Package ‘mxnet’

March 30, 2017

Type Package

Title MXNet

Version 0.9.5

Date 2015-12-23

Author Tianqi Chen, Qiang Kou, Tong He

Maintainer Qiang Kou <qkou@umail.iu.edu>

Repository DMLC

Description MXNet is a deep learning framework designed for both efficiency and flexibility. It allows you to mix the flavours of deep learning programs together to maximize the efficiency and your productivity.

License Apache License (== 2.0)

URL https://github.com/dmlc/mxnet/tree/master/R-package

BugReports https://github.com/dmlc/mxnet/issues

Imports methods,Rcpp (>= 0.12.1),DiagrammeR (>= 0.9.0),visNetwork (>= 1.0.3),data.table,jsonlite,magrittr,stringr

Suggests testthat,mlbench,knitr,rmarkdown,imager

LinkingTo Rcpp

VignetteBuilder knitr

RoxygenNote 5.0.1

R topics documented:

arguments . . . 4
as.array.MXNDArray . . . 4
as.matrix.MXNDArray . . . 4
ctx . . . 5
dim.MXNDArray . . . 5
graph.viz . . . 5
is.mx.context . . . 6
is.mx.dataiter . . . 6
is.mx.ndarray . . . 7
is.mx.symbol . . . 7
is.num.in.vect . . . 7
length.MXNDArray . . . 8
mx.apply . . . 8
mx.callback.early.stop . . . 9
mx.callback.log.speedometer . . . 9
mx.callback.log.train.metric . . . 10
mx.callback.save.checkpoint . . . 10
mx.cpu . . . 11
mx.ctx.default . . . 11
mx.exec.backward . . . 11
mx.exec.forward . . . 12
mx.exec.update.arg.arrays . . . 12
mx.exec.update.aux.arrays . . . 12
mx.exec.update.grad.arrays . . . 13
mx.gpu . . . 13
mx.gru . . . 13
mx.gru.forward . . . 14
mx.gru.inference . . . 15
mx.init.create . . . 16
mx.init.internal.default . . . 16
mx.init.normal . . . 17
mx.init.uniform . . . 17
mx.init.Xavier . . . 17
mx.io.arrayiter . . . 18
mx.io.extract . . . 18
mx.kv.create . . . 19
mx.lr_scheduler.FactorScheduler . . . 19
mx.lr_scheduler.MultiFactorScheduler . . . 20
mx.lstm . . . 20
mx.lstm.forward . . . 21
mx.lstm.inference . . . 22
mx.metric.accuracy . . . 22
mx.metric.custom . . . 23
mx.metric.mae . . . 23
mx.metric.mse . . . 23
mx.metric.rmse . . . 24

mx.metric.rmsle . . . 24
mx.metric.top_k_accuracy . . . 24
mx.mlp . . . 25
mx.model.FeedForward.create . . . 26
mx.model.load . . . 27
mx.model.save . . . 27
mx.nd.array . . . 28
mx.nd.copyto . . . 28
mx.nd.load . . . 29
mx.nd.ones . . . 29
mx.nd.save . . . 30
mx.nd.zeros . . . 30
mx.opt.adadelta . . . 31
mx.opt.adagrad . . . 31
mx.opt.adam . . . 32
mx.opt.create . . . 33
mx.opt.get.updater . . . 33
mx.opt.rmsprop . . . 34
mx.opt.sgd . . . 34
mx.profiler.config . . . 35
mx.profiler.state . . . 35
mx.rnn . . . 36
mx.rnn.forward . . . 37
mx.rnn.inference . . . 37
mx.rnorm . . . 38
mx.runif . . . 39
mx.set.seed . . . 39
mx.simple.bind . . . 40
mx.symbol.Concat . . . 40
mx.symbol.Group . . . 41
mx.symbol.infer.shape . . . 41
mx.symbol.load . . . 42
mx.symbol.load.json . . . 42
mx.symbol.save . . . 43
mx.symbol.Variable . . . 43
mxnet . . . 44
mxnet.export . . . 44
Ops.MXNDArray . . . 44
outputs . . . 45
predict.MXFeedForwardModel . . . 45
print.MXNDArray . . . 46

Index **47**

arguments — *Get the arguments of symbol.*

Description

Get the arguments of symbol.

Usage

```
arguments(x)
```

Arguments

x	The input symbol

as.array.MXNDArray — *as.array operator overload of mx.ndarray*

Description

as.array operator overload of mx.ndarray

Usage

```
## S3 method for class 'MXNDArray'
as.array(nd)
```

Arguments

nd	The mx.ndarray

as.matrix.MXNDArray — *as.matrix operator overload of mx.ndarray*

Description

as.matrix operator overload of mx.ndarray

Usage

```
## S3 method for class 'MXNDArray'
as.matrix(nd)
```

Arguments

nd	The mx.ndarray

ctx — *Get the context of mx.ndarray*

Description

Get the context of mx.ndarray

Usage

```
ctx(nd)
```

Arguments

nd	The mx.ndarray

dim.MXNDArray — *Dimension operator overload of mx.ndarray*

Description

Dimension operator overload of mx.ndarray

Usage

```
## S3 method for class 'MXNDArray'
dim(nd)
```

Arguments

nd	The mx.ndarray

graph.viz — *Convert symbol to Graphviz or visNetwork visualisation.*

Description

Convert symbol to Graphviz or visNetwork visualisation.

Usage

```
graph.viz(symbol, shape = NULL, direction = "TD", type = "graph",
  graph.width.px = NULL, graph.height.px = NULL)
```

Arguments

`symbol`	a `string` representing the symbol of a model.
`shape`	a `numeric` representing the input dimensions to the symbol.
`direction`	a `string` representing the direction of the graph, either TD or LR.
`type`	a `string` representing the rendering engine of the the graph, either graph or vis.
`graph.width.px`	a `numeric` representing the size (width) of the graph. In pixels
`graph.height.px`	a `numeric` representing the size (height) of the graph. In pixels

Value

a graph object ready to be displayed with the `print` function.

