情報処理学会論文誌　データベース　Vol. 2　No. 4　13–23 (Dec. 2009)

ストリームカーネルマシンによるパラレルブースティング

新美礼彦[†1]　小西修[†1]

データストリームは動的な大規模データであり，従来の静的なデータを対象としていたアルゴリズムをそのままデータストリームへ適用しても，高い性能は得られない．そこで我々はカーネル法を拡張して，大規模データストリームに有効に作用するストリームカーネルを提案した．これはデータストリームのある時点のデータだけではなく，そこから遡及して得られるいくつかの履歴情報をまとめた構造データを入力に用いるカーネル関数の枠組みであり，履歴情報から得られるデータストリームの時間的な変化を特徴にする．ストリームカーネルを適用した非線形サポートベクタマシン（SVM）は，実験で従来の非線形 SVM を上回る性能を示したが，多くの計算時間を要した．我々はこの問題をパラレルブースティングによって解決する．約 70 万件の実際のクレジットカードデータを用いた不正利用と正常利用の識別実験において，パラレルブースティングにより弱学習器の数を 10 以上にすると学習時間・検証時間を大幅に削減し，汎化誤差も削減する成果を得た．

Parallel Boosting with Stream Kernel Machine

Ayahiko Niimi[†1] and Osamu Konishi[†1]

Kernel-based Support Vector Machines (SVMs) have strong theoretical foundations and excellent empirical successes. The performance for the real data stream with the evolutionary changes is however not enough. Thus it is an important challenge to improve the performance. On the other hand, the kernel methods have be extended to input structured data such as string or graph. The classification using the kernel methods with structured form outperform vector form. This paper proposes non-linear classification approach to consider data stream as structured data by extended kernel methods. This structured data is constructed with the arrived online data and some historical information of the data stream. Then, the features of the data stream's evolutionary changes are obtained, and kernel function is built with the features. We call this a stream kernel. In the experiment, we apply the stream kernel to non-linear SVM, and differentiate normal use and illegal use for about 700,000 real credit card data. Our approach achieves that training time and evaluation time became faster than non-parallel SVM, and a number of the false classification decreased.

1. はじめに

近年のコンピュータ技術の発達により，様々な領域で大規模データの収集・蓄積・処理が可能となった．これにともない，データストリームが新しいタイプの大規模データとして注目を集めている．データストリームとは，膨大な量のデータが，高速なストリームを通じて，時間的に変化しながら，終わりなく到着し続けるという特性を持つ動的な大規模データである[1),2)]．たとえば Web ログ，センサネットワーク，金融や流通の取引記録，クレジットカードデータなどはその典型的な例である．そして現在，この蓄積されたデータをいかに分析し活用するかが重要な課題となっている．大量のデータから知識やルールを発見するための分析手法であるデータマイニングや機械学習は，その有力な手法である．しかし，データストリームは上述した特性を持つ動的な大規模データであり，従来の静的なデータを対象としていたアルゴリズムをそのままデータストリームへ適用しても，高い性能は得られない．このようなデータストリームを分析する手法として，我々はカーネル法[3),4)] を用いる．3.1 節で述べるが，カーネル法は元のデータ空間を高次元空間へ埋め込み，その空間で線形アルゴリズムを適用する．その計算はカーネル関数と呼ばれる高次元空間上の内積を表す関数で行われ，実際には高次元へ写像することなく，高次元空間上での動作を可能にする．この仕組みはカーネルトリックと呼ばれ，カーネルを用いた学習モデル（カーネルマシン）はカーネルトリックによって容易に非線形モデルを構成することができる．特に識別において，線形識別器であるサポートベクタマシン（SVM）にカーネルトリックを適用し，非線形識別関数を構成することで非線形識別を可能にした非線形 SVM[5)] は，最も識別性能に優れたモデルの 1 つである．

そこで我々は，カーネル法を拡張し，大規模データストリームに有効に作用するストリームカーネルを提案した[6),7)]．ストリームカーネルは，データストリームのある時点のデータだけでなく，そこから遡及して得られるいくつかの履歴情報をまとめたものを，1 つのデータ構造として入力に用いるカーネル関数の枠組みである．これにより，履歴情報から得られるデータストリームの時間的な変化を特徴にすることができる．大規模な実際のクレジットカードデータを用いた不正利用と正常利用の識別実験において，ストリームカーネルを用いた非線形 SVM は，構造データを用いない従来の非線形 SVM を上回る識別性能を示した

†1 公立はこだて未来大学システム情報科学部
School of Systems Information Science, Future University-HAKODATE

が，多くの計算時間を要した．

一方で，性能が低い学習モデル（弱学習器）を複数組み合わせて，最終的に強力な学習モデルを構築するアンサンブル学習が注目され，1 つの強力な学習モデルを構築する場合よりも優れた性能を得ている．バギング[8] やブースティング[9),10)] はその代表的な手法である．しかし，バギングは並列処理が可能である反面，一般的に性能はブースティングに劣る．また，ブースティングは逐次処理であるため多量の計算時間がかかるという問題がある．これらの利点をあわせ持つパラレルブースティング[11)–13)] は，並列処理が可能であり，かつ，バギングと同等かそれ以上の性能を得ることができる．特に，文献 11) では弱学習器にカーネルマシンを用いたパラレルブースティングの手法が提案され，1 つの非線形 SVM を用いるよりも計算時間を削減し，性能を改善できることが示されている．ただし，データ数の小さなベンチマーク用のデータセットを用いて評価しており，計算時間の大幅な軽減を見るには至っていない．我々の研究では，提案したストリームカーネルにより，実際の大規模データでパラレルブースティングの有効性を実証する．

そこで我々は，ストリームカーネルを用いたカーネルマシン（ストリームカーネルマシン）が持つ計算時間の問題を，パラレルブースティングによって解決する．また，これにより得られる多くの利点を以下にまとめる．

高速処理：　データストリームは，膨大な量のデータが高速に次々と到着し続けるため，高速な処理を行わなければならない．ストリームカーネルマシンは多くの計算時間を要するが，これを解決できる．

性能向上と安定性：　1 つのカーネルマシンを用いるよりも，性能を改善できることが示されている．また，アンサンブルモデルには安定性という重要な利点がある．

拡張性：　ストリームカーネルマシンは，データストリームのある時点のデータだけではなく，履歴情報を含めた構造データを入力するため，多くの計算資源を必要とする．パラレルブースティングにより，1 つのストリームカーネルマシンでは扱えない規模のデータを用いることができる．

