# 7 再帰型ニューラルネット

## 7.1 系列データの分類

- 系列データ：個々の要素が順序付きの集まりとして与えられるデータ: $\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3, \ldots, \boldsymbol{x}^T$
- 系列の長さ$T$は一般的に可変
- データの並びをインデックス$t = 1, 2, 3, \ldots$で表し、以降$t$を便宜上、時刻と呼ぶ
- 個々の要素が順序を持ち、しかもその並びに意味が隠れているようなデータ全般が対象となる

---

## 7.2 RNNの構造

> **再帰型ニューラルネット**(RNN)：内部に(有向)閉路を持つニューラルネットの総称

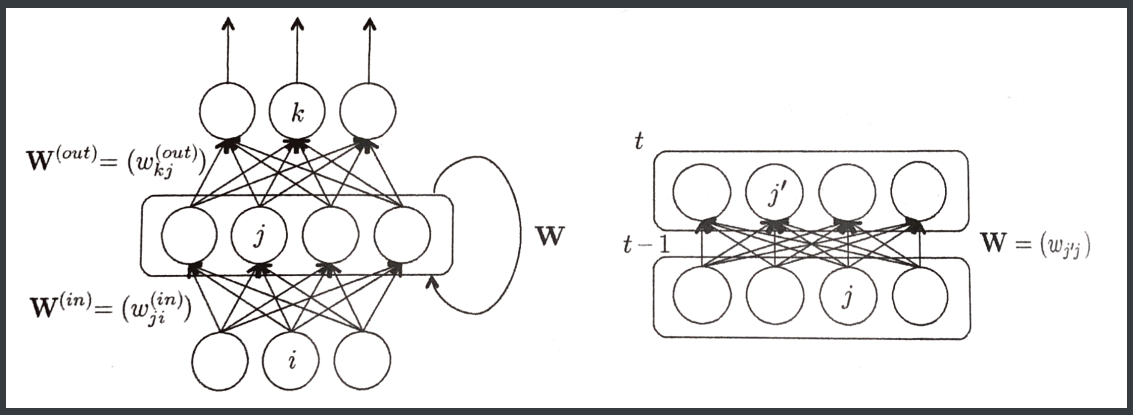
この構造のおかげで、情報を一時的に記録し、また振る舞いを動的に変化させることが可能になる

| RNNの種類 | 説明 |
| --- | --- |
| Elmanネット | ... |
| Jordanネット | ... |
| 時間遅れネット(time delay-) | ... |
| エコー状態ネット(echo state) | ... |
| 単純再帰型ネット(Simple RNN) | 順伝播型ネットワークと同様の構造を持ち、ただし中間層のユニットの出力が自分自身に戻される「帰還路」をもつシンプルなRNN. |
| LSTM | Simple RNNでも理論上は上手くいきますが、現実的にはかなり前の古い情報を考慮するようには学習されません.他のDeep NeuralNetworksと同様に、勾配消失の問題に直面したからです. LSTMは、従来のRNNセルでは長期依存が必要なタスクを学習することができなかった問題を解決したモデルです. |
| GRU(Gated Recurrent Unit) | LSTMをもう少しシンプルにしたモデルです。入力ゲートと忘却ゲートを「更新ゲート」として１つのゲートに統合しています. |
| 双方向性RNN(bidirectional RNN) | 双方向性RNNは、過去の情報だけでなく、未来の情報を加味することで精度を向上させるためのモデルです. 一般的なRNNでは、過去から未来のみの情報で学習しますが、双方向性RNNは未来から過去の方向でも同時に学習します. |

