Spatio-temporal Multi-task Learning via Tensor Decomposition

Jianpeng Xu, Jiayu Zhou, Pang-Ning Tan, *Member, IEEE*, Xi Liu, and Lifeng Luo

Abstract—Predictive modeling of large-scale spatio-temporal data is an important but challenging problem as it requires training models that can simultaneously predict the target variables of interest at multiple locations while preserving the spatial and temporal dependencies of the data. In this paper, we investigate the effectiveness of applying a multi-task learning approach based on supervised tensor decomposition to the spatio-temporal prediction problem. Our proposed framework, known as SMART, encodes the data as a third-order tensor and extracts a set of interpretable, spatial and temporal latent factors from the data. An ensemble of spatial and temporal prediction models are trained using the latent factors as their predictor variables. Outputs from the ensemble model are aggregated to make predictions on test instances. The framework also allows known patterns from the domain to be incorporated as constraints to guide the tensor decomposition and ensemble learning processes. As the data may grow over space and time, an incremental learning version of the framework is given to efficiently update the models. We perform extensive experiments using a global-scale climate dataset to evaluate the accuracy and efficiency of the models as well as interpretability of the latent factors.

Index Terms—multi-task learning; tensor decomposition; spatio-temporal data mining; incremental learning

1 Introduction

SPATIAL-temporal data, consisting of measurements for one or more raster fields, such as weather, traffic volume, crime rate, or disease incidents, have become increasingly prevalent in the current age of big data. The abundance of such data provides opportunities to develop sophisticated predictive modeling techniques for various scientific domains, including climatology [1], [21], [27], medicine [13], and crop sciences [5]. In addition to generating robust predictions at multiple locations, the models should also provide useful insights into the underlying factors governing the spatio-temporal variabilities observed in the data.

Conventional approaches for spatio-temporal prediction can be divided into two categories. In the first category, a temporal model is trained for each location with the spatial information acting as constraints on the relationship between models at different locations [28], [34], [39]. The spatial information are typically provided based on some simplified assumption about the relationships between models at different locations. For example, a typical assumption based on Tobler's first law of geography [31] is that the models for nearby locations should have similar parameter values. This assumption can be realized using a graph Laplacian regularizer such as the ones adopted in [34], [41]. Although such an assumption helps to ensure spatial smoothness of the models, it is insufficient to capture the detailed spatial patterns of the data. In contrast, the second category of approaches would develop a model for all locations, using spatial modeling methods such as Gaussian Markov Random Field [24] and kriging [12]. Although these approaches can reproduce the spatial patterns effectively, they may not be able to capture the detailed temporal patterns at each location.

To the best of our knowledge, none of these existing approaches combine both spatial and temporal models to generate more accurate predictions with consistent patterns across both space and time dimensions. Furthermore, they are not designed to incorporate known spatio-temporal patterns from the application domain. For example, it is known that the climate variability at a location can be influenced by broad-scale teleconnection patterns [21], [27] such as the El Niño phenomenon. How to seamlessly integrate such patterns into the predictive modeling framework while simultaneously identifying new, previously unknown patterns is a challenge that has not been sufficiently addressed in the literature.

To address these challenges, this paper presents a multi-task learning framework for predictive modeling of spatio-temporal data using a supervised tensor decomposition approach. The framework, called SMART (Spatio-temporal Multi-task Learning via Tensor Decomposition), represents the data as a third-order tensor, where the dimensions of the tensor correspond to the spatial, temporal, and multivariate features of the data. A CANDECOMP/PARAFAC (CP) decomposition [19] is performed on the tensor to extract a set of rank-1 latent factors from the data. As shown in Fig. 1, the multivariate spatio-temporal data can be decomposed into multiple rank-1 tensors, where each of the rank-1 tensor is calculated by the outer product of three rank-1 latent factors. Each latent factor can be described by its spatial, temporal, and feature dimensions, thus providing an interpretable way to characterize the underlying factors governing the variability of the data. The latent factors also serve as predictor variables for training an ensemble of spatial and temporal multi-task learning (MTL) models. SMART enables the tensor decomposition and ensemble

- *Jianpeng Xu, Jiayu Zhou, Pang-Ning Tan and Xi Liu are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, 48823.*
  E-mail: {xujianpe, jiayuz, ptan, liuxi4}@msu.edu
- *Lifeng Luo is with the Department of Geography, Michigan State University, East Lansing, MI, 48823.*
  E-mail: lluo@msu.edu

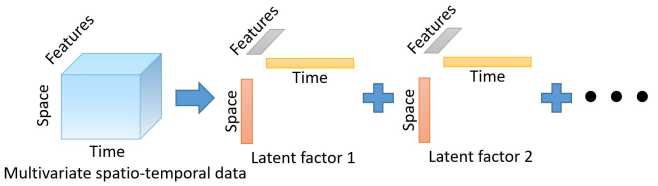


Fig. 1: CP decomposition on a multivariate spatio-temporal data tensor representation. Each latent factor can be described by its spatial, temporal, and feature dimensions.

learning steps to be performed simultaneously by optimizing a joint objective function. The ensemble model can be applied to the test data by aggregating the outputs from the spatial and temporal models to generate their final prediction. Furthermore, known patterns from the domain can be easily incorporated as constraints on the spatial or temporal latent factors derived by the proposed framework. Sparsity-inducing norms are also introduced as additional constraints to avoid overfitting and enhance the model interpretability.

Despite its advantages, one practical limitation of the proposed framework is its high storage and runtime complexity when applied to big data applications. As many spatio-temporal datasets may continue to grow over space and time, building the models repeatedly from scratch whenever new observation data becomes available is not a feasible solution. In our previous work [36], an incremental learning algorithm named WISDOM was introduced to provide an efficient implementation of the proposed SMART framework. WISDOM allows the latent factors and ensemble model parameters to be incrementally updated when data for a new location or a new time period is introduced. Using a global-scale climate data as our case study, we showed that the proposed framework can help identify the spatial and temporal latent factors that drive the variability observed in the climate data. In this journal version, we compare the performance of WISDOM against a batch implementation of SMART and demonstrate their trade-off in terms of model accuracy and training runtime, especially when applied to large-scale datasets. We also provide detailed time complexity and scalability analysis for both WISDOM and its corresponding batch algorithm.

The remainder of the paper is organized as follows. Section 2 briefly summarizes previous work related to this paper. The spatio-temporal predictive modeling problem is described in Section 3. The proposed SMART framework along with its batch and incremental learning implementations are introduced in Section 4 and 5. Section 6 describes the results of our experimental evaluation while Section 7 presents the conclusions of this study.

2 Related Work

Spatio-temporal datasets are prevalent across numerous application domains, from geophysical and environmental sciences to urban computing and healthcare. The spatio-temporal data can be generally classified into two types [25], either as spatial trajectories of moving objects or as time series observations at fixed geo-referenced locations. The former corresponds to the recorded paths of various mobile entities [42], such as the routes taken by vehicles [37]. The latter, also known as spatial time series data [25], consists of temporal measurements for one or more raster fields, such as climate, pollution level, or disease incidents, at a fixed set of geo-referenced locations [18]. The techniques presented in this paper focused only on the latter type of spatio-temporal data. Furthermore, we use climate modeling as the testbed application for evaluating the various techniques.

There has been extensive research which applies data mining and machine learning techniques to climate data analysis problem [4]. Novel techniques such as distribution preserving regression [20] have been developed to meet the specific needs of the domain. Since the climate modeling task requires making predictions at multiple locations within a region of interest, this makes it a natural choice for applying multi-task learning (MTL) [8]. MTL assumes that the generalization performance for multiple prediction tasks can be enhanced by learning the related tasks jointly [43]. It is essential in MTL methods to define the task relationships. Existing MTL methods either define the task relatedness explicitly using low-rank assumption [10], graph Laplacian [44], parameter sharing [15], [38], combination of the above methods [35], or learn the relatedness using structure learning techniques [16]. Unlike the previous works, in which the task relatedness are mainly defined by various assumptions between *task models*, such as low-rank assumption, the approach developed in this paper learns the task relatedness from the *spatio-temporal data itself* by performing a tensor decomposition [3], [19] on the data.

Existing tensor decomposition approaches can be classified as unsupervised [9], [11] or supervised methods [23], [32], [33], [39]. Unsupervised methods are designed to minimize reconstruction error as the tensor is decomposed into a product of its latent factors. For example, Ardavan et. al [2] proposed an Orthogonal Tensor Factorization Framework for spatio-temporal data analysis to enhance the interpretability of the model. Supervised methods consider the relationship between the predictor and response variables, and thus, are more appropriate for predictive modeling problems. Wu et. al [33] proposed a framework that couples non-negative tensor factorization with a maximum margin classifier for classification problems. Romera-Paredes et. al [23] and Milawarne et. al [32] presented a multilinear MTL framework using a supervised tensor decomposition approach. Similarly, Yu et. al [39] proposed a low-rank tensor learning approach for multivariate spatio-temporal data. These approaches encode the model parameters as a tensor, which is assumed to have a low rank representation. Thus, the tensor decomposition was performed on the model parameters instead of the spatio-temporal data. In addition, Yu et. al [40] provided a theoretical analysis on the tensor decomposition based regression by comparing with Gaussian Processes. In contrast, our SMART framework applies tensor decomposition to the spatio-temporal data itself, allowing us to derive meaningful interpretation of the latent factors in terms of their spatial, temporal, and feature dimensions. Furthermore, none of these existing methods consider building an ensemble model on the latent factors obtained from the different dimensions of the tensor, unlike the method proposed in this paper.

