1K4-OS-07a-4

木構造操作としての計算機操作の学習

Learning computer manipulations as tree structure manipulations

秦野 亮 *1　東条 敏 *1
Ryo Hatano　Satoshi Tojo

*1北陸先端科学技術大学院大学 情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology

Recent studies show that there are many algorithms to learn human activity on computer. But most of them had limited their subject of input/output e.g., spread sheet macro learning. In this study, we propose a method to learn user manipulation on computer without such limitation. We represent screen appearance of window management system in a tree structure. Thus, the manipulation for such subject means update of the screen in any time. In order to learn such manipulation, our method has a procedure to determine the position of each manipulation and result of them by solving 'node marking problem' and its application.

1. はじめに

近年，携帯機器を含む様々な計算機環境において，ウィンドウベースのグラフィカルなユーザーインターフェース (Graphic User Interface, GUI) が多く用いられている．この種類のインターフェースは，ユーザーが直感的に計算機の操作が可能なように，画面上に操作の指標となる要素を配置し，それらに対する操作のフィードバックをグラフィカルに提供するという特徴がある．例えば，画面上に存在するウィンドウ上のボタンやラベルなどの要素は，その時点で操作可能な対象 (及びその一覧)・説明を表し，それらに対する操作のフィードバックは任意の時刻が過ぎた後，画面上の変化として現れる *1．

本稿では，こうしたインターフェースに対する入力操作の規則性を「学習」する問題に着目する．このような学習問題は，自然言語処理から人間の行動の学習まで，分野も研究例も多岐に渡り存在する．例えば，ユーザーの操作履歴から，その規則性を分析・抽象化 (学習) する Task summarization[Saito 11] の研究や，さらにそうした結果を後で再利用可能にする事を目的とした Input prediction, Programming by Example(PbE)[Gulwani 12] の研究などが挙げられる．

一般に，こうした入力操作の規則性を「学習」するとは，ユーザーの計算機に対する行動に何らかの規則性を仮定し，そうした規則性を説明出来るような計算モデルを構成し，サンプルとなるデータの集合から学習手続きを通じてそのモデルのパラメータを決定する事を意味する．そのため，計算モデルの構成要素をどのように定義するかによって，得られる結果は大きく変わってくる．

従来の研究では，この計算モデルの構成要素について，何らかの制限を持つものが多い．例えば，先に挙げた [Elliott 05] や [Saito 11] の研究では，計算機への入力操作 (入力) または画面遷移 (出力) を表すいずれかの記号の集合を元に計算モデルを構成している．これは，入力・出力のペアを基本要素とする一般的な PbE の計算モデルの構成の仕方と比べると，特定の対象 (アプリケーション等) に依存しているわけではないものの，学習後に得られる結果は入力または出力いずれかのサンプルにのみ依存しており，計算モデルの構成要素に制限を持つといえる．また，[Gulwani 12] で言及されている多くの PbE の研究では，spread sheet など計算モデルが特定の対象に特化していることから，こちらも同様に制限を持つといえる．

こうした背景を踏まえ，本稿では，計算機画面上の時系列での各状態を木構造として扱うことでこうした制限を弱め，より広い対象に対する入力操作の集合を形式言語上の計算モデルとして構成し，学習する手法について述べる．制限を弱めるとは，入力，出力の両方の情報を用いること，また特定のアプリケーションに限定せず，計算機上で取り扱われる入出力の観測が可能なアプリケーションの集合を対象とする事を意味する．

2. 準備

2.1 GUI の状態に関する順序木表現

本節では，計算機の画面 (GUI) の状態に関する定義を与える．本稿では，一般的に計算機の画面及び内部の状態の制御に用いられるウィンドウベースの GUI を，Graphical Shell(ウィンドウ管理システム) と仮定する．実際のシステムでは，例えば Windows Explorer, X.Org などがこれに該当する．画面に描画される全ての要素 *2 が持つ，位置やサイズなどの描画に関する情報は，全てこのシステムが管理し，制御しているものとする．画面の状態は，ユーザーやプロセスからウィンドウ管理システムへの任意の時刻の入力に基いて，適切な処理がなされた後に更新される．ある時点での画面の状態に注目すると，画面上に描画される各要素は，表示位置に応じた平面座標系での位置関係と，他の要素との包含関係 (重なり) を持つ．こうした情報は，実際のシステムの内部では木構造 (またはグラフ構造) として取り扱われていることが知られているため，我々はこうした画面の状態をラベル付き根付き順序木と近似的に対応付けて取り扱う．