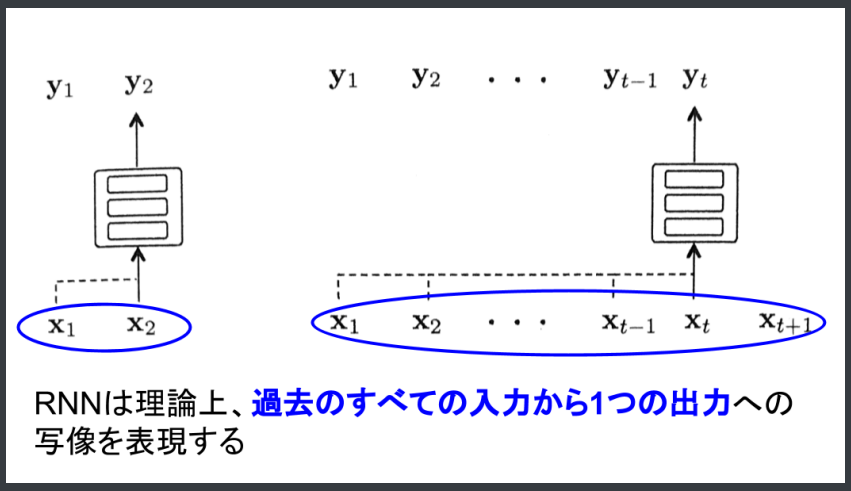
ここでは、以下の図のように、単純再帰型ネットを考える

## 単純再帰型ネット(Simple RNN)

### モデル




- ↑のRNNの動作：各時刻$t$につき1つの入力$\boldsymbol{x}^t$を受け取り、また同時に1つの出力$\boldsymbol{y}^t$を返す
- ネットワーク内部にある帰還路によって、出力を計算する際、RNNが過去に受け取ったすべての入力(=入力の履歴)が関与する




**出力層**

- 順伝播型ネットワーク同様に設計できる.
- ex) 分類問題：ソフトマックス関数, シグモイド関数

**誤差関数**

- 出力系列：$\mathbf{y}^1, \ldots, \mathbf{y}^T$
- 目標となる系列$\mathbf{d}^1, \ldots, \mathbf{d}^T$

目標系列と入力系列のペアからなる訓練データに対し、

$$E(\mathbf{w}) = \frac{1}{N} \sum_n \sum_t \ell\left(d_n^t, f^t\left(\mathbf{x}_n; \mathbf{w}\right)\right)$$

とする。

- $\ell(\cdot) \geq 0$: 個々の事例データに対して定義する誤差関数, 損失関数
- $d_n^t$ : $n$番目のサンプルの時刻$t$での**目標出力**
- $f^t(\mathbf{x}_n)$ : 目標出力と比較されるRNNの**出力関数**
- $\mathbf{x}_n$ : $n$番目のサンプルの**入力系列**
- 系列の長さはサンプル$n$ごとに違っていて構わない

---

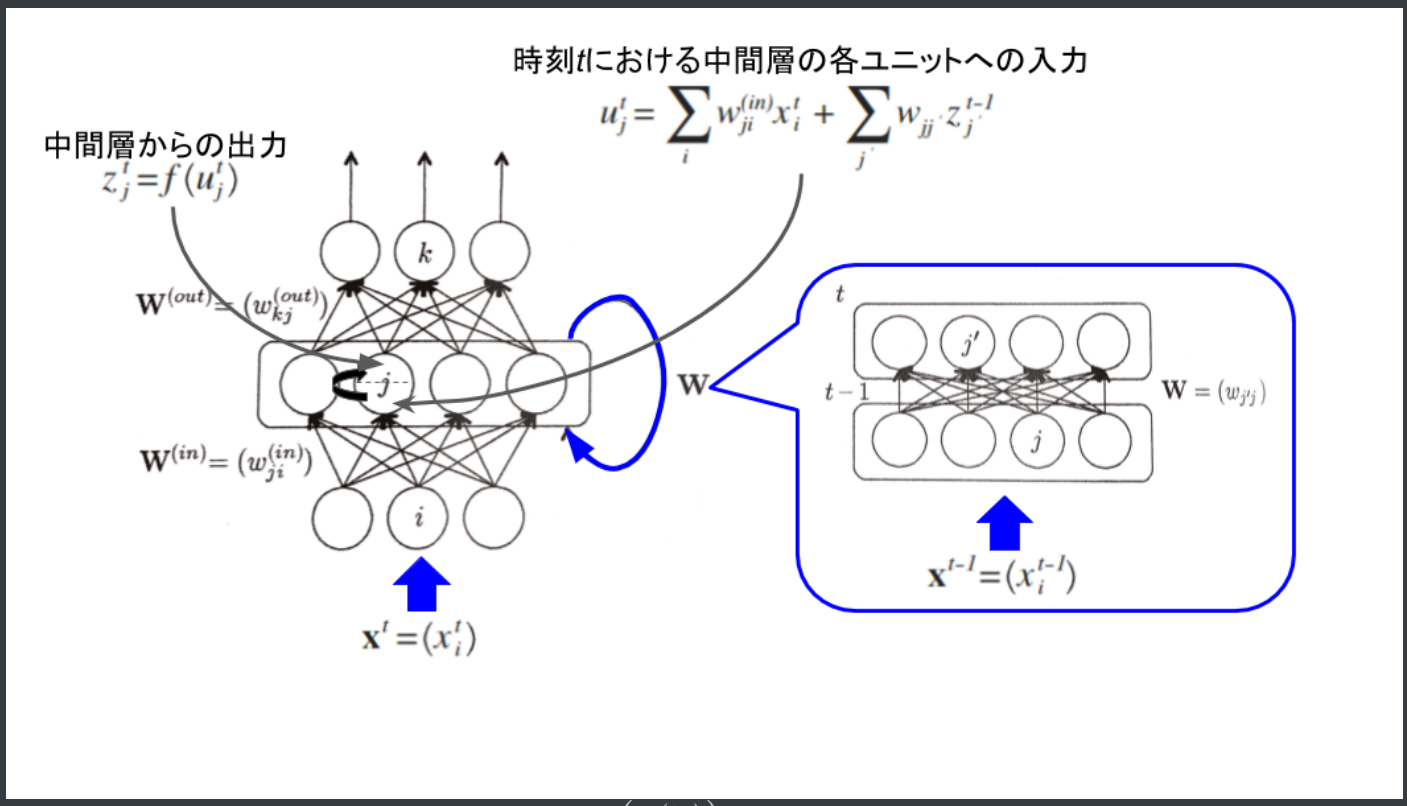
## 7.3 順伝播計算

入力系列から出力系列を得る計算手順を考える.

- ネットワークへの入力 : $\mathbf{x}^t = \left(x_i^t\right)$
- 中間層ユニットへの入力 : $\mathbf{u}^t = \left(u_j^t\right)$
- 中間層ユニットの出力 : $\mathbf{z}^t = \left(z_j^t\right)$
- 出力層ユニットへの入力 : $\mathbf{v}^t = \left(v_k^t\right)$
- 出力層ユニットの出力 : $\mathbf{y}^t = \left(y_k^t\right)$
- 目標出力 : $\mathbf{d}^t = \left(d_k^t\right)$

RNNの帰還路は、中間層の出力を自らの入力に戻すが、この間の結合は全ユニット間で存在する。下の図のように時刻$t-1$中間層の任意のユニット$j'$から時刻$t$中間層の任意のユニット$j$へ、重み$w_{jj'}$の結合が存在する。

重要なことは**この帰還が、時刻を1つ隔てて行われること**




- 入力層と中間層間の重み: $\mathbf{W}^{(in)} = \left(w_{ji}^{(in)}\right)$
- 中間層から中間層への帰還路の結合の重み : $\mathbf{W} = \left(w_{jj'}\right)$
- 中間層と出力層間の重み : $\mathbf{W}^{(out)} = \left(w_{kj}^{(out)}\right)$
- ※重みは時刻$t$とは関係なく、(学習によって更新はされるものの)順伝播計算中は定数であることに注意

### 各ユニットへの入力

上図から時刻$t$における中間層の各ユニットへの入力は、同時刻$t$にて入力層から届くものと、時刻$t-1$の中間層の出力をフィードバックしたものとの和になる。

$$u_j^t = \sum_i w_{ji}^{(in)} x_i^t + \sum_{j'} w_{jj'} z_{j'}^{t-1}$$

### 各ユニットの出力

通常の活性化関数$f$を経由して$z_j^t = f\left(u_j^t\right)$と計算される。まとめると、

$$\begin{bmatrix} u_1^t \\ \vdots \\ u_j^t \\ \vdots \\ u_J^t \end{bmatrix} = \begin{bmatrix} w_{11}^{(in)} & \cdots & w_{1i}^{(in)} & \cdots & w_{1I}^{(in)} \\ \vdots & \ddots & & & \vdots \\ w_{j1}^{(in)} & \cdots & w_{ji}^{(in)} & \cdots & w_{jI}^{(in)} \\ \vdots & & & \ddots & \vdots \\ w_{J1}^{(in)} & \cdots & w_{Ji}^{(in)} & \cdots & w_{JI}^{(in)} \end{bmatrix} \begin{bmatrix} x_1^t \\ \vdots \\ x_i^t \\ \vdots \\ x_I^t \end{bmatrix} + \begin{bmatrix} w_{11} & \cdots & w_{1j'} & \cdots & w_{1J'} \\ \vdots & \ddots & & & \vdots \\ w_{j1} & \cdots & w_{jj'} & \cdots & w_{jJ'} \\ \vdots & & & \ddots & \vdots \\ w_{J1} & \cdots & w_{Jj'} & \cdots & w_{JJ'} \end{bmatrix} \begin{bmatrix} z_1^{t-1} \\ \vdots \\ z_{j'}^{t-1} \\ \vdots \\ z_{J'}^{t-1} \end{bmatrix}$$

$$\mathbf{z}^t = \mathbf{f}\left(\mathbf{W}^{(in)}\mathbf{x}^t + \mathbf{W}\mathbf{z}^{t-1}\right)$$

となる。

- $t=1$から始め、$t$を1つずつ増やしながら、入力系列$\mathrm{x}^1, \mathrm{x}^2, \ldots$を使って、上式を繰り返し計算することで、任意の時刻$t$における中間層の状態$\mathbf{z}^t$を求めることができる
- ただし、$t=1$における初期値$\mathbf{z}^0 = \left(z_j^0\right)$を与える必要があり、通常は$z_j^0$とする

**出力層**

RNNの出力$\mathbf{y}^t$は次のように計算する.まず出力層の各ユニットへの入力は、中間層の出力$\mathbf{z}^t$から

$$v_k^t = \sum_j w_{kj}^{(out)} z_j^t$$

と決まる.なお、出力層の活性化関数は、順伝播型ネットワーク同様、適用したい問題によって選ぶ。活性化関数を$\mathbf{f}^{out}$と書くと、以上をまとめて

$$\mathbf{y}^t = \mathbf{f}^{out}\left(\mathbf{v}^t\right) = \mathbf{f}^{out}\left(\mathbf{W}^{(out)}\mathbf{z}^t\right)$$

と書くことができる

---

# 7.4 逆伝播計算

> RNNの学習には、順伝播型ネットワーク同様に確率的勾配降下法が使われる.

RNNの各層の重みについての誤差の微分を計算する方法

| 誤差微分の計算方法 | 説明 |
| --- | --- |
| **RTRL法**(realtime recurrent learning) | メモリ効率がよい |
| **BPTT法**(backpropagation through time) | 計算速度が速い&シンプル |

### BPTT法

P.117~P.120「7.4 逆伝播計算」『機械学習プロフェッショナルシリーズ　深層学習』を参照

---

# 7.5 長・短期記憶(LSTM)

### RNNの勾配消失問題

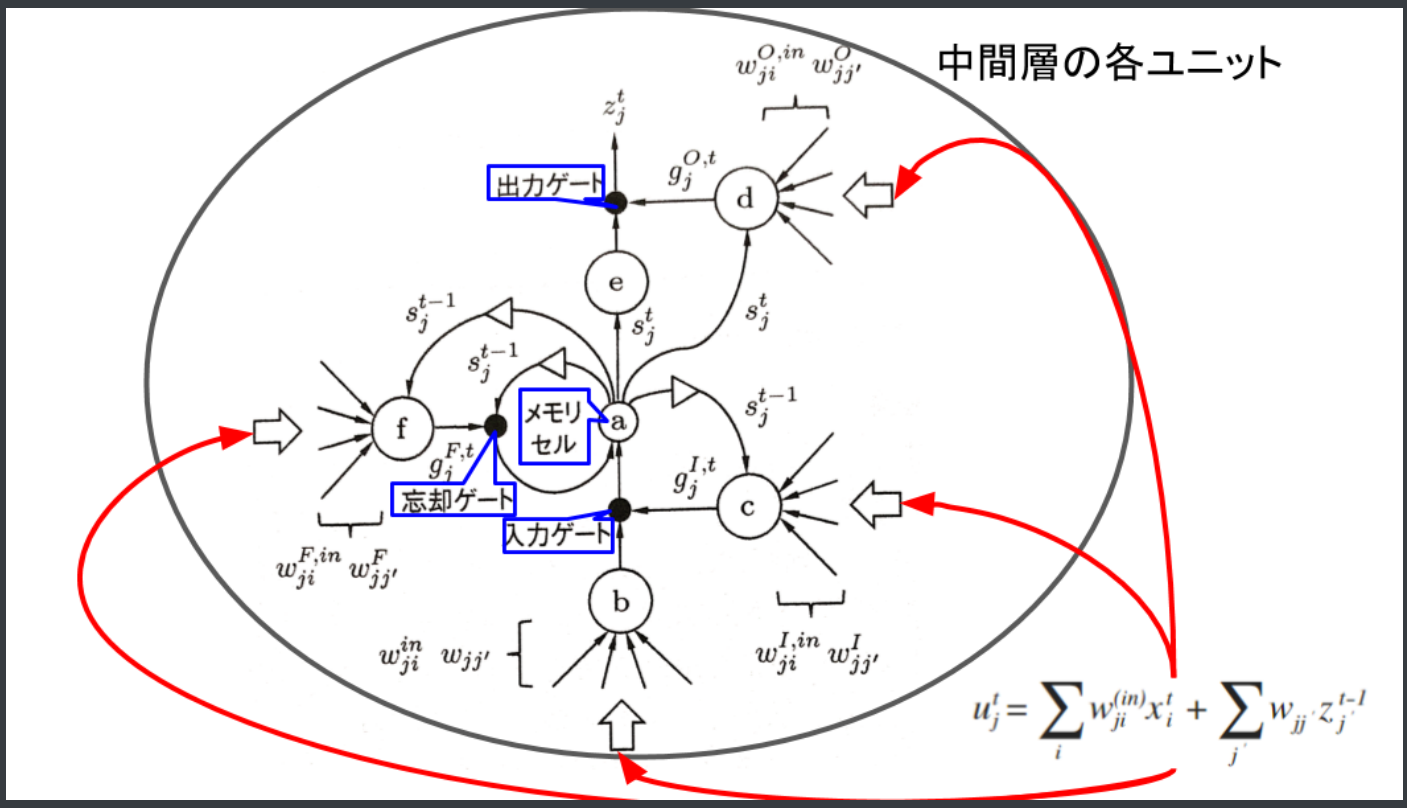
- 系列データの文脈を捉えて推定を行うため、現時刻からどれだけ遠い過去の入力を出力に反映させるかは重要です。
- 順伝播型ネットワークの勾配消失/爆発問題からRNNは実際、高々過去10時刻分程度しか反映されていない
- RNNでは、短期的な記憶は実現できても、より長期にわたる記憶を実現するのは難しい

### LSTMの概要

長期にわたる記憶を実現できるようにするモデル → **長・短期記憶**(**Long Short-Term Memory**; **LSTM**)

LSTMは、上で述べた基本的なRNNに対し、**その中間層の各ユニットをメモリユニットと呼ぶ要素で置き換えた構造を持つ**。入出力層などのそれ以外の構造は元のRNNとまったく変わらない。

メモリユニット1つの内部構造




大きな矢印は、外部からの入力を表し、これは入力層から届くものと、中間層(=全メモリユニット)の出力を帰還させたものを合わせたもの。ユニットb,c,d,fはすべてこの入力(に異なる重みを掛けたもの)を受け取る.ユニットeはメモリセルの出力に活性化関数を適用する.

| 項目 | 説明 |
| --- | --- |
| **メモリセル**(a) | 状態$s_j^t$を保持し、これを1時刻を隔ててメモリセル自身に帰還することで記憶を実現する |
| ユニットf | ユニット$f$の出力がゲートの値$g_j^{F,t} \in [0,1]$となる. |
| **忘却ゲート** | メモリセルの帰還路には途中に忘却ゲートが挿入されおり、$s_j^t$に$g_j^{F,t}$を掛けたものが伝えられ、<br>$g_j^{F,t}$が1に近ければ現状態がそのまま記憶され、0に近ければリセット(忘却)される |
| ユニットb(入力) | メモリユニットへの外部からの入力はユニットbが受け取り、その出力がメモリセルに入力される.通常のRNNの中間層のユニット1つに相当する |
| ユニットc | 出力がゲート値$g_j^{I,t} \in [0,1]$になっている. |
| **入力ゲート** | ユニットb(入力)の出力に$g_j^{I,t}$を掛けたものがメモリセルに伝えられる |
| ユニットe | このユニットeを経てメモリユニットから外部へ出力される. |
| **出力ゲート** | ユニットdの出力がゲートの値$g_j^{O,t} \in [0,1]$になっている. この値が1に近ければメモリセルの出力は外部に伝達され、0に近ければブロックされる. |

