ニコニコAIスクール「脳型人工知能開発者入門コース」#9

# 再帰的ニューラルネットワーク (1)

松森 匠哉

2018 / 03 / 11

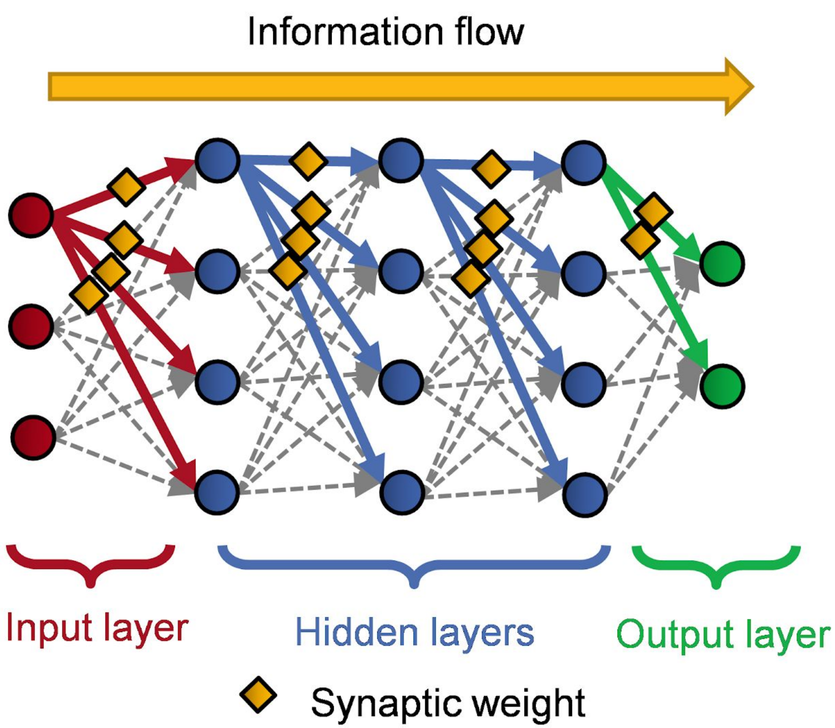
# 用語集

緑: 既知　青: 未習
として講義を進めます

- BP: Backpropagation
  - 誤差逆伝播法
- FFNN: Feedforward Neural Networks
  - 順方向にしか伝播しないネットワークの総称
- FBNN: Feedback Neural Networks
  - フィードバック構造を持つネットワークの総称
- RNN: Recurrent Neural Networks
  - FBNNの総称
- 線形回帰: Linear Regression
  - データ点に適合する関数を発見すること

# 今までのネットワーク

階層型ネットワーク

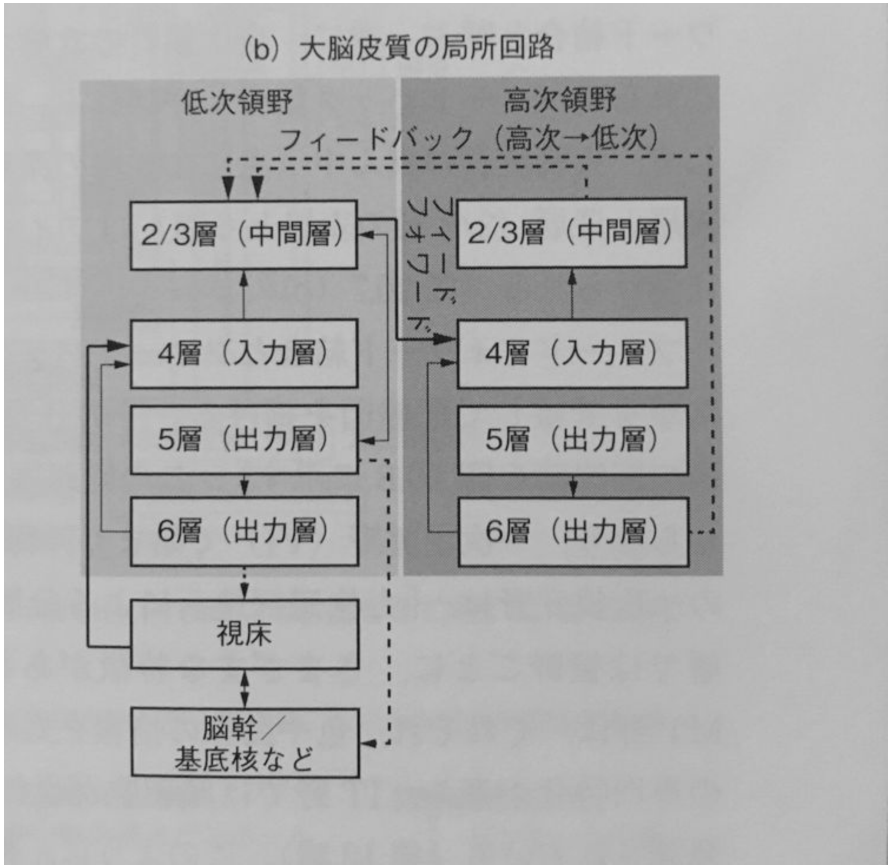



## 階層型NWの特徴

- 複数のニューロンがグループ化され、それぞれ層を形成している
- 各層は直列に連なっており、直前の層のニューロンのみから入力を受ける
- 情報は順方向にしか伝播しない

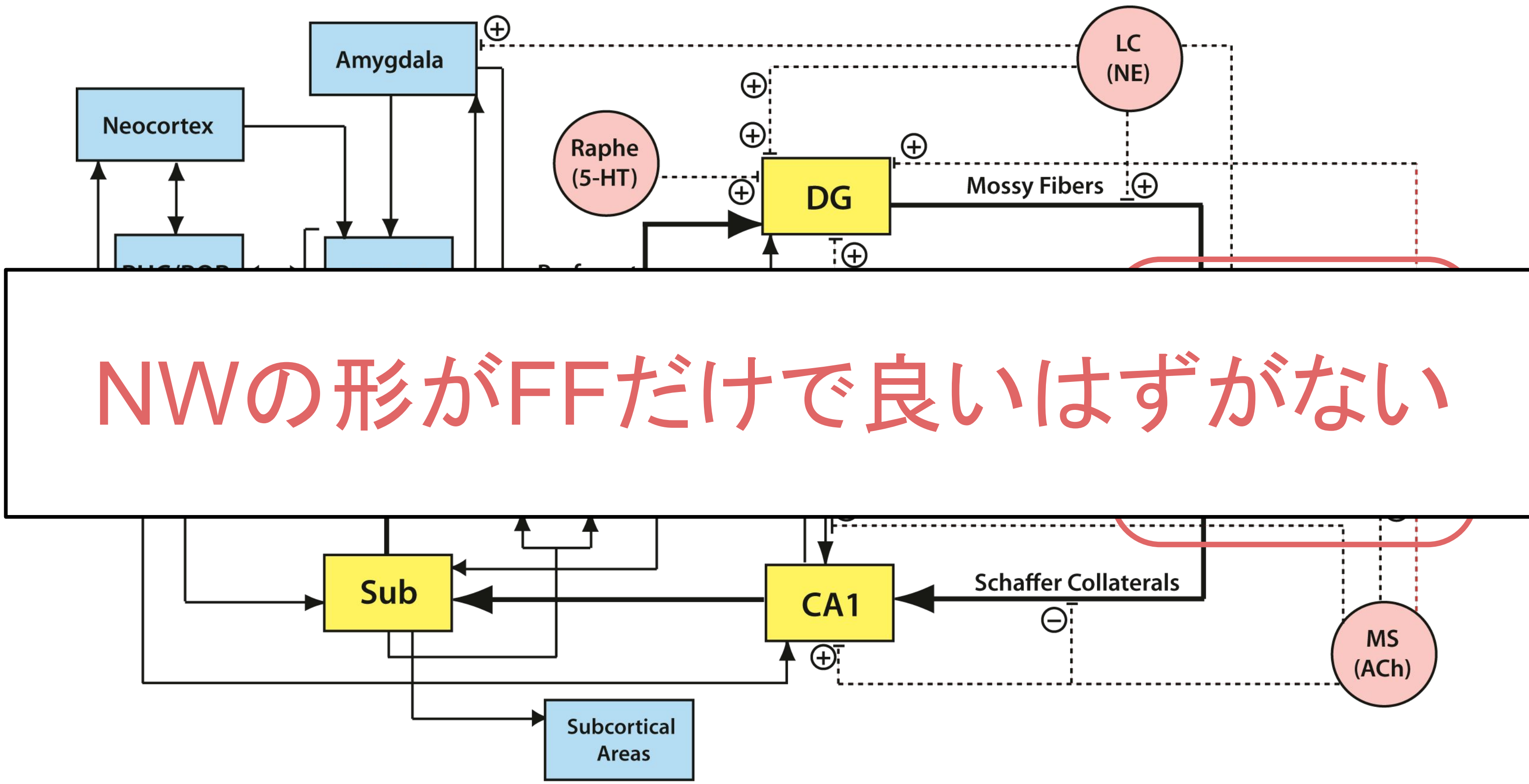
# FF vs FB

解剖学的には FF << FB



# FF vs FB

海馬のCA3に存在する再帰的神経投射

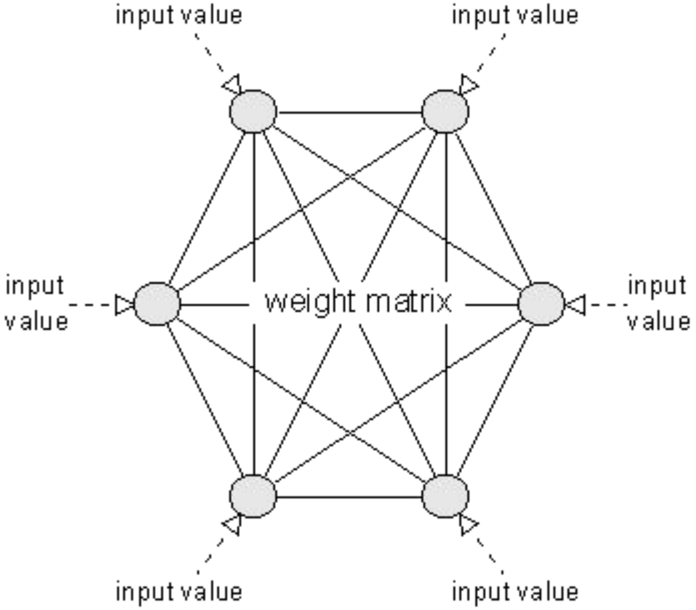


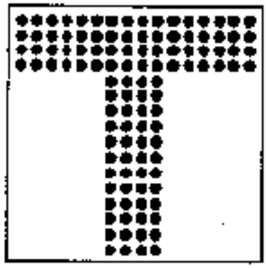

NWの形がFFだけで良いはずがない

# 相互結合のモデル化

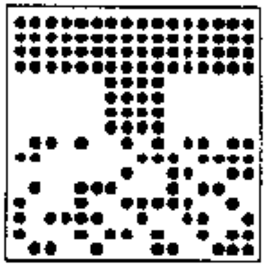
Hopfield Network

- 物理学者 J.J Hopfield (1982) が提唱
- 部分的情報からネットワークに記憶された情報を再構成することができる(連想記憶モデル)



Original 'T'

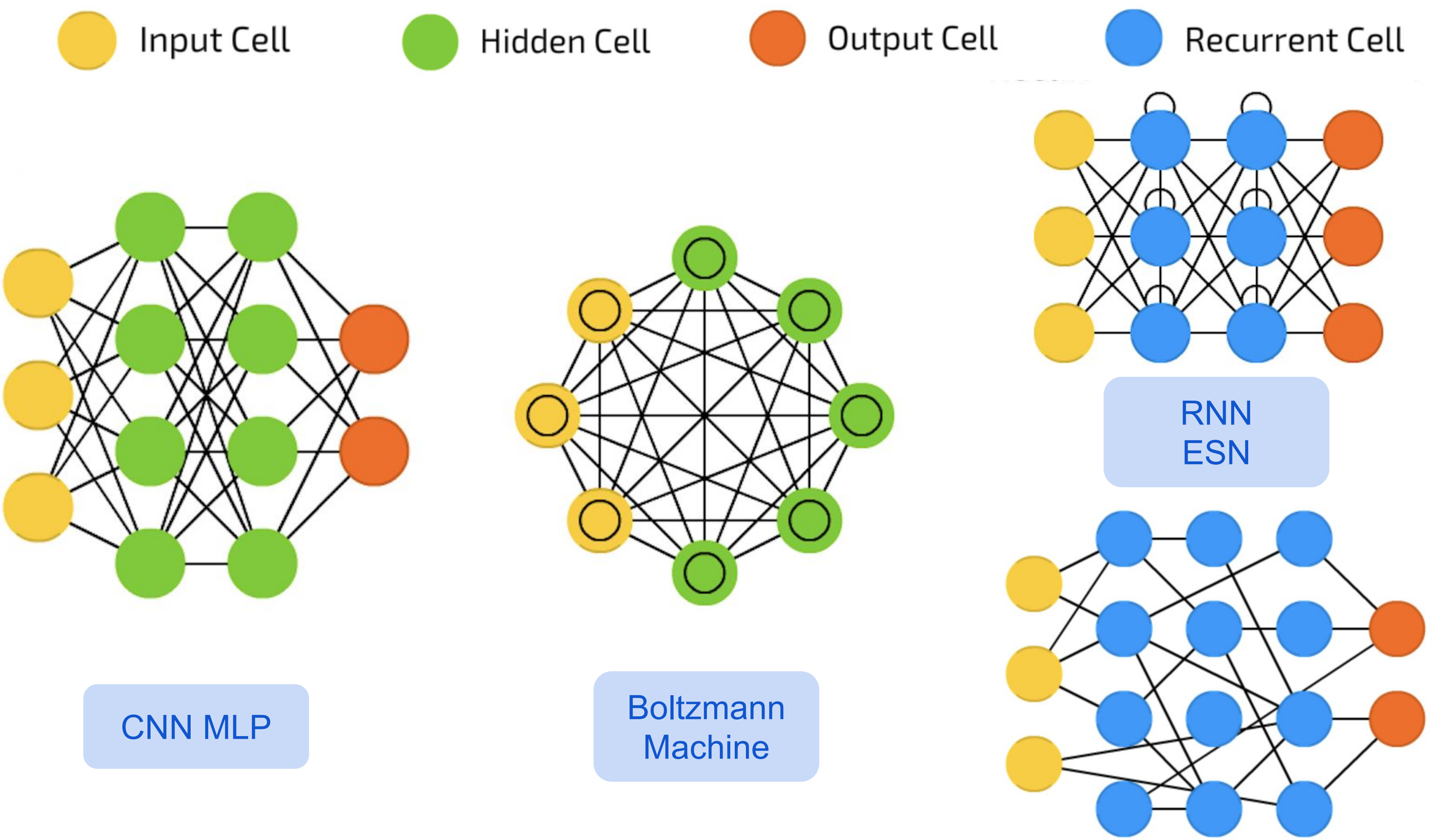


half of image corrupted by noise

# FF vs FB

今回はFB結合を含む、様々なネットワークを学んでいきます

様々なネットワークの形



# RNNの位置付け

## Neural Network

### (§5-7)Feedforward

- (§5) 単純パーセプトロン
- (§5) 多層パーセプトロン
- (§6,7) CNN

### (§9-10)Feedback

#### 再帰型

- (§9) Simple RNN
- (§10) LSTM
- (§10) GRU

#### 相互結合型

- (§8) RBM
- (§9) HFN
- (§9) ESN
- (§9) LSM

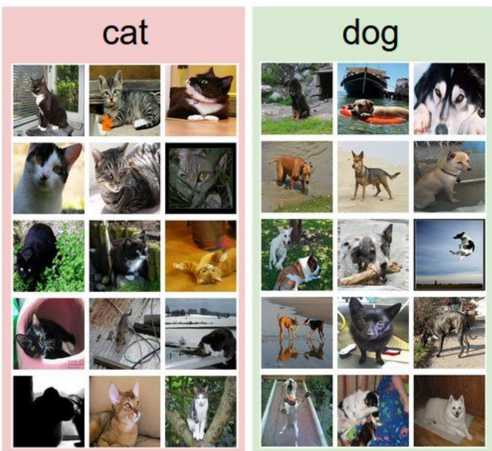
# 目次

講義 Part1

# RNNの基礎と学習

RNN and Back Propagation

# 人間は静的(static)に外界を捉えていない




(縦, 横, チャネル)