Finally, online tensor decomposition methods have also

been developed in recent years [29], [30]. These methods are mostly unsupervised, based on incremental versions of Tucker decomposition [30] [17] or CP decomposition [45] [26] [14]. Although there has been recent work on online supervised tensor decomposition for streaming data [39], it considers the new observations along the time dimension only, unlike our framework, which assumes the new observations may arrive along the space or time dimensions. Furthermore, since the tensor decomposition was applied to the model parameters, instead of the spatio-temporal data itself, this makes it harder to interpret the latent factors. Bulat et. al proposed an incremental multi-domain learning with tensor factorization [7], which performs an incremental tensor decomposition on new domains. However, it does not consider the temporal relatedness within tasks and does not automatically apply on spatio-temporal data.

3 Problem Statement

Let $\mathcal{S} = \{1, 2, \cdots, S\}$ denote the set of indices associated with the geo-locations of a given spatio-temporal dataset and $\mathcal{T} = \{1, 2, \cdots, T\}$ denote the set of indices associated with the timestamps of the data. For brevity, we represent matrices as boldface capital letters, such as $\mathbf{A}$ or $\mathbf{B}$, and tensors by boldface calligraphic fonts, such as $\boldsymbol{\mathcal{X}}$ or $\boldsymbol{\mathcal{Y}}$. Scalars are denoted by lowercase letters such as c whereas vectors are denoted by boldface lowercase letters such as $\mathbf{x}$. Furthermore, we use the ":" symbol to denote sub-arrays within a matrix or a tensor. For example, $\mathbf{A}_{:k}$ refers to the k-th column of matrix $\mathbf{A}$. To simplify the notation, $\mathbf{A}_{:k}$ is also written as $\mathbf{A}_k$ when the context is clear.

The **fiber** of a tensor is a vector obtained by fixing all indices of the tensor except for one of them. For example, given a 3-dimensional tensor $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{X}}_{i:j}$ refers to its mode-2 fiber, obtained by fixing the mode-1 index to i and mode-3 index to j. In the meantime, the **slice** of a tensor refers to a matrix obtained by fixing all but two of the indices of the tensor. For example, $\boldsymbol{\mathcal{X}}_{::i}$ is the i-th mode-3 slice of the tensor $\boldsymbol{\mathcal{X}}$ obtained by setting its mode-3 index to i. A tensor $\boldsymbol{\mathcal{X}} \in \Re^{p_1 \times \cdots \times p_N}$ can be unfolded into a matrix $\mathbf{X}_{(n)} \in \Re^{p_n \times q_n}$, where $q_n = \Pi_{k \neq n} p_k$, by arranging each mode-n fiber of the tensor as a column of $\mathbf{X}_{(n)}$. This process is also known as mode-n unfolding or mode-n matricization of the tensor. The **Khatri-Rao product** of two matrices is equivalent to applying a Kronecker product columnwise to the matrices. For example, the Khatri-Rao product for matrices $\mathbf{A} \in \Re^{N \times K}$ and $\mathbf{B} \in \Re^{M \times K}$ is given by:

$$\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2, \cdots, \mathbf{a}_K \otimes \mathbf{b}_K],$$

where $\otimes$ denote the Kronecker product.

The spatio-temporal prediction problem can be formalized as follows. We consider a spatio-temporal dataset, $\boldsymbol{\mathcal{D}} = (\boldsymbol{\mathcal{X}}, \mathbf{Y})$, where $\boldsymbol{\mathcal{X}} \in \Re^{S^* \times T^* \times d}$ represents the spatio-temporal tensor of predictor variables, with d the number of predictor variables, and $\mathbf{Y} \in \Re^{S \times T}$ represents a matrix containing the time series of response variable values for all locations. Note that S^* and S respectively denote the number of locations for which the values of predictor and response variables are available. Similarly, T^* denotes the length of time series of the predictors, while T is the length of time series of the responses. In the setting of this paper, we assume S^* and T^* are greater than or equal to S and T, respectively. Our goal is to learn a spatio-temporal target function, $f_{st}(\mathbf{x}_{st})$, that maps each input $\mathbf{x}_{st} \in \Re^d$ to its corresponding response value $y_{st} \in \Re$ in a way that minimizes some loss function, $\ell(f_{st}[\mathbf{x}_{st}, y_{st}])$.

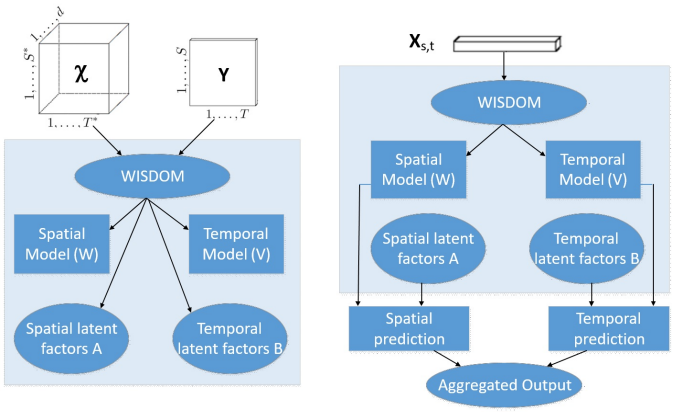
A standard approach to address the spatio-temporal prediction problem at multiple locations is to train a distinct temporal model for each location, $f_t(\mathbf{x}_{st}; \mathbf{w}_s)$, where $\mathbf{w}_s \in \Re^{d \times 1}$ is the model parameter for location s. The temporal models can be trained independently or jointly using a multi-task learning approach such as [34] to predict future values of the response variable at all locations, where each task corresponds to the modeling task at each location. Here, the spatial information can be used as constraints [28], [34] on the parameters of the temporal models to ensure their spatial smoothness. Alternatively, one could also treat the modeling problem at each time as one task in multi-task learning framework, and train a spatial prediction model at each time t, $f_s(\mathbf{x}_{st}; \mathbf{v}_t)$, where $\mathbf{v}_t \in \Re^{d \times 1}$ is the model parameter at time t and apply the model to predict the values of the response variable at a previously unobserved location s. Ensuring the temporal consistency of the predictions is a challenge that must be addressed by the spatial models. To overcome the challenge of preserving the spatial and temporal consistencies of the model predictions, this paper presents an ensemble multi-task learning approach that combines the predictions from the spatial and temporal multi-task models in a unified learning framework, where the multiple tasks are not only across over space, but also across time. Details of the proposed framework are given in the next section.

4 Proposed Ensemble Spatio-Temporal Multi-task Learning Framework

Fig. 2 presents a high-level overview of the training and prediction steps of the proposed SMART framework. The framework is novel in that it simultaneously learns an ensemble of spatial and temporal multi-task models using the latent factors derived from the spatio-temporal data. The ensemble outputs are then combined to obtain the final prediction. More specifically, the framework predicts the response value for a location s at time t as a weighted linear combination of its spatial and temporal models, i.e.,

$$\begin{aligned} \hat{y}_{st} = f(\mathbf{x}_{st}) &= \mathbf{x}_{st}^T \Big[\sum_k^K \mathbf{A}_{sk} \mathbf{w}_k + \sum_k^K \mathbf{B}_{tk} \mathbf{v}_k \Big] \\ &= \mathbf{x}_{st}^T (\mathbf{W}^T \mathbf{A}_s + \mathbf{V}^T \mathbf{B}_t) \\ &= f_s(\mathbf{x}_{st}^T) + f_t(\mathbf{x}_{st}^T), \end{aligned} \quad (1)$$

where $\mathbf{W} = [\mathbf{w}_1^T; \mathbf{w}_2^T; ...; \mathbf{w}_K^T] \in \Re^{K \times d}$, $\mathbf{V} = [\mathbf{v}_1^T; \mathbf{v}_2^T; ...; \mathbf{v}_K^T] \in \Re^{K \times d}$, $\mathbf{A}_s$ is the transpose of s-th row of $\mathbf{A} \in \Re^{S \times K}$, and $\mathbf{B}_t$ is the transpose of t-th row of $\mathbf{B} \in \Re^{T \times K}$. Note that $f_s(\mathbf{x}_{st}^T) = \mathbf{x}_{st}^T \mathbf{W}^T \mathbf{A}_s$ corresponds to an ensemble of spatial models, where $\mathbf{A}_s$ denotes the model parameters, expressed in terms of a linear combination of the K spatial latent factors $\mathbf{W}$. Similarly, $f_t(\mathbf{x}_{st}^T) = \mathbf{x}_{st}^T \mathbf{V}^T \mathbf{B}_t$ corresponds to an ensemble of temporal models parameterized by the weight matrix $\mathbf{B}_t$, which represents a linear combination of the temporal latent factors $\mathbf{V}$. The



(a) Training (b) Prediction

Fig. 2: Overview of the proposed SMART framework.

parameters are estimated by optimizing the following joint objective function:

$$\min_{\mathbf{W},\mathbf{V}} \sum_{s}^{S}\sum_{t}^{T} \ell[\hat{y}_{st}, y_{st}] + \Omega_m(\mathbf{W},\mathbf{V}) \tag{2}$$

where $\ell[\cdot]$ is the loss function and $\Omega(\mathbf{W},\mathbf{V})$ is a regularization term for the model parameters. In this paper, we consider a squared loss function, $\ell[\hat{y}_{st}, y_{st}] = (\hat{y}_{st} - y_{st})^2$, and define $\Omega_m(\mathbf{W},\mathbf{V}) = \|\mathbf{W}\|_1 + \|\mathbf{V}\|_1$ to ensure sparsity of the models.

