情報処理学会論文誌　数理モデル化と応用　Vol.6　No.1　10–19 (Mar. 2013)

SQLインジェクション攻撃自動検出支援モデルと予測誤差

松田 健[1,a)]

受付日 2012年1月4日，再受付日 2012年3月13日 / 2012年4月26日，
採録日 2012年5月16日

概要：SQL インジェクション攻撃は Web アプリケーション攻撃の一種であり，データベース駆動型の Web アプリケーションにとって重大な脅威となっている．SQL インジェクション攻撃を自動検出する試みとして攻撃の文字列に対する構文解析やパターン認識，ブラックリストを用いた検出法が提案されているが，これらの検出法を回避する攻撃が開発されることによってリストが肥大化したり，検出にかかるコストが増加したりするため，攻撃に対処することが容易でなくなってきている．この問題に対し，本研究では SQL インジェクション攻撃の文字列に含まれる文字を攻撃特徴とする攻撃検出のためのモデルを提案した．さらに提案モデルの誤検出の度合いを測るための予測誤差を定義し，2 値判別の問題で広く利用されているシグモイド関数を用いたモデルを定義して提案モデルとの予測誤差を比較した．その結果，提案モデルではシグモイド関数を用いたモデルより予測誤差を小さくできることを示した．

キーワード：SQL インジェクション攻撃，攻撃検出モデル，シグモイド関数，予測誤差

Automatic Detection Model of SQL Injection Attacks and Prediction Error

Takeshi Matsuda[1,a)]

Received: January 4, 2012, Revised: March 13, 2012/April 26, 2012,
Accepted: May 16, 2012

Abstract: SQL injection attacks are serious problem to the security of database driven applications. The prevention methods against SQL injection attacks, such as parsing, pattern recognition and blacklist, have been developed, but these techniques may not cope with the evasion of SQL injection attacks detection. Moreover, it is a problem that a ballooning blacklist by updating may result in lowering of the detection performance. In this paper, we proposed a detection model of SQL injection attacks, and defined prediction error to measure the performance of the detection models. Then we compared our proposed model with the detection model applying sigmoid function, and showed that the prediction error of our proposed model is less than the sigmoid model.

Keywords: SQL injection attacks, attack detection model, sigmoid function, prediction error

1. はじめに

近年，個人情報を含む商用のデータベース駆動型 Web アプリケーションが広く導入されており，ネットショッピングやインターネットバンクなどオンライン上での取引がさかんに行われている．そのような Web アプリケーションに対して，SQL インジェクション攻撃と呼ばれる Web アプリケーションの脆弱性を悪用した攻撃が増加しており，問題となっている．SQL インジェクション攻撃は Web ページのフォームを介して行われる単純な攻撃であるが，攻撃が成功すると Web アプリケーションのデータベースへ不正にアクセスされ，データベースの情報が閲覧されたり改ざんされたりする恐れがある．

SQL インジェクション攻撃は 1999 年に電子雑誌 Phrack [1] でその危険性についてはじめて報告されており，2007 年頃からは自動拡散するワームやボットネットを

[1] サイバー大学 IT 総合学部
Faculty of Information Technology and Business, Cyber University, Minato, Tokyo 105–0011, Japan
[a)] takeshi_matsuda@cyber-u.ac.jp

利用した攻撃が観測されるようになっている [2]．SQL インジェクション攻撃から Web アプリケーションを防御する技術的手段としては，入力を制限する方法や攻撃を無害化するサニタイズと呼ばれる方法などが開発されている [3]．そのほかにも，過去に観測された SQL インジェクション攻撃に対してパターン認識や構文解析を応用した方法や，Web アプリケーションファイアウォール（WAF）と呼ばれる攻撃パターンをブラックリスト化して入力された文字列とリストの照合を行う手法を用いた攻撃検出法が開発・実用化されている [4], [5], [6], [7], [8], [9], [10], [11], [12]．しかしながら，これらの方法では未知の攻撃やその亜種が新たに開発されるたびに攻撃検出ができなくなるため，新しい攻撃が観測されるたびにブラックリストを更新しなければならない．さらに，肥大化したリストで検出を行うと検出にかかる時間が増加するなどの問題が生じてしまい，攻撃検出に対応することが難しくなっている．

この問題に対し，我々は SQL インジェクション攻撃に含まれるセミコロンや括弧などの記号を中心とする文字を利用した攻撃検出アルゴリズムを提案している [13]．本研究では，その方法を応用して SQL インジェクション攻撃を検出するためのモデル [14] を提案する．SQL インジェクション攻撃の検出は，フォームに入力された文字列が攻撃であるかどうかの 2 値判別を行う問題として扱うことができる．2 値判別の問題ではシグモイド関数を応用した数理モデルが広く利用されており [15], [16], [17]，SQL インジェクション攻撃の検出に応用することも可能である．そこで本研究では，シグモイド関数を応用して SQL インジェクション攻撃を検出するモデルと，モデルの誤検出の度合いを評価する予測誤差とを定義し，提案モデルではシグモイド関数を応用したモデルより予測誤差を小さくできることを数学的に示した．本研究で提案するモデルは，フォームに入力された文字列に含まれるいくつかの特定の文字や記号の含有率を計算したうえで，入力された文字列が SQL インジェクション攻撃であるかどうかの判別を行う．リスト方式で攻撃を検出する WAF では，新しい攻撃が開発されるたびにパターン認識や正規表現の組合せを考えて新たな検出ルールを作成する必要がある．しかしながら本論文の提案手法では，新しい攻撃が開発され SQL インジェクション攻撃のパターンが変わった場合でも，攻撃検出に最適な文字を選び直したり，攻撃検出に利用する文字を追加したりすることによって検出ルールを自動的に更新できることに特徴があり，この点が WAF と提案手法との相違点となっている．さらに予測誤差の値を小さくできること，つまり SQL インジェクション攻撃に対して攻撃を正しく検出する検出率を高くするだけでなく，攻撃でない文字列を攻撃として誤って検出してしまう誤検出率の割合を低くすることができることも提案モデルの特徴の 1 つである．本論文の構成は以下のとおりである．2 章では SQL インジェクション攻撃の実例を用いて文献 [13] で提案した攻撃検出アルゴリズムの紹介を行う．3 章では SQL インジェクション攻撃検出のための提案モデルとシグモイド関数を応用したモデルを定義し，4 章で主定理の証明を行う．5 章で提案法に関する考察を行い，最後の 6 章でまとめをして，今後の課題を示す．