順応性：　データストリームは時間的な変化をともなうため，頻繁に学習モデルを最新のものへ更新する必要がある．パラレルブースティングにより，この更新は新たな学習データから構築した弱学習器を，アンサンブルモデルへ組み込むだけで済む．

特に，過去のデータによる弱学習器と新たに到着したデータによる弱学習器によるアンサンブルモデルについて述べておく．これにより，全データの到着を待たなくても，ある程度データが到着した時点で弱学習器を構築することにより，その時点までの学習結果をアンサンブルモデルに取り込むことが可能となり，ストリームカーネルによる履歴情報を含めている点とあわせて，ストリームデータに対する効率的な学習が行える．

これらのことから，ストリームカーネルマシンにとってパラレルブースティングは良い選択である．さらにここで，本実験で用いるデータが大規模な実際のデータストリーム（クレジットカードデータ）であることを強調する．大規模，かつ，実際のデータであるため信頼性はもちろん，クレジットカードデータは多重属性のデータストリームであるため，他のあらゆるデータストリームに対する普遍性を持つと考える．クレジットカードデータのようなストリームデータでは，データ発生時点では教師データ（そのデータが正常利用なのか不正利用なのか）の判断できない場合が多い．この点に関しても，提案手法は新たに教師データが判明したデータから学習を行えるという利点がある．

クレジットカードデータを用いた正常利用と不正利用の識別実験では，ストリームカーネルを用いた非線形 SVM が，パラレルブースティングによって計算時間を短縮させながら，識別性能を向上させることを示す．また同時に，従来の非線形 SVM とも性能を比較し，我々が提案したストリームカーネルがパラレルブースティングにおいても有効であることを示す．

本論文は以下のように構成されている．2 章では関連研究について述べる．3 章では，カーネル法と我々が提案したストリームカーネル[6),7)] について述べる．4 章で，非線形 SVM を用いたパラレルブースティングと，ストリームカーネルの適用について述べ，大規模な実際のクレジットカードデータを用いた識別実験の結果を 5 章で示し，最後に 6 章で結言と今後の課題を述べる．

2. 関 連 研 究

大規模に蓄積されたデータストリームからの知識発掘は，多くのアプリケーションにおいて有用であることから活発に研究が行われている[14),15)]．特に，カーネル法[3)] を用いた学習アルゴリズムは強力であり，実験的な結果からも多くのデータマイニング，機械学習のアルゴリズムへ適用されている．また，カーネル法は適切にカーネル関数を設計することで，配列[16)] やグラフ[17)] などの構造を持つデータを入力に扱えるように拡張された．このような構造データを入力に用いると，ベクトル形式では失われていたデータの特徴を分析に反映させることができ，従来のベクトル形式のデータを入力に用いるよりも高い性能が得られている．

そこで我々は，データストリームのある時点でのデータだけではなく，そこから遡及し

て得られるいくつかの履歴情報をまとめたものをストリームカーネルとして提案した[6),7)]．また，線形識別器である SVM にカーネルトリックを適用し，非線形識別関数を構成することで非線形識別を可能にした非線形 SVM[5)] は最も識別精度に優れたモデルの 1 つである．多くの研究によってその有用性が示されており[18)]，我々もストリームカーネルを非線形 SVM へ適用し，実際のクレジットカードデータを用いて正常利用と不正利用への識別実験を行い，ストリームカーネルを用いた非線形 SVM は従来の非線形 SVM を上回る性能を示した．しかし，実験では多くの計算時間を要した．

膨大で高速なデータストリームを扱うために，計算時間の短縮は重要である．その最も単純な方法は，オンライン学習[19)] や分散・並列アルゴリズム[20),21)] である．しかし，オンライン学習はバッチ学習に比べ，優れた汎化性能が得られない場合がある．また，分散・並列アルゴリズムは汎化性を保ち，かつ高速処理が可能であるが，これらの手法は使用する学習モデルに強く依存する．ストリームカーネルは非線形 SVM だけではなく，他の多くのカーネルマシンにおいても有効であると考えるため，我々は計算時間の問題をパラレルブースティングによって解決する．

パラレルブースティング[11)–13)] はアンサンブル学習の 1 つであり，分割された学習データのサブセットからそれぞれ異なる学習器を構築し，重み付き平均によってこれらを組み合わせる．データストリームに対するアンサンブル学習も提案されている[2),22)]．特に文献 2) では Concept-Drift という考え方を用いて，クレジットカード取引データを取引時間順と取引金額順の 2 つのストリームに対してクレジットカードの不正利用検出の実験を行っている．Concept-Drift とは学習対象の統計的性質の時間的変化のことである．文献 12)，13) ではブースティングを並列処理で実現する手法を提案しているが，カーネルマシンはこれに適用できない．これに対し，文献 11) では，弱学習器にカーネルマシンを用いたパラレルブースティングの手法が提案され，1 つの非線形 SVM を用いるよりも計算時間を削減し，識別性能も改善できることが示されている．しかし，文献 11) ではデータ数の小さなベンチマーク用のデータセットを用いて評価しており，膨大なデータ数の学習セットに関して特に有効的であると考えられるとしながらも，計算時間の大幅な軽減を見るには至っていない．したがって，我々はパラレルブースティングにより，ストリームカーネルマシンが持つ計算時間の問題を解決する．これにより，大規模データに対して高精度かつ短時間なモデル構築を行う手法を提案する．

データストリームへの SVM として，Incremental Support Vector Learning[23),24)] が提案されている．しかし，本論文で取り扱うクレジットカードデータは全データ中の不正利用の割合が 0.03%ときわめて低いため，Incremental Support Vector Learning では十分な汎化性能が得られない可能性が高い．これに対し，我々の手法ではストリームカーネルとパラレルブースティングを組み合わせることにより，汎化性能を落とさずにデータストリームの学習を行える仕組みを提案している．

3. ストリームカーネル

3.1 カーネル法

時間的な変化をともなう，動的な大規模データであるデータストリームを扱うために，我々は文献 6) でカーネル法を拡張した．本節ではまずカーネル法について概説する．カーネル法はデータの非線形構造を捉える強力な手法であり，以下の 2 つの本質的な要素からなる．

(1)　データを高次元特徴空間に埋め込む．

(2)　高次元特徴空間で，線形識別アルゴリズムを適用する．

しかし，カーネル法はデータを高次元特徴空間では明確に示さず，カーネル関数と呼ばれる半正定値関数 $K : \chi \times \chi \to R$ を用いて，高次元特徴空間での動作を可能にする．

$$K(\boldsymbol{x}, \boldsymbol{z}) = < \phi(\boldsymbol{x}), \phi(\boldsymbol{z}) > \tag{1}$$

Mercer の定理を満たす半正定値関数 K をカーネル関数と呼び，これは ϕ によって写像された高次元特徴空間での内積を意味する．このように実際に高次元特徴空間へ写像する計算を避け，入力空間でのカーネル関数を用いて高次元特徴空間上での内積を計算する仕掛けをカーネルトリックと呼ぶ．また，カーネル関数は 2 つの入力の類似度を定めていると考えることができる．これは適当に類似度 R を定義した関数 K を，カーネル関数として学習に組み込むことができることを示している．したがって，上述したカーネル関数の 2 つの入力 $\boldsymbol{x}_i$，$\boldsymbol{x}_j$ がベクトルではなく，構造を持つデータに対しても，カーネル法を適用することができる．我々はこの考えをデータストリームへ適用し，データストリームを構造データと見なして入力に用いるカーネル関数の枠組みを提案した．これをストリームカーネルと呼び，3.2 節で簡単に述べる．

3.2 ストリームカーネルの導出

データストリームに対する従来の学習アルゴリズムは，ある時点のデータをベクトル $\boldsymbol{x} = (x_1, x_2, \ldots, x_d)$ で表現し，これを入力に用いる．これに対し，我々は次の特徴を持つ構造データを入力に用いる．

(1)　ある時点のデータ $\boldsymbol{x}^{(1)}$ だけでなく，過去の履歴 n 個のデータを持つ．

(2)　n 個の前後のデータ間に，$n-1$ 個の時間間隔を持つ．

我々はこれをストリーム構造データと定義し，次の用に表現する．

$$\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(n)}\} \tag{2}$$

なお，$\boldsymbol{x}^{(1)}$ が新しいデータ，$\boldsymbol{x}^{(n)}$ が古いデータとし，括弧内の上付き添え字を履歴番号と呼ぶ．時間間隔とは，データが到着した時間の間隔であり，たとえば $\boldsymbol{x}^{(1)}$ と $\boldsymbol{x}^{(2)}$ の時間間隔は $t[1,2]$ と表す．また，時間間隔は $\sum_{i=1}^{n-1} t[i,i+1] = 1$ に正規化されている．このような構造を持つデータ間のカーネル関数の構築は，畳み込みカーネル[25]が基礎理論となっている．

次に，遡って得られたデータが，学習アルゴリズムにどれほどの影響を与えるかについて考える．あくまで学習アルゴリズムが対象とするのは，最も新しいデータ $\boldsymbol{x}^{(1)}$ であるため，新しいデータほど学習アルゴリズムに与える影響度を大きく，古いデータほど影響度を小さくすべきである．この影響度を，時間間隔 t と，新たに導入するパラメータ $\lambda \in (0,1)$ を用いた単調減少関数 λ^t で表す．

これらの変数を導入して，ストリーム構造データ $\boldsymbol{X}$ と $\boldsymbol{Z}$ の間のカーネル関数 $K(\boldsymbol{X}, \boldsymbol{Z})$ の枠組みを定義する．我々はこれをストリームカーネルと呼ぶ．まず，$K_n(\boldsymbol{X}, \boldsymbol{Z})$ を，$\boldsymbol{X}$ と $\boldsymbol{Z}$ がともに過去 n 件のストリーム構造データからなるときのストリームカーネルとする．また，過去 n 件からなる $\boldsymbol{X}$ に，さらにもう 1 件遡って得られる $\boldsymbol{x}^{(n+1)}$ が，$\boldsymbol{X}$ に付与されることを $\boldsymbol{X}\boldsymbol{x}^{(n+1)}$ と表現する．これは $\boldsymbol{Z}$ についても同様である．遡るデータが 1 つ増えることで，全体のカーネルに加えられる部分構造データのカーネルの量を J_n とすると，$K_n(\boldsymbol{x}, \boldsymbol{z})$ は以下の式のように再帰的に表現することができる．

$$K_n(\boldsymbol{X}\boldsymbol{x}^{(n)}, \boldsymbol{Z}\boldsymbol{z}^{(n)}) = K_{n-1}(\boldsymbol{X}, \boldsymbol{Z}) + J_n \tag{3}$$

$$J_n = (1 + J_{n-1}) K(\boldsymbol{x}^n, \boldsymbol{z}^n) \prod_{i=1}^{n-1} \lambda^{T[i,i+1]} \tag{4}$$

ただし，$K_0(\boldsymbol{X}, \boldsymbol{Z}) = 1$，$J_1 = K(\boldsymbol{x}^{(1)}, \boldsymbol{z}^{(1)})$ であり，$K(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)})$ は 3.1 節で述べた Mercer カーネルである．動的計画法を用いることにより，この計算量は遡るデータ数 n に対して $O(n)$ で済む．また，$\boldsymbol{X}$ と $\boldsymbol{Z}$ のデータ数が異なる場合，このままでは，データ数が少ない部分集合のカーネルのみを考えてしまう．たとえば，$\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3\}$，$\boldsymbol{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2\}$ とすると，$\boldsymbol{x}_3$ はカーネルの計算にまったく用いられない．そこで，以下の式で正規化を行い，これをストリームカーネルの最終出力値とする．

$$K(\boldsymbol{X}, \boldsymbol{Z}) = \frac{K_n(\boldsymbol{X}, \boldsymbol{Z})}{\sqrt{K_n(\boldsymbol{X}, \boldsymbol{X}) K_n(\boldsymbol{Z}, \boldsymbol{Z})}} \tag{5}$$

式 (4) で示すように，ストリームカーネルは計算過程で Mercer カーネルを使用する．したがってストリームカーネルは，既存の Mercer カーネルを使用して，データストリームへ適用できるように拡張したカーネル関数の枠組みである．また，このデータストリームが半正定値性を満たすことは容易に示すことができる．本節ではストリームカーネルについて簡単に述べたが，詳細な説明は文献 6) を参照されたい．

4. ストリームカーネルマシンのパラレルブースティング

4.1 アンサンブル学習

一般的なカーネル関数（ガウスカーネルやシグモイドカーネルなど）と比べ，3.2 節で述べたストリームカーネルは，遡る履歴数を n とすると $O(n)$ の計算量を必要とする．文献 6) で示すように，ストリームカーネルを用いた非線形 SVM は従来の非線形 SVM に比べ多くの計算時間を要した．我々はこの問題をアンサンブル学習の 1 つであるパラレルブースティングによって解決する．アンサンブル学習は，初めから 1 つの強力な学習モデルを構築するのではなく，性能が低い学習モデル（弱学習器）を複数組み合わせて，最終的に強力な学習モデルを構築する学習法である．ブースティングはその代表的な手法であり，本節ではブースティングとパラレルブースティングの違いについて述べる．

