現場で使えるディープラーニング基礎講座

DAY6

SkillUP AI

前回の復習

- 前回は何を学びましたか？

機械学習で扱うデータと典型的なタスク

目次

機械学習で扱うデータと典型的なタスク

- 機械学習では様々なデータを扱うが、ここではその代表として画像データ、時系列データ、テキストデータを取り上げ、それらデータの特徴と典型的なタスクを紹介する。

画像データ

画像データ

- 画像データの例
  - 写真、手書き文字、MRI画像
- 画像データを扱う際によく行う前処理
  - 画像を切り出す
    - 対象物が写っているところを中心にする
  - サイズを揃える
  - 解像度を小さくする
  - グレースケールに変換する

画像データを用いた典型的なタスク

- 画像認識(クラス分類)
  - 例、入力された写真をネコ、イヌ、トラなどのクラスに分類する。
- 物体検出
  - 例、入力された写真に対し、人、自動車、樹木など、あらかじめ定められたクラスの物体を四角い領域で特定する。
- セマンティックセグメンテーション
  - 例、入力された写真に対し、その写真のピクセルごとに、人、自動車、樹木などのクラス分類を行う。
- 画像生成
  - 例、全く新しい画像(顔写真など)を生成する。
- 画像キャプション生成
  - 例、人が写っている写真に対し、その人が何をしているかを説明する自然言語を生成する。

オープンな画像データ

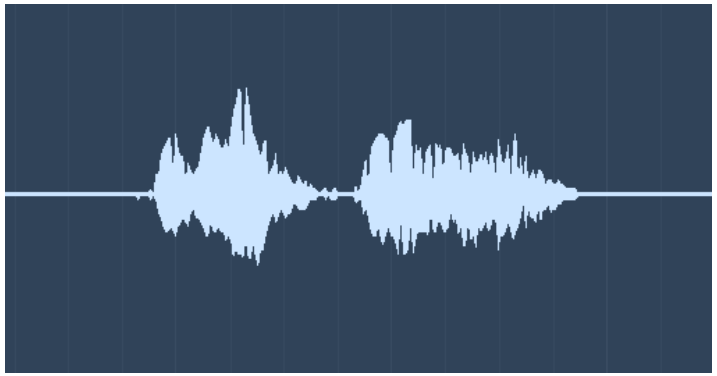
- MNIST
  - http://yann.lecun.com/exdb/mnist/
  - 28×28サイズのグレースケール画像データ
  - ライセンスはCC0
  - アメリカ国立標準技術研究所(NIST)が整備したデータ
- ImageNet
  - http://image-net.org
  - 画像へのリンクが格納されているデータベースであり、そのデータベース利用は非商用に限られている。
  - リンク先の画像については、撮影者に著作権があり、ImageNetは関与しないという立場。
    - http://image-net.org/about-overview
- CIFAR-10, CIFAR-100
  - https://www.cs.toronto.edu/~kriz/cifar.html
  - 32×32サイズのカラー画像とラベルのデータセット
  - CIFAR-10は10種類のラベル、 CIFAR-100は100種類のラベルを持つ。
- COCO
  - http://cocodataset.org/#home
  - Microsoft社が提供しているデータセット
  - 30万点以上の画像に、80種類の物体の領域情報に加えて画像の内容について説明したテキストも含まれている。
- Open Images dataset
  - https://storage.googleapis.com/openimages/web/index.html
  - Google社が提供している約900万点の画像URLとそられについてのラベルや物体領域などのデータ。
  - 画像のライセンスはCCBY2.0。

Any Questions?

時系列データ

時系列データ

- 時系列データの例
  - 株価データ、気象データ、センサーデータ、音声データ、販売数データ、音楽データ
- 時系列データを扱う際によく行う前処理
  - 欠損値の補完
  - 異常値の除去
  - データ間隔のリサンプル
    - 例、時間別データを日別データに変換
  - 周波数空間への変換
    - フーリエ変換、ウェーブレット変換



音声データの例
何を発言した時の波形でしょうか？
ヒント：アルファベット9文字

答え：skillupai

時系列データを用いた典型的なタスク

[センサーデータ]

- 予測
  - 例、電力需要を予測する。
- 故障検知
  - 例、機器の故障を検知する。

[音声データ]

- 音声認識
  - 例、発話のデータに含まれる単語列を推定する。
- 自動通訳
  - 例、日本語を英語に翻訳する。
- 話者分離
  - 例、それぞれの話者が同時にしゃべったときに、人毎に発話内容をわかける。

[音楽データ]

- 音楽の変換
  - 例、ある音楽をバッハ風の音楽に変換する。
- 音楽の生成
  - 例、全く新しい音楽を生成する。

Any Questions?

テキストデータ

テキストデータ

- テキストデータの例
  - 小説、新聞、SNSの投稿文
- テキストデータを扱う際によく行う前処理
  - 不要文字列の除去、置換
  - 文章の単語分割(分かち書き)
  - 正規化処理(ﾘﾝｺﾞ, リンゴ, りんご, 林檎 -> リンゴ)
  - ストップワードの除去
    - 一般的で役に立たない単語などを除去すること(「は」「です」「を」)
- コーパス
  - 一般的に、自然言語処理の研究を目的に集められたテキストデータのことをコーパスと呼ぶ。学習時には、大量のテキストデータが必要になることが多く、そういった場合にコーパスを活用する。

テキストデータを用いた典型的なタスク

- トピック抽出
  - 例、ある新聞記事のトピックを抽出する。ニュース、政治経済、スポーツ、、、
- 文章分類
  - 例、似ている新聞記事をグルーピングする。
- 文章要約
  - 例、ある新聞記事の内容を理解し、100文字で表現する。
- 機械翻訳
  - 例、日本語を英語に翻訳する。
- 文章生成
  - 例、全く新しい小説を生成する。
- 対話
  - 例、チャットボット。

オープンなテキストデータ

- 青空文庫
  - https://www.aozora.gr.jp/
  - ボランティアによって運営されているインターネット文庫
  - 著作権切れの小説が約14000作品公開されている
- livedoor ニュースコーパス
  - https://www.rondhuit.com/download.html#ldcc
  - クリエイティブ・コモンズライセンス

Any Questions?

データの権利

何らかのデータを学習に使用する際はライセンスと著作権に注意

- 何らかのデータを学習に使用する際は、そのデータのライセンスと著作権に注意すること。
- ライセンスの形態としては、クリエイティブ・コモンズ・ライセンスなどがある。
  - 参考：クリエイティブ・コモンズ・ライセンス
    - https://creativecommons.jp/licenses/
- 著作権については、著作権利者の所在国ではなく、学習行為を行う国の著作権法が適応される。なお、日本の著作権法には例外規定があるので、それも併せて確認しておくとよい。
  - 参考：著作権法の例外規定　(著作権法 第30条の4、2019年1月1日施行)
    - 「著作物は，技術の開発等のための試験の用に供する場合，情報解析の用に供する場合，人の知覚による認識を伴うことなく電子計算機による情報処理の過程における利用等に供する場合その他の当該著作物に表現された思想又は感情を自ら享受し又は他人に享受させることを目的としない場合には，その必要と認められる限度において，利用することができることを規定しています。これにより，例えば人工知能（AI）の開発のための学習用データとして著作物をデータベースに記録する行為等，広く著作物に表現された思想又は感情の享受を目的としない行為等を権利者の許諾なく行えることとなるものと考えられます。」
      https://www.bunka.go.jp/seisaku/chosakuken/hokaisei/h30_hokaisei/　より引用
  - 参考：「日本は機械学習パラダイス」　その理由は著作権法にあり
    http://www.itmedia.co.jp/news/articles/1710/10/news040.html

Any Questions?

再帰型ニューラルネットワーク

目次

再帰型ニューラルネットワーク概要

再帰型ニューラルネットワークとは

- 再帰型ニューラルネットワーク(recurrent neural network)とは、時間方向に状態を引き継ぎながら計算を進めることができるニューラルネットワーク。
- 自然言語などのように単語が順番に並んでいるデータを自然に扱うことができるため、自然言語処理や音声認識に向いている。
- 時間方向に計算するネットワークであるため、「時系列の数値データをディープラーニングで計算するときはRNN」と言われることがあるが、自己回帰を考慮することは通常の全結合型ニューラルネットワークでもモデル化可能。
- 自然言語のように入力長が毎回異なるデータを扱う場合は、全結合型ニューラルネットワークよりもRNNの方が向いている。

再帰型ニューラルネットワークの種類

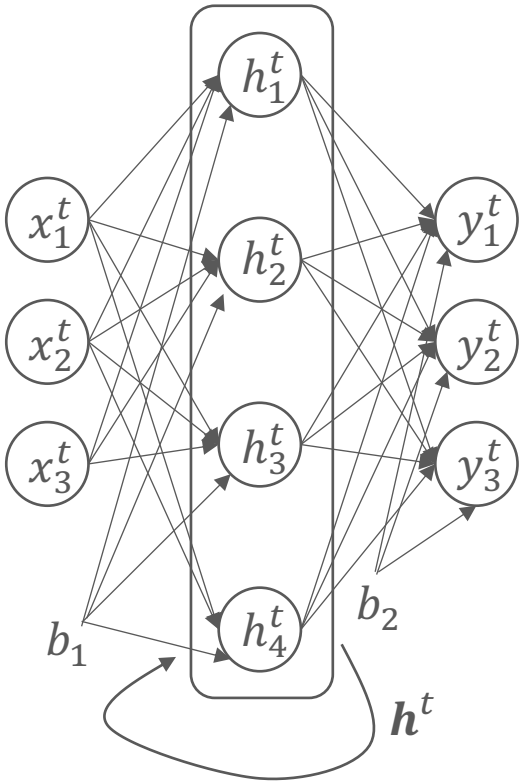
- シンプルなRNN
  - 単純な全結合層を用いて状態を更新していく。
  - 長い系列を学習しても、過去の情報がほとんど反映されないという欠点がある。
- LSTM (Long Short-Term Memory)
  - シンプルなRNNの課題を解決するために提案されたRNN。
  - 入力ゲート(input gate)、出力ゲート(output gate)、忘却ゲート(forget gate)、記憶セル(memory cell)と呼ばれる仕組みが導入され、必要な情報を長く記憶できるようになった。
  - “Long Short-Term Memory”とは、”短期記憶を長い時間継続できること”を意味する。
- GRU(Gated Recurrent Unit)
  - LSTMをシンプルにしたRNNであり、LSTMよりもパラメータの数が少ない。
  - リセットゲート(reset gate)と更新ゲート(update gate)のみで構成される。

RNNの計算グラフ表現

x^t 右上の添字は時刻を表す
(大文字のTは最終時刻)

