Distribution Preserving Multi-Task Regression for Spatio-Temporal Data

Xi Liu, Pang-Ning Tan
Michigan State University
{liuxi4, ptan}@cse.msu.edu

Zubin Abraham
Bosch Research, USA
Zubin.Abraham@us.bosch.com

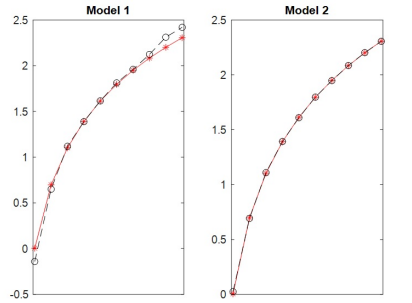
Lifeng Luo, Pouyan Hatami
Michigan State University
{lluo, pouyanhb}@msu.edu

***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 reduces the distribution error for more than 78% of the stations without degrading the prediction accuracy significantly.**

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

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.

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.

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. Therefore, multi-task learning algorithms should be developed for multi-location problems (e.g. [4]–[6]).

This paper presents 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 [6]. 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 our proposed framework outperformed the non-distribution preserving approaches in more than 78% of the weather stations considered in this study, with an average reduction in distribution error between $7.5\% - 17.8\%$. Our approach also outperforms a distribution-preserving single-task regression method called contour regression [1] in at least 78% of the weather stations.

II. Proposed Framework

This section introduces our proposed framework for distribution-preserving multi-task regression.

A. Divergence Measurement

In this paper, we use kernel density estimation (KDE) to estimate the probability density function and the L^2-Distance to measure the divergence between two density functions. 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:

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

where, $G(y|\mu, \sigma)$ represents Gaussian kernel with mean μ and standard deviation σ, and h is the Parzan window width.

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:

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

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

The following theorem presents the closed-form formula for computing the L^2-Distance between two estimated density functions (proof omitted due to lack of space).

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

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

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 $\mathbf{X}_s \in \mathbb{R}^{N_s \times d}$ is a d-dimensional multivariate time series of predictor variables at location s, and each $\mathbf{y}_s \in \mathbb{R}^{N_s}$ is the time series for observations 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 $\mathbf{W} \in \mathbb{R}^{d \times S}$ is not a full rank matrix and can be decomposed into a product of two low-rank matrices, $\mathbf{U} \in \mathbb{R}^{d \times k}$ and $\mathbf{V} \in \mathbb{R}^{k \times S}$, where $k < d$. These low-rank matrices can be derived as follows:

$$\begin{aligned} \min_{\mathbf{U},\mathbf{V}} \quad & \mathcal{L}_1 + \alpha\mathcal{L}_2 + \lambda\mathcal{L}_3, \\ \text{s.t.} \quad & \mathbf{W} = \mathbf{U}\mathbf{V},\ \hat{\mathbf{y}}_s = \mathbf{X}_s\mathbf{w}_s \end{aligned} \tag{2}$$

where

$$\begin{aligned} \mathcal{L}_1 &= \sum_{s}^{S} ||\mathbf{y}_s - \hat{\mathbf{y}}_s)||_2^2 \\ \mathcal{L}_2 &= \mathbf{tr}[\mathbf{W}(\mathbf{D} - \mathbf{A})\mathbf{W}^T)] \\ \mathcal{L}_3 &= \sum_{s}^{S}\Bigg\{\frac{1}{N_s^2}\sum_{i,j}^{N_s}\Bigg(G(y_{si}|y_{sj}, 2h_s^2) \\ &\quad + G(\hat{y}_{si}|\hat{y}_{sj}, 2\hat{h}_s^2) - 2G(y_{si}|\hat{y}_{sj}, h_s^2 + \hat{h}_s^2)\Bigg)\Bigg\} \end{aligned} \tag{3}$$

Our objective function consists of three loss functions: (1) $\mathcal{L}_1$, which measures the residual errors on the training data between ground truth samples in $\mathbf{y}_s$ and estimated samples in $\hat{\mathbf{y}}_s$, (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 [7], 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. The second loss function ensures that spatial autocorrelation is preserved by our framework (following Tobler's First Law of Geography [8]). It accomplishes this by using a graph Laplacian regularizer, where $\mathbf{D}$ is a diagonal matrix whose diagonal elements are $D_{ii} = \sum_j A_{ij}$.

