Tim/VAE_sym_TD

```
%sh

echo $PWD
rm TXLoader.py*
wget --no-check-certificate --no-cache --no-cookies https://raw.githubusercontent.com/tianle91
ls -l --block-size=M
```

FINISHED

```
/home/hadoop
--2019-03-01 22:35:19--  https://raw.githubusercontent.com/tianle91/forecastor/master/TXLoade
r.py
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.248.133
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101.248.133|:443... co
nnected.
HTTP request sent, awaiting response... 200 OK
Length: 3820 (3.7K) [text/plain]
Saving to: ‘TXLoader.py’
     0K ...                                                   100% 69.7M=0s
2019-03-01 22:35:19 (69.7 MB/s) - ‘TXLoader.py’ saved [3820/3820]
total 2M
-rw-r--r-- 1 root   root   1M Mar  1 20:53 ae_1mo-1h_SYM:TD.h5
-rw-r--r-- 1 root   root   1M Mar  1 20:27 ae.h5
-rwxrwxr-x 1 hadoop hadoop 1M Mar  1 03:50 attach.sh
-rw-r--r-- 1 root   root   1M Mar  1 20:50 Book.py
-rw-rw-r-- 1 hadoop hadoop 1M Mar  1 03:50 complete.out
drwxr-xr-x 2 root   root   1M Mar  1 22:35 data
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:19 PM.

```
%sh

cd data
ls -ln --block-size=M
```

FINISHED

```
total 2M
-rw-r--r-- 1 0 0 1M Mar  1 20:53 1mo-1h_SYM:BMO_dates.pickle
-rw-r--r-- 1 0 0 1M Mar  1 20:59 1mo-1h_SYM:BMO_dt:2018-04-02 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 22:34 1mo-1h_SYM:BMO_dt:2018-04-02 00:00:00_trades.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:04 1mo-1h_SYM:BMO_dt:2018-04-03 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 22:35 1mo-1h_SYM:BMO_dt:2018-04-03 00:00:00_trades.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:10 1mo-1h_SYM:BMO_dt:2018-04-04 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 22:35 1mo-1h_SYM:BMO_dt:2018-04-04 00:00:00_trades.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:14 1mo-1h_SYM:BMO_dt:2018-04-05 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:20 1mo-1h_SYM:BMO_dt:2018-04-06 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:25 1mo-1h_SYM:BMO_dt:2018-04-09 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:29 1mo-1h_SYM:BMO_dt:2018-04-10 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:34 1mo-1h_SYM:BMO_dt:2018-04-11 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:39 1mo-1h_SYM:BMO_dt:2018-04-12 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:43 1mo-1h_SYM:BMO_dt:2018-04-13 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:47 1mo-1h_SYM:BMO_dt:2018-04-16 00:00:00_orders.pickle.gz
-rw-r--r-- 1 0 0 1M Mar  1 21:51 1mo-1h_SYM:BMO_dt:2018-04-17 00:00:00_orders.pickle.gz
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:19 PM.

FINISHED

```
%python

import os
import sys
import gzip
import pickle

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import preprocessing

#from TXLoader import TXLoader
exec(open(os.getcwd() + '/TXLoader.py').read())
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:20 PM.

FINISHED

```
%python

symbol = 'TD'
jobname = '1mo-1h'
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:20 PM.

FINISHED

```
%python

xm = TXLoader(jobname=jobname, symbol=symbol).getxm(byday=False)
ntime, ncov = xm.shape
print ('xm.shape:', xm.shape)
print (xm[0, ...])
```

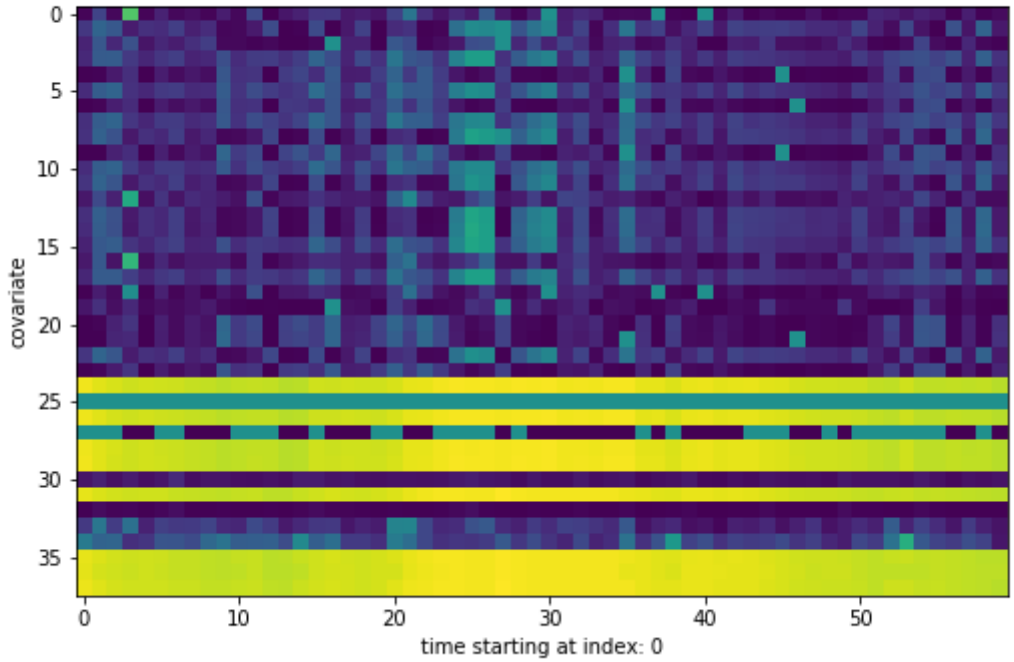
```
xm.shape: (1320, 38)
[400.0 258.0 10.0 57500.0 1800.0 229.0 2000.0 268.0 55700.0 10.0 509.0
 219.0 12.0 103500.0 45800.0 47800.0 2400.0 107700.0 2.0 2000.0 22.0 10.0
 487.0 4200.0 72.69 None 72.67 0.01999999999999602 72.67923076923077 72.68
 168.30107526881721 72.632043011 111.45527324274002 93.0
 0.02261728560699821 72.632563 72.6 72.68]
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:20 PM.

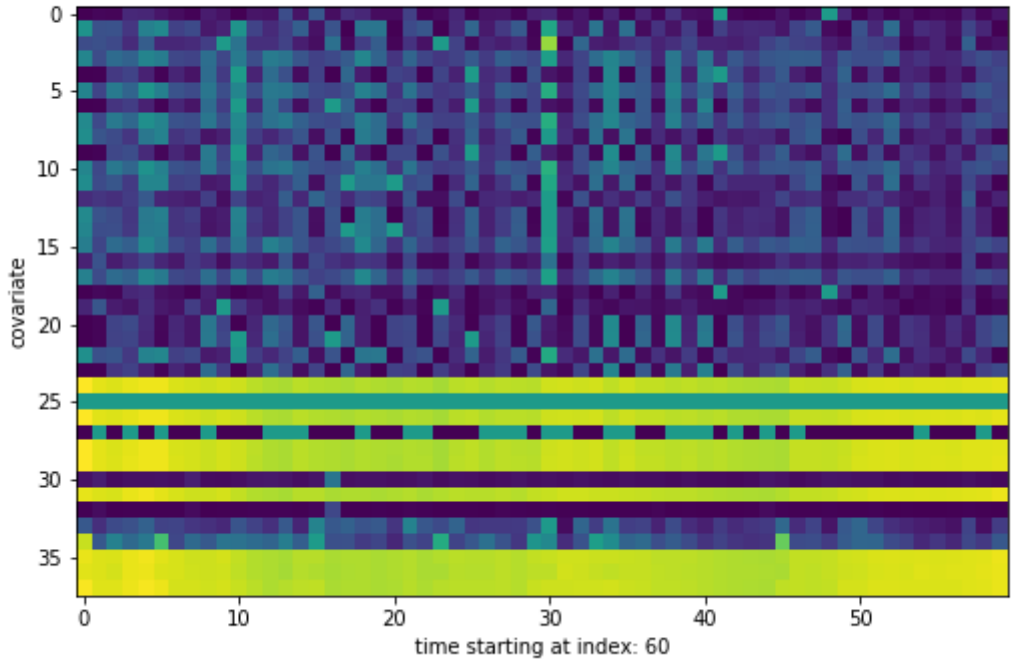
FINISHED

