3F3-05

固有空間におけるコンピュータシステムの障害検知

Eigenspace Approach to Anomaly Detection of Computer Systems

井手剛　　鹿島久嗣
Tsuyoshi Idé　　Hisashi Kashima

IBM 東京基礎研究所
IBM Research, Tokyo Research Laboratory

We report on an automated runtime anomaly detection method at the application layer of multi-node computer systems. We model a Web-based system as a weighted graph, where each node represents a service and each edge represents a dependency between services (or requests). Since the edge weights vary greatly over time, the problem we address is that of anomaly detection from a time sequence of graphs. In our method, we first extract a feature vector from the adjacency matrix that represents the activities of all of the services. The heart of our method is to use the principal eigenvector of the eigenclusters of the graph. We demonstrate that a fault in a Web application can be automatically detected and the faulty services are identified without using detailed knowledge of the behavior of the system.

1. はじめに

インターネット関連インフラの整備をひとつの契機として、従来の関係データの枠を飛び出すような様々なデータ形式がデータマイニングの研究の対象として意識されるようになってきた。グラフは、アイテム同士の関係を記述する上で非常に一般的なデータ構造であり、グラフ集合からのマイニング技術は、今後ますます注目を集めるものと考えられる [Washio 03]。

グラフマイニングは、いわば、従来のデータマイニング技術を「空間」方向へ拡張することを目指したものと言える。もうひとつの一般化の方向として、時間軸方向への問題の拡張がありえる。先駆的な仕事としては、半構造データの時系列から順序木パターンを検出する手法を示した Asai ら [Asai 02] の研究が挙げられる。

当然のことながら、これらストリームマイニングないしグラフマイニングの研究の多くは、現在のところ、たとえば頻出アイテム集合の検出といった古典的な問題設定に基づくことが多い。本稿では、その種の問題設定からやや離れ、多くの変数が強く絡み合い、しかも時間的に激しく変化する系の異常検知の問題を、教師なし学習の枠組みで考える。具体的には、Web 系コンピュータシステムの「サービス」同士の関係を表すグラフを例に取り、アプリケーション層における障害検知について論ずる。

2. コンピュータシステムのモデル化

2.1 サービス関連度行列

Web 系システムの「サービス」を、下記の 4 つ組で定義する：

$$(I_s, I_d, P, Q),$$

I_s と I_d はそれぞれ、呼び出し元と呼び出し先の IP アドレス、P は呼び出し先のアプリケーションのポート番号、Q は要求の種別を表す。たとえば、図 1 のベンチマークシステムにおいて、(i_1, i_3, p_1, q_1) がサービスの例である。ここで、i_1 と i_3 はたとえば、それぞれ 192.168.0.19 と 192.168.0.53 というような IP アドレスであり、p_1 はポート番号 80、q_1 にはたとえばベンチマークアプリケーションの名前が来る。

連絡先: 井手剛, IBM 東京基礎研究所, 242-8502 大和市下鶴間 1623-14（LAB-S7B）, E-mail: goodidea@jp.ibm.com

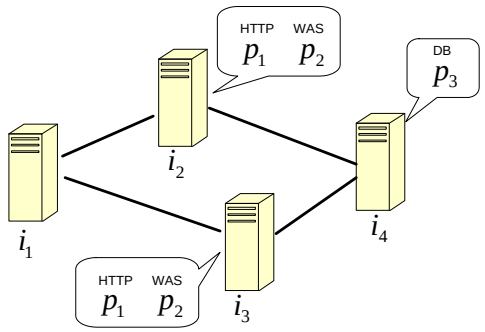

図 1: ベンチマークシステムの構成。IP アドレスとポート番号をそれぞれ i_k $(k = 1, .., 4)$ と p_j $(j = 1, 2, 3)$ で表してある。

次に、このようなサービスを頂点とするグラフを考える（図 2）。IP アドレスをふたつ含むという意味で、図 1 のような素朴な IP ネットワークよりも高次の空間を考えていることに注意されたい。この高次元化にともない、図 1 のような単純なシステムにおいても、サービスのグラフは比較的複雑になりえる。