まず，一般的なブースティングによる学習の手順を**図 1** に沿って述べる．ブースティングは，正しく識別された学習データの重要度を下げ，誤って識別された学習データの重要度を上げるように，学習データの重みを変化させながら逐次的に異なる弱学習器を構築していく．図 1 の丸の大きさは，この重みの大きさを表している．最初はすべての学習データ $(x_1, x_2, \ldots, x_N)$ に等しく重みが与えられ，この重みを用いて 1 つ目の弱学習器 f_1 を構築する．次に，学習データに対応する正解 y_k と f_1 の出力結果を比較し，構築した f_1 が正しく識別した学習データの重みを下げ，誤って識別した学習データの重みを上げる．図では × を識別に失敗したデータ，○ を識別に成功したデータとして表している．次の弱学習器 f_2 は，この変化させた重みを用いて構築される．このような処理を繰り返すことで，誤って識別した学習データを重点的に学習することになる．図では M 回学習を繰り返し，弱学習器 $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})$ をもとにアンサンブル学習器 $F(\boldsymbol{x})$ を構築した．これがブースティングの特徴であり，AdaBoost や LogitBoost などはこのような学習法に基づく代表的なアルゴリズムである[8)–10)]．**図 2** に，ブースティングのアルゴリズムを示す．

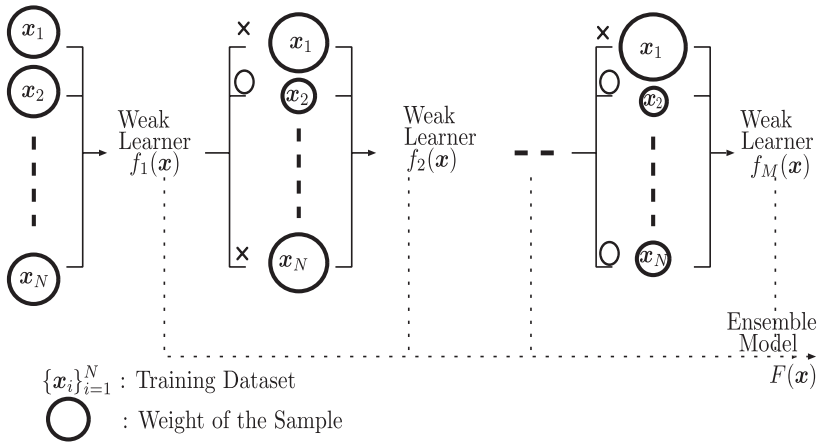

図 1　ブースティングによる学習
Fig. 1　Learning by boosting.

Algorithm Boosting(T,M)
1. For k = 1 to N
2. $D_1(x_i) = 1/N$
3. For $i = 1$ to M
4. f_i = weak learned model from T with distribution D_i
5. For $k = 1$ to N
6. Reset $w(x)$ using $f_i(x_k, w)$ and y_k
7. D_{i+1} = updated distribution by w
8. For each new query instance q
9. F = ensemble model from all f and w
10. Class(q) = F(q)

where:
T is the training set
N is the number of training set
M is the number of samples drawn from T

図 2　ブースティング・アルゴリズム
Fig. 2　Boosting algorithm for classification.

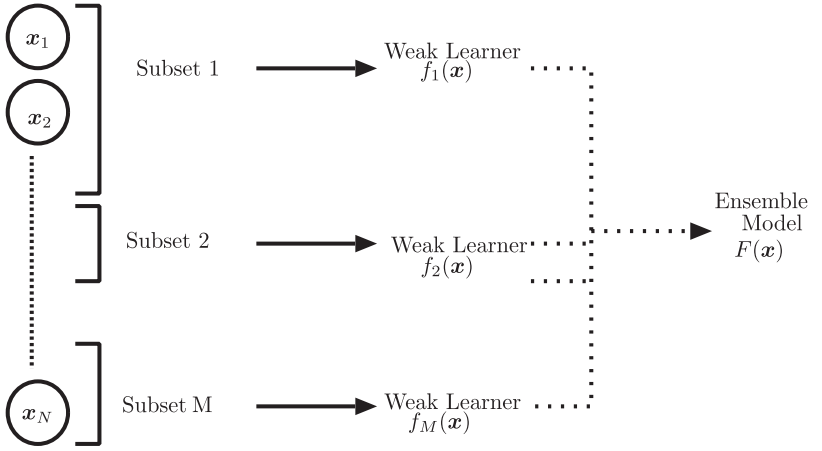

図 3　パラレルブースティングによる学習
Fig. 3　Learning by parallel boosting.

次に，パラレルブースティングによる学習の手順を**図 3** で示す．パラレルブースティングは，上述したような学習データの重みの逐次的な操作は行わない．まず，全学習データ $(x_1, x_2, \ldots, x_N)$ を M 個のサブセット（Subset 1, Subset 2, …, Subset M）に分割し，各サブセットからそれぞれ並列的に異なる弱学習器（$f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_M(\boldsymbol{x})$）を構築する．ここで，図 3 では理解が単純になるよう，非復元抽出による分割を示しているが，一般的には復元抽出が用いられることに注意する．次に，構築した弱学習器を組み合わせてアンサンブル学習器（$F(\boldsymbol{x})$）を構築するが，このとき，単純に平均をとればバギング[8] と呼ばれる手法と変わらない．パラレルブースティングは，各弱学習器の出力の重み付き平均をとり，この重みは，一般的なブースティングで用いられる評価関数を最適化するように決定される．**図 4** に，パラレルブースティングのアルゴリズムを示す．

AdaBoost や LogitBoost のような一般的なブースティングの処理は逐次的であるため，計算時間は構築する弱学習器の数に比例して長くなる．しかし，パラレルブースティングの処理は並列的であるため，並列コンピュータを用いれば計算時間は大きく削減できる．弱学習器にカーネルマシンを用いたパラレルブースティングの手法が提案されている[11)–13)]．これを用いれば，3.2 節で述べたストリームカーネルは，弱学習器として用いるカーネルマシ

Algorithm ParallelBoosting(T,M)
1. For $i = 1$ to M
2. S_i = random sample drawn from T
3. f_i = weak learned model from S_i
4. F = ensemble model by weighted summation of all f
5. For each new query instance q
6. Class(q) = $F(q)$

where:
T is the training set
M is the number of samples drawn from T

図 4　パラレルブースティング・アルゴリズム
Fig. 4　Parallel boosting algorithm for classification.

