RESEARCH ARTICLE

Prediction of hierarchical time series using structured regularization and its application to artificial neural networks

Tomokaze Shiratori[1], **Ken Kobayashi**[2]*, **Yuichi Takano**[3]

1 Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki, Japan, **2** Artificial Intelligence Laboratory, Fujitsu Laboratories Ltd., Kawasaki, Kanagawa, Japan, **3** Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Ibaraki, Japan

* ken-kobayashi@fujitsu.com

OPEN ACCESS

Citation: Shiratori T, Kobayashi K, Takano Y (2020) Prediction of hierarchical time series using structured regularization and its application to artificial neural networks. PLoS ONE 15(11): e0242099. https://doi.org/10.1371/journal.pone.0242099

Editor: Ivan Kryven, Utrecht University, NETHERLANDS

Received: July 22, 2020

Accepted: October 26, 2020

Published: November 12, 2020

Data Availability Statement: All real-world datasets are available from the e-Stat, a portal site for official Japanese statistics (https://www.e-stat.go.jp/en).

Funding: One of the authors, Ken Kobayashi (KK) is employed by Fujitsu Laboratories Ltd. The founder provided support in the form of salaries for KK, but did not have any additional role in the study design, data collection and analysis.

Competing interests: KK is an employee of Fujitsu Laboratories Ltd. but this does not alter our

Abstract

This paper discusses the prediction of hierarchical time series, where each upper-level time series is calculated by summing appropriate lower-level time series. Forecasts for such hierarchical time series should be coherent, meaning that the forecast for an upper-level time series equals the sum of forecasts for corresponding lower-level time series. Previous methods for making coherent forecasts consist of two phases: first computing base (incoherent) forecasts and then reconciling those forecasts based on their inherent hierarchical structure. To improve time series predictions, we propose a structured regularization method for completing both phases simultaneously. The proposed method is based on a prediction model for bottom-level time series and uses a structured regularization term to incorporate upper-level forecasts into the prediction model. We also develop a backpropagation algorithm specialized for applying our method to artificial neural networks for time series prediction. Experimental results using synthetic and real-world datasets demonstrate that our method is comparable in terms of prediction accuracy and computational efficiency to other methods for time series prediction.

Introduction

Multivariate time series data often have a hierarchical (tree) structure in which each upper-level time series is calculated by summing appropriate lower-level time series. For instance, numbers of tourists are usually counted on a regional basis, such as sites, cities, regions, or countries [1]. Similarly, many companies require regionally aggregated forecasts to support resource allocation decisions [2]. Product demand is often analyzed by category to reduce the overall forecasting burden [3].

Forecasts for such hierarchical time series should be *coherent*, meaning that the forecast for an upper-level time series equals the sum of forecasts for corresponding lower-level time series [4, 5]. Smoothing methods such as the moving average and exponential smoothing are widely

adherence to PLOS ONE policies on sharing data and materials.

used in academia and industry for time series predictions [6, 7]. Although these methods provide coherent forecasts for hierarchical time series, they have low accuracy, especially for rapidly changing time series.

Another common approach for making coherent forecasts is the use of bottom-up and top-down methods [3, 8–10]. These methods first develop base forecasts by separately predicting each time series and then reconcile those base forecasts based on their inherent hierarchical structure. The bottom-up method calculates base forecasts for bottom-level time series and then aggregates them for upper-level time series. In contrast, the top-down method calculates base forecasts only for a root (total) time series and then disaggregates them according to historical proportions of lower-level time series. Park and Nassar [11] considered a hierarchical Bayesian dynamic proportions model for the top-down method to disaggregate upper-level forecasts sequentially. The middle-out method calculates base forecasts for intermediate-level time series and then applies the bottom-up and top-down methods to make upper- and lower-level forecasts. However, the bottom-up method often accumulates prediction errors as the time series level rises, and the top-down method cannot exploit detailed information about lower-level time series. Notably, when base forecasts are unbiased, only the bottom-up method gives unbiased forecasts [12].

Hyndman et al. [12] proposed a linear regression approach to optimal base forecasts by the bottom-up method. This forecast reconciliation method worked well for predicting tourism demand [1] and monthly inflation [13], and this approach can be extended to hierarchical and grouped time series [14]. van Erven and Cugliari [15] devised a game-theoretically optimal reconciliation method. Regularized regression models have also been employed to deal with high-dimensional time series [16, 17]. Wickramasuriya et al. [5] devised a sophisticated method for optimal forecast reconciliation through trace minimization. Their experimental results showed that this trace minimization method performed very well with synthetic and real-world datasets. Note, however, that all of these forecast reconciliation methods consist of two phases: first computing base forecasts and then reconciling those forecasts based on a hierarchical structure. This study aimed to produce better time series predictions by simultaneously completing these two phases.

Structured regularization uses inherent structural relations among explanatory variables to construct a statistical model [18–20]. Various regularization methods have been proposed for multivariate time series [21, 22], hierarchical explanatory variables [23–26], and artificial neural networks [27]. Prediction of multivariate time series is related to multitask learning, which shares useful information among related tasks to enhance the prediction performance for all tasks [28, 29]. Tailored regularization methods have been developed for multitask learning [30, 31] and applied to artificial neural networks [32]. To the best of our knowledge, however, no prior studies have applied structured regularization methods to predictions of hierarchical time series.

In this study, we aimed to develop a structured regularization method that takes full advantage of hierarchical structure for better time series predictions. Our method is based on a prediction model for bottom-level time series and uses a structured regularization term to incorporate upper-level forecasts into the prediction model. This study particularly focused on applying our method to artificial neural networks, which have been effectively used in time series prediction [33–38]. We developed a backpropagation algorithm specialized for our structured regularization model based on artificial neural networks. Experiments involving the application of our method to synthetic and real-world datasets demonstrated that our method was comparable in terms of prediction accuracy and computational efficiency to other methods that develop coherent forecasts for hierarchical time series.

Methods

This section starts with a brief review of forecasts for hierarchical time series. For such time series, we present our structured regularization model and its application to artificial neural networks. A backpropagation algorithm is also described for artificial neural networks with structured regularization.

Forecasts for hierarchical time series

We address the prediction of multivariate time series where each series is represented as a node in a hierarchical (tree) structure. Let y_{it} be an observation of node $i \in N$ at time $t \in T$, where N is the set of nodes, and T is the set of time points. For simplicity, we focus on two-level hierarchical structures. Fig 1 shows the example of a two-level hierarchical structure with $|N| = 7$ nodes, where $| \cdot |$ denotes the number of set elements. The nodes are classified as

$$N = \{1\} \cup M \cup B, \quad M = \{2, 3\}, \quad B = \{4, 5, 6, 7\},$$

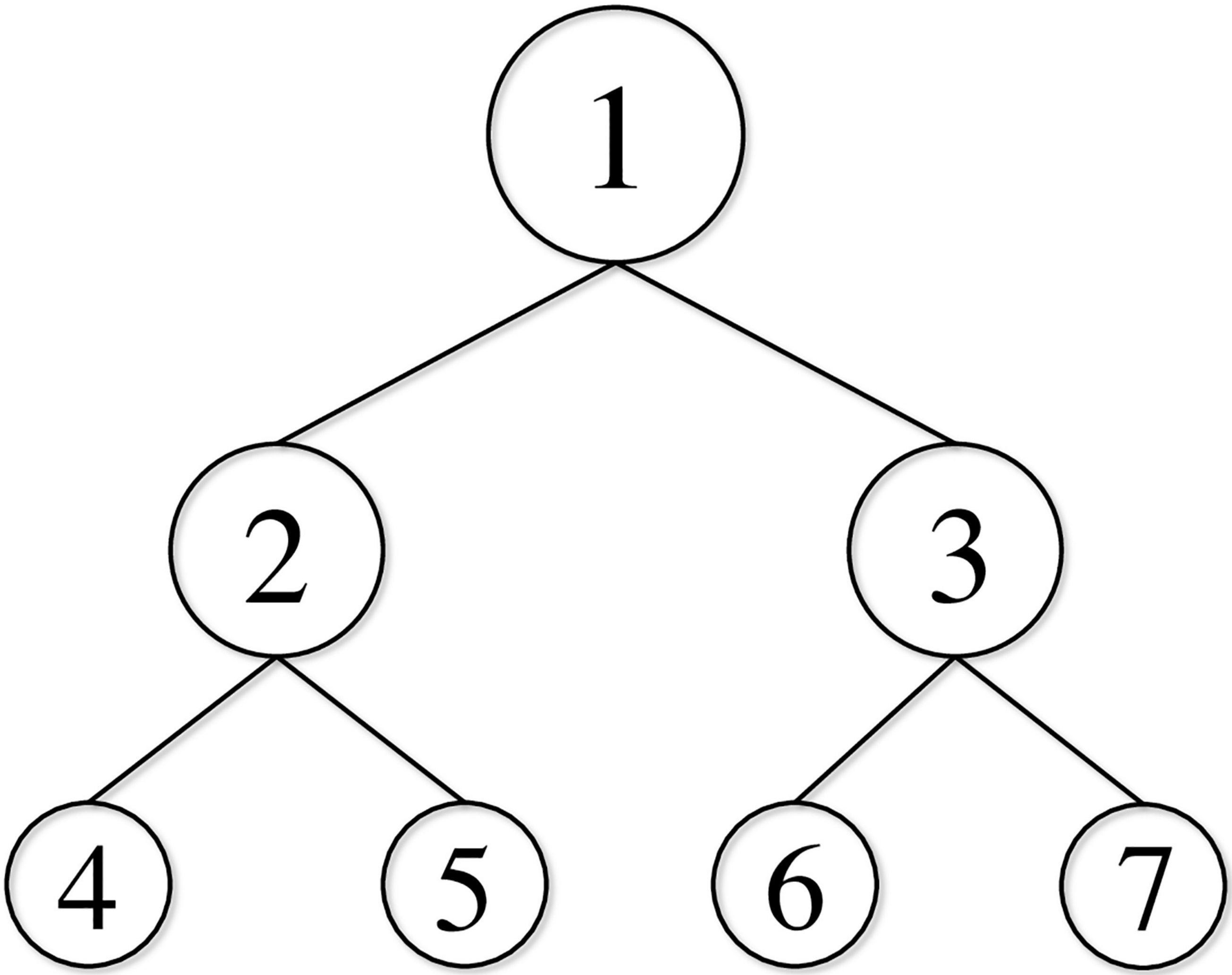

Fig 1. Two-level hierarchical structure with $|N| = 7$.

https://doi.org/10.1371/journal.pone.0242099.g001 where node 1 is the root (level-zero) node, and M and B are sets of mid-level (level-one) and bottom-level (level-two) nodes, respectively. The associated time series is characterized by the *aggregation constraint*

$$\begin{cases} y_{1t} = y_{4t} + y_{5t} + y_{6t} + y_{7t}, \\ y_{2t} = y_{4t} + y_{5t}, \qquad\qquad (t \in T). \\ y_{3t} = y_{6t} + y_{7t}, \end{cases} \tag{1}$$

Each upper-level time series is thus calculated by summing the corresponding lower-level time series.

