УДК 004.032.26, 004.048 DOI: 10.14529/cmse210401

ОБ ОДНОМ МЕТОДЕ ВОССТАНОВЛЕНИЯ ПРОПУЩЕННЫХ ЗНАЧЕНИЙ ПОТОКОВОГО ВРЕМЕННОГО РЯДА В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

© 2021 М.Л. Цымблер, В.А. Полонский, А.А. Юртин

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

Поступила в редакцию: 03.09.2021

Проблема восстановления пропущенных значений потокового временного ряда в режиме реального времени возникает в широком спектре практических приложений цифровой индустрии и интернета вещей. В статье предложен новый метод восстановления на основе совместного применения технологий интеллектуального анализа временных рядов и искусственных нейронных сетей. Метод предполагает три этапа восстановления: предварительная обработка данных, распознавание и реконструкция. Предварительная обработка предполагает однократную предварительную подготовку обучающих выборок данных. Распознавание и реконструкция реализуются с помощью нейронных сетей, обучаемых на указанных выборках. Предварительной обработке подвергается заранее сохраненный фрагмент потокового временного ряда без пропусков, в котором выполняется поиск набора типичных подпоследовательностей (сниппетов). Распознавание реализуется с помощью сверточной нейронной сети, на вход которой подается вектор из элементов временного ряда, предшествующих пропуску. Распознаватель выдает сниппет, на который более всего похожа входная подпоследовательность. Реконструкция реализуется с помощью рекуррентной нейронной сети, на вход которой подается конкатенация вывода распознавателя и вектора элементов ряда, предшествующих пропуску. Реконструктор выдает восстановленное значение. Представлены результаты экспериментов, показывающих высокую точность восстановления и преимущество предложенного метода перед аналогами.

Ключевые слова: временной ряд, восстановление пропущенных значений, режим реального времени, сверточная нейронная сеть, рекуррентная нейронная сеть, типичные подпоследовательности.

ОБРАЗЕЦ ЦИТИРОВАНИЯ

Цымблер М.Л., Полонский В.А., Юртин А.А. Об одном методе восстановления пропущенных значений потокового временного ряда в режиме реального времени // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 4. С. 5–25. DOI: 10.14529/cmse210401.

Введение

Временной ряд представляет собой совокупность вещественных значений, взятых в хронологическом порядке. В настоящее время широкий спектр приложений цифровой индустрии и интернета вещей связан с обработкой временных рядов в режиме реального времени: мониторинг технического состояния сложных машин и механизмов [1, 2], интеллектуальное управление системами жизнеобеспечения [3, 4], мониторинг показателей функциональной диагностики организма человека [5], моделирование климата [6], финансовое прогнозирование [7] и др. В силу различных обстоятельств временной ряд может содержать пропущенные значения, например, ввиду сбоя соответствующего датчика или соединительной сети. Однако для указанных выше приложений пропуски недопустимы и должны быть незамедлительно заменены на правдоподобные синтетические значения. В соответствии с этим одной из актуальных задач интеллектуальной обработки данных временных рядов

является создание методов и алгоритмов, которые обеспечивают восстановление пропущенных значений временного ряда в режиме реального времени и являются эффективными в смысле точности и быстродействия восстановления.

В рамках данного исследования нами разработан метод SANNI (Snippet and Neural Network based Imputation), обеспечивающий восстановление пропущенных значений временного ряда в режиме реального времени на основе совместного применения концепции сниппетов и технологий искусственных нейронных сетей. Термин сниппет (snippet) был предложен Кеогом и др. в работе [8] как уточнение понятия типичной подпоследовательности временного ряда. Сниппет неформально может быть определен как подпоследовательность ряда, имеющая заданную длину, на которую похожи многие другие подпоследовательности данного ряда. Набор сниппетов имеет существенно меньшую мощность, чем множество подпоследовательностей ряда, имеющих заданную длину, и потому может использоваться для аннотирования исходного временного ряда.

Статья организована следующим образом. Раздел 1 содержит краткий обзор работ по тематике исследования. В разделе 2 представлен метод SANNI для восстановления пропущенных значений временного ряда в режиме реального времени. В раздел 3 приведены результаты вычислительных экспериментов по оценке эффективности предложенного метода. Заключение резюмирует полученные результаты и описывает направления будущих исследований.

1. Обзор связанных работ

Метод восстановления Hot Deck [9] заменяет отсутствующий элемент временного ряда значением наиболее близкого к нему по времени присутствующего элемента данного ряда. В случае двух одинаково близких по времени элементов берется элемент с меньшим значением индекса. В общем случае данный метод дает низкую точность восстановления.

Восстановление на основе среднего или медианы [10] предполагает замену пропущенных значений на среднее или наиболее часто встречающееся среди всех известных значений во временном ряде соответственно. Данные техники сравнительно просты для реализации, однако, как правило, дают низкую точность восстановления, поскольку для замены всех отсутствующих значений используется одна и та же константа.

Техники восстановления на основе интерполяции (например, линейная и сплайновая интерполяции) [11] конструируют отсутствующее значение на основе значений, непосредственно предшествующих отсутствующему и идущих за ним. Если количество последовательно отсутствующих значений достаточно большое, указанные техники дают плохие результаты восстановления: например, отсутствующий период синусоиды будет заменен прямой.

Регрессионные методы [12] вычисляют отсутствующее значение временного ряда на основе значений в других временных рядах. Например, в работе [13] описано восстановление показаний метеостанции как среднее арифметическое соответствующих показаний других метеостанций, расположенных поблизости. Линейная регрессия [12] основывается на предположении линейной зависимости между значениями временных рядов, что на практике зачастую не имеет места, в силу чего данный метод не подходит для временных рядов с нелинейной зависимостью между ними.

Модель ARIMA (AutoRegressive Integrated Moving Average, интегрированная модель авторегрессии — скользящего среднего) [14] представляет собой обобщение авторегрессионой

модели, используемой в прогнозировании значений временного ряда. В данной модели предполагается линейная зависимость будущего значения временного ряда от значений данного ряда в прошлом. Недостатком этого алгоритма является большое количество параметров p, q, d, нахождение которых представляет сложный и трудоемкий процесс. Модель имеет относительно большое количество параметров (p, q, d), подбор которых представляет собой кропотливую ручную работу, известную как методология Бокса—Дженкинса [14].

Батиста (Batista) и др. в работе [10] предложили алгоритм восстановления значений временного ряда kNNI (k-Nearest Neighbor Imputation), использующий принцип ближайших соседей. В работе исследовалось множество многомерных временных рядов, в котором некоторый ряд T имеет отсутствующие значения в измерении A. Предложенный подход предполагает нахождение k временных рядов, в которых измерения, отличные от A, являются похожими на соответствующие измерения в ряде T.

Троянская (Troyanskaya) и др. в работе [15] модифицировали алгоритм kNNI использованием взвешенных ближайших соседей, где вес соседа прямо пропорционален значению меры схожести с образцом поиска. В качестве меры схожести используется евклидова метрика.

Хаяти (Khayati) и др. предложили алгоритм REBOM (Recovery Blocks of Missing values) [16] который используется для восстановления блоков пропущенных значений в нерегулярном временном ряде (ряде, не имеющем выраженных повторяющихся трендов). Алгоритм выполняет построение матрицы, в которой хранятся значения неполного ряда и k рядов, имеющих наибольшее значение коэффициента корреляции Пирсона с данным рядом. Отсутствующие значения инициализируются с помощью интерполяции. Затем матрица итеративно подвергается сингулярному разложению и наименее значимые сингулярные числа матрицы отбрасываются. Ввиду квадратичной временной сложности алгоритм REBOM плохо масштабируется в случае временных рядов большой длины. Далее авторы в работе [17] представили улучшение REBOM на базе центроидного разложения, которое является приближением сингулярного разложения и обеспечивает линейную пространственную сложность восстановления. Улучшенный алгоритм тоже предполагает линейную корреляцию ряда, имеющего пропуски значений, и рядов-потенциальных соседей, иначе демонстрирует низкую точность восстановления.

