`retnfit`: Replica Exchange Ternary Network Fit

Harry A. Stern and Matthew N. McCall

January 9, 2018

`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')
> i_exp <- as.integer(c(0,0,0,0,0,0,0,0,0,1,1,1,
+                       1,1,1,1,1,1,1,1,1,1,1,1,
+                       1,1,1,1))
> i_node <- as.integer(c(0,0,0,1,1,1,2,2,2,0,0,0,
+                       1,1,1,2,2,2,0,0,0,1,1,1,
+                       2,2,2))
> 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))
> value <- c(2.0,1.0,0.0,2.0,1.0,0.0,2.0,1.0,0.0,
+             2.0,1.0,0.0,2.0,1.0,0.0,2.0,1.0,0.0,
+             2.0,1.0,0.0,2.0,1.0,0.0,2.0,1.0,0.0)
> is_perturbation <- as.integer(c(0,0,1,0,0,0,0,0,0,0,0,0,
+                                 0,0,0,1,0,0,0,0,0,0,0,0,
+                                 0,0,0,0,1))
> indata <- data.frame(i_exp,i_node,outcome,value,is_perturbation)
> results <- parallelFit(indata,
+                       max_parents=1,
+                       n_cycles=10000,
+                       n_write=10,
+                       T_lo=0.001,
+                       T_hi=1.0,
+                       target_score=0,
+                       n_proc=3,
+                       logfile='test.log',
```

```
+                          seed=12345)
> 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,]    2
[2,]    0
[3,]    1

> print('Outcomes:')

[1] "Outcomes:"

> print(lowest_temp_results$outcomes)

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

>
```

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," manuscript in preparation