言語とテキストの機械学習

持 橋　大 地

1. 身近にあふれるテキスト

世界のIT化が進むにつれ, われわれの周りには電子化されたテキストがあふれるようになりました. われわれは毎日, 大量の電子メールやSNSの文章を読み書きしていますし, Webページや SNSにある情報源は, そのほとんどがテキストからなっています. こうしたテキストは, 数学的にはどのようにモデル化されるでしょうか.

いま, 日本語も英語のように単語に分けられているとすると[*1)], 例えば, 藤原正彦『若き数学者のアメリカ』に出てくる単語を辞書順に横軸に, 頻度を縦軸にとってプロットすると, 図1のようになります. 作品の全単語数を N (この場合 $N=104232$), 単語 i の頻度を n_i と表せば, i の確率 p_i はほぼ

$$p_i = \frac{n_i}{N} \tag{1}$$

と思っていいでしょう. 例えば, この作品で「アメリカ」は231回現れるので,

$$p_{\text{アメリカ}} = \frac{231}{104232} \simeq 0.0022$$

になります. 図1はこうした単語の確率を並べたもの, つまり離散的な確率分布ということになります. 語彙が V 個のとき, これを $\mathbf{p}=(p_1, p_2, \ldots, p_V)$ と書きましょう. こうした $\mathbf{p}$ は離散分布, またはサンプル数を1に固定した多項分布とよばれます.

*1) 日本語の場合, MeCabのような形態素解析器とよばれるソフトウェアを使うと, `% mecab -O wakati input.txt` のようにして入力テキストを単語に分割できます.

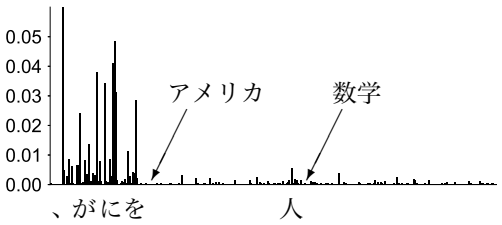


図1　藤原正彦『若き数学者のアメリカ』における単語の確率分布. 横軸は辞書順の単語です.

$\mathbf{p}$ は当然, 作品の内容によって異なりますから, $\mathbf{p}$ の確率分布, つまり確率分布の確率分布を考えるのが自然です. このための最も簡単な分布がディリクレ分布とよばれる確率分布で,

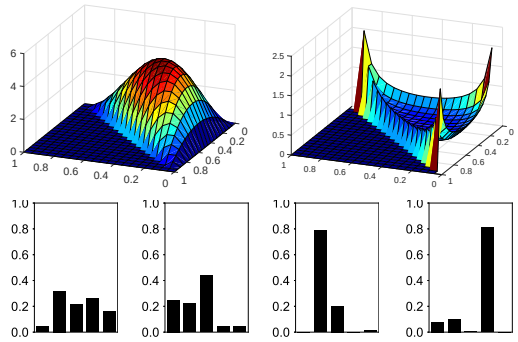
$$p(\mathbf{p}) = \frac{\Gamma(\sum_{i=1}^{V}\alpha_i)}{\prod_{i=1}^{V}\Gamma(\alpha_i)} \prod_{i=1}^{V} p_i^{\alpha_i - 1} \tag{2}$$

と表され, パラメータ $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_V)$ の値によって図2のようにさまざまな形をとります. $\alpha_i > 1$ のときは上に凸ですが, $\alpha_i = 1$ のときは式(2)により一様分布, $\alpha_i < 1$ のときは下に凸な分布となり, 一部の p_i が非常に大きく, 残りが0に近い偏った分布となります. 図1でみたように, 言語の場合はほぼ $\alpha_i < 1$ が当てはまるのは明らかでしょう. ディリクレ分布に従う $\mathbf{p}$ の期待値は, $\boldsymbol{\alpha}$ を総和1に正規化した

$$E[\mathbf{p}|\boldsymbol{\alpha}] = \left(\frac{\alpha_1}{\alpha}, \frac{\alpha_2}{\alpha}, \ldots, \frac{\alpha_V}{\alpha}\right) \tag{3}$$

となります. ここで, $\alpha = \sum_i \alpha_i$ と書きました.

