Joint Learning of Representation and Structure for Sparse Regression on Graphs[*]

Chao Han[†] Shanshan Zhang[†] Mohamed Ghalwash[†‡] Slobodan Vucetic[†]
Zoran Obradovic[†]

Abstract

In many applications, including climate science, power systems, and remote sensing, multiple input variables are observed for each output variable and the output variables are dependent. Several methods have been proposed to improve prediction by learning the conditional distribution of the output variables. However, when the relationship between the raw features and the outputs is nonlinear, the existing methods cannot capture both the nonlinearity and the underlying structure well. In this study, we propose a structured model containing hidden variables, which are nonlinear functions of inputs and which are linearly related with the output variables. The parameters modeling the relationships between the input and hidden variables, between the hidden and output variables, as well as among the output variables are learned simultaneously. To demonstrate the effectiveness of our proposed method, we conducted extensive experiments on eight synthetic datasets and three real-world challenging datasets: forecasting wind power, forecasting solar energy, and forecasting precipitation over U.S. The proposed method was more accurate than state-of-the-art structured regression methods.

1 Introduction

Structured learning models such as Conditional Random Fields (CRFs) [3]) have been widely used for classification and segmentation, since their inception a decade ago. However, use of structured models for regression is less explored. Recent years witnessed development of Gaussian CRFs (GCRFs) [10, 8, 13, 14], which are elegant and powerful models for prediction of interdependent continuous output variables given high-dimensional input variables. GCRFs model the conditional probability of outputs given inputs as a multivariate Gaussian distribution. The originally proposed GCRFs [8, 13] can model linear dependencies between inputs and outputs. Such models result in convex optimization, but might be too rigid for practical applications. To improve the representational power, the authors of [12] train GCRF on transformed features that are non-linearly projected from the original features using unsupervised methods such as radial basis functions (RBFs). However, the representations learned by unsupervised methods are not necessarily optimized for regression. Another recently proposed idea are Neural GCRFs [9], which is a model used for expert integration. The idea of Neural GCRF is to allow experts to be a nonlinear combination of inputs. The limitation of the Neural GCRF is that it is constrained to a specific class of experts integration problems, and it cannot learn underlying structure among output variables.

The objective of this paper is to improve the representational power of a general class of GCRFs proposed in [13]. Our proposed solution relies on introduction of hidden variables that are nonlinear functions of input variables, such that our probabilistic framework for structured regression learns more informative representations and structural dependencies *simultaneously.* In such manner, our model can (1) model complex relationships between inputs and outputs; (2) improve modeling of relationships between outputs; and (3) enhance the structure learning with better representations.

It should be mentioned that modeling of complex relationships in CRFs used for classification is well studied, and that some of those ideas serve as inspirations for our proposed approach. However, our proposed modeling is different from any of previously published works. A common way to increase representational power of classification CRFs is to exploit kernels in the potential functions of CRFs [4, 11], but it comes at the price of scalability. Another type of methods, called hidden state CRFs [5, 7] introduce discrete hidden variables between outputs and inputs to model high order dependencies, and allow for more flexible decision boundaries of CRFs. In contrast the proposed model introduces continuous hidden variables to stand for new input variables. Neural CRFs [1, 6] for structured classification

[*]This research was supported in part by DARPA grant FA9550-12-1-0406 negotiated by AFOSR, NSF BIGDATA grant 14476570, NSF grant NSF-IIS-1117433, and ONR grant N00014-15-1-2729.

[†]Center for Data Anaytics and Biomedical Informatics, Temple University, zoran.obradovic@temple.edu

[‡]Assistant Professor, Ain Shams University, Egypt.

were also proposed for introducing nonlinear features, but our proposed approach aims for solving structured regression.

2 Sparse Gaussian Conditional Random Fields

In this paper, we use capital letters to denote matrices, bold lower-case letters to denote column vectors and lower-case letters to denote scalars. For example, $\boldsymbol{x}$ represents a column vector and X represents a matrix. The ith row of X is denoted as $X_{i:}$.

Suppose we are given a dataset with m i.i.d graph instances, where each graph instance has n input variables and p output variables. Let $\boldsymbol{x} \in \mathbb{R}^n$ denote the input variables and $\boldsymbol{y} \in \mathbb{R}^p$ denote the output variables. Gaussian Conditional Random Fields (GCRF)[8, 10] model the conditional distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ as a multivariate Gaussian distribution. The probability density function (pdf) of GCRF over a single graph instance as defined in [13] is

$$P(\boldsymbol{y}|\boldsymbol{x}, \Lambda, \Theta) = \frac{1}{Z(\boldsymbol{x})} \exp(-\boldsymbol{y}^T \Lambda \boldsymbol{y} - 2\boldsymbol{x}^T \Theta \boldsymbol{y}), \tag{2.1}$$

where the inverse covariance matrix $\Lambda \in \mathbb{R}^{p \times p}$ models the structure (or conditional dependencies) among p output variables, and $\Theta \in \mathbb{R}^{n,p}$ models the dependency of p output variables on n input variables. $Z(\boldsymbol{x})$ is the partition function, which is the integral of the exponent term over $\boldsymbol{y}$. The pdf of GCRF in (2.1) is a multivariate Gaussian with expectation $-\Lambda^{-1}\Theta^T\boldsymbol{x}$ and covariance Λ^{-1}. Therefore, the negative log-likelihood $\log P(\boldsymbol{y}|\boldsymbol{x}; \Lambda, \Theta)$ is given by

$$l(\Lambda, \Theta) = \frac{1}{2}(\boldsymbol{y} + \Lambda^{-1}\Theta^T\boldsymbol{x})^T \Lambda (\boldsymbol{y} + \Lambda^{-1}\Theta^T\boldsymbol{x}) - \frac{1}{2}\log|\Lambda|, \tag{2.2}$$

where $|\Lambda|$ is the determinant of Λ.

