Package ‘PCSinR’

January 20, 2025

Title Parallel Constraint Satisfaction Networks in R

Version 0.1.0

URL https://github.com/felixhenninger/PCSinR

Description Parallel Constraint Satisfaction (PCS) models are an increasingly common class of models in Psychology, with applications to reading and word recognition (McClelland & Rumelhart, 1981), judgment and decision making (Glöckner & Betsch, 2008; Glöckner, Hilbig, & Jekel, 2014), and several other fields (e.g. Read, Vanman, & Miller, 1997). In each of these fields, they provide a quantitative model of psychological phenomena, with precise predictions regarding choice probabilities, decision times, and often the degree of confidence. This package provides the necessary functions to create and simulate basic Parallel Constraint Satisfaction networks within R.

Depends R (>= 3.3.1)

License GPL (>= 3)

Encoding UTF-8

LazyData true

RoxygenNote 5.0.1

Suggests testthat

NeedsCompilation no

Author Felix Henninger [aut, cre]

Maintainer Felix Henninger <mailbox@felixhenninger.com>

Repository CRAN

Date/Publication 2016-10-19 22:12:25

Contents

PCSinR — *PCS: Parallel Constraint Satisfaction networks in R*

Description

The PCS package contains all necessary functions for building and simulation Parallel Constraint Satisfaction (PCS) network models within R.

Details

PCS models are an increasingly used framework throughout psychology: They provide quantitative predictions in a variety of paradigms, ranging from word and letter recognition, for which they were originally developed (McClelland & Rumelhart, 1981; Rumelhart & McClelland, 1982), to complex judgments and decisions (Glöckner & Betsch, 2008; Glöckner, Hilbig, & Jekel, 2014), and many other applications besides.

Theoretical overview

PCS networks embody the concept of *consistency maximization* in perception and cognition, in that they assume that a cognitive system will attempt to achieve a coherent state, in which all available information is weighted to provide a maximally consistent representation of a given task. Their central qualitative prediction follows from this basic assumption, namely that the weights assigned to available information are reevaluated during the decision process. These coherence shifts are a unique prediction of PCS models, and have been found in multiple domains (c.f. Glöckner, Betsch, & Schindler, 2010; Holyoak & Simon, 1999, Simon & Holyoak, 2002).

PCS models are implemented as neural networks, though they do not assume a direct mapping from model nodes and connections onto neurons and dendrites. Instead, the *nodes* represent concepts, and the *links* between them the degree to which the concepts are compatible or reconcilable. The assumption is that a PCS network is instantiated whenever a decision maker faces a choice (Glöckner & Betsch, 2008).

At any given time, a node exhibits a certain level of *activation*, which it passes through any present links to other nodes. If the level is positive, the node is activated, otherwise it is labelled inhibited. Activation is passed between nodes along the links, to varying degrees depending on their strength and nature, which determines the spread of activation in the network. Links can be excitatory, in that an activated node on one side leads to an increasing activation of any connected node, or inhibitory, in which connected nodes assume the opposite activation level. Thus, nodes can be mutually supportive regarding their level of activation, or restrain one another. Besides this qualitative difference, links also differ in their weight, a number which denotes the proportion of activation that is passed along the link. A link's magnitude captures the connection weight, and its sign the qualitative type of influence (excitatory or inhibitory). Links are always bidirectional, in that both nodes reciprocally influence one another, in the same manner and to the same extent.

Within the network, processing occurs in discontinuous cycles, *iterations*. In each cycle anew, nodes pass a proportion of their activation level along the links to connected siblings. At each receiving node, the total arriving activation is termed the total input. Because the amount of activation passed through a link is multiplied by the link weight, the total input is a weighted sum of the activation of all connected nodes. The input does not, however, influence the node directly, but instead is subject to two additional influences: First, the activation of each node is reduced by a fixed proportion at each iteration, so that the activation level *decays* to a fixed neutral point. Second, the current activation level of the node determines the influence of the arriving input: A node that is already active is less susceptible to further excitatory input, and more so to external inhibition. The converse holds for an inhibited node: Excitatory input is amplified, and further inhibition dampened. These forces constrain the activation between a floor and ceiling value.

