SPRINT23

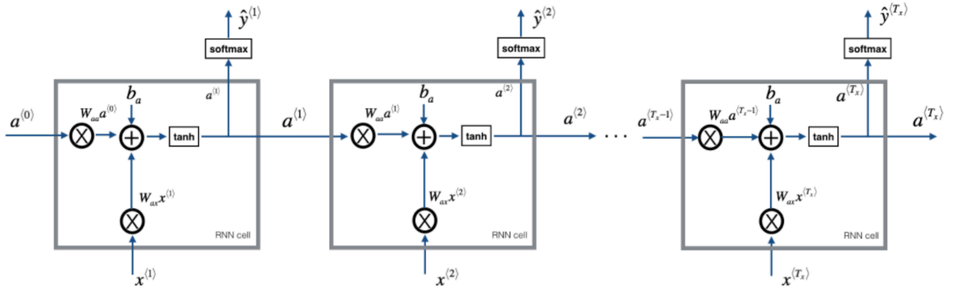
RNNとは

（背景）

RNNは、各タイムステップにおいて、以前のタイムステップからアクティベーションされた値を**情報として加えられる**というものでした。具体的には、この値に対し隠れ状態から隠れ状態への推移行列（transition matrix）が**掛け合わされて足し合わされます**。

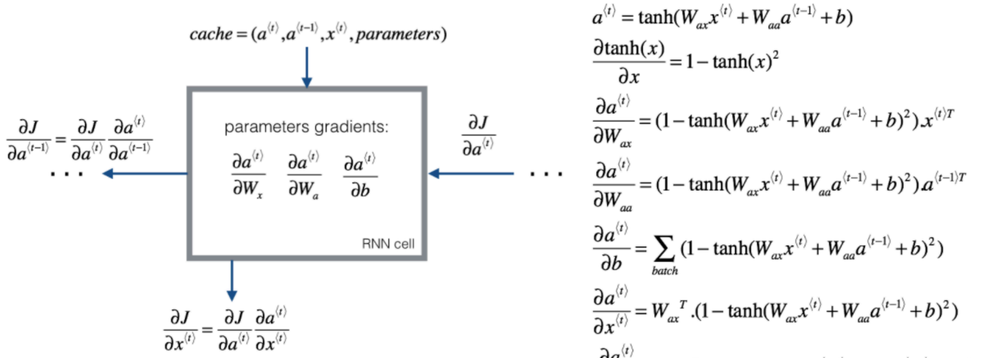
この推移行列の特異値が1（絶対値）以外であるとき、勾配消失または勾配爆発を起こすという問題がありました。

RNN Forward Pass



[Andrew Ng, Sequential Models Course, Deep Learning Specialization]

RNN Backward Pass



$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b)$$

$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh(x)^2$$

$$\frac{\partial a^{\langle t \rangle}}{\partial W_{ax}} = (1 - \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b)^2)x^{\langle t \rangle T}$$

$$\frac{\partial a^{\langle t \rangle}}{\partial W_{aa}} = (1 - \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b)^2)a^{\langle t-1 \rangle T}$$

$$\frac{\partial a^{\langle t \rangle}}{\partial b} = \sum_{batch} (1 - \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b)^2)$$

$$\frac{\partial a^{\langle t \rangle}}{\partial x^{\langle t \rangle}} = W_{ax}{}^T.(1 - \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b)^2)$$

$$\frac{\partial a^{\langle t \rangle}}{\partial a^{\langle t-1 \rangle}} = W_{aa}{}^T.(1 - \tanh(W_{ax}x^{\langle t-1 \rangle} + W_{aa}a^{\langle t-1 \rangle} + b)^2)$$

[Andrew Ng, Sequential Models Course, Deep Learning Specialization]

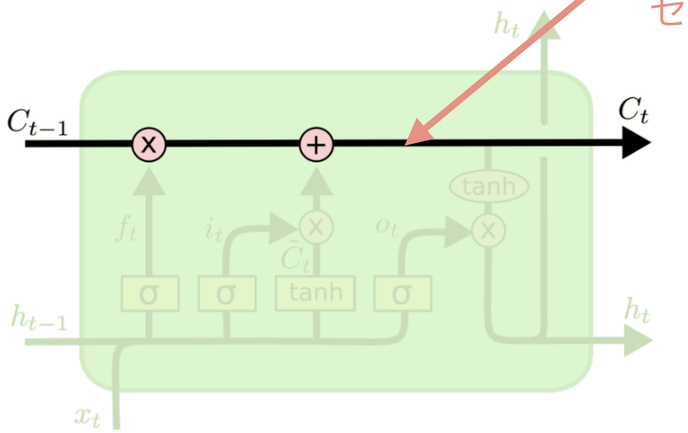
LSTMとは

Long Short-Term Memory（1997年）

https://www.bioinf.jku.at/publications/older/2604.pdf

勾配の消失の問題を解決する方法として
1997年にSepp HochreiterとJürgenSchmidhuberによって提案されました。

ポイント：
セルの情報を保持および制御する機構が加わった



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

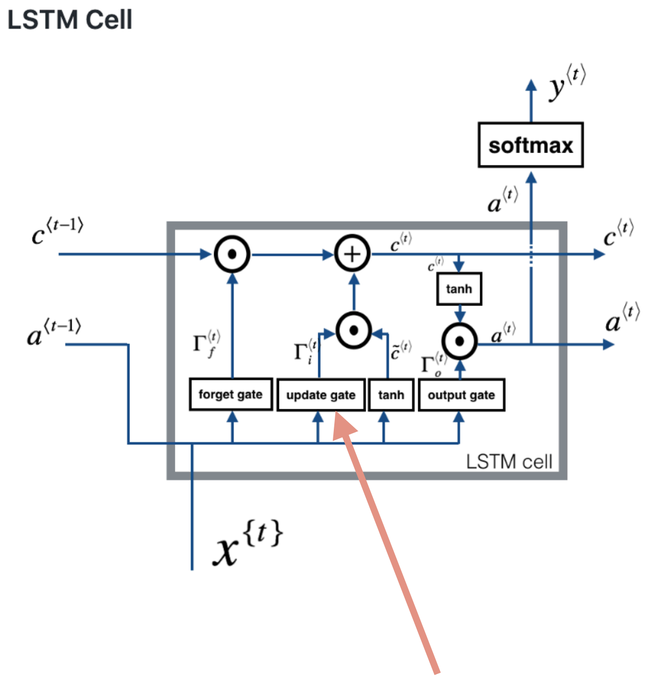
LSTMのネットワーク

RNNと比較されるLSTMのネットワークの特徴は、状態を保存するユニットと以下の**ゲート（gate）と呼ばれる制御の機構**を保有することです。

Input gate：情報を書き込むかどうかを制御する(write)

Output gate：情報を出力するかどうかを制御する(read)

Forget gate：情報を削除かどうかを制御する(forget)



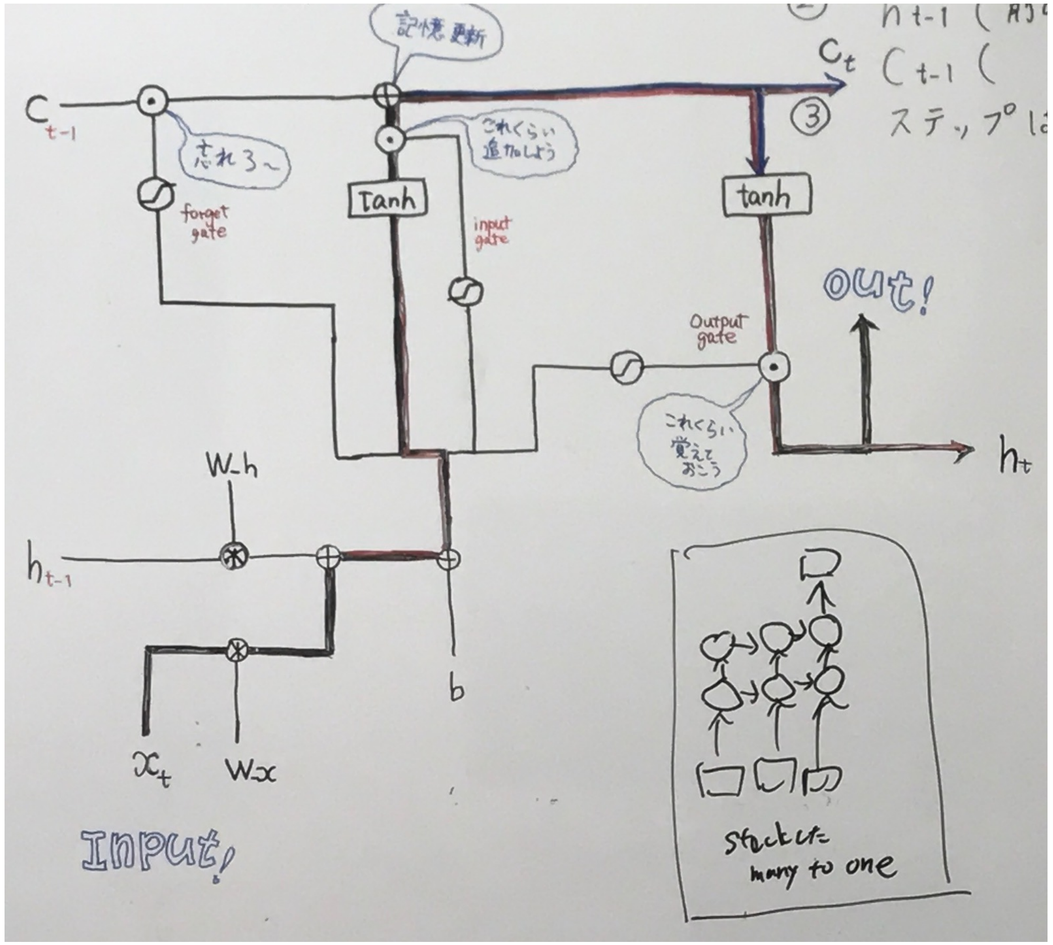
- Input gate:
  - 0.0のとき、何も書き込まない
  - 1.0のとき、全部書き込む
  - 0.0~1.0の間、一部書き込む
- Output gate:
  - 0.0のとき、何も出力しない
  - 1.0のとき、全部出力する
  - 0.0~1.0の間、一部出力する
- Forget gate:
  - 0.0のとき、全部削除する
  - 1.0のとき、何も削除しない
  - 0.0~1.0の間、一部削除する

Input gateは、Update gateと呼ばれることも

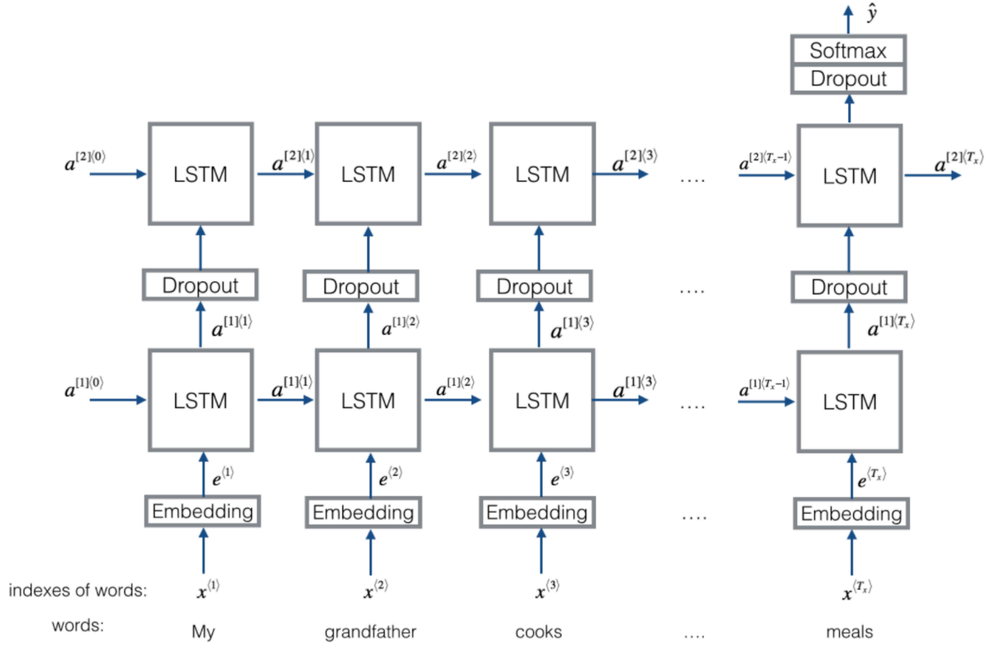
LSTMのネットワーク

先述した構造によって、LSTMは**セル状態**と呼ばれる長期記憶ユニットと、**隠れ状態**と呼ばれる短期記憶ユニットという部分を持つことになります。これらのユニットがタイムステップと層を通じて誤差逆伝搬します。エラーの減衰を防ぐ本質的な装置は**長期記憶のプラス**（加算）の部分です（乗算の方ではなく）。

ゲートはコンピュータのメモリのように、セルに情報を読み込み、書き出しする役割を果たします。メモリと異なる点は、ゲートが**入力、出力、削除をどれだけ許可するかを学習する**ところです。



LSTMのネットワーク



[Andrew Ng, Sequential Models Course, Deep Learning Specialization]

Parameters:

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 10)	0
embedding_1 (Embedding)	(None, 10, 50)	20000050
lstm_1 (LSTM)	(None, 10, 128)	91648
dropout_1 (Dropout)	(None, 10, 128)	0
lstm_2 (LSTM)	(None, 128)	131584
dropout_2 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 5)	645
activation_1 (Activation)	(None, 5)	0

Total params: 20,223,927
Trainable params: 20,223,927
Non-trainable params: 0

LSTMチューニングのベストプラクティス

https://cs.stackexchange.com/questions/79241/what-is-temperature-in-lstm-and-neural-networks-generally

① 学習率が高いとperplexity（確率の逆数によって、予測候補をどれくらい絞り込めたかを表す指標。正解ラベルを用いることなく計算可能）が発散する。

② softmax sample temperature（softmaxの温度）の調整。

③tanhの代わりにsoftsignを用いる（計算が早く、飽和しにくい）。tanhは指数関数的に収束し、softsignは多項式的に収束する。(p8)

④parameters > samples は過剰適合（オーバーフィッティング）する。

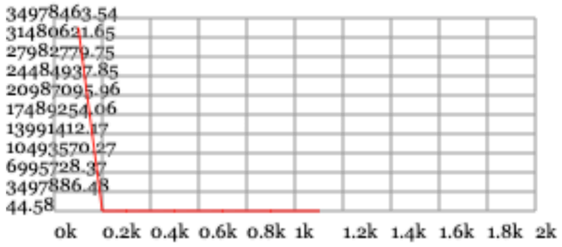
⑤レイヤーを積み重ねるとよい。

⑥epoch数を複数パターン試してパフォーマンスを評価し、早期終了（early stopping）を試す。

⑦ RMSProp、AdaGrad（学習率を減衰させる）を用いる。

⑧Xavierの初期値を用いることで、出力の分散が小さすぎたり大き過ぎるのを防ぐ。

（おまけ）perplexity



Model samples:

Softmax sample temperature: lower setting will generate more likely predictions, but you'll see more of the same common words again and again. Higher setting will generate less frequent words but you might see more spelling errors.

0.32

the a there the there was there end the tors could startup the of intere the tors to in the the there

in to the tions

in tes a startups is and the the there a tartups the and the the things the and compang the to a star

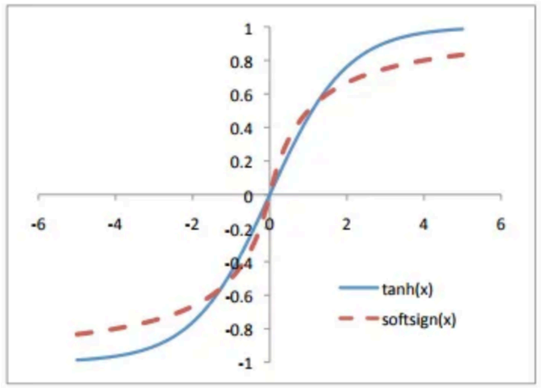
the best a tors to of the there the the things a startups the is investors to the in the is the tors

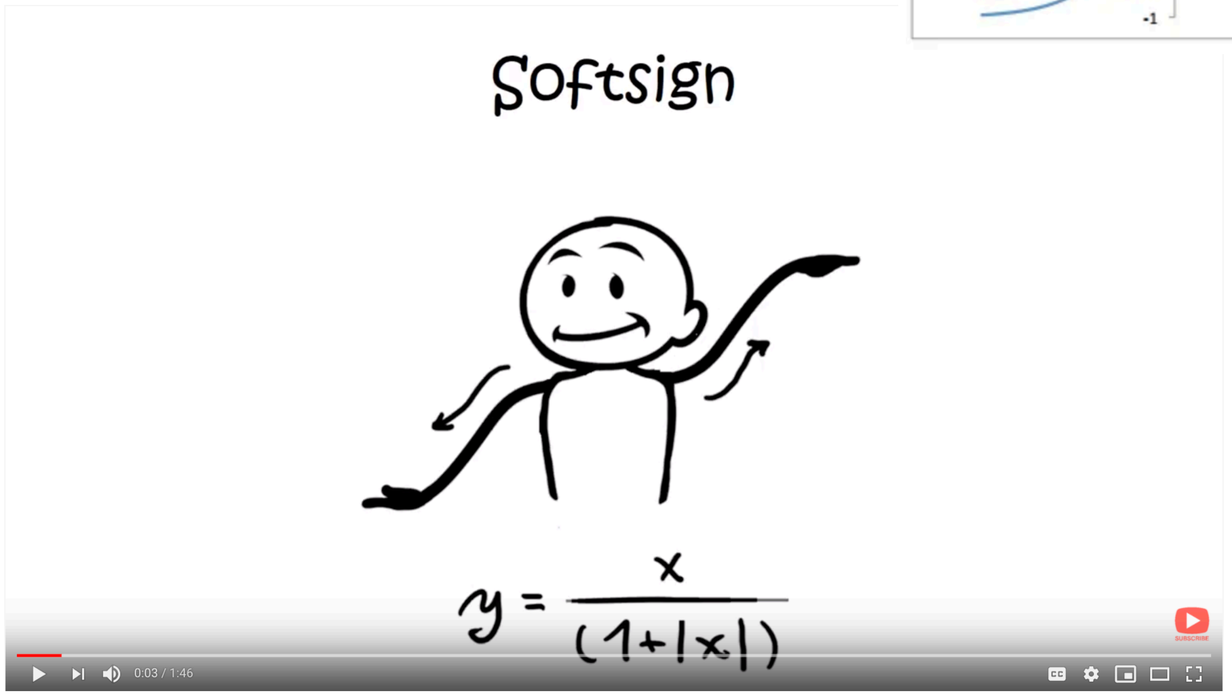
in the of the and and you make and the a tions was the the a startups the the tars a startup the thin

https://cs.stanford.edu/people/karpathy/recurrentjs/

（おまけ）活性化関数

;D 体で覚えよう!





https://www.youtube.com/watch?time_continue=47&v=1Du1XScHCww&feature=emb_logo

もう一つの対策

勾配発散への対策

勾配クリッピング という手法が提案されました。

ニューラルネットワークで使用されるすべてのパラメータの最大値（あるいは閾値）を保持し、

勾配Δwのノルム |Δw| ＞Δmax のとき

$$\Delta w \leftarrow \Delta w \frac{\Delta_{max}}{|\Delta w|}$$

に制限します。（閾値を用いるときはΔmaxをthresholdとする）

勾配の方向は変わらず、大きさだけを変えられます。

これによって、1つのパラメータ更新ステップが大きくならないように調整できます。

発展の話題

RNNおよびLSTMの構造は、今のタイムステップからの予測には、このタイムステップにおける入力のみならず、それ以前のタイムステップからの出力も（隠れ状態やセル状態のパスを通って）合わせて寄与するようなアーキテクチャだといえる。

これらのアーキテクチャの問題点は、シーケンス全体のうち前方のタイムステップの情報のみを用いて予測をするという点である。**後方のタイムステップの情報に現在のタイムステップでの予測に寄与できる重要な情報**があってもそれを用いることができない。

この問題に対処した手法に、Bi-directional RNN（双方向性RNN）がある。これは単方向ではなく双方向のタイムステップからの情報を予測に寄与させるアーキテクチャである。