Eigenspace-based anomaly detection in computer systems

IBM Research, Tokyo Research Laboratory
Tsuyoshi Ide and Hisashi Kashima
Outline

- Motivation
- Modeling Web-based systems
- Problem statement
- Feature extraction
- Anomaly detection
- Experiment
- Summary
Motivation: Fault detection in computer systems at the application layer

- Faults at the *application layer* are hard to detect using existing technologies
  - This is especially true for Web-based systems with redundancy

- Why?
  - The *dependencies* between servers make everything complicated
  - They are highly *dynamic*: Observed metrics greatly vary over time.

Computer system = strongly-correlated dynamic system
Motivation:
Knowledge discovery from correlated (or structured) dynamic systems

- Data mining from structured data has recently attracted attention

- However, most of the graph mining studies focus mainly on static data
  - We address the dynamic correlated systems, and
  - We develop a tool suitable for analyzing these systems.
Modeling Web-based systems: definition

- **Service**
  - \( s \equiv (I_{\text{source}}, I_{\text{dst}}, \text{port}#, \text{trans. type}) \)
  - contains *two* IP addresses

- **Service dependency**
  - \# of a service’s request for another service
  - log transform and symmetrize
  \[
  D_{i,j} = (\tilde{d}_{i,j} + \tilde{d}_{j,i})(1 - \delta_{i,j}) + \alpha_i \delta_{i,j}
  \]

- **Service dependency graph**
  - nodes: services
  - edge weights: service dependencies
  - defined as an undirected graph
Modeling Web-based systems: How to find D
Modeling Web-based systems: considerations

- **# of edges is relatively large**
  - e.g. more than 1000 edges for 50 services
  - the dependency (or adjacency) matrix might be sparse, but generally we do not know how sparse it is

- **Edge weights greatly vary over time**
  - autoregressive models are inappropriate in a time scale of several minutes

### Services in a benchmark system

<table>
<thead>
<tr>
<th>Index</th>
<th>$I_s$</th>
<th>$I_d$</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>0</td>
<td>(none)</td>
</tr>
<tr>
<td>1</td>
<td>192.168.0.19</td>
<td>192.168.0.53</td>
<td>80</td>
<td>Plants</td>
</tr>
<tr>
<td>2</td>
<td>192.168.0.19</td>
<td>192.168.0.54</td>
<td>80</td>
<td>Plants</td>
</tr>
<tr>
<td>3</td>
<td>192.168.0.19</td>
<td>192.168.0.53</td>
<td>80</td>
<td>Trade</td>
</tr>
<tr>
<td>4</td>
<td>192.168.0.19</td>
<td>192.168.0.54</td>
<td>80</td>
<td>Trade</td>
</tr>
<tr>
<td>5</td>
<td>192.168.0.54</td>
<td>192.168.0.53</td>
<td>5558</td>
<td>JMS</td>
</tr>
<tr>
<td>6</td>
<td>192.168.0.53</td>
<td>192.168.0.54</td>
<td>9081</td>
<td>Plants</td>
</tr>
<tr>
<td>7</td>
<td>192.168.0.53</td>
<td>192.168.0.54</td>
<td>9081</td>
<td>Trade</td>
</tr>
<tr>
<td>8</td>
<td>192.168.0.54</td>
<td>192.168.0.53</td>
<td>9081</td>
<td>Plants</td>
</tr>
<tr>
<td>9</td>
<td>192.168.0.54</td>
<td>192.168.0.53</td>
<td>9081</td>
<td>Trade</td>
</tr>
<tr>
<td>10</td>
<td>192.168.0.53</td>
<td>192.168.0.52</td>
<td>50000</td>
<td>DB2</td>
</tr>
<tr>
<td>11</td>
<td>192.168.0.54</td>
<td>192.168.0.52</td>
<td>50000</td>
<td>DB2</td>
</tr>
</tbody>
</table>

### Service dependency between 9 & 11

![Graph showing service dependency between 9 & 11 over time](image)
Problem statement:
Online anomaly detection from a time series of graphs

- Given a time-dependent graph with a fixed structure,
- detect anomalies **online** in an **unsupervised** manner.

Practical requirements
- Use a simpler feature rather than the graph itself.
- Establish a simple thresholding policy.

Why challenging?
- Edge weights are highly dynamic
- A change in individual edge weights does not necessarily indicates a fault

We wish to detect a “phase transition” of the graph
Feature extraction:
The principal eigenvector is the summary of the activity of services

- Definition of the “service activity vector (SAV)”

\[ u(t) \equiv \arg \max_{\tilde{u}} \{ \tilde{u}^T D(t) \tilde{u} \} \quad \text{subject to} \quad \tilde{u}^T \tilde{u} = 1 \]

dependency matrix at \( t \)

Why “activity”?

