Network Lasso: Clustering and Optimization in Large Graphs

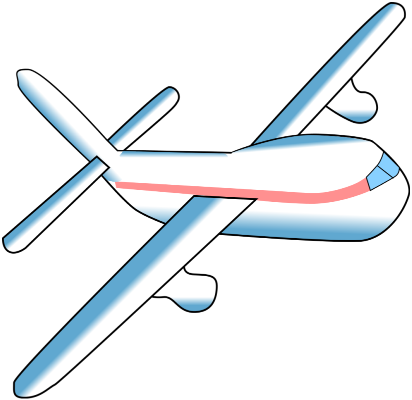
David Hallac, Jure Leskovec, Stephen Boyd

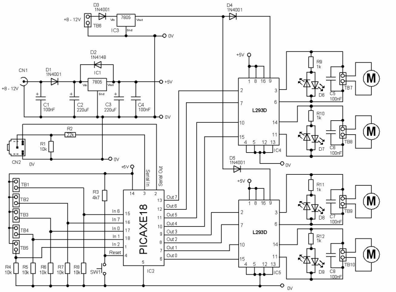
Stanford University

September 28, 2015

Convex optimization

- Convex optimization is everywhere





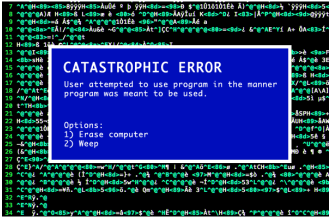
Tradeoff between generality and scalability

General Solver (CVX)

```
cvx_begin
    variable x(3);
    minimize(norm(A*x - b, 1))
    subject to
        x >= 0;
        x <= 1;
cvx_end
```

- Simple
- Easy to use
- **BUT...**
- Slow ($O(n^3)$ without sparsity)
- Cannot scale to large problems

Problem-Specific Solver



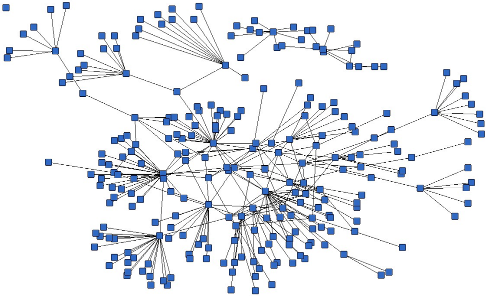
- Fast
- Efficient
- **BUT...**
- Large resource/time investment
- Limited to one specific problem

- Tradeoff that everyone has to make
  - General solvers don't scale; scalable solvers aren't general

Networks

- Large problems can often be represented as a network
  - Nodes - series of subproblems
  - Edges - relationships that define the coupling between the different nodes (entities)

- Examples: cyber-physical, social, financial transactions, ...

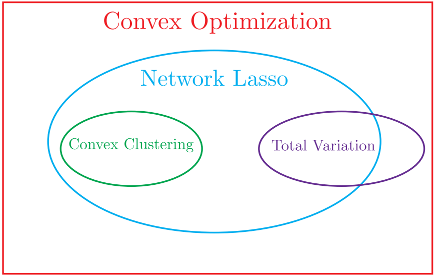


Our contributions

- Formally define the **network lasso**, a specific class of optimization problems on networks
- Develop a scalable method for solving problems of this form
- Show that many common and useful problems can be formulated as an instance of the network lasso
  - Focus on housing price prediction and network-enhanced classification

Related work

- Special case of some methods and a generalization of others



- Special type of Bayesian inference where we learn a set of models or dependencies based on latent clustering
- Many network lasso problems can be rewritten probabilistically
  - Two different ways to think of similar problems

Related work (cont'd)

- Instances of the network lasso
  - Convex clustering (Hocking et al. 2011)
  - Fused lasso (Tibshirani et al. 2005)
  - Total variation (Yang et al. 2013)
- Probabilistic Graphical Models (PGMs)
  - Latent variable mixture models (Muthen 2001)
  - Hinge-Loss MRFs with ADMM (Bach et al. 2013)
- Splitting and decomposition
  - Proximal algorithms (Parikh et al. 2014)
  - Alternating direction method of multipliers (Boyd et al. 2011)
  - Optimization decomposition (Chiang et al. 2007)

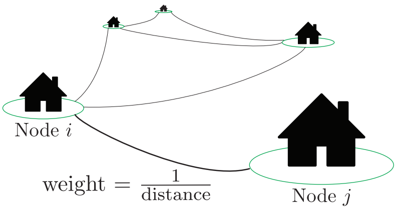
Application — housing price prediction

- The old way: linear regression
  - Use the same feature weights for every house in the dataset
- Problems with linear regression

- We instead want to cluster the houses into "neighborhoods" that share a common model

Using the network lasso

- Build a housing network where neighboring houses (nodes) are connected by edges



- Each house solves for its own regression parameters
- Network lasso penalty encourages nearby houses to cluster together and share a common model
  - Lets the data empirically determine the neighborhoods

Network lasso

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

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

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

What does it mean?

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

- $f_i(x_i)$ 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

What does it mean?

