リカレントニューラルネットワーク RNN

- リカレントニューラルネットワークのデモ

```
# for recurrent neural networks
cd ~/study/2018karapthy_recurrentjs.git
open character_demo.html
```

リカレントニューラルネットワーク（RNN）とは， 時間的な変化や順序といった系列情報を扱うニューラルネットワークモデルです。 このため音声認識，自然言語処理，ロボットの生成制御などに用いられています。 時々刻々変化するデータを扱うには，それまでに処理されたデータの系列を文脈として 保持しておく必要があります。 リカレントニューラルネットワークを拡張した長短期記憶モデル(LSTM: Long short-term memory)を 用いる場合が多いです。

応用事例としては，最近マイクロソフトが開発した女子高生人工知能「りんな」の対話生成アルゴリズムが有名です。 LINEであたかも女子高生と会話しているかのようなコミュニケーションができることで話題になりました。

そのほかにも自動翻訳，画像と文章と相互変換（画像を入力するとその画像を説明する文章を生成する，逆にある文章を与えると対応する画像を生成する）などがあります。 リカレントニューラルネットワークを用いた技術は他の従来手法の性能を上回り，現時点での最高性能と認められています。

現段階で，人間が書いたものなのかコンピュータが書いた文章なのか区別がつかない場合すらあります。 新たなチューリングテスト（あるいはチューリングチャレンジ， コンピュータの生成した文か人間が書いた文かが見分けがつかなければ， もはやコンピュータに知性が宿っていると言っても良いだろうという考え方）と言ったりもします。

- 実習1 シーケンシャル kmnist
- 実習1 足し算 RNN
-

- 勾配消失問題， 勾配爆発問題:
- 勾配チェック， 勾配クリップ:
- 言語モデル:
- 埋め込みモデルによる意味表現:
- 潜在意味解析， トピックモデル (TDIDF， LSI， LSA， topic model):
- 注意(seq2seq， transformer):
- 機械翻訳， sentence vector (skip-thought， ELMo， BERT， Big bird, Tough-to-beat):
- 各種評価指標 sacre BLUE， negative log likelihood， perprexity:
- 画像脚注付け (NIC)， pix2pix， img2txt， VQA:
- ニューラルチューリングマシン:

リカレントニューラルネットワークの特徴

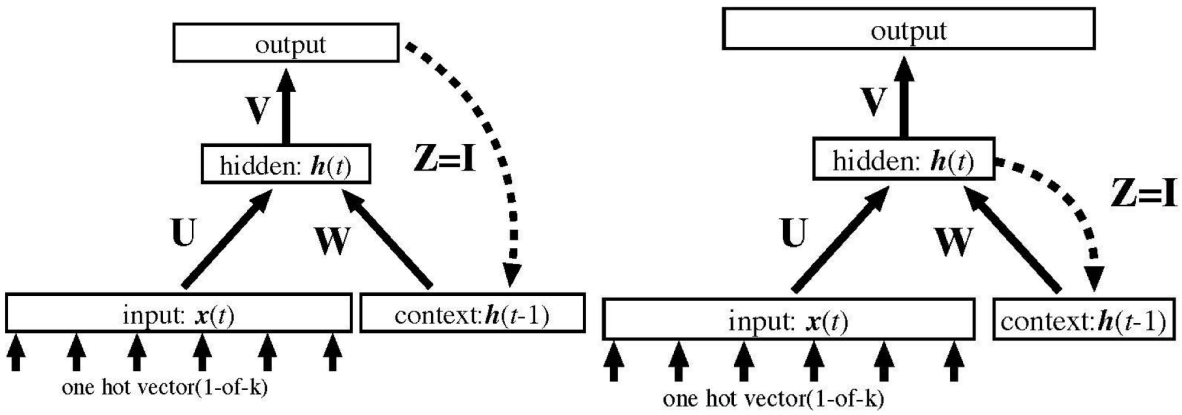
- 過去の状態を多様に保持する中間層
- 中間層更新は非線形
- 深層化（多層中間層）
- 駄菓子菓子，数多くの黒魔法が存在
  - 勾配クリップ，勾配正規化，勾配チェック，忘却バイアス，正規化，正則化，IRNN

--

最近の成果

1. 手書き文字認識(Graves et al., 2009)
2. 音声認識(Graves & Jaitly, 2014; Graves, Mohamed, & Hinton, 2013)
3. 手書き文字生成(Graves, 2013)
4. 系列学習(Sutskever, Vinyals, & Le, 2014)
5. 機械翻訳(Bahdanau, Cho, & Bengio, 2015; Luong, Sutskever, Le, Vinyals, & Zaremba, 2015)
6. 画像脚注付け(Kiros, Salakhutdinov, & Zemel, 2014; Vinyals, Toshev, Bengio, & Erhan, 2015)
7. 構文解析(Vinyals et al., 2015)
8. プログラムコード生成(Zaremba & Sutskever, 2015)

古典的リカレントニューラルネットワーク



ミコロフ革命

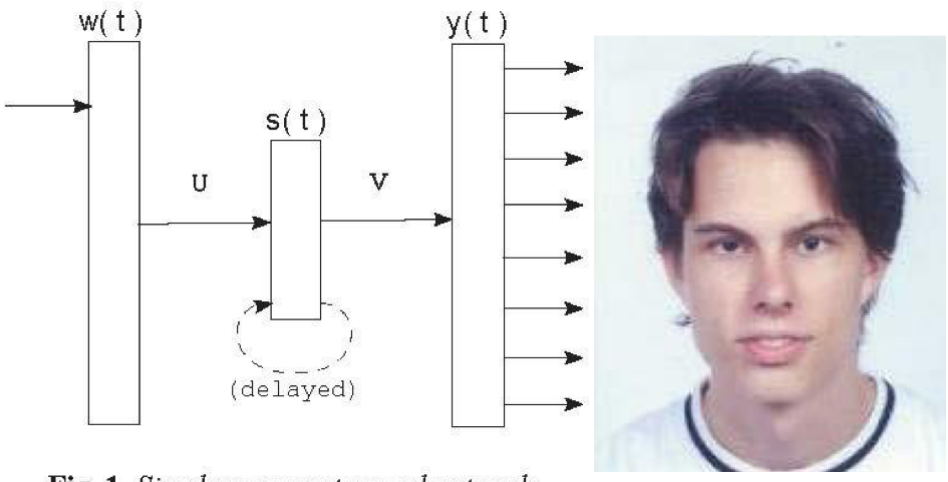


Fig. 1. *Simple recurrent neural network.*

$$
\begin{aligned}
s(t) &= f(\mathbf{U}\mathbf{w}(t) + \mathbf{W}\mathbf{s}(t-1)) \\
y(t) &= g(\mathbf{V}\mathbf{s}(t-1)) \\
U(t+1) &= U(t) + \alpha \mathbf{w}(t)\mathbf{e}_h(t)^{\top}
\end{aligned}
$$

リカレントニューラルネットワークの仲間

- アトラクターネットワーク
- ホップフィールドネットワーク
- エコーステートネットワーク
- ボルツマンマシン（制限付きではない方）

系列情報処理におけるターゲット

- 機械学習アルゴリズムを系列情報へ適用 := 入力系列を(別領域でもよい)出力系列へ変換
  - 例: 音圧の時系列変動を文字系列へ
- 連続系列から離散系列へ変換(逆も)
  - 出力系列は一時刻前の入力信号
  - 画像処理における画素情報の変換に比べれば時間的制約があるので入出力関係は明確
  - すなわち時系列情報処理では入出力関係に自然な制約が存在.
- 次項予測 := 時間遅延情報の教師あり学習，および，教師なし学習の混合分布
  - 教師あり学習だが，明確な教師信号が不要

記憶なし系列情報処理モデル

- 自己回帰モデル AR
  - 次項 := 固定時間窓による時間遅延予測
- フィードフォワードニューラルネットワーク
  - 多層パーセプトロンは AR の一般化とみなすこともできる (Bengio 言語モデル)

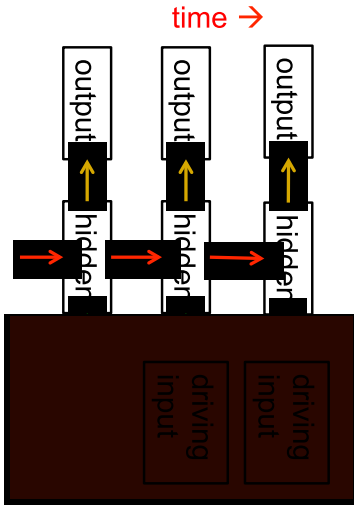
記憶あり系列情報処理モデル

- 隠れ状態つき生成モデル，隠れ状態つき内部力学系モデル → 記述可能性向上
  1. 内部状態 := 長期記憶
  2. 変動因子(雑音) 内部力学系モデル := 出力状態も変動。正確な内部状態予測が困難
  3. これが内部状態の確率分布の推論モデル作成のモチベーション

- 内部状態が 2 だけつの場合，解析可能

線形力学系モデル(工学者がお好み)

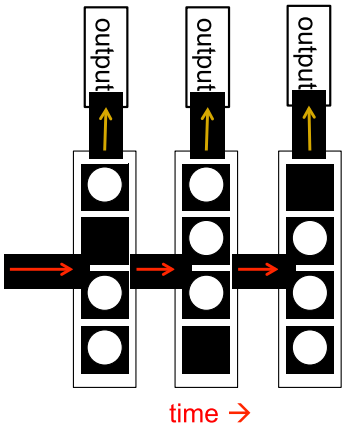
- 直接的には観測不能な内部状態を評価する生成モデル
  - 内部状態 := ガウス雑音つき線形力学系モデル。観測変数 := ガウス雑音つき線形モデル
  - 外部入力項 (driving input) つきモデルもあり.
- 次項予測 := 内部状態を推論 (ミサイル迎撃ミサイルなど)
  - ガウス分布の線形変換もまたガウス分布(正規分布の再生性)



線形力学モデル

隠れマルコフモデル (コンピュータサイエンティストがお好み)

- HMM は N 個の離散的内部状態の中から，ある時刻には 1 つの状態を持つ。状態遷移は状態遷移行列によって確率的に定まる。出力も確率的
  - 隠れ状態から生成される出力は決定論的に定まらない → 隠れ状態を仮定
  - N 個の隠れ状態が確率分布を仮定
- 次時刻の出力を予測 := 隠れ状態の分布を推論する
  - HMMs を解くためのアルゴリズムが提案されている



