構文森を用いた実コーパスからの 大規模な文脈自由文法の高速学習法

Efficient Grammar Induction Algorithm with Parse Forests from Real Corpora

栗原 賢一

Kenichi KURIHARA

東京工業大学大学院情報理工学研究科計算工学専攻

Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology
kurihara@mi.cs.titech.ac.jp, <http://sato-www.cs.titech.ac.jp/kurihara/>

亀谷 由隆

Yoshitaka KAMEYA

(同 上)

kameya@mi.cs.titech.ac.jp, <http://sato-www.cs.titech.ac.jp/kameya/>

佐藤 泰介

Taisuke SATO

(同 上)

sato@mi.cs.titech.ac.jp, <http://sato-www.cs.titech.ac.jp/sato/>**keywords:** grammar induction, probabilistic context-free grammars, the EM algorithm

Summary

The task of inducing grammar structures has received a great deal of attention. The reasons why researchers have studied are different; to use grammar induction as the first stage in building large treebanks or to make up better language models. However, grammar induction has inherent computational complexity. To overcome it, some grammar induction algorithms add new production rules incrementally. They refine the grammar while keeping their computational complexity low. In this paper, we propose a new efficient grammar induction algorithm. Although our algorithm is similar to algorithms which learn a grammar incrementally, our algorithm uses the graphical EM algorithm instead of the Inside-Outside algorithm. We report results of learning experiments in terms of learning speeds. The results show that our algorithm learns a grammar in constant time regardless of the size of the grammar. Since our algorithm decreases syntactic ambiguities in each step, our algorithm reduces required time for learning. This constant-time learning considerably affects learning time for larger grammars. We also reports results of evaluation of criteria to choose nonterminals. Our algorithm refines a grammar based on a nonterminal in each step. Since there can be several criteria to decide which nonterminal is the best, we evaluate them by learning experiments.

1. は じ め に

現在, 統計的言語モデルのクラスの一つとして確率文脈自由文法 (probabilistic context-free grammar; 以下 PCFG) が広く知られている. PCFG は文脈自由文法 (context-free grammar; 以下 CFG) に確率パラメータが付与されたものである. また, PCFG を自然言語以外のデータの確率モデルとして用いた研究も行われている [Bockhorst 01, Sakakibara 94].

コーパス^{*1} からの PCFG の学習は Lari & Young が Inside-Outside アルゴリズム [Baker 79] (以下, I-O アルゴリズム) を用いた方法を提案している [Lari 90]. その後, Pereira & Schabes は括弧付けされたコーパス

を訓練コーパスに用いる方法を提案し, 括弧なしの訓練コーパスに比べ高速に学習されることを実験的に示した [Pereira 92]. しかし, 依然として規模の大きい PCFG の学習は計算量的に困難であった. この困難の克服のために, Hogenhout & Matsumoto や Bockhorst & Craven は小さい文法から徐々に文法の規模を大きくしていく学習法を提案している [Hogenhout 98, Bockhorst 01].

本論文では, Hogenhout & Matsumoto や Bockhorst & Craven のように小さい文法から徐々に文法の規模を大きくする文法学習の枠組みに於いて, 新しい高速学習法を提案する. 提案手法のパラメータ学習は, I-O アルゴリズムではなく gEM アルゴリズム [Kameya 00, Sato 01b] に基づいており, 文法の規模に依らずほぼ一定時間で学習可能であることを実コーパスに基づく実験により示す. また, 本提案手法では文法規模を大きくするための非終端記号の選択に自由度があるが, いくつかの選択基準を実験により評価する.

^{*1} 本論文では例文の集合をコーパスと呼ぶ. また, 訓練に用いる文の集合を訓練コーパス, テストに用いる文の集合をテストコーパスと呼ぶ. コーパスに含まれる各文に構文構造 (内部の非終端記号は除く) が付与されている場合, コーパスが括弧付けされているという.

- (1) $G_0(\theta_0)$ を用意し, $t \leftarrow 0$ とする
- (2) $G_t(\theta_t)$ に gEM アルゴリズムを用いてパラメータ学習を行い, これを $G_t(\hat{\theta}_t)$ とする
- (3) $G_t(\hat{\theta}_t)$ の生成規則のうちパラメータが 0 のものを削除し $G'_t(\hat{\theta}_t)$ を作る
- (4) $G'_t(\hat{\theta}_t)$ を拡大した文法を作り, $G_{t+1}(\theta_{t+1})$ とする
- (5) $t \leftarrow t+1$ とし, (2) に戻る

図 1 提案手法のアルゴリズム

2. 提案手法

2.1 準備

まず, CFG を 4 項組 $G = (V_N, V_T, R, S)$ により定義する. V_N, V_T, R はそれぞれ, 非終端記号, 終端記号, 生成規則の集合であり, S は開始記号 ($S \in V_N$) である. 次に, PCFG を $G(\theta)$ により定義する. θ は $|R|$ 次元のパラメータのベクトルである. 生成規則 $r (r \in R)$ のパラメータを $\theta(r)$ により参照する. 本論文では文を導出する適用規則列 r は最左導出列であるとし, ϵ 生成規則や非終端記号 A から 1 回以上の生成規則の適用により A が生成されるような生成規則は生成規則集合に含まれないとする. CFG G に対して, 文 w を導出する全ての適用規則列の集合を $\Phi_G(w)$ と表わすことにする.