次に，本稿で用いるラベル付き根付き順序木 (labeled-rooted-ordered tree) の定義を与える．今，ある木を T としたとき，T 上の全てのノードを $Ver(T)$、T の根を $Root(T)$ とする．ノードはウィンドウやボタンなど画面上に描画される各要素に対応し，根はデスクトップに対応する．任意の 2 つのノード $x, y \in Ver(T)$ について，y が x の祖先であるとき，その関係は階層順序であり $x \leq y$ と記す．$x < y$ で，かつ y と x の間に

連絡先: 秦野 亮，北陸先端科学技術大学院大学，石川県能美市旭台 1-1，r-hatano@jaist.ac.jp

*1　携帯機器など，画面の大きさに制限がある場合でも，表示が多少工夫されるだけで同様の特徴を持つ

*2　ウィンドウや，ウィンドウ上に描画されるボタンやテキストなど

ノードが存在しなければ，y を x の親とよび，$Parent(x) = y$ と記す．また，あるノード x がトップレベルであるとは，そのノードが $Parent(x) = Root(T)$ を充たすときかつその時に限る．同様に，任意の 2 つのノード $x, y \in Ver(T)$ について，x が y の左にあるとき，その関係は兄弟順序であり，$x \preceq y$ と記す．任意の木についてのノードの全順序は以上の関係に基いて定義され，画面上に描画される順序もこれに対応するものとなる．$x \prec y$ であれば，x は画面の手前側に描画される要素に対応し，y は後ろ側に描画される要素に対応する．同様に，$x \prec y$ であれば，y に対応する要素は画面の手前側に描画され，x に対応する要素は後ろ側に描画される．また，ある木に属する全てのノードは根から左側に深さ優先の行き掛け順序 (pre-order) $n \in N$ によっても一意に識別する事ができる．この行き掛け順序に基いてノードを返す関数を，$v_T : N \to Ver(T)$ とする．後に定義される木に対する編集操作によって，木のノードの位置が変更されるときは，これら関係に基いて位置が変更される．次に，ノードに付されるラベルに関する定義を与える．まず，ウィンドウ及びその構成要素の型を示す識別子の集合をウィンドウ・クラスとよぶ．例えば，Button や ComboBox， Window といった文字列がこの要素に該当する．$Text$ を GUI に現れるテキストの集合とし，$WClass$ をウィンドウ・クラスの集合とする．今，$\Sigma \subseteq WClass \times (Text \cup \{\varepsilon\})$ を木の各ノードに与えられたラベルの集合とする．木 T 上の各ノードからウィンドウ・クラスへの関数を $w_T : Ver(T) \to WClass$ とし，各ノードからラベルへの関数を $l_T : Ver(T) \to \Sigma$ とする．このとき，Σ によって生成される全ての木の集合を $\mathcal{T}_\Sigma$ と定義する．ここで，あるノードから，そのノードを含む全ての子孫からなる部分木を返す関数を $subtree : Var(T) \to \mathcal{T}_\Sigma$ とする．任意の 2 つのノード $x_i, x_j \in Ver(T)$ が $x_i \leq x_j$ であるとき，ノード x_i と x_j 間のパスを以下の様に定義する：

$$WPath_T(x_i, x_j) := \langle\, w_T(x_i), ..., w_T(x_j) \,\rangle,$$
$$Path_T(x_i, x_j) := \langle\, l_T(x_i), ..., l_T(x_j) \,\rangle.$$

パス上の全てのノードは，上記のタプルに含まれているものとする．木の定義より，任意のノード間のパスは常に 1 つである．即ち，ノード x_i から x_j について 2 つ以上の道を持つことはない．本稿では，任意の 2 つの木 T 及び T' とそれらのノード $x, y \in T, u, v \in T'$ について，2 つのパスについての Identity relation を以下の様に定義する：

$$WPath_T(x, y) = WPath_{T'}(u, v)$$
$$\overset{\text{def.}}{\iff} (w_T(x) = w_{T'}(u)) \text{ かつ} \cdots \text{ かつ } (w_T(y) = w_{T'}(v)),$$

$$Path_T(x, y) = Path_{T'}(u, v)$$
$$\overset{\text{def.}}{\iff} (l_T(x) = l_{T'}(u)) \text{ かつ} \cdots \text{ かつ } (l_T(y) = l_{T'}(v)).$$