A hierarchical structure is represented by the *structure matrix* $\boldsymbol{H} := (h_{ki})_{(k,i)\in(N\setminus B)\times B}$ as

$$h_{ki} = \begin{cases} 1 & \text{if node } k \text{ is an ascendant of node } i, \\ 0 & \text{otherwise,} \end{cases} \qquad (k \in N \setminus B,\ i \in B).$$

We define the *summing matrix* as

$$\boldsymbol{S} := (s_{ki})_{(k,i)\in N\times B} := \begin{bmatrix} \boldsymbol{H} \\ \boldsymbol{I}_{|B|} \end{bmatrix},$$

where $\boldsymbol{I}_n$ is the identity matrix of size n. In Fig 1, we have

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}, \quad \boldsymbol{S} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Let $\boldsymbol{y}_t := (y_{it})_{i\in N}$ be a column vector comprising observations of all nodes at time $t \in T$. Similarly, for a node subset $A \subseteq N$ we define $\boldsymbol{y}_t^A := (y_{it})_{i\in A}$ as the observation vector of nodes $i \in A$ at time $t \in T$. In Fig 1, we have

$$\boldsymbol{y}_t = (y_{1t}, y_{2t}, y_{3t}, y_{4t}, y_{5t}, y_{6t}, y_{7t})^\top,$$
$$\boldsymbol{y}_t^M = (y_{2t}, y_{3t})^\top, \quad \boldsymbol{y}_t^B = (y_{4t}, y_{5t}, y_{6t}, y_{7t})^\top \quad (t \in T).$$

The aggregation constraint (1) is then expressed as

$$\boldsymbol{y}_t^{N\setminus B} = \boldsymbol{H}\boldsymbol{y}_t^B \quad (t \in T) \tag{2}$$

or, equivalently,

$$\boldsymbol{y}_t = \boldsymbol{S}\boldsymbol{y}_t^B \quad (t \in T). \tag{3}$$

Let $\hat{\boldsymbol{y}}_t := (\hat{y}_{it})_{i\in N}$ be a column vector comprising *base forecasts* at time $t \in T$. Note that the base forecasts are calculated separately for each node $i \in N$, so they do not satisfy the aggregation constraint (2). For a node subset $A \subseteq N$, we define $\hat{\boldsymbol{y}}_t^A := (\hat{y}_{it})_{i\in A}$ at time $t \in T$. Such base forecasts can be converted into *coherent forecasts* satisfying the aggregation constraint (2) by using the *reconciliation matrix* $\boldsymbol{P} := (p_{ij})_{(i,j)\in B\times N}$. Specifically, we develop bottom-level forecasts $\tilde{\boldsymbol{y}}_t^B = \boldsymbol{P}\hat{\boldsymbol{y}}_t$ and use the aggregation constraint (3) to obtain coherent forecasts, as

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{S}\boldsymbol{P}\hat{\boldsymbol{y}}_t \quad (t \in T). \tag{4}$$

A typical example of a reconciliation matrix is

$$\boldsymbol{P} = [\boldsymbol{O}_{|B|\times|N\setminus B|},\ \boldsymbol{I}_{|B|}],$$

where $\boldsymbol{O}_{m\times n}$ is an $m \times n$ zero matrix. This leads to the bottom-up method

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{S}\hat{\boldsymbol{y}}_t^B \quad (t \in T). \tag{5}$$

Another example is

$$\boldsymbol{P} = [\boldsymbol{p},\ \boldsymbol{O}_{|B|\times|N\setminus\{1\}|}],$$

where $\boldsymbol{p} = (p_i)_{i\in B}$ is a column vector comprising historical proportions of bottom-level time series. This results in the top-down method

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{S}(\hat{y}_{1t}\boldsymbol{p}) \quad (t \in T).$$

In this manner, we can make coherent forecasts from various reconciliation matrices. The condition $\boldsymbol{SPS} = \boldsymbol{S}$ is proven to ensure that when base forecasts are unbiased, the resultant coherent forecasts (4) are also unbiased [12]. This condition is also known to be fulfilled only by the bottom-up method [12].

Forecast reconciliation methods

Hyndman et al. [12] introduced the following linear regression model for given base forecasts:

$$\hat{\boldsymbol{y}}_t = \boldsymbol{S}\boldsymbol{\beta}_t + \boldsymbol{\varepsilon}_t \quad (t \in T),$$

where $\boldsymbol{\beta}_t := (\beta_{it})_{i\in B}$ is a column vector of bottom-level estimates, and $\boldsymbol{\varepsilon}_t := (\varepsilon_{it})_{i\in B}$ is a column vector of errors having zero mean and covariance matrix $\mathrm{var}(\varepsilon_t) := \Sigma_t$. The bottom-up method (5) with $\hat{\boldsymbol{y}}_t^B = \boldsymbol{\beta}_t$ is then used to makes coherent forecasts.

If the base forecasts are unbiased and the covariance matrix Σ_t is known, the generalized least-squares estimation yields the minimum variance unbiased estimate of β_t. However, the covariance matrix Σ_t is nonidentifiable and therefore impossible to estimate [5].

In contrast, Wickramasuriya et al. [5] focused on differences between observations and coherent forecasts (4),

$$e_t := \boldsymbol{y}_t - \tilde{\boldsymbol{y}}_t = \boldsymbol{y}_t - \boldsymbol{SP}\hat{\boldsymbol{y}}_t \quad (t \in T).$$

The associated covariance matrix is derived as

$$\mathrm{var}(e_t) = \boldsymbol{SPW}_t\boldsymbol{P}^\top\boldsymbol{S}^\top \quad (t \in T), \tag{6}$$

where $\boldsymbol{W}_t := \mathbb{E}[(y_t - \hat{y}_t)(y_t - \hat{y}_t)^\top]$ is the covariance matrix of base forecasts. The trace of the covariance matrix (6) is minimized subject to the unbiasedness condition $\boldsymbol{SPS} = \boldsymbol{S}$. This yields the optimal reconciliation matrix

$$\boldsymbol{P} = (\boldsymbol{S}^\top\boldsymbol{W}_t^{-1}\boldsymbol{S})^{-1}\boldsymbol{S}^\top\boldsymbol{W}_t^{-1},$$

and coherent forecasts (4) are given by

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{S}(\boldsymbol{S}^\top \boldsymbol{W}_t^{-1} \boldsymbol{S})^{-1} \boldsymbol{S}^\top \boldsymbol{W}_t^{-1} \hat{\boldsymbol{y}}_t \quad (t \in T). \tag{7}$$

See Wickramasuriya et al. [5] for the full details.

Note, however, that in these forecast reconciliation methods, base forecasts are first determined regardless of the underlying hierarchical structure, then those forecasts are corrected based on the hierarchical structure. In contrast, our proposal is a structured regularization model that directly computes high-quality forecasts based on the hierarchical structure.

Structured regularization model

We consider a prediction model for bottom-level time series. Its predictive value is denoted by the column vector $\hat{\boldsymbol{y}}_t^B(\boldsymbol{\Theta}) \coloneqq (\hat{y}_{it}(\boldsymbol{\Theta}))_{i \in B}$, where $\boldsymbol{\Theta}$ is a tuple of model parameters. As an example, the first-order vector autoregressive model is represented as

$$\hat{y}_{it}(\boldsymbol{\Theta}) = \sum_{j \in B} \theta_{ij} y_{j,t-1} \quad (i \in B,\ t \in T),$$

where $\boldsymbol{\Theta} = (\theta_{ij})_{(i,j) \in B \times B}$.

The residual sum of squares for bottom-level time series is given by

$$\sum_{t \in T} \|\boldsymbol{y}_t^B - \hat{\boldsymbol{y}}_t^B(\boldsymbol{\Theta})\|_2^2 = \sum_{t \in T} \sum_{i \in B} (y_{it} - \hat{y}_{it}(\boldsymbol{\Theta}))^2. \tag{8}$$

We also introduce a structured regularization term that quantifies the error for upper-level time series based on the hierarchical structure. Let $\boldsymbol{\Lambda} \coloneqq \mathrm{Diag}(\boldsymbol{\lambda})$ be a diagonal matrix of regularization parameters, where $\boldsymbol{\lambda} \coloneqq (\lambda_i)_{i \in N \setminus B}$ is a vector of its diagonal entries. Then, we construct a structured regularization term based on the aggregation constraint (2) as

$$\sum_{t \in T} \|\boldsymbol{\Lambda}(\boldsymbol{y}_t^{N \setminus B} - \boldsymbol{H}\hat{\boldsymbol{y}}_t^B(\boldsymbol{\Theta}))\|_2^2. \tag{9}$$

Minimizing this term aids in correcting bottom-level forecasts, thus improving the upper-level forecasts.

Adding the regularization term (9) to the residual sum of squares (8) yields the objective function $E(\boldsymbol{\Theta})$ to be minimized. Consequently, our structured regularization model is posed as

$$\boldsymbol{\Theta}^* \in \underset{\boldsymbol{\Theta}}{\operatorname{argmin}} \left\{ E(\boldsymbol{\Theta}) \coloneqq \frac{1}{2} \sum_{t \in T} \|\boldsymbol{y}_t^B - \hat{\boldsymbol{y}}_t^B(\boldsymbol{\Theta})\|_2^2 + \frac{1}{2} \sum_{t \in T} \|\boldsymbol{\Lambda}(\boldsymbol{y}_t^{N \setminus B} - \boldsymbol{H}\hat{\boldsymbol{y}}_t^B(\boldsymbol{\Theta}))\|_2^2 \right\}. \tag{10}$$

Here, matrix $\boldsymbol{\Lambda}$ adjusts the tradeoff between minimizing the error term (8) for bottom-level times series and minimizing the error term (9) for upper-level time series. In the experiments section, we set its diagonal entries as

$$\lambda_i = \begin{cases} \lambda_1 & (i = 1), \\ \lambda_M & (i \in M), \end{cases} \tag{11}$$

where λ_1 and λ_M are regularization parameters for root and mid-level time series, respectively.

After solving the structured regularization model (10), we use the bottom-up method (5) to obtain coherent forecasts

$$\tilde{\boldsymbol{y}}_t = \boldsymbol{S}\hat{\boldsymbol{y}}_t^B(\boldsymbol{\Theta}^*).$$

Our structured regularization model based on the bottom-up method may not work well when upper-level time series are easier to predict than bottom-level time series. To remedy this situation, we can adopt a methodology proposed by Panagiotelis et al. [39], where the summing matrix is redefined by replacing a bottom-level time series with an upper-level time series.

Application to artificial neural networks

This study focused on application of our structured regularization model (10) to artificial neural networks for time series prediction; see Bishop [40] and Goodfellow et al. [41] for general descriptions of artificial neural networks. For simplicity, we consider a two-layer neural network like the one shown in Fig 2, where the input vector $\boldsymbol{z}^{(1)} := (z_i^{(1)})_{i \in B}$ is defined as

$$z_i^{(1)} = y_{i,t-1} \quad (i \in B).$$

First, we calculate the vector $\boldsymbol{u}^{(2)} := (u_j^{(2)})_{j \in D}$ as the weighted sum of the input entries

$$u_j^{(2)} = \sum_{i \in B} w_{ji}^{(2)} z_i^{(1)} \quad (j \in D), \tag{12}$$

where $\boldsymbol{W}^{(2)} := (w_{ji}^{(2)})_{(j,i) \in D \times B}$ is a weight matrix to be estimated. This vector $\boldsymbol{u}^{(2)}$ is transferred from the input units to *hidden units*, as shown in Fig 2.

