Package 'catlearn'

March 31, 2025

Type Package

Title Formal Psychological Models of Categorization and Learning

Version 1.1

Date 2025-03-31

Encoding UTF-8

Maintainer Andy Wills <andy@willslab.co.uk>

Description Formal psychological models of categorization and learning, independently-replicated data sets against which to test them, and simulation archives.

License GPL (>= 2)

URL https://github.com/ajwills72/catlearn

BugReports https://github.com/ajwills72/catlearn/issues

Imports Rcpp (>= 1.0.0), doParallel, foreach, tidyr, dplyr

LinkingTo Rcpp, RcppArmadillo (>= 0.10.7.5.0)

LazyData true

Suggests testthat

Depends R (>= 3.5)

NeedsCompilation yes

Author Andy Wills [aut, cre],
Lenard Dome [aut],
Charlotte Edmunds [aut],
Garrett Honke [aut],
Angus Inkster [aut],
René Schlegelmilch [aut],
Stuart Spicer [aut]

Repository CRAN

Date/Publication 2025-03-31 12:30:06 UTC

Contents

`catlearn-package` *Formal Modeling for Psychology.*

Description

Formal psychological models, independently-replicated data sets against which to test them, and simulation archives.

Details

For a complete list of functions, use `library(help = "catlearn")`.

For a complete table of simulations, use `data(thegrid)`.

All functions are concisely documented, use the help function e.g `?shin92`.

For more detailed documentation, see the references listed by the help documentation.

For a tutorial introduction, see Wills et al. (2016a).

For a guide to contributing to this package, Catlearn Research Group (2016).

Author(s)

Andy Wills

Maintainer: Andy Wills <andy@willslab.co.uk>

References

Catlearn Research Group (2016). Contributing to catlearn. http://catlearn.r-forge.r-project.org/intro-catlearn.pdf

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2016). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation*.

`act2probrat` *Convert output activation to a rating of outcome probability*

Description

Logistic function to convert output activations to rating of outcome probability (see e.g. Gluck & Bower, 1988).

Usage

```
act2probrat(act, theta, beta)
```

Arguments

`act`	Vector of output activations
`theta`	Scaling constant
`beta`	Bias constant

Details

The contents of this help file are relatively brief; a more extensive tutorial on using act2probrat can be found in Spicer et al. (n.d.).

The function takes the output activation of a learning model (e.g. slpRW), and converts it into a rating of the subjective probability that the outcome will occur. It does this separately for each activation in the vector `act`. It uses a logistic function to do this conversion (see e.g. Gluck & Bower, 1988, Equation 7). This function can produce a variety of monotonic mappings from activation to probability rating, determined by the value set for the two constants:

`theta` is a scaling constant; as its value rises, the function relating activation to rating becomes less linear and at high values approximates a step function.

`beta` is a bias parameter; it is the value of the output activation that results in an output rating of P = 0.5. For example, if you wish an output activation of 0.4 to produce a rated probability of 0.5, set beta to 0.4.

Value

Returns a vector of probability ratings.

Note

As this function returns probabilities, the numbers returned are always in the range 0-1. If the data you are fitting use a different range, convert them. For example, if your data are ratings on a 0-10 scale, divide them by 10. If your data are something other than probability estimates (e.g. you asked participants to use negative ratings to indicate preventative relationships), don't use this function unless you are sure it is doing what you intend.

Author(s)

Andy Wills

References

Gluck, M.A. & Bower, G.H. (1988). From conditioning to category learning: An adaptive network model. *Journal of Experimental Psychology: General, 117*, 227-247.

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (n.d.). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

convertSUSTAIN *Convert nominal-dimension input representation into a 'padded' (slp-SUSTAIN) format*

Description

Changes a nominal-dimension input representation (e.g. 3 1 2) into a padded representation (e.g. 001 100 010). This form out input representation is required by e.g. `slpSUSTAIN`.

Usage

```
convertSUSTAIN(input, dims)
```

Arguments

`input`	A matrix containing the nominal-dimension input representation. Each row is a trial and each column is a stimulus dimension.
`dims`	A vector of the number of nominal values for each dimension. For example, if there are three dimensions with three, one and two possible values, then `dims = c(3, 1, 2)`.

Value

Returns a matrix containing the padded input representation.

Author(s)

Lenard Dome, Andy Wills

See Also

`slpSUSTAIN`

Examples

```
## Create a dummy training matrix with two dimensions. The first
## two dimensions have two possible nominal values, while the
## third and fourth have three possible nominal values.

dummy <- cbind(matrix(sample(1:2, 20, replace=TRUE), ncol = 2),
               matrix(sample(1:3, 20, replace=TRUE), ncol = 2))

## Specify the number of nominal spaces for each dimension
dims <- c(2, 2, 3, 3)

## Convert the input representation into a binary padded representation
convertSUSTAIN(dummy, dims)
```

homa76 *Category breadth CIRP*

Description

In some category-learning experiments, category members are distortions of an underlying base pattern. Where this is the case, 'category breadth' refers to the magnitude of such distortions. Broad categories take longer to learn than narrow categories. Once trained to an errorless criterion, the effect of category breadth on performance on novel items depends on category size. For small categories, narrow categories are better than broad ones. For larger categries, the reverse is true. Homa & Vosburgh (1976) provide the data for this CIRP.

Usage

```
data(homa76)
```

Format

A data frame with the following columns:

phase Experimental phase (within-subjects). Takes values : 'train','imm'. The training phase is 'train', 'imm' is the immediate test phase.

cond Category breadth (between-subjects). Takes values : 'mixed', 'uni-low'

stim Stimulus type (within-subjects). Takes values : 'proto', 'low', 'med', 'high', 'old-low', 'old-med', 'old-high', 'rand'. All refer to novel stimuli in the test phase, except those beginning 'old-', which are stimuli from the training phase presented during the test phase. 'low', 'med', 'high' refer to distortion level. 'proto' are prototypes. 'rand' are a set of 10 random stimuli, generated from prototypes unrelated to those used in training. These random stimuli are not mentioned in the Method of the paper, but are mentioned in the Results section - they are presented at the end of the test session. Empty cell for training phase.

catsize Category size (within-subjects). Takes values : 3, 6, 9. NA for training phase, where category size is not a meaningful variable given that the DV is blocks to criterion. Also NA for old stimuli; Homa & Vosburgh's (1976) Results section collapses across category size for old stimuli

val For test phases: probability of a correct response, except for random stimuli, where 'val' is the probability with which the random stimuli were placed into the specified category. For training phase: number of blocks to criterion

Details

Wills et al. (n.d.) discuss the derivation of this CIRP. In brief, the effects have been independently replicated. Homa & Vosburgh (1976) was selected as the only experiment to contain all three independently replicated effects.

Homa & Vosburgh's experiment involved the supervised classification of nine-dot random dot patterns. Stimuli had three different levels of distortion from the prototype - low (3.5 bit), medium (5.6 bit), and high (7.7 bit). There were three categories in training, one with 3 members, one with 6

members, and one with 9 members. Participants were either trained on stimuli that were all low distortion (narrow categories), or on an equal mix of low, medium, and high distortion stimuli (broad categories). Training was to an errorless criterion. The test phase involved the presentation of the prototypes, old stimuli, and novel stimuli of low, medium, and high distortion.

The data for the prototype, and other novel test stimuli, were estimated from Figure 1 of Homa & Vosburgh (1976), using `plot digitizer` (Huwaldt, 2015). The data for old stimuli were estimated from Figure 3, using the same procedure. The data for the training phase, and for random stimuli, were reported in the text of Homa & Vosburgh (1976) and are reproduced here. All data are averages across participants.

Homa & Vosburgh's (1976) experiment also includes results for further test phases, delayed by either 1 week, or 10 weeks, from the day of training. These data are not the focus of this category breadth CIRP and have not been included.

Source

Homa, D. & Vosburgh, R. (1976). Category breadth and the abstraction of prototypical information. *Journal of Experimental Psychology: Human Learning and Memory, 2*, 322-330.

Huwaldt, J.A. (2015). Plot Digitizer [software]. https://plotdigitizer.sourceforge.net/

Wills et al. (n.d.). Benchmarks for category learning. *Manuscript in preparation*.

`krus96` *Inverse Base-rate Effect AP*

Description

In the inverse base-rate effect, participants are trained that a compound of two cues (I + PC) leads to a frequently-occurring outcome (C), while another two-cue compound (I + PR) leads to a rarely-occuring outcome (R). The key results are that, at test, participants tend to respond 'C' to cue I on its own, but 'R' to the cue compound (PC + PR). This latter response is striking because PC and PR had been perfectly predictive of diseases C and R respectively, and disease C is more common, so the optimal response to PC + PR is 'C'. Participants respond in opposition to the underlying disease base rates.

Usage

```
data(krus96)
```

Format

A data frame with the following columns:

symptom Symptom presented. Take values: I, PC, PR, PC+PR, I+PC+PR, I+PCo, I+PRo, PC+PRo, I+PC+PRo, as defined by Kruschke (1996).

disease Response made. Takes values: C, R, Co, Ro, as defined by Kruschke (1996).

prop Mean probability of response, averaged across participants.

Details

Wills et al. (n.d.) discuss the classification of these data as a Auxilliary Phenomenon, rather than a CIRP (Canonical Independently Replicated Phenomenon). In brief, these particular results have been independently replicated, but are arguably not the best exemplar of the known phenomena in this area (in particular, they lack a demonstration of the shared-cue effect in IBRE). Auxilliary Phenomena may be included in catlearn if are the subject of a simulation archived in catlearn.

The data are from Experiment 1 of Kruschke (1996), which involved the diagnosis of hyopthetical diseases (F, G, H, J) on the basis of symptoms presented as text (e.g. "ear aches, skin rash"). Participants were trained with feedback across 15 blocks of 8 trials each. They were then tested without feedback on 18 test stimuli, each presented twice.

The data are as shown in Table 2 of Kruschke (1996). The data are mean response probabilities for each stimulus in the test phase, averaged across the two presentations of the stimulus, the two copies of the abstract design, and across participants.

Author(s)

Andy J. Wills, René Schlegelmilch

Source

Kruschke, J.K. (1996). Base rates in category learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22*, 3-26.

References

Wills et al. (n.d.). Benchmarks for category learning. *Manuscript in preparation*.

See Also

`krus96train`

`krus96exit` *Simulation of AP krus96 with EXIT model*

Description

Runs a simulation of the `krus96` AP using the `slpEXIT` model implementation and `krus96train` as the input representation.

Usage

```
krus96exit (params = c(2.87, 2.48, 4.42, 4.42, .222, 1.13, .401))
```

Arguments

params	A vector containing values for c, P, phi, l_gain, l_weight, l_ex, and sigma_bias (i.e. the sigma for the bias unit), in that order. See slpEXIT for an explanation of these parameters.

Details

A simulation using slpEXIT and krus96train. The stored exemplars are the four stimuli present during the training phase, using the same representation as in krus96train.

Other parameters of slpEXIT are set as follows: iterations = 10, sigma for the non-bias units = 1. These values are conventions of modeling with EXIT, and should not be considered as free parameters. They are set within the krus96exit function, and hence can't be changed without re-writing the function.

This simulation is discussed in Spicer et al. (n.d.). It produces the same response probabilities (within rounding error) as the simulation reported in Kruschke (2001), with the same parameters.

56 simulated participants are used in this simulation, the same number as used by Kruschke (2001). Kruschke reports using the same trial randomizations as used for his 56 real participants. These randomizations were not published, so it we couldn't reproduce that part of his simulation. It turns out that the choice of set of 56 randomizations matters, it affects some of the predicted response probabilities. We chose a random seed that reproduced Kruschke's response probabilities to within rounding error. As luck would have it, Kruschke's reported response probabilities (and hence this simulation) are the same (within rounding error) as the results of large sample (N = 500) simulations we have run.

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in krus96.

Author(s)

René Schlegelmilch, Andy Wills

References

Kruschke, J. K. (2001). The inverse base rate effect is not explained by eliminative inference. *Journal of Experimental Psychology: Learning, Memory & Cognition, 27*, 1385-1400.

Spicer, S.G., Schlegelmilch, R., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (n.d.). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

See Also

krus96, krus96train, slpEXIT

krus96train *Input representation of krus96 for models input-compatible with slpEXIT*

Description

Create randomized training blocks for AP krus96, in a format suitable for the slpEXIT model, and other models that use the same input representation format.

Usage

```
krus96train(blocks = 15, subjs = 56, ctxt = TRUE, seed = 1)
```

Arguments

blocks	Number of training blocks to generate. Omit this argument to get the same number of blocks (15) as used in krus96.
subjs	Number of simulated subjects to be run.
ctxt	If TRUE, include a context cue (x7) that appears on every trial.
seed	Sets the random seed.

Details

A data frame is produced, with one row for each trial, and with the following columns:

ctrl - Set to 1 (reset model) for trial 1 of each simulated subject, set to zero (normal trial) for all other training trials, and set to 2 for test trials (i.e. those with no feedback).

block - training block

stim - Stimulus code, as described in Kruschke (1996).

x1, x2, ... - symptom representation. Each column represents one symptom, in the order I1, PC1, PR1, I2, PC2, PR2, context. 1 = symptom present, 0 = symptom absent

t1, t2, ... - Disease representation. Each column represents one disease, in the order C1, R1, C2, R2. 1 = disease present. 0 = disease absent.

Although the trial ordering is random, a random seed is used, so multiple calls of this function with the same parameters should produce the same output. This is usually desirable for reproducibility and stability of non-linear optimization. To get a different order, use the seed argument to set a different seed.

This routine was originally developed to support Wills et al. (n.d.).

Value

A data frame, where each row is one trial, and the columns contain model input.

Author(s)

René Schlegelmilch, Andy Wills

References

Kruschke, J.K. (1996). Base rates in category learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 22*, 3-26.

Wills et al. (n.d.). Benchmarks for category learning. *Manuscript in preparation*.

See Also

krus96

---

medin87train — *Input representation of Exp. 1 in Medin et al. (1987) for models input-compatible with slpALCOVE or slpSUSTAIN.*

Description

Creates randomized training blocks for Experiment 1 in Medin et al. (1987), in a format that is suitable for slpALCOVE, slpSUSTAIN, and other models that use either of those input-representation formats.

Usage

```
medin87train(blocks = 2, subjs = 2, seed = 7649, missing = 'pad')
```

Arguments

subjs	Number of simulated participants to run.
blocks	Number of blocks to generate. The ten trial types are randomized within each block.
seed	Set random seed.
missing	If set to 'geo', output missing dimension flags (see below). If set to 'pad', use the padded stimulus representation format of slpSUSTAIN.

Details

A matrix is produced, with one row for each trial, and with the following columns:

ctrl - Set to 4 on the first trial for each participant - 4 resets the model to the initial state and does unsupervised learning afterwards. Set to 3 for unsupervised trials - normal unsupervised learning trial.

blk - Training block.

stim - Stimulus number, ranging from 1 to 10. The numbering scheme is the same as in Medin et al. (1987, Fig. 1).

`x1, x2, ...` - input representation. Where `missing='geo'`, x1, x2, and x3 are returned, each set at 1 or 0. This is the binary dimensional representation required by models such as slpALCOVE, where e.g. x2 is the value on the second dimension. Where `missing='pad'`, w1, w2, x1, x2, y1, y2, z1, z2, are returned. This is the padded represenation required by models such as slpSUSTAIN; e.g. y1 and y2 represent the two possible values on dimension 3, so if y1 is black, y2 is white, and the stimulus is white, then [y1, y2] = [0, 1].

Although the trial ordering is random, a random seed is used, so multiple calls of this function with the same parameters should produce the same output. This is usually desirable for reproducibility and stability of non-linear optimization. To get a different order, use the seed argument to set a different seed.

Value

R by C matrix, where each row is one trial, and the columns contain model input.

Author(s)

Lenard Dome, Andy Wills

References

Medin, D. L., Wattenmaker, W. D., & Hampson, S. E. (1987). Family resemblance, conceptual cohesiveness, and category construction. *Cognitive Psychology, 19(2)*, 242–279.

---

nosof88 *Instantiation frequency CIRP*

Description

Instantiation frequency is the number of times a stimulus has been observed as a member of a specific category (Barsalou, 1985). Increasing instantiation frequency of a stimulus increases categorization accuracy for that stimulus ('direct' effect), and for other similar stimuli ('indirect' effect). Experiment 1 of Nosofsky (1988) provides the data for this CIRP.

Usage

```
data(nosof88)
```

Format

A data frame with the following columns:

cond Experimental condition, see 'details'. 1 = 'B', 2 = 'E2', 3 = 'E7'

stim Stimulus number, see Nosofsky (1988), Figure 1. Takes values: 1-12

c2acc Mean probability, across participants, of responding that the item belongs to category 2.

Details

Wills et al. (n.d.) discuss the derivation of this CIRP. In brief, both the direct and indirect effects have been independently replicated. Experiment 1 of Nosofsky (1988) was selected due to the availability of a multidimensional scaling solution for the stimuli, see nosof88train.

Experiment 1 of Nosofsky(1988) involved the classification of Munsell chips of fixed hue (5R) varying in brightness (value) and saturation (chroma). Instantiation frequency was manipulated between subjects. In condition B, all stimuli were equally frequent. In condition E2 (E7), stimulus 2 (7) was approximately five times as frequent as each of the other stimuli. In condition E2 (E7), stimulus 4 (9) indexes the indirect effect. There were three blocks of training. Block length was 48 trials for condition B and 63 trials for conditions E2 and E7. The training phase was followed by a transfer phase, which is not included in this CIRP (see Nosofsky, 1988, for details).

The data are as shown in Table 1 of Nosofsky (1988). The data are mean response probabilities for each stimulus in the training phase, averaged across blocks and participants.

Author(s)

Andy J. Wills <andy@willslab.co.uk>

Source

Nosofsky, R.M. (1988). Similarity, frequency, and category representations, *Journal of Experimental Psychology: Learning, Memory and Cognition, 14*, 54-65.

