大規模素性集合に対する教師あり縮約モデリング

鈴木 潤　　永田 昌明
NTT コミュニケーション科学基礎研究所
{suzuki.jun, nagata.masaaki}@lab.ntt.co.jp

1 はじめに

形態素解析，固有表現抽出，係り受け解析といった，計算機により自然言語を解析する自然言語処理タスクでは，教師あり学習により解析モデルのパラメタ推定を行うことで良好な解析精度が得られることが多くの研究で示されてきた．以降，本稿では，上記自然言語処理タスクを指す総称として「自然言語解析タスク」と呼ぶ．本稿では，自然言語解析タスクにおける教師あり学習に関して，特に，利用する素性集合を縮約して表現する方法について議論する．

自然言語解析タスクでは，より詳細な素性を追加すると解析精度が向上する場合が多い．端的な例として，一次依存構造の素性に二次依存構造の素性を追加すると，解析精度が大幅に向上する (*i.e.*,[1])．一方で，詳細な素性を導入すると解析精度は向上するが，解析速度の低下やメモリ使用量の増加といった実用上無視できない要因が悪化する，といったトレードオフが存在する．近年，このトレードオフを軽減する試みがいくつか提案されている [2, 3].

本稿では，文献 [2] で提案した「教師あり縮約モデリング」の実用的な拡張について述べる．この方法は，一般的な教師あり学習の最適化問題に対して，素性重み (最適化問題の文脈では最適化変数) に離散制約を加えることで，解の自由度を大幅に制限し，多くの素性重みを同じ値にするというユニークなモデル学習法である．これは，学習後に同じ素性重みを持つ素性を融合することができることから，このモデル学習法により縮約した素性表現を獲得できるという効果がある．ただし，学習時に用いる離散制約の値集合は，データにより最適な値集合が違うため，一般的には開発セットを用いて人手によりチューニングして獲得する．そのため，試行錯誤的にモデル学習を複数回行う必要があり，このチューニングコストは比較的大きなものになる可能性がある．そこで本稿では，データに適した離散制約値集合を学習アルゴリズム内で自動獲得する方法論を追加し，結果としてチューニングせずとも従来と同等かそれ以上の縮約素性表現を獲得できる学習法を構築する．また，モデル学習一回当りの総計算コストは従来とほぼ同等のため，結果として，離散制約値集合のチューニングを行うために必要なモデル学習の繰り返し数の計算コストを削減できたことと等価の効果が得られる．

2 自然言語解析タスクでの教師あり学習

本稿では，自然言語解析タスクの入力を $\mathbf{x}$，出力を $\mathbf{y}$ で表す．また，$\mathbf{x}$ と $\mathbf{y}$ は，タスクの定義に従い計算機が扱い易い離散的な構造で表現されていると仮定する．次に，$\phi(\mathbf{x}, \mathbf{y})$ を，入力 $\mathbf{x}$ と出力 $\mathbf{y}$ のペアを受け取り，$\{0, 1\}$ のいずれかを値を返す (二値) 素性関数とする．また $\mathbf{f}_{\mathbf{x},\mathbf{y}}$ を，1 から N 番目までの N 個の素性関数が返す値を順番に並べてベクトル表現したものとする．

このとき，自然言語解析タスクは，入力 $\mathbf{x}$ に最も適した出力 $\hat{\mathbf{y}}$ を，$\mathbf{x}$ が与えられた時の解候補の集合 $\mathcal{Y}(\mathbf{x})$ から選択する問題として定式化できる．具体的には，以下ような線形判別モデルによる最適化問題により定式化される場合がほとんどである．

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}(\mathbf{x})} \{\mathbf{w} \cdot \mathbf{f}_{\mathbf{x},\mathbf{y}}\} \tag{1}$$

ただし，$\mathbf{w}$ を N 次元ベクトルとし，各素性の重要度に相当するスコア（重み）を表すとする．式 1 は，タスクやモデルに応じて選択された様々な探索アルゴリズムにより決定される．