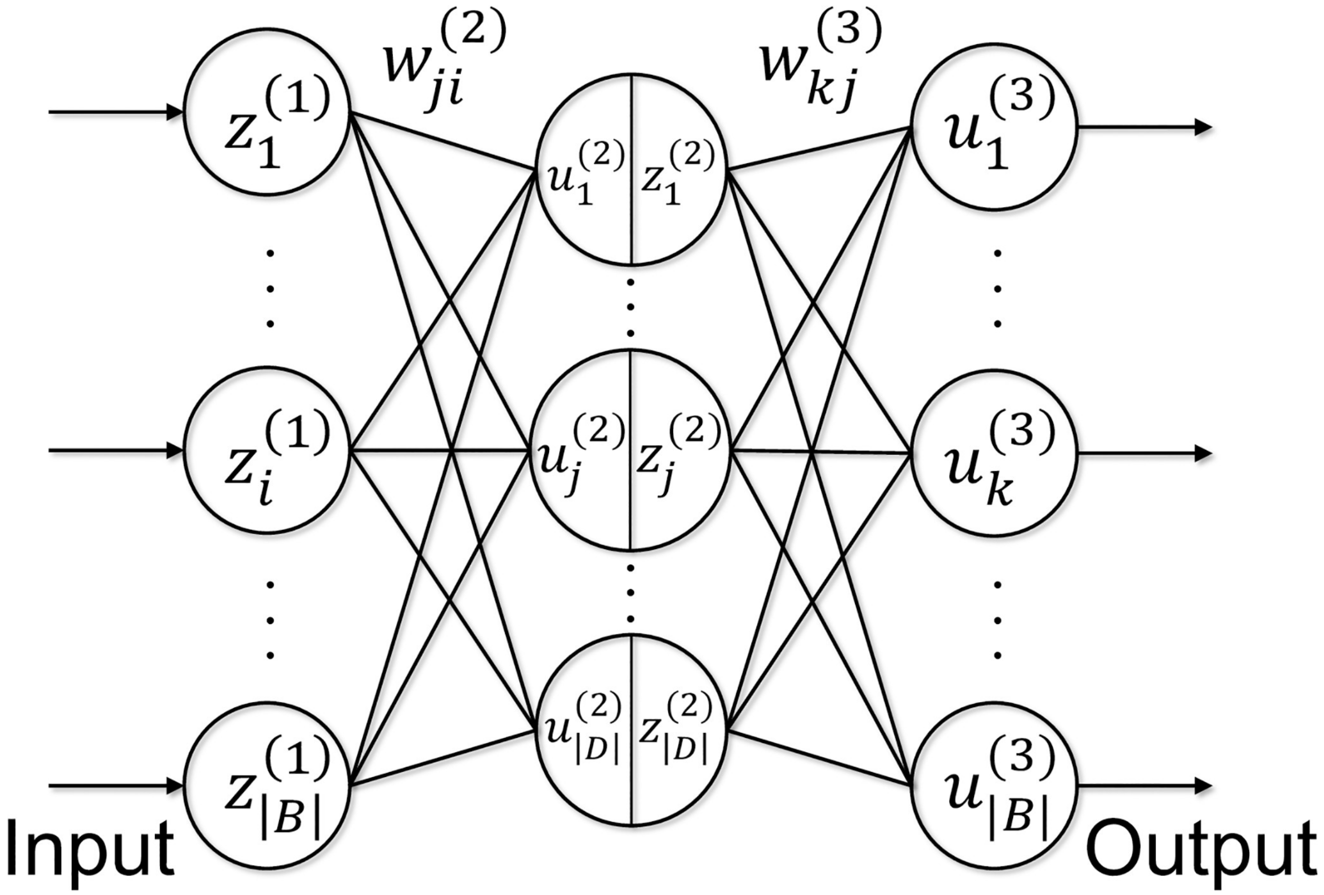

Fig 2. Network diagram for a two-layer neural network.

https://doi.org/10.1371/journal.pone.0242099.g002

Next, we generate the vector $\boldsymbol{z}^{(2)} := (z_j^{(2)})_{j \in D}$ by nonlinear *activation functions* as

$$z_j^{(2)} = f(u_j^{(2)}) \quad (j \in D).$$

Typical examples of activation functions include the logistic sigmoid function

$$f(u) = \frac{1}{1 + \exp(-u)} \tag{13}$$

and the rectified linear function

$$f(u) = \max\{u, 0\}.$$

The vector $\boldsymbol{z}^{(2)}$ is transferred from the hidden units to the output units as shown in Fig 2.

Finally, we calculate the vector $\boldsymbol{u}^{(3)} := (u_k^{(3)})_{k \in B}$ as the weighted sum of the output entries from the hidden units as

$$u_k^{(3)} = \sum_{j \in D} w_{kj}^{(3)} z_j^{(2)} = \sum_{j \in D} w_{kj}^{(3)} f(u_j^{(2)}) \quad (k \in B), \tag{14}$$

where $\boldsymbol{W}^{(3)} := (w_{kj}^{(3)})_{(k,j) \in B \times D}$ is a weight matrix to be estimated.

This process is summarized as

$$\boldsymbol{z}^{(1)} = \boldsymbol{y}_{t-1}^{B}, \quad \boldsymbol{u}^{(2)} = \boldsymbol{W}^{(2)} \boldsymbol{z}^{(1)}, \quad \boldsymbol{z}^{(2)} = \boldsymbol{f}(\boldsymbol{u}^{(2)}), \quad \boldsymbol{u}^{(3)} = \boldsymbol{W}^{(3)} \boldsymbol{z}^{(2)}, \tag{15}$$

where the tuple of model parameters is

$$\boldsymbol{\Theta} = (\boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)}).$$

This neural network outputs $\hat{\boldsymbol{y}}_t^B(\boldsymbol{\Theta}) = \boldsymbol{u}^{(3)}$ as a vector of predictive values.

Backpropagation algorithm

We develop a backpropagation algorithm specialized for training artificial neural networks in our structured regularization model (10); see Bishop [40] and Goodfellow et al. [41] for overviews of backpropagation algorithms. Our algorithm sequentially minimizes the following error function for time $t \in T$:

$$E_t(\boldsymbol{\Theta}) := \frac{1}{2} \|\boldsymbol{y}_t^B - \boldsymbol{u}^{(3)}\|_2^2 + \frac{1}{2} \|\boldsymbol{\Lambda}(\boldsymbol{y}_t^{N \setminus B} - \boldsymbol{H}\boldsymbol{u}^{(3)})\|_2^2 \quad (t \in T). \tag{16}$$

We first define vectors $\boldsymbol{\delta}^{(2)} := (\delta_j^{(2)})_{j \in D}$ and $\boldsymbol{\delta}^{(3)} := (\delta_k^{(3)})_{k \in B}$, which consist of partial derivatives of the error function (16) with respect to intermediate variables (12) and (14) as follows:

$$\delta_j^{(2)} := \frac{\partial E_t(\boldsymbol{\Theta})}{\partial u_j^{(2)}} \quad (j \in D), \qquad \delta_k^{(3)} := \frac{\partial E_t(\boldsymbol{\Theta})}{\partial u_k^{(3)}} \quad (k \in B).$$

From Eqs (12) and (14), the partial derivatives of the error function (16) can be calculated as

$$\frac{\partial E_t(\boldsymbol{\Theta})}{\partial w_{ji}^{(2)}} = \frac{\partial E_t(\boldsymbol{\Theta})}{\partial u_j^{(2)}}\frac{\partial u_j^{(2)}}{\partial w_{ji}^{(2)}} = \delta_j^{(2)} z_i^{(1)} \quad (i \in B,\ j \in D), \tag{17}$$

$$\frac{\partial E_t(\boldsymbol{\Theta})}{\partial w_{kj}^{(3)}} = \frac{\partial E_t(\boldsymbol{\Theta})}{\partial u_k^{(3)}}\frac{\partial u_k^{(3)}}{\partial w_{kj}^{(3)}} = \delta_k^{(3)} z_j^{(2)} \quad (j \in D,\ k \in B). \tag{18}$$

From Eq (14), we have

$$\frac{\partial u_k^{(3)}}{\partial u_j^{(2)}} = w_{kj}^{(3)} f'(u_j^{(2)}) \quad (j \in D,\ k \in B).$$

Therefore,

$$\delta_j^{(2)} = \frac{\partial E_t(\boldsymbol{\Theta})}{\partial u_j^{(2)}} = \sum_{k\in B}\frac{\partial E_t(\boldsymbol{\Theta})}{\partial u_k^{(3)}}\frac{\partial u_k^{(3)}}{\partial u_j^{(2)}} = \sum_{k\in B}\delta_k^{(3)} w_{kj}^{(3)} f'(u_j^{(2)}) \quad (j \in D). \tag{19}$$

It follows from Eq (16) that

$$\begin{aligned}\boldsymbol{\delta}^{(3)} := \frac{\partial E_t(\boldsymbol{\Theta})}{\partial \boldsymbol{u}^{(3)}} &= -(\boldsymbol{y}_t^B - \boldsymbol{u}^{(3)}) - (\boldsymbol{\Lambda H})^\top \boldsymbol{\Lambda}(\boldsymbol{y}_t^{N\setminus B} - \boldsymbol{H u}^{(3)}) \\ &= -(\boldsymbol{y}_t^B - \boldsymbol{u}^{(3)}) - \boldsymbol{H}^\top \boldsymbol{\Lambda}^2(\boldsymbol{y}_t^{N\setminus B} - \boldsymbol{H u}^{(3)}) \\ &= -[\boldsymbol{H}^\top \boldsymbol{\Lambda}^2,\ \boldsymbol{I}_{|B|}]y_t + (\boldsymbol{I}_{|B|} + \boldsymbol{H}^\top \boldsymbol{\Lambda}^2 \boldsymbol{H})\boldsymbol{u}^{(3)}.\end{aligned} \tag{20}$$

Algorithm 1 summarizes our backpropagation algorithm.

```
**Algorithm 1** Backpropagation algorithm.
**Step 0 (Initialization):** Let $\eta > 0$ be a step size and $\varepsilon > 0$ be a thresh-
old for convergence. Set $\boldsymbol{\Theta} = (\boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)})$ as initial parameter values,
and $E \leftarrow E(\boldsymbol{\Theta}) = \sum_{t\in T} E_T(\boldsymbol{\Theta})$ as an incumbent value of the objective
function.
**Step 1 (Backpropagation):** Repeat the following steps for all $t \in T$:
  **Step 1.1:** Compute $\boldsymbol{z}^{(1)}$, $\boldsymbol{u}^{(2)}$, $\boldsymbol{z}^{(2)}$, and $\boldsymbol{u}^{(3)}$ from Eq (15).
  **Step 1.2:** Compute $\boldsymbol{\delta}^{(3)}$ from Eq (20) and then $\boldsymbol{\delta}^{(2)}$ from Eq (19).
  **Step 1.3:** Compute the partial derivatives (17) and (18).
**Step 2 (Gradient Descent):** Update the weight parameter values as
```

$$\begin{cases} w_{ji}^{(2)} \leftarrow w_{ji}^{(2)} - \eta \sum_{t\in T} \dfrac{\partial E_i(\boldsymbol{\Theta})}{\partial w_{ji}^{(2)}} & (i \in B, j \in D), \\ w_{ji}^{(3)} \leftarrow w_{ji}^{(3)} - \eta \sum_{t\in T} \dfrac{\partial E_i(\boldsymbol{\Theta})}{\partial w_{ji}^{(3)}} & (j \in D, k \in B), \end{cases}$$

```
**Step 3(Termination Condition):** If $E(\boldsymbol{\Theta}) > (1 - \varepsilon)E$, terminate the
algorithm with $\boldsymbol{\Theta} = (\boldsymbol{W}^{(2)}, \boldsymbol{W}^{(3)})$. Otherwise, set $E \leftarrow E(\boldsymbol{\Theta})$ and return to
Step 1.
```

Experimental results and discussion

The experimental results reported in this section evaluate the effectiveness of our structured regularization model when applied to artificial neural networks. These experiments focused on

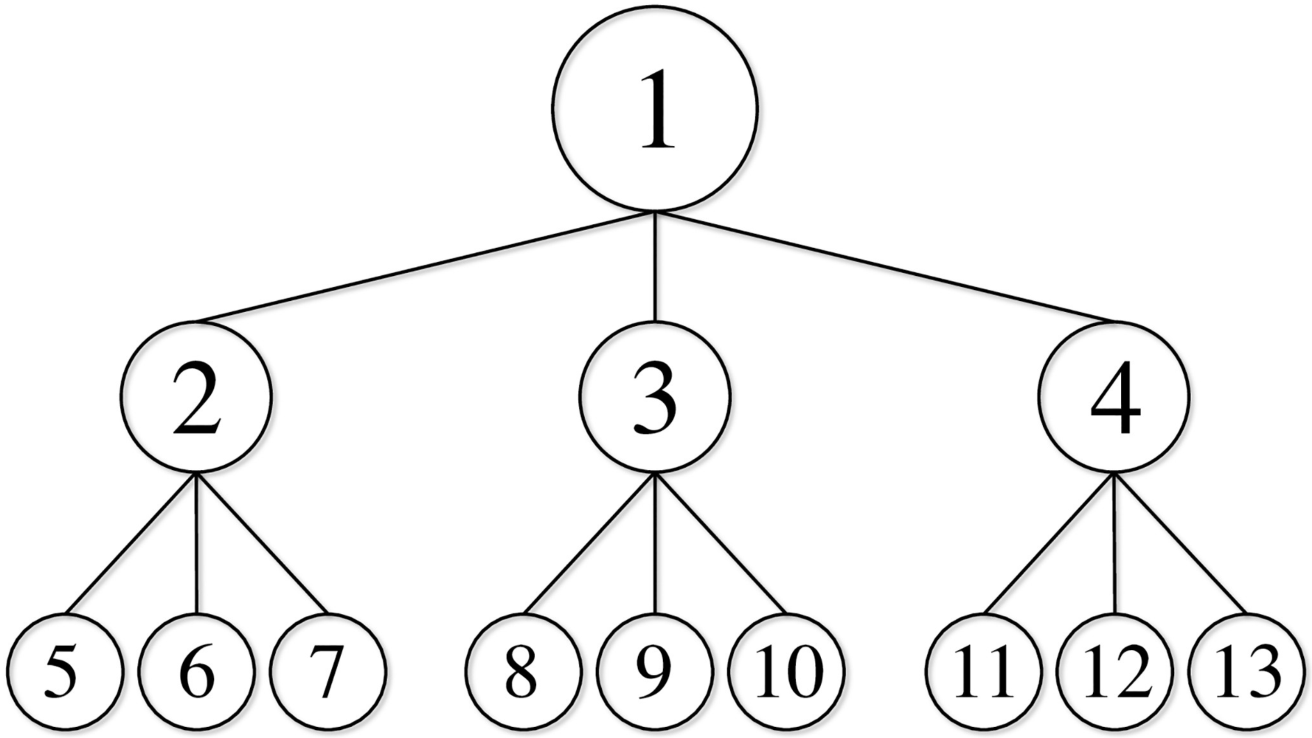

Fig 3. Two-level hierarchical structure with $|N| = 13$.

https://doi.org/10.1371/journal.pone.0242099.g003

the two-level hierarchical structure shown in Fig 3, where

$$N = \{1\} \cup M \cup B, \quad M = \{2, 3, 4\}, \quad B = \{5, 6, \ldots, 13\}.$$

Performance evaluation methodology

To evaluate out-of-sample prediction performance, we considered training and test periods of time series data, where the training period was used to train prediction models, and the test period was used to compute prediction errors in the trained models. We calculated the root-mean-squared error (RMSE) for each node $i \in N$ during the test period $\hat{T}$ as

$$\text{RMSE} := \sqrt{\frac{\sum_{t \in \hat{T}} (y_{it} - \tilde{y}_{it})^2}{|\hat{T}|}} \quad (i \in N).$$

We compared the performance of the following methods for time series prediction.

MA(n): moving average of the previous n values,

$$\tilde{y}_{it} = \frac{\sum_{k=1}^{n} y_{i,t-k}}{n} \ (i \in N, \ t \in T)$$

ES(α): exponential smoothing with a smoothing parameter $\alpha \in [0, 1]$,

$$\tilde{y}_{it} = \alpha y_{i,t-1} + (1 - \alpha)\tilde{y}_{i,t-1} \ (i \in N, \ t \in T)$$

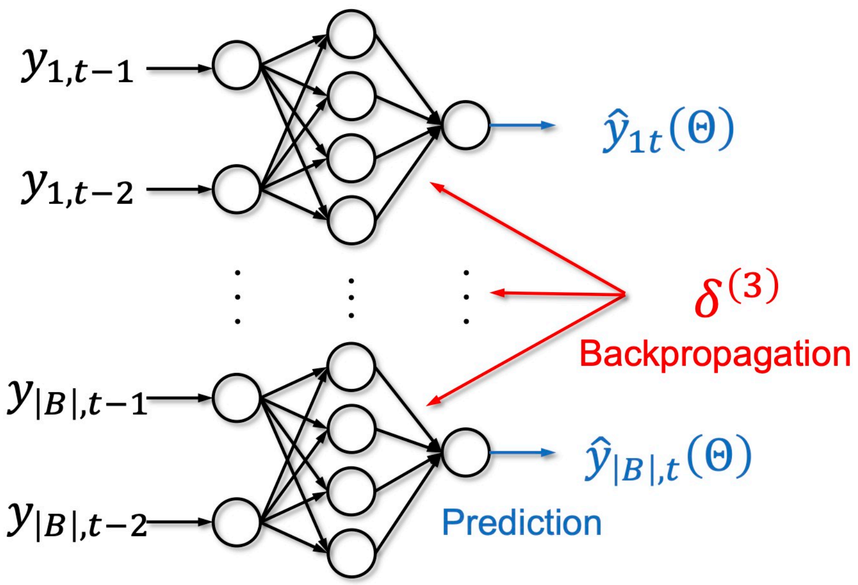

Fig 4. Two-layer neural network adopted in experimental results.

https://doi.org/10.1371/journal.pone.0242099.g004

NN+BU: bottom-up method (5) using artificial neural networks for base forecasts $\hat{y}_t^B$

NN+MinT: forecast reconciliation method (7) through the trace minimization (i.e., MinT (Shrink) [5]) using artificial neural networks for base forecasts $\hat{y}_t$

NN+SR(λ_1, λ_M): our structured regularization model (10) applied to artificial neural networks with regularization parameters λ_1 and λ_M; see also Eq (11)

Here, we determined parameter values for n and α that minimized RMSE in the training period. During the training period, we tuned regularization parameters λ_1 and λ_M through hold-out validation [42].

We adopted two-layer artificial neural networks (Fig 4) for NN+BU, NN+MinT, and NN +SR. Here, prediction $\hat{y}_{it}(\mathbf{\Theta})$ of each time series depends on its own two lags $y_{i,t-1}$ and $y_{i,t-2}$, and the backpropagation simultaneously updates weight parameters of all the series. Note also that NN+BU is equivalent to NN+SR(0,0). Following prior studies [43, 44], we set the number of hidden units to twice the number of input units (i.e., $|D| = 4 \cdot |B|$). Bias parameters were added to hidden and output units.

We implemented the backpropagation algorithm (Algorithm 1) in the R programming language, with the convergence threshold set as $\varepsilon = 5 \cdot 10^{-5}$. The step size was kept constant and set as $\eta = 1 \cdot 10^{-5}$, which was small enough for the backpropagation algorithm to converge. We employ the logistic sigmoid function (13) as an activation function. The algorithm was repeated 30 times by randomly generating initial values for the parameter $\mathbf{\Theta}$ from a standard normal distribution. The following sections show average RMSE values with 95% confidence intervals over the 30 trials.

Synthetic datasets

We generated common factors to express correlations among time series. Denote as $N(\mu, \sigma^2)$ a normal distribution with mean μ and standard deviation σ. For common factors, we used the first-order autoregressive models

$$\psi_{it} \sim \mathrm{N}(\phi_i \psi_{i,t-1}, \sigma_i^2) \ \ (i \in \{1\} \cup M, \ \ t \in T),$$

where ϕ_i is an autoregressive coefficient, and σ_i is the standard deviation of white noise for the ith common factor. Note that ψ_{it} reflects the overall trend for $i = 1$ and mid-level trends for $i \in M = \{2, 3, 4\}$.

Bottom-level time series were produced by combining the overall trend, mid-level trends, and autocorrelation. We denote the parent (mid-level) node of node i as

$$m(i) = \begin{cases} 2 & (i \in \{5, 6, 7\}), \\ 3 & (i \in \{8, 9, 10\}), \\ 4 & (i \in \{11, 12, 13\}). \end{cases}$$

For bottom-level time series, we used the first-order autoregressive models

$$y_{it} \sim \mathrm{N}(\rho_i \psi_{1t} + \theta_i \psi_{m(i),t} + \phi_i y_{i,t-1}, \sigma_i^2) \quad (i \in B, \ \ t \in T),$$

where ρ_i and θ_i respectively indicate effects of the common factors ψ_{1t} and $\psi_{m(i),\,t}$ on the ith time series. After that, we generated upper-level time series (y_{it} for $i \in N\backslash B$) according to the aggregation constraint (2).

We prepared three synthetic datasets: NgtvC, WeakC, and PstvC. Table 1 lists the parameter values used to generate these datasets. Time series are negatively correlated in the NgtvC dataset, weakly correlated in the WeakC dataset, and positively correlated in the PstvC dataset. Each dataset consists of time series data at 100 time points; the first 70 and latter 30 times were used as training and test periods, respectively.

We standardized the generated time series according to the mean and variance over the training period when using artificial neural networks. Specifically, we standardized each bottom-level time series for NN+BU and NN+SR and summed them appropriately to calculate upper-level time series for NN+SR. We standardized each time series at all levels for NN +MinT. We then computed predictive values for these time series and transformed the obtained predictive values into the original (unstandardized) scale. After that, we applied the bottom-up method (NN+BU and NN+SR) and the forecast reconciliation method

Table 1. Parameter values for the synthetic datasets.

