Problem 3

In [1]:
```
import d2l
import math
import mxnet as mx
from mxnet import autograd, gluon, init, nd
from mxnet.gluon import loss as gloss, nn, rnn
from mxnet.gluon import data as gdata
import time
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

3.1 Data Iterator

In [2]:
```
feature_df = pd.read_csv("feature.csv", header = None)
featureMatrix = nd.array(feature_df.values)
label_df = pd.read_csv("label.csv", header = None)
labelMatrix = nd.array(label_df.values)

ctx = d2l.try_gpu()
featureMatrix = featureMatrix[890:1233,:].as_in_context(ctx)
labelMatrix = labelMatrix[890:,:].as_in_context(ctx)
print(featureMatrix.shape, labelMatrix.shape)
```

```
(343, 2525) (343, 505)
```

In [3]:
```
testfeature_df = pd.read_csv("test_feature.csv", header = None)
testfeatureMatrix = nd.array(testfeature_df.values)

testlabel_df = pd.read_csv("test_label.csv", header = None)
testlabelMatrix = nd.array(testlabel_df.values)

print(testfeatureMatrix.shape, testlabelMatrix.shape)
```

```
(26, 2525) (26, 505)
```

3.1 Model Definition

We encountered problem when training the rnn with the data provided (NaN value in prediction at the 25^{th} batch. We could prove that the network is defined correctly and the data iterator works normally. And when we used Keras, it works fine too. We spent over 80 hours on that but could not find why, which is really frustrating. We understand the basic idea in time series model using RNN.It is predicting a continuous function using the current prices of that day and the past prices integrated in the hidden state. Our goal is to predict a particular function: $Y_{t+1} = f(X_t, H_t)$.

In [4]:
```python
import sys
sys.path.insert(0, '..')
```

In [5]:
```python
class RNNReg(nn.Block):
    def __init__(self, rnn_layer, out_size=505, **kwargs):
        super(RNNReg, self).__init__(**kwargs)
        self.rnn = rnn_layer
        self.out_size = out_size # we only predict the open price next day
        self.dense = nn.Dense(out_size)

    def forward(self, inputs, state):
        # the shape is (batch_size, time_step_forward, sample_length)
        X = inputs.reshape(1, inputs.shape[0], inputs.shape[1])
        Y, state = self.rnn(X, state)
        output = self.dense(Y.reshape((-1, Y.shape[-1])))
        #output = output.asnumpy()
        #output[np.isnan(output)] == 3.
        #output = nd.array(output).as_in_context(ctx)
        #print(output)
        return output, state

    def begin_state(self, *args, **kwargs):
        return self.rnn.begin_state(*args, **kwargs)
```

In [6]:
```python
def predict_rnn_gluon(inputs, step_forward, model, ctx):
    # inputs should be of dimension (days of that year)*2525
    state = model.begin_state(batch_size=1, ctx=ctx)
    output = [inputs[0]]
    for t in range(len(inputs) + step_forward - 1):
        X = nd.array(output[-1], ctx=ctx).reshape(1,2525)
        (Y, state) = model(X, state)
        if t < len(inputs) - 1:
            output.append(inputs[t+1])
        else:
            output.append(Y.reshape((-1, Y.shape[-1])))
    return output
```

In [7]:
```python
def grad_clipping_gluon(model, theta, ctx):
    params = [p.data() for p in model.collect_params().values()]
    d2l.grad_clipping(params, theta, ctx)
```

In [8]:

```
# this is the dummy data, need to be replaced with
# X = (batch_size, all_data_per_day), Y = (batch_size, all_open_price_next_day)
# where X and Y are both ndarrays, so that just treat them as train_features and
#train_features = nd.zeros((67, 2525), ctx=ctx)
#train_labels = nd.ones((67, 505), ctx=ctx)
#train_iter = gdata.DataLoader(gdata.ArrayDataset(train_features, train_labels),


def train_and_predict_rnn_gluon(model, num_hiddens, data_iter, ctx, num_epochs,\
                                num_steps, lr, clipping_theta, batch_size):
    loss = gloss.L2Loss()
    model.initialize(ctx=ctx, force_reinit=True, init=init.Normal(0.01))
    trainer = gluon.Trainer(model.collect_params(), 'sgd',
                            {'learning_rate': lr, 'momentum': 0, 'wd': 0})
    start = time.time()
    for epoch in range(num_epochs):
        state = model.begin_state(batch_size=batch_size, ctx=ctx)
        for X, Y in data_iter:
            for s in state:
                s.detach()
            with autograd.record():
                (output, state) = model(X, state)
                # print('output: ',output.shape)
                y = Y.T.reshape((-1,))
                # print('y: ',y.shape)
                l = loss(output, y).mean()
            l.backward()
            # Clip the gradient
            grad_clipping_gluon(model, clipping_theta, ctx)
            # Since the error has already taken the mean, the gradient does
            # not need to be averaged
            trainer.step(1)
        if (epoch + 1) % 20 == 0:
            print('epoch: ', epoch+1, ', loss: ', l.asscalar())
```

3.1.1 Single Layer RNN

In [9]:
```
num_steps = 1
num_epochs, batch_size, lr, clipping_theta = 200, 50, 10, 1e-3

