HW3

Shiyu Liu

Exercise 1.1

1.

I re-use the code “clocksync_nnet.py” from HW2, with some small modifications (add a switch of turning on/off the coded packet filtering).

Without coded pkt filtering, the estimation error:

Drift: mean = 12.4 ns/sec, std = 290.1 ns/sec

Offset: mean = -18.8 ns, std = 334.4 ns

2.

With coded pkt filtering, the estimation error:

Drift: mean = 3.2 ns/sec, std = 36.8 ns/sec

Offset: mean = -4.3 ns, std = 59.9 ns

The errors are much smaller than the estimation without coded packet filtering.

Exercise 1.2

1.

The codes for constructing loop matrix A are in “loopwiseVerify.py”

2.

The codes are in “loopwiseVerify.py” (building loop matrix A and estimating errors) and “compareError.py” (calculate errors after correction).

For the estimated discrepancy vector, $\|\Delta^P - \Delta\|_2$ = 42.0ns

For the corrected discrepancy vector, $\|\Delta^F - \Delta\|_2$ = 13.3ns

The error is reduced by 68.4%, which is very close to the theoretical results.

Exercise 2.1

The NS3 simulation has been run, and data are generated.

Because Exercise 2.2 (SVD), 2.4 (NN) require a large amount of data, I simulated a smaller network: 2-stage 16 servers (4 servers/rack * 4 racks). It's a 1Gbps network, K = 10, reconstruction period = 10ms.

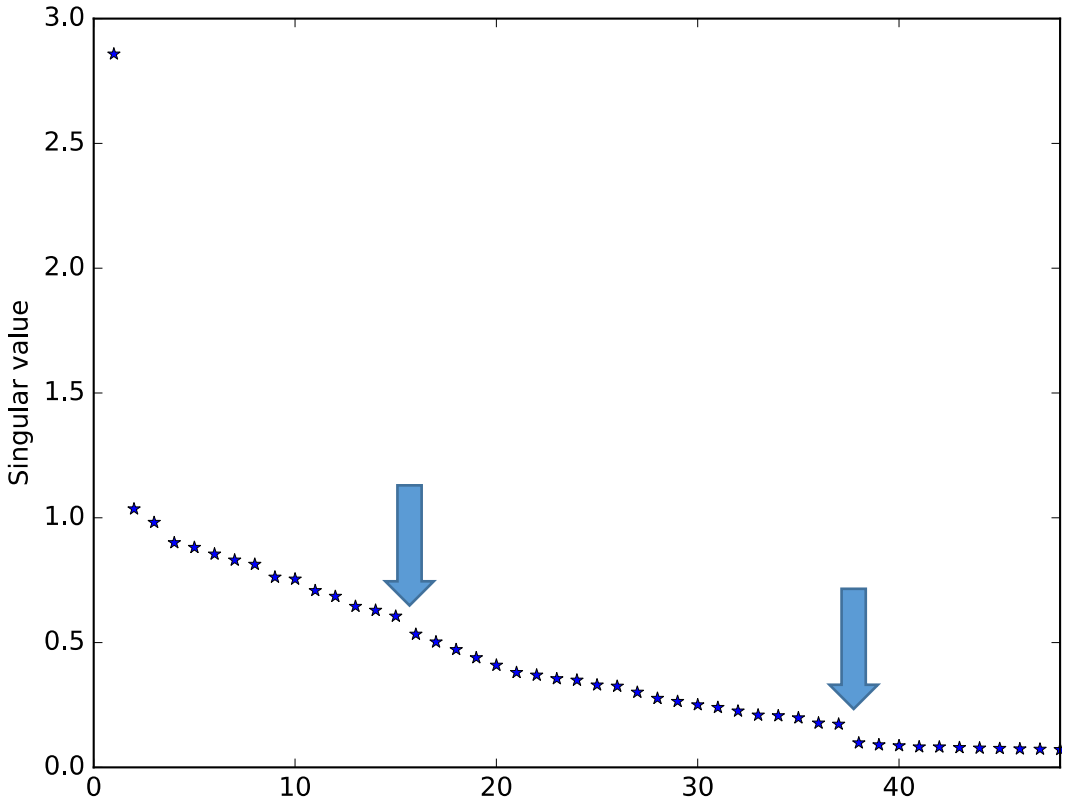
I made 12 runs, 20 sec per run, therefore 240 sec data in total, i.e. 24000 reconstruction intervals.

