A Computational Bayesian Approach to Ternary Network Estimation (ternarynet)

Matthew N. McCall and Anthony Almudevar and Harry A. Stern

April 15, 2025

Contents

1 Introduction

This document describes `ternarynet`, which implements a computational Bayesian algorithm to estimate a ternary network from perturbation data. We strong recommend reading the paper, *Fitting Boolean Networks from Steady State Perturbation Data* (Almudevar *et. al* 2011) before proceeding with this vignette.

2 Getting Started

First begin by downloading and installing the ternarynet package.

```
> library(ternarynet)
```

`parallelFit` function

The `ternarynet` package contains a parallel implementation of the replica exchange algorithm for fitting ternary network models. The `parallelFit` function takes the following arguments:

`experiment_set` data frame containing five columns:

- `i_exp` an experiment index: an integer from 0 to $N_{\text{exp}} - 1$, where N_{exp} is the number of experiments.
- `i_node` a node index: an integer from 0 to $N_{\text{node}} - 1$, where N_{node} is the number of nodes.
- `outcome` a value of -1, 0, or +1, denoting a particular outcome for that node in that experiment
- `value` a cost for obtaining that outcome. For instance, if the cost function is the Hamming distance, and the observed outcome is +1, the cost would be would be +2, +1, or 0 for an outcome of -1, 0, or +1, respectively.
- `is_perturbation` a Boolean value (or a value of 0/1) denoting whether this outcome is due to an applied perturbation or not.

`max_parents` maximum number of parents allowed for each node

`n_cycles` maximum number of Monte Carlo cycles

`n_write` number of times to write output during the run

`T_lo` T for lowest-temperature replica

`T_h` T for highest-temperature replica

`target_score` run will terminate if this is reached

`n_proc` number of replicas

`logfile` filename for log file

`n_thread` number of openMP threads to run per process; default=1

`init_parents` initial parents; randomize if null

`init_outcomes` inital outcomes; set to '.' if null

`exchange_interval` steps between replica exchanges; default=1000

`adjust_move_size_interval` steps between move size adjustments, default=7001

`max_states` maximum number of states to propagate to find a repetition; default=10

`callback` callback function, should take one integer argument (the replica number), used to call set.seed with different seed for each replica

The return value is a list with an element for each replica. Each element is itself a list of the best unnormalized score, normalized score (unnormalized score divided by product of number of nodes and number of experiments), list of parents for each node, and array describing the transition rule, giving the outcome of a node for each possible configuration of parent nodes.

Examples

The following shows a subset of the simple model regulatory network given in Example 1 of Reference 1 (nodes 1-4 only). There are four nodes and eight experiments (the first four rows of Table 4). The cost function for each possible outcome is the Hamming distance with the observed steady-state outcome, given a persistent perturbation. The output corresponds with the parents and transitions described on page 13 of Almudevar et al. (2011).

```
> library(ternarynet)
> i_exp <- as.integer(c(0,0,0, 0,0,0, 0,0,0, 0,0,0,
+                        1,1,1, 1,1,1, 1,1,1, 1,1,1,
```

```
+                         2,2,2, 2,2,2, 2,2,2, 2,2,2,
+                         3,3,3, 3,3,3, 3,3,3, 3,3,3,
+                         4,4,4, 4,4,4, 4,4,4, 4,4,4,
+                         5,5,5, 5,5,5, 5,5,5, 5,5,5,
+                         6,6,6, 6,6,6, 6,6,6, 6,6,6,
+                         7,7,7, 7,7,7, 7,7,7, 7,7,7))
> i_node <- as.integer(c(0,0,0, 1,1,1, 2,2,2, 3,3,3,
+                         0,0,0, 1,1,1, 2,2,2, 3,3,3,
+                         0,0,0, 1,1,1, 2,2,2, 3,3,3,
+                         0,0,0, 1,1,1, 2,2,2, 3,3,3,
+                         0,0,0, 1,1,1, 2,2,2, 3,3,3,
+                         0,0,0, 1,1,1, 2,2,2, 3,3,3,
+                         0,0,0, 1,1,1, 2,2,2, 3,3,3,
+                         0,0,0, 1,1,1, 2,2,2, 3,3,3))
> outcome <- as.integer(c(-1,0,1, -1,0,1, -1,0,1, -1,0,1,
+                         -1,0,1, -1,0,1, -1,0,1, -1,0,1,
+                         -1,0,1, -1,0,1, -1,0,1, -1,0,1,
+                         -1,0,1, -1,0,1, -1,0,1, -1,0,1,
+                         -1,0,1, -1,0,1, -1,0,1, -1,0,1,
+                         -1,0,1, -1,0,1, -1,0,1, -1,0,1,
+                         -1,0,1, -1,0,1, -1,0,1, -1,0,1,
+                         -1,0,1, -1,0,1, -1,0,1, -1,0,1))
> value <- c(0,1,2, 0,1,2, 0,1,2, 0,1,2,
+             2,1,0, 0,1,2, 0,1,2, 0,1,2,
+             2,1,0, 2,1,0, 0,1,2, 0,1,2,
+             2,1,0, 2,1,0, 2,1,0, 0,1,2,
+             2,1,0, 2,1,0, 2,1,0, 2,1,0,
+             0,1,2, 2,1,0, 2,1,0, 2,1,0,
+             0,1,2, 0,1,2, 2,1,0, 2,1,0,
+             0,1,2, 0,1,2, 0,1,2, 2,1,0)
> is_perturbation <-
+ c(TRUE,TRUE,TRUE,  FALSE,FALSE,FALSE, FALSE,FALSE,FALSE, FALSE,FALSE,FALSE,
+     FALSE,FALSE,FALSE,  TRUE,TRUE,TRUE, FALSE,FALSE,FALSE, FALSE,FALSE,FALSE,
+     FALSE,FALSE,FALSE,  FALSE,FALSE,FALSE, TRUE,TRUE,TRUE, FALSE,FALSE,FALSE,
+     FALSE,FALSE,FALSE,  FALSE,FALSE,FALSE, FALSE,FALSE,FALSE, TRUE,TRUE,TRUE,
+     TRUE,TRUE,TRUE,  FALSE,FALSE,FALSE, FALSE,FALSE,FALSE, FALSE,FALSE,FALSE,
+     FALSE,FALSE,FALSE,  TRUE,TRUE,TRUE, FALSE,FALSE,FALSE, FALSE,FALSE,FALSE,
+     FALSE,FALSE,FALSE,  FALSE,FALSE,FALSE, TRUE,TRUE,TRUE, FALSE,FALSE,FALSE,
+     FALSE,FALSE,FALSE,  FALSE,FALSE,FALSE, FALSE,FALSE,FALSE, TRUE,TRUE,TRUE)
> indata <- data.frame(i_exp,i_node,outcome,value,is_perturbation)
> results <- parallelFit(indata,
+                         max_parents=1,
```

```
+                          n_cycles=100000,
+                          n_write=10,
+                          T_lo=0.001,
+                          T_hi=2.0,
+                          target_score=0,
+                          n_proc=1,
+                          logfile='try.log')
> lowest_temp_results <- results[[1]]
> print('Unnormalized score:')

[1] "Unnormalized score:"

> print(lowest_temp_results$unnormalized_score)

[1] 0

> print('Normalized score:')

[1] "Normalized score:"

> print(lowest_temp_results$normalized_score)

[1] 0

> print('Parents:')

[1] "Parents:"

> print(lowest_temp_results$parents)

     [,1]
[1,]    3
[2,]    0
[3,]    1
[4,]    2

> print('Outcomes:')
```

```
[1] "Outcomes:"

> print(lowest_temp_results$outcomes)

     [,1] [,2] [,3]
[1,]    1    0   -1
[2,]   -1    0    1
[3,]   -1    0    1
[4,]   -1    0    1

>
```

Subsequent fits may be started using the network from a previous fit as the initial conditions, as in the following example (the initial network in the case already has a score of 0).

```
> results <- parallelFit(indata,
+                        max_parents=1,
+                        n_cycles=10,
+                        n_write=10,
+                        T_lo=0.001,
+                        T_hi=2.0,
+                        target_score=0,
+                        n_proc=1,
+                        logfile='try.log',
+                        init_parents=lowest_temp_results$parents,
+                        init_outcomes=lowest_temp_results$outcomes)
> lowest_temp_results <- results[[1]]
> print('Unnormalized score:')

[1] "Unnormalized score:"

> print(lowest_temp_results$unnormalized_score)

[1] 0

> print('Normalized score:')

[1] "Normalized score:"
```

```
> print(lowest_temp_results$normalized_score)

[1] 0

> print('Parents:')

[1] "Parents:"

> print(lowest_temp_results$parents)

     [,1]
[1,]    3
[2,]    0
[3,]    1
[4,]    2

> print('Outcomes:')

[1] "Outcomes:"

> print(lowest_temp_results$outcomes)

     [,1] [,2] [,3]
[1,]    1    0   -1
[2,]   -1    0    1
[3,]   -1    0    1
[4,]   -1    0    1

>
```

3 Session Info

```
> sessionInfo()
```

```
R version 4.5.0 RC (2025-04-04 r88126)
Platform: x86_64-pc-linux-gnu
Running under: Ubuntu 24.04.2 LTS

Matrix products: default
BLAS:   /home/biocbuild/bbs-3.21-bioc/R/lib/libRblas.so
LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0  LAPACK version 3.12.0

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=en_GB              LC_COLLATE=C
 [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C

time zone: America/New_York
tzcode source: system (glibc)

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] ternarynet_1.52.0

loaded via a namespace (and not attached):
 [1] compiler_4.5.0      magrittr_2.0.3      cli_3.6.4
 [4] parallel_4.5.0      tools_4.5.0         igraph_2.1.4
 [7] codetools_0.2-20    BiocParallel_1.42.0 lifecycle_1.0.4
[10] pkgconfig_2.0.3     rlang_1.1.6
```