研究報告

# 学内 Web サイトログからの閲覧者の行動分析

学習院大学 計算機センター　久保山　哲　二
計算機センター　磯　上　貞　雄
計算機センター　城　所　弘　泰
計算機センター　村　上　登志男

## 1　はじめに

計算機センターでは、様々なネットワーク機器の通信状況などを記録した大量のログデータを蓄えているが、これらは、主として異常発生時の原因特定や、利用状況把握のための統計情報の集計などに活用されている。これらのログデータがもつ情報を有効に活用し、学内ネットワークシステムの利用状況の定性的な把握と今後のシステム構成や利用形態の改善に活用可能なログデータからの知識発見手法の開発と分析が望まれる。

ログデータはメールサーバや WWW サーバをはじめとする様々なネットワーク機器で蓄えられている。たとえば、メールサーバのログデータからは、時間あたりの電子メールの処理件数や、送信したデータ量、配送にかかった時間等の統計情報を集計することが可能であり、実際に、メールサーバの負荷を把握するために、このような統計情報が活用されている。しかし、従来的な統計情報を超えて、より抽象的なレベルでユーザーの振る舞いをモデル化することができれば、学内システムの構成改善や効率化に役立つ知識が得られるだけでなく、効率的な情報伝達や、学生のインターネット利用の定性的な把握と情報処理教育への活用も期待できる。

一方、ログの背後にある複雑な構造の分析については、近年、友人関係、企業の取引関係、生体内の遺伝子やタンパク質の相互作用など、人間社会や自然界の様々な現象によって形成されるネットワーク構造の分析や、時系列データのモデル化が注目を浴びており、大量ログの背後にある構造を分析・理解するための数理的手法が開発されている。

本稿では、個々人の閲覧行動の推定については報告の対象とせず、一般的な分析手法について報告する。また、Web のログデータにとどまらず、コンピュータ上に蓄えられている大量の操作ログを対象とした分析の試みについても報告する。

## 2　Web サーバーのログ分析

学習院の Web サイトには、大学、高等科、中等科、女子中・高等科、初等科、幼稚園のコンテンツが含まれている。2010 年 10 月の Web サーバ (apache) のアクセスログ約 2900 万行 (約 8.5GByte) を分析対象にした。分析は、まず、閲覧者の Web サイト内での振る舞いをページ間移動の動線の構造として抽出することによって、閲覧者の閲覧パターンに基づくデータの可視化を行った。具体的には、サーチエンジンから学習院 Web サイトへのアクセスに着目し、どのようなキーワードでどのページにアクセスし、ページをたどっているのかを追跡し、グラフ構造を作成した。このグラフ構造に基づき、タグクラウドを作成した (図 1 参照)。頻度の高いキーワードは大きくし、構造的に近いページにアクセスしているか、または、結果的に近いページに短いパスでたどり着いているキーワード同士を近い位置に配置した。また、サイト内の閲覧パターンの成すグ


図 1　ユーザーの訪問キーワードと訪問パターンから作成したタグクラウド

ラフ構造から、モジュラリティ値に基づく Newman アルゴリズムを用いたクラスタリングを行い、色付けをした。このタグクラウドにより、サーチエンジンからアクセスしている閲覧者の需要を可視化し、対象ページ群を抽出することができた。

以下では、閲覧行動からのユーザーの所属判定の試みの 1 つとして、サポートベクトルマシンと木カーネルを用いた手法について、一般のベンチマークデータを対象に報告する。

## 2.1 ユーザの閲覧行動によるドメイン判定

一般的に Web サイトのリンク構造は、閉路や合流を含むグラフ構造をなしている。これを Web リンクグラフと呼ぶことにする。Web リンクグラフ上のユーザの閲覧経路も、一般に木構造ではなく閉路や合流が存在する。この閲覧経路から、出発点 (Web ページ) を根ノードにして閲覧順にリンクを繋ぎ、閉路や合流が発生する場合には、そのリンクを削除することにより、ユーザの閲覧経路を木構造として表現する。

このようにして、Web サイトを訪れたユーザの閲覧行動を木構造にしたデータを閲覧木とよび、形式的には以下のように定義する。

定義 1（閲覧木）　Web ページ $v_1$(リファラ) から $v_2$(リクエストページ) への訪問がログに記録されているとき、次の手順にしたがって構成される木を閲覧木という。

1. ノード $v_1$ が存在しないとき、$v_2$ を根とする 1 ノードの木 $T$ を新たに作り、$v_2 \in V(T)$ とする。
2. さもなくば、すでに存在する $v_1$ をノードとする木 $T$ に対して、$v_1$ の子ノードとして $v_2$ を追加する。ただし、この追加によって、$T$ が木でなくなる場合には追加しない。

