カーネル法を用いた言語解析における高速化手法

工藤　拓[†,☆]　松本裕治[†]

近年，Support Vector Machine を中心とするカーネル法が注目され，自然言語処理においても良い結果を示している．カーネル法により，これまで巧妙に選択する必要があった「組合せ素性」を一般性や計算量を落とすことなく取り入れることができる．しかし，カーネルを用いた場合には，素性の組合せは陰に展開されるため，有効な素性の分析が難しく，さらに，解析時の計算量が大きくなる問題がある．本稿では，頻出部分集合列挙アルゴリズムを用いて，サポートベクターの集合から有効な素性の組合せを発見し，単純な線形分類器に変換することで，解析の速度向上を試みる．英語名詞句同定，日本語わかち書き，および，日本語係り受け解析における実験では，カーネルを用いた通常の解析器に比べ，約 30〜300 倍の速度向上に成功した．

Fast Methods for Kernel-based Text Analysis

Taku Kudo[†,☆] and Yuji Matsumoto[†]

Kernel-based learning (e.g., Support Vector Machines) has been successfully applied to many hard problems in Natural Language Processing (NLP). In NLP, while feature combinations are crucial to improving performance, they have been heuristically selected. Kernel methods change this situation. The merit of the kernel methods is that *effective feature combinations* are implicitly expanded without loss of generality and increase of the computational costs. Kernel-based text analysis methods show an excellent performance in terms of accuracy; however, these methods are usually too slow to apply to large-scale text analysis. In this paper, we extend a *Basket Mining* algorithm to convert a kernel-based classifier into a simple and fast linear classifier. Experimental results on English BaseNP Chunking, Japanese Word Segmentation and Japanese Dependency Parsing show that our new classifiers are about 30 to 300 times faster than the standard kernel-based classifiers.

1. はじめに

近年，学習理論に基づく自然言語処理が進展しており，多くの研究者から注目されている．特に，Support Vector Machine (SVM)[1] は，テキストチャンキング[2]，固有名詞抽出[3),4)]，構文解析[5),6)] 等の自然言語処理タスクに応用され，いずれにおいても高い精度を示している．今後もそのタスクの裾野をひろげていくであろう．

パターン認識における従来の方法論は，与えられた学習データを分類に寄与するであろうできるだけ小さな素性集合で表現し[☆]，その後に，分類誤差が小さくなるような識別関数を構成するものであった．前処理の素性選択こそが，汎化誤差を小さくする鍵となる．一方，SVM では，カーネル関数を使い，超高次元空間上にデータを写像し，そこでの線形分離を考える．このとき，データを表現する素性空間は非常に大きいにもかかわらず，分離平面はマージン最大化により決定されるため，結果的に汎化誤差を小さくすることができる．

この方法論の違いは非常に大きい．従来の機械学習を用いた自然言語処理の多くは，前者の方法論であったため，素性集合の設計が 1 つの腕の見せどころであった．もちろん，基本素性としてどのような素性集合を仮定すればよいかが問題となるが，多くの複雑な自然言語処理タスクでは，1 つの素性のみで決定できることが少ないため，それら基本素性の任意の個数の組合せも同時に考慮しなければならない．たとえば，係り受け解析タスクは，係り元と係り先の基本素性

† 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
☆ 現在，日本電信電話株式会社 NTT コミュニケーション科学基礎研究所
Presently with NTT Communication Science Laboratories, NTT Corporation

☆ 人手で巧妙に素性集合を表現したり，PCA 等の次元圧縮手法を用いて素性集合を選択したりする．

(単語，品詞，活用等）の組合せを新たな素性として追加する必要がある．結果として，候補となる素性集合が膨大になり，この中から有効な素性を慎重に選択する必要があった．

一方，SVM を用いた場合，人手で巧妙に素性選択した場合と比較して，少なくとも同等か場合によってはそれ以上の認識率が得られることが実験的に知られている[4),5)]．また，素性の組合せに関しても，多項式カーネルを用いることで，計算量や一般性を落とすことなく考慮することが可能である．

しかし，SVM に問題点がないわけではない．一番の問題点は，分類速度の遅さである．たとえば，従来のルールベースの固有表現抽出システムに比べ，カーネル法を用いたシステムは，数百倍遅いという報告がある[4)]．カーネル法を用いた言語解析システムは，質問応答，情報検索，テキストマイニングといった大規模テキストデータの解析を必要とする応用には事実上適用困難である．

本稿では，言語解析の多くのタスクに応用されている多項式カーネルに着目し，その高速化を目的とする 2 つの手法，PKI（Polynomial Kernel Inverted），PKE（Polynomial Kernel Expanded）を提案する．PKI では，情報検索の分野で用いられる転置インデックスを用い，分類の高速化を試みる．PKE では，学習後のサポートベクターの集合から分類に寄与する有効な素性の組合せ（一般には事例の中の部分構造）を抽出し，単純な線形分類器に変換することで高速化を実現する．英語名詞句同定，日本語わかち書き，および日本語係り受け解析における実験では，従来の分類器に比べ，PKI で約 3〜12 倍，PKE で約 30〜300 倍の速度向上に成功した．

2. カーネル法と Support Vector Machine

正例，負例の 2 つのクラスに属する学習データのベクトル集合を，$\{\langle \mathbf{x}_1, y_1\rangle, \ldots, \langle \mathbf{x}_L, y_L\rangle\}$ ($\mathbf{x}_i \in \mathbb{R}^M$, $y_i \in \{+1, -1\}$) とする．SVM では，二値分類を仮定しているために，y_i は，クラスに応じて $+1$ もしくは，-1 をとるものとする．このとき，SVM の分離関数は，次式で与えられる．

$$y = \mathrm{sgn}\Big(\sum_{j \in SV} y_j \alpha_j \phi(\mathbf{x}_j) \cdot \phi(\mathbf{x}) + b\Big) \qquad (1)$$

