目　次

内容

１． section1_再起型ニューラルネットワークの概念

１．１． RNN の概念

RNN とは時系列データに対応可能な、ニューラルネットワークである。

１．２．時系列データとは

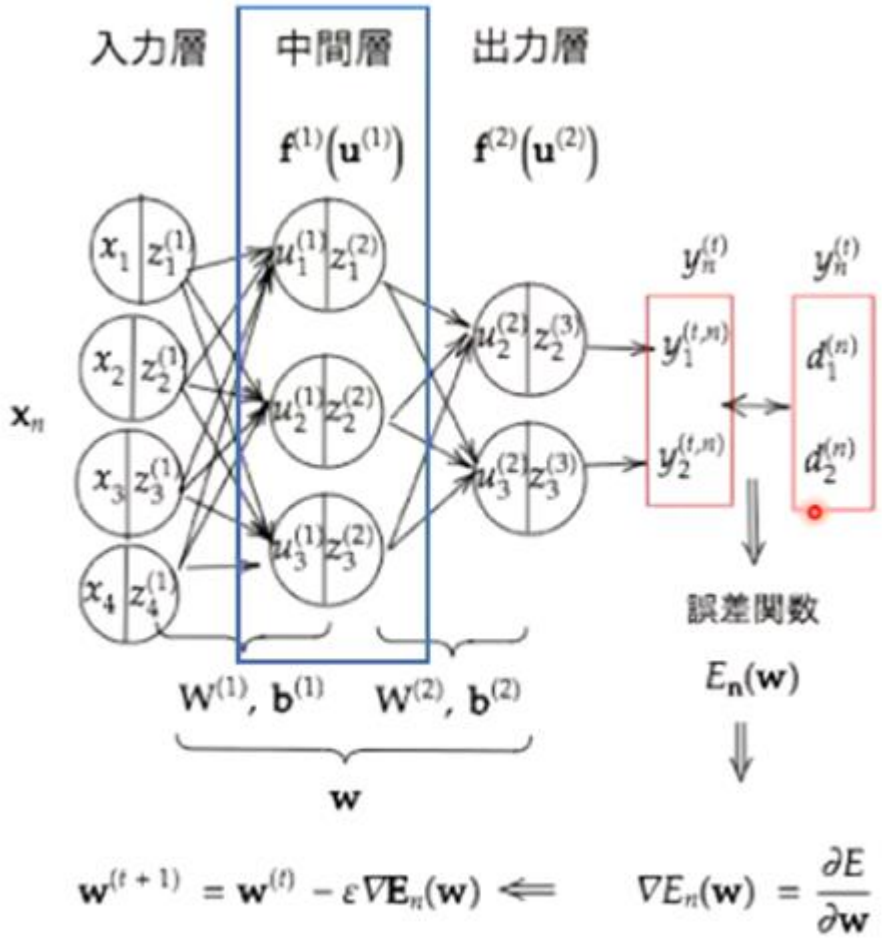
時間的順序を追って一定間隔毎に観察され、しかも相互に統計的依存関係が認められるようなデータの系列である。

時系列データには

１）音声データ・株価：時系列データ（時間的なつながりがあるデータ）

２）テキストデータ：単語１つ１つを前の単語に繋がって次の単語が出てくる。

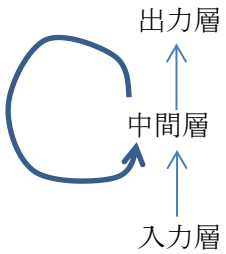
１．３．RNN について



$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \varepsilon \nabla \mathbf{E}_n(\mathbf{w}) \Leftarrow \quad \nabla E_n(\mathbf{w}) = \frac{\partial E}{\partial \mathbf{w}}$$

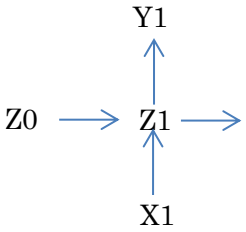
中間層：特徴を取得する。

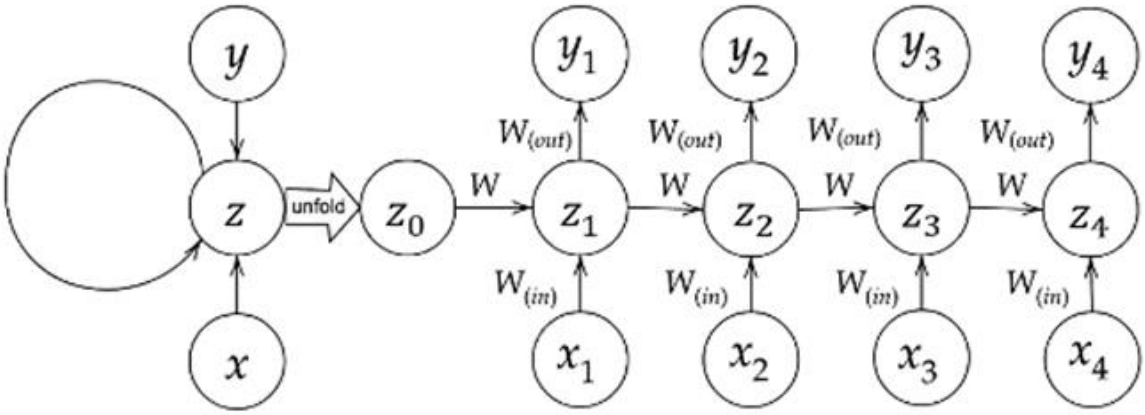
１．４．RNN の全体像



①．入力層－＞中間層－＞出力層

②．中間層－＞中間層





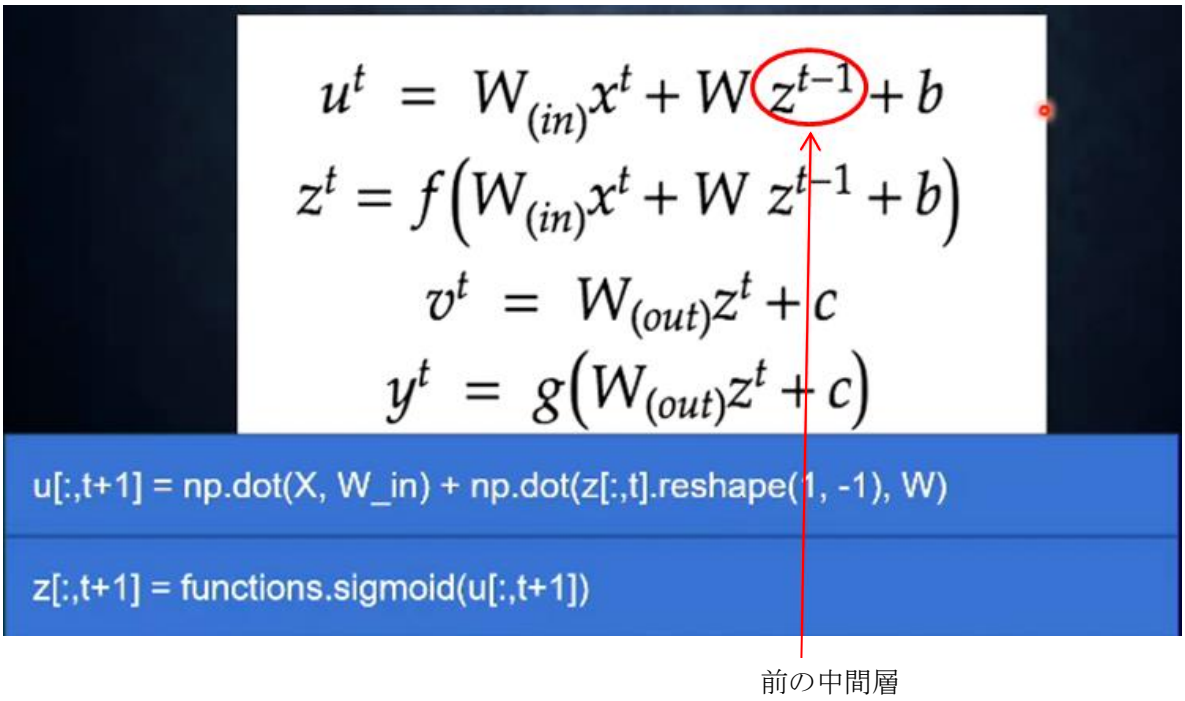
重みが３か所ある

１）Win：入力層から中間層への重み

２）Wout：中間層から出力層への重み

３）W：前の層からの入力用重み（中間層から中間層）

１．５．RNN の数学的記述



１）RNN の順伝播

F()、g()が活性化関数

ｚ＝ｆ（ｕｔ）

ｙ＝ｇ（ｖｔ）

１．６．RNN の特徴

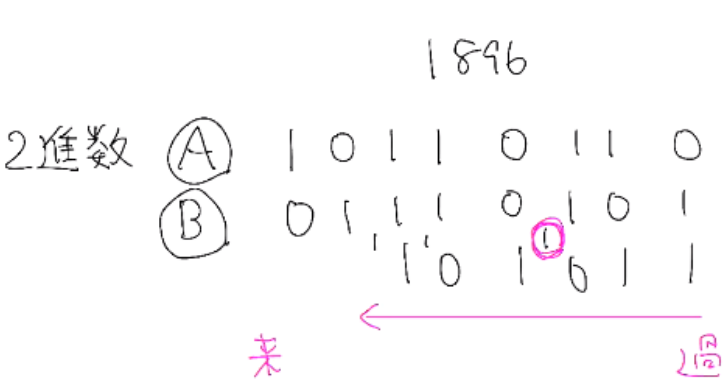
時系列モデルを扱うには、初期の状態と過去の時間（t-1）の状態を保持し、そこから次の時間での t を再帰的に求める再起構造が必要になる。