References

Barsalou, L.W. (1985). Ideals, central tendency, and frequency of instantiation as determinants of graded structure in categories. *Journal of Experimental Psychology: Learning, Memory & Cognition, 11*, 629-654.

Wills et al. (n.d.). Benchmarks for category learning. *Manuscript in preparation*.

See Also

nosof88train, nosof88oat

---

nosof88exalcove *Simulation of CIRP nosof88 with ex-ALCOVE model*

Description

Runs a simulation of the nosof88 CIRP using the slpALCOVE model implementation as an exemplar model and nosof88train as the input representation.

Usage

```
nosof88exalcove(params = NULL)
```

Arguments

params	A vector containing values for c, phi, la, and lw, in that order, e.g. params = c(2.1, 0.6, 0.09, 0.9). See slpALCOVE for an explanation of these parameters. Where params = NULL, best-fitting parameters are derived from optimzation archive nosof88exalcove_opt

Details

An exemplar-based simulation using slpALCOVE and nosof88train. The co-ordinates for the radial-basis units are taken from the multdimensional scaling solution for these stimuli reported by Nosofsky (1987).

Other parameters of slpALCOVE are set as follows: r = 2, q = 1, initial alpha = 1 / (number of input dimensions), initial w = 0. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the nosof88exalcove function, and hence can't be changed without re-writing the function.

This simulation is reported in Wills & O'Connell (n.d.).

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in nosof88.

Author(s)

Andy Wills & Garret O'Connell

References

Nosofsky, R.M. (1987). Attention and learning processes in the identification and categorization of integral stimuli, *Journal of Experimental Psychology: Learning, Memory and Cognition, 13*, 87-108.

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

nosof88, nosof88oat, nosof88train, slpALCOVE

---

nosof88exalcove_opt *Parameter optimization of ex-ALCOVE model with nosof88 CIRP*

Description

Uses nosof88exalcove to find best-fitting parameters for the ex-ALCOVE model for the nosof88 CIRP.

Usage

```
nosof88exalcove_opt(recompute = FALSE)
```

Arguments

recompute	When set to TRUE, the function re-runs the optimization. When set to FALSE, the function returns a stored copy of the results of the optimization.

Details

This function is an archive of the optimization procedure used to derive the best-fitting parameters for the nosof88exalcove simulation; see Spicer et al. (2017) for a tutorial introduction to the concept of simulation archives.

Optimization used the L-BFGS-B method from the optim function of the standard R stats package. The objective function was sum of squared errors. Please inspect the source code for further details (e.g. type nosof88exalcove_opt). The optimization was repeated for 16 different sets of starting values.

Where recompute = TRUE, the function can take many hours to run, depending on your system, and there is no progress bar. You can use Task Manager (Windows) or equivalent if you want some kind of visual feedback that the code is working hard. The code uses all the processor cores on the local machine, so speed of execution is a simple function of clock speed times processor cores. So, for example, a 4 GHz i7 processor (8 virutal cores) will take a quarter of the time to run this compared to a 2 GHz i5 processor (4 virtual cores).

Value

A vector containing the best-fitting values for c, phi, la, and lw, in that order. See slpALCOVE for an explanation of these parameters.

Author(s)

Andy Wills

References

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (2017). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

`nosof88oat` *Ordinal adequacy test for simulations of nosof88 CIRP*

Description

Tests whether a model output passes the ordinal adequacy criteria for the `nosof88` CIRP.

Usage

```
nosof88oat(dta, xtdo=FALSE)
```

Arguments

`dta`	Matrix containing model output. The matrix must have the same format, row order, and column names, as `data(nosof88)`; with that proviso, the output of any simulation implementation can be handled by this function.
`xtdo`	eXTenDed Output: Either `TRUE` or `FALSE`

Details

This function implements the Wills & O'Connell (n.d.) ordinal adequacy tests for the `nosof88` CIRP. Specifically, a model passes this test if it passes all four component tests: 1. E2(2) > B(2), 2. E7(7) > B(7), 3. E2(4) > B(4), 4. E7(9) > B(9). These tests refer to classification accuracy for particular stimuli in particular experimental conditions. For example, E7(9) indicates stimulus 9 in experimental condition E7.

Alternatively, by setting `xtdo` to `TRUE`, this function returns the summary model predictions reported by Wills & O'Connell (2016).

Value

Where `xtdo=FALSE`, this function returns TRUE if the ordinal adequacy tests are passed, and FALSE otherwise.

Where `xtdo=TRUE`, this function returns a summary matrix. The columns are stimulus numbers. The rows ('B','E') indicate the baseline (equal frequency) condition ('B') and the experimental conditions ('E2' or 'E7', depending on the column).

Author(s)

Andy Wills and Garret O'Connell

References

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

`nosof88`

nosof88protoalcove *Simulation of CIRP nosof88 with proto-ALCOVE model*

Description

Runs a simulation of the nosof88 CIRP using the slpALCOVE model implementation as a prototype model and nosof88train as the input representation.

Usage

```
nosof88protoalcove(params = NULL)
```

Arguments

params	A vector containing values for c, phi, la, and lw, in that order, e.g. params = c(2.1, 0.6, 0.09, 0.9). See slpALCOVE for an explanation of these parameters. Where params = NULL, best-fitting parameters are derived from optimzation archive nosof88protoalcove_opt

Details

An prototype-based simulation using slpALCOVE and nosof88train. There is one radial-basis unit for each category, representing the prototype. These prototypes are calculated by taking the mean of the co-ordinates of the stimuli in a category, with the stimulus co-ordinates coming from the multdimensional scaling solution reported by Nosofsky (1987). The calculations of the means are weighted by the instantiation frequency of the stimuli. Hence, the prototypes for each condition of the experiment are different.

Other parameters of slpALCOVE are set as follows: r = 2, q = 1, initial alpha = 1 / (number of input dimensions), initial w = 0. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the nosof88protoalcove function, and hence can not be changed without re-writing the function.

This simulation is reported in Wills & O'Connell (n.d.).

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in nosof88.

Author(s)

Andy Wills & Garret O'Connell

References

Nosofsky, R.M. (1987). Attention and learning processes in the identification and categorization of integral stimuli, *Journal of Experimental Psychology: Learning, Memory and Cognition, 13*, 87-108.

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

nosof88, nosof88oat, nosof88train, slpALCOVE

`nosof88protoalcove_opt`

Parameter optimization of proto-ALCOVE model with nosof88 CIRP

Description

Uses nosof88protoalcove to find best-fitting parameters for the ex-ALCOVE model for the nosof88 CIRP.

Usage

```
nosof88protoalcove_opt(recompute = FALSE)
```

Arguments

recompute	When set to TRUE, the function re-runs the optimization. When set to FALSE, the function returns a stored copy of the results of the optimization.

Details

This function is an archive of the optimization procedure used to derive the best-fitting parameters for the nosof88protoalcove simulation; see Spicer et al. (2017) for a tutorial introduction to the concept of simulation archives.

Optimization used the L-BFGS-B method from the optim function of the standard R stats package. The objective function was sum of squared errors. Please inspect the source code for further details (e.g. type nosof88protoalcove_opt). The optimization was repeated for 16 different sets of starting values.

Where recompute = TRUE, the function can take many hours to run, depending on your system, and there is no progress bar. You can use Task Manager (Windows) or equivalent if you want some kind of visual feedback that the code is working hard. The code uses all the processor cores on the local machine, so speed of execution is a simple function of clock speed times processor cores. So, for example, a 4 GHz i7 processor (8 virutal cores) will take a quarter of the time to run this compared to a 2 GHz i5 processor (4 virtual cores).

Value

A vector containing the best-fitting values for c, phi, la, and lw, in that order. See `slpALCOVE` for an explanation of these parameters.

Author(s)

Andy Wills

References

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (2017). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

`nosof88train` *Input representation of nosof88 for models input-compatible with slpALCOVE.*

Description

Create randomized training blocks for CIRP `nosof88`, in a format suitable for the `slpALCOVE` model, and any other model that uses the same input representation format. The stimulus co-ordinates come from a MDS solution reported by Nosofsky (1987) for the same stimuli.

Usage

```
nosof88train(condition = 'B', blocks = 3, absval = -1, subjs = 1, seed =
4182, missing = 'geo')
```

Arguments

`condition`	Experimental condition 'B', 'E2', or 'E7', as defined by Nosofsky (1988).
`blocks`	Number of blocks to generate. Omit this argument to get the same number of blocks as the published study (3).
`absval`	Teaching value to be used where category is absent.
`subjs`	Number of simulated subjects to be run.
`seed`	Sets the random seed
`missing`	If set to 'geo', output missing dimension flags (see below)

Details

A matrix is produced, with one row for each trial, and with the following columns:

`ctrl` - Set to 1 (reset model) for trial 1, set to zero (normal trial) for all other trials.

`cond` - 1 = condition B, 2 = condition E2, 3 = condition E7

`blk` - training block

`stim` - stimulus number (as defined by Nosofsky, 1988)

`x1, x2` - input representation. These are the co-ordinates of an MDS solution for these stimuli (see Nosofsky, 1987).

`t1, t2` - teaching signal (1 = category present, absval = category absent)

`m1, m2` - Missing dimension flags (always set to zero in this experiment, indicating all input dimensions are present on all trials). Only produced if `missing = 'geo'`.

Although the trial ordering is random, a random seed is used, so multiple calls of this function with the same parameters should produce the same output. This is usually desirable for reproducibility and stability of non-linear optimization. To get a different order, use the seed argument to set a different seed.

This implementation assumes a block length of 64 trials for conditions E2 and E7, rather than the 63 trials reported by Nosofsky (1988).

This routine was originally developed to support simulations reported in Wills & O'Connell (n.d.).

Value

R by C matrix, where each row is one trial, and the columns contain model input.

Author(s)

Andy Wills & Garret O'Connell

References

Nosofsky, R.M. (1987). Attention and learning processes in the identification and categorization of integral stimuli, *Journal of Experimental Psychology: Learning, Memory and Cognition, 13*, 87-108.

Nosofsky, R.M. (1988). Similarity, frequency, and category representations, *Journal of Experimental Psychology: Learning, Memory and Cognition, 14*, 54-65.

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

`nosof88`, `nosof88oat`, `slpALCOVE`

nosof94 *Type I-VI category structure CIRP*

Description

Shepard et al. (1961) stated that where there are two, equal-sized categories constructed from the eight stimuli it is possible to produce from varying three binary stimulus dimensions, there are only six logically distinct category structures. Shepard et al. (1961) labeled these structures as Types I through VI (see e.g. Nosofsky et al., 1994, Figure 1, for details). The CIRP concerns the relative difficulty of learning these category structures, as indexed by classification accuracy. The result, expressed in terms of accuracy, is:

I > II > [III, IV, V] > VI

The experiment reported by Nosofsky et al. (1994) provides the data for this CIRP.

Usage

```
data(nosof94)
```

Format

A data frame with the following columns:

type Type of category structure, as defined by Shepard et al. (1961). Takes values : 1-6

block Training block. Takes values: 1-16

error Mean error probability, averaged across participants

Details

Wills et al. (n.d.) discuss the derivation of this CIRP. In brief, the effect has been independently replicagted. Nosofsky et al. (1994) was selected as the CIRP because it had acceptable sample size (N=40 per Type), and included simulations of the results with a number of different formal models. Inclusion of this dataset in `catlearn` thus permits a validation of `catlearn` model implementations against published simulations.

In Nosofsky et al. (1994) the stimuli varied in shape (squares or triangles), type of interior line (solid or dotted), and size (large or small). Each participant learned two problems. Each problem was trained with feedback, to a criterion of four consecutive sub-blocks of eight trials with no errors, or for a maximum of 400 trials.

The data are as shown in the first 16 rows of Table 1 of Nosofsky et al. (1994). Only the first 16 blocks are reported, for comparability with the model fitting reported in that paper. Where a participant reached criterion before 16 blocks, Nosofsky et al. assumed they would have made no further errors if they had continued.

Author(s)

Andy J. Wills <andy@willslab.co.uk>

Source

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepaard, Hovland, and Jenkins (1961). *Memory and Cognition, 22*, 352-369.

References

Shepard, R.N., Hovland, C.I., & Jenkins, H.M. (1961). learning and memorization of classifications. *Psychological Monographs, 75*, Whole No. 517.

Wills et al. (n.d.). Benchmarks for category learning. *Manuscript in preparation*.

See Also

nosof94train, nosof94oat

---

nosof94bnalcove *Simulation of CIRP nosof94 with BN-ALCOVE model*

Description

Runs a simulation of the nosof94 CIRP using the slpALCOVE model implementation as an exemplar model and nosof94train as the input representation. This simulation replicates the one reported by Nosofsky et al. (1994).

Usage

```
nosof94bnalcove(params = c(6.33,0.011,0.409,0.179))
```

Arguments

params	A vector containing values for c, phi, la, and lw, in that order. See slpALCOVE for an explanation of these parameters.

Details

An exemplar-based simulation using slpALCOVE and nosof94train. The co-ordinates for the radial-basis units are assumed, and use the same binary representation as the abstract category structure.

The defaults for params are the best fit of the model to the nosof94 CIRP. The derivation of this fit is described by Nosofsky et al. (1994).

The other parameters of slpALCOVE are set as follows: r = 1, q = 1, initial alpha = 1 / number of dimensions, initial w = 0. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the nosof88bnalcove function, and hence can't be changed without re-writing the function.

This is a replication of the simulation reported by Nosofsky et al. (1994). Compared to other published simulations with the ALCOVE model, their simulation is non-standard in a number of respects:

1. A background noise ('BN') decision rule is used (other simulations use an exponential ratio rule).

2. As a consequence of #1, absence of a category label is represented by a zero (other simulations use -1).

3. The sum of the attentional weights is constrained to be 1 on every trial (other simulations do not apply this constraint).

The current simulation replicates these non-standard aspects of the Nosofsky et al. (1994) simulation.

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in nosof94.

Author(s)

Andy Wills

References

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepaard, Hovland, and Jenkins (1961). *Memory and Cognition, 22*, 352–369

See Also

nosof94, nosof94oat, nosof94train, slpALCOVE, nosof94bnalcove

---

nosof94exalcove *Simulation of CIRP nosof94 with ex-ALCOVE model*

Description

Runs a simulation of the nosof94 CIRP using the slpALCOVE model implementation as an exemplar model and nosof94train as the input representation.

Usage

```
nosof94exalcove(params = NULL)
```

Arguments

params	A vector containing values for c, phi, la, and lw, in that order, e.g. params = c(2.1, 0.6, 0.09, 0.9). See slpALCOVE for an explanation of these parameters. Where params = NULL, best-fitting parameters are derived from optimzation archive nosof94exalcove_opt

Details

N.B.: This simulation uses a standard version of ALCOVE. For a replication of the ALCOVE simulation of these data reported by Nosofsky et al. (1994), which is non-standard in a number of respects, see nosof94bnalcove.

An exemplar-based simulation using slpALCOVE and nosof94train. The co-ordinates for the radial-basis units are assumed, and use the same binary representation as the abstract category structure.

Other parameters of slpALCOVE are set as follows: r = 1, q = 1, initial alpha = 1/3, initial w = 0. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the nosof88exalcove function, and hence can't be changed without re-writing the function.

This simulation is reported in Wills & O'Connell (n.d.).

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in nosof94.

Author(s)

Andy Wills

References

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepaard, Hovland, and Jenkins (1961). *Memory and Cognition, 22*, 352–369

Wills, A.J. & O'Connell (n.d.). Averaging abstractions. *Manuscript in preparation*.

See Also

nosof94, nosof94oat, nosof94train, slpALCOVE, nosof94bnalcove

nosof94exalcove_opt *Parameter optimization of ex-ALCOVE model with nosof94 CIRP*

Description

Uses nosof94exalcove to find best-fitting parameters for the ex-ALCOVE model for the nosof94 CIRP.

Usage

```
nosof94exalcove_opt(recompute = FALSE, xtdo = FALSE)
```

Arguments

recompute	When set to TRUE, the function re-runs the optimization. When set to FALSE, the function returns a stored copy of the results of the optimization.
xtdo	eXTenDed Output; where set to TRUE, some further details of the optimization procedure are printed to the console.

Details

This function is an archive of the optimization procedure used to derive the best-fitting parameters for the nosof94exalcove simulation; see Spicer et al. (2017) for a tutorial introduction to the concept of simulation archives.

Optimization used the L-BFGS-B method from the optim function of the standard R stats package. The objective function was sum of squared errors. Please inspect the source code for further details (e.g. type nosof94exalcove_opt). The optimization was repeated for 15 different sets of starting values.

Where recompute = TRUE, the function can take many hours to run, depending on your system, and there is no progress bar. You can use Task Manager (Windows) or equivalent if you want some kind of visual feedback that the code is working hard. The code uses all the processor cores on the local machine, so speed of execution is a simple function of clock speed times processor cores. So, for example, a 4 GHz i7 processor (8 virutal cores) will take a quarter of the time to run this compared to a 2 GHz i5 processor (4 virtual cores).

Value

A vector containing the best-fitting values for c, phi, la, and lw, in that order. See slpALCOVE for an explanation of these parameters.

Author(s)

Andy Wills

References

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (2017). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

`nosof94oat` *Ordinal adequacy test for simulations of nosof94 CIRP*

Description

Tests whether a model output passes the ordinal adequacy criteria for the `nosof94` CIRP.

Usage

```
nosof94oat(dta, xtdo=FALSE)
```

Arguments

`dta`	Matrix containing model output. The matrix must have the same format, row order, and column names, as `data(nosof94)`; with that proviso, the output of any simulation implementation can be handled by this function.
`xtdo`	eXTenDed Output: Either `TRUE` or `FALSE`

Details

This function implements a standard ordinal adequacy test for the `nosof94` CIRP. Specifically, a model passes this test if the mean errors (averaged across blocks), obey the following:

I < II < [III, IV, V] < VI

Note that '[III, IV, V]' indicates that the these three problems can be in any order of difficulty (or all be of equal difficulty), as long as all three are harder than Problem 2 and all three are easier than Problem 6.

