南京邮电大学学报(自然科学版)
Journal of Nanjing University of Posts and Telecommunications(Natural Science Edition)
ISSN 1673-5439,CN 32-1772/TN

《南京邮电大学学报(自然科学版)》网络首发论文

题目：基于时间序列预测的冷数据压缩方法
作者：张永潘，张正，徐良红，钱超，丁诚，潘甦
收稿日期：2024-09-27
网络首发日期：2025-04-21
引用格式：张永潘，张正，徐良红，钱超，丁诚，潘甦．基于时间序列预测的冷数据压缩方法[J/OL]．南京邮电大学学报(自然科学版).
https://link.cnki.net/urlid/32.1772.tn.20250421.0846.002

网络首发时间：2025-04-21 14:08:53　网络首发地址：https://link.cnki.net/urlid/32.1772.tn.20250421.0846.002

基于时间序列预测的冷数据压缩方法

张永潘[1]，张　正[1]，徐良红[1] 钱　超[1]，丁　诚[1]，潘　甦[2]

(1. 中国电信江苏分公司，江苏 南京 210003
2. 南京邮电大学 物联网学院，江苏 南京 210003)

摘要：提出了基于时间序列预测的时序数据库的冷数据压缩方法。该方法采用一种双自回归预测模型对时序数据进行预测，只储存预测值和真实值的残差以及预测模型系数。在解码端通过预测模型系数构建预测模型输出预测值，然后加上残差恢复原始值。由于残差远远小于原始值，因此可以用很少的比特量化，同时预测模型系数在贮存一次后在长时间内不必重复存贮，因此达到压缩时序数据的目的。论文在 8 个不同大小类型的真实数据集上的实验结果证明了该压缩方法良好的压缩表现，适合处理数据量大但对算法时延没有要求的历史老数据。

关键词:时间序列预测；模型系数；数据压缩

中图分类号：TP311.13　　**文献标识码：**　A

A Cold Data Compression Method Based on Time Series Prediction

ZHANG Yongpan[1], ZHANG Zheng[1], XU Lianghong[1],

QIAN Chao[1], DING Cheng[1], PAN Su[2]

（1. China Telecom Jiangsu Branch, Nanjing 210003, China

2. School of Internet of Things, Nanjing University of Posts and Telecommunications, Nanjing 210003, China）

Abstract: A method based on time series prediction is proposed for cold data compression. This method utilizes a dual autoregressive prediction model to forecast temporal data, storing only the residuals between the predicted and actual values, along with the coefficients of the predictive model. During decoding, the stored coefficients are used to reconstruct the prediction model to generate predicted values, which are then combined with the residuals to recover the original values. Since the residuals are significantly smaller than the original values, they can be quantized using fewer bits, while the prediction model coefficients, once stored, do not need to be stored repeatedly over time. Thereby, this approach achieves the good compression performance of temporal data. Experimental results on eight real datasets of varying sizes and types demonstrate the excellent compression performance of this method. This method is suitable for processing historical cold data with a large amount of data but no requirements for algorithm latency.

Keywords: Time Series Prediction; Model Coefficients; Data Compression

在物联网和工业互联网等智能互联技术快速普及应用的背景下，智能设备数量的急剧增加导致时序数据[1-4]爆炸式增长，随之而来的海量时序数据存储处理问题为时序数据库的存储能力提出了新的要求。

在时序数据库存储技术的研究中，数据压缩是一个基础而又重要的问题。时序数据压缩不仅能够减少压缩后数据的存储需求，还能够降低数据传输所需的资源成本；因此各大数据

收稿日期：2024-09-27；修回日期：2024-11-17　　本刊网址：http://nyzr.njupt.edu.cn
基金项目：国家自然科学基金（62071244）和中国电信江苏分公司数据库技术项目资助项目
作者简介：张永潘，男，工程师；潘甦（通信作者），男，博士，教授，博士生导师，supan@njupt.edu.cn

库软件如 InfluxDB、TDengine、Prometheus 和 OpenTSDB 等都对时序数据压缩展开了深入研究。

InfluxDB[5]在数据写入阶段(热数据进入内存)，先将数据存储在内存中，此时的压缩依赖于 RLE（Run-Length Encoding）算法[6]。RLE 算法通过将连续出现的相同字段替换为两个部分来实现压缩:一个是出现的值，另一个是该值连续出现的次数，从而减少了存储空间。而数据需要存入磁盘时（冷数据入库保存阶段），其使用 TSM(Time-Structured Merge Tree)文件格式进行数据复杂压缩。在 TSM 文件格式中，InfluxDB 会针对不同数据类型使用不同的压缩方式，对时间戳使用差分编码(delta 算法)，存储时间序列的第一个时间戳，然后存储后续时间戳与前一个时间戳的差值，大多数差值可以用更少的比特来表示。整数型数据的压缩也采用类似于时间戳的差分编码，存储后续值与前一个值的差值。对于浮点数，InfluxDB 采用 Facebook 的 Gorila 压缩算法，首先存储第一个值完整的浮点数，然后对于后续的数值，只存储与前一个数值相比发生变化的位，因此这种方法能有效减少所需存储空间。对于字符串数据，InfluxDB 会使用一种基于字典的压缩方法。它首先会构建一个字符串字典，将不同字符串映射到一个 ID。然后，在存储数据时，它只存储这些 ID 而不是完整的字符串。总的来说，InfluxDB 利用多种压缩算法，以确保在冷热数据都能够达到高效的压缩效果。

