Title: Network Inference for Genomics

Jesse Lingeman, Dennis Shasha

Courant Institute, New York University

* The Gene Network Inference Problem
	+ Definition
	+ Why important
	+ Approach of the book: break down into component steps and then mix and match into full workflows
		- Gene network inference is extremely data dependent. There is likely no one-size-fits-all algorithm to infer the best network from any dataset. Mixing and matching components of algorithms allows for creation of new algorithms that can be tailored to new data.
* Experimental Inputs
	+ Transcriptome
		- Wild type
		- Knock-out
		- Knock-down
		- Time Series
	+ Other
		- Genome sequence data
		- Operon data
		- Functional Associations and Annotations
* Running Examples
	+ Generated using GeneNetWeaver (how to guide)
	+ Small example
	+ Large example
* Overall Workflow of Inference
	+ Inferring a network requires the coordination of several different analysis steps. Each step can be done in different ways. The first step clusters the elements of large data sets in order to reduce the size of the inference problem. The second step uses steady state knock-out data (if any) to infer certain strong relationships. The third step infers a dynamical graph model whose nodes are either single genes clusters and whose edges are either inductive or repressive edges which may be “seeded” based on the results of the knock-out data. The fourth step prunes the dynamical model based on resampling and consensus techniques.
	+ Workflow diagram of the three steps

Step 1: Clustering for large datasets

* + - Biclustering
			* cMonkey
				+ Uses steady state, genome sequence, operon, and functional association data to cluster genes.
				+ Available data is automatically downloaded
				+ Use each cluster as a gene by averaging data
				+ Parameters:

Weights on each type of data used

Max number of biclusters a gene can belong to

* + - Gene clustering
	+ Step 2: Using steady-state knock-out/knock-down data
		- Notes on how this data is often incomplete, because generating gene knockouts for each gene in the dataset is costly
		- Median Corrected Z-Scores (from Inferelator 2.0 / Pinna 2010)
		- Ordinary Differential Equations for learning networks from only steady-state data (NIR/NTW)
	+ Step 3: Building a dynamical model
		- Dynamic Bayesian Networks (BANJO)
			* BANJO
				+ Dynamic Bayesian Network

Works well when there are far more experiments (time points) than genes.

Does not make assumptions about linearity or non-linearity of relationships

Uses greedy search with many random restarts, simulated annealing, or genetic algorithms to create the networks. The networks are then evaluated based on scoring criteria.

(Yu, Smith, Wang, & Hartemink, 2004)

* + - Information Theoretic Approach (Time-Delay ARACNE)
			* Uses Mutual Information
			* Examples of generating networks based on mutual information and context likelihood of relatedness (CLR), which is an extension of MI.
		- Dynamic Factor Graphs (DFG4GRN)
			* Uses an ODE to learn the edges between nodes on the factor graph
			* Few parameters (3)
			* Only requires time series data
			* Models the noise in the data, removes it, then uses an ODE to model the “idealized” values
			* Works well on small datasets
			* (Krouk, Mirowski, LeCun, Shasha, & Coruzzi, 2010)
		- Ordinary Differential Equations (Inferelator 1.0)
			* NTW / NIR
				+ Inference Algorithm

Ordinary Differential Equation

Uses only steady state data

NIR: Perturbations to each gene must be known *a priori*

NTW: Perturbations are inferred

* + - * Inferelator 1.0
				+ Learn a sparse dynamical model of regulation for each gene by regulators in some set of genes. These regulators can be selected by the previous two steps, or with a known list of transcription factors. LARS is used to implement an l\_1 constraint and enforce sparsity.
			* An ODE example with time series data
			* LARS/LASSO
	+ Step 4: Pruning the dynamical model
		- Thresholds
		- Data Processing Inequality (Time-Delay ARACNE)
		- Resampling
			* Consensus network (DFG4GRN)
			* Markov-Chain Monte-Carlo Methods (Inferelator 2.0)
	+ Sample Pipelines
	+ Example 1: Inferelator 2.0
		- Data:
			* Steady-state Knock-out data (optional)
			* Time-Series data
		- Handling Steady-state/Knock-out data
			* Median Corrected Z-Scores

Predicts the topology of the network (tied for first in DREAM4)

Takes the median of the wild-type and knock-out values for each gene

Calculates the z-score between the median value of knockouts and wildtypes to see when the current gene is affected by another one being knocked out.

* + - Inference Algorithm
			* Time-Lagged Context Likelihood of Relatedness
				+ Uses an ODE to model temporal changes
				+ Calculates the a static (based on steady state) and dynamic (time series) mutual information score between each pair of genes.
				+ Applies a filter to trim weaker links between genes
			* Inferelator 1.0
				+ Learn a sparse dynamical model of regulation for each gene by regulators in some set of genes. These regulators can be selected by the previous two steps, or with a known list of transcription factors. LARS is used to implement an l\_1 constraint and enforce sparsity.
	+ Example 2: Time-Delay ARACNE
		- Original formulation, ARACNE, was unable to infer directionality of edges, but with time delay extension, it can
		- Data:
			* Time Series Data
		- Parameters:
			* K: The how many steps to look ahead for influence?
			* Pruning tolerance
		- Works in 3 steps:
			* Step 1: Preprocessing Time-Series data.
				+ Detect time point at which each gene becomes excited/inhibited
			* Step 2: Network Inference.
				+ Estimate the joint probability between gene\_a at time t and gene\_b at time t+k, trying to find the best k. Then calculate the mutual information between those genes using the joint probability. Because the MI is calculated using the time shifting above, directionality can be inferred. If information between a pair is above a certain threshold (calculated automatically by bootstrapping on the data type), then a directed edge is drawn between the nodes.
			* Step 3: Pruning the Network.
				+ Prune the gene networks using the data processing inequality and a corresponding threshold (default: 15%). This essentially penalizes indirect edges and only accepts them if they are strong
	+ Example 3: ….
* Roll Your Own Pipeline
	+ Considerations and issues to think about when creating your own pipeline. Mostly, taking into account features of your data. If you have very little time-series (few experiments and few genes) data, then you’re going to want to use a different inference algorithm than if you have many experiments by many genes. How does one use the steady-state data in the most effective manner? What kind of post-processing needs to be done? For each of these sample pipelines, show what happens if we remove a pre- or post-processing step. I.e., what happens when we don’t use clustering on a giant network with DFG4GRN? How do the results compare?
	+ Sample Pipeline 1: Using DFG4GRN on very large networks by using biclusters
		- Data: Only time-series
		- Works in three steps:
			* Step 1: Generate biclusters using cMonkey
			* Step 2: Network inference with DFG4GRN
			* Step 3: Build consensus network from ~20 runs of DFG4GRN
	+ Sample Pipeline 2: Using NTW to build a consensus network with Inferelator 2.0
		- Data: Steady-state knock-out data and time-series
		- Works in three steps:
			* Step 1: Compute network from steady state data with NTW
			* Step 2: Compute network from steady state and time series data with Inferelator 2.0
			* Step 3: Compare networks, select edges that are in both. Or select via some heuristic on the strength of each edge in both networks. (Will need to try this out)
	+ 2-3 more examples, …
* Appendix
	+ Data Generation
		- GeneNetWeaver
			* Examples include DREAM3/4, E. Coli, Yeast
			* Parameters to use in generating your own data
	+ Bibliography
	+ Further Reading
	+ Software Examples
	+ Datasets used