大域的素性を用いたタグ付けのためのパーセプトロン学習

風間 淳一 (kazama@jaist.ac.jp)　鳥澤 健太郎 (torisawa@jaist.ac.jp)
北陸先端科学技術大学院大学 情報科学研究科

1 概要

CRF などの現在主流となっているタグ付け手法では，計算量の問題から大域的な素性を考慮することが困難である．そこで，本研究では，大域的な素性を用いたタグ付けのための新しいパーセプトロン学習法を提案する．従来の手法とは異なり，提案手法では値が単語列とタグの列から決まるものならどのような大域的素性でも扱うことができる．そして，局所的素性と大域的素性の重みは，収束性の保証された学習法により同時に学習される．実験では，CoNLL 2003 shared task の固有表現認識データを用いて，提案手法の性能を調べる．

2 背景

近年，品詞タグ付け・チャンキング・固有表現認識など，多くの自然言語処理が単語列に対するタグ付けとして解かれている．現在，その高い精度から主流になっているのが，CRF (Lafferty et al. 2001) や，パーセプトロン (Collins 2002) に代表される discriminative モデルと呼ばれる手法である．これらの手法の利点は，互いに独立でない様々な素性を用いることができることにあるが[*1]，これらの手法には，ごく少数のタグのみ (通常，現在のタグと一つ前のタグ) に依存するような「局所的」な素性しか用いることができないという制限がある．この制限を課すことで, forward-backward や Viterbi アルゴリズムなどの効率的な計算法が可能になり，学習やタグ付けが現実的な計算量で可能になるからである．

一方, 精度をさらに向上させるための一つの手法として，列に対する全てのタグが決まってからその値が分かる「大域的」な素性を利用することが提案されている (Finkel et al. 2005; Roth and Yih 2005; Krishnan and Manning 2006)．例えば，固有表現認識では，「同一の文書内の同じ単語列は（それが固有表現ならば）異なる固有表現クラスを持つことはない」といった大域的素性が有効であることが示されている (Finkel et al. 2005; Krishnan and Manning 2006)．

本研究では，これまでの局所的素性に加えてこのような大域的素性を扱うことのできる新しいパーセプトロンを提案する．上で述べたように，大域的素性を用いるための手法はいくつか提案されている (Finkel et al. 2005; Roth and Yih 2005; Krishnan and Manning 2006)．しかし，これらの手法では，用いることのできる大域的素性の種類が多かれ少なかれ制限されている．例えば，Finkel et al. (2005) ではタグ付け時の計算量の問題を Gibbs sampling を用いることにより解決している．しかし，提案されている大域的素性の重みを求める手法を論文中の大域的素性以外の大域的素性にどのように適用するかは明らかではない．Krishnan and Manning (2006) はモデルを二つの CRF に分け，局所的素性のみを用いる第一の CRF の出力を第二の CRF で用いることで，「第一の CRF の出力中で一番多い固有表現クラス」といった大域的情報の利用を可能にしている．しかし，このモデルではタグ付け候補そのものに依存するような一般の大域的素性を用いることはできない．また，Roth and Yih (2005) では出力タグ列に対する制約を integer linear programming という手法を使って可能にしている．しかし，これは重みが負の無限大に固定された大域的素性のみを用いることと同等である．

本研究では，これらの研究とは異なり，値が単語列とタグの列から決まるようなものならどのような大域的素性でも扱うことができ，その適切な重みを学習コーパスから学習することができる手法の開発をめざす．

3 提案手法の概要

提案手法は，Collins (2002) のタグ付けに対するパーセプトロンを基にし，大域的素性を扱えるように拡張したものである．大域的素性の存在により，Collins (2002) で必要な Viterbi アルゴリズムによる最良解の発見が不可能になる．そこで，提案手法では，局所的素性のみの重みの下での n-best 解を A*アルゴリズムで効率的に求め，その n-best 解に対してのみ全体でのスコアを計算し，素性の重みの更新に利用すべき候補解を発見するという方法で計算量の問題を解決する．加えて，その n-best 解が十分正しい解になるように，局所的素性の重み部分を学習中適時更新する．提案手法では，Collins (2002) のように厳密な最良解を用いることができないが，それでも学習の収束性は保証される．

また，最終的なアルゴリズムでは，過学習の問題に対処するため，マージンパーセプトロン (Krauth and Mézard 1987) をタグ付けに対して拡張したものを用い，加えて，初期値の与え方として，Bayes Point Machine (Herbrich and Graepel 2000) の考えを取り入れる．