このとき，$x_i, x_j \in \mathcal{T}_\Sigma$ となる全てのノードのペアから生成されるパスの集合を，それぞれ $WPath_{\mathcal{T}_\Sigma}, Path_{\mathcal{T}_\Sigma}$ と定義する．

2.2 GUI への入力操作とその出力に関する記号表現

本節では，GUI への入出力に関する記号の集合を定義する．一般に，GUI の核となるウィンドウ管理システムは，イベント駆動システムとして設計される事が多い．この様なシステムは，ある入力 (メッセージと呼ばれる) が与えられると，それに対応する内部的な手続き (イベントと呼ばれる) を経て出力を返す仕組みを持つ．メッセージとは，任意の時刻でユーザーや計算機上の他のプロセスにより生成される，システムの制御に関わる一連のデータの事である．これらのデータには，例えばユーザーからのマウス・キーボードの操作命令や入力対象などの補足情報，画面の更新命令などが含まれている．

以下ではまず，この入力となる記号に関する定義を与える．本稿では，計算機への入力操作の規則性を学習する問題を取り扱うため，ここで定義する入力に関する記号の集合はマウス・キーボードによる入力操作に関連するもののみと仮定する．今，入力操作の識別子の集合を Op_{in} とし，入力操作と関連付けた値の集合を $Attr$ とする．ここで，入力操作の対象となる画面上の位置に対応した順序木上の位置を返す関数の集合を Pos とすると，ウィンドウ管理システムへの入力を表す記号の集合 I は以下の様に定義される：

$$I \subseteq (Op_{in} \times Attr) \times Pos.$$

入力操作の識別子の集合 Op^{in} は，例えば左クリックを意味する $WM\_LButtonDown$ の様な，ウィンドウ管理システム上で異なる操作として識別される単位の要素の集合が対応する．同様に，入力操作と関連付けられる値の集合 $Attr$ も，仮想キーコードや修飾キー [*3] との組み合わせなど，各入力操作に付加される補助的な情報の集合が対応する．本稿では，簡単のためそうした値の集合をキーボードのものにのみ制限して定義する．今，$KeyCode$ を仮想キーコードの集合，$Modifiers$ を修飾キーの集合とした時，集合 $Attr$ を以下の様に定義する：

$$Attr \subseteq (KeyCode \times Modifiers) \cup \{\varepsilon\}.$$

$Pos = N \cup Classifier$ は，本稿では学習時も考慮して，行き掛け順序の番号または 2 値分類をする関数 (分類器) とする．分類器とした理由は，2.4 節で示すノードマーク問題を解く都合上必要であるからである．本来，分類器は学習時，予測時でそれぞれ異なる関数を持ち，分類パラメータなど煩雑な表記を必要とするが，ここでは学習済みの分類器と関連した予測用の関数と仮定し，簡略な表記にて用いる．入力操作やその出力を識別するにあたって問題となるのは，画面上でそれぞれが対象とする位置の同一性を如何に判定するかである．このような問題は，例えば以下に示すような，学習時と予測時でノードの名前が微妙に変わっていたり，似たような操作対象が複数存在するようなケースにおいて特に問題となる [*4]．

学習時 $(a\ (x\ (y\ b)\ c)) \Rightarrow (a\ (c))$
予測時 $(a\ (x\ (z\ b)\ x'\ (y\ b)\ c)) \Rightarrow (a\ (x\ (z\ b)\ c))$

こうしたケースは，ウィンドウの入れ替えであったり，エディタのウィンドウにファイル名が含まれている様なケースであったり実に様々である．そこで本稿では，こうした位置の指定に用いる分類器について，同値関係とマージの操作とを定義する．今，学習済み分類器を $f : X \to Y, g : X' \to Y'$, としたとき，これら 2 つの分類器が同値である事を以下の様に定義する：

$$f =_{X \cap X'} g \overset{\text{def.}}{\iff} \forall_{x \in X \cap X'} f(x) = g(x)$$

ただし X, X' を学習に用いたサンプルデータのドメイン, Y, Y' をその値域，$X \cap X' \neq \emptyset$ とする．次に，2 つの分類器のマージに

