Cosmos Documentation

Release 1.0

Marcel van Gerven

May 13, 2017

CONTENTS

Contents:

CHAPTER

ONE

SUPERVISED_LEARNING PACKAGE

1.1 Submodules

1.2 supervised_learning.agents module

class `supervised_learning.agents.`**`SupervisedAgent`**(*model*, *optimizer*, *cutoff=None*)

Bases: `object`

Agent which trains on labelled data

`__call__`(*data*)

Runs networks in forward mode and applies optional output function

Parameters **`data`** –

Returns post-processed output

`reset_state`()

Resets persistent states

`test`(*data*)

Returns the loss for one batch

Parameters **`data`** –

Returns loss

`train`(*data*)

Train agent on one batch :param data: :return: loss

1.3 supervised_learning.iterators module

class `supervised_learning.iterators.`**`RandomIterator`**(*data*, *batch_size=None*)

Bases: `object`

Generates random subsets of data

`next`()

class `supervised_learning.iterators.`**`SequentialIterator`**(*data*, *batch_size=None*)

Bases: `object`

Generates subsets of data such that each batch contains data for the next time point

`next`()

1.4 supervised_learning.models module

class `supervised_learning.models.`**`Classifier`**(*net*, *gpu=-1*)

Bases: *`supervised_learning.models.Model`*

Wrapper for classification problems

class `supervised_learning.models.`**`Model`**(*net*, *loss_function*, *output_function=<function <lambda>>*, *gpu=-1*)

Bases: `chainer.link.Chain`

Model which wraps a network to compute loss and generate actual predictions

`__call__`(*data*)

Compute loss for minibatch of data

Parameters **`data`** – list of minibatches (e.g. inputs and targets for supervised learning)

Returns loss

`has_state`

Checks if a network has persistent states

Returns bool

`predict`(*data*)

Returns prediction, which can be different than raw output (e.g. for softmax function)

Parameters **`data`** – minibatch or list of minibatches representing input to the model

Returns prediction

`reset_state`()

class `supervised_learning.models.`**`Regressor`**(*net*, *gpu=-1*)

Bases: *`supervised_learning.models.Model`*

Wrapper for regression problems

1.5 supervised_learning.networks module

class `supervised_learning.networks.`**`MLP`**(*n_input=None*, *n_output=1*, *n_hidden=10*)

Bases: `chainer.link.Chain`

Multilayer perceptron

`has_state`

Checks if a network has persistent states

Returns bool

class `supervised_learning.networks.`**`RNN`**(*n_input=None*, *n_output=1*, *n_hidden=10*)

Bases: `chainer.link.Chain`

`has_state`

Checks if a network has persistent states

Returns bool

`reset_state`()

Resets persistent states

1.6 supervised_learning.unit_test module

class `supervised_learning.unit_test.`**`UnitTest`**(*methodName='runTest'*)

Bases: `unittest.case.TestCase`

`test_gpu`()

Test training procedure for stateless network on GPU

`test_stateful_network`()

Test training procedure for stateful network

`test_stateless_network`()

Test training procedure for stateless network

1.7 supervised_learning.world module

class `supervised_learning.world.`**`World`**(*agents*, *out='result'*)

Bases: `object`

Wrapper object which takes care of training and testing on some data iterator for one or more agents

`test`(*test_iter*)

Parameters **`test_iter`** – iterator over the test data

Returns test loss

`train`(*train_iter*, *n_epochs*, *test_iter=None*, *snapshot=0*)

Parameters

- **`train_iter`** – iterator over the training data
- **`n_epochs`** (*int*) – number of epochs to train on
- **`test_iter`** – optional iterator over the test data (returns optimal model)
- **`snapshot`** (*int*) – whether or not to save model after each epochs modulo snapshot

Returns train loss and optional test loss

1.8 Module contents

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX