LONG SHORT-TERM MEMORY

Hochreiter, S., & Schmidhuber, J. (1997).
Neural computation, *9*(8), 1735-1780.

理論談話会2020

6月16日

M2 石井 健太

はじめに

■要旨

時系列データを扱うためのニューラルネットの構造を提案.

既往研究では扱うことが難しかった，長期的にわたる関係性も考慮できるよう，CEC・入力ゲート・出力ゲートという3つの構造を導入している.

実験により，長期短期双方の関係性を考慮したうえで予測可能なことを証明した.

■交通分野での用途

- 交通量予測 (Toon Bogaerts et al. 2020など)
- オンデマンド交通の逐次需要予測 (Jintao Ke et al. 2017など)
- レーン変更選択モデル (Xiaohui Zhang et al. 2019など)
- GPSデータからの交通機関特定 (Sina Dabiri et al. 2020)
- 災害時の交通レジリエンスの時空間パターン推定・予測 (Hong-Wei Wang et al. 2020)

などなど

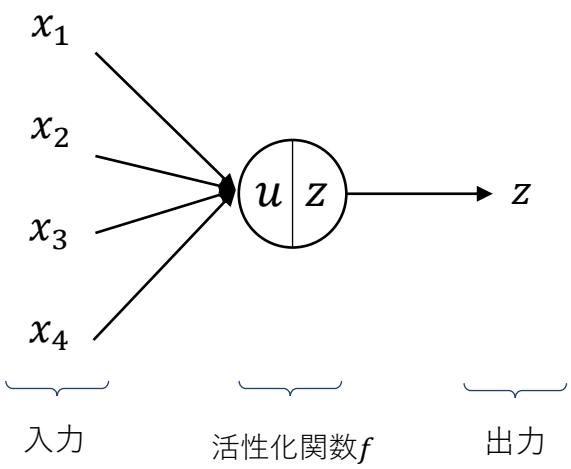
■ **論文構成**

1. Introduction
2. Previous work
3. Constant error backprop
4. Long short-term memory
5. Experiments
6. Discussion
7. Conclusion

目次

■ユニット

ネットワークを構成する基本単位



$u = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + b$

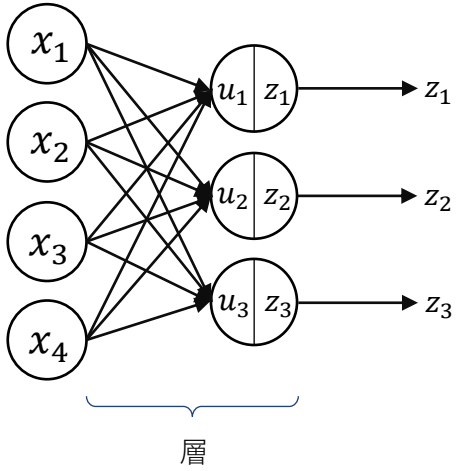
w_1, w_2, w_3, w_4：各入力の重み

b ：バイアス

$z = f(u)$：活性化関数

■順伝播型ネットワーク

ユニットが層状に並び，層間でのみ結合を持つ



$$\mathbf{u} = \mathbf{W}\mathbf{x} + \mathbf{b}$$

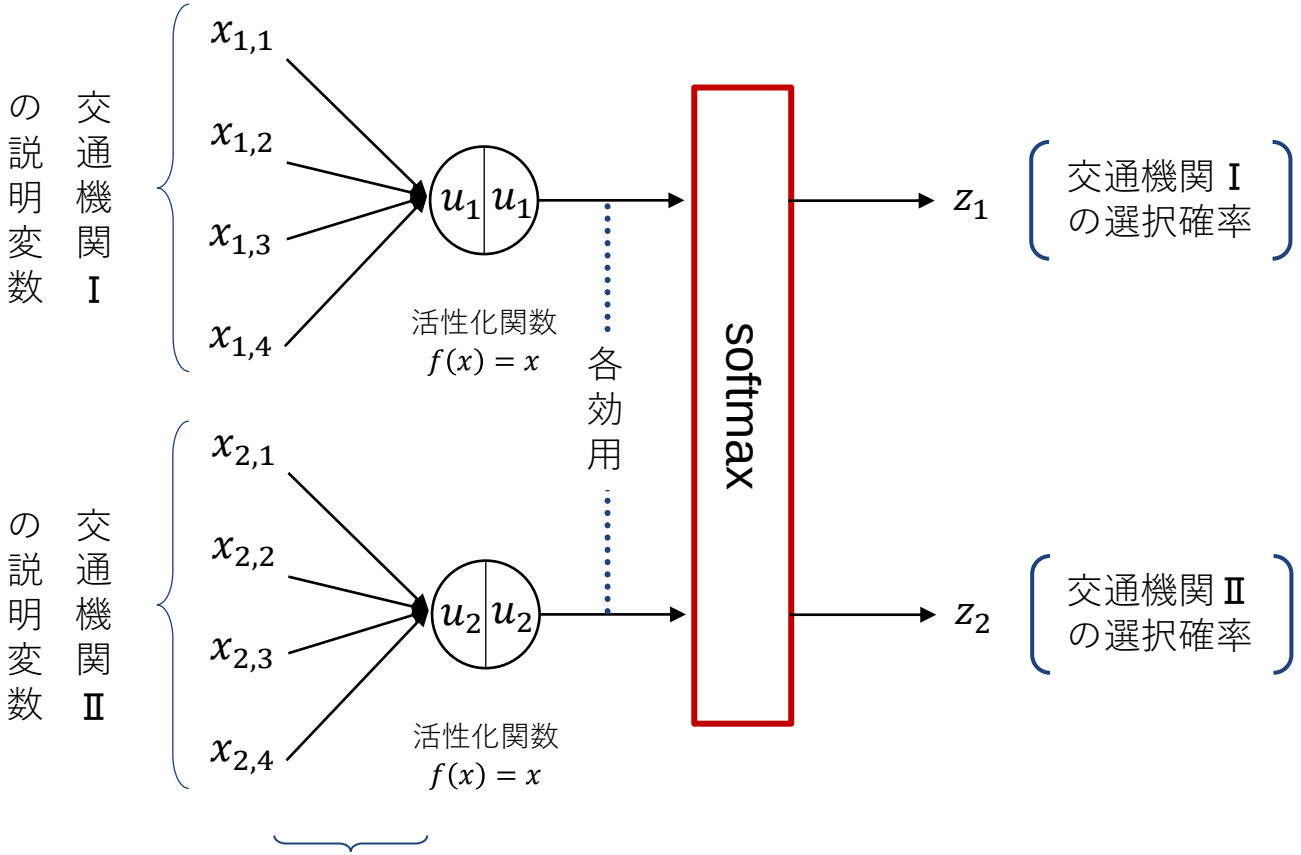
$$\mathbf{z} = \mathbf{f}(\mathbf{u})$$

$$\mathbf{W} = \begin{bmatrix} w_{11} \cdots w_{1I} \\ \vdots \ \ddots \ \vdots \\ w_{J1} \cdots w_{JI} \end{bmatrix}, \quad \mathbf{f}(\mathbf{u}) = \begin{bmatrix} f(u_1) \\ \vdots \\ f(u_J) \end{bmatrix}$$

$$\mathbf{u} = \begin{bmatrix} u_1 \\ \vdots \\ u_J \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_I \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} b_1 \\ \vdots \\ b_J \end{bmatrix}, \quad \mathbf{z} = \begin{bmatrix} z_1 \\ \vdots \\ z_J \end{bmatrix}$$

■イメージ

離散選択モデルはどう表せる？(交通機関選択の場合)



■損失関数：訓練データとの近さを表す尺度

出力の数によって異なり，訓練データの統計的性質に応じて選ぶのが理想.
以下に代表的なものを挙げる.