4.1 Supervised Tensor Decomposition

The formulation presented in Eq.(1) requires knowledge about the latent factors of the spatio-temporal tensor. These latent factors are represented by the factor matrices $\mathbf{A}$ and $\mathbf{B}$, which can be derived using tensor decomposition techniques. There are two standard ways to decompose a tensor, namely, Tucker and CANDECOMP/PARAFAC (CP) decompositions [19]. Tucker decomposition factorizes a tensor into a core tensor and a product of its factor matrices for each mode. Though it provides a more general representation, it is harder to interpret the latent factors as the number of latent factors for each mode can be different. In contrast, CP decomposition factorizes a tensor into a sum of rank-1 tensors, i.e., $\boldsymbol{\mathcal{X}} = [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!] = \sum_{k=1}^{K} \mathbf{A}_k \circ \mathbf{B}_k \circ \mathbf{C}_k$, where $\circ$ denote the outer product operation between two vectors while $\mathbf{A}_k$, $\mathbf{B}_k$ and $\mathbf{C}_k$ correspond to the vectors associated with the k-th latent factor. The vectors $\mathbf{A}_k$, $\mathbf{B}_k$ and $\mathbf{C}_k$ also denote the k-th columns of the matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, respectively.

In this work, we apply CP decomposition to the spatio-temporal tensor $\boldsymbol{\mathcal{X}} \in \Re^{S^* \times T^* \times d}$, where the (s,t)-th element of its mode-3 fiber corresponds to the values of the predictor values for location s at time t, i.e., $\boldsymbol{\mathcal{X}}_{st:} = \mathbf{x}_{st}$. The latent factors are obtained by optimizing the following objective function:

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \frac{1}{2}\|\boldsymbol{\mathcal{X}} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 + \Omega_d(\mathbf{A},\mathbf{B},\mathbf{C})$$

where $\|\boldsymbol{\mathcal{X}}\|_F = \sqrt{\sum_{ijk} \mathcal{X}_{ijk}^2}$ is the Frobenius norm of the tensor $\boldsymbol{\mathcal{X}}$ and $\Omega_d(\mathbf{A},\mathbf{B},\mathbf{C})$ is a regularization term for the factor matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. To ensure interpretability of the latent factors, the following regularization penalty is used:

$$\Omega_d(\mathbf{A},\mathbf{B},\mathbf{C}) = \|\mathbf{A}\|_1 + \|\mathbf{B}\|_1 + \|\mathbf{C}\|_1$$

Putting everything together, the objective function for the proposed SMART framework can be stated as follows:

$$\begin{aligned} &\min_{\boldsymbol{\Gamma}=\{\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C}\}} \mathcal{F}(\boldsymbol{\Gamma};\boldsymbol{\mathcal{X}},\mathbf{Y}) \\ = \;& \frac{1}{2}\sum_{s}^{S}\sum_{t}^{T}(\mathbf{x}_{s,t}^T(\mathbf{W}^T\mathbf{A}_s + \mathbf{V}^T\mathbf{B}_t) - y_{s,t})^2 \\ + \;& \frac{\lambda}{2}\|\boldsymbol{\mathcal{X}} - [\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_F^2 + \beta\|\boldsymbol{\Gamma}\|_1 \end{aligned} \tag{3}$$

where $\boldsymbol{\Gamma} = \{\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C}\}$ and $\|\boldsymbol{\Gamma}\|_1$ denotes applying an ℓ_1 norm on each of the $\mathbf{W}$, $\mathbf{V}$, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ matrices, respectively. The model parameters $\mathbf{W}$, $\mathbf{V}$ and the latent factors $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ can be learned by iteratively optimizing the objective function with respect to each set of variables to be solved. Note that the multiple tasks in this framework are ensembles of spatial multi-task learning over locations and temporal multi-task learning over timestamps.

4.2 Batch Learning

In this subsection, we present a learning approach based on an alternating minimization strategy for optimizing Eq. (3). Since not all terms in the objective function are differentiable, we employ the proximal gradient descent method [22] to solve each subproblem. Consider a non-differentiable objective function $f(x)$ that can be decomposed into a smooth, differentiable function $g(x)$ and a non-smooth function $h(x)$, i.e., $f(x) = g(x) + h(x)$. For example, the terms involving Frobenius norms in our objective function are differentiable whereas those involving the sparsity-inducing $L1$-norms are non-differentiable. The proximal gradient descent method would iteratively update the model parameters as follows:

$$x^{(k)} = \mathbf{prox}_{t_k,h}\left(x^{(k-1)} - t_k \nabla g(x^{(k-1)})\right)$$

where $x^{(k)}$ is the parameter to be estimated at step k. $\mathbf{prox}_{t_k,h}$ is the proximal operator for the nondifferentiable function h, $\nabla g(x^{(k-1)})$ is the gradient on the smooth function g w.r.t. $x^{(k-1)}$ and t_k is the step size for the gradient descent update. The proximal operator for ℓ_1 norm function is the soft-thresholding operator [22]:

$$\mathbf{prox}_{\lambda,h}(v) = (v-\lambda)_+ - (-v-\lambda)_+$$

where λ is the threshold parameter. The parameters are updated iteratively by calculating the gradient on the smooth part of the objective function, and then apply the soft-thresholding operator (proximal mapping function for ℓ_1 norm) to determine its next value. The step size can be found using a line search algorithm. The gradients of the objective function with respect to each parameter $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{W}$, and $\mathbf{V}$ during the alternating minimization step are omitted due to lack of space.

Note that the batch learning approach described here requires fitting the model on the entire training set available at training time. As a result, such a model would have to be retrained from scratch each time new observation data

become available, unlike the incremental learning approach to be described in Section 5.

4.3 Complexity Analysis

This subsection provides the detailed time and space complexity analysis for the batch implementation of the SMART framework. We assume T and T^* (S and S^*) are in the same magnitude, and will use T and S to simplify the notations for T, T^* and S, S^* through the analysis. Since the runtime of the algorithm depends on the number of iterations, we provide the time complexity for calculating the gradient of each variable in each iteration. For example, the calculation for gradient with respect to $\mathbf{A}_s$ takes $O(dTk)$ time, where d is the number of features, S is the number of locations, T is the length of the time series, and k is the number of latent factors. Since the update has to be performed for every location, its runtime complexity per iteration is $O(dSTk)$. Similarly, it can be shown that the update formula for other matrices ($\mathbf{B}$, $\mathbf{C}$, $\mathbf{W}$, and $\mathbf{V}$) also take $O(dSTk)$ computations per iteration.

In the batch implementation, SMART requires the entire spatio-temporal dataset as well as the parameter set $\mathbf{\Gamma} = \{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{W}, \mathbf{V}\}$ to be available in order to perform the optimization. Thus, the overall space complexity for SMART is $O(dST + Sk + Tk + dk)$. Since $k \ll T, S$, the space complexity is $O(dST)$. Since S and T may continue to grow, both the storage and runtime complexities can be very high, which makes it infeasible to apply the batch learning algorithm. In next section, we present WISDOM, which is an incremental learning algorithm to reduce the computational complexity of SMART so that it can scale up to large-scale spatio-temporal prediction problems.

5 WISDOM: Weighted Incremental Spatio-Temporal Multi-task Learning Algorithm

As the size of many spatio-temporal datasets can be very large, efficient algorithms are needed to learn the parameters of the SMART framework described in the previous section. In an incremental learning setting, the spatio-temporal data $\boldsymbol{\mathcal{D}} = (\boldsymbol{\mathcal{X}}, \mathbf{Y})$ is assumed to be periodically augmented with a new data chunk, $(\boldsymbol{\mathcal{X}}_{\text{new}}, \mathbf{Y}_{\text{new}})$, where $\boldsymbol{\mathcal{X}}_{\text{new}} \in \Re^{S\times 1\times d}$ and $\mathbf{Y}_{\text{new}} \in \Re^{S\times 1}$ if the data is for a new time period, or $\boldsymbol{\mathcal{X}}_{\text{new}} \in \Re^{1\times T\times d}$ and $\mathbf{Y}_{\text{new}} \in \Re^{1\times T}$ if the data is from a new location. We termed the former as incremental learning over time and the latter as incremental learning over space. Two implementations of the WISDOM algorithm have been developed—one for incremental learning over space and the other for incremental learning over time. A hybrid approach that combines both learning strategies can be easily developed to handle new observation data along the space and/or time dimensions.