2.1 教師あり学習によるモデルパラメタ推定

自然言語解析タスクでは，式 1 の $\mathbf{w}$ の各要素の値は，主に教師あり学習を用いて決定する．一般的に以下の最適化問題で定式化される．

$$\begin{aligned} \hat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \big\{\mathcal{O}(\mathbf{w}; \mathcal{D})\big\}, \\ \mathcal{O}(\mathbf{w}; \mathcal{D}) &= \mathcal{L}(\mathbf{w}; \mathcal{D}) + \Omega(\mathbf{w}) \end{aligned} \tag{2}$$

ただし，$\mathcal{D}$ は正解データを表し，入力 $\mathbf{x}$ と出力 $\mathbf{y}$ のペアで構成される．つまり $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ である．$\mathbf{w}$ は，最適化問題の文脈では N 次元ベクトルで表現された最適化変数である．$\mathcal{L}(\mathbf{w}; \mathcal{D})$ と $\Omega(\mathbf{w})$ は機械学習における損失関数と正則化項である．本稿では以下の 2 つの仮定が常に成り立つことを前提とする．

Assumption 1. $\Omega(\mathbf{w})$ は閉凸関数であり，その有効領域は $dom\Omega = \{\mathbf{w} \in \mathbb{R}^N \mid \Omega(\mathbf{w}) < +\infty\}$ である．

Assumption 2. $\mathcal{L}(\mathbf{w}; \mathcal{D})$ は，全ての学習データ $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}$ と有効領域 $dom\Omega$ 内の $\mathbf{w}$ に対して凸かつ劣微分可能な関数である．

つまり，上記仮定は凸最適化問題のみを扱うことを意味する．以下，上記仮定を満たしている限り損失関数 $\mathcal{L}(\mathbf{w}; \mathcal{D})$ と正則化項 $\Omega(\mathbf{w})$ が具体的にどのような関数であるかは本稿の議論に影響を与えないため，特定の定義は与えない状態で議論を進める．ただし，実験では，自然言語解析タスクで典型的に用いられる損失関数と正則化項を用いて実験を行う．

2.2 学習後のモデルを簡潔な表現にする取り組み

教師あり学習により解析モデルを簡潔にする最も有名な方法は，スパースモデリングと呼ばれる L_1 正則化項を用いた教師あり学習法である [4]．スパースモデリングでは，L_1 正則化項の効果で，学習後の最適化変数 $\hat{\mathbf{w}}$ に多くの 0 が含まれるような結果が得られる．式 1 の線形モデルでは，$\mathbf{w}$ の要素の値が 0 だと式 1 に全く影響を与えないので，学習後に $\mathbf{w}$ の要素の値が 0 の素性は解析モデルから削除できる．この削除処理によって，スパースモデリングでは，学習前より相対的に小さい素性集合で学習後の解析を実行することができる．例えば，本稿の実験結果では，数千万素性集合から解析モデルを学習した場合に，最低でも十分の一程度は素性数を削減できる．

近年では，学習時に最適化変数がなるべく同じ値となるような制約を与えることで，スパースモデリングから

更に簡潔なモデルを獲得する素性グルーピングに関する研究が行われている [5, 6]．また，文献 [2] では，教師あり学習の最適化問題式 2 に対して離散制約を追加することで最適化変数の自由度を制限し，学習後に得られる最適化変数の値の種類数が必ず自由度以下になるようなモデル化を提案している．本稿では，この方法論を「スパースモデリング」に対する用語として「縮約モデリング」と呼ぶ．具体的には，文献 [2] での縮約モデリングは，$\mathbf{w} \in \mathcal{S}_\zeta^N$ という制約を式 2 に加えた以下の式で定式化される．

$$\begin{aligned} \hat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \left\{ \mathcal{O}(\mathbf{w}; \mathcal{D}) \right\} \\ \mathcal{O}(\mathbf{w}; \mathcal{D}) &= \mathcal{L}(\mathbf{w}; \mathcal{D}) + \Omega(\mathbf{w}) \\ \text{subject to} \quad & \mathbf{w} \in \mathcal{S}_\zeta^N \end{aligned} \tag{3}$$

このとき，$\mathcal{S}_\zeta^N$ は，ある実数値集合 $\mathcal{S}_\zeta$ のデカルト冪である．$\mathcal{S}_\zeta$ は以下で定義される．

Definition 3 (実数値集合 $\mathcal{S}_\zeta$)**.** ζ は正の整数，$\mathcal{R}_\zeta$ を $\mathcal{R}_\zeta = \{v | 0 < v < -\infty\}$ かつ $|\mathcal{R}_\zeta| = \zeta$ が成り立つ集合，$\sigma = \{-1, 1\}$ とする．このとき，$\mathcal{S}_\zeta = \{yv | (v, y) \in \mathcal{R}_\zeta \times \sigma\} \cup \{0\}$.