3. 上記のいずれの条件も満たさないとき、$v_1$ を根、$v_2$ を葉とする 2 ノードの木 $T$ を新たに作る。

閲覧木はユーザの巡回行動を表現するデータ構造であり、文献 [3] では論理セッションと呼ばれている。

本報告では、Zaki らによる Web の巡回閲覧ログデータ (CSLOG データ) [8] を用いた。このデータは Web サーバーログ用のマークアップ言語である LOGML により記述されている。ユーザセッションは Web グラフ上の閲覧木として XML の木構造にマッピングされている。

CSLOG データから 500 のセッションを抽出した。各々のセッションの木構造としての性質は、平均のノード数が 12.0、平均の高さが 4.3 である。このデータを対象に、大学関係のドメインとそれ以外のドメインからの訪問者にユーザを分類する。具体的には、“.edu” または “.ac.” を含むドメインから訪問しているユーザーと、それ以外のドメインから訪問しているユーザーに分類する問題とした。Web サイトの訪問者の所属により、閲覧行動が異なれば、ユーザーを分類できる。そのために、閲覧木間の類似度 (カーネル関数) を設計し、サーポートベクターマシン (SVM) による分類性能を評価した。閲覧行動を表す閲覧木の特徴量として、次に示す LCA 保存パターンを用 いた。

## 2.2 LCA 保存パターンカーネル

閲覧木は順序付き根付き木の形をしている。以下では、根つき木 $T$ の頂点集合を $V(T)$ と表記することにする。木 $T$ の 2 頂点 $x, y \in V(T)$ について、両者の共通の祖先 (根の方向に遡ったときに共通に現れるノード) で、かつ、もっとも両者に近い位置にあるノードを、木 $T$ における $x, y \in V(T)$ の最近共通祖先 (nearest common ancestor: NCA) という。$\mathrm{nca}(x, y)$ により、2 頂点 $x$,$y$ の NCA を表す。また、2 つの木の構造的な対応関係を表す概念としてマッピングを導入し次のように定義する。

定義 2　木 $T_1$ から木 $T_2$ へのマッピング $M$ は、$M \subseteq V(T_1) \times V(T_2)$ で、かつ、任意の $(x_1, y_1), (x_2, y_2) \in M$ について、次の条件をみたす頂点対の集合である。

1. $x_1 = x_2$ かつ、そのときに限り $y_1 = y_2$ である。
2. $x_1$ が $y_1$ の祖先であるとき、かつ、そのときに限り $x_2$ も $y_2$ の祖先である。

さらに、木 $T_1$ から $T_2$ へのマッピング $M$ の任意の要素 $(x_1, y_1), (x_2, y_2) \in M$ について、$(\mathrm{nca}(x_1, x_2), \mathrm{nca}(y_1, y_2)) \in M$ を満たすマッピングを *LCA* 保存マッピングという。木 $T_1$ から $T_2$ への LCA 保存マッピング $M$ のすべての集合 $\mathcal{M}(T_1, T_2)$ について、カーネル関数 $\mathbf{K}(T_1, T_2)$ を次のように定義し、LCA 保存パターンカーネルと呼ぶ。

$$\mathbf{K}(T_1, T_2) = \sum_{M \in \mathcal{M}(T_1, T_2)} M$$

研究代表者らは、これまでに様々な木カーネルを提案している [7]。このカーネルは実質的に Kashima と Koyanagi による Elastic カーネルと等価である [1]。このカーネル関数を用いて、SVM により分類性能を評価した。その結果、正確度 (accuracy) は 98.600 であった。この結果から、Web サイトの閲覧行動のパターンから、閲覧者の接続元ドメインを高い性能で推定することが可能であることがわかる。今後、接続元ドメインだけでなく様々な閲覧者の属性推定についての応用を行う予定である。

表 1　ウィンドウ遷移ログ

| Application | Start Time | Duration (sec) |
|---|---|---|
| Web | 2010-04-01 8:00:30 | 403 |
| Mail | 2010-04-01 8:07:13 | 165 |
| ⋮ | ⋮ | ⋮ |
| Word | 2010-04-01 9:04:30 | 328 |

