# A Nonlinear Dimensionality Reduction-Based Approach to Biological Network Reconstruction – Interim Status Report

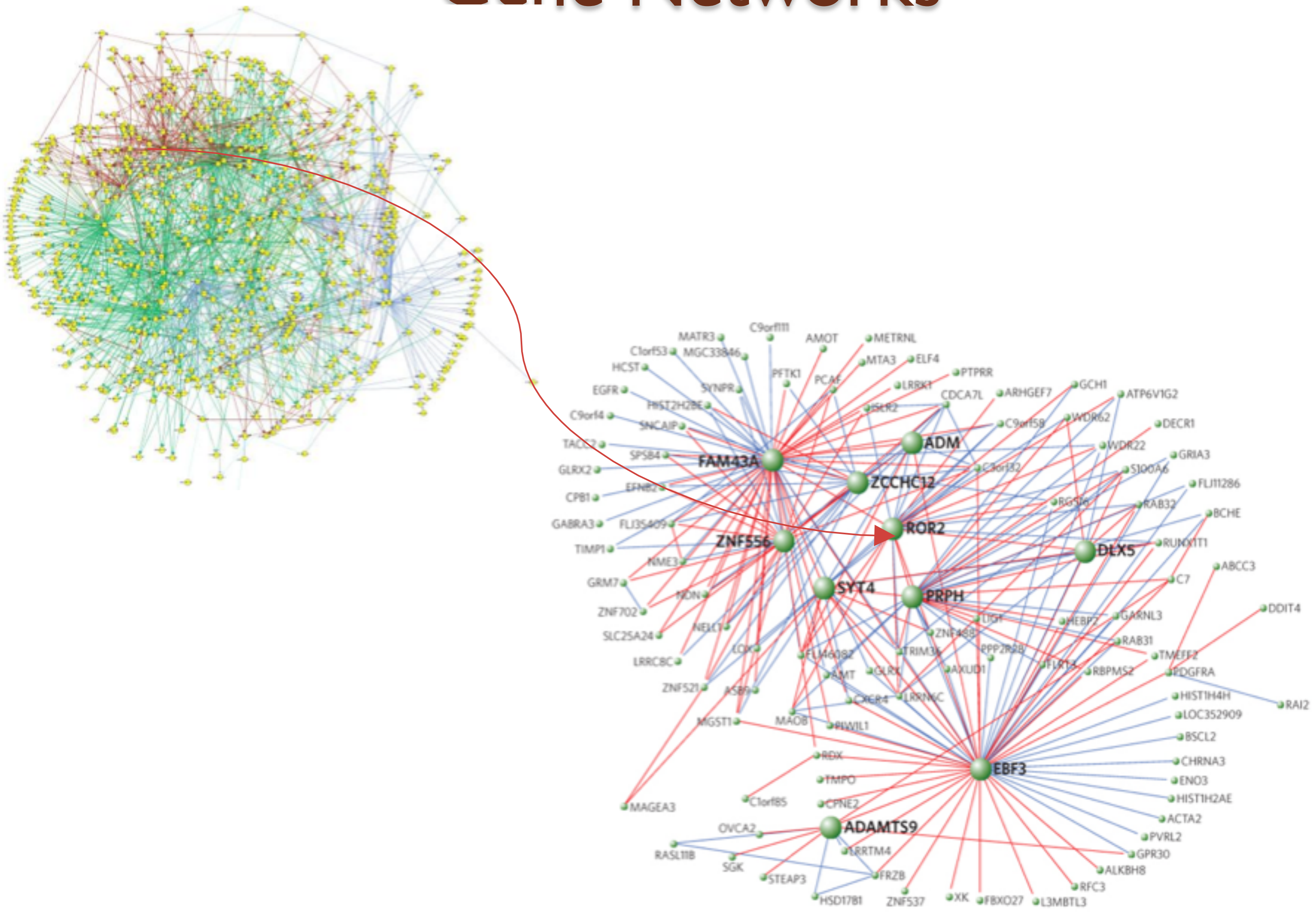
Vinodh N. Rajapakse (vinodh@math.umd.edu)

