学位論文　博士（工学）

# 離散分布に従う系列データの<br>高精度かつ安定な学習のための<br>Recurrent Neural Network の研究

2020 年度

慶應義塾大学大学院理工学研究科

金 井 関 利

# 目次

# 図目次

# 表目次

# 第 1 章

# 序論

本論文は

- 時系列のための深層学習モデル gated recurrent unit（GRU）の学習の安定化
- Softmax の表現能力に関する解析と，高い表現力を持つ出力関数の提案

を目的とした研究成果をまとめたものである．

## 1.1 背景

通信機器やセンサのコモディティ化とこれらのデバイスの通信規格の標準化を背景に，internet of things（IoT）市場が活性化している．IoT によって非常に多くのデバイスがインターネットにつながる．インターネットと接続されるデバイスの数は，2013 年時点で約 158 億個，2020 年には 530 億個まで増大すると推定されている．このように非常に多くのデバイスがインターネットに接続されると，音声や動画像，センサデータといった多量のデータが流通する．このように市場に流通するデータは複数の特徴量を持った高次元データであり，前後関係を持つような時系列データであることが期待される．そのため，こうした高次元で時系列のビッグデータに対する分析技術を適用したサービス創出が期待される．

近年，ビッグデータ分析技術の中で注目されている技術が深層学習（deep learning）である．深層学習は画像や音声認識，自然言語処理において人を超える，または人と近い精度を達成できるとして注目を集めている [6–9]．すでに画像検索や音声検索，機械翻訳などで深層学習は実用化されている．

今後の IoT 市場において重要となる時系列データを扱う深層学習のモデルは recurrent neural network（RNN）である [10]．RNN はニューラルネットワークのノードの出力を次の時刻のノードの入力とすることで，過去の状態を記憶するニューラルネットワークであり，時系列データの解析に適している．そのため RNN は音声認識や自然言語処理，センサデータの

解析などに用いられるが [9, 11, 12]，過去の情報を使用しないフィードフォワード型のニューラルネットワークより学習が難しい．またニューラルネットワークは多くの非線形変換，線形変換を行うため，モデルの挙動がブラックボックスであり，学習する際には多数のパラメータを試行錯誤的にチューニングして使用しなければならない．そのためデータ解析者の経験や勘を必要としており，データ解析者不足が深層学習を使ったサービス創出の問題の一つとなっている．さらに深層学習のモデルは多数のパラメータを持っており，このパラメータ数が学習や推論に必要な計算量を増大させる．そのため，この計算コストがサービスの開発や展開に大きな障害となる．

## 1.2 本論文の位置づけと目的

本論文ではニューラルネットワークを使用する際に必要な経験や勘を削減し，計算コストを増やすことなく高精度な深層学習を，より使用しやすくするような技術の確立を目的とする．

まず，時系列データの解析に用いられる RNN について，学習の難しさを解決する技術を提案する．RNN の学習には，主に back propagation through time（BPTT）が用いられるが，BPTT を利用した RNN の学習は，勾配消失や勾配爆発という現象により困難である [13, 14]．勾配消失は過去の情報が長期に渡って依存するような時系列データの学習の精度を悪化させる．この問題は long short-term memory（LSTM）や，LSTM を簡素化した gated recurrent unit（GRU）というモデル構造によって軽減されている [15, 16]．一方，勾配爆発は学習中に勾配が非常に大きな値をとり学習を破綻させてしまう問題であり，勾配クリッピングと呼ばれるヒューリスティックな方法で回避される [13]．しかし勾配クリッピングは試行錯誤的にパラメータをチューニングする必要があり，経験と勘を必要とする．そこで本論文では，勾配消失を解決するモデル GRU の挙動を解析することで勾配爆発の原因を調査し，GRU の勾配爆発を回避し，学習を安定化する技術を確立する．

次に，深層学習を使った離散分布に従うデータのモデリングに対して，深層学習の出力関数の解析を通して，モデルの出力の数とモデルの持つ状態の数の関係の一部を明らかにする．深層学習を使ったクラス分類や言語モデルなどの離散確率を扱う多くのタスクにおいて，ニューラルネットワークの出力に用いられる活性化関数は softmax である [9, 11, 16–20]．しかし [21] で softmax が離散分布の取りうる値の数が大きいようなデータ（例えば語彙数の大きな言語データ）を深層学習によってモデリングする際の精度のボトルネックとなりうることが示された（これは softmax bottleneck と呼ばれる）．本論文では，softmax bottleneck を softmax の対数の値域を解析することで再解釈し，データの高精度なモデリングに望ましい出力関数の特性を提示する．これにより，softmax に代わる新たな出力関数 sigsoftmax を提案する．既存技術では softmax bottleneck の解消に，試行錯誤の必要な調整パラメータや追加パラメータによる追加の計算量が必要であったが，sigsoftmax は試行錯誤と計算量を増やすことなく

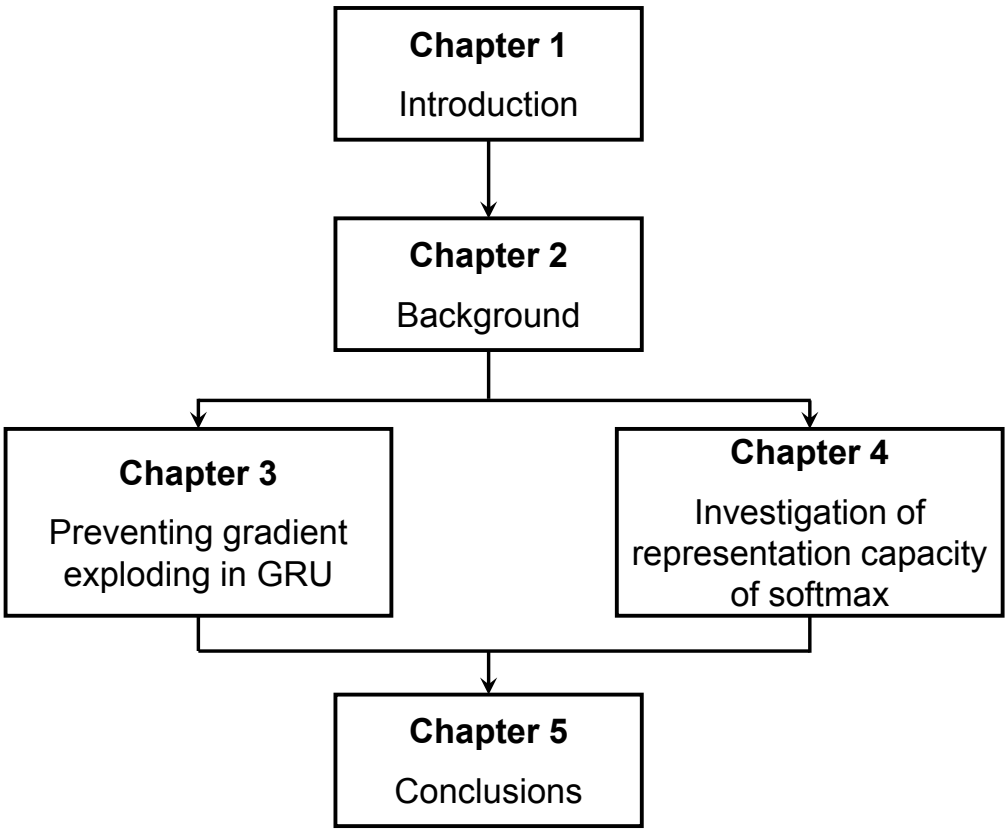



Fig. 1.1 Overview of this thesis.

解消できる．

以上の技術により，言語モデルのような取りうる値の数が大きな時系列データの解析において，試行錯誤を低減し高精度なモデリングを達成する．

## 1.3 本論文の構成

本論文の構成を Fig. 1.1 に示す．

第 2 章では，本論文の背景となる深層学習の基礎技術を紹介し，関連研究について概説する．第 3 章では，深層学習の中で RNN の一つである GRU についてそのダイナミクスを解析し，GRU の学習を安定化するための勾配爆発を抑制する学習方法について説明する．前節で述べた通り，勾配爆発によって GRU の学習は失敗してしまう．この勾配爆発は GRU の状態の振る舞いが変化する点で生じるため，そのダイナミクスを解析し，振る舞いの変化点を導出する．次に，その点を回避する学習方法とその計算量を削減する方法を提案し，実験によって評価する．第 4 章では，深層学習を使って離散分布をモデリングする際に必要な出力関数の特

性について解析する．既存の出力関数は，出力の取りうる値の数に対して状態数が少ないときに真の確率分布を表現できないことが示されている．しかし，その原因は明らかではなく，これを改善する出力関数は提案されていなかった．本論文の解析によって既存の活性化関数は，指数関数で構成されているため上記のような限界があることが明らかとなる．そして，その解消法として指数関数とシグモイド関数（sigmoid）で構成される活性化関数を提案する．提案した活性化関数と複数の活性化関数を実験的に評価し，提案法の有効性を示す．最後に，第 5 章では，本論文の結論をまとめる．

本論文で用いる記号を Table 1.1 にまとめる．また，本論文で用いる略記を Table 1.2 にまとめる．

Table 1.1 Notation.

| Notation | Description |
|---|---|
| $(\boldsymbol{x}, \boldsymbol{y})$ | Data points: $\boldsymbol{x}$ is usually input, and $y$ is output |
| $\boldsymbol{h}$ | State vector of recurrent neural networks |
| $\boldsymbol{\theta}$ | Parameter vector |
| $\boldsymbol{W}$ | Weight matrix |
| $\boldsymbol{f}(\cdot)$ | Activation function |
| $C$ | Loss function |
| $\sigma_i$ | $i$-th largest singular value |
| sigm | Sigmoid function |

Table 1.2 Important abbreviations.

| Abbreviation | Full name |
|---|---|
| DNN | Deep neural network |
| RNN | Recurrent neural network |
| GRU | Gated recurrent unit |
| BPTT | Back propagation through time |
| SGD | Stochastic gradient descent |
| ReLU | Rectified linear unit |
| MoS | Mixture of softmax |
| MoSS | Mixture of sigsoftmax |
| SVD | Singular value decomposition |

# 第 2 章

# 研究背景

## 2.1　深層学習の基礎

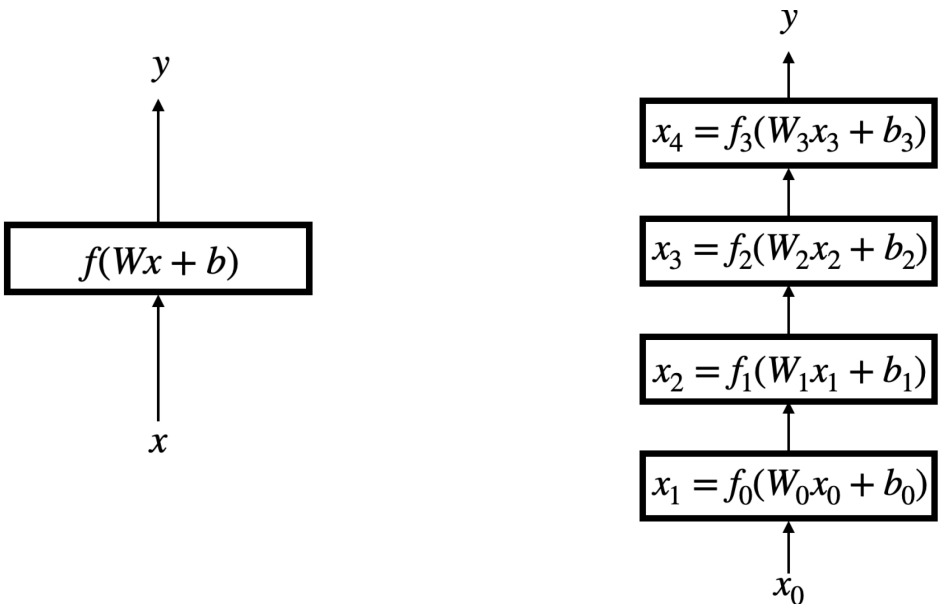
データ解析の多数のアプリケーションで深層学習が用いられている．例えば深層学習に注目を集めるきっかけとなった画像認識 [19, 22, 23] や機械翻訳 [20, 24]，音声認識 [9, 25] などで，深層学習の精度の高さを発揮している．深層学習のモデルである深層ニューラルネットワーク（deep neural networks: DNN）とはニューラルネットワークを複数層重ねた数理モデルである．各層の計算は

$$\boldsymbol{x}_{l+1} = \boldsymbol{f}_l(\boldsymbol{W}_l \boldsymbol{x}_l + \boldsymbol{b}_l) \tag{2.1}$$

である．ただし，$l$ 層の入力を $\boldsymbol{x}_l \in \mathbb{R}^{d_l \times 1}$，出力を $\boldsymbol{x}_{l+1} \in \mathbb{R}^{d_{l+1} \times 1}$ とした．$\boldsymbol{f}_l$ は活性化関数と呼ばれる非線形変換である．$\boldsymbol{W}_l \in \mathbb{R}^{d_{l+1} \times d_l}$ と $\boldsymbol{b}_l \in \mathbb{R}^{d_{l+1} \times 1}$ は学習によって最適化される重み行列とバイアスと呼ばれるパラメータである．この線形変換と非線形変換の組み合わせを $L$ 回行って最終的な出力 $\boldsymbol{y} = \boldsymbol{f}_L(\boldsymbol{x}_L)$ を生成する．入力 $\boldsymbol{x}_0$ から順に各層の変換を計算して出力 $\boldsymbol{y}$ を計算することを順伝播（forward propagation）と呼ぶ．この計算を続けて行う深層学習のモデルを Fig. 2.1 に示す．Fig. 2.1(a) に示すように 1 層毎の計算を長方形と矢印で表現し，これを 4 層（$L = 4$）積んだモデルが Fig. 2.1(b) である．モデルの構造はタスクによって異なり，例えば画像認識においては畳み込みニューラルネットワーク（convolutional neural network: CNN）と呼ばれる特殊な重みの構造が用いられ，時系列データのモデリングでは後述する recurrent neural network（RNN）がよく用いられる．

### 2.1.1　DNN の活性化関数

ニューラルネットワークで活性化関数としてよく用いられる非線形関数を紹介する．式 (2.1) において活性化関数 $\boldsymbol{f}_l$ を恒等写像とし，線形変換のみによって構成されたニューラルネットワークは線形分離不可能なデータを表現できない．非線形関数を用いることでニューラ




(a) One-layer neural network.

(b) Multi-layer neural network.

Fig. 2.1 Illustration of deep neural networks. Each box corresponds to one layer, i.e., a linear transformation and a nonlinear activation function. $x_*$ is input, and $y$ is output. $W_*$ and $b_*$ are weight matrices and biases, respectively. $f_:$ is activation function.

ルネットワークは万能近似能力を持ち，任意の関数を任意の精度で表現できるようになる．主な活性化関数とその導関数を Fig. 2.2 に示す．図の上段は活性化関数の入力 $x$ に対する出力を，下段は入力 $x$ に対する導関数の出力を示しており，左から rectified linear unit (ReLU), sigmoid，tanh，そして softplus である．それぞれについて以下で説明する．

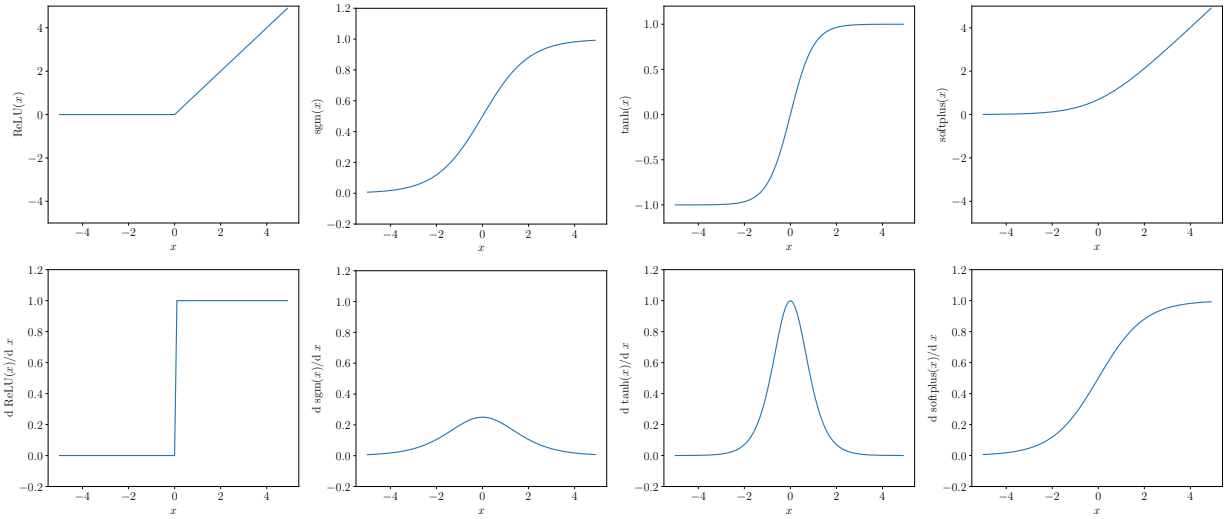
### (a) Rectified linear unit (ReLU) [26]

ReLU は

$$f(x) = \max(x, 0) \tag{2.2}$$

である．これはランプ関数とも呼ばれる．入力が負のときは 0，正の入力に対しては恒等写像となる．これにより，原点以外における，その微分が

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \begin{cases} 1, & \text{if} \quad x > 0 \\ 0, & \text{if} \quad x < 0 \end{cases} \tag{2.3}$$




(a) ReLU (b) Sigmoid (c) tanh (d) Softplus

Fig. 2.2 Activation functions (top) and their derivatives (bottom).

となり，正の入力に対して微分が常に 1 になる．これにより後述する勾配計算において勾配の過度な増大や減少を抑えることができる．ReLU は $x = 0$ で微分不可能となるが，深層学習では一般に $\mathrm{d}f(x)/\mathrm{d}x \approx 0$ として計算する．ReLU の原点における劣微分は $\partial f(0) = [0, 1]$ なので $0 \in \partial f(0)$ は $f$ の劣勾配である．なおフィードフォワード型のニューラルネットワークと比べ，RNN では ReLU を使用することが少ない．

### (b) Sigmoid

Sigmoid は

$$f(x) = \mathrm{sigm}(x) = \frac{1}{1 + \exp(-x)} \tag{2.4}$$

である．Sigmoid の微分は 1 以下の小さな値となり勾配を小さくするため，フィードフォワード型の DNN ではあまり用いられないが，RNN ではよく用いられる．

### (c) Hyperbolic tangent

tanh は

$$f(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)} \tag{2.5}$$

である．これも sigmoid と同様にフィードフォワード型の DNN ではあまり用いられないが，RNN ではよく用いられる．

(d)　Softplus

Softplus は

$$f(x) = \log(1 + \exp(x)) \tag{2.6}$$

である．これは $\lim_{x\to+\infty} f(x)/x = 1$，$\lim_{x\to-\infty} f(x) = 0$ となり，ReLU を滑らかな関数で近似したものとみなすことができる．その他に ReLU の亜種として leaky ReLU[27] や PReLU[28] などの活性化関数が提案されているが，本論文では使用しないため割愛する．

### 2.1.2　深層学習の学習方法

深層学習の教師あり学習について説明する．教師あり学習では，事前に教師データと呼ばれるデータをモデルに学習させ，学習済みモデルを用いてクラス分類や言語モデルといった推論を行う．深層学習の教師あり学習とは目的関数を $J_{\mathrm{e}}$ とすると，一般に次の最適化によってパラメータを求めることである．

$$\min_{\boldsymbol{\theta}} J_{\mathrm{e}} = \min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{j=1}^{N} C(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}; \boldsymbol{\theta}). \tag{2.7}$$

ただし，$\boldsymbol{\theta}$ は $\boldsymbol{W}$ などのモデルのすべてのパラメータを並べたベクトルであり，$\boldsymbol{x}^{(j)}$ と $\boldsymbol{y}^{(j)}$ はそれぞれ $j$ 番目の教師データの入力と出力である． $C(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}; \boldsymbol{\theta})$ は後述する負の対数尤度などの損失関数であり，$N$ はデータ数である．

データ数 $N$ が非常に大きな大規模データを扱う深層学習において，式 (2.7) の最適化は確率的勾配降下法（stochastic gradient descent: SGD）によって解かれる．SGD はランダムにサンプリングされたミニバッチと呼ばれるデータの集合に対する勾配の平均にしたがってパラメータを反復更新する．$\tau$ 回目のパラメータ更新は

$$\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \nabla_{\boldsymbol{\theta}} \frac{1}{|D_\tau|} \sum_{(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}) \in D_\tau} C^{(j)}, \tag{2.8}$$

で与えられる．ただし，$D_\tau$ はランダムにサンプリングされたミニバッチで $|D_\tau|$ はそのサイズを表し，$\eta$ は SGD の学習率，$C^{(j)} = C(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}; \boldsymbol{\theta})$ である．ミニバッチのサンプリングは一般にデータ全体を選ぶまで重複しないようにサンプリングされる．このミニバッチ 1 つに対する勾配更新を 1 反復，ミニバッチとしてデータ点がすべて一度ずつ更新に使用されることを 1 エポックと数えて，事前に決めたエポック数か停止条件を満たすまで学習させるのが DNN を使った一般的な学習である．

勾配の計算は微分の連鎖律によって各層の微分の積をとることで行われるが，この計算はニューラルネットワークでは逆誤差伝搬（back propagation）と呼ばれる．例えば，$l$ 層のパ

ラメータ（$l = 1, \ldots, L$）を $\boldsymbol{\theta}_l$ とし，状態を $\boldsymbol{x}_l = \boldsymbol{f}_{l-1}(\boldsymbol{x}_{l-1}, \boldsymbol{\theta}_{l-1})$，損失関数を $C$ とする．このとき $\boldsymbol{\theta}_l$ に関する $C$ の勾配は

$$\nabla_{\boldsymbol{\theta}_l} C = \frac{\partial C}{\partial \boldsymbol{x}_L} \frac{\partial \boldsymbol{x}_L}{\partial \boldsymbol{x}_{L-1}} \frac{\partial \boldsymbol{x}_{L-1}}{\partial \boldsymbol{x}_{L-2}} \cdots \frac{\partial \boldsymbol{x}_{l+2}}{\partial \boldsymbol{x}_{l+1}} \frac{\partial \boldsymbol{x}_{l+1}}{\partial \boldsymbol{\theta}_l} = \frac{\partial C}{\partial \boldsymbol{x}_L} \prod_{i=l+1}^{L-1} \frac{\partial \boldsymbol{x}_{i+1}}{\partial \boldsymbol{x}_i} \frac{\partial \boldsymbol{x}_{l+1}}{\partial \boldsymbol{\theta}_l} \tag{2.9}$$

となる．なお $\boldsymbol{x}_{l+1} = \boldsymbol{f}_l(\boldsymbol{W}_l \boldsymbol{x}_l + \boldsymbol{b}_l)$ とすると，$\dfrac{\partial \boldsymbol{x}_{l+1}}{\partial \boldsymbol{x}_l}$ は

$$\frac{\partial \boldsymbol{x}_{l+1}}{\partial \boldsymbol{x}_l} = \frac{\partial \boldsymbol{f}_l(\boldsymbol{W}_l \boldsymbol{x}_l + \boldsymbol{b}_l)}{\partial \boldsymbol{x}_l} = \boldsymbol{D}_l \boldsymbol{W}_l \tag{2.10}$$

となる．ただし，

$$\boldsymbol{D}_l = \begin{bmatrix} [f_l'(\boldsymbol{W}_l \boldsymbol{x}_l + \boldsymbol{b}_l)]_1 & 0 & \cdots & & 0 \\ 0 & [f_l'(\boldsymbol{W}_l \boldsymbol{x}_l + \boldsymbol{b}_l)]_2 & 0 & \cdots & \vdots \\ \vdots & & \ddots & & \\ & & & & 0 \\ 0 & \cdots & & 0 & [f_l'(\boldsymbol{W}_l \boldsymbol{x}_l + \boldsymbol{b}_l)]_{d_{l+1}} \end{bmatrix} \tag{2.11}$$

であり，$f'(\cdot)$ は $f(\cdot)$ の導関数である．よって，式 (2.9) は

$$\nabla_{\boldsymbol{\theta}_l} C = \frac{\partial C}{\partial \boldsymbol{x}_L} \left( \prod_{i=l+1}^{L-1} \boldsymbol{D}_i \boldsymbol{W}_i \right) \frac{\partial \boldsymbol{x}_{l+1}}{\partial \boldsymbol{\theta}_l} \tag{2.12}$$

となる．この計算を Fig. 2.3 に示した．図のように，まず $\boldsymbol{x}_0$ から $\boldsymbol{y}$ まで順伝播で出力を計算したのち，それらの微分を逆方向に計算して積をとっていく．式 (2.12) のように，勾配計算は活性化関数の導関数と重み行列の複数の積からなるため，これらの値が DNN の学習において重要である．

深層学習において，式 (2.7) の最適化の目的関数 $J_\mathrm{e}$ は一般に非凸関数である．そのため，大域的な最適解を得ることは困難であり，SGD は局所最適解か鞍点などで停止すると考えられている．よって，深層学習では一般に最適性条件を満たす解を求めることはせずに，後述する early stopping や固定の反復数を実施することなどを停止条件としている．このようなヒューリスティックな学習方法であっても，深層学習は高性能を示すことが経験的に知られており，これに関してどの局所解も性能が高いとする研究 [29] や，過剰にパラメータがあるときに大域的な最適解に SGD が収束する [30] などの研究が進められている．

### 2.1.3 汎化性能と過学習

深層学習を含む機械学習の学習の目的は，確率分布 $p_D$ に従うデータ $\boldsymbol{x}, \boldsymbol{y}$，損失関数 $C$ に対して，以下の目的関数 $J_\mathrm{g}$

$$J_\mathrm{g} = \mathbb{E}_{p_D} \left[ C(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{\theta}) \right] \tag{2.13}$$

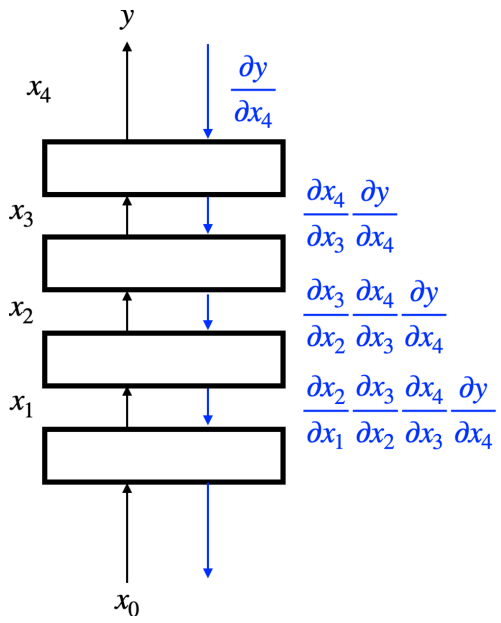



Fig. 2.3 Illustration of back propagation. Blue arrows correspond to back propagation, and black arrows correspond to forward propagation. In back propagation, a derivative of each layer is computed and propagated from output to input.

を最小化するモデルのパラメータ $\boldsymbol{\theta}$ を求めることである．式 (2.13) の目的関数 $J_{\mathrm{g}}$ は汎化誤差と呼ばれる．$\mathbb{E}_{p_D}$ はデータの母集団に対する期待値をとる演算であるが，実際には母集団の確率分布 $p_D$ は得られない．そこで，前述のとおり，用意した $N$ 個の標本（training data）$(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$ に対してサンプル平均をとった目的関数

$$J_{\mathrm{e}} = \frac{1}{N} \sum_{j=1}^{N} C(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}; \boldsymbol{\theta}) \tag{2.14}$$

の最小化を行う．式 (2.14) の目的関数 $J_{\mathrm{e}}$ は訓練誤差と呼ばれる．しかし式 (2.13) と式 (2.14) の違いから，有限個の training data に対する式 (2.14) の最小化，は式 (2.13) の損失関数を増大させる可能性がある．すると，training data を学習したモデルを使って，新しく $p_D$ からサンプルされたデータに対して推論を行うと誤差が大きくなる．このような現象は過学習（overfitting）と呼ばれ，特に自由度の高いモデルほど training data に過学習し汎化誤差が大きくなる．深層学習のモデルは一般に数百万個以上のパラメータを持つ自由度の高いモデルであるため，過学習しやすく，これを防ぐために次のような技術が提案されている．

### (a) Dropout

汎化性能を上げるためにニューラルネットワークによく用いられる技術が dropout である．これはニューラルネットワークの出力を確率的に選定する技術である [31]．入力を $\boldsymbol{x}$ とする

と，dropout は

$$\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{q} \odot \boldsymbol{x} \tag{2.15}$$

である．ただし，$\boldsymbol{q}$ の各要素 $q_i$ は

$$q_i \sim \text{Bernoulli}(p) \tag{2.16}$$

である [32]．また，$\odot$ は要素ごとの積で，Bernoulli($p$) は確率 $p$ で 1 を，確率 $1-p$ で 0 をとるベルヌーイ分布である（$0 \leq p \leq 1$）．$p$ は事前に設定する．確率 $p$ で次の層へニューラルネットの入力を伝達し，確率 $1-p$ で出力を 0 として入力を伝達させない．このようにして学習したモデルを用いて推論を行う際にはすべての入力に $p$ を乗じて

$$\boldsymbol{f}(\boldsymbol{x}) = p\boldsymbol{x} \tag{2.17}$$

とし期待値計算としてすべての入力を伝達させる．ただし，DNN のフレームワークによっては学習時に dropout の入力を $1/p\boldsymbol{x}$ とすることで，推論時には $\boldsymbol{f}(\boldsymbol{x}) = p/p\boldsymbol{x} = \boldsymbol{x}$ とする実装上の工夫がなされている．

### (b) 重み減衰

汎化誤差を小さくする方法の 1 つとして重み減衰（weight decay）もよく用いられる．これは $L_2$ 正則化とも呼ばれ，パラメータを求める最適化の目的関数を

$$J_{\text{w}} = \frac{1}{N}\sum_{j=1}^{N} C(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}; \boldsymbol{\theta}) + \frac{\nu}{2}||\boldsymbol{\theta}||_2^2 \tag{2.18}$$

とする．第 2 項を正則化項と呼び，ここで $\nu$ は正則化の重みを決める係数で，$||\cdot||_2$ は $L_2$ ノルムである．$J_{\text{w}}$ の $\boldsymbol{\theta}$ に関する勾配を計算すると，

$$\nabla_{\boldsymbol{\theta}} \left\{ \frac{1}{N}\sum_{j=1}^{N} C(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}; \boldsymbol{\theta}) + \frac{\nu}{2}||\boldsymbol{\theta}||_2^2 \right\} = \nabla_{\boldsymbol{\theta}} \frac{1}{N}\sum_{j=1}^{N} C(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}; \boldsymbol{\theta}) + \nu\boldsymbol{\theta} \tag{2.19}$$

となり，パラメータの更新ではパラメータの大きさに $\nu$ を乗じたものを減算するような更新となり，パラメータが小さな値に制約される．パラメータの探索空間が小さく制約されることにより，過学習が抑えられて汎化誤差が小さく抑えられる．正則化項としては他に $L_1$ ノルムを用いた $L_1$ 正則化があり，これはスパースな解が得られることから，モデルの軽量化などの目的で用いられる．

(c) ハイパーパラメータチューニングについて

深層学習をはじめとして機械学習のモデルやアルゴリズムには，多数のハイパーパラメータが存在する．ハイパーパラメータとは学習によって最適化される他のパラメータの分布に影響を与え，試行錯誤的に調整しなければならないパラメータを指す [33]．例えば深層学習においては，ニューラルネットワークのユニット数（隠れベクトルの長さ）や層の数，上記の dropout や正則化項の重み $\nu$ などを試行錯誤的に決定しなければならない．この試行錯誤を行うための指標が必要であるが，学習データの損失を小さくするように調整すると，前述のとおり過学習してしまい，実際の未知データに合わない汎化性能の低いモデルとなる．過学習を避け，汎化性能が向上するよう調整するためには，学習に使用するデータ以外のデータに対して評価する必要がある．そこで用いられるデータに検証用のデータが用意されていることがあり，それを validation data と呼び，それに対する損失を validation loss と呼ぶ．ただし，validation data に対して過学習することも考えられるため最終的なモデルの評価にはパラメータチューニングにも使用しない test data が使用される．

この validation loss を用いた汎化性能向上の方法として early stopping が提案されている．これは学習の途中で過学習を起こして validation loss が上昇する場合に validation loss が最も低かったモデルを使用する方法である．

### 2.1.4 学習アルゴリズム

深層学習の基本的な学習のアルゴリズムの擬似コードを Algorithm 2.1 に示し，その擬似コードについて説明する．まず，重み行列やバイアスなどのパラメータを初期化する（Algorithm 2.1 内の line 1）．この初期化では，一般に乱数で生成した値を使用し，その乱数の設定方法には [34] や [35] などで提案された方法が用いられる．ただし，RNN では状態の遷移に関わる行列の初期化として直交行列を用いる方法が提案されている [16, 36]．次に，各エポックにおいて，重複がないようにデータからミニバッチをサンプルする（line 4）．そしてミニバッチに対する損失関数のパラメータに関する勾配を計算して，パラメータの更新を行う（line 5）．最後に，事前に設定した全エポック数 $E$ が実行されるか，validation loss が上昇するなどの停止条件が満たされたら学習を終了する（line 7, 8）．ハイパーパラメータチューニングを行う際にはこれを複数のパラメータ設定でそれぞれ実行し，最終的な validation loss が最も良いモデルを選択する．

**Algorithm 2.1:** Training of deep neural networks

Initialize parameters $\boldsymbol{\theta}$
**for** epoch $\in 1, \ldots,$ E **do**
  **for** $\tau \in 1, \ldots, N/|D|$ **do**
    Sample a minibatch $D_\tau$
    $\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \nabla_{\boldsymbol{\theta}} \frac{1}{|D_\tau|} \sum_{(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}) \in D_\tau} C(\boldsymbol{\theta}, \boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)})$
  **end for**
  **if** $\boldsymbol{\theta}$ satisfies the stop condition **then**
    Return $\boldsymbol{\theta}$
  **end if**
**end for**

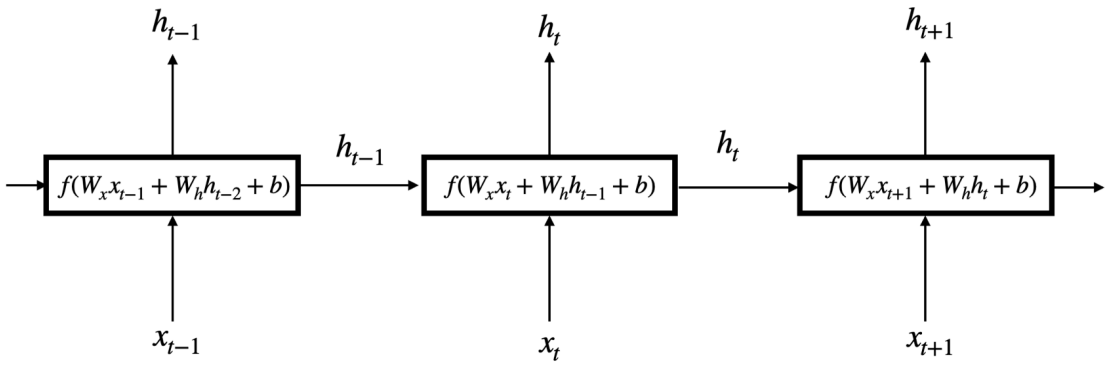



Fig. 2.4 Illustration of a one-layer RNN. State transition is represented by right arrows. At each time step, an input data point and previous state is applied into an RNN, and current state is computed by using them.