Алгоритм MUSCLES предложен Йи (Yi) и др. в работе [12] и выполняет восстановление пропусков в режиме реального времени на основе множественной авторегрессионной модели. Параметры указанной модели обновляются с помощью рекурсивного метода наименьших квадратов (Recursive Least Squares, RLS), чтобы минимизировать ошибку восстановления пропущенного значения. MUSCLES основывается не только на предыдущих значениях неполного ряда, но и на предыдущих и текущих значениях других временных рядов, имеющих линейную корреляцию с данным рядом. Алгоритм плохо масштабируется для большого количества временных рядов, поскольку имеет квадратичные пространственную и временную сложности.

Пападимитроу (Papadimitriou) и др. предложили алгоритм SPIRIT [18]. SPIRIT использует метод главных компонент (Principal Component Analysis, PCA) [19] для сокращения набора совместно развивающихся и коррелированных потоков с небольшим количеством скрытых переменных, которые суммируют основные тенденции во всей коллекции потоков. Каждой скрытой переменной соответствует одна модель авторегрессии для прошлых значений, которые постепенно обновляются по мере поступления новых данных. Если значение

отсутствует, модель авторегрессии используется для прогнозирования текущего значения каждой переменной, из которой выводится оценка пропущенного значения. Восстановленное значение вместе с пропущенными значениями используется для обновления моделей прогнозирования.

Алгоритм TKCM (Top-k Case Matching) [20], предложенный Велленцоном (Wellenzohn) и др., использует набор *связанных временных рядов (reference time series)*, имеющих в силу особенностей данной предметной области семантическую связь с неполным временным рядом. Алгоритм TKCM использует для восстановления отсутствующих значений сходные исторические ситуации в связанных временных рядах. Ситуация описывается *опорной точкой (anchor point)* шаблона, который состоит из ℓ последовательных значений временного ряда, связанного с данным. Пропущенное значение во временном ряде вычисляется из исторических значений этого же ряда в опорных точках k наиболее похожих *шаблонов*. Алгоритм TKCM обеспечивает высокую точность восстановления, если шаблон длины ℓ к моменту времени t_n повторяется как минимум k раз в связанных временных рядах и соответствующие значения данного временного ряда в опорных точках примерно равны друг другу. Алгоритм способен обрабатывать временные ряды, между которыми отсутствует линейная корреляция (например, имеющие фазовые сдвиги) и показывает в среднем более высокую точность восстановления в случае блоков отсутствующих значений, чем REBOM и MUSCLES. Высокая точность восстановления, однако, обеспечивается за счет существенных временных затрат на стадии поиска шаблонов.

Хсу (Hsu) и др. в работе [21] предложили алгоритм восстановления отсутствующих значений, основанный на использовании принципа k ближайших соседей и меры схожести DTW [22]. Алгоритм используется для восстановления отсутствующих значений во временном ряде, который представляет собой значения уровня экспрессии генов, полученные с помощью ДНК-микрочипов в серии экспериментов.

Цао (Cao) и др. разработали модель двунаправленной рекуррентной нейронной сети BRITS [23] для восстановления пропусков в многомерных временных рядах. BRITS рассматривает отсутствующие значения как переменные двунаправленного RNN графа, что делает восстановление более точным из-за получения отложенных градиентов в обоих направлениях.

Гуо (Guo) и др. в работе [24] предложили метод восстановления многомерного временного ряда с помощью генеративно-состязательной нейронной сети (Generative Adversarial Network, GAN). Подход использует операцию многоканальной свертки в GAN при моделировании распределения многомерных рядов, что позволяет увеличить точность восстановления в сравнении с аналогами.

2. Метод восстановления пропусков временного ряда в режиме реального времени

Потоковый временной ряд (streaming time series) представляет собой набор вещественных значений, поступающих последовательно одно за другим в режиме реального времени: $T = (\ldots, t_{n-2}, t_{n-1}, t_n)$, $t_i \in \mathbb{R}$. Мы полагаем, что n-элементное окно ряда T достаточно велико (сотни тысяч элементов) и его данные могут быть размещены в оперативной памяти. Рассматриваемая в данном исследовании задача предполагает, что значение элемента ряда t_n в текущий момент времени пропущено (что обозначается нами как t_n=`NULL`) и вместо него в режиме реального времени должно быть вычислено значение $\tilde{t}_n$.

Ниже приводится детальное описание архитектуры и основных модулей предложенного метода.

2.1. Архитектура метода

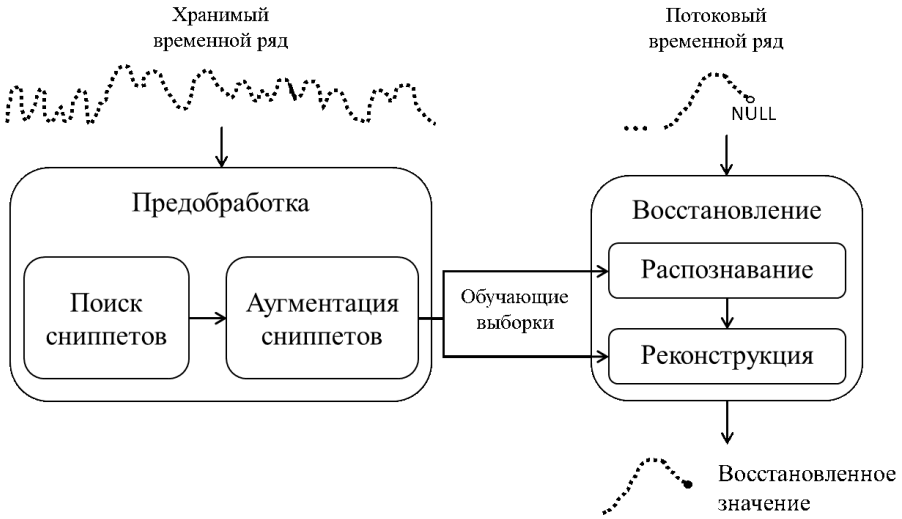

Рис. 1. Архитектура метода SANNI

Метод SANNI предполагает выполнение следующих этапов для восстановления пропущенных значений временного ряда (см. рис. 1): предварительная обработка данных, распознавание и реконструкция. *Предварительная обработка* предполагает однократную предварительную подготовку обучающих выборок данных, обучение на указанных выборках и настройку параметров нейронных сетей, которые реализуют этапы распознавания и реконструкции, выполняющиеся в режиме реального времени. Предварительная обработка данных выполняется регулярно для обновления обучающих выборок и тонкой настройки параметров нейронных сетей.

Для предварительной обработки выбирается заранее сохраненная подпоследовательность исходного потокового временного ряда, которая не содержит пропусков и может быть размещена в оперативной памяти. Мы полагаем, что сохраненная подпоследовательность является достаточно представительной для данной предметной области и охватывает подавляющее большинство потенциально возможных ситуаций при снятии показаний в режиме реального времени. Для удобства дальнейшего изложения и без существенного ограничения общности мы полагаем, что предварительной обработке подвергается временной ряд $T = (t_1, \ldots, t_n)$. Предварительная обработка состоит из следующих последовательно выполняемых шагов: поиск сниппетов и аугментация сниппетов, рассматриваемых далее в разделах 2.2 и 2.3 соответственно.

Распознавание реализуется с помощью сверточной нейронной сети (Convolutional Neural Network, CNN). На вход данной сети подается вектор из m значений, предшествующих текущему элементу t_n=`NULL`, т.е. подпоследовательность $T_{n-m,m}$. Распознаватель выдает сниппет, на который более всего похожа указанная подпоследовательность. Описание методов реализации Распознавателя приведено в разделе 2.4.