Alternatively, by setting `xtdo` to `TRUE`, this function returns the mean classification error by Problem type.

Value

Where `xtdo=FALSE`, this function returns TRUE if the ordinal adequacy tests are passed, and FALSE otherwise.

Where `xtdo=TRUE`, this function returns a summary matrix, containing mean errors (across blocks) for each of the six problem types.

Author(s)

Andy Wills

See Also

nosof94

---

nosof94plot *Plot Nosofsky et al. (1994) data / simulations*

Description

Produce a line graph similar to that shown in Nosofsky et al. (1994, Figures 1, 6-9).

Usage

```
nosof94plot(results,title = 'Nosofsky et al. (1994)')
```

Arguments

results	Mean error probability by block and problem, in the same format as data set nosof94
title	Title to appear at top of plot

Author(s)

Andy Wills

References

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepaard, Hovland, and Jenkins (1961). *Memory and Cognition, 22*, 352–369.

Examples

```
data(nosof94)
nosof94plot(nosof94)
```

nosof94sustain *Simulation of CIRP nosof94 with the SUSTAIN model*

Description

Runs a simulation of the nosof94 CIRP using the slpSUSTAIN model implementation and nosof94train as the input representation.

Usage

```
nosof94sustain(params = c(9.01245, 1.252233, 16.924073, 0.092327))
```

Arguments

params	A vector containing values for r, beta, d, and eta, in that order, e.g. params = c(8.1, 1.5, 9.71, 0.8). See slpSUSTAIN for an explanation of these parameters.

Details

NOTE: The underlying slpSUSTAIN function is currently written in R, and hence this simulation will take several minutes to run. slpSUSTAIN may be converted to C++ in a future release, which will reduce the run time of this simulation to a few seconds.

A simulation using slpSUSTAIN and nosof94train, i.e. a simulation of Nosofsky et al. (1994) with the Love et al. (2004) SUTAIN model.

Other parameters of slpSUSTAIN are set as follows: tau = 0, initial lambda = 1, initial w = 0, inital cluster centered on the first stimulus presented to the siumulated subject. These values are conventions of modelling with SUSTAIN, and should not be considered as free parameters. They are set within the nosof94sustain function, and hence can't be changed without re-writing the function.

The simulation uses 100 simulated subjects. Like the simulations nosof94exalcove and nosof94protoalcove, all simulated participants complete 16 blocks of training. This differs from the Nosofsky et al. (1994) experiment, in which participants are trained to a criterion of four consecutive errorless 8-trial subblocks.

The simulation by Gureckis (2014) builds this criterion-based training into their simulation by using a random number generator to turn the response probability on each trial into a correct or incorrect response. This feature of the Gureckis (2014) simulation is not incorporated here, because the instability in ouput this generates makes parameter optimization (e.g. via optim) less reliable.

A comparison of 10,000 simulated participants in the Gureckis (2014) simulation with 1,000 simulated participants in the current simulation reveals a mean difference in the 96 reported response probabilities of less than 0.01.

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in nosof94.

Author(s)

Lenard Dome, Andy Wills

References

Love, B. C., Medin, D. L., & Gureckis, T. M. (2004). SUSTAIN: a network model of category learning. *Psychological Review, 111*, 309-332.

Gureckis, T. M. (2014). sustain_python. *https://github.com/NYUCCL/sustain_python*

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepaard, Hovland, and Jenkins (1961). *Memory and Cognition, 22*, 352–369.

See Also

nosof94, nosof94oat, nosof94train, slpALCOVE, nosof94bnalcove

---

nosof94train — *Input representation of nosof94 for models input-compatible with slpALCOVE or slpSUSTAIN*

Description

Create randomized training blocks for CIRP nosof94, in a format suitable for the slpALCOVE or slpSUSTAIN models, and other models that use the same input representation formats.

Usage

```
nosof94train(cond = 1, blocks = 16, absval = -1, subjs = 1, seed = 7624,
missing = 'geo', blkstyle = 'accurate')
```

Arguments

cond	Category structure type (1-6), as defined by Shepard et al. (1961).
blocks	Number of blocks to generate. Omit this argument to get the same number of blocks (16) as used in the simulations reported by Nosofsky et al. (1994).
absval	Teaching value to be used where category is absent.
subjs	Number of simulated subjects to be run.
seed	Sets the random seed.
missing	If set to 'geo', output missing dimension flags (see below). If set to 'pad', use the padded stimulus representation format of slpSUSTAIN. If set to 'pad', set absval to zero.
blkstyle	If set to 'accurate', reproduce the randomization of this experiment, as described in Nosofsky et al. (1994). If set to 'eights', use instead the randomization used in the Gureckis (2016) simulation of this experiment.

Details

A matrix is produced, with one row for each trial, and with the following columns:

`ctrl` - Set to 1 (reset model) for trial 1 of each simulated subject, set to zero (normal trial) for all other trials.

`blk` - training block

`stim` - Stimulus number, ranging from 1 to 8. The numbering scheme is the same as in Nosofsky et al. (1994, Figure 1), under the mapping of dim_1_left = 0, dim_1_right = 1, dim_2_front = 0, dim_2_back = 1, dim_3_bottom = 0, dim_3_top = 1.

`x1, x2, ...` - input representation. Where `missing='geo'`, x1, x2, and x3 are returned, each set at 1 or 0. This is the binary dimensional representation required by models such as slpALCOVE, where e.g. x2 is the value on the second dimension. Where `missing='pad'`, x1, x2, y1, y2, z1, z2, are returned. This is the padded represenation required by models such as slpSUSTAIN; e.g. y1 and y2 represent the two possible values on dimension 2, so if y1 is black, y2 is white, and the stimulus is white, then [y1, y2] = [0, 1].

`t1, t2` - Category label (1 = category present, absval = category absent)

`m1, m2, m3` - Missing dimension flags (always set to zero in this experiment, indicating all input dimensions are present on all trials). Only produced if `missing = 'geo'`.

Although the trial ordering is random, a random seed is used, so multiple calls of this function with the same parameters should produce the same output. This is usually desirable for reproducibility and stability of non-linear optimization. To get a different order, use the seed argument to set a different seed.

This routine was originally developed to support Wills et al. (n.d.).

Value

R by C matrix, where each row is one trial, and the columns contain model input.

Author(s)

Andy Wills, Lenard Dome

References

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepaard, Hovland, and Jenkins (1961). *Memory and Cognition, 22*, 352–369

Gureckis, T. (2016). https://github.com/NYUCCL/sustain_python

Wills et al. (n.d.). Benchmarks for category learning. *Manuscript in preparation*.

See Also

`nosof94train`, `nosof94oat`

shin92 *Category size CIRP*

Description

Category size is the number of examples of a category that have been presented to the participant. The category-size effect (e.g. Homa et al., 1973) is the phenomenon that, as category size increases, the accuracy of generalization to new members of that category also increases. The equal-frequency conditions of Experiment 3 of Shin & Nosofsky (1992) provides the data for this CIRP.

Usage

```
data(shin92)
```

Format

A data frame with the following columns:

catsize Experimental condition (category size). Takes values : 3, 10

cat Category membership of stimulus. Takes values: 1, 2

stim Stimulus code, as defined by Shin & Nosofsky (1992). Stimuli beginning 'RN' or 'URN' are the 'novel' stimuli. Stimuli beginning 'P' are prototypes. The remaining stimuli are the 'old' (training) stimuli.

c2acc Mean probability, across participants, of responding that the item belongs to category 2.

Details

Wills et al. (2017) discuss the derivation of this CIRP, with Wills et al. (n.d.) providing further details. In brief, the effect has been independently replicated. Experiment 3 of Shin & Nosofsky (1992) was selected due to the availability of a multi-dimensional scaling solution for the stimuli, see shin92train.

Experiment 3 of Shin & Nosofsky (1992) involved the classification of nine-vertex polygon stimuli drawn from two categories. Category size was manipulated between subjects (3 vs. 10 stimuli per category). Participants received eight blocks of training, and three test blocks.

The data are as shown in Table 10 of Shin & Nosofsky (1992). The data are mean response probabilities for each stimulus in the test phase, averaged across test blocks and participants.

Author(s)

Andy J. Wills <andy@willslab.co.uk>

Source

Shin, H.J. & Nosofsky, R.M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General, 121*, 278-304.

References

Wills et al. (n.d.). Benchmarks for category learning. *Manuscript in preparation*.

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation, 66*, 79-115.

See Also

shin92train, shin92oat

`shin92exalcove` *Simulation of CIRP shin92 with ex-ALCOVE model*

Description

Runs a simulation of the shin92 CIRP using the slpALCOVE model implementation as an exemplar model and shin92train as the input representation.

Usage

```
shin92exalcove(params = NULL)
```

Arguments

params	A vector containing values for c, phi, la, and lw, in that order, e.g. params = c(2.1, 0.6, 0.09, 0.9). See slpALCOVE for an explanation of these parameters. Where params = NULL, best-fitting parameters are derived from optimzation archive shin92exalcove_opt

Details

An exemplar-based simulation using slpALCOVE and shin92train. The co-ordinates for the radial-basis units are derived from the test stimuli in shin92train. The output is the average of 100 simulated subjects.

The defaults for params are the best fit of the model to the shin92 CIRP. They were derived through minimization of SSE using non-linear optimization from 16 different initial states (using code not included in this archive).

The other parameters of slpALCOVE are set as follows: r = 2, q = 1, initial alpha = 1 / (number of input dimensions), inital w = 0. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the shin92exaclove function, and hence can't be changed without re-writing the function.

This simulation was reported in Wills et al. (2017).

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in shin92.

Author(s)

Andy Wills & Garret O'Connell

References

Shin, H.J. & Nosofsky, R.M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General, 121*, 278–304.

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation, 66*, 79-115.

---

shin92exalcove_opt *Parameter optimization of ex-ALCOVE model with shin92 CIRP*

Description

Uses shin92exalcove to find best-fitting parameters for the ex-ALCOVE model for the shin92 CIRP.

Usage

```
shin92exalcove_opt(params = c(2, 1, 0.25, 0.75), recompute = FALSE,
trace = 0)
```

Arguments

params	A vector containing the initial values for c, phi, la, and lw, in that order. See slpALCOVE for an explanation of these parameters. Where recompute is FALSE, this argument has no effect.
recompute	When set to TRUE, the function re-runs the optimization (which takes about 25 minutes on a 2.4 GHz processor). When set to FALSE, the function returns a stored copy of the results of the optimization (which is instantaneous).
trace	Sets the level of tracing information (i.e. information about the progress of the optimization), as defined by the optim function. Set to 6 for maximally verbose output. Where recompute is FALSE, this argument has no effect.

Details

This function is an archive of the optimization procedure used to derive the best-fitting parameters for the shin92exalcove simulation; see Spicer et al. (2017) for a tutorial introduction to the concept of simulation archives.

Optimization used the L-BFGS-B method from the optim function of the standard R stats package. The objective function was sum of squared errors. Please inspect the source code for further details (e.g. type shin92exalcove_opt).

This function was run in 16 times from different starting points, using 8 threads on a Core i7 3.6 GHz processor. The default parameters of this function are those for the best fit from those 16 starting points. The 16 starting points were

```
pset <- rbind( c(2,1,.25,.25),c(2,1,.25,.75),c(2,1,.75,.25),c(2,1,.75,.75), c(2,3,.25,.25),c(2,3,.25,
c(8,1,.25,.25),c(8,1,.25,.75),c(8,1,.75,.25),c(8,1,.75,.75), c(8,3,.25,.25),c(8,3,.25,.75),c(8,3,.75
)
```

not all of which converged successfully.

Value

A vector containing the best-fitting values for c, phi, la, and lw, in that order. See slpALCOVE for an explanation of these parameters.

Author(s)

Andy Wills

References

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (2017). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

---

shin92oat *Ordinal adequacy test for simulations of shin92 CIRP*

Description

Tests whether a model output passes the ordinal adequacy criterion for the shin92 CIRP.

Usage

```
shin92oat(dta, xtdo=FALSE)
```

Arguments

dta	Matrix containing model output. The matrix must have the same format, row order, and column names, as that returned by shin92exalcove; with that proviso, the output of any simulation implementation can be handled by this function.
xtdo	eXTenDed Output: Either TRUE or FALSE

Details

This function implements the Wills et al. (2017) ordinal adequacy test for the shin92 CIRP. Specifically, a model passes this test if response accuracy is higher for novel items from the size-10 condition than novel items from the size-3 condition.

Alternatively, by setting xtdo to TRUE, this function returns the summary model predictions reported by Wills et al. (2017).

Value

Where xtdo=FALSE, this function returns TRUE if the ordinal adequacy test is passed, and FALSE otherwise.

Where xtdo=TRUE, this function returns a summary matrix. The rows are the two category sizes, the columns are the three principal stimulus types (old, prototype, new), and the values are predicted accuracy scores.

Author(s)

Andy Wills and Garret O'Connell

References

Shin, H.J. & Nosofsky, R.M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General, 121*, 278–304.

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation, 66*, 79-115.

See Also

shin92

---

shin92protoalcove *Simulation of CIRP shin92 with proto-ALCOVE model*

Description

Runs a simulation of the shin92 CIRP using the slpALCOVE model implementation as a prototype model and shin92train as the input representation.

Usage

```
shin92protoalcove(params = NULL)
```

Arguments

params	A vector containing values for c, phi, la, and lw, in that orderr, e.g. params = c(2.1, 0.6, 0.09, 0.9). See slpALCOVE for an explanation of these parameters. Where params = NULL, best-fitting parameters are derived from optimzation archive shin92exalcove_opt

Details

An exemplar-based simulation using slpALCOVE and shin92train. The co-ordinates for the radial-basis units for the two prototypes are derived from the arithmetic means of the test stimuli in shin92train. The output is the average of 100 simulated subjects.

The defaults for params are the best fit of the model to the shin92 CIRP. They were derived through minimization of SSE using non-linear optimization from 16 different initial states (using code not included in this archive).

The other parameters of slpALCOVE are set as follows: r = 2, q = 1, initial alpha = 1 / (number of input dimensions), inital w = 0. These values are conventions of modeling with ALCOVE, and should not be considered as free parameters. They are set within the shin92exaclove function, and hence can't be changed without re-writing the function.

This simulation was reported in Wills et al. (2017).

Value

A matrix of predicted response probabilities, in the same order and format as the observed data contained in shin92.

Author(s)

Andy Wills & Garret O'Connell

References

Shin, H.J. & Nosofsky, R.M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General, 121*, 278–304.

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation, 66*, 79-115.

shin92protoalcove_opt *Parameter optimization of proto-ALCOVE model with shin92 CIRP*

Description

Uses shin92protoalcove to find best-fitting parameters for the proto-ALCOVE model for the shin92 CIRP.

Usage

```
shin92protoalcove_opt(params = c(2,1,.25,.75), recompute = FALSE,
trace = 0)
```

Arguments

params	A vector containing the initial values for c, phi, la, and lw, in that order. See slpALCOVE for an explanation of these parameters. Where recompute is FALSE, this argument has no effect.
recompute	When set to TRUE, the function re-runs the optimization (which takes about 10 minutes on a 2.4 GHz processor). When set to FALSE, the function returns a stored copy of the results of the optimization (which is instantaneous).
trace	Sets the level of tracing information (i.e. information about the progress of the optimization), as defined by the optim function. Set to 6 for maximally verbose output. Where recompute is FALSE, this argument has no effect.

Details

This function is an archive of the optimization procedure used to derive the best-fitting parameters for the shin92protoalcove simulation; see Spicer et al. (2017) for a tutorial introduction to the concept of simulation archives.

Optimization used the L-BFGS-B method from the optim function of the standard R stats package. The objective function was sum of squared errors. Please inspect the source code for further details (e.g. type shin92protoalcove_opt).

This function was run in 16 times from different starting points, using 8 threads on a Core i7 3.6 GHz processor. The default parameters of this function are those for the best fit from those 16 starting points. The 16 starting points were

```
pset <- rbind( c(2,1,.25,.25),c(2,1,.25,.75),c(2,1,.75,.25),c(2,1,.75,.75), c(2,3,.25,.25),c(2,3,.25,
c(8,1,.25,.25),c(8,1,.25,.75),c(8,1,.75,.25),c(8,1,.75,.75), c(8,3,.25,.25),c(8,3,.25,.75),c(8,3,.75
)
```

not all of which converged successfully.

Value

A vector containing the best-fitting values for c, phi, la, and lw, in that order. See slpALCOVE for an explanation of these parameters.

Author(s)

Andy Wills

References

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (2017). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

`shin92train` *Input representation of shin92 for models input-compatible with slpALCOVE.*

Description

Creates randomized training and transfer blocks for CIRP `shin92` , in a format suitable for the `slpALCOVE` model, and any other model that uses the same input representation format. The stimulus co-ordinates come from a MDS solution reported by Shin & Nosofsky (1992).

Usage

```
shin92train(condition = 'equal3', learn.blocks = 8, trans.blocks = 3,
        absval = -1, format = 'mds', subjs = 1, seed = 8416, missing =
        'geo')
```

Arguments

`condition`	Experimental condition 'equal3', 'equal10', 'unequal3', or 'unequal10', as defined by Shin & Nosofsky (1992).
`learn.blocks`	Number of training blocks to generate. Omit this argument to get the same number of training blocks as the published study (8).
`trans.blocks`	Number of transfer blocks to generate. Omit this argument to get the same number of transfer blocks as the published study (3).
`absval`	Teaching value to be used where category is absent.
`format`	Specifies format used for input representation. Only one format is currently suported, so this option is provided solely to support future development.
`subjs`	Number of simulated subjects to be run.
`seed`	Sets the random seed
`missing`	If set to 'geo', output missing dimension flags (see below)

Details

A matrix is produced, with one row for each trial, and with the following columns:

`ctrl` - Set to 1 (reset model) for trial 1, set to zero (normal trial) for all other training trials, and set to 2 (freeze learning) for all transfer trials.

