A formal Framework for positive and negative Detection Schemes

Ausarbeitung zum Seminarvortrag Künstliche Immunsysteme

Michael Schmitt
TU Darmstadt
2005

Einleitung

Ausgehend vom Vorbild des menschlichen Immunsystems stellt sich die Frage, wie man in technischen Systemen Abweichungen vom Normverhalten feststellen kann. Diese Abweichungen können von Fehlern bei der Ausführung von Programmen bis hin zu absichtlichen Angriffen auf vernetzte Rechnersysteme reichen. Zur Erkennung solcher Sicherheitsverletzungen kann man neben den heute bereits bekannten Verfahren möglicherweise neuartige Methoden verwenden, wie die r-contiguous-bit-Erkennung (rcb) oder die Methode der r-chunks, deren Anwendung jedoch auch nicht uneingeschränkt möglich ist. Verbunden mit der grundlegenden Anwendbarkeit neuer Methoden stellt sich zusätzlich die Frage nach deren Effizienz im Hinblick auf Geschwindigkeit und Platzbedarf, sowie weitere Fragen nach leistungsfähigeren Modifikationen für die Praxis.

Verwendete Begriffe und Definitionen

- U: ein Universum aller möglichen Zeichenketten einer bestimmten Länge l
- s: eine einzelne Zeichenkette aus U (der Länge l)
- RS: eine Menge aller echt systemeigenen Zeichenketten s („real self" oder kurz „self")
- S: eine Auswahl von Beispielen (Samples) aus RS
- sample(t): eine Auswahl von Beispielen aus RS zu einem Zeitpunkt t
- U - RS: die Menge der Elemente, die nicht im „self" liegen
- r: ein Ähnlichkeitsmaß zu den Elementen aus S, angegeben als maximal erlaubte Anzahl unterschiedlicher Bits
- Löcher: Zeichenketten, die aufgrund von Einschränkungen der Detektoren nicht erkannt werden können
- xMd: Relation, die angibt, daß ein Element x von einem Detektor d erkannt wird („Match")
- O.B.d.A. werden hier nur binäre Zeichenketten betrachtet

Biologischer Hintergrund

Grundlage der hier vorgestellten Anomalie-Erkennungsmethoden ist das menschliche Immunsystem und hierbei insbesondere die Systematik, mit der Antikörper, vor allem T-Zellen, darauf trainiert werden, körpereigene Zellen und Stoffe von Fremdstoffen und Krankheitserregern zu unterscheiden.
Diese Unterscheidungsfähigkeit erlangen die T-Zellen während der Reifungsphase, in welcher sie isoliert vom restlichen Kreislaufsystem in der Thymusdrüse verschiedene Wachstumsstadien durchlaufen. Während des Reifungsprozesses werden die T-Zellen lediglich mit körpereigenen Zellen konfrontiert. Andere Stoffe können nicht in die Thymusdrüse gelangen.
Findet beim Kontakt der heranreifenden T-Zellen mit den Körperzellen eine Immunreaktion statt, wird die entsprechende T-Zelle abgetötet. Die überlebenden Zellen erzeugen daher keine Immunreaktion. So wird sichergestellt, daß am Ende der Reifungsphase nur solche T-Zellen den Thymus verlassen, die auf körperfremde Zellen reagieren.

Übertragung auf technische Systeme

Um das biologische Erkennungssystem auf technische Systeme anwenden zu können, muß zunächst festgelegt werden, welche Kriterien eine Anomalie in einem technischen System charakterisieren. Grundsätzlich wird Anomalie hier mit einer Sicherheitsverletzung gleichgesetzt.
Technische Systeme, die dem Schutz vor Sicherheitsverletzungen dienen oder bei denen Sicherheitsverletzungen eine wichtige Rolle spielen sind Antiviren-Programme, Systemangriffe über unterschiedlichste Netzwerkprotokolle, wie TCP, UDP oder ICMP, sowie Mehrbenutzersysteme, deren Anwender unterschiedliche Rechte auf dem System haben und nur bestimmte Aktionen am System ausführen dürfen.

Arbeitsweise bisheriger Systeme

In den meisten der heute verwendeten Systeme kommen andere Methoden zum Einsatz, um Systemabweichungen festzustellen. Hiervon kann man zwei grundlegende Methoden herausgreifen: die Wahrscheinlichkeitsmethode, sowie die Abstandsmethode.
Bei der Wahrscheinlichkeitsmethode werden Systemanomalien reziprok zur Häufigkeit bestimmt, in welcher ein Ereignis in der Vergangenheit aufgetreten ist, also $\text{Verdacht} \approx \frac{1}{\text{Häufigkeit in der Vergangenheit}}$. Je häufiger somit ein Ereignis bereits früher aufgetreten ist, um so wahrscheinlicher ist es, daß dieses Ereignis dem Normverhalten entspricht.
Die Abstandsmethode hingegen legt die Verdächtigkeit eines Ereignisses proportional zum minimalen Abstand bekannter Elementen im Ereignisraum fest. Dieser Abstand unterscheidet sich je nach verwendeter Norm. Bei Verwendung der L-Norm wird das Merkmal betrachtet, das den kleinsten Abstand zu den bekannten Ereignissen besitzt. Der Hamming-Abstand hingegen betrachtet die Anzahl der sich unterscheidenden Zeichen bzw. Bits in einer