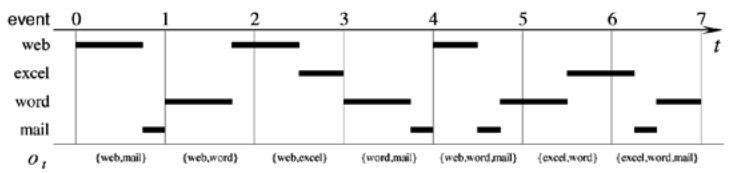


図 2　ログから作成される観測値

## 3　ログとユーザー行動モデル

以下で報告するログ分析は、研究代表者と齋藤良平氏らとの研究 [5] に基づくものである。

### 3.1　クライアント PC のウインドウ遷移ログ

本節では、分析の対象となるウインドウ遷移ログについて述べる。ウィンドウ遷移ログは Microsoft Windows の操作中にアクティブになっているウィンドウのプロセスと時間が保持される（表 1)。

プロセス名の時系列ログは抽象度が低く、この情報からユーザーのワークフローを推定することは難しい。そこで、このような低レベルログから、より抽象度の高い操作パターンを推定する手法について提案する。ウィンドウ遷移ログをイベント系列として定式化する。まず、$\mathcal{E}$ をログ中 Excel や Word の様に記録されているイベント $\{e_1,\ldots,e_m\}$ の集合とする。

等間隔に測定された離散的なインターバル $t \in \{0,\ldots,T\}$ に対して、インターバル $[t,t+1)$ 中に観測されたイベントの集合を観測値とする。例として $t \in \{8\ 00, 8\ 10 \ldots, 21\ 00\}$ とすると $[8\ 00, 8\ 10)$ は 8:00 から 8:09:59 までの期間である。インターバル $[t,t+1)$ に観測されたイベントの集合を $o_t$ とし、$o_t = \{e \in \mathcal{E} \mid e \text{ occurs in } [t,t+1)\}$ と定義する。これにより、ログから観測値としてイベント集合の系列を得る。

$$\boldsymbol{O} = (o_1,\ldots,o_T), \text{ where } o_t \subseteq \mathcal{E} \text{ for } t \in \{1,\ldots,T\}.$$

ウィンドウ遷移をタイムテーブルで表わした図とその時刻におけるイベント集合を図 2 に示す。

表 2　表記

| | | | |
|---|---|---|---|
| $0,\ldots,T$ | 離散時間系列 | $\mathcal{E}=\{e_1,\ldots,e_m\}$ | イベントの集合 |
| $\boldsymbol{O}=(o_1,\ldots,o_T)$ | イベント集合の系列 | $o_t\subseteq\mathcal{E}$ | 時刻 $t$ に観測されたイベントの集合 |
| $\boldsymbol{I}=(I_1,\ldots,I_\tau)$ | セグメントの系列 | $S=\{s_1,\ldots,s_K\}$ | タスク状態の集合 |
| $\boldsymbol{S}=(\sigma(1),\ldots,\sigma(\tau))$ | $\boldsymbol{I}$ に対応する状態の系列 | | |

## 3.2　ユーザー行動モデル

イベント列をイベントの混合比により抽象化したタスクとして表現し、このタスクの時系列が隠れマルコフモデル (Hidden Markov Model:HMM) の遷移により生成されるモデルをユーザーの行動モデルとする。まず、抽象度の高いタスクパターンモデルを作成するため、イベント集合系列 $\boldsymbol{O}=(o_1,\ldots,o_T)$ を $\mathcal{I}=(I_1,\ldots,I_\tau)$ により分節化する。

$$\underbrace{o_1,\ldots,o_{t_1-1}}_{I_1},\underbrace{o_{t_1},\ldots,o_{t_2-1}}_{I_2},o_{t_2},\ldots,o_{t_\tau-1},\underbrace{o_{t_\tau},\ldots,o_T}_{I_\tau},$$

ここで $t_1<t_2<\cdots<t_\tau\leq T$ とし、$i\in\{1,\ldots,\tau\}$ に対して各セグメントを $I_i=(o_{t_i},\ldots,o_{t_{i+1}-1})$ とする。ユーザーは $K$ 個のタスク状態を持つとして、それぞれに固有の状態 $s_i\in\{s_1,\ldots,s_K\}=S$ を与える。また、$\boldsymbol{I}$ 中の各セグメント $I_i$ は $i\in\{1,\ldots,\tau\}$ 中のイベントの混合比に応じて、タスク $\sigma(i)\in S$ を割り当てる。ユーザーはセグメント $I_1$ に対応するタスク $\sigma(1)$ からタスクを開始しセグメント $I_\tau$ 中の $\sigma(\tau)$ でタスクを終える。このとき、タスク遷移は $\boldsymbol{S}=(\sigma(1),\ldots,\sigma(\tau))$ となる。状態 $s_i$ から状態 $s_j$ への遷移確率を $a(s_i,s_j)\in S\times S$ とすると、任意の 2 状態間の遷移確率は遷移行列 $A=\{a_{ij}\}$ により表現できる。初期確率 $\pi_s$ はユーザーが状態 $s\in S$ からタスクを開始する確率であり $\Pi$ は $\{\pi_{s_1},\ldots,\pi_{s_K}\}$ の集合とする。