Together, these two forces determine the reaction of a node to input. In particular, from their joint activity a non-linear *activation function* emerges: The level of activation a node approches over many interations is an s-shaped function of the input for excitatory links, concave for positive and convex for negative input. For an inhibitory link, this relationship is inverted.

Activation initially enters a network through the *source node*, which provides a constant level of activation. As activation enters the network and is passed between nodes, the properties sketched above ensure that the relationships between the concepts represented will increasingly be satisfied, and after some time, the network reaches a stable state in which nodes connected by excitatory links will share broadly similar levels of activation, and those connected by inhibitory links dissimilar states. Thus, the constraints represented in the network will be increasingly satisfied (giving the model family its name), and the representation will become *coherent*.

When a network has converged into this state, *behavioral predictions* can be derived: The number of iterations that passed during processing is used as a proxy for decision time, of the nodes representing choice alternatives, the one with the highest activation is assumed to be the chosen one, and the difference between the activations of these nodes is used to predict the confidence with which a decision is made or a course of action taken.

Package contents

This package contains all necessary simulation code to build and run PCS models. In particular, it contains a full, optimized implementation of the core model as specified by McClelland and Rumelhart (1981) as well as Glöckner and Betsch (2008), as well as several variants commonly used in the literature so that existing findings may be replicated.

`PCS_run` is the central function provided by the package. It creates, and runs, a model of a PCS network given a connection matrix and the necessary parameters.

Please see the function-specific documentation for additional information

References

PCS

Glöckner, A., & Betsch, T. (2008). Modeling option and strategy choices with connectionist networks: Towards an integrative model of automatic and deliberate decision making. Judgment and Decision Making, 3(3), 215–228.

Glöckner, A., Betsch, T., & Schindler, N. (2010). Coherence shifts in probabilistic inference tasks. Journal of Behavioral Decision Making, 23(5), 439–462. doi:10.1002/bdm.668

Glöckner, A., Hilbig, B. E., & Jekel, M. (2014). What is adaptive about adaptive decision making? A parallel constraint satisfaction account. Cognition, 133(3), 641–666. doi:10.1016/j.cognition.2014.08.017

Holyoak, K. J., & Simon, D. (1999). Bidirectional reasoning in decision making by constraint satisfaction. Journal of Experimental Psychology: General, 128(1), 3–31.

McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception: I. An account of basic findings. Psychological Review, 88(5), 375–407.

Rumelhart, D. E., & McClelland, J. L. (1982). An interactive activation model of context effects in letter perception: II. The contextual enhancement effect and some tests and extensions of the model. Psychological Review, 89(1), 60–94.

Simon, D., & Holyoak, K. (2002). Structural dynamics of cognition: From consistency theories to constraint satisfaction. Personality and Social Psychology Review, 6(4), 283–294.

`PCS_convergence_McCandR` *Check a PCS network for convergence*

Description

This function applies the convergence criterion defined by McClelland and Rumelhart to a given network, and returns either a (qualitative) boolean value that represents the convergence state, or a (quantitative) value that represents the number of iterations (of the last 10) that have met the convergence threshold.

Usage

```
PCS_convergence_McCandR(iteration, current_energy, memory.matrix,
  stability_criterion = 10^-6, output = "qualitative")
```

Arguments

- `iteration` The iteration to consider – in most cases, this will be the current iteration during a simulation run, however, the check can also be applied to a model output retroactively, and the iteration specified manually.
- `current_energy` The current energy level within the network
- `memory.matrix` A matrix of iteration, energy and node states (in columns, in that order), across all previous iterations (in rows).
- `stability_criterion` Criterion for stability. Changes below this value are no longer considered significant, and ten iterations without significant changes to the energy level in succession will trigger the convergence check.
- `output` Either `'qualitative'` (default), in which case the check returns a boolean value representing whether it has passed or not, or `'quantitative'`, in which case the number of checked trials for which the convergence criterion was met is returned. This last option is of most value for debugging convergence.

Details

The check requires the following parameters:

PCS_run *Simulate the run of a PCS model*

Description

PCS_run simulates a PCS network given a pre-specified interconnection matrix and model parameters, according to the mechanism outlines by McClelland and Rumelhart (1981).