ンに適用することで容易にパラレルブースティングに導入できる．したがって我々はパラレルブースティングによってストリームカーネルマシンが持つ計算時間の問題を解決する．

4.2　非線形 SVM

本節ではパラレルブースティングの弱学習器として用いる非線形 SVM と，3.2 節で述べたストリームカーネルの適用について述べる．SVM は非線形識別器として提案された学習モデルであり，異なるクラスのデータ間の距離（マージン）を最大にする超平面を考える．このときの目的関数と識別関数は，特徴ベクトルの内積のみに依存した形で記述される．たとえば識別関数は，クラスラベル変数 t，サポートベクタの集合 S，Lagrange 乗数 $\alpha_i^* > 0$，最適な閾値 h^* を用いて次式で書ける．

$$f(x) = \sum_{i \in S} \alpha_i^* t_i \boldsymbol{x}_i^T \boldsymbol{x} - h^* \tag{6}$$

この線形識別関数は，カーネルトリックを適用することにより以下の形に書き換えることができる．

$$f(x) = \sum_{i \in S} \alpha_i^* t_i \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}) - h^* \tag{7}$$

$$= \sum_{i \in S} \alpha_i^* t_i K(\boldsymbol{x}_i, \boldsymbol{x}) - h^* \tag{8}$$

ここで，元の空間よりも高次元に写像するカーネル関数 K を選択すれば，高次元空間上で非線形 SVM を実行したことになり，これは元の空間で非線形識別を実行したことと等価である．このように，カーネルトリックによって非線形識別関数を構成することで，非線形識別を可能にした SVM を，非線形 SVM という．また，カーネル関数に関して 3.2 節で述べたストリームカーネルを選択すれば，データストリームに対して有効に作用するストリームカーネルマシンとなる．

4.3　パラレルブースティング

パラレルブースティングは，学習データセット全体の集合 $D = \{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^N$ を，M 個のサブセットに分割し，各サブセットから独立に生成されたカーネルマシンを組み合わせるものである．ストリームカーネルはこのカーネルマシンに適用することで，容易にパラレルブースティングに適用できる．まず，構築した M 個のカーネルマシンを $\{\boldsymbol{f}_m\}_{m=1}^M$ とし，図 3 のようにしてこれを組み合わせたアンサンブルマシン $F(\boldsymbol{x})$ を以下で表現する．

$$F(\boldsymbol{x}) = \sum_{m=1}^{M} c_m f_m(\boldsymbol{x}) \tag{9}$$

f_m に重み付けられている係数 $\boldsymbol{c} = \{c_i\}_{m=1}^M$ は，一般的なブースティングで用いられる評価関数を最適化することによって得られる．我々は過学習に対してロバストな結果が得られている以下の指数評価を用いる．

$$Z(\boldsymbol{c}) = \sum_{i=1}^{N} \exp(-y_i F(\boldsymbol{x}_i)) \tag{10}$$

係数 $\boldsymbol{c}$ はこれを最小化することにより得られる．ここで，式 (10) は全学習データセットを用いて定義されていることに注意する．また文献 11) では，これに $\sum_{i=1}^{l} c_i = 1$，$0 \le c_i \le 1$ という制約を加えることで過学習を回避することができる結果が述べられている．したがって我々は実験でこの制約を用いる．次に，学習データセット全体の集合 D をどのようにサブセットへ分割するかを考える必要がある．本実験では全学習データから復元抽出を行い，ランダムに弱学習器の数に分割した．

パラレルブースティングの注目すべき特徴は，並列処理が可能なことであり，並列コンピュータを用いれば，計算時間は大きく削減できる．

ストリームカーネルで新しいデータほど学習に与える影響を大きく，古いデータほど学習に与える影響が小さくなるように設計してあるため，ストリームカーネルを用いたパラレルブースティングにおいてもデータをランダムに分割した場合は，新しいデータほど学習に与える影響が大きくなるという特徴を持つ．また，過去のデータによる弱学習器と，新に到着したデータによる弱学習器によるアンサンブルモデルを考えることもできる．これにより，

全データの到着を待たなくても，ある程度データが到着した時点で弱学習器を構築することにより，その時点までの学習結果をアンサンブルモデルに取り込むことが可能となる．

5. 実験評価

本実験では，実際のデータストリームであるクレジットカードデータを用いて，正常利用と不正利用への識別を行った．記述の簡略化のため，以降では実験で用いた学習モデルを以下のように記す．なお，SKSVM と SKSVM-PBOOST が，我々が文献 6) で提案したストリームカーネルを適用した識別モデルである．

SVM： 従来の非線形 SVM

SKSVM： ストリームカーネルを適用した非線形 SVM

SVM-PBOOST： SVM を弱学習器に用いたパラレルブースティング

SKSVM-PBOOST： SKSVM を弱学習器に用いたパラレルブースティング

クレジットカードデータは文献 6) で用いたものと同じものである．クレジットカードデータは，約 2 年分のデータが用意されており，そのデータ量は約 1 TB に及び．クレジットカードデータの属性は，大きく次の 2 つのグループに分類される．

(1)　オーソリデータ属性：利用時の状況を記述した属性

(2)　振舞いデータ属性：顧客の行動パターンを記述した属性

属性 (1) は，単にクレジットカードの取引データの属性である．顧客 ID，生年月日，利用金額，購入商品コードなどがあり，計 84 属性からなる．属性 (2) は，過去の履歴情報から作成した顧客モデル（過去の利用金額の平均や分散，過去の利用時間帯の頻度など）との乖離値であり，顧客の行動パターンを記述した属性であり計 40 属性からなる．本実験で用いたクレジットカードデータセットは，属性 (1) と属性 (2) から 55 属性を分析に用いた．実験で用いたデータ件数，正常利用と不正利用の割合に関しては，次の節で詳しく述べる．

5.1 実験準備

弱学習器に用いる SVM は OSS である svm-light [26]（C 言語）を用い，SKSVM はこれを改良して実装した．使用したカーネル関数はガウスカーネル $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/\sigma^2)$ である．ストリームカーネルのパラメータ λ は，新しいデータであるほど識別の影響が大きく，古いデータであるほど識別の影響を小さくする重みであり，3.2 節で詳述したように一番最近のデータが全体の影響の 7 割程度を占めるように設定した．また，顧客の遡る履歴数 n は $n = 5$ に設定した．これら実験で用いたパラメータを**表 1** で示す．