`cond` - 1 = equal3, 2 = equal10, 3 = unequal3, 4 = unequal10

`phase` - 1 = training, 2 = transfer

`blk` - block of trials

`stim` - stimulus number; these correspond to the rows in Tables A3 and A4 of Shin & Nosofsky (1992)

`x1 ... x6` - input representation. These are the co-ordinates of an MDS solution for these stimuli (see Shin & Nosofsky, 1992, Tables A3 and A4). Note: Size 3 conditions have a four-dimensional MDS solution, so the output is x1 ... x4

`t1, t2` - teaching signal (1 = category present, absval = category absent)

`m1 ... m6` - Missing dimension flags (always set to zero in this experiment, indicating all input dimensions are present on all trials). Note: ranges from m1 to m4 for Size 3 conditions. Only produced if `missing = 'geo'`.

Although the trial ordering is random, a random seed is used, so multiple calls of this function with the same parameters should produce the same output. This is usually desirable for reproducibility and stability of non-linear optimization. To get a different order, use the seed argument to set a different seed.

This function was originally developed to support simulations reported in Wills et al. (2017).

Value

R by C matrix, where each row is one trial, and the columns contain model input.

Author(s)

Andy Wills

References

Shin, H.J. & Nosofsky, R.M. (1992). Similarity-scaling studies of dot-pattern classification and recognition. *Journal of Experimental Psychology: General, 121*, 278-304.

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation, 66*.

See Also

`shin92`, `shin92oat`, `slpALCOVE`

`slpALCOVE` *ALCOVE category learning model*

Description

Kruschke's (1992) category learning model.

Usage

```
slpALCOVE(st, tr, dec = 'ER', humble = TRUE, attcon = FALSE, absval = -1,
          xtdo = FALSE)
```

Arguments

`st`	List of model parameters
`tr`	R-by-C matrix of training items
`dec`	String defining decision rule to be used
`humble`	Boolean specifying whether a humble or strict teacher is to be used
`attcon`	Boolean specifying whether attention is constrained
`absval`	Real number specifying teaching value for category absence
`xtdo`	Boolean specifying whether to write extended information to the console (see below).

Details

The coverage in this help file is relatively brief; Catlearn Research Group (2016) provides an introduction to the mathematics of the ALCOVE model, whilst a more extensive tutorial on using slpALCOVE can be found in Wills et al. (2016).

The functions works as a stateful list processor. Specifically, it takes a matrix as an argument, where each row is one trial for the network, and the columns specify the input representation, teaching signals, and other control signals. It returns a matrix where each row is a trial, and the columns are the response probabilities at the output units. It also returns the final state of the network (attention and connection weights), hence its description as a 'stateful' list processor.

Argument `st` must be a list containing the following items:

`colskip` - skip the first N columns of the tr array, where N = colskip. colskip should be set to the number of optional columns you have added to matrix tr, PLUS ONE. So, if you have added no optional columns, colskip = 1. This is because the first (non-optional) column contains the control values, below.

`c` - specificity constant (Kruschke, 1992, Eq. 1). Positive real number. Scales psychological space.

`r` - distance metric (Kruschke, 1992, Eq. 1). Set to 1 (city-block) or 2 (Euclidean).

`q` - similarity gradient (Kruschke, 1992, Eq. 1). Set to 1 (exponential) or 2 (Gaussian).

phi - decision constant. For decision rule ER, it is referred to as mapping constant phi, see Kruschke (1992, Eq. 3). For decision rule BN, it is referred to as the background noise constant b, see Nosofsky et al. (1994, Eq. 3).

lw - associative learning rate (Kruschke, 1992, Eq. 5) . Real number between 0 and 1.

la - attentional learning rate (Kruschke, 1992, Eq. 6). Real number between 0 and 1.

h - R by C matrix of hidden node locations in psychological space, where R = number of input dimensions and C = number of hidden nodes.

alpha - vector of length N giving initial attention weights for each input dimension, where N = number of input dimensions. If you are not sure what to use here, set all values to 1.

w - R by C matrix of initial associative strengths, where R = number of output units and C = number of hidden units. If you are not sure what to use here, set all values to zero.

Argument tr must be a matrix, where each row is one trial presented to the network. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

ctrl - vector of control codes. Available codes are: 0 = normal trial, 1 = reset network (i.e. set attention weights and associative strengths back to their initial values as specified in h and w (see below)), 2 = Freeze learning. Control codes are actioned before the trial is processed.

opt1, opt2, ... - optional columns, which may have any names you wish, and you may have as many as you like, but they must be placed after the ctrl column, and before the remaining columns (see below). These optional columns are ignored by this function, but you may wish to use them for readability. For example, you might include columns for block number, trial number, and stimulus ID number. The argument colskip (see above) must be set to the number of optional columns plus 1.

x1, x2, ... - input to the model, there must be one column for each input unit. Each row is one trial.

t1, t2, ... - teaching signal to model, there must be one column for each output unit. Each row is one trial. If the stimulus is a member of category X, then the teaching signal for output unit X must be set to +1, and the teaching signal for all other output units must be set to absval.

m1, m2, ... - missing dimension flags, there must be one column for each input unit. Each row is one trial. Where m = 1, that input unit does not contribute to the activation of the hidden units on that trial. This permits modelling of stimuli where some dimensions are missing on some trials (e.g. where modelling base-rate negelct, Kruschke, 1992, p. 29–32). Where m = 0, that input unit contributes as normal. If you are not sure what to use here, set to zero.

Argument dec, if specified, must take one of the following values:

ER specifies an exponential ratio rule (Kruschke, 1992, Eq. 3).

BN specifies a background noise ratio rule (Nosofsky et al., 1994, Eq. 3). Any output activation lower than zero is set to zero before entering into this rule.

Argument humble specifies whether a humble or strict teacher is to be used. The function of a humble teacher is specified in Kruschke (1992, Eq. 4b). In this implementation, the value -1 in Equation 4b is replaced by absval.

Argument attcon specifies whether attention should be constrained or not. *If you are not sure what to use here, set to FALSE*. Some implementations of ALCOVE (e.g. Nosofsky et al., 1994) constrain the sum of the attentional weights to always be 1 (personal communication, R. Nosofsky,

June 2015). The implementation of attentional constraint in *alcovelp* is the same as that used by Nosofsky et al. (1994), and present as an option in the source code available from Kruschke's website (Kruschke, 1991).

Argument `xtdo` (eXTenDed Output), if set to TRUE, will output to the console the following information on every trial: (1) trial number, (2) attention weights at the end of that trial, (3) connection weights at the end of that trial, one row for each output unit. This output can be quite lengthy, so diverting the output to a file with the `sink` command prior to running `alcovelp` with extended output is advised.

Value

Returns a list containing three components: (1) matrix of response probabilities for each output unit on each trial, (2) attentional weights after final trial, (3) connection weights after final trial.

Author(s)

Andy Wills

References

Catlearn Research Group (2016). Description of ALCOVE. http://catlearn.r-forge.r-project.org/desc-alcove.pdf

Kruschke, J. (1991). *ALCOVE.c*. Retrieved 2015-07-20, page since removed, but archival copy here: https://web.archive.org/web/20150605210526/http://www.indiana.edu/~kruschke/articles/ALCOVE.c

Kruschke, J. (1992). ALCOVE: an exemplar-based connectionist model of category learning. *Psychological Review, 99*, 22-44

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepaard, Hovland, and Jenkins (1961). *Memory and Cognition, 22*, 352-369.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation, 66*, 79-115.

`slpBM` *Bush & Mosteller (1951) simple associative learning model*

Description

A model often attributed to Bush & Mosteller (1951), more precisely this is the separable error term learning equation discussed by authors such as Mackintosh (1975) and Le Pelley (2004); see Note 1.

Usage

```
slpBM(st, tr, xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	R matrix of training items
xtdo	Boolean specifying whether to include extended information in the output (see below)

Details

The function operates as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (tr) as an argument, where each row represents a single training trial, while each column represents the different types of information required by the model, such as the elemental representation of the training stimuli, and the presence or absence of an outcome. It returns the output activation on each trial (a.k.a sum of associative strengths of cues present on that trial), as a vector. The slpBM function also returns the final state of the model - a vector of associative strengths between each stimulus and the outcome representation.

Argument `st` must be a list containing the following items:

`lr` - the learning rate (fixed for a given simulation), as denoted by, for example, `theta` in Equation 1 of Mackintosh (1975). If you want different elements to differ in salience (different alpha values) use the input activations (x1, x2, ..., see below) to represent element-specific salience.

`w` - a vector of initial associative strengths. If you are not sure what to use here, set all values to zero.

`colskip` - the number of optional columns to be skipped in the tr matrix. colskip should be set to the number of optional columns you have added to the tr matrix, PLUS ONE. So, if you have added no optional columns, colskip=1. This is because the first (non-optional) column contains the control values (details below).

Argument `tr` must be a matrix, where each row is one trial presented to the model. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

`ctrl` - a vector of control codes. Available codes are: 0 = normal trial; 1 = reset model (i.e. set associative strengths (weights) back to their initial values as specified in w (see above)); 2 = Freeze learning. Control codes are actioned before the trial is processed.

`opt1, opt2, ...` - any number of preferred optional columns, the names of which can be chosen by the user. It is important that these columns are placed after the control column, and before the remaining columns (see below). These optional columns are ignored by the function, but you may wish to use them for readability. For example, you might choose to include columns such as block number, trial number and condition. The argument colskip (see above) must be set to the number of optional columns plus one.

`x1, x2, ...` - activation of any number of input elements. There must be one column for each input element. Each row is one trial. In simple applications, one element is used for each stimulus (e.g. a simulation of blocking (Kamin, 1969), A+, AX+, would have two inputs, one for A and one for X). In simple applications, all present elements have an activation of 1 and all absence elements have an activation of 0. However, slpBM supports any real number for activations, e.g. one might use values between 0 and 1 to represent differing cue saliences.

`t` - Teaching signal (a.k.a. lambda). Traditionally, 1 is used to represent the presence of the outcome, and 0 is used to represent the absence of the outcome, altough slpBM supports any real values for lambda..

Argument `xtdo` (eXTenDed Output) - if set to TRUE, function will return the associative strengths for the end of each trial (see Value).

Value

Returns a list containing two components (if xtdo = FALSE) or three components (if xtdo = TRUE, xout is also returned):

`st`	Vector of final associative strengths
`suma`	Vector of output activations for each trial
`xout`	Matrix of associative strengths at the end of each trial

Note

1. Bush & Mosteller's (1951) Equations 2 outputs response probability, not associative strength. Also, it has two learning rate paramters, `a` and `b`. At least to a first approximation, `b` serves a similar function to `beta-outcome-absent` in Rescorla & Wagner (1972), and `a-b` is similar to `beta-outcome-present` in that same model.

Author(s)

Lenard Dome, Stuart Spicer, Andy Wills

References

Bush, R. R., & Mosteller, F. (1951). A mathematical model for simple learning. *Psychological Review, 58(5)*, 313-323.

Kamin, L.J. (1969). Predictability, surprise, attention and conditioning. In Campbell, B.A. & Church, R.M. (eds.), *Punishment and Aversive Behaviour*. New York: Appleton-Century-Crofts, 1969, pp.279-296.

Le Pelley, M.E. (2004). The role of associative history in models of associative learning: A selective review and a hybrid model, *Quarterly Journal of Experimental Psychology, 57B*, 193-243.

Mackintosh, N.J. (1975). A theory of attention: Variations in the associability of stimuli with reinforcement, *Psychological Review, 82*, 276-298.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black & W. F. Prokasy (Eds.), *Classical conditioning II: Current research and theory* (pp. 64-99). New York: Appleton-Century-Crofts.

Spicer, S., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (n.d.). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation, 66*, 79-115.

slpCOVIS *COVIS category learning model*

Description

COmpetition between Verbal and Implicit Systems model of category learning (Ashby et al. 1998), as described in Ashby et al. (2011). The current implementation supports two-category experiments, and uses only single-dimension, not-below-chance, rules in the Explicit system.

Usage

```
slpCOVIS(st, tr, crx = TRUE, respt = FALSE, rgive = TRUE, xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	R-by-C matrix of training items
crx	Boolean. Explicit System. If set to TRUE, the current rule is included in the random selection of a rule to receive a weight increase from the Possion distribution. If set to FALSE, the current rule is not included in this random selection.
respt	Set to FALSE for the behaviour described in Note 5; behaviour when TRUE is undocumented
rgive	Set to TRUE; FALSE is undocumented
xtdo	Set to FALSE; TRUE is undocumented

Details

The coverage in this help file is relatively brief; for a more extensive tutorial, see Inkster et al. (n.d.).

The function works as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (tr) as an argument, where each row is one trial for the network, and the columns specify the input representation. It returns a List containing the predictions made by the model and the final state of the model, hence its description as a 'stateful' list processor.

Argument st must be a list containing the following information. Parameter names given in brackets in the descriptions below follow the naming conventions of Ashby et al. (2011), and Edmunds & Wills (2016). Equation numbers are from Ashby et al. (2011); where there is no equation, the page number is given instead.

Explicit system variables:

envar - (sigma^2_E) - p. 68 - Variance of the noise distribution used to determine which response the explicit system makes on the current trial. See Note 4, below.

decbound - (C) - Eq. 1 - location of the decision bound on a single dimension. In the current implementation of slpCOVIS, this location is the same for all dimensions.

corcon - (delta_c) - Eq. 2 - constant by which to increase current rule saliency in the case of a correct response.

errcon - (delta_e) - Eq. 3 - constant by which to decrease current rule saliency in the case of an incorrect response.

perscon - (gamma) - Eq. 4 - perseveration constant, i.e. value to add to the salience of the current rule to obtain its rule weight.

lambda - (lambda) - Eq. 5 - Mean of the Poission distribution. A value randomly sampled from the Poisson distribution is added to a randomly-selected rule when calculating the weights for new rule selection.

decsto - (a) - Eq. 7 - decision stochasticity when using rule weights to select the rule for the next trial. For Ashby et al. (2011)'s implementation, a = 1. For other uses, see Edmunds & Wills (2016).

Procedural system variables:

sconst - (alpha) - Eq. 8 - scaling constant for cortical unit activation. See Note 3, below.

invar - (sigma^2_p) - Eq. 9 - Variance of the normally-distributed noise used to calculate striatal unit activation.

dbase - (D_base) - Eq. 10 - baseline dopamine level.

alphaw - (alpha_w) - Eq. 10 - Learning rate parameter in force when striatal activation is above the NMDA threshold, and dopamine is above baseline.

betaw - (beta_w) - Eq. 10 - Learning rate parameter in force when striatal activation is above the NMDA threshold, and dopamine is below baseline.

gammaw - (gamma_w) - Eq. 10 - Learning rate parameter in force when striatal activation is between the AMPA and NMDA thresholds.

nmda - (theta_NMDA) - Eq. 10 - Activation threshold for post-synaptic NMDA.

ampa - (theta_AMPA) - Eq. 10 - Activation threshold for post-synaptic AMPA. See Note 1, below.

wmax - (w_max) - Eq. 10 - Intended upper weight limit for a cortico-striatal link. See Note 2, below.

prep - (P_(n-1)) - Eq. 12 - predicted reward value immediately prior to first trial. If unsure, set to zero.

prer - (R_(n-1)) - Eq. 12 - obtained reward value immediately prior to first trial. If unsure, set to zero.

Competition / decision system variables:

emaxval - p.77 - The maximum possible value of the the Explicit system's discriminant variable. For example, if the stimulus value varies from zero to one, and C (see above) is 0.5, then the maximum value is 1-0.5 = 0.5

etrust - (theta_E) - Eq. 15 - trust in the explicit system immediately prior to first trial. If unsure, set to .99.

itrust - (theta_P) - p. 77 - trust in the procedural system immediately prior to first trial. If unsure, set to .01. See also Note 7, below.

ocp - (delta_OC) - Eq. 15 - constant used to increase trust in the Explicit system after it suggests a response that turns out to be correct.

oep - (delta_OE) - Eq. 16 - constant used to decrease trust in the Explicit system after it suggests a response that turns out to be incorrect.

Initial state of model:

`initrules` - vector of length `stimdim`, representing the initial salience of each single-dimensional rule in the Explicit system.

`crule` - a number indicating which rule is in use immediately prior to the first trial (1 = dimension 1, 2 = dimension 2, etc). If this is not meaningful in the context of your simulation, set it to zero, and ensure ctrl = 1 in the first row of your training matrix (see below). This will then randomly pick an initial rule.

`initsy` - matrix of `stimdim` rows and two columns - contains the initial values for the cortico-striatal connection strengths.

`scups` - matrix of `stimdim` columns and as many rows as you wish to have cortical input units. Each row represents the position of a cortical unit in N-dimensional stimulus space.

And finally, a couple of things slpCOVIS needs to interpret your tr matrix (see below):

`stimdim` - number of stimulus dimensions in the input representation.

`colskip` - skip the first N columns of the tr array, where N = colskip. colskip should be set to the number of optional columns you have added to matrix tr, PLUS ONE. So, if you have added no optional columns, colskip = 1. This is because the first (non-optional) column contains the control values, see below.

Argument `tr` must be a matrix, where each row is one trial presented to the network. Trials are always presented to the model in the order specified. The columns must be as described below, in the order described below:

`ctrl` - vector of control codes. Available codes are: 0 = normal trial, 1 = reset network (i.e. set back to the state defined in list `st` and randomly select an initial rule for the Explicit System using Eq. 7) , 2 = Freeze learning. Control codes are actioned before the trial is processed.