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 (3). 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 (2). 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:

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{U}} = \sum_{s}^{S} 2\mathbf{X}_s^T\mathbf{X}_s\mathbf{U}\mathbf{v}_s\mathbf{v}_s^T - 2\mathbf{X}_s^T\mathbf{y}_s\mathbf{v}_s^T \quad (4)$$

$$\frac{\partial \mathcal{L}_1}{\partial \mathbf{v}_s} = 2\mathbf{U}^T\mathbf{X}_s^T\mathbf{X}_s\mathbf{U}\mathbf{v}_s - 2\mathbf{U}^T\mathbf{X}_s^T\mathbf{y}_s \quad (5)$$

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

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{U}} = 2\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T \quad (6)$$

$$\frac{\partial \mathcal{L}_2}{\partial \mathbf{V}} = 2\mathbf{U}^T\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A}) \quad (7)$$

For $\mathcal{L}_3$, the partial derivative w.r.t. $\hat{y}_{sm}$ is given by:

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

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:

$$\frac{\partial \mathcal{L}_3}{\partial \mathbf{U}} = \sum_{s}^{S}\sum_{m}^{N_s}\frac{\partial \mathcal{L}_3}{\partial \hat{y}_{sm}} \times \frac{\partial \hat{y}_{sm}}{\partial \mathbf{U}} \quad (8)$$

$$\frac{\partial \mathcal{L}_3}{\partial \mathbf{v}_s} = \sum_{m}^{N_s}\frac{\partial \mathcal{L}_3}{\partial \hat{y}_{sm}} \times \frac{\partial \hat{y}_{sm}}{\partial \mathbf{v}_s} \quad (9)$$

D. Algorithm

Equation (8) 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 $\hat{y}_s$.

$$\mathbf{p}_s = \frac{1}{l^2}\left(\frac{\mathbf{G}^s \circ \mathbf{E}^s}{h^2+\hat{h}^2} - \frac{\mathbf{H}^s \circ \mathbf{F}^s}{2\hat{h}^2}\right)\cdot \mathbb{1} \quad (10)$$

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

Thus, the gradient w.r.t $\mathbf{U}$ can be obtained by combining Equations (4), (6), and (8) as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = 2\alpha\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A})\mathbf{V}^T + 2\sum_{s}^{S}\mathbf{X}_s^{(t)T}\mathbf{X}_s^{(t)}\mathbf{U}\mathbf{v}_s\mathbf{v}_s^T - 2\sum_{s}^{S}\mathbf{X}_s^{(t)T}(\mathbf{y}_s^{(t)} - \lambda\mathbf{p}_s)\mathbf{v}_s^T \quad (11)$$

Similarly, the gradient w.r.t $\mathbf{V}$ is obtained by combining Equations (5), (7), and (9) to obtain:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{V}} = 2\alpha\mathbf{U}^T\mathbf{U}\mathbf{V}(\mathbf{D}-\mathbf{A}) + [\Delta\mathbf{a}_1, \Delta\mathbf{a}_2, ..., \Delta\mathbf{a}_S] \quad (12)$$

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 [9]. A summary of the algorithm is given in Algorithm 1.

III. 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 [10]. 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.

We selected 13 predictor variables from the NCEP Reanalysis [11] gridded dataset with the help of our domain expert, such as longwave radiation flux, surface lifted index, and sea level pressure. The mapping between the GHCN station and its NCEP Reanalysis grid is established by finding the closest

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,$\hat{h}_s$, τ_u, τ_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 (10);
  end for
  $\mathbf{U} = \mathbf{U} - \tau_u \frac{\mathcal{L}}{\mathbf{U}}$ by equation (11);
  $\mathbf{V} = \mathbf{V} - \tau_v \frac{\mathcal{L}}{\mathbf{V}}$ by equation (12);
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

grid cell to each GHCN station. Variables of each station is deseasonalized by subtracting the seasonal mean of that station and dividing by the corresponding seasonal standard deviation.

B. Experimental Setup

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** [6]: A MTL framework for spatio-temporal data. This framework is equivalent to setting the hyper-parameter λ in DPMTL to 0 and adding L_1 regularizers to the model parameters $\mathbf{U}$ and $\mathbf{V}$.
- **Contour Regression**: A distribution-preserving method for time series prediction [1]. Unlike DPMTL, 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. Furthermore, contour regression is a single-task learning method, unlike the multi-task learning method used in DPMTL.

The evaluation 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.

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

- **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.

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

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

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

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 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^{-8} and 10^{-7} and gradually decreased with increasing number of iterations.

The hyperparameters α and λ are tuned via nested cross-validation on the training data. Since we want to minimize both residual and distribution errors, their trade-off must be considered during hyperparameter tunning. Let RMSSUM= $(1-\beta)$RMSE + β RMS-CDF, where β is a parameter that controls the tradeoff between minimizing RMSE and RMS-CDF. The hyperparameters for all competing algorithms are chosen in such a way to minimize the RMSSUM on the validation set. To ensure fair comparison, we report the performance of all the algorithms for each β chosen in the range between [0,1]. For example, if $\beta = 0$, the chosen hyperparameters will be biased toward minimizing RMSE.

