

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

1. 画像データ
2. 時系列データ
3. テキストデータ
4. データの権利

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

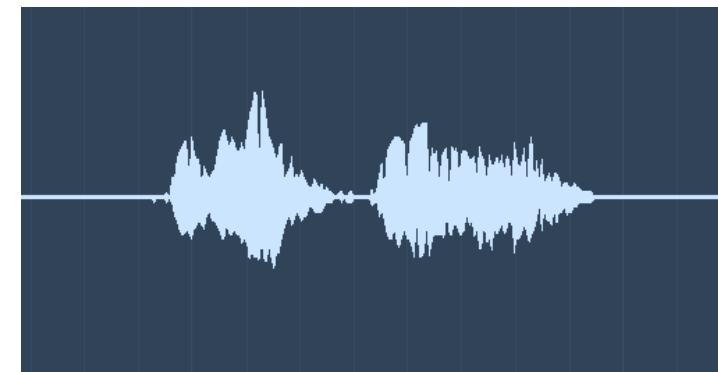
- <https://github.com/openimages/dataset>
- Google社が提供している約900万点の画像URLとそれについてのラベルや物体領域などのデータ。
- 画像のライセンスはCCBY2.0。

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

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(リンゴ, 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/>
- そのデータが著作物かどうかは、その国の著作権法を調べる必要がある。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いよい。
 - 参考：著作権法の例外規定
 - 「情報解析のための複製等（第47条の7）コンピュータ等を用いて情報解析（※）を行うことを目的とする場合には、必要と認められる限度において記録媒体に著作物を複製・翻案することができる。ただし、情報解析用に広く提供されているデータベースの著作物については、この制限規定は適用されない。
※情報解析とは、大量の情報から言語、音、映像等を抽出し、比較、分類等の統計的な解析を行うことをいう。」
http://www.bunka.go.jp/seisaku/chosakuken/seidokaisetsu/gaiyo/chosakubutsu_jiyu.html
- 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

1. 再帰型ニューラルネットワーク概要
2. シンプルなRNN
3. LSTM
4. GRU
5. RNNの発展モデル
6. 外部メモリを持つニューラルネットワーク
7. その他の話題

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。

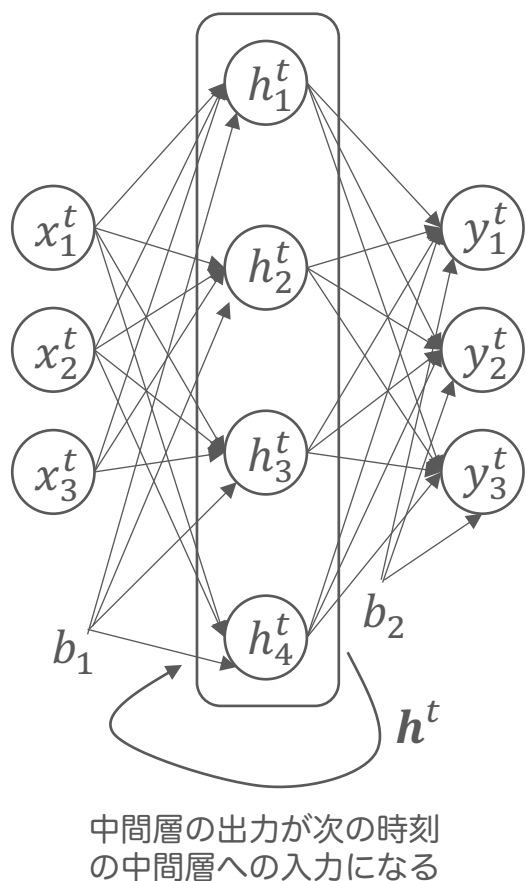
RNNの計算グラフ表現

x^t 右上の添字は時刻を表す
(大文字の T は最終時刻)