Prometheus[7]在热数据阶段和冷数据入库保存阶段保持了相同的压缩策略，它与 InfluxDB 类似，针对不同的数据类型采用不同的压缩方法，Prometheus 对每个时间序列的时间戳、整数型和浮点数值都使用 Gorila 压缩算法，而对于字符串类型和布尔型的数据，与 InfluxDB 一样，分别采用字典压缩和比特压缩方法。

TDengine 与 InfluxDB 相反，在热数据阶段，对不同类型的数据单体使用不同的压缩算法，而在冷数据阶段，使用整体压缩方法。在单体压缩时，TDengine 使用 Delta-of-Delta 算法对时间戳进行压缩，它是差分编码的变体，该算法先得到相邻数据点的差值，然后再对差值进行第二次差分编码，得到相邻差值的差值。针对无符号的整数和有符号的整数分别使用 Simple8B 算法和 Zig-Zag 对其进行压缩。Simple8B 算法将 64 位字划分为多个块，每个块包含一个控制位和 7 个数据位，控制位指示了该块中存储的数据的编码方式，并采用变长编码以实现对整数数据的高效压缩。Zig-Zag 算法通过将有符号整数转换为无符号整数，并通过位运算的方式进行编码，从而减少整数的位数，减小存储空间的占用。在整体压缩时，TDengine[8]使用 LZ4 算法，它将数据分成固定大小的块，并计算每个片段的哈希值。如果两个片段具有相同的哈希值，则它们可能是相同的或相似的片段。LZ4 算法会进一步验证当前位置与哈希表中对应位置的数据是否真的相同。如果验证成功，LZ4 算法会记录下这个重复片段的起始位置和长度，将其替换为对应的指针。

不同于上述数据库，OpenTSDB[9]自身不实现特定的压缩算法，而是依赖于其底层数据库 HBase 支持的压缩技术，如 Snappy、LZ4 等。Snappy 算法是 Google 开发的一种快速压缩和解压缩算法，与 LZ4 算法类似，Snappy 算法不分冷热数据，它将输入数据分成大小为 64KB 的块，使用哈希表查找重复片段。如果发现块中存在重复的片段，Snappy 算法会将这些重复片段替换为对应的指针，并将指针存储在压缩后的数据中。

从上述可见，尽管目前使用的方法繁多，然而在冷数据压缩时，他们基本思路是一致的，即尽量减少对重复字段的贮存，因此这些方法难以通过串联使用来进一步压缩数据。

本文探索从完全不同于现有文献的思路来压缩时序数据的冷数据。随着机器学习的发展，精确预测数据成为可能，因此我们可以完全不存储原始数据而只存储预测模型参数和预测数据与原始数据的残差。由于模型参数可以长期不变，残差的值很小，所以可以使用较少的空间来储存。在解码时，解码端根据参数构建预测模型产生预测数据，加上残差后可以做到无损恢复。由于冷数据是需要归档保存的老数据，使用频率不高，因此对压缩算法在时间上和算法复杂度上没有严格的限制。很适合采用机器学习这种方法较为耗时的方法处理量大但用