Реконструкция реализуется с помощью нейронной сети на основе механизма управляемых рекуррентных блоков (Gated Recurrent Units, GRU). На вход данной сети подается

вектор, представляющий собой конкатенацию двух следующих векторов: сниппет, являющийся результатом этапа распознавания, и вектор из $m-1$ значений, предшествующих текущему элементу t_n=NULL (т.е. подпоследовательность $T_{n-m+1,\,m}$). Реконструктор выдает восстановленное значение $\tilde{t}_n$. Описание реализации Реконструктора приведено в разделе 2.5.

2.2. Поиск сниппетов

Для дальнейшего изложения введем понятие подпоследовательности следующим образом. *Подпоследовательность (subsequence)* $T_{i,m}$ временного ряда T представляет собой непрерывное подмножество T из m элементов, начиная с позиции i: $T_{i,m} = (t_i, \ldots, t_{i+m-1})$, $1 \leqslant m \ll n$, $1 \leqslant i \leqslant m-n+1$.

Концепция *сниппета (snippet)* [8] уточняет понятие типичной подпоследовательности временного ряда следующим образом. Временной ряд T может быть разбит на сегменты заданной длины m, где каждый сегмент S_i представляет собой подпоследовательность ряда $T_{m\cdot(i-1)+1,\,m}$, где $1 \leqslant i \leqslant n/m$. Здесь и далее без существенного ограничения общности мы можем считать, что n кратно m, поскольку $m \ll n$.

Каждый сниппет представляет собой один из сегментов временного ряда. Со сниппетом ассоциируются его ближайшие соседи — подпоследовательности ряда, имеющие ту же длину, что и сниппет, которые более похожи на данный сниппет, чем на другие сегменты. Для вычисления схожести подпоследовательностей используется специализированная мера схожести, основанная на евклидовом расстоянии. Сниппеты упорядочиваются по убыванию мощности множества своих ближайших соседей. Формальное определение сниппетов выглядит следующим образом.

Обозначим множество сниппетов ряда T, имеющих длину $m \ll n$, как C_T^m, а элементы этого множества как $C_1, \ldots, C_K$. Число K $(1 \leqslant K \leqslant n/m)$ представляет собой параметр, задаваемый прикладным программистом, и отражает соответствующее количество наиболее типичных сниппетов. С каждым сниппетом ассоциированы следующие атрибуты: индекс сниппета, ближайшие соседи и значимость данного сниппета.

Индекс сниппета $C_i \in C_T^m$ обозначается как $C_i.index$ и представляет собой номер j сегмента S_j, которому соответствует подпоследовательность ряда $T_{m\cdot(j-1)+1,\,m}$.

Множество ближайших соседей сниппета $C_i \in C_T^m$ обозначается как $C_i.NN$ *(Nearest Neighbors)* и содержит подпоследовательности ряда, которые более похожи на данный сниппет, чем на другие сегменты ряда, в смысле меры схожести MPdist [25]:

$$C_i.NN = \{T_{j,\,m} \mid S_{C_i.index} = \arg\min_{1 \leqslant k \leqslant n/m} \mathrm{MPdist}(T_{j,\,m}, S_k), 1 \leqslant j \leqslant n-m+1\}. \qquad (1)$$

Значимость сниппета $C_i \in C_T^m$ обозначается как $C_i.frac$ представляет собой долю множества ближайших соседей сниппета в общем количестве подпоследовательностей ряда, имеющих длину m:

$$C_i.frac = \frac{|C_i.NN|}{n-m+1}. \qquad (2)$$

Сниппеты упорядочиваются по убыванию их значимости:

$$\forall C_i, C_j \in C_T^m : i < j \Leftrightarrow C_i.frac \geqslant C_j.frac. \qquad (3)$$

Мера MPdist предложена Кеогом и др. в работе [25], и используется для вычисления схожести подпоследовательностей при нахождении сниппетов. MPdist неформально определяется следующим образом. Два временных ряда равной длины m тем более похожи друг на друга в смысле меры MPdist, чем больше в каждом из них имеется подпоследовательностей заданной длины ℓ ($3 \leqslant \ell \leqslant m$), близких друг к другу в смысле евклидова расстояния. Мера MPdist устойчива к выбросам, шумам и пропущенным значениям во временном ряде, а также инвариантна к амплитуде, сдвигу и фазе временного ряда [25]. Отметим также, что MPdist как симметричная неотрицательная функция является мерой, но не метрикой, поскольку для нее выполнены аксиомы тождества и симметрии, но не выполняется аксиома треугольника [25].

2.3. Аугментация сниппетов

Найденные множества ближайших соседей сниппетов являются источником данных выборок для обучения нейронных сетей, реализующих этап распознавания. Мощность указанных множеств может существенно различаться, что влечет за собой дисбаланс прецедентов обучающей выборки, соответствующих различным сниппетам, и низкую точность нейронных сетей, реализующих распознавание восстановление. В соответствии с этим после нахождения сниппетов выполняется *аугментация (искусственное расширение)* множества ближайших соседей каждого малозначимого сниппета.

Введем параметр *порог аугментации* ψ ($0 < \psi < 1$), показывающий минимальную значимость сниппета, ближайшие соседи которого должны быть включены в обучающую выборку Распознавателя. Множество ближайших соседей сниппета C, имеющего значимость $C_i.frac < \psi$, подвергается аугментации. Если сниппет C_i имеет значимость $C_i.frac > \psi$, то взятая случайным образом $(C_i.frac - \psi)$-я часть ближайших соседей данного сниппета не включается в обучающую выборку. В качестве порога аугментации на практике может быть взято значение $C_1.frac$ (значимость наиболее важного сниппета).

Дополним сниппет атрибутом *SynthNN (Synthetic Nearest Neighbors)*, который представляет собой множество синтетических ближайших соседей, используемых для аугментации малозначимых сниппетов. Мощность указанного множества вычисляется следующим образом:

$$|C_i.SynthNN| = \begin{cases} \lceil (\psi - C_i.frac) \cdot (n - m + 1) \rceil, & C_i.frac < \psi \\ 0, & otherwise. \end{cases} \quad (4)$$

Аугментация ближайшего соседа сниппета выполняется таким образом, чтобы каждый полученный синтетический образец был отличен от любого из ближайших соседей сниппета. Это свойство обеспечивается, если потребовать, чтобы каждый синтетический образец был похож на сниппет больше, чем аугментируемый ближайший сосед сниппета. Основная идея аугментации проиллюстрирована на рис. 2.

Обозначим ближайшего соседа сниппета C_i как $neighbor$, величину евклидова расстояния от сниппета до ближайшего соседа обозначим как ε: $\varepsilon = \mathrm{ED}(C_i, neighbor)$. Число ε может быть разложено на сумму m неотрицательных слагаемых в соответствии с алгоритмом, описанным в работе [26]. Указанный алгоритм предполагает, что ε разлагается на k слагаемых, каждое из которых равно ε/k, и число k может принимать значения от 1 до m. Количество разложений равно C_m^{m+k-1} [26].

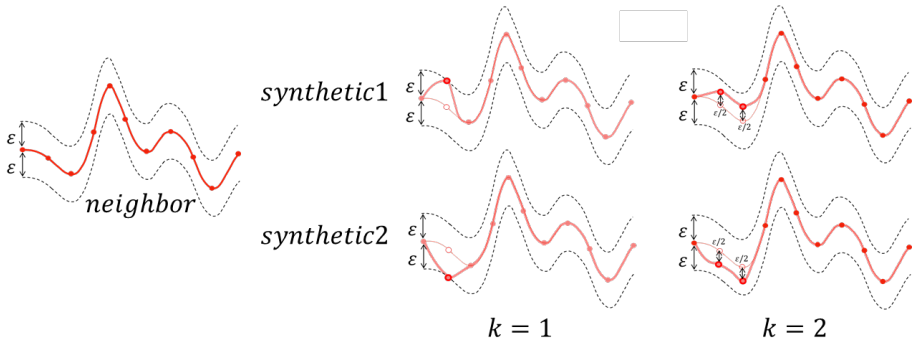

Рис. 2. Аугментация сниппетов

Если мы рассмотрим подпоследовательность *neighbor* и занумерованный набор слагаемых разложения ε как два вектора в евклидовом пространстве $\mathbb{R}^m$, тогда сумма и разность указанных векторов дает пару синтетических соседей S_i, отличных от *neighbor*. Обозначим полученные синтетические образцы как *synthetic*1 и *synthetic*2, тогда приведенные выше рассуждения могут быть формально описаны следующим образом:

$$\begin{gathered} \forall neighbor \in C_i.NN \quad \exists c \in \mathbb{R}^m : \quad \sum_{k=1}^{m} c_k = \varepsilon, \\ synthetic1 = neighbor + c, \quad synthetic2 = neighbor - c, \\ \mathrm{ED}(C_i, synthetic1) < \varepsilon, \quad \mathrm{ED}(C_i, synthetic2) < \varepsilon. \end{gathered} \tag{5}$$

Без существенного ограничения общности мы можем считать, что ближайшие соседи каждого сниппета S_i упорядочены по убыванию схожести с данным сниппетом. Для аугментации возьмем $\lceil \frac{1}{2} |C_i.SynthNN| \rceil$ последних элементов множества ближайших соседей (наименее похожих на сниппет), чтобы обеспечить лучшее обучение нейронной сети распознаванию подпоследовательностей, слабо похожих на сниппет. Вектор $c \in \mathbb{R}^m$, упомянутый в формуле (4), формируется следующим образом. Возьмем случайное натуральное число q $(1 \leqslant q \leqslant m)$, затем возьмем случайные не повторяющиеся натуральные числа $p_1, \ldots, p_q$ $(1 \leqslant p_i \leqslant m)$. Тогда координаты вектора $c \in \mathbb{R}^m$ определяются следующим образом:

$$c \in \mathbb{R}^m, c_i = \begin{cases} \varepsilon/q, & i \in \{p_1, \ldots, p_q\} \\ 0, & otherwise. \end{cases} \tag{6}$$

Описанный выше подход к аугментации сниппетов может быть применен для решения широкого спектра конкретных прикладных задач интеллектуального анализа временных рядов с помощью технологий искусственных нейронных сетей, где требуется синтетическая генерация исходных данных для обучающих выборок.

2.4. Распознавание

В дальнейшем изложении мы будем обозначать обучающую выборку как множество пар $D = \{< X; Y >\}$, где X представляет собой входные данные, а Y — соответствующие им выходные данные.

Кортеж обучающей выборки для нейронной сети Распознавателя формируется следующим образом. Входным данным полагается ближайший сосед сниппета, выходным дан-

ным — номер соответствующего сниппета. Выборка формируется по всем сниппетам и включает в себя как реальных, так и синтетических ближайших соседей сниппета:

$$D_{Recognizer} = \{< x; i > \mid x \in C_i.NN \cup C_i.SynthNN, 1 \leqslant i \leqslant K\}. \quad (7)$$

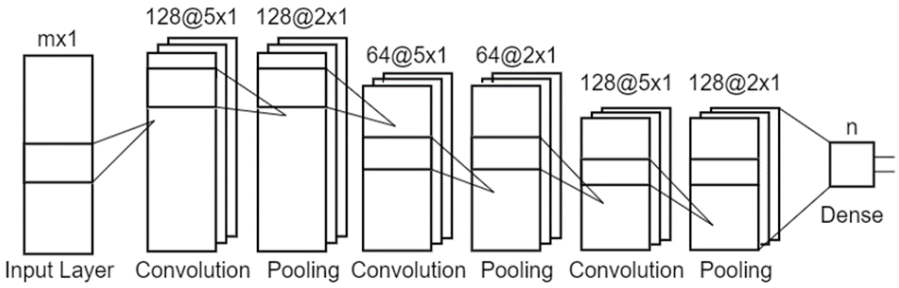

Рис. 3. Топология нейронной сети Распознавателя

На рис. 3 представлена архитектура нейронной сети, реализующей этап распознавания. Нейронная сеть включает в себя три сверточных слоя, каждый из которых чередуется со слоем пулинга на основе усредняющей функции. На каждом слое свертки применяется ядро размера 5×1, на слое пулинга — ядро размера 2×1. Сверточные слои нейронной сети содержат 128, 64 и 128 нейронов соответственно. Для предотвращения переобучения нейронной сети выполняется прореживание (Dropout) 5% нейронов внутренних слоев, за исключением последнего из них, в котором отбрасывается 25% нейронов.

Размерность входного слоя нейронной сети m совпадает с длиной сниппета. На вход сети подается вектор из m значений, предшествующих текущему (восстанавливаемому) элементу временного ряда. Размерность выходного слоя Dense совпадает со значением параметра K (количество наиболее значимых сниппетов), заданным на этапе предварительной обработки. На выходе нейронной сети формируется K-мерный вещественный вектор, i-й элемент которого отражает вероятность похожести входного вектора на сниппет S_i. В качестве результата Распознаватель выдает сниппет с максимальной вероятностью похожести.

2.5. Реконструкция

Кортеж обучающей выборки для нейронной сети Реконструктора формируется следующим образом. Элементом входных данных полагается вектор, который является конкатенацией двух векторов: сниппет и ближайший сосед сниппета, из которого исключен последний элемент. Элементом выходных данных является исключенный элемент вышеуказанного ближайшего соседа. Выборка формируется по всем сниппетам и включает в себя как реальных, так и синтетических ближайших соседей сниппета:

$$D_{Reconstructor} = \{< C_i \cdot (x_1, \ldots, x_{m-1}); x_m > \mid x \in C_i.NN \cup C_i.SynthNN, 1 \leqslant i \leqslant K\}. \quad (8)$$

На рис. 4 представлена архитектура нейронной сети, реализующей Реконструктор. Нейронная сеть включает в себя три слоя: входной слой, скрытый слой и выходной слой. Размерность входного слоя нейронной сети m совпадает с длиной сниппета. Скрытый слой состоит 128 нейронов, каждый из которых представляет собой управляемый рекуррент-

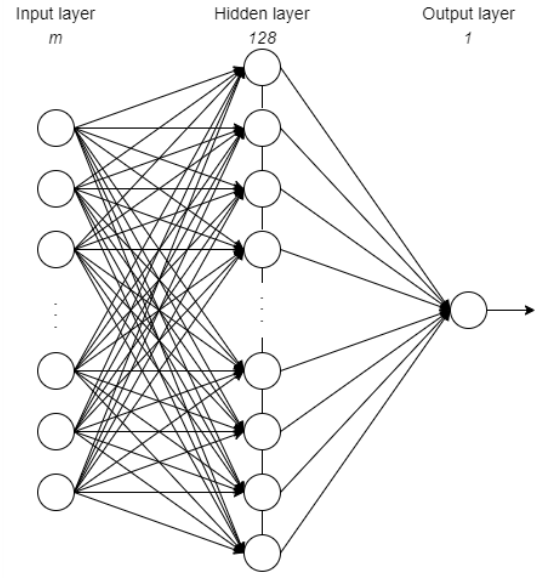

Рис. 4. Топология нейронной сети Реконструктора

ный блок (Gated Recurrent Units, GRU) [27]. Выходной слой состоит из одного нейрона, результатом которого является предсказанное значение $\tilde{t}_n$.

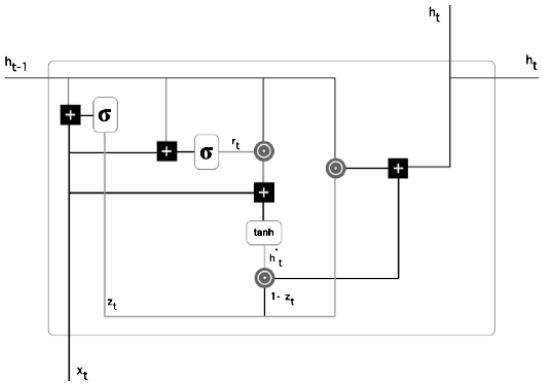

Рис. 5. Структура управляемого рекуррентного блока (операции: $\odot$ — произведение Адамара, $\boxplus$ — сложение; функции: σ — сигмоида, tanh — гиперболический тангенс; векторы (на шаге t): x_t — входной, h_t — скрытое состояние, r_t — сброс, z_t — обновление)

Структура GRU-нейрона представлена на рис. 5. GRU используется для решения проблемы «исчезающего градиента» [28], которая возникает при обучении искусственных нейронных сетей с помощью методов обучения на основе градиента и обратного распространения ошибки. В указанных методах каждый из весов нейронной сети получает обновление, пропорциональное частной производной функции ошибок по текущему весу на каждой итерации обучения. Проблема заключается в том, что градиент может иметь настолько малые значения, что это предотвращает изменение значения веса. В худшем случае это может полностью остановить дальнейшее обучение нейронной сети.

Управляемый рекуррентный блок содержит *вектор сброса* состояний, который позволяет «забыть» информацию, неуместную в будущем:

$$r_t = \sigma(x_t W_{xr} + h_{t-1} W_{hr} + b_r). \tag{9}$$

Для контроля информации, которую необходимо передавать из предыдущего состояния, применяется *вектор обновления*, вычисляемый аналогичным образом:

$$z_t = \sigma(x_t W_{zr} + h_{t-1} W_{hz} + b_z). \tag{10}$$

Далее, исходя из этого, вычисляется *текущее состояние*:

$$\begin{gathered} h_t = z_t \odot h_{t-1} + (1 - z_t) \odot h'_t, \\ h'_t = \tanh(x_t W_{xh} + (r_t \odot h_{t-1}) W_{hh} + b_h). \end{gathered} \tag{11}$$

Таким образом, если вектор сброса принимает значение, близкое к нулю, то нейрон «забывает» информацию о предыдущем состоянии и рассчитывает состояние так, как если бы x_t было бы началом новой подпоследовательности временного ряда. Благодаря наличию векторов сброса и обновления градиент не принимает значения ниже определенной величины, определяемой значением сигнала сохраняемого контента.

3. Вычислительные эксперименты

3.1. Цели и наборы данных

Для оценки эффективности метода SANNI нами были проведены вычислительные эксперименты, в которых исследовалась точность восстановления пропущенных значений во временных рядах из различных предметных областей, в том числе в сравнении с аналогичными методами, а также влияние параметров метода на точность восстановления.

Для оценки точности восстановления пропущенных значений нами используется мера среднеквадратичной ошибки *RMSE (Root Mean Square Error)*, определяемая следующим образом:

$$RMSE = \sqrt{\frac{1}{h} \sum_{h}^{i=1} (t_i - \hat{t}_i)^2}, \tag{12}$$

где t_i — фактическое значение временного ряда, $\hat{t}_i$ — восстановленное значение, h — количество пропущенных значений.

Таблица 1. Временные ряды для экспериментов

№ п/п	Временной ряд	Длина ряда n	Семантика
1	HumanActivity-2	7 002	Различные виды физической активности человека
2	HumanActivity-12	100 800	
3	AppliancesEnergy	100 000	Энергопотребление бытовых приборов в доме в течение 2 месяцев
4	Household-Power	100 000	Энергопотребление домашнего хозяйства в течение 4 лет
5	Household-Voltage	100 000	
6	Household-Kitchen	100 000	

Для проведения экспериментов были использованы следующие наборы данных, резюмированные в табл. 1. Ряды HumanActivity-2 и HumanActivity-12 получены на основе набора данных PAMAP [29], который представляет собой записи о физической активности различ-

ных людей (бег, ходьба, прыжки и др., всего 12 видов активности). Ряд HumanActivity-2 содержит записи о двух активностях (ходьба, сменяющаяся бегом) одного человека. Ряд HumanActivity-12 соединяет в себе записи о 12 активностях одного человека.

Ряд AppliancesEnergy представляет собой показания общего энергопотребления бытовых приборов дома и получен на основе набора данных, использованного в работе [30].

Мультивариативный ряд Household отражает энергопотребление домашнего хозяйства (посудомоечная машина, микроволновая печь, духовка, стиральная машина, водонагреватель, кондиционер и др.) в течение 4 лет [31] и включает в себя следующие ряды: Power — суммарная мощность потребленной энергии в минуту, Voltage — суммарное напряжение потребленной энергии в минуту, Kitchen — общая мощность энергии, потребленной кухонными приборами.

3.2. Сравнение с аналогами

В проведенных экспериментах конкурентами метода SANNI выступали как аналитические алгоритмы, так и методы на основе нейронных сетей. В качестве соперников-аналитических алгоритмов были рассмотрены Mean-, Median-, Mode-imputation (восстановление с помощью среднего, медианы и моды соответственно), ImputeTS (библиотека алгоритмов на языке R для одноименной среды интеллектуального анализа данных, выполняющих восстановление пропусков в одномерных временных рядах) [32] и kNNI[1] [10]. Нейросетевые методы-конкуренты — BRITS [23] и ANNI. Метод ANNI реализован авторами данного исследования как нейронная сеть из GRU-нейронов и представляет собой метод SANNI, из которого исключены поиск и аугментация сниппетов, а также распознавание подпоследовательностей временного ряда на основе сниппетов. Метод BRITS предназначен для восстановления в мультивариативных временных рядах, поэтому в экспериментах с этим методом нами была выполнена 3-кратная репликация рядов HumanActivity и AppliancesEnergy: на вход BRITS подавался мультивариативный ряд из трех одинаковых значений.

Таблица 2. Параметры экспериментов

№ п/п	Временной ряд	Длина сегмента m	Количество сниппетов K
1	HumanActivity-2	100	2
2	HumanActivity-12	300	12
3	AppliancesEnergy	1 440	2
4	Household-Power	60	2
5	Household-Voltage	60	2
6	Household-Kitchen	60	2

В табл. 2 приведены параметры метода SANNI длина сегмента m и количество наиболее значимых сниппетов K, использованные в экспериментах. В экспериментах в качестве порога аугментации (см. раздел 2.3) было взято значение $\psi = 0.01$. Для формирования обучающей и тестовой выборок использовались 70% и 30% случайно выбранных подпоследовательностей исходного временного ряда соответственно.

[1]В экспериментах взято значение параметра $k = 10$, обеспечившее наибольшую точность восстановления.

Таблица 3. Точность восстановления (RMSE $\cdot 10^{-2}$) приведены параметры метода SANNI длина сегмента m и количество наиболее значимых сниппетов K, использованные в экспериментах

Метод	Human Activity-2	Human Activity-12	Appliances Energy	Household		
				Power	Voltage	Kitchen
Аналитические методы						
Mean	10.29	3.85	14.58	13.37	12.24	8.76
Median	10.56	3.92	14.77	13.44	12.28	8.87
Mode	10.56	3.92	21.21	19.86	12.44	8.77
kNNI	3.93	1.88	4.99	5.95	3.23	3.42
imputeTS	11.05	4.16	20.36	18.38	11.04	9.15
Нейросетевые методы						
ANNI	3.53	0.98	**0.69**	**4.36**	**2.60**	**2.20**
BRITS	8.94	9.63	6.57	3.74		
SANNI	**2.84**	**0.69**	0.72	4.38	2.67	2.44

Результаты экспериментов по сравнению точности восстановления метода SANNI с аналогами приведены в табл. 3. Можно видеть, что на временных рядах HumanActivity предлагаемый в данной работе метод SANNI опережает как аналитические алгоритмы, так и нейросетевые методы. Для остальных временных рядов лучшую точность восстановления показывает метод ANNI.