ただし，

(1) ϕ は，$\mathbb{R}^M$ から $\mathbb{R}^H$（一般に $M \ll H$）への写像関数，

(2) $\alpha_i, b \in \mathbb{R},\ 0 \le \alpha_i \le C$,

(3) $SV = \{j | j = 1, \ldots, L\ \ \alpha_j \neq 0\}$

（サポートベクターの事例番号の集合）

式 (1) は，テストデータを高次元空間に写像したデータ $\phi(\mathbf{x})$ と学習データを写像したデータ $\phi(\mathbf{x}_j)$ との内積を全サポートベクター（SV）について計算し，重み $y_i\alpha_j$ で線形結合した形となっている．

ここで，高次元空間への写像関数 $\phi(\cdot)$ は，すべての事例が，$\mathbb{R}^H$ における超平面で分離できるようにユーザが設計する．通常，H は M よりもはるかに高い次元になるために計算量が大きくなる問題がある．この問題を解決する手段としてカーネル関数がある．

式 (1) より，実際に SVM を構成する場合には，写像関数は $\phi(\mathbf{x}_j) \cdot \phi(\mathbf{x})$ という内積の形で出現する．本稿では，α_j の最適化手法について言及しないが，SVM における学習の式も，写像関数の内積のみで表現できる．したがって，高次元空間 $\mathbb{R}^H$ 上での内積 $\phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2)$ が効率良く計算できれば，$\mathbb{R}^H$ が高次元空間であることの計算量的問題は解決できる．このように，写像された高次元空間上での内積を表現するための関数 $K(\mathbf{x}_1, \mathbf{x}_2) = \phi(\mathbf{x}_1) \cdot \phi(\mathbf{x}_2)$ をカーネル関数と呼ぶ．カーネル関数を用いると，式 (1) は以下のように書き直すことができる．

$$y = \mathrm{sgn}\Big(\sum_{j \in SV} y_j \alpha_j K(\mathbf{x}_j, \mathbf{x}) + b\Big) \qquad (2)$$

式 (2) より，SVM の学習，分類には，陽に表現された素性の集合は必ずしも必要ではなく，事例間の類似度（内積）を与えるカーネル関数のみ定義できればよいことが分かる☆．この「分類したいデータ構造に関する情報を，カーネル関数という形で学習アルゴリズムとは独立に設計できる」という性質こそが，SVM を中心とするカーネル法が急激に脚光を浴びることとなった 1 つの要因と考えられる．

3. d 次の多項式カーネル

ここで，分類したい事例 $\mathbf{x}$ が**集合**として表現される場合を考える．自然言語解析においては，このように事例を集合で表現することが多い．これは，言語データそのものが離散データあることに起因する．

まず，素性集合 $F = \{1, 2, \ldots, N\}$ を与える．すべての事例 $X_j (j = \{1, 2, \ldots, L\})$ は，F の部分集合とする ($X_j \subseteq F$)．このとき，集合 X_j は，2 値ベクトル $\mathbf{x}_j = (x_{j1}, x_{j2}, \ldots, x_{jN})$ （ただし $x_{ji} = I(i \in X_j)$，$I(\cdot)$ は Indicator function）の素性空間として表現できる．$\mathbf{x}_1$ と $\mathbf{x}_2$ の内積は，$\mathbf{x}_1 \cdot \mathbf{x}_2 = |X_1 \cap X_2|$ と

☆ 実際にはカーネルとして満たすべき制約（Mercer's condition）が存在する．

なる．

定義 1 d 次の多項式カーネル
2つの集合 X，Y（それぞれは，2値ベクトル $\mathbf{x}$，$\mathbf{y}$ に対応する）に対し，d 次の多項式カーネル $K_d(X, Y)$ は，以下で与えられる．

$$K_d(X, Y) = (1 + |X \cap Y|)^d, \quad d = 1, 2, \ldots \quad (3)$$

本稿では，式 (3) を**陰表現**による多項式カーネルと呼ぶこととする．

言語解析では，精度向上のために素性の組合せ（一般には素性集合 F の部分集合）を新たな素性として追加することが多い．従来研究では，このような組合せは人手によって発見的に与えられることが多かった．一方，d 次の多項式カーネルは，元の空間 F を F^d の空間に写像するような関数 $\phi(\cdot)$ を持つことが知られている．つまり，多項式カーネルにより，素性の d 個の組合せを一般性を落とすことなく考慮することができる．この性質は，言語解析において特に重要であり，多項式カーネルの有効性が種々のタスクにおいて示されている[2),4)~6)]．

ここで，以下のような**陽表現**による多項式カーネルを与え，さきに述べた多項式カーネルの性質について検証する．

補題 1 陽表現による多項式カーネル
d 次の多項式カーネルは，以下のように変形できる

$$K_d(X, Y) = \sum_{r=0}^{d} c_d(r) \cdot |P_r(X \cap Y)| \quad (4)$$

ただし

- $P_r(X)$ は X の部分集合（ただしサイズ r）の集合．
- $c_d(r) = \sum_{l=r}^{d} \binom{d}{l} \left(\sum_{m=1}^{r} (-1)^{r-m} \cdot m^l \binom{r}{m} \right)$

証明 付録 A.