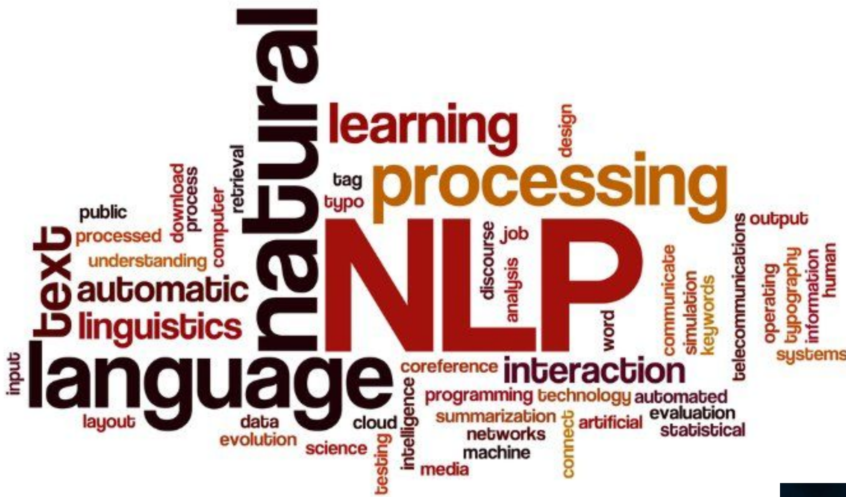
(縦, 横, チャネル, **時間**)