2.2 文法の拡大

図 1 が提案手法のアルゴリズムである. 提案手法ではパラメータ学習の後, 文法を拡大する (図 1 の (4)). この拡大に於いて, ある非終端記号 A を含む全ての生成規則を $A_1, A_2, \dots, A_n$ を用いて複製する. 次の例は, 非終端記号 A を A_1, A_2 により複製した際の, 生成規則集合の変化を示している.

$$\begin{array}{ccc} \text{拡大前} & & \text{拡大後} \\ \left\{ \begin{array}{l} X \rightarrow \beta A \gamma \\ A \rightarrow \delta A \epsilon \\ A \rightarrow \alpha \end{array} \right\} & \rightarrow & \left\{ \begin{array}{ll} X \rightarrow \beta A_1 \gamma & X \rightarrow \beta A_2 \gamma \\ A_1 \rightarrow \delta A_1 \epsilon & A_1 \rightarrow \delta A_2 \epsilon \\ A_2 \rightarrow \delta A_1 \epsilon & A_2 \rightarrow \delta A_2 \epsilon \\ A_1 \rightarrow \alpha & A_2 \rightarrow \alpha \end{array} \right\} \end{array}$$

この例では A を A_1, A_2 の二つに拡大しているので, 本論文では「文法を A について 2 倍に拡大する」と呼ぶことにする. 以後, 文法 G の非終端記号 A を n 倍に拡大することを, 拡大後の文法を G' とし $G' = \text{Expand}(G, A, n)$ と書く.

次に, 拡大後のパラメータについて述べる. PCFG $G = (V_N, V_T, R, S)$ とパラメータ θ を非終端記号 A について拡大した PCFG を $G'(\theta')$ とする. この時パラメータ θ' は以下の式(2) と(3) を満たすように与える. ただし, $\alpha \in (V_N \cup V_T \setminus \{A\})^+$, 任意の k に対し $\beta_k \in (V_N \cup V_T \setminus \{A\})^*$, $X \in V_N \cup \bigcup_{1 \leq k \leq n} \{A_k\}$ である. Y は式(2)

と(3) とともに,

$$Y = \begin{cases} X & \text{if } X \neq A_k \ (1 \leq k \leq n) \\ A & \text{otherwise} \end{cases} \quad (1)$$

である. また式(3) の i の値は生成規則の右辺に現われる A_k ($1 \leq k \leq n$) の数である.

$$\theta'(X \rightarrow \alpha) = \theta(Y \rightarrow \alpha) \quad (2)$$

$$\begin{aligned} \sum_{k_1=1}^n \sum_{k_2=1}^n \cdots \sum_{k_i=1}^n \theta'(X \rightarrow \beta_1 A_{k_1} \beta_2 A_{k_2} \cdots \beta_i A_{k_i} \beta_{i+1}) \\ = \theta(Y \rightarrow \beta_1 A \beta_2 \cdots \beta_i A \beta_{i+1}) \end{aligned} \quad (3)$$

2.3 文法の拡大の性質

一般に, EM アルゴリズムにより推定されるパラメータは初期パラメータによる. しかし, 初期パラメータの探索空間は非常に広い. このため, 提案手法では文法の拡大により追加される生成規則の初期パラメータへの制約として, 式(2) と(3) を与える. 式(2) と(3) を満たすことにより, 拡大の前後におけるコーパスの尤度 ($= \prod_{w \in C} P_\theta(w)$) を保存するということが証明される.

[定理 1] PCFG $G(\theta)$ ($G = (V_N, V_T, R, S)$) に対して, $A \in V_N$ を $A_1, A_2, \dots, A_n$ により拡大した後の PCFG を $G^+(\theta^+)$ とする. ただし, $G^+ = \text{Expand}(G, A, n)$ であり, θ^+ は式(2), (3) を満たすとする.

この時, 任意の文 w に対して

$$P_\theta(w) = P_{\theta^+}(w) \quad (4)$$

である. □

定理 1 よりコーパスの尤度が保存されることは明らかである. 定理 1 の証明は, 付録 A に与える.

文法の拡大が尤度を保存する一方, EM アルゴリズムは尤度を増加させる. これら 2 つから, EM アルゴリズムと文法の拡大を交互に繰り返す提案手法は, 尤度を単調増加させるということが言える^{*2}.

