# MULTI-TASK LEARNING AND ITS APPLICATION TO GEOSPATIO-TEMPORAL DATA

By

Jianpeng Xu

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Computer Science and Engineering – Doctor of Philosophy

2017

## ABSTRACT

MULTI-TASK LEARNING AND ITS APPLICATION TO GEOSPATIO-TEMPORAL DATA

By

Jianpeng Xu

Multi-task learning (MTL) is a data mining and machine learning approach for modeling multiple prediction tasks simultaneously by exploiting the relatedness among the tasks. MTL has been successfully applied to various domains, including computer vision, healthcare, genomics, recommender systems, and natural language processing. The goals of this thesis are: (1) to investigate the feasibility of applying MTL to geospatio-temporal prediction problems, particularly those encountered in the climate and environmental science domains and (2) to develop novel MTL frameworks that address the challenges of building effective predictive models from geospatio-temporal data.

The first contribution of this thesis is to develop an online temporal MTL framework called ORION for ensemble forecasting problems. Ensemble forecasting uses a numerical method to simulate the evolution of nonlinear dynamic systems, such as climate and hydrological systems. ORION aims to effectively aggregate the forecasts generated by different ensemble members for a future time window, where each forecast is obtained by perturbing the starting condition of the computer model or using a different model representation. ORION considers the prediction for each time point in the forecast window as a distinct prediction task, where the task relatedness is achieved by imposing temporal smoothness and mean regularization constraints. A novel, online update with restart strategy is proposed to handle missing observations in the training data. ORION can also be optimized for different objectives, such as $\varepsilon$-insensitive and quantile loss functions.

The second contribution of this thesis is to propose a MTL framework named GSpartan that can perform inferences at multiple locations simultaneously while allowing the local models for different locations to be jointly trained. GSpartan assumes that the local models share a common, low-rank representation and employs a graph Laplacian regularization to enforce constraints due to the inherent spatial autocorrelation of the data. Sparsity and non-negativity constraints are also incorporated into the formulation to ensure interpretability of the models.

GSpartan is a MTL framework that considers only the spatial autocorrelation of the data. It is also a batch learning algorithm, which makes it difficult to scale up to global-scale data. To address these limitations, a new framework called WISDOM is proposed, which can incorporate the task relatedness across both space and time. WISDOM encodes the geospatio-temporal data as a tensor and performs supervised tensor decomposition to identify the latent factors that capture the inherent spatial and temporal variabilities of the data as well as the relationship between the predictor and target variables. The framework is unique in that it trains distinct spatial and temporal prediction models from the latent factors of the decomposed tensor and aggregates the outputs of these models to obtain the final prediction. WISDOM also employs an incremental learning algorithm that can systematically update the models when training examples are available for a new time period or for a new location.

Finally, the geospatio-temporal data for many scientific applications are often available at varying spatial scales. For example, they can be generated by computer models simulated at different grid resolutions (e.g., the global and regional models used in climate modeling). A simple way to handle the predictor variables generated from the multi-scale data is to concatenate them into a single feature vector and train WISDOM using the concatenated vectors. However, this strategy may not be effective as it ignores the inherent dependencies between variables at different scales. To overcome this limitation, this thesis presents an extension of WISDOM called MUSCAT for handling multi-scale geospatio-temporal data. MUSCAT considers the consistency of the latent factors extracted from the spatio-temporal tensors at different scales while inheriting the benefits of WISDOM. Given the massive size of the multi-scale spatio-temporal tensors, a novel, supervised, incremental multi-tensor decomposition algorithm is develop to efficiently learn the model parameters.

## ACKNOWLEDGEMENTS

Spending five or more years of the life-time for a Ph.D degree is a great commitment for any human beings, including myself. During this fantastic journey as part of my limited life, what I have experienced has and will continue to influence the paths of my personal and career life. There are always moments and fragments in the memory that are related to those wonderful people I met, lived, and worked with. I would like to sincerely acknowledge each of them for being supportive, informative, and resourceful during my adventurous Ph.D study.

First and foremost, I would like to thank my advisor, Dr. Pang-Ning Tan for the continuous support of my Ph.D study and research. He is a very patient, motivated, knowledgable and humble mentor. He is willing to spend time on discussing not only the big pictures of my research, but also technical details of the work. His guidance on how to correctly do research, how to identify research problems and how to dive deep in the analysis of results is valuable and greatly benefits my research. I am extremely grateful to those opportunities he provided for pursuing problems of my interest, for attending conferences to make my work visible to other researchers, and for interning at industrial and research organizations to expose myself to different real world problems. Because of him, my Ph.D study in Michigan State University was memorable, enlightening, and less stressful. It is my honor to be able to work with Dr. Tan and I will cherish everything he dedicated during my stay.

Next, I would like to thank the rest of my Ph.D committee members, Dr. Jiayu Zhou, Dr. Lifeng Luo and Dr. Abdol-Hossein Esfahanian for their support and guidance throughout my Ph.D program. From Dr. Zhou, I gained considerably amount of practical machine learning techniques for solving optimization problems efficiently, and it is always fruitful in every discussion with him. His expertise in machine learning had influential impact on my research. I would also specially thank Dr. Luo for his continuously support and discussion on the domain knowledge in my research work. Without his domain insight, my research will not be as meaningful. Finally I would thank Dr. Esfahanian for always being supportive for my thesis. It was great pleasure to work with each

of them and I look forward to continue my collaborations in my future career.

Further, I would like to spread my acknowledgement to many current and previous peer colleagues, especially, Prakash Mandayam Comar, Zubin Abraham, Lei Liu, Shuai Yuan, Courtland VanDam, Xi Liu, Ding Wang, Kaixiang Lin and Qi Wang from CSE Department, as well as Pouyan Hatami from Geography Department. They are not only keeping my lab hours interesting and informative via discussions on different research topics, but also being helpful to make my life happier and easier at MSU. I also want to express my spacial thanks to the graduate director Eric Torng for his extensive help and support over my Ph.D time, as well as numerous department secretaries for all administrative and travel assistance.

Last but not least, I gratefully thank my parents and my wife for their continuous support, encouragement, and unconditional love to me. This thesis would not be accomplished without their support. I also thank my daughter coming into my life in the last year of my Ph.D, who has brought me tremendous happiness as being a father.

Particularly, this thesis is partially supported by National Science Foundation through grant NSF III-1615612, NOAA Climate Program office through grant NA12OAR4310081, and NASA Terrestrial Hydrology Program through grant NNX13AI44G.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

Multi-task learning (MTL) is a data mining and machine learning approach for learning multiple related tasks jointly, by incorporating the relatedness between tasks [20]. It was first introduced in the context of neural networks inspired from models of learning in the human brain. To date, MTL has been successfully applied to many applications, including computer vision [119, 103], healthcare and medical applications [126, 14, 15], recommender systems [79, 101], natural language processing [4], genomics [81] and social media applications [121]. MTL is also applicable to various learning tasks such as classification [113], regression [126], and clustering [52].

The rationale for applying MTL to multiple predictive modeling problems [113] are as follows: (i) the prediction tasks are not identical, so fitting a single model for all tasks using their combined data is often not as effective; and (ii) the prediction tasks are related, so the model for each task could benefit by sharing information across the tasks. However, identifying what type of information to be shared between the different tasks and how to integrate them into a unified learning formulation are among the key challenges that must be addressed. Based on different assumptions of particular applications, numerous methods for representing and modeling the common task structures have been developed in recent years [107, 38, 7, 56, 63, 25, 49, 126, 120, 87, 31].

## 1.1 Geospatio-temporal Data

Geospatio-temporal data sets are prevalent across many disciplines, including geophysical and environmental sciences [40, 8], medical informatics [32], computational molecular and fluid dynamics [115], and real estate economics [92]. The availability of such data sets have grown in recent years due to advances in sensing technology and large-scale scientific simulation studies. Predictive modeling of such massive data sets is important as it supports various scientific and engineering applications.

A geospatio-temporal data set consists of time series observations of various fields associat-

ed with each geo-referenced object, e.g., measurements of meteorological variables at a weather station, frequency of infectious disease incidents in a county, or average housing price, annual income, and crime rate in a given neighborhood. Figure 1.1 shows an example of a geospatio-temporal data set, where each location is associated with a pair of time series (e.g., temperature and precipitation). A standard prediction problem for this type of data is to forecast the values of the time series at each location based on their historical observations as well as the spatial relationships among the locations. In addition to temporal forecasting, one might also be interested in inferring the values at locations with limited or no historical data.

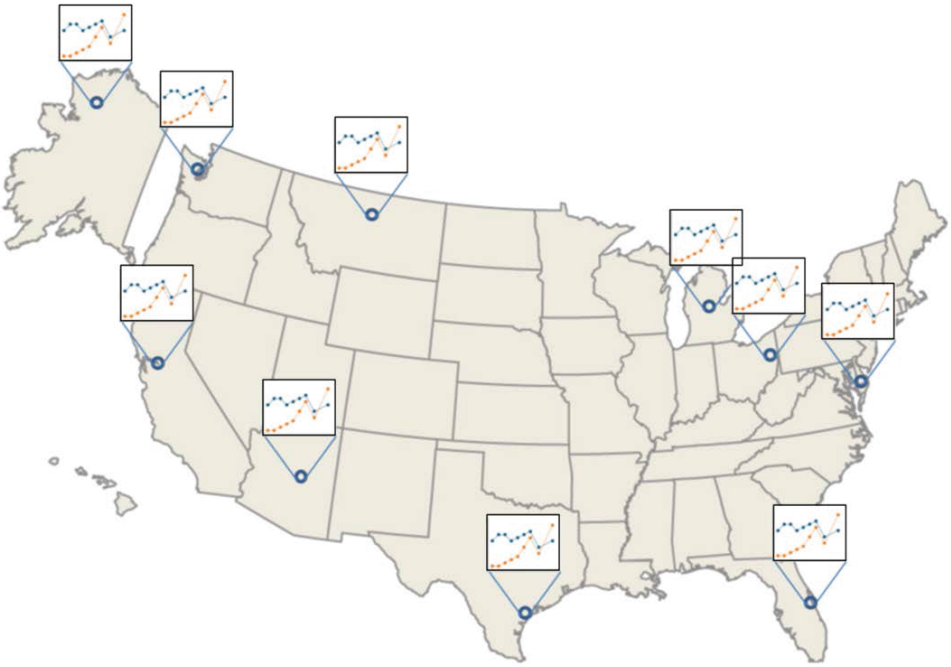


Figure 1.1: A geospatio-temporal data set, where each location has a pair of time series.

The geospatio-temporal data considered in this study (as depicted in Figure 1.1) is different from the another type of geospatio-temporal data known as trajectory data [43, 74, 95, 68]. An example of the latter is shown in Figure 1.2, which shows the trajectory of a hurricane. Unlike Figure 1.1, where the data are generated by stationary geo-referenced objects, trajectory data are generated from movements of the geo-referenced objects. In terms of the prediction task for trajectory data, one is often interested to predict the locations in the future path of the moving object.

[1]Image courtesy of `http://www.weather.gov/images/okx/Sandy/Track_NHCreport`.

## 1.2 Application of MTL to Geospatio-Temporal Data

There are numerous applications involving geospatio-temporal data that can be naturally mapped into MTL problems. In the following, we describe several examples of such applications from the climate and environmental science domains.

- **Ensemble Forecasting:**

  Ensemble forecasting is a well-known numerical prediction technique for modeling the evolution of nonlinear dynamic systems. The ensemble member forecasts are generated from multiple runs of a computer model, where each run is obtained by perturbing the starting condition or using a different model representation of the dynamic system. The set of forecasts produced by the ensemble members are used to quantify the range of uncertainties in the future state of the system. However, for many applications, it is also desirable to obtain a point estimate of the forecasts for decision making purposes. To do so, the ensemble mean

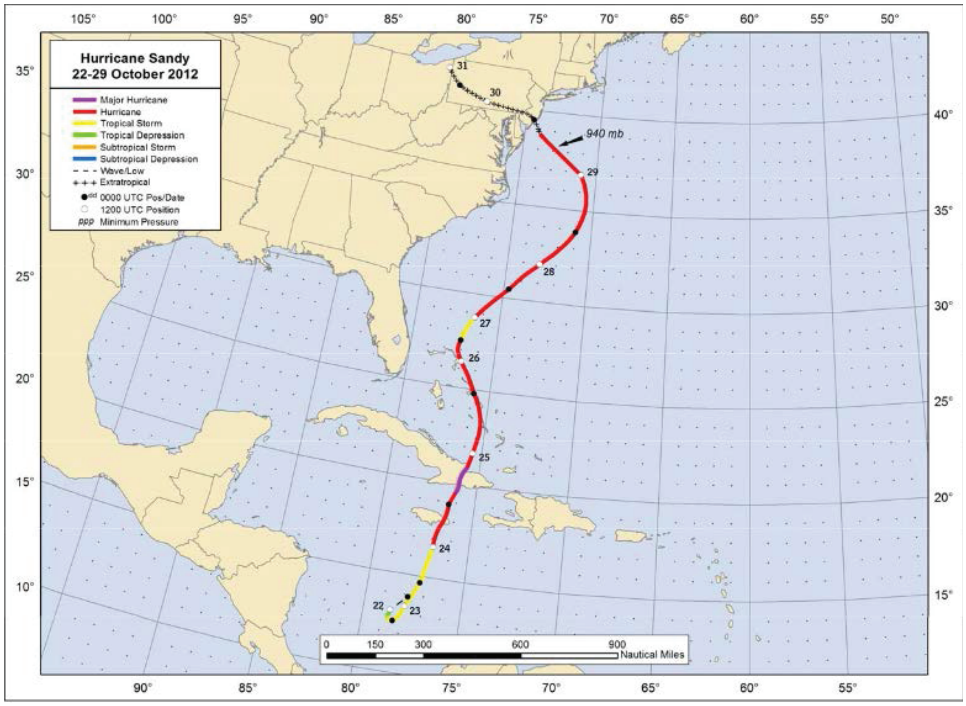



Figure 1.2: Example of trajectory data for hurricane Sandy in 2012[1].

[1] png

or median are typically chosen as the consensus estimate of the aggregated forecasts. These approaches are limited in that they assume each member is equally skillful and ignore the potential correlations between the ensemble member forecasts. Furthermore, each forecast typically represents the predictions for an extended future time period, e.g., for a 6-month forecast duration. The ensemble mean or median approaches do not guarantee the temporal predictions would be smooth throughout the forecast window since the mean or median aggregation is performed at each time step independently. This presents an opportunity for developing a temporal MTL framework to learn the optimal weights for combining the ensemble member forecasts, where the prediction at each time step is considered a separate learning task and task relatedness constraints can be imposed to ensure temporal smoothness of the predictions.

- **Multi-Location Prediction:**

  Another obvious application of MTL is to predict the values of one or more response variables simultaneously at multiple locations. For example, climate scientists are interested to obtain projections of the future climate (e.g., temperature and precipitation) at multiple locations in a study region. Similarly, in hydrology, soil moisture forecasts are needed for various river basins for drought monitoring applications. One way to do this is to train a local model at each location using only its historical observations. This single-task learning (STL) approach may not be effective especially if there are limited training examples available at the locations. This is because the STL models fail to exploit the spatial relationships of the predictions, which is important for many geospatio-temporal prediction problems following Tobler's first law of geography, which states that "Everything is related to everything else, but near things are more related than distant things." [98]. Spatio-temporal MTL approaches can be developed that allow the local models at multiple locations to be jointly trained, taking into account the shared information between locations (e.g., their spatial proximity or autocorrelation).

- **Multi-Scale Prediction:**

  The geospatio-temporal data generated for many scientific applications are often available at multiple spatial scales. For example, the climate projections generated by global and regional climate models are available at a wide range of spatial resolutions, from tens to several hundred kilometers. As another example, the soil moisture estimates provided by the Climate Forecast System Reanalysis (CFSR) and the North American Land Data Assimilation System (NLDAS-2) for hydrology applications are available at 35km and 12km resolutions respectively over the conterminous United States. It has been shown that integrating data from multiple scales for predictive modeling could improve the performance, as different analysis scales have the potential to produce different prediction skills [75]. Many existing methods [82, 35, 75] would concatenate the variables from multiple scales into a single feature vector, enabling us to apply existing approaches for building predictive models from the multi-scale data. However, this strategy may not be effective as it fails to consider the relationships among the multi-scale variables. A MTL framework can be developed to exploit such relationships in order to build more robust and effective predictive models. For example, the predictive modeling using predictor variables at each scale for each location can be treated as a separate learning task and the task relatedness can be defined based on the structure of the models (e.g., the shared latent factors of the predictor variables across multiple scales).

## 1.3 Research Challenges

This section presents the challenges of applying MTL to geospatio-temporal data.

### 1.3.1 Growth and Other Characteristics of Data

Geospatio-temporal data sets often grow linearly over time. For example, climate observations from weather stations are typically available at hourly, 3-hourly, or daily basis. Due to the continuous growth of the data, building predictive models for a large number of locations simultaneously

using MTL in a batch mode is computationally demanding as the models have to be trained from scratch each time new observation data become available. An online or incremental MTL approach is more suitable as the models need to be revised using the new observations only instead of re-trained using the entire data set. Furthermore, such an approach is more adaptive to changes due to concept drifts in the data. Although online versions of MTL have been developed in recent years [21, 34, 114, 87, 101], these methods are not specifically designed to exploit properties of the geospatio-temporal data.

In particular, the characteristics of the data could make it difficult to apply existing MTL methods to certain applications. For example, the ensemble forecasting task described in the previous section can be viewed as a sliding window prediction problem, which requires making predictions for multiple consecutive time steps into the future. As time progresses, the sliding window will be shifted from its current forecast window to the next forecast window. In addition, the number of observed values in its previous forecast windows varies depending on the current time period, making it difficult to apply existing online MTL methods. To illustrate this, consider an ensemble of weather forecasting models. Every day, the ensemble members would generate their predictions for the next 7 days. For example, on a Sunday, the ensemble members would generate their predictions for a 7-day forecast window starting from the next day (Monday) to the following Sunday. On the next day, which is a Monday, the ensemble members would generate a new set of forecasts for a new 7-day time window (from Tuesday to the following Monday). At this time, the previous forecast window generated on Sunday would have 1 observed value out of its 7 predicted days whereas the forecast window generated on the previous Saturday would have 2 observed values instead of 1. The online MTL approach will have to revise its models using the partially observed (incomplete) data from not just 1 but several of its previously forecasted time windows. This problem will be further elaborated in Chapter 3.

### 1.3.2 Spatio-temporal Autocorrelation

Geospatio-temporal data often contain spatial and temporal autocorrelations that can be utilized to improve the performance of the predictive models. For example, temporal autocorrelation can be used to ensure that the predicted values vary smoothly over time. The spatial autocorrelation enables the models to leverage data from other locations based on the similarity between the locations. How to effectively incorporate both the spatial and temporal autocorrelations into the MTL framework is a challenge that needs to be addressed.

### 1.3.3 Capturing Broader-Scale Phenomena

The geospatio-temporal variabilities of the data are potentially influenced by broader-scale phenomena, whose impact may vary from one location to another. For example, the well-known El-Ninö phenomenon has different effects on weather patterns in North America and the rest of the world. To generate realistic predictions, the predictive models for different locations should be able to capture such broader-scale phenomena. Designing a geospatio-temporal MTL framework that can effectively perform multi-location predictions while capturing such broader-scale phenomena is a challenge in this research.

### 1.3.4 Multi-Scale Data

As previously noted, the geospatio-temporal data are often available at multiple spatial resolutions. Such multi-scale data provide useful information that can be harnessed for building robust prediction models. Constructing a MTL framework that can utilize the multi-scale data, while accounting for the relationships between variables at different scales, is another challenge that needs to be addressed.

## 1.4 Thesis Contributions

This section summarizes the key contributions of this thesis in terms of addressing the challenges described in the previous section.

### 1.4.1 ORION

In Chapter 3, an online temporal MTL framework called ORION (which stands for **O**nline **R**egularized mult**I**-task regressi**ON**) is proposed to estimate the weights of the ensemble members for the ensemble forecasting application. The framework uses an *online learning with restart* strategy to deal with the partially observed data for different forecast windows. It regards the prediction at each time point as a single learning task and employs graph regularization constraints to ensure smoothness in the model parameters while taking into account the task relatedness within each time window. The framework is applicable to different types of loss functions, including $\varepsilon$-insensitive and quantile loss. The quantile loss function is particularly appealing for forecasting extreme events in a time series. Experimental results suggest that ORION reduces the forecast error of ensemble median for all major river basins datasets, and performs better than other baseline algorithms in most cases.

### 1.4.2 GSpartan

In order to address the multi-location prediction problem, Chapter 4 proposes a novel MTL framework called GSpartan, which stands for GeoSPAtio-tempoRal mulTi-tAsk learNing. GSpartan considers the prediction at each location as a single learning task and employs a multi-modal MTL approach to fit the training data simultaneously at multiple locations, where each modality corresponds to a latent factor derived from the data. The multi-modal approach is based on the assumption that the prediction models for all locations share a common set of low-rank base models. The local model at each location is then composed of a linear combination of these base models. The coefficients of the linear combination along with the base models are jointly estimated from the data using a matrix factorization approach. A Laplacian regularization is also added to the MTL formulation to constrain the relatedness between different task models under the assumption of spatial autocorrelation. To ensure interpretability of the models, sparsity and non-negativity constraints are also added into the proposed MTL formulation. The performance of GSpartan is evaluated against several baseline methods on climate data collected from 37 randomly chosen

weather stations in Canada. The experimental results show that GSpartan outperforms single-task learning and other existing MTL methods in more than 75% of the stations.

### 1.4.3 WISDOM

GSpartan considers only the spatial autocorrelation between locations, while ignoring the temporal consistency of its predictions. In addition, it is designed for batch learning, which makes it difficult to scale up the algorithm for the growing geospatio-temporal data sets collected by many applications. To overcome these limitations, an incremental, geospatio-temporal MTL framework named WISDOM (**W**eighted **I**ncremental **S**patio-temporal Multi-task Learning via Tensor **D**ec**om**position) is proposed in Chapter 5. The framework performs a supervised tensor factorization on the geospatio-temporal data, from which it constructs distinct spatial and temporal predictive models from the derived latent factors. The framework can also incorporate known patterns from the domain by adding constraints on the latent factors. WISDOM can be applied to incremental learning over space, time, or both when new observation data become available. The effectiveness of the proposed framework for multi-location time series prediction is demonstrated on a large-scale global climate data.

### 1.4.4 MUSCAT

Finally, Chapter 6 presents a multi-scale MTL framework named MUSCAT (Multi-scale Spatio-Temporal Muti-task Learning via Incremental Multi-tensor Decomposition), which not only learns the prediction models at multiple locations simultaneously, but can also take advantage of the task relatedness between predictor variables at different scales. Unlike WISDOM , MUSCAT represents the data at each scale as a separate tensor and performs a supervised, joint multiple tensor decomposition across all tensors. The consistency of the learned latent spatial and temporal factors are enforced to model the relatedness of the predictor variables from different scales. A comparison between MUSCAT to other baseline methods including WISDOM demonstrates the

effectiveness of the proposed framework in terms of handling the multi-scale geospatio-temporal data.

## 1.5 Related Publications

This thesis is based on materials taken from several conference or journal papers that have been published or submitted as this thesis is written. The materials for Chapter 3 are adopted from two papers, entitled "ORION: Online Regularized multI-task regressiON and its application to ensemble forecasting" [107] and "Online Multi-task Learning Framework for Ensemble Forecasting" [111], respectively. Chapter 4 is based on the results published in the paper entitled "GSpartan: a Geospatio-Temporal Multi-task Learning Framework for Multi-location Prediction" [109]. Chapter 5 is adopted from the paper "WISDOM: Weighted Incremental Spatio-Temporal Multi-Task Learning via Tensor Decomposition" [110]. This paper also won the best paper award at the IEEE BigData Conference in 2017. Finally, the materials in Chapter 6 are based on a paper that has been submitted to a conference. I am the first author and main contributor for all of these papers.

## 1.6 Thesis Outline

The reminder of this thesis is organized as follows. Chapter 2 presents the background and related works of this thesis. Chapter 3 presents the temporal MTL framework named ORION for ensemble forecasting [107, 111]. Chapter 4 introduces the spatio-temporal MTL framework for multi-location prediction (GSpartan) [109]. Chapter 5 presents an incremental, spatio-temporal MTL framework named WISDOM [110] while Chapter 6 describes the extension of the framework to multi-scale geospatio-temporal data. Finally, Chapter 7 summarizes the dissertation and discusses possible extensions and future directions of this research.

# CHAPTER 2

# BACKGROUND

The objective of this thesis is to develop MTL frameworks for the predictive modeling of geospatio-temporal data. In this chapter, previous works related to the key topics investigated in this thesis, including spatio-temporal data mining, multi-task learning, multi-scale modeling, and tensor decomposition, are reviewed.

## 2.1 Spatio-temporal Data Mining

Spatio-temporal data mining can be either related to stationary geo-referenced objects which consist of time series for each object [110, 48, 46, 47], such as soil moisture measurement as weather stations over time, or related to in-motion objects which are associated with movements of the object over time [9, 27, 45], such as trajectory data. This thesis focuses on the methods belonging to the first type of the spatio-temporal data mining.

Spatio-temporal data mining is important to many application domains, such as climatology, medicine, and crop sciences. It usually can be categorized into three classes: spatial data mining, temporal data mining, and spatio-temporal data mining. Spatial data mining is to learn, model, and discover the interesting unknown knowledge or patterns with spatial dependencies such as spatial auto-correlation, while temporal data mining is to model or discover the temporal unknown knowledge or patterns from timeseries data by exploring the temporal dependencies such as temporal auto-correlation. Spatio-temporal data mining usually consumes both spatial and temporal information in the modeling process. There are other information or properties that are often considered in the modeling process, such as spatial heterogeneity, non-stationarity, as well as multiple scale effect. Existing tasks for spatial, temporal, spatio-temporal data mining include prediction (classification and regression) [58, 17, 83], clustering [51, 18, 53], association rule mining [62, 73, 96] and anomaly detection [71, 59, 70]. This thesis will focus on the prediction tasks in spatial, and spatio-temporal data mining.

Spatial prediction modeling is to predict the target variable at unobserved locations by modeling the data in observed locations. One of the common approach for modeling spatial data is spatial statistics [5, 44], which is composed by three sets of methods based on different types of spatial data: (a). geostatistics for point referenced data, such as Kriging [10]; (b). lattice statistics for areal data, such as spatial autoregressive model (SAR) [58], conditional autoregressive model (CAR) [33] and Markov random fields (MRF) [54]; and (c). point process for spatial point patterns, such as complete spatial randomness (CSR) [44].

Spatio-temporal prediction refers to the methods that perform predictions on the response variable by learning models from predictor variables from spatio-temporal data [88]. Spatio-temporal prediction common approaches include spatio-temporal autoregressive regression [42, 99], spatio-temporal kriging [123, 89], and Bayesian hierarchical spatio-temporal models [104, 85, 13]. A comprehensive survey on spatio-temporal data mining is provided by Shekhar et. al. [88]. Recent work has shown that MTL could potentially gain better performance on spatio-temporal prediction, by taking into account the correlation between tasks across time, space, or time and space [107, 109, 110, 121, 117].

## 2.2 Multi-scale Data Mining

The concept of multi-scale learning has been used in the literature to describe different classes of methods. For example, in traditional machine learning, multi-scale learning is related to the use of techniques based on multi-kernel [11], multi-covariance matrices [100] or multi-basis functions [80]. These methods are not designed to handle data containing multi-scale variables. Instead, they are focused on extracting and combining different models of the same raw data, which is available at a single resolution.

For geospatio-temporal data, the goal of multi-scale learning is to build predictive models from data whose predictor variables are collected at multiple resolutions. A straightforward approach for handling multi-scale data is to concatenate the predictor variables together into one single feature vector and then apply existing prediction methods [35, 75]. These approaches are limited in that it

ignores the relationships among the multi-scale variables.

With the explosive growing of deep learning techniques, there have been several recent works on multi-scale modeling, in the context of computer vision. The concept of multi-scale data in this case typically refers to different image resolutions. For example, the multi-scale concept used in deep convolutional networks refers to the extraction of features from an image using windows or patches of different sizes [41, 12]. Deep learning can also be used to learn multi-scale spatial features from images that have multiple resolutions [122]. However, these methods assume that the spatial data has a gridded structure, and thus, are inapplicable to geospatio-temporal data where the spatial data may not be gridded. Nevertheless, there have been some recent works using deep learning to model data with multiple resolutions. For example, a multi-scale deep learning approach was proposed in [78] to build models and to perform predictions independently at each scale of the data, followed by majority voting to combine the predictions. Another approach performs global learning on coarser scale data and then refines the model locally on finer scale data [37]. However, unlike our proposed approach, none of these methods consider the relationships between data at different scales.

## 2.3 Multi-task learning

MTL is a well-known machine learning approach for solving multiple, related prediction tasks simultaneously by considering the shared information among the tasks [20]. The rationale of MTL is that the information propagated between related tasks may enhance the overall predictive accuracy if the models are trained jointly instead of independently. The MTL scheme has been successfully adopted into various learning problems including classification [113], regression [126], and clustering [52]. Online learning versions of MTL have also been developed [21, 34, 114, 87, 101], which allows the models of multiple tasks to be updated incrementally. In this section, the existing MTL approaches are reviewed.

Based on the information sharing scheme between multiple tasks, MTL methods can be broadly classified into the following categories:

**MTL based on Low-rank Representation**: This category of methods assume that the underlying models for different tasks share a low-rank representation, either additively or multiplicatively. For example, in the multiplicative low-rank representation, the tasks models can be factorized into two terms, one representing the base models shared across tasks, the other representing the combination coefficients. The task models are linear combinations of those base models, weighted by the combination coefficients. An illustration for multiplicative low-rank representation is shown in Figure 2.1. Additive low-rank representation simply replaces the multiplication into summation, with additional constraints for those additive terms.

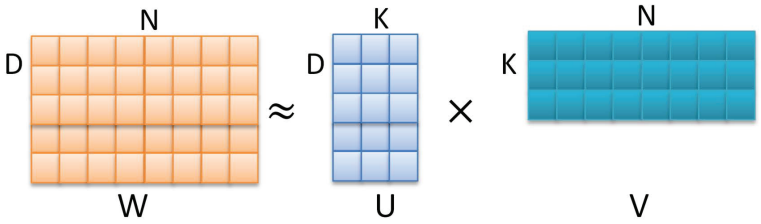



Figure 2.1: Multiplicative low-rank representation

Previous works that belong to this category include [23, 7, 56, 63, 24, 25, 49]. For example, Chen et al. [23, 24] assumed the tasks models are composed additively by a low-rank structure capturing the relatedness over the tasks and a group sparse structure that identifies the outlying tasks. A similar idea was proposed in Gong et al. [49], which uses a combination of a task model that enforces group sparsity on features and another that enforces group sparsity for detecting outlying tasks. Argyriou et al. [7] and Kang et al. [56] projected the features into a low dimensional space in which the predictive models for different tasks are inferred and the discriminant features are selected using an $\ell_{2,1}$ norm. Kumar et al. [63] assumed that each task model is a linear combination of a set of underlying base models, where the coefficients of the linear combination along with the base models are coupled in a multiplicative way. Sparsity constraints are also enforced to ensure the models are interpretable. Chen et al. [25] incorporated both additive and multiplicative representation into their formulation with a low-dimensional feature map shared across the different tasks. Tensor decomposition methods are also adopted in MTL to extract low-rank representations [118]. These existing MTL methods may not be suitable to leverage some of the task relations specially

in geospatio-temporal predictions, or fails to capture geospatio-temporal patterns from the data.

**MTL based on Explicit Task Relationship**: This category of methods explicitly incorporates the task relationship inferred from the application domain into the MTL formulation. The task relationships are typically encoded in a similarity matrix, whose elements represent the closeness between two tasks [39]. Concrete constraints will be enforced on objective functions based on the relations defined by the similarity matrix, in such a way that the models will be similar if the tasks are close to each other. Figure 2.2 illustrates the general idea of using explicit task relationships in MTL. Matrix $A$ represents the similarity matrix of tasks, where $A_{ij}$ is the similarity score or closeness between task $i$ and task $j$.

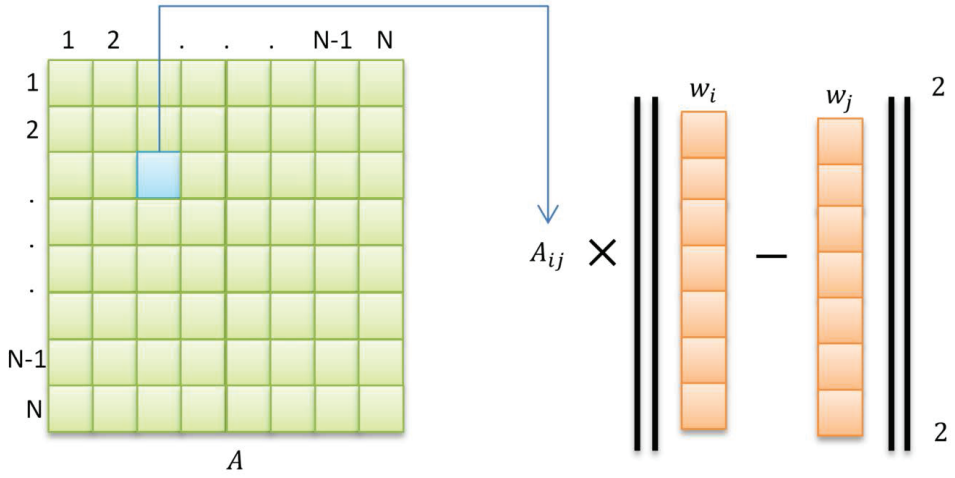



Figure 2.2: Illustration of MTL based on explicit task relationship

Several works can be categorized under this framework. For example, Zhou et al. [126] employed a task relationship to ensure smoothness in time series prediction, where each time point corresponds to a separate prediction task. Zhang et al. [120] proposed a MTL regularization formulation to simultaneously learn the task relationship and task models. Saha et al. [87] also proposed an online MTL to learn both the task models and their relationships. This category of methods is useful when the task relatedness can be clearly defined. They are inapplicable if the task relationship is not available or if the implicit task relationship needs to be inferred from data.

**MTL based on Sharing Common Parameters**: The assumption in this category is that different

tasks are related by sharing a set of common parameters. One of the examples is proposed by Evgeniou et al. [38], which is shown in Figure 2.3. In this example, the models $\mathbf{w}_1$ to $\mathbf{w}_n$ for task 1 to task $n$ are composed by a common term $\mathbf{w}_0$, and $n$ task specific terms $\mathbf{v}_1$ to $\mathbf{v}_n$, where the common term $\mathbf{w}_0$ is shared across different tasks.

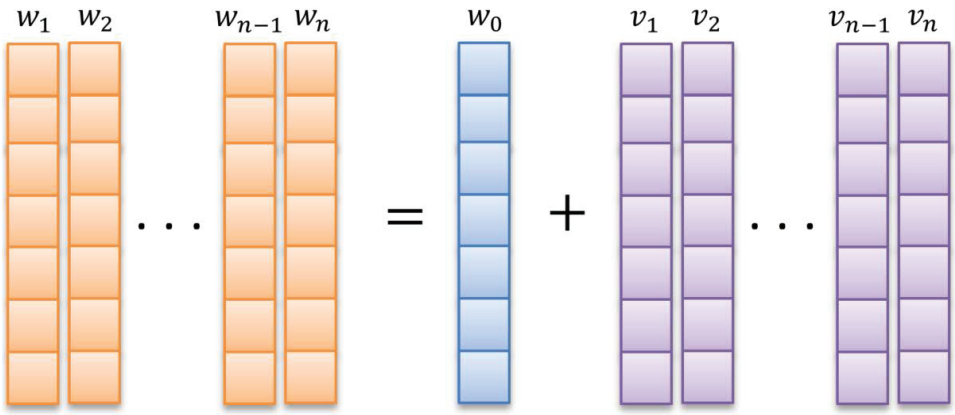



Figure 2.3: Illustration of MTL based on sharing common parameters

There are other ways for sharing model parameters for multi-task learning. For example, Lawrence et al. [64] and Yu et al. [116] proposed a multi-task Gaussian process where the prior parameters are shared across different generative processes. Lee et al. [65] gave a formulation to share hyperparameters between distributions of different task models while Daume et al. [31] learned hierarchical Bayesian models that share a common structure parameterized by a covariance matrix. Note that sharing common terms additively across tasks is equivalent to using a specific task relation where all the tasks are close to the (weighted) averaged models [38].

**MTL based on Hybrid Information Sharing Methods**: More recently, there have been some efforts to combine two or more MTL information sharing methods described above. For example, Xu et al. [108] combined both low-rank representation and explicit task relations as it models the heterogeneity between patients using low-rank modeling and constrains the task models to be close to the weighted average models from the neighboring tasks.