2. SQL インジェクション攻撃とその検出アルゴリズム

この章では SQL インジェクション攻撃の概要と，文献 [13] で提案した SQL インジェクション攻撃検出のアルゴリズムを紹介する．

2.1 SQL インジェクション攻撃

本論文では SQL 文に利用される文字に着目して攻撃を検出する方法を提案する．これ以降，Web ページのフォームに入力された文字列（以下，入力文字列という）が SQL インジェクション攻撃である場合それを攻撃文字列と呼び，SQL インジェクション攻撃でない場合は正常文字列と呼ぶことにする．SQL インジェクション攻撃は Web ページのフォームを介して行われる攻撃である．脆弱性を持つデータベース駆動型の Web アプリケーションに悪意のある SQL クエリが入力された場合，データベースへの不正なアクセスを認めてしまう恐れがあり問題となっている．本論文では，SQL インジェクション攻撃に含まれる文字，特にシングルクォーテーション，セミコロン，カンマなどの記号を攻撃特徴として攻撃検出を行うモデルの提案を行う．そのために，攻撃文字列に頻出する記号とそれらの SQL インジェクション攻撃における役割について攻撃の例を用いながら説明する．

例 1　ユーザ ID とパスワードを入力するためのフォームが用意されている Web ページにおいて，ユーザ ID の入力フォームに

　name

パスワードの入力フォームに

　777’ OR ’7=7

という文字列が入力されたとき，

　SELECT * FROM CardInfo WHERE ID=’name’ and
　PASS=’777’ OR ’7=7’

という SQL クエリが生成される Web アプリケーションがあるとする．このとき，この入力文字列の最後の部分の恒等式 $7 = 7$ はつねに真となる命題であるため，入力時に無害化や入力制限などの処理が適切にされない場合，データベースは攻撃者に対してすべての情報へのアクセスを許してしまう．

例 2　例 1 と同様に，ユーザ ID とパスワードを入力するフォームが用意されている Web ページを想定する．いま，ユーザ ID の入力フォームに

name

パスワードの入力フォームに

7;drop table cardinfo

という文字列が入力されたとする．このとき，Web アプリケーションは以下のような SQL クエリを生成する．

SELECT * FROM CardInfo WHERE ID='name' and pass=7; DROP TABLE CardInfo

この攻撃では区切り文字のセミコロン（;）が使われているところに特徴があり，区切り文字の後に悪意のある SQL クエリが余分に付け加えられている．区切り文字セミコロン（;）に対して脆弱性を持つ Web アプリケーションは，セミコロン（;）が入力されていることから，元々の SQL クエリだけでなく付け加えられた悪意のある SQL クエリも通してしまう．その結果，この攻撃が成功すると，データベースから CardInfo という名前のテーブルが消去される．

例 1，2 のタイプの SQL インジェクション攻撃では，スペースやシングルクォーテーション，セミコロンといった記号が攻撃文字列の中に多く含まれていることが分かる．セミコロンは例 2 において区切り文字として利用されているが，例 1 に頻出しているシングルクォーテーションとともに，文字列をプログラム内での文字型変数に定数として格納する際にも用いられる．また，スペースは区切り文字の役割として通常の SQL クエリにも頻繁に出現するが，SQL インジェクション攻撃を目的とせずに Web ページのユーザ ID やパスワードのフォームに入力する場合，基本的に名前や番号などの入力が中心となるためそれほど多くのスペースが入力されることは考えにくい．したがって，スペース，シングルクォーテーション，セミコロンといった記号は SQL インジェクション攻撃を表す特徴として利用できるものと考えられる．例 1，2 は Web アプリケーションに対して直接攻撃するタイプの SQL インジェクション攻撃であるが，そのほかにも 7 = 1 のように，わざとエラー表示をさせる文字列を入力して Web アプリケーションの脆弱性を探り，その後で攻撃を行うパターンのものも存在する．このような間接的に行われる攻撃のパターンにも例 1，2 の攻撃と同様にスペース，シングルクォーテーション，セミコロンといった記号が含まれることが多い．そのため，本論文では正常文字列の入力以外を攻撃文字列と判別する SQL インジェクション攻撃検出モデルについて考える．提案モデルでは，文字列に含まれる記号を攻撃特徴として攻撃検出を行うが，攻撃特徴となる文字の選定方法については次の節で述べる．

2.2 文字による SQL インジェクション攻撃検出アルゴリズム

本節では，攻撃文字列に含まれるいくつかの文字を攻撃特徴として抽出し，それらの文字を用いた SQL インジェクション攻撃の検出アルゴリズム [13] を紹介する．

表 1　攻撃特徴文字の候補

Table 1　Candidate of attack feature symbols.

変数	文字候補
s_{i}	半角スペース
s_{ii}	;（セミコロン）
s_{iii}	'（シングルクォーテーション）
s_{iv}	(（左側丸括弧）
s_{v}	)（右側丸括弧）
s_{vi}	{（左側中括弧）
s_{vii}	}（右側中括弧）
s_{viii}	[（左側大括弧）
s_{ix}	]（右側大括弧）
s_{x}	#（シャープ）
s_{xi}	%（パーセント）
s_{xii}	"（ダブルクォーテーション）
s_{xiii}	&（アンパサンド）
s_{xiv}	\（バックスラッシュ）
s_{xv}	|（パイプ）
s_{xvi}	=（等号）
s_{xvii}	>（大なり記号）
s_{xviii}	<（小なり記号）
s_{xix}	*（アスタリスク）
s_{xx}	/（スラッシュ）

フォームから入力される入力文字列を l，記号・数字・アルファベットなどの文字を s_a（$a = \mathrm{i}, \mathrm{ii}, \mathrm{iii}, \cdots$）とする．文献 [13] では，**表 1** に示す 20 個の文字（$s_{\mathrm{i}}, s_{\mathrm{ii}}, \cdots, s_{\mathrm{xx}}$）を用意して SQL インジェクション攻撃の特徴をとらえる文字の選定を行った．用意した 20 個の文字は SQL 文によく利用される記号を中心として構成した．

SQL インジェクション攻撃の攻撃特徴となる文字の選定方法は以下のとおりである．入力文字列 l に対して，

$$x_{l,a} = \frac{\#s_a}{|l|}$$

を定義し，0 から 1 の有理数に値をとる $x_{l,a}$ を入力として扱う．ここで，$|l|$ は入力文字列長を，$\#s_a$ は入力文字列に含まれる文字 s_a の総数を表すものとする．SQL インジェクション攻撃の検出では，入力文字列が攻撃文字列であるか正常文字列であるかを判別する問題を考える．そのために次のような判別関数を定義する．

$$h(x_{l,a}) = \begin{cases} 1 & (x_{l,a} \geq \alpha) \\ 0 & (x_{l,a} < \alpha) \end{cases}$$