提案モデルでは、各タスク $s$ は観測されたウィンドウ (イベント) の出現確率によって決定される。例えば、Web ページを作成するタスクに対応する状態があるとすると、Web ブラウザーと HTML エディターの使用頻度が、それ以外のアプリケーションに比べて高くなる。よって、タスク $s\in S$ で観察されるアプリケーションウィンドウ $o\subseteq\mathcal{E}$ は出力確率 $b_s(o)$ に従うと仮定する。ここで、全ての状態 $s\in S$ に対する出力確率 $b_s(o)$ の集合を $B$ とし、$o\subseteq\mathcal{E}$ とする。与えられた集合 $B$ に対して状態 $s$ でアプリケーションウィンドウ $e$ がアクティブになる確率を $P_s(e)$ は次のようになる。

$$P_s(e)=\sum_{o\subseteq\mathcal{E}\text{ such that }e\in o}b_s(o)$$

従って、アプリケーションの数を $m$ としたとき、タスク $s$ におけるユーザーはそれぞれのアプリケーションウィンドウ $e_1,\ldots,e_m$ に対して確率 $P_s(e_1),\ldots,P_s(e_m)$ で出現する。タスク $s$ に対して、これらの出現確率を $m$ 次元のベクトルとして

$$M_s=(P_s(e_1),\ldots,P_s(e_m))$$

と定義し、$s$ のタスク要約と呼ぶ。

## 3.3　ユーザー行動モデルの推定

HMM はパラメーター $\lambda = (\Pi, A, B)$ で定義される。HMM の各隠れ状態はそれぞれタスクと対応し、その状態がアプリケーションの組み合わせを生成する確率は $B$ によって決定され、初期確率 $\Pi$、遷移確率 $A$ によって系列が生成される。提案手法ではウィンドウ遷移ログから得られた観測値の系列を用いて HMM のパラメーター $\lambda$ を推定することでユーザーの作業の要約を行う。

Baum-Welch アルゴリズム [4] を用いてパラメーター $\lambda$ の推定を行う。エルゴード性 HMM (全ての状態が任意の状態に遷移出来る HMM にこのアルゴリズムを適用する場合、パラメーターの初期値が結果に大きく影響することが知られている。提案するユーザー行動モデルはエルゴード性を持っているため、適切な初期値の設定が重要となる。

本研究ではタスクが一定の期間連続して行われることを仮定し、$k$-means クラスタリングを用いて状態の自己遷移確率がある程度大きくなるような初期化を行う。初期化は以下の 3 ステップで行う。

1. 類似する隣接するセグメントを結合することで観測値系列 $\boldsymbol{O}$ を $L$ 個のセグメントに分割する。
2. セグメントを $k$-means クラスタリングを用いてクラスタリングする。
3. 各クラスタを別々の状態とみなして HMM の初期パラメーターを決定する。

ステップ 1 では $\boldsymbol{O}$ 上の各観測値を $\boldsymbol{I} = (o_1, \ldots, o_T) = (I_1, \ldots, I_T)$ のように細分化した 1 つのセグメントとしてスタートし、それぞれのセグメント $I_i$ に対して次のセグメントと結合した場合の結合コストを計算する。これら結合コストが最小となるセグメントのペアを結合することで新たなセグメントを作成する。セグメント $I_i$ と $I_{i+1}$ を結合したセグメントを $I_i \circ I_{i+1}$ とする。このとき、結合コストは以下のように再帰的に定義する。

$$\mathrm{cost}(I_i, I_{i+1}) = -\log_2(P(I_i \circ I_{i+1} \mid M'_{i,i+1})) + \log_2(P(I_i \mid M'_i)) + \log_2(P(I_{i+1} \mid M'_{i+1}))$$

ここで $-\log_2(P(I_i \mid M'_i))$ はセグメント $I_i$ をモデル $M'_i$ を用いて記述するための記述長 (ビット数) を表す。モデル $M'_i$ は $I_i$ から推定される各アプリケーションの出現頻度である $(P_i(e_1), \ldots, P_i(e_m))$ で表現される。各 $P_i(e_j)$ は $n(e, I)$ をセグメント $I$ 中に $e$ が観測された回数とすると $P_i(e_j) = n(e_j, I_i)|/|I_i|$ で計算できる。同様に $n(\bar{e}, I)$ はセグメント $I$ 中に $e$ が観測されなかった回数である。($M'_{i,i+1}$ も同様にセグメント $I_i \circ I_{i+1}$ から推定される。)

$$P(I_i \mid M'_i) = \prod_{e \in \mathcal{E}} P_i(e)^{n(e,I_i)} (1 - P_i(e))^{n(\bar{e},I_i)}$$

