Online Multi-task Learning Framework for Ensemble Forecasting

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

Abstract—Ensemble forecasting is a widely-used numerical prediction method for modeling the evolution of nonlinear dynamic systems. To predict the future state of such systems, a set of ensemble member forecasts is generated from multiple runs of computer models, where each run is obtained by perturbing the starting condition or using a different model representation of the system. The ensemble mean or median is typically chosen as a point estimate for the ensemble member forecasts. These approaches are limited in that they assume each ensemble member is equally skillful and may not preserve the temporal autocorrelation of the predicted time series. To overcome these limitations, we present an online multi-task learning framework called *ORION* to estimate the optimal weights for combining the ensemble member forecasts. Unlike other existing formulations, the proposed framework is novel in that its learning algorithm must backtrack and revise its previous forecasts before making future predictions if the earlier forecasts were incorrect when verified against new observation data. We termed this strategy as *online learning with restart*. Our proposed framework employs a graph Laplacian regularizer to ensure consistency of the predicted time series. It can also accommodate different types of loss functions, including ϵ-insensitive and quantile loss functions, the latter of which is particularly useful for extreme value prediction. A theoretical proof demonstrating the convergence of our algorithm is also given. Experimental results on seasonal soil moisture forecasts from 12 major river basins in North America demonstrate the superiority of ORION compared to other baseline algorithms.

Index Terms—Online learning, multi-task learning, ensemble forecasting.

1 Introduction

Ensemble forecasting is a popular numerical prediction method for modeling nonlinear dynamic systems, such as climate [1], agriculture [2], ecological [3], and hydrological [4] 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 [5] 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. 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. Our 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. 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 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 paper 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*. Multi-task learning has been successfully used to solve multiple related learning problems in many applications, including computer vision [6] [7], healthcare [8] [9], recommender systems [10]

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

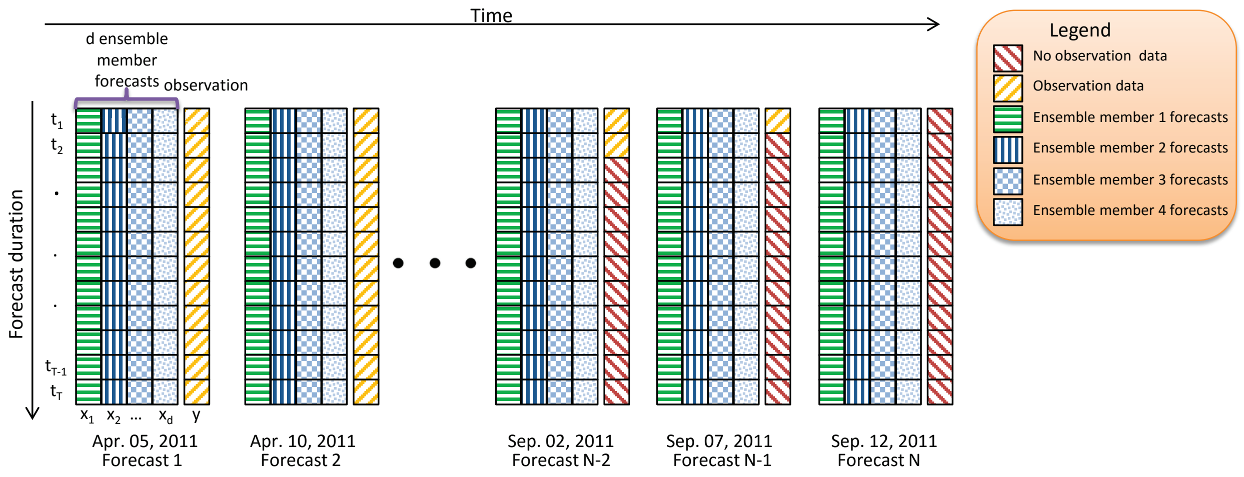


Fig. 1: A schematic illustration of ensemble forecasting task (diagram is best viewed in color). Assuming the latest forecast was generated on September 12, 2011 for the time period between September 17 and October 22, there is one observation value available to verify Forecast $(N-1)$, two observations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

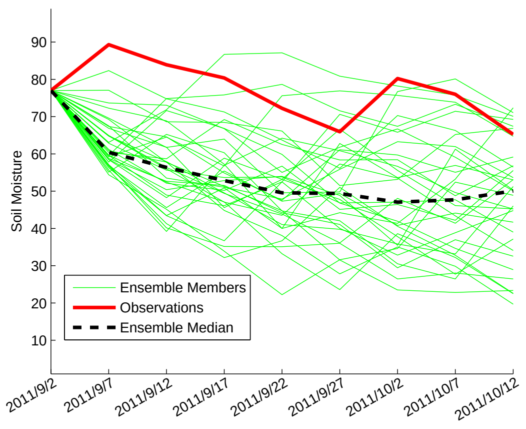


Fig. 2: Seasonable soil moisture forecasts and the observed time series at a major river basin in North America.

[11], natural language processing [12] and genomics [13]. In particular, previous studies have shown that the predictive performance of various learning tasks can be improved by exploiting the commonalities among the tasks.

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. 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 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.

In this paper, we 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 in the model parameters while taking into account the temporal autocorrelation of the predictions within each time window. A preliminary version of this work appeared in [14], where the ORION framework was introduced using the ϵ-insensitive loss function to predict the weighted conditional mean of the ensemble member predictions. In this journal submission, the framework is extended to incorporate a quantile loss function, which is useful for predicting extreme values of a time series. Forecasting of extreme values is essential for applications such as climate change assessment and natural resource management due to their potential impact on human and natural systems. We also provide a theoretical proof demonstrating the convergence of our online learning algorithm. Finally, new experiments are added to evaluate sensitivity of the results to changes in the parameter setting of the ORION framework.