Advisor: Prof. Wojciech Czaja (wojtek@math.umd.edu)

## Presentation Outline

- Problem Review
- Solution Approach
- Work Accomplished This Term
- Upcoming Steps
- Questions and Comments

# Gene Networks



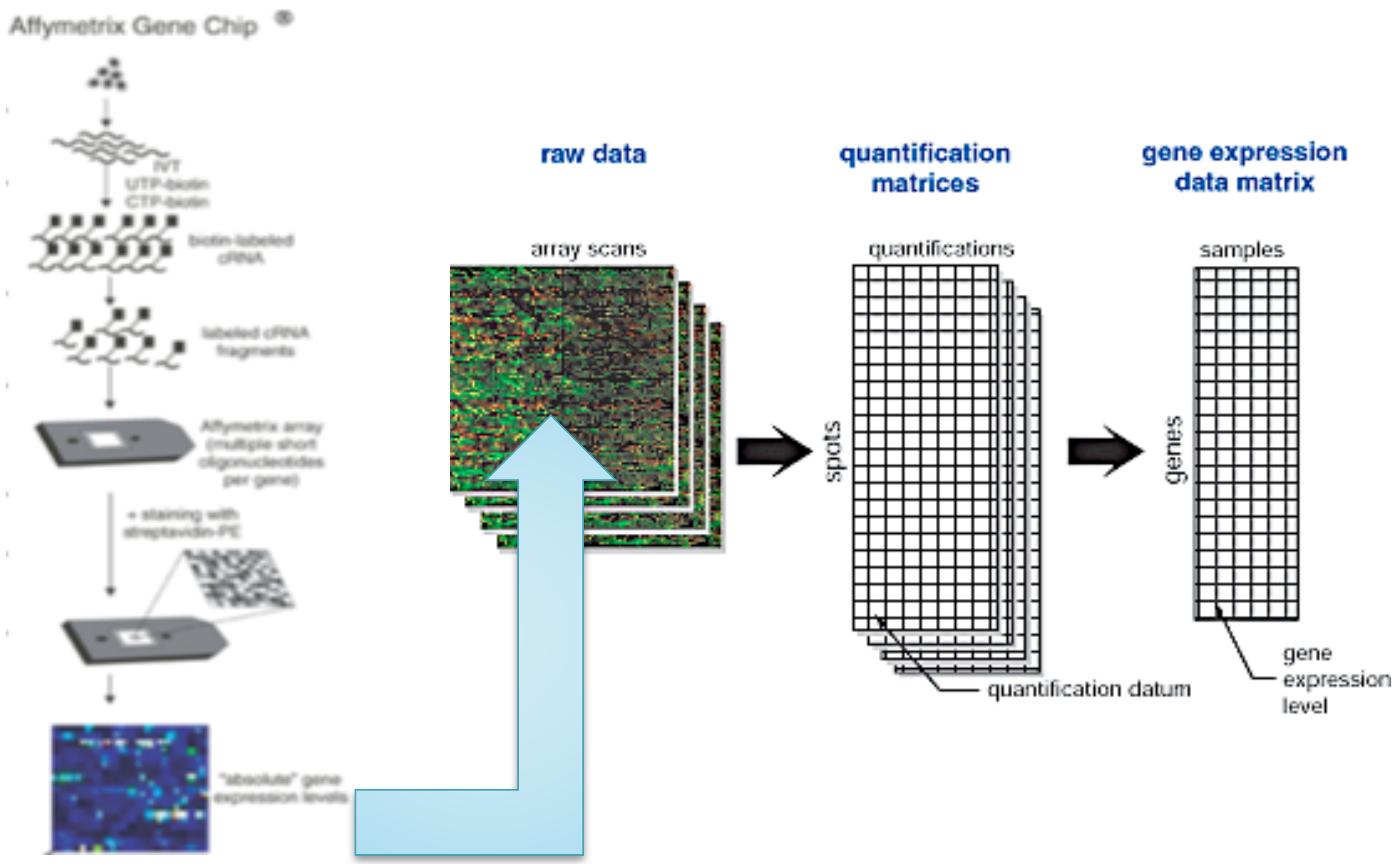
## Why Build Gene (etc.) Networks?

