Distribution Preserving Multi-Task Regression for Spatio-Temporal Data

Submitted for Blind Review

***Abstract*—For many spatio-temporal applications, building regression models that can reproduce the true data distribution is often as important as building models with high prediction accuracy. For example, knowing the future distribution of daily temperature and precipitation can help scientists determine their long-term trends and assess their potential impact on human and natural systems. As conventional methods are designed to minimize residual errors, the shape of their predicted distribution may not be consistent with their actual distribution. To overcome this challenge, this paper presents a novel, distribution-preserving multi-task learning framework for multi-location prediction of spatio-temporal data. The framework employs a non-parametric density estimation approach with L2-distance to measure the divergence between the predicted and true distribution of the data. Experimental results using climate data from more than 1500 weather stations in the United States show that the proposed framework can reduce the distribution error by more than 15% compared to non-distribution preserving approaches without degrading the prediction accuracy significantly.**

***Index Terms*—Multi-task learning, spatio-temporal data, regression**

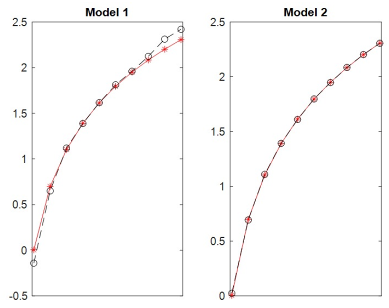
I. Introduction

Regression methods play an important role in many spatio-temporal applications as they can be used to solve a wide variety of prediction problems such as projecting future changes in the climate system, predicting the crime rate in urban cities, or forecasting traffic volume on highways. Although accuracy is an important requirement, building models that can replicate the future distribution of data is just as important since the predicted distribution can be used for planning, risk assessment, and other decision making purposes. For example, in climate modeling, knowing the changes in future distribution of climate variables such as temperature and precipitation can help scientists to better estimate the severity and frequency of adverse weather events in the future. In agricultural production, the predicted distribution can be used to derive statistics such as average length of future growing season or persistency of wet and dry spells, which are important metrics for farmers and agricultural researchers.

However, previous studies have shown that the distribution of predicted values generated by traditional regression methods are not always consistent with the true distribution of the data even when the prediction errors are relatively low [1], [2]. While distribution-preserving methods such as quantile mapping [3] have been developed to overcome this limitation, their prediction errors can still be high [1]. For example, Figure 1(a) shows a comparison between the predicted values of a non-distribution preserving model (Model 1) against a distribution-preserving model (Model 2) on a set of 10 values. Although the first model has a much lower root mean square error (RMSE) it does not fit well the tails of the predicted distribution, as shown in Figure 1, compared to the second model, which fits the distribution almost perfectly but has a considerably higher RMSE. This has led to the growing interest in developing techniques that can minimize both prediction error and the divergence between the true and predicted distributions [1], [2]. However, current techniques are mostly designed for single task learning problems, i.e., to build a regression model for a single location. For multi-location prediction, these models are trained independently, and thus, often fail to capture the inherent autocorrelations of the spatio-temporal data. In addition, their accuracy and distribution fit are likely to be suboptimal for locations with limited training data.

True Value	Model 1	Model 2
0.0043	-0.1411	0.6924
0.6974	0.6501	0.0241
1.1029	1.1194	1.1095
1.3906	1.3881	1.6097
1.6137	1.6157	1.3931
1.7960	1.8120	1.7977
1.9502	1.9603	1.9473
2.0837	2.1237	2.2026
2.2015	2.3125	2.0850
2.3069	2.4202	2.3059
RMSE	0.0713	0.3243

(a) Predicted values



(b) Cumulative distribution function of predicted values

Figure 1: Comparison between the predictions of non-distribution preserving (Model 1) and distribution preserving (Model 2) methods in terms of their root mean squared errors and cumulative distribution functions.

To account for spatial autocorrelation and the imbalanced distribution of training data, there have been several recent studies focusing on the development of multi-task learning (MTL) methods [4] [5] for spatio-temporal data [6] [7]. MTL learns a local model for each location, but leverages data from other locations to improve its model performance. It accomplishes this by assuming that the local models share some common structure, which can be exploited to enhance their predictive performance. Unfortunately, existing MTL approaches are mostly focused on minimizing the residual error, paying scant attention to how realistic is the overall predicted distribution. As an example, consider the histogram of monthly precipitation shown in Fig. 2 for a weather station located in Bridgeport, Nebraska. The dashed blue line shows the marginal distribution for precipitation predicted by

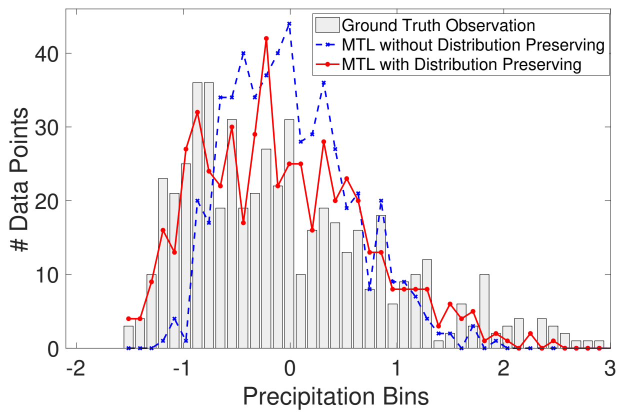


Figure 2: Histograms of observed precipitation against estimated precipitation produced by both distribution preserving and non-distribution preserving multi-task regression methods.

GSpartan [7], a spatio-temporal multi-task learning approach. Observe that GSpartan does not fully capture the shape of the true distribution, particularly the dual peaks and the tails of the curve. In contrast, the distribution-preserving multi-task learning approach proposed in this paper does a better job at representing the true distribution compared to GSpartan. Although its RMSE increases slightly by 3.6%, the divergence in its distribution improves significantly by more than 30%.

In this paper, we present a novel distribution-preserving multi-task learning framework for spatio-temporal data. Our framework assumes that the local models share a common low-rank representation, similar to the assumption used in [7]. It also employs a graph Laplacian regularizer based on the Haversine spatial distance to preserve the spatial autocorrelation in the data. A non-parametric kernel density estimation approach with L2-distance is used to determine the divergence between the predicted and true distributions of the data. Both the distribution fitting and multi-task learning are integrated into a unified objective function, which is optimized using a mini-batch accelerated gradient descent algorithm. Experimental results using a real-world climate dataset from the Global Historical Climatology Network (GHCN) showed that the proposed distribution preserving multi-task regression method reduces the distribution error for more than 95% of the weather stations considered in this study, with an average gain of 20%. Our approach also outperforms a distribution-preserving single-task regression method called contour regression [1] in more than 77% of the weather stations.

II. Preliminaries

Our framework is designed not only to learn accurate local regression models, but also to generate predictions that are consistent with the true distribution of the data. This requires an approach for estimating the density function of the response variable and a divergence measure to compute the difference between two distributions. We review these approaches in this section.

A. Density Estimation

There are various density estimation methods that have been proposed in the literature. In general, these methods can be divided into two categories:

- **Parametric Methods**, which assume that the density function follows certain parametric distribution, such as Gaussian, Gamma, exponential and so on. The sampled data points are used to estimate parameters of the distribution. As the number of parameters tends to be small, parametric methods has an advantage in that they do not require a large number of points to fit the density function. Unfortunately, many real-world datasets may not follow the standard distributions as they often comprise of complex mixtures of distributions, which leads to imprecise estimation of their true distribution.
- **Non-parametric Methods**, which avoid making *a priori* assumption about the shape of the distributions. Two popular methods are the K-nearest neighbor (KNN) approach and kernel density estimation. The KNN approach uses only the k nearest neighbors to estimate the density function whereas the kernel density estimation (KDE) approach uses a mixture of Gaussian distributions centered at all the data points with a smoothing kernel width parameter, h.