The main contributions of this paper are as follows:

- We introduce the problem of online regularized multi-task regression with partially observed data and demonstrate its relevance to the ensemble forecasting task.
- We present a novel framework called ORION, 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 ϵ-insensitive loss function.
- We extended the framework to incorporate a 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.

- We performed extensive experiments using a real-world soil moisture data set and showed that ORION outperforms several baseline algorithms, including the ensemble median, for the majority of the river basins in our data set.

The remainder of this paper is organized as follows. Section 2 reviews the related work of this research. Section 3 formalizes the problem while Section 4 presents the ORION framework with ϵ-insensitive loss function and its solution. Section 5 extends the framework to a quantile loss function. Section 6 presents the experimental results for seasonal soil moisture prediction. Finally, Section 7 summarizes the work and presents our concluding remarks.

2 Related Work

This section reviews some of the previous works on topics closely related to this paper. An ensemble forecasting task requires predicting the future values of a time series over a finite time window, which is quite similar to the multi-step ahead time series prediction problem [15] [16]. Nevertheless, there is a fundamental difference between the two prediction problems. Multi-step-ahead time series prediction methods consider only the historical values of a time series to infer its future values. Thus, it is susceptible to the error accumulation problem [15]. In contrast, ensemble forecasting methods employ multivariate time series generated from computer models to predict the future values of a time series. These models generate their outputs by considering the physical processes that govern the evolution of the dynamic system.

2.1 Multi-task Learning

Multi-task learning [17] is an approach designed to improve predictive performance by learning from multiple related tasks simultaneously, taking into account the relationships and information shared among the different tasks. Depending on how information is shared among the different tasks, multi-task learning can be broadly classified into four categories. (a) **Low-rank representation**: This category of methods assumes that the models for different tasks share a low-rank representation, either additively [18], [19], [20] or multiplicatively [21], [22], [23]. Chen et al. [24] incorporated both additive and multiplicative representation into their formulation with a low-dimensional feature map shared across the different tasks. (b) **Explicit task relationship**: This category of methods explicitly incorporates the task relationship into the multi-task learning formulation. For example, Zhou et al. [8] employed a task relationship to ensure smoothness between time series predictions. Zhang et al. [25] proposed a regularization formulation to simultaneously learn the task relationship and task models. (c) **Parameter sharing**: These methods assume that the different tasks share a common set of parameters. For example, Lawrence et al. [26] and Yu et al. [27] proposed a multi-task Gaussian process where the prior parameters are shared across different generative processes. Lee et al. [28] assumed sharing of hyperparameters between the distributions for different task models while Daume et al. [29] learned hierarchical Bayesian models that share a common structure parameterized by a covariance matrix. (d) **Hybrid information sharing**: There have been some recent attempts to combine the information sharing methods described above. For example, Xu et al. [30] presented a formulation that combines both low-rank representation and explicit task relations for developing personalized medical models of patients.

2.2 Online Multi-task Learning

Since the amount of data to be processed in multi-task learning is generally much larger than single-task learning, an obvious solution is to extend multi-task learning to an online learning setting. Online learning is a family of learning algorithms, in which the data are observed in a sequential manner and the model must be updated incrementally based on the new observations. Some of the popular online learning methods include the weighted majority algorithm [31], online Passive Aggressive (PA) algorithm [32], and confidence-weighted algorithm [33].

Dekel et al. [34] presented an approach for online multi-task perceptron using a global loss function to define the relationships among the different tasks. Unlike our proposed work, this method assumes the observation values for all the learning tasks are available when the model is updated. Relationships among the tasks are also not explicitly defined in their objective function. Another perceptron-based online multi-task learning method was proposed in [35], 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. Our method overcomes this problem by incorporating the task relationship into the objective function and providing an update solution that allows all the tasks to be updated simultaneously. Saha et al. [36] extended the work in [35] by proposing a method that simultaneously learns the model and task relation matrix using an alternating optimization scheme. Li et al. [37] proposed a collaborative online multi-task learning approach, which is an extension of the method given in [38] to an online learning setting. Sun et al. [39] employs a stochastic gradient descent approach to solve its multi-task learning problem. Another recent work on online multi-task learning considers a regret-minimization approach under expert advice model [40]. None of the previous works are designed for partially observed data or for predicting extreme events in a time series.

3 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], [35], 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 paper.

From an ensemble forecasting perspective, each online round corresponds to the scenario when a new set of forecasts is generated, as shown in Fig. 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. 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. 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 $\mathcal{D}_{n-1}^{(n)} = (\mathbf{x}^{(n-1)}, \mathbf{y}_{n-1}^{(n)})$. 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 $\mathcal{D}^{(n-2)}$, $\mathcal{D}^{(n-3)}$, $\cdots$ $\mathcal{D}^{(n-T+1)}$ 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 $\mathcal{D}^{(n-T)}$ to obtain a new model $f^{(n-T+1)}$. We then update $f^{(n-T+1)}$ with the expanded labeled data $\mathcal{D}^{(n-T+1)}$ 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 $\mathcal{D}^{(n)}$ 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.

4 Online Regularized multiI-task regressiON (ORION)

This section presents the ORION framework for the ϵ-insensitive loss function. An extension of the framework to the quantile loss function is given in Section 5.

4.1 ORION for ϵ-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 [32] to a multi-task learning setting with an ϵ-insensitive loss function:

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

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 (1) can be simplified as follows:

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

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, $\mathrm{Tr}[\cdot]$ 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

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

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

4.2 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:

$$\begin{aligned} \mathcal{L}(\mathbf{z}, \boldsymbol{\tau}) &= \frac{1}{2}(\mathbf{z} - \mathbf{z}^{(n-1)})^T\mathbf{R}(\mathbf{z} - \mathbf{z}^{(n-1)}) + \frac{1}{2}\mathbf{z}^T\mathbf{Q}\mathbf{z} \\ &+ \left[(\mathbf{z}^T\tilde{\mathbf{X}}^{(n)} - \mathbf{y}^{(n)^T})\mathbf{S} - \epsilon\mathbf{1}^T\right]\mathbf{P}\boldsymbol{\tau} \end{aligned} \tag{3}$$

where $\tilde{\mathbf{X}}^{(n)}$, $\mathbf{R}$, $\mathbf{Q}$, $\mathbf{P}$, and $\mathbf{S}$ are defined in Table 1.

