SPRINT22

# Sprint22

- 振り返り
- Word2Vec
- RNNs
  - 歴史的発展
    - 相互結合型タイプ
    - 回帰結合型タイプ
      - Elman
      - Jordan
- 勾配計算
- ネットワーク図
- 問題点

# BoWとは何だったか

| | a | bad | film | good | is | movie | this | very |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| **1** | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| **2** | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 3 |

```
(0, 3)        1
(0, 7)        1
(0, 4)        1
(0, 5)        1
(0, 6)        1
(1, 0)        1
(1, 2)        1
(1, 3)        1
(1, 4)        1
(1, 6)        1
(2, 1)        2
(2, 7)        3
```

BoWによる表現（TF-IDFを含む）は、文書と単語からなる行列でした。そして、一つの次元が一つの単語に対応するという関係から、それは巨大な疎行列になりかねません。

scikit-learnの「Feature extraction /4.2.3.2. Sparsity」ではこれを以下のように評しています。

「ほとんどの文書においては、コーパス（BoWの列のこと）のとても小さな単語集合を扱っているため、結果として得られるBoWの99%以上が0になってしまいます。例として10,000の短文の文書群を考えた際に、100,000の単語によって構成される一方で、一つの文書において使用する重複のないユニークな単語数は100~1,000ほどです。
この疎行列を扱うために、scipy.sparseのようなsparse representationを用います。」

https://scikit-learn.org/0.16/modules/feature_extraction.html#sparsity

# 局所表現と分散表現

前述のBoW的な表現は、**局所表現**（local representation）と呼ばれていました。これは、非零要素が１つあるいは非常に少ない非零要素からなる、ベクトルによって文章を表現するような手法でした。
これに対して、Word2Vecから抽出される表現は **分散表現**（distributed representation）と呼ばれ、**複数の次元が一つの単語を構成**し、また**同じ一つの次元が複数の単語を構成する**ために用いられるものです。
ある事象を説明するとき、分散表現は、他の事象と**概念を共有する多種多様な特徴の集まり**を用いて行います。

一方、局所表現は、**その事象が持つ少数あるいは１つの特徴的な要素**を指示していると言えます。
「分散表現」という用語は、認知心理学と神経科学、さらにニューラルネットの研究分野からの応用として1980年代にHintonらによって提案されたものがその起源と言われています[1]。
脳のモデル化を考える際に、事象や概念のような離散的な表現を複数の連続値の集まり（ベクトル）で表現する方法として提案されました。

[1] Geoffrey E. Hinton, Learning distributed representations of concepts. 1989
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.408.7684&rep=rep1&type=pdf

# Sprint22

- 振り返り
- Word2Vec
- RNNs
  - 歴史的発展
    - 相互結合型タイプ
    - 回帰結合型タイプ
      - Elman
      - Jordan
- 勾配計算
- ネットワーク図
- 問題点

# Word2Vec = Word to Vector

分散表現を生成する前処理用のニューラルネットワーク：Word2Vec

Word2Vecは、どのようにして分散表現を作るのでしょうか。

まず、入力に100,000ほどのOne-hot 表現（局所表現）を用い、200ほどノードを持つ隠れ層を通して次の単語を予測するモデルを作り、ある程度うまく予測できるようになったとします。
このとき、100,000個の単語の情報は、200個のノードつまり次元圧縮された潜在変数の空間にマッピングされたと捉えることができます。
言い換えると、ここの200個のパラメータは、100,000個の単語の情報を代替していると考えることができます。

ゆえに、この200個のパラメータを分散表現として用いることが可能だというのが、Word2Vecのアイデアです。

この分散表現は、深層学習の言語モデルの代表 RNNの入力データになります。
以降ではこのRNNs（多様な種類のRNN）について説明していきます。

# 深層学習の言語モデル：RNNsとは？

Recurrent Neural Networks

**回帰結合型ニューラルネットワーク**（recurrent neural networks, **RNNs**）とは、時系列のような**系列データ**の処理を目的とするニューラルネットワークの一種で、1980年代後半からその研究が始まりました。