```
%python

scaler = preprocessing.MinMaxScaler((-1, 1))
scaler.fit(xm)

xmnormd = scaler.transform(xm)
xmnormd[np.isnan(xmnormd)] = 0

limmultiple = 5
for i in range(0, min(limmultiple*60, xm.shape[0]), 60):
    plt.imshow(np.transpose(xmnormd[i:(i+60), :]))
    plt.xlabel('time starting at index: %s' % (i))
    plt.ylabel('covariate')
    plt.show()
```

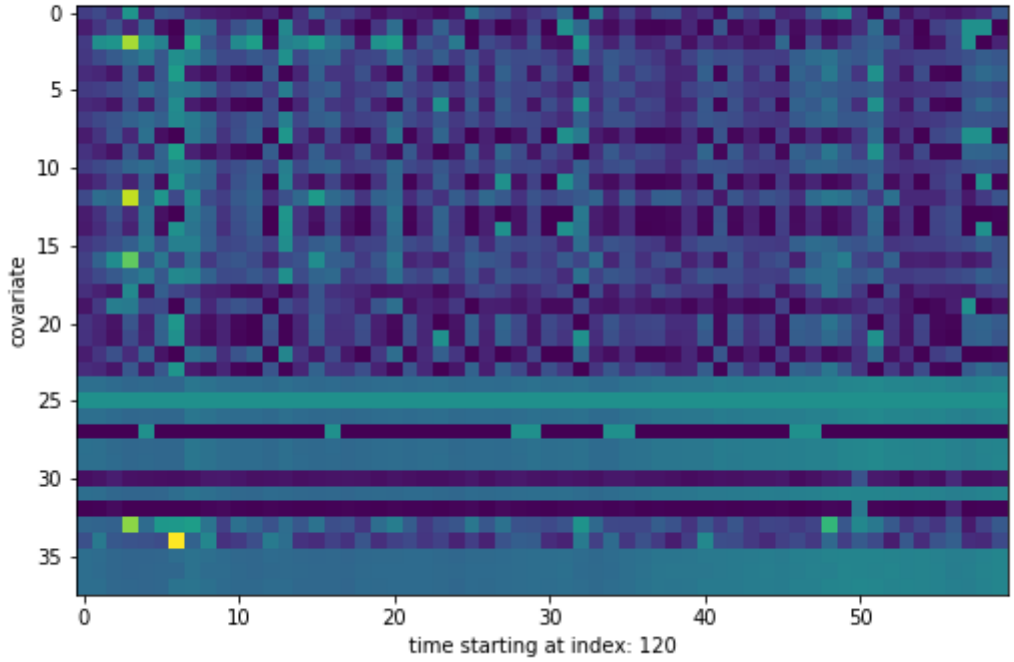
```
<matplotlib.image.AxesImage object at 0x7faf8d239400>
Text(0.5,29.75,'time starting at index: 0')
Text(51.25,0.5,'covariate')
```



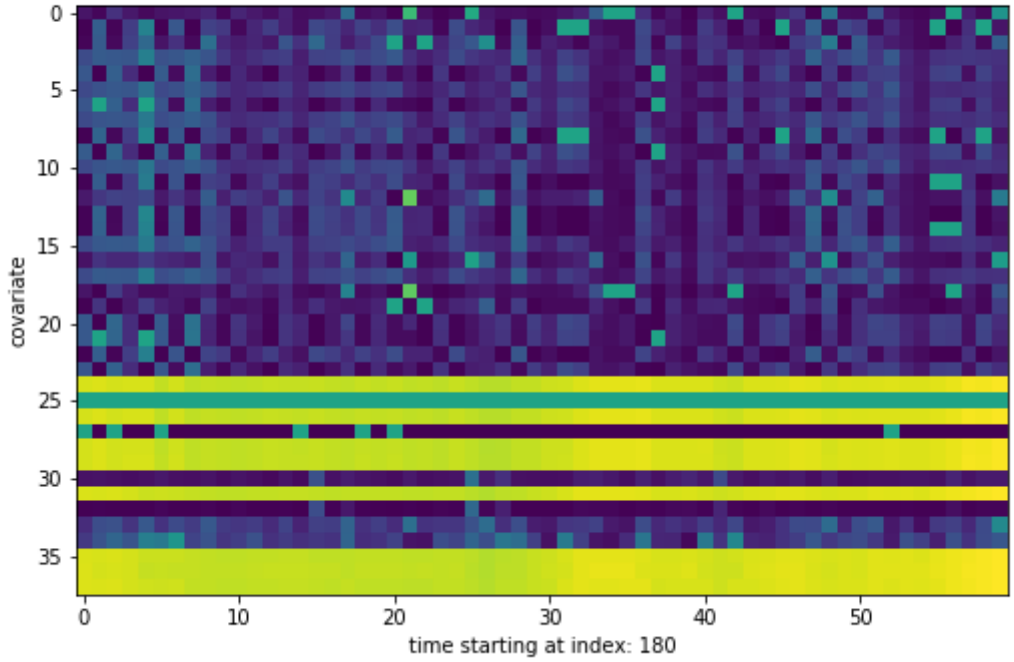
Text(0.5,29.75,'time starting at index: 60') Text(51.25,0.5,'covariate')



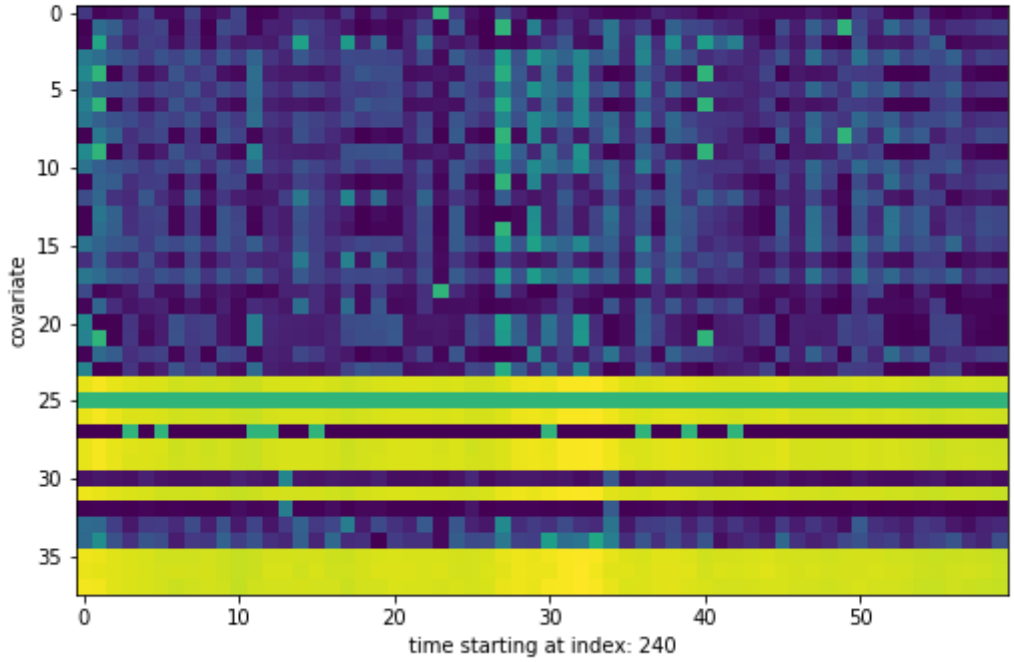
Text(0.5,29.75,'time starting at index: 120') Text(51.25,0.5,'covariate')



Text(0.5,29.75,'time starting at index: 180') Text(51.25,0.5,'covariate')



Text(0.5,29.75,'time starting at index: 240') Text(51.25,0.5,'covariate')



Took 4 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:24 PM.

VAE from here: FINISHED

https://github.com/keras-team/keras/blob/master/examples/variational_autoencoder.py (https://github.com/keras-team/keras/blob/master/examples/variational_autoencoder.py)

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:20 PM.