TABLE 1: Notations used in Equation (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 $\mathbf{B}$
$\tilde{\mathbf{X}}^{(n)}$	$\begin{bmatrix} \mathbf{x}_1^{(n)} & \mathbf{x}_2^{(n)} & \cdots & \mathbf{x}_T^{(n)} \\ \mathbf{x}_1^{(n)} & \mathbf{0}_d & \cdots & \mathbf{0}_d \\ \mathbf{0}_d & \mathbf{x}_2^{(n)} & \cdots & \mathbf{0}_d \\ \vdots & \vdots & \vdots & \vdots \\ \mathbf{0}_d & \mathbf{0}_d & \cdots & \mathbf{x}_T^{(n)} \end{bmatrix}$
$\mathbf{y}^{(n)}$	$[y_1^{(n)}; y_2^{(n)}; \cdots; y_{m_n}^{(n)}; \mathbf{0}_{(T-m_n)}]$
$\mathbf{P}$	$\mathbf{P}_{i,j} = \begin{cases} 1 & \text{if } i = j \text{ and } i \in \mathcal{O}^{(n)} \\ 0 & \text{otherwise} \end{cases}$
$\mathbf{S}$	$\mathbf{S}_{i,j} = \begin{cases} \mathrm{sign}(\mathbf{w}_i^{(n)^T}\mathbf{x}_i^{(n)} - y_i^{(n)}) & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$
$\boldsymbol{\tau}$	$[\tau_1; ...; \tau_T]$
$\mathbf{R}$	$\begin{bmatrix} \lambda\mathbf{I}_d & \mathbf{0}_{d\times Td} \\ \mathbf{0}_{Td\times d} & \beta\mathbf{I}_{Td} \end{bmatrix}$
$\mathbf{Q}$	$\begin{bmatrix} \mathbf{0}_{d\times d} & \mathbf{0}_{d\times Td} \\ \mathbf{0}_{Td\times d} & (\mathbf{L} + \mu\mathbf{I}_T) \otimes \mathbf{I}_d \end{bmatrix}$

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

$$\begin{aligned} \frac{\partial\mathcal{L}(\mathbf{z}, \boldsymbol{\tau})}{\partial\mathbf{z}} &= \mathbf{R}(\mathbf{z} - \mathbf{z}^{(n-1)}) + \mathbf{Q}\mathbf{z} + \tilde{\mathbf{X}}^{(n)}\mathbf{S}\mathbf{P}\boldsymbol{\tau} = 0 \\ \mathbf{z} &= \mathbf{M}(\mathbf{R}\mathbf{z}^{(n-1)} - \tilde{\mathbf{X}}^{(n)}\mathbf{S}\mathbf{P}\boldsymbol{\tau}) \end{aligned} \tag{4}$$

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 (4) back into Equation (3) leads to the following equation after simplification

$$\begin{aligned} \mathcal{L}(\boldsymbol{\tau})\tilde{\mathbf{X}}^{(n)}\mathbf{S}\mathbf{P}\boldsymbol{\tau} &= -\frac{1}{2}\boldsymbol{\tau}^T\tilde{\mathbf{X}}_{PS}^{(n)T}\mathbf{M}^T\tilde{\mathbf{X}}_{PS}^{(n)}\boldsymbol{\tau} + \ell_n^T(\hat{\mathbf{z}}^{(n-1)})\boldsymbol{\tau} \\ &+ \text{constant} \end{aligned} \tag{5}$$

where

$$\begin{aligned} \tilde{\mathbf{X}}_{PS}^{(n)} &= \tilde{\mathbf{X}}^{(n)}\mathbf{S}\mathbf{P} \\ \ell_n^T(\hat{\mathbf{z}}^{(n-1)}) &= \left[(\hat{\mathbf{z}}^{(n-1)T}\tilde{\mathbf{X}}^{(n)} - \mathbf{y}^{(n)T})\mathbf{S} - \epsilon\mathbf{1}^T\right]\mathbf{P} \\ \hat{\mathbf{z}}^{(n-1)} &= \mathbf{M}^T\mathbf{R}^T\mathbf{z}^{(n-1)} \end{aligned} \tag{6}$$

Note that $\mathbf{P}$ is a diagonal matrix, whose diagonal element $\mathbf{P}_{t,t}$ is zero if $t \notin \mathcal{O}^{(n)}$. In other words, if the target value for task t is either unavailable or predicted correctly (within the ϵ-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, τ_t for $t \notin \mathcal{O}^{(n)}$ has no impact on Equation (5) and can be set to zero. In the following derivation, we assume the rows and columns corresponding to all the tasks $t \notin \mathcal{O}^{(n)}$ in τ, $\tilde{\mathbf{X}}_{PS}^{(n)}$, 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

$$\begin{aligned} \frac{\partial\mathcal{L}}{\partial\boldsymbol{\tau}} &= -\tilde{\mathbf{X}}_{PS}^{(n)T}\mathbf{M}^T\tilde{\mathbf{X}}_{PS}^{(n)}\boldsymbol{\tau} + \ell_n(\hat{\mathbf{z}}^{(n-1)}) = 0 \\ \boldsymbol{\tau} &= \left[\tilde{\mathbf{X}}_{PS}^{(n)T}\mathbf{M}^T\tilde{\mathbf{X}}_{PS}^{(n)}\right]^{-1}\ell_n(\hat{\mathbf{z}}^{(n-1)}) \end{aligned} \tag{7}$$

