模倣学習による決定的解析での誤り伝播の回避

坪井 祐太
日本アイ・ビー・エム株式会社 東京基礎研究所
yutat@jp.ibm.com

1 はじめに

多くの NLP タスクは構造予測問題として定式化できる．たとえば，形態素解析・情報抽出は系列構造の予測，構文解析は木構造の予測である．予測を逐次的・決定的に繰り返すことで構造全体を予測する手法（決定的解析）はシンプルで解析が高速であるが，過去の予測誤りが以降の誤りを招く「誤り伝播」の問題がある．

決定的な Shift-reduce 依存構造解析（arc-eager 法 [9]）を例にとる．Shift-reduce 法ではスタックとキューに中間状態を保持しながら，文の先頭の単語（文節）から依存関係の判断をおこなっていく．各状態では，図 1 に示すような周辺情報を参照して次の状態を選択する．一般的に，次の状態を決める行動（action）は（多クラス）分類器を用いて選択するが，分類器は以前選択した行動によって決まる状態（図 1 の色付き部）を参照しているため，一度行動を誤ると連続して行動選択を誤ってしまう可能性がある．

この問題に対して，条件付き確率場（CRF）や構造化パーセプトロンなど構造全体の最適解を求める大域学習（global training）が広く用いられている．大域学習では指数個の構造候補を扱う必要があるため，次の二種類の近似手法を用いる必要ある：

1. 特徴量を小さな部分構造のみに制限する一方，動的計画法などによって正確な推定を行う手法；
2. 比較的広い範囲を考慮した特徴量を用いて，ビーム探索などの近似的な推定を行う手法．

特に，依存構造解析の研究では 1 は Graph-base 法，2 は Transition-base 法と呼ばれている．ただし，手法 1 は部分構造の大きさに対しては指数的な解析時間がかかり，手法 2 は近似精度を上げると計算量が増える．

本稿では NLP 分野で一般的でない模倣学習（Imitation Learning）を用いることで決定的解析のままでも誤り伝播を回避することができることを示す．模倣学習は強化学習の一種であり，エキスパートの一連の行動を模倣することで膨大な行動・状態空間の中から報酬の高い領域に集中して決定過程を学習することができる．逐次的な構造予測は，訓練データから生成した行動列を基に決定過程の学習を行う点や行動を選択する方策（policy）は分類器で近似する点など模倣学習と関連が深い．本稿では特に Dataset Aggregation（DAGGER）アルゴリズム [10] によって誤り伝播を回避する方法を示す．既存手法では訓練コーパスから正解行動列を生成して分類器を学習するが，正解行動列だけで学習した分類器は誤った行動の後に行動を選択することを想定していないことが問題である．DAGGER では正解行動列からだけでなく，学習した分類器が生成した状態に対しても正しい行動を対応させて訓練データとする（方策反復；policy iteration）．その結果，誤った後であっても正しい行動を選択できるように分類器を学習できる．2 節では DAGGER を説明する．

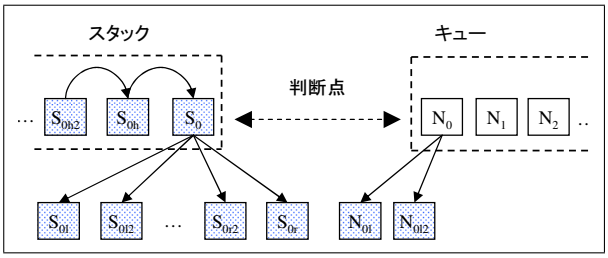

図 1: 決定的な Shift-reduce 依存構造解析の状態例．矢印は親 → 子の関係を示す．

模倣学習では，教師役として各状態に対して正しい行動を示す関数をオラクルと呼ぶ．NLP タスクへ模倣学習を適用するにはオラクルの設計が必要となる．3 節でシンプルな系列ラベリング，4 節で Shift-reduce 依存構造解析のためのオラクルを示す．5 節では系列ラベリングである固有表現抽出と英語 Shift-reduce 依存構造解析における実験により，正解行動列だけから学習した場合に比べて大幅な性能向上が確認できた．

