

Dynamic Deterministic Effects Propagation Networks (DDEPN) - exemplary workflow

Christian Bender *

October 25, 2010

Abstract

Network modelling in systems biology has become an important tool to study molecular interactions, especially in the medical field like cancer research. The understanding of the interplay of proteins in cellular signalling is the basis for the development of novel drugs and therapies. Here, we set up a new method for the reconstruction of signalling networks from time course protein data after external perturbation. We show how to use protein expression and phosphorylation data measured on Reverse Phase Protein Arrays to infer a signalling network among proteins of the ERBB signalling cascade in a human breast cancer cell line.

Our method ([1]) models the signalling dynamics by a boolean signal propagation mechanism that defines a sequence of state transitions for a given network structure. A likelihood score is proposed that describes the probability of our measurements given a particular state transition matrix. We identify the optimal sequence of state transitions via a Hidden Markov Model. Network structure search is performed by a genetic algorithm that optimises the overall likelihood of a population of candidate networks. We test our method on simulated networks and data and show its increased performance in comparison to another Dynamical Bayesian Network approach. The reconstruction of a network in our real data results in several known signalling chains from the ERBB network, showing the validity and usefulness of our approach.

*German Cancer Research Center, Im Neuenheimer Feld 580, 69120 Heidelberg, Germany. eMail: c.bender@dkfz-heidelberg.de

1 QuickStart: using DDEPN for network inference on simulated data sets

This section shows an exemplary workflow to reconstruct a signalling network from simulated data. An analysis on real data can be performed analogously. Details on formatting the input data matrix as well as arguments for the function calls can be found in subsequent sections.

1.1 Simulating data

In this section we show how to generate artificial networks and data. A reference signalling network is simulated and used to sample measurements that incorporate the network structure.

First, simulate a network with 6 nodes and 2 distinct input stimuli.

```
> set.seed(12345)
> n <- 6
> signet <- signalnetwork(n = n, nstim = 2, cstim = 0, prop.inh = 0.2)
> net <- signet$phi
> stimuli <- signet$stimuli
> weights <- signet$weights
```

Second get intensities for each protein that are based on the network structure generated above.

```
> dataset <- makedata(net, stimuli, mu.bg = 1200, sd.bg = 400,
+   mu.signal.a = 2000, sd.signal.a = 1000)
> data <- dataset$datx
```

1.2 Running the Genetic Algorithm (GA)

Now run the genetic algorithm to reconstruct the network from the data generated above and compare it to the originally sampled network *net*.

```
> ret <- ddepn(data, phiorig = net, inference = "netga", maxiterations = 15,
+   p = 30, q = 0.3, m = 0.8, usebics = TRUE)
```

After the reconstruction, the generated network can be viewed as follows:

```
> plotresult(ret)
```

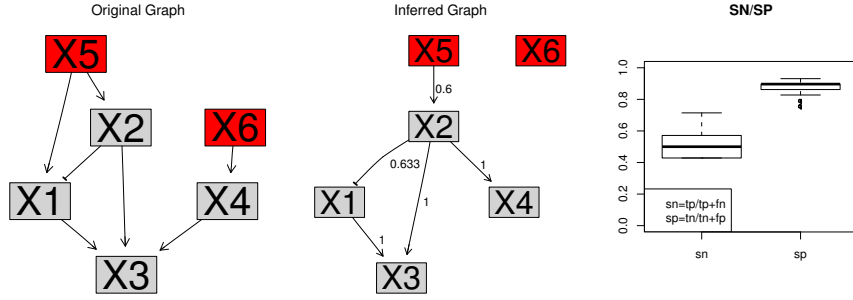


Figure 1: Result plot of the genetic algorithm. Left: the given graph; Middle: the inferred graph; Right: sensitivity/specificity plot for comparing the original and inferred graphs

1.3 Running Markov Chain Monte Carlo Sampling (MCMC)

An example run for MCMC sampling follows. Here, the package *multicore* is needed, since two parallel and independent MCMC runs are performed. If *multicore* is not available on the machine, just set *multicores=FALSE* to perform sampling for a single chain.

```
> B <- net
> B[B == 2] <- -1
> mi <- 5000
> burnin <- 1000
> if (require(multicore)) {
+   ret <- ddepn(data, phiorig = net, inference = "mcmc", maxiterations = mi,
+     burnin = burnin, usebics = FALSE, lambda = 0.01, B = B,
+     multicores = TRUE, cores = 2, priortype = "laplaceinhib")
+ }
```

