Perturbation biology tutorial

For questions:
xli27@mdanderson.org
akorkut@mdanderson.org
chris@sanderlab.org
sonursumer@gmail.com

This tutorial summarizes the steps for generating the network models using the BP-based network inference algorithm and perdicting steady states of network models under in silico perturbations using ODE simulations. The tutorial covers the computational components of the perturbation biology method. Here we generate network models and perform prediction using

-Data: Perturbation response data in SkMel133
-Algorithm: BP-guided decimation and DLSODE-based simulation
-Prior information generated by the PERA.

First you need a gfortran compiler (Download the binaries):
https://gcc.gnu.org/wiki/GFortranBinaries#MacOS

The source code is provided in
https://github.com/korkutlab/perbio

The results are published in
Korkut et al, 2015, eLIFE
http://elifesciences.org/content/4/e04640v1

Molinelli et al, 2013, Plos Comp Bio
http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003290

For a tutorial of PERA algorithm to generate prior informations:
https://bitbucket.org/armish/bp_prior

I assume you downloaded the source code and the data from
https://github.com/korkutlab/perbio

Let's start.

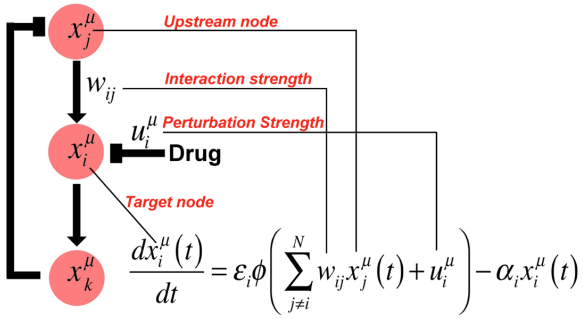
Contents

0. Background

0.1. Network models

Our network models quantitatively link external perturbations to proteomic entities and phenotypic outcomes (i.e. apoptosis, cell viability) that influence each other

Input: response to systematic perturbations
Nodes: quantitative measure of molecular & phenotypic species
Edges: influence of one node on the time derivative of the other



0.2. Inference problem.

The inference problem involves finding the optimum {wij} such that the model equations:

1. Best reproduce a training set of experimental data
2. Have predictive power beyond the training set

1. Compile the source code

You will need the source code and the variable description files (https://github.com/korkutlab/perbio/tree/master/decima_bp/src)

decima_bp.f
common.main
bp2.main

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c    <Inference of quantitative & data driven networks in biological systems using
c    BP-based Decimation Algorithm with and without prior information>
c    Copyright (C) 2015  MSKCC, Authors: Chris Sander, Anil Korkut
c
c
c    This program is free software: you can redistribute it and/or modify
c    it under the terms of the GNU General Public License as published by
c    the Free Software Foundation, either version 3 of the License, or
c    (at your option) any later version.
c
c    This program is distributed in the hope that it will be useful,
c    but WITHOUT ANY WARRANTY; without even the implied warranty of
c    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
c    GNU General Public License for more details.
c
c    You should have received a copy of the GNU General Public License
c    along with this program.  If not, see <http://www.gnu.org/licenses/>.
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c
c       decimation/BP algorithm for network inference from perturbation data
c
c       Hopfield network models and perturbation
c       model XDATA'=beta*phi(sigma(Jij*Xj)+Ui)-apha*XDATA
c
c       last updated 01/11 a.korkut/ sander lab @ mskcc akorkut@cbio.mskcc.org
c
c
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
       program bp decimation
```

Box 1. The header of the BP-based network inference source code.

Compile with the command (for the gfortran compiler https://gcc.gnu.org/wiki/GFortranBinaries):

>>> **sh install.sh**

The executable will be in the directory ../bin.

2. Prepare the input files

2.1. Options of the program

To list the options of the program, you can directly run the program:

>>> **../bin/decima_bp**

To get help information, you need to run the command:

>>> **../bin/decima_bp -h**

This will give you an example of input.

2.2. run.inp

Format: 2 columns delimitted by white spaces or tab; the rest of columns are not used; the texts following # are comments

The input file contains the parameters necessary for running the BP-based decimation code. You can copy-paste the below input parameter set and generate your input parameter set according to your wish.