4 タグ付けのためのパーセプトロン

タグ付けにおける目的は，列中の各単語 $x_i \in \mathcal{X}$ にタグ $y_i \in \mathcal{Y}$ を割り当てることである．本論文では，以後，列 $x_1, \ldots, x_T$ を $\boldsymbol{x}$ と表し，対応するタグ列を $\boldsymbol{y}$ で表すことにする．Collins (2002) は，二値分類のためのものであったパーセプトロン (Rosenblatt 1958) をタグ付けを含む構造出力に対して拡張した．Collins (2002) の方法では，$(\boldsymbol{x}, \boldsymbol{y})$ から素性ベクトル $\Phi(\boldsymbol{x}, \boldsymbol{y}) = (\Phi_1(\boldsymbol{x}, \boldsymbol{y}), \cdots, \Phi_d(\boldsymbol{x}, \boldsymbol{y})) \in R^d$ への写像と重みベクトル $\boldsymbol{\alpha} \in R^d$ があると仮定する．そして，列 $\boldsymbol{x}$ が与えられた時のタグ列を，

$$\boldsymbol{y}' = \mathrm{argmax}_{\boldsymbol{y} \in \mathcal{Y}^{|\boldsymbol{x}|}} \Phi(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha} \quad (1)$$

によって決定する．ただし，$\cdot$ はベクトルの内積である．学習の目的は，与えられた学習データ $\{(\boldsymbol{x}_1, \boldsymbol{y}_1^*), \cdots, (\boldsymbol{x}_L, \boldsymbol{y}_L^*)\}$ から，上の規則で正しいタグ列が得られるような適切な重みベクトル $\boldsymbol{\alpha}$ を求めることである．学習は，まず，重みベクトルをゼロで初期化し，次に，学習データを順に取り出し，その時点の重みを用いて上の規則でタグ列 $\boldsymbol{y}'$ を得る．もし，$\boldsymbol{y}'$ が正解 $\boldsymbol{y}^*$ と異なる場合には，

$$\boldsymbol{\alpha}_{new} = \boldsymbol{\alpha} + \Phi(\boldsymbol{x}, \boldsymbol{y}^*) - \Phi(\boldsymbol{x}, \boldsymbol{y}').$$

という更新則にしたがって更新する．学習データを一巡しても更新が起こらなくなった時点で学習が終了する．この学習アルゴリズムは，収束することが証明されており (Collins 2002), 例えば, 学習データがマージン δ で分離可能で, $\forall i, \forall \boldsymbol{y} \in \mathcal{Y}^{|\boldsymbol{x}_i|}$ $||\Phi(\boldsymbol{x_i}, \boldsymbol{y_i}^*) - \Phi(\boldsymbol{x_i}, \boldsymbol{y})|| \leq R$ とすると，R^2/δ^2 回以下の更新で学習が終了する．[*2]

[*1] 単語そのものとサフィックスと同時に使うなど.

[*2] (Collins 2002) では分離不可能な場合の収束の証明も与えているが，本論文では，分離可能な場合のみ扱うこととする.

Algorithm 5.1: タグ付けのためのマージンパーセプトロン (パラメータ: C)

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$
until no more updates **do**
  for $i \leftarrow 1$ **to** L **do**
    $\boldsymbol{y}' = \mathrm{argmax}_{\boldsymbol{y}} \Phi(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    $\boldsymbol{y}'' = \text{2nd-best}_{\boldsymbol{y}} \Phi(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    if $\boldsymbol{y}' \neq \boldsymbol{y_i}^*$ **then**
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi(\boldsymbol{x}, \boldsymbol{y}^*) - \Phi(\boldsymbol{x}, \boldsymbol{y}')$
    else if $\Phi(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi(\boldsymbol{x}_i, \boldsymbol{y}'') \cdot \boldsymbol{\alpha} \leq C$ **then**
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi(\boldsymbol{x}, \boldsymbol{y}^*) - \Phi(\boldsymbol{x}, \boldsymbol{y}'')$

5 タグ付けのためのマージンパーセプトロン

本研究では，当初，前節の Collins のパーセプトロンをそのまま拡張しようとしたが，そのままでは，CRF に比べてベースラインとして大幅に劣ることが分かった．Collins (2002) では，過学習の問題に対処するため，学習途中の全ての重みベクトルの平均を最終的な重みベクトルとする多数決パーセプトロン（voted perceptron）も提案されており，性能の改善が報告されている．しかし，予備実験から，多数決パーセプトロンを用いても CRF に比べてかなり劣る性能しか得られないことが分かった．そこで，本研究では，過学習を低減する他の方法として知られるマージンパーセプトロン (Krauth and Mézard 1987) を利用することを試みた．マージンパーセプトロンでは，学習データが分離できた時点で学習を止めてしまうのではなく，重みベクトルによるマージンを真のマージン δ に近づけるようさらに学習を行なう．マージンを大きくすることは，SVM (Cortes and Vapnik 1995) などでも示されているように，過学習を防ぐ有効な手段である．