Besides of the taxonomy induced by the way of sharing information between tasks, MTL algorithms can be also divided as batch [38, 126, 120, 87, 56, 63, 24, 25, 49, 116, 31, 65, 20] versus

online multi-task learning [34, 21, 87, 3], according to the data availability and the way of training the models.

**Batch Multi-task Learning Methods**: For batch multi-task learning approaches, all the data points from all the tasks have to be available when the learning procedure starts. Similar to single task learning approaches, the advantage of batch learning methods is that all the information can be taken account at the same time and hence the learnt models can be considered as an optimal solution. Most of the aforementioned MTLs are batch methods.

**Online Multi-task Learning Methods**: Online learning is a family of learning algorithms, in which the data is observed in a sequential manner and the models must be updated incrementally based on the new observations. The merit of applying online learning is that it does not consume as much memory compared to batch learning, which is advantageous for handling large, streaming data sets. Some of the popular online learning methods include the weighted majority algorithm [69], online Passive Aggressive (PA) algorithm [29], and confidence-weighted algorithm [36]. Since the amount of data to be processed in multi-task learning is generally much larger than single-task learning, a straightforward solution is to extend MTL to an online learning setting. Dekel et al. [34] presented an approach for multi-task perceptron using a global loss function to define the relationships among the different tasks. This method assumes the true values for all the learning tasks are available when updating the model. Relationships among the tasks are also not explicitly defined in their objective function. As a result, only those tasks that have committed a prediction error will have their weights updated. Cavallanti et al. [21] proposed another perceptron-based online MTL method, which uses an empirical loss function with co-regularization among the tasks. Since the method performs an update one task at a time, the results are order-dependent. Saha et al. [87] extended the work in [21] by proposing a method that simultaneously learns the model and task relation matrix using an alternating optimization scheme. However, this approach has the same limitation as [21] because it uses similar assumptions as [21]. [67] proposed a collaborative online MTL method by extending [38] into an online version. However, it does not consider the possible task relationships explicitly. Another recent work on online multi-task learning consid-

ers a regret-minimization approach under expert advice model [3]. However, none of the existing works were designed for geospatio-temporal data to address its challenges.

## 2.4 Tensor Decomposition

Existing tensor decomposition approaches can be categorized into unsupervised [57, 22, 28] and supervised methods [105, 86, 106, 118]. The former is designed to minimize reconstruction error whereas the latter considers the relationship between the predictor and response variables, and thus, is more suitable for predictive modeling problems.

Several implementations of supervised tensor decomposition approaches have been developed in recent years. For example, Wu et. al [106] proposed the SNTFM framework to map the representation of each sample from a tensor into a vector and build a predictive model on the new representation of the data. Bernardino et. al [86] and Kishan et. al [105] presented a MTL framework for data sets with multi-modal structures using a supervised tensor decomposition approach. Similarly, Yu et. al [118] proposed a low-rank tensor learning approach for multivariate spatio-temporal data. These approaches encode the parameters of their predictive models as a tensor, which is assumed to have a low rank. Tensor decomposition was performed on the model parameters, unlike our proposed framework, which performs the decomposition on the spatio-temporal tensor data. This strategy allows us to provide meaningful interpretation of the latent factors in terms of their spatial, temporal, and feature dimensions.

Incremental/online tensor decomposition methods have been developed for streaming data as well as for data sets that grow dynamically over time [93]. Current incremental methods can be divided into two categories, one is based on Tucker decomposition, while the other is based on CP decomposition. Existing incremental Tucker decomposition methods are mostly based on incremental SVD, applied to the matricization of the tensor for different modes [94, 93, 55]. The incremental SVD based methods assume orthogonality of the latent factors, which is somewhat restrictive for interpretability reasons. Zhou, et al. [127] developed an online CP decomposition approach, where the the latent factors are updated when there are new data. However, these meth-

ods are inapplicable to our problem setting since they were developed for unsupervised learning. Although there is a recent work on online supervised tensor decomposition [118], it is restricted to new observation data along the time dimension, whereas our WISDOM and MUSCAT framework consider new observations in both space and time.

# CHAPTER 3

# ONLINE REGULARIZED MULTI-TASK REGRESSION

[1] Ensemble forecasting is a popular numerical prediction method for modeling nonlinear dynamic systems, such as climate [97], agriculture [19], ecological [6], and hydrological [72] systems. Specifically, the future states of the systems are predicted using computer models that simulate the physical processes governing the behavior of such systems. Since the models may not fully capture all the underlying processes as well as their parameterization accurately, their forecast errors tend to amplify with increasing lead time. Ensemble forecasting [66] aims at obtaining more robust prediction results by combining outputs from multiple runs of the computer models. Each run is obtained by perturbing the starting condition or using a different model representation of the dynamic system. The forecast generated from each run corresponds to a series of predictions for a future time window, $T$, known as the forecast duration. As an example, consider the ensemble forecasting task shown in Fig. 3.1. There are altogether $N$ sets of forecasts generated every 5 days, from April 5, 2011 to September 12, 2011. Each set of forecasts contains time series predictions generated by $d$ ensemble members ($\mathbf{x}_1$, ..., $\mathbf{x}_d$) for a forecast duration $T$. The goal is to combine the $d$ ensemble member forecasts in such a way that produces an aggregated prediction that is consistent with the true observation data, $\mathbf{y}$.

The ensemble mean or median is often used as a point estimate of the aggregated forecasts. These estimates assume that every ensemble member prediction is equally plausible, and thus, their predictions should be weighted equally. However, as some ensemble members may fit the observed data less accurately than others, this may degrade the overall predictive performance. To illustrate this, consider the example given in Fig. 3.2, which shows the basin-averaged soil moisture percentile forecasts of a hydrological model ensemble, consisting of 33 members (shown as thin green lines), along with the ensemble median (shown as a dashed line) and the true observed data (shown as a solid red line). The ensemble members were initially calibrated using soil moisture

[1] This chapter is based on the previous publication [111].

data from September 2, 2011. Their outputs for that day are therefore the same. Each ensemble member would then generate a set of forecasts for a 40-day time window, from September 7, 2011 to October 12, 2011. Though the forecasts were quite similar at the beginning, they began to diverge with increasing lead time. Some ensemble member forecasts no longer fit the observed data well, thus affecting the accuracy of the ensemble median approach. This example illustrates the need to learn an optimal set of weights for combining the ensemble member forecasts.

To address this need, this chapter presents an online learning model that can update the weights of the ensemble members according to their predictive skills. Unlike conventional online learning, the ensemble forecasting task requires making multiple predictions for a time window of length $T$. As the predictions within the window are not independent due to the temporal autocorrelation of the time series, the ensemble forecasting task can be naturally cast as an *online multi-task regression problem*.

Another difference between conventional online learning and the requirements of ensemble forecasting is that not all observation data are available when the model is updated. For example, in Fig. 3.1, suppose the ensemble members generate a new set of forecasts every 5 days. Let forecast $N-2$ be the set of forecasts generated on September 2 for a 40-day period from September 7 to

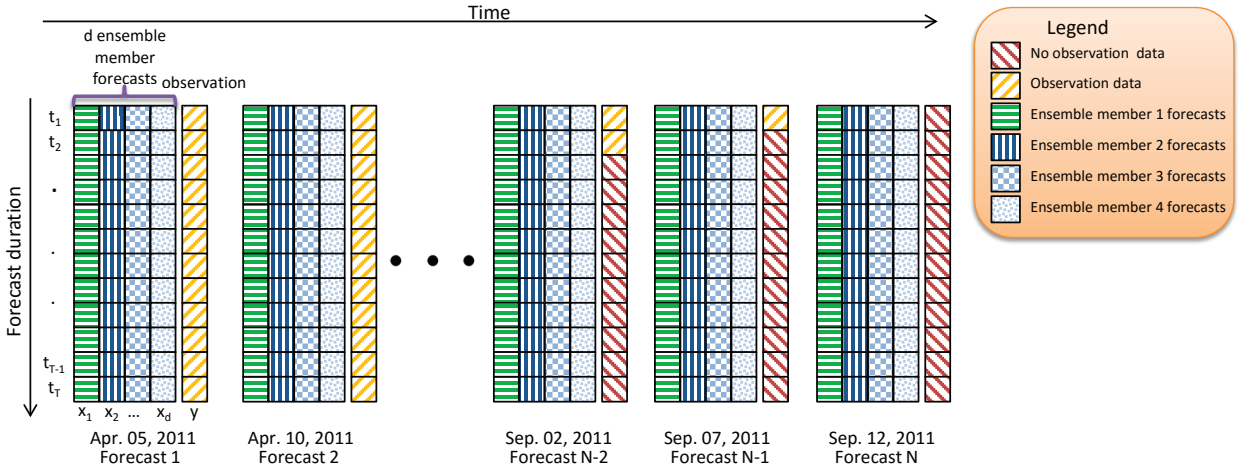



Figure 3.1: A schematic illustration of ensemble forecasting task (diagram is best viewed in color). Assuming the latest forecast was generated on September 12, 2011 for the time period between September 17 and October 22, there is one observation value available to verify Forecast $(N-1)$, two observations available to verify Forecast $(N-2)$, and so on.

October 12, forecast $N-1$ be the corresponding forecast set generated on September 7 for the time window September 12 to October 17, and forecast $N$ be the forecast set generated on September 12 for September 17 to October 22. We assume forecast $N$ to be the most current forecast. When the online learning model is updated on September 12, forecast $N-2$ has two observed values in its time window, including September 7 and September 12, while forecast $N-1$ has a new observed value for September 12. This means the observation data are not only incomplete in each time window, the number of observations also varies from one window to another. We call this problem *online multi-task learning with partially observed data*. Due to this property of the data, instead of updating the model from its most recent model, we need to backtrack and revise some of the older models when new observation data are available.

This chapter is to develop a framework called ORION (which stands for **O**nline **R**egularized mult**I**-task regressi**ON**) that uses an *online learning with restart* strategy to deal with the partially observed data. The framework also employs graph regularization constraints to ensure smoothness

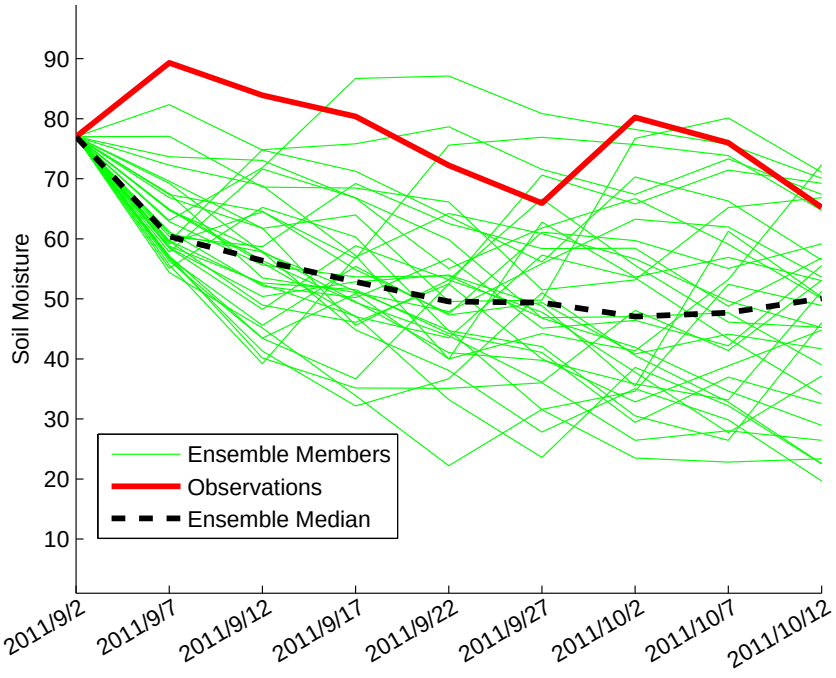



Figure 3.2: Seasonable soil moisture forecasts and the observed time series at a major river basin in North America.

in the model parameters while taking into account the temporal autocorrelation of the predictions within each time window. It is capable to incorporate different loss functions, such as $\varepsilon$-insensitive loss and quantile loss function, which is useful for predicting extreme values of a time series. Theoretical proofs are also provided to demonstrate the convergence of the proposed online learning algorithm.

The main contributions of this chapter are summarized as follows:

- The problem of online regularized multi-task regression with partially observed data is introduced and demonstrated the relevance to the ensemble forecasting task.
- A novel framework called ORION is presented, which uses an online learning with restart strategy to solve the problem. It also uses a graph Laplacian to capture relationships among the learning tasks along with a passive aggressive update scheme to optimize the $\varepsilon$-insensitive loss function.
- The framework is extended to incorporate the quantile loss function for predicting extreme events. To the best of our knowledge, ORION is the first multi-task regression framework that has been tailored for extreme value prediction.
- Extensive experiments are performed using a real-world soil moisture data set and the results show that ORION outperforms several baseline algorithms, including the ensemble median, for the majority of the river basins in our data set.

## 3.1 Problem Formulation

We consider a variation of the online multi-task learning process described in [34], in which the learning proceeds in a sequence of rounds. At the start of round $n$, where $n \in \{1,2,\cdots,N\}$, the algorithm observes $T$ unlabeled instances, $\mathbf{x}^{(n)} = \{\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, \cdots, \mathbf{x}_T^{(n)}\}$, where each instance $\mathbf{x}_i^{(n)} \in \Re^d$ is a $d$-dimensional vector of predictor variables, and $T$ is the number of instances to be predicted in each round. The algorithm then predicts the target value $f(\mathbf{x}_i)$ for each unlabeled instance. We consider the prediction of each instance as a separate learning task. Our goal is to

learn a set of prediction functions for the $T$ tasks such that their cumulative loss over the $N$ rounds is minimized. Similar to previous works [34, 21], we consider only linear prediction functions of the form $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$, where $\mathbf{w} \in \Re^d$ is the parameter vector. Extending the formulation to non-linear models is beyond the scope of this work.

From an ensemble forecasting perspective, each online round corresponds to the scenario when a new set of forecasts is generated, as shown in Fig. 3.1. After $N$ rounds, there are $N$ sets of forecasts generated. Each set of forecasts is generated by $d$ ensemble members, which collectively form the set of predictor variables for our online learning task. Note that each ensemble member produces a time series of length $T$, which is equivalent to the number of instances (tasks) that must be predicted in each online round. Since the prediction tasks are related due to the inherent temporal autocorrelation of the time series, we cast the ensemble forecasting task into a multi-task learning problem. The number of prediction tasks in each round is equal to the forecast duration, $T$, and the number of online rounds is equal to the number of forecast sets, $N$.

As can be seen from Fig. 3.1, the number of observed values available to update the predictions at the end of each round varies from one forecast set to another. Let forecast $N$ be the most recent set of forecasts generated for the time window $[t_{N+1}, t_{N+2}, \cdots, t_{N+T}]$. There are no observed values available for the given time window. However, the forecast set $N-1$, which was previously generated for the time window $[t_N, t_{N+1}, \cdots, t_{N+T-1}]$ now has a new observed value for the time $t_N$. Similarly, the number of observed values for the forecast set $N-2$ increases from 1 to 2. More generally, let $\mathbf{y}_m^{(n)} = \{y_1^{(n)}, y_2^{(n)}, \cdots, y_{m_n}^{(n)}\}$ denote the set of observed values available for the forecast set $m$ in round $n$, where $m \leq n$. If $T$ is the forecast duration, then the number of observed values available to update the forecast set $m$ in round $n$ is given by $m_n = \min(n-m, T)$. This partially observed data scenario distinguishes our work from other online multi-task learning formulations.

**Example 1.** *Consider the scenario shown in Fig. 3.1. Assume the most recent forecast set N was generated on September 12, 2011. The forecast set* $N-1$*, which was generated on September 7, 2011 for predicting a 40-day time window from September 12 to October 17, will have a new observed value for September 12. The forecast set* $N-2$*, which was generated on September 2,*

*2011 for predicting the time series from September 7 to October 12, now has two observed values, for September 7 and 12, respectively. If the forecast duration is T, then all previous forecast sets from Forecast 1 to Forecast $N-T$ will have complete observation values for their entire data sets.*

Let $f^{(n-1)}$ be the model obtained at the end of round $n-1$. After round $n$, a new labeled data $\mathscr{D}_{n-1}^{(n)} = (\mathbf{x}^{(n-1)}, \mathbf{y}_{n-1}^{(n)})$. is available to update $f^{(n-1)}$, where $\mathbf{y}_{n-1}^{(n)}$ includes the latest observed value for the time $t_n$. The number of labeled examples in $\mathscr{D}^{(n-2)}$, $\mathscr{D}^{(n-3)}, \cdots$ $\mathscr{D}^{(n-T+1)}$ also increases by 1 in round $n$ since all of them involves a prediction for the time step $t_n$. It is therefore insufficient to generate $f^{(n)}$ directly from $f^{(n-1)}$ since the models $f^{(n-1)}$, $f^{(n-2)}, \cdots f^{(n-T)}$ should also be revised given their expanded labeled data sets. Thus, we employ the following *online learning with restart strategy* to update our models. In round $n$, we first backtrack to round $n-T$ and revise $f^{(n-T)}$ with the expanded labeled data $\mathscr{D}^{(n-T)}$ to obtain a new model $f^{(n-T+1)}$. We then update $f^{(n-T+1)}$ with the expanded labeled data $\mathscr{D}^{(n-T+1)}$ to obtain $f^{(n-T+2)}$ and repeat the procedure until $f^{(n)}$ is obtained. To implement this strategy, the algorithm only needs to maintain two sets of weights, $\mathbf{w}^{(n-1)}$ and $\mathbf{w}^{(n-T-1)}$. At the start of round $n$, the algorithm makes it prediction using $\mathbf{w}^{(n-1)}$. When $\mathscr{D}^{(n)}$ is available, the algorithm will backtrack and progressively update the models starting with $\mathbf{w}^{(n-T-1)}$, which was the last set of weights trained from a completely labeled data set, until $\mathbf{w}^{(n)}$ is obtained.

## 3.2 Online Regularized multI-task regressiON (ORION)

This section presents the ORION framework for the $\varepsilon$-insensitive loss function. An extension of the framework to the quantile loss function is given in Section 3.3.

### 3.2.1 ORION for $\varepsilon$-insensitive Loss Function

Although our framework requires restarting the online learning process at round $n-T$ to deal with the partially observed data problem, the update formula and optimization step in each round are identical. Specifically, in round $n$, the ORION framework assumes that the weights are co-

regularized as follows:

$$\mathbf{w}_t^{(n)} = \mathbf{w}_0^{(n)} + \mathbf{v}_t^{(n)}, \quad \forall t \in \{1,2,\cdots,T\}.$$

In other words, the prediction functions for all $T$ tasks share a common term $\mathbf{w}_0$ and a task-specific weight $\mathbf{v}_t$, which is expected to be small when the predictions are correlated. To estimate the weights, we employ the following objective function, which extends the Passive Aggressive online algorithm given in [29] to a multi-task learning setting with an $\varepsilon$-insensitive loss function:

$$\begin{aligned} \arg\min_{\mathbf{w}_0,\{\mathbf{v}_t\}} \quad & \frac{1}{2}\sum_{t=2}^{T} ||\mathbf{w}_t - \mathbf{w}_{t-1}||_2^2 + \frac{\mu}{2}\sum_{t=1}^{T} ||\mathbf{v}_t||_2^2 \\ + \quad & \frac{\lambda}{2}||\mathbf{w}_0 - \mathbf{w}_0^{(n-1)}||_2^2 + \frac{\beta}{2}\sum_{t=1}^{T} ||\mathbf{v}_t - \mathbf{v}_t^{(n-1)}||_2^2 \\ s.t. \quad & \forall t \le m_n : \ |\mathbf{w}_t^T \mathbf{x}_t^{(n)} - y_t^{(n)}| \le \varepsilon \\ & \forall t \in \{1,2,\cdots,T\} : \ \mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t \\ & \mu, \lambda, \beta \text{ and } \varepsilon \ge 0 \end{aligned} \tag{3.1}$$

where $m_n$ is the number of labeled observations, $\mathbf{x}_t^{(n)}$ is a vector of predictor variables for task $t$ in the $n$-th round, and $y_t^{(n)}$ is the target value. For brevity, we have omitted the superscript $n$ in our notations for $\mathbf{v}_t$, $\mathbf{w}_t$, and $\mathbf{w}_0$. Since $\mathbf{w}_t - \mathbf{w}_{t-1} = \mathbf{v}_t - \mathbf{v}_{t-1}$, Equation (3.1) can be simplified as follows:

$$\begin{aligned} \arg\min_{\mathbf{w}_0,\mathbf{V}} \quad & \frac{1}{2}\mathrm{Tr}\left[\mathbf{V}^T(\mathbf{L} + \mu\mathbf{I}_T)\mathbf{V})\right] \\ + \quad & \frac{\lambda}{2}||\mathbf{w}_0 - \mathbf{w}_0^{(n-1)}||_2^2 + \frac{\beta}{2}||\mathbf{V} - \mathbf{V}^{(n-1)}||_F^2 \\ s.t. \quad & \forall t \le m_n, \ |\mathbf{w}_t^T \mathbf{x}_t^{(n)} - y_t^{(n)}| \le \varepsilon \\ & \forall t \in \{1,2,\cdots,T\}, \ \mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t \end{aligned} \tag{3.2}$$

where $\mathbf{V} = [\mathbf{v}_1^T; \mathbf{v}_2^T; \cdots; \mathbf{v}_T^T]$ is a $T \times d$-dimensional matrix, $\mathbf{I}_T$ is a $T \times T$ identity matrix, Tr$[\cdot]$ is the trace operator and

$$\mathbf{L}_{i,j} = \begin{cases} 1 & \text{if } i = j = 1 \text{ or } i = j = T \\ 2 & \text{if } i = j \neq 1 \text{ and } i = j \neq T \\ -1 & \text{if } i = j+1 \text{ or } i = j-1 \\ 0 & \text{otherwise} \end{cases}$$

is a graph Laplacian capturing the relationships among the $T$ tasks. The Lagrange formulation is given by

$$\begin{aligned} \mathscr{L}(\mathbf{w}_0, \mathbf{V}, \boldsymbol{\tau}) &= \frac{1}{2}\text{Tr}(\mathbf{V}^T(\mathbf{L} + \mu\mathbf{I}_T)\mathbf{V}) \\ &+ \frac{\lambda}{2}||\mathbf{w}_0 - \mathbf{w}_0^{(n-1)}||_2^2 + \frac{\beta}{2}||\mathbf{V} - \mathbf{V}^{(n-1)}||_F^2 \\ &+ \sum_{t \in \mathscr{O}^{(n)}} \tau_t(|\mathbf{w}_t^T \mathbf{x}_t^{(n)} - y_t^{(n)}| - \varepsilon) \end{aligned}$$

where $\mathscr{O}^{(n)} = \{t | t \leq m_n \text{ and } |\mathbf{w}_t^T \mathbf{x}_t^{(n)} - y_t^{(n)}| > \varepsilon\}$ is the feasible set and $\boldsymbol{\tau} = \{\tau_t\}$ is the set of Lagrangian multipliers such that $\tau_t \geq 0$ for all $t \in \mathscr{O}^{(n)}$ and $\tau_t = 0$ for all $t \notin \mathscr{O}^{(n)}$. In the next subsection, we present the solution for this optimization problem.

### 3.2.2 Optimization

To simplify the notation, we first vectorize the matrix $\mathbf{V}$ and concatenate it with $\mathbf{w}_0$. Let $\mathbf{z} = [\mathbf{w}_0; \mathbf{v}_1; ..., \mathbf{v}_T]$ denote the resulting weight vector to be solved. The Lagrangian can now be written into the following form:

$$\begin{aligned} \mathscr{L}(\mathbf{z}, \boldsymbol{\tau}) &= \frac{1}{2}(\mathbf{z} - \mathbf{z}^{(n-1)})^T \mathbf{R}(\mathbf{z} - \mathbf{z}^{(n-1)}) + \frac{1}{2}\mathbf{z}^T \mathbf{Q}\mathbf{z} \\ &+ \left[(\mathbf{z}^T \tilde{\mathbf{X}}^{(n)} - \mathbf{y}^{(n)^T})\mathbf{S} - \varepsilon \mathbf{1}^T\right]\mathbf{P}\boldsymbol{\tau} \end{aligned} \tag{3.3}$$

where $\tilde{\mathbf{X}}^{(n)}$, $\mathbf{R}$, $\mathbf{Q}$, $\mathbf{P}$, and $\mathbf{S}$ are defined in Table 3.1.

Taking the partial derivative of $\mathscr{L}$ with respect to $\mathbf{z}$ and setting it to zero yields the following

$$\begin{aligned} \frac{\partial \mathscr{L}(\mathbf{z}, \boldsymbol{\tau})}{\partial \mathbf{z}} &= \mathbf{R}(\mathbf{z} - \mathbf{z}^{(n-1)}) + \mathbf{Q}\mathbf{z} + \tilde{\mathbf{X}}^{(n)}\mathbf{S}\mathbf{P}\boldsymbol{\tau} = 0 \\ \mathbf{z} &= \mathbf{M}(\mathbf{R}\mathbf{z}^{(n-1)} - \tilde{\mathbf{X}}^{(n)}\mathbf{S}\mathbf{P}\boldsymbol{\tau}) \end{aligned} \tag{3.4}$$

Table 3.1: Notations used in Equation (3.3)

| Notation | Definition |
|---|---|
| $\mathbf{0}_d$ | a $d$-dimensional column vector of zeros |
| $\mathbf{0}_{d\times T}$ | a $d \times T$ matrix of zeros |
| $\mathbf{I}_d$ | a $d \times d$ identity matrix |
| $\mathbf{A} \otimes \mathbf{B}$ | Kronecker product between matrices $\mathbf{A}$ and $\mathbf{B}$ |
| $\tilde{\mathbf{X}}^{(n)}$ | $\begin{bmatrix} \mathbf{x}_1^{(n)} & \mathbf{x}_2^{(n)} & \cdots & \mathbf{x}_T^{(n)} \\ \mathbf{x}_1^{(n)} & \mathbf{0}_d & \cdots & \mathbf{0}_d \\ \mathbf{0}_d & \mathbf{x}_2^{(n)} & \cdots & \mathbf{0}_d \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_d & \mathbf{0}_d & \cdots & \mathbf{x}_T^{(n)} \end{bmatrix}$ |
| $\mathbf{y}^{(n)}$ | $[y_1^{(n)}; y_2^{(n)}; \cdots; y_{m_n}^{(n)}; \mathbf{0}_{(T-m_n)}]$ |
| $\mathbf{P}$ | $\mathbf{P}_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ and } i \in \mathscr{O}^{(n)} \\ 0 & \text{otherwise} \end{cases}$ |
| $\mathbf{S}$ | $\mathbf{S}_{i,j} = \begin{cases} \text{sign}(\mathbf{w}_i^{(n)^T} \mathbf{x}_i^{(n)} - y_i^{(n)}) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$ |
| $\boldsymbol{\tau}$ | $[\tau_1; ...; \tau_T]$ |
| $\mathbf{R}$ | $\begin{bmatrix} \lambda \mathbf{I}_d & \mathbf{0}_{d\times Td} \\ \mathbf{0}_{Td\times d} & \beta \mathbf{I}_{Td} \end{bmatrix}$ |
| $\mathbf{Q}$ | $\begin{bmatrix} \mathbf{0}_{d\times d} & \mathbf{0}_{d\times Td} \\ \mathbf{0}_{Td\times d} & (\mathbf{L} + \mu \mathbf{I}_T) \otimes \mathbf{I}_d \end{bmatrix}$ |

where $\mathbf{M} = (\mathbf{R} + \mathbf{Q})^{-1}$. It can be easily shown that $\mathbf{R} + \mathbf{Q}$ is a positive definite matrix, which means it is invertible and its inverse is also positive definite.

Plugging $\mathbf{z}$ in Equation (3.4) back into Equation (3.3) leads to the following equation after simplification

$$\begin{aligned} \mathscr{L}(\boldsymbol{\tau})\tilde{\mathbf{X}}^{(n)}\mathbf{S}\mathbf{P}\boldsymbol{\tau} &= -\frac{1}{2}\boldsymbol{\tau}^T \tilde{\mathbf{X}}_{PS}^{(n)T} \mathbf{M}^T \tilde{\mathbf{X}}_{PS}^{(n)} \boldsymbol{\tau} + \ell_n^T(\hat{\mathbf{z}}^{(n-1)})\boldsymbol{\tau} \\ &+ \text{constant} \end{aligned} \tag{3.5}$$

where

$$\begin{aligned} \tilde{\mathbf{X}}_{PS}^{(n)} &= \tilde{\mathbf{X}}^{(n)}\mathbf{S}\mathbf{P} \\ \ell_n^T(\hat{\mathbf{z}}^{(n-1)}) &= \left[(\hat{\mathbf{z}}^{(n-1)T}\tilde{\mathbf{X}}^{(n)} - \mathbf{y}^{(n)T})\mathbf{S} - \varepsilon \mathbf{1}^T\right]\mathbf{P} \\ \hat{\mathbf{z}}^{(n-1)} &= \mathbf{M}^T \mathbf{R}^T \mathbf{z}^{(n-1)} \end{aligned} \tag{3.6}$$

Note that $\mathbf{P}$ is a diagonal matrix, whose diagonal element $\mathbf{P}_{t,t}$ is zero if $t \notin \mathscr{O}^{(n)}$. In other words, if the target value for task $t$ is either unavailable or predicted correctly (within the $\varepsilon$-insensitive bound), all the elements in the $t$-th column of $\tilde{\mathbf{X}}_{PS}^{(n)}$ become 0, and the corresponding $t$-th element in $\ell_n^T(\hat{\mathbf{z}}^{(n-1)})$ is also 0. Thus, $\tau_t$ for $t \notin \mathscr{O}^{(n)}$ has no impact on Equation (3.5) and can be set to zero. In the following derivation, we assume the rows and columns corresponding to all the tasks $t \notin \mathscr{O}^{(n)}$ in $\tau$, $\tilde{\mathbf{X}}_{PS}^{(n)}$, and $\ell_n^T(\hat{\mathbf{z}}^{(n-1)})$ have been removed.

Taking the partial derivative of the "reduced" Lagrangian with respect to $\boldsymbol{\tau}$ and setting it to zero yields

$$\begin{aligned} \frac{\partial \mathscr{L}}{\partial \boldsymbol{\tau}} &= -\tilde{\mathbf{X}}_{PS}^{(n)T}\mathbf{M}^T\tilde{\mathbf{X}}_{PS}^{(n)}\boldsymbol{\tau} + \ell_n(\hat{\mathbf{z}}^{(n-1)}) = 0 \\ \boldsymbol{\tau} &= \left[\tilde{\mathbf{X}}_{PS}^{(n)T}\mathbf{M}^T\tilde{\mathbf{X}}_{PS}^{(n)}\right]^{-1}\ell_n(\hat{\mathbf{z}}^{(n-1)}) \end{aligned} \tag{3.7}$$

There are several points worth noting regarding the update formula for $\mathbf{z}$ and its learning rate $\boldsymbol{\tau}$. First, note that Equation (3.7) is only applicable to tasks that belong to $\mathscr{O}^{(n)}$. The columns in $\tilde{\mathbf{X}}_{PS}$ for $t \notin \mathscr{O}^{(n)}$ must be removed before calculating $\tau$. Otherwise, the matrix $\tilde{\mathbf{X}}_{PS}^{(n)T}\mathbf{M}^T\tilde{\mathbf{X}}_{PS}^{(n)}$ is not invertible. For $t \notin \mathscr{O}^{(n)}$, we set $\tau_t = 0$ before calculating $\mathbf{z}$. Second, even when $\tau_t = 0$, the corresponding weight for $\mathbf{v}_t$ may still change due to the first term of Equation (3.4). This distinguishes our approach from other online algorithms, where a zero learning rate implies the weights will not change in the next round. Finally, our formula for $\boldsymbol{\tau}$ has a similar form as the learning rate for the single-task learning given in [29], $\tau_n = \ell_n/||\mathbf{x}_n||^2$. The main difference is that the $\tau$ for multi-task learning must take into account the task relatedness in both $\ell_n$ and the inverse of $\tilde{\mathbf{X}}_{PS}^{(n)T}\mathbf{M}^T\tilde{\mathbf{X}}_{PS}^{(n)}$.

### 3.2.3 Algorithm

A summary of the ORION framework for $\varepsilon$-insensitive loss function is given in Algorithm 1. The algorithm proceeds in a sequence of rounds. During round $n$, the algorithm receives the instances $\mathbf{x}_t^{(n)}$ for each task $t \in \{1,..,T\}$. Using the online learning with restart strategy, it will backtrack to round $n-T$ and update the set of labeled observations to include the most recent target value.

After computing the loss for each task, it identifies the set of tasks for which the loss exceeds the $\varepsilon$-bound. The weights associated with the tasks will be updated using the formula given in Equation (3.4). Note that $\tau_t$ is set to zero for tasks that do not belong to $\mathscr{O}^{(n)}$. In each round, the algorithm only needs to maintain two sets of weights, $\mathbf{z}^{(n-T)}$ and $\mathbf{z}^{(n)}$, along with the predictor variables $\{\mathbf{x}^{(n-T)}, \mathbf{x}^{(n-T+1)}, \cdots, \mathbf{x}^{(n)}\}$ and the observed target values $\{y^{(n-T)}, y^{(n-T+1)}, y^{(n)}\}$. Its storage complexity is therefore $T$ times the complexity of single-task learning.

#### 3.2.3.1 Time complexity

In this section, we analyze the time complexity of the ORION framework for the $\varepsilon$-insensitive loss function in terms of the number of online rounds $N$, the number of tasks $T$ and the number of features $d$. For ensemble forecasting, $T$ refers to the forecast duration and $d$ is the number of ensemble members (see Fig. 3.1). Each round requires calculations of Equations (3.4) and (3.7). For Equation (3.7), $\tilde{\mathbf{X}}_{PS}^{(n)}$ needs to be computed first, which requires $O(T^3 d)$ floating point operations (flops). Calculating $\tilde{\mathbf{X}}_{PS}^{(n)T}\mathbf{M}^T\tilde{\mathbf{X}}_{PS}^{(n)}$ requires $O(T^3 d^2)$ flops and its inverse will take $O(T^3)$ flops. According to Equation (3.6), calculating both $\hat{\mathbf{z}}^{(n-1)}$ and $\ell_n(\hat{\mathbf{z}}^{(n-1)})$ will require $O(T^3 d)$ flops. The time complexity for calculating Equation (3.7) is $O(T^3 d^2)$ whereas the time complexity for Equation (3.4) is $O(T^3 d + T^2 d^2)$. Therefore, the model update for each task requires $O(T^3 d^2)$ flops. Since there are $T$ tasks, the time complexity for each online round is $O(T^4 d^2)$. There are other computations that need to be performed only once throughout the entire online learning process, which is the calculation for $\mathbf{M} = (\mathbf{R} + \mathbf{Q})^{-1}$, whose complexity is $O(T^3 d^3)$. Thus, after $N$ rounds, the overall time complexity is $O(N(T^4 d^2 + T^3 d^3))$, which is linear in the number of rounds (similar to other online learning algorithms). The number of tasks $T$ and number of features $d$ are domain-dependent, though they are both generally much smaller than $N$.

### 3.2.4 Theoretical Analysis of ORION-$\varepsilon$

This section presents theoretical analysis on the average loss bound of the ORION-$\varepsilon$ algorithm.

**Input**: $\mu$, $\lambda$, $\beta$, $\varepsilon = 0.001$ ;
**Initialize**: $\mathbf{w}_0 = \mathbf{0}_d; \forall t \in \{1,...,T\}, \mathbf{v}_t = \mathbf{0}_d$ ;
Compute $\mathbf{R}$ and $\mathbf{Q}$ using the formula in Table 3.1 ;
**for** $n = 2, \cdots, N$ **do**
  Receive $\mathbf{x}_1^{(n)}, \mathbf{x}_2^{(n)}, ..., \mathbf{x}_T^{(n)}$ ;
  **for** $m = n - T, \cdots, n$ **do**
    Set $m_n = n - m$ ;
    **for** $t = 1, 2, \cdots, T$ **do**
      Predict $\hat{y}_t^{(m)} = \left[\mathbf{w}_0^{(m-1)} + \mathbf{v}_t^{(m-1)}\right]^T \mathbf{x}_t^{(m)}$ ;
    **end**
    Update $\mathbf{y}_m^{(n)} = \mathbf{y}_m^{(n-1)} \cup \{y^{(n)}\}$ ;
    Set $\mathscr{O}^n = \{t | t \leq m_n; |\mathbf{w}_t^{(m)T} \mathbf{x}_t^{(m)} - y_{m,t}^{(n)}| > \varepsilon\}$ ;
    Compute $\boldsymbol{\tau}$ using Equation (3.7) and set $\tau_t = 0$ when $t \notin \mathscr{O}^{(n)}$ ;
    Update $\mathbf{z}^{(m)}$ using Equation (3.4) ;
  **end**
**end**

**Algorithm 1:** Pseudocode for ORION-$\varepsilon$ Algorithm

**Lemma 1.** *Let* $\mathbf{U}\blacksquare\mathbf{U}^T$ *be an eigendecomposition of the real symmetric matrix* $\mathbf{R}^{-1}\mathbf{Q}$*, where* $\mathbf{U}$ *is an orthogonal matrix and* $\blacksquare$ *is a diagonal matrix containing the eigenvalues of* $\mathbf{R}^{-1}\mathbf{Q}$*. The eigendecomposition of* $\mathbf{MR}$ *is given by* $\mathbf{U}(\mathbf{I}+\blacksquare)^{-1}\mathbf{U}^T$.