`is.mx.context` *Check if the type is mxnet context.*

Description

Check if the type is mxnet context.

Usage

```
is.mx.context(x)
```

Value

Logical indicator

`is.mx.dataiter` *Judge if an object is mx.dataiter*

Description

Judge if an object is mx.dataiter

Usage

```
is.mx.dataiter(x)
```

Value

Logical indicator

is.mx.ndarray — *Check if src.array is mx.ndarray*

Description

Check if src.array is mx.ndarray

Usage

```
is.mx.ndarray(src.array)
```

Value

Logical indicator

Examples

```
mat = mx.nd.array(1:10)
is.mx.ndarray(mat)
mat2 = 1:10
is.mx.ndarray(mat2)
```

is.mx.symbol — *Judge if an object is mx.symbol*

Description

Judge if an object is mx.symbol

Usage

```
is.mx.symbol(x)
```

Value

Logical indicator

is.num.in.vect — *Helper function for top-k accuracy*

Description

Helper function for top-k accuracy

Usage

```
is.num.in.vect(vect, num)
```

length.MXNDArray *Length operator overload of mx.ndarray*

Description

Length operator overload of mx.ndarray

Usage

```
## S3 method for class 'MXNDArray'
length(nd)
```

Arguments

nd	The mx.ndarray

mx.apply *Apply symbol to the inputs.*

Description

Apply symbol to the inputs.

Usage

```
mx.apply(x, ...)
```

Arguments

x	The symbol to be applied
kwargs	The keyword arguments to the symbol

mx.callback.early.stop

Early stop with different conditions

Description

Early stopping applying different conditions: hard thresholds or epochs number from the best score. Tested with "epoch.end.callback" function.

Usage

```
mx.callback.early.stop(train.metric = NULL, eval.metric = NULL,
  bad.steps = NULL, maximize = FALSE, verbose = FALSE)
```

Arguments

train.metric	Numeric. Hard threshold for the metric of the training data set (optional)
eval.metric	Numeric. Hard threshold for the metric of the evaluating data set (if set, optional)
bad.steps	Integer. How much epochs should gone from the best score? Use this option with evaluation data set
maximize	Logical. Do your model use maximizing or minimizing optimization?
verbose	Logical

mx.callback.log.speedometer

Calculate the training speed

Description

Calculate the training speed

Usage

```
mx.callback.log.speedometer(batch.size, frequency = 50)
```

Arguments

frequency	The frequency of the training speed update
batch_size	The batch size

mx.callback.log.train.metric

Log training metric each period

Description

Log training metric each period

Usage

```
mx.callback.log.train.metric(period, logger = NULL)
```

Arguments

period	The number of batch to log the training evaluation metric
logger	The logger class

mx.callback.save.checkpoint

Save checkpoint to files each period iteration.

Description

Save checkpoint to files each period iteration.

Usage

```
mx.callback.save.checkpoint(prefix, period = 1)
```

Arguments

prefix	The prefix of the model checkpoint.

`mx.cpu` *Create a mxnet CPU context.*

Description

Create a mxnet CPU context.

Arguments

`dev.id`	optional, default=0 The device ID, this is meaningless for CPU, included for interface compatiblity.

Value

The CPU context.

`mx.ctx.default` *Set/Get default context for array creation.*

Description

Set/Get default context for array creation.

Usage

```
mx.ctx.default(new = NULL)
```

Arguments

`new,`	optional takes `mx.cpu()` or `mx.gpu(id)`, new default ctx.

Value

The default context.

`mx.exec.backward` *Peform an backward on the executors This function will MUTATE the state of exec*

Description

Peform an backward on the executors This function will MUTATE the state of exec

Usage

```
mx.exec.backward(exec, ...)
```

`mx.exec.forward` *Peform an forward on the executors This function will MUTATE the state of exec*

Description

Peform an forward on the executors This function will MUTATE the state of exec

Usage

```
mx.exec.forward(exec, is.train = TRUE)
```

`mx.exec.update.arg.arrays` *Update the executors with new arrays This function will MUTATE the state of exec*

Description

Update the executors with new arrays This function will MUTATE the state of exec

Usage

```
mx.exec.update.arg.arrays(exec, arg.arrays, match.name = FALSE,
  skip.null = FALSE)
```

`mx.exec.update.aux.arrays` *Update the executors with new arrays This function will MUTATE the state of exec*

Description

Update the executors with new arrays This function will MUTATE the state of exec

Usage

```
mx.exec.update.aux.arrays(exec, arg.arrays, match.name = FALSE,
  skip.null = FALSE)
```

mx.exec.update.grad.arrays

Update the executors with new arrays This function will MUTATE the state of exec

Description

Update the executors with new arrays This function will MUTATE the state of exec

Usage

```
mx.exec.update.grad.arrays(exec, arg.arrays, match.name = FALSE,
  skip.null = FALSE)
```

mx.gpu

Create a mxnet GPU context.

Description

Create a mxnet GPU context.

Arguments

dev.id	optional, default=0 The GPU device ID, starts from 0.

Value

The GPU context.

mx.gru

Training GRU Unrolled Model

Description

Training GRU Unrolled Model

Usage

```
mx.gru(train.data, eval.data = NULL, num.gru.layer, seq.len, num.hidden,
  num.embed, num.label, batch.size, input.size, ctx = mx.ctx.default(),
  num.round = 10, update.period = 1, initializer = mx.init.uniform(0.01),
  dropout = 0, optimizer = "sgd", ...)
```

Arguments

`train.data`	mx.io.DataIter or list(data=R.array, label=R.array) The Training set.
`eval.data`	mx.io.DataIter or list(data=R.array, label=R.array), optional The validation set used for validation evaluation during the progress.
`num.gru.layer`	integer The number of the layer of gru.
`seq.len`	integer The length of the input sequence.
`num.hidden`	integer The number of hidden nodes.
`num.embed`	integer The output dim of embedding.
`num.label`	integer The number of labels.
`batch.size`	integer The batch size used for R array training.
`input.size`	integer The input dim of one-hot encoding of embedding
`ctx`	mx.context, optional The device used to perform training.
`num.round`	integer, default=10 The number of iterations over training data to train the model.
`update.period`	integer, default=1 The number of iterations to update parameters during training period.
`initializer`	initializer object. default=mx.init.uniform(0.01) The initialization scheme for parameters.
`dropout`	float, default=0 A number in [0,1) containing the dropout ratio from the last hidden layer to the output layer.
`optimizer`	string, default="sgd" The optimization method.
`...`	other parameters passing to `mx.gru`/.