1. 回帰：主に出力が連続値をとる関数が目的関数の場合

二乗誤差

$$E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N} \|\mathbf{d}_n - \boldsymbol{y}(\mathbf{x}_n; \mathbf{w})\|^2 \qquad (1)$$

2.多クラス分類：出力が複数次元の関数の場合

クロスエントロピー誤差

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{k=1}^{K} d_{nk} \log y_k(\mathbf{x}_n; \mathbf{w}) \qquad (2)$$

ソフトマックス関数を通す

$(\mathbf{x}_n, \mathbf{d}_n)$：訓練データの$n$番目のサンプル
$\mathbf{x}_n$：訓練データの入力
$\mathbf{d}_n$：訓練データの出力
$\boldsymbol{y}(\mathbf{x}_n; \mathbf{w})$：関数の出力
K：出力層の次元

■誤差逆伝播法

層が深くなる (多くなる) と，出力層で計算される誤差を入力層に近い層まで伝える必要がある

例) 1つの訓練サンプル$(\mathbf{x}_n, \mathbf{d}_n)$ に対する二乗誤差E_nを考える

$$E_n = \frac{1}{2}\|\boldsymbol{y}(\mathbf{x}_n) - \mathbf{d}_n\|^2$$

第l層の重みの1成分 $w_{ji}^{(l)}$ の更新のため，　E_nを$w_{ji}^{(l)}$で偏微分する

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = (\boldsymbol{y}(\mathbf{x}_n) - \mathbf{d}_n)^{\mathrm{T}} \frac{\partial y}{\partial w_{ji}^{(l)}} \quad (3)$$

$w_{ji}^{(l)}$は活性化関数の深い入れ子の中に現れる

$$\begin{aligned}\boldsymbol{y}(\boldsymbol{x}) &= \mathbf{f}(\boldsymbol{u}^{(L)}) \\ &= \mathbf{f}(\mathbf{W}^{(L)}\boldsymbol{z}^{(L-1)} + \mathbf{b}^{(L)}) \\ &= \mathbf{f}(\mathbf{W}^{(L)}\mathbf{f}(\mathbf{W}^{(L-1)}\boldsymbol{z}^{(L-2)} + \mathbf{b}^{(L-1)}) + \mathbf{b}^{(L)})\end{aligned}$$

L：モデルの層数 (第L層は最も出力に近い層)

微分の連鎖規則を繰り返し用いる必要がある
- プログラミングが面倒
- 計算量が大きい

誤差逆伝播法の出現

1. 順伝播型ネットワーク

■誤差逆伝播法

出力層から順に勾配を計算し，入力層まで伝播する手法 (向きが予測方向と逆)

第l層の重みの1成分 $w_{ji}^{(l)}$ に関する勾配を考える

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \frac{\partial E_n}{\partial u_j^{(l)}} \frac{\partial u_j^{(l)}}{\partial w_{ji}^{(l)}} \quad (4)$$

層間の誤差伝播式と
その誤差を用いた各重みに関する勾配を求める

◦ 誤差の伝播

$$\frac{\partial E_n}{\partial u_j^{(l)}} = \sum_k \frac{\partial E_n}{\partial u_k^{(l+1)}} \frac{\partial u_k^{(l+1)}}{\partial u_j^{(l)}} \quad (5)$$

$\delta_j^{(l)} \equiv \frac{\partial E_n}{\partial u_j^{(l)}}$ と置く

$$\delta_j^{(l)} = \sum_k \delta_k^{(l+1)} (w_{kj}^{(l+1)} f'(u_j^{(l)})) \quad (6)$$

◦ 勾配の計算

$$\frac{\partial u_j^{(l)}}{\partial w_{ji}^{(l)}} = z_i^{(l-1)} \quad (\because u_j^{(l)} = \sum_i w_{ji}^{(l)} z_i^{(l-1)}) \quad (7)$$

(4)に代入して

$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} z_i^{(l-1)} \quad (8)$$

1. 順伝播型ネットワーク

■誤差逆伝播法のイメージ

勾配の計算

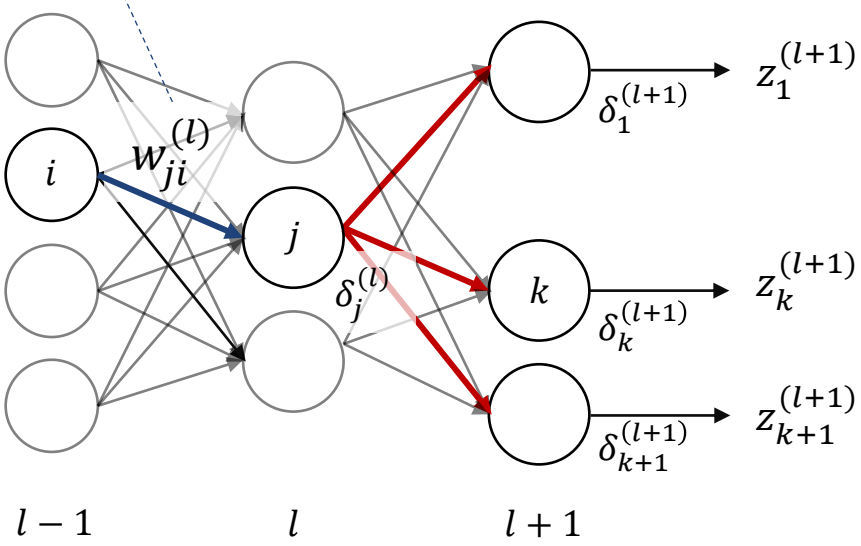
$$\frac{\partial E_n}{\partial w_{ji}^{(l)}} = \delta_j^{(l)} \, z_i^{(l-1)} \qquad (8)$$

再掲

$u_j^{(l)}$：l層jユニットが受ける入力の重みづけ和

$z_k^{(l)}$：l層kユニットの出力

f：活性化関数



誤差の伝播

$$\delta_j^{(l)} = \sum_k \delta_k^{(l+1)} (w_{kj}^{(l+1)} f'(u_j^{(l)})) \qquad (6)$$

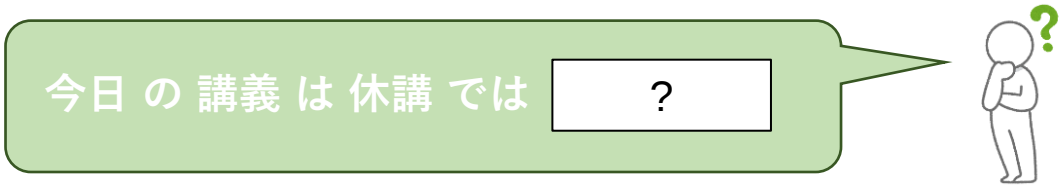
2. Recurrent Neural Network

■時系列データを扱う場合

音声・言語・軌跡といった順序が存在するデータを扱うことを考える

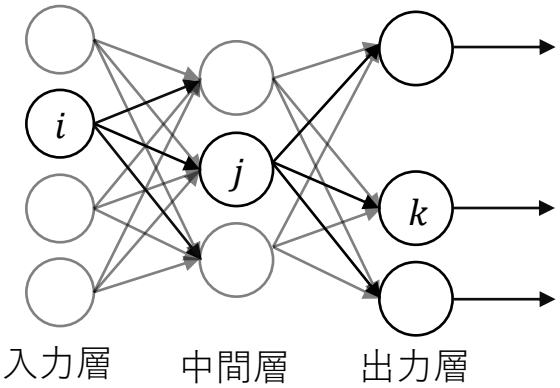
例)

