Package 'NeuralGas'

July 16, 2021

Type Package

Title Neural Gas Prototype Learning

Version 1.0

Date 2021-05-25

Author Josh Taylor

Maintainer <jtay@alumni.rice.edu>

Description Provides Online and Batch versions of Neural Gas learning for vector quantization. Partial support was provided by E. Merényi, Rice University, D75802-790000.

License MIT + file LICENSE

Imports Rcpp (>= 1.0.5)

LinkingTo Rcpp, RcppArmadillo, RcppParallel, VQTools

RoxygenNote 7.1.1

Encoding UTF-8

Suggests knitr,
rmarkdown,
dplyr,
tidyr,
ggplot2,
ggpubr,
kableExtra

R topics documented:

NGBatch *Neural Gas Batch Learning*

Description

Neural Gas Batch Learning

Usage

```
NGBatch(
  X,
  W,
  lambda = 0.25 * nrow(W),
  lambda_decay = 0.9,
  lambda_schedule = NULL,
  tol_delBMU = 1,
  tol_delQE = 1,
  max_epochs = -1,
  XLabel = NULL,
  parallel = TRUE,
  verbose = TRUE
)
```

Arguments

X	a data matrix, one observation per row.
W	the number of NG prototypes and their initialization, see NGWInitialization.
lambda	the starting value of the neighborhood factor for cooperative learning. Typical values are 25-50% of the number of prototypes in the network. Optional, default = 0.25*nrow(W).
lambda_decay	a multiplicative decay factor applied to lambda after every learning epoch. Ex.: lambda(t) = lambda(t-1)*decay. Optional, default = 0.9. Must be < 1 to ensure convergence. Higher values require longer training time, but typically give better quantization.
lambda_schedule	a named vector to set a custom annealing schedule for lambda instead of the multiplicative decay controlled by lambda and lambda_decay. names(lambda_schedule) should be integers defining the epoch **through** which the corresponding elements of the vector are applied. Optional; if given, this schedule over-rides any multiplicative annealing specified by lambda and lambda_decay.
tol_delBMU	tolerance controlling convergence of the learning, see NGConvergence. Optional, default = 1.
tol_delQE	tolerance controlling convergence of the learning, see NGConvergence. Optional, default = 1.
max_epochs	tolerance controlling convergence of the learning, see NGConvergence. Optional, default = -1.
parallel	whether to compute in parallel (recommended, default = TRUE).
verbose	whether to print learning history to the console after each epoch. Default = TRUE.

`Xlabel` optionally, a vector of labels for the data in the rows of X. These can be of any type (character, factor, numeric) but will be converted to integers internally. If given, Xlabel allows reporting additional quality measures during the learning process as described in `NGLearnHist`.

Details

Neural gas finds prototypes `W` which minimize the following cost function:

$$\sum_i \sum_j h_{ij} d(x_i, w_j)$$

where the neighborhood function

$$h_{ij} = exp(-k_{ij}/\lambda)$$

and k_{ij} = the rank of $d(x_i, w_j)$, with respect to all other $d(x_i, w_k)$. Ranks are ascending, and by convention start at 0 (instead of 1).

Batch learning updates the prototypes after presentation of all data to the network. The update rule is implemented according to the method of: Cottrell, M., Hammer, B., Hasenfuss, A., & Villmann, T. (2006). *Batch and median neural gas*

Value

a list with components:

W the learned prototype matrix, one prototype per row

age the age of the network (number of epochs trained). age=max_epochs (if given), otherwise it records the number of epochs required to attain convergence according to the criteria in `tol_delBMU` and `tol_delQE`.

lambda_start the value of lambda at the beginning of learning (the value of initial lambda given)

lambda_end the value of lambda at the end of leanring

lambda_decay the supplied decay factor

lambda_schedule the schedule set in `lambda_schedule`, if given

tol_delBMU the supplied value of tol_delBMU

tol_delQE the supplied value of tol_delQE

max_epochs the supplied value of max_epochs

exec_time execution time, in minutes

LearnHist a data frame recording various learning histories, see `NGLearnHist`

`NGConvergence` *Convergence for NeuralGas Objects*

Description

Note: This is not a function, merely a description of the convergence criteria used in the `NGBatch` and `NGOnline` functions.

Details

NeuralGas learning is terminated by the first occurrence of the following convergence criteria reaching their user-supplied tolerances:

- The training age (number of learning epochs performed) exceeding the parameter `max_epochs`
- The `tol_delBMU` AND `tol_delQE` tolerances being met for three consecutive epochs. These criteria, described below, measure the stability of the vector quantization over training time.

`delBMU` reports the percentage of training vectors whose BMU has changed from one epoch to the next. For example, `delBMU < tol_delBMU = 1` means that less than 1% of the data have changed their BMU from epoch `t-1` to `t`.

`delQE` reports the average (absolute) percentage change in each datum's quantization error (QE) from one epoch to the next. For example, `delQE < tol_delQE = 1` means that, on average, each training vector's QE has changed by less than 1% from epoch `t-1` to `t`.

The default value `max_epochs = -1` indicates convergence (training cessation) is measured ONLY by `delBMU` and `delQE`. Thus, to train for a fixed number of epochs `E`, set `max_epochs = E` and `tol_delBMU = tol_delQE = 0`.

For online learning an epoch is equivalent to `N` learning iterations, where `N` = number of training vectors.

During training early termination can be achieved by user interruption (typing CTRL-C in the R console window). Upon detecting early termination, both `NGBatch` and `NGOnline` will return the current state of the network (existing values of the prototypes and any training history logged).

`NGLearnHist` *Learning History for NeuralGas Objects*

Description

Note: This is not a function, merely a description of the `LearnHist` data frame returned from the `NGBatch` and `NGOnline` functions.

Details

At the end of each epoch the following monitoring measures are computed, reported, and stored in the `LearnHist` data frame returned in the output list. For online learning one 'Epoch' is equivalent to `N` learning iterations, where `N` is the number of training vectors: `N = nrow(X)` where `X` is the matrix of training data.

Epoch the epoch for which measures are reported

alpha the effective learning rate

lambda the effective neighborhood lambda

Cost the value of the NG cost function, divided by the number of data vectors `N`. The purpose of division by `N` is to put `Cost` on a similar scale to `MQE`.

MQE Mean Quantization Error of all data

NhbEff the effect of the current value of lambda on the network, calculated as `Cost / MQE`. `NhbEff` is always >= 1, with values = 1 indicating no neighborhood effect, which occurs as lambda -> 0.

delCost the relative absolute percent change in Cost, from epoch(t-1) to epoch(t), = abs(Cost(t) - Cost(t-1)) / Cost(t) * 100

delQE the relative absolute percentage change in the quantization error for each data vector, from epoch(t-1) to epoch(t), averaged over all data.

delBMU the proportion of data whose BMU has changed from epoch(t-1) to epoch(t)

RFEnt Normalized Shannon Entropy of the VQ mapping at epoch(t)

If data labels are given (by supplying a value for `Xlabel`) we can use the learned VQ mapping to project these labels onto the prototypes. Each prototype's label (denoted RFL = Receptive Field Label) is decided by plurality vote of the labels of the data in its receptive field (RF). In the presence of labels, additional measures are computed, reported and stored in `LearnHist`:

RFLPur Average of the individual Purity scores of each receptive field. The Purity score of each RF = \#(label(x)==RFL)/\#(RF), for all data x in the RF. This measures how much label confusion exists in the RF; ideally, if the labels indicate well separated classes / clusters we would have Purity=1. The value reported in RFLPur is the average Purity score of each RF, weighted by the RF's size (number of data vectors mapped to it). Purity -> 0 as intra-RF label confusion increases.

RFLUnq The number of unique RFLabels. This is a helpful measure to determine how well the prototypes represent X in situations with unbalanced class size, particularly when there are rare classes (of small size). Ideally, all unique labels found in Xlabel should be represented by some (set of) prototype(s).

\itemRFLHellThe Hellinger Distance between the empirical categorical distributions of Xlabels and RFLabels. RFLHell=0 means the distributions perfectly align; any value > 0 indicates disagreement. For example, assume the data are labeled one of A, B, or C, and (empirically, according to Xlabel), pX(A) = 0.20, pX(B) = 0.30 and pX(C) = 0.50. If the VQ has produced the mapping pRF(A) = 0.20, pRF(B) = 0.30 and pRF(C) = 0.50, then the distributions of data and RF labels perfectly agree, and RFLHell = 0.

`NGOnline` *Neural Gas Online Learning*

Description

Neural Gas Online Learning

Usage

```
NGOnline(
  X,
  W,
  alpha = 0.5,
  alpha_decay = 0.9^(1/nrow(X)),
  alpha_schedule = NULL,
  lambda = 0.25 * nrow(W),
  lambda_decay = 0.9^(1/nrow(X)),
  lambda_schedule = NULL,
  tol_delBMU = 1,
  tol_delQE = 1,
  max_epochs = -1,
```

```
  XLabel = NULL,
  parallel = TRUE,
  verbose = TRUE,
  Xseed = NULL
)
```