There are several points worth noting regarding the update formula for $\mathbf{z}$ and its learning rate $\boldsymbol{\tau}$. First, note that Equation (7) is only applicable to tasks that belong to $\mathcal{O}^{(n)}$. The columns in $\tilde{\mathbf{X}}_{PS}$ for $t \notin \mathcal{O}^{(n)}$ must be removed before calculating τ. Otherwise, the matrix $\tilde{\mathbf{X}}_{PS}^{(n)T}\mathbf{M}^T\tilde{\mathbf{X}}_{PS}^{(n)}$ is not invertible. For $t \notin \mathcal{O}^{(n)}$, 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 (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 [32], $\tau_n = \ell_n/||\mathbf{x}_n||^2$. The main difference is that the τ for multi-task learning must take into account the task relatedness in both ℓ_n and the inverse of $\tilde{\mathbf{X}}_{PS}^{(n)T}\mathbf{M}^T\tilde{\mathbf{X}}_{PS}^{(n)}$.

4.3 Algorithm

A summary of the ORION framework for ϵ-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 ϵ-bound. The weights associated with the tasks will be updated using the formula given in

Equation (4). Note that τ_t is set to zero for tasks that do not belong to $\mathcal{O}^{(n)}$. 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. Theoretical analysis of the algorithm is given in the Appendix section.

4.3.1 Time complexity

In this section, we analyze the time complexity of the ORION framework for the ϵ-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. 1). Each round requires calculations of Equations (4) and (7). For Equation (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 $O(T^3d^2)$ flops and its inverse will take $O(T^3)$ flops. According to Equation (6), calculating both $\hat{\mathbf{z}}^{(n-1)}$ and $\ell_n(\hat{\mathbf{z}}^{(n-1)})$ will require $O(T^3d)$ flops. The time complexity for calculating Equation (7) is $O(T^3d^2)$ whereas the time complexity for Equation (4) is $O(T^3d + T^2d^2)$. Therefore, the model update for each task requires $O(T^3d^2)$ flops. Since there are T tasks, the time complexity for each online round is $O(T^4d^2)$. 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 $O(T^3d^3)$. Thus, after N rounds, the overall time complexity is $O(N(T^4d^2 + T^3d^3))$, 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.

```
Input: $\mu$, $\lambda$, $\beta$, $\epsilon = 0.001$ ;
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 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 $\mathcal{O}^n = \{t | t \le m_n; |\mathbf{w}_t^{(m)T}\mathbf{x}_t^{(m)} - y_{m,t}^{(n)}| > \epsilon\}$ ;
        Compute $\boldsymbol{\tau}$ using Equation (7) and set $\tau_t = 0$
        when $t \notin \mathcal{O}^{(n)}$ ;
        Update $\mathbf{z}^{(m)}$ using Equation (4) ;
    end
end
```
Algorithm 1: Pseudocode for ORION-ϵ Algorithm

4.4 Theoretical Analysis of ORION-ϵ

This section presents theoretical analysis on the average loss bound of the ORION-ϵ algorithm.

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

Proof. First, we can write

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

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

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

□

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

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

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 $(\mathbf{MR})^T\mathbf{MR} = \mathbf{U}(\mathbf{I}+\mathbf{\Lambda})^{-2}\mathbf{U}$, the largest eigenvalue of $(\mathbf{MR})^T\mathbf{MR}$ must also be equal to 1. Thus, $\|\mathbf{MR}\|_2 = 1$. □

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

Proof. Following Lemma 1, it is easy to see that $\mathbf{I}-\mathbf{MR} = \mathbf{U}[\mathbf{I}-(\mathbf{I}+\mathbf{\Lambda})^{-1}]\mathbf{U}^T$. 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-ϵ algorithm. Using the notations given in Table 1, the following bound holds for any $\mathbf{u} \in \Re^{(T+1)d}$.*

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

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

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,

$$\begin{aligned} \sum_{n=1}^N \Delta_n &= \sum_{n=1}^N \|\mathbf{z}^{(n)} - \mathbf{u}\|^2 - \|\mathbf{z}^{(n+1)} - \mathbf{u}\|^2 \\ &= \|\mathbf{z}^{(1)} - \mathbf{u}\|^2 - \|\mathbf{z}^{(N+1)} - \mathbf{u}\|^2 \qquad (8) \\ &= \|\mathbf{u}\|^2 - \|\mathbf{z}^{(N+1)} - \mathbf{u}\|^2 \\ &\le \|\mathbf{u}\|^2 \end{aligned}$$

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

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

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

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

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

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

Putting them together, we have

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

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

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

□

5 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 [41] 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 τ^{th} conditional quantile of the prediction by minimizing the following sum of asymmetrically weighted absolute residuals:

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

The τ^{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 τ, 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 τ 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.

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

Compared to ORION-ϵ, 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 [42] to estimate the parameters using $\tau = 0.95$ to detect extreme (high) value events.

6 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.

6.1 Performance Comparison for ORION-ϵ

We compared the performance of ORION-ϵ against the following baseline methods.

1) **Ensemble Median (EM)**: This is an unbiased aggregation of the ensemble member forecasts.
2) **Passive-Aggresive (PA) Algorithm** [32]: 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 ℓ is the ϵ-insensitive loss.
3) **Tracking Climate Models (TCM)** [43]: This method uses a Hidden Markov Model to track the current best ensemble member. Unlike ORION-ϵ, 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 ϵ-insensitive loss function for our regression problem and ignore the use of slack variables. The modified objective function is given by:
$$\arg\min_{\mathbf{w}_t} \frac{1}{2}\sum_{t=1}^{T}\|\mathbf{w}_t - \mathbf{w}_t^{(n-1)}\|_2^2$$
$$\text{s.t.} \quad \forall t \le m_n,\ |\mathbf{w}_t^T\mathbf{x}_t^{(n)} - y_t^{(n)}| \le \epsilon$$
The optimization problem can be solved using a similar technique as that for ORION-ϵ.
5) **Linear Algorithms for Online Multi-task Learning(LAOM)** [35]: 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 [35] 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-ϵ does not require such an assumption.

