Accurate reconstruction of gene regulatory networks using optPANDA

Author: Marouen Ben Guebila

Introduction

PANDA [1] reconstructs gene regulatory networks using a message passing algorithm that optimizes the agreement between three data sources: Transcription Factor (TF)-TF Protein-Protein Interaction (PPI) network, gene expression data, and TF to gene binding data from Position Weight Matrices (PWMs). optPANDA increases the accuracy of the reconstruction process through identifying three parameter sets for three types of regulatory networks. The parameter identification process uses Bayesian optimisation through the whole solution space and will not be discussed in this tutorial.

The three types of regulation considered in optPANDA are binding as defined by ChIP-seq peaks, gene differential expression upon TF knockdown, and functional binding which refers to the intersection of both i.e., edges when TFs bind in the promoter region of the gene and induce differential expression of the target gene when the TF is knocked down.

As opposed to PANDA that computes regulation networks at large, optPANDA provides a flexible tool for the reconstruction of gene regulatory networks with a precise definition of a TF to gene edge. Wether it is binding, differential expression upon knock down, or functional binding, optPANDA uses differential parameter sets to compute the regulatory network of interest.

Downloading the data

The files needed for this tutorial are hosted in Amazon Web Servies (AWS) S3 bucket, so we will use their client (AWS CLI) to do the download. Alternatively, we can use the MATLAB function webread to get individual files directly in our working space. You can get AWS CLI here: https://aws.amazon.com/cli/

```
%setenv('LD_LIBRARY_PATH',''); %uncomment when MATLAB throws an error with AWS CLI cal
system('aws s3 cp s3://granddb/optPANDA/ppi . --recursive')
system('aws s3 cp s3://granddb/optPANDA/motifs/regMatPval1e3.txt .')
system('aws s3 cp s3://granddb/optPANDA/expression/Hugo_exp1_lcl.txt .')
system('aws s3 cp s3://granddb/optPANDA/motifs/Hugo_motifCellLine.txt .')
system('aws s3 cp s3://granddb/optPANDA/groundTruths/LCL_CHIP_SEQ_REMAP_ALL.txt .')
system('aws s3 cp s3://granddb/optPANDA/groundTruths/CUSANOVICH_TFKO_PVAL.txt .')
system('aws s3 cp s3://granddb/optPANDA/groundTruths/directBinding_LCL.csv .')
```

Generation of TF-gene prior and network reconstruction

For this tutorial, we will need netZooM > 0.4.1. Please refer to the package webpage for installation instruction: https://github.com/netZoo/netZooM.

Baseline PANDA network

The generation of a baseline PANDA network using optPANDA requires the following parameters:

```
precomputed = 0;  % Use presaved results
motifWeight = 0;  % When addCorr==1. Weight of TF-Gene coexpression when added to the T
motifcutOff = 0;  % Cutoff values in the TF-Gene coexpression, when addCorr==1
```

```
respWeight  = 0.5; % Weight of responsability matrix, i.e., PPI(TFs,TFs)*regulation pri
                   % the availability matrix i.e., regulation prior(TFs,Genes)*Coexpres
                   %  takes a weight of 1-respWeight
alpha       = 0.1; % Learning rate
addCorr     = 0; % augment the TF-gene regulation prior by adding TF-gene coexpression
absCoex     = 0; % Takes the absolute value of the coexpression matrix
incCoverage = 0; % Increases TF coverage by adding rows of zeros in TF-gene regulation
qpval       = 0; % Use p-value or corrected p-value for TF-gene binding motif
thresh      = NaN; % P-value threshold to assign binding in TF-gene regulation prior
oldMotif    = 1;   % Use a TF-gene regulation prior that was generated previously in [2
bridgingProteins = NaN; % Computation of higher order PPI interaction network through I
addChip     = 0; % Adds chip-seq in the TF-gene regulation prior
ctrl        = 0; % Dummy variable used in the optimisation process to compute the perfo
motif_fil   = '';% A structure containing precomputed results
```

optPANDA is a parameter-aware version of PANDA that expands its flexibility and allows feature engineering to improve accuracy.

