Networks and Optimization

Jure Leskovec (@jure)

Joint work with Austin Benson, David Hallac, Aditya Grover, Rok Sosič, Steven Boyd
Optimization in Networks

- Large problems can often be represented as a network!
  - Nodes... a series of subproblems
  - Edges... relationships that define the coupling between different subproblems

- Examples: cyber-physical and social networks, financial transactions, ...
Application: House Prices

House price prediction:

- **Nodes:** houses
- **Edges:** nearby houses
- Each house $i$ has its own price model:
  \[ p_i = x_{i1} \cdot SQFT + x_{i2} \cdot \#bedrms + x_{i3} \cdot \#bathrms \]
- But, nearby houses have similar models:
  - Houses in the same “neighborhood” share a common model
House Prices as Network Lasso

Simultaneous house clustering \((x_i = x_j)\) and network optimization!

\[ f(x_j) \]

\[ f(x_i) \quad \| x_i - x_j \|_2 \]

\( x_i \) ... house \( i \) model parameters

\( f(x_i) \) ... prediction error at house \( i \)

\( \| x_i - x_j \|_2 \) ... houses \( i \) and \( j \) share model parameters
Network Lasso

Simultaneous node clustering \((x_i = x_j)\) and network optimization
Network Lasso

Definition:

- Undirected graph $G = (V, E)$ with $m$ nodes, $n$ edges
- Solve for a set of variables $x_i \in \mathbb{R}^p$, $i = 1, \ldots, m$, one per node

\[
\text{minimize } \sum_{i \in V} f_i(x_i) + \lambda \sum_{(j,k) \in E} w_{jk} \|x_j - x_k\|_2
\]

- $\lambda \geq 0$, $w_{jk} \geq 0$

Network lasso penalty encourages nearby nodes to cluster together and share common values
Network Lasso: Nodes

\[
\text{minimize } \sum_{i \in \mathcal{V}} f_i(x_i) + \lambda \sum_{(j,k) \in \mathcal{E}} w_{jk} \|x_j - x_k\|_2
\]

- \( f_i(x_i) \) is the (convex) cost function at node \( i \)

- \( x_i \): possible examples
  - **Housing**: Parameter weights in regression model
  - Optimal actions to undertake in a control system

- \( f_i \)
  - **Housing**: How well the regression parameters fit the actual price
  - \(-1 \times \) expected profit
  - Fuel usage
Network Lasso: Edges

\[
\text{minimize} \quad \sum_{i \in V} f_i(x_i) + \lambda \sum_{(j,k) \in E} w_{jk} \| x_j - x_k \|_2
\]

- Not Laplacian regularization!
  - Incentivizes edge differences to be exactly zero
- When many edges are in consensus, the nodes are clustered into sets with equal values of \( x_i \)
  - Houses share the exact same regression weights
- Network lasso problem can be thought of as simultaneous clustering and optimization
**Regularization Penalty $\lambda$**

\[
\text{minimize } \sum_{i \in \mathcal{V}} f_i(x_i) + \lambda \sum_{(j,k) \in \mathcal{E}} w_{jk} \|x_j - x_k\|_2
\]

- Varying $\lambda$ can yield insight into the network structure
  - Cross-validation
- At $\lambda = 0$, the edges have no effect
- For $\lambda > \lambda_{\text{critical}}$, it turns into the consensus problem

\[
\text{minimize } \sum_{i \in \mathcal{V}} f_i(x)
\]

where the solution $x$ is common to every node
Solution: ADMM

- For large graphs, standard solvers don’t scale!

