Structured Regression on Multiscale Networks

Jesse Glass and Zoran Obradovic, *Temple University*

Using multiscale networks, the exact solution for graph-based regression on networks of millions of nodes and trillions of links can be solved quickly, as demonstrated on a real-life health informatics application.

Conditional random fields (CRFs) are part of a broad functional framework that allows for the efficient interaction over graphical models using an assumption of conditional independence of feature functions. Still, the learning process for graphical models including CRFs remains a challenging task.[1] The challenge is simplified in Gaussian CRFs (GCRFs) by restricting feature functions to the set of all quadratic functions. This yields a CRF probability function that's transposable into a Gaussian multivariate probability function. Gaussian multivariate distributions are convex and can be optimized using the gradient descent algorithm.

A host of research has been done with GCRFs.[2–8] The models in general are faster than other structured regression approaches. Two advances that resulted in significant efficiency improvement have focused on speed and memory efficiency of the GCRF model: one is based on using a mean field approximation combined with fast filtering,[9] and the other uses a single network with linear bounds on convexity and fast calculation of gradients.[10] The work presented here handles multiscale networks far more efficiently than either of these two.

GCRFs work as a multiple output regression (we present them as applied to multivariate time-series data). The motivating task in our study is to predict monthly hospital admissions by disease for nearly 500 hospitals in the state of California by learning from millions of hospitalization records. The traditional approach for this large a dataset would be to model each disease separately at a specific hospital or in a network of hospitals. Although efficient, that approach isn't as accurate because it accounts for neither disease comorbidities nor hospital similarities. An alternative choice is a locally weighted regression model, which for this dataset takes the form of a vector autoregressive integrated moving average (VARIMA) model.[11] However, the traditionally efficient VARIMA model struggles with scalability compared to GCRF for multiscale

1541-1672/17/$33.00 © 2017 IEEE
Published by the IEEE Computer Society

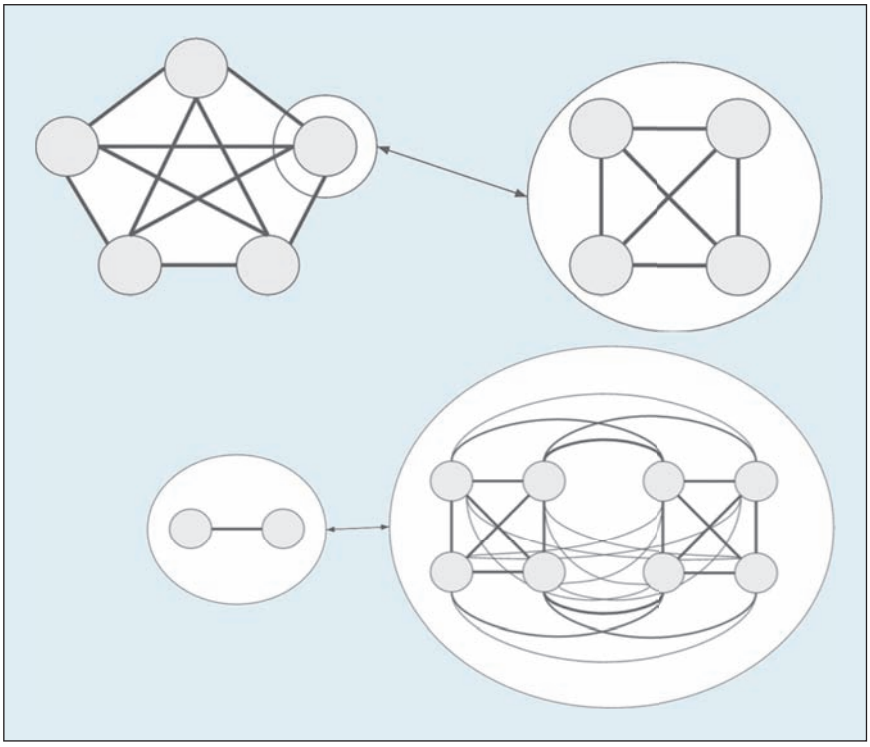

Figure 1. Network of disease similarities. The outer network (top left) is the hospitals in our example, and the inner network (top right) is located in a node of the outer network (diseases, in our example). A single link (bottom left) from the outer network shows a relationship between hospitals in our example; two joined inner networks (bottom right) represent a network of the relationships among all disease admission rates at two different hospitals.

networks as proposed here, which we call GCRF-MSN.