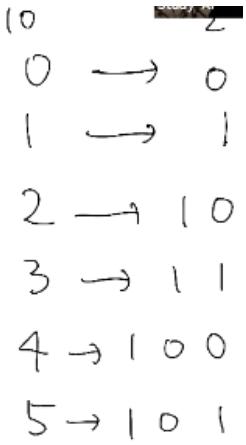
再起構造

Ｓｉｍｐｌｅ RNN

ﾊﾞｲﾅﾘ加算

右から左に時間的情報がある。





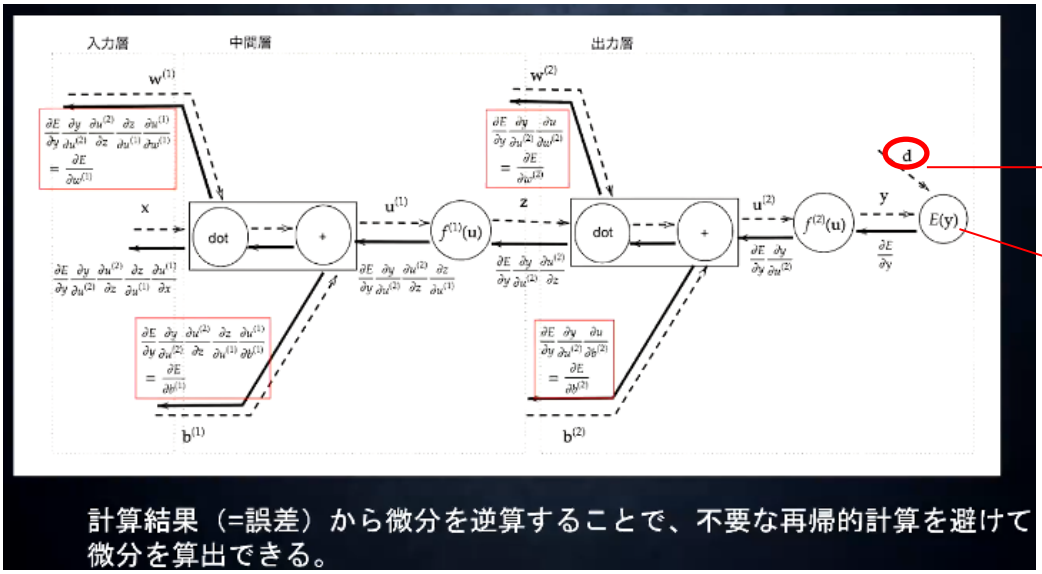
１．７．BPTT

１．７．１　BPTT とは

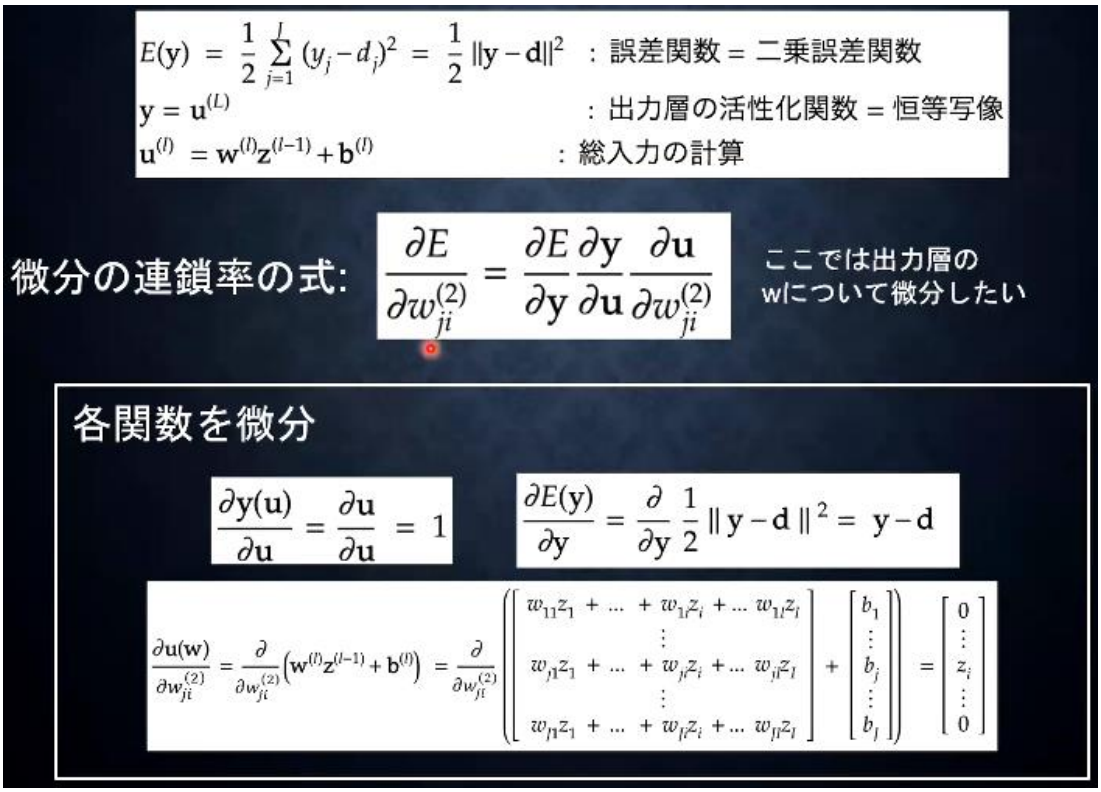
BackPropagation Through Time

RNN においてのパラメータ調整方法の一種（誤差逆伝播の一種）

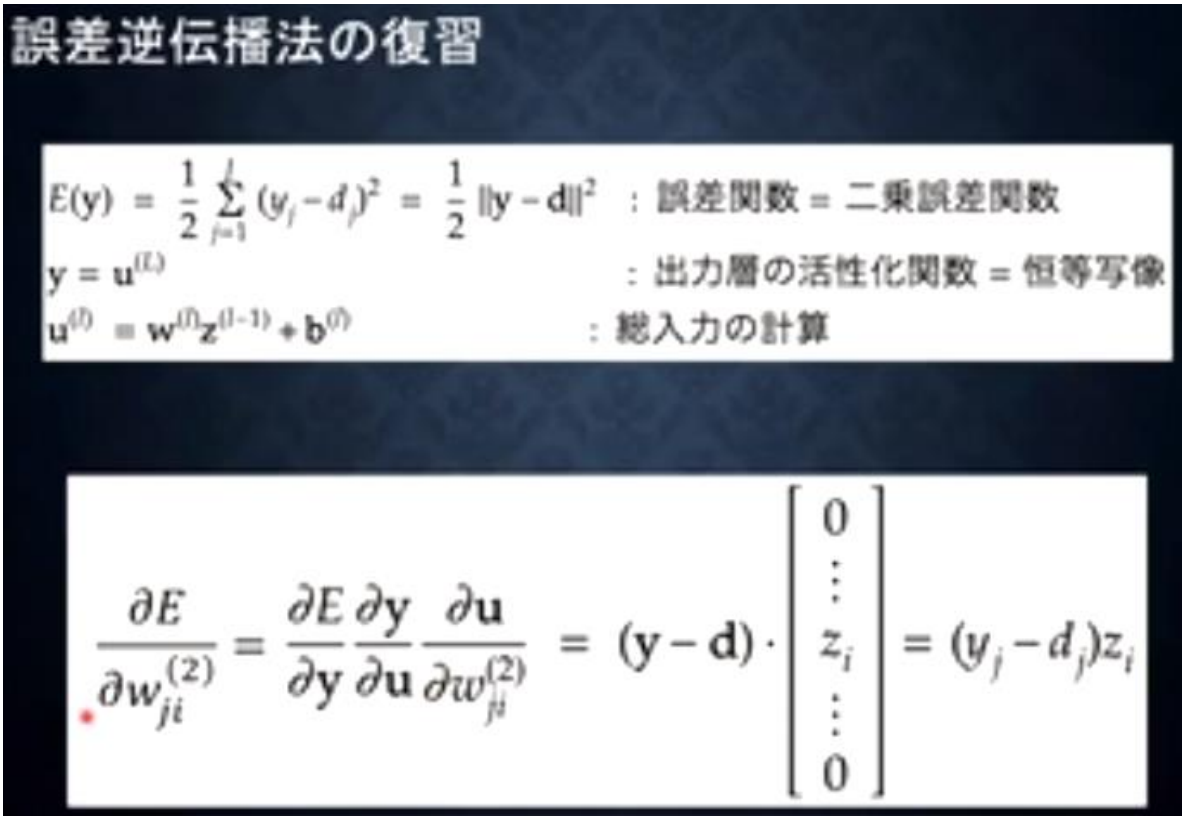
１）誤差逆伝播



微分の連鎖律－＞W,B を求める。



微分の連鎖式の３つの式が求められる。



２）確認

Z=t**2　　t=x+y

Z=(x+y)**2

Z=x**2+2xy+y**2

dz/dx=2x+2y

１．７．２　BPTT の数学的記述

１）BPTT の数学的記述１

$$\frac{\partial E}{\partial W_{(in)}} = \frac{\partial E}{\partial u^t}\left[\frac{\partial u^t}{\partial W_{(in)}}\right]^T = \delta^t\left[x^t\right]^T$$