ここで，α は 0 から 1 の値をとる実数で，入力文字列 l が攻撃文字列であるか正常文字列であるかを判別するための閾値である．本論文では $h(x_{l,a}) = 1$ なら l を攻撃文字列と，$h(x_{l,a}) = 0$ なら l を正常文字列と判別することにする．以下に，例 1 で紹介した例に対する判別関数の計算例を示す．

例 3　入力文字列 l_1 を

SELECT * FROM CardInfo WHERE ID='name' and PASS='777' OR '7=7'

とする．この攻撃に対して，表にある文字の s_{i}（半角スペース），s_{ii}（セミコロン），s_{iii}（シングルクォーテーション）を用いて判別関数の値を計算する．

$$|l_1| = 62$$
$$\#s_{\mathrm{i}} = 9$$
$$\#s_{\mathrm{ii}} = 0$$
$$\#s_{\mathrm{iii}} = 4$$

であることから，

$$x_{l_1,\mathrm{i}} = \frac{9}{62} = 0.15$$
$$x_{l_1,\mathrm{ii}} = \frac{0}{62} = 0$$
$$x_{l_1,\mathrm{iii}} = \frac{4}{62} = 0.06$$

となる．計算結果は小数点第 3 位で四捨五入を行ったものであり，以下同様の計算を行うものとする．この場合，閾値を $\alpha = 0.06$ ととれば s_{i}（半角スペース）または s_{iii}（シングルクォーテーション）であれば攻撃の検出が可能であるが，s_{ii}（セミコロン）を用いた場合は攻撃検出をすることができない．また，次のような電話番号を想定した入力文字列 l_2

01 2345 6789

が入力された場合，s_{i}（半角スペース）を用いて攻撃検出を行うと，

$$x_{l_2,\mathrm{i}} = \frac{2}{12} = 0.17$$

となり，閾値を $\alpha = 0.06$ とした場合，判別関数は l_2 を攻撃文字列であると誤判別することになる．

表 2 は，文献 [18], [19], [20], [21] などから収集した 624 個の攻撃文字列サンプルとフォームに入力される文字を想定して作成した 234 個の正常文字列サンプルを用いて判別関数の計算（$\alpha = 0.1$ とした）を行い，攻撃文字列を攻撃であると判別した割合（以下，攻撃検出率 P_A という）と正常文字列を攻撃であると判別した割合（以下，正常誤検出率 P_N という）を計算してまとめたものである．

表 2 の結果から，攻撃特徴となる文字を 1 文字だけ用いて攻撃検出を行ったときの攻撃検出率は，半角スペース以外を用いた場合それほど高くならないことが分かる．また半角スペースの攻撃検出率は非常に高い一方，正常誤検出率についても他の文字と比較すると高くなっている．そこで入力文字列 l に対する $x_{l,a}$ の計算式を次のように変更して攻撃検出を行う．

$$x_{l,k} = \frac{\#S_k}{|l|}$$

ここで S_k は攻撃特徴となる複数の文字からなる集合であり，$\#S_k$ を入力文字列 l 中の S_k に属する文字の総数と定義する．いま，$S_1 = \{s_{\mathrm{i}}, s_{\mathrm{ii}}, s_{\mathrm{iii}}\}$（半角スペース，セミコロン，シングルクォーテーション）と定義すると例 3 の入力文字列 l_1 に対して，

$$x_{l_1,1} = \frac{9+0+4}{62} = \frac{13}{62} = 0.21$$

となる．この場合，閾値を $0 \leq \alpha < 0.21$ とすれば入力 l_1 を攻撃であると判別することができる．さらに例 3 の電話番号を想定した入力文字列 l_2 に対しても，

$$x_{l_2,1} = \frac{2}{12} = 0.17 < 0.21$$

となることから，l_2 は攻撃でないと判別されることが分かる．以上の考察により，表 2 にある攻撃検出率 P_A が高い文字を組み合わせて攻撃を検出することで攻撃検出率 P_A が高く，かつ正常誤検出率 P_N を低くするような攻撃検出ができるものと期待される．文献 [13] では，文字集合として $S = \{s_{\mathrm{i}}, s_{\mathrm{ii}}, s_{\mathrm{iv}}\}$（半角スペース，セミコロン，右側丸括弧）を用いて攻撃検出した場合，攻撃検出率 P_A と正常検出率 $1 - P_N$ の加重平均 μ に対して，閾値が $\alpha = 0.08$ のとき

$$\mu = \beta P_A + (1-\beta)(1-P_N) > 0.9$$

が $b = 0, 0.1, 0.2, \cdots, 1.0$ で成り立つことを実験によって確認した．なお，実験に使用したサンプルは文献 [18], [19], [20], [21] から収集したものである．

表 2　攻撃検出率 P_A と正常誤検出率 P_N

Table 2　Attack detection rate P_A and normal detection rate P_N.

攻撃特徴文字候補	P_A	P_N
s_{i}（スペース）	0.971	0.098
s_{ii}（セミコロン）	0.664	0.000
s_{iii}（シングルクォーテーション）	0.483	0.000
s_{iv}（左側丸括弧）	0.459	0.060
s_{v}（右側丸括弧）	0.414	0.090
s_{vi}（左側中括弧）	0.000	0.060
s_{vii}（右側中括弧）	0.000	0.026
s_{viii}（左側大括弧）	0.000	0.026
s_{ix}（右側代括弧）	0.000	0.026
s_{x}（シャープ）	0.032	0.021
s_{xi}（パーセント）	0.024	0.026
s_{xii}（ダブルクォーテーション）	0.010	0.000
s_{xiii}（アンパサンド）	0.019	0.021
s_{xiv}（バックスラッシュ）	0.032	0.000
s_{xv}（パイプ）	0.040	0.103
s_{xvi}（等号）	0.294	0.060
s_{xvii}（大なり記号）	0.000	0.090
s_{xviii}（小なり記号）	0.000	0.038
s_{xix}（アスタリスク）	0.128	0.094
s_{xx}（スラッシュ）	0.112	0.073

3. 提案モデル

本章では，SQL インジェクション攻撃を検出するための

モデルを提案する．さらに 2 値判別の問題に広く利用されているシグモイド関数を用いた攻撃検出モデルの定義を行う．前章で紹介したとおり我々の提案アルゴリズム [13] の特徴は，攻撃検出に用いる攻撃特徴文字をうまく選ぶことで，小さな閾値 α によって高い確率の攻撃検出の可能性を持つことである．実際，文献 [13] の実験では，入力文字列 l に対する $x_{l,k}$（以下，記号が煩雑になることを避けるため x とする）の値が 0.1 程度の小さな値で l が攻撃文字列である可能性が高くなり，x の値が 0 に近ければ近いほど l が正常文字列である可能性が高くなるという結果を得ている．これは，正常文字列には攻撃文字列に頻出する記号がほとんど出現しないという性質によるものであるが，この正常文字列に関する性質については 5 章の考察で紹介を行う．以上のことをふまえ，本論文ではベルヌーイ分布を基にした SQL インジェクション攻撃を自動検出するためのモデル

$$p_1(y|x,\theta)=\sum_{i=1}^{I} a_i (x^{b_i})^y (1-x^{b_i})^{1-y}$$

を定義する[*1]．ただし，$i=1,2,\cdots,I$ は自然数であり，$\theta=\{(a_i,b_i)\}_{i=1}^{I}$, すべての i に対して $a_i\geq 0$ かつ $\sum_{i=1}^{I} a_i=1$ である．次に，シグモイド関数を用いて攻撃検出を行うモデルを次のように定義する．シグモイド関数は

$$s(t)=\frac{1}{1+e^{-bt}}$$

で定義される $(-\infty,\infty)$ から $[0,1]$ への実連続関数である．ここで b は任意の実数である．本論文で提案する SQL インジェクション攻撃を検出するためのモデル p_1 は混合ベルヌーイ分布を基にして構成したものである．混合ベルヌーイ分布にシグモイド関数を適用してできるモデルは次のように定義することができる．

$$q(y|x,\theta)=\sum_{i=1}^{I}\Big(\frac{1}{1+e^{-b_i x}}\Big)^y\Big(1-\frac{1}{1+e^{-b_i x}}\Big)^{1-y}\ {}^{*2}.$$

前章で定義したとおり，本研究では入力文字列 l に対して 0 以上 1 以下の有理数 x を定義し，x の値から判別関数 $h(x)$ の値を求めることで攻撃検出を行う．シグモイド関数の定義域は実数全体であるため，これを以下の関数

$$t=\begin{cases} -\infty & \Big(x=0\Big)\\ \log\frac{x}{1-x} & \Big(x\in(0,1)\Big)\\ \infty & \Big(x=1\Big)\end{cases}$$

を用い，この関数と関数 $s(t)$ を合成してできる関数

$$g(x)=\frac{x^b}{x^b+(1-x)^b}$$

をシグモイド関数として再定義する．関数 $g(x)$ の定義域と値域はそれぞれ $[0,1]$ である．本論文ではこの関数 $g(x)$ を用いて SQL インジェクション攻撃を検出するモデルを

$$p_2(y|x,\theta)=\sum_{i=1}^{I} a_i\Big(\frac{x^{b_i}}{x^{b_i}+(1-x)^{b_i}}\Big)^y\Big(1-\frac{x^{b_i}}{x^{b_i}+(1-x)^{b_i}}\Big)^{1-y}$$

と定義し，$q(y|x,\theta)$ の代わりに $p_2(y|x,\theta)$ をシグモイド関数を応用したモデルとして取り扱うことにする．我々の目標は前章で述べたとおり攻撃検出率を高く，正常誤検出率を低くすることである．そこで，

$$e(p)=\int_{\{x|h(x)=1\}} p(x,y=0)dx+\int_{\{x|h(x)=0\}} p(x,y=1)dx$$

を考え，$e(p)$ をモデル $p(y|x,\theta)$ の予測誤差として定義する．また $p(x)$ を以下の性質を満足する関数と定義する．$\{x_0,x_1,x_2,\cdots,x_n\}$ を実数の閉区間 $[0,\frac{1}{2}]$ の $x_0\leq x_1\leq\cdots\leq x_n$ を満足する実数列とする．$x\in[0,\frac{1}{2}]$ に対して，

$$p(x)=\sum_{j=0}^{n-1}\tau_j \mathbf{1}(x)_{(x_j,x_{j+1})}$$

$$p(1-x)=\sum_{j=0}^{n-1}\tau'_j \mathbf{1}(x)_{(1-x_{j+1},1-x_j)}$$

かつ

$$p(x)\leq p(1-x) \tag{1}$$

が成り立つと仮定する．ただし，$j=0,1,\cdots,n-1$ において τ_j，τ'_j はある実数であり，

$$\mathbf{1}(x)_{(x_j,x_{j+1})}=\begin{cases}1, & x\in(x_j,x_{j+1})\\ 0, & x\text{:それ以外}\end{cases}$$

である．このとき，本論文で提案するモデル $p_1(y|x,\theta)$ とシグモイド関数を応用したモデル $p_2(y|x,\theta)$ との間に次の関係が成り立つ．

定理 1　次の 2 つの条件

(1) $0<b<1$

(2) $0\leq\alpha\leq\frac{1}{2}$

が成り立つとき，

$$e(p_2)>e(p_1)$$

である．

[*1] 提案モデル $p_1(y|x,\theta)$ は混合ベルヌーイモデルに基づくモデルであり，確率 a_i で観測された入力が攻撃である確率が x_i^b であることを意味する．なお，Bishop [22] の p.204，p.205 にある式 (4.87)，(4.89) の形と提案モデル p_1 の形はよく似ているが，式 (4.89) はベルヌーイ分布 $y^t(1-y)^{1-t}$ の尤度関数である．本論文では式 (4.87) として関数 x^{b_i} を提案しているが，提案モデル p_1 はベルヌーイ分布の尤度関数とは別の関数である．提案モデルの尤度関数はサンプルを $D=\{(x_1,y_1),(x_2,y_2),\cdots,(x_n,y_n)\}$ とおけば，$P_1(\theta|D)=\prod_{j=1}^{n}\{\sum_{i=1}^{I} a_i (x_j^{b_i})^{y_j}(1-x_j^{b_i})^{1-y_j}\}$ となる．