この結合処理は最小の結合コストが与えられた閾値以上 (実験的に 8.0 を用いた) になるまで繰り返される。

ステップ 2 において、各セグメント $I_i$ は $k$-means クラスタリングによって状態 $s \in S$ が付与される。クラスタリングを行う際に 2 つのセグメント $I_i$ と $I_j$ の距離を測る関数として対応するモデル $M'_i$、$M'_j$ に対する対称 Kullback-Leibler divergence を用いる。

$$\mathrm{dist}(I_i, I_j) = \sum_{e \in \mathcal{E}} P_i(e) \log \frac{P_i(e)}{P_j(e)} + \sum_{e \in \mathcal{E}} P_j(e) \log \frac{P_j(e)}{P_i(e)}.$$

得られたクラスタリング結果において各観測値 $o_t$ に対してラベル付けされた状態を $c_t \in S$ とする。

ステップ 3 ではステップ 2 で得られた結果を基に HMM の初期パラメーターを決定する。$D(s)$ をクラスタ $s$ に割り当てられた観測値の集合とし、$n(e, D(s))$ を $D(s)$ でイベント $e$ が観測された回数とする。

まず、タスク $s$ のタスク要約 $M_s = (P_s(e_1), \ldots, P_s(e_m))$ は $P_s(e) = n(e, D(s))/|D(s)|$ から決定される。次に $C(s)$ を $\{t \in \{1, \ldots, T\} \mid c_t = s\}$ とし、HMM の初期パラメーターを、それぞれ

$$\pi_s = \frac{D(s)}{T}, \qquad a(s_i, s_j) = \frac{|C(s_i) \cap C(s_j)|}{|C(s_i)|}, \qquad b_s(o) = \prod_{e \in \mathcal{E}} q_s(e, o)$$

とする。ここで、

$$q_s(e, o) = \begin{cases} P_s(e) & \text{if } e \in o \\ 1 - P_s(e) & \text{otherwise.} \end{cases}$$

である。この初期パラメーターからスタートし、Baum-Welch アルゴリズム [4] を用いてパラメーターを推定する。

## 3.4 ユーザー行動モデルの比較

前章で述べた方法で得られるにユーザー行動モデルはエルゴード的マルコフモデルである。本章ではこの抽象度でモデルを比較するための類似度の測定方法、及び可視化、モデルのクラスタリング手法について述べる。

ユーザー行動モデルを各頂点がタスク要約でラベル付けされ、各辺がタスク間の遷移確率でラベル付けされた有向グラフであるとし、このグラフをタスク遷移グラフと呼ぶ。2 つのタスク遷移グラフ間の類似度として、Kashima らが提案したラベル付きグラフ間の周辺化カーネル [2] を用いる。

このカーネル関数は、直感的には 2 つのグラフ上を 2 人のランダムウォーカーが移動した軌跡を比較し、その軌跡の類似度をグラフの類似度とするものである。ランダムウォーカーはある頂点を出発し、辺の遷移確率に従って次の頂点へと移動する。$\tau$ ステップ後にはランダムウォーカーは $\tau$ 個のタスクを経て移動を終了する。タスクの各系列は系列内のタスク間遷移確率の全ての積の形で重みづけされる。カーネル関数は 2 つのグラフ上の長さが同じ全ての重みづけされた系列同士に対して類似度を計算し、長さが 1 から無限大に対してその類似度を足し合わせることでグラフ間の類似度を求める。

タスク遷移グラフを $G = (S, E)$ で表わす。$S$ は頂点集合であり、頂点はタスク遷移グラフのタスク要約と一致する。また、$E \subseteq S \times S$ は有向辺の集合である。長さ $\tau$ のタスク系列はグラフ $G$ 上のウォーク $\boldsymbol{S} = (v_1, \ldots, v_\tau)$ で表記する。頂点 $v_i$ と $v_{i+1}$ は有効辺 $(v_i, v_{i+1}) \in E$ で結ばれている。また、グラフ $G$ 上の長さ $\tau$ のウォーク (タスク系列) の集合を $\boldsymbol{S}^\tau(G)$ と表記する。

$G$ のランダムウォークによってタスク系列 $\boldsymbol{S} \in \boldsymbol{S}^\tau(G)$ が得られる確率は

$$P(\boldsymbol{S} \mid G) = \pi_{v_1} \left( \prod_{i=2}^{\tau} a(v_{i-1}, v_i) \right) \pi_{v_\tau}$$

