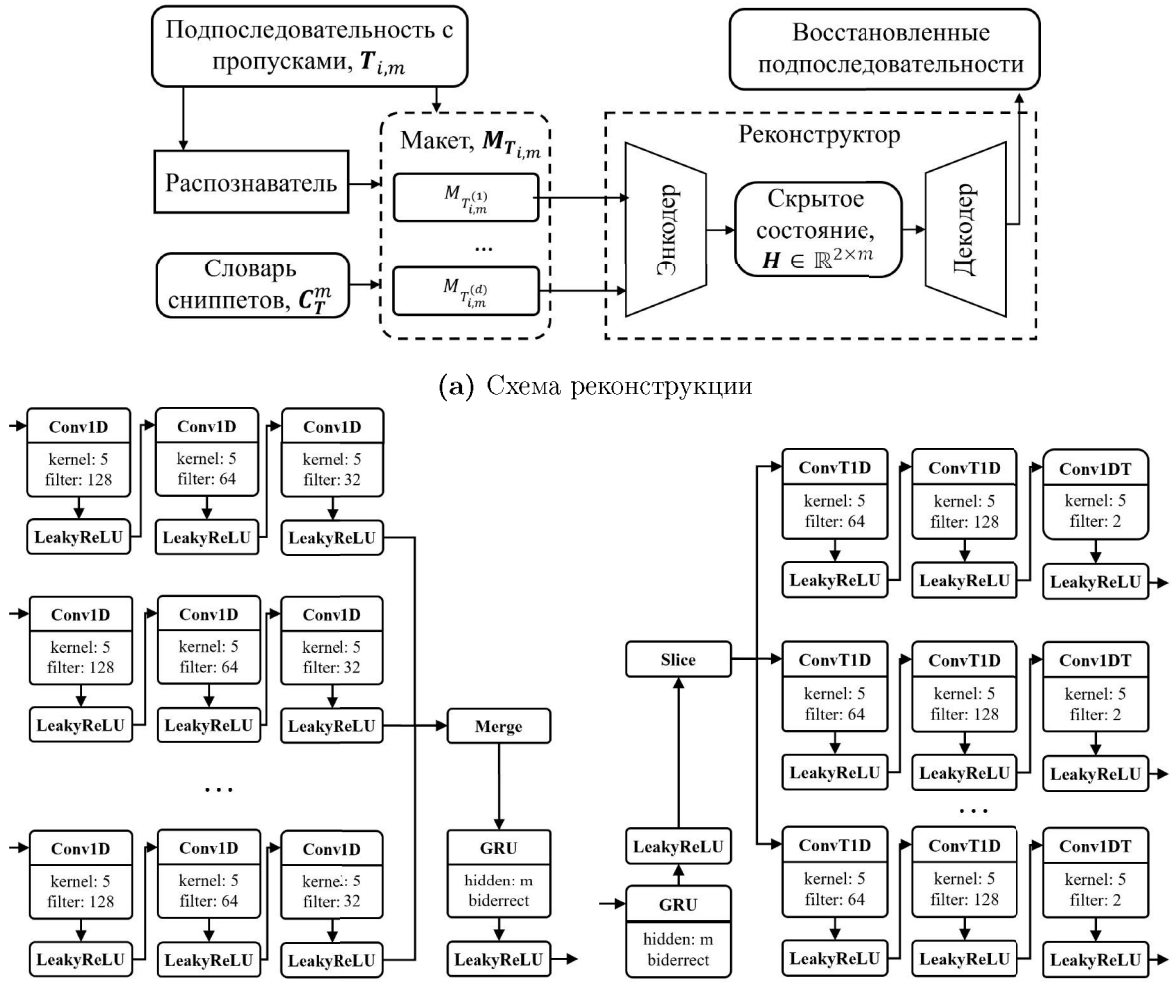
На рис. 2 представлена структура нейронной сети, реализующей Распознаватель. На вход сети поступает многомерная подпоследовательность ряда, пропущенные значения в которой заменены на ноль. На выходе сети для каждого измерения k входной подпоследовательности формируется *вектор активностей* $P_{T^{(k)}_{i,m}} \in \mathbb{R}^K$, состоящий из неотрицательных элементов, сумма которых равна 1. В данном векторе каждый элемент показывает вероятность, с которой одномерная подпоследовательность $T^{(k)}_{i,m}$, соответствующая измерению, принадлежит множеству ближайших соседей сниппета $C^m_{T^{(k)}}$, имеющего соответствующий номер.