的很少的“历史老数据”。

1 基于模型重构的时序数据压缩方法

基于预测模型重构的时序数据压缩方法，分为压缩编码和解码恢复两个模块。本方法整体实现流程如图 1 所示。

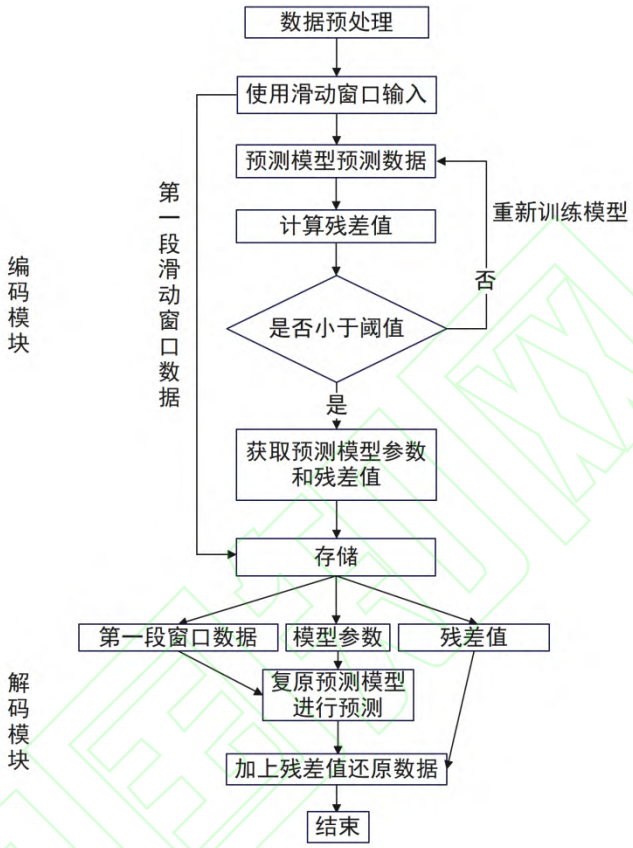

图 1　压缩方法整体实现流程

压缩编码模块负责时序数据的压缩功能，使用预测模型对时序数据进行预测，将预测数据和原始数据对比计算残差。由于预测模型参数固定，随着预测的进行，会出现残差随着时间增大的现象。因此本文引入了残差阈值判断更新模型参数机制来监测残差的变化情况，当残差超过一定阈值时，触发对模型参数的更新。这种机制可以有效应对数据分布或趋势发生变化时的情况，保持预测的准确性和稳定性。为了能使解码端恢复预测模型，第一段窗口内的部分原始数据和预测模型参数将存储于数据库中。在解码端利用这些数据预测后面的时序数据加上存贮的残差可以无损恢复原始数据。

1.1 编码模块

编码模块围绕着预测模块设计。对时序序列[10]的预测有多种模型，从复杂的 Transformer,LSTM 到简单的自回归模型[11-14]都可以预测数据，由于我们要存贮模型系数和预测残差，因此模型系数和残差都会影响压缩效果，复杂模型预测残差小，但系数多。简单模型则反之。在这里，我们提出了一种双递归算法，可以取得模型系数个数和残差大小的平衡。

第一个自回归步骤，我们把当前时刻的值用之前 m 个值的线性组合加上误差项。即：

$$a_t = \phi_1 a_{t-1} + \phi_2 a_{t-2} + \cdots + \phi_m a_{t-m} + \varepsilon_t \tag{1}$$

其中 a_t 为当前值，$\phi_1,\phi_2,...\phi_m$ 为自回归系数，ε_t 为预测误差。

我们使用最小二乘法进行计算自回归系数 $\phi_1,\phi_2,\cdots,\phi_m$。为了使用最小二乘法来估计自回归系数，我们需要将(1)式构成方程组。在构建这个模型时，我们在时间序列中移动窗口，每次移动一个时间步，移动 T-m 次。可以认为 T 个数据构成了测试集，因此总共有 T-m 个待测点 a_t，由 m 个过去的预测值预测而来。将预测 T-m 个 a_t 所需的 $(T-m)\times m$ 个过去观测值和 m 个常数项（预测误差）构成 X 矩阵大小为 $(T-m)\times(m+1)$，其中每一行包含 m 个观测值和一个常数项，将 m 个 a_t 构成向量 Y。自回归系数 ϕ 可以表示为（2）式的解：

$$X^T X\phi = X^T Y \tag{2}$$

其中 X^T 表示矩阵 X 的转置。因为需要求解 m 个参数，故需要 m 个方程求解，所以设置 T=2m，后 m 个数据用于构造 m 个方程组求解模型所需的 m 个参数。

仅仅使用一次自回归方法，预测误差 ε_t 较大，ε_t 仍和 ε_t 的前期的值相关，即 ε_t 的自相关性较大。因此我们用二次自回归方法降低预测误差。即把预测误差分解为当前误差与过去多点误差的和，因此（1）式修改为：

$$a_t = \phi_1 a_{t-1} + \phi_2 a_{t-2} + \cdots + \phi_m a_{t-m} + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \theta_q\varepsilon_{t-q} + \varepsilon_t \tag{3}$$

ε_{t-q} 是 q 时刻之前的误差值。因此，我们下一步的任务变为寻找最优的系数 $\phi_1,\ ..,\phi_m,\theta_1,..,\theta_q$，使得对 a_t 的预测尽量精确，由于 ε_t 的自相关性在第二次自回归中被基本去除，在这种情况下，ε_t 将是一个白噪声。我们分两步完成 $\phi_1,\ ..,\phi_m,\theta_1,..,\theta_q$ 的优化；

1. $\phi_1,\ ..,\phi_m,$ 的初始化：

将（3）中的 $\theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \theta_q\varepsilon_{t-q} + \varepsilon_t$ 部分用一个随机数来代替，这样可以通过最小二乘法（Least Squares Method）（2）式来估计 $\phi_1,\ ..,\phi_m,$。