- Alternating direction method of multipliers (ADMM) splits the problem into subproblems
  - Parallelizable & Scalable
  - Each component (node/edge) solves its own private objective function, passes this solution on to its neighbors, and repeats
  - Without any global coordination, the entire network converges to the optimal solution!
    - Works for any convex $f_i, x_i$
Scalability

Convergence Time vs. Problem Size

- Centralized
- ADMM

Time (Seconds) for Entire Regularization Path

Number of Unknowns
## Housing: Performance

<table>
<thead>
<tr>
<th>Method</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geographic ($\lambda = 0$)</td>
<td>0.6013</td>
</tr>
<tr>
<td>Linear Regression ($\lambda \geq \lambda_{critical}$)</td>
<td>0.8611</td>
</tr>
<tr>
<td>Naive Prediction (Global Mean)</td>
<td>1.0245</td>
</tr>
<tr>
<td>Network Lasso</td>
<td>0.4630</td>
</tr>
</tbody>
</table>
Housing: Regularization Path

![Graph showing the regularization path for Housing data](image.png)
Housing: Neighborhoods
Housing: Neighborhoods
Housing: Neighborhoods
Housing: Neighborhoods
How do we infer networks?

Inferring Time-Varying Networks via the Graphical Lasso
D. Hallac, S. Boyd, J. Leskovec.
Sensing Complex Systems

Sensor arrays measure complex systems

Machines

Biomedicine

Infrastructure

Multidimensional time-series data!
From Time-Series to Networks

How do we encode structure and dependencies in temporal data?

Networks are a great way of encoding structure in the data.
Networks form Time Series

Model dynamic dependency structure

\[ x_1 \]
\[ x_2 \]
\[ x_3 \]

\[ a \]
\[ b \]

\[ \begin{align*}
  x_1 & \quad x_2 & \quad x_3 \\
  x_2 & \quad x_3 \\
  x_2 & \quad x_3 \\
\end{align*} \]
Our Approach

- **Goal:** Given a set of time-series data, learn the dependency network
  - Estimate inverse covariance matrix $\Theta$
    - If $\Theta_{ij} = 0$ then $i$ is conditionally independent of $j$
  - Network can evolve over time
    - Allows us to gain insights into the process and detect anomalies
Benefits

To learn a time-varying network, we can encode different types of temporal evolution:

- Single nodes rewiring
- Smoothly varying graph over time
- A few sharp breakpoints where the entire network changes
Our Approach: TVGL

Time-varying Graphical Lasso

- **Given:** Multivariate time series data
- **Goal:** Infer a sequence of networks (inverse covariance matrices)
  - 1) Segment the time series
  - 2) Infer the network for each segment

Note: We have $N^2T$ parameters

$N$ ... number of sensors
$T$ ... number of time points
Time-varying Graphical Lasso

Sequence of inv. cov. matrices $\Theta_i$:

$$\text{minimize} \quad \sum_{i=1}^{T} -l_i(\Theta_i) + \lambda \|\Theta_i\|_{od,1} + \beta \sum_{i=2}^{T} \psi(\Theta_i - \Theta_{i-1})$$

where (3 components):

- $l_i(\Theta_i)$: Log-likelihood of data at time $i$
- $\|\Theta_i\|_{od,1}$: Sparsity of the cov. matrix
- $\psi(\Theta_i - \Theta_{i-1})$: Temporal consistency
Time-varying Graphical Lasso

Temporal consistency $\psi(\Theta_i - \Theta_{i-1})$ models different types of network evolution:

- Single edge rewiring:
  $$\psi(X) = \sum_{jk} |X_{jk}|$$

- Edges smoothly rewiring over time:
  $$\psi(X) = \sum_{jk} X_{jk}^2$$

- Block-wise restructuring:
  $$\psi(X) = \sum_j \max_k |X_{jk}|$$

- Entire graph to rewire:
  $$\psi(X) = \sum_j \|X_j\|_2$$
Inferring the Networks

- Represent the problem as a graph:

\[- \log \text{det} \Theta_1 + \text{Tr}(S_1 \Theta_1) + \lambda \|\Theta_1\|_{od,1} \]
\[- \log \text{det} \Theta_2 + \text{Tr}(S_2 \Theta_2) + \lambda \|\Theta_2\|_{od,1} \]
\[- \log \text{det} \Theta_T + \text{Tr}(S_T \Theta_T) + \lambda \|\Theta_T\|_{od,1} \]

\[t_1 \quad \beta \psi(\Theta_2 - \Theta_1) \quad t_2 \quad \beta \psi(\Theta_3 - \Theta_2) \quad \cdots \quad \beta \psi(\Theta_T - \Theta_{T-1}) \quad t_T\]

