AN ATTENTION-MODULATED ASSOCIATIVE NETWORK – A MATLAB SIMULATOR

THORWART, A., LIVESEY, E, & HARRIS, J. (2012). NORMALISATION BETWEEN STIMULUS ELEMENTS IN A MODEL OF PAVLOVIAN CONDITIONING: SHOWJUMPING ON AN ELEMENTAL HORSE. *LEARNING & BEHAVIOR*, 40, 334-346. DOI:10.3758/S13420-012-0073-7

The following describes a simulator for a version of Harris & Livesey's (2010) attention-modulated associative network (AMAN). We hope that its handling is straightforward and simplifies the application of the theory to various learning problems. The simulator was developed in context of a special issue of Learning & Behavior. We would appreciate it if you would cite the later paper when you have used the simulator in your research.

If there is any problem or if you have any questions, please contact Anna Thorwart (anna.thorwart@staff.uni-marburg.de)

1. INTRODUCTION

This is only a short and general introduction as descriptions that are more detailed are given *inside* the corresponding files. The programme, with accompanying instructions, can be freely downloaded from our website:

http://www.psych.usyd.edu.au/staff/justinh/downloads/AMAN Simulator/

The programme has been written in Matlab® as two .m files, plus one .fig file. These files have also been compiled as executable files that can be run as standalone programmes, for use by researchers who do not have the Matlab software package. Currently, standalone versions for Windows 32-bit, Windows 64-bit, and Mac 64-bit can be downloaded directly from the above website. If you need a standalone version for a different OS, please contact us.

In general, performance is the same in all versions. If you are running the simulator from within Matlab, you will be able to edit the resulting graphs directly using the Matlab figure editor.

2. RUNNING THE SIMULATOR FROM WITHIN MATLAB

The simulator was written using Matlab2010b 64-bit on a Windows 7 PC. Additional tests were conducted with Mac 32-bit and 64-bit as well as Windows 32-bit. Because of major changes in Matlab 7 to the user interfaces, there might be problems in running the simulator in Matlab versions earlier than 7.0 (R14). If this is the case, please use the standalone version.

The actual simulator is implemented in three files, which are located in the MatlabCode folder. Those need to be saved within the same folder on your computer.

1. AMANcontrol.m
2. AMANcontrol.fig

3. AMANmodel.m

The remaining files are simple text files that specify the learning task and the parameters. Examples can be downloaded from the above webpage. You can choose any names and/or locations for those files.

4. stimulus definition (stimulidef.txt)
5. trial definition (trialdef.txt)
6. phase definition (phasedef.txt)
7. parameter definition. (paramdef.txt)

To start the simulator, change your current directory in Matlab to the folder containing the simulator and run the **AMANcontrol.m** file.

3. RUNNING THE STANDALONE APPLICATION

The standalone applications run without Matlab. Before using the simulator, the MCR libraries, a MATLAB® runtime component provided by The MathWorks, Inc., have to be installed. Note that the MCR is specific for the MATLAB compiler that we use to generate the standalone applications. You have to ensure that you have the correct version of the MCR installed on your system.

(The small print: The authors' limited rights to the deployment of the standalone version are governed by a license agreement between the authors and the MathWorks. The license agreement can be found at http://www.mathworks.com/license/.)

Windows 64 or 32 users will need the following files:

1. MCRinstaller.exe
2. AMAN.exe

Mac user will need:

1. MCRinstaller.dmg
2. AMAN.sh
3. the folder AMAN.zip, which contains the AMAN.app folder

The remaining files are simple text files that specify the learning task and the parameters. Examples can be downloaded from the above webpage. You can choose any names and/or locations for those files.

1. stimulus definition (stimulidef.txt)
2. trial definition (trialdef.txt)
3. phase definition (phasedef.txt)
4. parameter definition. (paramdef.txt)

INSTALLING THE MCR AND SETTING SYSTEM PATHS

1. Open the package containing the software needed at run time.
2. Run MCRInstaller once on the target machine, that is, the machine where you want to run the AMAN. The MCRInstaller opens a command window and begins preparation for the installation. *Note for Windows Applications: You must have administrative privileges to install the MCR on a*

target machine since it modifies both the system registry and the system path. Running the MCRInstaller after the MCR has been set up on the target machine requires only user-level privileges.