これは従来のフィードフォワードニューラルネットワークと違い、ネットワークが**循環する構造**を持っています。

自然言語処理分野においても、現在もRNNsが用いられています。

Recurrentとは、英語で「循環する」という意味を持ちます。

循環する構造とはいかなる構造でしょうか？

以下では、RNNsの代表的なものとして、**相互結合型**と**回帰結合型**の循環構造を紹介します。

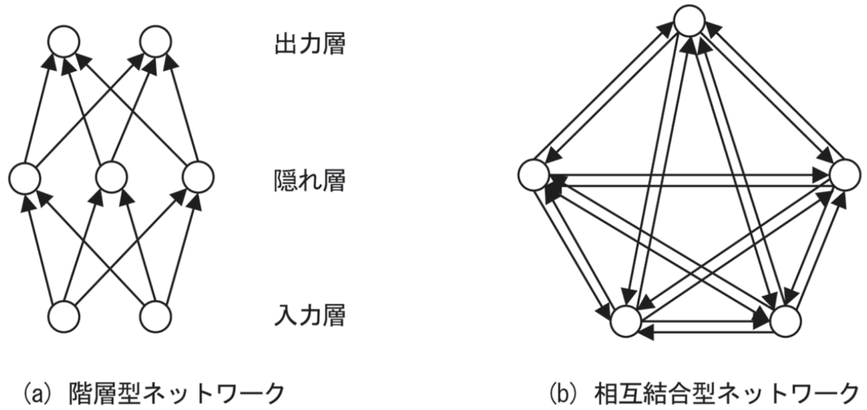
# Sprint22

- 振り返り
- Word2Vec
- RNNs
  - 歴史的発展
    - 相互結合型タイプ
    - 回帰結合型タイプ
      - Elman
      - Jordan
- 勾配計算
- ネットワーク図
- 問題点

# RNNs：相互結合型タイプ

fully interconnected network

ニューラルネットワークの構造例 [3]




1982年に提案された**Hopfieldネットワーク**[2]は循環構造を持っており、これは初期のRNNsとされています。その構造は**完全相互結合型**と呼ばれ、**自分以外のすべてのノード**と相互に接続しています。

Hopfieldは、学習によって複数の記憶パターンを保持することができます（記憶可能なパターン数はノード数の約0.15倍）。学習後の推定時には、未知の入力に含まれるノイズに対し、あらかじめ記憶されたものから最も類似したパターンを出力することが可能です。この機構は連想記憶モデルと呼ばれ、離散（組み合わせ）最適化問題に応用できることが知られています。

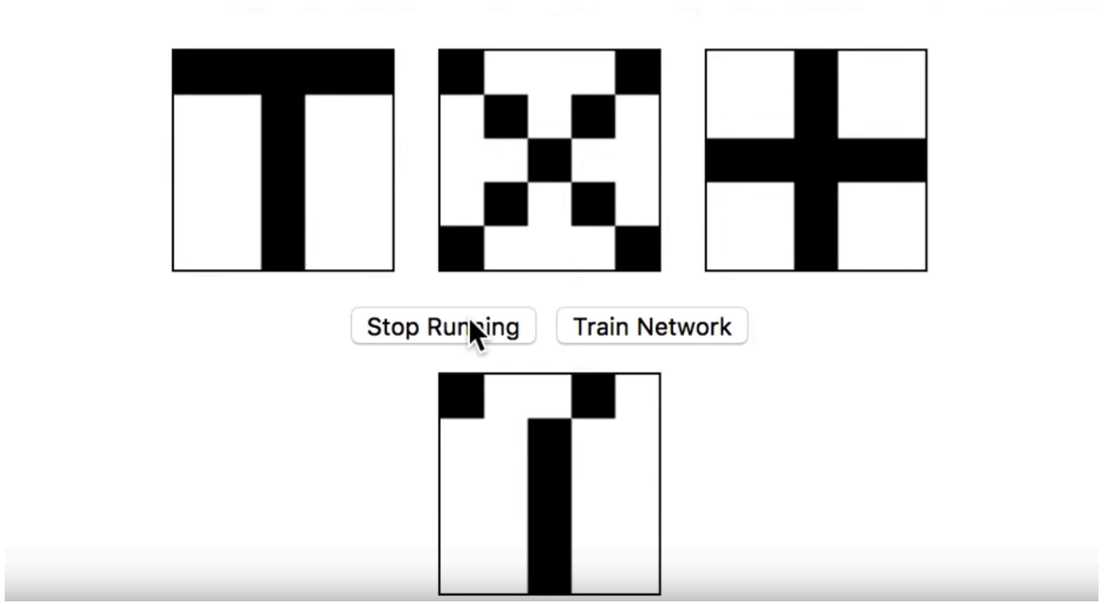
[2] John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. 1982
[3] Hirokazu IWASE The Influence of Parameter Settings on the Hopfield Neural Network. 2017