```
%python

from tensorflow.keras.layers import Lambda, Input, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.losses import mse
from tensorflow.keras import backend as K
from tensorflow.keras.optimizers import Adam
```

FINISHED

Took 4 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:24 PM.

```
%python

from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
```

FINISHED

```
[name: "/device:CPU:0"
device_type: "CPU"
memory_limit: 268435456
locality {
}
incarnation: 2910643900903435296
, name: "/device:XLA_CPU:0"
device_type: "XLA_CPU"
```

```
memory_limit: 17179869184
locality {
}
incarnation: 17148552153544613968
physical_device_desc: "device: XLA_CPU device"
]
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:24 PM.

```
%python

input_shape = (ncov, )
interm_dim = 16
latent_dim = 8
```

FINISHED

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:24 PM.

```
%python

# VAE model = encoder + decoder
# build encoder model
inputs = Input(shape=input_shape, name='encoder_input')
x = Dense(interm_dim, activation='tanh')(inputs)
z_mean = Dense(latent_dim, name='z_mean')(x)
z_log_var = Dense(latent_dim, name='z_log_var')(x)

# use reparameterization trick to push the sampling out as input
# note that "output_shape" isn't necessary with the TensorFlow backend
def sampling(args):
    z_mean, z_log_var = args
    batch = K.shape(z_mean)[0]
    dim = K.int_shape(z_mean)[1]
    # by default, random_normal has mean = 0 and std = 1.0
    epsilon = K.random_normal(shape=(batch, dim))
    return z_mean + K.exp(0.5 * z_log_var) * epsilon

z = Lambda(sampling, output_shape=(latent_dim,), name='z')([z_mean, z_log_var])


# instantiate encoder model
encoder = Model(inputs, [z_mean, z_log_var, z], name='encoder')
encoder.summary()
```

FINISHED

```
_________________________________________________________________________________________________
____
Layer (type)                 Output Shape              Param #     Connected to
=================================================================================================
====
encoder_input (InputLayer)   (None, 38)           0
_________________________________________________________________________________________________
____
dense_12 (Dense)             (None, 16)           624         encoder_input[0][0]
_________________________________________________________________________________________________
____
z_mean (Dense)               (None, 8)            136         dense_12[0][0]
_________________________________________________________________________________________________
____
z_log_var (Dense)            (None, 8)            136         dense_12[0][0]
```

```
_________________________________________________________________
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:24 PM.

```
%python

# build decoder model
latent_inputs = Input(shape=(latent_dim,), name='z_sampling')
x = Dense(interm_dim, activation='tanh')(latent_inputs)
outputs = Dense(ncov, activation='tanh')(x)

# instantiate decoder model
decoder = Model(latent_inputs, outputs, name='decoder')
decoder.summary()
```

FINISHED

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
z_sampling (InputLayer)      (None, 8)                 0         
_________________________________________________________________
dense_13 (Dense)             (None, 16)                144       
_________________________________________________________________
dense_14 (Dense)             (None, 38)                646       
=================================================================
Total params: 790
Trainable params: 790
Non-trainable params: 0
_________________________________________________________________
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:24 PM.

```
%python

# instantiate VAE model
# zsampled -> x
outputs = decoder(encoder(inputs)[2])
vae = Model(inputs, outputs, name='vae')
```

FINISHED

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:24 PM.

```
%python

# VAE loss = mse_loss or xent_loss + kl_loss
# losses defined in terms of [inputs, outputs], which point to keras layers
reconstruction_loss = ncov*mse(inputs, outputs)
kl_loss = 1 + z_log_var - K.square(z_mean) - K.exp(z_log_var)
kl_loss = K.sum(kl_loss, axis=-1)
kl_loss *= -0.5

vae_loss = K.mean(reconstruction_loss + kl_loss)
vae.add_loss(vae_loss)
```

FINISHED

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:24 PM.

```
%python
```

FINISHED

```
vae.compile(optimizer=Adam(lr=0.00001))
```

```
  data_min = np.nanmin(X, axis=0)
/usr/local/lib64/python3.4/site-packages/sklearn/preprocessing/data.py:354: RuntimeWarning: Al
l-NaN slice encountered
  data_max = np.nanmax(X, axis=0)
WARNING:tensorflow:Output "decoder" missing from loss dictionary. We assume this was done on p
urpose. The fit and evaluate APIs will not be expecting any data to be passed to "decoder".
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
encoder_input (InputLayer)   (None, 38)                0
_________________________________________________________________
encoder (Model)              [(None, 8), (None, 8), (N 896
_________________________________________________________________
decoder (Model)              (None, 38)                790
=================================================================
Total params: 1,686
Trainable params: 1,686
Non-trainable params: 0
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:24 PM.

FINISHED

Run VAE

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:22 PM.

```
%python

ntest = 60
# test on last day
xtrain, xtest = xmnormd[:-ntest, :], xmnormd[-ntest:, :]
```

FINISHED

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:25 PM.

```
%python

weightsfname = 'vae_%s_SYM:%s.h5' % (jobname, symbol)

if dotraining:
    history = vae.fit(xtrain, epochs=5000, batch_size=64, verbose=2, validation_data=(xtest, N
    vae.save_weights(weightsfname)

    plt.plot(history.history['loss'], color='green', label='training_loss')
    plt.plot(history.history['val_loss'], color='red', label='validation_loss')
    plt.xlabel('iters')
    plt.ylabel('loss')
    plt.legend()
    plt.show()

else:
    vae.load_weights(weightsfname)
```

FINISHED

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:25 PM.

```
%python

latentprojections = encoder.predict(xmnormd)
```

FINISHED

Took 1 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:26 PM.

FINISHED

Extract means of latent projections

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:22 PM.

```
%python

zmean, zlogvar, zsampled = latentprojections
zstd = np.exp(.5*zlogvar)
```

FINISHED

Took 1 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:26 PM.

```
%python

from sklearn.manifold import TSNE

# find 2-dim embeddings of zmean
zmeanembd = TSNE(n_components=2).fit_transform(zmean)
```

FINISHED

Took 8 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:34 PM.

GMM provided by https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture (https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture)

FINISHED

Tuning for number of components by https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.Gri (https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html#sklearn.model_selection.Gri

Using `CV+loglike` instead of `AIC` mostly because not implemented in `GridSearchCV` and also because we are already doing `CV`

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:23 PM.

```
%python

from sklearn.mixture import GaussianMixture
from sklearn.model_selection import GridSearchCV

param_grid = {
    'n_components': np.arange(1, latent_dim),
    'covariance_type': ['full', 'tied', 'diag', 'spherical']
    #'covariance_type': ['diag']
}
```

FINISHED

```
cv = GridSearchCV(GaussianMixture(random_state=0), param_grid=param_grid)
```

```
GridSearchCV(cv='warn', error_score='raise-deprecating',
       estimator=GaussianMixture(covariance_type='full', init_params='kmeans', max_iter=100,
        means_init=None, n_components=1, n_init=1, precisions_init=None,
        random_state=0, reg_covar=1e-06, tol=0.001, verbose=0,
        verbose_interval=10, warm_start=False, weights_init=None),
       fit_params=None, iid='warn', n_jobs=None,
       param_grid={'covariance_type': ['full', 'tied', 'diag', 'spherical'], 'n_components': a
rray([1, 2, 3, 4, 5, 6, 7])},
       pre_dispatch='2*n_jobs', refit=True, return_train_score='warn',
       scoring=None, verbose=0)
```

Took 1 min 27 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:36:53 PM.

```
%python

print (cv.best_params_)
GMMo = cv.best_estimator_
```

FINISHED

```
{'covariance_type': 'full', 'n_components': 7}
```

Took 1 min 19 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:36:53 PM.