Due to the generality of the approach for fitting unknown distribution, we employ the KDE approach to approximate the density function. Given N data points sampled from an unknown distribution of a variable Y , $\mathbf{y} = \{y_i|, i = 1, 2, ..., N\}$, the density function of Y is estimated as follows:

$$P_{\mathbf{y}}(Y = y) = \frac{1}{N}\sum_{i}^{N} G(y|y_i, h^2) \tag{1}$$

B. Divergence Measures

A divergence measure can be used to estimate the difference between two density functions. This section reviews some of the divergence measures used in this paper. Although there are other measures available, evaluating them is beyond the scope of this paper and is a subject for future research.

1) RMS-CDF: RMS-CDF is a measure defined in [1] to compare two empirical cumulative distribution functions. Let $\mathbf{y}$ and $\hat{\mathbf{y}}$ denote vectors of length N sampled from two distributions. The RMS-CDF measure between the pair of distributions is computed as follows:

$$\text{RMS-CDF} = \sqrt{\frac{1}{N}\sum_{i}^{N}(y_{(i)} - \hat{y}_{(i)})^2} \tag{2}$$

where $y_{(i)}$ represents the i-th largest value in $\mathbf{y}$ and $\hat{y}_{(i)}$ is the i-th largest value in $\hat{\mathbf{y}}$. The measure is obtained by sorting the values in $\mathbf{y}$ and $\hat{\mathbf{y}}$ and computing the average sum of squared difference between their sorted values.

Table I: Summary of notations used in the paper.

Notation	Definition
S	number of stations(tasks)
N_s	number of observations in task s
d	number of features
k	number of latent factors, $k < d$
$\mathbf{X}_s \in \Re^{N_s \times d}$	predictor matrix for station s
$\mathbf{y}_s$ or $\hat{\mathbf{y}}_s \in \Re^{N_s \times 1}$	observed or estimated response samples at station s
$P_{\mathbf{y}}(Y)$ or $P_{\hat{\mathbf{y}}}(Y)$	density function of Y estimated by observed(or estimated) samples
$\mathbf{W} \in \Re^{d \times S}$	model parameter
$\mathbf{U} \in \Re^{d \times k}$	latent factors
$\mathbf{V} \in \Re^{k \times S}$	linear coefficients for latent factors
h, $\hat{h}$	Parzen window widths for $\mathbf{y}$ and $\hat{\mathbf{y}}$
$G(y\|\mu, \sigma)$	Gaussian distribution
$\mathbf{Q} \in \Re^{S \times S}$	pairwise Haversine distances
$\mathbf{A} \in \Re^{S \times S}$	adjacency matrix, where $A_{ij} = \frac{1}{\exp(Q_{ij}/\gamma)}$
$\mathbf{D} \in \Re^{S \times S}$	a diagonal matrix, $D_{ii} = \sum_j A_{ij}$

2) L2 Distance: Given a pair of random variables, Y and $\hat{Y}$, along with their respective probability density functions, P_Y and $P_{\hat{Y}}$, their L^2-Distance can be calculated as follows [8]:

$$L^2(P_Y, P_{\hat{Y}}) = \int (P_Y(y) - P_{\hat{Y}}(y))^2 dy$$

Unlike RMS-CDF, L^2-Distance measures the divergence of two distribution based on probability density functions instead of their cumulative distribution functions.

III. Proposed Framework

This section introduces our proposed framework for distribution-preserving multi-task regression. The framework uses kernel density estimation (KDE) to estimate the probability density function. KDE provides a flexible approach for modeling the density function of unknown distributions unlike other parametric approaches. We also employ an L^2-Distance to measure the difference between two density functions.

A. L^2-Distance for two KDEs

We now derive our approach for computing the divergence between two estimated probability distributions, using the Gaussian kernel density estimator with L^2-Distance. Let Y be a random variable. Consider two N-dimensional vectors $\mathbf{y} = [y_1, y_2, ...y_N]^T$ and $\hat{\mathbf{y}} = [\hat{y}_1, \hat{y}_2, ...\hat{y}_N]^T$, where the y_i's and $\hat{y}_j$'s are randomly drawn from the sample space of Y. The density functions for Y estimated using Gaussian KDE from the two sample vectors, $P_{\mathbf{y}}(Y)$ and $P_{\hat{\mathbf{y}}}(Y)$, can be written as:

$$P_{\mathbf{y}}(Y = y) = \frac{1}{N} \sum_i^N G(y|y_i, h^2)$$

$$P_{\hat{\mathbf{y}}}(Y = y) = \frac{1}{N} \sum_i^N G(y|\hat{y}_i, \hat{h}^2)$$

where h and $\hat{h}$ are the respective smooth window widths. The following theorem presents the closed-form formula for computing the L^2-Distance between two estimated density functions:

Theorem 1: L^2-Distance between $P_{\mathbf{y}}(Y)$ and $P_{\hat{\mathbf{y}}}(Y)$ is:

$$\begin{aligned} L^2(P_{\mathbf{y}}, P_{\hat{\mathbf{y}}}) &= \int (P_{\mathbf{y}}(y) - P_{\hat{\mathbf{y}}}(y))^2 dy \\ &= \frac{1}{N^2} \sum_{i,j}^N \Big[G(y_i|y_j, 2h^2) + G(\hat{y}_i|\hat{y}_j, 2\hat{h}^2) \\ &\quad - 2G(y_i|\hat{y}_j, h^2 + \hat{h}^2) \Big] \end{aligned}$$

a) Proof:: We begin by expressing the L^2-Distance in terms of their KDE functions:

$$\begin{aligned} &L^2(P_{\mathbf{y}}, P_{\hat{\mathbf{y}}}) \\ &= \int (P_{\mathbf{y}}(y) - P_{\hat{\mathbf{y}}}(y))^2 dy \\ &= \int \left(\frac{1}{N} \sum_i^N G(y|y_i, h^2) - \frac{1}{N} \sum_i^N G(y|\hat{y}_i, \hat{h}^2) \right)^2 dy \end{aligned}$$

Expanding the square yields the following expression:

$$\begin{aligned} L^2(P_{\mathbf{y}}, P_{\hat{\mathbf{y}}}) &= \frac{1}{N^2} \sum_{i,j}^N \int G(y|y_i, h^2) G(y|y_j, h^2) dy \\ &+ \frac{1}{N^2} \sum_{i,j}^N \int G(y|\hat{y}_i, \hat{h}^2) G(y|\hat{y}_j, \hat{h}^2) dy \\ &- \frac{2}{N^2} \sum_{i,j}^N \int G(y|y_i, h^2) G(y|\hat{y}_j, \hat{h}^2) dy \quad (3) \end{aligned}$$

Based on the fact that the product of two Gaussian distributions, $G(y|\mu_1, \sigma_1^2)$ and $G(y|\mu_2, \sigma_2^2)$, can be written as follows [9]:

$$G(y|\mu_1, \sigma_1^2) G(y|\mu_2, \sigma_2^2) = G(\mu_1|\mu_2, \sigma_1^2 + \sigma_2^2) \times G(y|\mu_{12}, \sigma_{12}),$$

where $\mu_{12} = \frac{\mu_1 \sigma_2^2 + \mu_2 \sigma_1^2}{\sigma_2^2 + \sigma_1^2}, \sigma_{12} = \sqrt{\frac{\sigma_1^2 \sigma_2^2}{\sigma_1^2 + \sigma_2^2}}$.
Since $\int G(y|\mu, \sigma) = 1$, the integral for the product of two Gaussians can be written as:

$$\int G(y|\mu_1, \sigma_1^2) G(y|\mu_2, \sigma_2^2) dy = G(\mu_1|\mu_2, \sigma_1^2 + \sigma_2^2)$$