となる。ここで、ランダムウォーカーがウォークを頂点 $v_1$ から開始する確率を $\pi_{v_1}$ とし、頂点 $v_\tau$ で終了する確率を $\pi_{v_\tau}$ とする。$a(v_{i-1}, v_i) \in E$ は頂点 $v_{i-1}$ から頂点 $v_i$ への遷移確率である。本手法ではタスクの開始と終了にそれぞれ $v_1 = s_1$、$v_\tau = s_K$ を割り当てる。

2 つのタスク遷移グラフ $G_1, G_2$ 間の類似度 $K(G_1, G_2)$ は、次式で計算できる。

$$K(G_1, G_2) = \sum_{\tau=1}^{\infty} \sum_{\boldsymbol{S}_1 \in \boldsymbol{S}^\tau(G_1)} \sum_{\boldsymbol{S}_2 \in \boldsymbol{S}^\tau(G_2)} k(\boldsymbol{S}_1, \boldsymbol{S}_2) P(\boldsymbol{S}_1 \mid G_1) P(\boldsymbol{S}_2 \mid G_2),$$

ここで $k(\boldsymbol{S}_1, \boldsymbol{S}_2)$ はタスク間の遷移確率を無視した際のタスク遷移の類似性である。

Kashima ら [2] は不動点計算により上記のグラフ間類似度を効率よく計算する方法を提案した。このグラフ間類似度を用いて、HMM により求めたマルコフ遷移モデル間の類似度を計算する。ベースとなるカーネル関数として、$\boldsymbol{S}_1 = (v_1, \ldots, v_\tau)$ と $\boldsymbol{S}_2 = (v'_1, \ldots, v'_\tau)$ の 2 つのタスク系列間の類似度を次のように定義する。

$$k(\boldsymbol{S}_1, \boldsymbol{S}_2) = \prod_{i=1}^{\tau} k_s(v_i, v'_i),$$

ここで、$k_s(v_i, v'_i)$ はタスク要約 $M_s = (P_s(e_1), \ldots, P_s(e_m))$ と $M_{s'} = (P_{s'}(e_1), \ldots, P_{s'}(e_m))$ 間の類似度である。状態 $s$ と $s'$ がそれぞれ $v_i$ と $v'_i$ に対応するとして、$k_s(v, v')$ をタスク要約同士のコサイン類似度として以下のように計算する。

$$k_s(v, v') = \frac{\langle M_s, M_{s'} \rangle}{||M_s||\ ||M_{s'}||}$$

$M_s$ の要素各 $P_s(e_i)$ は頻度であり、$P_s(e_i) \geq 0$ であることから $k_s(v, v') \geq 0$ を満たす。このようにして、2 つのユーザー行動モデルを表す状態遷移グラフ $G_1, G_2$ の類似度としてカーネル関数 $K(G_1, G_2)$ が計算できる。このカーネル関数を用いて、カーネル主成分分析 [6] を行い、各モデルの主成分得点をプロットすることで、ユーザーモデルクラスタリングおよび可視化を試みた。

## 3.5　分析結果と考察

実際に日本の IT 企業からウィンドウ遷移ログを取得し、提案手法に基づいてログを分析した。

### 3.5.1　対象ログ

ログを収集した企業はソフトウェアの設計、開発、製造、販売を行う日本のソフトウェアメーカーである。各従業員は各自専用のデスクトップ PC とユーザーアカウントを使用している。分析では平日の 8:00 から 21:00 までのログを対象にした。

対象企業全体のアプリケーション使用率を計算することにより上位 8 種類のアプリケーションをイベントとして用い、これら以外のアプリケーションは “Others” として合計 9 つのカテゴリを使用した ($m = 9$)。

$$\mathcal{E} = \{\text{Mail, Web, Explorer, Word, Excel, PowerPoint, Editor, Viewer, Others}\}$$

これらのうち “Mail”、“Web”、“Editor”、“Viewer” はアプリケーショングループであり、それぞれ業務上果たす役割が同一であると考えられるアプリケーションをまとめたものである。例えば “Web” はウェブブラウザの集まりであり、Internet Explorer や Firefox 、Google Chrome などが含まれる。

### 3.5.2　HMM によるタスク遷移の推定

ログから HMM を用いてタスクの遷移モデルを推定するにあたり、$K = 12$ の隠れ状態を用いた。この内、状態 1 と状態 12 は開始と終了状態であり、一日の中で最初にログが記録された直前と最後にログが記録された直後の時刻に挿入されている。また、状態 2 はログがない状態である。これらを含め、状態に対応するタスク要約は全てのユーザーで共有する。