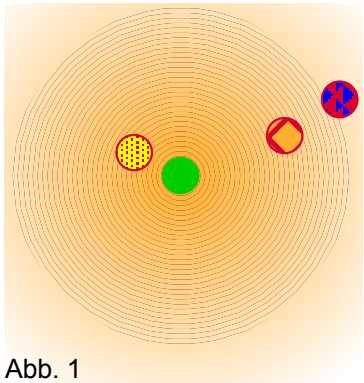


Abb. 1
Je geringer der Hamming-Abstand zu bereits bekannten Objekten ist (d.h. je näher es am Zentrum liegt), umso eher wird ein neues Objekt dem self (großer Kreis) zugeordnet.

Zeichenkette und ähnelt dem Manhattan-Abstand, wenn man jede Bit-Position als eine Dimension ansieht. Dies ist anschaulich in Abb. 1 dargestellt.
Die beiden Ansätze lassen sich jedoch auch kombinieren, evtl. mit zusätzlichen Erkennungsmethoden.
Betrachtet man die Einsatzgebiete heutiger Systeme, so handelt es sich dabei meist um genau bekannte Systeme von geringem Umfang, konstanter Aufgabenstellung und Erkennungsmenge.

Funktionsweise der Anomalie-Erkennung

Um Anomalien erkennen zu können, bedarf es zunächst einer Spezifikation des Normverhaltens eines Systems. Dieses Normverhalten geht von einem Modell aus, das wiederum mittels Konzeptlernen aus Beispielen (Samples) erschlossen wird. Problematisch ist hier die Größe der Beispielmenge; ist sie zu klein, kann dies zu einer Übergeneralisierung des aufgestellten Modells führen. In der Praxis ist das Problem, ein gutes Modell zu finden, häufig nicht lösbar, d.h. die Einordnung neuer Objekte ist nicht eindeutig bestimmt.
Zudem müssen Ähnlichkeiten zwischen Objekten sinnvoll definiert sein, da ansonsten die getroffene Klassifikation nur geringe Aussagekraft besitzt.
Benutzt man zur Identifizierung von Anomalien Partial Matching, kann die Erkennung Löcher enthalten. Diese Löcher enthalten Merkmalskombinationen, die durch keinen Detektor festgestellt werden können, weil dies zu einem Widerspruch mit bekannten Beispielen des self führen würde.
Ein entscheidendes Merkmal bei der Erkennung ist die Wahl der verwendeten Methode, d.h. ob man Elemente positiv oder negativ erkennt. Bestehende Systeme basieren hauptsächlich auf positiver Erkennung, bei der ein Detektor diejenigen Elemente erkennt, die im self liegen. Möglicherweise finden sich auch Bereiche, bei denen die negative Erkennung mehr Vorteile bringt als die positive. Eine genauere Definition findet sich im Abschnitt Erkennungsschemata.

Ähnlichkeitsmaße

Die Entscheidung, ob ein aktuell vorliegendes Objekt, eine Zeichenkette o.ä., eine Anomalie darstellt, hängt davon ab, wie ähnlich dieses Objekt den bereits bekannten Objekten ist. Dazu muß ein Ähnlichkeitsmaß definiert werden.
Als Maß bietet sich u.a. der kleinste Hamming-Abstand zu Elementen aus *sample(t)* an, also die geringste Anzahl an Zeichen, die sich von einer Zeichenkette aus *sample(t)* unterscheiden. Für Elemente aus *sample(t)* ergibt sich bei dieser Methode allerdings immer der Abstand Null.
Bewertet man die Ähnlichkeit reziprok zur Anzahl der Vorkommen der aktuell betrachteten Zeichenkette in der Beispielmenge, erhält man als weiteres Ähnlichkeitsmaß $\frac{1}{\text{Anzahl der Instanzen } p_k \text{ in sample(t)}}$. Diesmal jedoch mit den Schwierigkeiten, daß sich für nicht in *sample(t)* vorkommende Elemente der Bruch $\frac{1}{0}$ ergibt, welcher natürlich nicht lösbar ist. Daher legt man in diesem Fall die Ähnlichkeit auf eine große Konstante C fest. (Theoretisch würde eine Konstante größer als 1 bereits ausreichen, da die geringste Anzahl an Vorkommen 1 sein kann, wodurch sich die kleinste berechenbare Ähnlichkeit ebenfalls zu 1 ergibt.) Das zweite Problem hängt unmittelbar mit der eben angesprochenen Nicht-Berechenbarkeit zusammen. Da die Ähnlichkeit für Elemente, die nicht in *sample(t)* vorkommen, immer auf dieselbe Konstante gesetzt wird, lassen sich unbekannte Elemente nicht unterscheiden.

