Incomplete Label Multi-task Deep Learning for Spatio-temporal Event Subtype Forecasting(Supplemental Material)

Related Work

Spatio-temporal Event Forecasting. Most previous research in this area has focused on temporal event, in various studies on forecasting elections (O'Connor et al. 2010), stock market movements (Argyriou, Evgeniou, and Pontil 2007), disease outbreaks (Achrekar et al. 2011), and box office ticket sales (Arias, Arratia, and Xuriguera 2013). There are several existing approaches that provide true spatiotemporal resolution for predicted events. For example, Gerber et al. (Gerber 2014) utilized a logistic regression model for spatiotemporal event forecasting using topic-related tweet volumes as features, Ramakrishnan et al. (Ramakrishnan et al. 2014) built separate LASSO models for different locations to predict the occurrence of civil unrest events, and Zhao et al. (Zhao et al. 2015a) designed a new predictive model based on a topic model that jointly characterizes the temporal evolution in terms of both the semantics and geographical burstiness. However, all these focus on the occurrence only, and are not able to handle specific subtypes of future events. There are few existing reports of research on event subtype forecasting. Ning et al. (Ning et al. 2016) performs a primitive experiment in forecasting event populations, but the model proposed in this paper is designed for the distant supervised learning setting (e.g., multi-instance learning), which can not be applied directly to generic multi-class classification.

Multi-task Learning. Multi-task learning (MTL) refers to models that learn multiple related tasks simultaneously to improve their generalization performance (Arias, Arratia, and Xuriguera 2013; Thrun and OSullivan 1998). For event forecasting, many MTL approaches have been proposed (Tutz 2003). For example, Evgeniou et al. (Evgeniou and Pontil 2004) proposed a regularized MTL framework that constrains all task models to be close to each other. The task relatedness can also be modeled by constraining multiple tasks to share a common underlying structure, e.g., a common set of features (Argyriou, Evgeniou, and Pontil 2007), or a common subspace (Ando and Zhang 2005). Zhao et al. (Zhao et al. 2015b) demonstrated the utility of applying a Multi-Task Learning framework for spatiotemporal event forecasting.

Recent years, multi-task learning has also been well studied by the deep learning community. One of the most widely used multi-task learning models was proposed by Caruana (Caruana 1998; Caruna 1993). This model has a shared-bottom structure, where the bottom hidden layers are shared across tasks. Instead of having shared hidden layers and same model parameters across tasks, more recent approaches for modeling task relationships add different types of constraints to task specific parameters (Duong et al. 2015; Misra et al. 2016; Yang and Hospedales 2016). For example, when training two tasks, Duong et al. (Duong et al. 2015) adds L-2 constraints between the two sets of parameters, while a cross stitch network (Misra et al. 2016) learns a unique combination of task-specific hidden layer embedding for each task and Yang et al. (Yang and Hospedales 2016) uses a tensor factorization model to generate hidden layer parameters for each task. However, none of these frameworks can be adopted directly for the spatial event subtype forecasting problem as the challenges mentioned in the introduction neutralize the existing approaches.

Lemma 1's Proof

For a time interval t, given two locations i and j that are close in geo-spatial distance, the probability of the event subtype $\mathcal{C}_a$ at location i denoted as $P(Y_{i,t} = \mathcal{C}_a|X_{i,t})$, will be similar to that at location j, leads to the following equation:

$$P(Y_{i,t} = \mathcal{C}_a|X_{i,t}) \approx P(Y_{j,t} = \mathcal{C}_a|X_{j,t})$$

Likewise, the ratio of the probability of the event subtype at location i being equal to event subtype $\mathcal{C}_a$ compared to event subtype $\mathcal{C}_b$, should also be similar to that at location j. This can be expressed as:

$$\frac{P(Y_{i,t} = \mathcal{C}_a|X_{i,t})}{P(Y_{i,t} = \mathcal{C}_b|X_{i,t})} \approx \frac{P(Y_{j,t} = \mathcal{C}_a|X_{j,t})}{P(Y_{j,t} = \mathcal{C}_b|X_{j,t})} \quad (1)$$

The posterior probability $P(Y_{i,t} = \mathcal{C}_a|X_{i,t})$ can be equivalently represented by any multi-class based models. The similarity pattern based on the ratio of the probability in Equation (1) can thus be equivalently denoted by input X and weight coefficient Θ, as shown in Lemma 1.