本研究では，Collins (2002) が通常のパーセプトロンを構造出力に拡張したように，マージンパーセプトロン (Krauth and Mézard 1987) を構造出力に拡張した．

まず，構造出力の場合のマージンを次のように定義できる[*3]．

$$\gamma(\boldsymbol{\alpha}) = \min_{\boldsymbol{x}_i} \min_{\boldsymbol{y} \neq \boldsymbol{y}_i^*} \frac{\Phi(\boldsymbol{x_i}, \boldsymbol{y_i}^*) \cdot \boldsymbol{\alpha} - \Phi(\boldsymbol{x_i}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}}{||\boldsymbol{\alpha}||}$$

これは，収束後に最良解が正解と一致するとして，最良解の次にスコアの高い第二解を $\boldsymbol{y}'' = \text{2nd-best}_{\boldsymbol{y}} \Phi(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$ とすると，

$$= \min_{\boldsymbol{x}_i} \frac{\Phi(\boldsymbol{x_i}, \boldsymbol{y_i}^*) \cdot \boldsymbol{\alpha} - \Phi(\boldsymbol{x_i}, \boldsymbol{y''}) \cdot \boldsymbol{\alpha}}{||\boldsymbol{\alpha}||}.$$

と書きかえることができる．この関係から，Algorithm 5.1 の構造出力に対するマージンパーセプトロンが得られる．

Algorithm 5.1 から分かるように，このアルゴリズムは Collins のパーセプトロンに「$\Phi(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi(\boldsymbol{x}_i, \boldsymbol{y}'') \cdot \boldsymbol{\alpha} \leq C$ をチェックし，必要なら更新を行なう」という処理を付け加えただけであり，実装は非常に簡単である．また，Collins のパーセプトロンと同じような証明で，学習が $(R^2 + 2C)/\delta^2$ 以内の更新で終了し，そのとき，上で定義したマージンが $\gamma(\boldsymbol{\alpha}) \geq \delta C/(R^2 + 2C) = (\delta/2)(1 - (R^2/(R^2 + 2C)))$ となることが証明できる（紙面の都合上省略）．これから分かる通り，アルゴリズム中のパラメータ C がマージンと学習時間のトレードオフを制御する．

Collins のパーセプトロンやこのアルゴリズムでは，最良解や第二解などを効率的に発見する必要がある．素性が局所的素性のみの場合には，最良解は Viterbi アルゴリズムを用いることで，また，第二解（より一般的に n-best 解）は A*アルゴリズムを用いることで効率的に求めることができる．この点では，計算量の観点から局所的素性しか用いることのできない CRF と同じであるが，重要な点は，全てのタグ付け候補を考慮して期待値を計算する CRF とは違い，高々，最良解と第二解さえ求まれば収束性のある学習が行なえる点である．

[*3] このようなマージンの定義は (Taskar et al. 2003) などに見られる．

Algorithm 6.1: 候補アルゴリズム (パラメータ: C)

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$
until no more updates **do**
  for $i \leftarrow 1$ **to** L **do**
    $\{\boldsymbol{y}^n\} = \text{n-best}_{\boldsymbol{y}} \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    $\boldsymbol{y}' = \mathrm{argmax}_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}} \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    $\boldsymbol{y}'' = \text{2nd-best}_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}} \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
    if $\boldsymbol{y}' \neq \boldsymbol{y_i}^*$ & $\Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}') \cdot \boldsymbol{\alpha} \leq C$ **then**
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^a(\boldsymbol{x}, \boldsymbol{y}^*) - \Phi^a(\boldsymbol{x}, \boldsymbol{y}')$
    else if $\Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}'') \cdot \boldsymbol{\alpha} \leq C$ **then**
      $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^a(\boldsymbol{x}, \boldsymbol{y}^*) - \Phi^a(\boldsymbol{x}, \boldsymbol{y}'')$

6 提案アルゴリズム

以上，基になるパーセプトロンを説明したので，次に，本研究で提案する局所的素性と大域的素性の重みを同時に学習するパーセプトロンについて説明する．

まず，局所的素性のベクトルを $\Phi^l(\boldsymbol{x}, \boldsymbol{y})$，大域的素性のベクトルを $\Phi^g(\boldsymbol{x}, \boldsymbol{y})$ で表す．また，これらは

$$\Phi^l(\boldsymbol{x}, \boldsymbol{y}) = (\Phi_1^l(\boldsymbol{x}, \boldsymbol{y}), \cdots, \Phi_n^l(\boldsymbol{x}, \boldsymbol{y}), 0, \cdots, 0)$$
$$\Phi^g(\boldsymbol{x}, \boldsymbol{y}) = (0, \cdots, 0, \Phi_{n+1}^g(\boldsymbol{x}, \boldsymbol{y}), \cdots, \Phi_d^g(\boldsymbol{x}, \boldsymbol{y}))$$

