統計数理(2016)
第 64 巻 第 2 号 179–200

特集「統計的言語研究の現在」
[原著論文]

条件付き確率場の理論と実践

岡崎 直観†

(受付 2016 年 6 月 17 日；改訂 8 月 25 日；採択 9 月 14 日)

要　　旨

自然言語処理のタスクの多くは，入力から出力のラベルを予測する問題として定式化できる．言語は構造を持つと考えられるので，入力や出力に単語列や木などの構造を持たせることで，さらに多くのタスクが予測問題として定式化できる．本稿では，系列ラベリング問題，すなわち入力と出力が系列データの場合の条件付き確率場を解説する．条件付き確率場は，多クラスロジスティック回帰を系列データに適用するため，ラベル列のマルコフ性を仮定した素性関数を導入し，動的計画法でラベル列の予測とパラメータの学習を効率化している．そこで，ロジスティック回帰の素性関数，確率的勾配降下法による学習，正則化などの基礎理論を復習し，条件付き確率場の理論全体を説明する．また，能動学習，部分的に正解が付与された訓練データからの学習，深層ニューラルネットワークの適用など，条件付き確率場の最近の研究動向や実践について概観する．

キーワード：条件付き確率場，ロジスティック回帰，確率的勾配降下法．

1. はじめに

自然言語処理は，言葉を操る賢いコンピュータを実現することを究極の目標としているが，その目標到達への道程は長い．一方で，自然言語処理は情報検索，機械翻訳，質問応答，自動要約，対話生成，評判分析，ソーシャルネットワーク分析など，幅広い応用を生み出している．これらを支えているのは，品詞タグ付け(形態素解析)，チャンキング，固有表現抽出，構文解析，共参照解析，意味役割付与などの基盤解析技術である．

このように，自然言語処理は多種多様なタスクに取り組んでいるが，その多くのタスクは入力 x から出力 $\hat{y}$ を予測する問題として定式化できる．入力 x に対する出力 y の「良さ」をスコア付けする関数を $s(x, y)$ と書くと，与えられた入力に対して，可能な出力の集合 $\mathcal{Y}$ から最適な出力 $\hat{y}$ を選ぶ問題は，次式で与えられる．

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} s(x, y) \tag{1.1}$$

例えば，入力文を単語列 $\boldsymbol{x}$ で表し，その文がポジティブ(+1)な内容であるか，ネガティブ(−1)な内容であるか，どちらでもないか(0)，を判定する評判分析タスクは，次式で定式化される．

$$\hat{y} = \operatorname*{argmax}_{y \in \{+1,0,-1\}} s(\boldsymbol{x}, y) \tag{1.2}$$

関数 $s(x, y)$ はナイーブベイズ，パーセプトロン，ロジスティック回帰，サポートベクトルマシ

†東北大学大学院 情報科学研究科：〒 980–8579 宮城県仙台市青葉区荒巻字青葉 6–6–05

ン，ニューラルネットワークなどの手法を用い，訓練データから自動的に導出することが多い．

言語は構造を持つので，出力にも単語列や木などの構造を持たせることにすると，さらに多くのタスクが予測問題として定式化される．例えば，M 個の単語の列 $\boldsymbol{x} = (x_1, x_2, \ldots, x_M)$ から M 個の品詞ラベルの列 $\boldsymbol{y} = (y_1, y_2, \ldots, y_M)$ を予測すると，品詞タグ付け(part-of-speech tagging)が実現する．予測するラベルの種類を B-NP, I-NP, B-VP などのチャンクタグに変更すると，浅い句構造解析(shallow parsing)をモデル化できる．このように，系列 $\boldsymbol{x}$ を入力として系列 $\hat{\boldsymbol{y}}$ を出力する問題は，系列ラベリング(sequential labeling)と呼ばれる．

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}^M} s(\boldsymbol{x}, \boldsymbol{y}) \tag{1.3}$$

ここで，出力されるラベルの集合を $\mathcal{Y}$ と書くと，予測されるラベル列 $\hat{\boldsymbol{y}}$ は $\mathcal{Y}^M$ の中から選ぶ．また，スコア付け関数 $s(\boldsymbol{x}, \boldsymbol{y})$ は，隠れマルコフモデル，構造化パーセプトロン，条件付き確率場などで学習する．

本稿では，系列データにおける条件付き確率場の理論と実践を解説する．条件付き確率場の本質は，多クラスロジスティック回帰を系列データに適用するため，ラベル列のマルコフ性を仮定した素性関数を導入し，動的計画法でラベル列の予測とパラメータの学習を効率化した点にある．そこで，本稿では条件付き確率場の導入として，ロジスティック回帰，および多クラスロジスティック回帰を復習する．これらの理論をベースに条件付き確率場を説明し，能動学習や部分アノテーションなどの拡張や，最近の研究動向を概観する．

2. 線形二値分類器

線形二値分類器(linear binary classifier)は，入力 x に対してスコア $s(x) \in \mathbb{R}$ を計算し，その正負によって二値のラベル $\hat{y} \in \{+1, -1\}$ を推定する．

$$s(x) = \boldsymbol{w}^{\intercal} \boldsymbol{\phi}(x) \tag{2.1}$$

$$\hat{y} = \begin{cases} +1 & (s(x) = \boldsymbol{w}^{\intercal} \boldsymbol{\phi}(x) \geq 0) \\ -1 & (\text{それ以外の場合}) \end{cases} \tag{2.2}$$

ただし，$\boldsymbol{\phi}(x) \in \mathbb{R}^d$ は x を素性ベクトル(feature vector)で表現したものである(d は素性空間の次元数)．$\boldsymbol{w} \in \mathbb{R}^d$ は重みベクトル(weight vector)と呼ばれ，訓練データに適合するように学習で求める．式(2.2)は，入力の素性ベクトルと重みベクトルの内積値を計算し，その正負でラベルを推定するという，単純明快な分類ルールである．以降では，素性関数 $\boldsymbol{\phi}(x)$ の設計と，重みベクトル $\boldsymbol{w}$ の学習について説明する．

2.1 素性空間

式(2.2)からも明らかなように，分類器は入力 x を素性ベクトル $\boldsymbol{\phi}(x)$ を通して観測し，そのラベルを予測する．したがって，高性能の分類器を開発するためには，入力の特徴をよく反映し，ラベルの予測に効きそうな素性空間(feature space)を設計する必要がある．残念ながら，よい素性を設計するための汎用的な方法はないが，素性の作り方には一定の慣習がある．

題材として，文章中の英単語 x が人名の一部であるか($y = +1$)，そうでないか($y = -1$)を推定するタスクを考える．単語の先頭が大文字から始まる場合，その単語は固有名詞である可能性が高くなる．例えば，素性ベクトルの 2 次元目として，次式で表現される素性関数 $\phi_2(x)$ を定義する．

$$\phi_2(x) = \begin{cases} 1 & (x \text{ が大文字で始まる場合}) \\ 0 & (\text{それ以外の場合}) \end{cases} \tag{2.3}$$

他にも，全ての文字が小文字か(人名の可能性は低い)，全てが大文字の単語か(人名の可能性は低い)，x が *Mar* という綴りで始まるか(*Mary*, *Maria*, *Margaret* のように女性の名前である可能性が高い)，文章中で直前の単語が *Ms* か(人名の可能性は高い)など，x の特徴を表す様々な事象を素性として定義する．このように，x の特徴を表す事象(条件)を考え，その条件の成立時のみ 1 を返すような素性関数をたくさん定義し，それぞれを特徴ベクトルの各次元に割り当てる．入力 x に対して素性関数 $\phi_k(x)$ が 0 以外の値を返すとき，入力 x に対して素性 ϕ_k が「発火」したと呼ぶ．

先ほど，「単語が *Mar* という綴りで始まるか」という素性を説明したが，「単語が *ie* という綴りで終わるか(*Stephanie*, *Marie*, *Julie*)」や「単語が *ard* という綴りで終わるか(*Richard* や *Edward*)」など，人名の推定に効きそうな特徴はたくさんある．また，人名ではない英単語を $y = -1$ と予測することも重要であるので，「人名らしくない単語の特徴」にも着目すべきである．しかしながら，これらの素性を人間が網羅的に発見し，定義することは困難である．

そこで，入力の特徴を学習データから掘り起こし，素性関数を定義するアプローチもよく採用される．例えば，学習データに含まれる単語から「長さ 3 の接頭辞を取り出す」というルールを設計し，抽出された全ての接尾辞に対して素性関数を定義することで，「単語が *Mar* という綴りで始まる」「単語が *Ale* という綴りで始まる」などの素性関数が自動的に導出される．素性関数を取り出すルールは，素性テンプレート(feature template)と呼ばれる．

単語 x の長さ $2, 3, 4$ の接頭辞や接尾辞，単語 x の前後に出現する単語など，様々な素性テンプレートを設計・適用することで，人名の予測に効きそうな素性を系統的に作成できる．一方，素性テンプレートは機械的な処理なので，人名の予測に役に立たない素性も多く作り出されてしまう．しかし，線形二値分類器の学習では，それぞれの素性関数 $\phi_i(x)$ の有用性が重み w_i で調整され，役に立たない素性の重みは 0 に近くなると期待できる．ゆえに，役に立たない素性を生成するリスクを気にせず，網羅性重視で素性テンプレートを設計することが多い．

ところで，どのような入力に対しても，常に 1 を返すような素性を導入しておくと，原点を通らない分離平面を表現することができる．例えば，入力 x によらず常に $\phi_1(x) = 1$ を返すような素性を定義すると，$y = +1$ と予測する条件は次式で表される(説明のため，素性ベクトルの 1 次元目を用いたが，次元のインデックス番号は任意である)．

$$\boldsymbol{w}^\intercal \boldsymbol{\phi}(x) = w_1 + \sum_{k=2}^{K} w_k \phi_k(x) \geq 0 \Leftrightarrow \sum_{k=2}^{K} w_k \phi_k(x) \geq -w_1 \tag{2.4}$$

したがって，1 次元目の重み w_1 は分類時の閾値を表しており，この閾値をも学習で自動的に調節することができる．w_1 はバイアス項(bias term)と呼ばれ，素性の重みベクトルと分けて明示的に記述されることもある．

式(2.3)は，ある条件を満たした場合は 1 を，それ以外は 0 を返す関数として設計されている．ところが，線形二値分類器の枠組みでは，素性関数は 0 や 1 だけでなく，実数値を返してもよい．したがって，式(2.3)の素性関数が 1 の代わりに 2 や 0.1 の値を返しても問題ない．実際には，ある素性関数の値を常に定数倍しても，対応する重みがその定数倍を打ち消すように調整されることが多いので，1 以外の素性値を用いることは稀である．

ただし，ある特徴が出現した回数や，ある特徴に関連する事象が別のコーパス中で出現する回数など，入力 x に応じて素性値を変化させたい場合は，実数値を返す素性を定義する．例えば，大規模なコーパスにおいて，単語 x の直前に *Ms* という語が出現した回数(出現頻度)の対

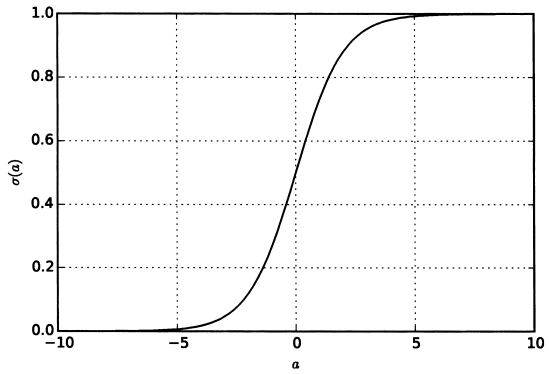

図 1. シグモイド関数 $\sigma(a)$.

数を素性値として用いることで，単語 x の名前らしさを連続値で示唆する素性を設計できる．

$$\phi(x) = \log(x \text{ がコーパスにおいて } Ms \text{ の後に出現する回数}) \tag{2.5}$$

なお，素性値は線形二値分類器のスコアに線形の影響を与える．式(2.5)では，少数の高頻度語の影響を抑えるため，出現回数の対数をとっている．このような非線形の変換は，実数値を返す素性の側に盛り込んでおく必要がある．

2.2 ロジスティック回帰

ロジスティック回帰(logistic regression)は線形二値分類器の一種で，入力 x に対するラベル $y \in \{+1, -1\}$ の条件付き確率 $p(y|x)$ を，シグモイド関数 σ(sigmoid function)でスコア $s(x)$ を確率値に変換することにより計算する．

$$p(y|x) = \sigma(ys(x)) = \frac{1}{1 + \exp\left(-y\boldsymbol{w}^\intercal \boldsymbol{\phi}(x)\right)} \tag{2.6}$$

シグモイド関数 $\sigma(a) = \frac{1}{1+\exp(-a)}$ は，$(-\infty, +\infty) \to (0, 1)$ の単調増加関数で，図 1 のような形状をしている．$p(+1|x) = \sigma(s(x))$ であるから，入力の素性ベクトルと重みベクトルの内積値 $s(x)$ を図 1 の横軸にとり，$y = +1$ と予測する確率 $p(+1|x)$ を縦軸から求めていることに相当する．図 1 からも明らかなように，$s(x)$ が大きければ $p(+1|x)$ は 1 に近づき，$s(x)$ が小さければ $p(+1|x)$ は 0 に近づく．また，

$$1 - \sigma(a) = \frac{\left\{1 + e^{-a}\right\} - 1}{1 + e^{-a}} = \frac{e^{-a}}{1 + e^{-a}} = \frac{e^a e^{-a}}{e^a(1 + e^{-a})} = \frac{1}{1 + e^a} = \sigma(-a) \tag{2.7}$$

であるから，

$$p(-1|x) = \sigma(-s(x)) = 1 - \sigma(s(x)) = 1 - p(+1|x) \tag{2.8}$$

が成り立つ．式(2.8)より，$y = +1$ と予測する確率と，$y = -1$ と予測する確率の和が 1 になることが確認できる．

なお，入力 x を $y = +1$ と予測する確率が 0.5 を上回る条件を求めると，式(2.2)と整合することが確認できる．

$$p(+1|x) \geq 0.5 \Leftrightarrow \frac{1}{1 + \exp(-\boldsymbol{w}^\intercal \boldsymbol{\phi}(x))} \geq 0.5 \Leftrightarrow \boldsymbol{w}^\intercal \boldsymbol{\phi}(x) \geq 0 \tag{2.9}$$

2.3 ロジスティック回帰モデルの学習

入力 $x^{(i)}$ に対して正解の二値ラベル $y^{(i)}$ を付与した N 件の訓練事例 $\mathcal{D} = (x^{(i)}, y^{(i)})_{i=1}^{N}$ がある．ロジスティック回帰モデルの学習とは，学習データの各訓練事例の入力 $x^{(i)}$ に対して，その正解ラベル $y^{(i)}$ を再現(正しく予測)できるように，重みベクトル $\boldsymbol{w}$ を調整することである．ある訓練事例 $(x^{(i)}, y^{(i)})$ に関して，ロジスティック回帰モデルの予測の正しさは $p(y^{(i)}|x^{(i)})$ で見積もることができる．すなわち，$p(y^{(i)}|x^{(i)})$ が 1 に近ければモデルの予測が正しく，0 に近ければモデルの予測が間違っていることになる．

そこで，全ての学習事例に対して $p(y^{(i)}|x^{(i)})$ を計算し，その確率の積を求める．

$$\prod_{i=1}^{N} p(y^{(i)}|x^{(i)}) \tag{2.10}$$

式(2.10)は，学習データ $\mathcal{D}$ におけるモデルの尤もらしさを表すので，尤度(likelihood)と呼ばれる．したがって，式(2.10)を重みベクトル $\boldsymbol{w}$ の関数とみなし，式(2.10)を最大化するような重みベクトル $\boldsymbol{w}^*$ を求めると，訓練事例を再現できる分類器を学習したことになる．このように，尤度関数を最大化することでモデルのパラメータを学習することを最尤推定(maximum likelihood estimation)と呼ぶ．ただ，式(2.10)は，小さい数(確率値)の積を計算するため，数値計算ではアンダーフローなどの問題を引き起こす．そこで，式(2.10)の対数をとった対数尤度(log likelihood)を最大化することが多い．

$$\mathcal{L}^{\mathrm{LR}} = \log \prod_{i=1}^{N} p(y^{(i)}|x^{(i)}) = \sum_{i=1}^{N} \log p(y^{(i)}|x^{(i)}) \tag{2.11}$$

まとめると，ロジスティック回帰モデルの学習は，以下の目的関数 $E^{\mathrm{LR}}(\boldsymbol{w})$ を最小化することに帰着する．

$$E^{\mathrm{LR}}(\boldsymbol{w}) = -\mathcal{L}^{\mathrm{LR}} = -\sum_{i=1}^{N} \log p(y^{(i)}|x^{(i)}) \tag{2.12}$$

幸いなことに，この目的関数は凸関数であり，大域最適解 $\boldsymbol{w}^*$ を持つ．式(2.12)を最小化する手法として，L-BFGS 法や確率的勾配降下法(stochastic gradient descent)が代表的である．ここでは，理論と実装が簡単な確率的勾配降下法を説明する．

確率的勾配降下法は勾配降下法(gradient descent)をベースにしている．勾配降下法は，次の漸化式を $t = 1, 2, \ldots, T$ に関して適用し，式(2.12)の解を求める．

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} - \eta^{(t)} \frac{\partial E^{\mathrm{LR}}(\boldsymbol{w})}{\partial \boldsymbol{w}^{(t)}} = \boldsymbol{w}^{(t)} + \eta^{(t)} \frac{\partial \mathcal{L}^{\mathrm{LR}}}{\partial \boldsymbol{w}^{(t)}} \tag{2.13}$$

重みベクトルの初期値 $\boldsymbol{w}^{(1)}$ は $\mathbf{0}$ など，適当な値に設定すればよい．$\eta^{(t)}$ は学習率と呼ばれ，$1/t$, $1/\sqrt{t}$, $1/(t_0 + t)$ など，反復が進むにつれて減衰していくように設定する($t_0 > 0$ はハイパー・パラメータ)．式(2.13)は，t 回目の反復において $\boldsymbol{w}^{(t)}$ を $\mathcal{L}^{\mathrm{LR}}$ の最急方向(steepest direction)に向け，幅 $\eta^{(t)}$ だけ動かすという更新式を表している．反復を終了する回数 T は，ハイパー・パラメータとして予め設定したり，収束判定などで漸化式の適用時に自動的に決定する．例えば，反復において重みベクトルの変化量の 2-ノルムが初めて $\epsilon > 0$ を下回った時を収束と判定するには，次式を用いる．

$$\|\boldsymbol{w}^{(T+1)} - \boldsymbol{w}^{(T)}\|_2 < \epsilon \tag{2.14}$$

ところで，訓練事例ごとの対数尤度

$$l(x, y) = \log p(y|x) \tag{2.15}$$

を定義すると，勾配は

$$\frac{\partial \mathcal{L}^{\mathrm{LR}}}{\partial \boldsymbol{w}} = \sum_{i=1}^{N} \frac{\partial l(x^{(i)}, y^{(i)})}{\partial \boldsymbol{w}} \tag{2.16}$$

のように，各訓練事例の勾配の和として書ける．したがって，式(2.13)の勾配 $\frac{\partial \mathcal{L}^{\mathrm{LR}}}{\partial \boldsymbol{w}^{(t)}}$ を求めるには，学習データ $\mathcal{D}$ に含まれている全ての訓練事例 $(x^{(i)}, y^{(i)})$ に関する対数尤度 $\frac{\partial l(x^{(i)}, y^{(i)})}{\partial \boldsymbol{w}^{(t)}}$ を計算し，その和を計算することになる．しかしながら，学習データの訓練事例数が多くなると，式(2.16)の計算に時間がかかるだけでなく，重みベクトル $\boldsymbol{w}$ を更新する間隔が長くなり，収束が遅くなる．

確率的勾配降下法は，勾配が式(2.16)のように各事例の勾配の和で求められるとき，勾配 $\frac{\partial \mathcal{L}^{\mathrm{LR}}}{\partial \boldsymbol{w}^{(t)}}$ の代わりに各事例 $(x^{(i)}, y^{(i)})$ の勾配 $\frac{\partial l(x^{(i)}, y^{(i)})}{\partial \boldsymbol{w}}$ を用いる．すなわち，式(2.13)の代わりに次の漸化式を用いる．

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta^{(t)} \frac{\partial l(x^{(i)}, y^{(i)})}{\partial \boldsymbol{w}^{(t)}}, \quad i = t \bmod N \tag{2.17}$$

なお，$t \bmod N$ は t を訓練事例数 N で割ったときの余りである．

式(2.17)の更新式を得るため，訓練事例 (x, y) 毎の対数尤度 $l(x, y)$ を重みベクトル $\boldsymbol{w}$ で微分する．$l(x, y)$ を合成関数とみなすと，

$$\frac{\partial l(x, y)}{\partial \boldsymbol{w}} = \frac{\partial l(x, y)}{\partial p(y|x)} \frac{\partial p(y|x)}{\partial s(x, y)} \frac{\partial s(x, y)}{\partial \boldsymbol{w}}. \tag{2.18}$$

式(2.15)より，

$$\frac{\partial l(x, y)}{\partial p(y|x)} = \frac{\partial}{\partial p(y|x)} \log p(y|x) = \frac{1}{p(y|x)}. \tag{2.19}$$