For incremental learning, our goal is to adapt the existing models without rebuilding the model from scratch each time new observations become available. To ensure that the model parameters and latent factors do not vary significantly from their previous values, a smoothness criterion can be added to the objective function. The optimization problem for incremental learning is formulated as follows:

$$\min_{\mathbf{\Gamma}} \mathcal{Q}(\mathbf{\Gamma}, \tilde{\mathbf{\Gamma}}) = \mathcal{F}(\mathbf{\Gamma}; \boldsymbol{\mathcal{X}}_{\text{new}}, \mathbf{Y}_{\text{new}}) + \mathcal{G}(\mathbf{\Gamma}, \tilde{\mathbf{\Gamma}}),$$

where $\tilde{\mathbf{\Gamma}}$ are the previous parameter values before the update, $\mathcal{F}(\mathbf{\Gamma}; \boldsymbol{\mathcal{X}}_{\text{new}}, \mathbf{Y}_{\text{new}})$ is given by Eq. (3) and

$$\begin{aligned}\mathcal{G}(\mathbf{\Gamma}, \tilde{\mathbf{\Gamma}}) &= \|\mathbf{W} - \tilde{\mathbf{W}}\|_F^2 + \|\mathbf{V} - \tilde{\mathbf{V}}\|_F^2 + \|\mathbf{A} - \tilde{\mathbf{A}}\|_F^2 \\ &+ \|\mathbf{B} - \tilde{\mathbf{B}}\|_F^2 + \|\mathbf{C} - \tilde{\mathbf{C}}\|_F^2. \end{aligned} \quad (4)$$

5.0.1 Incremental Learning over Space

First, we discuss WISDOM's approach for incremental learning over space, when data from a new location becomes available. Let T be the current time and S be the current number of locations. We assume that the new location has historical observation data from time t_0 to T. If the location has only one observation data, then $t_0 = T$. We further assume that the spatial latent features for other locations are unaffected by the addition of the new location, i.e., $\forall s : \tilde{\mathbf{A}}_s = \mathbf{A}_s$. However, the latent features for other modes ($\mathbf{B}$ and $\mathbf{C}$) as well as the parameters of the prediction models ($\mathbf{W}$ and $\mathbf{V}$) can be affected by the addition of the new data, $\boldsymbol{\mathcal{X}}_{\text{new}} = \{\mathbf{x}_{S+1,t_0}, \mathbf{x}_{S+1,t_0+1}, \cdots, \mathbf{x}_{S+1,T}\}$. For brevity, we denote the feature vectors for the new location as $\boldsymbol{\mathcal{X}}_{S+1}$, which is a tensor of size $1 \times (T - t_0 + 1) \times d$.

The objective function for incremental learning over space can be expressed as follows:

$$\begin{aligned}&\min_{\mathbf{W},\mathbf{V},\boldsymbol{\mathcal{V}},\mathbf{A}_{S+1},\mathbf{B},\mathbf{C}} \mathcal{Q}(\mathbf{W}, \mathbf{V}, \mathbf{A}_{S+1}, \mathbf{B}, \mathbf{C}, \tilde{\mathbf{W}}, \tilde{\mathbf{V}}, \tilde{\mathbf{B}}, \tilde{\mathbf{C}}) \\ =\ & \frac{1}{2}\sum_{t=t_0}^{T}\left[\mathbf{x}_{S+1,t}^T(\mathbf{W}^T\mathbf{A}_{S+1} + \mathbf{V}^T\mathbf{B}_t) - y_{S+1,t}\right]^2 \\ &+ \frac{\lambda_1}{2}\|\boldsymbol{\mathcal{X}}_{S+1} - [\![\mathbf{A}_{S+1}^T, \mathbf{B}, \mathbf{C}]\!]\|_F^2 + \frac{\eta_1}{2}\Big[\|\mathbf{W} - \tilde{\mathbf{W}}\|_F^2 \\ &\qquad + \|\mathbf{V} - \tilde{\mathbf{V}}\|_F^2 + \|\mathbf{B} - \tilde{\mathbf{B}}\|_F^2 + \|\mathbf{C} - \tilde{\mathbf{C}}\|_F^2\Big] \\ &+ \beta_1\|\mathbf{W}, \mathbf{V}, \mathbf{A}_{S+1}, \mathbf{B}, \mathbf{C}\|_1 \end{aligned} \quad (5)$$

η_1 is the hyperparameter to control the model consistency from previous learnt models, which can be tuned using cross validation. Note that $\mathbf{A}_{S+1}$ is a column vector that represents the spatial latent features for the new location and $\mathbf{x}_{S+1,t}$ denote the feature vector of the location at time t. The smoothness parameter η_1 determines the extent to which the previous model parameters should be retained. The optimization problem can be solved using the proximal gradient descent method. Derivation of the gradients of the objective function with respect to the various parameters are provided in the conference version of this paper [36].

5.0.2 Incremental Learning over Time

Next, we examine WISDOM's strategy for incremental learning over time. Let S be the number of locations and T be the current time. Similar to other online learning schemes, we assume the availability of the feature vectors of predictor variables for all S locations at time $T + 1$. This information will be used to determine the temporal latent factor $\mathbf{B}_{T+1}$ for the new time period. Similar to the strategy used for incremental learning over space, we assume the new data for time $T + 1$ does not affect previous temporal latent factors $\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_T$.

Our strategy for incremental learning over time is to perform the following two steps: first, we learn the temporal latent factor $\mathbf{B}_{T+1}$ based on the values of the predictor

variables for the new time period. The updated temporal latent factor is used to make an ensemble prediction using Eq. (1). Next, the model parameters and other latent factors are subsequently updated when the true values of the target variable for the new time period $\mathbf{Y}_{T+1}$ is available at all locations.

Step 1: Update formula for $\mathbf{B}_{T+1}$. The objective function for updating the temporal latent factor is given below:

$$\begin{aligned} \min_{\mathbf{B}_{T+1}} \mathcal{Q}(\mathbf{B}_{T+1}) &= \frac{\lambda_2}{2}\|\boldsymbol{\mathcal{X}}_{T+1} - [\![\tilde{\mathbf{A}}, \mathbf{B}_{T+1}^T, \tilde{\mathbf{C}}]\!]\|_F^2 \qquad (6) \\ &= \frac{\lambda_2}{2}\|\mathbf{X}_{T+1_{(2)}} - \mathbf{B}_{T+1}^T \left(\tilde{\mathbf{C}} \odot \tilde{\mathbf{A}}\right)^T \|_2^2, \end{aligned}$$

which can be solved in closed form. The decomposition can be performed before $\mathbf{Y}_{T+1}$ is observed.

Step 2: Update formula for model parameters and latent factors. After $\mathbf{Y}_{T+1}$ is observed, the update formula for the model parameters and other latent factors can be found by optimizing the following objective function:

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B}_{T+1},\mathbf{C}} \mathcal{Q}(\mathbf{W}, \mathbf{V}, \mathbf{A}, \mathbf{B}_{T+1}, \mathbf{C}, \tilde{\mathbf{W}}, \tilde{\mathbf{V}}, \tilde{\mathbf{A}}, \tilde{\mathbf{C}}) \\ = &\frac{1}{2}\sum_{s}^{S}(\mathbf{x}_{s,T+1}^T(\mathbf{W}^T\mathbf{A}_s + \mathbf{V}^T\mathbf{B}_{T+1}) - y_{s,T+1})^2 \\ + &\frac{\lambda_2}{2}\|\boldsymbol{\mathcal{X}}_{T+1} - [\![\mathbf{A}, \mathbf{B}_{T+1}^T, \mathbf{C}]\!]\|_F^2 \\ + &\frac{\eta_2}{2}(\|\mathbf{W} - \tilde{\mathbf{W}}\|_F^2 + \|\mathbf{V} - \tilde{\mathbf{V}}\|_F^2 + \|\mathbf{A} - \tilde{\mathbf{A}}\|_F^2 \\ + &\|\mathbf{C} - \tilde{\mathbf{C}}\|_F^2) + \beta_1(\|\mathbf{W}, \mathbf{V}, \mathbf{A}, \mathbf{B}_{T+1}, \mathbf{C}\|_1) \qquad (7) \end{aligned}$$

Solving Eq. (7) to obtain the update formula for $\mathbf{W}$, $\mathbf{V}$, $\mathbf{A}$, $\mathbf{B}_{T+1}$ and $\mathbf{C}$ is similar to the approach described for incremental learning over space. We omit their details due to lack of space.

5.0.3 Incremental Learning over Space-Time

The approaches described in the previous subsections can be combined to create a hybrid approach for incremental learning over both space and time. Specifically, the WISDOM algorithm can be initially applied to a subset of the locations at a given starting time. As time progresses, it will apply the model update approach for incremental learning over time to the newly acquired observation data. Similarly, when data from a new location becomes available, it will then invoke the update strategy for incremental learning over space.

A summary of the detailed algorithm is given in Algorithm 1.

Input: $\boldsymbol{\mathcal{X}}_{\text{new}}$, $\mathbf{Y}_{\text{new}}$, $\tilde{\mathbf{W}}$, $\tilde{\mathbf{V}}$, $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, $\tilde{\mathbf{C}}$, λ_1, λ_2, β_1, β_2, η_1, η_2, K;
if $\boldsymbol{\mathcal{X}}_{new}$ *is* $\boldsymbol{\mathcal{X}}_{S+1}$ **then**
  Solve $\mathbf{A_{S+1}}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{W}$ and $\mathbf{V}$ by optimizing Eq. (5);
  $\mathbf{A} = [\mathbf{A}; \mathbf{A}_{S+1}^T]$;
  $S = S + 1$;
else if $\boldsymbol{\mathcal{X}}_{new}$ *is* $\boldsymbol{\mathcal{X}}_{T+1}$ **then**
  Solve $\mathbf{B}_{T+1}$ by optimizing Eq. (6);
  $\forall s' < S$, predict $\hat{y}_{s',T+1}$ by Eq. (1) using $\mathbf{B}_{T+1}$ and previous $\mathbf{A}$, $\mathbf{W}$ and $\mathbf{V}$;
  $\forall s' < S$, observe $y_{s',T+1}$;
  Solve $\mathbf{A}$, $\mathbf{B}_{T+1}$, $\mathbf{C}$, $\mathbf{W}$ and $\mathbf{V}$ by optimizing Eq. (7);
  $\mathbf{B} = [\mathbf{B}; \mathbf{B}_{T+1}^T]$;
  $T = T + 1$;

Algorithm 1: Pseudocode for incremental learning over space and time by WISDOM algorithm.