[*2] $I=1$ の場合においては文献 [23] でも同様のモデルが紹介されている．

4. 証明

この章では，4.1 節で定理の証明に必要な補題について整理を行い，4.2 節で前章で紹介した定理 1 の証明を行う

4.1 補題

補題 1　すべての $i = 1, 2, \cdots, I$ に対して，$0 \leq b_i \leq 1$ であるとき，

$$f_i(x) = x^{b_i} - \Big(\frac{x^{b_i}}{x^{b_i} + (1-x)^{b_i}}\Big)$$

とすると，$f_i(x) \geq 0$ である．

［補題 1 の証明］i を固定して考える．

$$\begin{aligned} f_i(x) &= x^{b_i}\Big(1 - \frac{1}{x^{b_i} + (1-x)^{b_i}}\Big) \\ &= \frac{x^{b_i}}{x^{b_i} + (1-x)^{b_i}}\Big(x^{b_i} + (1-x)^{b_i} - 1\Big) \end{aligned}$$

において $0 \leq x \leq 1$ で

$$0 \leq \frac{x^{b_i}}{x^{b_i} + (1-x)^{b_i}} \leq 1$$

であるから，$h_i(x) = x^{b_i} + (1-x)^{b_i} - 1$ とおいて $h_i(x) > 0$ を示す．x について微分すると

$$\frac{dh_i}{dx}(x) = b_i x^{b_i - 1} - b_i (1-x)^{b_i - 1}$$

であり，

$0 \leq x \leq \frac{1}{2}$ なら $x^{b_i - 1} < (1-x)^{b_i - 1}$

$x = \frac{1}{2}$ なら $x^{b_i - 1} = (1-x)^{b_i - 1}$

$\frac{1}{2} \leq x \leq 1$ なら $x^{b_i - 1} > (1-x)^{b_i - 1}$

となるから，$0 \leq b_i \leq 1$ に対して $h_i(\frac{1}{2}) = 2(\frac{1}{2})^{b_i} - 1 \geq 0$ が成り立つ．ただし，等号が成立するのは $b_i = 1$ のときである．したがって，すべての $i = 1, 2, \cdots, I$ に対して $f_i(x) \geq 0$ である．（証明終わり）

補題 2　$0 \leq \epsilon \leq \frac{1}{2}$ を満足する実数 ϵ に対して，

$$f_i(1-\epsilon) \geq f_i(\epsilon)$$

が成り立つ．

［補題 2 の証明］

$$f_i(\epsilon) = \epsilon^{b_i}\Big(1 - \frac{1}{\epsilon^{b_i} + (1-\epsilon)^{b_i}}\Big)$$

$$f_i(1-\epsilon) = (1-\epsilon)^{b_i}\Big(1 - \frac{1}{\epsilon^{b_i} + (1-\epsilon)^{b_i}}\Big)$$

より，

$$\begin{aligned} &f_i(1-\epsilon) - f_i(\epsilon) \\ &\quad = \Big\{(1-\epsilon)^{b_i} - \epsilon^{b_i}\Big\}\Big(1 - \frac{1}{\epsilon^{b_i} + (1-\epsilon)^{b_i}}\Big) \end{aligned}$$

である．$0 \leq \epsilon \leq \frac{1}{2}$ より $(1-\epsilon)^{b_i} - \epsilon^{b_i} \geq 0$，また補題 1 より $\epsilon^{b_i} + (1-\epsilon)^{b_i} - 1 \geq 0$ であるから，

$$f_i(1-\epsilon) - f_i(\epsilon) \geq 0$$

が成り立つ．等号は $\epsilon = \frac{1}{2}$ のときに成立する．（証明終わり）

4.2 定理 1 の証明

すべての $i = 1, 2, \cdots, I$ に対して，$0 \leq b_i \leq 1$ である b_i を固定して考える．

$$\begin{aligned} e(p_2) - e(p_1) = &\Big\{\int_{\{x|h(x)=1\}} p_2(y=0|x)p(x)dx \\ &+ \int_{\{x|h(x)=0\}} p_2(y=1|x)p(x)dx\Big\} \\ &-\Big\{\int_{\{x|h(x)=1\}} p_1(y=0|x)p(x)dx \\ &+ \int_{\{x|h(x)=0\}} p_1(y=1|x)p(x)dx\Big\} \end{aligned}$$

であり，$i = 1, 2, \cdots, I$ に対して

$$f_i(x) = x^{b_i} - \frac{x^{b_i}}{x^{b_i} + (1-x)^{b_i}}$$

とおくと

$$\begin{aligned} e(p_2) - e(p_1) = \sum_{i=1}^{I} a_i\Big(&\int_{\{x|h(x)=1\}} f_i(x)p(x)dx \\ &- \int_{\{x|h(x)=0\}} f_i(x)p(x)dx\Big) \end{aligned}$$

と書くことができる．したがって，$e(p_2) - e(p_1) > 0$ を示すには，

$$\int_{\{x|h(x)=1\}} f_i(x)p(x)dx > \int_{\{x|h(x)=0\}} f_i(x)p(x)dx$$

を示せばよいことが分かる．もし $\alpha = \frac{1}{2}$ のとき，

$$\int_{\{x|h(x)=0\}} f_i(x)p(x)dx < \int_{\{x|h(x)=1\}} f_i(x)p(x)dx$$

が成り立つならば，

$$\int_{\{x|h(x)=1\}} f_i(x)p(x)dx = \int_{\frac{1}{2}}^{1} f_i(x)p(x)dx$$

$$\int_{\{x|h(x)=0\}} f_i(x)p(x)dx = \int_{0}^{\frac{1}{2}} f_i(x)p(x)dx$$

より，$0 \leq \alpha^* < \frac{1}{2}$ である実数 α^* に対して

$$\begin{aligned} \int_0^{\alpha^*} f_i(x)p(x)dx &< \int_0^{\frac{1}{2}} f_i(x)p(x)dx \\ &< \int_{\frac{1}{2}}^{1} f_i(x)p(x)dx \\ &< \int_{\alpha^*}^{1} f_i(x)p(x)dx \end{aligned}$$

であるから $0 \leq \alpha \leq \frac{1}{2}$ で

$$\int_{\{x|h(x)=0\}} f_i(x)p(x)dx < \int_{\{x|h(x)=1\}} f_i(x)p(x)dx$$

が成り立つことが分かる．したがって，$\alpha = \frac{1}{2}$ の場合のみ示せば $e(p_2) > e(p_1)$ を示すことができる．$\{x_0, x_1, \cdots, x_n\}$ を $x_0 \leq x_1 \leq \cdots \leq x_n$ を満たす閉区間 $[0, \frac{1}{2}]$ の実数列において，

$$\triangle_j = x_{j+1} - x_j$$
$$\triangle'_j = (1 - x_j) - (1 - x_{j+1})$$

と定義する．このとき，$\triangle_j = \triangle'_j$ である．$f_i(x)$ は閉区間 $[0, 1]$ 上の連続関数であり，

$$\int_0^{\frac{1}{2}} f_i(x)p(x)dx = \lim_{n\to\infty} \sum_{j=0}^{n-1} f_i(\xi_j)p(\xi_j)\triangle_j,$$
$$\int_{\frac{1}{2}}^1 f_i(x)p(x)dx = \lim_{n\to\infty} \sum_{j=0}^{n-1} f_i(\xi'_j)p(\xi'_j)\triangle'_j,$$