Нейронная сеть состоит из следующих последовательно применяемых слоев: трех сверточных, одного рекуррентного и двух полносвязных. Три сверточных слоя, включающие 256, 128 и 64 карт признаков (feature maps) с размером ядра 5 соответственно, отвечают за извлечение признаков из данных входной подпоследовательности. После каждого сверточного слоя в целях уменьшения размерности данных и выделения наиболее важных признаков применяется операция подвыборки по максимальному значению (MaxPooling) с размером ядра 2. В качестве функции активации используется Линейный выпрямитель (ReLU, Rectified linear unit) [27]. Рекуррентный слой анализирует выделенные предыдущими слоями признаки с учетом временного контекста и реализуется с помощью управляемого рекуррентного блока (Gated Recurrent Units, GRU) [28] с длиной вектора скрытого состояния, равной 32. В качестве функции активации этого и всех последующих слоев применяется Линейный выпрямитель с «утечкой» (Leaky ReLU) [29].

Полносвязные слои содержат 32 и $d \cdot K$ нейронов соответственно. Слои, анализируя данные предыдущих слоев, формируют *матрицу активностей* $\boldsymbol{P}_{\boldsymbol{T}_{i,\,m}} \in \mathbb{R}^{d \times K}$, в которой каждая строка представляет собой вектор активностей, рассмотренный выше. Матрица активностей имеет следующее формальное определение:

$$\begin{aligned} &\boldsymbol{P}_{\boldsymbol{T}_{i,\,m}} = [\{P_{T^{(k)}_{i,\,m}}\}_{k=1}^{d}]^{\mathsf{T}}\colon\ P^{(k)}(j) = \Pr(T^{(k)}_{i,\,m} \in C^{m}_{T^{(k)}}), \\ &1 \leqslant i \leqslant n - m + 1,\ 1 \leqslant k \leqslant d,\ 1 \leqslant j \leqslant K. \end{aligned} \tag{19}$$

3.4. Реконструктор



(a) Схема реконструкции

(b) Структура Энкодера

(c) Структура Декодера

Рис. 3. Структура Реконструктора

Рисунок 3 детализирует выполнение реконструкции многомерной подпоследовательности на этапе восстановления ряда. Схема реконструкции (см. рис. 3а) выглядит следующим образом.

Входная подпоследовательность передается на вход предварительно обученного Распознавателя, который выдает матрицу активностей. На основе указанной матрицы и словаря сниппетов, полученного в рамках предобработки, формируется вход Реконструктора, который, в свою очередь, выдает исходную подпоследовательность, где пропуски заменены на синтезированные значения. Данные, поступающие на вход Реконструктора, назовем *маке-*

том и определим как набор матриц, каждая из которых соответствует одному измерению входной многомерной подпоследовательности и состоит из двух строк. Первой строкой матрицы, входящей в макет, является одномерная подпоследовательность, а в качестве второй строки фигурирует соответствующий ей сниппет. В итоге формальное определение заготовки выглядит следующим образом:

$$\begin{aligned} \boldsymbol{M}_{\boldsymbol{T}_{i,\,m}} &= \{M_{I_{i,\,m}^{(k)}}\}_{k=1}^{d},\ M_{I_{i,\,m}^{(k)}} \in \mathbb{R}^{2\times m}, \\ M_{T_{i,\,m}^{(k)}}(1,\cdot) &= \hat{T}_{i,\,m}^{(k)},\ \boldsymbol{M}_{\hat{\boldsymbol{T}}_{i,\,m}} = \texttt{Zero}_{\beta\equiv 0}(\boldsymbol{T}_{i,\,m}), \\ M_{T_{i,\,m}^{(k)}}(2,\cdot) &= C_{T^{(k)}}^{m}\big(\arg\max_{1\leqslant j\leqslant K} P_{T_{i,\,m}^{(k)}}(j)\big),\ 1\leqslant k\leqslant d. \end{aligned} \tag{20}$$

Далее макет поступает на вход Энкодера для формирования скрытого состояния. Скрытое состояние макета представляет собой матрицу $H \in \mathbb{R}^{2\times m}$, в которой первая и вторая строки символизируют результаты обработки Энкодером входных данных, взятых в обычном и инвертированном порядке соответственно. Декодер, используя полученное скрытое состояние, формирует копию входной подпоследовательности, в которой пропущенные значения заменены синтезированными.

Обученный Распознаватель используется для обучения Реконструктора по следующим причинам. Полагается, что таким образом будет сформирована обучающая выборка Реконструктора, позволяющая адаптироваться модели к возможным ошибкам классификации, допускаемым Распознавателем, в процессе работы.