Value

model A trained gru unrolled model.

`mx.gru.forward` *Using forward function to predict in gru inference model*

Description

Using forward function to predict in gru inference model

Usage

```
mx.gru.forward(model, input.data, new.seq = FALSE)
```

Arguments

`model`	gru model A gru inference model
`input.data,`	array.matrix The input data for forward function
`new.seq`	boolean, default=FALSE Whether the input is the start of a new sequence

Value

result A list(prob=prob, model=model) containing the result probability of each label and the model.

---

mx.gru.inference *Create a GRU Inference Model*

Description

Create a GRU Inference Model

Usage

```
mx.gru.inference(num.gru.layer, input.size, num.hidden, num.embed, num.label,
  batch.size = 1, arg.params, ctx = mx.cpu(), dropout = 0)
```

Arguments

`num.gru.layer`	integer The number of the layer of gru.
`input.size`	integer The input dim of one-hot encoding of embedding
`num.hidden`	integer The number of hidden nodes.
`num.embed`	integer The output dim of embedding.
`num.label`	integer The number of labels.
`batch.size`	integer, default=1 The batch size used for R array training.
`arg.params`	list The batch size used for R array training.
`ctx`	mx.context, optional Model parameter, list of name to NDArray of net's weights.
`dropout`	float, default=0 A number in [0,1) containing the dropout ratio from the last hidden layer to the output layer.

Value

model list(rnn.exec=integer, symbol=mxnet symbol, num.rnn.layer=integer, num.hidden=integer, seq.len=integer, batch.size=integer, num.embed=integer) A gru inference model.

mx.init.create *Create initialization of argument like arg.array*

Description

Create initialization of argument like arg.array

Usage

```
mx.init.create(initializer, shape.array, ctx, skip.unknown = TRUE)
```

Arguments

initializer	The initializer.
shape.array	named-list The shape of the weights
ctx	mx.context The context of the weights
skip.unknown	Whether skip the unknown weight types

mx.init.internal.default *Internal default value initialization scheme.*

Description

Internal default value initialization scheme.

Usage

```
mx.init.internal.default(name, shape, ctx, allow.unknown = FALSE)
```

Arguments

name	the name of the variable.
shape	the shape of the array to be generated.

mx.init.normal — *Create a initializer that initialize the weight with normal(0, sd)*

Description

Create a initializer that initialize the weight with normal(0, sd)

Usage

```
mx.init.normal(sd)
```

Arguments

sd	The standard deviation of normal distribution

mx.init.uniform — *Create a initializer that initialize the weight with uniform [-scale, scale]*

Description

Create a initializer that initialize the weight with uniform [-scale, scale]

Usage

```
mx.init.uniform(scale)
```

Arguments

scale	The scale of uniform distribution

mx.init.Xavier — *Xavier initializer*

Description

Create a initializer which initialize weight with Xavier or similar initialization scheme.

Usage

```
mx.init.Xavier(rnd_type = "uniform", factor_type = "avg", magnitude = 3)
```

Arguments

`rnd_type`	A string of `character` indicating the type of distribution from which the weights are initialized.
`factor_type`	A string of `character`.
`magnitude`	A `numeric` number indicating the scale of random number range.

`mx.io.arrayiter` *Create MXDataIter compatible iterator from R's array*

Description

Create MXDataIter compatible iterator from R's array

Usage

```
mx.io.arrayiter(data, label, batch.size = 128, shuffle = FALSE)
```

Arguments

`data`	The data array.
`label`	The label array.
`batch.size`	The batch size used to pack the array.
`shuffle`	Whether shuffle the data

`mx.io.extract` *Extract a certain field from DataIter.*

Description

Extract a certain field from DataIter.

Usage

```
mx.io.extract(iter, field)
```

mx.kv.create — *Create a mxnet KVStore.*

Description

Create a mxnet KVStore.

Arguments

`type`	string(default="local") The type of kvstore.

Value

The kvstore.

mx.lr_scheduler.FactorScheduler — *Learning rate scheduler. Reduction based on a factor value.*

Description

Learning rate scheduler. Reduction based on a factor value.

Usage

```
mx.lr_scheduler.FactorScheduler(step, factor_val, stop_factor_lr = 1e-08,
  verbose = TRUE)
```

Arguments

`step`	(integer) Schedule learning rate after n updates
`factor`	(double) The factor for reducing the learning rate

Value

scheduler function

mx.lr_scheduler.MultiFactorScheduler

Multifactor learning rate scheduler. Reduction based on a factor value at different steps.

Description

Multifactor learning rate scheduler. Reduction based on a factor value at different steps.

Usage

```
mx.lr_scheduler.MultiFactorScheduler(step, factor_val, stop_factor_lr = 1e-08,
  verbose = TRUE)
```

Arguments

`step`	(array of integer) Schedule learning rate after n updates
`factor`	(double) The factor for reducing the learning rate

Value

scheduler function

mx.lstm

Training LSTM Unrolled Model

Description

Training LSTM Unrolled Model

Usage

```
mx.lstm(train.data, eval.data = NULL, num.lstm.layer, seq.len, num.hidden,
  num.embed, num.label, batch.size, input.size, ctx = mx.ctx.default(),
  num.round = 10, update.period = 1, initializer = mx.init.uniform(0.01),
  dropout = 0, optimizer = "sgd", ...)
```

Arguments

`train.data`	mx.io.DataIter or list(data=R.array, label=R.array) The Training set.
`eval.data`	mx.io.DataIter or list(data=R.array, label=R.array), optional The validation set used for validation evaluation during the progress.
`num.lstm.layer`	integer The number of the layer of lstm.
`seq.len`	integer The length of the input sequence.