- Distributed ADMM optimization solver:
  - Parallelizable and Scalable
  - Without any global coordination, the message passing algorithm quickly converges to the optimal solution
Particularly difficult to solve via standard methods. To infer a single solution time is due to the closed-form ADMM solutions we derived in §4). We experiment on a single 40-core CPU where the entire problem fits into memory. We set dates that we developed in §4). We compare our solvers to two semidefinite programming solvers (CVXOPT [5] and SCS [24]) and a naive ADMM method (without the closed form update). We solve the TVGL problem: two semidefinite programming solvers (CVXOPT [5] and SCS [24]) and a naive ADMM method (without the closed form update). As shown, problems which take hours for the other solvers can be solved in seconds using TVGL. For example, to solve for 50,000 points, our method is capable of solving problems with much larger values of parameters (largest perturbed node penalty, choosing this penalty leads to a 5% increase in reconstruction accuracy).

Selection of Penalty Parameter. As shown in Table 1, there are clear benefits from using certain penalties in certain situations. For the Local shift, the TVGL evolutionary penalty is better able to identify the specific node that was disabled. For the Global shift, the TVGL evolutionary penalty is better able to identify the specific node that was disabled. While many algorithms exist to efficiently solve the standard optimization problem and increase the regularization parameter, which to the best of our knowledge has not been previously done, it is intractable to scale these other methods beyond this point, until twice as large as both the static graphical lasso problem (with its TD ratio 23% larger than either method outlined in Section 6.1, but we vary our problem size over several orders of magnitude. Here, we are estimating the baselines are unable to detect that there was a shift in the net.

Table 1:

<table>
<thead>
<tr>
<th>Shift</th>
<th>Score</th>
<th>Temporal Deviation (TD) ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Static GL</td>
<td>0.496</td>
<td>1.06</td>
</tr>
<tr>
<td>Kernel</td>
<td>0.688</td>
<td>2.41</td>
</tr>
<tr>
<td>TVGL</td>
<td>0.939</td>
<td>0.819</td>
</tr>
<tr>
<td>TVGL (Perturbed Node)</td>
<td>0.853</td>
<td>0.817</td>
</tr>
</tbody>
</table>

7.1 Applications in Financial Data

We compare the performance of the four solvers in Figure 3.

Experiment: Scalability

- CVXOPT
- Naive ADMM
- SCS
- TVGL
Example Application: Stocks

- Each stock is a “sensor”
- Stock prices are correlated
- Use stock data to find interesting patterns and changes in the correlation network

Jure Leskovec, Stanford
Network reveals relationships between the different companies

- Apple
- Microsoft
- Intel
- Amazon
- Boeing
- FedEx
Result: Evolution

On Jan 27 network abruptly changes

What happened? Apple rewired. Why?

Perturbed Node Detection for Finance Data

What insights can we find by looking at how the stock network evolves?

Perturbed-node penalty:

The event that had the largest single-node effect on the dynamics of the network.

Temporal Deviation


10^{-3}  10^{-2}  10^{-1}  10^{0}

Apple: 1
Microsoft: 2
Amazon: 3
Intel: 4
Boeing: 5
FedEx: 6

Jure Leskovec, Stanford
Result: iPad was Introduced

iPad (1st generation)

From Wikipedia, the free encyclopedia

The first-generation iPad (/ˈaɪpæd/ eye-pad) is a tablet computer designed and marketed by Apple Inc. as the first in the iPad line. The device features an Apple A4 processor, a 9.7" touchscreen display, and on certain variants the capability of accessing cellular networks. Using the iOS operating system, the iPad can play music, send and receive email and browse the web. Other functions, which include the ability to play games and access references, GPS navigation software and social network services can be enabled by downloading apps.