2.4 非終端記号の選択

拡大 $\text{Expand}(G, A, n)$ により作られる文法は, 非終端記号 A の選択により大きく異なる. このため, よりよい文法を学習するために適切な A を選択する必要があるが, 選択の基準には様々なものが考えられる.

選択基準を定義する前に, いくつかの記号と Bockhorst & Craven が提案している context という用語を定義する. $n(A \rightarrow \alpha, r)$ は適用規則列 r の中に生成規則 $A \rightarrow \alpha$ が用いられている回数である. context は $(X \rightarrow \alpha A \beta, A)$ と

*2 図 1 の 3 に於いて, 実装上は閾値を設け, 閾値以下のパラメータを持つ生成規則を削除した. このため, 正確には尤度は保存されない. なお, この論文では全ての実験に於いてこの閾値を 10^{-10} とした.

いう生成規則と非終端記号から成る二項組である． $n(A \rightarrow \alpha, (X \rightarrow \beta A \gamma, A), r)$ とは，context $(X \rightarrow \beta A \gamma, A)$ に条件付けられた $n(A \rightarrow \alpha, r)$ である．つまり，適用規則列 r の導出において， $X \rightarrow \beta A \gamma$ の右辺の A に対して $A \rightarrow \alpha$ が適用された回数である^{*3}．

次に，コーパス中の生成規則の適用回数と非終端記号の出現回数を式(5)と(6)により定義する．また，context により条件付けられた生成規則の適用回数と非終端記号の出現回数を式(7)と(8)により定義する．ただし，コーパスを C とし， c とは $(X \rightarrow \beta A \gamma, A)$ という形の context であるとする．

$$n(A \rightarrow \alpha) = \sum_{w \in C} \sum_{r \in \Phi_G(w)} P_\theta(r|w) n(A \rightarrow \alpha, r) \quad (5)$$

$$n(A) = \sum_{\alpha; A \rightarrow \alpha \in R} n(A \rightarrow \alpha) \quad (6)$$

$$n(A \rightarrow \alpha, c) = \sum_{w \in C} \sum_{r \in \Phi_G(w)} P_\theta(r|w) n(A \rightarrow \alpha, c, r) \quad (7)$$

$$n(A, c) = \sum_{\alpha; A \rightarrow \alpha \in R} n(A \rightarrow \alpha, c) \quad (8)$$

式(5)から式(8)を用いて，式(9)から式(11)を定義する．

$$P(\alpha|A) = \frac{n(A \rightarrow \alpha)}{\sum_{\alpha} n(A \rightarrow \alpha)} = \theta(A \rightarrow \alpha) \quad (9)$$

$$P(\alpha|A, c) = \frac{n(A \rightarrow \alpha, c)}{\sum_{\alpha} n(A \rightarrow \alpha, c)} \quad (10)$$

$$P(c|A) = \frac{n(A, c)}{\sum_c n(A, c)} \quad (11)$$

以上を用いて，選択基準 S_f, S_{pKL}, S_{fKL} を定義する．ただし，K-L divergence $D(P||Q)$ は式(12)により定義される．

$$D(P||Q) = \sum_x P(x) \log P(x)/Q(x) \quad (12)$$

$$S_f(A) = n(A) \quad (13)$$

$$S_{pKL}(A) = \sum_c P(c|A) D(P(\alpha|A, c)||P(\alpha|A)) \quad (14)$$

$$S_{fKL}(A) = \sum_c n(A, c) D(P(\alpha|A, c)||P(\alpha|A)) \quad (15)$$

いずれの基準を用いる場合も，基準を最大にする非終端記号 A に対して文法の拡大を行う．

^{*3} 右辺に複数の A がある場合はそれぞれを区別する．例えば， $X \rightarrow AA$ に対しては $(X \rightarrow AA', A)$ や $(X \rightarrow AA', A')$ という context を考え，2つの A を区別する．ただし， A と A' は context の中で便宜的に区別しているにすぎず，導出の過程では同じ非終端記号 A である．

図2 構文木の例

2.5 EM アルゴリズム

亀谷らは文法が既知の時の PCFG のパラメータ学習において，パース後に gEM アルゴリズムを用いる手法が I-O アルゴリズムに比べ極めて高速であることを実験的に示している [亀谷 01, Sato 01a]．

I-O アルゴリズムと gEM アルゴリズムについて簡単に述べる．I-O アルゴリズムは一回のパラメータ更新の中で，inside 確率の計算と outside 確率の計算が行われる．inside 確率は CYK テーブルを用いて訓練文をパースすると同時に計算される．このため，I-O アルゴリズムはパラメータ更新の度にパースしていることになる．また，inside 確率の計算過程では，最終的に不必要になる inside 確率も計算しなくてはならない．これに対して gEM アルゴリズムは最初にパースを一度だけ行い支持グラフ^{*4}というデータ構造を作る．パラメータ更新はこの支持グラフのみを用いて行われ，CYK テーブルを用いる I-O アルゴリズムと違い必要な計算のみ行われる．以上から，gEM アルゴリズムを用いると学習の効率向上が期待できる．また，文法の曖昧性が少ないと支持グラフは小さくなるため，I-O アルゴリズムと gEM アルゴリズムの効率の違いはより顕著になる．