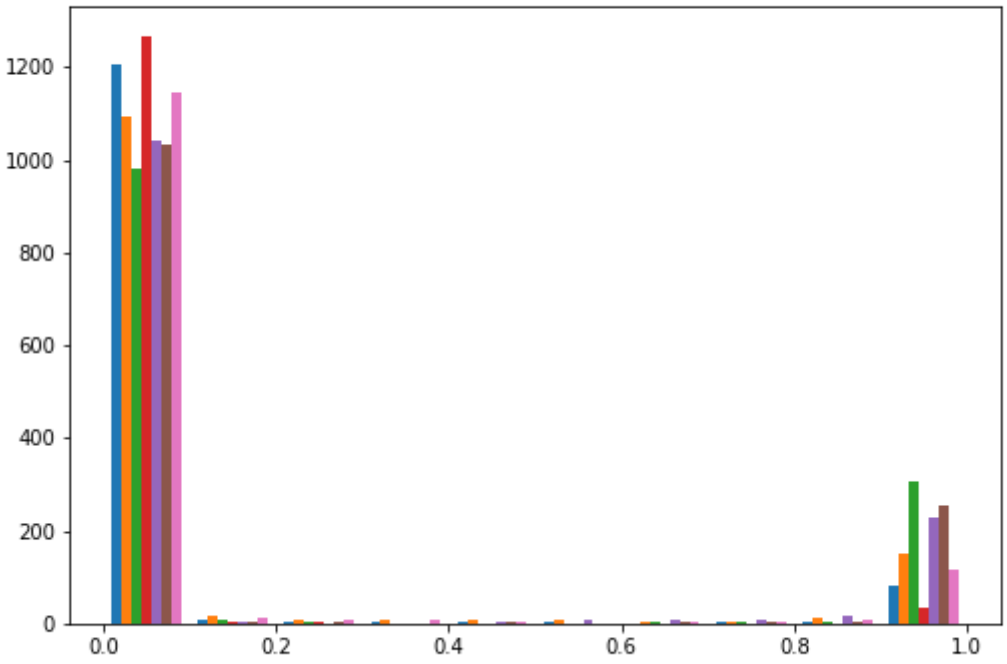
```
%python

gmmlabel = GMMo.fit_predict(zmean)
```

FINISHED

Took 3 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:36:56 PM.

```
%python

gmmproba = GMMo.predict_proba(zmean)
plt.hist(gmmproba)
plt.show()
```

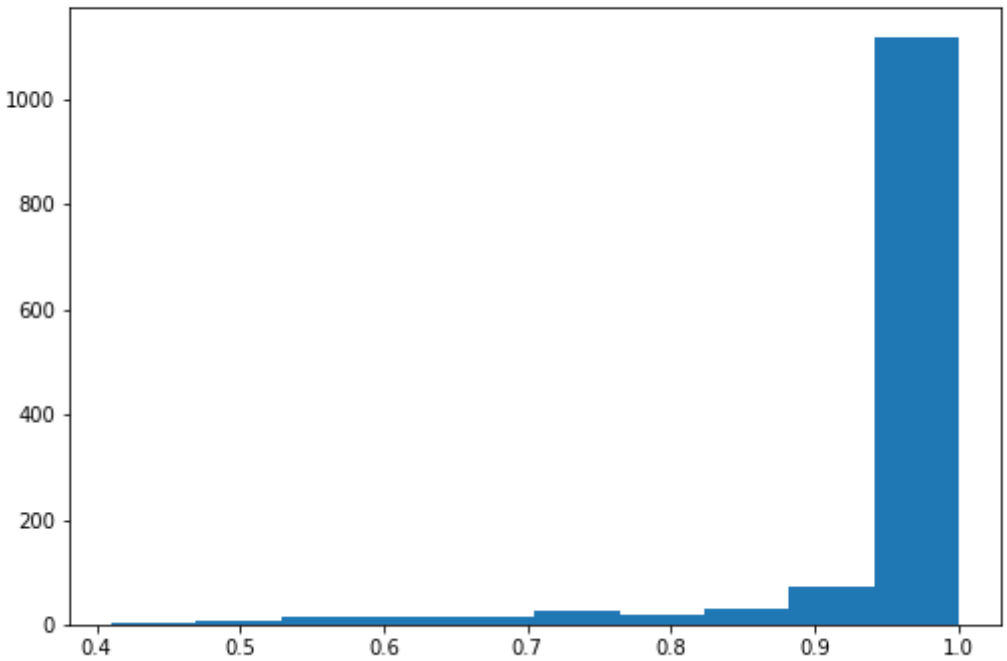
FINISHED



Took 11 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:37:05 PM.

FINISHED

```
%python

# max prob over all classes
gmmprobamax = np.max(gmmproba, axis=1)
plt.hist(gmmprobamax)
plt.show()
```



Took 10 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:37:07 PM.

```
%python

# define abnormal condition as: unlikely to be any class
pcutoff = .1
isabnormal = np.all(gmmproba <= pcutoff, axis=1)
np.sum(isabnormal)
```

FINISHED

```
0
```

Took 1 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:37:07 PM.

```
%python

np.unique(gmmlabel, return_counts=True)
```

FINISHED

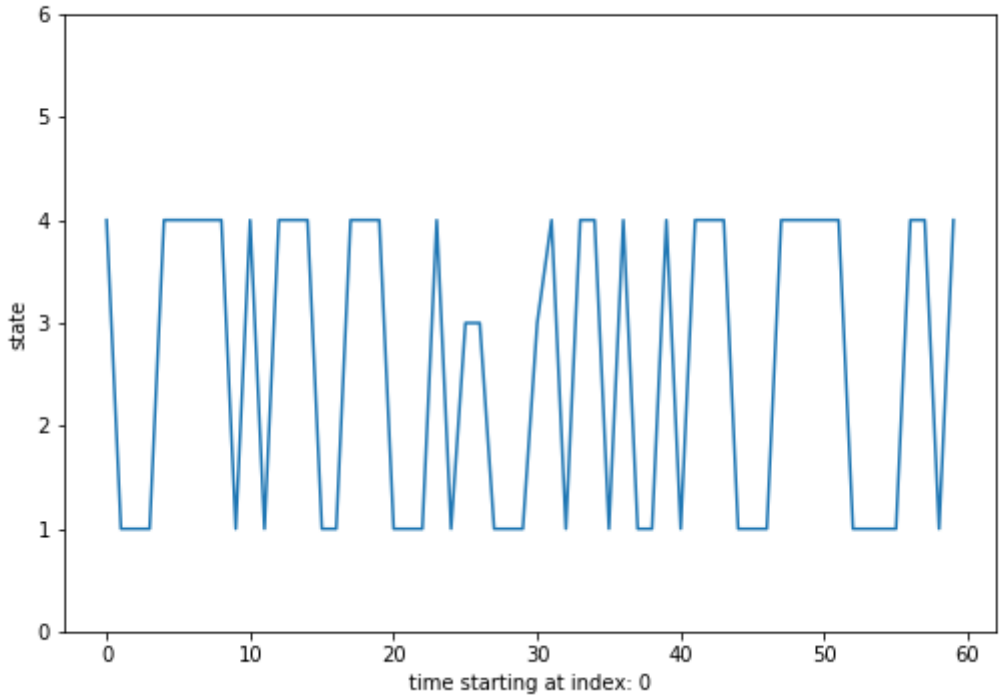
```
(array([0, 1, 2, 3, 4, 5, 6]), array([100, 181, 321,  41, 265, 271, 141]))
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:37:07 PM.

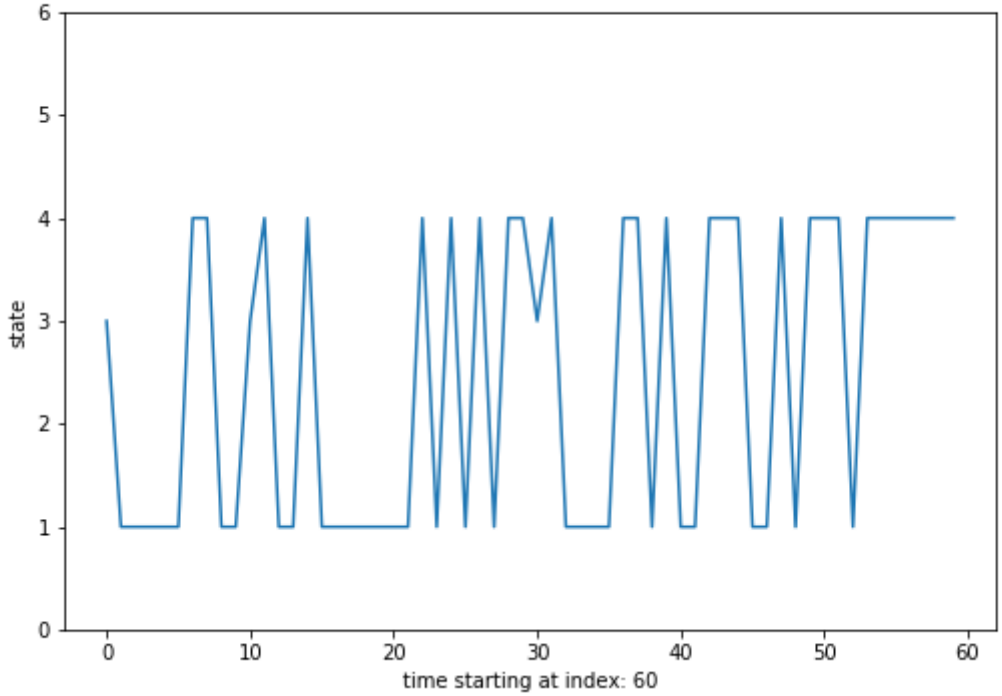
```
%python