5.1 Complexity Analysis

This section presents the computational complexity of the WISDOM algorithm. For incremental learning over space, the time complexity for calculating the gradients for each iteration for the proximal gradient descent algorithm is $O(dTk)$. Since both d and k are constants and are typically smaller than T, the complexity is linear in time. Similarly, the runtime complexity for incremental learning over time can be shown to be $O(dSk)$, whenever the data for a new time period is observed.

In order to update the latest models, WISDOM will need to maintain the model parameters ($\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{W}$ and $\mathbf{V}$), and the current observation $\boldsymbol{\mathcal{X}}_{\text{new}} \in R^{S\times 1\times d}$ and $\mathbf{Y}_{\text{new}} \in R^{S\times 1}$ for new station (or $\boldsymbol{\mathcal{X}}_{\text{new}} \in R^{1\times T\times d}$ and $\mathbf{Y}_{\text{new}} \in R^{1\times T}$ for new time). Hence, the space complexity is $O(Sd + Sk + Tk + dk)$ or $O(Td + Sk + Tk + dk)$. The WISDOM algorithm is clearly much more efficient, both in terms of its runtime and space complexities, compared to its batch counterpart.

6 Experimental Evaluation

We use a global-scale climate dataset to evaluate the performance of the SMART framework. We consider two implementations of the framework, a batch learning algorithm (denoted as SMART-b), which must be retrained from scratch each time new observations are available, and an incremental learning algorithm (denoted as WISDOM).[1] The goals of our experiments are as follows:

1) To demonstrate the value of using an ensemble of spatial and temporal prediction models.
2) To show the trade-off in accuracy and efficiency between SMART-b and WISDOM.
3) To compare the performance of WISDOM against other baseline algorithms.
4) To provide possible interpretations with domain knowledge of the latent factors derived by the SMART framework.
5) To determine the value of incorporating known patterns from the domain into the SMART framework.

6.1 Dataset Description

The climate data used in this study was obtained from two different sources. First, we downloaded the monthly climate observation data from the *Global Surface Summary of Day* (GSOD)[2] website. These monthly values of total precipitation (prcp), maximum (tmax), minimum (tmin), and average (tmean) temperature are used to represent the target variable of our prediction tasks. We created 4 datasets,

1. We published the SMART-b and WISDOM code at https://github.com/Jianpeng-Xu/TKDE-SMART
2. https://data.noaa.gov/dataset/dataset/global-surface-summary-of-the-day-gsod

one for each response variable. The second data source corresponds to a coarse-scale gridded climate data from NCEP reanalysis[3]. We use the data to create the predictor variables for our climate prediction task. Although there are hundreds of variables available in the NCEP reanalysis data, we select 13 of them as our predictor variables with the help of our domain expert. A detailed description of the selected features is given by Table I in [36].

GSOD provides monthly climate data from more than 30,000 monitoring sites worldwide, spanning a time period between 1942 to the present time. Since the data from earlier time periods are mostly missing, we restrict our experiment to data from January 1985 to November 2015 (when we collected our dataset) for a total of 371 months. During preprocessing, we remove the sites that have missing values. Meanwhile, we randomly choose one site if multiple sites are co-located in the same grid. Thus, each grid of the NCEP reanalysis data contains only one GSOD site. This reduces the number of sites in our data set to 1,118. Hence, the size of the spatio-temporal tensor after preprocessing is 1118 locations $\times$ 371 time steps $\times$ 13 predictor variables, which contains more than 5.3 million elements. We take the first 20 years of the data as training set and the last 11 years as test set. As a data pre-processing step, we use the seasonal mean and standard deviation from the training data for each timeseries to deseasonalize and standardize both training and test sets.

6.2 Analysis of SMART Framework

In this section, we demonstrate the value of using an ensemble of spatial and temporal multi-task learning models in supervised learning. For this experiment, we train the models on one snapshot of the data (predictor variables of all 31-year data, and response variable of the first 20-year data from all stations), and evaluate them using the rest of the data (the last 11-year data of all stations). Note that the nature of the application[4] allows us to utilize the predictor variables for the test set during the training process. We compare SMART-b against its two variants: SMART-S and SMART-T, where SMART-S is a variation of the framework that considers only the spatial component of SMART-b (using only $\mathbf{A}$ in the first term of Eq. 3), and SMART-T is the variation that considers only the temporal component of SMART-b (using only $\mathbf{B}$ in the first term of Eq. 3). We use mean absolute error (MAE) over locations to evaluate the performance. The comparison result is shown in Table 1. Note that SMART-b performs slightly better than SMART-S in all 4 response variables and significantly outperforms SMART-T in all response variables. This suggests that having an ensemble of spatial and temporal models can improve predictive performance compared to modeling using the spatial or temporal factors individually.

6.3 Experimental Setup for Incremental Learning

In this section, we describe the setup for our following experiments, where the models can either be generated from scratch (using a batch algorithm) or updated from an existing model (using an incremental learning approach). In simulating the incremental learning process, each new observation may correspond to a randomly selected new location or a subsequent time period. The models must make a prediction first before they receive feedback on whether their predictions are correct. All the models will be continuously updated for the entire 30-year time period. We use the first 10 years (1985 to 1994) of data as an initial training period, and the performance of the models for the next 10 years (1995 to 2004) will be used for validation. We use the predictions made in the last 11 years (2005 to 2015) to assess the accuracy of the models. The same setup also applies to SMART-b, where the spatial and temporal models are re-built from scratch each time a new observation is available during the test period. For SMART-b and WISDOM, we set their number of latent factors to $k = 5$ and randomly select 100 sites as the initial starting locations.

TABLE 1: MAE for SMART-b and its variants

	tmax	tmin	tmean	prcp
SMART-b	**0.5178**	**0.5962**	**0.4696**	**0.7411**
SMART-S	0.5268	0.6005	0.4698	0.7451
SMART-T	0.6016	0.7041	0.5756	0.7660

A customized MAE is used to evaluate the performance of the various algorithms:

$$\text{MAE} = \sum_{t}^{T} \frac{\sum_{s=1}^{S_t} |y_{s,t} - \hat{y}_{s,t}|}{\sum_{s=1}^{S_t} 1} / T,$$

where S_t denotes all available locations at time t.

The MAE is calculated for each station s starting from its corresponding test period. For example, if a station was introduced during the training or validation period, then its MAE is calculated for the entire 10-year test period. However, if the station was first introduced during the testing period, its MAE is computed starting from the time it was introduced. We repeated the experiments 5 times and reported their averaged MAE and the standard deviation over the 5 runs, where the sequence of observations (from new location or new time) in each run is randomly chosen using different seeds.

6.4 Comparison between SMART-b and WISDOM

We first compare the batch learning algorithm SMART-b against the incremental learning algorithm WISDOM in Table 2. As expected, SMART-b performs better than WISDOM in all datasets, due to the additional information from the historical data. However, SMART-b will not be able to handle newly observed data efficiently as it will need to re-trained from scratch each time when there are new time points or locations available. This reflects the trade-off between the accuracy and efficiency of the modeling. In particular, using the same computing environment, the average runtime for each update step for SMART-b and WISDOM over four response variables are given in Table 2. Note that there are altogether 1388 update steps in our experiment, including 1018 steps for incremental update over space and 370 steps for incremental updates over time. The results suggest that the average runtime for WISDOM is over 60 times faster than that of SMART-b.

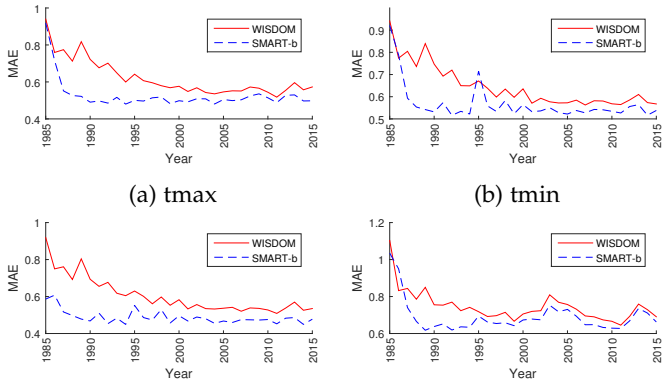
3. http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.derived.html

4. In practice, this is considered a semi-supervised learning scenario, where the predictors are obtained from the outputs of climate models.

TABLE 2: Performance comparison for SMART-b and WISDOM

	MAE (std)			Average Runtime per Update		
	SMART-b	WISDOM	Accuracy Loss	SMART-b	WISDOM	Runtime Gain
tmax	0.5445(0.0033)	0.5632(0.0032)	3.43%	35.74(1.5137)	0.5270(0.0068)	66.8×
tmin	0.5982(0.0042)	0.6164(0.0045)	3.04%	35.51(1.8677)	0.5245(0.0015)	66.7×
tmean	0.5005(0.0048)	0.5331(0.0059)	6.51%	35.76(1.2443)	0.5261(0.0073)	67.0×
prcp	0.7723(0.0191)	0.7756(0.0149)	0.43%	36.52(3.0064)	0.5232(0.0016)	68.8×

In addition, we also examine how well WISDOM performs over time compared to SMART-b. The yearly averaged MAE results are shown in Fig. 3 (this figure is generated based on one run of the experiments). As expected, SMART-b has a lower error compared to WISDOM on all 4 datasets for almost all of the years. Meanwhile, the MAE of WISDOM gradually becomes closer to the MAE of SMART-b after the first 10 years of training period, with the difference between the two stabilizing after the year 2000. The results given in Table 2 further suggest that the prediction accuracy for WISDOM is only marginally worse than that for SMART-b, with an accuracy loss of at most 6.51%.