The codes are in "NS-3-Ex2.1-2.4.tar.gz".

When running data-sim-queuelength.py, the parameters are 4 4 4 1 0

Exercise 2.2

The codes are in "HW3_SVD.py". The singular values of the delay matrix M is shown below:



From the plot, we can see 2 jumps in the sequence of singular values (where the blue arrows point to). So we can tell that the network has 2 stages.

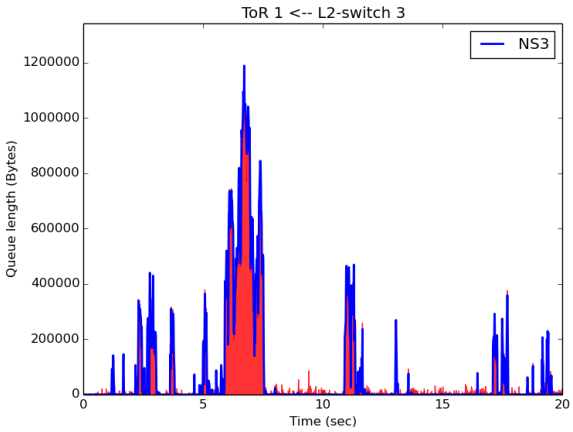
From the plot, it's easy to find that there are 37 non-zero singular values, so there are 37 independent paths in this 2-stage 16-server network.

Exercise 2.3

The LASSO recon codes are in “recon.py”. The comparison with ground truth and plotting are in codes “recon-plot.py”.

The RMSE is 10.4KB, the relative error is 8.3%.

One example plot of recon results & ground truth is shown below. The red part is the recon results, the blue line is NS3 ground truth.



Exercise 2.4

“HW3_getDataForNN.py” reads NS3 data and convert to delay matrix and queue matrix, as the input data of NN.

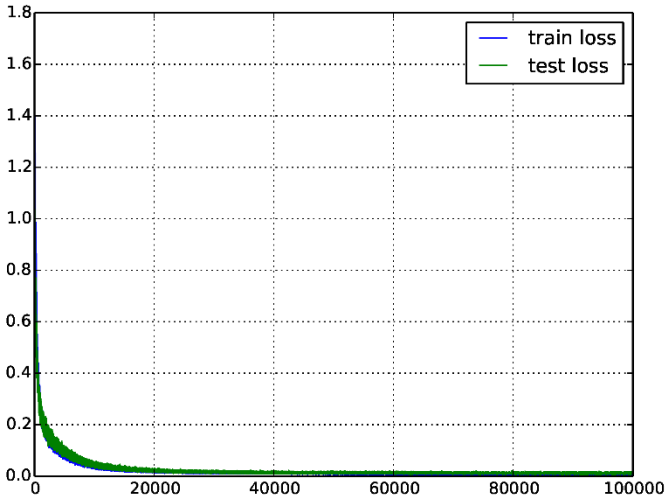
“queue_recon_nnet.py” construct NN and do learning & predicting.

The NN I implemented has an input size of 141 (141 probes in each interval) and an output size of 48 (48 queues in total). It has only 1 hidden layer with 96 neurons.

The data include 23976 samples, and are split into 2 parts: 80% are training data (19180 samples), 20% are testing data (4796 samples).

In each iteration, the NN is trained by 1 mini-batch with 100 training samples. Then it is tested by 100 testing samples. After 100000 iterations, I test the NN with all testing data, to get the final RMSE.

The training and testing errors vs. iterations are shown below:



After training, the final test gives RMSE = 10.5 Kbytes, and relative error = 10.9%. The following figures are the reconstructed length & ground truth of 3 queues. As we can see, NN gives pretty good reconstruction quality which is close to LASSO. By increasing the size of training set, and training for more iterations to make it fully converged, hopefully we could get lower RMSE and beat LASSO.