# RNNs：相互結合型タイプ

fully interconnected network

しかしながら、Hopfieldネットワークはその結合に**方向性がない**（無向グラフである）ことから、時系列データには適していませんでした。

時系列を取り扱うには、過去の情報から未来の情報を推定し、その情報でさらに次の未来の情報を推定するという、時間的な方向性を持つネットワークが必要になります。




https://www.youtube.com/watch?v=HOxSKBxUVpg

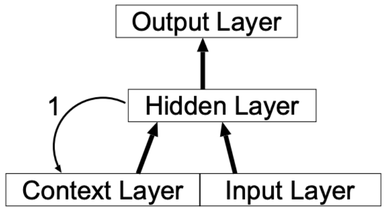
# Sprint22

- 振り返り
- Word2Vec
- RNNs
  - 歴史的発展
    - 相互結合型タイプ
    - 回帰結合型タイプ
      - Elman
      - Jordan
- 勾配計算
- ネットワーク図
- 問題点

# RNNs：回帰結合型タイプ

simple recurrent networks

エルマンネット [5]




1990年に**Elman**が考案した**単純回帰結合型**（simple reccurent networks）ネットワーク[4]は、**一方向の回帰的な結合**（有向グラフ）によって、循環構造を成立させます。
Hopfieldネットワークと違って、**結合の形状に方向性がある**ことから、Elmanのネットワークは時系列データに適用することができます。
なぜなら、一方向つまり過去からの情報を回帰結合から受け取ることができるためです。

その基本構造は、従来のフィードフォワードニューラルネットワークと同じ階層型ネットワーク（入力層から中間層、中間層から出力層への結合をもつ）であり、さらに**直前の中間層の状態を保持**する（コピーする）文脈層が加わった、3層からなるネットワークです。
文脈層はある時点での情報をコピーして中間層へ渡し（同じ情報を出力層へも渡す）、次の時点では入力層からの情報とともに中間層にその情報を渡すことから、**ひとつ前の時刻の自分自身（中間層）からの接続をもつ**という循環構造を実現しています。

[4] Jeffrey L.Elman. Finding structure in time. 1990
[5] 浅川伸一 エルマンネットの応用可能性 2009

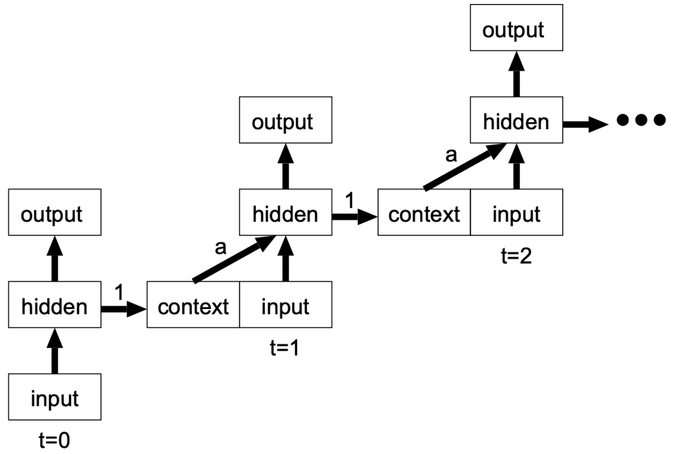
# RNNs：回帰結合型タイプ

simple recurrent networks

ある時点 t で生じる内容とは、その時点での入力と、それ以前の時点 t−1 までに処理されたネットワークの状態を同時に処理することです。

中間層（文脈層）は、**t−1 時点までの過去の状態を保持**していると解釈できるので、t 時点でのネットワークの状態は、現在の入力と過去の入力履歴の全体によって推定されるといえます。

エルマンネットの時間発展 [5]




文脈層からの結合の加重が1より小さい場合は、過去からの入力履歴の影響が指数関数的に小さくなり、1より大きい場合は、過去からの入力履歴の影響が指数関数的に大きくなります。

# Sprint22

- 振り返り
- Word2Vec
- RNNs
  - 歴史的発展
    - 相互結合型タイプ
    - 回帰結合型タイプ
      - Elman
      - Jordan
- 勾配計算
- ネットワーク図
- 問題点

# RNNs：回帰結合型タイプ

simple recurrent networks

また、Elmanネットワークと同じ単純回帰結合型のネットワークに、1986年に発表された**Jordan**ネットワーク[6]があります。Elmanネットワークとの違いは、**中間層の代わりに出力層から**文脈層（Jordanでは状態層と呼ばれる）へ過去の履歴が入力される点です。