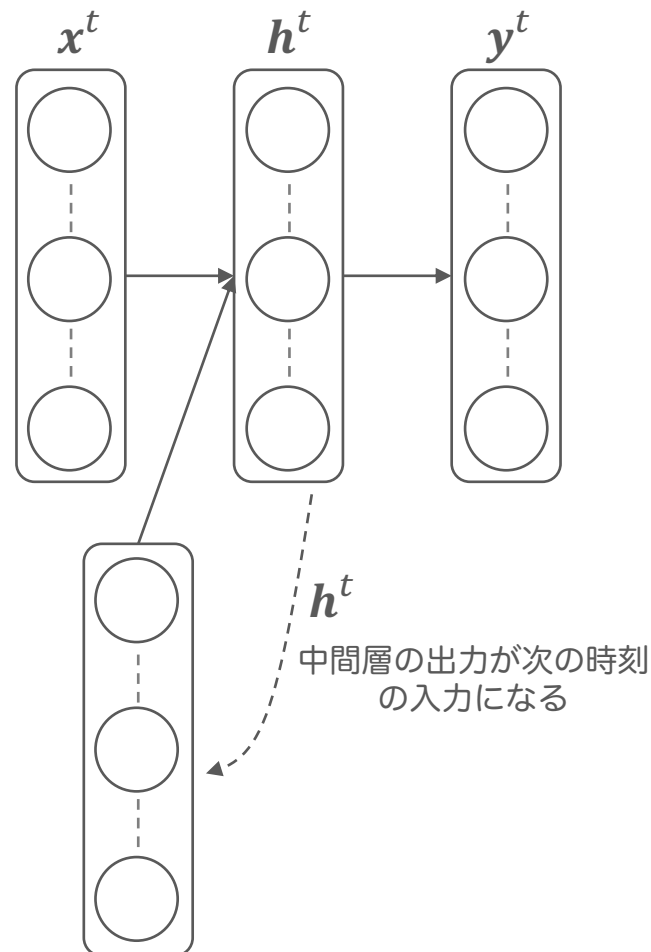
- RNNの計算グラフは、簡略化して表現されることが多い。

$$h^t = f(h^{t-1}W_h + x^t W_x + b)$$

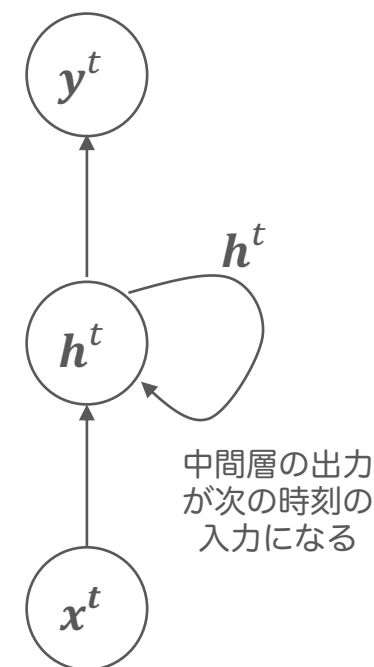
f : 活性化関数



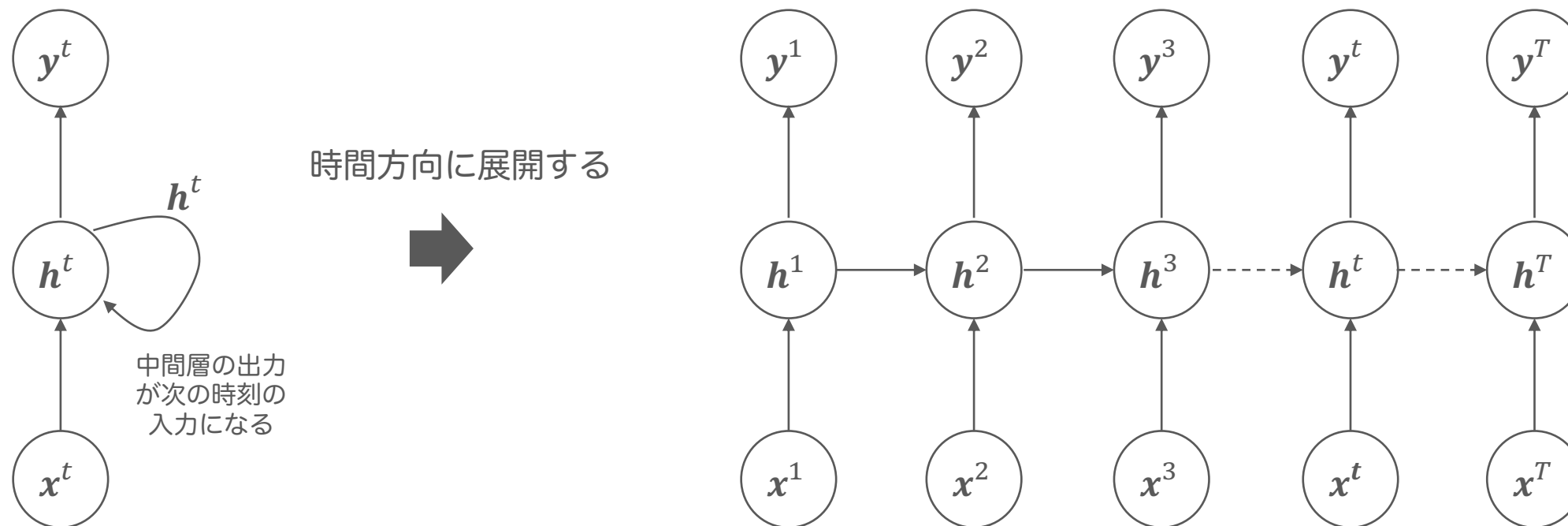
簡略化する



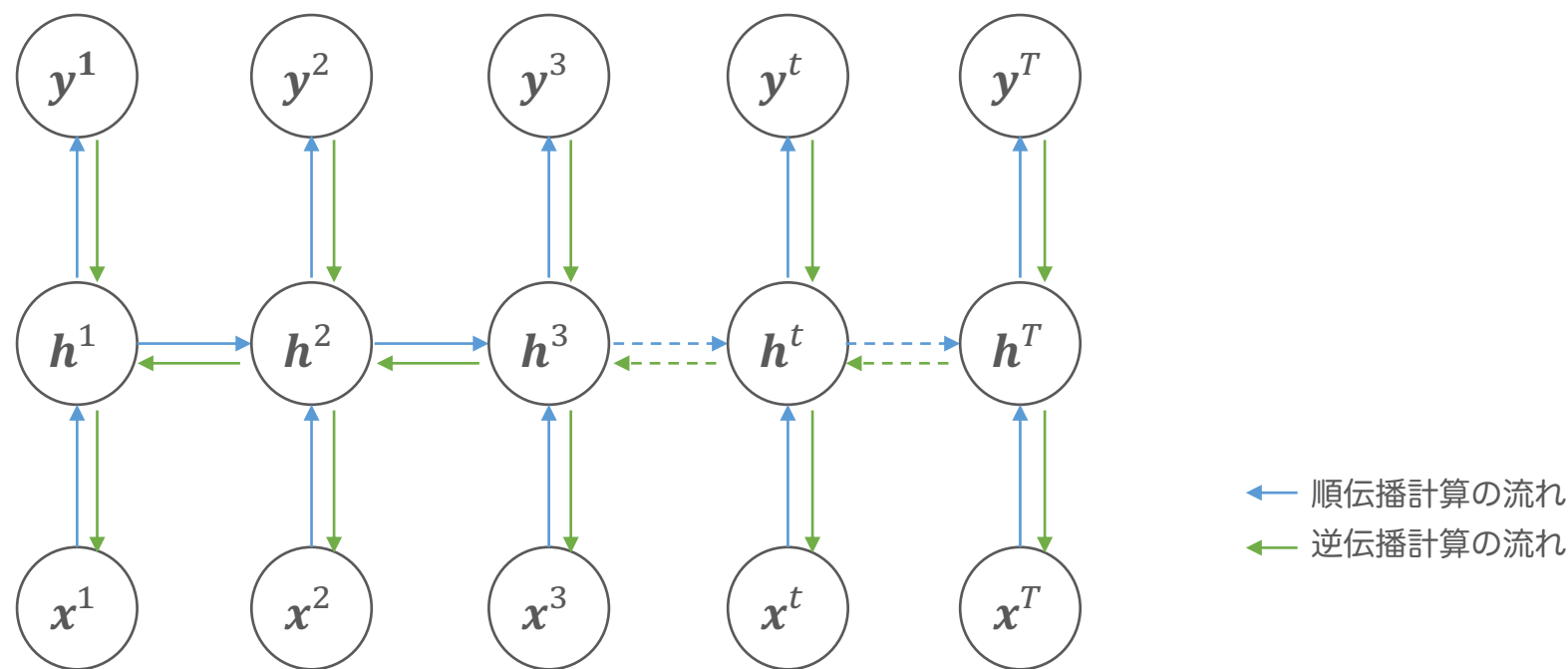
もっと簡略化する



- RNNの計算グラフを展開すると以下ようになる。



- 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?