`num.hidden`	integer The number of hidden nodes.
`num.embed`	integer The output dim of embedding.
`num.label`	integer The number of labels.
`batch.size`	integer The batch size used for R array training.
`input.size`	integer The input dim of one-hot encoding of embedding
`ctx`	mx.context, optional The device used to perform training.
`num.round`	integer, default=10 The number of iterations over training data to train the model.
`update.period`	integer, default=1 The number of iterations to update parameters during training period.
`initializer`	initializer object. default=mx.init.uniform(0.01) The initialization scheme for parameters.
`dropout`	float, default=0 A number in [0,1) containing the dropout ratio from the last hidden layer to the output layer.
`optimizer`	string, default="sgd" The optimization method.
`...`	other parameters passing to `mx.lstm`/.

Value

model A trained lstm unrolled model.

`mx.lstm.forward` *Using forward function to predict in lstm inference model*

Description

Using forward function to predict in lstm inference model

Usage

```
mx.lstm.forward(model, input.data, new.seq = FALSE)
```

Arguments

`model`	lstm model A Lstm inference model
`input.data,`	array.matrix The input data for forward function
`new.seq`	boolean, default=FALSE Whether the input is the start of a new sequence

Value

result A list(prob=prob, model=model) containing the result probability of each label and the model.

mx.lstm.inference *Create a LSTM Inference Model*

Description

Create a LSTM Inference Model

Usage

```
mx.lstm.inference(num.lstm.layer, input.size, num.hidden, num.embed, num.label,
  batch.size = 1, arg.params, ctx = mx.cpu(), dropout = 0)
```

Arguments

num.lstm.layer	integer The number of the layer of lstm.
input.size	integer The input dim of one-hot encoding of embedding
num.hidden	integer The number of hidden nodes.
num.embed	integer The output dim of embedding.
num.label	integer The number of labels.
batch.size	integer, default=1 The batch size used for R array training.
arg.params	list The batch size used for R array training.
ctx	mx.context, optional Model parameter, list of name to NDArray of net's weights.
dropout	float, default=0 A number in [0,1) containing the dropout ratio from the last hidden layer to the output layer.

Value

model list(rnn.exec=integer, symbol=mxnet symbol, num.rnn.layer=integer, num.hidden=integer, seq.len=integer, batch.size=integer, num.embed=integer) A lstm inference model.

mx.metric.accuracy *Accuracy metric for classification*

Description

Accuracy metric for classification

Usage

```
mx.metric.accuracy
```

Format

An object of class `mx.metric` of length 3.

mx.metric.custom — *Helper function to create a customized metric*

Description

Helper function to create a customized metric

Usage

```
mx.metric.custom(name, feval)
```

mx.metric.mae — *MAE (Mean Absolute Error) metric for regression*

Description

MAE (Mean Absolute Error) metric for regression

Usage

```
mx.metric.mae
```

Format

An object of class `mx.metric` of length 3.

mx.metric.mse — *MSE (Mean Squared Error) metric for regression*

Description

MSE (Mean Squared Error) metric for regression

Usage

```
mx.metric.mse
```

Format

An object of class `mx.metric` of length 3.

mx.metric.rmse *RMSE (Root Mean Squared Error) metric for regression*

Description

RMSE (Root Mean Squared Error) metric for regression

Usage

```
mx.metric.rmse
```

Format

An object of class `mx.metric` of length 3.

mx.metric.rmsle *RMSLE (Root Mean Squared Logarithmic Error) metric for regression*

Description

RMSLE (Root Mean Squared Logarithmic Error) metric for regression

Usage

```
mx.metric.rmsle
```

Format

An object of class `mx.metric` of length 3.

mx.metric.top_k_accuracy *Top-k accuracy metric for classification*

Description

Top-k accuracy metric for classification

Usage

```
mx.metric.top_k_accuracy
```

Format

An object of class `mx.metric` of length 3.

mx.mlp *Convenience interface for multiple layer perceptron*

Description

Convenience interface for multiple layer perceptron

Usage

```
mx.mlp(data, label, hidden_node = 1, out_node, dropout = NULL,
  activation = "tanh", out_activation = "softmax",
  device = mx.ctx.default(), ...)
```

Arguments

data	the input matrix. Only mx.io.DataIter and R array/matrix types supported.
label	the training label. Only R array type supported.
hidden_node	a vector containing number of hidden nodes on each hidden layer as well as the output layer.
out_node	the number of nodes on the output layer.
dropout	a number in [0,1) containing the dropout ratio from the last hidden layer to the output layer.
activation	either a single string or a vector containing the names of the activation functions.
out_activation	a single string containing the name of the output activation function.
device	whether train on cpu (default) or gpu.
...	other parameters passing to mx.model.FeedForward.create/
eval_metric	the evaluation metric/

Examples

```
require(mlbench)
data(Sonar, package="mlbench")
Sonar[,61] = as.numeric(Sonar[,61])-1
train.ind = c(1:50, 100:150)
train.x = data.matrix(Sonar[train.ind, 1:60])
train.y = Sonar[train.ind, 61]
test.x = data.matrix(Sonar[-train.ind, 1:60])
test.y = Sonar[-train.ind, 61]
model = mx.mlp(train.x, train.y, hidden_node = 10, out_node = 2, out_activation = "softmax",
               learning.rate = 0.1)
preds = predict(model, test.x)
```

`mx.model.FeedForward.create` *Create a MXNet Feedforward neural net model with the specified training.*

Description

Create a MXNet Feedforward neural net model with the specified training.