Um diese Probleme auszuschalten, bietet sich als weiteres Ähnlichkeitsmaß die Verwendung einer hybriden Methode an, die die beiden vorgenannten Methoden kombiniert und je nachdem, ob das aktuelle Objekt bereits in *sample(t)* vorkommt oder nicht, die Ähnlichkeit in Abhängigkeit der Anzahl der Instanzen oder nach dem Hamming-Abstand berechnet.
Im Folgenden wird als Ähnlichkeitsmaß lediglich eine zweiwertige Kategorisierung vorgenommen, die besagt, daß ein Objekt zur Beispielmenge *sample(t)* ähnlich ist oder daß es unähnlich zu *sample(t)* ist. Diese Unterscheidung reicht für die hier vorgestellten Detektormethoden aus und führt zu einer effektiveren Erkennung.

Erzeugungswahrscheinlichkeit

Ausgehend von einer Beispielmenge *S* werden neue Elemente mittels einer Erzeugungsregel *Q(S)* erzeugt. Diese Regel beinhaltet zwei Prozesse *G* und *B*, die jeweils neue Elemente erzeugen. Dabei erzeugt *B* stets Elemente, die im non-self liegen, während *G* nur Elemente aus dem self erzeugt. Formal werden die beiden Prozesse durch $P(G) = \{x(t)|y(t) = 0, \text{sample}(t), t\}$ und $P(B) = \{x(t)|y(t) = 1, \text{sample}(t), t\}$ dargestellt. $y(t) = 0$ bedeutet in diesem Zusammenhang, daß das neue Element durch den Prozeß G erzeugt wurde, während es bei $y(t) = 1$ durch den Prozeß *B* erzeugt wurde. Die Erzeugung selbst ist abhängig von den Elementen, die sich zum aktuellen Zeitpunkt in der Beispielmenge *sample(t)* befinden, was bedeutet, daß sich die Erzeugung ändert, wenn neue Elemente zu *sample(t)* hinzukommen. Um den Erzeugungsprozeß zu beschreiben, wäre es nötig, den zeitlichen Verlauf von *sample(t)* zu speichern, um das neu erzeugte Element berechnen zu können, was sich am Beispiel des Prozesses *G* wie folgt darstellen würde: $P(G) = \{x(t)|y(t) = 0, x(t-1), x(t-2), ..., x(0), t\}$. Die o.g. Formeln stellen also bereits eine Vereinfachung dar, weil bei diesen der zeitliche Verlauf der Elementerzeugung unberücksichtigt bleibt.
Da in der Praxis nicht bekannt ist, von welchem Prozeß ein Element erzeugt wurde, werden die Prozesse versteckt und es werden lediglich die Wahrscheinlichkeiten betrachtet, mit denen Elemente in *Q(S)* und außerhalb durch den Prozeß *P(G)* erzeugt wurden. Allgemein kann man die Erzeugungswahrscheinlichkeit durch $p = \frac{1}{Q(S)}$ausdrücken. Weil es wahrscheinlicher ist, daß *P(G)* Elemente aus *Q(sample(t))* erzeugt als Elemente außerhalb dieser Menge, verwendet man zwei Wahrscheinlichkeiten $p_1' = \frac{f}{|Q(sample(t))|}, x(t) \in Q(sample(t))$ und $p_2' = \frac{1}{|U - Q(sample(t))|}, x(t) \notin Q(sample(t))$, die angeben, daß *P(G)* wahrscheinlicher Elemente aus *Q(sample(t))* erzeugt als Elemente, die nicht in *Q(sample(t))* sind. Der Faktor f sagt aus, daß die Wahrscheinlichkeit für ein Element aus *Q(sample(t))* höher ist als für ein Element aus dem Komplement. Da i.A. im Komplement mehr Elemente enthalten sind, ist die Punktwahrscheinlichkeit für ein Element in *Q(sample(t))* größer als außerhalb. Ist dies nicht der Fall, muß der Faktor f gesondert angepaßt werden. Im Normalfall gilt natürlich $0 \leq p_2 < p_1 \leq 1$und $p_1 + p_2 = 1$.
Durch die Änderung von *sample(t)* im Laufe der Zeit ändern sich auch die Wahrscheinlichkeiten von *x(t)* in Abhängigkeit von neuen Elementen in *sample(t)*. Allgemein gilt für den Erzeugungsprozeß *G* $\Pr\{x(t)|y(t) = 0, \text{sample}(t), t\} = \left\{ \begin{array}{l} p_1, \text{ wenn } x(t) \in Q(sample(t)) \\ p_2, \text{wenn} x(t) \notin Q(sample(t)) \end{array} \right\}$.

Erzeugungs- und Erkennungsmethoden

Wie kann man nun erkennen, ob ein Element zur Menge self gehört oder nicht? Dazu bieten sich unterschiedlich Methoden an, von denen an dieser Stelle drei vorgestellt werden sollen.

