

[ANN]cl-ctrnn

Source: <http://coding.derkeiler.com/Archive/Lisp/comp.lang.lisp/2007-12/msg00616.html>

- *From:* Oyvin Halfdan Thuv <oyvinht@xxxxxxxxxxxxx>
 - *Date:* Mon, 10 Dec 2007 16:45:20 +0000 (UTC)
-

I have made a library for CTRNN simulations available at
<http://www.pvv.ntnu.no/~oyvinht/static/OSS/cl-ctrnn/>

Short info:

- Simulates CTRNNs (Contiuous–Time Recurrent Neural Networks)
- Useful for artificial life and genetic algorithms/programs
- BSD license
- Written for readability (but should be reasonably fast too ... speed of the CTRNN is usually not the problem in complex simulation environments anyway)

Comments are extremely welcome.

Long info:

Se docs and the example.lisp file:

```
;;; Assuming that the library is loaded
```

```
(defpackage #:ctrnn-example  
  (:use "COMMON-LISP" "CTRNN"))
```

```
(in-package :ctrnn-example)
```

```
;;;-----  
;;; First example: Test the interplay of two random neurons  
;;;-----  
(defun example-1 ()  
  ;; Create a network  
  (let ((net (make-instance 'neural-network))  
        (neuron-1 (make-instance 'neuron  
  :bias (- (random 40) 20)  
  :time-constant (+ (/ (random 10) 10)  
  *timestep*)))  
        (neuron-2 (make-instance 'neuron  
  :bias (- (random 40) 20)  
  :time-constant (+ (/ (random 10) 10)  
  *timestep*))))  
    ;; Add the neurons to the network  
    (add-neuron! net neuron-1)
```

```

(add-neuron! net neuron-2)
;; Add some synaptic connections between the neurons
(add-dendrite!
neuron-1 (make-instance 'synapse
:from-neuron neuron-2
:strength (random 5)))
(add-dendrite!
neuron-2 (make-instance 'synapse
:from-neuron neuron-1
:strength (random 5)))
;; Simulate the network for 20 (simulated seconds)
;; and see (every 3 seconds) how the firing frequency of the neurons change
(let ((*timestep* 0.001)) ; Set simulation precision to 1 msec
(dotimes (var 20)
;; Update potential for 3 seconds
(dotimes (var 300)
(synchronously-update-membrane-potentials! net))
;; Se what the firing-frequency for each of the neurons are
(format t "t=~2,'0D:~%" (+ var 1))
(format t " Neuron 1: ff=~.6F~% Neuron 2: ff=~.6F~%"
(firing-frequency neuron-1)
(firing-frequency neuron-2))))))

```

```

;;;-----
;;; Second example: Se how firing freq. changes by sensor reading
;;;-----
(defparameter *my-sensor-reading* 1.0)
(defparameter *my-motor-force* 0.0)

(defun example-2 ()
;; Reset
(setq *my-sensor-reading* 1.0)
(setq *my-motor-force* 0.0)
;; Create a network
(let ((net (make-instance 'neural-network))
(neuron-1 (make-instance 'sensor-neuron
:bias 1.0
:sensor-function
#'(lambda (self)
(setf (neuron-external-current self)
*my-sensor-reading*))
:time-constant 0.1))
(neuron-2 (make-instance 'motor-neuron
:bias -2.0
:motor-function
#'(lambda (self)
(setf *my-motor-force*
(* 10 (firing-frequency self))))))
:time-constant 0.2)))

```

```
;; Add the neurons to the network
(add-neuron! net neuron-1)
(add-neuron! net neuron-2)
;; Add some synaptic connections between the neurons
(add-dendrite!
neuron-1 (make-instance 'synapse
:from-neuron neuron-2
:strength 2))
(add-dendrite!
neuron-2 (make-instance 'synapse
:from-neuron neuron-1
:strength -3))
;; Simulate the network for 20 (simulated seconds)
;; and see (every 3 seconds) how the firing frequency of the neurons change
(let ((*timestep* 0.001)) ; Set simulation precision to 1 msec
(dotimes (var 20)
(when (> var 10)
(setf *my-sensor-reading* 0.0))
;; Update potential for 3 seconds
(dotimes (var 300)
(synchronously-update-membrane-potentials! net))
;; See what happens to the motor
(format t "motor force = ~A~%" *my-motor-force*))))
```

--

Oyvin

.