Структура Энкодера представлена на рис. 3b. Энкодер состоит d серий сверточных слоев и одного рекуррентного слоя. Каждая серия сверточных слоев принимает на вход соответствующую матрицу макета. Серия слоев состоит трех сверточных слоев, включающих 64, 32 и 1 карту признаков с размером ядра 5 соответственно. Задача каждой серии сверточных слоев сводится к выделению значимых признаков, важных для качественного сжатия, и связанных со схожестью входной одномерной подпоследовательности и соответствующего ей сниппета. Выходы серий сверточных слоев объединяются в матрицу размером $d \times m$, которая подается на вход рекуррентного слоя. Указанный слой состоит из двунаправленного GRU блока с размером скрытого состояния m. Рекуррентный слой анализирует извлеченные предыдущими слоями признаки, учитывая скрытые временные зависимости между измерениями подпоследовательности, и выдает скрытое состояние подпоследовательности.

Скрытое состояние поступает на вход Декодера (см. рис. 3c), структура которого представляет собой зеркальную копию Энкодера, в котором каждый сверточный слой заменен на соответствующий ему транспонированный сверточный слой. *Транспонированный сверточный слой (transposed convolutional layer)* [30] используется для увеличения пространственного разрешения входных данных. Транспонированная свертка представляет собой операцию, обратную свертке. Такой слой увеличивает размерность данных, вставляя нулевые значения между исходными входными значениями. Затем к получившимся данным применяется свертка, генерируя более крупные выходные данные. Выходы серий транспонированных сверточных слоев по всем измерениям объединяются в матрицу размером $d \times m$, представляющую собой восстановленную многомерную подпоследовательность.

4. Вычислительные эксперименты

Для исследования эффективности предложенного метода были проведены вычислительные эксперименты на оборудовании Лаборатории суперкомпьютерного моделирования

ЮУрГУ [31]. В экспериментах исследовалась точность восстановления предложенного метода в различных предметных областях и сравнивалась с различными передовыми методами восстановления.

4.1. Описание экспериментов

Таблица 1. Наборы данных, используемые в экспериментах

№	Набор	Длина, $n \times 10^3$	Количество измерений, d	Предметная область
Группа А: Сезонность и цикл				
1.	BAFU [32]	50	10	Сброс воды в реках Швейцарии
2.	Climate [33]	5	10	Погода в различных локациях Северной Америки
3.	MAREL [34]	50	10	Характеристики морской воды в Ла-Манше
4.	MeteoSwiss [35]	10	10	Погода в городах Швейцарии
5.	Saaleaue [36]	23	14	Погода в городах Германии
Группа Б: Активности субъекта				
6.	Electricity [37]	5	9	Потребление электроэнергии в нескольких домашних хозяйствах
7.	Madrid [38]	25	10	Трафик автомобильных дорог в Мадриде
8.	Soccer [39]	100	10	Показания носимых датчиков футболистов во время матча
9.	WalkRun [11]	100	11	Показания датчиков смартфона во время прогулки/пробежки

Оценка предложенного метода и его сравнения с аналогами проводилась с использованием временных рядов, резюмированных в табл. 1. Временные ряды разделены на две группы в соответствии с особенностями данных: группа А включает в себя ряды 1–5, которые демонстрируют сезонность и цикл (циклические изменения уровня ряда с постоянным и переменным периодом соответственно); в группу Б входят ряды 6–9, для которых характерно отсутствие сезонности и циклических компонентов ввиду возможности случайного изменения активности субъекта. Целевой группой предложенного в данной работе метода SAETI являются временные ряды группы Б: ожидается, что SAETI покажет более высокую точность восстановления пропусков в тех данных, где могут быть выявлены устойчивые поведенческие шаблоны.

Формирование пропусков многомерного временного ряда осуществлялось в соответствии со сценарием MCAR (Missing Completely at Random, полностью случайное отсутствие), который был предложен в работе [7]. MCAR симулирует многократное отключение на непродолжительное время случайных источников данных. В итоге формируется множество относительно коротких пропусков (несколько подряд идущих точек) в случайных измерениях временного ряда. Для оценки точности восстановления в данной работе используется мера корня из среднеквадратичной ошибки RMSE (Root Mean Square Error) как одна из наиболее часто применяемых для этих целей метрик [40], определяемая следующим образом:

$$\mathrm{RMSE} = \sqrt{\frac{1}{h}\sum_{i=0}^{n}(y_i - \hat{y}_i)^2}, \tag{21}$$

где y_i — фактическое значение, $\hat{y}_i$ — восстановленное значение, h — количество восстановленных точек.

В экспериментах предложенный метод сравнивался с нейросетевыми методами восстановления NAOMI [21], BRITS [15], GP-VAE[22], M-RNN [16], SAITS и TRANSFORM [19] (исходные тексты реализации взяты из свободно доступных репозиториев, созданных авто-

рами данных разработок), а также передовыми аналитическими алгоритмами CDRec [13], DynaMMo [12], ORBITS [6], ROSL [41], GROUSE [42], SoftImpute [14], SVDImpute [43], SVT [44] и TeNMF [45] (исходные тексты реализации взяты из свободно доступного репозитория работы [7]).