3. When the MCR Installer wizard appears, click Next to begin the installation. Click Next to continue.
4. In the Select Installation Folder dialog box, specify where you want to install the MCR and whether you want to install the MCR for just yourself or others. Click Next to continue.

   Note The "Install MATLAB Compiler Runtime for yourself or for anyone who uses this computer" option is not implemented for this release. The current default is Everyone.
5. Confirm your selections by clicking Next.

   The installation begins. The process takes some time due to the quantity of files that are installed. The MCRInstaller automatically:

   a. Copies the necessary files to the target folder you specified.
   b. Registers the components as needed.
   c. Updates the system path to point to the MCR binary folder, which is <target_directory>/<version>/runtime/win32|win64.
6. When the installation completes, click Close on the Installation Completed dialog box to exit

After installing the MCR *once*, run **AMAN.exe** on Windows PCs to start the simulator. For 64-bit Macintosh, you run the application through **AMAN.sh** .

4. THE SIMULATOR

Whether running AMANcontrol.m in Matlab or using the compiled files AMAN.exe or AMAN.sh, the main Graphical User Interface (GUI) opens.

The simulator is controlled via a user interface that contains six buttons and a message box. Use the buttons to specify the names of the text files containing the stimulus, trial and phase definition as well as the file containing all parameter values and options.

To simulate the model, the user must create three separate text files that define the stimuli, the trial structure, and the phases of the simulated experiment. The exact details of the content required for each file are described in commented example files that can be downloaded from the above webpage. To open and edit the files, you can use any common text editor[1]. To increase the readability of the files, we recommend using a text editor with syntax highlighting for C, Java, VB or Python, for instance.

[1] If you use the Notepad application to view files, you might see carriage return and line feed symbols instead of line endings. This makes the files less readable in the Notepad application. There are no problems with files you create or edit in the Notepad application, and then use in MATLAB. The files have line endings in the MATLAB Editor, and continue to have line endings when you open them in the Notepad application.

The *stimulidef.txt* file specifies the name of each stimulus (including the US and context), the range of input strengths for the elements of each stimulus, and which stimuli belong to the same modality. The range of the input strength is used to determine the activation of E and A units of an element by external stimuli. More salient stimuli are assumed to have higher input strength. We have implemented three different ways to calculate the external sensory input for each element:

(1) The input strengths across the elements of a stimulus can decrease in equal steps from the maximal to the minimal activation (i.e. across the input range);

(2) for each element, a value is drawn randomly from the uniform distribution across the input range; or

(3) all elements of a stimulus are activated to the mean of the specified input range.

The modality is used to determine the amount of normalisation an I unit receives from other elements' E units. Instead of a graded change in similarity, we specified three different levels of similarity between two elements:

(1) elements representing features of the same stimulus,

(2) elements belonging to representation of another stimulus that belongs to the same modality;

(3) elements belonging to another stimulus in another modality.

The *trialdef.txt* file specifies a name for each type of trial in the simulation, which stimuli are present in that trial and what their onset and offset times are. This file also nominates which stimulus is mapped to the dependent variable in each trial, and when the activation of E units that represent that stimulus is measured. For example, if one is simulating a conditioned response, this would normally be done by nominating activity in the US elements as the dependent variable, and specifying that this is recorded during the time when the CS is presented.

The *phasedef.txt* file provides the name for each phase of the experiment (e.g., training and extinction), and specifies which trial types occur in that phase, how often each occurs within a single cycle (the order of trials is randomised for each cycle), and how many times the cycle is repeated in that phase. For example, in a negative patterning discrimination, the phase may include one A+ trial, one B+ trial, and two AB- trials per cycle, and this might be repeated 100 times in the training phase

A final *paramdef.txt* file includes details such as the number of elements per stimulus, how the input strengths vary between elements and all other parameters needed for the model. A standard version of this file is provided at the above website, and most parameters defined there should not need to be changed.