The input to a GCRF method is a similarity network and feature functions. Feature functions are the description of how targets, **y**, interact with observable data, **X**; a learner function is one that directly models how **x** determines *y*. In GCRF, learner functions are typically used as feature functions. When learner functions are input into another method, that method is an ensemble method. Learner functions are often referred to as input functions. It has been shown that an ensemble method that uses inaccurate input functions (often referred to as weak learners) can combine to produce a state-of-the-art prediction.[12,13] A similarity network is a unique collection of data that describes a relation between two or more outputs—for example, in the dataset used, the similarity among diseases is based on the number of symptoms that they share.

In the motivating example, we predict admissions by disease for various hospitals in California. Two networks were obtainable—one compares diseases to diseases, and the other compares hospitals to each other. Intuitively, we can visualize each hospital containing a network of disease nodes, with the diseases interacting with each other in and across hospitals. A network with networks inside nodes is a multiscale network, which can be modeled as a Kronecker product of networks. An example to illustrate our case is when we have a network of disease similarities (right side of Figure 1), and we also have a network of hospital similarities (left side of Figure 1). Each node on the left has a graph within it that looks like the graph on the right.

Structured regression methods are expected to achieve higher accuracy compared to unstructured regression methods, but their drawback is computational complexity. On problems similar to our motivating multiscale problem, GCRF-MSN computes faster than VAR. When using structured methods, we must choose from many network weight design choices including robust weight updating, sparse weight learning, or weights as prior information. GCRF^2 uses network weights as prior information, which is by far the fastest approach. The multiscale structured regression approach described here finds the optimal solution to the GCRF problem. When using multiscale networks, the proposed method can handle trillions of links in minutes while alternatives require weeks or months.

Background

In a CRF model, the observables, **X**, interact with each of the targets, *y*, directly and independently of one another. For a general network structure, the outputs, **y**, also have independent pairwise interaction functions. Thus, the CRF probability function takes the form

$$P(\boldsymbol{y}|\boldsymbol{X}) = \frac{1}{Z(\boldsymbol{X},\alpha,\beta)} \exp \left[\sum_{i=1}^{N} A(\alpha, y_i, \boldsymbol{X}) + \sum_{i\sim j} I(\beta, y_i, y_j) \right].$$

If we set the feature functions to be the quadratic difference between a function of observables, $\mathbf{f}(\mathbf{X})$, and targets, **y**, we produce a convex ensemble method:

$$A(\alpha, y_i, \boldsymbol{X}) = -\sum_{i=1}^{N}\sum_{k=1_k}^{K} \alpha_k \left(y_i - R_k(\boldsymbol{X})\right)^2.$$

When we incorporate quadratic pairwise interaction functions among

outputs, **y**, we produce a convex general graph structure ensemble method:

$$I\left(\beta, y_i, y_j\right) = -\sum_{l=1}^{L}\sum_{i\sim j_l} \beta_l S_{ij}^{l}\left(y_i - y_j\right)^2.$$

The GCRF model is a CRF model with both quadratic feature and quadratic interaction functions that can be transposed directly onto a Gaussian multivariate probability distribution:

$$P(\boldsymbol{y}|\boldsymbol{X}) = \frac{1}{\sqrt{2\pi\,|\Sigma|}}\exp\left[-\frac{1}{2}\left(y-\mu\right)^T Q\left(y-\mu\right)\right].$$

When setting these two conditional probability models equal to one another, the result is a precision matrix, Q, defined in terms of the confidence of our input predictors and the pairwise interaction structure, measured by α and β, respectively. Define L_l as the Laplacian matrix of pairwise interaction structure matrix S^l. The precision matrix is given by

$$Q = \sum_{k=1}^{K} a_k I + \sum_{l=1}^{L} \beta_l L_l.$$

Representing input predictions as a matrix, R, we can concisely write the formula for the final prediction:

$$\mu = Q^{-1} R\alpha.$$

The only remaining constraint is that Q is positive semidefinite, which is a bound on convexity but also a byproduct of the multivariate Gaussian assumption. As long as we satisfy the positive semidefinite constraint, we have a convex model and can optimize it using gradient descent. When first introduced,[2] GCRF used a diagonally dominant assumption to guarantee positive semidefiniteness, implying that all links (similarities) and all parameters were strictly non-negative. In another study,[10] the parameter constraints were relaxed because bounds on positive semidefiniteness were solved as a set of linear functions of the hidden parameters. The discussed method with faster optimization and fewer link weight and parameter restrictions was referred to as unimodal GCRF (UmGCRF) because it assumes at most one link between each node. UmGCRF operates on the precision matrix, Q, by diagonalizing the Laplacian matrix, L. Because L is a symmetric real valued matrix, $L = UDU^T$, where $UU^T = I$ and D is a diagonal matrix. We can then substitute this decomposed formula for L into the formula for Q. This restricts the hidden parameters for GCRF, α and β, to be operators in a diagonal matrix:

$$\begin{aligned} Q = \sum_{k=1}^{K}\alpha_k I + \beta L &= \sum_k \alpha_k I + \beta UDU^T \\ &= U\left(\sum_k \alpha_k U^T I U + \beta D\right)U^T \\ &= U\left(\sum_k \alpha_k I + \beta D\right)U^T. \end{aligned}$$

Using a traditional notation for a diagonalized matrix, we can write $Q = U\Lambda U^T$ and then define all combinations of α and β that produce feasible eigenvalues for Q. This formulation is efficient because calculating the partition function for GCRF was previously $O(n^3)$ and now is only $O(n)$. Because GCRF defines the hidden parameters α and β with respect to the precision matrix, we must invert the precision matrix for every update of α and β. Instead of inverting the entire precision matrix, which would take $O(n^3)$ operations, with the following equation we can compute the $\log(|\,Q^{-1}\,|) = \Sigma_{i=1}^{N}\log\left(\lambda_i^{-1}\right)$ in $O(n)$ operations:

$$\lambda_i = \sum_{k=1}^{K}\alpha_k + \beta d_i.$$

The first-order derivatives with respect to the partition function would also require inverting the precision matrix, except that we only need the trace of the inverse. Thus, we can use the eigenvalue update mentioned previously to reduce computation time. When GCRF was introduced, the first-order derivatives were solved in the form

$$\begin{aligned} \frac{\partial l}{\partial \alpha_k} &= -\frac{1}{2}\left(y^T y + 2\left(y-\mu\right)^T R_k + \mu^T\mu\right) \\ &\quad + \frac{1}{2}Tr(Q^{-1}) \\ \frac{\partial l}{\partial \beta} &= -\frac{1}{2}\left(y^T L y + \mu^T L\mu\right) + \frac{1}{2}Tr(Q^{-1}L). \end{aligned}$$

With a couple of preprocessing steps such as, $C = U^T R$, we can derive first-order derivatives that can be computed in linear time. Previously, inference was a component during each iteration of learning, which is an $O(n^2)$ process. It's also possible to remove inference from the learning process, which brings learning down to linear complexity. We use $\times$ to represent element-wise multiplication:

$$\begin{aligned} \frac{\partial l}{\partial \alpha_k} &= -\frac{1}{2}\left(c_1 + 2\left(C_k\ .\times \lambda^{-1}\right)C\alpha + \alpha^T C^T \Lambda^{-2} C\alpha\right) \\ &\quad -1^T\lambda^{-1} \\ \frac{\partial l}{\partial \beta} &= -\frac{1}{2}\left(c_2 + \alpha^T C^T\left(\left(d\ .\times \lambda^{-2}\right).\times C\right)\right) \\ &\quad + \frac{1}{2}d^T\lambda^{-1}. \end{aligned}$$

Using the established formula for eigenvalues of Q, positive semidefiniteness of Q is easily verified. If we know that all the eigenvalues of Q are greater than or equal to zero, then Q is positive semidefinite:

$$Q \succeq 0 \Leftrightarrow \begin{cases} \sum_{k=1}^{K}\alpha_k + \beta\ d_0 \geq 0 \\ \sum_{k=1}^{K}\alpha_k + \beta d_{n-1} \geq 0 \end{cases}.$$

These parameter boundaries allow the model to include negative links and remain positive semidefinite. The method proposed here improves UmGCRF's computation speeds on multiscale networks. Another approach for speeding up GCRF is an approximate method called FF-GCRF,[9] which uses a mean-field approximation for network interactions. However, it requires a network to be defined on a Euclidean space. The network in our motivating example is not, making this approach dependent on an embedding preprocessing step.

All the discussed implementations of GCRF use inputs the same way—specifically, they use input learners for feature functions and incorporate network weight as prior knowledge. We use a VARIMA model for the structured regression benchmark.[11]

GCRF-MSN Method

We introduce an implementation of GCRF that operates on nested network similarities, otherwise known as a Kronecker product of matrices. This method is much faster and requires less memory than standard approaches, and it provides a framework for incorporating various types of network information into a structured regression. To develop this model, we formally introduce two properties of Laplacians of Kronecker products currently absent from literature.

