Optimising ACT-R models using differential evolution

David Peebles

31st July 2016

Introduction

The key file in this repository is *diff-evol.lisp* which contains lisp code implementing the original differential evolution (DE) algorithm described in [2] for optimising parameters of ACT-R cognitive models. A brief introduction to the DE algorithm is available in [1] (available in this repository).

In order to use the code for your ACT-R model you must:

- Ensure that the output of your model is a measure of fit of the model to data (e.g., the value returned from ACT-R's *correlation* function).
- Define a function to run the model that suppresses any output to standard output and any warnings that may be generated due to poor parameter choices (see examples below).
- Provide the values for the four DE algorithm parameters (reasonable values for each of these parameters are discussed in [1]). These are defined in the file with the following default values:

```
(defparameter *ngen* 10 "Number of generations")
(defparameter *NP* 25 "Population size")
(defparameter *F* 0.5 "Weighted difference computation scale factor")
(defparameter *CR* 0.5 "Recombination computation crossover constant")
```

Running the DE function

To use this code you should load the elements in the following order:

1. load-act-r.lisp
2. diff-evol.lisp
3. Your ACT-R model code

The optim function

The main function to call is (optim) which requires two parameters:

- A function (which must take no parameters) that will be called to generate a result for a trial.
- A list of parameter specifications. Each parameter specification must be a list of three items:
  - The first is an indicator for the parameter which will be provided to the parameter assignment function.
  - The second must be a real value that indicates the minimum value that parameter can have.
  - The third must be a real value that indicates the maximum value that parameter can have.

In addition, there are several keyword parameters which may be provided.

Keyword	Description
:ngen	Number of generations to run (default = *ngen*).
:np	The size of each generation (default = *NP*).
:f	Difference calculation scale factor. Must be a real value in the range 0 < f <= 1.2 (default = *F*).
:cr	The crossover rate. Must be a real value in the range 0 <= cr <= 1 (default = *CR*).
:assign-model-parameters	Must be a function which accepts one parameter. That function will be passed a list of lists prior to calling the result function. Each sublist will have two elements: 1. the first value will be a parameter indicator (as given in the parameter specification), 2. the second value will be the current value for that parameter in the vector for which a result is desired. If ACT-R is also loaded then it will have a default value of a function that will set ACT-R general parameters for a model after it is reset. Otherwise, a function must be specified.
:direction	Either the keyword :min or :max indicating how to find the best result in a generation. The default is :max.
:output	An indicator for a stream to which all of the output will be written. The default is t, which outputs to *standard-output*. A value of nil will suppress the output.

Example 1. The Paired Associate *model*

The first example below is the *paired associate* model from unit 4 of the ACT-R tutorial. First load the fan effect model from unit 4 of the ACT-R tutorial. Then change the output function used in the code for the experiment to just return the correlation. Write a function to run the task suppressing the model output and any warnings which may result from poor parameter choices. Finally specify the three parameters to adjust in the model with their ranges and use the default DE configuration

```
(actr-load "ACT-R:tutorial;unit4;paired.lisp")

(defun output-data (data n)
  (let ((probability (mapcar (lambda (x) (/ (first x) n)) data))
        (latency (mapcar (lambda (x) (/ (or (second x) 0) n)) data)))
    ;; (correlation latency *paired-latencies*)
    (correlation probability *paired-probability*)))

(defun run-paired-with-no-output ()
  (let ((*standard-output* (make-string-output-stream)))
    (suppress-warnings (paired-experiment 100))))

(optim 'run-paired-with-no-output '((:rt -5.0 -0.01) (:lf 0.1 1.5) (:ans 0.1 0.8)))
```

Example 2. The Fan Effect *model*

The second example is the *fan effect* model from unit 5 of the ACT-R tutorial. First load the fan effect model. Then change the output function used in the code for the experiment to just return the correlation. Write a function to run the task suppressing the model output and any warnings which may result from poor parameter choices. Finally specify the two parameters to adjust in the model with their ranges and use the default DE configuration

```lisp
(actr-load "ACT-R:tutorial;unit5;fan.lisp")

(defun output-person-location (data)
  (let ((rts (mapcar 'first data)))
    (correlation rts *person-location-data*)))

(defun run-fan-with-no-output ()
  (let ((*standard-output* (make-string-output-stream)))
    (suppress-warnings (fan-experiment))))

(optim 'run-fan-with-no-output '((:mas 1.4 3.0) (:lf 0.1 1.5)))
```

Now change the output function to return the mean deviation and see what we get when minimizing that

```lisp
(defun output-person-location (data)
  (let ((rts (mapcar 'first data)))
    (mean-deviation rts *person-location-data*)))

(optim 'run-fan-with-no-output '((:mas 1.4 3.0) (:lf 0.1 1.5)) :direction :min :output nil)
```

Acknowledgements

As with every other of my ACT-R related projects, this work has benefited enormously from the advice and support of Dan Bothell at CMU. For this project, my original lisp code has been improved beyond recognition by Dan.

References

[1] D. Peebles. "Two methods for search and optimising cognitive model parameters". In: *Proceedings of the 14th International Conference on Cognitive Modeling*. Ed. by F. Ritter and D. Reitter. 2016.

[2] R. Storn and K. Price. "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of global optimization* 11.4 (1997), pp. 341–359.