NMF Documentation

Module name:	**NMF**
Description:	Non-negative Matrix Factorization
Author:	Pablo Tamayo (Broad Institute), gp-help@broad.mit.edu
Date:	May 22, 2003
Release:	1.0

Summary: NMF is an efficient method for identification of distinct molecular patterns and provides a powerful method for class discovery. For class discovery, NMF appears to have higher resolution than other methods such as hierarchical clustering or self-organizing maps, and to be less sensitive to a priori selection of genes. Rather than separating gene clusters based on distance computation, NMF detects context-dependent patterns of gene expression in complex biological systems.

The basic principle of dimensionality reduction via matrix factorization operates as follows: given an N x M data matrix A with positive entries, the NMF algorithm iteratively computes an approximation, A ~ WH, where W is an N x k matrix, H is a k x M matrix, and both are constrained to have positive entries. The rank of A (i.e., the number of its linearly independent columns) has been reduced from M to k < M. For DNA microarrays, N, the number of genes, is typically in the thousands. M, the number of experiments, rarely exceeds a hundred, while k, the number of classes to be determined depends on the heterogeneity of the dataset. The algorithms starts with randomly initialized matrices of the appropriate size, W and H. These are iteratively updated to minimize the Euclidean distance between V and WH. The program also computes row and column factor memberships according to maximum amplitudes. This membership information is also used to sort the output matrices according the row and column membership (the row and columns are then relabeled: <name>_f<NMF factor>.

This version is a Perl version of the Euclidean NMF equations written in Perl. It is slow and is intended for exploratory use. A faster version of the NMF including model selection is available in Matlab and will be added to GenePattern in the near future.

References:

- Jean-Philippe Brunet, Pablo Tamayo, Todd Golub, Jill Mesirov. Matrix Factorization for Molecular Pattern Recognition (in press, PNAS)
- Lee, D.D and Seung, H.S. (1999), 'Learning the parts of objects by non-negative matrix factorization', Nature 401, 788-793.
- Lee, D.D., and Seung, H.S., (2001), 'Algorithms for Non-negative Matrix Factorization', Adv. Neural Info. Proc. Syst. 13, 556-562.

Parameters:

Name	Description
seed:	Random seed used to initialize W and H matrices by the random number generator. e.g. 4585, 4567, 5980 default value: 12345
niter:	Number of NMF iterations. e.g. 100, 250, 500 default value: 100

nfact: Number of NMF factors. e.g. 2, 4, 10
default value: 4

input.filename: Input file in GCT format with input data (V matrix)
(GCT format only). e.g. ALL_vs_AML.gct

print.dump: Print matrices in output for debug
choose from the following values: 0=no, 1 = yes
default value: 0

cnorm.flag: Column normalization flag
choose from the following values: 0=no normalization,
1 = normalization
default value: 0

rnorm.flag: Row normalization flag
choose from the following values: 0=no normalization,
1 = normalization
default value: 0

gshift: Global additive shift after normalization. e.g. 1, 4
default value: 0

norm.type: Type of normalization
choose from the following values: 0= stardardize, 1 = rescale: min=0,
max=1
default value: 0

output.file: Output log file. e.g. NMF.log

Return Value:

1. Stdout.txt: the"stdout" text output from running the program.
2. output.file: as defined in the input parameters.
3. W.gct: W matrix
4. W_s.gct: W matrix sorted by factor
5. H.gct: H matrix
6. H_s.gct: H matrix sorted by factor
7. V.gct: V matrix (original input data matrix)
8. V_s.gct: V matrix sorted by factor
9. WH.gct: product WH
10. WH_s.gct: product WH sorted by factor

Platform dependencies:

Task type:	Projection
CPU type:	any
OS:	any
Java JVM level:	n/a
Language:	Perl