という形をしているとする．そうすると，全体の素性ベクトルは $\Phi^a(\boldsymbol{x}, \boldsymbol{y}) = \Phi^l(\boldsymbol{x}, \boldsymbol{y}) + \Phi^g(\boldsymbol{x}, \boldsymbol{y}) = (\Phi_1^l(\boldsymbol{x}, \boldsymbol{y}), \cdots, \Phi_n^l(\boldsymbol{x}, \boldsymbol{y}), \Phi_{n+1}^g(\boldsymbol{x}, \boldsymbol{y}), \cdots, \Phi_d^g(\boldsymbol{x}, \boldsymbol{y}))$ となる．

理想的には，式 (1) のようにして全体の素性の下での最良解を求めたいが，大域的素性が存在するためそれは不可能である．そこで，条件を緩めて

$$\{\boldsymbol{y}^n\} = \text{n-best}_{\boldsymbol{y}} \Phi^l(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$$
$$\boldsymbol{y}' = \mathrm{argmax}_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}} \Phi^a(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha} \qquad (2)$$

という形のモデルの中から適切なモデルを見つけることにする．つまり，まず，局所的モデル $\Phi^l(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$ の下で n-best 解を生成し（これは，前述の通り A*アルゴリズムで効率的に行なえる），次に，その n-best 解についてのみ全体のモデル $\Phi^a(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$ の下でスコアを求めて，その中で最良の解を求める（これも，n が適切な大きさならば，現実的な時間で行なうことができる）．これは，re-ranking の手法と類似しているが，局所的モデル $\Phi^l(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$ と大域的素性を含んだ全体モデル $\Phi^a(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$ が重みの一部を共有しているため，学習中に相互作用を及ぼす点が大きく異なる．re-ranking では，第一の局所的素性のみのモデルの学習と第二の大域的素性を含んだモデルの学習には，異なる学習コーパスを用いたり，交差検定法的に学習するなどの工夫が必要であり，有効な学習コーパスの量が少なくなったり，学習時間が増加するといった問題があるが，提案手法ではそのような問題はない．

このアプローチでは，もはや厳密な最良解や第二解を求めることはできないため，節 4 や節 5 で説明した学習アルゴリズムを用いることはできないように思われるかもしれない．しかし，(Collins 2002) の証明を詳しく見ると，収束性のための本質的な条件は，厳密な最良解や第二解を用いることではなく，$\Phi(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha} \leq 0$ （マージンパーセプトロンの場合 $\leq C$）となるような $\boldsymbol{y}$ $(\neq \boldsymbol{y}^*)$ を用いて更新を行なうことであることが分かる．すなわち，式 (2) を用いる場合でも，少なくとも収束性のある学習アルゴリズムを構成することができる．

以上の考察から，我々が最初に考案したアルゴリズムは，Algorithm 6.1 に示したものである．これを「候補アルゴリズム」と呼ぶことにする．このアルゴリズムでは，学習が終わった時には $\boldsymbol{y} \in \{\boldsymbol{y}^n\}, \boldsymbol{y} \neq \boldsymbol{y}^*$ であるような全ての $(\boldsymbol{x}_i, \boldsymbol{y})$ につ

Algorithm 6.2: 大域的素性のためのパーセプトロン (パラメータ: n, C^a, C^l)

$\boldsymbol{\alpha} \leftarrow \mathbf{0}$
until no more updates **do**
  for $i \leftarrow 1$ **to** L **do**
$\{\boldsymbol{y}^n\} = \text{n-best}_{\boldsymbol{y}} \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
$\boldsymbol{y}' = \text{argmax}_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}} \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
$\boldsymbol{y}'' = \text{2nd-best}_{\boldsymbol{y} \in \{\boldsymbol{y}^n\}} \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$
if $\boldsymbol{y}' \neq \boldsymbol{y}_i{}^*$ & $\Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}') \cdot \boldsymbol{\alpha} \leq C^a$ **then**
  $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^a(\boldsymbol{x}, \boldsymbol{y}^*) - \Phi^a(\boldsymbol{x}, \boldsymbol{y}')$ (A)
else if $\Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}'') \cdot \boldsymbol{\alpha} \leq C^a$ **then**
  $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^a(\boldsymbol{x}, \boldsymbol{y}^*) - \Phi^a(\boldsymbol{x}, \boldsymbol{y}'')$ (A)
else
(B) **if** $\boldsymbol{y}^1 \neq \boldsymbol{y}_i{}^*$ **then**
  $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^l(\boldsymbol{x}, \boldsymbol{y}^*) - \Phi^l(\boldsymbol{x}, \boldsymbol{y}^1)$
else if $\Phi^l(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^l(\boldsymbol{x}_i, \boldsymbol{y}^2) \cdot \boldsymbol{\alpha} \leq C^l$ **then**
  $\boldsymbol{\alpha} = \boldsymbol{\alpha} + \Phi^l(\boldsymbol{x}, \boldsymbol{y}^*) - \Phi^l(\boldsymbol{x}, \boldsymbol{y}^2)$