Для проведения вычислительных экспериментов были установлены следующие параметры SAETI: размер входной подпоследовательности $m = 200$, количество активностей $K = 2$, доля подпоследовательностей ряда с пропусками $\alpha = 0.5$, доля обнуляемых элементов при подготовке обучающей выборки Реконструктора $\beta = 0.25$.

4.2. Анализ результатов

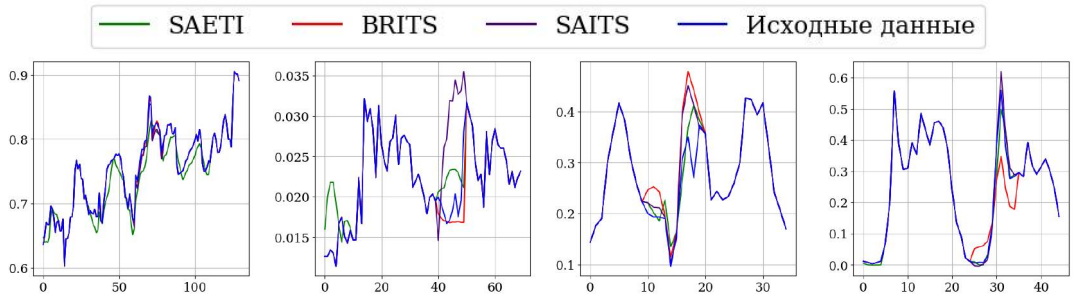
Таблица 2. Точность восстановления, RMSE·10^{-3}

Методы		Наборы данных											
		Группа А					Ср. ошибка	Группа Б				Ср. ошибка	Ср. ошибка
Тип	Название	BAFU	Climate	MAREL	MeteoSwiss	Saaleaue		Electricity	Madrid	Soccer	WalkRun		
Аналитические	CDRec	75(13)	156.2(12)	358(16)	78(9)	119(14)	157(15)	107(13)	103(14)	125(13)	177(15)	128(14)	144(14)
	DynaMMo	39(5)	142(8)	146(6)	73(5)	85.3(7)	97(7)	101(8)	71(6)	79(6)	127(6)	94(6)	96(6)
	GROUSE	206(16)	226(16)	214(13)	191(16)	155(16)	198(16)	134(15)	280(16)	163(15)	163(12)	185(16)	192(16)
	ORBITS	109(15)	166(14)	192(12)	85(11)	104(10)	131(13)	105(10)	90(11)	113.5(12)	174(14)	120.6(13)	126(13)
	ROSL	60(9)	179(15)	168(8)	101(13)	85(6)	119(10)	111(14)	99(13)	105(7)	135(8)	113(10)	116(10)
	SoftImpute	65(11)	152(10)	188(11)	80(10)	100(9)	117(9)	103(9)	76(8)	106(8)	156(10)	110(9)	114(9)
	SVDImpute	62(10)	156(11)	215(14)	77(7)	107(12)	123(12)	105.2(11)	80(10)	110(10)	161(11)	114(11)	119(11)
	SVT	74(12)	143(9)	180(10)	78(8)	75(5)	110(8)	95(7)	80(9)	108(9)	154(9)	109(8)	110(8)
	TeNMF	49(8)	163(13)	216(15)	77(6)	106(11)	122(11)	106(12)	91(12)	113(11)	168(13)	120(12)	121(12)
Нейросетевые	BRITS	17(4)	52(2)	**77(1)**	52(2)	**54(1)**	**50(1)**	**57(1)**	35(2)	20(5)	68(4)	45(2)	48(2)
	GP-VAE	46(7)	**15(1)**	172(9)	129(14)	92(8)	91(6)	74(4)	73(7)	142(14)	123(5)	103(7)	96.3(7)
	M-RNN	88(14)	138(7)	161(7)	169(15)	140(15)	139(14)	180(16)	108(15)	164(16)	227(16)	170(15)	153(15)
	NAOMI	40(6)	76(6)	128(5)	88(12)	107.1(13)	88(5)	75(5)	49(5)	9(2)	134(7)	67(5)	78(5)
	SAITS	15.2(2)	57.8(5)	79(2-3)	65(3-4)	57(2-3)	54.4(4)	73(3)	45(3-4)	14(3)	49(2-3)	45.1(3)	50(3)
	Transformer	**15(1)**	57(4)	79(2-3)	65(3-4)	57(2-3)	54(3)	81(6)	45(3-4)	20(4)	49(2-3)	49(4)	52(4)
	SAETI	16(3)	53(3)	88(4)	**52(1)**	58(4)	53(2)	60(2)	**31(1)**	**6(1)**	**42(1)**	**35(1)**	**45(1)**

Таблица 2 содержит сравнение точности методов восстановления на различных наборах данных. В таблице сравниваемые методы даны построчно и разделены на две группы: аналитические и нейросетевые конкуренты. По столбцам записаны две указанные выше группы наборов данных. В ячейках дается показатель точности и в скобках рейтинг данного результата среди всех конкурентов. Имеются столбцы среднего значения внутри каждой группы и по итогам экспериментов в целом. Лучшие результаты выделяются полужирным шрифтом.