提案手法では，図1の(2)において，パースによる支持グラフの作成とパラメータ学習が行われる．しかし， $t \geq 1$ の時は $t-1$ ステップ目の支持グラフを直接修正することにより効率的に t ステップ目の支持グラフを作成できる．このため，パースは図1の最初のステップ $t=0$ の(2)において一度行えばよく，以後， $t \geq 1$ の(2)では支持グラフの修正のみが行われる．図2はある文の文法 G による構文木 $t1, t2$ である．これら構文木に対応する支持グラフの一部が図3の左である． G を拡大した $G' (= \text{Expand}(G, A, 2))$ の支持グラフは図3右のようになる． Expand による文法の変化は通常文法全体に比べ小さなものであるため，支持グラフの変更も支持グラフ全体に対して小さい変更で済む．

2.6 計 算 量

I-O アルゴリズムと gEM アルゴリズムの計算量及び，非終端記号の選択基準の計算量について述べる．以下では，文法はチョムスキー標準形 (Chomsky normal form 以下，CNF) であるとする．

^{*4} PCFG の場合は構文森 (parse forests [Billot 89]) と同様な構造である．

図 3 G による支持グラフの一部 (左) と G' による支持グラフの一部 (右)

訓練文が括弧付けされていない時の I-O アルゴリズムの時間計算量はコーパスに含まれる文の数を N ，コーパス内の最大の文長を l ，生成規則集合を R として， $O(N|R|l^3)$ である．同様に括弧付けされていない時の gEM アルゴリズムは支持グラフの大きさを S として， $O(NS)$ である．ここで， $O(S) = O(|R|l^3)$ である．訓練文が完全に括弧付けされている時は，I-O アルゴリズムの時間計算量は $O(N|R|l)$ であり，gEM アルゴリズムの時間計算量は $O(NS)$ である．ただし，この時， $O(S) = O(|R|l)$ である．以上より，括弧付けがあるないに関わらず，I-O アルゴリズムと gEM アルゴリズムの時間計算量は等しい．また，括弧付けがあることにより高速にパラメータ推定を行うことができると期待できる．

非終端記号の選択基準 S_f の計算には式(5)，選択基準 S_{pKL}, S_{fKL} の計算には式(7)が必要である．式(5)は EM アルゴリズムにより計算されるので，全ての非終端記号について S_f を求める時間計算量は I-O アルゴリズムや gEM アルゴリズムと同じである．訓練文が括弧付けされていない時 $O(N|R|l^3)$ であり，完全に括弧付けされている時 $O(N|R|l)$ である．式(7)の計算は context 毎に生成規則の適用回数を数える必要がある．このため，全ての非終端記号について S_{pKL}, S_{fKL} を求める時間計算量は，括弧付けされていない時 $O(N|R||V_N|^2l^3)$ であり，完全に括弧付けされている時 $O(N|R||V_N|^2l)$ である．ここで， $|V_N|^2$ は非終端記号 A を固定した時， $(X \rightarrow AY, A)$ または $(X \rightarrow YA, A)$ という形をした context から成る集合の最大の要素数である．

3. 実験

以下の実験では訓練コーパスに田中ら [田中 97] が手を加えた ATR 対話コーパス [浦谷 94] を用いる．このコーパスには日本語の対話文が 10,995 文収録されており，文長は平均 9.97，最短 2，最長 49 である．訓練に

表 1 30 回の *Expand* に要した時間

初期文法	1	2	3
I-O(時間)	14.6	9.6	13.0
gEM(時間)	3.0	1.9	2.9

は単語列ではなく品詞を細分化したカテゴリーの列を用いた．全ての文は括弧付けされている．以降の実験に用いる初期文法は次のように与えた．終端記号は訓練コーパスに現われるもの全て，非終端記号は開始記号を含め 3 つ，生成規則集合はこれら終端記号と非終端記号から作られる CNF の生成規則全てである．断わりのない限り初期パラメータには一様分布に分散 10^{-8} のガウスノイズを与えた分布を用いた．また，文法の学習では常に文法を 2 倍に拡大した．ところで，ATR 対話コーパスの括弧付けは CNF ではない．これに対して，初期文法が CNF であるため，学習される文法も CNF である．このため，学習される文法からは ATR 対話コーパスに付与されている括弧は部分的な括弧付けに見える．例えば，生成規則 $A \rightarrow bcd$ による括弧は (bcd) であるが，チョムスキー標準形の文法から付与される括弧は $((bc)d)$ か $(b(cd))$ のいずれかである．このため，3.2 節で用いる評価基準 consistent brackets は bracketed precision とは必ずしも一致しない．