Let $X \in \mathbb{R}^{m \times n}$ and $Y \in \mathbb{R}^{m \times p}$ denote the input variables and the output variables of m graph instances, respectively. The negative log-likelihood function over m graph instances can be expressed as

$$\begin{aligned} L &= \frac{1}{2m}\sum_{i=1}^{m} \log P(Y_{i:}|X_{i:}) \\ &= \frac{1}{2}\{-\log|\Lambda| + tr(S_{yy}\Lambda + 2S_{yx}\Theta + \Lambda^{-1}\Theta^T S_{xx}\Theta)\}, \end{aligned} \tag{2.3}$$

where $S_{yy} = \frac{1}{m}Y^TY$, $S_{yx} = \frac{1}{m}Y^TX$ and $S_{xx} = \frac{1}{m}X^TX$ correspond to empirical covariance terms. Given m training graph instances, the objective is to find Θ and Λ that minimize negative log-likelihood (2.2). Sparse GCRF (SGCRF) assumes that both Λ and Θ are sparse, therefore, the objective function is defined as [13]:

$$\underset{\Lambda,\Theta}{\operatorname{argmin}}\, 2L(\Lambda, \Theta) + \lambda(\|\Lambda\|_{1,*} + \|\Theta\|_1), \tag{2.4}$$

where $\|\Lambda\|_{1,*}$ and $\|\Theta\|_1$ are $\ell1$ norms over off-diagonal elements of Λ, and $\ell1$ norms of Θ, respectively. The problem (2.4) is a constrained nonsmooth convex optimization due to the sparsity of parameters and positive definiteness constraint over Λ. Many solutions have been proposed to solve this problem [10, 13, 14]. In case of SGCRF, Newton coordinate descent was used[2, 13].

3 Representation Learning based Structured Regression

In real life applications, raw features might not be linearly dependent with output variables. As SGCRF can only model linear dependencies, the structure among data is possibly not revealed correctly. To improve the representational power of SGCRF, we propose a novel iterative approach called Representation Learning based Structured Regression (RLSR) that is able to learn hidden representation of inputs, and structure among outputs *simultaneously.* This approach is motivated of two aspects: First, by considering structure information, representation learning can learn more predictive input features than raw features. Second, given more informative representations, structure learning can reveal intrinsic structure among data.

3.1 RLSR Model Our proposed approach can model nonlinear dependency. To achieve this, we introduce hidden variables $\boldsymbol{h} \in \mathbb{R}^q$, which are learned based on input variables, and model the conditional probability of outputs given *learned hidden variables* as a multivariate Gaussian distribution. The pdf of RLSR is given by

$$P(\boldsymbol{y}|\boldsymbol{h}, \Lambda, \Theta) = \frac{1}{Z(\boldsymbol{h})} \exp(-\boldsymbol{y}^T \Lambda \boldsymbol{y} - 2\boldsymbol{h}^T \Theta \boldsymbol{y}), \tag{3.5}$$

where $\boldsymbol{h}$ is a nonlinear mapping of $\boldsymbol{x}$. The mapping is formulated as a parametric function

$$\boldsymbol{h} = f(\boldsymbol{x}, \boldsymbol{w}), \tag{3.6}$$

where $\boldsymbol{w}$ is a vector of parameters of f. Note that $\Lambda \in \mathbb{R}^{p \times p}$ still models the dependency among outputs $\boldsymbol{y}$, but $\Theta \in \mathbb{R}^{q \times p}$ models the dependency of outputs $\boldsymbol{y}$ on the new representation $\boldsymbol{h}$. A general illustration of the proposed model on a single graph instance is presented in Figure 1a, where each of q hidden variables is learned based on n input variables, and p output variables are dependent on hidden variables.

Given m graph instances, the negative log-likelihood of the model defined in (3.5) and (3.6) is

$$L = \frac{1}{2}\{-\log|\Lambda| + tr(S_{yy}\Lambda + 2S_{yh}\Theta + \Lambda^{-1}\Theta^T S_{hh}\Theta)\}, \tag{3.7}$$

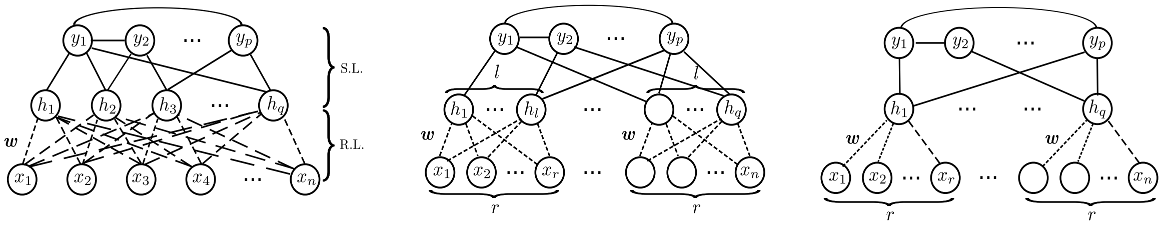


Figure 1: Illustration of RLSR model on a single graph instance with (a)$\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{h} \in \mathbb{R}^q$ and $\boldsymbol{y} \in \mathbb{R}^p$, (b) optimized architecture: $r = n/p$ and $l = q/p$, (c) optimized architecture: $r = n/p$ and $q = p$.

where $S_{yy} = \frac{1}{m}Y^TY$, $S_{yh} = \frac{1}{m}Y^TH$, $S_{hh} = \frac{1}{m}H^TH$ and $H \in \mathbb{R}^{m\times q}$ denotes the learned hidden variables of m graph instances, whose ith row are values of hidden variables $\boldsymbol{h}$ of the ith graph instance. Then the expectation and covariance of $Y|H$ becomes $-\Lambda^{-1}\Theta^TH^T$ and Λ^{-1}, respectively. Therefore, the optimization problem is to find $\boldsymbol{w}$, Λ and Θ that minimize negative log-likelihood

$$\underset{\boldsymbol{w},\Lambda,\Theta}{\operatorname{argmin}}\ 2L(\boldsymbol{w},\Lambda,\Theta) + \lambda(\|\Lambda\|_{1,*} + \|\Theta\|_1), \tag{3.8}$$

where we added a sparsity-inducing regularization term.

Note that the optimization function (3.8) is convex with respect to Λ and Θ, but not convex with respect to $\boldsymbol{w}$, because H is a nonlinear function of X. A natural approach to solve this optimization problem is to alternately update a subset of parameters. In the proposed RLSR model, $\boldsymbol{w}$ is learned by fixing Λ, Θ, then Λ, Θ are learned by fixing $\boldsymbol{w}$, and the process is repeated until termination(Algorithm 1). The optimization problem L_r in (3.9), is obtained by removing constant terms (w.r.t $\boldsymbol{w}$) from (3.7). Minimizing L_s in (3.10) is equivalent to minimizing (3.8) while fixing $\boldsymbol{w}$.

Algorithm 1 RLSR

```
Input: $X \in \mathbb{R}^{m\times n}$, $Y \in \mathbb{R}^{m\times p}$, $\lambda \in \mathbb{R}$.
Output: $\boldsymbol{w}$, $\Lambda$ and $\Theta$
1: Initialization: $\boldsymbol{w}_0$, $\Lambda_0$ and $\Theta_0$   ▷ See Section 3.5.1
2: do
3:     $t = t + 1$
4:     Representation Learning:   ▷ See Section 3.2
       (3.9)
         $\boldsymbol{w}_t = \underset{\boldsymbol{w}}{\operatorname{argmin}} L_r$
         $L_r = \frac{1}{m}tr(2Y^TH\Theta_{t-1} + \Lambda_{t-1}^{-1}\Theta_{t-1}^TH^TH\Theta_{t-1})$
         (where $H = f(X, \boldsymbol{w}_{t-1})$)
5:     Structure Learning:   ▷ See Section 3.3
       (3.10)
         $(\Lambda_t, \Theta_t) = \underset{\Lambda,\Theta}{\operatorname{argmin}} L_s = L(\boldsymbol{w}_t, \Lambda, \Theta) + \lambda(|\Lambda| + |\Theta|)$
6: while not (stopping criteria)   ▷ See Section 3.5.2
```

3.2 Representation Learning The representation learning phase(R.L. in Figure 1a). The nonlinear mapping function (3.6) is modeled using feedforward neural network (NN). Since there may be multiple hidden layers in feedforward neural network, the indirect connections in Figure 1a are represented as dashed lines. $\boldsymbol{w}$ refers to the parameters of neural network. The gradient of (3.9) with respect to $\boldsymbol{w}$ is

$$\begin{aligned} \frac{\partial L_r}{\partial \boldsymbol{w}} &= \frac{\partial L_r}{\partial H}\frac{\partial H}{\partial \boldsymbol{w}} \\ &= \frac{1}{m}(2Y\Theta^T + 2H\Theta\Lambda^{-1}\Theta^T)\frac{\partial H}{\partial \boldsymbol{w}}, \end{aligned} \tag{3.11}$$

where the derivative $\frac{\partial H}{\partial \boldsymbol{w}}$ depends on the structure of NN and is calculated using backpropagation. Since the optimization of NN is not convex, we use stochastic gradient descent as the learning algorithm.

3.3 Structure Learning The structure learning phase (S.L. in Figure 1a), which aims to learn the sparse dependency (Θ) between $\boldsymbol{h}$ and $\boldsymbol{y}$, and sparse structure (Λ) among $\boldsymbol{y}$. Those sparse dependencies are represented as solid connections in Figure 1a. The optimization problem of structure learning L_s, which is formulated as a $\ell 1$-regularized quadratic programming problem (3.10), is similar to SGCRF. Hence, we adopt Newton coordinate descent in [13] for solving (3.10). The gradients with respect to Λ and Θ are given by

$$\frac{\partial L_s}{\partial \Lambda} = Y^TY - \Lambda^{-1}\Theta^TH^TH\Theta\Lambda^{-1} - \Lambda^{-1}, \tag{3.12}$$

$$\frac{\partial L_s}{\partial \Theta} = 2H^TY + 2H^TH\Theta\Lambda^{-1}, \tag{3.13}$$

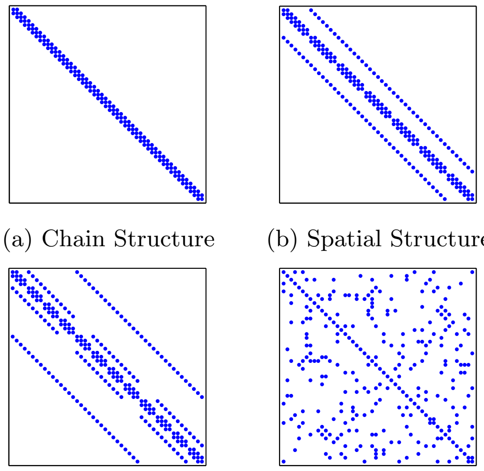
3.4 Optimized Architecture in Representation Learning When the number of input variables is large, the proposed model in Figure 1a will be less efficient. In particular, the number of parameters in $\boldsymbol{w}$ is $k(n+q)$ assuming there is a hidden layer in neural network with k hidden neurons between $\boldsymbol{x}$ and $\boldsymbol{h}$. As a result, the model could become unstable, easy to overfit and computationally costly. In order to resolve this issue, we propose to modify the architecture in representation learning procedure, such that the number of weights is reduced. In practice, each output variable is explicitly associated with its own $r = n/p$ input variables. So, we propose to reserve $l = q/p$ hidden variables for each such group of inputs. Two typical cases of proposed architecture are presented in Figure 1b and Figure 1c.