サービス i と j を結ぶ辺には、サービスの出現頻度に対応した重みを与える。すなわち、このグラフの隣接行列 D の i, j 要素として、

$$\mathsf{D}_{i,j} = (\tilde{d}_{i,j} + \tilde{d}_{j,i})(1 - \delta_{i,j}) + \alpha_i \delta_{i,j} \qquad (1)$$

という重みを考える。ここで、$\tilde{d}_{i,j}$ は、ある所定の期間においてサービス i がサービス j を呼び出した回数 $d_{i,j}$ の単調増加関数で、ここでは対数変換された量をとることにする。また、α_i は数値計算の安定化のために導入された小さい正数、$\delta_{i,j}$ はクロネッカーのデルタである。$d_{i,j}$ の測定ないし推定の方法は自明ではないが、いくつかの既知の方法（たとえば [Gupta 03]）で計算できるので、ここでは論じない。上で与えられる行列を、以降、サービス関連度行列と呼ぶ。

2.2 問題設定

本稿で考える問題を改めてまとめておく。われわれは、システムの振る舞いがサービス関連度行列 D によって特徴付けら

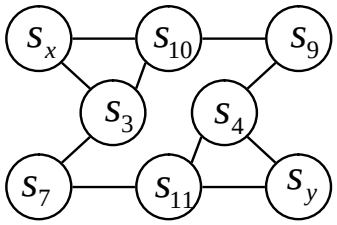

図 2: 図 1 のシステムにおけるサービス関連度グラフの例。.

れると仮定する。その次元 N と、各要素の定義は固定されているものとみなすが、その値は時間的に激しく変動する。問題は、この行列を、グラフの隣接行列と見て、その時系列から異常を検知することである。

考えるべき点としては、隣接行列の次元はしばしば 100 のオーダーとなり、個別の行列要素もしくはそれらの恣意的な組み合わせの監視は現実的ではないこと、また、隣接行列の要素は強く時間に依存するので、個々の要素を個別に眺めるだけでは障害の検出のしようがないこと（単なるトラフィックの変動かもしれない）、障害を判定する基準からできるだけ恣意的なパラメターを排除すること、が挙げられる。

3. グラフ時系列からの特徴抽出

3.1 活動度ベクトルの定義

サービス関連度行列は、ある離散的な時刻 t=1,2,... において時系列的に得られるものとする。サービス関連度グラフの特徴ベクトル $\boldsymbol{u}$ として、下記のようなものを考える。

$$\boldsymbol{u}(t) \equiv \arg\max_{\tilde{\boldsymbol{u}}} \left\{ \tilde{\boldsymbol{u}}^T \mathsf{D}(t) \tilde{\boldsymbol{u}} \right\} \tag{2}$$

ただし $\tilde{\boldsymbol{u}}^T \tilde{\boldsymbol{u}} = 1$ とし、さしあたり関連度グラフが単一の連結成分しか持っていないと仮定しておく。ベクトルは列ベクトルと考える。T は転置を表す。行列 D は非負なので、上式の目的関数を最大化するには、大きい行列要素に対応するサービスにおいて、特徴ベクトルの要素が大きくなっていなければならない。すなわち、もしあるサービス s が活発に他のサービスを呼び出していれば、特徴ベクトルの s 成分は大きいはずである。この意味で、この特徴ベクトルを、活動度ベクトルと呼ぶ。

ラグランジュ係数 λ を導入すれば式 (2) は下記のように書きなおせる。

$$\frac{d}{d\tilde{\boldsymbol{u}}} \left[\tilde{\boldsymbol{u}}^T \mathsf{D}(t) \tilde{\boldsymbol{u}} - \lambda \tilde{\boldsymbol{u}}^T \tilde{\boldsymbol{u}} \right] = 0, \tag{3}$$

すなわち

$$\mathsf{D}(t) \tilde{\boldsymbol{u}} = \lambda \tilde{\boldsymbol{u}}. \tag{4}$$