Lemma 1. *Based on the model shown in Equation (2) , Equation* (1) *is theoretically equivalent to the following:*

$$X_i(\Theta_{i,a} - \Theta_{i,b})^T \approx X_j(\Theta_{j,a} - \Theta_{j,b})^T \quad (2)$$

where i and j are two tasks that are close in geo-spatial distance and a and b are any two different event subtypes.

Proof. We can derive the lemma from the following equations:

$$P(Y_{i,t} = \mathcal{C}_a|X_{i,t}) = e^{X_{i,t}\Theta_{i,a}^T} / \sum_{k=1}^{K} e^{X_{i,t}\Theta_{i,k}^T} \quad (3)$$

Equation (3) is the definition of the softmax regression. From this, we can derive an equivalent expression in logarithmic form as follows:

$$\log P(Y_{i,t} = \mathcal{C}_a|X_{i,t}) = \log e^{X_{i,t}\Theta_{i,a}^T} - \log \sum_{k=1}^{K} e^{X_{i,t}\Theta_{i,k}^T}$$

We can now subtract any pair of classes $\mathcal{C}_a$ and $\mathcal{C}_b$ to omit the common denominator, as shown below:

$$\log \frac{P(Y_{i,t} = \mathcal{C}_a|X_{i,t})}{P(Y_{i,t} = \mathcal{C}_b|X_{i,t})} = X_{i,t}\Theta_{i,a}^T - X_{i,t}\Theta_{i,b}^T \quad (4)$$

Thus, combining Equation (1) and Equation (4), the proof is completed. □

Theorem 1's Proof

Theorem 1. *In the SIMDA framework, for any deep learning architectures that use the softmax function as their output layer, equation* (1) *is theoretically equivalent to the following:*

$$f(X_i)(\Theta_{i,a} - \Theta_{i,b})^T \approx f(X_j)(\Theta_{j,a} - \Theta_{j,b})^T \quad (5)$$

where $\Theta_{i,b}$ denotes the task specific output layer weight coefficient vector for task i and class $\mathcal{C}_b$.

Proof. We can derive the theorem from the following equations:

$$P(Y_{i,t} = \mathcal{C}_a|X_{i,t}) = \frac{e^{f(X_{i,t})\Theta_{i,a}^T}}{\sum_{k=1}^{K} e^{f(X_{i,t})\Theta_{i,k}^T}} \quad (6)$$

Equation (6) is the definition of the posterior probability of the softmax output layer for a given input X and function $f(\cdot)$. From this, we can derive an equivalent expression in logarithmic form as follows:

$$\log P(Y_{i,t} = \mathcal{C}_a|X_{i,t}) = \log e^{f(X_{i,t})\Theta_{i,a}^T} - \log \sum_{k=1}^{K} e^{f(X_{i,t})\Theta_{i,k}^T}$$

We can now subtract any pair of classes $\mathcal{C}_a$ and $\mathcal{C}_b$ to omit the common denominator in equation (6), as shown below:

$$\log \frac{P(Y_{i,t} = \mathcal{C}_a|X_{i,t})}{P(Y_{i,t} = \mathcal{C}_b|X_{i,t})} = f(X_{i,t})\Theta_{i,a}^T - f(X_{i,t})\Theta_{i,b}^T \quad (7)$$