2 Dataset Aggregation

方策を現在の状態 $s \in S$ から次の行動 $a \in A$ への写像 $\pi : S \to A$ と定義し，π が訪れた状態を s_π と書

Algorithm 1 DAGGER

初期化: $D \leftarrow \emptyset, \pi_1 \leftarrow \pi^*$
for $k = 1, 2, \cdots, K$ **do**
  π_k を実行し $D_k = \{(\boldsymbol{\phi}(s_{\pi_k}), \pi^*(s_{\pi_k}))\}$ を収集
  データを集約: $D \leftarrow D \cup D_k$
  D を用いて π_k を学習
end for
検証用データで性能の良い π_k を選択

く．一般的には，方策は分類器によって近似する．本研究では線形分類器 $\pi(s) = \mathrm{argmax}_a \boldsymbol{w}_a^\top \boldsymbol{\phi}(s)$ を用いる．ただし，$\boldsymbol{w}_a$ は行動 a に対するパラメータベクトル，$\boldsymbol{\phi}(s)$ は s に対する特徴関数である．また，正しい行動を返すオラクルを π^* と書く．

DAGGER は方策反復と呼ばれるアルゴリズムで，各反復でオラクルとそれまでに学習済みのすべての方策が訪れた状態の下で方策を学習する [10]．DAGGER の最初の反復では，既存の教師付き学習と同じようにオラクルが訪れた状態を訓練データとして方策を学習する．次の反復以降は，オラクルが訪れた状態に加えて学習した方策が訪れた状態も訓練データに加えて学習．簡略版の DAGGER をアルゴリズム 1 に示す．

方策が実際に訪れるであろう状態をサンプリングしその下での経験誤差を最小化するため，方策が選ぶ行動がオラクルと異なる，つまり過去に誤りがあった場合にもそれ以降の行動は誤らないことが期待できる．理論的にも，決定数（予測数）を T としたとき，事例間の独立性を仮定している教師付き学習は誤り伝播の影響を反映して誤差の上界は $O(T^2)$ であるが，DAGGER の誤差は $O(T)$ で抑えられることが示されている [10]．

DAGGER を適用するにはオラクルが必要になるが，教師データのある NLP のタスクではオラクル π^* はコーパスの正解アノテーションを入力としたルールとして実装することが可能である．以降では，系列ラベリングと Shift-reduce 依存構造解析に DAGGER を適用するためのオラクル（ルール）を示す．

3 系列ラベリング

系列ラベリングは，長さ T の入力列 $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T)$ に対して，ラベル列 $(y_1, y_2, \cdots, y_T)$ を予測する問題である．前から決定的に予測する場合には，点 t における状態 s_t は入力と $t-1$ までの予測ラベル列，$(\boldsymbol{x}, \hat{y}_{1:t-1})$，で表す．ただし，$y_{1:t-1} = (y_1, y_2, \cdots, y_{t-1})$ かつ $\hat{y}_t$ は点 t での方策の予測ラベル．系列ラベリングのためのオラクルは文献 [3] にあるように，状態とは独立にコーパスだけを参照して t の正解ラベルを返せばよい．つまり $\pi^*(s_t) = y_t$ である．

4 Shift-reduce 依存構造解析

決定的な Shift-reduce 依存構造解析手法のひとつである arc-eager 法 [9] について説明する．arc-eager 法は遷移システムでは次の 4 つの行動を定めている．

Left-Arc(c)**:** スタックから先頭 S_0 を取り出し，S_0 からキューの先頭 N_0 へラベル c の依存関係を作る；

Right-Arc(c)**:** キューの先頭 N_0 からスタックの先頭 S_0 へラベル c の依存関係を作り，N_0 をスタックに移動する；

Reduce**:** スタックの先頭 S_0 に親がいる場合に S_0 を取り出す；

Shift**:** キューの先頭 N_0 をスタックに移動する．

ただし，上記では文献 [14] と同様にラベル付きの依存関係を同定することを想定している．この遷移システムは E 個の入力トークンに対して最大 $2E-1$ 回の状態遷移で解析が終了することを保障する．