Replacing this into Eq. (3) leads to the following;

$$\begin{aligned} &\frac{1}{N^2} \sum_{i,j}^N \Big(G(y_i|y_j, 2h^2) + G(\hat{y}_i|\hat{y}_j, 2\hat{h}^2) \\ &-2G(y_i|\hat{y}_j, h^2 + \hat{h}^2) \Big), \end{aligned}$$

which completes the proof. □

B. DPMTL: Distribution-Preserving MTL Framework

Let $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ be a spatial-temporal dataset, where $\mathcal{X} = \{\mathbf{X}_s | s = 1, 2, ..., S\}$, $\mathcal{Y} = \{\mathbf{y}_s |, s = 1, 2, ...S\}$, and S is the number of locations. Each $X_s \in \mathbb{R}^{N_s \times d}$ is a d-dimensional multivariate time series of predictor variables at location s. Furthermore, N_s is the number of training examples available at location s. The motivation for our spatial-temporal distribution preserving approach is to learn a set of local linear models, $f_s(\mathbf{x}; \mathbf{w}_s)$ that minimizes the prediction error while fitting the marginal distribution of Y.

Learning the local models amounts to estimating the weight matrix $\mathbf{W} = [\mathbf{w}_1\ \mathbf{w}_2 \cdots \mathbf{w}_S]$ for all the stations. Due to the inherent relationships between the prediction tasks at multiple locations, our proposed framework assumes that $\mathbf{W}$ is not a full rank matrix and can be decomposed into a product of two low-rank matrices, $\mathbf{U}$ and $\mathbf{V}$. These low-rank matrices can be derived as follows:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V}} \quad & \mathcal{L}_1 + \alpha\mathcal{L}_2 + \lambda\mathcal{L}_3, \\ \text{s.t.} \quad & \mathbf{W} = \mathbf{U}\mathbf{V},\ \hat{\mathbf{y}}_s = \mathbf{X}_s\mathbf{w}_s \end{aligned} \tag{4}$$

where

$$\begin{aligned} \mathcal{L}_1 &= \sum_s^S ||\mathbf{y}_s - \hat{\mathbf{y}}_s)||_2^2 \\ \mathcal{L}_2 &= \mathbf{tr}[\mathbf{W}(\mathbf{D}-\mathbf{A})\mathbf{W}^T)] \\ \mathcal{L}_3 &= \sum_s^S \Bigg\{ \frac{1}{N_s^2} \sum_{i,j}^{N_s} \Bigg(G(y_{si}|y_{sj}, 2h_s^2) \\ &\quad + G(\hat{y}_{si}|\hat{y}_{sj}, 2\hat{h}_s^2) - 2G(y_{si}|\hat{y}_{sj}, h_s^2 + \hat{h}_s^2) \Bigg) \Bigg\} \end{aligned} \tag{5}$$

Our objective function consists of three loss functions: (1) $\mathcal{L}_1$, which measures the residual errors on the training data, (2) $\mathcal{L}_2$, which is a regularizer to ensure that the model parameters for two neighboring locations should be close to each other, and (3) $\mathcal{L}_3$, which measures the divergence between the true and predicted distributions for Y.

For $\mathcal{L}_2$, an $S \times S$ similarity matrix $\mathbf{A}$ is calculated by applying an RBF kernel on the Haversine distance [10], Q_{ij}, between two locations i and j, i.e., $A_{ij} = \exp[-Q_{ij}/\gamma]$. $\mathbf{W} \in \mathbb{R}^{d\times S}$ is the model parameter matrix, where each column corresponds to the weights of the regression model for a given station. Since $\mathbf{W}(\mathbf{D}-\mathbf{A})\mathbf{W}^T = \sum_s^S \sum_r^S A_{rs}(\mathbf{w}_s - \mathbf{w}_r)^2$, the second term corresponds to a graph Laplacian regularizer to ensure that the spatial autocorrelation is preserved (due to Tobler's First Law of Geography [11]).

The first loss function $\mathcal{L}_1$ is designed to minimize prediction error by learning the conditional probability function $P_{\mathbf{y}_s}(Y|\mathbf{X})$. In contrast, the third loss function, $\mathcal{L}_3$, is designed to accurately fit the marginal distribution $P_{\mathbf{y}_s}(Y)$ by minimizing the L^2-Distance between the true and estimated density functions for all stations using Equation (5). The hyperparameter λ is used to control the trade-off between minimizing the two loss functions. Note that the local models $\mathbf{w}_s$ are assumed to share a common latent matrix $\mathbf{U}$. Such an assumption is useful especially for learning models at locations with limited training data. Specifically, the model parameters for each station s are assumed to be formed using a linear combination of the dictionary (latent factors) in $\mathbf{U}$, with $\mathbf{v}_s$(the s-th column of $\mathbf{V}$) specifying the coefficients of the linear combination.

C. Optimization

We employ a mini-batch accelerated gradient descent approach to solve the optimization problem given in Equation (4). This requires us to derive the gradient of each term, $\mathcal{L}_1$, $\mathcal{L}_2$ and $\mathcal{L}_3$, with respect to the model parameters.

For $\mathcal{L}_1$, the partial derivatives are given by:

$$\frac{\partial\mathcal{L}_1}{\partial\mathbf{U}} = \sum_s^S 2\mathbf{X}_s^T\mathbf{X}_s\mathbf{U}\mathbf{v}_s\mathbf{v}_s^T - 2\mathbf{X}_s^T\mathbf{y}_s\mathbf{v}_s^T \tag{6}$$

$$\frac{\partial\mathcal{L}_1}{\partial\mathbf{v}_s} = 2\mathbf{U}^T\mathbf{X}_s^T\mathbf{X}_s\mathbf{U}\mathbf{v}_s - 2\mathbf{U}^T\mathbf{X}_s^T\mathbf{y}_s \tag{7}$$

For $\mathcal{L}_2$, the partial derivatives are given by:

$$\frac{\partial\mathcal{L}_2}{\partial\mathbf{U}} = 2\alpha\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T \tag{8}$$

$$\frac{\partial\mathcal{L}_2}{\partial\mathbf{V}} = 2\alpha\mathbf{U}^T\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A}) \tag{9}$$

For $\mathcal{L}_3$, we first show the partial derivative of $L2$-Distance with respect to $\hat{y}_{sm}$. The L^2-Distance of each station, given in Equation (5), is composed of three Gaussian kernels. Since $G(y_{si}|y_{sj}, 2h^2)$ is a constant with respect of $\hat{y}_{sm}$, its partial derivative is:.

$$\sum_{i,j}^{N_s} \frac{\partial G(y_{si}|y_{sj}, 2h^2)}{\partial\hat{y}_{sm}} = 0$$

The derivative for the sum of pairwise Gaussian kernel of $\hat{y}_s$ is given as follows:

$$\begin{aligned} & \sum_{i,j}^{N_s} \frac{\partial G(\hat{y}_{si}|\hat{y}_{sj}, 2\hat{h}^2)}{\partial\hat{y}_m} \\ = & \frac{-1}{2\hat{h}^2} G(\hat{y}_{si}|\hat{y}_{sj}, 2\hat{h}^2) \\ & \times \Big[1\{i=m\} - 1\{j=m\} \Big] (\hat{y}_{si} - \hat{y}_{sj}) \\ = & \sum_j^{N_s} \frac{-1}{2\hat{h}^2} G(\hat{y}_{sm}|\hat{y}_{sj}, 2\hat{h}^2)(\hat{y}_{sm} - \hat{y}_{sj}) \\ & - \sum_i^{N_s} \frac{-1}{2\hat{h}^2} G(\hat{y}_{si}|\hat{y}_{sm}, 2\hat{h}^2)(\hat{y}_{si} - \hat{y}_{sm}) \\ = & \frac{-1}{\hat{h}^2} \sum_i^{N_s} G(\hat{y}_{sm}|\hat{y}_{si}, 2\hat{h}^2)(\hat{y}_{sm} - \hat{y}_{si}) \end{aligned}$$

