RBM-MCMC Documentation

Release 0.0.1

refraction-ray

Aug 02, 2018

Contents:

CHAPTER 1

RBM-MCMC

1.1 mcmc module

1.2 rbm module

The basic module for Restricted Boltzmann machine(RBM) and its generalizations.

class rbm.**FBM**(*sites*)

Bases: object

Class designed as a special limit of general Boltzmann machine, where only visible layer exists.

Parameters **sites** – a list giving the size information on the model

energy(*state*)

Calculate the effective energy of the system.

Parameters **state** – array of binary configurations of 1d shape or system shape

Returns real value of energy of the model

fit(*traindatas*, *testdatas*, *batch=20*, *epoch=10*, *learningrate=0.01*, *regulation1=0*, *regulation2=0*, *mc-steps=100*, *presteps=10000*, *debuglog=True*)

Train the model and optimize paramters with traindata given.

Parameters

- **traindatas** – the array of arrays of data for training
- **testdatas** – the array of arrays of data for testing and evaluation
- **batch** – integer for the size of batch for SGD
- **epoch** – integer for the numbers of epochs of training
- **learningrate** – real value for the update rate
- **regulation1** – L1 regularization term

- **regulation2** – L2 regularization term
- **mcsteps** – integer value for steps of Monte Carlo update for each batch
- **presteps** – integer value for steps of Monte Carlo before training
- **debuglog** – boolean, true for information print after each epoch

mask(*updateweights*)
Mask the weight matrix.

Parameters **updateweights** – array of weights shape

Returns array of weights shape transformed from the input

maskh(*updatebias*)
Mask the bias vector.

Parameters **updatebias** – 1d array of bias shape

Returns 1d array of bias shape after transformation

mcupdate(*states*, *steps=1*)
Do Monte Carlo update on the model based on present weights.

Parameters

- **states** – array of state arrays
- **steps** – integer for steps of MC updates

Returns array of state arrays after updates

Note the input states should be changed after the function. Besides, this function is implemented in non-parallel fashion and is deprecated: not used in self.fit() function.

paramcupdate(*states*, *p*, *steps=1*)
Monte Carlo update of the given group of states implemented in parallel fashion.

Parameters

- **states** – array of arrays of states (in 1d or model shape)
- **p** – the Pool object from multiprocess module
- **steps** – integer value for the steps of update

Returns array of arrays of states after updates

updatecore(*state*, *steps=1*)
Monte Carlo update scheme for specific state.

Parameters

- **state** – array of one state of the system in 1D or system shape
- **steps** – integer for steps of updates of the state

Returns array of the state after updates

class rbm.**NNNFBM**(*sites*)
Bases: *rbm.uniformFBM*

Isotropic model with NN and NNN couplings as well as external field.

getNNNmask()
Get the mask matrix for the NNN coupling part.

getNNmask()
Get the mask matrix for the NN coupling part.

mask(*updateweights*)
Mask the weight matrix.

Parameters **updateweights** – array of weights shape

Returns array of weights shape transformed from the input

class rbm.**RBM**(*visible*, *hidden*)
Bases: object

RBM class

Parameters

- **visible** – a list with the visble layer size, eg. [28,28] for MNIST data
- **hidden** – a list with the hidden layer size

Some trainning details are inspired by : https://www.cs.toronto.edu/~hinton/absps/guideTR.pdf.

Gibbsupdate(*visibledatas*, *nosteps=1*)
Gibbs update for the model: start from visible layer

Parameters

- **visibledatas** – array of configuration of visible layer (both 1d and visible layer shape are ok)
- **nosteps** – integer for the Gibbs update steps

Returns list of two arrays, the first is configuration of visible layer and the second is for hidden layer

Note one step is v->h->v, so the hidden layer configurations is half step before visble ones.

cdk(*visibledatas*, *nosteps=1*)
Modified Gibbs update used for CD-k training.

Parameters

- **visibledatas** – array of configuration of visible layer (both 1d and visible layer shape are ok)
- **nosteps** – integer for the Gibbs update steps or the k in CD-k

Returns list of two arrays, the first is configuration of visible layer and the second is for hidden layer

Note the difference between cdk update and Gibbs update. In the last step, the visible data are given by probability intead of states and then we use the probability to calculate probability of hidden layer as data for hidden layer which is half step later compared to visibledata.

energy(*visibledata*, *hiddendata*)
Calculate the energy of the model given configuration of both layers.

Parameters

- **visibledata** – array of configuration of visible layer (both 1d and visible layer shape are ok)
- **hiddendata** – array of configuration of hidden layer (both 1d and hidden layer shape are ok)

Returns real value of the energy

error(*visibledatas*)
Calculate the reconstruction error of specified visible data after one Gibbs update.

Parameters **visibledatas** – array of configuration of visible layer (both 1d and visible layer shape are ok)

fit(*visibledatas*, *testdatas*, *batch=20*, *epoch=50*, *learningrate=0.05*, *regulation1=0*, *regulation2=0*, *cdkstep=1*, *debuglog=True*)
Fit the RBM.

Parameters

- **visibledatas** – the array of arrays of datas for training, eg. the shape [60000,28,28] for training
- **testdatas** – the array of arrays of datas for testing, eg. the shape [60000,20,20] for training
- **batch** – integer for the size of batch for SGD
- **epoch** – integer for the numbers of epochs of training
- **learningrate** – real value for the update rate
- **regulation1** – L1 regularization term
- **regulation2** – L2 regularization term
- **cdkstep** – integer value of k in CD-k training
- **debuglog** – boolean, true for information print after each epoch

freeenergy(*visibledata*)
Caculate the free energy of the model given visible data.

Parameters **visibledata** – array of configuration of visible layer (both 1d and visible layer shape are ok)

Returns real value of the free energy of the visible configuration

getbias()
Get bias of the model in the shape of visible and hidden layer.

Returns list of two array, the first one is bias on visible layer while the second array is bias on hidden layer

getweights()
Get weights of the model in the shape of visible and hidden layer.

Returns weights array, eg. the shape is (28,28,10,5) for RBM([28,28],[10,5]).

mask(*updateweights*)
Mask the updateweights in some pattern.

Parameters **updateweights** – array in the shape of weights

Returns array in the shape of weights after some processing

probabilityonhidden(*hiddendatas*)
Get the conditional activation probability of visible layer based on given configuration of hidden layer.

Parameters **hiddendatas** – array of hidden configuration arrays (both in 1d or in the shape of hidden layers are ok)

Returns array of probability 1d arrays with the number of elements the same as visible neurons

probabilityonvisible(*visibledatas*)
Get the conditional activation probability of hidden layer based on given configuration of visible layer.

Parameters **visibledatas** – array of visible configuration arrays (both in 1d or in the shape of visible layers are ok)

Returns array of probability 1d arrays with the number of elements the same as hidden neurons

randomvisible(*no=1*, *aim='D'*)
Provide random samples whose shape consistent with the model.

Parameters

- **no** – integer for the number of samples one want to generate
- **aim** – string, 'D' for configuration generation while 'P' for probability generation

Returns array of arrays of configuration or probability with the shape of visible layer

sampleonhidden(*hiddendatas*)
Get one sample configuration of visible layer based on given configuration of vhidden layer.

Parameters **visibledatas** – array of hidden configuration arrays (both in 1d or in the shape of hidden layers are ok)

Returns array of 1d configuration arrays with the number of elements the same as visible neurons

sampleonvisible(*visibledatas*)
Get one sample configuration of hidden layer based on given configuration of visible layer.

Parameters **visibledatas** – array of visible configuration arrays (both in 1d or in the shape of visible layers are ok)

Returns array of 1d configuration arrays with the number of elements the same as hidden neurons

summary()
Print the basic information on this RBM.

rbm.**chunks**(*l*, *n*)
Generator for iteration which divides list l into n batches randomly.

Parameters

- **l** – the list to be divided
- **n** – the integer for the number of elements in one batch

Yields list of batch size

Note the last batch may be smaller if 1%n != 0.

class rbm.**localFBM**(*sites*, *feedmask=array([1.])*)
Bases: *rbm.FBM*

Spin model with masked couplings, only certain weights are allowed.

Parameters **feedmask** – (optional) the mask matrix of weights shape

getmask()
Get the default mask matrix: NN Ising couplings.

Returns arrays of the weights matrix shape with only NN elements 1 and others zero

mask(*updateweights*)
Mask the weight matrix.

Parameters **updateweights** – array of weights shape

Returns array of weights shape transformed from the input

class rbm.**localRBM**(*visible*, *window*, *stride*)
Bases: *rbm.RBM*

RBM with locality, where only the weights within windows are nonzero for each hidden layer neuron.

Parameters

- **visible** – a list with the visble layer size, eg. [28,28] for MNIST data
- **window** – a list for the size of window, eg. [2,2]
- **stride** – a list for the size of stride, eg. [2,2]

getmask()
Get the mask matrix with zero elements in required vanishing weights links.

Returns the array of the shape of weights with zero and one

mask(*updateweights*)
Mask the updateweights in some pattern.

Parameters **updateweights** – array in the shape of weights

Returns array in the shape of weights after some processing

rbm.**reshapeinput**(*inputarray*)
Reshape the inputarray to 1D.

Parameters **inputarray** – array with any shape

Returns array with 1D shape

rbm.**sampleinput**(*arrays*)
Sample the arrays elementwise.

Parameters **arrays** – the input array whose elements are between 0 to 1 as probabilities

Returns the array with the same shape as the input with all elements whose values are 1 or 0

class rbm.**uniformFBM**(*sites*)
Bases: *rbm.localFBM*

Isotropic model with equal weights and bias within nonzero coupling windows.

mask(*updateweights*)
Mask the weight matrix.

Parameters **updateweights** – array of weights shape

Returns array of weights shape transformed from the input

maskh(*updatebias*)
Mask the bias vector.

Parameters **updatebias** – 1d array of bias shape

Returns 1d array of bias shape after transformation

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Python Module Index

r

Index