			NgtvC		WeakC		PstvC	
Node i	ϕ_i	σ_i	ρ_i	θ_i	ρ_i	θ_i	ρ_i	θ_i
1	0.3	0.3	—	—	—	—	—	—
2	0.3	0.3	—	—	—	—	—	—
3	0.3	0.3	—	—	—	—	—	—
4	0.3	0.3	—	—	—	—	—	—
5	0.3	0.3	0.1	1.0	0.1	0.1	1.0	1.0
6	0.3	0.3	−0.1	−1.0	0.1	0.1	1.0	1.0
7	0.3	0.3	1.0	0.1	0.1	0.1	1.0	1.0
8	0.3	0.3	0.1	1.0	0.1	0.1	1.0	1.0
9	0.3	0.3	−0.1	−1.0	0.1	0.1	1.0	1.0
10	0.3	0.3	−1.0	0.1	0.1	0.1	1.0	1.0
11	0.3	0.3	0.1	1.0	0.1	0.1	1.0	1.0
12	0.3	0.3	−0.1	−1.0	0.1	0.1	1.0	1.0
13	0.3	0.3	1.0	0.1	0.1	0.1	1.0	1.0

https://doi.org/10.1371/journal.pone.0242099.t001

(NN+MinT) to make coherent forecasts. Finally, we calculated RMSEs on the original scale to evaluate prediction performance.

Results for synthetic datasets

Tables 2–4 list the out-of-sample RMSE values provided by each method for each node in the NgtvC, WeakC, and PstvC datasets. In the tables, the rows labeled "Mid-level" and "Bottom-level" show the average RMSE values over the mid- and bottom-level nodes, respectively, with

Table 2. Prediction performance for the NgtvC dataset.

	RMSE				
Node *i*	**MA(12)**	**ES(0.20)**	**NN+BU**	**NN+MinT**	**NN+SR(0.0, 2.1)**
Root	**1.09**	1.10	1.16 ± 0.04	**1.09** ± 0.01	1.15 ± 0.01
2	0.63	0.64	0.66 ± 0.03	**0.60** ± 0.01	**0.60** ± 0.01
3	0.80	0.76	0.76 ± 0.01	**0.73** ± 0.01	**0.73** ± 0.01
4	0.71	0.72	0.70 ± 0.02	0.69 ± 0.01	**0.67** ± 0.01
Mid-level	0.71	0.71	0.71 ± 0.01	**0.67** ± 0.01	**0.67** ± 0.01
5	0.53	0.48	0.48 ± 0.02	0.47 ± 0.02	**0.44** ± 0.01
6	0.67	**0.64**	0.65 ± 0.02	**0.64** ± 0.01	**0.64** ± 0.02
7	0.39	**0.37**	0.38 ± 0.00	0.39 ± 0.01	0.38 ± 0.00
8	0.42	**0.39**	0.41 ± 0.01	0.41 ± 0.01	0.41 ± 0.01
9	**0.35**	**0.35**	0.38 ± 0.01	0.38 ± 0.01	0.38 ± 0.01
10	0.58	0.56	0.55 ± 0.02	**0.53** ± 0.01	**0.53** ± 0.01
11	0.58	0.49	**0.47** ± 0.01	**0.47** ± 0.01	**0.47** ± 0.01
12	0.50	0.47	0.47 ± 0.01	**0.46** ± 0.01	**0.46** ± 0.00
13	0.48	0.48	0.47 ± 0.01	0.47 ± 0.01	**0.46** ± 0.01
Bottom-level	0.50	0.47	0.47 ± 0.00	0.47 ± 0.00	**0.46** ± 0.00
Average	0.60	0.57	0.58 ± 0.00	**0.56** ± 0.00	**0.56** ± 0.00

https://doi.org/10.1371/journal.pone.0242099.t002

Table 3. Prediction performance for the WeakC dataset.

	RMSE				
Node *i*	**MA(12)**	**ES(0.00)**	**NN+BU**	**NN+MinT**	**NN+SR(0.0, 1.2)**
Root	1.06	**1.00**	1.06 ± 0.02	1.05 ± 0.01	1.06 ± 0.01
2	0.46	**0.41**	0.45 ± 0.01	0.43 ± 0.01	0.44 ± 0.01
3	0.60	**0.56**	0.60 ± 0.01	0.58 ± 0.01	0.58 ± 0.01
4	0.61	0.60	0.59 ± 0.01	0.58 ± 0.01	**0.57** ± 0.01
Mid-level	0.56	**0.52**	0.55 ± 0.00	0.53 ± 0.01	0.53 ± 0.00
5	0.32	**0.30**	0.31 ± 0.01	0.31 ± 0.01	**0.30** ± 0.00
6	0.39	**0.37**	0.39 ± 0.01	0.38 ± 0.01	0.38 ± 0.01
7	**0.24**	**0.24**	0.25 ± 0.00	0.25 ± 0.01	**0.24** ± 0.00
8	0.30	**0.29**	0.33 ± 0.01	0.32 ± 0.01	0.31 ± 0.01
9	0.27	**0.26**	0.27 ± 0.01	0.27 ± 0.01	**0.26** ± 0.01
10	0.37	**0.34**	0.35 ± 0.01	**0.34** ± 0.01	0.35 ± 0.01
11	0.39	0.36	0.34 ± 0.01	0.34 ± 0.01	**0.33** ± 0.01
12	0.37	**0.36**	0.37 ± 0.01	**0.36** ± 0.01	**0.36** ± 0.01
13	0.29	0.29	0.29 ± 0.00	0.29 ± 0.01	**0.28** ± 0.00
Bottom-level	0.33	**0.31**	0.32 ± 0.00	0.32 ± 0.00	**0.31** ± 0.00
Average	0.44	**0.41**	0.43 ± 0.00	0.42 ± 0.00	0.42 ± 0.00

https://doi.org/10.1371/journal.pone.0242099.t003

Table 4. Prediction performance for the PstvC dataset.

	RMSE				
Node *i*	**MA(1)**	**ES(0.89)**	**NN+BU**	**NN+MinT**	**NN+SR(0.4, 2.4)**
Root	2.69	2.69	2.90 ± 0.05	2.58 ± 0.07	**2.49** ± 0.03
2	1.20	1.20	1.33 ± 0.03	1.16 ± 0.03	**1.12** ± 0.01
3	1.49	1.42	**1.12** ± 0.01	1.16 ± 0.03	1.27 ± 0.02
4	1.11	1.11	1.25 ± 0.03	1.63 ± 0.02	**1.06** ± 0.01
Mid-level	1.27	1.24	1.23 ± 0.01	1.32 ± 0.03	**1.15** ± 0.01
5	0.53	0.53	0.56 ± 0.01	**0.50** ± 0.01	0.53 ± 0.01
6	0.49	0.49	0.55 ± 0.02	**0.48** ± 0.01	0.51 ± 0.02
7	0.46	0.46	0.49 ± 0.01	0.46 ± 0.01	**0.43** ± 0.01
8	0.56	0.55	**0.48** ± 0.01	**0.48** ± 0.01	0.54 ± 0.03
9	0.57	0.54	**0.42** ± 0.01	0.46 ± 0.02	0.53 ± 0.04
10	0.49	0.47	0.43 ± 0.01	**0.42** ± 0.01	0.45 ± 0.01
11	**0.48**	**0.48**	0.55 ± 0.02	0.51 ± 0.01	0.49 ± 0.02
12	0.59	0.57	0.51 ± 0.01	**0.48** ± 0.01	0.55 ± 0.02
13	0.44	0.44	0.45 ± 0.02	**0.42** ± 0.01	0.43 ± 0.01
Bottom-level	0.51	0.50	0.49 ± 0.00	**0.47** ± 0.01	0.50 ± 0.00
Average	0.85	0.84	0.85 ± 0.00	0.83 ± 0.01	**0.80** ± 0.00

https://doi.org/10.1371/journal.pone.0242099.t004

smallest RMSE values for each node indicated in bold. Note that average RMSE values with 95% confidence intervals are shown for NN+BU, NN+MinT, and NN+SR.

For the NgtvC dataset (Table 2), our structured regularization method NN+SR was comparable to the forecast reconciliation method and outperformed the other methods, except for the RMSE of the root node. For the WeakC dataset (Table 3), our method was slightly inferior to the exponential smoothing method, but the differences were minimal. For the PstvC dataset (Table 4), our method attained the best prediction performance in terms of average RMSE. These results show that our structured regularization method delivered good prediction performance for the three synthetic datasets. Our method was especially effective when the time series were strongly correlated, as in the NgtvC and PstvC datasets.

We next focus on the parameter values for our structured regularization. Only for the PstvC dataset, our method NN+SR(λ_1, λ_M) adopted $\lambda_1 > 0$ and performed significantly better than the bottom-up method in terms of the RMSE of the root node. Additionally, our method employed $\lambda_M > 0$ for all three datasets and outperformed the bottom-up method for mid-level RMSEs. These results show an association between regularization weights and prediction accuracy at each time series level. Our method adjusts the regularization parameters to fit the data characteristic, thereby achieving better prediction performance.

Fig 5 shows the training RMSE values as a function of the epoch (number of iterations) in the backpropagation algorithm for the synthetic datasets. Note that the computational efficiency can be evaluated based on epochs because little difference existed between NN+SR and NN+BU in the computation time required for one epoch.

RMSEs decreased faster for our structured regularization method NN+SR than for the bottom-up method NN+BU. The convergence performance of the two methods greatly differed, especially for the PstvC dataset and upper-level time series. Consequently, our structured regularization method improved both prediction accuracy and convergence speed of the backpropagation algorithm. This suggests that our method will deliver good prediction performance even if the backpropagation algorithm is terminated in the middle of computation.