Using the parameter set, we will reconstruct a new prior data that will direct the network reconstruction process into a specific type of network i.e., binding networks, differential expression upon TF knock-down, and functional bidning. First, let's set the gene expression file for LCL cell lines.

```
exp_file='Hugo_exp1_lcl.txt';
```

Then we generate a new TF-gene regulation prior, we use the following function, which can run on a 8Gb RAM laptop:

```
motif_file=generateOptPrior(exp_file,incCoverage,bridgingProteins,oldMotif,...
    qpval,precomputed,motif_fil,motifWeight,motifcutOff,addCorr,absCoex,thresh,...
    addChip,ctrl);
% the new motif is 'Hugo_exp1_lcl_Hugo_motifCellLine_MW0_MC0_AC0_ABS0_THRNaN_OM1_IC0_QP
```

Note that a new TF-gene regulation prior file has been generated. The output network can be obtained using the regular PANDA function using the new regulation prior, the expression data, and the PPI data.

```
% Generate new network
lib_path  ='..';
panda_out ='';
save_temp ='';
save_pairs=0;
ppi_file  = 'ppi2015_freezeCellLine.txt'; %corresponding to incCoverage == 0
disp('Computing baseline optPANDA network!');
baselineNet=panda_run(lib_path, exp_file, motif_file, ppi_file, panda_out, save_temp,..
        alpha, save_pairs, 'legacy',respWeight, absCoex);
```

Obviously, this is the same as generating a PANDA network but it gives a sense of the parameters corresponding to the baseline PANDA network.

Now, we will modify the parameters to direct the generation of the network depending of the definition of a regulatory edge between TF and gene.

ChIP-seq binding

The parameter set corresponding to an output bindng network as learned from ChIP-seq data were identified using Bayesian optimisation. They corrspond to the following values:

```
precomputed = 0;
motifWeight = NaN;
motifcutOff = NaN;
respWeight  = 0.90938;
alpha       = 0.073931;
addCorr     = 0;
absCoex     = 1;
incCoverage = 1;
qpval       = 0;
thresh      = 0.0008862;
oldMotif    = 0;
bridgingProteins = 0;
addChip     = 0;
ctrl        = 0;
motif_fil   = '';
```

We start by generating a regulation prior optimized for generating binding networks, using those parameters and LCL gene expression.

```
[motif_file,allTFNameChip,GeneNames]=generateOptPrior(exp_file,incCoverage,bridgingProt
    qpval,precomputed,motif_fil,motifWeight,motifcutOff,addCorr,absCoex,thresh,...
    addChip,ctrl);
% the new motif is 'Hugo_exp1_lcl_regMatPval1e3_MWNaN_MCNaN_AC0_ABS1_THR0.0008862_OM0_I
```

Note that a new TF-gene regulation prior file has been generated. The output network can be obtained using the regular PANDA function using the new regulation prior, the expression data, and the PPI data.

```
% Generate new network
lib_path  ='..';
panda_out ='';
save_temp ='';
save_pairs=0;
ppi_file  ='ppi_complete.txt';%corresponding to bridgingProteins == 0
disp('Computing ChIP-seq optPANDA network!');
chipseqNet=panda_run(lib_path, exp_file, motif_file, ppi_file, panda_out, save_temp,...
        alpha, save_pairs, 'legacy',respWeight, absCoex);
```

Differential expression upon TF knockdown

The following parameter set outputs a network where an edge between a TF and a gene represents the differential expression of the gene when the TF is knocked down. This network is extremely useful as it represents transcriptional control that shapes cell identity and disease progression. However, it accounts for direct and indirect effects.

```
precomputed = 0;
motifWeight = 0.36819;
motifcutOff = 0.37092;
respWeight  = 0.32969;
alpha       = 0.099458;
```

```
addCorr     = 1;
absCoex     = 1;
incCoverage = 1;
qpval       = 0;
thresh      = 0.00073311;
oldMotif    = 0;
bridgingProteins = 3;
addChip     = 0;
ctrl        = 0;
motif_fil   = '';
```

We start by generating a regulation prior optimized for generating differential expression networks, using these parameters and LCL gene expression.

```
[motif_file,allTFNameDiff,GeneNames]=generateOptPrior(exp_file,incCoverage,bridgingProt
    qpval,precomputed,motif_fil,motifWeight,motifcutOff,addCorr,absCoex,thresh,...
    addChip,ctrl);
% the new motif is
% 'Hugo_exp1_lcl_regMatPval1e3_MW0.36819_MC0.37092_AC1_ABS1_THR0.00073311_OM0_IC1_QP0_P
```

