Readme for Regression by Latent Tensor Reconstruction

March 15, 2021

1 Demonstration environment

This demonstration contains two files:

ltr_tensor_solver_actxu_v_cls_010.py Python module implementing the Regression by Latent Tensor Reconstruction(LTR).

ltr_main_generated_data_ext.py Python script implementing a test environment for LTR by generating random multivariate polynomials, and running the training and test, and provides detailed statistics and a summary graph.

The demonstration can be started by running the main script:

```
python3 ltr_main_generated_data_ext.py
```

The demonstration code assumes python version ≥ 3.5, and the following standard **python** modules:

```
sys, time, pickle,
```

and additionally it also requires

```
numpy, matplotlib.
```

The demonstration main file "ltr_main_generated_data_ext.py" also provides the an implementation of the interface to the learning module "ltr_tensor_matrix_ext_cls.py", see the next sections.

2 Interface of LTR

Creating the learning object sets the parameters of the polynomial(tensor), assuming the learning module is

```
import ltr_tensor_solver_actxu_v_cls_010 as tensor_cls
```

Construct solver object `cmodel=tensor_cls.tensor_latent_vector_cls(norder=norder,ran`

where

norder is the degree(order) of the polynomial(tensor)

rank is the rank of the tensor

rankuv is the internal rank, the common dimesnsion of U and V parameters, the rank of the parameter decomposition.

Setting parameters Parameters for the optimization procedure can be set by this method

```
cmodel.update_parameters(nsigma = nsigma, \
                         mblock = mblock, \
                         sigma0 = sigma, \
                         gamma = gamma, \
                         gammanag = gammanag, \
                         gammanag2 = gammanag2, \
                         cregular = cregular, \
                         sigmamax = nsigmamax}
```

where

```
## initial learning speed
sigma=0.01

## regularization
cregular=0.05    ## regularization weight of lambdas in the objective function

## mini-batch parameters
mblock=500             ## mini-batch size

## Parameters for the optimization
## !!! They might not need to change !!!
nsigma=1              ## learning speed correction interval
gamma=0.9999999       ## discount factor of gradient update
gammanag=0.99        ## discount for the ADAM method, scalling gradient
```

```
gammanag2=0.99       ## discount for the ADAM method, scalling norm
nsigmamax=1           ## maximum gradient length
                      ## relative to the ADAM aggregated gradient
```

Other parameters, see the source of **ltr_tensor_matrix_ext_cls.py**, can be changed in a similar way by including them into the **cmodel.update_parameters**.

Training The training can be carried out by this method

```
cmodel.fit(Xtrain,ytrain,nepoch=nrepeat)
```

where

Xtrain the training part of the input data matrix(2D array)

ytrain the training part of the output data matrix(2D array), if the output is scalar valued then it could be a vector(1D array)

nepoch=nrepeat sets the number of epochs, the default is 10.

Prediction The prediction method can be called by

```
test_prediction=cmodel.predict(Xtest)
```

where

Xtest is the test part of the input data matrix(2D array)

test_prediction contains the predicted values, it is 2D array. In case of scalar prediction the second dimension is 1.