*Proof.* First, we can write

$$\mathbf{MR} = (\mathbf{R}+\mathbf{Q})^{-1}\mathbf{R} = (\mathbf{I}+\mathbf{R}^{-1}\mathbf{Q})^{-1} = (\mathbf{I}+\mathbf{U}\blacksquare\mathbf{U}^T)^{-1}$$

Since $\mathbf{U}$ is an orthogonal matrix, $\mathbf{U}\mathbf{U}^T = \mathbf{I}$. Hence

$$\mathbf{MR} = (\mathbf{U}\mathbf{U}^T + \mathbf{U}\blacksquare\mathbf{U}^T)^{-1} = \mathbf{U}(\mathbf{I}+\blacksquare)^{-1}\mathbf{U}^T$$

□

**Lemma 2.** $\|\mathbf{MR}\|_2 = 1$*, where* $\|\cdot\|_2$ *denote the induced 2-norm of a matrix.*

*Proof.* The induced 2-norm of a matrix $\mathbf{A}$ is defined as

$$\|\mathbf{A}\|_2 = \max_{\|\mathbf{x}\|_2=1} \|\mathbf{A}\mathbf{x}\|_2 = \sqrt{\lambda_{\max}},$$

where $\lambda_{\max}$ is the largest eigenvalue of the matrix $\mathbf{A}^T\mathbf{A}$. Since

$$\mathbf{R}^{-1}\mathbf{Q} \;=\; \begin{bmatrix} \mathbf{0}_{d\times d} & \mathbf{0}_{d\times Td} \\ \mathbf{0}_{Td\times d} & \frac{1}{\beta}(\mathbf{L}+\mu\mathbf{I}_T)\otimes\mathbf{I}_d \end{bmatrix}$$

the determinant $|\mathbf{R}^{-1}\mathbf{Q}| = 0$ because the matrix contains rows and columns of all zeros. In addition, it can be shown that $\mathbf{R}^{-1}\mathbf{Q}$ is a positive semi-definite matrix since it is diagonally dominant, which means all of its eigenvalues must be non-negative. Since $|\mathbf{R}^{-1}\mathbf{Q}| = \prod_k \lambda_k = 0$, this implies that the smallest eigenvalue of $\mathbf{R}^{-1}\mathbf{Q}$ is $\lambda_{\min} = 0$.

Following Lemma 1, the largest eigenvalue of $\mathbf{MR}$ is $(1+\lambda_{\min})^{-1} = 1$. Finally, given that $(\mathbf{MR})^T\mathbf{MR} = \mathbf{U}(\mathbf{I}+\blacksquare)^{-2}\mathbf{U}$, the largest eigenvalue of $(\mathbf{MR})^T\mathbf{MR}$ must also be equal to 1. Thus, $\|\mathbf{MR}\|_2 = 1$. □

**Lemma 3.** $\|\mathbf{I}-\mathbf{MR}\|_2 = 1 - \frac{1}{1+\lambda_{\max}} \leq 1$, *where* $\|\cdot\|_2$ *denote the induced 2-norm of a matrix and* $\lambda_{\max} \geq 0$ *is the largest eigenvalue of* $\mathbf{R}^{-1}\mathbf{Q}$.

*Proof.* Following Lemma 1, it is easy to see that $\mathbf{I}-\mathbf{MR} = \mathbf{U}[\mathbf{I}-(\mathbf{I}+\blacksquare)^{-1}]\mathbf{U}^T$. Thus, the largest eigenvalue of $\mathbf{I}-\mathbf{MR}$ is $1-\frac{1}{1+\lambda_{\max}}$, which is the induced 2-norm of the matrix. □

**Theorem 1.** *Let* $\mathbf{z}^{(1)}, \mathbf{z}^{(2)}, \cdots, \mathbf{z}^{(n)}$ *be a sequence of weights learned using the ORION-*$\varepsilon$ *algorithm. Using the notations given in Table 3.1, the following bound holds for any* $\mathbf{u} \in \Re^{(T+1)d}$.

$$\begin{aligned} & \frac{1}{N}\sum_n^N ||\ell_n(\mathbf{M}^T\mathbf{MRz}^{(n)})|| \\ \leq\; & \frac{1}{N}\sum_n^N \|\ell_n^T(\mathbf{M}^T\mathbf{u})\| + \frac{1}{2C}\left[\frac{\|\mathbf{u}\|^2}{N} + \|\mathbf{u}\|\Psi + C^2\rho^2\right], \end{aligned}$$

*where* $||\boldsymbol{\tau}|| \leq C$, $\|\mathbf{z}^{(n)}\| \leq \Psi$, *and* $\|\mathbf{M}\tilde{\mathbf{X}}^{(n)}\mathbf{SP}\| \leq \rho$.

*Proof.* Define $\Delta_n = \|\mathbf{z}^{(n)}-\mathbf{u}\|^2 - \|\mathbf{z}^{(n+1)}-\mathbf{u}\|^2$. We will derive the relative loss bound by finding

the upper and lower bound of $\sum_{n=1}^{N}\Delta_n$. For the upper bound,

$$\begin{aligned}\sum_{n=1}^{N}\Delta_n &= \sum_{n=1}^{N}\|\mathbf{z}^{(n)}-\mathbf{u}\|^2-\|\mathbf{z}^{(n+1)}-\mathbf{u}\|^2 \\ &= \|\mathbf{z}^{(1)}-\mathbf{u}\|^2-\|\mathbf{z}^{(N+1)}-\mathbf{u}\|^2 \\ &= \|\mathbf{u}\|^2-\|\mathbf{z}^{(N+1)}-\mathbf{u}\|^2 \\ &\leq \|\mathbf{u}\|^2\end{aligned} \tag{3.8}$$

where $\mathbf{z}^{(1)}=\mathbf{0}$. Next, we derive the lower bound of $\Delta_n$.

$$\begin{aligned}\Delta_n &= \|\mathbf{z}^{(n)}-\mathbf{u}\|^2-\|\mathbf{z}^{(n+1)}-\mathbf{u}\|^2 \\ &= \|\mathbf{z}^{(n)}-\mathbf{u}\|^2-\|\mathbf{MRz}^{(n)}-\mathbf{M}\tilde{\mathbf{X}}^{(n)}\mathbf{SP}\boldsymbol{\tau}-\mathbf{u}\|^2 \\ &= \|\mathbf{z}^{(n)}-\mathbf{u}\|^2-\|\mathbf{MRz}^{(n)}-\mathbf{u}\|^2-\|\mathbf{M}\tilde{\mathbf{X}}^{(n)}\mathbf{SP}\boldsymbol{\tau}\|^2+\ 2(\mathbf{MRz}^{(n)}-\mathbf{u})^T\mathbf{M}\tilde{\mathbf{X}}^{(n)}\mathbf{SP}\boldsymbol{\tau}\end{aligned}$$

A lower bound on the first two terms is given as follows

$$\begin{aligned}&\|\mathbf{z}^{(n)}-\mathbf{u}\|^2-\|\mathbf{MRz}^{(n)}-\mathbf{u}\|^2 \\ =\ &\|\mathbf{z}^{(n)}\|^2-\|\mathbf{MRz}^{(n)}\|^2-2\mathbf{u}^T(\mathbf{I}-\mathbf{MR})\mathbf{z} \\ \geq\ &\|\mathbf{z}^{(n)}\|^2-\|\mathbf{MR}\|^2\|\mathbf{z}^{(n)}\|^2-2\|\mathbf{u}\|\|\mathbf{I}-\mathbf{MR}\|\|\mathbf{z}^{(n)}\| \\ \geq\ &-2\|\mathbf{u}\|\|\mathbf{z}^{(n)}\|\end{aligned}$$

where we have applied Lemmas 2 and 3 and used the fact that $\|\mathbf{Ax}\|_2\leq\|\mathbf{A}\|_2\|\mathbf{x}\|_2$ and $\mathbf{u}^T\mathbf{v}\leq\|\mathbf{u}\|\|\mathbf{v}\|$. Furthermore,

$$\begin{aligned}&\mathbf{z}^{(n)^T}\mathbf{R}^T\mathbf{M}^T\mathbf{M}\tilde{\mathbf{X}}^{(n)}\mathbf{SP}\boldsymbol{\tau}-\mathbf{u}^T\mathbf{M}\tilde{\mathbf{X}}^{(n)}\mathbf{SP} \\ =\ &[(\mathbf{z}^{(n)^T}\mathbf{R}^T\mathbf{M}^T\mathbf{M}\tilde{\mathbf{X}}^{(n)}-\mathbf{y}^{(n)^T})\mathbf{S}-\varepsilon\mathbf{1}^T]\mathbf{P}\boldsymbol{\tau}-[(\mathbf{u}^T\mathbf{M}\tilde{\mathbf{X}}^{(n)}-\mathbf{y}^{(n)^T})\mathbf{S}-\varepsilon\mathbf{1}^T]\mathbf{P}\boldsymbol{\tau} \\ \geq\ &\ell_n^T(\mathbf{M}^T\mathbf{MRz}^{(n)})\boldsymbol{\tau}-\ell_n^T(\mathbf{M}^T\mathbf{u})\boldsymbol{\tau}\end{aligned}$$

Putting them together, we have

$$\begin{aligned}
\|\mathbf{u}\|^2 &\geq \sum_{n=1}^{N} \Delta_n \\
&\geq -2\|\mathbf{u}\| \sum_{n}^{N} \|\mathbf{z}^{(n)}\| - \sum_{n}^{N} \|\mathbf{M}\tilde{\mathbf{X}}^{(n)}\mathbf{SP}\|^2 \|\boldsymbol{\tau}\|^2 \\
&\quad +2\sum_{n}^{N} \ell_n^T(\mathbf{M}^T\mathbf{M}\mathbf{R}\mathbf{z}^{(n)})\boldsymbol{\tau} - 2\sum_{n}^{N} \ell_n^T(\mathbf{M}^T\mathbf{u})\boldsymbol{\tau}
\end{aligned}$$

Assuming $||\boldsymbol{\tau}|| \leq C$, $\|\mathbf{z}^{(n)}\| \leq \Psi$, $\|\mathbf{M}\tilde{\mathbf{X}}^{(n)}\mathbf{SP}\| \leq \rho$, and after re-arranging the equation, we obtain

$$\begin{aligned}
&\frac{1}{N}\sum_{n}^{N} ||\ell_n(\mathbf{M}^T\mathbf{M}\mathbf{R}\mathbf{z}^{(n)})|| \\
\leq\ &\frac{1}{N}\sum_{n}^{N} \|\ell_n^T(\mathbf{M}^T\mathbf{u})\| + \frac{1}{2C}\left[\frac{\|\mathbf{u}\|^2}{N} + \|\mathbf{u}\|\Psi + C^2\rho^2\right]
\end{aligned}$$

□

## 3.3 Online Regularized Multi-Task Quantile Regression (ORION-QR)

Predicting extreme value events are important for applications such as weather and hydrological forecasting due to their adverse impacts on both human and natural systems. Unfortunately, most of the existing work on multi-task learning have considered only squared or hinge loss functions, and thus, are not suitable for extreme value prediction. In this section, we describe an extension of the ORION framework to predict extreme values in a time series by incorporating a quantile loss function. To describe the approach, we first present the quantile regression (QR) method [60] and introduce the quantile loss function.

QR is a statistical method for estimating the conditional quantiles of a target variable as a function of its predictor variables. By focusing on the upper or lower quantiles of the distribution, this may help bias the algorithm towards learning the extreme values of the target distribution. Specifically, QR is designed to improve the estimate of the $\tau^{th}$ conditional quantile of the prediction by minimizing the following sum of asymmetrically weighted absolute residuals:

$$\sum_{i=1}^{n} \rho_\tau(y_i - \mathbf{x}_i^T\beta), \text{ where } \rho_\tau(u) = \begin{cases} \tau u & u > 0 \\ (\tau - 1)u & u \leq 0 \end{cases}$$

The $\tau^{th}$ quantile of a random variable $Y$ is defined as

$$Q_Y(\tau) = F^{-1}(\tau) = \inf\{y : F_Y(y) \geq \tau\}$$

The quantile loss function is asymmetric around $\tau$, i.e., it incurs a higher penalty if the predicted function underestimates the true value of the target variable and lower penalty if it overestimates the true value. By choosing $\tau$ close to 1, quantile regression is biased towards predicting higher values of the time series. In the case when $\tau = 0.5$, the solution reduces to the conditional median of the target distribution. The preceding objective function is equivalent to solving the following linear programming problem:

$$\begin{aligned} \min_{\mathbf{p},\mathbf{q}} \quad & \tau \mathbf{1}_T^T \mathbf{p} + (1-\tau)\mathbf{1}_T^T \mathbf{q} \\ \text{s.t.} \quad & \mathbf{y} - \mathbf{X}\beta = \mathbf{p} - \mathbf{q} \\ & \mathbf{p} \geq \mathbf{0}, \mathbf{q} \geq \mathbf{0} \end{aligned}$$

The ORION framework for quantile loss function is designed to solve the following optimization problem.

$$\begin{aligned} \min_{\mathbf{p},\mathbf{q},\mathbf{w}_0,\mathbf{V}} \quad & \tau \mathbf{1}_T^T \mathbf{p} + (1-\tau)\mathbf{1}_T^T \mathbf{q} \\ & + \frac{1}{2}\mathrm{Tr}(\mathbf{V}^T(\mathbf{L} + \mu \mathbf{I}_T)\mathbf{V}) \\ & + \frac{\lambda}{2}||\mathbf{w}_0 - \mathbf{w}_0^{(n)}||_2^2 + \frac{\beta}{2}||\mathbf{V} - \mathbf{V}^{(n-1)}||_F^2 \\ \text{s.t.} \quad & \forall t \leq m_n, \mathbf{y}_t^{(n)} - \mathbf{w}_t^T \mathbf{x}_t^{(n)} = p_t - q_t \\ & \forall t, \mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t \\ & \mathbf{p} \geq \mathbf{0}, \mathbf{q} \geq \mathbf{0} \end{aligned}$$

Compared to ORION-$\varepsilon$, there are two additional parameters, $\mathbf{p}$ and $\mathbf{q}$, that must be estimated from the data. With this formulation, the prediction function is trained to fit the conditional quantiles and to ensure smoothness of the model parameters across the different tasks. The latter is attained by using a graph Laplacian to encode the task relationships. Unlike the original QR formulation, ORION-QR requires solving a quadratic programming problem. We employed the CVX software [50] to estimate the parameters using $\tau = 0.95$ to detect extreme (high) value events.

## 3.4 Experimental Evaluation

The proposed framework was applied to the soil moisture ensemble forecasting problem. The soil moisture forecasts were obtained from a seasonal hydrological prediction system for 12 major river basins in North America. 33 ensemble member forecasts were generated by running the model multiple times with different initial conditions. The data correspond to 40-day forecasts generated every 5 days for the time period between April, 2011 and September, 2011. The number of learning tasks to be predicted in each forecast set is $T = 8$.

The number of forecast sets in the data set is $N = 33$, which is equivalent to the number of online rounds. Since the model parameters were initialized to zero, the initial forecasts were poor until the model has been sufficiently trained. We use the first 23 forecast sets as "training data" and report the performance based on the predictions generated for the last 10 forecast sets ("test data"). We evaluated the performance of the different methods in terms of their mean absolute error (MAE) on the test data:

$$\text{MAE} = \frac{1}{80} \sum_{n=24}^{33} \sum_{t=1}^{8} |y_t^{(n)} - \hat{y}_t^{(n)}|,$$

where $\hat{\mathbf{y}}$ is the vector of predicted values.

### 3.4.1 Performance Comparison for ORION-$\varepsilon$

We compared the performance of ORION-$\varepsilon$ against the following baseline methods.

1. **Ensemble Median (EM)**: This is an unbiased aggregation of the ensemble member forecasts.

2. **Passive-Aggresive (PA) Algorithm** [29]: This single-task learning method assumes there is only one set of weights $\mathbf{w}$ to be estimated. Starting from the initial weights $\mathbf{w} = 0$, we update the weights in each round as follows (from the first to the last task):

   $$\mathbf{w}^{(n)} = \mathbf{w}^{(n-1)} + \text{sign}(y^{(n)} - \hat{y}^{(n)})\tau\mathbf{x}^{(n)}$$

   where $\tau = \ell_n / \|\mathbf{x}^{(n)}\|_2^2$, and $\ell$ is the $\varepsilon$-insensitive loss.

3. **Tracking Climate Models (TCM)** [77]: This method uses a Hidden Markov Model to track the current best ensemble member. Unlike ORION-$\varepsilon$, the weights estimated by TCM range between 0 and 1, which means the aggregated forecast always fall within the range of the ensemble member forecasts. Since the method is designed for single-task learning, we modify its implementation to handle $T$ instances in each round. Instead of using squared error between the observed and predicted target values, the loss is computed based on the average squared error over $m_n$ instances.

4. **Online Multi-task Learning with a Shared Loss(OMTLSL)** [34]: This is a modified implementation of the approach given in [34], which was originally proposed for the hinge loss function. Here, we use the $\varepsilon$-insensitive loss function for our regression problem and ignore the use of slack variables. The modified objective function is given by:

$$\begin{aligned} &\arg\min_{\mathbf{w}_t} \frac{1}{2}\sum_{t=1}^{T} \|\mathbf{w}_t - \mathbf{w}_t^{(n-1)}\|_2^2 \\ \text{s.t.} \quad &\forall t \leq m_n,\ |\mathbf{w}_t^T \mathbf{x}_t^{(n)} - y_t^{(n)}| \leq \varepsilon \end{aligned}$$

   The optimization problem can be solved using a similar technique as that for ORION-$\varepsilon$.

5. **Linear Algorithms for Online Multi-task Learning(LAOM)** [21]: The original method was proposed for multi-task classification. We modify the loss function to be squared loss for regression problem. The default approach given in [21] assumes the data is only available one task at a time, and thus, the models are updated one task at a time according to the task relationship matrix. As a consequence, the learning process depends on ordering of the task. ORION-$\varepsilon$ does not require such an assumption.

For a fair comparison, all the baseline methods adopt the same online learning with restart strategy (similar to ORION-$\varepsilon$) to deal with the partially observed data.

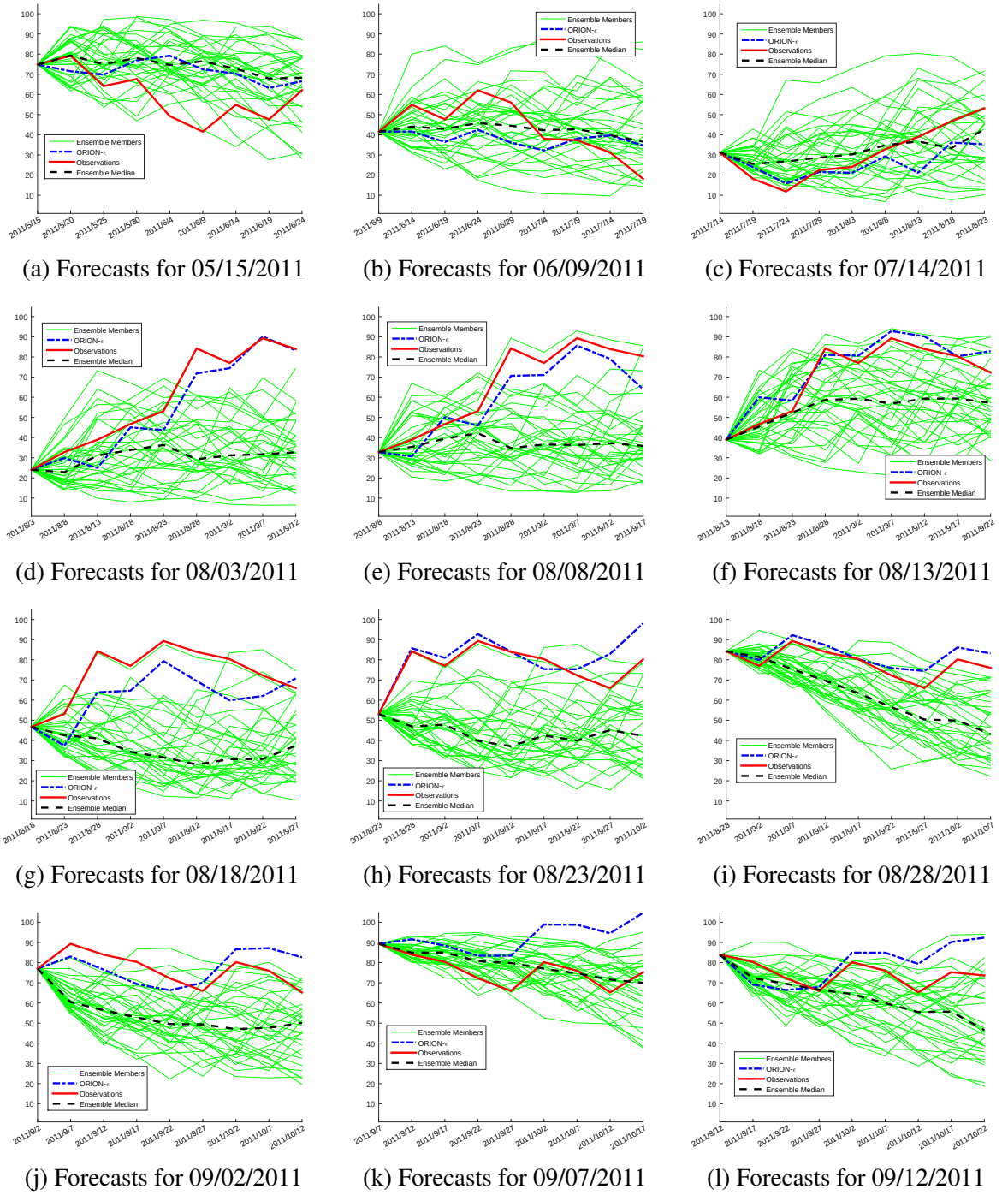
Table 3.2 compares the results of the different methods. As can be seen from the table, ORION-$\varepsilon$ works better than the baseline methods on all 12 data sets. In particular, it outperforms OMTLSL, which is a state-of-the-art online multi-task learning algorithm, on all the data sets. Unlike OMTL-

Table 3.2: Comparison of mean absolute error (MAE) for ORION-$\varepsilon$ against baseline methods on soil moisture data

| | ORION-$\varepsilon$ | EM | PA | TCM | OMTLSL | LAOM |
|---|---|---|---|---|---|---|
| arkansusred | **2.740** | 4.189 | 3.788 | 3.659 | 5.423 | 2.767 |
| calinevada | **3.398** | 4.919 | 4.281 | 4.265 | 4.422 | 4.257 |
| colorado | **4.362** | 5.934 | 5.741 | 5.634 | 6.068 | 5.674 |
| columbia | **4.411** | 6.000 | 6.439 | 6.475 | 6.225 | 5.370 |
| lowermiss | **9.891** | 12.023 | 11.639 | 10.671 | 14.975 | 11.951 |
| midatlantic | **13.473** | 24.381 | 25.140 | 20.961 | 23.143 | 27.507 |
| missouri | **3.699** | 6.029 | 5.470 | 6.575 | 6.913 | 5.269 |
| northcentral | **6.292** | 8.789 | 8.700 | 9.157 | 10.838 | 7.298 |
| northeast | **7.422** | 22.040 | 20.490 | 19.471 | 24.877 | 23.824 |
| ohio | **14.535** | 17.023 | 15.107 | 15.021 | 19.064 | 16.436 |
| southeast | **8.229** | 8.951 | 8.778 | 9.136 | 10.966 | 9.158 |
| westgulf | **3.790** | 4.697 | 4.490 | 5.689 | 6.150 | 4.369 |

SL, ORION-$\varepsilon$ enforces the constraint $\mathbf{w}_t = \mathbf{w}_0 + \mathbf{v}_t$, which helps to improve the performance of the ensemble forecasting task. As will be shown in Table 3.5, the improvement is still observed even when the task relationship is removed (i.e., comparing OMTLSL against ORION-$\varepsilon$-NR). Comparing ORION-$\varepsilon$ against LAOM, the results suggest the benefit of updating the multiple tasks simultaneously instead of updating them one task at a time.

As further evidence, Fig. 3.3 shows the predicted time series for ORION-$\varepsilon$ and the Ensemble Median (EM) method on the northeast data set. The first five figures are from the training set and the remaining ten figures are from the test set. Initially, the time series predicted by ORION-$\varepsilon$ is similar to EM (see Fig. 3.3a to 3.3c). As more data becomes available, the predictions by ORION-$\varepsilon$ is closer to observation data than EM (Fig. 3.3d to 3.3j). In Fig. 3.3k, there appears to be a sudden shift that causes the performance of ORION-$\varepsilon$ to degrade significantly. However, after one update, ORION-$\varepsilon$ recovers from the mistake and its prediction follows closely the observation data again (Fig. 3.3l). Fig. 3.4 shows the absolute error of ORION-$\varepsilon$ and EM during the last 15 rounds of the training data and the 10 rounds in test data, for both Northeast and Midatlantic data. Although the performance of ORION-$\varepsilon$ is slightly worse than EM at the beginning, after sufficient



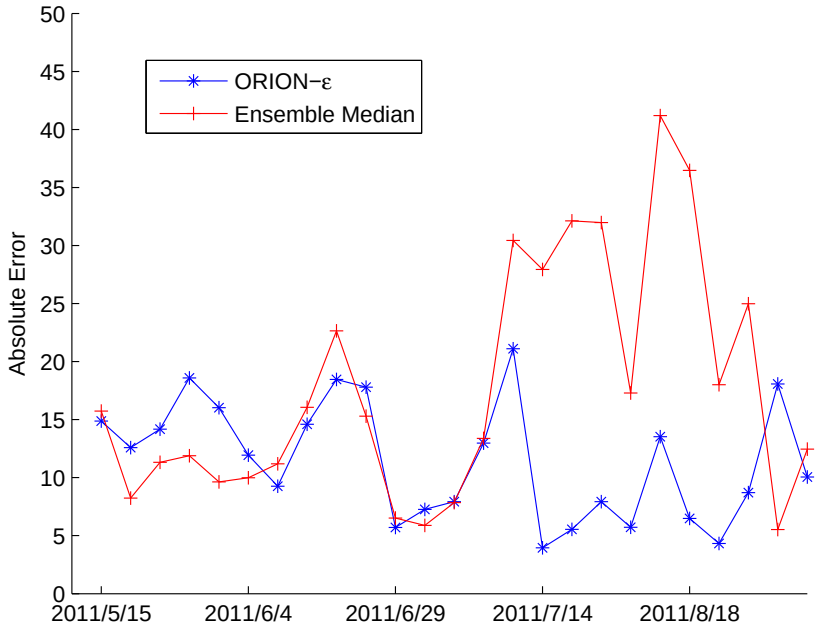

(a) Forecasts for 05/15/2011 (b) Forecasts for 06/09/2011 (c) Forecasts for 07/14/2011

(d) Forecasts for 08/03/2011 (e) Forecasts for 08/08/2011 (f) Forecasts for 08/13/2011

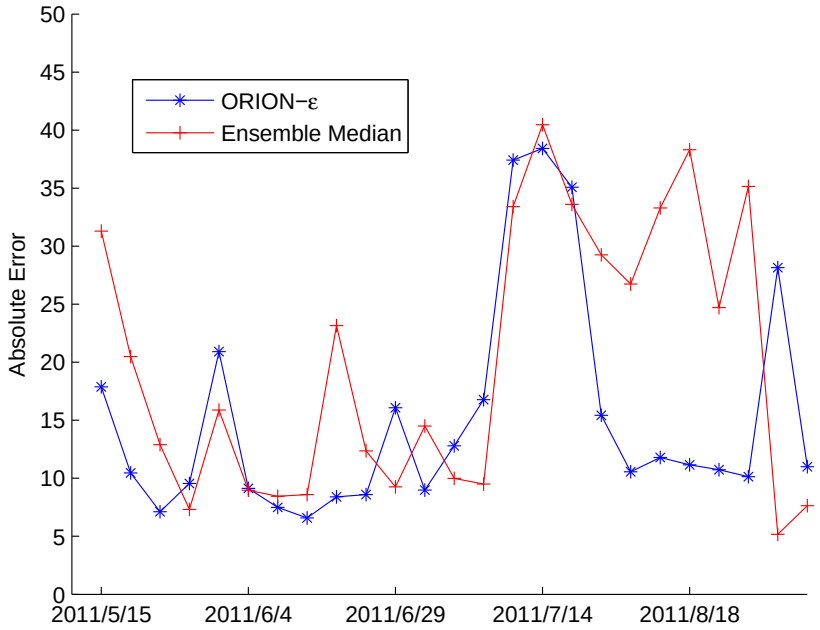
(g) Forecasts for 08/18/2011 (h) Forecasts for 08/23/2011 (i) Forecasts for 08/28/2011

(j) Forecasts for 09/02/2011 (k) Forecasts for 09/07/2011 (l) Forecasts for 09/12/2011

Figure 3.3: Forecasts on Dataset Northeast for ORION-$\varepsilon$. Fig. 3.3a - 3.3c are results from the training set and Fig. 3.3d - 3.3l are results from the test set. Note that in the early stage of the online learning process, ORION-$\varepsilon$ performs similar to Ensemble Median (see Fig. 3.3a - 3.3b), and ORION-$\varepsilon$ starts to follow the observation from Fig. 3.3c.




(a) Northeast Data




(b) Mid-Atlantic Data

Figure 3.4: Mean absolute error for ORION-$\varepsilon$ and Ensemble Median on the Northeast and Midatlantic data.

Table 3.3: Comparison of runtime (in seconds) on the Northeast data set.

| | ORION | PA | TCM | OMTLSL | LAOM |
|---|---|---|---|---|---|
| Runtime per Round | 0.0430 | 0.0005 | 0.0070 | 0.0013 | 0.0014 |
| Total Runtime | 1.0909 | 0.0208 | 0.1252 | 0.0647 | 0.0606 |

training, ORION-$\varepsilon$ appears to perform significantly better than EM.

We also compared the runtime of ORION against other baseline methods. The total runtime as well as average runtime per round for the Northeast data set is shown in Table 3.3. ORION is relatively slower than other baselines, which is not surprising since it has to backtrack and revise some the older models in each round unlike other methods. Nevertheless, the additional overhead, which is less than 50 ms for each update round, is reasonable for many ensemble forecasting problems that require only a one-time daily update of their models. It is therefore an acceptable tradeoff to achieve the accuracy improvement shown in Table 3.2.

### 3.4.2 ORION with Quantile Regression (ORION-QR)

To evaluate the performance of the ORION framework with quantile loss function, the observation data were initially preprocessed to identify the forecast periods when the soil moisture value is extremely high, i.e., more than 1.64 standard deviations away from the mean. Non-extremes are defined as those values located within the 90% confidence interval of the mean. Only 5 of the 12 data sets contain extreme events in the test period. The number of extreme events during the test period for the five data sets are as follows: arkansusred (14), colorado (14), midatlantic (44), southeast (6), and westgulf (41). We report our experimental results for these data sets only, comparing ORION-QR against EM, ORION-$\varepsilon$, and the following two baselines:

1. **Quantile Regression (QR)**: This is the original QR method used in a batch mode. It assumes all the tasks share the same set of weights $\mathbf{w}$.

2. **Online Quantile Regression (OQR)**: This is a variant of the PA [29] algorithm using a

Table 3.4: Comparison of F1 measure for predicting occurrence of extreme events.

| | ORION-QR | EM | ORION-$\varepsilon$ | QR | OQR |
|---|---|---|---|---|---|
| arkansusred | 0.465 | 0.4167 | 0.200 | **0.500** | **0.500** |
| colorado | **0.593** | 0.148 | 0.500 | 0.406 | 0.3030 |
| midatlantic | **0.500** | 0.3019 | **0.500** | 0.387 | 0.275 |
| southeast | **0.444** | 0.3077 | 0.250 | 0 | 0.286 |
| westgulf | 0.598 | 0.6122 | 0.451 | 0.568 | **0.625** |

quantile loss function. The objective function is modified as follows:

$$\begin{aligned} \min_{\mathbf{p},\mathbf{q},\mathbf{w}} \quad & \tau \mathbf{1}_T^T \mathbf{p} + (1-\tau)\mathbf{1}_T^T \mathbf{q} \\ + \quad & \frac{\lambda}{2}\|\mathbf{w} - \mathbf{w}^{(n-1)}\|_2^2 \\ \text{s.t.} \quad & \forall t, \mathbf{y}_t^{(n)} - \mathbf{w}^T \mathbf{x}_t^{(n)} = p_t - q_t \\ & \mathbf{p} \geq \mathbf{0}, \mathbf{q} \geq \mathbf{0} \end{aligned} \tag{3.9}$$

which can be solved using standard quadratic programming solvers such as CVX.

For this experiment, we are interested in the predictions of extreme events. A predicted event is a true positive (TP) if it is a real event and a false positive (FP) if it does not correspond to a real event. A false negative (FN) corresponds to a real event that was not detected by the algorithm. We use the following F1-measure as our evaluation metric:

$$\text{F1} = \frac{2\ \text{TP}}{2\ \text{TP} + \text{FP} + \text{FN}}$$

Table 3.4 shows that ORION-QR outperforms both EM and ORION-$\varepsilon$ in 4 out of 5 data sets. The latter suggests quantile loss is more effective than $\varepsilon$-insensitive loss when dealing with extreme value prediction. Compared to single-task learning methods, ORION-QR outperforms QR in 4 out of 5 data sets and OQR in 3 out of 5 data sets. Furthermore, there is no significant difference when comparing the number of data sets in which the batch version of QR outperforms its online version, OQR.

For the southeast dataset, the F1-measure for QR is zero, which suggests that the method fails to correctly predict extreme events in the test data. This is because there are only 6 extreme events

in the test period of southeast dataset, which makes it a hard prediction problem. In contrast, the frequency of extreme events for other datasets is at least 14. The presence of concept drift in the time series data also makes QR less effective compared to OQR. While OQR performs better on the southeast dataset, it is still significantly worse than our proposed ORION-QR algorithm.

### 3.4.3 Sensitivity Analysis

This section analyzes the sensitivity of the three input parameters of the ORION framework.[2] Although the experimental results shown here are for the northeast data set, a similar behavior was observed in other data sets. For each experiment, we vary the value of one parameter and fix the values of the remaining two.

The parameter $\mu$ controls sparsity of the weights for $\mathbf{v}_t$. Note that $\mu$ must be non-negative to ensure the matrix $\mathbf{L} + \mu \mathbf{I}_T$ is positive semi-definite. Fig. 3.5(a) shows that MAE improves as $\mu$ increases. As long as $\mu$ is sufficiently large ($> 50$), its MAE becomes stable. This result is not surprising since the prediction tasks are highly correlated. Therefore, the weights for $\mathbf{v}_t$ are expected to be small.

The parameters $\beta$ and $\lambda$ affect how much information should be retained from previous rounds. Since the weights for $\mathbf{v}_t$ are small, the results are not that sensitive to changes in $\beta$ (see Fig. 3.5(a)). The results are more sensitive to choice of $\lambda$. If $\lambda$ is too small, $\mathbf{w}_0$ deviates significantly from its previous value. Conversely, if $\lambda$ is set too large, the variation in $\mathbf{w}_0$ becomes too slow and the algorithm requires more training examples in order to converge. In practice, $\lambda$ can be set based on its performance on the training data.

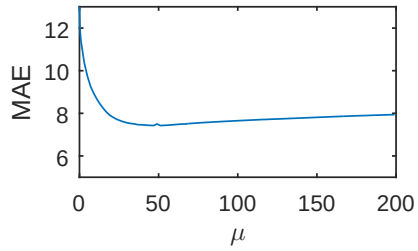
### 3.4.4 Variations of ORION-$\varepsilon$ Framework

The ORION framework uses two types of information to update its model parameters. First, it uses the $\lambda$ and $\beta$ regularizers to control the amount of information retained from previous rounds.