Note that a new TF-gene regulation prior file has been generated. The output network can be obtained using the regular PANDA function using the new regulation prior, the expression data, and the PPI data.

```
% Generate new network
lib_path  ='..';
panda_out ='';
save_temp ='';
save_pairs=0;
ppi_file  ='ppi_complete_bind.txt';%corresponding to bridgingProteins == 3
disp('Computing differential expression optPANDA network!');
tfKoNet=panda_run(lib_path, exp_file, motif_file, ppi_file, panda_out, save_temp,...
        alpha, save_pairs, 'legacy',respWeight, absCoex);
```

Functional binding

Functional binding networks have an edge between a TF and a gene if the TF binds in the promoter region of the gene as assessed by ChIP-seq and if the gene exhibits differential expression when the TF is knocked down. In other words, it is the intersection of the two previous networks. The parameter set of optPANDA that allows to identify functional binding networks are:

```
precomputed = 0;
motifWeight = 0.18413;
motifcutOff = 0.97812;
respWeight  = 0.8706;
alpha       = 0.01986;
addCorr     = 2;
absCoex     = 1;
incCoverage = 1;
qpval       = 0;
thresh      = 0.00098689;
oldMotif    = 0;
bridgingProteins = 0;
addChip     = 0;
```

```
ctrl        = 2;
motif_fil   = '';
```

We start by generating a regulation prior optimized for generating functional binding networks, using these parameters and LCL gene expression.

```
[motif_file,allTFNameFunbind,GeneNames]=generateOptPrior(exp_file,incCoverage,bridgingP
    qpval,precomputed,motif_fil,motifWeight,motifcutOff,addCorr,absCoex,thresh,...
    addChip,ctrl);
% the new motif is '
```

Note that a new TF-gene regulation prior file has been generated. The output network can be obtained using the regular PANDA function using the new regulation prior, the expression data, and the PPI data.

```
% Generate new network
lib_path  ='..';
panda_out ='';
save_temp ='';
save_pairs=0;
ppi_file  ='ppi_complete.txt';%corresponding to bridgingProteins == 0
disp('Computing functional binding optPANDA network!');
funBindNet=panda_run(lib_path, exp_file, motif_file, ppi_file, panda_out, save_temp,...
        alpha, save_pairs, 'legacy',respWeight, absCoex);
```

Validation of optPANDA networks

To valide our networks, we will compare them against three types of ground truth networks. ChIP-seq binding network of LCL cell lines taken from Remap 2018 [3], differential expression upon TF knock down [4], and functional binding networks which are the intersection of two previous networks.

Since the generation of networks can be computationally intensive, we can use precomputed results. There is no need to download the data here since MATLAB

has functionalities that allow to read data from the cloud directly into the workspace.

```
precomputedNetworks=0;% set to 1 to use precomputed networks
if precomputedNetworks
    fds1_gt = fileDatastore('s3://granddb/optPANDA/results/Benchmark1_results_bl.mat',
    fds1 = fileDatastore('s3://granddb/optPANDA/results/Benchmark1_results_opt.mat', "R
    fds2 = fileDatastore('s3://granddb/optPANDA/results/Benchmark2_results_opt.mat', "R
    fds3 = fileDatastore('s3://granddb/optPANDA/results/Benchmark3_results_opt.mat', "R
    % Read from a single MAT file.
    data = read(fds1_gt);
    baselineNet = data.output.AgNet;
    data = read(fds1);
    chipseqNet = data.output.AgNet;
    data = read(fds2);
    tfKoNet = data.output.AgNet;
    data = read(fds3);
    funBindNet = data.output.AgNet;
    % read gene names
    expTbl   = readtable(exp_file,'FileType','text');
    Exp = expTbl{:,2:end};
```

```
    GeneNames = expTbl{:,1};
    % read TF names
    ppi_file  ='ppi_complete.txt';
    [TF1, TF2, weight] = textread(ppi_file, '%s%s%f');
    allTFNameFunbind  = unique(TF1);
    allTFNameChip=allTFNameFunbind;
    ppi_file  ='ppi_complete_bind.txt';
    [TF1, TF2, weight] = textread(ppi_file, '%s%s%f');
    allTFNameDiff  = unique(TF1);
    ppi_file  = 'ppi2015_freezeCellLine.txt';
    [TF1, TF2, weight] = textread(ppi_file, '%s%s%f');
    allTFNameBaseline  = unique(TF1);
end
```

ChIP-seq network

We will start by validating the optPANDA network against ChIP-seq

```
[gtChip]=readGt('LCL_CHIP_SEQ_REMAP_ALL.txt');
gtTFNamesChip = gtChip{:,1};
```

