

ニコニコAIスクール「脳型人工知能開発者入門コース」#10

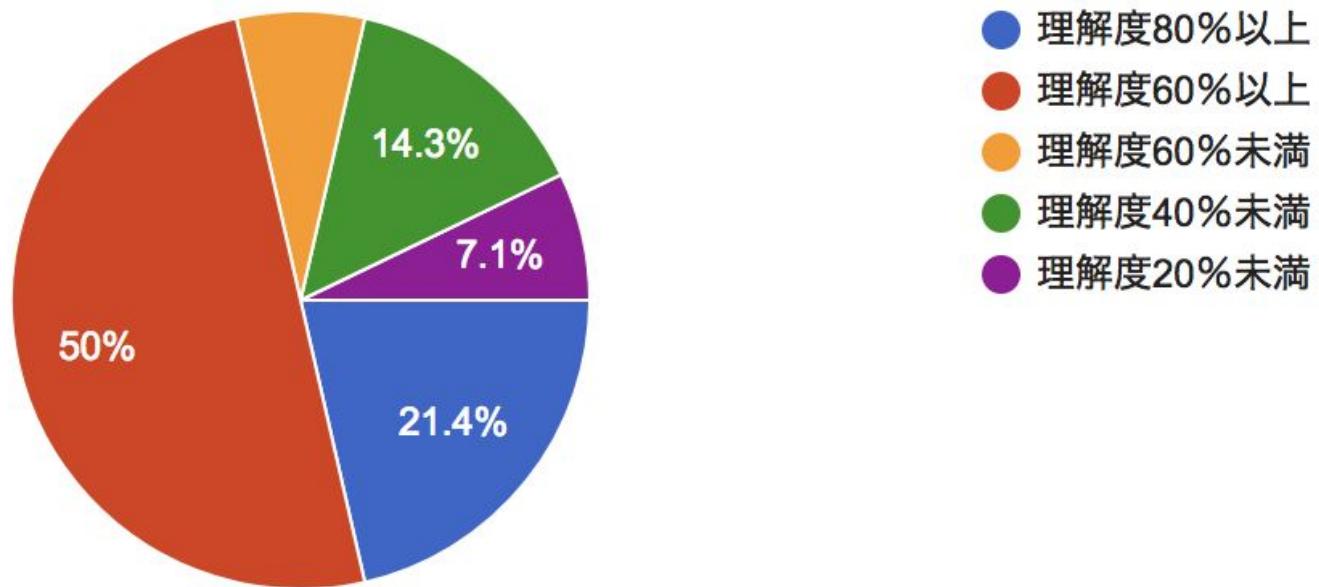
再帰的ニューラルネットワーク (2)

松森 匠哉

2018 / 03 / 17

前回のFeedback (1/8)

総合的な理解度



前回のFeedback (2/8)

アンケート: 運営改善点

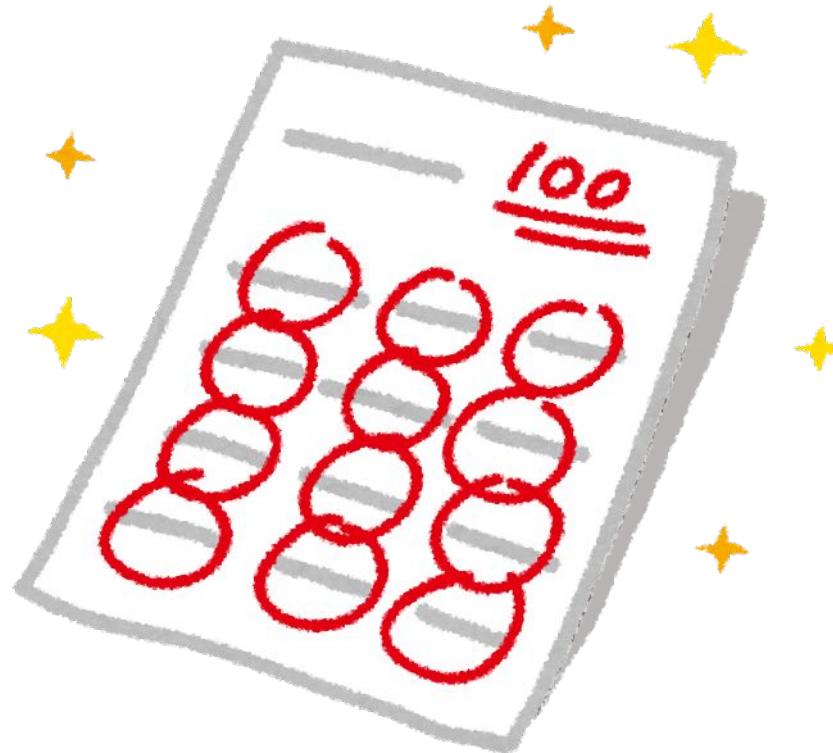
- 資料配布を早くしてほしい



前回のFeedback (3/8)

アンケート: 運営改善点

- 演習マスターのアップロードをしてほしい



前回のFeedback (4/8)

アンケート: 講義改善点

- 板書が画面上で行われていてよかったです



前回のFeedback (5/8)

アンケート: 講義改善点

- カタカナ語が多い



→ 仕様です。

前回のFeedback (6/8)

アンケート: 演習改善点

- Chainerの流儀が違って困った.
→ もっとwrapperを使いこなそう

__call__について

__call__はlossを返す. forwardingはpredict()とか作って中でloss()を呼び、lossの中でselfに保存しておいた値を使う感じ. その時にno_backprop_modeにするのを忘れないように.
あとは__call__にはforwardingを書いて, L.Classifierでwrapすることもできる.

前回のFeedback (7/8)

アンケート: 演習改善点

● CPUで動いてるの？GPUで動いてるの？

If we use `chainer.training.Trainer`, what we have to do is just let the updater know the device ID to send each mini-batch.

```
updater = training.StandardUpdater(train_iter, optimizer, device=0)
trainer = training.Trainer(updater, (20, 'epoch'), out='result')
```

We also have to specify the device ID for an evaluator extension as well.

```
trainer.extend(extensions.Evaluator(test_iter, model, device=0))
```

When we write down the training loop by hand, we have to transfer each mini-batch to the GPU manually:

```
model.to_gpu()
batchsize = 100
datasize = len(x_train)
for epoch in range(20):
    print('epoch %d' % epoch)
    indexes = np.random.permutation(datasize)
    for i in range(0, datasize, batchsize):
        x = Variable(cuda.to_gpu(x_train[indexes[i : i + batchsize]]))
        t = Variable(cuda.to_gpu(y_train[indexes[i : i + batchsize]]))
        optimizer.update(model, x, t)
```

前回のFeedback (8/8)

講義中の質問まとめ

- いちいちRNNに入れるんじゃなくて,
全時刻のデータをまとめてCNNとかに突っ込めば学習できる
んじゃないの？

→ できます。

ただ, CNNとかだと可変長列は基本的に扱えない。

最長の入力に揃え, それより短い系列の場合は**padding**
をして調整する必要がある。

(※ SPP-netなどの特殊な例を除き)

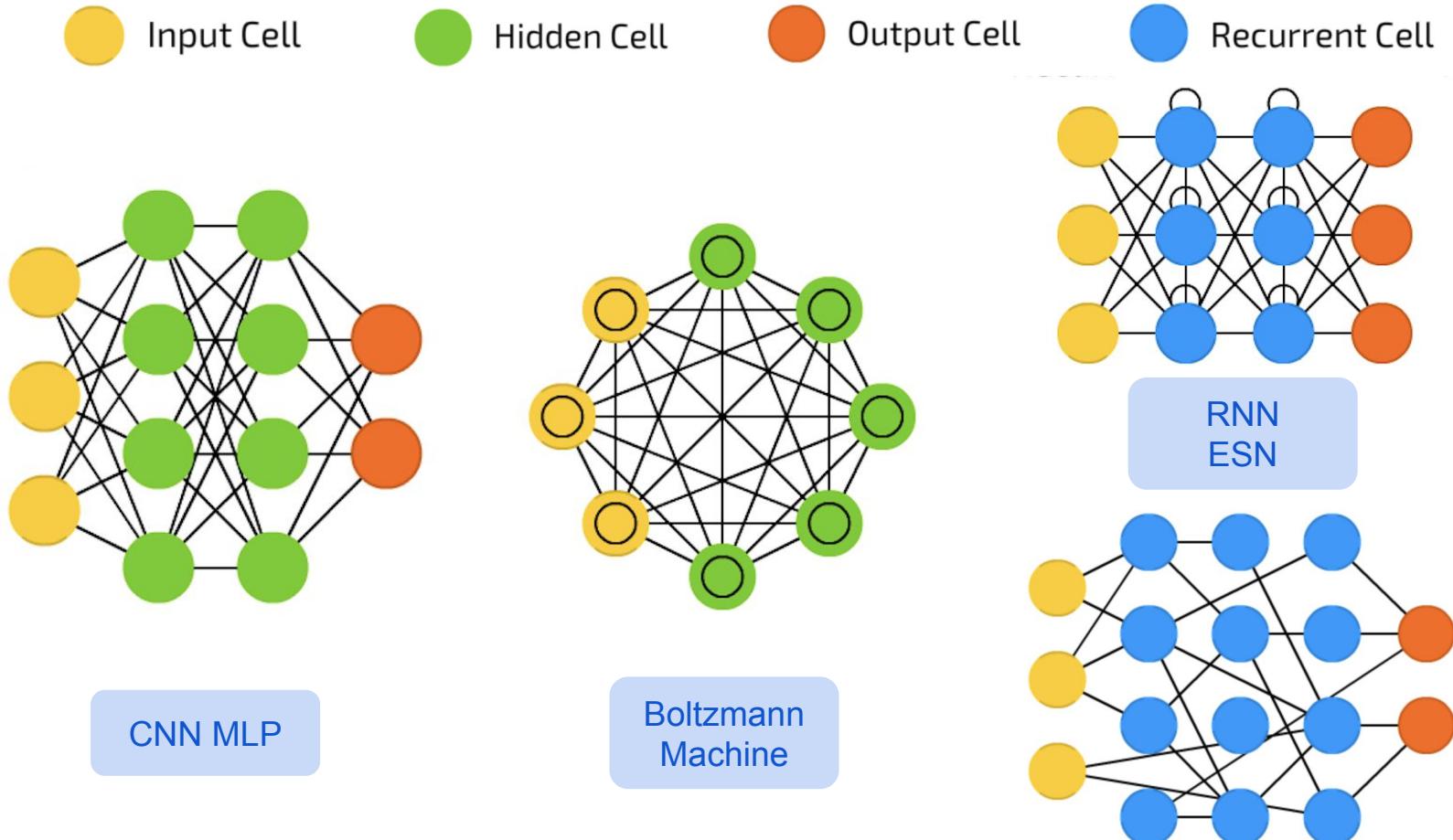
用語集

緑: 既知 青: 未習
として講義を進めます

- **BPTT**: Backpropagation Through Time
 - RNNの誤差逆伝播法
- **Gradient Vanishing (Exploding)**:
 - RNNの学習中に勾配が消失・爆発する問題
- **LSTM**: Long Short Term Memory
 - RNNの改良版、長期依存問題を解消
- **GRU**: Gated Recurrent Unit
 - LSTMをよりシンプルにしたRNN

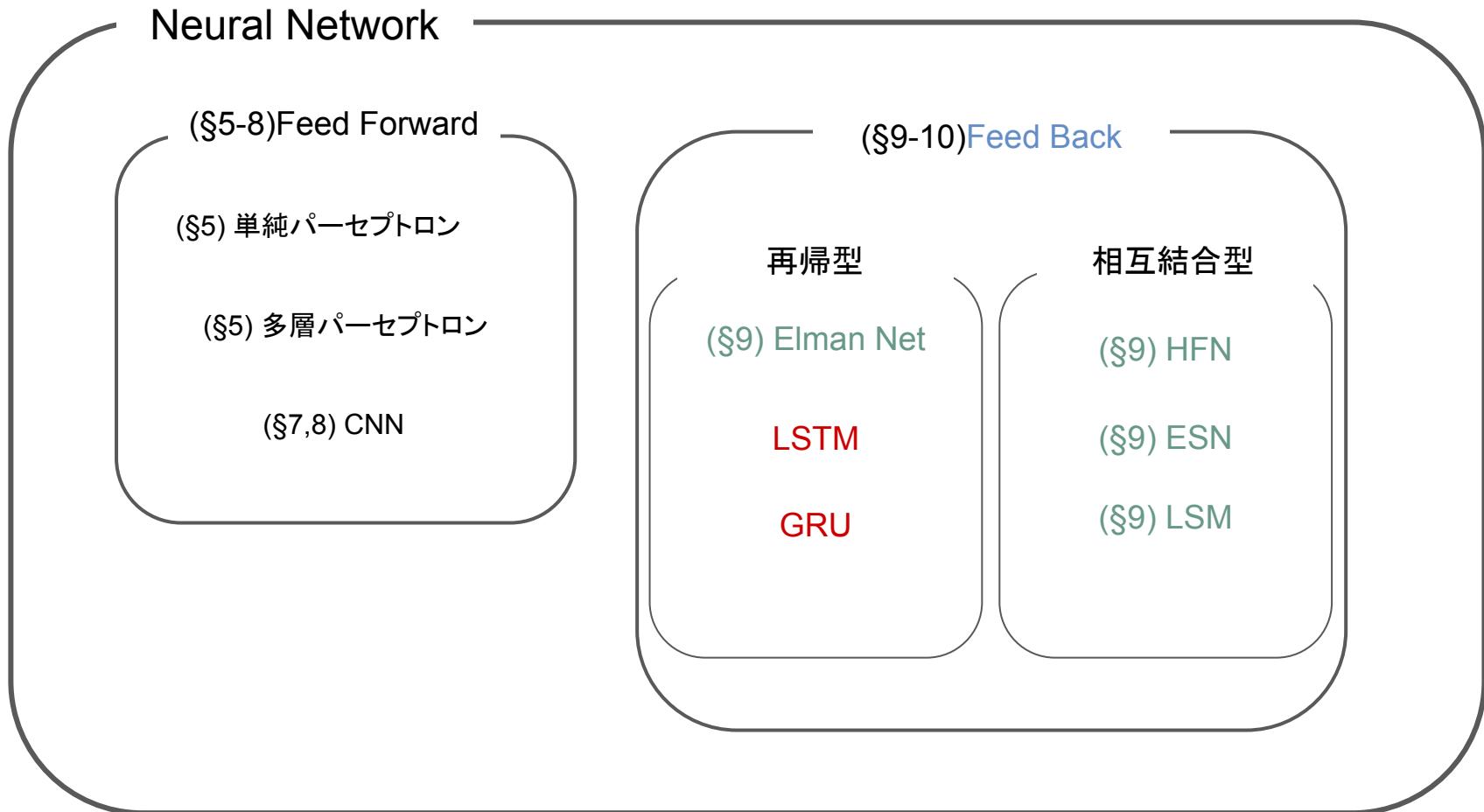
FF vs FB

様々なネットワークの形

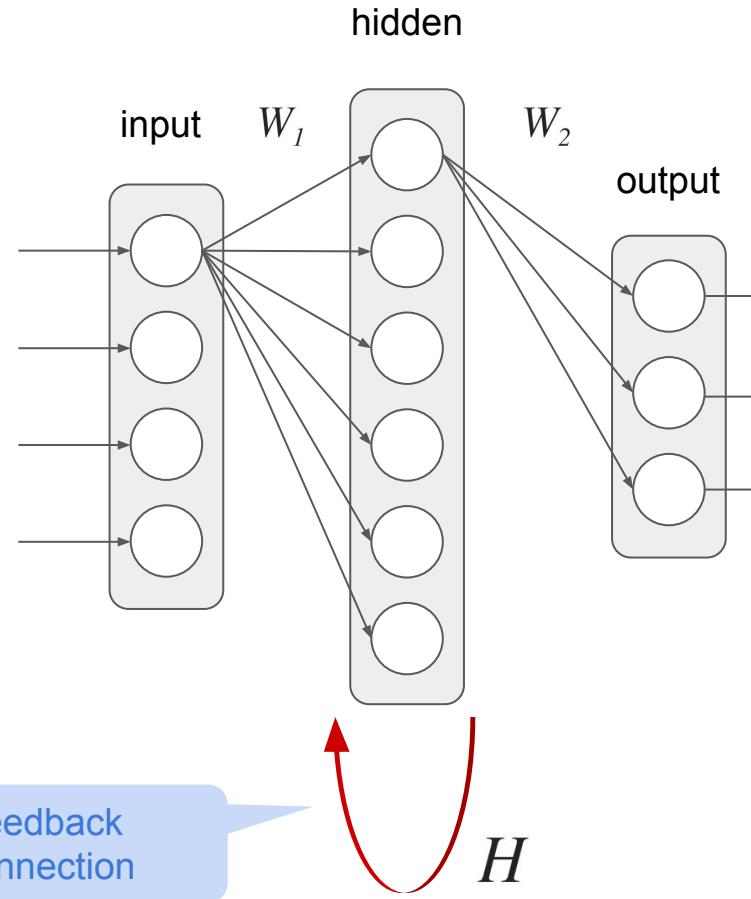


今回扱うNW

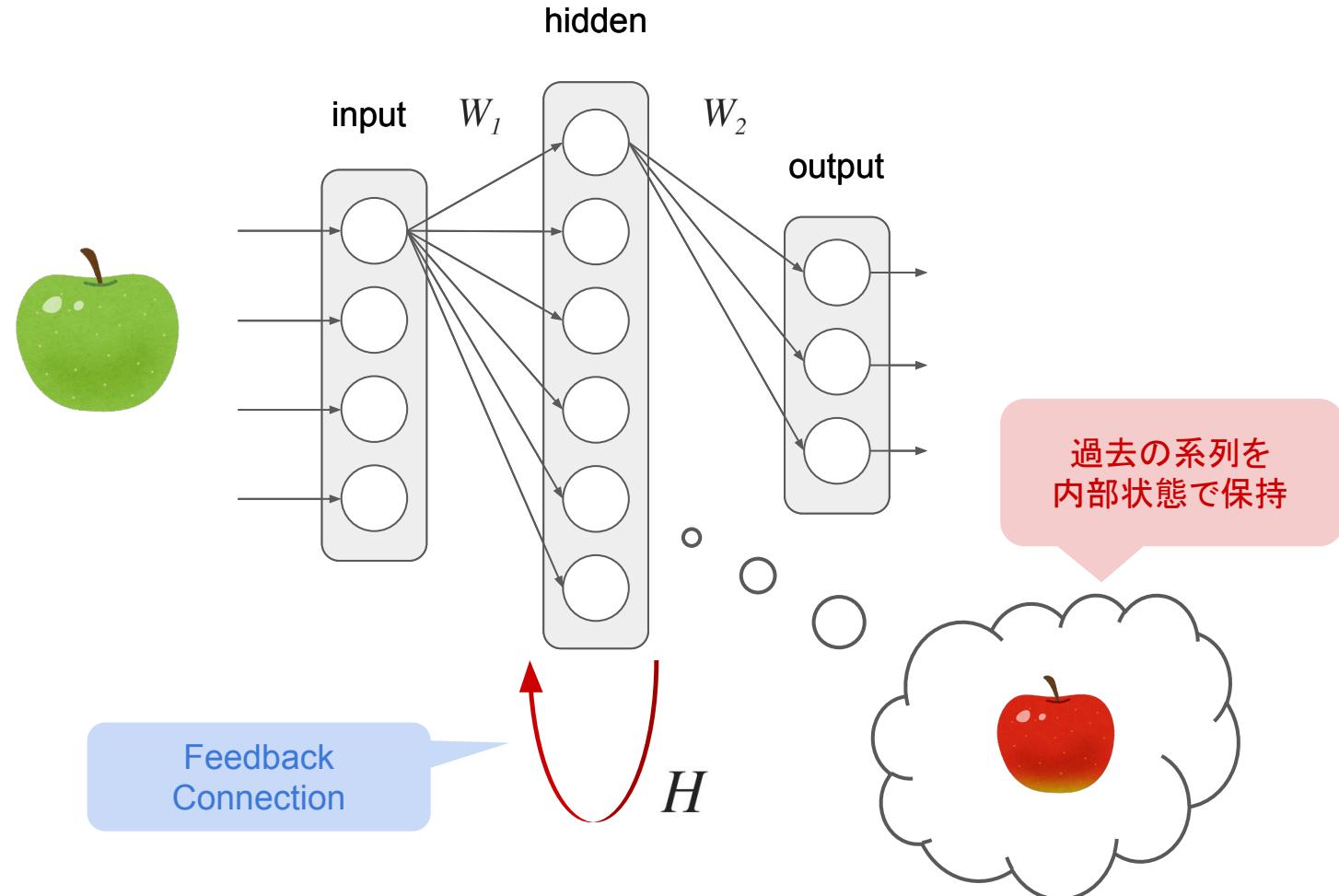
LSTM, GRUの位置付け



(復習) RNN (2/2)

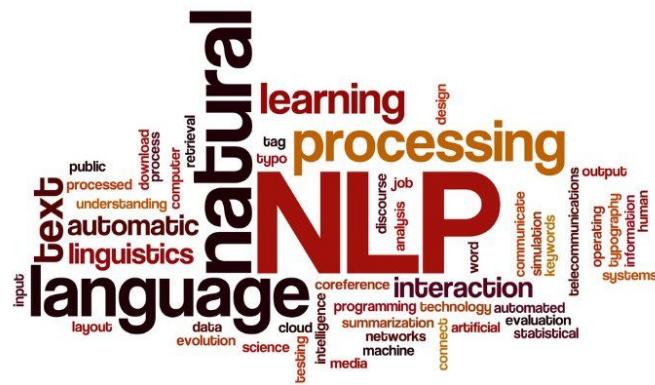


(復習) RNN (2/2)



時系列処理の問題設定

RNNの発明によって時系列データ $x = (x_1, x_2, \dots, x_n)$ を上手く学習することが出来るようになった



時系列データの例

Simple RNNの問題点

勾配消失・爆発問題

- 単純RNNは『勾配消失・爆発』の問題で、長期依存の時系列データを扱えないことが分かってきた
 - (c.f.) 第9回 練習問題1: MySimpleRNNの実装

長期依存データの例:

クイズ: ○○○には何が入りますか？

“さて、いったい何がラオスにあるというのか？良い質問だ。たぶん。でもそんなことを訊かれても、僕には答えようがない。だって、その何かを探すために、これから○○○まで行こうとしているわけなのだから。それがそもそも、旅行というものではないか。”
(2015, 村上春樹、ラオスにいったい何があるというんですか？)

Simple RNNの問題点

勾配消失・爆発問題

- 単純RNNは『勾配消失・爆発』の問題で、長期依存の時系列データを扱えないことが分かってきた
 - (c.f.) 第9回 練習問題1: MySimpleRNNの実装

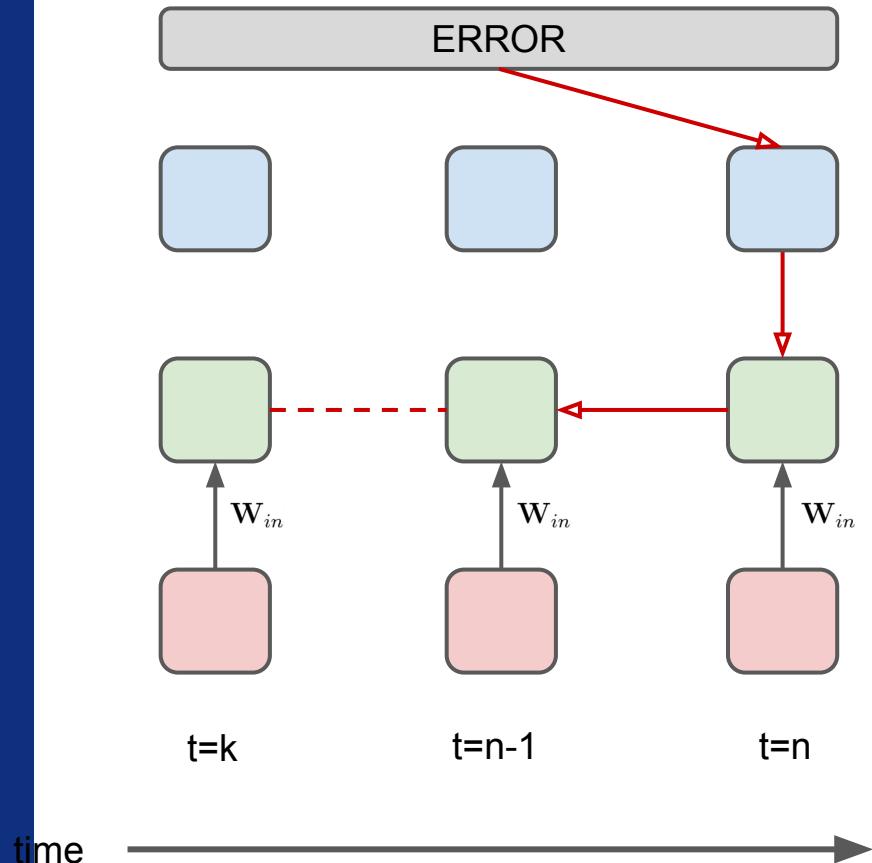
長期依存データの例:

正解: ラオス

“さて、いったい何が**ラオス**にあるというのか？良い質問だ。たぶん。でもそんなことを訊かれても、僕には答えようがない。だって、その何かを探すために、これから**ラオス**まで行こうとしているわけ

ラオスを推定するためにはかなり前に遡らなければならぬ。

(復習) BPTTによる学習: 勾配爆発・消失



◇ 誤差逆伝搬

$t=n$ の場合

□ C_n の偏微分計算 (まとめ)

$$\frac{\partial C^{(n)}}{\partial \theta} = \frac{\partial C^{(n)}}{\partial \mathbf{y}^{(n)}} \frac{\partial \mathbf{y}^{(n)}}{\partial \mathbf{h}^{(n)}} \left(\prod_{k < i \leq n} \Phi_H^{(i)} W_{\text{rec}}^T \right) \frac{\partial \mathbf{h}^{(k)}}{\partial \theta}$$

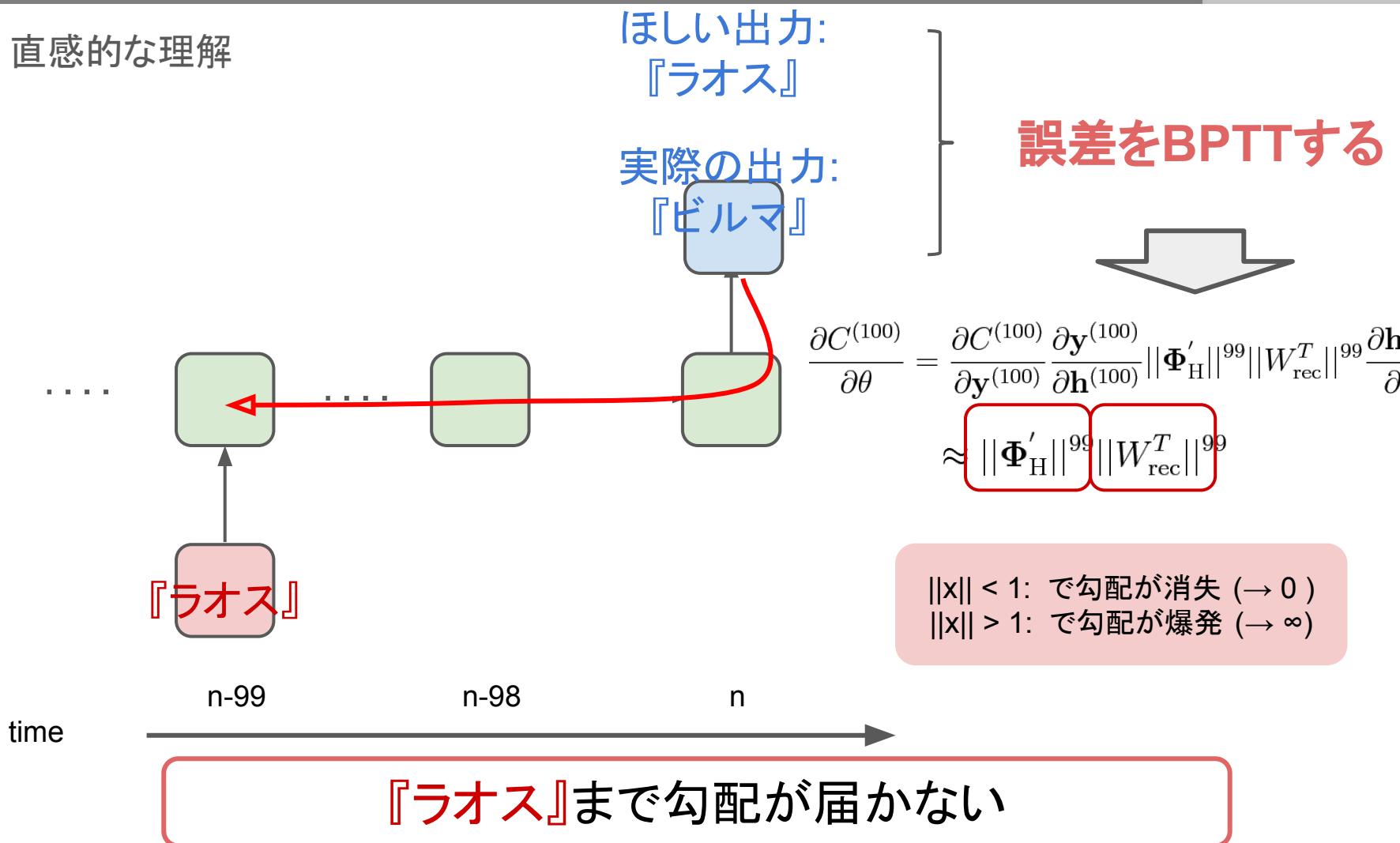
$$\approx \|\Phi_H'\|^{n-k} \|W_{\text{rec}}^T\|^{n-k}$$

$\|\mathbf{x}\| < 1$: で勾配が消失 ($\rightarrow 0$)
 $\|\mathbf{x}\| > 1$: で勾配が爆発 ($\rightarrow \infty$)

勾配爆発・消失の原因

(復習) 勾配爆発・消失の例

直感的な理解



講義 Part1

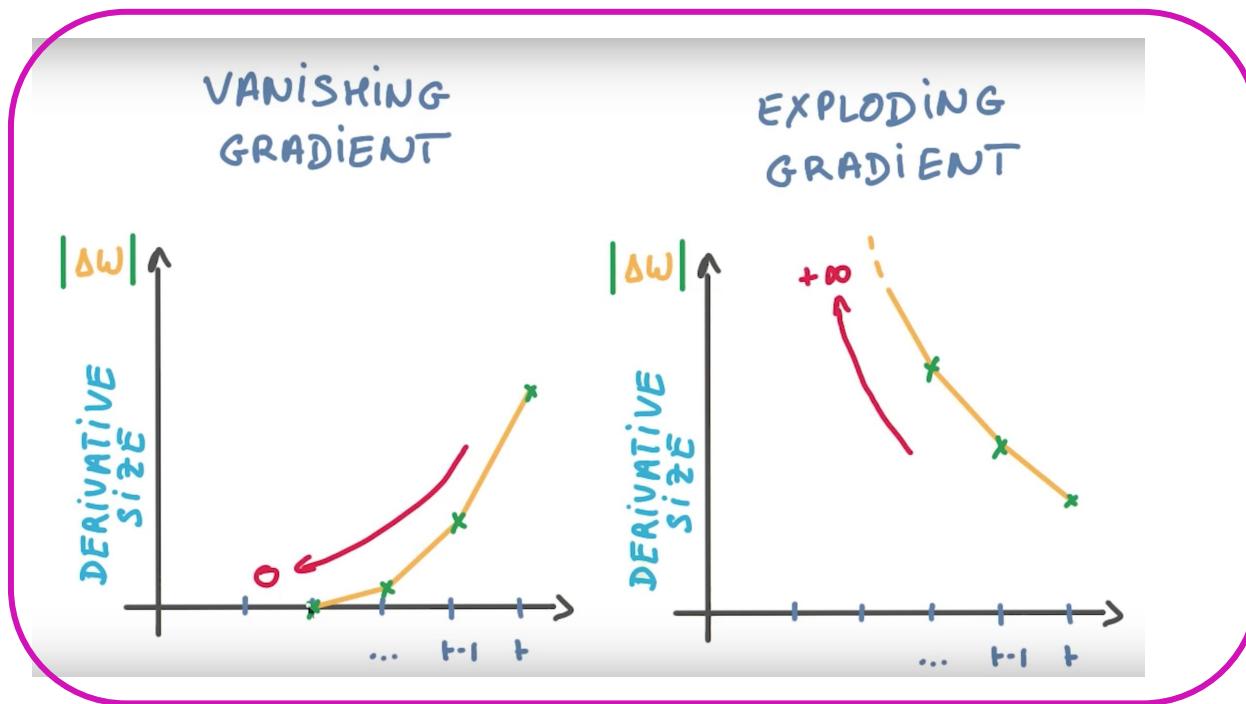
LSTMの基礎と学習

The Structure of Long Short Term Memory

Part1で扱うもの

- 長期依存問題に対するアプローチ
 - Gradient Clipping
 - Initialize and ReLu
 - Long Short Term Memory(LSTM)の解説
 - Gated Recurrent Unit(GRU)の解説
- RNN可視化ツール (LSTM-Viz)

長期依存問題に対するアプローチ



学習の方法を
工夫
Gradient Clipping
Initialize & Relu

Gradient Clipping

- 勾配爆発問題に対するアプローチ
- 勾配が閾値を超えた際に切り取ってしまえば良い(クリッピング)という単純明快なアルゴリズム

```
1.  $\hat{g} \leftarrow \frac{\partial \epsilon}{\partial \theta}$ 
2. if  $\|\hat{g}\| \geq threshold$  then
    $\hat{g} \leftarrow \frac{threshold}{\|\hat{g}\|} \hat{g}$ 
endif
```

} 勾配を計算

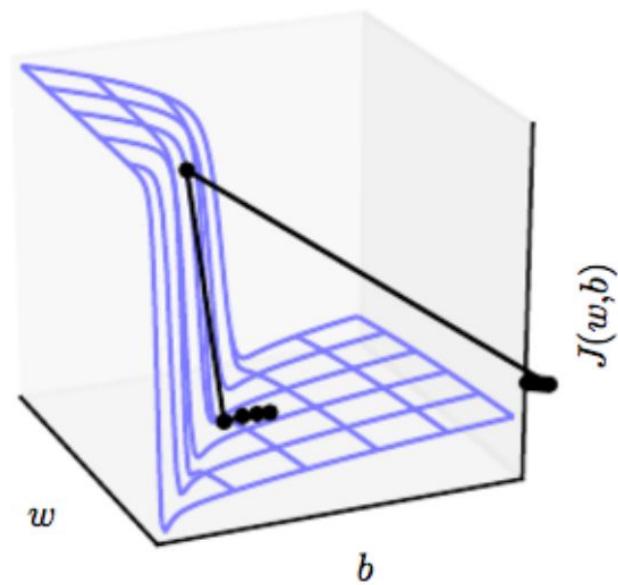
} 勾配が閾値を超えた場合はクリッピング

勾配は $\|gradient\| < threshold$ に収まる

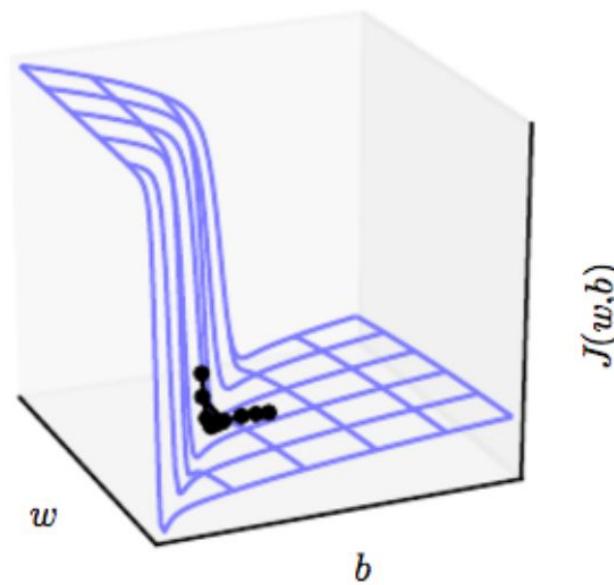
Gradient Clippingの効果

Tips: 重みのノルム平均の0.5倍～10倍程度を閾値として設定すると良いらしい

Without clipping



With clipping



Initialize & Relu [Jaity & Hinton 15]

- 勾配消失に対する最も直接的な解決策
- 重みを単位行列で初期化し、活性化関数をReLUとする

活性化関数と重みの微分 (<1) で消失

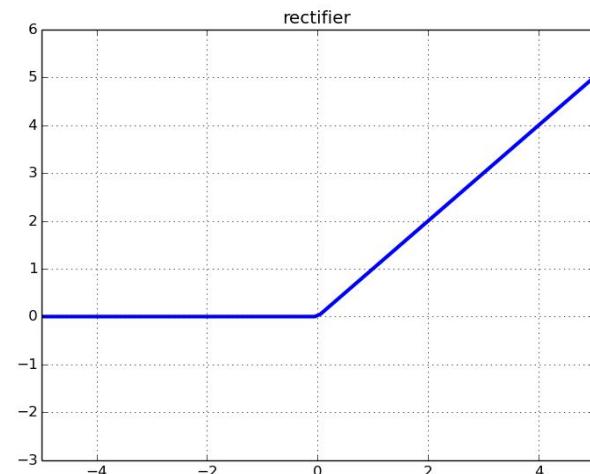
$$\frac{\partial C^{(n)}}{\partial \theta} \approx \|\Phi'_H\|^{n-k} \|W_{\text{rec}}^T\|^{n-k}$$

$$f(x) = x^+ = \max(0, x)$$

微分が1 ($x \geq 1$)

$$\forall n \in \mathbb{N} I = I^n$$

幂乗が単位行列



Rectified Linear Units

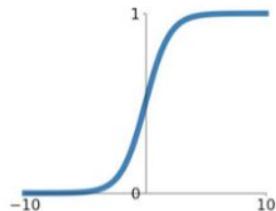
ところで「活性化関数」には...

それぞれの性質とPROS and CONS

Activation Functions

Sigmoid

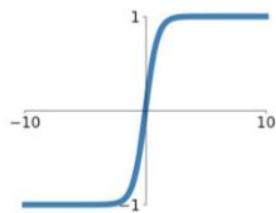
$$\sigma(x) = \frac{1}{1+e^{-x}}$$



(+) 確率的解釈が可能。ゲートに使える
(e.g. LSTM)

tanh

$$\tanh(x)$$



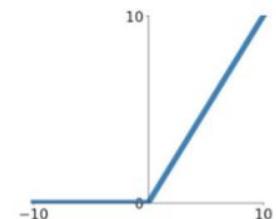
(+) 原点対称、出力が[-1, 1]

σ に比べて勾配消失に強い

(-) 原点付近で正負が逆転。
意味が反転 (否定の否定 → 肯定)

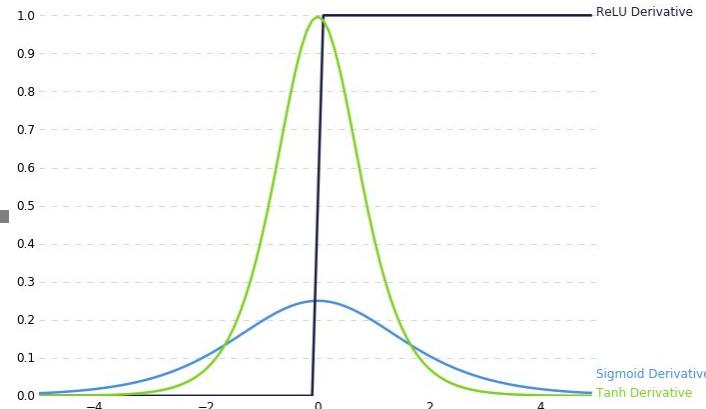
ReLU

$$\max(0, x)$$

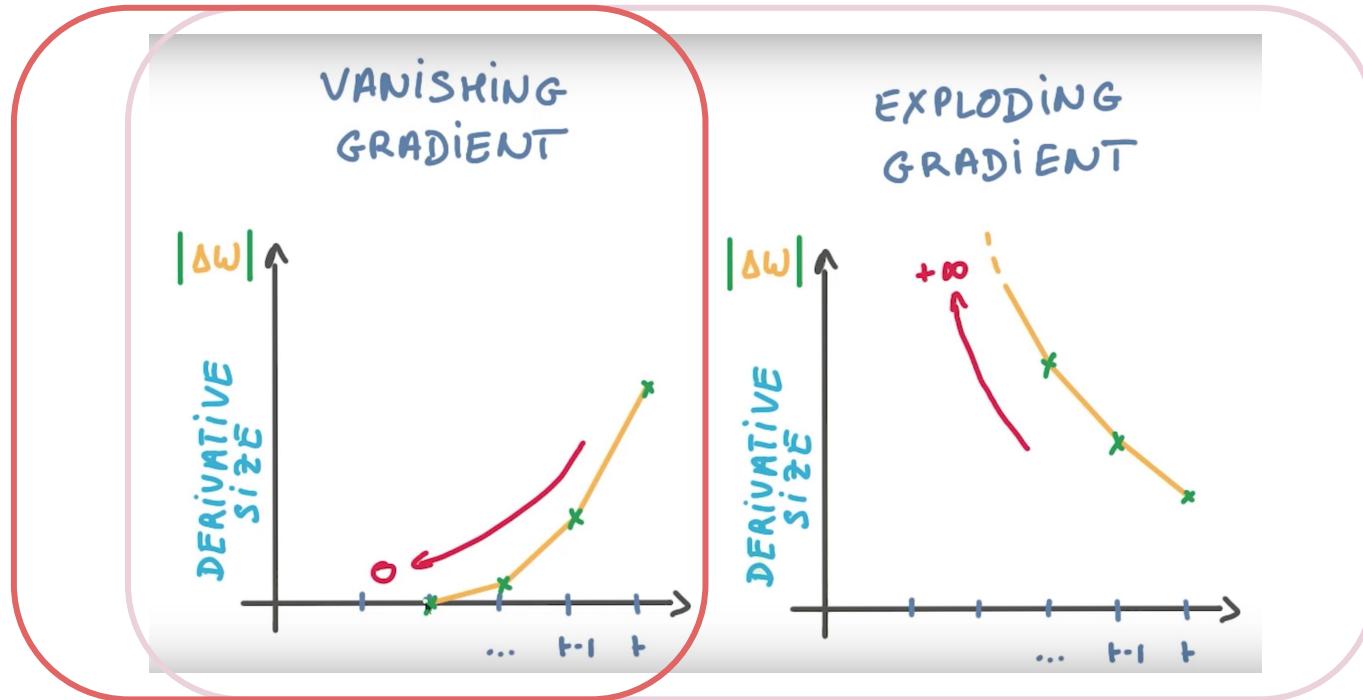


(+) 解釈が簡単。飽和せず、勾配消失に強い

(-) 無駄死にするニューロンが増える



長期依存問題に対するアプローチ

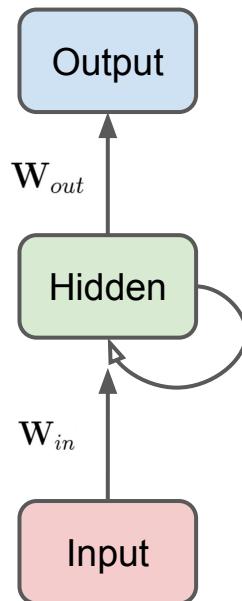


ネットワークの構
造を工夫
LSTM・GRU

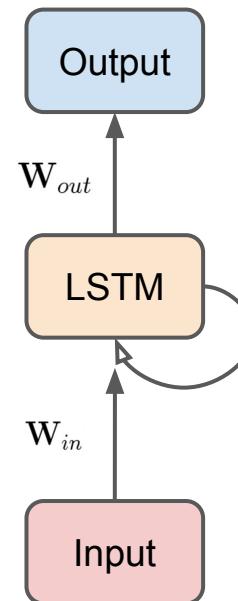
学習の方法を
工夫
Gradient Clipping
Initialize & Relu

Long Short Term Memory

- 長期依存問題に対するアプローチ
- Simple RNNの改良
- RNN cellをLSTMブロックに置き換えれば良い！(単純!)



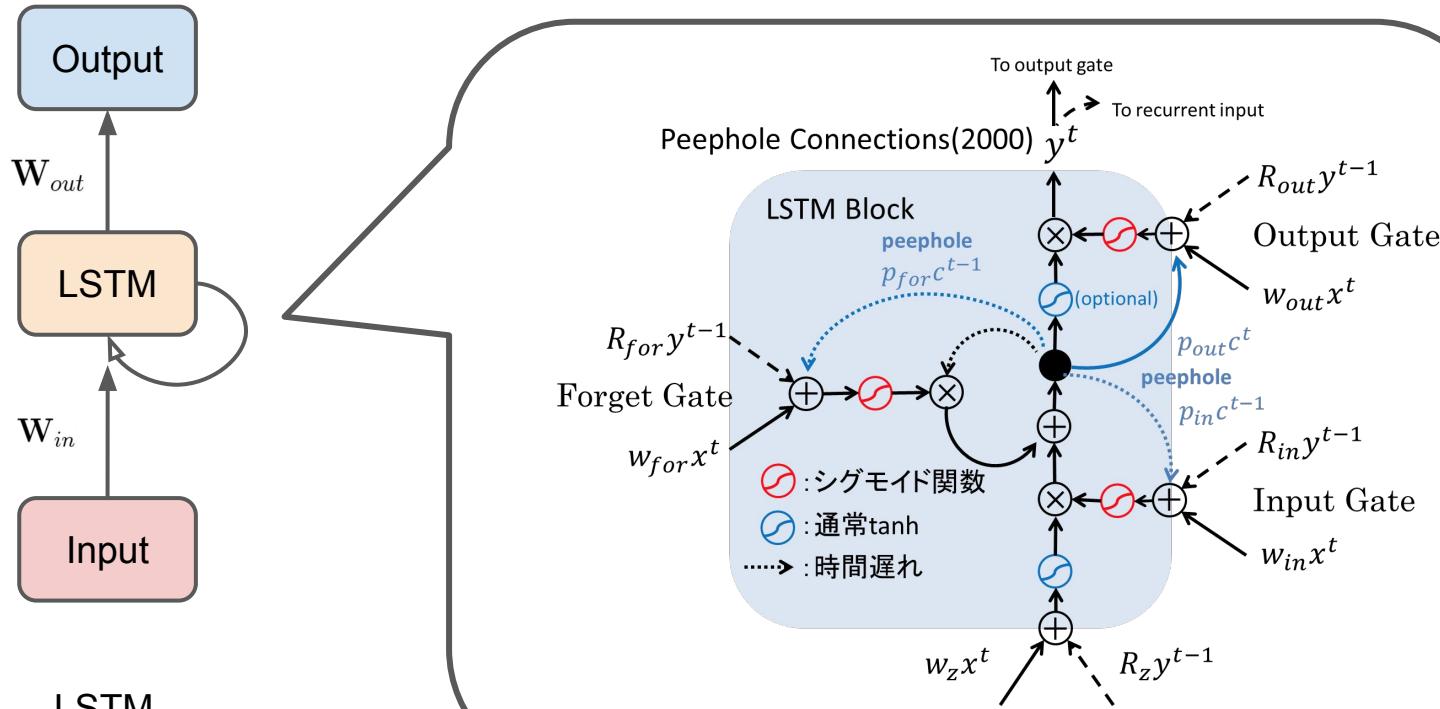
Simple RNN



LSTM

LSTMの中身は...?

ネットでよく出てくるLSTMの図代表

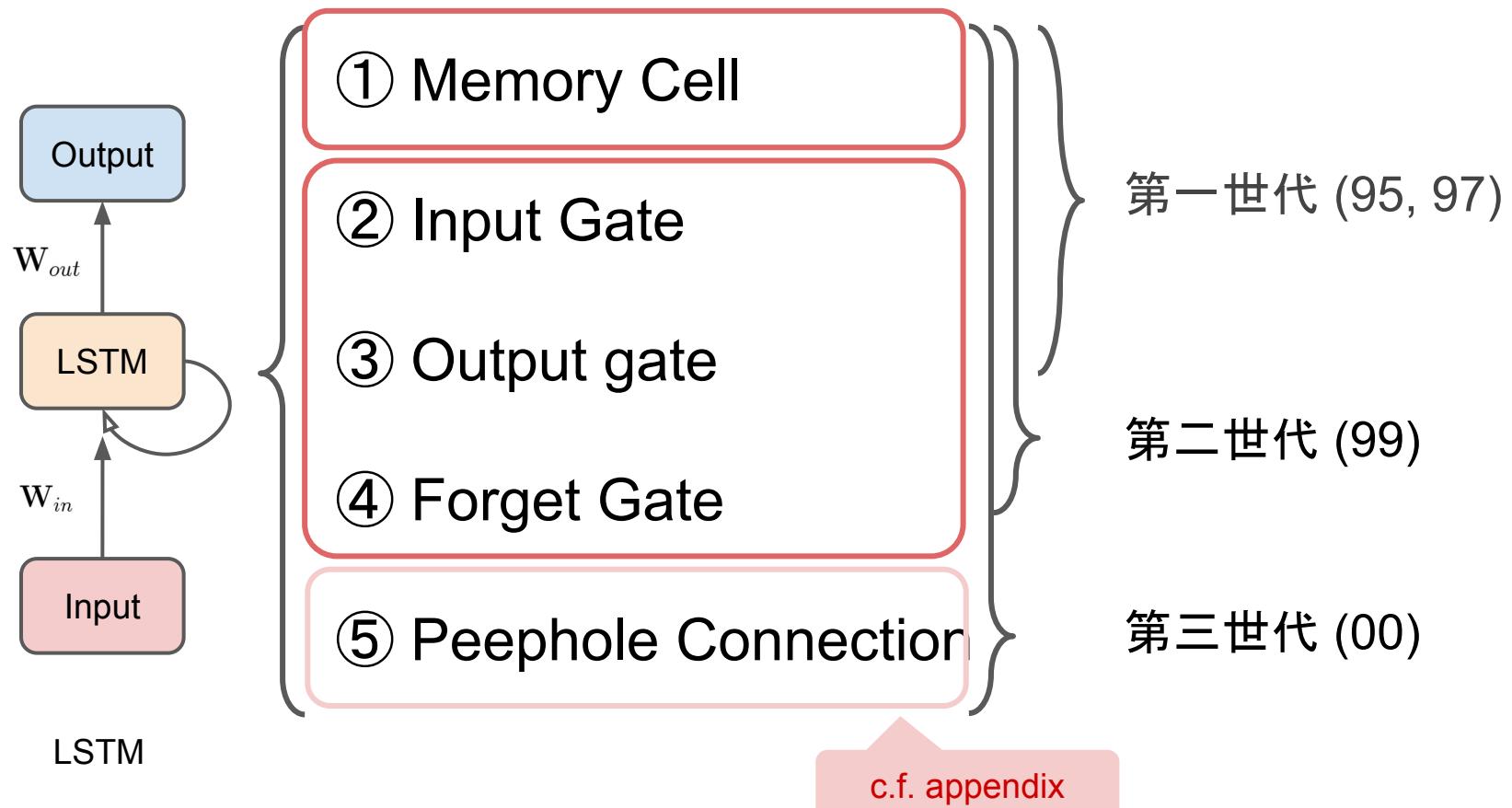


ひとつずつ丁寧に解説していきます

LSTMのきほん

<LSTM Blockの構成>

<世代>



LSTM: Constant Error Carousel(CEC)の導入 (1/3)

- 勾配消失問題をNW構造を変えて解消する試み
(c.f. [Jaitly & Hinton 15])
- そもそもの原因は...

$$\frac{\partial C^{(n)}}{\partial \theta} \approx \left\| \Phi_H' \right\|^{n-k} \left\| W_{\text{rec}}^T \right\|^{n-k}$$

$\prod_{k < i \leq n} \frac{\partial h^{(i)}}{\partial z^{(i)}} \frac{\partial z^{(i)}}{\partial h^{(i-1)}}$

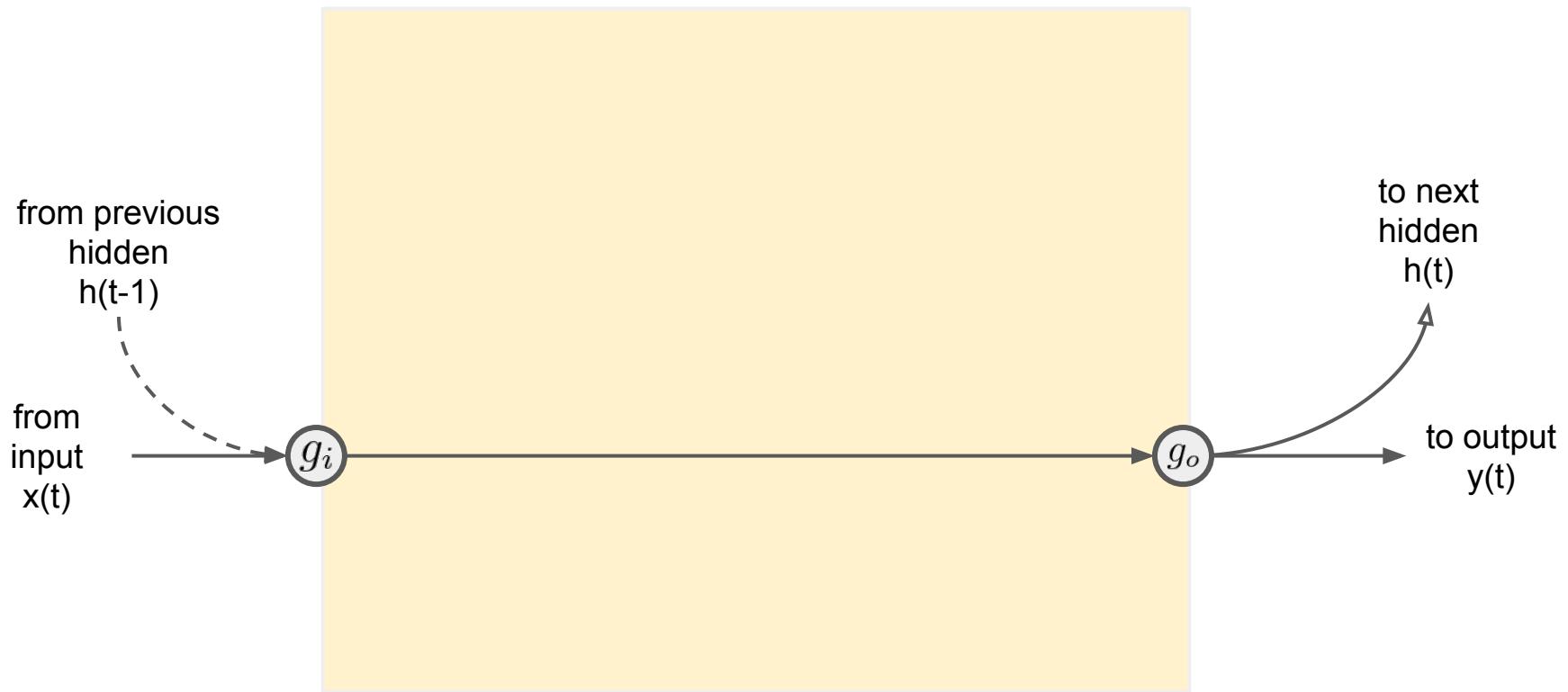
$\|x\| < 1$: で勾配が消失 ($\rightarrow 0$)
 $\|x\| > 1$: で勾配が爆発 ($\rightarrow \infty$)

i.e. 勾配消失を防ぐためには以下の式を満たせば良い

$$W_r = I, f_h(x) = x$$

LSTM: Constant Error Carousel(CEC)の導入 (2/3)

ネットワーク図

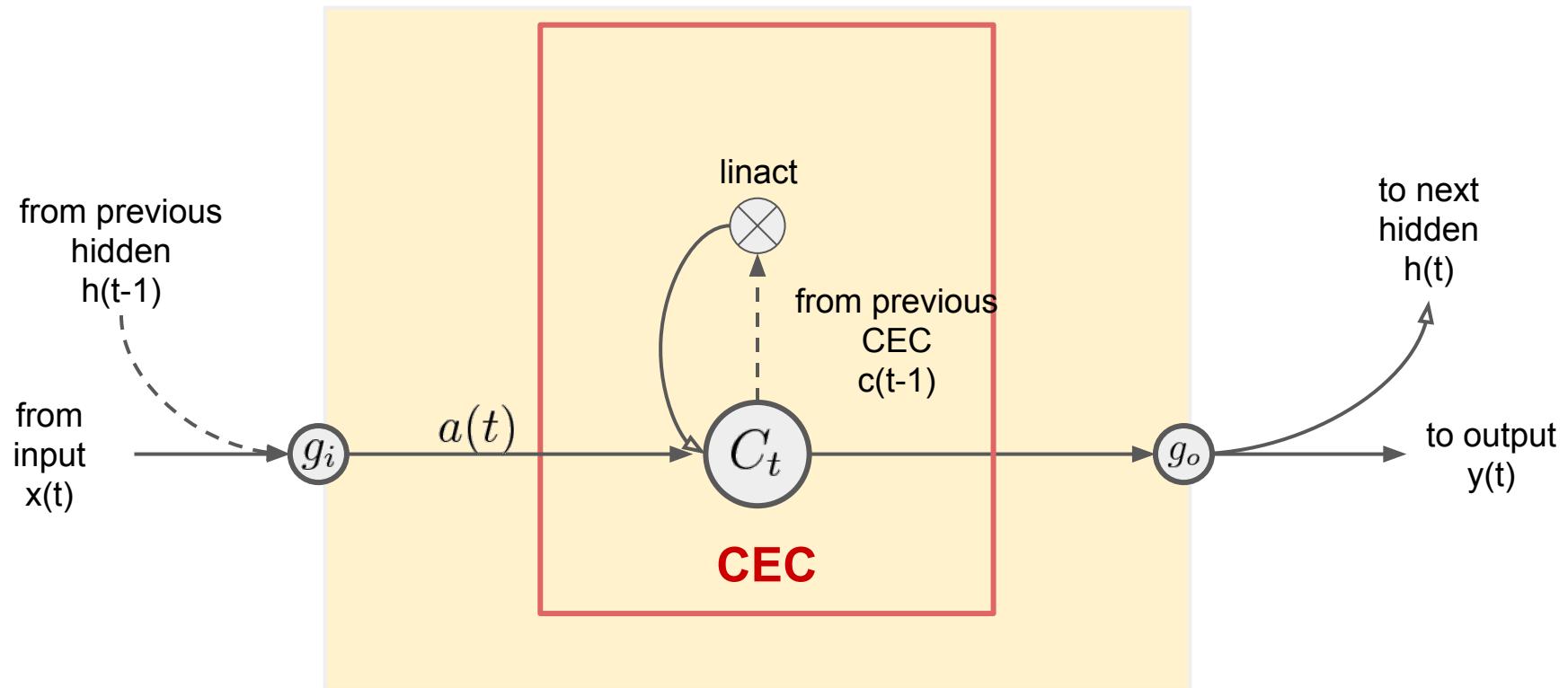


LSTM: Constant Error Carousel(CEC)の導入 (3/3)

ネットワーク図

C_t

= 隠れ層の状態を保持するメモリセル



Forwarding CEC

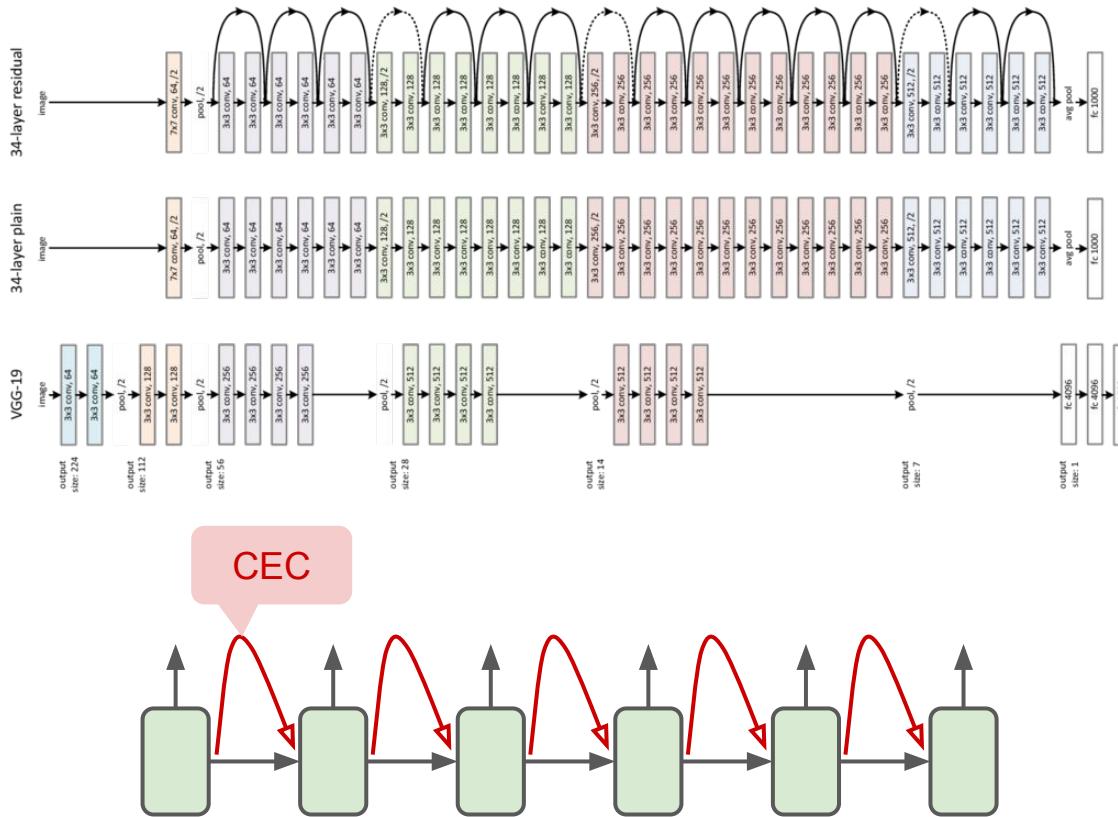
$$c(t) = a(t) + c(t - 1)$$

Backprop CEC

$$\frac{\partial E}{\partial \mathbf{c}(t-1)} = \frac{\partial E}{\partial \mathbf{c}(t)} \odot \frac{\partial \mathbf{c}(t)}{\partial \mathbf{c}(t-1)} = \frac{\partial E}{\partial \mathbf{c}(t)}$$

LSTM: CECと類似した構造

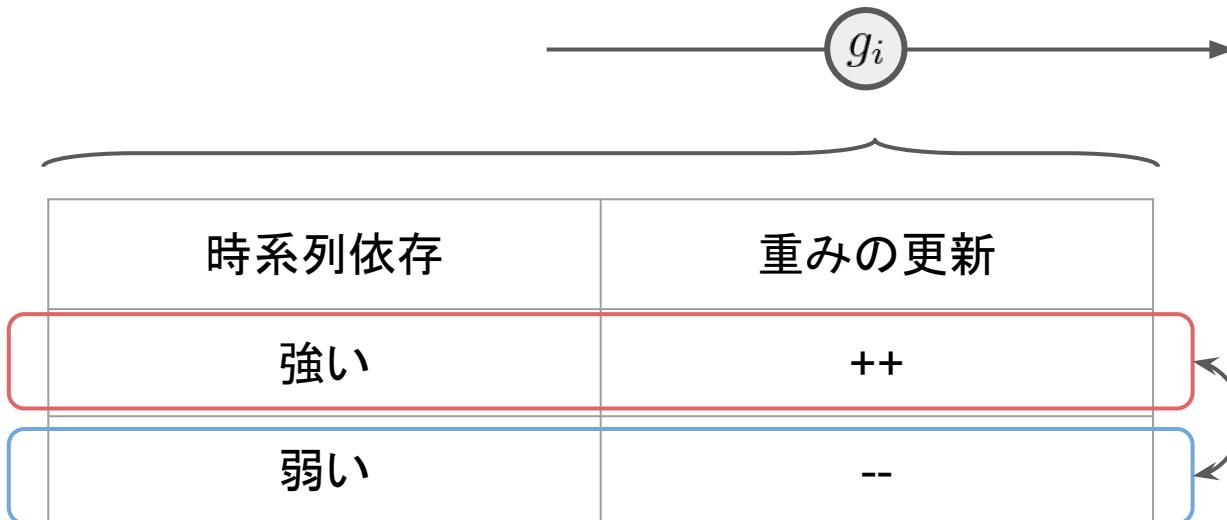
Skip
Connection



Skip Connection?

LSTM: 入力重み衝突と出力重み衝突

- CECでは勾配が消失しないように工夫した
- 一方、学習をすすめると入/出力重み衝突という問題が浮き彫りになってきた



重みは一つしか無い
ため衝突(conflict)が
起こる

依存するものは残す vs 関係ないものは残さない

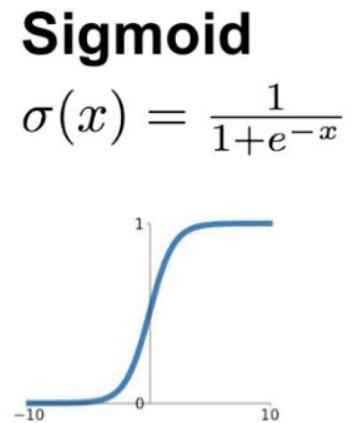
LSTM: 入出力ゲートの導入 (1/4)

- 入出力重み衝突を解消するため入出力ゲートを導入



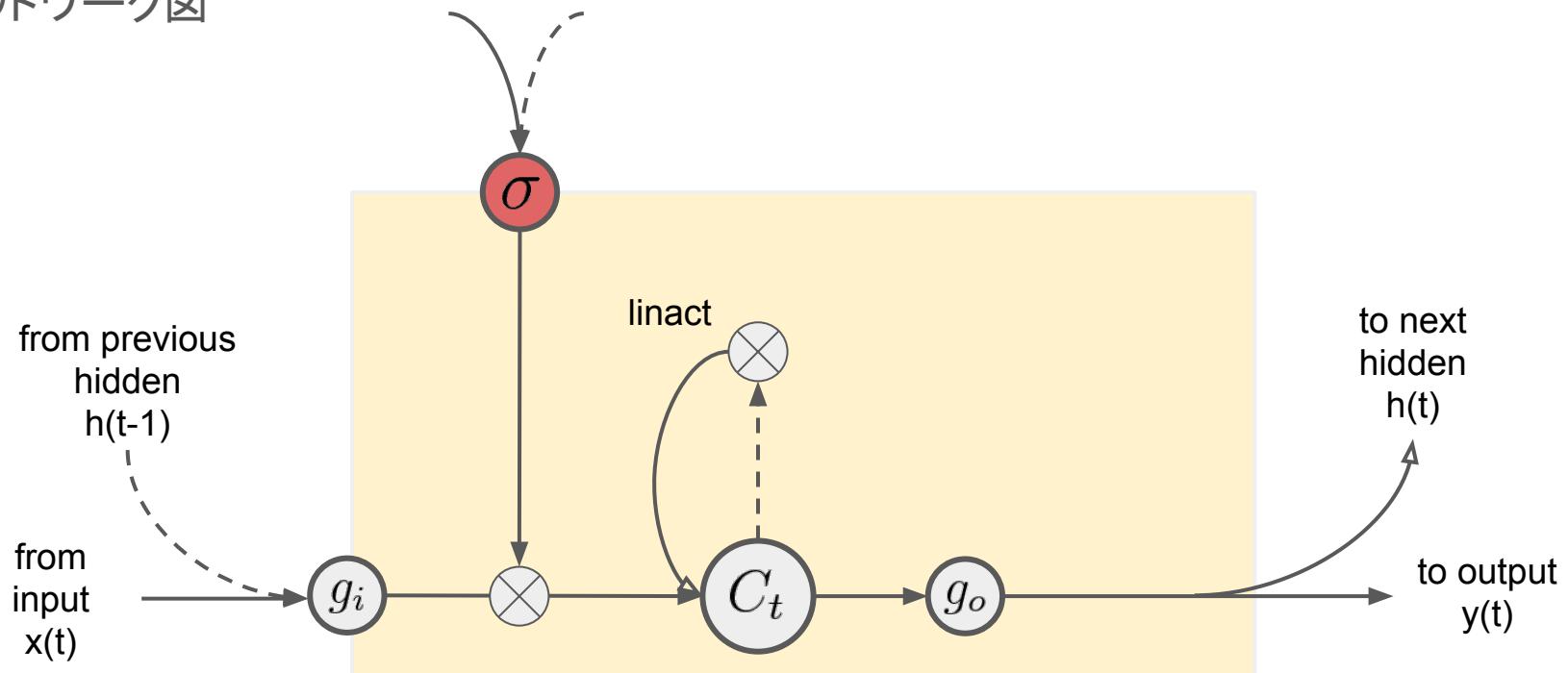
依存するものは残す vs 関係ないものは残さない

→ 依存するデータを受け取った時だけ、
情報を伝播する「ゲート」を導入する



LSTM: 入出力ゲートの導入 (2/4)

ネットワーク図

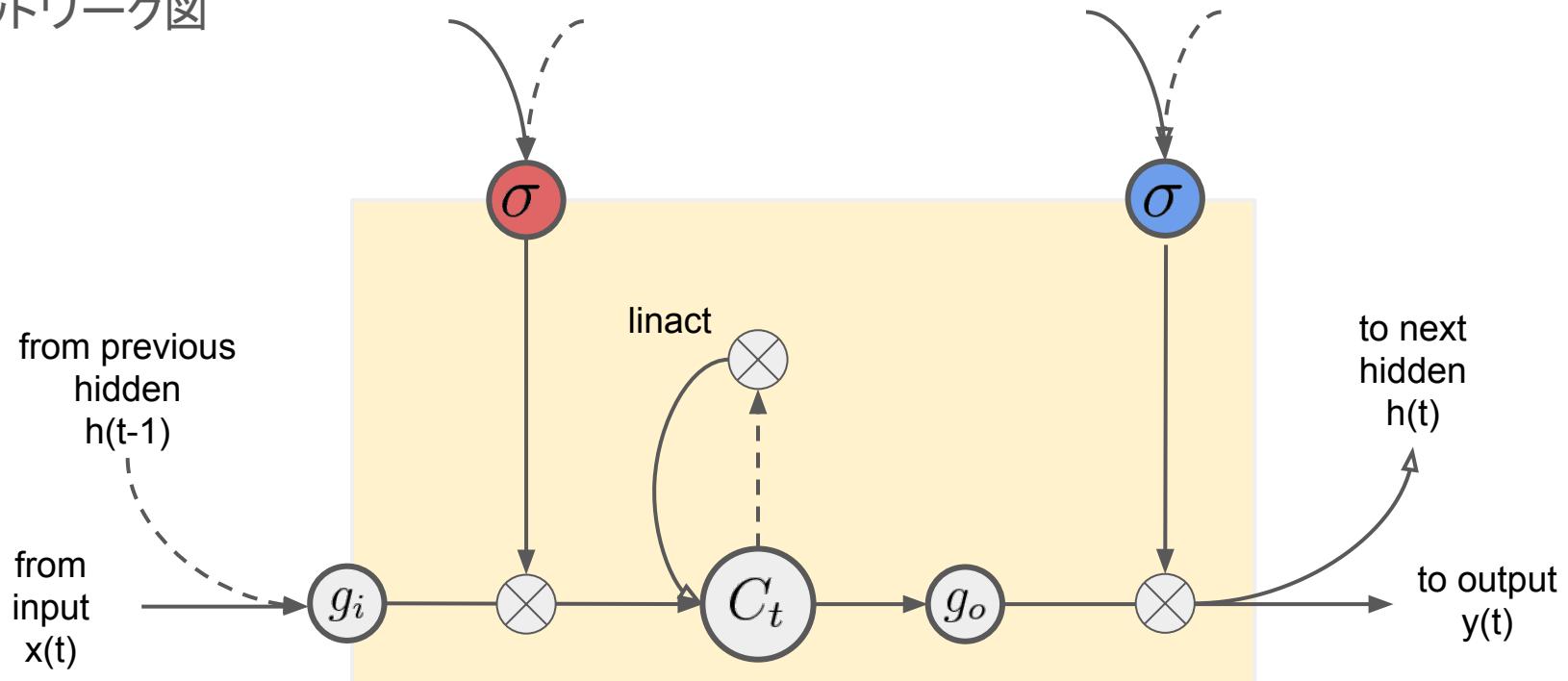


Forwarding input gate

$$\mathbf{i}(t) = \sigma(W_i \mathbf{x}(t) + U_i \mathbf{h}(t-1) + \mathbf{b}_i)$$

LSTM: 入出力ゲートの導入 (3/4)

ネットワーク図

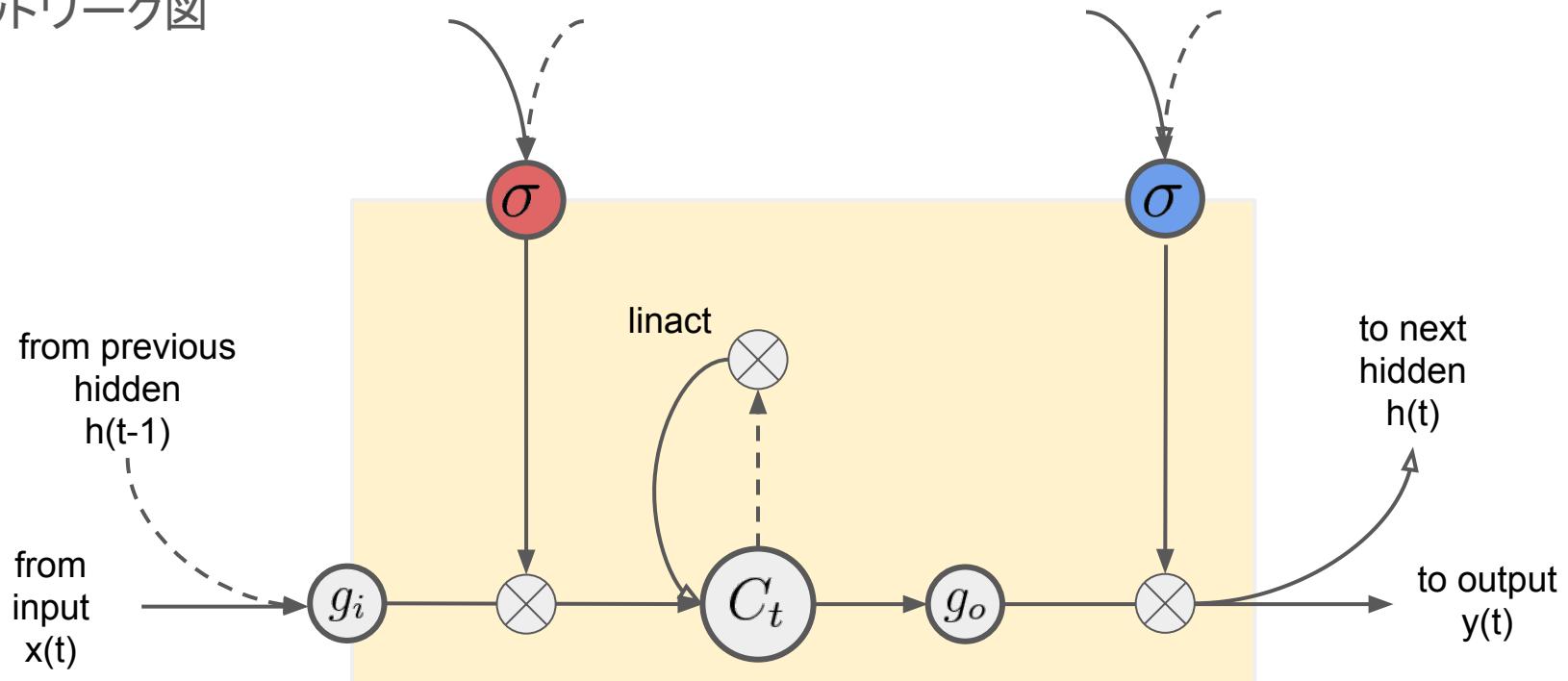


Forwarding output gate

$$\mathbf{o}(t) = \sigma(W_o \mathbf{x}(t) + U_o \mathbf{h}(t-1) + \mathbf{b}_o)$$

LSTM: 入出力ゲートの導入 (4/4)

ネットワーク図



Forwarding CEC

$$\mathbf{c}(t) = \mathbf{i}(t) \odot \mathbf{a}(t) + \mathbf{c}(t - 1)$$

LSTM: メモリセルの値を消去する

依存するものは残す vs 依存しないものは残さない

ここまで...

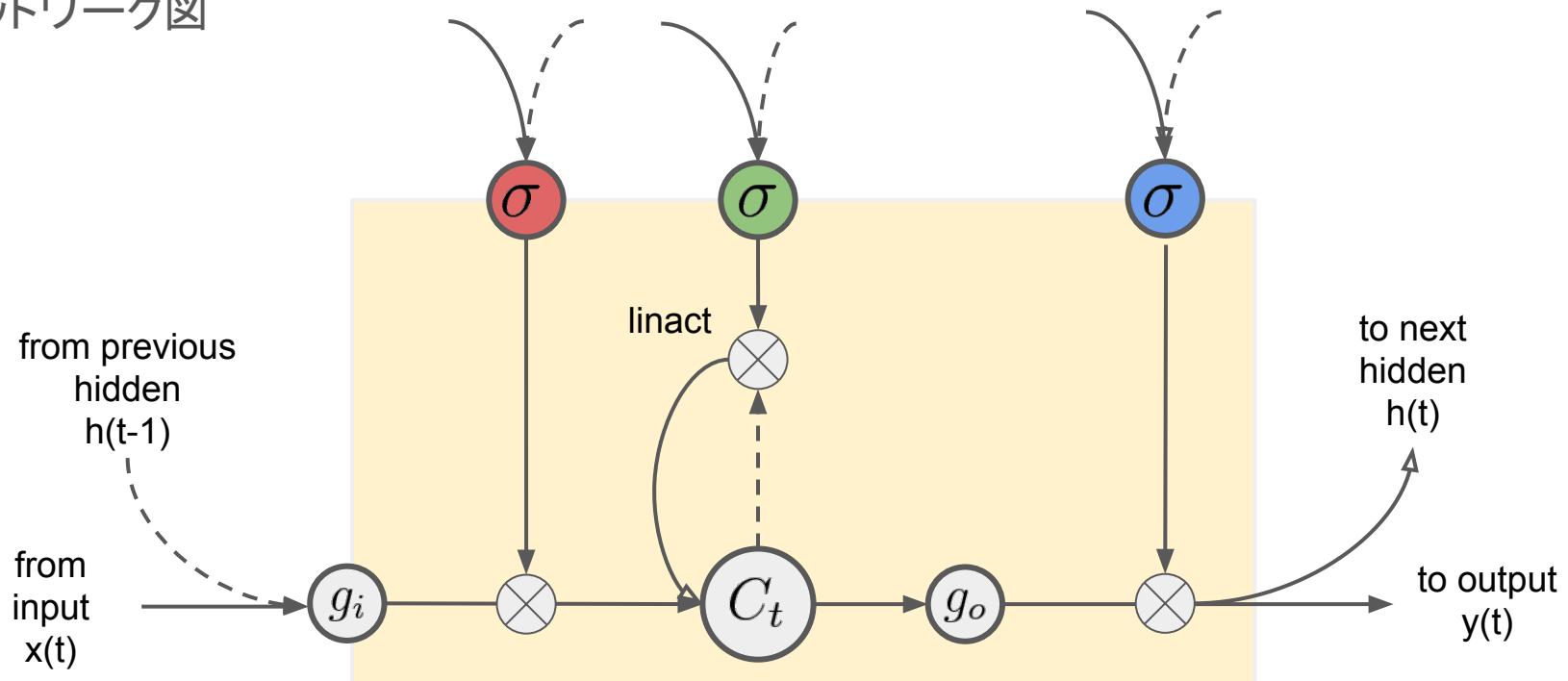
入出力重み衝突 → 「入出力ゲート」を定義

一方、入力系列のパターンが劇的に変化する場合などでは、
積極的にメモリセル(CEC)の値を消去することが望ましい。

依存しないものは残さない → 「忘却ゲート」を定義

LSTM: 忘却ゲートの導入 (1/2)

ネットワーク図

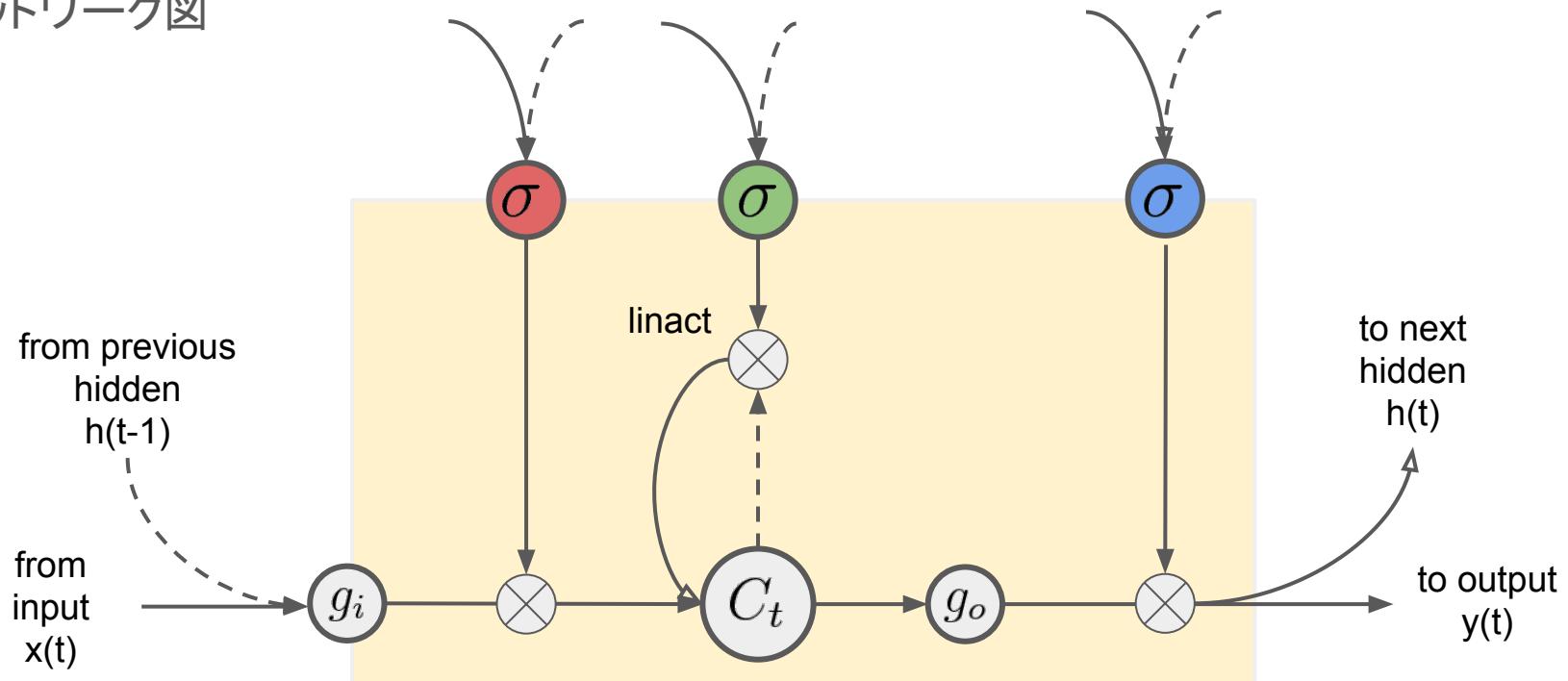


Forwarding forget gate

$$\mathbf{f}(t) = \sigma(W_f \mathbf{x}(t) + U_f \mathbf{h}(t-1) + \mathbf{b}_f)$$

LSTM: 忘却ゲートの導入 (2/2)

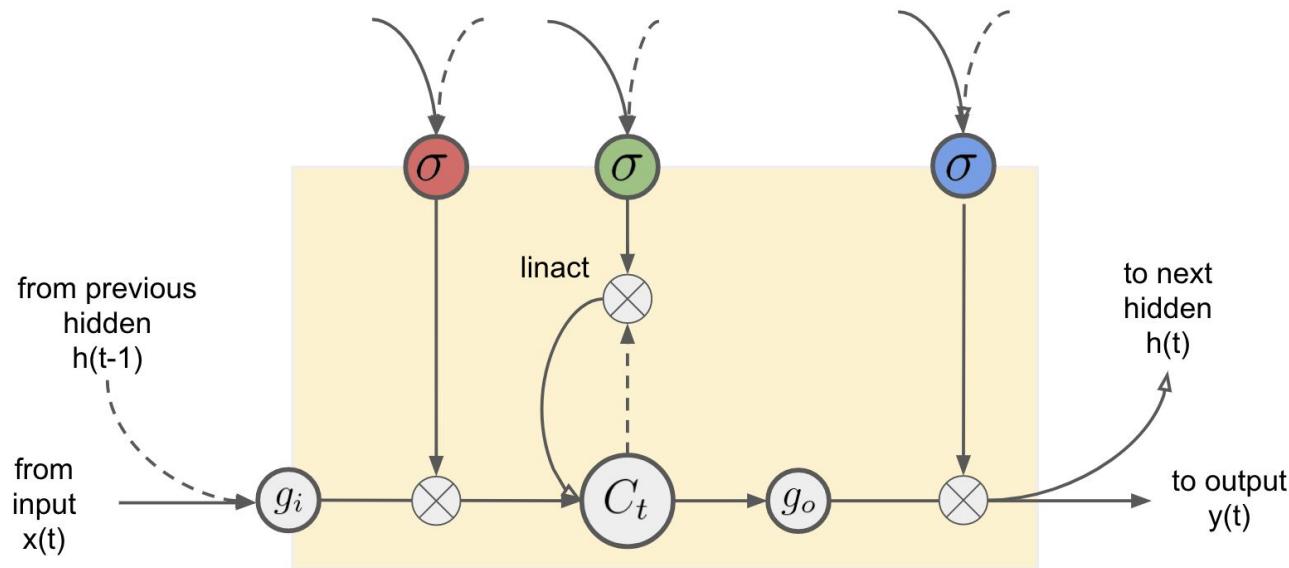
ネットワーク図



Forwarding CEC

$$\mathbf{c}(t) = \mathbf{i}(t) \odot \mathbf{a}(t) + \mathbf{f}(t) \odot \mathbf{c}(t - 1)$$

LSTM: まとめ



- Gate Status
- CEC, Hidden Status

$$\begin{pmatrix} \mathbf{a}(t) \\ \mathbf{i}(t) \\ \mathbf{o}(t) \\ \mathbf{f}(t) \end{pmatrix} = \begin{pmatrix} \text{tanh} \\ \text{sigm} \\ \text{sigm} \\ \text{sigm} \end{pmatrix} \left[\widetilde{\mathbf{W}}_x \widetilde{\mathbf{W}}_h \right] \begin{pmatrix} \mathbf{x}(t) \\ \mathbf{h}(t-1) \end{pmatrix}$$

$$\mathbf{c}(t) = \mathbf{f}(t) \odot \mathbf{c}(t-1) + \mathbf{i}(t) \odot \mathbf{a}(t)$$

$$\mathbf{h}(t) = \mathbf{o}(t) \odot \tanh(\mathbf{c}(t))$$

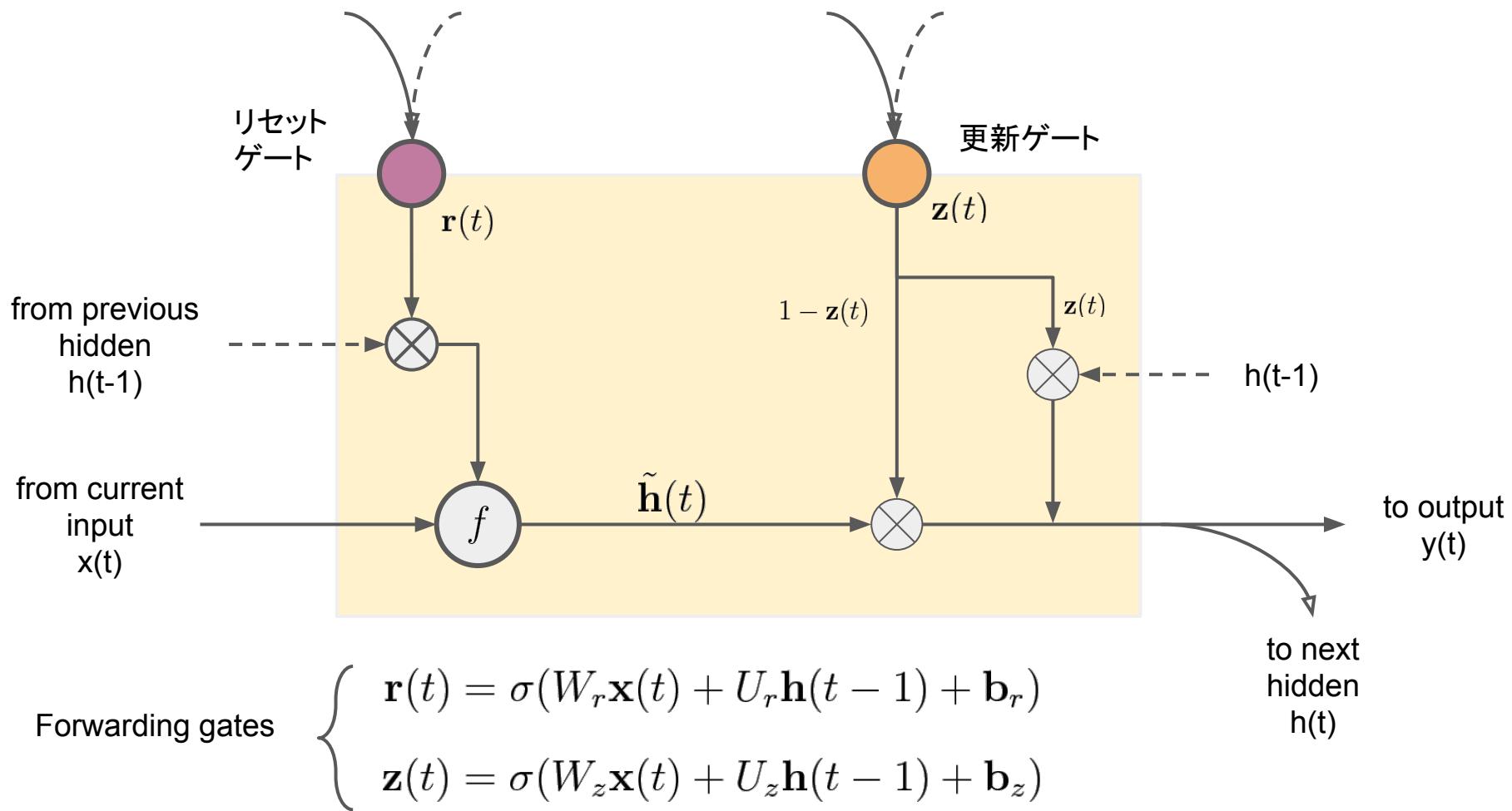
GRU [Cho+ 14]

Gated Recurrent Unit (GRU)

- 一般的にRNNは計算コストが高い
 - 特にLSTMはパラメタ数が多い (e.g. peepholeを入れて最大15個)
- よりシンプルなモデルとしてGRUを提案
- リセットゲート + 更新ゲートという構成
- パラメタ数は9個で学習が比較的容易

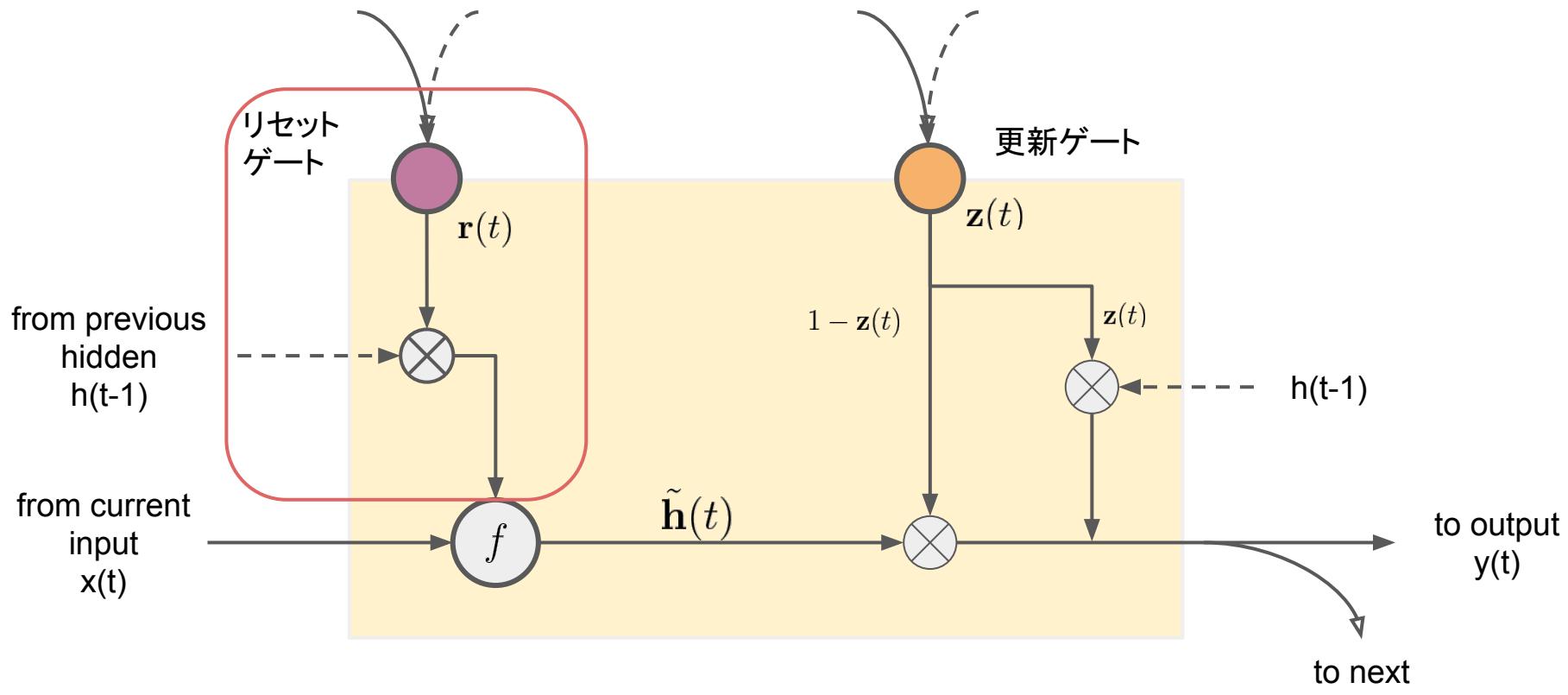
GRU: ゲート構造の導入

ネットワーク図



GRU: 順伝搬 (1/2)

ネットワーク図

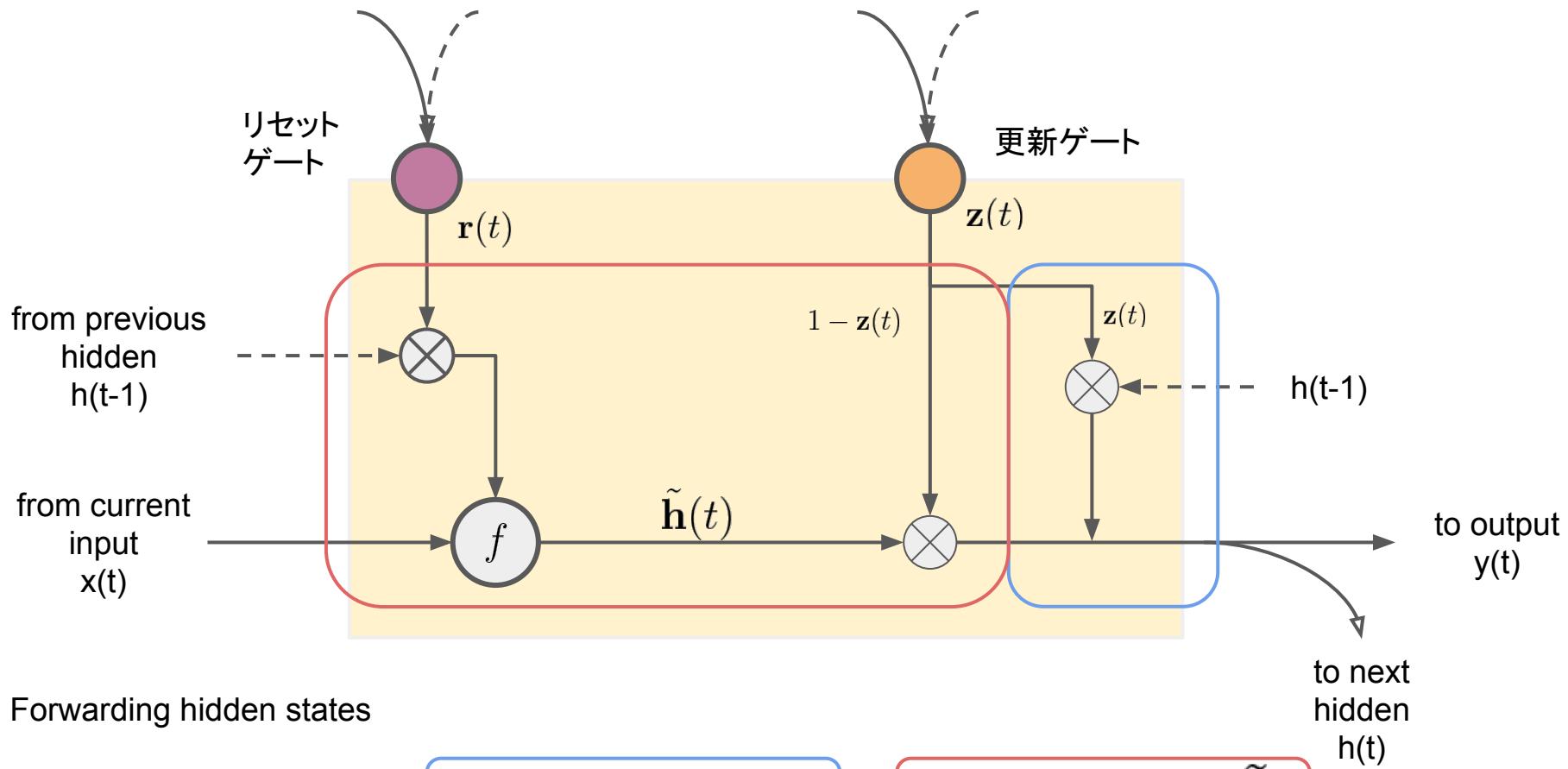


Forwarding internal hidden states

$$\tilde{h}(t) = f(W_h \mathbf{x}(t) + U_h(\mathbf{r}(t) \odot \mathbf{h}(t-1)) + \mathbf{b}_h)$$

GRU: 順伝搬 (2/2)

ネットワーク図



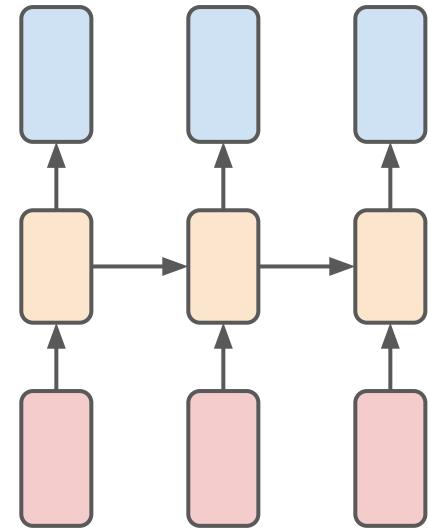
LSTMの学習 (復習)

Gradient Descent (GD)

- 最急降下法

勾配を求めたい

$$\theta^{k+1} = \theta^k - \alpha \frac{\partial J}{\partial \theta}$$

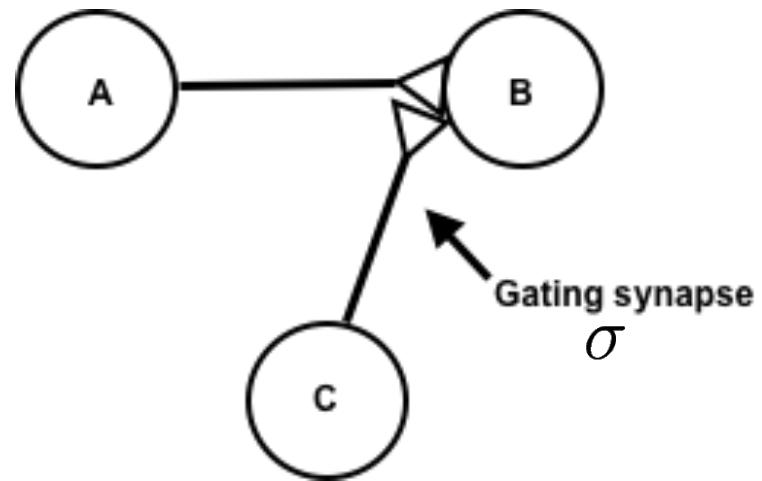
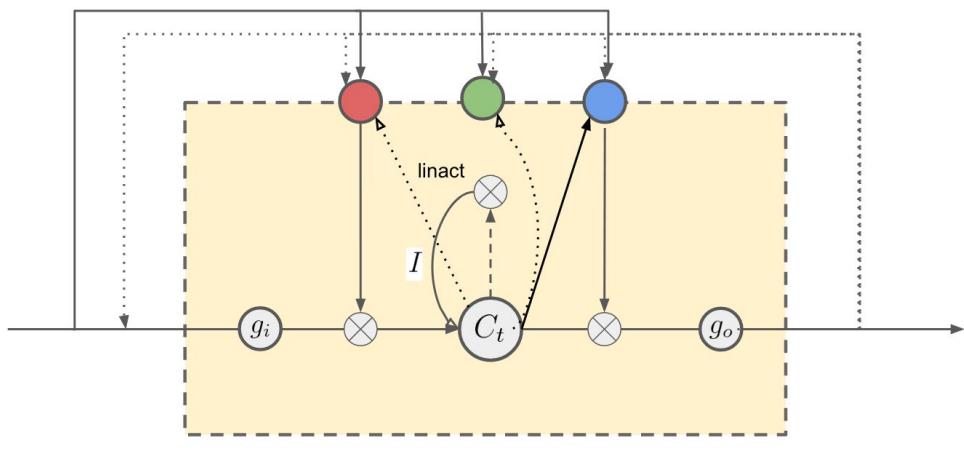


Backpropagation Through Time (BPTT)

LSTMを展開する
(c.f. lecture9)

- 勾配を計算
- LSTMを展開して、FFNNとしてBPを行う
- 時間軸方向(Through Time)のBackpropagation

LSTM, GRUの生物学的根拠(?)



LSTM & GRU まとめ

- LSTMはCECとゲーティング構造を持つネットワーク
 - 長期依存問題を解決
- しかし、一般的に計算コストが高い
 - RNNでは並列計算の実現が難しい
 - 特にLSTMはパラメタが15個と多い
- よりシンプルなモデルとしてGRUがある
 - パラメタ9個で学習が比較的容易

小演習 (Check1)

隠れ素子の可視化ツールを試そう！

LSTM Vis

RNNの学習状態をモニタリングするのは通常困難であるが、隠れ素子の状態をわかりやすく可視化するツールも存在する



<http://lstm.seas.harvard.edu/>

Live Server

で試せる

(e.g.) 隠れ素子の発火状態

Cell sensitive to position in line:

The sole importance of the crossing of the Berezina lies in the fact that it plainly and indubitably proved the fallacy of all the plans for cutting off the enemy's retreat and the soundness of the only possible line of action--the one Kutuzov and the general mass of the army demanded--namely, simply to follow the enemy up. The French crowd fled at a continually increasing speed and all its energy was directed to reaching its goal. It fled like a wounded animal and it was impossible to block its path. This was shown not so much by the arrangements it made for crossing as by what took place at the bridges. When the bridges broke down, unarmed soldiers, people from Moscow and women with children who were with the French transport, all--carried on by vis inertiae--pressed forward into boats and into the ice-covered water and did not, surrender.

Cell that turns on inside quotes:

"You mean to imply that I have nothing to eat out of.... on the contrary, I can supply you with everything even if you want to give dinner parties," warmly replied Chichagov, who tried by every word he spoke to prove his own rectitude and therefore imagined Kutuzov to be animated by the same desire.

Kutuzov, shrugging his shoulders, replied with his subtle penetrating smile: "I meant merely to say what I said."

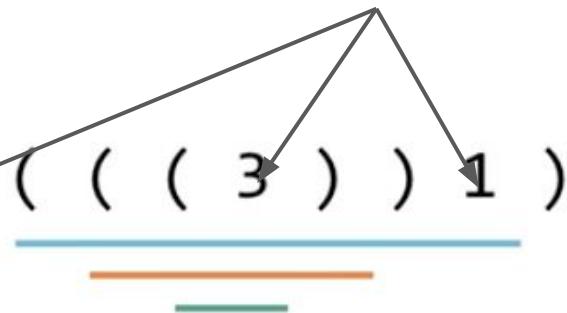
LSTMVis でネストの深さに反応するセルを見つける

(e.g.) Character Model (Parentheses)

alphabet: () 0 1 2 3 4

(ランダムな位置で)
ネストの深さを示す数字が挿入される

corpus: (1 (2) ()) 0 (((3)) 1)



LSTMVis - Visual Analysis for Recurrent Neural Networks

LSTMVis allows you to interactively analyze the hidden state vectors of a recurrent neural network model, by simply selecting and comparing regions of the input. To demonstrate the system, we have provided a set of real example models and datasets to play with, including several word and character language models for text, music and code, a sequence auto-encoder, a German <-> English neural translation system, and a sentence summarization system. We recommend that you begin with the [parentheses dataset](#), which has a simple and regular structure to demonstrate the use of the tool.

Character Model (Parentheses)	Word Model (Children's Books)	Word Model (Children's Books)
A simple synthetic dataset to test bracketing and counting. The language can open and close parents, and generate a number indicating the current nesting level.	A 1x200 LSTM language model trained on the Gutenberg Children's Book corpus.	A 4x500 LSTM language model trained on the Gutenberg Children's Book corpus.
meta: --- index: false length: 1000001	meta: named_entity, part_of_speech index: true length: 1271912	meta: named_entity, part_of_speech index: true length: 1271912
layer 1 cell (size: 999600 x 650) layer 1 hidden (size: 999600 x 650)	layer 1 cell (size: 1271900 x 200) layer 1 hidden (size: 1271900 x 200)	layer 1 cell (size: 317975 x 500) layer 1 hidden (size: 317975 x 500) layer 4 cell (size: 317975 x 500) layer 4 hidden (size: 317975 x 500)
Word Model (Wall Street Journal)	Character Model (Wall Street Journal)	Sequence Auto-Encoder
A 2x650 LSTM language model trained on the Wall Street Journal. Annotated with gold-standard part-of-speech tags.	A 2x650 LSTM character language model trained on the Wall Street Journal.	The encoder states of an attention-based sequence auto-encoder. Trained on the Wall Street Journal. Implemented using seq2seq-attn.
meta: named_entity, part_of_speech index: true length: 929589	meta: vowels index: false length: 4879470	meta: named_entity, part_of_speech index: true length: 1000000
layer 1 cell (size: 840000 x 650)	layer 1 cell (size: 840000 x 650)	

meta: ---
index: false
length: 1000001

layer 1 cell (size: 999600 x 650)
layer 1 hidden (size: 999600 x 650)
layer 2 cell (size: 999600 x 650)
layer 2 hidden (size: 999600 x 650)
layer 2 gates (size: 999600 x 650)
layer 2 cell pca (size: 999600 x 100)

調べてみよう

以下の条件に選択的に反応するニューロンを発見せよ

1. Character Model(Parenthesis)で、
特定のネストの深さに反応する
2. Word Modelで、人物名に反応する
3. Word Modelで、noun phraseに反応する
4. Word Modelで、if clauseに反応する

Part2で扱うもの

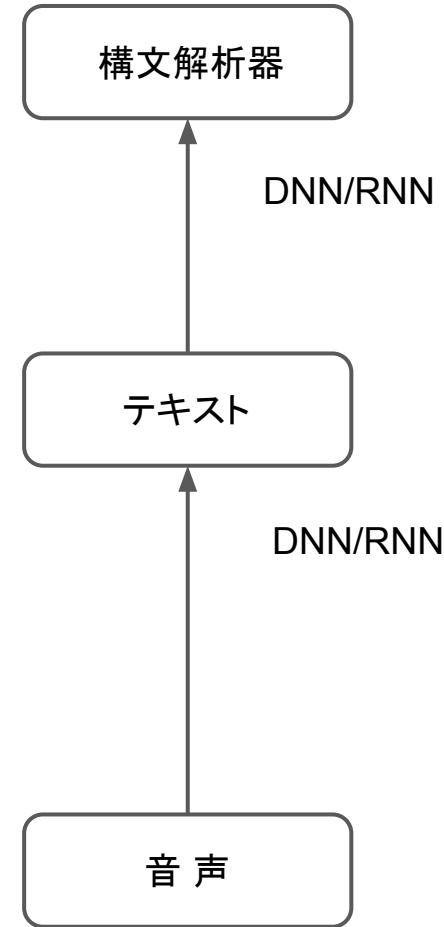
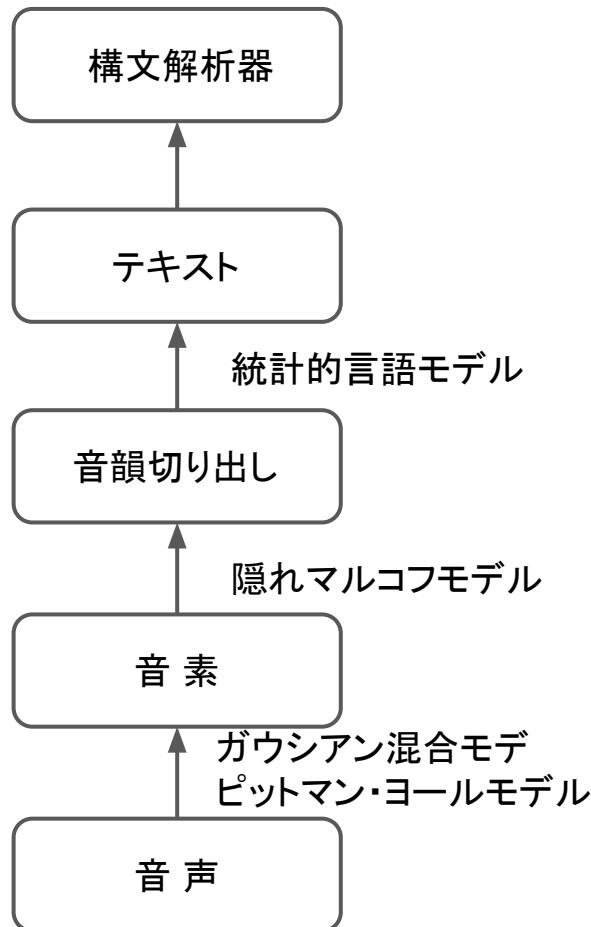
- LSTM・GRUの応用例
- Chainer LSTMのtips
 - Chainer LSTM のversion
 - Gradient Clipping
 - Truncated Backprop
 - F.pad_sequence

講義 Part2

LSTM・GRUの応用

Applications of LSTM and GRU.

音声認識



End to Endで学習する試みがある

Image Captioning(1/2) [Vinyals+ 15]

A person riding a motorcycle on a dirt road.



A group of young people playing a game of frisbee.



A herd of elephants walking across a dry grass field.



Two dogs play in the grass.



Two hockey players are fighting over the puck.



A close up of a cat laying on a couch.



A skateboarder does a trick on a ramp.



A little girl in a pink hat is blowing bubbles.



A red motorcycle parked on the side of the road.



A dog is jumping to catch a frisbee.



A refrigerator filled with lots of food and drinks.



A yellow school bus parked in a parking lot.



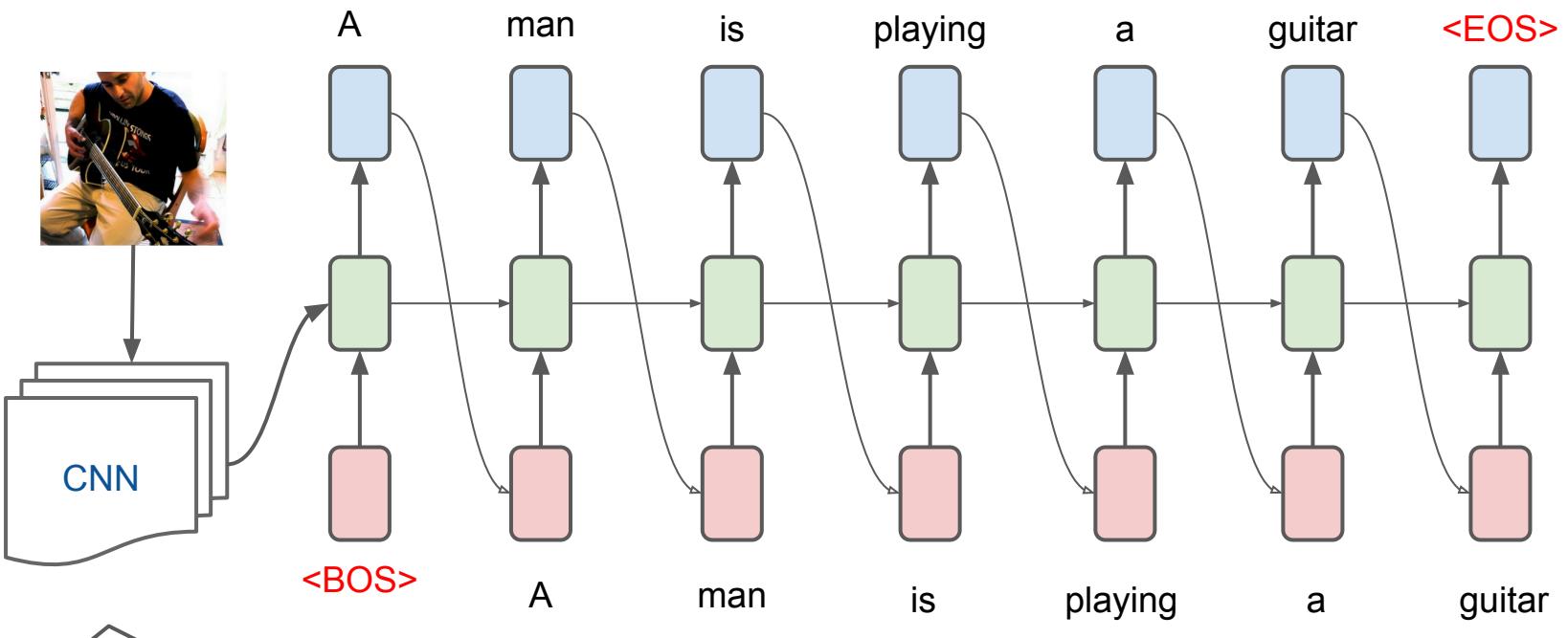
Describes without errors

Describes with minor errors

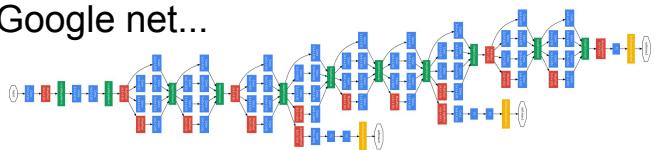
Somewhat related to the image

Unrelated to the image

Image Captioning(2/2) [Vinyals+ 15]



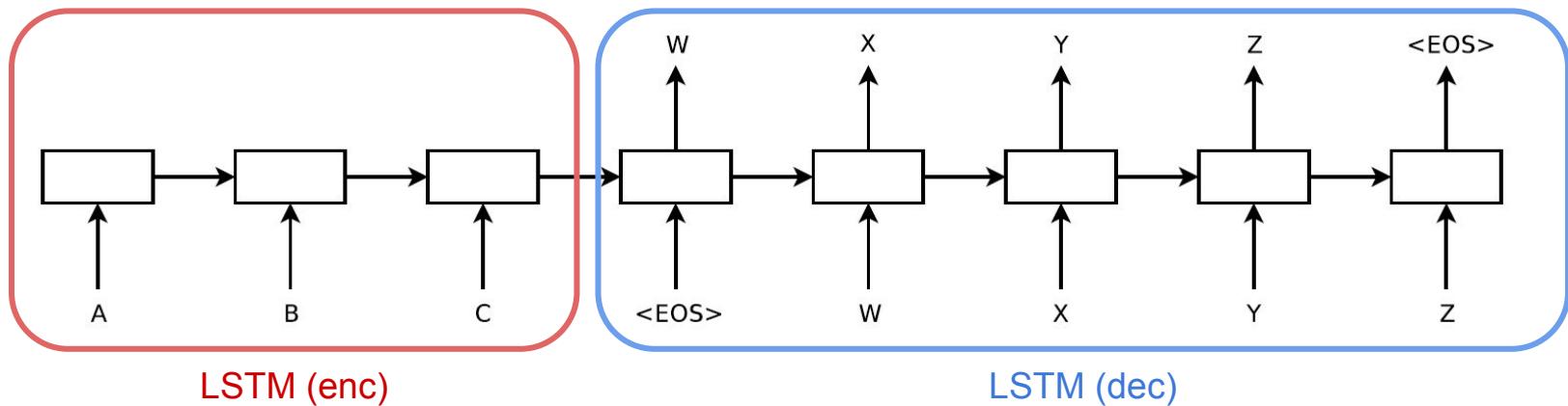
Google net...



CNN + LSTMのNW

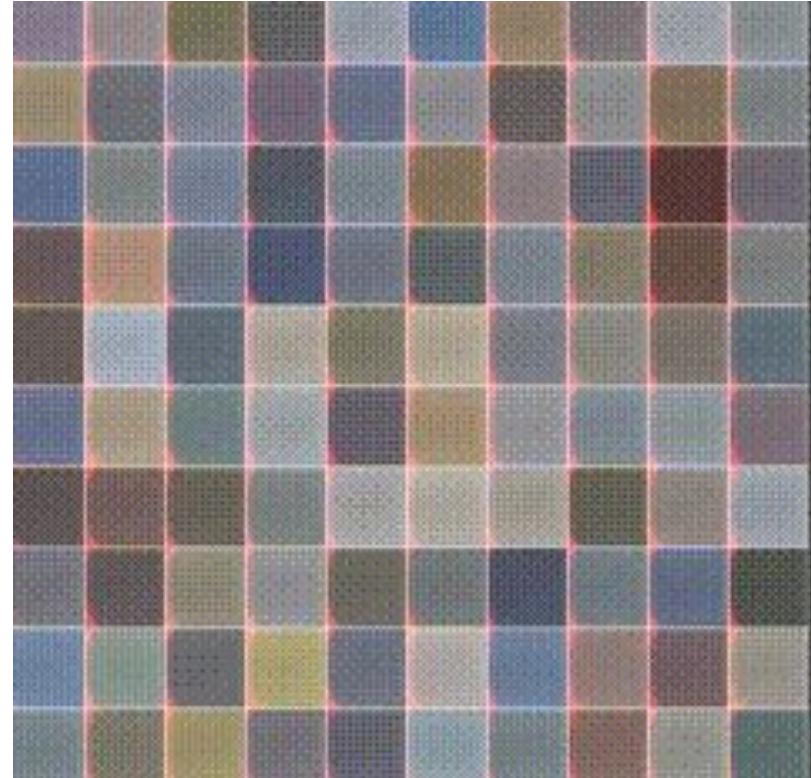
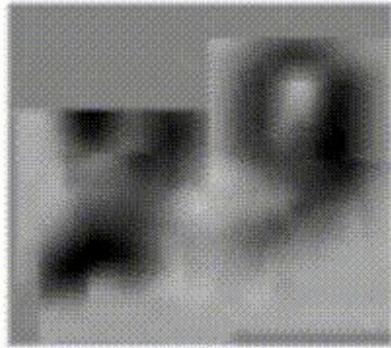
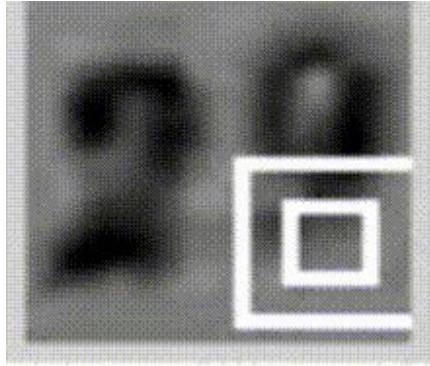
Seq 2 Seq [Sutskever+ 14]

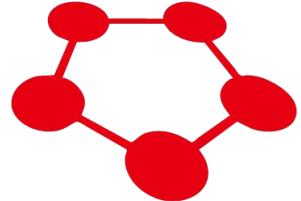
- NMT(Neural Machine Translation)における重要な手法
- 可変長なsequenceから可変長なsequenceを出力
- 2つのLSTM、エンコーダとデコーダを結合
- Trainとtestにおいて、入力系列を逆順にして与えると性能が向上する(後付的な理由に思える)



非時系列データを扱う [Ba+ 14][Gregor+ 15]

- 画像データ(非時系列)を時系列データとして処理
- House Numberを効率的に読む(*Visual Attention*) [Ba+ 14]
- (上記を応用し) MNISTやHouse Numberを生成 [Gregor+ 15]

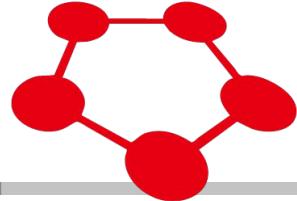




講義 Part3

LSTM・GRUの学習tips

Training tips for LSTM and GRU.



LSTMの実装: Chainer v3.4

素のLSTM
(chainer.links.LSTM)は
第二世代(99)の実装

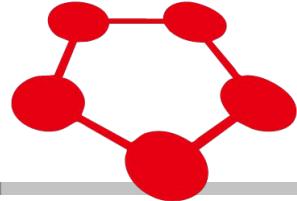
CEC

Input gate

Forget gate

Output gate

```
... 145 class LSTM(LSTMBase):
146
147     """Fully-connected LSTM layer.
148
149     This is a fully-connected LSTM layer as a chain. Unlike the
150     :func:`~chainer.functions.lstm` function, which is defined as a stateless
151     activation function, this chain holds upward and lateral connections as
152     child links.
153
154     def forward(self, inputs):
155         c_prev, x = inputs
156         a, i, f, o = _extract_gates(x)
157         batch = len(x)
158
159         if isinstance(x, numpy.ndarray):
160             self.a = numpy.tanh(a)
161             self.i = _sigmoid(i)
162             self.f = _sigmoid(f)
163             self.o = _sigmoid(o)
164
165             c_next = numpy.empty_like(c_prev)
166             c_next[:batch] = self.a * self.i + self.f * c_prev[:batch]
167             h = self.o * numpy.tanh(c_next[:batch])
168
169         else:
170             raise ValueError('Unsupported input type: %s' % type(x))
```



学習tips: Gradient Clipping

chainer.optimizer.GradientClipping

`class chainer.optimizer.GradientClipping(threshold) [source]`

Optimizer hook function for gradient clipping.

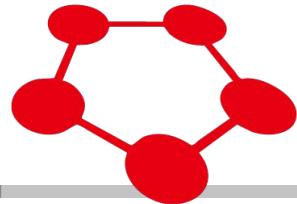
This hook function scales all gradient arrays to fit to the defined L2 norm threshold.

Parameters: `threshold (float)` – L2 norm threshold.

Variables: `threshold (float)` – L2 norm threshold of gradient norm.

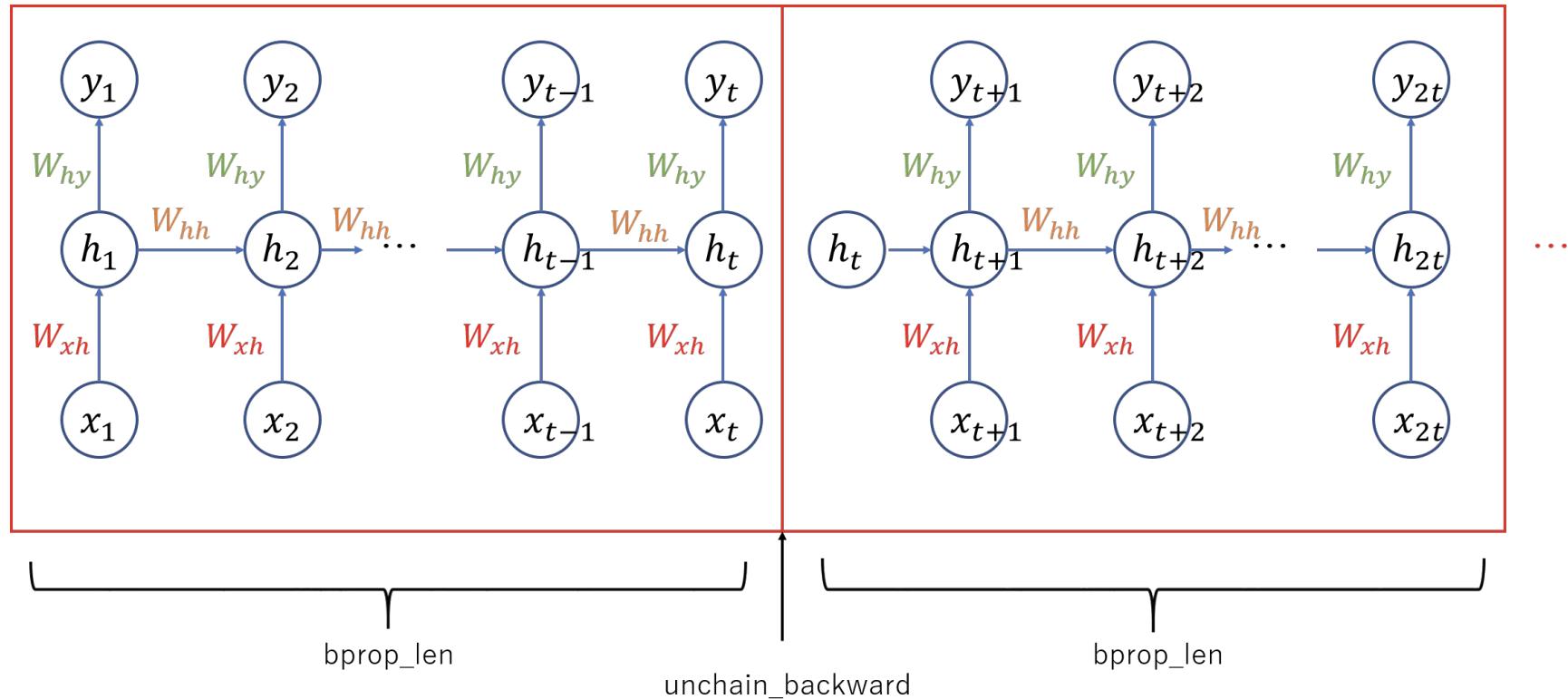
```
if || $\hat{g}$ || ≥ threshold then
     $\hat{g} \leftarrow \frac{\text{threshold}}{\|\hat{g}\|} \hat{g}$ 
endif
```

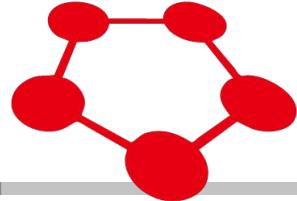
```
669 class GradientClipping(object):
...
693     def __call__(self, opt):
694         norm = numpy.sqrt(_sum_sqnorm(
695             [p.grad for p in opt.target.params(False)]))
rate = self.threshold / norm
if rate < 1:
    for param in opt.target.params(False):
        grad = param.grad
        with cuda.get_device_from_array(grad):
            grad *= rate
```



学習tips: Truncated Backprop

勾配計算を途中で断ち切る





学習tips: Truncated Backprop

勾配計算を途中で断ち切る

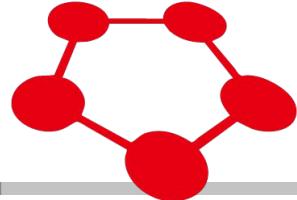
```
# train rnn with truncated backpropagation
def partial_fit():
    loss = 0
    count = 0
    seqlen = len(dataset)

    rnn.reset_state()

    for i in trange:
        cur_word = dataset[i]
        next_word = dataset[i+1]
        loss += model(cur_word, next_word)
        count += 1
        if count % 100 == 0 or count == seqlen:
            model.cleargrads()
            loss.backward()
            loss.unchain_backward()
            optimizer.update()
            print('Train Error: {:.4f}'.format(float(loss.data)))
    return loss
```

時系列データを与える

100 time stepごとに損失を
計算して、計算グラフを切り
離す(unchain)

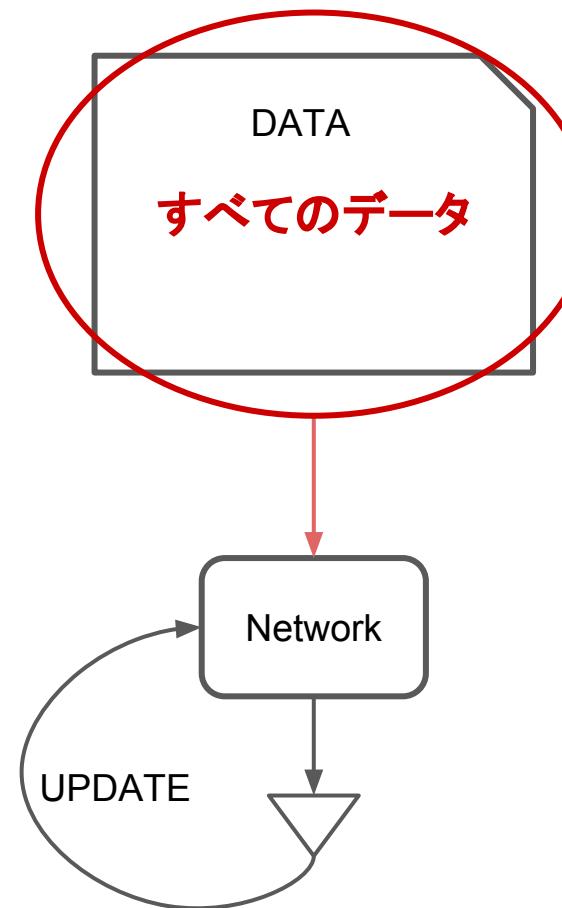


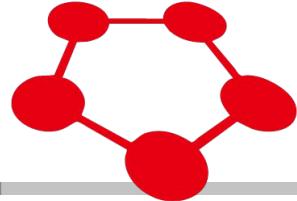
学習tips: ミニバッチの長さを揃える(1/3)

復習

ネットワークに対して、どのようにデータを与えるか、については以下の3つの方法が考えられる

1. **バッチ処理**
データすべてを与えてから重み更新



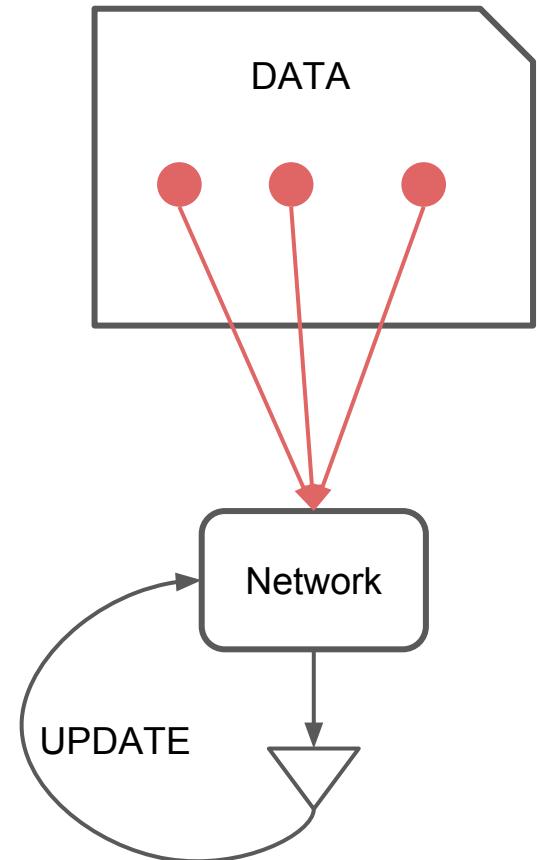


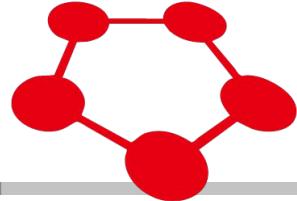
学習tips: ミニバッチの長さを揃える(1/3)

復習

ネットワークに対して、どのようにデータを与えるか、については以下の3つの方法が考えられる

1. バッチ処理
データすべてを与えてから重み更新
2. オンライン処理
一個づつデータを与えて重みを更新



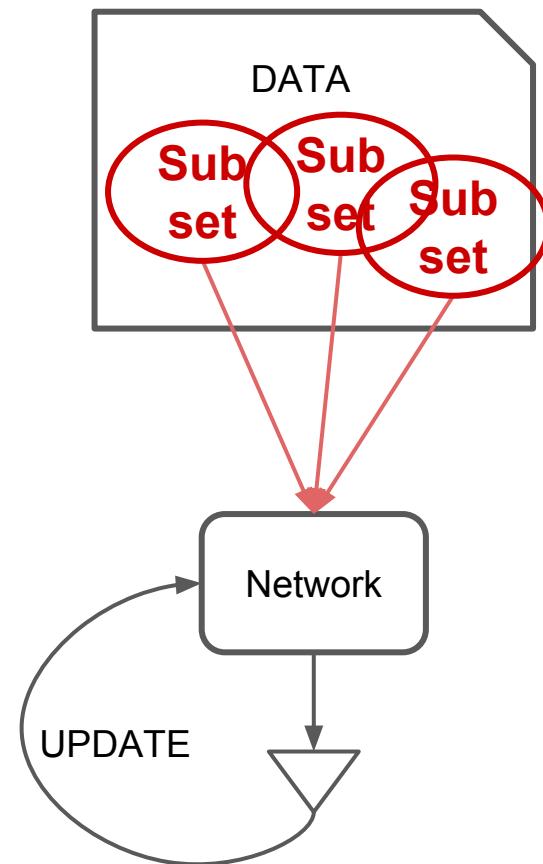


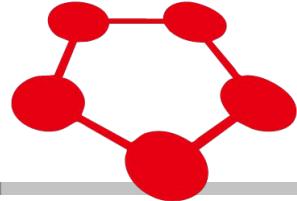
学習tips: ミニバッチの長さを揃える(1/3)

復習

ネットワークに対して、どのようにデータを与えるか、については以下の3つの方法が考えられる

1. バッチ処理
データすべてを与えてから重み更新
2. オンライン処理
一個づつデータを与えて重みを更新
3. **ミニバッチ処理**
サブセットをランダムに与えて重みを更新





学習tips: ミニバッチの長さを揃える(2/3)

I am no Jedi.

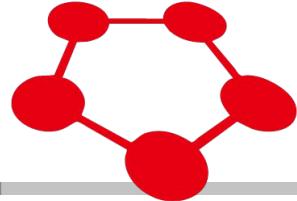
The Force will be with you.

Never tell me the odds!

このままでは長さが違うので、同時にNWで処理できない

①	②	③	④	⑤	⑥	⑦	⑧
The	Force	will	be	with	you	.	<EOS>
Never	tell	me	the	odds	!	<EOS>	
I	am	no	Jedi	.	<EOS>		

ここに共通のコードを埋め込む必要がある



学習tips: ミニバッチの長さを揃える(3/3)

chainer.functions.pad_sequence

```
chainer.functions.pad_sequence(xs, length=None, padding=0) [source]
```

Pad given arrays to make a matrix.

- Parameters:**
- `xs` (*list of ~chainer.Variable*) – Variables you want to concatenate.
 - `length` (*None or int*) – Size of the first dimension of a padded array. If it is `None`, the longest size of the first dimension of `xs` is used.
 - `padding` (*int or float*) – Value to fill.

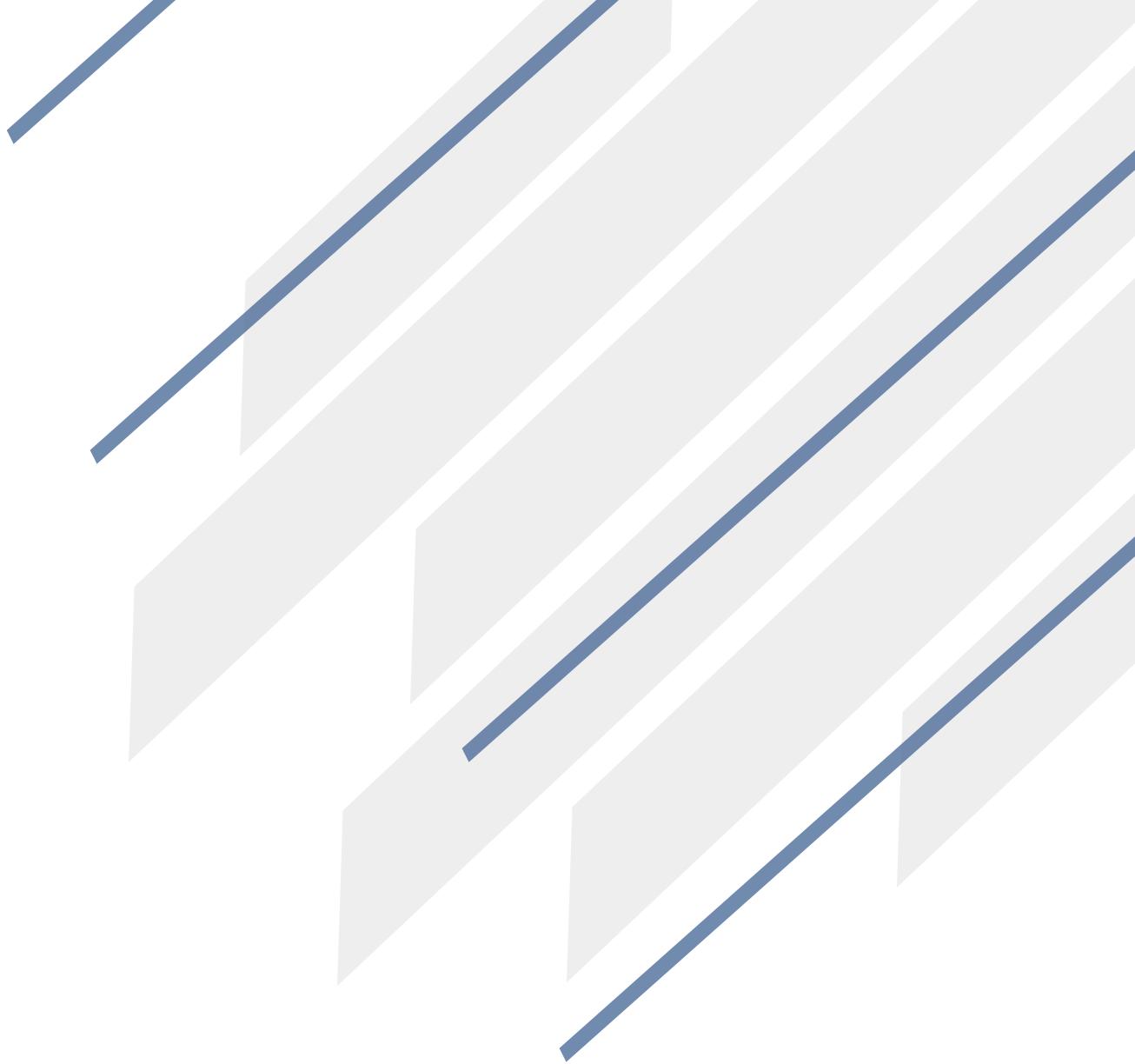
Returns:

It returns a padded matrix. Its shape is

`(n, length, ...)`, where `n == len(xs)`.

実践演習

Exercise

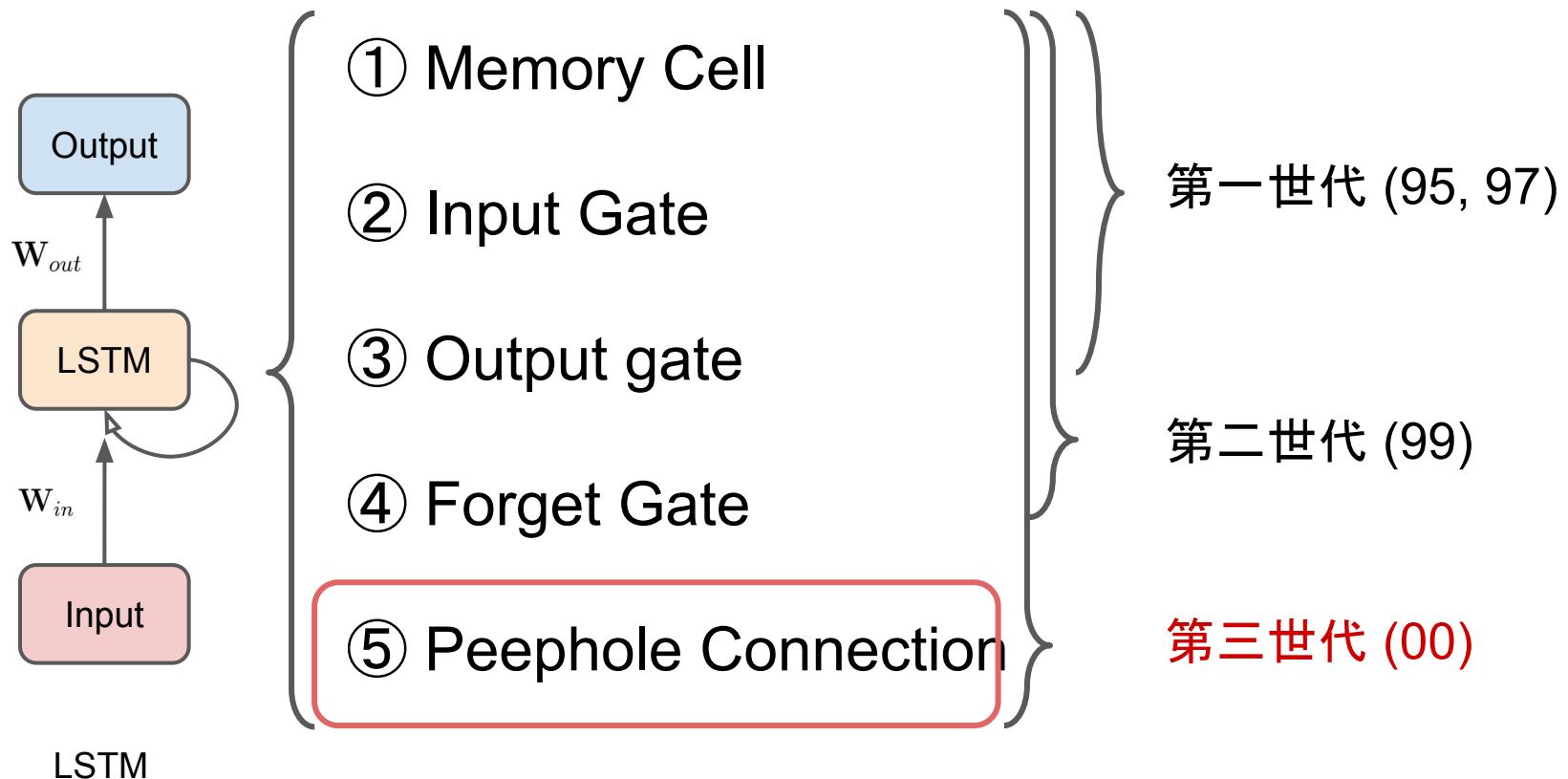


APPENDIX

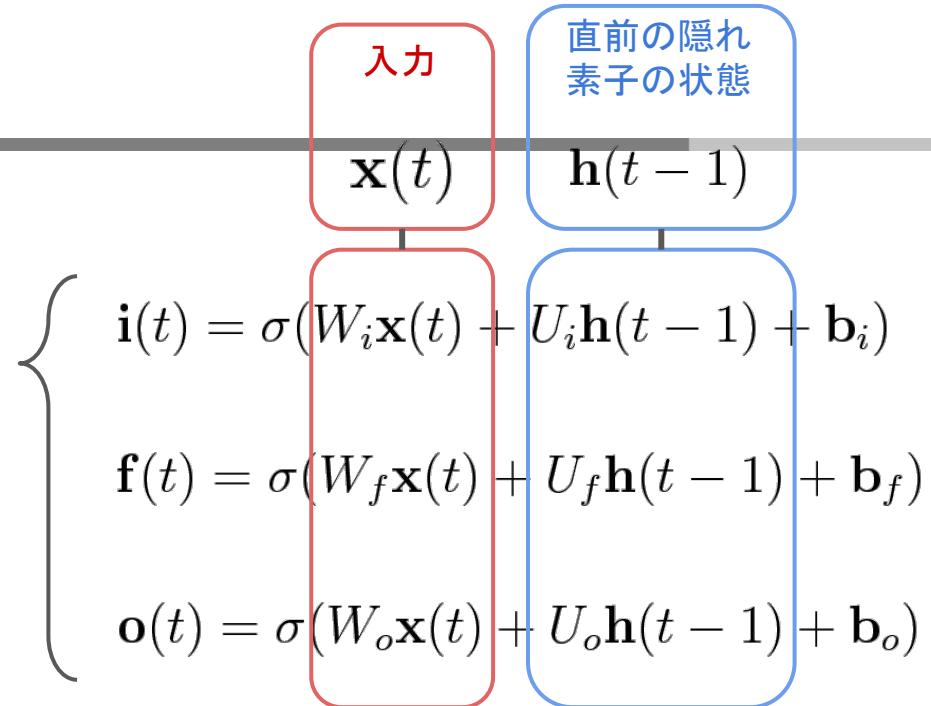
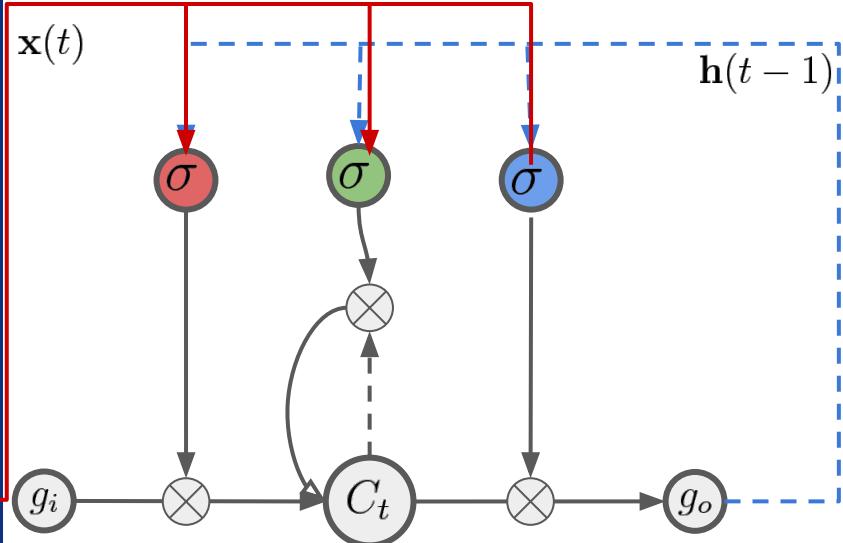
LSTMのきほん

<LSTM Blockの構成>

<世代>



GL3LSTM: 覗き穴の導入



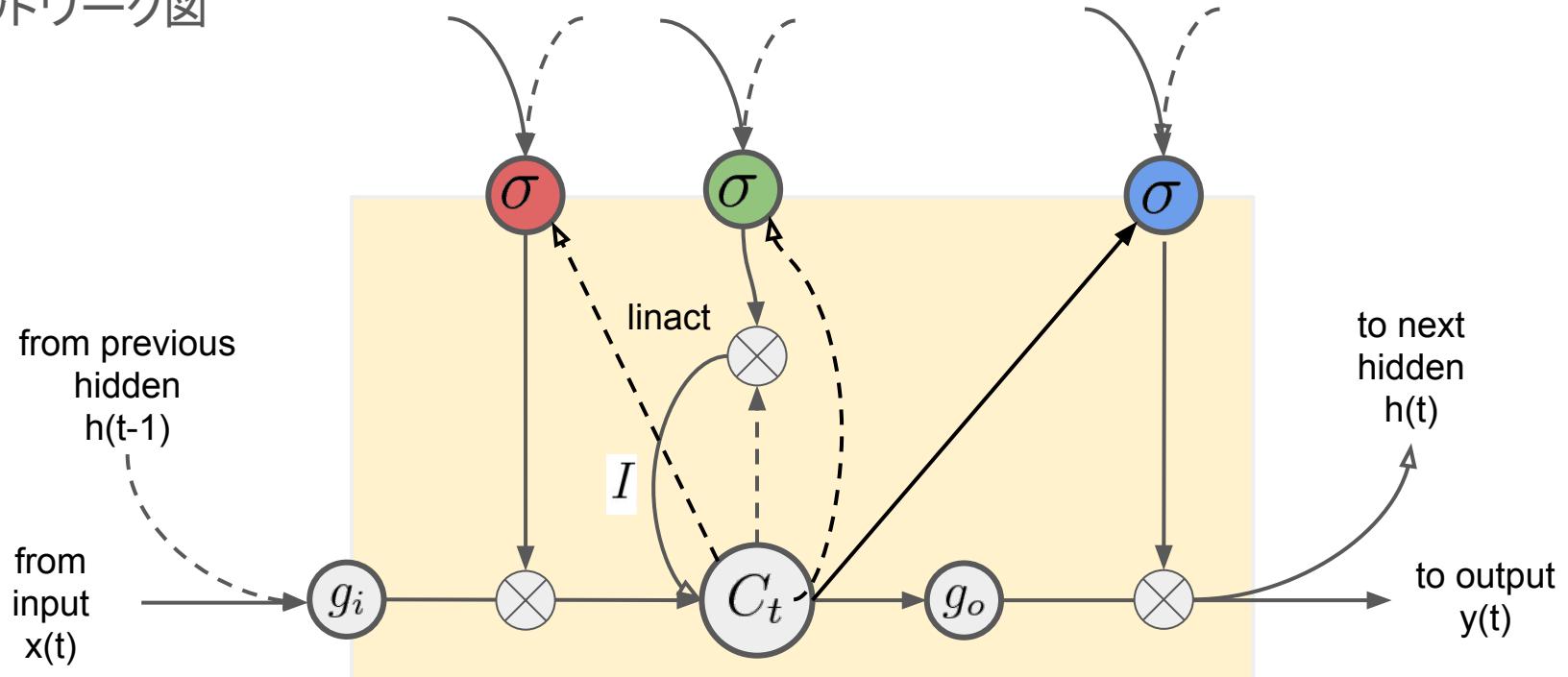
G1, G2 LSTM: 3つのゲートはメモリセルの状態を参照せず

メモリセルを制御するためにその状態を考慮するほうが望ましい

GL3LSTM: 3つのゲートはメモリセルの状態を参照する覗き穴を導入

G3LSTM: 覗き穴の導入

ネットワーク図

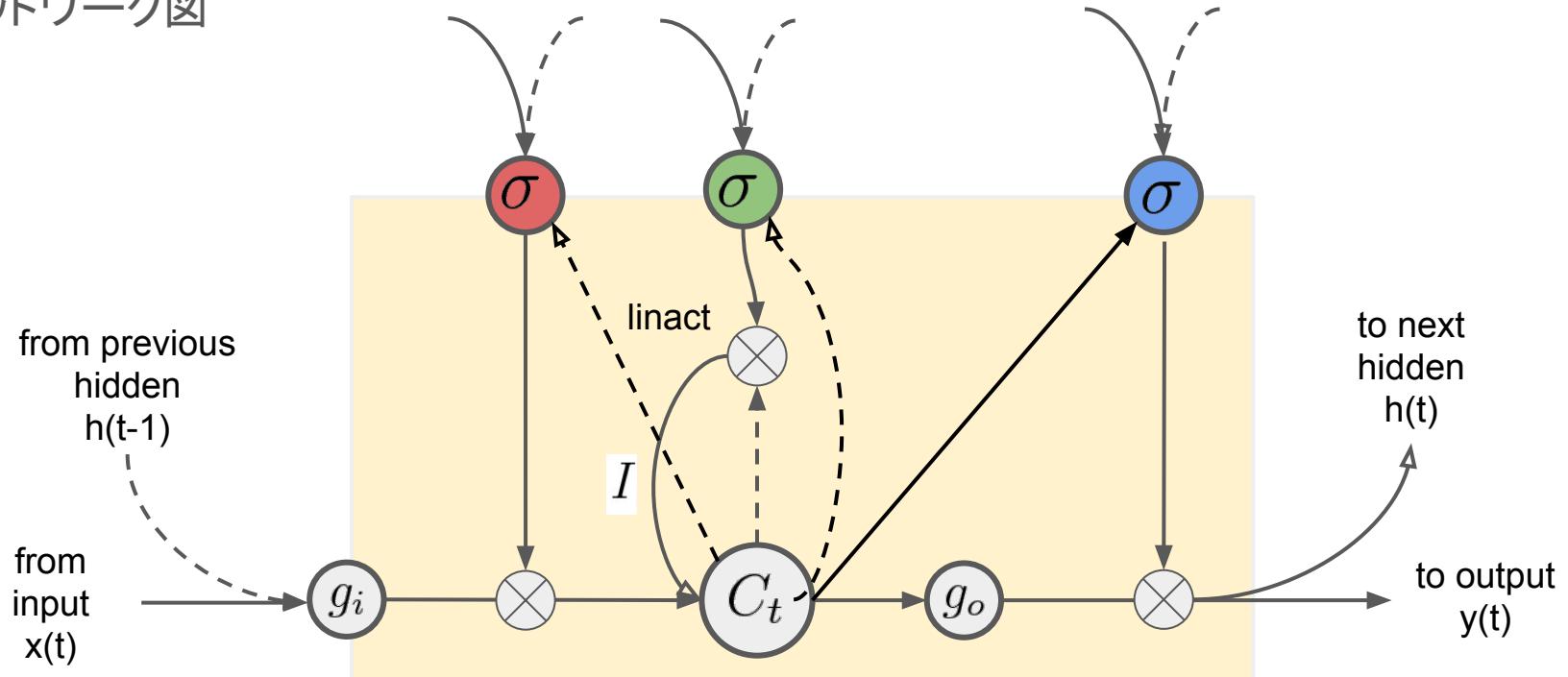


入力ゲートの順伝搬

$$\mathbf{i}(t) = \sigma(W_i \mathbf{x}(t) + U_i \mathbf{h}(t-1) + V_i \mathbf{c}(t-1) + \mathbf{b}_i)$$

G3LSTM: 覗き穴の導入

ネットワーク図

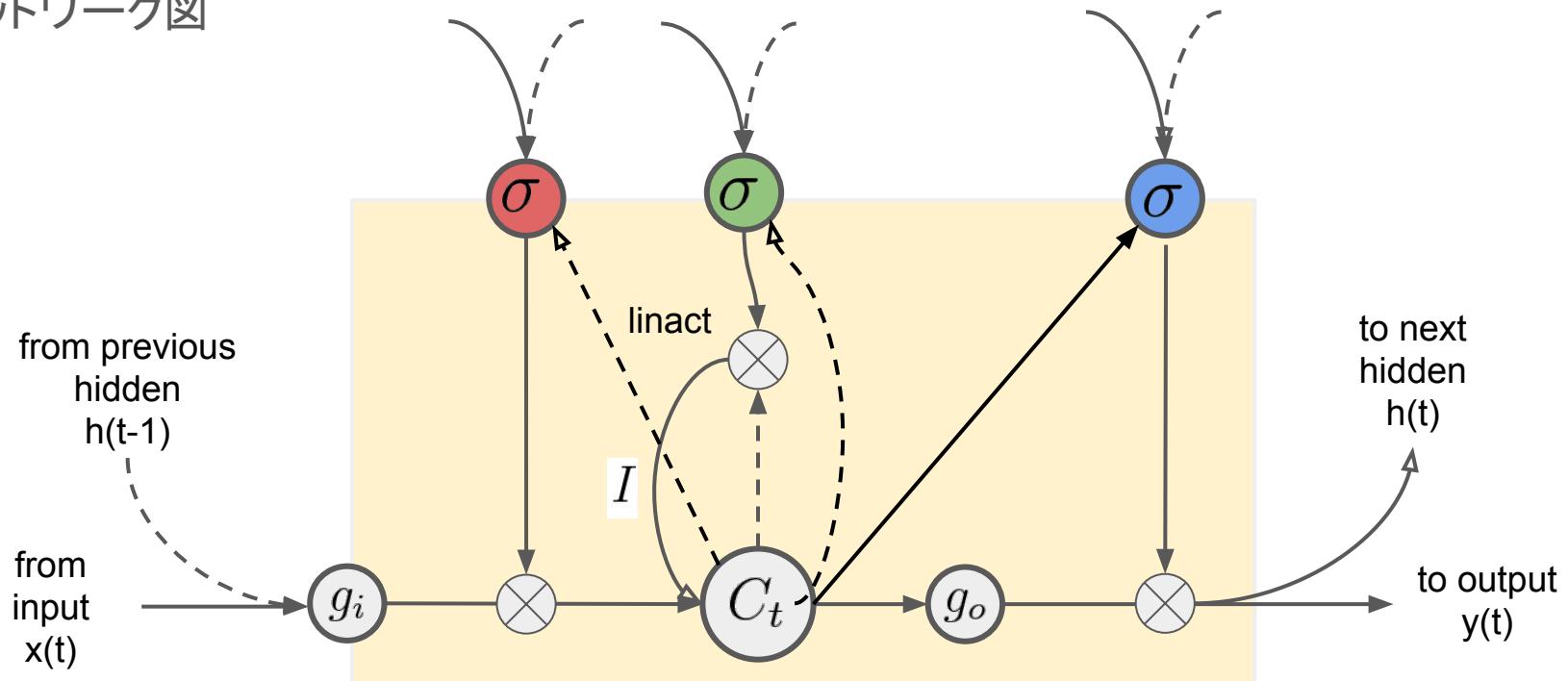


忘却ゲートの順伝搬

$$\mathbf{f}(t) = \sigma(W_f \mathbf{x}(t) + U_f \mathbf{h}(t-1) + V_f \mathbf{c}(t-1) + \mathbf{b}_f)$$

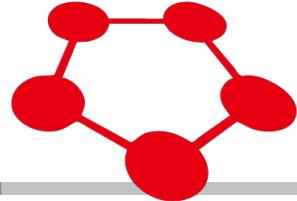
G3LSTM: 覗き穴の導入

ネットワーク図



出力ゲートの順伝搬

$$\mathbf{o}(t) = \sigma(W_o \mathbf{x}(t) + U_o \mathbf{h}(t-1) + V_o \mathbf{c}(t) + \mathbf{b}_o)$$



LSTMの実装: Chainer v3.4

chainer.links.StatefulPeepholeLSTM

`class chainer.links.StatefulPeepholeLSTM(in_size, out_size)` [\[source\]](#)

第三世代の実装も追加された

Fully-connected LSTM layer with peephole connections.

This is a fully-connected LSTM layer with peephole connections as a chain. Unlike the [LSTM](#) link, this chain holds `peep_i`, `peep_f` and `peep_o` as child links besides `upward` and `lateral`.

Given a input vector x , Peephole returns the next hidden vector h' defined as

$$\begin{aligned}
 a &= \tanh(upwardx + lateralh), \\
 i &= \sigma(upwardx + lateralh + peep_i c), \\
 f &= \sigma(upwardx + lateralh + peep_f c), \\
 c' &= a \odot i + f \odot c, \\
 o &= \sigma(upwardx + lateralh + peep_o c'), \\
 h' &= o \tanh(c'),
 \end{aligned}$$

109
110
111
112
113
114
115
116
117
118
119
120
121
122

```

a, i, f, o = split_axis.split_axis(lstm_in, 4, 2)
a = reshape.reshape(a, (len(a.data), a.shape[1]))
i = reshape.reshape(i, (len(i.data), i.shape[1]))
f = reshape.reshape(f, (len(f.data), f.shape[1]))
o = reshape.reshape(o, (len(o.data), o.shape[1]))
peep_in_i = self.peep_i(self.c)
peep_in_f = self.peep_f(self.c)
a = tanh.tanh(a)
i = sigmoid.sigmoid(i + peep_in_i)
f = sigmoid.sigmoid(f + peep_in_f)
self.c = a * i + f * self.c
peep_in_o = self.peep_o(self.c)
o = sigmoid.sigmoid(o + peep_in_o)
self.h = o * tanh.tanh(self.c)

```

PEEPHOLEの実装部分

IRNN: 単位行列による初期化 [Jaitly&Hinton 15]

- 勾配消失・爆発問題に対するアプローチとして

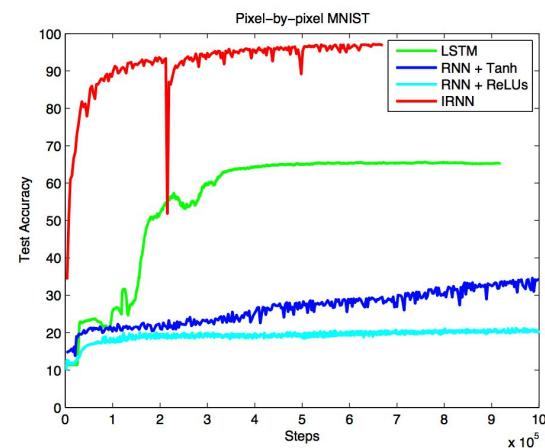
- ReLU関数を導入
- 再帰重みを単位行列で初期化
- バイアスを0で初期化

直前の値をコピー
BPTTの微分項が1になるため勾配が爆発、消失しにくい

- 単純なRNNで長期依存データの学習が可能

Methods	Test perplexity
LSTM (512 units)	68.8
IRNN (4 layers, 512 units)	69.4
IRNN (1 layer, 1024 units + linear projection with 512 units before softmax)	70.2
RNN (4 layer, 512 tanh units)	71.8
RNN (1 layer, 1024 tanh units + linear projection with 512 units before softmax)	72.5

言語モデルの学習結果(↑)
MNISTの学習結果(→)



LSTM・GRUの現状

参考文献の表記

参考文献は以下のように表記されています

- 論文の場合: [Hinton+ 12]
 - APPENDIXのReferencesはAPA形式
- 本の場合: [Asakawa: 16]
 - APPENDIXのReferencesはAPA形式
- WEBページの場合: [Karpathy:: 17]
 - APPENDIXのReferencesはAPA形式

References

Note: Lecture materials are available on google drive

- [Gregor+ 15] Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., & Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. arXiv preprint arXiv:1502.04623.
- [Ba+ 14] Ba, J., Mnih, V., & Kavukcuoglu, K. (2014). Multiple object recognition with visual attention. arXiv preprint arXiv:1412.7755.
- [Vinyals+ 15] Vinyals, O., Toshev, A., Bengio, S., & Erhan, D. (2015). Show and tell: A neural image caption generator. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3156-3164).
- [Lin+ 14] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.
- [Graves+ 14] Graves, A., Mohamed, A. R., & Hinton, G. (2013, May). Speech recognition with deep recurrent neural networks. In Acoustics, speech and signal processing (icassp), 2013 ieee international conference on (pp. 6645-6649). IEEE.

References

- [Strobelt+ 16] Strobelt, H., Gehrman, S., Huber, B., Pfister, H., & Rush, A. M. (2016). Visual analysis of hidden state dynamics in recurrent neural networks. arXiv preprint arXiv:1606.07461.
- [Sutskever+ 14] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems(pp. 3104-3112).
- [Maas+ 13] Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013, June). Rectifier nonlinearities improve neural network acoustic models. In Proc. ICML (Vol. 30, No. 1).
- [Andrey:: 17] <https://hackernoon.com/how-to-debug-neural-networks-manual-dc2a200f10f2>
- [Strobelt+:: 16] <http://lstm.seas.harvard.edu/>
- [Cho+ 14] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.