この方程式は $\mathsf{D}(t)$ の任意の固有ベクトルに対して成り立つが、活動度ベクトルは主固有ベクトル（最大固有値に属する固有ベクトル）として定義される。

ひとつ大切な点は、式 (4) が $\tilde{\boldsymbol{u}}$ について同次形であることである。このため、たとえば、トラフィックの急増により、すべてのサービス関連度が一斉に k 倍されたとしても、その変化はもっぱら固有値に吸収され、活動度ベクトルの方向には影響しない。

活動度ベクトルは、運動方程式が

$$\boldsymbol{x}(\tau + 1) = \mathsf{D}(t) \boldsymbol{x}(\tau), \tag{5}$$

で与えられる離散時間の線形力学系の定常状態と結びつけて解釈することもできる。ここで、τ は実時間 t とは独立なある仮想時間を表し、$\boldsymbol{x}$ は $\boldsymbol{u}$ と、$\boldsymbol{u}$=$\boldsymbol{x}$/||$\boldsymbol{x}$|| という関係で結ばれる。$\mathsf{D}(t)$ は定義より対称で、また、少なくとも $\alpha > 0$ に対してはフルランクなので、すべての固有値は実である。対応する固有ベクトルを用いて $\boldsymbol{x}(0)$ を展開することで、

$$\boldsymbol{x}(\infty) = \lim_{n \to \infty} [\mathsf{D}(t)]^n \boldsymbol{x}(0) = \lim_{n \to \infty} \sum_{i=1}^{N} [\lambda_i(t)]^n c_i(t) \boldsymbol{u}_i(t),$$

ただし $c_i(t)$ は対応する展開係数である。明らかに、$n \to \infty$ において、

$$\boldsymbol{u}(t) = \boldsymbol{x}(\infty) / ||\boldsymbol{x}(\infty)||.$$

を得る。すなわち、この力学系の状態ベクトルは、無限回の遷移の後に活動度ベクトルに収束する。このことから、コンピュータシステムでは、この定常状態は、ある仮想時刻 τ において、システムのコントロール・トークンを、あるサービスが保持する確率振幅として解釈できることがわかる。

3.2 活動度ベクトルの諸性質

われわれのサービス関連度行列は非負行列であるため、数学的には Perron-Frobenius の定理などに基づき、さまざまな都合のよい性質が成り立つことを示せる。詳細は別の機会に譲るとして、ここでは結果だけを述べる。

まず、これは前節ですでに述べたが、全体的なトラフィック変動に対する頑強性を持っていることを言える：

性質 1 $\boldsymbol{u}$ の方向は、変換 $\mathsf{D}(t) \to k\mathsf{D}(t)$ に対して不変である。ただし、k は任意の正数とする。

次に、下記のふたつの性質から、活動度の値は負にならず、しかも活動度ベクトルの方向は一意に決まることが分かる。

性質 2 それぞれの連結成分において、主固有ベクトルは正ベクトルである。すなわち、すべての要素が正になるようにできる。

性質 3 それぞれの連結成分において、主固有値には縮退がない。

一般には、時系列的に得られるグラフは非連結グラフとなるが、その場合は、最大の固有値を持つ連結成分（これを主固有クラスター [Sarkar 98] と呼ぶ）の主固有ベクトルを、全系の活動度ベクトルと定める。幸いなことに、Web 系のシステムでは、HTTP サーバーが Web アプリケーションサーバーを呼び出すというような重要なサービスは主固有クラスターに集中しているので、単純なルールで主固有クラスターを識別することができる。

最後にふたつほどコメントを述べておく。まず最初に、この特徴抽出手法は、D に含まれる情報の効率のよい圧縮手法となっている。すなわち、全系を表すグラフから、主固有クラスターを抽出し、さらにそこから活動度ベクトルを抽出することで、自由度を大幅に削減している。特徴量の意味づけが上記のように明確であることと併せ、素朴に全系の隣接行列の全要素を一列に並べて特徴ベクトルにする、というような手法に比べて、より利口な手法だと言える。