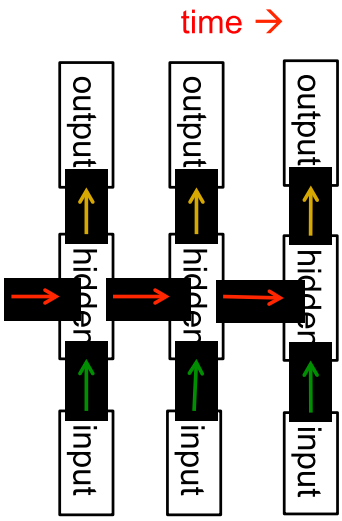
HMM

HMM の制限

- HMM からデータが生成される場合:
  - 各時刻で一つの隠れ状態が選ばれる。すなわち N 個の隠れ状態数では，たかだか $\log(N)$ の過去の状態しか符号化できない
- 2 人の対話状態を考えてみると，2 人が全く同じ情報の符号化を行っていると仮定しても，語彙情報，意味情報，イントネーション，アクセント，発話速度，音量，音域特性を制御可能であったとしても，100 ビットの情報で会話する場合 2^{100} ビットの情報が伝達されなければならない

リカレントニューラルネットワーク RNN

- RNN s are very powerful, because they combine two properties:
  - 隠れ状態の分散表現 := 過去の情報を効率よく保存，表現，処理可能。統計的自然言語処理モデルにおける N-グラムモデルが N=4 程度で飽和してしまうことを考えれば良い
  - 隠れ層の非線形性 := 複雑な状態更新を可能にする
- 十分な数のニューロンを用いれば複雑な現象をモデル化可能



リカレントニューラルネットワーク

生成モデルは確率的か？

- 線形力学系，HMM は確率モデル
  - ある時刻までの隠れ状態が与えられた時の条件付き事後分布は決定論的関数となる
- リカレントニューラルネットワークは決定論的
  - 線形力学系，あるいは HMM の隠れ状態の分布が決定論的に定まっている場合の等価な決定論的確率分布に従うリカレントニューラルネットワークモデルを考える

リカレントニューラルネットワークの挙動

- 振動 運動制御
  - 点アトラクタ 記憶検索
  - カオス的振る舞い 情報処理には不向き？
  - RNN は複数の同時進行して相互作用する要因から生成される知識を獲得，系列情報を実行可能
- 反面，計算コストが高い，収束に時間が要する。数多の黒魔法が存在する
  - 長年，実用可能な応用例を示すことができなかった

BPTT, LSTM

How does LSTM work?

1. LSTM はロジスティックユニットやハイパータンジェントでアナログ値を保持している隠れ層のメモリユニットを置き換えたモデル
2. 各メモリセルは入出力ゲートを制御
3. メモリセルに保存されているアナログ値には忘却ゲートが付いている
4. 入出力ゲートが閉じていてで忘却ゲートが開いていれば，次の時刻も状態を保持する Le, Jaitly, & Hinton (2015)

BPTT, LSTM 解題

単純再帰型ニューラルネットワークSRN

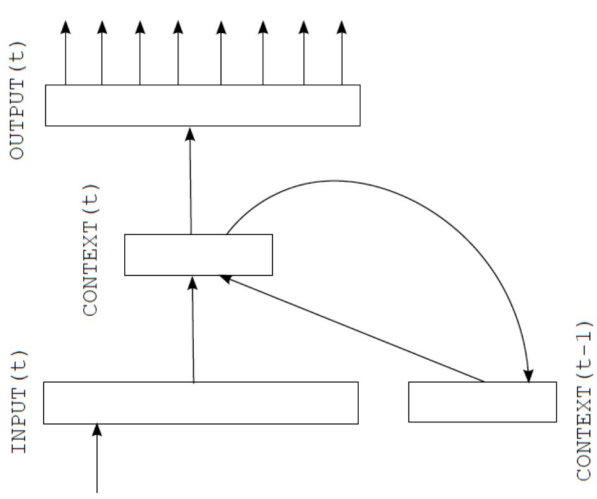


Figure 1: *Simple recurrent neural network.*

Mikolov (2010) Fig. 1

単純再帰型ニューラルネットワークSRN

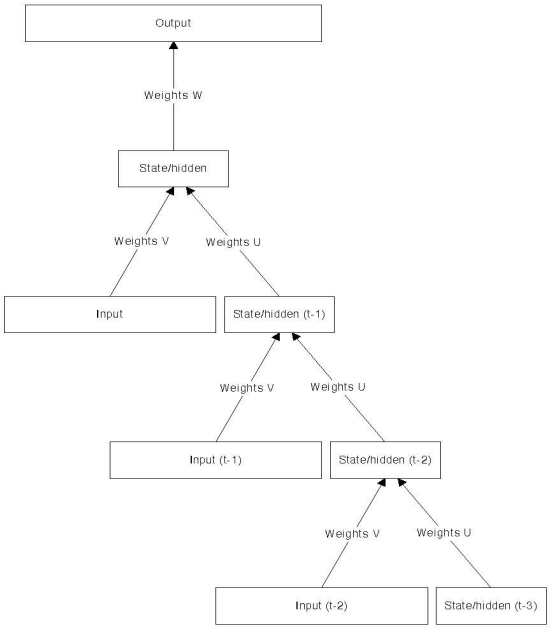
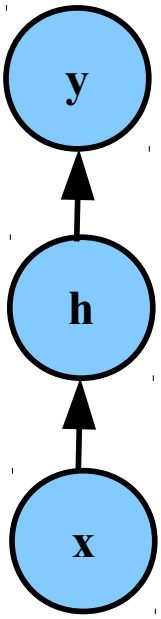


Figure 5: The effect of unfolding a network for BPTT ($\tau = 3$).

Booden (2001) Fig. 5

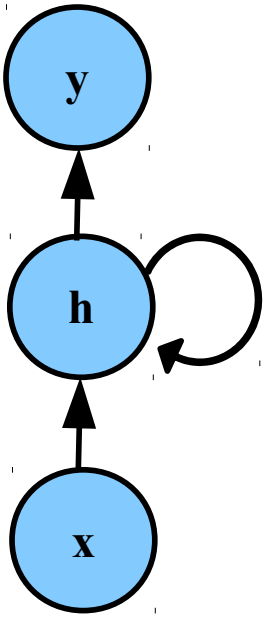
表記と基本グラフ



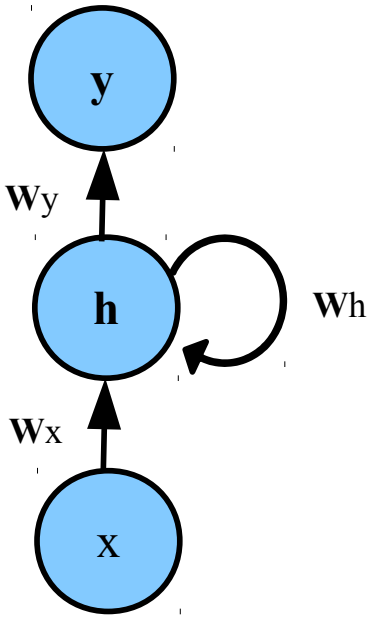
y: 出力層ニューロン

h: 中間層ニューロン

x: 入力層ニューロン



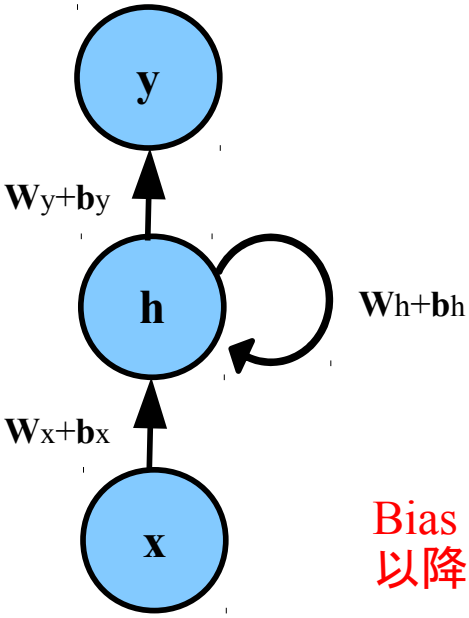
再帰結合（recurrent connections)



$\mathbf{W}_y$:結合係数行列(中間から出力)

$\mathbf{W}_h$:結合係数行列(再帰結合)

$\mathbf{W}_x$:結合係数行列(入力から中間)

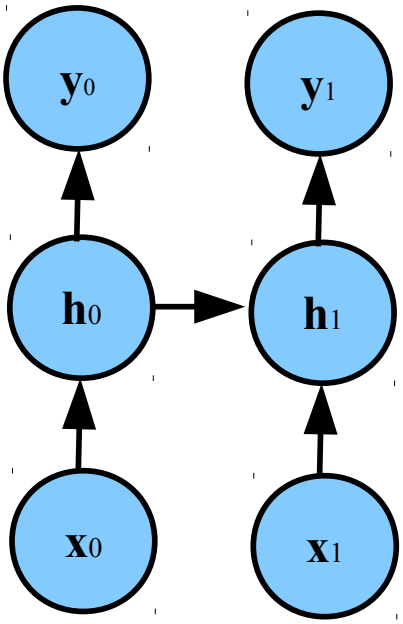


$\mathbf{b}_y$:バイアス(中間から出力)

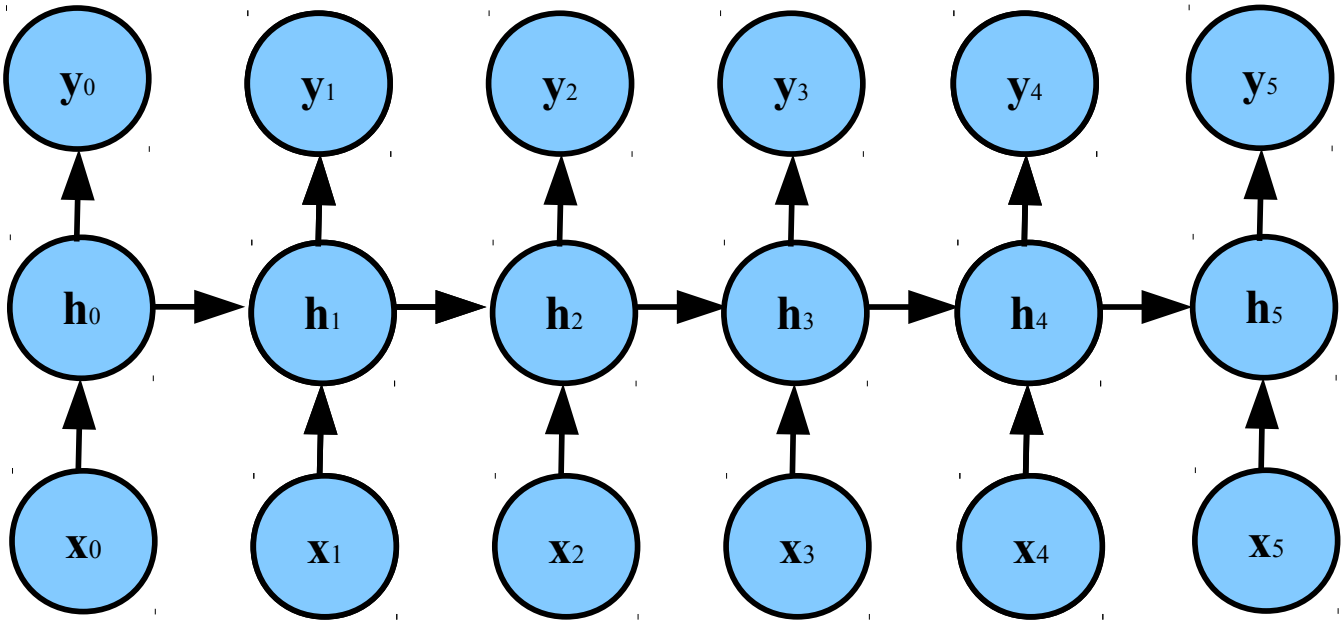
$\mathbf{b}_h$:バイアス(再帰結合)