TABLE 2: Comparison of mean absolute error (MAE) for ORION-ϵ against baseline methods on soil moisture data

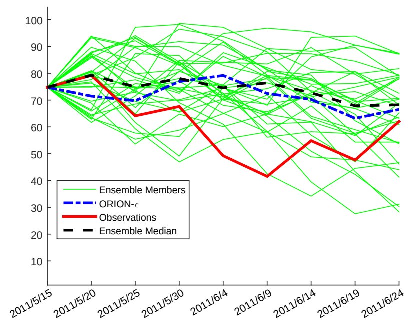
	ORION-ϵ	EM	PA	TCM	OMTLSL	LAOM
arkansusred	**2.740**	4.189	3.788	3.659	5.423	2.767
calinevada	**3.398**	4.919	4.281	4.265	4.422	4.257
colorado	**4.362**	5.934	5.741	5.634	6.068	5.674
columbia	**4.411**	6.000	6.439	6.475	6.225	5.370
lowermiss	**9.891**	12.023	11.639	10.671	14.975	11.951
midatlantic	**13.473**	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

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

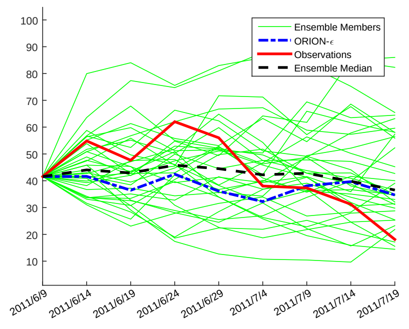
Table 2 compares the results of the different methods. As can be seen from the table, ORION-ϵ works better than the baseline methods on all 12 data sets. In particular, it outperforms OMTLSL, which is a state-of-the-art online multi-task learning algorithm, on all the data sets. Unlike OMTLSL, ORION-ϵ 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 5, the improvement is still observed even when the task relationship is removed (i.e., comparing OMTLSL against ORION-ϵ-NR). Comparing ORION-ϵ 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 shows the predicted time series for ORION-ϵ 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-ϵ is similar to EM (see Fig. 3a to 3c). As more data becomes available, the predictions by ORION-ϵ is closer to observation data than EM (Fig. 3d to 3j). In Fig. 3k, there appears to be a sudden shift that causes the performance of ORION-ϵ to degrade significantly. However, after one update, ORION-ϵ recovers from the mistake and its prediction follows closely the observation data again (Fig. 3l). Fig. 4 shows the absolute error of ORION-ϵ 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-ϵ is slightly worse than EM at the beginning, after sufficient training, ORION-ϵ appears to perform significantly better than EM.

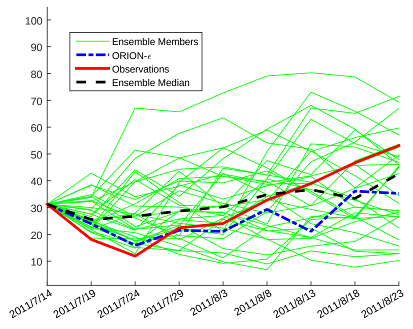
We also compared the runtime of ORION against other baseline methods. The total runtime as well as average runtime per round for the Northeast data set is shown in Table 3. ORION is relatively slower than other baselines,