Usage

```
mx.model.FeedForward.create(symbol, X, y = NULL, ctx = NULL,
  begin.round = 1, num.round = 10, optimizer = "sgd",
  initializer = mx.init.uniform(0.01), eval.data = NULL,
  eval.metric = NULL, epoch.end.callback = NULL,
  batch.end.callback = NULL, array.batch.size = 128,
  array.layout = "auto", kvstore = "local", verbose = TRUE,
  arg.params = NULL, aux.params = NULL, ...)
```

Arguments

`symbol`	The symbolic configuration of the neural network.
`X`	mx.io.DataIter or R array/matrix The training data.
`y`	R array, optional label of the data This is only used when X is R array.
`ctx`	mx.context or list of mx.context, optional The devices used to perform training.
`begin.round`	integer (default=1) The initial iteration over the training data to train the model.
`num.round`	integer (default=10) The number of iterations over training data to train the model.
`optimizer`	string, default="sgd" The optimization method.
`initializer,`	initializer object. default=mx.init.uniform(0.01) The initialization scheme for parameters.
`eval.data`	mx.io.DataIter or list(data=R.array, label=R.array), optional The validation set used for validation evaluation during the progress
`eval.metric`	function, optional The evaluation function on the results.
`epoch.end.callback`	function, optional The callback when iteration ends.
`batch.end.callback`	function, optional The callback when one mini-batch iteration ends.
`array.batch.size`	integer (default=128) The batch size used for R array training.
`array.layout`	can be "auto", "colmajor", "rowmajor", (detault=auto) The layout of array. "rowmajor" is only supported for two dimensional array. For matrix, "rowmajor" means dim(X) = c(nexample, nfeatures), "colmajor" means dim(X) = c(nfeatures, nexample) "auto" will auto detect the layout by match the feature size, and will report error when X is a square matrix to ask user to explicitly specify layout.

`kvstore`	string (default="local") The parameter synchronization scheme in multiple devices.
`verbose`	logical (default=TRUE) Specifies whether to print information on the iterations during training.
`arg.params`	list, optional Model parameter, list of name to NDArray of net's weights.
`aux.params`	list, optional Model parameter, list of name to NDArray of net's auxiliary states.

Value

model A trained mxnet model.

`mx.model.load` *Load model checkpoint from file.*

Description

Load model checkpoint from file.

Usage

```
mx.model.load(prefix, iteration)
```

Arguments

`prefix`	string prefix of the model name
`iteration`	integer Iteration number of model we would like to load.

`mx.model.save` *Save model checkpoint into file.*

Description

Save model checkpoint into file.

Usage

```
mx.model.save(model, prefix, iteration)
```

Arguments

`model`	The feedforward model to be saved.
`prefix`	string prefix of the model name
`iteration`	integer Iteration number of model we would like to load.

mx.nd.array — *Create a new* `mx.ndarray` *that copies the content from src on ctx.*

Description

Create a new `mx.ndarray` that copies the content from src on ctx.

Usage

```
mx.nd.array(src.array, ctx = NULL)
```

Arguments

`src.array`	Source array data of class `array`, `vector` or `matrix`.
`ctx`	optional The context device of the array. mx.ctx.default() will be used in default.

Value

An `mx.ndarray`

An Rcpp_MXNDArray object

Examples

```
mat = mx.nd.array(x)
mat = 1 - mat + (2 * mat)/(mat + 0.5)
as.array(mat)
```

mx.nd.copyto — *Generate an mx.ndarray object on ctx, with data copied from src*

Description

Generate an mx.ndarray object on ctx, with data copied from src

Usage

```
mx.nd.copyto(src, ctx)
```

Arguments

`src`	The source mx.ndarray object.
`ctx`	The target context.

mx.nd.load — *Load an mx.nd.array object on disk*

Description

Load an mx.nd.array object on disk

Usage

```
mx.nd.load(filename)
```

Arguments

filename	the filename (including the path)

Examples

```
mat = mx.nd.array(1:3)
mx.nd.save(mat, 'temp.mat')
mat2 = mx.nd.load('temp.mat')
as.array(mat)
as.array(mat2)
```

mx.nd.ones — *Generate an mx.ndarray object with ones*

Description

Generate an mx.ndarray object with ones

Usage

```
mx.nd.ones(shape, ctx = NULL)
```

Arguments

shape	the dimension of the `mx.ndarray`
ctx	optional The context device of the array. mx.ctx.default() will be used in default.

Examples

```
mat = mx.nd.ones(10)
as.array(mat)
mat2 = mx.nd.ones(c(5,5))
as.array(mat)
mat3 = mx.nd.ones(c(3,3,3))
as.array(mat3)
```

mx.nd.save — *Save an mx.nd.array object*

Description

Save an mx.nd.array object

Usage

```
mx.nd.save(ndarray, filename)
```

Arguments

ndarray	the mx.nd.array object
filename	the filename (including the path)

Examples

```
mat = mx.nd.array(1:3)
mx.nd.save(mat, 'temp.mat')
mat2 = mx.nd.load('temp.mat')
as.array(mat)
as.array(mat2[[1]])
```

mx.nd.zeros — *Generate an mx.nd.array object with zeros*

Description

Generate an mx.nd.array object with zeros

Usage

```
mx.nd.zeros(shape, ctx = NULL)
```

Arguments

shape	the dimension of the mx.nd.array
ctx	optional The context device of the array. mx.ctx.default() will be used in default.

Examples

```
mat = mx.nd.zeros(10)
as.array(mat)
mat2 = mx.nd.zeros(c(5,5))
as.array(mat)
mat3 = mx.nd.zeroes(c(3,3,3))
as.array(mat3)
```

`mx.opt.adadelta` *Create an AdaDelta optimizer with respective parameters.*

Description

AdaDelta optimizer as described in Zeiler, M. D. (2012). *ADADELTA: An adaptive learning rate method.* http://arxiv.org/abs/1212.5701

Usage

```
mx.opt.adadelta(rho = 0.9, epsilon = 1e-05, wd = 0, rescale.grad = 1,
  clip_gradient = NULL)
```

Arguments

`rho`	float, default=0.90 Decay rate for both squared gradients and delta x.
`epsilon`	float, default=1e-5 The constant as described in the thesis.
`wd`	float, default=0.0 L2 regularization coefficient add to all the weights.
`rescale.grad`	float, default=1.0 rescaling factor of gradient.
`clip_gradient`	float, optional clip gradient in range [-clip_gradient, clip_gradient].