In Figure 1b, there are r input variables corresponding to each output variable y. Therefore, $n = p \times r$ and $q = p \times l$. The number of parameters in representation learning is reduced to $\frac{k}{p}(r + l)p = \frac{k}{p}(n + q)$ assuming there is a hidden layer in each neural network with k/p hidden neurons. The architecture presented in Figure 1c is a special case of the one presented in Figure 1b, where there is only one hidden variable per output. Hence $l = 1$, $q = p$ and the number of parameters in representation learning drops to $\frac{k}{p}(n+p)$. We note that the dimensionality of Θ in this case becomes $p \times p$. Furthermore, by sharing structure, all data instances are used to train the same neural network, which leads to a more robust representation.

3.5 Implementation Details

3.5.1 Initialization Since the objective in representation learning is to minimize negative log-likelihood (3.9), neural network can handle arbitrary dimension of output. If there is only one output, then W_0 is initialized by learning neural network through minimizing mean square error. Λ_0 and Θ_0 are initialized using the estimated Λ and Θ from SGCRF by using H_0 as input. This typical case is illustrated in Figure 1c, and it is used as the setting in all of our experiments. It can be easily extended to initialize the case with multiple output, H_0 is initialized using the hidden neurons of neural network, then Λ_0 and Θ_0 can be initialized by using H_0 as input of SGCRF. Figure 1b presents the corresponding architecture.

3.5.2 Stopping Criteria Since (3.9) is not convex, RLSR is not guaranteed to converge to a global optimum, hence, a part of the data is used for validation and the model stops when the accuracy on the validation data does not improve in the consecutive K iterations. After stopping, the estimated parameters of the current iteration are chosen as the optimized parameters of RLSR. In case the stopping criteria is not met, RLSR stops when the maximum number of iterations T is reached. In our implementation, we set $K = 10$ and $T = 30$. In addition, Newton coordinate descent converges when subgradient is less than a small tolerance $1e-6$.


(a) Chain Structure (b) Spatial Structure

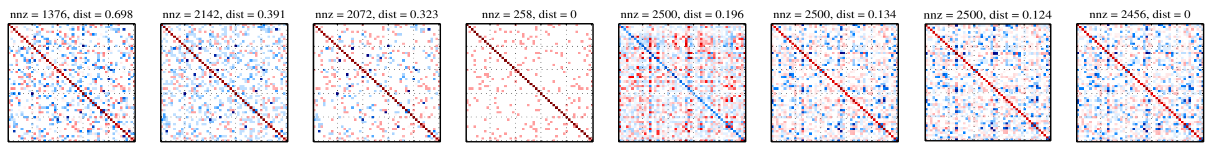
(c) ST Structure (d) Random Structure

Figure 2: Four different structures of precision matrix Λ.

4 Synthetic Data Experiments

4.1 Data Generation We generated data with 4 different structures and 2 different relationships between X and representation H, resulting in 8 different datasets. The 4 versions of $\Lambda \in \mathbb{R}^{p\times p}$ included chain structure, spatial (grid) structure, spatiotemporal (cube) structure, and random structure (see Figure 2). Each output variable y corresponds to a node. For chain structure, there are $p = 50$ nodes in each graph instance. The spatial structure is generated as 7×7 grid, with a total of $p = 49$ nodes. Each node is connected to its 4 nearest neighbours. For spatiotemporal structure, we assume a 4×4 grid is observed in 3 consecutive timestamps, which resulted in $p = 48$ nodes. In addition to being connected to its 4 nearest spatial neighbour nodes, each node was also connected to its two temporal neighbours. For random structure, there are $p = 50$ nodes in each graph instance. In order to ensure sparsity of Λ, each entry of Λ was generated as 1 with probability 0.1, and as 0 with probability 0.9. To guarantee positive definiteness, we performed a line search to increase the values of diagonal entries until Λ became diagonally dominant.

For each structure, $\Theta \in \mathbb{R}^{q\times p}$ was generated as a diagonally dominant sparse matrix. For each column of Θ, each entry was generated as 0.2 with probability 0.1, and 0 otherwise. The diagonal entries of Θ were generated as 1. We used this setting due to the fact that each h always contributes the most in predicting corresponding y, which reflects many real data. $X \in \mathbb{R}^{m\times n}$ was generated as a uniform distribution from -1 to 1, where $m = 1600$. Y was generated according to



(a) Λ-2 (b) Λ-3 (c) Λ-4 (d) Λ-5 (e) $\Theta\Lambda^{-1}$-2 (f) $\Theta\Lambda^{-1}$-3 (g) $\Theta\Lambda^{-1}$-4 (h) $\Theta\Lambda^{-1}$-5

Figure 3: Left (Right)4 figures are the learned Λ ($\Theta\Lambda^{-1}$) from four models on 'R-Non' dataset, respectively. *dist* is the distance between the estimated Λ and the ground truth Λ. 2: SGCRF. 3:NN+SGCRF. 4: RLSR. 5:GT

Table 1: Evaluation of RLSR versus 3 alternatives on 8 synthetic data.