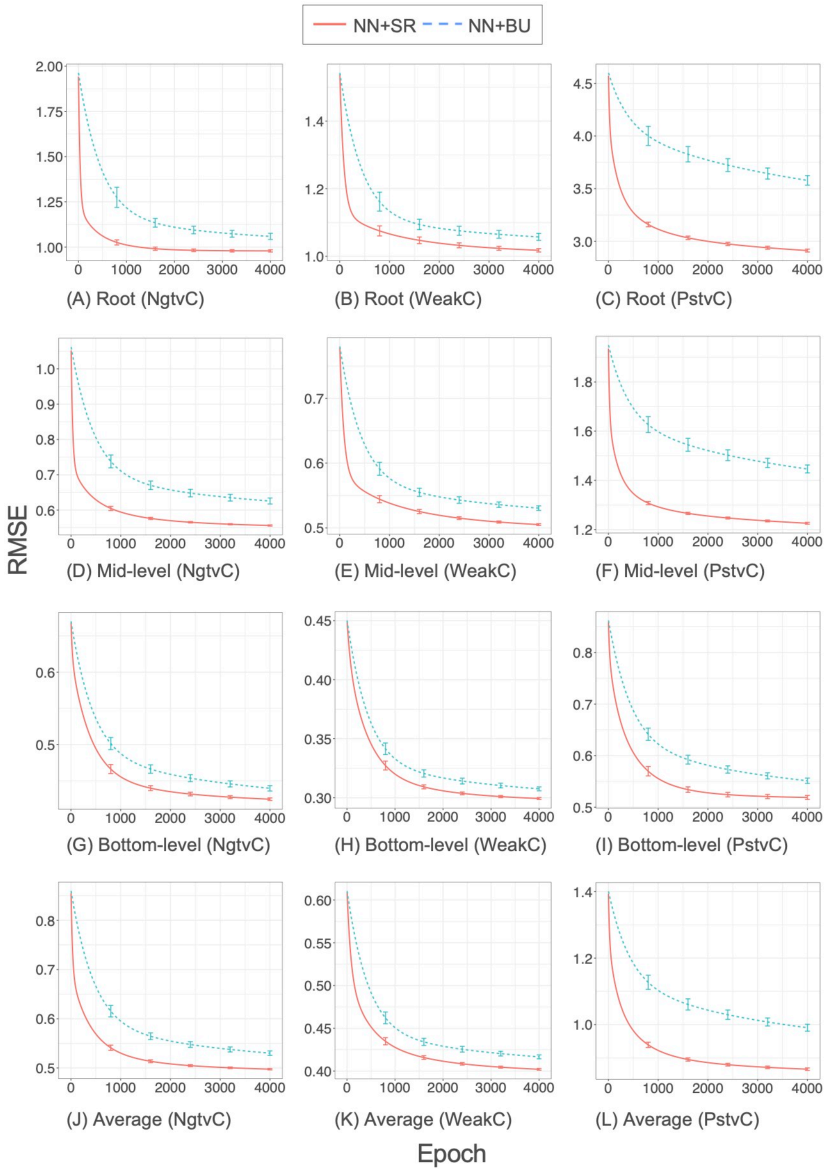

Fig 5. Convergence of the backpropagation algorithm for the synthetic datasets.

https://doi.org/10.1371/journal.pone.0242099.g005

Fig 6 shows heat maps of the out-of-sample relative RMSE values provided by our structured regularization method NN+SR(λ_1, λ_M) for the synthetic datasets. Here, the vertical and horizontal axes are the values of regularization parameters λ_1 and λ_M, respectively. This figure shows how regularization for each time series level affects the prediction performance.

Note that RMSE values were normalized in Fig 6 such that the RMSE for (λ_1, λ_M) = (0, 0) was zero (white-colored) in each trial. Accordingly, the corresponding regularization is effective if the relative RMSE value is negative (red-colored), and it is ineffective if the relative RMSE value is positive (blue-colored). RMSEs were consistently reduced in the NgtvC dataset. Regularization was particularly effective for the root time series in the PstvC dataset. RMSE values tend to vary drastically from left to right in each heat map, which suggests that the regularization for the mid-level time series greatly impacted prediction performance.

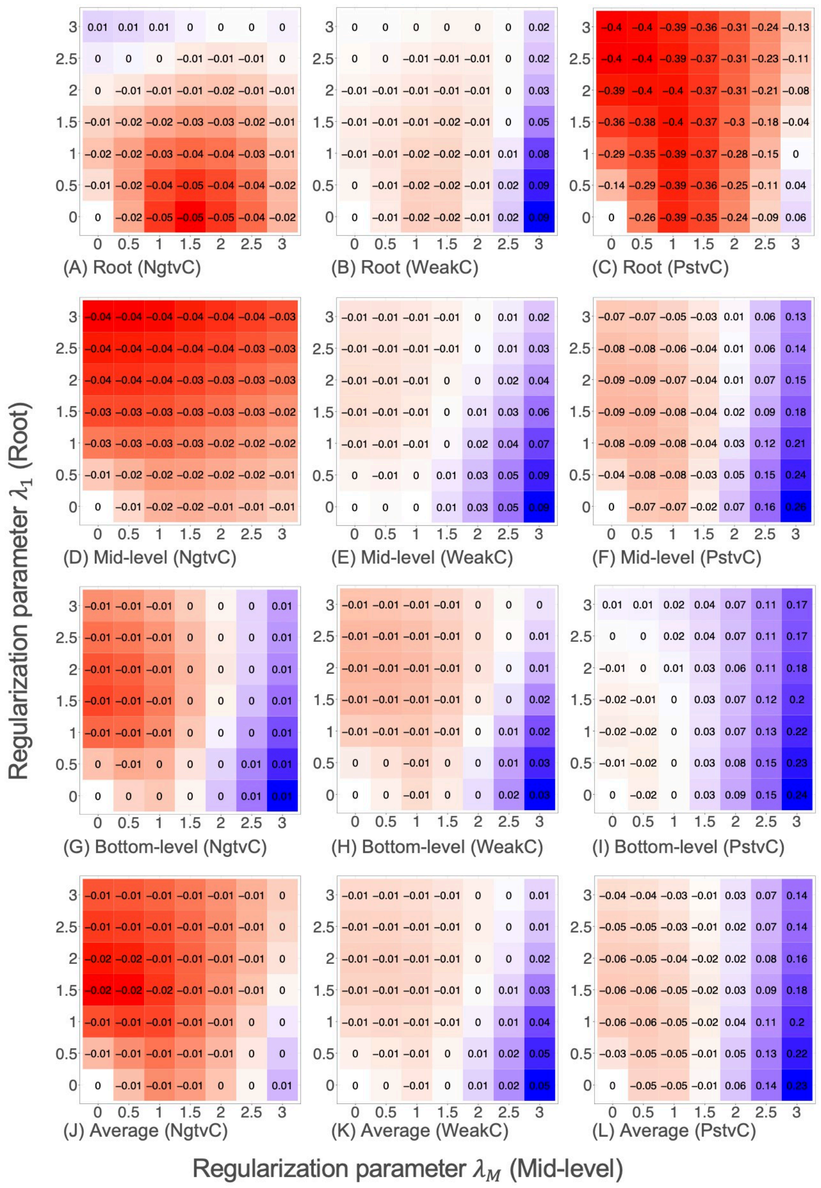

Fig 6. Heat maps of relative RMSEs provided by NN+SR(λ_1, λ_M) in the synthetic datasets.

https://doi.org/10.1371/journal.pone.0242099.g006

Real-world datasets

We downloaded historical data describing unemployment rates in Japan from e-Stat, a portal site for official Japanese statistics (https://www.e-stat.go.jp/en). Using these data, we prepared three real-world datasets for Japanese regions: Tohoku, Chubu, and Kansai. Table 5 lists the prefectures forming the two-level hierarchical structure (Fig 3).

We used quarterly statistics (model-based estimates) of unemployment rates during 90 time periods from January 1997 to June 2019, taking the first 60 and last 30 time periods as the training and test periods, respectively. As a preprocessing step, we removed seasonal and trend components by means of the `stl` function in the R `stats` package. We next calculated upper-level time series according to the aggregation constraint (2). After that, we standardized

Table 5. List of prefectures in the real-world datasets.

	Prefectures		
Node *i*	**Tohoku**	**Chubu**	**Kanasi**
5	Aomori	Niigata	Mie
6	Iwate	Toyama	Shiga
7	Miyagi	Ishikawa	Kyoto
8	Akita	Fukui	Osaka
9	Yamagata	Yamanashi	Hyogo
10	Fukushima	Nagano	Nara
11	Ibaraki	Gifu	Wakayama
12	Tochigi	Shizuoka	Tottori
13	Gunma	Mie	Okayama

https://doi.org/10.1371/journal.pone.0242099.t005

time series, computed predicted values, and calculated RMSEs on the original scale, in the same way as for the synthetic datasets.

Results for real-world datasets

Tables 6–8 list the out-of-sample RMSE values provided by each method for each node in the Tohoku, Chubu, and Kansai datasets. For the Tohoku dataset (Table 6), our structured regularization method NN+SR was comparable to the forecast reconciliation method and substantially outperformed the other methods. For the Chubu dataset (Table 7), our method attained the second-best value for average RMSE.

For the Kansai dataset (Table 8), our method greatly exceeded the prediction performance of the other methods. These results demonstrate that our structured regularization method achieved good prediction performance for the three real-world datasets.

Fig 7 shows the training RMSE values as a function of epoch in the backpropagation algorithm for the real-world datasets. The convergence of RMSE values was consistently faster for

Table 6. Prediction performance for the Tohoku dataset.

	RMSE				
Node *i*	**MA(20)**	**ES(0.12)**	**NN+BU**	**NN+MinT**	**NN+SR(0.4, 1.5)**
Root	6.32	6.45	5.98 ± 0.07	5.87 ± 0.06	**5.70** ± 0.06
2	2.83	2.91	2.77 ± 0.05	**2.65** ± 0.05	2.66 ± 0.03
3	2.06	2.13	2.02 ± 0.05	2.04 ± 0.04	**2.01** ± 0.03
4	2.86	2.92	2.70 ± 0.04	2.68 ± 0.03	**2.63** ± 0.01
Mid-level	2.58	2.65	2.50 ± 0.01	2.46 ± 0.03	**2.43** ± 0.01
5	1.69	1.76	1.68 ± 0.06	**1.59** ± 0.04	1.65 ± 0.05
6	0.76	0.77	0.77 ± 0.03	**0.70** ± 0.02	0.72 ± 0.02
7	1.15	1.17	1.14 ± 0.04	1.12 ± 0.03	**1.11** ± 0.03
8	0.79	0.82	0.79 ± 0.03	0.76 ± 0.03	**0.74** ± 0.02
9	0.88	0.91	0.86 ± 0.03	0.87 ± 0.02	**0.83** ± 0.03
10	1.01	1.04	1.01 ± 0.03	**1.00** ± 0.03	**1.00** ± 0.03
11	1.21	1.24	1.21 ± 0.03	**1.19** ± 0.02	1.25 ± 0.04
12	0.90	0.92	**0.88** ± 0.03	**0.88** ± 0.01	0.89 ± 0.02
13	0.98	1.00	0.94 ± 0.02	**0.88** ± 0.03	0.94 ± 0.02
Bottom-level	1.04	1.07	1.03 ± 0.01	**1.00** ± 0.01	1.01 ± 0.00
Average	1.80	1.85	1.75 ± 0.01	1.71 ± 0.02	**1.70** ± 0.01

https://doi.org/10.1371/journal.pone.0242099.t006

Table 7. Prediction performance for the Chubu dataset.