系列ラベリングと比較して，Shift-reduce 依存構造解析のオラクル設計には，過去の誤りによって正しい依存関係が作られる前に対象トークンがスタックから削除されてしまう可能性に注意が必要である．また，複数の行動列が同じ木構造を構成しうるため，Shift-reduce 依存構造解析の行動列はコーパスからは一意に特定できない点にも注意する必要がある．

$w_1 \cdots w_E$ を入力トークン列，$(w_i, c, w_j) \in L$ をラベル付き依存関係，L^* を正解の依存構造木の依存関係集合，L を方策 π_k による解析中の依存関係集合，スタックを N とする．本研究ではオラクルとして以下を提案する．

$$
\begin{cases}
\text{Left-Arc}(c) & \text{if } (N_0, c, S_0) \in L^* \\
\text{Right-Arc}(c) & \text{if } (S_0, c, N_0) \in L^* \\
\text{Reduce} & \text{if } \exists w_i, c \text{ s.t. } (w_i, c, S_0) \in L \wedge \\
 & \nexists w_j \in N, c' \text{ s.t. } (S_0, c', w_j) \in L^* \\
\text{Shift} & \text{if } \exists w_i, c \text{ s.t. } (w_i, c, S_0) \in L \\
 & \exists w_{i'} \in N, c' \text{ s.t. } (w_{i'}, c', S_0) \in L^* \\
a \in LS & \text{otherwise,}
\end{cases}
$$

ただし，LS はすべての c における Left-Arc(c) 行動と Shift 行動の集合である．最後の otherwise 節に到達した場合には，過去の誤りによって S_0 は真の修飾先

学習手法	精度	再現率	F 値
ロジスティック回帰（LR）	69.29	69.06	69.17
DAGGER w/ LR	77.49	77.01	77.25
SVM	68.05	69.79	68.91
DAGGER w/ SVM	77.93	**77.68**	**77.80**
CRF	**78.04**	77.09	77.56

表 1: CoNLL2002 固有表現抽出でのテスト性能比較

または修飾元との依存関係を作れる可能性が無い状態にある．その場合にオラクルは，Left-Arc(c) によって S_0 をスタックから除外するか，Shift によって S_0 を判断ポイントから外す行動をとる．ただし，ランダムに選ぶよりも方策を用いる方が実験的に良い性能を示したため，方策 π_k を用いて LS の中からひとつの行動を選ぶ．なお，本研究とは独立に文献 [4] でも誤りを考慮した arc-eager 法のためのオラクルが提案されている．

5 実験

5.1 固有表現抽出

本節では CoNLL2002 [11] のデータセットを使い固有表現抽出タスクにおける模倣学習の有効性を検証する．文献 [1] と同じ特徴テンプレートを使用した．決定的解析は長い履歴を特徴量に用いても解析時間への影響は小さいため，ラベルの予測履歴の特徴量は $1,2,3,4$ グラムを用いた（点 t において $\{\hat{y}_{t-1}, \hat{y}_{t-2}, \hat{y}_{t-3}, \hat{y}_{t-4}, \hat{y}_{t-1:t-2}, \hat{y}_{t-1:t-3}, \hat{y}_{t-1:t-4}\}$）．方策の実装にはロジスティック回帰（LR）および多クラス SVM [2] を用い，開発セットを用いて π_k を選択した．また，比較対象として動的計画法を用いた全域学習手法である 1 次の CRF を用いた．なお，DAGGER と CRF とも正則化パラメータは開発セットを用いて選択した．