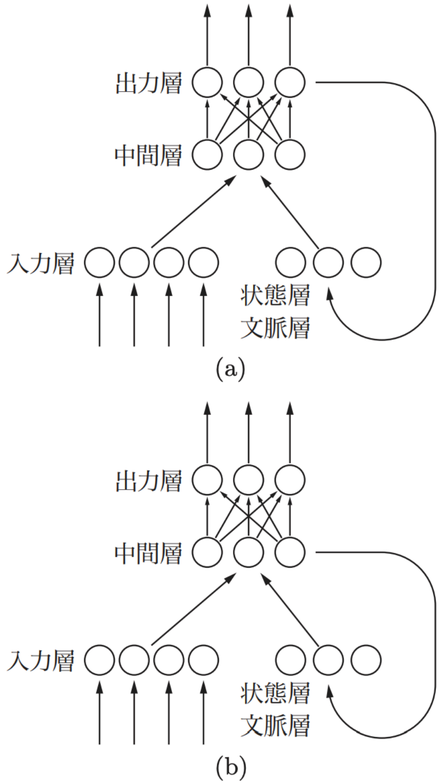



**図 2** (a) ジョーダンネットワークと，(b) エルマンネットワーク

https://www.cis.twcu.ac.jp/~asakawa/2019cnps_handson/6657.pdf

[6] Michael I. Jordan. Serial Order: A Parallel Distributed Processing Approach. 1986

# RNNs：回帰結合型タイプ

simple recurrent networks

Recurrent Hidden Units

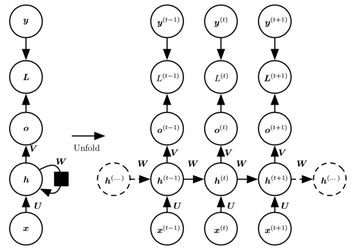


Figure 10.3

Recurrence through only the Output

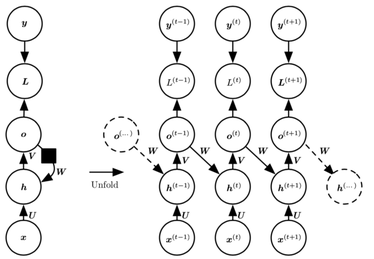


Figure 10.4

ある t-1 時点の出力層は、系列データのターゲット（t時点の情報）と一致するように訓練されるため、出力層のもつ情報は、t-1 時点の情報というよりは、t 時点の予測に関する情報だと考えられます。

例えば、自然言語処理タスクにおいて、文脈層は過去の履歴を入力として受け取りたいところですが、Jordanネットワークの場合、引き渡されるコピーは出力層のもののため、過去（t時点）の履歴を十分に保持していません。

これゆえに、自然言語処理への応用は、Jordanネットワークよりも**中間層の回帰的な接続のあるElmanネットワークが適して**います。

一方、t 時点の出力層の情報を用いることは、ロボットアームや姿勢、運動などの制御には有用であるので、Jordanネットワークは、動作制御への応用に適しているといわれます。

# Sprint22

- 振り返り
- Word2Vec
- RNNs
  - 歴史的発展
    - 相互結合型タイプ
    - 回帰結合型タイプ
      - Elman
      - Jordan
- 勾配計算
- ネットワーク図
- 問題点

# RNNs：勾配計算

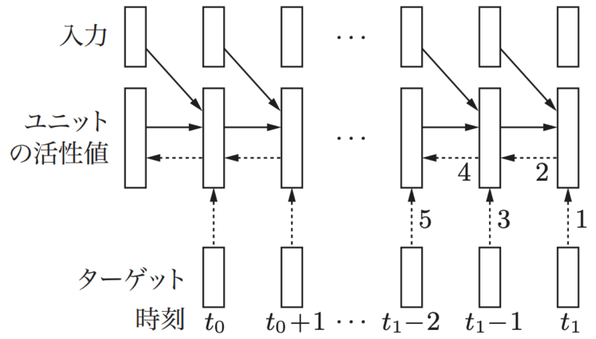
フィードフォワードニューラルネットワークのような階層型ネットワークの有名な学習則に、後ろ向き自動微分といわれる**誤差逆伝播（error Back Propagation, BP）法**があります。

BP法が発表された論文の中で、単純回帰結合型ネットワークは、中間層に入力情報が増えたことを除けば、回帰結合のない通常のフィードフォワードニューラルネットワークと相違はないため、**一般化した誤差逆伝播法**により学習が可能であると示唆されました[7]。

それは、**結合荷重を固定**し、**出力の伝播を時間方向に展開することで、階層化が可能である**（ネットワーク全体の時間ステップをTとした場合、各時点におけるノードから構成されるT層のニューラルネットワークとみなすことができる）という発想に基づいています。

[7] David E. Rumelhart, Geoffrey E. Hinton and Ronald J. Williams. Learning representations by back-propagating errors. 1986