$c_d(r)$ を，d 次の多項式カーネルの**部分構造重み**と呼ぶ．この重みは，サイズが r である部分集合 $s(|s| = r)$ の事前重みと解釈できる．式 (4) から，d 次の多項式カーネルでは，X，Y それぞれの集合の d 個までの素性の組合せが新たな素性として追加されることが分かる．以下に具体例を示す．

例 1 2 次および 3 次の多項式カーネル
2つの集合 $X = \{a, b, c, d\}$ および $Y = \{a, b, d, e\}$ に対し，2次の多項式カーネル $K_2(X, Y)$ および3次の多項式カーネル $K_3(X, Y)$ は，陰表現を用いると以下のように計算できる．$K_2(X, Y) = (1 + |X \cap Y|)^2 = (1 + 3)^2 = 16$，$K_3(X, Y) = (1 + |X \cap Y|)^3 = (1 + 3)^3 = 64$.

補題 1 により，部分構造重みは次のようになる．$c_2(0) = 1$，$c_2(1) = 3$，$c_2(2) = 2$，$c_3(0) = 1$，$c_3(1) = 7$，$c_3(2) = 12$，$c_3(3) = 6$．さらに，部分集合 $P_r(X \cap Y)$ $(r = 0, 1, 2, 3)$ は，$P_0(X \cap Y) = \{\phi\}$，$P_1(X \cap Y) = \{\{a\}, \{b\}, \{d\}\}$，$P_2(X \cap Y) = \{\{a, b\}, \{a, d\}, \{b, d\}\}$，$P_3(X \cap Y) = \{\{a, b, d\}\}$ となる．以上より，$K_2(X, Y)$，$K_3(X, Y)$ は陽表現を用いると以下のように計算できる．$K_2(X, Y) = 1 \cdot 1 + 3 \cdot 3 + 2 \cdot 3 = 16$，$K_3(X, Y) = 1 \cdot 1 + 7 \cdot 3 + 12 \cdot 3 + 6 \cdot 1 = 64$.

```
function PKI_classify(X)
    r = 0 # 配列，値を 0 に初期化
    foreach i ∈ X
        foreach j ∈ h(i)
            r_j = r_j + 1
        end
    end
    result = b #戻り値，実数
    foreach j ∈ (1 .. L)
        result += y_j α_j · (1 + r_j)^d
    end
    return sgn(result)
end
```

図 1 PKI の擬似コード
Fig. 1 Pseudo code for PKI.

4. 多項式カーネルの高速化

本章では，次式で与えられる分類関数の高速化手法を述べる．

$$y = \mathrm{sgn}\Big(\sum_{j \in SV} y_j \alpha_j \cdot (1 + |X_j \cap X|)^d + b \Big). \quad (5)$$

便宜的に本稿では，式 (5) の分類手法を **PKB**（Polynomial Kernel Baseline）と呼ぶ．PKB の分類計算量は，事例 X とサポートベクター X_j との共通要素を列挙するのに $O(|X|)$，さらにサポートベクターが $|SV|$ 個あるため，全体として $O(|X| \cdot |SV|)$ となる．

4.1 PKI（Inverted Representation）

X 中の各要素 $i \in X$ が，どのサポートベクターに出現するかが事前に分かっていれば，共通要素の数を全事例について列挙する必要はなくなる．これは，情報検索の転置インデックスと同じ考えである．**図 1** にアルゴリズム PKI の擬似コードを示す．$h(i)$ は要素（素性）i を含むサポートベクターを返す関数（テーブル）である．

PKI は近似手法ではなく，厳密解を導出する手法である．また，PKI の計算量は，$O(|X| \cdot B + |SV|)$ である（ただし，B は全素性 $i \in F$ に対する $|h(i)|$ の

平均値）．PKI は B が小さいとき，すなわち素性空間が疎の場合に特に高速となる．

4.2 PKE（Expanded Representation）

4.2.1 基本的なアイデア

補題 1 より，分類関数 (5) は，以下のように陽表現で書くことができる．

$$y = \mathrm{sgn}\Big(\sum_{j \in SV} y_j \alpha_j \big(\sum_{r=0}^{d} c_d(r) \cdot |P_r(X_j \cap X)|\big) + b\Big). \tag{6}$$

もし全部分集合 $s \in \bigcup_{r=0}^{d} P_r(F)$ に対し，

$$w(s) = \sum_{j \in SV} y_j \alpha_j c_d(|s|) I(s \subseteq X_j) \tag{7}$$

を事前に計算しておけば（ただし $I(\cdot)$ は Indicator function），式 (6) は以下のような単純な線形分類器となる．

$$y = \mathrm{sgn}\Big(\sum_{s \in \Gamma_d(X)} w(s) + b\Big). \tag{8}$$

ただし $\Gamma_d(X) = \bigcup_{r=0}^{d} P_r(X)$．

式 (8) による分類アルゴリズムを，本稿では PKE（Polynomial Kernel Expanded）と呼ぶ．PKE の計算量は $O(|\Gamma_d(X)|) = O(|X|^d)$ となり，サポートベクター数 $|SV|$ に依存しない．

4.2.2 PKE の実装

PKE を適用するには，まず，ベクトル

$$\mathbf{w} = (w(s_1), w(s_2), \ldots, w(s_{|\Gamma_d(F)|})).$$

を事前に算出する必要がある．2 次の多項式カーネルのように，サイズ 2 までの部分集合のみを列挙すればよいときは，式 (7) を用い直接算出できる．これは，磯崎らが 2 次の多項式カーネルに限り，素性中のすべての 2 つまでの組合せを全展開した手法と本質的に同一である[4]．しかし，3 次や 4 次の多項式カーネルの場合，組合せの数が指数的に増え列挙が困難である．そこで，以下のような近似解 $\mathbf{w}'$ を求め代用する．

