An investigation of multivariate adaptive estimation-based change detection for NETFLOW data

Dean Bodenham
Imperial College London

Abstract

•The utility of a multivariate adaptive forgetting factor-based change detection framework is explored for NETFLOW data

•NETFLOW data represents TCP flows between source and destination network devices

•Framework is online, to handle the large volume/high frequency data

•Proposed approach monitors the volume of traffic flowing through various ports, and attempts to detect a significant change

•Results are presented for real data obtained from monitoring Imperial College's internal computer network

•Monitoring for malicious network behaviour is an important security application

Methods

Suppose we have a data stream $x_1, x_2, x_3 \dots$

and we define $$\bar{x}_N = \frac{1}{N}\sum_{k=1}^{N} x_k$$

In this sum, each observation is given equal weight (namely, 1/*N*). However, we would like to place more weight on more recent observations, and less weight on less recent observations, since this better reflects the current regime of the data stream.

We introduce an exponential **forgetting factor** $\lambda \in [0,1]$

and define the **forgetting factor mean to be** $$\bar{x}_{N,\lambda} = \frac{1}{w_{N,\lambda}}\sum_{k=1}^{N} \lambda^{N-k} x_k$$

Our forgetting factor can be **fixed** (FFF) or **adaptive** (AFF).

If adaptive, we choose a function that we wish to minimise, e.g.

$$L_{N,\vec{\lambda}} = \left[\bar{x}_{N-1,\vec{\lambda}} - \bar{x_N}\right]^2$$

and we update its value after each observation using gradient descent,

$$\lambda_{N+1} = \lambda_N - \eta \frac{\partial}{\partial \vec{\lambda}} L_{N,\vec{\lambda}}$$

If we knew the parameters of the underlying distributions, we could obtain the distribution of the forgetting factor mean and then obtain a p-value for each realisation at each data point.

However, this information is seldom available. In this proposal, we monitor the stream using both a FFF and an AFF. We choose the FFF to be close to 1, so that it will react slowly to changes in the data. The AFF will react faster to changes in the data. We define a **confidence interval** for our AFF to be

$$\left(AFF - c\,FFF\,,\, AFF + c\,FFF\right)$$

for a value *c*. We call this the **FFAFF** scheme.

Results

A sample of our data is as follows:

Date flow start,	Duration,	Proto,	Src IP Addr,	Dst IP Addr,	Src Pt,	Dst Pt,	Packets	,Bytes
2009-04-22 12:03:02.017,	259.39,	TCP,	100.253.214.75,	62.136.125.167,	80,	48696,	1115,	633058
2009-04-22 12:04:36.404,	17.60,	TCP,	100.253.210.138,	100.253.192.99,	1104,	80,	6017,	276899
2009-04-22 12:04:44.664,	13.63,	TCP,	126.253.5.69,	124.195.12.246,	49882,	80,	1507,	2202009

We monitor the traffic of the five source ports with the most data flows. Simply counting the flows in a particular time period leads to **Figure 1**. We have binned approximately 14 days worth of the data into 100-minute time intervals. Note how port 80 has significantly more traffic than the other ports.

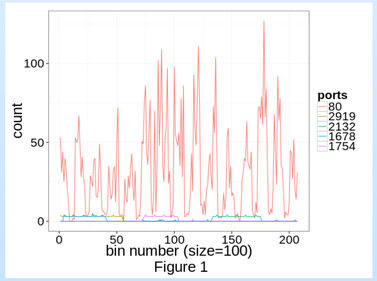

Figure 1

It would perhaps be more informative to monitor the volume of information flowing between the ports, and this doing leads us to **Figure 2**. Again, port 80 dominates the other ports, which can be seen more clearly in **Figure 3**. These pictures show that there are changes in all the ports, most notably in port 80 around bin 100.

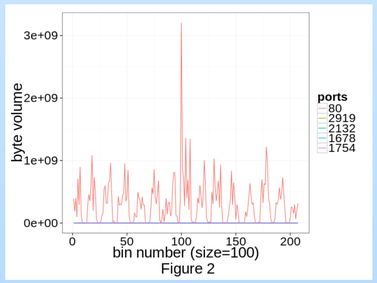

Figure 2

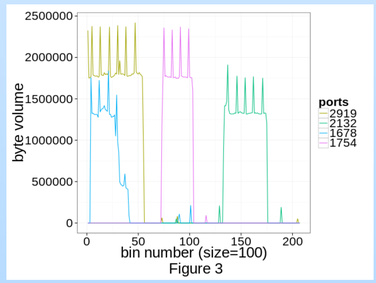

Figure 3

We monitor the port 80 bins using the FFAFF scheme, and arrive at **Figure 4**. The vertical red dashed line shows the time a change was detected. The vertical green dashed lines signify the end of a short burn in period that calibrates the AFF, and change detection starts. **Figure 5** shows the value of the AFF over time. Note how it drops sharply before a change and then "recovers" when the stream recovers.

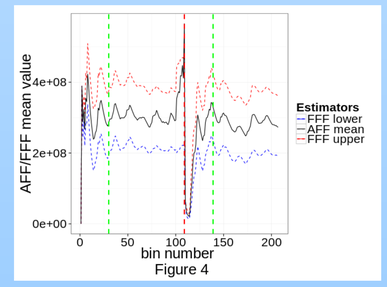

Figure 4

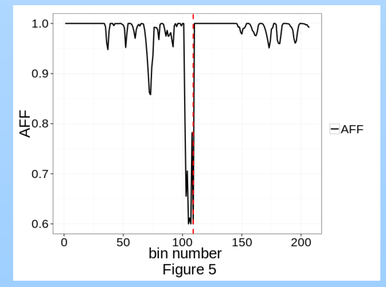

Figure 5

Multivariate setting

The FFAFF scheme can be easily extended to the multivariate setting:

1) Each stream is monitored by its own FFAFF scheme

2) After each observation/bin, a p-value can be calculated for each stream

3) These p-values can be combined using Fisher's method

4) A (weighted) version of Stouffer's method could also be used

One of the benefits to this approach is that each stream is evaluated on its "own scale", and a change in any stream will be significant, regardless of the stream's scale. For instance, Figure 2 shows that there is one large change in port 80, while the other ports seem insignificant. However, Figure 3 shows that the other ports have changes near bins 50, 60, 140, etc. In the multivariate FFAFF, these changes would be recognised.

Conclusion

The FFAFF framework provides a possible method for monitoring netflow data.

Advantages:

- Fast and efficient implementation
- Distribution free, and streams can be on different scales
- Possible extensions to the multivariate case
- Continuous monitoring is easily implemented

Future work:

- Deciding how to set the confidence interval width
- Extending the implementation of the scheme to a graph (network)

References

D. Bodenham, N. Adams, Continuous changepoint monitoring of data streams using adaptive estimation (submitted 2012)

S.W. Roberts, (1959) Control chart tests based on geometric moving averages, *Technometrics*, **1** 239-250

M. Whitlock, (2005) Combining probability from independent tests: the weighted Z-method is superior to Fisher's approach, *Journal of Evolutionary Biology*, **18** 1368-1373

Acknowledgements

I would like to thank my supervisors, Niall Adams and Nicholas Heard.

This work was fully support by a ROTH studentship provided by the Department of Mathematics, Imperial College London.

NETFLOW data was provided by Imperial's ICT department.