2. $\theta_1,..,\theta_q$ 的初始化：

将（3）改写为

$$a_t = \mu + \theta_1\varepsilon_{t-1} + \theta_2\varepsilon_{t-2} + \theta_q\varepsilon_{t-q} + \varepsilon_t \tag{4}$$

μ 相对于随机数 $\varepsilon_{t-1},\ ..,\varepsilon_{t-q},\varepsilon_t$ 是个固定值。（4）和（1）式类似，依然可以用类似式（2）的最小二乘法获得系数 $\theta_1,..,\theta_q$。这样我们得到一个初始参数集

$$\Theta=\left\{\phi_1,\phi_2,\ldots,\phi_{\mathrm{m}},\theta_1,\theta_2,\ldots,\theta_q\right\}$$

3. 最大似然化算法优化参数集

在训练阶段我们要使用数据集为$\left\{a_1,a_2,\ldots,a_T\right\}$，每个$a_t$独立服从的概率分布为$f(a_t)$，由于$a_t$可用参数$\Theta$来估计，所以$f(a_t)$可记作$f(a_t;\Theta)$。我们采用最大似然估计的方法找到最优的参数集$\Theta$，即在给定这些参数时，观测到的数据$\left\{a_1,a_2,\ldots,a_T\right\}$出现的概率最大。

因此，有理由定义似然函数 L(Θ)为所有观测值的联合概率密度函数的乘积：

$$L(\Theta)=\prod_{t=1}^{T}f(a_t;\Theta) \tag{5}$$

我们要找到最优的参数集Θ使（5）式最大。

由于连乘需要较大计算量，为了简化计算，我们对似然函数取对数，得到对数似然函数：

$$\ln L(\Theta)=\sum_{t=1}^{T}\ln f(a_t;\Theta) \tag{6}$$

以正态分布为例。假设a_t服从均值为μ，方差为σ^2的正态分布$N(\mu,\sigma^2)$。

若数据不服从正态分布，可以根据数据的实际分布选择合适的概率分布来构建似然函数。不同的概率分布有不同的概率密度函数（PDF），根据这些函数来定义似然函数。

其概率密度函数为：$f(a_t;\mu,\sigma^2)=\frac{1}{\sqrt{2\pi\sigma^2}}\exp(-\frac{(a_t-\mu)^2}{2\sigma^2})$

则对于数据集$\left\{a_1,a_2,\ldots,a_T\right\}$，对数似然函数为：

$$\begin{aligned}\ln L(\mu,\sigma^2)&=\sum_{t=1}^{T}(\ln(\frac{1}{\sqrt{2\pi\sigma^2}})+\ln(\exp(-\frac{(a_t-\mu)^2}{2\sigma^2})))\\&=-\frac{T}{2}\ln(2\pi\sigma^2)-\frac{1}{2\sigma^2}\sum_{t=1}^{T}(a_t-\mu)^2\end{aligned} \tag{7}$$

由 （3）式可得：

$$\varepsilon_t=a_t-(\sum_{i=1}^{m}\phi_i a_{t-1}+\sum_{j=1}^{q}\theta_j\varepsilon_{t-j}) \tag{8}$$

根据定义可得：

$$\sum_{t=1}^{T}(a_t-\mu)^2\approx\sum_{t=1}^{T}\varepsilon_t^2, \tag{9}$$

$$\sigma^2\approx\frac{\sum_{t=1}^{T}(a_t-\mu)^2}{T} \tag{10}$$

将（8），（9）和（10）带入（7）可得：

$$\ln L(\Theta)=-\frac{T}{2}\ln(2\pi\sigma^2)-\frac{T}{2\sum_{t=1}^{T}(Y_t-u)^2}\sum_{t=1}^{T}(a_t-(\sum_{i=1}^{m}\phi_i a'_{t-i}+\sum_{j=1}^{q}\theta_j\varepsilon_{t-j}))^2$$

（11）

$$\mu \approx \frac{1}{T}\sum_{t=1}^{T} a_t$$

给定 a_t， $t=1,\cdots,m$ 和初始参数集 Θ， $\max(\ln L(\Theta))$ 是一个无约束优化问题。我们采用 BFGS[15]算法来得到使（11）最大的参数 Θ_k，其步骤如下：

1、计算梯度：

$$g_k = \nabla \ln L(\Theta_k)$$

2、更新参数：

$$\Theta_{k+1} = \Theta_K - H_K g_k$$

3、计算梯度变化

$$y_k = g_{k+1} - g_k$$

4、计算参数变化

$$s_k = \Theta_{k+1} - \Theta_k$$

5、更新 Hessian 矩阵近似值

$$H_{k+1} = H_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k}$$

其中 Hessian 矩阵的 H 的元素 H_{ij} 定义为：

$$H_{ij} = \frac{\partial^2 \ln L(\Theta)}{\partial \theta_i \partial \theta_j} \tag{12}$$