・n番目までの単語から$n+1$番目の単語を予測する



◦前述の順伝播型ネットワークでは？

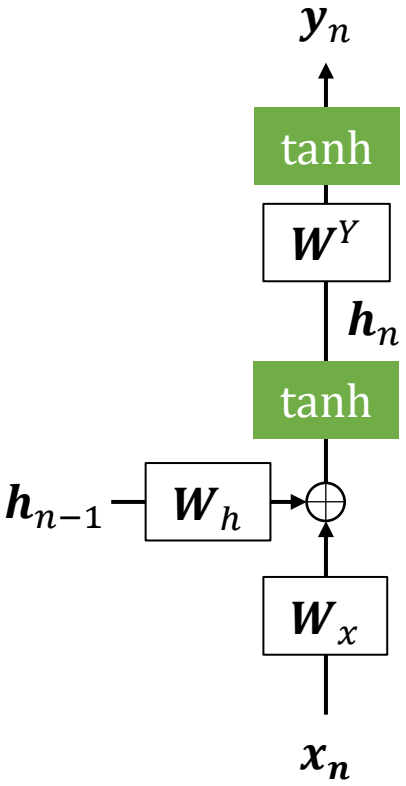
▶ 系列の前後間の関係を考慮するのが難しい



系列を考慮するような
ネットワークになっていない

■ Recurrent Neural Network (RNN) の構造

中間層に回帰結合を持つNeural Network



入力：ベクトル系列$x_1, x_2, \ldots, x_N$

出力：ベクトル系列$y_1, y_2, \ldots, y_N$

仮定：出力系列長と入力系列長が等しい

$$h_n = \tanh(W_h h_{n-1} + W_x x_n)$$

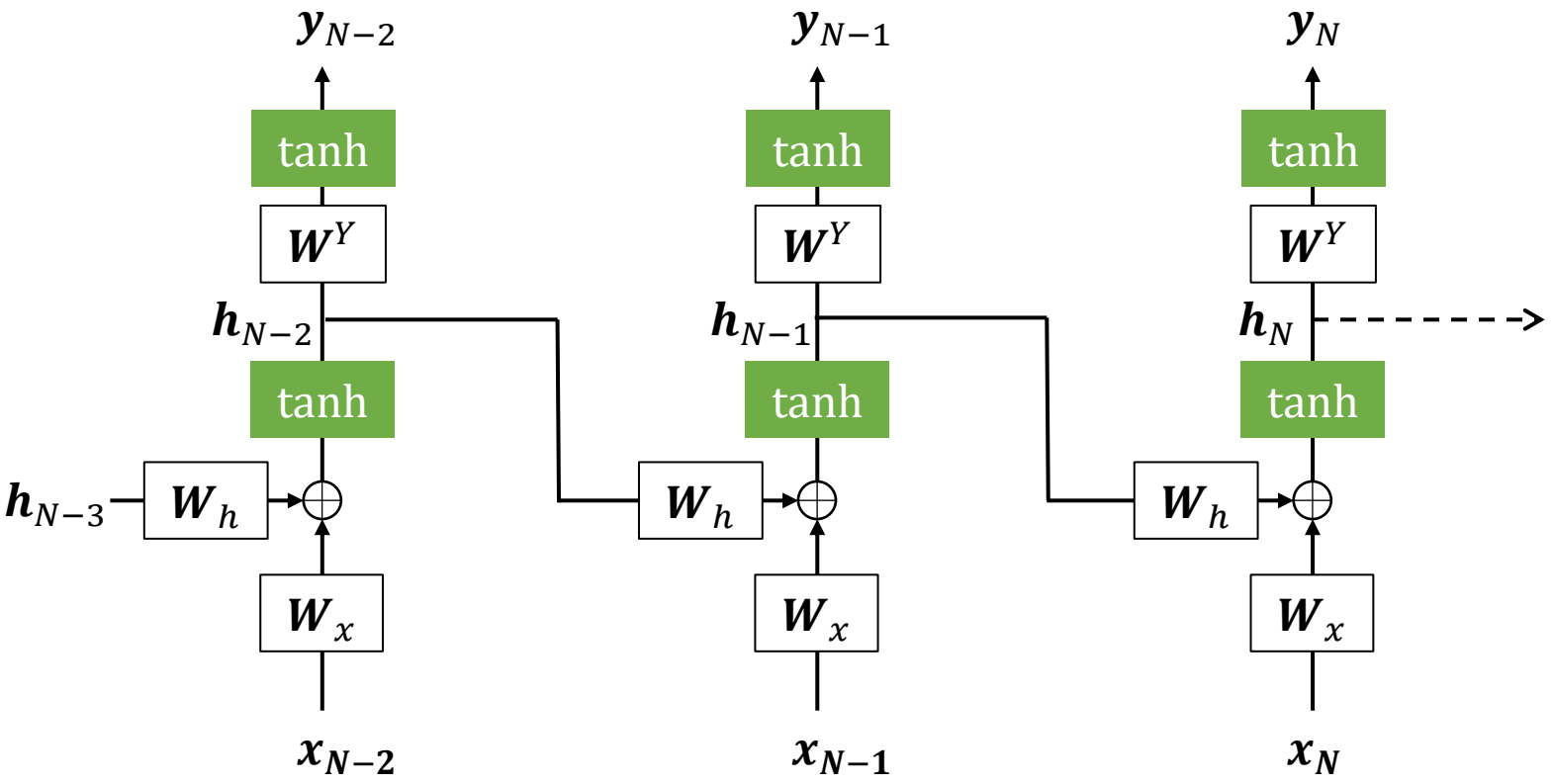
$$y_n = \tanh(W^Y h_n)$$

h_{n-1}：$n-1$番目の中間層の出力

W_x, W_h, W^Y：それぞれに対応する重みベクトル

■RNNの誤差逆伝播

前時期の中間層まで誤差を返す必要がある

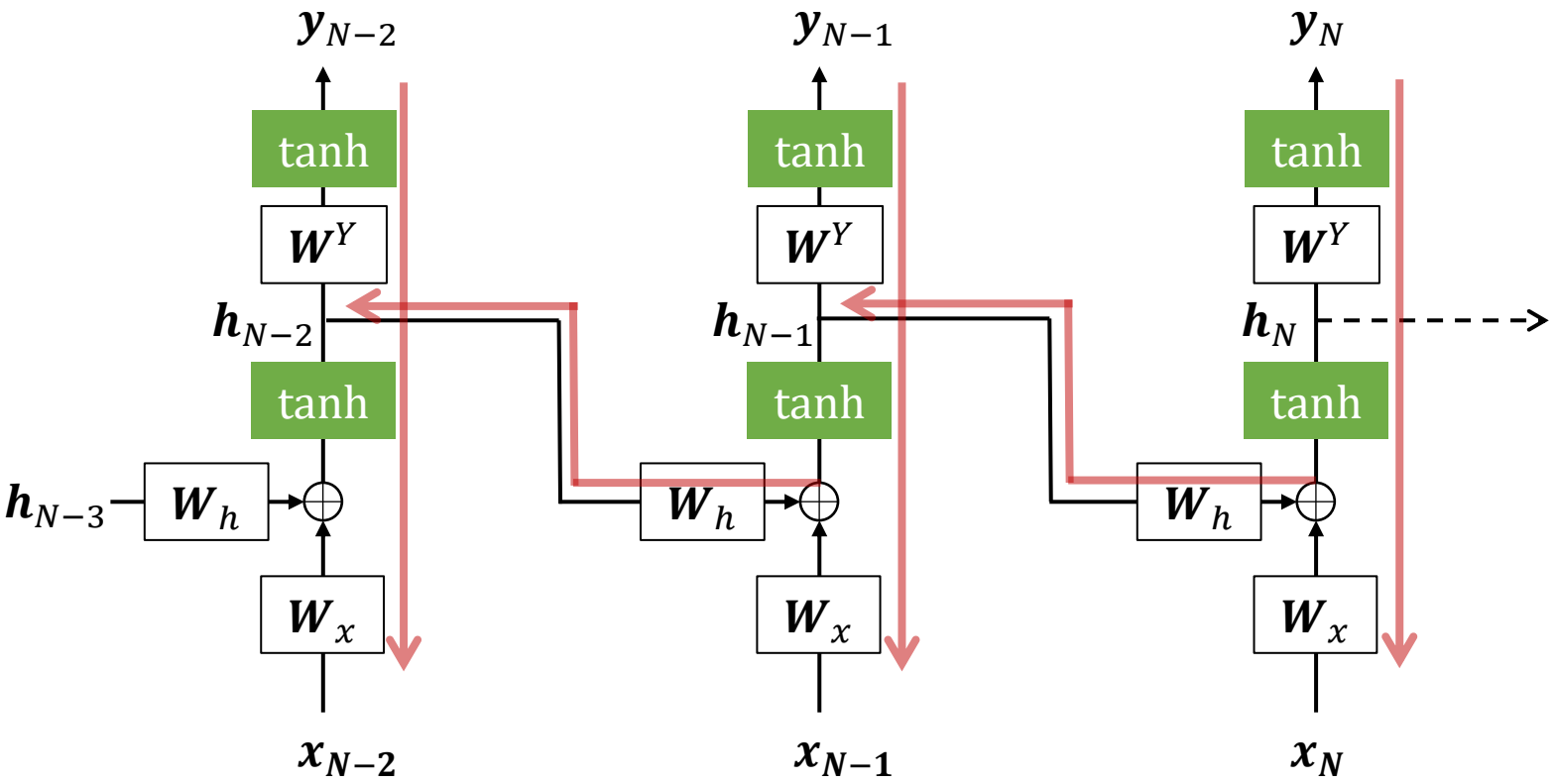


2. Recurrent Neural Network

■RNNの誤差逆伝播

Back Propagation Through Time (BPTT) [Williams & Zipser 1992, Werbos 1988]