3.3. Влияние параметров

В данном разделе представлены результаты экспериментов по исследованию влияния параметров метода SANNI, количества сниппетов K и длина сниппета m, на точность восстановления. Эксперименты проведены на временном ряде HumanActivity-12 (см. табл. 1). В экспериментах в качестве порога аугментации (см. раздел 2.3) было взято значение $\psi = 0.01$. Для формирования обучающей и тестовой выборок использовались 70% и 30% случайно выбранных подпоследовательностей исходного временного ряда соответственно.

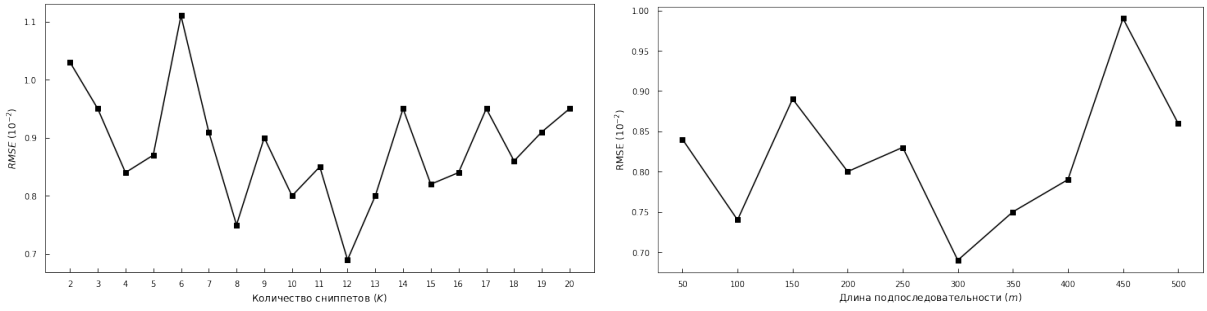

Рис. 6. Зависимость точности восстановления от параметров метода SANNI

Зависимость точности восстановления с помощью метода SANNI от количества сниппетов представлена на рис. 6. Можно видеть, что наиболее высокую точность метод показывает при $K = 12$, когда количество сниппетов совпадает с фактическим значением различных видов физической активности человека, представленных в ряде (см. табл. 2).

Метод SANNI показывает наиболее высокую точность, когда параметр равен фактическому значению ($m = 300$), а при отклонении от такового точность уменьшается.

Заключение

В статье рассмотрена проблема восстановления пропущенных значений потокового временного ряда в режиме реального времени. Данная задача возникает в широком спектре практических приложений цифровой индустрии и интернета вещей, в которых пропуски в показаниях датчиков недопустимы и должны быть незамедлительно заменены на правдоподобные синтетические значения.

Авторами предложен новый метод восстановления пропущенных значений потокового временного ряда в режиме реального времени, названный SANNI (Snippet and Neural Network based Imputation). Метод SANNI предполагает следующие этапы восстановления пропусков: предварительная обработка данных, распознавание и реконструкция. Предварительная обработка предполагает однократную подготовку обучающих выборок данных. Распознавание и реконструкция реализуются с помощью нейронных сетей, обучаемых на указанных выборках.

Предварительная обработка включает в себя поиск сниппетов и аугментацию сниппетов. Сниппет [8] представляет собой уточнение понятия типичной подпоследовательности временного ряда. Временной ряд логически разбивается на сегменты (непересекающиеся подпоследовательности) заданной длины, сниппет — один из сегментов ряда. Со сниппетом ассоциируются его ближайшие соседи — подпоследовательности ряда, имеющие ту же длину, что и сниппет, которые более похожи на данный сниппет, чем на другие сегменты. Для вычисления схожести подпоследовательностей используется специализированная мера схожести MPdist [25], основанная на евклидовом расстоянии. Сниппеты упорядочиваются по убыванию мощности множества своих ближайших соседей. Далее выполняется аугментация (искусственное расширение) каждого маломощного множества ближайших соседей сниппета, которая обеспечивает сбалансированность обучающих выборок нейронных сетей, реализующих распознавание и реконструкцию.

Распознавание реализуется с помощью сверточной нейронной сети. На вход данной сети подается вектор из элементов временного ряда, предшествующих пропущенному элементу; длина указанного вектора совпадает с длиной сниппета. Распознаватель выдает сниппет, на который более всего похожа входная подпоследовательность. Реконструкция реализуется с помощью нейронной сети из управляемых рекуррентных блоков (Gated Recurrent Units, GRU). На вход данной сети подается вектор, представляющий собой конкатенацию двух следующих векторов: сниппет, являющийся результатом этапа распознавания, и вектор элементов ряда, предшествующих пропуску (длина указанного вектора на единицу меньше длины сниппета). Реконструктор выдает восстановленное значение.

Представлены результаты экспериментов на реальных временных рядах, показывающих высокую точность восстановления метода SANNI и его преимущество перед аналогами.

В качестве возможного направления будущих исследований мы рассматриваем внедрение в метод SANNI адаптивного автоматизированного подбора параметра, отвечающего за длину сниппета.

Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 20-07-00140) и Министерства науки и высшего образования РФ (государственное задание FENU-2020-0022).

Литература

1. Kumar S., Tiwari P., Zymbler M. 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. Xu L.D., Duan L. Big Data for cyber physical systems in Industry 4.0: A survey // Enterp. Inf. Syst. 2019. Vol. 13, no. 2. P. 148–169. DOI: 10.1080/17517575.2018.1442934.

3. Цымблер М.Л., Краева Я.А., Латыпова Е.А. и др. Очистка сенсорных данных в интеллектуальных системах управления отоплением зданий // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2021. Т. 10, № 3. С. 16–36. DOI: 10.14529/cmse210302.

4. Иванов С.А., Никольская К.Ю., Радченко Г.И. и др. Концепция построения цифрового двойника города // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2020. Т. 9, № 4. С. 5–23. DOI: 10.14529/cmse200401.

5. Епишев В.В., Исаев А.П., Миниахметов Р.М. и др. Система интеллектуального анализа данных физиологических исследований в спорте высших достижений // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2013. Т. 2, № 1. С. 44–54. DOI: 10.14529/cmse130105.

6. Абдуллаев С.М., Ленская О.Ю., Гаязова А.О. и др. Алгоритмы краткосрочного прогноза с использованием радиолокационных данных: оценка трасляции и композиционный дисплей жизненного цикла // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2014. Т. 3, № 1. С. 17–32. DOI: 10.14529/cmse140102.

7. Дышаев М.М., Соколинская И.М. Представление торговых сигналов на основе адаптивной скользящей средней Кауфмана в виде системы линейных неравенств // Вестник ЮУрГУ. Серия: Вычислительная математика и информатика. 2013. Т. 2, № 4. С. 103–108. DOI: 10.14529/cmse130408.

8. 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.

9. Sande I.G. Hot-deck imputation procedures // Incomplete data in sample surveys. 1983. Vol. 3. P. 339–349.

10. Batista G.E.A.P.A., Monard M.C. An Analysis of Four Missing Data Treatment Methods for Supervised Learning // Appl. Artif. Intell. 2003. Vol. 17, no. 5-6. P. 519–533. DOI: 10.1080/713827181.

11. de Carvalho Jr. O.A., Guimarães R.F., Gomes R.A.T., da Silva N.C. Time series interpolation // IEEE International Geoscience & Remote Sensing Symposium, IGARSS 2007, Barcelona, Spain, July 23–28, 2007. Proceedings. IEEE, 2007. P. 1959–1961. DOI: 10.1109/IGARSS.2007.4423211.