limmultiple = 5
for i in range(0, min(limmultiple*60, ntime), 60):
    plt.plot(gmmlabel[i:(i+60)])
    plt.xlabel('time starting at index: %s' % (i))
    plt.ylabel('state')
    plt.ylim((0, max(np.unique(gmmlabel))))
    plt.show()
```

FINISHED

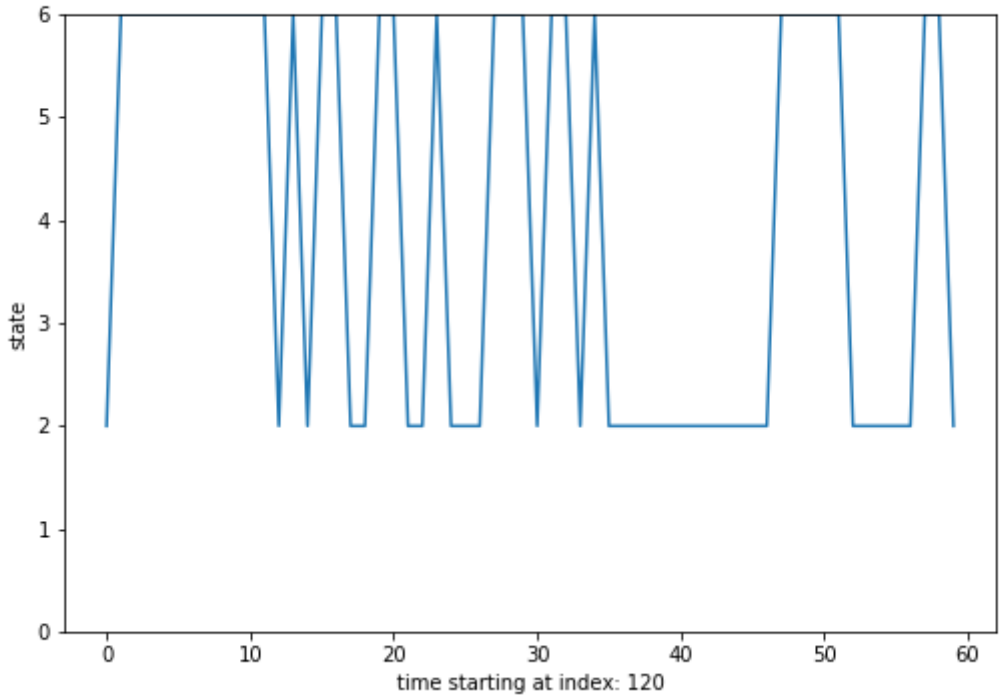
```
[<matplotlib.lines.Line2D object at 0x7faf8c957e48>]
Text(0.5,23,'time starting at index: 0')
Text(48,0.5,'state')
(0, 6)
```



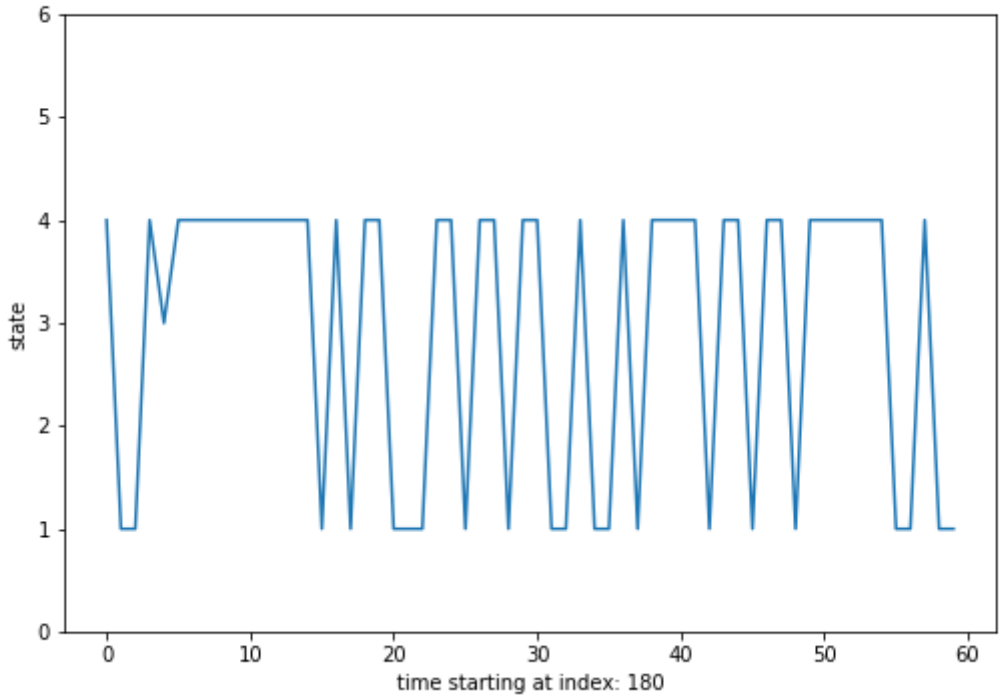