- Gain a broader, systems level view of biological processes and their underlying functional elements
  - Avoid a narrow focus on a limited subset of driving elements
  - Incisively identify the most promising targets for experimental exploration (to derive focused data for iteratively refining models)

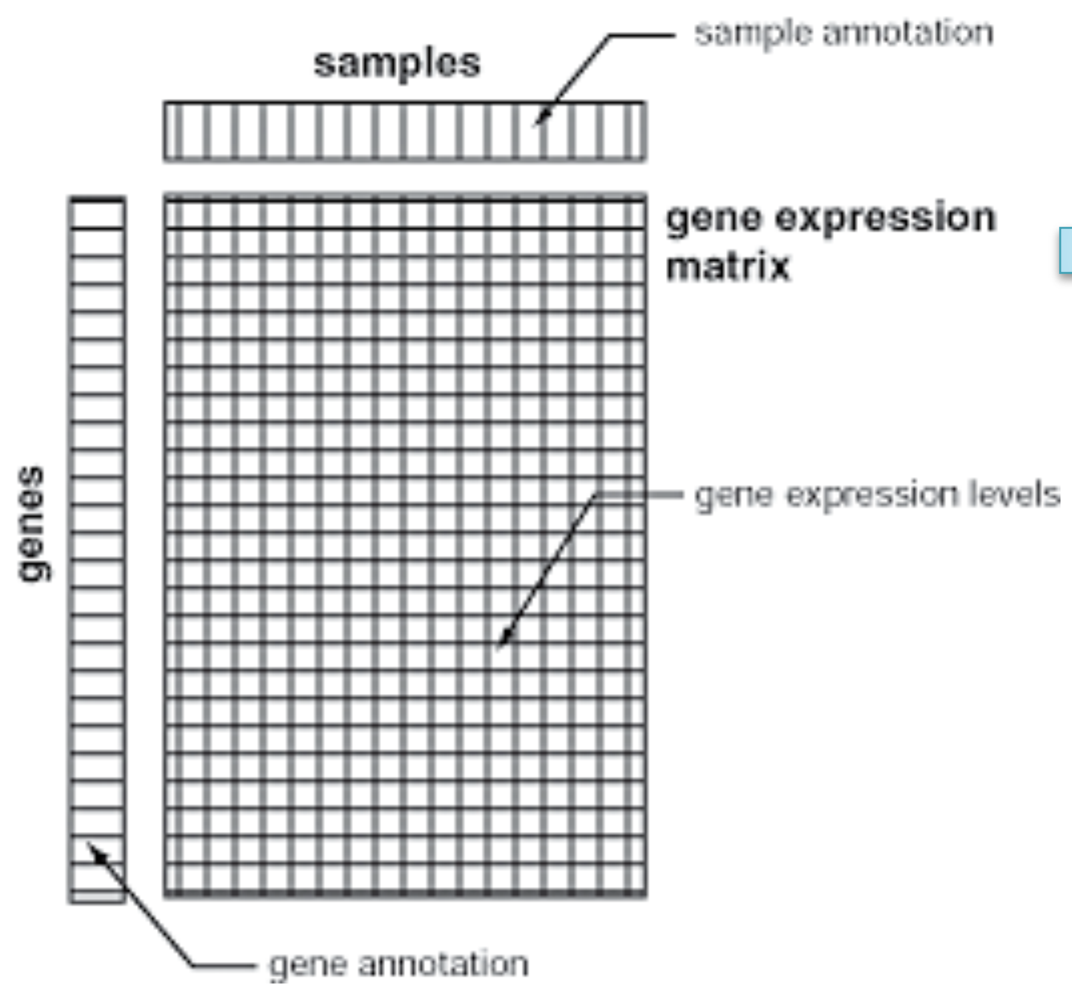
## How to Build Biological Networks?