つまり，$\mathcal{R}_\zeta$ は ζ 個の要素で構成される正の実数値集合であり，$\mathcal{S}_\zeta$ は $\mathcal{R}_\zeta$ に含まれる実数の正負両方の値と 0 の $2\zeta + 1$ 個の要素で構成される実数値集合である．

式 3 の縮約モデリングでは，モデル学習後，自由度を決定するパラメタ ζ に対して，各素性スコアは，0 を必ず含んだ最大 $2\zeta + 1$ 個の値のいずれか一つに必ずなる．よって，スパースモデリングと同様に，学習後に $\mathbf{w}$ の要素が 0 の素性を解析モデルから削除し，さらに同じ素性スコア値をもつ素性を融合する処理が可能であり，結果としてスパースモデリングより圧倒的に簡潔な素性表現を実現できる．また，驚くべきことに，固有表現抽出や係り受け解析タスクにおいては，$\zeta = 4$(自由度 8) といった非常に小さい自由度でも，従来のトップシステムの解析精度と同等の精度を維持できることが示されている．

3 教師あり縮約モデリングの自動獲得

本稿では，文献 [2] の方法に対してを利便性を高める拡張を行う．文献 [2] では，Def.3 中の離散制約を決定する集合 $\mathcal{S}_\zeta$ に含まれる実数値は人手により事前に決定する．実際文献 [2] では，開発セットを用いて効果的な集合 $\mathcal{D}_\zeta$ を獲得している．

これは，効果的な集合 $\mathcal{S}_\zeta$ が，モデル学習を何度も繰り返し行わなくては得られないことを意味するので，実用上の計算コスト的な課題があると言える．特に ζ が 4 以下のような小さい領域では，$\mathcal{S}_\zeta$ の定義の善し悪しで，解析精度が大きくばらつくため，チューニングコストもそれだけ大きくなってしまう．そこで，本稿では集合 $\mathcal{S}_\zeta$ を学習アルゴリズム内で学習データから自動的に獲得する拡張を提案する．

3.1 最適化の概略

本稿で提案する離散制約値集合 $\mathcal{S}_\zeta$ を学習データから自動獲得可能な教師あり縮約モデリングは基本的に式 3 と同じである．近年の機械学習分野での近接勾配法や双対分解法の発展により，一見困難と思われる最適化問題が，問題をうまく分解することで容易に解くことができる場合がしばしばあることがわかってきた．例えば文献 [8] では，問題を分解しない場合は，組合せ最適化問題に属する計算困難な問題でも，近接勾配法により $N \log N$ 程度の計算量で解くことができることが示される等，非常に興味深い結果が得られている．本稿でも，式 3 を双対分解 [9] の考えに基づいて従来の教師あり学習に相当する問題と，縮約素性表現を構築する問題の二つに分割する．

入力: 学習データ:$\mathcal{D}$, ハイパーパラメタ:ρ, η, ϵ_{primal}, ϵ_{dual}
初期化: $\mathbf{w}^{(1)} = \mathbf{0}$, $\mathbf{u}^{(1)} = \mathbf{0}$, $\boldsymbol{\alpha}^{(1)} = \mathbf{0}$, and $t = 1$.

