Adaptive Anomaly Detection System for Cloud Computing Infrastructures

Husanbir Singh Pannu, Jianguo Liu (Department of Mathematics), Song Fu (Department of Computer Science & Engineering)
University of North Texas, Denton, Texas 76203-5017

Motivation

Networked computer systems continue to grow in scale and in the complexity of their components and interactions. In these systems, abnormalities become norms instead of exceptions.

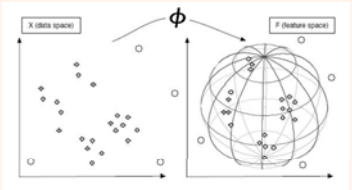
These require anomaly management tasks have significantly higher levels of automation.

Anomaly prediction is a crucial technique for understanding emergent, system-wide phenomena and self-managing resource burdens. Based on the analysis of failure data in a system, a failure predictor aims to determine possible occurrences of fatal events in the near future and help develop more effective failure tolerant solutions for improving system availability.

Adaptive Anomaly Detection Mechanism

We use 1-SVM to describe cloud performance data, detect anomalies and adapt the detector when verified detection results are available. The reduced dimensional initial normal mode data is enclosed within the minimum enclosing sphere in the Gaussian kernel space. Such sphere can be found by solving

$$\min R^2 + A\sum_i \xi_i,$$
$$\text{subject to} \quad \|\phi(x_i) - o\| \le R^2 + \xi_i \text{ and } \xi_i \ge 0,\ i = 1, 2, \ldots$$



Get dual by using Lagrange variables αi

$$\max \sum_i \alpha_i K(x_i, x_i) - \sum_i \sum_j \alpha_i \alpha_j K(x_i, x_j),$$
$$\text{subject to } \sum_i \alpha_i = 1 \text{ and } 0 \le \alpha_i \le A,\ i = 1, 2, \ldots$$

AAD Algorithm

The vectors which lie on the surface of hyper-sphere are called Support Vectors. Only they decide the center and radius of the sphere.

Condition	Details
$0 < \alpha_i < A$	On the surface of hypersphere
$\alpha_i = A$	Outside the hypersphere
$\alpha_i = 0$	Inside the hypersphere

Formulas for center and radius are the following. Φ is the mapping function from original space to Gaussian kernel space

$$o = \sum_i \alpha_i \phi(x_i),$$
$$R^2 = \|\phi(x_i) - o\|^2 = K(x_i, x_i) - 2\sum_j^n \alpha_j K(x_j, x_i) + \sum_j^n \sum_l^n \alpha_j \alpha_l K(x_j, x_l)$$

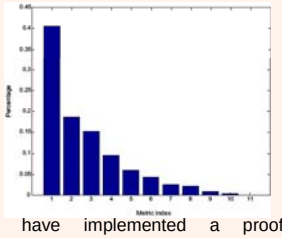
Algorithm

AFDFailureDetection() {

1: X = cloud performance dataset;
2: $q = \frac{-1}{2\sigma^2}$; //Initialize kernel width
3: $A = \frac{1}{n}$; //Initialize soft constraint penalty
4: α = solution to *Dual*(X, q, A);
5: $o = \sum_{x_i \in X} \alpha_i \phi(x_i)$; //Center of the hypersphere
6: $R^2 = \|\phi(x_i) - o\|^2$; //Radius of the hypersphere
7: **while** (TRUE) **do**
8: *On receipt of a cloud performance data record* x_i
9: **if** $\|x_i - o\|^2 > R^2$ **then**
10: report a possible failure with performance states x_i;
11: **end if**
12: *On receipt of a verified detection or an observed but undetected failure* f_j
13: **if** (f_j is *normal* AND $\|\phi(f_j) - o\|^2 > R^2$) OR ($f_j$ is a *failure* AND $\|\phi(f_j) - o\|^2 < R^2$) **then**
14: $q = q + \delta$; //Adapt q
15: $A = A + \Delta$; //Adapt A
16: α = solution to *Dual*(X, q, A);
17: update the center o and radius R;
18: **end if**
19: **end while**
20: }

q and A are updated by δ and Δ respectively to adapt the hyper-sphere to cover most of the normal data and exclude anomalies outside. These values are tuned at runtime to achieve a high ROC slope for failure detection.

Experimental Results



We have implemented a proof-of-concept prototype of AFD system and tested it in a cloud computing environment on campus consisting of 362 servers connected by gigabit Ethernet. Fault injection program is developed inject 4 major and 17 minor faults to mimic CPU, memory, disk & network faults. In total, 518 metrics are profiled 10 times per hour for one month and about 601.4 GB data were collected from the cloud. 112 them display zero variance and we further apply ICA and reduce dimensions to 3 which capture 81.3% of the total variance.

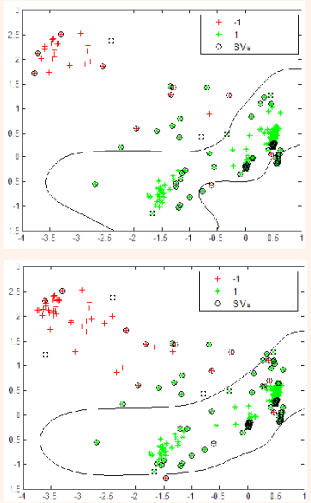
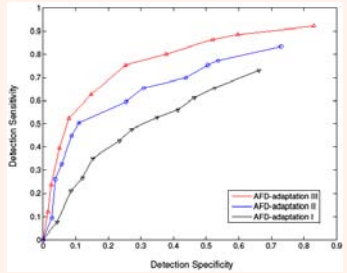


Fig show the adaptations of data description contours for anomaly detection during self evolvement. Less than 1% of the normal data points are plotted for better readability. As more verified failure detection results are exploited, the contours become tighter which improves the accuracy of failure detection.



Detection sensitivity and specificity are defined as:
Sensitivity = detected failures / total failures
Specificity = detected normal / total normal

The above figure depicts the detection performance after three selected rounds of adaptation. Both detection sensitivity and specificity improves as AAD detector adapts. After adaptation III is applied, the detector achieves 92.1% sensitivity and 83.8% specificity. The results indicate that AAD is well suitable for building dependable cloud computing systems.

Comparison with other advanced methods: Subspace regularization which is better than other smoothness based methods (Gaussian Random Field, Manifold Regularization) achieves 67.8% sensitivity. Bayesian sub-models and decision tree classifier could got 72.5% sensitivity.

Conclusion

We presented Adaptive Anomaly Detection framework with mechanisms which does not require prior failure history and can self adapt by learning from observed anomalies at runtime. We plan to explore other advanced statistical learning methods along with proactive and reactive approaches to achieve even higher cloud dependability.

Selected Publications

[1] H. S. Pannu, J. Liu, S. Fu : Self-evolving anomaly detection for developing highly dependable utility clouds, *GLOBECOM 2012*.

[2] H. S. Pannu, J. Liu, S. Fu : Adaptive failure detection system for cloud computing infrastructures, *IEEE PCCC 2012*.

[3] H. S. Pannu, J. Liu, S. Fu : A hybrid anomaly detection framework in cloud computing using 1 and 2 SVM, *ADMA 2012*.