Hamming-Abstand

Erkennung mittels des Hamming-Abstands besagt, daß ein Detektor d ein Element dann erkennt, wenn maximal T Stellen von d verschieden sind. Die durch d erkannten Elemente werden somit in ein Kategoriensystem mit Grenzwert T eingeteilt. In dem hier betrachteten Fall ist T als $l - r$ definiert.
$Hd_T(S)$ gibt dann die Menge der Strings an, die maximal den Abstand T haben, und es gilt $xMd \leftrightarrow Hd(x, d) \leq l - r$, d.h. ein Objekt x wird von einem Detektor d erkannt, wenn es sich in höchstens $l - r$ Positionen von d unterscheidet.
Verwendet man den Hamming-Abstand als Erzeugungsmethode, so bedeutet dies, daß durch einen Erzeugungsprozeß Elemente erzeugt werden, die ebenfalls maximal den Abstand T vom Ausgangselement haben.

n-Gramme

Eine zweite Erkennungsmethode, die v.a. bei zur Überprüfung von Programmen genutzt werden kann, sind n-Gramme. Sie funktionieren ähnlich wie die Crossover Closure im nächsten Abschnitt, mit der Vereinfachung, daß die n-Gramme unabhängig von einer Positionsangabe zur Prüfung herangezogen werden.
N-Gramme speichern gültige Programmzustände als eine Art Trace in einem gerichteten azyklischen Graph. Dieser Trace wird bei einer Beispielausführung zusätzlich zum ausführbaren Programmcode gespeichert. Bei jeder erneuten Ausführung des Programms wird in jedem Schritt überprüft, ob es ein n-Gramm gibt, daß zum aktuellen Zustand paßt. Ist dies nicht der Fall, hat man eine Anomalie gefunden, d.h. das Programm wurde möglicherweise modifiziert, evtl. durch einen Virenbefall.
Ein gültiges Programm ist dann die Menge aller möglichen Vereinigungen aller n-Gramme oder in formaler Schreibweise $CC_n^*(S) = \bigcup_{l \in L} CC_n^l(S)$, was die Crossover Closure der Menge der n-Gramme bezeichnet.

Crossover Closure

Die Crossover Closure, kurz *CC*, stellt, wie eben beschrieben, die Menge aller möglichen Merkmalskombinationen einer Auswahl bzw. Beispielmenge S dar. Sie ist eine weitere Möglichkeit zur Erkennung von Anomalien und zur Erzeugung neuer Detektoren. Die *CC* ist definiert als $CC(s) = \{u \in U | \forall$ Merkmale $w.s \in S.u[w] = s[w]\}$. Ein Merkmal muß dabei nicht genau einem Attribut entsprechen, sondern kann auch eine Kombination von Attributen darstellen.
Zur Verdeutlichung soll das nachfolgende Beispiel dienen.

Fahrzeuge

Räder	Farbe	Geschwindigkeit
4	rot	100
2	schwarz	200

Nach Crossover Closure

Räder	Farbe	Geschwindigkeit
4	rot	100
4	rot	200
4	schwarz	100
4	schwarz	200
2	rot	100
2	rot	200
2	schwarz	100
2	schwarz	200

Die Crossover Closure der Beispielmenge beinhaltet alle Fahrzeuge mit 2 oder 4 Rädern, in schwarz oder rot und mit jeder der möglichen Höchstgeschwindigkeiten.

In bezug auf Bitstrings bedeutet *CC(S),* daß es in der Beispielmenge *sample(t)* für jedes Element in *CC(S)* ein Element geben muß, das an derselben Fensterposition *w* identisch mit dem betrachteten Element aus *CC(S)* ist, wobei *w* ein Teilstring der Länge *r* ist.
Man spricht von einer abgeschlossenen Menge *S*, wenn gilt $CC(S) = S$.
Die Crossover Closure ist bezogen auf Anwendungen vergleichbar mit dem JOIN-Operator in relationalen Datenbanken. Ausgangspunkt für die *CC* ist das Crossing Over in der Genetik, das in abgewandelter Form auch in genetischen Algorithmen zu finden ist.
Bei der Anwendung der Crossover Closure zur Erzeugung von *RS* aus einer Beispielmenge ergibt sich das Problem, daß die *CC* nicht nur Elemente aus dem self erzeugt, sondern auch solche, die im Komplement liegen (vgl. Abb. 2). Wie man mit diesem Problem umgeht, wird später noch erläutert.