定義 2　$\mathbf{w}$ の近似解 $\mathbf{w}'$

$\mathbf{w} = (w(s_1), w(s_2), \ldots, w(s_{|\Gamma_d(F)|}))$ について，$\sigma_{neg} < w(s) < \sigma_{pos}$ ($\sigma_{neg} < 0$, $\sigma_{pos} > 0$) となる場合は，$w(s) = 0$ に近似する．結果，作成されたベクトルを $\mathbf{w}$ の近似解 $\mathbf{w}'$ と定義する．

ここで，$w(s) \geq \sigma_{pos}$ または $w(s) \leq \sigma_{neg}$ となる部分集合 s を**有効部分集合**と呼ぶこととする．$\mathbf{w}'$ は，0 付近の微少な範囲 $(\sigma_{neg}, \sigma_{pos})$ 内にある重みを持つ部分集合は分類に寄与しないという仮定のもとでの近似解である．$\mathbf{w}'$ の導出は，次に述べる部分集合の列挙問題と本質的に同一である．

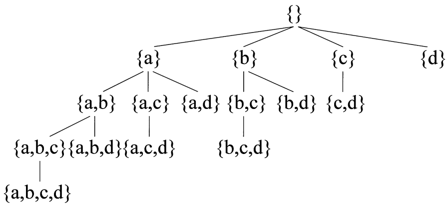


図 2　集合 {a,b,c,d} に対する SE-Tree
Fig. 2　SE-Tree for a set {a,b,c,d}.

問題 1　有効部分集合の列挙問題

サポートベクターの集合 $\{\langle(y_j, \alpha_j, \mathbf{x}_j\rangle | j \in SV\}$，部分集合重み $\{c_d(0), c_d(1), \ldots, c_d(d)\}$ が与えられたときに，$w(s) \geq \sigma_{pos}$ または $w(s) \leq \sigma_{neg}$ となるような部分集合 s と，その重み $w(s)$ を網羅的に求めよ．

有効部分集合の列挙問題は，**頻出部分集合の列挙問題（バスケットマイニング）**の変形と見なすことができる．頻出部分集合の列挙問題では，ある閾値以上出現する部分集合 s が列挙対象となる．これまでに，Apriori[7] をはじめとする種々のアルゴリズムが頻出部分集合の列挙問題を効率良く解くために提案されている．これらのアルゴリズムは，以下の 2 つの戦略に従い頻出部分集合のみを効率良く列挙する．

(1) 部分集合を枚挙する正準的な探索木，もしくは探索ラティスを定義する．
(2) 探索効率向上のため，(1) で与えられる空間を枝刈りする．

(1) にはいくつかの方法があるが，本稿では，効率の良さ，実装のしやすさから Bayardo による Set Enumeration Tree (SE-Tree)[8] を用いる．**図 2** に SE-Tree の例を示す．SE-Tree では，集合の要素に順序（たとえば辞書順）を与え，集合の枚挙の順番をその要素順に限定することで，正準的な探索木を構築する．(2) の枝刈りでは，「ある部分集合が頻出でなければ，その上位集合も頻出ではない」という性質を用いる．つまり，ある部分集合が頻出でないと分かると，その上位集合に対する探索は行わなくてよい．

有効部分集合の列挙も，上記の (1)，(2) の戦略を用いて実現できる．(1) の列挙法はほぼ同じであるが，d 次の多項式カーネルを用いる場合は，d 個のまでの部分集合を列挙するだけで十分である．(2) の枝刈り条件に関しては，集合 s の頻度ではなく，$w(s)$ の大きさが条件となるため，頻出部分集合列挙アルゴリズムの枝刈り条件をそのままの形で使うことはできない．ここで，有効部分集合列挙アルゴリズムの枝刈り条件を構築するにあたっての重要な性質を以下の補題 2 に示す．

補題 2 $w(s)$ の上限，下限値
s の全上位集合 s' について（$\forall s', s' \supseteq s$ について），$\mu_{neg}(s) \leq w(s') \leq \mu_{pos}(s)$ が成立する．ただし $\mu_{pos}(s)$, $\mu_{neg}(s)$ は以下で与えられる．

$$\mu_{pos}(s) \stackrel{\mathrm{def}}{=} \sum_{\{j|j \in SV, y_j=+1\}} \alpha_j C_d I(s \subseteq X_j)$$

$$\mu_{neg}(s) \stackrel{\mathrm{def}}{=} -\sum_{\{j|j \in SV, y_j=-1\}} \alpha_j C_d I(s \subseteq X_j)$$

$$C_d \stackrel{\mathrm{def}}{=} \max_{r=0,\ldots,d} c_d(r)$$

証明

$$\begin{aligned} w(s') = & \sum_{\{j|j \in SV, y_j=+1\}} \alpha_j c_d(|s'|) I(s' \subseteq X_j) - \\ & \sum_{\{j|j \in SV, y_j=-1\}} \alpha_j c_d(|s'|) I(s' \subseteq X_j) \\ \leq & \sum_{\{j|j \in SV, y_j=+1\}} \alpha_j c_d(|s'|) I(s' \subseteq X_j) \\ \leq & \sum_{\{j|j \in SV, y_j=+1\}} \alpha_j C_d I(s' \subseteq X_j) \\ \leq & \sum_{\{j|j \in SV, y_j=+1\}} \alpha_j C_d I(s \subseteq X_j) \\ = & \mu_{pos}(s) \ \ (\geq 0) \end{aligned}$$

（$\because$ $\alpha_j \geq 0, C_d \geq 0$, $\forall s', s' \supseteq s$ について $|\{i|y_j = +1, s' \subseteq X_j\}| \leq |\{j|y_j = +1, s \subseteq X_j\}|$ が成立するため）．同様に，

$$\begin{aligned} w(s') \geq & -\sum_{\{j|j \in SV, y_j=-1\}} \alpha_j C_d I(s \subseteq X_j) \\ = & \mu_{neg}(s) \ \ (\leq 0) \end{aligned}$$