## 2.2 Recurrent neural network

RNN は出力を次の時刻のノードの入力とするようなニューラルネットワークであり，これにより過去の状態を記憶し，前後関係に依存のある時系列データのモデリングに適した構造である．単純な RNN の状態更新は，時刻 $t$ の状態を $\boldsymbol{h}_t \in \mathbb{R}^{n \times 1}$ とすると

$$\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{b}) \tag{2.20}$$

と表現される．ただし，$\boldsymbol{W}_*$ は入力と状態それぞれに対する重み行列で $\boldsymbol{f}$ は活性化関数である．1 時刻前の状態 $\boldsymbol{h}_{t-1}$ が入力されることで，時系列データの前後関係の依存関係を表現することができる．RNN の状態遷移を Fig. 2.4 に示す．この図のように，RNN は各時刻の状態遷移を左から右へ展開して表現される．RNN の学習においても，フィードフォワード型のニュー

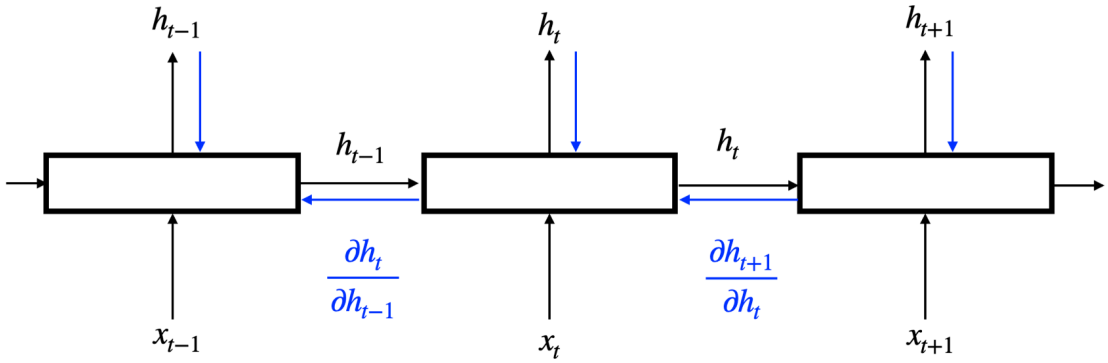



Fig. 2.5 Illustration of back propagation through time. Blue arrows correspond to back propagation, and black arrows correspond to forward propagation. In BPTT, a derivative of each time step is computed and propagated from output to input and from the current state to the past state.

ラルネットワークと同様に微分の連鎖律による SGD が用いられる．時刻 $t$（$t = 1, \ldots, T$）における損失関数を $c_t$ とし，全体の損失関数を $C = \sum_{t=1}^{T} c_t$ とすると，

$$\nabla_{\boldsymbol{\theta}_l} C = \sum_{t=1}^{T} \nabla_{\boldsymbol{\theta}_l} c_t \tag{2.21}$$

となる．簡単のため 1 層の RNN とし，時刻 $t$ の状態 $\boldsymbol{h}_{t+1} = \boldsymbol{f}(\boldsymbol{W}_x \boldsymbol{x}_{t+1} + \boldsymbol{W}_h \boldsymbol{h}_t + \boldsymbol{b})$ とし，いま $\boldsymbol{W}_h$ に関する勾配を考えると

$$\begin{aligned}
\nabla_{\boldsymbol{W}_h} c_t &= \frac{\partial c_t}{\partial \boldsymbol{h}_t} \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{W}_h} = \frac{\partial c_t}{\partial \boldsymbol{h}_t} \left( \frac{\partial \bar{\boldsymbol{h}}_t}{\partial \boldsymbol{W}_h} + \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} \frac{\partial \boldsymbol{h}_{t-1}}{\partial \boldsymbol{W}_h} \right) \\
&= \frac{\partial c_t}{\partial \boldsymbol{h}_t} \left( \frac{\partial \bar{\boldsymbol{h}}_t}{\partial \boldsymbol{W}_h} + \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}} \left( \frac{\partial \bar{\boldsymbol{h}}_{t-1}}{\partial \boldsymbol{W}_h} + \frac{\partial \boldsymbol{h}_{t-1}}{\partial \boldsymbol{h}_{t-2}} \frac{\partial \boldsymbol{h}_{t-2}}{\partial \boldsymbol{W}_h} \right) \right) \\
&= \frac{\partial c_t}{\partial \boldsymbol{h}_t} \left( \sum_{\tau=1}^{t} \prod_{i=\tau}^{t-1} \frac{\partial \boldsymbol{h}_{i+1}}{\partial \boldsymbol{h}_i} \frac{\partial \bar{\boldsymbol{h}}_\tau}{\partial \boldsymbol{W}_h} \right)
\end{aligned} \tag{2.22}$$

となる．ただし，$\partial \bar{\boldsymbol{h}}_t / \partial \boldsymbol{W}_h$ は $\boldsymbol{h}_{t-1}$ を定数とみなしたときの $\boldsymbol{W}_h$ による $\boldsymbol{h}_t$ の微分である．以上のように，勾配計算が時間に渡って積をとるため back propagation through time (BPTT) と呼ばれる．BPTT を Fig. 2.5 に示す．図のように勾配計算が時間方向に伝播される．RNN の時間方向への逆誤差伝搬は計算量が大きいため，数十から百程度の適当な時間ステップ数で逆誤差伝搬を打ち切る truncated BPTT が用いられる．

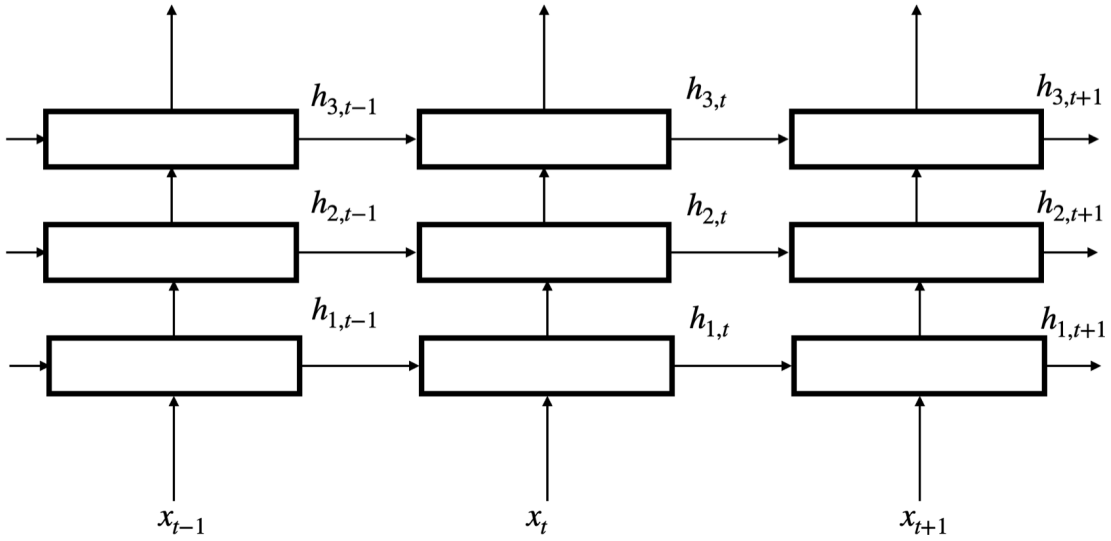



Fig. 2.6 Illustration of a multi-layer RNN. RNNs are stacked to obtain an accurate model.

RNN もフィードフォワード型と同様に精度を向上させるために，RNN を複数層重ねて

$$\begin{aligned}
\boldsymbol{h}_{1,t} &= \boldsymbol{f}_1(\boldsymbol{W}_x\boldsymbol{x}_t + \boldsymbol{W}_h\boldsymbol{h}_{1,t-1} + \boldsymbol{b}) \\
\boldsymbol{h}_{2,t} &= \boldsymbol{f}_2(\boldsymbol{W}_x\boldsymbol{h}_{1,t} + \boldsymbol{W}_h\boldsymbol{h}_{2,t-1} + \boldsymbol{b}) \\
&\vdots \\
\boldsymbol{h}_{L,t} &= \boldsymbol{f}_L(\boldsymbol{W}_x\boldsymbol{h}_{L-1,t} + \boldsymbol{W}_h\boldsymbol{h}_{L,t-1} + \boldsymbol{b})
\end{aligned} \tag{2.23}$$

としてよく用いられる（Fig. 2.6）． ただし，RNN の場合は数十層重ねることは少なく，数層程度であることが多い．その他に逆方向，すなわち未来から過去へ向かって状態を更新する RNN と組み合わせる bidirectional RNN や各時刻の状態を重みづけする attention などの構造が提案されているが，本論文の目的への関連が少ないため割愛する．

### 2.2.1 モデル構造

RNN の主なモデル構造として Elman 型，Jordan 型の RNN について簡単に説明し，本論文で主に扱う GRU については詳しく説明する．

#### (a) Elman 型，Jordan 型の RNN

RNN の中で最も単純で simple RNN や vanilla RNN と呼ばれるモデルが Elman 型の RNN と Jordan 型の RNN である．ただし，これらのモデルは後述する LSTM や GRU に性能が劣るため，本論文では使用しない．

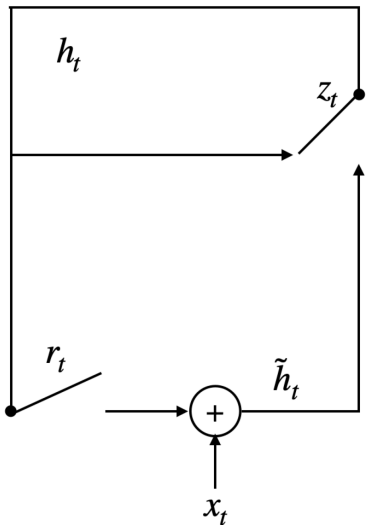



Fig. 2.7 Illustration of GRU. The update gate $z$ controls the effect of input $x$ on state $h$, and the reset gate $r$ controls the effect of state $h$ on a new candidate of state $\tilde{h}$.

### (b) Long short term memory（LSTM）

時系列データは，時間ステップの離れた長期的な依存関係と短期的な依存関係が存在し，そのような複雑な依存関係を表現するためにはモデルは適切に情報の保持，あるいは忘却をしなければならない．LSTM はこの問題を解決するため，メモリーセル $\boldsymbol{c}_t$ と呼ばれる状態を持ち，これに忘却ゲート $\boldsymbol{z}_t$，入力ゲート $\boldsymbol{i}_t$，出力ゲート $\boldsymbol{o}_t$ と呼ばれる構造を持って

$$\boldsymbol{z}_t = \mathrm{sigm}(\boldsymbol{W}_{zx}\boldsymbol{x}_t + \boldsymbol{W}_{zh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_z) \tag{2.24}$$
$$\boldsymbol{i}_t = \mathrm{sigm}(\boldsymbol{W}_{ix}\boldsymbol{x}_t + \boldsymbol{W}_{ih}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{2.25}$$
$$\boldsymbol{o}_t = \mathrm{sigm}(\boldsymbol{W}_{ox}\boldsymbol{x}_t + \boldsymbol{W}_{oh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{2.26}$$
$$\boldsymbol{c}_t = \boldsymbol{z}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tanh(\boldsymbol{W}_{cx}\boldsymbol{x}_t + \boldsymbol{W}_{ch}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{2.27}$$
$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{2.28}$$

である．式 (2.27) では，sigmoid によって [0,1] の値をとる $\boldsymbol{z}_t$ によって 1 時刻前の $\boldsymbol{c}_{t-1}$ をどれだけ保持するかが調整される．次に，$\boldsymbol{i}_t$ によってメモリーセルの更新量が調整される．最後に，メモリーセルからの出力が出力ゲート $\boldsymbol{o}_t$ によって調整されて状態 $\boldsymbol{h}_t$ が決まる．この構造によって勾配消失を解消して長期的な系列データの依存関係を表現することができる．

### (c) Gated recurrent unit（GRU）

前述の LSTM によって時系列データを精度よく表現できるが，式 (2.24)-(2.28) は複雑で計算量が大きい．そこで，より簡素な構造で長期記憶を実現するために GRU が提案された．過

去の系列データの情報の保持，忘却のために GRU は状態を更新ゲート，リセットゲートと呼ばれる 2 つのゲートを使って調節する（Fig. 2.7）．時刻 $t$ における更新ゲート $\boldsymbol{z}_t \in \mathbb{R}^{n\times 1}$ は

$$\boldsymbol{z}_t = \mathrm{sigm}(\boldsymbol{W}_{xz}\boldsymbol{x}_t + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1}), \tag{2.29}$$

である．ただし，$\boldsymbol{x}_t \in \mathbb{R}^{m\times 1}$ は入力を，$\boldsymbol{h}_t \in \mathbb{R}^{n\times 1}$ は状態を表す．$\boldsymbol{W}_{xz} \in \mathbb{R}^{n\times m}$ と $\boldsymbol{W}_{hz} \in \mathbb{R}^{n\times n}$ は重み行列であり，$\mathrm{sigm}(\cdot)$ は要素ごとの sigmoid である．

一方，リセットゲート $\boldsymbol{r}_t \in \mathbb{R}^{n\times 1}$ は

$$\boldsymbol{r}_t = \mathrm{sigm}(\boldsymbol{W}_{xr}\boldsymbol{x}_t + \boldsymbol{W}_{hr}\boldsymbol{h}_{t-1}), \tag{2.30}$$

である．ここで $\boldsymbol{W}_{xr} \in \mathbb{R}^{n\times m}$ と $\boldsymbol{W}_{hr} \in \mathbb{R}^{n\times n}$ は重み行列である．状態 $\boldsymbol{h}_t$ は

$$\boldsymbol{h}_t = \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (\boldsymbol{1} - \boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t, \tag{2.31}$$

によって更新される．ただし，$\boldsymbol{1}$ はすべての要素が 1 のベクトルであり，$\odot$ は要素ごとの積を表す．式 (2.31) をみると，更新ゲート $\boldsymbol{z}_t$ が 1 時刻前の状態 $\boldsymbol{h}_{t-1}$ と要素ごとに積をとられている．$\boldsymbol{z}_t$ の要素は 0 から 1 の値をとるため，$\boldsymbol{z}_t$ が 1 に近いと状態の更新がされず，0 に近いと新しい状態 $\tilde{\boldsymbol{h}}_t$ に更新される．この新しい状態の候補 $\tilde{\boldsymbol{h}}_t$ は

$$\tilde{\boldsymbol{h}}_t = \tanh(\boldsymbol{W}_{xh}\boldsymbol{x}_t + \boldsymbol{W}_{hh}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})), \tag{2.32}$$

で計算される．ただし，$\tanh(\cdot)$ は要素ごとの hyperbolic tangent で $\boldsymbol{W}_{xh} \in \mathbb{R}^{n\times m}$ と $\boldsymbol{W}_{hh} \in \mathbb{R}^{n\times n}$ は重み行列である．式 (2.32) においてもリセットゲート $\boldsymbol{r}_t$ と $\boldsymbol{h}_{t-1}$ の積がとられており，同様に次の状態をどの程度，過去の状態に依存させるかを調節している．なお本論文で GRU のバイアスはすべて 0 であるものとする．状態 $\boldsymbol{h}_t$ の初期値は一般にすべて 0，$\boldsymbol{h}_0 = \boldsymbol{0}$ とする．そのため $\boldsymbol{h}_t$ が $\boldsymbol{0}$ になると GRU が過去の情報をすべて忘却したことに対応する．

### 2.2.2 RNN の勾配爆発

勾配爆発は動的システムの安定性の変化によって生じると言われている．本項では非線形システムと安定性について簡単に説明し，勾配爆発との関係について述べる．そして既存研究である勾配クリッピングについて説明する．

#### (a) 動的システムと勾配爆発

式 (2.20) の単純な RNN や式 (2.28) の LSTM，式 (2.31) の GRU といった RNN は一般に次の非線形な動的システム

$$\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{h}_{t-1}, \boldsymbol{\theta}), \tag{2.33}$$

とみなすことができる．ただし，$\boldsymbol{h}_t$ は $t$ 時刻の状態ベクトルで $\boldsymbol{\theta}$ はパラメータ，$\boldsymbol{f}$ は非線形関数である．もし状態がある時刻 $t_*$ において $\boldsymbol{h}_{t_*} = \boldsymbol{f}(\boldsymbol{h}_{t_*}, \boldsymbol{\theta})$ を満たすと，外部から入力が与えられない限り状態は変化せず，そのような点 $\boldsymbol{h}_*$ は平衡点と呼ばれる．状態が平衡点 $\boldsymbol{h}_*$ 近傍で平衡点に収束するか離れるかは $\boldsymbol{f}$ と $\boldsymbol{\theta}$ に依存し，これは安定性と呼ばれる重要な特性である [37]．もし初期値 $\boldsymbol{h}_0$ が $||\boldsymbol{h}_0 - \boldsymbol{h}_*|| < \varepsilon$ を満たす状態 $\boldsymbol{h}_t$ に対して $\lim_{t\to\infty} ||\boldsymbol{h}_t - \boldsymbol{h}_*|| = 0$ となるような定数 $\varepsilon$ が存在するとき，平衡点 $\boldsymbol{h}_*$ は漸近安定と呼ぶ．一方，$\boldsymbol{h}_*$ が安定[*1]でなければ，この平衡点は不安定である．安定性や状態 $\boldsymbol{h}_t$ の平衡点近傍での振る舞いは $\boldsymbol{\theta}$ の滑らかな変化で大きく変化する．この現象は分岐と呼ばれ，分岐を起こすパラメータ点は分岐点と呼ばれる [37]．

[13, 38, 39] は分岐によって勾配爆発が生じて，RNN の学習が失敗することを示した．SGD によってパラメータが更新され分岐点に達すると，状態の振る舞いが急激に変化する．この不連続な振る舞いの変化によって損失関数がこの点において不連続となる．その結果として分岐点で勾配が非常に大きな値となり（勾配爆発），勾配に基づく SGD による学習は失敗する．

ここで勾配爆発のもう 1 つの解釈について説明する．式 (2.22) を再掲する：

$$\nabla_{\boldsymbol{W}_h} c_t = \frac{\partial c_t}{\partial \boldsymbol{h}_t} \left( \sum_{\tau=1}^{t} \prod_{i=\tau}^{t-1} \frac{\partial \boldsymbol{h}_{i+1}}{\partial \boldsymbol{h}_i} \frac{\partial \bar{\boldsymbol{h}}_\tau}{\partial \boldsymbol{W}_h} \right). \tag{2.22}$$

式 (2.22) のように，勾配計算にはヤコビ行列の積 $\prod_{i=\tau}^{t-1} \partial \boldsymbol{h}_{i+1}/\partial \boldsymbol{h}_i$ が含まれる．この積が指数関数的に勾配を小さくする現象を勾配消失と呼び，LSTM と GRU はこれを防ぐためにゲート構造を持つ．一方，指数関数的に勾配を大きくする現象が勾配爆発といえる．分岐とこの解釈との関係を簡単に述べる．まず，平衡点近傍で局所的に安定で $\partial \boldsymbol{h}_{i+1}/\partial \boldsymbol{h}_i$ の最大固有値が 1 より小さいとき，この平衡点近傍の状態は平衡点に収束し，$\prod_{i=\tau}^{t-1} \partial \boldsymbol{h}_{i+1}/\partial \boldsymbol{h}_i$ が増大しない．この平衡点において $\partial \boldsymbol{h}_{i+1}/\partial \boldsymbol{h}_i$ の最大固有値が 1 より大きくなる，つまり平衡点が不安定化する分岐が生じると，この平衡点近傍にいた状態のヤコビ行列の最大固有値が 1 より大きくなることから，勾配が指数関数的に増大する．よって安定なまま分岐しない，つまり分岐が生じなければその平衡点近傍で勾配爆発は生じないといえる．

なお，単純な RNN：$\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{b})$ であれば，この計算は

$$\prod_{i=\tau}^{t-1} \frac{\partial \boldsymbol{h}_{i+1}}{\partial \boldsymbol{h}_i} = \prod_{i=\tau}^{t-1} \frac{\partial \boldsymbol{f}(\boldsymbol{W}_x \boldsymbol{x}_{i+1} + \boldsymbol{W}_h \boldsymbol{h}_i + \boldsymbol{b})}{\partial \boldsymbol{h}_i} = \prod_{i=\tau}^{t-1} (\boldsymbol{D}_i \boldsymbol{W}_h). \tag{2.34}$$

[*1] 安定にはリアプノフ安定などの漸近安定以外の安定が含まれるが，これらについては本論文では割愛する

**Algorithm 2.2:** Gradient clipping

1: $\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{\theta}} C_{D_\tau}$
2: **if** $||\boldsymbol{g}|| \geq threshold$ **then**
3: $\quad \boldsymbol{g} \leftarrow \dfrac{threshold}{||\boldsymbol{g}||}\boldsymbol{g}$
4: **end if**

となる．ただし，

$$\boldsymbol{D}_i = \begin{bmatrix} [f'(\boldsymbol{W}_x\boldsymbol{x}_{i+1} + \boldsymbol{W}_h\boldsymbol{h}_i + \boldsymbol{b})]_1 & 0 & \cdots & 0 \\ 0 & [f'(\boldsymbol{W}_x\boldsymbol{x}_{i+1} + \boldsymbol{W}_h\boldsymbol{h}_i + \boldsymbol{b})]_2 & 0 \quad \cdots & \vdots \\ \vdots & & \ddots & \\ 0 & \cdots & 0 & [f'(\boldsymbol{W}_x\boldsymbol{x}_{i+1} + \boldsymbol{W}_h\boldsymbol{h}_i + \boldsymbol{b})]_n \end{bmatrix} \tag{2.35}$$

である．[40] では，$\boldsymbol{W}_h$ をユニタリ行列に制約し，最大特異値を 1 に制約することで $\prod_{i=\tau}^{t-1}(\boldsymbol{D}_i\boldsymbol{W}_h)$ が増大しないようにする．しかし，実データのモデリングにはこの制約が強すぎることが示されている [41].

(b) 勾配クリッピング（gradient clipping）

勾配爆発による学習の破綻を防ぐため，勾配クリッピングが提案された [13]．勾配クリッピングでは事前に設定した閾値を，勾配のノルム $\nabla_{\boldsymbol{\theta}} 1/|D_\tau| \sum_{(\boldsymbol{x}^{(j)},\boldsymbol{y}^{(j)})\in D_\tau} C(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}; \boldsymbol{\theta})$ が超えた場合に，ノルムが閾値以下となるようにクリップする．その擬似コードを Algorithm 2.2 に示す．この方法は理論的な保証はないが，試行錯誤的に閾値を決めることで学習が進むことが経験的に知られている．

ここで GRU に対して勾配クリッピングを行うときの計算量を考える．$\alpha$ を GRU 層以外のパラメータ数として，重み行列 $\boldsymbol{W}_{h*}$ のサイズ $n\times n$，$\boldsymbol{W}_{x*}$ のサイズ $n\times m$ から全パラメータ $\boldsymbol{\theta}$ の要素数は $3(n^2+mn)+\alpha$ であり，勾配クリッピングの計算量は $O(n^2+mn+\alpha)$ となる．

### 2.2.3 RNN に対する dropout

Dropout を RNN へ適用する場合，時間方向への情報の伝播が重要であるため工夫が必要である．[1] では時間方向への情報の伝播の阻害を防ぐため，時間方向への状態遷移には dropout を適用しない方法を提案している．この方法を Fig. 2.8 で示す．図の破線部分には dropout を適用しない．一方，[2] では，dropout で 0 にマスクされるニューラルネットのユニットを時間にわたって共通させる方法を提案している．Fig. 2.9 にこの方法を示す．図では dropout のマスクを共通する計算を色を共通することで示した．本論文では，第 3 章の GRU の安定化で前者の方法を，第 4 章の softmax の改良において後者の方法を用いて評価する．

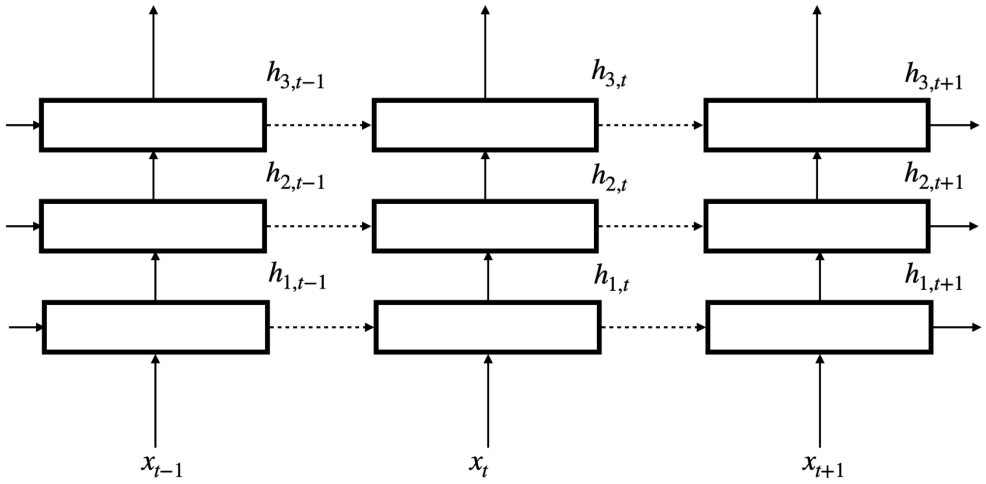



Fig. 2.8　Connections applied dropout in [1]. Solid lines correspond to connections applied dropout, and dashed lines correspond to connections not applied dropout.

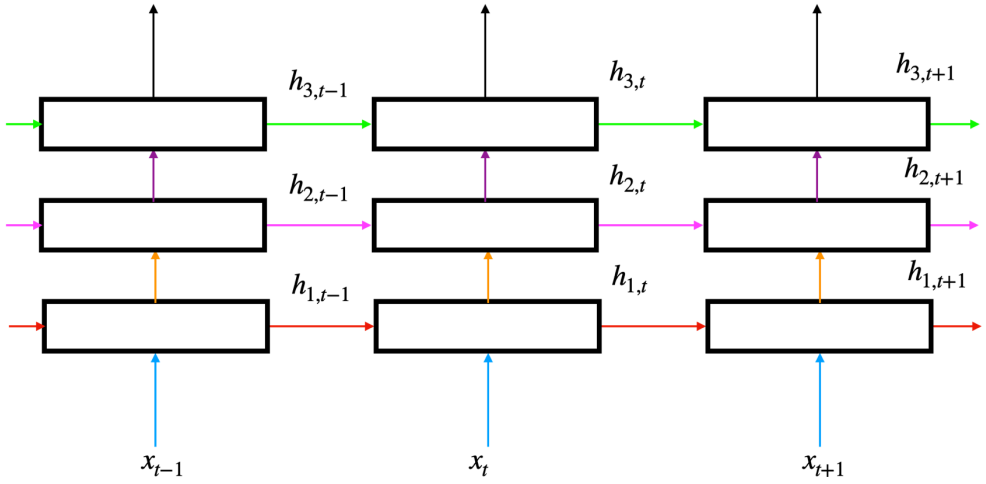



Fig. 2.9　Connections applied dropout in [2]. The same color connections share the same dropout masks.

### 2.2.4　各時刻のデータが離散分布に従う時系列データのモデリング

本項では RNN などの深層学習を使ったデータのモデリングについて説明する．まず，離散分布を表現するために使用される softmax を説明する．次に，RNN を使った時系列データのモデリング方法について説明する．最後に，具体的に言語データに RNN を適用した際のモデリングの詳細について説明する．

### (a) Softmax

Softmax を出力とする深層学習のモデルについて簡単のためクラス分類を例に説明する．入力 $\boldsymbol{x}$ とそれに対する $M$ 個のクラスを $\boldsymbol{y} \in \mathbb{R}^M$，モデルのパラメータを $\boldsymbol{\theta}$ とする．深層学習によるクラス分類では，最終層の入力を $\boldsymbol{h}(\boldsymbol{x}) \in \mathbb{R}^d$，最終層の重みを $\boldsymbol{W}^{M\times d}$ とすると，$\boldsymbol{x}$ が $i$ 番目のクラスである確率のモデルを，

$$p_{\boldsymbol{\theta}}(y_i|\boldsymbol{x}) = [\boldsymbol{f}_{\boldsymbol{s}}(\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}))]_i = \frac{\exp([\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x})]_i)}{\sum_m \exp([\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x})]_m)} \tag{2.36}$$

として softmax 関数 $\boldsymbol{f}_s$ を使って表現する．ただし，$[\boldsymbol{f}_{\boldsymbol{s}}(\cdot)]_i$ は $\boldsymbol{f}_{\boldsymbol{s}}(\cdot)$ の $i$ 番目の要素である．式 (2.36) では $\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x})$ の各要素が指数関数で変換された後，全要素の和で割られることで $\boldsymbol{f}_{\boldsymbol{s}}(\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x}))$ の各要素が [0,1]，かつ和が 1 となり確率を表現できる．また，指数関数が単調増加の関数であるため，予測ラベルのみを必要とするときは，softmax 関数に通す前の $\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x})$ の中の最も大きいものを選べばよい．

こうしたモデルの学習では，損失関数として負の対数尤度 $-\log(p_{\boldsymbol{\theta}}(y_i|\boldsymbol{x}))$ を用いてこれを最小化するようにモデルを学習する．前述したように，この対数尤度の最小化は SGD が用いられるため，勾配の性質が重要であり [13, 26, 42]，softmax を用いる利点の 1 つは負の対数尤度を損失関数としたときの勾配の計算が減算で計算しやすいことである [17, 33]．実際に $\boldsymbol{z} = \boldsymbol{W}\boldsymbol{h}(\boldsymbol{x})$ として softmax の出力の対数をとった log-softmax の勾配を計算すると

$$\frac{\partial\,[\log \boldsymbol{f}_s(\boldsymbol{z})]_i}{\partial z_j} = \begin{cases} 1 - [\boldsymbol{f}_s(\boldsymbol{z})]_j\,, & \text{if } j = i, \\ -[\boldsymbol{f}_s(\boldsymbol{z})]_j\,, & \text{if } j \neq i, \end{cases} \tag{2.37}$$

となり，出力の減算で表現できる．一般に対数 $\log(z)$ の微分は $1/z$ となり $z$ が小さいとゼロ割りを起こしうるが，式 (2.37) は除算を含まず数値的に安定である．

### (b) Softmax の導出

[17] では勾配が減算になることから，softmax は自然な出力関数と説明されている．もう 1 つの解釈として，softmax は次のような仮定から導出される [33]．最終層の状態 $\boldsymbol{h}$ について，クラス（または出力ラベル）$y_i$ が与えられた条件付き確率を

$$p(\boldsymbol{h}|y_i) = \exp\,\{A(\boldsymbol{\theta}_i) + B(\boldsymbol{h}, \boldsymbol{\phi}) + \boldsymbol{\theta}_i \boldsymbol{h}\} \tag{2.38}$$

とする．これは指数型分布族と呼ばれる確率分布であり，例えば正規分布やベルヌーイ分布などが指数型分布族に含まれる．$\boldsymbol{\theta}$ と $\boldsymbol{\phi}$ はパラメータで，$A$，$B$ は既知の関数とされる．ただし，$\boldsymbol{\phi}$ は $i$ によらないと仮定する．

一方，$\boldsymbol{h}$ が与えられたときのラベル $y_i$ の条件付き確率を考えると，ベイズ則より

$$\begin{aligned} P(y_i|\boldsymbol{h}) &= \frac{p(\boldsymbol{h}, y_i)}{p(\boldsymbol{h})} \\ &= \frac{p(\boldsymbol{h}, y_i)}{\sum_m p(\boldsymbol{h}, y_m)} \\ &= \frac{p(\boldsymbol{h}|y_i)P(y_i)}{\sum_m p(\boldsymbol{h}|y_m)p(y_m)} \end{aligned} \tag{2.39}$$

となる．ここで，式 (2.39) に式 (2.38) を代入して整理すると

$$P(y_i|\boldsymbol{h}) = \frac{\exp(\boldsymbol{w}_i^T \boldsymbol{h} + b_i)}{\sum_m \exp(\boldsymbol{w}_m^T \boldsymbol{h} + b_m)} \tag{2.40}$$

となる．ただし，

$$\boldsymbol{w}_i = \boldsymbol{\theta}_i \tag{2.41}$$

$$b_i = A(\boldsymbol{\theta}_i) + \ln P(y_i) \tag{2.42}$$

である．以上のように最終層の隠れ層の状態 $\boldsymbol{h}$ が $y_i$ が与えられたときに指数型分布族に従うと仮定すると，softmax は自然に導出できる．指数型分布族は解析などに優れた性質を持つがすべての確率分布が指数型とは限らず，深層学習の中で非線形関数で複雑に変換された後の最終層の状態 $\boldsymbol{h}$ が指数型分布族に従うとは限らない．また，$\phi$ が $y_i$ に依存しないことも仮定している．よって，softmax が必ずしもデータを表現するのに適した出力関数とは限らない．