次に，実験で使用したクレジットカードデータを**表 2** に示す．ただし，ストリームカーネルを用いた学習モデル（SKSVM，SKSVM-PBOOST）の学習・検証データの各レコードには，利用時点から最大で過去 5 回の履歴情報と時間間隔が付与されている．パラレルブースティングの弱学習器の数は，10，30，50，100，200 で実験を行った．また，各弱学習器が用いる学習データサブセットは，全学習データから復元抽出を行い，ランダムに弱学習器の数に分割した．弱学習器の数と使用するデータ量の関係は**表 3** のようにした．弱学習器の数と 1 つの弱学習器が使用するデータの量は，学習器を増やしたとき，データセットを小さくするようにした．学習データの内訳は，正常利用が 384,516 件，不正利用が 4,096 件である．後述する検証データと正常利用と不正利用の割合が異なるのは，検証データと同じ割合で学習させると，不正利用を正しく学習できなくなる可能性があるためである．

表 1　実験で用いたパラメータ
Table 1　The parameters used in the experiment.

	σ	C	n	λ
SVM，SVM-PBOOST	0.05	5	–	–
SKSVM，SKSVM-PBOOST	0.05	5	5	0.1

表 2　実験データ
Table 2　Credit card dataset in our experiemnt.

全データ	学習データ	検証データ	属性
1,109,531	388,611	720,920	55

表 3　弱学習器の数と使用するデータ量の関係
Table 3　Relation between amount of data and number of weak learner.

弱学習器の数	1 つの弱学習器が使用する学習データの量
10	約 38,861 件 (10%)
30	約 12,953 件 (3%)
50	約 7,772 件 (2%)
100	約 3,886 件 (1%)
200	約 1,943 件 (0.5%)

復元抽出を行う際は，時間順序を保存するように抽出を行った．また，実験に使用したデータセットはピーク時には毎秒 100 トランザクション以上であるため，200 分割程度なら復元抽出によるデータの時間的性質は損なわれない．ランダム抽出による性能への影響について，予備実験により影響がほとんどないことを確かめている．

5.2 実験結果

本実験は並列コンピュータではなく，1 台のコンピュータ上で行った．パラレルブースティ

ングによる学習時間の削減を示すために，並列コンピュータで実行した場合の目安として，学習時間・検証時間を以下によって算出する．

(1)　学習時間 = 最長の弱学習器の学習時間 + アンサンブル係数決定の最適化時間

(2)　検証時間 = 最長の弱学習器の検証時間

パラレルブースティングでは，学習中は学習用データ，計算，学習器の評価が完全に独立しているので，他の並列計算に影響を及ぼさない．弱学習器の計算時間は，並列に実行しても順次実行しても個々の学習器の計算時間は変わらないことになる．学習データの分割および学習した弱学習器の統合で同期をとる必要があるため，パラレルブースティングの計算時間は最長の弱学習器の時間を考慮する必要がある．学習中はデータセットが巨大であり，全データをメモリ上に展開できないのでディスクへのアクセスが頻繁に発生する．そのため，ディスクアクセスが並列化によるパフォーマンスに影響を及ぼすため，シェアードナッシング方式の分散環境を想定している．検証時の弱学習器のアンサンブルにかかる時間は 5 秒程度であった．実験結果から，並列計算を行ったときの通信データ量を求めた．

学習時：

各弱学習器の評価結果：　分割なしで 3.67 MB，200 分割で 743 MB

アンサンブル係数：　200 分割で 4 KB

評価時：

各 SVM による予測値：　分割なしで 6.83 MB，200 分割で 41.3 MB

各 SVM による予測値によるアンサンブル SVM の予測値：　6.87 MB

本実験では学習時に全学習データを用いて弱学習器の評価を行っているので，弱学習器の評価結果は分割数に比例したサイズになることが確認できた．評価時はアンサンブル係数が閾値（本実験では 0 と設定）より大きい学習器のみを使用することにした．本実験では，どの分割数の場合も 10 程度の学習器のみでアンサンブル SVM が構築できることが分かったため，評価時の通信データ量は分割なし時の 10 倍程度になることが確認できた．分割なしの学習時間・検証時間がそれぞれ 6 時間ほどであるから，学習・評価時間の比較に通信時間はあまり影響ないと考えられる．

図 5 は，SVM-PBOOST と SKSVM-PBOOST の学習時間を示している．横軸は弱学習器の数であるが，1 というのは 1 つの SVM と SKSVM を表している．これを見ると，弱学習器の数を 10 としたとき，SKSVM-PBOOST は学習時間を大きく改善しており，これは 30，50，100 としたときともそれほど変わらないことが分かる．最も学習時間が少ないのは，弱学習器の数を 30 としたときである．また，並列コンピュータを用いれば弱学習器

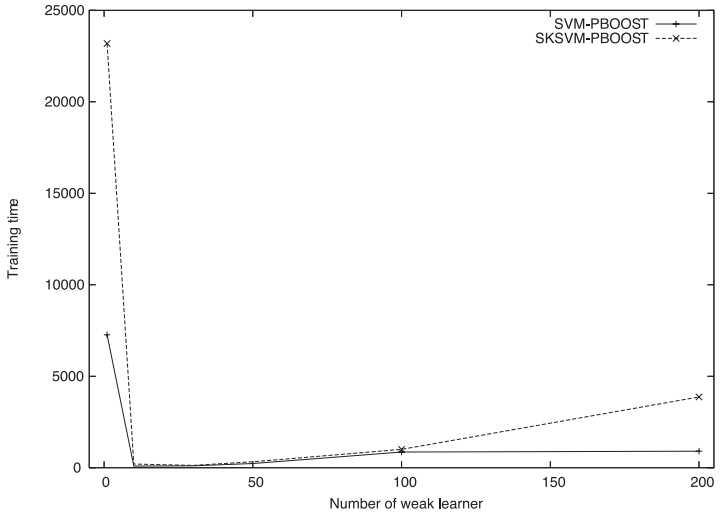

図 5　学習時間の比較

Fig. 5　Comparison of training time.