C. Experimental Results

Figure 2 summarizes the average performance of the various algorithms after 9-fold nested cross validation. For each algorithm, their RMSE and RMS-CDF metrics are computed as β is varied from 0, 0.25, 0.5, 0.75 to 1. The results suggest that Global (lasso) and GSpartan are not sensitive to the

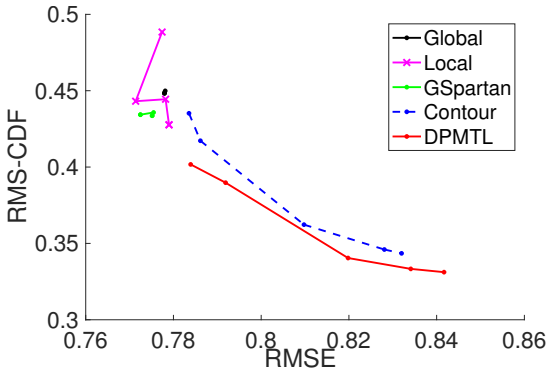


Figure 2: Performance comparison between DPMTL and baseline approaches in terms of RMSE and RMS-CDF when varying the tradeoff parameter β between 0 and 1.

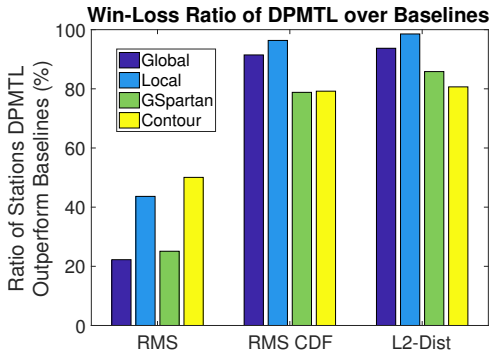
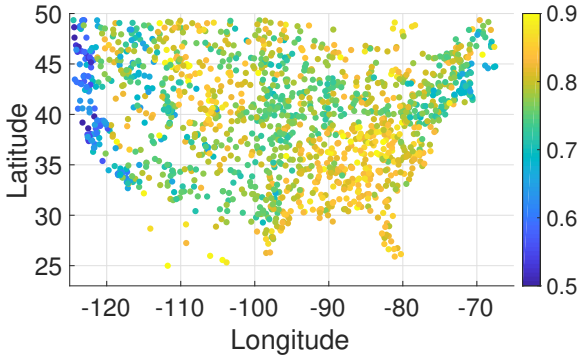
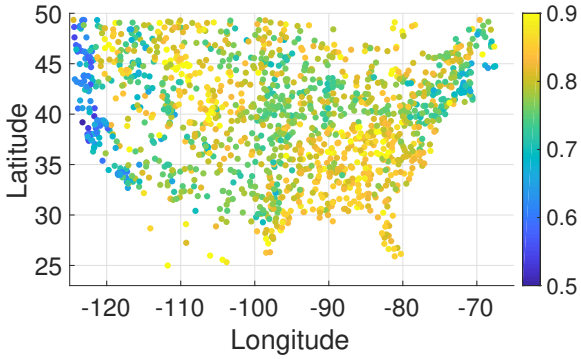


Figure 3: Percentage of stations in which DPMTL outperforms the baseline methods (for β = 0).



(a) RMSE results for GSpartan.



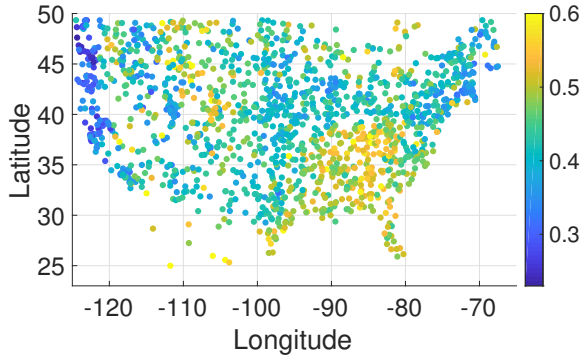
(b) RMSE results for DPMTL.

Figure 4: Comparison between the RMSE of GSpartan and DPMTL for $\beta = 0$ (figure best viewed in color).