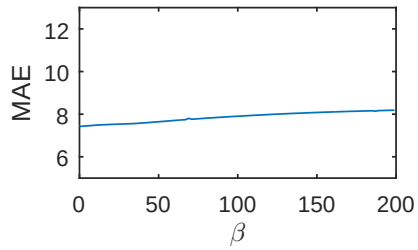
[2]Similar to other works, $\varepsilon$ is typically fixed to a small number. So we set $\varepsilon = 10^{-3}$ in all our experiments.

Second, it relies on the $\mathbf{Q}$ matrix to control the amount on information shared among the tasks.

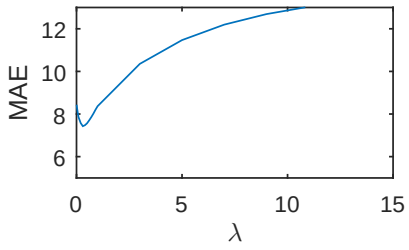
We investigate two variations of the ORION-$\varepsilon$ framework. We first consider the case when $\beta = 0$, which implies that the weight vectors $\mathbf{v}_t$ are independent of their values in the previous round.[3] We denote the approach as ORION-$\varepsilon$-$\beta$. Experimental results given in Table 3.5 showed that ORION-$\varepsilon$ outperforms ORION-$\varepsilon$-$\beta$ in 9 out of 12 data sets. However, the difference is not that significant except for 3 of 12 the data sets. The second variation of our framework removes the task relationship by setting $\mathbf{Q} = \mathbf{0}$. This approach is denoted as ORION-$\varepsilon$-NR. Based on the results




(a) Effect of varying $\mu$ on MAE




(b) Effect of varying $\beta$ on MAE




(c) Effect of varying $\lambda$ on MAE

Figure 3.5: Sensitivity Analysis of ORION-$\varepsilon$. Fig. 3.5a shows that ORION-$\varepsilon$ tends to choose a large value of $\mu$; Fig. 3.5b shows that ORION-$\varepsilon$ is not that sensitive to $\beta$; Fig. 3.5c shows that $\lambda$ is the parameter to be tuned in practice.

[3] Setting $\lambda = 0$ makes $\mathbf{R} + \mathbf{Q}$ becomes a singular matrix. This situation is not considered in this study.

Table 3.5: Comparison of mean absolute error (MAE) for different variations of ORION-$\varepsilon$ framework

| | ORION-$\varepsilon$ | ORION-$\varepsilon$-NR | ORION-$\varepsilon$-$\beta$ |
|---|---|---|---|
| arkansusred | **2.740** | 3.937 | **2.740** |
| calinevada | 3.398 | 4.781 | **3.390** |
| colorado | **4.362** | 4.599 | 4.410 |
| columbia | 4.411 | **4.278** | 5.156 |
| lowermiss | **9.891** | 10.038 | 12.047 |
| midatlantic | **13.473** | 13.809 | 13.527 |
| missouri | 3.699 | **3.370** | 5.049 |
| northcentral | 6.292 | **6.163** | 6.475 |
| northeast | **7.422** | 7.814 | 7.427 |
| ohio | 14.535 | **14.463** | 14.987 |
| southeast | **8.229** | 9.583 | 8.232 |
| westgulf | 3.790 | 5.002 | **3.780** |

given in Table 3.5, ORION-$\varepsilon$ outperforms ORION-$\varepsilon$-NR in 8 out of 12 datasets, with substantial improvements in at least 4 of them. This shows the value of incorporating the task relationship into the ORION-$\varepsilon$ framework.

## 3.5 Conclusion

This chapter presents an online regularized multi-task regression framework for ensemble forecasting tasks. This framework is unique in that it uses an online learning with restart strategy to update its models. It is also flexible in that it can accommodate both $\varepsilon$-insensitive and quantile loss functions. Experimental results confirm the superiority of the proposed framework compared to several baseline methods.

# CHAPTER 4

# MULTI-TASK LEARNING FRAMEWORK FOR MULTI-LOCATION PREDICTION

[1] The geospatio-temporal prediction task typically requires making predictions for a response variable at multiple locations [121]. For example, climate scientists are interested to obtain future climate projections of temperature or precipitation for multiple locations in a geographical region of interest. Similarly, in hydrology, soil moisture forecasts are periodically generated at multiple river and lake basins for drought monitoring applications. To perform such predictions, the simplest approach would be to fit a model at each location using only its local observations. This approach may not be effective because it fails to exploit the shared information among the prediction tasks. This is especially true for geospatio-temporal prediction problems following Tobler's first law of geography, which states that "Everything is related to everything else, but near things are more related than distant things." [98].

To overcome this challenge, this thesis considers the application of multi-task learning (MTL) [20] to geospatio-temporal prediction problems. MTL is a widely-used approach for solving multiple, related learning tasks by exploiting the common structure of the problem. For geospatio-temporal data, the task relatedness could be represented by the spatial proximity or landscape similarity between different locations (e.g., elevation, slope, and other topographic variables).

However, incorporating such information alone into existing MTL framework may not be sufficient since the spatio-temporal variabilities of the data are potentially influenced by several macroscale phenomena, whose impact varies from one location to another. For example, the well-known El-Ninö phenomenon affects differentially weather patterns in North America and the rest of the world. A flexible MTL framework that can capture the spatial autocorrelation of the data as well as the influence of broader scale effects is therefore needed.

To address this problem, we present a novel multi-location prediction framework called GSpartan, which stands for **G**eo**SPA**tio-tempo**R**al mul**T**i-t**A**sk lear**N**ing. GSpartan is developed based

[1] This chapter is based on the previous publication [109].

on the assumption that the prediction models for all locations share a common set of low-rank base models, where each base model may represent a macroscale phenomenon that potentially explains the variability of the data. The local model at each location is constructed as a linear combination of these base models. A graph Laplacian regularization is introduced to capture the spatial autocorrelation of the data, thus providing a natural way to specify the relationships among the prediction tasks. To ensure interpretability of the models, we also add sparsity and non-negativity constraints into the GSpartan formulation. We evaluated the performance of GSpartan against several baseline methods on climate data collected from 37 randomly chosen weather stations in Canada. Our experimental results for predicting monthly precipitation showed that GSpartan outperformed single-task learning (STL) and two other existing multi-task learning (MTL) methods in at least 75% of the stations. The ability of GSpartan to outperform other baseline methods increases to more than 84% of the stations if the training data available at each station is limited to only 1 year.

## 4.1 Preliminaries

Let $S \subset \Re^2$ be a set of geo-referenced locations, where each location $s \in S$ is associated with a set of temporal fields. One of the fields is designated as the response variable we are interested in predicting, while the rest are considered predictor variables. For instance, in climate modeling, the response variable may correspond to monthly precipitation values recorded at a weather station whereas the predictor variables correspond to outputs generated from a global or regional climate model [26]. An example of the multi-location prediction task here is to infer future monthly values of precipitation for all the locations based on their historical observations and outputs from the climate models.

Formally, consider a geospatio-temporal data set $\mathscr{D} = \{(\mathbf{X}_1, \mathbf{y}_1), (\mathbf{X}_2, \mathbf{y}_2), \cdots, (\mathbf{X}_{|S|}, \mathbf{y}_{|S|})\}$, where each tuple, $(\mathbf{X}_s, \mathbf{y}_s)$, denote the temporal fields at location $s$. Let $\mathbf{X}_s \in \mathscr{R}^{n_s \times d} = [\mathbf{x}_{s,1}^T, \cdots, \mathbf{x}_{s,n_s}^T]$ be the matrix of predictor variables and $\mathbf{y}_s \in \mathscr{R}^{n_s}$ be the time series for the response variable observed at the discrete time points $1, 2, \cdots, n_s$.

For single-task learning (STL), each location $s$ is treated as a separate learning task. Let $n_s$ be the number of training examples available for task $s$ and $d$ be the number of predictor variables. STL seeks to learn a (local) task model $f_s(\mathbf{x};\mathbf{w}_s)$ for each location in such a way that the following loss function is minimized:

$$\min_{\mathbf{W}} \sum_{s=1}^{|S|} \sum_{i=1}^{n_s} \ell_s \left[ f_s(\mathbf{x}_{s,i};\mathbf{w}_s), y_{s,i} \right]$$

where $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_{|S|}] \in \mathscr{R}^{d \times |S|}$ denote the model parameters and $\ell_s(\cdot)$ represents the loss function for task $s$. For brevity, we consider only task models of the form $f_s(\mathbf{x}_s;\mathbf{w}_s) = \mathbf{x}_s^T \mathbf{w}_s$ with a squared loss function.

## 4.2 GSpartan

This section presents our proposed GSpartan framework for multi-location prediction. The framework was designed to satisfy the following three requirements:

1. It should learn a low-rank representation of the task models. The low-rank representation, defined by a set of base models, represents the possible macroscale phenomena that could help explain the temporal variability of the data.

2. It should incorporate domain knowledge about the spatial autocorrelation of the response variable among the various locations.

3. To ensure interpretability, each base model should depend only on a small subset of the predictor variables. In addition, each local model should be comprised of a small number of base models.

The following objective function is used in GSpartan to train the local models jointly:

$$\min_{\mathbf{W}} \sum_{s=1}^{|S|} \sum_{i=1}^{n_s} \ell_s \left[ f_s(\mathbf{x}_{s,i};\mathbf{w}_s), y_{s,i} \right] + \Omega(\mathbf{W})$$

where $\Omega(\mathbf{W})$ is a regularization term. In the following, we will discuss how to formulate the objective function to meet the requirements stated above.

**Low-rank Representation**: We assume that the local models can be expressed as a product of two low-rank matrices, i.e., $\mathbf{W} = \mathbf{UV}$, where $\mathbf{U} \in \mathscr{R}^{d \times k}$ and $\mathbf{V} \in \mathscr{R}^{k \times |S|}$. The matrix $\mathbf{U}$ is a feature representation of the base models while $\mathbf{V}$ expresses the weighted combination of the base models that form the local model at each location. Specifically, $\mathbf{u}_i$ is a column vector in $\mathbf{U}$ that represents the feature vector for the $i$-th base model, while $\mathbf{v}_j$ is a column vector in $V$ that represents the weights of the base models defining the $j$-th local model.

**Task Relation Matrix**: We employ a graph Laplacian regularization to incorporate information about the spatial autocorrelation between locations. Let $\mathbf{A}$ be the task relation matrix, where $\mathbf{A}_{i,j}$ measures the spatial autocorrelation between locations $i$ and $j$. The graph Laplacian regularizer can be written as follows:

$$\Omega_r(\mathbf{W}) = \sum_{i,j=1}^{|S|} \mathbf{A}_{i,j} \|\mathbf{w}_i - \mathbf{w}_j\|_2^2 = \mathrm{Tr}\left[\mathbf{W}(\mathbf{D} - \mathbf{A})\mathbf{W}^T\right]$$

where $\mathbf{D}$ is a diagonal matrix with $\mathbf{D}_{i,i} = \sum_j \mathbf{A}_{ij}$. Intuitively, if $\mathbf{A}_{i,j}$ is large, the graph Laplacian regularizer term will also be large unless $\mathbf{w}_i$ is similar to $\mathbf{w}_j$. Thus, the graph Laplacian is simply a re-statement of Tobler's first law of geography.

**Model Interpretability**: Sparsity constraints can be imposed to ensure that each base model depends only on a small subset of the predictor variables and each task model is a linear combination of a few base models. To improve interpretability, the coefficients of the weighted linear combination should also be non-negative. To satisfy these requirements, the following $L_1$ regularization penalty is added to the objective function:

$$\begin{aligned} & \Omega_s(\mathbf{W}) = \lambda_1 \|\mathbf{V}\|_1 + \lambda_2 \|\mathbf{U}\|_1 \\ \text{s.t.} \quad & \mathbf{W} = \mathbf{UV}, \;\; \mathbf{V} \succeq 0 \end{aligned}$$

where $\lambda_1$ and $\lambda_2$ are the regularization parameters. The notation $\mathbf{V} \succeq 0$ implies all elements of $\mathbf{V}$ must be non-negative.

Putting everything together, we can now express our objective function for GSpartan (assuming a squared loss function) as follows:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{U},\mathbf{V}} \quad & \frac{1}{2}\sum_{s=1}^{|S|} \|\mathbf{X}_s\mathbf{w}_s - \mathbf{y}_s\|_2^2 + \lambda_1\|\mathbf{V}\|_1 + \lambda_2\|\mathbf{U}\|_1 + \frac{\lambda_3}{2}\operatorname{Tr}(\mathbf{W}(\mathbf{D}-\mathbf{A})\mathbf{W}^T) \\ \text{s.t.} \quad & \mathbf{V} \succeq 0,\ \mathbf{W} = \mathbf{U}\mathbf{V} \end{aligned}$$

Since $\mathbf{W} = \mathbf{U}\mathbf{V}$, the objective function reduces to the following simplified expression:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V}} \quad & \frac{1}{2}\sum_{i=1}^{|S|} \|\mathbf{X}_s\mathbf{U}\mathbf{v}_i - \mathbf{y}_s\|_2^2 + \lambda_1\|\mathbf{V}\|_1 + \lambda_2\|\mathbf{U}\|_1 + \frac{\lambda_3}{2}\operatorname{Tr}(\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T\mathbf{U}^T) \\ \text{s.t.} \quad & \mathbf{V} \succeq 0 \end{aligned} \tag{4.1}$$

The preceding constrained optimization problem can be solved using a block coordinate descent approach, by alternately solving for $\mathbf{U}$ and $\mathbf{V}$. Details for solving each step efficiently is given below.

**Solve U, given V:**

When $\mathbf{V}$ is fixed, the objective function can be simplified as follows:

$$\min_{\mathbf{U}} \quad \frac{1}{2}\sum_{i=1}^{|S|} \|\mathbf{X}_s\mathbf{U}\mathbf{v}_i - \mathbf{y}_s\|_2^2 + \lambda_2\|\mathbf{U}\|_1 + \frac{\lambda_3}{2}\operatorname{Tr}(\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T\mathbf{U}^T) \tag{4.2}$$

This optimization problem can be efficiently solved using the proximal gradient descent method. Proximal gradient descent is commonly used to solve optimization problems containing non-differentiable components. The algorithm also has faster convergence compared to other methods such as subgradient descent. The basic idea here is to minimize a corresponding upper bound function of the original objective function [84]. Based on standard assumptions such as Lipschitz continuity on the partial gradient of the differentiable part of the objective function, we can use a tighter upper bound to approximate the original objective function. Here, we will use *Prox-linear* [112] to solve our optimization problem, with the following update formula:

$$\mathbf{U}^k = \operatorname*{argmin}_{\mathbf{U}} (\mathbf{U} - \hat{\mathbf{U}}^{k-1})^T \hat{\mathbf{g}}_U^k + \frac{\tau_U^{k-1}}{2}\|\mathbf{U} - \hat{\mathbf{U}}^{k-1}\|_F^2 + \lambda_2\|\mathbf{U}\|_1, \tag{4.3}$$

where

$$\hat{\mathbf{g}}_U^k = \sum_{s=1}^{|S|} \left( -\mathbf{X}_s^T\mathbf{y}_s\mathbf{v}_s^T + \mathbf{X}_s^T\mathbf{X}_s\mathbf{U}\mathbf{v}_s\mathbf{v}_s^T \right) + \lambda_3\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T$$

and

$$\hat{\mathbf{U}}^{k-1} = \mathbf{U}^{k-1} + \omega_U^{k-1}(\mathbf{U}^{k-1} - \mathbf{U}^{k-2})$$

The solution for problem (4.3) is given by

$$\mathbf{U}^k = \mathscr{S}_{\tau_U^{k-1}/\lambda_2}(\hat{\mathbf{U}}^{k-1} - \frac{\hat{\mathbf{g}}^{k-1}}{\tau_{\mathbf{U}}^{k-1}}) \tag{4.4}$$

where $\mathscr{S}_\alpha(t) = \text{sign}(t)(\max(|t| - \alpha, 0)$ is a component-wise soft-thresholding function.

**Solve V, given U**

Similarly, when $\mathbf{U}$ is fixed, the objective function becomes:

$$\min_{\mathbf{V}} \quad \frac{1}{2}\sum_{i=1}^{|S|} \|\mathbf{X}_s\mathbf{U}\mathbf{v}_i - \mathbf{y}_s\|_2^2 + \lambda_1\|\mathbf{V}\|_1 + \frac{\lambda_3}{2}\text{Tr}(\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T\mathbf{U}^T) \tag{4.5}$$

The update formula for $\mathbf{V}$ using proximal gradient descent approach is:

$$\mathbf{V}^k \quad = \quad \underset{\mathbf{V}}{\text{argmin}}(\mathbf{V} - \hat{\mathbf{V}}^{k-1})^T\hat{\mathbf{g}}_V^k + \frac{\tau_V^{k-1}}{2}\|\mathbf{V} - \hat{\mathbf{V}}^{k-1}\|_F^2 + \lambda_1\|\mathbf{V}\|_1 \tag{4.6}$$

where

$$\hat{\mathbf{g}}_V^k \quad = \quad \mathbf{P} + \lambda_3\mathbf{U}^T\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})$$

The $s$-th column of matrix $\mathbf{P}$ is given by

$$\mathbf{p}_s = -\mathbf{U}^T\mathbf{X}_s^T\mathbf{y}_s + \mathbf{U}^T\mathbf{X}_s^T\mathbf{X}_s\mathbf{U}\mathbf{v}_s$$

and

$$\hat{\mathbf{V}}^{k-1} = \mathbf{V}^{k-1} + \tau_V^{k-1}(\mathbf{V}^{k-1} - \mathbf{V}^{k-2})$$

The solution for problem (4.6) is given by

$$\mathbf{V}^k = \mathscr{S}_{\omega_V^{k-1}/\lambda_2}(\hat{\mathbf{V}}^{k-1} - \frac{\hat{\mathbf{g}}^{k-1}}{\tau_V^{k-1}}) \tag{4.7}$$

A further projection step is needed to ensure that the elements of $\mathbf{V}$ are non-negative.

Note that the learning rates $\tau_U^{k-1}$ and $\tau_V^{k-1}$ are provided by the Lipschitz continuous constant of the partial gradient $\hat{\mathbf{g}}_U^k$ and $\hat{\mathbf{g}}_V^k$, respectively. The learning rates are given in Theorems 2 and 3 below. The extrapolation term $\omega$ is selected to be $0 \leq \omega^k \leq \delta_\omega\sqrt{\frac{\tau^{k-2}}{\tau^{k-1}}}$ for $\delta_\omega < 1$.

**Theorem 2.** *The partial gradient $\hat{\mathbf{g}}_U$ is Lipschitz continuous with the constant*

$$\tau_U \quad = \quad \sum_{s=1}^{|S|} \|\mathbf{X}_s^T \mathbf{X}_s\| \|\mathbf{v_s} \mathbf{v_s^T}\| + \lambda_3 \|\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T\|$$

*Proof.* For any $U$ and $U^* \in \mathscr{R}^{d\times k}$,

$$\begin{aligned}
& \|\hat{\mathbf{g}}_U - \hat{\mathbf{g}}_{U^*}\| \\
= & \left\| \sum_{s=1}^{|S|} \left( \mathbf{X}_s^T \mathbf{X}_s \mathbf{U} \mathbf{v}_s \mathbf{v}_s^T - \mathbf{X}_s^T \mathbf{X}_s \mathbf{U}^* \mathbf{v}_s \mathbf{v}_s^T \right) + \lambda_3 (\mathbf{U}-\mathbf{U}^*)\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T \right\| \\
\leq & \sum_{s=1}^{|S|} \|\mathbf{X}_s^T \mathbf{X}_s \mathbf{U} \mathbf{v}_s \mathbf{v}_s^T - \mathbf{X}_s^T \mathbf{X}_s \mathbf{U}^* \mathbf{v}_s \mathbf{v}_s^T\| + \lambda_3 \|\mathbf{U}-\mathbf{U}^*\| \|\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T\| \\
\leq & \sum_{s=1}^{|S|} \|\mathbf{X}_s^T \mathbf{X}_s\| \|\mathbf{v_s} \mathbf{v_s^T}\| \|\mathbf{U}-\mathbf{U}^*\| + \lambda_3 \|\mathbf{U}-\mathbf{U}^*\| \|\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T\| \\
= & \|\mathbf{U}-\mathbf{U}^*\| \left( \sum_{s=1}^{|S|} \|\mathbf{X}_s^T \mathbf{X}_s\| \|\mathbf{v_s} \mathbf{v_s^T}\| + \lambda_3 \|\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T\| \right)
\end{aligned}$$

□

**Theorem 3.** *Assuming $\|\mathbf{X}_s\| \leq R$, the partial gradient $\hat{\mathbf{g}}_V$ is Lipschitz continuous with the constant*

$$\tau_V \quad = \quad \|\mathbf{U}\|^2 R^2 + \lambda_3 \|\mathbf{U}^T \mathbf{U}\| \|\mathbf{D}-\mathbf{A}\|$$

*Proof.* For any $\mathbf{V}$ and $\mathbf{V}^* \in \mathscr{R}^{k\times S}$,

$$\begin{aligned}
& \|\hat{\mathbf{g}}_V - \hat{\mathbf{g}}_{V^*}\| \\
= & \left\| \mathbf{P} + \lambda_3 \mathbf{U}^T \mathbf{U} \mathbf{V}(\mathbf{D}-\mathbf{A}) - \mathbf{P}^* - \lambda_3 \mathbf{U}^T \mathbf{U} \mathbf{V}^* (\mathbf{D}-\mathbf{A}) \right\| \\
\leq & \|\mathbf{P}-\mathbf{P}^*\| + \lambda_3 \|\mathbf{U}^T \mathbf{U}\| \|\mathbf{D}-\mathbf{A}\| \|\mathbf{V}-\mathbf{V}^*\| \\
= & \sum_{s=1}^{|S|} \|\mathbf{U}^T \mathbf{X}_s^T \mathbf{X}_s \mathbf{U}(\mathbf{v}_s - \mathbf{v}_s^*)\| + \lambda_3 \|\mathbf{U}^T \mathbf{U}\| \|\mathbf{D}-\mathbf{A}\| \|\mathbf{V}-\mathbf{V}^*\| \\
\leq & \sum_{s=1}^{|S|} \|\mathbf{U}^T \mathbf{X}_s^T \mathbf{X}_s \mathbf{U}\| \|\mathbf{v}_s - \mathbf{v}_s^*\| + \lambda_3 \|\mathbf{U}^T \mathbf{U}\| \|\mathbf{D}-\mathbf{A}\| \|\mathbf{V}-\mathbf{V}^*\| \\
\leq & \|\mathbf{U}\|^2 R^2 \sum_{s=1}^{|S|} \|\mathbf{v}_s - \mathbf{v}_s^*\| + \lambda_3 \|\mathbf{U}^T \mathbf{U}\| \|\mathbf{D}-\mathbf{A}\| \|\mathbf{V}-\mathbf{V}^*\| \\
= & \|\mathbf{V}-\mathbf{V}^*\| \left( \|\mathbf{U}\|^2 R^2 + \lambda_3 \|\mathbf{U}^T \mathbf{U}\| \|\mathbf{D}-\mathbf{A}\| \right)
\end{aligned}$$

□

A summary of the GSpartan framework is shown in Algorithm 2 below.

**Input**: Dataset $\mathscr{D} = \{(\mathbf{X}_1, \mathbf{y}_1), ..., (\mathbf{X}_S, \mathbf{y}_S)\}$, Task relation matrix $A$, parameters $\lambda_1$, $\lambda_2$, $\lambda_3$;
**Initialize**: Randomly generate $\mathbf{U}$ and $\mathbf{V}$ and set $k = 1$
**Block coordinate descent:**
**while** *not converge* **do**
    **Solve U given V:**
    Compute $\tau_U^k$ using Theorem 2
    Update $\mathbf{U}^k$ using Equation (4.4)
    **Solve V given U:**
    Compute $\tau_V^k$ using Theorem 3
    Update $\mathbf{V}^k$ using Equation (4.7)
    $k = k + 1$
**end**
**return** $\{U^k, V^k\}$

**Algorithm 2:** Pseudocode for GSpartan framework

**Theorem 4.** *Let* $\{\mathbf{U}^k, \mathbf{V}^k\}$ *be the sequence generated by Algorithm 2 with* $0 \leq \omega^k \leq \delta_\omega \sqrt{\frac{\tau^{k-2}}{\tau^{k-1}}}$ *for* $\delta_\omega < 1$. *Then the sequence of* $\{\mathbf{U}^k, \mathbf{V}^k\}$ *will converge.*

The proof of convergence given by Theorem 4 can be found in Lemma 2.2 of [112].

## 4.3 Experimental Evaluation

This section presents the experiment results to evaluate the effectiveness of the proposed GSpartan framework.

### 4.3.1 Dataset Description

We evaluated the performance of GSpartan on climate data from 37 randomly chosen weather stations in Canada[2]. We use monthly precipitation data from the weather stations as the response variable. The precipitation data spans a 40-year period from January, 1961 to December, 2000. The predictor variables for building the local models were obtained from NCEP-reanalysis[3] data, which is a coarse-scale global environmental data that integrates observations with output from a

[2] `http://climate.weather.gc.ca/`

[3] `http://www.cccsn.ec.gc.ca/?page=pred-hadcm3`

numerical weather prediction model. There are 26 predictor variables, including mean temperature at 2 meters, mean sea level pressure, 500 hPa geopotential height, and near surface relative humidity[4]. We deseasonalize the precipitation time series by subtracting each monthly values with the average value for that month over the entire 40 year period. We then created multiple versions of the training set for each location by varying the length of the training period from 1 to 30 years. For example, the first version uses monthly precipitation data from 1961 for training and the remaining 39 years for testing while the last version uses the first 30 years of monthly precipitation for training and the remaining 10 years for testing.

### 4.3.2 Baseline Methods

We compared the performance of GSpartan against the following baseline:

- **LASSO**: We applied LASSO regression to the data set at each location independently. The Lasso results serve as a baseline for single-task learning.

- **MRMTL**: The mean regularized MTL (MRMTL) is an algorithm developed in [38] based on the assumption of shared common parameters among task models. Specifically, the objective function of MRMTL is given by

$$\min_{W} \sum_{s=1}^{S} \|X_s \mathbf{w}_s - \mathbf{y}_s\|_2^2 + \rho_1 \|W\|_1 + \rho_2 \sum_{s=1}^{S} \left\| \mathbf{w}_s - \frac{1}{S} \sum_{i=1}^{S} \mathbf{w}_i \right\|_2^2$$

  We use the MRMTL implementation given in the MALSAR software package [125]. Note that instead of using $\ell_2$ norm on $W$ as the original work in [38], we use $\ell_1$ norm on $W$ for a fair comparison.

- **SLMTL**: This is an MTL algorithm proposed in [24], which assumes that the tasks are related using an incoherent rank-sparsity structure. Unlike GSpartan, SLMTL does not explicitly consider the relationships among tasks (e.g., spatial autocorrelation between locations). The

[4] A complete list of the features and their description is available at `http://www.cccsn.ec.gc.ca/?page=pred-help`

objective function for SLMTL is given by [24],

$$\min_{W} \sum_{s=1}^{S} \|X_s \mathbf{w}_s - \mathbf{y}_s\|_2^2 + \gamma \|P\|_1$$
$$\text{s.t. } W = P + Q, \|Q\|_* \leq \tau$$

where $W \in \mathscr{R}^{d \times S}$ and $W = [\mathbf{w}_1, ..., \mathbf{w}_S]$. We use the SLMTL implementation provided by the MALSAR software package [125] for our experiments.

In addition to the three baseline algorithms, we also investigate the following two variants of GSpartan.

- **GSpartan-NTR**: In this variant, we remove the graph Laplacian regularizer from the objective function given in Equation (4.1). This allows us to evaluate the importance of incorporating spatial autocorrelation into the framework.

$$\begin{aligned} \min_{W,U,V} \quad & \frac{1}{2} \sum_{s=1}^{S} \|X_s \mathbf{w}_s - \mathbf{y}_s\|_2^2 + \lambda_1 \|V\|_1 + \lambda_2 \|U\|_1 \\ s.t. \quad & V \succeq 0, W = UV \end{aligned}$$

- **GSpartan-norm**: In [124] a normalized graph Laplacian regularizer was used to facilitate transfer of information:

$$\sum_{i,j=1}^{S} A_{i,j} \left\| \frac{1}{\sqrt{D_{i,i}}} \mathbf{w}_i - \frac{1}{\sqrt{D_{j,j}}} \mathbf{w}_j \right\|_2^2.$$

We will compare the normalized graph Laplacian against the unnormalized one used in GSpartan.

### 4.3.3 Task relationship matrix

We use the inverse of a modified variogram measure to estimate the spatial autocorrelation between locations. Variogram is a measure developed in spatial statistics to determine the spatial dependence between a pair of locations [30]. The measure is computed based on the variance of

the difference in field values for two locations:

$$\mathbf{A}_{i,j} = \begin{cases} 1 & \text{if } i = j \\ \frac{1}{var(\mathbf{y}_i - \mathbf{y}_j)} & \text{otherwise} \end{cases}$$

where $var(z)$ denote variance of $z$. Since we have time series data at each location, we compute $var(\mathbf{y}_i - \mathbf{y}_j)$ using monthly precipitation from the first year (1961).

### 4.3.4 Experimental Results

We evaluated the performance of various methods on the test set in terms of their root-mean-square-error (RMSE):

$$R = \sqrt{\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2}$$

where $n$ is the number of test points. Figure 4.1 compares the RMSE for different methods when applied to the first version of the data set (which has 1 year of data for training and 39 years of data for testing). The horizontal axis of the plot corresponds to indices for the 37 weather stations. The results suggest that GSpartan outperforms other baselines for most of the stations. In fact, looking at Table 4.1, which summarizes the number of wins achieved by each method compared to others, GSpartan outperforms other baselines in at least 32 (86.5%) out of 37 stations. Furthermore, by comparing LASSO against other methods, we observe that MTL is generally better than STL especially when there are limited training data available.

To investigate the strength of GSpartan, Figure 4.2 compares its RMSE against other variants of GSpartan. Observe that GSpartan performs no worse than its variants for most of the stations and is significantly better in at least 8 of the stations. Furthermore, the results in Table 4.1 suggest that GSpartan outperforms GSpartan -NTR in all 37 stations, which demonstrates the importance of incorporating spatial autocorrelation into the geospatio-temporal MTL framework. In addition, since GSpartan-NTR outperform other MTL methods in at least 27 stations, this shows the importance of using low-rank representation for modeling the data.

The previous results were obtained using a data set with limited training examples (1 year for training and 39 years for testing). We next investigate the relative performance of GSpartan against

Table 4.1: Win-loss table comparing performance of various methods when applied to the data set with limited training examples (1 year of training data and 39 years of test data).

| | GSpartan | GSpartan-norm | GSpartan-NTR | MRMTL | SLMTL | LASSO |
|---|---|---|---|---|---|---|
| GSpartan | - | 32 | 37 | 33 | 32 | 36 |
| GSpartan-norm | 5 | - | 20 | 31 | 27 | 34 |
| GSpartan-NTR | 0 | 17 | - | 31 | 27 | 34 |
| MRMTL | 4 | 6 | 6 | - | 6 | 13 |
| SLMTL | 5 | 10 | 10 | 31 | - | 34 |
| LASSO | 1 | 3 | 3 | 24 | 7 | - |

other methods as the training set size increases from 1 to 10 years. Specifically, we compare the percentage of stations in which the RMSE for GSpartan is lower than that for other methods. For example, if GSpartan outperforms another method in 32 out of 37 stations, the ratio is 0.865. Figure 4.3 shows the results when the training set size increases. The horizontal axis of the plot corresponds to the index of the data set (which is equivalent to number of training years), while the vertical axis corresponds to the GSpartan outperform ratio. The result shows that, on average, GSpartan outperforms the baseline methods for more than 75% of the stations, and is even higher

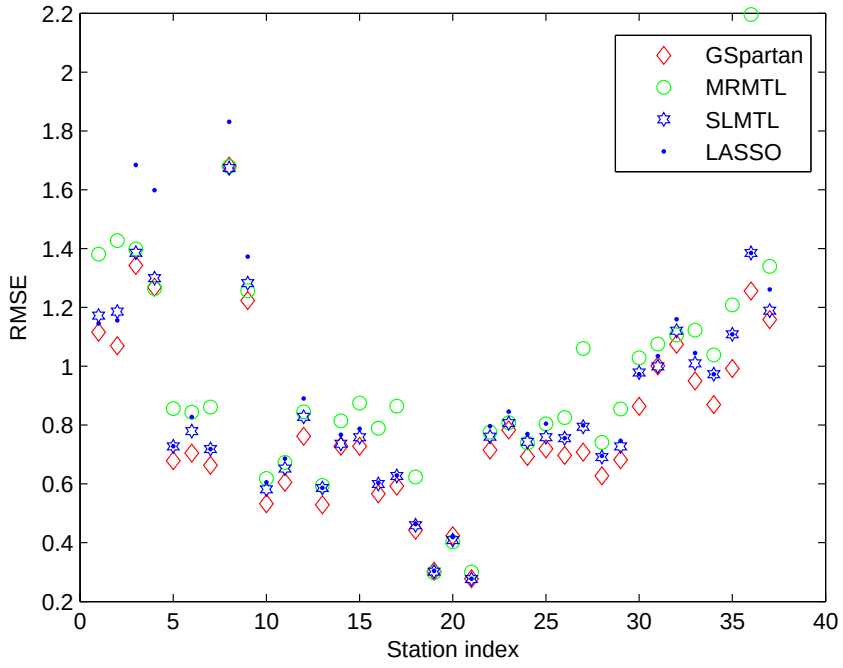



Figure 4.1: Comparison of GSpartan against three baseline methods

when there are fewer training examples. This confirms our hypothesis that GSpartan can effectively train local models when there are limited training examples.

In addition, by comparing GSpartan with GSpartan-NTR, we can see that incorporating spatial autocorrelation enhances the performance of GSpartan irrespective of the training set size. By comparing GSpartan against GSpartan-norm, we also see that a normalized graph Laplacian regularizer indeed degrades the performance of GSpartan . This is because the normalization attenuates the spatial autocorrelation among the tasks, which causes the task relationship to be ineffective. Finally, comparing the results for GSpartan against the two baseline MTL algorithms, it appears that when the training set is small, GSpartan outperforms both MRMTL and SLMTL. However, with increasing training set size, both MRMTL and SLMTL perform better than GSpartan. One possible explanation is that, when there are enough labeled examples available at each station, incorporating the spatial autocorrelation information (which was computed using the first year training data only) might adversely affect the local models.

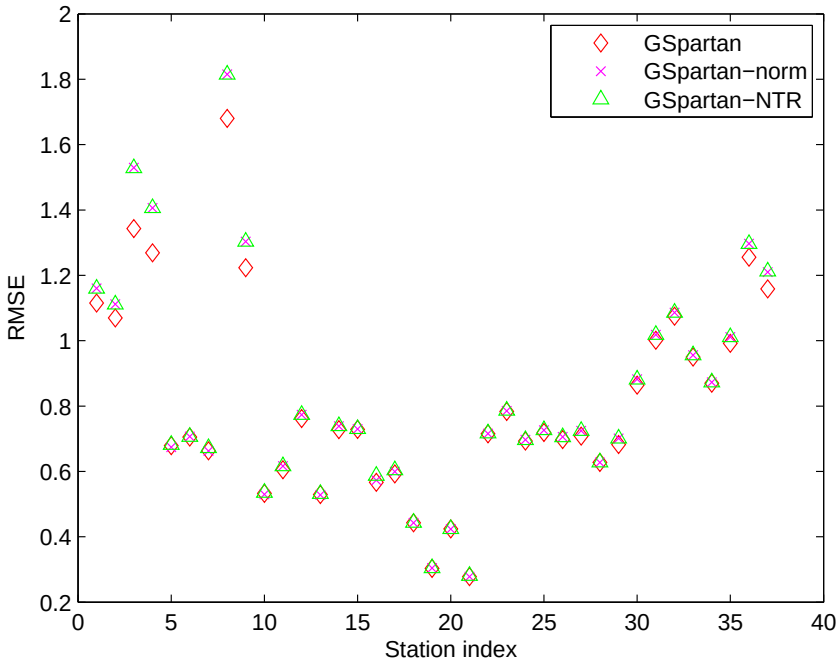



Figure 4.2: Comparison of GSpartan against its variants

### 4.3.5 Sensitivity Analysis

Since there are four parameters ($\lambda_1$, $\lambda_2$, $\lambda_3$ and $k$) that must be tuned in GSpartan, this subsection analyzes the performance of the framework as each parameter is varied. For this experiment, we use the data set with 30 years of training and 10 years of testing. The results using other data sets are quite similar, so we omit them due to lack of space. Figure 4.4 shows the results of our experiment. The horizontal axis corresponds to each index location while the vertical axis represents RMSE values. The results from the figure show that GSpartan is not sensitive to changes in $\lambda_1$ and $\lambda_2$ for all 37 locations (see Figures 4.4a and 4.4b). Furthermore, Figure 4.4c showed that smaller values of $\lambda_3$ should be preferred. Figure 4.4d showed that GSpartan is not that sensitive to $k$ for many locations. However, for those locations where $k$ is sensitive, a small value of $k$ tends to produce lower RMSE.