These files are parsed when you press the "*Simulate*" button. Thus, simulations are always based on the latest saved version of the files and you can change the files after you specified their file names and locations once.

5. THE SIMULATION

The simulator is a real-time implementation of the model and in order to solve the differential equations of the model, it uses a fourth-order Runge–Kutta procedure. Depending on your hardware, the number of stimuli and trials of the simulated learning task and the number of elements and time steps specified, simulations will take from a few seconds to several hours. You can press the "*Abort*" button to stop a running simulation. The "Abort" process can take a few seconds.

6. THE OUTPUT

The simulator records the activation of all E, I and A units in each time step. To reduce the size of the matrices, it only saves the final associative weights.

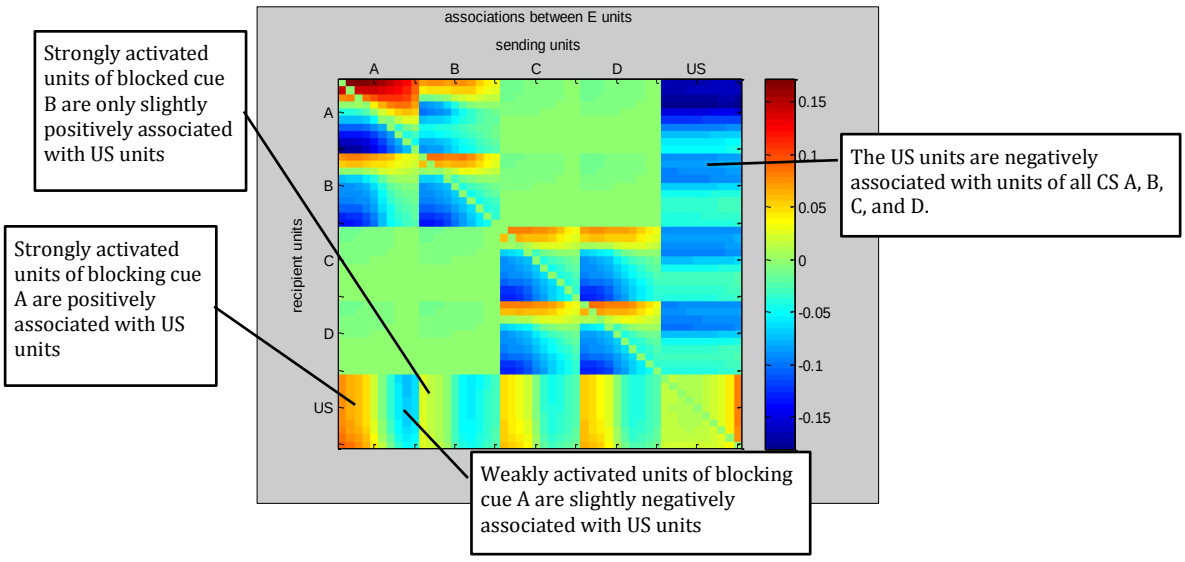
In the final section of the parameter definition file, you can choose which output you would like to be depicted and how to save the output calculated for each trial.

FIGURES

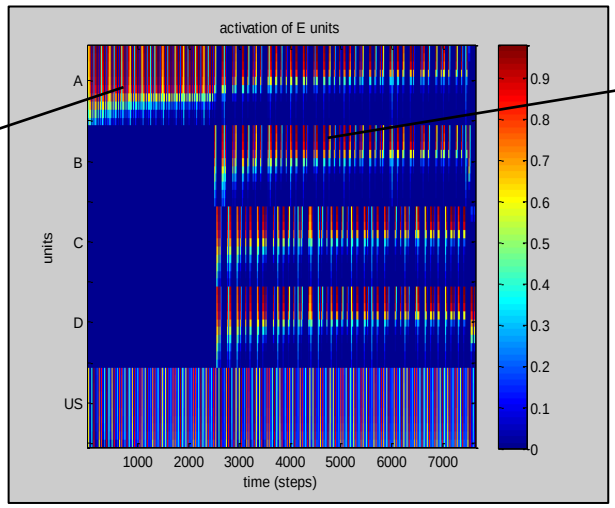
Five different figures can be created. In both compiled and non-compiled versions of the simulator, you can zoom in and out of the figures and retrieve information about the values of certain data points. You also can save the figures as Matlab fig- files or in a number of different file formats (see also "Export setup …" in the menu File of the figure). In the non-compiled version, the full functionality of Matlab plotting tools is available.