表 1 に実験結果を示す．LR と SVM の行は DAGGER の反復を行わなかったときの結果であり，DAGGER w/ LR と DAGGER w/ SVM の行はそれぞれ DAGGER の反復を行った結果である．LR と SVM とも DAGGER を用いることで F 値が 8 ポイント向上した．長い予測履歴を特徴に使うことで誤り伝播が起こりやすい事が予想されるが，模倣学習によって長い予測履歴を特徴に使っても誤り伝播を回避できていると言える．CRF との比較では，広範囲の特徴が利用できることで，動的計画法による全域最適化と同等の性能を決定的解析でも達成できた．特に，DAGGER w/ SVM の F 値と再現率は CRF を若干上回った．

学習手法	ビーム幅	UAS	LAS	UEM
SVM	1	88.9	87.6	34.2
DAGGER w/ SVM	1	90.0	88.7	34.7
DAGGER w/ SVM	2	90.5	89.3	37.2
パーセプトロン	1	88.6	87.1	34.0
パーセプトロン	2	90.3	88.9	40.0
パーセプトロン	64	92.8	91.7	48.0

表 2: 英語依存構造解析のテスト性能比較

5.2 英語依存構造解析

本節では英語の依存構造解析タスクにおいて模倣学習の有効性を確認する．Penn Treebank III（PTB）[7] を文献 [13] で提案されたルールにより依存構造木に変換した．Left-Arc と Right-Arc 行動で決定する依存ラベルは 12 種であり，行動の種類数 $|A|$ は 26 となった．PTB は標準的な評価方法に従ってセクションごとに，訓練：2 ～ 21，開発：22，テスト： 23 に分割して使用した． 品詞は CRF によって付与した（PTB での精度 97.3%）．ただし，訓練データに対する品詞は文献 [14] と同様に 10 分割ジャックナイフ法によって付与した．

特徴テンプレートは文献 [14] と同じものを用いた．DAGGER の方策の実装には多クラス SVM [2] を用いた．なお，計算量の観点から SVM の学習には平均化確率的勾配法 [12] を使用し，停止条件および正則化パラメータの選択には開発セットを用いた．また，比較対象としてビーム探索を用いた全域学習手法である平均化した構造化パーセプトロンも評価した [5]. DAGGER および構造化パーセプトロンとも停止条件には開発セットを用い，決定的な解析だけでなくビーム探索を用いた実装も評価した．評価には，テストセットでのトークン単位での評価，ラベルなし依存関係正答率（UAS ; unlabeled attachment scores）およびラベルあり依存関係正答率（LAS ; labeled attachment scores），と文単位での評価，ラベルなし完全一致率（UEM ; unlabeled exact match)，を用いた．ただし，先行研究に倣い，品詞が {“ ” : , .} のいずれかであるトークンは評価から除外した．

表 2 に各手法のテスト性能を示す．パーセプトロンの行は構造化パーセプトロンの評価結果である．ビーム幅列は決定的解析（$b = 1$）とビーム幅 $b = 2$ を示すが，構造化パーセプトロンは文献 [14] と同等の性能が出ていることを示すためにビーム幅 $b = 64$ の結果も示す．

決定的解析での SVM との性能比較では，DAGGER を用いることで UAS/LAS とも 1 ポイントほど上昇し，その有効性が確認された．一方で木構造全体の予測を

	CPU	言語	UAS	速度	計算量
SVM b=1	2.8GHz	Java™	88.9	1408	$O(E)$
DAGGER b=1	2.8GHz	Java™	90.0	1310	$O(E)$
DAGGER b=2	2.8GHz	Java™	90.5	637	$O(bE)$
文献 [8]	3.2GHz	Java™	90.2	8	$O(E^2)$
文献 [6]	3.2GHz	Python	92.1	25	$O(bE)$
文献 [14]	2.0GHz	C++	92.9	29	$O(bE)$

表 3: 先行研究との比較．速度は文数／秒．文献 [8] と文献 [6] の値は文献 [6] で報告された値，文献 [14] の値は文献 [14] で報告された値．

