Applying SnapVX to Real-World Problems

David Hallac

Stanford University

Goal

- Other resources (MLOSS Paper, website, documentation, ...) describe the math/software side of SnapVX
- This presentation is meant to explain how SnapVX can be applied to real-world problems
  - Assumptions: Basic knowledge of optimization, graph theory
- For additional examples of how to solve machine learning problems in SnapVX, see http://snap.stanford.edu/snapvx/#examples

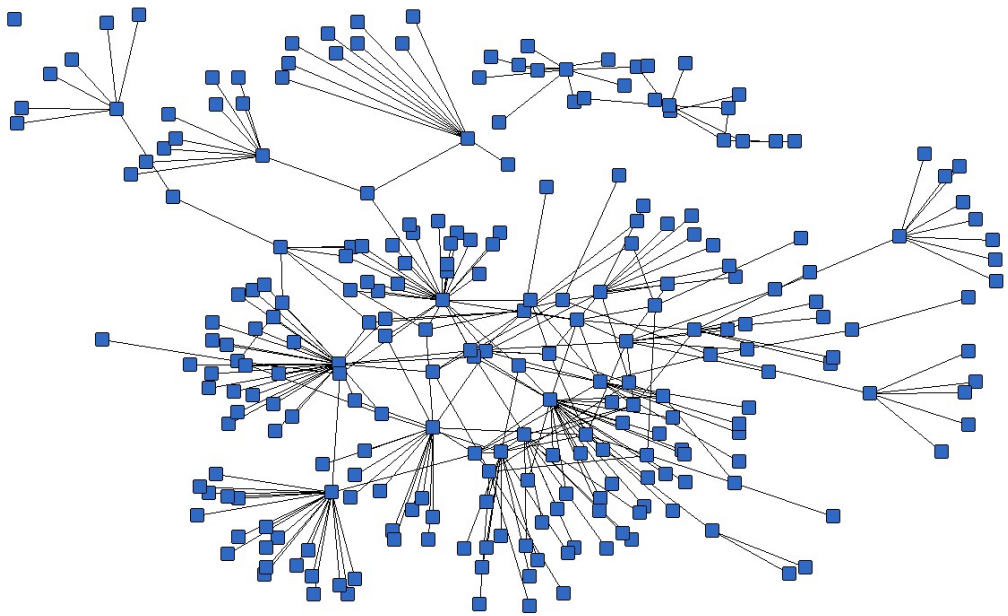
Putting a problem in SnapVX form

- Step 1: What is the network?
- Step 2: What are the objectives, constraints at each node and edge?
- Step 3: Are they convex?
  - If not, how can you form a convex relation of the original problem?
- Step 4: Solve!
- In this presentation, we use a running example to illustrate the process (predicting housing prices)
  - Example comes from "Network Lasso", KDD 2015

Step 1: What is the network?

- 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, . . .
  - Representational networks count too! (i.e. graphical models)

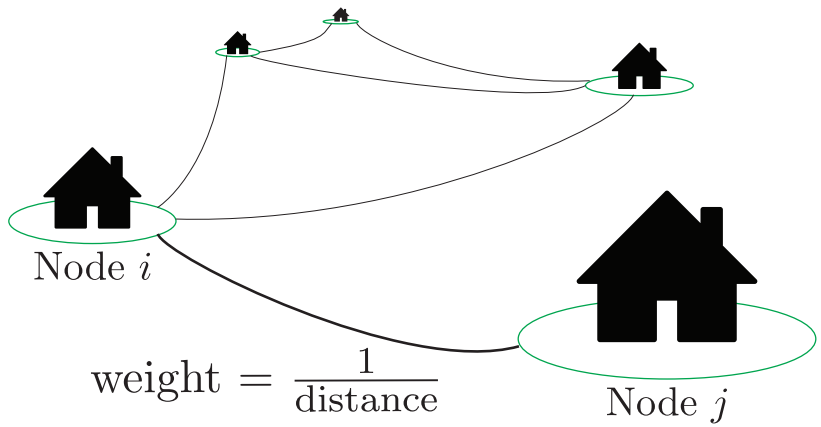


Application — housing price prediction

- Given: A set of houses
  - **Attributes**: latitude, longitude, general features (# of bedrooms, square footage, . . .)
- Goal: Build a regression model to predict the sales price of a house

Modeling the problem on a network

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



- Now, each node will solve for a vector x_i, the parameters weights in the regression model used to predict the price of house i

What are the objectives and constraints?

- Housing nodes want to build a good model to accurately predict price
  - For example, linear regression: $f_i(x_i) = \|Ax_i - b\|_2^2$
  - A are the feature values, b the sales price
- However, we are building a linear regression model with only one training example (the house itself!)
- This is where the edge objectives come in...

Edge objectives

- Housing edges connect nearby houses
  - Intuitively, we'd want nearby houses to share similar (or the same) regression model
- Depending on how we want the penalty to work, we define edge objectives to yield our desired behavior!
  - Network Lasso ($\|x_i - x_j\|_2$) penalty encourages the graph to **cluster** into groups of nodes which share common models of x
  - This can be thought of as simultaneously solving for a price prediction model while also discovering neighborhoods in the real estate market
- Other penalties can lead to different behaviors
  - For example, with Laplacian regularization ($\|x_i - x_j\|_2^2$), the housing model x_i will be unique at every house, but will change slowly and smoothly across the network

Tradeoff between node and edge objectives

- For each house, there is a tradeoff between accurately predicting its own price (the node objective), and agreeing on a similar model as its neighbors (the objective at each edge)
  - In other words, should it pick a "local" or a "global" model?
  - Too local: there is not enough data to build a robust classifier, so we will overfit our model
  - Too global: houses will be forced to share common models with houses very far away on the other side of town
    - And with real estate, location is everything!
- SnapVX makes it easy to tune these parameters to find the right values for each application
  - Edge weights, regularization, etc. . .

Solving the problem

- Once you've set up your graph and objectives, check convexity
  - If non-convex, there are many available resources to help "convexify" your problem (see "Convex Optimization" by Boyd and Vandenberghe)
- Now, just plug into SnapVX and solve!

Sample applications

- For more details on the housing example, see Hallac, Leskovec, and Boyd, KDD 2015
- Lots of common problems from many different fields can be efficiently solved via SnapVX
  - Event detection
  - Consensus and exchange
  - Pagerank
  - Fixed Routing
  - Network inference
  - Information diffusion
- See the SnapVX website or the "Examples" folder in the software download for additional concrete examples!

Summary

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

Thanks for reading!