`mx.opt.adagrad` *Create an AdaGrad optimizer with respective parameters. AdaGrad optimizer of Duchi et al., 2011,*

Description

This code follows the version in http://arxiv.org/pdf/1212.5701v1.pdf Eq(5) by Matthew D. Zeiler, 2012. AdaGrad will help the network to converge faster in some cases.

Usage

```
mx.opt.adagrad(learning.rate = 0.05, epsilon = 1e-08, wd = 0,
  rescale.grad = 1, clip_gradient = NULL, lr_scheduler = NULL)
```

Arguments

learning.rate	float, default=0.05 Step size.
epsilon	float, default=1e-8
wd	float, default=0.0 L2 regularization coefficient add to all the weights.
rescale.grad	float, default=1.0 rescaling factor of gradient.
clip_gradient	float, optional clip gradient in range [-clip_gradient, clip_gradient].
lr_scheduler	function, optional The learning rate scheduler.

---

mx.opt.adam — *Create an Adam optimizer with respective parameters. Adam optimizer as described in [King2014].*

Description

[King2014] Diederik Kingma, Jimmy Ba, Adam: A Method for Stochastic Optimization, http://arxiv.org/abs/1412.6980

Usage

```
mx.opt.adam(learning.rate = 0.001, beta1 = 0.9, beta2 = 0.999,
  epsilon = 1e-08, wd = 0, rescale.grad = 1, clip_gradient = NULL,
  lr_scheduler = NULL)
```

Arguments

learning.rate	float, default=0.001 Step size.
beta1	float, default=0.9 Exponential decay rate for the first moment estimates.
beta2	float, default=0.999 Exponential decay rate for the second moment estimates.
epsilon	float, default=1e-8
wd	float, default=0.0 L2 regularization coefficient add to all the weights.
rescale.grad	float, default=1.0 rescaling factor of gradient.
clip_gradient	float, optional clip gradient in range [-clip_gradient, clip_gradient].
lr_scheduler	function, optional The learning rate scheduler.

mx.opt.create — *Create an optimizer by name and parameters*

Description

Create an optimizer by name and parameters

Usage

```
mx.opt.create(name, ...)
```

Arguments

name	The name of the optimizer
...	Additional arguments

mx.opt.get.updater — *Get an updater closure that can take list of weight and gradient and return updated list of weight.*

Description

Get an updater closure that can take list of weight and gradient and return updated list of weight.

Usage

```
mx.opt.get.updater(optimizer, weights)
```

Arguments

optimizer	The optimizer
weights	The weights to be optimized

`mx.opt.rmsprop` *Create an RMSProp optimizer with respective parameters. Reference: Tieleman T, Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude[J]. COURSERA: Neural Networks for Machine Learning, 2012, 4(2). The code follows: http://arxiv.org/pdf/1308.0850v5.pdf Eq(38) - Eq(45) by Alex Graves, 2013.*

Description

Create an RMSProp optimizer with respective parameters. Reference: Tieleman T, Hinton G. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude[J]. COURSERA: Neural Networks for Machine Learning, 2012, 4(2). The code follows: http://arxiv.org/pdf/1308.0850v5.pdf Eq(38) - Eq(45) by Alex Graves, 2013.

Usage

```
mx.opt.rmsprop(learning.rate = 0.002, gamma1 = 0.95, gamma2 = 0.9,
  wd = 0, rescale.grad = 1, clip_gradient = NULL, lr_scheduler = NULL)
```

Arguments

`learning.rate`	float, default=0.002 Step size.
`gamma1`	float, default=0.95 decay factor of moving average for gradient, gradient^2.
`wd`	float, default=0.0 L2 regularization coefficient add to all the weights.
`rescale.grad`	float, default=1.0 rescaling factor of gradient.
`clip_gradient`	float, optional clip gradient in range [-clip_gradient, clip_gradient].
`lr_scheduler`	function, optional The learning rate scheduler.
`gamm2`	float, default=0.9 "momentum" factor.

`mx.opt.sgd` *Create an SGD optimizer with respective parameters. Perform SGD with momentum update*

Description

Create an SGD optimizer with respective parameters. Perform SGD with momentum update

Usage

```
mx.opt.sgd(learning.rate, momentum = 0, wd = 0, rescale.grad = 1,
  clip_gradient = NULL, lr_scheduler = NULL)
```

mx.profiler.config *Set up the configuration of profiler.*

Description

Set up the configuration of profiler.

Usage

```
mx.profiler.config(mode = MX.PROF.MODE$SYMBOLIC, filename = "profile.json")
```

Arguments

mode	Indicting whether to enable the profiler, can be 'MX.PROF.MODE$SYMBOLIC' or 'MX.PROF.MODE$ALL'. Default is 'MX.PROF.MODE$SYMBOLIC'.
filename	The name of output trace file. Default is 'profile.json'

mx.profiler.state *Set up the profiler state to record operator.*

Description

Set up the profiler state to record operator.

Usage

```
mx.profiler.state(state = MX.PROF.STATE$STOP)
```

Arguments

state	Indicting whether to run the profiler, can be 'MX.PROF.STATE$RUN' or 'MX.PROF.STATE$STOP'. Default is 'MX.PROF.STATE$STOP'.
filename	The name of output trace file. Default is 'profile.json'

`mx.rnn` *Training RNN Unrolled Model*

Description

Training RNN Unrolled Model

Usage

```
mx.rnn(train.data, eval.data = NULL, num.rnn.layer, seq.len, num.hidden,
  num.embed, num.label, batch.size, input.size, ctx = mx.ctx.default(),
  num.round = 10, update.period = 1, initializer = mx.init.uniform(0.01),
  dropout = 0, optimizer = "sgd", batch.norm = FALSE, ...)
```

Arguments

`train.data`	mx.io.DataIter or list(data=R.array, label=R.array) The Training set.
`eval.data`	mx.io.DataIter or list(data=R.array, label=R.array), optional The validation set used for validation evaluation during the progress.
`num.rnn.layer`	integer The number of the layer of rnn.
`seq.len`	integer The length of the input sequence.
`num.hidden`	integer The number of hidden nodes.
`num.embed`	integer The output dim of embedding.
`num.label`	integer The number of labels.
`batch.size`	integer The batch size used for R array training.
`input.size`	integer The input dim of one-hot encoding of embedding
`ctx`	mx.context, optional The device used to perform training.
`num.round`	integer, default=10 The number of iterations over training data to train the model.
`update.period`	integer, default=1 The number of iterations to update parameters during training period.
`initializer`	initializer object. default=mx.init.uniform(0.01) The initialization scheme for parameters.
`dropout`	float, default=0 A number in [0,1) containing the dropout ratio from the last hidden layer to the output layer.
`optimizer`	string, default="sgd" The optimization method.
`batch.norm`	boolean, default=FALSE Whether to use batch normalization.
`...`	other parameters passing to `mx.rnn`/.