Despite the vast literature on Kronecker products and Laplacian matrices,[14–17] there isn't a reference that contains the formula for the Laplacian of the Kronecker products of matrices. To define the Laplacian of a Kronecker product, we use the following notation: similarity network, S, has a diagonal sum matrix, $D(S)$, each entry of which is denoted D. The Laplacian of S is $L(S)$, and the standard formula for the Laplacian is written as $L(S) = D(S) - S$. We then can formulate $d_i = D_{ii}$ as a summation of the entries of S, denoted s_{ij},

$$d_i = \sum_{\forall j} s_{i,j}.$$

Claim: The Laplacian of a Kronecker product is

$$L(S_1 \otimes S_2) = D(S_1) \otimes D(S_2) - S_1 \otimes S_2.$$

Proof: The Kronecker product can be concisely represented via block matrices. The Kronecker of diagonal matrices is particularly clean:

$$D(S_1) \otimes D(S_2) = \begin{bmatrix} d_1 D(S_2) & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & d_{n_1} D(S_2) \end{bmatrix}.$$

When we fully carry out the multiplication of the above block matrix, it's clear that the Kronecker product of diagonal matrices is diagonal, and the exact diagonals are easy to position using ceiling and modulo indexing:

$$= \begin{bmatrix} d_1^1 d_1^2 & 0 & \cdots\cdots & 0 & 0 \\ 0 & d_1^1 d_1^2 & \cdots\cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots\cdots & d_{n_1}^1 d_{n_2-1}^2 & 0 \end{bmatrix}.$$

The above diagonal matrix shows entries that are a product of the diagonal entries of smaller matrices, which are computed by summing across each row of the matrix. Now compute the Kronecker product of similarities, $S_1 \otimes S_2$, and calculate the diagonal of this matrix, verifying that it equals the above matrix:

$$S_1 \otimes S_2 = \begin{bmatrix} s_{1,1}^1 s_{1,1}^2 & s_{11}^1 s_{12}^2 & \cdots\cdots & s_{1,1}^1 s_{1,n_2-1}^2 & s_{1,1}^1 s_{1,n_2}^2 \\ s_{1,1}^1 s_{2,1}^2 & s_{11}^1 s_{22}^2 & \cdots\cdots & s_{1,1}^1 s_{2,n_2-1}^2 & s_{1,1}^1 s_{2,n_2}^2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ s_{n_1,1}^1 s_{n_2,1}^2 & s_{n_1,1}^1 s_{n_2,2}^2 & \cdots & s_{n_1,n_1}^1 s_{n_2,n_2-1}^2 & s_{n_1,n_1}^1 s_{n_2,n_2}^2 \end{bmatrix}.$$

If we use index k to represent positions in the matrices $S_1 \otimes S_2$ and $D(S_1 \otimes S_2)$, it isn't hard to verify

$$d_k = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} s^1_{i,\, ceil\left[\frac{k}{n_1}\right]} \cdot s^2_{j,\left[k \bmod n_1\right]}.$$

Because $s^1_{i,\, ceil[k/n_1]}$ is independent of j,

$$\begin{aligned} d_k &= \sum_{i=1}^{n_1} s^1_{i,\, ceil\left[\frac{k}{n_1}\right]} \sum_{j=1}^{n_2} s^2_{j,\left[k \bmod n_1\right]} \\ &= \sum_{i=1}^{n_1} s^1_{i,\, ceil\left[\frac{k}{n_1}\right]} d^2_{\left[k \bmod n_1\right]} \\ &= d^1_{\,ceil\left[\frac{k}{n_1}\right]} d^2_{\left[k \bmod n_1\right]}. \end{aligned}$$

This is the same as the Kronecker product of diagonals where k maps to the ordering resultant of Kronecker products.

Decomposition of a Regularized Laplacian

With this new formula for the Laplacian of the Kronecker of two adjacency matrices, we can formalize the Eigen decomposition of the Laplacian with respect to its input matrices. To do this, we use a regularized Laplacian of the form

$$L(S) = I - D(S)^{-\frac{1}{2}}\ S\ D(S)^{-\frac{1}{2}}.$$

It has specific bounds on the sum of its eigenvalues, and the rows and columns of the S matrix are now normalized. We denote the unnormalized network weight matrix, S_0, and we can obtain a regularized Laplacian by normalizing S_0 by performing the following operation:

$$S = D(S_0)^{-\frac{1}{2}}\ S_0\ D(S_0)^{-\frac{1}{2}}.$$

With our regularized Laplacian, we have a Laplacian in which the

eigenvectors are determined entirely by the eigenvectors of the input adjacency matrix, S:

$$L(S) = I - S = I - U\Lambda U^T = U(I - \Lambda)U^T.$$

Decomposition of the Kronecker Product of Matrices

Applying our definition of the Laplacian of a Kronecker to our normalized adjacency matrices, we get

$$L(S_1 \otimes S_2) = I_1 \otimes I_2 - S_1 \otimes S_2.$$

We can perform an Eigen decomposition on each adjacency matrix in relatively little time. It's then possible to exploit known properties of the Kronecker product of decomposed matrices, as seen below:

$$= I_1 \otimes I_2 - \left(U_1\Lambda_1 U_1^T\right) \otimes \left(U_2\Lambda_2 U_2^T\right)$$
$$= I_1 \otimes I_2 - \left(U_1 \otimes U_2\right)\left(\Lambda_1 \otimes \Lambda_2\right)\left(U_1 \otimes U_2\right)^T.$$