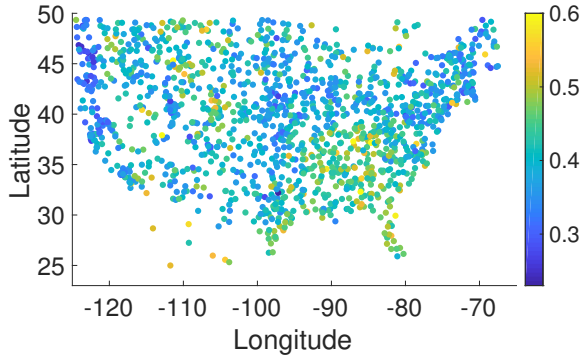
tradeoff parameter β, which is obvious since they are both non-distribution preserving methods. For Local (lasso) models, varying β reduces RMS-CDF slightly, though the change is more erratic as the tuned hyperparameters are highly sensitive to the training set size. For distribution preserving methods such as contour regression and DPMTL, Figure 2 suggests that there is a trade-off between minimizing prediction error and distribution error. Increasing β monotonically reduces RMSE-CDF at the expense of increasing RMSE. When the hyperparameters are tuned to minimize RMSE ($\beta = 0$), DPMTL has the lowest RMS-CDF with only a slight increase in RMSE compared to the non-distribution preserving methods. Specifically, the increase in RMSE is only between $0.75\% - 1.48\%$ compared to the significant reduction in RMS-CDF between $7.5\% - 17.75\%$. DPMTL also outperforms contour regression, a distribution-preserving single-task learning method. In fact, the curve for DPMTL is lower than that for contour regression, which justifies our rationale for developing a distribution-preserving MTL framework. Furthermore, the improvement in RMS-CDF for DPMTL is more pronounced for larger values of β. For example, when $\beta = 0.5$, DPMTL achieves a reduction in RMS-CDF by more than 21.9% compared to global, local, and GSpartan, with an increase in RMSE by at most 5.72%.

We also analyze the performance of the algorithms on a station by station basis. Figure 3 shows the percentage of stations in which DPMTL outperforms each baseline method according to the given metrics. The results show that DPMTL outperforms both Global and Local models in more than 91% of the stations. It also outperforms GSpartan and Contour in more than 78% of the stations (in terms of RMS-CDF) and in more than 80% of the stations (in terms of $L2$-Distance).

Figure 4 compares the RMSE values of GSpartan and DPTML for all stations in the dataset. While the overall RMSE for DPMTL is slightly worse than GSpartan, the maps shown in Figure 4 are quite similar to each other. The poor performance of DPMTL tends to occur at locations where GSpartan also has high RMSE. However, in terms of their RMS-CDF, the maps shown in Figure 5 suggest that the distribution fit of DPMTL improves significantly 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 the 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.



(a) RMS-CDF error for GSpartan.



(b) RMS-CDF error for DPMTL.

Figure 5: Comparison between the RMS-CDF of GSpartan and DPMTL for $\beta = 0$ (figure best viewed in color).

Finally, we also examine characteristics of the predicted distribution generated by different algorithms. Figure 6 shows the precipitation histograms obtained using DPMTL, contour regression, and GSpartan for a station located at $[38.25°N, 82.99°W]$. 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 skewness and heavy tail distribution. DPMTL also fits the distribution of below average precipitation values more effectively than the other two methods. Although the plot was shown only for one station, the good fit obtained by DPTML was found in many other stations in our dataset.

IV. 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 four other competing baselines in at least 78% of the stations.

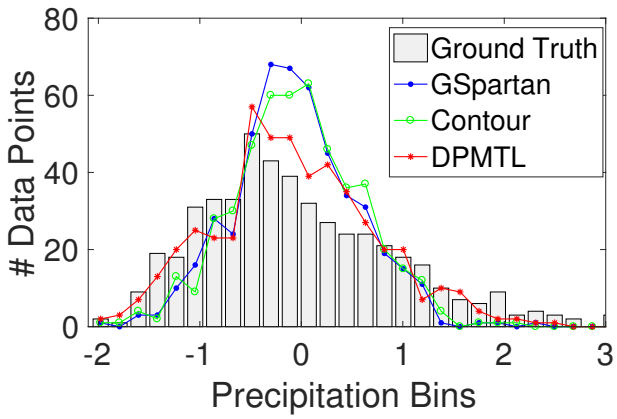


Figure 6: Histogram comparison of precipitation distribution for a weather station located at $[38.25°N, 82.99°W]$.

Acknowledgment

This research is partially supported by the National Science Foundation under grant IIS-1615612. Any use of trade, firm, or product names is for descriptive purposes only and does not imply endorsement by the U.S. Government.

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. Xu, X. Liu, T. Wilson, P.-N. Tan, P. Hatami, and L. Luo, "Muscat: Multi-scale spatio-temporal learning with application to climate modeling." in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI)*, 2018, pp. 2912–2918.

[5] J. Xu, J. Zhou, P.-N. Tan, X. Liu, and L. Luo, "Wisdom: Weighted incremental spatio-temporal multi-task learning via tensor decomposition," in *IEEE International Conference on Big Data*. IEEE, 2016, pp. 522–531.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.