と書ける．ただし，$\xi_j \in [x_j, x_{j+1}]$，$\xi'_j = 1 - \xi_j$ である．さらに補題 2 と式 (1) から，すべての $j = 0, 1, \cdots, n-1$ に対して

$$f_i(\xi'_j)p(\xi'_j)\triangle'_j > f_i(\xi_j)p(\xi_j)\triangle_j$$

が成り立つ．したがって，$\alpha = \frac{1}{2}$ のとき

$$\int_{\{x|h(x)=0\}} f_i(x)p(x)dx < \int_{\{x|h(x)=1\}} f_i(x)p(x)dx$$

であり，$0 \le \alpha \le \frac{1}{2}$ のとき $e(p_2) - e(p_1) > 0$ である．（証明終わり）

5. 考察

本論文では，SQL インジェクション攻撃を検出するためのモデル

$$p_1(y|x, \theta) = \sum_{i=1}^{I} a_i (x^{b_i})^y (1 - x^{b_i})^{1-y}$$

を提案した．SQL インジェクション攻撃では，攻撃文字列と正常文字列とでかなり異なる性質を持っている．本研究では SQL インジェクション攻撃に含まれる文字，特にセミコロンやシングルクォーテーションなどの記号を用いて攻撃検出を行う方法を考察しているが，これらの記号は正常文字列に含まれることはまれである．そのために提案モデル p_1 も単純なものになっている．本研究で使用した正常文字列のサンプルは，Web 上のフォームに入力される通常の入力を文字列として

- ID やパスワード，電話番号，日本語も想定した氏名や住所
- ブログで頻繁に使用される Wiki の文法
- Web ページで使用される顔文字

などに注意してサンプルの生成を行った．提案手法では，検出率は攻撃検出に利用する文字集合 S と検出閾値 α を変えることで変化する．文献 [13] では攻撃検出率 P_A と正常検出率 $1 - P_N$ の加重平均 μ を大きくする文字集合として {半角スペース，セミコロン，右側丸括弧} が選ばれたが，シングルクォーテーションが頻出する SQL インジェクション攻撃も多く存在する．そこで，次の 2 つの文字集合を用意し，それぞれの場合での提案方法の検出能力について考察を行う．

- $S_1 =$ {半角スペース，セミコロン，右側丸括弧}
- $S_2 =$ {半角スペース，シングルクォーテーション}

ただし，入力文字列 l に含まれる S_k の含有率 $x_{l,k} = \frac{\#S_k}{|l|}$ が 0.10 より大きいとき，入力文字列 l を攻撃と判別することにする．次のようなシングルクォートを含まない SQL クエリ

l_1 =SELECT accountsNumber, customersName FROM CardInfomationTable WHERE ID=1 and PASS=777

に対して，次のように攻撃を埋め込める場合を考える．

l_2 =SELECT accountsNumber, customersName FROM CardInfomationTable WHERE ID=1 and PASS=777' OR '7=7'

この場合，$x_{2,1} = \frac{12}{97} = 0.12 > 0.10$，$x_{2,2} = \frac{15}{97} = 0.15 > 0.10$（小数第 3 位で四捨五入）となり，どちらの文字集合を利用しても攻撃検出可能である．参考までに $x_{1,1} = \frac{8}{85} = 0.09 < 0.10$，$x_{1,2} = \frac{8}{85} = 0.09 < 0.10$ であることから，l_1 は正常文字列と判別されることも分かる．しかし，攻撃ではないが次のようなシングルクォーテーションを多く含む文字列

l_3 =SELECT accountsNumber, customersName FROM CardInfomationTable WHERE ID='\'\'\'\'\'\'\'\'\'\'"

といった文字列が入力された場合は，$x_{3,1} = \frac{6}{91} = 0.07 < 0.10$，$x_{3,2} = \frac{17}{91} = 0.19 > 0.10$ となり，入力文字列 l_3 は文字集合 S_1 を用いた場合では正常文字列と判別されるが，S_2 を用いた場合では攻撃文字列と判別される．入力文字列 l_3 にはシングルクォーテーションが 11 個含まれているが，仮に '\'\'\'\'\'\'\'\'\'\'" の部分を '\" のように減らしたとしてできる入力文字列を l_4 としても，$x_{4,2} = \frac{9}{75} = 0.12 > 0.10$ となり，やはり文字集合 S_2 を用いて検出する場合は攻撃文字列と誤判別してしまう．l_3 や l_4 は正常文字列であるが通常このような文字列を入力しないという意味で，このような極端な SQL クエリに対しても攻撃文字列として扱う運用も考えられるが，このような極端な入力に対する検出方法を考えることも今後の課題としたい．

例 3 でも示したとおり，電話番号・氏名・住所では区切り文字である「スペース」が多用されるため，攻撃検出に「スペース」を用いると正常誤検出を起こしやすくなる．同様に，顔文字にはセミコロンや括弧などの記号を多用するため，これらの文字を用いて攻撃検出を行うと正常誤検出が起きやすい．このような場合も 2 章で示したとおりつねに安定している保証はないが，攻撃検出に使用する適当な文字の組合せを選ぶことで正常誤検出率を小さくすることが可能である．提案モデルはベルヌーイモデルの混合モデルであるが，これは正常文字列のサンプルの上述の傾向に配慮して仮定したものである．以下，簡単のため $I = 1$，$b_1 = b$ として考察を行う．提案モデル

$$p_1(y|x,b) = (x^b)^y(1-x^b)^{1-y}$$

では入力文字列 l が攻撃文字列である確率を x^b で表している．一方，シグモイド関数を応用したモデル

$$p_2(y|x,b) = \left(\frac{x^b}{x^b+(1-x)^b}\right)^y\left(1-\frac{x^b}{x^b+(1-x)^b}\right)^{1-y}$$

では入力文字列 l が攻撃文字列である確率を $\frac{x^b}{x^b+(1-x)^b}$ で表している．$b=0.5$ と $b=0.1$ の場合について，関数 x^b と $g(x)=\frac{x^b}{x^b+(1-x)^b}$ のグラフの形状を**図 1**，**図 2** に示す．