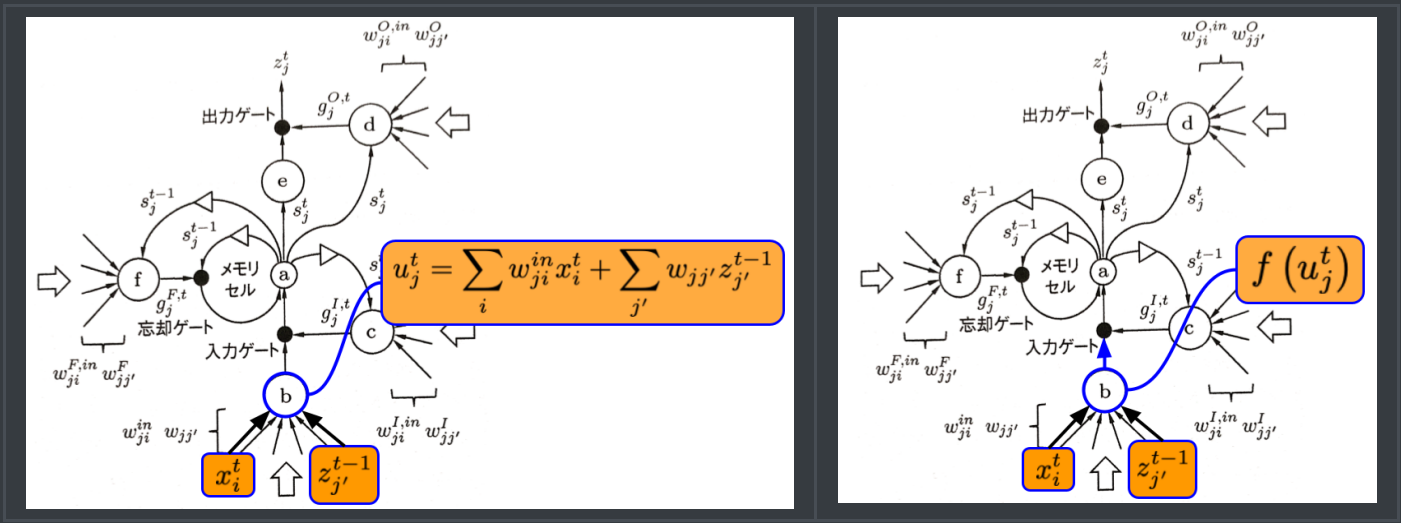
> 以上の仕組みは、**短期間の記憶しか実現できないというRNNの限界を緩和することを狙ったもの**である.
>
> 単純なケースでは忘却ゲートを1(オープン)、入力ゲートを0(クローズ)にし続けると、メモリセルの状態は永遠に記憶され続ける.

## 順伝播計算

### ユニットbの計算

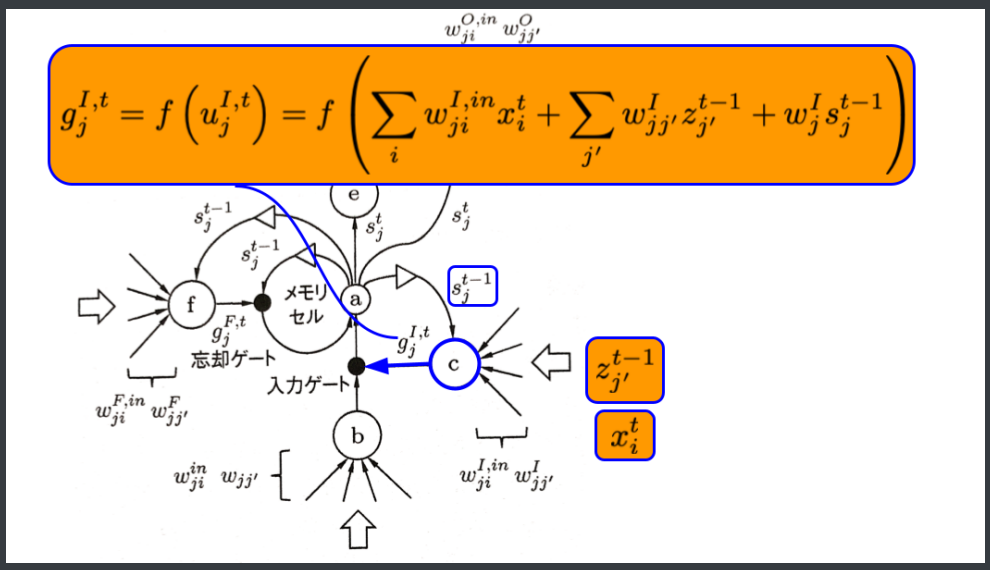
メモリユニット$j$が受け取る入力は、元のRNN同様、入力層と前の時刻の中間層から次のように入力を受け取る.