After the sampling one can examine the sampling run:

```
> plotresult(ret$samplings[[1]])
```

The returned list *ret* contains two elements, another list with name *samplings* (*ret\$samplings*), which holds the different runs for the MCMC. In case of *multicores=FALSE*, only one run is performed and *ret\$samplings* holds only one element. Otherwise *cores* runs are performed independently in parallel, and *ret\$samplings* holds *cores* elements. The second element in *ret* with name *ltraces* is a matrix and holds the score traces of all runs, where each column corresponds to one trace. Output diagnostics can be produced using the R-package *coda*. See figure 3 for some example plots.

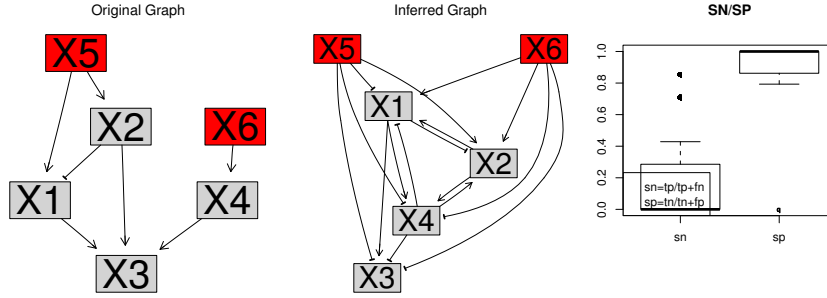


Figure 2: Result plot of MCMC sampling, analogous to figure 1.

```
> if (require(coda)) {
+   mcmc1 <- mcmc(data = ret$ltraces[-c(1:burnin, mi), 1])
+   mcmc2 <- mcmc(data = ret$ltraces[-c(1:burnin, mi), 2])
+   mcmc1 <- mcmc.list(mcmc1, mcmc2)
+   plot(mcmc1)
+   gelman.plot(mcmc1)
+ }
```

2 Notes on formatting constraints for the arguments of *ddepn*

There is only one necessary argument to the function call of *ddepn*: The data matrix *dat*. Optionally, a reference network *phiorig* and seed networks *phi* can be passed to *ddepn*. Each of these arguments is described briefly below.

Input data matrix *dat*

The data matrix contains all measurements for the nodes in the rows (e.g. proteins or genes), and the experiments and time points in the columns. There are some special needs on how to name the columns. We allow several treatments to be included in the data matrix. Examples for these treatments are stimulation by growth factors or inhibition by application of a drug. We refer generally to each of these as 'treatment'.

Each of the treatments will be included in the final network as a node, e.g. stimulation by the growth hormone *EGF* is added to the data matrix as row with name *EGF* (and thus appears as node *EGF* in the final network). The expression values for the stimuli nodes are set to 0 in each column of the

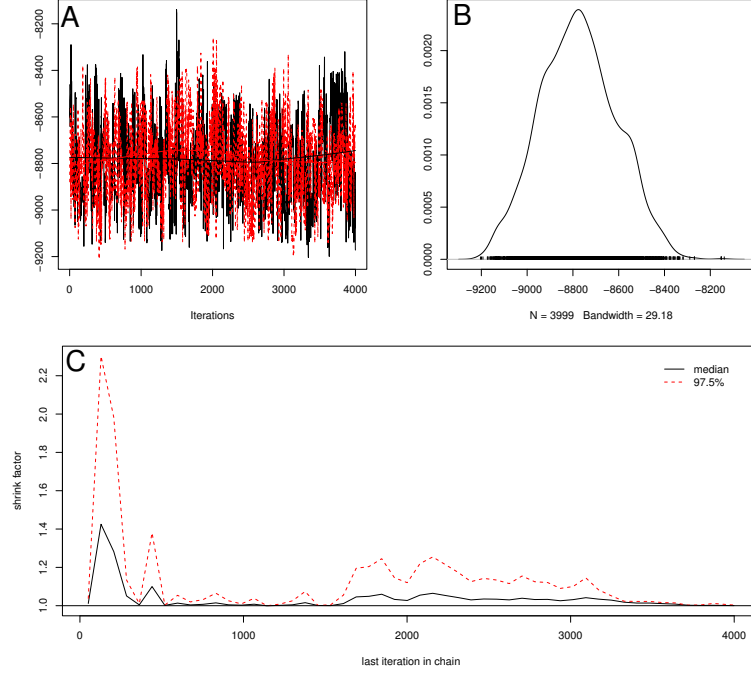


Figure 3: Using package coda for some MCMC output analysis. A: Traces for 2 MCMC runs; B: Distributionplot of the sampling; C: gelman.plot of two MCMC samplings