いて $\Phi^a(\boldsymbol{x}_i, \boldsymbol{y}_i^*) \cdot \boldsymbol{\alpha} - \Phi^a(\boldsymbol{x}_i, \boldsymbol{y}) \cdot \boldsymbol{\alpha} > C$ が成り立つ．これは，一見，十分な条件に見えるが，実はそうではない．学習が終わった後でも，$\boldsymbol{y}^* \notin \{\boldsymbol{y}^n\}$ の場合には，式 (2) のタグ付けでは正しい解 $\boldsymbol{y}^*$ を見つけられないからである．実際，この候補アルゴリズムでは学習がうまくいかないことを実験で示す．

つまり，局所的モデル $\Phi^l(\boldsymbol{x}, \boldsymbol{y}) \cdot \boldsymbol{\alpha}$ は，少なくとも $\boldsymbol{y}^* \in \{\boldsymbol{y}^n\}$ となる程度に良いモデルである必要がある．これを達成するため，我々は上の候補アルゴリズムに，「全体のモデルが十分良い場合でも，局所的モデルがまだ悪い場合には，局所的素性の重みを更新する」という処理を付け加えた．

最終的なアルゴリズムは，Algorithm 6.2 に示したものになる．(B) が付け加えられた部分である．このアルゴリズムでは，局所的モデルの更新 (B) よりも，全体モデルの更新 (A) のほうが優先されている．これは，局所的モデルの更新を優先させると，全体モデル（つまり大域的素性の重み）の更新がほとんど起こらなくなってしまうからである．また，局所的モデルの更新に関しては，局所的モデルが正解を出力しマージンの条件を満たすまで更新を続けるようになっている．局所的モデルによる n-best 解に正解が含まれるようになった時点で更新を止めることも考えられたが，上記の方が精度が良いことが分かったためである．

パラメータ C^l, C^a については，本研究では，探索領域を減らすため $C^l = C^a = C$ と仮定した．また，多少複雑になるが，これまで説明したパーセプトロンと同様に，このアルゴリズムの収束性を示すことができる（付録 A を参照）．

これまで，重みベクトルはゼロで初期化されるとしてきたが，実際にはどのような初期値を与えても収束することを示せる．そこで，本研究では過学習に対する頑健性をさらに上げるため Bayes Point Machine (BPM)(Herbrich and Graepel 2000) の考えを取り入れた初期化を用いる．元々の BPM は，学習データの順番をランダムに変えて複数回学習を行い，その平均を最終的な重みベクトルとすることで，過学習を抑えることを狙っている[*4]．しかし，学習を何度も行なうのはコストが大きいため，本研究では，アドホックな方法として，学習データを一巡するだけの学習を順番をランダムに変えて複数回行ない，そのベクトルの平均を，初期値とするようにした (BPM 初期化)．これは，学習の初期段階において学習データの順番による影響が特に大きいと考えたためである．

7 実験

提案手法の性能を，CoNLL 2003 shared task (Tjong et al. 2003) の英語固有表現認識のデータを用いて調べた．このデータでは，PER・ORG・LOC・MISC の 4 種類の固有表現がタグ付けされており，学習セット 14,987 文, 開発セット 3,466 文，評価セット 3,684 文が含まれる．また，POS・チャンクタグ，各クラスの gazetteer が提供されている．

表 1　性能比較のまとめ (F_1).