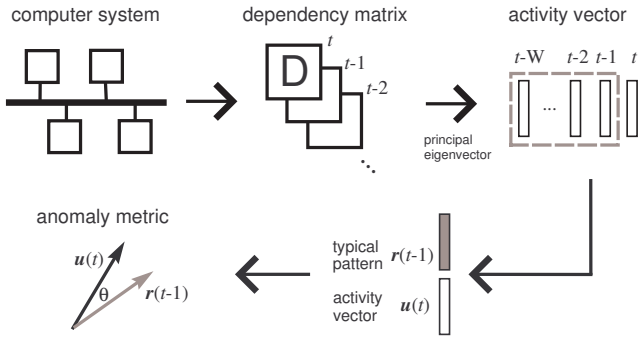

図 3: 異常検知手順のまとめ

第二に、主固有値および主固有ベクトルを求めるための数値計算は、べき乗法と呼ばれる手法を使って非常に高速に実行できる。活動度ベクトルはオンラインに、われわれの実験環境であれば 10 秒のオーダーの時間間隔の間に計算する必要があるが、N が数百程度では計算時間は全く問題にならない。

4. 異常検知手法

活動度ベクトルの時系列 $\{\boldsymbol{u}^{(t)}\}$ for $t = 1, 2, ...$ からの異常検知問題を考える。活動度ベクトルは規格化されているので、これは方向データ [Banerjee 03] の時系列からの異常検知ということになる。基本的な手順は、過去の正常時のパターンをオンライン学習して、現時点の活動度ベクトルとの相違度を見る、というものである。

ここで便宜上、行列 $\mathsf{U}(t)$ を以下に定義する。

$$\mathsf{U}(t) = [\boldsymbol{u}(t), \boldsymbol{u}(t-1), \cdots, \boldsymbol{u}(t-W+1)], \qquad (6)$$

ここで W は時間窓のサイズである。明らかに $\mathsf{U}(t)$ は $N \times W$ の行列となる。W の時間窓の範囲での代表パターンを、列ベクトルの 1 次結合として、

$$\boldsymbol{r}(t) = c \sum_{i=1}^{W} v_i \boldsymbol{u}(t-i+1), \qquad (7)$$

のようにおく。ただし c は $\boldsymbol{r}$ を 1 に規格化するための係数である。単純には、$\{v_i\}$ を定数とし、$\boldsymbol{r}(t)$ を相加平均とみなすことができる。ここでは話を分かりやすくするため窓のサイズを限定したが、忘却率を取り入れて $\boldsymbol{r}(t)$ をオンライン学習する形の定式化もまた可能である。

この特徴パターンと、現時点での観測ベクトルとの相違度を測るために、スカラー量 $z(t)$ を下記に導入する：

$$z(t) \equiv 1 - \boldsymbol{r}(t-1)^T \boldsymbol{u}(t). \qquad (8)$$

この値は、時刻 $t-1$ での特徴パターンが t での活動度ベクトルと直交していれば 1、平行であればゼロとなる。今の文脈で言えば、この $z(t)$ がある閾値よりも大きければ、現時刻において何か異常な状況が起こっていると判断するわけである。われわれの異常検知手法を、図 3 にまとめた。

5. 実験

5.1 実験構成

図 1 に示した 2 重冗長構成の Web 系システムにおいて、異常検知の実験を行った。ウェブアプリケーションサーバー（WAS）の上では、Trade3 [IBM 02] と Plants というふたつのアプリケーションが動作しているものとする [*1]。両者ともクライアント数は 16、いわゆる think time は 0 から 4 秒の間でランダムに選択した。

表 1: 主固有クラスターに現れるサービスの一覧

Index	I_s	I_d	P	Q
0	0.0.0.0	0.0.0.0	0	(none)
1	192.168.0.19	192.168.0.53	80	Plants
2	192.168.0.19	192.168.0.54	80	Plants
3	192.168.0.19	192.168.0.53	80	Trade
4	192.168.0.19	192.168.0.54	80	Trade
5	192.168.0.54	192.168.0.53	5558	JMS
6	192.168.0.53	192.168.0.54	9081	Plants
7	192.168.0.53	192.168.0.54	9081	Trade
8	192.168.0.54	192.168.0.53	9081	Plants
9	192.168.0.54	192.168.0.53	9081	Trade
10	192.168.0.53	192.168.0.52	50000	DB2
11	192.168.0.54	192.168.0.52	50000	DB2