# RNNs：勾配計算



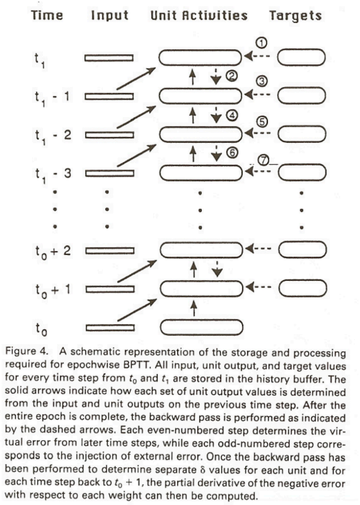




Figure 4. A schematic representation of the storage and processing required for epochwise BPTT. All input, unit output, and target values for every time step from $t_0$ and $t_1$ are stored in the history buffer. The solid arrows indicate how each set of unit output values is determined from the input and unit outputs on the previous time step. After the entire epoch is complete, the backward pass is performed as indicated by the dashed arrows. Each even-numbered step determines the virtual error from later time steps, while each odd-numbered step corresponds to the injection of external error. Once the backward pass has been performed to determine separate δ values for each unit and for each time step back to $t_0 + 1$, the partial derivative of the negative error with respect to each weight can then be computed.

https://web.stanford.edu/class/psych209a/ReadingsByDate/02_25/Williams%20Zipser95RecNets.pdf

このような一般化として、系列情報を一定の時間窓のあいだは保持し、その時間窓内の情報について学習を行うアルゴリズムとして、**BPTT**（back-propagation through time; 通時的誤差逆伝搬）、および**RTRL**（real time recurrent learning; リアルタイム回帰学習）が提案されました。

通常のBP法では、最上層でのみ誤差を算出しますが、BPTT法では、**すべての時間において誤差の算出が行われます**。これはつまり、最上層の誤差を伝播するだけでなく、各時点の層において、**上層の誤差とその層の誤差を足し合わせる**必要があるので、中間層のノード数がN、ネットワーク全体の時間ステップをTとする場合、計算量のオーダーは O(TN^2) で、メモリのオーダーは O(TN) となることを意味します。

一方、RTRL法は時間を遡る必要はなく、リアルタイム（ t 時点で発生した誤差で、次の時点 t+1 の重みを更新）で学習することが可能ですが、計算量のオーダーが　O(TN^4)、メモリのオーダーが O(TN^3) と、さらに大きくなります。このような計算量を抑えるための提案手法も研究されています。

# RNNs：学習則

余談

相互結合型のHopfieldネットワークで用いられる学習則は、ヘブ則(Hebbian rule; 外積則ともいう) [8]といい、脳のシナプスの可逆性を模倣した規則を基にしています。

これは、連続する前後のノードの発火が起きれば、それを接続する結合荷重の値が大きくなって伝達効率が上がる一方、発火が起きなければ伝達効率が減衰するという仕組みを実現します。これに対し、デルタ則を一般化したBP法は、工学的には有用な手法ですが、脳内で実際にそのような学習機構が可能かどうかについては研究途上の段階にあります。