- Manually - using expert knowledge, detailed review of research results
  - Only avenue until relatively recently
  - Necessarily small scale – a few reliable (experimentally verified) nodes and links, many, many missing ones.
- Computationally – using large scale measurements of molecular expression (abundance) values over many biological samples

# Gene Expression Microarrays



## Starting Point: Gene Expression Data Matrix



$\rightarrow$ $\mathbf{X}$ **(N x D)**

N = ~10000 genes
D = ~250 samples

## Basic Network Construction Workflow

- Starting from from the N x D gene expression data matrix **X**, derive an N x d matrix **Y**, (d < D) using Laplacian Eigenmaps (or another dimensionality reduction technique).
- Construct an N x N matrix **W*** capturing pairwise Euclidean distances between row vectors of **Y** ('reduced gene profiles').
- Apply a threshold to the elements of **W*** to obtain a network (adjacency matrix) representation.

## Laplacian Eigenmaps

- Input: **X** (N x D) → Output: **Y** (N x d)
  - Let $\mathbf{x}$ = ( $x_1, x_2, \ldots, x_D$) denote a row of **X**
  - Let $\mathbf{y}$ = ( $y_1, y_2, \ldots, y_d$) denote a row of **Y**
- Step 1: Model Data Point Relationships
  - Build a graph G, with nodes i and j connected if $\mathbf{x}_i$ is one of the k nearest neighbors of $\mathbf{x}_j$ or vice versa (Euclidean distances used, alternatives are possible)
  - k is a local structure resolution parameter

## Laplacian Eigenmaps

- Step 2: Form Weight Matrix
  - Form a diffusion weight matrix W, with entry $W_{i,j} = \exp\{ - \| x_i - x_j \|^2 \ / \ \sigma \}$, if i and j are connected; 0 otherwise.
- Step 3: Solve Minimization Problem

$$\min_{(Y^T DY=I)} \frac{1}{2} \sum_{i,j} \| y_i - y_j \|^2 W_{i,j}$$

## Laplacian Eigenmaps

- Step 3 (cont.) Solve Eigenvalue Problem
  - Given weight matrix **W**, let **D** be a N x N diagonal ('connectivity') matrix with entries recording the sum of edge weights for each data point-derived node
  - Let **L** = **D** – **W** denote the Laplacian matrix
  - We have:

$$\min_{(Y^T DY=I)} \frac{1}{2} \sum_{i,j} \| y_i - y_j \|^2 W_{i,j} = \min_{(Y^T DY=I)} \operatorname{trace}(Y^T LY),$$

## Laplacian Eigenmaps

- Given: $\min_{(Y^T DY=I)} \frac{1}{2}\sum_{i,j} \| y_i - y_j \|^2 W_{i,j} = \min_{(Y^T DY=I)} \mathrm{trace}(Y^T LY),$
- Basic results from linear algebra show that the optimal mapping can be obtained by solving generalized eigenvalue problem **Lx** = $\lambda$ **Dx**, under the above constraint.
- In particular, coordinates for the mapped vector $y_i$ can be extracted from the ith coordinates of the d eigenvectors with smallest nonzero eigenvalues.