num_hiddens = 1024
rnn_layer = rnn.RNN(num_hiddens)
rnn_layer.initialize(ctx=ctx)

single_rnn_model = RNNReg(rnn_layer, 505)
single_rnn_model.initialize(force_reinit=True, ctx = ctx)

train_iter = gdata.DataLoader(gdata.ArrayDataset(featureMatrix, labelMatrix), bat

train_and_predict_rnn_gluon(single_rnn_model, num_hiddens, train_iter, ctx, num_e
                            num_steps, lr, clipping_theta, batch_size)
```

```
epoch:  20 , loss:  0.29008391
epoch:  40 , loss:  0.28979078
epoch:  60 , loss:  0.28905156
epoch:  80 , loss:  0.2893017
epoch:  100 , loss:  0.29063302
epoch:  120 , loss:  0.29117203
epoch:  140 , loss:  0.29128182
epoch:  160 , loss:  0.29107174
epoch:  180 , loss:  0.29138774
epoch:  200 , loss:  0.29053017
```

In [10]:
```
predict = predict_rnn_gluon(testfeatureMatrix, 1, single_rnn_model, ctx)
predict[-1]
```

Out[10]:
```
[[4.1881795 4.4396496 4.1789064 4.520386  4.1682577 4.185415  4.1361494
  4.289607  4.352634  4.112878  4.077543  4.1304235 4.145678  4.109722
  4.1369247 4.319725  4.083404  4.3764725 4.3716235 4.3006854 4.0621767
  4.2879024 4.1203113 4.351359  4.135375  4.162312  4.3727336 4.356408
  4.218062  4.4475865 4.477019  4.3464546 4.2486477 4.319681  4.334162
  4.379688  4.4660344 4.2817826 4.529161  4.6391997 4.361038  4.313606
  4.1821733 4.376094  4.4018235 4.1538424 4.309873  4.3158965 4.127034
  4.4419994 4.327807  4.4094715 4.399322  4.3169675 4.5986977 4.2755084
  4.3626313 4.188605  4.415745  4.5694275 4.3959694 4.503953  4.3918386
  4.319938  4.504621  4.4031563 4.3071113 4.5456676 4.2660975 4.4628515
  4.470452  4.5557594 4.4190145 4.359193  4.5538993 4.466226  4.672306
  4.3558893 4.3511424 4.515755  4.445055  4.1247196 4.512501  4.492761
  4.345106  4.4913635 4.315505  4.5104494 4.474726  4.4010096 4.3342957
  4.382322  4.5273128 4.5084715 4.2781243 4.5795355 4.6790795 4.4933696
  4.297127  4.4270177 4.5624914 4.3722105 4.268285  4.617367  4.6258025
  4.545555  4.417334  4.5115657 4.3059273 4.333788  4.454448  4.3860383
  4.6009626 4.3446364 4.2670965 4.5711765 4.4249    4.684702  4.2916665
  4.389802  4.315543  4.584459  4.475813  4.568372  4.396027  4.6310215
  4.4227657 4.462055  4.1724534 4.218629  4.3930497 4.3683934 4.524188
  4.484877  4.6103854 4.3376226 4.463432  4.49222   4.522874  4.4824333
  4.0529265 4.550115  4.440011  4.4695506 4.427037  4.32315   4.2781897
  4.348941  4.466195  4.548265  4.301691  4.180839  4.246786  4.2428956
  4.1769423 4.429079  4.604107  4.378961  4.351573  4.2758713 4.456281
  4.505477  3.9755383 4.3881435 4.3467555 4.6817465 4.398725  4.4379606
  4.371284  4.508142  4.2829785 4.560067  4.3540425 4.3663964 4.45834
  4.483651  4.4956694 4.2648215 4.4215474 4.583127  4.3504004 4.443366
  4.1794367 4.279498  4.3461223 4.2619443 4.477511  4.388888  4.4556527
  4.366335  4.2781    4.391987  4.455513  4.4954557 4.11834   4.298743
  3.9806929 4.4001307 4.3816414 4.223075  4.434826  4.1734524 4.3037047
  4.4690166 4.033562  4.228308  4.391083  3.9973726 4.4614677 4.261359
  4.036935  4.3085074 4.2584624 4.1682897 4.2388864 4.1870747 4.272182
  4.290857  4.181231  4.164324  4.458402  4.101671  4.2208586 4.2650223
  4.484548  4.1086373 4.1712747 4.3046083 4.250036  4.087135  4.3791924
  4.1709027 4.3208704 4.069361  4.206694  4.2754807 4.342524  4.1755376
  4.346052  4.326028  4.2154903 4.108364  4.1890554 4.307903  4.2517304
  4.265238  3.9995954 4.1376905 4.290211  4.273009  4.1698594 4.32317
  4.0788736 4.013223  4.0812917 4.159992  4.229023  4.062604  3.8285513
  4.249548  4.2136364 4.009707  4.0275145 4.2597475 4.154443  4.004173
  4.349908  3.9746802 4.165792  4.319636  3.9186947 4.254696  4.1946282
  3.9528494 4.0398693 4.377534  4.2523494 4.200009  4.1140733 4.386727
  4.394963  4.252523  4.087912  4.374378  4.292756  4.4034076 4.316575
  4.2593756 4.0803514 4.471423  4.3163276 4.105184  4.3230767 4.112249
  4.3799596 4.3465886 4.2974424 4.4395175 4.34898   4.166336  4.156699
  4.3220487 4.4586883 4.163857  4.3096585 4.376001  4.182272  4.2876596
  4.2644873 4.1670203 4.3404756 4.354896  4.073664  4.103201  4.127166
  4.5558043 4.2918253 4.6017027 4.3576775 3.9256785 4.398777  4.3768983
  4.2824793 4.1748123 4.2476444 4.421947  4.544196  4.490995  4.830138
  4.2607307 4.542361  4.476418  4.5021515 4.5984263 4.2688503 4.50397
  4.3773913 4.2489724 4.441477  4.3921666 4.4822435 4.311435  4.3697815
  4.503956  4.53024   4.6120276 4.603463  4.6998453 4.306417  4.517115
  4.545903  4.5751987 4.761828  4.599499  4.5225396 4.7183156 4.659539
  4.507246  4.5109396 4.5852966 4.6748276 4.4890285 4.7364545 4.570488
  4.6660366 4.5740533 4.783895  4.639114  4.70262   4.4144683 4.5637546
  4.7147818 4.5456796 4.6994247 4.508074  4.3749733 4.51791   4.6449757
  4.5189934 4.3281026 4.4009314 4.6180396 4.4322944 4.530361  4.175234
```