Можно видеть, что в группе А (сезонность и цикл) метод BRITS показывает в среднем лучшую точность как среди аналитических, так и среди нейросетевых аналогов. Тем не менее, вторую позицию в этой группе занимает предложенный в данной работе метод SAETI, правда, несущественно при этом обгоняя методы SAITS и Transformer. Столь высокий результат SAETI в не целевой для себя группы данных, возможно, связан с тем, что в рамках сезона и (или) цикла временного ряда имели место колебания в данных, позволившие выявить существенно отличающиеся между собой поведенческие шаблоны, использование которых привело к увеличению точности восстановления. В целевой группе Б (активности)

предложенный метод уверенно обгоняет всех конкурентов, только в одном наборе данных показывая второй (после BRITS) результат.



(a) Saaleaue **(b)** MAREL **(c)** Electricity **(d)** Madrid

Рис. 4. Пример восстановленных рядов для различных наборов данных

Значимость поведенческих шаблонов проиллюстрирована на рис. 4, где изображены примеры фрагментов временных рядов, восстановленных с помощью SAETI и двух лучших конкурентов, BRITS и SAITS. При восстановлении рядов Saaleaue и MAREL (см. рис. 4a и 4b соответственно), в которых наблюдается возрастающий тренд и отсутствуют явные поведенческие шаблоны, SAETI уступает конкурентам по точности восстановления. Однако при обработке временных рядов Electricity и Madrid (см. рис. 4c и 4d соответственно), метод SAETI успешно использует шаблоны поведения людей в конкретный день недели, существенно обгоняя конкурентов.

Заключение

В данной статье затронута проблема разработки методов восстановления пропущенных значений в многомерных временных рядах, которая является актуальной в широком спектре предметных областей. Предложен новый метод восстановления, названный SAETI (Snippet-based Autoencoder for Time-series Imputation), который основан на совместном применении нейросетевых моделей-автоэнкодеров и аналитического поиска во временном ряде поведенческих шаблонов (сниппетов).

Метод SAETI предполагает фазы подготовки и восстановления данных. Во время первой фазы Препроцессор выполняет поиск сниппетов в каждом измерении входного временного ряда и формирует обучающие выборки нейросетевых моделей. Восстановление подпоследовательности, содержащей пропуски, заключается в последовательном применении к ней двух следующих нейросетевых моделей. Первая модель, Распознаватель, получает на вход подпоследовательность, в которой пропуски предварительно заменены на нули, и для каждого измерения определяет соответствующий сниппет. Распознаватель состоит из следующих последовательно применяемых слоев: трех сверточных, слоя GRU и двух полносвязных слоев. Вторая модель, Реконструктор, принимает на вход подпоследовательность и набор сниппетов, полученных Распознавателем, и заменяет пропуски временного ряда на правдоподобные синтетические значения.

Реконструктор реализован на основе автоэнкодеров, предполагающей два последовательно применяемых компонента: Энкодер и Декодер. Энкодер формирует скрытое состояние входной подпоследовательности и распознанных сниппетов. Декодер, принимая на вход скрытое состояние, производит восстановление исходной подпоследовательности. Энкодер включает в себя следующие слои: d серий сверточных слоев по три слоя в каждой серии и

общий слой GRU, где d — количество измерений. Декодер состоит из слоя GRU и d серий транспонированных сверточных слоев по три слоя в каждой серии.

Описаны вычислительные эксперименты, в которых предложенный метод сравнивался по точности восстановления пропусков с передовыми аналитическими алгоритмами и нейросетевыми моделями на реальных временных рядах из различных предметных областей. Результаты экспериментов показывают, что SAETI в среднем опережает остальных конкурентов и показывает лучшие результаты в случае, когда восстанавливаются данные, отражающие активность некоего субъекта.

В будущих исследованиях планируется изучить влияние способов разбиения временных рядов, содержащих пропущенные значения, при формировании обучающих выборок моделей, на точность восстановления.

Работа выполнена при финансовой поддержке Российского научного фонда (грант № 23-21-00465).