$$u_j^t = \sum_i w_{ji}^{in} x_i^t + \sum_{j'} w_{jj'} z_{j'}^{t-1}$$



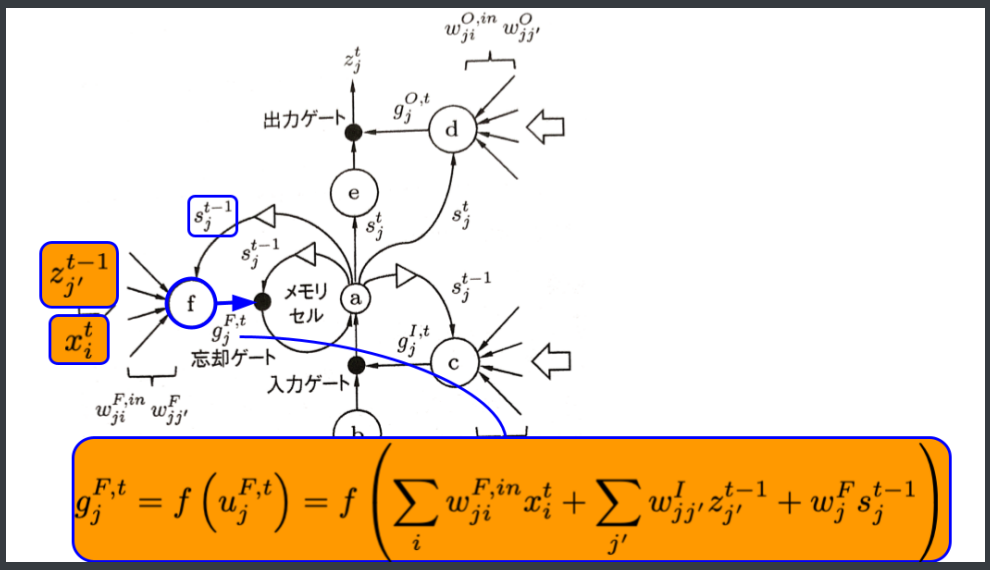

### 入力ゲートの値$g_j^{I,t}$の計算

$$g_j^{I,t} = f\left(u_j^{I,t}\right) = f\left(\sum_i w_{ji}^{I,in} x_i^t + \sum_{j'} w_{jj'}^I z_{j'}^{t-1} + w_j^I s_j^{t-1}\right)$$




### 忘却ゲートの値$g_j^{F,t}$の計算

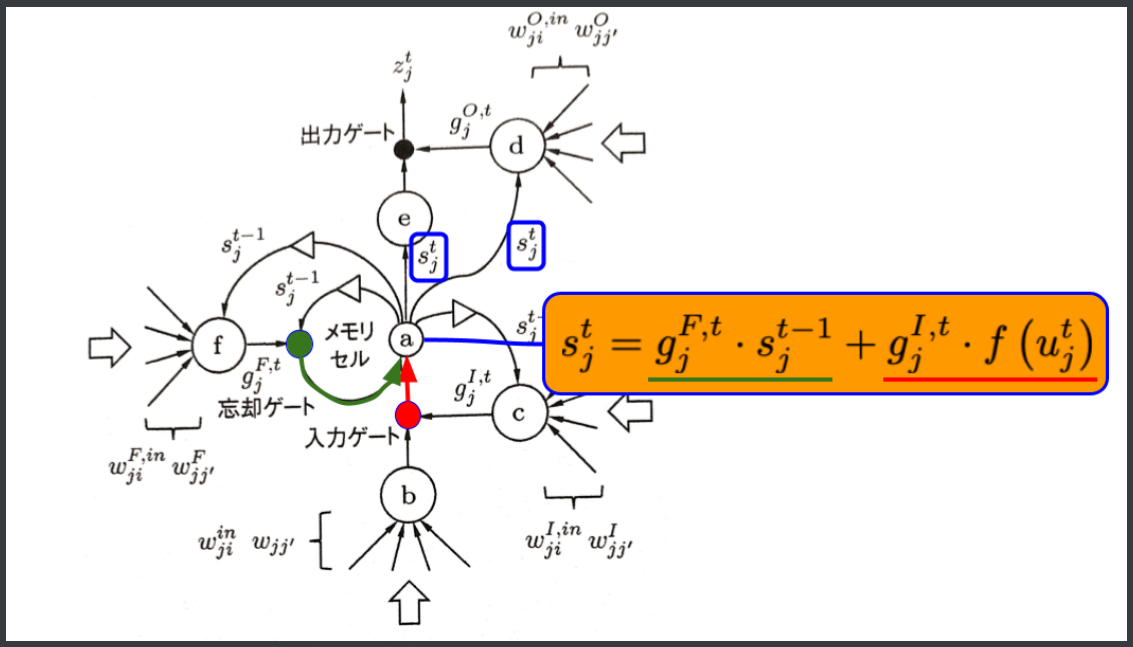
$$g_j^{F,t} = f\left(u_j^{F,t}\right) = f\left(\sum_i w_{ji}^{F,in} x_i^t + \sum_{j'} w_{jj'}^I z_{j'}^{t-1} + w_j^F s_j^{t-1}\right)$$