```
  4.361845  4.939455  4.654832  4.4865656 4.445834  4.567841  4.599501
  4.1506777 4.3853483 4.4692225 4.358509  4.4013257 4.513316  4.359702
  4.6385083 4.083259  4.5397267 4.1399016 4.3702393 4.1863847 4.2930384
  4.384549  4.3503394 4.28198   4.417181  4.093589  4.15954   4.1447325
  4.309003  4.2373953 4.3469024 4.1362934 4.396023  4.3645186 4.2266026
  4.3522696 4.283894  4.3113146 4.3859034 4.4561205 4.1299524 4.087761
  4.2931857 4.2526097 4.3258657 4.3462296 4.1564975 4.2641582 4.3782377
  4.465062  4.261258  4.2734146 4.300128  4.173198  4.13158   4.261663
  4.411954  4.183697  4.570612  4.1664925 4.2595453 4.5217338 4.2268286
  4.3043675 4.2447934 4.370814  4.5479307 4.56428   4.285775  4.0634875
  4.031266  4.2295575 4.1286426 4.2269382 4.1033845 4.322987  4.4200006
  4.327729  4.5784497 4.300579  4.412547  4.301372  4.108586  4.1419425
  4.382287  4.522571  4.552255  4.420993  4.48382   4.2322702 4.2741203
  4.547503  4.364925  4.2817707 4.240322  4.4749146 4.038536  4.29867
  4.2739797 4.383325  4.358176  4.209751  4.344235  4.279485  3.9559815
  4.3900776 4.3824506 4.350583  4.419975  4.315593  4.294198  4.2136803
  4.323758  4.096654  4.2590075 4.2099657 4.308947  4.4278054 4.158452
  4.142184 ]]
<NDArray 1x505 @gpu(0)>
```

3.2 GRU

In [9]:

```python
num_steps = 1
num_epochs, batch_size, lr, clipping_theta = 200, 50, 10, 1e-3

num_hiddens = 1024
rnn_layer = rnn.GRU(num_hiddens)
rnn_layer.initialize(ctx=ctx)

gru_model = RNNReg(rnn_layer, 505)
gru_model.initialize(force_reinit=True, ctx = ctx)

train_iter = gdata.DataLoader(gdata.ArrayDataset(featureMatrix, labelMatrix), ba

train_and_predict_rnn_gluon(gru_model, num_hiddens, train_iter, ctx, num_epochs,\
                            num_steps, lr, clipping_theta, batch_size)
```