The device was announced and unveiled on January 27, 2010, at a media conference. On April 3, 2010, the Wi-Fi variant of the device was released in the United States, followed by the release of the Wi-Fi + Cellular variant on April 30. On May 28, it was released in Australia, Canada, France, Japan, Italy, Germany, Spain, Switzerland and the United Kingdom.

The device received primarily positive reviews from various technology blogs and publications. Reviewers praised the device for its wide range of capabilities and labelled it as a competitor to laptops and netbooks. Some aspects were criticized, including the closed nature of the operating system and the lack of support for the Adobe Flash multimedia format. During the first 80 days, three million iPads were sold. By the launch of the second generation iPad (iPad 2), Apple sold more than 15 million iPads.

On March 2, 2011, Apple announced the second generation iPad and the discontinuation of production of the original iPad.[6]
Time Series Segmentation and Clustering
Using the Resulting Networks

- Can we use these networks for other tasks as well?

- Another high-value task in multivariate time series analysis is in segmentation and discovery of repeated motifs
  - We can use networks to achieve this goal!
Task Idea

- **Given:** Multivariate time series data
- **Goal:** Split the time series into a sequential timeline of a few key events
  - For example, from a fitness tracking device:
    - “Walking” for 30 minutes, then “running” for 1 hour, then “walking” for 10 minutes, then “sitting” for 5 minutes
The Challenge

- Can we simultaneously split the time series while also learning the structural “network signature” that defines each cluster?

- Our solution: Toeplitz Inverse Covariance-Based Clustering (TICC)
Solution Idea
Solution Idea

Segment the time series into a sequence of clusters (i.e., the alphabet) with stable temporal dependency structure

- For each cluster infer a temporal multilayer network
TICC

- Jointly tries to optimize three goals:
  - **Log Likelihood** – each point in the time series should “look like” the resulting network it is assigned to
  - **Sparsity** – the networks should be sparse
  - **Temporal Consistency** – assignments should not change too frequently across the time series

\[
\arg\min_{\Theta \in \mathcal{T}, \mathcal{P}} \sum_{i=1}^{K} \left[ \text{sparsity} \left\| \lambda \circ \Theta_i \right\|_1 + \sum_{X_t \in P_i} \left( \text{log likelihood} - \ell(X_t, \Theta_i) + \beta \mathbb{1}\{X_{t-1} \notin P_i\} \right) \right]
\]
Why Toeplitz Matrices?

- Enforces time invariance and allows for “cross-time” edges!
- Covariance structure of cluster $i$:
Solving the TICC Problem

- TICC is highly non-convex
- We solve it using an EM-like algorithm
  - Alternate between:
    - Dynamic Programming to assign points to clusters
    - ADMM to update the cluster parameters
Assigning Points to Clusters

- Find minimum cost path 1...T
  - Node cost: negative log-likelihood
  - Edge cost: Beta
Scalability

- 10m observations, each 50-dimensional, in just 25 minutes!
  - Scales linearly with the number of observations
Case Study: Automobiles

- Analyzed a 1 hour driving session
  - 36,000 samples @ 10Hz

- 7 sensors:
  - Brake pedal position
  - Forward acceleration
  - Lateral acceleration
  - Steering wheel angle
  - Vehicle velocity
  - Engine RPM
  - Gas pedal position
Running TICC

- Run TICC with k=5 clusters
- We plot the centrality score of each node in each cluster

<table>
<thead>
<tr>
<th>Brake</th>
<th>X-Acc</th>
<th>Y-Acc</th>
<th>SW Angle</th>
<th>Vel</th>
<th>RPM</th>
<th>Gas</th>
</tr>
</thead>
<tbody>
<tr>
<td>25.64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27.16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4.24</td>
<td>66.01</td>
<td>17.56</td>
<td>0</td>
<td>5.13</td>
<td>135.1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16.00</td>
<td>0</td>
<td>4.50</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32.2</td>
<td>0</td>
<td>26.8</td>
</tr>
<tr>
<td>4.52</td>
<td>0</td>
<td>4.81</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>94.8</td>
</tr>
</tbody>
</table>
Running TICC