Step1 ($\mathbf{w}$ の更新):

$$\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w}} \{ \mathcal{O}(\mathbf{w}; \mathcal{D}, \mathbf{u}^{(t)}, \boldsymbol{\alpha}^{(t)}) \} \tag{4}$$

本稿では $\mathcal{O}(\mathbf{w}; \mathcal{D}, \mathbf{u}^{(t)}, \boldsymbol{\alpha}^{(t)})$ は以下の式になる:

$$\mathcal{O}(\mathbf{w}; \mathcal{D}, \mathbf{u}, \boldsymbol{\alpha}) = \mathcal{O}(\mathbf{w}; \mathcal{D}) + \frac{\rho}{2} ||\mathbf{w} - \mathbf{a}||_2^2, \tag{5}$$

ただし $\mathbf{a} = \mathbf{u} - \boldsymbol{\alpha}$.

Step2 ($\mathbf{u}$ の更新):

$$\mathbf{u}^{(t+1)} = \arg\min_{\mathbf{u}} \{ \mathcal{O}(\mathbf{u}; \mathcal{D}, \mathbf{w}^{(t+1)}, \boldsymbol{\alpha}^{(t)}) \} \tag{6}$$

本稿では $\mathcal{O}(\mathbf{u}; \mathcal{D}, \mathbf{w}^{(t+1)}, \boldsymbol{\alpha}^{(t)})$ は以下の式になる:

$$\begin{aligned} \mathcal{O}(\mathbf{u}; \mathcal{D}, \mathbf{w}, \boldsymbol{\alpha}) &= \frac{\rho}{2} ||\mathbf{b} - \mathbf{u}||_2^2 \\ \text{s.t.} \quad & \mathbf{u} \in \mathcal{S}^N, \end{aligned} \tag{7}$$

ただし $\mathbf{b} = \mathbf{w} + \boldsymbol{\alpha}$.

Step3 ($\boldsymbol{\alpha}$ の更新):

$$\boldsymbol{\alpha}^{(t+1)} = \boldsymbol{\alpha}^{(t)} + \eta(\mathbf{w}^{(t+1)} - \mathbf{u}^{(t+1)}) \tag{8}$$

Step4 (収束判定):

$$\begin{aligned} ||\mathbf{w}^{(t+1)} - \mathbf{u}^{(t+1)}||_2^2 / N < \epsilon_{\text{primal}} \\ ||\mathbf{u}^{(t+1)} - \mathbf{u}^{(t)}||_2^2 / N < \epsilon_{\text{dual}} \end{aligned} \tag{9}$$

上記二つの収束条件を満たしたら終了
満たしてなければ $t = t + 1$ として Step1 に戻る

Output: $\mathbf{u}^{(t+1)}$

図 1　ADMM[7] に基づく最適化アルゴリズムの概略

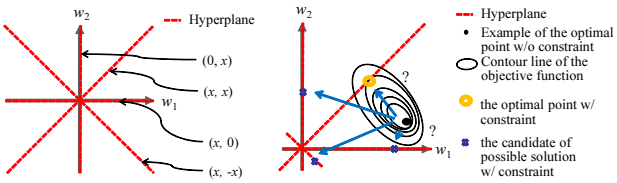
$$\begin{aligned} \hat{\mathbf{w}} &= \arg\min_{\mathbf{w}} \left\{ \mathcal{O}(\mathbf{w}; \mathcal{D}) \right\} \\ \mathcal{O}(\mathbf{w}; \mathcal{D}) &= \mathcal{L}(\mathbf{w}; \mathcal{D}) + \Omega(\mathbf{w}) \\ \text{subject to} \quad & \mathbf{w} = \mathbf{u}, \text{ and } \mathbf{u} \in \mathcal{S}_\zeta^N. \end{aligned} \tag{10}$$

式 10 の最適化問題を解くには，文献 [2] と同様に ADMM に基づくアルゴリズムを用いる． 図 1 に本稿で用いる最適化アルゴリズムの概要を示す．最適化の枠組みは文献 [2] の方法を踏襲する．ADMM の詳細は，文献 [7] を参考にされたい．本稿では，最適化中にデータから自動的に離散制約値集合 $\mathcal{S}_\zeta$ を自動獲得できるようにアルゴリズムを拡張する．具体的には，式 7 で表されている最適化アルゴリズムの変更を行う．それ以外の処理 (Step1,Step3,Step4) に関しては，文献 [2] と同じなので，本稿では詳細な説明は省略する．