図 3 の上図はあるユーザーのログから推定された HMM の状態遷移図の抜粋である。下図は定常状態分布から得られる上位 2 つのタスク要約を可視化した図である。

状態 7 と状態 10 のタスク要約をみると、その違いは Mail の頻度であり、他のアプリケーションでは顕著

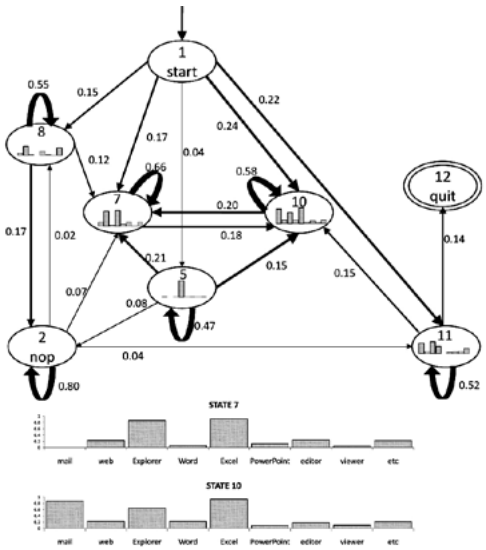


図 3　タスク遷移図 (上); 定常状態分布上位 2 つのタスク要約 (下)

な差は見られない．これらからこのユーザーは業務において Explorer と Excel をメインに使用し、時折メールを使用しているということが予想できる。

### 3.5.3　カーネル PCA を用いたユーザー行動モデルのクラスタリング

従業員 60 人に対し、各ユーザー 1 月毎のログからユーザー行動モデルを作成し、前章でのべたグラフカーネルに基づきカーネル主成分分析を適用した。図 4(上) にカーネル主成分分析で得られた主成分得点をプロッ

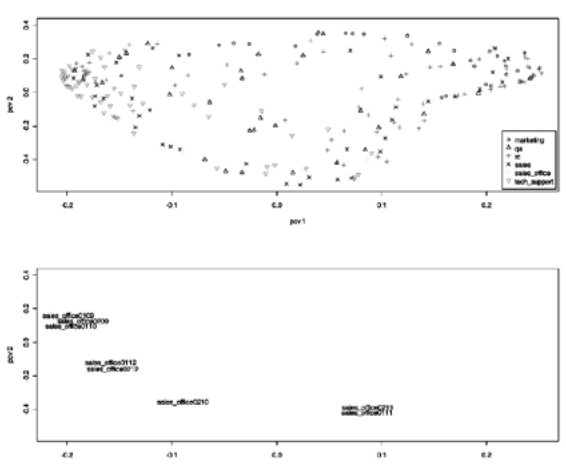


図 4　カーネル PCA で得られた各ユーザー月ごとのプロット (上) ; 共同で業務を行う 2 人のプロット (下)

トした様子示す。各プロットは広範囲に分布していることが分かる。図 4(下) は同一部署に所属する 2 人のロ

表 3　クラスターと実際の業務

| No | 日付 | | 業務 |
|---|---|---|---|
| 1 | 11/1 | Mon | 一般 |
| 1 | 11/5 | Fri | 一般 |
| 1 | 11/26 | Fri | 一般 |
| 2 | 11/2 | Tue | 一般 |
| 2 | 11/8 | Mon | 一般 |
| 2 | 11/12 | Fri | 一般 |
| 2 | 11/18 | Thu | 一般 |
| 2 | 11/24 | Wed | 一般 |
| 2 | 11/29 | Mon | 一般 |
| 3 | 11/4 | Thu | 校正 |
| 3 | 11/10 | Wed | 校正 |
| 3 | 11/11 | Thu | 校正 |
| 3 | 11/19 | Fri | 校正 |
| 3 | 11/25 | Thu | 校正 |
| 4 | 11/9 | Tue | 原稿作成 |
| 4 | 11/15 | Mon | 原稿作成 |
| 4 | 11/16 | Tue | 原稿作成 |
| 4 | 11/17 | Wed | 原稿作成 |
| 4 | 11/22 | Mon | 一般 |
| 4 | 11/30 | Tue | 一般 |

グから生成されたユーザー行動モデルを選択的にプロットした図である。10 月を除いて各月の 2 人のプロットは非常に近くに配置されていることが分かる。また、各月において配置される位置が異なることが読み取れる。

この 2 人の従業員は 9 月に入社し新人研修を行い、10 月に OJT と業務の引き継ぎを行い、11 月から実際の業務に従事している。また、この 2 人は研修期間を含め、同一の業務を分担および共同で行っている。