### (c)　RNN による時系列データのモデル

RNN による時系列データのモデリングを考える．時系列データ $\boldsymbol{Y} = (Y_1, \ldots, Y_T)$ が与えられたとすると．この時系列データの同時確率 $P(\boldsymbol{Y})$ は

$$\begin{aligned} P(\boldsymbol{Y}) &= P(Y_T|Y_{T-1}, Y_{T-1}, \ldots, Y_1)P(Y_{T-1}, Y_{T-1}, \ldots, Y_1) \\ &= P(Y_T|Y_{T-1}, Y_{T-1}, \ldots, Y_1)P(Y_{T-1}|Y_{T-2}, \ldots, Y_1) \ldots P(Y_2|Y_1)P(Y_1) \\ &= \prod_{t=1}^{T} P(Y_t|Y_{<t}) \\ &= \prod_{t=1}^{T} P(Y_t|X_t) \end{aligned} \tag{2.43}$$

のように分解できる．ただし，$Y_t$ の値は $M$ 個の離散値 $i = 1, \ldots, M$ をとるとする．ここで，$X_t = Y_{<t} = (Y_1, \ldots, Y_{t-1})$ をコンテキストと呼ぶこととする．さらに対数をとると，

$$\log P(\boldsymbol{Y}) = \log \left( \prod_{t=1}^{T} P(Y_t|X_t) \right) = \sum_{t=1}^{T} \log P(Y_t|X_t) \tag{2.44}$$

となる．時刻 $t$ の RNN の入力を $\boldsymbol{x}_t$，状態を $\boldsymbol{h}_t$，出力を $\boldsymbol{y}_t$，パラメータを $\boldsymbol{\theta}_*$ として

$$\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t, \boldsymbol{\theta}_h) \tag{2.45}$$
$$\boldsymbol{y}_t = \boldsymbol{g}(\boldsymbol{h}_t, \boldsymbol{\theta}_y) \tag{2.46}$$

とする．ただし，$Y_t$ を離散値とするため，$\boldsymbol{y}_t$ の出力は最後，softmax 関数などを通して各要素が [0,1]，和が 1 であるとする．このとき $[y_t]_i = P_{\boldsymbol{\theta}}(Y_t = i|X_t)$ として系列データをモデリングする．入力 $\boldsymbol{x}_t = Y_{t-1}$ とすれば，$\boldsymbol{h}_t$ と $\boldsymbol{x}_t$ によってコンテクスト $X_t$ を RNN が記憶できると考えられる．学習時は負の対数尤度

$$-\sum_{t=1}^{T} \log P(Y_t|X_t) = -\sum_{t=1}^{T} \log [y_t]_{i_t} \tag{2.47}$$

を最小化して学習する．ただし，$i_t = Y_t$ である．

### (d) 言語モデル

自然言語処理で文や文書も系列データとみなすことができるが，これらの生成確率モデルを言語モデルと呼ぶ [43]．言語モデルには，各系列 $Y_t$ が単語を表すような単語レベルの言語モデルと，文字を表す文字レベルの言語モデルがある．それぞれ $M$ 個の語彙があるとすると，各単語（あるいは各文字）に 1 から $M$ の数値を割り当てれば前述の系列データのモデリングとして言語モデルを定式化できる．このとき $M$ 次元のベクトル $\boldsymbol{x} \in \mathbb{R}^M$ を用意し，$i$ 番目の単語に対して

$$x_j = \begin{cases} 1, & \text{if} \quad j = i \\ 0, & \text{otherwise} \end{cases} \tag{2.48}$$

として 0 と 1 からなるベクトル（one-hot vector と呼ぶ）を割り当てる方法がよく用いられる．このような入力を用いると，$i$ 番目の単語はモデルの第 1 層の線形演算 $\boldsymbol{W}\boldsymbol{x}$ によって

$$\boldsymbol{W}\boldsymbol{x} = \begin{bmatrix} \boldsymbol{w}_1 & \dots & \boldsymbol{w}_{i-1} & \boldsymbol{w}_i & \boldsymbol{w}_{i+1} & \dots & \boldsymbol{w}_M \end{bmatrix} \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \boldsymbol{w}_i \tag{2.49}$$

として重み行列 $\boldsymbol{W}$ の $i$ 番目の列ベクトルに対応する．このように学習によって単語に対応したベクトルを得ることを埋め込み（embedding）と呼び，この重み行列による層 $\boldsymbol{W}\boldsymbol{x}$ は embedding layer と呼ばれる．言語モデルにおいて第 1 層の重み行列と最終層の重み行列は同

じく単語を表す行列と考えられることから，これらの重み行列と共通させる tied-weight と呼ばれる方法が提案されている [44]．この場合，第 1 層と最終層の状態の長さを同じにする必要がある．

言語モデルの評価指標には，パープレキシティ（perplexity）がよく使用される．これは負の対数尤度の平均を指数関数に入れた

$$\text{perplexity} = \exp\left(-\frac{1}{N}\sum_{i}^{N}\log(P(Y_i))\right) \tag{2.50}$$

である．ただし，$P(Y_i)$ は $i$ 番目のデータで正解ラベルの確率を表す．本論文では perplexity が小さいほど精度が高い性能の良いモデルであるとする．

## 2.3　関連研究

ここでは，本論文の目的である “時系列のための深層学習モデル gated recurrent unit の学習の安定化” と “softmax の表現能力に関する解析と高い表現力を持つ出力関数の提案” について，その関連研究を概説する．

### 2.3.1　RNN の学習の安定化に関連する研究

勾配クリッピング以外の勾配爆発を防ぐ方法として，[45] は勾配爆発を防ぐために，RNN のダイナミクスを状態が安定となるように制約する学習法を提案した．また [46] はリアプノフ関数から安定となる学習率を求める学習法を提案している．しかし，これらの方法は Jordan 型や Elman 型の単純な RNN を対象とした学習方法であり，GRU のような複雑で長期記憶可能なモデルに対して直接適用することは難しい．さらに，これらの方法では平均二乗誤差を損失関数として仮定しているが，本論文で提案する方法は損失関数を限定しない．前節で述べた BPTT によって分岐が生じて勾配爆発が発生することは [38] によって示され，また，勾配爆発の起こらない BPTT に代わる real-time recurrent learning (RTRL) という学習法を提案した．しかし，RTRL は $n$ を状態数，$u$ を出力ユニット数とすると各反復ごとに $O((n+u)^4)$ という大きな計算量を必要とする [47]．前述の通り，勾配爆発と勾配消失を防ぐために RNN の重み行列をユニタリ行列に制約する方法 [40] が提案されているが，実問題にはこの制約が強すぎることが示されている [41]．[48] は RNN の再帰的な結合に対する重み行列を特異値分解後のユニタリ行列と対角行列の形でパラメータを割り当て学習させる方法を提案した．

単純な RNN のダイナミクスに関してはさまざまな解析が行われているが [49–53]，GRU のような最近のモデルに対する解析はまだまだ少ない．[54] は ReLU を使った RNN のダイナミクスを解析し，[55] は実験的に LSTM と GRU がカオス的な振る舞いを起こすことを示して安定な新しいモデルを提案している．[56] では LSTM や GRU の重みと状態の局所的な安

定性との関係について解析されている．

本論文では GRU の重み行列の最大特異値（スペクトルノルム）を制約する．そのようなスペクトルノルムを制約した学習法の研究が convolutional neural network などのフィードフォワード型のニューラルネットに対して行われており，このことにより画像認識の汎化性能やロバスト性の向上，generative adversarial networks の学習が安定することが報告されている [57–60]．

### 2.3.2 深層学習の出力関数に関する研究

これまで softmax 関数に代わる出力関数はいくつか提案されている [61–63]．[61] では spherical softmax と Taylor softmax が提案されている．しかし，これらの関数は出力の数が大きいときに softmax の性能を超えないことが実験的に示されている．また，spherical softmax は数値的な安定性から注意深く調整しなければならないハイパーパラメータを持ち [61]，Taylor softmax は softmax の近似であるため，softmax と同様に softmax bottleneck を解消できない．[62] は秘密計算を使った機械学習のために ReLU で構成された出力関数を提案している．しかし，この関数はゼロ割による数値計算の不安定性がある．そのほかに softmax の計算を効率化するような研究が複数あるが [64–67]，これらは表現能力を向上することを目的にしていない．

表現能力を向上させる関数として [68] では学習可能な単調増加関数を提案している．また，[69] では表現能力の高い mixture of softmax に対して，より計算量の小さい mixtape という層を提案している．しかし，本論文の目的はパラメータを増やすことなく，表現能力を比較することであるため，学習するパラメータを持つこれらの研究については本論文内では比較しない．

# 第 3 章

# GRU の勾配爆発を防ぐ学習方法

本章では，勾配爆発を抑制して gated recurrent unit（GRU）を学習する方法を提案する[*1]．勾配爆発は，小さなパラメータ変化によって recurrent neural network（RNN）の状態のダイナミクスが劇的に変化する分岐（bifurcation）によって生じる．そこで GRU のダイナミクスを解析し，分岐点の一つを明らかにする．この分岐点を回避するため，GRU の重み行列に制約をつけた学習法を提案し，さらにそれを効率的に行う手法を考案する．RNN の性能評価によく用いられる言語と音楽のモデリングの実験でこれを評価し，この手法によって勾配爆発が抑えられること，精度が同等以上であることを確認する．

## 3.1 はじめに

第 2 章で述べたように，勾配の積による勾配消失や勾配爆発という現象によって backpropagation through time（BPTT）を使った RNN の学習は困難である [13, 14]．勾配が小さくなり過去の情報が伝えられなくなる勾配消失を回避するために，単純なモデルとし GRU が提案された [16]．一方，勾配爆発は勾配が非常に大きな値となる現象であり，これが生じるとパラメータが非常に大きな値となり学習が失敗してしまう．そこで勾配のノルムがある閾値を超えた場合に，勾配の大きさを小さくする勾配クリッピング（gradient clipping）が提案された [13]．これは簡単でタスクによらず使用できるが，ヒューリスティックな方法なので閾値を試行錯誤的に調整しなければならない．

そこで本章では，勾配爆発が生じる原因である平衡点の安定性の変化について，GRU のモデルを解析し，安定的に学習できるアルゴリズムを提案する．

---

[*1] 本章は電子情報通信学会和文論文誌 D (Copyright©2019 IEICE) に含まれる “Gated recurrent unit の局所安定化による勾配爆発の抑制”[3] および NeurIPS2018 発表論文 “Preventing Gradient Explosions in Gated Recurrent Units”[70] に基づく．本章の一部データは電子情報通信学会和文論文誌 D において発表され，また電子情報通信学会の許可のもと論文 [3] から再利用されている．

## 3.2　局所安定化による勾配爆発の抑制

第 2 章で述べたように分岐が勾配爆発を生じさせる．この節では GRU のダイナミクスの解析を通して，分岐を回避し勾配爆発を防ぐ方法を提案する．

### 3.2.1　状態を安定に制約し勾配爆発を防ぐ学習法

まず，提案する学習法を特異値に関する制約付き最適化問題として定式化する．簡単のため一層の GRU の学習法を説明し，その後，複数層の場合に拡張する．

#### (a)　一層 GRU

2.4 節で示した一層の GRU は

$$
\begin{aligned}
\boldsymbol{z}_t &= \mathrm{sigm}(\boldsymbol{W}_{xz}\boldsymbol{x}_t + \boldsymbol{W}_{hz}\boldsymbol{h}_{t-1}),\\
\boldsymbol{r}_t &= \mathrm{sigm}(\boldsymbol{W}_{xr}\boldsymbol{x}_t + \boldsymbol{W}_{hr}\boldsymbol{h}_{t-1}),\\
\boldsymbol{h}_t &= \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1} + (\boldsymbol{1} - \boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t,\\
\tilde{\boldsymbol{h}}_t &= \tanh(\boldsymbol{W}_{xh}\boldsymbol{x}_t + \boldsymbol{W}_{hh}(\boldsymbol{r}_t \odot \boldsymbol{h}_{t-1})),
\end{aligned}
$$

と記述できる．ただし，$\boldsymbol{x}_t \in \mathbb{R}^{m\times 1}$ は入力を，$\boldsymbol{h}_t \in \mathbb{R}^{n\times 1}$ は状態を表す．$\boldsymbol{z}_t \in \mathbb{R}^{n\times 1}$ は更新ゲート，$\boldsymbol{r}_t \in \mathbb{R}^{n\times 1}$ はリセットゲート，$\tilde{\boldsymbol{h}}_t$ は新しい状態の候補である．$\boldsymbol{W}_*$ は重み行列であり，$\mathrm{sigm}(\cdot)$ は要素ごとの sigmoid である．

通常の GRU の学習は

$$
\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{j=1}^{N} C(\boldsymbol{x}^{(j)}, \boldsymbol{y}^{(j)}; \boldsymbol{\theta}),
$$

のような最適化問題として定式化されるが，勾配爆発を防ぐために一層の GRU の学習を次の制約付き最適化問題

$$
\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{j=1}^{N} C^{(j)},\ \text{s.t. } \sigma_1(\boldsymbol{W}_{hh}) < 2, \tag{3.1}
$$

で行う方法を提案する．ただし，$\sigma_i(\cdot)$ は行列の $i$ 番目に大きな特異値であり．$\sigma_1(\cdot)$ はスペクトルノルムと呼ばれる．式 (3.1) の最適化は次の定理で示すように GRU を局所的に安定に保ち，この平衡点の安定性の変化による勾配爆発を防ぐことができる：

**定理 3.1.** もし $\sigma_1(\boldsymbol{W}_{hh}) < 2$ ならば，一層の GRU の平衡点 $\boldsymbol{h}_* = \boldsymbol{0}$ は安定である．

この定理は式 (3.1) による学習法が，平衡点 $\boldsymbol{h}_* = \boldsymbol{0}$ の安定性を変化させないことを示している．そのため提案法は $\boldsymbol{h}_*$ の安定性の変化による勾配爆発を起こさない．この定理を証明するために，次の 3 つの補題を用いる：

**補題 3.1.** 一層の GRU は $\boldsymbol{h}_* = \boldsymbol{0}$ に平衡点を持つ．

**証明.** 入力 $\boldsymbol{x}_t = \boldsymbol{0}$ と状態 $\boldsymbol{h}_{t-1} = \boldsymbol{0}$ をそれぞれ式 (2.29) と (2.30) に代入すると，更新ゲートが $\boldsymbol{z}_t = \boldsymbol{1}/2$ となり，リセットゲートは $\boldsymbol{r}_t = \boldsymbol{1}/2$ となる．次に，$\boldsymbol{x}_t = \boldsymbol{0}$ と $\boldsymbol{h}_{t-1} = \boldsymbol{0}$，$\boldsymbol{r}_t = \boldsymbol{1}/2$ を式 (2.32) に代入すると，$\tilde{\boldsymbol{h}}_t = \boldsymbol{0}$ となる．最後に，$\boldsymbol{h}_{t-1} = \boldsymbol{0}$ と $\boldsymbol{z}_t = \boldsymbol{1}/2$，$\tilde{\boldsymbol{h}}_t = \boldsymbol{0}$ を式 (2.31) に代入すると，新たな状態が $\boldsymbol{h}_t = \boldsymbol{0}$ となり，したがって，$\boldsymbol{h}_{t-1} = \boldsymbol{h}_t = \boldsymbol{0}$ が成り立つため，GRU は平衡点 $\boldsymbol{h}_* = \boldsymbol{0}$ を持つ． □

**補題 3.2.** $\boldsymbol{I}$ を $n \times n$ の単位行列, $\lambda_i(\cdot)$ を $i$ 番目に絶対値の大きな固有値, $\boldsymbol{J} = 1/4\boldsymbol{W}_{hh} + 1/2\boldsymbol{I}$ とする．もしスペクトル半径[*2] $|\lambda_1(\boldsymbol{J})|$ が $|\lambda_1(\boldsymbol{J})| < 1$ であるならば，入力のない一層の GRU は，$\boldsymbol{h}_t = \boldsymbol{0}$ 近傍で次のように線形近似できる：

$$\boldsymbol{h}_t = \boldsymbol{J}\boldsymbol{h}_{t-1} \tag{3.2}$$

このとき平衡点 $\boldsymbol{h}_* = \boldsymbol{0}$ は安定である．

**証明.** 局所的な安定性は平衡点近傍で線形化したシステムの安定性の解析によって判断できる [37]．入力がないとき，テイラー展開 [71] によって $\boldsymbol{h}_* = \boldsymbol{0}$ で線形化した GRU は

$$\boldsymbol{h}_t = \boldsymbol{J}\boldsymbol{h}_{t-1} \tag{3.3}$$

となる．ただし，$\boldsymbol{J}$ は $\boldsymbol{h}_t$ の，$\boldsymbol{h}_{t-1} = \boldsymbol{0}$ と $\boldsymbol{x}_t = \boldsymbol{0}$ における $\boldsymbol{h}_{t-1}$ に関するヤコビ行列であり，

$$\boldsymbol{J} = \left.\frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}}\right|_{\boldsymbol{h}_{t-1}=\boldsymbol{0}, \boldsymbol{x}_t=\boldsymbol{0}} = \frac{1}{4}\boldsymbol{W}_{hh} + \frac{1}{2}\boldsymbol{I} \tag{3.4}$$

である．式 (3.3) から

$$\boldsymbol{h}_t = \boldsymbol{J}^t \boldsymbol{h}_0 \tag{3.5}$$

が成り立つ．$\boldsymbol{J}^t$ はその固有値の $t$ 乗に依存し，式 (3.5) は $\boldsymbol{J}$ の固有値が線形化した GRU の振る舞いを決めることを示す．ハートマン＝グロブマンの定理より，双曲的な平衡点近傍の振る舞いは線形化したシステムの振る舞いで近似可能である [37]．$|\lambda_1(\boldsymbol{J})| < 1$ であれば $\boldsymbol{h}_* = \boldsymbol{0}$ は双曲的な平衡点であるため，一層の GRU は式 (3.3) で近似可能である．したがって安定性はスペクトル半径 $|\lambda_1(\boldsymbol{J})|$ によって決まり，$|\lambda_1(\boldsymbol{J})| < 1$ であれば，

$$\lim_{t\to\infty} ||\boldsymbol{h}_t - \boldsymbol{h}_*|| = \lim_{t\to\infty} ||\boldsymbol{h}_t|| = \lim_{t\to\infty} ||\boldsymbol{J}^t \boldsymbol{h}_0|| = 0$$

[*2] スペクトルノルムが最大特異値であるのに対し，スペクトル半径とは固有値の絶対値の最大値である．

が初期状態 $\boldsymbol{h}_0$ が十分に $\boldsymbol{h}_* = \boldsymbol{0}$ に近いすべての $\boldsymbol{h}_t$ に成り立つ．よって，$\boldsymbol{h}_* = \boldsymbol{0}$ は安定である．□

補題 3.2 は

$$\left|\lambda_1(\frac{1}{4}\boldsymbol{W}_{hh} + \frac{1}{2}\boldsymbol{I})\right| < 1 \tag{3.6}$$

という条件を用いれば，平衡点の安定性を変化させないことができることを示している．この制約は固有値に対する制約であり，半正定値計画問題などで解くことも考えられるが，深層学習において制約の範囲での学習率の探索などは計算量が大きく困難であるため，既存の制約付き最適化を適用することは難しいと思われる．そこで，次の補題により式 (3.1) のような特異値制約とする：

**補題 3.3.** もし，$\sigma_1(\boldsymbol{W}_{hh}) < 2$ であるならば，

$$\left|\lambda_1\left(\frac{1}{4}\boldsymbol{W}_{hh} + \frac{1}{2}\boldsymbol{I}\right)\right| < 1 \tag{3.7}$$

が成り立つ．

**証明．** $\boldsymbol{M}$ を $(n \times n)$ の行列，$c$ をスカラとすると，$\boldsymbol{M} + c\boldsymbol{I}$ の固有値は $\lambda_i(\boldsymbol{M}) + c$ となるため，

$$\left|\lambda_1\left(\frac{1}{4}\boldsymbol{W}_{hh} + \frac{1}{2}\boldsymbol{I}\right)\right| = \left|\frac{1}{4}\lambda_1\left(\boldsymbol{W}_{hh}\right) + \frac{1}{2}\right|$$

が成り立つ．また，三角不等式から

$$\left|\frac{1}{4}\lambda_1\left(\boldsymbol{W}_{hh}\right) + \frac{1}{2}\right| \leq \frac{1}{4}|\lambda_1(\boldsymbol{W}_{hh})| + \frac{1}{2}$$

が成り立つ．固有値と特異値について Weyl の不等式

$$\sum_{i=1}^{k}|\lambda_i(\boldsymbol{M})| \leq \sum_{i=1}^{k}\sigma_i(\boldsymbol{M})$$

がすべての $k = 1, 2, \ldots, n$ に成り立つ．したがって，

$$|\lambda_1(\boldsymbol{W}_{hh})| \leq \sigma_1(\boldsymbol{W}_{hh})$$

であり，

$$\frac{1}{4}|\lambda_1(\boldsymbol{W}_{hh})| + \frac{1}{2} \leq \frac{1}{4}\sigma_1(\boldsymbol{W}_{hh}) + \frac{1}{2}$$

が成り立つ．よって，$\sigma_1(\boldsymbol{W}_{hh}) < 2$ であれば，

$$1 > \frac{1}{4}\sigma_1(\boldsymbol{W}_{hh}) + \frac{1}{2} \geq \frac{1}{4}|\lambda_1(\boldsymbol{W}_{hh})| + \frac{1}{2} \ \geq \left|\lambda_1\left(\frac{1}{4}\boldsymbol{W}_{hh} + \frac{1}{2}\boldsymbol{I}\right)\right|$$

が成り立つ．よって $\sigma_1(\boldsymbol{W}_{hh}) < 2$ のとき，

$$\left|\lambda_1\left(\frac{1}{4}\boldsymbol{W}_{hh} + \frac{1}{2}\boldsymbol{I}\right)\right| < 1$$

が成り立つ． □

補題 3.1〜3.3 を用いると，定理 3.1 は次のように証明できる：

**証明.** 補題 3.1 より，平衡点 $\boldsymbol{h}_* = \boldsymbol{0}$ が一層の GRU に存在する．この平衡点は補題 3.2 より，

$$\left|\lambda_1\left(\frac{1}{4}\boldsymbol{W}_{hh} + \frac{1}{2}\boldsymbol{I}\right)\right| < 1$$

のとき安定であり，補題 3.3 より $\sigma_1(\boldsymbol{W}_{hh}) < 2$ であれば，

$$\left|\lambda_1\left(\frac{1}{4}\boldsymbol{W}_{hh} + \frac{1}{2}\boldsymbol{I}\right)\right| < 1$$

が成り立つ．よって $\sigma_1(\boldsymbol{W}_{hh}) < 2$ であれば，一層の GRU の平衡点 $\boldsymbol{h}_* = \boldsymbol{0}$ は安定な平衡点である． □

補題 3.1 は 1 層の GRU が平衡点を持つことを示し，補題 3.2 はその平衡点を安定に保つ条件を示す．補題 3.3 は固有値に関する制約の代わりに特異値を使用できることを示し，これらの補題によって定理 3.1 が証明された．この定理は提案法が平衡点の不安定化による勾配爆発を防ぐことを示している．

提案法で注目する平衡点は $\boldsymbol{h}_* = \boldsymbol{0}$ である．この平衡点は通常，状態の初期値として設定される値であり，第 2 章で説明したように，$\boldsymbol{0}$ にリセットされることで GRU は完全に過去の情報を忘却する．もし $\boldsymbol{h}_* = \boldsymbol{0}$ が安定であれば，$\boldsymbol{0}$ 近傍の状態が漸近的に $\boldsymbol{0}$ に収束する．これは，入力なしに十分に時間がたてば，GRU が過去の情報を完全に忘却できることを示している．一方，$|\lambda_1(\boldsymbol{J})| > 1$ のとき，$\boldsymbol{0}$ の平衡点は不安定となる．これは $\boldsymbol{h}_t$ が自動的に $\boldsymbol{0}$ となることがなく，人為的にリセットしなければ GRU が過去の情報を完全に忘却できないことを示している．また [55] は状態が 0 に収束する安定な RNN のモデルが LSTM や GRU などに匹敵する性能を達成することを示しており， GRU を $\boldsymbol{h}_* = \boldsymbol{0}$ が安定となるように学習することは有効であると考えられる．

(b)　多層 GRU

提案法を多層 GRU に拡張する．$L$ 層の GRU を

$$\begin{aligned}
\boldsymbol{h}_{1,t} &= \boldsymbol{f}_1(\boldsymbol{h}_{1,t-1}, \boldsymbol{x}_t) \\
\boldsymbol{h}_{2,t} &= \boldsymbol{f}_2(\boldsymbol{h}_{2,t-1}, \boldsymbol{h}_{1,t}) \\
&\vdots \\
\boldsymbol{h}_{L,t} &= \boldsymbol{f}_L(\boldsymbol{h}_{L,t-1}, \boldsymbol{h}_{L-1,t})
\end{aligned} \tag{3.8}$$

とする．ただし，$\boldsymbol{h}_{l,t} \in \mathbb{R}^{n_l \times 1}$ は長さ $n_l$ の $l$ 層の状態であり，$\boldsymbol{f}_l$ は $l$ 層の GRU である．1 層の GRU と同様に $\boldsymbol{h}_t = [\boldsymbol{h}_{1,t}^{\mathrm{T}}, \ldots, \boldsymbol{h}_{L,t}^{\mathrm{T}}]^{\mathrm{T}} = \boldsymbol{0}$ は平衡点であり，次の補題が成り立つ．

**補題 3.4.** もし $l = 1, \ldots, L$ について，

$$\left| \lambda_1 \left( \frac{1}{4} \boldsymbol{W}_{l,hh} + \frac{1}{2} \boldsymbol{I} \right) \right| < 1$$

であれば，多層に接続した GRU の平衡点 $\boldsymbol{h}_* = \boldsymbol{0}$ は安定である．

**証明.** 一層の GRU と同様に $\boldsymbol{h}_t = [\boldsymbol{h}_{1,t}^{\mathrm{T}}, \ldots, \boldsymbol{h}_{L,t}^{\mathrm{T}}]^{\mathrm{T}} = \boldsymbol{0}$ は平衡点であり，$\boldsymbol{h}_{t-1} = \boldsymbol{0}$ と $\boldsymbol{x}_t = \boldsymbol{0}$ におけるそのヤコビ行列 $\dfrac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}}$ は，

$$\boldsymbol{J} = \begin{bmatrix}
\dfrac{\partial \boldsymbol{h}_{1,t}}{\partial \boldsymbol{h}_{1,t-1}} & \boldsymbol{O} & \cdots & & \boldsymbol{O} \\
\dfrac{\partial \boldsymbol{h}_{2,t}}{\partial \boldsymbol{h}_{1,t-1}} & \dfrac{\partial \boldsymbol{h}_{2,t}}{\partial \boldsymbol{h}_{2,t-1}} & \boldsymbol{O} & \cdots & \vdots \\
\vdots & \ddots & & \ddots & \\
\dfrac{\partial \boldsymbol{h}_{L-1,t}}{\partial \boldsymbol{h}_{1,t-1}} & \dfrac{\partial \boldsymbol{h}_{L-1,t}}{\partial \boldsymbol{h}_{2,t-1}} & \cdots & \dfrac{\partial \boldsymbol{h}_{L-1,t}}{\partial \boldsymbol{h}_{L-1,t-1}} & \boldsymbol{O} \\
\dfrac{\partial \boldsymbol{h}_{L,t}}{\partial \boldsymbol{h}_{1,t-1}} & \dfrac{\partial \boldsymbol{h}_{L,t}}{\partial \boldsymbol{h}_{2,t-1}} & \cdots & \dfrac{\partial \boldsymbol{h}_{L,t}}{\partial \boldsymbol{h}_{L-1,t-1}} & \dfrac{\partial \boldsymbol{h}_{L,t}}{\partial \boldsymbol{h}_{L,t-1}}
\end{bmatrix} \tag{3.9}$$

となる．この行列はブロック下三角行列である．そのブロック対角行列，非対角行列はそれぞれ

$$\begin{aligned}
&\frac{\partial \boldsymbol{h}_{l,t}}{\partial \boldsymbol{h}_{l,t-1}} = \boldsymbol{A}_l, \\
&\frac{\partial \boldsymbol{h}_{l,t}}{\partial \boldsymbol{h}_{l-d,t-1}} = \sum_{p=0}^{d} \left[ \prod_{k=0}^{p-1} (\boldsymbol{B}_{l-k}) \boldsymbol{A}_{l-p} \prod_{q=0}^{-p+d-1} (\boldsymbol{B}_{l-p-q}) \right]
\end{aligned}$$

で与えられる．ただし，

$$\boldsymbol{A}_l = \frac{1}{4}\boldsymbol{W}_{l,hh} + \frac{1}{2}\boldsymbol{I}$$
$$\boldsymbol{B}_l = \frac{1}{2}\boldsymbol{W}_{l,xh}$$

であり，$\boldsymbol{W}_{l,hh}$ と $\boldsymbol{W}_{l,xh}$ は $l$ 層の GRU の重み行列である．ブロック下三角行列の固有値はその対角行列の固有値に一致し，$\boldsymbol{J}$ の固有値は $l = 1, \ldots, L$ の $\partial \boldsymbol{h}_{l,t}/\partial \boldsymbol{h}_{l,t-1}$ の固有値から求められる．したがって，スペクトル半径は

$$|\lambda_1(\boldsymbol{J})| = \max_l \left|\lambda_1\left(\frac{1}{4}\boldsymbol{W}_{l,hh} + \frac{1}{2}\boldsymbol{I}\right)\right| \tag{3.10}$$

となる．結果として，すべての $l = 1, \ldots, L$ に対して，

$$\left|\lambda_1\left(\frac{1}{4}\boldsymbol{W}_{l,hh} + \frac{1}{2}\boldsymbol{I}\right)\right| < 1$$

であれば，

$$|\lambda_1(\boldsymbol{J})| = \max_l \left|\lambda_1\left(\frac{1}{4}\boldsymbol{W}_{l,hh} + \frac{1}{2}\boldsymbol{I}\right)\right| < 1$$

が成り立つ． □

一層の GRU と同様に補題 3.3 より $\sigma_1(\boldsymbol{W}_{l,hh}) < 2$ のとき

$$\left|\lambda_1\left(\frac{1}{4}\boldsymbol{W}_{l,hh} + \frac{1}{2}\boldsymbol{I}\right)\right| < 1$$

が成り立つので，提案法を

$$\min_{\boldsymbol{\theta}} \frac{1}{N}\sum_{j=1}^{N} C^{(j)}, \ \text{s.t.} \ \sigma_1(\boldsymbol{W}_{l,hh}) < 2, \ \sigma_1(\boldsymbol{W}_{l,xh}) \leq 2 \ \text{ for } l = 1, \ldots, L \tag{3.11}$$

とする．ここで $\sigma_1(\boldsymbol{W}_{l,xh}) \leq 2$ という制約を加えたのは，入力によって状態が平衡点 $\boldsymbol{h}_* = \boldsymbol{0}$ の吸引領域を出ないように制約するためである．式 (3.11) の学習法によって多層の GRU を安定に保った学習ができる．

### 3.2.2 提案法による学習のアルゴリズム

式 (3.1) の最小化問題を解くためには $\{\boldsymbol{W}_{hh} | \boldsymbol{W}_{hh} \in \mathbb{R}^{n\times n}, \sigma_1(\boldsymbol{W}_{hh}) < 2\}$ を満たす領域（実行可能集合）で解を探索しなければならない．そこで，stochastic gradient descent（SGD）を修正し，

$$\boldsymbol{\theta}^{(\tau)}_{-W_{hh}} = \boldsymbol{\theta}^{(\tau-1)}_{-W_{hh}} - \eta \nabla_{\boldsymbol{\theta}} C_{D_\tau}(\boldsymbol{\theta})$$
$$\boldsymbol{W}^{(\tau)}_{hh} = \mathcal{P}_\delta(\boldsymbol{W}^{(\tau-1)}_{hh} - \eta \nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})) \tag{3.12}$$

としてパラメータを更新する方法を提案する．ここで $C_{D_\tau}(\boldsymbol{\theta})$ は $1/|D_\tau| \sum_{(\boldsymbol{x}^{(j)},\boldsymbol{y}^{(j)})\in D_\tau} C^{(j)}$ であり，$\boldsymbol{\theta}^{(\tau)}_{-W_{hh}}$ は $\boldsymbol{W}^{(\tau)}_{hh}$ を除くすべてのパラメータである．式 (3.12) において $\mathcal{P}_\delta(\cdot)$ は次の手順で計算する：

最大特異値を制約した学習アルゴリズム

Step 1.　$\hat{\boldsymbol{W}}^{(\tau)}_{hh} := \boldsymbol{W}^{(\tau-1)}_{hh} - \eta \nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})$ を次式のように特異値分解する．

$$\hat{\boldsymbol{W}}^{(\tau)}_{hh} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}} \tag{3.13}$$

Step 2.　閾値 $(2-\delta)$ 以上の特異値を閾値に置換する．

$$\bar{\boldsymbol{\Sigma}} = \mathrm{diag}(\min(\sigma_1, 2-\delta), \dots \min(\sigma_n, 2-\delta)) \tag{3.14}$$

Step 3.　$\boldsymbol{W}^{(\tau)}_{hh}$ を $\boldsymbol{U}$ と $\boldsymbol{V}$，$\bar{\boldsymbol{\Sigma}}$ により再構築する．

$$\boldsymbol{W}^{(\tau)}_{hh} \leftarrow \boldsymbol{U}\bar{\boldsymbol{\Sigma}}\boldsymbol{V}^{\mathsf{T}} \tag{3.15}$$

この手順によって $\boldsymbol{W}_{hh}$ は $(2-\delta)$ 以下のスペクトルノルムを持つことが保証される．$\delta$ を $0 < \delta < 2$ となるように設定すれば，提案法の $\sigma_1(\boldsymbol{W}_{hh})$ が 2 より小さいという制約が満たされる．$\mathcal{P}_\delta(\cdot)$ は SGD によってパラメータが実行可能集合の外に出た場合に，実行可能集合に戻すアルゴリズムであり，次の補題に示すように実行可能集合への最適な射影である [72].