[8] Hebb, D. O. The organization of behavior. 1949

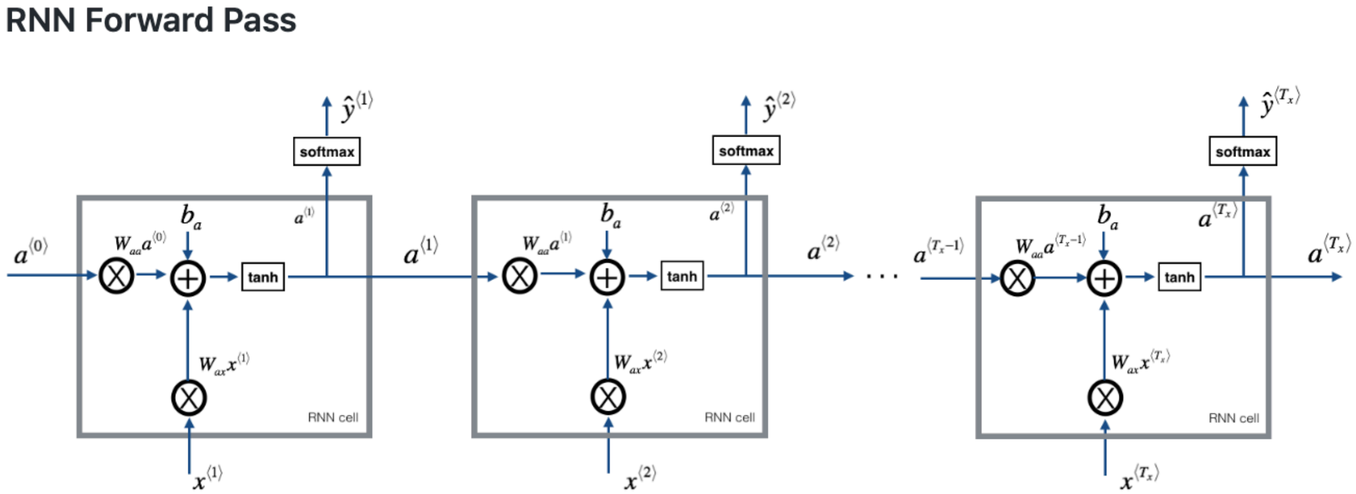
# Sprint22

- 振り返り
- Word2Vec
- RNNs
  - 歴史的発展
    - 相互結合型タイプ
    - 回帰結合型タイプ
      - Elman
      - Jordan
- 勾配計算
- ネットワーク図
- 問題点