## Laplacian Eigenmaps

- Additional Details:
  - $Lx = \lambda Dx \Leftrightarrow (D^{-1/2} L D^{-1/2})u = \lambda u$, where $u = D^{1/2}x$, $(D^{-1/2} L D^{-1/2}) = L_{sym}$
  - Estimation of intrinsic data dimensionality d
  - Selection of local neighborhood resolution parameter k
  - Selection of kernel width parameter $\sigma$
  - Approximate Nearest Neighbor Selection algorithms for managing larger data sets

## Basic Network Construction Workflow

- Starting from from the N x D gene expression data matrix **X**, derive an N x d matrix **Y**, (d < D) using Laplacian Eigenmaps (or another dimensionality reduction technique).
- Construct an N x N matrix **W*** capturing pairwise Euclidean distances between row vectors of **Y** (‘reduced gene profiles’).
- Apply a threshold to the elements of **W*** to obtain a network (adjacency matrix) representation.

## Network Derivation

- Given:
  - Target (fractional) value $\alpha$
  - N x N matrix **W*** capturing pairwise Euclidean distances between mapped gene expression profiles in reduced dimensional space.
- Estimate: Distance Threshold
  - Rank mapped data space pairwise distances.
  - Select distance threshold that excludes upper ($1-\alpha$) fraction of observed values.

## Basic Network Construction Workflow

- Starting from from the N x D gene expression data matrix **X**, derive an N x d matrix **Y**, (d < D) using Laplacian Eigenmaps (or another dimensionality reduction technique).
- Construct an N x N matrix **W*** capturing pairwise Euclidean distances between row vectors of **Y** ('reduced gene profiles').
- Apply a threshold to the elements of **W*** to obtain a network (adjacency matrix) representation.

## Work Accomplished This Term

- Phase I: Laplacian Eigenmaps + Network Reconstruction
  - Target Date: Middle of December 2010
  - Milestones:
    - (C++) Implementation and focused validation of Laplacian Eigenmaps
    - Basic (distance matrix threshold-based) network reconstruction.

## Implementation Challenges

- Selecting and integrating appropriate (public domain) software to efficiently solve eigenvalue problem.
- Organizing data structures and operations to conserve memory and support scalability.

## Linear Algebra Libraries

- Need to solve sparse, symmetric eigenvalue problem.
- Basic BLAS/LAPACK largely emphasize dense matrices.
- Evaluated several C++ packages
  - uBLAS (BLAS routines, relatively slow)
  - Armadillo++, Eigen (nice, almost MATLAB-like interface w/operator overloading, but meager support for sparse matrices/eigenproblems)

## ARPACK

- ARnoldi PACKage: Fortran 77 library for solving large scale sparse eigenvalue problems
- Used by MATLAB (e.g., eigs function)
- For symmetric matrices, applies Lanczos Algorithm

## ARPACK and Memory Management

- Reverse Communication Interface:
  - ARPACK routines do not operate directly on matrices
  - Instead: work with function defining matrix vector product. Allows matrices to be stored in any suitable format (or not at all).
- Implementation exploits this to represent matrices using compact adjacency lists, with fast 'in-place' operations where possible

## Linear Algebra/ARPACK interface

- Organized ARPACK interface code, compressed matrix classes into convenient package, with overloaded operators and high-level, template-based methods.
- Basic, re-usable building block which will facilitate additional algorithm implementations.

