Package ‘hann’

July 25, 2025

Version 1.0

Date 2025-07-23

Title Hopfield Artificial Neural Networks

Description Builds and optimizes Hopfield artificial neural networks (Hopfield, 1982, <doi:10.1073/pnas.79.8.2554>). One-layer and three-layer models are implemented. The energy of the Hopfield network is minimized with formula from Krotov and Hopfield (2016, <doi:10.48550/ARXIV.1606.01164>). Optimization (supervised learning) is done through a gradient-based method. Classification is done with S3 methods predict(). Parallelization with 'OpenMP' is used if available during compilation.

URL `https://github.com/emmanuelparadis/hann`

BugReports `https://github.com/emmanuelparadis/hann/issues`

License GPL-3

NeedsCompilation yes

Author Emmanuel Paradis [aut, cre, cph] (ORCID: <`https://orcid.org/0000-0003-3092-2199`>)

Maintainer Emmanuel Paradis <`Emmanuel.Paradis@ird.fr`>

Repository CRAN

Date/Publication 2025-07-25 15:40:07 UTC

Contents

hann-package *Hopfield Artificial Neural Networks*

Description

hann provides tools to build Hopfield-based artificial neural networks. Two types of networks can be built: one-layer Hopfield network, and three-layer network with a hidden (convoluted) layer.

The complete list of functions can be displayed with `library(help = hann)`. See the vignette "IntroductionHopfieldNetworks" for several examples.

More information on **hann** can be found at https://github.com/emmanuelparadis/hann.

Author(s)

Emmanuel Paradis

Maintainer: Emmanuel Paradis <Emmanuel.Paradis@ird.fr>

buildSigma *Hopfield Network Energy*

Description

Minimize the energy of the Hopfield network.

Usage

```
buildSigma(xi, n = 20, nrep = 100, quiet = FALSE)
```

Arguments

`xi`	a matrix of patterns coded with 1 and -1.
`n`	the parameter of the energy function (integer).
`nrep`	the number of attempts.
`quiet`	a logical value indicating whether to print the details for each attempt.

Details

The number of columns in `xi` is equal to the size of the Hopfield network (i.e., the number of input neurons denoted as N), whereas the number of columns is the number of memories denoted as K (Krotov and Hopfield, 2016).

A random vector 'sigma' is first generated and then updated in order to minimize the energy level of the Hopfield network. The convergence to a low energy level depends on the initial values in 'sigma', so the procedure is repeated several times. The vector with the lowest energy level is returned.

Value

a vector of integers (-1/1). The length of this vector (N) is equal to the number of columns in `xi`.

Author(s)

Emmanuel Paradis

References

Krotov, D. and Hopfield, J. J. (2016) Dense associative memory for pattern recognition. doi:10.48550/ARXIV.1606.01164.

See Also

`hann1`

Examples

```
xi <- matrix(NA, K <- 1000, N <- 60)
xi[] <- sample(c(1L, -1L), K * N, TRUE)
(sigma <- buildSigma(xi))
```

`control.hann` *Parameters for Neural Network Optimization*

Description

Set the parameters for the Hopfield artificial neural network optimization.

Usage

```
control.hann(...)
```

Arguments

`...`	named arguments to be modified (see examples).

Details

When the user modifies one or several parameters by giving them as named arguments, if some names are incorrect they are ignored with a warning.

The parameters with their default values are:

- `iterlim = 100`: an integer giving the number of iterations.
- `quiet = FALSE`: a logical controlling whether to print the value of the objective (loss) function at each iteration.

- `quasinewton = FALSE`: a logical. If `TRUE`, quasi-Newton steps are performed (not recommended unless for networks with few parameters and/or for a small number of iterations).
- `fullhessian = FALSE`: (ignored if `quasinewton = FALSE`) a logical, by default only some blocks of the Hessian matrix are computed. If `TRUE`, the full Hessian matrix is computed (very time consuming).
- `trace.error = FALSE`: a logical. If `TRUE`, the error rate is printed at each iteration of the optimization process.
- `wolfe = FALSE`: a logical. If `TRUE`, Wolfe's conditions are tested and printed at each iteration.
- `target = 0.001`: the target value of the loss function to stop the optimization.
- `beta = 0.2`: the hyperparameter of the activation function.
- `mc.cores = 1`: an integer. The number of cores used when computing the loss function.

If `mc.cores` is greater than one, the optimization process calls a multithreaded code using OMP. So, do *not* do this together with functions from the package **parallel**. On the other hand, if you leave this parameter to its default value, you should be able to run several optimizations in parallel, for instance with `mclapply`.

See the vignette for applications.

Value

a list with named elements as detailed above.

Note

For the moment, the parameter `mc.cores` is accepted only by `hann1`.