```
epoch:  20 , loss:  4.6150594
epoch:  40 , loss:  1.4953979
epoch:  60 , loss:  0.29515418
epoch:  80 , loss:  0.28262606
epoch:  100 , loss:  0.2824281
epoch:  120 , loss:  0.28235632
epoch:  140 , loss:  0.28231528
epoch:  160 , loss:  0.28255248
epoch:  180 , loss:  0.2822188
epoch:  200 , loss:  0.2820875
```

In [10]:
```
predict = predict_rnn_gluon(testfeatureMatrix, 1, gru_model, ctx)
predict[-1]
```

Out[10]:
```
[[4.3243923 4.3712196 4.2999554 4.3228035 4.370968  4.243898  4.24508
  4.260172  4.2744026 4.2926326 4.133703  4.1503797 4.1636024 4.166166
  4.1735983 4.2187915 4.1441245 4.1811996 4.209031  4.228094  4.2453647
  4.2537217 4.2141094 4.2291927 4.292108  4.3069077 4.293301  4.314372
  4.331689  4.321029  4.334781  4.332673  4.385876  4.396977  4.372663
  4.353887  4.3971214 4.3631296 4.4043894 4.395302  4.3683343 4.3805194
  4.331608  4.371286  4.3912764 4.34378   4.3389363 4.3791585 4.359755
  4.36171   4.3342    4.3327365 4.3825035 4.3644032 4.3447666 4.3327527
  4.3867536 4.36746   4.3402143 4.3239417 4.456101  4.491688  4.499715
  4.4754367 4.455524  4.407231  4.3669066 4.4294105 4.404113  4.4086637
  4.4513197 4.407497  4.440373  4.468761  4.416025  4.4803414 4.458407
  4.5010395 4.4680195 4.4959383 4.452929  4.452687  4.461529  4.4398236
  4.498156  4.4103856 4.4506035 4.454333  4.448293  4.41614   4.4063644
  4.430148  4.3778877 4.432131  4.4275146 4.5139103 4.5172887 4.539809
  4.492002  4.48374   4.4773107 4.47864   4.4597406 4.412051  4.488376
  4.49646   4.479258  4.4647064 4.493534  4.4576163 4.4388666 4.4620886
  4.4814544 4.4671984 4.4816113 4.415739  4.435778  4.4382777 4.405477
  4.461494  4.4289055 4.430612  4.4055886 4.434807  4.41618   4.4431033
  4.4068403 4.4108987 4.4159036 4.3920403 4.3869033 4.4015317 4.3707314
  4.393167  4.3495474 4.413126  4.4265256 4.442393  4.414072  4.4358807
  4.4438796 4.450759  4.457162  4.405572  4.4577923 4.3760047 4.360598
  4.377735  4.366035  4.344984  4.416061  4.3623343 4.4297357 4.394718
  4.4375753 4.3967547 4.3755226 4.4150615 4.400267  4.4122376 4.354832
  4.3653684 4.384923  4.397888  4.42769   4.49346   4.4544263 4.445093
  4.4937515 4.468975  4.4312053 4.492381  4.450809  4.459275  4.4846206
  4.435464  4.453887  4.4083486 4.469505  4.4380565 4.3912616 4.354562
  4.3766093 4.360592  4.3836064 4.3790083 4.36886   4.360545  4.3602905
  4.396006  4.322386  4.3188562 4.36833   4.3408446 4.3357105 4.3343825
  4.334987  4.284894  4.3282304 4.326543  4.417909  4.4092083 4.4070005
  4.3304424 4.428676  4.371606  4.333963  4.316894  4.342611  4.374566
  4.264973  4.3495154 4.2732067 4.332018  4.316192  4.2483234 4.2090435
  4.241964  4.270158  4.270219  4.279534  4.301209  4.2828627 4.2620106
  4.230772  4.2965136 4.2642365 4.2388153 4.270497  4.255042  4.249255
  4.252997  4.2586045 4.2600527 4.215715  4.239491  4.2279987 4.217479
  4.2495756 4.2248487 4.2071857 4.273718  4.221498  4.2194667 4.2170224
  4.126083  4.2024827 4.160259  4.162285  4.1724963 4.127249  4.1326327
  4.049733  4.144122  4.114087  4.1729307 4.190652  4.1630297 4.1940603
  4.2125483 4.213132  4.168861  4.1435814 4.205591  4.1503406 4.2193117
  4.2045135 4.1883936 4.2167654 4.196247  4.1563935 4.149891  4.144577
  4.144937  4.141251  4.138092  4.1516185 4.102146  4.105381  4.1334195
  4.1946115 4.2252393 4.2493143 4.215395  4.2024775 4.275173  4.287273
  4.270218  4.2518296 4.2671294 4.2764387 4.2481203 4.2402606 4.274661
  4.262083  4.300558  4.247294  4.283817  4.289648  4.3028345 4.23975
  4.2499    4.260034  4.2507954 4.2593594 4.274295  4.3140554 4.300373
  4.2899704 4.3349714 4.3055105 4.357376  4.349964  4.3087225 4.340069
  4.292029  4.319495  4.3459096 4.29928   4.324249  4.416234  4.418101
  4.410727  4.409057  4.399262  4.459319  4.4816    4.469992  4.488451
  4.51938   4.491698  4.4537196 4.47888   4.500422  4.475869  4.401472
  4.4108653 4.4169655 4.4334354 4.4406967 4.487427  4.507602  4.4387927
  4.4874206 4.477848  4.509491  4.540551  4.5525613 4.5286546 4.5822363
  4.6210537 4.613786  4.600302  4.608563  4.6259413 4.6196203 4.599956
  4.5942764 4.6182146 4.5786347 4.6306295 4.608999  4.63009   4.6640797
  4.6478357 4.6650667 4.6844845 4.651017  4.686619  4.65842   4.563662
  4.6094694 4.608091  4.5999537 4.67192   4.5464025 4.552743  4.557026
  4.588202  4.598175  4.499315  4.538725  4.5581946 4.5272875 4.5099397
```