ディリクレ分布を使うと, あるテキスト $\mathbf{w} =$



$\boldsymbol{\alpha}=(2,2,\cdots,2)$　　$\boldsymbol{\alpha}=(0.2,0.2,\cdots,0.2)$

図 2　ディリクレ分布の確率密度 (3 次元の場合) と, そこからサンプルされた多項分布 $\mathbf{p}$ の例 (5 次元の場合).

$w_1 w_2 \cdots w_N$) は次のようにして生成されたとモデル化できます.

- まず多項分布 $\mathbf{p} \sim p(\mathbf{p})$ が生成され,
- $n = 1, \ldots, N$ について, $w_n \sim \mathbf{p}$ を生成.

このとき, $\mathbf{w}$ の中で単語 i が現れた回数を n_i とおけば, $p(\mathbf{w}|\mathbf{p}) = \prod_{i=1}^{V} p_i^{n_i}$ ですから, 逆に $\mathbf{w}$ を知ったときの $\mathbf{p}$ の確率分布は, ベイズの定理によれば, 式 (2) を用いると

$$
\begin{aligned}
p(\mathbf{p}|\mathbf{w}) &\propto p(\mathbf{w}|\mathbf{p})p(\mathbf{p}) && (4)\\
&= \prod_{i=1}^{V} p_i^{n_i} \cdot \frac{\Gamma(\sum_i \alpha_i)}{\prod_i \Gamma(\alpha_i)} \prod_{i=1}^{V} p_i^{\alpha_i - 1}\\
&\propto \prod_{i=1}^{V} p_i^{\alpha_i + n_i - 1} && (5)
\end{aligned}
$$

となり, これは $(\alpha_1 + n_1, \ldots, \alpha_V + n_V)$ を新しいパラメータにもつディリクレ分布です. よって, その期待値は式 (3) から,

$$
E[p_i|\mathbf{w}] = \frac{n_i + \alpha_i}{\sum_i (n_i + \alpha_i)} = \frac{n_i + \alpha_i}{N + \alpha} \qquad (6)
$$

となります. これは, 頻度がゼロ ($n_i = 0$) の単語にも, $\alpha_i/(N+\alpha)$ の確率を与えることに注意しましょう. これにより, 式 (1) と異なって, たまたま頻度が 0 でもすべての単語に正の確率を与えることができます.

なお, $\mathbf{p}$ について期待値をとって積分消去してみると, $\mathbf{w}$ の確率は

$$
\begin{aligned}
p(\mathbf{w}) &= \int p(\mathbf{w}|\mathbf{p})p(\mathbf{p})d\mathbf{p} && (7)\\
&= \int \prod_{i=1}^{V} p_i^{n_i} \cdot \frac{\Gamma(\sum_{i=1}^{V} \alpha_i)}{\prod_{i=1}^{V} \Gamma(\alpha_i)} \prod_{i=1}^{V} p_i^{\alpha_i - 1} d\mathbf{p}\\
&= \frac{\Gamma(\sum_i \alpha_i)}{\Gamma(N + \sum_i \alpha_i)} \prod_{i=1}^{V} \frac{\Gamma(\alpha_i + n_i)}{\Gamma(\alpha_i)} && (8)
\end{aligned}
$$

と表すことができます. これをディリクレ–多項分布あるいは **Pólya** 分布とよびます[1].