Thus, combining Equation (1) and Equation (7), we can safely conclude that given two tasks i and j that are close geo-spatially, the difference between the products of the hidden representation of corresponding input and weight coefficients of any pair of classes $f(X_i)(\Theta_{i,a} - \Theta_{i,b})^T$ and $f(X_j)(\Theta_{j,a} - \Theta_{j,b})^T$ should be similar, as shown in Equation (5). □

Algorithm

Based on the ADMM formulation, the original objective function of SIMDA can now be re-written as follows:

$$\mathcal{L}_D(\Phi, \Theta) + \frac{\beta}{2} \sum_s^{\mathcal{S}} \sum_{i,j}^{C_k^2} \|Z_s(V_{s,i} - V_{s,j})^T - \frac{1}{N_s} \sum_c^{\mathcal{S}} adj(s,c) Z_c(W_{c,i} - W_{c,j})^T\|_2^2 \quad (8)$$
$$\text{s.t. } \Theta = V, \Theta = W, Z = f(X)$$

Thus, by decoupling the output layer parameter set Θ that appears both in deep model loss and regularization term, the original problem is transformed into a simpler one with auxiliary variables V, W and Z. The augmented Lagrangian that uses additional quadratic penalty terms with penalty parameter ρ is further computed as follows:

$$\underset{\Phi,\Theta,V,W,Z}{\operatorname{argmin}} \mathcal{L}_D(\Phi,\Theta) + \operatorname{tr}(y^{(1)}(Z - f(X))^T) + \frac{\rho}{2}||Z - f(X)||_2^2 +$$
$$\frac{\beta}{2} \sum_s^{\mathcal{S}} \sum_{i,j}^{C_k^2} \|Z_s(V_{s,i} - V_{s,j})^T - \frac{1}{N_s} \sum_c^{\mathcal{S}} adj(s,c) Z_c(W_{c,i} - W_{c,j})^T\|_2^2$$
$$+ \operatorname{tr}(y^{(2)}(\Theta - V)^T) + \frac{\rho}{2}||\Theta - V||_2^2 + \operatorname{tr}(y^{(3)}(\Theta - W)^T) + \frac{\rho}{2}||\Theta - W||_2^2$$

where the $tr(\cdot)$ operator denotes the trace of the matrix which will return the sum of the elements on the main diagonal.

The pseudo-code of the proposed algorithm is summarized in **Algorithm 1**. The parameter set $\{\Phi, \Theta, V, W, Z, y^{(1)}, y^{(2)}, y^{(3)}\}$ is alternately solved by the proposed algorithm until convergence is achieved. Lines 3-13 show the alternating optimization for each of the variables. The detailed optimization for all the variables are described in more detail below.

Algorithm 1: The Proposed Algorithm

Require: $X, Y, \rho, \beta, \lambda$
Ensure: solution Φ, Θ
1: initialize $\Phi^0, \Theta^0, V^0, W^0, Z^0, y^{(1)0}, y^{(2)0}, y^{(3)0}, i = 0$
2: **repeat**
3: $\Phi^i, \Theta^i \Leftarrow$ Equation (9)
4: **for** $s \Leftarrow 1$ **to** K **do**
5: $V_s^i \Leftarrow$ Equation (12)
6: **end for**
7: **for** $c \Leftarrow 1$ **to** K **do**
8: $W_c^i \Leftarrow$ Equation (14)
9: **end for**
10: **for** $s \Leftarrow 1$ **to** K **do**
11: $Z_s^i \Leftarrow$ Equation (16)
12: **end for**
13: $y^{(1)i}, y^{(2)i}, y^{(3)i} \Leftarrow$ Equation (17)
14: $i \Leftarrow i + 1$
15: **until** convergence