```
  4.556025  4.5732994 4.5274186 4.5746765 4.5519633 4.515947  4.5173416
  4.518216  4.5252743 4.4943995 4.4940243 4.5256925 4.445374  4.5071716
  4.515895  4.3965774 4.4078045 4.4034677 4.4065776 4.414477  4.3446217
  4.3469014 4.3598323 4.359664  4.327418  4.280011  4.270406  4.2468386
  4.237871  4.27725   4.285221  4.2456603 4.275841  4.2504425 4.240871
  4.3074913 4.306092  4.264782  4.284648  4.2963524 4.2765408 4.267366
  4.265163  4.2796135 4.283333  4.303718  4.286025  4.3145046 4.323087
  4.3182206 4.267154  4.274719  4.2881145 4.26981   4.2556024 4.2984343
  4.3030696 4.2717767 4.26513   4.268382  4.3169975 4.3320813 4.293169
  4.3173876 4.2710543 4.295517  4.336199  4.273008  4.315981  4.279177
  4.2904654 4.3034215 4.254034  4.280437  4.3010435 4.3435025 4.3796315
  4.3341856 4.3674397 4.3459682 4.3371954 4.393912  4.3352966 4.3559046
  4.339404  4.401858  4.3166113 4.3522463 4.3536887 4.353239  4.2384343
  4.272703  4.281256  4.2457223 4.278132  4.223187  4.200085  4.2872314
  4.25482   4.2784843 4.277882  4.2643895 4.2423906 4.2942953 4.248582
  4.2772484 4.229235  4.22591   4.2642627 4.2519236 4.2763624 4.264406
  4.252542  4.275178  4.2455826 4.26263   4.2705383 4.2509484 4.2904196
  4.278229 ]]
<NDArray 1x505 @gpu(0)>
```

3.3 LSTM

In [12]:

```python
num_steps = 1
num_epochs, batch_size, lr, clipping_theta = 200, 50, 15, 1e-3

num_hiddens = 1024
rnn_layer = rnn.LSTM(num_hiddens)
rnn_layer.initialize(ctx=ctx)

lstm_model = RNNReg(rnn_layer, 505)
lstm_model.initialize(force_reinit=True, ctx = ctx)

#train_iter = gdata.DataLoader(gdata.ArrayDataset(featureMatrix, labelMatrix), b

train_and_predict_rnn_gluon(lstm_model, num_hiddens, train_iter, ctx, num_epochs
                            num_steps, lr, clipping_theta, batch_size)
```