3.2 Step2: 離散制約値集合 $\mathcal{S}_\zeta$ の自動獲得

Step2 の最適化問題は式 7 に示す通りである．この最適化問題の解法が，本稿での提案法の主要部である．式 7 は，式 3 と同様に離散制約を持つため組合せ数最適化問題である．しかし，式 7 は，式 3 の一般形と違い現実的な計算量で効率的に解くアルゴリズムを構築可能である．ポイントは，目的関数が単純な L_2 ノルム (最小二乗誤差の形) $\frac{\rho}{2} ||\mathbf{b} - \mathbf{u}||_2^2$ になっている点である．

理解を容易にするため，まずはじめに式 7 の制約なしの最適値 $\hat{\mathbf{u}}'$ を考えると，単純に $\hat{\mathbf{u}}' = \mathbf{b}$ であることは解析的に自明である．次に，容易に得られる制約なしの最適解 $\hat{\mathbf{u}}' = \mathbf{b}$ から，目的関数である L_2 ノルムの観点で最近傍の実行可能領域（離散制約を満たす解）が見つかれば，それが式 7 の解であることも自明である．素性空間の観点では，実行可能領域は N 次元空間内の原点を通る ζ 次元の超平面である (図 2 参照)．また，この最近傍の実行可能領域を探索する問題は，全ての n に対する値 b_n



(a) $N = 2, \zeta = 1$ (b) 最近傍超平面への射影

図 2 $\mathcal{S}_\zeta^N$ で構成する N 次元空間内の ζ 次元超平面の例

を二乗誤差が最小となる ζ 個の量子化値（クラスタ）に射影する問題，または，一次元 K-means クラスタリング問題 [10] と等価である．ここで，Def.3 から $\mathcal{S}_\zeta$ は原点に対して対称な値が必ず含まれるという性質がある．また，式 7 の目的関数も原点に対して対称なので，簡単のため全ての値を符号が正として以降の処理を考える．実際には，最適値が得られたあとに符号のみ復元する．すると，一次元 K-means クラスタリング問題は，以下の式でかける．

$$\hat{u}_n = \arg\min_{v \in \mathcal{Q}_\zeta} \frac{1}{2} \sum_{n=1}^{N} (v - |b_n|)^2, \tag{11}$$

ただし，$\mathcal{Q}_\zeta = \mathcal{R}_\zeta \cup \{0\}$ とする．一次元 K-means クラスタリングは組合せ最適化問題ではあるが，最適解を得ることができる動的計画法 (DP) に基づく多項式時間アルゴリズムが存在する [10]．本稿では，このアルゴリズムを文献 [10] に従って Ckmeans.1d.dp と呼ぶ．時間および空間計算量はそれぞれ $O(KN^2)$ と $O(KN)$ である．ただし，Ckmeans.1d.dp は事前に値が昇順に並んでいることを仮定するため，実際には一般的なソートの時間計算量 $O(N \log N)$ 分の処理が加算される．

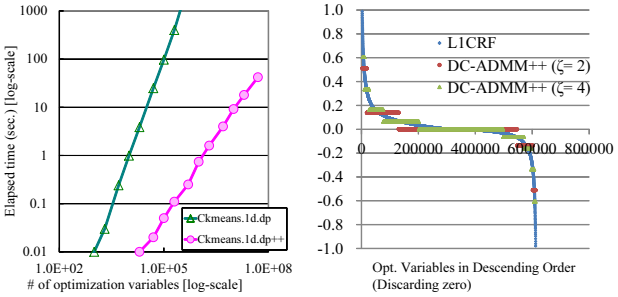
Ckmeans.1d.dp では，$N \times K$ の DP テーブルと，$N \times K$ のバックトラック用テーブル (BT テーブル) を用いる．次に，以下の定義に従って DP テーブル $T[k,i]$ と BT テーブル $B[k,i]$ を埋める [10]．ただし i, j, k は，テーブル内の各インデックスを表し $i, j \in \{1, \ldots, N\}$，$k \in \{1, \ldots, K\}$ である．