3.1 学習時間

提案手法において gEM アルゴリズムの代わりに I-O アルゴリズムを用いたアルゴリズムと提案手法について学習時間の比較を行った．訓練にはコーパスの全文を用いている．互いに異なった初期パラメータを持つ初期文法を 3 つ作り，それぞれに提案手法と I-O アルゴリズムを用いた提案手法を適用した^{*5}．訓練文は括弧付けされているため，I-O アルゴリズムは Pereira & Schabes が提案した括弧付けを考慮するものである [Pereira 92]．また支持グラフの作成には括弧付けを考慮する CYK パーザを用いた．*Expand* を 30 回行うのに要した時間は表 1 のようになった．いずれの初期文法においても 4 倍以上提案手法の方が高速である．図 4 は EM アルゴリズムの一回のパラメータ更新に要した時間である．I-O アルゴリズムは *Expand* により文法の規模が大きくなるにつれ，パラメータ更新にかかる時間が長くなっている．これに対して，gEM アルゴリズムを用いている提案手法は文法の規模に関わらずほぼ一定の時間でパラメータ更新が行われている．

表 2 は訓練コーパスのうち文長が 10^{*6} である 699 文において，各文の構文木数の合計と平均である．表 2 の G_0 と G'_{30} は図 1 の初期文法と $t = 30$ のパラメータ学習

^{*5} 実験に用いた gEM アルゴリズムは阿部の実装 (<http://sato-www.cs.titech.ac.jp/gemini/ja/>)，CYK パーザと I-O アルゴリズムは著者の実装である．実験に用いた計算機は CPU Athlon 1.7 GHz，メモリ 1 GB，OS は GNU / Linux 2.4 である．

^{*6} ATR 対話コーパスの平均文長は 9.97 である．

図 4 一回のパラメータ更新に要した時間

表 2 構文木数の変化

文法	G_0	G'_{30}
構文木数の平均	4.57×10^8	9.40×10^3
構文木数の合計	3.19×10^{11}	6.58×10^6

済みの文法である。学習により、文法の生成規則数が増加するが、構文木数は減少しているのが分かる。構文木数の減少は文法の曖昧性が減少していることを意味する。曖昧性の減少は 2.5 節で述べたように支持グラフを小さくするため、gEM アルゴリズムは文法が大きくなっても効率よく学習できる。逆に I-O アルゴリズムの学習時間は生成規則数に比例するため、文法が大きくなるにつれ、より学習に時間がかかるようになる。

3.2 文法の精度

ATR 対話コーパスには正解構文木が付与されているので、学習された文法とそのパラメータから得られる最尤 (Viterbi) 構文木の精度を正解構文木を用いて評価した。学習される文法の非終端記号は機械的に作成されるので、正解構文木に付与された非終端記号と対応付けできない。また、学習される文法は CNF であるのに対し、正解構文木は CNF ではない文法から作られている。以上の理由から、評価基準には consistent brackets (以下、CB) を用いた [Goodman 98]。CB は最尤構文木と正解構文木において交差がない括弧の割合の平均である。実験では 5-fold の交差検定を行った。訓練コーパスには 8,796 文、テストコーパスには 2,199 文が含まれている。

図 5 は非終端記号の選択基準に S_f , S_{fKL} , S_{pKL} を用いて CB により評価した結果である。横軸は $Expand$ を行った回数である。 S_{fKL} を用いた時 CB が最も高くなっている。 S_f , S_{fKL} , S_{pKL} を用いた時の学習時間の平均はそれぞれ、1.1, 10.8, 9.1 時間であった。この学習時間の差は、2.6 節で述べたように、 S_{fKL} と S_{pKL} の時間計算量が S_f と比較して大きいためである。実際、 S_{fKL} と S_{pKL} を用いた時、学習時間は約 10 時間である

図 5 選択基準による精度の違い

表 3 文長に対する精度

文長	0-10	0-15	10-19	20-30
提案手法	0.905	0.870	0.782	0.748
P&S	0.857	0.812	0.710	0.680
ベースライン	0.778	0.745	0.655	0.619

が、そのうち gEM アルゴリズムの計算にかかった時間は約 1 時間であり、残りの約 9 時間は S_{fKL} と S_{pKL} の計算にかかっていた。

表 3 は提案手法と Pereira & Schabes の方法 [Pereira 92] を CB を用いて比較している。Pereira & Schabes の方法は最初に非終端記号数を決める。本実験では提案手法と Pereira & Schabes らの方法にかかる学習時間がほぼ等しくなるように、Pereira & Schabes の方法に用いる非終端記号数は 8 とした*7。提案手法は非終端記号の選択基準に S_f を用い $Expand$ を 40 回行った*8。表 3 のベースラインとは、CNF の括弧付けをランダムに生成し、正解構文木に対し CB を用いて評価したものである。表 3 より、文長に依らず提案手法は Pereira & Schabes の方法に比べ精度が向上していることが分かる。