```
epoch:  20 , loss:  4.412005
epoch:  40 , loss:  1.1982883
epoch:  60 , loss:  0.28241357
epoch:  80 , loss:  0.2824133
epoch:  100 , loss:  0.28240848
epoch:  120 , loss:  0.2824054
epoch:  140 , loss:  0.282399
epoch:  160 , loss:  0.28238133
epoch:  180 , loss:  0.28237864
epoch:  200 , loss:  0.2823773
```

In [13]:
```
predict = predict_rnn_gluon(testfeatureMatrix, 1, lstm_model, ctx)
predict[-1]
```

Out[13]:
```
[[4.2438455 4.274467  4.2459426 4.264615  4.2791185 4.168339  4.221478
  4.198837  4.184799  4.2064896 4.1220484 4.1182723 4.104424  4.122088
  4.1127567 4.138766  4.1382113 4.1442885 4.1504517 4.1326456 4.185283
  4.1807885 4.2040486 4.1866646 4.172534  4.2495813 4.2334733 4.2404413
  4.2253013 4.2363625 4.2378225 4.276188  4.288872  4.297225  4.2685957
  4.2896833 4.2971916 4.3015337 4.297376  4.2895894 4.307162  4.2927856
  4.295434  4.3026447 4.3052654 4.263713  4.295984  4.2808375 4.2877936
  4.309084  4.2891555 4.284462  4.3026967 4.288589  4.25309   4.279607
  4.2804976 4.253899  4.281111  4.2849045 4.3891554 4.391986  4.3770847
  4.3900948 4.363426  4.3179493 4.33246   4.311776  4.34629   4.335685
  4.3384824 4.3510237 4.3620086 4.3564305 4.3654556 4.378461  4.4030676
  4.4086905 4.409833  4.416149  4.3993716 4.371451  4.4016895 4.397653
  4.398169  4.378773  4.335505  4.374634  4.364743  4.3745823 4.355263
  4.345561  4.372878  4.3546715 4.3378325 4.4449377 4.435758  4.451446
  4.4389873 4.420121  4.3804135 4.392227  4.3935814 4.393215  4.401109
  4.4134827 4.4027476 4.415333  4.4418497 4.4290543 4.3926406 4.382882
  4.3889594 4.39587   4.4143095 4.3857446 4.359542  4.376716  4.3824058
  4.3967013 4.367175  4.3514256 4.3586497 4.3458548 4.3641243 4.3308268
  4.343882  4.3344483 4.3315043 4.3526235 4.2907987 4.3002124 4.3007355
  4.2941656 4.2958207 4.3239703 4.3441677 4.3493056 4.338006  4.3446817
  4.366202  4.389724  4.34592   4.34488   4.3670774 4.265917  4.3017197
  4.3185496 4.2795353 4.277172  4.3166475 4.316505  4.345812  4.3012567
  4.3264213 4.325998  4.362139  4.3032703 4.308531  4.3163157 4.326782
  4.328915  4.3058357 4.3266287 4.3142333 4.3953166 4.38451   4.388489
  4.3663244 4.3857017 4.3456516 4.3525896 4.374156  4.3635807 4.3886194
  4.400458  4.3485255 4.379867  4.361095  4.3632603 4.291196  4.2932806
  4.276087  4.3023143 4.3106227 4.3248873 4.301866  4.3066425 4.294186
  4.29423   4.252079  4.277564  4.276392  4.2688193 4.263927  4.267557
  4.2493076 4.2441545 4.2568965 4.2875357 4.3151712 4.325887  4.3043914
  4.3200645 4.311364  4.2704034 4.292536  4.2739425 4.290399  4.289927
  4.2185893 4.242008  4.2218113 4.229557  4.243215  4.1978135 4.2097254
  4.1969624 4.1875978 4.2307262 4.2016478 4.205594  4.2181196 4.209203
  4.2077956 4.1960945 4.210072  4.190845  4.184884  4.1760883 4.177627
  4.1763153 4.205182  4.1713624 4.1917214 4.190166  4.163738  4.1727357
  4.157854  4.1767516 4.1788516 4.152583  4.1593904 4.16985   4.17976
  4.0736713 4.0835547 4.065131  4.0963326 4.1121783 4.054647  4.0619636
  4.066217  4.0448627 4.0606756 4.0950136 4.1136475 4.1107736 4.113361
  4.124551  4.1062756 4.1156006 4.131424  4.1169033 4.1175365 4.1144676
  4.0963726 4.1250105 4.138285  4.140837  4.038892  4.075515  4.052133
  4.068308  4.060719  4.08781   4.0729165 4.069266  4.0665717 4.0765243
  4.145517  4.142407  4.150461  4.1475854 4.1391544 4.2154984 4.1931252
  4.2125278 4.2050433 4.2275095 4.18909   4.206211  4.1974797 4.2040873
  4.177966  4.2233105 4.227685  4.210373  4.238731  4.222957  4.176727
  4.190519  4.1772976 4.1811953 4.1901693 4.2364264 4.2334833 4.236706
  4.239136  4.260058  4.238448  4.254856  4.272894  4.2672133 4.2625384
  4.240966  4.240567  4.248196  4.2329288 4.209001  4.3681717 4.330826
  4.342791  4.3313165 4.342431  4.422348  4.41257   4.410739  4.387272
  4.4116116 4.4016542 4.3969717 4.3995266 4.402804  4.4139376 4.345274
  4.352838  4.3509398 4.3318644 4.3357034 4.3992915 4.415233  4.4097686
  4.4024134 4.425909  4.4687138 4.4580736 4.459864  4.45498   4.476637
  4.5070252 4.5267777 4.536192  4.493335  4.5050287 4.549573  4.5391874
  4.521272  4.545942  4.4940844 4.568481  4.567374  4.569069  4.5728335
  4.5451303 4.596632  4.5643477 4.5902095 4.59303   4.575482  4.52638
  4.5412836 4.5606604 4.5235734 4.521837  4.5097322 4.505832  4.4833865
  4.495521  4.48826   4.453995  4.4593406 4.4399796 4.4327636 4.4514933
```