[] Text(0.5,23,'time starting at index: 60') Text(48,0.5,'state') (0, 6)



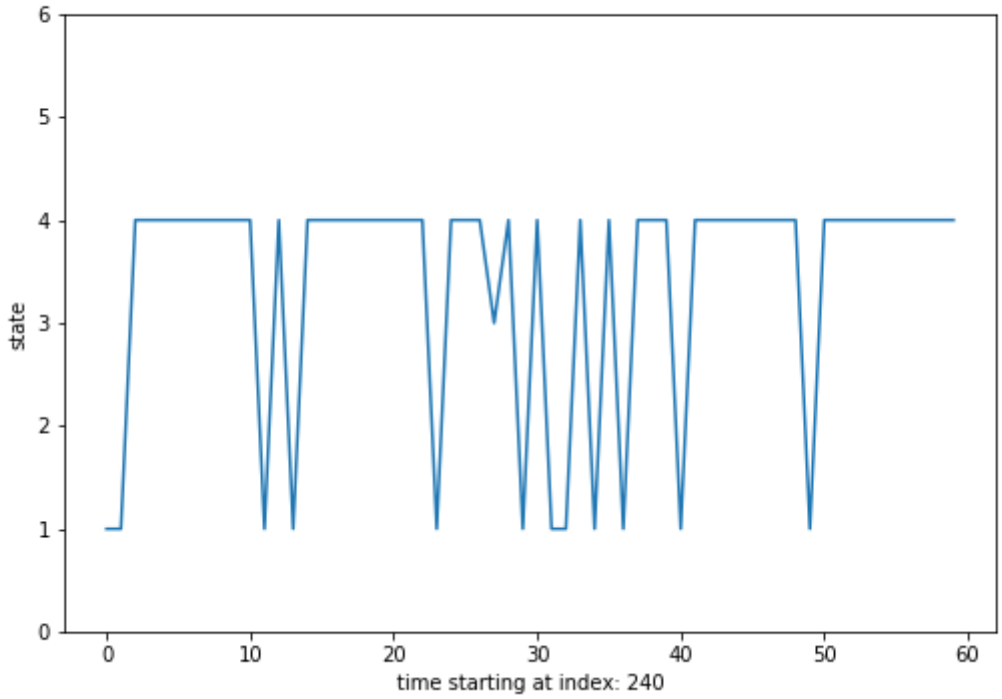
[] Text(0.5,23,'time starting at index: 120') Text(48,0.5,'state') (0, 6)



[] Text(0.5,23,'time starting at index: 180') Text(48,0.5,'state') (0, 6)



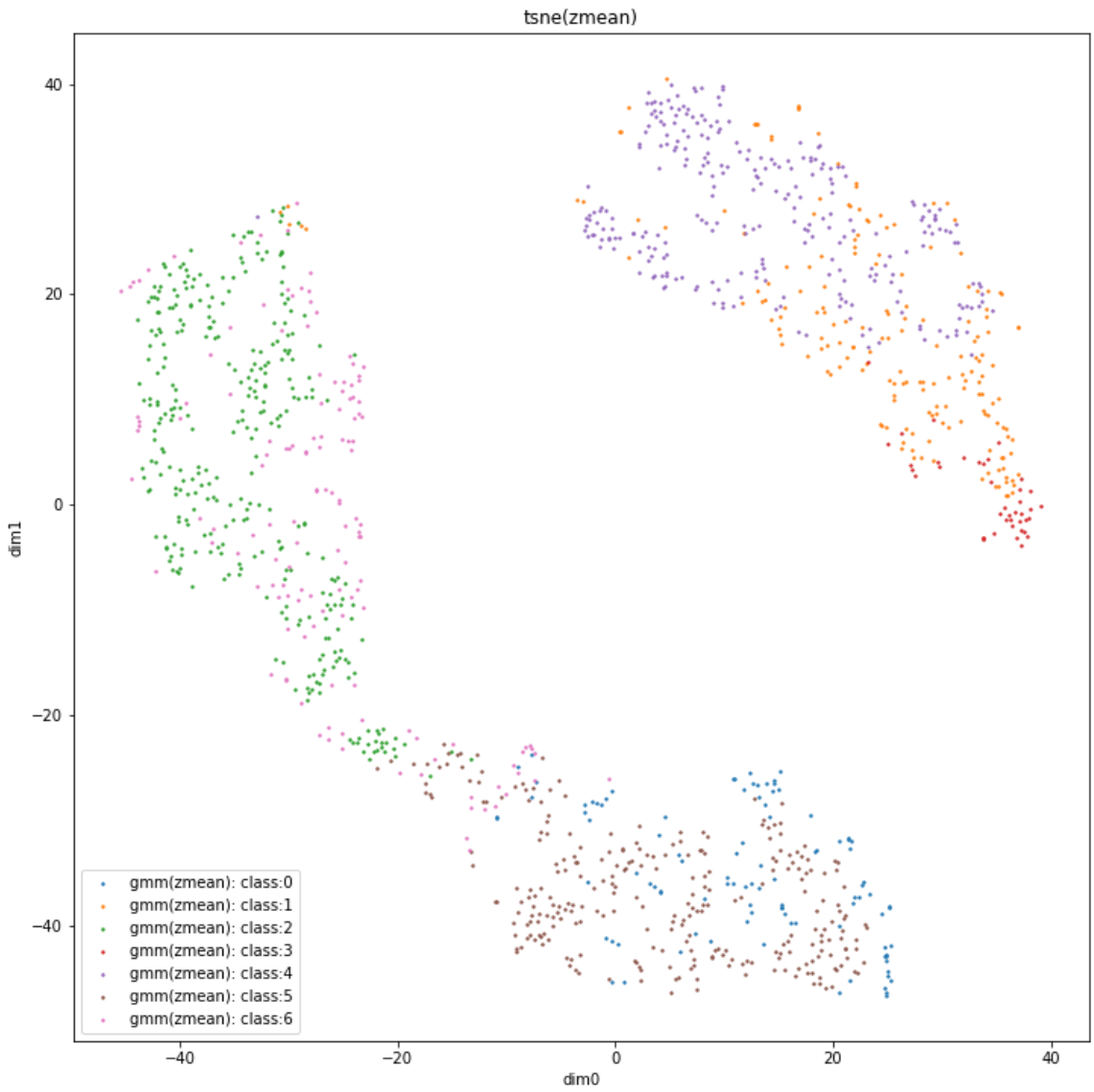
[] Text(0.5,23,'time starting at index: 240') Text(48,0.5,'state') (0, 6)



Took 3 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:37:10 PM.

FINISHED