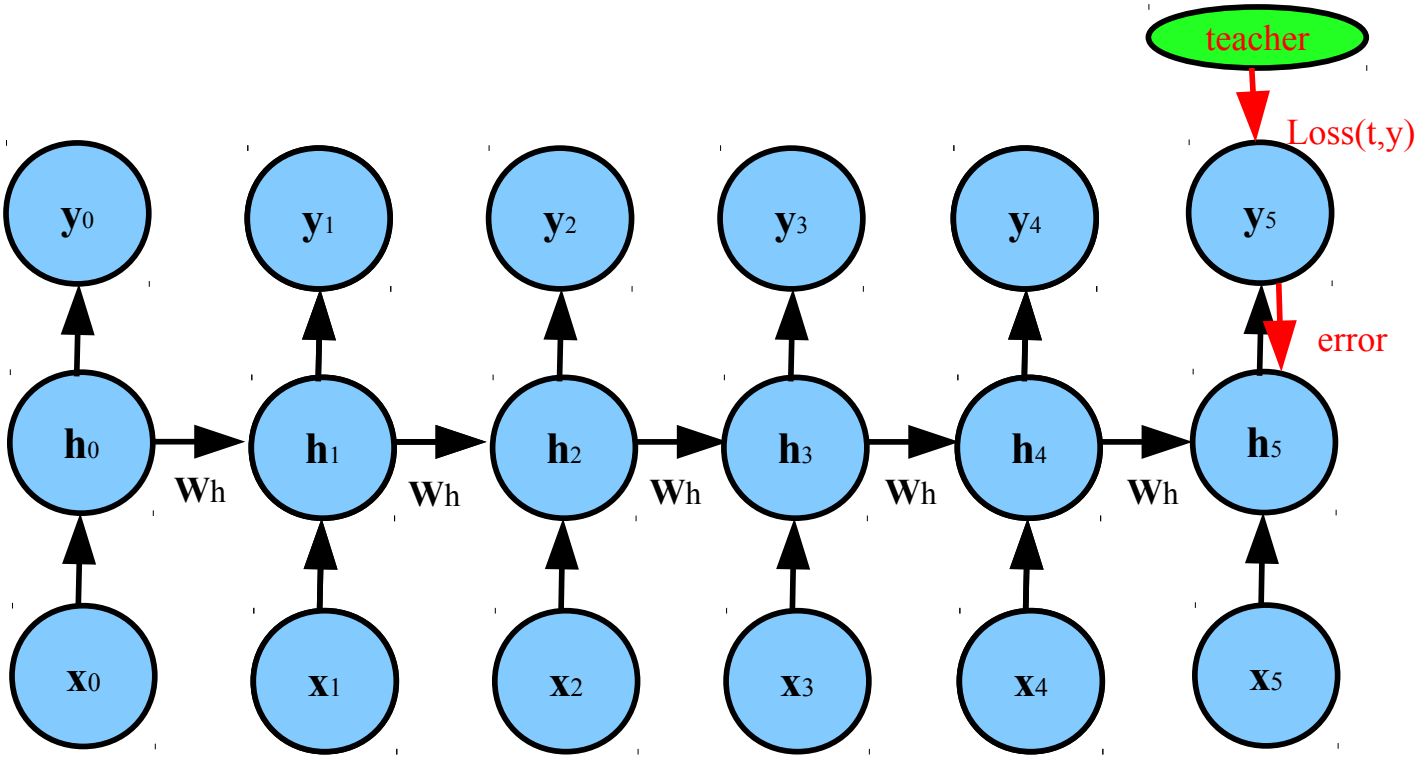
$\mathbf{b}_x$:バイアス(入力から中間)

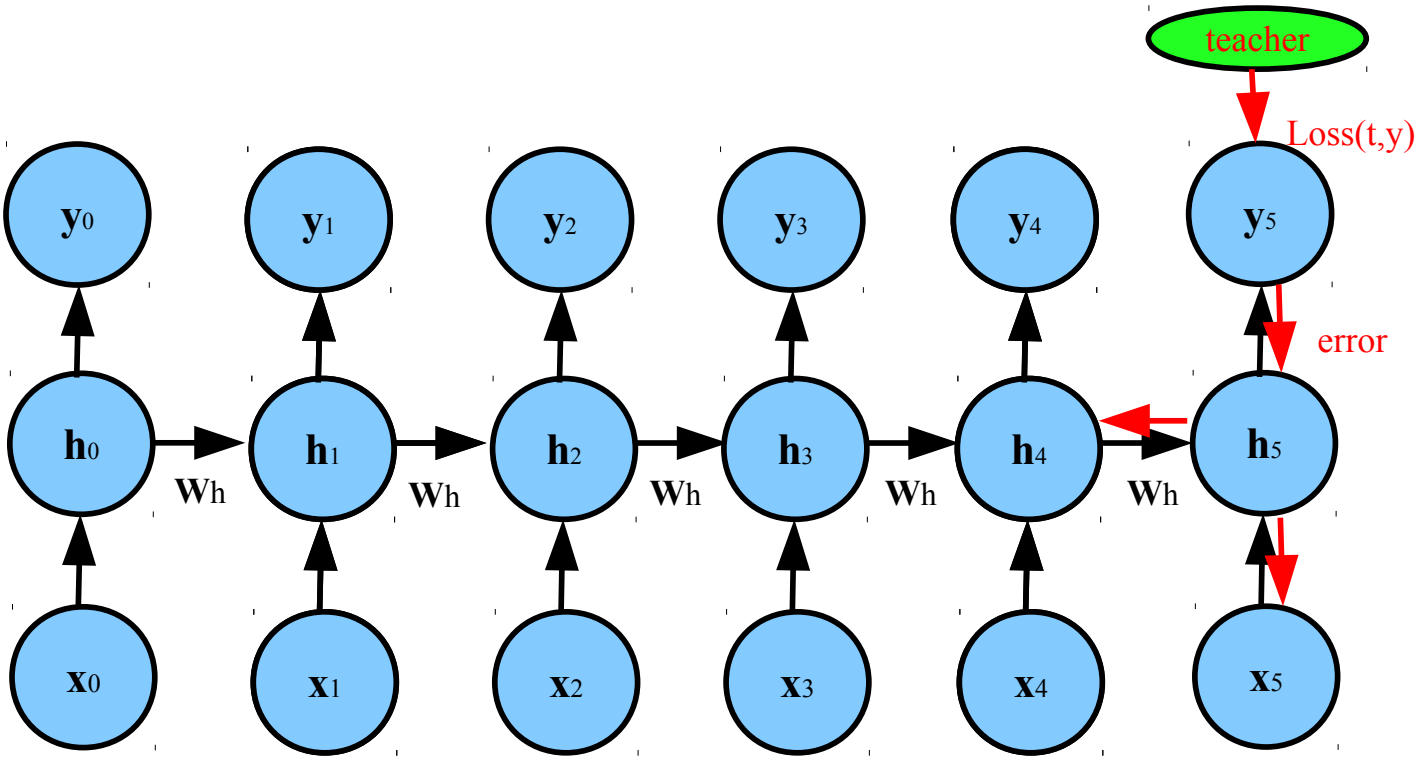
Bias terms will be omitted, henceforth
以降バイアス項は省略

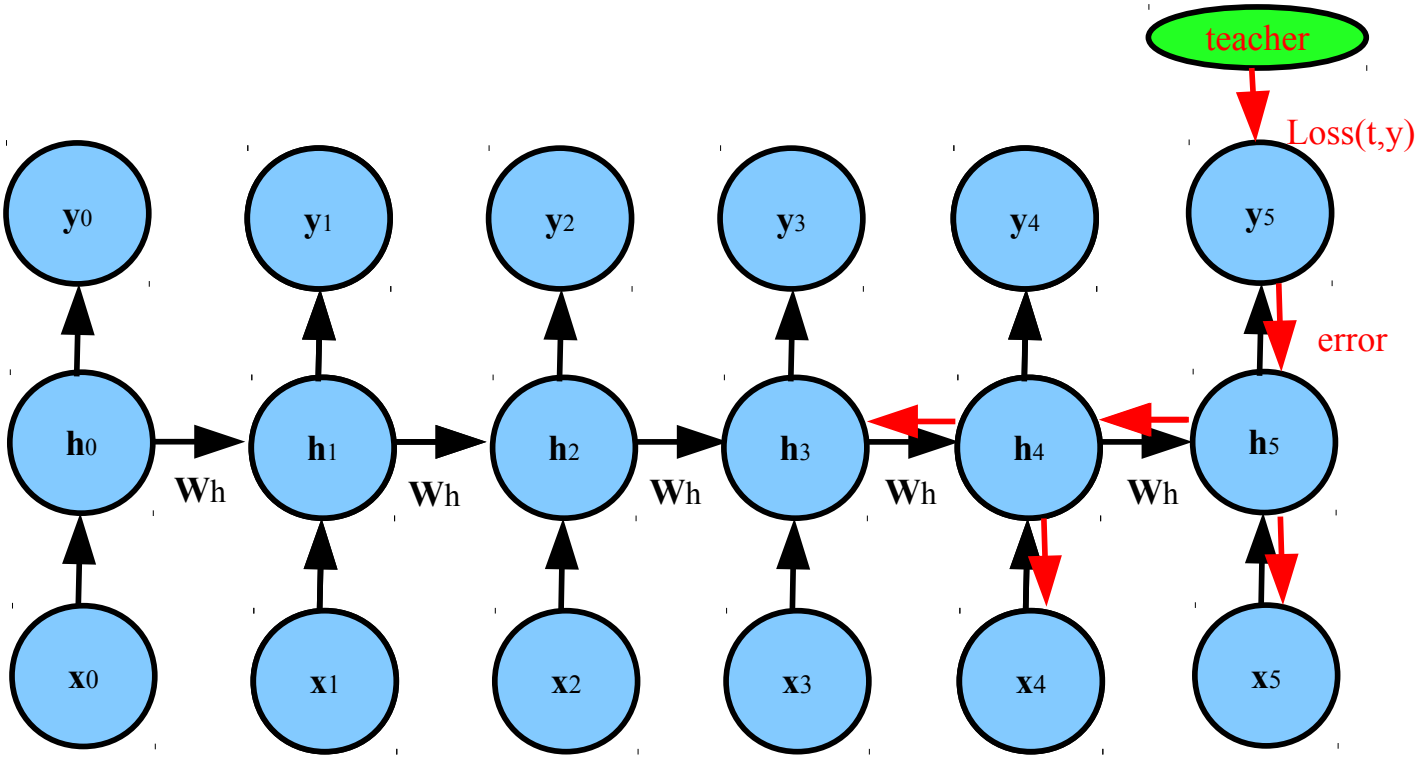


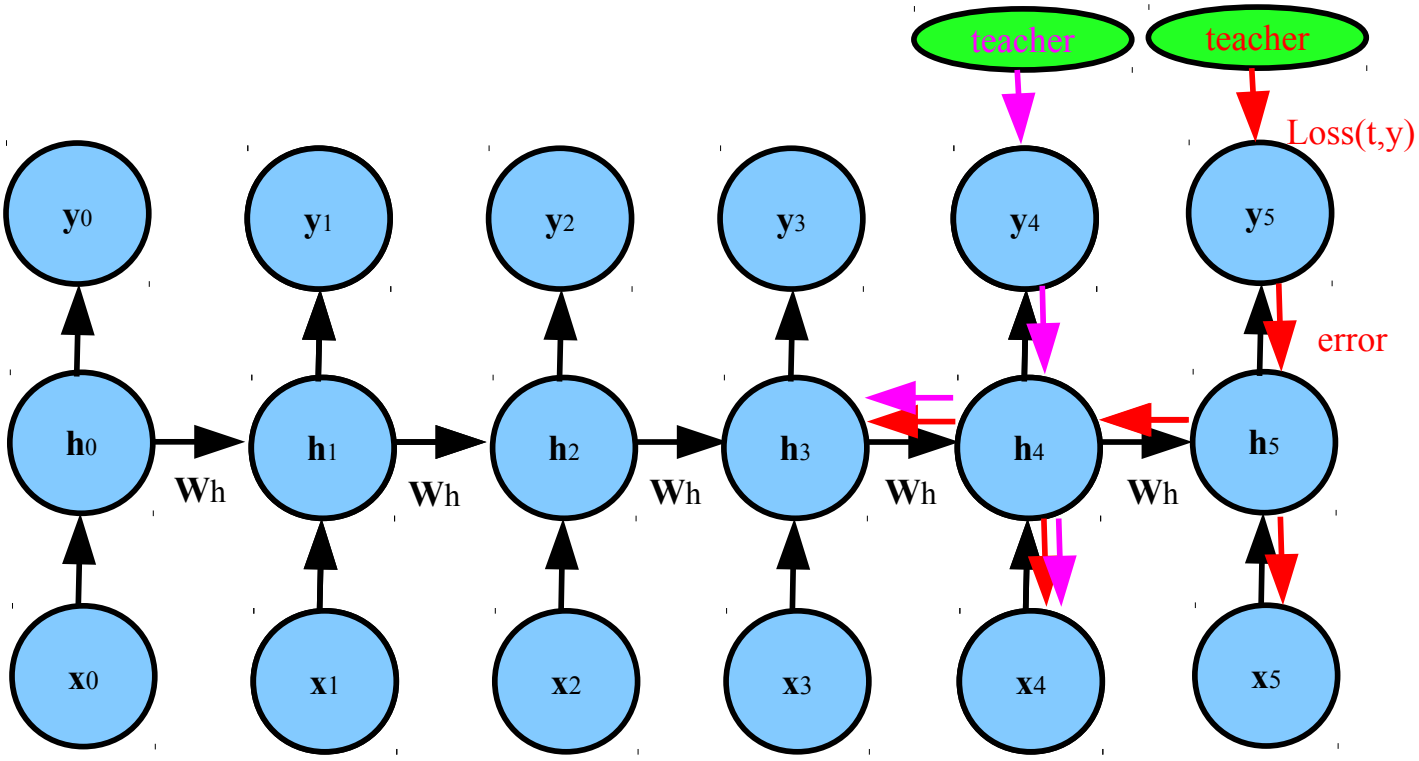
Digits subscripted indicate time
$\tau := 0 \ldots$
下付き添字は時刻を表す。
カッコで表記する流儀もある
(e.g. x(t))

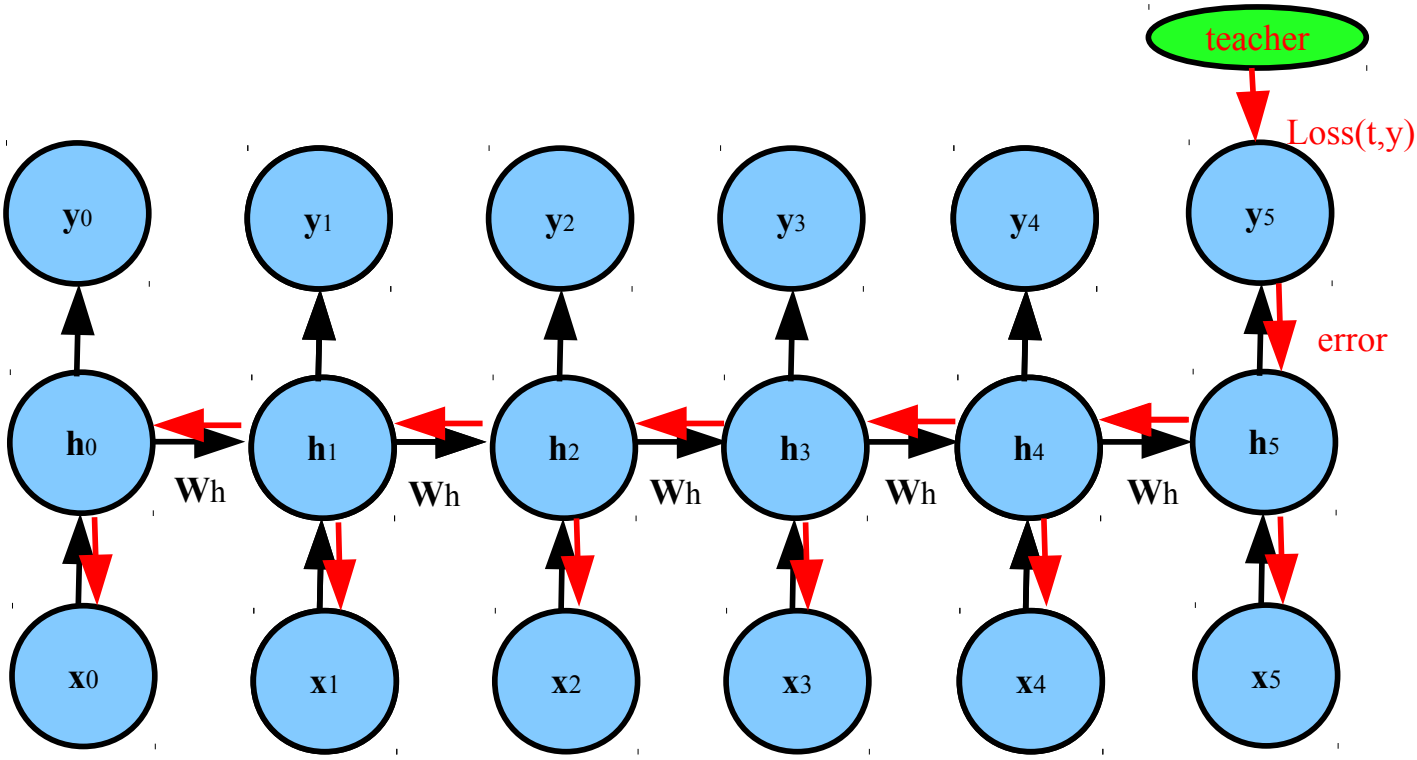



teacher
Loss(t,y)
y0
y1
y2
y3
y4
y5
error
h0
h1
h2
h3
h4
h5
Wh
Wh
Wh
Wh
Wh
x0
x1
x2
x3
x4
x5

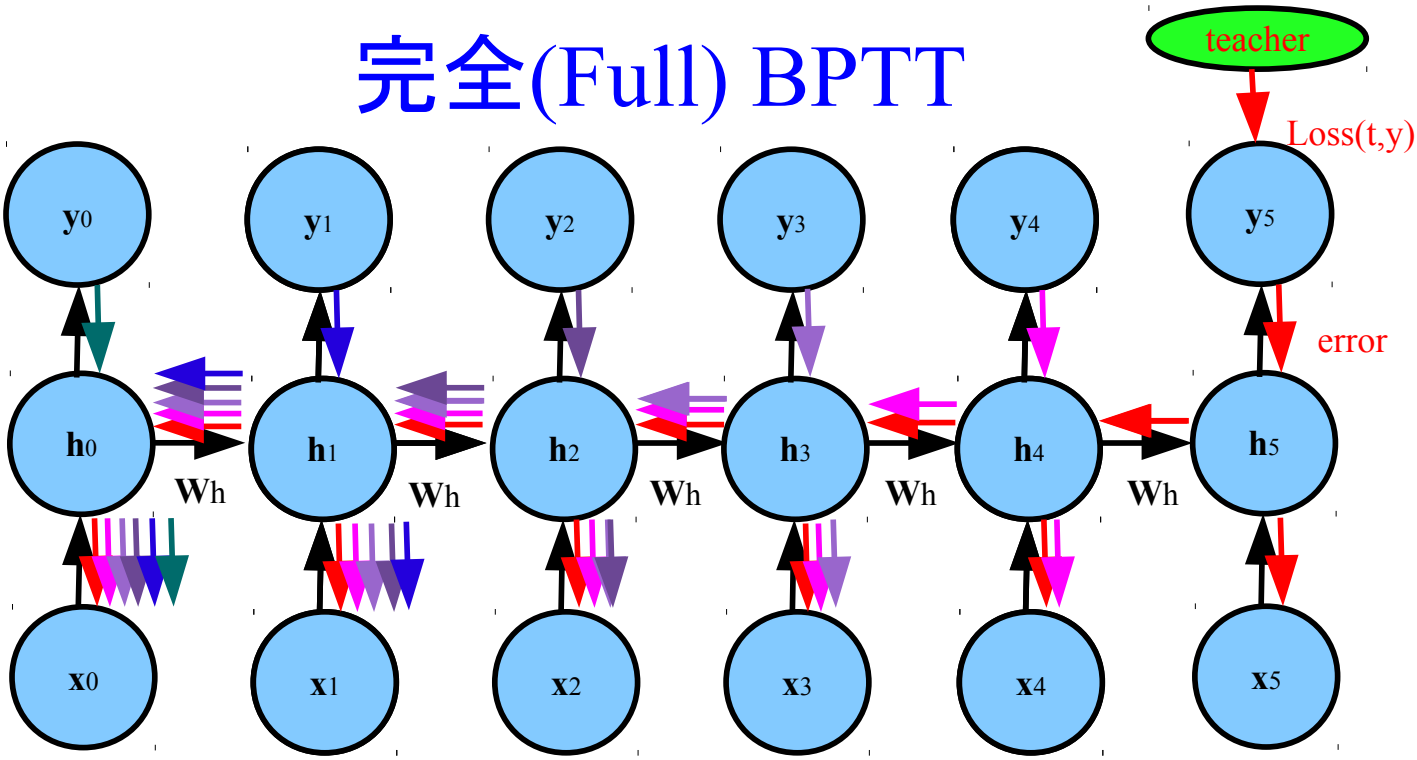

teacher
Loss(t,y)
y0
y1
y2
y3
y4
y5
error
h0
h1
h2
h3
h4
h5
Wh
Wh
Wh
Wh
Wh
x0
x1
x2
x3
x4
x5

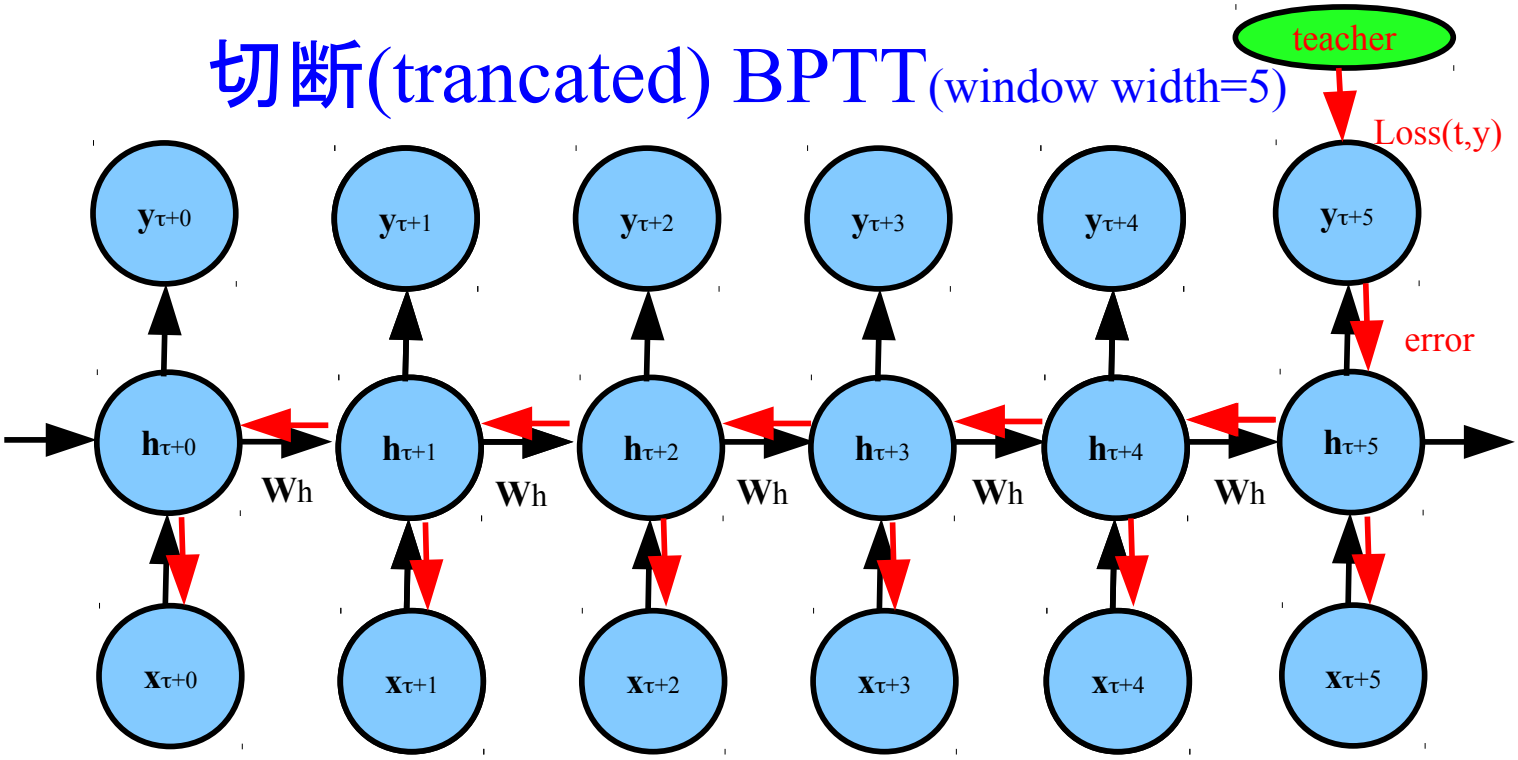

teacher
Loss(t,y)
y0
y1
y2
y3
y4
y5
error
h0
h1
h2
h3
h4
h5
Wh
Wh
Wh
Wh
Wh
x0
x1
x2
x3
x4
x5


teacher
teacher
Loss(t,y)
y0
y1
y2
y3
y4
y5
error
h0
h1
h2
h3
h4
h5
Wh
Wh
Wh
Wh
Wh
x0
x1
x2
x3
x4
x5


teacher
Loss(t,y)
y0
y1
y2
y3
y4
y5
error
h0
h1
h2
h3
h4
h5
Wh
Wh
Wh
Wh
Wh
x0
x1
x2
x3
x4
x5

完全(Full) BPTT

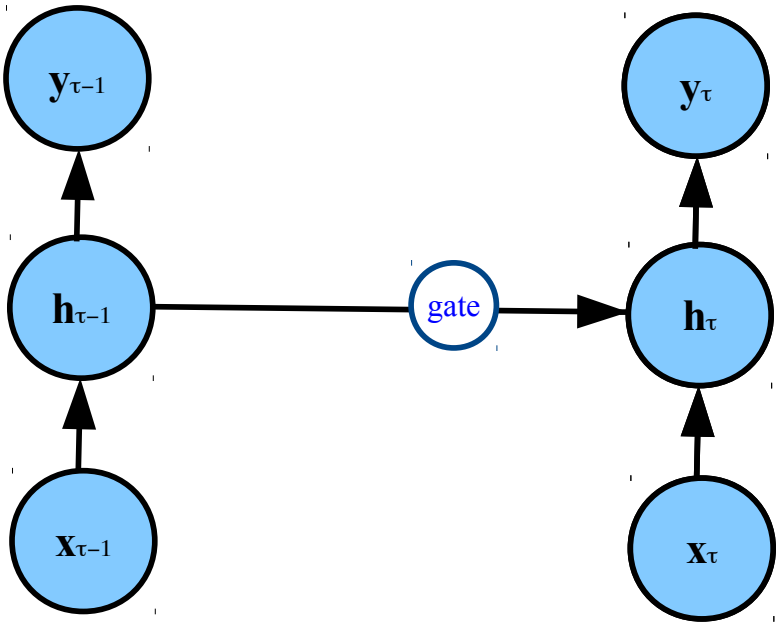

teacher
Loss(t,y)
y0
y1
y2
y3
y4
y5
error
h0
h1
h2
h3
h4
h5
Wh
Wh
Wh
Wh
Wh
x0
x1
x2
x3
x4
x5



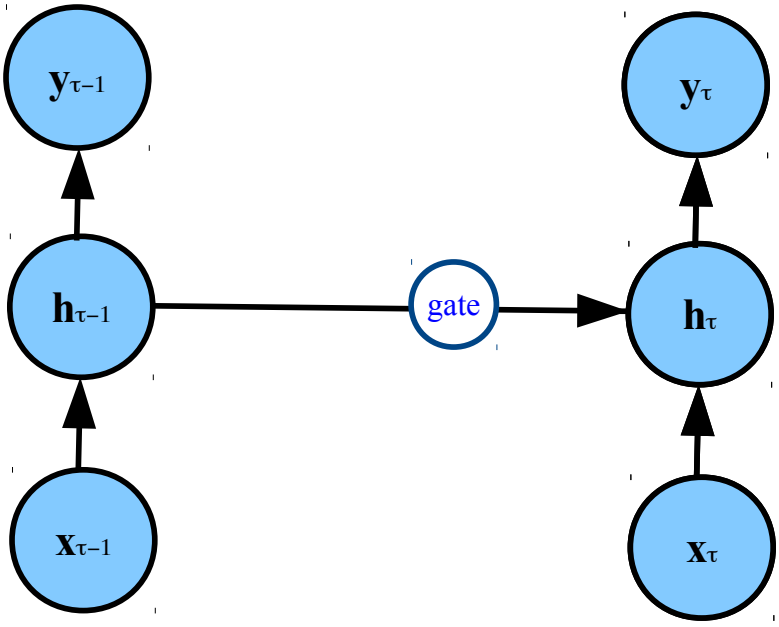
改良可能？

Can we improve?

ゲートの導入 introducing gates to control hidden state

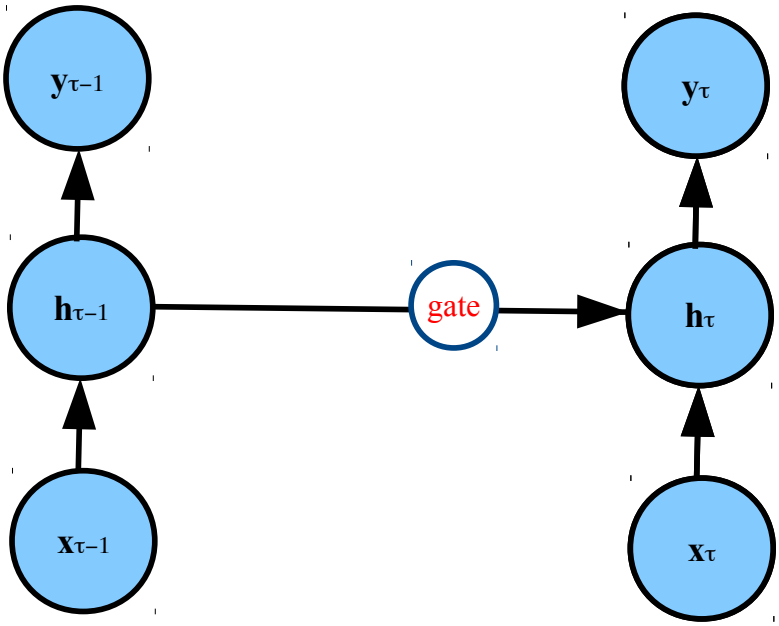


ゲートの導入 introducing gates to control hidden state



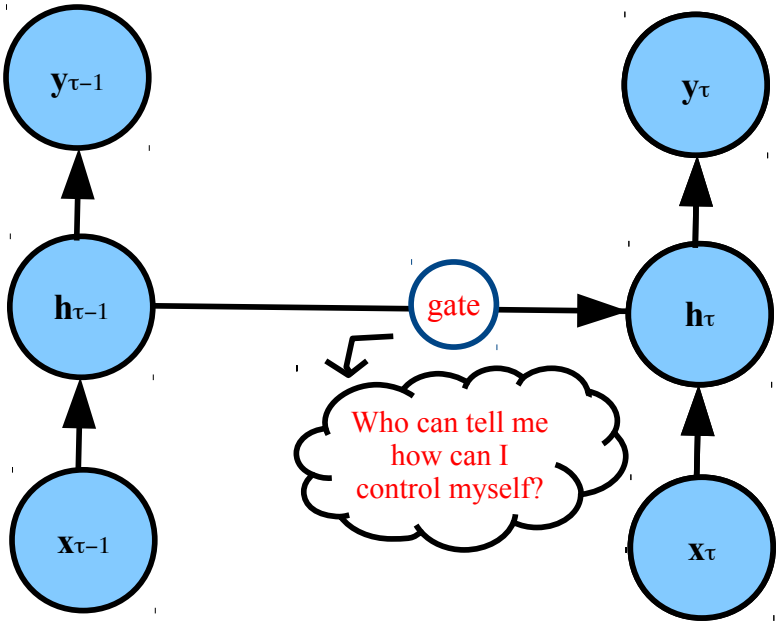
でも,
なぜゲート?
Why gates?

忘却ゲートの導入



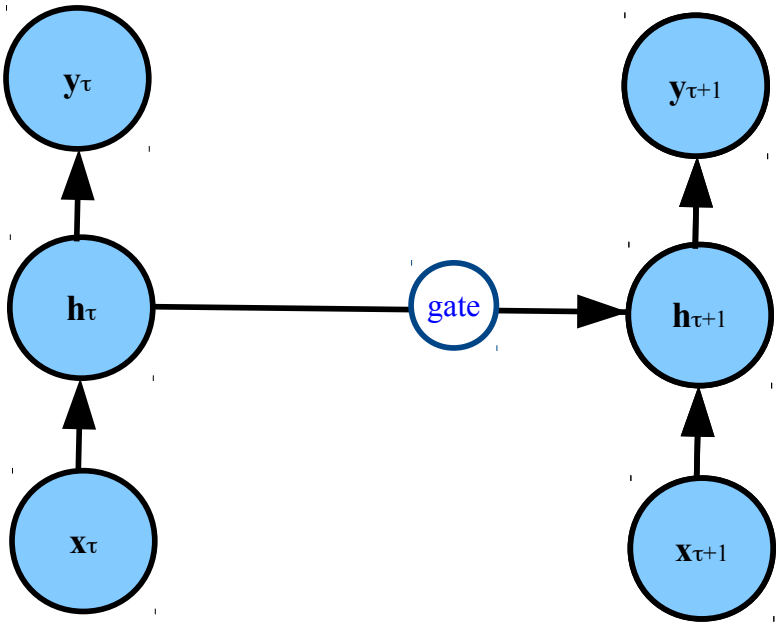
Who can control gates?
誰がどうやってゲート制御？

忘却ゲートの導入



Who can control gates?
誰がどうやってゲート制御？

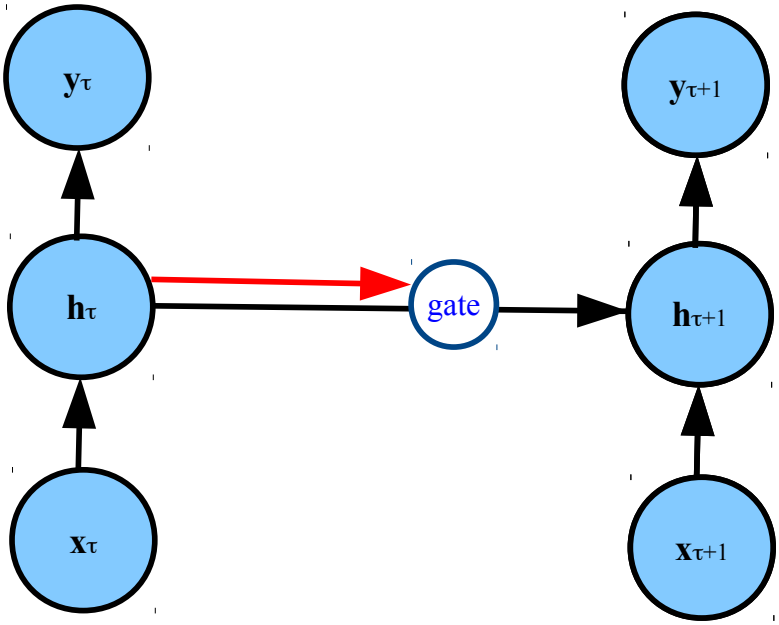
忘却ゲートの導入



who can control gates?
誰がどうやって
ゲートを制御？

３つ候補

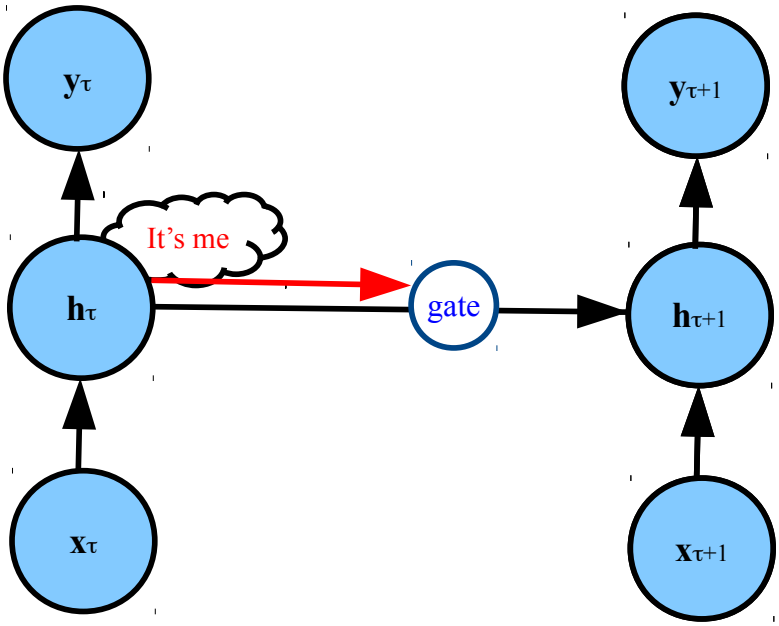
忘却ゲートの導入



who can control gates?
誰がどうやって
ゲートを制御？

３つ候補
1. hτ

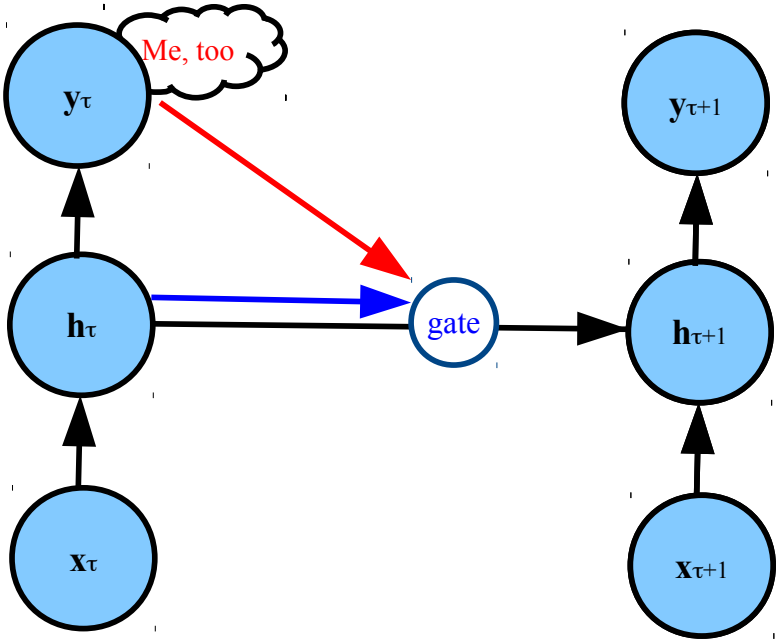
忘却ゲートの導入



who can control gates?
誰がどうやって
ゲートを制御？

３つ候補
1. hτ

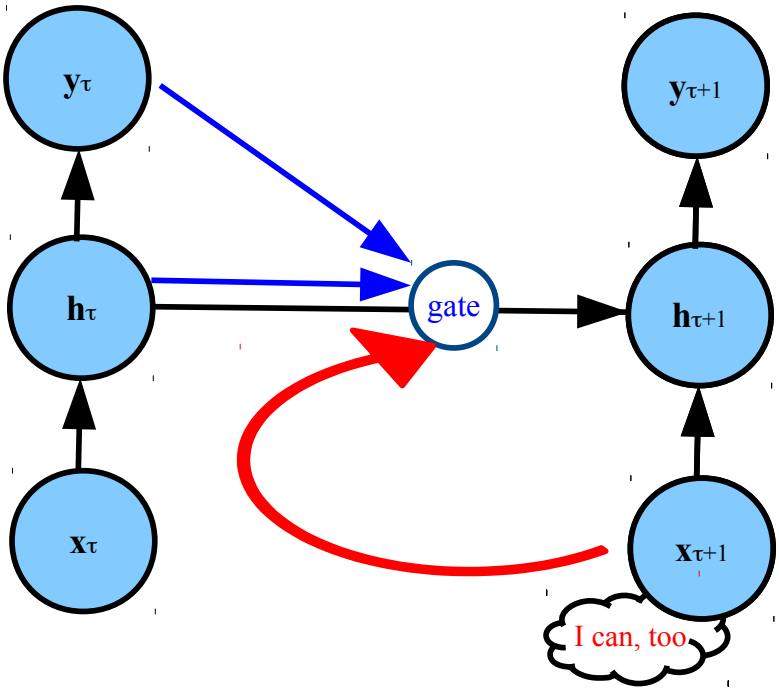
忘却ゲートの導入



who can control gates?
誰がどうやって
ゲートを制御？

３つ候補
1. hτ
2. yτ

忘却ゲートの導入

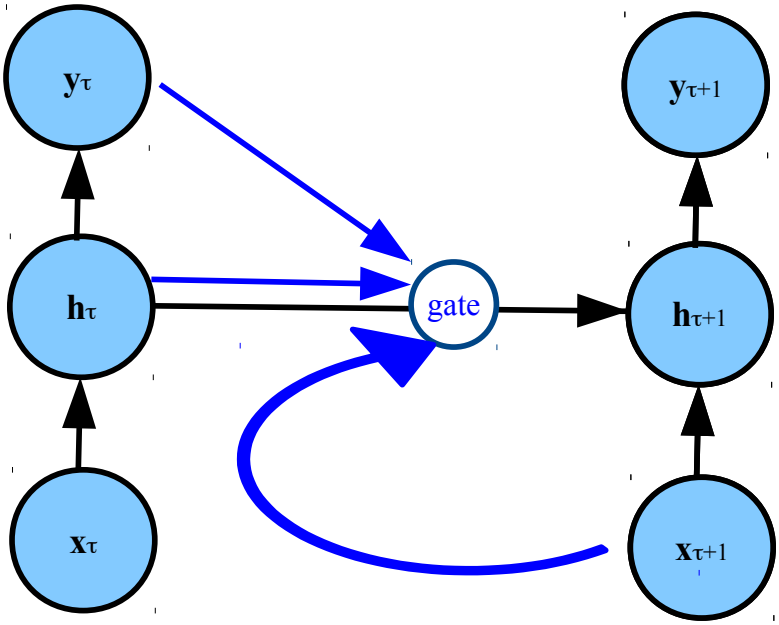


who can control gates?
誰がどうやって
ゲートを制御？

３つ候補

1. hτ
2. yτ
3. xτ+1

忘却ゲートの導入

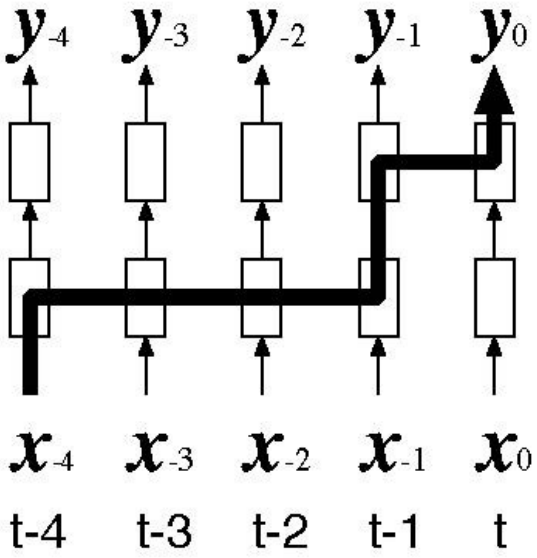


1. hτ
2. yτ　　ゲート制御
3. xτ+1

$h_{\tau+1} = h_\tau \ \sigma(x)$

- $\sigma(x) = (1+e^{-x})^{-1}$
- $x = \mathbf{W}_f (y_\tau + h_\tau + x_{\tau+1})$

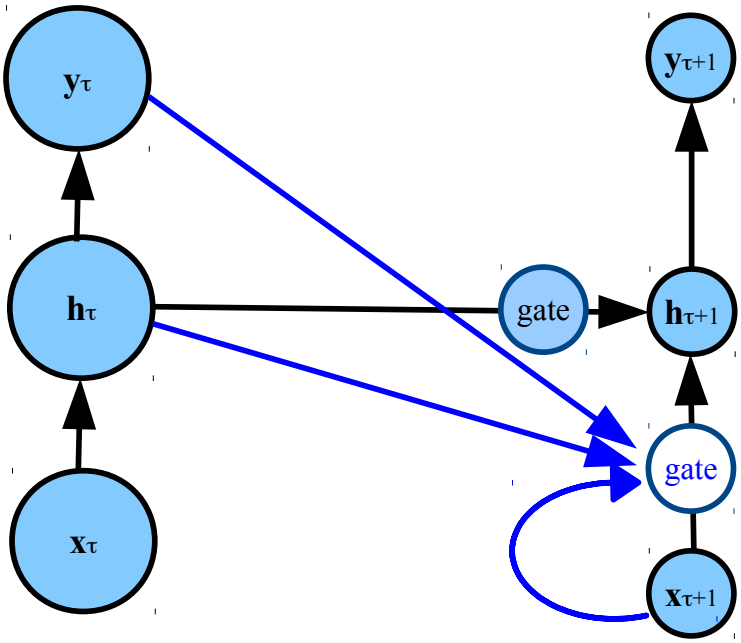
ゲートによって長距離依存LTDを解消可能



もっと改良可能？

Can we improve more?

入力ゲートの導入

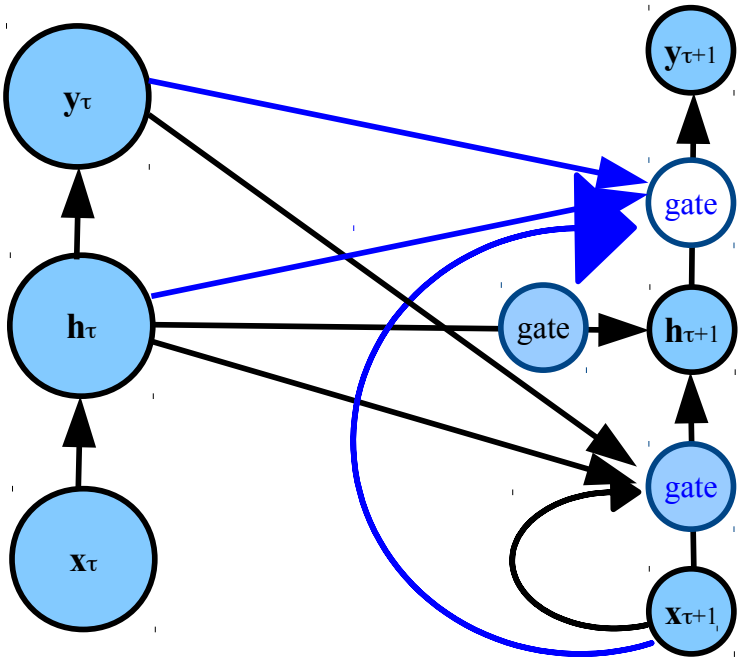


$$h_{\tau+1} = h_\tau\ \sigma(\mathbf{w}(h_\tau + x_{\tau+1}))$$

- $\sigma(x) = (1+e^{-x})^{-1}$
- $x = y_\tau + h_\tau + x_{\tau+1}$

もっともっと可能？
You need more?

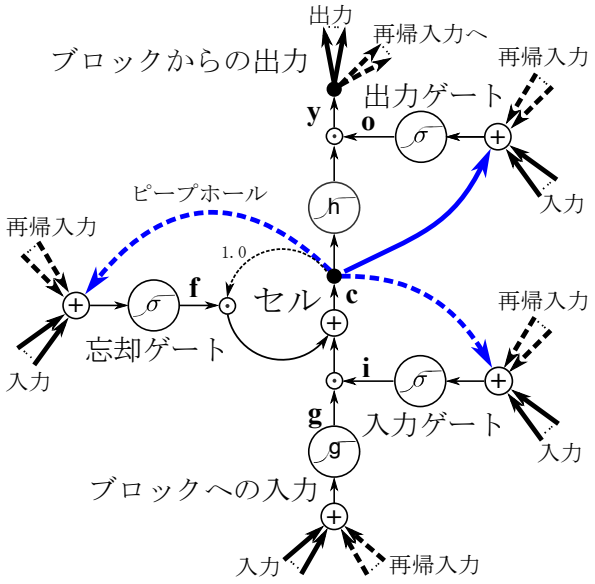
出力ゲートの導入



$$h_{\tau+1} = h_\tau \ \sigma\left(\mathbf{W}(\mathbf{h}_\tau + \mathbf{x}_{\tau+1} + \mathbf{y}_{\tau+1})\right)$$

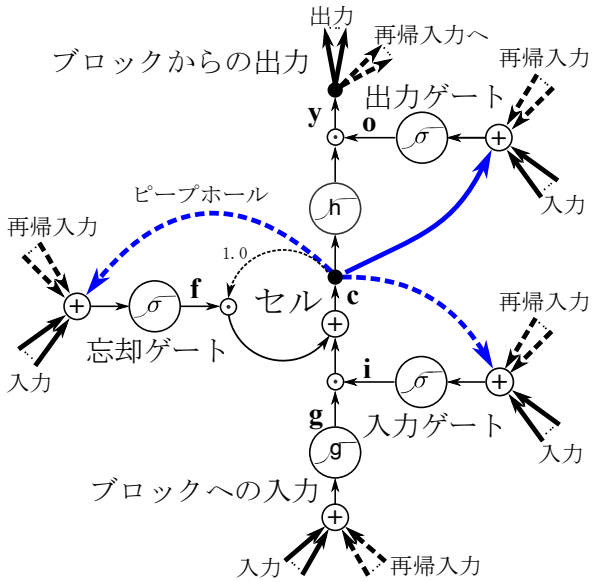
- $\sigma(x) = (1+e^{-x})^{-1}$
- $x = y_\tau + h_\tau + x_{\tau+1}$

LSTM



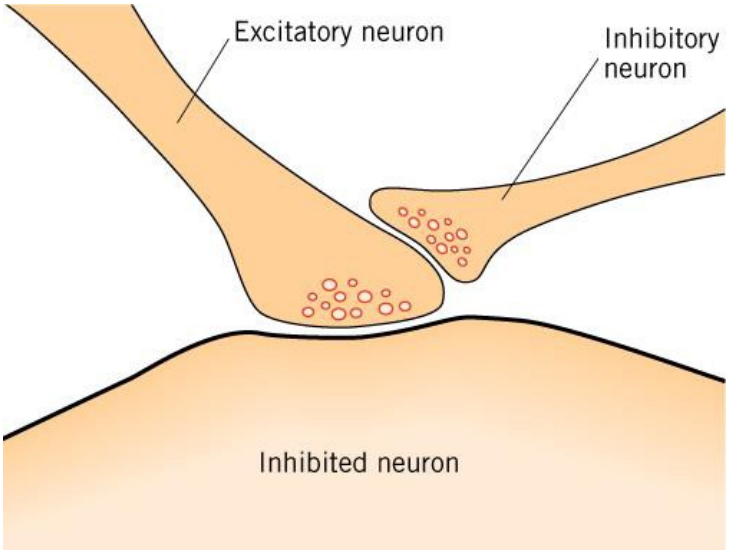
- 入力ゲート $i = \sigma(\ )$
- 忘却ゲート $f = \sigma(\ )$
- 出力ゲート $o = \sigma(\ )$
- 入力全体 $g = \phi(\ )$
- セル $c = f @ c + i @ g$
- 出力 $y = o @ \phi(\ )$

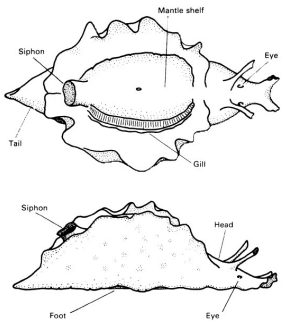
LSTM

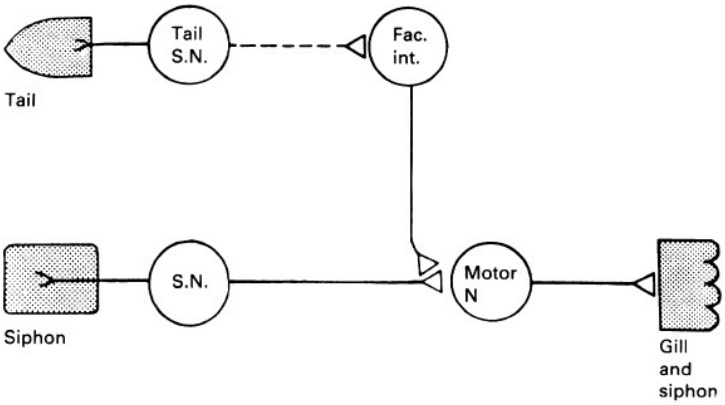


$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + b_i\right), \\
f_t &= \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right), \\
o_t &= \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + b_o\right), \\
g_t &= \phi\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right), \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t, \\
y_t &= o_t \odot \phi\left(c_t\right)
\end{aligned}
$$

LSTMの生理学的対応物

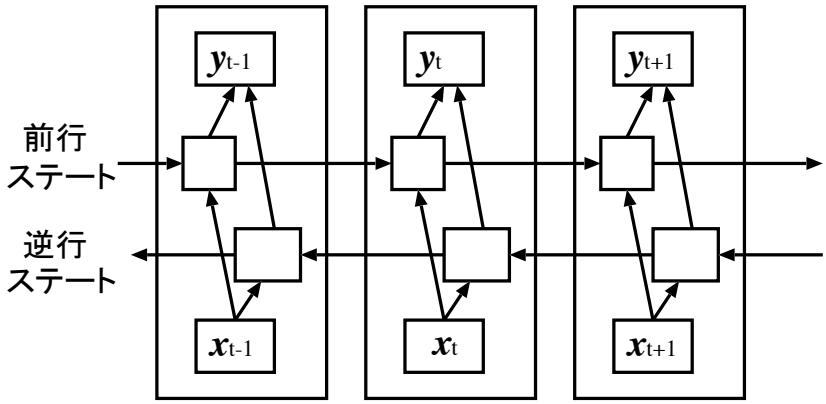






http://kybele.psych.cornell.edu/~edelman/Psych-2140/week-2-2.html

双方向 RNN



word2vec

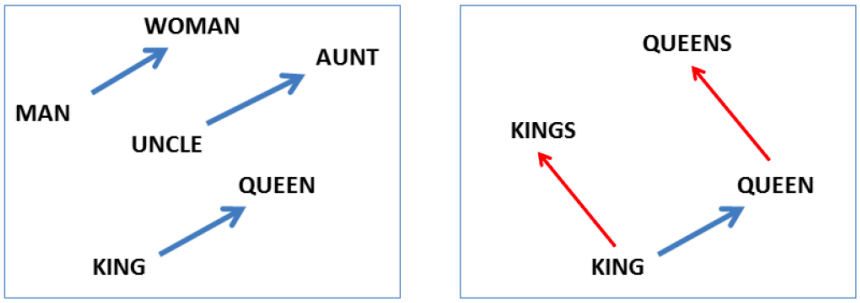
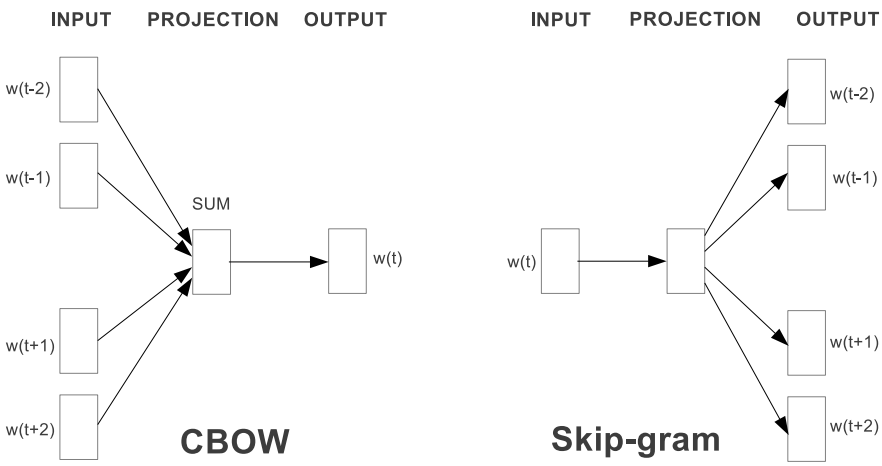
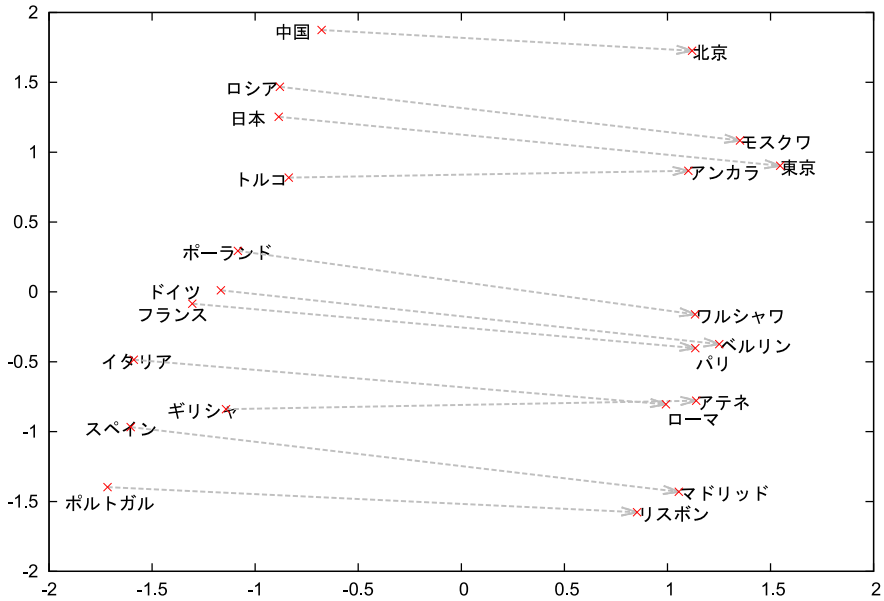


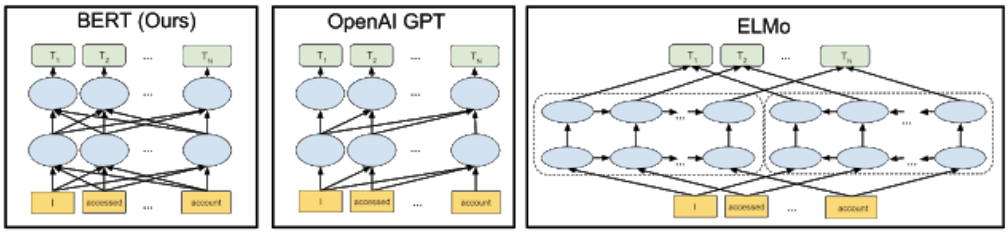
図 6: 左図：3 単語対の性差を表す関係。右図：単数形と複数形の関係。各単語は高次元空間に埋め込まれている



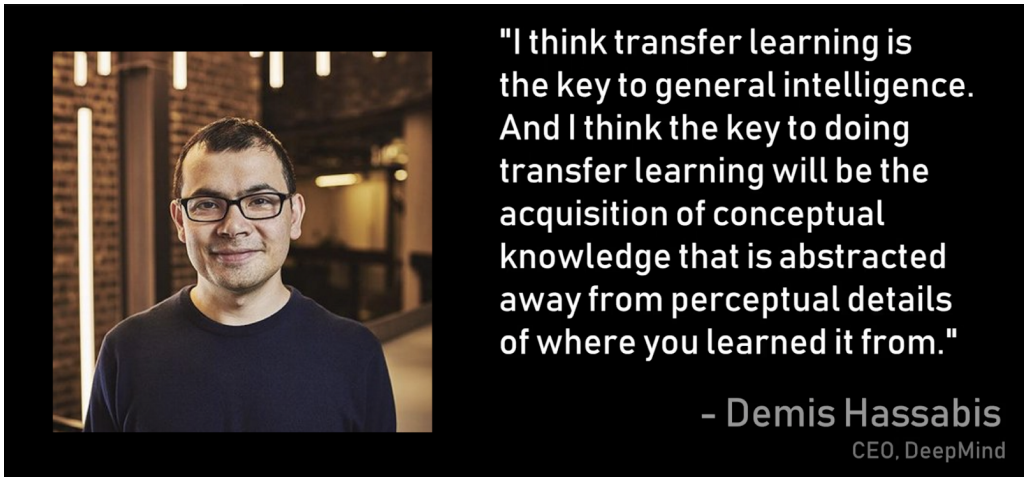


- word2vec の実習

センテンス埋め込みモデル



- 2018年は自然言語処理にとってのイメージネットブレークの時であったと言われたりします。すなわち人間超えしました性能で話題となりました。
- また最近炎上した GPT-2 に関する open AI のブログ も参照してください
- BERT 論文
- BERT の colaboratory デモ
- これらのモデルは一つのモデルで複数の課題に転移学習可能であることを示しました。



https://towardsdatascience.com/transfer-learning-946518f95666