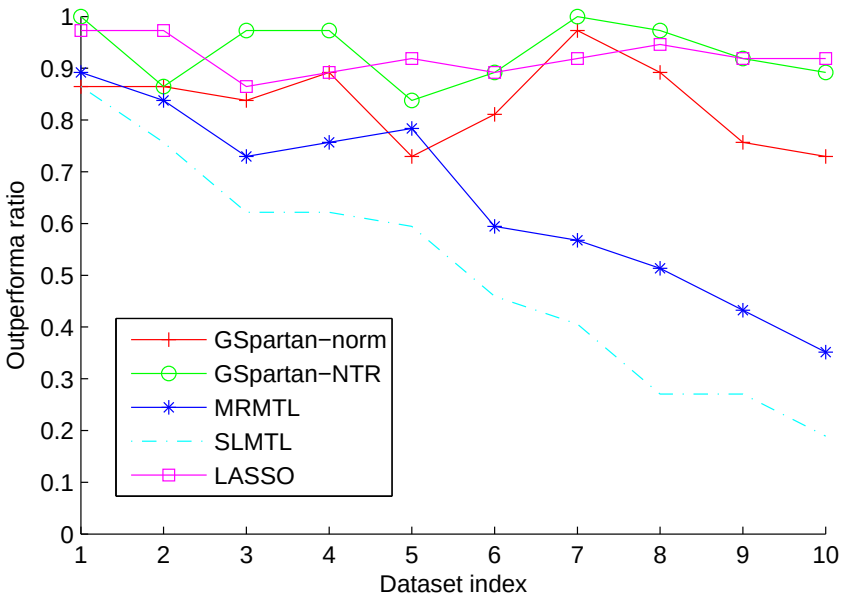
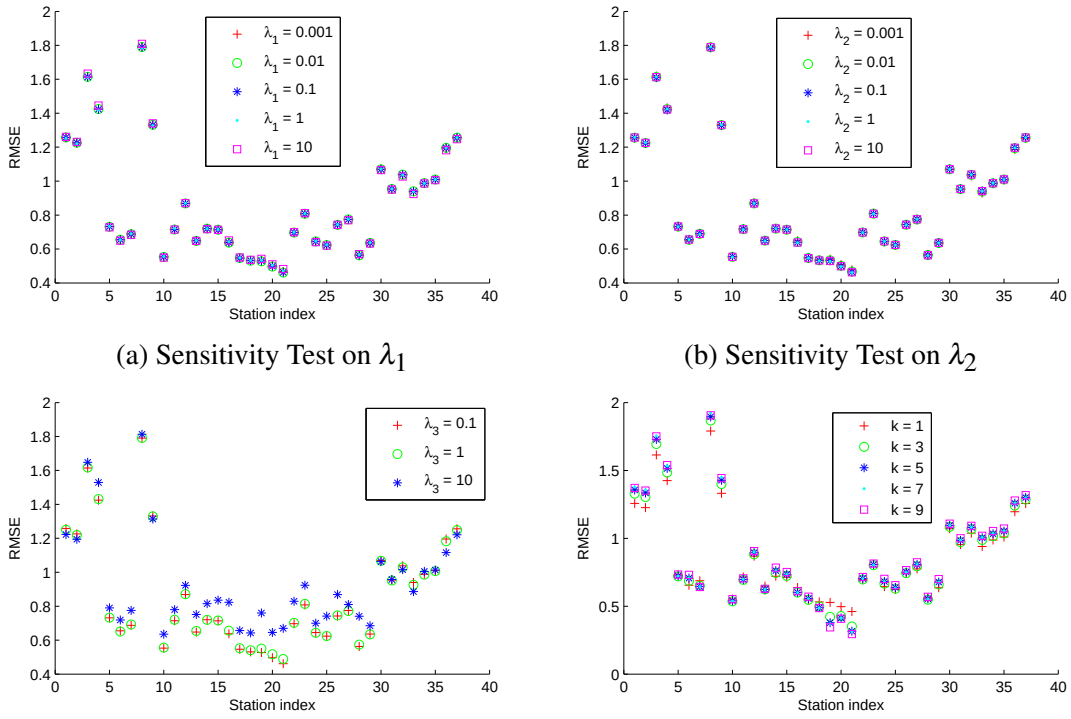



Figure 4.3: Performance comparison between GSpartan against baseline methods as training set size increases.

## 4.4 Conclusion

This chapter presents a novel geospatio-temporal multi-task learning framework called GSpartan for multi-location prediction. GSpartan assumes that the local models share a common low-rank representation. The framework also enables domain-specific constraints such as spatial auto-correlation to be integrated into its formulation. Experimental results on a real world climate data set showed that the proposed framework outperformed other baseline algorithms especially when there are limited training examples available at each location.




(a) Sensitivity Test on $\lambda_1$

(b) Sensitivity Test on $\lambda_2$

(c) Sensitivity Test on $\lambda_3$

(d) Sensitivity Test on $k$

Figure 4.4: Results of sensitivity analysis on $\lambda_1$, $\lambda_2$, $\lambda_3$, and $k$ for GSpartan. The horizontal axis represents the index of a station and the vertical axis corresponds to the RMSE value.

# CHAPTER 5

# WEIGHTED INCREMENTAL SPATIO-TEMPORAL MULTI-TASK LEARNING VIA TENSOR DECOMPOSITION

[1] Predictive modeling of geospatio-temporal data is an important task for many application domains, such as climatology [1, 90, 76], medicine [32], and crop sciences [16]. Such a task typically requires making robust predictions of a target variable at multiple geo-referenced locations based on their historical observation data and other predictor variables. For example, climate scientists are interested to obtain projections of the future climate for multiple locations by downscaling the coarse-scale outputs from regional or global climate models as predictor variables. The multi-location prediction problem can be naturally cast into a multi-task learning (MTL) framework, in which the time series prediction at each location can be regarded as a single learning task. Recent studies [109] have demonstrated the merits of performing a joint learning of the models for multi-location predictions using MTL instead of learning the model at each location independently.

While there have been several previous studies on modeling the predictions at various locations by taking into account the spatial autocorrelation [109] or spatial smoothness of the predictions [91], these methods are often developed for batch learning, thus hindering their applicability to large-scale spatio-temporal data. In addition, as many of the previous works have focused primarily on improving prediction accuracy, their resulting models are often too complicated for interpretation by the domain experts. Incorporating known patterns that drive the variability of the spatio-temporal data into a predictive modeling framework is also non-trivial. For example, it is well-known that the climate variability at a location can be influenced by broad-scale teleconnection patterns such as El Niño (see Fig. 5.1). How to seamlessly integrate such patterns into the predictive modeling framework and derive new, previously unknown patterns that could capture other variability in the spatio-temporal data are challenges that have not been sufficiently addressed in the literature [90, 76].

[1] This paper is based on the previous publication [110].

To overcome these challenges, this chapter presents a novel, incremental spatio-temporal learning algorithm called WISDOM (**W**eighted **I**ncremental **S**patio-temporal Multi-task Learning via Tensor **D**ec**om**position) for multi-location prediction. The algorithm represents the spatio-temporal data as a third-order tensor, where the dimensions (modes) of the tensor represent the temporal, spatial, and predictor variables of the data. By performing tensor decomposition, the latent factors that characterize the variability of the data along each of the three dimensions can be identified. For climate data, known temporal patterns such as El Niño can be directly integrated as a constraint on one of the temporal latent factors of the spatio-temporal tensor. Sparsity-inducing norms can also be added as additional constraints to avoid model overfitting and enhance model interpretability by the domain experts.

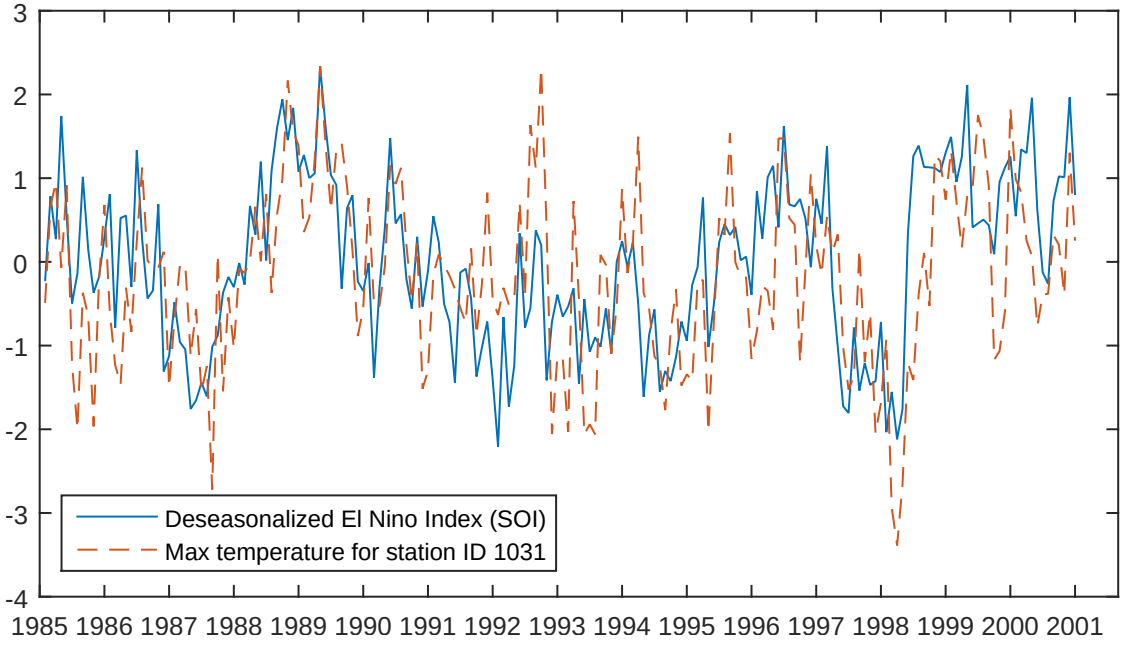



Figure 5.1: The standardized monthly maximum temperature of a weather station in French Polynesia, which correlates strongly with the deseasonalized El-Nino Southern Oscillation Index.

Our proposed tensor decomposition approach is supervised in that the latent factors of the tensor are estimated jointly with the parameters of the prediction models in a unified learning framework. A unique aspect of our formulation is that it constructs two types of prediction models—spatial and temporal—by regressing on the spatial and temporal latent factors inferred from the data. This is a significant departure from conventional spatio-temporal prediction ap-

proaches [118, 91, 109], which typically learns a temporal model for each location from the historical observations and adds spatial constraints to guide the learning algorithm. An alternative approach is to develop a spatial prediction model such as Gaussian Markov Random Field and kriging that considers the spatial distribution of the target variable, where temporal dependencies are used to compute the covariance function of the model. In both types of approaches, the model is trained on one of the dimensions (space or time) while the other dimension is used as side information that constrains the modeling process. Instead, our formulation enables both space and time to be treated equally as it explicitly trains prediction models from both spatial and temporal latent factors. To make a prediction for location $s$ at time $t$, we first apply the spatial model to the spatial latent features for $s$ and the temporal model to the temporal latent features for $t$. We then compute their weighted average to determine the final prediction.

Another challenge is that the spatio-temporal data for many applications often grow rapidly over space and time. The prediction models have to be re-trained whenever new observation data become available, either for a new location (e.g., data from a newly deployed sensor) or as time progresses (when labeled data from the most recent time period are available to verify earlier predictions). Instead of performing the joint tensor factorization and model building steps repeatedly from scratch, which is computationally prohibitive due to the time and memory constraints, it would be desirable to develop a framework that can gradually update its previous latent factors and model parameters based on the newly observed data. Thus, we develop an incremental learning algorithm called WISDOM to solve the optimization problem associated with our proposed formulation.

In short, the main contributions of the chapter are as follows:

1. A supervised tensor factorization framework is presented for spatio-temporal predictive modeling. The framework is unique in that it constructs both spatial and temporal prediction models from the data and can incorporate known patterns from the domain.

2. A scalable algorithm called WISDOM is developed to effectively solve the optimization

problem of the proposed framework. The algorithm can be applied to incremental learning over space, time, or both when new observation data become available.

3. The effectiveness of the proposed framework for multi-location time series prediction is demonstrated on a large-scale global climate data.

## 5.1 Preliminaries

We begin with the notations used in the chapter. A scalar is denoted by a lowercase letter such as $c$ whereas a vector is denoted by a boldface lowercase letter such as $\mathbf{x}$. We denote a matrix by a boldface capital letter, such as $\mathbf{A}$ and a tensor by a boldface Euler script letter, such as $\mathscr{X}$. The symbol ":" is used to denote sub-arrays within a matrix or tensor, e.g., $\mathbf{A}_{:,i}$ denote the $i$-th column of matrix $\mathbf{A}$.

A tensor is a multidimensional array, whose order refers to the number of dimensions (or modes). Each dimension in the tensor can be referred to by its index. A **fiber** of a tensor is a vector obtained by fixing all the indices of the tensor except for one of them. For example, in a 3-dimensional tensor $\mathscr{X}$, $\mathscr{X}_{i,:,j}$ is the mode-2 fiber, which is obtained by fixing the mode-1 index to $i$ and mode-3 index to $j$. A **slice** of a tensor refers to a matrix obtained by fixing all but two of the indices of the tensor. For example, $\mathscr{X}_{:,:,i}$ is the $i$-th mode-3 slice of the tensor $\mathscr{X}$ obtained by setting its mode-3 index to $i$.

**Mode-$n$ matricization** is the process of reordering the elements of a tensor $\mathscr{X} \in \Re^{p_1 \times \ldots \times p_N}$ into a matrix $\mathbf{X}_{(n)} \in \Re^{p_n \times q_n}$, where $q_n = \Pi_{k \neq n} p_k$. The mode-$n$ matricization is obtained by arranging the mode-$n$ fibers of the tensor so that each of them is a column of $\mathbf{X}_{(n)}$. This process is also known as mode-$n$ unfolding.

The **mode-$n$ product** of a tensor $\mathscr{X} \in \Re^{p_1 \times \ldots \times p_N}$ with a matrix $\mathbf{A} \in \Re^{q \times p_n}$ is defined as

$$(\mathscr{X} \times_n \mathbf{A})_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{p_n} \mathscr{X}_{i_1,\ldots,i_N} \mathbf{A}_{j,i_n}$$

which results in a new tensor of dimensions $p_1 \times \ldots \times p_{n-1} \times q \times p_{n+1} \times \ldots \times N$. Furthermore, if $\mathscr{Y} = \mathscr{X} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \ldots \times_N \mathbf{A}^{(N)}$ is an $N$th-order tensor, then $\mathbf{Y}_{(n)} = \mathbf{A}^{(n)} \mathbf{X}_{(n)} \Big( \mathbf{A}^{(N)} \otimes \ldots \otimes$

$\mathbf{A}^{(n+1)} \otimes \mathbf{A}^{(n-1)} \otimes \ldots \otimes \mathbf{A}^{(1)} \Big)^T$ [61], where $\otimes$ denotes the Kronecker product.

The **Khatri-Rao product** of two matrices is equivalent to applying a Kronecker product columnwise to the matrices. For example, given matrices $\mathbf{A} \in \Re^{N \times K}$ and $\mathbf{B} \in \Re^{M \times K}$, then their Khatri-Rao product is given by:

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

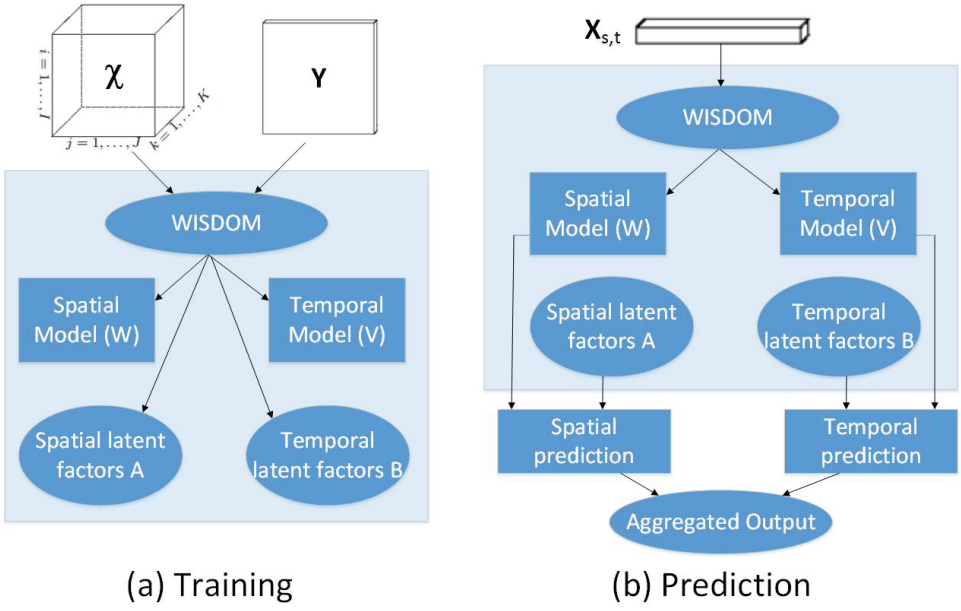



Figure 5.2: Overview of the proposed WISDOM framework.

## 5.2 WISDOM: An Incremental Spatio-Temporal Multi-Task Learning Framework

Let $\mathscr{D} = (\mathscr{X}, \mathbf{Y})$ be a spatio-temporal data set, where $\mathscr{X} \in \Re^{S \times T \times d}$ denote the spatio-temporal tensor of predictor variables, $\mathbf{Y} \in \Re^{S \times T}$ denote the response variable for all the locations, $S$ is the number of locations, $T$ is the length of the time series, and $d$ is the number of predictor variables. For incremental learning, the data is assumed to be periodically augmented with a new data chunk, $(\mathscr{X}_{\text{new}}, \mathbf{Y}_{\text{new}})$, where $\mathscr{X}_{\text{new}} \in \Re^{S \times 1 \times d}$ and $\mathbf{Y}_{\text{new}} \in \Re^{S \times 1}$ if the data is for a new time period or $\mathscr{X}_{\text{new}} \in \Re^{1 \times T \times d}$ and $\mathbf{Y}_{\text{new}} \in \Re^{1 \times T}$ if the data is from a new location.

### 5.2.1 Spatio-temporal Predictive Models

A standard approach to address the spatio-temporal prediction problem is to train a temporal prediction 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 $s$. Note that the temporal models can be trained independently or jointly using a multi-task learning approach such as [109] to predict values of the response variable at a future time $t$. In the latter case, the spatial information is typically used as constraints [91, 109] to guide the training of the temporal prediction model. Alternatively, one could also train a spatial prediction model for 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 $s$.

The framework proposed in this study is novel in that it simultaneously learns the temporal and spatial prediction models using the latent factors derived from the spatio-temporal tensor, as shown in Fig. 5.2. Unlike other previous approaches, it builds separate models from the spatial and temporal latent factors and combines the output of both models to obtain the final prediction. Assuming a linear model, the framework predicts the value for a location $s$ at time $t$ as a weighted linear combination of its spatial and temporal models, i.e.,

$$\hat{y}_{s,t} \quad = \quad \mathbf{x}_{s,t}^T(\sum_{k}^{K}\mathbf{A}_{s,k}\mathbf{w}_k + \sum_{k}^{K}\mathbf{B}_{t,k}\mathbf{v}_k) \tag{5.1}$$

where $\mathbf{A}_{s,k}$ denote the weight of the $k$-th spatial latent feature for location $s$, $\mathbf{B}_{t,k}$ denote the weight of the $k$-th temporal latent feature for time $t$, $\mathbf{w}_k$ and $\mathbf{v}_r$ are the parameters for the corresponding spatial and temporal prediction models of the $k$-th latent feature. The model parameters can be represented in matrix form as $\mathbf{W} = [\mathbf{w}_1^T;\mathbf{w}_2^T;...;\mathbf{w}_K^T] \in \Re^{K\times d}$ and $\mathbf{V} = [\mathbf{v}_1^T;\mathbf{v}_2^T;...;\mathbf{v}_K^T] \in \Re^{K\times d}$ and are estimated by optimizing the following joint objective function:

$$\min_{\mathbf{W},\mathbf{V}} \sum_{s}^{S}\sum_{t}^{T}\mathscr{L}(\mathbf{x}_{s,t},\mathbf{W},\mathbf{V},y_{s,t}) + \Omega_m(\mathbf{W},\mathbf{V}) \tag{5.2}$$

where $\mathscr{L}(\mathbf{x}_{s,t},\mathbf{W},\mathbf{V},y_{s,t})$ is the loss function and $\Omega(\mathbf{W},\mathbf{V})$ is the regularizer for the model parameters. In this work, we consider a squared loss function, $\mathscr{L}(\mathbf{x}_{s,t},\mathbf{W},\mathbf{V},y_{s,t}) = (\hat{y}_{s,t} - y_{s,t})^2$, and define $\Omega_m(\mathbf{W},\mathbf{V}) = \|\mathbf{W}\|_1 + \|\mathbf{V}\|_1$ to ensure sparsity of the models.

### 5.2.2 Supervised Tensor Decomposition

The formulation presented in Eq.(5.2) requires knowledge about the latent factors of the spatio-temporal tensor. These latent factors are represented by the loading matrices $\mathbf{A}$ and $\mathbf{B}$, which can be derived from the data using tensor decomposition techniques. There are two standard approaches for decomposing a tensor, namely, Tucker and CANDECOMP/PARAFAC (CP) decompositions [61]. Tucker decomposition factorizes a tensor into a core tensor and a product of its loading matrices along each mode. Though it provides a more general representation, the latent factors are harder to be interpreted as the number of latent factors along each mode does not have to be identical. In contrast, CP decomposition factorizes a tensor into a sum of rank-1 tensors, i.e., $\mathscr{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{k=1}^{K} \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k$, 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 loading matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, respectively.

In this work, we apply CP decomposition on the spatio-temporal tensor. Let $\mathscr{X} \in \Re^{S \times T \times d}$ be a spatio-temporal tensor, where the $(s,t)$-th mode-3 fiber of $\mathscr{X}$ corresponds to the feature vector for location $s$ at time $t$, i.e., $\mathscr{X}_{s,t,:} = \mathbf{x}_{s,t}$. To estimate the latent factors, the objective function for CP decomposition can be written as follows:

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

where $\| \mathscr{X} \|_F = \sqrt{\sum_{ijk} \mathscr{X}_{ijk}^2}$ is the Frobenius norm of the tensor $\mathscr{X}$ and $\Omega_d(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is a regularization term for the loading matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$. To ensure model sparsity, the following regularization penalty can be 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 our spatio-temporal MTL framework

can be stated as follows:

$$\begin{aligned} & \min_{\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C}} \mathscr{F}(\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C}) \\ = \; & \frac{1}{2}\sum_{s}^{S}\sum_{t}^{T}(\mathbf{x}_{s,t}^{T}(\mathbf{W}^{T}\mathbf{A}_{s}+\mathbf{V}^{T}\mathbf{B}_{t})-y_{s,t})^{2} \\ + \; & \frac{\lambda}{2}\|\mathscr{X}-[\![\mathbf{A},\mathbf{B},\mathbf{C}]\!]\|_{F}^{2}+\beta\|[\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C}]\|_{1} \end{aligned} \tag{5.3}$$

where we have used $\|[\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C}]\|_1$ to denote the $\ell_1$ norm for **W**, **V**, **A**, **B** and **C**, respectively.

### 5.2.3 WISDOM Algorithm

As the size of many spatio-temporal data sets can be very large, efficient algorithms are needed to learn the spatio-temporal predictive models of the data. To optimize the objective function in Eq.(5.3), we develop an incremental learning algorithm called WISDOM to learn the model parameters as well as the latent factors of the tensor. While most incremental learning algorithms consider only updating the parameters over time, for spatio-temporal data, there is also a need to update the parameters over space. For example, new observation data may be available from sensors deployed at a new location or when a scientific research is expanded to include a new study region. To support this, we present two implementations of WISDOM—one for incremental learning over space and the other for incremental learning over time. A hybrid approach that combines both strategies can be easily developed when new observations can be generated over space and time.

For incremental learning, our goal is to adapt the existing models without rebuilding the model from scratch as new data 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. We reformulate the optimization problem for incremental learning as follows:

$$\begin{aligned} & \min_{\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C}} \mathscr{Q}(\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C},\tilde{\mathbf{W}},\tilde{\mathbf{V}},\tilde{\mathbf{A}},\tilde{\mathbf{B}},\tilde{\mathbf{C}}) \\ = \; & \mathscr{F}(\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C})+\Gamma(\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C},\tilde{\mathbf{W}},\tilde{\mathbf{V}},\tilde{\mathbf{A}},\tilde{\mathbf{B}},\tilde{\mathbf{C}}), \end{aligned}$$

where $\tilde{\mathbf{W}}$, $\tilde{\mathbf{V}}$, $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, and $\tilde{\mathbf{C}}$ are the previous values before the update, $\mathscr{F}(\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C})$ is given by Eq. (5.3) and

$$\begin{aligned} & \Gamma(\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B},\mathbf{C},\tilde{\mathbf{W}},\tilde{\mathbf{V}},\tilde{\mathbf{A}},\tilde{\mathbf{B}},\tilde{\mathbf{C}}) \\ = \; & \|\mathbf{W}-\tilde{\mathbf{W}}\|_F^2 + \|\mathbf{V}-\tilde{\mathbf{V}}\|_F^2 + \|\mathbf{A}-\tilde{\mathbf{A}}\|_F^2 + \|\mathbf{B}-\tilde{\mathbf{B}}\|_F^2 + \|\mathbf{C}-\tilde{\mathbf{C}}\|_F^2. \end{aligned} \tag{5.4}$$

#### 5.2.3.1 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, $\{\mathbf{x}_{S+1,t_0}, \mathbf{x}_{S+1,t_0+1}, \cdots, \mathbf{x}_{S+1,T}\}$. For brevity, we denote the feature vectors for the new location as $\mathscr{X}_{S+1}$, 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:

$$\begin{aligned} & \min_{\mathbf{W},\mathbf{V},\mathscr{V},\mathbf{A}_{S+1},\mathbf{B},\mathbf{C}} \mathscr{Q}(\mathbf{W},\mathbf{V},\mathbf{A}_{S+1},\mathbf{B},\mathbf{C},\tilde{\mathbf{W}},\tilde{\mathbf{V}},\tilde{\mathbf{B}},\tilde{\mathbf{C}}) \\ = \; & \frac{1}{2}\sum_{t=t_0}^{T}\left[\mathbf{x}_{S+1,t}^T(\mathbf{W}^T\mathbf{A}_{S+1} + \mathbf{V}^T\mathbf{B}_t) - y_{S+1,t}\right]^2 \\ & + \frac{\lambda_1}{2}\|\mathscr{X}_{S+1} - [\![\mathbf{A}_{S+1}^T,\mathbf{B},\mathbf{C}]\!]\|_F^2 \\ & + \frac{\eta_1}{2}\left[\|\mathbf{W}-\tilde{\mathbf{W}}\|_F^2 + \|\mathbf{V}-\tilde{\mathbf{V}}\|_F^2 + \|\mathbf{B}-\tilde{\mathbf{B}}\|_F^2 + \|\mathbf{C}-\tilde{\mathbf{C}}\|_F^2\right] \\ & + \beta_1\|\mathbf{W},\mathbf{V},\mathbf{A}_{S+1},\mathbf{B},\mathbf{C}\|_1 \end{aligned} \tag{5.5}$$

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 $\eta_1$ determines the extent to which the previous model parameters should be retained.

We employ an alternating minimization strategy to solve the optimization problem. Since not all terms in the objective function are differentiable, we employ the proximal gradient descent method [84] to solve each subproblem. The method is applicable to non-differentiable objective functions that can be decomposed into a smooth part and a non-smooth part. Let $f(x) = g(x) + h(x)$, where $g(x)$ is a differentiable function and $h(x)$ is a non-differentiable function. For example, the loss function involving the Frobenius norm terms in our objective function is differentiable whereas the sparsity-inducing $L1$-norm terms are non-differentiable. The proximal gradient descent method updates its parameter as following:

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

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 $\ell_1$ norm function is the soft-thresholding operator: $\mathbf{prox}_{\lambda,h}(v) = (v-\lambda)_+ - (-v-\lambda)_+$. 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 $\ell_1$ norm) to determine its next value. The step size can be found using a line search algorithm. In the following, we provide the gradient of the objective function for each alternating minimization step.

**I. Solving for $\mathbf{A}_{S+1}$ by fixing W, V, B, C:**

The objective function can be simplified to retain only terms involving $\mathbf{A}_{S+1}$ as follows:

$$\begin{aligned}\min_{\mathbf{A}_{S+1}} \quad & \frac{1}{2}\sum_{t=t_0}^{T}(\mathbf{x}_{S+1,t}^T(\mathbf{W}^T\mathbf{A}_{S+1} + \mathbf{V}^T\mathbf{B}_t) - y_{S+1,t})^2 \\ & + \frac{\lambda_1}{2}\|\mathbf{X}_{S+1(1)} - \mathbf{A}_{S+1}^T(\mathbf{C}\odot\mathbf{B})^T\|_F^2 + \beta_1\|\mathbf{A}_{S+1}\|_1\end{aligned}$$

where $\mathbf{X}_{S+1(1)}$ is the mode-1 unfolding of the tensor $\mathscr{X}_{S+1}$. The gradient on the smooth part of the objective function w.r.t. $\mathbf{A}_{S+1}$ is

$$\begin{aligned}&\sum_{t=t_0}^{T}\left(\mathbf{x}_{S+1,t}^T(\mathbf{W}^T\mathbf{A}_{S+1} + \mathbf{V}^T\mathbf{B}_t) - y_{S+1,t}\right)\mathbf{W}\mathbf{x}_{S+1,t} \\ &- \lambda_1\left([\mathbf{X}_{S+1(1)} - \mathbf{A}_{S+1}^T(\mathbf{C}\odot\mathbf{B})^T](\mathbf{C}\odot\mathbf{B})\right)^T\end{aligned}$$

**II. Solving for B by fixing $\mathbf{A}_{S+1}$, W, V, C:**

The terms in the objective function involving matrix $\mathbf{B}$ include

$$\begin{aligned}\min_{\mathbf{B}} \quad & \frac{1}{2}\sum_{t=t_0}^{T}(\mathbf{x}_{S+1,t}^T(\mathbf{W}^T\mathbf{A}_{S+1}+\mathbf{V}^T\mathbf{B}_t)-y_{S+1,t})^2 \\ & +\frac{\lambda_1}{2}\|\mathbf{X}_{S+1(2)}-\mathbf{B}(\mathbf{C}\odot\mathbf{A}_{S+1}^T)^T\|_F^2 \\ & +\frac{\eta_1}{2}\|\mathbf{B}-\tilde{\mathbf{B}}\|_F^2+\beta_1\|\mathbf{B}\|_1 \end{aligned} \tag{5.6}$$

The gradient on the smooth part of the objective function w.r.t. $\mathbf{B}_t$ is given by

$$\begin{aligned} & \left(\mathbf{x}_{S+1,t}^T(\mathbf{W}^T\mathbf{A}_{S+1}+\mathbf{V}^T\mathbf{B}_t)-y_{S+1,t}\right)\mathbf{V}\mathbf{x}_{S+1,t} \\ + \quad & \lambda_1\left([\mathbf{x}_{S+1,t}^T-\mathbf{B}_t^T(\mathbf{C}\odot\mathbf{A}_{S+1}^T)^T](\mathbf{C}\odot\mathbf{A}_{S+1}^T)\right)^T \\ - \quad & \eta_1(\mathbf{B}_t-\tilde{\mathbf{B}}_t) \end{aligned}$$

**III. Solving for C by fixing $\mathbf{A}_{S+1}$, B, W, V:**

Similarly, we can simplify the objective function to include only terms involving the matrix $\mathbf{C}$:

$$\begin{aligned}\min_{\mathbf{C}} \quad & \frac{\lambda_1}{2}\|\mathbf{X}_{S+1(3)}-\mathbf{C}(\mathbf{B}\odot\mathbf{A}_{S+1}^T)^T\|_F^2 \\ & +\frac{\eta_1}{2}\|\mathbf{C}-\tilde{\mathbf{C}}\|_F^2+\beta_1\|\mathbf{C}\|_1 \end{aligned} \tag{5.7}$$

The gradient on the smooth part of the objective function w.r.t. $\mathbf{C}$ is given by

$$-\lambda_1\left(\mathbf{X}_{S+1(3)}-\mathbf{C}(\mathbf{B}\odot\mathbf{A}_{S+1}^T)^T\right)(\mathbf{B}\odot\mathbf{A}_{S+1}^T)+\eta_1(\mathbf{C}-\tilde{\mathbf{C}})$$

**IV. Solving for W by fixing V, $\mathbf{A}_{S+1}$, B, C:**

The terms in the objective function involving the model parameter $\mathbf{W}$ is

$$\begin{aligned}\min_{\mathbf{W}} \quad & \frac{1}{2}\sum_{t=t_0}^{T}(\mathbf{x}_{S+1,t}^T(\mathbf{W}^T\mathbf{A}_{S+1}+\mathbf{V}^T\mathbf{B}_t)-y_{S+1,t})^2 \\ & +\frac{\eta_1}{2}\|\mathbf{W}-\tilde{\mathbf{W}}\|_F^2+\beta_1\|\mathbf{W}\|_1 \end{aligned} \tag{5.8}$$

The gradient on the smooth part of the objective function w.r.t. $\mathbf{W}$ is given by

$$\begin{aligned} & \sum_{t=t_0}^{T}\mathbf{x}_{S+1,t}\left(\mathbf{x}_{S+1,t}^T(\mathbf{W}^T\mathbf{A}_{S+1}+\mathbf{V}^T\mathbf{B}_t)-y_{S+1,t}\right)\mathbf{A}_{S+1}^T \\ & +\eta_1(\mathbf{W}^T-\tilde{\mathbf{W}}^T) \end{aligned}$$

**V. Solving for V by fixing W, $\mathbf{A}_{S+1}$, B, C:**

Finally, the objective function can be simplified for terms involving $\mathbf{V}$ as follows:

$$\min_{\mathbf{V}} \quad \frac{1}{2}\sum_{t=t_0}^{T}(\mathbf{x}_{S+1,t}^T(\mathbf{W}^T\mathbf{A}_{S+1}+\mathbf{V}^T\mathbf{B}_t)-y_{S+1,t})^2 + \frac{\eta_1}{2}\|\mathbf{V}-\tilde{\mathbf{V}}\|_F^2+\beta_1\|\mathbf{V}\|_1 \tag{5.9}$$

The gradient on the smooth part of the objective function w.r.t. $\mathbf{V}$ is given by

$$\sum_{t=t_0}^{T}\mathbf{x}_{S+1,t}(\mathbf{x}_{S+1,t}^T(\mathbf{W}^T\mathbf{A}_{S+1}+\mathbf{V}^T\mathbf{B}_t)-y_{S+1,t})\mathbf{B}_t^T + \eta_1(\mathbf{V}-\tilde{\mathbf{V}})$$

#### 5.2.3.2 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. Next, the model parameters and latent factors for other modes are updated when the target variable for the new time period is observed for all the locations.

**Step 1: Updating the temporal latent factor $\mathbf{B}_{T+1}$ before observing target variable.** The objective function for updating the temporal latent factor is given below:

$$\min_{\mathbf{B}_{T+1}} \mathscr{Q}(\mathbf{B}_{T+1}) \;=\; \frac{\lambda_2}{2}\|\mathscr{X}_{T+1}-[\![\mathbf{A},\mathbf{B}_{T+1}^T,\mathbf{C}]\!]\|_F^2 \tag{5.10}$$

Note that the loading matrices $\mathbf{A}$ and $\mathbf{C}$ correspond to the values obtained from the previous update.