data matrix, but are never used in the algorithm and regarded as dummy values. Effects originating in these nodes are estimated in the reconstruction process.

To distinguish the different experimental conditions in the matrix, the columns of the data matrix have to be named in the format *treatment_time*, where treatment also can be a combination of several treatments, e.g. stimulation by *EGF* and simultaneous inhibition by a drug *X*. In this case, each stimulus has to be separated by an ampersand (&). The time point is separated from the stimuli via an underscore character (_), and can be on whatever scale (minutes, hours etc.). An example data matrix is shown below. In this table, the dummy rows for the treatments are already included (rows *EGF* and *X*). However, they are not mandatory as input to *ddepn* and, if missing, will be generated automatically, only requiring the correct labeling of the columns.

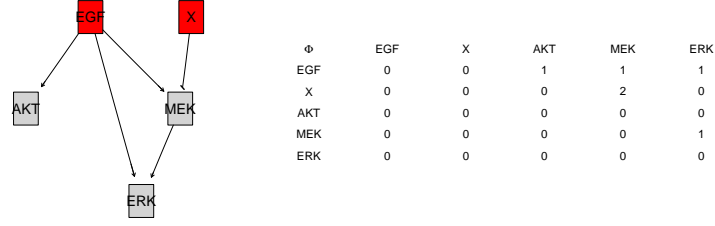


Figure 4: Example network. Left: graph representation, Right: adjacency matrix

	EGF_1	EGF_1	EGF_2	EGF_2	EGF&X_1	EGF&X_1	EGF&X_2	EGF&X_2
EGF	0	0	0	0	0	0	0	0
X	0	0	0	0	0	0	0	0
AKT	1.45	1.8	0.99	1.6	1.78	1.8	1.56	1.58
ERK	1.33	1.7	1.57	1.3	0.68	0.34	0.62	0.47
MEK	0.45	0.8	0.99	0.6	0.78	0.8	0.56	0.58

Reference network *phiorig*

If desired, a reference network *phiorig* can be given, used to compare the edges of the inferred network to it. The user *must* ensure that all treatments are included as nodes, since the inference will estimate effects from these. The format of the network must be an adjacency matrix, where each entry corresponds to an edge from the node specified by the rowname to the node specified by the column name. Two types of edges are allowed: 1 for activation, 2 for inhibition. 0 means no edge between the pair of nodes. An example network and corresponding adjacency matrix are shown in figure 4.

Seed networks *phi*

Both the GA and MCMC sampler require single or multiple seed networks. If not given, an unconnected network is used as seed for each individual in the GA population or the start networks for each MCMC run, respectively. However, the user can provide own seed networks using the argument *phi*. This can either be an adjacency matrix or a list of adjacency matrices. Again, the treatments *must* be included as nodes. If given a single adjacency matrix, it is used as seed network for each of the individuals in the population of networks during the genetic algorithm, or as independent seeds for parallel MCMC samplings. If given as list, its length must equal the number of individuals in the population in the GA (specified by the function argument *p*), or the number of independent runs in the MCMC sampler (specified by the argument *cores*). The format of the adjacency matrices is the same as for *phiorig*.

3 Prior knowledge inclusion

Currently, two methods for using biological prior knowledge are implemented. We refer to the first as *Laplace* prior ([2, 5]), and to the second as *ScaleFree* prior ([3]).