**補題 3.5.** $\mathcal{P}_\delta(\cdot)$ によって得られる $\boldsymbol{W}^{(\tau)}_{hh}$ は，最適化問題

$$\min_{\boldsymbol{W}^{(\tau)}_{hh}} ||\hat{\boldsymbol{W}}^{(\tau)}_{hh} - \boldsymbol{W}^{(\tau)}_{hh}||^2_F, \quad \text{s.t. } \sigma_1(\boldsymbol{W}^{(\tau)}_{hh}) \leq 2-\delta$$

の解である．ここで $||\cdot||_F$ はフロベニウスノルムである．

**証明.** 2 つの正方行列 $\boldsymbol{M} \in \mathbb{R}^{q\times q}$ と $\boldsymbol{K} \in \mathbb{R}^{q\times q}$ に対して

$$\sum_{i=1}^{q} \{\sigma_i(\boldsymbol{M}) - \sigma_i(\boldsymbol{K})\}^2 \leq ||\boldsymbol{M} - \boldsymbol{K}||^2_F \tag{3.16}$$

が成り立つ [73]．式 (3.16) から，

$$||\hat{\boldsymbol{W}}^{(\tau)}_{hh} - \boldsymbol{W}^{(\tau)}_{hh}||^2_F \geq \sum_{i=1}^{n} \{\sigma_i(\hat{\boldsymbol{W}}^{(\tau)}_{hh}) - \sigma_i(\boldsymbol{W}^{(\tau)}_{hh})\}^2 \geq \sum_{i=1}^{s} \{\sigma_i(\hat{\boldsymbol{W}}^{(\tau)}_{hh}) - (2-\delta)\}^2 \tag{3.17}$$

が得られる．ただし $s$ は $(2-\delta)$ より大きな特異値の数である．

一方，提案法によってそのフロベニウスノルムは

$$\begin{aligned}||\hat{\boldsymbol{W}}_{hh}^{(\tau)} - \boldsymbol{W}_{hh}^{(\tau)}||_{\mathrm{F}}^2 &= ||\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^{\mathsf{T}} - \boldsymbol{U}\bar{\boldsymbol{\Sigma}}\boldsymbol{V}^{\mathsf{T}}||_{\mathrm{F}}^2 = ||\boldsymbol{U}(\boldsymbol{\Sigma} - \bar{\boldsymbol{\Sigma}})\boldsymbol{V}^{\mathsf{T}}||_{\mathrm{F}}^2 \\ &= \mathrm{tr}(\boldsymbol{V}(\boldsymbol{\Sigma} - \bar{\boldsymbol{\Sigma}})\boldsymbol{U}^{\mathsf{T}}\boldsymbol{U}(\boldsymbol{\Sigma} - \bar{\boldsymbol{\Sigma}})\boldsymbol{V}^{\mathsf{T}}) = \mathrm{tr}(\boldsymbol{V}(\boldsymbol{\Sigma} - \bar{\boldsymbol{\Sigma}})^2\boldsymbol{V}^{\mathsf{T}}) \\ &= \mathrm{tr}(\boldsymbol{V}^{\mathsf{T}}\boldsymbol{V}(\boldsymbol{\Sigma} - \bar{\boldsymbol{\Sigma}})^2) = \mathrm{tr}((\boldsymbol{\Sigma} - \bar{\boldsymbol{\Sigma}})^2) \\ &= \sum_{i=1}^{s}\{\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - (2-\delta)\}^2\end{aligned}$$

である．ただし，$\mathrm{tr}(\cdot)$ は行列のトレースである．これは

$$\min_{\boldsymbol{W}_{hh}^{(\tau)}} ||\hat{\boldsymbol{W}}_{hh}^{(\tau)} - \boldsymbol{W}_{hh}^{(\tau)}||_F^2, \quad \text{s.t. } \sigma_1(\boldsymbol{W}_{hh}^{(\tau)}) \leq 2 - \delta$$

の下界を達成している．したがって，このときの $\boldsymbol{W}_{hh}^{(\tau)}$ はその最適化問題の解である． □

補題 3.5 は，提案法がフロベニウスノルムの最小の変化でパラメータ $\boldsymbol{W}_{hh}$ を実行可能集合に戻すことを示し，このことから $\mathcal{P}_\delta(\cdot)$ が損失関数の最小化に与える影響は小さい．

提案法の最適性はフロベニウスノルムに限らず，次で示すようにトレースノルムの意味でも最適である．まず，行列 $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m\times n}$ の特異値には

$$\sum_i^k |\sigma_i(\boldsymbol{A}) - \sigma_i(\boldsymbol{B})| \leq \sum_i^k \sigma_i(\boldsymbol{A} - \boldsymbol{B}) \quad \text{for} \quad k = 1, 2, \ldots, \min(m, n) \tag{3.18}$$

という関係がある．いま，$\hat{\boldsymbol{W}}_{hh}^{(\tau)}, \boldsymbol{W}_{hh}^{(\tau)}$ について式 (3.18) の左辺を考える．全特異値について $\sigma_1(\boldsymbol{W}_{hh}^{(\tau)}) \leq 2 - \delta$ より

$$|\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - \sigma_i(\boldsymbol{W}_{hh}^{(\tau)})| \geq \max(0, \sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - (2-\delta)) \tag{3.19}$$

が成り立つ．よって，

$$\sum_i^k |\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - \sigma_i(\boldsymbol{W}_{hh}^{(\tau)})| \geq \sum_i^k \max(0, \sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - (2-\delta)) \tag{3.20}$$

である．よって，

$$\begin{aligned}\sum_i^k \max(0, \sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - (2-\delta)) &\leq \sum_i^k |\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - \sigma_i(\boldsymbol{W}_{hh}^{(\tau)})| \\ &\leq \sum_i^k \sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)} - \boldsymbol{W}_{hh}^{(\tau)}) \quad \text{for} \quad k = 1, \ldots, n\end{aligned} \tag{3.21}$$

となる．

ここで提案法を使うと

$$\sum_{i}^{k} |\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - \sigma_i(\boldsymbol{W}_{hh}^{(\tau)})| = \sum_{i}^{k} \max(0, \sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - 2 - \delta) \tag{3.22}$$

となる．また，提案法において

$$\begin{aligned}\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)} - \boldsymbol{W}_{hh}^{(\tau)}) =& \sigma_i(\boldsymbol{U}(\boldsymbol{\Sigma} - \bar{\boldsymbol{\Sigma}})\boldsymbol{V}^{\mathsf{T}}) \\ =& \sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - \sigma_i(\boldsymbol{W}_{hh}^{(\tau)})\end{aligned} \tag{3.23}$$

が成り立ち，$\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) \geq \sigma_i(\boldsymbol{W}_{hh}^{(\tau)})$ より

$$\sum_{i}^{k} \max(0, \sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - (2 - \delta)) = \sum_{i}^{k} \sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)} - \boldsymbol{W}_{hh}^{(\tau)}) \quad \text{for} \quad k = 1, \ldots, n \tag{3.24}$$

となる．よって，提案法は行列間の差の特異値の和（トレースノルム）を最小化している．

なお，提案法は学習率の設定法によらないため，深層学習でよく用いられる学習率の調整方法 Adam [74] などともに使用することも可能である．Fig. 3.1 に提案法の概要図を示す．図で示すように，制約を満たす領域にある $\boldsymbol{W}_{hh}^{\tau-1}$ を勾配によって更新した $\hat{\boldsymbol{W}}_{hh}^{\tau}$ が制約を満たさない場合，$||\boldsymbol{W}_{hh}^{\tau} - \hat{\boldsymbol{W}}_{hh}^{\tau}||_F$ が最も小さく制約を満たす点に $\hat{\boldsymbol{W}}_{hh}^{\tau}$ を更新する．これを繰り返して学習させるのが提案法である．

## 3.3　提案法の計算量の低減化

$n$ を状態 $\boldsymbol{h}_t$ のサイズとすると，特異値分解には $O(n^3)$ の計算量が必要である．$n$ が数百程度に設定され，数万回以上のパラメータ更新を行う GRU の学習においてこの計算量は実用の障害となる．そこで，本節では提案法の計算量を低減化する方法を与える．まず，$\mathcal{P}_{\delta}(\cdot)$ の計算方法について再考する．式 (3.13)-(3.15) は

$$\boldsymbol{W}_{hh}^{(\tau)} = \hat{\boldsymbol{W}}_{hh}^{(\tau)} - \sum_{i=1}^{s} \left[\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) - (2 - \delta)\right] \boldsymbol{u}_i \boldsymbol{v}_i^{\mathsf{T}} \tag{3.25}$$

とみなすことができる．ただし，$s$ は $(2 - \delta)$ より大きな特異値の数であり，$\boldsymbol{u}_i$ と $\boldsymbol{v}_i$ は $i$ 番目の左右の特異ベクトルである．式 (3.25) は提案法が計算しなければならない特異値，特異ベクトルが $\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) > 2 - \delta$ を満たす $s$ 個のみでよいことを示している．これを利用して計算コストを減らすため上から $s$ 個の特異値，特異ベクトルを計算できる高速な特異値分解 [75] を用いる．以後，この一部の特異値，特異ベクトルを求める方法を truncated SVD と呼ぶ．この方法は $s$ 個の特異値を $O(n^2 \log(s))$ 時間で計算可能である．ただし SGD を使ってパラメータを更新している中で，何個の特異値が条件 $(2 - \delta)$ より大きいかを事前に得ることは難しく

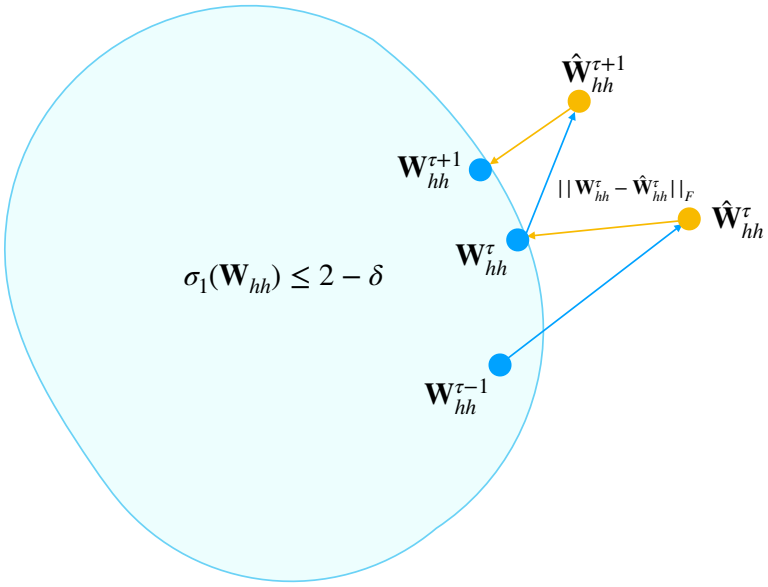



Fig. 3.1 Illustration of proposed method. A blue region represents a feasible set of the proposed constrained optimization problem. First, $\boldsymbol{W}_{hh}^{\tau-1}$ is updated to $\hat{\boldsymbol{W}}_{hh}^{\tau}$ by using gradient as shown by a blue arrow. Next, $\hat{\boldsymbol{W}}_{hh}^{\tau}$ is projected into the feasible set by using SVD; i.e., $\hat{\boldsymbol{W}}_{hh}^{\tau}$ is updated to $\boldsymbol{W}_{hh}^{\tau}$ as shown by a yellow arrow. These two steps are iterated until the stop condition is satisfied.

truncated SVD で指定する $s$ がわからない．そのため $\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) > 2-\delta$ である特異値の数を見積もらなければならない．そこで SGD で更新する前のパラメータの特異値とパラメータの更新量を使い，次の補題による上界を使って $s$ を効率的に見積もる．

**補題 3.6.** $\hat{\boldsymbol{W}}_{hh}^{(\tau)}$ の特異値は

$$\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) \leq \sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)}) + |\eta|||\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})||_F \tag{3.26}$$

によって上から抑えられる．

**証明.** 2 つの正方行列 $\boldsymbol{M} \in \mathbb{R}^{q\times q}$ と $\boldsymbol{K} \in \mathbb{R}^{q\times q}$ に対して，

$$\sigma_{i+j-1}(\boldsymbol{M}+\boldsymbol{K}) \leq \sigma_i(\boldsymbol{M}) + \sigma_j(\boldsymbol{K}) \text{ と } \sigma_1(\boldsymbol{K}) \leq ||\boldsymbol{K}||_F$$

が成り立つ [73]． これより

$$\sigma_i(\boldsymbol{M}+\boldsymbol{K}) \leq \sigma_i(\boldsymbol{M}) + \sigma_1(\boldsymbol{K}) \leq \sigma_i(\boldsymbol{M}) + ||\boldsymbol{K}||_F$$

が成り立つ．したがって，

$$\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) = \sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)} - \eta\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})) \leq \sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)}) + \sigma_1(-\eta\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta}))$$

であり，これは

$$\begin{aligned}\sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)}) + \sigma_1(-\eta\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})) &= \sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)}) + |\eta|\sigma_1(\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})) \\ &\leq \sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)}) + |\eta|||\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})||_F\end{aligned}$$

となる．　□

式 (3.26) の上界を使うことにより，更新前のパラメータの特異値と勾配がわかれば，$(2-\delta)$ より大きな特異値の数 $s$ の値を推定できる．なお，この上界の計算量は $\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})$ のサイズが $(n \times n)$ であり，$\tau$ ステップで $\sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)})$ が既に得られているため，$O(n^2)$ である．前回の更新で特異値の計算をしなかったものについては，この上界をそのまま適用できないが，例えば $(\tau - K)$ ステップから $(\tau - 1)$ ステップまで過去の特異値を計算しなかった場合は，

$$\sigma_i(\boldsymbol{W}_{hh}^{(\tau-K-1)}) + \sum_{k=0}^{K} |\eta|||\nabla_{\boldsymbol{W}_{hh}} C_{D_{\tau-k}}(\boldsymbol{\theta})||_F \tag{3.27}$$

として $\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)})$ の上界を計算できる．なお，もともとの制約が $\sigma_1(\boldsymbol{W}_{hh}^{(\tau)}) < 2$ であることから，$s$ を $\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) > 2 - \delta$ の代わりに $\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) \geq 2$ を満たす特異値の数とすると，さらに速度を向上できる．以上より，提案法は毎ステップ $O(n^2)$ で $s$ の大きさを見積もることができる．そして $s$ が 1 以上のときだけ，$O(n^2\log(s))$ の計算量で特異値分解を行う．

## 3.4　提案法の擬似コード

提案法の全体の学習の擬似コードを Algorithm 3.1 に示す．まず通常の学習と同様にパラメータを初期化する．その後，データをサンプリングしてミニバッチを作り，勾配に従ってパラメータを更新する（line 3-4）．次に，特異値分解を $\boldsymbol{W}^\tau$ に適用し，制約を満たさないパラメータを置換する（line 5-7）．これを停止条件を満たすまで繰り返す．

次に，低計算量のアルゴリズムを Algorithm 3.2 に示す．ここで，$i$ 番目の特異値の上界を $\bar{\sigma}_i$ とする．まず重み行列の初期値 $\boldsymbol{W}^{(0)}$ に対して特異値分解を行い，この結果より $(2-\delta)$ より大きな特異値のおよその数 $s$ と特異値の上界の値を初期化する（line 1-3）．次に，通常の学習と同様にミニバッチに対してパラメータの更新を行う（line 5, 6）．上界を計算するための $c = |\eta|||\nabla_{\boldsymbol{W}} 1/|D|_\tau \sum_{(x,y)\in D_\tau} C(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})||_{\mathrm{F}}$ を計算し（line 7, $O(n^2)$ の計算量），各特異値の上界を更新，同時に上界が 2 を超える個数 $s$ を計算する (line 8-12)．最後に，$s$ が 1 より大きい場合，上から $s$ 個の特異値分解を適用し，それに基づき重み行列 $\boldsymbol{W}^{(\tau)}$ から特異ベクトルを減算して制約を満たすように修正する（line 14-17, $O(n^2\log(s))$ の計算量）．なお，$(2-\delta)$ より大きな特異値の個数 $s$ は正確ではないため，式 (3.25) と異なり 0 と $\sigma_i(\boldsymbol{W}^{(\tau)}) - (2-\delta)$ の大きさを比較して $(2-\delta)$ より大きな特異値に対応する特異ベクトルのみ減算している．これを validation loss が下がらなくなるなどの停止条件を満たすまで繰り返す．

**Algorithm 3.1:** Training of GRU with local stability constraints

Initialize parameters $\boldsymbol{\theta}$
**for** $\tau \in 1, \ldots, T$ **do**
  Sample a minibatch $D_\tau = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}^B$
  $\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \nabla_{\boldsymbol{\theta}} \frac{1}{|D|_\tau} \sum_{(x,y)\in D_\tau} C(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$
  Apply SVD to $\boldsymbol{W}_{hh}^{(\tau)}$ as $\boldsymbol{W}_{hh}^{(\tau)} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\mathsf{T}$
  $\bar{\Sigma} = \mathrm{diag}(\min(\sigma_1, 2-\delta), \ldots, \min(\sigma_n, 2-\delta))$
  $\boldsymbol{W}_{hh}^{\tau} = \boldsymbol{U}\bar{\boldsymbol{\Sigma}}\boldsymbol{V}^\mathsf{T}$
  **if** $\boldsymbol{\theta}$ satisfies the stop condition **then**
    Return $\boldsymbol{\theta}$
  **end if**
**end for**

## 3.5 実験

### 3.5.1 実験条件

提案法を評価するため，言語モデリングの実験と音楽のモデリングの実験を行った．GRU を学習させ，学習の成功率と validation data に対する損失の平均と分散を評価した．学習の成功は，初期化したモデルに対する検証損失を各エポック終了後の検証損失が一度も超えることなく全エポック終了することとした．各実験条件は以下のとおりである．

#### (a) Language modeling

本実験では Penn Treebank (PTB) [76] と WikiText-2 dataset (WT2) [77] という 2 つのデータセットを用いて言語モデリングの実験を行った．これらのデータセットは RNN の性能評価に広く用いられている．各データの単語数や語彙数を Table 3.1 にまとめた．PTB は training と validation，test set にわかれ，それぞれ約 930k，74k，80k 単語である．語彙数は 10k とし，語彙にない単語については特別な単語＜ UNK ＞を割り当てた．一方，WT2 は training が約 2100k 単語，validation が 220k 単語，そして test が 250k 単語からなり，語彙数は 33,278 単語である．

実験条件は既存研究の [1] に基づいて設定した．モデルの一層目を $650 \times 10,000$ のバイアスなしの線形層（これは embedding layer と呼ばれる）とした．提案法で入力の値が大きすぎると平衡点近傍から状態が離れてしまうことが考えられるので一層の出力に 0.01 倍した．次

**Algorithm 3.2:** The proposed projection with low-complexity.

Apply SVD to $\boldsymbol{W}_{hh}^{(0)}$, and obtain $\sigma_i(\boldsymbol{W}_{hh}^{(0)})$.
Initialize parameters $s$ as $s = |\{\sigma_i | \sigma_i \geq 2 - \delta\}|$
Initialize upper bound $\bar{\sigma_i}$ as $\bar{\sigma_i} = \sigma_i(\boldsymbol{W}_{hh}^{(0)})$ for $i = 1, \ldots, n$
**for** $\tau \in 1, \ldots, T$ **do**
  Sample a minibatch $D_\tau = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}^B$
  $\boldsymbol{\theta}^{(\tau)} = \boldsymbol{\theta}^{(\tau-1)} - \eta \nabla_{\boldsymbol{\theta}} \dfrac{1}{|D|_\tau} \sum_{(x,y)\in D_\tau} C(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$
  $c = |\eta| || \nabla_{\boldsymbol{W}_{hh}} \frac{1}{|D|_\tau} \sum_{(x,y)\in D_\tau} C(\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y}) ||_{\mathrm{F}}$
  **for** $i \in 1, \ldots, n$ **do**
    $\bar{\sigma_i} = \bar{\sigma_i} + c$
    **if** $\bar{\sigma_i} \geq 2$ **then**
      $s = i$
    **end if**
  **end for**
  **if** $s > 0$ **then**
    Apply truncated SVD to $\boldsymbol{W}_{hh}^{(\tau)}$ to obtain $\sigma_i$, $\boldsymbol{u}_i$, $\boldsymbol{v}_i$ for $i = 1, \ldots, s$
    $\boldsymbol{W}_{hh}^{(\tau)} = \boldsymbol{W}_{hh}^{(\tau)} - \sum_{i=1}^{s} \max\{\left[\sigma_i(\boldsymbol{W}_{hh}^{(\tau)}) - (2 - \delta)\right], 0\} \boldsymbol{u}_i \boldsymbol{v}_i^{\mathsf{T}}$
  **end if**
  **if** $\boldsymbol{\theta}$ satisfies the stop condition **then**
    Return $\boldsymbol{\theta}$
  **end if**
**end for**

Table 3.1　Dataset statistics. The 2nd-4th columns are number of tokens of each dataset.

| Dataset | Training dataset | Valid. dataset | Test dataset | Vocabulary size |
|---|---|---|---|---|
| PTB | 929,589 | 73,760 | 82,430 | 10,000 |
| WT2 | 2,088,628 | 217,646 | 245,569 | 33,278 |

の層は 650 ユニットの GRU 層とし，出力に softmax を用いた．50 % の dropout を GRU の再帰的な結合を除く各層に適用した [1]．学習には予備実験において Adam と RMSprop の性能が SGD に劣ったため，SGD を用いた．

まず，重み行列の初期化として $\boldsymbol{W}_{hh}$ を除くすべての行列を正規分布 $\mathcal{N}(0, 1/650)$ から生成した．$\boldsymbol{W}_{hh}$ は乱数生成した行列を特異値分解し，得られた特異ベクトルを用いることで直交

行列となるように初期化した [16, 36]．ミニバッチの大きさを 20，SGD の学習率は 1 とし，最初の 10 エポックの後は毎ステップ 1.1 で除算して 75 エポックまで学習させた．なお，学習時には再帰的な結合の途中で逆誤差伝搬を打ち切る truncated BPTT を使用し，35 時間ステップで打ち切った．提案法において $\delta$ を [0.2, 0.5, 0.8, 1.1, 1.4] としてそれぞれ学習させた．一方，勾配クリッピングでは [13] の勾配のノルムの平均に注目するというヒューリスティクスにしたがって平均を調べたところ PTB が約 10，WT2 が約 7 であったので，それをもとに PTB では閾値を [5, 10, 15, 20]，WT2 では [3.5, 7.0, 10.5, 14] とした．各エポックの後に validation データに対する損失（以後，validation loss と呼ぶ）を評価し，75 エポックの間で最も小さな validation loss となったモデルを用いて test データで評価した．

(b) Polyphonic music modeling

この実験では，音楽の MIDI データに対して過去のノート番号[*3]から次に出てくるノート番号を予測する実験を行った．なお，言語モデリングと異なりノート番号は各時刻で複数出力される．データセットには 1200 のフォークミュージックで構成される Nottingham データセット [78] を用いた．なお，データセットは PTB と同様に training と validation，test set に分かれる．実験条件は [79] をもとに設定した．各時刻のノート番号を 93 次元のバイナリベクトルで表現した．モデルの一層目をバイアスなしの $200 \times 93$ の線形層とし，言語モデルと同様に出力に 0.01 倍した．2 層と 3 層を 200 ユニットの GRU 層とし，出力はロジスティック関数とした．50 % の dropout を GRU の再帰的な結合をのぞく各層に適用した．

学習は言語モデルと同様に SGD を使用し，初期化は $\boldsymbol{W}_{hh}$ を除くすべての行列を正規分布 $\mathcal{N}(0, 10^{-4}/200)$ から生成し，$\boldsymbol{W}_{hh}$ は同様の手順で直交行列となるように初期化した．ミニバッチの大きさを 20，BPTT の打ち切りを 35 とし，SGD の学習率は 0.1 として 10 エポック連続して validation loss の低下がみられないとき 1.25 で除算した．この手順で学習率が $10^{-4}$ 以下となるまで学習させた．提案法は $\delta$ を [0.2, 0.5, 0.8, 1.1, 1.4] とし，勾配クリッピングは勾配のノルムの平均が約 30 であったため [15, 30, 45, 60] とした．各エポックの後に validation data に対する損失を評価し，最も小さな validation loss となったモデルを用いて test data で評価した．

### 3.5.2 成功率と精度

Table 3.2 と 3.3 に言語モデルと音楽モデル，それぞれにおいて提案法と勾配クリッピングにおいてそれぞれのパラメータに対する学習成功率を示す．まず，提案法と勾配クリッピングの学習の成功率を比較すると，Table 3.2 の言語モデリングの結果において勾配クリッピング

[*3] 鍵盤の鍵の番号であり，鍵盤の中央の C を 60 として 0 から 127 まで番号を割り振ったもの．

Table 3.2　Comparison of success rates on language modeling. Top: PTB, bottom: WT2.

| Proposed method | | | | | | Gradient clipping | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\delta$ | 0.2 | 0.5 | 0.8 | 1.1 | 1.4 | Threshold | 5 | 10 | 15 | 20 |
| Success rate | 100 % | 100 % | 100 % | 100 % | 100 % | Success rate | 100 % | 40 % | 0 % | 0 % |
| $\delta$ | 0.2 | 0.5 | 0.8 | 1.1 | 1.4 | Threshold | 3.5 | 7 | 10.5 | 14 |
| Success rate | 100 % | 100 % | 100 % | 100 % | 100 % | Success rate | 100 % | 0 % | 0 % | 0 % |

Table 3.3　Comparison of success rates on music data modeling.

| Proposed method | | | | | | Gradient clipping | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\delta$ | 0.2 | 0.5 | 0.8 | 1.1 | 1.4 | Threshold | 15 | 30 | 45 | 60 |
| Success rate | 100 % | 100 % | 100 % | 100 % | 100 % | Success rate | 100 % | 100 % | 100 % | 100 % |

は勾配のノルムの平均 10（PTB）や 7（WT2）に閾値を設定しても学習が失敗した．そのため勾配クリッピングは学習の成功に試行錯誤が必要である．一方，提案法はどのような $\delta$ を用いても学習が成功しており，試行錯誤の軽減が期待できる．次に，Table 3.3 の音楽のモデリングの結果を比較すると，勾配クリッピング，提案法ともに学習が成功している．

次に，精度の比較として，言語モデリングと音楽のモデリングにおいて validation data と test data に対する損失の平均と標準偏差をそれぞれ Table 3.4 と 3.5 に示す．Table 3.4 の勾配クリッピングの言語モデリングの結果を見ると，PTB では閾値を 10 とした結果が最も小さい．しかし，閾値 10 は学習が半分以上失敗するパラメータであり，既存方法の学習が不安定であることを示す．Table 3.5 の音楽のモデリングでは閾値を 60 に設定すると非常に標準偏差が大きく，前述したように学習の失敗はないが，高精度なモデルを得るためには試行錯誤が必要である．提案法は，Table 3.4 と 3.5 より，どちらの実験においても $\delta = 0.2$ の損失が最も低く勾配クリッピングより高い精度が得られた．このように，提案法は制約により解空間が小さくなっているにも関わらず，GRU の性能が向上したのは状態が安定であるためと考えられる．例えば [80] において安定化した RNN が複数のタスクで性能が向上することが示されている．また [14] では安定な平衡点近傍に状態があるとき，ノイズに対してロバストになることが指摘されており，GRU の LSTM に対する優位点であるノイズにロバストという利点 [81] が提案法により強調されたと考えられる．

### 3.5.3　$\delta$ の大きさと状態の収束

Table 3.5 より，$\delta$ を 1.1 あるいは 1.4 に設定すると GRU の性能が落ちる．これは $\delta$ が大きいほど状態の平衡点への収束速度が上がり，長期的な依存関係をとらえられなくなるためであ

Table 3.4 Comparison of perplexities on language modeling. Top: PTB, bottom: WT2. Valid. and perp. denote validation and perplexity, respectively.

| Proposed method | | | | | | Gradient clipping | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\delta$ | 0.2 | 0.5 | 0.8 | 1.1 | 1.4 | Threshold | 5 | 10 | 15 | 20 |
| Valid. perp. | 102.0±0.3 | 102.8±0.3 | 103.7±0.2 | 105.2±0.2 | 107.0±0.4 | Valid. perp. | 109.3±0.4 | 103.1±0.4 | N/A | N/A |
| Test perp. | 97.6±0.4 | 98.4±0.3 | 99.0±0.4 | 100.3±0.2 | 102.1±0.2 | Test perp. | 106.9±0.4 | 100.4±0.5 | N/A | N/A |
| $\delta$ | 0.2 | 0.5 | 0.8 | 1.1 | 1.4 | Threshold | 3.5 | 7 | 10.5 | 14 |
| Valid. perp. | 130.4±0.7 | 130.9±0.9 | 132.0±0.4 | 134.0±0.6 | 135.0±0.6 | Valid. perp. | 140±1 | N/A | N/A | N/A |
| Test perp. | 121.6±0.5 | 122.0±0.5 | 123.0±0.8 | 124.1±0.6 | 124.9±0.2 | Test perp. | 134±1 | N/A | N/A | N/A |

Table 3.5 Comparison of negative log-likelihood on music data modeling. Valid. and NLL denote validation and negative log-likelihood, respectively.

| Proposed method | | | | | | Gradient clipping | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| $\delta$ | 0.2 | 0.5 | 0.8 | 1.1 | 1.4 | Threshold | 15 | 30 | 45 | 60 |
| Valid. NLL | 3.46±0.05 | 3.47±0.07 | 3.59±0.1 | 4.58±0.2 | 4.64±0.2 | Valid. NLL | 3.57±0.01 | 3.61±0.2 | 3.88±0.2 | 5.26±3 |
| Test NLL | 3.53±0.04 | 3.53±0.04 | 3.64±0.2 | 4.56±0.2 | 4.62±0.2 | Test NLL | 3.64±0.04 | 3.64±0.2 | 3.89±0.2 | 5.36±3 |

ると考えられる．提案法において $\boldsymbol{W}_{hh}$ のスペクトルノルムは $(2-\delta)$ 以下である．これはスペクトル半径 $|\lambda_1(\boldsymbol{J})|$ の上界であり，$|\lambda_1(\boldsymbol{J})|$ は線形化した GRU (式 (3.2)) の収束性を決める．そのため $\boldsymbol{h}_t = \boldsymbol{0}$ 近傍の状態は $\delta$ が 2 に近いほど速く平衡点に収束すると考えられる．

実際に状態の収束性を確認するため，簡単なシミュレーションを行った．提案法のそれぞれ $\delta = 0.2, 0.8, 1.4$ で学習したモデルの GRU 層に時刻 1 で正規分布 $\mathcal{N}(\boldsymbol{0}, 0.01\boldsymbol{I})$ に従うノイズを入力として加え，その後，入力を与えずに状態を時間変化させて各時刻の状態のノルム $||\boldsymbol{h}_t||_2$ を計算した．また特定の入力に対する結果とならないように同様の実験を 50 回行った．

各時刻の状態ベクトルの $l_2$ ノルムを 50 回の試行に対して平均した結果を Fig. 3.2 に示す．図より，提案法の GRU は状態のノルムが 0 に指数関数的に収束して $\boldsymbol{h} = \boldsymbol{0}$ が安定であるとともに，$\delta$ が大きくなるに従いその収束が速くなっている．よって，提案法は $\delta$ の値によって状態の収束性を調整できる．収束が速いほど過去のノイズの影響が小さくなりロバストであると考えられるが，一方で GRU が過去の情報を長期間にわたって記憶できなくなる．こうした特性を考慮してノイズの多いデータには大きな $\delta$ を，長期間の依存性があるデータには小さな $\delta$ を使用する．なお，同様の実験を勾配クリッピングで学習したモデルで行ったところ，勾配のノルムはほとんど変化せず約 4.7 となり，提案法と比べ非常に大きな値となった．

以上のように，提案法の調整パラメータはモデルへの影響が解釈しやすく，また，$0 < \delta < 2$ という範囲に制限されている．一方で，勾配クリッピングの閾値はモデルにどのような影響を与えるかの解釈が困難であり，その範囲も制限されず調整が難しい．

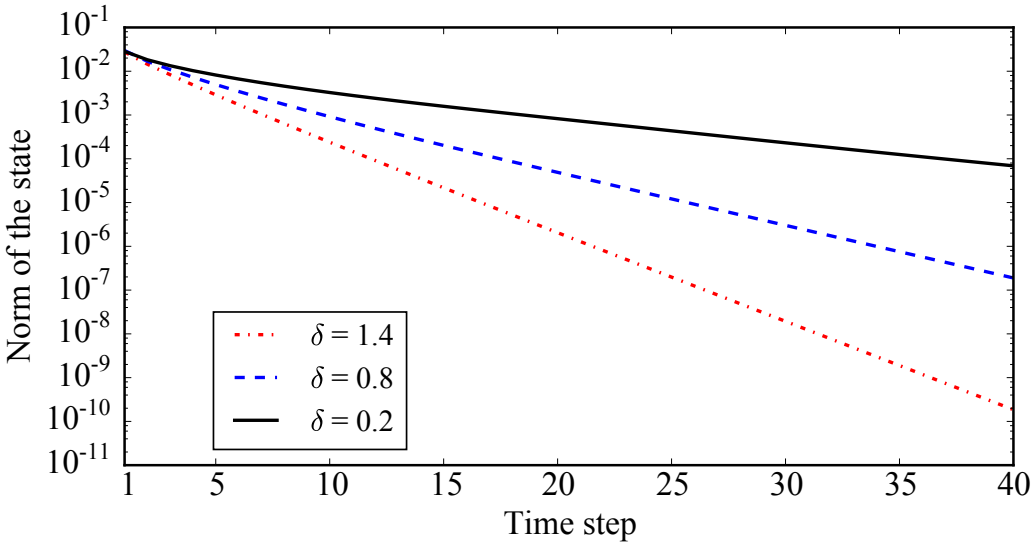
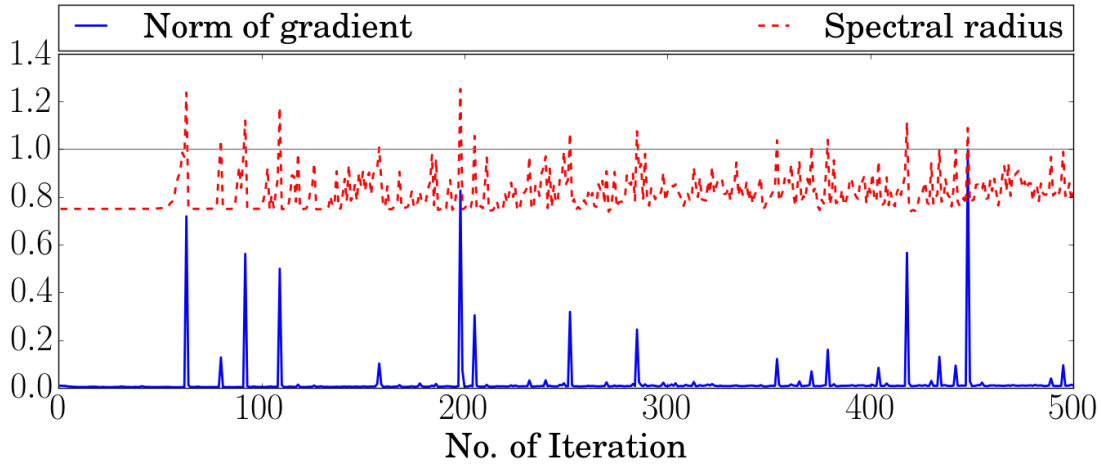


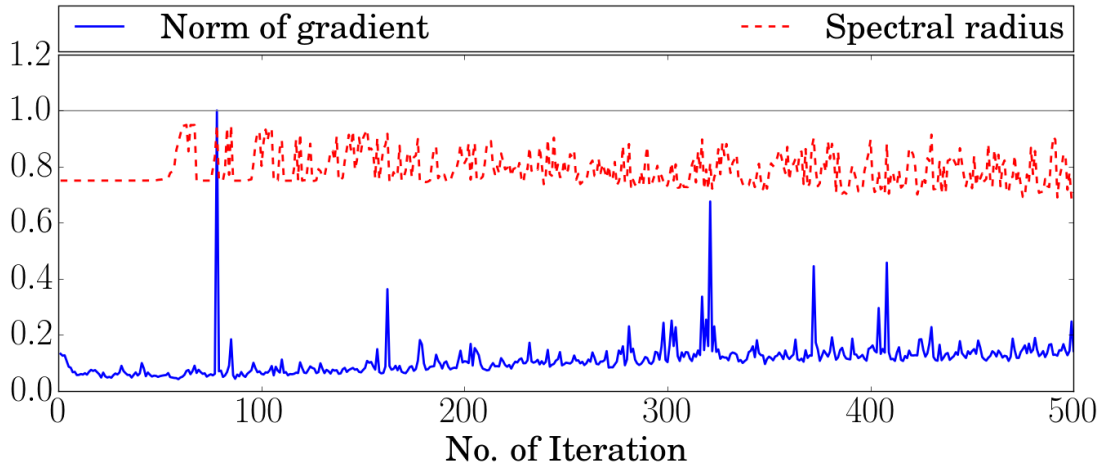

Fig. 3.2　$l_2$ norms of state vectors of GRU vs time steps. We did not apply any inputs and update state vectors after we applied input into GRU at time step of one (Copyright©2019 IEICE, [3] Fig. 1). The norms state vector converges to zero, and its convergence speed increased along with decreasing of $\delta$.