Datasets	**C-Lin**	**C-Non**	**S-Lin**	**S-Non**	**ST-Lin**	**ST-Non**	**R-Lin**	**R-Non**
NN	0.6949	0.9484	2.9373	19.1401	1.9242	9.1369	0.8961	2.8797
SGCRF	8.734	3.3549	2.4648	27.6782	1.5427	12.7278	1.8053	6.1107
NN+SGCRF	0.2477	0.3088	0.4429	0.8312	0.3594	0.5615	0.3053	0.3995
RLSR	**0.2308**	**0.2305**	**0.4395**	**0.6603**	**0.3365**	**0.4381**	**0.2867**	**0.2746**
GT	0.2181	0.2192	0.4044	0.411	0.3034	0.2984	0.2624	0.254

a multivariate Gaussian distribution (3.5). In synthetic data, we assumed $n = q$ for simplicity of presentation. Regarding the dependency between X and H, it was generated either as a linear relationship $H = 5X + 5$, or as nonlinear relationship $H = 10\sin(5X)$. The first row of Table 1 lists the resulting 8 datasets. 'Lin' and 'Non' refer to linear and nonlinear relationship between X and H, respectively. C stands for chain structure, S stands for spatial structure, ST stands for spatiotemporal structure, and R stands for random structure.

4.2 Effectiveness of RLSR To investigate the effectiveness of RLSR, we evaluated the model on all 8 datasets we generated and compared it with 3 baseline methods:

- Feedforward Neural Network (NN). NN is a competitive unstructured baseline. We trained a neural network to predict each output directly using its corresponding input.
- SGCRF [13]. A baseline for structured models, which can only model linear relationship between inputs and outputs.
- NN + SGCRF. Similar to [12], in this baseline, NN is first trained in a supervised way to learn a mapping from X to Y. The outputs of NN are treated as hidden variables and SGCRF is applied to learn relationship between the hidden and output variables.
- Ground Truth (GT). It is the generated expectation of $P(Y|H)$ and represents the optimal model.

In this experiment, we used 1,000 graph instances for training, 300 graph instances for validation and 300 graph instances for testing. The regularization parameter $\lambda = \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-2}, 1\}$ was optimized using the validation data for SGCRF and NN+SGCRF.

Mean square error (MSE) of all models is shown in Table 1. The results indicate that NN works worse in the dataset with complicated underlying structure. SGCRF was not able to handle nonlinear relationship between X and H, while NN+SGCRF was able to handle such relation, and outperformed SGCRF. However, RLSR outperformed all other baselines on all datasets. This observation reveals the fact that *learning simultaneously the representation and structure in RLSR is mutually beneficial*, and that it outperforms NN+GCRF, *where learning representation and structure are performed independently*. We also notice that RLSR performs close to GT predictions on many datasets, which indicates the robustness of RLSR in achieving good performance even though the optimization is non-convex.

4.3 Analysis of RLSR In addition to conducting regression, another important task of RLSR is structure learning. We analyzed the results of RLSR on the 'R-Non' dataset because it mimics real applications. The visualization of Λ of SGCRF, NN+SGCRF, RLSR, and GT are presented in Figure 3{(a),(b),(c) and (d)}.

It is clear that the estimated Λ from SGCRF, NN+SGCRF, and RLSR exhibited the pattern of true Λ, however, they are estimated with different levels of noise. This is also noticed when computing the Euclidean distance *dist* between the estimated Λ and the true Λ. The estimated Λ from SGCRF contains more noise ($dist = 0.698$) than other baselines because it uses the raw features as input, while other baselines use estimated representation as input. The estimated Λ of RLSR has less noise ($dist = 0.323$) than Λ estimated from NN+SGCRF ($dist = 0.391$). *This is because the representation of RLSR is learned along with structure learning. In contrast, in NN+SGCRF, structure is not taken into consideration in learning 'representation'.*

Figure 3{(e),(f),(g),(h)} shows the dependency between X or H and Y. As we can see, the dependency matrix of SGCRF which uses the raw input X is far ($dist = 0.196$) from the true dependency matrix. Both NN+SGCRF and RLSR present more precise pattern, however, the estimated $\Theta\Lambda^{-1}$ from RLSR is closer ($dist = 0.124$) to the true one.

5 Real Data Experiments

We evaluated RLSR and the baseline models on 3 real-world challenging datasets: forecasting wind power for multiple farms, forecasting solar energy, and forecasting precipitation for multiple locations in U.S.

In all 3 datasets, the task is to forecast the target variable in the future, therefore, *k-fold cross validation* is no longer a suitable experimental setting because it is not reasonable to use future data to predict the past. Hence, we applied the following settings. We consider a window with $r + v + t$ graphs, where we train models on the first r graphs using the following v graphs for validation. Then, we forecast the final t graphs using the tuned model. Afterwards, we shift the window forward by t graphs and repeat the same process on the new window. Therefore, with m graphs in total, we are able to evaluate models on $\left\lfloor \frac{m-(r+v)}{t} \right\rfloor$ windows.

5.1 Wind Power Forecasting Wind power data is obtained from the Global Energy Forecasting 2012 competition[1]. The task is to hourly predict wind power at 7 nearby wind farms for the next 48-hour period. Each farm has 4 features (zonal and meridional wind components, wind speed, and wind direction). The data is available for 1080 days (36 months) from 2009/07 to 2012/06. We used 48 hours of data to generate a single graph, resulting in 540 graphs. Each graph has $4 * 7 * 48 = 1344$ features and $7 * 48 = 336$ outputs ($X \in \mathbb{R}^{540\times1344}$ and $Y \in \mathbb{R}^{540\times336}$). Missing values in Y are imputed using the average of the non-missing values from the same farm in the same or neighbor day.