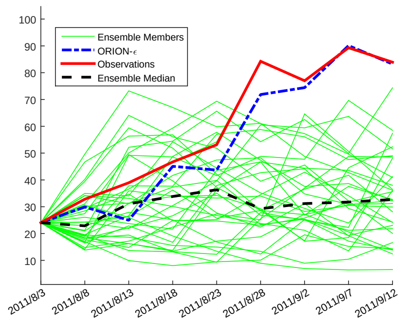
(a) Forecasts for 05/15/2011



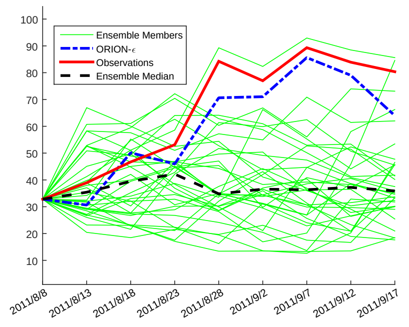
(b) Forecasts for 06/09/2011



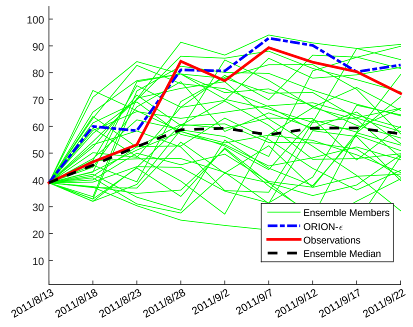
(c) Forecasts for 07/14/2011



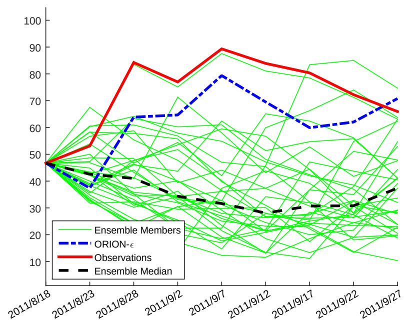
(d) Forecasts for 08/03/2011



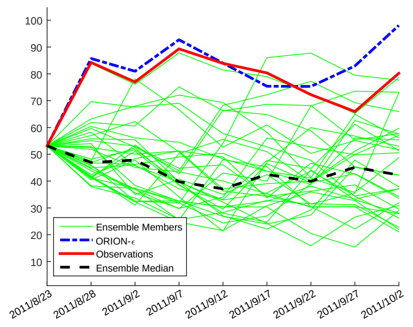
(e) Forecasts for 08/08/2011



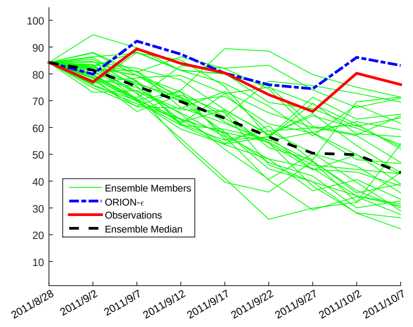
(f) Forecasts for 08/13/2011



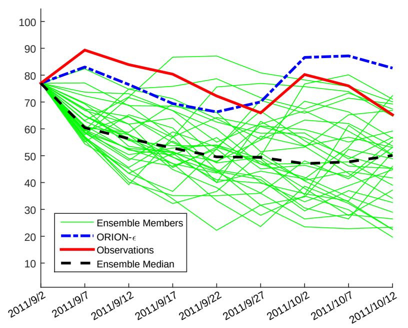
(g) Forecasts for 08/18/2011



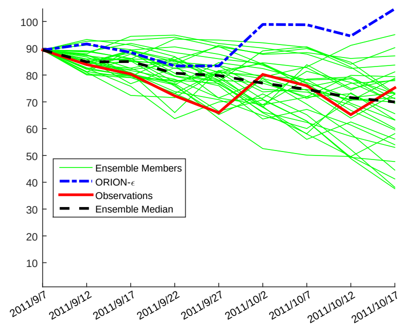
(h) Forecasts for 08/23/2011



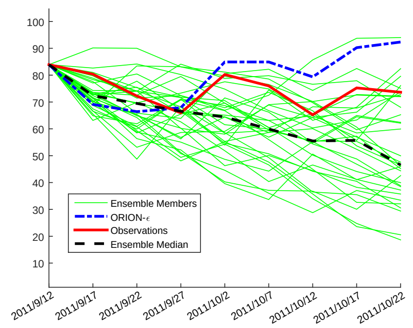
(i) Forecasts for 08/28/2011



(j) Forecasts for 09/02/2011

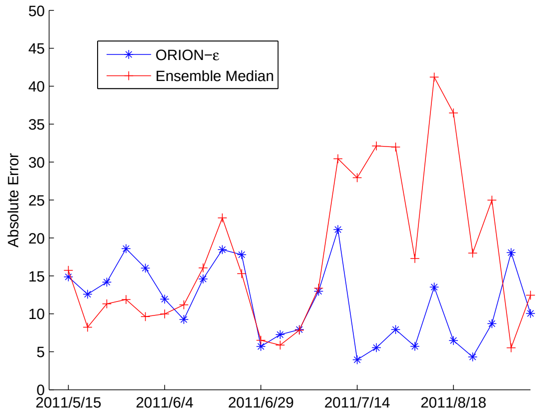


(k) Forecasts for 09/07/2011

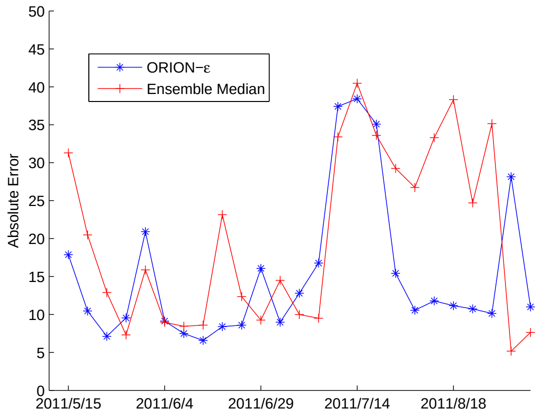


(l) Forecasts for 09/12/2011

Fig. 3: Forecasts on Dataset Northeast for ORION-ϵ. Fig. 3a - 3c are results from the training set and Fig. 3d - 3l are results from the test set. Note that in the early stage of the online learning process, ORION-ϵ performs similar to Ensemble Median (see Fig. 3a - 3b), and ORION-ϵ starts to follow the observation from Fig. 3c.



(a) Northeast Data



(b) Mid-Atlantic Data

Fig. 4: Mean absolute error for ORION-ϵ and Ensemble Median on the Northeast and Midatlantic data.

TABLE 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

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 2.

6.2 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-ϵ, and the following two baselines:

TABLE 4: Comparison of F1 measure for predicting occurrence of extreme events.

	ORION-QR	EM	ORION-ϵ	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**

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 [32] algorithm using a quantile loss function. The objective function is modified as follows:

$$\begin{aligned}\min_{\mathbf{p},\mathbf{q},\mathbf{w}} \quad & \tau\mathbf{1}_T^T\mathbf{p} + (1-\tau)\mathbf{1}_T^T\mathbf{q} \\ & + \frac{\lambda}{2}\|\mathbf{w}-\mathbf{w}^{(n-1)}\|_2^2 \\ \text{s.t.} \quad & \forall t, \mathbf{y}_t^{(n)} - \mathbf{w}^T\mathbf{x}_t^{(n)} = p_t - q_t \\ & \mathbf{p} \geq \mathbf{0}, \mathbf{q} \geq \mathbf{0}\end{aligned} \tag{9}$$

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 4 shows that ORION-QR outperforms both EM and ORION-ϵ in 4 out of 5 data sets. The latter suggests quantile loss is more effective than ϵ-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.