Литература

1. Kumar S., Tiwari P., Zymbler M.L. Internet of Things is a revolutionary approach for future technology enhancement: a review // J. Big Data. 2019. Vol. 6. P. 111. DOI: 10.1186/S40537-019-0268-2.
2. Gratius N., Wang Z., Hwang M.Y., *et al.* Digital Twin Technologies for Autonomous Environmental Control and Life Support Systems // J. Aerosp. Inf. Syst. 2024. Vol. 21, no. 4. P. 332–347. DOI: 10.2514/1.I011320.
3. Zhou Z., Tang W., Li M., *et al.* A Novel Hybrid Intelligent SOPDEL Model with Comprehensive Data Preprocessing for Long-Time-Series Climate Prediction // Remote. Sens. 2023. Vol. 15, no. 7. P. 1951. DOI: 10.3390/RS15071951.
4. Majumdar S., Laha A.K. Clustering and classification of time series using topological data analysis with applications to finance // Expert Syst. Appl. 2020. Vol. 162. P. 113868. DOI: 10.1016/J.ESWA.2020.113868.
5. Yen N.Y., Chang J., Liao J., Yong Y. Analysis of interpolation algorithms for the missing values in IoT time series: a case of air quality in Taiwan // J. Supercomput. 2020. Vol. 76, no. 8. P. 6475–6500. DOI: 10.1007/S11227-019-02991-7.
6. Khayati M., Arous I., Tymchenko Z., Cudré-Mauroux P. ORBITS: Online Recovery of Missing Values in Multiple Time Series Streams // Proc. VLDB Endow. 2020. Vol. 14, no. 3. P. 294–306. DOI: 10.5555/3430915.3442429.
7. Khayati M., Lerner A., Tymchenko Z., Cudré-Mauroux P. Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series // Proc. VLDB Endow. 2020. Vol. 13, no. 5. P. 768–782. DOI: 10.14778/3377369.3377383.
8. Fang C., Wang C. Time Series Data Imputation: A Survey on Deep Learning Approaches // CoRR. 2020. Vol. abs/2011.11347. arXiv: 2011.11347. URL: https://arxiv.org/abs/2011.11347.
9. Wang J., Du W., Cao W., *et al.* Deep Learning for Multivariate Time Series Imputation: A Survey // CoRR. 2024. Vol. abs/2402.04059. DOI: 10.48550/ARXIV.2402.04059. arXiv: 2402.04059.

10. Imani S., Madrid F., Ding W., *et al.* Introducing time series snippets: A new primitive for summarizing long time series // Data Min. Knowl. Discov. 2020. Vol. 34, no. 6. P. 1713–1743. DOI: 10.1007/s10618-020-00702-y.

11. Цымблер М.Л., Юртин А.А. Восстановление пропущенных значений временного ряда на основе совместного применения аналитических алгоритмов и нейронных сетей // Вычислительные методы и программирование. 2023. Т. 24, № 3. С. 243—259. DOI: 10.26089/NumMet.v24r318.

12. Li L., McCann J., Pollard N.S., Faloutsos C. DynaMMo: mining and summarization of coevolving sequences with missing values // Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009 / ed. by J.F.E. IV, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 507–516. DOI: 10.1145/1557019.1557078.

13. Khayati M., Cudré-Mauroux P., Böhlen M.H. Scalable recovery of missing blocks in time series with high and low cross-correlations // Knowl. Inf. Syst. 2020. Vol. 62, no. 6. P. 2257–2280. DOI: 10.1007/S10115-019-01421-7.

14. Mazumder R., Hastie T., Tibshirani R. Spectral Regularization Algorithms for Learning Large Incomplete Matrices // J. Mach. Learn. Res. 2010. Vol. 11. P. 2287–2322. DOI: 10.5555/1756006.1859931.

15. Cao W., Wang D., Li J., *et al.* BRITS: Bidirectional Recurrent Imputation for Time Series // Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada / ed. by S. Bengio, H.M. Wallach, H. Larochelle, *et al.* 2018. P. 6776–6786. URL: https://proceedings.neurips.cc/paper/2018/hash/734e6bfcd358e25ac1db0a4241b95651-Abstract.html.

16. Yoon J., Zame W.R., Schaar M. van der Estimating Missing Data in Temporal Data Streams Using Multi-Directional Recurrent Neural Networks // IEEE Trans. Biomed. Eng. 2019. Vol. 66, no. 5. P. 1477–1490. DOI: 10.1109/TBME.2018.2874712.

17. Luo Y., Zhang Y., Cai X., Yuan X. E^2GAN: End-to-End Generative Adversarial Network for Multivariate Time Series Imputation // Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019 / ed. by S. Kraus. ijcai.org, 2019. P. 3094–3100. DOI: 10.24963/IJCAI.2019/429.

18. Wu Z., Ma C., Shi X., *et al.* BRNN-GAN: Generative Adversarial Networks with Bidirectional Recurrent Neural Networks for Multivariate Time Series Imputation // 27th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2021, Beijing, China, December 14-16, 2021. IEEE, 2021. P. 217–224. DOI: 10.1109/ICPADS53394.2021.00033.