	RMSE				
Node *i*	**MA(16)**	**ES(0.03)**	**NN+BU**	**NN+MinT**	**NN+SR(0.4, 0.6)**
Root	4.11	4.09	3.99 ± 0.04	4.00 ± 0.04	**3.97** ± 0.03
2	1.77	1.75	1.72 ± 0.03	**1.71** ± 0.03	1.72 ± 0.03
3	1.38	1.37	1.37 ± 0.03	**1.35** ± 0.02	1.36 ± 0.03
4	2.19	2.17	2.18 ± 0.03	**2.15** ± 0.03	2.18 ± 0.03
Mid-level	1.78	1.76	1.76 ± 0.01	**1.74** ± 0.02	1.75 ± 0.01
5	0.80	0.79	0.81 ± 0.03	**0.78** ± 0.02	0.81 ± 0.03
6	0.67	0.65	0.65 ± 0.03	**0.64** ± 0.02	0.66 ± 0.03
7	0.76	0.76	**0.74** ± 0.03	**0.74** ± 0.02	**0.74** ± 0.03
8	0.82	0.81	0.77 ± 0.02	0.77 ± 0.03	**0.76** ± 0.02
9	0.71	0.70	0.68 ± 0.02	0.69 ± 0.02	**0.67** ± 0.02
10	0.95	**0.97**	0.99 ± 0.02	**0.97** ± 0.02	0.98 ± 0.02
11	0.81	**0.80**	0.88 ± 0.04	0.86 ± 0.03	0.89 ± 0.04
12	0.99	0.98	0.98 ± 0.03	**0.96** ± 0.02	0.97 ± 0.03
13	**0.73**	**0.73**	0.75 ± 0.02	0.75 ± 0.02	0.77 ± 0.02
Bottom-level	**0.80**	**0.80**	0.81 ± 0.00	**0.80** ± 0.01	0.81 ± 0.00
Average	1.28	1.27	1.27 ± 0.00	**1.26** ± 0.01	1.27 ± 0.00

https://doi.org/10.1371/journal.pone.0242099.t007

our structured regularization method NN+SR than for the bottom-up method NN+BU. For the Tohoku and Chubu datasets, our method greatly accelerated convergence for upper-level time series. For the Kansai dataset, our method was superior to the bottom-up method in terms of both prediction accuracy and convergence speed. These results suggest that our structured regularization method improves the convergence performance of the backpropagation algorithm.

Fig 8 shows heat maps of the out-of-sample relative RMSE values provided by our structured regularization method NN+SR(λ_1, λ_M) for the real-world datasets. For the Tohoku

Table 8. Prediction performance for the Kansai dataset.

	RMSE				
Node *i*	**MA(18)**	**ES(0.05)**	**NN+BU**	**NN+MinT**	**NN+SR(0.4, 1.2)**
Root	13.88	13.84	13.60 ± 0.68	12.47 ± 0.30	**12.20** ± 0.39
2	2.57	2.56	2.58 ± 0.09	2.43 ± 0.05	**2.37** ± 0.04
3	12.78	12.79	12.56 ± 0.69	11.57 ± 0.30	**11.14** ± 0.41
4	1.90	1.90	1.83 ± 0.04	1.81 ± 0.04	**1.68** ± 0.03
Mid-level	5.75	5.75	5.66 ± 0.12	5.27 ± 0.11	**5.06** ± 0.07
5	0.73	0.74	0.77 ± 0.03	**0.72** ± 0.02	0.79 ± 0.02
6	1.80	1.82	1.87 ± 0.07	**1.77** ± 0.04	1.83 ± 0.04
7	1.35	1.36	1.33 ± 0.07	1.35 ± 0.05	**1.22** ± 0.04
8	11.31	11.34	11.29 ± 0.66	10.38 ± 0.27	**10.02** ± 0.39
9	2.71	2.69	2.62 ± 0.14	2.60 ± 0.12	**2.43** ± 0.10
10	1.50	1.49	1.48 ± 0.07	1.45 ± 0.06	**1.43** ± 0.06
11	1.16	1.14	1.14 ± 0.04	1.12 ± 0.04	**1.03** ± 0.03
12	0.82	0.82	0.79 ± 0.02	0.81 ± 0.01	**0.78** ± 0.01
13	0.99	0.99	0.96 ± 0.03	0.96 ± 0.03	**0.95** ± 0.02
Bottom-level	2.49	2.49	2.47 ± 0.04	2.35 ± 0.04	**2.28** ± 0.02
Average	4.12	4.11	4.06 ± 0.08	3.80 ± 0.07	**3.68** ± 0.05

https://doi.org/10.1371/journal.pone.0242099.t008

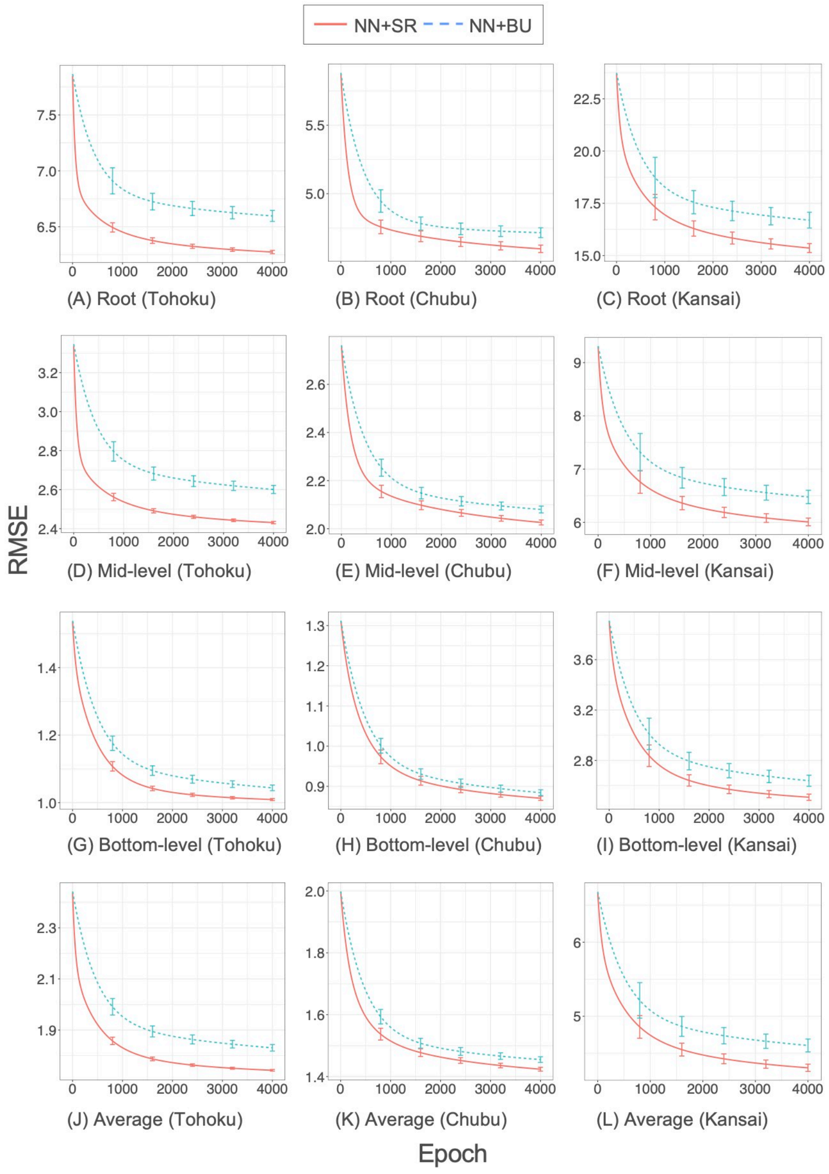

Fig 7. Convergence of the backpropagation algorithm for the real-world datasets.

https://doi.org/10.1371/journal.pone.0242099.g007

dataset, the reduction in RMSE values was particularly large for the root time series. For the Chubu dataset, RMSE values changed greatly from left to right in each heat map, meaning that the regularization for mid-level time series was the most effective. For the Kansai dataset, RMSE values can be reduced greatly for all time series levels if the regularization parameters are properly tuned.

Conclusion

We proposed a structured regularization model for predicting hierarchical time series. Our model uses the regularization term for improving upper-level forecasts to correct bottom-level forecasts. We demonstrated the application of our model to artificial neural networks for time

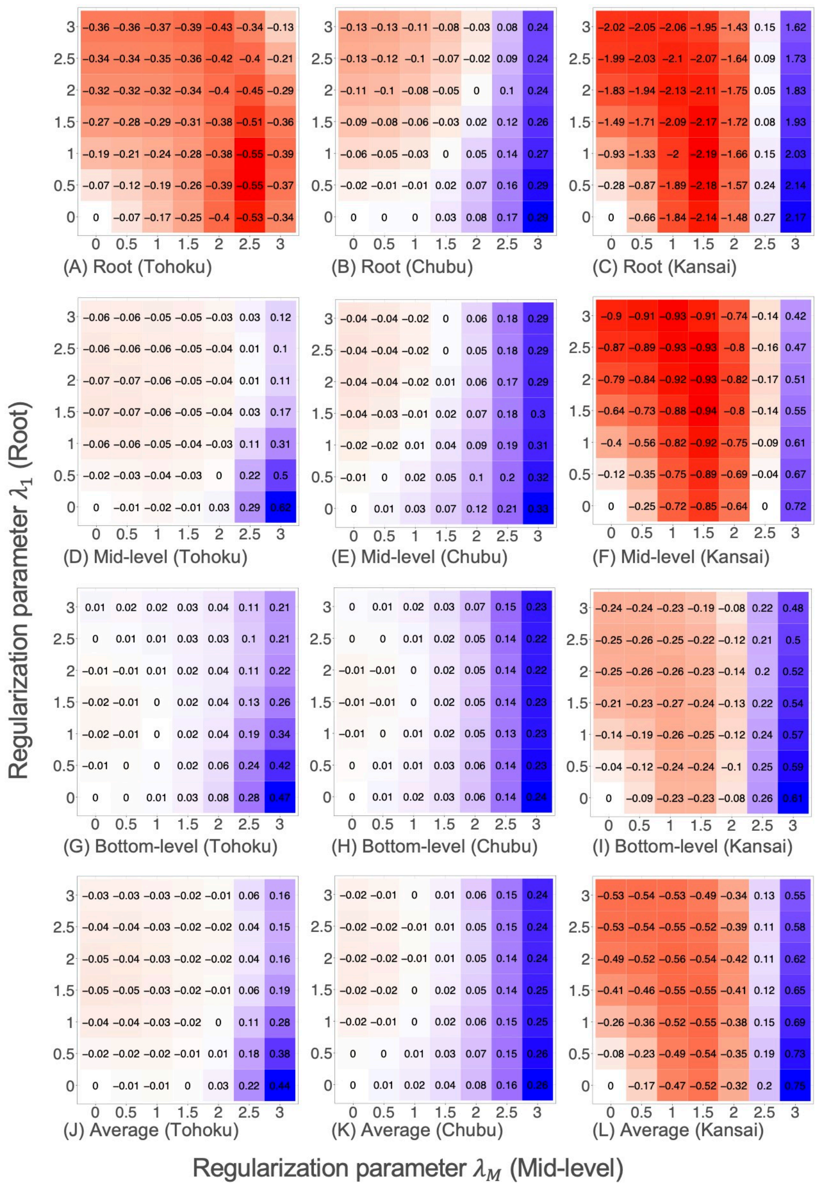

Fig 8. Heat maps of relative RMSEs provided by NN+SR(λ_1, λ_M) in the real-world datasets.

https://doi.org/10.1371/journal.pone.0242099.g008

series prediction. We also developed a backpropagation algorithm specialized for training our model based on artificial neural networks.

We investigated the efficacy of our method through experiments using synthetic and real-world datasets. The experimental results demonstrated that our method, which can adjust regularization parameters to fit data characteristics, compared favorably in prediction performance with other methods that develop coherent forecasts for hierarchical time series. Our regularization term accelerated the backpropagation algorithm. Regularization for mid-level time series was closely related to prediction performance.