6.3 Sensitivity Analysis

This section analyzes the sensitivity of the three input parameters of the ORION framework.[1] 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 μ controls sparsity of the weights for $\mathbf{v}_t$. Note that μ must be non-negative to ensure the matrix $\mathbf{L} + \mu\mathbf{I}_T$ is positive semi-definite. Fig. 5(a) shows that MAE improves as μ increases. As long as μ 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 β and λ 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 β (see Fig. 5(a)). The results are more sensitive to choice of λ. If λ is too small, $\mathbf{w}_0$ deviates significantly from its previous value. Conversely, if λ 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, λ can be set based on its performance on the training data.

6.4 Variations of ORION-ϵ Framework

The ORION framework uses two types of information to update its model parameters. First, it uses the λ and β regularizers to control the amount of information retained from previous rounds. 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-ϵ 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.[2] We denote the approach as ORION-ϵ-β. Experimental results given in Table 5 showed that ORION-ϵ outperforms ORION-ϵ-β 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-ϵ-NR. Based on the results given in Table 5, ORION-ϵ outperforms ORION-ϵ-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-ϵ framework.

7 Conclusion

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

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

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

TABLE 5: Comparison of mean absolute error (MAE) for different variations of ORION-ϵ framework

	ORION-ϵ	ORION-ϵ-NR	ORION-ϵ-β
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**

Acknowledgments

The research is partially supported by NOAA Climate Program office through grant NA12OAR4310081, NASA Terrestrial Hydrology Program through grant NNX13AI44G, and Office of Naval Research through grant N00014-14-1-0631.

References

[1] C. Tebaldi and R. 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*, vol. 365, no. 1857, pp. 2053–2075, 2007.

[2] P. Cantelaube and J.-M. Terres, "Seasonal weather forecasts for crop yield modeling in europe," *Tellus A*, vol. 57, no. 3, pp. 476–487, 2005.

[3] M. B. Araujo and M. New, "Ensemble forecasting of species distributions," *Trends in Ecology & Evolution*, vol. 22, no. 1, pp. 42–47, 2007.

[4] L. Luo and E. F. Wood, "Use of bayesian merging techniques in a multimodel seasonal hydrologic ensemble prediction system for the eastern united states," *Journal of Hydrometeorology*, vol. 9, pp. 866–884, 2008.

[5] M. Leutbecher and T. Palmer, "Ensemble forecasting," *Journal of Computational Physics*, vol. 227, pp. 3515–3539, 2008.

[6] X.-T. Yuan and S. Yan, "Visual classification with multi-task joint sparse representation," in *CVPR 2010*, pp. 3493–3500.

[7] X. Wang, C. Zhang, and Z. Zhang, "Boosted multi-task learning for face verification with applications to web image and video search," in *CVPR 2009*, pp. 142–149.

[8] J. Zhou, L. Yuan, J. Liu, and J. Ye, "A multi-task learning formulation for predicting disease progression," in *KDD 2011*, pp. 814–822.

[9] S. Bickel, J. Bogojeska, T. Lengauer, and T. Scheffer, "Multi-task learning for hiv therapy screening," in *ICML 2008*, pp. 56–63.

[10] X. Ning and G. Karypis, "Multi-task learning for recommender system." in *ACML 2010*, pp. 269–284.

[11] J. Wang, S. C. Hoi, P. Zhao, and Z.-Y. Liu, "Online multi-task collaborative filtering for on-the-fly recommender systems," in *RecSys 2013*, pp. 237–244.

[12] R. K. Ando and T. Zhang, "A framework for learning predictive structures from multiple tasks and unlabeled data," *JMLR*, vol. 6, pp. 1817–1853, Dec. 2005.

[13] G. Obozinski, B. Taskar, and M. Jordan, "Joint covariate selection and joint subspace selection for multiple classification problems," *Statistics and Computing*, vol. 20, no. 2, pp. 231–252, 2010.

[14] J. Xu, P.-N. Tan, and L. Luo, "ORION: Online Regularized multI-task regressiON and its application to ensemble forecasting," in *ICDM 2014*, pp. 1061–1066.

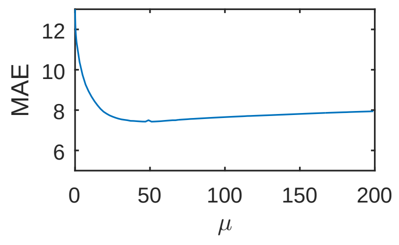
[15] H. Cheng, P.-N. Tan, G. Jing, and J. Scripps, "Multi-step ahead time series prediction," in *PAKDD 2006*, pp. 765–774.

[16] Y. Bao, T. Xiong, and Z. Hu, "Multi-step-ahead time series prediction using multiple-output support vector regression," *Neurocomputing*, vol. 129, pp. 482 – 493, 2014.

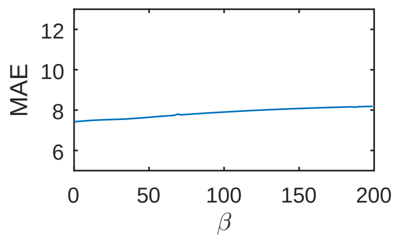
[17] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, no. 1, pp. 41–75, Jul. 1997.

[18] J. Chen, J. Zhou, and J. Ye, "Integrating low-rank and group-sparse structures for robust multi-task learning," in *KDD 2011*, pp. 42–50.

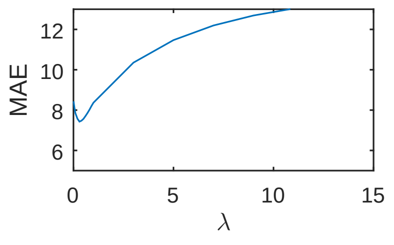
[19] J. Chen, J. Liu, and J. Ye, "Learning incoherent sparse and low-rank patterns from multiple tasks," *TKDD*, vol. 5, no. 4, pp. 22:1–22:31, 2012.