以上より，s の全上位集合 s' について（$\forall s', s' \supseteq s$ について），$\mu_{neg}(s) \leq w(s') \leq \mu_{pos}(s)$ □

補題 2 より，$\mu_{pos}(s) < \sigma_{pos}$ かつ $\mu_{neg}(s) > \sigma_{neg}$ のとき，s の全上位集合 s' ($\forall s', s' \supseteq s$) は有効部分集合になりえない．つまり，有効部分集合の列挙は，頻出部分集合列挙アルゴリズムの枝刈り条件を $\mu_{pos}(s) < \sigma_{pos}$ かつ $\mu_{neg}(s) > \sigma_{neg}$ に変更することで実現できる．

4.2.3 σ_{pos}，σ_{neg} の設定方法

σ_{pos} および σ_{neg} は，どれくらいの近似精度を要求するかを決定する変数である．一般に，正例と負例の事例数には偏りがある．その偏りを考慮し，ユーザは閾値 $\sigma(>0)$ のみを与え，それを以下のように正例と負例の数で線形分配して σ_{pos}，σ_{neg} を決定する．

$$\sigma_{pos} = \ \ \sigma \cdot |\{j|j \in SV, y_j = +1\}|/|SV|$$
$$\sigma_{neg} = -\sigma \cdot |\{j|j \in SV, y_j = -1\}|/|SV|$$

解析速度と解析精度は一般にトレードオフの関係にあり，近似パラメータ σ によってどちらを重視するか調整することができる．適切なパラメータは個々のアプリケーション，計算機リソース，ならびに実行環境に強く依存する．しかし，一般には以下の方法でパラメータを選択可能である．

まず，大量の未解析テキストを近似なしのモデルで解析する．その結果を擬似的な「正解」と見なし，近似ありのモデルの精度を算出する．近似パラメータを変化させながら，2 つの精度の差が十分小さくなる（もしくは同一になる）ようにパラメータを選択する．すなわち，一定の精度の差を認めたうえで，最速の速度が達成できるようパラメータを設定する．

逆に速度を基準にパラメータを調整することも可能である．タスクの要求速度が与えられたときに，大量の未知データを解析しながら，その要求速度を満たす範囲でパラメータをできる限り小さく設定する．

これら 2 つの設定方法は，未知テキストを用いることから，容易に実現可能である．

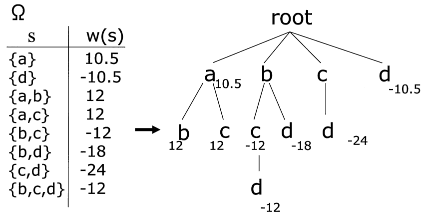


図 3 Ω の TRIE による表現
Fig. 3 Ω in TRIE representation.

4.2.4 TRIE による部分集合の格納

近似ベクトル $\mathbf{w}'$ は，集合 s と，それに対応する重み $w(s)$ のタプルの集合 $\Omega = \{\langle s, w(s) \rangle\}$ として表現できる．集合 s のそれぞれは，多くの共通した部分構造を持つため，それを単純なハッシュ等を使って格納するのは記憶効率が悪い．

その問題を解決するために，共通部分を共有する TRIE を作成する．具体的には，部分集合の要素を辞書順にソートし，要素の左から順に TRIE に格納する．TRIE の各ノードには，root からノードまでの集合に対応する重み $w(s)$ が格納される．**図 3** に，TRIE を用いた集合 Ω の格納例を示す．

事例（集合）X を分類するときは，X の部分集合を SE-Tree を使って列挙しながら，TRIE のノードを探索していけばよい．TRIE の実装として，Double-Array [9] を用いた．

5. 実　　験

PKI，PKE の有効性を示すために，英語名詞句同

表 1 実験データの詳細
Table 1 Details of data set.

データセット	EBC	JWS	JDP
事例数	135,692	265,413	110,355
|SV| SV 数	11,690	57,672	34,996
SV の正例数	5,637	28,440	17,528
SV の負例数	6,053	29,232	17,468
$\|F\|$（素性集合のサイズ）	17,470	11,643	28,157
$\|X_j\|$ の平均	11.90	11.73	17.63
B（$\|h(i)\|$ の平均）	7.74	58.13	21.92
d（カーネルの次元）	2	3	3
素性候補数の上限値	約 270 万	約 1600 万	約 3800 万

表 2 実験結果（英語名詞句同定 EBC）
Table 2 Results: English Base-phrase Chunking, EBC.

PKE σ	解析時間 (秒/文)	速度改善率	F 値	$\|\Omega\|$ ($\times$ 1000)
0.1	0.0010	163.4	92.98	43
0.05	0.0013	127.8	93.84	141
0.01	0.0016	105.2	93.79	518
0.005	0.0016	101.3	93.85	668
0.001	0.0017	97.7	93.84	858
0.0005	0.0017	96.8	93.84	889
PKI	0.020	8.3	93.84	
PKB	0.164	1.0	93.84	

表 3 実験結果（日本語わかち書き JWS）
Table 3 Results: Japanese Word Segmentation, JWS.