12. Yi B., Sidiropoulos N.D., Johnson T. *et al.* Online Data Mining for Co-Evolving Time Sequences // Proceedings of the 16th International Conference on Data Engineering, San Diego,

California, USA, February 28 – March 3, 2000 / Ed. by Lomet D.B., Weikum G. IEEE Computer Society, 2000. P. 13–22. DOI: 10.1109/ICDE.2000.839383.

13. Paulhus J., Kohler M. Interpolation of missing precipitation records // Monthly Weather Review. 1952. Vol. 80, no. 8. P. 129–133. DOI: 10.1175/1520-0493(1952)080<0129:IOMPR>2.0.CO;2.

14. Box G.E., Jenkins G.M., Reinsel G.C., Ljung G.M. Time Series Analysis: Forecasting and Control, 5th Edition. John Wiley & Sons, 2015. 712 p.

15. 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.

16. Khayati M., Böhlen M.H. REBOM: Recovery of Blocks of Missing Values in Time Series // Proceedings of the 18th International Conference on Management of Data, COMAD 2012, Pune, India / Ed. by Sahasrabuddhe C., Abbadi A.E., Murthy K., Bhattacharya A. Computer Society of India, 2012. P. 44–55. URL: http://comad.in/comad2012/pdf/khayati.pdf.

17. Khayati M., Böhlen M.H., Gamper J. Memory-efficient centroid decomposition for long time series // IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 – April 4, 2014 / Ed. by Cruz I.F., Ferrari E., Tao Y. *et al.* IEEE Computer Society, 2014. P. 100–111. DOI: 10.1109/ICDE.2014.6816643.

18. Papadimitriou S., Sun J., Faloutsos C., Yu P.S. Dimensionality Reduction and Filtering on Time Series Sensor Streams // Managing and Mining Sensor Data / Ed. by Aggarwal C.C. Springer, 2013. P. 103–141. DOI: 10.1007/978-1-4614-6309-2_5.

19. Pearson K. On lines and planes of closest fit to systems of points in space // Philosophical Magazine. 1901. Vol. 2. P. 559–572. DOI: 10.1080/14786440109462720.

20. Wellenzohn K., Böhlen M.H., Dignös A. *et al.* Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series // Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21–24, 2017 / Ed. by Markl V., Orlando S., Mitschang B. *et al.* OpenProceedings.org, 2017. P. 330–341. DOI: 10.5441/002/edbt.2017.30.

21. Hsu H., Yang A.C., Lu M. KNN-DTW Based Missing Value Imputation for Microarray Time Series Data // J. Comput. 2011. Vol. 6, no. 3. P. 418–425. DOI: 10.4304/jcp.6.3.418-425.

22. Berndt D.J., Clifford J. Using Dynamic Time Warping to Find Patterns in Time Series // Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03 / Ed. by Fayyad U.M., Uthurusamy R. AAAI Press, 1994. P. 359–370.

23. 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, Montréal, Canada, December 3–8, 2018 / Ed. by Bengio S., Wallach H.M., Larochelle H. *et al.* 2018. P. 6776–6786. URL: https://proceedings.neurips.cc/paper/2018/hash/734e6bfcd358e25ac1db0a4241b95651-Abstract.html.

24. Guo Z., Wan Y., Ye H. A data imputation method for multivariate time series based on generative adversarial network // Neurocomputing. 2019. Vol. 360. P. 185–197. DOI: 10.1016/j.neucom.2019.06.007.

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. Reingold E., Nievergelt J., Deo N. Combinatorial Algorithms: Theory and Practice. Prentice Hall, 1977. 433 p.

27. Cho K., van Merrienboer B., Gülçehre Ç. *et al.* Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation // Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, October 25–29, 2014, A meeting of SIGDAT, a Special Interest Group of the ACL / Ed. by Moschitti A., Pang B., Daelemans W. ACL, 2014. P. 1724–1734. DOI: 10.3115/v1/d14-1179.

28. Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies // A Field Guide to Dynamical Recurrent Neural Networks / Ed. by Kremer S., Kolen J. Wiley-IEEE Press, 2001. P. 237–243. DOI: 10.1109/9780470544037.ch14.

29. Reiss A., Stricker D. Introducing a New Benchmarked Dataset for Activity Monitoring // 16th International Symposium on Wearable Computers, ISWC 2012, Newcastle, United Kingdom, June 18–22, 2012. IEEE Computer Society, 2012. P. 108–109. DOI: 10.1109/ISWC.2012.13.

30. Candanedo L., Feldheim V., Deramaix D. Data driven prediction models of energy use of appliances in a low-energy house // Energy and Buildings. 2017. Vol. 140. P. 81–97. DOI: 10.1016/j.enbuild.2017.01.083.

31. Individual household electric power consumption data set. UCI Machine Learning Repository. URL: `https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption` (дата обращения: 03.09.2021).

32. Moritz S., Bartz-Beielstein T. imputeTS: Time Series Missing Value Imputation in R // R Journal. 2017. Vol. 9, no. 1. P. 207. DOI: 10.32614/rj-2017-009.

Цымблер Михаил Леонидович, д.ф.-м.н., доцент, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Полонский Вячеслав Александрович, магистрант, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

Юртин Алексей Артемьевич, магистрант, кафедра системного программирования, Южно-Уральский государственный университет (национальный исследовательский университет) (Челябинск, Российская Федерация)

DOI: 10.14529/cmse210401

ON ONE METHOD OF IMPUTATION MISSING VALUES OF A STREAMING TIME SERIES IN REAL TIME

© 2021 M.L. Zymbler, V.A. Polonsky, A.A. Yurtin

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

E-mail: mzym@susu.ru, s.polonski@mail.ru, lideor@yandex.ru

Received: 03.09.2021

The problem of the imputation of missing values in a streaming time series arises in a wide range of Industry 4.0 and Internet of Things applications. In the article, we propose a novel imputation method based on time series mining techniques and artificial neural networks. The method involves three steps of imputation: data preprocessing, recognition, and reconstruction. Preprocessing is a one-time preparation of training data samples. Recognition and reconstruction are implemented through two neural networks trained on the samples above. Preprocessing supposes the discovery of a set of typical subsequences (snippets) in a pre-stored fragment of the streaming time series without misses. Recognition is implemented through a Convolutional Neural Network, and its input is a vector of the elements preceding the current (missing) value. The Recognizer outputs the snippet that the input subsequence is most similar to. Reconstruction is implemented through a Recurrent Neural Network, and its input is a concatenation of the Recognizer's output and the vector of the elements preceding the missing value. The Reconstructor outputs the value to be imputed. The experimental results show high accuracy and the advantage of the proposed method over analogs.

Keywords: time series, imputation of missing values, online mode, artificial neural networks, CNN, RNN, time series snippets.

FOR CITATION

Zymbler M.L., Polonsky V.A., Yurtin A.A. On One Method of Imputation Missing Values of a Streaming Time Series in Real Time. Bulletin of the South Ural State University. Series: Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 4. P. 5–25. (in Russian) DOI: 10.14529/cmse210401.

This paper is distributed under the terms of the Creative Commons Attribution-Non Commercial 4.0 License which permits non-commercial use, reproduction and distribution of the work without further permission provided the original work is properly cited.

References

1. Kumar S., Tiwari P., Zymbler M. 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. Xu L.D., Duan L. Big Data for cyber physical systems in Industry 4.0: A survey. Enterp. Inf. Syst. 2019. Vol. 13, no. 2. P. 148–169. DOI: 10.1080/17517575.2018.1442934.