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

- The edge between nodes j and k has weight λw_{jk}
  - w_{jk} sets the relative weights among the edges of the network
  - λ scales the edge objectives relative to the node objectives f_i
- Edge objective penalizes differences between the variables at adjacent nodes
  - ℓ_2-norm of the difference

Network lasso penalty

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

- Not Laplacian regularization!
  - Incentivizes edge differences to be *exactly* zero (i.e. not just $x_j \approx x_k$, but $x_j = x_k$)
- When many edges are in consensus, the nodes are clustered into sets with equal values of x
  - Houses share the *exact* same regression weights
- The network lasso problem can be thought of as simultaneous clustering and optimization

Regularization path

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

- Varying λ 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

$$\text{minimize} \quad \sum_{i \in \mathcal{V}} f_i(x)$$

where the solution x is common to every node

Alternating direction method of multipliers (ADMM)

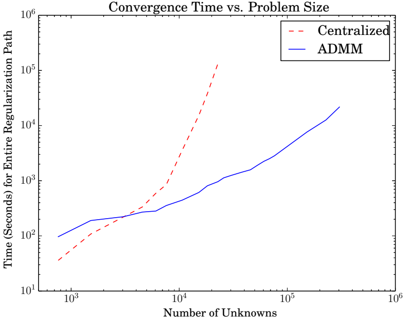
- For large graphs, standard (centralized) solvers cannot scale
- Alternating direction method of multipliers (ADMM) splits the problem up into a series of subproblems
  - **Parallelizable**
  - **Scalable**
- Each component (node/edge) solves its own private objective function, passes this solution 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

Experiments

- Scale our ADMM implementation to very large problems
  - Code and solver available at http://snap.stanford.edu/snapvx
- We see how the network lasso can be used to model many common examples
  - Compare accuracy/performance to baselines

Scalability

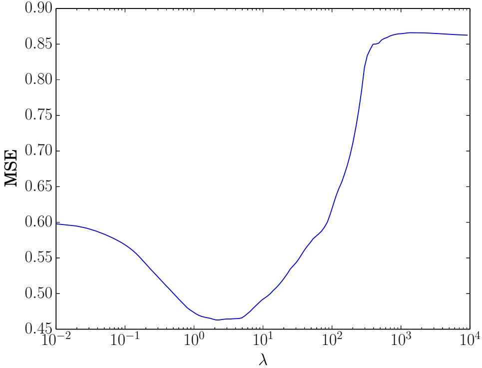
- ADMM can solve for 100 million variables in under 14 minutes!



Number of Unknowns	ADMM Convergence (seconds)
100,000	12.20
1 million	18.16
10 million	128.98
100 million	822.62

Housing

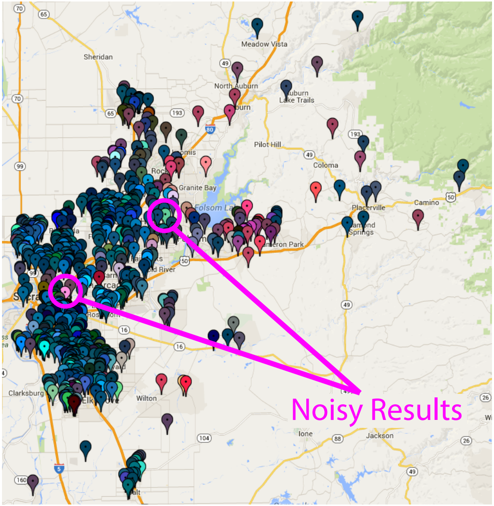
- Use network lasso to empirically find housing neighborhoods
- Compute mean squared error of sales price on test set
- Regularization path endpoints are important baselines



Method	MSE
Geographic ($\lambda = 0$)	0.6013
Linear Regression ($\lambda \geq \lambda_{\text{critical}}$)	0.8611
Naive Prediction (Global Mean)	1.0245
Network Lasso	0.4630

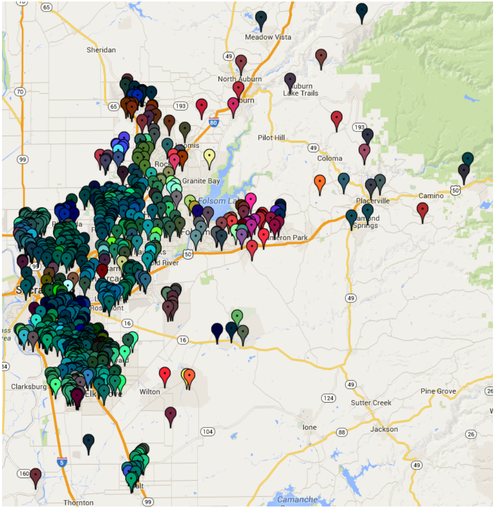
Regularization path heat map: $\lambda = 0.1$

- λ too small!