Arguments

X	a data matrix, one observation per row.
W	the number of NG prototypes and their initialization, see NGWInitialization.
alpha	the starting value of the learning rate used for prototype updates. Optional, default = 0.5.
alpha_decay	a multiplicative decay factor applied to alpha after every iteration. Ex.: alpha(t) = alpha(t-1)*decay. Optional, default = 0.9^(1/nrow(X)).
alpha_schedule	a named vector to set a custom annealing schedule for alpha instead of the multiplicative decay controlled by alpha and alpha_decay. names(alpha_schedule) should be integers defining the epoch **through** which the corresponding elements of the vector are applied. Optional; if given, this schedule over-rides any multiplicative annealing specified by alpha and alpha_decay.
lambda	the starting value of the neighborhood factor for cooperative learning. Typical values are 25-50% of the number of prototypes in the network. Optional, default = 0.25*nrow(W).
lambda_decay	a multiplicative decay factor applied to lambda after every learning epoch. Ex.: lambda(t) = lambda(t-1)*decay. Optional, default = 0.9^(1/nrow(X)).
lambda_schedule	a named vector defining an annealing schedule for lambda, in the same format as alpha_schedule. Mandatory if alpha_schedule is set.
tol_delBMU	tolerance controlling convergence of the learning, see NGConvergence. Optional, default = 1.
tol_delQE	tolerance controlling convergence of the learning, see NGConvergence. Optional, default = 1.
max_epochs	tolerance controlling convergence of the learning, see NGConvergence. Optional, default = -1.
parallel	whether to compute in parallel (recommended, default = TRUE).
verbose	whether to print learning history to the console after each epoch. Default = TRUE.
Xseed	the seed value controlling the random sampling of X for presentation to the network at each iteration. Optional, default = NULL does not set this random seed.
Xlabel	optionally, a vector of labels for the data in the rows of X. These can be of any type (character, factor, numeric) but will be converted to integers internally. If given, Xlabel allows reporting additional quality measures during the learning process as described in NGLearnHist.

Details

Neural gas finds prototypes W which minimize the following cost function:

$$\sum_i \sum_j h_{ij} d(x_i, w_j)$$

where the neighborhood function

$$h_{ij} = exp(-k_{ij}/\lambda)$$

and k_{ij} = the rank of $d(x_i, w_j)$, with respect to all other $d(x_i, w_k)$. Ranks are ascending, and by convention start at 0 (instead of 1).

Online learning updates the prototypes after presentation of a single datum to the network (one learning iteration). The update rule is

$$w(t+1) = w(t) + \alpha h_{ij}(x_i - w_j)$$

Value

a list with components:

W the learned prototype matrix, one prototype per row

age the age of the network (number of epochs trained). age=max_epochs (if given), otherwise it records the number of epochs required to attain convergence according to the criteria in `tol_delBMU` and `tol_delQE`.

alpha_start the value of alpha at the beginning of learning (the value of initial lambda given)

alpha_end the value of alpha at the end of learning

alpha_decay the supplied decay factor

alpha_schedule the schedule set in `alpha_schedule`, if given

lambda_start the value of lambda at the beginning of learning (the value of initial lambda given)

lambda_end the value of lambda at the end of leanring

lambda_decay the supplied decay factor

lambda_schedule the schedule set in `lambda_schedule`, if given

tol_delBMU the supplied value of tol_delBMU

tol_delQE the supplied value of tol_delQE

max_epochs the supplied value of max_epochs

exec_time execution time, in minutes

LearnHist a data frame recording various learning histories, see `NGLearnHist`

`NGWInitialization` *Prototype Initialization for NeuralGas Objects*

Description

Note: This is not a function, merely a description of the types of prototype initialization understood by the `NGBatch` and `NGOnline` functions.

Details

Both NGBatch and NGOnline require the input parameter W which describes both the number of prototypes in the network, and how they are initialized. As such, the value supplied for W can take one of the three following forms:

a single number giving the number of prototypes in the network. In this case, prototypes are initialized randomly and uniformly in the range of training data X.

a length=2 vector giving the number of prototypes (first element) and the random seed used to initialize them (second element). Use this option for random prototype initialization with a fixed seed for reproducibility.

a matrix giving the initial prototypes in its rows. ncols(W) should = ncol(X), and the number of prototypes in the network is set equal to nrow(W) If initial prototypes are given they should occupy the same range as X, as they will be linearly scaled internally from \[max(X),min(X)\] to \[0,1\] prior to learning.

---

vis_NGLearnHist *Visualization of NeuralGas Learning Histories*

Description

Visualization of NeuralGas Learning Histories

Usage

```
vis_NGLearnHist(NGLearnHist)
```

Arguments

NGLearnHist	a LearnHist data frame, as returned from a call to NGOnline or NGBatch.

Details

This function is a wrapper to view plots of del_BMU, del_QE, MQE and Neural Gas's Cost as a function of training epoch.

Value

none, a ggplot is produced

Index