19. Du W., Côté D., Liu Y. SAITS: Self-attention-based imputation for time series // Expert Syst. Appl. 2023. Vol. 219. P. 119619. DOI: 10.1016/J.ESWA.2023.119619.

20. Oh E., Kim T., Ji Y., Khyalia S. STING: Self-attention based Time-series Imputation Networks using GAN // IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, December 7-10, 2021 / ed. by J. Bailey, P. Miettinen, Y.S. Koh, *et al.* IEEE, 2021. P. 1264–1269. DOI: 10.1109/ICDM51629.2021.00155.

21. Liu Y., Yu R., Zheng S., *et al.* NAOMI: Non-Autoregressive Multiresolution Sequence Imputation // Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada / ed. by H.M. Wallach, H. Larochelle, A. Beygelzimer, *et al.* 2019. P. 11236–11246. URL: https://proceedings.neurips.cc/paper/2019/hash/50c1f44e426560f3f2cdcb3e19e39903-Abstract.html.

22. Fortuin V., Baranchuk D., Rätsch G., Mandt S. GP-VAE: Deep Probabilistic Time Series Imputation // The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]. Vol. 108 / ed. by S. Chiappa, R. Calandra. PMLR, 2020. P. 1651–1661. Proceedings of Machine Learning Research. URL: http://proceedings.mlr.press/v108/fortuin20a.html.

23. Kingma D.P., Welling M. Auto-Encoding Variational Bayes // CoRR. 2013. Vol. abs/1312.6114. URL: https://api.semanticscholar.org/CorpusID:216078090.

24. Roberts S.J., Osborne M.A., Ebden M., *et al.* Gaussian processes for time-series modelling // Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 2013. Vol. 371. URL: https://api.semanticscholar.org/CorpusID:556194.

25. Gharghabi S., Imani S., Bagnall A.J., *et al.* An ultra-fast time series distance measure to allow data mining in more complex real-world deployments // Data Min. Knowl. Discov. 2020. Vol. 34, no. 4. P. 1104–1135. DOI: 10.1007/s10618-020-00695-8.

26. Benschoten A.V., Ouyang A., Bischoff F., Marrs T. MPA: a novel cross-language API for time series analysis // Journal of Open Source Software. 2020. Vol. 5, no. 49. P. 2179. DOI: 10.21105/joss.02179.

27. Hochreiter S. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions // Int. J. Uncertain. Fuzziness Knowl. Based Syst. 1998. Vol. 6, no. 2. P. 107–116. DOI: 10.1142/S0218488598000094.

28. Chung J., Gülçehre Ç., Cho K., Bengio Y. Gated Feedback Recurrent Neural Networks // Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015. Vol. 37 / ed. by F.R. Bach, D.M. Blei. JMLR.org, 2015. P. 2067–2075. JMLR Workshop and Conference Proceedings. URL: http://proceedings.mlr.press/v37/chung15.html.

29. Guo Y., Li S., Lerman G. The effect of Leaky ReLUs on the training and generalization of overparameterized networks // International Conference on Artificial Intelligence and Statistics, 2-4 May 2024, Palau de Congressos, Valencia, Spain. Vol. 238 / ed. by S. Dasgupta, S. Mandt, Y. Li. PMLR, 2024. P. 4393–4401. Proceedings of Machine Learning Research. URL: https://proceedings.mlr.press/v238/guo24c.html.

30. Dumoulin V., Visin F. A guide to convolution arithmetic for deep learning // CoRR. 2016. Vol. abs/1603.07285. arXiv: 1603.07285. URL: http://arxiv.org/abs/1603.07285.

31. Биленко Р.В., Долганина Н.Ю., Иванова Е.В., Рекачинский А.И. Высокопроизводительные вычислительные ресурсы Южно-Уральского государственного университет // Вычислительные методы и программирование. 2022. Т. 11, № 1. С. 15—30. DOI: 10.14529/cmse220102.

32. BundesAmt Für Umwelt – Swiss Federal Office for the Environment. Accessed: 2023-09-03. https://www.hydrodaten.admin.ch/.

33. Lozano A.C., Li H., Niculescu-Mizil A., *et al.* Spatial-temporal causal modeling for climate change attribution // Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009 / ed. by J.F.E. IV, F. Fogelman-Soulié, P.A. Flach, M.J. Zaki. ACM, 2009. P. 587–596. DOI: 10.1145/1557019.1557086.

34. Lefebvre A. MAREL Carnot data and metadata from Coriolis Data Centre. SEANOE. 2015. Accessed: 2023-09-03 DOI: 10.17882/39754.

35. MeteoSwiss: Federal Office of Meteorology and Climatology. 2023. Accessed: 2023-09-03. https://www.meteoswiss.admin.ch/services-and-publications/service/open-government-data.html.