它描述了对数似然函数的二阶曲率。用于优化过程中的参数更新方向和步长选择。本章设计的编码模块具体流程如图 2 所示。

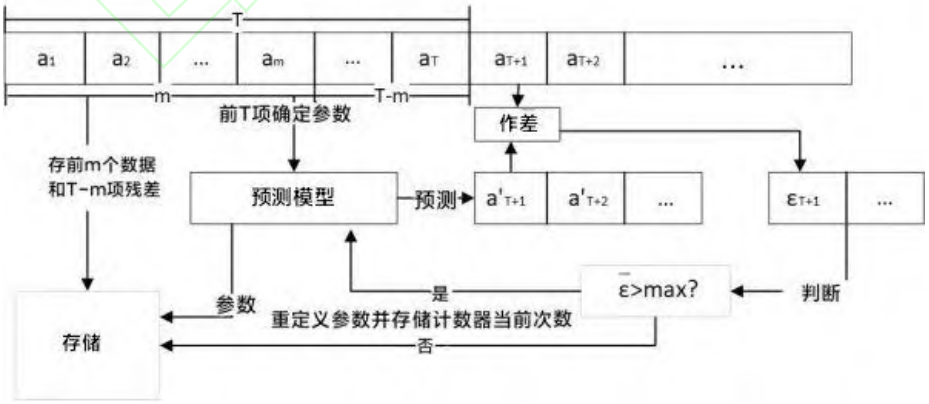

图 2　编码过程示意图

由于时序数据源源不断地产生，因此我们以窗口的形式将其划成数据段输入编码器，如图 2，窗口的大小为 m,我们采用的预测模型方法在前 T 个数据中通过捕捉数据中的时序依

赖关系。而将 a_{m+1} 至 a_{T-m} 构成（1）式，从而确定模型系数来预测下一个时间点 a'_{T+1} 的值、a'_{T+2} 。同时贮存系数和残差。

随着预测的进行，使用同样系数的预测模型预测精度将逐渐降低，即 ε_i 越来越大。这将导致对 ε_i 的储存空间变大。因此我们引入了残差阈值 $T_{\max}$ 这一概念，通过控制残差小于 $T_{\max}$ 来平衡预测残差与模型系数更新次数，模型参数的空间占有量降低。具体的，当 $\bar{\varepsilon} \le \max$ 时，直接存储预测模型参数和残差值，$\bar{\varepsilon} = \frac{1}{m}\sum_{i=1}^{m}\varepsilon_i$ 为残差在一个窗口中的均值。当 $\bar{\varepsilon} > T_{\max}$ 时，就会重新拟合模型的参数找到当下状态的最优参数，并使用更新后的预测模型继续对接下来的窗口数据进行预测，并存储模型参数和残差值。值得注意的是重新训练时，需要启动解码端程序，获得训练参数所需的 T 个数据值。

使用计数器记录预测次数，当误差均值大于门限时，整个流程将重新进行，贮存当前计数器的值，并将计数器归零。为了解码模块能成功复原预测模型，需要保存的模型参数如表 1 所示。

表 1　复原模型所需参数

参数/信息	描述
参数	预测模型的自回归项系数、滑动平均项系数，计数器次数数据。
残差数据	原始数据与预测数据之差

压缩编码算法伪代码如表 2 所示。

表 2　压缩算法伪代码

算法 3.2:基于预测模型的参数压缩算法
1：按窗口读取数据:
2：使用最小二乘法估计系数得到参数初始集合：
3：定义对数似然函数
4：使用 BFGS 算法进行优化
5：得到优化后的参数：
6：根据得到的参数集和模型参数定义模型
7：使用预测模型对 T 时刻之后后面的数据进行预测，并使用计数器记录预测次数：
8：计算和原始数据的残差:
9：检查残差是否大于设置的阈值，如果大于回到第二步重现拟合参数并记录下当前计数器的次数再将计数器归零，不大于则进入下一步:
10：存储模型参数、0~m 时刻的数据和残差：
11：滑动窗口
12：回第 1 步