- $T[k,i] = \min_{j \in \{k,\ldots,i\}} \{\mathcal{W}(k,j,i)\}$
- $B[k,i] = \arg\min_{j \in \{k,\ldots,i\}} \{\mathcal{W}(k,j,i)\}$
- $\mathcal{W}(k,j,i) = T[k-1, j-1] + \nu_{j,i} - \frac{\mu_{j,i}^2}{2(i-j)}$，ただし，$\nu_{j,i} = \frac{1}{2} \sum_{l=j}^{i} x_l^2$，$\mu_{j,i} = \sum_{l=j}^{i} x_l$．

自然言語処理タスクでは，例えば N は $N > 1,000,000$ のような場合もしばしば起こりえる．よって，時間計算量が $O(KN^2)$ なのは実用上よいとは言えない．そこで，本稿では Ckmeans.1d.dp アルゴリズムを改良し時間計算量を下げることを考える．本稿では，Ckmeans.1d.dp の以下の性質に着目する．

Proposition 4. x_i を昇順ソートした値の i 番目の値とする．また，$c_{i,j}$ を，x_i から x_j までの値で構成されたクラスタとし，その平均値を $\mu_{i,j}$ とする．このとき，一次元 K-means クラスタリングの解である最適クラスタ集合を $(\hat{c}_{1,i}, \ldots, \hat{c}_{j,K})$ とすると，最適クラスタ集合中の隣接する二つのクラスタ $\hat{c}_{i,j-1}$ と $\hat{c}_{j,k}$ には，必ず以下の関係が成り立つ；$x_{j-1} \leq \frac{1}{2}(\mu_{i,j-1} + \mu_{j,k}) \leq x_j$.

Proof. もし $x_{j-1} > \frac{1}{2}(\mu_{i,j-1} + \mu_{j,k})$ が成り立つなら x_{j-1} はクラスタ $c_{j,k}$ に属するはずである．なぜなら，$(x_{j-1} - \mu_{i,j-1})^2 > (x_{j-1} - \mu_{j,k})^2$ であり，$c_{j,k}$ に属した方が目的関数の値は必ず小さくなるからである．同様に，もし $\frac{1}{2}(\mu_{i,j-1} + \mu_{j,k}) > x_j$ なら，$(x_j - \mu_{i,j-1})^2 < (x_j - \mu_{j,k})^2$ が成り立つので x_{j-1} はクラスタ $c_{i,j-1}$ に属するはずである．よって，上記二つの場合は最適クラスタリングの条件を満たさない．これに対して，もし $x_{j-1} \leq \frac{1}{2}(\mu_{i,j-1} + \mu_{j,k}) \leq x_j$ なら，$(x_{j-1} - \mu_{i,j-1})^2 \leq (x_{j-1} - \mu_{j,k})^2$ と $(x_j - \mu_{i,j-1})^2 \geq (x_j - \mu_{j,k})^2$ が必ず成り立つ．よって，x_{j-1} と x_j は最適クラスタに属していると言える． □



(a) 実行時間と最適化変数の総数の関係 (b) 学習後の最適化変数の値

図 3 Ckmeans.1d.dp++ の効果

Prop. 4 を利用することで，Ckmeans.1d.dp を高速化できる．基本的なアイディアは，従来の Ckmeans.1d.dp では Prop. 4 のような指標が考えられていなかったため，全ての DP テーブルの値を計算して埋める必要があったが，提案法では，Prop. 4 の性質上絶対に最適クラスタとして選択されないクラスタ候補を評価しない (DP テーブルの計算をしない) ことで必要な処理量を削減し実行時間を短縮する．また，テーブルの値計算が必要か不要かは Prop. 4 内の x_{j-1} と x_j に挟まれる領域を二分探索で求めることができるため，計算量も $O(KN^2)$ から $O(KN \log N)$ に削減することができる．

4 実験

文献 [2] と同様，固有表現抽出 (NER) と係り受け解析 (DEPER) の実験を行う．なお，実験の設定に関しては文献 [2] に従う．よって，実験データは，NER は CoNLL'03 data [11] であり，DEPER は，Penn Treebank (PTB) III コーパスを文献 [12] のルールに従って係り受けに変更したものである．

