Jump3: documentation

Contact: Vân Anh Huynh-Thu, vahuynh@uliege.be

This is the documentation for the MATLAB[1] implementation of Jump3. The implementation is a research prototype and is provided "as is". No warranties or guarantees of any kind are given.

The Jump3 method is described in the following paper:
Huynh-Thu V. A. and Sanguinetti G. (2015) Combining tree-based and dynamical systems for the inference of gene regulatory networks. *Bioinformatics*, 31(10):1614-1622.

1 Run Jump3

First of all, add the directory 'jump3_code' into MATLAB's path:

```
path(path,'[path_to_the_directory/jump3_code')
```

A MATLAB file 'exampleData.mat' is provided for this tutorial.

```
load exampleData.mat
who
```

```
## Your variables are:
##
## data       genes      obsTimes
```

- `data` contains the expression data of 10 genes from 5 times series (which is actually the toy data related to Network 1 size-10, as described in the paper).
- `obsTimes` contains the observation time points for each time series of `data`.
- `genes` contains the names of the genes (for illustration purposes only).

Noise parameters

Jump3 has two noise parameters. These parameters must be put in a structure array (that we name here `noiseVar`) with two fields that *must* be named `sysNoise` and `obsNoise`.

- `noiseVar.sysNoise` is the variance of the intrinsic noise (σ^2 in the paper); one single value.
- `noiseVar.obsNoise` is the variance of the observation noise ($s^2_{i,k}$ in the paper); one value for each data value.

[1] http://www.mathworks.com

The optimal values of the noise parameters are difficult to identify (in most cases, trying to optimise these parameters would result in local optima). Some heuristic should thus be used to set the values of the noise parameters. For example, a dynamic noise can be used for the observation noise, assuming that the observation noise is higher for higher expression values:

```
noiseVar.obsNoise = cell(1,length(data));
for k=1:length(data)
    % A dynamic noise is used for the observation noise
    noiseVar.obsNoise{k} = (data{k}/10).^2;
    % Replace zero values with a small number to avoid numerical errors
    noiseVar.obsNoise{k}(noiseVar.obsNoise{k}==0) = 1e-6;
end
```

The intrinsic noise can be assumed to be much smaller than the observation noise:

```
noiseVar.sysNoise = 1e-4;
```

Run Jump3 with its default parameters

```
[w,exprMean,exprVar,promState,kinParams,kinParamsVar,trees] = ...
  jump3(data,obsTimes,noiseVar);
```

(This should take a few minutes.)

The algorithm outputs the following:

- `w`: weights of the regulatory links. `w(i,j)` is the weight of the link directed from gene j to gene i.
- `exprMean`: posterior mean of the expression of each gene i in each time series ($m_i(t)$ in the paper).
- `exprVar`: posterior variance of the expression of each gene i ($c_i(t,t)$ in the paper).
- `promState`: posterior state of the promoter of each gene i in each time series ($\mu_i(t)$ in the paper).
- `kinParams`: (optimised) values of the kinetic parameters A_i, b_i, and λ_i for each gene i.
- `kinParamsVar`: variances of the kinetic parameters A_i and b_i.
- `trees`: ensemble of jump trees predicting the promoter state of each gene i.

Restrict the candidate regulators to a subset of genes

```
% Indices of the genes that are used as candidate regulators
tfidx = [2 5 6 8 9];
w2 = jump3(data,obsTimes,noiseVar,tfidx);
```

In `w2`, the links that are directed from genes that are not candidate regulators have a score equal to 0.

Change the settings of the Extra-Trees

```
% Number of randomly chosen candidate regulators at each node of a tree
K = 3;

% Number of trees per ensemble
ntrees = 50;

% Run the method with these settings
w3 = jump3(data,obsTimes,noiseVar,1:10,K,ntrees);
```

Obtain more information

```
help jump3
```

2 Write the predictions

Get the predicted ranking of all the regulatory links

```
getLinkList(w)
```

The output will look like this:

```
## G8 G6 0.886894
## G3 G7 0.841794
## G9 G10 0.831725
## G1 G5 0.782503
## G1 G2 0.715546
## G7 G3 0.512323
## G5 G1 0.479648
## G1 G4 0.357726
## G2 G4 0.236863
## G1 G8 0.231089
## ...
```

Each line corresponds to a regulatory link. The first column shows the regulator, the second column shows the target gene, and the last column indicates the score of the link.
If the gene names are not provided, the i-th gene is named “Gi”.

Note that the ranking that is obtained will be slightly different from one run to another. This is due to the intrinsic randomness of the Extra-Trees. The variance of the ranking can be decreased by increasing the number of trees per ensemble.

Show only the links that are directed from the candidate regulators

```
getLinkList(w2,tfidx)
```

Show the first 5 links only

```
getLinkList(w2,tfidx,{},5)
```

Show the names of the genes

```
getLinkList(w2,tfidx,genes,5)
```

```
## FAM47B RXRA 0.983707
## ZNF618 RFC2 0.839112
## GPX4 CYB5R4 0.808789
## CYB5R4 NPY2R 0.674845
## CYB5R4 GPX4 0.673964
```

Write the predicted links in a file

```
getLinkList(w2,tfidx,genes,5,'ranking.txt')
```

Obtain more information

```
help getLinkList
```

3 Plot the modelling results

```
% Plot the results for the fifth gene in the first time series
TSidx = 1;
geneidx = 5;
plotPosteriors(data,obsTimes,exprMean,exprVar,promState,TSidx,geneidx)
```

The top plot shows the posterior mean expression of the gene (solid red line) with the confidence intervals (shaded red area), as well as the observed expression of the gene (black dots). The bottom plot shows the predicted state of the promoter of the gene.

Obtain more information

```
help plotPosteriors
```