One contribution made by this study is the establishment of a new computational framework of artificial neural networks for time series predictions. Moreover, our experiments using synthetic and real-world datasets demonstrated the potential of specialized prediction methods for hierarchical time series. We hope that this study will stimulate further research on exploiting structural properties for better time series predictions.

In future studies, we will extend our structured regularization model to other time series prediction methods, such as the autoregressive integrated moving average model [6, 7] and support vector regression [8]. Another direction of future research will be to develop a high-performance estimation algorithm for our method based on various mathematical optimization techniques [45–50].

Author Contributions

Conceptualization: Tomokaze Shiratori, Ken Kobayashi, Yuichi Takano.

Data curation: Tomokaze Shiratori, Yuichi Takano.

Formal analysis: Tomokaze Shiratori, Yuichi Takano.

Investigation: Tomokaze Shiratori, Yuichi Takano.

Methodology: Tomokaze Shiratori, Ken Kobayashi, Yuichi Takano.

Project administration: Yuichi Takano.

Software: Tomokaze Shiratori.

Validation: Tomokaze Shiratori.

Visualization: Tomokaze Shiratori.

Writing – original draft: Tomokaze Shiratori, Yuichi Takano.

Writing – review & editing: Tomokaze Shiratori, Ken Kobayashi, Yuichi Takano.

References

1. Athanasopoulos G., Ahmed R. A., & Hyndman R. J. (2009). Hierarchical forecasts for Australian domestic tourism. International Journal of Forecasting, 25(1), 146–166. https://doi.org/10.1016/j.ijforecast.2008.07.004
2. Kremer M., Siemsen E., & Thomas D. J. (2016). The sum and its parts: Judgmental hierarchical forecasting. Management Science, 62(9), 2745–2764. https://doi.org/10.1287/mnsc.2015.2259
3. Fliedner G. (1999). An investigation of aggregate variable time series forecast strategies with specific subaggregate time series statistical correlation. Computers & Operations Research, 26(10–11), 1133–1149. https://doi.org/10.1016/S0305-0548(99)00017-9
4. Ben Taieb S., Taylor J. W., & Hyndman R. J. (2017, August). Coherent probabilistic forecasts for hierarchical time series. In Proceedings of the 34th International Conference on Machine Learning-Volume 70 (pp. 3348–3357). JMLR. org.
5. Wickramasuriya S. L., Athanasopoulos G., & Hyndman R. J. (2019). Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. Journal of the American Statistical Association, 114(526), 804–819. https://doi.org/10.1080/01621459.2018.1448825
6. Brockwell P. J., Davis R. A., & Fienberg S. E. (1991). Time series: Theory and methods. Springer Science & Business Media.
7. Hyndman R. J., & Athanasopoulos G. (2018). Forecasting: principles and practice. OTexts.
8. Karmy J. P., & Maldonado S. (2019). Hierarchical time series forecasting via support vector regression in the European travel retail industry. Expert Systems with Applications, 137, 59–73. https://doi.org/10.1016/j.eswa.2019.06.060
9. Lütkepohl H. (2011). Forecasting aggregated time series variables. OECD Journal: Journal of Business Cycle Measurement and Analysis, 2010(2), 1–26.
10. Widiarta H., Viswanathan S., & Piplani R. (2009). Forecasting aggregate demand: An analytical evaluation of top-down versus bottom-up forecasting in a production planning framework. International Journal of Production Economics, 118(1), 87–94. https://doi.org/10.1016/j.ijpe.2008.08.013

11. Park M., & Nassar M. (2014). Variational Bayesian inference for forecasting hierarchical time series. In International conference on machine learning (ICML), workshop on divergence methods for probabilistic inference, Beijing.

12. Hyndman R. J., Ahmed R. A., Athanasopoulos G., & Shang H. L. (2011). Optimal combination forecasts for hierarchical time series. Computational Statistics & Data Analysis, 55(9), 2579–2589. https://doi.org/10.1016/j.csda.2011.03.006

13. Capistrán C., Constandse C., & Ramos-Francia M. (2010). Multi-horizon inflation forecasts using disaggregated data. Economic Modelling, 27(3), 666–677. https://doi.org/10.1016/j.econmod.2010.01.006

14. Hyndman R. J., Lee A. J., & Wang E. (2016). Fast computation of reconciled forecasts for hierarchical and grouped time series. Computational Statistics & Data Analysis, 97, 16–32. https://doi.org/10.1016/j.csda.2015.11.007

15. van Erven T., & Cugliari J. (2015). Game-theoretically optimal reconciliation of contemporaneous hierarchical time series forecasts. In Modeling and stochastic learning for forecasting in high dimensions (pp. 297–317). Springer, Cham.

16. Ben Taieb S., & Koo B. (2019, July). Regularized Regression for Hierarchical Forecasting Without Unbiasedness Conditions. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 1337–1347).

17. Ben Taieb, S., Yu, J., Barreto, M. N., & Rajagopal, R. (2017, February). Regularization in hierarchical time series forecasting with application to electricity smart meter data. In Thirty-First AAAI Conference on Artificial Intelligence.

18. Hastie T., Tibshirani R., & Wainwright M. (2015). Statistical learning with sparsity: the lasso and generalizations. CRC press.

19. Jenatton R., Audibert J. Y., & Bach F. (2011). Structured variable selection with sparsity-inducing norms. Journal of Machine Learning Research, 12(Oct), 2777–2824.

20. Zhao P., Rocha G., & Yu B. (2009). The composite absolute penalties family for grouped and hierarchical variable selection. The Annals of Statistics, 37(6A), 3468–3497. https://doi.org/10.1214/07-AOS584

21. Nicholson W. B., Matteson D. S., & Bien J. (2017). VARX-L: Structured regularization for large vector autoregressions with exogenous variables. International Journal of Forecasting, 33(3), 627–651. https://doi.org/10.1016/j.ijforecast.2017.01.003

22. Schimbinschi F., Moreira-Matias L., Nguyen V. X., & Bailey J. (2017). Topology-regularized universal vector autoregression for traffic forecasting in large urban areas. Expert Systems with Applications, 82, 301–316. https://doi.org/10.1016/j.eswa.2017.04.015

23. Bien J., Taylor J., & Tibshirani R. (2013). A lasso for hierarchical interactions. Annals of statistics, 41 (3), 1111. https://doi.org/10.1214/13-AOS1096 PMID: 26257447

24. Kim S., & Xing E. P. (2012). Tree-guided group lasso for multi-response regression with structured sparsity, with an application to eQTL mapping. The Annals of Applied Statistics, 6(3), 1095–1117. https://doi.org/10.1214/12-AOAS549

25. Lim M., & Hastie T. (2015). Learning interactions via hierarchical group-lasso regularization. Journal of Computational and Graphical Statistics, 24(3), 627–654. https://doi.org/10.1080/10618600.2014.938812 PMID: 26759522

26. Sato T., Takano Y., & Nakahara T. (2019). Investigating consumers' store-choice behavior via hierarchical variable selection. Advances in Data Analysis and Classification, 13(3), 621–639. https://doi.org/10.1007/s11634-018-0327-0

27. Wen W., Wu C., Wang Y., Chen Y., & Li H. (2016). Learning structured sparsity in deep neural networks. In Advances in neural information processing systems (pp. 2074–2082).

28. Caruana R. (1997). Multitask learning. Machine learning, 28(1), 41–75. https://doi.org/10.1023/A:1007379606734

29. Zhang Y., & Yang Q. (2017). A survey on multi-task learning. arXiv preprint arXiv:1707.08114.

30. Evgeniou, T., & Pontil, M. (2004, August). Regularized multi-task learning. In Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 109–117).

31. Jacob L., Vert J. P., & Bach F. R. (2009). Clustered multi-task learning: A convex formulation. In Advances in neural information processing systems (pp. 745–752).

32. Ruder S. (2017). An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098.

33. Gao J., Murphey Y. L., & Zhu H. (2018). Multivariate time series prediction of lane changing behavior using deep neural network. Applied Intelligence, 48(10), 3523–3537. https://doi.org/10.1007/s10489-018-1163-9

34. Hsieh W. W. (2004). Nonlinear multivariate and time series analysis by neural network methods. Reviews of Geophysics, 42(1). https://doi.org/10.1029/2002RG000112

35. Khashei M., & Bijari M. (2010). An artificial neural network (*p*, *d*, *q*) model for timeseries forecasting. Expert Systems with Applications, 37(1), 479–489. https://doi.org/10.1016/j.eswa.2009.05.044

36. Lai, G., Chang, W. C., Yang, Y., & Liu, H. (2018, June). Modeling long-and short-term temporal patterns with deep neural networks. In The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval (pp. 95–104).

37. Zhang G. P. (2003). Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing, 50, 159–175. https://doi.org/10.1016/S0925-2312(01)00702-0

38. Zhang G. P., & Qi M. (2005). Neural network forecasting for seasonal and trend time series. European Journal of Operational Research, 160(2), 501–514. https://doi.org/10.1016/j.ejor.2003.08.037

39. Panagiotelis A., Athanasopoulos G., Gamakumara P., & Hyndman R. J. (2020). Forecast reconciliation: A geometric view with new insights on bias correction. International Journal of Forecasting. https://doi.org/10.1016/j.ijforecast.2020.06.004

40. Bishop C. M. (2006). Pattern recognition and machine learning. Springer.

41. Goodfellow I., Bengio Y., & Courville A. (2016). Deep learning. MIT press.

42. Bergmeir C., Hyndman R. J., & Koo B. (2018). A note on the validity of cross-validation for evaluating autoregressive time series prediction. Computational Statistics & Data Analysis, 120, 70–83. https://doi.org/10.1016/j.csda.2017.11.003

43. Karsoliya S. (2012). Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture. International Journal of Engineering Trends and Technology, 3(6), 714–717.

44. Lippmann R. (1987). An introduction to computing with neural nets. IEEE ASSP Magazine, 4(2), 4–22. https://doi.org/10.1109/MASSP.1987.1165576

45. Bertsimas D., King A., & Mazumder R. (2016). Best subset selection via a modern optimization lens. The annals of statistics, 813–852. https://doi.org/10.1214/15-AOS1388

46. Bertsimas D., Pauphilet J., & Van Parys B. (2019). Sparse regression: Scalable algorithms and empirical performance. arXiv preprint arXiv:1902.06547.

47. Kudo K., Takano Y., & Nomura R. (2020). Stochastic discrete first-order algorithm for feature subset selection. IEICE Transactions on Information and Systems, 103(7), 1693–1702. https://doi.org/10.1587/transinf.2019EDP7274

48. Takano Y., & Miyashiro R. (2020). Best subset selection via cross-validation criterion. TOP, 28(2), 475–488. https://doi.org/10.1007/s11750-020-00538-1

49. Tamura R., Kobayashi K., Takano Y., Miyashiro R., Nakata K., & Matsui T. (2017). Best subset selection for eliminating multicollinearity. Journal of the Operations Research Society of Japan, 60(3), 321–336. https://doi.org/10.15807/jorsj.60.321

50. Tamura R., Kobayashi K., Takano Y., Miyashiro R., Nakata K., & Matsui T. (2019). Mixed integer quadratic optimization formulations for eliminating multicollinearity based on variance inflation factor. Journal of Global Optimization, 73(2), 431–446. https://doi.org/10.1007/s10898-018-0713-3