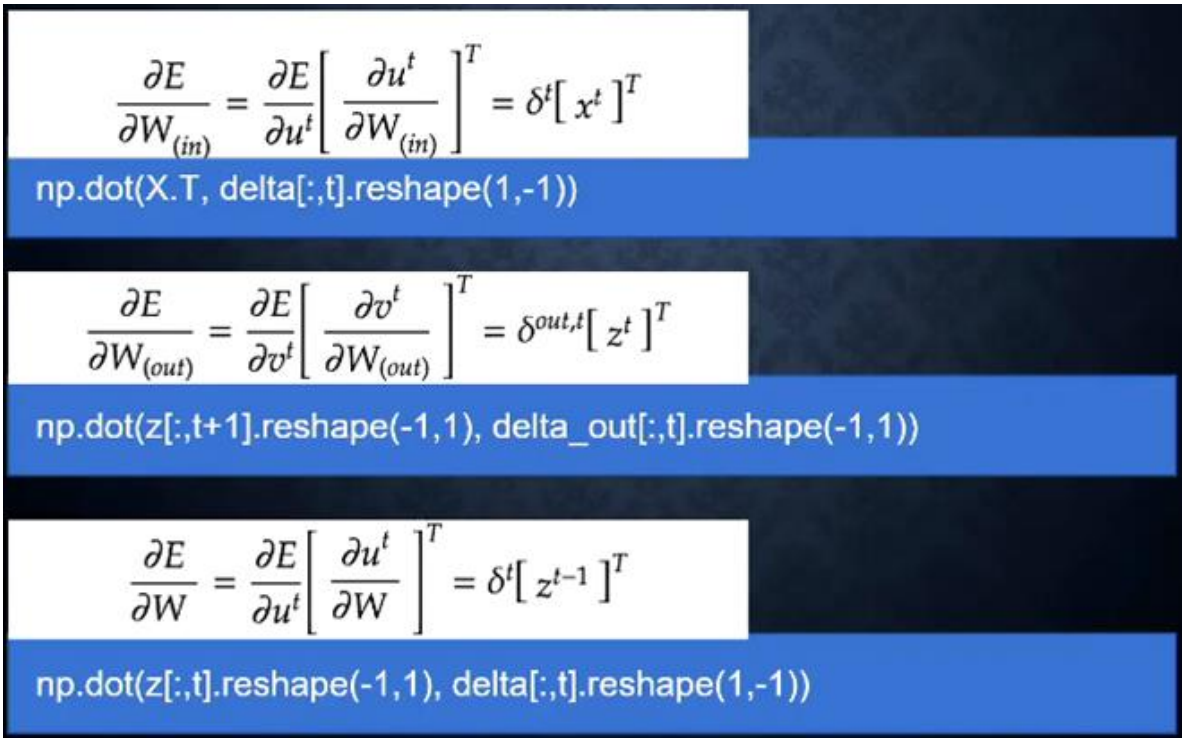
$$\frac{\partial E}{\partial W_{(out)}} = \frac{\partial E}{\partial v^t}\left[\frac{\partial v^t}{\partial W_{(out)}}\right]^T = \delta^{out,t}\left[z^t\right]^T$$

$$\frac{\partial E}{\partial W} = \frac{\partial E}{\partial u^t}\left[\frac{\partial u^t}{\partial W}\right]^T = \delta^t\left[z^{t-1}\right]^T$$

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial u^t}\frac{\partial u^t}{\partial b} = \delta^t$$

$$\frac{\partial E}{\partial c} = \frac{\partial E}{\partial v^t}\frac{\partial v^t}{\partial c} = \delta^{out,t}$$

３つに対して微分する



δ＝∇E／∇ｕ＊＊ｔ

X.T は時間の意味（転置とは違う）

2）BPTT の数学的記述 2

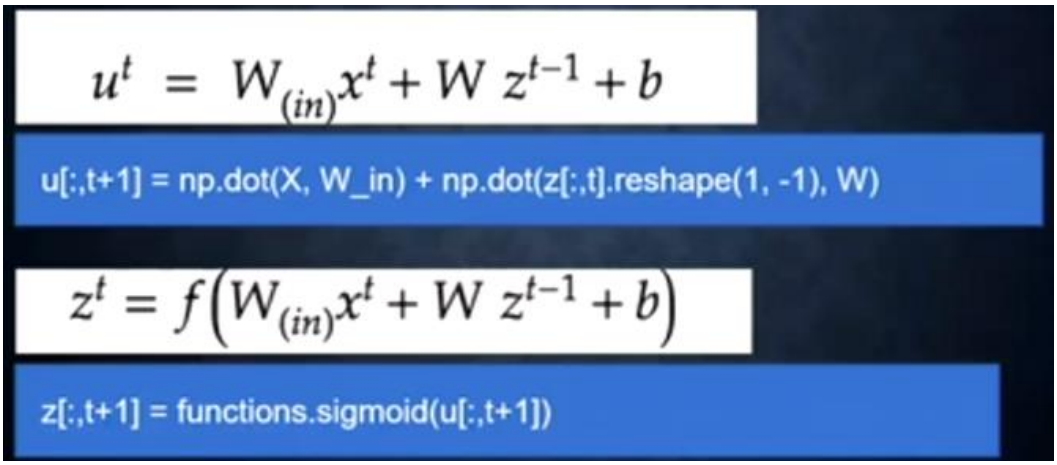
$$u^t = W_{(in)}x^t + W\,z^{t-1} + b$$

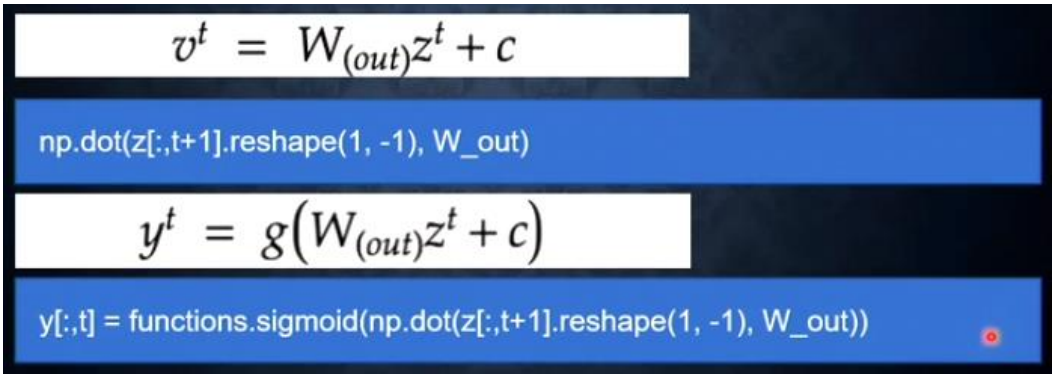
$$z^t = f\left(W_{(in)}x^t + W\,z^{t-1} + b\right)$$