- If \( D_{12} \) is large, then \( u_1 \) and \( u_2 \) should be large because of \( \arg \max \) (note: \( D \) is a positive matrix).
- So, if \( s_1 \) actively calls other services, then the weight in \( s_1 \) should be large.

Mathematically, this equation is reduced to the eigenvalue equation:

\[ D(t) \tilde{u} = \lambda \tilde{u}, \quad \text{subject to} \quad \tilde{u}^T \tilde{u} = 1 \]
Feature extraction:
Also interpreted as the “stationary state” of the system

- If we regard D as the time evolution operator, then the service activity vector can be interpreted as the stationary state of the system.

Dependency matrix

\[ D = \begin{bmatrix}
  D_{11} & D_{12} & \cdots & D_{1N} \\
  D_{21} & D_{22} & \cdots & D_{2N} \\
  \vdots & \vdots & \ddots & \vdots \\
  D_{N1} & \cdots & D_{NN}
\end{bmatrix} \]

Equation of motion

\[ x(\tau + 1) = Dx(\tau) \]

stationary state

\[ u = x(\infty) / \| x(\infty) \| \]

the probability amplitude that the system is holding the control token at the service s2.

\[ u = \begin{bmatrix}
  u_1 \\
  u_2 \\
  \vdots \\
  u_N
\end{bmatrix} \]

service activity vector = summary of the system
Feature extraction: Mathematical properties

- **SAV is invariant with respect to uniform changes in traffic.**
  - we can separate normal fluctuations in traffic from anomalies

- **SAV is a positive vector.**
  - we never have negative activities.

- **SAV has no degeneracy.**
  - we are free from such subtle problems as level crossings
Anomaly detection:
From a graph sequence to a vector sequence

- The problem was reduced to anomaly detection from a time sequence of *directional data* (normalized vector).

**computer system** → **dependency matrix** → **activity vector**

**Question 1:**
How can we define the anomaly metric?

**Question 2:**
How can we determine its threshold?
Anomaly detection: Cosine-measure-like anomaly metric

- **Definition of anomaly metric**
  \[ z(t) \equiv 1 - r(t - 1)^T u(t) \]
  - \( u(t) \): activity vector at time \( t \)
  - \( r(t - 1) \): typical activity pattern at \( t - 1 \)

- **To find the typical activity pattern,**
  - We employ an LSI (Latent semantic indexing) like pattern extraction technique.
  - Perform SVD for
    - \( U = [ u(t - 1), u(t - 2), \ldots, u(t - W) ] \)
  - The principal left singular vector is the solution.
Anomaly detection:
A generative model for the anomaly metric

- For directional data, Gaussian models do not work well
  - The distribution of $u(t)$ degenerates on the surface of a hypersphere

- Our starting point is the von Mises-Fisher distribution
  - $p(u) \propto \exp \left( \frac{r^T u}{\Sigma} \right)$, $\Sigma$: angular variance
  - We can approximately derive the pdf for $z(t)$ itself.

The pdf of $z$ can be expressed as the chi-squared distribution with $N-1$ degrees of freedom.
Explicitly, the distribution of the anomaly metric, \( z \), is given by
\[
q(z) = \frac{1}{2^{\frac{N-1}{2}} \Gamma\left(\frac{N-1}{2}\right)} e^{-z/(2\Sigma)} \left( \frac{z}{\Sigma} \right)^{\frac{N-1}{2} - 1} \frac{1}{\Sigma}
\]

We wish to determine a threshold of \( z \) online
- We need to construct an online algorithm to update the parameters
- Seemingly, we have a single fitting parameter, \( \Sigma \), but this model doesn’t work well because of the “curse of dimension”

We regard \( N \) as a fitting parameter \( n \).
- \( n \): “effective dimension”
  - the actual degrees of freedom in action
  - this model works well when there are inactive degrees of freedom
Anomaly detection:
A novel online algorithm to update \( n \) and \( \Sigma \)

- **Parameter estimation is still challenging**
  - Because MLE has difficulties
    - The gamma function makes everything difficult

- **Our approach: the moment method**
  - The chi-squared distribution has explicit expressions for the 1\(^{st}\) and 2\(^{nd}\) moments
    
    \[
    q(z) = \frac{1}{2^{n-1} \Gamma \left( \frac{n-1}{2} \right)} e^{-z/(2\Sigma)} \left( \frac{z}{\Sigma} \right)^{n-1} \frac{1}{\Sigma}
    \]
    
    \[
    \langle z \rangle = \int dz q(z) z = (n - 1)\Sigma, \quad \langle z^2 \rangle = \int dz q(z) z^2 = 2(n^2 - 1)\Sigma^2
    \]
    