本稿の目的は，タスクの解析精度を向上させるというよりは，モデルの複雑さを低減することで，学習後のモデルをコンパクトにしようという試みである．よって，実験の評価軸は解析精度とモデルの簡潔さの二軸になる．解析精度に関しては，各タスクで標準的に用いられている評価指標を使用する．モデルの簡潔さに関しては，文献 [2] と同様に，学習後の最適化変数が非零の数 (**#nzF**) と自由度 DoF (**#DoF**) で評価する．

4.1 Ckmeans.1d.dp++ による離散制約値獲得法の効果

本稿で提案した Ckmeans.1d.dp の速度改良版 (**Ckmeans.1d.dp++**) の効果を実データを用いて検証する．この Ckmeans.1d.dp++ が，文献 [2] との本質的に差分になる．

図 3(a) に NER の $\zeta = 4$ を用いた実験での実行時間 (秒) と最適化変数の総数の関係を示す．図 3 は，対数-対数スケールであることに注意されたい．例えば，Ckmeans.1d.dp は，最適化変数が一万 (10K) の時 97.24 秒かかったが，Ckmeans.1d.dp++ は 0.1 秒以下であっ

	NER	Test COMP	Test F-sc	Model complex. #nzF	Model complex. #DoF
	L2CRF	84.88	89.97	61.6M	38.6M
	L1CRF	84.85	89.99	614K	321K
$\zeta=4$	L1CRF w/ QT	78.39	85.33	**568K**	8
	DC-ADMM	**84.96**	89.92	643K	8
	DC-ADMM++	84.79	**90.02**	586K	8
$\zeta=2$	L1CRF w/ QT	73.40	81.45	454K	4
	DC-ADMM	84.04	89.35	455K	4
	DC-ADMM++	**84.61**	**89.77**	**199K**	4
$\zeta=1$	L1CRF w/ QT	65.53	75.87	454K	2
	DC-ADMM	83.06	88.62	364K	2
	DC-ADMM++	**83.79**	**89.02**	**58.9K**	2

	DEPER	Test COMP	Test UAS	Model complex. #nzF	Model complex. #DoF
	L2PA	49.67	93.51	15.5M	5.59M
	L1RDA	49.54	93.48	7.76M	3.56M
$\zeta=4$	L1RDA w/ QT	38.58	90.85	6.32M	8
	DC-ADMM	49.83	**93.55**	5.81M	8
	DC-ADMM++	**50.00**	93.51	**1.62M**	8
$\zeta=2$	L1RDA w/ QT	34.19	89.42	3.08M	4
	DC-ADMM	**48.97**	93.18	4.11M	4
	DC-ADMM++	48.51	**93.20**	**623K**	4
$\zeta=1$	L1RDA w/ QT	30.42	88.67	3.08M	2
	DC-ADMM	**46.56**	**92.86**	6.37M	2
	DC-ADMM++	44.77	92.37	**598K**	2

表 1　評価データの解析精度 (K: thousand, M: million)

た．また，Ckmeans.1d.dp++ は変数が一千万 (10M) でも 10 秒以下で実行できている．実際に，実験データでのモデル学習は，比較的高速なオンライン (L1 正則化) 学習法を用いたとしても，全体で数時間程度の時間がかかる．また，教師あり縮約モデリング全体の学習で Step2 が呼ばれる回数は多くてもせいぜい数十回程度（繰り返し回数）である．よって，モデル学習全体の実行時間からすれば，Ckmeans.1d.dp++ の処理時間はほぼ無視できる程度には小さいことがわかる．逆に，オリジナルの Ckmeans.1d.dp では，N が非常に大きいときには実行時間が無視できないほど大きくなることも実験結果からわかった．追加情報として，提案法の最悪時間計算量は $O(KN \log N)$ であるが，実際は N に対してほぼ線形の関係になっていることが図 3(a) からみてとれる．このことからも，Ckmeans.1d.dp++ は，N が今後さらに増加したとしてもモデル学習全体からすればほとんど問題にならないと言える．

図 3(b) に NER における学習後の最適化変数を降順にソートした値を示す．提案法では，離散制約値がデータから自動的に獲得されていることが見て取れる．自由度のパラメタ ζ 以外の人間が事前に情報を与えなくても，データからうまく最適化変数を縮約できていることがみてとれる．