`opt1, opt2, ...` - optional columns, which may have any names you wish, and you may have as many as you like, but they must be placed after the ctrl column, and before the remaining columns (see below). These optional columns are ignored by this function, but you may wish to use them for readability. For example, you might include columns for block number, trial number, and stimulus ID number. The argument colskip (see above) must be set to the number of optional columns plus 1.

`x1, x2, ...` - stimulus input to the model; there must be one column for each stimulus dimension.

`t1` - teaching signal to model. If the correct response is Category 1, t = 1. If the correct response is Category 2, t = -1. Experiments with something other than two categories are not supported in the current implementation.

`optend1, optend2, ...` - optional columns, which may have any names you wish, and you may have as many as you like, but they must be placed after the t1 column. These optional columns are ignored by this function, but may help with cross-compatibility with other model implementations. For example, the additional 't' and 'm' columns of input representations generated for slpALCOVE will be safely ignored by slpCOVIS.

Value

Returns a List containing eight components:

`foutmat` A two-column matrix, representing the model's response on each trial. For any given trial, [1,0] indicates a Category 1 response; [0,1] indicates a Category 2 response. Responses are reported in this manner to facilitate cross-compatibility with models that produce response probabilities on each trial.

frules	Explicit system - rule saliences after final trial
fsystr	Procedural system - cortico-striatal synaptic strengths after final trial)
fetrust	Decision system - trust in explicit system after final trial
fitrust	Decision system - trust in procedural system after final trial
frule	Explicit system - rule used by explicit system on final trial
fprep	Implicit system - predicted reward value on final trial
fprer	Implicit system - obtained reward value on final trial

Note

1. Ashby et al. (2011) state (p. 74) that the intended operation of COVVIS is theta_NMDA > theta_AMPA, but the values they report are theta_NMDA = .0022, theta_AMPA = .01.

2. Ashby et al. (2011) did not specify a value for w_max; Edmunds & Wills (2016) assumed the intended value was 1.

3. Ashby et al. (2011) do not use Eq. 8 in their simulation, they manually set sensory cortex activation to 1 for the presented stimulus and 0 for all the others (p. 78). They thus do not have a value for alpha. Edmunds & Wills (2016) set alpha to 0.14, which produces similar behaviour for 0,1 coded stimulus dimensions, without having to manually set the activations.

4. In Ashby et al. (2011) and Edmunds & Wills (2016), sigma^2_E is set to zero. In this implementation of slpRW, positive values should also work but have not been extensively tested.

5. In the descriptions provided by Ashby et al. (2011, p. 69 & p. 75), there is some ambiguity about the meaning of the term 'response' - does this mean the response of a system (e.g. the Explicit system), or the overall response (i.e. the output of the decision system). In the current implementation, the response of the Explicit System is compared to the feedback to determine whether the Explicit System was correct or incorrect, and the response of the Procedural System is compared to the feedback to determine whether the Procedural System was correct or incorrect.

6. It seems that in Ashby et al.'s (2011) simulations, each dimension generates only one single-dimension rule for a two-category problem, rather than two as one might expect (e.g. small = A, large = B, but also large = A, small = B). Rules that would produce below-chance responding are excluded from the rule set.

7. Ashby et al. (2011) state that theta_E + theta_P = 1. However, slpCOVIS does not perform this check on the initial state, so it is important to check this manually.

Author(s)

Angus Inkster, Andy Wills, Charlotte Edmunds

References

Ashby, F.G., Alfonso-Reese, L.A., Turken, A.U. & Waldron, E.M. (1998). A neuropsychological theory of multiple systems in category learning. *Psychological Review, 105*, 442-481.

Ashby, F.G., Paul, E.J., & Maddox, W.T. (2011). COVIS. In Pothos, E.M. & Wills, A.J. (2011). *Formal approaches in categorization*. Cambridge, UK: Cambridge University Press.

Edmunds, C.E.R., & Wills, A.J. (2016). Modeling category learning using a dual-system approach: A simulation of Shepard, Hovland and Jenkins (1961) by COVIS. In A. Papfragou, D. Grodner,

D. Mirman, & J.C. Trueswell (Eds.). *Proceedings of the 38th Annual Conference of the Cognitive Science Society* (pp. 69-74). Austin, TX: Cognitive Science Society.

Inkster, A.B., Edmunds, C.E.R., & Wills, A.J. (n.d.). A distributed-collaboration resource for dual-process modeling in category learning. *Manuscript in preparation*.

Pothos, E.M. & Wills, A.J.(2011). *Formal approaches in Categorisation.*Cambridge: University Press.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation, 66*, 79-115.

`slpDGCM` *Similarity-Dissimilarity Generalized Context Model (DGCM)*

Description

Stewart and Morin (2007)'s extension to Nosofsky's (1984, 2011) Exemplar-based Generalized Context Model. The implementation also contains O'Bryan et al. (2018)'s version of the Similarity-Dissimilarity Generalized Context Model, see Note 1.

Usage

```
slpDGCM(st, test, dec = "BIAS", exemplar_mute = FALSE, exemplar_decay = TRUE)
```

Arguments

`st`	List of model parameters
`test`	Test matrix.
`dec`	Decision mechanism. If `NOISE`, use O'Bryan et al. (2018)'s background-noise decision rule. If `BIAS` (default), use Stewart and Morin (2007)'s category-bias decision rule.
`exemplar_mute`	If `TRUE`, only those exemplars contribute evidence to the decision rule, which have at least one feature common with the current stimuli (O'Bryan et al., 2018). If `FALSE` (default), all exemplars contribute.
`exemplar_decay`	If `TRUE` (default), exemplar weightings decay as specified by Stewart and Morin (2007). If `FALSE`, exemplar weightings are static.

Details

This implementation houses the two version of DGCM. In order to use the instantiation of DGCM described in O'Bryan et al. (2018), set `exemplar_decay = FALSE` and `exemplar_mute = TRUE`. The default settings of the function will run the model that corresponds to Stewart and Morin (2007).

The functions works as a stateful list processor. Specifically, it takes a data frame as an argument, where each row is one trial for the model, and the columns specify the input representation, teaching signals, and other control signals. It returns two matrices containing, for each trial, response

probabilities and the accumulated evidence for each category. It also returns the final state of the network (e.g. memory decay), hence its description as a 'stateful' list processor, see Note 1.

This implementation took the assumption that when `exemplar_decay = TRUE`, memory strengths for exemplar are equal to each other at the beginning of the test phase. In future releases, we plan to implement a feature that allows initial memory strengths to be treated as freely varying parameters.

`st` must be a list containing the following items:

`attentional_weights` - vector of attentional weights, where sum of all elements equal to 1.

`c` - generalization constant.

`r` - The Minkowski metric parameter *r* gives a city block metric when r = 1 (used for separable-dimension stimuli) and a Euclidean metric when r = 2 (used for integral-dimension stimuli).

`s` - similarity and dissimilarity weighting. If 0, evidence for a category will be purely based on the dissimilarity between current input vector and all exemplars from the other categories. If it is 1, evidence for a given category will be solely based on similarity to its own exemplars.

`t` - exemplar weighting. If `memory_decay = FALSE`, it is a vector of exemplar-specific memory strength. If `memory_decay = TRUE` (default), it is a vector of exemplar-specific memory strengths that will update according to the function as specified in Equation 4 in Stewart and Morin (2007).

`beta` - category bias vector. Only used when `dec` set to BIAS, otherwise ignored. Currently, there is no restriction in place on what values are allowed in this implementation, but Stewart and Morin (2007) specifies that elements of `beta` should sum to 1.

`base` - a vector of baseline level of similarity. This parameter will control how much noise will spread over all categories in the background-noise decision rule. It is only used if `dec` is set to NOISE.

`gamma` - decision scaling constant. Only used when `dec` is set to BIAS.

`theta` - decay rate. If `exemplar_decay = FALSE`, theta is ignored.

`colskip` - the number of optional columns to skip in test plus one. If you have no optional columns, set it to one.

`outcomes` - the number of categories.

`exemplars` - a matrix of exemplars and their corresponding category indicated by a single integer.

`test` must be a `data.matix` with the following columns:

`opt1, opt2, ...` - any number of optional columns, the names of which can be chosen by the user. These optional columns are ignored by the slpDGCM function, but you may wish to use them for readability.

`x1, x2, x3, ...` - input to the model, there must be one column for each input unit. Each row is one trial. DGCM uses a nominal stimulus representation, which means that features are coded as either 0 (absent) or 1 (present).

Value

If `exemplar_decay = FALSE`, returns a list of the following matrices:

`v` A matrix of evidence accumulated for each category (columns) on each trial (rows) as output by Equation 3 in Stewart and Morin (2007).

`p` A matrix of response probabilities. Category responses (columns) for each trial (rows).

If `exemplar_decay = TURE`, the function also returns memory decay for each trial, `decay`.

Note

1. O'Bryan et al. (2018)'s version of the DGCM is not a stateful list processor, but we decided to include it in the same implementation. In fact, Stewart and Morin (2007)'s version only classifies as a stateful list processor, because of the memory decay function.

Author(s)

Lenard Dome, Andy Wills

References

Nosofsky, R. M. (1984). Choice, similarity, and the context theory of classification. *Journal of Experimental Psychology: Learning, memory, and cognition, 10*, 104.

O'Bryan, Sean R., et al. (2018). Model-based fMRI reveals dissimilarity processes underlying base rate neglect. *ELife 7*: e36395.

Stewart, N., & Morin, C. (2007). Dissimilarity is used as evidence of category membership in multidimensional perceptual categorization: A test of the similarity–dissimilarity generalized context model. *Quarterly Journal of Experimental Psychology, 60*, 1337-1346.

Examples

```
## Replicate O'Bryan et al. (2018)
# Exemplars
stim = matrix(c(
              1,1,0,0,0,0, 1,
              1,0,1,0,0,0, 2,
              0,0,0,1,1,0, 3,
              0,0,0,1,0,1, 4), ncol = 7, byrow = TRUE)

# Transfer/test stimuli
# This is a row for each unique transfer stimulus
tr = matrix(c(
             1, 1, 0, 0, 0, 0, #0,1,2
             1, 0, 1, 0, 0, 0, #3
             0, 0, 0, 1, 1, 0, #4,5,6
             0, 0, 0, 1, 0, 1, #7
             1, 0, 0, 0, 0, 0, #8
             0, 0, 0, 1, 0, 0, #9
             0, 1, 0, 0, 0, 0, #10
             0, 0, 1, 0, 0, 0, #11
             0, 0, 0, 0, 1, 0, #12
             0, 0, 0, 0, 0, 1, #13
             0, 1, 1, 0, 0, 0, #14, 15
             0, 0, 0, 0, 1, 1, #16, 17
             1, 0, 0, 0, 1, 0, #18
             1, 0, 0, 0, 0, 1, #19
             0, 1, 0, 1, 0, 0, #20
             0, 0, 1, 1, 0, 0, #21
             0, 0, 1, 0, 1, 0, #22, 23
             0, 1, 0, 0, 0, 1 #24, 25
```

```
              ),
              ncol = 6,
              byrow = TRUE)

  # parameters from paper
  aweights = c(0.27692188, 0.66524089, 0.88723335, 0.16967400, 0.71206208,
              0.87939732)

  st <- list(attentional_weights = aweights/sum(abs(aweights)),
             c = 9.04906080,
             s = 0.94614863,
             b = 0.02250668,
             t = c(3, 1, 3, 1),
             beta = c(1, 1, 1, 1)/4,
             gamma = 1,
             theta = 0.4,
             r = 1,
             colskip = 1,
             outcomes = 4,
             exemplars = stim)

  slpDGCM(st, tr, exemplar_decay = FALSE, exemplar_mute = TRUE, dec = "NOISE")
```

---

slpDIVA *DIVA category learning model*

Description

DIVergent Autoencoder (Kurtz, 2007; 2015) artificial neural network category learning model

Usage

```
slpDIVA(st, tr, xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	R-by-C matrix of training items
xtdo	When set to TRUE, produce extended output

Details

This function works as a stateful list processor (Wills et al., 2017). Specifically, it takes a matrix as an argument, where each row is one trial for the network, and the columns specify the input representation, teaching signals, and other control signals. It returns a matrix where each row is a trial, and the columns are the response probabilities for each category. It also returns the final state of the network (connection weights and other parameters), hence its description as a 'stateful' list processor.

Argument st must be a list containing the following items:

st must contain the following principal model parameters:

learning_rate - Learning rate for weight updates through backpropagation. The suggested learning rate default is learning_rate = 0.15

beta_val - Scalar value for the Beta parameter. beta_val controls the degree of feature focusing (not unlike attention) that the model uses to make classification decisions (see: Conaway & Kurtz, 2014; Kurtz, 2015). beta_val = 0 turns feature focusing off.

phi - Scalar value for the phi parameter. phi is a real-valued mapping constant, see Kruschke (1992, Eq. 3).

st must also contain the following information about network architecture:

num_feats - Number of input features.

num_hids - Number of hidden units. A rough rule of thumb for this hyperparameter is to start with num_feats = 2 and add additional units if the model fails to converge.

num_cats - Number of categories.

continuous - A Boolean value to indicate if the model should work in continuous input or binary input mode. Set continuous = TRUE when the inputs are continuous.

st must also contain the following information about the initial state of the network:

in_wts - A matrix of initial input-to-hidden weights with num_feats + 1 rows and num_hids columns. Can be set to NULL when the first line of the tr matrix includes control code 1, ctrl = 1.

out_wts - A matrix of initial hidden-to-output weights with num_feats + 1 rows, num_hids columns and with the third dimension being num_cats in extent. Can be set to NULL when the first line of the tr matrix includes control code 1, ctrl = 1.

st must also contain the following information so that it can reset these weights to random values when ctrl = 1 (see below):

wts_range - A scalar value for the range of the randomly-generated weights. The suggested weight range deafult is wts_range = 1

wts_center - A scalar value for the center of the randomly-generated weights. This is commonly set to wts_center = 0

st must also contain the following parameters that describe your tr array:

colskip - Skip the first N columns of the tr array, where N = colskip. colskip should be set to the number of optional columns you have added to matrix tr, PLUS ONE. So, if you have added no optional columns, colskip = 1. This is because the first (non-optional) column contains the control values, below.

Argument tr must be a matrix, where each row is one trial presented to the network. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

ctrl - column of control codes. Available codes are: 0 = normal learning trial, 1 = reset network (i.e. initialize a new set of weights following the st parameters), 2 = Freeze learning. Control codes are actioned before the trial is processed.

opt1, opt2, ... - optional columns, which may have any names you wish, and you may have as many as you like, but they must be placed after the ctrl column, and before the remaining columns

(see below). These optional columns are ignored by this function, but you may wish to use them for readability. For example, you might include columns for block number, trial number, and stimulus ID number. The argument `colskip` (see above) must be set to the number of optional columns plus 1.

`x1, x2, ...` - input to the model, there must be one column for each input unit. Each row is one trial. Dichotomous inputs should be in the format `-1, 1`. Continuous inputs should be scaled to the range of `-1, 1`. As the model's learning objective is to accurately reconstruct the inputs, the input to the model is also the teaching signal. For testing under conditions of missing information, input features can be set to 0 to negate the contribution of the feature(s) for the classification decision of that trial.

`t1, t2, ...` - Category membership of the current stimulus. There must be one column for each category. Each row is one trial. If the stimulus is a member of category X, then the value in the category X column must be set to `+1`, and the values for all other category columns must be set to `-1`.

Value

Returns a list containing two components: (1) matrix of response probabilities for each category on each trial, (2) an `st` list object that contains the model's final state. A weight initialization history is also available when the extended output parameter is set `xtdo = TRUE` in the `slpDIVA` call.

Note

A faster (Rcpp) implementation of slpDIVA is planned for a future release of catlearn.

Author(s)

Garrett Honke, Nolan B. Conaway, Andy Wills

References

Conaway, N. B., & Kurtz, K. J. (2014). Now you know it, now you don't: Asking the right question about category knowledge. In P. Bello, M. Guarini, M. McShane, & B. Scassellati (Eds.), *Proceedings of the Thirty-Sixth Annual Conference of the Cognitive Science Society* (pp. 2062-2067). Austin, TX: Cognitive Science Society.

Kruschke, J. (1992). ALCOVE: an exemplar-based connectionist model of category learning. *Psychological Review, 99*, 22-44

Kurtz, K.J. (2007). The divergent autoencoder (DIVA) model of category learning. *Psychonomic Bulletin & Review, 14*, 560-576.

Kurtz, K. J. (2015). Human Category Learning: Toward a Broader Explanatory Account. *Psychology of Learning and Motivation, 63*.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation, 66*, 79-115.

slpEXIT *EXIT Category Learning Model*

Description

EXemplar-based attention to distinctive InpuT model (Kruschke, 2001)

Usage

```
slpEXIT(st, tr, xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	R-by-C matrix of training items
xtdo	if TRUE extended output is returned

Details

The contents of this help file are relatively brief; a more extensive tutorial on using slpEXIT can be found in Spicer et al. (n.d.).

The functions works as a stateful list processor. Specifically, it takes a data frame as an argument, where each row is one trial for the network, and the columns specify the input representation, teaching signals, and other control signals. It returns a matrix where each row is a trial, and the columns are the response probabilities at the output units. It also returns the final state of the network (cue -> exemplar, and cue -> outcome weights), hence its description as a 'stateful' list processor.

References to Equations refer to the equation numbers used in the Appendix of Kruschke (2001).

Argument `tr` must be a data frame, where each row is one trial presented to the network, in the order of their occurence. `tr` requires the following columns:

`x1, x2, ...` - columns for each cue (`1` = cue present, `0` = cue absent). These columns have to start with `x1` ascending with features `..., x2, x3, ...` at adjacent columns. See Notes 1, 2.

`t1, t2, ...` - columns for the teaching values indicating the category feedback on the current trial. Each category needs a single teaching signal in a dummy coded fashion, e.g., if the first category is the correct category for that trial, then `t1` is set to `1`, else it is set to `0`. These columns have to start with `t1` ascending with categories `..., t2, t3, ...` at adjacent columns.