*3 ALT, Ctrl, Shift キーなど，補助的な制御に用いられるキーを指す．

*4 S 式「風」の表記を用いて木構造を表現している．Lisp などのプログラミング言語においては，データを表す S 式 (リスト) の頭にクォート記号を付ける慣習があるが，ここではあくまでも S 式風ということでそれらは省略した．

ついて定義する．今，$f : X \to Y,\ g : X' \to Y' s.t., f =_{X \cap X'} g$ とする．ここで，分類器のマージ $f \cup g : X \cup X' \to Y \cup Y'$ を，f, g 両方の分類器の学習に用いたサンプル $X \cup X'$ を用いて，$Y \cup Y'$ の出力を持つようにしたものと本稿では定義する．直観的には，2 つの学習済みの分類器が，互いが持つ全てのサンプルデータを交換して予測して同様の結果が得られる場合に，互いのサンプルデータを併せた新しい分類器を構成するという事に対応する．

次にウィンドウ管理システムへの入力によって生じる出力に関する記号の集合を定義する．ここで定義する記号の集合は，外部 (ユーザー) からウィンドウ管理システムが公開している情報 (画面) を通じて観測可能なもののみと仮定する．即ち，ウィンドウ管理システムへのある入力に対応したある出力とは，画面の状態に対応した順序木上の変化として現れるものとする．今，$\mathcal{T}_\Sigma$ の任意の要素に対する編集操作の集合を $Op_{out} \subseteq \{Ins, Del, Repl\}$[*5] とし，$Pos$ を画面上の変化が生じる位置に対応した順序木錠の位置を返す様な関数の集合とする．このとき，画面の変化を反映した順序木に対する出力を表す記号の集合 O を以下の様に定義する：

$$O \subseteq (Op_{out} \times \mathcal{T}_\Sigma) \times Pos.$$

そして，これまでの定義を用いて入力と出力のペアの集合は $I \times O$ として定義される．このペアの要素について，本稿では入力は常にひとつという仮定を置くが，出力は複数の箇所で生じる可能性があるため，必ずしも一つとは限らない．

また，入出力の和集合 $I + O$ を以下の様に定義する：

$$I + O \subseteq Op \times Val \times Pos$$

ただし，$Op \subseteq Op_{in} \cup Op_{out}$, $Val \subseteq Attr \cup \mathcal{T}_\Sigma$ とする．

2.3 射影と Identity relation

今，X を要素を任意の集合とする任意の集合とし，$x = \langle x_1, \ldots x_n \rangle \in X$ とする．このとき，$1 \leq i \leq n$ について，p_i を i 番目の要素を取り出す射影として以下の様に定義する：

$$p_i^X(x_1, \ldots, x_i, \ldots x_n) = x_i$$

次に，2 つの記号 $x = x_1, \ldots, x_n, x' = x'_1, \ldots, x'_n \in X$ を与えた時，Identity relation を次のように定義する：

$$x =_X x' \overset{\text{def.}}{\iff} \bigwedge_{i \in \{1,\ldots,n\}} (p_i^X(x) = p_i^X(x'))$$

本稿ではこれ以降，$I + O$ など，本稿で定義した任意の集合についても上記の射影と関係が存在するものとして扱う．

2.4 ノードマーク問題

本節では，画面上のある要素に対応する順序木上のノードの位置を，分類器によって指定するために解くべき問題について説明する．ノードマーク問題 [Kashima 02] とは，いくつかのノードがマークされた例となる木の集合から，木のどの部分をマークすべきかを学習する問題のことである．[Kashima 02] では分類器，特に順序木カーネルを用いたアプローチについて述べており，本稿で行う各入力操作の対象の位置，及び各出力の発生位置を特定する手続きにおいても，同じ問題を取り扱っている．木カーネルとは，学習器のクラスのひとつであるカーネル法 [Shawe-Taylor 04] において用いられる，カーネル関数という 2 つのデータの内積を計算する関数を木構造に対応させたものである．以下に，上記の文献から引用した，ノードのマーク位置に関する分類器の学習手続きを示す：

1. 正しくマークされた木の集合 T_p を入力する
2. T_p から，誤ってマークされた木の集合 T_n を構成する
3. 正例の集合 E_p を T_p から構成し，同様に負例の集合 E_n を T_n から構成する
4. E_p, E_n を用いてカーネル分類器の学習を行う