関数 $g(x)$ は b の値が 0 に近づけば近づくほど閉区間 $[0,1]$ のほとんどの点で $\frac{1}{2}$ の値をとる．したがって，シグモイド関数を応用したモデル p_2 で SQL インジェクション攻撃の検出を行った場合，$0 \le b \le 1$ のどの b を選んでも $x \in [0, \frac{1}{2}]$ となる入力文字列 l に対して，l が攻撃である確率は $\frac{1}{2}$ 以下であると判定される．2 章の判別関数の計算例でも紹介したとおり，入力文字列 l が攻撃文字列である場合，l に対応する x の値は $\frac{1}{2}$ より小さな値をとることが多い．そのため，シグモイド関数を応用したモデル p_2 で攻撃検出を行った場合，攻撃であるにもかかわらず攻撃でないと判別する誤検出が起こる可能性が高くなるといえる．一方，正常文字列の性質と文献 [13] の実験結果をふまえると，入力文字列 l に対する x の値が 0.08 以上であれば l は攻撃文字列である可能性が高いことから，関数 x^b は SQL インジェクション攻撃の特徴をとらえた関数の一種であると考えられる．以上の考察から，我々の提案モデルは入力文字列 l に対する x の値が小さな値でも攻撃検出の確率を高くできるという意味で SQL インジェクション攻撃検出に有効であると考えられる．

最後に，定理 1 では提案モデルの予測誤差がシグモイド関数を応用したモデルの予測誤差より小さくなることを理論的に示したが，数値的にどの程度予測誤差が小さくなるかということを実験によって示す．**図 3**，**図 4**，**図 5** は以下の条件のもとで提案モデルとシグモイド関数を応用した場合のモデルの予測誤差の比較を行ったものである．

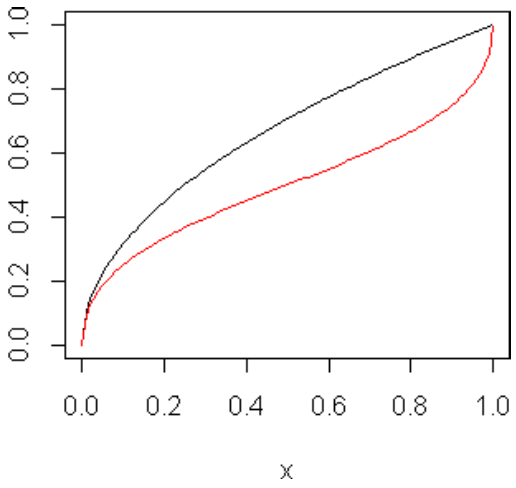

図 1　$b=0.5$ のときの x^b（黒線）と $g(x)$（赤線）
Fig. 1　x^b (black line) and $g(x)$ (red line) ($b=0.5$).

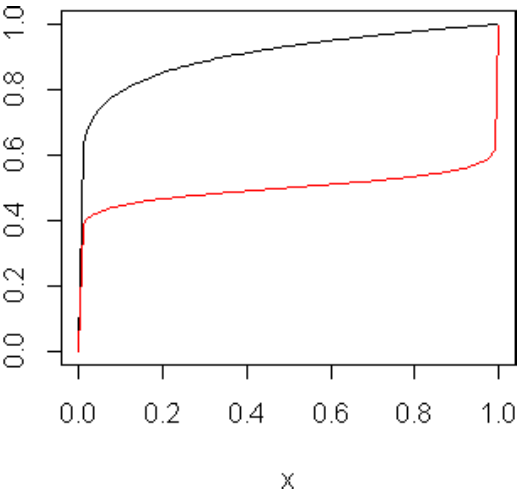

図 2　$b=0.1$ のときの x^b（黒線）と $g(x)$（赤線）
Fig. 2　x^b (black line) and $g(x)$ (red line) ($b=0.1$).

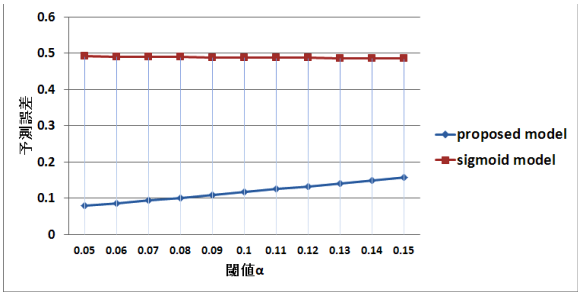

図 3　予測誤差の比較（$b=0.05$）
Fig. 3　Comparison of prediction error ($b=0.05$).

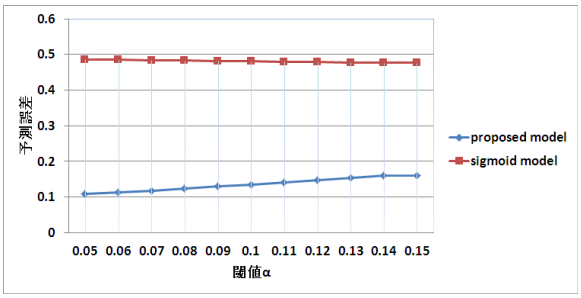

図 4　予測誤差の比較（$b=0.1$）
Fig. 4　Comparison of prediction error ($b=0.1$).

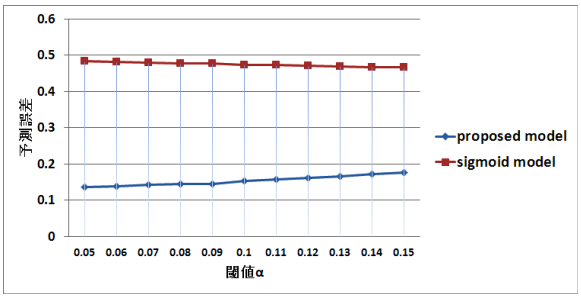

図 5　予測誤差の比較（$b=0.15$）
Fig. 5　Comparison of prediction error ($b=0.15$).