の数を増やしても学習時間はそれほど変わらない．つまり，各弱学習器が用いる学習データを，全学習データセットの 10 分の 1 程度かそれ以下にすれば，学習時間を大きく短縮できることがいえる．この結果は SKSVM の学習時間（23,186 sec）が，パラレルブースティングにより弱学習器の数を 30 としたとき約 177 分の 1 に縮小していることを示している．弱学習器の数を 200 としたときに，SKSVM-PBOOST が大きく学習時間が悪化しているのは，学習に用いるデータ数が減ったことにより，学習が困難なデータセットが含まれるようになったためであると考えられる．

次に，**図 6** は，SVM-PBOOST と SKSVM-PBOOST の検証時間を示している．横軸は図 5 と同様，弱学習器の数であり，1 は 1 つの SVM と SKSVM を表している．これを見ると，検証においても弱学習器の数を 10 としたとき，SKSVM-PBOOST は検証時間を大きく改善できていることが分かる．最も検証時間が少ないのは，弱学習器の数を 50 としたときである．この結果は SKSVM の検証時間（18,083 sec）が，パラレルブースティングにより約 51 分の 1 に縮小していることを示している．

最後に，**図 7** は SVM-PBOOST と SKSVM-PBOOST の汎化誤差を示している．同様に横軸は弱学習器の数であり，1 は 1 つの SVM と SKSVM を表している．本実験では汎

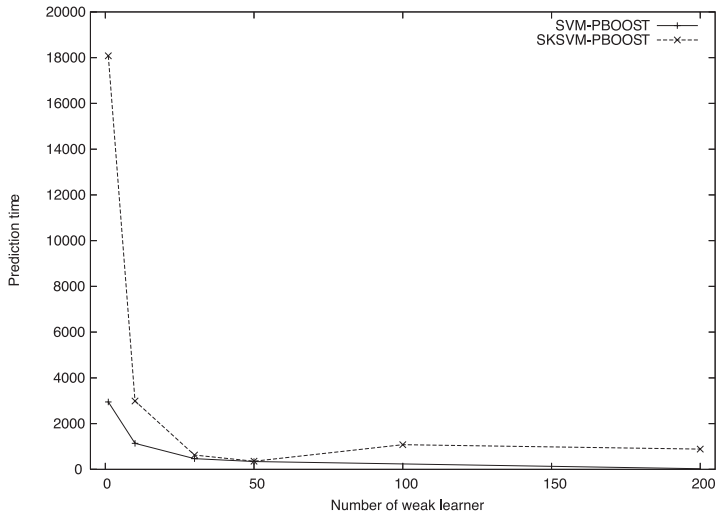

図 6　検証時間の比較
Fig. 6　Comparison of prediction time.

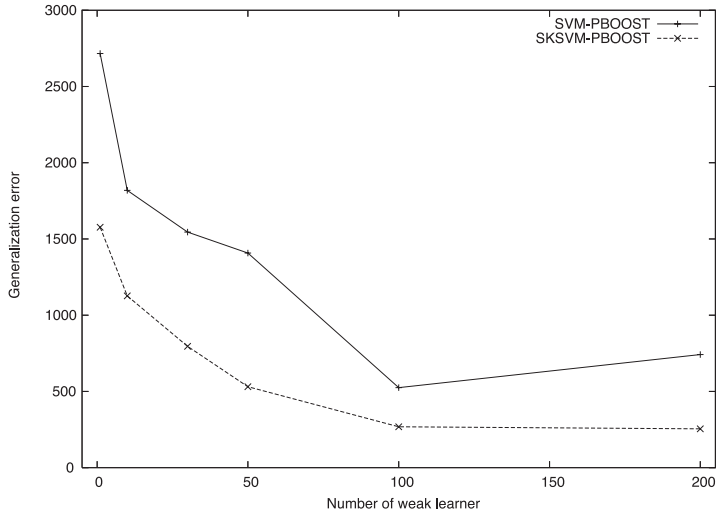

図 7　汎化誤差の比較
Fig. 7　Comparison of generalization error.

化誤差を誤識別の数とする．これを見ると，弱学習器の数を増やし，少数のデータで多くのマシンを用いることによって，SVM-PBOOST と SKSVM-PBOOST の性能が改善されていることが分かる．最も良い性能を示すのは，弱学習器の数を 200 としたときであり，さらに，SVM-PBOOST よりも，SKSVM-PBOOST の方がつねに良い性能である．この結果は SKSVM の汎化誤差（1,577）が，パラレルブースティングにより約 84%削減していることを示している．

以上より，パラレルブースティングにより弱学習器の数を 10 以上にすると学習時間・検証時間を大幅に削減でき，汎化誤差も削減できるという実験結果を得た．

5.3 考　　察

検証に用いたデータは 720,920 件である．そのうち，720,718 件が正常利用，202 件が不正利用である．もし，すべての検証データを正常利用と判断してしまうモデルがあった場合，正識別数は 720,718，誤識別数は 202 件となり，汎化誤差は 202 となる．復元抽出により分割数を増やした場合，1 つの学習器に含まれる不正利用件数がほとんどない状態となってしまい，不正利用を正しく学習できなる可能性が考えられる．つまり，図 7 で示す，弱学習器の数を多くした場合は，このようなまったく説明力のないモデルになってしまっているといえる．おそらく，弱学習器の数を 300，500 と増やせば，汎化誤差はいずれ 202 となると考える．非復元抽出によるサンプリングを行えば，不正利用データがどの学習器にも含まれないということはなくなるが，カーネルマシンを組み合わせる際の係数 c により，不正利用データを含む学習モデルの評価が低く評価されることになり，説明力のないモデルを作成してしまうことが考えられる．本問題は，クレジットカードデータのように 2 クラスの分布が極端に異なる場合に起こる問題であり，2 クラスの分布が近い場合には，データセットを小さくしてもそれほど性能劣化は起こらないものと考えられる．

6. おわりに

本論文では，我々が文献 6) で提案したストリームカーネルマシンが持つ計算時間の問題を，パラレルブースティングによって解決した．約 70 万件の実際のクレジットカードデータを用いた正常利用と不正利用への識別実験の結果，ストリームカーネルを用いた非線形 SVM は，パラレルブースティングによって学習時間を約 177 分の 1（弱学習器の数が 30 の場合），検証時間を約 51 分の 1（弱学習器の数が 50 の場合），誤識別の数を約 84%削減（弱学習器の数が 200 の場合）とする成果を得た．

また，弱学習器が用いる学習データの量を全体の 10%程度にするだけでも，かなりの計

算時間を削減できることが分かった．今後の課題として，1 つの弱学習器が用いる学習データの量を全体の 10%程度にして，弱学習器の数を増やして実験を行うこと，また，様々な実データを用いた実証実験により，ストリームカーネルマシンとそのパラレルブースティングの有効性を検証することがあげられる．