	開発セット	評価セット	σ^2/C
局所的素性のみ			
CRF	91.10	86.26	100
パーセプトロン	89.01	84.03	-
多数決パーセプトロン	89.32	84.08	-
マージンパーセプトロン	90.98	85.64	11313
＋大域的素性			
候補アルゴリズム ($n' = 100$)	90.71	84.90	4000
提案アルゴリズム ($n' = 100$)	**91.95**	**86.30**	5657

このデータを用いて，局所的素性のみを用いた時の CRF・パーセプトロン・多数決パーセプトロン・マージンパーセプトロンの性能，また，大域的素性を用いた時の候補アルゴリズム・提案アルゴリズムの性能を比較する．

IOB2 タグを採用し，局所的素性としては，前後 2 単語の，単語・小文字化した単語・POS タグ・チャンクタグ・prefix・suffix・単語の表層タイプ (Bikel et al. 1999)・gazetteer とのマッチ（IOB2 タグで表現），を用いた．これらと，現在位置のタグとの組合わせ，直前・現在位置のタグとの組合わせを用いた．また，小文字化した単語・POS タグ・チャンクタグ・gazetteer についてはその bigram も用いた．

大域的素性としては，Finkel et al. (2005), Krishnan and Manning (2006) を参考にして，文書内の同一単語列の固有表現クラスの一致，文書内の部分一致単語列の固有表現クラスの一致，文書内の同一単語列での最多固有表現クラスとの一致などの素性を実装した．加えて，上記の既存研究では用いられていないものとして，並列表現内での固有表現クラスの一致を見る素性も実装した[*5]．現時点では，各素性の効果の詳しい分析は行なっていないので，実装の詳細は省く．CoNLL データ中の文書の区切りマーカーにしたがい，同一文書中の文を特別な単語をはさんで連結した．これが，アルゴリズム中の $\boldsymbol{x}$ となる．連結の結果，学習セット 964 文書, 開発セット 216 文書，評価セット 231 文書が得られた．

実装は，CRF++ (ver 0.44)[*6]をベースにした．ただし，今回の実験では，連結により語数が非常に大きくなり数値的な問題が起きる可能性があったため，CRF の forward-backward については HMM で用いられるような scaling を実装した[*7]．また，都合により，最適化モジュールは Amis[*8]で用いられているものを代わりに用いた．

CRF の学習では，Gaussian 正規化を行い，パラメータ σ^2 は開発セットを用いて調整した．マージンパーセプトロンの C も開発セットを用いて調整した[*9]．学習の収束判定は，CRF については，log-likelihood の値を見て判定した．パーセプトロンの場合，理論的には更新がなくなった時点が収束であるが，現実的にはそこまで行かないので，タグのエラー率を見て判定することにした．いずれも，これらの値の相対的な変化量が 3 反復の間 0.0001 を下回った場合に収束したと見なし，学習を終了した．

提案手法では，n-best の n がもう一つのパラメータになる．本実験では，学習に関しては，経験上実用的な学習時間になる $n = 20$ を用いた．一方，タグ付け時の n（これを n' とする）については，タグ付け自体の時間は実験のごく一部なので，より大きな n' を試すことができる．普通には, $n = n'$ と

[*4] パーセプトロンの学習結果は学習データの順番によって変わる．

[*5] "... in U.S, EU, and Japan" のように並列される固有表現は同一クラスになることが多いことをとらえるため．

[*6] http://chasen.org/~taku/software/CRF++

[*7] 局所的素性のみの場合，本来連結は必要ないが，CRF の予備的実験では連結によって精度が下がるということはなかった（むしろ上昇傾向にあった）ので実験は全て連結したデータで行なった．

[*8] http://www-tsujii.is.s.u-tokyo.ac.jp/amis

[*9] この実験では，$\sigma^2 = \{13, 25, 50, 100, 200, 400, 800\}$, $C = \{500, 1000, 1414, 2000, 2828, 4000, 5657, 8000, 11313, 16000, 32000\}$ という値をテストした．

表 2　n' の効果.

	開発セット	評価セット	C
$n'=20$	91.76	86.19	5657
$n'=100$	91.95	86.30	5657
$n'=400$	92.13	86.39	5657
$n'=800$	92.09	86.39	5657
$n'=1600$	92.13	**86.46**	5657
$n'=6400$	**92.19**	86.38	5657

するのが良いと思われるが，予備実験から，n' が n よりも大きい方が精度が高くなることが分かった．そこで，実験では，$n'=100$ を用いた．その後，C を最適な値に固定して n' を変化させる実験も行なった．

表 1 が各手法の比較結果である[*10]．まず，局所的素性のみの精度は CRF がパーセプロトンを大きく上回った．多数決パーセプトロンにより，精度は多少向上するが，その差は小さい．それに比べて，マージンパーセプトロンによる精度向上はかなり大きい．しかし，CRF が最も優れてい点は変わらなかった．次に，大域的素性の効果をみると，提案手法によってマージンパーセプトロンと比較して F 値で 0.66 向上していることが分かる．また，候補アルゴリズムの精度を見ると，精度は大幅に低く，大域的素性の学習のためには Algorithm 6.2 で加えた変更が非常に重要であったことが分かる．