The derivative for the cross-term Gaussian kernels is

$$\begin{aligned} & \sum_{i,j}^{N_s} \frac{\partial G(\hat{y}_{si}|y_{sj}, h^2 + \hat{h}^2)}{\partial\hat{y}_{sm}} \\ = & \sum_{i,j}^{N_s} \frac{-1}{h^2 + \hat{h}^2} G(\hat{y}_{si}|y_{sj}, h^2 + \hat{h}^2) \\ & \times 1\{i=m\}(\hat{y}_{si} - y_{sj}) \\ = & \frac{-1}{h^2 + \hat{h}^2} \sum_i^{N_s} G(\hat{y}_{sm}|y_{si}, h^2 + \hat{h}^2)(\hat{y}_{sm} - y_{si}) \end{aligned}$$

Putting together all three derivatives, the partial derivative of $\mathcal{L}_3$ w.r.t. $\hat{y}_{sm}$ is given by

$$\frac{\partial \mathcal{L}_3}{\partial \hat{y}_{sm}} = \frac{\lambda}{{N_s}^2}\Bigg[\frac{2}{h^2+\hat{h}^2}\sum_i^{N_s} G(\hat{y}_{sm}|y_{si}, h^2+\hat{h}^2)(\hat{y}_{sm}-y_{si}) - \frac{1}{\hat{h}^2}\sum_i^{N_s} G(\hat{y}_{sm}|\hat{y}_{si}, 2\hat{h}^2)(\hat{y}_{sm}-\hat{y}_{si})\Bigg] \quad (10)$$

Furthermore, denoting $\mathbf{x}_{sm} \in \Re^{d\times 1}$ (the m-th row of $\mathbf{X}_s$), the partial derivative of $\hat{y}_{sm}$ with respect to $\mathbf{U}$ and $\mathbf{V}$ are:

$$\frac{\partial \hat{y}_{sm}}{\partial \mathbf{U}} = \mathbf{x}_{sm}^T \mathbf{v}_s^T \in \mathbb{R}^{d\times k}, \quad \frac{\partial \hat{y}_{sm}}{\partial \mathbf{v}_s} = \mathbf{U}^T \mathbf{x}_{sm}^T \in \mathbb{R}^{k\times 1}$$

By applying chain rule, we obtain:

$$\frac{\partial \mathcal{L}_3}{\partial \mathbf{U}} = \sum_s^S \sum_m^{N_s} \frac{\partial \mathcal{L}_3}{\partial \hat{y}_{sm}} \times \frac{\partial \hat{y}_{sm}}{\partial \mathbf{U}} \quad (11)$$

$$\frac{\partial \mathcal{L}_3}{\partial \mathbf{v}_s} = \sum_m^{N_s} \frac{\partial \mathcal{L}_3}{\partial \hat{y}_{sm}} \times \frac{\partial \hat{y}_{sm}}{\partial \mathbf{v}_s} \quad (12)$$

D. Algorithm

Equation (10) requires us to compute both $G(\hat{y}_{sm}|y_{si}, h^2 + \hat{h}^2)$ and $G(\hat{y}_{sm}|\hat{y}_{si}, 2\hat{h}^2)$, which takes $O(N_s)$. Furthermore, computing the partial derivative of $\mathcal{L}_3$ w.r.t all $y_{sm}, s = 1, 2, ..., S, m = 1, 2, ..., N_s$ requires $O(SN^2)$. To speed up the computation, we employ a mini-batch Gradient Descent (MGD) approach at each iteration, where instead of using all N_s points from each station, we use only a subset of size $l < N_s$. This reduces significantly the amount of computations needed from $O(SN^2)$ to $O(Sl^2)$.

For each gradient descent update step at iteration t, let the mini-batch data matrix be $\mathbf{X}_s^{(t)} \in \mathbb{R}^{l\times d}$ and the mini-batch response vector be $\mathbf{y}_s^{(t)} \in \mathbb{R}^{l\times 1}$. For each station s, we compute the following vector $\mathbf{p}_s \in \mathbb{R}^{l\times 1}$ as the derivative of $\frac{1}{2}\mathcal{L}_3$ on $\hat{y}_s$.

$$\mathbf{p}_s = \frac{1}{l^2}\Big(\frac{\mathbf{G}^s \circ \mathbf{E}^s}{h^2+\hat{h}^2} - \frac{\mathbf{H}^s \circ \mathbf{F}^s}{2\hat{h}^2}\Big)\cdot \mathbb{1} \quad (13)$$

$$\begin{aligned} \text{s.t} \quad & \mathbf{G}^s \in \Re^{l\times l} : \mathbf{G}_{ij}^s = G(\hat{y}_{si}^{(t)}|y_{sj}^{(t)}, h^2+\hat{h}^2) \\ & \mathbf{H}^s \in \Re^{l\times l} : \mathbf{H}_{ij}^s = G(\hat{y}_{si}^{(t)}|\hat{y}_{sj}^{(t)}, 2\hat{h}^2) \\ & \mathbf{E}^s \in \Re^{l\times l} : \mathbf{E}_{ij}^s = \hat{y}_{si}^{(t)} - y_{sj}^{(t)} \\ & \mathbf{F}^s \in \Re^{l\times l} : \mathbf{F}_{ij}^s = \hat{y}_{si}^{(t)} - \hat{y}_{sj}^{(t)} \\ & i, j = 1, 2, ..., l \end{aligned}$$

Thus, the gradient w.r.t $\mathbf{U}$ can be obtained by combining Equations (6), (8), and (11) as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = 2\alpha \mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T + 2\sum_s^S \mathbf{X}_s^{(t)T}\mathbf{X}_s^{(t)}\mathbf{U}\mathbf{v}_s\mathbf{v}_s^T - 2\sum_s^S \mathbf{X}_s^{(t)T}(\mathbf{y}_s^{(t)} - \lambda \mathbf{p}_s)\mathbf{v}_s^T \quad (14)$$

Similarly, the gradient w.r.t $\mathbf{V}$ is obtained by combining Equations (7), (9), and (12) to obtain:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{V}} = 2\alpha \mathbf{U}^T\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A}) + [\Delta \mathbf{a}_1, \Delta \mathbf{a}_2, ..., \Delta \mathbf{a}_S] \quad (15)$$

where, $\Delta \mathbf{a}_s = 2\mathbf{U}^T\mathbf{X}_s^{(t)T}\mathbf{X}_s^{(t)}\mathbf{U}\mathbf{v}_s - 2\mathbf{U}^T\mathbf{X}_s^{(t)T}(\mathbf{y}_s^{(t)} - \lambda \mathbf{p}_s)$. Observe that $\mathbf{p}_s$ can be viewed as an adjustment weighted by λ on $\mathbf{y}_s$ when calculating the gradients. In other words, the gradient of the distribution-preserving term $\mathcal{L}_3$ adjusts the role of $\mathbf{y}$ in calculating the gradients. Finally, the Mini-Batch Gradient Descent(MGD) is implemented using the Accelerated Gradient Descent (AGD) approach in order to speed up the search for local optimum [12]. A summary of the algorithm is given in Algorithm 1.