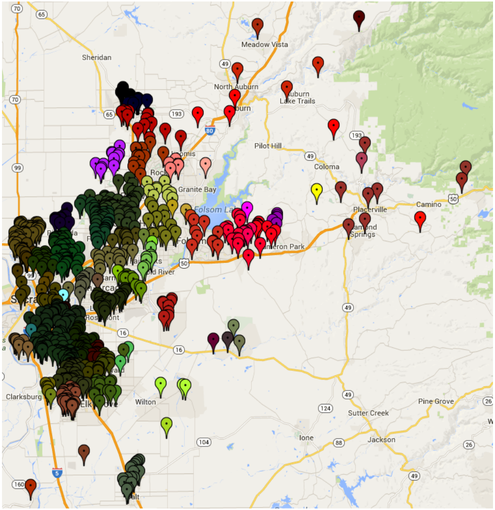
Regularization path: $\lambda = 1$

- Still too small...
- ...but houses are starting to group together



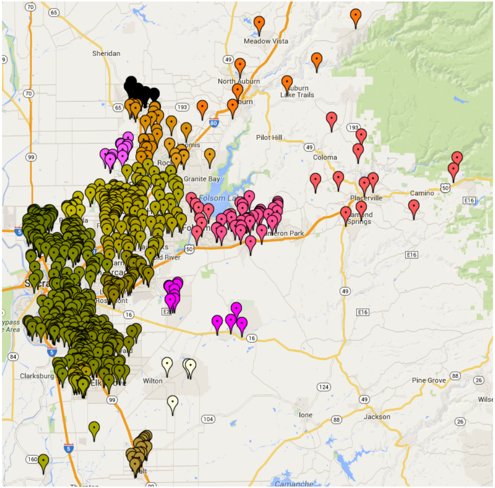
$\lambda = 10$

- Much better!



$\lambda = 100$

- Now it's too large...
- The network lasso pulls together clusters which are actually quite different

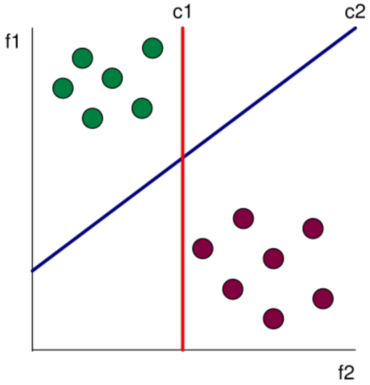


Network-enhanced classifier

- Support vector machine (SVM) at each node
  - Given an input $x \in \mathbf{R}^{50}$, predict an output $y \in \{-1, 1\}$
- However, individual nodes have insufficient information to properly classify new inputs
- Borrow statistical power from the network (nearby nodes) to improve prediction accuracy
  - Assumption: neighboring nodes often have similar (or the same) underlying classifier
- Example: music preferences in a social network

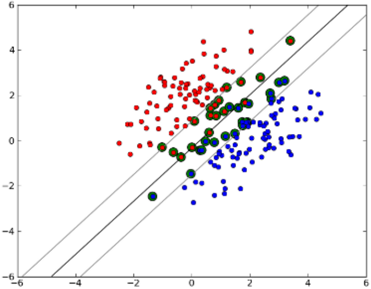
Training set

- Each node has very few training examples (known (x, y)-pairs)
- In higher dimensions, the uncertainty becomes even more significant
  - 50-dimensional data with only 25 training examples!



Network effect

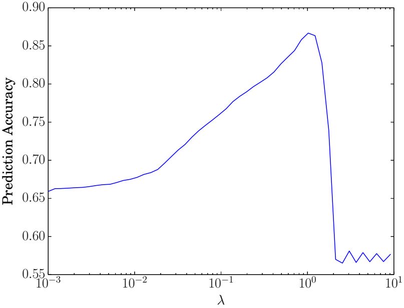
- Where can we find more data?
  - If we use training examples from our relevant neighbors, we can improve our own classifier
  - But how do we know who our relevant neighbors are? Use the network lasso!



Regularization path

- $\lambda = 0$: Each node does not have enough data, so there is uncertainty in our classifier
- $\lambda > \lambda_{\text{critical}}$: All the nodes will be clumped together and averaged into one uniform classifier
- We calculate the regularization path and evaluate performance on a separate test set to choose a good λ

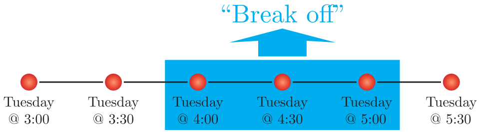
SVM results



Method	Prediction Accuracy
Local SVM ($\lambda = 0$)	65.9%
Global SVM ($\lambda \geq \lambda_{\text{critical}}$)	57.1%
Network Lasso	86.7%

Additional Applications

- Event detection in time series data



- ...and many more! (Image denoising, financial modeling, etc...)

Summary

- The **network lasso**
- A computationally tractable method of leveraging network data
- Fast, scalable, and robust
- The same setup can solve a variety of different problems
- Solver available at `http://snap.stanford.edu/snapvx`

Thanks for listening!

- Website: `http://www.stanford.edu/~hallac`
- Code and Solver: `http://snap.stanford.edu/snapvx`
- Acknowledgements: Tim Althoff, Stephen Bach, Trevor Hastie, Christopher Ré, Rok Sosič
- Questions?