### 3.5.4　勾配とスペクトル半径の関係

提案法は状態の平衡点の安定性の変化が勾配爆発を引き起こすという仮説のもとで，状態を安定に制約している．本項ではこの仮説を検証するため，学習の最初の 500 反復の間の安定性を決める $\boldsymbol{J}$ のスペクトル半径の時間変化と勾配のノルムの時間変化（勾配クリッピングの結果はクリップ前の勾配）を Fig. 3.3 に示す．Fig. 3.3 (a) は閾値 5 の勾配クリッピングを使用した学習の結果で，Fig. 3.3 (b) は $\delta$ を 0.2 に設定した提案法の学習の様子である．なお，勾配のノルムは最大が 1 となるように正規化しており，正規化前の提案法の値は勾配クリッピングと比べ非常に小さい．Fig. 3.3 (a) より，勾配のノルムは平衡点が不安定化するスペクトル半径が 1 を超えるときに非常に大きな値をとっている．また，勾配が大きくなったのちに，スペクトル半径は 1 以下となっており，勾配爆発が生じた際の勾配の方向がスペクトル半径を小さくする方向に動いている．一方，提案法は Fig. 3.3 (b) より明らかなように，$\boldsymbol{W}_{hh}$ のスペクトルノルムを制約することにより，スペクトル半径が 1 よりも小さくなるように制約されており，勾配爆発を起こさずに学習が進んでいる．




(a) Gradient clipping (閾値 5).




(b) Proposed method (delta of 0.2).

Fig. 3.3 Gradient exploding in language modeling experiments (Copyright©2019, [3] Fig. 2). Norms of gradient are normalized to [0,1] for the clarity. When a spectral radius exceeds one, the gradient norm becomes extremely large values in gradient clipping (a). On the other hand, since the spectral radius does not exceed one, the gradient norm does not become large values in proposed method (b).

Table 3.6　Computation time for language modeling on PTB (delta 0.2, threshold 5)

| Run time (s) | | |
|---|---|---|
| Naive SVD | Truncated SVD | gradient clipping |
| $5.02 \times 10^4$ | $4.55 \times 10^4$ | $4.96 \times 10^4$ |

### 3.5.5　計算量の評価

#### (a)　計算時間の比較

PTB の言語モデルの実験において計算時間を比較した結果を Table 3.6 に示す．この表は提案法に通常の特異値分解（naive SVD）と truncated SVD を用いてそれぞれ学習させた計算時間，そして勾配クリッピングの計算時間である．表より，naive SVD を用いた提案法が勾配クリッピングと同程度の計算量であり，truncated SVD を用いた提案法が最も速かった．2.2.2 項で述べたように勾配クリッピングでは，モデル全体のパラメータに対する勾配のノルムを計算するため，言語モデルにおいて語彙数に依存した入出力層の大きなパラメータ数が計算時間に影響をあたえる．一方，提案法は GRU の状態ベクトルのサイズのみに依存し，さらに 3.3 節のように制約を超える特異値が存在しなければ高速に計算できる．結果として提案法と勾配クリッピングの計算時間は同程度となった．

#### (b)　特異値計算の必要な $s$ の比較

Truncated SVD を使った提案法で，特異値を求める個数 $s$ の学習中の変化を Fig. 3.4 に示す．これは前述の計算時間の実験と同じ PTB の言語モデルの結果である．$s$ は 1 反復毎に計算するが，わかりやすさのためにこれをエポック毎に平均して各エポックの平均の $s$ を図示した．図より，学習の初期においてほぼすべての特異値の計算が必要であった．しかし，学習が進むと学習率 $\eta$ が小さくなることと勾配が小さくなることの 2 つの要因によって，上界に使う $|\eta|||\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})||_F$ の値が小さくなり，$s$ の大きさは小さくなった．その結果，学習後半に勾配クリッピングより効率的に学習が可能で勾配クリッピングより速く学習が終わった．また $\delta$ の値による $s$ の違いを比較すると，低計算量による方法は特異値を更新する際は $(2-\delta)$ 以下となるように更新し，一方で上界を使った $s$ の推定には $(2-\delta)$ ではなく 2 としているため（Algorithm 3.2 の line 10），$\delta$ が大きいほど計算量が削減される．

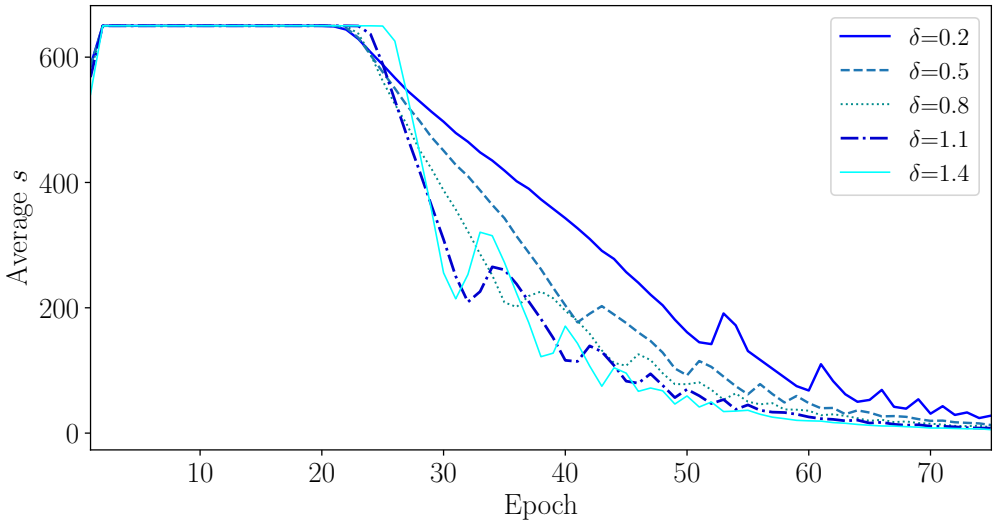



Fig. 3.4 Number of singular values $s$ computed by truncated SVD against epochs. $s$ was averaged in each epoch. $s$ decreased along with the progress of training.

### (c) よりタイトな上界による $s$ の推定

補題 3.6 では $s$ を推定するための上界として

$$\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) \leq \sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)}) + |\eta||||\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})||_F$$

という関係を用いた．しかしこの補題の証明中に示したように

$$\sigma_i(\hat{\boldsymbol{W}}_{hh}^{(\tau)}) \leq \sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)}) + |\eta|\sigma_1(\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})) \leq \sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)}) + |\eta||||\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta})||_F$$

という関係があるから，$\sigma_i(\boldsymbol{W}_{hh}^{(\tau-1)}) + |\eta|\sigma_1(\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta}))$ の方がよりタイトな上界となる．それにもかかわらず，ここまでの評価でフロベニウスノルムを用いた理由は簡単に並列計算可能で GPU との相性もよいと考えられるためである．ここではさらなる発展として，このタイトなスペクトルノルム $\sigma_1(\nabla_{\boldsymbol{W}_{hh}} C_{D_\tau}(\boldsymbol{\theta}))$ を使った上界による $s$ を評価した．$\delta = 0.2$ としたときの学習中のそれぞれの上界による $s$ の推定値を Fig. 3.5 に示す．図より，スペクトルノルムを用いた上界がよりタイトなため $s$ の大きさがフロベニウスノルムに基づく方法より小さい．そのため，特異値計算にかかるコストを小さく抑えることができる．一方で，上界の計算に必要な時間はスペクトルノルムの方が大きいことを確認した．学習全体の計算時間ではフロベニウスノルムに基づく方法が $5.4 \times 10^4$ 秒，スペクトルノルムに基づく方法が $4.9 \times 10^4$ 秒となり，スペクトルノルムに基づく上界の方が優れていた[*4]．上界の計算時間はフロベニウス

[*4] こちらは 3.5.5(a) と同じ計算環境を用意することができなかったため，フロベニウスノルムに基づく方法の計算時間が 3.5.5(a) の結果より大きい．

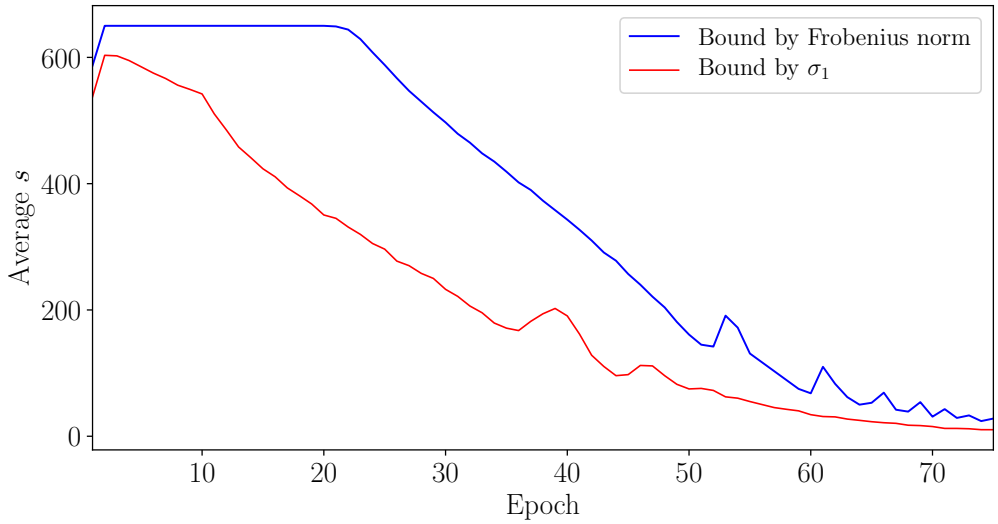



Fig. 3.5　Comparison estimated $s$ using upper bound based on a Frobenius norm with those using upper bound based on a spectral norm. $\delta$ was set to 0.2. Since the upper bound based on the spectral norm is tighter than the upper bound based on the Frobenius norm, red line is always smaller than blue line.

ノルムの方が短いことから，学習全体の速度に関しては実験データや計算環境に依存するが，この方法も有効である．

## 3.6　まとめ

本章では GRU の状態の振る舞いを解析し，勾配爆発を防ぐ新たな学習法を提案した．言語モデルと音楽モデルの実験において提案法が勾配クリッピングと同程度の計算時間で勾配爆発を抑制し，さらに勾配クリッピングより高い精度を達成できることを示した．勾配クリッピングは試行錯誤的にその閾値を調整する必要があるが，閾値がモデルに与える影響は不明瞭であり，またその調整の範囲も限定されていない．一方，提案法の $\delta$ も多少の調整を必要とするが，その効果は長期記憶性能と関係して解釈しやすく，またその範囲も $0 < \delta < 2$ に限定されるために GRU を使ったモデル構築にかかる試行錯誤を削減できる．ただし，入力の存在を考慮していないため，入力によって状態が平衡点の吸引領域から出ないように十分小さい必要があり，その入力に関する条件は今後の課題である．

# 第 4 章

# Softmax の表現能力に関する解析と高い表現力を持つ出力関数の提案

本章では，既存の出力関数 softmax より表現能力の高い出力関数を提案する[*1]．まず，離散値をとる時系列データであって，言語モデルのような取りうる値の数が大きなデータを表現するために必要な出力関数の特性を考察する．次に，この特性を満たす新たな出力関数を提案する．最後に，提案方法と既存の出力関数およびいくつかの代替関数の候補について実験的に性能を評価する．

## 4.1 はじめに

深層学習ではタスクによってさまざまなモデルが使用されるが，クラス分類や言語モデルなどの離散確率を扱う多くのタスクにおいて，使用される出力関数は softmax である [9, 11, 16–20]．これは出力ベクトルの全要素の和が 1，各値が $[0, 1]$ の範囲をとるような非線形関数であり，これにより深層学習の出力を各次元に対する確率として見なすことができる．

しかし [21] で softmax が深層学習のモデルの表現能力のボトルネックとなりうることが示された．これは softmax bottleneck と呼ばれている．これは，深層学習のモデルの最終層のユニット数が出力の数から 1 を引いた値より小さいとき，softmax を使ったモデルでは表現できない確率分布があるというものである．特に，言語モデルに softmax を出力とする深層学習モデルを用いる場合，状態のユニット数が数百であるのに対して出力数は語彙数に対応した数万という値になるため，softmax bottleneck が顕在化すると考えられる．[21] では，softmax

[*1] 本章は電子情報通信学会和文論文誌 D (Copyright©2020 IEICE) に含まれる “Softmax Bottleneck の再解釈とそれを解消する出力関数の評価”[4] および NeurIPS2018 発表論文 “Sigsoftmax: Reanalysis of the softmax bottleneck”[82] に基づく．本章の一部データは電子情報通信学会和文論文誌 D において発表され，また電子情報通信学会の許可のもと論文 [4] から再利用されている．

bottleneck を解決する方法として，softmax を混合した mixture of softmax（MoS）を提案している．しかし MoS は追加のパラメータによって計算量が多くなり，また混合数を調整しなければならず softmax に代わる出力関数とはいえなかった．

本章では，softmax bottleneck を解決する出力関数を提案し，言語モデルの実験によって評価する．まず，softmax の対数である log-softmax の値域から softmax bottleneck を再考し，log-softmax の値域が含まれる空間の次元を解析して，改めて softmax bottleneck を示す．この解析を基にそれを解消する新たな出力関数として，それぞれ sigmoid，softplus，rectified linear unit（ReLU）から構成される出力関数と sigmoid と指数関数の積をとった sigsoftmax を提案する．言語モデルの実験によって，softmax とこれらの出力関数および混合した出力関数の性能を評価する．

### 4.1.1 Softmax bottleneck

Softmax bottleneck について説明するため，2.2.4 項で示した recurrent neural network（RNN）による言語モデルを考える．コーパスとして $\boldsymbol{Y} = (Y_1, \ldots, Y_T)$ が与えられたとすると，その同時確率 $p(\boldsymbol{Y})$ は $P(\boldsymbol{Y}) = \prod_t P(Y_t|Y_{<t}) = \prod_t P(Y_t|X_t)$ と分解できる．ここで $X_t = Y_{<t} = (Y_1, \ldots, Y_{t-1})$ をコンテキストと呼ぶ．クラス分類と同様に $P(Y_t|X_t)$ を softmax で表現する．自然言語 $\mathcal{L}$ が有限で $N$ 個のコンテキストとそれに対する条件付き確率の集合 $\mathcal{L} = \{(x_1, P^*(Y|x_1)), \ldots, (x_N, P^*(Y|x_N))\}$ で表現できると仮定すると，言語モデルの目的は，分布のモデル $P_{\boldsymbol{\theta}}(Y|X)$ を真の確率分布 $P^*(Y|X)$ と一致させることである．以上の仮定のもとで，softmax bottleneck [21] では，$N$ 個のコンテキストに対する $\boldsymbol{h}(\boldsymbol{x}_N)$ を並べた行列

$$\boldsymbol{H}_{\boldsymbol{\theta}} = \begin{bmatrix} \boldsymbol{h}(x_1)^\mathsf{T} \\ \boldsymbol{h}(x_2)^\mathsf{T} \\ \vdots \\ \boldsymbol{h}(x_N)^\mathsf{T} \end{bmatrix} \tag{4.1}$$

と重み行列 $\boldsymbol{W}$ および各単語の真の確率の対数をとって並べた行列，

$$\boldsymbol{A} = \begin{bmatrix} \log P^*(y_1|x_1), & \log P^*(y_2|x_1), & \ldots & \log P^*(y_M|x_1) \\ \log P^*(y_1|x_2), & \log P^*(y_2|x_2), & \ldots & \log P^*(y_M|x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \log P^*(y_1|x_N), & \log P^*(y_2|x_N), & \ldots & \log P^*(y_M|x_N) \end{bmatrix} \tag{4.2}$$

を考える．いま，softmax に入る前の隠れベクトル $\boldsymbol{h}(\boldsymbol{x})$ が $d$ 個の要素を持つベクトル $\boldsymbol{h}(\boldsymbol{x}) \in \mathbb{R}^d$ であるとすると，softmax bottleneck は次の定理で与えられる．

**定理 4.1** (Softmax bottleneck [21])**.** $d < \mathrm{rank}(\boldsymbol{A}) - 1$ であれば，モデルにいかなる万能近似

可能性を持つ関数 $\mathcal{U}$ と，いかなるパラメータ $\boldsymbol{\theta}$ を用いても，$P_{\boldsymbol{\theta}}(Y|x) \neq P^*(Y|x)$ となる文脈 $x$ が存在する．

この定理より，出力層の状態ベクトル $\boldsymbol{h}$ の長さ $d$ が真の確率の対数のランクより小さいと，真の確率を softmax を用いたモデルが表現できないことがわかる．言語モデルでは $M$ が語彙数に相当してとりうる値の数が数万であるのに対し，計算量などの問題から $d$ はしばしば数百次元に設定するため，このボトルネックが顕在化すると考えられる．なお [21] における，この定理の証明を付録 A に示す．

### 4.1.2 Mixture of softmax

Softmax bottleneck を解消するために，混合 softmax (MoS) が提案されている．MoS では $K$ 個の softmax を混合し，

$$P_{\boldsymbol{\theta}}(y_i|x) = \sum_{k=1}^{K} \pi(x,k) \frac{\exp([\boldsymbol{W}\boldsymbol{h}(x,k)]_i)}{\sum_m \exp([\boldsymbol{W}\boldsymbol{h}(x,k)]_m)} \tag{4.3}$$

とする．ただし，$\pi(x,k)$ は $k$ 番目の softmax に対する重み（$\sum_{k=1}^{K} \pi(x,k) = 1$）であり，$\boldsymbol{h}(x,k)$ はその $k$ 番目の要素に対応する隠れベクトルである．$\boldsymbol{h}'(x)$ を出力層の入力となる隠れベクトルとすると，重みと $\boldsymbol{h}(x,k)$ はそれぞれ

$$\pi(x,k) = \frac{\exp(\boldsymbol{w}_{\pi,k}^{\mathsf{T}} \boldsymbol{h}'(x))}{\sum_{k'=1}^{K} \exp(\boldsymbol{w}_{\pi,k'}^{\mathsf{T}} \boldsymbol{h}'(x))} \tag{4.4}$$

$$\boldsymbol{h}(x,k) = \tanh(\boldsymbol{W}_{h,k} \boldsymbol{h}'(x)) \tag{4.5}$$

として計算される．MoS によって近似された $\boldsymbol{A}$ のランクが任意の値を取りうるため，softmax bottleneck を解消できるとされている [21].

しかし，MoS は $K$ という試行錯誤的に調整しなければならないハイパーパラメータと重み $\boldsymbol{W}_{h,k}$ と $\boldsymbol{w}_{\pi,k}$ という追加のパラメータを持ち，新たな重みに対する勾配計算により，計算量が増加するため，softmax の代替となる出力関数とはいえない．

## 4.2 ボトルネックを解消する関数

### 4.2.1 Log-softmax からの解釈

[21] は softmax が表現能力のボトルネックとなりうることを示したが，その原因を陽に示してはいなかった．そこで，softmax bottleneck の原因を明らかにするため，log-softmax の値域から softmax bottleneck を再解釈する．

第 2 章で述べたように，softmax による学習は負の対数尤度の最小化で行われるため，log-softmax が学習に用いられる．log-softmax は softmax の出力に対数をとったものであり，

$$[\log(\boldsymbol{f}_s(\boldsymbol{z}))]_i = \log\left(\frac{\exp(z_i)}{\sum_m \exp(z_m)}\right) = z_i - \log\left(\sum_m \exp(z_m)\right) \tag{4.6}$$

である．これをベクトルで表現すると，

$$\log\left(\boldsymbol{f}_s(\boldsymbol{z})\right) = \boldsymbol{z} - \log\left(\sum_m \exp(z_m)\right)\mathbf{1} \tag{4.7}$$

となる．ただし，$\mathbf{1}$ は全要素が 1 のベクトルである．Softmax が真の確率を表現するためには，真の確率の対数をとったものと一致するために十分に log-softmax の値域が大きくなければならない．いま，隠れベクトル $\boldsymbol{h}$ が $\mathbb{R}^d$ の任意のベクトルをとれるとし，$d \leq M$ とする．また，重み $\boldsymbol{W} \in \mathbb{R}^{M\times d}$ がフルランクでランク $d$ だとする．すると，softmax の入力ベクトルの空間 $S$ ($\boldsymbol{z} \in S$) は $d$ 次元のベクトル空間となる[*2]．このとき log-softmax の値域 $\{\log(\boldsymbol{f}_s(\boldsymbol{z}))|\boldsymbol{z} \in S\}$ に対して次の定理が成り立つ．

**定理 4.2.** $S \subseteq \mathbb{R}^M$ を $d$ 次元ベクトル空間とし，$\boldsymbol{z} \in S$ を log-softmax の入力とすると，log-softmax の値域 $\{\log(\boldsymbol{f}_s(\boldsymbol{z}))|\boldsymbol{z} \in S\}$ は，たかだか $(d+1)$ 次元ベクトル空間の部分集合である．

**証明.** $S \subseteq \mathbb{R}^M$ は $d$ 次元ベクトル空間であり，その $d$ 個の基底を用いて $S = \{\sum_{l=1}^{d} k^{(l)}\boldsymbol{u}^{(l)}|k^{(l)} \in \mathbb{R}\}$ である．ただし，$\boldsymbol{u}^{(l)} \in \mathbb{R}^M$ は線形独立なベクトルであり，$k^{(l)}$ はその係数である．式 (4.7) から $\boldsymbol{u}^{(l)}$ と $k^{(l)}$ を用いると，log-softmax の値域 $\{\log(\boldsymbol{f}_s(\boldsymbol{z}))|\boldsymbol{z} \in S\}$ は

$$\{\log(\boldsymbol{f}_s(\boldsymbol{z}))|\,\boldsymbol{z} \in S\} = \left\{\sum_{l=1}^{d} k^{(l)}\boldsymbol{u}^{(l)} - c\left(\sum_{l=1}^{d} k^{(l)}\boldsymbol{u}^{(l)}\right)\mathbf{1}|k^{(l)} \in \mathbb{R}\right\} \tag{4.8}$$

となる．ただし，

$$c\left(\sum_{l=1}^{d} k^{(l)}\boldsymbol{u}^{(l)}\right) = \log\left(\sum_{m=1}^{M} \exp(\left[\sum_{l=1}^{d} k^{(l)}\boldsymbol{u}^{(l)}\right]_m)\right)$$

[*2] $\boldsymbol{h}$ が任意のベクトルではない場合，データ点 $\boldsymbol{x}$ の集合を $\mathcal{X}$ とすると，$\{\boldsymbol{h}(\boldsymbol{x})|\boldsymbol{x} \in \mathcal{X}\} \subseteq \mathbb{R}^d$ であり，$\{\boldsymbol{W}\boldsymbol{h}(\boldsymbol{x})|\boldsymbol{x} \in \mathcal{X}\}$ は $d$ 次元以下の部分空間の部分集合となりうる．また $\boldsymbol{W}$ がフルランクではない場合も $d$ 次元以下の部分集合となりうる．しかしどちらの場合においても $d$ 次元の部分空間の部分集合である．

である．これは $d$ 個の線形独立なベクトル $\boldsymbol{u}^{(l)}$ と $\mathbf{1}$ の線形結合である．そのため

$$\left\{\sum_{l=1}^{d} k^{(l)}\boldsymbol{u}^{(l)} - c\left(\sum_{l=1}^{d} k^{(l)}\boldsymbol{u}^{(l)}\right)\mathbf{1} | k^{(l)} \in \mathbb{R}\right\} \subseteq \left\{\sum_{l=1}^{d} k^{(l)}\boldsymbol{u}^{(l)} + k^{(d+1)}\mathbf{1} | k^{(l)} \in \mathbb{R}\right\} \tag{4.9}$$

が成り立つ．ただし，$\{\sum_{l=1}^{d} k^{(l)}\boldsymbol{u}^{(l)} + k^{(d+1)}\mathbf{1}|k^{(l)} \in \mathbb{R}\}$ は $\boldsymbol{u}^{(l)}$ と $\mathbf{1}$ が張る空間である．$Y$ をベクトル空間 $\{\sum_{l=1}^{d} k^{(l)}\boldsymbol{u}^{(l)} + k^{(d+1)}\mathbf{1}|k^{(l)} \in \mathbb{R}\}$ とすると，$Y$ の次元は

$$\dim(Y) = \begin{cases} d+1, & \text{if } \mathbf{1} \notin S \\ d, & \text{if } \mathbf{1} \in S \end{cases} \tag{4.10}$$

となり，$Y$ は $d$ か $(d+1)$ 次元の $\mathbb{R}^M$ の部分空間である．式 (4.9) と (4.10) より，log-softmax の出力は $(d+1)$ 次元のベクトル空間に存在する． □

式 (4.7) のより log-softmax は入力ベクトル $\boldsymbol{z}$ と $\mathbf{1}$ の線形結合である．この線形結合によって増える線形独立なベクトルの数は，元の入力ベクトルの線形独立なベクトルの数からたかだか 1 である．よって，log-softmax の出力の線形独立なベクトルの数はたかだか $(d+1)$ であり，$(d+1)$ 次元の部分空間の部分集合である．$d=1$，$M=3$ としたときの図を Fig. 4.1 に示す．図では青曲線が真の確率分布がとりうる範囲を示しており，これは 3 次元上の曲線となっている．一方，softmax の対数の出力の範囲は黒曲線であり，平面上の曲線となっている．

定理 4.2 より，もし真の確率の対数が $(d+1)$ 次元より大きな空間に広がっていると，softmax ではその確率すべてを表現できない．前述したように，この原因は log-softmax が $\boldsymbol{z}$ と $\mathbf{1}$ の線形結合であるためであり，これは $\log(\exp(z)) = z$ となるためである．$\log(g(z))$ が非線形関数となるような関数を用いると，線形結合ではなくなり，$(d+1)$ 次元より大きな空間に出力の対数が広がる．そこで新たな活性化関数について考える．

### 4.2.2 出力関数に求められる性質

前項では softmax で使われる関数を指数関数以外の非線形関数にすることで softmax bottleneck を解消しうることを示した．ここではその非線形関数に求められる性質について述べる．新しい出力関数 $\boldsymbol{f}(\cdot)$ を

$$[\boldsymbol{f}(\boldsymbol{z})]_i = \frac{[\boldsymbol{g}(\boldsymbol{z})]_i}{\sum_m [\boldsymbol{g}(\boldsymbol{z})]_m} \tag{4.11}$$

とする．この新しい関数は非線形関数 $\boldsymbol{g}(\boldsymbol{z})$ と正規化のための除算で構成されている．Softmax の代替として，$\boldsymbol{f}(\boldsymbol{z})$ とその $\boldsymbol{g}(\boldsymbol{z})$ は以下のすべての性質を満たしていることが望ましい．

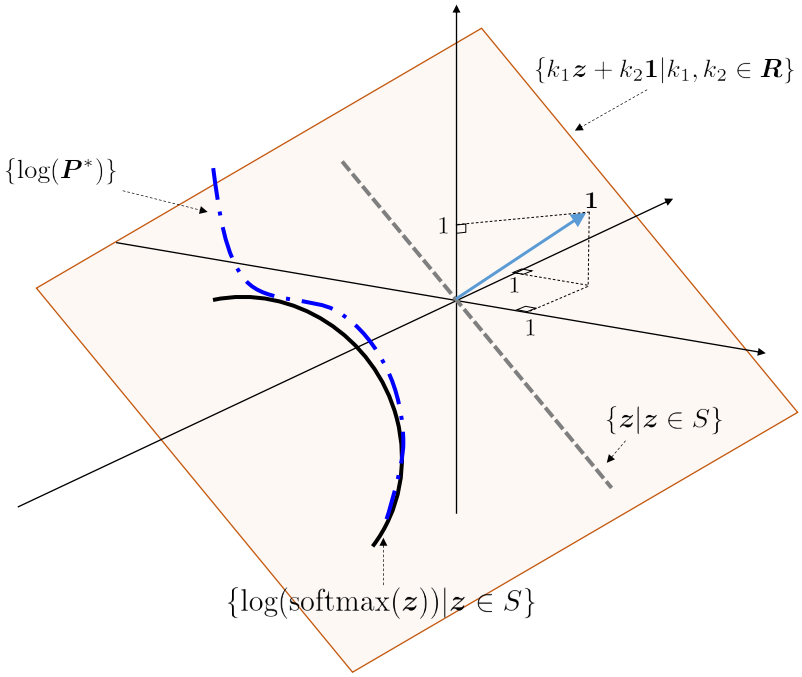



Fig. 4.1　Inputs space $\{\boldsymbol{z}|\boldsymbol{z}\in S\}$ and the range of outputs of log-softmax for $d=1$. We can see that log-softmax is curve on the plane $\{k_1\boldsymbol{z}+k_2\mathbf{1}|k_1,k_2\in\mathbb{R}\}$, and it cannot match the true probabilities $\log(P*)$ in three dimensional space (Copyright©2020 IEICE, [4] Fig. 1).

1. $\log(\boldsymbol{g}(\boldsymbol{z}))$ の非線形性:
   4.2.1 項で述べたように, $\log(\exp(\boldsymbol{z}))$ が $\boldsymbol{z}$ であるため，softmax は表現能力のボトルネックになる．$\log(\boldsymbol{g}(\boldsymbol{z}))$ が線形の関数である限り，$\{\log(\boldsymbol{f}(\boldsymbol{z}))|\boldsymbol{z}\in S\}$ は $(d+1)$ 次元ベクトル空間の部分集合であるため，ボトルネックの解消には $\log(\boldsymbol{g}(\boldsymbol{z}))$ が非線形でなければならない．
2. 数値的な安定性:
   深層学習では勾配に基づく最適化が行われる．$[\boldsymbol{f}(\boldsymbol{z})]_i$ の $z_j$ に関する微分は
   $$\frac{\partial\log([\boldsymbol{f}(\boldsymbol{z})]_i)}{\partial z_j}=\frac{1}{[\boldsymbol{f}(\boldsymbol{z})]_i}\frac{\partial\,[\boldsymbol{f}(\boldsymbol{z})]_i}{\partial z_j} \tag{4.12}$$
   となる．ここで，$[\boldsymbol{f}(\boldsymbol{z})]_i$ による除算が存在する．$[\boldsymbol{f}(\boldsymbol{z})]_i$ は入力に対して誤ったラベルを出力するような学習初期において小さな値をとりうるため，ゼロ割りを発生させる可能性がある．そこで代替関数は softmax のように除算を含まずに勾配計算できることが望ましい．
3. 非負性:
   式 (4.11) において出力が [0, 1] に制限されるためには $\boldsymbol{g}(\boldsymbol{z})$ の全要素が非負（もしくは非正）でなければならない．そのため $\boldsymbol{g}(\boldsymbol{z})$ は非負の関数とする，すなわち，$[\boldsymbol{g}(\boldsymbol{z})]_i\geq 0$.

4. 単調増加性:

   Softmax は argmax の平滑化バージョンとしても知られており，この性質を満たすためには $\boldsymbol{g}(\boldsymbol{z})$ は単調増加である必要がある [17, 83]. $\boldsymbol{g}(\boldsymbol{z})$ が単調増加であれば，最も確率の大きい要素を softmax を計算せずに，$\boldsymbol{z}$ の各要素の比較により求められる.

### 4.2.3 Softmax bottleneck を解消する出力関数

深層学習でよく使用される非線形関数を用いて softmax bottleneck を解消することができる出力関数と，その性質について以下に述べる.

#### (a) ReLU に基づく出力関数

$\mathrm{ReLU}(z) = \max(0, z)$ を使った出力関数は，

$$[\boldsymbol{f}(\boldsymbol{z})]_i = \frac{\mathrm{ReLU}(z_i)}{\sum_m \mathrm{ReLU}(z_m)} \tag{4.13}$$

である．ReLU による関数の性質について以下に示す.