The following are examples for a blocking experiment consisting of two training stages (A+; AB+ CD+) and one test stage (B, C, D). Each stimulus (A, B, C, D, US) is represented by 10 and their external sensory input is decreasing linearly.

(1) Final associative weights: Connections between all E units at the end of the experiment



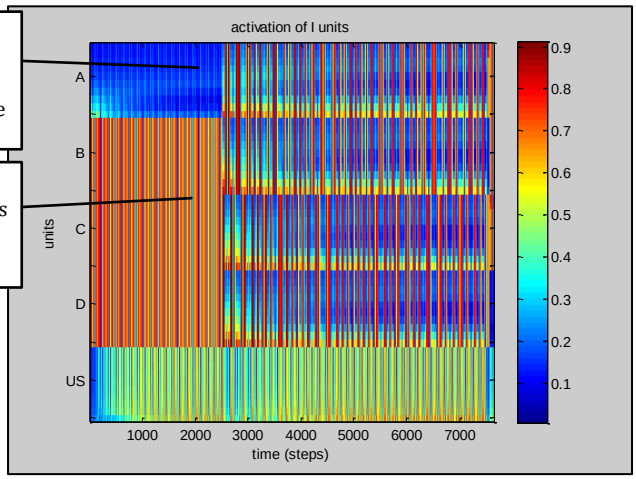
(2) Activation of all E units in each time step of the simulation



Stage 1: Stimulus A is activated and then followed by the US. One can see the linearly decreasing activation of the E units by the external input. At the end of this stage, the weakly activated units of A are getting inhibited.

Stage 2: Either compound AB or CD is presented.

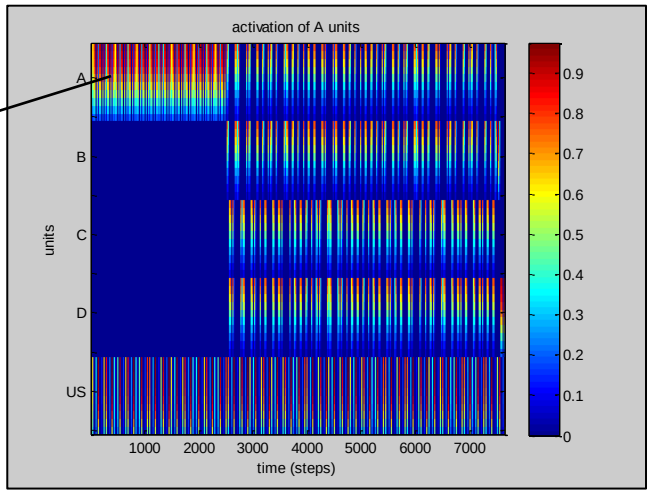
(3) Activation of all I units in each time step of the simulation



Stage 1: Presentation and activation of E units of A strongly activates its I units. At the same time, the latter are inhibited by the A units of stimulus A.

Stage 1: As all CS A, B, C, and D are from the same dimension, stimulus A also activates the I units of the other stimuli.

(4) Activation of all A units in each time step of the simulation



Stage 1: Presentation of stimulus of A activates its A units. Again, the external input is decreasing linearly.

(5) Recorded and averaged activation of E units of test stimulus for each trial as specified in the trial definition. Each panel depicts one phase. In the example below, the US was measured in final five time steps of the CS presentation, and before its own presentation. Note that the Y-axis is scaled automatically and different for each phase.