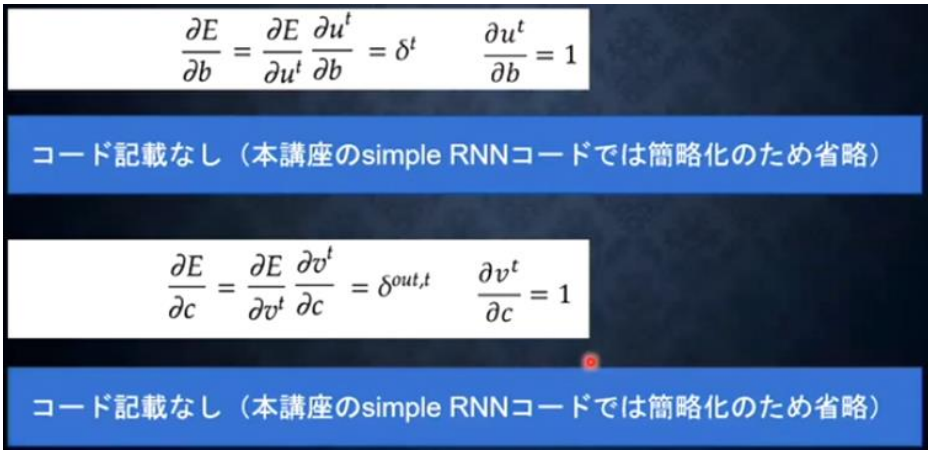
$$v^t = W_{(out)}z^t + c$$

$$y^t = g\left(W_{(out)}z^t + c\right)$$

順伝播の式

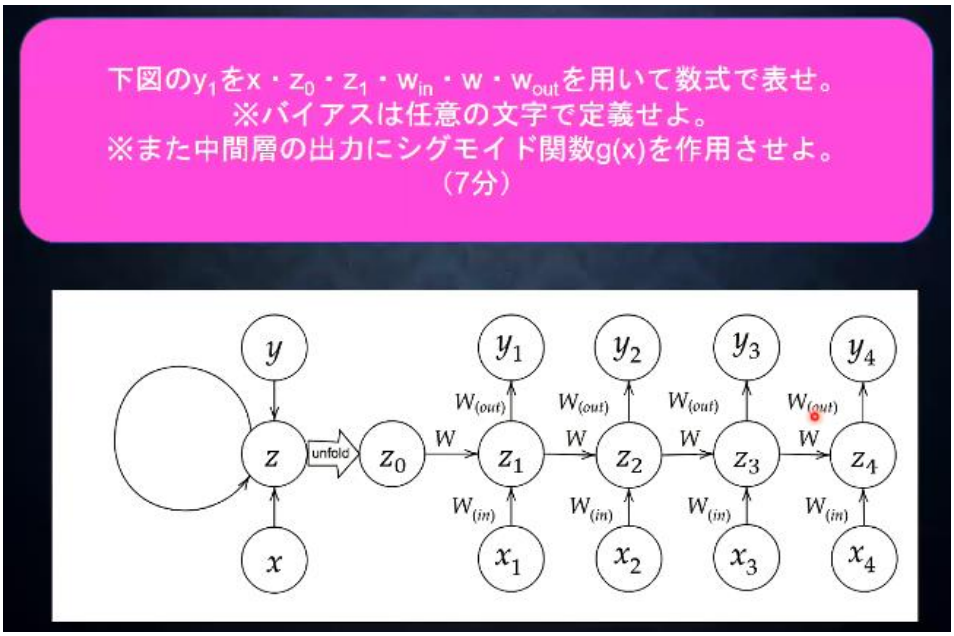


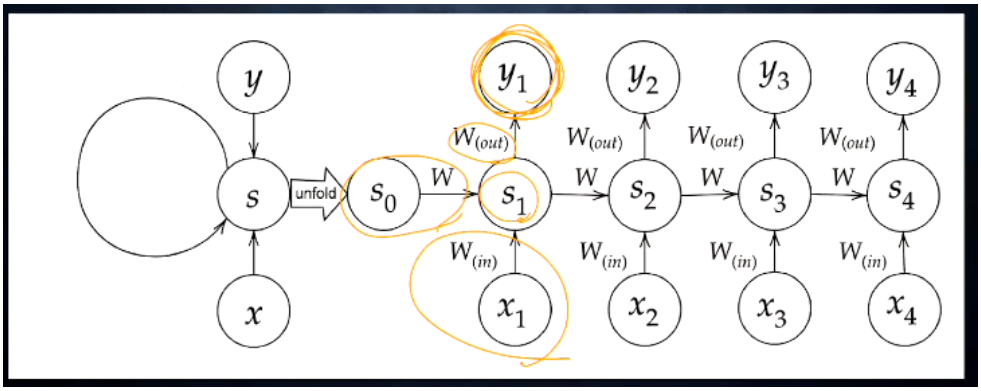


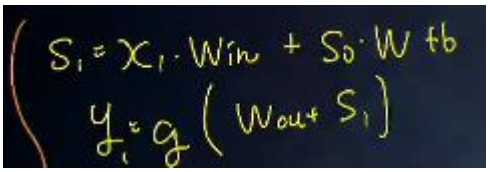


バイアスの微分式

●確認



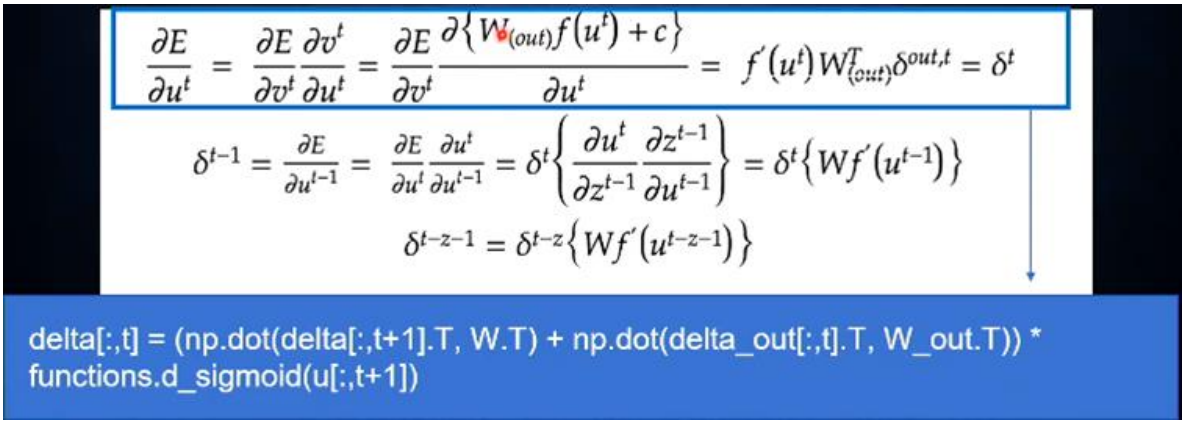




ｙ１＝ｇ（Ｗｏｕｔ・Ｓ１＋ｃ）

Ｓ１＝ｆ（Ｗｉｎ・ｘ１＋W・Ｓ０＋ｂ）

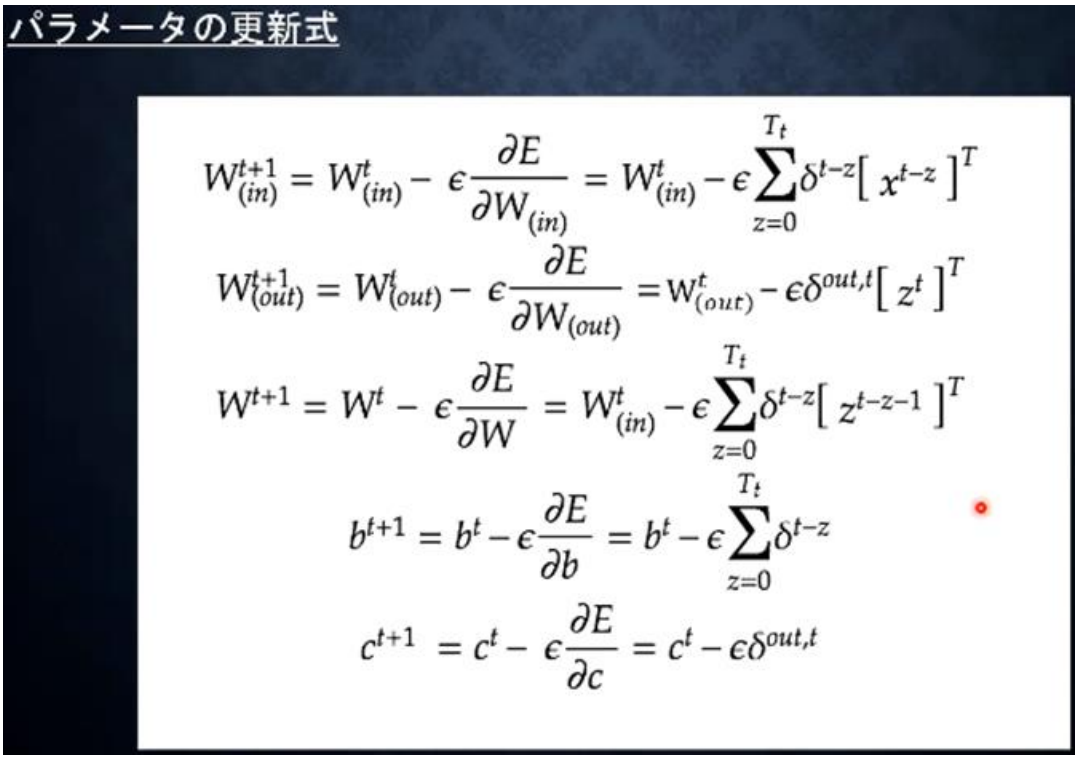
３）BPTT の数学的記述３



時間的つながりがある

ｔ時間目とｔ－１時間目とは関係がある。

４）BPTT の数学的記述４



重み

バイアス

Ｗｏｕｔの更新式には時間的に遡る処理はない

Ｗｉｎには時間的遡りがあるのでΣがある。

ε：学習率

$$W_{(in)}^{t+1} = W_{(in)}^{t} - \epsilon \frac{\partial E}{\partial W_{(in)}} = W_{(in)}^{t} - \epsilon \sum_{z=0}^{T_t} \delta^{t-z} \left[x^{t-z} \right]^T$$

```
W_in -= learning_rate * W_in_grad
```

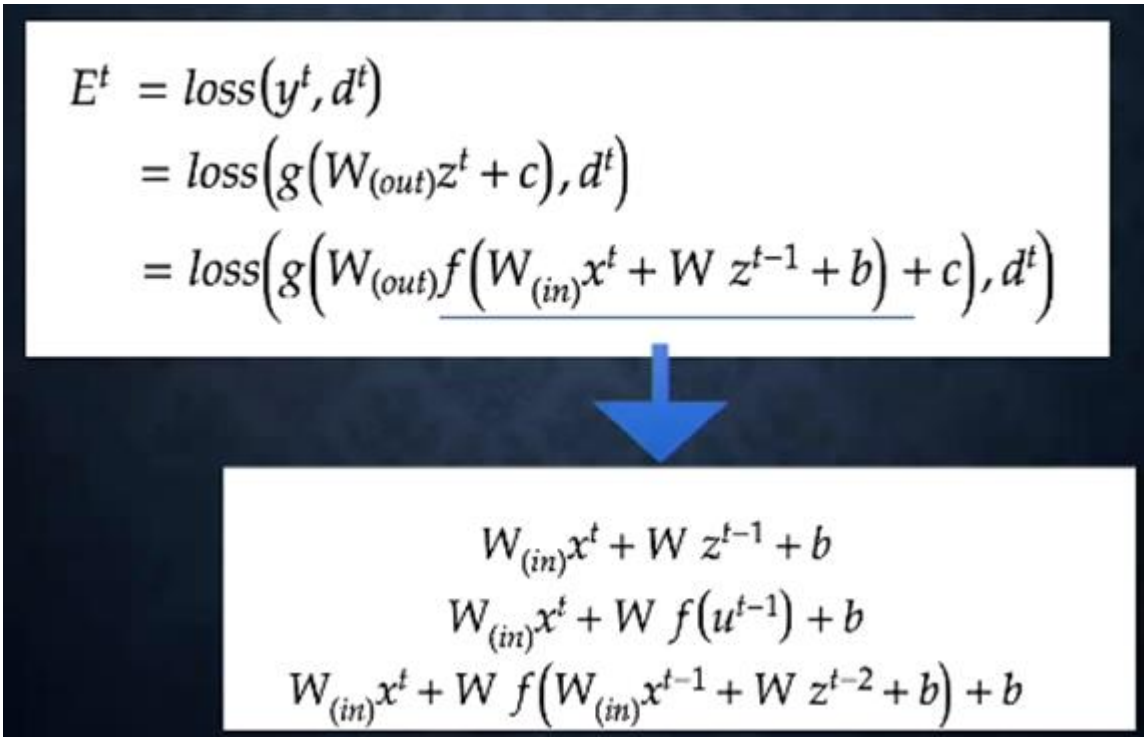
$$W_{(out)}^{t+1} = W_{(out)}^{t} - \epsilon \frac{\partial E}{\partial W_{(out)}} = W_{(out)}^{t} - \epsilon \delta^{out,t} \left[z^t \right]^T$$

```
W_out -= learning_rate * W_out_grad
```

$$W^{t+1} = W^{t} - \epsilon \frac{\partial E}{\partial W} = W_{(in)}^{t} - \epsilon \sum_{z=0}^{T_t} \delta^{t-z} \left[z^{t-z-1} \right]^T$$

```
W -= learning_rate * W_grad
```