式 (8) は $\boldsymbol{\alpha}$ が与えられた下での一つの $\mathbf{w}$ の確率であり, これから $\boldsymbol{\alpha}$ を知ることはできません[*2]. しかし, テキストが複数あれば最適な $\boldsymbol{\alpha}$ を知ることができます. いま, D 個のテキスト $\mathbf{w}_1, \ldots, \mathbf{w}_D$ があり, d 番目のテキスト $\mathbf{w}_d$ での単語 i の頻度を n_{di} とおけば, 全体の確率は式 (8) の積ですから,

$$
\begin{aligned}
&p(\mathbf{w}_1, \ldots, \mathbf{w}_D)\\
&= \prod_{d=1}^{D} \frac{\Gamma(\sum_i \alpha_i)}{\Gamma(N_d + \sum_i \alpha_i)} \prod_{i=1}^{V} \frac{\Gamma(\alpha_i + n_{di})}{\Gamma(\alpha_i)} \qquad (9)
\end{aligned}
$$

となります. 式 (9) は二階微分をとると $\boldsymbol{\alpha}$ について上に凸であることがわかりますので, 例えば Newton 法を用いて求めることができます. 導出は若干面倒なため, 文献[1] に譲りますが, $\Psi(x) = d/dx \log \Gamma(x)$ として

$$
\alpha_i' = \alpha_i \cdot \frac{\sum_{d=1}^{D} \Psi(n_{di} + \alpha_i) - \Psi(\alpha_i)}{\sum_{d=1}^{D} \Psi(N_d + \sum_i \alpha_i) - \Psi(\sum_i \alpha_i)} \qquad (10)
$$

を各 $i = 1, \ldots, V$ について収束するまで繰り返すことにより, $\boldsymbol{\alpha}$ を最適化することができます.

たとえば『若き数学者の〜』には「カナダ」は一度も現れませんが, 標準的な新聞記事 10000 記事を使って最適化すると, $\alpha_{\text{カナダ}} = 0.03$, $\sum_i \alpha_i = 454.11$ となりましたので, 式 (6) による確率は 0 ではなく,

$$
p_{\text{カナダ}} = \frac{0 + 0.03}{104232 + 454.11} = 2.87 \times 10^{-7}
$$

と計算することができます.

*2)　単独の $\mathbf{w}$ について式 (8) を $\boldsymbol{\alpha}$ に関して最大化すると, $\alpha_i = 0$ が解として得られてしまいます.

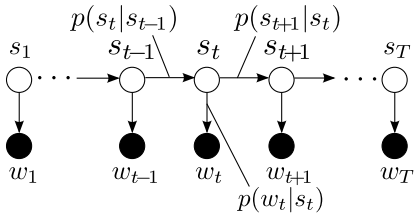


図 3　隠れマルコフモデル (HMM) の構造.

2. 潜在状態の導入

上の話では簡単のため, 各単語は独立に生成されるとしていました. 実際には単語の間には依存関係があり, 上の話をその場合に拡張する (nグラムモデルとよばれています) ことも可能ですが, ここでは別の生成過程を考えてみましょう.

言語の単語には一般に, 動詞や名詞, 形容詞のように品詞があると言われています. 英語の場合は主語である名詞の後には動詞が, 形容詞の後には修飾される名詞が続くことが多いので, たとえば「名詞–動詞–冠詞–形容詞–名詞」という品詞列がまずあり, それぞれの品詞から表層の単語が生成されて “She reads a long book.” のような文が生まれた, という単純なモデルが考えられるでしょう. このモデルは, 機械学習や統計学では隠れマルコフモデル (HMM) とよばれています.

数学的には, 隠れマルコフモデルは次のように表現されます. 隠れ状態が K 個あるとし, テキスト $\mathbf{w} = w_1, \ldots, w_N$ の各時刻 n での状態を $s_n \in \{1, \ldots, K\}$ とおきましょう. このとき, $\mathbf{w}$ と隠れ状態の全体 $\mathbf{s}$ の同時確率は図 3 のように,

$$p(\mathbf{w}, \mathbf{s}) = \prod_{n=1}^{N} p(w_n|s_n)p(s_n|s_{n-1}) \quad (11)$$

と表すことができます. ここで $p(w_n|s_n)$ は状態 s_n から単語 w_n への出力確率で, その全体は s_n ごとに, 図 1 でみたような語彙上の確率分布になります. $p(s_n|s_{n-1})$ は状態 s_{n-1} から s_n に遷移する遷移確率で, 全体は $K \times K$ 次元の行列です.

すべての単語 w_n についてその隠れ状態 s_n が分かっていれば, 出力確率と遷移確率は s_n ごとにそこから出力された単語と, 次に遷移した状態の頻度を数えて式 (1) または式 (6) を計算すれば得られます. しかし, 通常何百万個もある各単語に人手で状態番号 $1, \ldots, K$ を振るのは大変な作業ですし, 未知の言語であればそもそも不可能です. こうした場合, s_n を潜在変数とみなし, 式 (11) の確率を最大化する s_n を推定していくことになります. 機械学習の分野では, こうしたモデルは潜在変数モデルとよばれています.

1		**2**		**3**	
she	432	the	1026	was	277
to	387	a	473	had	126
i	324	her	116	said	113
it	265	very	84	be	77
you	218	its	50	is	73
alice	166	my	46	went	58
and	147	no	44	were	56
they	76	his	44	see	52
there	61	this	39	could	52
he	55	an	37	know	50
that	39	your	36	thought	44
who	37	as	31	herself	42

4		**5**		**6**	
and	466	way	45	little	92
of	343	mouse	41	great	23
in	262	thing	39	very	22
said	174	queen	37	long	22
to	163	head	36	large	22
as	163	cat	35	right	20
that	125	hatter	34	same	17
for	123	duchess	34	good	17
at	122	well	31	white	11
but	121	time	31	other	11
with	114	tone	28	poor	10
on	83	rabbit	28	first	10

表 1　『不思議の国のアリス』で無限 HMM の隠れ状態に割り当てられた単語とその回数.「主語」「動詞」「一般名詞」などの概念が, 自動的に学習されていることがわかります.

潜在変数 $s_1, \ldots, s_N$ はどのようにして推定すればよいでしょうか. 単純にはすべての可能性を考えて, 式 (11) による確率を比較すればよさそうですが, この組み合わせは K^N 個 (たとえば $K = 10, N = 1000000$ の場合 $10^{1000000}$ 個) あり, 現実的に計算不可能です. 古典的にはこのために, Baum-Welch アルゴリズムとよばれる動的計画法

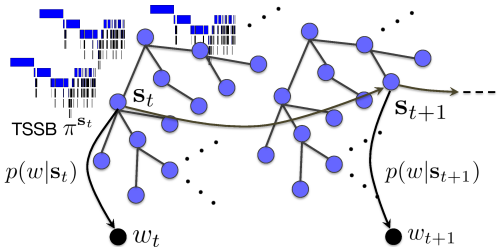


図 4　無限木構造 HMM(iTHMM) の概要図. 無限個の分岐と深さをもつ隠れた木構造上の状態遷移を, 単語列のみから学習します.

を用いた方法がありますが[2], この方法は統計モデルの最尤推定を行うため, 容易に品質の悪い局所解に陥ってしまいます. かわりに, $s_1, \ldots, s_N$ をランダムに初期化した後で, 図 3 からわかる $s_n \in \{1, \ldots, K\}$ の事後確率

$$p(s_n|\mathbf{s}\backslash s_n, \mathbf{w}) \propto p(w_n|s_n)p(s_{n+1}|s_n)p(s_n|s_{n-1})$$

に従ってランダムに s_n を更新する **Gibbs サンプリング**を行うことで, 局所解に陥りにくい隠れマルコフモデルのベイズ推定が行え, 高い性能を示すことが知られています. 上の式は, 統計物理学でイジング模型の計算などに用いられる熱浴法と基本的に同じものです.

なお, 言語での「品詞」の数にあたる潜在状態の数 K は, ディリクレ分布の無限次元版ともいえるディリクレ過程を使った無限隠れマルコフモデルを使うと推定することができます[3]. 表 1 に, 無限 HMM によって『不思議の国のアリス』のテキストの各単語に割り当てられた状態とその回数を示しました. 最近筆者は, 無限隠れマルコフモデルをさらに拡張し, 潜在状態を $s_n = [2\,1\,3], [4], [5\,12\,7\,2]$ のように木構造で推定可能にした無限木構造隠れマルコフモデル (iTHMM) を提案・実装しました[4]. iTHMM では図 4 のように, 無限個の分岐と無限の深さをもつ潜在状態の木構造から, データを最もよく説明する木とその間の状態遷移を学習します. この場合, 状態遷移は通常の HMM のように $K \times K$ 次元の行列で書くことはできず, 無限木構造の各ノードに, 次の時刻の木構造の各ノードへの確率分布が存在する

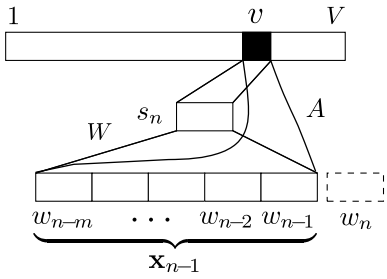


図 5　2003 年に最初に提案されたニューラルネット言語モデル[5] の概要図.

きわめて複雑なモデルになります. ノードは無限個あるため, 通常の方法では学習することができませんが, 統計学の分野で提案されたスライスサンプリングと遡及的サンプリングという方法を巧妙に組み合わせることで, 無限の潜在状態の中から適切な状態を Gibbs サンプリングすることができます.

3. 非線形モデル

上の隠れマルコフモデルでは, 状態 s_n は $1, \ldots, K$ または $[2\,1\,5]$ のような離散値をとっていました. より表現力を高めるために, これを D 次元の実数ベクトルとしてみたらどうでしょうか. これが現在の深層学習ブームのきっかけとなった, 2003 年に発表された**ニューラル言語モデル**[5] です.

ニューラル言語モデルでは, 各時刻での $s_n \in \mathbb{R}^D$ に対し, 単語 w にも同じ次元のベクトル表現 $\vec{w} \in \mathbb{R}^D$ があると考えます. s_n からの w の確率は, 基本的に内積

$$\vec{w}^T s_n \tag{12}$$

に比例して定まる, としましょう. 式 (12) は負になることもありますから, e の肩に乗せて

$$p(w|s_n) = \frac{\exp\left(\vec{w}^T s_n + b_w\right)}{Z} \tag{13}$$

と定義すれば, 単語の出力確率が得られます. ここで b_w は単語 w の平均的な確率の対数にあたるバイアス項, 分母は分子のすべての単語についての和である分配関数 $Z = \sum_{w=1}^{V} \exp\left(\vec{w}^T s_n + b_w\right)$

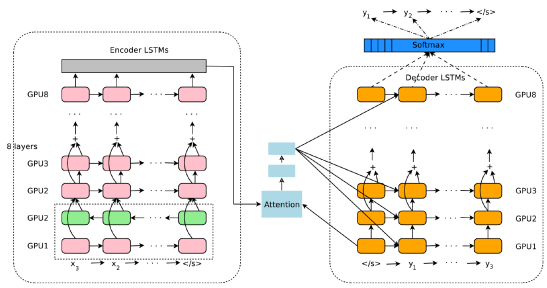


図 6　Google ニューラル機械翻訳の概要図 (文献 7) から引用). □ は LSTM の状態です.

です. この定義から, 式 (12) は一種のハミルトニアンともいえ, 言語の統計モデルは統計物理とも繋がりを持っていることがわかります.

状態 s_n はどのように決めたらよいのでしょうか. ニューラル言語モデルでは, 図 5 のように直近の m 個の単語ベクトル $\vec{w}_{n-1}, \ldots, \vec{w}_{n-m}$ を並べたベクトル $\mathbf{x}_{n-1}$ を行列 W で D 次元に線形変換し, tanh で二値化した

$$s_n = \tanh(W\mathbf{x}_{n-1}) \tag{14}$$

を状態とし, 式 (13) で b_w に $\mathbf{x}$ の線形変換 $A\mathbf{x}$ を加えた式で単語の確率を計算します.