ここでマークをするとは，特定の位置を表したいノードの親として，木に含まれていない特殊なノードを埋め込んで分類器の目印にする事をさす．次節で，この手続きは $Mark : N \times T \to Classifier$ として参照する．学習後にマークを行う手続きやカーネルの定義など，他の詳細は [Kashima 02] を参照されたい．

3. 入力操作とその出力に関する集合の学習

本節では，前節までに定義した各種記号を用いて，入力操作とその出力に関する記号の集合の学習手続きについて述べる．本節で定義される学習手続きは，以下の様な関数 f とみなせる．

$$f : I \times \mathcal{T}_\Sigma \to I \times O$$

おおまかに手続きの方針を述べると，入力操作，出力それぞれが一意となる様にサンプル同士を比較して，クラスタリングに相当する事を行う．しかし，ナイーブにサンプル同士の比較を行うと，膨大な組み合わせが生じてしまうため，予めわかっている同値関係をヒントに，組み合わせの数を減らしていくという方針を取る．特に留意が必要な点は，我々から見て画面上の同じ位置に入力・出力しているはずの操作であっても，サンプルによっては木が大きく異なってくるケースが存在するという点である．学習手続きを通じて，画面上の入力・出力の位置は分類器によって指し示せるようになるが，問題はその後，同じ入力・出力とみなせるものについては分類器同士が本当に同じものを指しているということを保証しつつ，クラスタリングを進めていかなければならないという点である．

3.1 サンプル，トレーニングデータの構成

本節では，学習手続きの入力となるデータの定義について述べる．今，I を入力操作の集合，$\mathcal{T}_\Sigma$ を木の集合とする．このとき，観測可能な全てのサンプルの集合 S を以下のように定義する：

$$S \subseteq I \times \mathcal{T}_\Sigma$$

今，任意のサンプル $s_1, \ldots, s_n \in S$ を与えた時，サンプルのリストを

$$C_r := \langle s_1, \ldots, s_n \rangle = \langle \langle in_1, T_1 \rangle, \ldots, \langle in_n, T_n \rangle \rangle.$$

とする．ただし，入力操作のサンプルの Pos の要素は，入力対象となった画面上の要素に対応する，順序木上のノードの行き掛け順序での識別子 (番号) とする．次に，訓練データの定義について述べる．

$$TS(Cr) = \{\langle op_i, attr_i, n_i, T_i, T_{i+1} \rangle : 1 \leq i < n, \langle op_i, attr_i, n_i \rangle \in I, T_i \in \mathcal{T}_\Sigma\}$$

3.2 入力記号の集合の学習

1. まず，入力操作の識別子，入力操作と関連する値，入力対象となったノードのラベル，デスクトップから入力対象まで

*5 Insert, Delete, Replace の操作を意味する．

のウィンドウクラスでのパスが等しい集合を構成し，ノードマーク手続きを以下の定義に従って行う：

$$D_I(op, attr, label, wpath) =$$
$$\{Mark(p_4^{TS}(w), p_5^{TS}(w)) \in Classifier :$$
$$p_1^{TS}(w) = op \text{ かつ } p_2^{TS}(w) = attr \text{ かつ}$$
$$l_{T_i}(p_3^{TS}(w)) = label \text{ かつ}$$
$$WPath_{T_i}(Root(p_4^{TS}), V_{T_i}(p_3^{TS}(w))) = wpath\}$$

2. 任意の $f : X \to Y, g : X' \to Y' \in D_I \subset Classifier$ について，$f =_{X \cap X'} g$ のとき，マージした分類器 $f \cup g$ を構成する．
3. $D_I = D_I/f, g \cup (f \cup g)$ として，これ以上マージができなくなるまで 2-3 を繰り返す．

3.3 出力記号の集合の学習

$TS'(Cr)$ を，前節で学習した部分について置換した集合とする．まず，以下に木の差分を求める手続き $TDiff : T \times T \to O$ の概要を示す．$u \in T, u' \in T'$，$O_T D = \emptyset \subset O$ のとき，以下を繰り返す：

$u = v$　次のノードの比較を行う

$u \neq v$　u が変化が生じたノードである場合，$O_T D = O_T D \cup \langle op, Mark(u, T), subtree(u) \rangle$ とする．ただし，$op \in Op_o ut$ であり，変化の内容によっていずれか一つが選択される．$x \in subtree(u), y \in subtree(v)$ となるノードの比較はスキップし，スキップするノード数の差について比較再開の位置を調整した後に，次の差分を求める手続きを行う．