次に，$p(y|x)$ を $s(x, y)$ で微分する（簡略化のために $s = s(x, y)$ と略記する）．最終行の変形では，式(2.6)と式(2.7)を用いる．

$$\begin{aligned} \frac{\partial p(y|x)}{\partial s(x, y)} &= \frac{\partial}{\partial s} \left(\frac{1}{1 + e^{-ys}} \right) \\ &= (-1) \cdot \frac{1}{(1 + e^{-ys})^2} \cdot e^{-ys} \cdot (-y) \\ &= y \cdot \frac{1}{1 + e^{-ys}} \cdot \frac{e^{-ys}}{1 + e^{-ys}} = y \cdot p(y|x) \left\{ 1 - p(y|x) \right\} \end{aligned} \tag{2.20}$$

また，

$$\frac{\partial s(x, y)}{\partial \boldsymbol{w}} = \frac{\partial}{\partial \boldsymbol{w}} \left(\boldsymbol{w}^{\intercal} \boldsymbol{\phi}(x) \right) = \boldsymbol{\phi}(x) \tag{2.21}$$

ゆえに，

$$\frac{\partial l(x, y)}{\partial \boldsymbol{w}} = \frac{1}{p(y|x)} \cdot y \cdot p(y|x) \left\{ 1 - p(y|x) \right\} \cdot \boldsymbol{\phi}(x) = y \left\{ 1 - p(y|x) \right\} \boldsymbol{\phi}(x). \tag{2.22}$$

最終的に，式(2.17)の更新式は次式で表される．

$$\boldsymbol{w}^{(t+1)} = \boldsymbol{w}^{(t)} + \eta^{(t)} y^{(i)} \{ 1 - p(y^{(i)}|x^{(i)}) \} \boldsymbol{\phi}(x^{(i)}), \quad i = t \bmod N \tag{2.23}$$

理論的な説明が長くなったが，確率的勾配降下法によるロジスティック回帰モデルの学習は

Algorithm 1 の擬似コードで表される．この擬似コードでは，学習データに対する反復回数を T 回と定めているが，式(2.14)を使って収束を判定してもよい．また，学習率 η の計算も，$1/t$, $1/\sqrt{t}$, $1/(t_0+t)$ などに変更してもよい．

Algorithm 1: 確率的勾配降下法によるロジスティック回帰モデルの学習
入力: 学習データ $\mathcal{D} = (x^{(i)}, y^{(i)})_{i=1}^{N}$
出力: 重みベクトル $\boldsymbol{w}$

$t \leftarrow 1$;
$\boldsymbol{w} = \boldsymbol{0}$;
for epoch $\leftarrow 1$ **to** T **do**
  for $i \leftarrow 1$ **to** N **do**
    $\eta \leftarrow 1/t$(※他の計算方法でも可);
    $q \leftarrow 1 - p(y^{(i)}|x^{(i)})$(※現在の重みベクトル $\boldsymbol{w}$ を用いて);
    $\boldsymbol{w} \leftarrow \boldsymbol{w} + \eta y^{(i)}(1-q)\boldsymbol{\phi}(x^{(i)})$;
    $t \leftarrow t + 1$;
  end
end

学習率 η を 1 に固定して考えると，Algorithm 1 は各訓練事例 $(x^{(i)}, y^{(i)}) \in \mathcal{D}$ に対して，以下の処理を行うと解釈できる．

(1) 現在の重みベクトル $\boldsymbol{w}$ を使い，訓練事例の入力 $x^{(i)}$ からラベル $y^{(i)}$ を予測する確率(事例の尤度) q を計算する．
(2) 重みベクトル $\boldsymbol{w}$ を次式で更新する: $\boldsymbol{w}^{\mathrm{new}} \leftarrow \boldsymbol{w}^{\mathrm{old}} + y^{(i)}(1-q)\boldsymbol{\phi}(x^{(i)})$.

$y^{(i)} = +1$ のとき，更新後の重みベクトル $\boldsymbol{w}^{\mathrm{new}}$ は次式で与えられる．

$$\boldsymbol{w}^{\mathrm{new}} = \boldsymbol{w} + (1-q)\boldsymbol{\phi}(x^{(i)}) \tag{2.24}$$

基本的に，重みベクトル $\boldsymbol{w}$ に訓練事例の素性ベクトルを $\boldsymbol{\phi}(x^{(i)})$ を足し込むことになり，その量は $(1-q)$ で調整される．ここで，$(1-q)$ は予測の外れ度合いを表すので，予測が大幅に外れているときは重みベクトルの更新幅が大きくなり，予測がそれほど外れていないときは，重みベクトルの更新幅が小さくなる．

更新後の重みベクトル $\boldsymbol{w}^{\mathrm{new}}$ を使ってスコア $s^{\mathrm{new}}(x^{(i)})$ を再計算する．

$$\begin{aligned} s^{\mathrm{new}}(x^{(i)}) &= (\boldsymbol{w} + (1-q)\boldsymbol{\phi}(x^{(i)}))^{\intercal}\boldsymbol{\phi}(x^{(i)}) \\ &= \boldsymbol{w}^{\intercal}\boldsymbol{\phi}(x^{(i)}) + (1-q)(\boldsymbol{\phi}(x^{(i)}))^{\intercal}\boldsymbol{\phi}(x^{(i)}) \geq s(x^{(i)}) \end{aligned} \tag{2.25}$$

したがって，訓練事例 $(x^{(i)}, y^{(i)})$ に関して重みベクトルを更新することで，スコア $s(x^{(i)})$ が上昇するので，この事例のラベルを $+1$ と予測しやすくなることが分かる．

$y^{(i)} = -1$ のときも同様に，

$$\begin{aligned} s^{\mathrm{new}}(x^{(i)}) &= (\boldsymbol{w} - (1-q)\boldsymbol{\phi}(x^{(i)}))^{\intercal}\boldsymbol{\phi}(x^{(i)}) \\ &= \boldsymbol{w}^{\intercal}\boldsymbol{\phi}(x^{(i)}) - (1-q)(\boldsymbol{\phi}(x^{(i)}))^{\intercal}\boldsymbol{\phi}(x^{(i)}) \leq s(x^{(i)}) \end{aligned} \tag{2.26}$$

であるから，スコア $s(x^{(i)})$ を減少させる作用があり，この事例のラベルを -1 と予測しやすく

なる．

2.4 正則化

ここで，式(2.12)の目的関数が最小となる条件について考えてみたい．これは，式(2.10)の尤度が最大となる時であるから，全ての学習事例の尤度が 1 となる場合である．しかし，式(2.6)から明らかなように，ロジスティック回帰モデルで $p(y|x) = 1$ となるのは，$s(x) \to \pm\infty$ の時だけである．素性ベクトル $\boldsymbol{\phi}(x)$ の大きさを ∞ とすることはあり得ないので，$s(x) \to \pm\infty$ を成立させるには，重みベクトル $\boldsymbol{w}$ の大きさが ∞ になる必要がある．

一般的に，重みベクトル $\boldsymbol{w}$ が大きくなると，素性ベクトル $\boldsymbol{\phi}(x)$ の僅かな差でもスコア $s(x)$ が大きく変動するため，学習データのノイズに対する耐性が低下する．また，重みベクトルの要素の分散が大きくなるため，それぞれの訓練事例のみで発火する素性に依存してスコア付けを行うようになり，過学習を引き起こす．そこで，実際にロジスティック回帰モデルを学習するときは，目的関数に正則化項(regularization term)を追加し，重みベクトル $\boldsymbol{w}$ が大きくなり過ぎないように制御する．例えば，L2-正則化(l_2-regularization)では重みベクトル $\boldsymbol{w}$ の 2-ノルムをペナルティ項として目的関数に導入する．

$$E^{\mathrm{L2LR}}(\boldsymbol{w}) = -\mathcal{L}^{\mathrm{LR}} + \frac{1}{2}C\|\boldsymbol{w}\|_2^2 = -\sum_{i=1}^{N} \log p(y^{(i)}|x^{(i)}) + \frac{1}{2}C\|\boldsymbol{w}\|_2^2 \tag{2.27}$$

ただし，$C > 0$ は L2-正則化の強さを調整するハイパー・パラメータである．C を大きくすると，訓練事例への適合よりも $\boldsymbol{w}$ の 2-ノルムの大きさを抑える作用が強くなる．C を小さくすると，訓練事例への適合を重視する傾向が強くなる．

式(2.27)の目的関数を $\boldsymbol{w}$ で微分し，確率的勾配降下法を適用すると，重みベクトル $\boldsymbol{w}$ の更新式は次式で表される．

$$\boldsymbol{w}^{(t+1)} = (1 - \eta^{(t)}C)\boldsymbol{w}^{(t)} + \eta^{(t)}y^{(i)}\{1 - p(y^{(i)}|x^{(i)})\}\boldsymbol{\phi}(x^{(i)}), \quad i = t \bmod N \tag{2.28}$$

したがって，L2-正則化付きでロジスティック回帰モデルを学習するには，Algorithm 1 の重みベクトルの更新式を式(2.28)に変更するだけでよい．正則化を行わなかった場合の更新式(2.23)と比較すると，各反復で重みベクトル $\boldsymbol{w}$ を $(1 - \eta^{(t)}C)$ 倍するという処理が加わり，重みベクトルを縮小させようとする力が働く．

また，目的関数に重みベクトル $\boldsymbol{w}$ の 1-ノルムをペナルティ項として加えた L1-正則化(l_1-regularization)もよく用いられる．

$$E^{\mathrm{L1LR}}(\boldsymbol{w}) = -\mathcal{L}^{\mathrm{LR}} + C|\boldsymbol{w}| = -\sum_{i=1}^{N} \log p(y^{(i)}|x^{(i)}) + C|\boldsymbol{w}| \tag{2.29}$$

ここで，$C > 0$ は L1-正則化の強さを調整するハイパー・パラメータである．L1-正則化は，素性の重みの値を 0 に落とそうとする作用があることが知られている．学習の結果，素性の重みが 0 になったということは，その素性は無くてもよいと学習アルゴリズムが判断したことになるため，L1-正則化は学習と素性選択を同時に行う方法としても知られている．ただ，式(2.29)の目的関数は $w_k = 0$ において微分不可能であるため，そのままでは確率的勾配降下法を適用できない．しかし，FOrward-Backward Splitting(FOBOS)などの手法を用いることにより，Algorithm 1 に近い流れの擬似コードで重みベクトルを学習できる（Duchi and Singer, 2009）．

3. 線形多クラス分類器

本節では，入力 x に対して，L 個のラベル $\mathcal{Y} = \{1, 2, \ldots, L\}$ の中から 1 つのラベル $y \in \mathcal{Y}$ を

推定する線形多クラス分類器(linear multi-class classifier)を説明する．説明を簡単にするため，ラベル y は正の整数を取ることにするが，$y=1$ は「人名」，$y=2$ は「組織名」など，ラベルの番号に任意の分類カテゴリを割り当ててよい．線形二値分類器では単一の重みベクトル $\boldsymbol{w} \in \mathbb{R}^d$ を用いたが，線形多クラス分類器では L 個のラベルに対応する重みベクトル $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L$ を用いる．線形二値分類器と同様に，入力素性ベクトル $\boldsymbol{\phi}(x) \in \mathbb{R}^d$ と重みベクトル $\boldsymbol{w}_y \in \mathbb{R}^d$ との内積を，入力 x をラベル y に分類するスコア $s(x,y)$ と定義する．

$$s(x,y) = \boldsymbol{w}_y^\intercal \boldsymbol{\phi}(x) \tag{3.1}$$

そして，入力 x に対するラベル $\hat{y}$ を次式で推定する．

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} s(x,y) = \operatorname*{argmax}_{y \in \mathcal{Y}} \boldsymbol{w}_y^\intercal \boldsymbol{\phi}(x) \tag{3.2}$$

式(3.4)は，候補となる全てのラベル $y \in \mathcal{Y}$ に関してスコア $s(x,y)$ を計算し，最も高いスコアを与えたラベル $\hat{y}$ に分類するという，単純明快なルールである．素性関数 $\boldsymbol{\phi}(x)$ の設計は，2.1 節で説明した方針のままでよい．

線形多クラス分類器の原理は，式(3.1)と(3.2)で理解しておけばよいが，より一般的には次のように定式化される．

$$s(x,y) = \boldsymbol{\lambda}^\intercal \boldsymbol{f}(x,y) = \sum_{k=1}^{K} \lambda_k f_k(x,y) \tag{3.3}$$

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y}} s(x,y) = \operatorname*{argmax}_{y \in \mathcal{Y}} \boldsymbol{\lambda}^\intercal \boldsymbol{f}(x,y) \tag{3.4}$$

この定式化での素性空間の次元数を K とすると，$\boldsymbol{\lambda} \in \mathbb{R}^K$ は重みベクトル，$\boldsymbol{f}(x,y) \in \mathbb{R}^K$ は入力 x とラベル y に関する素性ベクトルを表す．

式(3.1)と(3.2)よりも式(3.3)と(3.4)の定式化の方が一般性が高く，表現力も高いため，本稿では式(3.3)と(3.4)の定式化を採用する．しかし，式(3.3)と(3.4)の定式化では，素性空間が入力 x だけでなくラベル y にも依存するため，最初は分かりづらい．イメージを掴んでもらうため，式(2.3)に示した線形二値分類器の素性関数を多クラス分類に拡張し，ラベル $y=1$ の時のみに発火する素性関数の例を示す．

$$f_2(x,y) = \begin{cases} 1 & (x \text{ が大文字で始まる} \wedge y = 1) \\ 0 & (\text{それ以外の場合}) \end{cases} \tag{3.5}$$

f_2 は，x が大文字で始まり，かつラベルを $y=1$ と予測する場合に発火する素性である．さらに，「x が大文字で始まる」という特徴に関して，予測するラベル $y = 1, 2, \ldots, L$ 毎に素性関数を $f_2, f_3, \ldots, f_{L+1}$ などと定義する．すると，$\lambda_2, \lambda_3, \ldots, \lambda_{L+1}$ は「x が大文字で始まる」という特徴から，それぞれラベル $y = 1, 2, \ldots, L$ を予測するときの重みを表すようになる．

式(3.3)と(3.4)の定式化は，$\boldsymbol{f}(x,y)$ と $\boldsymbol{\lambda}$ を次のように定義すると，式(3.1)と(3.2)の定式化と等価になる．

$$\boldsymbol{f}(x,y) = \underbrace{\mathbf{0} \oplus \cdots \oplus \mathbf{0}}_{y-1} \oplus \boldsymbol{\phi}(x) \oplus \underbrace{\mathbf{0} \oplus \cdots \oplus \mathbf{0}}_{L-y} \tag{3.6}$$

$$\boldsymbol{\lambda} = \boldsymbol{w}_1 \oplus \boldsymbol{w}_2 \oplus \cdots \oplus \boldsymbol{w}_L \tag{3.7}$$

ただし，$\mathbf{0} \in \mathbb{R}^d$，$\oplus$ はベクトルの連結を表す．すなわち，素性空間を d 次元ずつ L 個のグループに分け，素性空間の各グループをラベル $y \in \mathcal{Y}$ に対応付ける($K = dL$)．重みベクトル $\boldsymbol{\lambda}$ は

各ラベルに対応した重みベクトル $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_L$ を並べたものである．素性関数 $\boldsymbol{f}(x, y)$ は，ラベル y に対応する素性空間グループで入力 x の素性ベクトル $\boldsymbol{\phi}(x)$ を展開し，その他のグループでは $\mathbf{0}$ とする．したがって，$s(x, y) = \boldsymbol{\lambda}^\intercal \boldsymbol{f}(x, y) = \boldsymbol{w}_y^\intercal \boldsymbol{\phi}(x)$ となり，式(3.1)と式(3.3)が一致する．

実際に多クラス分類器を構築するときは，素性関数 $\boldsymbol{f}(x, y)$ を直接設計することは稀である．代わりに，入力 x に関する素性ベクトル $\boldsymbol{\phi}(x)$ を定義し，式(3.6)の変換を用いて多クラス分類用の素性関数 $\boldsymbol{f}(x, y)$ を自動的に導出することが多い．したがって，多クラス分類器のツールを使うだけであれば素性関数 $\boldsymbol{f}(x, y)$ を意識する必要はない．しかし，式(3.3)と(3.4)の定式化を用いると，次式のように複数のラベル($y = 1$ or 2)で共通に発火する素性を定義できる(素性の次元の番号 2045 は適当である)．

$$f_{2045}(x, y) = \begin{cases} 1 & (x \text{ が大文字で始まる} \wedge (y = 1 \vee y = 2)) \\ 0 & (\text{それ以外の場合}) \end{cases} \tag{3.8}$$

例えば，$y = 1$ を人名，$y = 2$ を組織名であることにすると，これらの固有名詞で共通に発火する素性を定義したことになる．

3.1 多クラスロジスティック回帰

(二値分類である)ロジスティック回帰では，素性ベクトルと重みベクトルの内積で計算されるスコアを，シグモイド関数(sigmoid function)を使って確率値に変換していた．多クラスロジスティック回帰では，ラベル y 毎に計算されるスコア $s(x, y)$ にソフトマックス関数(softmax function)を適用し，入力 x に対してラベル y が予測される条件付き確率を求める．

$$p(y|x) = \frac{\exp s(x, y)}{\sum_{y' \in \mathcal{Y}} \exp s(x, y')} = \frac{\exp\left(\boldsymbol{\lambda}^\intercal \boldsymbol{f}(x, y)\right)}{\sum_{y' \in \mathcal{Y}} \exp\left(\boldsymbol{\lambda}^\intercal \boldsymbol{f}(x, y')\right)} \tag{3.9}$$

式(3.9)の分母は，分配関数(partition function)と呼ばれる．

$$Z(x) = \sum_{y' \in \mathcal{Y}} \exp\left(\boldsymbol{\lambda}^\intercal \boldsymbol{f}(x, y')\right) \tag{3.10}$$

なお，多クラスロジスティック回帰モデルは，最大エントロピー法(maximum entropy modeling)(Berger et al., 1996)とも呼ばれる．

多クラスロジスティック回帰モデルの学習では，(二値分類の)ロジスティック回帰モデルの理論とアルゴリズムをそのまま流用できる．言い換えれば，各事例の対数尤度の勾配さえ求めることができれば，確率的勾配降下法による最尤推定や事後確率最大化(正則化)の手順は全く同じである．ここでは，最尤推定について説明する．学習データ $\mathcal{D} = (x^{(i)}, y^{(i)})_{i=1}^N$ の各事例 $(x^{(i)}, y^{(i)})$ の対数尤度を $l(x^{(i)}, y^{(i)}) = \log p(y^{(i)}|x^{(i)})$ とすると，学習データ全体の対数尤度 $\mathcal{L}^{\mathrm{MLR}}$，最小化すべき目的関数 $E^{\mathrm{MLR}}(\boldsymbol{\lambda})$ は，

$$\mathcal{L}^{\mathrm{MLR}} = \log \prod_{i=1}^N p(y^{(i)}|x^{(i)}) = \sum_{i=1}^N \log p(y^{(i)}|x^{(i)}) = \sum_{i=1}^N l(x^{(i)}, y^{(i)}) \tag{3.11}$$

$$E^{\mathrm{MLR}}(\boldsymbol{\lambda}) = -\mathcal{L}^{\mathrm{MLR}} = -\sum_{i=1}^N l(x^{(i)}, y^{(i)}). \tag{3.12}$$

確率的勾配降下法を適用するため，各事例の対数尤度 $l(x, y)$ を展開する．

$$
\begin{aligned}
l(x,y) &= \log p(y|x) \\
&= \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(x,y) - \log \sum_{y' \in \mathcal{Y}} \exp\left(\boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(x,y')\right) \\
&= \sum_{k=1}^{K} \lambda_k f_k(x,y) - \log \sum_{y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(x,y')\right)
\end{aligned} \tag{3.13}
$$

そして，式(3.13)を λ_k に関して偏微分する．第 1 項では k に関する部分のみを考えればよい．第 2 項には合成関数，対数関数，指数関数の微分公式を適用する．

$$
\begin{aligned}
\frac{\partial l(x,y)}{\partial \lambda_k} &= \frac{\partial}{\partial \lambda_k} \left\{ \sum_{k=1}^{K} \lambda_k f_k(x,y) - \log \sum_{y' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(x,y')\right) \right\} \\
&= f_k(x,y) - \frac{\sum_{y' \in \mathcal{Y}} \left\{ \exp\left(\sum_{k=1}^{K} \lambda_k f_k(x,y')\right) \cdot f_k(x,y') \right\}}{\sum_{y'' \in \mathcal{Y}} \exp\left(\sum_{k=1}^{K} \lambda_k f_k(x,y'')\right)}
\end{aligned} \tag{3.14}
$$

$$
= f_k(x,y) - \sum_{y' \in \mathcal{Y}} p(y'|x) f_k(x,y') \tag{3.15}
$$

ゆえに，k 番目の素性に関する確率的勾配降下法の更新式は，

$$
\lambda_k^{(t+1)} = \lambda_k^{(t)} + \eta^{(t)} \left\{ f_k(x^{(i)}, y^{(i)}) - \sum_{y' \in \mathcal{Y}} p(y'|x^{(i)}) f_k(x^{(i)}, y') \right\}, \quad i = t \bmod N \tag{3.16}
$$

式(3.15)および(3.16)の解釈を考えてみたい．簡単のため，素性 f_k はどれか 1 つのラベルに関してのみ発火し(1 を返す)，それ以外のラベルに関しては発火しない(0 を返す)こととする．もし，訓練事例 $(x^{(i)}, y^{(i)})$ に関して素性 f_k が発火する場合，$f_k(x^{(i)}, y^{(i)}) = 1$ かつ $\forall y' \neq y^{(i)} : f_k(x^{(i)}, y') = 0$ であるから，

$$
\frac{\partial l(x^{(i)}, y^{(i)})}{\partial \lambda_k} = 1 - p(y^{(i)}|x^{(i)}) \geq 0. \tag{3.17}
$$

式(3.16)に当てはめて考えると，予測誤差に応じて素性 f_k の重みが増加する．これは，f_k が訓練事例に関して発火しているため，その信頼度を増やそうとしていると解釈できる．

一方，訓練事例 $(x^{(i)}, y^{(i)})$ に関して素性 f_k は発火しないものの，正解のラベル $y^{(i)}$ を別のラベル $\tilde{y}$ に置き換えると発火する場合を考える．これは，素性 f_k は入力 $x^{(i)}$ の特徴を捉えようとするが，正解とは異なるラベル $\tilde{y} \neq y^{(i)}$ を予測しようとする状況である．このとき，$f_k(x^{(i)}, \tilde{y}) = 1$ かつ $\forall y' \neq \tilde{y} : f_k(x^{(i)}, y') = 0$，よって $f_k(x^{(i)}, y^{(i)}) = 0$ であるから，

$$
\frac{\partial l(x^{(i)}, y^{(i)})}{\partial \lambda_k} = 0 - p(\tilde{y}|x^{(i)}) \leq 0 \tag{3.18}
$$

訓練事例 $(x^{(i)}, y^{(i)})$ は，入力 $x^{(i)}$ に対してラベル $\tilde{y}$ を予測することは間違いであることを示唆しているため，$p(\tilde{y}|x^{(i)})$ は予測誤差を表す．式(3.16)に当てはめて考えると，予測誤差に応じて素性 f_k の重みが減少する．これは，f_k が訓練事例に関して発火せず，正解とは異なるラベルで発火してしまうため，その信頼度を下げようとしていると解釈できる．

なお，訓練事例 $(x^{(i)}, y^{(i)})$ の入力 $x^{(i)}$ に対して，どのようなラベル $y' \in \mathcal{Y}$ に関しても素性 f_k が発火しない場合，$\forall y' \in \mathcal{Y} : f_k(x^{(i)}, y') = 0$ であるから，

$$
\frac{\partial l(x^{(i)}, y^{(i)})}{\partial \lambda_k} = 0. \tag{3.19}
$$

これは，訓練事例 $(x^{(i)}, y^{(i)})$ の入力 $x^{(i)}$ とは全く関連がない素性 f_k に関しては，その重み λ_k を更新しないことを表している．

Algorithm 2 に，確率的勾配降下法による多クラスロジスティック回帰モデルの学習の擬似コードを示す．

Algorithm 2: SGD による多クラスロジスティック回帰モデルの学習
入力: 学習データ $\mathcal{D} = (x^{(i)}, y^{(i)})_{i=1}^{N}$
出力: 重みベクトル $\boldsymbol{\lambda} \in \mathbb{R}^K$

$t \leftarrow 1$;
$\boldsymbol{\lambda} = \mathbf{0}$;
for epoch $\leftarrow 1$ **to** T **do**
  for $i \leftarrow 1$ **to** N **do**
    $\eta \leftarrow 1/t$(※他の計算方法でも可);
    for $k \leftarrow 1$ **to** K **do**
      $\lambda_k \leftarrow \lambda_k + \eta \left\{ f_k(x^{(i)}, y^{(i)}) - \sum_{y' \in \mathcal{Y}} p(y'|x^{(i)}) f_k(x^{(i)}, y') \right\}$;
    end
    $t \leftarrow t + 1$;
  end
end

4. 条件付き確率場

多クラスロジスティック回帰では，ラベル y は単一の確率変数であった．これに対し，条件付き確率場(conditional random fields)（Lafferty et al., 2001)では，複数の確率変数を同時に予測する．予測される確率変数は入力 x に依存するだけでなく，系列や木構造などのグラフ構造上でマルコフ性に従うと仮定する．入力 x も構造を持つと仮定してもよいが，入力と出力の構造は一致しなくてもよい．

自然言語の単語や文は無秩序に並んでいるのではなく，何らかの規則性を持っている．その規則性を分類モデルに取り込むことで，タスクの予測精度の向上が期待できる．例えば，英語では「文頭の単語は名詞になりやすい」「冠詞の後には名詞や形容詞が続きやすい」などの規則性があり，この規則性は単語列から品詞列を推定するのに役立つ．

本稿では，条件付き確率場の代表例として，系列 $\boldsymbol{x} = x_1, \ldots, x_M$ が与えられた時，ラベル列 $\boldsymbol{y} = y_1, \ldots, y_M$ を予測する系列ラベリング(sequential labeling)問題を扱う(M は系列の要素数)．先ほどの品詞タグ付けの例では，入力 $\boldsymbol{x}$ が単語列，ラベル列 $\boldsymbol{y}$ が品詞ラベル列となる．ここで，隣り合うラベル同士 y_m, y_{m+1} $(m = 1, \ldots, M-1)$ の依存関係(線形連鎖一次マルコフ性)を考慮し，ラベル列 $\boldsymbol{y}$ を予測することを考える．品詞タグ付けの例では，「直前の品詞の予測結果が冠詞ならば現在の単語の品詞は名詞や形容詞になりやすい」などの依存関係を考慮することになる．

条件付き確率場の予測モデルは，多クラス線形分類器の入力 x を系列 $\boldsymbol{x}$ に，ラベル y をラベル列 $\boldsymbol{y}$ に置き換えたものになる．線形多クラス分類器(式(3.3))と同様に，入力系列 $\boldsymbol{x}$ のラベル系列 $\boldsymbol{y}$ を予測するスコア $s(\boldsymbol{x}, \boldsymbol{y})$ を，素性ベクトル $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^K$ と重みベクトル $\boldsymbol{\lambda} \in \mathbb{R}^K$ の内積で計算する．素性関数 $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$ は，系列 $\boldsymbol{x}$ とラベル列 $\boldsymbol{y}$ の全体から素性ベクトルを取り出す関数である．

$$s(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\lambda}^\mathsf{T} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{k=1}^{K} \lambda_k f_k(\boldsymbol{x}, \boldsymbol{y}) \tag{4.1}$$

式(3.2)と同様に，入力系列 $\boldsymbol{x}$ に対するラベル列 $\hat{\boldsymbol{y}}$ を次式で推定する．

$$\hat{\boldsymbol{y}} = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} s(\boldsymbol{x}, \boldsymbol{y}) = \operatorname*{argmax}_{\boldsymbol{y} \in \mathcal{Y}} \boldsymbol{\lambda}^\mathsf{T} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) \tag{4.2}$$

ここで，各ラベル $y_m \in \boldsymbol{y}$ が取りうるラベルの集合を $\mathcal{Y}$ とすると，ラベル列の候補集合は $\mathcal{Y} = \mathcal{Y}^M$ である．

式(3.2)で考慮すべきラベルの候補数は $|\mathcal{Y}|$ であったが，式(4.2)ではラベル列の候補は $|\mathcal{Y}|^M$ 通りである．例えば，ラベルの種類数 $|\mathcal{Y}| = 9$ とし，長さ $M = |\boldsymbol{x}| = 10$ のラベル列を予測する場合でも，ラベル列の候補数は $3,486,784,401$ に膨れ上がる．したがって，式(4.1)をラベル列の全候補 $\mathcal{Y}$ に関して計算し，最大のスコアを与えたラベル系列 $\hat{\boldsymbol{y}}$ を求めることは事実上不可能である．式(4.2)を効率よく求めるアルゴリズムは，4.2 節で紹介する．

式(3.9)と同様に，ラベル列 $\boldsymbol{y}$ 毎に計算されるスコア $s(\boldsymbol{x}, \boldsymbol{y})$ にソフトマックス関数を適用し，入力系列 $\boldsymbol{x}$ に対してラベル列 $\boldsymbol{y}$ が予測される条件付き確率を求める．

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp s(\boldsymbol{x}, \boldsymbol{y})}{\sum_{\boldsymbol{y'} \in \mathcal{Y}} \exp s(\boldsymbol{x}, \boldsymbol{y'})} = \frac{\exp \left(\boldsymbol{\lambda}^\mathsf{T} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})\right)}{\sum_{\boldsymbol{y'} \in \mathcal{Y}} \exp \left(\boldsymbol{\lambda}^\mathsf{T} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y'})\right)} \tag{4.3}$$

ここで，式(4.3)の分母(分配関数)の計算には，ラベル列の全候補 $\boldsymbol{y'} \in \mathcal{Y}$ に関する内積とその和が必要である．

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y'} \in \mathcal{Y}} \exp \left(\boldsymbol{\lambda}^\mathsf{T} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y'})\right) \tag{4.4}$$

式(4.4)を愚直に計算すると，指数オーダー($|\mathcal{Y}|^M$ 回)の内積計算と加算演算が必要になり，分配関数の値を求めることも事実上不可能である．式(4.4)を効率よく計算する方法は，4.3 節で説明する．

条件付き確率場の学習では，(多クラス)ロジスティック回帰モデルの学習の理論とアルゴリズムをそのまま流用できる．すなわち，各訓練事例の対数尤度さえ求めることができれば，確率的勾配降下法による最尤推定や事後確率最大化(正則化)の手順は全く同じである．学習データ $\mathcal{D} = (\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})_{i=1}^N$ の各事例 $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$ の対数尤度を $l(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) = \log p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)})$ とすると，学習データ全体の対数尤度 $\mathcal{L}^{\mathrm{MLR}}$ は，

$$\mathcal{L}^{\mathrm{MLR}} = \log \prod_{i=1}^{N} p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) = \sum_{i=1}^{N} \log p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) = \sum_{i=1}^{N} l(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)}) \tag{4.5}$$

確率勾配降下法による重みベクトル $\boldsymbol{\lambda}$ の更新式は，式(2.17)と同一である．

$$\boldsymbol{\lambda}^{(t+1)} = \boldsymbol{\lambda}^{(t)} + \eta^{(t)} \frac{\partial l(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})}{\partial \boldsymbol{\lambda}^{(t)}}, \quad i = t \bmod N \tag{4.6}$$

そこで，式(4.6)の更新式で必要な勾配を求める．多クラスロジスティック回帰から条件付き確率場に拡張する際に変更したのは，入力とラベルの型だけなので，式(3.15)の導出を再利用すると，重み λ_k に関する偏微分は次式で与えられる．

$$\frac{\partial l(\boldsymbol{x}, \boldsymbol{y})}{\partial \lambda_k} = f_k(\boldsymbol{x}, \boldsymbol{y}) - \sum_{\boldsymbol{y'} \in \mathcal{Y}} p(\boldsymbol{y'}|\boldsymbol{x}) f_k(\boldsymbol{x}, \boldsymbol{y'}) \tag{4.7}$$

式(4.7)の第 2 項は，ラベル列の全候補 $\boldsymbol{y'} \in \mathcal{Y}$ において現在の確率モデルが素性関数 f_k を発火

させる期待値である．ラベル列の全候補 $\boldsymbol{y}' \in \mathcal{Y}$ が必要であることに加えて，素性関数 f_k が発火するかどうかが $\boldsymbol{y}'$ に依存しているため，この値を求めることも容易ではない．

まとめると，条件付き確率場は多クラスロジスティック回帰の入力 x を入力系列 $\boldsymbol{x}$ に，ラベル y をラベル列 $\boldsymbol{y}$ に置換しただけのモデルであるが，計算上の困難が 3 つある．

(1) スコアを最大にするラベル列 $\hat{\boldsymbol{y}}$ の予測: 式(4.2)
(2) 分配関数 $Z(\boldsymbol{x})$ の計算: 式(4.4)
(3) 素性関数のモデルにおける発火の期待値: 式(4.7)

これらを効率よく計算するための鍵は，いずれも，ラベル列の一次マルコフ性に基づく素性関数 $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$ の設計と，動的計画法である．以降では，これらについて説明していく．

4.1 線形連鎖 1 次マルコフ素性

本稿では，入力系列 $\boldsymbol{x}$ が与えられた時，ラベル列に関して一次マルコフ性を仮定する．すなわち，位置 m のラベル y_m の予測結果は，入力 x_m，隣接するラベル y_{m-1}，および y_{m+1} に依存すると仮定する．この依存関係は，x_m, y_{m-1}, y_m を引き数とした素性関数で記述できる．例えば，現在の単語が “Ltd” で，直前の単語が組織名の途中(I-ORG)で，現在の単語も組織名(I-ORG)である場合に発火する素性は，次式で表される．

$$\pi_k^{(\mathrm{xyy})}(x_m, y_{m-1}, y_m) = \begin{cases} 1 & (x_m \text{ が “Ltd”} \wedge y_{m-1} \text{ が I-ORG} \wedge y_m \text{ が I-ORG}) \\ 0 & (\text{それ以外の場合}) \end{cases} \tag{4.8}$$

なお，この素性関数は 1 つの入力 x_m，2 つのラベル y_{m-1}, y_m に依存するので，(xyy) という印を付けてある．

また，x_m，y_m，y_{m-1} のどれかに依存しない素性を考えてもよい．例えば，直前のラベル y_{m-1} に関わらず，現在の単語 “Ltd” と組織名(I-ORG)の依存関係を表現する素性は，次式で表される．

$$\pi_k^{(\mathrm{xy})}(x_m, y_m) = \begin{cases} 1 & (x_m \text{ が “Ltd”} \wedge y_m \text{ が I-ORG}) \\ 0 & (\text{それ以外の場合}) \end{cases} \tag{4.9}$$

さらに，位置 m を任意とし，隣接するラベル間の事象のみを記述した素性は，次式で表される．

$$\pi_k^{(\mathrm{yy})}(y_{m-1}, y_m) = \begin{cases} 1 & (y_{m-1} \text{ が B-PERSON} \wedge y_m \text{ が I-PERSON}) \\ 0 & (\text{それ以外の場合}) \end{cases} \tag{4.10}$$

線形連鎖一次マルコフ素性に基づく条件付き確率場では，式(4.8)，式(4.9)，式(4.10)のいずれかの型を持つ素性関数を定義することになる．

ところで，線形多クラス分類器では，入力 x と予測ラベル y の両方で条件付けされた素性関数 $\boldsymbol{f}(x, y)$ を直接設計することは稀で，入力 x に関する素性ベクトル $\boldsymbol{\phi}(x)$ を定義し，素性関数 $\boldsymbol{f}(x, y)$ を自動的に導出することを説明した．条件付き確率場でも，入力 x に関する素性ベクトル $\boldsymbol{\phi}(x)$ の設計に注力し，式(4.8)，式(4.9)，式(4.10)の形の素性関数を自動的に導出するのが一般的である．

例えば，式(4.9)の形は多クラス線形分類器のものと同一であり，(3)節で説明した方法で自動的に導出できる．また，式(4.10)の形の素性は，すべてのラベルの組み合わせ $(y_{m-1}, y_m) \in \mathcal{Y}^2$ に対して定義するか，訓練事例に出現する隣り合うラベルの組み合わせに対して定義すればよい．式(4.8)の形の素性は，以上の方法を組み合わせることによって導出できる．このように，条件付き確率場のツールを使うだけであれば，素性関数を直接設計しなくてもよい．

さて，式(4.8)の形の素性関数を K_1 個定義し，その全ての素性関数の値を K_1 次元ベクトルで返す素性関数ベクトルを $\boldsymbol{\pi}^{(\mathrm{xyy})}(x_m, y_{m-1}, y_m) \in \mathbb{R}^{K_1}$ と書く．同様に，式(4.9)と式(4.10)の素性関数をそれぞれ，K_2 個，K_3 個定義し，素性関数ベクトルで表現したものを，それぞれ，$\boldsymbol{\pi}^{(\mathrm{xy})}(x_m, y_m) \in \mathbb{R}^{K_2}$，$\boldsymbol{\pi}^{(\mathrm{yy})}(y_{m-1}, y_m) \in \mathbb{R}^{K_3}$ と書く(ただし，$K = K_1 + K_2 + K_3$)．線形連鎖一次マルコフ素性は，式(4.8)，式(4.9)，式(4.10)のいずれかで表されるので，時刻 m における素性関数ベクトル $\boldsymbol{\pi}(x_m, y_{m-1}, y_m) \in \mathbb{R}^K$ は次式で表される．

$$\boldsymbol{\pi}(x_m, y_{m-1}, y_m) = \boldsymbol{\pi}^{(\mathrm{xyy})}(x_m, y_{m-1}, y_m) \oplus \boldsymbol{\pi}^{(\mathrm{xy})}(x_m, y_m) \oplus \boldsymbol{\pi}^{(\mathrm{yy})}(y_{m-1}, y_m) \tag{4.11}$$

$\boldsymbol{\pi}(x_m, y_{m-1}, y_m)$ は，位置 m における素性ベクトルを表すので，局所素性ベクトルと呼ぶ．この記法に基づくと，線形連鎖一次マルコフ素性による大域素性ベクトルは次式で表される．

$$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) = \sum_{m=1}^{M} \boldsymbol{\pi}(x_m, y_{m-1}, y_m) \tag{4.12}$$

ただし，$m = 1$ のとき，y_0 は系列の先頭を表す特殊なシンボル BOS とし，x_1 と y_1 のみに依存する素性とするか，系列の先頭のラベルを捉える素性としてもよい．同様に，系列の末尾のラベルを捉える素性を導入してもよい．

4.2 ラベル列の推定

式(4.2)を効率よく求める方法を説明するため，以下の量を定義する．

$$r_{\boldsymbol{x}}(m, i, j) = \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{\pi}(x_m, i, j) \tag{4.13}$$

$r_{\boldsymbol{x}}(m, i, j)$ は，入力系列 $\boldsymbol{x}$ が与えられた時，$y_{m-1} = i$ かつ $y_m = j$ の時に発火する素性関数の重みの和である．

式(4.11)，式(4.12)，式(4.13)より，式(4.1)は次のように展開できる．

$$s(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\lambda}^{\mathsf{T}} \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\lambda}^{\mathsf{T}} \sum_{m=1}^{M} \boldsymbol{\pi}(x_m, y_{m-1}, y_m) = \sum_{m=1}^{M} r_{\boldsymbol{x}}(m, y_{m-1}, y_m) \tag{4.14}$$

これは，系列全体の大域素性ベクトル $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y})$ を先に求めてから重み $\boldsymbol{\lambda}$ との内積を計算する代わりに，先に各時刻 $m = 1, 2, \ldots, M$ において局所素性ベクトル $\boldsymbol{\pi}(x_m, y_{m-1}, y_m)$ と重みベクトル $\boldsymbol{\lambda}$ の内積を求め，その和を計算しても結果が同じであることを示している．式(4.14)の計算は大変そうに見えるが，素性関数は式(4.8)，式(4.9)，式(4.10)のいずれかの形をしているため，入力 $\boldsymbol{x}$ の時刻 m における特徴量に対応した素性や，ラベルのペアに関する素性の次元番号を列挙し，その次元番号に対応する重み取り出し，その和を計算するだけでよい．

図 2 は，式(4.14)の計算をラティスとして図示したものである．横軸は時刻 $m = 1, 2, 3$，縦軸はラベル y を表し，図中の各ノード (m, j) は時刻 m のラベル $y_m = j$ である事象を表している．また，各点をつなぐ線(エッジ)は，$r_{\boldsymbol{x}}(m, y_{m-1}, y_m)$ の値を示している．この図を用いると，$m = 1$ から $m = 3$ に至る任意の経路がラベル列 $\boldsymbol{y}$ を表し，その経路上の重みの和がスコア $s(\boldsymbol{x}, \boldsymbol{y})$ である．したがって，式(4.2)を求める問題は，図 2 で $m = 1$ から $m = 3$ に至る経路の中で，スコアが最大となる経路を求める問題に帰着する．

この問題は，最短経路問題と同様に，以下の漸化式で求めることができる．

$$\psi_{\boldsymbol{x}}(m, j) = \begin{cases} r_{\boldsymbol{x}}(1, \mathsf{EOS}, j) & (m = 1 \text{ のとき}) \\ \max_{i \in \mathcal{Y}} \{\psi_{\boldsymbol{x}}(m-1, i) + r_{\boldsymbol{x}}(m, i, j)\} & (m > 1 \text{ のとき}) \end{cases} \tag{4.15}$$