3. Zymbler M.L., Kraeva Y.A., Latypova E.A. *et al.* Cleaning sensor data in intelligent heating control system. Bulletin of the South Ural State University. Series: Computational Mathematics and Computational Mathematics and Software Engineering. 2021. Vol. 10, no. 3. P. 16–36. (in Russian) DOI: 10.14529/cmse210302.

4. Ivanov S.A., Nikolskaya K.Y., Radchenko G.I. *et al.* Digital twin of a city: concept overview. Bulletin of the South Ural State University. Series: Computational Mathematics and Com-

putational Mathematics and Software Engineering. 2020. Vol. 9, no. 4. P. 5–23. (in Russian) DOI: 10.14529/cmse200401.

5. Epishev V.V., Isaev A.P., Miniakhmetov R.M. *et al.* Physiological data mining system for elite sports. Bulletin of the South Ural State University. Computational Mathematics and Software Engineering. 2013. Vol. 2, no. 1. P. 44–54. (in Russian) DOI: 10.14529/cmse130105.

6. Abdullaev S.M., Lenskaya O.Y., Gayazova A.O. *et al.* Short-range forecasting algorithms using radar data: translation estimate and life-cycle composite display. Bulletin of the South Ural State University. Series: Computational Mathematics and Computational Mathematics and Software Engineering. 2014. Vol. 3, no. 1. P. 17–32. (in Russian) DOI: 10.14529/cmse140102.

7. Dyshaev M.M., Sokolinskaya I.M. Representation of trading signals based on the Kaufman's Adaptive Moving Average in the form of a system of linear inequalities. Bulletin of the South Ural State University. Series: Computational Mathematics and Computational Mathematics and Software Engineering. 2013. Vol. 2, no. 4. P. 103–108. (in Russian) DOI: 10.14529/cmse130408.

8. 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.

9. Sande I.G. Hot-deck imputation procedures. Incomplete data in sample surveys. 1983. Vol. 3. P. 339–349.

10. Batista G.E.A.P.A., Monard M.C. An Analysis of Four Missing Data Treatment Methods for Supervised Learning. Appl. Artif. Intell. 2003. Vol. 17, no. 5-6. P. 519–533. DOI: 10.1080/713827181.

11. de Carvalho Jr. O.A., Guimarães R.F., Gomes R.A.T., da Silva N.C. Time series interpolation. IEEE International Geoscience & Remote Sensing Symposium, IGARSS 2007, Barcelona, Spain, July 23–28, 2007. Proceedings. IEEE, 2007. P. 1959–1961. DOI: 10.1109/IGARSS.2007.4423211.

12. Yi B., Sidiropoulos N.D., Johnson T. *et al.* Online Data Mining for Co-Evolving Time Sequences. Proceedings of the 16th International Conference on Data Engineering, San Diego, California, USA, February 28 – March 3, 2000 / Ed. by Lomet D.B., Weikum G. IEEE Computer Society, 2000. P. 13–22. DOI: 10.1109/ICDE.2000.839383.

13. Paulhus J., Kohler M. Interpolation of missing precipitation records. Monthly Weather Review. 1952. Vol. 80, no. 8. P. 129–133. DOI: 10.1175/1520-0493(1952)080<0129:IOMPR>2.0.CO;2.

14. Box G.E., Jenkins G.M., Reinsel G.C., Ljung G.M. Time Series Analysis: Forecasting and Control, 5th Edition. John Wiley & Sons, 2015. 712 p.

15. 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.

16. Khayati M., Böhlen M.H. REBOM: Recovery of Blocks of Missing Values in Time Series. Proceedings of the 18th International Conference on Management of Data, COMAD 2012, Pune, India / Ed. by Sahasrabuddhe C., Abbadi A.E., Murthy K., Bhattacharya A. Computer Society of India, 2012. P. 44–55. URL: http://comad.in/comad2012/pdf/khayati.pdf.

17. Khayati M., Böhlen M.H., Gamper J. Memory-efficient centroid decomposition for long time series. IEEE 30th International Conference on Data Engineering, Chicago, ICDE 2014, IL, USA, March 31 – April 4, 2014 / Ed. by Cruz I.F., Ferrari E., Tao Y. *et al.* IEEE Computer Society, 2014. P. 100–111. DOI: 10.1109/ICDE.2014.6816643.

18. Papadimitriou S., Sun J., Faloutsos C., Yu P.S. Dimensionality Reduction and Filtering on Time Series Sensor Streams. Managing and Mining Sensor Data / Ed. by Aggarwal C.C. Springer, 2013. P. 103–141. DOI: 10.1007/978-1-4614-6309-2_5.

19. Pearson K. On lines and planes of closest fit to systems of points in space. Philosophical Magazine. 1901. Vol. 2. P. 559–572. DOI: 10.1080/14786440109462720.

20. Wellenzohn K., Böhlen M.H., Dignös A. *et al.* Continuous Imputation of Missing Values in Streams of Pattern-Determining Time Series. Proceedings of the 20th International Conference on Extending Database Technology, EDBT 2017, Venice, Italy, March 21–24, 2017 / Ed. by Markl V., Orlando S., Mitschang B. *et al.* OpenProceedings.org, 2017. P. 330–341. DOI: 10.5441/002/edbt.2017.30.

21. Hsu H., Yang A.C., Lu M. KNN-DTW Based Missing Value Imputation for Microarray Time Series Data. J. Comput. 2011. Vol. 6, no. 3. P. 418–425. DOI: 10.4304/jcp.6.3.418-425.

22. Berndt D.J., Clifford J. Using Dynamic Time Warping to Find Patterns in Time Series. Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03 / Ed. by Fayyad U.M., Uthurusamy R. AAAI Press, 1994. P. 359–370.

23. 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, Montréal, Canada, December 3–8, 2018 / Ed. by Bengio S., Wallach H.M., Larochelle H. *et al.* 2018. P. 6776–6786. URL: https://proceedings.neurips.cc/paper/2018/hash/734e6bfcd358e25ac1db0a4241b95651-Abstract.html.

24. Guo Z., Wan Y., Ye H. A data imputation method for multivariate time series based on generative adversarial network. Neurocomputing. 2019. Vol. 360. P. 185–197. DOI: 10.1016/j.neucom.2019.06.007.

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. Reingold E., Nievergelt J., Deo N. Combinatorial Algorithms: Theory and Practice. Prentice Hall, 1977. 433 p.

27. Cho K., van Merrienboer B., Gülçehre Ç. *et al.* Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, October 25–29, 2014, A meeting of SIGDAT, a Special Interest Group of the ACL / Ed. by Moschitti A., Pang B., Daelemans W. ACL, 2014. P. 1724–1734. DOI: 10.3115/v1/d14-1179.

28. Hochreiter S., Bengio Y., Frasconi P., Schmidhuber J. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. A Field Guide to Dynamical Recurrent Neural Networks / Ed. by Kremer S., Kolen J. Wiley-IEEE Press, 2001. P. 237–243. DOI: 10.1109/9780470544037.ch14.

29. Reiss A., Stricker D. Introducing a New Benchmarked Dataset for Activity Monitoring. 16th International Symposium on Wearable Computers, ISWC 2012, Newcastle, United Kingdom, June 18–22, 2012. IEEE Computer Society, 2012. P. 108–109. DOI: 10.1109/ISWC.2012.13.

30. Candanedo L., Feldheim V., Deramaix D. Data driven prediction models of energy use of appliances in a low-energy house. Energy and Buildings. 2017. Vol. 140. P. 81–97. DOI: 10.1016/j.enbuild.2017.01.083.

31. Individual household electric power consumption data set. UCI Machine Learning Repository. URL: `https://archive.ics.uci.edu/ml/datasets/individual+household+electric+power+consumption` (accessed: 03.09.2021).

32. Moritz S., Bartz-Beielstein T. imputeTS: Time Series Missing Value Imputation in R. R Journal. 2017. Vol. 9, no. 1. P. 207. DOI: 10.32614/rj-2017-009.