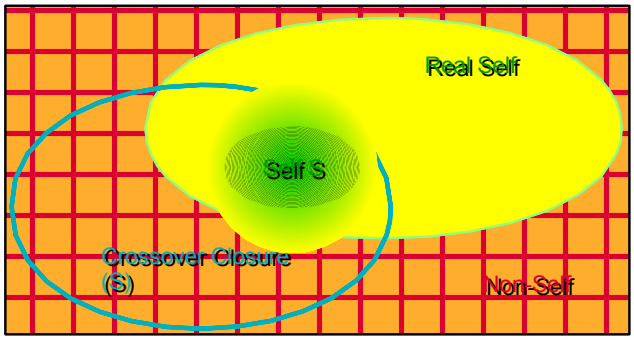


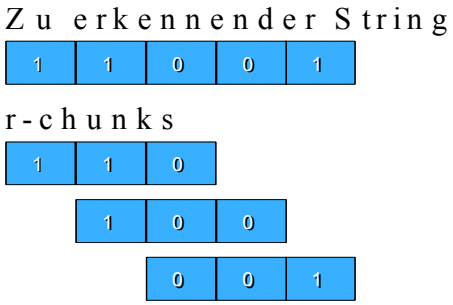
Abb. 2
Ausgehend von Beispielelementen aus dem self S erzeugt die Crossover Closure neue Elemente, die teilweise zum real self gehören, jedoch auch solche, die im non-self liegen.

r-contiguous-bit-Erkennung (rcb)

Als reiner Erkennungsalgorithmus konzipiert ist die r-contiguous-bit-Erkennung. Bei dieser wird ein String *x* dann erkannt, wenn mindestens *r* benachbarte Zeichen aus *x* mit den Zeichen an derselben Stelle im Detektor übereinstimmen, also $dMx \leftrightarrow \exists$ Fenster $w.d[w] = x[w]$.
Prinzipiell besteht eine starke Ähnlichkeit zum Hamming-Abstand, lediglich die Beschränkung auf benachbarte Zeichen ist beim Hamming-Abstand nicht gefordert.
Betrachtet man als Beispiel den rcb-Detektor *11001*, sowie Zeichenketten mit $l = 5$ und $r = 3$, so erkennt der Detektor u.a. die Strings *11011, 01001, 00001*, während *00011* und *10011* nicht erkannt werden. Würde man den Hamming-Abstand mit denselben Parametern als Detektor verwenden, würde dieser auch *10011* erkennen.

r-chunk-Erkennung

Eine Abwandlung der rcb-Methode sind r-chunks. Diese sind Teilstrings der zu erkennenden Zeichenketten, allerdings wird als Erkennungskriterium eine genaue Übereinstimmung mit dem zu erkennenden String gefordert. Ein r-chunk-Detektor *d* ist jeweils nur für ein Fenster *w* definiert, daher gilt $dMx \leftrightarrow x[w] = d$.
Zur Verdeutlichung sei das folgende Beispiel angegeben mit $l = 5$ und $r = 3$.



Erkennungsschemata

Nachdem im vorigen Abschnitt auf verschiedene Erkennungsmethoden eingegangen wurde, stellt sich im Folgenden die Frage, wie man die Methoden zur Identifizierung von Anomalien einsetzen kann, d.h. wie man self-Elemente von non-self-Elementen unterscheiden kann. Diese Frage führt zu 4 möglichen Kombinationsmöglichkeiten verschiedener Erkennungsschemata, wobei die Methoden in der Praxis zum Teil bessere Ergebnisse liefern als in theoretischen Untersuchungen vorausgesagt.

Positive Erkennung

Die positive Erkennung funktioniert derart, daß man bei ihr direkt Detektoren verwendet, die auf self-Elemente matchen. Sie scheint auf den ersten Blick die günstigste Methode zu sein, da es normalerweise wesentlich weniger Elemente im self gibt als außerhalb. Außerdem ist die positive Erkennung genauer, weil eine Übereinstimmung immer bedeutet, daß das Element im self liegt. Dies führt aber auch unmittelbar zum Nachteil der positiven Erkennung, da sich durch die genaue Übereinstimmung, speziell in Verbindung mit r-chunks oder rcb-Erkennung, auch ein höherer Berechnungsaufwand ergibt.

Negative Erkennung

Gegenüber der positiven Erkennung bedeutet ein Match bei der negativen Erkennung, daß das untersuchte Element nicht im self liegt und somit gerade eine Anomalie darstellt. Die durch diese Methode gefundenen Elemente sind, wie gerade verdeutlicht, i.A. wesentlich mehr als die Elemente des self. Da aber zur Identifizierung eines Elements als nicht im self liegend bei der negativen Erkennung bereits ein Match an einer beliebigen Stelle ausreicht, ist zum einen der Berechnungsaufwand geringer und zum zweiten läßt sich die Erkennung verteilt durchführen. Evtl. ist auch die Anwendung genetischer Algorithmen möglich.
Stillschweigend wurde hier die Tatsache verwendet, daß in einer abgeschlossenen Umgebung die positive und negative Erkennung denselben Informationsgehalt haben, wodurch beide Methoden in diesem Punkt gleichwertig sind.
Erzeugt man die Detektoren unabhängig voneinander, führt dies zu einer besseren Erkennung, da diese dann unterschiedliche Erkennungsfähigkeiten besitzen.