次に，出力記号の学習手続きに付いて述べる．

1. 入力操作が等しい集合を構成する：
$w = \langle op, attr, c, t, t' \rangle / inTS'(Cr)$ としたとき，
$$D_{o_1}(op_i n, attr, c_i n, t, t') = \{TDiff(t, t') :$$
$$p_1^{TS}(w) = op_i n \text{ かつ } p_2^{TS}(w) = attr \text{ かつ } p_3^{TS}(w) = c\}$$
2. 出力の集合が等しくなる集合を構成する：
$w = w_1, \ldots, w_n \subset D_{o_1}, X \subset O$ としたとき，
$$O^-(X) = \{\langle p_1^{I+O}(x), p_2^{I+O}(x) \rangle : x \in X\}$$
$$D_{o_2}(w_1, \ldots, w_n) = \{w \in O :$$
$$\forall_{x \in O}\{x \in w_1, \ldots, w_n \leftrightarrow x \in w\}\}$$
[*6].
3. 個別の出力が等しくなる集合を構成する：$w = \langle op, t, c \rangle \in D_{o_2}$ としたとき，
$$D_{o_3}(op_o ut, st) = \{p^I + O_3(w) \in Classifier :$$
$$p^I + I + O_1(w) = op \text{ かつ } p_2^{I+O}(w) = st\}$$
4. 前節の 2-3 と同様の手続きを，D_{o_3} について繰り返す．

以上の手続きを経て以下の集合を得る：

$$V_T = I \times O$$

これが求めていた集合となる．

[*6] 少しでも異なる場合，他のプロセスからのノイズなのか本当に異なるのか判断がつけ難いため，確実に一致するもの同士で集合を構成する．

4. 構文規則の学習

学習した記号で置き換えたサンプルの列に対して，文法圧縮など，繰り返し現れる記号列を抽象化してまとめ上げる様な手続きを通すことで構文規則を学習することが出来る．例えば，[Hatano 13] では，Sequtiur というアルゴリズムを通すことにより，文脈自由文法の形で記号列の規則の集合を学習する手法を示している．

5. おわりに

本稿では，計算機画面上の時系列での各状態を木構造として扱うことで，人間の計算機への入力操作とその出力に対応する記号の集合を学習する手法について述べた．本稿では紙面の都合上，主に記号の集合を学習する手続きについてのみ述べたが，木構造上の位置を特定するために特別な配慮が必要であった他，分類器同士のマージなど他ではあまり取り扱わない問題にも触れた．今後の課題としては，分類器同士の比較・マージについて，アンサンブル学習など他の手法との組み合わせについても検討が必要と考えられる．また，計算機に限らないより広い問題設定としては，木構造ではなくより一般的なグラフ構造を取り扱うといった事も考えられる．今後の発展に期待したい．

参考文献

[Elliott 05] Elliott, F. and Huber, M.: Learning Macros with an Enhanced LZ78 Algorithm., in Russell, I. and Markov, Z. eds., *FLAIRS Conference*, pp. 412–417, AAAI Press (2005)

[Gulwani 12] Gulwani, S.: Synthesis from examples: Interaction models and algorithms, in *14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing* (2012)

[Hatano 13] Hatano, R. and Tojo, S.: Detecting Context Free Grammar for GUI Operation, *Proceedings of KICSS'2013*, pp. 126–137 (Kraków 2013)

[Kashima 02] Kashima, H. and Koyanagi, T.: Kernels for semi-structured data, in *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pp. 291–298 (2002)

[Memon 03] Memon, A., Banerjee, I., and Nagarajan, A.: GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing, Proceedings of the 10th Working Conference on Reverse Engineering (WCRE' 03) (2003)

[Saito 11] Saito, R., Kuboyama, T., Yamakawa, Y., and Yasuda, H.: Understanding user behavior through summarization of window transition logs, in *Databases in Networked Information Systems*, pp. 162–178, Springer (2011)

[Shawe-Taylor 04] Shawe-Taylor, J. and Cristianini, N.: *Kernel methods for pattern analysis*, Cambridge university press (2004)

[申吉 09] 申吉浩：木の半正定値カーネル フレームワークとサーベイ: フレームワークとサーベイ, 人工知能学会論文誌, Vol. 24, No. 6, pp. 459–468 (2009)