3.3 尤度の保存

まず、尤度の保存が行なわれていることを実験的に確認する。ここでは、コーパスの全文を訓練に用い、 $Expand$ を 30 回行った。なお、非終端記号の選択基準には非終端記号の出現回数 S_f を用いた。この時の EM アルゴリズムのパラメータ更新毎の対数尤度は図 6 のようになった。横軸は、EM アルゴリズムによって行われるパラメータ更新の延回数である。図 6 中の記号 “ ” によりプロットされている時点で $Expand$ が適用されている。2.3 節に

*7 Pereira & Schabes の方法は最初に与える非終端記号数の 3 乗に比例する。また、学習時間は EM アルゴリズムのパラメータ更新が何回行われるかによるので、学習時間を正確に等しくさせることは難しい。本実験では提案手法、Pereira & Schabes らの方法共に 1 時間から 3 時間で学習が終わった。

*8 図 5 より S_f よりも S_{fKL} を用いた方が精度の向上を期待できるが、 S_{fKL} は 2.6 節で述べたように時間計算量が大きく学習に時間がかかるため、ここでは S_f を用いた。

図 6 EM アルゴリズムのパラメータ更新毎の対数尤度

において示したように, *Expand* は文法構造を変更するにも関わらず, 対数尤度は単調増加していることが分かる.

次に, 尤度を保存した場合 (式(2) と(3) を満たした場合) と尤度を保存しなかった場合 (式(2) と(3) を満たさない場合) の比較を行った. 互いに異なる初期パラメータを持つ初期文法を 7 つ用意し, それぞれに, 尤度を保存する場合と保存しない場合の提案手法を適用した. 非終端記号の選択基準には S_f を用い, 40 回 *Expand* を行った. 学習された 7 つの文法のうち, 3 つで尤度を保存した方が CB が高く, 4 つで保存しない方が尤度が高かった. 尤度を保存した際と保存しなかった際の CB の平均値はそれぞれ, 0.834, 0.835 であった. この結果から尤度を保存するという制約は少なくとも解析精度に悪影響を与えていないことが分かる.

4. 関 連 研 究

提案手法は Hogenhout & Matsumoto [Hogenhout 98] や Bockhorst & Craven [Bockhorst 01] の方法に近い. これらは, 全て EM アルゴリズムを用いている点と, 文法を拡大する操作がある点で同じである.

Hogenhout & Matsumoto の方法では, 非終端記号の選択基準に非終端記号の出現回数を用いている. これは本論文で用いた S_f と同一である. 同方法では拡大する非終端記号を選択すると, その非終端記号を含む生成規則を全て削除し, その非終端記号を含む全ての生成規則を追加する. 追加される生成規則の数はおよそ $|V_N|^2$ になる. 提案手法では生成規則集合に含まれる形の生成規則のみを追加するので, 追加する生成規則の数は $|V_N|^2$ に比べ小さい. コーパスには [Schabes 93] と同様 Penn Treebank を用い, 訓練文は 1,000 文であり, 括弧付けされている. また, Hogenhout & Matsumoto は文法の拡大に於いて尤度のある程度保存できると述べている. ただし, 実コーパスに対する実験は行っていない.

Bockhorst & Craven はコーパスに RNA を用いている. 括弧付けはされていない. 提案手法が生成規則集合

中の, ある非終端記号 A 全ての出現箇所を拡大するのは異なり, Bockhorst & Craven の手法はある context を満たす生成規則を拡大する. このため, 追加される生成規則は通常, 提案手法よりも少ない. 非終端記号の選択基準には $n(A, c)D(P(\alpha|A, c)||P(\alpha|A))$ (これを c について和を取れば式(15) になる), χ^2 統計量を用いている. Bockhorst & Craven の行った選択基準の比較実験では, χ^2 統計量の方がよい結果であった. 提案手法とは拡大の仕方が異なるため, 提案手法では χ^2 統計量を用いた基準を使えなかった. 逆に, Bockhorst & Craven の方法には非終端記号の出現回数を選択基準には使えない.

5. ま と め

徐々に文法の規模を大きくしていくという文法学習アプローチの枠組みに於ける高速な学習アルゴリズムを提案し, このアルゴリズムによる大幅な学習時間の短縮を実コーパスを用いて実験的に示した. 学習時間は 10,000 文のコーパスから 2 時間前後であった. 実験の結果, 提案手法の学習時間が文法の規模に対してほぼ一定であったことから, より規模の大きな文法を学習させた場合, 学習時間の短縮はより顕著になると考えられる. また, 文法拡大の対象になる非終端記号の選択基準について実データを用いて比較実験を行った. 実験では S_{fKL} を用いた時, 文法の精度は最もよかった. Pereira & Schabes の方法との比較では文長に依らず精度が向上することが確認できた.

今後の課題としては, 提案手法を他のコーパスに適用した際の評価や, より規模の大きいコーパスへの適用が挙げられる. 学習される文法の精度の向上に関しては, 非終端記号の他の選択基準やビーム探索を検討する必要があると考えられる.