Laplace prior

The laplace prior penalises deviations of edges in the inferred network from prior edge probabilities, aquired from network databases (e.g. KEGG [4]). The package includes a snapshot of the KEGG database (containing signalling and disease pathways), downloaded in October 2010.

```
> data(kegggraphs)
> length(kegggraphs)
```

```
[1] 78
```

The list *kegggraphs* includes 78 elements, each of which has 3 members, a string *id* specifying the KEGG pathway id, a graphNEL object *g* and an adjacency matrix *phi*.

```
> names(kegggraphs)[1]
```

```
[1] "MAPK signaling pathway"
```

```
> kegggraphs[[1]]
```

```
$id
```

```
[1] "04010"
```

```
$g
```

```
A graphNEL graph with directed edges
```

```
Number of Nodes = 267
```

```
Number of Edges = 882
```

```
$phi
```

```
hsa:5923 hsa:5924 hsa:11072 hsa:11221 hsa:1843...
hsa:5923      0      0      0      0      0...
...
```

Each graphNEL object was converted to a detailed adjacency list phi (including inhibitions as entries with value 2) using the 'kegggraph.to.detailed.adjacency' function:

```
> kegggraph.to.detailed.adjacency(gR = kegggraphs[[1]]$g)
```

To obtain prior probabilities for each edge between all pairs of nodes present in *kegggraphs*, an approach of [5] is followed. The total number of node pairs M occurring in all reference networks is counted as well as the number of node pairs m that are connected via an edge in all reference networks. The prior probability matrix \mathcal{B} is defined as:

$$\mathcal{B} = \frac{m}{M},$$

where each entry is multiplied with -1 if the edge occurred as inhibiting edge in the KEGG pathways. This kind of prior matrix can be used together with the *laplaceinhib* prior: In the function call to *ddepn*, just pass the arguments B and λ and set *usebics*=*FALSE* and *priortype*="laplaceinhib" to use the laplace prior for the inference.

```
> ddepn(data, lambda = 0.01, B = B, usebics = FALSE, priortype = "laplaceinhib")
```

If no information is available on the type of the edges in the reference networks, one should use the *laplace* prior type. Here, all entries in \mathcal{B} are positive, and for the prior calculation, the edge type in the inferred network is ignored.

```
> ddepn(data, lambda = 0.01, B = B, usebics = FALSE, priortype = "laplace")
```

ScaleFree prior

According to [3] we set up a prior distribution that penalises high node degrees in the inferred network. The assumption is that for biological networks the degree of a node follows a power law distribution, i.e. the probability of seeing k nodes follows

$$P(k) \propto k^{-\gamma}.$$

We set up the prior distribution as described in [3]. To use the ScaleFree prior, just pass the arguments *gam* (the exponent γ), *it* (the number of permutations) and factor K to the function call of *ddepn*, and again set argument *usebics*=*FALSE*.


```
> ddepn(data, gam = 2.2, it = 500, K = 0.8, usebics = FALSE)
```

4 Use cases for GA and MCMC inference

This section shows the various types of calls to *ddepn* with all of the different settings (inference type, prior type).

Data generation:

```
> library(ddepn)
> set.seed(12345)
> n <- 6
> signet <- signalnetwork(n = n, nstim = 2, cstim = 0, prop.inh = 0.2)
> net <- signet$phi
> stimuli <- signet$stimuli
> weights <- signet$weights
> dataset <- makedata(net, stimuli, mu.bg = 1200, sd.bg = 400,
+   mu.signal.a = 2000, sd.signal.a = 1000)
> data <- dataset$datx
> mi = 15
> p = 30
> q = 0.3
> m = 0.8
> mi <- 1000
> burnin <- 100
> lambda <- 5
> lambda = 0.01
> Bpos <- net
> Bpos[Bpos == 2] <- 1
> B <- net
> B[B == 2] <- -1
> gam <- 2.2
> it <- 500
> K <- 0.8
```

GA, use BICs optimisation and no prior

```
> ret <- ddepn(data, phiorig = net, inference = "netga", maxiterations = mi,
+   p = p, q = q, m = m, usebics = TRUE)
```

GA, use laplaceinhib prior

```
> ret <- ddepn(data, phiorig = net, inference = "netga", maxiterations = mi,  
+   p = p, q = q, m = m, usebics = FALSE, lambda = lambda, B = B,  
+   priortype = "laplaceinhib")
```

GA, use laplace prior

```
> ret <- ddepn(data, phiorig = net, inference = "netga", maxiterations = mi,  
+   p = p, q = q, m = m, usebics = FALSE, lambda = lambda, B = Bpos,  
+   priortype = "laplace")
```