1. $\log(\boldsymbol{g}(\boldsymbol{z}))$ の非線形性：

$$[\log(\mathrm{ReLU}(\boldsymbol{z}))]_i = \begin{cases} \log(z_i), & \text{if} \quad z_i > 0 \\ -\infty, & \text{if} \quad z_i \leq 0 \end{cases}$$

2. 数値的な安定性：

   ReLU からなる関数の微分は

$$\frac{\partial \log [\boldsymbol{f}(\boldsymbol{z})]_i}{\partial z_j} = \begin{cases} \dfrac{1}{\mathrm{ReLU}(z_i)} \dfrac{\partial \mathrm{ReLU}(z_i)}{\partial z_j} \\ \quad - \dfrac{1}{\sum_m \mathrm{ReLU}(z_m)} \dfrac{\partial \mathrm{ReLU}(z_i)}{\partial z_j}, & \text{if} \quad i = j \\ - \dfrac{1}{\sum_m \mathrm{ReLU}(z_m)} \dfrac{\partial \mathrm{ReLU}(z_j)}{\partial z_j}, & \text{if} \quad i \neq j \end{cases}$$

   であり，ReLU による除算を含み，ReLU が 0 となりやすいから，数値的に不安定である.

3. 非負性：$[\boldsymbol{g}(\boldsymbol{z})]_i = \max(z_i, 0) \geq 0$
4. 単調増加性: $z_1 \leq z_2 \Rightarrow \max(z_1, 0) \leq \max(z_2, 0)$

### (b) Sigmoid に基づく出力関数

Sigmoid: $\mathrm{sigm}(z) = \dfrac{1}{1+\exp(-z)}$ を使った出力関数は,

$$[\boldsymbol{f}(\boldsymbol{z})]_i = \frac{\mathrm{sigm}(z_i)}{\sum_m \mathrm{sigm}(z_m)} \tag{4.14}$$

である．この関数は次のように所望の性質を満たす．

1. $\log(\boldsymbol{g}(\boldsymbol{z}))$ の非線形性：$\log(\mathrm{sigm}(\boldsymbol{z})) = \boldsymbol{z} - \log(\boldsymbol{1} + \exp(\boldsymbol{z}))$
2. 数値的な安定性：

$$\frac{\partial \log [\boldsymbol{f}(\boldsymbol{z})]_i}{\partial z_j} = \begin{cases} (1 - [\boldsymbol{f}(\boldsymbol{z})]_j)(1 - \mathrm{sigm}(z_j)), & \text{if } \ i = j \\ -[\boldsymbol{f}(\boldsymbol{z})]_j \, (1 - \mathrm{sigm}(z_j)), & \text{if } \ i \neq j \end{cases}$$

   この計算は除算を含まないという特徴をもつ．
3. 非負性：$[\boldsymbol{g}(\boldsymbol{z})]_i = \mathrm{sigm}(z_i) \geq 0$
4. 単調増加性: $z_1 \leq z_2 \Rightarrow \mathrm{sigm}(z_1) \leq \mathrm{sigm}(z_2)$

### (c) Softplus に基づく出力関数

Softplus $g(z) = \log(1 + \exp(z))$ を使った出力関数は,

$$[\boldsymbol{f}(\boldsymbol{z})]_i = \frac{\log(1 + \exp(z_i))}{\sum_m \log(1 + \exp(z_m))} \tag{4.15}$$

である．この関数は以下の性質を持つ．

1. $\log(\boldsymbol{g}(\boldsymbol{z}))$ の非線形性：$\log(\log(1 + \exp(z)))$
2. 数値的な安定性：

   Softplus からなる関数の微分は

$$\frac{\partial \log [\boldsymbol{f}(\boldsymbol{z})]_i}{\partial z_j} = \begin{cases} \dfrac{\mathrm{sigm}(z_j)(1 - [\boldsymbol{f}(\boldsymbol{z})]_j)}{\log(1 + \exp(z_j))}, & \text{if } \ i = j \\ -\dfrac{\mathrm{sigm}(z_m)}{\sum_m \log(1 + \exp(z_m))}, & \text{if } \ i \neq j \end{cases}$$

   であり，この計算は $i = j$ のときに $\log(1+\exp(z_j))$ による除算が含まれている．ただし，$\mathrm{sigm}(z_j) = 1/(1+\exp(-z_j))$ がかかっており，これは $1/((1+\exp(-z_j))\log(1+\exp(z_j))$ となるため，$z_j \to -\infty$ で 1 となる．しかし，予備実験で評価したところ，学習中に損失関数が発散して数値的に不安定となることを観測した．これは $\log(1 + \exp(z_j))$ の除算の数値計算による問題と思われる．計算順序等によって学習を安定化させることがで

きる可能性はあるが，本論文では後述するように，小さな定数を分母に加えることで学習を安定化して実験で評価する．

3. 非負性：$[\boldsymbol{g}(\boldsymbol{z})]_i = \log(1+\exp(z_i)) \geq 0$
4. 単調増加性: $z_1 \leq z_2 \Rightarrow \log(1+\exp(z_1)) \leq \log(1+\exp(z_2))$

以上では深層学習によく使用される関数の中では，ReLU は勾配計算が不安定となりうること，sigmoid は所望の性質をすべて満たすことを示した．しかし，sigmoid はその最大値が 1 よりも小さく，これによって表現力が制限される可能性がある．また，これらの関数は非線形な関数であるため，値域の含まれる空間の次元だけで値域の大小を比較することはできない．つまり，softmax bottleneck を解消することができても，出力の値域が高次元に広がるが，その集合の大きさは softmax の値域より小さくなる可能性があるため，実際のところ表現能力が向上しているかわからない．そこで上記の性質を満たしてボトルネックを解消しつつ，そして softmax と値域の大きさを直接比較可能な出力関数を提案する．

### 4.2.4 Sigsoftmax の提案

提案する sigsoftmax は次の通りである．

**定義 4.1.** Sigsoftmax は，

$$[\boldsymbol{f}(\boldsymbol{z})]_i = \frac{\exp(z_i)\mathrm{sigm}(z_i)}{\sum_m \exp(z_m)\mathrm{sigm}(z_m)} \tag{4.16}$$

で与えられる．

#### (a) Sigsoftmax の性質

Sigsoftmax がボトルネックを解消し，所望の性質を満たすことを示す．4.2.1 項の softmax と同様に log-sigsoftmax の値域を調べる．Sigmoid の対数は，

$$\log(\mathrm{sigm}(z)) = \log(\frac{1}{1+\exp(-z)}) = z - \log(1+\exp(z))$$

なので，log-sigsoftmax は

$$\log(\boldsymbol{f}(\boldsymbol{z})) = 2\boldsymbol{z} - \log(\boldsymbol{1}+\exp(\boldsymbol{z})) + c'(\boldsymbol{z})\boldsymbol{1} \tag{4.17}$$

となる．ただし，$c'(\boldsymbol{z}) = \log(\sum_m \exp(z_m)\mathrm{sigm}(z_m))$ である．$\log(\boldsymbol{1}+\exp(\boldsymbol{z}))$ は softplus であり，非線形関数である [18]. Log-sigsoftmax が非線形関数で構成されているため，出力ベクトルは $(d+1)$ 以上の線形独立ベクトルを持ちうる．そのため次の定理が成り立つ．

**定理 4.3.** $S \subseteq \mathbb{R}^M$ を $d$ 次元ベクトル空間とし，$\boldsymbol{z} \in S$ を log-sigsoftmax の入力とする．このとき，log-sigsoftmax の値域 $\{\log(\boldsymbol{f}(\boldsymbol{z}))|\boldsymbol{z} \in S\}$ が $d+1$ 次元のベクトル空間に包含されないような $d$ 及び $S$ が少なくとも一つある．

**証明.** 背理法によって証明する．もし定理 4.3 が成り立たないならば log-sigsoftmax の値域 $\{\log(\boldsymbol{f}(\boldsymbol{z}))|\boldsymbol{z} \in S\}$ はすべて $(d+1)$ 次元のベクトル空間の部分集合である．この命題は定理 4.3 の否定であるため，これの反例を示すことで定理 4.3 を証明する．反例として $S$ が 1 次元のベクトル空間（つまり $d=1$）とし，$S = \{k\boldsymbol{u}|k \in R\}$ で $\boldsymbol{u} = [1,2,0]^\mathsf{T}$ であるとする．この条件において，定義より log-sigsoftmax の出力は

$$\log(\boldsymbol{f}(\boldsymbol{z})) = \begin{bmatrix} 2k \\ 4k \\ 0 \end{bmatrix} - \begin{bmatrix} \log(1+\exp(k)) \\ \log(1+\exp(2k)) \\ \log 2 \end{bmatrix} - \log\left(\frac{1+2\mathrm{sigm}(k)\exp(k)+2\mathrm{sigm}(2k)\exp(2k)}{2}\right)\mathbf{1} \tag{4.18}$$

となる．$S = \{k\boldsymbol{u}|k \in R\}$ から 3 つの出力 $\boldsymbol{z}_1 = [0,0,0]^\mathsf{T}, \boldsymbol{z}_2 = [1,2,0]^\mathsf{T}, \boldsymbol{z}_3 = [-1,-2,0]^\mathsf{T}$ を選び，その出力について調べる．Log-sigsoftmax の値域は

$$\log(\boldsymbol{f}(\boldsymbol{z}_1)) = -\log 3\mathbf{1} \tag{4.19}$$

$$\log(\boldsymbol{f}(\boldsymbol{z}_2)) = \begin{bmatrix} 2-\log(1+\exp(1)) \\ 4-\log(1+\exp(2)) \\ -\log 2 \end{bmatrix} - \log\left(\frac{1+2\mathrm{sigm}(1)\exp(1)+2\mathrm{sigm}(2)\exp(2)}{2}\right)\mathbf{1} \tag{4.20}$$

$$\log(\boldsymbol{f}(\boldsymbol{z}_3)) = \begin{bmatrix} -2-\log(1+\exp(-1)) \\ -4-\log(1+\exp(-2)) \\ -\log 2 \end{bmatrix} - \log\left(\frac{1+2\mathrm{sigm}(-1)\exp(-1)+2\mathrm{sigm}(-2)\exp(-2)}{2}\right)\mathbf{1} \tag{4.21}$$

である．線形独立性を調べるために

$$\alpha_1\log(\boldsymbol{f}(\boldsymbol{z}_1)) + \alpha_2\log(\boldsymbol{f}(\boldsymbol{z}_2)) + \alpha_3\log(\boldsymbol{f}(\boldsymbol{z}_3)) = \mathbf{0} \tag{4.22}$$

の解を調べる．もしその解がただ一つ $\alpha_1 = \alpha_2 = \alpha_3 = 0$, $\log(\boldsymbol{f}(\boldsymbol{z}_1))$, $\log(\boldsymbol{f}(\boldsymbol{z}_2))$ であれば

$\log(\boldsymbol{f}(\boldsymbol{z}_3))$ は線形独立である．式 (4.22) のそれぞれの要素は次の連立方程式となる：

$$
\begin{cases}
-\alpha_1 \log 3 + \alpha_2 \left\{ 2 - \log(1+\exp(1)) - \log\left(\dfrac{1+2\mathrm{sigm}(1)\exp(1)+2\mathrm{sigm}(2)\exp(2)}{2}\right) \right\} \\
\qquad + \alpha_3 \{ -2 - \log(1+\exp(-1)) \\
\qquad\qquad - \log\left(\dfrac{1+2\mathrm{sigm}(-1)\exp(-1)+2\mathrm{sigm}(-2)\exp(-2)}{2}\right) \Big\} = 0 \qquad (4.23) \\
-\alpha_1 \log 3 + \alpha_2 \left\{ 4 - \log(1+\exp(2)) - \log\left(\dfrac{1+2\mathrm{sigm}(1)\exp(1)+2\mathrm{sigm}(2)\exp(2)}{2}\right) \right\} \\
\qquad + \alpha_3 \{ -4 - \log(1+\exp(-2)) \\
\qquad\qquad - \log\left(\dfrac{1+2\mathrm{sigm}(-1)\exp(-1)+2\mathrm{sigm}(-2)\exp(-2)}{2}\right) \Big\} = 0 \qquad (4.24) \\
-\alpha_1 \log 3 + \alpha_2 \{ -\log(2) - \log\left(\dfrac{1+2\mathrm{sigm}(1)\exp(1)+2\mathrm{sigm}(2)\exp(2)}{2}\right) \} \\
\qquad + \alpha_3 \{ -\log 2 \\
\qquad\qquad - \log\left(\dfrac{1+2\mathrm{sigm}(-1)\exp(-1)+2\mathrm{sigm}(-2)\exp(-2)}{2}\right) \Big\} = 0 \qquad (4.25)
\end{cases}
$$

式 (4.25) から

$$
\begin{aligned}
\alpha_1 = & \frac{\alpha_2}{\log(3)} \left\{ -\log(2) - \log\left(\frac{1+2\mathrm{sigm}(1)\exp(1)+2\mathrm{sigm}(2)\exp(2)}{2}\right) \right\} \\
& + \frac{\alpha_3}{\log 3} \left\{ -\log 2 - \log\left(\frac{1+2\mathrm{sigm}(-1)\exp(-1)+2\mathrm{sigm}(-2)\exp(-2)}{2}\right) \right\} \qquad (4.26)
\end{aligned}
$$

が成り立つ．式 (4.26) を式 (4.23) と式 (4.24) に代入すると

$$
\begin{cases}
\alpha_2 \{ 2 - \log(1+\exp(1)) + \log(2) \} + \alpha_3 \{ -2 - \log(1+\exp(-1)) + \log(2) \} = 0 & (4.27) \\
\alpha_2 \{ 4 - \log(1+\exp(2)) + \log(2) \} + \alpha_3 \{ -4 - \log(1+\exp(-2)) + \log(2) \} = 0 & (4.28)
\end{cases}
$$

が得られる．式 (4.27) より

$$
\alpha_2 = \alpha_3 \frac{\{ 2 + \log(1+\exp(-1)) - \log(2) \}}{\{ 2 - \log(1+\exp(1)) + \log(2) \}} \qquad (4.29)
$$

が成り立ち，式 (4.29) を式 (4.28) へ代入すると

$$
\begin{aligned}
\alpha_3 \Bigg[ & \frac{\{ 2 + \log(1+\exp(-1)) - \log(2) \} \{ 4 - \log(1+\exp(2)) + \log(2) \}}{\{ 2 - \log(1+\exp(1)) + \log(2) \}} \\
& + \{ -4 - \log(1+\exp(-2)) + \log(2) \} \Bigg] = 0 \qquad (4.30)
\end{aligned}
$$

が成り立つ．式 (4.30) の解は $\alpha_3 = 0$ ただ一つであり，よって，$\alpha_1 = \alpha_2 = \alpha_3 = 0$ である．式 (4.22) は，$\alpha_1 = \alpha_2 = \alpha_3 = 0$ のときのみ成り立つ．したがって，$\log(\boldsymbol{f}(\boldsymbol{z}_1))$, $\log(\boldsymbol{f}(\boldsymbol{z}_2))$ と $\log(\boldsymbol{f}(\boldsymbol{z}_3))$ は線形独立であり，すなわち $d+1=2$ にもかかわらず出力ベクトルは 3 つ線

形独立となる．よって，log-sigsoftmax の値域は $(d+1)$ 次元以上の線形独立の空間であり，log-sigsoftmax の値域は $(d+1)$ 次元のベクトル空間の部分集合ではない．これは，次の命題，「Log-sigsoftmax のすべての出力空間 $\{\log(\boldsymbol{f}(\boldsymbol{z}))|\boldsymbol{z} \in S\}$ は，$(d+1)$ 次元のベクトル空間の部分空間である」に矛盾しており，よって，log-sigsoftmax の値域は $(d+1)$ 次元のベクトル空間の部分空間とは限らない． □

定理 4.3 は sigsoftmax が softmax のボトルネックを解消しうることを示している．すなわち，たとえ真の確率の対数のベクトルが $(d+1)$ 個より多くの線形独立なベクトルを持つとしても sigsoftmax ベースのモデルは真の確率に一致しうる．

定理 4.3 の結果は，少なくとも一つは softmax bottleneck を解消する可能性がある入力を示したが，より一般に成り立つ結果として，次の定理が成り立つ．

**定理 4.4.** $S \subseteq \mathbb{R}^M$ を 1 次元のアフィン空間とし， $\boldsymbol{z} \in S$ を log-sigsoftmax の入力とする．このとき，log-sigsoftmax の出力ベクトルが張る空間 $\mathrm{span}\{\log(\boldsymbol{f}(\boldsymbol{z}))|\boldsymbol{z} \in S\}$ が $\mathbb{R}^M$ となるような $S$ が存在する．

**証明.** まず，$\tilde{\boldsymbol{f}}$ と $g$ を

$$\begin{aligned}\log(\tilde{\boldsymbol{f}}(\boldsymbol{z})) &= 2\boldsymbol{z} - \log(\mathbf{1} + \exp(\boldsymbol{z})) \\ &= \begin{bmatrix} g(z_1) \\ \vdots \\ g(z_M) \end{bmatrix}\end{aligned}$$

と定義する．$\tilde{\boldsymbol{f}}$ を使うと，log-sigsoftmax は

$$\log(\boldsymbol{f}(\boldsymbol{z})) = \log(\tilde{\boldsymbol{f}}(\boldsymbol{z})) + c'\boldsymbol{z}\mathbf{1} \tag{4.31}$$

である．一方，1 次元のアフィン空間 $S$ を

$$S = \{\boldsymbol{z} = x\boldsymbol{a} + b\mathbf{1}|x \in \mathbb{R}\}$$

とする．ただし，$\boldsymbol{a} \in \mathbb{R}^M$ は各要素が排反，つまり，$i \neq j$ に対して $a_i \neq a_j$ とする．また，$b \in \mathbb{R}$ は $b \neq 0$ で $g^{(n)}(b) \neq 0$ とする．ただし，$g^{(n)}$ は $g$ の $n$ 階の微分である．$g(z)$ には softplus が含まれており，softplus の微分が $C^\infty$ 級の関数である sigmoid であることから $g$ は無限回微分可能である．そして，$n \geq 3$ で $n$ が奇数のとき $g^{(n)}(0) = 0$ となる．このとき，

ロンスキアン $W(x)$ は

$$
\begin{aligned}
W(x) &= \begin{vmatrix} g(a_1x+b) & g(a_2x+b) & \cdots & g(a_Mx+b) \\ \frac{d}{dx}g(a_1x+b) & \ddots & & \vdots \\ \vdots & & & \vdots \\ \frac{d^{M-1}}{d^{M-1}x}g(a_1x+b) & \cdots & \cdots & \frac{d^{M-1}}{d^{M-1}x}g(a_Mx+b) \end{vmatrix} \\
&= \begin{vmatrix} g(a_1x+b) & g(a_2x+b) & \cdots & g(a_Mx+b) \\ a_1g'(a_1x+b) & \ddots & & a_Mg'(a_Mx+b) \\ \vdots & & & \vdots \\ a_1^{M-1}g^{(M-1)}(a_1x+b) & \cdots & \cdots & a_M^{M-1}g^{(M-1)}(a_Mx+b) \end{vmatrix}
\end{aligned}
$$

となり，$x=0$ のとき

$$
\begin{aligned}
W(0) &= \begin{vmatrix} g(b) & g(b) & \cdots & g(b) \\ a_1g'(b) & \ddots & & a_Mg'(b) \\ \vdots & & & \vdots \\ a_1^{M-1}g^{(M-1)}(b) & \cdots & \cdots & a_M^{M-1}g^{(M-1)}(b) \end{vmatrix} \\
&= \begin{vmatrix} g(b) & 0 & \cdots & 0 \\ 0 & g'(b) & 0 & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & g^{(M-1)}(b) \end{vmatrix} \begin{vmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_M \\ \vdots & \ddots & \cdots & \vdots \\ a_1^{M-1} & \ddots & \cdots & a_M^{M-1} \end{vmatrix} \\
&= \prod_{j=0}^{M-1} g^{(j)}(b) \prod_{1\le i<j\le M} (a_j-a_i) \neq 0
\end{aligned}
$$

が成り立つ．ただし，行列積の行列式が行列式の積となること，対角行列の行列式が対角成分の積であること，ヴァンデルモンドの行列式，そして $g^{(j)}(b)\neq 0$，$a_i\neq a_j$ を使った．$W(0)\neq 0$ より，ロンスキアンの性質から，$g(a_1x+b),\ldots,g(a_Mx+b)$ は $\mathbb{R}$ 上の線形独立な関数である．よって，$\log\tilde{\boldsymbol{f}}(\boldsymbol{z})$ の張る空間，$\mathrm{span}\left\{\log\tilde{\boldsymbol{f}}(\boldsymbol{z})|\boldsymbol{z}\in S\right\}$ を考えると，

$$
\begin{aligned}
\mathrm{span}\left\{\log\tilde{\boldsymbol{f}}(\boldsymbol{z})|\boldsymbol{z}\in S\right\} &= \mathrm{span}\left\{\log\tilde{\boldsymbol{f}}(x\boldsymbol{a}+b\boldsymbol{1})|x\in\mathbb{R}\right\} \\
&= \mathrm{span}\left\{(g(a_ix+b))_{i=1}^{M}|x\in\mathbb{R}\right\}
\end{aligned}
$$

は，$g(a_ix+b)$ の線形独立性から

$$
\mathrm{span}\left\{\log\tilde{\boldsymbol{f}}(x\boldsymbol{a}+b\boldsymbol{1})|x\in\mathbb{R}\right\} = \mathbb{R}^M \tag{4.32}
$$

となる．また，$\boldsymbol{p}\notin\mathrm{span}\{\log(\boldsymbol{f}(\boldsymbol{z}))|\boldsymbol{z}\in S\}$ なる $\boldsymbol{p}$ が存在すると仮定すると，式 (4.32) より，

$$
\boldsymbol{p}\in\mathbb{R}^M = \mathrm{span}\left\{\log\tilde{\boldsymbol{f}}(x\boldsymbol{a}+b\boldsymbol{1})|x\in\mathbb{R}\right\} \tag{4.33}
$$

である．このとき，式 (4.31) より $\boldsymbol{p} \notin \mathrm{span}\left\{\log\tilde{\boldsymbol{f}}(\boldsymbol{z}) + c'(\boldsymbol{z})\mathbf{1} | \boldsymbol{z} \in S\right\}$ であり，かつ $\boldsymbol{p} \in \mathrm{span}\left\{\log\tilde{\boldsymbol{f}}(\boldsymbol{z}) | \boldsymbol{z} \in S\right\}$ より，$\boldsymbol{p} \in \mathrm{span}\{\mathbf{1}\}$ である．しかし，

$$\begin{aligned}\log\boldsymbol{f}(b\mathbf{1}) &= \log\tilde{\boldsymbol{f}}(b\mathbf{1}) + c'(b\mathbf{1})\mathbf{1} \\ &= \{g(b) + c'(b\mathbf{1})\}\mathbf{1} \neq \mathbf{0}\end{aligned} \tag{4.34}$$

となる $b$ をとることが可能であり，$b\mathbf{1} \in S$ より，$\boldsymbol{p} \notin \mathrm{span}\{\log\boldsymbol{f}(\boldsymbol{z}) | \boldsymbol{z} \in S\}$ に矛盾する．よって，このような $\boldsymbol{p}$ は存在せず，$\mathrm{span}\{\log\boldsymbol{f}(\boldsymbol{z}) | \boldsymbol{z} \in S\} = \mathbb{R}^M$ であることが示された．　□

定理 4.4 は，一般に，softmax 関数に入力される前の層がバイアス項を持っているため，$d = 1$ でバイアス項がある場合，つまり $\boldsymbol{z} = h\boldsymbol{w} + b\mathbf{1}$ であるときには，softmax bottleneck を解消する可能性があることを示している．さらに条件を満たすアフィン空間が含まれるような 2 次元以上の線形空間 $S$ があることから[*3]，$d \geq 2$ であれば，$M$ 次元の空間に log-sigsoftmax の出力は存在し，softmax bottleneck を解消できる可能性があることを示す．

しかしながら，softmax と sigsoftmax は共に非線形関数であるため，前述の通り値域の含まれるベクトル空間の次元のみを基準に表現能力を比較するのは難しい．そこで直接 2 つの値域を比較すると，次の定理が成り立つ．

**定理 4.5.** $\boldsymbol{z} \in S$ を sigsoftmax $\boldsymbol{f}(\cdot)$ と softmax $\boldsymbol{f}_s(\cdot)$ の入力とする．もし $S$ が $d$ 次元ベクトル空間で $\mathbf{1} \in S$ であるとすると, softmax の値域は sigsoftmax の値域の閉包 $\overline{\{\boldsymbol{f}(\boldsymbol{z}) | \boldsymbol{z} \in S\}}$ に含まれる．すなわち

$$\{\boldsymbol{f}_s(\boldsymbol{z}) | \boldsymbol{z} \in S\} \subseteq \overline{\{\boldsymbol{f}(\boldsymbol{z}) | \boldsymbol{z} \in S\}}. \tag{4.35}$$

**証明.** $\mathbf{1} \in S$ であるとすると，$S$ は

$$S = \left\{ \sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)} + k'^{(d)}\mathbf{1} \,\middle|\, k'^{(l)} \in R \right\}$$

とかける．ここで $\boldsymbol{u}'^{(l)}$ $(l = 1, \ldots, d-1)$ と $\mathbf{1}$ は線形独立なベクトルである．また，$S$ の任意の要素は $\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)} + k'^{(d)}\mathbf{1}$ となり，よって $\boldsymbol{z} = \sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)} + k'^{(d)}\mathbf{1}$ である．Softmax の入力として $\boldsymbol{z} = \sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)} + k'^{(d)}\mathbf{1}$ を代入すると

$$[\boldsymbol{f}_s(\boldsymbol{z})]_i = \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_i + k'^{(d)}\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_m + k'^{(d)}\right)} = \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_i\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_m\right)} \tag{4.36}$$

[*3] $T_{ij} := \{\boldsymbol{z} \in \mathbb{R}^M | z_i = z_j\}$ のような部分空間か，もしくはそれに含まれる部分空間である場合は条件を満たすアフィン空間が含まれない．

であり，よって softmax の値域は

$$\left\{ \boldsymbol{f}_s\left(\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)} + k'^{(d)}\mathbf{1}\right) \middle| k'^{(l)} \in \mathbb{R} \right\} = \left\{ \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_i\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_m\right)} \middle| k'^{(l)} \in \mathbb{R} \right\} \tag{4.37}$$

である．一方，$\boldsymbol{z} = \sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)} + k'^{(d)}\mathbf{1}$ を sigsoftmax に代入すると，sigsoftmax の出力は

$$[\boldsymbol{f}(\boldsymbol{z})]_i = \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_i\right)\mathrm{sigm}\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_i + k'^{(d)}\right)}{\sum_m \exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_m\right)\mathrm{sigm}\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_m + k'^{(d)}\right)} \tag{4.38}$$

となる．ここで，$k'^{(l)}$ を $l = 1, \ldots, d-1$ について固定し，$k'^{(d)} \to +\infty$ とすると，$v$ が固定されているとき $\lim_{k\to+\infty}\mathrm{sigm}(v+k)=1$ であるため，

$$\begin{aligned}\lim_{k'^{(d)}\to+\infty} &\frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_i\right)\mathrm{sigm}\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_i + k'^{(d)}\right)}{\sum_m \exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_m\right)\mathrm{sigm}\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_m + k'^{(d)}\right)} \\ &= \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_i\right)}{\sum_m \exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_m\right)}\end{aligned} \tag{4.39}$$

である．式 (4.39) より sigsoftmax の値域は，

$$\left\{ \frac{\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_i\right)}{\sum_{m=1}^{M}\exp\left(\left[\sum_{l=1}^{d-1} k'^{(l)}\boldsymbol{u}'^{(l)}\right]_m\right)} \middle| k'^{(l)} \in \mathbb{R} \right\} \subseteq \overline{\{\boldsymbol{f}(\boldsymbol{z})|\boldsymbol{z} \in S\}} \tag{4.40}$$

となる．式 (4.37) と (4.40) より，sigsoftmax の値域の閉包に softmax の値域が含まれる．よって，$\{\boldsymbol{f}_s(\boldsymbol{z})|\boldsymbol{z} \in S\} \subseteq \{\boldsymbol{f}(\boldsymbol{z})|\bar{\boldsymbol{z}} \in S\}$ が成り立つ． □

定理 4.5 は，もし $\mathbf{1} \in S$ であれば，sigsoftmax が softmax の出力を任意の精度で近似できることを示している．$\mathbf{1} \in S$ という仮定は，クラス分類ではすべてのクラスに対して同じ確率，すなわち $p_{\boldsymbol{\theta}}(y_i|\boldsymbol{x}) = 1/M$ for all $i$，を出力できることを示している．たとえ $\mathbf{1} \notin S$ であっても，スカラのパラメータ $b$ を導入して，

$$[\boldsymbol{f}(\boldsymbol{z} + b\mathbf{1})]_i = \frac{\exp(z_i)\mathrm{sigm}(z_i + b)}{\sum_m \exp(z_m)\mathrm{sigm}(z_m + b)}$$

とすると同様の性質を持つ．Softmax が真の確率に一致するときは $b$ が十分に大きくなれば softmax を sigsoftmax で近似できる．以上より，sigsoftmax は softmax のボトルネックを解消し，かつ表現力が大きくなることを示した．

Sigsoftmax は 4.2.2 項で述べた性質を満たす．

**定理 4.6.** Sigsoftmax は次の性質を持つ.

1. $\log(\boldsymbol{g}(\boldsymbol{z}))$ の非線形性: $\log(\boldsymbol{g}(\boldsymbol{z})) = 2\boldsymbol{z} - \log(\boldsymbol{1} + \exp(\boldsymbol{z}))$
2. 数値的な安定性:

$$\frac{\partial \log\left[\boldsymbol{f}(\boldsymbol{z})\right]_i}{\partial z_j} = \begin{cases} (1 - \left[\boldsymbol{f}(\boldsymbol{z})\right]_j)(2 - \mathrm{sigm}(z_j)), & i = j \\ -\left[\boldsymbol{f}(\boldsymbol{z})\right]_j (2 - \mathrm{sigm}(z_j)), & i \neq j \end{cases}$$

3. 非負性: $[\boldsymbol{g}(\boldsymbol{z})]_i = \exp(z_i)\mathrm{sigm}(z_i) \geq 0$.
4. 単調増加性: $z_1 \leq z_2 \Rightarrow \exp(z_1)\mathrm{sigm}(z_1) \leq \exp(z_2)\mathrm{sigm}(z_2)$

**証明.** まず,

$$[\boldsymbol{g}(\boldsymbol{z})]_i = \exp(z_i)\mathrm{sigm}(z_i) = \frac{\exp(z_i)}{1 + \exp(-z_i)} = \frac{\exp(2z_i)}{1 + \exp(z_i)}$$

より, $\log(\boldsymbol{g}(\boldsymbol{z})) = 2\boldsymbol{z} - \log(\boldsymbol{1} + \exp(\boldsymbol{z}))$ が成り立つ. $\log(1 + \exp(z))$ は softplus と呼ばれる非線形関数であるため, $\log(\boldsymbol{g}(\boldsymbol{z}))$ は非線形関数である. 次に, $\mathrm{d}(\exp(z)(/\mathrm{d}z = \exp(z)$ と $\mathrm{d}(\mathrm{sigm}(z))/\mathrm{d}z = \mathrm{sigm}(z)(1 - \mathrm{sigm}(z))$ が成り立つため,

$$\begin{aligned} \frac{\partial \log\left[\boldsymbol{f}(\boldsymbol{z})\right]_i}{\partial z_j} &= \frac{1}{\left[\boldsymbol{f}(\boldsymbol{z})\right]_i} \frac{\partial \left[\boldsymbol{f}(\boldsymbol{z})\right]_i}{\partial z_j} \\ &= \frac{1}{\left[\boldsymbol{f}(\boldsymbol{z})\right]_i} \Bigg\{ \frac{1}{\sum_{m=1}^{M} \exp(z_m)\mathrm{sigm}(z_m)} \frac{\partial \exp(z_i)\mathrm{sigm}(z_i)}{\partial z_j} \\ &\qquad - \frac{\left[\boldsymbol{f}(\boldsymbol{z})\right]_i}{\sum_{m=1}^{M} \exp(z_m)\mathrm{sigm}(z_m)} \frac{\partial \sum_{m=1}^{M} \exp(z_m)\mathrm{sigm}(z_m)}{\partial z_j} \Bigg\} \\ &= \begin{cases} (1 - \left[\boldsymbol{f}(\boldsymbol{z})\right]_j)(2 - \mathrm{sigm}(z_j)), & i = j \\ -\left[\boldsymbol{f}(\boldsymbol{z})\right]_j (2 - \mathrm{sigm}(z_j)), & i \neq j \end{cases} \end{aligned} \tag{4.41}$$

さらに, $\exp(z) \geq 0$ と $\mathrm{sigm}(z) \geq 0$ より, $\exp(z)\mathrm{sigm}(z) \geq 0$ が成り立つ. 最後に, $\exp(z)\mathrm{sigm}(z)$ の微分はすべての $z$ に対して

$$\frac{\mathrm{d}\exp(z)\mathrm{sigm}(z)}{\mathrm{d}z} = \frac{\exp(z) + 2}{(1 + \exp(-z))^2} \geq 0$$

であるため, $\exp(z)\mathrm{sigm}(z)$ は単調増加である. □

### (b) Mixture of sigsoftmax

Sigsoftmax は softmax に代わる関数なので softmax と同様に混合し，mixture of sigsoftmax (MoSS) とできる．これは

$$P_{\boldsymbol{\theta}}(y_i|x) = \sum_{k=1}^{K} \pi(x,k) \frac{\exp([\boldsymbol{W}\boldsymbol{h}(x,k)]_i)\mathrm{sigm}([\boldsymbol{W}\boldsymbol{h}(x,k)]_i)}{\sum_m \exp([\boldsymbol{W}\boldsymbol{h}(x,k)]_m)\mathrm{sigm}([\boldsymbol{W}\boldsymbol{h}(x,k)]_m)} \tag{4.42}$$

であり，$\pi(x,k)$ も同様に sigsoftmax で

$$\pi(x,k) = \frac{\exp(\boldsymbol{w}_{\pi,k}^{\mathsf{T}}\boldsymbol{h}'(x))\mathrm{sigm}(\boldsymbol{w}_{\pi,k}^{\mathsf{T}}\boldsymbol{h}'(x))}{\sum_{k'=1}^{K} \exp(\boldsymbol{w}_{\pi,k'}^{\mathsf{T}}\boldsymbol{h}'(x))\mathrm{sigm}(\boldsymbol{w}_{\pi,k'}^{\mathsf{T}}\boldsymbol{h}'(x))} \tag{4.43}$$

とする．実験ではこの他に，mixture of softmax の softmax を sigmoid に置き換えた mixture of sigmoid，ReLU に置き換えた mixture of ReLU，そして softplus に置き換えた mixture of softplus を用意し，それぞれ性能を評価する．

### (c) 実装上の工夫について

定理 4.5 の証明にあったように，softmax は分母と分子に同じ値が入ると相殺するという特徴がある．また $\boldsymbol{z}$ の値が大きいと $\sum_m \exp(z_m)$ の値が非常に大きくなり，計算が不安定になりうる．以上のことから，log-softmax の実装はフレームワークによっては

$$[\boldsymbol{f}_s(\boldsymbol{z})]_i = \frac{\exp(z_i - \min_i z_i)}{\sum_m \exp(z_m - \min_i z_i)} \tag{4.44}$$

$$\log(\boldsymbol{f}_s(\boldsymbol{z})) = \boldsymbol{z} - \min_i z_i \mathbf{1} - \log \sum_m \exp(z_m - \min_i z_i)\mathbf{1} \tag{4.45}$$

としている．Sigsoftmax においても数値計算上の安定化のため，これをもとにして

$$[\boldsymbol{f}(\boldsymbol{z})]_i = \frac{\mathrm{sigm}(z_i)\exp(z_i - \min_i z_i)}{\sum_m \mathrm{sigm}(z_m)\exp(z_m - \min_i z_i)} \tag{4.46}$$

$$\log(\boldsymbol{f}(\boldsymbol{z})) = 2\boldsymbol{z} - \min_i z_i \mathbf{1} - \log(\mathbf{1} + \exp(\boldsymbol{z})) - \log(\sum_m \exp(z_m - \min_i z_i)\mathrm{sigm}(z_m))\mathbf{1} \tag{4.47}$$

とした．ここで，sigsoftmax は任意の定数 $c$ に対して，

$$[\boldsymbol{f}(\boldsymbol{z})]_i = \frac{\exp(z_i)\mathrm{sigm}(z_i)}{\sum_m \exp(z_m)\mathrm{sigm}(z_m)} = \frac{\exp(z_i - c)\mathrm{sigm}(z_i)}{\sum_m \exp(z_m - c)\mathrm{sigm}(z_m)} \tag{4.48}$$

であることを利用した．

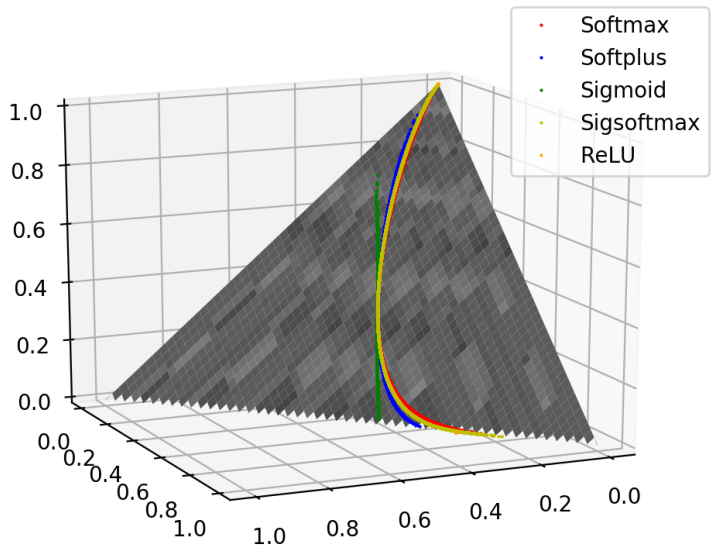



Fig. 4.2 Image of output. The number of possible labels is set to three, and thus, a set of possible probabilities is a two-dimensional simplex that is shown by gray plane.