(a) Effect of varying μ on MAE



(b) Effect of varying β on MAE



(c) Effect of varying λ on MAE

Fig. 5: Sensitivity Analysis of ORION-ϵ. Fig. 5a shows that ORION-ϵ tends to choose a large value of μ; Fig. 5b shows that ORION-ϵ is not that sensitive to β; Fig. 5c shows that λ is the parameter to be tuned in practice.

[20] P. Gong, J. Ye, and C. Zhang, "Robust multi-task feature learning," in *KDD 2012*, pp. 895–903.

[21] A. Argyriou, T. Evgeniou, and M. Pontil, "Convex multi-task feature learning," *Machine Learning*, vol. 73, no. 3, pp. 243–272, Dec. 2008.

[22] Z. Kang, K. Grauman, and F. Sha, "Learning with whom to share in multi-task feature learning." in *ICML 2011*, pp. 521–528.

[23] A. Kumar and H. Daumé III, "Learning task grouping and overlap in multi-task learning." in *ICML 2012*.

[24] J. Chen, L. Tang, J. Liu, and J. Ye, "A convex formulation for learning a shared predictive structure from multiple tasks," *PAMI*, vol. 35, no. 5, pp. 1025–1038, May 2013.

[25] Y. Zhang and D.-Y. Yeung, "A convex formulation for learning task relationships in multi-task learning," in *UAI 2010*, pp. 733–442.

[26] N. D. Lawrence and J. C. Platt, "Learning to learn with the informative vector machine," in *ICML 2004*, pp. 65–72.

[27] K. Yu, V. Tresp, and A. Schwaighofer, "Learning gaussian processes from multiple tasks," in *ICML 2005*, pp. 1012–1019.

[28] S.-I. Lee, V. Chatalbashev, D. Vickrey, and D. Koller, "Learning a meta-level prior for feature relevance from multiple related tasks," in *ICML 2007*, pp. 489–496.

[29] H. Daumé III, "Bayesian multitask learning with latent hierarchies," in *UAI 2009*, pp. 135–142.

[30] J. Xu, J. Zhou, and P.-N. Tan, "Formula: Factorized multi-task learning for task discovery in personalized medical models," in *SDM 2015*, pp. 496–504.

[31] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," *Information and Computation*, vol. 108, no. 2, pp. 212–261, Feb. 1994.

[32] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *JMLR*, vol. 7, pp. 551–585, Dec. 2006.

[33] M. Dredze, K. Crammer, and F. Pereira, "Confidence-weighted linear classification," in *ICML 2008*, pp. 264–271.

[34] O. Dekel, P. M. Long, and Y. Singer, "Online learning of multiple tasks with a shared loss," *JMLR*, vol. 8, pp. 2233–2264, Dec. 2007.

[35] G. Cavallanti, N. Cesa-Bianchi, and C. Gentile, "Linear algorithms for online multitask classification," *JMLR*, vol. 11, pp. 2901–2934, Dec. 2010.

[36] A. Saha, P. Rai, H. D. III, and S. Venkatasubramanian, "Online learning of multiple tasks and their relationships." in *AISTATS 2011*, pp. 643–651.

[37] G. Li, S. Hoi, K. Chang, W. Liu, and R. Jain, "Collaborative online multitask learning," *TKDE 2014*, vol. 26, pp. 1866–1876, Aug 2014.

[38] T. Evgeniou and M. Pontil, "Regularized multi-task learning," in *KDD 2004*, pp. 109–117.

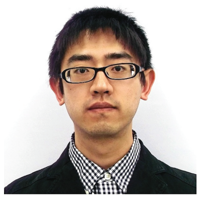
[39] X. Sun, H. Kashima, and N. Ueda, "Large-scale personalized human activity recognition using online multitask learning," *TKDE*, vol. 25, pp. 2551–2563, 2013.

[40] A. Agarwal, A. Rakhlin, and P. Bartlett, "Matrix regularization techniques for online multitask learning," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2008-138, Oct 2008. [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-138.html

[41] R. Koenker, *Quantile Regression*, ser. Econometric Society Monographs. Cambridge University Press, 2005.

[42] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.0 beta," http://cvxr.com/cvx, Sep. 2013.

[43] C. Monteleoni, G. A. Schmidt, and S. Saroha, "Tracking climate models," in *NASA Conference on Intelligent Data Understanding 2010*, pp. 1–15.



Jianpeng Xu is a Ph.D student in the Department of Computer Science and Engineering at Michigan State University. He received his MS in Computer Science from Harbin Institute of Technology in 2010 and BS in Computer Science from Shandong University in 2007. His research focuses on geospatio-temporal data mining, recommendation system, and personalized modeling. He has published in top data mining conferences such as KDD, ICDM, and SDM. One of his papers received the best research paper award at IEEE BigData 2016.

Pang-Ning Tan is an Associate Professor in the Department of Computer Science and Engineering at MSU. He received his M.S degree in Physics and Ph.D. degree in Computer Science from University of Minnesota. His research interests span a broad range of data mining problems, from pattern discovery to predictive modeling. He has published more than 100 peer-reviewed papers in journals and conference proceedings.

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

Lifeng Luo is an Associate Professor in the Department of Geography at Michigan State University. He received his B.Sc. from Peking University in 1998 and Ph.D. from Rutgers University in 2003. He was a research scientist at Princeton University before joining MSU in August, 2009. He is also affiliated with the Environmental Sciences and Policy Program, Center for Water Sciences, and Center for Global Change and Earth Observations. His research covers a range of topics related to land-atmosphere interaction and its impact on the global climate and hydrological cycle at various spatial and temporal scales. His recent research focuses on the predictability and prediction of climate extremes such as drought, floods, heat waves at subseasonal to seasonal scale.