```
#################################################################
n_expts         89      # Number of perturbation conditions
n_nodes         99      # Total number of nodes included in calculations
n_wvals         11      # Number of Wij values allowed
```

```
wmax            1.00E+00        # Range of Wij. If 1 then -1 =< Wij =< 1
threshold       1.00E-06        # Convergence criteria in BP iterations
lambda          5.00    # Weight of the model complexity term in the cost function
beta            2.00    # Weight of the SSE in the cost function
n_priors        154     # Number of priors used
n_models        80      # Number of models to be generated
file_node_index         node_index.txt          # File name of the node list
file_expt_index         expt_index.txt          # File name of the experiment list
file_prio               prio.txt                # File name of the prior information
file_data               data.txt                # File name of the drug response data
file_pert               pert.txt                # File name of the perturbation data
############################################################
```

```
n_expts         89
n_nodes         99
n_wvals         11
wmax    1.00E+00
threshold       1.00E-06
lambda 5.00
beta    2.00
n_priors        154
n_models        80
file_node_index         node_index.txt
file_expt_index         expt_index.txt
file_prio       prio.txt
file_data       data.txt
file_pert       pert.txt
```

Box 2. A sample run.inp file

2.3. node_index.txt

The list of the proteomic, phenotypic, and activity nodes, their inclusion in the models and node types.

Format: 3 columns delimitted by white spaces or tab; the rest of columns are not used

Column 1: Names of nodes (preferably gene name or a protein name)
- 100 Characters MAX
- PTM type (phoshorylation/cleavege etc)
- PTMsite
- For example, AKTpS473 (protein name/p/site)

Column 2: Inclusion status
- 1: Include the node to the models in the inference
- 0: Do not include

Column 3: Node types
- 1: Proteomic node
- 2: Phenotypic node
- 3: Activity (drug coupling) node

MAPKpT202	1	1
MEKpS217	1	1
S6	1	1
PAI-1	1	1
AKTpS473	1	1
AMPKpT172	1	1
b-Catenin	1	1
BIM	1	1
Caveolin	1	1

Box 3. Sample lines from the node_index.txt file

2.4. expt_index.txt

List of experiments (perturbation, drug name, dose).

Format: 2 columns delimitted by white spaces or tab; the rest of columns are not used

Column 1: Names of the experimental conditions (perturbations/drugs/doses etc)
100 Characters MAX

Column 2: Inclusion status
1: Include the experiment in modeling
0: Do not include

901|1.5	1
901|1.5,HN|6	1
901|1.5,NT|3	1
901|1.5,P6|20	1
901|1.5,PLX|60	1
901|1.5,RO|3.5	1
901|1.5,SR|2.4	1

Box 4. Sample lines from exp_index.txt

2.5. data.txt

The drug response data (x).

Format: tab (or white spaces) delimited file of data matrix

Experimental response map. Each readout is normalized wrt no drug control and in log2.
Columns: Nodes (measurements on proteiomic, phenotypic and activity nodes)
Rows: Experimental conditions (samples)

2.6. pert.txt

The impact of perturbations (u) on each node.

Format: tab (or white spaces) delimited file of perturbation matrix

The strengths of the perturbations acting on node
Columns: Nodes (perturbations on proteiomic, phenotypic and activity nodes)
Rows: Experimental conditions (samples)

2.7. prio.txt

Prior information generated by PERA or others.

Format: tab (or white spaces) delimited file

Column 1: Node1

Column 2: Distance

Distance between Node1 and Node2 in the pathway obtained from database
It is not used in the decimation code

Column 3: Node2

Column 4: Priv

Specifying if Node1 is activating or inhibiting Node2
if Priv = 1, prior activating edge;
if Priv = -1, prior inhibitory edge;
if Priv = 0, generic prior for interaction (no preference for activation/deactivation).

Column 5: Support

Specifying if the prior is supported by other evidence
The actual prize is scaled by the Support value (float).