Value

model A trained rnn unrolled model.

mx.rnn.forward *Using forward function to predict in rnn inference model*

Description

Using forward function to predict in rnn inference model

Usage

```
mx.rnn.forward(model, input.data, new.seq = FALSE)
```

Arguments

model	rnn model A rnn inference model
input.data,	array.matrix The input data for forward function
new.seq	boolean, default=FALSE Whether the input is the start of a new sequence

Value

result A list(prob=prob, model=model) containing the result probability of each label and the model.

mx.rnn.inference *Create a RNN Inference Model*

Description

Create a RNN Inference Model

Usage

```
mx.rnn.inference(num.rnn.layer, input.size, num.hidden, num.embed, num.label,
  batch.size = 1, arg.params, ctx = mx.cpu(), dropout = 0,
  batch.norm = FALSE)
```

Arguments

num.rnn.layer	integer The number of the layer of rnn.
input.size	integer The input dim of one-hot encoding of embedding
num.hidden	integer The number of hidden nodes.
num.embed	integer The output dim of embedding.
num.label	integer The number of labels.
batch.size	integer, default=1 The batch size used for R array training.
arg.params	list The batch size used for R array training.

`ctx`	mx.context, optional Model parameter, list of name to NDArray of net's weights.
`dropout`	float, default=0 A number in [0,1) containing the dropout ratio from the last hidden layer to the output layer.
`batch.norm`	boolean, default=FALSE Whether to use batch normalization.

Value

model list(rnn.exec=integer, symbol=mxnet symbol, num.rnn.layer=integer, num.hidden=integer, seq.len=integer, batch.size=integer, num.embed=integer) A rnn inference model.

`mx.rnorm` *Generate nomal distribution with mean and sd.*

Description

Generate nomal distribution with mean and sd.

Usage

```
mx.rnorm(shape, mean = 0, sd = 1, ctx = NULL)
```

Arguments

`shape`	Dimension, The shape(dimension) of the result.
`mean`	numeric, The mean of distribution.
`sd`	numeric, The standard deviations.
`ctx,`	optional The context device of the array. mx.ctx.default() will be used in default.

Examples

```
mx.set.seed(0)
as.array(mx.runif(2))
# 0.5488135 0.5928446
mx.set.seed(0)
as.array(mx.rnorm(2))
# 2.212206 1.163079
```

mx.runif — *Generate uniform distribution in [low, high) with specified shape.*

Description

Generate uniform distribution in [low, high) with specified shape.

Usage

```
mx.runif(shape, min = 0, max = 1, ctx = NULL)
```

Arguments

shape	Dimension, The shape(dimension) of the result.
min	numeric, The lower bound of distribution.
max	numeric, The upper bound of distribution.
ctx,	optional The context device of the array. mx.ctx.default() will be used in default.

Examples

```
mx.set.seed(0)
as.array(mx.runif(2))
# 0.5488135 0.5928446
mx.set.seed(0)
as.array(mx.rnorm(2))
# 2.212206 1.163079
```

mx.set.seed — *Set the seed used by mxnet device-specific random number generators.*

Description

Set the seed used by mxnet device-specific random number generators.

Usage

```
mx.set.seed(seed)
```

Arguments

seed	the seed value to the device random number generators.

Details

We have a specific reason why `mx.set.seed` is introduced, instead of simply use `set.seed`.

The reason that is that most of mxnet random number generator can run on different devices, such as GPU. We need to use massively parallel PRNG on GPU to get fast random number generations. It can also be quite costly to seed these PRNGs. So we introduced `mx.set.seed` for mxnet specific device random numbers.

Examples

```
mx.set.seed(0)
as.array(mx.runif(2))
# 0.5488135 0.5928446
mx.set.seed(0)
as.array(mx.rnorm(2))
# 2.212206 1.163079
```

---

mx.simple.bind — *Simple bind the symbol to executor, with information from input shapes.*

Description

Simple bind the symbol to executor, with information from input shapes.

Usage

```
mx.simple.bind(symbol, ctx, grad.req = "null", ...)
```

---

mx.symbol.Concat — *Perform an feature concat on channel dim (dim 1) over all the inputs.*

Description

Perform an feature concat on channel dim (dim 1) over all the inputs.

Usage

```
mx.symbol.Concat(data, num.args, dim = NULL, name = NULL)
```

Arguments

`data`	list, required List of tensors to concatenate
`num.args`	int, required Number of inputs to be concated.
`dim`	int, optional, default='1' the dimension to be concated.
`name`	string, optional Name of the resulting symbol.

Value

out The result mx.symbol

---

mx.symbol.Group *Create a symbol that groups symbols together.*

Description

Create a symbol that groups symbols together.

Usage

```
mx.symbol.Group(...)
```

Arguments

kwarg	Variable length of symbols or list of symbol.

Value

The result symbol

---

mx.symbol.infer.shape *Inference the shape of arguments, outputs, and auxiliary states.*

Description

Inference the shape of arguments, outputs, and auxiliary states.