`ctrl` - vector of control codes. Available codes are: 0 = normal trial, 1 = reset network (i.e. reset connection weights to the values specified in `st`). 2 = freeze learning. Control codes are actioned before the trial processed.

`opt1, opt2, ...` - optional columns, which may have any name you wish. These optional columns are ignored by this function, but you may wish to use them for readability. For example, you might include columns for block number, trial number, and stimulus ID..

Argument `st` must be a list containing the following items:

nFeat - integer indicating the total number of possible stimulus features, i.e. the number of x1, x2, ... columns in tr.

nCat - integer indicating the total number of possible categories, i.e. the number of t1, t2, ... columns in tr.

phi - response scaling constant - Equation (2)

c - specificity parameter. Defines the narrowness of receptive field in exemplar node activation - Equation (3).

P - Attentional normalization power (attentional capacity) - Equation (5). If P equals 1 then the attention weights will satisfy the constraint that attention strength for currently present features will sum to one. The sum of attention strengths for present features grows as a function of P.

l_gain - attentional shift rate - Equation (7)

l_weight - learning rate for feature to category associations. - Equation (8)

l_ex - learning rate for exemplar_node to gain_node associations - Equation (9)

iterations - number of iterations of shifting attention on each trial (see Kruschke, 2001, p. 1400). If you're not sure what to use here, set it to 10.

sigma - Vector of cue saliences, one for each cue. If you're not sure what to put here, use 1 for all cues except the bias cue. For the bias cue, use some value between 0 and 1.

w_in_out - matrix with nFeat columns and nCat rows, defining the input-to-category association weights, i.e. how much each feature is associated to a category (see Equation 1). The nFeat columns follow the same order as x1, x2, ... in tr, and likewise, the nCat rows follow the order of t1, t2, ...

exemplars - matrix with nFeat columns and n rows, where n is the number of exemplars, such that each row represents a single exemplar in memory, and their corresponding feature values. The nFeat columns follow the same order as x1, x2, ... in tr. The n-rows follow the same order as in the w_exemplars matrix defined below. See Note 3.

w_exemplars - matrix which is structurally equivalent to exemplars. However, the matrix represents the associative weight from the exemplar nodes to the gain nodes, as given in Equation 4. The nFeat columns follow the same order as x1, x2, ... in tr. The n-rows follow the same order as in the exemplars matrix.

Value

Returns a list containing three components (if xtdo = FALSE) or four components (if xtdo = TRUE, g is also returned):

p	Matrix of response probabilities for each outcome on each trial
w_in_out	Matrix of final cue -> outcome associative strengths
w_exemplars	Matrix of final cue -> exemplar associative strengths
g	Vector of gains at the end of the final trial

Note

1. Code optimization in slpEXIT means it's essential that every cue is either set to 1 or to 0. If you use other values, it won't work properly. If you wish to represent cues of unequal salience, use sigma.

2. EXIT simulations normally include a 'bias' cue, i.e. a cue that is present on all trials. You will need to explicitly include this in your input representation in `tr`. For an example, see the output of `krus96train`.

3. The bias cue should be included in these exemplar representations, i.e. they should be the same as the representation of the stimuli in `tr`. For an example, see the output of `krus96train`.

Author(s)

René Schlegelmilch, Andy Wills, Angus Inkster

References

Kruschke, J. K. (1996). Base rates in category learning. *Journal of Experimental Psychology-Learning Memory and Cognition, 22*(1), 3-26.

Kruschke, J. K. (2001). The inverse base rate effect is not explained by eliminative inference. *Journal of Experimental Psychology: Learning, Memory & Cognition, 27*, 1385-1400.

Spicer, S.G., Schlegelmilch, R., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (n.d.). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

---

slpLMSnet *Gluck & Bower (1988) network model*

Description

Gluck and Bower (1988) adaptive least-mean-square (LMS) network

Usage

```
slpLMSnet(st, tr, xtdo = FALSE, dec = "logistic")
```

Arguments

`st`	List of model parameters
`tr`	Numerical matrix of training items, use `data.matrix()` if matrix is not numeric.
`xtdo`	Boolean specifying whether to include extended information in the output (see below)
`dec`	Specify what response rule to use. `logistic`, Equation 7 in Gluck and Bower (1988), which will output probability ratings for each outcome (these will not necessary sum to one). `softmax`, Footnote 2 in Gluck and Bower (1988), which will output response probabilities for each outcome (these will sum to one).

Details

The function operates as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix as an argument. Each row represents a single trial. Each column represents different types of information required by the implementation of the model, such as the elemental representation of stimuli, teaching signals, and other variables specifying the model's behaviour (e.g. freezing learning).

Argument `st` must be a list containing the following items:

`beta` - the learning rate (fixed for a given simulation) for the LMS learning rule. The upper bound of this parameter is not specified, but we suggest $0 < beta \leq 1$.

`theta` - is a positive scaling constant. When theta rises, the logistic choice function will become less linear. When theta is high, the logistic function will approximate the behaviour of a step function.

`bias` - is a bias parameter. It is the value of the output activation that results in an output probability rating of P = 0.5. For example, if you wish an output activation of 0.4 to produce a rated probability of 0.5, set beta to 0.4. If you are not sure what to use here, set it to 0. The bias parameter is not part of the original Gluck and Bower (1988) LMS network, see Note 1.

`w` - is a matrix of initial connection weights, where each row is an outcome, and each column is a feature or cue. If you are not sure what to use here, set all values to 0.

`outcomes` - is the number of possible categories or outcomes.

`colskip` - the number of optional columns to be skipped in the tr matrix. colskip should be set to the number of optional columns PLUS ONE. So, if you have added no extra columns, colskip = 1.

Argument `tr` must be a matrix, where each row is one trial presented to the model. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

`ctrl` - a vector of control codes. Available codes are: 0 = normal trial; 1 = reset model (i.e. set associative strengths (weights) back to their initial values as specified in w (see above)); 2 = Freeze learning. Control codes are actioned before the trial is processed.

`opt1, opt2, ...` - any number of preferred optional columns, the names of which can be chosen by the user. It is important that these columns are placed after the control column, and before the remaining columns (see below). These optional columns are ignored by the slpLMSnet function, but you may wish to use them for readability. For example, you might choose to include columns such as block number, trial number and condition. The argument colskip (see above) must be set to the number of optional columns plus one.

`x1, x2, ...` - activation of input nodes of corresponding features. Feature patterns usually represented as a bit array. Each element in the bit array encodes the activations of the input nodes given the presence or absence of the corresponding features. These activations can take on either 1 or 0, present and absent features respectively. For example, Medin and Edelson's (1988) inverse base-rate effect with stimuli AB and AC can be represented as [1 1 0] and [1 0 1] respectively. In a more unconventional scenario, you can set activation to vary between present 1 and absent -1, see Note 2. slpLMSnet can also support any positive or negative real number for activations, e.g. one might use values between 0 and 1 to represent the salience of the features.

`d1, d2, ...` - teaching input signals indicating the category feedback on the current trial. It is a bit array, similar to the activations of input nodes. If there are two categories and the stimuli on the current trial belongs to the first, then this would be represented in `tr` as [1 0], on edge cases see Note 3. The length of this array must be provided via `outcomes` in `st`.

Value

Returns a list with the following items if `xtdo = FALSE`:

`p` A matrix with either the probability rating for each outcome on each trial if `dec = "logistic"`, or response probabilities for each outcome on each trial if `dec = "softmax"`.

`nodeActivation` Output node activations on each trial, as output by Equation 3 in Gluck and Bower (1988).

`connectionWeightMatrix`
A connection weight matrix, W, where each row represents the corresponding element in the teaching signals array in `tr`, while each column represents the corresponding element from the input activation array from `tr`. So cell $w_1 2$ would be the connection weight between the second stimulus and the first category.

If `xtdo = TRUE`, the following item also returned:

`squaredDifferences`
The least mean squeared differences between desired and actual activations of output nodes on each trial (Eq. 4 in Gluck and Bower, 1988). This metric is an indicator of the network's performance, which is measured by its accuracy.

Note

1. The `bias` parameter is not part of the original Gluck and Bower (1988) model. `bias` in the current implementation helps comparisons between simulations using the `act2probrat` logistic choice function. Set bias to 0 for operation as specified in Gluck & Bower (1988). Also note that, where there is more than one output node, the same bias value is subtracted from the output of each node. This form of decision mechanism is not present in the literature as far as we are aware, although using a negative bias value would, in multi-outcome cases, approximate a 'background noise' decision rule, as used in, for example, Nosofsky et al. (1994).

2. slpLMSnet can support both positive and negative real numbers as input node activations. For example, one might wish to follow Markman's (1989) suggestion that the absence of a feature element is encoded as -1 instead of 0.

3. slpLMSnet can process a bit array of teaching signals, where the model is told that the stimulus belongs to more than one category. slpLMSnet uses matrix operations to update weights, so it can encode and update multiple teaching signals on the same trial.

Author(s)

Lenard Dome, Andy Wills

References

Gluck, M. A., & Bower, G. H. (1988). From conditioning to category learning: An adaptive network model. *Journal of Experimental Psychology: General, 117*, 227-247.

Markman, A. B. (1989). LMS rules and the inverse base-rate effect: Comment on Gluck and Bower (1988). *Journal of Experimental Psychology: General, 118*, 417-421.

Medin, D. L., & Edelson, S. M. (1988). Problem structure and the use of base-rate information from experience. *Journal of Experimental Psychology: General, 117*, 68-85.

Nosofsky, R.M., Gluck, M.A., Plameri, T.J., McKinley, S.C. and Glauthier, P. (1994). Comparing models of rule-based classification learning: A replication and extension of Shepard, Hovland, and Jenkins (1961). *Memory and Cognition, 22*, 352-369.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation, 66*, 79-115.

Examples

```
## load catlearn
library(catlearn)

## create st with initial state
st <- list(beta = 0.025, # learning rate
           theta = 1, # decision scaling parameter
           bias = 0, # decision bias parameter
           # initial weight matrix,
           # row = number of categories,
           # col = number of cues
           w = matrix(rep(0, 6*4), nrow = 4, ncol = 6),
           outcomes = 4, # number of possible outcomes
           colskip = 3)

## create inverse base-rate effect tr for 1 subject and without bias cue
tr <- krus96train(subjs = 1, ctxt = FALSE)

# run simulation and store output
out <- slpLMSnet(st, data.matrix(tr))

out$connectionWeightMatrix
```

`slpMack75` *Mackintosh (1975) associative learning model*

Description

Mackintosh's (1975) attentional learning model, as implemented by Le Pelley et al. (2016).

Usage

```
slpMack75(st, tr, xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	Matrix of training items
xtdo	Boolean specifying whether to include extended information in the output (see below)

Details

The function operates as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (tr) as an argument, where each row represents a single training trial, while each column represents the different types of information required by the model, such as the elemental representation of the training stimuli, and the presence or absence of an outcome. It returns the output activation on each trial (a.k.a. sum of associative strengths of cues present on that trial), as a vector. The slpMack75 function also returns the final state of the model - a vector of associative and attentional strengths between each stimulus and the outcome representation.

Argument `st` must be a list containing the following items:

`lr` - the associative learning rate (fixed for a given simulation), as denoted by `theta` in Equation 1 of Mackintosh (1975).

`alr` - the attentional learning rate parameter. It can be set without limit (see alpha below), but we recommend setting this parameter to somewhere between 0.1 and 1.

`w` - a vector of initial associative strengths. If you are not sure what to use here, set all values to zero.

`alpha` - a vector of initial attentional strengths. If the updated value is above 1 or below 0.1, it is capped to 1 and 0.1 respectively.

`colskip` - the number of optional columns to be skipped in the tr matrix. colskip should be set to the number of optional columns you have added to the tr matrix, PLUS ONE. So, if you have added no optional columns, colskip=1. This is because the first (non-optional) column contains the control values (details below).

Argument `tr` must be a matrix, where each row is one trial presented to the model. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

`ctrl` - a vector of control codes. Available codes are:

0 = normal trial 1 = reset model (i.e. set associative strengths back to their initial values as specified in w) 2 = Freeze learning 3 = Reset associative weights to initial state, but keep attentional strengths in alpha 4 = Reset attentional strengths to initial state, but keep association weights.

Control codes are actioned before the trial is processed.

`opt1, opt2, ...` - any number of preferred optional columns, the names of which can be chosen by the user. It is important that these columns are placed after the control column, and before the remaining columns (see below). These optional columns are ignored by the function, but you may wish to use them for readability. For example, you might choose to include columns such as block number, trial number and condition. The argument colskip (see above) must be set to the number of optional columns plus one.

`x1, x2, ...` - activation of any number of input elements. There must be one column for each input element. Each row is one trial. In simple applications, one element is used for each stimulus (e.g. a

simulation of blocking (Kamin, 1969), A+, AX+, would have two inputs, one for A and one for X). In simple applications, all present elements have an activation of 1 and all absence elements have an activation of 0. However, slpMack75 supports any real number for activations, e.g. one might use values between 0 and 1 to represent differing cue saliences.

`t` - Teaching signal (a.k.a. lambda). Traditionally, 1 is used to represent the presence of the outcome, and 0 is used to represent the absence of the outcome, although slpMack75 supports any real values for lambda. If you are planning to use multiple outcomes, see Note 2.

Argument `xtdo` (eXTenDed Output) - if set to TRUE, function will additionally return trial-level data including attentional strengths and the updated associative strengths after each trial (see Value).

Value

Returns a list containing three components (if xtdo = FALSE) or five components (if xtdo = TRUE, xoutw and xouta is also returned):

`suma`	Vector of summed associative strength for each trial.
`w`	Vector of final associative strengths.
`alpha`	Vector of final attentional weights.
`xoutw`	Matrix of trial-level data of the associative strengths at the end of the trial, after each has been updated.
`xouta`	Matrix of trial-level data of the attentional strengths at the end of the trial, after each has been updated.

Note

1. Mackintosh (1975) did not formalise how to update the cues' associability, but described when associability increases or decreases in Equation 4 and 5. He assumed that the change in alpha would reflect the difference between the prediction error generated by the current cue and the combined influence (a sum) of all other cues. Le Pelley et al. (2016) provided a linear function in Equation 2 that adheres to this description. This expression is probably the simplest way to express Mackintosh's somewhat vague description in mathematical terms. A linear function is also easier to computationally implement. So we decided to use Equation 2 from Le Pelley et al. (2016) for updating attentional strengths.

2. At present, only single-outcome experiments are officially supported. If you want to simulate a two-outcome study, consider using +1 for one outcome, and -1 for the other outcome. Alternatively, run a separate simulation for each outcome.

Author(s)

Lenard Dome, Andy Wills, Tom Beesley

References

Kamin, L.J. (1969). Predictability, surprise, attention and conditioning. In Campbell, B.A. & Church, R.M. (eds.), *Punishment and Aversive Behaviour*. New York: Appleton-Century-Crofts, 1969, pp.279-296.

Le Pelley, M. E., Mitchell, C. J., Beesley, T., George, D. N., & Wills, A. J. (2016). Attention and associative learning in humans: An integrative review. *Psychological Bulletin*, 142(10), 1111–1140. https://doi.org/10.1037/bul0000064

Mackintosh, N.J. (1975). A theory of attention: Variations in the associability of stimuli with reinforcement, *Psychological Review, 82*, 276-298.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation, 66*, 79-115.

`slpMBMF` *MB/MF reinforcement learning model*

Description

Gillan et al.'s (2015) model-free / model-based hybrid Reinforcement Learning model (see Note 1).

Usage