ここで，$\psi_{\boldsymbol{x}}(m, j)$ は図 2 の左端からノード (m, j) に至る経路の中で，最大のスコアを記録した

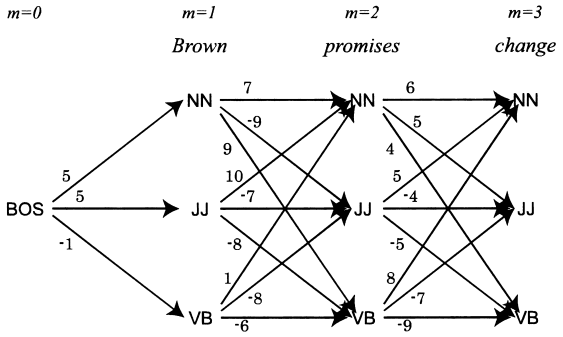

図 2. ラティスで表現した素性の重み．エッジの数字は $r_{\boldsymbol{x}}(m,i,j)$ の値を表す．

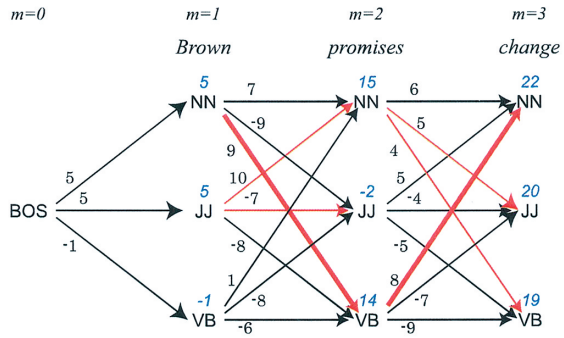

図 3. ビタビ・アルゴリズムの動作例．ノードは各時点 m におけるラベル y_m を表す．エッジの数字は $r_{\boldsymbol{x}}(m,i,j)$ の値を表す．ノードの左上の青色の数字は $\psi_{\boldsymbol{x}}(m,j)$ を，赤色のエッジはその値がどこに由来するのかを示している．

ものである．式(4.15)は，ノード $(m-1,i)$ を経由してノード (m,j) に至る経路のスコアの最大値は，$\psi_{\boldsymbol{x}}(m-1,i)+r_{\boldsymbol{x}}(m,i,j)$ で与えられることを利用し，経路のスコアを最大にする経由地 $(m-1,i)$ を選ぶことで，ノード (m,j) に至る経路のスコアの最大値を求めている．

式(4.15)の漸化式を使い，$\psi_{\boldsymbol{x}}(m,j)$ の値を $m=1$ から $m=M$ まで求めたとき，スコアが最大となる経路の終点 $(M,\hat{y}_M)$ は次式で表される．

$$\hat{y}_M = \operatorname*{argmax}_{i\in\mathcal{Y}} \psi_{\boldsymbol{x}}(M,i) \tag{4.16}$$

そして，この終点に至るまでの経路を逆向きに辿ることで，式(4.2)の解を求めることができる．すなわち，$m=M-1$ から $m=1$ まで，以下の漸化式を適用していけばよい．

$$\hat{y}_m = \operatorname*{argmax}_{i\in\mathcal{Y}} \{\psi_{\boldsymbol{x}}(m,i)+r_{\boldsymbol{x}}(m+1,i,\hat{y}_{m+1})\} \tag{4.17}$$

実際には，式(4.17)を計算しなくても，式(4.15)の漸化式を適用する際に選ばれた経由地への逆向きのリンクを保持しておけば，終点 $(M,\hat{y}_M)$ から逆向きのリンクを辿っていくだけでよい．この計算過程を示したのが図 3 である．図中のノード (m,i) の上に $\psi_{\boldsymbol{x}}(m,i)$ の値を示し，その値がどのノードから来たのか，赤色の矢印で示してある．式(4.16)に従い，$m=3$ のラベルを

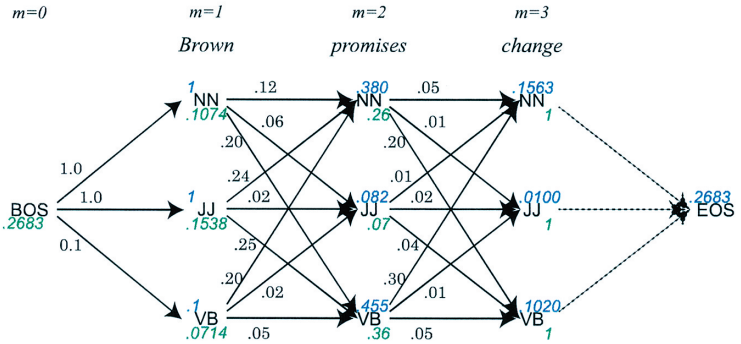

図 4. 前向き・後ろ向きアルゴリズムの計算例．ノードは各時点 m におけるラベル y_m を表す．エッジの数字は $\check{r}_{\boldsymbol{x}}(m,i,j)$ の値を表す．ノードの左上の青色の数字は $\alpha_{\boldsymbol{x}}(m,j)$ を，右下の緑色の数字は $\beta_{\boldsymbol{x}}(m,i)$ を表す．式(4.22)で求めた分配関数の値を BOS の右下に，式(4.24)で求めた分配関数の値を EOS の左上に示した．前向き・後ろ向きのどちらを使っても，分配関数の値が同じになることを確認できる．

NN と決定したら，そのノードに至る矢印を逆向きに辿ることで，式(4.2)の解，すなわち予測されるラベル列 $\hat{\boldsymbol{y}}$ を求めることができる．図 3 の場合は，NN VB NN が解となる．このアルゴリズムは，ビタビ・アルゴリズム(Viterbi algorithm)と呼ばれる．

4.3 分配関数の計算

次に，分配関数(式(4.4))を効率よく求める方法を説明する．まず，$r_{\boldsymbol{x}}(m,i,j)$ の指数を取った関数 $\check{r}_{\boldsymbol{x}}(m,i,j)$ を定義する．

$$\check{r}_{\boldsymbol{x}}(m,i,j) = \exp r_{\boldsymbol{x}}(m,i,j) \tag{4.18}$$

分配関数の定義を $\check{r}_{\boldsymbol{x}}(m,i,j)$ の式として書き直す．

$$\begin{aligned} Z(\boldsymbol{x}) &= \sum_{y_1,\dots,y_t \in \mathcal{Y}^L} \exp s(\boldsymbol{x},\boldsymbol{y}) \\ &= \sum_{y_1,\dots,y_t \in \mathcal{Y}^L} \prod_{m=1}^{M} \check{r}_{\boldsymbol{x}}(m,y_{m-1},y_m) \end{aligned} \tag{4.19}$$

式(4.19)を展開し，共通部分を $m=1$ から括り出すと，次のようになる．

$$Z(\boldsymbol{x}) = \sum_{y_1=1}^{L} \check{r}_{\boldsymbol{x}}(1,\mathsf{BOS},y_1) \sum_{y_2=1}^{L} \check{r}_{\boldsymbol{x}}(2,y_1,y_2) \sum_{y_3=1}^{L} \dots \sum_{y_M=1}^{L} \check{r}_{\boldsymbol{x}}(M,y_{M-1},y_M) \tag{4.20}$$

ここで，式(4.20)の形に着目すると，右側($m=M$)から左側($m=1$)に向かって，$\check{r}_{\boldsymbol{x}}(m,i,j)$ の和を求め，その値を左側に伝播させながら，和の計算を進めればよいことが分かる．

この計算過程を図示したのが図 4 である．あるノード (m,i) の右側から終端($m=M$)に至る経路上のエッジにある $\check{r}_{\boldsymbol{x}}(m,i,j)$ の積を計算し，全ての経路で和を取ったものを $\beta_{\boldsymbol{x}}(m,i)$ と定義すると，以下の漸化式が成り立つ．

$$\beta_{\boldsymbol{x}}(m,i) = \begin{cases} 1 & (m=M \text{ のとき}) \\ \displaystyle\sum_{j=1}^{L} \check{r}_{\boldsymbol{x}}(m,i,j)\beta_{\boldsymbol{x}}(m+1,j) & (m<M \text{ のとき}) \end{cases} \tag{4.21}$$

この漸化式で各ノードの $\beta_{\boldsymbol{x}}(m,i)$ の値を求めたものを，ノードの下側に緑色で示した．式(4.20)との対比により，分配関数は次式で求まることが分かる．

$$Z(\boldsymbol{x}) = \beta_{\boldsymbol{x}}(0, \mathsf{BOS}) \tag{4.22}$$

一方，式(4.19)を展開するとき，$m = M$ から括り出すと，次のようになる．

$$Z(\boldsymbol{x}) = \sum_{y_M=1}^{L} \sum_{y_{M-1}=1}^{L} \check{r}_{\boldsymbol{x}}(M, y_{M-1}, y_M) \sum_{y_{M-2}=1}^{L} \cdots \sum_{y_1=1}^{L} \check{r}_{\boldsymbol{x}}(1, y_0, y_1)$$

ここで，左端($m = 1$)からノード (m,j) に至る経路上にあるエッジの $\check{r}_{\boldsymbol{x}}(m,i,j)$ の積を計算し，全ての経路で和を取ったものを $\alpha_{\boldsymbol{x}}(m,j)$ と定義すると，以下の漸化式が得られる．

$$\alpha_{\boldsymbol{x}}(m,j) = \begin{cases} \check{r}_{\boldsymbol{x}}(1, \mathsf{BOS}, j) & (m = 1 \text{ のとき}) \\ \displaystyle\sum_{i=1}^{L} \check{r}_{\boldsymbol{x}}(m,i,j)\alpha_{\boldsymbol{x}}(m-1,i) & (1 < m \text{ のとき}) \end{cases} \tag{4.23}$$

この漸化式で各ノードの $\alpha_{\boldsymbol{x}}(m,i)$ の値を求めたものを，ノードの上側に青色で示した．式(4.20)との対比により，分配関数は次式で求めることもできる．

$$Z(\boldsymbol{x}) = \sum_{i=1}^{L} \alpha_{\boldsymbol{x}}(M,i) \tag{4.24}$$

分配関数の値を求めるだけであれば，$\alpha_{\boldsymbol{x}}(m,i)$ と $\beta_{\boldsymbol{x}}(m,i)$ のどちらを用いてもよい．ただ，次節で説明する周辺確率の計算には，これらの両方の値が必要になる．このように，$\alpha_{\boldsymbol{x}}(m,i)$, $\beta_{\boldsymbol{x}}(m,i)$ の値を求めるアルゴリズムのことを，前向き・後ろ向きアルゴリズムと呼ぶ．

4.4 素性関数のモデルにおける発火の期待値

最後に，式(4.7)の計算方法を説明する．式(4.11)を用い，大域素性ベクトルを局所素性ベクトルに変換すると，式(4.7)は以下のように変形できる．

$$\begin{aligned} \frac{\partial l(\boldsymbol{x},\boldsymbol{y})}{\partial \lambda_k} &= f_k(\boldsymbol{x},\boldsymbol{y}) - \sum_{\boldsymbol{y'} \in \mathcal{Y}} p(\boldsymbol{y'}|\boldsymbol{x}) f_k(\boldsymbol{x},\boldsymbol{y'}) \\ &= \sum_{m=1}^{M} \left\{ \pi_k(x_m, y_{m-1}, y_m) - \sum_{i \in \mathcal{Y}} \sum_{j \in \mathcal{Y}} p(i,j|\boldsymbol{x},m) \pi_k(x_m, i, j) \right\} \end{aligned} \tag{4.25}$$