```
  4.482356  4.4948907 4.501989  4.51223   4.4871254 4.4360275 4.419601
  4.4255767 4.417349  4.435082  4.4191823 4.423809  4.38016   4.4062195
  4.397347  4.32023   4.3325863 4.3258357 4.3394976 4.353879  4.287507
  4.2651405 4.290233  4.291818  4.2781    4.199162  4.207681  4.2015533
  4.1756697 4.1982207 4.206421  4.2092795 4.233303  4.1700606 4.206326
  4.2219224 4.216819  4.212736  4.2156453 4.215252  4.2321324 4.2007966
  4.2270465 4.2345247 4.2110186 4.2367134 4.2294106 4.2280684 4.2361336
  4.2415113 4.211003  4.1940346 4.2025867 4.2051935 4.231733  4.1963263
  4.1909485 4.184857  4.171513  4.214905  4.2093573 4.242005  4.2302146
  4.2491984 4.235461  4.2404566 4.2228584 4.225548  4.264985  4.2479553
  4.220985  4.205606  4.2058644 4.223909  4.217893  4.284025  4.2662096
  4.280469  4.2839675 4.2737775 4.293958  4.282961  4.278727  4.289611
  4.2855854 4.2767587 4.2922173 4.303302  4.280327  4.2973204 4.209307
  4.199556  4.2169023 4.191009  4.1974764 4.1703987 4.2055917 4.174386
  4.1967936 4.1805553 4.163545  4.2013526 4.19239   4.1751714 4.1684294
  4.181634  4.1864705 4.2056646 4.177148  4.1955237 4.1795354 4.2000904
  4.191513  4.2025604 4.193453  4.2088103 4.1692204 4.178996  4.1847167
  4.1885004]]
<NDArray 1x505 @gpu(0)>
```

3.4 Two-layer LSTM

In [19]:

```
num_steps = 1
num_epochs, batch_size, lr, clipping_theta = 220, 50, 10, 2e-3

num_hiddens = 1024
rnn_layer = rnn.LSTM(num_hiddens,2)
rnn_layer.initialize(ctx=ctx)

lstm2_model = RNNReg(rnn_layer, 505)
lstm2_model.initialize(force_reinit=True, ctx = ctx)

#train_iter = gdata.DataLoader(gdata.ArrayDataset(featureMatrix, labelMatrix), b

train_and_predict_rnn_gluon(lstm2_model, num_hiddens, train_iter, ctx, num_epoch
                            num_steps, lr, clipping_theta, batch_size)
```