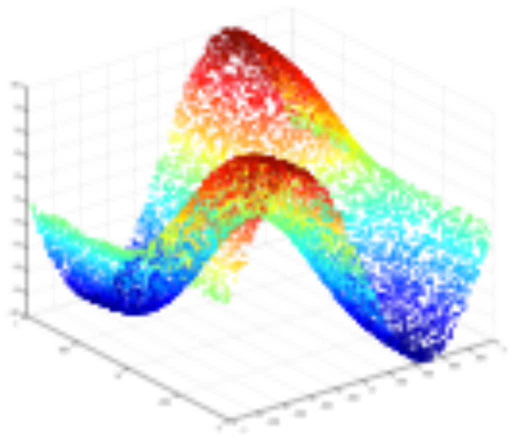
## Linear Algebra/ARPACK Interface

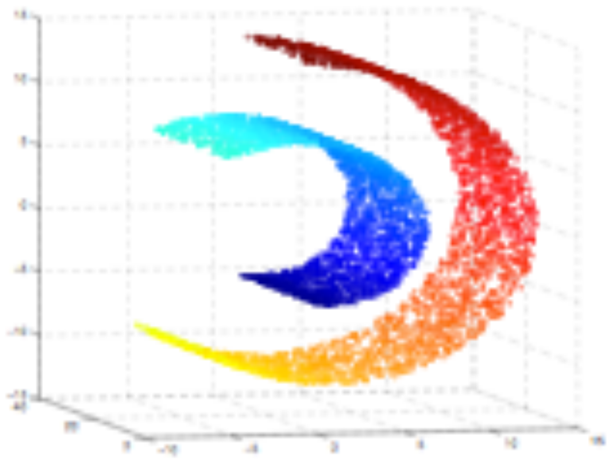
-
  ```
  extern "C" void dsaupd_(int *ido, char *bmat,
  int *n, char *which,int *nev, double *tol,
  double *resid, int *ncv, double *v, int *ldv,
  int *iparam, int *ipntr, double *workd,
  double *workl, int *lworkl, int *info);
  ```

-
  ```
  template<typename T>
  void sparseSymEigSolve(
  const CompressedMatrix<T>& M,
  const Matrix<T>& evecs,
  const NumVector<T>& evals);
  ```

## Validation and Testing

- Laplacian Eigenmaps Implementation
  - Compared ARPACK interface code to corresponding MATLAB routines (leigs)
  - Compared to established MATLAB implementation over 4 published synthetic data sets.


(c) Twinpeaks dataset.


(d) Broken Swiss roll dataset.

## Upcoming Work

- Phase II: Integrated Testing of Network Reconstruction + Possible Extensions
  - Target Date: end of March 2011
  - Milestones:
    - Integrated testing of network reconstruction
    - Comparison of results obtained using nonlinear dimensionality reduction (LE), linear dimensionality reduction (PCA), and original data
    - Approximate Nearest Neighbor Algorithm
    - Possible: Diffusion Maps

## Upcoming Work

- Integrated Network Reconstruction
  - Compare to published results for leading ARACNE network reconstruction method over:
    - Synthetic Gene Expression Data Set (allows objective scoring of 'true' edge recovery)
    - Well-studied biological data set (compare network reconstruction around MYC oncogene)
  - Compare results Laplacian Eigenmaps-based networks with those derived from original data, PCA-processed data.

## References

- **M. Belkin and P. Niyogi, *Laplacian Eigenmaps for dimensionality reduction and data representation, Neural Computation.* 15 (2004), No. 6, 1373-1396**
- **Chung - Spectral Graph Theory http://www.math.ucsd.edu/~fan/research/revised.html**
- M. Ehler, V. Rajapakse, B. Zeeberg, B. Brooks, J. Brown, W. Czaja, and R. F. Bonner, *Analysis of temporal-spatial co-variation within gene expression microarray data in an organogenesis model.* 6th International Symposium on Bioinformatics Research and Applications (ISBRA'10), Lecture Notes in Bioinformatics, Springer Verlag, 2010
- A. A. Margolin, I Nemenman, K. Basso, C. Wiggins, G. Stolovitzky, R Dalla Favera, A. Califano, *ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context.* BMC Bioinformatics. 2006 Mar 20;**7** Suppl 1:S7.
- van der Maaten, Postma, van den Herik, *Dimensionality Reduction: A Comparative Review.* Tilburg Centre for Creative Computing Technical Report 2009-005
- **von Luxburg, U. A Tutorial on Spectral Clustering. Statistics and Computing 17(4), 395-416 (12 2007)**
- Zeeberg B, Qin H, Narasimhan S, et al. *High-Throughput GoMiner, an 'industrial-strength' integrative gene ontology tool for interpretation of multiple-microarray experiments, with application to studies of Common Variable Immune Deficiency (CVID).* BMC Bioinformatics 2005; Jul 5; **6**:168.