その後 2007 年に, n からの位置 j ごとに異なる行列 C_j を用いて, 二値化なしに

$$s_n = \sum_{j=1}^{m} C_j \vec{w}_{n-j} \tag{15}$$

とするモデルがより性能が高いことが示されました[6]. 式 (15) を式 (13) に代入すれば,

$$p(w|w_{n-1}, \ldots, w_{n-m}) = \frac{\exp\left(\sum_{j=1}^{m} \vec{w}^T C_j \vec{w}_{n-j} + b_w\right)}{Z} \tag{16}$$

の形になり, 分子の対数が式 (15) の双線形形式になることから, このモデルは対数双線形 **(Log-bilinear)** 言語モデルとよばれています.

こうしたモデルのパラメータ θ は, 観測語 w_n が得られるごとに, 最も基本的には確率的勾配法

$$\theta^{(t)} = \theta^{(t-1)} + \epsilon \frac{\partial}{\partial \theta} \log p(w_n|w_{n-1}, \ldots, w_{t-m})$$

によって最適化します. 上式での微分の計算には, 式 (13) の分母で通常は数万以上になる語彙全体について和をとる分配関数の計算が各時刻 n ごとに必要になるため, 非常に計算量が大きいことに注意してください.

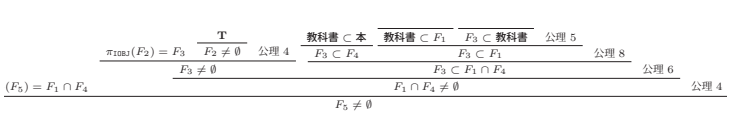
現在ではさらに, LSTM とよばれるより複雑なニューラルネットを使うと, 言語らしい文を簡単に生成できることが知られています. 特に, 機械翻訳は原言語 (たとえば英語) の文 $\mathbf{x}$ を聞いたとき, 対応する目標言語 (たとえば日本語) の文 $\mathbf{y}$ を生成する問題, つまり $p(\mathbf{y}|\mathbf{x})$ からの生成ととらえることができますから, 翻訳関係にある文対 $\{\langle \mathbf{x}_n, \mathbf{y}_n \rangle | n = 1, \ldots, N\}$ を集め, 全体の確率 $\prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{x}_n)$ を最大にする統計モデルを学習すれば, $\mathbf{x}$ から $\mathbf{y}$ への機械翻訳を行うことができます.

Google 社は実際に, 2016 年に機械翻訳システムをニューラルネット化し, 翻訳精度が大きく改善されたことが話題になりました. そのシステムは図 6 のように, 入力文の単語列に対応する LSTM の隠れ状態をさらに上位の LSTM の入力とするカスケードを 8 段も重ね, 最終層をまた 8 段の LSTM で目標言語の単語列 $\mathbf{y}$ に戻す, という驚異的な構造を持っています. その他にも様々な改善が取り入れられており[7], 仏語 → 英語の 3600 万文対を使用して, 学習には 96 個の GPU を用いて高速化しても 6 日間を要したと報告されています.

4. 論理表現の機械学習

今までは, テキストの実体としての言葉の確率モデルについてみてきました. 実際には, テキストは何か伝えたい命題を表す意味内容を持っており, 意味内容を捉えることがより重要であるはずです. 古典的には意味解析とよばれるこの分野も, 機械学習の高度化によってより数理的に捉えられるようになりつつあります.