```
epoch:  20 , loss:  5.6780076
epoch:  40 , loss:  0.81478447
epoch:  60 , loss:  0.28883514
epoch:  80 , loss:  0.28445688
epoch:  100 , loss:  0.2831436
epoch:  120 , loss:  0.28266755
epoch:  140 , loss:  0.28249323
epoch:  160 , loss:  0.28242385
epoch:  180 , loss:  0.28239024
epoch:  200 , loss:  0.2823689
epoch:  220 , loss:  0.28238726
```

In [20]:
```
predict = predict_rnn_gluon(testfeatureMatrix, 1, lstm2_model, ctx)
predict[-1]
```

Out[20]:
```
[[4.2803154 4.2838435 4.281447  4.277385  4.275619  4.21211   4.209841
  4.2117643 4.2139306 4.2110105 4.125894  4.1299963 4.1267133 4.1288676
  4.1340437 4.1670766 4.1668787 4.1687274 4.169542  4.1671405 4.210121
  4.210006  4.211241  4.2082953 4.2116776 4.2748413 4.273813  4.270367
  4.273207  4.2663116 4.2929616 4.2980046 4.2934504 4.2961345 4.2966986
  4.3162746 4.317229  4.319422  4.318446  4.3178735 4.318786  4.3219347
  4.3180103 4.3225975 4.324579  4.311994  4.3121943 4.311072  4.310684
  4.3093233 4.3093567 4.309365  4.3091054 4.3098235 4.306874  4.2983828
  4.3009906 4.3079147 4.306141  4.3073716 4.410345  4.4118295 4.412899
  4.4182134 4.416949  4.3570075 4.3592167 4.360351  4.3640633 4.3618956
  4.3832393 4.3845425 4.3834934 4.3837276 4.3903403 4.4290113 4.4360294
  4.4332266 4.43539   4.440132  4.4282403 4.430736  4.430102  4.4294133
  4.4331684 4.3969574 4.3964357 4.3958383 4.3952518 4.395268  4.379858
  4.378283  4.3789544 4.3785996 4.3756824 4.455346  4.4533043 4.450044
  4.4547925 4.4550447 4.4108515 4.4085097 4.413085  4.4083486 4.4078984
  4.4380307 4.4393125 4.4432306 4.4465723 4.443863  4.410701  4.4089413
  4.410658  4.4122334 4.411612  4.377264  4.3805685 4.379881  4.3825555
  4.386321  4.3707848 4.36966   4.369224  4.3712316 4.3728685 4.3518596
  4.355541  4.3540616 4.3549232 4.350564  4.324331  4.32426   4.3193803
  4.324423  4.324938  4.3682275 4.369385  4.3716297 4.3732157 4.3739595
  4.3898954 4.391363  4.3864264 4.3881702 4.3849773 4.3073554 4.3085217
  4.3078365 4.309337  4.3109293 4.3451705 4.3445263 4.3449917 4.3460684
  4.3489347 4.3354516 4.336334  4.3358083 4.3403263 4.338747  4.332918
  4.3338394 4.333496  4.3380785 4.3316717 4.400935  4.4045725 4.406672
  4.405376  4.41384   4.394642  4.3918953 4.39207   4.394408  4.3967566
  4.3945975 4.396293  4.392731  4.3917985 4.398758  4.322321  4.327335
  4.3232327 4.326961  4.330167  4.326178  4.3240395 4.321996  4.321342
  4.3238444 4.281791  4.28045   4.2768354 4.279887  4.280954  4.271511
  4.274669  4.2730417 4.2677455 4.2811174 4.3331313 4.3374043 4.331034
  4.3341193 4.332782  4.2902555 4.2932816 4.29023   4.293678  4.295909
  4.2494583 4.2526526 4.249811  4.250879  4.256983  4.218436  4.218908
  4.216089  4.2164    4.2198534 4.223873  4.222318  4.2276864 4.2312865
  4.2293983 4.209631  4.210266  4.209247  4.210571  4.209622  4.205719
  4.2099967 4.2095714 4.2121606 4.2119813 4.19279   4.19346   4.192679
  4.191978  4.1940556 4.174383  4.174405  4.1688695 4.1757555 4.1773767
  4.0994205 4.106071  4.096457  4.098582  4.099479  4.0694838 4.070842
  4.0682697 4.07304   4.0695596 4.122806  4.1214595 4.11901   4.1234407
  4.1259108 4.128106  4.127307  4.1336985 4.1353793 4.126786  4.129661
  4.1323595 4.1308846 4.1345887 4.133814  4.0920587 4.092094  4.0908766
  4.0894074 4.0928273 4.0898476 4.0909195 4.0917325 4.0946746 4.0994654
  4.157141  4.1576085 4.1568046 4.1562734 4.1650033 4.228477  4.226396
  4.22863   4.226375  4.2243595 4.222927  4.223315  4.220466  4.218888
  4.2195253 4.2408733 4.2416472 4.240757  4.241437  4.245482  4.201901
  4.202007  4.2062364 4.209494  4.209724  4.253896  4.2547092 4.255419
  4.255493  4.259062  4.281565  4.2803497 4.2836237 4.280068  4.2826533
  4.2528076 4.25067   4.25231   4.2504067 4.2548323 4.3540273 4.358621
  4.353615  4.35567   4.35664   4.424443  4.425565  4.425471  4.4272075
  4.4280424 4.415714  4.41731   4.414997  4.4173717 4.4194484 4.36391
  4.3630867 4.3618574 4.362056  4.3672714 4.4188128 4.421574  4.423238
  4.4246984 4.4250393 4.4806514 4.4781365 4.476179  4.48457   4.4806185
  4.5363293 4.5340395 4.5341663 4.5330625 4.5342793 4.540149  4.538357
  4.536092  4.538226  4.541631  4.5759754 4.5758953 4.5729346 4.5800548
  4.5774264 4.598746  4.599602  4.6001062 4.600696  4.6047235 4.5396447
  4.5370307 4.536611  4.5377207 4.537869  4.4933734 4.493862  4.4981284
  4.4960737 4.4995174 4.457005  4.4567866 4.4590364 4.4580913 4.4531345
```

```
  4.4950657 4.500629  4.499945  4.5038886 4.502917  4.4522734 4.4561424
  4.452754  4.454986  4.4562674 4.426538  4.426088  4.42424   4.4267874
  4.4240403 4.353928  4.355773  4.3515368 4.356619  4.3624883 4.297899
  4.3010902 4.299578  4.3033633 4.301741  4.219605  4.222964  4.218736
  4.2225575 4.2182183 4.236197  4.2387643 4.2359443 4.2316217 4.2344227
  4.2532024 4.2534146 4.2501206 4.2523136 4.251331  4.246197  4.245571
  4.247482  4.2501593 4.25414   4.264865  4.2624364 4.2630553 4.260697
  4.266658  4.2270474 4.2272143 4.227782  4.2226667 4.2262936 4.2096
  4.2207413 4.2153454 4.223518  4.224197  4.244844  4.250715  4.2490726
  4.249508  4.251171  4.257187  4.2584853 4.2577367 4.2603307 4.2602024
  4.226914  4.223221  4.2235394 4.230239  4.230591  4.2913036 4.285883
  4.2863626 4.286617  4.2877254 4.291364  4.290101  4.2943535 4.2935815
  4.2954197 4.2989116 4.2998033 4.3009596 4.299134  4.301496  4.213179
  4.209905  4.208606  4.216629  4.2097187 4.188484  4.1890297 4.18697
  4.1877513 4.186573  4.199343  4.2042136 4.1996627 4.19706   4.1968884
  4.1951094 4.19606   4.197191  4.19893   4.2042866 4.2173862 4.218455
  4.2176585 4.219947  4.2198124 4.214702  4.2166147 4.2152133 4.212341
  4.2086916]]
<NDArray 1x505 @gpu(0)>
```

In [ ]: