retnfit: Replica Exchange Ternary Network Fit

Harry A. Stern and Matthew N. McCall

January 6, 2017

parallelFit function

The retnfit package contains a parallel implementation of the replica exchange algorithm for fitting ternary network models. The model is the same as that described in reference 1 and implemented in the ternarynet package.

The retnfit package contains a single function, parallelFit, for fitting a ternary network model to target data consisting of the steady-state responses given a set of perturbations, again as described in reference 1. The function takes the following arguments:

experiment_set data frame containing five columns: i_exp (experiment index), i_node (node index), outcome (-1/0/1), value (cost for that outcome), is_perturbation (0 or 1)

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

seed seed for random number generator

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 transition rule, giving the outcome of a node for each configuration of parent node.

Example

```
library('retnfit')
results <- parallelFit(read.csv('n8.1.csv'),
                       max_parents=4,
                       n_cycles=10000,
                       n_write=10,
                       T_lo=0.001,
                       T_hi=1.0,
                       target_score=0,
                       n_proc=12,
                       logfile='a.log',
                       seed=525108)
save(results, file='a.rda')

lowest_temp_results <- results[[1]]

print('Unnormalized score:')
print(lowest_temp_results$unnormalized_score)

print('Normalized score:')
print(lowest_temp_results$normalized_score)

print('Parents:')
print(lowest_temp_results$parents)

print('Outcomes:')
```

```
print(lowest_temp_results$outcomes)


results <- parallelFit(read.csv('sampledata.csv'),
                       max_parents=4,
                       n_cycles=10000,
                       n_write=10,
                       T_lo=0.001,
                       T_hi=1.0,
                       target_score=0,
                       n_proc=12,
                       logfile='a.log',
                       seed=525108)
```

In this example, the input file (`sampledata.csv`) is of the form

```
i_exp, i_node, outcome, value, is_perturbation
0, 0, -1, 2.0, 0
0, 0, 0, 1.0, 0
0, 0, 1, 0.0, 1
0, 1, -1, 1.0, 0
0, 1, 0, 0.0, 0
0, 1, 1, 1.0, 0
0, 2, -1, 1.0, 0
0, 2, 0, 0.0, 0
0, 2, 1, 1.0, 0
0, 3, -1, 1.0, 0
0, 3, 0, 0.0, 0
0, 3, 1, 1.0, 0
0, 4, -1, 1.0, 0
0, 4, 0, 0.0, 0
0, 4, 1, 1.0, 0
0, 5, -1, 1.0, 0
0, 5, 0, 0.0, 0
0, 5, 1, 1.0, 0
0, 6, -1, 1.0, 0
0, 6, 0, 0.0, 0
```

```
0, 6, 1, 1.0, 0
0, 7, -1, 1.0, 0
0, 7, 0, 0.0, 0
0, 7, 1, 1.0, 0
1, 0, -1, 1.0, 0
1, 0, 0, 0.0, 0
1, 0, 1, 1.0, 0
.
.
.
```

The output will be of the form

```
[1] "Exiting Rmpi. Rmpi cannot be used unless relaunching R."
[1] "Unnormalized score:"
[1] 4
[1] "Normalized score:"
[1] 0.0625
[1] "Parents:"
     [,1] [,2] [,3] [,4]
[1,]    1    2    6    7
[2,]    2    3    4    7
[3,]    0    5    6    7
[4,]    1    2    5    7
[5,]    0    1    3    5
[6,]    0    1    3    6
[7,]    2    3    5    7
[8,]    0    2    4    5
[1] "Outcomes:"
, , 1, 1, 1

     [,1] [,2] [,3]
[1,]    0    0   -1
[2,]    0    1    0
[3,]    0    0    0
[4,]    0   -1    0
[5,]   -1    1   -1
```

```
[6,]    1    1    1
[7,]    1    1    0
[8,]    1    1    1

, , 2, 1, 1

     [,1] [,2] [,3]
[1,]   -1    0    1
[2,]   -1    1   -1
[3,]    0    0   -1
[4,]    0   -1    1
[5,]    1    0   -1
[6,]   -1    1    1
[7,]    1    1    1
[8,]    1   -1    0
.
.
.
```

References

1. Almudevar, A., McCall, M. N., McMurray, H., and Land. H., "Fitting Boolean networks from steady state perturbation data," *Statistical Applications in Genetics and Molecular Biology* **10** (2011)

2. Harry A. Stern and Matthew N. McCall, "Fitting ternary network models of gene regulatory networks by replica exchange Monte Carlo," submitted to *Bioinformatics*