PKE σ	解析時間 (秒/文)	速度改善率	精度 (%)	$\|\Omega\|$ ($\times$ 1000)
0.1	0.0007	1290.6	96.09	21
0.05	0.0010	846.7	97.36	84
0.01	0.0024	358.2	97.93	1,228
0.005	0.0028	300.1	97.95	2,327
0.001	0.0034	242.6	97.94	4,392
0.0005	0.0035	238.8	97.94	4,820
PKI	0.4989	1.7	97.94	
PKB	0.8535	1.0	97.94	

定（EBC），日本語わかち書き（JWS），日本語係り受け解析（JDP）の 3 つの言語解析タスクで実験を行った．これらのタスクの詳細（実験データ，条件，素性等）は，次章で述べる．**表 1** に，実験データの詳細情報（サポートベクター数，素性サイズ）をまとめる．ただし，素性候補数の上限値とは各事例の d 個までの組合せを全展開して素性を構築した場合に素性の異なり数がとりうる上限値である．換言すると，この数に比例する空間（メモリ）があれば素性の全展開が可能となる．

EBC では 2 次の多項式カーネルを，JWS，JDP では 3 次の多項式カーネルを用いた．これらのカーネルは，予備実験の結果から最適な次数を選択した．3 次の多項式カーネルが作る素性空間は，陽に展開すると非常に大きくなる．そのため，頻出部分集合列挙アルゴリズムを用いる提案手法の有効性を検証するのに適切なタスクだと考える．

全実験は，XEON 2.4 Ghz，主記憶 3.5 Gbyte の Linux 上で行った．全システムは，C++で実装した．

5.1 英語名詞句同定（EBC）

名詞句同定タスクとは，入れ子がない最小の名詞句を同定するタスクである．名詞句同定タスクは，名詞句を，**B**：名詞句の先頭，**I**：先頭以外の名詞句，**O**：名詞句の外，の 3 つのタグを付与する問題として扱われることが多い．本稿では，工藤ら[2) と同一の設定（素性，解析手法）で実験を行った．実験に用いたコーパスは Penn Tree Bank の一部で，学習にはセクション 15〜19 を，テストにはセクション 20 を用いた．これは，名詞句同定タスクに用いられる標準データ[10) と同一である．

5.2 日本語のわかち書きタスク（JWS）

明示的な単語境界がない日本語では，単語の区切り位置の同定（わかち書き）はきわめて重要なタスクである．ここでは，辞書等の存在を仮定しないわかち書きモデルを考える．詳細は，文献 11)，12) に譲るが，基本的に，ある固定長のウインドウの中にある文字，文字種を素性とし，学習，分類を行う．実験に用いたコーパスは京都大学テキストコーパス（Version 2.0）[13) の一部で，学習には新聞記事の 1 月 1 日から 8 日分（7,958 文），テストには 1 月 9 日分の（1,246 文）を用いた．

5.3 日本語係り受け解析（JDP）

文節単位の係り受け解析タスクである．採用した係り受け解析モデルは，Cascaded Chunking Model[5) で，直後に係るか係らないかのみで動作し，その判定に SVM を用いている．解析モデル，用いる素性等は，文献 5) を参照されたい．実験に用いたデータは，JWS と同一である．

5.4 結　　果

表 2，**表 3**，**表 4** に，PKI および PKE において σ を 0.1 から 0.005 まで変化させたときの，解析時間☆，PKB に対する速度の改善率，解析精度☆☆，有効部分集合のサイズ $|\Omega|$ をまとめる．

PKI は，EBC で約 8 倍，JWS で約 1.7 倍，JDP で約 12 倍の速度向上が確認された．PKI は，一般に素性空間が疎のときに高速になる．JDP では，B（$|h(i)|$

☆ 実際の応用を考え，1 文あたりの解析時間とした．
☆☆ EBC のみ，解析精度は F 値で行った．

表 4　実験結果（日本語係り受け JDP）
Table 4　Results: Japanese Dependency Parsing, JDP.

PKE σ	解析時間 (秒/文)	速度 改善率	精度 (%)	$\|\Omega\|$ ($\times$ 1000)
0.1	0.0008	338.2	82.02	7
0.05	0.0014	200.0	86.27	30
0.01	0.0042	66.8	88.91	73
0.005	0.0060	47.8	89.05	1,924
0.001	0.0086	33.3	89.26	6,686
0.0005	0.0090	31.8	89.29	8,262
PKI	0.0226	12.6	89.29	
PKB	0.2848	1.0	89.29	

の平均値）が小さく，素性空間が疎となっているため，他のタスクに比べ速度向上率が高い．逆に，JWS は B の値がほかに比べ大きく，素性空間が密となっているため，速度向上率が低い．これは，JWS で用いられる素性が「文字」であり，「単語」を使う他の手法に比べ素性の種類数が小さくなる傾向になるためだと考えられる．

PKE の速度向上率は，PKI のそれらに比べ顕著に高い．PKB と同程度の精度が得られるよう σ を十分小さく設定した場合，EBC で約 100 倍（$\sigma = 0.005$），JWP で約 300 倍（$\sigma = 0.005$），JDP で約 30 倍（$\sigma = 0.0005$）の速度向上が確認された．

5.5　頻度によるフィルタリング

3 次の多項式カーネルを用いた場合，$|\Omega|$ が非常に大きくなる傾向にある（JWS で約 2 百万，JDP で約 8 百万）．一般に，$|\Omega|$ が小さくなれば，TRIE を探索する空間が減少するために，速度向上につながる．そこで，不必要な部分集合を削除するために，サポートベクターの集合から，Ω 中の各部分集合の出現頻度を計算し，高頻度の部分集合のみを残し，残りを削除するという単純なフィルタリング実験を行った．このようなフィルタリングは，頻出部分集合列挙アルゴリズム（バスケットマイニングアルゴリズム）を用いて，効率良く実現できる．JWS では $\sigma = 0.005$，JDP では $\sigma = 0.0005$ に固定し，頻度閾値 $\xi(=\{1,2,3\})$ 回以上の有効部分集合のみを残した実験結果を**表 5**，**表 6** に示す．

結果，JDP において $\xi = 2$ とした場合，TRIE のサイズを約 1/3 にできた．さらに，速度向上（0.0097 秒/文 → 0.0074 秒/文）も確認できた．また，有意差はないが，若干の精度向上（89.29%→89.34%）が確認できた．これは，低頻度にもかかわらず，高い重みを持つ部分構造が削除されたからと考える．このような部分集合は，学習データの誤りや特殊な事例と考えられ，過学習の原因になりやすい．

表 5　頻度によるフィルタリング（日本語わかち書き JWS）
Table 5　Frequency-based Pruning (JWS).