```
%python

plt.figure(figsize=(12, 12))

for classdex in np.unique(gmmlabel):
    isclass = gmmlabel == classdex
    plt.scatter(zmeanembd[isclass, 0], zmeanembd[isclass, 1], s=2, label='gmm(zmean): class:%s

plt.title('tsne(zmean)')
plt.xlabel('dim0')
plt.ylabel('dim1')
plt.legend()
plt.show()
```



Took 5 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:37:12 PM.

FINISHED

Evaluation

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:35:24 PM.

FINISHED

```
%python

TXLoader(jobname=jobname, symbol=symbol).getcovnames()
```

```
['orders:sum(ABS(book_change))_for_type:Executed_side:Buy_orders_at_touch', 'orders:count(*)_f
or_type:New_side:Sell_orders_at_touch', 'orders:count(*)_for_type:Executed_side:Sell_orders_at
_touch', 'orders:sum(ABS(book_change))_for_type:New_side:All_orders_at_touch', 'orders:sum(ABS
```

```
(book_change))_for_type:New_side:Buy_orders_at_touch', 'orders:count(*)_for_type:Cancelled_sid
e:All_orders_at_touch', 'orders:sum(ABS(book_change))_for_type:Cancelled_side:Buy_orders_at_to
uch', 'orders:count(*)_for_type:New_side:All_orders_at_touch', 'orders:sum(ABS(book_change))_f
or_type:New_side:Sell_orders_at_touch', 'orders:count(*)_for_type:New_side:Buy_orders_at_touc
h', 'orders:count(*)_for_type:All_side:All_orders_at_touch', 'orders:count(*)_for_type:Cancell
ed_side:Sell_orders_at_touch', 'orders:count(*)_for_type:Executed_side:All_orders_at_touch',
 'orders:sum(ABS(book_change))_for_type:All_side:Sell_orders_at_touch', 'orders:sum(ABS(book_c
hange))_for_type:Cancelled_side:Sell_orders_at_touch', 'orders:sum(ABS(book_change))_for_type:
Cancelled_side:All_orders_at_touch', 'orders:sum(ABS(book_change))_for_type:Executed_side:All_
orders_at_touch', 'orders:sum(ABS(book_change))_for_type:All_side:All_orders_at_touch', 'order
s:count(*)_for_type:Executed_side:Buy_orders_at_touch', 'orders:sum(ABS(book_change))_for_typ
e:Executed_side:Sell_orders_at_touch', 'orders:count(*)_for_type:All_side:Buy_orders_at_touc
h', 'orders:count(*)_for_type:Cancelled_side:Buy_orders_at_touch', 'orders:count(*)_for_type:A
ll_side:Sell_orders_at_touch', 'orders:sum(ABS(book_change))_for_type:All_side:Buy_orders_at_t
```

Took 1 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:37:12 PM.

%python FINISHED

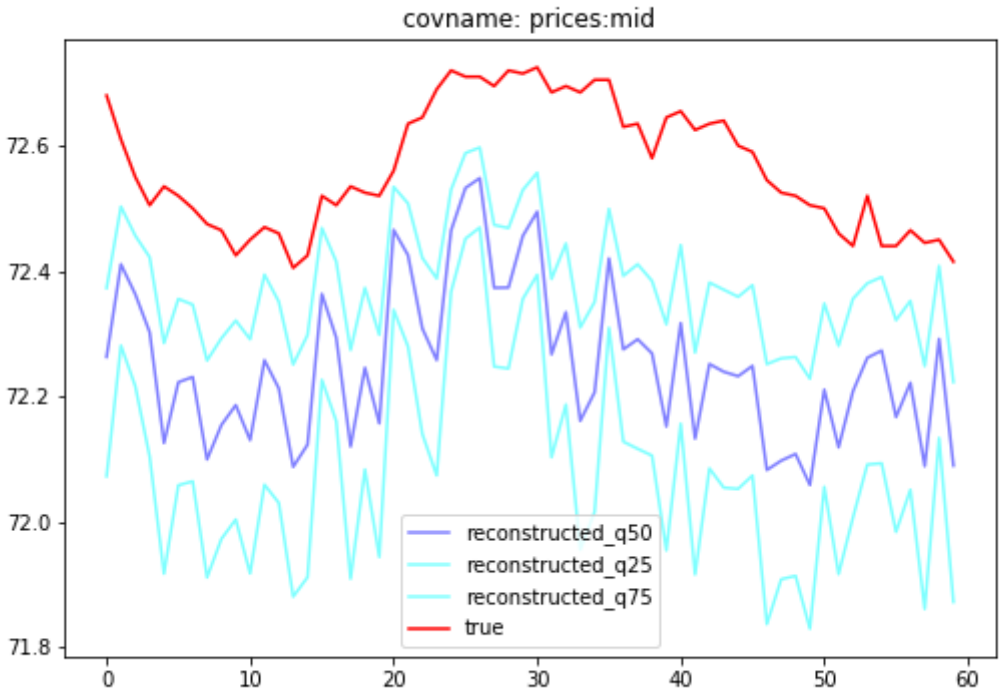
```
def mcsample(nsamples, xm):
    resl = [scaler.inverse_transform(vae.predict(xm)) for i in range(nsamples)]
    resl = [l.reshape((1,)+l.shape) for l in resl]
    return np.concatenate(resl, axis=0)


def evaluate(xm, covname, xlim=None, plot=True):

    covnames = TXLoader(jobname=jobname, symbol=symbol).getcovnames()
    iscov = np.array(covnames)==covname

    predmcsample = mcsample(1000, xm)
    pred = np.quantile(predmcsample, q=.50, axis=0)[:, iscov]
    predlo = np.quantile(predmcsample, q=.25, axis=0)[:, iscov]
    predhi = np.quantile(predmcsample, q=.75, axis=0)[:, iscov]
    true = scaler.inverse_transform(xm)[:, iscov]
    print ('mape_q50: %.6f' % (np.mean(np.abs(pred-true)/true)))

    if plot:

        predplot = pred
        trueplot = true
        predloplot = predlo
        predhiplot = predhi
        if xlim is not None:
            predplot = predplot[xlim[0]:xlim[1]]
            trueplot = trueplot[xlim[0]:xlim[1]]
            predloplot = predloplot[xlim[0]:xlim[1]]
            predhiplot = predhiplot[xlim[0]:xlim[1]]

        plt.plot(predplot, color=(0,0,1,.5), label='reconstructed_q50')
        plt.plot(predloplot, color=(0,1,1,.5), label='reconstructed_q25')
        plt.plot(predhiplot, color=(0,1,1,.5), label='reconstructed_q75')
        plt.plot(trueplot, color='red', label='true')
        plt.title('covname: %s' % (covname))
        plt.legend()
        plt.show()

    return pred, predlo, predhi, true
```

Took 0 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:37:12 PM.

```
%python

evaluate(xtrain, 'prices:mid', xlim=(0, 60))
```

FINISHED

```
mape_q50: 0.001795
```




```
(array([[72.26305008], [72.41083145], [72.36302185], ..., [72.51491547], [72.43466187], [72.5436821 ]]),
array([[72.0727005 ], [72.28107262], [72.21635056], ..., [72.42501831], [72.32607651], [72.46842766]]),
array([[72.37291527], [72.50282097], [72.45640182], ..., [72.57067871], [72.50421333], [72.59588051]]),
array([[72.68 ], [72.61 ], [72.55 ], ..., [72.53 ], [72.515], [72.53 ]]))
```

Took 42 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:37:55 PM.

```
%python

evaluate(xtrain, 'orders:sum(ABS(book_change))_for_type:New_side:All_orders_at_touch', xlim=(0
```

FINISHED

```
mape_q50: 0.424452
```

(array([[110973.4921875], [126835.171875], [123685.17578125], ..., [148256.3046875], [134947.3828125], [160564.578125]]), array([[99155.88867188], [111733.71679688], [109659.76367188], ..., [132078.109375], [119473.04101562], [141960.30859375]]), array([[125813.91796875], [142340.47265625], [139415.72265625], ..., [166049.4921875], [149378.92578125], [179953.30078125]]), array([[57500.], [125600.], [126800.], ..., [283270.], [264800.], [342000.]]))

Took 1 min 18 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:38:31 PM.