4.2 教師あり縮約モデリングの効果

表 1 に，最終的な評価データの解析精度を示す．本稿では，文献 [2] の学習法の拡張に位置づけられるので，実験のベースラインは文献 [2] の **DC-ADMM** になる．また，提案法を **DC-ADMM++** と呼ぶ．ただし，自然言語解析タスクでより一般的に用いられているベースラインという位置づけで，L_1 正則化によるスパースモデリングの結果も示す．さらに，文献 [2] と同様に，公平な評価をするためにスパースモデリングにより獲得したモデルを，学習の後処理として DC-ADMM の枠組みに即して量子化した結果 (**w/ QT**) も示す．

まず，DC-ADMM または DC-ADMM++ が，解析精度を保持したまま，自由度 8 といった驚異的に小さい自由度でモデルを縮約できていることが見てとれる．さらに全体として DC-ADMM++ は，従来法である DC-ADMM と同等かそれ以上の結果が得られた．一番注目すべき点は，DC-ADMM の結果は開発データを用いて人間の知識を加味し，離散制約値を求めた末に得られた結果であるのに対して，DC-ADMM++ は一回のモデル学習で得られた結果であることである．実際に開発データを用いた人手チューニングにどの程度のコストがかかるかはデータやタスクにより違うため定量的な評価は難しいところである．しかし，実用では一回のモデル学習でよい結果が得られることが経験的にわかっていることは利用者側の立場からは非常にポジティブな性質と言える．少なくとも本実験では，DC-ADMM は最低十回程度のモデル学習を繰り返した結果であるので，DC-ADMM++ は，潜在的にモデル学習時間を十分の一程度に削減できたとも捉えることができる．

5 まとめ

本稿では，文献 [2] と同様に，自然言語解析タスクでよく用いられる教師あり学習の枠組みを拡張して，学習後のモデルを縮約する方法論に関して議論を行った．また，文献 [2] の方法では，離散制約値を開発セットを用いて人手によりチューニングする必要があったが，本稿では，離散制約値をデータから自動獲得する方法に拡張した．提案法により，効果的な離散制約値を獲得するチューニングコスト（複数回のモデル学習コスト）を削減することが可能となり，結果としてモデル学習に必要な時間的コストを大幅に軽減することに成功した．

参考文献

[1] Xavier Carreras. Experiments with a Higher-Order Projective Dependency Parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961, 2007.

[2] Jun Suzuki and Masaaki Nagata. Supervised model learning with feature grouping based on a discrete constraint. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 18–23, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.

[3] Daniel Golovin, D. Sculley, H. Brendan McMahan, and Michael Young. Large-scale learning with less ram via randomization. In *ICML (2)*, volume 28 of *JMLR Proceedings*, pages 325–333. JMLR.org, 2013.

[4] Jianfeng Gao, Galen Andrew, Mark Johnson, and Kristina Toutanova. A comparative study of parameter estimation methods for statistical natural language processing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 824–831, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[5] Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and Smoothness via the Fused Lasso. *Journal of the Royal Statistical Society Series B*, pages 91–108, 2005.

[6] Howard D. Bondell and Brian J. Reich. Simultaneous Regression Shrinkage, Variable Selection and Clustering of Predictors with OSCAR. *Biometrics*, 64(1):115, 2008.

[7] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Foundations and Trends in Machine Learning, 2011.

[8] Leon Wenliang Zhong and James T. Kwok. Efficient Sparse Modeling with Automatic Feature Grouping. In *ICML*, 2011.

[9] Hugh Everett. Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources. *Operations Research*, 11(3):399–417, 1963.

[10] Haizhou Wang and Mingzhou Song. Ckmeans.1d.dp: Optimal *k*-means Clustering in One Dimension by Dynamic Programming. *The R Journal*, 3(2):29–33, 2011.

[11] Erik Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of CoNLL-2003*, pages 142–147, 2003.

[12] Hiroyasu Yamada and Yuji Matsumoto. Statistical Dependency Analysis with Support Vector Machines. In *Proc. of IWPT*, 2003.