Usage

```
PCS_run(interconnection_matrix, initial_state, resting_levels, reset,
  node_names = NULL, stability_criterion = 10^-6, max_iterations = Inf,
  convergence_criteria = c(PCS_convergence_McCandR),
  convergence_names = NULL)
```

Arguments

`interconnection_matrix`
: A square, matrix representing the link weights between nodes, such that each entry w_ij represents the link strength between nodes i and j. Accordingly, for a network of n nodes, the matrix must be of six n*n. In most applications, the matrix will be symmetric, meaning that links are bidirectional.

`initial_state`
: Initial node activations before the first iteration is run. In most cases, this will be a vector of zeros, with the length corresponding to the number of nodes in the network.

`resting_levels`
: Resting activation level for each node. In most cases, this will be a vector of zeros, with its length corresponding to the number of nodes in the network.

`reset`
: Vector denoting nodes with stable activation values. The vector contains a value for each node; if it is unequal to zero, the node activation will be reset to this value after each iteration.

`node_names`
: Vector specifying human-readable labels for every node, or `'default'`, in which case nodes are automatically named.

`stability_criterion`
: Stability theshold for convergence criteria. If energy changes across iterations fall below this threshold, the model is considered to have converged.

`max_iterations`
: Maximum number of iterations to run before terminating the simulation.

`convergence_criteria`
: Array of convergence criteria to apply. This PCS implementation allows users to define and observe multiple convergence criteria in one model. Each entry in this array is a convergence criterion, which is representated as a function that receives the current iteration, energy, model state history and the `stability_criterion` defined above and returns a boolean value representing whether the particular criterion is met given the model's current state.

`convergence_names`
: Human-readable labels for the convergence criteria, or `'default'`, in which case the criteria are numbered automatically, in which case the criteria are numbered automatically.

Value

A list representing the model state after all convergence criteria have been fullfilled. The key `iterations` contains the model state over its entire run, while the key `convergence` defines which convergence criteria have been met at which iteration. Together, these provide an exhaustive summary of the model's behavior.

`PCS_run_from_interconnections`
Simulate the run of a PCS model based on only the interconnection matrix

---

Description

`PCS_run_from_interconnections` simulates a PCS network given *only* the pre-specified interconnection matrix and convergence criteria, substituting default values from the literature for all other parameters. Thereby, it provides a convenient shorthand for the `PCS_run` function that covers the vast majority of applications.

Usage

```
PCS_run_from_interconnections(interconnection_matrix,
  convergence_criteria = c(PCS_convergence_McCandR),
  convergence_names = "default")
```

Arguments

`interconnection_matrix`
: A square, matrix representing the link weights between nodes, such that each entry w_ij represents the link strength between nodes i and j. Accordingly, for a network of n nodes, the matrix must be of six n*n. In most applications, the matrix will be symmetric, meaning that links are bidirectional.

`convergence_criteria`
: Array of convergence criteria to apply. This PCS implementation allows users to define and observe multiple convergence criteria in one model. Each entry in this array is a convergence criterion, which is representated as a function that receives the current iteration, energy, model state history and the `stability_criterion` defined above and returns a boolean value representing whether the particular criterion is met given the model's current state.

`convergence_names`
: Human-readable labels for the convergence criteria, or `'default'`, in which case the criteria are numbered automatically, in which case the criteria are numbered automatically.

Examples

```
# Build interconnection matrix
interconnections <- matrix(
  c( 0.0000,  0.1015,  0.0470,  0.0126,  0.0034,  0.0000,  0.0000,
    0.1015,  0.0000,  0.0000,  0.0000,  0.0000,  0.0100, -0.0100,
    0.0470,  0.0000,  0.0000,  0.0000,  0.0000,  0.0100, -0.0100,
    0.0126,  0.0000,  0.0000,  0.0000,  0.0000,  0.0100, -0.0100,
    0.0034,  0.0000,  0.0000,  0.0000,  0.0000, -0.0100,  0.0100,
    0.0000,  0.0100,  0.0100,  0.0100, -0.0100,  0.0000, -0.2000,
    0.0000, -0.0100, -0.0100, -0.0100,  0.0100, -0.2000,  0.0000 ),
  nrow=7
  )

# Run model
result <- PCS_run_from_interconnections(interconnections)

# Examine iterations required for convergence
result$convergence

# Examine final model state
result$iterations[nrow(result$iterations),]
```

Index