１．７．３　BPTT の全体像



誤差関数はｙとｄから求まる。

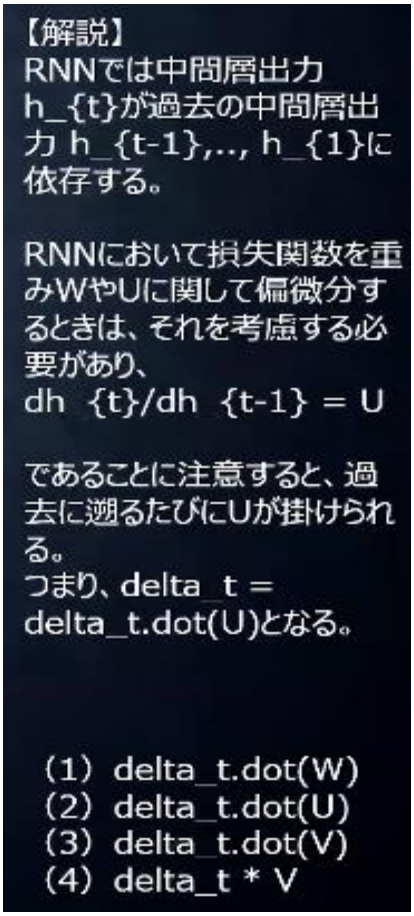
Ｚｔ－＞Ｘｔ－１－＞Ｚｔ－２・・・・・過去の繋がりがある

１）プログラム３＿１＿ｓｉｍｐｌｅ＿ＲＮＮ＿ａｆｔｅｒ

２）コードの演習問題

中間層の週力の出力

Delta_t.dot(U)



２．　section2_LSTM

２．１ RNN の課題

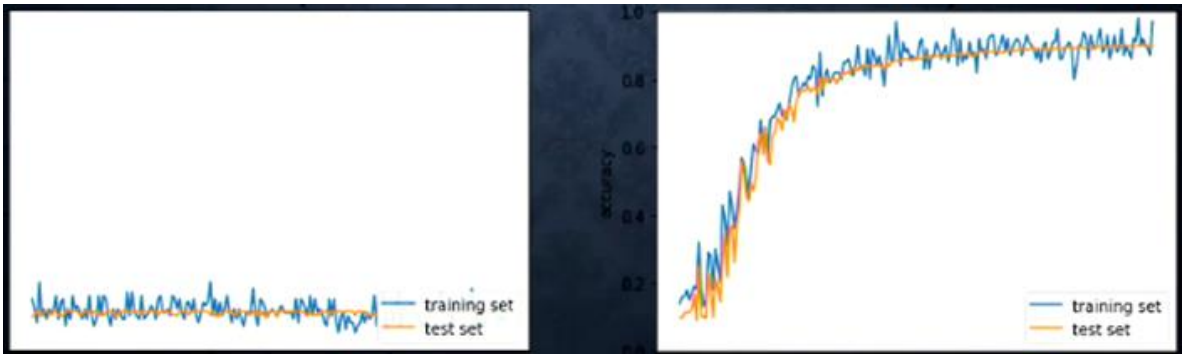
時系列を遡れば遡るほど、勾配が消失していくーーー＞長い時系列の学習が困難

解決策）

以前の勾配消失とは別で、構造自体を変えて解決したものが LSTM である。

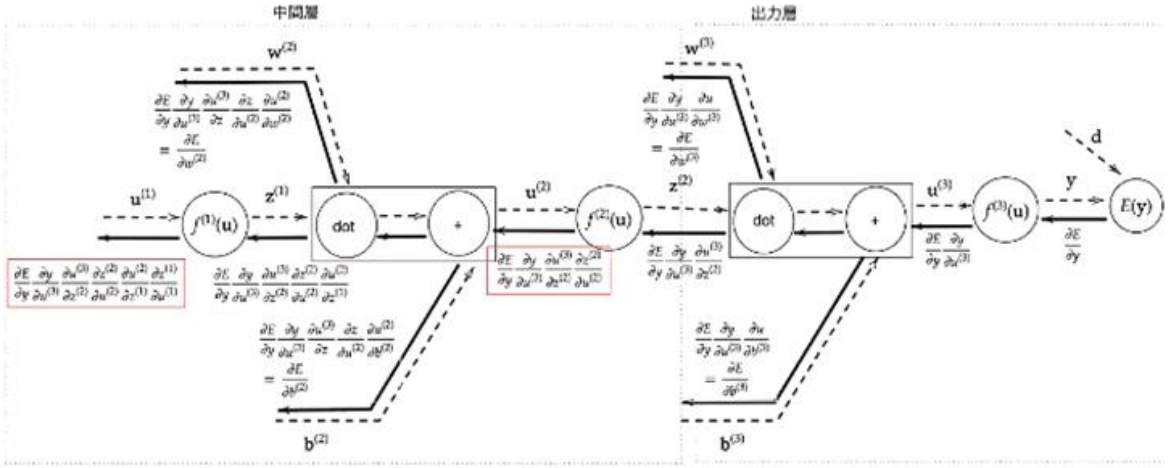
２．１．１ 勾配消失問題の復習

誤差逆伝播法が階層に進んでいくに連れて、勾配がどんどん緩やかになっていく。そのため、勾配降下法による更新では、階層のパラメータはほとんど変わらず、訓練は最適値に収束しなくなる。