◇ 参 考 文 献 ◇

- [Baker 79] Baker, J. K.: Trainable grammars for speech recognition. Proceedings of the Spring Conference of the Acoustical Society of America, pp.547-550, 1979.
- [Billot 89] Billot, Sylvie. and Lang, Bernard.: The Structure of Shared Forests in Ambiguous Parsing. Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics, 1989.
- [Bockhorst 01] Bockhorst, J. and Craven, M.: Refining the Structure of a Stochastic Context-Free Grammar. Proceedings of the 17 th International Joint Conference on Artificial Intelligence, 2001.
- [Goodman 98] Goodman, J.: Parsing inside-out, Ph.D. Thesis, Harvard University, 1998.
- [Hogenhout 98] Hogenhout, W. R. and Matsumoto, Y.: A fast method for statistical grammar induction, Natural Language Engineering, 4, 3, pp.191-209, 1998.
- [Kameya 00] Kameya, Y. and Sato, T.: Efficient EM learning with tabulation for parameterized logic programs, Proceedings of the 1st International Conference on Computational Logic, pp.269-294, 2000.
- [亀谷 01] 亀谷由隆, 森高志, 佐藤泰介: WFST に基づく確率文脈自由文法およびその拡張文法の高速 EM 学習, 自然言語処理,

8, 1, pp.49-84, 2001.

- [Lari 90] Lari, K. and Young, S. J.: The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, 4, 35-56, 1990.
- [Pereira 92] Pereira, F. and Schabes, Y.: Inside-Outside reestimation from partially bracketed corpora. *Proceedings of 30th Annual Meeting of the Association for Computational Linguistics*, 128-135, 1992.
- [Sakakibara 94] Sakakibara, Y., Brown, M., Hughey, R., Mian, I. S., Sjölander, K., Underwood, R. C., and Haussler, D.: Stochastic context-free grammars for tRNA modeling. *Nucleic Acids Research*, 22:5, 112-5120, 1994.
- [Sato 01a] Sato, T., Abe, S., Kameya, Y., and Shirai, K.: A separate-and-learn approach to EM learning of PCFGs. *Proceedings of the 6th Natural Language Processing Pacific Rim Symposium (NLP-RS-2001)*, 255-262, 2001.
- [Sato 01b] Sato, T. and Kameya, Y.: Parameter Learning of Logic Programs for Symbolic-Statistical Modeling, *Journal of Artificial Intelligence Research*, 15, pp.391-454, 2001.
- [Schabes 93] Schabes, Y., Roth, M. and Osborne, R.: Parsing the Wall Street Journal with the Inside-Outside algorithm. *Proceedings of the 6th Conference of the European Chapter of Association for Computational Linguistics*, 341-347, 1993.
- [田中 97] 田中穂積, 竹澤寿幸, 衛藤純司: MSLR 法を考慮した音声認識用日本語文法 - LR 表工学 (3) -. 音声言語情報処理研究会, 情報処理学会, Vol.97.
- [浦谷 94] 浦谷則好, 竹澤寿幸, 松尾秀彦, 森田千帆.: 音声言語データベースの構成 (ATR Integrated Speech and Language Database). ATR Technical Report, TR-IT-0056, エイ・ティ・アル音声翻訳通信研究所, 1994.

〔担当委員: 堀 浩一〕

2003 年 10 月 10 日 受理

◇ 付 録 ◇

A. 定理 1 の証明

本付録では, 定理 1 の証明を与える.

《証明》 まず, 操作 e を定義する. 整数 n , PCFG $G(\theta) = (V_N, V_T, R, S)$ が与えられた時, 図 A.1 に示す操作により PCFG $G'(\theta') = (V'_N, V_T, R', S)$ を作ることを操作 $e(Y \rightarrow \alpha A \beta, n)$ とする.

次に, $I_1, I_2, \phi_{G, G'}(r)$ を式(A.1) から式(A.3) により定義する. ただし, 適用規則列 r の長さを $l(r)$ とし, r の i 番目の生成規則を r_i ($1 \leq i \leq l(r)$) により参照する. なお, $|\phi_{G, G'}(r)| = n^{|I_2(r)|}$

V'_N, R' を以下により作る. また, θ' を式(2) と(3) を満たすように作る. ただし, $Y, A \in V_N, \alpha, \beta \in (V_N \cup V_T)^*$ であり, α, β には A が含まれてもよい.

$$\begin{aligned} V'_N &= V_N \cup \bigcup_{1 \leq i \leq n} \{A_i\} \\ R' &= R'' \cup \bigcup_{\gamma; A \rightarrow \gamma \in R''} \bigcup_{1 \leq i \leq n} \{A_i \rightarrow \gamma\} \\ R'' &= R \setminus \{Y \rightarrow \alpha A \beta\} \cup \bigcup_{1 \leq i \leq n} \{Y \rightarrow \alpha A_i \beta\} \end{aligned}$$