To get the formula written in a way that yields a computational speedup for GCRF, we must project the identity matrix onto the new orthonormal basis, which produces an identity matrix, thus

$$L(S_1 \otimes S_2) = (U_1 \otimes U_2)$$
$$(I_1 \otimes I_2 - \Lambda_1 \otimes \Lambda_2)\,(U_1 \otimes U_2)^T.$$

We've diagonalized the Laplacian of a Kronecker product of networks by operating on the component networks before the Kronecker product was taken. Diagonalizing in this manner is magnitudes more efficient than a naive approach. Revisiting GCRF's formulation for the precision matrix, Q, we can see

$$Q = \sum_{k=1k}^{K} I + \beta L\left(S_1 \otimes S_2\right)$$
$$= \left(U_1 \otimes U_2\right)\left(\sum_{k=1}^{K}\alpha_k I - \beta\Lambda_1 \otimes \Lambda_2\right)\left(U_1 \otimes U_2\right)^T.$$

GCRF provides a closed form solution for the partition function, but it's computed in $O(n^3)$ operations. In another study,[10] the GCRF learning process was reduced from $O(n^3 \times I)$ to $O(n^3 + n \cdot I)$. The remaining bottleneck can be addressed with GCRF-MSN, resulting in $O(h^3 + d^3 + n \cdot I)$ such that $n = h \cdot d$, where h represents the number of nodes in the outer graph (hospitals) and d the inner graph (diseases). The speedup varies according to the size of the subnetworks. The optimally efficient case of GCRF-MSN for two networks can be seen by minimizing $h^3 + d^3$ such that $n = h \cdot d$, which results in $h = d = \sqrt{n}$. Then, we can see that $h^3 + d^3 = 2\sqrt{n}^3 = 2n^{3/2}$, which implies the entire GCRF learning process would be $O(n^{3/2} + n \cdot I)$. If we nest more than two networks, the speedups are even more dramatic.

The current method is also more efficient at storing information. In our motivating example on Healthcare Cost and Utilization Project (HCUP) data, the network changes over time, which is often referred to as an evolving network. We end up with a unique network for every year spanning nine years. Predicting 250 diseases across 500 hospitals means predicting 100,000 different node values each month, meaning that each monthly network has 10 billion links. GCRF-MSN can optimize parameters while not storing every link—to do so, it needs only 312,500 links. What's more, we can capture the interaction between these networks without executing the Kronecker product. The space requirement of GCRF-MSN is only logarithmic compared to traditional GCRF or other network methods.

It was shown in another study[10] that

$$Q \succeq 0 \leftrightarrow \begin{cases} \sum_{k=1}^{K}\alpha_k + \beta\ d_0 \geq 0 \\ \sum_{k=1}^{K}\alpha_k + \beta d_{n-1} \geq 0 \end{cases}.$$

Here, we show that we can extend this definition to networks built on the combination of multiple scales of networks via the Kronecker product. The new equivalent set of constraints are given by the following lemma.

Lemma:

$$Q \succeq 0 \leftrightarrow \begin{cases} \sum_{k=1}^{K}\alpha_k + \beta\prod_{l=1}^{L} d_{l,0} \geq 0 \\ \sum_{k=1}^{K}\alpha_k + \beta d_{j,0}\prod_{l\neq j} d_{l,n-1} \geq 0 \quad \forall j. \\ \sum_{k=1}^{K}\alpha_k + \beta\prod_{l=1}^{L} d_{l,n-1} \geq 0 \end{cases}$$

Proof:

It was proven previously that only the smallest and largest eigenvalues must be greater than zero for all eigenvalues of Q to be guaranteed to be greater than zero.[10] We know the multiplication of ordered positive numbers maintains ordinal relationships. If all the links are positive, we know the diagonal of the Laplacian has all non-negative entries, in which case,

$$Max\left(\Lambda_{S_1 \otimes S_2 \ldots \otimes S_L}\right) = \prod_{l=1}^{L} Max\left(\Lambda_{S_l}\right)$$

and

$$Min\left(\Lambda_{S_1 \otimes S_2 \ldots \otimes S_L}\right) = \prod_{l=1}^{L} Min\left(\Lambda_{S_l}\right).$$