In wind power forecasting, we compared proposed model with NN, SGCRF, NN+SGCRF and RBF+SGCRF, which is used in [12] on same dataset. For the NN+SGCRF model, we tuned λ on each window with values taken from $[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1]$. The best λ is chosen as the regularization parameter for all other baselines. For RBF+SGCRF, we used 10 RBF functions as suggested in [12] on same data.

[1]https://www.kaggle.com/c/GEF2012-wind-forecasting

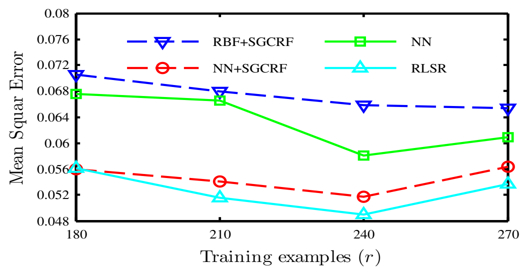


Figure 4: Generalization performance for wind data.

First, we evaluated the effect of varying training sizes on the performance of all models. In order to have fair comparison among all models with different training sizes, we created 8 windows, each of which is created by setting $v = 30, t = 30$. We consider 4 different training sample sizes $r = \{180, 210, 240, 270\}$, Under this setting, each chunk of test data contains wind energy record of 60 days. In this way, all models with different training sizes will use exactly the same validation and will be evaluated on exactly the same test data. The results are shown in Figure 4. It is evident that by increasing the training data up to 240 graph instances, the model has better MSE. However, the MSE increased when $r = 270$ due to noise from past 1.5 years when predicting 2 months. As NN is subject to overfiting with noise, both NN+SGCRF and RLSR are effected. Error of RBF+SGCRF has not increased when learning from 270 graph instances, but it was still the least accurate of four methods compared at Figure 4. Due to the much larger error of SGCRF, we removed the performances of SGCRF for Figure 4 for the ease of presentation.

In the subsequent experiments, we compare all models using 16 months (240 graphs) for training. Table 2 shows the comparison between our model and the baseline models. In addition, we compared our proposed model with RBF + SGCRF, which is used for forecasting wind power [12]. Raw inputs are mapped to a new space using Radial Basis Functions and SGCRF is directly trained with the new inputs. As shown in Table 2, RLSR outperforms SGCRF NN + SGCRF, NN, and RBF + SGCRF by 47 times, 6.1%, 18.3%, and 34.7%, respectively.

Finally, we compared Λ and Θ learned by all of the baseline methods in window 7 (where RLSR wins the most) as shown in Figure 5 and Figure 6, respectively. It is clear that Λ are learned similarly in all of the other 3 models except in SGCRF model. SGCRF barely learn right Λ or Θ, which results in its poor performance. We noticed that RLSR learns denser Θ than NN + SGCRF, although they used the same λ value. Our conclusion is that RLSR not only learns better representation, but also reveals underlying dependency more precisely.

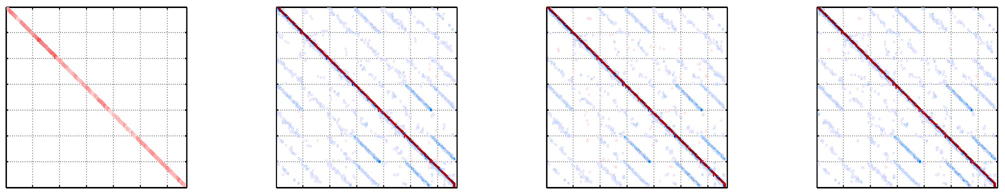

(a) SGCRF (b) RBF+SGCRF (c) NN + SGCRF (d) RLSR

Figure 5: Λ learned by all baselines on window 7 of the wind data.

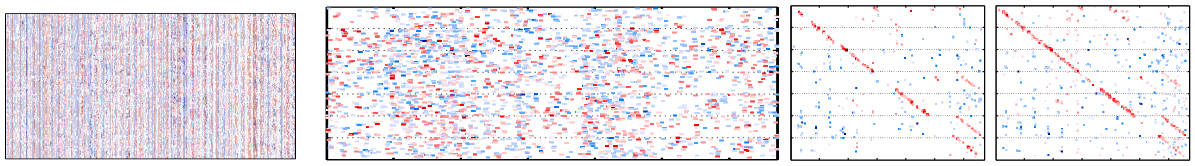

(a) SGCRF (b) RBF+SGCRF (c) NN + SGCRF (d) RLSR

Figure 6: Θ learned by all baselines on window 7 of the wind data.

Table 2: MSE of NN, NN + SGCRF, RLSR on wind data for all 8 windows.

	Win1	**Win2**	**Win3**	**Win4**	**Win5**	**Win6**	**Win7**	**Win8**	*mean* $\pm$ *std*
SGCRF	2.188	2.99	2.159	1.726	1.787	2.256	3.579	2.456	2.393 ± 0.6204
RBF + SGCRF	0.067	0.060	0.042	0.058	0.086	0.072	0.075	0.067	0.066 ± 0.013
NN	0.074	0.061	0.040	**0.041**	0.057	0.059	0.079	0.054	0.058 ± 0.014
NN + SGCRF	**0.064**	0.055	**0.031**	0.045	0.052	**0.058**	0.063	0.046	0.052 ± 0.011
RLSR	**0.065**	**0.044**	**0.032**	**0.042**	**0.050**	0.063	**0.056**	**0.039**	0.049 ± 0.012

5.2 Solar Energy Forecasting The task is to predict the daily solar energy income at 98 Oklahoma Mesonet sites. The features come from the NOAA/ESRL Global Ensemble Forecast System (GEFS) Reforecast Version 2, collected over 144 sites in United States. Both the Mesonet data and GEFS data are available every day for 14 years from 1994 – 2007. The data for this task is available on the website of AMS 2013-2014 Solar Energy Prediction Contest[2]. The data we used for experiments contain $X \in \mathbb{R}^{5110\times 7350}, Y \in \mathbb{R}^{5110\times 98}$. Missing Y are imputed using the average of non-missing data of Y from same days over all years. Also, the values of both X and Y of raw data vary a lot in a large range. We did logarithm smoothing over the values of X and Y.