# simple reccurent networks
# ：ネットワーク図

現在、RNNsを代表するsimple reccurent networks（Elmanベース）のフィードフォワードは、右図のように表すことができます。

前方の四角い範囲（t-1 時点）から後方の四角い範囲（t 時点）へ t-1 時点の状態がパスされています。



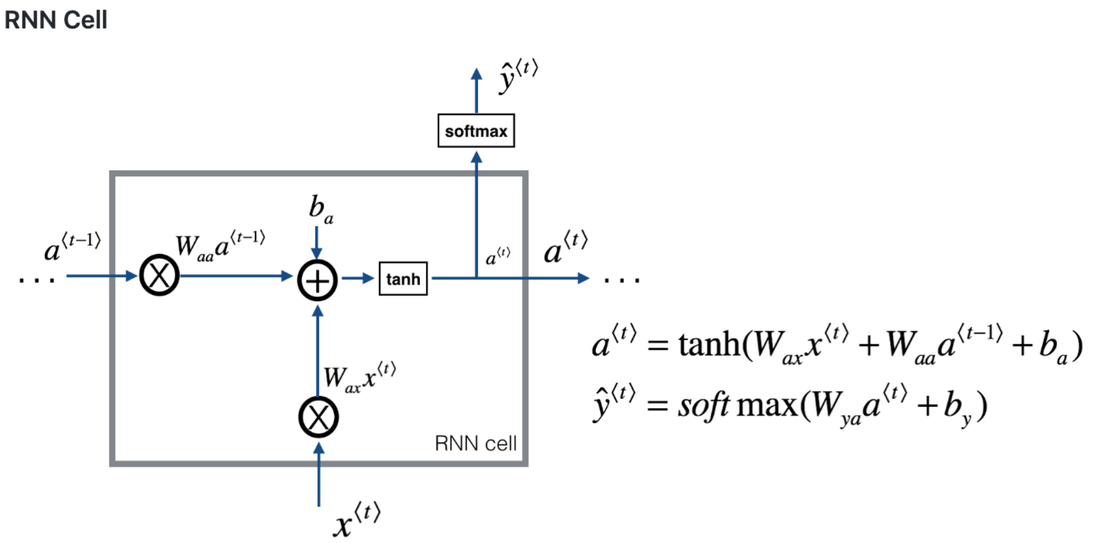
# simple reccurent networks
# ：ネットワーク図

四角い範囲の並びは、同一のネットワーク（**セル**と呼ぶ）を時間軸方向に展開したものです。

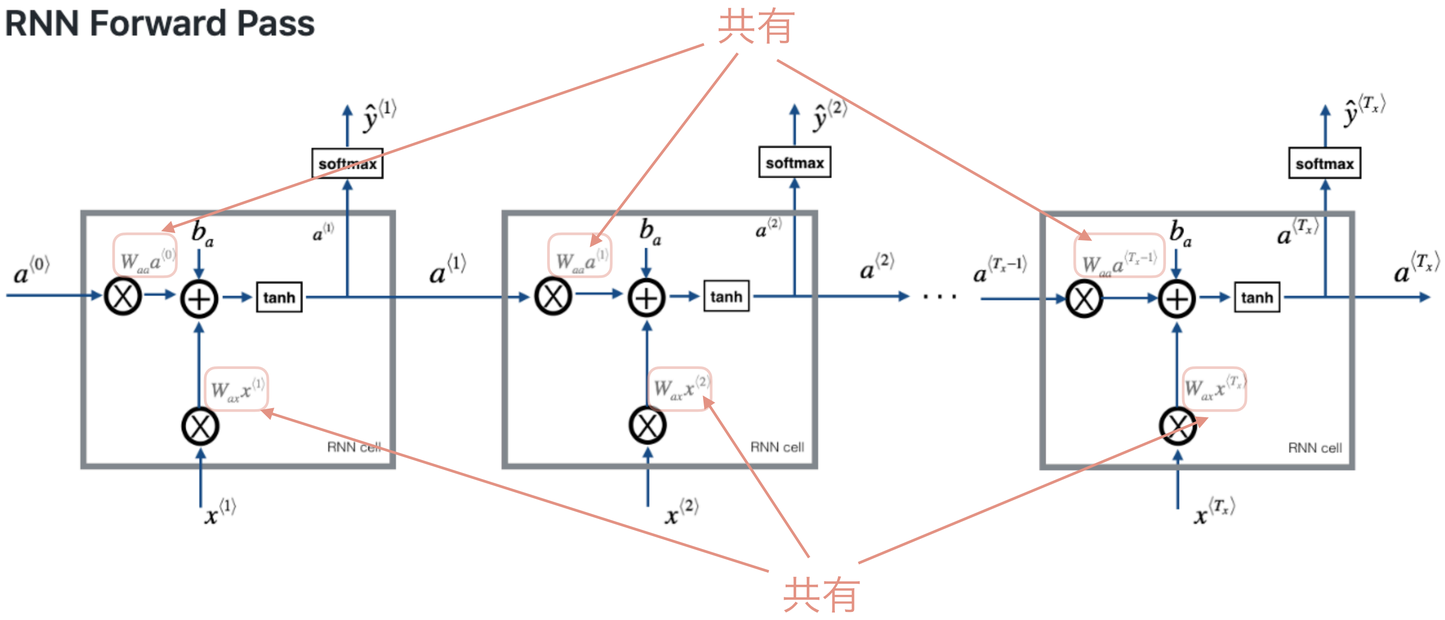
同じネットワークで演算された出力（a (t-1)）を自分の中へ入力（a (t)）し、再び演算することを反復します。

このような循環するループを持つゆえに回帰的なネットワークと呼ばれます。

二種類の重み、W_axとW_aaは、それぞれの時間において**共有**されます。



# simple reccurent networks
## ：ネットワーク図



# Sprint22

- 振り返り
- Word2Vec
- RNNs
  - 歴史的発展
    - 相互結合型タイプ
    - 回帰結合型タイプ
      - Elman
      - Jordan
- 勾配計算
- ネットワーク図
- 問題点

# 問題点：長期の系列を学習できない

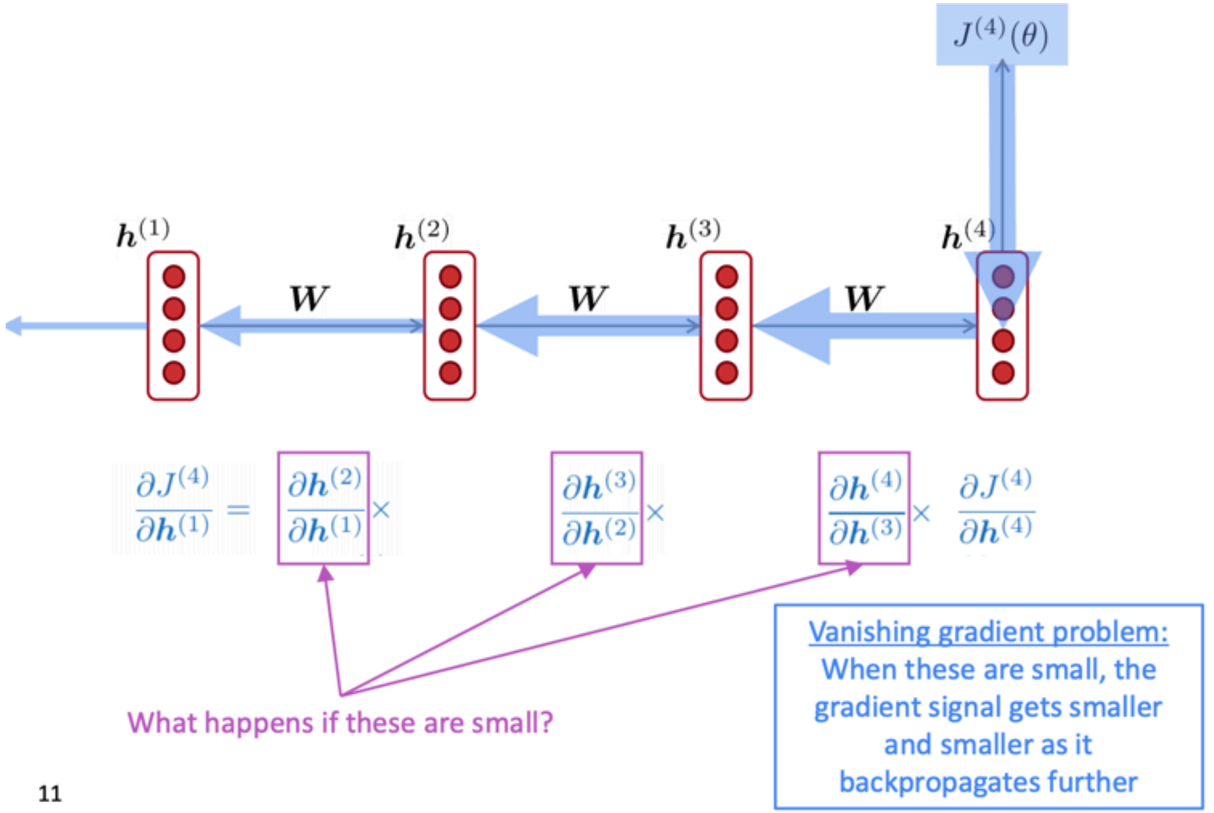
RNNは、理論上「長期的な(long-term)依存関係」を処理することが可能と考えられていました。