(a) tmax (b) tmin

(c) tmean (d) prcp

Fig. 3: Error comparison over time between SMART-b and WISDOM.

6.5 Comparison between WISDOM and Baseline Methods

Next, we compare the performance of WISDOM against the following baseline incremental learning algorithms. To ensure fairness, the order in which new observations are introduced over space and time is the same for all the algorithms of each run:

1) **STL** (Single Task Learning): Each location has its own local (linear) model that is incrementally updated using a gradient descent approach when it has a new observation data. When a new location is introduced, its parameters are randomly initialized and updated only when new observation data for the location becomes available.

2) **ALTO**: This is an adaptation of the method in [39], which assumes the model parameters for multiple response variables are in the form of a tensor. To extend ALTO to our problem setting, we make the following changes: First, the tensor is reduced to a matrix $\mathbf{W}$ since each data set has only one response variable. Second, we replace tensor decomposition with singular value decomposition and apply it to the noise-perturbed weight matrices to obtain the updated model parameters. We have also extended ALTO to perform incremental learning over space: when data from a new location is available, we apply linear regression to the new data and add the estimated parameters as a new row in $\mathbf{W}$. The modified $\mathbf{W}$ is then projected to its low-rank representation.

In addition to the two baseline methods, we also consider the following two variations of WISDOM:

3) **WISDOM-S**: This variant considers only the spatial component of the framework. Specifically, we remove all terms related to $\mathbf{V}$ in Eq. (5) and (7).

4) **WISDOM-T**: This variant considers only the temporal component of the framework. We remove all terms related to $\mathbf{W}$ in Eq. (5) and (7).

We first present the results comparing WISDOM against the two baseline algorithms, STL and ALTO. Table 3 shows that WISDOM outperforms STL for all target variables, which suggests the importance of incorporating spatial autocorrelation into the learning framework. WISDOM also outperforms ALTO, which is another online tensor learning approach for spatio-temporal data. There are two possible reasons for this. First, ALTO performs the following simple update to its weight matrix each time new observation data is available[5]: $\mathbf{W}^{(k)} = (1-\alpha)\mathbf{W}^{(k-1)} + \alpha\mathbf{X}\mathbf{Z}^{\dagger}$ [39]. The single-step update may not be sufficient to learn the right weights of the prediction model. In contrast, WISDOM learns the optimal weights that minimize an incrementally updated objective function. Second, ALTO performs a low-rank decomposition on a perturbed weight matrix whereas WISDOM decomposes the data tensor itself. The results suggest that the latter strategy is more effective as the observation data is potentially noisy.

Next, we compare WISDOM against its variants in Table 3. Observe that WISDOM and WISDOM-S outperform WISDOM-T on 3 out of 4 data sets, which suggest the importance of incorporating a predictive model from the spatial latent factors. Furthermore, both WISDOM and WISDOM-T outperform WISDOM-S in terms of precipitation prediction. This makes sense as precipitation generally tends to have a smaller spatial autocorrelation compared to temperature [6], which is why temporal autocorrelation plays a more significant role in improving its prediction.

6.6 Convergence Analysis of WISDOM

To demonstrate the convergence of WISDOM, Fig. 4 shows the average MAE of WISDOM for all locations across time for one of the 5 experimental runs. A location is included in the average MAE calculation only after the data for the location becomes available. WISDOM starts to converge after the first 10 years, which is our initial training period.

5. We use incremental update of the weight matrix instead of exact update since the latter requires the entire data to be available in memory.

TABLE 3: MAE for baseline methods, WISDOM and its variant methods

	STL	ALTO	WISDOM	WISDOM-S	WISDOM-T	WISDOM-KP
tmax	0.7550 (0.0043)	0.7656 (0.0023)	**0.5632** (0.0032)	0.594 (0.0220)	0.5982 (0.0083)	0.5804 (0.0052)
tmin	0.7376 (0.0050)	0.7614 (0.0010)	**0.6164** (0.0045)	0.6304 (0.0119)	0.6509 (0.0061)	0.6349(0.0024)
tmean	0.7138 (0.0030)	0.7375 (0.0018)	**0.5331** (0.0059)	0.5512 (0.0212)	0.5604 (0.0118)	0.5516(0.0026)
prcp	0.9550 (0.0161)	0.7963 (0.0140)	**0.7756** (0.0149)	0.8850 (0.0358)	0.8445 (0.0269)	0.7823 (0.0158)

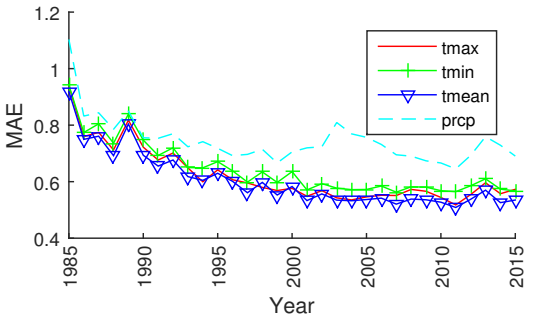


Fig. 4: Changes in MAE over time for WISDOM

Note that for precipitation dataset, the error increases after the training period and converges gradually in the test period. This is perhaps due to the way the dataset was deseasonalized and standardized (see Section 6.1). Specifically, we assume that the seasonal means and standard deviations do not change significantly from the training period to the test period. While more sophisticated preprocessing steps can be performed, this goes beyond the scope of this paper.

6.7 Analysis of Spatial Latent Factors

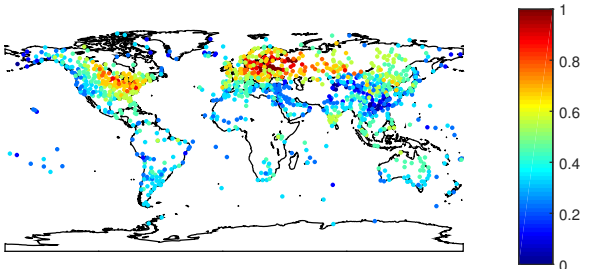
Next, we investigate the spatial latent factors derived by WISDOM. Each spatial latent factor is a vector whose elements represent the membership of each location to the given latent factor. Fig. 5 shows one example of the spatial distribution of the latent factors for precipitation prediction. The figure shows that the latent factors have varying spatial distributions, which suggests that they capture different aspects of the spatial variability in the data. For example, the first latent factor is dominant in Europe and northeast of America whereas the second latent factor is more influential in the eastern part of China.

WISDOM utilizes the spatial latent factors to perform incremental learning over space. To further demonstrate the benefit of incremental learning over space, we compare the average annual MAE for the first 100 randomly chosen locations when the model is updated with and without incremental learning over space. Specifically, in the latter case, no new locations are added into the data set as time progresses. As shown in Fig 6, adding data from new locations indeed helps to improve the MAE of the first 100 randomly chosen locations.

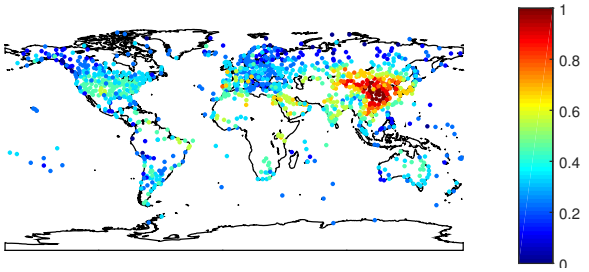
6.8 Analysis of Temporal Latent Factors

Each temporal latent factor derived by WISDOM can be represented as a time series. To understand their significance, we correlate[6] the temporal latent factors against the known climate indices, including Arctic Oscillation Index (AOI),

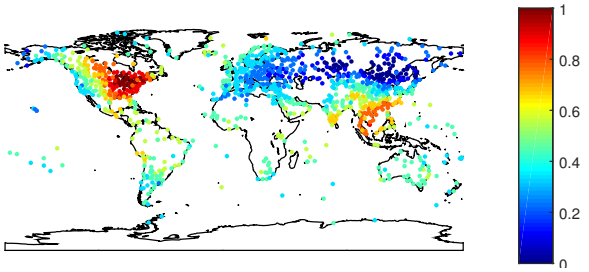
6. We used the absolute value of the correlation to remove the sign effect.



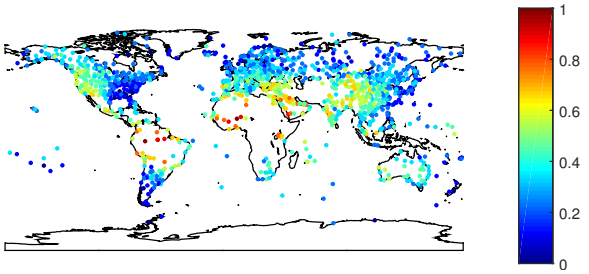
(a) Factor 1



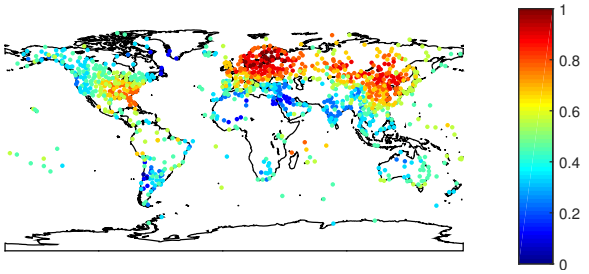
(b) Factor 2



(c) Factor 3

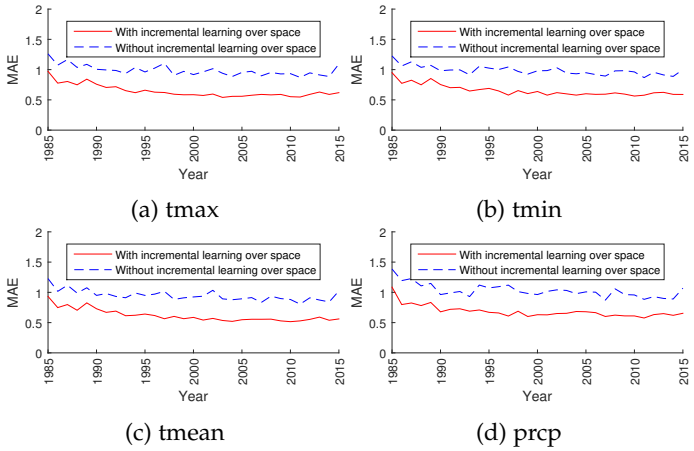


(d) Factor 4



(e) Factor 5

Fig. 5: Spatial distribution of the spatials factor learned by WISDOM for prcp. (Figure is best viewed in color).

