SIMPLEX Proposal: Network Lasso

David Hallac
Stanford University
hallac@stanford.edu

Jure Leskovec
Stanford University
jure@cs.stanford.edu

Stephen Boyd
Stanford University
boyd@stanford.edu

Abstract

Convex optimization is an essential tool for modern data analysis, as it provides a framework to formulate and solve many problems in machine learning and data mining. However, general convex optimization solvers do not scale well, and scalable solvers are often specialized to only work on a narrow class of problems. Therefore, there is a need for simple, scalable algorithms that can solve many common optimization problems. We introduce the *network lasso*, a generalization of the group lasso to a network setting that allows for simultaneous clustering and optimization on graphs. We develop an algorithm based on the Alternating Direction Method of Multipliers (ADMM) to solve this problem in a distributed and scalable manner, which allows for guaranteed global convergence even on large graphs. We then demonstrate that many types of problems can be expressed in our framework. We focus on three in particular — binary classification, predicting housing prices, and event detection in time series data — comparing the network lasso to baseline approaches and showing that it is both a fast and accurate method of solving large optimization problems.

1 Problem Setup

Consider the following problem on a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the vertex set and $\mathcal{E}$ the set of edges:

$$\text{minimize} \quad \sum_{i \in \mathcal{V}} f_i(x_i) + \sum_{(j,k) \in \mathcal{E}} g_{jk}(x_j, x_k). \tag{1}$$

The variables are $x_1, \ldots, x_m \in \mathbb{R}^p$, where $m = |\mathcal{V}|$. (The total number of scalar variables is mp.) Here $x_i \in \mathbb{R}^p$ is the variable at node i, $f_i : \mathbb{R}^p \to \mathbb{R} \cup \{\infty\}$ is the cost function at node i, and $g_{jk} : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R} \cup \{\infty\}$ is the cost function associated with edge (j, k). Extended (infinite) values of f_i and g_{jk} describe constraints on the variables, or pairs of variables across an edge, respectively.

Network lasso. Our focus will be on the special case in which the f_i are convex, and $g_{jk}(x_j, x_k) = \lambda w_{jk} \|x_j - x_k\|_2$, with $\lambda \geq 0$ and user-defined weights $w_{jk} \geq 0$:

$$\text{minimize} \quad \sum_{i \in \mathcal{V}} f_i(x_i) + \lambda \sum_{(j,k) \in \mathcal{E}} w_{jk} \|x_j - x_k\|_2. \tag{2}$$

The edge objectives penalize differences between the variables at adjacent nodes, where the edge between nodes i and j has weight λw_{ij}. Here we can think of w_{ij} as setting the relative weights among the edges of the network, and λ as an overall parameter that scales the edge objectives relative to the node objectives. We call problem (2) the *network lasso* [1] problem, since the edge cost is a sum of norms of differences of the adjacent edge variables.

Regularization Path. Although the regularization parameter λ can be incorporated into the w_{ij}'s by scaling the edge weights, it is best viewed separately as a single parameter which is tuned to yield different global results. λ defines a trade-off for the nodes between minimizing its own objective and agreeing with its neighbors. At $\lambda = 0$, $x_i^\star$, the solution at node i, is simply a minimizer of f_i.

This can be computed locally at each node, since when $\lambda = 0$ the edges of the network have no effect. At the other extreme, as $\lambda \to \infty$, problem (2) turns into

$$\text{minimize} \quad \sum_{i \in \mathcal{V}} f_i(\tilde{x}), \tag{3}$$

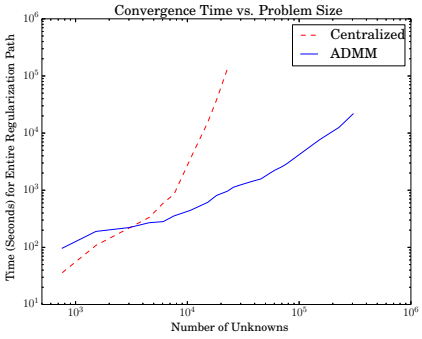
since a common $\tilde{x}$ must be the solution at every node. This is solved by $x^{\text{cons}} \in \mathbb{R}^p$. We refer to (3) as the *consensus problem* and to x^{cons} as the consensus solution. If a solution to (3) exists, it can be shown that there is a finite $\lambda_{\text{critical}}$ such that for any $\lambda \geq \lambda_{\text{critical}}$, the consensus solution holds. For λ's in between $\lambda = 0$ and $\lambda_{\text{critical}}$, the family of solutions is known as the *regularization path*.

2 Experiments

We analyze both the scalability and generality of our approach, examining several common examples. Our code and solver are available at `http://snap.stanford.edu/snapvx`.

2.1 Timing Results

Without any structure or sparsity to the problem, general (centralized) convex optimization solvers scale with the cube of the problem size. Our approach, based on the alternating direction method of multipliers (ADMM) [2], splits the problem into a series of subproblems and solves it iteratively via message passing between neighbors on the graph. To do so, it combines two pieces of open-source software, Snap.py [3] (for network analysis) and CVXPY [4] (for a convex solver). Our solution is parallelizable, scalable, and provides guaranteed convergence without any global coordination. We compare our approach with a general solver for the entire regularization path below.



2.2 Applications

The network lasso can apply to a variety of applications. This is because the algorithm works regardless of the network structure (as long as the graph is connected) and for **any** convex f_i and x_i. Thus, by changing these parameters, we can analyze problems from many different fields.

- **Event detection in time series data** - We analyze a building's entry and exit data over a 4 month interval. We use the network lasso to find "outliers", or times with anomalous traffic, and compare it to the official events that occurred in the building during that interval.
- **Housing price prediction** - We cluster houses into empirically determined neighborhoods that share a common regression model. Each house solves for its own regression parameters, but the lasso encourages nearby homes to cluster together with a shared model.
- **Network-enhanced classifier** - We look at a synthetic graph where each node has its own SVM, but individual nodes have insufficient information to build robust classifiers. Assuming that neighboring nodes have similar (or the same) classifiers, nodes can "borrow" statistical power from the network to improve prediction accuracy.

For each of these examples, we use the network lasso to make predictions on the network, and in all cases we outperform common baselines, showing that our approach is both a general and powerful method of solving convex optimization problems defined on networks.

References

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

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3:1–122, 2011.

[3] J. Leskovec and R. Sosič. Snap.py: SNAP for Python. `http://snap.stanford.edu`, 2014.

[4] S. Diamond, E. Chu, and S. Boyd. CVXPY. `http://cvxpy.org/`, 2014.