Disjunktive Erkennung

Äquivalent zur mathematischen Definition einer disjunktiven Formel bedeutet disjunktive Erkennung, daß bereits ein Match pro untersuchtem Objekt ausreicht, um das Objekt zu erkennen.

Konjunktive Erkennung

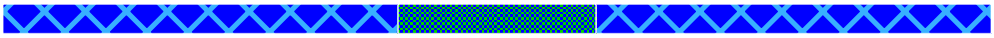
Die konjunktive Erkennung benötigt passende Pakete an allen Stellen des untersuchten Objekts.

Vier Kombinationen

Aus den vorgestellten Erkennungsmethoden ergeben sich vier Kombinationsmöglichkeiten, genannt Schemata, und zwar:

- $\text{Scheme}_{ND}(\Upsilon) = \{x \in U | \forall w. \nexists d \in \Upsilon. dMx\}$ mit negativ-disjunktiver Erkennung

- $\text{Scheme}_{PD}(\Upsilon) = \{x \in U | \exists w. \exists d \in \Upsilon. dMx\}$mit positiv-disjunktiver Erkennung



- $\text{Scheme}_{PC}(\Upsilon) = \{x \in U | \forall w. \exists d \in \Upsilon. dMx\}$ mit positiv-konjunktiver Erkennung und
- $\text{Scheme}_{NC}(\Upsilon) = \{x \in U | \exists w. \nexists d \in \Upsilon. dMx\}$mit negativ-konjunktiver Erkennung.

Υ bezeichnet dabei jeweils eine Menge von Detektoren.
(Meiner Meinung nach sind die Definitionen der negativen Erkennung, wie sie von Esponda, Forest und Helman verwendet werden, genau falsch herum angegeben, wenn man sich nach der Definition von konjunktiv und disjunktiv richtet!)

Vergleich der Erkennungsmethoden

Um die Stärken und Schwächen der vorgestellten Erkennungsschemata herauszufinden, werden die einzelnen Methoden miteinander verglichen.

Ähnlichkeiten

Nimmt man die Erkennungsmethode der r-chunks und vergleicht sie mit der Menge der Sprachklassen, die durch die Crossover Closure erkannt werden können, stellt man fest, daß Scheme_{PC} und Scheme_{ND} genau dieselben Sprachklassen wie die *CC* erkennen können. Die folgende mengentheoretische Betrachtung soll dies verdeutlichen. Es gilt nämlich

$$Scheme_{ND}(\Upsilon) = (Scheme_{PD}(\Upsilon))' = Scheme_{PC}(\Upsilon') = (Scheme_{NC}(\Upsilon'))'$$

$$\subset (Scheme_{ND}(\Upsilon'))' = Scheme_{PD}(\Upsilon') = (Scheme_{PC}(\Upsilon))' = Scheme_{NC}(\Upsilon)$$

Der Beweis findet sich bei Esponda, Forest und Helman.
Was den Zusammenhang zwischen r-chunk-Erkennung und rcb-Erkennung anbetrifft, so umfaßt die Erkennung mittels r-chunks die von rcb. Als Beispiel sei der rcb-Detektor *1011* mit Länge $l = 4$ gegeben, sowie die r-chunk-Detektoren d[1] = *10*, d[2] = *01* und d[3] = *11*, mit $r = 2$. Die erkannten Elemente sind *0010, 0011, 0111, 1000, 1001, 1010, 1011, 1111*.

Unterschiede und Probleme

Nimmt man die Mengengleichung und die These aus dem vorigen Abschnitt her, folgt daraus, daß die rcb-Erkennung nach Scheme_{PC} und Scheme_{ND} nicht alle Sprachklassen erkennen kann, die unter der Crossover Closure abgeschlossen sind, da die durch r-chunks auf jeden Fall alle Sprachklassen erkannt werden, die mittels rcb erkannt werden und die r-chunk-Erkennung äquivalent zur Crossover Closure ist. Wird die abgeschlossene Sprache *S* durch Scheme_{PD} erkannt, so wird ihr Komplement *S'* durch Scheme_{ND} erkannt.
Umgekehrt können durch die r-chunk-Erkennung nicht alle unter der Crossover Closure abgeschlossenen Sprachen erkannt werden, aber einige, die nicht abgeschlossen sind. Wird *S* von Scheme_{ND} erkannt, so erkennt Scheme_{PD} die Sprache *S'*. Nimmt man als Beispiel $l = 3$, $r = 2$ und die abgeschlossene Sprache $S = CC(S) = \{000\}$, dann enthält *S'* die Elemente *001, 010, 011, 100, 101, 110, 111* und $CC(S') = U$, was bedeutet, daß das Komplement eines Schemas, welches *CC(S')* erkennt, *S* nicht erkennen kann, wie oben behauptet.