1.2 解码模块

解码过程也就是编码模块的反向操作过程。通过提取存储的模型参数来复原数据段对应的预测模型和预测的数据，加上储存的残差来无损的回复原始数据。

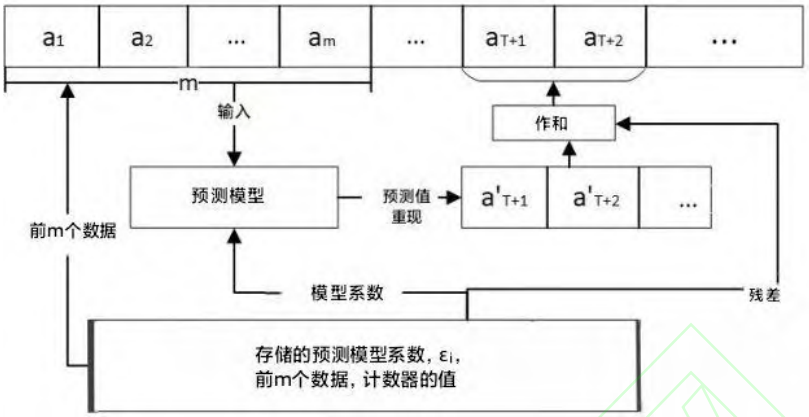

图 3　解压过程示意图

表 3　解压算法伪代码

算法 5.2：基于预测模型参数的解压缩算法
1：读取存储的模型参数:
2：根据读取到的系数重构预测模型:
3：输入前 m 项数据，进行预测值重现，并使用计数器统计预测次数：
4：使用残差和重构的预测值进行数据还原：
5：当解码端预测次数和压缩端记录的次数相同时，将解码端计数器归零，重新提取存储的预测系数，回到第一步重复上述流程
5：结束

解码模块具体实现流程如图 3 所示。首先从存储中获取初始的预测模型参数，复原预测模型，并从存储中获取第一个滑动窗口的数据$a_1,a_2,\cdots,a_m$输入至预测模型，得到预测值$a'_{T+1},a'_{T+2},\cdots$。之后提取对应的残差值。根据残差值复原对应时间步的真实值$a_{T+1},a_{T+2},\cdots$，并将真实值加入滑动窗口，依旧以窗口大小为m，进行滑动预测，直到当前预测次数和所贮存的计数器记录的次数一样时，意味着预测模块系数被更新了，此时要重新提取系数，并循环上述过程直到完全复原数据。

2 实验与结果分析

2.1 数据集与指标评价

为了验证提出的基于预测模型参数提取的时序数据压缩方法的可靠性，本研究选取了 8 个不同大小和类型的测试数据(F1~F8)。其中，F1 是贵州茅台股票 2001-2024 年的部分收盘价数据，F2~F4 是开源数据集 UCR 中的 InlineSkate、MALLAT 和 UWaveGestureLibraryAll 数据。F5~F8 是 2020-2024 年江苏电信真实电信设备不同端口流入的通信流量数据，电信端口每隔 20min 就会流入新的数据，它们的大小如表 4 所示。

表 4　压缩测试集

数据集	数据点数	数据集	数据点数

F1	5349	F5	87974
F2	16929	F6	93168
F3	6138	F7	89180
F4	115168	F8	4655

实验在 Tensorflow 框架和 Python3.8 的环境下进行，操作系统为 Windows 11，处理器为 Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz，GPU 为 NVIDIA GeForce RTX 2060。

本研究选用压缩比、压缩时间和解压缩时间作为提出算法的评价指标。压缩比表示为：

$$CR = M / N \tag{13}$$

其中，M表示原始数据大小，N表示压缩数据大小，CR越高，压缩效果越好。

2.2 算法验证与结果讨论

为了验证本文提出的基于预测模型重构的时序数据压缩方法（以下简称为 Reconstruct prediction model Rpm）的性能，我们选择了两种常见的压缩算法进行对比实验，分别是 Snappy 算法和 Zlib 算法。Snappy 算法是由 Google 开发的，它在处理较小数据块时压缩和解压速度都非常快；Zlib 算法融合了 LZ77 算法和哈夫曼编码机制，实现了压缩效率与处理速度的优良平衡。此外，本文提出的 Rpm 不仅可作为独立的时序数据预压缩方法使用，还能与其他通用压缩算法结合，提升压缩效率。而所有的算法包括 Zlib 和 LZ77 本质上都是一类的：即去除重复的字段，因此，它们难以合并使用。所以本文以 Rpm 结合 Snappy（称为 Rpm+Snappy）为例，先通过 Rpm 进行预压缩，随后利用 Snappy 进行二次压缩。

表 5 展示了各压缩算法的压缩率。由表可知，在 F1~F8 数据集上，Rpm 的压缩率均高于 Snappy，特别是在 F1~F4 数据集上的压缩率均超过了 5，而 Snappy 在这些数据集上的压缩表现平平，稳定在[1,2]的区间内。Rpm 在 F1-F3 和 F7 数据集上的压缩率显著高于 Zlib,但在其它数据集上略低于 Zlib。Rpm 算法在 F1~F8 数据集上的压缩率表现出较大的差异，这是因为电信流量数据重复数据段较小，所以 Rpm 压缩算法在电信数据集 F5~F8 上的压缩性能相较其他数据集有着明显的下降。值得注意的是，由于我们方法 Rpm 和 Snappy 及 Zlib 在不同的层面压缩，使用 Rpm 后可以使用 Zlib 或 Snappy 再压缩一次，表 5 可见。而由于 Zlib 和 Snappy 使用类似的方法压缩，因此 Zlib 压缩后再使用 Snappy 压缩几乎没有效果。