［条件］

- モデルのパラメータについて $b = 0.05$，0.1，0.15 の 3 つの場合を考える．
- 閾値 α を 0.05 から 0.15 まで 0.01 ずつ増加させた場合で予測誤差の計算を行う．

積分の計算にはニュートン・コーツ法を用い，ステップ長を 0.01 として計算した．図 3，4，5 の結果が示すとおり，$b = 0.05$，0.1，0.15 の場合では，提案モデルの予測誤差はシグモイド関数を応用したモデルの予測誤差より 3.5 以上小さくなることを確認することができた．

6. まとめ

本論文では，攻撃文字列に含まれる文字を攻撃特徴として SQL インジェクション攻撃を自動検出するモデルを提案し，提案モデルではシグモイド関数を応用したモデルの予測誤差より小さくできることを示した．提案モデルとシグモイド関数を応用したモデルとの予測誤差を比較した理由は

- パラメータを変更したときの提案モデルの性能と，攻撃検出率および正常誤検出率について調べたかったこと
- 他の研究 [10], [12] との比較では攻撃検出率や正常誤検出率の比較しかできないこと

にある．モデルの性能に関して提案手法と他の手法との比較法を開発し，提案モデルの有用性に対する検討を行うことは今後の課題である．また，本研究で使用したデータは文献 [18], [19], [20], [21] から収集したデータを基に作成した人工サンプルであり，SQL インジェクションの新しい攻撃パターンが開発された場合，攻撃検出に用いる文字の組合せが文献 [13] で選ばれたものから変わることも考えられる．そのため，実データの収集を行い，他のサンプルを用いた場合の提案モデルの有用性について検証することも重要な課題である．また，提案モデルのパラメータを最適化する方法や他の Web アプリケーション攻撃への適用について検討を行うことも今後の課題である．

謝辞　本研究の一部は，独立行政法人日本学術振興会学術研究助成基金助成金（若手研究（B），課題番号：23740094）および平成 23 年度（第 44 回）倉田奨励金の助成による．

参考文献

[1] NT Web Technology Vulnerabilities, available from ⟨http://www.phrack.com/issues.html?issue=54⟩.
[2] 独立行政法人情報処理推進機構：ソフトウェア等の脆弱性関連情報に関する届け出状況（2010 年第四半期），入手先 ⟨http://www.ipa.go.jp/security/vuln/report/vuln2010q4.html⟩.
[3] 小菅祐史，花岡美幸，河野健二：SQL インジェクション攻撃の脆弱性の効果的な自動検出手法，情報処理学会研究報告 CSEC［コンピュータセキュリティ］，Vol.45, pp.103–108 (2008).
[4] Robertson, W., Vigna, G., Kruegel, C. and Kemmerer, R.A.: Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks, *Proc. Network and Distributed System Security* (*NDSS*) *Symposium*, San Diego, CA (Feb. 2006).
[5] Valeur, F., Mutz, D. and Vigna, G.: A Learning-Based Approach to the Detection of SQL Attacks, available from ⟨http://www.cs.ucsb.edu/?vigna/publications/2005 valeur mutz vigna dimva05.pdf⟩ (2005).
[6] Kirda, E., Kruegel, C., Vigna, G. and Jovanovic, H.: Noxes: A Client-Side Solution for Mitigating Cross Site Scripting Attacks, *Security Track of the 21st ACM Symposium on Applied Computing* (*SAC 2006*), Dijon, France (Apr. 2006).
[7] Rietta, F.S.: Application layer intrusion detection for SQL injection, *Proc. 44th Annual Southeast Regional Conference*, *ACMSE 44* (2006).
[8] Kosuga, Y., Hanaoka, M. and Kono, K.: Effective Automated Testing for Detecting SQL Injection vulnerabilities, *The Special Interest Group Notes of IPSJ*, *CSEC*, pp.103–108 (2008).
[9] Song, Y., Keromytis, A.D. and Stolfo, S.J.: Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic, *Proc. 16th Annual Network and Distributed System Security Symposium* (*NDSS*) (2009).
[10] Ariu, D. and Giacinto, G.: HMMPayl: An application of HMM to the analysis of the HTTP Payload, *Proc. JMLR: Workshop and Conference*, Vol.11, pp.81–87 (2010).
[11] Tajpour, A., Masrom, M., Heydari, M.Z. and Ibrahim, S.: SQL injection detection and prevention tools assessment, *2010 3rd IEEE International Conference on Computer Science and Information Technology* (*ICCSIT 2010*), pp.518–522 (2010).
[12] 伊波　靖，高良富夫：SVM を用いた WAF の検知機能の提案，情報処理学会全国大会講演論文集，pp.445–447 (2011).
[13] Sonoda, M., Matsuda, T., Koizumi, D. and Hirasawa, S.: On the Automatic Detection of the SQL Injection Attacks by the Feature Extraction of the Single Character, *Proc. 4th International Conference on Security Information and Networks*, pp.81–86, ACM (2011).
[14] Matsuda, T., Koizumi, D., Sonoda, M. and Hirasawa, S.: On Predictive Errors of SQL Injection Attack Detection by the Feature of the Single Character, *Proc. 2011 IEEE International Conference on Systems, Man and Cybernetics*, pp.1722–1727 (2011).
[15] Isa, I.S., Saad, Z., Omar, S., Osman, M.K., Ahmad, K.A. and Sakim, H.A.M.: Suitable MLP Network Activation Functions for Breast Cancer and Thyroid disease Detection, *Proc. 2nd International Conference on Computational Intelligence, Modeling and Simulation* (2010).
[16] Takiguchi, T., Sako, A., Yamagata, T. and Ariki, Y.: *System Request Utterance Detection Based on Acoustic and Linguistic Features*, pp.539–550, I-Tech Education and Publishing (2008).
[17] Ko, J.H., Joo, J.S. and Lee, Y.H.: On the Use of Sigmoid Functions for Multistage Detection in Asynchronous CDMA Systems, *IEEE Trans. Vehicular Technology*, Vol.48, No.2, pp.522–526 (1999).
[18] Clarke, J.: *SQL Injection Attacks And Defense*, Syngress Publishing Inc. (2009).
[19] SQL Injection Cheat Sheet, available from ⟨http://ferruh.mavituna.com/

sql-injection-cheatsheet-oku/⟩.
[20] SQL Injection Cheat Sheet, available from ⟨http://michaeldaw.org/sql-injection-cheat-sheet⟩.
[21] MySQL SQL Injection Cheat Sheet, available from ⟨http://pentestmonkey.net/blog/mysql-sql-injection-cheat-sheet⟩.
[22] Bishop, C.: *Pattern recognition and machine learning*, Springer (2006).
[23] 小西貞則，竹内純一，若山正人：統計的モデリング/情報理論と学習理論，講談社サイエンティフィク (2008).

松田 健　（正会員）

1979 年生．2010 年東京工業大学大学院総合理工学研究科知能システム科学専攻博士課程修了．2011 年よりサイバー大学 IT 総合学部専任講師．情報数理の研究に従事．日本応用数理学会会員．