$n = N$から始め誤差の合計を計算する



2. Recurrent Neural Network

■RNNの誤差逆伝播

Back Propagation Through Time (BPTT)

$n = N$から始め誤差の合計を計算する

それぞれの誤差信号は，誤差逆伝播法と同様
各ユニットの入力v_k^t, u_j^tによる偏微分で表せる

$$\delta_k^{out,t} \equiv \frac{\partial E}{\partial v_k^t}, \qquad \delta_j^t \equiv \frac{\partial E}{\partial u_j^t}$$

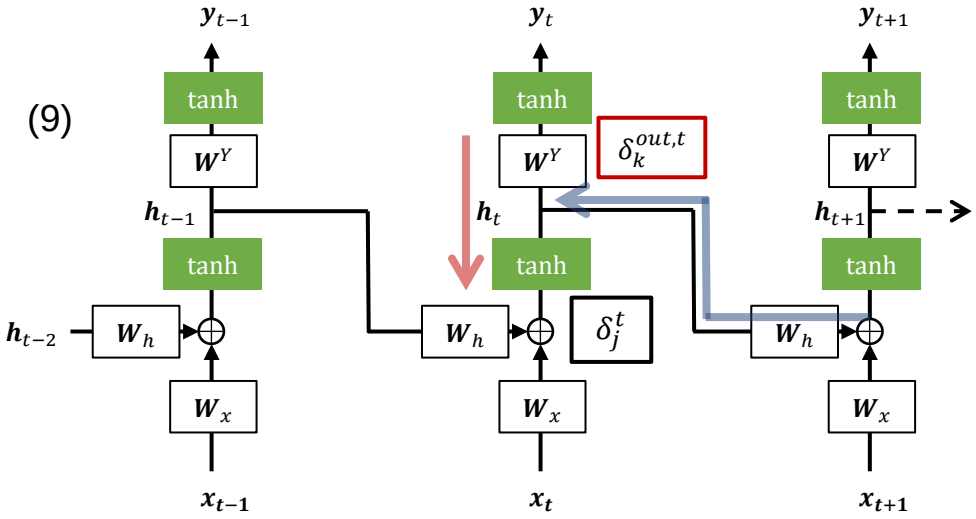
誤差信号の伝播は接続ユニットの和をとるので

$$\delta_j^t = \left(\underbrace{\sum_k w_{kj}^{out} \delta_k^{out,t}}_{\text{出力層由来}} + \underbrace{\sum_{j'} w_{jj'} \delta_{j'}^{t+1}}_{\text{一期先の中間層由来}} \right) f'(u_j^t) \qquad (9)$$

中間層の重み$w_{jj'}$の勾配は

$$\frac{\partial E}{\partial w_{jj'}} = \sum_{t=1}^{T} \frac{\partial E}{\partial u_j^t} \frac{\partial u_j^t}{\partial w_{jj'}} = \sum_{t=1}^{T} \delta_j^t z_j^{t-1} \qquad (10)$$

$\delta_k^{out,t}$:時刻tの出力層ユニットkの誤差信号
δ_j^t:時刻tの中間層ユニットjの誤差信号
$w_{jj'}$:中間層ユニットj, j'間の重み
w_{kj}^{out}:出力層ユニットkと中間層ユニットj間の重み
z_j^{t-1}:時刻$t-1$の中間層ユニットjの出力



2. Recurrent Neural Network

■BPTTの課題

時刻を遡るにつれて勾配の値が爆発的に増加 or 消滅しやすい性質がある

今，BPTTの時刻方向の誤差伝播のみを考える

$$\delta_j^t = f'(u_j^t)\sum_{j'} w_{jj'}\delta_{j'}^{t+1} \qquad (11)$$

時刻tにおける誤差信号δ_j^tの変化由来の，時刻$t-q$における誤差信号δ_v^{t-q}の変化率をみる

$$\frac{\partial \delta_v^{t-q}}{\partial \delta_j^t} = \begin{cases} f'(u_v^{t-1})w_{jv} & q = 1 \\ f'(u_v^{t-q})\sum_{l=1}^{n}\frac{\partial \delta_v^{t-q+1}}{\partial \delta_j^t}w_{lv} & q > 1 \end{cases} \qquad (12)$$

$l_q = v, l_0 = j$とすると，(13)は以下のように書き換えることができる

$$\frac{\partial \delta_v^{t-q}}{\partial \delta_j^t} = \sum_{l_1=1}^{n}\cdots\sum_{l_{q-1}=1}^{n}\prod_{m=1}^{q} f'(u_v^{t-m})w_{l_m l_{m-1}} \qquad (13)$$

2. Recurrent Neural Network

■BPTTの課題

$$\frac{\partial \delta_v^{t-q}}{\partial \delta_j^t} = \sum_{l_1=1}^{n} \cdots \sum_{l_{q-1}=1}^{n} \boxed{\prod_{m=1}^{q} f'(u_v^{t-m}) w_{l_m l_{m-1}}} \qquad (13)$$

・$\left| f'(u_v^{t-m}) w_{l_m l_{m-1}} \right| > 1.0$のとき

qが増大するにしたがって，$\frac{\partial \delta_v^{t-q}}{\partial \delta_j^t}$は指数関数的に増加する

・$\left| f'(u_v^{t-m}) w_{l_m l_{m-1}} \right| < 1.0$のとき

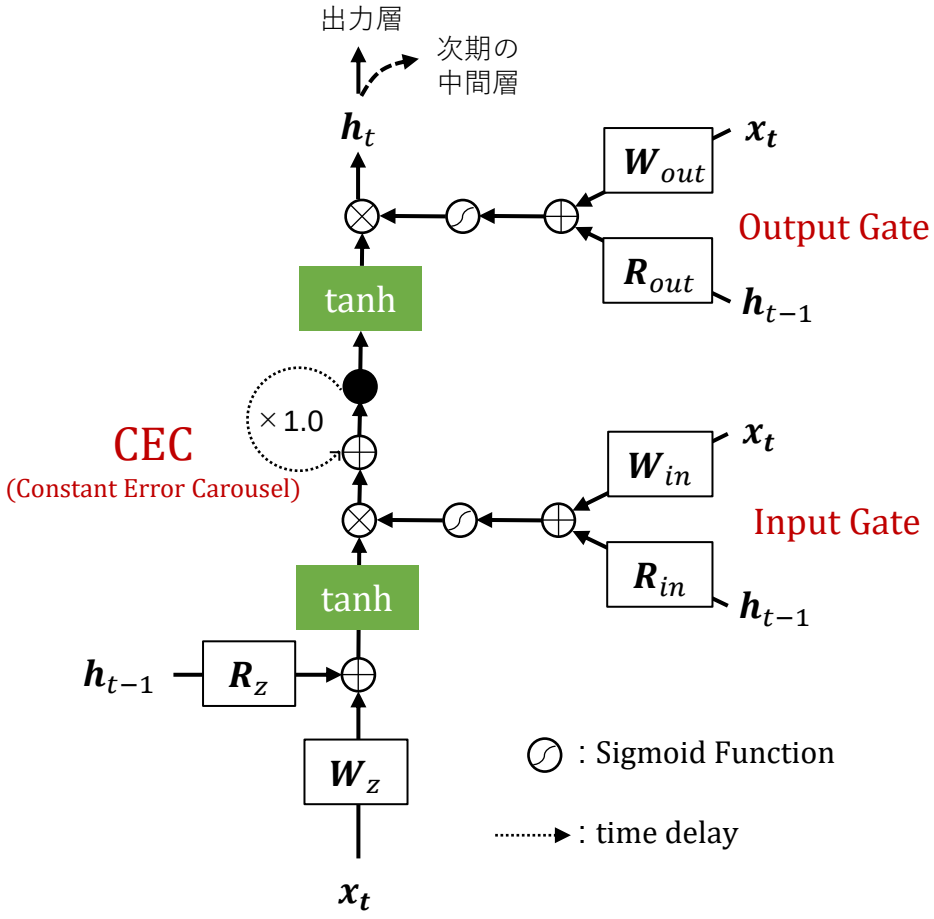
qが増大するにしたがって，$\frac{\partial \delta_v^{t-q}}{\partial \delta_j^t}$は指数関数的に0に近づく