```
%python

evaluate(xtest, 'prices:mid')
```

FINISHED

```
mape_q50: 0.002068
```

(array([[71.90044785], [72.05753326], [72.55506134], [72.05882645], [72.24345779], [71.992836], [71.94952011], [72.13569641], [71.84088898], [71.83257675], [72.11091995], [71.91559982], [71.77503967], [72.01104355], [71.92921448], [71.78493118], [71.83562088], [71.83609772], [71.93016052], [71.87775421], [71.95138168], [71.86733246], [71.96997833], [71.84139252], [71.80587006], [71.90124512], [71.85871124], [72.03827667], [71.80445862], [71.92365646], [71.8985405], [72.14691544], [72.05832672], [71.89338303], [71.91875076], [72.00173569], [71.88177872], [71.89980316], [71.8931694], [71.87229538], [71.87877655], [71.99327469], [71.98723984], [71.95034027], [71.94104767], [71.84588623], [71.96457291], [71.87484741], [71.95484543], [71.91415787], [71.87386703], [71.91466522], [71.91791153], [71.91464615], [71.87085342], [72.02932739], [71.93327713], [71.95361328], [71.95537186], [72.00236511]]), array([[71.64979935], [71.82876968], [72.44952774], [71.81105804], [72.07691956], [71.74656105], [71.66734695], [71.89112091], [71.57292175], [71.58779144], [71.86124229], [71.64568138], [71.49410629], [71.78559685], [71.6564312], [71.53130531], [71.53932762], [71.54289055], [71.701931], [71.61340523], [71.72132683], [71.60987282], [71.70480728], [71.56921959], [71.55758286], [71.65126991], [71.60572815], [71.81304359], [71.53400993], [71.65085983], [71.66663361], [71.95526695], [71.8106308], [71.66493797], [71.64190102], [71.75354958], [71.65060806], [71.63931656], [71.62945557], [71.62812424], [71.63726425], [71.75560188], [71.77648163], [71.71965408], [71.68821907], [71.60828972], [71.70960045], [71.60210419], [71.72772789], [71.67573357], [71.6162796], [71.65988541], [71.68156624], [71.67888451], [71.64240646], [71.81794739], [71.67695236], [71.70303345], [71.71178055], [71.75970459]]), array([[72.13020515], [72.24064827], [72.61125755], [72.22677994], [72.39102364], [72.17770767], [72.15088844], [72.28740692], [72.066185], [72.06208229], [72.27655411], [72.11427498], [71.99687386], [72.20474815], [72.14392853], [71.99611855], [72.04131317], [72.06325912], [72.13222122], [72.07551384], [72.14840889], [72.07907104], [72.13926888], [72.05661583], [72.02058601], [72.09732246], [72.07400513], [72.20720673], [72.01213646], [72.12734985], [72.09602547], [72.29674721], [72.22527313], [72.0867424], [72.11684227], [72.16861916], [72.08498001], [72.09113121], [72.08468628], [72.07463646], [72.07059288], [72.18282509], [72.20686913], [72.1333313], [72.10802269], [72.05381775], [72.15163803], [72.06674004], [72.15597153], [72.11037636], [72.08274841], [72.12374306], [72.09451675], [72.12341881], [72.07111359], [72.20245171], [72.12499619], [72.13943481], [72.15838814], [72.19541359]]), array([[72.045], [72.085], [72.125], [72.055], [72.035], [71.985], [71.97], [71.995], [71.995], [72.005], [72.005], [71.99], [71.965], [71.965], [71.995], [71.975], [71.985], [71.985], [71.975], [71.945], [71.975], [71.935], [71.935], [71.97], [71.955], [71.965], [72.005], [72.005], [72.005], [72.025], [72.055],

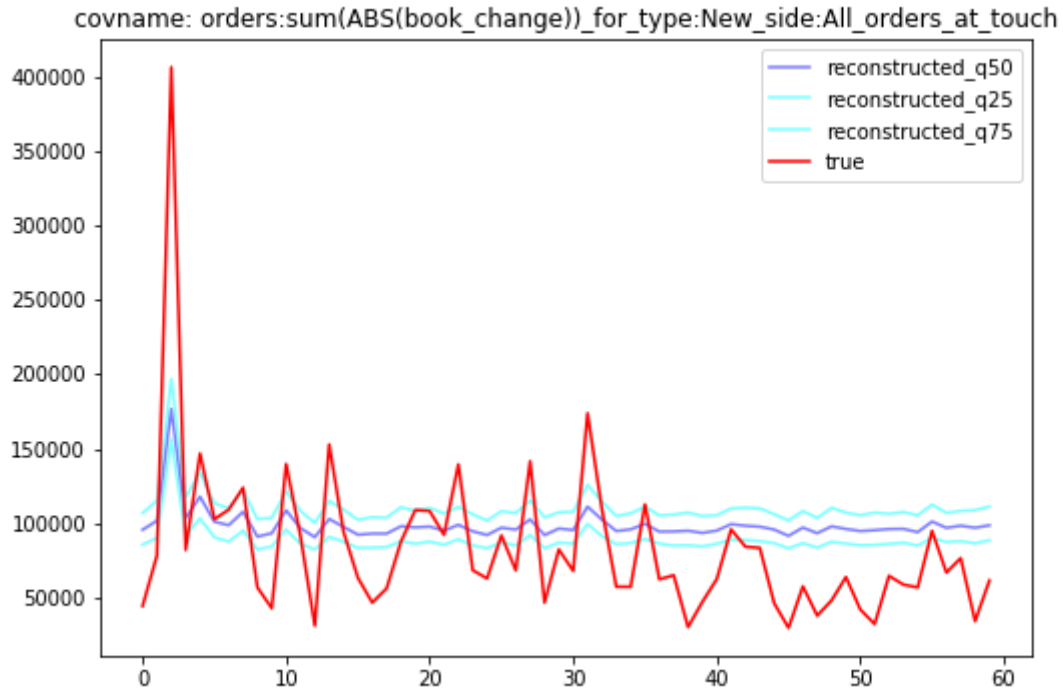
[72.035], [72.1], [72.115], [72.115], [72.095], [72.105], [72.085], [72.105], [72.105], [72.105], [72.135], [72.155], [72.09], [72.095], [72.115], [72.125], [72.125], [72.145], [72.105], [72.125], [72.115], [72.105], [72.155], [72.125], [72.155], [72.145], [72.165], [72.15], [72.165]]))

Took 44 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:38:39 PM.

```
%python

evaluate(xtest, 'orders:sum(ABS(book_change))_for_type:New_side:All_orders_at_touch')
```

FINISHED

```
mape_q50: 0.583610
```



(array([[95501.10546875], [101535.203125], [176733.75], [103406.40625], [117786.125], [100964.1015625], [98545.7890625], [107641.77734375], [90882.859375], [93150.7421875], [108549.26953125], [96580.24609375], [90532.2734375], [102770.87109375], [97409.09765625], [92170.5390625], [92885.890625], [92890.93359375], [97857.34375], [97102.75], [97615.078125], [94857.03125], [98823.6015625], [94445.98046875], [91971.9609375], [96872.66015625], [95627.3515625], [102470.88671875], [91858.8359375], [96339.23828125], [95282.8359375], [111070.8671875], [102348.88671875], [94637.21875], [95738.71875], [99511.5546875], [94272.546875], [94341.1328125], [94834.52734375], [93377.99609375], [94911.203125], [99256.45703125], [98193.265625], [97567.69921875], [95621.80078125], [91373.1484375], [96954.29296875], [93195.30859375], [97780.5703125], [96046.37890625], [94637.09765625], [95204.53515625], [95971.078125], [96116.49609375], [93889.03125], [101020.45703125], [96776.55078125], [98222.26953125], [96840.30859375], [98478.0078125]]), array([[85534.18554688], [89984.75195312], [156048.37890625], [91439.71484375], [103178.49804688], [90452.6953125], [87574.61523438], [94862.84960938], [82279.44335938], [84189.08789062], [95633.71484375], [86043.49804688], [82231.27539062], [90743.63671875], [87505.66015625], [83293.77929688], [83462.70898438], [83859.76953125], [87757.71875], [86114.58789062], [87580.51367188], [85403.8515625], [89091.0859375], [84915.66210938], [83105.66015625], [87024.36523438], [84977.73632812], [91898.21679688], [83128.29296875], [86823.64257812], [86186.07226562], [99384.31054688], [91257.56640625], [

85763.09570312], [86548.97070312], [89322.6484375], [86170.13476562], [84882.15625], [85116.07617188], [84295.40429688], [86086.0234375], [88766.7109375], [88517.22851562], [87889.234375], [86461.05078125], [82959.4609375], [86562.76953125], [83455.55859375], [87407.90039062], [86178.38867188], [85057.9765625], [85288.42578125], [86017.41992188], [86824.36914062], [84852.31835938], [90075.71875], [87204.31445312], [87836.35546875], [86413.22851562], [88433.87695312]]), array([[106812.74609375], [115059.43164062], [196902.51953125], [117251.875], [134066.91015625], [113363.43359375], [109997.20117188], [121848.44140625], [102751.109375], [103541.45507812], [121350.30078125], [108207.9453125], [100092.35351562], [114951.84960938], [109121.390625], [102221.08984375], [103963.578125], [103651.43164062], [110720.90625], [108239.9921875], [110216.37304688], [106476.94921875], [110878.40820312], [105864.70117188], [101705.13867188], [108024.07226562], [106789.48632812], [115105.125], [103855.40429688], [107247.96289062], [107627.04492188], [125631.56054688], [114467.828125], [104795.16992188], [106518.89648438], [111061.58398438], [105203.89648438], [105711.23242188], [107111.0234375], [104715.53125], [105235.91210938], [109879.65429688], [110318.72460938], [109911.72070312], [105930.07226562], [101766.24609375], [108126.40234375], [103409.27148438], [110258.27929688], [107062.734375], [105221.69140625], [107074.5234375], [106230.5234375], [107425.3515625], [105267.93554688], [112460.10742188], [106868.35351562], [108178.14648438], [108728.63867188], [111159.5546875]]), array([[44200.], [78200.], [406800.], [82000.], [146800.], [102600.], [109000.], [123800.], [56719.], [42700.], [139775.], [91200.], [30947.], [152900.], [94200.], [63200.], [46500.], [56000.], [87500.], [108800.], [108100.], [92100.], [139500.], [68300.], [62700.], [91658.], [68200.], [141600.], [46510.], [82229.], [67800.], [173900.], [109300.], [57200.], [57100.], [112600.], [62150.], [64900.], [30000.], [47000.], [62300.], [95730.], [84100.], [83300.], [46100.], [29500.], [57400.], [37700.], [48000.], [63655.], [41900.], [32000.], [64500.], [58600.], [56700.], [94750.], [66700.], [76400.], [34000.], [61400.]]))

Took 15 sec. Last updated by tianlechen@gmail.com at March 01 2019, 5:38:46 PM.

```
%python
```

READY