**Step 2: Updating model parameters and other latent factors after observing target variable.** After the new observation for target variable at time $T+1$, we can derive the update formula by

optimizing the following objective function:

$$\begin{aligned}
&\min_{\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B}_{T+1},\mathbf{C}} \mathscr{Q}(\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B}_{T+1},\mathbf{C},\tilde{\mathbf{W}},\tilde{\mathbf{V}},\tilde{\mathbf{A}},\tilde{\mathbf{C}}) \\
= \;& \frac{1}{2}\sum_{s}^{S}(\mathbf{x}_{s,T+1}^{T}(\mathbf{W}^{T}\mathbf{A}_{s}+\mathbf{V}^{T}\mathbf{B}_{T+1})-y_{s,T+1})^{2} \\
+ \;& \frac{\lambda_2}{2}\|\mathscr{X}_{T+1}-[\![\mathbf{A},\mathbf{B}_{T+1}^{T},\mathbf{C}]\!]\|_{F}^{2} \\
+ \;& \frac{\eta_2}{2}(\|\mathbf{W}-\tilde{\mathbf{W}}\|_{F}^{2}+\|\mathbf{V}-\tilde{\mathbf{V}}\|_{F}^{2}+\|\mathbf{A}-\tilde{\mathbf{A}}\|_{F}^{2} \\
+ \;& \|\mathbf{C}-\tilde{\mathbf{C}}\|_{F}^{2})+\beta_1(\|\mathbf{W},\mathbf{V},\mathbf{A},\mathbf{B}_{T+1},\mathbf{C}\|_{1}) \qquad (5.11)
\end{aligned}$$

Solving Eq. (5.11) 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.

#### 5.2.3.3 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.

## 5.3 Experimental Evaluation

We applied WISDOM to a global-scale climate data set and compared its performance against several baseline algorithms.

### 5.3.1 Dataset Description

The climate data was obtained from two sources. First, we downloaded the monthly climate observation data from the *Global Surface Summary of Day* (GSOD)[2] website. These monthly values

[2] `https://data.noaa.gov/dataset/global-surface-summary-of-the-day-gsod`

of total precipitation (prcp), maximum (tmax), minimum (tmin), and average (tmean) temperature are used to define the target/response variable for our prediction tasks. We created 4 data sets, one for each response variable, to evaluate the performance of WISDOM. Though the four response variables can be jointly modeled in a multi-task learning framework, this is beyond the scope of the current work.

The second source corresponds to a coarse-scale gridded climate data from NCEP reanalysis[3]. We use the data to define the predictor variables for our climate prediction task. Although there are hundreds of variables available in the NCEP reanalysis data, we selected 13 of them as our predictor variables with the help of our domain expert. A detailed description of the selected features is given in Table 5.1.

GSOD provides climate data from more than 30,000 monitoring sites worldwide, spanning a time period from 1942 to the present time. We use the monthly data from January 1985 to November 2015 (for a total of 371 months) in our experiment. During preprocessing, we remove the sites that have missing values as well as sites that are co-located in the same grid as another previously chosen site, i.e., we restrict each grid to contain only one GSOD site. This reduces the number of sites in our data set to 1,110. The variables are deseasonalized by subtracting each monthly value from the average value of the given month and then standardized to obtain their Z-scores. The dimensionality of the resulting spatio-temporal tensor after preprocessing is $1110 \times 371 \times 13$.

### 5.3.2 Experimental Setup

We use the data from the first 20 years (1985 - 2004) for training and the rest (2005 - 2015) for testing the efficacy of the prediction models. The last 10 years of the training data are used as validation set to determine the model parameters. We set the number of latent factors in WISDOM to $k = 5$ and randomly select 100 sites as our initial starting locations. For incremental learning,

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

Table 5.1: List of predictor variables selected from NCEP reanalysis data.

| Variable | Description |
|---|---|
| cprat.sfc | Monthly mean convective precipitation rate at surface |
| dlwrf.sfc | Monthly mean longwave radiation flux at surface |
| dswrf.sfc | Monthly mean solar radiation flux at surface |
| prate.sfc | Monthly mean of precipitation rate |
| tmax.2m | Monthly mean maximum temperature at 2 m |
| tmin.2m | Monthly mean minimum temperature at 2 m |
| lftx.sfc | Monthly mean surface lifted index |
| omega.sig995 | Monthly mean omega at sigma level 0.995 |
| pr_wrt.eatm | Monthly mean of precipitable water content |
| rhum.sig995 | Monthly mean relative humidity at sigma level 0.995 |
| slp | Sea level pressure |
| thick_1000500 | Monthly mean of thickness for 1000-500mb |
| thick_850500 | Monthly mean of thickness for 850-500mb |

the new observation may come from a new location or for a new time point. In the former case, we perform incremental learning over space for the new location and update the prediction models for other locations. In the latter case, we perform incremental learning over time to update the models for all the locations. The mean absolute error (MAE) for all locations are used as our evaluation metric and we also report the standard deviation of MAE over time.

We compared WISDOM against the following two baseline algorithms. The order in which new observations are introduced over space and time is the same for all the algorithms:

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 [118], 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 compute the model for the new location using linear regression and adds the estimated parameters as a new row in $\mathbf{W}$. The updated $\mathbf{W}$ is then projected into its low-rank matrix representation.

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

3) **WISDOM-S**: This baseline considers only the spatial component of the framework. Specifically, we remove all terms related to $\mathbf{V}$ in Eq. (5.5) and (5.11).

4) **WISDOM-T**: This baseline considers only the temporal component of the framework. We remove all terms related to $\mathbf{W}$ in Eq. (5.5) and (5.11).

### 5.3.3 Comparison against Baseline Methods

We first present the results comparing WISDOM against the two baseline algorithms, STL and ALTO. Table 5.2 shows the average MAE for the various methods whereas Table 5.4 shows the number of locations in which each method outperforms another. The results suggest that WISDOM outperform STL and ALTO in more than 75% of the locations for all 4 data sets evaluated. The percentage is even higher ($> 90\%$) when compared against ALTO on the three temperature data sets. By outperforming STL, this 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[4]: $\mathbf{W}^{(k)} = (1-\alpha)\mathbf{W}^{(k-1)} + \alpha\mathbf{X}\mathbf{Z}^{\dagger}$ [118]. 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 Tables 5.3 and 5.4. Observe that WISDOM and WISDOM-S outperform WISDOM-T on all four data sets, which suggest the importance of incorporating a predictive model from the spatial latent factors. Furthermore, WISDOM performs better than WISDOM-S especially for precipitation prediction. This makes sense as precipitation has less spatial autocorrelation compared to temperature, which is why temporal autocorrelation plays a more significant role in improving its prediction.

Table 5.2: MAE and its standard deviation for WISDOM and other baseline methods

| | tmax | tmin | tmean | prcp |
|---|---|---|---|---|
| WISDOM | **0.4751**$\pm$ 0.0739 | **0.5016** $\pm$ 0.0777 | **0.4438** $\pm$ 0.0798 | **0.5700** $\pm$ 0.1157 |
| STL | 0.5580 $\pm$ 0.0599 | 0.5670 $\pm$ 0.0627 | 0.5233 $\pm$ 0.0606 | 0.6930 $\pm$ 0.0897 |
| ALTO | 0.6824 $\pm$ 0.1448 | 0.6598 $\pm$ 0.1660 | 0.6570 $\pm$ 0.1691 | 0.6087 $\pm$ 0.1197 |

Table 5.3: MAE and its standard deviation for WISDOM and its variants

| | tmax | tmin | tmean | prcp |
|---|---|---|---|---|
| WISDOM | 0.4751 $\pm$ 0.0739 | **0.5016** $\pm$ 0.0777 | 0.4438 $\pm$ 0.0798 | **0.5700** $\pm$ 0.1157 |
| WISDOM-S | **0.4685** $\pm$ 0.1031 | 0.5030 $\pm$ 0.0952 | **0.4380** $\pm$ 0.1065 | 0.5824 $\pm$ 0.1206 |
| WISDOM-T | 0.4832 $\pm$ 0.1233 | 0.5285 $\pm$ 0.1033 | 0.4607 $\pm$ 0.1036 | 0.6075 $\pm$ 0.1221 |

### 5.3.4 Convergence Analysis of WISDOM

To demonstrate its convergence, Fig. 5.3 shows the average MAE of WISDOM for all locations across time. A location is included into the average MAE calculation only after the data for the

[4]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 5.4: Comparison between the number of locations (out of 1,100) in which one method outperforms another

| Variable | | WISDOM | WISDOM-S | WISDOM-T | ALTO | STL |
|---|---|---|---|---|---|---|
| tmax | WISDOM | 0 | 621 | 842 | 1031 | 904 |
| | WISDOM-S | 489 | 0 | 901 | 1036 | 968 |
| | WISDOM-T | 268 | 209 | 0 | 976 | 848 |
| | ALTO | 79 | 74 | 134 | 0 | 163 |
| | STL | 206 | 142 | 262 | 947 | 0 |
| tmin | WISDOM | 0 | 642 | 823 | 1007 | 883 |
| | WISDOM-S | 468 | 0 | 869 | 1016 | 901 |
| | WISDOM-T | 287 | 241 | 0 | 956 | 792 |
| | ALTO | 103 | 94 | 154 | 0 | 192 |
| | STL | 227 | 209 | 318 | 918 | 0 |
| tmean | WISDOM | 0 | 621 | 767 | 1033 | 898 |
| | WISDOM-S | 489 | 0 | 779 | 1044 | 951 |
| | WISDOM-T | 343 | 331 | 0 | 1003 | 869 |
| | ALTO | 77 | 66 | 107 | 0 | 131 |
| | STL | 212 | 159 | 241 | 979 | 0 |
| prcp | WISDOM | 0 | 756 | 946 | 838 | 990 |
| | WISDOM-S | 354 | 0 | 789 | 685 | 997 |
| | WISDOM-T | 164 | 321 | 0 | 651 | 910 |
| | ALTO | 272 | 425 | 495 | 0 | 852 |
| | STL | 120 | 113 | 200 | 258 | 0 |

location becomes available. Although there are some instabilities in its performance during the first 8 years, WISDOM begins to converge after the first 10 years, which is our initial training period, on all four data sets.

### 5.3.5 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. Figure 5.4 and 5.5 show the spatial distribution of the latent factors for prcp. 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 north of China whereas the second latent factor emphasizes more in US and east of China.

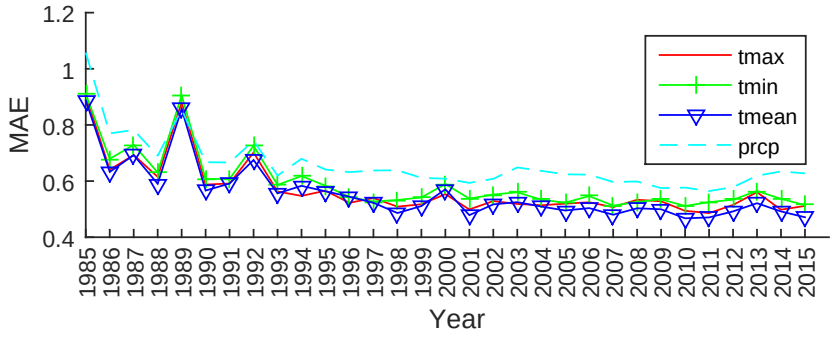


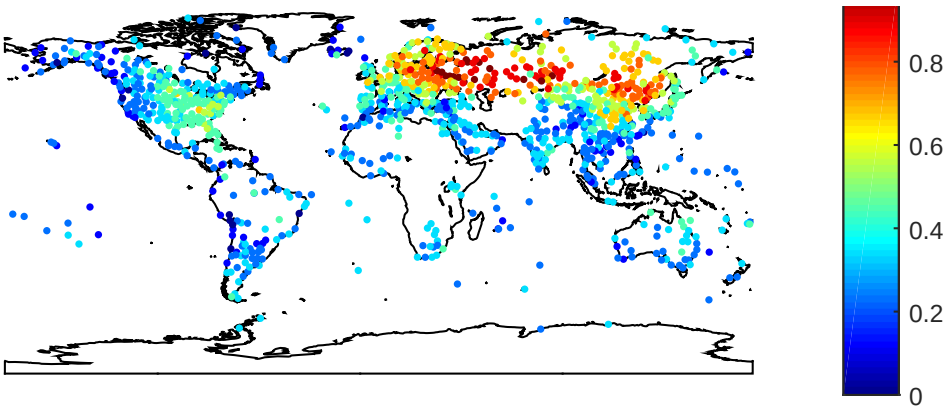

Figure 5.3: Changes in MAE over time for WISDOM

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. Indeed, as shown in Fig 5.6, adding data from new locations helps to improve the MAE of the first 100 randomly chosen locations.

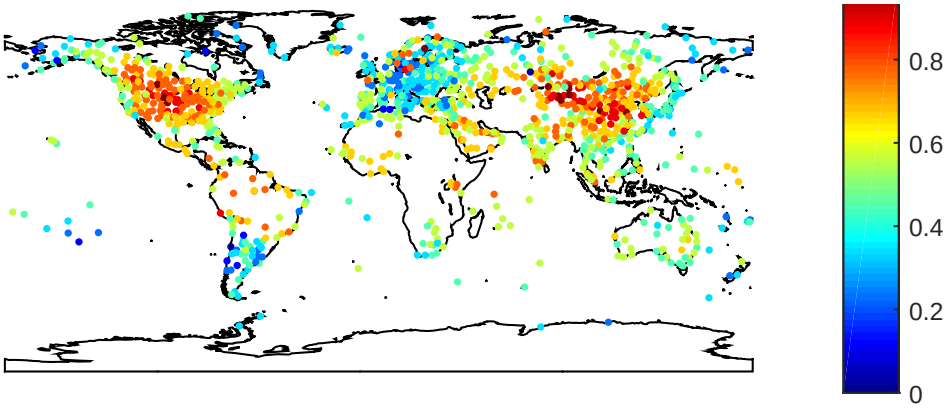
### 5.3.6 Analysis of Temporal Latent Factors

Each temporal latent factor derived by WISDOM can be represented as a time series. To understand their significance, we correlate the temporal latent factors against the known climate indices given in Table 5.5. Fig. 5.7 shows the resulting correlation for the tmean and prcp data sets. Though the temporal latent factors for both 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. 5.8 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

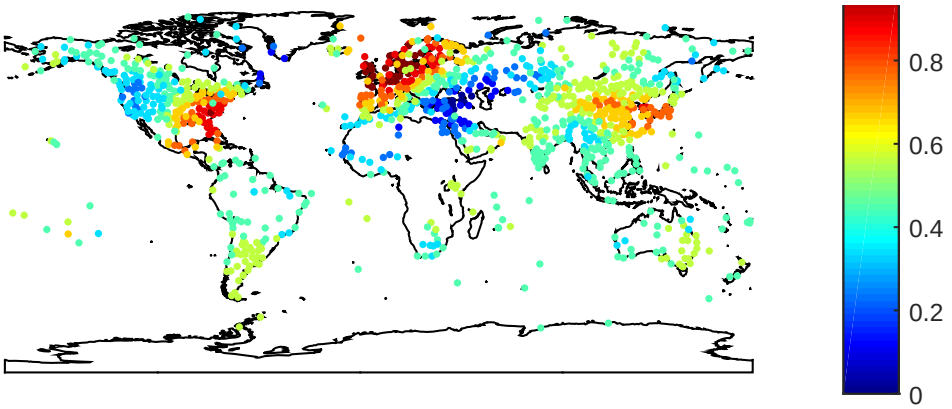
locations, comparable to the known indices. More importantly, as some of the latent factors do not correlate highly with the known indices, this suggests that our framework can potentially discover new indices that capture the climate variability for many locations.




(a) Factor 1




(b) Factor 2



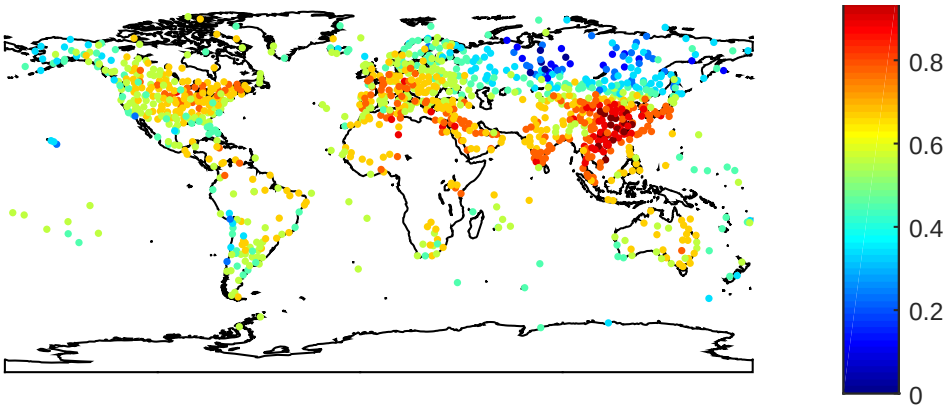

(c) Factor 3

Figure 5.4: Spatial distribution of the spatials factor learned by WISDOM for prcp. (Figure is best viewed in color).

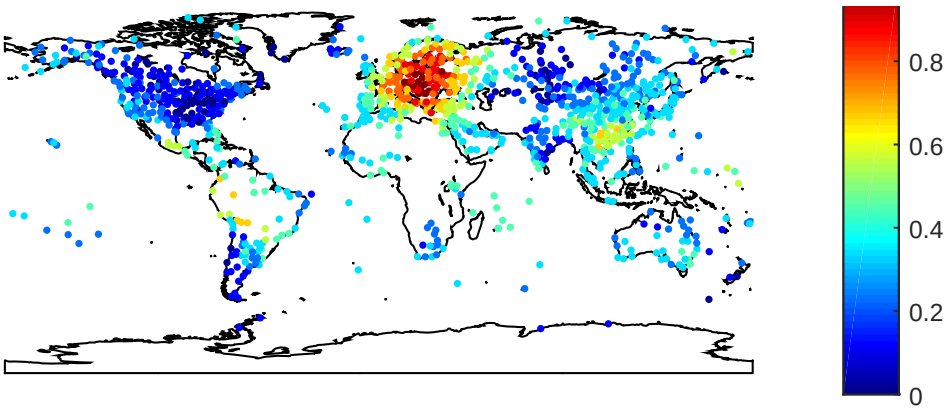
Table 5.5: List of the climate indices used to correlate with the temporal factors learned from WISDOM.

| Climate Index | Description |
|---|---|
| AOI | Arctic Oscillation Index |
| NAO | North Atlantic Oscillation |
| WPI | West Pacific Pattern |
| QBO | Quasi-Biennial Oscillation |
| PDO | Pacific Decadal Oscillation |
| SOI | Southern Oscillation Index |

Surprisingly, 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

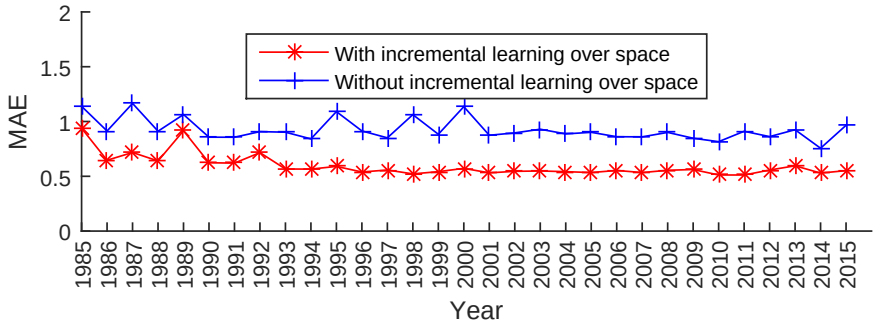



(a) Factor 4

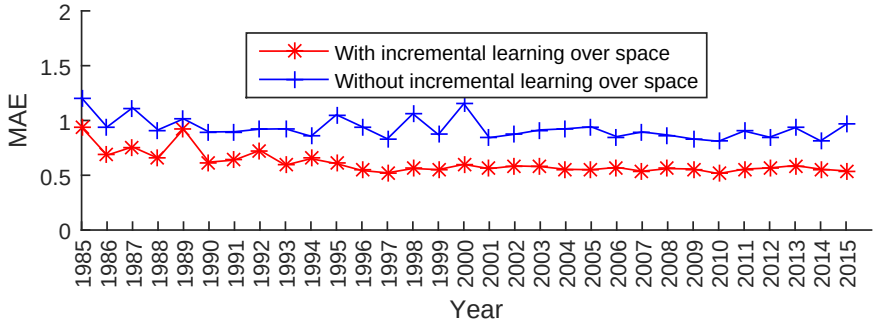



(b) Factor 5

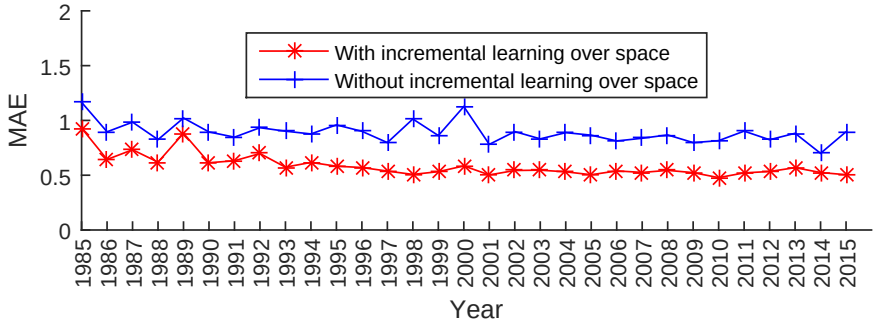
Figure 5.5: Spatial distribution of the spatials factor learned by WISDOM for prcp, continued. (Figure is best viewed in color).



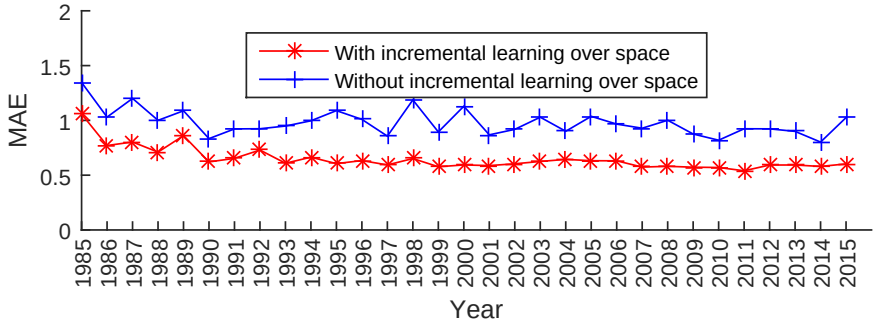

(a) tmax




(b) tmin




(c) tmean




(d) prcp

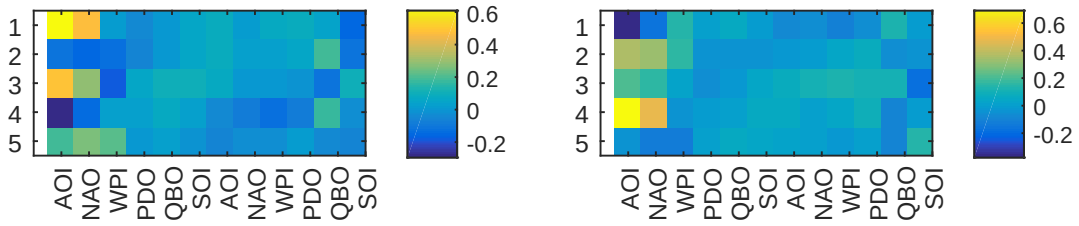
Figure 5.6: Average annual MAE comparison between WISDOM with incremental learning over space and WISDOM without incremental learning over space for the 100 initially chosen locations

Table 5.6: Comparing MAE of WISDOM and WISDOM-KP

| | tmax | tmin | tmean | prcp |
|---|---|---|---|---|
| WISDOM | 0.4751 | **0.5016** | 0.4438 | **0.5700** |
| WISDOM-KP | **0.4678** | 0.5037 | **0.4343** | 0.5725 |

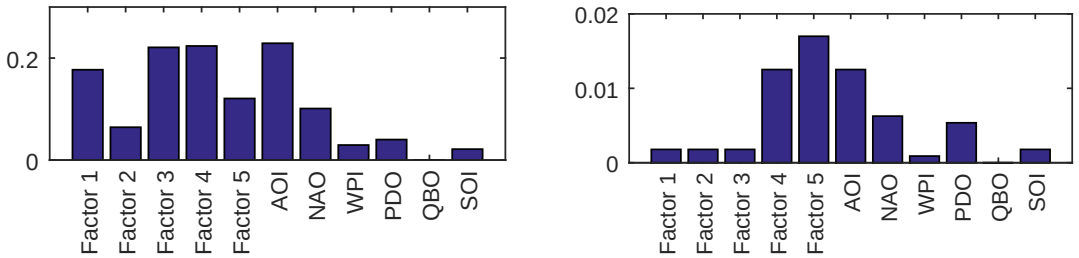
patterns such as SOI, we simply fix one of the columns in the temporal latent factor matrix **B** to be the time series of SOI and learn the remaining spatial and temporal latent factors using WISDOM. We denote this approach as **WISDOM-KP**. The MAE results comparing WISDOM against WISDOM-KP is shown in Table 5.6. The results suggest that WISDOM-KP achieves comparable results as WISDOM in terms of their average MAE.

In addition, we also compared the number of locations where WISDOM-KP outperforms WISDOM. For tmax, tmean, and prcp, the MAE for WISDOM-KP is lower than WISDOM in at least 49% of the locations whereas for tmin, the percentage is around 43%. This is not surprising as we do not expect SOI to accurately capture the climate variability for all locations. Instead, there




(a) tmean (b) prcp

Figure 5.7: Correlations between the climate indices and the temporal factors learned from WISDOM for tmean and prcp




(a) tmean (b) prcp

Figure 5.8: Percentage of locations whose response variables has a correlation above 0.3 with the temporal factors and climate indices learned from WISDOM for tmean and prcp

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 t-mean in Fig. 5.9. 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, and locations around Arctic Ocean. Some of these locations are consistent with the results of previous studies [90].

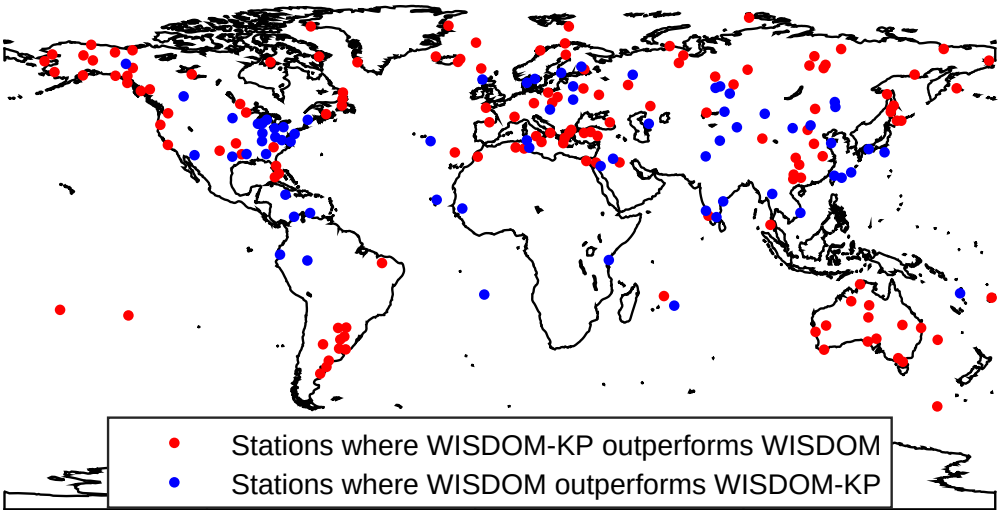



Figure 5.9: Stations where WISDOM-KP outperforms WISDOM more than 0.05 in MAE evaluation for tmean and vise versa.

## 5.4 Conclusion

This chapter presents a spatio-temporal learning framework for multi-location prediction based on supervised tensor decomposition. Unlike conventional methods, the proposed framework constructs both spatial and temporal models of the data and aggregates their output to obtain the final prediction. A novel incremental learning algorithm called WISDOM is developed to simultaneously extracts the latent factors of the spatio-temporal data and learns the spatial and temporal prediction models. We show that WISDOM outperforms several baseline algorithms and can easily accommodate known patterns from the spatio-temporal domain.

# CHAPTER 6

# MULTI-SCALE SPATIO-TEMPORAL MULTI-TASK LEARNING VIA INCREMENTAL MULTI-TENSOR DECOMPOSITION

The spatio-temporal data generated from many scientific domains are often available at multiple spatial scales. For example, the climate projections generated by global and regional climate models are available at a wide range of spatial resolutions, from tens to several hundred kilometers (see Table 6.1). Such multi-scale data can potentially be used to train models for predicting and understanding our future climate. However, since the data at different scales may be correlated to one another, concatenating their variables together into a single feature vector for building predictive models may not be an effective strategy. Similarly, considering the data from a single resolution only, e.g., from the finest scale, may not be sufficient as the models that generate the data at different resolutions may have varying degrees of predictive skills at different locations. How to effectively integrate the data from multiple spatial scales into a unified formulation to build predictive models is therefore a challenge that must be addressed [35, 82, 75].

Note that the problem investigated in this study is different than some of the previous multi-scale modeling tasks investigated by the machine learning, computer vision, and signal processing communities. For example, some of the previous multi-scale modeling techniques would extract a multi-resolution representation of a single dataset (e.g., using wavelet transform [80]), whereas our focus here is on modeling multiple datasets with inherently different spatial scales. Another example is the multi-resolution image data, which are often created by aggregating the pixels within a grid to obtain a low resolution image representation. Unlike such representation, the geospatio-temporal data investigated in this study are obtained from multiple sources (e.g., climate models from various research groups), each based on different model parameterization and with potentially different spatial scales. Furthermore, the multi-resolution image dataset is static whereas the geospatio-temporal dataset is dynamic in nature.

In addition to the challenges posed by the multi-scale data, the complexity of the problem

Table 6.1: Spatial resolutions for various climate datasets

| Climate Dataset | Spatial Resolution |
|---|---|
| NCEP North American Regional Reanalysis | 32 km |
| Canadian Regional Climate Model (CRCM) | 45 km |
| Weather Research & Forecasting model (WRFG) | 50km |
| NCEP Reanalysis | 250km |
| HadCM3 Global Climate Model | 300km |

is further exacerbated by the need to build predictive models for multiple locations instead of building the model for a single location. For example, in climate modeling, robust predictions are often needed for multiple locations within a study region for some future time period. Although a prediction model can be trained for each location separately, the resulting predicted distribution may not be spatially consistent, i.e., it may not preserve the inherent spatial autocorrelation of the data. Recent works [109, 110, 48, 46, 47, 118] have demonstrated the effectiveness of using multi-task learning to produce predictions that are more consistent across space in the study region. These approaches require making assumptions about the relationship between modeling tasks at different locations and utilizing such relationship to jointly train the multiple models. However, none of these previous studies have dealt with the effect of multi-scale data. Furthermore, with the exception of [110], most of these works are primarily batch learning algorithms, which makes them harder to scale up to very large spatio-temporal datasets. How to efficiently deal with the large scale data without affecting the model accuracy is another important challenge that must be addressed in the predictive modeling of geospatio-temporal data.

In addition to its accuracy, interpretability of the model is often a desirable feature. For example, climate scientists are interested to understand the driving factors that influence the variability observed in the climate data. Some of the broad-scale driving factors, such as El Niño, are well-known to the scientists. Thus, having a geospatio-temporal predictive model that can reproduce the known patterns of the domain as well as accurate predictions would help boost our confidence in the validity of the model. While there have been some previous works based on tensor [110] and matrix [109] factorization for deriving such patterns from geospatio-temporal data, it is unclear

whether such patterns would be preserved when applied to multi-scale data.

To address these challenges, this chapter presents a multi-task learning framework named MUSCAT (MUlti-SCAle Spatio-Temporal Learning via Incremental Multi-Tensor Decomposition) to build predictive models from multi-scale spatio-temporal data. The multiple tasks in the proposed framework are not only defined over different scales, but are also defined over space and time. In the proposed framework, the multi-scale data is represented using a set of 3-dimensional tensors, which are jointly decomposed into their latent factors via a multi-tensor decomposition approach. The task relationships among the predictor variables at different scales are captured by assuming that the latent spatial and temporal factors are invariant across the scales. The framework also employs a supervised learning approach to incorporate the latent factors into a loss function for predictive modeling. In short, MUSCAT provides a unified formulation for simultaneous decomposition of multiple tensors and fitting the latent factors to the response variable of interest. In addition, to handle large-scale spatio-temporal data, an incremental learning algorithm is proposed enabling the model to be iteratively constructed over space and time, thus avoiding the need to build the model from scratch each time there is new data available.

The contributions of this work are summarized as follows:

1. A geospatio-temporal multi-task learning framework called MUSCAT is proposed to build prediction models for multi-scale data at multiple locations.

2. The proposed framework allows the latent spatial and temporal patterns shared by the multiple scales to be derived via a multi-tensor decomposition approach.

3. An incremental learning algorithm over space and over time is proposed to scale the MUSCAT framework to large datasets.

4. Experimental evaluation performed using climate data from the United States Historical Climate Network (USHCN) shows the superiority of the framework compared to several baseline multi-scale learning methods.

## 6.1 Background

In this section, we first introduce the notations used in this chapter followed by our problem definition. We then describe the tensor factorization approach employed by our proposed framework.

### 6.1.1 Notations

To facilitate the discussion, we adopt the following notation throughout the chapter. First, scalars are denoted by lowercase italic letters such as $v$ while vectors are denoted by lowercase boldface letters such as $\mathbf{x}$. Note that $\mathbf{x}$ is assumed to be a column vector unless stated otherwise. Its corresponding row vector is denoted as $\mathbf{x}^T$, where the superscript $T$ denote the transpose operation. Matrices are denoted by boldface capital letters such as $\mathbf{A}$ while tensors are represented as boldface calligraphic letters such as $\mathscr{X}$. Furthermore, $\mathbf{a}_k$ denote the vector corresponding to the $k$-th column of the matrix $\mathbf{A}$. A summary of the notations used is shown in Table 6.2.

Table 6.2: Glossary of symbols used in the chapter.

| Symbol | Notation |
|---|---|
| $v$ | A scalar |
| $\mathbf{x}$ | A column vector |
| $\mathbf{A}$ | A matrix |
| $\mathbf{A}_{:,k}$ or $\mathbf{a}_k$ | The $k$-th column of matrix $\mathbf{A}$ |
| $\mathscr{X}$ | A tensor |
| $\mathscr{X}_{i,:,j}$ | The mode-2 fiber of $\mathscr{X}$, which is obtained by fixing the mode-1 index to $i$ and mode-3 index to $j$ |
| $\mathscr{X}_{:,:,i}$ | The $i$-th mode-3 slice of the tensor $\mathscr{X}$ obtained by setting its mode-3 index to $i$ |

We use $\mathbf{x}^T\mathbf{y}$ to denote the dot product between two vectors and $\mathbf{x} \circ \mathbf{y}$ to denote their outer product. Furthermore, let $\|\mathbf{A}\|_F = \sum_{ij} a_{ij}^2$ be the Frobenius norm of matrix $\mathbf{A}$ and $\|\mathscr{X}\|_F = \sqrt{\sum_{ijk} x_{ijk}^2}$ be the Frobenius norm of the tensor $\mathscr{X}$. For other tensor operations, the notations used are given in Table 6.3.

Table 6.3: Glossary of tensor/matrix operations.

| | |
|---|---|
| mode-$n$ matricization | reordering the elements of a tensor $\mathscr{X} \in \Re^{p_1 \times \ldots \times p_N}$ into a matrix $\mathbf{X}_{(n)} \in \Re^{p_n \times q_n}$, where $q_n = \Pi_{k \neq n} p_k$, and each of the mode-$n$ fibers of $\mathscr{X}$ is a column of $\mathbf{X}_{(n)}$ |
| mode-$n$ product | $(\mathscr{X} \times_n \mathbf{A})_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{p_n} \mathscr{X}_{i_1,\ldots,i_N} \mathbf{A}_{j,i_n}$ |
| Khatri-Rao product | $\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1, \mathbf{a}_2 \otimes \mathbf{b}_2, \ldots, \mathbf{a}_K \otimes \mathbf{b}_K]$, where $\otimes$ is the Kronecker product. |

### 6.1.2 Tensor Factorization

We employ the following CANDECOMP/PARAFAC (CP) decomposition [61] to factorize a 3rd-order tensor $\mathscr{X}$ into its corresponding latent factors **A**, **B**, and **C**:

$$\mathscr{X} = [\![\mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_{k=1}^{K} \mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k. \tag{6.1}$$

Specifically, the tensor is factorized into the sum of $K$ rank-1 tensors, each of which is given by the outer product $\mathbf{a}_k \circ \mathbf{b}_k \circ \mathbf{c}_k$. In addition, the vectors $\mathbf{a}_k$, $\mathbf{b}_k$ and $\mathbf{c}_k$ denote the $k$-th columns of the matrices **A**, **B**, and **C**, respectively

### 6.1.3 Spatio-Temporal Prediction

Let $\mathscr{D} = (\mathscr{X}^{(1)}, \ldots, \mathscr{X}^{(L)}, \mathbf{Y})$ be a multi-scale spatio-temporal data set, where $\mathscr{X}^{(l)} \in \Re^{S \times T \times d_l}$ denote the spatio-temporal tensor of predictor variables from the $l$-th resolution, $\mathbf{Y} \in \Re^{S \times T}$ denote the response variable for all the locations, $S$ is the number of locations, $T$ is the length of the time series, and $d_l$ is the number of predictor variables for scale $l$.

The goal of spatio-temporal predictive modeling is to learn a target function $f(\{\mathbf{x}^{(l)}\}; \Pi)$ that maps each multi-scale variable $\{\mathbf{x}^{(l)}\}$ to its corresponding response variable $y \in \mathbb{R}$ in such a way that minimizes the difference between its predicted and true value. Note that $\Pi$ denotes the set of parameters associated with the target function.

## 6.2 MUSCAT

This section presents our proposed framework for multi-scale multi-task learning. We begin with the discussion on multi-tensor decomposition.

### 6.2.1 Multi-tensor decomposition

Inspired by previous work on discovering climate indices using SVD [90], tensor decomposition, as a generalization of SVD to higher dimensions, is employed to learn the spatial and temporal latent factors of the data. Since the data are available at multiple scales, the tensor decomposition can be performed at each scale.

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}^{(l)}} \frac{1}{2(S \times T \times d_l)} \|\mathscr{X}^{(l)} - [\![\mathbf{A},\mathbf{B},\mathbf{C}^{(l)}]\!]\|_F^2 + \Omega_d(\mathbf{A},\mathbf{B},\mathbf{C}^{(l)})$$

where $\mathbf{A}$ is the spatial latent factor with size $S \times K$ and $\mathbf{B}$ is the temporal latent factor with size $T \times K$. We assume the underlining climate variability of the data from different scales should be consistent, and thus, the latent factors $\mathbf{A}$ and $\mathbf{B}$ learned from the data should be shared between scales. $\frac{1}{S \times T \times d_l}$ is used to normalize the norm so that it does not depend on the size of the tensor. $\Omega_d(\mathbf{A},\mathbf{B},\mathbf{C})$ is a regularization term for matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$, and we enforce sparsity of the latent factors by using $\ell_1$-norm:

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

### 6.2.2 Supervised Multi-tensor Decomposition

The latent factors derived from the multi-tensor decomposition may capture the spatial and temporal climate variabilities of the data, and could be beneficial to spatio-temporal prediction modeling. In this section, we discuss how to incorporate the latent factors into the supervised learning framework.

We assume the prediction function is composed by a weighted average of the predictions from multiple scales:

$$\hat{y}_{s,t} \quad = \quad \sum_{l}^{L} \alpha_l \hat{y}_{s,t}^{(l)}$$

where $\alpha_l$ is the weights for the prediction from the $l$-th scale, and is assumed to be probability simplex ($\alpha_l > 0$ and $\sum_l^L \alpha_l = 1$).

For each scale $l$, we assume the prediction function is composed by a temporal model for multiple locations, and a spatial model for multiple time points, which is defined as follows:

$$\hat{y}_{s,t}^{(l)} = \mathbf{x}_{s,t}^{(l)^T} (\sum_{k}^{K} \mathbf{A}_{s,k} \mathbf{w}_k^{(l)} + \sum_{k}^{K} \mathbf{B}_{t,k} \mathbf{v}_k^{(l)})$$

where $\mathbf{w}_k^{(l)}$ and $\mathbf{v}_k^{(l)}$ denote the parameters for spatial and temporal prediction models for the $k$-th latent factor at scale $l$, while $\mathbf{A}_{s,k}$ and $\mathbf{B}_{t,k}$ are the weights of location $s$ or time point $t$ to the $k$-th latent factor. Note that $\mathbf{x}_{s,t}^{(l)^T} \sum_k^K \mathbf{A}_{s,k} \mathbf{w}_k^{(l)}$ is a temporal model prediction for location $s$, and $\mathbf{x}_{s,t}^{(l)^T} \sum_k^K \mathbf{B}_{t,k} \mathbf{v}_k^{(l)}$ is a spatial model prediction for time $t$. The relatedness between the multi-scale data is defined by the consistency of the latent spatial and temporal factors between scales. In other words, the $L$ prediction components share the same spatial factor $\mathbf{A}$ and temporal factor $\mathbf{B}$.

Putting the multi-tensor decomposition and supervised learning together with squared loss function, the objective function can be written as:

$$\begin{aligned}
&\min_{\substack{\{\alpha_l, \mathbf{W}^{(l)}, \mathbf{V}^{(l)}, \mathbf{C}^{(l)}\} \\ \mathbf{A}, \mathbf{B}}} \mathscr{F}\left(\{\mathbf{W}^{(l)}, \mathbf{V}^{(l)}, \mathbf{C}^{(l)}\}, \mathbf{A}, \mathbf{B}\right) \\
= \quad & \frac{1}{2} \sum_{s}^{S} \sum_{t}^{T} \left[ \sum_{l}^{L} \alpha_l \mathbf{x}_{s,t}^{(l)^T} (\sum_{k}^{K} \mathbf{A}_{s,k} \mathbf{w}_k^{(l)} + \sum_{k}^{K} \mathbf{B}_{t,k} \mathbf{v}_k^{(l)}) - y_{s,t} \right]^2 \\
+ \quad & \frac{\lambda}{2} \sum_{l}^{L} \frac{1}{S \times T \times d_l} \| \mathscr{X}^{(l)} - [\![ \mathbf{A}, \mathbf{B}, \mathbf{C}^{(l)} ]\!] \|_F^2 + \beta \sum_{l}^{L} \| [\{\mathbf{W}^{(l)}, \mathbf{V}^{(l)}, \mathbf{C}^{(l)}\}, \mathbf{A}, \mathbf{B}] \|_1
\end{aligned}$$

s.t.

$$\alpha_l > 0 \text{ for all } l \text{ and } \sum_{l}^{L} \alpha_l = 1 \tag{6.2}$$

where we use $\|[\{\mathbf{W}^{(l)},\mathbf{V}^{(l)},\mathbf{C}^{(l)}\},\mathbf{A},\mathbf{B}]\|_1$ to denote the $\ell_1$ norm regularization term for $\mathbf{W}^{(l)}$, $\mathbf{V}^{(l)}$, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}^{(l)}$, respectively.

### 6.2.3 MUSCAT

Spatio-temporal data are usually growing over space and time, when new spatial objects become available, or new measurements are observed for further timestamps. It is expensive to learn the models repeatedly over the whole dataset whenever there are new data available. In this section, we propose an efficient algorithm called MUSCAT to optimize the objective function in Eq.(6.2), which incrementally learns the model parameters and latent factors over space or time. MUSCAT performs update only on the new available data, and the models are constrained to be close to the previous learned models, such that the newly learned model still contains information from the old data. Besides, MUSCAT does not require old data to perform the learning on new data, and hence no old data need to be stored in memory.

The optimization problem for incremental learning of the objective function in Eq.(6.2) can be formulated as follows:

$$\begin{aligned}
&\min_{\substack{\{\alpha_l,\mathbf{W}^{(l)},\mathbf{V}^{(l)},\mathbf{C}^{(l)}\}\\ \mathbf{A},\mathbf{B}}} \mathscr{Q}\left(\begin{array}{c}\{\alpha_l,\tilde{\alpha}_l,\mathbf{W}^{(l)},\tilde{\mathbf{W}}^{(l)},\mathbf{V}^{(l)},\tilde{\mathbf{V}}^{(l)},\\ \mathbf{C}^{(l)},\tilde{\mathbf{C}}^{(l)}\},\mathbf{A},\tilde{\mathbf{A}},\mathbf{B},\tilde{\mathbf{B}}\end{array}\right)\\
=\;& \mathscr{F}(\{\alpha_l,\mathbf{W}^{(l)},\mathbf{V}^{(l)},\mathbf{C}^{(l)}\},\mathbf{A},\mathbf{B})\\
+\;& \Gamma\left(\begin{array}{c}\{\alpha_l,\tilde{\alpha}_l,\mathbf{W}^{(l)},\tilde{\mathbf{W}}^{(l)},\mathbf{V}^{(l)},\tilde{\mathbf{V}}^{(l)},\mathbf{C}^{(l)},\tilde{\mathbf{C}}^{(l)}\},\\ \mathbf{A},\tilde{\mathbf{A}},\mathbf{B},\tilde{\mathbf{B}}\end{array}\right)
\end{aligned}$$

where $\{\alpha_l\}$ , $\{\tilde{\mathbf{W}}^{(l)}\}$, $\{\tilde{\mathbf{V}}^{(l)}\}$, $\tilde{\mathbf{A}}$, $\tilde{\mathbf{B}}$, and $\{\tilde{\mathbf{C}}^{(l)}\}$ denote the previous model parameters before the incremental update, $\mathscr{F}(\{\alpha_l,\mathbf{W}^{(l)},\mathbf{V}^{(l)},\mathbf{C}^{(l)}\},\mathbf{A},\mathbf{B})$ is given by Eq.(6.2) and

$$\begin{aligned}
&\Gamma\left(\begin{array}{c}\{\alpha_l,\tilde{\alpha}_l,\mathbf{W}^{(l)},\tilde{\mathbf{W}}^{(l)},\mathbf{V}^{(l)},\tilde{\mathbf{V}}^{(l)},\mathbf{C}^{(l)},\tilde{\mathbf{C}}^{(l)}\},\\ \mathbf{A},\tilde{\mathbf{A}},\mathbf{B},\tilde{\mathbf{B}}\end{array}\right)\\
=\;& \sum_{l}^{L}\left((\alpha_l-\tilde{\alpha}_l)^2+\|\mathbf{W}^{(l)}-\tilde{\mathbf{W}}^{(l)}\|_F^2+\|\mathbf{V}^{(l)}-\tilde{\mathbf{V}}^{(l)}\|_F^2+\|\mathbf{C}^{(l)}-\tilde{\mathbf{C}}^{(l)}\|_F^2\right)\\
+\;& \|\mathbf{A}-\tilde{\mathbf{A}}\|_F^2+\|\mathbf{B}-\tilde{\mathbf{B}}\|_F^2, \qquad (6.3)
\end{aligned}$$

which controls how much information will be retained from the previous model.

#### 6.2.3.1 Incremental Learning over Space

In this section, we describe the MUSCAT framework for incremental learning over space.

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, then $t_0 = T$. We further assume that the spatial latent factors 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 factors for other modes ($\mathbf{B}$ and $\mathbf{C}^{(l)}$) as well as the parameters of the prediction models ($\alpha_l$, $\mathbf{W}^{(l)}$ and $\mathbf{V}^{(l)}$) can be affected by the addition of the new data, $\{\mathbf{x}_{S+1,t_0}, \mathbf{x}_{S+1,t_0+1}, \cdots, \mathbf{x}_{S+1,T}\}$. For brevity, we denote the feature vectors for the new location as $\mathscr{X}_{S+1}$, which is a tensor of size $1 \times (T - t_0 + 1) \times d$ composed by $\mathscr{X}^{(l)}_{S+1} \in R^{1\times(T-t_0+1)\times d_l}$ for $l = 1, ..., L$.

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

$$
\begin{aligned}
&\min_{\substack{\{\alpha_l, \mathbf{W}^{(l)}, \mathbf{V}^{(l)}, \mathbf{C}^{(l)}\}, \\ \mathbf{A}_{S+1}, \mathbf{B}}} \mathscr{Q}\left(\begin{array}{l}\{\mathbf{W}^{(l)}, \tilde{\mathbf{W}}^{(l)}, \mathbf{V}^{(l)}, \tilde{\mathbf{V}}^{(l)}, \\ \mathbf{C}^{(l)}, \tilde{\mathbf{C}}^{(l)}\}, \mathbf{A}_{S+1}, \mathbf{B}, \tilde{\mathbf{B}}\end{array}\right) \\
= \;& \frac{1}{2}\sum_{t=t_0}^{T}\left[\sum_{l}^{L}\alpha_l {\mathbf{x}^{(l)}_{S+1,t}}^T(\mathbf{W}^{(l)T}\mathbf{A}_{S+1} + \mathbf{V}^{(l)T}\mathbf{B}_t) - y_{S+1,t}\right]^2 \\
&+ \frac{\lambda_1}{2}\sum_{l}^{L}\frac{1}{T \times d_l}\|\mathscr{X}^{(l)}_{S+1} - [\![\mathbf{A}^T_{S+1}, \mathbf{B}, \mathbf{C}^{(l)}]\!]\|^2_F \\
&+ \frac{\eta_1}{2}\left[\sum_{l}^{L}((\alpha_l - \tilde{\alpha}_l)^2 + \|\mathbf{W}^{(l)} - \tilde{\mathbf{W}}^{(l)}\|^2_F + \|\mathbf{V}^{(l)} - \tilde{\mathbf{V}}^{(l)}\|^2_F + \|\mathbf{C}^{(l)} - \tilde{\mathbf{C}}^{(l)}\|^2_F) + \|\mathbf{B} - \tilde{\mathbf{B}}\|^2_F\right] \\
&+ \beta_1\|[\{\mathbf{W}^{(l)}, \mathbf{V}^{(l)}, \mathbf{C}^{(l)}\}, \mathbf{A}_{S+1}, \mathbf{B}]\|_1
\end{aligned}
\tag{6.4}
$$

s.t.

$$\alpha_l > 0 \text{ for all } l \text{ and } \sum_{l}^{L}\alpha_l = 1$$

Note that $\mathbf{A}_{S+1}$ is a column vector that represents the spatial latent factors for the new location and $\mathbf{x}_{S+1,t}$ denote the feature vector of the location at time $t$. The smoothness parameter $\eta_1$ determines

the extent to which the previous model parameters should be retained.

An alternating minimization strategy is used to solve the optimization problem, and each sub-problem is solved by the proximal gradient descent method [84]. The parameters are updated iteratively by calculating the gradient on the smooth part of the objective function, and then apply the soft-thresholding operator to determine its next value. The step size can be found using a line search algorithm. In the following, we provide the gradient of the objective function for each alternating minimization step.

**I. Solving for $\mathbf{A}_{S+1}$ by fixing $\{\alpha_l, \mathbf{W}^{(l)}, \mathbf{V}^{(l)}, \mathbf{C}^{(l)}\}$, B:**

The objective function with respect to $\mathbf{A}_{S+1}$ can be written as follows:

$$\begin{aligned}\min_{\mathbf{A}_{S+1}} \quad & \frac{1}{2}\sum_{t=t_0}^{T}\left[\sum_{l}\alpha_l {\mathbf{x}^{(l)}_{S+1,t}}^T(\mathbf{W}^{(l)T}\mathbf{A}_{S+1}+\mathbf{V}^{(l)T}\mathbf{B}_t)-y_{S+1,t}\right]^2 \\ & +\frac{\lambda_1}{2}\sum_{l}^{L}\frac{1}{T\times d_l}\|\mathbf{X}^{(l)}_{S+1(1)}-\mathbf{A}^T_{S+1}(\mathbf{C}^{(l)}\odot\mathbf{B})^T\|_F^2 \\ & +\beta_1\|\mathbf{A}_{S+1}\|_1 \end{aligned} \tag{6.5}$$

where $\mathbf{X}^{(l)}_{S+1(1)}$ is the mode-1 unfolding of the tensor $\mathscr{X}^{(l)}_{S+1}$.The gradient on the smooth part of the objective function w.r.t. $\mathbf{A}_{S+1}$ is

$$\begin{aligned}&\sum_{t=t_0}^{T}\left[\sum_{l}\alpha_l {\mathbf{x}^{(l)}_{S+1,t}}^T(\mathbf{W}^{(l)T}\mathbf{A}_{S+1}+\mathbf{V}^{(l)T}\mathbf{B}_t)-y_{S+1,t}\right]\sum_{l}\alpha_l\mathbf{W}^{(l)}\mathbf{x}^{(l)}_{S+1,t} \\ &-\sum_{l}^{L}\frac{\lambda_1}{T\times d_l}\left([\mathbf{X}^{(l)}_{S+1(1)}-\mathbf{A}^T_{S+1}(\mathbf{C}^{(l)}\odot\mathbf{B})^T](\mathbf{C}^{(l)}\odot\mathbf{B})\right)^T\end{aligned}$$

**II. Solving for B by fixing $\mathbf{A}_{S+1}$, $\{\alpha_l, \mathbf{W}^{(l)}, \mathbf{V}^{(l)}, \mathbf{C}^{(l)}\}$:**

The terms in the objective function involving matrix **B** include

$$\begin{aligned}\min_{\mathbf{B}} \quad & \frac{1}{2}\sum_{t=t_0}^{T}\left[\sum_{l}\alpha_l {\mathbf{x}^{(l)}_{S+1,t}}^T(\mathbf{W}^{(l)T}\mathbf{A}_{S+1}+\mathbf{V}^{(l)T}\mathbf{B}_t)-y_{S+1,t}\right]^2 \\ & +\frac{\lambda_1}{2}\sum_{l}^{L}\frac{1}{T\times d_l}\|\mathbf{X}^{(l)}_{S+1(2)}-\mathbf{B}(\mathbf{C}^{(l)}\odot\mathbf{A}^T_{S+1})^T\|_F^2 \\ & +\frac{\eta_1}{2}\|\mathbf{B}-\tilde{\mathbf{B}}\|_F^2+\beta_1\|\mathbf{B}\|_1 \end{aligned} \tag{6.6}$$

The gradient on the smooth part of the objective function w.r.t. $\mathbf{B}_t$ is given by

$$\begin{aligned}
&\left[\sum_l^L \alpha_l \mathbf{x}_{S+1,t}^{(l)}{}^T (\mathbf{W}^{(l)T}\mathbf{A}_{S+1} + \mathbf{V}^{(l)T}\mathbf{B}_t) - y_{S+1,t}\right] \sum_l^L \alpha_l \mathbf{V}^{(l)} \mathbf{x}_{S+1,t}^{(l)} \\
- &\sum_l^L \frac{\lambda_1}{T \times d_l} \left[ (\mathbf{x}_{S+1,t}^{(l)}{}^T - \mathbf{B}_t^T (\mathbf{C}^{(l)} \odot \mathbf{A}_{S+1}^T)^T)(\mathbf{C}^{(l)} \odot \mathbf{A}_{S+1}^T) \right]^T \\
+ &\eta_1 (\mathbf{B}_t - \tilde{\mathbf{B}}_t)
\end{aligned}$$

**III. Solving for $\{\mathbf{C}^{(l)}\}$ by fixing $\mathbf{A}_{S+1}$, B, $\{\alpha_l, \mathbf{W}^{(l)}, \mathbf{V}^{(l)}\}$:**

We will solve $\mathbf{C}^{(l)}$ for $l = 1, ..., L$ individually. We can simplify the objective function to include only terms involving the matrix $\mathbf{C}^{(l)}$:

$$\begin{aligned}
\min_{\mathbf{C}^{(l)}} \quad & \frac{\lambda_1}{2T \times d_l} \|\mathbf{X}_{S+1(3)}^{(l)} - \mathbf{C}^{(l)} (\mathbf{B} \odot \mathbf{A}_{S+1}^T)^T\|_F^2 \\
& + \frac{\eta_1}{2} \|\mathbf{C}^{(l)} - \tilde{\mathbf{C}}^{(l)}\|_F^2 + \beta_1 \|\mathbf{C}^{(l)}\|_1
\end{aligned} \tag{6.7}$$

The gradient on the smooth part of the objective function w.r.t. $\mathbf{C}^{(l)}$ is given by

$$-\frac{\lambda_1}{T \times d_l} \left[ \mathbf{X}_{S+1(3)}^{(l)} - \mathbf{C}^{(l)} (\mathbf{B} \odot \mathbf{A}_{S+1}^T)^T \right] (\mathbf{B} \odot \mathbf{A}_{S+1}^T) + \eta_1 (\mathbf{C}^{(l)} - \tilde{\mathbf{C}}^{(l)})$$

**IV. Solving for $\{\mathbf{W}^{(l)}\}$ by fixing $\mathbf{A}_{S+1}$, B, $\{\alpha_l, \mathbf{V}^{(l)}, \mathbf{C}^{(l)}\}$:**

The terms in the objective function involving the model parameter $\mathbf{W}^{(l)}$ is

$$\begin{aligned}
\min_{\mathbf{W}^{(l)}} \quad & \frac{1}{2} \sum_{t=t_0}^T \left[ \sum_l^L \alpha_l \mathbf{x}_{S+1,t}^{(l)}{}^T (\mathbf{W}^{(l)T}\mathbf{A}_{S+1} + \mathbf{V}^{(l)T}\mathbf{B}_t) - y_{S+1,t} \right]^2 \\
& + \frac{\eta_1}{2} \|\mathbf{W}^{(l)} - \tilde{\mathbf{W}}^{(l)}\|_F^2 + \beta_1 \|\mathbf{W}^{(l)}\|_1
\end{aligned} \tag{6.8}$$

The gradient on the smooth part of the objective function w.r.t. $\mathbf{W}^{(l)}$ is given by

$$\begin{aligned}
&\sum_{t=t_0}^T \alpha_l \mathbf{x}_{S+1,t}^{(l)} \left[ \sum_i^L \alpha_i \mathbf{x}_{S+1,t}^{(l)}{}^T (\mathbf{W}^{(i)T}\mathbf{A}_{S+1} + \mathbf{V}^{(i)T}\mathbf{B}_t) - y_{S+1,t} \right] \mathbf{A}_{S+1}^T \\
&+ \eta_1 (\mathbf{W}^{(l)T} - \tilde{\mathbf{W}}^{(l)T})
\end{aligned}$$

**V. Solving for $\{\mathbf{V}^{(l)}\}$ by fixing $\{\alpha_l, \mathbf{W}^{(l)}, \mathbf{C}^{(l)}\}$, $\mathbf{A}_{S+1}$, B:**

The objective function can be simplified for terms involving $\mathbf{V}^{(l)}$ as follows:

$$\begin{aligned}
\min_{\mathbf{V}^{(l)}} \quad & \frac{1}{2} \sum_{t=t_0}^T \left[ \sum_l^L \alpha_l \mathbf{x}_{S+1,t}^{(l)}{}^T (\mathbf{W}^{(l)T}\mathbf{A}_{S+1} + \mathbf{V}^{(l)T}\mathbf{B}_t) - y_{S+1,t} \right]^2 \\
& + \frac{\eta_1}{2} \|\mathbf{V}^{(l)} - \tilde{\mathbf{V}}^{(l)}\|_F^2 + \beta_1 \|\mathbf{V}^{(l)}\|_1
\end{aligned} \tag{6.9}$$

The gradient on the smooth part of the objective function w.r.t. $\mathbf{V}^{(l)}$ is given by

$$\sum_{t=t_0}^{T} \alpha_l \mathbf{x}_{S+1,t}^{(l)} \left[ \sum_i \alpha_i \mathbf{x}_{S+1,t}^{(i)}{}^T (\mathbf{W}^{(i)T} \mathbf{A}_{S+1} + \mathbf{V}^{(i)T} \mathbf{B}_t) - y_{S+1,t} \right] \mathbf{B}_t^T$$
$$+\eta_1 (\mathbf{V}^{(l)} - \tilde{\mathbf{V}}^{(l)})$$

**VI. Solving for $\alpha_l$ by fixing $\{\mathbf{W}^{(l)}, \mathbf{V}^{(l)}, \mathbf{C}^{(l)}\}$, $\mathbf{A}_{S+1}$, B:**

Finally, the parameters for reweighting models learned for different scales can be solved by the following objective function:

$$\min_{\alpha_l} \quad \frac{1}{2} \sum_{t=t_0}^{T} \left[ \sum_l \alpha_l \mathbf{x}_{S+1,t}^{(l)}{}^T (\mathbf{W}^{(l)T} \mathbf{A}_{S+1} + \mathbf{V}^{(l)T} \mathbf{B}_t) - y_{S+1,t} \right]^2$$
$$+ \frac{\eta_1}{2} (\alpha_l - \tilde{\alpha}_l)^2$$

s.t.

$$\alpha_l > 0 \text{ for all } l \text{ and } \sum_l^L \alpha_l = 1$$

The gradient on the objective function w.r.t $\alpha_l$ is given by:

$$\sum_{t=t_0}^{T} \left[ \sum_i \alpha_i \mathbf{x}_{S+1,t}^{(i)}{}^T (\mathbf{W}^{(i)T} \mathbf{A}_{S+1} + \mathbf{V}^{(i)T} \mathbf{B}_t) \right.$$
$$\left. - y_{S+1,t} \right] \mathbf{x}_{S+1,t}^{(l)}{}^T (\mathbf{W}^{(l)T} \mathbf{A}_{S+1} + \mathbf{V}^{(l)T} \mathbf{B}_t)$$
$$+\eta_1 (\alpha_l - \tilde{\alpha}_l)$$

The probability simplex constraint is solved by performing a Euclidean projection onto the probability simplex in proximal mapping step for the proximal gradient descent algorithm [102].

#### 6.2.3.2 Incremental Learning over Time

Similar to incremental learning over space, we also develop incremental learning over time for MUSCAT when new observations become available at new timestamps. Let $S$ be the number of locations and $T$ be the current time and we assume the availability of the feature vectors of predictor variables for all $S$ stations at time $T+1$. This information will be used to calculate the temporal latent factor $\mathbf{B}_{T+1}$ for the new time period. Similar to the strategy for incremental learning over

space, we assume the new data for time $T+1$ does not affect previous temporal latent factors $\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_T$.

Different from incremental learning over space, there are two steps for incremental learning over time. First, we learn the temporal latent factor $\mathbf{B}_{T+1}$ based on the values of the predictor variables for the new time period before the target variable is observed. Note that the predictions for the next timestamp is performed based on the parameters at current stage. Next, the model parameters and latent factors for other modes are updated when the target variable for the new time period is observed for all the locations.

**Step 1: Updating the temporal latent factor $\mathbf{B}_{T+1}$ before observing target variable.** The objective function for updating the temporal latent factor is given below:

$$\min_{\mathbf{B}_{T+1}} \mathscr{Q}(\mathbf{B}_{T+1}) = \frac{\lambda_2}{2} \sum_{l}^{L} \frac{1}{S \times d_l} \| \mathscr{X}^{(l)}_{T+1} - [\![ \mathbf{A}, \mathbf{B}^T_{T+1}, \mathbf{C}^{(l)} ]\!] \|^2_F$$

Note that the loading matrices $\mathbf{A}$ and $\{\mathbf{C}^{(l)}\}$ correspond to the values obtained from the previous update. The predictions for the target variable are performed using $\mathbf{B}_{T+1}$ and other model parameters from previous update.

**Step 2: Updating model parameters and latent factors after observing target variable.** After obtaining the new observations for target variable at time $T+1$, we can derive the update formula

by optimizing the following objective function:

$$\min_{\substack{\{\alpha_l,\mathbf{W}^{(l)},\mathbf{V}^{(l)},\mathbf{C}^{(l)}\},\\ \mathbf{A},\mathbf{B}^{T+1}}} \mathscr{Q}\left(\begin{array}{c}\{\mathbf{W}^{(l)},\mathbf{V}^{(l)},\mathbf{C}^{(l)}\},\mathbf{A},\mathbf{B}^{T+1},\\ \{\tilde{\mathbf{W}}^{(l)},\tilde{\mathbf{V}}^{(l)},\tilde{\mathbf{C}}^{(l)}\},\tilde{\mathbf{A}}\end{array}\right)$$

$$\begin{aligned}
= \;& \frac{1}{2}\sum_{s}^{S}\left[\sum_{l}^{L}\alpha_l \mathbf{x}_{s,T+1}^{(l)\,T}(\mathbf{W}^{(l)\,T}\mathbf{A}_s + \mathbf{V}^{(l)\,T}\mathbf{B}_{T+1}) - y_{s,T+1}\right]^2 \\
+ \;& \frac{\lambda_2}{2}\sum_{l}^{L}\frac{1}{S\times d_l}\|\mathscr{X}_{T+1}^{(l)} - [\![\mathbf{A},\mathbf{B}_{T+1}^{T},\mathbf{C}^{(l)}]\!]\|_F^2 \\
+ \;& \frac{\eta_2}{2}\left[\sum_{l}^{L}((\alpha_l-\tilde{\alpha}_l)^2 + \|\mathbf{W}^{(l)}-\tilde{\mathbf{W}}^{(l)}\|_F^2\right. \\
& \left. + \|\mathbf{V}^{(l)}-\tilde{\mathbf{V}}^{(l)}\|_F^2 + \|\mathbf{C}^{(l)}-\tilde{\mathbf{C}}^{(l)}\|_F^2) + \|\mathbf{A}-\tilde{\mathbf{A}}\|_F^2\right] \\
+ \;& \beta_2(\|\{\mathbf{W}^{(l)},\mathbf{V}^{(l)},\mathbf{C}^{(l)}\},\mathbf{A},\mathbf{B}_{T+1}\|_1)
\end{aligned}$$

s.t.

$$\alpha_l > 0 \text{ for all } l \text{ and } \sum_{l}^{L}\alpha_l = 1$$

The optimization approach for solving Eq. (6.10) to obtain the update formula for $\{\alpha_l,\mathbf{W}^{(l)},\mathbf{V}^{(l)},\mathbf{C}^{(l)}\}$, $\mathbf{A}$ and $\mathbf{B}_{T+1}$ is similar to the approach for incremental learning over space.

## 6.3 Experimental Evaluation

This section describes the experiments performed to evaluate the performance of MUSCAT using a multi-scale climate dataset.

### 6.3.1 Dataset Description

We use a multi-scale climate dataset that has three spatial resolutions. The data at the finest scale correspond to monthly observations obtained from various weather stations. The dataset was obtained from the United States Historical Climatology Network (USHCN)[1]. Four variables from this dataset—maximum (tmax), minimum (tmin), mean (tmean) temperature and precipitation (prcp)—

[1] http://cdiac.ornl.gov/epubs/ndp/ushcn/ushcn.html

Table 6.4: List of variables from NARR data as predictors.

| Variable | Description |
|---|---|
| acpcp | Monthly mean convective precipitation accumulation at surface |
| air.2m | Monthly mean air temperature at 2 m |
| dlwrf | Monthly mean downward longwave radiation flux at surface |
| dswrf | Monthly mean downward shortwave radiation flux at surface |
| lftx4 | Monthly mean (4-layer) lifted index |
| prate | Monthly mean precipitation rate at surface |
| prmsl | Monthly mean pressure at mean sea level |
| pr_wtr | Monthly mean precipitable water for entire atmosphere |
| rhum | Monthly mean relative humidity at 2 m |

are selected as the response variables for our multi-scale modeling task. The prediction for each response variable will be modeled independently[2].

For predictor variables, we will use two other climate datasets, NARR and NCEP reanalysis. NARR[3] corresponds to the North American regional reanalysis dataset and has a spatial resolution of 0.3° (32 km). Nine predictor variables are selected from this data source with the help of our domain expert. The variables are shown in Table 6.4. NCEP reanalysis[4] is a gridded climate dataset with a coarser spatial resolution of 2.5°. Seven surface variables are selected as the predictor variables based on the suggestion of our domain experts. A detailed description of the selected variables is shown in Table 6.5.

The collected datasets cover the entire United States, spanning the months of January, 1985 until November, 2015. However, since there are missing values in the response variables for some

[2]Even though the four variables can be modeled jointly using a multi-task learning framework, this is currently beyond the scope of this work.

[3]https://www.esrl.noaa.gov/psd/data/gridded/data.narr.html

[4]`http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.derived.html`

Table 6.5: List of surface variables selected from NCEP reanalysis data as predictors.

| Variable | Description |
|---|---|
| cprat.sfc | Monthly mean convective precipitation rate at surface |
| dlwrf.sfc | Monthly mean downward longwave radiation flux at surface |
| dswrf.sfc | Monthly mean downward shortwave radiation flux at surface |
| prate.sfc | Monthly mean of precipitation rate at surface |
| tmax.2m | Monthly mean maximum temperature at 2 m |
| tmin.2m | Monthly mean minimum temperature at 2 m |
| lftx.sfc | Monthly mean surface lifted index |

Table 6.6: Number of weather stations and grid cells for each response variable.

| | USHCN | NARR | NCEP |
|---|---|---|---|
| tmax | 357 | 350 | 146 |
| tmin | 341 | 336 | 144 |
| tmean | 333 | 328 | 143 |
| prcp | 790 | 635 | 159 |

weather stations over the 30-year period, the number of data points (weather stations) varies from one response variable to another. We also remove the grid cells in the NARR and NCEP reanalysis datasets that do not have any corresponding weather station data. Table 6.6 summarizes the number of weather stations and grid cells for each response variable.

After removing the stations with missing values, we apply the following preprocessing step to all the time series of predictor and response variables. For each variable at a given scale, we deseasonalize the time series by subtracting each monthly value from the mean value of its corresponding month. The time series is then standardized to obtain a sequence of Z-scores.

Next, we divide the dataset into 3 equal-sized partitions for training, validation, and testing. We use the data from the first 10 years (1985-1994) for training, the next 10 years (1995-2004) for validation, and the rest (2005-2015) for testing. The validation set is used to perform parameter tuning. We adopt a similar evaluation strategy as WISDOM [110] by setting the number of latent

factors to be $k = 5$. Note that the number of weather stations included in the training, validation, and testing may vary depending on when the station was first introduced into our incremental learning formulation. Specifically, we first randomly choose 100 weather stations as our starting locations. At each step during the incremental learning process, we randomly introduce either a new station (along with its historical data) or a new time period (where the data for the new time period becomes available to all the previously selected stations) for updating the latent factors as well as the spatial and temporal multi-scale models. With this incremental learning strategy, a station could be introduced during the training period, the validation period, or the testing period.

We used the following mean absolute error (MAE) metric to evaluate the performance of various algorithms:

$$\text{MAE}(\{(\mathbf{y}_t, \hat{\mathbf{y}}_t)\}) = \frac{\sum_{s=1}^{S} \sum_{n=1}^{N} |y_{s,n} - \hat{y}_{s,n}|}{S \times N} \tag{6.10}$$

Note that the evaluation metric is calculated for each station starting from the test period in which incremental learning is performed for the given station. For example, if a station was first introduced into the incremental learning formulation during the training or validation period, then its MAE is calculated for the entire test period. However, if the station was first introduced during the testing period, its MAE is computed starting from the test period in which it was first introduced. The incremental learning process is repeated 10 times, each time with a different random initialization. Results are reported based on the average performance over the 10 trials.

### 6.3.2 Baseline Algorithm

We compare the performance of MUSCAT against single-task learning (STL) and several state-of-the-art multi-task learning algorithms. For these baseline algorithms, the predictor variables from NARR and NCEP reanalysis are concatenated together into a single feature vector to be fitted against the response variable.

1. **STL** (Single Task Learning): In this approach, an independent linear model is incrementally trained for each location using stochastic gradient descent algorithm when a new observation

becomes available. When a weather station is first introduced during the incremental learning process, the parameters of the model are initialized randomly. The parameters are then updated as new observations for the station become available.

2. **ALTO**: This is variation of the spatio-temporal multi-task learning algorithm proposed in [118] which was designed to build models for multiple response variables simultaneously.

3. **WISDOM**: This is a recent spatio-temporal multi-task learning approach proposed in Chapter 5. It applies tensor decomposition on the spatio-temporal data but does not distinguish between features from different scales.

### 6.3.3 Experimental Results

#### 6.3.3.1 Comparison against Baselines

As previously noted, we repeated the incremental learning process 10 times for each method by randomly introducing new station or new time period into the learning process. We then report the mean and standard deviation of MAE over the 10 runs for each method in Table 6.7. The results suggest that WISDOM and MUSCAT outperform STL in all datasets, which shows the effectiveness of using a multi-task learning approach. Although ALTO also employs a multi-task learning strategy, it performs a simple update on its weight matrix, which may not be sufficient to learn the optimal weight for the prediction models. Comparing WISDOM and MUSCAT , the results show that MUSCAT significantly outperforms WISDOM on all four datasets evaluated in this study. This shows the effectiveness of our proposed framework, which uses a multi-tensor decomposition approach to jointly factorize the multi-scale tensors, instead of factorizing a single tensor with concatenated features from multiple scales, which is the approach used in WISDOM .