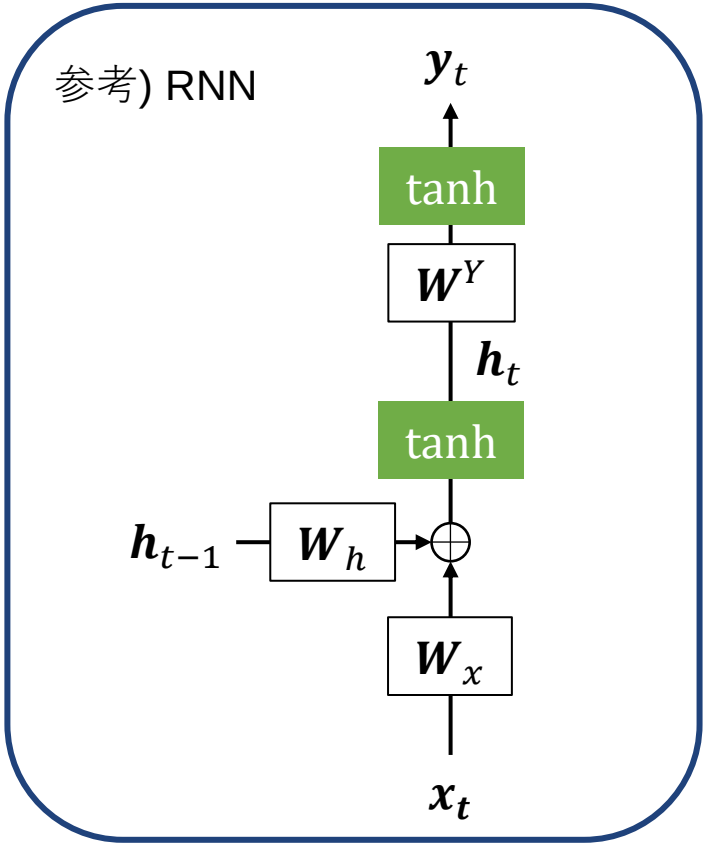
$f'(u_v^{t-m}) w_{l_m l_{m-1}} = 1.0$で誤差伝播ができれば，安定した学習になる

RNNで出力に反映できる過去履歴は高々10時刻程度といわれている

■ Long Short-Term Memory (LSTM) の構造

RNNの課題を解決し，長期記憶を実現できるよう構造を工夫





■ Long Short-Term Memory (LSTM) の構造

・Constant Error Carousel (CEC)

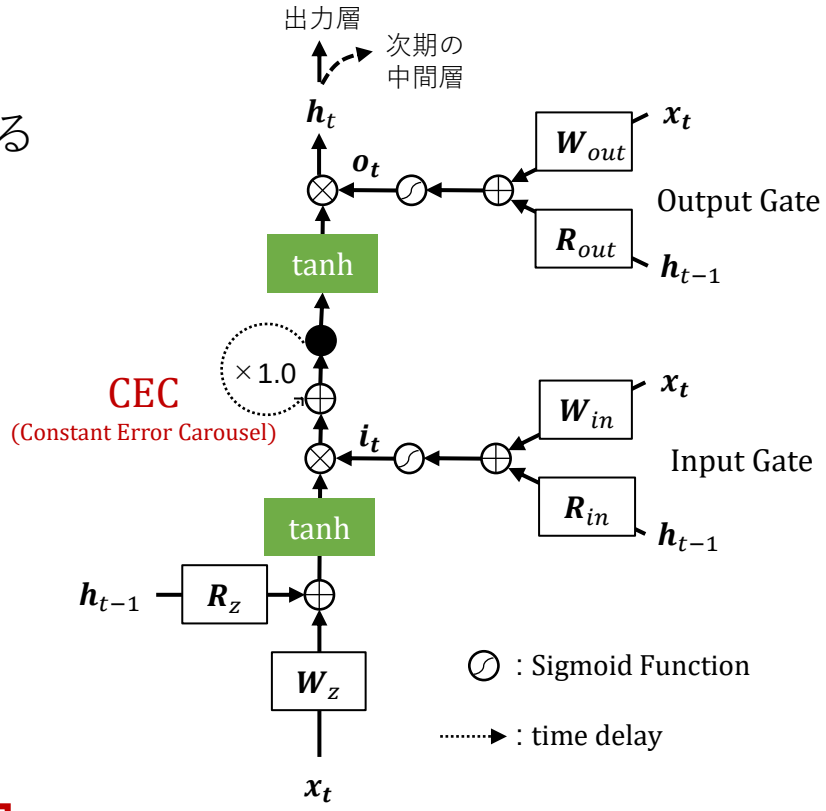
$f'(u_v^t)w_{jj} = 1.0$を満たすようにf, w_{jj}を決める

両辺u_v^tで積分

$$f(u_v^t) = \frac{u_v^t}{w_{jj}} \qquad (14)$$

よってユニットjの出力は,

$$y_j^{t+1} = f(u_v^{t+1}) = f\left(w_{jj}y_j^t\right) = y_j^t \qquad (15)$$



常に重み1, $f(x) = x$で時期の入力へ伝播

■ Long Short-Term Memory (LSTM) の構造

・CECが引き起こす問題

時系列データ$\{s_1, a_1, a_2, a_3, a_4, s_2\}$を予測する.
ここでs同士, a同士が関連を持つ場合を考える.

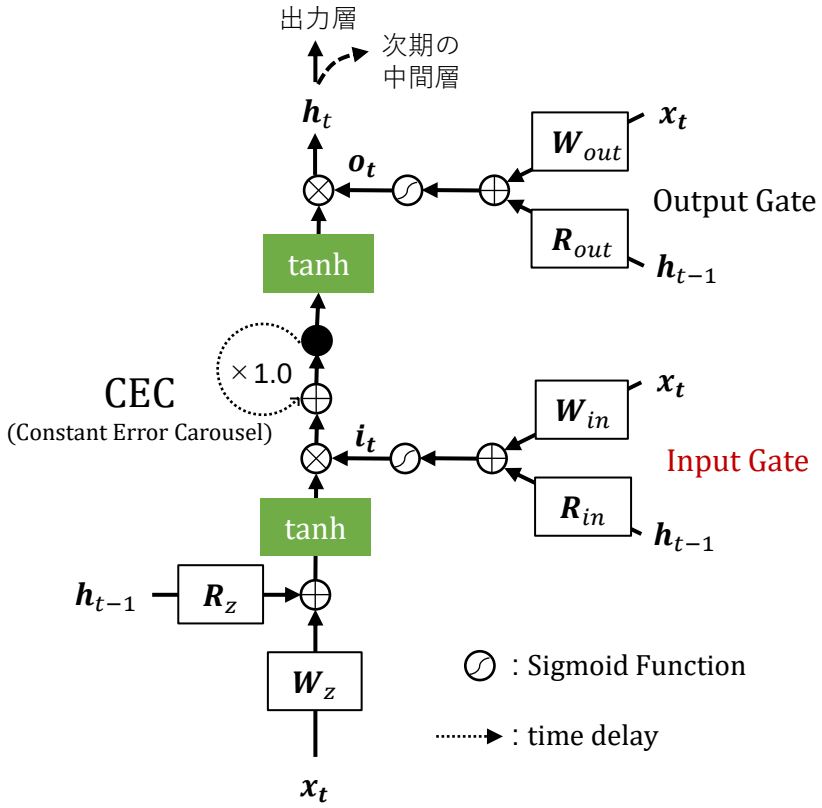
1. Input weight conflict

①入力を保持する

s_2を予測するためにs_1の情報を用いたい.
よって入力を受け入れたい

②入力を防ぐ

s_2を予測するためにa_1, a_2, a_3, a_4の情報は必要ない. よって入力を妨げたい.