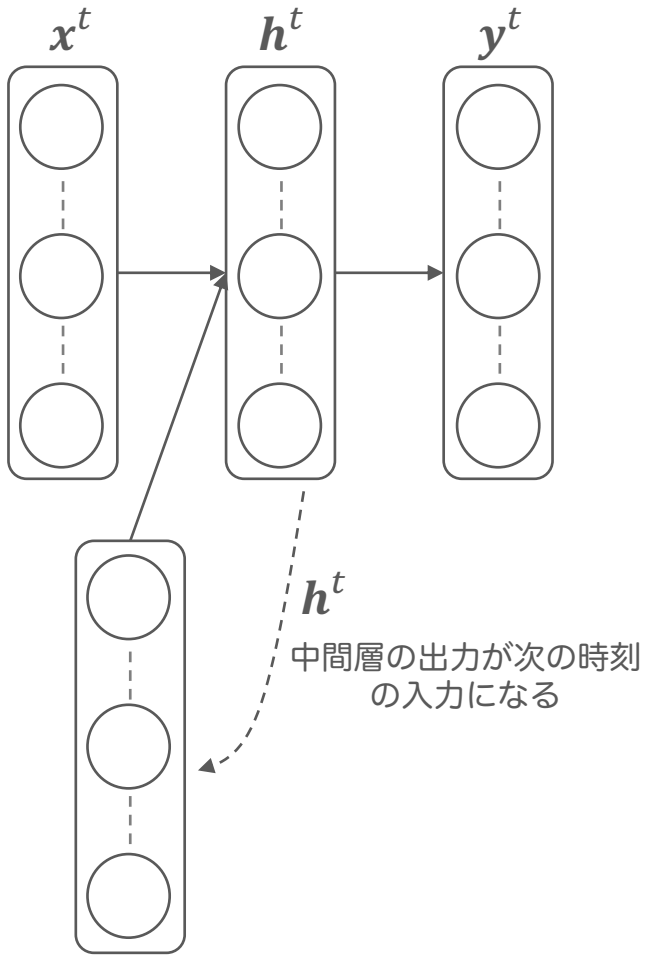
- RNNの計算グラフは、簡略化して表現されることが多い。

$$h^t = f(h^{t-1}W_h + x^t W_x + b)$$
f：活性化関数



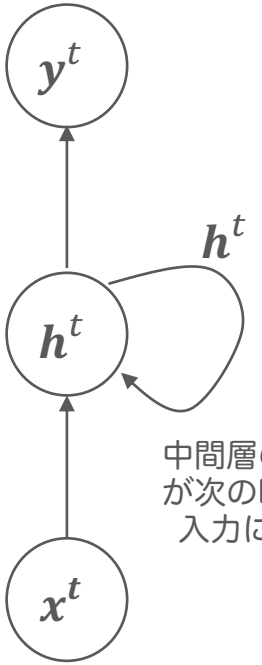
中間層の出力が次の時刻
の中間層への入力になる

簡略化する



中間層の出力が次の時刻
の入力になる

もっと
簡略化する

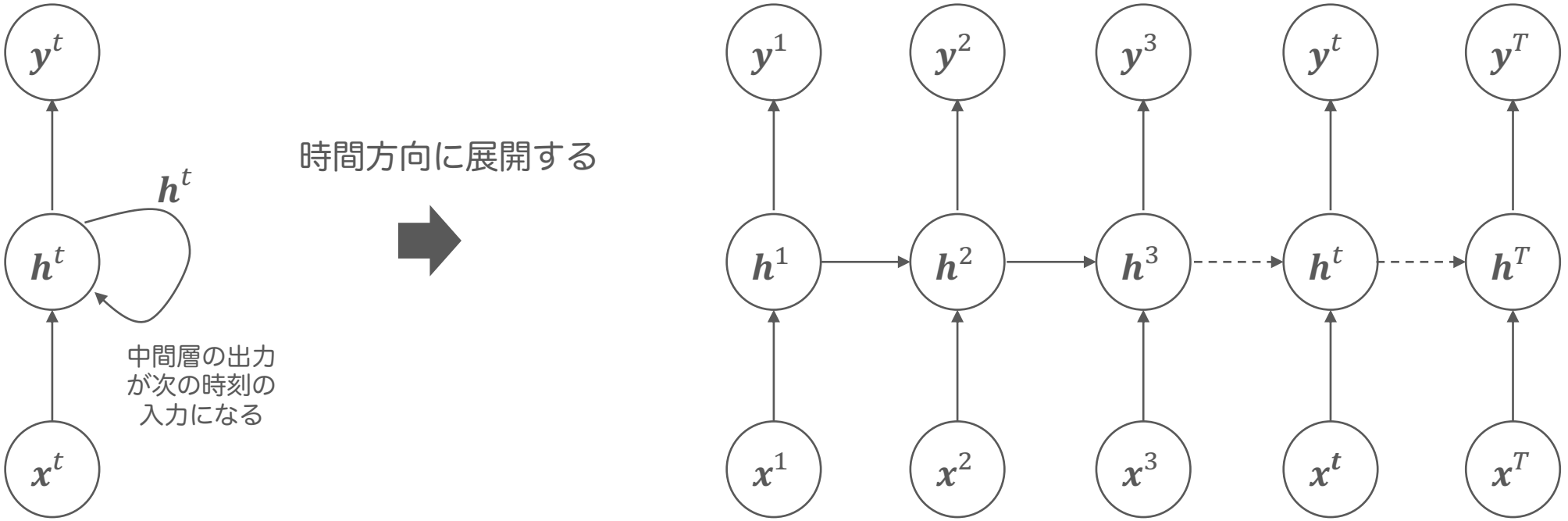


中間層の出力
が次の時刻の
入力になる

RNNの計算グラフの展開

x^t 右上の添字は時刻を表す

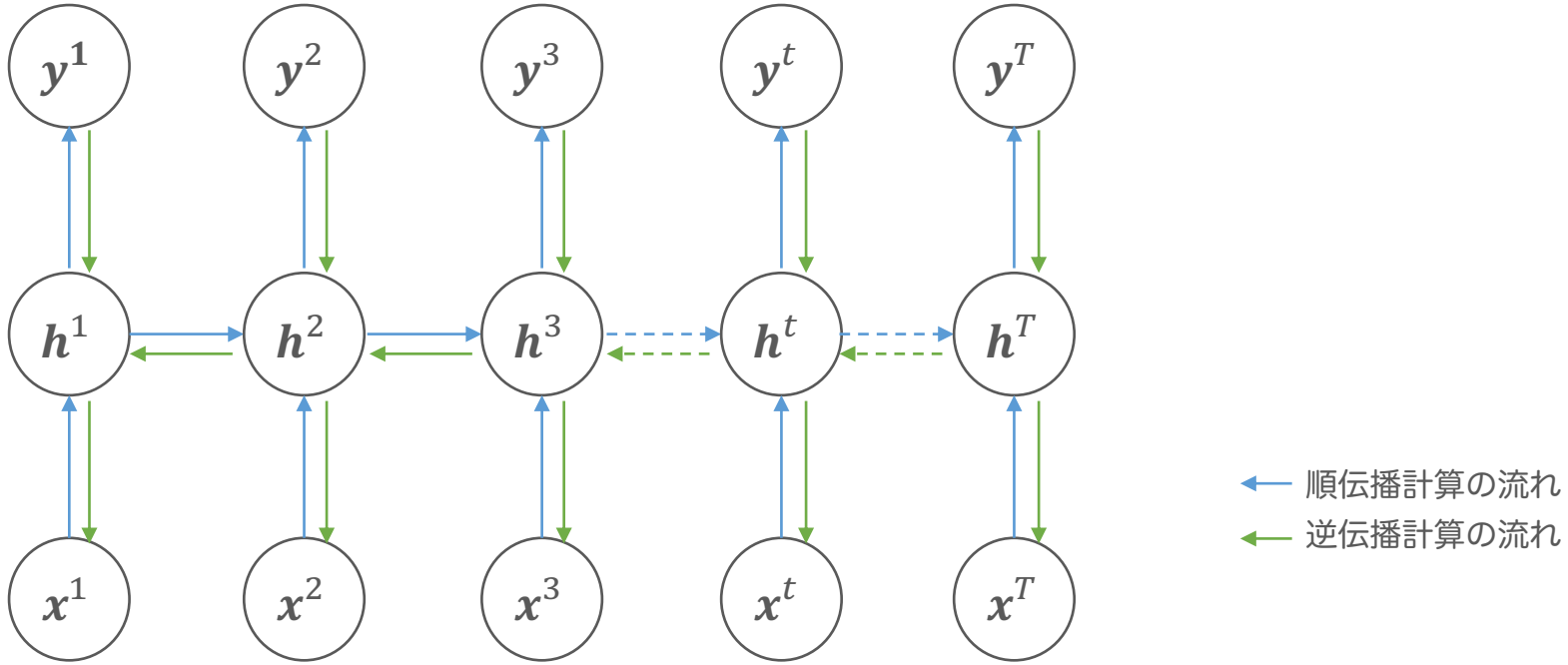
- RNNの計算グラフを展開すると以下のようになる。



RNNの誤差逆伝播法

x^t 右上の添字は時刻を表す

- 展開された計算グラフは通常のNNなので、誤差逆伝播法により勾配を求められる。
- 展開された計算グラフに対しての誤差逆伝播法をBPTT(backpropagation through time) という。



Any Questions?

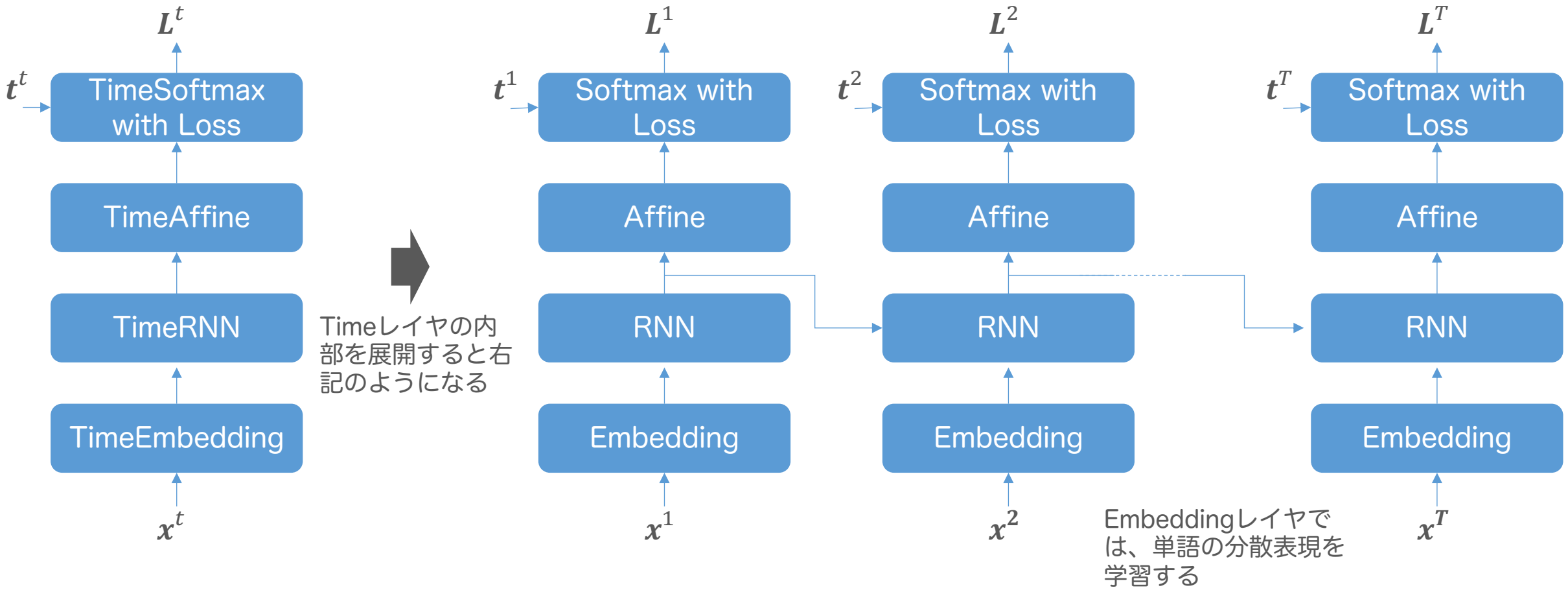
シンプルなRNN

本節の概要

- 本節では、シンプルなRNNについて説明する。
- まず、スライドで仕組みを説明し、次に、Notebookで実装していく。
- Notebookでは、個別のレイヤを実装していき、その後に、実装した個別のレイヤを使って言語モデルを構築する。
- その言語モデルは、SimpleRNNLM(Simple RNN language model)というクラス名で実装する。
- 構築した言語モデルの学習には、 Penn TreeBank データセットを用いる。

SimpleRNNLMクラス

- SimpleRNNLMクラスの構成を以下に示す。



Penn TreeBank データセット

- Penn TreeBank データセットは、英語のコーパス。
- 総単語数90万語、異なり単語数1万語の小規模なデータセット。
- ダウンロード元
  - Train用データ
    - https://raw.githubusercontent.com/tomsercu/lstm/master/data/ptb.train.txt
  - Test用データ
    - https://raw.githubusercontent.com/tomsercu/lstm/master/data/ptb.test.txt
  - Valid用データ
    - https://raw.githubusercontent.com/tomsercu/lstm/master/data/ptb.valid.txt

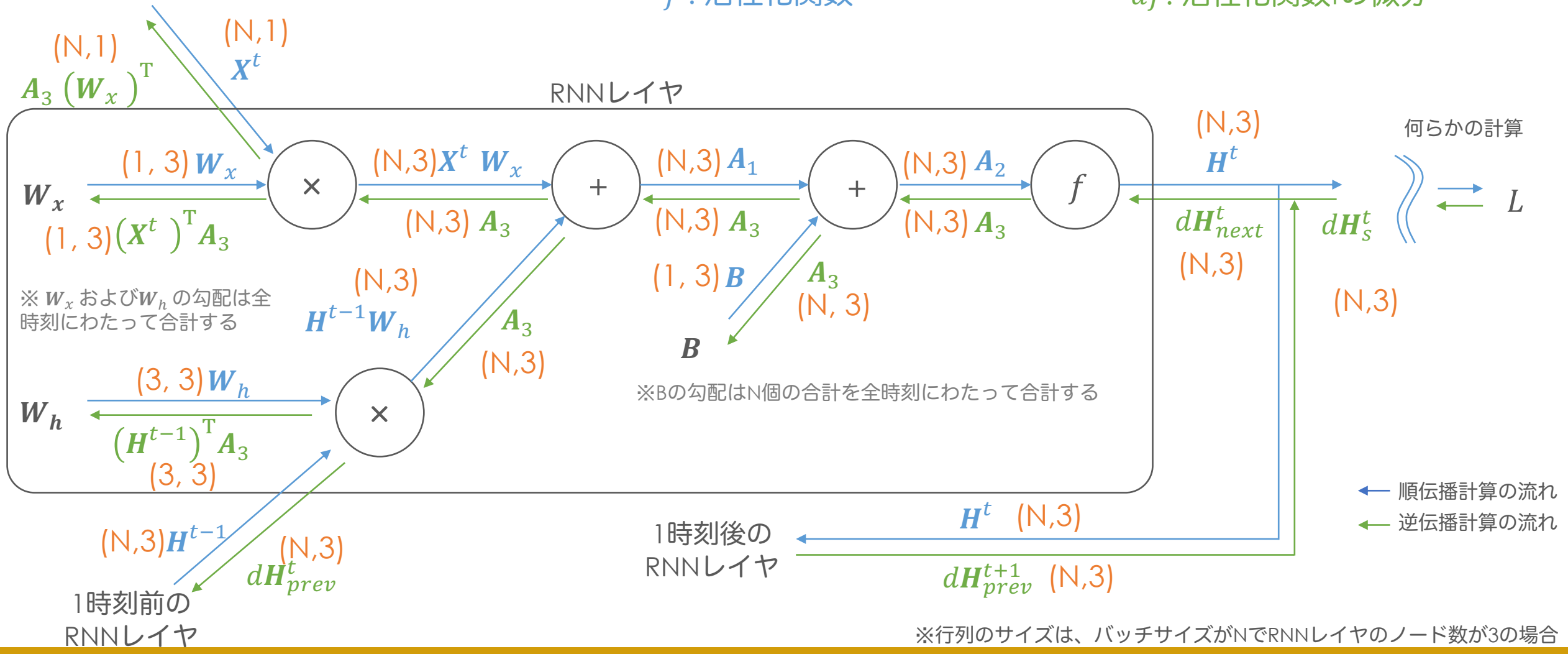
RNNレイヤとTimeRNNレイヤ

- シンプルなRNNの実装を考える。
- ある時刻t における順伝播計算および逆伝播計算を行うクラスをRNNレイヤという名前で実装することにする。
  - RNNレイヤの計算グラフを次頁に示す。
- 全ての時刻のRNNレイヤをまとめるクラスをTimeRNNレイヤという名前で実装することにする。
  - TimeRNNレイヤの計算グラフを2頁後に示す。