PKE ξ	解析時間 (秒/文)	速度 改善率	精度 (%)	$\|\Omega\|$ ($\times$ 1000)
1	0.0028	300.1	97.95	2,327
2	0.0025	337.3	97.83	954
3	0.0023	367.0	97.83	591
PKB	0.8535	1.0	97.94	

表 6　頻度によるフィルタリング（日本語係り受け JDP）
Table 6　Frequency-based Pruning (JDP).

PKE ξ	解析時間 (秒/文)	速度 改善率	精度 (%)	$\|\Omega\|$ ($\times$ 1000)
1	0.0090	31.8	89.29	8,262
2	0.0072	39.3	89.34	2,450
3	0.0068	41.8	89.31	1,360
PKB	0.2848	1.0	89.29	

一方，JWS では $\xi = 2$ の場合，TRIE のサイズを約半分にできたが，解析精度の低下（97.94%→97.83%）が確認された．これは頻度 1 の素性も分類に寄与していることを示唆している．

5.6　マイニングアルゴリズムの効果

素性の全展開に基づく単純な手法を適用した場合，素性の候補数の上限値は，EBC，JWS，JDP でそれぞれ約 270 万，1600 万，3800 万であった（表 1 最下行）．仮に 1 つの素性とその重みを表現するのに 20 バイト必要だとした場合，全展開に必要なメモリは，それぞれ 54 MB，320 MB，760 MB となる．これらは現在の計算機で計算可能とはいえ，非常に大きい．

一方，本手法は深さ優先探索に基づき素性を列挙しているために，基本的に全サポートベクターがメモリ上に保持できればよく，必要なメモリは全展開に基づく手法に比べ小さい．実際に本手法で用いられたメモリは，3 つのタスクいずれにおいてもたかだか数十 MB であった．

この性質は，カーネルの次元数が増えても変わらない．たとえば，4 次の多項式カーネルを用いた場合，JDP に必要なメモリは 3.5 GB 程度になり，全展開は困難である．

さらに，サポートベクターの数に対しても本手法は頑健である．JDP についてコーパスの全データ（約 4 万文）を用いてモデルを構築した場合，必要なメモリは 3 GB 程度になり，この場合も全展開は困難となる．

データマイニングアルゴリズムの適用により，空間的使用量を大幅に削減でき，カーネルの次元，および事例（サポートベクター）数に対して頑健なアルゴリズムが構築できたといえる．

5.7 関連研究との比較

SVM の分類を高速にするいくつかの手法がこれまで提案されている．

磯崎らは，固有表現抽出タスクの高速化のために，XQK（eXpand the Quadratic Kernel）という手法を提案している[4]．XQK と PKE は，基礎となるアイデアは本質的に同一である．ともに素性空間を陽に展開し，単純な線形分類器に変換することで高速化を試みる．

違いは，手法の一般性である．XQK は，2 次の多項式カーネルの高速化だけを目的としているが，PKE は，2 次の多項式カーネルだけでなく，一般の多項式カーネル $(1+|X \cap Y|)^d$ に適用可能である☆．PKE は，XQK より一般的な手法といえる．

XQK が，2 次の多項式カーネルに限定されている理由は，XQK が単純な完全列挙を用いて，素性空間を構築している点にある．このような手法は，2 次の多項式カーネル（2 つの組合せ素性）には適用可能だが，規模耐性がなく，効率が悪いため，一般の多項式カーネルには適用困難である．PKE では，頻出部分集合列挙アルゴリズムを拡張することで，この問題点を改善し，一般の多項式カーネルに適用範囲をひろげた．実際に，2 次だけでなく，3 次の多項式カーネルを用いた実験でも，PKE の有効性を示すことができた．

6. おわりに

本稿では，言語解析に多く用いられる d 次の多項式カーネルに着目し，2 つの高速な分類アルゴリズムを提案した．1 つは，転置インデックスの拡張である PKI，もう 1 つは，素性が陽に展開し，単純な線形分類器に変換する PKE である．

英語名詞句同定，日本語わかち書き，および，日本語係り受け解析における実験では，従来の分類器に比べ，PKI で約 3〜12 倍，PKE で約 30〜300 倍の速度向上に成功した．

今後は，他の離散構造 Kernel（Tree Kernel[14]，String Kernel[15),16]）について，部分木や部分系列のマイニングアルゴリズムを用いた高速分類手法を提案したいと考える．

参考文献

1) Vapnik, V.N.: *The Nature of Statistical Learning Theory*, Springer (1995).

2) 工藤　拓，松本裕治：Support Vector Machine を用いた Chunk 同定，自然言語処理，Vol.9, No.5, pp.3–21 (2002).

3) 山田寛康，工藤　拓，松本裕治：Support Vector Machine を用いた日本語固有表現抽出，情報処理学会論文誌，Vol.43, No.1, pp.44–53 (2002).

4) 磯崎秀樹，賀沢秀人：固有表現抽出のための SVM の高速化，情報処理学会論文誌，Vol.44, No.3, pp.970–979 (2003).

5) 工藤　拓，松本裕治：チャンキングの段階適用による日本語係り受け解析，情報処理学会論文誌，Vol.43, No.6, pp.1834–1842 (2002).