## 4.3 人工的な入力を使った softmax bottleneck の確認

Softmax bottleneck を確認するために，人工データをつかって確認した．まず，とりうる離散値が 3 つである $M = 3$ とし，1 次元のベクトル空間（直線）を入力空間とした．3 次元空間上の 1 次元の入力（ベクトルのスカラ倍）を乱数生成し，これを softmax，sigmoid からなる関数，softplus からなる関数, ReLU からなる関数, sigmoid にそれぞれ入力し，その出力と対数出力を可視化した．結果を Fig. 4.2 と 4.3 に示す．

Fig. 4.2 の灰色の三角形は 3 次元の離散分布がとりうる範囲を示している．3 次元の離散分布のとりうる範囲は各要素が 0 以上 1 以下で，和が 1 となるため，2 次元の単体となり三角形となる．この上にそれぞれの出力は曲線を描いている．対数をとった Fig. 4.3 を見ると，確かに softmax はある 2 次元の平面上に曲がった曲線を描いている．一方，その他の出力はこの平面ではなく，3 次元空間上で曲がった曲線である．入力を学習ではなく乱数で生成したため，表現能力の比較をすることは困難であるが，sigmoid からなる出力関数の出力の範囲は今回生成した入力に対しては他と比較して小さな範囲にとどまっている，これは sigmoid の出力が [0,1] の範囲に限定されるためと考えられる．

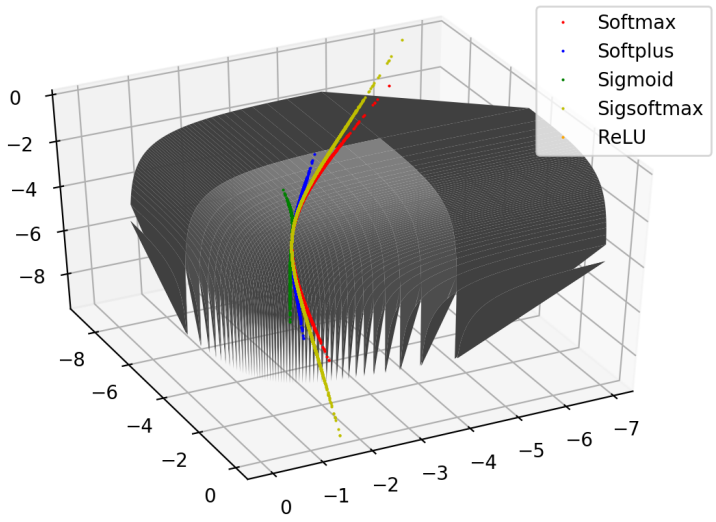



Fig. 4.3　Image of log-output. Logarithm of possible probabilities become a curved plane. The image of log-softmax is a curved line on a flat plane, and images of other functions are curved lines in a curved plane.

## 4.4　単語レベルの言語モデリングの実験による評価

### 4.4.1　実験条件

Softmax bottleneck を解消する出力関数の性能を評価するために，言語モデルの実験を行った．Sigsoftmax と softmax，$g$ を ReLU とした関数と sigmoid，softplus とした関数で比較し，これらの混合モデルである MoS と MoR（mixture of ReLU），MoSig（mixture of sigmoid），MoSp（mixture of softplus），MoSS も比較する．使用したデータセットは，先行研究 [5, 21, 84] にならい，Penn Treebank dataset (PTB) [11, 76] と WikiText-2 dataset (WT2) [77] とした．PTB と WT2 の詳細は 3.5.1 項に示したが，それぞれの語彙数 $M$ は PTB が 1 万単語であり，WT2 は 33,278 である．WT2 は語彙数，単語数ともに PTB より大きなデータセットであるため，WT2 のモデリングは PTB より大きな表現能力を必要とする．

結果に表現力以外の要素（最適化性能など）ができるだけ影響をなくすため，実験条件は本研究を報告した [82] の時点で最も高い精度を達成した条件とするため，先行研究 [5, 21, 84] の実験条件をそのまま使用した．これに従い，使用するモデルは 3 層の long short-term

memory（LSTM）とした．また，より公平な比較を行うため，[5, 21, 84] の実験コード[*4] を使用し，ニューラルネットのユニット数，最適化手法などの実験条件をすべて同じとした．ただし，学習の反復回数にあたるエポック数のみ，検証データに対する損失（validation loss）を収束させるために 2 倍とした．学習ののち，学習済みモデルを再度 1 から再学習させる finetune と dynamic evaluation を適用してそれぞれのステップでの結果を比較した．Dynamic evaluation はニューラルネットワークを基にした言語モデルにおいて，評価のときに過去の系列を用いて適応的にモデルを更新しながら評価する方法である．これを複数回実験しその検証データと test データに対する perplexity の平均と標準偏差を求めた．以下，詳細な実験条件である．

### (a) 活性化関数の実験条件

Softmax と sigsoftmax，ReLU，sigmoid，そして softplus を用いた関数を比較するため，3 層の LSTM を先行研究 [84] にならって学習させた．[5, 84] で用意されている実験コードを使用した．[*5] [84] は彼らのコードの中でいくつかのパラメータを，原著論文より良い結果を得るためにチューニングしている．公平な評価をするために，[84] のコードを (i) 各関数との入れ替え，(ii) 複数の乱数シードの使用，そして (iii) エポック数を 2 倍にするという変更のみ加えた．ReLU と softplus は数値的に不安定となりうるため，

$$[\boldsymbol{f}(\boldsymbol{z})]_i = \frac{g(z_i) + \varepsilon}{\sum_{m=1}^{M} g(z_m) + \varepsilon} \tag{4.49}$$

として，分母分子に $\varepsilon = 10^{-8}$ を加えた．

実験条件は [84] と同じにした．LSTM のユニット数は 1150 とし，埋め込みサイズ $d$ は 400 とした．重み行列は一様分布 $U(-0.1, 0.1)$ を使って最終層は初期化し，そのほかは一様分布 $U(-1/\sqrt{H}, 1/\sqrt{H})$ によって初期化した．ただし，$H$ は隠れユニットの数である．

すべてのモデルは a non-monotonically triggered variant of averaged SGD (NT-ASGD) [84] によって学習し，学習率は 30 とした．また，gradient clipping を用い，その閾値を 0.25 とした．Dropout connect [84] を，word vectors と LSTM 間の出力，最後の LSTM 層の出力，そして，最終層に使用し，その dropout rate はそれぞれ (0.4, 0.25, 0.4, 0.1) を PTB に使用し，(0.65, 0.2, 0.4, 0.1) を WT2 に使用した．バッチサイズは PTB に 20 を，WT2 に 80 を使用した．学習のエポック数は PTB に対し 1000 とし WT2 に対して 500 とした．学習後，finetune として ASGD を停止基準を満たすまで行った．学習には random

[*4] https://github.com/salesforce/awd-lstm-lm(ただし，[84] のコードは良い結果を得るために，その著者によって元の論文からさらにパラメータが調整されている．);
https://github.com/benkrause/dynamic-evaluation; https://github.com/zihangdai/mos

[*5] https://github.com/salesforce/awd-lstm-lm;
https://github.com/benkrause/dynamic-evaluation

backpropagation through time (BPTT) を使い，その長さは $\mathcal{N}(70, 5)$ を 0.95 の確率で使用し，残り 0.05 の確率で $\mathcal{N}(35, 5)$ を使用した．Activation regularization (AR) と temporal activation regularization (TAR) を RNN の最終層に使用した．それぞれの係数は 2 と 1 とした．

Finetune のあとに，dynamic evaluation [5] を使用した．このステップにおいて， [5] によって用意されたハイパーパラメータチューニングを行った．学習率 $\eta$ は $[3 \times 10^{-5}, 4 \times 10^{-5}, 5 \times 10^{-5}, 6 \times 10^{-5}, 7 \times 10^{-5}, 1 \times 10^{-4}]$ から選び，decay rate は $[1 \times 10^{-3}, 2 \times 10^{-3}, 3 \times 10^{-3}, 5 \times 10^{-3}]$ からチューニングした．$\epsilon$ は 0.001 とし，バッチサイズは 100 とした．

以上の実験条件は [5, 84] の時点で最も高い精度を達成したものである．上記の手順を 10 回行い，validation perplexity の最小の値の平均と test perplexity の平均を計算した．

### (b) 混合モデルに対する実験条件

3 層の LSTM を既存研究 [21] に従って学習させた．こちらも [21][*6]によって用意された実験コードを用いて，(i) MoS から各混合モデルへの交換, (ii) 複数乱数シードの変更を行った．モデルを学習させたあと，finetune させて dynamic evaluation を行った．

上記の通り，実験条件は [21] を用いた．この実験において LSTM 層のユニット数は PTB では [960, 960, 620]，埋め込み層のサイズは 280 とした．WT2 では LSTM のユニット数は [1150, 1150, 650] とし，埋め込み層のサイズは 300 とした．混合数はどちらのデータセットにおいても 15 とした．重みの初期化は活性化関数の比較と同じである．どちらのデータセットにおいても word level variational dropout を dropout rate0.10 で使用し，recurrent weight dropout を dropout rate0.5 で使用，そして context vector level variational dropout を dropout rate0.30 で使用した．さらに，PTB では embedding level variational dropout を dropout rate0.55 で使用し，hidden level variational dropout を dropout rate0.225 で使用した．WT2 では embedding level variational drop out を dropout rate0.40 で使用し hidden level variational drop out を dropout rate0.225 で使用した．最適化方法は活性化関数の比較と同様である．

Dynamic evaluation ステップでは学習率を 0.002 とし，バッチサイズは 100 を両方のデータセットに使用した．PTB では $\epsilon$ を 0.001 とし decay rate を 0.075 とした．WT2 では $\epsilon$ を 0.002 とし decay rate を 0.02 とした．

以上の実験条件は [21] の時点で最も高い精度を達成したものである．上記の手順を 5 回行い， validation の最小の平均と test perplexity の平均を計算した．

[*6] `https://github.com/zihangdai/mos`

Table 4.1 Perplexities of language models on PTB. Valid means validation, and w/o finetune means the results before finetune. dynamic eval. means dynamic evaluation [5].

| | softmax | $g$:ReLU | $g$:sigmoid | $g$:softplus | sigsoftmax |
|---|---|---|---|---|---|
| Valid. w/o finetune | 61.1±0.4 | (1.85±0.2)×$10^3$ | **60.7±0.2** | 61.0±0.2 | 61.0±0.2 |
| Test w/o finetune | 58.8±0.4 | (1.54±0.2)×$10^3$ | 58.5±0.2 | 58.5±0.1 | **58.4±0.2** |
| Valid. | 59.2±0.4 | (1.51±0.1)×$10^3$ | **58.7±0.4** | 59.3±0.3 | 59.2±0.4 |
| Test | 57.0±0.6 | (1.24±0.08)×$10^3$ | **56.4±0.2** | 56.6±0.2 | 56.6±0.4 |
| Valid.+dynamic eval. | 51.2±0.5 | (4.91±5)×$10^3$ | **49.2±0.4** | 49.5±0.2 | 49.7±0.5 |
| Test +dynamic eval. | 50.5±0.5 | (2.78±8)×$10^5$ | **48.9±0.3** | 49.1±0.1 | 49.2±0.4 |

| | MoS | MoR | MoSig | MoSp | MoSS |
|---|---|---|---|---|---|
| Valid. w/o finetune | **58.4±0.2** | 78.4±0.5 | 64.3±0.2 | 63.3±0.2 | **58.4±0.3** |
| Test w/o finetune | 56.3±0.3 | 71.6±0.7 | 61.6±0.2 | 60.9±0.2 | **56.2±0.2** |
| Valid. | **56.8±0.2** | 71.9±0.4 | 63.3±0.2 | 62.1±0.3 | 56.9±0.1 |
| Test | 54.7±0.08 | 66.2±0.4 | 60.3±0.3 | 59.4±0.2 | **54.6±0.2** |
| Valid.+dynamic eval. | 48.6±0.2 | 57.6±2 | 50.5±0.4 | 49.1±0.2 | **48.3±0.1** |
| Test +dynamic eval. | 55.5±0.3 | 50.0±0.4 | 48.0±0.1 | 48.8±0.2 | **47.7±0.07** |

### 4.4.2 実験結果

PTB と WT2 に対する validation perplexity と test perplexity を Table 4.1 と 4.2 にまとめる．Table 4.1 より，PTB では sigmoid を使用した関数の perplexity が最も小さい．しかしながら，Table 4.2 では，WT2 のデータに対して sigmoid による関数の perplexity は softmax より大きい．これは sigmoid の最大値が 1 に制限されるという性質により，大きなデータに対して sigmoid を使ったモデルの表現能力が不足するためと考えられる．Softplus は dynamic evaluation 後の性能は高いが，学習後，finetune 後の WT2 のデータでの perplexity が softmax より大きい．これは softplus が softmax bottleneck を解消できる表現能力の高さにより dynamic evaluation で過去の系列により適応した結果，perplexity が減少したと考えられる．一方で，sigsoftmax は学習後，finetune 後，dynamic evaluation のどのステップにおいても softmax 以上か同等の性能を示し，特に dynamic evaluation では PTB，WT2 共に softmax より小さな perplexity となっている．さらに，混合モデル間では sigsoftmax を

Table 4.2 Perplexities of language models on WT2. Valid means validation, and w/o finetune means the results before finetune. dynamic eval. means dynamic evaluation [5].

| | softmax | $g$:ReLU | $g$:sigmoid | $g$:softplus | sigsoftmax |
|---|---|---|---|---|---|
| Valid. w/o finetune | 68.0±0.2 | (8.74±0.7)×$10^2$ | 72.8±0.3 | 71.0 ±0.3 | **67.8±0.1** |
| Test w/o finetune | 65.2±0.2 | (7.97±0.7)×$10^2$ | 69.7±0.3 | 67.6 ±0.2 | **65.0±0.2** |
| Valid. | **67.4±0.2** | (6.48±0.1)×$10^2$ | 70.8±0.1 | 68.8±0.3 | **67.4±0.2** |
| Test | 64.7±0.2 | (5.93±0.08)×$10^2$ | 68.2±0.1 | 65.8±0.2 | **64.2±0.1** |
| Valid.+dynamic eval. | 45.3±0.2 | (1.79±0.8)×$10^3$ | 45.7±0.1 | **44.9**±0.2 | **44.9±0.1** |
| Test +dynamic eval. | 43.3±0.1 | (2.30±2)×$10^4$ | 43.5±0.1 | **42.6±0.2** | 42.9±0.1 |

| | MoS | MoR | MoSig | MoSp | MoSS |
|---|---|---|---|---|---|
| Valid. w/o finetune | 65.9±0.5 | 92.5±1 | 73.6±0.5 | 70.9±0.1 | **65.1±0.2** |
| Test w/o finetune | 63.3±0.4 | 87.2±1 | 70.0±0.3 | 67.4±0.1 | **62.5±0.3** |
| Valid. | 64.0±0.3 | 88.9±1 | 69.5±0.2 | 68.5±0.2 | **63.7±0.3** |
| Test | 61.4±0.4 | 83.9±1 | 66.4±0.1 | 65.3±0.1 | **61.1±0.3** |
| Valid.+dynamic eval. | 42.5±0.1 | 58.8±2 | 44.33±0.05 | 43.2±0.07 | **42.1±0.2** |
| Test +dynamic eval. | 40.8±0.03 | 55.2±2 | 42.10±0.08 | 41.12±0.08 | **40.3±0.2** |

混合した MoSS が他を混合したモデルより perplexity が小さい．本論文で使用した実験条件は [21, 84] において，softmax と MoS に最適化されていたものを使用したにもかかわらず，softmax を超える性能に達したため sigsoftmax は softmax より性能が高いといえる．

### 4.4.3 線形独立性の評価

Log-softmax の値域が $(d+1)$ 次元の部分空間の部分集合であること，log-sigsoftmax などのその対数に非線形性をもつ関数の値域が $(d+1)$ 次元以上の空間に含まれることを示すため，各関数の出力に対数をとったベクトルの線形独立性を評価する．まず，得られたモデルに対して test データを入力し，各時刻の出力の対数 $\log(P_{\boldsymbol{\theta}}(\boldsymbol{y}_t|\boldsymbol{x}_t))$，例えば，log-softmax，を計算した．次に，これらのベクトルをまとめた行列 $\hat{\boldsymbol{A}} = [\log(P_{\boldsymbol{\theta}}(\boldsymbol{y}_1|\boldsymbol{x}_1)), \ldots, \log(P_{\boldsymbol{\theta}}(\boldsymbol{y}_T|\boldsymbol{x}_T))] \in \mathbb{R}^{M\times T}$ を作成した．ただし，$T$ は test データの単語数である．PTB では $M$ と $T$ がそれぞれ 10,000 と 82,430 であり，WT2 では 33,278 と 245,570 である．

最後に，$\hat{\boldsymbol{A}}$ のランクを調べた．このランクが線形独立なベクトルの本数に対応する．ランク計算には数値誤差を含むため [21, 85] で使用されている基準を使用した．得られたランク

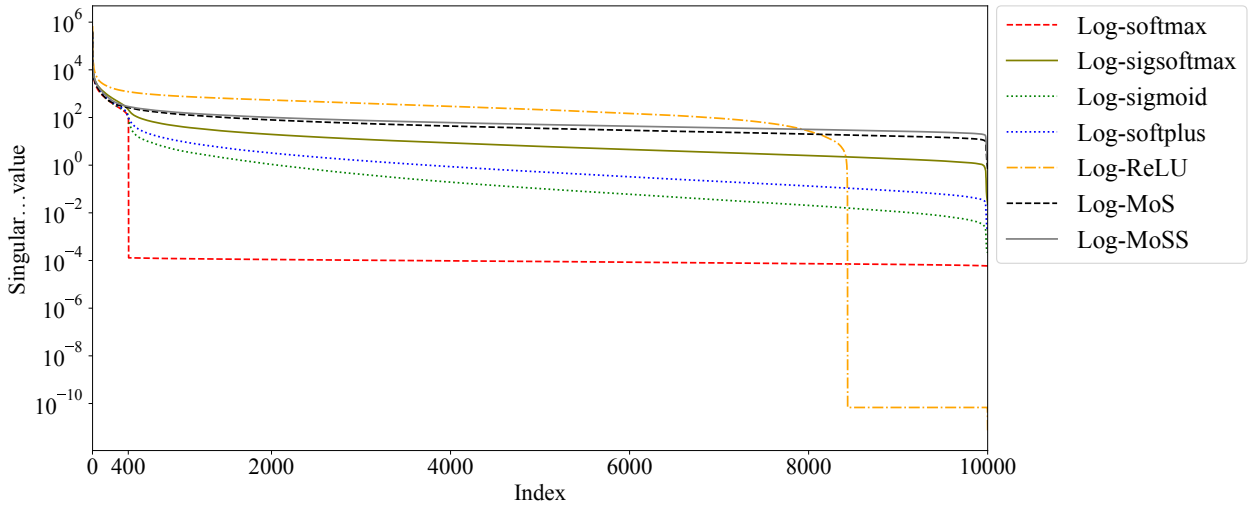
Table 4.3 Numbers of linearly independent vectors of log-outputs (The rank of $\hat{\boldsymbol{A}}$).

| | softmax | $g$:ReLU | $g$:sigmoid | $g$:softplus | sigsoftmax |
|---|---|---|---|---|---|
| PTB | 402 | 8243 | 542 | 1279 | 4640 |
| WT2 | 402 | 31400 | 463 | 593 | 5465 |

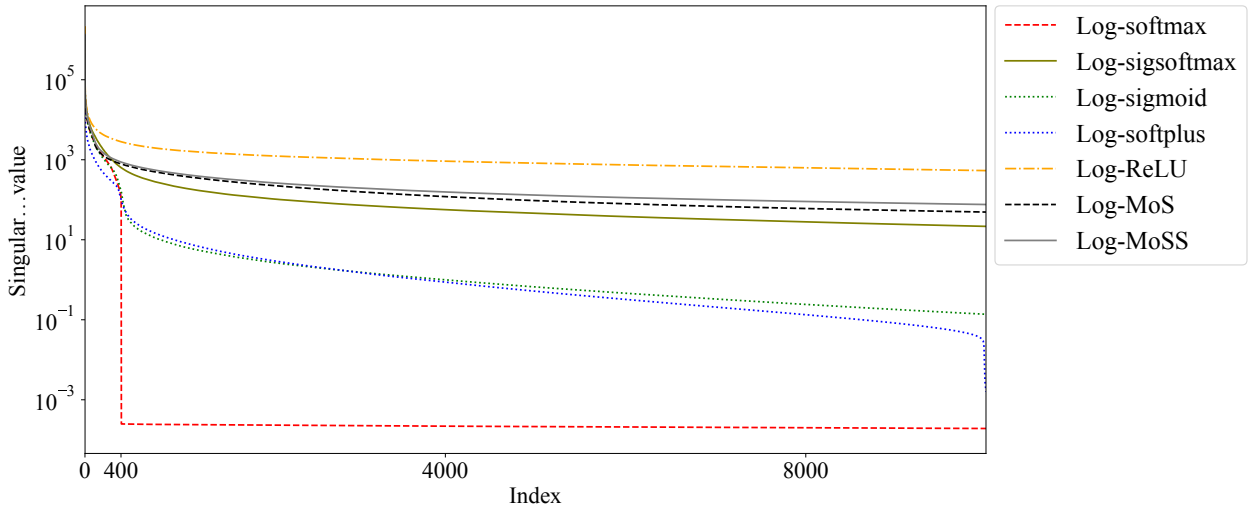
| | MoS | MoR | MoSig | MoSp | MoSS |
|---|---|---|---|---|---|
| PTB | 9980 | 9999 | 9977 | 9980 | 9986 |
| WT2 | 12093 | 30698 | 9021 | 10568 | 19834 |

を Table 4.3 に示す．またランクの計算に使用した特異値の結果を Fig. 4.4 に示す．これらの表より，log-softmax は 402 個の線形独立なベクトルを含む．実験では隠れユニット数を 400 としバイアスベクトルを加えたため，入力空間 $S$ の次元 $d$ は最大 401 である．定理 2 から log-softmax の出力は最大 $d + 1 = 402$ 個の線形独立なベクトルしか持たないので，定理通りの結果が得られた．一方，sigsoftmax と ReLU，sigmoid，softplus を使った関数は 402 に制限されない．そのため，これらの関数は softmax bottleneck を解消している．ReLU を使った関数はランクが高いが, 4.2.2 項で述べたように数値的に不安定であり，Table 4.1 と 4.2 で示したように効果的に学習できていない．MoSS の線形独立なベクトルの本数は MoS のものより大きく，MoSS の表現能力は MoS より大きい可能性が高い．

なお Fig. 4.4 より，明らかに特異値が下がっているのは softmax と ReLU のみである．ランク計算の数値誤差は大きく，Table 4.3 他のランクの計算精度は不明である．しかしながら，少なくとも sigsoftmax の特異値は他と比べて大きく，定理 4.4 のとおり，$M$ 個の線形独立なベクトルが得られた可能性が高い．




(a) The singular values of $\hat{\boldsymbol{A}}$ for PTB.




(b) The singular values of $\hat{\boldsymbol{A}}$ for WT2.

Fig. 4.4 The singular values of $\hat{\boldsymbol{A}}$. The rank of $\hat{\boldsymbol{A}}$ corresponds to the number of independent vectors of log-output vectors. Singular values of log-softmax drop at 402, and it indicates the softmax bottleneck occurs on these datasets. On the other hand, singular values of other functions decrease smoothly. It indicates that the number of linearly independent log-output vectors of these functions are larger than 402.

Table 4.4 Bit-per-characters of character-based language models on text8.

| | Softmax | Sigsoftmax |
|---|---|---|
| Valid | 1.49 | 1.48 |
| Test | 1.56 | 1.56 |

## 4.5 文字レベルの言語モデリング

ここまで，単語レベルの言語モデルで出力関数を評価した．これは単語レベルの言語モデルは softmax bottleneck が生じるといわれるためである．しかし，このデータセットによる評価だけでは sigsoftmax の性能向上が softmax bottleneck の解消によるものか確かではない．そこで，softmax bottleneck が生じない文字レベルの言語モデルの実験を行った．データセットには text8 を使用した [86]．文字レベルの言語モデルは $M$ が $d$ より小さいため，前節の単語レベルの言語モデルの実験と異なり softmax bottleneck が顕在化しない．そのため，この実験を通して sigsoftmax と softmax の結果があまり違いがない場合，単語レベルの言語モデルの sigsoftmax の性能の高さが softmax bottleneck の解消にあったと判断できる．

モデルとして 1 層 1024 ユニットの LSTM を使用し，学習には学習率 $\eta$ を 0.001 とした Adam を使用した．バッチサイズを 100 とし，BPTT を 200 ステップで打ち切った，text8 が 26 個の小文字のアルファベットと空白からなるため，$M$ は 27 であり $M$ は $d$ より小さかった．

Table 4.4 に文字レベルの言語モデルの結果をまとめる．表より，sigsoftmax の結果は softmax と違いがほとんどない．この実験は前述のとおり $M < d$ で softmax bottleneck が生じないため，この結果は単語レベルの言語モデルの sigsoftmax の性能の高さはボトルネックの解消によるものであることを示唆する．

## 4.6 One Billion Word dataset

さらなる有効性の確認のため One Billion Word という，より大きなデータセットで単語レベルの言語モデルの実験を行った [87].

### 4.6.1 実験条件

One Billion Word は 0.8 億単語，語彙数 $800 \times 10^3$ 個というこれまで使用したデータセットと比べて非常に大きいデータセットである．そのため，より表現力の高いモデルが求めら

Table 4.5 Test perplexities of language models of One Billion Word dataset.

| | Adaptive softmax | Adaptive sigsoftmax |
|---|---|---|
| Test | 33.80±0.04 | 33.62±0.05 |

れるのに加え，より効率的なモデルを用いる必要がある．Sigsoftmax は softmax の代替関数であるので，softmax のために提案された計算効率化方法と組み合わせることが可能であり，本論文では適応的なアプローチ [64] を使用した．Softmax の適応的なアプローチを adaptive softmax，sigsoftmax の適応的なアプローチを adaptive sigsoftmax と呼ぶことにする．この方法の調整パラメータであるカットオフは [4000, 40000, 200000] に設定した．モデルは 2 層の 2048 ユニットの LSTMs とし，学習率 $\eta = 20$ の SGD を使って 5 エポック学習した．2 回目のエポック以降は各エポックのあとに学習率を 2 で除算した．単語の埋め込みサイズは 256 とし，0.01 の dropout を使用し，バッチサイズを 128 とした．閾値 0.25 の勾配クリッピングを使用し，BPTT は 20 ステップで打ち切った．

### 4.6.2 実験結果

Table 4.5 に One Billion Word の結果をまとめる．表より adaptive sigsoftmax のパープレキシティは adaptive softmax より小さい．よって，非常に膨大なデータセットに対しても sigsoftmax は使用可能であり，効率的なアプローチも一緒に利用できる．

## 4.7 まとめ

本章では softmax による表現能力のボトルネックを log-softmax の値域から示し，それを解消する出力関数について評価した．その結果として softmax bottleneck を解消する関数として sigmoid や ReLU，softplus を使用することができるが，その性能は問題の規模や使用方法に依存することがわかった．Sigsoftmax は，所望の性質を満たしつつ softmax bottleneck を解消し，softmax を任意の精度で近似することができるため，実験においても softmax 以上の性能か最低でも同等の性能が得られることがわかった．2.2.4 項では softmax が $\boldsymbol{h}$ の分布を仮定することで導かれることを示したが，sigsoftmax やその他の出力関数がどのような仮定のもとで導出されるかは今後の課題である．

# 第 5 章

# 結論

本論文では離散分布に従う時系列データのモデリングにおいて，使いやすく高精度に recurrent neural network（RNN）を用いるための以下の 2 つについて述べた．

1. Gated recurrent unit（GRU）について，学習時の勾配爆発を抑制して学習を安定化する方法を提案し，その有効性を実験で検証した．
2. 既存の出力関数である softmax の表現能力の限界とその原因を調査し，それを解消する sigmoid と指数関数からなる sigsoftmax を提案し，有効性を実験で示した．

## 5.1 本論文による成果

第 2 章では深層学習の基礎についてまとめた．まず，深層学習の基本的なモデル構造や学習方法について概説し，RNN と RNN を使った時系列データのモデリングについて説明した．最後に，本論文の関連研究について述べた．

第 3 章では，時系列データのモデリングに有効な RNN のモデル構造の一つである GRU について，より安定的に学習することが可能な技術を提案した．まず，GRU の状態の振る舞いについて解析し，一つの平衡点の安定性と分岐点について明らかにした．また，この解析をさらに複数層重ねた GRU に拡張し，同様に平衡点とそれが安定である条件を明らかにした．解明された安定性の条件に基づき，その平衡点を安定に保つ学習方法を特異値に関する制約付きの最適化問題に定式化した．そして，この制約付き最適化問題を深層学習のような大規模データ，大規模モデルで解くための方法を提案し，さらに特異値制約付きの最適化問題の計算量を削減する学習方法を提案した．提案法の有効性を評価する実験として，言語データのモデリングと音楽データのモデリングを行った．実験によって，提案法は既存方法である勾配クリッピングより学習が成功しやすく，かつ精度も向上できることを明らかにした．提案法の学習速度についても，既存方法と同程度かより高速になることを示した．

第 4 章では離散値をとるデータのモデリングに用いられる深層学習の出力関数である softmax について，表現力の解析し，softmax より表現力の高い出力関数である sigsoftmax を提案した．まず，表現能力の限界を調査するために，softmax の対数である log-softmax の値域を解析した．その結果として，log-softmax の値域は，たかだか入力の空間の次元 +1 次元の空間の部分集合であることを明らかにした．これが表現力のボトルネックの原因である．次に，この原因を解消するための出力関数を複数個提示し，その性質について概説した．最後に，所望の性質を満たす新たな出力関数である sigsoftmax を提案し，sigsoftmax がボトルネックを解消する可能性があること，そして陽に softmax の出力を近似した出力ができることを示した．Sigsoftmax の有効性を評価する実験として，単語を一つのデータとしてみた言語モデルの実験，文字を一つとしてみた言語モデルの実験，そして約 10 億単語の巨大な言語モデル実験を行った．その結果として，単語レベルの言語モデルではボトルネック解消により sigsoftmax を使うことで精度向上が可能であり，文字レベルの言語モデルではボトルネックが顕在化しないため，sigsoftmax と softmax が同程度であった．出力の対数をとったベクトルの線形独立性を調べると，log-softmax は入力の空間の次元 +1 だけ線形独立なベクトルを持つことが確かめられた．一方で，log-sigsoftmax は入力の空間の次元 +1 よりも大きな数の線形独立なベクトルを持ち，softmax が表現力のボトルネックとなる問題を解消できることが確かめられた．

## 5.2 今後の発展性

本論文では勾配爆発を防ぐという目的のもとで，GRU の状態の振る舞いを解析し，一つの平衡点の安定性について解析した．この解析の発展として GRU や LSTM の状態の振る舞いをより広範に解析した研究が進んでいる [56, 88]．こうした研究を通じてブラックボックスな部分が多い深層学習モデルの解釈性が向上し，モデルが学習した時系列データの特徴を，より詳しく理解できるようになると期待される．