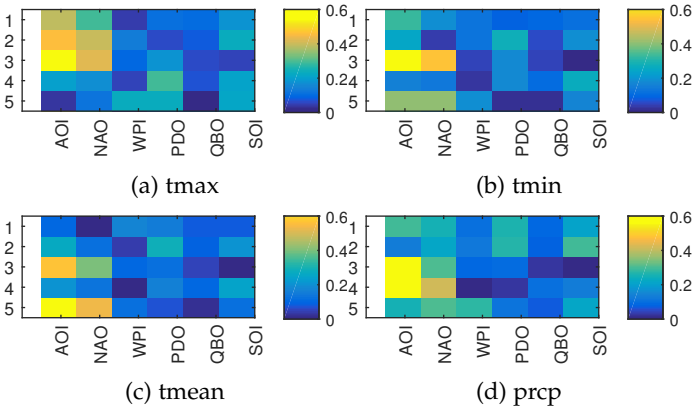


(a) tmax (b) tmin

(c) tmean (d) prcp

Fig. 6: Average annual MAE comparison between WISDOM with incremental learning over space and WISDOM without incremental learning over space for the 100 initially chosen locations

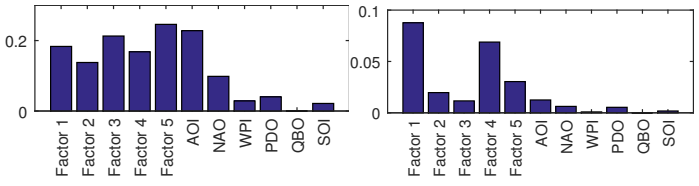
North Atlantic Oscillation (NAO), Western Pacific Index (WPI), Quasi-Biennial Oscillation (QBO), Pacific Decadal Oscillation (PDO), and Southern Oscillation Index (SOI). Fig. 7 shows the resulting correlation for the tmax, tmin, tmean and prcp data sets based on one run of the experiment. Though the temporal latent factors for all data sets are different, we found some of the factors correlate highly (over 0.6) with the existing climate indices. This result suggests that the temporal latent factors may capture some of the previously known climate phenomena, represented by the climate indices such as AOI and NAO. For each temporal latent factor and climate index, we also calculate the percent of locations whose temperature or precipitation has a correlation above 0.3. The results in Fig. 8 suggest that (1) not all climate indices have a significant number of locations highly correlated with them and (2) some latent factors have significant correlation with a relatively large number of locations, comparable to the known indices. More importantly, as some of the latent factors do not correlate highly with the known indices, this suggests that our framework can potentially discover new indices that capture the climate variability for many locations.



(a) tmax (b) tmin

(c) tmean (d) prcp

Fig. 7: Correlations between the climate indices and the temporal factors learned from WISDOM



(a) tmean (b) prcp

Fig. 8: Percentage of locations whose response variables has a correlation above 0.3 with the temporal factors learned from WISDOM for tmean/prcp and climate indices

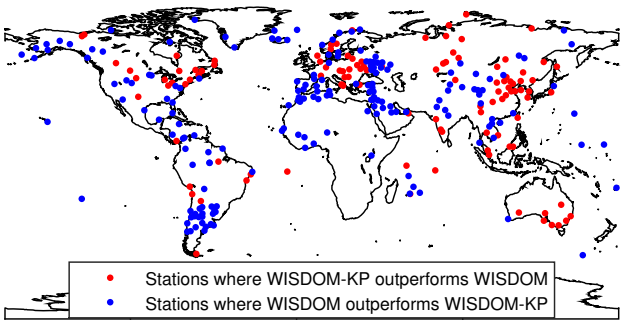


Fig. 9: Stations where WISDOM-KP outperforms WISDOM more than 0.05 in MAE evaluation for tmin and vise versa.

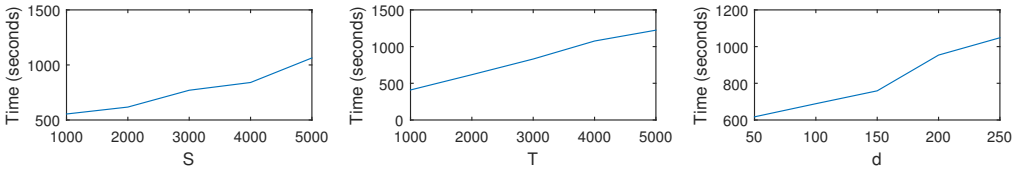
6.9 Incorporating Known Patterns (WISDOM-KP)

Surprisingly, the results from Section 6.8 show that none of the temporal latent factors were found to correlate highly with SOI, which is a surrogate time series for El Niño. One of the strengths of WISDOM is its ability to incorporate known domain patterns as additional constraints for its formulation. In order to incorporate known patterns such as SOI, we simply fix one of the columns in the temporal latent factor matrix $\mathbf{B}$ to be the time series of SOI and learn the remaining spatial and temporal latent factors using WISDOM. We denote this approach as **WISDOM-KP**. The MAE results comparing WISDOM against WISDOM-KP are shown in Table 3. The results suggest that WISDOM-KP achieves comparable (although slightly worse) results as WISDOM in terms of their average MAE.

In addition, we also compared the number of locations where WISDOM-KP outperforms WISDOM in one of the 5 runs. The MAE for WISDOM-KP is lower than WISDOM in around 30% of the locations. This is not surprising as we do not expect SOI to accurately capture the climate variability for all locations. Instead, there are locations that are expected to benefit from using SOI as one of the temporal latent factors. To identify such locations, we plot a map of the locations in which WISDOM-KP is better than WISDOM, and vice-versa, for predicting tmin in Fig. 9. The results suggest that by incorporating SOI, an improved predictive performance is observed in areas such as Australia, part of South America, northeast of North America, part of Europe and some locations around Arctic Ocean. Some of these locations are consistent with the results of previous studies [27].

6.10 Time Complexity and Scalability

We have analyzed the time complexity of WISDOM in Section 5.1. To show the scalability of WISDOM on different



(a) Runtime by varying S (b) Runtime by varying T (c) Runtime by varying d

Fig. 10: Scalability test for WISDOM on different dimensions.

dimensions empirically, we create a set of synthetic datasets by fixing the sizes of two dimensions of a 3-mode tensor and varying the size of the third dimension. To test the scalability of the algorithm over space, we randomly generate a set of datasets ($\boldsymbol{\mathcal{X}} \in \Re^{S\times T\times d}, \mathbf{Y} \in \Re^{S\times T}$) by setting $T = 2000$, $d = 20$, and varying $S = \{1000, 2000, 3000, 4000, 5000\}$. Similarly, to test the scalability over time, we set $S = 2000$, $d = 20$, and vary $T = \{1000, 2000, 3000, 4000, 5000\}$, and to test the scalability over feature space, we set $S = 2000$, $T = 2000$ and vary $d = \{50, 100, 150, 200, 250\}$. We run the WISDOM algorithm on each of the datasets and record the runtime, which are illustrated in Fig. 10. Note that the runtime is almost linear with respect to S, T and d, which is consistent with our theoretical analysis and validates the scalability of the WISDOM framework.

7 Conclusions

This paper presents a spatio-temporal multi-task learning framework named SMART for multi-location prediction based on supervised tensor decomposition. The proposed framework constructs both spatial and temporal prediction models of the data and aggregates the output to obtain the final prediction. In order to efficiently handle the incremental learning scenario where the data becomes available repeatedly over space or over time, a novel incremental learning algorithm called WISDOM is developed to simultaneously extract the latent factors of the spatio-temporal data and learn the spatial and temporal prediction models only based on the current observed data and the old models learned previously. The experiments for evaluating the methods are performed on a global-scale climate data. The results show the trade-off between accuracy and efficiency by comparing WISDOM and SMART. SMART outperforms WISDOM in all datasets in terms of accuracy, but has a much higher computational complexity. In addition, WISDOM outperforms several baseline algorithms under the incremental learning scheme and can easily accommodate known patterns from the spatio-temporal domain.

Acknowledgments

This research is partially supported by NOAA Climate Program office through grant #NA12OAR4310081, NASA Terrestrial Hydrology Program through grant #NNX13AI44G, and NSF grants #IIS-1565596, IIS-1615597 and IIS-1615612.

References

[1] Z. Abraham, M. Liszewska, Perdinan, P.-N. Tan, J. Winkler, and S. Zhong. Distribution regularized regression framework for climate modeling. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 333–341, 2013.

[2] A. Afshar, J. C. Ho, B. Dilkina, I. Perros, E. B. Khalil, L. Xiong, and V. Sunderam. Cp-ortho: An orthogonal tensor factorization framework for spatio-temporal data. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 67:1–67:4, 2017.