Update Φ, Θ

The sub-problem of updating Φ and Θ is jointly handled as follows:

$$\underset{\Phi,\Theta}{\operatorname{argmin}} \mathcal{L}_D(\Phi,\Theta) + \operatorname{tr}(y^{(1)}(Z - f(X))^T) + \frac{\rho}{2}||Z - f(X)||_2^2 + \quad (9)$$
$$\operatorname{tr}(y^{(2)}(\Theta - V)^T) + \frac{\rho}{2}||\Theta - V||_2^2 + \operatorname{tr}(y^{(3)}(\Theta - W)^T) + \frac{\rho}{2}||\Theta - W||_2^2$$

Since $\mathcal{L}_D(\Phi, \Theta)$ is a non-convex function with respect to Φ and Θ, we will use the conventional Stochastic Gradient Descent and Backpropagation to get local optima.

Update V

The sub-problem of updating V is as follows:

$$\underset{V}{\operatorname{argmin}} \frac{\rho}{2}||\Theta - V||_2^2 + \operatorname{tr}(y^{(2)}(\Theta - V)^T) + \quad (10)$$
$$\frac{\beta}{2} \sum_s^{\mathcal{S}} \sum_{i,j}^{C_k^2} \|Z_s(V_{s,i} - V_{s,j})^T - \frac{1}{N_s} \sum_c^{\mathcal{S}} adj(s,c) Z_c(W_{c,i} - W_{c,j})^T\|_2^2$$

The proposed regularization term introduces some difficulties for updating V, since every pair of class parameters for the same task is coupled in the same term. This makes elemental-wise updating of V impossible.

In order to address it, we treat the combination of every pair of classes as the matrix representation $M \in \mathbb{R}^{k \times C_k^2}$, and reformulate the problem as matrix multiplication as follows:

$$\underset{V}{\operatorname{argmin}} \frac{\rho}{2}||\Theta - V||_2^2 + \operatorname{tr}(y^{(2)}(\Theta - V)^T) + \tag{11}$$
$$\frac{\beta}{2}\sum_s^{\mathcal{S}} \|Z_s V_s^T M - \frac{1}{N_s}\sum_c^{\mathcal{S}} adj(s,c) Z_c W_c^T M\|_2^2$$

where matrix M represents all possible combinations of different classes. For instance, given 3 classes, the matrix M will be as follows:

$$M = \begin{bmatrix} 1 & 1 & 0 \\ -1 & 0 & 1 \\ 0 & -1 & -1 \end{bmatrix}$$

Now we can derive the analytical solution for V_s as:

$$vec(V_s) = \left(\beta(Z_s^T Z_s) \otimes (MM^T) + \rho I\right)^{-1} \tag{12}$$
$$vec\left(y_s^{(2)} + \rho\Theta_s + \beta M\left(\frac{1}{N_s}\sum_c^{\mathcal{S}} adj(s,c) Z_c W_c^T M\right)^T Z_s\right)$$

where $\otimes$ is the Kronecker product operator; and $vec(\cdot)$ is the vector obtained from the matrix by stacking its columns.

Update W

Similarly, the sub-problem of updating W is as follows:

$$\underset{W}{\operatorname{argmin}} \frac{\rho}{2}||\Theta - W||_2^2 + \operatorname{tr}(y^{(3)}(\Theta - W)^T) + \tag{13}$$
$$\frac{\beta}{2}\sum_s^{\mathcal{S}} \|Z_s V_s^T M - \frac{1}{N_s}\sum_c^{\mathcal{S}} adj(s,c) Z_c W_c^T M\|_2^2$$

The analytical solution for W_c then becomes:

$$vec(W_c) = \left(\beta\sum_s^{\mathcal{S}} \frac{adj(s,c)^2}{N_s^2}(Z_c^T Z_c) \otimes (MM^T) + \rho I\right)^{-1} \tag{14}$$
$$vec\left(y_c^{(3)} + \rho\Theta_c - \beta\sum_s^{\mathcal{S}} M\left(\frac{1}{N_s}\sum_{i\neq c}^{\mathcal{S}} adj(s,i) Z_i W_i^T M - Z_s V_s^T M\right) Z_c\right)$$