36. Weather Station Saaleaue, Max Planck Institute for Biogeochemistry, Germany. Accessed: 2023-09-03. https://www.bgc-jena.mpg.de/wetter/weather_data.html.

37. Trindade A. Electricity Load Diagrams 2011–2014. 2015. DOI: 10.24432/C58C86. UCI Machine Learning Repository.

38. Laña I., Olabarrieta I., Vélez M., Del Ser J. On the imputation of missing data for road traffic forecasting: New insights and novel techniques // Transportation Research Part C: Emerging Technologies. 2018. Vol. 90. P. 18–33. DOI: 10.1016/j.trc.2018.02.021.

39. Mutschler C., Ziekow H., Jerzak Z. The DEBS 2013 grand challenge // The 7th ACM International Conference on Distributed Event-Based Systems, DEBS '13, Arlington, TX, USA, June 29 - July 03, 2013 / ed. by S. Chakravarthy, S.D. Urban, P.R. Pietzuch, E.A. Rundensteiner. ACM, 2013. P. 289–294. DOI: 10.1145/2488222.2488283.

40. Minor B.D., Doppa J.R., Cook D.J. Learning Activity Predictors from Sensor Data: Algorithms, Evaluation, and Applications // IEEE Trans. Knowl. Data Eng. 2017. Vol. 29, no. 12. P. 2744–2757. DOI: 10.1109/TKDE.2017.2750669.

41. Shu X., Porikli F., Ahuja N. Robust Orthonormal Subspace Learning: Efficient Recovery of Corrupted Low-Rank Matrices // 2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014. IEEE Computer Society, 2014. P. 3874–3881. DOI: 10.1109/CVPR.2014.495.

42. Balzano L., Chi Y., Lu Y.M. Streaming PCA and Subspace Tracking: The Missing Data Case // Proc. IEEE. 2018. Vol. 106, no. 8. P. 1293–1310. DOI: 10.1109/JPROC.2018.2847041.

43. Troyanskaya O.G., Cantor M.N., Sherlock G., *et al.* Missing value estimation methods for DNA microarrays // Bioinform. 2001. Vol. 17, no. 6. P. 520–525. DOI: 10.1093/BIOINFORMATICS/17.6.520.

44. Cai J., Candès E.J., Shen Z. A Singular Value Thresholding Algorithm for Matrix Completion // SIAM J. Optim. 2010. Vol. 20, no. 4. P. 1956–1982. DOI: 10.1137/080738970.

45. Mei J., Castro Y. de, Goude Y., Hébrail G. Nonnegative Matrix Factorization for Time Series Recovery From a Few Temporal Aggregates // Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017. Vol. 70 / ed. by D. Precup, Y.W. Teh. PMLR, 2017. P. 2382–2390. Proceedings of Machine Learning Research. URL: http://proceedings.mlr.press/v70/mei17a.html.

Юртин Алексей Артемьевич, программист, Лаборатория больших данных и машинного обучения, аспирант кафедры системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse240203

IMPUTATION OF MULTIVARIATE TIME SERIES BASED ON THE BEHAVIORAL PATTERNS AND AUTOENCODERS

© 2024 A.A. Yurtin

South Ural State University (pr. Lenina 76, Chelyabinsk, 454080 Russia)

E-mail: iurtinaa@susu.ru

Received: 01.05.2024

Currently, in a wide range of subject domains, the problem of imputation missing points or blocks of time series is topical. In the article, we present SAETI (Snippet-based Autoencoder for Time-series Imputation), a novel method for imputation of missing values in multidimensional time series that is based on the combined use of autoencoders and a time series of behavioral patterns (snippets). The imputation of a multidimensional subsequence is performed using the following two neural network models: The Recognizer, which receives a subsequence as input, where the gaps are pre-replaced with zeros, and determines the corresponding snippet for each dimension; and the Reconstructor, which takes as input a subsequence and a set of snippets received from the Recognizer, and replaces the missing elements with plausible synthetic values. The Reconstructor is implemented as a combination of the following two models: An Encoder that forms a hidden state for a set of input sequences and recognized snippets; and a Decoder that receives a hidden state as input, which imputes the original subsequence. In the article, we present a detailed description of the above models. The results of experiments over time series from real-world subject domains showed that SAETI is on average ahead of state-of-the-art analogs in terms of accuracy and shows better results when input time series reflect the activity of a certain subject.

Keywords: time series, imputation of missing values, autoencoders, behavioral patterns (snippets) of time series, neural networks.

FOR CITATION

Yurtin A.A. Imputation of Multivariate Time Series Based on the Behavioral Patterns and Autoencoders. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2024. Vol. 13, no. 2. P. 39–55. (in Russian) DOI: 10.14529/cmse240203.