Effizienz der Erkennungsmethoden

Effizienz bei positiver und negativer Erkennung

Um eine zuverlässige Erkennung durchzuführen, steigt bei positivem Erkennungsschema mit der Anzahl der Elemente im self auch die Anzahl der Detektoren. Nimmt man die

Informationsgleichheit von positiver und negativer Erkennung in einer abgeschlossenen Umgebung her, kommt man unmittelbar zu der Frage, ob und an welchem Punkt eine negative Erkennung Vorteile durch eine geringere Anzahl an Detektoren bringt. Dazu muß die Detektoranzahl abhängig von $|S|$ bestimmt werden.
Ausgangspunkt ist die Anzahl der Zeichenketten aus U mit Länge l, die ein bestimmtes Pattern enthalten, bei der Fenstergröße r. Dies sind in unserem binären Fall 2^{l-r}, da die Zeichen an den r Stellen ja bereits festgelegt sind und somit nur noch die übrigen $l-r$ Stellen variiert werden können.
Ein solchen String aus U wählt man zufällig mit der Wahrscheinlichkeit $\frac{2^{l-r}}{2^l} = \frac{1}{2^r}$.
Wählt man r klein im Vergleich zu l und bestimmt die Wahrscheinlichkeit, daß das gesuchte Pattern in einem zufälligen self-Menge S vorkommt, erhält man $1-(1-\frac{1}{2^r})^{|S|}$. $1-\frac{1}{2^r}$ ist die Wahrscheinlichkeit, daß das Pattern nicht in einem ausgewählten String ist, da im self $|S|$ Elemente vorhanden sind, ist wahrscheinlich in $(1-\frac{1}{2^r})^{|S|}$ der Fälle das Pattern nicht im self enthalten und daher enthalten die restlichen Fälle das gesuchte Pattern.
Für die Fenstergröße r ergeben sich dann $E_r \approx 2^r - 2^r(1-\frac{1}{2^r})^{|S|}$ unterschiedliche Pattern in S.
Bei positiver Erkennung benötigt man für diese Pattern $E_{pos} = tE_r$ Detektoren, während es bei negativer Erkennung $E_{neg} = t(2^r - E_r)$ Detektoren sind. Durch Substitution von E_r ergibt sich dann im Günstigervergleich $E_{pos} = E_{neg} = t(2^r(1-\frac{1}{2^r})^{|S|}) = t(2^r - 2^r(1-\frac{1}{2^r})^{|S|})$ und als Lösung für die Größe von S: $|S| = \frac{-\ln(2)}{\ln(1-\frac{1}{2^r})} \approx 0,693 \cdot 2^r$ für den binären Fall. Im allgemeinen Fall gilt $|S| \approx 0,693 \cdot \boldsymbol{A}^r$.

Reduktion der Detektoranzahl

Um die Detektoranzahl unabhängig von der Erkennungsmethode zu reduzieren, kann man zusätzlich Redundanzen ausnutzen, die sich durch die Erkennung vorausgehender Fenster ergeben. Dazu geht man am Beispiel der negativen Erkennung wie folgt vor:

- Für das erste Fenster t wird ein Detektor erstellt, wenn t nicht in S liegt.
- Für jedes weitere Fenster und jedes Paar von Pattern $w_1 = v_i...v_{r+1-2}a$, $w_2 = v_i...v_{r+1-2}\bar{a}$
  - Sind w_1 und w_2 in S, kann kein komplementärer Detektor erstellt werden.
  - Sind w_1 und w_2 nicht in S, ist kein Detektor nötig, da der Detektor im vorhergehenden Fenster schon beide Alternativen abdeckt.
  - Ist nur ein Pattern w_1 oder w_2 in S, muß der komplementäre Detektor erstellt werden, da kein solches Pattern in S existiert.

Die minimalen Anzahlen der auf diese Weise benötigten Detektoren sind dann $E_{\min N} = 2^r - E_r + (l-r)(E_r - 2(E_r - E_{r-1}))$ im bei negativer Erkennung und $E_{\min P} = E_r + (l-r)(E_r - 2(E_r - E_{r-1}))$ im positiven Fall.
Bestimmt man hier den Punkt, an dem die negative Erkennung vorteilhafter ist als die positive, kommt man für die Größe von S zu demselben Ergebnis wie im Falle der vollen

Detektoranzahl, wie folgende Grafik verdeutlicht.

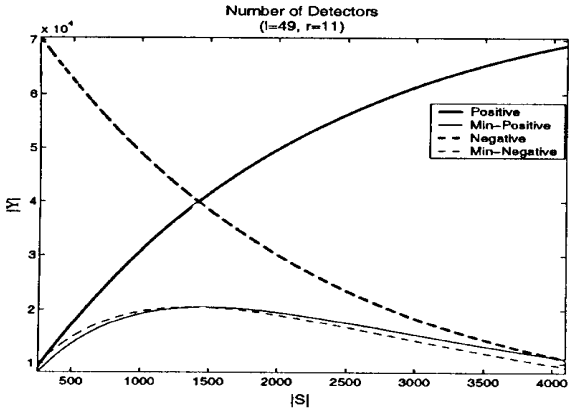


Fig. 2. Number of positive and negative detectors as a function of the size of the self set, assuming the self set was generated randomly. The plot shows both complete and reduced detector sets.