Author(s)

Emmanuel Paradis

References

https://en.wikipedia.org/wiki/Wolfe_conditions

See Also

`hann1`

Examples

```
control.hann() # default values
ctrl <- control.hann(iterlim = 1000)
ctrl

## verbose is not a parameter:
ctrl <- control.hann(iterlim = 1000, verbose = TRUE)
```

hann1 *One-layer Hopfield ANN*

Description

This optimizes a one-layer Hopfield-based artificial neural network. The structure of the network is quite simple: a Hopfield network with N input neurons all connected to C output neurons. The number of parameters (N and C) is determined by the input data: `xi` has N columns (which is also the length of `sigma`) and the number of unique values of `classes` is equal to C.

See the vignette of this package for an example and some background.

Usage

```
hann1(xi, sigma, classes, net = NULL, control = control.hann())

## S3 method for class 'hann1'
print(x, details = FALSE, ...)
```

Arguments

`xi`	a matrix of patterns with K rows.
`sigma`	a vector coding the Hopfield network.
`classes`	the classes of the patterns (vector of length K).
`net, x`	an object of class `"hann1"`.
`control`	the control parameters.
`details`	a logical value (whether to print the parameter values of the network).
`...`	further arguments passed to `print.default`.

Details

By default, the parameters of the neural network are initialized with random values from a uniform distribution between -1 and 1 (except the biases which are initialized to zero).

If an object of `"hann1"` is given to the argument `net`, then its parameter values are used to initialize the parameters of the network.

The main control parameters are given as a list to the `control` argument. They are detailed in the page of the function `control.hann()`.

Value

an object of class `"hann1"` with the following elements:

`parameters`	a list with one matrix, `W`, and one vector, `bias`.
`sigma`	the Hopfield network.
`beta`	the hyperparameter of the activation function.
`call`	the function call.

Author(s)

Emmanuel Paradis

References

Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, **79**, 2554–2558. doi:10.1073/pnas.79.8.2554.

Krotov, D. and Hopfield, J. J. (2016) Dense associative memory for pattern recognition. doi:10.48550/ARXIV.1606.01164.

See Also

`buildSigma`, `predict.hann1`

`hann3` *Three-layer Hopfield ANN*

Description

This optimizes a three-layer Hopfield-based artificial neural network. The network is made of a Hopfield network with N input neurons all connected to H hidden neurons. The latter are all connected together (convoluation) which is equivalent to defining two hidden layers. Each hidden neuron is connected to C output neurons. The values of the parameters N and C are determined by the input data: `xi` has N columns (which is also the length of `sigma`) and the number of unique values of `classes` is equal to C. The value of H must be given by the user (a default of half the number of input neurons is defined).

See the vignette of this package for an example.

Usage

```
hann3(xi, sigma, classes,  H = 0.5 * length(sigma),
      net = NULL, control = control.hann())

## S3 method for class 'hann3'
print(x, details = FALSE, ...)
```

Arguments

`xi`	a matrix of patterns with K rows.
`sigma`	a vector coding the Hopfield network.
`classes`	the classes of the patterns (vector of length K).
`H`	the number of numbers in the hidden layer; by default half the number of input neurons (rounded to the lowest integer if the latter is odd).
`net, x`	an object of class `"hann1"`.

`control`	the control parameters.
`details`	a logical value (whether to print the parameter values of the network).
`...`	further arguments passed to `print.default`.

Details

By default, the parameters of the neural network are initialized with random values from a uniform distribution between -1 and 1 (expect the biases which are initialized to zero).

If an object of `"hann3"` is given to the argument `net`, then its parameter values are used to initialize the parameters of the network.

The main control parameters are given as a list to the `control` argument. They are detaild in the page of the function `control.hann()`.

Value

an object of class `"hann3"` with the following elements:

`parameters`	a list with three matrices, `W1`, `W2`, and `W3`, and two vectors, `bias1` and `bias3`.
`sigma`	the Hopfield network.
`beta`	the hyperparameter of the activation function.
`call`	the function call.

Author(s)

Emmanuel Paradis

References

Hopfield, J. J. (1982) Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences, USA*, **79**, 2554–2558. doi:10.1073/pnas.79.8.2554.

Krotov, D. and Hopfield, J. J. (2016) Dense associative memory for pattern recognition. doi:10.48550/ARXIV.1606.01164.

See Also

`buildSigma`, `control.hann`, `predict.hann3` `predict.hann1` *Prediction*

Description

Classification of patterns with Hopfield-based artificial neural networks.

Usage

```
## S3 method for class 'hann1'
predict(object, patterns, rawsignal = TRUE, ...)
## S3 method for class 'hann3'
predict(object, patterns, rawsignal = TRUE, ...)
```

Arguments

`object`	an object of class `"hann1"` or `"hann3"`.
`patterns`	the patterns to be classified.
`rawsignal`	a logical value (see details).
`...`	(ignored).

Details

The patterns have to be coded in the same way than the matrix `xi` used to train the networks.

If `rawsignal = TRUE`, the raw signal of each neuron is output for each pattern. Otherwise, a classification of each pattern is done by finding the neuron with the largest signal.

Value

If `rawsignal = TRUE` a matrix; if `rawsignal = FALSE` a vector.

Author(s)

Emmanuel Paradis

See Also

`hann1`, `hann3`

Index