RNNレイヤの計算グラフ

$$H^t = f(H^{t-1}W_h + X^t W_x + B)$$
$$A_1 = H^{t-1}W_h + X^t W_x$$
$$A_2 = A_1 + B$$
f：活性化関数

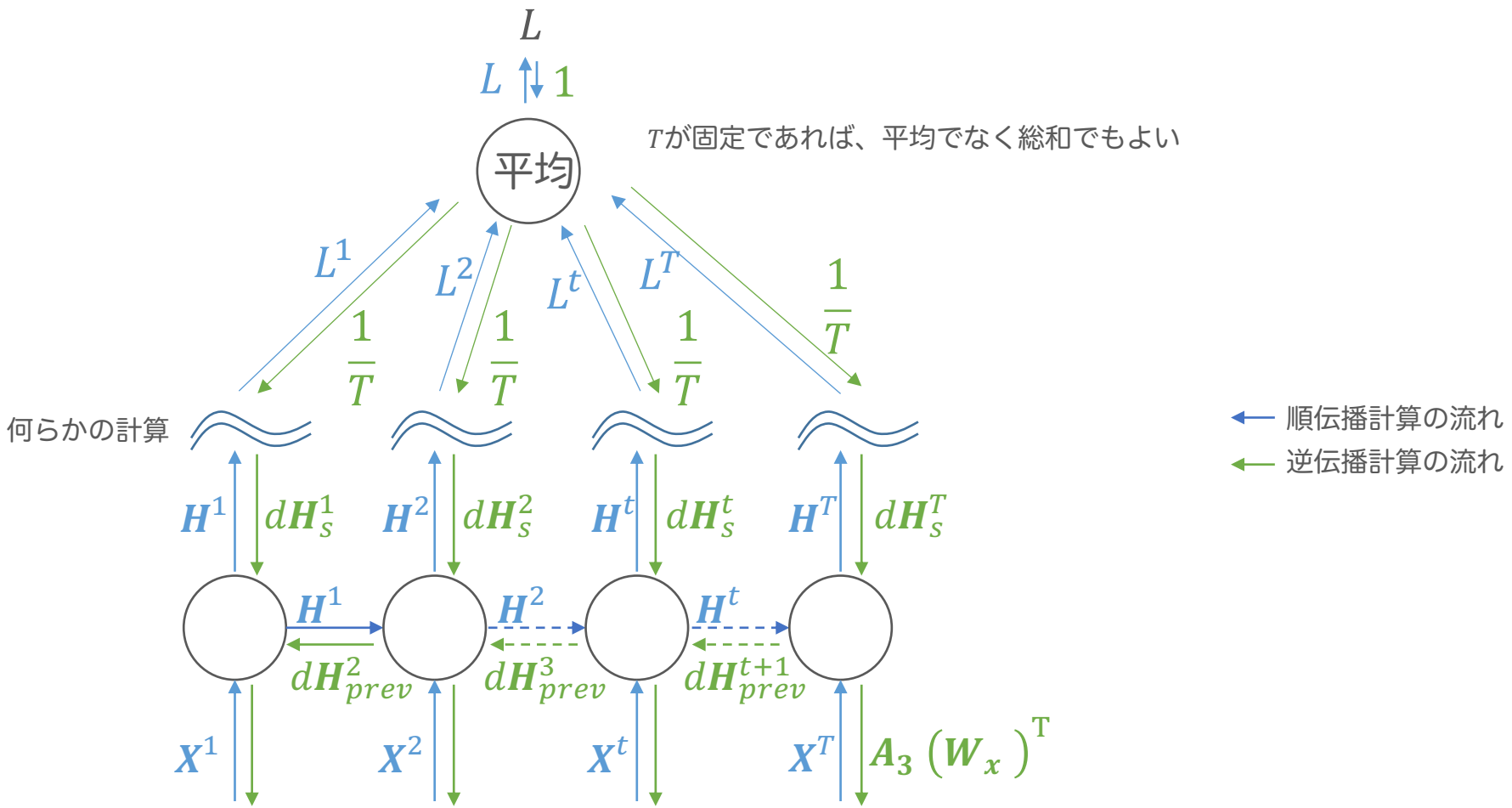
$$dH^t_{next} = dH^{t+1}_{prev} + dH^t_s$$
$$A_3 = df \odot dH^t_{next}$$
$$dH^t_{prev} = A_3 (W_h)^{\mathrm{T}}$$
df: 活性化関数fの微分



※行列のサイズは、バッチサイズがNでRNNレイヤのノード数が3の場合

TimeRNNレイヤの計算グラフ

x^t 右上の添字は時刻を表す

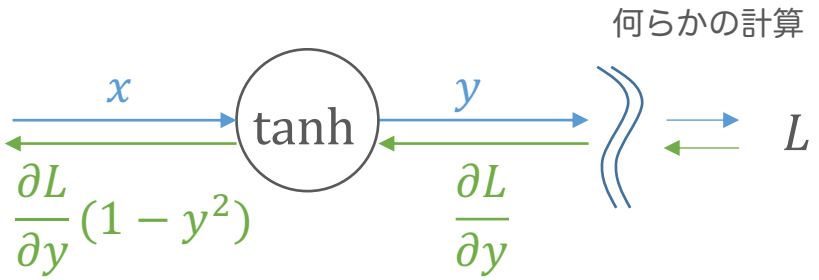


長期依存性の課題

- シンプルなRNNの逆伝播では、下式のように、同じ処理を繰り返していく。このように、同じ値の重みを繰り返し掛けていくと値が発散する(勾配爆発)か０になりやすい(勾配消失)。
  - $d\boldsymbol{H}_{prev}^{t} = df \odot d\boldsymbol{H}_{next}^{t}\left(\boldsymbol{W}_{h}\right)^{\mathrm{T}}$
- この勾配爆発または勾配消失は、入力または出力の系列が長くなるほど顕著になる。
  - これは、「シンプルなRNNは長期の記憶を保持することが難しい」とも言える。
- このような課題を「長期依存性の課題」という。
- ちなみに、全結合型NNの場合は、再帰的結合がなくレイヤー毎に重みが異なるので、RNNに比べ、このような問題は発生しにくい。
- 勾配爆発を抑える方法としては、勾配クリッピングという方法がある。DAY3スライドを参照。

tanhノード

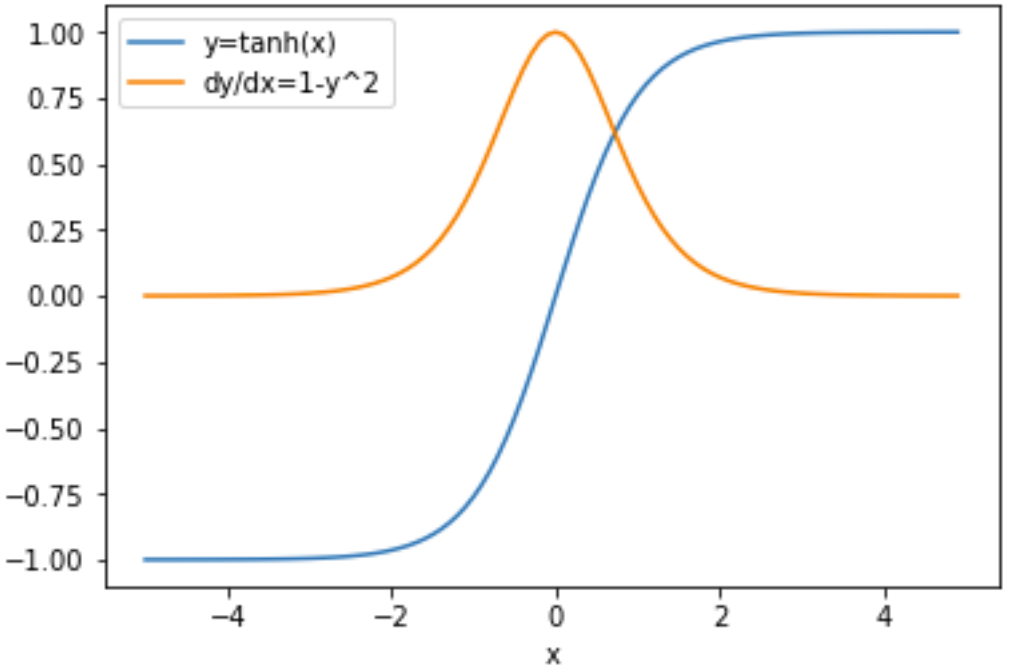
- RNNでは、活性化関数としてtanhがよく用いられる。
- tanhの微分値の最大値は1であり、シグモイド関数よりも勾配消失が起きにくい。
- tanhノードの順伝播および逆伝播を以下に示す。



$$y = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

分数関数の微分の公式を用いている

$$\frac{\partial y}{\partial x} = \frac{(e^x + e^{-x})(e^x + e^{-x}) - (e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2}$$
$$= 1 - \frac{(e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2}$$
$$= 1 - \left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right)^2$$
$$= 1 - y^2$$



[演習] tanh関数の確認

- 6_1_tanh.ipynb
  - tanh関数とその導関数を確認しましょう。

RNNにおけるReLUの利用

- RNNでは、活性化関数にReLUを用いるとsigmoidやtanhに比べ、勾配爆発が起きやすくなるので、 ReLUを用いる場合は勾配爆発を抑える工夫をした方が良い。
- 例えば、重みの初期値を工夫することで、ReLUの勾配爆発を抑える方法が提案されている。
  - Quoc V. Le, Navdeep Jaitly, Geoffrey E. Hinton , A Simple Way to Initialize Recurrent Networks of Rectified Linear Units,2015, https://arxiv.org/pdf/1504.00941.pdf
  - Sachin S. Talathi & Aniket Vartak, IMPROVING PERFORMANCE OF RECURRENT NEURAL NETWORK WITH RELU NONLINEARITY, ICLR 2016, https://openreview.net/pdf?id=wVqq536NJiG0qV7mtBNp

[演習] 勾配消失および勾配爆発に関するシミュレーション

- 6_2_gradient_simulation.ipynb
  - RNNは勾配消失や勾配爆発が起きやすい構造となっています。それら現象が起きる原理をシミュレーションによって確認します。
  - 活性化関数にReLUを用いるとsigmoidやtanhに比べ勾配爆発が起きやすくなることもここで確認します。

[演習] RNNレイヤの実装

- 6_3_colon.ipynb
  - 以降のRNNの実装において、 [:]を利用します。
  - ここでその動作を確認しておきましょう。
  - [:]を利用すると、同じメモリ位置に値を代入することができます。
- 6_4_RNN_layer_trainee.ipynb
  - RNNレイヤを実装しましょう。

[演習] TimeRNNレイヤの実装

- 6_5_TimeRNN_layer_trainee.ipynb
  - TimeRNNレイヤを実装しましょう。
  - TimeRNNレイヤの内部でRNNレイヤを用います。

[演習] 各レイヤの確認

- 6_6_Embedding.ipynb
  - 次のSimpleRNNLMクラスで用いるEmbeddingレイヤの中身を確認しておきましょう。
  - Embeddingレイヤは、単語IDを単語埋め込みベクトルに変換するためのレイヤです。単語埋め込みベクトルについては、DAY7の自然言語処理部分で解説します。
- 6_7_TimeEmbedding.ipynb
  - 次のSimpleRNNLMクラスで用いるTimeEmbeddingレイヤの中身を確認しておきましょう。
  - TimeEmbeddingレイヤは、Embeddingレイヤを時間方向に結合していくレイヤです。
- 6_8_TimeAffine.ipynb
  - 次のSimpleRNNLMクラスで用いるTimeAffineレイヤの中身を確認しておきましょう。
  - TimeAffineレイヤは、Affineレイヤを時間方向に結合していくレイヤです。

[演習] SimpleRNNLMクラスの実装

- 6_9_SimpleRNNLM_trainee.ipynb
  - SimpleRNNLMクラスを実装しましょう。
  - 実装したSimpleRNNLMクラスを用いて学習を行いましょう。
  - 埋め込み行列の初期値は、いくつかの考え方があるが、ここでは正規分布を100で割っている。
    - [参考] 初期値に関する検討論文: Tom Kocmi and Ondrej Bojar, An Exploration of Word Embedding Initialization in Deep-Learning Tasks, https://arxiv.org/pdf/1711.09160.pdf

ゲート付きRNN

- シンプルなRNNには長期依存性の課題がある。
- これを解決するための方法がいくつか提案されており、その中で最も成功しているのは、LSTM(Long Short-Term Memory)である。
- GRU(gated reccurent unit)は、 LSTMを簡略化したものである。
- LSTMやGRUなどは、ゲート付きRNNと呼ばれる。
- ゲートとは、データを通過させる程度を決めるノードのことである。流路の開き具合を調整しているように見えることから、こう呼ばれている。
- ゲートには、通常、シグモイド関数が用いられる。シグモイド関数の出力は0〜1であるため、開き具合が0~1の実数で調整されることになる。

Any Questions?

LSTM

LSTM

- LSTMは、シンプルなRNNのRNNレイヤがLSTMレイヤに置き換えられたものであり、モデル全体の考え方はシンプルなRNNと同じ。
- LSTMレイヤは、3つのゲートと記憶セルで構成される。
  - 出力ゲート(output gate)
    →取り出した記憶について、各要素を弱める度合いを決めるゲート
  - 忘却ゲート(forget gate)
    →記憶セルの各要素を弱める(忘れる)度合いを決めるゲート
  - 入力ゲート(input gate)
    →記憶セルに追加されるgの各要素を弱める度合いを決めるゲート
  - 記憶セル(memory cell)
    → 過去の情報を記憶する部分

LSTMレイヤの計算グラフ

$$\boldsymbol{F} = \sigma\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(f)} + \boldsymbol{H}^{t-1}\boldsymbol{W}_h^{(f)} + \boldsymbol{B}^{(f)}\right)$$

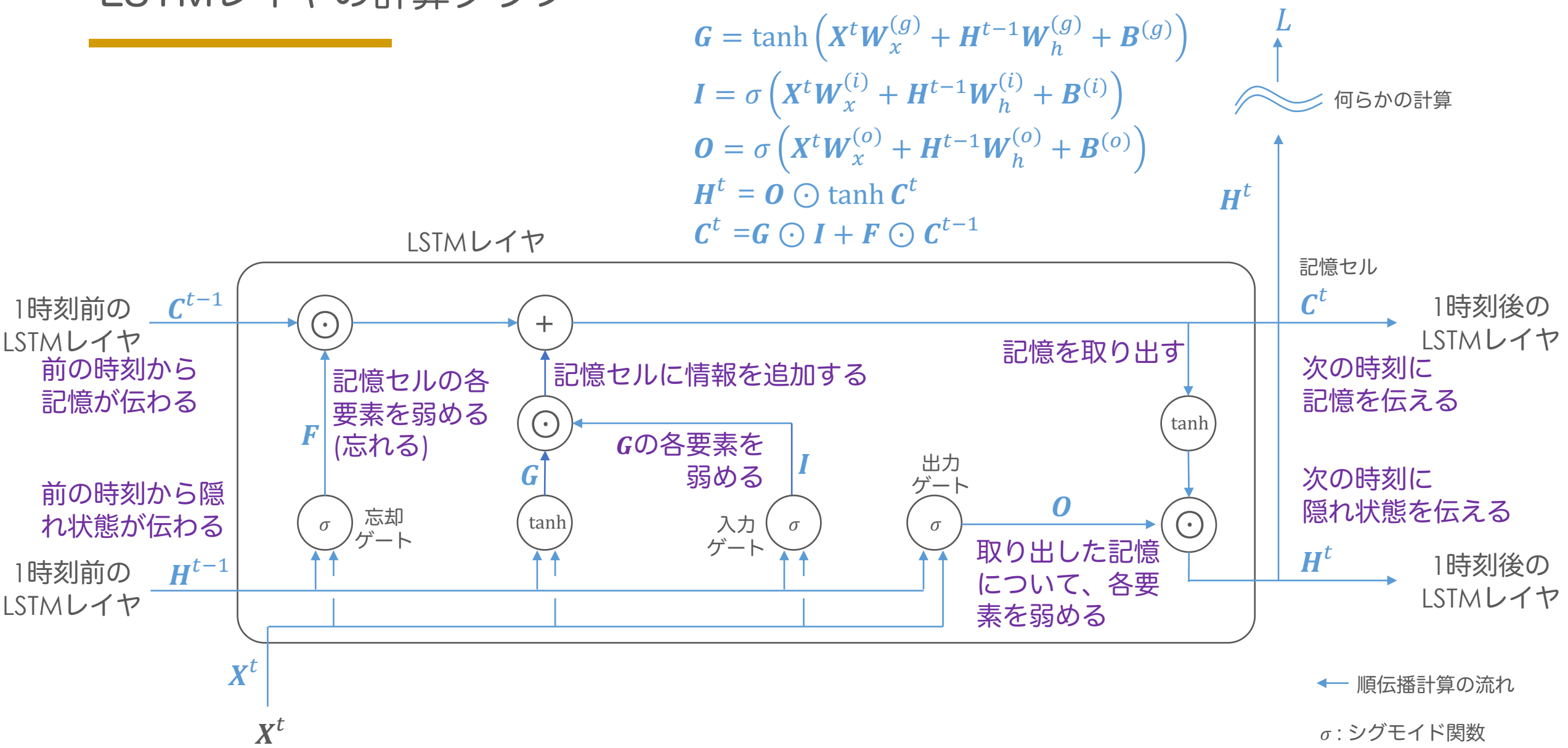
$$\boldsymbol{G} = \tanh\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(g)} + \boldsymbol{H}^{t-1}\boldsymbol{W}_h^{(g)} + \boldsymbol{B}^{(g)}\right)$$

$$\boldsymbol{I} = \sigma\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(i)} + \boldsymbol{H}^{t-1}\boldsymbol{W}_h^{(i)} + \boldsymbol{B}^{(i)}\right)$$

$$\boldsymbol{O} = \sigma\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(o)} + \boldsymbol{H}^{t-1}\boldsymbol{W}_h^{(o)} + \boldsymbol{B}^{(o)}\right)$$

$$\boldsymbol{H}^t = \boldsymbol{O} \odot \tanh \boldsymbol{C}^t$$

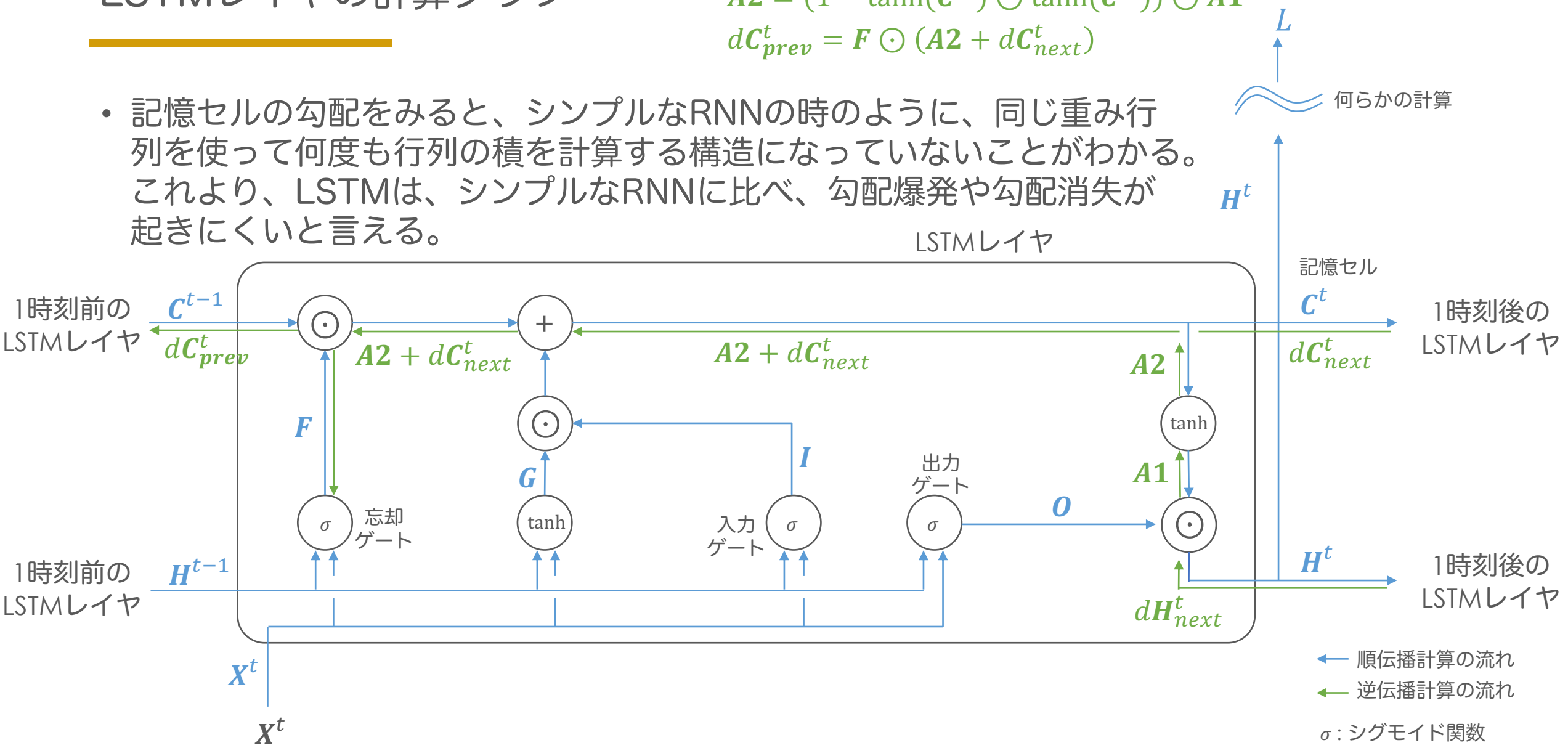
$$\boldsymbol{C}^t = \boldsymbol{G} \odot \boldsymbol{I} + \boldsymbol{F} \odot \boldsymbol{C}^{t-1}$$



LSTMレイヤの計算グラフ

$$A1 = O \odot dH_{next}^{t}$$
$$A2 = (1 - \tanh(C^{t}) \odot \tanh(C^{t})) \odot A1$$
$$dC_{prev}^{t} = F \odot (A2 + dC_{next}^{t})$$

- 記憶セルの勾配をみると、シンプルなRNNの時のように、同じ重み行列を使って何度も行列の積を計算する構造になっていないことがわかる。これより、LSTMは、シンプルなRNNに比べ、勾配爆発や勾配消失が起きにくいと言える。



LSTMレイヤを効率的に実装する方法

E資格対策

- 順伝播計算の式を見ると、X^tまたはH^{t-1}が共通なので、 W_x^*またはW_h^*を1つにまとめると、行列の積が1回で済む

$$
\begin{aligned}
F &: \left(X^t W_x^{(f)} + H^{t-1} W_h^{(f)} + B^{(f)}\right) \\
G &: \left(X^t W_x^{(g)} + H^{t-1} W_h^{(g)} + B^{(g)}\right) \\
I &: \left(X^t W_x^{(i)} + H^{t-1} W_h^{(i)} + B^{(i)}\right) \\
O &: \left(X^t W_x^{(o)} + H^{t-1} W_h^{(o)} + B^{(o)}\right)
\end{aligned}
$$

$X^t W_x$ ＋ $H^{t-1} W_h$ ＋ B

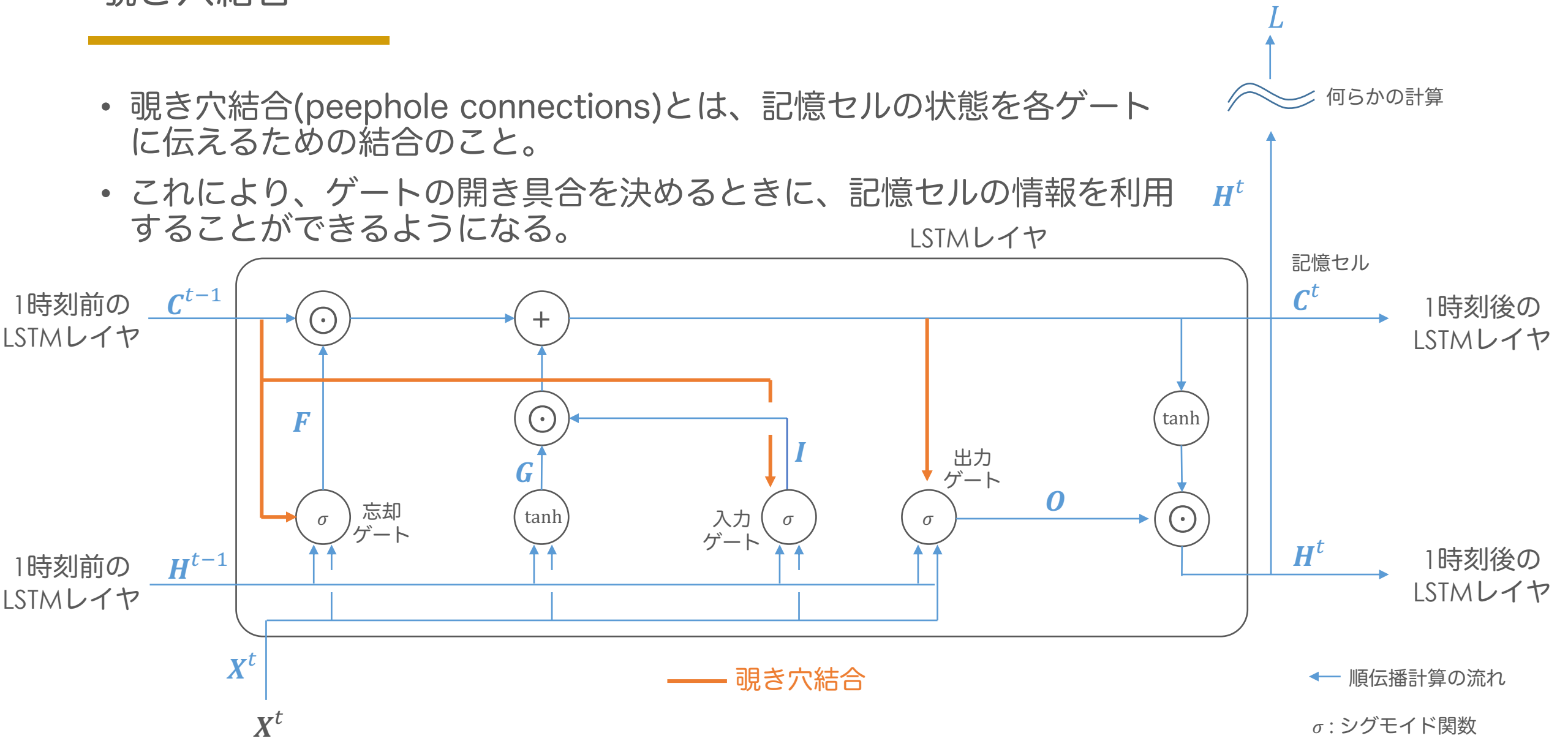
$W_x^{(f)}$	$W_x^{(g)}$	$W_x^{(i)}$	$W_x^{(o)}$

$W_h^{(f)}$	$W_h^{(g)}$	$W_h^{(i)}$	$W_h^{(o)}$

$b^{(f)}$	$b^{(g)}$	$b^{(i)}$	$b^{(o)}$

覗き穴結合

- 覗き穴結合(peephole connections)とは、記憶セルの状態を各ゲートに伝えるための結合のこと。
- これにより、ゲートの開き具合を決めるときに、記憶セルの情報を利用することができるようになる。



覗き穴結合

- 覗き穴結合利用時の順伝播計算

$$\boldsymbol{F} = \sigma\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(f)} + \boldsymbol{H}^{t-1}\boldsymbol{W}_h^{(f)} + \boldsymbol{C}^{t-1}\boldsymbol{W}_c^{(f)} + \boldsymbol{B}^{(f)}\right)$$

$$\boldsymbol{G} = \tanh\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(g)} + \boldsymbol{H}^{t-1}\boldsymbol{W}_h^{(g)} + \boldsymbol{B}^{(g)}\right)$$

$$\boldsymbol{I} = \sigma\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(i)} + \boldsymbol{H}^{t-1}\boldsymbol{W}_h^{(i)} + \boldsymbol{C}^{t-1}\boldsymbol{W}_c^{(i)} + \boldsymbol{B}^{(i)}\right)$$

$$\boldsymbol{O} = \sigma\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(o)} + \boldsymbol{H}^{t-1}\boldsymbol{W}_h^{(o)} + \boldsymbol{C}^t\ \boldsymbol{W}_c^{(o)} + \boldsymbol{B}^{(o)}\right)$$

$$\boldsymbol{H}^t = \boldsymbol{O} \odot \tanh \boldsymbol{C}^t$$

$$\boldsymbol{C}^t = \boldsymbol{G} \odot \boldsymbol{I} + \boldsymbol{F} \odot \boldsymbol{C}^{t-1}$$

覗き穴結合に関する注意点

E資格対策

- 覗き穴結合は、原著論文を見ると$\boldsymbol{c}$と$\boldsymbol{W}$の行列積で計算するモデルとして提案されている。
  - F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, “Learning precise timing with LSTM recurrent networks,” Journal of Machine Learning Research, vol. 3, pp. 115–143, Mar. 2003.
  - http://www.jmlr.org/papers/volume3/gers02a/gers02a.pdf
- その後、派生したモデルとして、$\boldsymbol{W}$を対角行列に制限して計算する例がいくつか見られる。この方がパラメータ数が少なくなる。
  - Has ¸im Sak, Andrew Senior, Franc ¸oise Beaufays. Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling.
  - https://www.isca-speech.org/archive/archive_papers/interspeech_2014/i14_0338.pdf
- また、$\boldsymbol{c}$と$\boldsymbol{W}$をアダマール積で計算する例も見られる。アダマール積にすると、計算量が減る。
  - Xingjian Shi, Zhourong Chen, Hao Wang Dit-Yan Yeung, Wai-kin Wong, Wang-chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting.
  - https://arxiv.org/pdf/1506.04214.pdf
- もし、E資格試験において、覗き穴結合の計算式に関する設問が出た場合、行列積の選択肢を選べばよいが、アダマール積の選択肢しかない場合は、それも正解の候補になる可能性がある。

[演習] LSTMレイヤの実装

- 6_10_LSTM_layer_trainee.ipynb
  - LSTMレイヤを実装しましょう。

Any Questions?

GRU

GRU

- GRUは、LSTMよりもシンプルなゲート付きRNN。
- LSTMよりもパラメータが少ない。
- GRUレイヤは、以下の2つのゲートで構成される。
  - リセットゲート(reset gate)
    → 過去の隠れ状態を弱める程度を決めるゲート
  - 更新ゲート(update gate)
    → 過去の隠れ状態と仮の隠れ状態の混合割合を決めるゲート
- 原著論文
  - Kyunghyun Choet al., Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, https://arxiv.org/pdf/1406.1078.pdf

GRUレイヤの計算グラフ

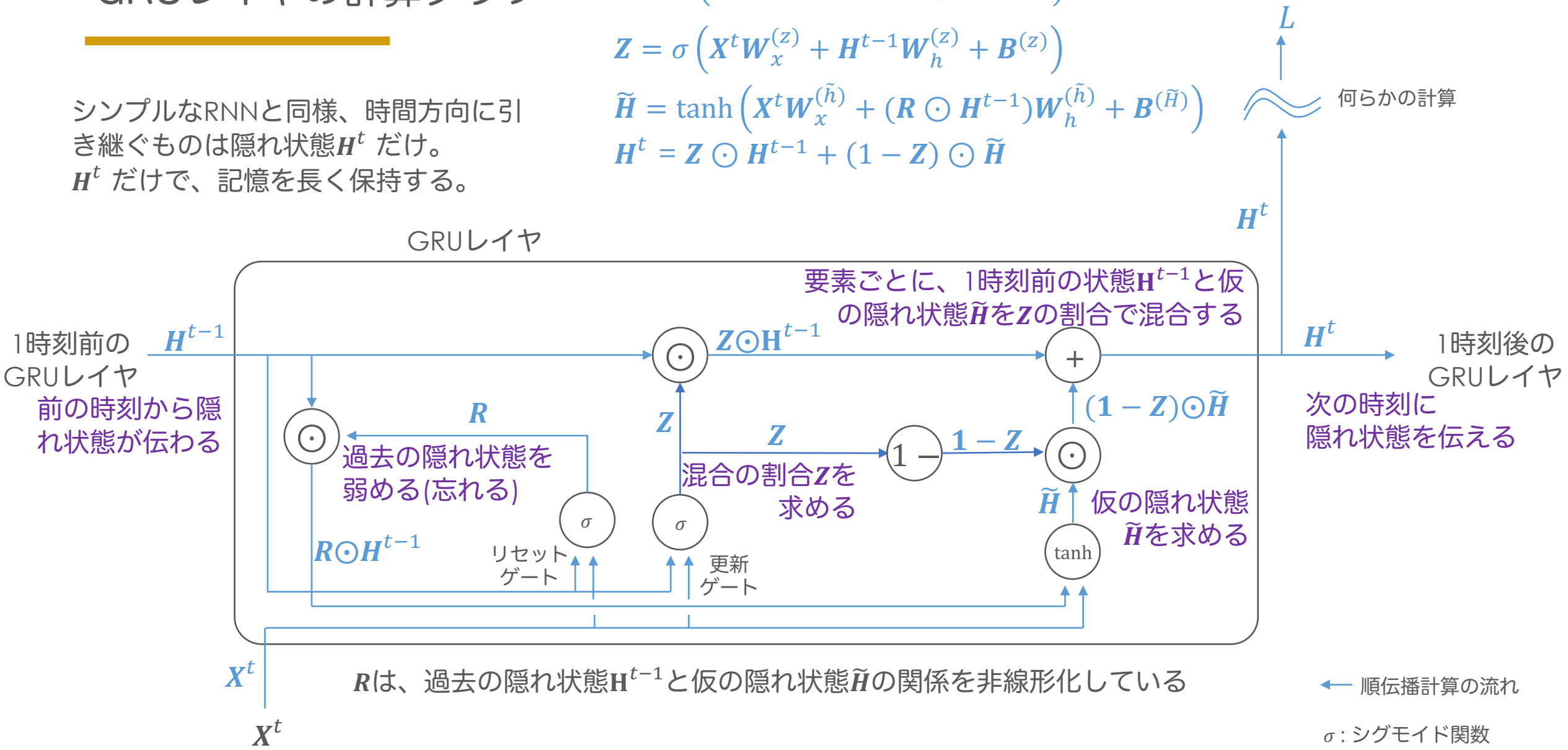
$$\boldsymbol{R} = \sigma\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(r)} + \boldsymbol{H}^{t-1}\boldsymbol{W}_h^{(r)} + \boldsymbol{B}^{(r)}\right)$$

$$\boldsymbol{Z} = \sigma\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(z)} + \boldsymbol{H}^{t-1}\boldsymbol{W}_h^{(z)} + \boldsymbol{B}^{(z)}\right)$$

$$\tilde{\boldsymbol{H}} = \tanh\left(\boldsymbol{X}^t\boldsymbol{W}_x^{(\tilde{h})} + (\boldsymbol{R}\odot\boldsymbol{H}^{t-1})\boldsymbol{W}_h^{(\tilde{h})} + \boldsymbol{B}^{(\tilde{H})}\right)$$

$$\boldsymbol{H}^t = \boldsymbol{Z}\odot\boldsymbol{H}^{t-1} + (1-\boldsymbol{Z})\odot\tilde{\boldsymbol{H}}$$

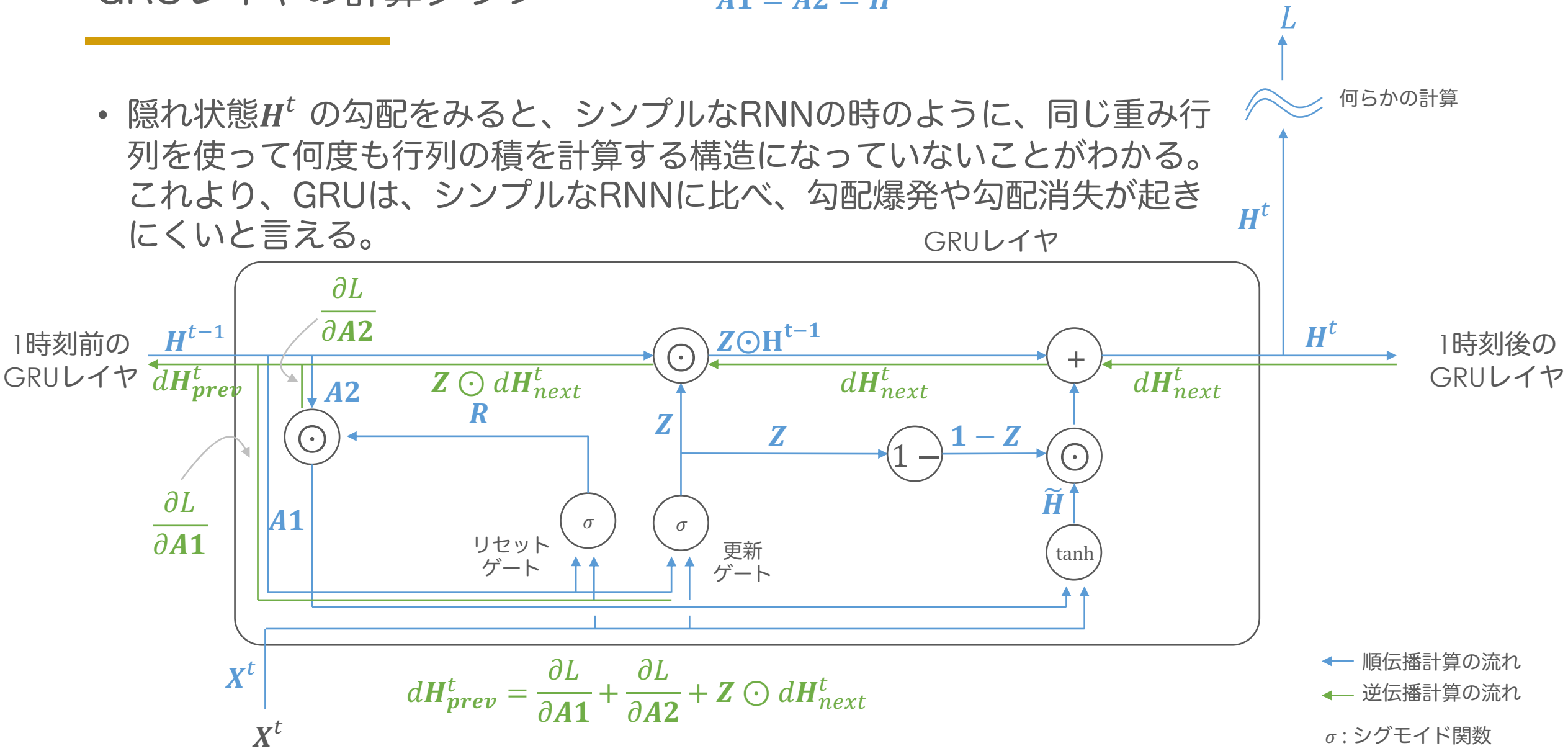
シンプルなRNNと同様、時間方向に引き継ぐものは隠れ状態$\boldsymbol{H}^t$ だけ。
$\boldsymbol{H}^t$ だけで、記憶を長く保持する。



GRUレイヤの計算グラフ

$$A1 = A2 = H^{t-1}$$

- 隠れ状態H^t の勾配をみると、シンプルなRNNの時のように、同じ重み行列を使って何度も行列の積を計算する構造になっていないことがわかる。これより、GRUは、シンプルなRNNに比べ、勾配爆発や勾配消失が起きにくいと言える。



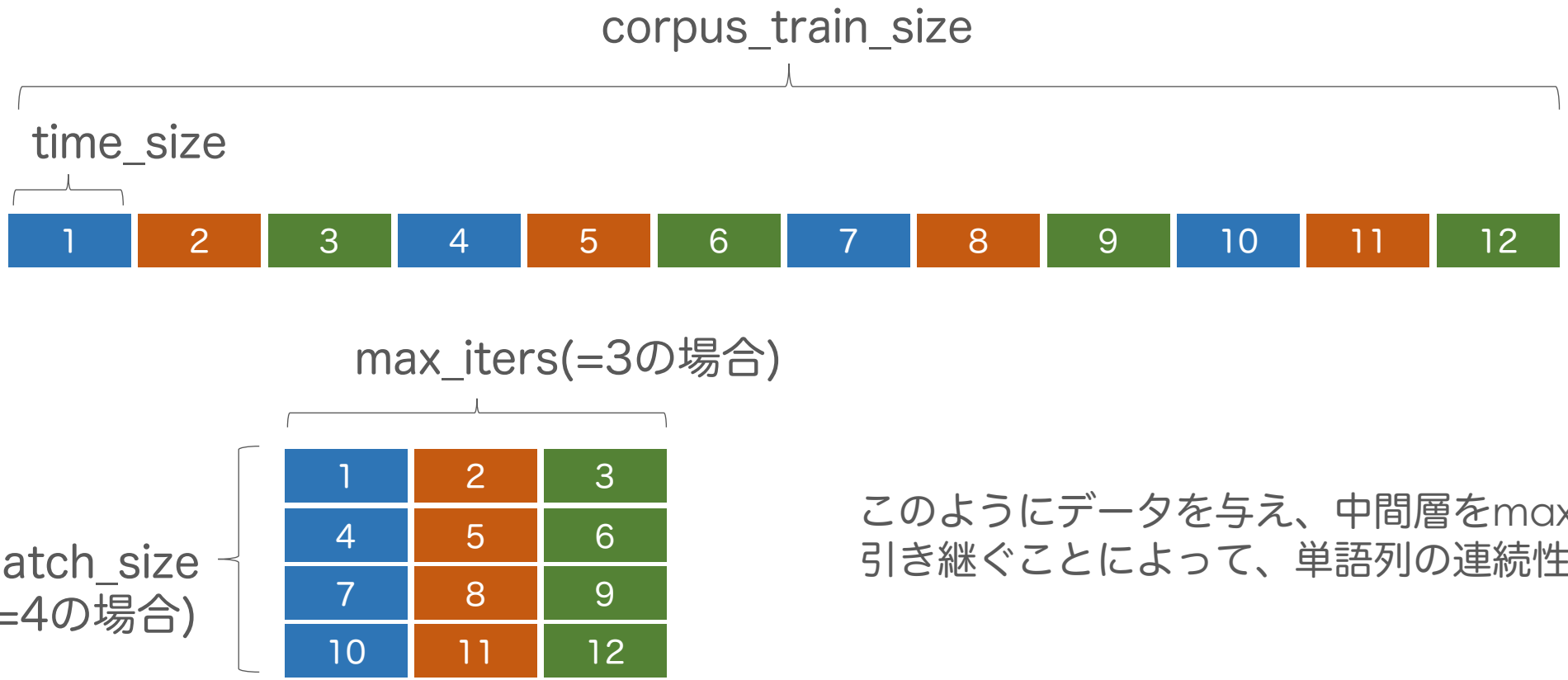
[演習] GRUレイヤの実装

- 6_11_GRU_layer_trainee.ipynb
  - GRUレイヤを実装しましょう。

[演習] RNNの学習

- 6_12_trainer.ipynb
  - ここまでに作成したクラスを用いて、単語を予測する問題を解いてみましょう。
  - SimpleRnnlm,、LSTMlm、 GRUlmは、modelで切り替えられます。
  - 学習すると予測精度の低いモデルができあがりますが、予測精度を上げるには、学習データを増やしたり、モデルを工夫する必要があります。
  - 余裕のある方は、予測精度を向上させるための工夫を検討してみましょう。
  - このNotebookおける学習用データの与え方を次頁に示します。

[演習] RNNの学習



このようにデータを与え、中間層をmax_itersの方向に
引き継ぐことによって、単語列の連続性を考慮している。

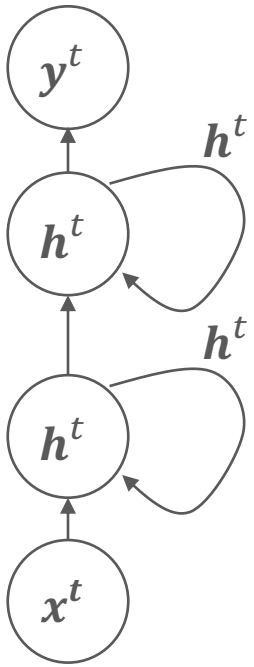
Any Questions?

RNNの発展モデル

RNNの層を深くすること

- RNNにおいて、層を深くすることは有益か？
- 実験結果によると有益とのこと(Graves *et al.*, 2013; Pascanu *et al.*, 2014a)
- 参考：(深層学習, GoodFellow, KADOKAWA, 10.5節)

階層的にしたRNNの例

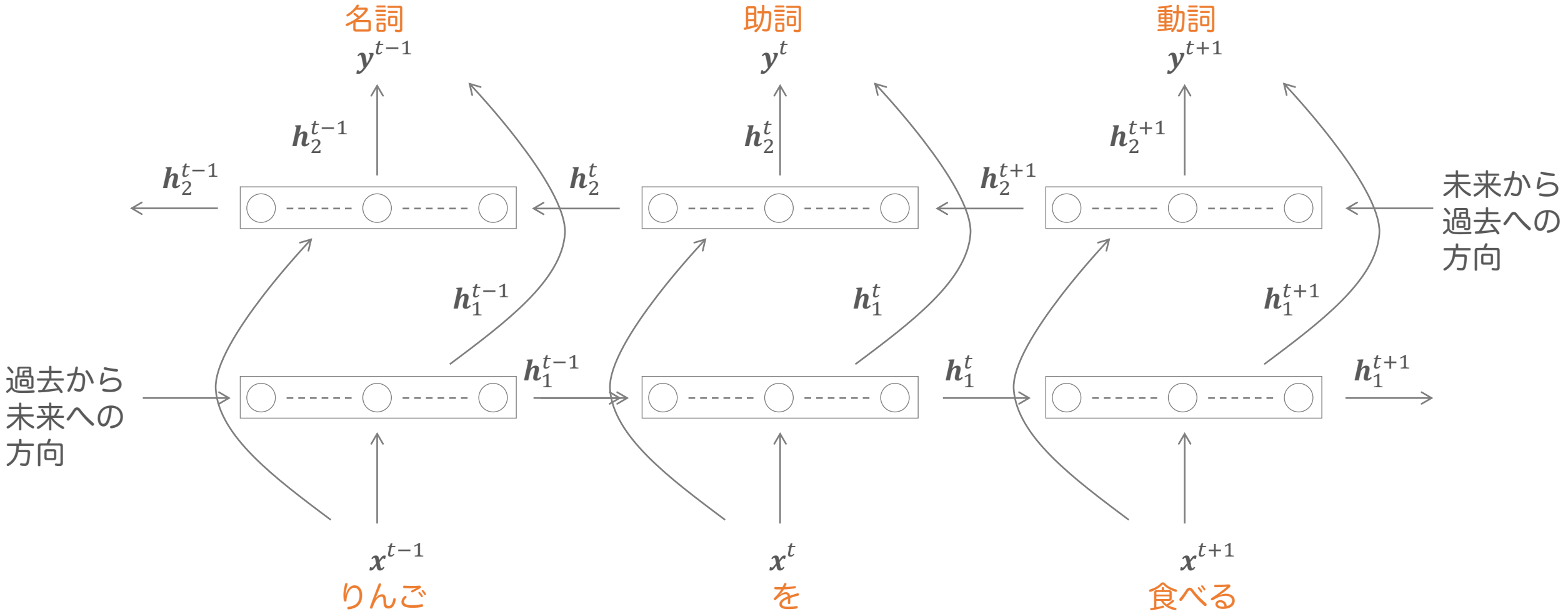


双方向RNN

- 過去から未来だけでなく、未来から過去への方向も考慮したRNN。
- 入力系列全体の情報を各時刻で考慮することができる。
- 未来の予測ではなく、すでに手元にあるデータに対して、何らかのクラス分類を行う場合は、過去から未来だけでなく、未来から過去への方向も考慮した方が良い精度が期待できるはずというのがコンセプト。

双方向RNN

双方向RNNの概念図



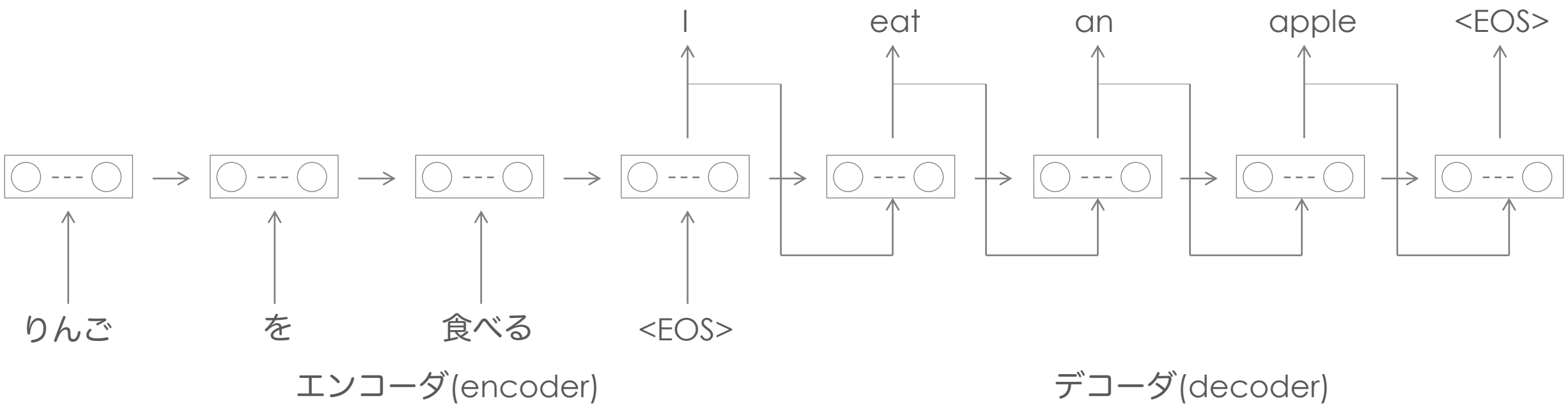
系列変換モデル

- 単語列などの系列(sequence)を受け取り、別の系列へ変換するモデルのことを系列変換モデル(sequence-to-sequence model, seq2seqモデル)という。
- 主に自然言語処理で用いられる。
- RNNエンコーダ・デコーダとも呼ばれる。
- 適応されるタスクの例
  - 機械翻訳 (翻訳元の言語から翻訳先の言語への変換)
  - 対話 (相手の発言から自分の発言への変換)
  - 質問応答 (質問文から返答文への変換)
  - 文書要約 (元文書から要約文への変換)

系列変換モデル

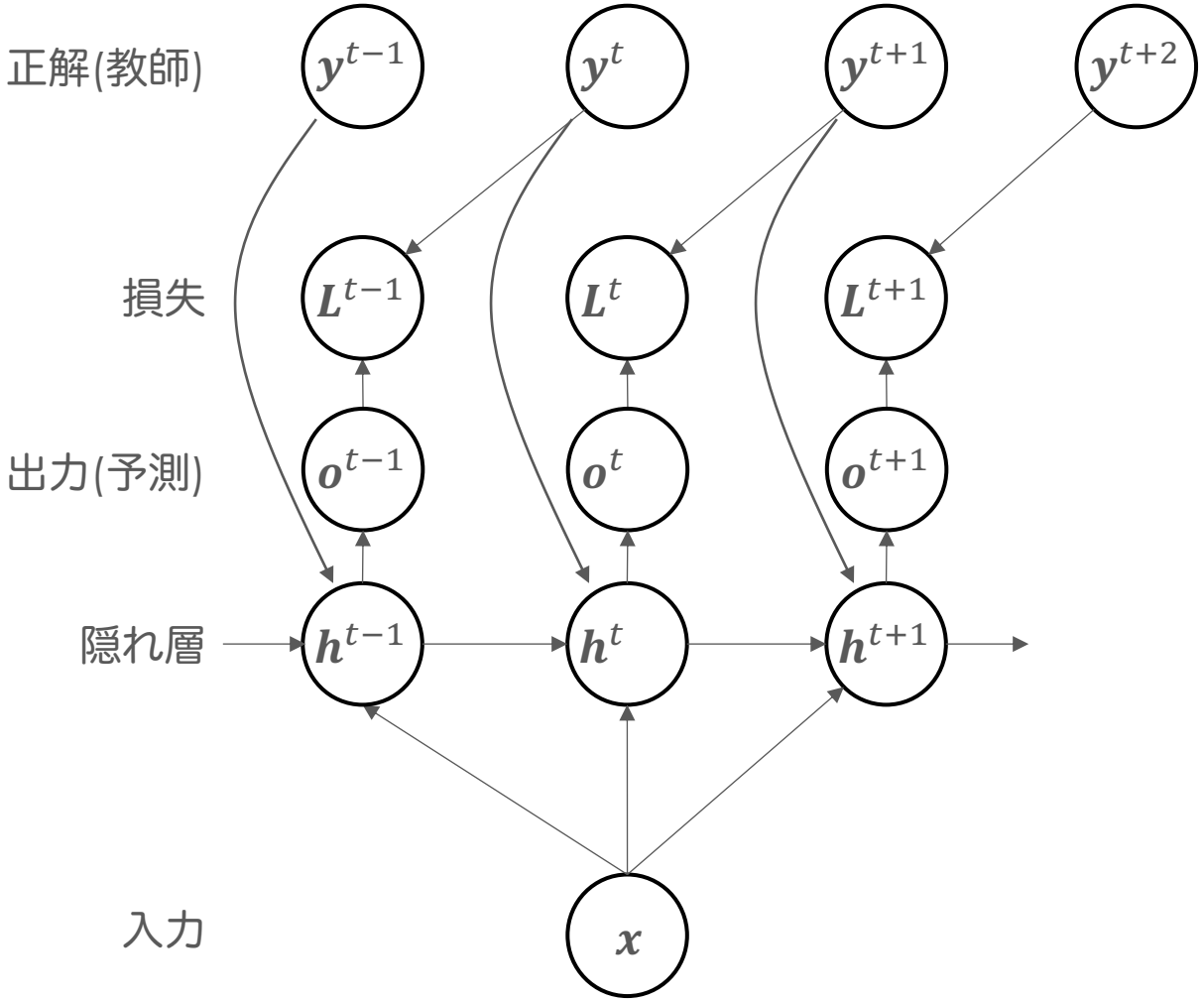
- 入力系列の長さと出力系列の長さは、一致していなくてもよい。
- 系列の長さは、データによって異なっていてもよい。

系列変換モデルの概念図



入力が系列でないRNN

- 出力は系列であるが、入力が系列でないRNNの例を右に示す。
  - 入力は固定長のベクトルで常に同じ。
  - 出力は系列となっており、誤差の計算以外に隠れ層への入力にも用いられる。
  - 画像に対するキャプション生成などのタスクに向いている。
- 参考:Ian Goodfellow, 深層学習, KADOKAWA, 10.2.4節



CNNとRNNの組み合わせ：画像キャプション生成タスク

- 画像の内容を自然言語で記述するタスク。
- 画像キャプション生成モデルの例として、Neural Image Captionがある。
  - Oriol Vinyals, Alexander Toshev, Samy Bengio, Dumitru Erhan. Show and Tell: A Neural Image Caption Generator. https://arxiv.org/pdf/1411.4555.pdf

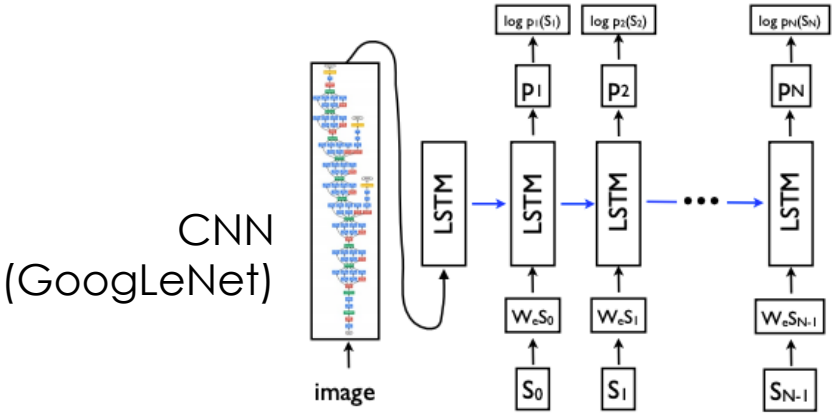


Figure 3. LSTM model combined with a CNN image embedder (as defined in [12]) and word embeddings. The unrolled connections between the LSTM memories are in blue and they correspond to the recurrent connections in Figure 2. All LSTMs share the same parameters.

Figure 5. A selection of evaluation results, grouped by human rating.

CNNとRNNの組み合わせ：画像質問応答タスク

- 画像に関する質問に答えるタスク。
- CNNとRNNを組み合わせたモデルが利用される。
- VQA用データセットが提供されているサイト。
  - https://visualqa.org/

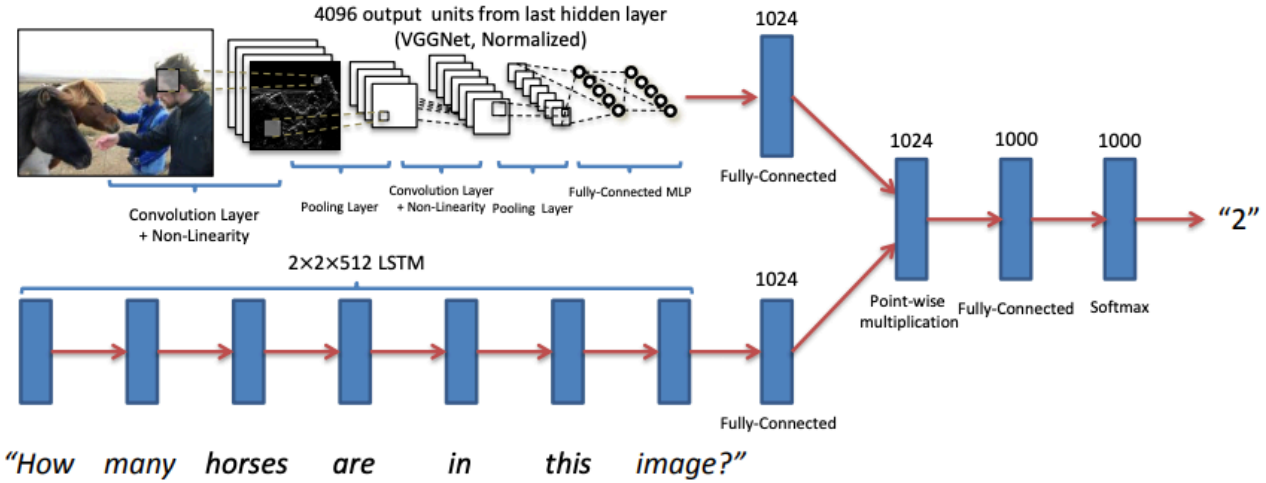


Fig. 8: Our best performing model (deeper LSTM Q + norm I). This model uses a two layer LSTM to encode the questions and the last hidden layer of VGGNet [48] to encode the images. The image features are then ℓ_2 normalized. Both the question and image features are transformed to a common space and fused via element-wise multiplication, which is then passed through a fully connected layer followed by a softmax layer to obtain a distribution over answers.

Fig. 1: Examples of free-form, open-ended questions collected for images via Amazon Mechanical Turk. Note that commonsense knowledge is needed along with a visual understanding of the scene to answer many questions.

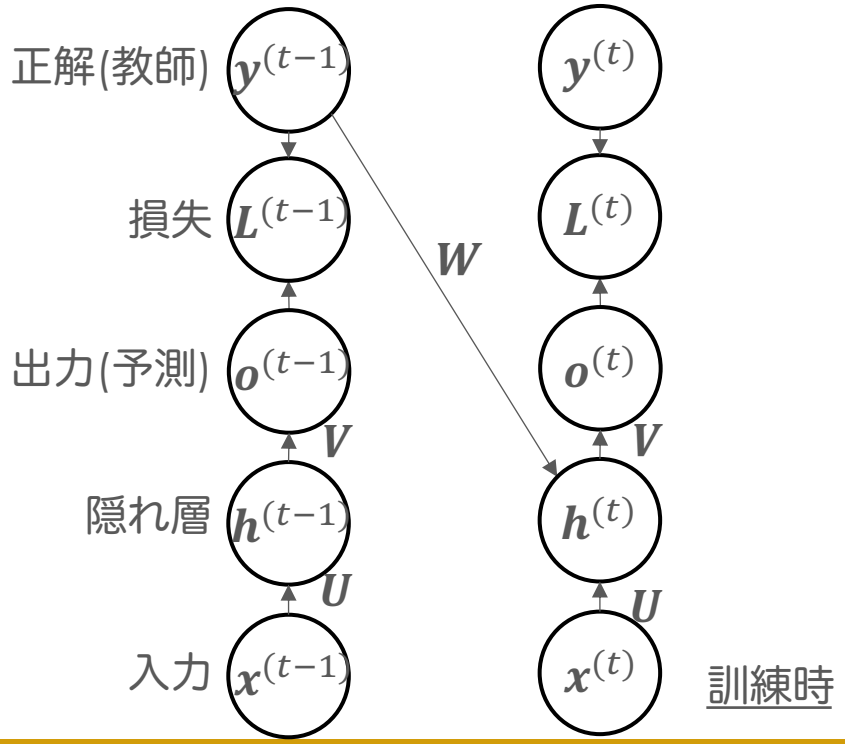
Aishwarya Agrawal∗ , Jiasen Lu∗ , Stanislaw Antol∗ , Margaret Mitchell, C. Lawrence Zitnick, Dhruv Batra, Devi Parikh.
VQA: Visual Question Answering.
https://arxiv.org/pdf/1505.00468.pdf

Any Questions?

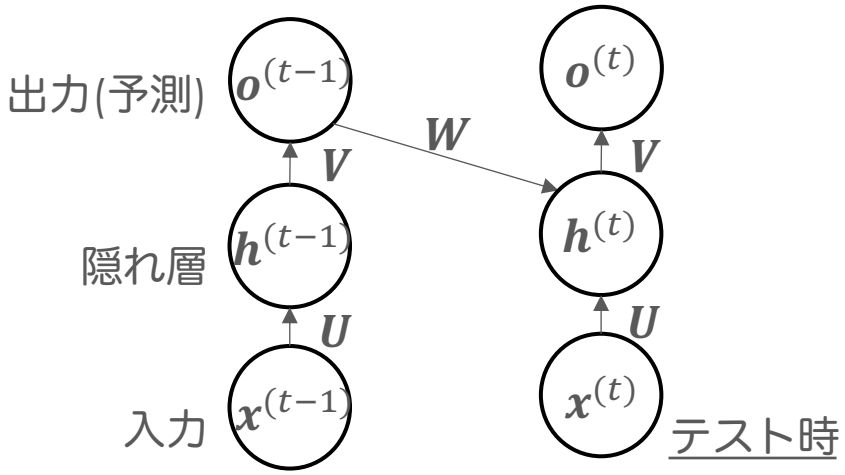
その他の話題

教師強制

- ある時刻の出力が次の時刻のノードに接続されるニューラルネットワークでは、教師強制(teacher forcing)という学習方法が使える。参考：深層学習,Ian Goodfellow,10.2.1節
- 教師強制の仕組み
  - 訓練時においては、ある時刻の教師データを強制的に次の時刻のノードに入力させる。
  - テスト時においては、計算によって求められた出力を次の時刻のノードに入れる。

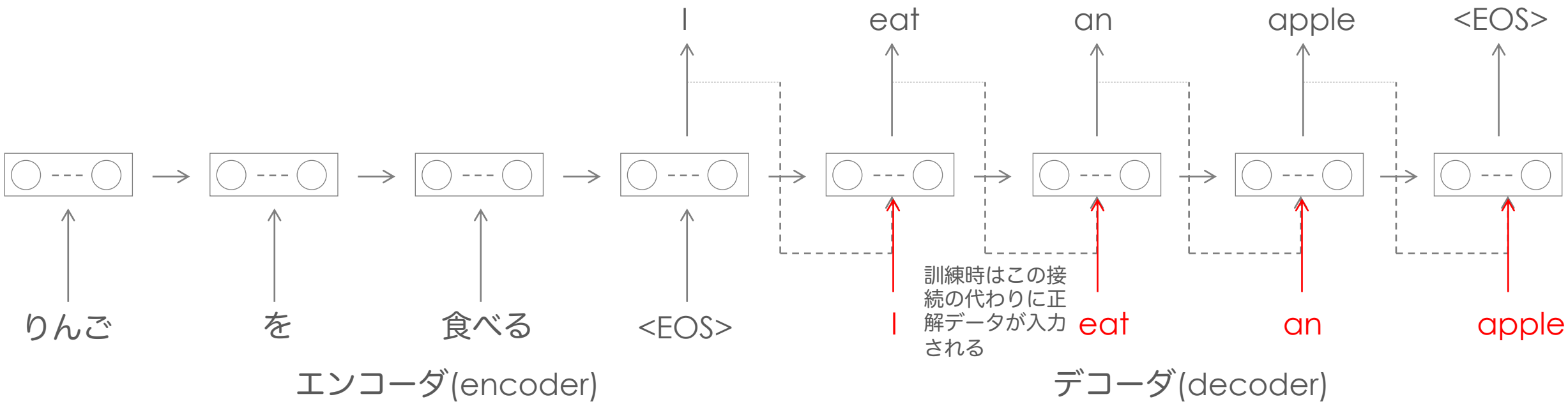


U, V, W: 重み行列



教師強制の適応例

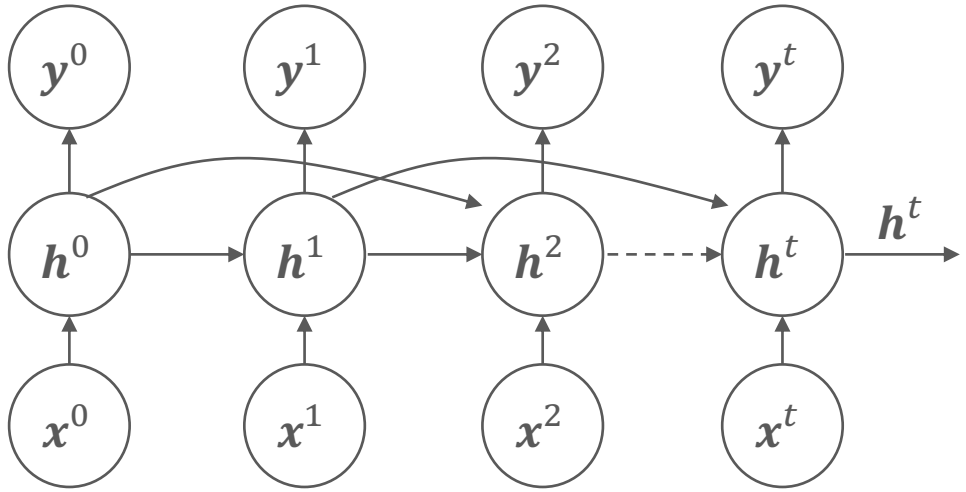
- 系列変換モデル(sequence-to-sequence model)を教師強制で学習させる場合の例を以下に示す。
- 教師強制で学習させる場合、訓練時のみ、decoderの入力に正解データを入れる。
- これにより、計算が安定し、収束までの時間が早くなることが期待できる。
- ただし、テスト時におけるdecoderの入力は、予測された値になるため、訓練時と分布が異なってしまうことに注意。



時間方向のスキップ接続

- 長期依存性の課題への対策の一つとして、過去の隠れ層から現在の隠れ層への接続を追加する方法がある。
- このような接続をスキップ接続という。
- スキップ接続により長期の依存性を得ることがある。
- 参考：深層学習, Ian Goodfellow, KADOKAWA,10.9.1節

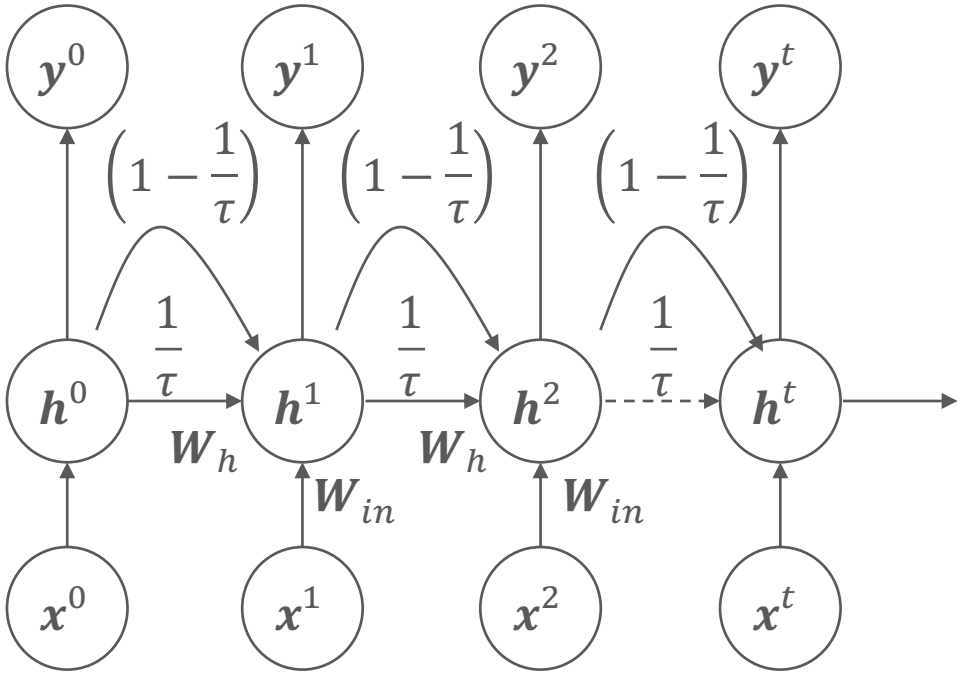
離れた過去と接続する例



Leakyユニット

E資格対策

- 隠れ層に線型結合を導入すると、移動平均のような効果が得られる。
- このような線形結合をもった隠れユニットは、Leakyユニットと呼ばれる。
- 参考:深層学習, Ian Goodfellow, KADOKAWA,10.9.2節



$$\boldsymbol{h}_t = \left(1-\frac{1}{\tau}\right)\boldsymbol{h}_{t-1} + \frac{1}{\tau} f(\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_{in}\boldsymbol{x}_t)$$

f:活性化関数
W_h:隠れ層の重み
W_{in}:入力の重み
τ:係数

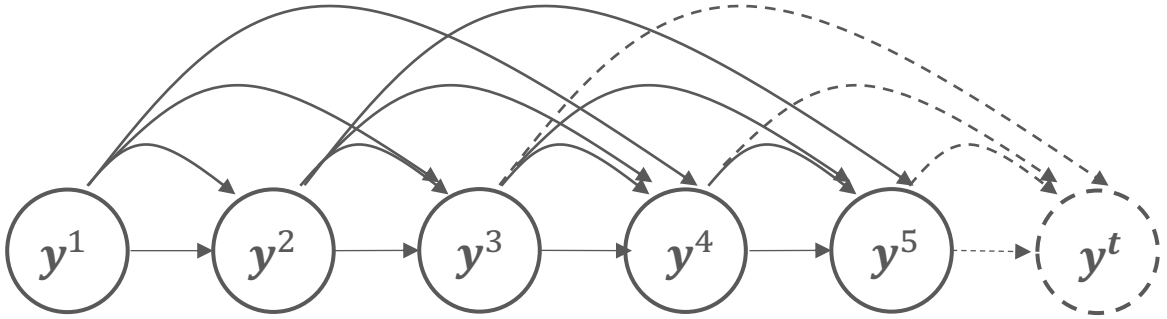
<移動平均>
$\mu^{(t)} \leftarrow \alpha\mu^{(t-1)} + (1-\alpha)v^{(t)}$
この式では、 $v^{(t)}$の移動平均が$\mu^{(t)}$。線形自己接続を持つ隠れユニットは、このような移動平均と似た挙動を示す。

有向グラフィカルモデルとしての再帰型ニューラルネットワーク

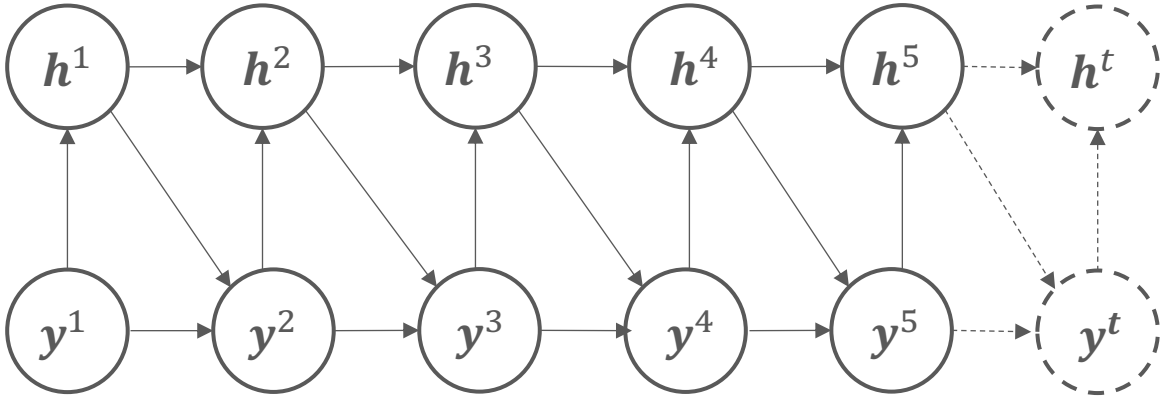
E資格対策

- RNNを時間方向の依存関係をもった有向グラフィカルモデルと捉えてみる。
- グラフィカルモデルとは、確率変数間の依存関係を表すモデルのこと。結合が双方向でない場合を有向グラフィカルモデルという。
- 時間方向の依存関係を通常のグラフィカルモデルで表現する場合、全てのノードを結合させることになり、パラメータが膨大になる。
- これに対し、RNNで時間方向の依存関係を表現すると、パラメータが効率化される。
- 参考：深層学習, Ian Goodfellow, KADOKAWA,10.2.3節

グラフィカルモデルで時間方向の依存関係を表現する場合



RNNで時間方向の依存関係を表現する場合



Any Questions?

[グループワーク] RNNを活用できるタスク

- 再帰型ニューラルネットワークは、翻訳、対話、質問応答など様々なタスクで利用することができます。
- RNNを活用できそうなタスクをできるだけ沢山挙げてみましょう。
- それらタスクに適したRNNのネットワークを考えてみましょう。
- それらネットワークを学習するためにはどのようなデータセットを用意すればいいかを考えてみましょう。
- 2~3名のグループに分かれて、上記課題に取り組みましょう。(20分)
- 最後に、グループごとに発表していただきます。 (15分)

[グループワーク] LSTMレイヤまたはGRUレイヤの計算グラフ作成

- LSTMレイヤまたはGRUレイヤのどちらかを選択し、その計算グラフを作成しましょう。
- 順伝播だけでなく、逆伝播の流れも記入してください。
- LSTMレイヤの場合は、$\boldsymbol{W}_x^{(f)}, \boldsymbol{W}_x^{(g)}, \boldsymbol{W}_x^{(i)}, \boldsymbol{W}_x^{(o)}, \boldsymbol{W}_h^{(f)}, \boldsymbol{W}_h^{(g)}, \boldsymbol{W}_h^{(i)}, \boldsymbol{W}_h^{(o)}$も計算グラフに載せて表現して下さい。
- GRUレイヤの場合は、$\boldsymbol{W}_x^{(r)}, \boldsymbol{W}_x^{(z)}, \boldsymbol{W}_x^{(\tilde{h})}, \boldsymbol{W}_h^{(r)}, \boldsymbol{W}_h^{(z)} \boldsymbol{W}_h^{(\tilde{h})}$も計算グラフに載せて下さい。
- 2~3名のグループに分かれて、上記課題に取り組みましょう。(30分)
- 最後に、グループごとに発表していただきます。 (15分)

予習の段階で一度計算グラフを描いてみてください

講座の時間が余ったら

- 今回の復習をします。
- 次回の予習をします。