6) Yamada, H. and Matsumoto, Y.: Statistical Dependency Analysis with Support Vector Machines, *IWPT 2003: 8th International Workshop on Parsing Technologies*, pp.195–206 (2003).

7) Agrawal, R. and Srikant, R.: Fast Algorithms for Mining Association Rules, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB* (1994).

8) Bayardo, R.J.: Efficiently Mining Long Patterns from Databases, *Proc. of SIGMOD 1998* (1998).

9) Aoe, J.: An Efficient Digital Search Algorithm by Using a Double-Array Structure, *IEEE Trans. Softw. Eng.*, Vol.15, No.9 (1989).

10) Ramshaw, L.A. and Marcus, M.P.: Text Chunking using Transformation-Based Learning, *Proc. VLC*, pp.88–94 (1995).

11) Sassano, M.: An Empirical Study of Active Learning with Support Vector Machines for Japanese Word Segmentation, *Proc. ACL*, pp.505–512 (2002).

12) 新納弘幸：決定リストを弱学習器としたアダブーストによる日本語単語分割，自然言語処理，Vol.8, No.2 (2001).

13) Kurohashi, S. and Nagao, M.: Kyoto University text corpus project, *Proc. ANLP*, pp.115–118 (1997).

14) Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N. and Watkins, C.: Text Classification using String Kernels, *Journal of Machine Learning Research*, Vol.2 (2002).

15) Kashima, H. and Koyanagi, T.: SVM Kernels for Semi-Structured Data, *Proc.ICML*, pp.291–298 (2002).

16) Collins, M. and Duffy, N.: Convolution Kernels for Natural Language, *Advances in Neural Information Processing Systems 14, Vol.1* (*NIPS 2001*), pp.625–632 (2001).

☆ さらに，部分集合重み $c_d(r)$ を再定義することで，一般の多項式カーネル $\sum_{n=0}^{d} a_n|X \cap Y|^n$ $(\forall n, a_n \in \mathbb{R})$ に適用可能である．

付 録 A

$$
\begin{aligned}
K_d(X,Y) &= \sum_{l=r}^{d} c_d(r) \cdot |P_r(X \cap Y)| \\
c_d(r) &= \sum_{l=r}^{d} \binom{d}{l} \Bigl(\sum_{m=1}^{r} (-1)^{r-m} \cdot m^l \binom{r}{m} \Bigr).
\end{aligned}
$$

略証

集合 X に対し，$|X|^d$ $(d = 1, 2, \ldots)$ は，以下のように展開できる．

$$
|X|^d = \sum_{r=0}^{d} \tau_d(r) \cdot |P_r(X)|
$$

ただし，$\tau_d(r)$ は，互いに異なる r $(r \leq d)$ 個のものを重複を許し，少なくともそれぞれ 1 つずつ d 個ならべる順列である．たとえば，$d = 3$ $r = 2$ の場合，[a,a,b]，[a,b,a]，[b,a,a]，[b,b,a]，[b,a,b]，[a,b,b] となり $\tau_3(2) = 6$ となる．この場合，集合 (a,b)（サイズ 2 の部分集合）の係数が 6 と計算される．この性質は $|X|^d$ の多項定理を用いた展開式から導出される．一般に，$\tau_d(r)$ は以下のように展開される．

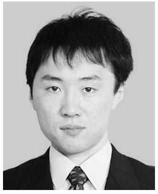
$$
\begin{aligned}
\tau_d(r) &= \sum_{k_n \geq 1, n=1,2,\ldots,r}^{k_1+\ldots+k_r=d} \frac{d!}{k_1! \ldots k_r!} \\
&= \sum_{m=1}^{r} (-1)^{r-m} \cdot m^d \binom{r}{m}
\end{aligned}
$$

少なくとも 1 つずつ並べるため，通常の多項定理とは異なり，$k_n \geq 1$ となることに注意されたい．上式の変換は指数型母関数 $(x^1/1!+x^2/2!+\ldots)^r = (\exp(x)-1)^r$ の $x^d/d!$ の係数から導出される．式展開の詳細は割愛する．

以上より，

$$
\begin{aligned}
K_d(X,Y) &= (1 + |X \cap Y|)^d \\
&= \sum_{l=0}^{d} \binom{d}{l} \Bigl(|X \cap Y|^l \Bigr) \\
&= \sum_{l=r}^{d} \binom{d}{l} \Bigl(\sum_{m=1}^{r} (-1)^{r-m} \\
&\qquad \cdot m^l \binom{r}{m} \cdot |P_r(X \cap Y)| \Bigr) \\
&= \sum_{l=r}^{d} c_d(r) \cdot |P_r(X \cap Y)| \qquad \square
\end{aligned}
$$

(平成 15 年 10 月 27 日受付)
(平成 16 年 7 月 1 日採録)



工藤 拓（正会員）

1999 年京都大学工学部電気電子工学科卒業．2001 年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．2004 年同大学院博士後期課程修了．同年より NTT コミュニケーション科学基礎研究所リサーチアソシエイト．2001 年度本学会山下記念研究賞受賞．工学博士．統計的自然言語処理，テキストマイニング，機械学習に興味を持つ．

松本 裕治（正会員）

1977 年京都大学工学部情報工学科卒業．1979 年同大学大学院工学研究科修士課程情報工学専攻修了．同年電子技術総合研究所入所．1984 年～1985 年英国インペリアルカレッジ客員研究員．1985 年～1987 年（財）新世代コンピュータ技術開発機構に出向．京都大学助教授を経て，1993 年より奈良先端科学技術大学院大学教授，現在に至る．工学博士．専門は自然言語処理．人工知能学会，日本ソフトウェア科学会，言語処理学会，認知科学会，AAAI，ACL，ACM 各会員．