For example, if Support = 0, the prior is not used;
if Support = 1.0, no scaling on the prize.

aSTAT3	1	STAT5pY694	0	1
bRAF	1	MEKpS217	1	1
aSRC	1	STAT3pY705	1	1
b-Catenin	1	EGFR	0	1
Fibronectin	1	SRCpY416	0	1
SRCpY416	1	AKTpS473	1	1
SRCpY416	1	PI3Kp85	1	1
EGFR	1	mTORpS244	0	1

Box 5. Sample lines from the prio.txt (prior model) file.

3. Execute the code

Run the program with the command

>>> **../bin/decima_bp –i run.inp >& run.out &**

Generates an ensemble of model solutions.

4. Output files

4.1 decimation_output.txt: Information of edge fixed at each decimation step in each round of the model generating process. For each fixed edge, the following information are displayed: model index (round), decimation step, name of the source node, name of the target node, the fixed Wij value, marginal probabilities of each Wij value of the edge.

4.2 model_X.txt: Three blocks: 1) Wij matrix for model solution X**.** The NXN matrix for each model solution with values Wij={-1,-0.8….0,...0.8,1.0}. i &j are the node indices. 2) Values of alpha for each node. 3) Values of epsilon for each node.

The ratio of alpha over epsilon are inferred based on the the dynamic range of each proteomic measurement sampled in the biological dataset. In calculations we take alpha = 1, therefore the epsilon for each mode equals to 1/(alpha/epsilon) for each node.

5. Average model

Average model is the average of the Wij values over all (or selected) number of model solutions. The program does not output the average model, you can compute yourself. In the average models, the Wij values are normalized with respect to the maximum value of the average Wij (<wij>) over all i & j's in all model solutions. If the max <Wij> is 0.93 as in the SkMel133 models, an average wij that equals to 0.186 will be normalized as 0.186/0.93=0.2**.** In the average model, **t**he strong edges (e.g., <Wij> >0.8) are frequently captured in model solutions and have high Wij values. On the other hand weak edges (~0.20) have low interaction strengths and/or represented in the small fraction of the models. In Korkut et al. we reported the <wij> with the cut-off value of 0.20 (0.186 before normalization with respect to max Wij). Feel free to use any value you like if you want to visualize weaker or stronger interactions. I do not recommend using a very low cut-off as you may start analyzing noise. **Most importantly, remember that the average model is not an executable model and it is only for visualisation purposes**.

6. Execute models w/ in silico perturbations

Network models are executed with specific in silico perturbations until all system variables {xi} reach steady state. The perturbations acting on node, i, are exerted as real-valued, {ui}, vectors in model Equation 1.

Equation 1. Network model ODEs

$$\frac{dx_i^{\mu}(t)}{dt} = \varepsilon_i \phi\left(\sum_{j\neq i}^{N} w_{ij} x_j^{\mu}(t) + u_i^{\mu}\right) - \alpha_i x_i^{\mu}(t)\,, \tag{1a}$$

$$\phi(z) = \tanh(z)\,. \tag{1b}$$

See Korkut et al, 2015 for details of the equation. We used the DLSODE integration method (ODEPACK) (Hindmarsh, 1993) settings , MF = 10, ATOL = 1e-10, RTOL = 1e-20).

6.1 Compile the souce code

You will need the source code and the variable description files (https://github.com/korkutlab/perbio/tree/master/sim_ode/src)

sim_ode.f90
dlsode.f

Compile with the command (for the gfortran compiler https://gcc.gnu.org/wiki/GFortranBinaries):
>>> **sh install.sh**

The executable will be in the directory ../bin.

6.2 Prepare the input files

To list the options of the simulation program, you can directly run the program:

>>> **../bin/sim_ode**

To get help information, you need to run the command:

>>> **../bin/sim_ode -h**

This will give you an example of input.

6.2.1. run.inp

Format: 2 columns delimitted by white spaces or tab; the rest of columns are not used; the texts following # are comments

The input file contains the parameters necessary for running the simulation code.