```
slpMBMF(st, tr, xtdo = FALSE)
```

Arguments

`st`	List of model parameters
`tr`	Matrix of training items
`xtdo`	Boolean. When TRUE, extended output is provided, see below

Details

The contents of this help file are relatively brief; a more extensive discussion of this model can be found in the supplementary materials of Gillan et al. (2015).

The function operates as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (tr) as an argument, where each row represents a single training trial, while each column represents the different types of information required by the model. It returns a matrix of predicted response probabilities for each stage 1 action on each trial. The slpMBMF function also returns the final Q values for the model.

The current implementation of slpMBMF deals only with relatively simple Reinforcement Learning experiments, of which Gillan et al. (2015, Exp. 2) is one example. Specifically, each trial has two stages. In the first stage of the trial, there is a single state, and the participant can emit one of `x` actions. In the second stage, there are `y` states. A reward follows (or doesn't) without a further action from the participant.

A hybrid MB/MF model thus has `2x` Q-values at stage 1 (`x` for the model-based system, and `x` for the model-free system), and `y` Q-values at stage 2 (one for each state; there are no actions at stage 2, and the MB and MF systems evaluate stage 2 Q-values the same way in this model). See Note 3.

Argument `st` must be a list containing the following items:

alpha - the model-free learning rate (range: 0-1)

lambda - the eligibility trace parameter (range: 0-1)

w - A number between 0 and 1, representing the relative contribution of the model-based and model-free parts of the model to the response (0 = pure model-free, 1 = pure model-based).

beta - Decision stochasticity parameter

p - Decision perseveration (p > 0) or switching (p < 0) parameter

tprob - A 2 x 2 matrix of transition probabilities, used by the model-based system. The rows are the actions at stage 1. The columns are the states at stage 2. The cells are transition probabilities (e.g. tprob[2,1] is the probability of arriving at stage 2 state #1 given action #2 at stage 1).

q1.mf - A vector of initial model-free Q values for the actions at stage 1.

q1.mb - A vector of initial model-based Q values for the actions at stage 1.

q2 - A vector of initial Q values for the states at stage 2 (the MB and MF systems share common Q values at stage 2).

If you are unsure what initial Q values to use, set all to 0.5.

Argument tr must be a matrix, where each row is one trial. Trials are always presented to the model in the order specified. The matrix must contain the following named columns (other columns will be ignored):

s1.act - The action made by the participant at stage 1, for each trial; must be an integer in the range 1-x.

s2.state - State of environment at stage 2, for each trial; must be an integer in the range 1-y.

t - Reward signal for trial; must be a real number. If you're unsure what to use here, use 1 = rewarded, 0 = not rewarded.

Value

When xtdo = FALSE, returns a list containing these components:

out - Matrix of response probabilities, for each stage 1 action on each trial.

q1.mf - A vector of final model-free Q values for the actions at stage 1.

q1.mb - A vector of final model-based Q values for the actions at stage 1

q2 - A vector of final Q values for the states at stage 2 (the MB and MF systems share common Q values at stage 2).

When xtdo = TRUE, the list also contains the following model-state information :

xout - A matrix containing the state of the model at the end of each trial. Each row is one trial. It has the following columns:

q1.mb.1, q1.mb.2, ... - One column for each model-based Q value at stage 1.

q1.mf.1, q1.mf.2, ... - One column for each model-free Q value at stage 1.

q2.1, q2.2, ... - One column for each Q value at stage 2.

q1.h.1, q1.h.2, ... - One column for each hybrid Q value at stage 1.

s1.d.mf - Model-free delta at stage 2, wrt. stage 1 action.

s2.d.mf - Model-free delta at outcome.

In addition, when xtdo = TRUE, the list also contains the following information that is not used by the model (but which might be handy as potential neural regressors).

`s1.d.mb` - Model-based delta at stage 2, wrt. stage 1 action.

`s1.d.h` - Hybrid delta (based on stage 1 hybrid Q values) at stage 2, wrt. stage 1 action.

`s1.d.diff` - s1.d.mf - s1.d.mb

Note

1. Gillan et al.'s (2015) choice rule, at least as stated in their supplementary materials, would lead to the response probabilities being infinite on switch trials, which is presumably an error. The current implementation uses Daw et al. (2011, suppl. mat., Eq. 2).

2. Gillan et al. (2015) decay Q values for unselected actions by (1-alpha). This is not part of the current implementation.

3. In the current implementation of the model, `x` must be 2 and `y` must be two, otherwise the model will fail or behave unpredictably. If you'd like to develop a more general version of this implementation, contact the author.

Author(s)

Andy Wills (andy@willslab.co.uk), Tom Sambrook

References

Daw, N.D., Gershman, S.J., Seymour, B., Dayan, P., & Dolan, R.J. (2011). Model-based influences on humans' choices and striatal prediction errors. *Neuron, 69*, 1204-1215.

Gillan, C.M., Otto, A.R., Phelps, E.A. & Daw, N.D. (2015). Model-based learning protects against forming habits. *Cogn. Affect. Behav. Neurosci., 15*, 523-536.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling thrX-Sough distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation, 66*, 79-115.

---

slpNNCAG *A Neural Network with Competitive Attentional Gating (NNCAG)*

Description

This is Model 4 from Paskewitz and Jones (2020). Model 4 is a Neural Network with Competitive Attentional Gating - a fragmented version of EXIT (Kruschke, 2001) lacking exemplar-based rapid attentional shifts.

Usage

```
slpNNCAG(st, tr, xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	R matrix of training items
xtdo	Boolean specifying whether to include extended information in the output (see below).

Details

The function operates as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (tr) as an argument, where each row represents a single training trial, while each column represents the different types of information required by the model, such as the elemental representation of the training stimuli, and the presence or absence of an outcome.

Argument st must be a list containing the following items:

P - attention normalization constant, P.

phi - decision-making constant, ϕ, also referred to as specificity constant.

lambda - learning rate, λ.

mu - attentional learning rate, μ.

outcomes - The number of categories.

w - a $k \times i$ matrix of initial weights, where k equals to the number of categories and i equals to the number of stimuli.

eta - η, a vector with i elements, where η^{th} is the salience of the i^{th} cue. In edge cases, η is capped at lower bound of 0.1, see Note 1.

colskip - The number of optional columns to be skipped in the tr matrix. colskip should be set to the number of optional columns you have added to the tr matrix, PLUS ONE. So, if you have added no optional columns, colskip=1. This is because the first (non-optional) column contains the control values (details below).

Argument trmust be a matrix, where each row is one trial presented to the model. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

ctrl - a vector of control codes. Available codes are: 0 = normal trial; 1 = reset model (i.e. set matrix of initial weights and vector of salience back to their initial values as specified in st); 2 = Freeze learning. Control codes are actioned before the trial is processed.

opt1, opt2, ... - any number of preferred optional columns, the names of which can be chosen by the user. It is important that these columns are placed after the control column, and before the remaining columns (see below). These optional columns are ignored by the function, but you may wish to use them for readability. For example, you might choose to include columns such as block number, trial number and condition. The argument colskip (see above) must be set to the number of optional columns plus one.

x1, x2, ... - columns for each cue (1 = cue present, 0 = cue absent). There must be one column for each input element. Each row is one trial. In simple applications, one element is used for each stimulus (e.g. a simulation of blocking (Kamin, 1969), A+, AX+, would have two inputs, one for A and one for X). In simple applications, all present elements have an activation of 1 and all absent elements have an activation of 0. However, slpNNCAG supports any real number for activations.

`t1, t2, ...` - columns for the teaching values indicating the category feedback on the current trial. Each category needs a single teaching signal in a dummy coded fashion, e.g., if there are four categories and the current stimulus belongs to the second category, then we would have `[0, 1, 0, 0]`.

Value

Returns a list containing three components (if xtdo = FALSE) or four components (if xtdo = TRUE).

if `xtdo = FALSE`:

`p`	Response probabilities for each trial (rows) and each category (columns).
`final_eta`	Salience at the end of training. η for each stimulus i.
`final_weights`	An $k \times i$ weight matrix at the end of training, where rows are categories and columns are stimuli. Order of stimuli and categories correspond to their order in `tr`.

if `xtdo = TRUE`, the following values are also returned:

`model_predictions`	The matrix for trial-level predictions of the model as specified by Equation 5 in Paskewitz and Jones (2021).
`eta`	The updated salience at the end of each trial.

Note

1. If there is only one stimulus present on a given trial with η = `0` or with g = `0`, Equation 12 of Paskewitz & Jones (2020) breaks down. In order to avoid this, eta and g are capped at the lower limit of `0.01`.

2. This model is implemented in C++ for speed.

Author(s)

Lenard Dome, Andy Wills

References

Kamin, L.J. (1969). Predictability, surprise, attention and conditioning. In Campbell, B.A. & Church, R.M. (eds.), *Punishment and Aversive Behaviour*. New York: Appleton-Century-Crofts, 1969, pp.279-296.

Kruschke, J. K. (2001). Toward a unified model of attention in associative learning. *Journal of Mathematical Psychology, 45(6)*, 812-863.

Paskewitz, S., & Jones, M. (2020). Dissecting EXIT. *Journal of Mathematical Psychology, 97*, 102371.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation, 66*, 79-115.

See Also

slpEXIT

---

slpNNRAS *A Neural Network with Rapid Attentional Shifts (NNRAS)*

Description

This is Model 5 from Paskewitz and Jones (2020). Model 5 is a Neural Network with Rapid Attentional Shifts the also contains an competitive attentional gating mechanism. It is a fragmented version of EXIT (Kruschke, 2001) lacking exemplar-mediated attention.

Usage

```
slpNNRAS(st, tr, xtdo = FALSE)
```

Arguments

st	List of model parameters
tr	R matrix of training items
xtdo	Boolean specifying whether to include extended information in the output (see below).

Details

The function operates as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (tr) as an argument, where each row represents a single training trial, while each column represents the different types of information required by the model, such as the elemental representation of the training stimuli, and the presence or absence of an outcome.

Argument st must be a list containing the following items:

P - attention normalization constant, P.

phi - decision-making constant, ϕ, also referred to as specificity constant.

lambda - learning rate, λ.

mu - attentional learning rate, μ.

rho - attentional shift rate, ρ. Attention shifts ten times per trial.

outcomes - The number of categories.

w - a $k \times i$ matrix of initial weights, where k equals to the number of categories and i equals to the number of stimuli.

eta - η, a vector with i elements, where η^{th} is the salience of the i^{th} cue. In edge cases, η is capped at lower bound of 0.1, see Note 1.

colskip - The number of optional columns to be skipped in the tr matrix. colskip should be set to the number of optional columns you have added to the tr matrix, PLUS ONE. So, if you have

added no optional columns, colskip=1. This is because the first (non-optional) column contains the control values (details below).

Argument `tr`must be a matrix, where each row is one trial presented to the model. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

`ctrl` - a vector of control codes. Available codes are: 0 = normal trial; 1 = reset model (i.e. set matrix of initial weights and vector of salience back to their initial values as specified in `st`); 2 = Freeze learning. Control codes are actioned before the trial is processed.

`opt1, opt2, ...` - any number of preferred optional columns, the names of which can be chosen by the user. It is important that these columns are placed after the control column, and before the remaining columns (see below). These optional columns are ignored by the function, but you may wish to use them for readability. For example, you might choose to include columns such as block number, trial number and condition. The argument colskip (see above) must be set to the number of optional columns plus one.

`x1, x2, ...` - columns for each cue (`1` = cue present, `0` = cue absent). There must be one column for each input element. Each row is one trial. In simple applications, one element is used for each stimulus (e.g. a simulation of blocking (Kamin, 1969), A+, AX+, would have two inputs, one for A and one for X). In simple applications, all present elements have an activation of 1 and all absence elements have an activation of 0. However, slpNNRAS supports any real number for activations.

`t1, t2, ...` - columns for the teaching values indicating the category feedback on the current trial. Each category needs a single teaching signal in a dummy coded fashion, e.g., if there are four categories and the current stimulus belongs to the second category, then we would have `[0, 1, 0, 0]`.

Value

Returns a list containing three components (if xtdo = FALSE) or four components (if xtdo = TRUE).

if `xtdo` = `FALSE`:

`p`	Response probabilities for each trial (rows) and each category (columns).
`final_eta`	Salience at the end of training. η for each stimulus i.
`final_weights`	An $k \times i$ weight matrix at the end of training, where rows are categories and columns are stimuli. Order of stimuli and categories correspond to their order in `tr`.

if `xtdo` = `TRUE`, the following values are also returned:

`model_predictions`	The matrix for trial-leve predictions of the model as specified by Equation 5 in Paskewitz and Jones (2020).
`eta`	The updated salience at the end of each trial.

Note

1. If there is only one stimulus present on a given trial with η = `0` or with g = `0`, Equation 12 breaks down. In order to avoid this, eta and g is capped at the lower limit of `0.01`.

2. This model is implemented in C++ for speed.

Author(s)

Lenard Dome, Andy Wills

References

Kamin, L.J. (1969). Predictability, surprise, attention and conditioning. In Campbell, B.A. & Church, R.M. (eds.), *Punishment and Aversive Behaviour*. New York: Appleton-Century-Crofts, 1969, pp.279-296.

Kruschke, J. K. (2001). Toward a unified model of attention in associative learning. *Journal of Mathematical Psychology, 45(6)*, 812-863.

Paskewitz, S., & Jones, M. (2020). Dissecting EXIT. *Journal of Mathematical Psychology, 97*, 102371.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation, 66*, 79-115.

See Also

slpEXIT

`slpRW` *Rescorla-Wagner (1972) associative learning model.*

Description

Rescorla & Wagner's (1972) theory of Pavlovian conditioning.

Usage

```
slpRW(st, tr, xtdo = FALSE)
```

Arguments

`st`	List of model parameters
`tr`	Matrix of training items
`xtdo`	Boolean specifying whether to include extended information in the output (see below)

Details

The contents of this help file are relatively brief; a more extensive tutorial on using slpRW can be found in Spicer et al. (n.d.).

The function operates as a stateful list processor (slp; see Wills et al., 2017). Specifically, it takes a matrix (tr) as an argument, where each row represents a single training trial, while each column represents the different types of information required by the model, such as the elemental representation of the training stimuli, and the presence/absence of an outcome. It returns the output activation on each trial (a.k.a. sum of associative strengths of cues present on that trial), as a vector. The slpRW function also returns the final state of the model - a vector of associative strengths between each stimulus and the outcome representation.

Argument `st` must be a list containing the following items:

`lr` - the learning rate (fixed for a given simulation). In order to calculate lr, calculate the product of Rescorla-Wagner parameters alpha and beta. For example, if you want alpha = 0.1 and beta = 0.2, set lr = 0.02. If you want different elements to differ in salience (different alpha values) use the input activations (x1, x2, ..., see below) to represent element-specific salience. For example, if alpha_A = 0.4, alpha_X = 0.2, and beta = 0.1, then set lr = 0.1, and the activations of A and B to 0.4 and 0.2, respectively.

`w` - a vector of initial associative strengths. If you are not sure what to use here, set all values to zero.

`colskip` - the number of optional columns to be skipped in the tr matrix. colskip should be set to the number of optional columns you have added to the tr matrix, PLUS ONE. So, if you have added no optional columns, colskip=1. This is because the first (non-optional) column contains the control values (details below).

Argument `tr` must be a matrix, where each row is one trial presented to the model. Trials are always presented in the order specified. The columns must be as described below, in the order described below:

`ctrl` - a vector of control codes. Available codes are: 0 = normal trial; 1 = reset model (i.e. set associative strengths (weights) back to their initial values as specified in w (see above)); 2 = Freeze learning. Control codes are actioned before the trial is processed.

`opt1, opt2, ...` - any number of preferred optional columns, the names of which can be chosen by the user. It is important that these columns are placed after the control column, and before the remaining columns (see below). These optional columns are ignored by the slpRW function, but you may wish to use them for readability. For example, you might choose to include columns such as block number, trial number and condition. The argument colskip (see above) must be set to the number of optional columns plus one.

`x1, x2, ...` - activation of any number of input elements. There must be one column for each input element. Each row is one trial. In simple applications, one element is used for each stimulus (e.g. a simulation of blocking (Kamin, 1969), A+, AX+, would have two inputs, one for A and one for X). In simple applications, all present elements have an activation of 1 and all absence elements have an activation of 0. However, slpRW supports any real number for activations, e.g. one might use values between 0 and 1 to represent differing cue saliences.

`t` - Teaching signal (a.k.a. lambda). Traditionally, 1 is used to represent the presence of the outcome, and 0 is used to represent the absence of the outcome, although slpRW suports any real values for lambda.

Argument `xtdo` (eXTenDed Output) - if set to TRUE, function will return associative strength for the end of each trial (see Value).

Value

Returns a list containing two components (if xtdo = FALSE) or three components (if xtdo = TRUE, xout is also returned):

`suma`	Vector of output activations for each trial
`st`	Vector of final associative strengths
`xout`	Matrix of associative strengths at the end of each trial

Author(s)

Stuart Spicer, Lenard Dome, Andy Wills

References

Kamin, L.J., (1969) Predictability, surprise, attention and conditioning. In Campbell, B.A. & Church, R.M. (eds.), *Punishment and Aversive Behaviour*. New York: Appleton-Century-Crofts, 1969, pp.279-296.

Rescorla, R. A., & Wagner, A. R. (1972). A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black & W. F. Prokasy (Eds.), *Classical conditioning II: Current research and theory* (pp. 64-99). New York: Appleton-Century-Crofts.

Spicer, S.G., Jones, P.M., Inkster, A.B., Edmunds, C.E.R. & Wills, A.J. (n.d.). Progress in learning theory through distributed collaboration: Concepts, tools, and examples. *Manuscript in preparation*.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation, 66*, 79-115.

`slpSUSTAIN` *SUSTAIN Category Learning Model*

Description

Supervised and Unsupervised STratified Adaptive Incremental Network (Love, Medin & Gureckis, 2004)

Usage

```
slpSUSTAIN(st, tr, xtdo = FALSE, ties = "random")
```

Arguments

st	List of model parameters
tr	Matrix of training items
xtdo	Boolean specifying whether to include extended information in the output (see below)
ties	Model behaviour where multiple clusters have the same highest activations. Options are: random, first, see below.

Details

This function works as a stateful list processor (slp; see Wills et al., 2017). It takes a matrix (tr) as an argument, where each row represents a single training trial, while each column represents some information required by the model, such as the stimulus representation, indications of supervised/unsupervised learning, etc. (details below).

Argument st must be a list containing the following items:

r - Attentional focus parameter, always non-negative.

beta - Cluster competition parameter, always non-negative.

d - Decision consistency parameter, always non-negative.

eta - Learning rate parameter, see Note 1.

tau - Threshold parameter for cluster recruitment under unsupervised learning conditions (Love et al., 2004, Eq. 11). If every trial is a supervised learning trial, set tau to 0. slpSUSTAIN can accomodate both supervised and unsupervised learning within the same simulation, using the ctrl column in tr (see below).

lambda - Vector containing the initial receptive field tuning value for each stimulus dimension; the order corresponds to the order of dimensions in tr, below. For a stimulus with three dimensions, where all receptive fields are equally tuned, lambda = [1, 1, 1].

cluster - A matrix of the initial positions of each recruited cluster. If set to NA, cluster = NA, then each time the network is reset, a single cluster will be created, centered on the stimulus presented on the current trial.

w - A matrix of initial connection weights. If set to NA as w = NA then, each time the network is reset, zero-strength weights to a single cluster will be created.

dims - Vector containing the length of each dimension (excluding category dimension, see tr, below), i.e. the number of nominal spaces in the representation of each dimension. For Figure 1 of Love et al. (2004), dims = [2, 2, 2].

maxcat - optional. If set, maxcat is an integer specifying the maximum number of clusters to be recruited during unsupervised learning. A similar restriction has been used by Love et al. (2004) to simulate an unsupervised free-sorting task from Experiment 1 in Medin, Wattenmaker, & Hampson (1987). In this experiment, participants needed to sort items into two predefined categories. This parameter will only be used during unsupervised learning. If it is not set, or if it is set to 0, there is no maximum to the number of clusters that can be created.

colskip - Number of optional columns skipped in tr, PLUS ONE. So, if there are no optional columns, set colskip to 1.