図 A.1 操作 $e(Y \rightarrow \alpha A \beta, n)$

$$r = (r_1, r_2, \dots, Y \rightarrow \alpha A \beta, \dots, r_{|r|})$$

$$\phi_{G, G'}(r) = \left\{ \begin{array}{c} (r_1, r_2, \dots, Y \rightarrow \alpha A_1 \beta, \dots, r_{|r|}) \\ \vdots \\ (r_1, r_2, \dots, Y \rightarrow \alpha A_n \beta, \dots, r_{|r|}) \end{array} \right\}$$

図 A.2 $\phi_{G, G'}(r)$ の例

である. 図 A.2 は, ある r に対する $\phi_{G, G'}(r)$ の例である.

$$I_1(r) = \left\{ i \mid \begin{array}{l} r_i \neq Y \rightarrow \alpha A \beta \text{ 且つ} \\ r_i \neq Y \rightarrow \alpha A_k \beta \ (1 \leq k \leq n) \end{array} \right\} \quad (\text{A.1})$$

$$I_2(r) = \{1, 2, \dots, l(r)\} \setminus I_1(r) \quad (\text{A.2})$$

$$\phi_{G, G'}(r) = \left\{ r' \mid \begin{array}{l} r \in \Phi_G(w), r' \in \Phi_{G'}(w), \\ r' \text{ は } r' \text{ 中の全ての } A_k \text{ を} \\ A \text{ に書換えると } r \end{array} \right\} \quad (\text{A.3})$$

操作 $e(Y \rightarrow \alpha A \beta, n)$ により $G(\theta)$ から $G'(\theta')$ を作った際に

$$P_\theta(w) = P_{\theta'}(w)$$

であることを示す. $r \in \Phi_G(w)$ かつ $r' \in \Phi_{G'}(w)$ の時, r'_i の全ての A_k ($1 \leq k \leq n$) を A により書き換えれば r_i になる. これより, $\Phi_{G'}(w) = \bigcup_{r \in \Phi_G(w)} \phi_{G, G'}(r)$ である. 以上と, 式(2) と(3) に注意して,

$$\begin{aligned} P_{\theta'}(w) &= \sum_{r' \in \Phi_{G'}(w)} \prod_{i=1}^{l(r')} \theta'(r'_i) \\ &= \sum_{r \in \Phi_G(w)} \sum_{r' \in \phi_{G, G'}(r)} \prod_{i \in I_1(r')} \theta'(r'_i) \prod_{j \in I_2(r')} \theta'(r'_j) \\ &= \sum_{r \in \Phi_G(w)} \prod_{i \in I_1(r)} \theta(r_i) \sum_{r' \in \phi_{G, G'}(r)} \prod_{j \in I_2(r')} \theta'(r'_j) \\ &= \sum_{r \in \Phi_G(w)} \prod_{i \in I_1(r)} \theta(r_i) \prod_{j \in I_2(r')} \sum_{k=1}^n \theta'(Y \rightarrow \alpha A_k \beta) \\ &= \sum_{r \in \Phi_G(w)} \prod_{i \in I_1(r)} \theta(r_i) \prod_{j \in I_2(r)} \theta(Y \rightarrow \alpha A \beta) \\ &= \sum_{r \in \Phi_G(w)} \prod_{i=1}^{l(r)} \theta(r_i) = P_\theta(w) \end{aligned} \quad (\text{A.4})$$

である.

操作 e を右辺に A が現われる生成規則がなくなるまで繰り返す. この操作により作られる文法の生成規則集合から $A \rightarrow \alpha$ (α は任意) の形の生成規則を全て削除し, 非終端記号集合から A を削除してできる文法が $\text{Expand}(G, A, n)$ である. よって式(4) が成立. □

著者紹介

栗原 賢一

2003 年東京工業大学工学部情報工学科卒業，2003 年同大学大学院情報理工学研究科計算工学専攻修士課程入学，現在在学中．

亀谷 由隆(正会員)

1995 年 3 月東京工業大学工学部情報工学科卒業，1997 年 3 月同大学大学院情報理工学研究科計算工学専攻修士課程修了，2000 年 9 月同研究科博士後期課程修了，技術補佐員などを経て 2001 年 4 月新日鉄ソリューションズ株式会社入社，2003 年 5 月より東京工業大学大学院情報理工学研究科計算工学専攻助手，博士（工学）．確率推論システムなどの研究に従事．人工知能学会会員．

佐藤 泰介(正会員)

1973 年東京工業大学工学部電子物理学科卒業，1975 年同大学大学院修士課程修了，工学博士．同年，電子技術総合研究所入所．以来，人工知能の研究に従事．1995 年以来，東京工業大学大学院情報理工学研究科教授．人工知能学会，情報処理学会，EATCS 各会員．