If some of the links are negative, there could be a negative diagonal entry in any one of Λ_{S_l}. With the reasonable assumption that the absolute value of the maximum eigenvalue is larger than the absolute value of the

Table 1. Equation segmenting.*

Identifier	Parameter sets	Data points per model
All	1	$h \cdot d \cdot t$
Disease	d	$h \cdot t$
Hospital	h	$d \cdot t$
Unique	$h \cdot d$	t

*h is the number of hospitals, d is the number of diseases, and t is the number of time steps.

minimum eigenvalue for each matrix S_l, we can exhaustively search all possible minimums relatively efficiently by multiplying the maximum of all the matrices excluding one. We then multiply that value by the minimum eigenvalue of the excluded matrix. Once cycled through all input matrices, we've produced all possible minimums

$$\left\{PossibleMin\left(\Lambda_{S_1 \otimes S_2 \ldots \otimes S_L}\right)\right\} = d_{j,0} \prod_{l \neq j} d_{l,n-1} \geq 0 \quad \forall j.$$

Because we can compute the precision matrix's eigenvalues as linear equations with respect to the Kronecker product of the diagonal matrices of the network structures, we can compute the maximum and minimum values of an eigenvalue and then have linear bounds on positive semidefiniteness.

This enables us to utilize improvements introduced elsewhere,[10] in which a more representationally powerful implementation of GCRF called UmGCRF was developed, which achieves greater accuracy in less time than the original implementation of GCRF.

Experimental Design and Baselines

For the first set of experiments, time trials were conducted in which the proposed GCRF-MSN was compared to three structured regression alternatives (GCRF, UmGCRF, and FF-GCRF). Then, we ran and evaluated a real-world data experiment in terms of execution time and mean-squared error (MSE). The task is to predict monthly admissions for each disease for each hospital in the state of California. The data comes from the California HCUP database, contains 35,844,800 inpatient discharge records collected over nine years, and uses the CCS disease coding schema.

Any datasets in which structured regression is applicable could alternatively use independent functions for each node. In our motivating dataset, this corresponds to giving each disease within each hospital its own time-series function. In Table 1, we label this scenario *unique*. If we use the following representations, where h is the number of hospitals, d is the number of diseases, and t is the number of time steps, then we can see that $n = h \cdot d \cdot t$. In the *unique* scenario, we have $h \cdot d$ equations and t data points per equation.

These various data segmentations are useful for understanding the differences between models. VARIMA and GCRF-MSN use a weighted mixture of *unique* and *all*. A VARIMA model with a network in which every weight was equal to one would be an ARIMA model in the *all* scenario. Because GCRF inputs unstructured predictions, we train regressions on all the above possible segmentations and then input those predictions into GCRF. This is interesting because GCRF can balance the gains of each. Some of the algorithms are very poor performers, but collectively they can produce a much more accurate prediction.

Autoregressive integrated moving average (ARIMA) models in particular have been favored by the research community due to their general applicability.[18] Only autoregressive (AR) was feasible due to well-known restrictions of the ARIMA model. When using AR or vector autoregressive (VAR), it's important to determine the appropriate lag, perhaps via a preliminary correlation analysis. Unsurprisingly, in a monthly time-step model, the observations 1 month and 12 months back were the most informative, so we ran our AR and VAR implementations with a lag of 12. We trained on months 13 through 80 and tested on months 81 through 107. We start on month 13 because we use a 12-month lag as our window size. Thus, $\boldsymbol{x}_i$ has 12 features for each y_i, which are the previous 12 values for y. The experimental setup was the same for every model, the only difference being that VAR and GCRF-MSN incorporate relational data that isn't input to unstructured methods. We applied a Z-score normalization to the data before applying any model. Both a neural network (NN) and VAR model were used for experimental comparison.

The structure used for VAR and GCRF-MSN boils down to n^2 pairwise relations, with each relation measured by a weight that's input to both methods as prior information. For the disease similarity network, we use symptom similarity scores[19] in which biomedical literature was parsed and binary indicators for correlated symptoms were generated. The relationship weight is calculated by taking the cosine similarities of each symptom vector. The disease network from another study[19] was built using MeSH terminology, but our dataset uses CCS codes, so we built a translation table by hand for CCS codes to MeSH terminology (http://astro.temple.edu~tud25892). The matching is not one to one: sometimes, we mapped several MeSH terms to a single CCS code. In these cases, we took the average of similarities.