Löcher (holes) und Permutationen

Da für die Erkennung der Elemente als Detektoren lediglich Teilstrings verwendet werden, deren Kombination die erkannte Menge definiert, ergeben sich Probleme in den Bereichen, die vor allem durch die Crossover Closure der Detektoren nicht werden, aber in der zu erkennenden Menge liegen. Man spricht in dem Fall von Löchern. Insbesondere dann, wenn es sich bei den Löchern, sprich Erkennungslücken, um Anomalien handelt, ist dies problematisch, weil hier nicht mehr eindeutig zwischen self und non-self unterschieden werden kann. Es gibt jedoch eine Möglichkeit, das Problem zu verringern, wenn auch nur in günstigen Fällen.

Durch Permutation kann man in manchen Fällen doch noch Detektoren generieren, die die Löcher beseitigen. Als Beispiel sei $l = 3$, $r = 2$ und $S = \{101, 000\}$. Die Strings $h_1 = 100$ und $h_2 = 001$ sind nicht in S. Geht man von negativer Erkennung mit rcb aus, können h_1 und h_2 nicht erkannt werden, weil jeder der möglichen Detektoren *00*, 10*, *00, *01* auch Elemente aus S erkennen würde.

Durch die Permutation π ergibt sich $\pi(s_1) = 110$, $\pi(s_2) = 000$ und $\pi(h_1) = 010$, $\pi(h_2) = 100$. In diesem Fall lassen sich für h_1 und h_2 die Detektoren *011* und *101* definieren, die nicht mehr mit Elementen aus S kollidieren.

Wie die nachfolgende Tabelle zeigt, existieren aber auch genügend Fälle, in denen selbst durch eine Permutation keine Verbesserung mehr erreicht werden kann.

Self	010	001	100	100	001	010
	011	011	101	110	101	110
	100	100	010	001	010	001
	101	110	011	011	110	101
h	110	101	110	101	011	011
detector	11*	10*	11*	10*	01*	01*
templates	*10	*01	*10	*01	*11	*11

Diskussion

Ungeklärt ist bislang u.a. die Frage, wie sich die negative Erkennung in der Praxis behaupten wird. Durch die steigende Vernetzung wäre sie meiner Meinung nach eine Alternative zur herkömmlichen Anomalie-Erkennung.
Ob die Ähnlichkeit der Erkennungsschemata $Scheme_{ND}$ und $Scheme_{PC}$ in relationalen Datenbanken zur Speicherung verwendet werden kann, um hier das Datenaufkommen zu reduzieren, ist eine weitere noch offene Frage.
Die in der behandelten Untersuchung verwendeten Beispielmengen waren alle konstant; welche Möglichkeiten für die Aktualisierung dynamischer Beispielmengen lassen sich anwenden?
Unter Umständen gibt es neben der rein konjunktiven und rein disjunktiven Erkennung Zwischenlösungen, die dann auch wieder mit einer feineren Unterteilung der Ähnlichkeiten arbeiten als die einfache Unterscheidung zwischen *ähnlich zu sample(t)* und *unähnlich zu sample(t)*. Persönlich kann ich den Vorteil einer feineren Unterteilung allerdings nicht erkennen, lediglich eine dritte Kategorie *möglicherweise ähnlich zu sample(t)*, in die solche Elemente eingeordnet werden, die eine gewisse Ähnlichkeit mit der Beispielmenge haben und die daher per Hand noch einmal untersucht werden sollten, erscheint mir sinnvoll. Vor allem bei Mehrbenutzersystemen erscheint mir diese Möglichkeit sinnvoll, wenn das System einen neuen autorisierter Benutzer erkennen soll.
Ein weiteres wichtiges Problem ist die Frage nach effizienteren Methoden zur Erstellung und Speicherung von Detektoren, sowie zum Zugriff auf sie. Bislang muß zur Erstellung der Detektormenge die Beispielmenge vollständig untersucht werden. Speziell beim Zugriff auf ein große Anzahl von Detektoren könnten Verbesserungen möglich sein, u.a. durch Hashtables und Caching-Verfahren. Vielleicht lassen sich auch gerade Erkenntnisse aus Datenbankalgorithmen für die Speicherung der Detektoren verwenden.
Zum Schluß läßt sich noch untersuchen, in wieweit genetische Algorithmen und rcb miteinander kombiniert werden können.

Quellen

- F. Esponda, S. Forest, P. Helman. A formal Framework for Positive and Negative Detection Schemes. In IEEE Transactions on Systems, man and Cybernetics - Part B: Cybernetics, Vol. 34, No. 1, Feb. 2004
- F. Esponda, S. Forest, P. Helman. The Crossover Closure and Partial Match Detection.