We call validate network function

```
[ROCoptChip,gtNet,AgNetVal,PRoptChip,interTFs]=validateNetwork(chipseqNet,gtChip,gtTFNa
            allTFNameChip,GeneNames);
```

We compute the accuracy of the baseline PANDA network

```
[ROCbaseChip,gtNet,AgNetVal,PRbaseChip,interTFs]=validateNetwork(baselineNet,gtChip,gtT
            allTFNameBaseline,GeneNames);
```

Then we can plot ROC and PR curves

```
subplot(1,2,1)
hold on
plot(ROCoptChip.X,ROCoptChip.Y,'LineWidth',4)
plot(ROCbaseChip.X,ROCbaseChip.Y,'LineWidth',4)
title('ROC curve')
ylabel('True positive rate')
xlabel('False negative rate')
subplot(1,2,2)
hold on
plot(PRoptChip.X,PRoptChip.Y,'LineWidth',4)
plot(PRbaseChip.X,PRbaseChip.Y,'LineWidth',4)
legend('optPANDA','PANDA')
xlabel('Recall')
ylabel('Precision')
title('PR curve')
```

Differential expression network

```
[gtDiff]=readGt('CUSANOVICH_TFKO_PVAL.txt');
gtTFNamesDiff = gtDiff{:,1};
```

We call validate network function

```
[ROCoptDiff,gtNet,AgNetVal,PRoptDiff,interTFs]=validateNetwork(tfKoNet,gtDiff,gtTFNames
            allTFNameDiff,GeneNames);
```

We compute the accuracy of the baseline PANDA network

```
[ROCbaseDiff,gtNet,AgNetVal,PRbaseDiff,interTFs]=validateNetwork(baselineNet,gtDiff,gtT
            allTFNameBaseline,GeneNames);
```

Then we can plot ROC and PR curves

```
subplot(1,2,1)
hold on
plot(ROCoptDiff.X,ROCoptDiff.Y,'LineWidth',4)
plot(ROCbaseDiff.X,ROCbaseDiff.Y,'LineWidth',4)
title('ROC curve')
ylabel('True positive rate')
xlabel('False negative rate')
subplot(1,2,2)
hold on
plot(PRoptDiff.X,PRoptDiff.Y,'LineWidth',4)
plot(PRbaseDiff.X,PRbaseDiff.Y,'LineWidth',4)
legend('optPANDA','PANDA')
xlabel('Recall')
ylabel('Precision')
title('PR curve')
```

Functional binding network

```
[gtFunbind]=readGt('directBinding_LCL.csv');
gtTFNamesFunbind = gtFunbind{:,1};
```

We call validate network function

```
[ROCoptFunbind,gtNet,AgNetVal,PRoptFunbind,interTFs]=validateNetwork(funBindNet,gtFunbi
            allTFNameFunbind,GeneNames);
```

We compute the accuracy of the baseline PANDA network

```
[ROCbaseFunbind,gtNet,AgNetVal,PRbaseFunbind,interTFs]=validateNetwork(baselineNet,gtFu
            allTFNameBaseline,GeneNames);
```

Then we can plot ROC and PR curves

```
subplot(1,2,1)
hold on
plot(ROCoptFunbind.X,ROCoptFunbind.Y,'LineWidth',4)
plot(ROCbaseFunbind.X,ROCbaseFunbind.Y,'LineWidth',4)
title('ROC curve')
ylabel('True positive rate')
xlabel('False negative rate')
```

```
subplot(1,2,2)
hold on
plot(PRoptFunbind.X,PRoptFunbind.Y,'LineWidth',4)
plot(PRbaseFunbind.X,PRbaseFunbind.Y,'LineWidth',4)
legend('optPANDA','PANDA')
xlabel('Recall')
ylabel('Precision')
title('PR curve')
```

References

[1] Glass, Kimberly, et al. "Passing messages between biological networks to refine predicted interactions." *PloS one* 8.5 (2013).

[2] Lopes-Ramos, Camila M., et al. "Regulatory network changes between cell lines and their tissues of origin." *BMC genomics* 18.1 (2017): 723.

[3] Chèneby, Jeanne, et al. "ReMap 2020: a database of regulatory regions from an integrative analysis of Human and Arabidopsis DNA-binding sequencing experiments." *Nucleic acids research* 48.D1 (2020): D180-D188.

[4] Cusanovich, Darren A., et al. "The functional consequences of variation in transcription factor binding." *PLoS genetics* 10.3 (2014).