#### 6.3.3.2 Comparison of Multi-scale analysis against single-scale analysis

In order to determine the value of using multi-scale data, we develop the following two variations of MUSCAT and compare its performance against MUSCAT and WISDOM:

Table 6.7: Mean and standard deviation of MAE for MUSCAT and other baseline methods for climate datasets over 10 trials.

| | tmax | tmin | tmean | prcp |
|---|---|---|---|---|
| STL | $0.4422 \pm 0.0016$ | $0.4412 \pm 0.0020$ | $0.4141 \pm 0.0018$ | $0.5446 \pm 0.0012$ |
| ALTO | $0.5854 \pm 0.0064$ | $0.5687 \pm 0.0031$ | $0.5656 \pm 0.0053$ | $0.5806 \pm 0.0051$ |
| WISDOM | $0.3543 \pm 0.0155$ | $0.4001 \pm 0.0075$ | $0.3850 \pm 0.0236$ | $0.4212 \pm 0.0054$ |
| MUSCAT | $\mathbf{0.3212} \pm 0.0074$ | $\mathbf{0.3454} \pm 0.0065$ | $\mathbf{0.2844} \pm 0.0112$ | $\mathbf{0.4115} \pm 0.0023$ |

1. **MUSCAT-S1** In this variation, MUSCAT uses only predictor variables from NCEP. This variation is equivalent to applying WISDOM on the NCEP data.

2. **MUSCAT-S2**: In this variation, MUSCAT uses only predictor variables from NARR. This variation is equivalent to applying WISDOM to the NARR data.

The mean and standard deviation of MAE over 10 runs are reported in Table 6.8. The results showed that MUSCAT outperforms WISDOM , MUSCAT-S1 and MUSCAT-S2 on all four datasets. Furthermore, WISDOM performs better than both MUSCAT-S1 and MUSCAT-S2 on the temperature datasets, which suggest the importance of incorporating data from multiple scales into the modeling framework. For the precipitation dataset, MUSCAT-S2 performs better than WISDOM . This result suggests that for precipitation, the predictor variables from the coarsest scale (NCEP reanalysis) does not help to predict the precipitation values of the stations. Nonetheless, MUSCAT still achieves the lowest MAE because it can learn the appropriate weight combination ($\alpha$) for NARR and NCEP reanalysis when making its predictions.

Table 6.8: Mean and standard deviation of MAE for MUSCAT against different variations of MUSCAT .

| | tmax | tmin | tmean | prcp |
|---|---|---|---|---|
| WISDOM | $0.3543 \pm 0.0155$ | $0.4001 \pm 0.0075$ | $0.3850 \pm 0.0236$ | $0.4212 \pm 0.0054$ |
| MUSCAT-S1 | $0.5492 \pm 0.0700$ | $0.4328 \pm 0.0115$ | $0.4543 \pm 0.0393$ | $0.6194 \pm 0.0026$ |
| MUSCAT-S2 | $0.3910 \pm 0.0183$ | $0.4094 \pm 0.0186$ | $0.4350 \pm 0.0532$ | $0.4208 \pm 0.0051$ |
| MUSCAT | $\mathbf{0.3212} \pm 0.0074$ | $\mathbf{0.3454} \pm 0.0065$ | $\mathbf{0.2844} \pm 0.0112$ | $\mathbf{0.4115} \pm 0.0023$ |

Table 6.9: Mean of $\alpha_1$ and $\alpha_2$ for the climate datasets over 10 trials.

| | tmax | tmin | tmean | prcp |
|---|---|---|---|---|
| $\alpha_1$ | 0.2876 | 0.3651 | 0.3360 | 0.0933 |
| $\alpha_2$ | **0.7124** | **0.6349** | **0.6640** | **0.9067** |

#### 6.3.3.3 Influence of Data from Multiple Scales

Next, we examine the impact of the data at different scales on the performance of MUSCAT . Our hypothesis is that the predictor variables from coarsest scale are less influential compared to those from the finer scale. To do this, we examine the mean values of the parameters $\alpha_1$ and $\alpha_2$ over 10 runs. The results are shown in Table 6.9. For all four datasets, $\alpha_2$ (NARR) has a consistently higher weight than $\alpha_1$ (NCEP reanalysis). In other words, that NARR predictors are more influential in the prediction of the response variables compared to the NCEP reanalysis predictors. This supports our hypothesis that the finer-level predictors have higher impact than the coarser-level predictors.

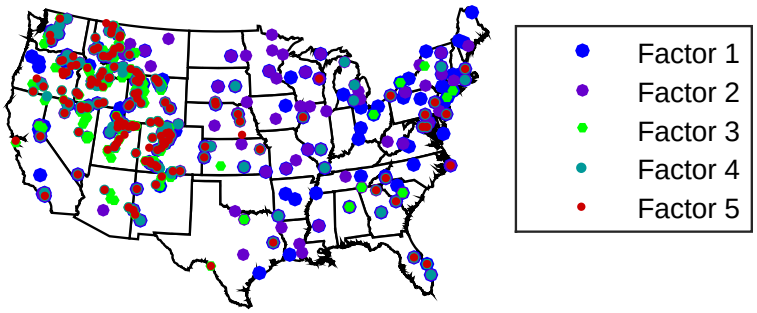



Figure 6.1: Spatial distribution of the spatial latent factors learned by MUSCAT for precipitation data (Figure is best viewed in color)

### 6.3.4 Analysis of Spatial Latent Factors

One of the advantages of using MUSCAT is that the latent factors can be used to identify the spatial patterns of the data. In this section, we examine the spatial latent factors $\mathbf{A}$ generated by MUSCAT . Figure 6.1 shows the spatial distribution of the latent factors for the precipitation data. For each of the 5 spatial latent factors, we plot its corresponding top 20% most influential stations

on the map. As can be seen from Figure 6.1, the distribution for spatial latent factors exhibit certain grouping phenomena. For example, the first latent factor is more dominant on the eastern part of the United States, while the third and fifth latent factors are more influential on the northwest part of the country.

### 6.3.5 Analysis of Temporal Latent Factors

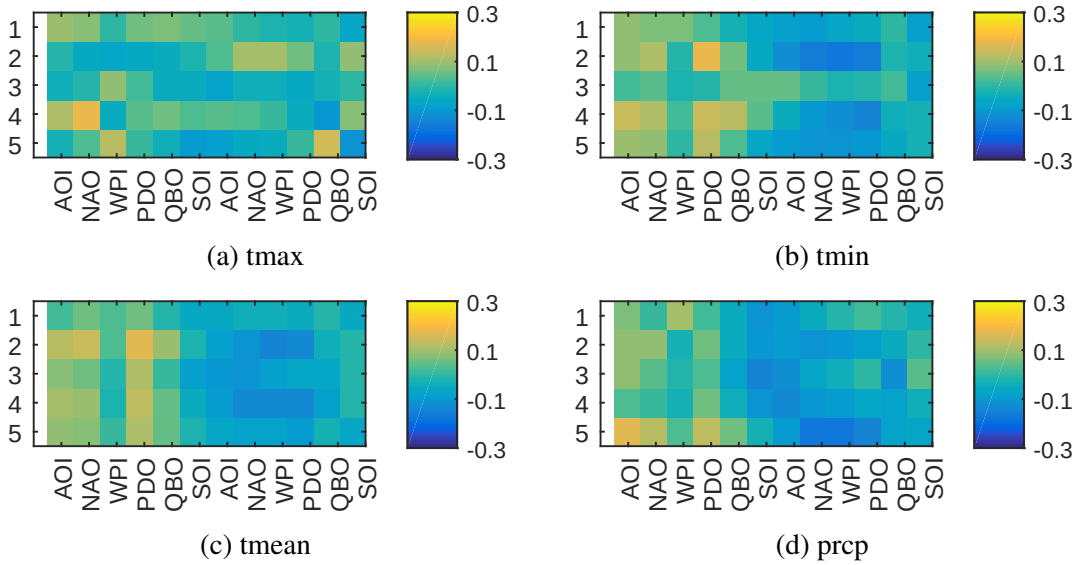
Finally, we examine the temporal latent factors **B** derived by MUSCAT. Since each column in **B** is a time-series, our goal is to determine whether the latent factors capture some of the known climate patterns previously described in the literature. To do this, we computed the correlation between each latent factor against the known climate indices such as El-Niño Southern Oscillation Index (SOI), North Atlantic Oscillation (NAO) index, Arctic Oscillation Index (AOI), etc. The description of the climate indices are shown in Table 6.10. Figure 6.2 shows the correlation between the selected climate indices and the temporal latent factors generated by MUSCAT for all four datasets. It is expected that the overall correlation may not be that high since the datasets used in this study are limited to weather stations in the United States, whereas most climate indices are defined over other regions of the world. Nevertheless, we still observe relatively high correlation between some indices and the temporal latent factors found. For example, in the tmax dataset, the fourth latent factor has a relatively high correlation with NAO. In the tmin and tmean datasets, the second latent factor has a high correlation with PDO, while for the prcp dataset, the fifth latent factor has a relatively high correlation with AOI. This result suggests that the temporal latent factors generated by MUSCAT was able to capture some of the previously known climate phenomena, which was represented by the climate indices.

## 6.4 Conclusion

This chapter presents a multi-scale multi-task learning framework for geospatio-temporal data by employing a supervised multi-tensor decomposition approach. The framework enables the multi-scale relationships to be harnessed by enforcing a constraint on the consistency between

Table 6.10: List of the climate indices used to correlate with the temporal factors learned from WISDOM.

| Climate Index | Description |
|---|---|
| AOI | Arctic Oscillation Index |
| NAO | North Atlantic Oscillation |
| WPI | West Pacific Pattern |
| QBO | Quasi-Biennial Oscillation |
| PDO | Pacific Decadal Oscillation |
| SOI | Southern Oscillation Index |




(a) tmax (b) tmin

(c) tmean (d) prcp

Figure 6.2: Correlations between the known climate indices and the temporal latent factors derived from MUSCAT .

the spatial and temporal latent factors derived from the multi-scale geospatio-temporal data. An incremental learning algorithm over space and time is then proposed to efficiently learn the weights of the model. Experiments performed on a real-world multi-scale climate dataset demonstrate the effectiveness of proposed method compared to several baseline algorithms.

# CHAPTER 7

# CONCLUSIONS AND FUTURE WORK

In this chapter, the major contributions of this thesis are summarized and possible future research directions are discussed.

## 7.1 Summary of Contributions

Geospatio-temporal data mining tasks can be naturally cast into MTL problems in spatial and temporal dimensions. This thesis proposed four novel MTL frameworks to investigate the possibility of applying MTL approaches on geospatio-temporal domain by addressing the challenges of building effective predictive models for geospatio-temporal data. It also evaluated the performance of the proposed MTL frameworks on different geospatio-temporal applications. The contribution of this thesis is summarized in the following.

In Chapter 3, an online temporal MTL framework named ORION is proposed to learn the optimal weights for ensemble members in applications of ensemble forecasting. The framework considers the prediction for each time point in the forecast window as one task, and the task relatedness is defined by the mean regularizer and Laplacian regularizer to constrain the temporal smoothness between tasks. ORION adopts a restart strategy to handle the missing value of the target variable in the forecast window. Various loss functions such as $\varepsilon$-insensitive and quantile loss functions can be adopted in the framework for different objectives. The experimental results on real world soil moisture data from 12 US river basins demonstrate the superior performance of ORION over other baselines.

In Chapter 4, in order to perform multi-location prediction, a MTL framework named GSpartan is developed to build models jointly at multiple geo-locations. The framework assumes that the models are sharing a set of hidden models and linearly combined by these hidden models. Spatial autocorrelation is also employed to explicitly define the relationship between tasks. Sparsity and non-negativity are also enforced in the objective to ensure interpretability of the learned models.

Experimental results on real world climate data show that GSpartan outperforms other baselines especially when there are limited training examples at each location.

GSpartan is limited in that it only considers the spatial autocorrelation of the data and it is difficult to scale up to global-scale data. In Chapter 5, a spatio-temporal MTL framework is proposed for multi-location prediction based on supervised tensor decomposition. This framework constructs multiple tasks across both space and time, and the task relations in spatial and temporal dimension are implicitly determined by the tensor decomposition. A novel incremental learning algorithm called WISDOM is developed to simultaneously learn the latent factors of the spatio-temporal data via tensor decomposition, as well as the prediction models. WISDOM outperforms several baseline algorithms and can easily accommodate existing patterns from the geospatio-temporal domain.

In Chapter 6, in order to handle data from multiple scales, a multi-scale spatio-temporal MTL framework is developed for multi-scale modeling on multiple locations simultaneously. In addition to the merit of WISDOM, this framework models the relatedness of the predictors from different resolution or scales, by considering the consistency of the spatial and temporal latent factors learned from the predictor tensor of different scales. A novel incremental multi-tensor decomposition algorithm called MUSCAT is developed to solve the objective efficiently. Experiments on real world climate data show the effectiveness of MUSCAT over other baselines, including WISDOM.

## 7.2 Future Directions

Although the results for the proposed MTL frameworks are promising, they are still limited in a few aspects. In this section, the future research directions are discussed, which are inspired from the work of this thesis.

The proposed MTL frameworks in this thesis assume the prediction function to be a linear form, which might not be true in some of the geospatio-temporal applications, such as climate and Earth science. As future work, the choice of non-linear prediction functions or kernel functions needs to be studied.

In addition, the features used in the proposed frameworks for multi-location prediction are only a very small portion of the whole feature set, which are mainly determined by the domain experts. However, removing the features outside of the hand-picked feature set might result in information loss. While it might not be effective and efficient to directly use all possible features, it is a challenge to perform feature selection and feature learning such that the learned features with a limited number of size can be effectively represent the original feature set. Since deep learning has been proposed in literature to learn informative features, as future work, MTL framework with feature learning can be proposed by incorporating the merit of deep learning approaches.

For many spatio-temporal applications, preserving the fidelity of the distribution is as important as minimizing the prediction error [2]. Although the proposed MTL frameworks do improve the regression prediction performance over the state-of-the-art methods, they still suffer from the fact that the regression will not always preserve the true distribution of the data. As future work, in order to address this problem, multi-task contour regression can be explored to simultaneously minimize the prediction error and maximize the alignment between the distributions of prediction and observation.

**BIBLIOGRAPHY**

# BIBLIOGRAPHY


[1] Zubin Abraham, Malgorzata Liszewska, Perdinan, Pang-Ning Tan, Julie Winkler, and Shiyuan Zhong. Distribution regularized regression framework for climate modeling. In *SDM*, pages 333–341. SIAM, 2013.

[2] Zubin Abraham, Pang-Ning Tan, Perdinan, Julie A. Winkler, Shiyuan Zhong, and Malgorzata Liszewska. Contour regression: A distribution-regularized regression framework for climate modeling. *Stat. Anal. Data Min.*, 7(4):272–281, August 2014. ISSN 1932-1864.

[3] Alekh Agarwal, Alexander Rakhlin, and Peter Bartlett. Matrix regularization techniques for online multitask learning. Technical Report UCB/EECS-2008-138, EECS Department, University of California, Berkeley, Oct 2008. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-138.html`.

[4] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, December 2005. ISSN 1532-4435.

[5] Luc Anselin and Arthur Getis. Spatial statistical analysis and geographic information systems. *The Annals of Regional Science*, 26(1):19–33, 1992. ISSN 1432-0592.

[6] Miguel B. Araújo and Mark New. Ensemble forecasting of species distributions. *Trends in Ecology & Evolution*, 22(1):42–47, 2007.

[7] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Mach. Learn.*, 73(3):243–272, December 2008. ISSN 0885-6125.

[8] N. H. Augustin, M. Musio, K. von Wilpert, E. Kublin, S. N. Wood, and M. Schumacher. Modeling spatiotemporal forest health monitoring data. 104:899–911, 2009.

[9] Prithu Banerjee, Pranali Yawalkar, and Sayan Ranu. Mantra: A scalable approach to mining temporally anomalous sub-trajectories. In *KDD*, pages 1415–1424, 2016.

[10] Sudipto Banerjee, Bradley P. Carlin, and Alan E. Gelfand. *Hierarchical Modeling and Analysis for Spatial Data*. Monographs on Statistics and Applied Probability. Chapman and Hall/CRC, 1 edition, 2004. ISBN 158488410X.

[11] F. Bellocchio, S. Ferrari, V. Piuri, and N.A. Borghese. Hierarchical approach for multiscale support vector regression. *IEEE Transactions on Neural Networks and Learning Systems*, 23(9):1448–1460, September 2012. 1045-9227.

[12] Gedas Bertasius, Jianbo Shi, and Lorenzo Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. In *CVPR*, pages 4380–4389, 2015.

[13] Martin Andrew Bezener. Bayesian spatiotemporal modeling using spatial hierarchical priors with applications to functional magnetic resonance imaging. *Ph.D dissertation, University of Minnesota*, 2015.

[14] Jinbo Bi, Tao Xiong, Shipeng Yu, Murat Dundar, and R.Bharat Rao. An improved multi-task learning approach with applications in medical diagnosis. In *Machine Learning and Knowledge Discovery in Databases*, volume 5211 of *Lecture Notes in Computer Science*, pages 117–132. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-87478-2.

[15] Steffen Bickel, Jasmina Bogojeska, Thomas Lengauer, and Tobias Scheffer. Multi-task learning for hiv therapy screening. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 56–63, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

[16] Shyam Boriah, Vipin Kumar, Michael Steinbach, Christopher Potter, and Steven Klooster. Land cover change detection: A case study. In *KDD*, pages 857–865, 2008.

[17] George Edward Pelham Box and Gwilym Jenkins. *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated, 1990. ISBN 0816211043.

[18] Jorge Caiado and Nuno Crato. A garch-based method for clustering of financial time series: International stock markets evidence. Mpra paper, University Library of Munich, Germany, 2007. URL `http://EconPapers.repec.org/RePEc:pra:mprapa:2074`.

[19] Pierre Cantelaube and Jean-Michel Terres. Seasonal weather forecasts for crop yield modelling in europe. *Tellus A*, 57(3):476–487, 2005.

[20] Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, July 1997. ISSN 0885-6125.

[21] Giovanni Cavallanti, Nicolò Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *Journal of Machine Learning Research*, 11:2901–2934, December 2010. ISSN 1532-4435.

[22] Kai-Wei Chang, Wen-Tau Yih, Bishan Yang, and Christopher Meek. Typed tensor decomposition of knowledge bases for relation extraction. In *EMNLP*, 2014.

[23] Jianhui Chen, Jiayu Zhou, and Jieping Ye. Integrating low-rank and group-sparse structures for robust multi-task learning. In *Proceedings of the 17th ACM SIGKDD Int'l Conf on Knowledge discovery and data mining*, pages 42–50. ACM, 2011.

[24] Jianhui Chen, Ji Liu, and Jieping Ye. Learning incoherent sparse and low-rank patterns from multiple tasks. *ACM Transactions on Knowledge Discovery from Data*, 5(4):22:1–22:31, 2012. ISSN 1556-4681.

[25] Jianhui Chen, Lei Tang, Jun Liu, and Jieping Ye. A convex formulation for learning a shared predictive structure from multiple tasks. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(5):1025–1038, May 2013. ISSN 0162-8828.

[26] Haibin Cheng and Pang-Ning Tan. Semi-supervised learning with data calibration for long-term time series forecasting. In *Proc of ACM SIGKDD Int'l Conf on Knowledge Discovery and Data Mining*, pages 133–141, 2008.

[27] Chi-Yin Chow and Mohamed F. Mokbel. Trajectory privacy in location-based services and data publication. *SIGKDD Explor. Newsl.*, 13(1):19–29, August 2011.

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

[29] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, December 2006. ISSN 1532-4435.

[30] N. Cressie. *Statistics for spatial data*. Wiley, New York, 1993.

[31] Hal Daumé, III. Bayesian multitask learning with latent hierarchies. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 135–142, Arlington, Virginia, United States, 2009. AUAI Press. ISBN 978-0-9749039-5-8.

[32] G.S. Davis, N Sevdalis, and L.N Drumright. Spatial and temporal analyses to investigate infectious disease transmission within healthcare settings. pages 227–243, 2014.

[33] Victor De Oliveira. Bayesian analysis of conditional autoregressive models. *Annals of the Institute of Statistical Mathematics*, 64(1):107–133, 2012.

[34] Ofer Dekel, Philip M. Long, and Yoram Singer. Online learning of multiple tasks with a shared loss. *Journal of Machine Learning Research*, 8:2233–2264, December 2007. ISSN 1532-4435.

[35] N. Diodato, G. Bellocchi, C. Bertolin, and D. Camuffo. Multiscale regression model to infer historical temperatures in a central mediterranean sub-regional area. *Climate of the Past Discussions*, 6:2625–3649, 2010.

[36] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 264–271, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

[37] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, pages 2366–2374, 2014.

[38] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 109–117, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1.

[39] Theodoros Evgeniou, Charles A. Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. *J. Mach. Learn. Res.*, 6:615–637, December 2005. ISSN 1532-4435.

[40] JamesH. Faghmous and Vipin Kumar. Spatio-temporal data mining for climate data: Advances, challenges, and opportunities. In *Data Mining and Knowledge Discovery for Big Data*, volume 1 of *Studies in Big Data*, pages 83–116. Springer Berlin Heidelberg, 2014.

[41] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *PAMI*, 35(8):1915–1929, Aug 2013.

[42] Robert J. Franzese, Jude C. Hays, and Scott J. Cook. Spatial- and spatiotemporal-autoregressive probit models of interdependent binary outcomes. *Political Science Research and Methods*, 4(1):151âĂŞ173, 2016.

[43] Scott John Gaffney. *Probabilistic Curve-Aligned Clustering and Prediction with Regression Mixture Models*. PhD thesis, 2004.

[44] Alan E Gelfand, Peter J Diggle, Montserrat Fuentes, and Peter Guttorp. *Handbook of spatial statistics*. CRC press, 2010.

[45] Fosca Giannotti, Mirco Nanni, Fabio Pinelli, and Dino Pedreschi. Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '07, pages 330–339. ACM, 2007.

[46] A. R. GonÃğalves, F. J. Von Zuben, and A. Banerjee. A multitask learning view on the earth system model ensemble. *Computing in Science Engineering*, 17(6):35–42, Nov 2015.

[47] A. R. GonÃğalves, A. Banerjee, and F. J. Von Zuben. Spatial projection of multiple climate variables using hierarchical multitask learning. In *AAAI*, 2017.

[48] André R. Gonçalves, Fernando J. Von Zuben, and Arindam Banerjee. Multi-task sparse structure learning with gaussian copula models. *JMLR*, 17(1):1205–1234, January 2016.

[49] Pinghua Gong, Jieping Ye, and Changshui Zhang. Robust multi-task feature learning. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 895–903, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1462-6.

[50] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.0 beta. `http://cvxr.com/cvx`, September 2013.

[51] Tony H Grubesic and Alan T Murray. Detecting hot spots using cluster analysis and gis. In *Proceedings from the fifth annual international crime mapping research conference*, volume 26, 2001.

[52] Quanquan Gu, Zhenhui Li, and Jiawei Han. Learning a kernel for multi-task clustering. In *Proceedings of the 25th Conference on Artificial Intelligence (AAAI)*, 2011.

[53] Forrest M Hoffman, William W Hargrove, and Anthony D Del Genio. Multivariate spatio-temporal clustering of time-series data: an approach for diagnosing cloud properties and understanding arm site representativeness.

[54] Sahyun Hong and W. M. Moon. Application of gaussian markov random field model to unsupervised classification in polarimetric sar image. In *2003 IEEE International Geoscience and Remote Sensing Symposium. Proceedings*, volume 2, pages 929–931 vol.2, 2003.

[55] Weiming Hu, Xi Li, Xiaoqin Zhang, Xinchu Shi, Stephen Maybank, and Zhongfei Zhang. Incremental tensor subspace learning and its applications to foreground segmentation and tracking. *IJCV*, 91(3):303–327, 2011.

[56] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *Proceedings of the 28st International Conference on Machine Learning*, pages 521–528. Omnipress, 2011.

[57] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In *RecSys 2010*, pages 79–86, 2010.

[58] Baris M. Kazar and Mete Celik. *Spatial AutoRegression (SAR) Model: Parameter Estimation Techniques*. Springer Publishing Company, Incorporated, 2012.

[59] Eamonn Keogh, Jessica Lin, and Ada Fu. Hot sax: Efficiently finding the most unusual time series subsequence. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, ICDM '05, pages 226–233, 2005. ISBN 0-7695-2278-5.

[60] Roger Koenker. *Quantile Regression*. Econometric Society Monographs. Cambridge University Press, 2005. ISBN 0521608279.

[61] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, August 2009.

[62] Krzysztof Koperski and Jiawei Han. Discovery of spatial association rules in geographic information databases. In *Proceedings of the 4th International Symposium on Advances in Spatial Databases*, SSD '95, pages 47–66, 1995.

[63] Abhishek Kumar and Hal DaumÂÍÂę III. Learning task grouping and overlap in multi-task learning. In *Proceedings of the 29st International Conference on Machine Learning*. icml.cc / Omnipress, 2012.

[64] Neil D. Lawrence and John C. Platt. Learning to learn with the informative vector machine. In *Proceedings of the 21st International Conference on Machine Learning*, ICML '04.

[65] Su-In Lee, Vassil Chatalbashev, David Vickrey, and Daphne Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 489–496, New York, NY, USA, 2007. ACM. ISBN 978-1-59593-793-3.

[66] M. Leutbecher and T.N. Palmer. Ensemble forecasting. *Journal of Computational Physics*, 227:3515–3539, 2008.

[67] Guangxia Li, S.C.H. Hoi, Kuiyu Chang, Wenting Liu, and R. Jain. Collaborative online multitask learning. *Knowledge and Data Engineering, IEEE Transactions on*, 26(8):1866–1876, Aug 2014. ISSN 1041-4347. doi: 10.1109/TKDE.2013.139.

[68] Mu Li, Amr Ahmed, and Alexander J. Smola. Inferring movement trajectories from gps snippets. In *Proceedings of the 8th ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 325–334, 2015.

[69] Nick Littlestone and Manfred K. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, February 1994. ISSN 0890-5401.

[70] Chang-Tien Lu and Lily R Liang. Wavelet fuzzy classification for detecting and tracking region outliers in meteorological data. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*, pages 258–265. ACM, 2004.

[71] Chang-Tien Lu, Dechang Chen, and Yufeng Kou. Algorithms for spatial outlier detection. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 597–, 2003. ISBN 0-7695-1978-4.

[72] Lifeng Luo and Eric F. Wood. Use of bayesian merging techniques in a multimodel seasonal hydrologic ensemble prediction system for the eastern united states. *Journal of Hydrometeorology*, 9:866–884, 2008.

[73] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Min. Knowl. Discov.*, 1(3):259–289, January 1997. ISSN 1384-5810.

[74] Wesley Mathew, Ruben Raposo, and Bruno Martins. Predicting future locations with hidden markov models. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 911–918, 2012.

[75] Bradley A. Miller, Sylvia Koszinski, Marc Wehrhan, and Michael Sommer. Impact of multi-scale predictor selection for modeling soil properties. *Geoderma*, 239âĂŞ240:97 – 106, 2015. ISSN 0016-7061.

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

[77] Claire Monteleoni, Gavin A. Schmidt, and Shailesh Saroha. Tracking climate models. In *NASA Conference on Intelligent Data Understanding*, pages 1–15. NASA Ames Research Center, 2010.

[78] Natalia Neverova, Christian Wolf, Graham W. Taylor, and Florian Nebout. *Multi-scale Deep Learning for Gesture Detection and Localization*, pages 474–490. 2014.

[79] Xia Ning and George Karypis. Multi-task learning for recommender system. In *Proceedings of the 2nd Asian Conference on Machine Learning*, volume 13 of *JMLR Proceedings*, pages 269–284. JMLR.org, 2010.

[80] Mohamed N. Nounou and Hazem N. Nounou. Multiscale latent variable regression. *International Journal of Chemical Engineering*, 2010:8, 2010.

[81] Guillaume Obozinski, Ben Taskar, and MichaelI. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20(2): 231–252, 2010. ISSN 0960-3174.

[82] K.P. Overmars, G.H.J. de Koning, and A. Veldkamp. Spatial autocorrelation in multi-scale land use models. *Ecological Modelling*, 164(2âĂŞ3):257 – 270, 2003. ISSN 0304-3800.

[83] R. Kelley Pace, Ronald Barry, John M. Clapp, and Mauricio Rodriquez. Spatiotemporal autoregressive models of neighborhood effects. *The Journal of Real Estate Finance and Economics*, 17(1):15–33, 1998.

[84] Neal Parikh and Stephen Boyd. Proximal algorithms. *Found. Trends Optim.*, 1(3):127–239, January 2014. ISSN 2167-3888.

[85] A. Riccio, G. Barone, E. Chianese, and G. Giunta. A hierarchical bayesian approach to the spatio-temporal modeling of air quality data. *Atmospheric Environment*, 40(3):554 – 566, 2006.

[86] Bernardino Romera-Paredes, Hane Aung, Nadia Bianchi-Berthouze, and Massimiliano Pontil. Multilinear multitask learning. In *ICML*, pages 1444–1452.

[87] Avishek Saha, Piyush Rai, Hal DaumÂÍÂę III, and Suresh Venkatasubramanian. Online learning of multiple tasks and their relationships. In *AISTATS*, volume 15 of *JMLR Proceedings*, pages 643–651. JMLR.org, 2011.

[88] Shashi Shekhar, Zhe Jiang, Reem Y. Ali, Emre Eftelioglu, Xun Tang, Venkata M. V. Gunturi, and Xun Zhou. Spatiotemporal data mining: A computational perspective. *ISPRS International Journal of Geo-Information*, 4(4):2306–2338, 2015.

[89] Balaji Vasan Srinivasan, Ramani Duraiswami, and Raghu Murtugudde. Efficient kriging for real-time spatio-temporal interpolation. In *PSAS*, pages 228–235, 2010.

[90] Michael Steinbach, Pang-Ning Tan, Vipin Kumar, Steven Klooster, and Christopher Potter. Discovery of climate indices using clustering. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, pages 446–455, 2003.

[91] Karthik Subbian and Arindam Banerjee. Climate multi-model regression using spatial smoothing. In *SDM*, pages 324–332, 2013.

[92] Hua Sun, Yong Tu, and Shi-Ming Yu. A spatio-temporal autoregressive model for multiunit residential market analysis. *The Journal of Real Estate Finance and Economics*, 31(2): 155–187, 2005.

[93] Jimeng Sun, Dacheng Tao, Spiros Papadimitriou, Philip S. Yu, and Christos Faloutsos. Incremental tensor analysis: Theory and applications. *TKDD*, 2(3):11:1–11:37.

[94] Jimeng Sun, Dacheng Tao, and Christos Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *KDD*, pages 374–383, 2006.

[95] C. Sung, D. Feldman, and D. Rus. Trajectory clustering for motion prediction. In *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1547–1552, Oct 2012.

[96] Yufei Tao, George Kollios, Jeffrey Considine, Feifei Li, and Dimitris Papadias. Spatio-temporal aggregation using sketches. In *Proceedings of the 20th International Conference on Data Engineering*, ICDE '04, pages 214–, 2004. ISBN 0-7695-2065-0.

[97] Claudia Tebaldi and Reto Knutti. The use of the multi-model ensemble in probabilistic climate projections. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 365(1857):2053–2075, 2007.

[98] W. Tobler. A computer movie simulating urban growth in the detroit region. *Economic Geography*, 46(2):234–240, 1970.

[99] Pedro A. Valdes-Sosa. Spatio-temporal autoregressive models defined over brain manifolds. *Neuroinformatics*, 2(2):239–250, 2004.

[100] Christian Walder, Kwang In Kim, and Bernhard Schölkopf. Sparse multiscale gaussian process regression. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 1112–1119, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

[101] Jialei Wang, Steven C.H. Hoi, Peilin Zhao, and Zhi-Yong Liu. Online multi-task collaborative filtering for on-the-fly recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, pages 237–244. ACM, 2013.

[102] Weiran Wang and Miguel Á. Carreira-Perpiñán. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. *CoRR*, abs/1309.1541, 2013. URL `http://arxiv.org/abs/1309.1541`.

[103] Xiaogang Wang, Cha Zhang, and Zhengyou Zhang. In *Proceedings of the 22nd IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '09.

[104] Christopher K. Wikle, Ralph F. Milliff, Doug Nychka, and L. Mark Berliner. Spatiotemporal hierarchical bayesian modeling: Tropical ocean surface winds. *Journal of the American Statistical Association*, 96(454):382–397, 2001.

[105] Kishan Wimalawarne, Masashi Sugiyama, and Ryota Tomioka. Multitask learning meets tensor factorization: task imputation via convex optimization. In *NIPS*, pages 2825–2833. 2014.

[106] Fei Wu, Xu Tan, Yi Yang, Dacheng Tao, Siliang Tang, and Yueting Zhuang. Supervised nonnegative tensor factorization with maximum-margin constraint. In *AAAI 2013*, 2013.

[107] Jianpeng Xu, Pang-Ning Tan, and Lifeng Luo. ORION: Online Regularized multI-task regressiON and its application to ensemble forecasting. In *Proceedings of the 14th IEEE International Conference on Data Mining*, ICDM '14.

[108] Jianpeng Xu, Jiayu Zhou, and Pang-Ning Tan. Formula: Factorized multi-task learning for task discovery in personalized medical models. In *Proceedings of the 15th SIAM International Conference on Data Mining*, SDM '15, pages 496–504, 2015.

[109] Jianpeng Xu, Pang-Ning Tan, Lifeng Luo, and Jiayu Zhou. Gspartan: a geospatio-temporal multi-task learning framework for multi-location prediction. In *SDM 2016*, 2016.

[110] Jianpeng Xu, Jiayu Zhou, Pang-Ning Tan, Xi Liu, and Lifeng Luo. Wisdom: Weighted incremental spatio-temporal multi-task learning via tensor decomposition. In *IEEE Big Data*, pages 522–531, Dec 2016.

[111] Jianpeng Xu, Pang-Ning Tan, Jiayu Zhou, and Lifeng Luo. Online multi-task learning framework for ensemble forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 29(6):1268–1280, 2017.

[112] Yangyang Xu and Wotao Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal of Imaging Sciences*, 6:1758–1789, 2013.

[113] Ya Xue, Xuejun Liao, Lawrence Carin, and Balaji Krishnapuram. Multi-task learning for classification with dirichlet process priors. *Journal of Machine Learning Research*, 8:35–63, May 2007. ISSN 1532-4435.

[114] Haiqin Yang, Irwin King, and Michael R. Lyu. Online learning for multi-task feature selection. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1693–1696. ACM, 2010. ISBN 978-1-4503-0099-5.

[115] Hui Yang, Srinivasan Parthasarathy, and Sameep Mehta. A generalized framework for mining spatio-temporal patterns in scientific data. In *Proc of ACM SIGKDD Int'l Conf on Knowledge Discovery and Data Mining*, pages 716–721, 2005.

[116] Kai Yu, Volker Tresp, and Anton Schwaighofer. Learning gaussian processes from multiple tasks. In *Proceedings of the 22Nd International Conference on Machine Learning*, ICML '05, pages 1012–1019, New York, NY, USA, 2005. ACM. ISBN 1-59593-180-5.

[117] Rose Yu, Mohammad Taha Bahadori, and Yan Liu. Fast multivariate spatio-temporal analysis via low rank tensor learning. In *NIPS*, pages 3491–3499, 2014.

[118] Rose Yu, Dehua Cheng, and Yan Liu. Accelerated online low rank tensor learning for multivariate spatiotemporal streams. In *ICML 2015*, volume 37, pages 238–247, 2015.

[119] Xiao-Tong Yuan and Shuicheng Yan. Visual classification with multi-task joint sparse representation. In *Proceedings of the 23rd IEEE Conference on Computer Vision and Pattern Recognition*, pages 3493–3500, June 2010.

[120] Yu Zhang and Dit-Yan Yeung. A convex formulation for learning task relationships in multi-task learning. In *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 733–442. AUAI Press, 2010. ISBN 978-0-9749039-6-5.

[121] Liang Zhao, Qian Sun, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. Multi-task learning for spatio-temporal event forecasting. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1503–1512, 2015.

[122] Wenzhi Zhao and Shihong Du. Learning multiscale and deep representations for classifying remotely sensed imagery. *Journal of Photogrammetry and Remote Sensing*, 113:155 – 165, 2016.

[123] Xu Zhong, Allison Kealy, and Matt Duckham. Stream kriging. *Computers & Geosciences*, 90(PA):134–143, May 2016.

[124] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard SchÃűlkopf. Learning with local and global consistency. In *The 16th Annual Conference on Neural Information Processing Systems*, pages 321–328. MIT Press, 2004.

[125] J. Zhou, J. Chen, and J. Ye. *MALSAR: Multi-tAsk Learning via StructurAl Regularization*. Arizona State University, 2011. URL `http://www.public.asu.edu/~jye02/Software/MALSAR`.

[126] Jiayu Zhou, Lei Yuan, Jun Liu, and Jieping 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*, KDD '11, pages 814–822, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0813-7.

[127] Shuo Zhou, Xuan Vinh Nguyen, James Bailey, Yunzhe Jia, and Ian Davidson. Accelerating Online CP Decompositions for Higher Order Tensors. In *KDD*, 2016.