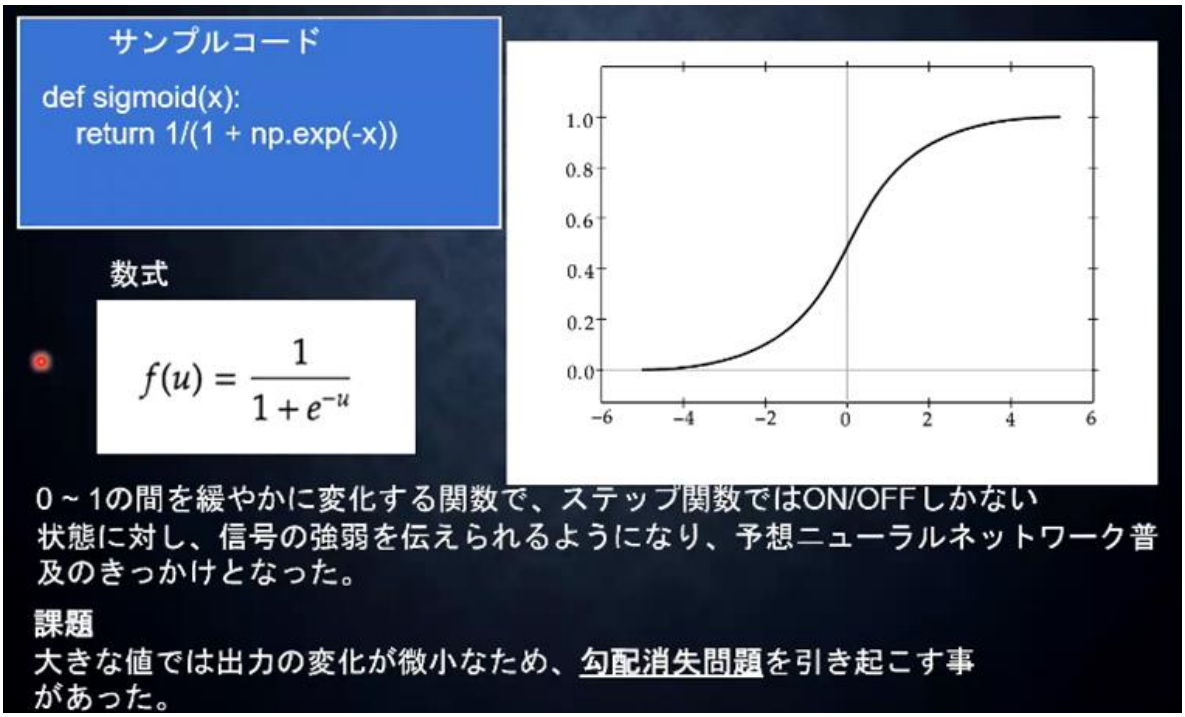


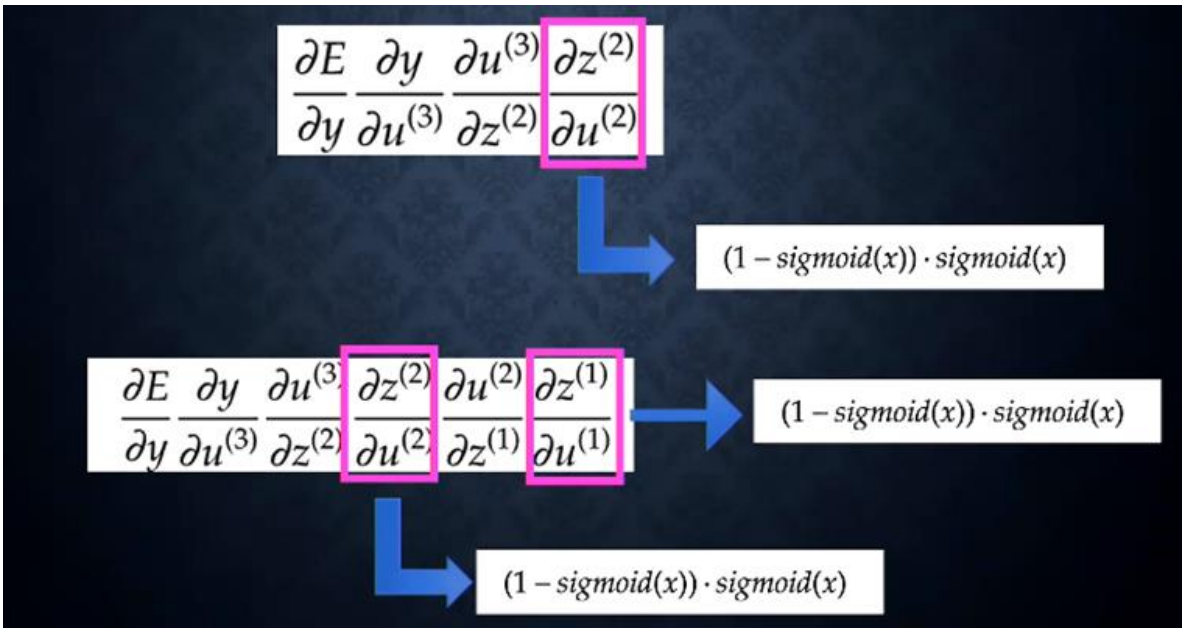
うまく進まない　　うまく進む

２．１．２ 勾配消失問題のビジョン



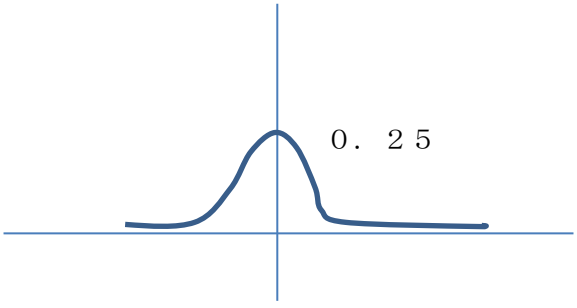
1）活性化関数：シグモイド関数





2）確認

シグモイド関数を微分した時最大値が０．２５となる。



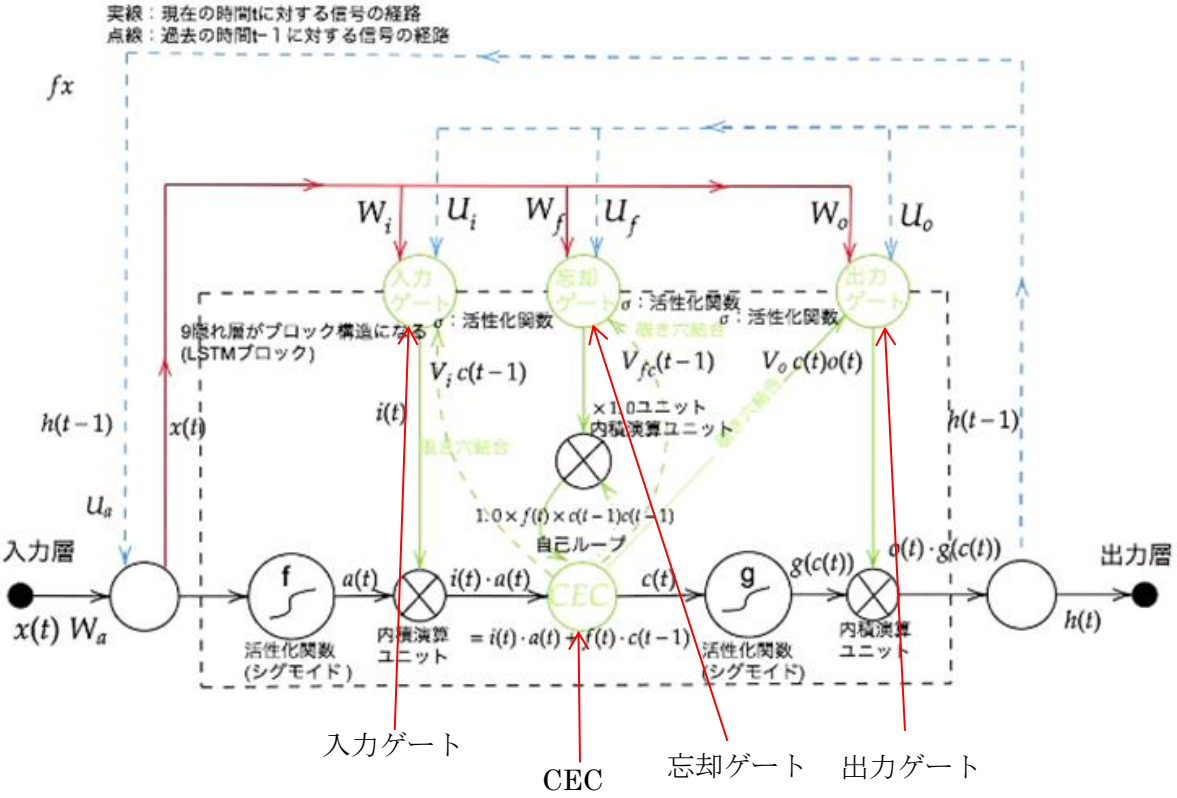
２．１．３ 勾配爆発

勾配が、層を逆伝播するごとに指数関数的に大きくなっていくこと。

逆伝播で勾配が大きくなる場合、恒等関数等を使用する。

２．１．４ LSTM の全体図

Long Short-Time Memory



RNN の一種

青い点線：時間的なﾙｰﾌﾟ

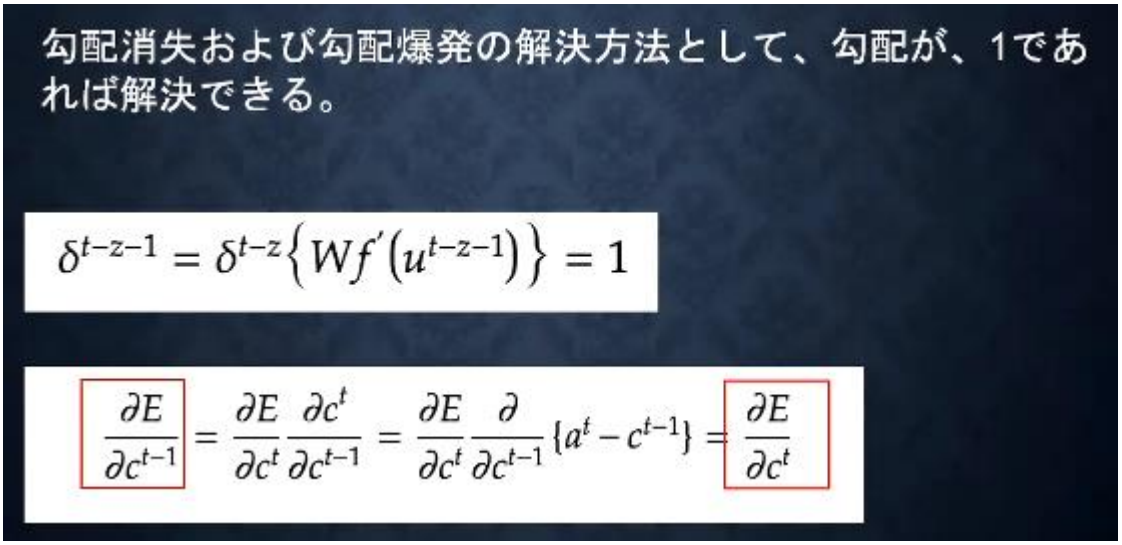
黒い点線：中間層

赤い矢印：今回の入力値（W）

ａ（ｔ）・ｉ（ｔ）＝ｃ（ｔ）：今回の状態

２．２ CEC

入力値や中間層の記憶を指せる。



勾配を１にするために導入されたのが CEC

１）CEC の課題

入力データについて、時間依存度に関係なく重みが一律である。

―――＞ニューラルネットワークの学習特性が無いということ。

入力層―＞隠れ層への重み：入力重み衝突

隠れ層―＞出力層への重み：出力重み衝突

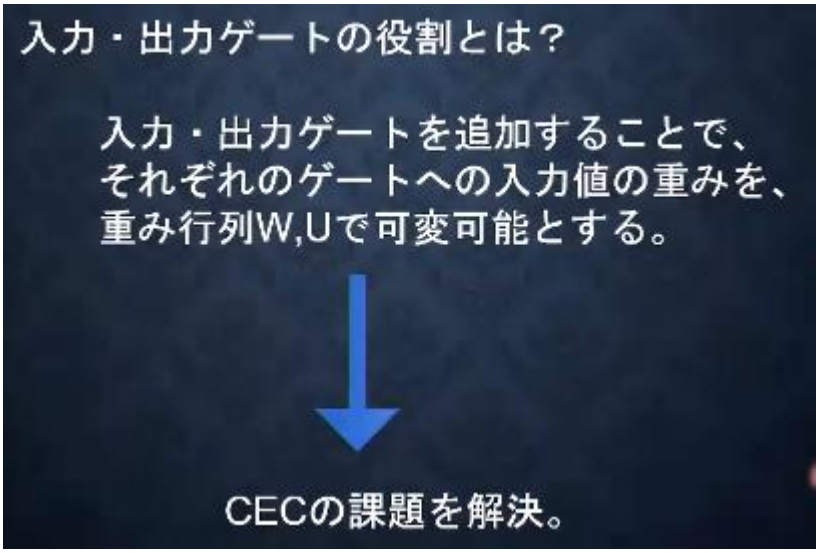
CEC は覚えることしかしないので、周りに学習機能を配置する。

２．３ 入力ゲートと出力ゲート

CEC の取り巻き

入力ゲート：CEC に覚え方を支持する。

出力ゲート：CEC のデータをどのように使用するか.