シグモイド関数により各要素[0, 1]となるInput Gateの出力$\boldsymbol{i_t}$と入力の内積をとることで, 重み以外で入力可否を決定

■ Long Short-Term Memory (LSTM) の構造

・CECが引き起こす問題

時系列データ$\{s_1, a_1, a_2, a_3, a_4, s_2\}$を予測する.
ここでs同士, a同士が関連を持つ場合を考える.

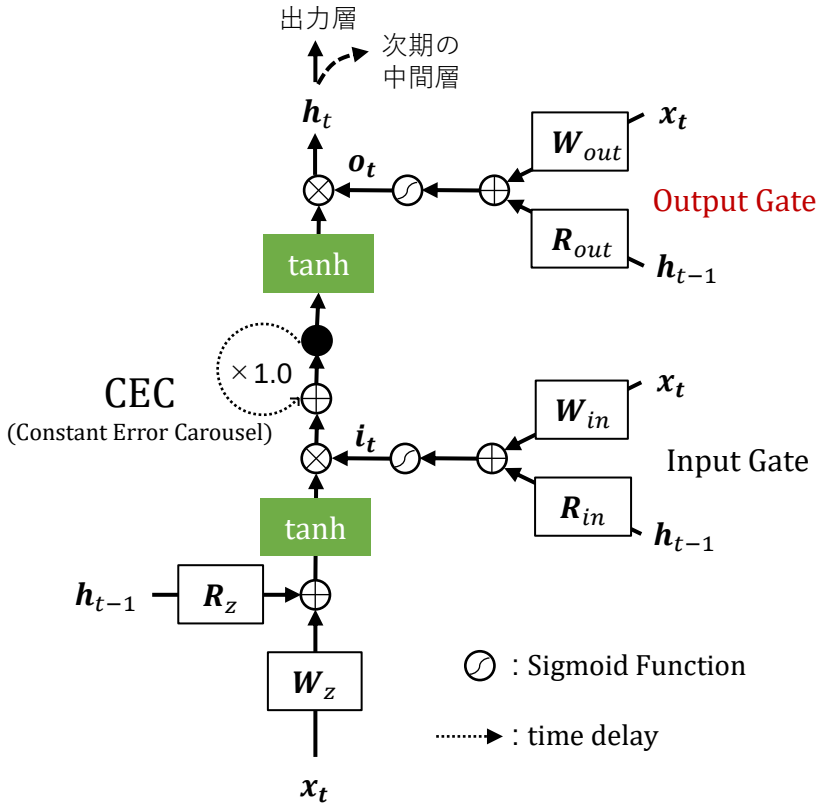
2. Output weight conflict

①出力を保持する

s_2を予測するためにs_1の情報を用いたい.
よって出力を受け入れたい

②出力を防ぐ

s_2を予測するためにa_1, a_2, a_3, a_4の情報は必要ない. よって出力を妨げたい.



シグモイド関数により各要素[0, 1]となるOutput Gateの出力o_tと入力の内積をとることで, 重み以外で入力可否を決定

■LSTMのその他の課題とその対処

・Abuse Problem

学習の初期段階では，前時期の情報を蓄えていなくても誤差は減少する

メモリセルを乱用されてしまう (例：バイアスを考慮するためのセルに充てられる)
⇒この場合，本来の用途へ学習を転換するのに時間がかかる

2つのメモリセルが同じ情報を記憶してしまう問題も現れる

解決策

(1) 誤差の減少が止まったとき，メモリセルとそれに対応するゲートを追加していく

(2) 出力ゲートを負の値で初期化する
⇒ゼロで初期化するより，メモリセルの学習に時間がかかるため，
序盤で別用途で学習されるのを防ぐ

■LSTMのその他の課題とその対処

・Initial state drift

CECでは活性化関数が線形

メモリセルへの入力がほとんど正 or ほとんど負 の場合，メモリセルの値が大きくなる
⇒中間状態が相対的に小さくなり，勾配が消失してしまう

解決策

学習初期入力ゲートの出力をゼロに近づくようfine-tuning

⇒他のユニットが学習を進めるのを待つ効果を持つ

■ Outline of experiments

1. **短期記憶用ベンチマークテスト**
   多くの論文で使用されており．既往手法でも成功を収めているテストでの確認

2. **長期記憶用データ，入力にノイズなし**
   長期記憶に対して，既往手法と精度の比較

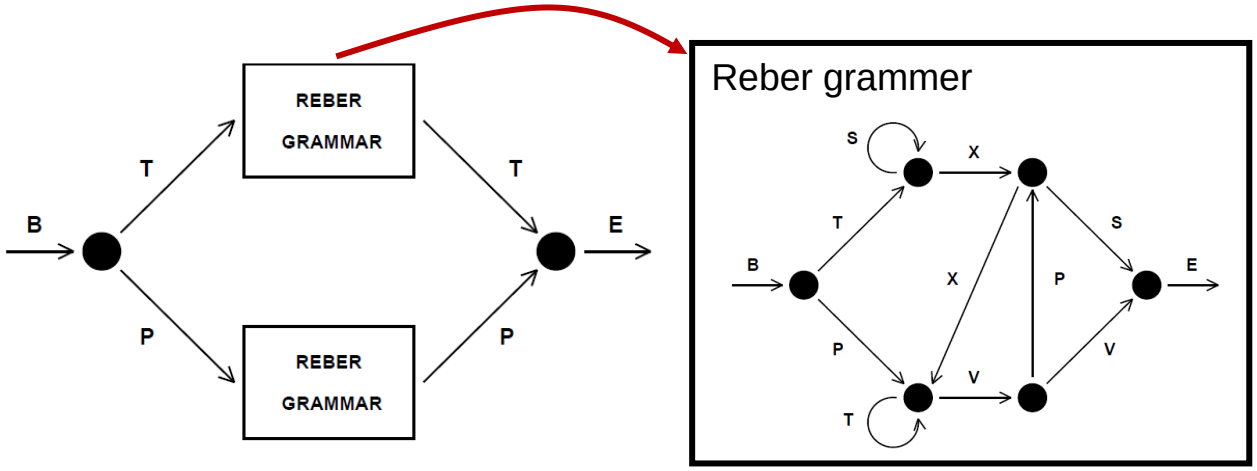
3. **長期記憶用データ，入力にノイズあり**
   2. でLSTMの優位性を確認し，ここではノイズへの頑健性を確認

4, 5. 入力が連続値
1~3までは入力がone-hot vector．ここでは連続値が入力の場合の精度を確認

6. **その他のリカレントアルゴリズムで解けなかったタスク**

1. 短期記憶用ベンチマークテスト

下記のグラフに従って状態を遷移するembedded Reber grammerでテストする



・リンク遷移確率は等確率
・遷移した先の文字を予測

・入出力
入出力層の次元：7 (文字に対応するone-hot vectorを入出力)

・比較する既往研究
ELM (Cleeremans et al. 1989), RCC (Fahlman 1991), RTRL (Smith and Zipser 1989)

1. 短期記憶用ベンチマークテスト

・比較

成功率 (試行：150)

成功するまでに必要なサンプル数

method	hidden units	# weights	learning rate	% of success	success after
RTRL	3	≈ 170	0.05	"some fraction"	173,000
RTRL	12	≈ 494	0.1	"some fraction"	25,000
ELM	15	≈ 435		0	>200,000
RCC	7-9	≈ 119-198		50	182,000
LSTM	4 blocks, size 1	264	0.1	100	39,740
LSTM	3 blocks, size 2	276	0.1	100	21,730
LSTM	3 blocks, size 2	276	0.2	97	14,060
LSTM	4 blocks, size 1	264	0.5	97	9,500
LSTM	3 blocks, size 2	276	0.5	100	8,440