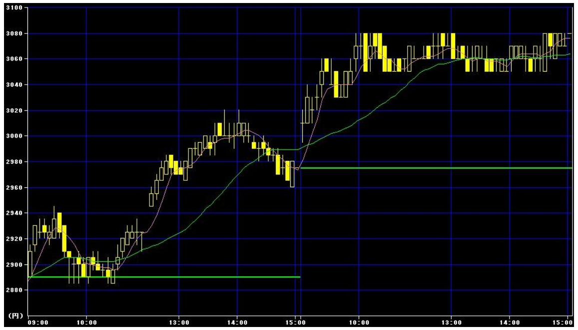
認知のプロセスには<u>時系列が大切</u>

# [動画] 記憶と時系列


逆からいえますか？

# (実は)時系列だらけの世界

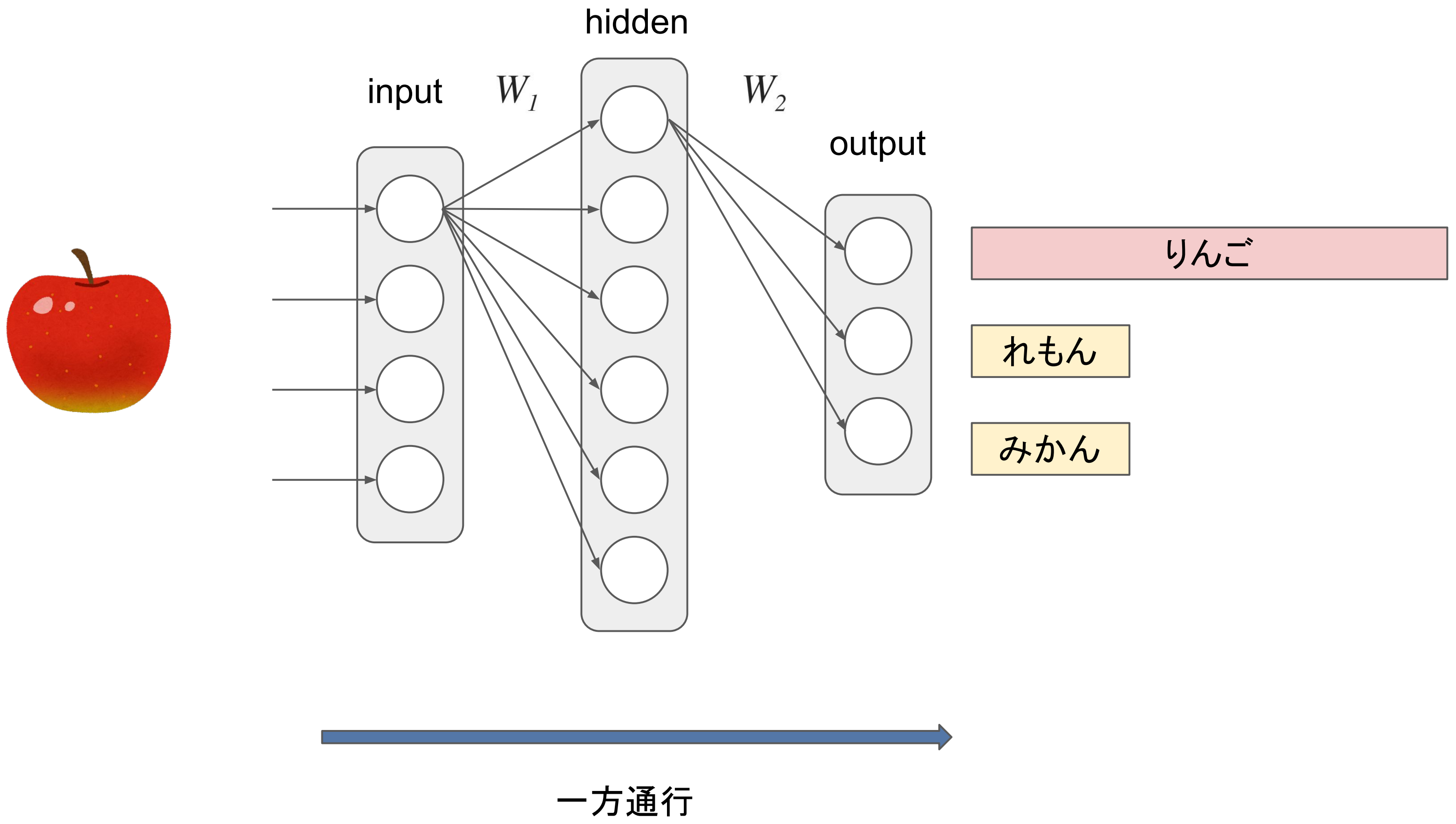


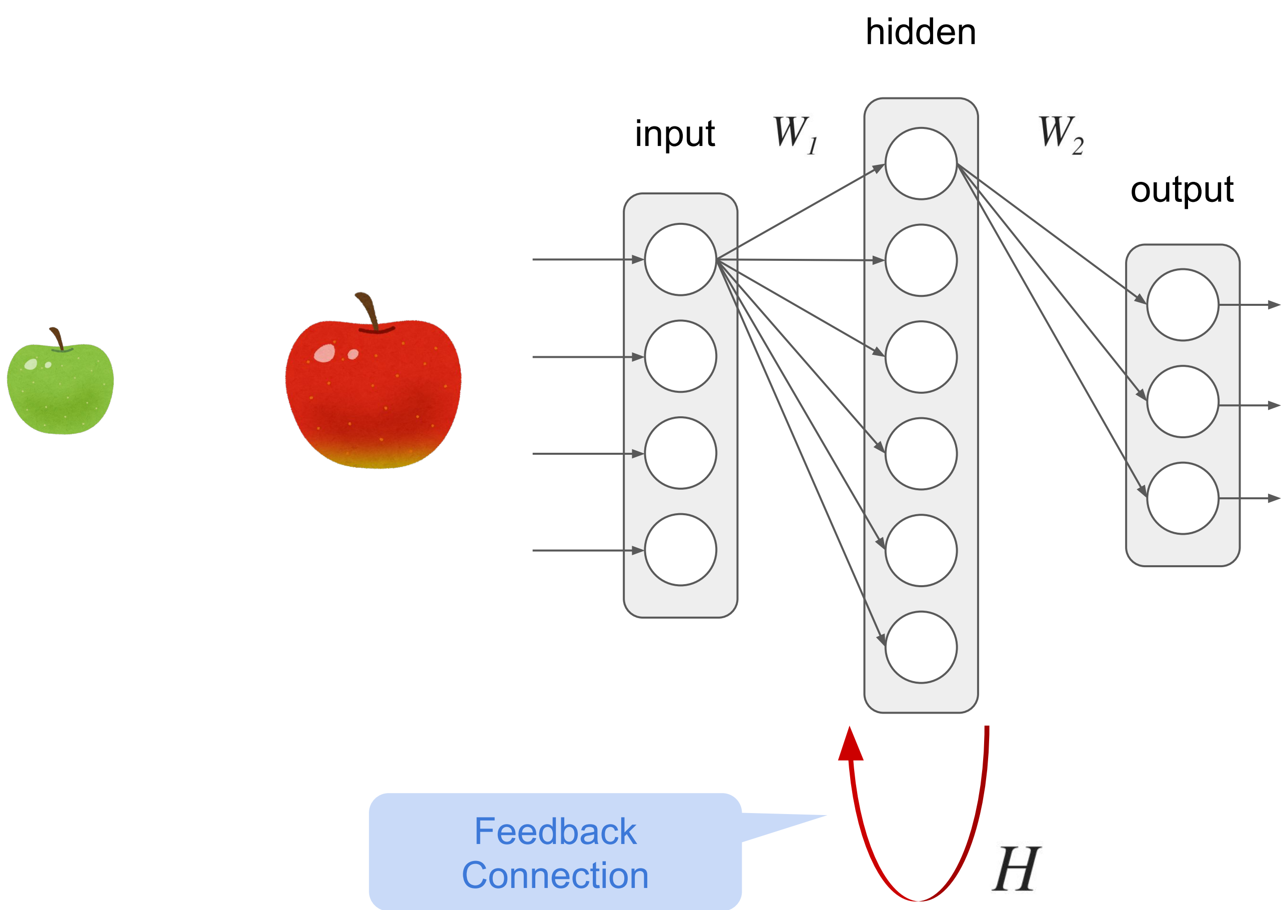
工学的に時系列を扱うことのできるネットワークは非常に重要

# Feedforward Network

## 復習

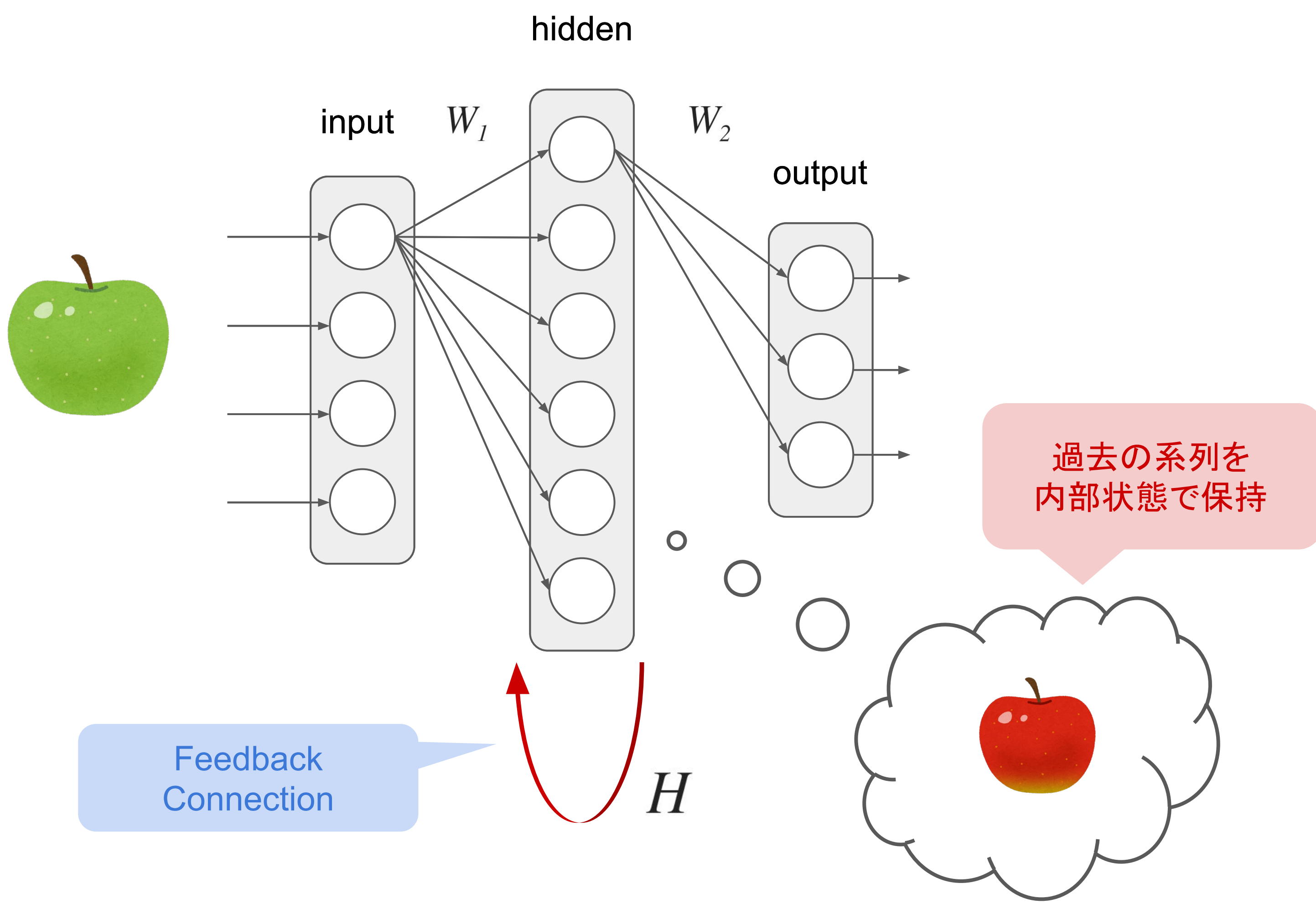


# Feedback Network

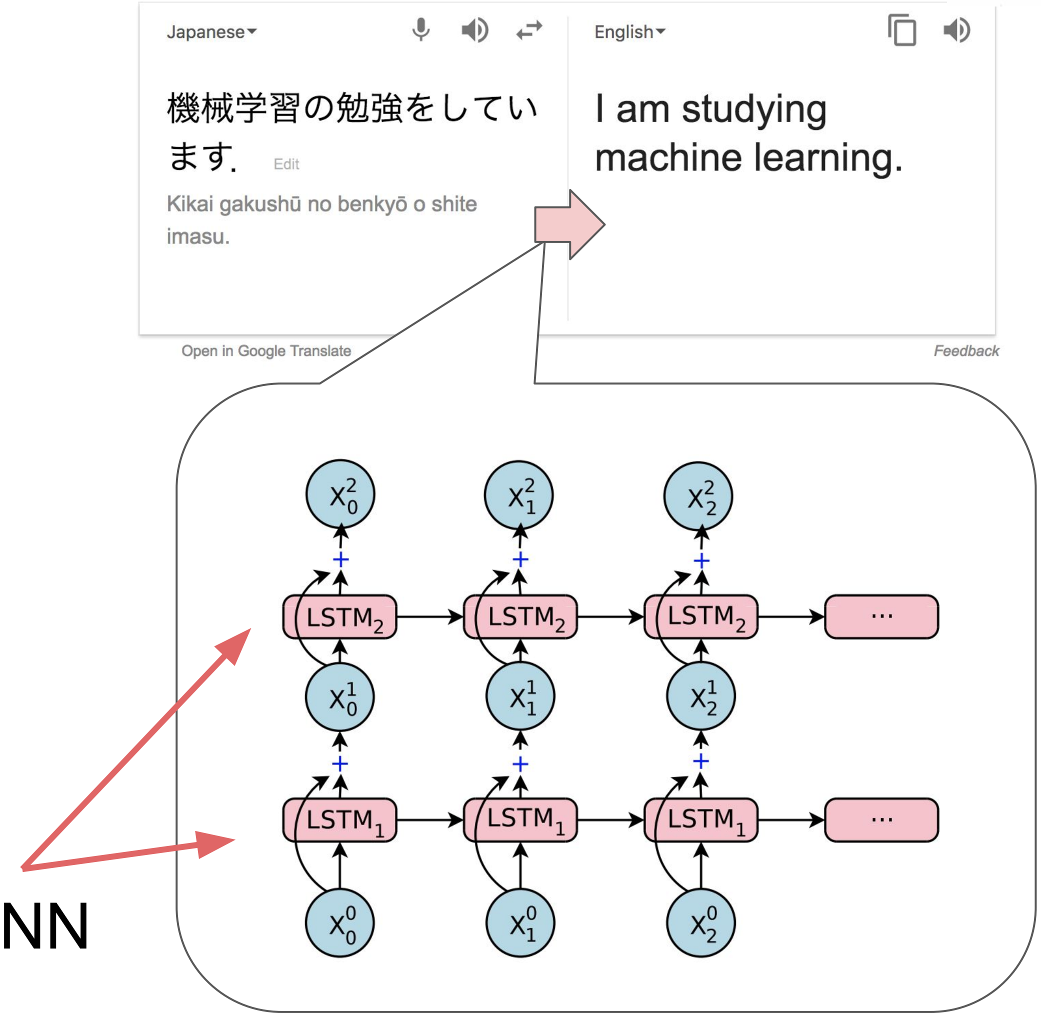


※ 隠れ層1層の場合

# Feedback Network



# 実際どこに使われているの？




RNN

# Recurrent Neural Network (RNN)

単純再帰型ネットワーク

- 初期モデルはElman(1990)が提唱
  - 文規則学習のシミュレーションに用いられた

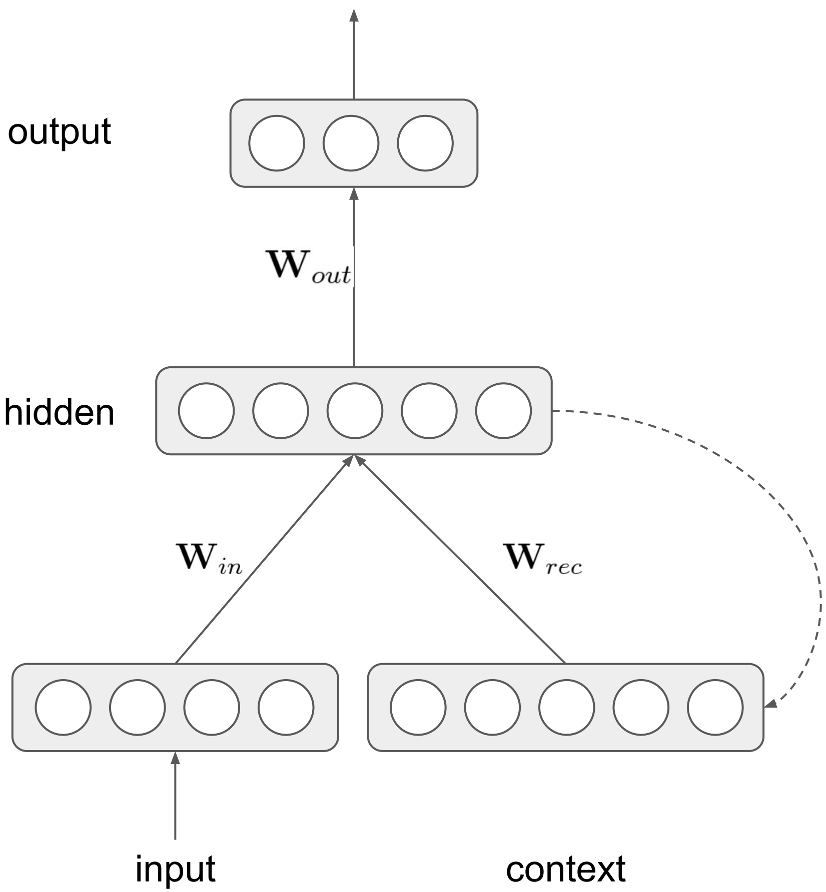
- 入力層，隠れ層，出力層
  直前の隠れ層を次時刻の隠れ層の入力に

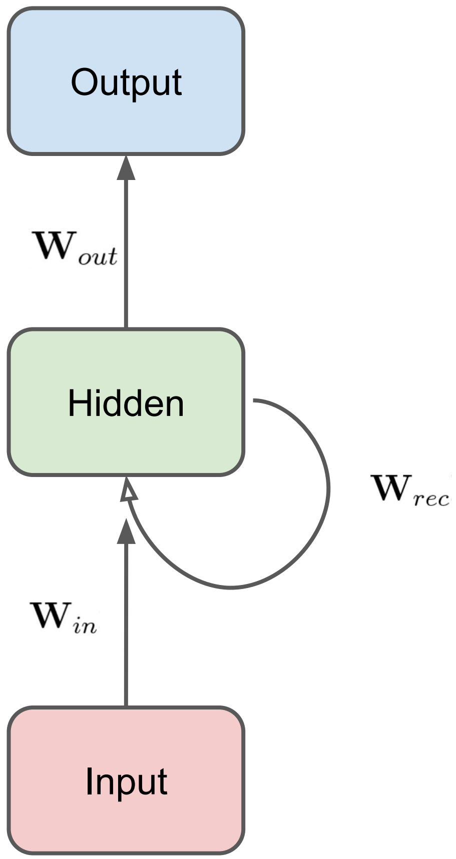
  → 時間方向の因果関係を考慮することが可能

- 可変長な入力系列を扱うことが可能

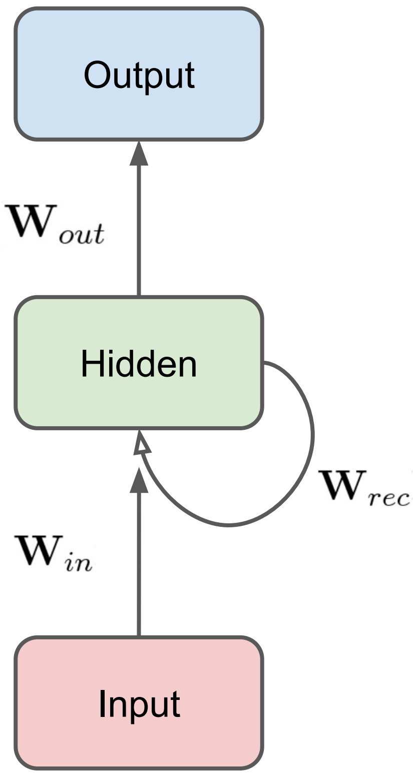
- 誤差逆伝播法 (BP)を用いて学習

# RNNの略記法


output
$\mathbf{W}_{out}$
hidden
$\mathbf{W}_{in}$
$\mathbf{W}_{rec}$
input
context


Output
$\mathbf{W}_{out}$
Hidden
$\mathbf{W}_{rec}$
$\mathbf{W}_{in}$
Input

# RNNの定式化




◇ パラメータ集合

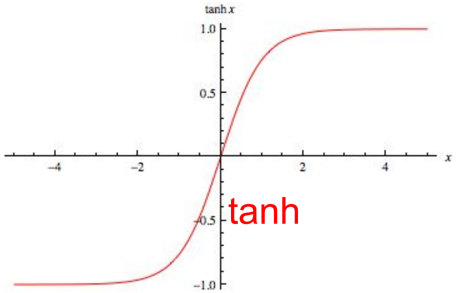
$$\theta = \{\mathbf{W}_{in}, \mathbf{W}_{rec}, \mathbf{W}_{out}, \mathbf{b}, \mathbf{c}\}$$

◇ 隠れ層(Hidden)の状態

$$\mathbf{h}(t) = \boldsymbol{\Phi}_{\mathrm{H}}(\mathbf{W}_{\mathrm{in}}\mathbf{x}^{(\mathbf{t})} + \mathbf{W}_{\mathrm{rec}}\mathbf{h}^{(\mathbf{t-1})} + \mathbf{b})$$

◇ 出力層(Output)の状態

$$\mathbf{y}(t) = \boldsymbol{\Phi}_{\mathrm{O}}(\mathbf{W}_{\mathrm{out}}\mathbf{h}^{(t)} + \mathbf{c})$$

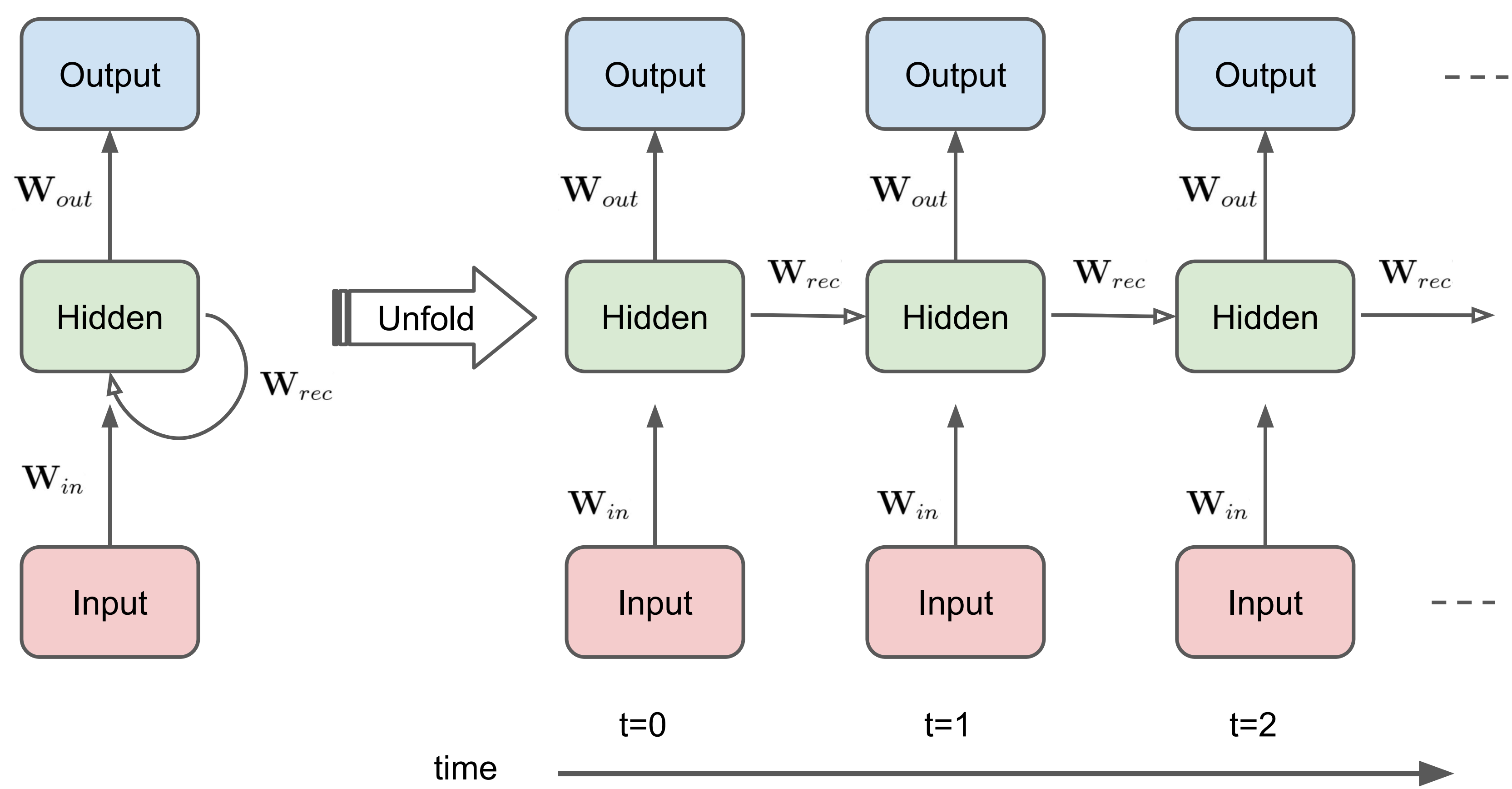



Activation function

# 時間軸方向への展開

この変形はよく出てくる

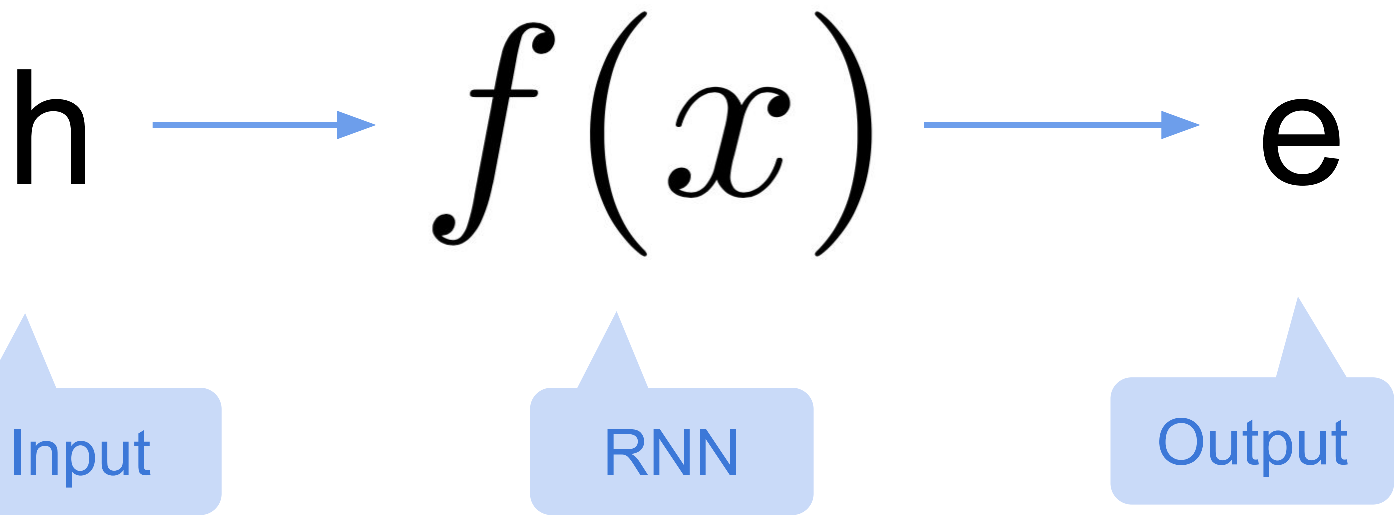
Feedback Network → Unfold → Feedforward Network



# (例) 順伝播の計算 (1/3)

“hello”を学習した
Character Level RNN

“hello”