次に，表 2 に n' の効果をまとめた．これを見ると，学習時の n よりもかなり大きい n' で最高精度を達成していることが分かる．この現象の詳細な理由は不明であるが，学習データと評価データに対する n-best の傾向が（過学習のため）食い違ってしまうため，評価時には余裕を持たせて候補を生成する必要がある，ということかもしれない．また，実用を考えると $n'=6400$ では重すぎるため，今後は，この実験結果をふまえて学習アルゴリズムの改良などを行なっていきたい．

次に，BPM 初期化の効果は，提案手法の場合，BPM 初期化無しでは期待通り精度が 91.89（開発）/86.03（評価）に低下した．しかし，マージンパーセプトロンの場合 90.98（開発）/85.90（評価）と逆に上昇してしまった．これは，小さな素性セットを用た予備実験とは食い違う結果で，原因を今後より詳しく調査していきたい．

大域的素性の効果という観点では，提案手法は表 3 に示した関連研究と比較して，上がり幅・最終的な精度ともに及んでない．しかし，最後の実験として，条件を少し変えた実験を行なった．具体的には，元々提供されている POS・チャンクタグの代わりに，より精度の高いものとして，TagChunk (Daumé III and Marcu 2005)[*11]で付与したタグを用いるというものである．表 4 に結果を示す．期待した通り，前の実験に比べて全ての手法で精度が向上している．これは，POS・チャンクタグの品質によるものと考えられる．ここで興味深い点は，ベースラインの精度が上昇しているにもかかわらず，提案手法による大域的素性の効果が 0.93 に上昇していることである．また，ベースラインの違い，素性セットの違いから公平な比較ではないが，最終的な精度も (Finkel, Grenager, and Manning 2005) より高く，(Krishnan and Manning 2006) に迫る程度になっている．

8　まとめ

本研究では，大域的素性を用いたタグ付けのための新しいパーセプトロンを提案した．このアルゴリズムはどのような大域的素性でも用いることができ，局所的素性と大域的素性の重みを同時に学習することができる．今後は，このアルゴリズムを基に，様々な種類の新しい大域的素性の調査を行ないたいと考えている．

[*10] CRF, パーセプトロン, 多数決パーセプトロンには BPM による初期化は用いていない．

[*11] http://www.cs.utah.edu/~hal/TagChunk/

表 3　既存研究の精度

	開発セット	評価セット	σ^2/C
(Finkel, Grenager, and Manning 2005)			
CRF（ベースライン）	-	85.51	-
＋ 大域的素性	-	86.86	-
(Krishnan and Manning 2006)			
CRF（ベースライン）	-	85.29	-
＋ 大域的素性	-	87.24	-

表 4　POS・チャンクタグに TagChunk を利用した場合.

	開発セット	評価セット	σ^2/C
局所的素性のみ			
CRF	91.39	86.30	200
パーセプトロン	89.36	84.35	-
多数決パーセプトロン	89.76	84.50	-
マージンパーセプトロン	91.06	86.24	32000
＋ 大域的素性			
提案アルゴリズム ($n'=100$)	92.23	87.04	5657
提案アルゴリズム ($n'=6400$)	**92.54**	**87.17**	5657

A. Algorithm 6.2 の収束性

ある重みベクトル $\boldsymbol{U}^l$ ($||\boldsymbol{U}^l||=1, U_i^l=0\ (n+1\le i\le d)$) と定数 R があって, $\forall i, \forall \boldsymbol{y}\in\mathcal{Y}^{|\boldsymbol{x}_i|}\ \Phi^l(\boldsymbol{x_i},\boldsymbol{y_i}^*)\cdot\boldsymbol{U}^l-\Phi^l(\boldsymbol{x_i},\boldsymbol{y})\cdot\boldsymbol{U}^l\ge\delta$, $\forall i, \forall \boldsymbol{y}\in\mathcal{Y}^{|\boldsymbol{x}_i|}\ ||\Phi^a(\boldsymbol{x_i},\boldsymbol{y_i}^*)-\Phi^a(\boldsymbol{x_i},\boldsymbol{y})||\le R$ を満たすとする．$\boldsymbol{\alpha}^k$ を k 番目の更新の前の重みベクトルとし，ϵ_k を k 番目の更新が (A) で行なわれたとき 1，(B) で行なわれたとき 0 をとる変数とすると，更新則は $\boldsymbol{\alpha}^{k+1}=\boldsymbol{\alpha}^k+\epsilon_k(\Phi^a(\boldsymbol{x_i},\boldsymbol{y_i}^*)-\Phi^a(\boldsymbol{x_i},\boldsymbol{y}))+(1-\epsilon_k)(\Phi^l(\boldsymbol{x_i},\boldsymbol{y_i}^*)-\Phi^l(\boldsymbol{x_i},\boldsymbol{y}))$ と書くことができる．