謝辞　本研究の実験で使用したクレジットカードのデータの提供，および実験に対するアドバイスをいただいた伊勢昌幸氏をはじめ，(株）インテリジェントウェイブの関係者の方々に御礼申し上げます．

参 考 文 献

1) 有村博紀：大規模データストリームのためのマイニング技術の動向，電子情報通信学会論文誌 D-I，情報・システム，I-情報処理，Vol.J88-D-I, No.3, pp.563–575 (2005).
2) Wang, H., Fan, W., Yu, P.S. and Han, J.: Mining Concept-Drifting Data Streams Using Ensemble Classifiers, *SIGKDD'03*, pp.226–235, ACM (2003).
3) Scholkopf, B. and Smola, A.J.: *Learning with Kernels*, MIT Press (2002).
4) Shawe-Taylor, J. and Cristianini, N.: *Kernel Methods for Pattern Analysis*, Cambridge University Press (2004).
5) Vapnik, V.N.: *The Nature of Statistical Learning Theory*, Springer, New York, NY, USA (1995).
6) 都築　学，小西　修：大規模データストリームのための履歴情報を用いたカーネル法の拡張，情報処理学会論文：データベース，Vol.1, No.3, pp.27–33 (2008).
7) 都築　学：拡張カーネル法に基づくデータストリームマイニングに関する研究，修士論文，公立はこだて未来大学大学院システム情報科学研究科 (2009).
8) Breiman, L.: Bagging predictors, *Machine Learning*, Vol.24, No.2, pp.123–140 (1996).
9) Friedman, J., Hastie, T. and Tibshirani, R.: Additive Logistic Regression: A Statistical View of Boosting, *The Annals of Statistics*, Vol.28, No.2, pp.337–374 (2000).
10) Freund, Y. and Schapire, R.E.: A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting, *Journal of Computer and System Sciences*, Vol.55, No.1, pp.119–139 (1997).
11) 山名美智子，中原裕之，Massimiliano, P.，甘利俊一：パラレルブースティングによるカーネルマシンを用いたアンサンブル学習，信学技報 NC2002-52，Vol.102, No.381, pp.47–51 (2002).
12) Lozano, F. and Rangel, P.: Algorithms for parallel boosting, *ICMLA '05: Proc. 4th International Conference on Machine Learning and Applications*, Washington, DC, USA, pp.368–373, IEEE Computer Society (2005).
13) Lazarevic, A. and Obradovic, Z.: Boosting Algorithms for Parallel and Distributed Learning, *Distributed and Parallel Databases*, Vol.11, No.2, pp.203–229 (2002).
14) Law, M.H.C., Zhang, N. and Anil, K.J.: Nonlinear Manifold Learning For Data Stream, *Proc. SIAM International Conference for Data Mining*, pp.34–44 (2004).
15) Jain, A., Zhang, Z. and Chang, E.Y.: Adaptive non-linear clustering in data streams, *CIKM '06: Proc. 15th ACM international conference on Information and knowledge management*, New York, NY, USA, pp.122–131, ACM (2006).
16) Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N. and Watkins, C.: Text classification using string kernels, *Journal of Machine Learning Research*, Vol.2, pp.419–444 (2002).
17) Zhang, S., Ning, X.M. and Zhang, X.S.: Graph kernels, hierarchical clustering, and network community structure: Experiments and comparative analysis, *The European Physical Jopurnal B — Condensed Matter and Complex Systems*, Vol.57, No.1, pp.67–74 (2007).
18) Milenova, B.L., Yarmus, J.S. and Campos, M.M.: SVM in Oracle Database 10g: Removing the Barriers to Widespread Adoption of Support Vector Machines, *VLDB'05: Proc. 31st international conference on Very large data bases, VLDB Endowment*, Trondheim, Norway, pp.1152–1163 (2005).
19) Cauwenberghs, G. and Poggio, T.: Incremental and decremental support vector machine learning, *Proc. Advances in Neural Information Processing Systems*, Vancouver, Canada, pp.409–415 (2000).
20) Graf, H.P., Cosatto, E., Bottou, L., Durdanovic, I. and Vapnik, V.: Parallel support vector machines: The cascade svm, *Advances in Neural Information Processing Systems*, pp.521–528, MIT Press (2005).
21) Bottou, L., Chapelle, O., DeCoste, D. and Weston, J.: *Large-Scale Kernel Machines* (*Neural Information Processing*), The MIT Press (2007).
22) Zhang, Y. and Jin, X.: An Automatic Construction and Organization Strategy for Ensemble Learning on Data Streams, *SIGMOD Record*, Vol.35, No.3, pp.28–33 (2006).
23) Cauwenberghs, G. and Poggio, T.: Incremental and Decremental Support Vector Machine Learning, *Advances in neural information processing systems*, Vol.13, pp.409–415 (2001).
24) Laskov, P., Gehl, C., Kruger, S. and Muller, K.-R.: Incremental Support Vector Learning: Analysis, Implementation and Applications, *Journal of Machine Learning Research*, Vol.7, pp.1909–1936 (2006).
25) Haussler, D.: Convolution Kernels on Discrete Structures, Technical report, UC Santa Cruz (1999).
26) Joachims, T.: SVM-Light Support Vector Machine (2006). http://svmlight.joachims.org/

(平成 21 年 6 月 19 日受付)
(平成 21 年 10 月 2 日採録)

(担当編集委員　喜田 拓也)

新美　礼彦（正会員）

公立はこだて未来大学システム情報科学部准教授．1998 年桐蔭横浜大学工学部制御システム工学科 3 年修了．2002 年同大学院博士後期課程制御システム工学専攻修了（博士（工学））．同年公立はこだて未来大学システム情報科学部助手．2009 年より現職．進化計算，データマイニングに関する研究に従事．IEEE-CS，IEEE-SMC，人工知能学会，日本知能情報ファジィ学会各会員．

小西　　修（正会員）

公立はこだて未来大学システム情報科学部教授．1966 年立命館大学理工学部卒業．京都大学工学博士．京都大学工学部，名古屋大学プラズマ研究所，文部省核融合科学研究所，高知大学理学部を経て，2002 年より現職．2008 年より公立大学法人公立はこだて未来大学理事・副学長．主にデータからの学習，創発型情報システムに関する研究に従事．ACM，IEEE-CS，電子情報通信学会，日本データベース学会各会員．