# (例) 順伝播の計算 (2/3)

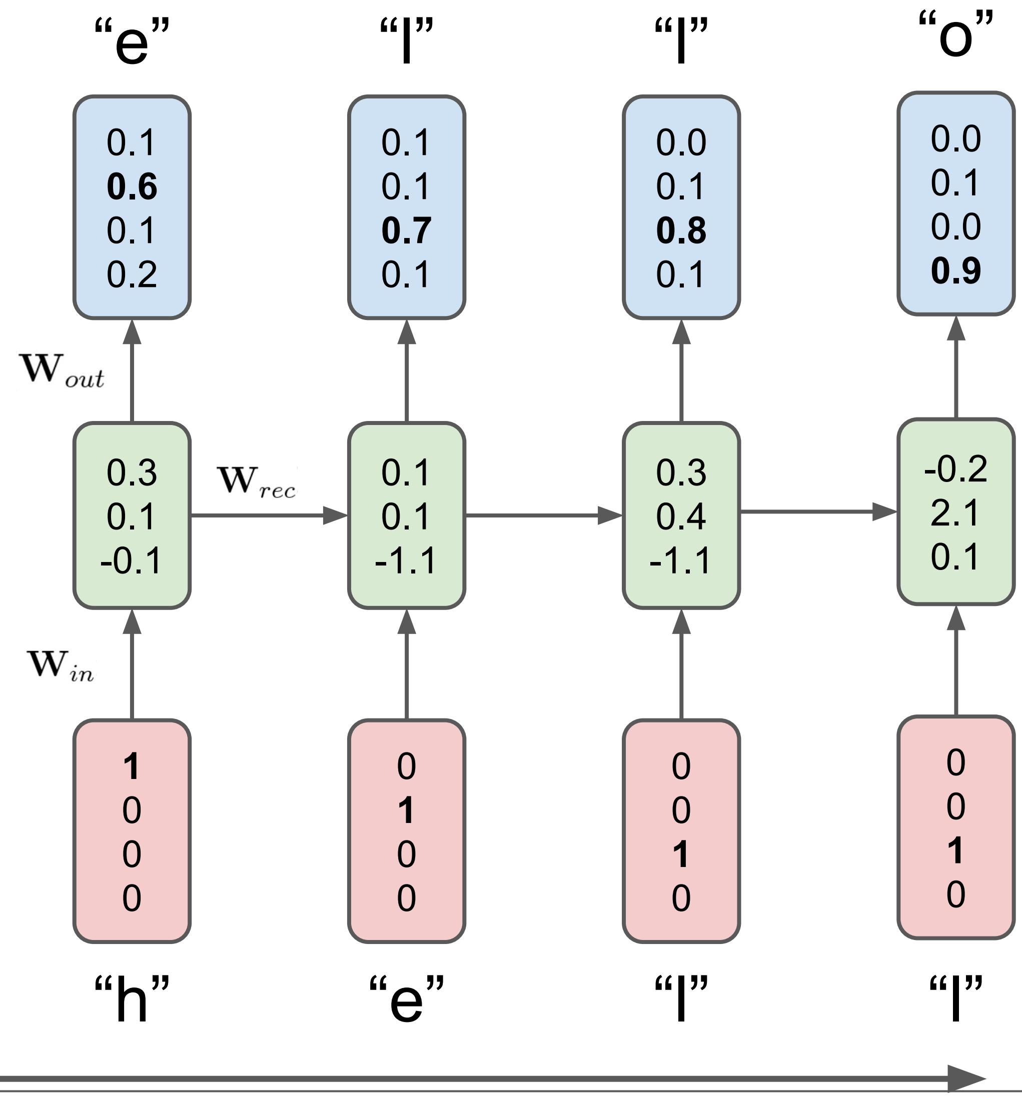
“hello”を学習した
Character Level RNN

“hello”

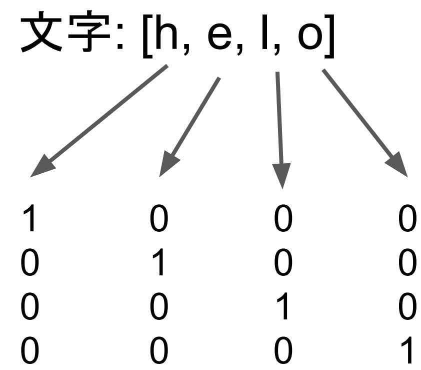
$$l \longrightarrow f(x) \longrightarrow o$$

Input　　RNN　　Output

# (例) 順伝播の計算 (3/3)

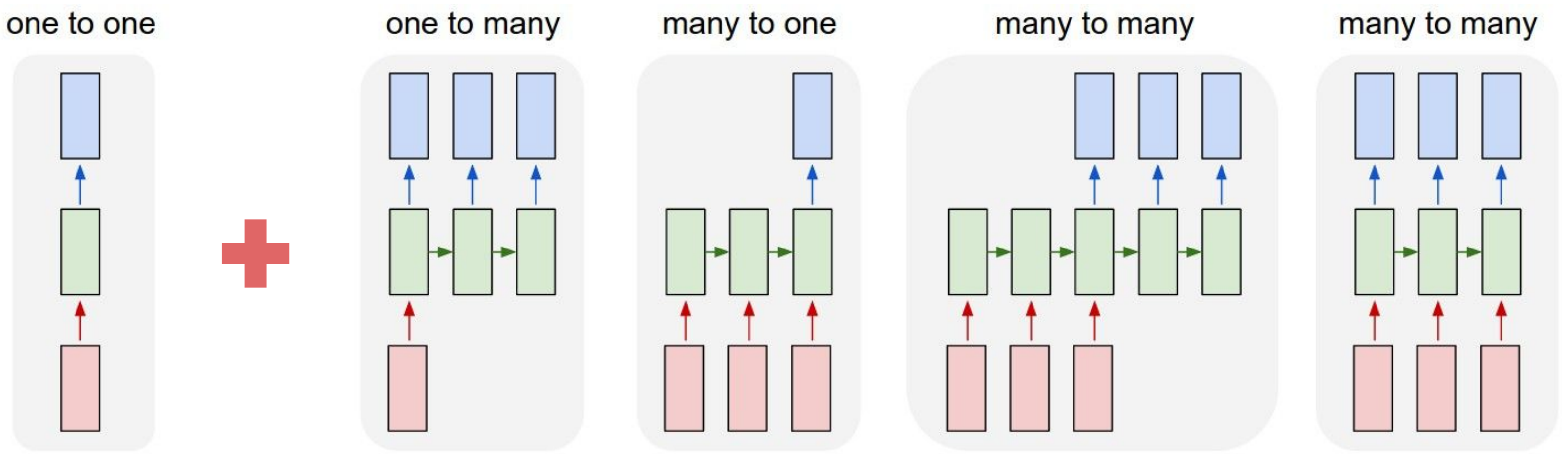



“hello”を学習した
Character Level RNN



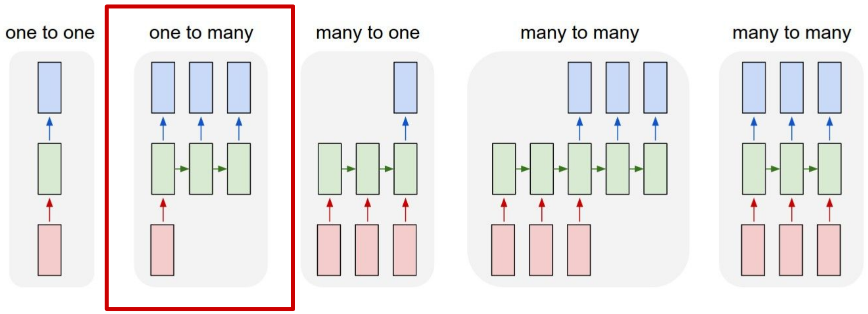

One-hot vector表現
分散表現の一種

# RNNの柔軟性 (1/5)

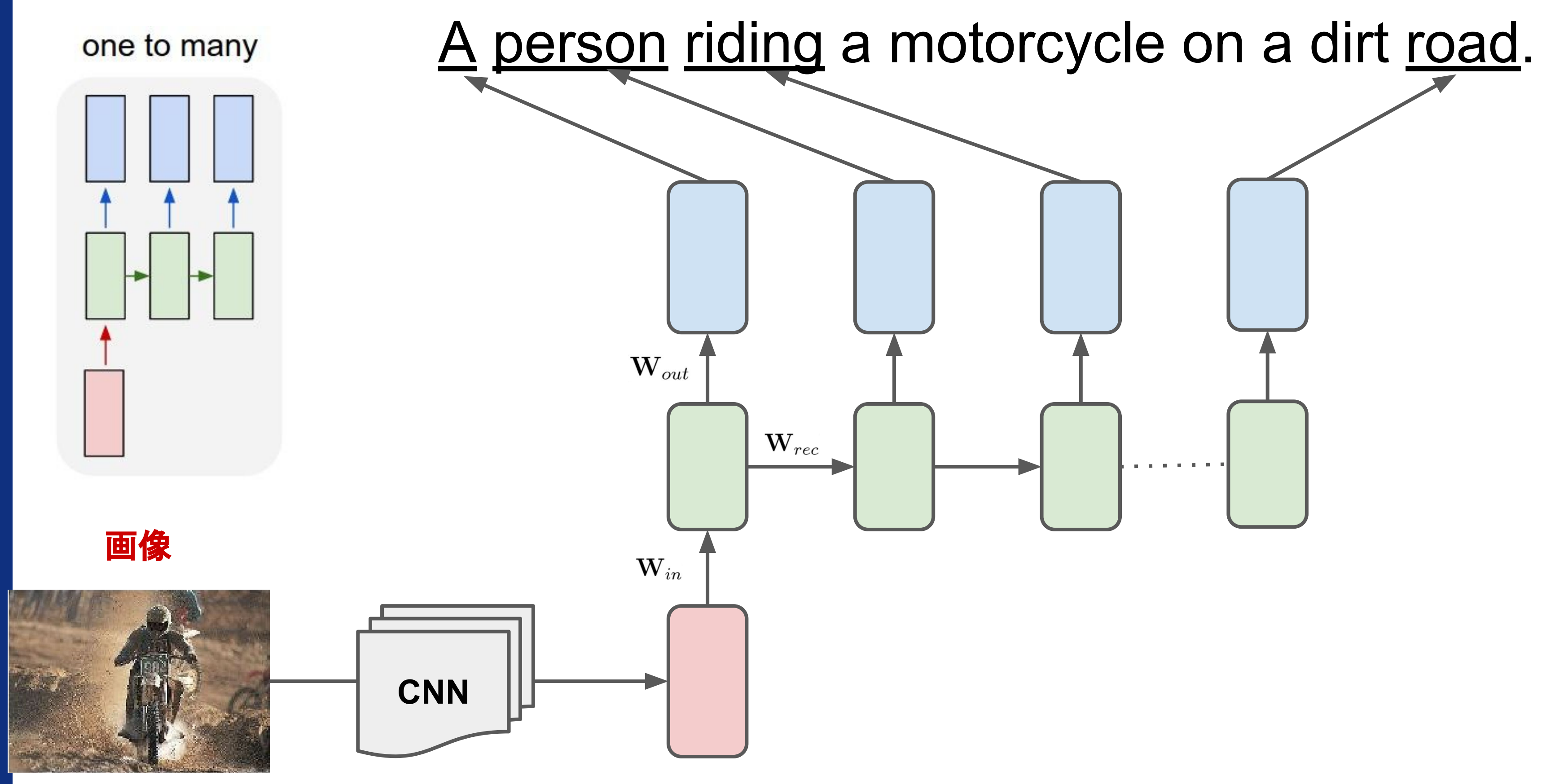



<u>入出力の対応関係</u>を変えることで様々な用途に対応することが可能

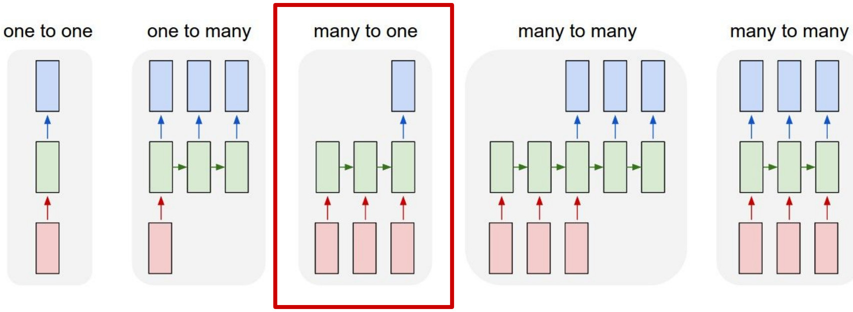
# RNNの柔軟性 (2/5)



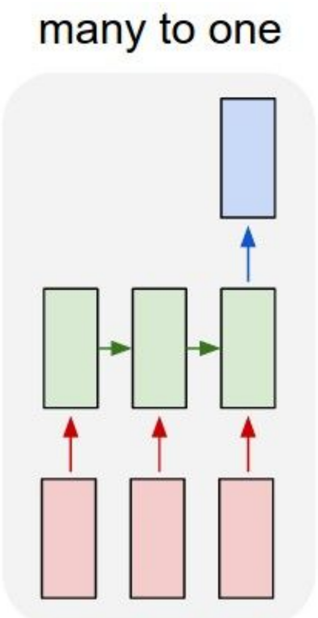

## 画像キャプショニング

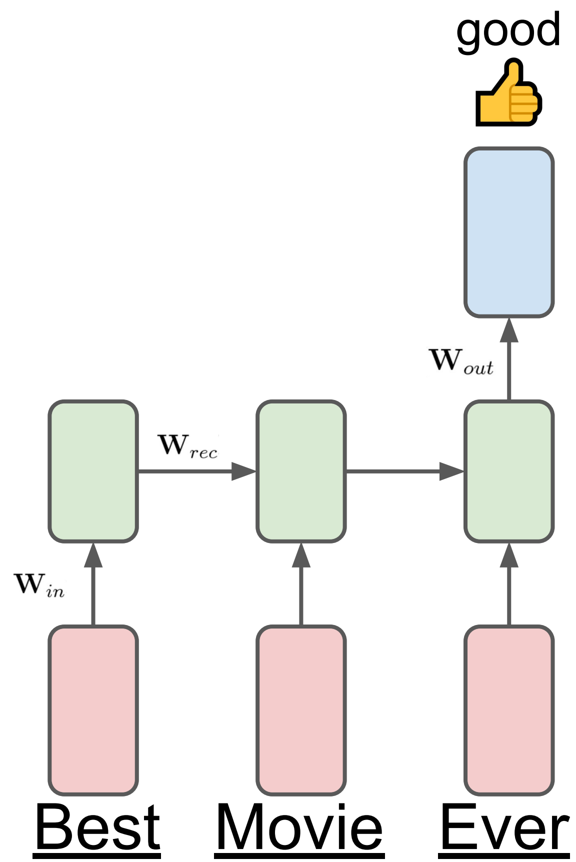


# RNNの柔軟性 (3/5)

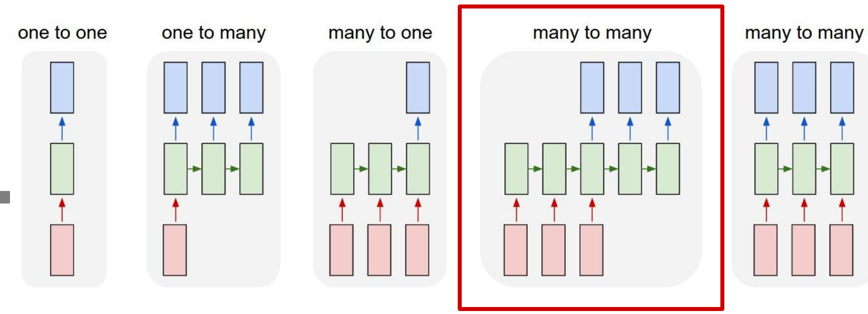



## 要約

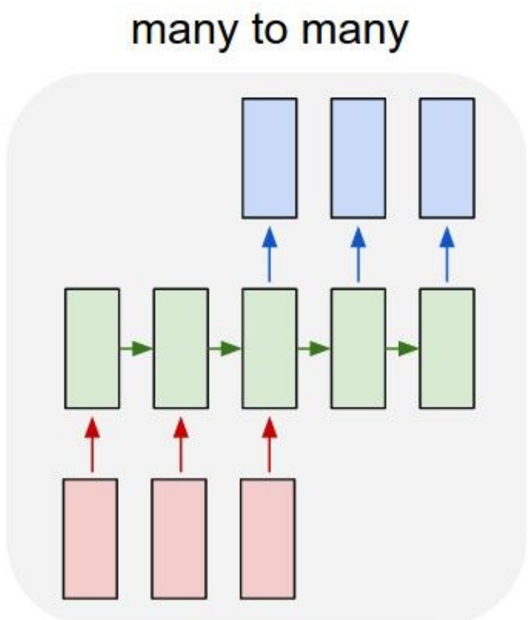


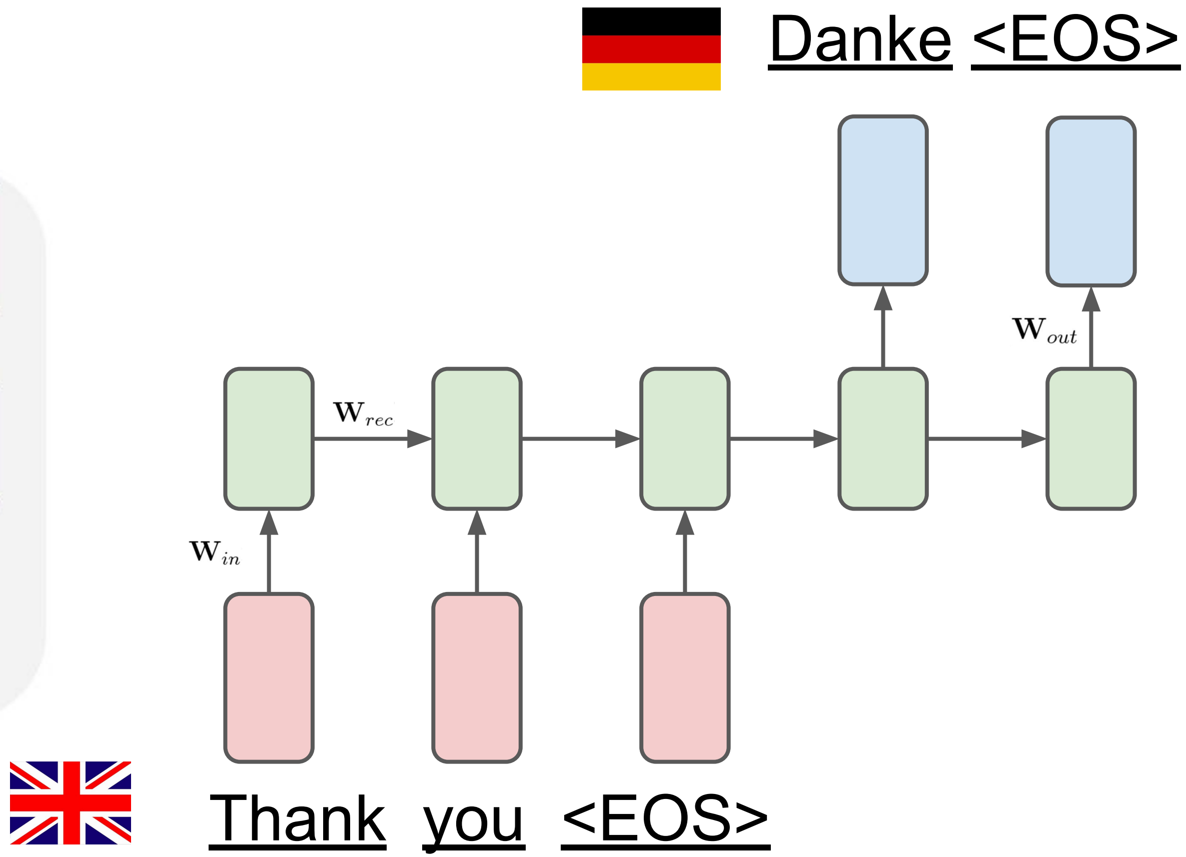



# RNNの柔軟性 (4/5)

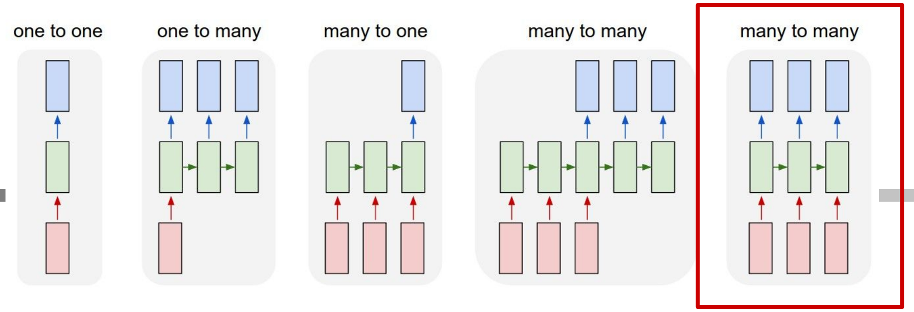



## 機械翻訳

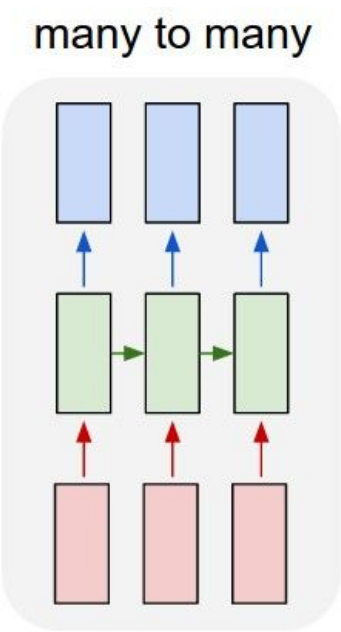


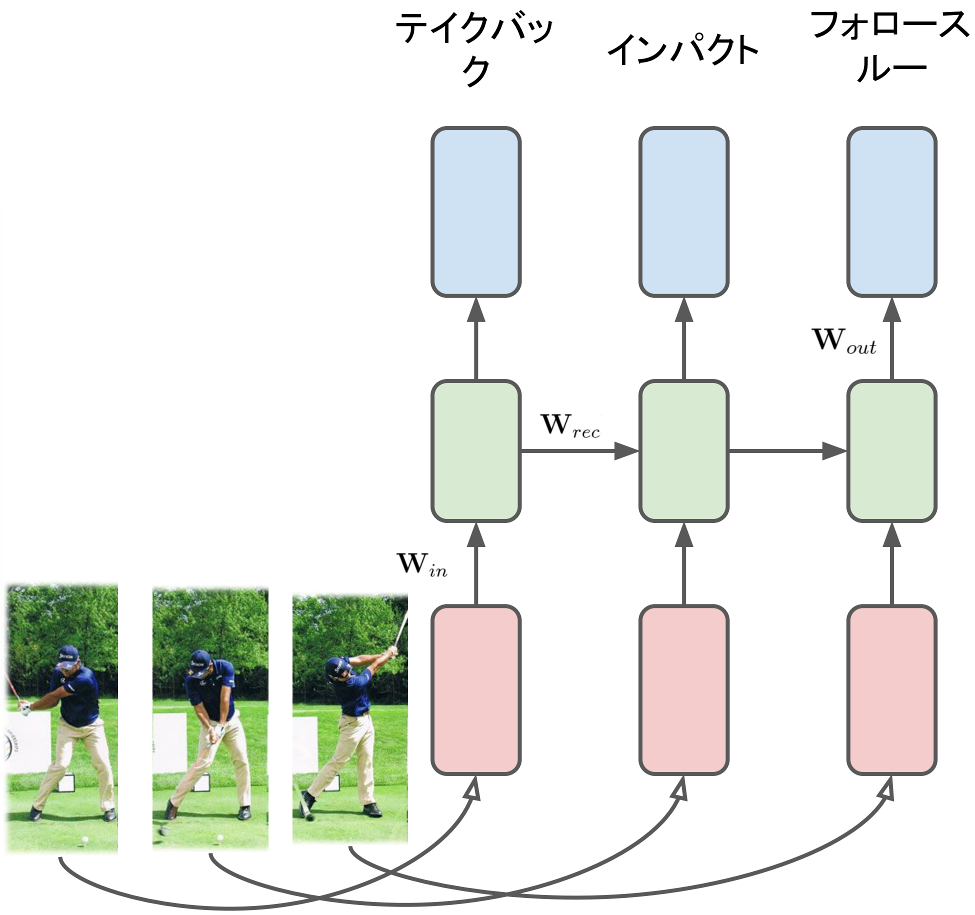



# RNNの柔軟性 (5/5)

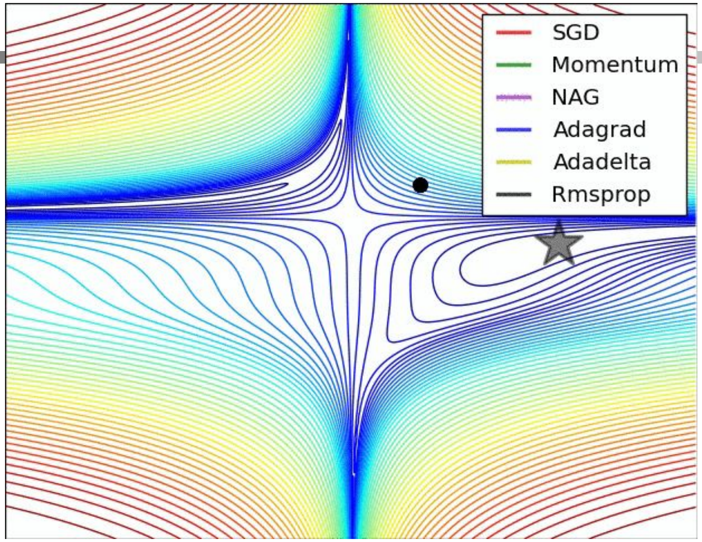



## 動画キャプショニング



# RNNの学習法




## Gradient Descent (GD)

- 最急降下法

勾配を求めたい

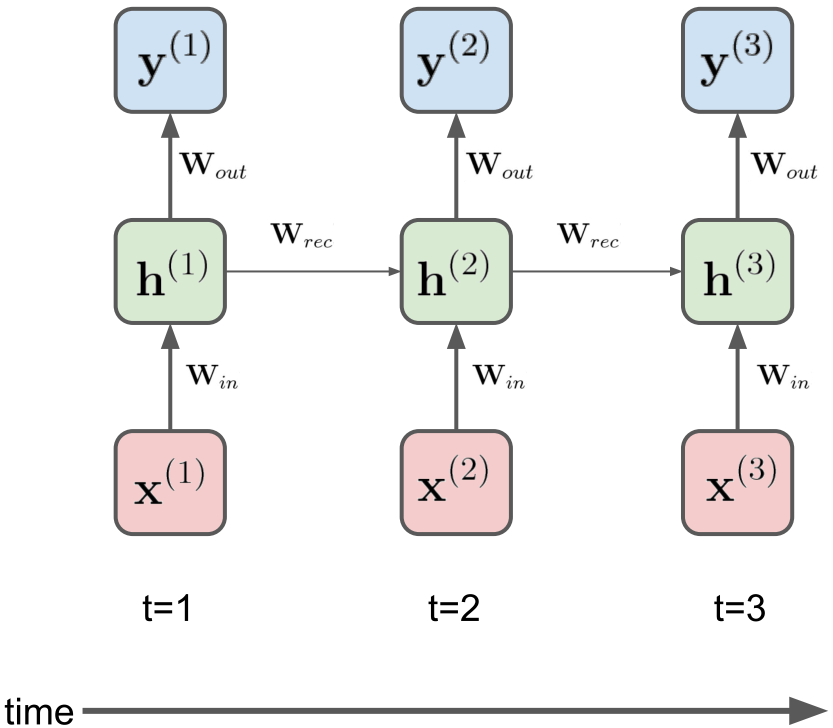
$$\theta^{k+1} = \theta^k - \alpha \boxed{\frac{\partial J}{\partial \theta}}$$

## Backpropagation Through Time (BPTT)

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

# BPTTによる学習: 具体例(1/6)

## 順伝搬計算




◇ 時刻 t = 1

定数

$$\mathbf{h}^{(1)} = \Phi_{\mathrm{H}}(\mathbf{W}_{\mathrm{in}}\mathbf{x}^{(1)} + \mathbf{W}_{\mathrm{rec}}\mathbf{h}^{(0)} + \mathbf{b})$$

$$\mathbf{y}^{(1)} = \Phi_{\mathrm{O}}(\mathbf{W}_{\mathrm{out}}\mathbf{h}^{(1)} + \mathbf{c})$$

◇ 時刻 t = 2

$$\mathbf{h}^{(2)} = \Phi_{\mathrm{H}}(\mathbf{W}_{\mathrm{in}}\mathbf{x}^{(2)} + \mathbf{W}_{\mathrm{rec}}\mathbf{h}^{(1)} + \mathbf{b})$$

$$\mathbf{y}^{(2)} = \Phi_{\mathrm{O}}(\mathbf{W}_{\mathrm{out}}\mathbf{h}^{(2)} + \mathbf{c})$$

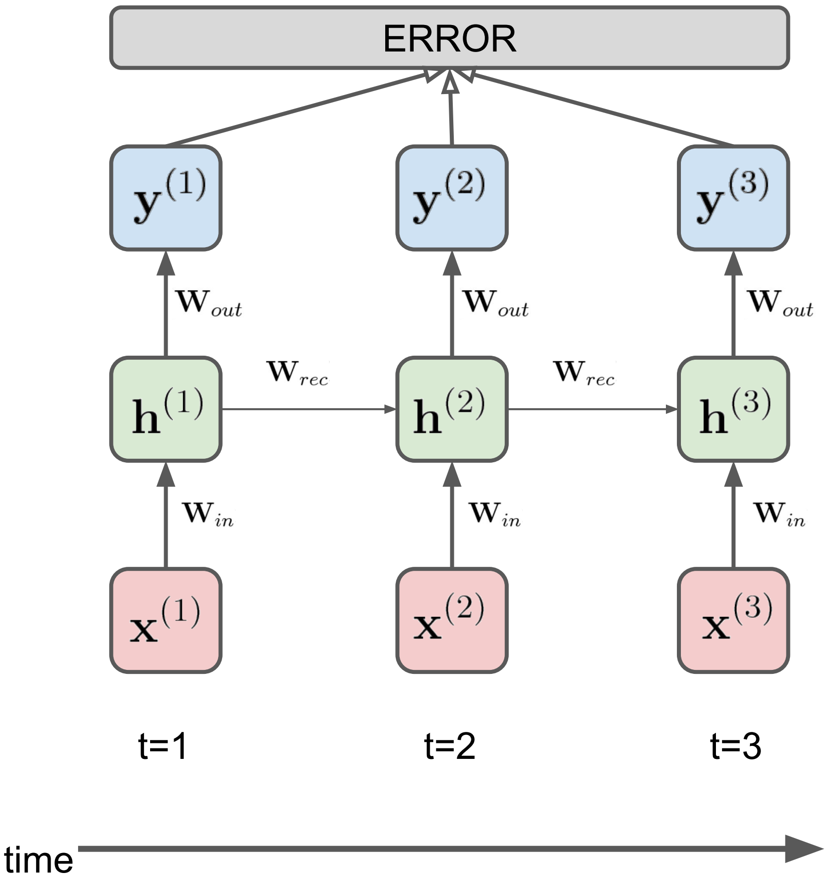
◇ 時刻 t = 3

$$\mathbf{h}^{(3)} = \Phi_{\mathrm{H}}(\mathbf{W}_{\mathrm{in}}\mathbf{x}^{(3)} + \mathbf{W}_{\mathrm{rec}}\mathbf{h}^{(2)} + \mathbf{b})$$

$$\mathbf{y}^{(3)} = \Phi_{\mathrm{O}}(\mathbf{W}_{\mathrm{out}}\mathbf{h}^{(3)} + \mathbf{c})$$

# BPTTによる学習: 具体例 (2/6)

最急降下法による最小値問題




◇ 誤差の計算 (L2ノルム)

教師信号

$$J = \frac{1}{2}\sum_{1\leq t\leq 3}(\mathbf{y}^{(t)} - \mathbf{d}^{(t)})^2$$

最小化したい

FFNETと同じように<u>最急降下法</u>で
最小値問題を解く

◇ 最急降下法

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

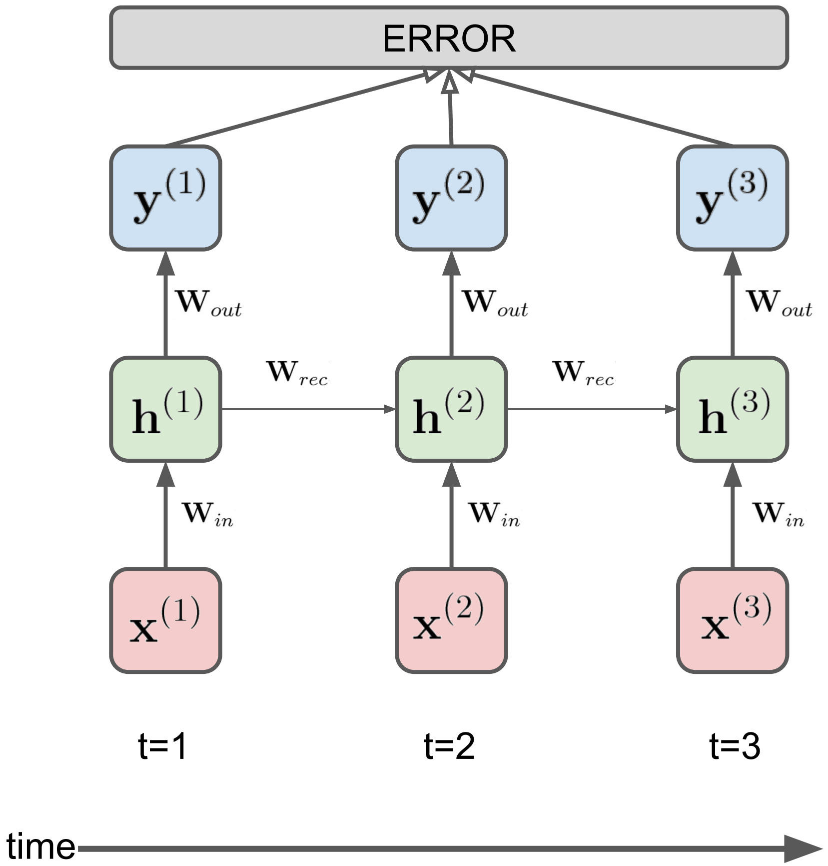
BPTTで
求める

□ パラメータ

$$\theta = \{\mathbf{W}_{in}, \mathbf{W}_{rec}, \mathbf{W}_{out}, \mathbf{b}, \mathbf{c}\}$$

# BPTTによる学習: 具体例 (3/6)

誤差逆伝播法




$$C^{(k)} \triangleq \frac{1}{2}(\boldsymbol{y}^{(k)} - \boldsymbol{d}^{(k)})$$

◇ 誤差逆伝播

□ 置き換え

$$J = \sum_{1 \leq t \leq 3} C^{(t)}$$

時刻tの出力誤差

□ 偏微分計算

$$\frac{\partial J}{\partial \theta} = \frac{\partial}{\partial \theta} \sum_{1 \leq t \leq 3} C^{(t)}$$

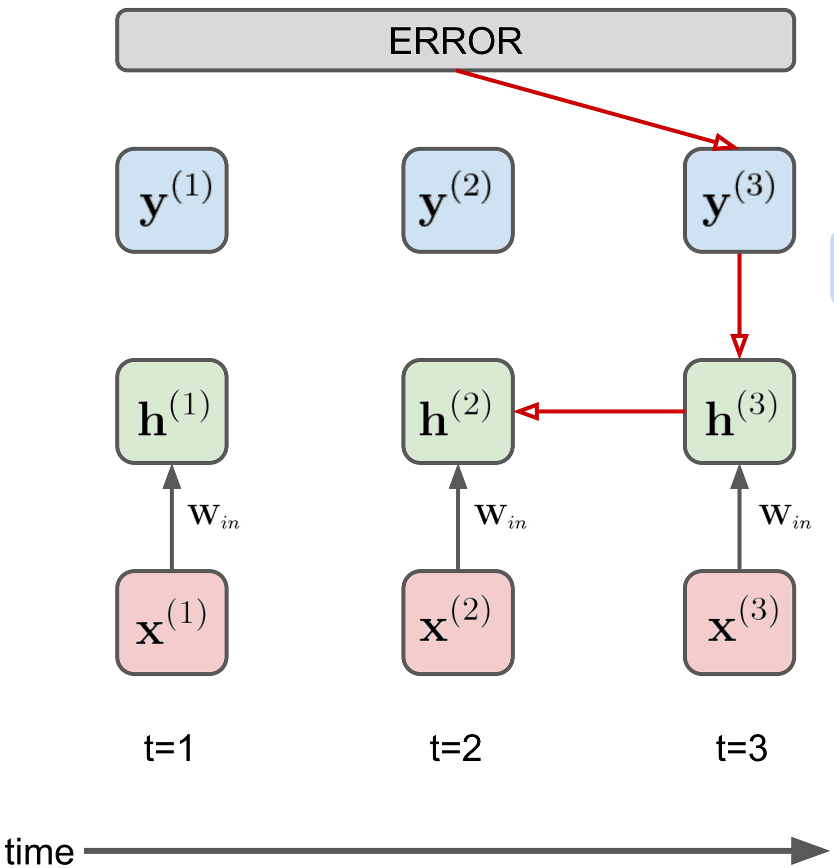
$$= \sum_{1 \leq t \leq 3} \boxed{\frac{\partial C^{(t)}}{\partial \theta}}$$

t=3の場合を求めてみる

※簡便のため行ベクトル表記を用いる

# BPTTによる学習: 具体例 (4/6)

誤差逆伝播法




◇ 誤差逆伝播 t=3の場合を求めてみる

□ C3の偏微分計算

$$\frac{\partial C^{(3)}}{\partial \theta} = \frac{\partial C^{(3)}}{\partial \mathbf{y}^{(3)}} \frac{\partial \mathbf{y}^{(3)}}{\partial \mathbf{h}^{(3)}}$$

y3の関数

h3の関数

$$\frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}}$$

h2の関数

◇ 時刻 t = 3

$$J = \frac{1}{2} \sum_{1 \le t \le 3} \left(\boxed{\mathbf{y}^{(t)}} - \mathbf{d}^{(t)}\right)^2$$

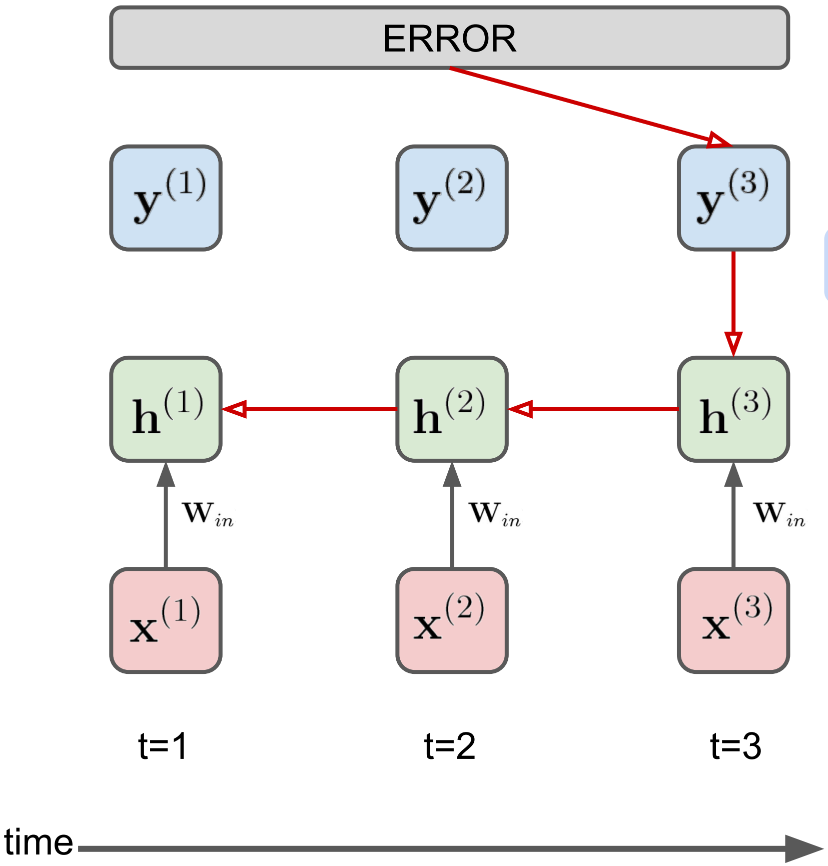
$$\mathbf{y}^{(3)} = \Phi_{\mathrm{O}}(\mathbf{W}_{\mathrm{out}} \boxed{\mathbf{h}^{(3)}} + \mathbf{c})$$

$$\mathbf{h}^{(3)} = \Phi_{\mathrm{H}}(\mathbf{W}_{\mathrm{in}} \mathbf{x}^{(3)} + \mathbf{W}_{\mathrm{rec}} \boxed{\mathbf{h}^{(2)}} + \mathbf{b})$$

※簡便のため行ベクトル表記を用いる

# BPTTによる学習: 具体例 (5/6)

誤差逆伝播法




◇ 誤差逆伝播　t=3の場合を求めてみる

□ C3の偏微分計算

$$\frac{\partial C^{(3)}}{\partial \theta} = \frac{\partial C^{(3)}}{\partial \mathbf{y}^{(3)}} \frac{\partial \mathbf{y}^{(3)}}{\partial \mathbf{h}^{(3)}}$$

y3の関数

h3の関数

$$\frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}}$$

h2の関数

h1の関数

$$\frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \frac{\partial \mathbf{h}^{(1)}}{\partial \theta}$$

◇ 時刻 t = 2, 1

$$\mathbf{h}^{(2)} = \mathbf{\Phi}_{\mathrm{H}}(\mathbf{W}_{\mathrm{in}}\mathbf{x}^{(2)} + \mathbf{W}_{\mathrm{rec}}\boxed{\mathbf{h}^{(1)}} + \mathbf{b})$$

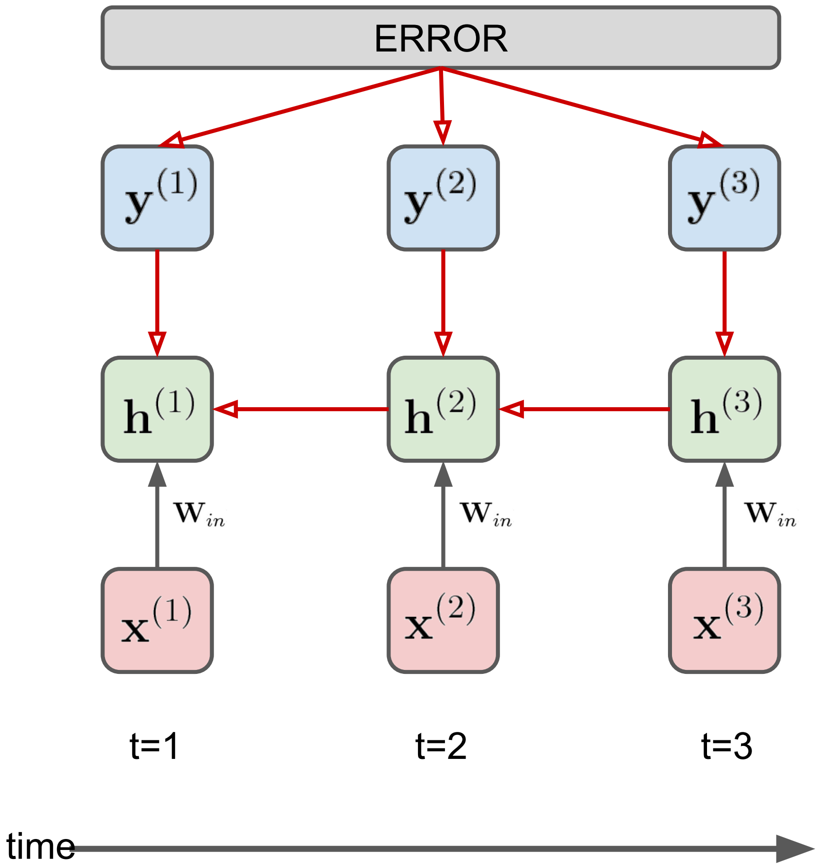
定数

$$\mathbf{h}^{(1)} = \mathbf{\Phi}_{\mathrm{H}}(\mathbf{W}_{\mathrm{in}}\mathbf{x}^{(1)} + \mathbf{W}_{\mathrm{rec}}\boxed{\mathbf{h}^{(0)}} + \mathbf{b})$$

※簡便のため行ベクトル表記を用いる

# BPTTによる学習: 具体例 (6/6)

誤差逆伝播法




## ◇ 誤差逆伝播

t=1, 2, 3の場合

□ C3の偏微分計算

$$\frac{\partial C^{(3)}}{\partial \theta} = \frac{\partial C^{(3)}}{\partial \mathbf{y}^{(3)}} \frac{\partial \mathbf{y}^{(3)}}{\partial \mathbf{h}^{(3)}} \frac{\partial \mathbf{h}^{(3)}}{\partial \mathbf{h}^{(2)}} \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \frac{\partial \mathbf{h}^{(1)}}{\partial \theta}$$

□ C2の偏微分計算

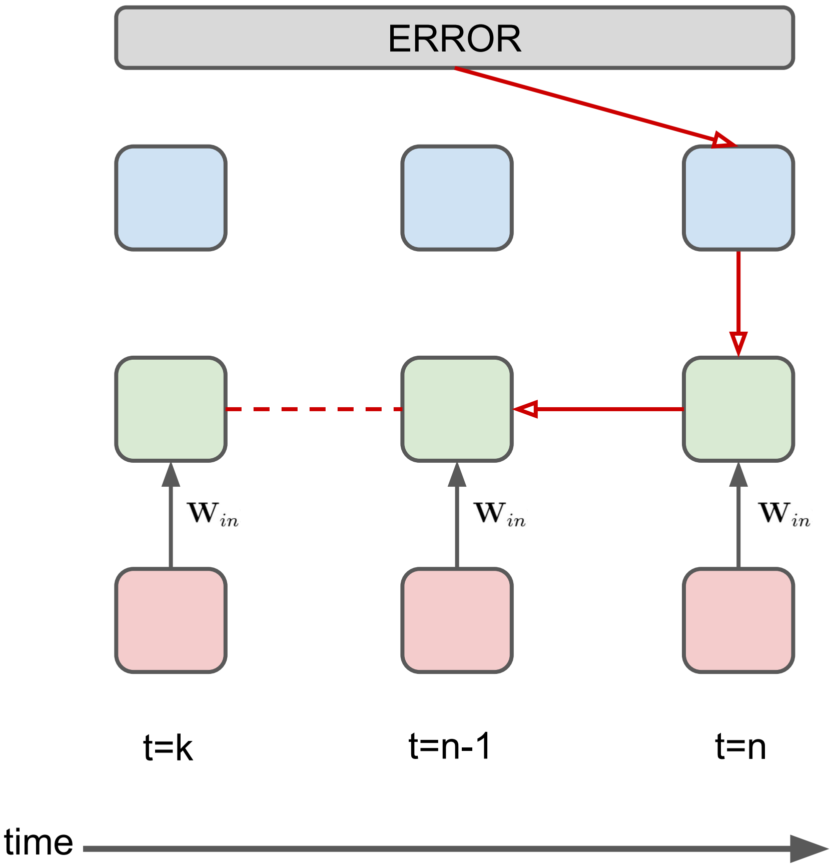
$$\frac{\partial C^{(2)}}{\partial \theta} = \frac{\partial C^{(2)}}{\partial \mathbf{y}^{(2)}} \frac{\partial \mathbf{y}^{(2)}}{\partial \mathbf{h}^{(2)}} \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \frac{\partial \mathbf{h}^{(1)}}{\partial \theta}$$

□ C1の偏微分計算

$$\frac{\partial C^{(1)}}{\partial \theta} = \frac{\partial C^{(1)}}{\partial \mathbf{y}^{(1)}} \frac{\partial \mathbf{h}^{(2)}}{\partial \mathbf{h}^{(1)}} \frac{\partial \mathbf{h}^{(1)}}{\partial \theta}$$

※簡便のため行ベクトル表記を用いる

# BPTTによる学習: 一般化




## ◇ 誤差逆伝播

t=n の場合

□ Cnの偏微分計算

k=1

$$\frac{\partial C^{(n)}}{\partial \theta} = \frac{\partial C^{(n)}}{\partial \mathbf{y}^{(n)}} \frac{\partial \mathbf{y}^{(n)}}{\partial \mathbf{h}^{(n)}} \frac{\partial \mathbf{h}^{(n)}}{\partial \mathbf{h}^{(k)}} \frac{\partial \mathbf{h}^{(k)}}{\partial \theta}$$

h(z)とする

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

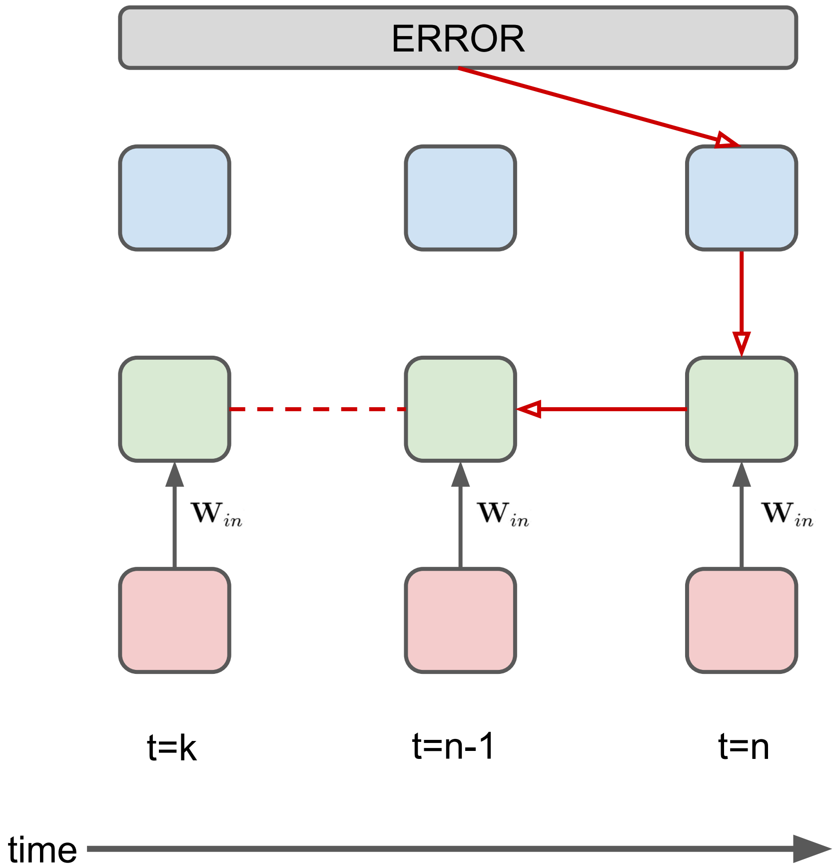
$$= \prod_{k<i\leq n} \frac{\partial \boldsymbol{h}^{(i)}}{\partial \boldsymbol{z}^{(i)}} \frac{\partial \boldsymbol{z}^{(i)}}{\partial \boldsymbol{h}^{(i-1)}}$$

$$= \prod_{k<i\leq n} \boldsymbol{\Phi}'_{\mathrm{H}}(\boldsymbol{z}^{(\boldsymbol{i})}) W^{T}_{\mathrm{rec}}$$

$$\triangleq \prod_{k<i\leq n} \boldsymbol{\Phi}'_{\mathrm{H}}{}^{(i)} W^{T}_{\mathrm{rec}}$$

※簡便のため行ベクトル表記を用いる

# BPTTによる学習: 勾配爆発・消失 (1/2)




## ◇ 誤差逆伝播

t=n の場合

□ Cnの偏微分計算

$$\frac{\partial C^{(n)}}{\partial \theta} = \frac{\partial C^{(n)}}{\partial \mathbf{y}^{(n)}} \frac{\partial \mathbf{y}^{(n)}}{\partial \mathbf{h}^{(n)}} \Big( \prod_{k<i\leq n} \boldsymbol{\Phi}_{\mathrm{H}}^{'(i)} W_{\mathrm{rec}}^{T} \Big) \frac{\partial \mathbf{h}^{(k)}}{\partial \theta}$$

時系列の長いデータ

n大, k小 → chainが長くなる

□ n = 100, k = 1の場合

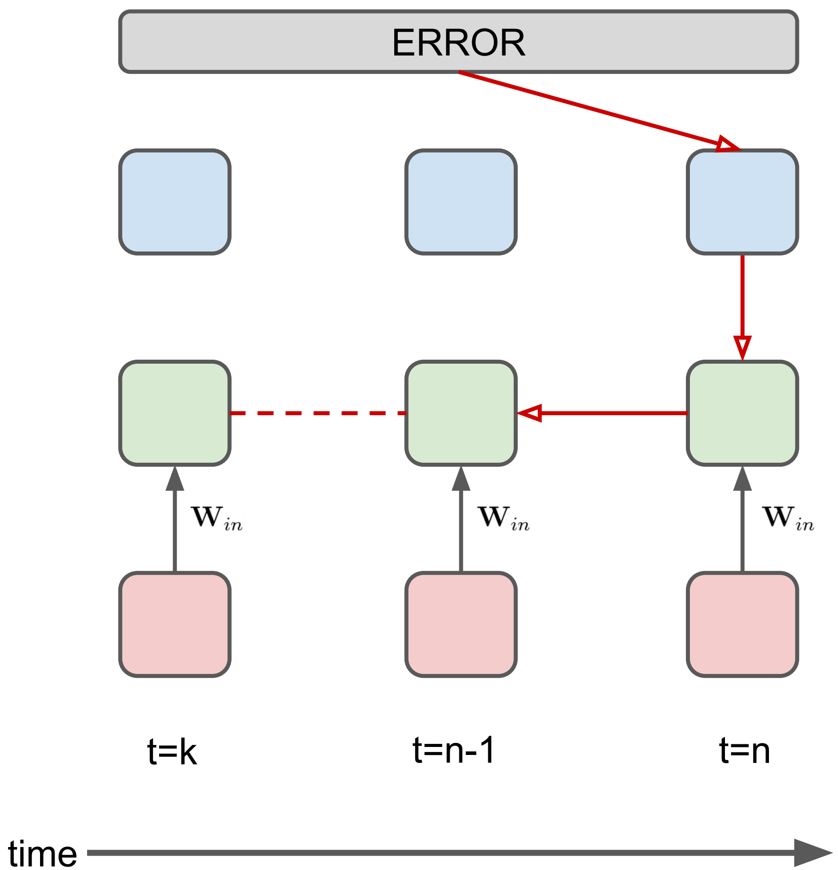
$$\frac{\partial C^{(100)}}{\partial \theta} = \frac{\partial C^{(100)}}{\partial \mathbf{y}^{(100)}} \frac{\partial \mathbf{y}^{(100)}}{\partial \mathbf{h}^{(100)}} ||\boldsymbol{\Phi}_{\mathrm{H}}^{'}||^{99} ||W_{\mathrm{rec}}^{T}||^{99} \frac{\partial \mathbf{h}^{(1)}}{\partial \theta}$$

$$\approx \boxed{||\boldsymbol{\Phi}_{\mathrm{H}}^{'}||^{99}} \boxed{||W_{\mathrm{rec}}^{T}||^{99}}$$

||x|| < 1: で勾配が消失 (→ 0 )
||x|| > 1: で勾配が爆発 (→ ∞)

※簡便のため行ベクトル表記を用いる

# BPTTによる学習: 勾配爆発・消失 (2/2)




## ◇ 誤差逆伝播

t=n の場合

□ Cnの偏微分計算 (まとめ)

$$\frac{\partial C^{(n)}}{\partial \theta} = \frac{\partial C^{(n)}}{\partial \mathbf{y}^{(n)}} \frac{\partial \mathbf{y}^{(n)}}{\partial \mathbf{h}^{(n)}} \left( \prod_{k<i\leq n} \mathbf{\Phi}_{\mathrm{H}}^{'(i)} W_{\mathrm{rec}}^{T} \right) \frac{\partial \mathbf{h}^{(k)}}{\partial \theta}$$

$$\approx ||\mathbf{\Phi}_{\mathrm{H}}^{'}||^{n-k} ||W_{\mathrm{rec}}^{T}||^{n-k}$$

系列の長いデータを扱う時(n大の時)は
時間の展開を小さくする(kを大きくする)
ことで勾配爆発・消失を回避する

※ 勾配爆発・消失に対する詳しいアプローチは第10回の講義で解説

# RNNまとめ

- RNNでは過去の隠れ層の状態を入力に加えることで、時系列を保持している
- ネットワークを時間軸方向に展開して、BPを行う(BPTT)
- BPTTでは勾配爆発・消失問題がある
- 長期時系列データを扱うことは難しい
  ⇨ LSTM, GRU, Gradient Clipping (第10章)

# Work #9-1

人工データを用いたRNNの学習

講義 Part 2

# BPを使わないRNN

Echo State Network

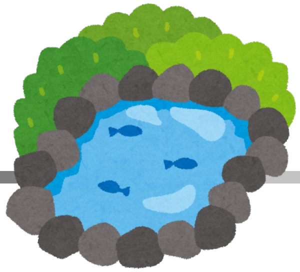
# 一般的なRNNの問題点

- RNNは計算コストが高すぎる
  - 勾配消失・爆発問題
  - LSTM, GRU, Gradient Clipping などの発明
  - それでも、学習が難しいのが現状

- そもそも、脳はBPTTなんてしているのか
  - 過去の入出力や結合荷重を保存して
    時間を遡って誤差を計算する...アヤシイ

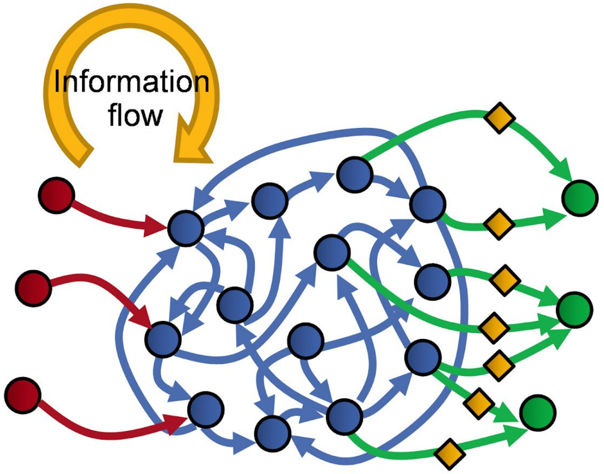
# Reservoir Computing

Reservoir: 貯水池 →



- 入力層, Reservoir層, 出力層からなる
- ランダムに結合したReservoir層が時系列を保持
- 学習は出力層の線形回帰だけ
  → 計算量が少ないため爆速
- 脳の神経モデル

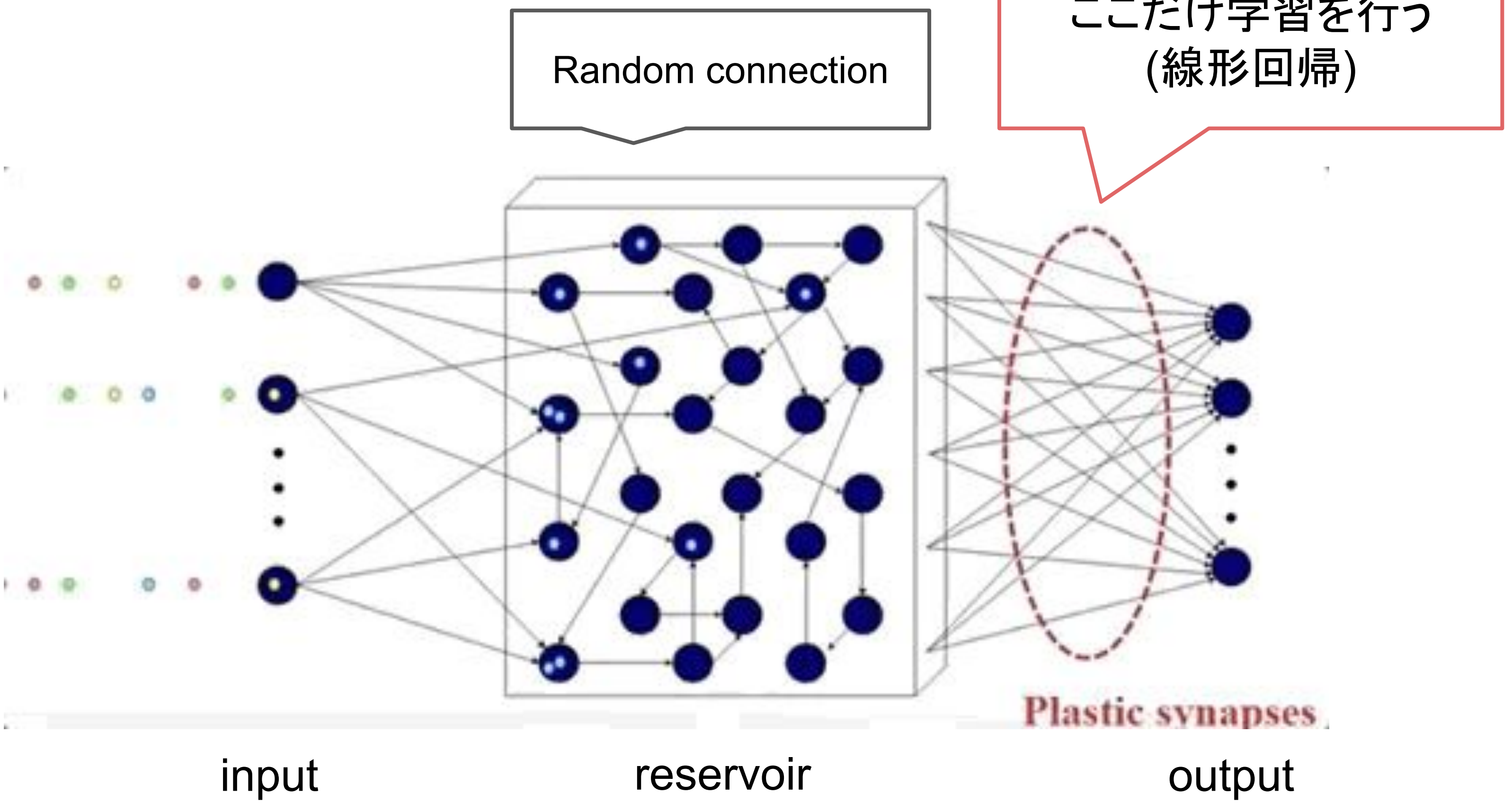
Reservoir computing



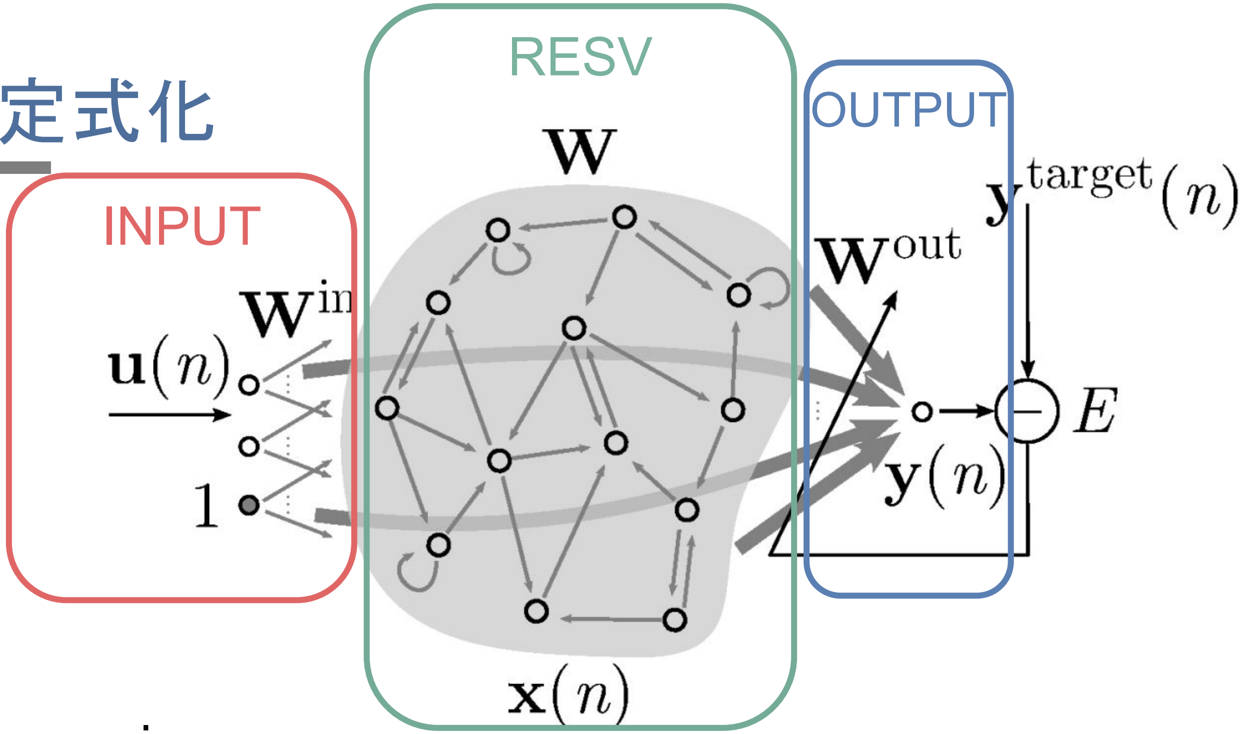

代表的なネットワーク:

- Echo State Networks (ESN)
- Liquid State Machine (LSM)

# Echo State Network



# ESNの定式化




Update: reservoir

$$\tilde{\mathbf{x}}(n) = \tanh\left(\boxed{\mathbf{W}^{\text{in}}[1;\mathbf{u}(n)]} + \mathbf{W}\mathbf{x}(n-1)\right),$$

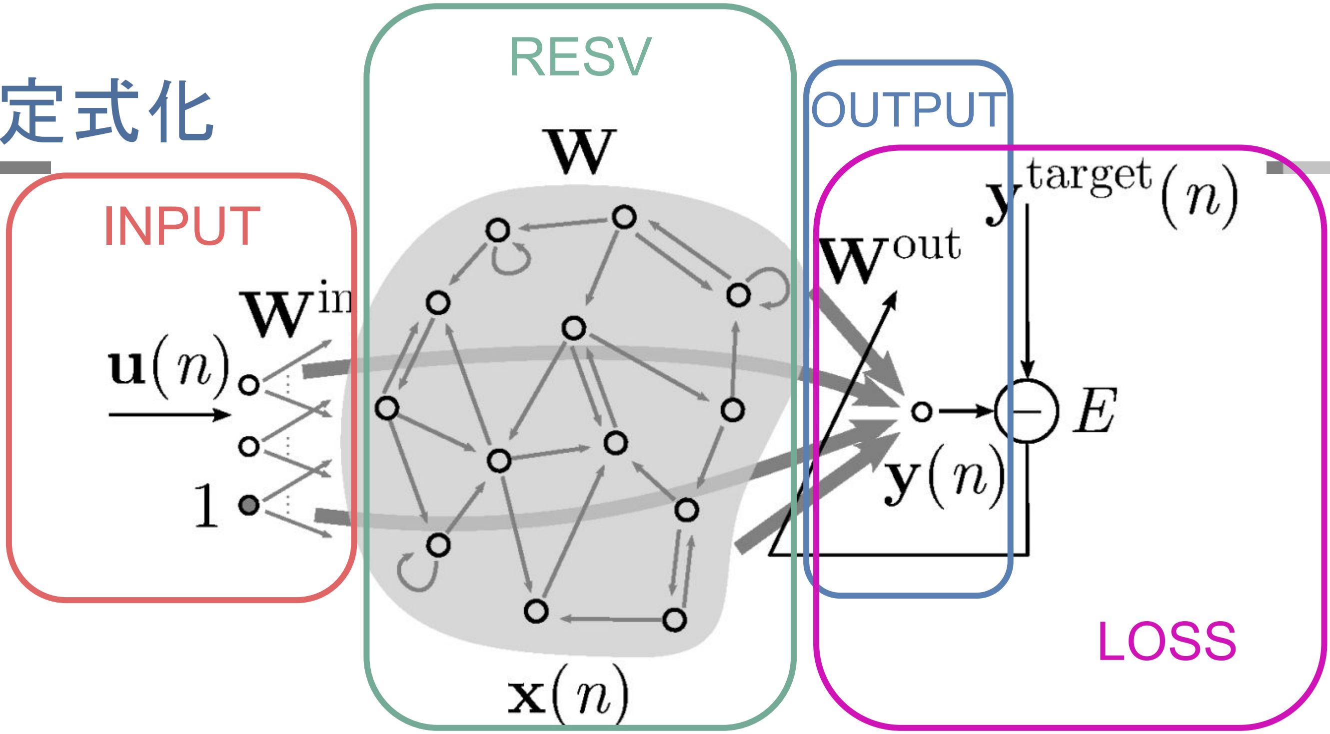
$$\mathbf{x}(n) = (1-\alpha)\mathbf{x}(n-1) + \boxed{\alpha\tilde{\mathbf{x}}}(n),$$

α: leaking rate
記憶の強さを調整

Compute: output

$$\mathbf{y}(n) = \mathbf{W}^{\text{out}}[1;\mathbf{u}(n);\mathbf{x}(n)],$$

# ESNの定式化




Compute: output

$$\mathbf{y}(n) = \mathbf{W}^{\mathrm{out}}[1; \mathbf{u}(n); \mathbf{x}(n)],$$

Compute: loss

$$E(\mathbf{y}, \mathbf{y}^{\mathrm{target}}) = \frac{1}{N_{\mathrm{y}}} \sum_{i=1}^{N_{\mathrm{y}}} \sqrt{\frac{1}{T} \sum_{n=1}^{T} \left(y_i(n) - y_i^{\mathrm{target}}(n)\right)^2}$$

RMSEを最小化
線形回帰で学習

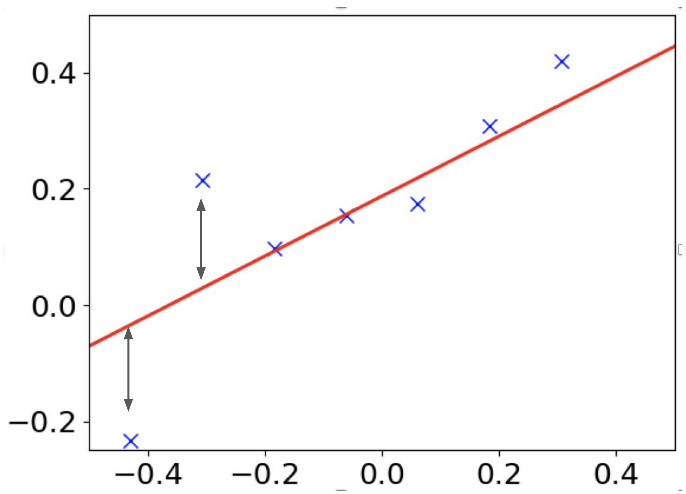
# ESNの学習: 線形回帰

L2正則化最小二乗回帰

最小二乗法の復習

- ESNの学習は通常リッジ回帰を使って行われる
  - 最も単純な方法では、すべての出力を用意して回帰計算を行う
  - リッジ回帰については『第三回: 線形回帰』で解説済み

今回は簡単のため、<u>単純線形回帰</u>で求めてみる




モデルの予測と正解の二乗和誤差が最小になるようなパラメータ**θ**を探す

$$L_{LS}(\boldsymbol{W}) = \frac{1}{2}\sum_{i=1}^{T}\left(f(y_i; \boldsymbol{W}) - y_i^{tgt}\right)^2$$

$$\boldsymbol{W}_{LS} = \mathrm{argmin}_{\boldsymbol{W}} L_{LS}(\boldsymbol{W})$$

# ESNの学習: 単純線形回帰

求めるWは自乗誤差Lを最小化するW

$$W_{LS}^{out} = \underset{W^{out}}{\arg\min} \overbrace{\frac{1}{2}||W^{out}X - Y^{tgt}||^2}^{L_{LS}(\boldsymbol{W})}$$

Wに関して2次関数の形 (凸関数) をしているので、 $L_{LS}(\boldsymbol{W})$ の偏微分=0を与えるWが求値

$$W_{LS}^{out} = Y^{tgt}X^T(XX^T)^{-1}$$

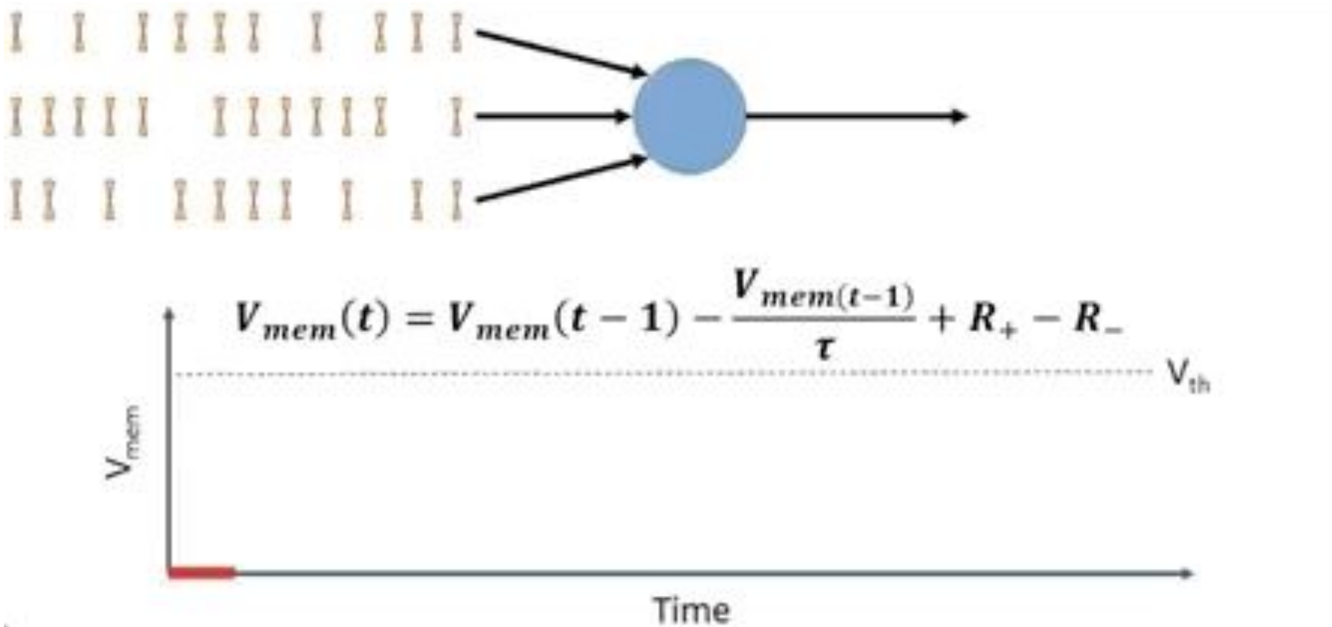
# Liquid State Machine

特徴

- ESNに類似したRCモデルの一つ
- 脳っぽさを謳っている
- ニューロン発火モデルとして、
  IAF model(積分発火モデル)を使用 (c.f. tanh for ESN)
- 連続的に変化するデータを扱うことができる
  (⇔ ESN: 離散的)
- ハードウェアと親和性が高く、VLSIなどでしばしば実装される

# LSM: IAFモデル

Integrate and Fire




時間を通してニューロンに電位が溜まっていき、

しきい値を超えた際に発火する (⇨積分発火モデル)

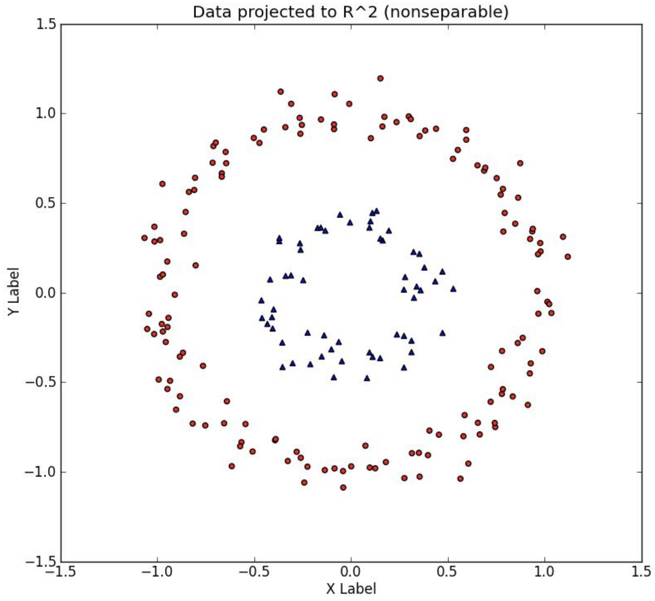
(c.f. ESNではtanhでactivationしていた)

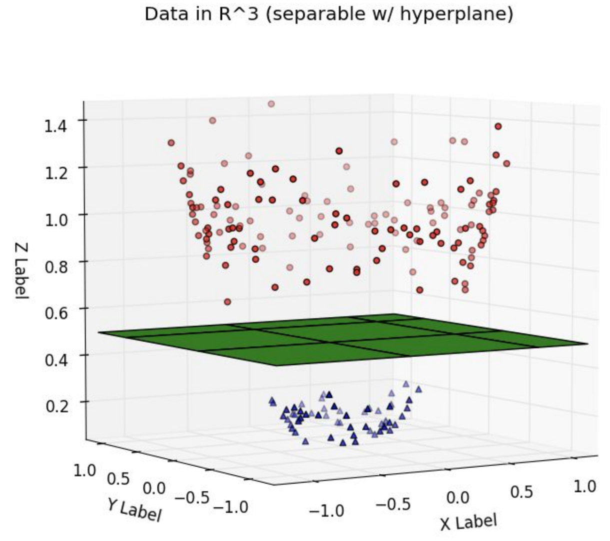
# なぜRCが上手くいくか

Kernel Trickとの類似性

## Kernel Trick:

サポートベクターマシン(SVM)で使用
線形分離不能なデータの次元を上げて
頑張って線形分離する手法




## Reservoir Computing:

- reservoir層で次元を拡張
  → Kernel Trickに相当

# RCまとめ

- RCではランダムに結合したreservoir層を使用
- Reader(output)では単純な線形回帰を使用
  - ⇨ 計算コスト低 (c.f. RNN)
- ２つのモデルを紹介
  - ESN: tanhを使用
  - LSM: スパイクモデルを使用
- 計算がうまくいくのはカーネルトリックと似た原理
- Reservoirの重みは基本的に学習しない
  - 学習するモデルも存在するが、
    あまりうまく行っていない模様

# Appendix

# 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


- [Reno911::18]https://www.youtube.com/watch?v=Ukgii7Yd_cU&index=4&list=PLdtFmJC5lu-AbZvwnTX7pQtsb6TpJqbiV&t=336s
- [Karpathy::18] (n.d.). Retrieved January 28, 2018, from http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- [Jaeger 07] Jaeger, H. (2007). Echo state network. *Scholarpedia*, *2*(9), 2330.
- [Lukoševičius 12] Lukoševičius, M. (2012). A practical guide to applying echo state networks. In *Neural networks: tricks of the trade* (pp. 659-686). Springer Berlin Heidelberg.
- [Maass+ 10] Maass, W. (2010). Liquid state machines: motivation, theory, and applications. *Computability in context: computation and logic in the real world*, 275-296.
- [Lukoševičius & Jaeger 09] Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. Computer Science Review, 3(3), 127-149.

# References

[Lecture] Evolution: from vanilla RNN to GRU & LSTMs
https://towardsdatascience.com/lecture-evolution-from-vanilla-rnn-to-gru-lstms-58688f1da83a
[Wu+ 16] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Klingner, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:1609.08144.