

Package ‘automl’

July 22, 2025

Type Package

Title Deep Learning with Metaheuristic

Version 1.3.2

BugReports <https://github.com/aboulaboul/automl/issues>

Description Fits from simple regression to highly customizable deep neural networks either with gradient descent or metaheuristic, using automatic hyper parameters tuning and custom cost function.
A mix inspired by the common tricks on Deep Learning and Particle Swarm Optimization.

URL <https://aboulaboul.github.io/automl>
<https://github.com/aboulaboul/automl>

License GNU General Public License

Encoding UTF-8

LazyData TRUE

Imports stats, utils, parallel

Suggests datasets

RoxygenNote 6.1.1

NeedsCompilation no

Author Alex Boulangé [aut, cre]

Maintainer Alex Boulangé <aboul@free.fr>

Repository CRAN

Date/Publication 2020-01-16 11:40:09 UTC

Contents

automl_predict	2
automl_train	3
automl_train_manual	4
autopar	5
hpar	8
pso	10

Index**12**

automl_predict	<i>automl_predict</i>
----------------	-----------------------

Description

Predictions function, to apply a trained model on datas

Usage

```
automl_predict(model, X, layoutputnum)
```

Arguments

model	model trained previously with automl_train or automl_train_manual
X	inputs matrix or data.frame (containing numerical values only)
layoutputnum	which layer number to output especially for auto encoding (default 0: no particular layer, the last one)

Examples

```
##REGRESSION (predict Sepal.Length given other parameters)
data(iris)
xmat <- as.matrix(cbind(iris[,2:4], as.numeric(iris$Species)))
ymat <- iris[,1]
amlmodel <- automl_train_manual(Xref = xmat, Yref = ymat,
  hpar = list(modexec = 'trainwpso', verbose = FALSE))
res <- cbind(ymat, automl_predict(model = amlmodel, X = xmat))
colnames(res) <- c('actual', 'predict')
head(res)
#
## Not run:
##CLASSIFICATION (predict Species given other Iris parameters)
data(iris)
xmat = iris[,1:4]
lab2pred <- levels(iris$Species)
lghlab <- length(lab2pred)
iris$Species <- as.numeric(iris$Species)
ymat <- matrix(seq(from = 1, to = lghlab, by = 1), nrow(xmat),
  lghlab, byrow = TRUE)
ymat <- (ymat == as.numeric(iris$Species)) + 0
amlmodel <- automl_train_manual(Xref = xmat, Yref = ymat,
  hpar = list(modexec = 'trainwpso', verbose = FALSE))
res <- cbind(ymat, round(automl_predict(model = amlmodel, X = xmat)))
colnames(res) <- c(paste('act',lab2pred, sep = '_'),
  paste('pred',lab2pred, sep = '_'))
head(res)

## End(Not run)
```

automl_train	<i>automl_train</i>
--------------	---------------------

Description

The multi deep neural network automatic train function (several deep neural networks are trained with automatic hyperparameters tuning, best model is kept)

This function launches the [automl_train_manual](#) function by passing it parameters for each particle at each converging step

Usage

```
automl_train(Xref, Yref, autopar = list(), hpar = list(), mdlref = NULL)
```

Arguments

Xref	inputs matrix or data.frame (containing numerical values only)
Yref	target matrix or data.frame (containing numerical values only)
autopar	list of parameters for hyperparameters optimization, see autopar section Not mandatory (the list is preset and all arguments are initialized with default value) but it is advisable to adjust some important arguments for performance reasons (including processing time)
hpar	list of parameters and hyperparameters for Deep Neural Network, see hpar section Not mandatory (the list is preset and all arguments are initialized with default value) but it is advisable to adjust some important arguments for performance reasons (including processing time)
mdlref	model trained with automl_train to start training with saved hpar and autopar (not the model) nb: manually entered parameters above override loaded ones

Examples

```
## Not run:
##REGRESSION (predict Sepal.Length given other Iris parameters)
data(iris)
xmat <- cbind(iris[,2:4], as.numeric(iris$Species))
ymat <- iris[,1]
amlmodel <- automl_train(Xref = xmat, Yref = ymat)

## End(Not run)
##CLASSIFICATION (predict Species given other Iris parameters)
data(iris)
xmat = iris[,1:4]
lab2pred <- levels(iris$Species)
lghlab <- length(lab2pred)
iris$Species <- as.numeric(iris$Species)
```

automl_train_manual	<i>automl_train_manual</i>
---------------------	----------------------------