評価する UEM の向上は比較的小さかった．これは，DAGGER が誤り伝播を回避する一方で構造全体を正答するようには学習していないことと合致する．また，ビーム探索と DAGGER を併用することでもさらに性能向上が確認された．構造化パーセプトロンとの比較では，ビーム幅 $b = 1, 2$ では DAGGER が UAS/LAS の値で上回った．この結果より全域学習が推定の近似性能に強く依存していることがわかる．ただし，UEM は $b = 2$ のときに構造化パーセプトロンが DAGGER の結果を上回った．これは，構造全体を正答するように学習する全域学習の特徴を反映した結果であろう．

表 3 は SVM と DAGGER の決定的解析（$b = 1$），ビーム幅 $b = 2$ の DAGGER，および先行研究で報告された性能・解析速度を示す．先行研究としてあげた文献 [8] は Graph-base 法，文献 [6]，文献 [14] は Transition-base 法である．ここで注目したいのは提案法の解析速度である．実験環境および実装が異なるため直接の比較は困難だが，決定的解析である SVM と DAGGER（$b = 1$）は既存手法に比べて桁違いの速度を示した．また，方策反復に伴って特徴次元が増えることによる速度低下の影響は小さく，SVM と DAGGER との間で大きな速度低下は見られなかった．一方，ビーム幅 $b = 2$ のビーム探索を用いた DAGGER の解析速度は決定的解析に比べて半減した．これは，行動の選択（分類）が解析時間のほとんどを占めるため，ビーム幅に比例して分類回数が増え，それに反比例して解析速度が下がってしまうことを示している．

6 おわりに

全域学習手法と比較したときの模倣学習の利点は特徴量の自由度と解析速度にある．5.1 節で示したように広範囲の特徴量を用いても誤り伝播を起こすことがないため全域学習の性能を上回る可能性がある．DAGGER はマルコフ性を仮定していないため，たとえば系列ラベリングで予測対象の単語にこれまで付与されたラベルの分布を使うなど，さらに広範囲の情報を特徴に使うことも可能になる．また，5.2 節で示したように模倣学習によって高速な解析速度のまま精度を向上させることが可能である．

模倣学習は，大域学習が実質的に適用できない広範囲の特徴が必要となるタスクや大規模データの処理で有望な学習手法である．しかし，オラクルによって性能が左右される面があり，さまざまなタスクに対する最適なオラクルの設計方法は今後の課題である．

参考文献

[1] Yasemin Altun, Mark Johnson, and Thomas Hofmann. Investigating loss functions and optimization methods for discriminative learning of label sequences. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 145–152, 2003.

[2] Koby Crammer and Yoram Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, Vol. 2, pp. 265–292, 2002.

[3] Hal Daumé III, John Langford, and Daniel Marcu. Searn in practice. 2006.

[4] Yoav Goldberg and Joakim Nivre. A dynamic oracle for arc-eager dependency parsing. In *Proceedings of the 24th International Conference on Computational Linguistics*, 2012.

[5] Liang Huang, Suphan Fayong, and Yang Guo. Structured perceptron with inexact search. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 142–151, 2012.

[6] Liang Huang and Kenji Sagae. Dynamic programming for linear-time incremental parsing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1077–1086, 2010.

[7] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, Vol. 19, No. 2, pp. 313–330, 1993.

[8] Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pp. 523–530, 2005.

[9] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies*, pp. 149–160, 2003.

[10] Stéphane Ross, Geoffrey J. Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, pp. 627–635, 2011.

[11] Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2002*, pp. 155–158, 2002.

[12] Wei Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *CoRR*, Vol. abs/1107.2490, , 2011.

[13] Hiroyasu Yamada and Yuji Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of the 8th International Workshop on Parsing Technologies*, Vol. 3, pp. 195–206, 2003.

[14] Yue Zhang and Joakim Nivre. Transition-based dependency parsing with rich non-local features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pp. 188–193, 2011.