We observed seasonal patterns with solar energy income within each year as shown in Figure 7. ($\boldsymbol{y}$ of the same day of different years have similar values.) Therefore, in our experiment, graphs from January to April belong to the first season, graphs from May to August belong to the second season and graphs from September to December belong to the third season. In such a partitioning all peak values were in the second season. And three seasons have 1680, 1708, 1722 graphs, respectively. We created 8 windows in each season by setting $r = 600, v = 120, t = 120$. We used the same settings for hyperparameter selection as in wind forecasting.

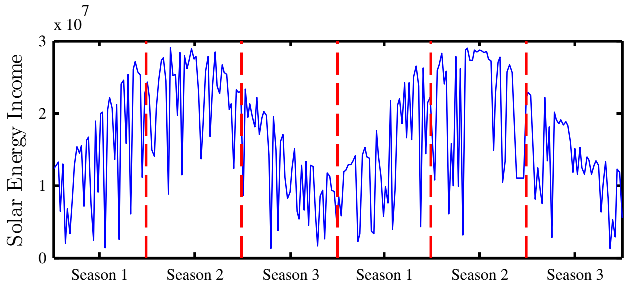


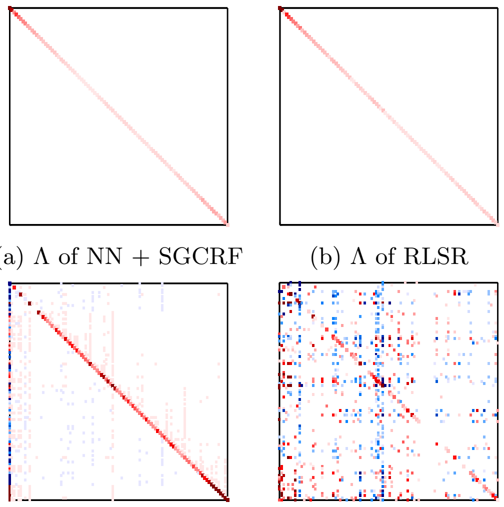
Figure 7: Seasonal patterns of solar energy income.

We report the *mean* and stand deviation of MSE on 8 windows in Table 3. Our proposed model outperforms all baselines by at least 50%. Since the raw input variable $\boldsymbol{x} \in \mathbb{R}^{7\times 350}$ has high dimension, we didn't get the result of SGCRF in 48 hours. The learned Λ and Θ of RLSR and NN + SGCRF are compared at Figure 8. There is little conditional dependency among $\boldsymbol{y}$ (Λ is diagonal), so the dependencies between $\boldsymbol{y}$ and $\boldsymbol{h}$ dominate. Therefore, when RLSR learns a better

[2]https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest

Table 3: *mean* and *std* of MSE on 8 windows on the solar energy forecasting application.

	Season 1	**Season 2**	**Season 3**
SGCRF	-	-	-
NN	0.078 ± 0.021	0.056 ± 0.017	0.024 ± 0.007
NN + SGCRF	0.074 ± 0.024	0.054 ± 0.018	0.024 ± 0.006
RLSR	$\mathbf{0.049 \pm 0.015}$	$\mathbf{0.034 \pm 0.007}$	$\mathbf{0.016 \pm 0.002}$



(a) Λ of NN + SGCRF (b) Λ of RLSR

(c) Θ of NN + SGCRF (d) Θ of RLSR

Figure 8: Λ and Θ learned by NN + SGCRF and RLSR on window 7 season 3 of the energy data.

representation and dependencies than NN + SGCRF, predictions of $\boldsymbol{y}$ become much more accurate.

5.3 Precipitation Forecasting

The task is to forecast daily precipitation across multiple locations based on several climatological features. We used the ground-based precipitation data from 1948 to 2007, collected on 1,218 sites over U.S. It is downloaded from NOAA's National Climate Data Center (NCDC) website[3]. For the climate features, we used the product from the NCEP/NCAR Reanalysis 1 project [6] in the same duration. It provides 6 climate indicators for 124 sites across the U.S. everyday, which are omega (Lagrangian tendency of air pressure), precipitable water, relative humidity, temperature, u-wind, and w-wind (zonal and meridional components of the wind). In order to collocate the climate indicators and the precipitation data, we took out only 124 sites, such that for each site, both precipitation and indicators are available. We also incorporated the following 3 geographical features for each site: latitude, longitude and the distance between the climate features and precipitation. After aggregating monthly data from the daily recordings, we have 124 nodes on each graph, i.e, $Y \in \mathbb{R}^{708\times 124}$, $X \in \mathbb{R}^{708\times 1116}$. Missing values in Y are imputed using the average of the non-missing data from same month over all years.

Our assumption is that precipitation is controlled by seasonal effects, which suggests training models for Spring, Summer, Fall and Winter separately. The number of graphs for each season is 177. We were also able to create 10 windows for each season, by setting $r = 90, v = 6, t = 6$. In total, we tested precipitation forecasting on 60 graphs for each season. The results are shown in Table 4.