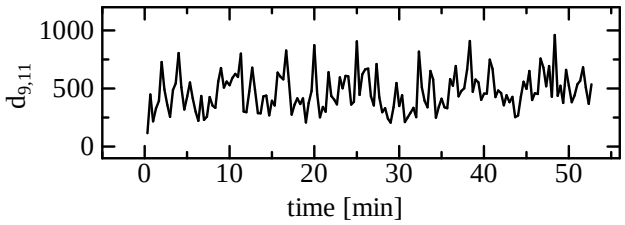

図 4: サービス 9 によるサービス 11 の呼び出し回数の時間変化.

サービス関連度行列は 20 秒間隔で生成するものとし、$d_{i,j}$ は採取されたイーサネットパケットから推定した。ループバックパケットは無視したので、図 2 における s_x と s_y のサービスは観測されていない。ただし i_1 = 192.168.0.53 および i_2 = 192.168.0.54 である。主固有クラスターは表 1 に定義されている。簡単のため、これらに対する摂動は無視し、活動度ベクトルはここに示されたサービスで張られる空間に限定する。表 1 において、第 0 サービスは呼び出し元と呼び出し先の最適な対が見出せない場合を表現するために人工的に導入されたもので、これを無視しても結果の解釈には影響はない。表で、DB2 は DB サーバーへのリクエストを表し、JMS は Java Messaging Service に関連した通信を表す。

サービス関連度行列の時間変化を見るために、図 4 に、一例として $d_{9,11}$ の挙動を示した。サービス関連度行列は 52.7 分にわたり 158 個が生成された。図より、上記実験条件の下では 20 秒間におよそ 500 回のコールがあることがわかる。揺らぎはきわめて大きく、その振幅は、ほとんど平均値と同じオーダーとなっている。対応する行列要素も、正規分布で扱えるような穏やかなものではなく、関連度行列の行列要素に直接閾値を設定してもあまり意味がないことがこの結果から分かる。

5.2 異常検知

異常検知能力を調べるために、サービス 11、すなわち、192.168.0.54 における “Plants” に変調を生じさせ、DB コールを止める実験を行った。動作は不調であるが、サーバーソフ

*1 後者は IBM WebSphere Application Server V5 付属のサンプルアプリケーションであり、花のオンライン販売を模擬する。

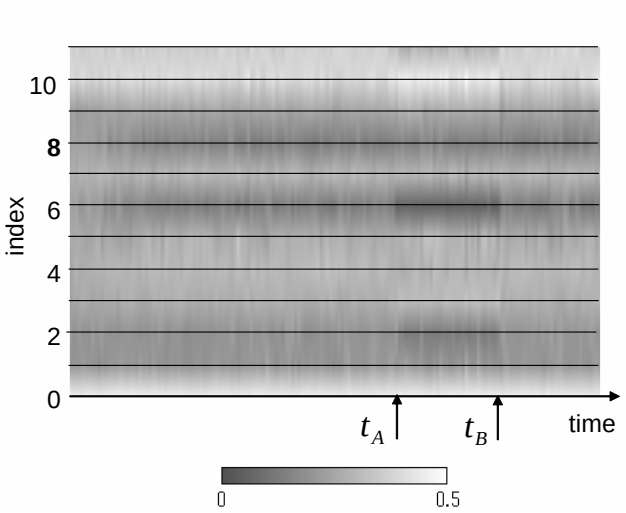

図 5: 活動度ベクトルの時間依存性。t_A から t_B までの期間が障害発生期間に対応している。

トウェアのプロセス自体は動いている。しかも、WAS の一方は正常に動いているために、トラフィックがさほど大きくないうちは応答時間にもほとんど異常は出ない。そのため通常のネットワーク管理ソフトでは検出が難しい。