表 5 各压缩算法的压缩率

数据集	压缩算法				
	Snappy	Zlib	Rpm	Rpm+Snappy	Zlib+Snappy
F1	1.33	2.64	8.63	16.15	2.64
F2	1.52	2.66	6.79	11.96	2.66

F3	1.37	2.19	8.21	15.28	2.19
F4	1.74	6.58	6.03	9.53	6.58
F5	1.49	2.55	2.22	2.56	2.55
F6	1.53	2.61	2.28	2.66	2.61
F7	1.55	2.64	3.26	3.52	2.64
F8	1.16	2.07	2.86	3.17	2.07

表 6、7 分别展示了各压缩算法的压缩时间和解压缩时间。从这些数据可以看出，Snappy 在所有数据集上均显示出极快的压缩和解压缩时间，Zlib 算法虽然压缩机制更为复杂，在压缩和解压缩时间上稍逊于 Snappy，但也都保持在 0.1s 以内。本文提出的 Rpm 压缩算法，由于在压缩过程中的预测模型的参数迭代训练需要额外的时间，加上解压缩的时候需要重构预测模型，因此 Rpm 算法压缩和解压缩的时间均长于 Snappy 和 Zlib 算法，当 Rpm 与 Snappy 结合使用时，压缩和解压缩的总体时间并未显著变长，而压缩效率却有了非常显著提升。这在时间不敏感的数据场景中比如对历史冷数据（不常使用的数据）的压缩将大大的节约贮存空间。

表 6　各压缩算法的压缩时间（秒）

数据集	压缩算法			
	Snappy	Zlib	Rpm	LZip+Snappy
F1	0.0001	0.0001	101.6438	101.6438
F2	0.0023	0.0090	253.0049	253.0049
F3	0.0001	0.0051	114.2091	114.2091
F4	0.0030	0.0249	479.5026	479.5046
F5	0.0029	0.0598	372.4472	372.4512
F6	0.0029	0.0702	454.9931	454.9972
F7	0.0029	0.0599	374.7118	374.7158
F8	0.0001	0.0001	99.9318	99.9318

表 7　各压缩算法的解压缩时间（秒）

数据集	解压算法			
	Snappy	Zlib	Rpm	LZip+Snappy
F1	0.0001	0.0000	5.8999	5.8999
F2	0.0003	0.0001	7.2092	7.2097
F3	0.0001	0.0001	5.8997	5.8998
F4	0.0019	0.0019	65.062	65.080
F5	0.0019	0.0039	79.121	79.137
F6	0.0020	0.0040	39.809	39.827
F7	0.0010	0.0030	43.610	43.618
F8	0.0001	0.0000	5.0061	5.0061

图 4 随机截取了 F8 这数据集在使用 Rpm 算法进行压缩过程的部分预测效果图。可见

预测残差非常小，因此贮存残差相比贮存原始值将大大减小贮存空间。

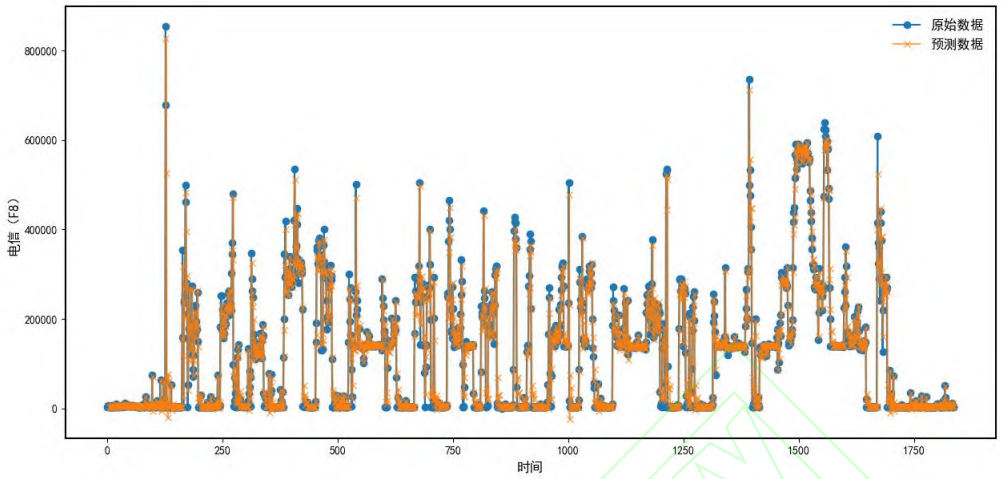

图 4　电信（F8）预测部分效果图

为了实现最优状态的压缩效果，我们选取了 F8 电信数据集作为典型，研究了在不同阈值和窗口情况下的压缩率。图 5 表明，随着阈值的增加，压缩率呈现先增加后减少的趋势。