また，本論文では安定性の制約のために重み行列の最大特異値を制約する方法とその低計算量化を提案した．この最大特異値（スペクトルノルム）は，GAN の学習安定性 [59] や畳み込みニューラルネットワークの汎化性能の向上 [89]，そして深層学習モデルのノイズや攻撃に対するロバスト性の向上 [58, 90, 91] などの目的においても制約される．そのため，この低計算量の最大特異値の制約のもとでの学習方法は，これらの目的にも広く適用可能である．

本論文の GRU の振る舞いの解析では入力の影響を考慮していない．また，バイアス項のない GRU のモデルを対象とした．入力があると，状態が必ずしも一つの平衡点の吸引領域にとどまるとは限らず，またバイアスがあると平衡点の値が変化する．これらを考慮した GRU の解析と安定な学習方法の提案は今後の課題である．

一方，とりうる値が大きい離散分布をモデリングするための出力関数については，本論文で

は言語モデルによって評価した．しかし，提案した sigsoftmax は自然言語や時系列データに限らず画像認識などにおいても利用可能である．一般に画像認識では出力の数が中間層の状態の数より小さいことが多いため，softmax のボトルネックが顕在することは少ない．しかし携帯端末などの限られたリソースのもとで，軽量なモデルを必要とするアプリケーションも多数存在し，そのようなケースにおいて状態数を制限した小さな画像認識モデルが用いられることが予想される．このようなユースケースにおいても提案の sigsoftmax は高い性能を示すと期待される．

本論文では出力の対数が存在する空間の次元を扱ったが，これは非線形関数の表現能力の解析としては必要条件であり，十分ではない．より一般的には，出力関数がモデリングする多様体について，その次元などを解析することが必要である．真の確率分布にどういった仮定をおけば，sigsoftmax が優れているといえるのか，例えば softmax のように最終層の状態にどのような確率分布が仮定されるのか，最適化のしやすさ，汎化性能に違いがあるかなどの softmax bottleneck 以外の性質の調査は今後の課題である．なお，このボトルネックの解消する出力関数についての研究はその後も続けられており [68, 69]，例えば [68] ではパラメータを持つ単調関数を使用することでより大きな表現能力を持つ出力関数を提案している．

# 謝辞

本研究は、著者が慶應義塾大学大学院理工学研究科後期博士課程在学中に、同大学理工学部物理情報工学科の足立修一教授の指導のもとに行ったものです．指導教員として，ご指導とご鞭撻，審査の主査をしていただいた足立先生に深謝いたします．本論文をまとめるにあたり，副査をご担当いただいた慶應義塾大学理工学部物理情報工学科内山孝憲教授，同電気情報工学科湯川正裕准教授，同数理科学科小林景准教授には，心から感謝の意を表します．内山孝憲教授には本論文をまとめるにあたり，よりわかりやすい構成や厳密な表現のご指導をいただました．湯川正裕准教授には GRU の安定性について，有益な最新研究をご提示いただき，また，低ランク近似の方法との関連性からの発展の方向をご教示いただきました．小林景准教授には的確なご指摘をいただき，特に GRU 学習安定化の低計算量削減では，さらなるアイデアをいただき，また sigsoftmax の表現力の考察についてはより強い理論的結果とその証明方法を詳細にご教示いただきました．

研究を進めるにあたり，研究テーマの妥当性と意義に関する部分から，論文執筆まで一つ一つ丁寧に教えていただいた NTT で指導者をしていただいた藤原靖宏博士に感謝いたします．藤原博士のおかげで論文を，より高い精度で書き上げることができました．softmax に関する表現力を調べるにあたり，議論の相手になっていただき，さまざまなヒントや洞察をいただいた NTT 同期の山中友貴氏に感謝いたします．

研究室の斎藤由美秘書には，学部，修士，博士と研究室生活で長期間にわたって大変お世話になりました．ありがとうございます．足立研究室の博士で半年先に入学された阿部侑真博士（NICT）には博士課程を進めるにあたり，さまざまな手続きなどを教えていただき感謝します．また研究室の学生の皆さまには，研究室の計算機，実験室の管理などのさまざまな部分で協力いただきました，改めて感謝いたします．

本論文の実験を行うにあたり，その計算基盤として NTT の Corevo Computing Infrastructure（CCI）を整備，運用そして私の質問に丁寧にご対応いただいた皆様に感謝いたします．数千時間にわたった実験を遂行できたのは CCI のおかげです．著者の所属する NTT で同グループの先輩の井田安俊氏には深層学習の基本的な部分についてたくさん教えていただきました，同グループの深層学習の研究テーマを推し進めてくださりありがとうございます．上司で

あった岩村相哲博士，飯田恭弘氏には研究に集中する環境を整えてくださり感謝いたします．また本論文をまとめる間，業務と並行して円滑に学位取得が進むようご協力いただいた上司や同僚の皆様に感謝いたします．NTT の高橋大志氏をはじめとした同期の皆様には研究や研究生活においてたくさんの助言をいただきありがとうございました．

最後に大学まで育ててくれて，就職した現在も支えてくれる両親に感謝いたします．

# 参考文献


[1] W. Zaremba, I. Sutskever and O. Vinyals: Recurrent neural network regularization, arXiv preprint arXiv:1409.2329 (2014)

[2] Y. Gal and Z. Ghahramani: A theoretically grounded application of dropout in recurrent neural networks, In Proc. The Annual Conf. on Neural Information Processing Systems, 1019/1027 (2016)

[3] 金井，藤原，岩村，足立：Gated recurrent unit の局所安定化による勾配爆発の抑制，電子情報通信学会論文誌 D，**102**-8, 530/541 (2019)

[4] 金井，藤原，山中，足立：Softmax Bottleneck の再解釈とそれを解消する出力関数の評価，電子情報通信学会論文誌 D，**103**-6, 518/528 (2020)

[5] B. Krause, E. Kahembwe, I. Murray and S. Renals: Dynamic evaluation of neural sequence models, arXiv preprint arXiv:1709.07432 (2017)

[6] J. Schmidhuber: Deep learning in neural networks: An overview, **61**, 85/117, Elsevier (2015)

[7] A. Krizhevsky, I. Sutskever and G. E. Hinton: Imagenet classification with deep convolutional neural networks, In Proc. The Annual Conf. on Neural Information Processing Systems, 1097/1105 (2012)

[8] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, A. Hannun, B. Jun, T. Han, P. LeGresley, X. Li, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, S. Qian, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, C. Wang, Y. Wang, Z. Wang, B. Xiao, Y. Xie, D. Yogatama, J. Zhan and Z. Zhu: Deep speech 2: End-to-end speech recognition in english and mandarin, In Proc. International Conf. on Machine Learning., 173/182 (2016)

[9] A. Graves, A.-r. Mohamed and G. Hinton: Speech recognition with deep recurrent neural networks, In Proc. IEEE International Conf. on Acoustics, Speech, and Signal Processing, 6645/6649, IEEE (2013)

[10] J. L. Elman: Finding structure in time, Cognitive science, **14**-2, 179/211 (1990)

[11] T. Mikolov: Statistical language models based on neural networks, PhD thesis, Brno University of Technology (2012)

[12] A. Graves and J. Schmidhuber: Offline handwriting recognition with multidimensional recurrent neural networks, In Proc. The Annual Conf. on Neural Information Processing Systems, 545/552 (2009)

[13] R. Pascanu, T. Mikolov and Y. Bengio: On the difficulty of training recurrent neural networks, In Proc. International Conf. on Machine Learning., 1310/1318 (2013)

[14] Y. Bengio, P. Simard and P. Frasconi: Learning long-term dependencies with gradient descent is difficult, IEEE Transactions on Neural Networks, **5**-2, 157/166 (1994)

[15] S. Hochreiter and J. Schmidhuber: Long short-term memory, Neural computation, **9**-8, 1735/1780 (1997)

[16] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk and Y. Bengio: Learning phrase representations using rnn encoder–decoder for statistical machine translation, In Proc. Conf. on Empirical Methods in Natural Language Processing, 1724/1734, ACL (2014)

[17] J. S. Bridle: Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, In Neurocomputing, Springer, 227/236 (1990)

[18] I. Goodfellow, Y. Bengio and A. Courville: Deep learning, 66,178, MIT press (2016)

[19] K. He, X. Zhang, S. Ren and J. Sun: Deep residual learning for image recognition, In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 770/778 (2016)

[20] I. Sutskever, O. Vinyals and Q. V. Le: Sequence to sequence learning with neural networks, In Proc. The Annual Conf. on Neural Information Processing Systems, 3104/3112 (2014)

[21] Z. Yang, Z. Dai, R. Salakhutdinov and W. W. Cohen: Breaking the softmax bottleneck: a high-rank rnn language model, In Proc. The International Conf. on Learning Representations (2018)

[22] A. Krizhevsky, I. Sutskever and G. E. Hinton: Imagenet classification with deep convolutional neural networks, In Proc. The Annual Conf. on Neural Information Processing Systems, 1097/1105 (2012)

[23] A. Krizhevsky, I. Sutskever and G. E. Hinton: Imagenet classification with deep convolutional neural networks, In Proc. The Annual Conf. on Neural Information Processing Systems, 1097/1105 (2012)

[24] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser and I. Polosukhin: Attention is all you need, In Proc. The Annual Conf. on Neural Information Processing Systems, 5998/6008 (2017)

[25] G. Hinton, L. Deng, D. Yu, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, B. Kingsbury *et al.*: Deep neural networks for acoustic modeling in speech recognition, IEEE Signal processing magazine, **29**-6, 82/97 (2012)

[26] V. Nair and G. E. Hinton: Rectified linear units improve restricted boltzmann machines, In Proc. International Conf. on Machine Learning., 807/814, Omnipress (2010)

[27] A. L. Maas, A. Y. Hannun and A. Y. Ng: Rectifier nonlinearities improve neural network acoustic models, In in ICML Workshop on Deep Learning for Audio, Speech and Language Processing (2013)

[28] K. He, X. Zhang, S. Ren and J. Sun: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, In Proc. ICCV, 1026/1034 (2015)

[29] K. Kawaguchi: Deep learning without poor local minima, In Proc. The Annual Conf. on Neural Information Processing Systems, 586/594 (2016)

[30] Z. Allen-Zhu, Y. Li and Z. Song: A convergence theory for deep learning via over-parameterization, arXiv preprint arXiv:1811.03962 (2018)

[31] 岡谷貴之：深層学習, 30/33，講談社 (2015)

[32] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov: Dropout: a simple way to prevent neural networks from overfitting., Journal of Machine Learning Research, **15**-1, 1929/1958 (2014)

[33] C. M. Bishop: Neural Networks for Pattern Recognition, 238/240,390 (1995)

[34] X. Glorot and Y. Bengio: Understanding the difficulty of training deep feedforward neural networks, In Proceedings of the thirteenth international conference on artificial intelligence and statistics, 249/256 (2010)

[35] K. He, X. Zhang, S. Ren and J. Sun: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, In Proceedings of the IEEE international conference on computer vision, 1026/1034 (2015)

[36] A. M. Saxe, J. L. McClelland and S. Ganguli: Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, In Proc. The International Conf. on Learning Representations (2014)

[37] S. Wiggins: Introduction to applied nonlinear dynamical systems and chaos, **2**, 7,350,356/363, Springer Science & Business Media (2003)

[38] K. Doya: Bifurcations in the learning of recurrent neural networks, In Proc. ISCAS,

**6**, 2777/2780, IEEE (1992)

[39] P. Baldi and K. Hornik: Universal approximation and learning of trajectories using oscillators, In Proc. The Annual Conf. on Neural Information Processing Systems, 451/457 (1996)

[40] M. Arjovsky, A. Shah and Y. Bengio: Unitary evolution recurrent neural networks, In Proc. International Conf. on Machine Learning., 1120/1128 (2016)

[41] E. Vorontsov, C. Trabelsi, S. Kadoury and C. Pal: On orthogonality and learning recurrent networks with long term dependencies, In Proc. International Conf. on Machine Learning. (2017)

[42] X. Glorot and Y. Bengio: Understanding the difficulty of training deep feedforward neural networks, In Proc. International Conf. on Artificial Intelligence and Statistics., 249/256 (2010)

[43] 高村，奥村：言語処理のための機械学習入門, 76，コロナ社 (2010)

[44] O. Press and L. Wolf: Using the output embedding to improve language models, In Proc. Conf. of the European Chapter of the Association for Computational Linguistics Short Papers, 157/163 (2017)

[45] C. M. Kuan, K. Hornik and H. White: A convergence result for learning in recurrent neural networks, Neural Computation, **6**-3, 420/440 (1994)

[46] W. Yu: Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms, Information sciences, **158**, 131/147 (2004)

[47] H. Jaeger: Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the" echo state network" approach, GMD-Forschungszentrum Informationstechnik (2002)

[48] J. Zhang, Q. Lei and I. Dhillon: Stabilizing gradients for deep neural networks via efficient SVD parameterization, In Proc. International Conf. on Machine Learning. **80**, 5806/5814, PMLR (2018)

[49] B. Doyon, B. Cessac, M. Quoy and M. Samuelides: Destabilization and route to chaos in neural networks with random connectivity, In Proc. The Annual Conf. on Neural Information Processing Systems, 549/555 (1993)

[50] N. E. Barabanov and D. V. Prokhorov: Stability analysis of discrete-time recurrent neural networks, IEEE Transactions on Neural Networks, **13**-2, 292/303 (2002)

[51] J. A. Suykens, B. De Moor and J. Vandewalle: Robust local stability of multilayer recurrent neural networks, IEEE Transactions on Neural Networks, **11**-1, 222/229 (2000)

[52] R. Haschke and J. J. Steil: Input space bifurcation manifolds of recurrent neural

networks, Neurocomputing, **64**, 25/38 (2005)

[53] H. Nakahara and K. Doya: Dynamics of attention as near saddle-node bifurcation behavior, In Proc. The Annual Conf. on Neural Information Processing Systems, 38/44 (1996)

[54] S. S. Talathi and A. Vartak: Improving performance of recurrent neural network with relu nonlinearity, arXiv preprint arXiv:1511.03771 (2015)

[55] T. Laurent and J. von Brecht: A recurrent neural network without chaos, In Proc. The International Conf. on Learning Representations (2017)

[56] D. M. Stipanović, B. Murmann, M. Causo, A. Lekić, V. R. Royo, C. J. Tomlin, E. Beigne, S. Thuries, M. Zarudniev and S. Lesecq: Some local stability properties of an autonomous long short-term memory neural network model, In Circuits and Systems (ISCAS), 2018 IEEE International Symposium on, 1/5, IEEE (2018)

[57] Y. Yoshida and T. Miyato: Spectral norm regularization for improving the generalizability of deep learning, arXiv preprint arXiv:1705.10941 (2017)

[58] M. Cisse, P. Bojanowski, E. Grave, Y. Dauphin and N. Usunier: Parseval networks: Improving robustness to adversarial examples, In International Conference on Machine Learning, 854/863 (2017)

[59] T. Miyato, T. Kataoka, M. Koyama and Y. Yoshida: Spectral normalization for generative adversarial networks, Proc. The International Conf. on Learning Representations (2018)

[60] H. Sedghi, V. Gupta and P. M. Long: The singular values of convolutional layers, arXiv preprint arXiv:1805.10408 (2018)

[61] A. de Brébisson and P. Vincent: An exploration of softmax alternatives belonging to the spherical loss family, In Proc. The International Conf. on Learning Representations (2016)

[62] P. Mohassel and Y. Zhang: Secureml: A system for scalable privacy-preserving machine learning, In Proc. IEEE Symposium on Security and Privacy, 19/38 (2017)

[63] Y. Ollivier: Riemannian metrics for neural networks i: feedforward networks, arXiv preprint arXiv:1303.0818 (2013)

[64] É. Grave, A. Joulin, M. Cissé, D. Grangier and H. Jégou: Efficient softmax approximation for GPUs, In Proc. International Conf. on Machine Learning., 1302/1310 (2017)

[65] K. Shim, M. Lee, I. Choi, Y. Boo and W. Sung: SVD-softmax: Fast softmax approximation on large vocabulary neural networks, In Proc. The Annual Conf. on Neural Information Processing Systems, 5469/5479 (2017)

[66] M. K. Titsias: One-vs-each approximation to softmax for scalable estimation of probabilities, In Proc. The Annual Conf. on Neural Information Processing Systems, 4161/4169 (2016)

[67] A. Martins and R. Astudillo: From softmax to sparsemax: A sparse model of attention and multi-label classification, In Proc. International Conf. on Machine Learning., 1614/1623 (2016)

[68] O. Ganea, S. Gelly, G. Becigneul and A. Severyn: Breaking the softmax bottleneck via learnable monotonic pointwise non-linearities, In Proc. International Conf. on Machine Learning., 2073/2082 (2019)

[69] Z. Yang, T. Luong, R. R. Salakhutdinov and Q. V. Le: Mixtape: Breaking the softmax bottleneck efficiently, In Proc. The Annual Conf. on Neural Information Processing Systems, 15922/15930 (2019)

[70] S. Kanai, Y. Fujiwara and S. Iwamura: Preventing gradient explosions in gated recurrent units, In Proc. The Annual Conf. on Neural Information Processing Systems, 435/444 (2017)

[71] G. Strang: Calculus (2nd ed.), Wellesley-Cambridge Press (2010)

[72] S. Lefkimmiatis, J. P. Ward and M. Unser: Hessian schatten-norm regularization for linear inverse problems, IEEE transactions on image processing, **22**-5, 1873/1888 (2013)

[73] R. A. Horn and C. R. Johnson: Topics in matrix analysis, 134/238, Cambridge university press (1991)

[74] D. Kingma and J. Ba: Adam: A method for stochastic optimization, In Proc. The International Conf. on Learning Representations (2015)

[75] N. Halko, P. Martinsson and J. Tropp: Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions, arXiv preprint arXiv:0909.4061 (2009)

[76] M. P. Marcus, M. A. Marcinkiewicz and B. Santorini: Building a large annotated corpus of english: The penn treebank, Computational linguistics, **19**-2, 313/330 (1993)

[77] S. Merity, C. Xiong, J. Bradbury and R. Socher: Pointer sentinel mixture models, In Proc. The International Conf. on Learning Representations (2017)

[78] N. Boulanger-Lewandowski, Y. Bengio and P. Vincent: Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription, In Proc. International Conf. on Machine Learning., 1159/1166 (2012)

[79] R. Jozefowicz, W. Zaremba and I. Sutskever: An empirical exploration of recurrent network architectures, In Proc. International Conf. on Machine Learning., 2342/2350 (2015)

[80] D. Krueger and R. Memisevic: Regularizing rnns by stabilizing activations, In Proc. The International Conf. on Learning Representations (2016)

[81] Z. Tang, Y. Shi, D. Wang, Y. Feng and S. Zhang: Memory visualization for gated recurrent neural networks in speech recognition, In Proc. IEEE International Conf. on Acoustics, Speech, and Signal Processing, 2736/2740, IEEE (2017)

[82] S. Kanai, Y. Fujiwara, Y. Yamanaka and S. Adachi: Sigsoftmax: Reanalysis of the softmax bottleneck, In Proc. The Annual Conf. on Neural Information Processing Systems, 284/294 (2018)

[83] C. M. Bishop: Pattern Recognition and Machine Learning., p. 198, Springer-Verlag New York (2006)

[84] S. Merity, N. S. Keskar and R. Socher: Regularizing and optimizing lstm language models, In Proc. The International Conf. on Learning Representations (2018)

[85] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery: Numerical Recipes 3rd Edition: The Art of Scientific Computing, 67/69, Cambridge University Press (2007)

[86] M. Mahoney: Large text compression benchmark, (2011)

[87] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, P. Koehn and T. Robinson: One billion word benchmark for measuring progress in statistical language modeling, Technical report, Google (2013)

[88] T. Can, K. Krishnamurthy and D. J. Schwab: Gating creates slow modes and controls phase-space complexity in grus and lstms, arXiv preprint arXiv:2002.00025 (2020)

[89] H. Sedghi, V. Gupta and P. M. Long: The singular values of convolutional layers, In Proc. The International Conf. on Learning Representations (2019)

[90] F. Farnia, J. Zhang and D. Tse: Generalizable adversarial training via spectral normalization, In International Conference on Learning Representations (2019)

[91] S. Kanai, Y. Ida, Y. Fujiwara, M. Yamada and S. Adachi: Absum: Simple regularization method for reducing structural sensitivity of convolutional neural networks, In Proc. AAAI Conf. on Artificial Intelligence, 4394/4403 (2020)

# 付録 A

# softmax bottleneck の証明

本付録では [21] 内の定理 4.1 の証明を引用する．まず，式 (4.2) について，その行ベクトルをシフトした行列の集合

$$F(\boldsymbol{A}) = \left\{ \boldsymbol{A} + \boldsymbol{\Lambda}\boldsymbol{J}_{N,M} \mid \boldsymbol{\Lambda}\text{は対角行列であり}\boldsymbol{\Lambda} \in \boldsymbol{R}^{N\times M} \right\} \tag{A.1}$$

を考える，ただし，$\boldsymbol{J}_{N\times M}$ は $N \times M$ のすべて 1 の行列である．この行ベクトルごとのシフトは，任意の実数を行列 $\boldsymbol{A}$ の各列ベクトルに加える操作といえる．この $F(\boldsymbol{A})$ について次の 2 つの性質が成り立つ．

**性質 A.1** ([21] 中の Property 1)**.** 任意の行列 $\boldsymbol{A}'$ は，$\boldsymbol{A}' \in F(\boldsymbol{A})$ であるとき，およびその場合に限り，$\mathrm{Softmax}(\boldsymbol{A}') = P*$ である．ただし，$\mathrm{Softmax}(\cdot)$ は行ベクトルごとの softmax とする．

**証明.** 任意の $\boldsymbol{A}' \in F(\boldsymbol{A})$ について，これに Softmax を適用したものを $P_{\boldsymbol{A}'}(Y|X)$ とする．$\boldsymbol{A}'$ の $i$ 行 $j$ 列は $A'_{i,j} = A_{i,j} + \Lambda_{i,i}$ である．このとき，

$$P_{\boldsymbol{A}}(y_j|x_i) = \frac{\exp\ A'_{i,j}}{\sum_m \exp\ A'_{i,m}} = \frac{\exp\ (A_{i,j} + \Lambda_{i,i})}{\sum_m \exp\ (A_{i,m} + \Lambda_{i,i})} = \frac{\exp\ A_{i,j}}{\sum_m \exp\ A_{i,m}} = P*(y_j|x_i) \tag{A.2}$$

となる．一方，任意の $\boldsymbol{A}'' \in \{\boldsymbol{A}''|\mathrm{Softmax}(\boldsymbol{A}'') = P*\}$ について

$$P_{\boldsymbol{A}''}(y_j|x_i = P_{\boldsymbol{A}}(y_j|x_i) \tag{A.3}$$

が成り立つ．また任意の $i$，$j$，$m$ について

$$\frac{P_{\boldsymbol{A}''}(y_j|c_i)}{P_{\boldsymbol{A}''}(y_m|c_i)} = \frac{\exp\ A''_{i,j}}{\exp\ A''_{i,m}} = \frac{\exp A_{i,j}}{\exp A_{i,m}} = \frac{P_{\boldsymbol{A}}(y_j|x_i)}{P_{\boldsymbol{A}}(y_m|x_i)} \tag{A.4}$$

が成り立ち，

$$A''_{i,j} - A_{i,j} = A''_{i,m} - A_{i,m} \tag{A.5}$$

となる．これは $\boldsymbol{A}$ の各行ベクトルに任意の実数を与えて $\boldsymbol{A}''$ が得られることを表し，

$$\boldsymbol{A}'' = \boldsymbol{A} + \boldsymbol{\Lambda}\boldsymbol{J}_{N\times M} \tag{A.6}$$

となる対角行列 $\boldsymbol{\Lambda}$ が存在し，$\boldsymbol{A}'' \in F(\boldsymbol{A})$ である． □

**性質 A.2** ([21] 中の Property 2)**.** 任意の $\boldsymbol{A}_1 \neq \boldsymbol{A}_2 \in F(\boldsymbol{A})$ について，$|\mathrm{rank}(\boldsymbol{A}_1) - \mathrm{rank}(\boldsymbol{A}_2)| \leq 1$ が成り立つ．

**証明.** $F(\boldsymbol{A})$ の定義より，$\boldsymbol{A}_1 = \boldsymbol{A} + \boldsymbol{\Lambda}_1\boldsymbol{J}_{N\times M}$，$\boldsymbol{A}_2 = \boldsymbol{A} + \boldsymbol{\Lambda}_2\boldsymbol{J}_{N\times M}$ である．よって

$$\boldsymbol{A}_1 = \boldsymbol{A}_2 + (\boldsymbol{\Lambda}_1 - \boldsymbol{\Lambda}_2)\boldsymbol{J}_{N\times M} \tag{A.7}$$

となる．いま $S$ を $\boldsymbol{A}_2$ が含む線形独立の行ベクトルの最大の集合とし，$\boldsymbol{e}_N$ をサイズ $N$ のすべて 1 のベクトルとする．$\boldsymbol{A}_1$ の $i$ 番目の行ベクトル $\boldsymbol{a}_{1i}$ は同様に $\boldsymbol{A}_2$ の $i$ 番目の行ベクトル $\boldsymbol{a}_{2i}$ を使って

$$\boldsymbol{a}_{1i} = \boldsymbol{a}_{2i} + (\Lambda_{1i,j} - \Lambda_{2i,j})\boldsymbol{e}_N \tag{A.8}$$

となる．$\boldsymbol{a}_{2i}$ が $S$ に含まれる線形独立なベクトルの線形結合となることから，$\boldsymbol{a}_{1i}$ は $S \cup \{\boldsymbol{e}_N\}$ 内のベクトルの線形結合となる．よって，

$$\mathrm{rank}(\boldsymbol{A}_1) \leq \mathrm{rank}(\boldsymbol{A}_2) + 1 \tag{A.9}$$

であり，同様に，

$$\mathrm{rank}(\boldsymbol{A}_2) \leq \mathrm{rank}(\boldsymbol{A}_1) + 1 \tag{A.10}$$

である．したがって，

$$|\mathrm{rank}(\boldsymbol{A}_1) - \mathrm{rank}(\boldsymbol{A}_2)| \leq 1 \tag{A.11}$$

である． □

性質 A.1 から次の補題が簡単に導かれる．

**補題 A.1** ([21] 中 Lemma 1)**.** モデルのパラメータ $\boldsymbol{\theta}$が与えられたとき，$P_{\boldsymbol{\theta}}(Y|X) = P*(Y|X)$ となるとき，およびその場合に限り，$\boldsymbol{H}_{\boldsymbol{\theta}}\boldsymbol{W}^T \in F(\boldsymbol{A})$ である．

以上から定理 4.1 は次のように証明される．

**証明.** 補題 A.1 より，任意の $X \in \mathcal{L}$ に対して $P_{\boldsymbol{\theta}}(Y|X) = P*(Y|X)$ が成り立つならば，$\boldsymbol{H}_{\boldsymbol{\theta}}\boldsymbol{W}^T \in F(\boldsymbol{A})$ が成り立つ．結果として $\boldsymbol{A}' = \boldsymbol{H}_{\boldsymbol{\theta}}\boldsymbol{W}^T$ なる行列 $\boldsymbol{A}' \in F(\boldsymbol{A})$ が存在する．$\boldsymbol{H}_{\boldsymbol{\theta}}$ のサイズは $(N \times d)$ であり，$\boldsymbol{W}$ は $(M \times d)$ であることから，

$$d \geq \mathrm{rank}(\boldsymbol{A}') \geq \min_{\boldsymbol{A}''\in F(\boldsymbol{A})} \mathrm{rank}(\boldsymbol{A}'') \tag{A.12}$$

が成り立つ．もし，$d \geq \min_{\boldsymbol{A}'' \in F(\boldsymbol{A})} \mathrm{rank}(\boldsymbol{A}'')$ であれば，行列 $\boldsymbol{A}'$ は $\boldsymbol{A}' = \boldsymbol{H}' bm W'^T$，ただし，$\boldsymbol{H}' \in \boldsymbol{R}^{N \times d}$．$\boldsymbol{W}' \in \boldsymbol{R}^{M \times d}$ と分解できる．このとき万能な関数 $\mathcal{U}$ が存在すれば $\boldsymbol{H}_{\boldsymbol{\theta}} = \boldsymbol{H}'$，$\boldsymbol{W} = \boldsymbol{W}'$ が得られ，$P_{\boldsymbol{\theta}}(Y|X) = P*(Y|X)$ が成り立つ．一方，性質 A.2 より $\min_{\boldsymbol{A}'' \in F(\boldsymbol{A})} \mathrm{rank}(\boldsymbol{A}'') = \mathrm{rank}(\boldsymbol{A}) - 1$ であるから，逆に $d < \mathrm{rank}(\boldsymbol{A}) - 1$ のとき，$P_{\boldsymbol{\theta}}(Y|X) \neq P*(Y|X)$ となる $X$ が存在する．□

# 付録 B

# 研究業績

## 学位請求に用いる定期刊行誌掲載原著論文

[1] 金井，藤原，岩村，足立：Gated recurrent unit の局所安定化による勾配爆発の抑制，電子情報通信学会和文論文誌 D，Vol.J102-D，No.8，pp.530-541（2019）

[2] 金井，藤原，山中，足立：Softmax Bottleneck の再解釈とそれを解消する出力関数の評価，電子情報通信学会和文論文誌 D，Vol.J103-D，No.6，pp.518-528（2020）

## 学位請求に用いる国際会議発表論文（査読付き，発表者*）

[1] S. Kanai*, Y. Fujiwara, Y. Yamanaka, and S. Adachi: Sigsoftmax: Reanalysis of the Softmax Bottleneck, The 32nd Annual Conference on Neural Information Processing Systems (NeurIPS 2018), pp. 284-294, Canada (2018)

[2] S. Kanai*, Y. Fujiwara, and S. Iwamura: Preventing Gradient Explosions in Gated Recurrent Units, The 31st Annual Conference on Neural Information Processing Systems (NIPS 2017), pp. 435-444, USA (2017)

## その他の定期刊行誌掲載原著論文

[1] S. Kanai, M. Sugaya, S. Adachi, and K. Matsui: Low-complexity simultaneous estimation of head-related transfer functions by prediction error method, Journal of the Audio Engineering Society, Vol.64, No.11, pp.895-904 (2016)

[2] 金井，松井，中山，足立：入力設計の改良による頭部伝達関数の多方向同時推定の高精度化，日本音響学会誌 研究速報，vol.71，No.7，pp.316-318，(2015)

## その他の国際会議発表論文（査読付き，発表者*）

[1] Y. Ida*, S. Kanai,Y. Fujiwara*, T. Iwata, K. Takeuchi, and H. Kashima: Fast Deterministic CUR Matrix Decomposition with Accuracy Assurance, The 37th International Conference on Machine Learning (ICML 2020), Online (2020)

[2] Y. Fujiwara*, A. Kumagai, S. Kanai, Y. Ida, and N. Ueda: Efficient Algorithm for the b-Matching Graph, The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2020), Online (2020)

[3] S. Kanai*, Y. Ida, Y. Fujiwara, M. Yamada, and S. Adachi: Absum: Simple Regularization Method for Reducing Structural Sensitivity of Convolutional Neural Networks, The 34th AAAI Conference on Artificial Intelligence (AAAI 2020), pp. 4394-4403, USA (2020)

[4] S. Yamaguchi*, S. Kanai, and T. Eda: Effective Data Augmentation with Multi-Domain Learning GANs, The 34th AAAI Conference on Artificial Intelligence (AAAI 2020), pp. 6566-6574, USA (2020)

[5] Y. Fujiwara*, Y. Ida, S. Kanai, A. Kumagai, J. Arai, and N. Ueda: Fast Random Forest Algorithm via Incremental Upper Bound, The 28th ACM International Conference on Information and Knowledge Management (CIKM 2019), pp. 2205-2208, China (2019)

[6] Y. Yamanaka*, T. Iwata, H. Takahashi, M. Yamada, and S. Kanai: Autoencoding Binary Classifiers for Supervised Anomaly Detection, The 16th Pacific Rim International Conference on Artificial Intelligence (PRICAI), pp.647-659, Fiji (2019)

[7] Y. Fujiwara*, S. Kanai, J. Arai, Y. Ida, and N. Ueda: Efficient Data Point Pruning for One-Class SVM, The 33rd AAAI Conference on Artificial Intelligence (AAAI 2019), USA (2019)

[8] Y. Fujiwara*, J. Arai, S. Kanai, Y. Ida, and N. Ueda: Adaptive Data Pruning for Support Vector Machines, IEEE BigData, USA (2018)

[9] S. Kanai*, K. Matsui, Y. Nakayama, S. Adachi: Uncorrelated Input Signals Design and Identification with Low-Complexity for Simultaneous Estimation of Head-Related Transfer Functions, Audio Engineering Society Convention 137, USA, (2014)

[10] S. Kanai*, K. Matsui, and S. Adachi, Identification input design for simultaneous estimation of head-related transfer functions. SICE Annual Conference (SICE), Japan, pp. 1673-1678 (2014)

## 関連する主な国内会議発表（発表者*）

[1] 金井*，藤原，岩村：局所的に安定に制約した Gated Recurrent units の学習法，第 61 回自動制御連合講演会，南山大学，名古屋 (2018)

[2] 金井*，藤原，山中，足立：Softmax bottleneck の再考，第 21 回情報論的学習理論ワークショップ（IBIS2018），かでる 2.7，札幌 (2018)

[3] 金井*，藤原，岩村：GRU 学習時の勾配爆発の抑制方法の提案，第 19 回情報論的学習理論ワークショップ，京都大学，京都 (2016)

## 関連する国内登録特許

1. 金井，藤原，山中，学習装置，学習方法及び学習プログラム，出願番号 2018-083122，2018 年 4 月 24 日出願.
2. 金井，藤原，飯田，岩村，学習装置、学習方法および学習プログラム，出願番号 2016-203546，2016 年 10 月 17 日出願.

## 関連する NTT 技術ジャーナル

[1] 井田, 金井, 藤原, 八木, 飯田：深層学習のための先進的な学習技術, NTT 技術ジャーナル, Vol. 30, No.6, pp. 31-33 (2018)

## 受賞

[1] 2019 年度 電子情報通信学会論文賞，Gated recurrent unit の局所安定化による勾配爆発の抑制 (2020)