Algorithm 1 DPMTL: Distribution Preserving Multi-task Learning

```
TRAINING PHASE
Input: Training data $\mathcal{X} = \{\mathbf{X}_s\}$, $\mathcal{Y} = \{\mathbf{y}_s\}$, $\mathbf{A}$, $h_s$,$\hat{h}_s$, $\tau_u$, $\tau_v$, $s = 1, 2...S$.
Output: $\mathbf{U}$, $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ...\mathbf{v}_S]$.
while not converge do
    $t = t + 1$;
    for s = 1,2,...,S
        randomly choose $l$ sample data, $\mathbf{X}_s^{(t)}$ and $\mathbf{y}_s^{(t)}$;
        $\hat{\mathbf{y}}_s^{(t)} = \mathbf{X}_s^{(t)}\mathbf{U}\mathbf{v}_s$;
        compute $\mathbf{p}_s$ with equation (13);
    end for
    $\mathbf{U} = \mathbf{U} - \tau_u \frac{\mathcal{L}}{\mathbf{U}}$ by equation (14);
    $\mathbf{V} = \mathbf{V} - \tau_v \frac{\mathcal{L}}{\mathbf{V}}$ by equation (15);
end while

PREDICTION PHASE
Input: Testing data $\mathcal{X}^* = \{\mathbf{X}_s^*\}$, $\mathbf{U}$, $\mathbf{V}$.
Output: Predictions $\mathcal{Y}^* = \{\hat{\mathbf{y}}_s^*\}$.
for s = 1,2,...,S
    $\hat{\mathbf{y}}_s^* = \mathbf{X}_s^*\mathbf{U}\mathbf{v}_s$;
end for
```

IV. Experimental Evaluation

We have conducted extensive experiments to evaluate the performance of our proposed framework. The dataset and baseline methods used in our experiments along with the results obtained are described in this section.

A. Data and Preprocessing

We evaluated the proposed approach on monthly precipitation data from the Global Historical Climatology Network (GHCN) data [13]. The dataset spans a 540-month time period, from January 1970 to December 2014. For brevity, we consider only data from weather stations in the United States (located between $24.74°N$ to $49.35°N$ and $66.95°W$ and $124.97°W$). We also omit any station that has more than 50% missing values in its time series. The resulting dataset contains precipitation data from 1,510 weather stations.

Table II: Predictor Variables from NCEP Reanalysis

Variable	Description
cprat	convective precipitation rate at surface
dlwrf	longwave radiation flux at surface
dswrf	solar radiation flux at surface
lftx	surface lifted index
omega	omega at sigma level 0.995
pr_wtr	precipitable water content
prate	precipitation rate
rhum	relative humidity at sigma level 0.995
slp	Sea level pressure
thick850	thickness for 850-500mb
thick1000	thickness for 1000-500mb
tmax	maximum temperature at 2 m
tmin	minimum temperature at 2 m

We selected 13 predictor variables from the NCEP Reanalysis [14] gridded dataset with the help of our domain expert. A summary of the dataset is shown in Table II. The mapping between the GHCN station and its NCEP Reanalysis grid is established by finding the closest grid cell to each GHCN station. Both the predictor and response variables of each station are deseasonalized by subtracting the seasonal mean for that station and dividing by the corresponding seasonal standard deviation.

B. Baseline Algorithms

We compared the proposed framework, DPMTL, against the following baseline algorithms:

- **Global model**: The data from all stations are combined and used to train a global, lasso regression model.
- **Local model**: A local model is trained for each station using only data from the given station.
- **GSpartan**: A spatio-temporal multi-task regression method proposed in [7].
- **Contour Regression**: A distribution-preserving method for time series prediction [1].

There are two major differences between the contour regression method [1] and DPMTL. First, contour regression is designed to improve distribution fit by minimizing the discrepancy between the empirical cumulative density function of the predicted and ground truth values, whereas DPMTL applies L^2-distance on probability density functions estimated using KDE. Second, contour regression is a single-task learning method, unlike the multi-task learning method used in DPMTL.

C. Evaluation Metrics

The evaluation metrics are defined on two scales—macro and micro. Macro-scale metrics are computed by evaluating the performance of each weather station independently and then taking an average over all the stations. In contrast, micro-scale metrics are computed by concatenating the predictions from all stations into a single vector first before comparing them against the ground truth, which is also from all stations. The macro- and micro-scale metrics used in this study are defined below, where $\hat{\mathbf{y}}_s$ corresponds to the predicted values for station s and $\mathbf{y}_s$ corresponds to their true values.

- **RMSE**: a measure of prediction error obtained by taking the square root of the average sum-of-squared errors in the predictions.

$$\text{Macro RMSE} = \frac{1}{S}\sum_s^S \sqrt{\frac{1}{N_s}\sum_i^{N_s}(y_{si}-\hat{y}_{si})^2}$$

$$\text{Micro RMSE} = \frac{1}{\sum_s^S N_s}\sqrt{\sum_s^S\sum_i^{N_s}(y_{si}-\hat{y}_{si})^2}$$

- **RMS-CDF**: a metric defined in [1] to evaluate the fit between two cumulative distribution functions created from a finite sample of observations. The metric is equivalent to applying RMSE on the ordered values of the data.

$$\text{Macro RMS-CDF} = \frac{1}{S}\sum_s^S \sqrt{\frac{1}{N_s}\sum_i^{N_s}(y_{s(i)}-\hat{y}_{s(i)})^2}$$

$$\text{Micro RMS-CDF} = \frac{1}{\sum_s^S N_s}\sqrt{\sum_s^S\sum_i^{N_s}(y_{s(i)}-\hat{y}_{s(i)})^2}$$

- L^2**-Distance**: another metric for measuring the divergence between two density functions, computed according to the formula given in Theorem 1.

$$\begin{aligned}&\text{Macro } L^2\text{-Distance}\\ =\ & \frac{1}{S}\sum_s^S \frac{1}{N_s^2}\sum_{i,j}^{N_s}\Big(G(y_i|y_j,2h^2)+G(\hat{y}_i|\hat{y}_j,2\hat{h}^2)\\ & -2G(y_i|\hat{y}_j,h^2+\hat{h}^2)\Big)\end{aligned}$$

$$\begin{aligned}&\text{Micro } L^2\text{-Distance}\\ =\ & \frac{1}{\sum_s^S N_s}\sum_{s,r}^S\sum_i^{N_s}\sum_j^{N_r}\Big(G(y_{si}|y_{rj},2h^2)\\ & +G(\hat{y}_{si}|\hat{y}_{rj},2\hat{h}^2)-2G(y_{si}|\hat{y}_{rj},h^2+\hat{h}^2)\Big)\end{aligned}$$

D. Experimental Setup

We apply 9-fold cross validation on the 45-year data (from 1970-2014) to evaluate the performance of various algorithms. Each fold corresponds to 5 years or 60 months worth of data. In each of the 9 rounds, 8 of the folds are selected to be the training set while the remaining fold is used as test set. Since the dataset has been standardized by their corresponding month, we set the Parzen window width h_s and $\hat{h}_s$ to be half of its variance, i.e., 0.5. The spatial autocorrelation matrix $\mathbf{A} = \exp(-\mathbf{Q}/\gamma)$ is computed using the Haversine distance $\mathbf{Q}$, with $\gamma = 100$. The hyper-parameters α and λ are tuned via nested cross-validation on the training data. The number of latent factors k is set to 10 while the mini-batch size l is chosen to be 64. For gradient descent, the step sizes τ_u and τ_v are initialized to 10^{-7} and gradually decreased with increasing number of iterations.

Table III: Summary of 9-fold cross validation results by using 8 folds for training and the remaining fold for testing.

Method	Macro			Micro		
	RMSE	RMS -CDF	L_2	RMSE	RMS -CDF	L_2 $(\times 10^{-5})$
Global	0.7779	0.4484	0.1137	0.7863	0.4588	8.56
Local	0.7790	0.4277	0.1051	0.7869	0.4411	7.88
GSpartan	0.7728	0.4341	0.1090	0.7812	0.4486	8.18
Contour	0.8071	0.3623	0.0784	0.8156	0.3752	5.90
DPMTL	0.8032	0.3524	0.0750	0.8124	0.3672	5.66

Table IV: Summary of 9-fold cross validation results by using only 1 fold for training and the remaining 8 folds for testing.

Method	Macro		Micro	
	RMSE	RMS-CDF	RMSE	RMS-CDF
Contour	1.0967	0.4332	1.8945	1.4712
DPMTL	0.8189	0.3325	0.8209	0.3474

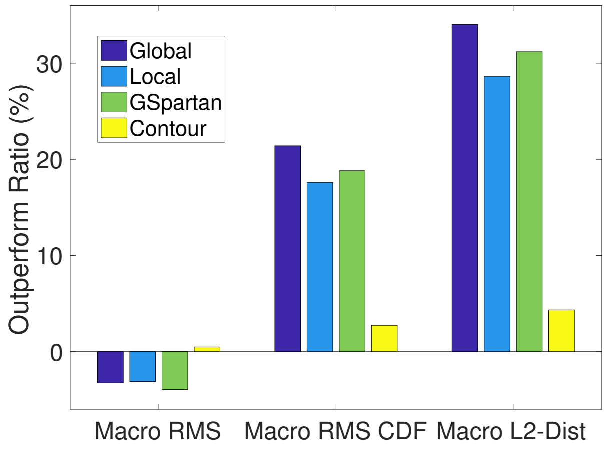
E. Experimental Results

Table III summarizes the performance of the various algorithms. We compute the macro- and micro-scale metrics of the 9-fold nested cross validation and report their average results. Figures 3a and 3b show the relative performance gain/loss of DPMTL compared to each of the baseline algorithms. The outperform ratio is calculated as follows:

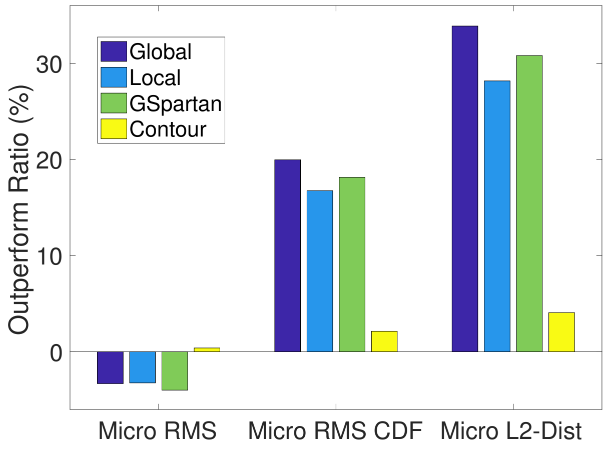
$$\text{Outperform Ratio} = \frac{\text{Error(Baseline)} - \text{Error(DPMTL)}}{\text{Error(Baseline)}}$$

The results suggest that there is a trade-off between minimizing prediction error and distribution error. Non-distribution preserving methods such as global lasso, local lasso, and GSpartan tend to have lower RMSE at the expense of higher distribution errors. The results in Figures 3a and 3b further suggest that, although DPMTL has higher RMSE, the increase in RMSE is only between $3\% - 4\%$. However, the reduction in distribution error it achieves is significantly higher, between $17.6\% - 21.4\%$ for macro average RMS-CDF and $16.7\% - 20.0\%$ for micro average RMS-CDF. DPMTL also outperforms contour regression, which is a distribution-preserving single-task regression method, on all 3 metrics—macro/micro RMSE, macro/micro RMS-CDF, and $L2$-Distance. The gain over contour regression does not look significantly high due to the large size of training data available at each station (using 8 fold of the data for training and remaining fold for testing during cross-validation). Table IV reports the cross validation results when using only 1 fold as training set and the remaining 8 folds as test set. With limited training data, DPMTL significantly outperforms contour regression due to the multi-task learning strategy it employs by leveraging data from other stations to improve its prediction accuracy.

Although the results shown in Table III and Figure 3 provide an aggregated view on the performance of the various algorithms, it is also useful to compare their performance with respect to the individual stations. Figure 4 shows that the percentage of stations in which DPMTL outperforms each baseline method according to the given metrics. The results suggest that DPMTL outperforms the global, local, and



(a) Macro-level evaluation metrics.



(b) Micro-level evaluation metrics.

Figure 3: Outperform ratio of DPMTL over baselines.

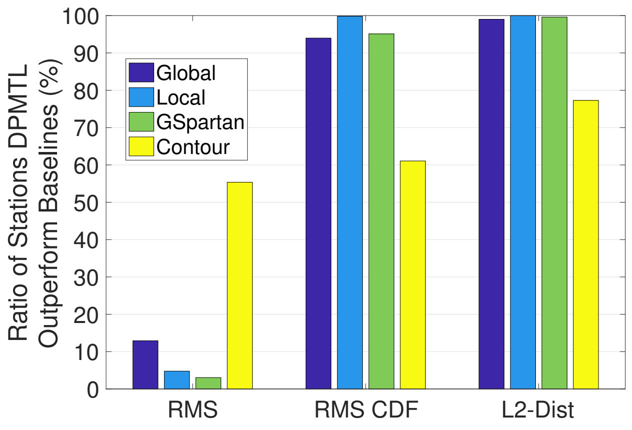
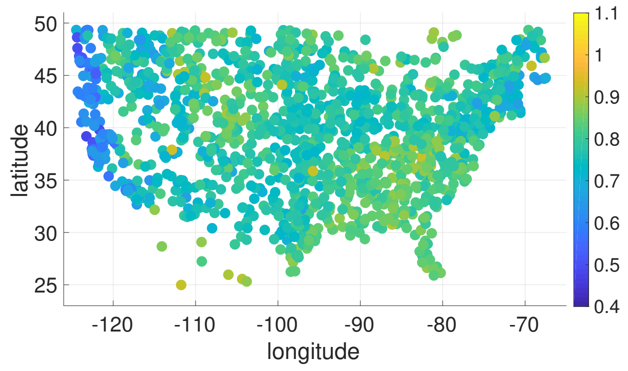
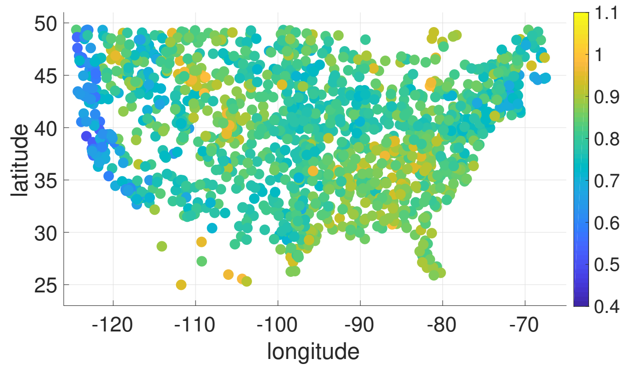


Figure 4: Percentage of stations in which DPMTL outperforms the baseline methods.

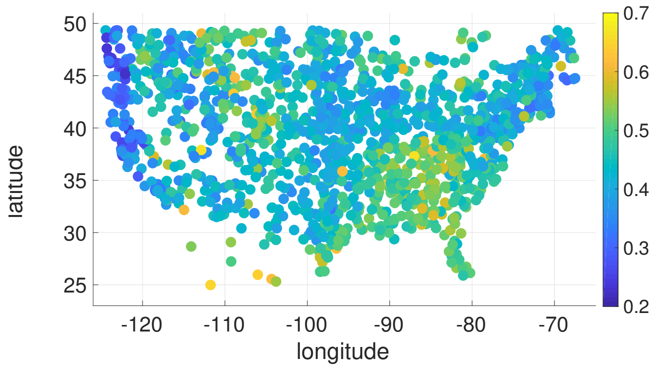


(a) RMSE results for GSpartan.

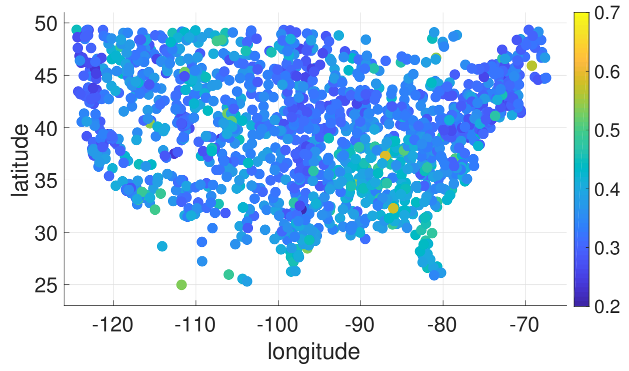


(b) RMSE results for DPMTL.

Figure 5: Comparison between the RMSE of each station for GSpartan and DPMTL (figure best viewed in color).



(a) RMS-CDF error for GSpartan.



(b) RMS-CDF error for DPMTL.

Figure 6: Comparison between the RMS-CDF of each station for GSpartan and DPMTL (figure best viewed in color)

GSpartan models in more than 95% of the stations (in terms of their RMS-CDF) and almost 100% of the stations (in terms of $L2$-Distance). DPMTL is also better than contour regression in 61.06% of the stations (for RMS-CDF) and 77.28% of the stations (for $L2$-Distance).

Figures 5a and 5b compare the RMSE obtained using GSpartan against DPTML for all the stations used in our dataset. While the RMSE for DPMTL is relatively worse than GSpartan, the poor performance of DPMTL occurs at locations where GSpartan also performs poorly. However, in terms of their RMS-CDF, the maps shown in Fig. 6b suggest that the distribution fit improves significantly for DPMTL, for the majority of the stations. GSpartan performs poorly with high prediction and distribution errors especially in the southeastern part of the United States, where there are more variability in their precipitation time series. Although its RMSE is also high for DPMTL in this region, its RMS-CDF improves significantly. The maps also show that the distribution error is generally lower for both methods along the Pacific and Atlantic coastal areas.

Finally, we also examine characteristics of the predicted distribution generated by the different algorithms. Fig. 7 shows the precipitation histograms obtained using DPMTL, contour regression and GSpartan for a station located at $[32.75^\circ N, -97.77^\circ W]$. The results suggest that DPMTL has the best fit to the ground truth distribution, compared to contour regression and GSpartan. In particular, DPMTL was able to capture the dual peaks and heavy tail of the distribution. Both DPMTL and GSpartan also appear to capture the below average precipitation values more effectively than GSpartan. Although the plot was shown only for one of the stations, the good fit obtained by DPTML was found in many other stations in our dataset. In addition to their overall distribution, we also examine how well the algorithms capture other characteristics of the precipitation time series such as their annual cycle (e.g., median monthly values over the prediction period). For example, Fig. 8 shows the median pattern of annual cycle for three stations, located at [26.59° N,-81.86° W], [30.11° N,-98.43° W], and [31.81° N,-106.38° W] respectively. The results once again suggest that DPMTL captures the median monthly values more effectively compared to a non-distribution preserving method such as GSpartan. The ability to better replicate the annual cycle is important particularly for estimating useful metrics such as length of growing season.

In addition to their temporal patterns, we also examine the spatial patterns of the predictions and compare them against the ground truth spatial distribution. Figure 9 shows the spatial distribution of precipitation for three separate time periods. The maps in the first column correspond to the ground

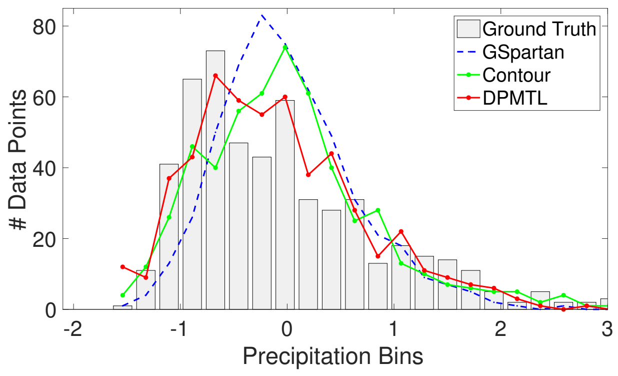
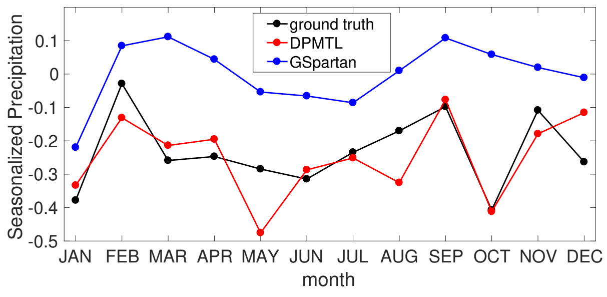
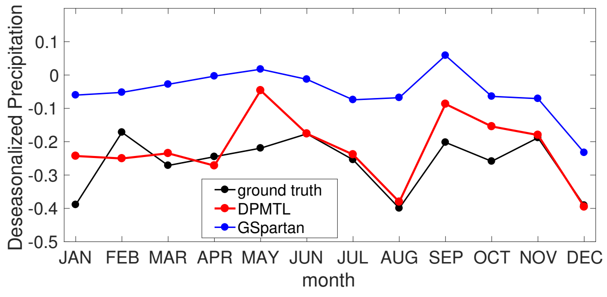


Figure 7: Histogram comparison of an example station.

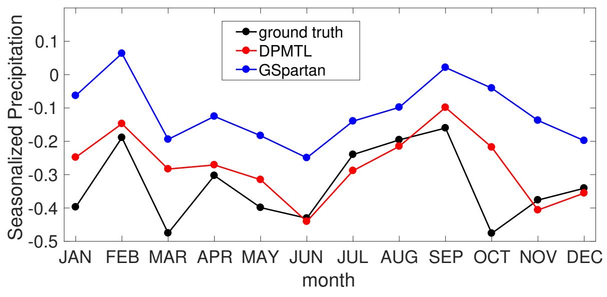
truth spatial distribution of precipitation, while the remaining two columns correspond to the estimated spatial distribution generated by contour regression and DPMTL, respectively. For these plots, both contour regression and DPMTL uses only $1/9$-th of the dataset (from January 1970 to December 1974) as its training set. Observe that the values generated by DPMTL are spatially distributed in a much smoother way, whereas contour regression has a higher tendency of producing more extreme values. As an example, contour regression predicts more extremely high precipitation events in the southern part of the United States for February 1975 compared to DPMTL and the ground truth. In contrast, contour regression predicts more extremely low precipitation events in the midwestern part of the United States for April 1975. DPMTL is more conservative, capturing some of the locations with extreme precipitation values without overestimating them. This is not surprising as MTL can leverage its neighborhood information to diminish some of its predicted extreme values. Nevertheless, DPTML can also miss some of the valid patterns such as the high precipitation events near the southeastern part of the United States in April 1975.



(a) Station at $[26.59^\circ$ N, -81.86° W].



(b) Station at $[30.11^\circ$ N, -98.43° W].



(c) Station at $[31.81^\circ$ N, -106.38° W].

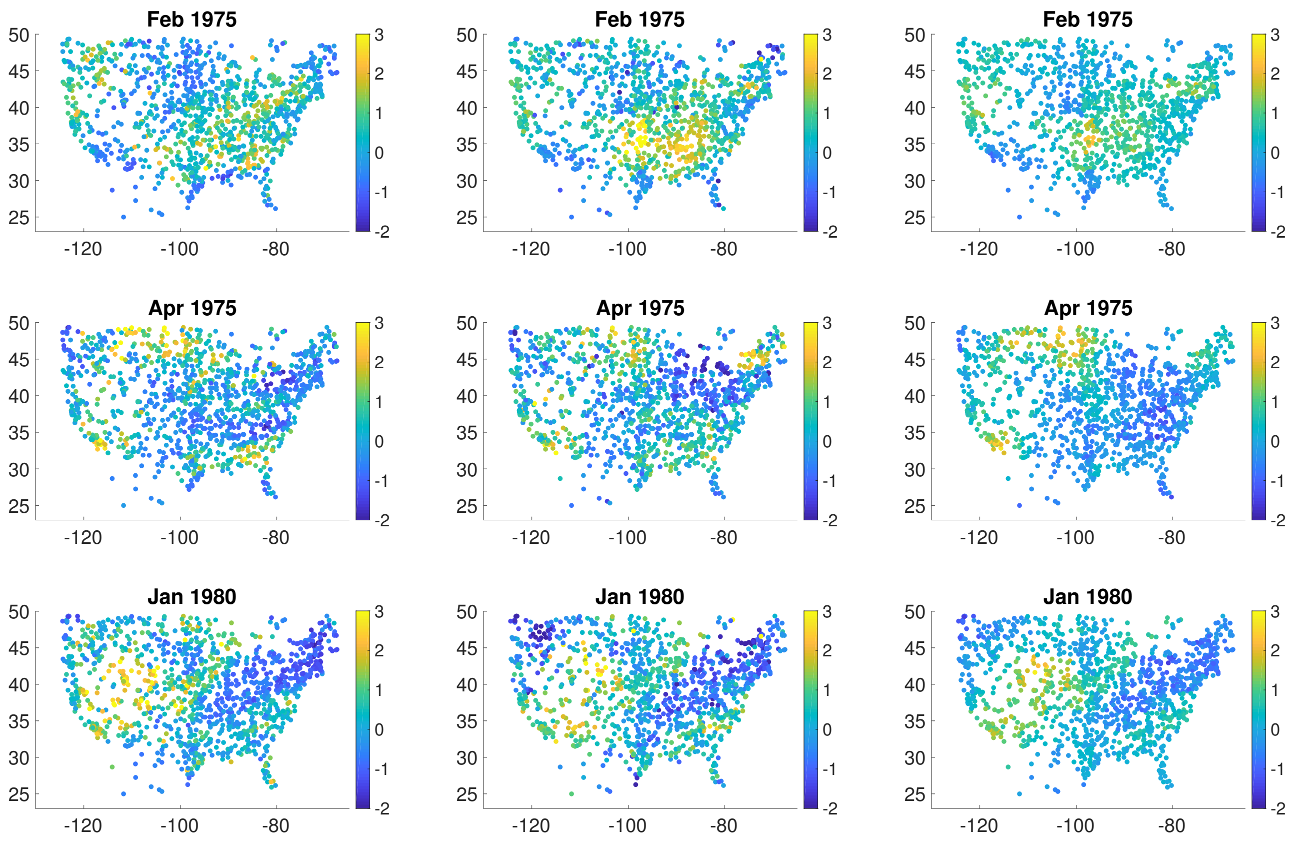
Figure 8: The median annual cycle for three example weather stations in the United States.

V. Related Work

Multi-task learning (MTL) is a machine learning approach for solving multiple, related learning tasks jointly by exploiting the common structure of the problem [4]. MTL improves generalization performance by leveraging domain-specific information to enable the pooling of information across different tasks, which is particularly useful when there are limited training examples available to solve each prediction task separately. It also provides a natural way to handle multi-location prediction problems [5] [6] [7]. For example, in [7], the prediction at each location can be considered a single task, which is related to the prediction tasks at other nearby locations. However, existing MTL methods focus primarily on minimizing point-wise prediction errors, ignoring how well the predicted distribution fits the true distribution of the data. Alternative approaches such as quantile mapping [15] have been developed to correct the bias between the true and predicted distribution. However, such techniques tend to have poor prediction accuracy [1]. While hybrid approaches such as contour regression has been developed [1], they are mostly designed for single-task learning.

VI. Conclusion

This paper presents a distribution preserving multi-task regression framework for spatio-temporal data. Our framework employs a Parzen window based kernel density estimation (KDE) approach to compute the probability density function and $L2$-distance to measure the difference between two distributions. We evaluated our method on a real-world climate dataset, containing more than 1500 stations in the United States, and showed that the proposed framework outperforms existing non-distribution preserving methods in more than 90% of the stations. Furthermore, it also outperforms another distribution preserving method based on single-task learning in more than 60% of the stations.



(a) Ground truth (b) Contour regression. (c) DPMTL.

Figure 9: Spatial maps showing snapshots of monthly precipitation at different time periods.

References

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

[2] Z. Abraham, P.-N. Tan, J. Winkler, S. Zhong, M. Liszewska *et al.*, "Position preserving multi-output prediction," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2013, pp. 320–335.

[3] B. Thrasher, E. P. Maurer, C. McKellar, and P. B. Duffy., "Bias correcting climate model simulated daily temperature extremes with quantile mapping," *Hydrology and Earth System Sciences*, vol. 16, no. 9, p. 3309, 2012.

[4] J. Zhou, J. Chen, and J. Ye, "Malsar: Multi-task learning via structural regularization," *Arizona State University*, vol. 21, 2011.

[5] L. Zhao, Q. Sun, J. Ye, F. Chen, C.-T. Lu, and N. Ramakrishnan, "Multi-task learning for spatio-temporal event forecasting," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2015, pp. 1503–1512.

[6] J. Xu, P.-N. Tan, J. Zhou, and L. Luo, "Online multi-task learning framework for ensemble forecasting," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 6, pp. 1268–1280, 2017.

[7] J. Xu, P.-N. Tan, L. Luo, and J. Zhou, "Gspartan: a geospatio-temporal multi-task learning framework for multi-location prediction," in *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, 2016, pp. 657–665.

[8] M. Sugiyama, S. Liu, M. C. Du Plessis, M. Yamanaka, M. Yamada, T. Suzuki, and T. Kanamori, "Direct divergence approximation between probability distributions and its applications in machine learning," *Journal of Computing Science and Engineering*, vol. 7, no. 2, pp. 99–111, 2013.

[9] P. Bromiley, "Products and convolutions of gaussian probability density functions," *Tina-Vision Memo*, vol. 3, no. 4, p. 1, 2003.

[10] K. von Lindenberg, "Comparative analysis of gps data," *Undergraduate Journal of Mathematical Modeling: One+ Two*, vol. 5, no. 2, p. 1, 2014.

[11] H. J. Miller, "Tobler's first law and spatial analysis," *Annals of the Association of American Geographers*, vol. 94, no. 2, pp. 284–289, 2004.

[12] A. Cotter, O. Shamir, N. Srebro, and K. Sridharan, "Better mini-batch algorithms via accelerated gradient methods," in *Advances in neural information processing systems*, 2011, pp. 1647–1655.

[13] M. Menne, I. Durre, B. Korzeniewski, S. McNeal, K. Thomas, X. Yin, S. Anthony, R. Ray, R. Vose, B. Gleason *et al.*, "Global historical climatology network-daily (ghcn-daily), version 3.22. noaa national climatic data center," 2016.

[14] E. Kalnay, M. Kanamitsu, R. Kistler, W. Collins, D. Deaven, L. Gandin, M. Iredell, S. Saha, G. White, J. Woollen *et al.*, "The ncep/ncar 40-year reanalysis project," *Bulletin of the American meteorological Society*, vol. 77, no. 3, pp. 437–471, 1996.

[15] D. Maraun, "Bias correction, quantile mapping, and downscaling: Revisiting the inflation issue," *Journal of Climate*, vol. 26, no. 6, pp. 2137–2143, 2013.