**homework HW1 Math6373 due date thursday feb 10th at midnight**

Data set:

use four years of daily market data; download 7 daily closing prices of

**Gold, Platinum, Silver, DowJones, Euro, Yen , Renminbi**,

Note: Renminbi = Yuan

On day “t”: V(t) = line vector of 7 prices = [V1(t) ... V7(t)]

V1(t) = price of Gold on day t =Gold(t) ... V2(t) = price of platinum on day t =Platinum(t)

the four years data set contains N actual days

replace calendar dates by index t=1,2,3 ... N

N= total number of days in whole data set

X(t) = feature vector has dimension 5x7 =35

X(t) = long line vector [ V(t), V(t-1), V(t-2), V(t-3), V(t-4)]

case # t is INITIALLY described by feature vector Xt

Goal: construct an MLP to predict (on day t) the future gold price Z(t) = V1(t+1) = Gold(t+1)

data set = { X(1) , X(2) , … X(N) } cases observed over 4 years

true value Z(t) is known on the data set for all t <= N-1

Q0

for each j = 1... 7 compute Mj= mean over all t of the values Vj(t)

Mj = (1/N)\*(Vj(1) + ... + Vj(N)

for j=1 .. 6 construct the graph displaying both Vj(t)/Mj and V7(t)/M7(t)

Visual interpretation?

Q1

replace each price Vj(t) by rate of return rVj(t) = [ Vj(t) - Vj(t-1) ] / Vj(t-1)]

replace Z(t) by rZ(t) = [Z(t) - Z(t-1)] / [Z(t-1) = [ V1(t+1) - V1(t) ] / V1(t)]

replace X(t) by rX(t) = [ rV(t), rV(t-1), rV(t-2), rV(t-3), rV(t-4)]

for case # t, the **new feature vector is rX(t),**

the **true target variable to be predicted is rZ(t)**

compute mean.rZ = average of the N absolute values rZ(t) , namely

mean.rZ = (1/N) \*( | rZ(1) | + ... + | rZ(N) | )

on one single graph display the 7 curves rV1(t) ... rV7(t)

Visual interpretation?

explain how a good prediction of rZ(t) on day t will easily provide

a good prediction on day t for Z(t) = Gold(t+1)

Q2

define the first attempted architecture of your MLP with 3 layers as follows

Input layer L1🡺 hidden layer L2 🡺 Output layer L3

size L1 = 35 ; size L2 = h ; size L3 = 1;

The integer h will be finalized below

denote param(h) the total # of weights and thresholds in this MLP

give a formula for param(h)

Q3

randomly select 80% of all cases as your training set; display TRN = size of training set

the remaining 20% of cases will be the test set

apply the parsimony principle : impose param(h) < # informations brought by the training set

compute the maximum value h\* of h , derived from this parsimony principle

Q4

fix 2 possible values for h namely h1= h\* and h2 = 3 h\*

note that h2 does not verify the parsimony principle

for each such value of h, launch the automatic learning of your MLP

you will need to select (and report your choices)

the type of response function( RELU is suggested)

the type of initialization of the weights and thresholds (default random choices in tensorflow} the type of gradient descent optimizer ( Adams is a good generic choice)

the Batch Size BATS ( try 4 possible BATS values : TRN/40, TRN/20, TRN/10, TRN/2 )

the type of loss function (MSE)

the **total number of epochs TOTEP** (suggestion : at least 100 or 150 epochs)

the criterion used to stop the automatic learning (explain the basic choices in tensorflow)

for each of the 8 choices of the pair (h , BATS)

display the computing time necessary for automatic learning

display the total number numBATS of batches

display the terminal value trainMSE of MSE on the whole training set

display the the curve MSE(m)

give a comparative interpretation of these results

Q5

Monitoring of EACH one of of the eight automatic learning

**for m =1,2, ...,TOTEP after each epoch # m**

compute trainMSE(m) on the whole training set and testMSE(m) on the whole test set

display the two curves trainMSE(m) and testMSE(m) versus m

compute and display the curve ||grad MSE(m) || / sqrt(param(h))

Q6

for each one of the eight automatic learning and for each epoch # **m =1,2, ...,TOTEP**

compute the two normalized accuracy curves

trainAcc(m) = sqrt(trainMSE(m))/ mean.rZ

testAcc(m) = sqrt(testMSE(m)) / mean.rZ

recall that mean.rZ is the mean of all absolute vales |rZ(t)|

on the same graph display the two curves { trainAcc(m) and testAcc(m) versus m}

interpret these results for each automatic learning ;

check if and when there is overfit;

comment the behaviour of the || gradMSE(m) || versus m

for each learning, determine an **optimal stopping epoch index** **m\*** and the corresponding optimal values obtained for trainAcc(m\*) , testAcc(m\*)

you shoul try to find m\* minimizing testAcc(m) but also

make sure that there is no overfit at m\*

Q7

use your preceding analyzis to determine the best pair (h , BATS), and the corresponding best weights Wij + thresholds Bi reached at optimal stopping epoch m\*

display the histogram of all |weights|= |Wij| linking neuron j of L1 to neuron i of L2

identify the 10 smallest and the 10 largest |Wij|

display the histogram of all |weights| = |m(i,1)| linking neuron i of L2 to neuron 1 of L3

rank the |m(i,1)| in increasing order and display this increasing curve

Q8

Most Influential Hidden Neurons

identify the neuron i\*in L2 such that |m(i\*,1)| > all |m(i,1)|

this neuron is strongly influential on the output

Q9

most influential explanatory variables

the neuron i\* is connected to 35 inputs by weights W(i\*,1) ...W(i\*, 35)

for each neuron j in L1, compute average impact of input(j) on neuron i\* by

impact(j on i\*)) = | W(i\*,j) | x mean| input(j) | where

mean | input(j) | = average value of | input(j) | over all cases

rank the impacts(j,i\*) in increasing order and display these 35 ordered values

identify the 2 explanatory variables which have the highest influence on neuron i\*

identify the 2 explanatory variables which have the lowest influence on neuron i\*

conclusions?

Q10

your suggestions to improve the architecture of the MLP for better testAcc, trinAcc?

try at least one of your suggestions