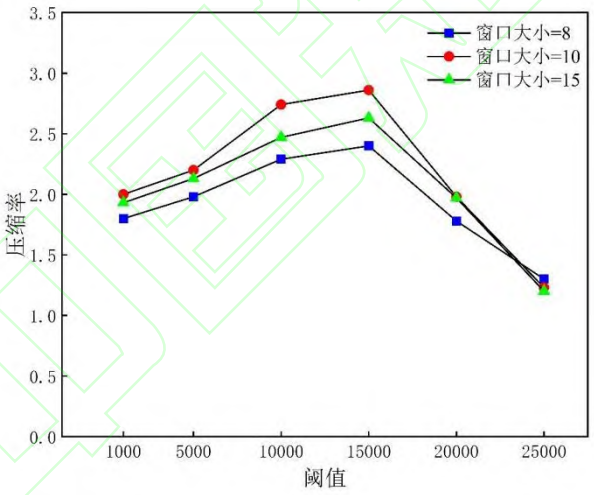

图 5　不同阈值和窗口大小下算法压缩率对比图

当max=1000 和max=5000 时，更新模型参数相对频繁导致存储参数量变大，从而使压缩率处于劣势。在max=20000 和max=25000 时，模型更新频率的大幅降低使模型的预测效果降低，残差数值增大导致压缩率逐渐下降。图 5 中也明显的发现窗口大小为 10 的情况下，算法的压缩率更高。因此，本文选取max=15000 和窗口大小 m=10。

3 结束语

本文提出了基于预测的冷数据压缩方法。通过不存储原始数据而只存储预测模型参数和

残差值来达到数据压缩的效果。由于该方法和现有压缩方法采用的降低相同数据块重复存储的思路完全不同，因此不仅可以作为独立的时序数据压缩方法使用，还能与其他通用压缩算法结合进行二次压缩，提高压缩效率。论文在 8 种不同大小类型的数据集上对比了 Rpm 压缩和现有方法，实验结果表明，所提出的 Rpm 压缩算法实现了较高的压缩率，虽然它的压缩时间相对较长，但对于冷数据压缩是可以接受的。

参考文献

[1] Fu T .A review on time series data mining[J].Engineering Applications of Artificial Intelligence,2010,24(1):164-181.

[2] M. Elbes, S. AlZu'bi and T. Kanan, "Deep Learning-Based Earthquake Prediction Technique Using Seismic Data," 2023 International Conference on Multimedia Computing, Networking and Applications (MCNA), Valencia, Spain, 2023, pp. 103-108.

[3] Y. Fang et al., "The Displacement Analysis and Prediction of a Creeping Ancient Landslide at Suoertou, Zhouqu County, China," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 17, pp. 4139-4163, 2024.

[4] B. Gong, Z. Xu, C. Lin and D. Wu, "Heterogeneous Traffic Flow Detection Using CAV-Based Sensor With I-GAIN," in IEEE Access, vol. 11, pp. 32616-32627, 2023.

[5] InfluxDB: https://www.influxdata.com/

[6] Galka A ,Moontaha S ,Siniatchkin M .Constrained expectation maximisation algorithm for estimating ARMA models in state space representation[J].EURASIP Journal on Advances in Signal Processing,2020,2020(1):716-723.

[7] TDengine: https://tdengine.com/

[8] Prometheus: https://prometheus.io/

[9] OpenTSDB: http://opentsdb.net/

[10] Hewamalage H, Bergmeir C, Bandara K. Recurrent Neural Networks for Time Series Forecasting: Current status and future directions[J]. International Journal of Forecasting, 2020, 37(1): 388-427.

[11] Chimmula V K R, Zhang L. Time series forecasting of COVID-19 transmission in Canada using LSTM networks[J]. Chaos, Solitons & Fractals, 2020, 135: 109864.

[12] S. Xie and H. Fan, "Research on CNN to Feature Extractionn Diseases Prediction," *2019 International Conference on Computer Network, Electronic and Automation (ICCNEA)*, Xi'an, China, 2019, pp. 197-202.

[13] D. Perdios, M. Vonlanthen, F. Martinez, M. Arditi and J. -P. Thiran, "CNN-Based Ultrasound Image Reconstruction for Ultrafast Displacement Tracking," in *IEEE Transactions on Medical Imaging*, vol. 40, no. 3, pp. 1078-1089, March 2021.

[14] Y. Zi, F. Xie, N. Zhang, Z. Jiang, W. Zhu and H. Zhang, "Thin Cloud Removal for Multispectral Remote Sensing Images Using Convolutional Neural Networks Combined With an Imaging Model," in *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 14, pp. 3811-3823, 2021.

[15] Liu D C, Nocedal J. On the limited memory BFGS method for large scale optimization[J]. Mathematical programming, 1989, 45(1): 503-528.