```
############################################################
n_nodes         99                  # Total number of nodes included in calculations
file_pert       pert_insilico.txt   # File name of the in silico perturbation setup
model_file_prefix       model_      # Prefix of file name of the models
model_file_suffix       .txt        # Suffix of file name of the models
model_id_beg            1           # The first index of the models
model_id_end            80          # The last index of the models
############################################################
```

```
n_nodes         99
file_pert       pert_insilico.txt
model_file_prefix       model_
model_file_suffix       .txt
model_id_beg    1
model_id_end    80
```

Box 6. A sample run.inp file

6.2.2. pert_insilico.txt

The setup of in silico perturbation conditions.

Format: tab (or white spaces) delimited file

Column 1: Number of nodes perturbed (N)

Column 2 to Column N+1: Index of nodes perturbed

Column N+2 to Column 2N+1: Inhibited percentage of activity for each node perturbed

Column 2N+2: Name of the in silico perturbation, 100 Characters MAX

```
2       89      90      0.45    0.83    aMEK12_aAKT
2       94      90      0.35    0.83    aBRAFm_aAKT
2       91      90      0.08    0.83    aHDAC_aAKT
2       92      90      0.25    0.83    aMDM2_aAKT
2       93      90      0.30    0.83    aJAK_aAKT
2       95      90      0.48    0.83    aPKC_aAKT
2       96      90      0.15    0.83    aSTAT3_aAKT
2       97      90      0.79    0.83    amTOR_aAKT
2       98      90      0.78    0.83    aPI3K_aAKT
2       99      90      0.33    0.83    aCDK_aAKT
```

Box 7. A sample pert_insilico.txt file

The u vector represents the in silico perturbations. We choose a u vector to quantify the strength of the vectors. For example you can calculate the strength of the u vector for inhibiting the activity node by %50 (i.e., have a value of -1 in log2 space).

6.2.3. Execute the code
Run the program with the command
>>> **../bin/sim_ode –i run.inp >& run.out &**
Predicts steady states of network models under each in silico perturbation.

6.2.4. Output files
predict_X.txt:
Information of steady states of each netwrok model.
Column 1: Index of network model
Column 2: ISTATE value returned after calling the ODE solver DLSODE
Column 3 to the end: Steady-state values of x's

ISTATE indicates convergence status of the ODE simulation. If ISTATE = -1, a steady state reached at the end of the simulation; etherwise, the result was not converged.

6.3. Reaching the steady state. In simulations you must check whether the trajectory reaches to steady state. In our experience, it does in most cases. However, in few cases, an oscillatory behaviour is observed.

6.4. Uniform scaling to avoid systematic errors due to network discontinuity (OPTIONAL):
A uniform scaling is applied to simulation results to avoid the systematic errors in models that arise due to lack of some edges in individual model solutions. Missing edges in individual solutions lead to disruption of the flow in the network models and cause an artificial down-prediction of drug responses.
Solution: A uniform scaling of the data that takes into account the systematic down-prediction.
Procedure:
-Build models with data.
-Simulate the models to recapitulate the underlying data.
-Compute the overall standard deviation of the "recapitulated/simulated data".
-Generate the uniform scaling factor to correct for the systematic down-prediction. Scaling factor is the ratio of the standard deviation of the actual data and the simulated data.

For the SkMel133 data, the scaling factor is computed as 1.4. As default, you may use the same value or re-compute a new scaling factor based on your data.

Important Note: This optional step of uniform scaling does not alter the rank of the predictions (therefore has no effect on the biological interpretation). It also does not alter the correlation coefficient in cross validations s in Figure 3 of Korkut et al.

7. Cross validation (optional)

To test the predictive power of the network models, you can follow a cross-validation scheme as in figure 3 of Korkut et al, 2015 paper (http://elifesciences.org/content/4/e04640v1/figure3). You can test the impact of different variables (prior information, choice of optimization parameters, data sets etc) on the predictive power. I suggest to use the same uniform scaling as in 6.4. Alternatively, you can generate a separate scaling factor for each calculation or simply do not apply any uniform scaling. Such scaling will not affect the resulting cross correlations for assessment of the predictive power. In our case, we used the same scaling factor computed in 6.4.

```
911|1.5          1
911|1.5,HN|6     0
911|1.5,NT|3     0
911|1.5,P6|21    0
911|1.5,PLX|61   0
911|1.5,RO|3.5   0
911|1.5,SR|2.4   0
911|1.5,ST|1.6   0
911|1.5,Tm|1.3   0
911|1.5,ZS|1.6   0
911|3            1
AK|11            1
AK|5             1
AK|5,911|1.5     1
AK|5,HN|6        1
AK|5,NT|3        1
AK|5,P6|21       1
AK|5,PLX|61      1
AK|5,RO|3.5      1
AK|5,SR|2.4      1
AK|5,ST|1.6      1
AK|5,Tm|1.3      1
AK|5,ZS|1.6      1
HN|12            1
HN|6             1
HN|6,NT|3        1
HN|6,RO|3.5      1
HN|6,SR|2.4      1
```

Box 8. The expt_index.txt for cross-validation with 901 (MEKi) condition.