**メモリセルの計算**

$j$番目のメモリユニット内部のメモリセルは変数$s_j^t$を保持する.メモリセルの帰還路は変数$s_j^t$の中身を1時刻分将来に引き継ぎます.

$$s_j^t = g_j^{F,t} \cdot s_j^{t-1} + g_j^{I,t} \cdot f\left(u_j^t\right)$$




**出力ゲートの計算**

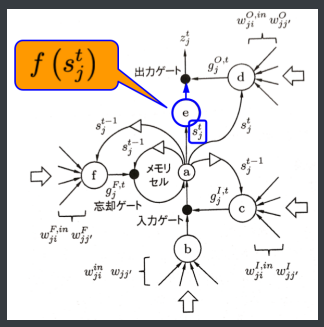
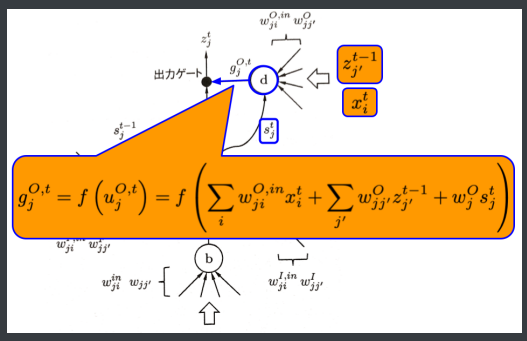
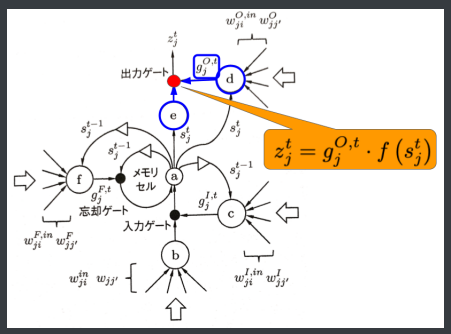
メモリユニットからの出力

$$z_j^t = g_j^{O,t} \cdot f\left(s_j^t\right)$$

ただし、$g_j^{O,t}$は出力ゲートの値

$$g_j^{O,t} = f\left(u_j^{O,t}\right) = f\left(\sum_i w_{ji}^{O,in} x_i^t + \sum_{j'} w_{jj'}^{O} z_{j'}^{t-1} + w_j^O s_j^t\right)$$

- 出力ゲートのみ、$s_j^{t-1}$**ではなく**$s_j^t$**を加算する**ことに注意すること！

| ユニットeの計算 | ユニットdの計算 | 出力ゲートの計算 |
|---|---|---|
|  |  |  |



### その他

- $w_j^F, w_j^I$：メモリセルから忘却ゲートと入力ゲートの値を決めるユニットへの結合の$s_j^{t-1}$の重み
- これらは、下で扱う出力ゲートに関する同様の結合と合わせて、「**のぞき穴(peephole)**」結合とも呼ばれている
- 各ゲートの値$g_j^{F,t}, g_j^{I,t}, g_j^{O,t}$は、その計算に用いる活性化関数$f$をロジスティックシグモイド関数とすることで、値域を$[0,1]$に制約する.
- メモリユニットの出力$z_j^t$は、次の時刻の3種のゲート(入力ゲート,忘却ゲート,出力ゲート)を制御するユニットへの入力となる他,出力層のユニットへの入力にもなり、さらに出力層の活性化関数に従って時刻$t$のネットワークの出力$\mathbf{y}^t$が確定する

## 逆順伝播計算

P.124~P.125「7.5.4 逆伝播計算」『機械学習プロフェッショナルシリーズ　深層学習』を参照

---

# 7.6 入出力間で系列長が異なる場合

## 隠れマルコフモデル

## コネクショニスト時系列分類法