Another prior used was a hospital similarity network. For this purpose, we used a hospital similarity network built in a previous study[8] to capture the overlap in specialization that each hospital has. Specifically, the specialization is represented by the distinct rate at which hospitals treated various diagnoses in the previous year. For instance, cardiovascular hospitals only treat a subset of diseases and therefore will have large similarity values with one another.

Results

We start by comparing the method proposed in this article to previous fast implementations of GCRF, as shown in Table 2. The relative speed of UmGCRF, FF-GCRF, and GCRF was investigated elsewhere,[10] but that analysis was done on a single static graph over two time steps. In the HCUP dataset, the network changes over time. The speed performance we examined is for optimizing evolving graphs with 100,000 nodes at each time step across 72 time points. Critically, this network can be segmented into two networks on different scales.

GCRF-MSN has a subquadratic preprocessing step, with linear learning time and inference. FF-GCRF has quadratic learning complexity, but within each iteration it requires another entire optimization procedure. FF-GCRF's gains are principally on Euclidean space where the process can be brought down to a linear cost with fast filtering techniques. As it's implemented, FF-GCRF requires a cubic preprocessing step to be run on datasets that have similarities that aren't defined as a distance on a Euclidean space. UmGCRF has a cubic preprocessing step that takes nearly a day per network, with nine networks, which means it takes more than a week. GCRF has cubic complexity at every iteration of gradient descent and takes months to complete learning on a dataset this size.

We used the four AR setups as our input learners for GCRF-MSN because they were all computable in around a minute or less. The comparison algorithms took hours (NN) or days (VAR), as shown in Table 3. In fact, the faster of two alternatives (NN) takes 36 times longer than GCRF-MSN and produces worse results. FF-GCRF is omitted because it requires that the network be defined in a Euclidean space, which this dataset is not. UmGCRF and GCRF-MSN produce the exact same answers except that GCRF-MSN is much faster, and the traditional implementation of GCRF is unfeasible for this dataset. The training set consisted of 68 months, and the testing was 27 months. The total number of testing points across all hospitals and diseases for the testing period is over 1.5 million. Results are averaged and standard error is reported.

GCRF-MSN used a combination of weak learners to outperform the next two most accurate regression methods. This is important because those weak learners weren't as accurate, but they were much more efficient to compute. VAR utilizes a network structure and learns a unique prediction function for each disease at each hospital, but this prediction function can closely be approximated when only using neighbors' prediction functions. The VAR model has particularly reasonable assumptions and nice imputation of missing functions if the network is well built. The time to compute VAR demonstrates the scaling difficulty of VAR as opposed to GCRF-MSN in this scenario. VAR is traditionally one of the most efficient structured approaches, taking $O(n^2 \cdot I)$ time. However, in the case where the structure can be represented as a Kronecker product, GCRF-MSN runs much faster.

Table 2. Algorithm runtime.

GCRF-MSN	UmGCRF	FF-GCRF	GCRF
10 min	1.1 weeks	1 week	2 months

Table 3. Runtime and accuracy (mean-squared error) on HCUP data.

Model	Time	Train MSE	Test MSE (standard error)*
NN	6.7 hrs	0.0214	0.0574 (3.71e-4)
VAR	166 hrs	0.0189	0.0548 (3.84e-4)
GCRF-MSN	10 min	0.0178	0.0531 (3.08e-4)

*Using a two-sided difference in means, there's a p-value <0.001 that these differences in error come from similar distributions.

The GCRF method is a representationally powerful structured regression algorithm and a framework that provides a closed-form solution for the partition function. Recent speedups for the GCRF method have allowed for exact solutions for networks of up to 100,000 nodes and 10 million links.[10] This solution also enabled negative link weight interactions, which has resulted in increased accuracy in various experiments.

Multiscaled networks are a naturally occurring type of information, as we presented here. Such structures have exploitable properties that enable them to scale up to millions of nodes and trillions of links. To use these special properties, we had to define the Laplacian of the Kronecker product of matrices, which is surprisingly absent from the literature covering both Laplacians and Kronecker products, even though many sources state that the Kronecker product of Laplacian matrices is not itself a Laplacian. By applying our findings to diagonalize a regularized Laplacian,

we uncover a self-affine parameterization of the eigenvalues of the precision matrix for the GCRF model, as was done for UmGCRF. GCRF and GCRF-MSN outperform state-of-the-art VARIMA models in terms of accuracy, and GCRF-MSN can solve systems much faster than the VARIMA model. ⬒

THE AUTHORS

Jesse Glass is a PhD student at Temple University. His research interests include probabilistic graphical models and Bayesian methods. Contact him at jglassemc2@gmail.com.