Argument tr must be a matrix, where each row is one trial presented to the model. Columns are always presented in the order specified below:

ctrl - A vector of control codes. The control codes are processed prior to the trial and prior to updating cluster's position, lambdas and weights (Love et al., 2004, Eq. 12, 13 and 14, respectively). The available values are:

0 = do supervised learning.

1 = reset network and then do supervised learning.

2 = freeze supervised learning.

3 = do unsupervised learning.

4 = reset network and then do unsupervised learning.

5 = freeze unsupervised learning

'Reset network' means revert w, cluster,and lambda back to the values passed in st.

Unsupervised learning in slpSUSTAIN is at an early stage of testing, as we have not yet established any CIRP for unsupervised learning.

opt1, opt2, ... - optional columns, which may have any names you wish, and you may have as many as you like, but they must be placed after the ctrl column, and before the remaining columns (see below). These optional columns are ignored by this function, but you may wish to use them for readability. For example, you might include columns for block number, trial number, and stimulus ID number.

x1, x2, y1, y2, y3, ... - Stimulus representation. The columns represent the kth nominal value for ith dimension. It's a 'padded' way to represent stimulus dimensions and category membership (as category membership in supervised learning is treated as an additional dimension) with varying nominal length, see McDonnell & Gureckis (2011), Fig. 10.2A. All dimensions for the trial are represented in this single row. For example, if for the presented stimulus, dimension 1 is [0 1] and dimension 2 is [0 1 0] with category membership [0 1], then the input representation is [0 1 0 1 0 0 1].

Argument ties can be either random or first. It specifies how the model behaves in the event, when there are multiple winning clusters with the same activations (see Note):

random - The model randomly selects one cluster from the ones that have the same activations. To increase the reproducibility of your simulation, set a specific random seed seed before calling slpSUSTAIN (use e.g.set.seed).

first - The model selects the cluster that was first recruited from the clusters that have the same activations. Up to and including version 0.7.1 of catlearn, this was the default behaviour of slpSUSTAIN.

Value

Returns a list with the following items if xtdo = FALSE:

probs	Matrix of probabilities of making each response within the queried dimension (e.g. column 1 = category A; column 2 = category B), see Love et al. (2004, Eq. 8). Each row is a single trial and columns are in the order presented in tr, see below. In the case of unsupervised learning, probabilities are calculated for all dimensions (as there is no queried dimension for unsupervised learning).

`lambda` Vector of the receptive field tunings for each stimulus dimension, after the final trial. The order of dimensions corresponds to the order they are presented in `tr`, see below.

`w` Matrix of connection weights, after the final trial. Each row is a separate cluster, reported in order of recruitment (first row is the first cluster to be recruited). The columns correspond to the columns on the input representation presented (see `tr` description, below).

`cluster` Matrix of recruited clusters, with their positions in stimulus space. Each row is a separate cluster, reported in order of recruitment. The columns correspond to the columns on the input representation presented (see `tr` description, below).

If `xtdo = TRUE`, `xtdo` is returned instead of `probs`:

`xtdo` A matrix that includes `probs`, and in addition includes the following columns: `winning` - number of the winning cluster; `activation` - its output activation after cluster competition, Eq. 6 in Love et al. (2004); `recognition_scores` - for the current stimulus from Eq. A6 in Love and Gureckis (2007), this measure represents the model's overall familiarity with the stimulus; `endorsement` - the probability of judging an item as old per Eq. 11 in Love and Gureckis (2007).

Note

1. Love et al. (2004) do not explicitly set a range for the learning rate; we recommend a range of 0-1.

2. The specification of SUSTAIN states that under supervised learning, a new cluster is recruited each time the model predicts category membership incorrectly. This new cluster is centered on the current stimulus. The implementation in slpSUSTAIN adds the stipulation that a new cluster is NOT recruited if it already exists, i.e. if its location in stimulus space is identical to the location on an existing cluster. Instead, it selects the existing cluster and updates as normal. Love et al. (2004) do not specify model behaviour under such conditions, so this is an assumption of our implementation. We'd argue that this is a reasonable implementation - without it SUSTAIN would add clusters indefinitely under conditions where the stimulus -> category associations are proabilistic rather than deterministic.

3. In some cases, two or more clusters can have identical activations because the presented stimulus is equally similar to multiple clusters. Love et al. (2004) does not specify how the model will behave in these cases. In our implementation, we make the assumption that the model picks randomly between the highest activated clusters (given that they have the same activations). This, we felt, was in line with the approximation of lateral inhibition in the SUTAIN specification (Love et al. 2004, Eq. 6).

Author(s)

Lenard Dome, Andy Wills

References

Love, B. C., & Gureckis, T.M. (2007). Models in Search of a Brain. *Cognitive, Affective, & Behavioral Neuroscience, 7*, 90-108.

Love, B. C., Medin, D. L., & Gureckis, T. M. (2004). SUSTAIN: a network model of category learning. *Psychological Review, 111*, 309-332.

McDonnell, J. V., & Gureckis, T. M. (2011). Adaptive clustering models of categorization. In E. M. Pothos & A. J. Wills (Eds.), *Formal Approaches in Categorization*, pp. 220-252.

Medin, D. L., Wattenmaker, W. D., & Hampson, S. E. (1987). Family resemblance, conceptual cohesiveness, and category construction. *Cognitive Psychology, 19(2)*, 242-279.

Wills, A.J., O'Connell, G., Edmunds, C.E.R., & Inkster, A.B.(2017). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *Psychology of Learning and Motivation, 66*, 79-115.

`ssecl` *Sum of squared errors*

Description

Calculate sum of squared errors

Usage

```
ssecl(obs,exp)
```

Arguments

`obs`	Vector of observed values
`exp`	Vector of expected values

Value

Returns sum of the squared differences.

Author(s)

Andy Wills

`stsimGCM` *Generalized Context Model*

Description

Nosofsky's (1984, 2011) Generalized Context Model; an exemplar-based model of categorization.

Usage

```
stsimGCM(st)
```

Arguments

`st`	List of model parameters

Details

Argument `st` must be a list containing the following required items: `training_items`, `tr`, `nCats`, `nFeat`, `sensitivity`, `weights`, `choice_bias`, `p`, `r_metric`, `mp`, and `gamma`

`nCats` - integer indicating the number of categories

`nFeat` - integer indicating the number of stimulus dimensions

`tr` - the stimuli presented to the model, for which the choice probabilities will be predicted. `tr` has to be a matrix or dataframe with one row for each stimulus. `tr` requires the following columns.

`x1, x2, ...` - columns for each dimension carrying the corresponding values (have to be coded as numeric values) for each exemplar (trial) given in the row. Columns have to start with `x1` ascending with dimensions `..., x2, x3, ...` at adjacent columns.

`tr` may have any number of additional columns with any desired name and position, e.g. for readability. As long as the feature columns `x1, x2, ...` are given as defined (i.e. not scattered, across the range of matrix columns), the output is not affected by optional columns.

`training_items` - all unique exemplars assumed to be stored in memory; has to be a matrix or dataframe with one row for each exemplar. The rownames have to start with 1 in ascending order. `training_items` requires the following columns:

`x1, x2, ...` - columns for each feature dimension carrying the corresponding values (have to be coded as numeric values) for each exemplar (row). Columns have to start with `x1` ascending with dimensions `..., x2, x3, ...` at adjacent columns.

`cat1, cat2, ...` - columns that indicate the category assignment of each exemplar (row). For example, if the exemplar in row 2 belongs to category 1 the corresponding cell of `cat1` has to be set to `1`, else `0`. Columns have to start with `cat1` ascending with categories `..., cat2, cat3, ...` at adjacent columns.

`mem` - (optional) one column that indicates whether an exemplar receives an extra memory weight, yes = `1`, no = `0`. For each exemplar (row) in the `training_items` with `mem` set to `0` the corresponding memory strength parameter is set to 1. When `mem` for an exemplar is set to `1` the memory strength parameter is set as defined in `mp`, see below.

training_items may have any number of additional columns with any desired name and position, e.g. for readability. As long as the feature columns x1, x2, ... and cat1, cat2, ... are given as defined (i.e. not scattered, across the range of matrix columns), the output is not affected by optional columns.

NOTE: The current model can be implemented as a prototype model if the training_items only carry one row for each category representing the values of the corresponding prototypes (e.g. see Minda & Smith, (2011).

mp - memory strength parameter (optional). Can take any numeric value between -Inf and +Inf. The default is 1, i.e. all exemplars have the same memory strength. There are two ways of specifying mp, i.e. either *globally* or *exemplar specific*:

When *globally* setting mp to a single integer, e.g. to 5, then all exemplars in training_items with mem = 1 will receive a memory strength 5 times higher than the memory strengths for the remaining exemplars.

For setting *exemplar specific* memory strengths mp has to be a vector of length n, where n is the overall number of of exemplars with mem = 1 in the training_items. The order of memory strengths defined in this vector exactly follows their row-wise ascending order of appearence in the training_items. E.g. if there are two exemplars with mem = 1 in the training_items, the first one in row 2 and the second one in row 10, then setting mp to c(3,2) will result in assigning a memory strength of 3 to the first exemplar (in row 2) and a memory strength of 2 to the second exemplar (in row 10). The memory strengths for all other exemplars will be set to 1. See Note 1.

sensitivity - sensitivity parameter c; can take any value between 0 (all exemplars are equally similar) and +infinity (towards being insensitive to large differences). There are two ways of specifying sensitivity, i.e. either *globally* or *exemplar specific*: When *globally* setting sensitivity to a single value, e.g. sensitivity=3, then the same parameter is applied to all exemplars. On the other hand, *exemplar specific* sensitivity parameters can be used by defining sensitivity as a vector of length n, where n is the number of rows in training_items. The sensitivity vector values then represent the sensitivity parameters for all exemplars in training_items at the corresponding row positions. E.g. if there are 3 exemplars (rows) in training_items, then setting sensitivity to c(1,1,3) assigns sensitivity = 1 to the first two exemplars, and sensitivity = 3 for the third exemplar. See Note 2.

weights - dimensional attention weights. Order corresponds to the definitions of x1, x2, ... in tr and training_items. Has to be a vector with length n-1 , where n equals to nFeat dimension weights, e.g. of length 2 when there are three features, leaving out the *last* dimension. A constraint in the GCM is that all attentional weights sum to 1. Thus, the sum of n-1 weights should be equal to or smaller than 1, too. The last n-th weight then is computed within the model with: 1 - (sum of n-1 feature weights). When setting the weights to 1/nFeat = equal weights. See Note 3.

choice_bias - Category choice biases. Has to be a vector with length n-1, where n equals to nCats category biases, leaving out the last category bias, under the constraint that all biases sum to 1. Order corresponds to the definitions of cat1, cat2 in the training_items. The sum of n-1 choice biases has to be equal to or smaller than 1. Setting the weights to 1/nCats = no choice bias. The bias for the last category then is computed in the model with: 1 - (sum of nCats-1 choice biases). See Note 3.

gamma - decision constant/ response scaling. Can take any value between 0 (towards more probabilistic) and +infinity (towards deterministic choices). Nosofsky (2011) suggests setting gamma higher than 1 when individual participants' data are considered. See Note 2.

`r_metric` - distance metric. Set to 1 (city-block) or 2 (Euclidean). See Nosofsky (2011), and Note 4, for more details.

`p` - similarity gradient. Set to 1 (exponential) or 2 (Gaussian). See Nosofsky (2011), for more details.

Value

A matrix of probabilities for category responses (columns) for each stimulus (rows) presented to the model (e.g. test trials). Stimuli and categories are in the same order as presented to the model in `st`, see below.

Note

1. Please note that setting mp = 1 or e.g. mp = 5 globally, will yield identical response probabilities. Crucially, memory strength is indifferent from the category choice bias parameter, if (and only if) mp's vary between categories, without varying within categories. Thus, the memory strength parameter can therefore be interpreted in terms of an exemplar choice bias (potentially related to categorization accuracy). In addition, if exemplar specific mp's are assigned during parameter fitting, one might want to calculate the natural log of the corresponding estimates, enabling direct comparisons between mp's indicating different directions, e.g. -log(.5) = log(2), for loss and gain, respectively, which are equal regarding their extent into different directions.

2. Theoretically, increasing global sensitivity indicates that categorization mainly relies on the most similar exemplars, usually making choices less probabilistic. Thus sensitivity c is likely to be correlated with gamma. See Navarro (2007) for a detailed discussion. However, it is possible to assume exemplar specific sensitivities, or specificity. Then, exemplars with lower sensitivity parameters will have a stronger impact on stimulus similarity and thus categorization behavior for stimuli. See Rodrigues & Murre (2007) for a related study.

3. Setting only the n-1 instead of all n feature weights (or bias parameters) eases model fitting procedures, in which the last weight always is a linear combination of the n-1 weights.

4. See Tversky & Gati (1982) for further info on r. In brief summary, r=2 (usually termed Euclidean), then a large difference on only one feature outweighs small differences on all features. In contrast, if r=1 (usually termed City-Block or Manhattan distance) both aspects contribute to an equal extent to the distance. Thus, r = 2 comes with the assumption that small differences in all features may be less recognized, than a large noticable differences on one feature, which may be depend on confusability of the stimuli or on the nature of the given task domain (perceptual or abstract).

Author(s)

Rene Schlegelmilch, Andy Wills

References

Minda, J. P., & Smith, J. D. (2011). Prototype models of categorization: Basic formulation, predictions, and limitations. Formal approaches in categorization, 40-64.

Navarro, D. J. (2007). On the interaction between exemplar-based concepts and a response scaling process. Journal of Mathematical Psychology, 51(2), 85-98.

Nosofsky, R. M. (1984). Choice, similarity, and the context theory of classification. *Journal of Experimental Psychology: Learning, memory, and cognition, 10*(1), 104.

Nosofsky, R. M. (2011). The generalized context model: An exemplar model of classification. In Pothos, E.M. & Wills, A.J. *Formal approaches in categorization*. Cambridge University Press.

Rodrigues, P. M., & Murre, J. M. (2007). Rules-plus-exception tasks: A problem for exemplar models?. *Psychonomic Bulletin & Review, 14*(4), 640-646.

Tversky, A., & Gati, I. (1982). Similarity, separability, and the triangle inequality. *Psychological review, 89*(2), 123.

Examples

```
## Three Categories with 2 Training Items each, and repeatedly presented
## transfer/test items (from nosof94train()). Each item has three
## features with two (binary) values: memory strength (st$mp and
## 'mem' column in st$training_items are optional) is
## equal for all exemplars

st<-list(
    sensitivity = 3,
    weights = c(.2,.3),
    choice_bias = c(1/3),
    gamma = 1,
    mp = 1,
    r_metric = 1,
    p = 1,
    nCats = 2,
    nFeat=3
)

## training item definitions
st$training_items <- as.data.frame(
    t(matrix(cbind(c(1,0,1,1,1,0,0),c(1,1,0,2,1,0,0),
                   c(0,1,0,5,0,1,0),c(0,0,1,1,0,1,0)),
             ncol=4, nrow=7,
             dimnames=list(c("stim","x1", "x2", "x3",
                             "cat1", "cat2", "mem"),
                           c(1:4)))))

st$tr <- nosof94train()

## get the resulting predictions for the test items

## columns of the output correspond to category numbers as defined
## above rows correspond to the column indices of the test_items

stsimGCM(st)

## columns of the output correspond to category numbers as defined
## above rows correspond to the column indices of the test_items

## Example 2
```

```
## Same (settings) as above, except: memory strength is 5 times higher
## for for some exemplars
st$mp<-5

## which exemplars?
## training item definitions
st$training_items <- as.data.frame(
    t(matrix(cbind(c(1,0,1,1,1,0,1),c(1,1,0,2,1,0,0),
                   c(0,1,0,5,0,1,0),c(0,0,1,1,0,1,1)),
             ncol=4, nrow=7,
             dimnames=list(c("stim","x1", "x2", "x3",
                             "cat1", "cat2", "mem"),
                           c(1:4)))))
## exemplars in row 1 and 4 will receive a memory strength of 5
## get predictions
stsimGCM(st)

## Example 3
## Same (settings) as above, except: memory strength is item specific
## for the two exemplars i.e. memory strength boost is not the same
## for both exemplars (3 for the first in row 1, and 5 for the
## second exemplar in row 4)
st$mp<-c(3,5)


## get predictions
stsimGCM(st)
```

`thegrid` *Ordinal adequacy results for all catlearn simulations*

Description

Records results of all ordinal adequacy tests registered in the catlearn package.

Usage

```
data(thegrid)
```

Format

A data frame with the following columns:

id Unique identifier number for each entry into the grid. When making a new entry, use the next available integer.

cirp The CIRP (Canonical Independently Replicated Phenomenon) against which a model was tested. This must correspond precisely to the name of a data set in the catlearn package.

model A one-word description of the model being tested. Simulations in the same row of The Grid must have precisely the same one-word description. Note, this is not the name of the function used to run the simulation, nor the name of the model implementation function. It is a descriptive term, defined by the modeler.

result Indicates the result of the simulation. 1 = passes ordinal adequacy test, 0 = fails ordinal adequacy test, OES = outside explanatory scope (in other words, this is not a result the model was designed to accommodate), 'pending' = the function listed in 'sim' is currently being written or tested.

sim The name of the catlearn function used to run the simulation.

oat The name of the catlearn function used to perform the Ordinal Adequacy Test.

Details

The Grid is a means of centrally recording the results of model simulations centrally, within the catlearn package. For further discussion, see Wills et al. (2016).

Author(s)

Andy J. Wills <andy@willslab.co.uk>

Source

```
citation('catlearn')
```

References

Wills, A.J., O'Connell, G., Edmunds, C.E.R. & Inkster, A.B. (2016). Progress in modeling through distributed collaboration: Concepts, tools, and category-learning examples. *The Psychology of Learning and Motivation*.

Index