・結果

-LSTMは成功率・学習速度共に既往手法より優位

-初期のT, Pに妨げられないよう学習する必要がある問題を解けている
⇒Output gateの重要性を確認

2. 長期記憶用データ，入力にノイズなし

以下の2つの系列を用意

$(y, a_1, a_2, \ldots, a_{p-1}, y)$

$(x, a_1, a_2, \ldots, a_{p-1}, x)$

a_i: i番目要素のみ1，それ以外0の$p+1$次元ベクトル

$x = a_p,\ y = a_{p+1}$

学習時，試行ごとにどちらかの系列を等確率で採択
末尾を予測するためには冒頭を記憶しておく必要がある

・入出力

入出力層の次元： $p+1$

・比較する既往研究

RTRL (Smith and Zipser 1989)
BPTT (Williams & Zipser 1992, Werbos 1988)
Neural Sequence Chunker (CH, Schmidhuber 1992)

2. 長期記憶用データ，入力にノイズなし

・比較

成功率

成功するまでに必要なサンプル数

Method	Delay p	Learning rate	# weights	% Successful trials	Success after
RTRL	4	1.0	36	78	1,043,000
RTRL	4	4.0	36	56	892,000
RTRL	4	10.0	36	22	254,000
RTRL	10	1.0-10.0	144	0	$>$ 5,000,000
RTRL	100	1.0-10.0	10404	0	$>$ 5,000,000
BPTT	100	1.0-10.0	10404	0	$>$ 5,000,000
CH	100	1.0	10506	33	32,400
LSTM	100	1.0	10504	100	5,040

・結果

-時間のラグが大きい場合，LSTMのみ高い精度で予測可能
-学習時間もLSTMが既往手法より非常に速い

短期記憶に規則性が存在しないデータでも実験を行ったが，
こちらもLSTMのみ高い精度を誇った

3. 長期記憶用データ，入力にノイズあり

以下の2つの系列を用意

$$(\underbrace{1.0, 1.0, \ldots, 1.0}_{N}, a_1, a_2, \ldots, a_{T-N}, 1.0)$$

$$(\underbrace{-1.0, -1.0, \ldots, -1.0}_{N}, a_1, a_2, \ldots, a_{T-N}, 0.0)$$

a_i: 平均0, 分散2.0のガウス分布より生成

$t \leq N$まで
class1は1.0, class2は-1.0を要素に持つ

$t > N$に存在するノイズが，LSTMにとって根本的な課題にならないことを示す

・入出力
入出力層の次元： 1

・比較対象
(T, N): $(100, 3), (100, 1), (1000, 3)$ の三通りについてLSTMの精度を検証

3. 長期記憶用データ，入力にノイズあり

・検証結果

出力の誤差の絶対値が0.2を上回った回数の
全体 (2560回) に対する百分率

T	N	stop: ST1	stop: ST2	# weights	ST2: fraction misclassified
100	3	27,380	39,850	102	0.000195
100	1	58,370	64,330	102	0.000117
1000	3	446,850	452,460	102	0.000078

・結果

-ノイズの系列が大きくなった場合でも推定精度に大きな影響を与えない
⇒ノイズへの頑健性が確認された

4. 入力が連続値の長時間ラグ問題

2つのComponentのペアで構成される2次元ベクトルを入力
Component2でマークされた値の合計値を予測する問題

- Component1：[-1, 1]からランダムで抽出される
- Component2：1.0, 0.0, -1.0のいずれか

➢ Component2の割り付けルール

最初から10番目までの要素からランダムで1ペアだけ1.0 (対応する値をX_1とする)
上記でマークされていない最初から$\frac{T}{2}-1$まで要素からランダムで1ペアのみ1.0
(対応する値をX_2とする)

それ以外は基本0.0
最初と最後のペアのみ-1.0

➢ 予測値

$$0.5+\frac{X_1+X_2}{4.0}$$

4. 入力が連続値の長時間ラグ問題

・入出力

入力層の次元： 2 (Component1, Component2)
出力層の次元： 1

誤差信号は系列末尾の出力のみ計算する

・検証結果

error > 0.04

T	minimal lag	# weights	# wrong predictions	Success after
100	50	93	1 out of 2560	74,000
500	250	93	0 out of 2560	209,000
1000	500	93	1 out of 2560	853,000

・結果

-分散された系列にも機能すること
-連続値の計算も学習が可能なこと
が確認された

■ Conclusion

- RNNの課題であった誤差の消失，爆発問題をCECの導入により解決した
  ⇒これにより，長期ラグをもつ系列の予測も可能になった

- 入力ゲートと出力ゲートを導入することにより，CECの情報にアクセスするか否かを判断できるようになった
  ⇒これにより，長期と短期の予測の矛盾を解消できるようになった

■ Future work

- 実問題に適用し，LSTMの実践的な限界を発見すること
  - 時系列予測，作曲，音声処理など

Appendix

■LSTMの発展モデル

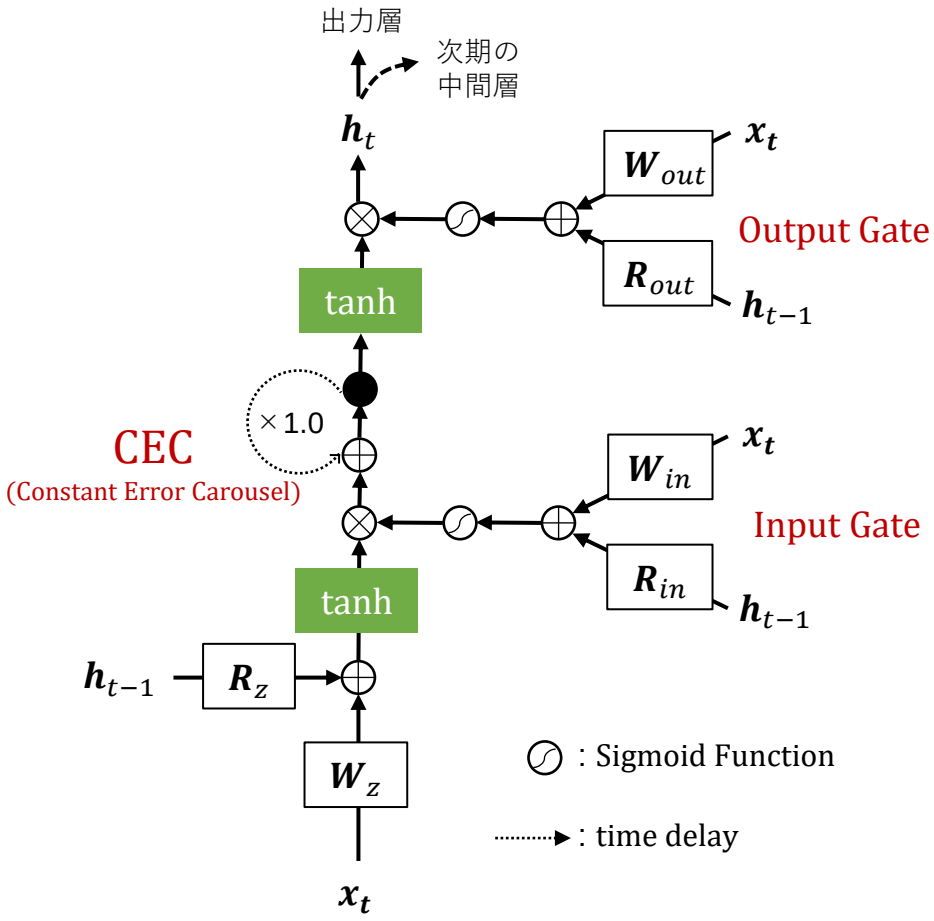
長期的な関係に周期がある場合

$\{s_1, a_1, a_2, a_3, a_4, s_2, m_1, a_5, a_6, a_7, a_8, m_2\}$

s, m間に関係はない

s_2が終わった時点で，CEC内にsの長期的依存に関する情報が必要ない

Forget gate (Gers, et al 1999)の導入

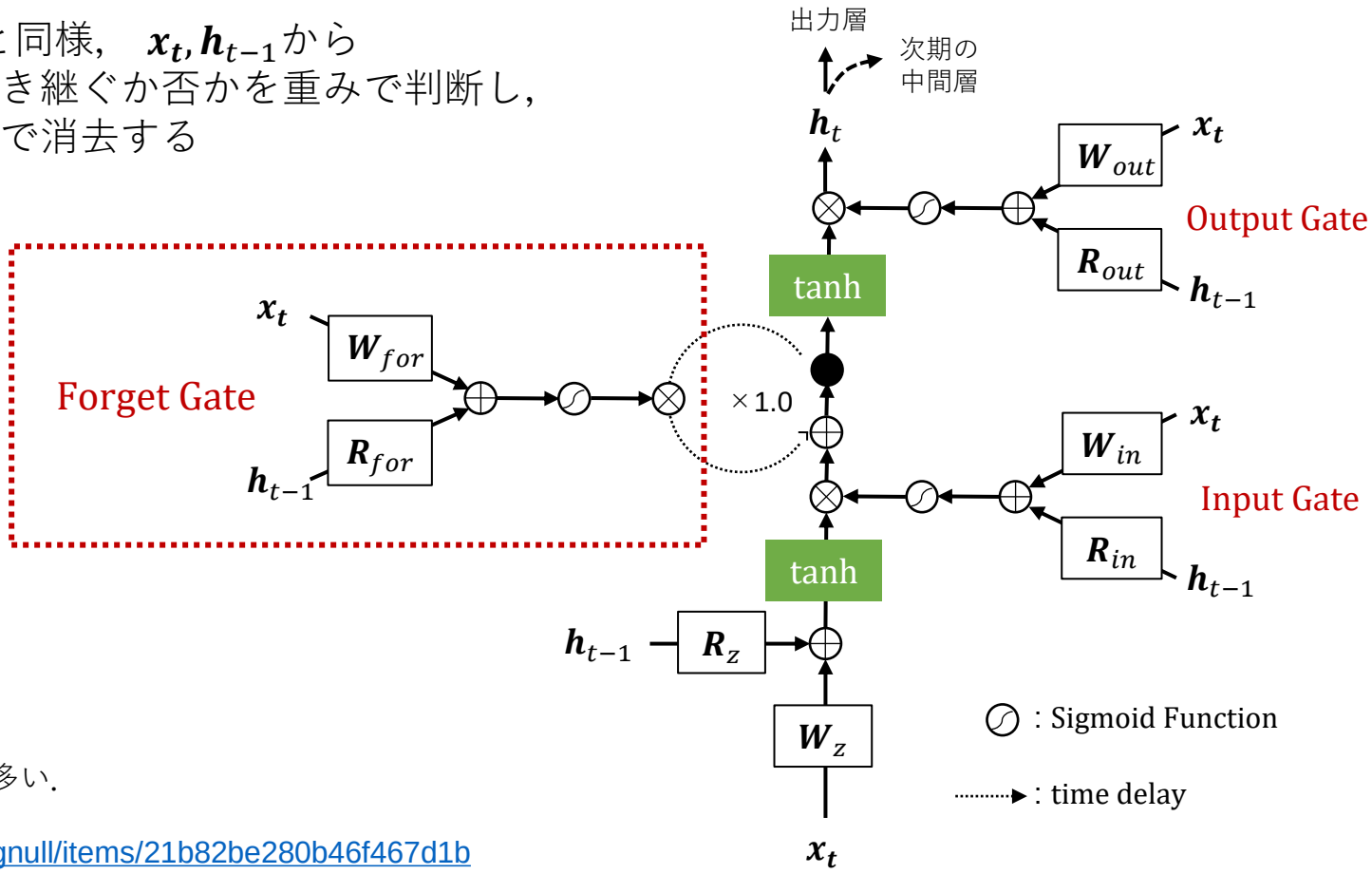


Appendix

■LSTMの発展モデル

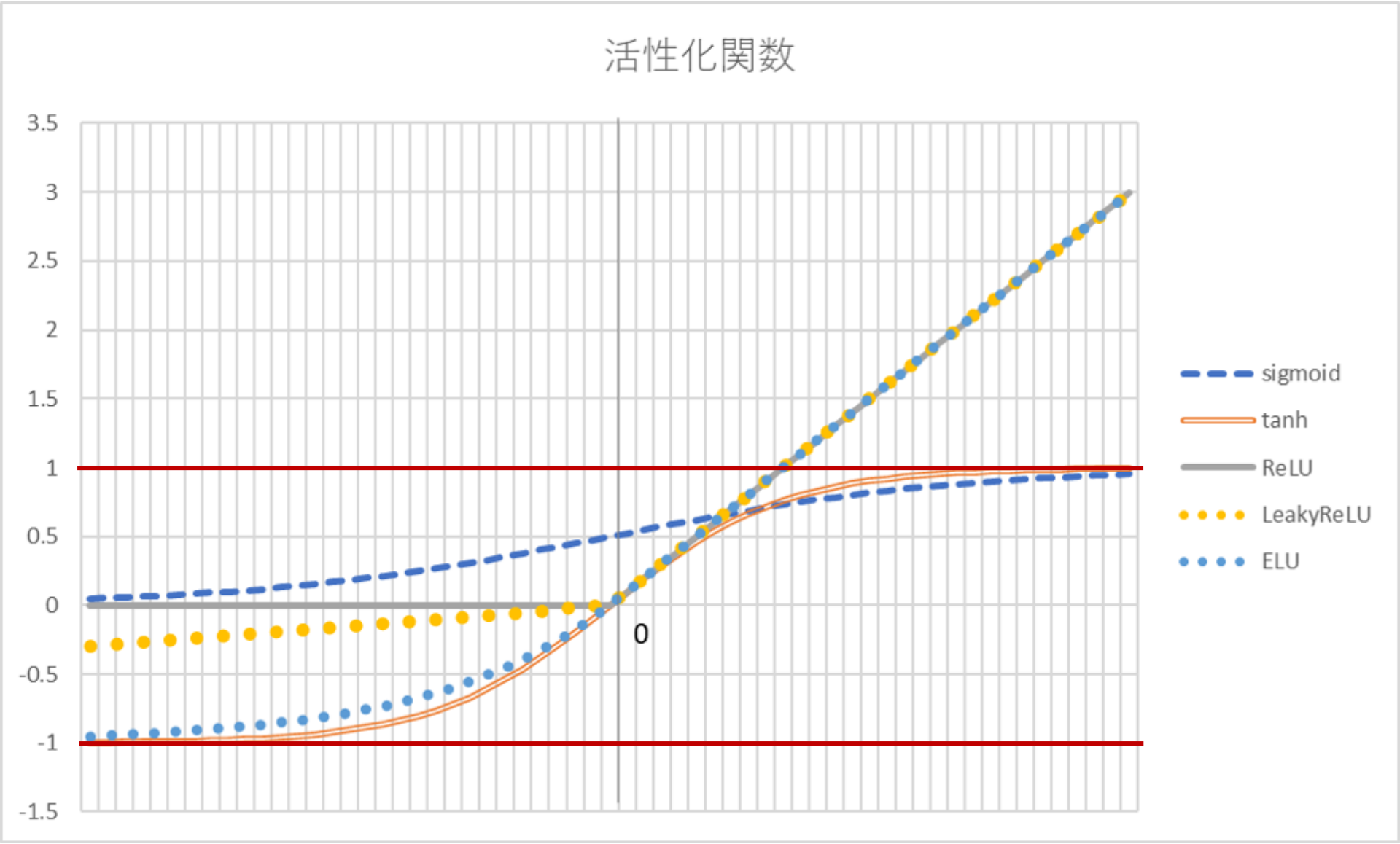
Forget gate (Gers, et al 1999)

Input gateなどと同様, $\boldsymbol{x_t}, \boldsymbol{h_{t-1}}$から
過去の情報を引き継ぐか否かを重みで判断し,
内積をとることで消去する



以降も発展モデルは多い.
発展は
https://qiita.com/t_Signull/items/21b82be280b46f467d1b
などが分かりやすい.

Appendix


活性化関数
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0
sigmoid
tanh
ReLU
LeakyReLU
ELU