２．４ 忘却ゲート

１）LSTM の現状

CEC は過去の情報が全て保管されている。

２）課題

過去の情報がいらなくなった場合、削除することはできず保管され続ける。

３）解決策

過去の情報がいらなくなった場合、そのタイミングで情報を忘却する機能が必要。

―――＞忘却ゲートの誕生

いらなくなった情報を CEC から消去する。

４）演習問題

「映画面白かったね。ところで、とてもお腹が空いたから何か・・・・。」

この文章を LSTM に入力し・・・・に当てはまる単語を予測したいとする。

文中の「とても」という単語は空欄の予測においてなくなっても影響を及ぼさないと考えられる。

このような場合、どのゲートが作用するか――＞忘却ゲート

２．５ 覗き穴結合

１）課題

CEC の保存されている過去の情報を、任意のタイミングで他のノードに伝播させたり、あるいは任意のタイミングで忘却させたい。

CEC 自身の値は、ゲート制御に影響を与えていない。

２）除き穴結合とは

CEC 自身の値に、重み行列を介して伝播可能にした構造。

３）結果

あまり大きな改善は見られなかった。

３．　section3_GRU

１）LSTM の課題

　LSTM ではパラメータ数が多く、計算負荷が高くなる問題があった。

２）GRU とは

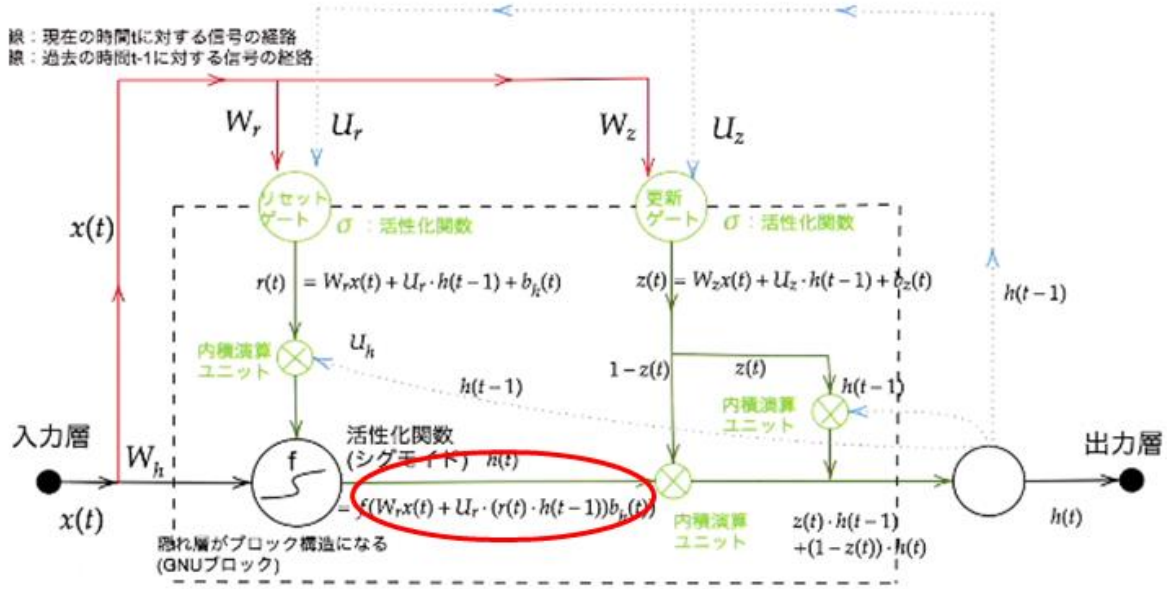
　従来の LSTM では、パラメータが多数存在していたため、計算負荷が大きかった。