The last column of Table 4 shows the performance on annual rain forecasting where 10 windows were created, with $r = 360, v = 24, t = 24$, such that we tested 240 graphs in total. For model learning and hyperparameter tuning, we used the same settings as in other real data experiments. Our proposed RLSR mdoel outperformed NN + SGCRF by as much as 6.2% and was much more accurate than SGCRF.

The estimated Λ of RLSR of yearly precipitation data is presented in Figure 9a, which exhibits obvious spatial dependency among 124 stations over U.S. As Λ is the precision matrix, each nonzero blue entry Λ_{ij} indicates the conditional dependency between the ith station and the jth station. The sparse structure of learned Λ is shown at Figures 9b,9c and 9d as red edges on the U.S. map for low, medium and high thresholds on the values of Λ, respectively. As we can see, the connections in the northern U.S. present for increased threshold. This implies the meteorological similarity across certain regions of North America.

6 Conclusion

In this paper, we proposed the RLSR model for joint learning of representation and structure in sparse regression. The experiments on synthetic data provide evidence that joint learning is mutually beneficial for discovering a more predictive representation and structure. This is further confirmed by experiments on challenging real-world applications which demonstrated the effectiveness of our proposed model.

[3]http://www.ncdc.noaa.gov/

Table 4: *mean* and *std* of MSE on 10 windows of seasonal rain forecasting application.

	Spring	**Summer**	**Fall**	**Winter**	**Year**
SGCRF	65.289 ± 85.192	154.717 ± 114.962	> 1000	> 1000	50.952 ± 159.882
NN	0.325 ± 0.047	0.417 ± 0.057	0.399 ± 0.042	0.376 ± 0.038	0.292 ± 0.025
NN + SGCRF	0.189 ± 0.017	0.213 ± 0.013	0.391 ± 0.514	0.161 ± 0.033	0.200 ± 0.025
RLSR	$\mathbf{0.178 \pm 0.012}$	$\mathbf{0.209 \pm 0.021}$	$\mathbf{0.241 \pm 0.034}$	$\mathbf{0.159 \pm 0.024}$	$\mathbf{0.189 \pm 0.010}$

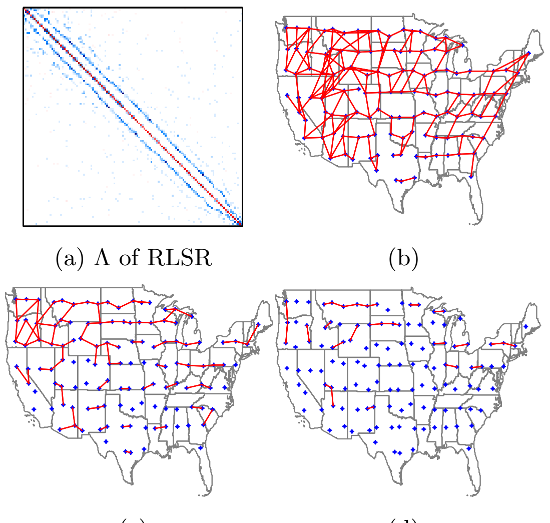

(a) Λ of RLSR (b) (c) (d)

Figure 9: Visualization of estimated Λ of yearly precipitation graph on continental U.S. Learned graph structure is shown for low, medium and high threshold Λ at panels (b), (c) and (d), respectively.

References

[1] T. Do and T. Arti. Neural conditional random fields. In *International Conference on Artificial Intelligence and Statistics*, pages 177–184, 2010.

[2] C.-J. Hsieh, I. S. Dhillon, P. K. Ravikumar, and M. A. Sustik. Sparse inverse covariance matrix estimation using quadratic approximation. In *Advances in Neural Information Processing Systems*, pages 2330–2338, 2011.

[3] J. Lafferty, A. McCallum, and F. C. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[4] J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: representation and clique selection. In *Proceedings of the twenty-first international conference on Machine learning*, page 64. ACM, 2004.

[5] L. Maaten, M. Welling, and L. K. Saul. Hidden-unit conditional random fields. In *International Conference on Artificial Intelligence and Statistics*, pages 479–488, 2011.

[6] J. Peng, L. Bo, and J. Xu. Conditional neural fields. In *Advances in neural information processing systems*, pages 1419–1427, 2009.

[7] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell. Hidden conditional random fields. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10):1848–1852, 2007.

[8] V. Radosavljevic, S. Vucetic, and Z. Obradovic. Continuous conditional random fields for regression in remote sensing. In *ECAI*, pages 809–814, 2010.

[9] V. Radosavljevic, S. Vucetic, and Z. Obradovic. Neural gaussian conditional random fields. In *Machine Learning and Knowledge Discovery in Databases*, pages 614–629. Springer, 2014.

[10] K.-A. Sohn and S. Kim. Joint estimation of structured sparsity and output structure in multiple-output regression via inverse-covariance regularization. In *International Conference on Artificial Intelligence and Statistics*, pages 1081–1089, 2012.

[11] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. In *Proceedings of the 22nd international conference on Machine learning*, pages 896–903. ACM, 2005.

[12] M. Wytock and J. Z. Kolter. Large-scale probabilistic forecasting in energy systems using sparse gaussian conditional random fields. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pages 1019–1024. IEEE, 2013.

[13] M. Wytock and Z. Kolter. Sparse gaussian conditional random fields: Algorithms, theory, and application to energy forecasting. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1265–1273, 2013.

[14] X.-T. Yuan and T. Zhang. Partial gaussian graphical model estimation. *Information Theory, IEEE Transactions on*, 60(3):1673–1687, 2014.