- Run TICC with k=5 clusters
- We plot the centrality score of each node in each cluster

<table>
<thead>
<tr>
<th>Interpretation</th>
<th>Brake</th>
<th>X-Acc</th>
<th>Y-Acc</th>
<th>SW Angle</th>
<th>Vel</th>
<th>RPM</th>
<th>Gas</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1 Slowing Down</td>
<td>25.64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27.16</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>#2 Turning</td>
<td>0</td>
<td>4.24</td>
<td>66.01</td>
<td>17.56</td>
<td>0</td>
<td>5.13</td>
<td>135.1</td>
</tr>
<tr>
<td>#3 Speeding Up</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>16.00</td>
<td>0</td>
<td>4.50</td>
</tr>
<tr>
<td>#4 Driving Straight</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>32.20</td>
<td>0</td>
<td>26.8</td>
</tr>
<tr>
<td>#5 Curvy Road</td>
<td>4.52</td>
<td>0</td>
<td>4.81</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>94.8</td>
</tr>
</tbody>
</table>
Plotting the Resulting Clusters

- Green = straight, white = slowing down, red = turning, blue = speeding up
- Results are very consistent across the data!
Conclusion
Many Data are Networks

Social networks

Collaboration networks

Systems biology networks

Information networks: Web & citations

Internet

Networks of neurons

Figure 3: Higher-order cluster in the C. elegans neuronal network (28). A: The 4-node “bi-fan” motif, which is over-expressed in the neuronal networks (1). Intuitively, this motif describes a cooperative propagation of information from the nodes on the left to the nodes on the right.

B: The best higher-order cluster in the C. elegans frontal neuronal network based on the motif in (A). The cluster contains three ring motor neurons (RMEL/V/R; cyan) with many outgoing connections, serving as the source of information; six inner labial sensory neurons (IL2DL/VR/R/DR/VL; orange) with many incoming connections, serving as the destination of information; and four URA neurons (purple) acting as intermediaries. These RME neurons have been proposed as pioneers for the nerve ring (20), while the IL2 neurons are known regulators of nictation (21), and the higher-order cluster exposes their organization. The cluster also reveals that RIH serves as a critical intermediary of information processing. This neuron has incoming links from all three RME neurons, outgoing connections to five of the six IL2 neurons, and the largest total number of connections of any neuron in the cluster.

C: Illustration of the higher-order cluster in the context of the entire network. Node locations are the true two-dimensional spatial embedding of the neurons. Most information flows from left to right, and we see that RME/V/R/L and RIH serve as sources of information to the neurons on the right.
Network Data

Network data brings several core data mining methodologies into play:

- Working with network data is messy
  - Not just “wiring diagrams” but also dynamics and (meta)-data (features, attributes)

- Computational challenges
  - Large-scale network data

- Algorithmic models as vocabulary for expressing complex scientific questions
  - Social science, physics, biology
Four Fundamental Problems

- How to compute over networks
  - SNAP & big-RAM machines
- How to infer and build networks
  - Time-varying Graphical Lasso, TICC
- How to do ML on networks
  - Node2vec, GraphSAGE
- How to detect organization
  - Higher-order structures in networks
Stanford Network Analysis Platform (SNAP) is a general purpose, high-performance system graph analytics system

- [http://snap.stanford.edu](http://snap.stanford.edu)
- Scales to massive networks with hundreds of millions of nodes and billions of edges
- **Released SNAP 3.0 for C++ and Python**
  - 12K+ downloads in 2016
  - 610 stars, 320 forks on GitHub
  - 904K hits on SNAP datasets in 2016
References

- **Network Lasso: Clustering and Optimization in Large Graphs.** D. Hallac, J. Leskovec, S. Boyd. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2015.


- **Network Inference via the Time-Varying Graphical Lasso.** D. Hallac, Y. Park, S. Boyd, J. Leskovec. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2017. [Code and data]

- **Toeplitz Inverse Covariance-Based Clustering of Multivariate Time Series Data.** D. Hallac, S. Vare, S. Boyd, J. Leskovec. *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2017. [Code and data]