ここで，$p(i,j|\boldsymbol{x},m)$ は，与えられた入力系列 $\boldsymbol{x}$ に対して，現在のモデルで出力の系列を予測したとき，時刻 $m-1$ のラベルが i で，時刻 m のラベルが j と予測される確率(周辺確率)を表す．したがって，任意の m, i, j に関して，$p(i,j|\boldsymbol{x},m)$ が計算できればよい．

ここで，$\alpha_{\boldsymbol{x}}(m,i)$ が左端($m = 1$)からノード (m,i) に至る経路のスコアの和，$\beta_{\boldsymbol{x}}(m,i)$ がノード (m,i) から右端($m = M$)に至るスコアの和であることを利用すると，$p(i,j|\boldsymbol{x},m)$ を次式で求めることができる．

$$\begin{aligned} p(i,j|\boldsymbol{x},m) &= \frac{(m-1,i)\text{—}(m,j) \text{ を通過する全経路のスコアの和}}{Z(\boldsymbol{x})} \\ &= \frac{\alpha_{\boldsymbol{x}}(m-1,i)\check{r}_{\boldsymbol{x}}(m-1,i,j)\beta_{\boldsymbol{x}}(m,j)}{Z(\boldsymbol{x})} \end{aligned} \tag{4.26}$$

ゆえに，4.3 節で分配関数を計算する時に用いた $\alpha_{\boldsymbol{x}}(m,i)$ および $\beta_{\boldsymbol{x}}(m,i)$ の値を保存しておけば，周辺確率 $p(i,j|\boldsymbol{x},m)$ の値を簡単に求めることができる．

4.5 その他の研究動向

2001 年の登場以降，条件付き確率場は様々な言語処理タスクに適用されてきた．典型例として，浅い句構造解析（Sha and Pereira, 2003），固有表現抽出（McCallum and Li, 2003），形態素解析（Kudo et al., 2004），情報抽出（Sarawagi and Cohen, 2005），構文解析（Finkel et al., 2008），ゾーニング（Hirohata et al., 2008）などがある．条件付き確率場を実装したツールとして，CRF++ [1], CRFsuite [2], Wapiti [3] などが有名である．

高性能の系列予測器を作るには，大量の訓練データを用意しなければならない．そこで，条件付き確率場の理論をうまく利用して，訓練データの構築を支援する研究もある．坪井ら（Tsuboi et al., 2008）は，入力系列の一部のみに正解ラベルを付与した訓練データや，正解を一つに絞り込むことができずに複数の正解ラベルが付与された訓練データから，重みベクトルを学習する手法を提案した．本手法は後に中国語の単語分割に適用され（Liu et al., 2014），CRFsuite のソースコードをベースにした実装が公開されている [4]．

Settles and Craven（2008）は，少量の訓練データと正解が付与されていない大量の事例があるとき，どの事例に正解を付与すべきかを示唆する能動学習（active learning）の戦略を比較・検討した．この研究では，少量の訓練データで学習したモデルを使い，正解が付与されていない事例の入力 $\boldsymbol{x}$ に対してラベル列 $\hat{\boldsymbol{y}}$ を予測し，予測されたラベル列の確率推定値 $p(\hat{\boldsymbol{y}}|\boldsymbol{x})$ が低い事例の正解を付与したり（least confidence），ラベルの周辺確率から計算されるラベル列のエントロピーの高い事例に対して正解を付与する戦略（token entropy）などを検討している．CRFsuite では，予測されたラベル列の確率は `-p`，`--probability` オプションで，予測されたラベル列の周辺確率は `-i`，`--marginal` オプションで得ることができる．また，CRFsuite の API（`Tagger::marginal`）を呼び出すことで，全てのラベルに関する周辺確率を計算できる．これらを活用することで，Settles らが比較・検討した能動学習の戦略を実装できる．

本稿では，ラベルに関する 1 次マルコフ性を仮定し，隣り合うラベルの依存関係を考慮した．しかし，言語のデータを扱っていると，2 単語先や 3 単語先など，距離が離れたラベルの依存関係をモデル化したくなることがある．最も単純な解決策は，2 次（連続する 3 個のラベル）や 3 次（連続する 4 個のラベル）など，2 次以上のマルコフ性を仮定した素性関数を導入することである．ただ，素性の次数を l とすると，ビタビ・アルゴリズムや前向き・後ろ向き・アルゴリズムの計算量は $|\mathcal{Y}|^{l+1}$ であるため，次数を上げると計算量が急増してしまう．代わりに，連続する複数のラベルをひと塊として扱うセミ・マルコフ条件付き確率場（Sarawagi and Cohen, 2005）が提案されている．また，入力列から出力列を予測するまでの間に，潜在変数のラベル列を導入することで，離れた距離の依存関係をモデル化しようとする研究もある（Morency et al., 2007）（Sun et al., 2008）．なお，CRFsuite の実装を拡張して，高次のマルコフ性，セミ・マルコフ性，木構造予測などを実現した実装 [5] も公開されている．

最近では，Recurrent Neural Network（RNN）や Long Short-Term Memory（LSTM）などの深層ニューラルネットワークに条件付き確率場のアイディアを導入し，品詞タグ付けや固有表現抽出を実現した研究も報告されている（Zhou and Xu, 2015）（Lample et al., 2016）．これらの研究は，潜在変数のラベル列の代わりに中間層の分散表現を用いるとともに，LSTM の記憶セルなどで離れた距離の依存関係をモデル化している．また，深層ニューラルネットワークに基づく条件付き確率場では，タスクに依存した素性関数を人間が設計しなくても，文字や単語の分散表現を学習することにより，特徴抽出を自動化できる．実際，Lample et al.（2016）は文字に関する素性や辞書などの外部知識を用いずに，深層ニューラルネットワークだけで獲得した素性を用いて，固有表現抽出の最高性能を達成できることを報告した．

5. おわりに

本稿では，系列データにおける条件付き確率場の理論と実践を解説した．条件付き確率場は，多クラスロジスティック回帰に基づいているため，これらの理論や学習方法を復習した．また，ラベル列のマルコフ性を仮定した素性関数と，ビタビ・アルゴリズムや前向き・後ろ向き・アルゴリズムなどの動的計画法でラベル列の予測とパラメータの学習を効率化する方法を詳説した．

注．

1) https://taku910.github.io/crfpp/
2) http://www.chokkan.org/software/crfsuite/
3) https://wapiti.limsi.fr/
4) https://github.com/ExpResults/partial-crfsuite
5) https://github.com/WladimirSidorenko/CRFSuite

参 考 文 献

Berger, A., Pietra, S. D. and Pietra, V. D. (1996). A maximum entropy approach to natural language processing, *Computational Linguestics*, **22**(1), 39–71.

Duchi, J. and Singer, Y. (2009). Efficient online and batch learning using forward backward splitting, *Journal of Machine Learning Research*, **10**, 2899–2934.

Finkel, J. R., Kleeman, A. and Manning, C. D. (2008). Efficient, feature-based, conditional random field parsing, *Proceedings of ACL-08*: *HLT*, 959–967.

Hirohata, K., Okazaki, N., Ananiadou, S. and Ishizuka, M. (2008). Identifying sections in scientific abstracts using conditional random fields, *Proceedings of the Third International Joint Conference on Natural Language Processing* (*IJCNLP 2008*): *Volume-I*, 381–388.

Kudo, T., Yamamoto, K. and Matsumoto, Y. (2004). Applying conditional random fields to Japanese morphological analysis, *Proceedings of EMNLP 2004*, 230–237.

Lafferty, J. D., McCallum, A. and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data, *Proceedings of the Eighteenth International Conference on Machine Learning* (*ICML 2001*), 282–289.

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K. and Dyer, C. (2016). Neural architectures for named entity recognition, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*: *Human Language Technologies* (*NAACL-HLT 2016*), 260–270.

Liu, Y., Zhang, Y., Che, W., Liu, T. and Wu, F. (2014). Domain adaptation for CRF-based Chinese word segmentation using free annotations, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (*EMNLP*), 864–874.

McCallum, A. and Li, W. (2003). Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons, *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, 188–191.

Morency, L.-P., Quattoni, A. and Darrell, T. (2007). Latent-dynamic discriminative models for continuous gesture recognition, *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR '07*), 1–8.

Sarawagi, S. and Cohen, W. W. (2005). Semi-markov conditional random fields for information extraction, *Advances in Neural Information Processing Systems 17* (*NIPS 2005*), 1185–1192.

Settles, B. and Craven, M. (2008). An analysis of active learning strategies for sequence labeling tasks,

Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, 1070–1079.

Sha, F. and Pereira, F. (2003). Shallow parsing with conditional random fields, *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, 134–141.

Sun, X., Morency, L.-P., Okanohara, D., Tsuruoka, Y. and Tsujii, J. (2008). Modeling latent-dynamic in shallow parsing: A latent conditional model with improved inference, *Proceedings of the 22nd International Conference on Computational Linguistics* (*Coling 2008*), 841–848.

Tsuboi, Y., Kashima, H., Mori, S., Oda, H. and Matsumoto, Y. (2008). Training conditional random fields using incomplete annotations, *Proceedings of the 22nd International Conference on Computational Linguistics* (*Coling 2008*), 897–904.

Zhou, J. and Xu, W. (2015). End-to-end learning of semantic role labeling using recurrent neural networks, *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing* (*ACL-IJCNLP 2015*) (*Volume 1*: *Long Papers*), 1127–1137.

200 Proceedings of the Institute of Statistical Mathematics Vol. 64, No. 2, 179–200 (2016)

Theory and Practice of Conditional Random Fields

Naoaki Okazaki

Graduate School of Information Sciences, Tohoku University

Most tasks of Natural Language Processing are formalized as a prediction problem of an output for a given input. Assuming that an input and output have a structure such as a sequence and tree, which is a natural assumption for a language, we can formalize more tasks as the prediction problem. This paper explains Conditional Random Fields (CRF) where an input and output are in the form of a sequence. In order to apply the multi-class logistic regression to the sequential labeling problem, CRF introduces feature functions that assume the Markov property for a label sequence and facilitates an efficient inference and parameter estimation by using dynamic programming. Therefore, this paper reviews the fundamental theories of logistic regression, feature functions, training with stochastic gradient descent, regularization, etc., and describes the overall theory of CRF. In addition, it covers recent research topics and practices including active learning for CRF, learning from partially-annotated supervision data, and models with deep neural networks.

Key words: Conditional Random Fields, logistic regression, stochastic gradient descent.