例えば, “He reads the book.” と “The book reads him.” は意味が異なりますし (後者は通常あり得ないでしょう), 前者から「誰かに読まれる本がある」という命題は真であることが導かれるは



$\pi_{\text{SBJ}}(learn)$	$\pi_{\text{OBJ}}(learn)$	$\pi_{\text{about}}(learn)$
teacher	*skill*	*otherness*
skill	*lesson*	*intimacy*
he	*technique*	*femininity*
she	*experience*	*life*

図 7　集合論に基づく論理的推論 (左) と, ベクトル表現と論理の組み合わせによる, 動詞に役割別に関係している語の学習 (右).

ずです.

これを数学的に表すために, 田ら [*3)] は最近, DCS と呼ばれる意味表現を拡張し, 集合に対する論理演算として意味を表現する方法を示しました[8]. 入力文が係り受け解析されているとき, 例えば「太郎が本を渡す」は図 8 上段のように解析され, これを集合

$$渡す \cap (太郎_{\text{SBJ}} \times 本_{\text{OBJ}})$$

で表します. これは,「渡す」を行いうる主語 (SBJ) と目的語 (OBJ) の集合の直積と, 具体的に「太郎」と「本」の意味する集合の直積との交わりをとった集合が命題の内容であることを意味します (偽であれば空集合になります). こうして全てを集合で表すと, 集合に対する公理と包含関係から推論が行えます. たとえば, 図 7 左では「太郎が教科書を読む」と「竹取物語が教科書にある」から,「太郎は竹取物語のある教科書を読む」が真であることが推論されています.

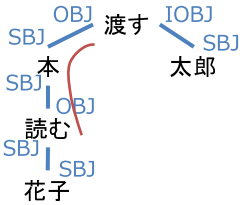


図 8　DCS 木.

上では「本」や「読む」といった言葉の意味を表す集合を事前に準備しておく必要がありましたが, 田らはさらにそれらをベクトルで表し, 教師なし学習する Vector-valued DCS[9] を提案しました. この方法では,「花子が読む本を太郎に渡す」という文を表す図 8 の DCS 木において, そうして渡された本が存在することから, 部分パスに当たる

$$v_{読む}^{T} M_{\text{OBJ}} M_{\text{SBJ}}^{-1} M_{SBJ} M_{\text{OBJ}}^{-1} u_{渡す}$$

の値を大きくするように, 言葉のベクトル $v_{読む}$, $u_{渡す}$, 主語や目的語を取り出す行列 M_{SBJ}, M_{OBJ} を最適化します. これにより, 同じ "learn" でも主語や目的語, 補語になるものを図 7 右のように区別して学習でき, 複合動詞も表現できるなど多くのメリットがあり, 意味関係のタスクでも高い精度を出すことが報告されています. 言語の深い数理モデルは, まだ研究が始まったばかりです.

*3)　田然氏 (産総研) は, 数学 (代数幾何) で博士号を持っています.

参考文献

1) Thomas P. Minka. Estimating a Dirichlet distribution, 2000. http://research.microsoft.com/~minka/papers/dirichlet/.
2) Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
3) Jurgen Van Gael, Andreas Vlachos, and Zoubin Ghahramani. The infinite HMM for unsupervised PoS tagging. In *EMNLP 2009*, pages 678–687, 2009.
4) 持橋大地, 能地宏. 無限木構造隠れ Markov モデルによる階層的品詞の教師なし学習. 情報処理学会研究報告 *2016-NL-226*, 12:1–11, 2016.
5) Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.
6) Andriy Mnih and Geoffrey Hinton. Three new graphical models for statistical language modelling. In *ICML 2007*, pages 641–648, 2007.
7) Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, and Mohammad Norouzi. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation, 2016. arXiv:1609.08144.
8) Ran Tian, Yusuke Miyao, and Takuya Matsuzaki. Logical Inference on Dependency-based Compositional Semantics. In *ACL 2014*, pages 79–89, 2014.
9) Ran Tian, Naoaki Okazaki, and Kentaro Inui. Learning Semantically and Additively Compositional Distributional Representations. In *ACL 2016*, pages 1277–1287, 2016.

(もちはし・だいち, 統計数理研究所)