[3] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832, Jan. 2014.

[4] G. Atluri, A. Karpatne, and V. Kumar. Spatio-temporal data mining: A survey of problems and methods. *ACM Comput. Surv.*, 51(4):83:1–83:41, Aug. 2018.

[5] S. Boriah, V. Kumar, M. Steinbach, C. Potter, and S. Klooster. Land cover change detection: A case study. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 857–865, 2008.

[6] T. A. Buishand and T. Brandsma. Multisite simulation of daily precipitation and temperature in the rhine basin by nearest-neighbor resampling. *Water Resources Research*, 37(11):2761–2776, 2001.

[7] A. Bulat, J. Kossaifi, G. Tzimiropoulos, and M. Pantic. Incremental multi-domain learning with network latent tensor factorization. In *arXiv*, 2019.

[8] R. Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, July 1997.

[9] K.-W. Chang, W.-T. Yih, B. Yang, and C. Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014.

[10] J. Chen, J. Zhou, and J. Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge discovery and data mining*, pages 42–50, 2011.

[11] A. Cichocki, D. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. Phan. Tensor decompositions for signal processing applications: From two-way to multiway component analysis. *Signal Processing Magazine*, 32(2):145–163, 2015.

[12] N. Cressie. The origins of kriging. *Mathematical Geology*, 22(3):239–252, Apr 1990.

[13] G. Davis, N. Sevdalis, and L. Drumright. Spatial and temporal analyses to investigate infectious disease transmission within healthcare settings. *Journal of Hospital Infection*, pages 227–243, 2014.

[14] Y. Du, Y. Zheng, K. Lee, and S. Zhe. Probabilistic streaming tensor decomposition. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 99–108.

[15] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 109–117, 2004.

[16] A. R. Goncalves, A. Banerjee, and F. J. V. Zuben. Spatial projection of multiple climate variables using hierarchical multitask learning. In *Proceedings of the 31st Conference on Artificial Intelligence (AAAI)*, 2017.

[17] W. Hu, X. Li, X. Zhang, X. Shi, S. Maybank, and Z. Zhang. Incremental tensor subspace learning and its applications to foreground segmentation and tracking. *International Journal of Computer Vision*, 91(3):303–327, 2011.

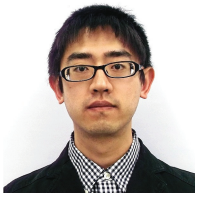
[18] Z. Jiang and S. Shekhar. *Spatial Big Data Science*. Springer, 2017.

[19] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, Aug. 2009.

[20] X. Liu, P.-N. Tan, Z. Abraham, L. Luo, and P. Hatami. Distribution preserving multi-task regression for spatio-temporal data. In *IEEE International Conference on Data Mining (ICDM)*, pages 1134–1139, Nov 2018.

[21] C. Monteleoni, G. Schmidt, and S. McQuade. Climate informatics: Accelerating discovering in climate science with machine learning. *Computing in Science Engineering*, 15(5):32–40, Sept 2013.

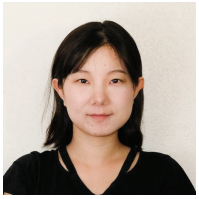
[22] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, Jan. 2014.
[23] B. Romera-Paredes, H. Aung, N. Bianchi-Berthouze, and M. Pontil. Multilinear multitask learning. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 1444–1452, 2013.
[24] H. Rue and L. Held. *Gaussian Markov Random Fields: Theory And Applications (Monographs on Statistics and Applied Probability)*. Chapman & Hall/CRC, 2005.
[25] S. Shekhar, Z. Jiang, R. Y. Ali, E. Efteliogolu, X. Tang, V. M. V. Gunturi, and X. Zhou. Spatiotemporal data mining: A computational perspective. *International Journal of Geo-Information*, 4:2306–2338, 2015.
[26] S. Smith, K. Huang, N. D. Sidiropoulos, and G. Karypis. Streaming tensor factorization for infinite data sources. In *Proceedings of the 2018 SIAM International Conference on Data Mining, SDM 2018*, pages 81–89.
[27] M. Steinbach, P.-N. Tan, V. Kumar, S. Klooster, and C. Potter. Discovery of climate indices using clustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 446–455, 2003.
[28] K. Subbian and A. Banerjee. Climate multi-model regression using spatial smoothing. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pages 324–332, 2013.
[29] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 374–383, 2006.
[30] J. Sun, D. Tao, S. Papadimitriou, P. S. Yu, and C. Faloutsos. Incremental tensor analysis: Theory and applications. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 2(3):11:1–11:37, Oct. 2008.
[31] W. Tobler. A computer movie simulating urban growth in the detroit region. *Economic Geography*, 46(2):234–240, 1970.
[32] K. Wimalawarne, M. Sugiyama, and R. Tomioka. Multitask learning meets tensor factorization: task imputation via convex optimization. In *Advances in Neural Information Processing Systems 27*, pages 2825–2833. 2014.
[33] F. Wu, X. Tan, Y. Yang, D. Tao, S. Tang, and Y. Zhuang. Supervised nonnegative tensor factorization with maximum-margin constraint. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.
[34] J. Xu, P.-N. Tan, L. Luo, and J. Zhou. Gspartan: a geospatio-temporal multi-task learning framework for multi-location prediction. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, 2016.
[35] J. Xu, P.-N. Tan, J. Zhou, and L. Luo. Online multi-task learning framework for ensemble forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 29(6):1268–1280, 2017.
[36] J. Xu, J. Zhou, P.-N. Tan, X. Liu, and L. Luo. Wisdom: Weighted incremental spatio-temporal multi-task learning via tensor decomposition. In *2016 IEEE International Conference on Big Data*, pages 522–531, Dec 2016.
[37] J. Ye, L. Sun, B. Du, Y. Fu, X. Tong, and H. Xiong. Co-prediction of multiple transportation demands based on deep spatio-temporal neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 305–313, 2019.
[38] K. Yu, V. Tresp, and A. Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of the 22Nd International Conference on Machine Learning*, pages 1012–1019, 2005.
[39] R. Yu, D. Cheng, and Y. Liu. Accelerated online low rank tensor learning for multivariate spatiotemporal streams. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning*, volume 37, pages 238–247, 2015.
[40] R. Yu, G. Li, and Y. Liu. Tensor regression meets gaussian processes. In *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 482–490.
[41] S. Yuan, J. Zhou, P.-N. Tan, E. Fergus, T. Wagner, and P. Soranno. Multi-Level Multi-Task Learning for Modeling Cross-Scale Interactions in Nested Geospatial Data. In *Proceedings of the IEEE International Conference on Data Mining*, 2017.
[42] Y. Zheng. Trajectory data mining: An overview. *ACM Transactions on Intelligent Systems and Technology*, 6(3):Article No. 9, 2015.
[43] J. Zhou, J. Chen, and J. Ye. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Arizona State University, 2011.
[44] J. Zhou, L. Yuan, J. Liu, and J. Ye. A multi-task learning formulation for predicting disease progression. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 814–822, 2011.
[45] S. Zhou, X. V. Nguyen, J. Bailey, Y. Jia, and I. Davidson. Accelerating Online CP Decompositions for Higher Order Tensors. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1375–1384, 2016.



Jianpeng Xu is a senior Data Scientist with WalmartLabs. He received his Ph.D from the Department of Computer Science and Engineering at Michigan State University, 2017. His research focuses on geospatio-temporal data mining, recommendation system, and personalized modeling. He has published in top data mining conferences such as KDD, ICDM, SDM and IJCAI. One of his papers won the best research paper award at IEEE BigData 2016.

Jiayu Zhou is an Assistant Professor in the Department of Computer Science and Engineering at Michigan State University. Before joining MSU, he was a staff research scientist at Samsung Research America. Jiayu received his Ph.D. degree in computer science at Arizona State University in 2014. He has a broad research interest in large-scale machine learning, data mining, and biomedical informatics. He has published in top machine learning and data mining venues including NIPS, KDD, and ICDM. One of his papers won the best student paper award at ICDM 2014.

Pang-Ning Tan is a Professor in the Department of Computer Science and Engineering at Michigan State University. He received his Ph.D. degree in Computer Science from University of Minnesota, 2002. His research interests focus on the development of novel data mining algorithms for a broad range of applications, including climate and ecological sciences, cybersecurity, and network analysis. He has served as associate editor and program committee chairs for several international journals and conferences. His research has been supported by the National Science Foundation, Office of Naval Research, Army Research Office, National Aeronautics and Space Administration, National Oceanic and Atmospheric Administration, National Institutes of Health, and Michigan State University.



Xi Liu is a Ph.D. student in the Department of Computer Science and Engineering at Michigan State University. She received her Bachelor's degree in Electronics and Communications Engineering Department from Xi'an Jiaotong University, 2011. Xi's research focuses on data mining and machine learning techniques on sensing data. One of her work received the 1st place winner in Discovery Challenge of ECML-PKDD'2016.

Lifeng Luo is an Associate Professor in the Department of Geography at Michigan State University. He received his Ph.D. degree from Rutgers University in 2003. He was a research scientist at Princeton University before joining MSU in August, 2009. He is also affiliated with the Environmental Sciences and Policy Program, Center for Water Sciences, and Center for Global Change and Earth Observations. His research covers a range of topics related to land-atmosphere interaction and its impact on the global climate and hydrological cycle at various spatial and temporal scales. His recent research focuses on the predictability and prediction of climate extremes such as drought, floods, heat waves at subseasonal to seasonal scale.