The base deep neural network train function (one deep neural network trained without automatic hyperparameters tuning)

```
automl_train_manual(Xref, Yref, hpar = list(), mdlref = NULL)
```

Xref	inputs matrix or data.frame (containing numerical values only)
Yref	target matrix or data.frame (containing numerical values only)
hpar	list of parameters and hyperparameters for Deep Neural Network, see hpar section Not mandatory (the list is preset and all arguments are initialized with default value) but it is advisable to adjust some important arguments for performance reasons (including processing time)
mdlref	model trained with automl_train or automl_train_manual to start training from a saved model (shape, weights...) for fine tuning nb: manually entered parameters above override loaded ones

[illegible]

```

                                psopartpopsize = 50))

#with PSO and custom cost function
f <- 'J=abs((y-yhat)/y)'
f <- c(f, 'J=sum(J[!is.infinite(J)],na.rm=TRUE)')
f <- c(f, 'J=(J/length(y))')
f <- paste(f, collapse = ';')
amlmodel <- automl_train_manual(Xref = xmat, Yref = ymat,
                                hpar = list(modexec = 'trainwpso',
                                              numiterations = 30,
                                              psopartpopsize = 50,
                                              costcustformul = f))

##CLASSIFICATION (predict Species given other Iris parameters)
data(iris)
xmat = iris[,1:4]
lab2pred <- levels(iris$Species)
lghlab <- length(lab2pred)
iris$Species <- as.numeric(iris$Species)
ymat <- matrix(seq(from = 1, to = lghlab, by = 1), nrow(xmat), lghlab, byrow = TRUE)
ymat <- (ymat == as.numeric(iris$Species)) + 0
#with gradient descent and 2 hidden layers
amlmodel <- automl_train_manual(Xref = xmat, Yref = ymat,
                                hpar = list(layersshape = c(10, 10, 0),
                                              layersacttype = c('tanh', 'relu', 'sigmoid'),
                                              layersdropoprob = c(0, 0, 0)))
#with gradient descent and no hidden layer (logistic regression)
amlmodel <- automl_train_manual(Xref = xmat, Yref = ymat,
                                hpar = list(layersshape = c(0),
                                              layersacttype = c('sigmoid'),
                                              layersdropoprob = c(0)))

#with PSO and softmax
amlmodel <- automl_train_manual(Xref = xmat, Yref = ymat,
                                hpar = list(modexec = 'trainwpso',
                                              layersshape = c(10, 0),
                                              layersacttype = c('relu', 'softmax'),
                                              layersdropoprob = c(0, 0),
                                              numiterations = 50,
                                              psopartpopsize = 50))

## End(Not run)

```

Description

List of parameters to allow multi deep neural network automatic hyperparameters tuning with Particle Swarm Optimization

Not mandatory (the list is preset and all arguments are initialized with default value) but it is advisable to adjust some important arguments for performance reasons (including processing time)

Arguments

psopartpopsize	number of particles in swarm, the main argument that should be tuned (default value 8, which is quite low) #tuning priority 1
psosxxx	see psos for other PSO specific arguments details
numiterations	number of convergence steps between particles (hyperparameters), default value 3) #tuning priority 1
auto_modexec	if 'TRUE' the type of Neural Net optimization will be randomly chosen between 'trainwgrad' and 'trainwpso' for each particle default value is 'FALSE' (so default value of argument 'modexec' in automl_train_manual function, actually 'trainwgrad' as default is more suited to large data volume) the value can be forced if defined in hpar list
auto_runtype	if '2steps' the 2 following steps will be run automatically (default value is 'normal'): 1st overfitting, the goal is performance 2nd regularization, the goal is generalization nb: 'overfitting' or 'regularization' may be directly specified to avoid the 2 steps
auto_minibatchsize	see below
auto_minibatchsize_min	see below
auto_minibatchsize_max	'auto_minibatch' default value 'TRUE' for automatic adjustment of 'minibatchsize' argument in automl_train_manual function the minimum and maximum value for 'minibatchsize' correspond to 2 to the power value (default 0 for 'auto_minibatchsize_min' and 9 for 'auto_minibatchsize_max')
auto_learningrate	see below
auto_learningrate_min	see below
auto_learningrate_max	'auto_learningrate' default value 'TRUE' for automatic adjustment of 'learningrate' argument in automl_train_manual function the minimum and maximum value for 'learningrate' correspond to 10 to the power negative value (default -5 for 'auto_learningrate_min' and -2 for 'auto_learningrate_max')
auto_beta1	see below
auto_beta2	'auto_beta1' and 'auto_beta2' default value 'TRUE' for automatic adjustment of 'beta1' and 'beta2' argument in automl_train_manual function
auto_psopartpopsize	see below
auto_psopartpopsize_min	see below

auto_psopartpopsize_max	‘auto_psopartpopsize’ default value ‘TRUE’ for automatic adjustment of ‘psopartpopsize’ argument in automl_train_manual function (concern only ‘modexec’ set to ‘trainwps’) the minimum and maximum value for ‘psopartpopsize’ ; default 2 for ‘auto_psopartpopsize_min’ and 50 for ‘auto_psopartpopsize_max’)
auto_lambda	see below
auto_lambda_min	see below
auto_lambda_max	‘auto_lambda’ default value ‘FALSE’ for automatic adjustment of ‘lambda’ regularization argument in automl_train_manual function the minimum and maximum value for ‘lambda’ correspond to 10 to the power value (default -2) for ‘auto_lambda_min’ and (default 4) for ‘auto_lambda_max’)
auto_psovelocitymaxratio	see below
auto_psovelocitymaxratio_min	see below
auto_psovelocitymaxratio_max	‘auto_psovelocitymaxratio’ default value ‘TRUE’ for automatic adjustment of ‘psovelocitymaxratio’ PSO velocity max ratio argument in automl_train_manual function the minimum and maximum value for ‘psovelocitymaxratio’; default 0.01 for ‘auto_psovelocitymaxratio_min’ and 0.5 for ‘auto_psovelocitymaxratio_max’
auto_layers	see below (‘auto_layers’ default value ‘TRUE’ for automatic adjustment of layers shape in automl_train_manual function)
auto_layers_min	(linked to ‘auto_layers’ above, set hpar ‘layersshape’ and ‘layersacttype’) the minimum number of hidden layers (default 1 no hidden layer)
auto_layers_max	(linked to ‘auto_layers’ above, set hpar ‘layersshape’ and ‘layersacttype’) the maximum number of hidden layers (default 2)
auto_layersnodes_min	(linked to ‘auto_layers’ above, set hpar ‘layersshape’ and ‘layersacttype’) the minimum number of nodes per layer (default 3)
auto_layersnodes_max	(linked to ‘auto_layers’ above, set hpar ‘layersshape’ and ‘layersacttype’) the maximum number of nodes per layer (default 33)
auto_layersdropo	see below
auto_layersdropoprob_min	see below
auto_layersdropoprob_max	‘auto_layersdropo’ default value ‘FALSE’ for automatic adjustment of hpar ‘layersdropoprob’ in automl_train_manual function) the minimum and maximum value for ‘layersdropoprob’; default 0.05 for ‘auto_layersdropoprob_min’ and 0.75 for ‘auto_layersdropoprob_max’

seed	seed for reproducibility (default 4)
nbccores	number of cores used to parallelize particles optimization, not available on Windows (default 1, automatically reduced if not enough cores)
verbose	to display or not the costs at each iteration for each particle (default TRUE)
subtimelimit	time limit in seconds for sub modelizations to avoid waiting too long for a specific particle to finish its modelization (default 3600)

back to [automl_train](#)

hpar

Deep Neural Net parameters and hyperparameters

Description

List of Neural Network parameters and hyperparameters to train with gradient descent or particle swarm optimization

Not mandatory (the list is preset and all arguments are initialized with default value) but it is advisable to adjust some important arguments for performance reasons (including processing time)

Arguments

modexec	‘trainwgrad’ (the default value) to train with gradient descent (suitable for all volume of data) ‘trainwpsa’ to train using Particle Swarm Optimization, each particle represents a set of neural network weights (CAUTION: suitable for low volume of data, time consuming for medium to large volume of data)
---------	--

Below specific arguments to ‘trainwgrad’ execution mode

learningrate	learningrate alpha (default value 0.001) #tuning priority 1
beta1	see below
beta2	‘Momentum’ if beta1 different from 0 and beta2 equal 0) ‘RMSprop’ if beta1 equal 0 and beta2 different from 0 ‘adam optimization’ if beta1 different from 0 and beta2 different from 0 (default) (default value beta1 equal 0.9 and beta2 equal 0.999) #tuning priority 2
lrdecayrate	learning rate decay value (default value 0, no learning rate decay, 1e-6 should be a good value to start with) #tuning priority 4
chkgradevery	epoch interval to run gradient check function (default value 0, for debug only)
chkgradeepsilon	epsilon value for derivative calculations and threshold test in gradient check function (default 0.0000001)

Below specific arguments to 'trainwpso' execution mode

psosxxx see [psos](#) for PSO specific arguments details

costcustformul custom cost formula (default '', no custom cost function)
 standard input variables: yhat (prediction), y (target actual value)
 custom input variables: any variable declared in hpar may be used via alias mydl
 (ie: hpar(list = (foo = 1.5)) will be used in custom cost formula as mydl\$foo))
 result: J
 see 'automl_train_manual' example using Mean Average Percentage Error cost
 function
 nb: X and Y matrices used as input into automl_train_manual or automl_train_manual
 functions are transposed (features in rows and cases in columns)

Below arguments for both execution modes

numiterations number of training epochs (default value 50))
 #tuning priority 1

seed seed for reproductibility (default 4)

minibatchsize mini batch size, 2 to the power 0 for stochastic gradient descent (default 2 to the
 power 5) #tuning priority 3

layersshape number of nodes per layer, each nodes number initialize a hidden layer
 output layer nodes number, may be left to 0 it will be automatically set by Y
 matrix shape
 default value one hidden layer with 10 nodes: c(10, 0)
 #tuning priority 4

layersacttype activation function for each layer; 'linear' for no activation or 'sigmoid', 'relu'
 or 'reluleaky' or 'tanh' or 'softmax' (softmax for output layer only supported in
 trainwpso exec mode)
 output layer activation function may be left to '', default value 'linear' for re-
 gression, 'sigmoid' for classification
 nb: layersacttype parameter vector must have same length as layersshape pa-
 rameter vector
 default value c('relu', '')
 #tuning priority 4

layersdropoprob drop out probability for each layer, continuous value from 0 to less than 1 (give
 the percentage of matrix weight values to drop out randomly)
 nb: layersdropoprob parameter vector must have same length as layersshape
 parameter vector
 default value no drop out: c(0, 0)
 #tuning priority for regularization

printcostevery epoch interval to test and print costs (train and cross validation cost: default
 value 10, for 1 test every 10 epochs)

testcvsize size of cross validation sample, 0 for no cross validation sample (default 10, for
 10 percent)

testgainunder	threshold to stop the training if the gain between last train or cross validation cost is smaller than the threshold, 0 for no stop test (default 0.000001)
costtype	cost type function name 'mse' or 'crossentropy' or 'custom' 'mse' for Mean Squared Error, set automatically for continuous target type ('mape' Mean Absolute Percentage Error may be specified) 'crossentropy' set automatically for binary target type 'custom' set automatically if 'costcustformul' different from ''
lambda	regularization term added to cost function (default value 0, no regularization)
batchnor_mom	batch normalization momentum for j and B (default 0, no batch normalization, may be set to 0.9 for deep neural net)
epsil	epsilon the low value to avoid dividing by 0 or log(0) in cost function, etc ... (default value 1e-12)
verbose	to display or not the costs and the shapes (default TRUE)

back to [automl_train](#), [automl_train_manual](#)

See Also

Deep Learning specialization from Andrew NG on Coursera

pso

PSO parameters and hyperparameters

Description

List of parameters and hyperparameters for Particle Swarm Optimization

Arguments

All PSO parameters and hyperparameters are preset with default value

psopartpopsize number of particles in swarm (discrete value)
(‘autopar’ context: default value 8, which means that 8 different neural net hyperparameters sets will be tested
(‘hpar’ context: default value 50, which means that 50 neural net weights sets will be tested
#tuning priority 1

(impact on memory consumption)

CAUTION: you should only change the values below if you know what you are doing

psovarvalmin Minimum value for particles positions (default value -10)

psovarvalmax	maximum value for particles positions (default value 10)
psovelocitymaxratio	ratio applied to limit velocities (continuous value between 0 and 1, default value 0.2)
psoinertiadampratio	inertia damp ratio (continuous value between 0 and 1, default value 1 equivalent to OFF)
psokappa	kappa (default value 1)
psophi1	Phi 1 (default value 2.05)
psophi2	Phi 2 (default value 2.05)

back to [autopar](#), [hpar](#)

See Also

PSO video tutorial from Yarpiz

Index

automl_predict, [2](#)
automl_train, [2](#), [3](#), [3](#), [4](#), [8](#), [10](#)
automl_train_manual, [2–4](#), [4](#), [6](#), [7](#), [10](#)
autopar, [3](#), [5](#), [11](#)

hpar, [3](#), [4](#), [6](#), [7](#), [8](#), [11](#)

pso, [6](#), [9](#), [10](#)