まず，更新則の両辺に $\boldsymbol{U}^l$ を掛けて，$\boldsymbol{\alpha}^{k+1}\cdot\boldsymbol{U}^l=\boldsymbol{\alpha}^k\cdot\boldsymbol{U}^l+\epsilon_k(\Phi^a(\boldsymbol{x_i},\boldsymbol{y_i}^*)\cdot\boldsymbol{U}^l-\Phi^a(\boldsymbol{x_i},\boldsymbol{y})\cdot\boldsymbol{U}^l)+(1-\epsilon_k)(\Phi^l(\boldsymbol{x_i},\boldsymbol{y_i}^*)\cdot\boldsymbol{U}^l-\Phi^l(\boldsymbol{x_i},\boldsymbol{y})\cdot\boldsymbol{U}^l)\ge\boldsymbol{\alpha}^k\cdot\boldsymbol{U}^l+\epsilon_k\delta+(1-\epsilon_k)\delta=\boldsymbol{\alpha}^k\cdot\boldsymbol{U}^l+\delta\ge\boldsymbol{\alpha}^1\cdot\boldsymbol{U}^l+k\delta=k\delta$ から，$(k\delta)^2\le(\boldsymbol{\alpha}^{k+1}\cdot\boldsymbol{U}^l)^2\le(||\boldsymbol{\alpha}^{k+1}||||U^l||)^2=||\boldsymbol{\alpha}^{k+1}||^2$ であることが分かる (1)．一方，更新則から，$||\boldsymbol{\alpha}^{k+1}||^2\le||\boldsymbol{\alpha}^k||^2+2\epsilon_k\boldsymbol{\alpha}(\Phi^a(\boldsymbol{x_i},\boldsymbol{y_i}^*)-\Phi^a(\boldsymbol{x_i},\boldsymbol{y}))+2(1-\epsilon_k)\boldsymbol{\alpha}(\Phi^l(\boldsymbol{x_i},\boldsymbol{y_i}^*)-\Phi^l(\boldsymbol{x_i},\boldsymbol{y}))+R^2\le||\boldsymbol{\alpha}^k||^2+2C+R^2\le||\boldsymbol{\alpha}^1||^2+k(R^2+2C)=k(R^2+2C)$ が分かる (2)．ここでは，仮定から分かる $||\Phi^l(\boldsymbol{x_i},\boldsymbol{y_i}^*)-\Phi^l(\boldsymbol{x_i},\boldsymbol{y})||\le||\Phi^a(\boldsymbol{x_i},\boldsymbol{y_i}^*)-\Phi^a(\boldsymbol{x_i},\boldsymbol{y})||\le R$ という関係と，$||\boldsymbol{x}_i||\le R$, $\sum_i p_i=1$ なら $||\sum_i p_i\boldsymbol{x}_i||\le R$ という関係を利用した．(1) と (2) を組み合わせると，$k\le(R^2+2C)/\delta^2$ であることが分かり，収束性を示すことができる．

参考文献

Bikel, D. M., R. L. Schwartz, and R. M. Weischedel (1999). An algorithm that learns what's in a name. *Machine Learning 34* (1-3).

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *EMNLP 2002*.

Cortes, C. and V. Vapnik (1995). Support vector networks. *Machine Learning 20*, 273–297.

Daumé III, H. and D. Marcu (2005). Learning as search optimization: Approximate large margin methods for structured prediction. In *ICML 2005*.

Finkel, J. R., T. Grenager, and C. Manning (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In *ACL 2005*.

Herbrich, R. and T. Graepel (2000). Large scale Bayes point machines. In *NIPS 2000*.

Krauth, W. and M. Mézard (1987). Learning algorithms with optimal stability in neural networks. *Journal of Physics A 20*.

Krishnan, V. and C. D. Manning (2006). An effective two-stage model for exploiting non-local dependencies in named entity recognitioin. In *ACL-COLING 2006*.

Lafferty, J., A. McCallum, and F. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML 2001*, pp. 282–289.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psycological Review*, 386–407.

Roth, D. and W. Yih (2005). Integer linear programming inference for conditional random fields. In *ICML 2005*.

Taskar, B., C. Guestrin, and D. Koller (2003). Max-margin Markov networks. In *NIPS 2003*.

Tjong, E. F., K. Sang, and F. De Meulder (2003). Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *CoNLL 2003*.