Zoran Obradovic is an academician at the Academia Europaea, a foreign academician at the Serbian Academy of Sciences and Arts, and an L.H. Carnell Professor of Data Analytics at Temple University, where he's also professor in the Department of Computer and Information Sciences with a secondary appointment in statistics and director of the Center for Data Analytics and Biomedical Informatics. He's also the executive editor of the *Journal on Statistical Analysis and Data Mining*, the official publication of the American Statistical Association, and an editorial board member of 11 other journals. Contact him at zoran.obradovic@temple.edu.

Acknowledgments

This research was supported in part by DARPA grant FA9550-12-1-0406 negotiated by AFOSR, NSF BIGDATA grant 14476570, ONR grant N00014-15-1-2729, and SNSF joint research project (SCOPES), ID: IZ73Z0_152415. The Healthcare Cost and Utilization Project (HCUP), Agency for Healthcare Research and Quality, provided data used in this study.

References

1. C. Sutton and A. McCallum, "An Introduction to Conditional Random Fields," *Foundations and Trends in Machine Learning*, Now Publishers, 2011.
2. V. Radosavljevic, S. Vucetic, and Z. Obradovic, "Continuous Conditional Random Fields for Regression in Remote Sensing," *Proc. European Conf. Artificial Intelligence*, 2010, pp. 809–814.
3. H. Guo, "Modeling Short-Term Energy Load with Continuous Conditional Random Fields," *European Conf. Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2013, pp. 433–448.
4. J. Stojanovic et al., "Semi-supervised Learning for Structured Regression on Partially Observed Attributed Graphs," *Proc. 2015 SIAM Int'l Conf. Data Mining*, 2015; www.dabi.temple.edu/~zoran/papers/StojanovicSDM15.pdf.
5. D. Gligorijevic, J. Stojanovic, and Z. Obradovic, "Improving Confidence While Predicting Trends in Temporal Disease Networks," *Proc. 4th Workshop Data Mining for Medicine and Healthcare, SIAM Int'l Conf. Data Mining*, 2015; www.dabi.temple.edu/~zoran/papers/GligorijevicSDMDMMH15.pdf.
6. V. Radosavljevic, S. Vucetic, and Z. Obradovic, "Neural Gaussian Conditional Random Fields," *Proc. European Conf. Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2014; www.dabi.temple.edu/~zoran/papers/ngcrf_ecml14.pdf.
7. N. Djuric et al., "Gaussian Conditional Random Fields for Aggregation of Operational Aerosol Retrievals," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 4, 2015, pp. 761–765.
8. A. Polychronopoulou and Z. Obradovic, "Hospital Pricing Estimation by Gaussian Conditional Random Fields Based Regression on Graphs," *IEEE Int'l Conf. Bioinformatics and Biomedicine*, 2014, pp. 564–567.
9. K. Ristovski et al., "Continuous Conditional Random Fields for Efficient Regression in Large Fully Connected Graphs," *Proc. 27th AAAI Conf. Artificial Intelligence*, 2013; www.dabi.temple.edu/~zoran/papers/KostaAAAI13.pdf.
10. J. Glass et al., "Extending the Modeling Capacity of Gaussian Conditional Random Fields While Learning Faster," *Proc. 30th AAAI Conf. Artificial Intelligence*, 2016; www.dabi.temple.edu/~zoran/papers/jesseAAAI2016.pdf.
11. H. Lutkepohl, *New Introduction to Multiple Time Series Analysis*, Springer, 2007.
12. P. Buhlmann and T. Hothorn, "Boosting Algorithms: Regularization, Prediction, and Model Fitting," *Statistical Science*, vol. 22, no. 4, 2007, pp. 477–505.
13. H. Chipman, E. George, and R. McCulloch, "BART: Bayesian Additive Regression Trees," *Annals Applied Statistics*, vol. 4, no. 1, 2010, pp. 266–298.
14. L. Lovasz, *Eigenvalues of Graphs*, 2007; www.cs.elte.hu/~lovasz/eigenvals-x.pdf.
15. R. Merris, "Laplacian Graph Eigenvectors," *Linear Algebra and its Applications*, vol. 278, 1998, pp. 221–236.
16. P. Zumstein, "Comparison of Spectral Methods Through the Adjacency," *Matrix and Laplacian of a Graph*, 2005; www.yaroslavvb.com/papers/zumstein-comparison.pdf.
17. K. Schacke, *On the Kronecker Product*, Univ. Waterloo Reports, 2013.
18. K. Hipel and A. McLeod, *Time Series Modelling of Water Resources and Environmental Systems*, Elsevier Science, 1994.
19. X. Zhou et al., "Human Symptoms-Disease Network," *Nature Comm.*, 2014.