GA, use scalefree prior

```
> ret <- ddepn(data, phiorig = net, inference = "netga", maxiterations = mi,  
+   p = p, q = q, m = m, usebics = FALSE, gam = gam, it = it,  
+   K = K, priortype = "scalefree")
```

MCMC, use laplaceinhib prior

```
> ret <- ddepn(data, phiorig = net, inference = "mcmc", maxiterations = mi,  
+   burnin = burnin, usebics = FALSE, lambda = lambda, B = B,  
+   priortype = "laplaceinhib")
```

MCMC, use laplace prior

```
> ret <- ddepn(data, phiorig = net, inference = "mcmc", maxiterations = mi,  
+   burnin = burnin, usebics = FALSE, lambda = lambda, B = Bpos,  
+   priortype = "laplace")
```

MCMC, use scalefree prior

```
> ret <- ddepn(data, phiorig = net, inference = "mcmc", maxiterations = mi,  
+   burnin = burnin, usebics = FALSE, gam = gam, it = it, K = K,  
+   priortype = "scalefree")
```

Application to example data set from HCC1954 cell line

We show how we apply DDEPN to a data set from HCC1954 breast cancer cell line. Phosphorylation of 16 proteins was measured on Reverse Phase Protein Arrays. We performed time courses over 10 time points (0,4,8,12,16,20,30,40,50,60 minutes). Each measurement was replicated five times independently and for each of these five biological replicates, spotting was done in triplicate for each time point, leaving 15 replicates per time point and protein. Cells were stimulated with the EGFR ligand Epidermal Growth Factor (EGF) and the ERBB3 ligand Heregulin (HRG), both as single treatments as well as combined treatment (EGF&HRG). The data is included in the dataset:

```
> data(hcc1954)
```

First, change the column names. These contain, which of the replicates belong to the same biological replicate, which is not used in the inference in DDEPN.

```
> dat <- format_ddepn(hcc1954)
```

Now start a genetic algorithm to infer a signalling network between the 16 phosphoproteins (takes a while):

```
> mi = 1000
> p = 500
> q = 0.3
> m = 0.8
> ret <- ddepn(dat, phiorig = NULL, inference = "netga", maxiterations = mi,
+             p = p, q = q, m = m, usebics = TRUE)
```

Session Information

The version number of R and packages loaded for generating the vignette were:

- R version 2.11.0 Under development (unstable) (2010-01-22 r51017),
x86_64-unknown-linux-gnu

- Locale: LC_CTYPE=de_DE@euro, LC_NUMERIC=C,
LC_TIME=de_DE@euro, LC_COLLATE=de_DE@euro, LC_MONETARY=C,
LC_MESSAGES=de_DE@euro, LC_PAPER=de_DE@euro, LC_NAME=C,
LC_ADDRESS=C, LC_TELEPHONE=C, LC_MEASUREMENT=de_DE@euro,
LC_IDENTIFICATION=C
- Base packages: base, datasets, graphics, grDevices, methods, stats,
tools, utils

References

- [1] Christian Bender, Frauke Henjes, Holger Fröhlich, Stefan Wiemann, Ulrike Korf, and Tim Beißbarth. Dynamic deterministic effects propagation networks: learning signalling pathways from longitudinal protein array data. *Bioinformatics*, 26(18):i596–i602, Sep 2010.
- [2] Holger Fröhlich, Mark Fellmann, Holger Sülthmann, Annemarie Poustka, and Tim Beißbarth. Large scale statistical inference of signaling pathways from rnai and microarray data. *BMC Bioinformatics*, 8(11):386, 2007.
- [3] Takeshi Kamimura and Hidetoshi Shimodaira. A scale-free prior over graph structures for bayesian inference of gene networks. Online.
- [4] M. Kanehisa, M. Araki, S. Goto, M. Hattori, M. Hirakawa, M. Itoh, T. Katayama, S. Kawashima, S. Okuda, T. Tokimatsu, and Y. Yamanishi. KEGG for linking genomes to life and the environment. *Nucleic Acids Res.*, 36:D480 – D484, 2008.
- [5] Adriano V. Werhli and Dirk Husmeier. Reconstructing gene regulatory networks with bayesian networks by combining expression data with multiple sources of prior knowledge. *Stat Appl Genet Mol Biol*, 6:Article15, 2007.