図 5 に、計算された活動度ベクトルを示す。縦軸が表 1 に対応するインデックス、横軸が時間である。時刻 t_A と t_B において、段差状の変化が見て取れる。これはサービス 11 の機能不全とその復旧に対応している。この区間においては、サービス 2、6、11 に活動度の現象が観察される。サービス 6 はサービス 11 の上流側にあり、11 の不調に伴い、この活動度も低下したと解釈される。サービス 2 が強く影響を受けたのは、ループバックパケットが非観測であるためである。Web 系システムのようなサービス同士の相関が強いシステムでは、ひとつのサービスの異常な挙動が、このように、システム全体にある種の相変化を生じさせる。活動度ベクトルは、そのような系全体にわたる変化を抽出するために好適な手法であると言える。図 5 は、システム全体の可視化手法という点でも興味深いものである。

この障害を自動検知するために、z をオンラインで計算した。図 6 において、z の値は $\boldsymbol{r}(t)$ を相加平均で計算したもの（method 1）と、ノイズ除去したもの（method 2）の両方を示してある。窓の大きさは W =25 に選んだ。比較のため、統計的に 0.5%危険域に対応する値をオンラインで学習し、併せてプロットしてある。図では、t =35.0 と 45.7 の区間で明瞭な構造が見てとれる。明らかにこれは図 5 の $t_A < t < t_B$ に対応しており、障害の自動検知における本手法の有効性が確認できる。

6. むすび

時系列的にグラフの隣接行列が得られる状況において、コンピュータシステムの異常検知の問題を定式化し、従来困難であった冗長構成のアプリケーション層における障害検知に成功した。

本稿では、非連結グラフの活動度ベクトルの安定性の問題、閾値設定における方向データのモデル化の問題、特徴パターン抽出の最適化の問題などには詳しく触れなかったが、別の機会に論ずるつもりである。

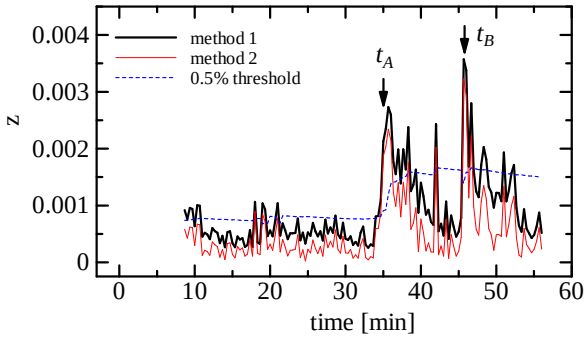

図 6: z の時間依存性

観測量の奥深くに潜んだある構造を掘り出して、その変化から何かの兆候を探るというモチーフは、比較的汎用性の高い視点だと思われる。関連分野の刺激となれば幸いである。

参考文献

[Washio 03] T. Washio and H. Motoda, “State of the art of graph-based data mining,” ACM SIGKDD Explorations Newsletter archive, No.5, Issue 1, pp.59-68, 2003.

[Asai 02] T. Asai, H. Arimura, K. Abe, S. Kawasoe, S. Arikawa, “Online algorithms for mining semi-structured data stream,” Proceedings. 2002 IEEE International Conference on Data Mining, pp.27-34.

[Gupta 03] M. Gupta, A. Neogi,M. Agarwal and Gautam Kar, “Discovering Dynamic Dependencies in Enterprise Environments for Problem Determination.” Proceedings of 14th IFIP/IEEE Workshop on Distributed Systems: Operations and Management, 2003.

[Sarkar 98] S. Sarkar and K. Boyer, “Quantitative Measures for Change Based on Feature Organization: Eigenvalues and Eigenvectors,” Computer Vision and Image Understanding, Vol. 71, pp. 110-136, 1998.

[Banerjee 03] A. Banerjee, I. Dhillon, J. Ghosh and S. Sra, “Generative Model-based Clustering of Directional Data,” Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.19-28, 2003

[IBM 02] IBM, Trade 3, `http://www-306.ibm.com/software/webservers/appserv/benchmark3.html`.