    - These can be easily solved wrt \( n \) and \( \Sigma \)
      
      \[
      n - 1 = \frac{2\langle z^2 \rangle}{\langle z^2 \rangle - \langle z \rangle^2}, \quad \Sigma = \frac{\langle z^2 \rangle - \langle z \rangle^2}{2\langle z \rangle}
      \]
      
    - Online estimation for the moments are easy:
      
      \[
      \langle z \rangle^{(t)} = (1 - \beta) \langle z \rangle^{(t-1)} + \beta z(t) \quad \langle z^2 \rangle^{(t)} = (1 - \beta) \langle z^2 \rangle^{(t-1)} + \beta z(t)^2
      \]
Anomaly detection:
Summary of our algorithm

dependency matrix

D(t) \rightarrow \mathbf{u}(t)

activity vector

anomaly metric
z(t)

1^{\text{st}} and 2^{\text{nd}} moments of z

Find z_{\text{th}} such that
\int_{0}^{z_{\text{th}}} dz q(z) = p_c

"0.5%"

Critical boundary, \( p_c \)

 Raise an alert if
\( z(t) > z_{\text{th}} \)

input parameter to define anomaly
Experiment:
A bug in one of the Web applications

- **Settings**
  - On each of the two WAS, two applications are running (“Trade” and “Plants”)
  - Service dependency matrices are generated every 20 seconds
  - The principal eigencluster has 12 services

- **A bug**
  - One of the “Trade” applications malfunctions at time $t_A$ and recovers at $t_B$.
  - The server process itself continues running, so the network communication is normal at the TCP layer or below.
  - *Potentially* dangerous: The throughput is hardly affected for relatively low load

---

**Table:**

<table>
<thead>
<tr>
<th>Index</th>
<th>$I_s$</th>
<th>$I_d$</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.0.0.0</td>
<td>0.0.0.0</td>
<td>0</td>
<td>(none)</td>
</tr>
<tr>
<td>1</td>
<td>192.168.0.19</td>
<td>192.168.0.53</td>
<td>80</td>
<td>Plants</td>
</tr>
<tr>
<td>2</td>
<td>192.168.0.19</td>
<td>192.168.0.54</td>
<td>80</td>
<td>Plants</td>
</tr>
<tr>
<td>3</td>
<td>192.168.0.19</td>
<td>192.168.0.53</td>
<td>80</td>
<td>Trade</td>
</tr>
<tr>
<td>4</td>
<td>192.168.0.19</td>
<td>192.168.0.54</td>
<td>80</td>
<td>Trade</td>
</tr>
<tr>
<td>5</td>
<td>192.168.0.54</td>
<td>192.168.0.53</td>
<td>5558</td>
<td>JMS</td>
</tr>
<tr>
<td>6</td>
<td>192.168.0.53</td>
<td>192.168.0.54</td>
<td>9081</td>
<td>Plants</td>
</tr>
<tr>
<td>7</td>
<td>192.168.0.53</td>
<td>192.168.0.54</td>
<td>9081</td>
<td>Trade</td>
</tr>
<tr>
<td>8</td>
<td>192.168.0.54</td>
<td>192.168.0.53</td>
<td>9081</td>
<td>Plants</td>
</tr>
<tr>
<td>9</td>
<td>192.168.0.54</td>
<td>192.168.0.53</td>
<td>9081</td>
<td>Trade</td>
</tr>
<tr>
<td>10</td>
<td>192.168.0.53</td>
<td>192.168.0.52</td>
<td>50000</td>
<td>DB2</td>
</tr>
<tr>
<td>11</td>
<td>192.168.0.54</td>
<td>192.168.0.52</td>
<td>50000</td>
<td>DB2</td>
</tr>
</tbody>
</table>
Experiment:
The malfunction could be detected

- The malfunction started at $t_A$ and finished at $t_B$

- Time evolution of SAV
  - clearly visualizes the malfunction period
  - the malfunction of the single service (#11) causes a massive change

- Anomaly metric
  - Two features clearly indicate the malfunction period
    - The latter is the evidence that the online calculation works well

- Calculated threshold value
  - dynamically adapted to the situation
  - $n \sim 4$ is much smaller than $N=12$
Summary

- We have considered the issue of anomaly detection from a highly dynamic graph sequence.

- We have introduced several new concepts
  - service activity vector
  - cosine-measure-like anomaly metric
  - effective dimension
  - moment method

- We demonstrated the utility of our approach in a benchmark system.