Usage

```
mx.symbol.infer.shape(symbol, ...)
```

Arguments

symbol	The `mx.symbol` object

mx.symbol.load *Load an mx.symbol object*

Description

Load an mx.symbol object

Usage

```
mx.symbol.load(filename)
```

Arguments

filename	the filename (including the path)

Examples

```
data = mx.symbol.Variable('data')
mx.symbol.save(data, 'temp.symbol')
data2 = mx.symbol.load('temp.symbol')
```

mx.symbol.load.json *Load an mx.symbol object from a json string*

Description

Load an mx.symbol object from a json string

Arguments

str	the json str represent a mx.symbol

mx.symbol.save — *Save an mx.symbol object*

Description

Save an mx.symbol object

Usage

```
mx.symbol.save(symbol, filename)
```

Arguments

symbol	the mx.symbol object
filename	the filename (including the path)

Examples

```
data = mx.symbol.Variable('data')
mx.symbol.save(data, 'temp.symbol')
data2 = mx.symbol.load('temp.symbol')
```

mx.symbol.Variable — *Create a symbolic variable with specified name.*

Description

Create a symbolic variable with specified name.

Arguments

name	string The name of the result symbol.

Value

The result symbol

mxnet — *MXNet: Flexible and Efficient GPU computing and Deep Learning.*

Description

MXNet is a flexible and efficient GPU computing and deep learning framework.

Details

It enables you to write seamless tensor/matrix computation with multiple GPUs in R.

It also enables you construct and customize the state-of-art deep learning models in R, and apply them to tasks such as image classification and data science challenges.

mxnet.export — *Internal function to generate mxnet_generated.R Users do not need to call this function.*

Description

Internal function to generate mxnet_generated.R Users do not need to call this function.

Usage

```
mxnet.export(path)
```

Arguments

path	The path to the root of the package.

Ops.MXNDArray — *Binary operator overloading of mx.ndarray*

Description

Binary operator overloading of mx.ndarray

Usage

```
## S3 method for class 'MXNDArray'
Ops(e1, e2)
```

Arguments

e1	The first operand
e1	The second operand

outputs — *Get the outputs of a symbol.*

Description

Get the outputs of a symbol.

Usage

```
outputs(x)
```

Arguments

x	The input symbol

predict.MXFeedForwardModel — *Predict the outputs given a model and dataset.*

Description

Predict the outputs given a model and dataset.

Usage

```
## S3 method for class 'MXFeedForwardModel'
predict(model, X, ctx = NULL,
  array.batch.size = 128, array.layout = "auto")
```

Arguments

model	The MXNet Model.
X	The dataset to predict.
ctx	mx.cpu() or mx.gpu(i) The device used to generate the prediction.
array.batch.size	The batch size used in batching. Only used when X is R's array.
array.layout	can be "auto", "colmajor", "rowmajor", (detault=auto) The layout of array. "rowmajor" is only supported for two dimensional array. For matrix, "rowmajor" means dim(X) = c(nexample, nfeatures), "colmajor" means dim(X) = c(nfeatures, nexample) "auto" will auto detect the layout by match the feature size, and will report error when X is a square matrix to ask user to explicitly specify layout.

`print.MXNDArray` *print operator overload of mx.ndarray*

Description

print operator overload of mx.ndarray

Usage

```
## S3 method for class 'MXNDArray'
print(nd)
```

Arguments

`nd`	The mx.ndarray

Index

∗Topic **datasets**
mx.metric.accuracy, 22
mx.metric.mae, 23
mx.metric.mse, 23
mx.metric.rmse, 24
mx.metric.rmsle, 24
mx.metric.top_k_accuracy, 24

arguments, 4
as.array.MXNDArray, 4
as.matrix.MXNDArray, 4

ctx, 5

dim.MXNDArray, 5

graph.viz, 5

is.mx.context, 6
is.mx.dataiter, 6
is.mx.ndarray, 7
is.mx.symbol, 7
is.num.in.vect, 7

length.MXNDArray, 8

mx.apply, 8
mx.callback.early.stop, 9
mx.callback.log.speedometer, 9
mx.callback.log.train.metric, 10
mx.callback.save.checkpoint, 10
mx.cpu, 11
mx.ctx.default, 11
mx.exec.backward, 11
mx.exec.forward, 12
mx.exec.update.arg.arrays, 12
mx.exec.update.aux.arrays, 12
mx.exec.update.grad.arrays, 13
mx.gpu, 13
mx.gru, 13
mx.gru.forward, 14
mx.gru.inference, 15
mx.init.create, 16
mx.init.internal.default, 16
mx.init.normal, 17
mx.init.uniform, 17
mx.init.Xavier, 17
mx.io.arrayiter, 18
mx.io.extract, 18
mx.kv.create, 19
mx.lr_scheduler.FactorScheduler, 19
mx.lr_scheduler.MultiFactorScheduler, 20
mx.lstm, 20
mx.lstm.forward, 21
mx.lstm.inference, 22
mx.metric.accuracy, 22
mx.metric.custom, 23
mx.metric.mae, 23
mx.metric.mse, 23
mx.metric.rmse, 24
mx.metric.rmsle, 24
mx.metric.top_k_accuracy, 24
mx.mlp, 25
mx.model.FeedForward.create, 26
mx.model.load, 27
mx.model.save, 27
mx.nd.array, 28
mx.nd.copyto, 28
mx.nd.load, 29
mx.nd.ones, 29
mx.nd.save, 30
mx.nd.zeros, 30
mx.opt.adadelta, 31
mx.opt.adagrad, 31
mx.opt.adam, 32
mx.opt.create, 33
mx.opt.get.updater, 33
mx.opt.rmsprop, 34
mx.opt.sgd, 34

mx.profiler.config, 35
mx.profiler.state, 35
mx.rnn, 36
mx.rnn.forward, 37
mx.rnn.inference, 37
mx.rnorm, 38
mx.runif, 39
mx.set.seed, 39
mx.simple.bind, 40
mx.symbol.Concat, 40
mx.symbol.Group, 41
mx.symbol.infer.shape, 41
mx.symbol.load, 42
mx.symbol.load.json, 42
mx.symbol.save, 43
mx.symbol.Variable, 43
mxnet, 44
mxnet-package *(mxnet)*, 44
mxnet.export, 44

Ops.MXNDArray, 44
outputs, 45

predict.MXFeedForwardModel, 45
print.MXNDArray, 46