これらの事実は図 4 から読み取れることと一致すると言え、本提案手法により業務の変化と類似性が抽出可能であることを示唆している。

最後に、USER1 の 4 か月分 (80 日) のログに対して、1 日ごとのユーザー行動モデルを作成し、カーネル PCA を用いてクラスタリングを行った。図 5（右）は第 1 主成分と第 3 主成分に対してプロットを行ったものである。これらプロットがはっきりと 4 つのクラスターに別れていることが確認できる。図 5（左）はアプリケーションの使用率を通常の PCA を用いてプロットしたものであり、明確なクラスタに分かれていないことが分かる。このようにユーザー行動モデルを作成し比較することで、抽象度の高いユーザーの振る舞いが抽出できていることがわかる。

USER1 は非定形業務従事者であり、プロットから 4 つの業務パターンがあることが予想出来る。他の従業員ではこの様にはっきりとクラスタが分かれる場合があるが、そうでない従業員も多数おり、これは定型、非定形で区別が無い。カーネル PCA の結果と実際の業務の関連を調べるため USER1 にヒアリングを行った。

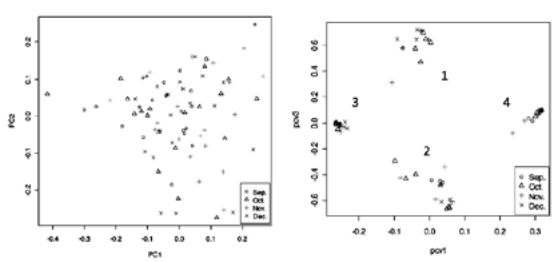


図 5　アプリケーション使用率のプロット（左）；日別のユーザー行動モデルのプロット (右)

表 3 はヒアリング結果の要約であり 11 月分のみ記載している。

“一般” は複数のタスクを含み、USER1 が意識的に特定の業務に集中していなかった日を意味する。また “校正” と “原稿作成” は一日を通して USER1 が同一の業務を行った日を意味する。この表よりクラスター 3 と 4 がそれぞれ “校正” と “原稿作成” と対応することが分かる。11/22 および 11/30 は “一般” であるが複数の業務の中で “原稿作成” に含まれるタスクが多くあったことが推定できる。また、クラスター 1 と 2 はともに “一般” であるが、クラスター 1 は全ての月において月曜日と金曜日が割あたることが多い。これに関してはヒアリングから USER1 は月曜日と金曜日に定例会議や外出が多いことが原因として考えられる。このように、提案手法により作業ウィンドウ遷移ログから具体性の高いタスクパターンが抽出できることがわかる。

## 4 おわりに

本報告では、Web サーバのログと、PC のウインドウ遷移ログの 2 つのログを対象に、分析の方法と結果について報告した。いずれの手法でも、キーとなるのは、膨大なログを抽象化し要約するためのモデル化である。低レベルのログから、抽象度の高いユーザーの振る舞いを抽出するためのモデルについて、現在、さらに研究を進めている。一方、システムの保守管理の観点からは、通常と異なる振る舞いを発見する「異常検知」が重要である。ウインドウ遷移ログの分析では、ユーザーのランダムなウインドウの切替タイミングを負の二項分布によりモデル化し、この分布からの距離によって、ユーザーの業務タイプの判定やタスクへの集中の計量化を試みている。

## 参考文献

[1] H. Kashima and T. Koyanagi. Kernels for semi-structured data. In *Proc. of 9th International Conference on Machine Learning (ICML)*, pages 291–298, 2002.

[2] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of 20th International Conference on Machine Learning(ICML)*, pages 321–328, 2003.

[3] M. Meiss, J. Duncan, B. Gonçalves, J. J. Ramasco, and F. Menczer. What's in a session: tracking individual behavior on the web. In *Proceedings of the 20th ACM Conference on Hypertext and Hypermedia*, pages 173–182. ACM, 2009.

[4] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

[5] R. Saito, T. Kuboyama, Y. Yamakawa, and H. Yasuda. Understanding user behavior through summarization of window transition logs. In *Databases in Networked Information Systems - 7th International Workshop, DNIS*, volume 7108 of *Lecture Notes in Computer Science*, pages 162–178. Springer, 2011.

[6] B. Schölkopf and A.J. Smola. *Learning with kernels*. “The” MIT Press, 2002.

[7] K. Shin, M. Cuturi, and T. Kuboyama. Mapping kernels for trees. In *Proceedings of the 28th International Conference on Machine Learning, ICML*, pages 961–968. Omnipress, 2011.

[8] M. J. Zaki and C. C. Aggarwal. Xrules: An effective algorithm for structural classification of xml data. *Machine Learning*, 62:137–170, 2006.