Update Z

The sub-problem of updating Z is as follows:

$$\underset{Z}{\operatorname{argmin}} \frac{\rho}{2}||Z - f(X)||_2^2 + \operatorname{tr}(y^{(1)}(Z - f(X))^T) + \tag{15}$$
$$\frac{\beta}{2}\sum_s^{\mathcal{S}} \|Z_s V_s^T M - \frac{1}{N_s}\sum_c^{\mathcal{S}} adj(s,c) Z_c W_c^T M\|_2^2$$

The analytical solution for Z_s is straightforward:

$$Z_s = \left(-y_s^{(1)} + \rho f(X_s) + \beta\left(\frac{1}{N_s}\sum_c^{\mathcal{S}} adj(s,c) Z_c W_c^T M\right) M^T V_s\right)$$
$$\left(\beta V_s^T M M^T V_s + \rho I\right)^{-1} \tag{16}$$

Update y

Finally, update $y^{(1)}, y^{(2)}, y^{(3)}$ as follows:

$$y^{(1)} = y^{(1)} + \rho(Z - f(X)), y^{(2)} = y^{(2)} + \rho(\Theta - V) \tag{17}$$
$$y^{(3)} = y^{(3)} + \rho(\Theta - W)$$

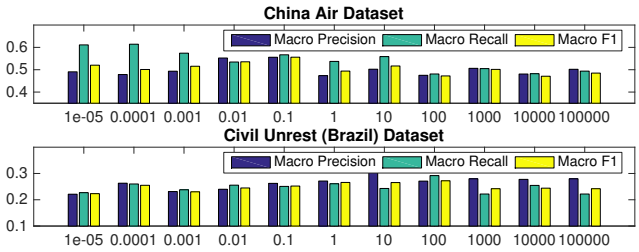

Figure 1: Sensitivity analysis for hyper-parameter β

Baselines and parameters setting

1) *SVC1V1* (Support Vector Classifier with OneVsOne) is a C-Support Vector Classifier with OneVsOne binary decomposition. The kernel function is set to linear and the regularization parameter C is set based on 10 fold cross validation.

2) *SVC1VA* (Support Vector Classifier with OneVsAll) is a C-Support Vector Classifier with OneVsAll binary decomposition. These two methods represent the main approaches generally used to apply SVM to multi-class problems. The kernel function is set to linear and the regularization parameter C is set based on 10 fold cross validation.

3) *SR* (Softmax Regression) is a classification method that generalizes logistic regression to multi-class classification problems . It has no tuning parameter.

4) *MLP* (MultiLayer Perceptron) is the plain vanilla neural network model. The network structure is tuned via the validation set.

5) *SBM* (Shared-Bottom Model) is a multi-task deep learning approach where all tasks share the same bottom hidden layers. The network structure is tuned via the validation set. The detail introduction and hyper-parameter setting is included in supplemental material.

Parameter Sensitivity Study

The hyper-parameter in the proposed SIMDA model is β, where it controls the proposed regularization term on Θ. Figure 1 shows the macro average precision, recall and F1-score of the model for various values of β. Once again, only the results for the Brazil dataset are shown here to represent the civil unrest datasets due to space limitations. The China air dataset shown here is the next day (1-day) air pollution event forecasting format. The bar chart at the top of Figure 1 shows the macro average precision, recall and F1-score of the model versus various values of β for the China air dataset. By varying β across a range from 0.00001 to 100000, the performance, especially for the macro average F1-score, exhibits a concave curvature, peaking at a β value between 0.01 and 0.1. For the civil unrest dataset, a similar concave curvature is found for the model performance by varying β across the same range. In general, the performance is good when β is small, but deteriorates once β becomes too large. This is because a large β will force the model to pay too much attention to ensuring similarities between adjacent tasks which lead to the loss of their own characteristics and a consequent decrease in overall performance.