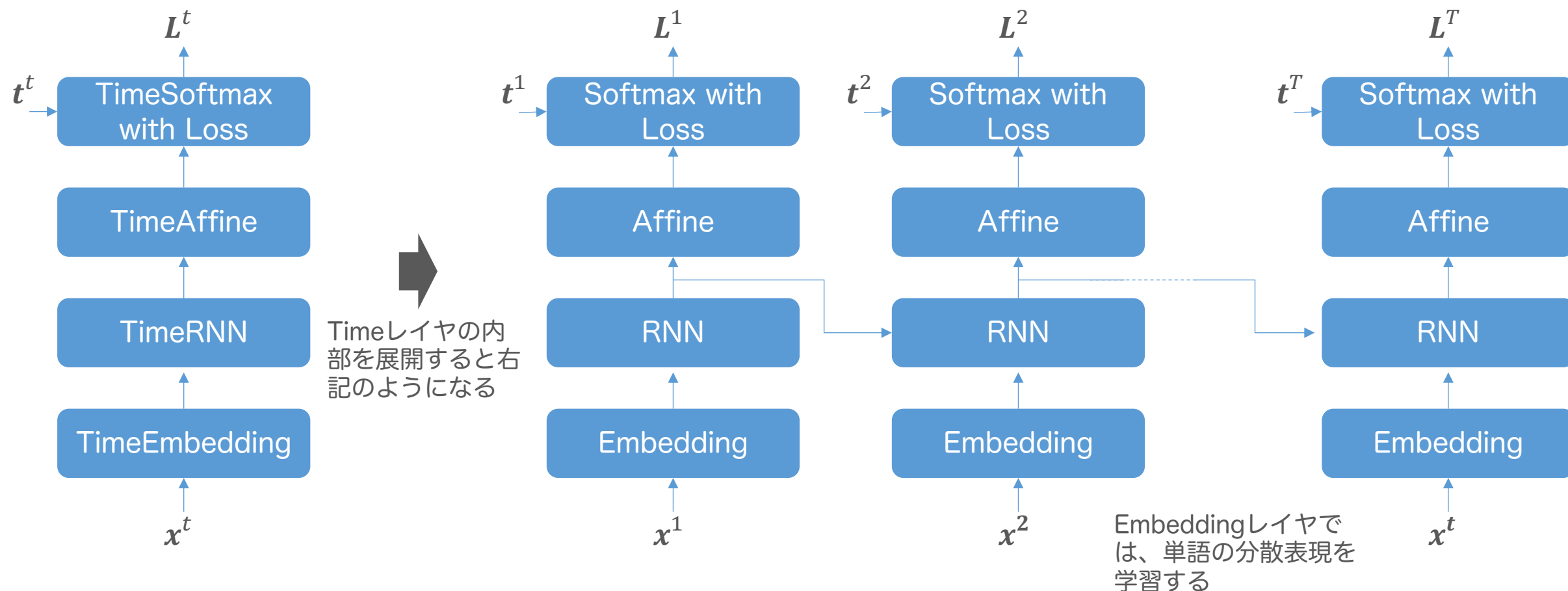
シンプルなRNN

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

$$H^t = f(H^{t-1}W_h + X^t W_x + B)$$

$$A_1 = H^{t-1}W_h + X^t W_x$$

$$A_2 = A_1 + B$$

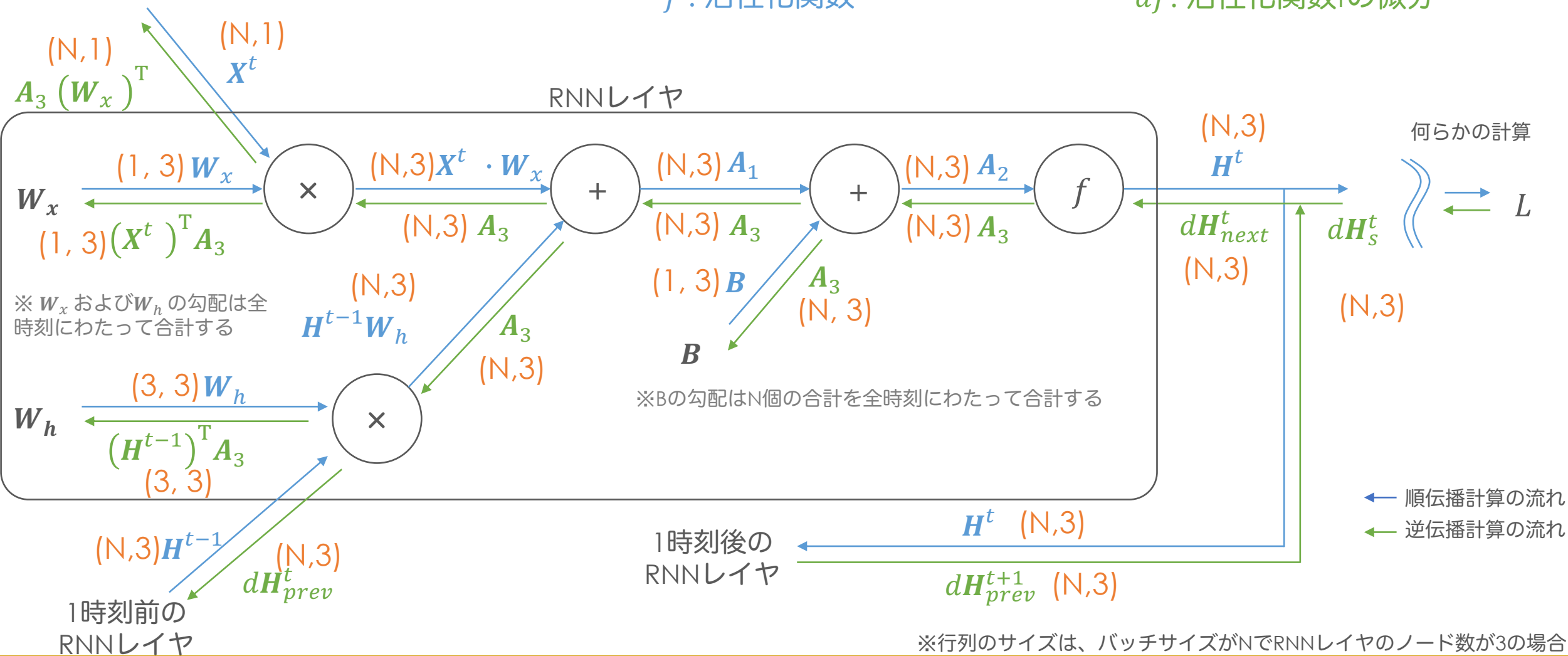
f : 活性化関数

$$dH_{next}^t = dH_{prev}^{t+1} + dH_s^t$$

$$A_3 = df \odot dH_{next}^t$$

$$dH_{prev}^t = A_3 (W_h)^T$$

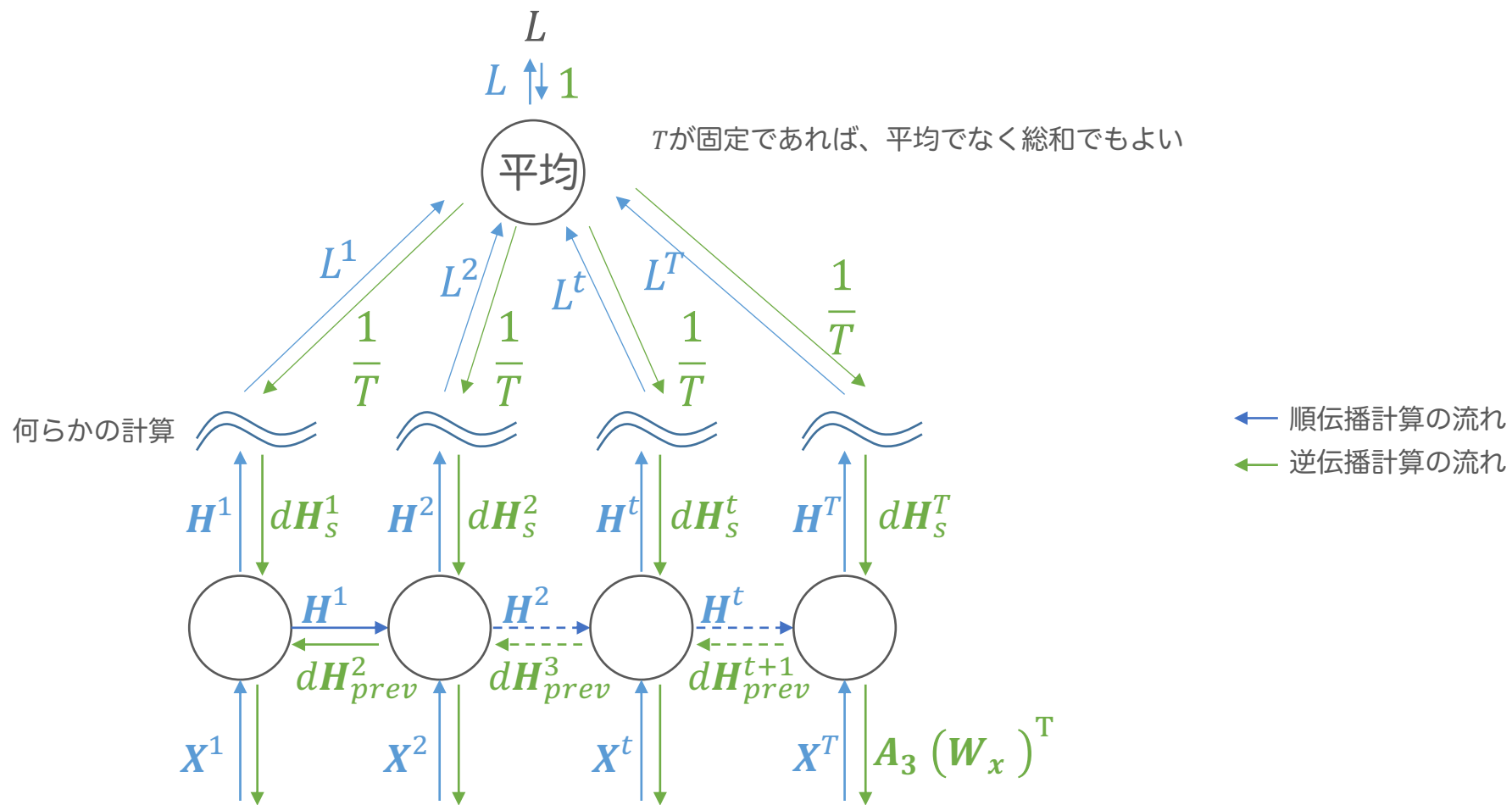
df : 活性化関数 f の微分



※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

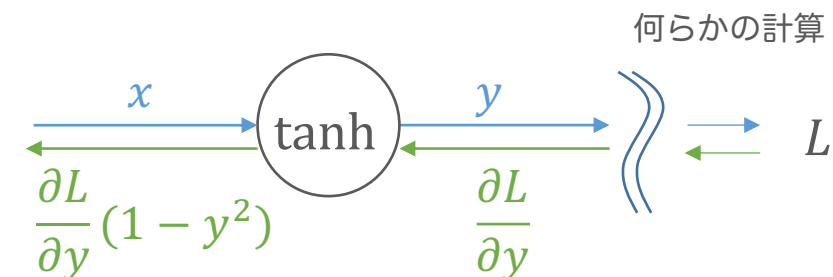
TimeRNNレイヤの計算グラフ

x^t 右上の添字は時刻を表す



tanhノード

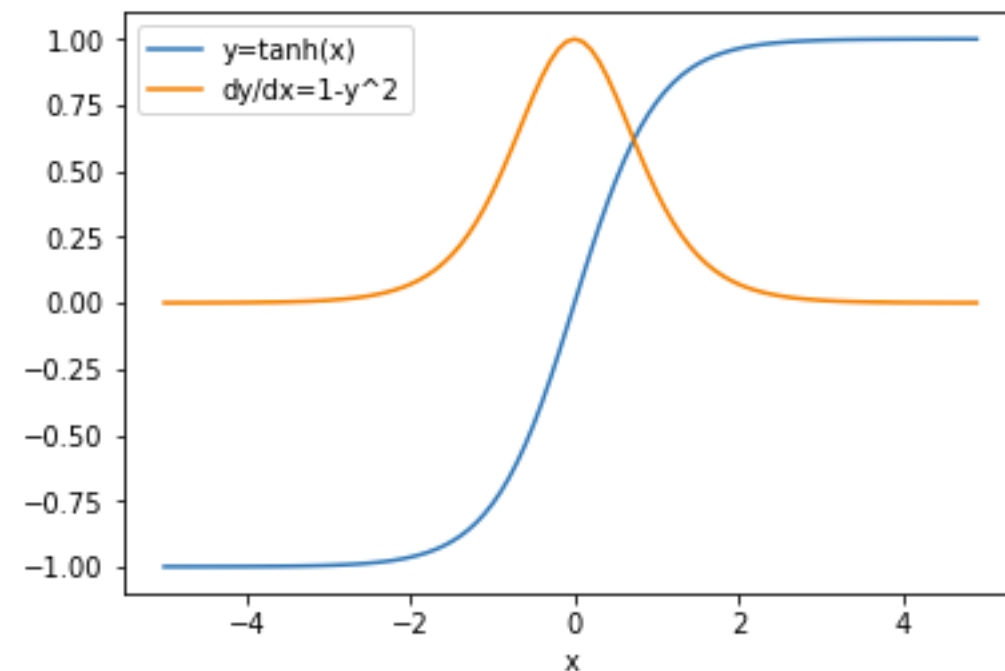
- RNNでは、活性化関数としてtanhがよく用いられる。
- tanhの微分値の最大値は1であり、シグモイド関数よりも勾配消失が起きにくい。
- tanhノードの順伝播および逆伝播を以下に示す。



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

分数関数の微分の公式を用いている

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



[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でReLUがあまり使われない理由

- RNNで、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。
- 重みの初期値を工夫することで、ReLUでも勾配爆発を抑えることができるという研究がいくつか見られる。
 - 参考: 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>

[演習] RNNレイヤの実装

- 6_2_threepoint_reader.ipynb
 - 以降のRNNの実装において、3点リーダー[...]を利用します。
 - ここでその動作を確認しておきましょう。
- 6_3_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_4_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_5_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_6_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_7_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_8_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。

長期依存性の課題

- 勾配消失、もしくは勾配爆発の問題により、RNNは長期の記憶を保持するのが難しい。
- RNNは入力に同じ処理を繰り返すために、ほぼ同じ値の重みを繰り返し掛けていくと値が発散するか0に近くなる。
 - $d\mathbf{H}_{prev}^t = df \odot d\mathbf{H}_{next}^t (\mathbf{W}_h)^T$ の式をみると、上記の理由がわかる。
- 全結合型のNNに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、勾配クリッピングという方法がある。DAY3スライドを参照。

[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_9_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。

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

$$F = \sigma \left(X^t W_x^{(f)} + H^{t-1} W_h^{(f)} + B^{(f)} \right)$$

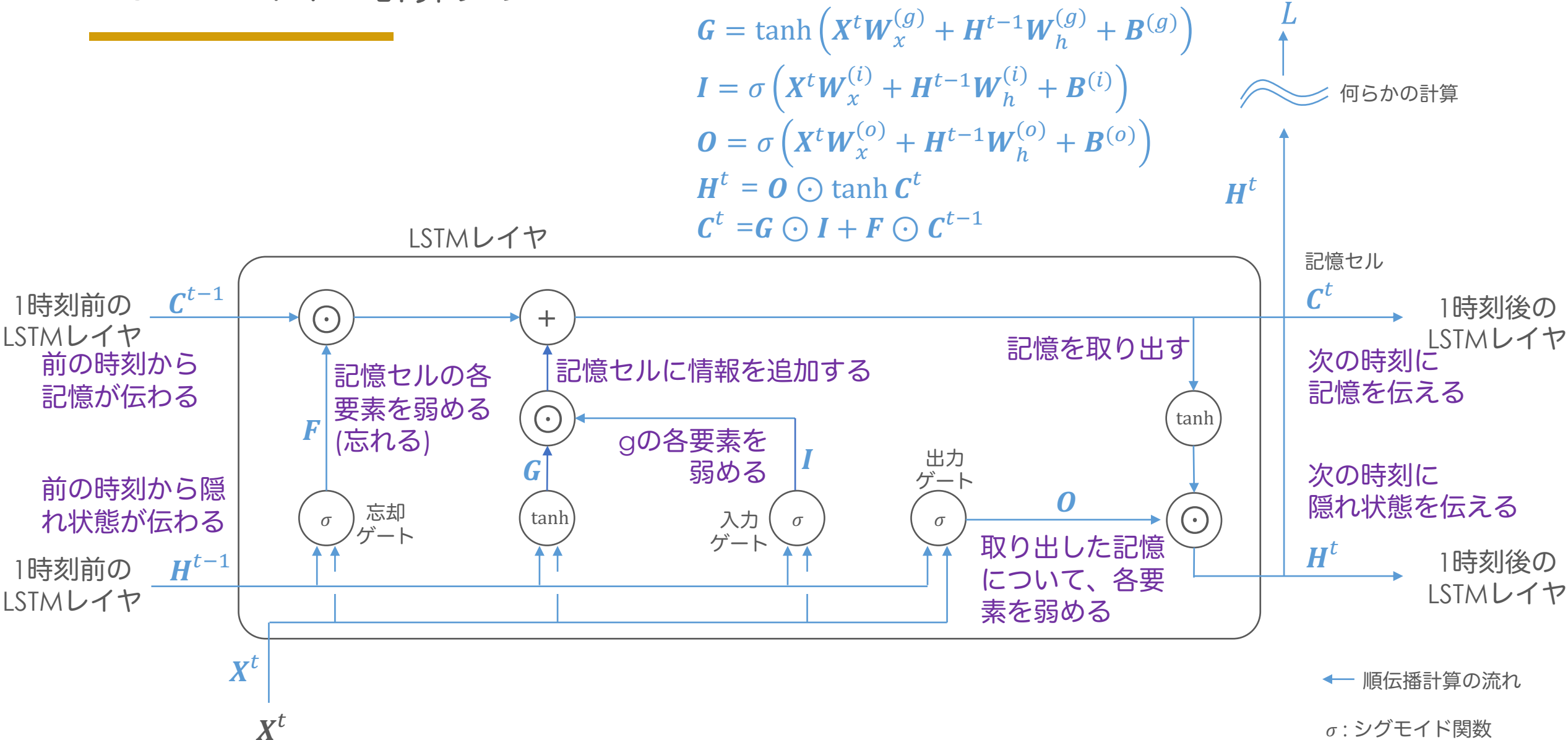
$$G = \tanh \left(X^t W_x^{(g)} + H^{t-1} W_h^{(g)} + B^{(g)} \right)$$

$$I = \sigma \left(X^t W_x^{(i)} + H^{t-1} W_h^{(i)} + B^{(i)} \right)$$

$$O = \sigma \left(X^t W_x^{(o)} + H^{t-1} W_h^{(o)} + B^{(o)} \right)$$

$$H^t = O \odot \tanh C^t$$

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



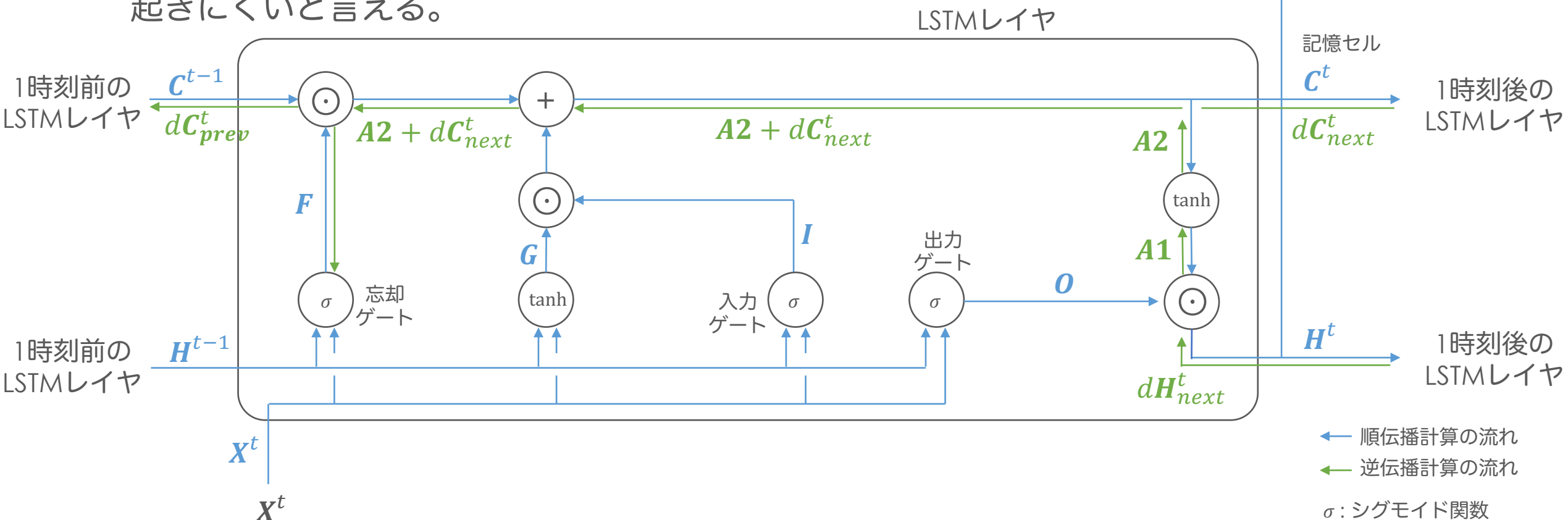
LSTMレイヤの計算グラフ

$$A1 = O \odot dH_{next}^t$$

$$A2 = (1 - \tanh(C^t)^2) \odot A1$$

$$dC_{prev}^t = F \odot (A2 + dC_{next}^t)$$

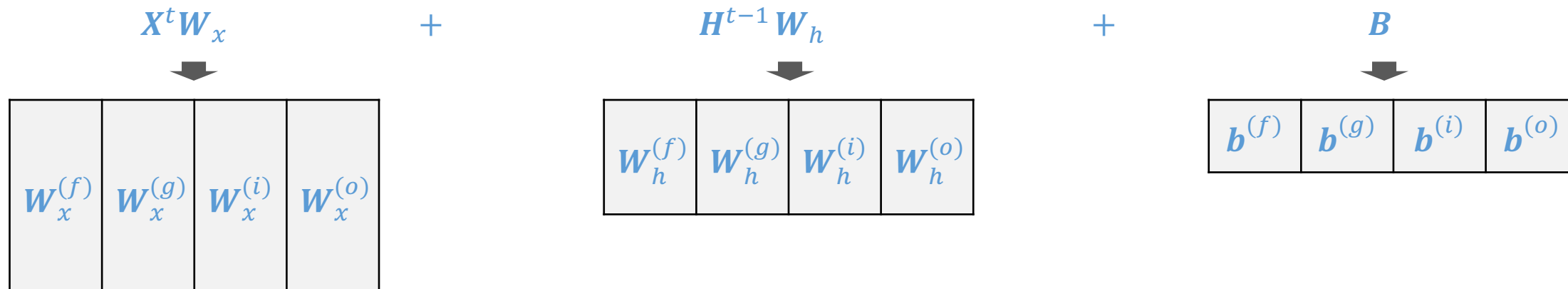
- 記憶セルの勾配をみると、シンプルなRNNの時のように、同じ重み行列を使って何度も行列の積を計算する構造になっていないことがわかる。これより、LSTMは、シンプルなRNNに比べ、勾配爆発や勾配消失が起きにくいと言える。



LSTMレイヤを効率的に実装する方法

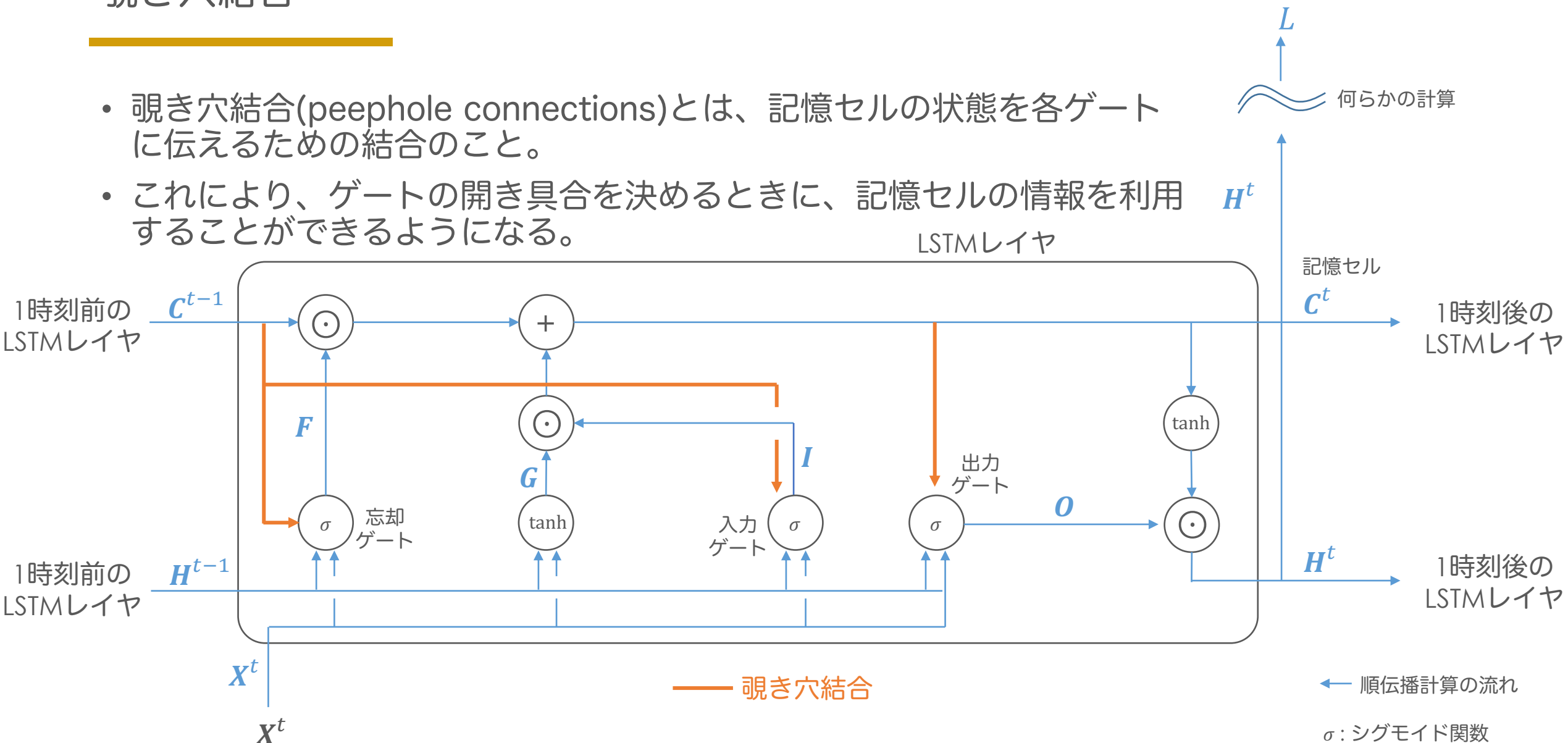
- 順伝播計算の式を見ると、 X^t または H^{t-1} が共通なので、 W_x^* または W_h^* を1つにまとめると、行列の積が1回で済む

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



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

$$F = \sigma \left(X^t W_x^{(f)} + H^{t-1} W_h^{(f)} + \underline{C^{t-1} W_c^{(f)}} + B^{(f)} \right)$$

$$G = \tanh \left(X^t W_x^{(g)} + H^{t-1} W_h^{(g)} + B^{(g)} \right)$$

$$I = \sigma \left(X^t W_x^{(i)} + H^{t-1} W_h^{(i)} + \underline{C^{t-1} W_c^{(i)}} + B^{(i)} \right)$$

$$O = \sigma \left(X^t W_x^{(o)} + H^{t-1} W_h^{(o)} + \underline{C^t W_c^{(o)}} + B^{(o)} \right)$$

$$H^t = O \odot \tanh C^t$$

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

- 覗き穴結合は、原著論文を見ると C と 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、 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。
 - Hasim Sak, Andrew Senior, Francoise 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、 C と 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

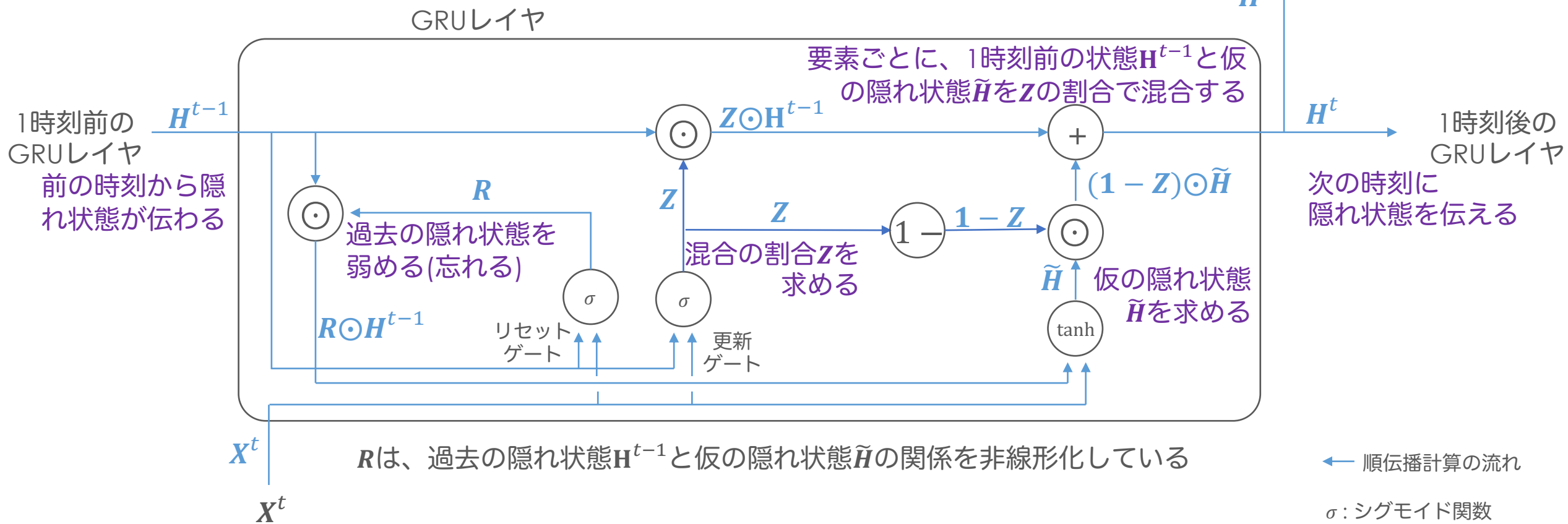
- GRUは、LSTMよりもシンプルなゲート付きRNN。
- LSTMよりもパラメータが少ない。
- GRUレイヤは、以下の2つのゲートで構成される。
 - リセットゲート(reset gate)
 - 過去の隠れ状態を弱める程度を決めるゲート
 - 更新ゲート(update gate)
 - 過去の隠れ状態と仮の隠れ状態の混合割合を決めるゲート
- 原著論文
 - Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, Kyunghyun Cho et al.,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

GRUレイヤの計算グラフ

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 H^t だけ。
 H^t だけで、記憶を長く保持する。

$$R = \sigma \left(X^t W_x^{(r)} + H^{t-1} W_h^{(r)} + B^{(r)} \right)$$
$$Z = \sigma \left(X^t W_x^{(z)} + H^{t-1} W_h^{(z)} + B^{(z)} \right)$$
$$\tilde{H} = \tanh \left(X^t W_x^{(\tilde{h})} + (R \odot H^{t-1}) W_h^{(\tilde{h})} + B^{(\tilde{H})} \right)$$
$$H^t = Z \odot H^{t-1} + (1 - Z) \odot \tilde{H}$$

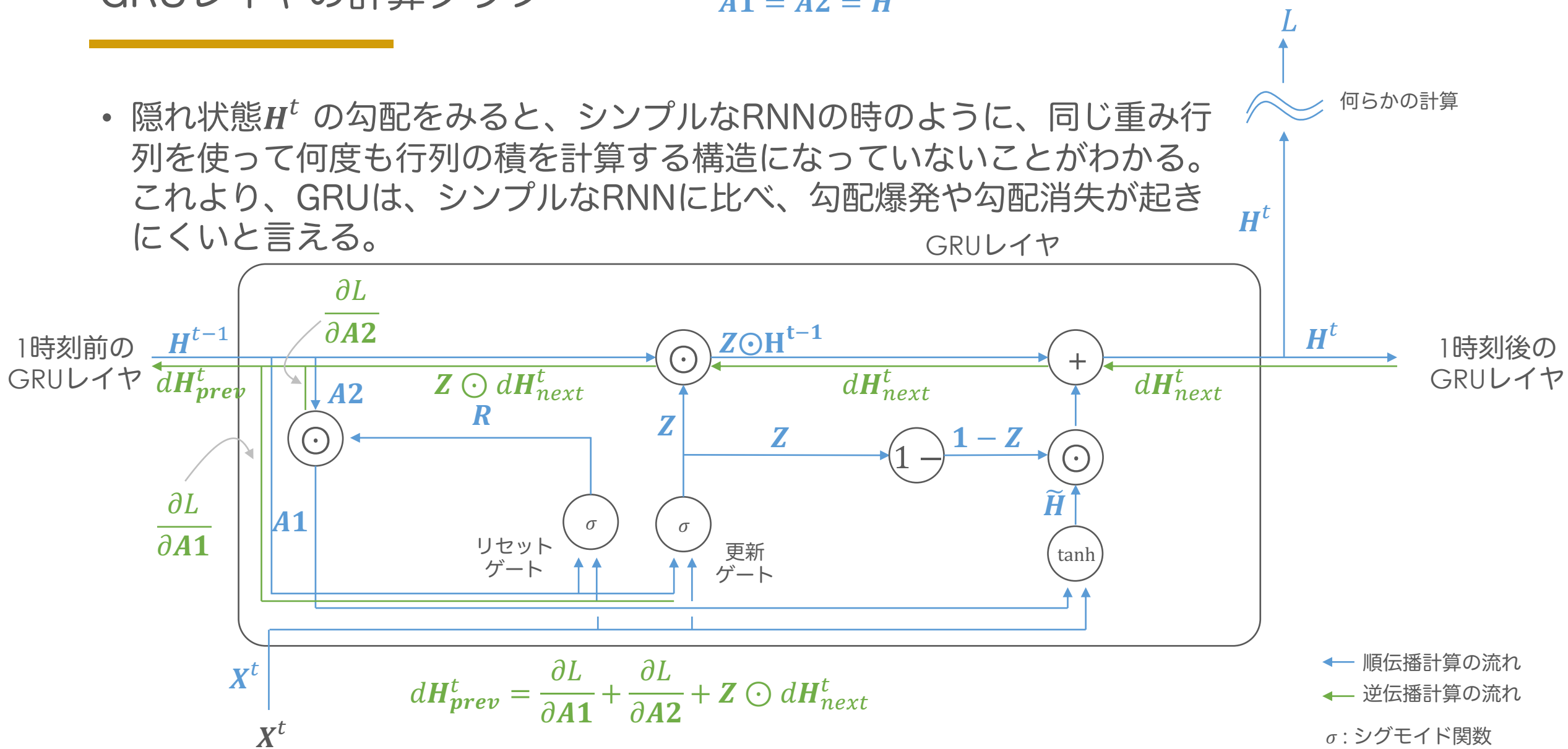
L
何らかの計算



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。



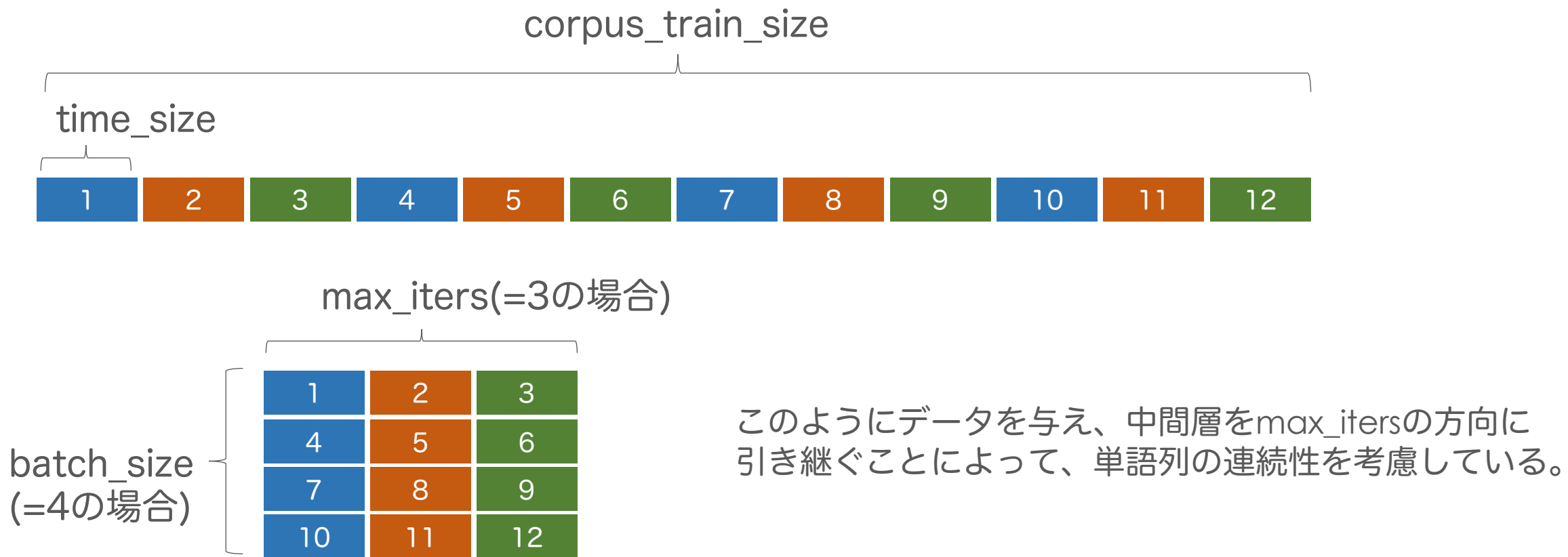
[演習] GRUレイヤの実装

- 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に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



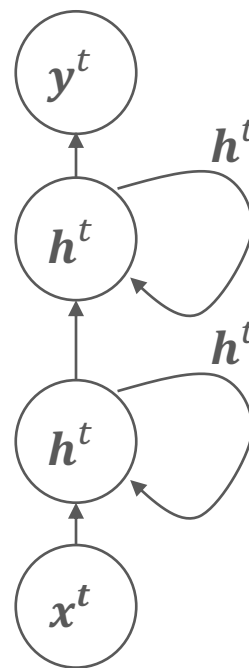
Any Questions?

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, p.287)

階層的にしたRNNの例

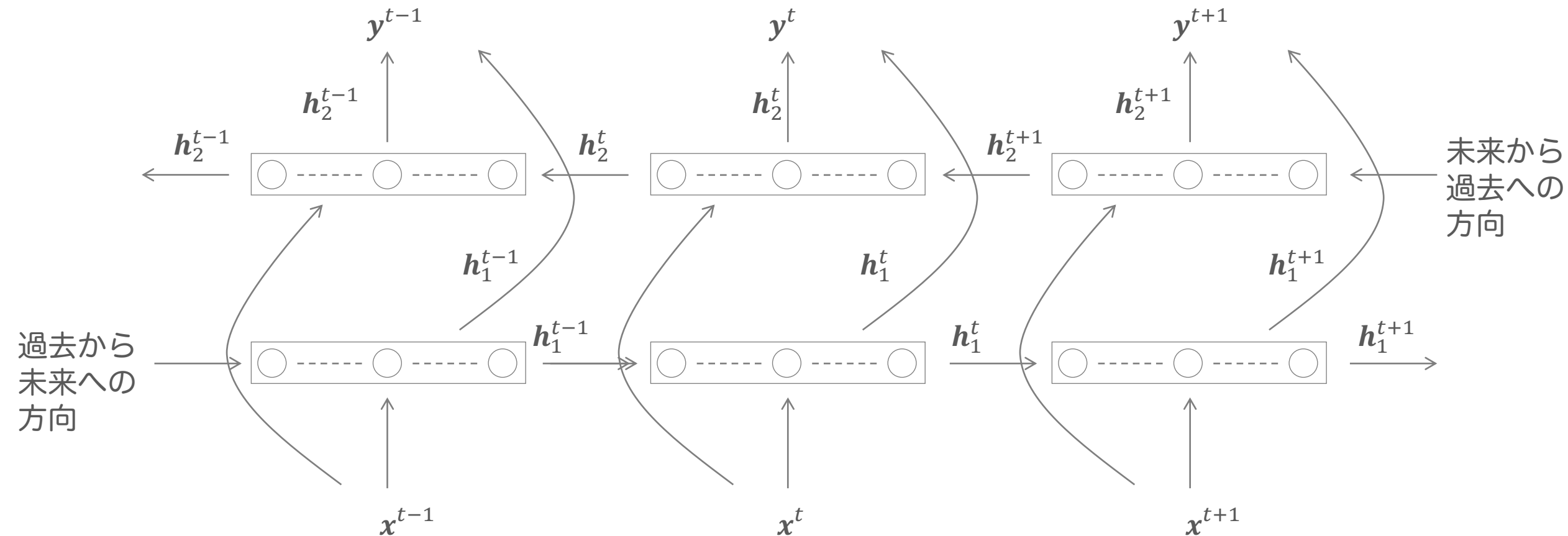


双方向RNN

- 過去から未来だけでなく、未来から過去への方角も考慮した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角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



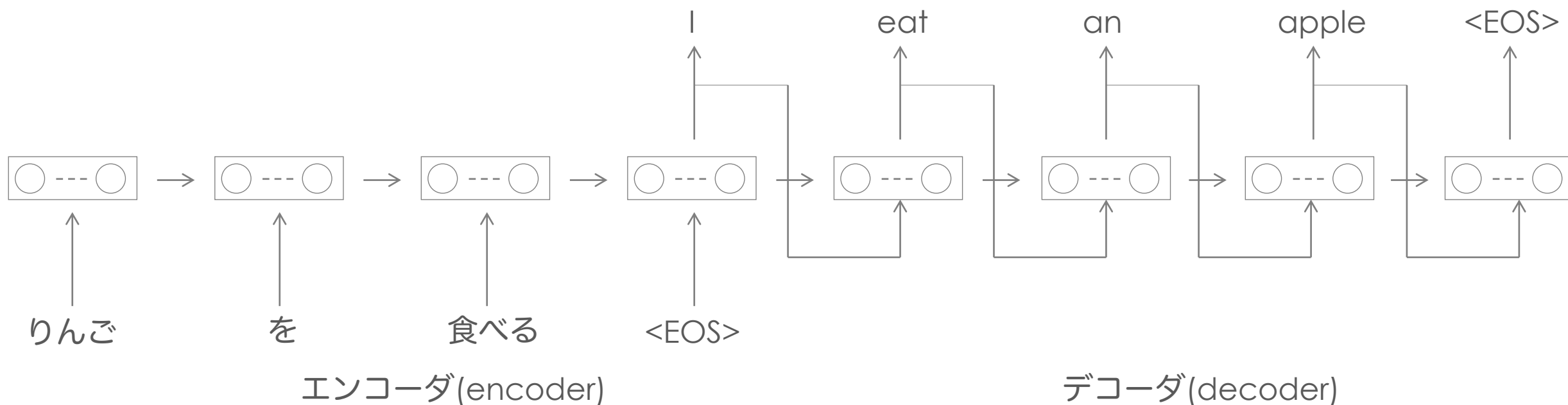
系列変換モデル

- 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

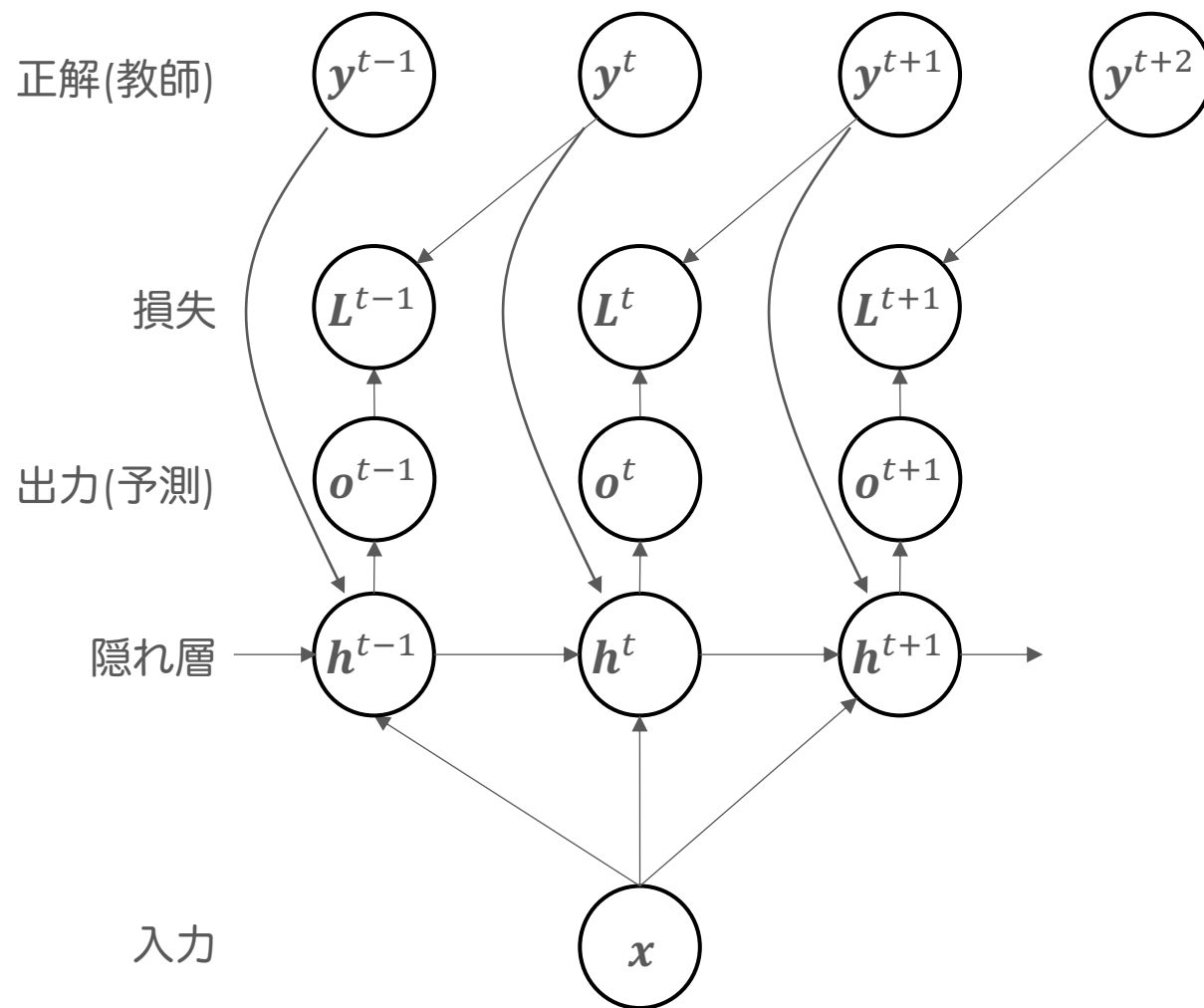
- 入力系列の長さと出力系列の長さは、一致していなくてもよい。
- 系列の長さは、データによって異なってもよい。

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, p.283



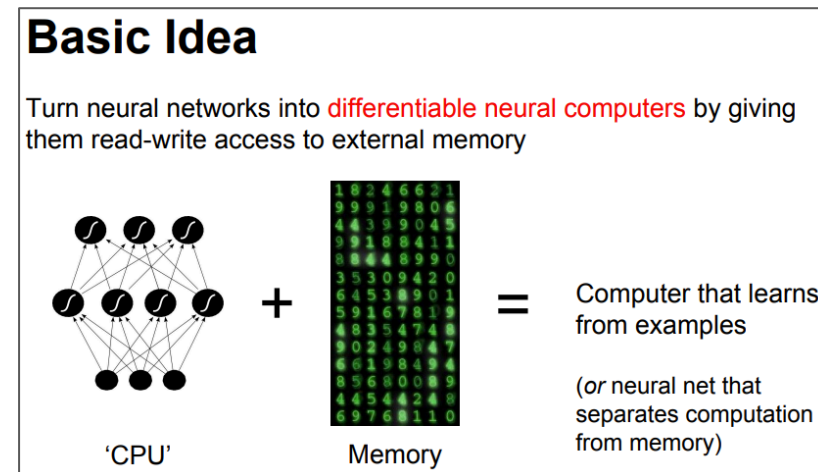
Any Questions?

外部メモリを持つニューラルネットワーク

外部メモリを持つニューラルネットワークとは

- 近年、外部メモリを持つニューラルネットワークがいくつか提案されている。
- 外部メモリを持つNNでは、通常のLSTMなどとは異なり、メモリとモデルを切り離して考える。これは、PCにおいて、CPUとメモリが別々に存在していることに例えられる。
- メモリを切り離すことにより、通常のLSTMなどよりも、表現力を豊かにできたり(必要な箇所のみ更新または取り出しができる)、多くの情報を蓄えられるようになる。
- メモリに対して、必要な情報を読み書きするアドレスを決める機構のことを**アテンション (attention)**という。
- 具体的なモデルの計算アルゴリズムについては、
DAY7で紹介する。

引用：Differentiable Neural Computers Alex Graves, Greg Wayne & many others
<http://people.idsia.ch/~rupesh/rnnsymposium2016/slides/graves.pdf>



外部メモリを持つニューラルネットワークの例

- 外部メモリを持つニューラルネットワークの例
 - Jason Weston et. al., Memory Networks, 2014
 - <https://arxiv.org/pdf/1410.3916.pdf>
 - <http://www.thespermwhale.com/jaseweston/icml2016/icml2016-memnn-tutorial.pdf>
 - Graves et. al., Neural Turing Machines, 2014
 - <https://arxiv.org/pdf/1410.5401.pdf>
 - Ankit Kumar et. al., Ask Me Anything: Dynamic Memory Networks for Natural Language Processing, 2015
 - <https://arxiv.org/pdf/1506.07285.pdf>
 - Sainbayar Sukhbaatar ,End-To-End Memory Networks, 2015
 - <https://arxiv.org/pdf/1503.08895.pdf>
 - Alex Graves et. al., Differentiable Neural Computers, 2016
 - Neural Turing Machine (NTM) の後継。より複雑なタスクを解けるようになった。
 - <https://www.nature.com/articles/nature20101>
 - <http://people.idsia.ch/~rupesh/rnnsymposium2016/slides/graves.pdf>
 - https://www.nature.com/articles/nature20101.epdf?author_access_token=lmTXBI8aWbYxYQ51Plys8NRgN0jAjWel9jnR3ZoTv0MggmpDmwIjGswxVdeocYSurJ3hxupzWuRNeGvvXnoO8o4jTJcnAyhGuZzXJ1GEaD-Z7E6X_a9R-xqJ9TfJWBqz

外部メモリを持つニューラルネットワークでできることの例

- 外部メモリを持つニューラルネットワークでできること

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.
Where is the milk now? **A: office**
Where is Joe? **A: bathroom**
Where was Joe before the office? **A: kitchen**

Sheep are afraid of wolves.
Cats are afraid of dogs.
Mice are afraid of cats.
Gertrude is a sheep.
What is Gertrude afraid of? **A:wolves**

引用：Jason Weston et. al., Memory Networks, 2014

上：<https://arxiv.org/pdf/1410.3916.pdf>

下：

<http://www.thespermwhale.com/jaseweston/icml2016/icml2016-memnn-tutorial.pdf>

外部メモリを持つニューラルネットワークの精度比較

- 各手法の精度比較を以下に示す。

NTM:Neural Turing Machines

DNC : Differentiable Neural Computers

MemN2N : End-To-End Memory Networks

DM : Dynamic Memory Networks

bAbI Results

Task	bAbI Best Results						
	LSTM (Joint)	NTM (Joint)	DNC1 (Joint)	DNC2 (Joint)	MemN2N (Joint) ²¹	MemN2N (Single) ²¹	DMN (Single) ²⁰
1: 1 supporting fact	24.5	31.5	0.0	0.0	0.0	0.0	0.0
2: 2 supporting facts	53.2	54.5	1.3	0.4	1.0	0.3	1.8
3: 3 supporting facts	48.3	43.9	2.4	1.8	6.8	2.1	4.8
4: 2 argument rels.	0.4	0.0	0.0	0.0	0.0	0.0	0.0
5: 3 argument rels.	3.5	0.8	0.5	0.8	6.1	0.8	0.7
6: yes/no questions	11.5	17.1	0.0	0.0	0.1	0.1	0.0
7: counting	15.0	17.8	0.2	0.6	6.6	2.0	3.1
8: lists/sets	16.5	13.8	0.1	0.3	2.7	0.9	3.5
9: simple negation	10.5	16.4	0.0	0.2	0.0	0.3	0.0
10: indefinite knowl.	22.9	16.6	0.2	0.2	0.5	0.0	0.0
11: basic coreference	6.1	15.2	0.0	0.0	0.0	0.1	0.1
12: conjunction	3.8	8.9	0.1	0.0	0.1	0.0	0.0
13: compound coref.	0.5	7.4	0.0	0.1	0.0	0.0	0.2
14: time reasoning	55.3	24.2	0.3	0.4	0.0	0.1	0.0
15: basic deduction	44.7	47.0	0.0	0.0	0.2	0.0	0.0
16: basic induction	52.6	53.6	52.4	55.1	0.2	51.8	0.6
17: positional reas.	39.2	25.5	24.1	12.0	41.8	18.6	40.4
18: size reasoning	4.8	2.2	4.0	0.8	8.0	5.3	4.7
19: path finding	89.5	4.3	0.1	3.9	75.7	2.3	65.5
20: agent motiv.	1.3	1.5	0.0	0.0	0.0	0.0	0.0
Mean Err. (%)	25.2	20.1	4.3	3.8	7.5	4.2	6.4
Failed (err. > 5%)	15	16	2	2	6	3	2

引用 : Differentiable Neural Computers Alex Graves, Greg Wayne & many others

<http://people.idsia.ch/~rupesh/rnnsymposium2016/slides/graves.pdf>

bAbIプロジェクトとは

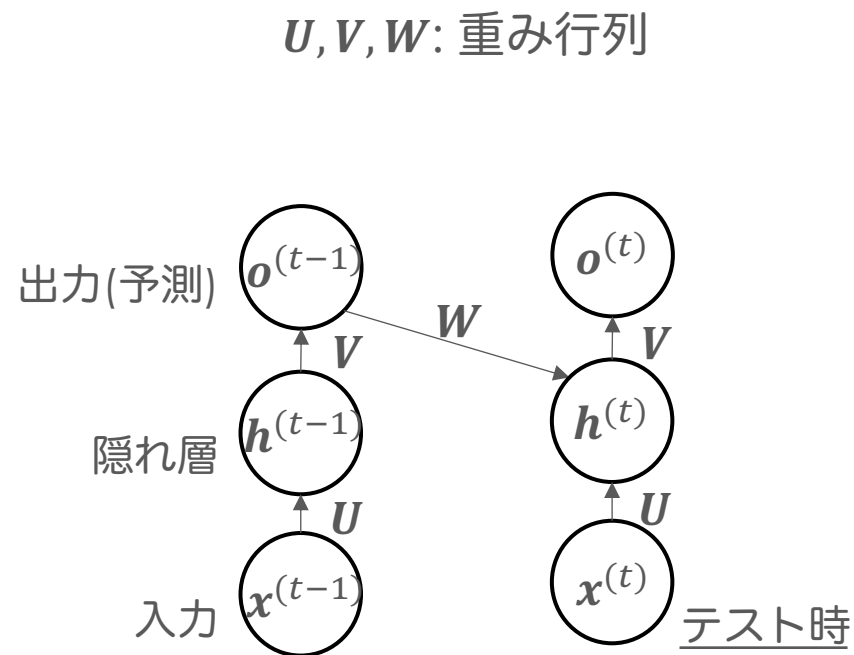
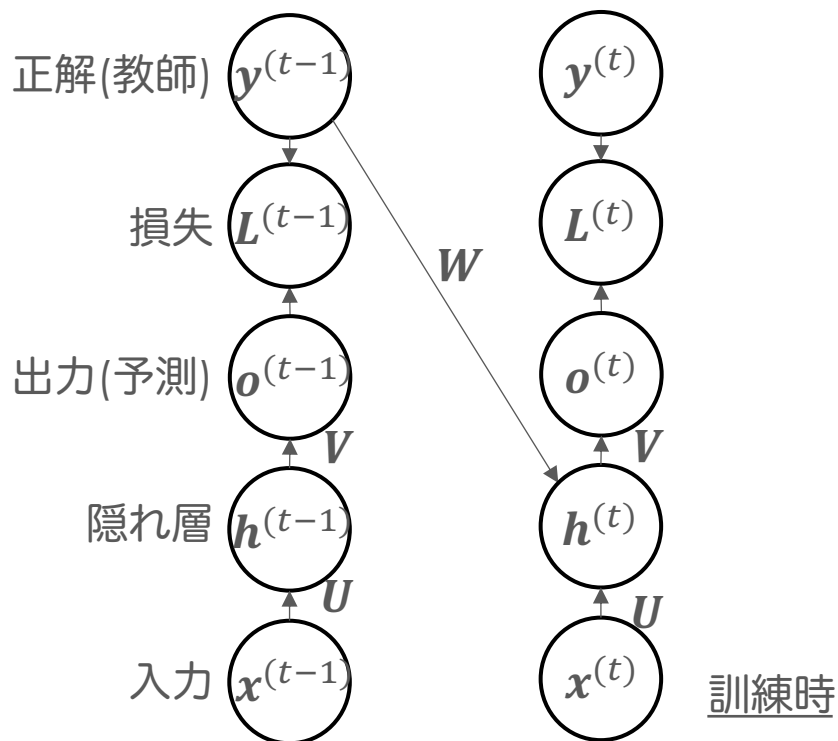
- bABIプロジェクトとは、文章理解や文章意味付けの自動化を目指すプロジェクト。
- Facebook AI Research が主導している。
- このプロジェクトでは、言語理解を測るためのテキストデータセットを提供している (bAbIタスク)。
- <https://research.fb.com/downloads/babi/>

Any Questions?

その他の話題

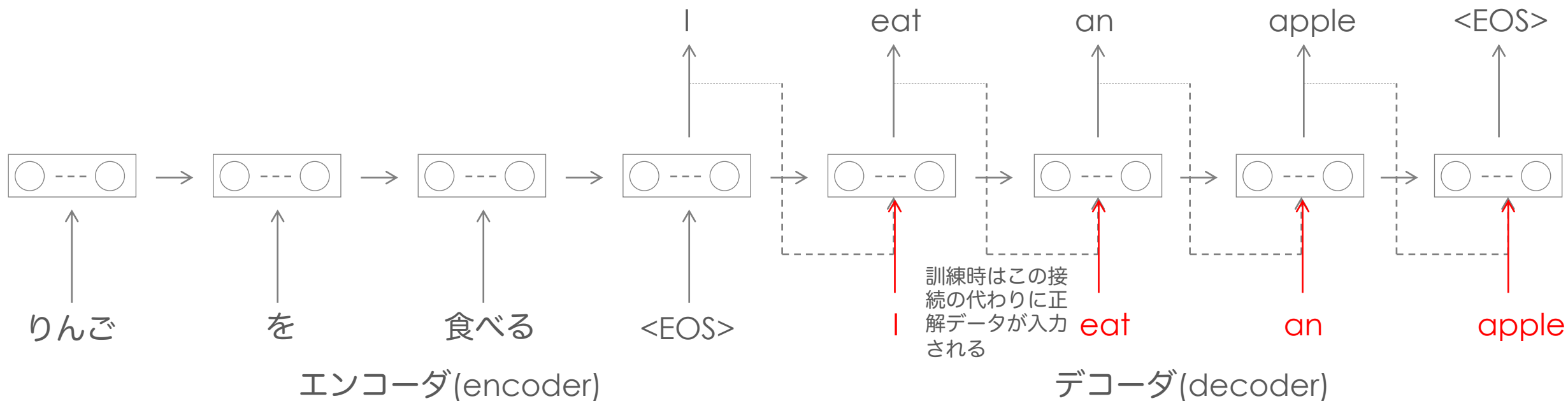
教師強制

- 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。
- 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。



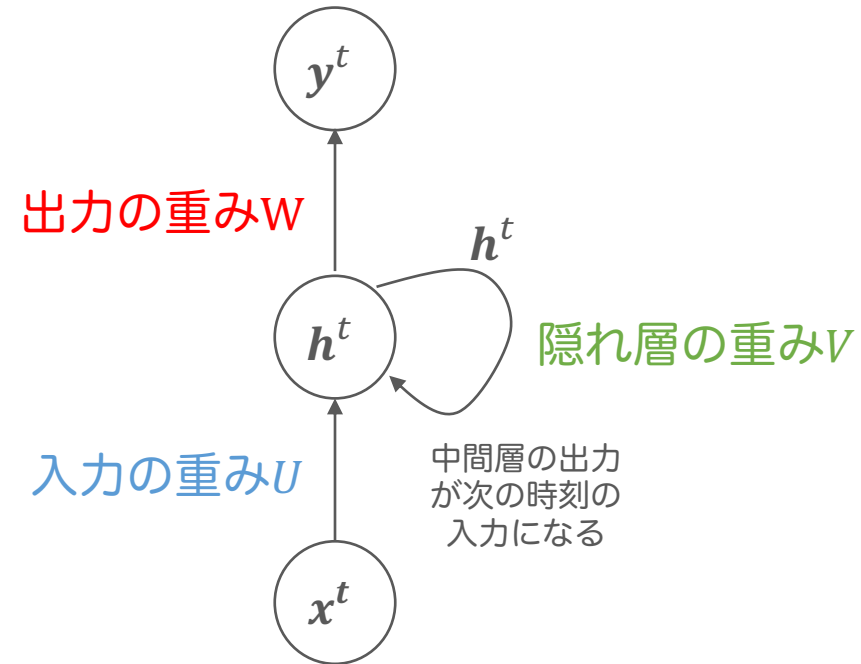
教師強制の適応例

- 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。



エコーステートネットワーク

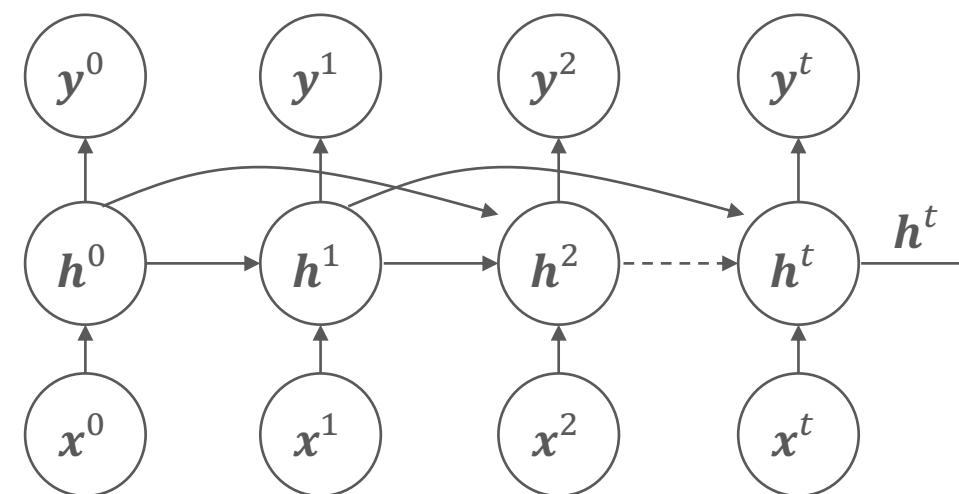
- エコーステートネットワーク(echo state networks)では、**隠れ層から出力層への重み(出力の重み) W** のみを学習し、**入力層から隠れ層への重み(入力の重み) U** 、および**隠れ層の重み V** は固定する。
- **入力の重み U** と**隠れ層の重み V** は、過去の情報を捕捉できる程度の重み(ランダム設定など)に固定し、出力の重みだけを学習することに専念することによって計算時間の短縮を図っている。
- シンプルな線形回帰の問題になるので、勾配消失や勾配爆発は発生しない。
- 学習速度も速い。
- 参考:lan Goodfellow, 深層学習, KADOKAWA,p.291



時間方向のスキップ接続

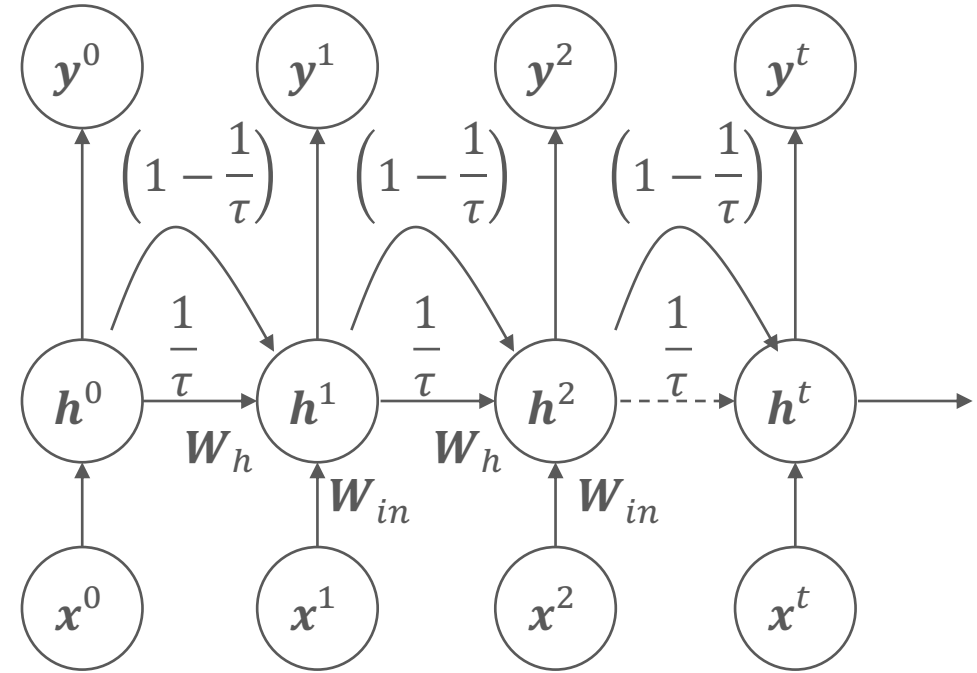
- 長期依存性の課題への対策の一つとして、過去の隠れ層から現在の隠れ層への接続を追加する方法がある。
- このような接続をスキップ接続という。
- スキップ接続により長期の依存性を得ることがある。
- 参考: Ian Goodfellow, 深層学習, KADOKAWA, p.293

離れた過去と接続する例



Leakyユニット

- 隠れ層に線形結合を導入すると、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: Ian Goodfellow, 深層学習, KADOKAWA, p.293



$$h_t = \left(1 - \frac{1}{\tau}\right) h_{t-1} + \frac{1}{\tau} f(W_h h_{t-1} + W_{in} x_t)$$

f : 活性化関数

W_h : 隠れ層の重み

W_{in} : 入力重み

τ : 係数

<移動平均>

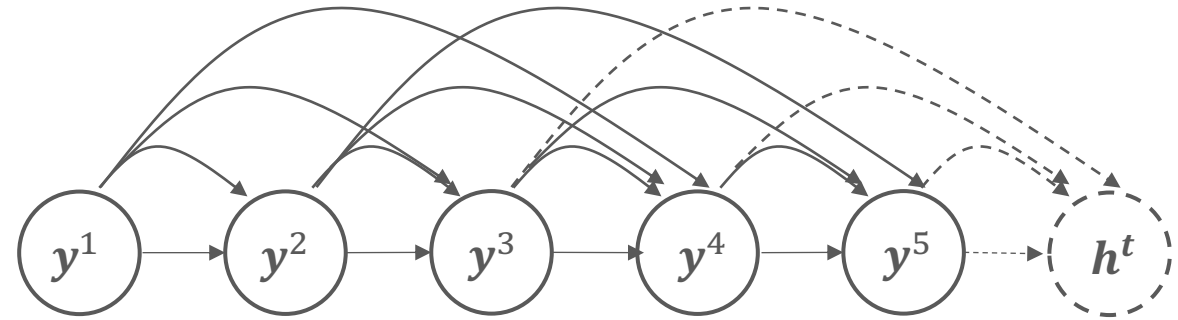
$$\mu^{(t)} \leftarrow \alpha \mu^{(t-1)} + (1 - \alpha) v^{(t)}$$

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。

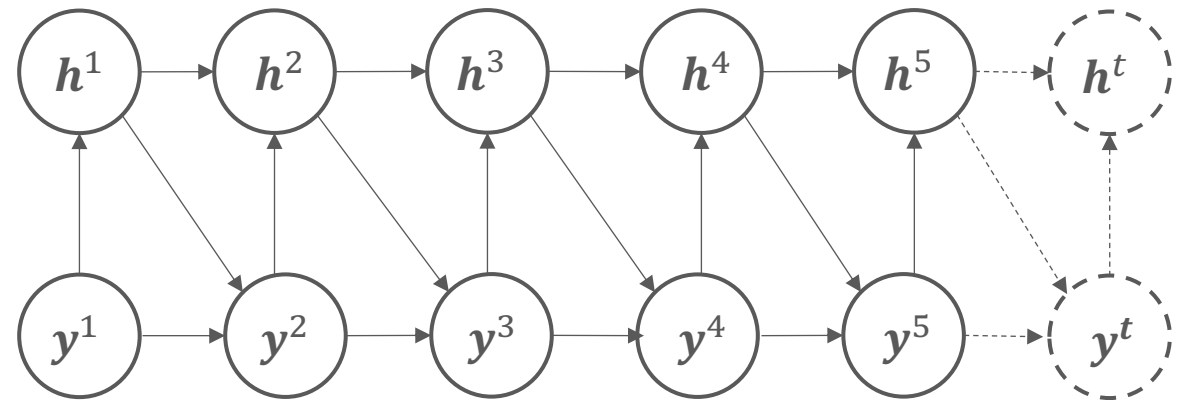
有向グラフィカルモデルとしての再帰型ニューラルネットワーク

- 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: Ian Goodfellow, 深層学習, KADOKAWA, p.280

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] 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、 $W_x^{(f)}, W_x^{(g)}, W_x^{(i)}, W_x^{(o)}, W_h^{(f)}, W_h^{(g)}, W_h^{(i)}, W_h^{(o)}$ 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、 $W_x^{(r)}, W_x^{(z)}, W_x^{(\tilde{h})}, W_h^{(r)}, W_h^{(z)}, W_h^{(\tilde{h})}$ 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

- 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。
- それたタスクに適した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)

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。