ところが、数10ステップの短期 (short-term) 依存には対応できても、1000ステップのような**長期の系列を学習することは困難**とされています。

# 原因：勾配消失 or 勾配爆発

RNNに限らず、層の深いDNNでは、多くの状況において勾配消失あるいは爆発を引き起こしやすいと言われています。

RNNにこれらが起こりやすいのは、**どのレイヤでも同じ重みを共有**しているためです[9]。




https://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture07-fancy-rnn.pdf

[9] Razvan Pascanu, Tomas Mikolov, Yoshua Bengio, On the difficulty of training Recurrent Neural Networks. 2013

# 重み共有：BPTTにおいて同じ重みWを何回もかけること

t ステップ後は$W^t$ になります。この$W^t$はどんな値になるか考えてみよう。

例えば、行列Wについて固有値分解（固有ベクトルと固有値からなる対角行列に分解）を施すと以下のように分解することができます。

$$W = V \times diag(\lambda) \times V^{-1}$$

diag(λ)とは、固有値からなる対角行列（非対角要素がすべて0となる行列）です。このとき、対角要素である 固有値をλiと表せます。

このとき、行列W の t 乗は以下のようになります。

$$W^t = V \times diag(\lambda)^t \times V^{-1}$$

[参考] 対角行列を累乗する計算

○ $P^{-1}AP$ の形の式の特徴

・ $P^{-1}AP=\begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_3 \end{pmatrix}$ のとき

左辺のｎ乗については，

$(P^{-1}AP)^n=P^{-1}A^nP$ が成り立つ．

右辺のｎ乗については

$\begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_3 \end{pmatrix}^n=\begin{pmatrix} {\lambda_1}^n & & 0 \\ & {\lambda_2}^n & \\ 0 & & {\lambda_3}^n \end{pmatrix}$ が成り立つ．

したがって，

$P^{-1}A^nP=\begin{pmatrix} {\lambda_1}^n & & 0 \\ & {\lambda_2}^n & \\ 0 & & {\lambda_3}^n \end{pmatrix}$

両辺に左から $P$ を，右から $P^{-1}$ を掛けると

$A^n=P\begin{pmatrix} {\lambda_1}^n & & 0 \\ & {\lambda_2}^n & \\ 0 & & {\lambda_3}^n \end{pmatrix}P^{-1}$

が求まる．

※ 対角行列$\begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \lambda_3 \end{pmatrix}$を$D$ で表すとき，次のように変形してもよい．

$P^{-1}AP=D \Rightarrow A=PDP^{-1}$

のとき

$A^n=(PDP^{-1})^n=PD^nP^{-1}=P\begin{pmatrix} {\lambda_1}^n & & 0 \\ & {\lambda_2}^n & \\ 0 & & {\lambda_3}^n \end{pmatrix}P^{-1}$

# 重み共有：BPTTにおいて同じ重みWを何回もかけること

絶対値

$W^t$の最大固有値 < 1 のとき、**指数関数的に勾配が縮小する**

$W^t$の最大固有値 > 1 のとき、**指数関数的に勾配が発散する**