GRU では、そのパラメータを大幅に削減し、精度は同等又はそれ以上が望めるようになった構造をしている。

３）メリット

　計算負荷が低い

３．１　GRU の全体像



全体的にシンプル。ＣＥＣがなくなった

リセットゲート・更新ゲート追加

ｈ（ｔ－１）：前回の記憶

更新ゲート：前回の記憶と今回の記憶をどれくらいの割合で混ぜ合わせるか

１）問題

　LTM と CEC が抱える課題について、それぞれ簡潔に述べよ

パラメータが多いので計算時間がかかる。

１）ＬＳＴＭの課題

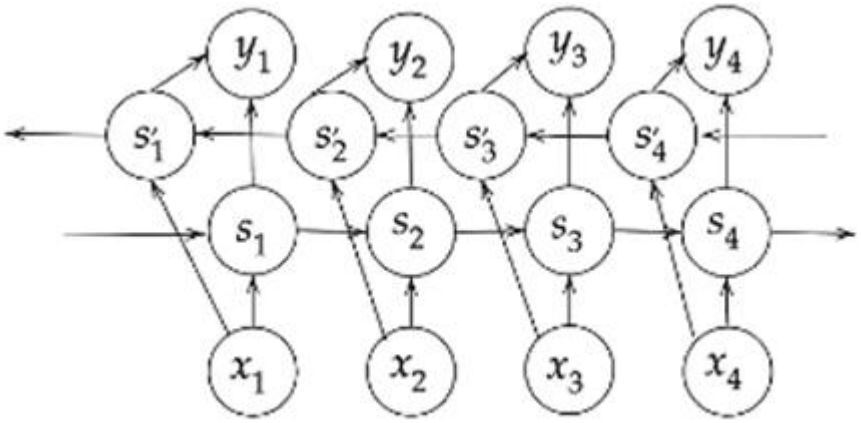
入力ゲート、出力ゲート、忘却ゲート、ＣＥＣはパラメータが多い

２）ＣＥＣの課題

勾配が１なので学習機能が無い

４．　section4_双方向 RNN

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



未来から過去

過去から未来

文章：過去の情報と未来の情報

実用例

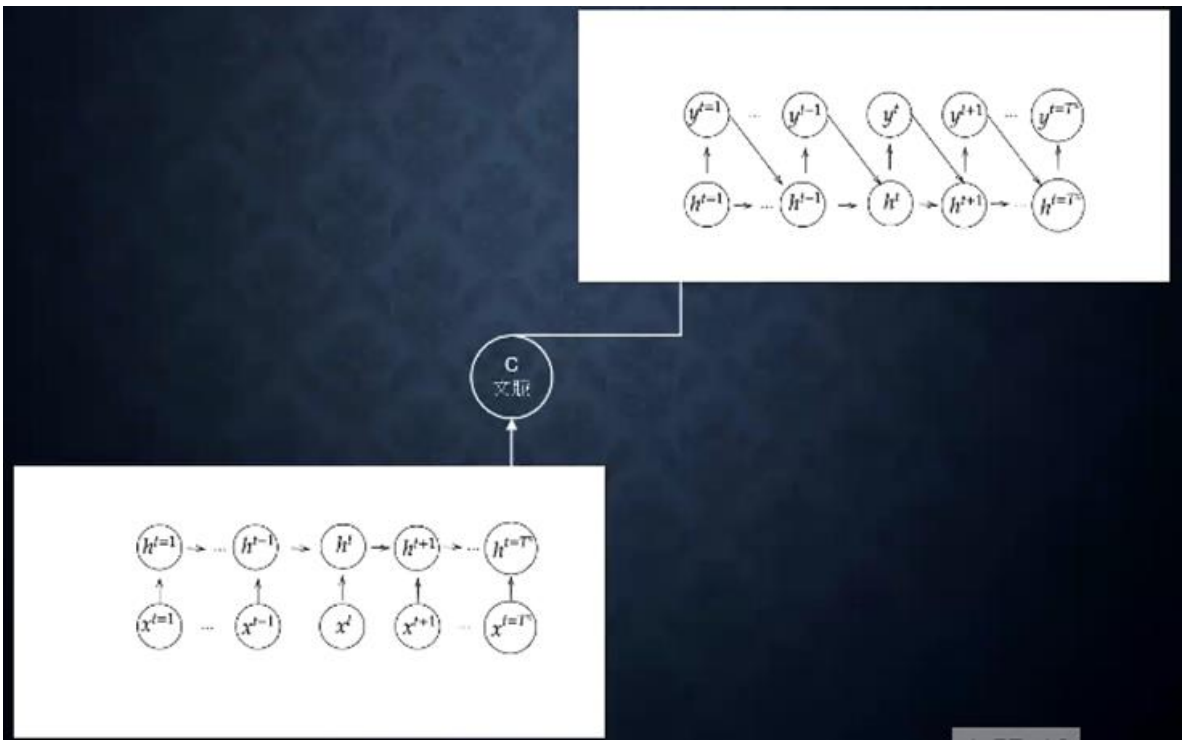
文章の推敲や、機械翻訳等

５．　section5_Seq2Seq

Encoder-Decoder モデルの一種を指す。

具体的な用途は、機械対話ら、機械翻訳などに使用される。

５．１　全体図



２つのニューラルネットワーク

１つ目のニューラルネットワークには、単語が入っていく。

蓄積されたデータ：中間層には入力されたものの記憶が溜まる。

意味がベクトル表現で保持される。

これをもう１つのニューラルネットワークに渡す。

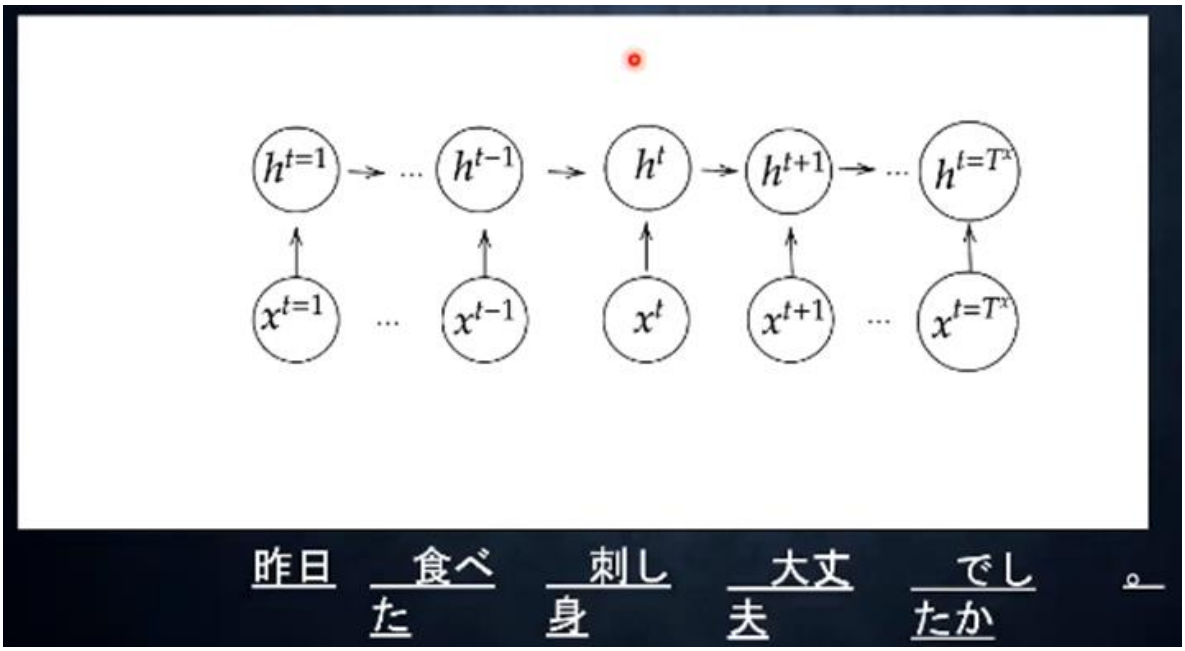
翻訳の形式

１）エンコーダ

２）デコーダ

５．２　EncoderRNN

ユーザーがインプットしたテキストを、単語等のトークンに区切って渡す構造。



文の意味を集約する。

全ての単語が集まって１つの文章を形成する。

１）Taking：文章を単語等のトークン毎に分割し、トークン毎の ID に分割する。

２）Embedding：ID から、そのトークンを表す分散表現ベクトルに変換する。

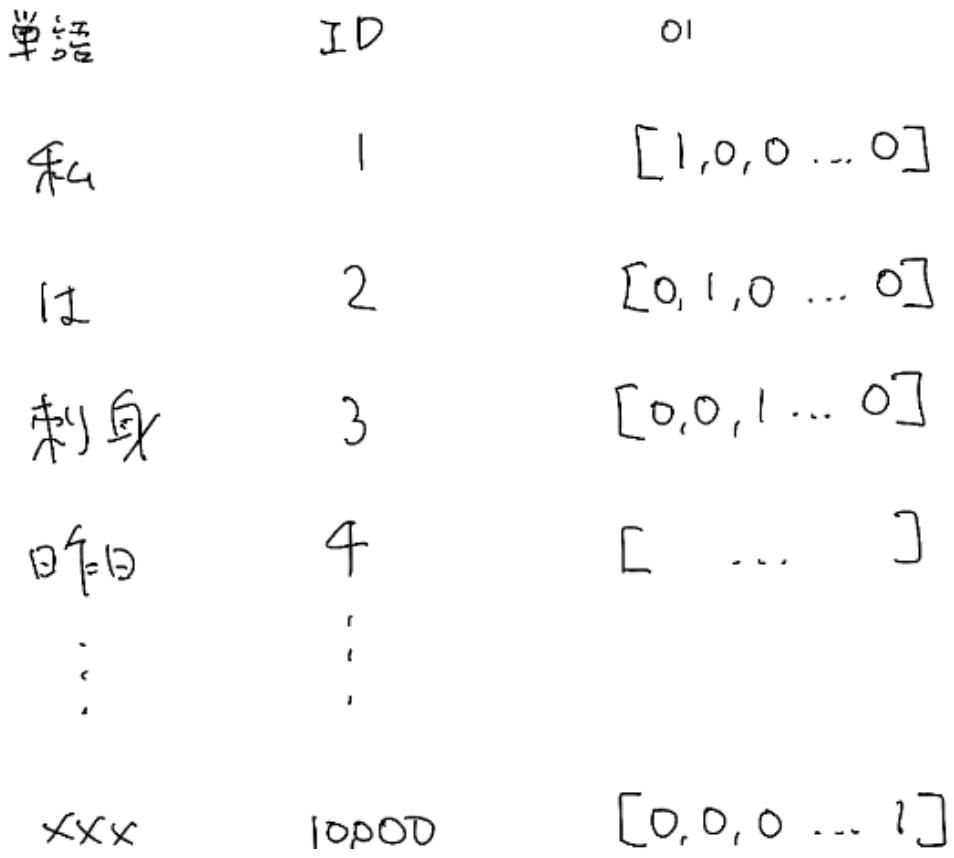
３）EncoderRNN：ベクトルを順番に RNN に入力していく。

自然言語のベクトル化

単語が無数にある。どのように登録するか

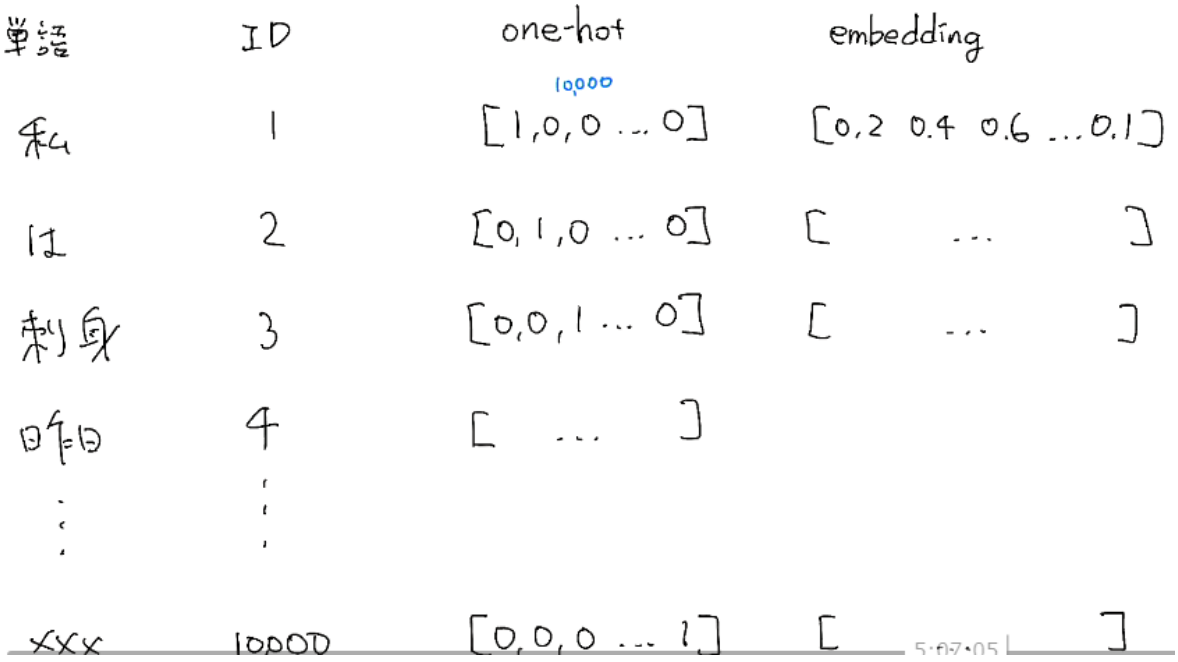
単語に番号を付ける。

Ｏｎｅ　Ｈｏｔ　Ｖｅｃｔｏｒ



１０００個の単語

エンベディング：数百程度（ディープラーニング）



単語の意味が似ているものはＥｍｂｅｄｄｉｎｇ表現で似通った数値になる。

５．２．１　EncoderRNN 処理手順

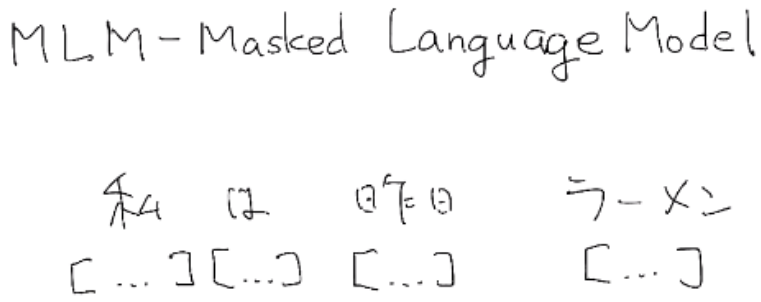
1）vec1 を RNN に入力し、hidden state を出力。

　この hidden state と次の入力 vec2 をまた RNN に入力してきた hidden state を出力という流れを繰り返す。

2）最後の vec を入れた時の hidden state を final state として取っておく。この final state が thought vector と呼ばれ、入力した文の意味を表すベクトルとなる。

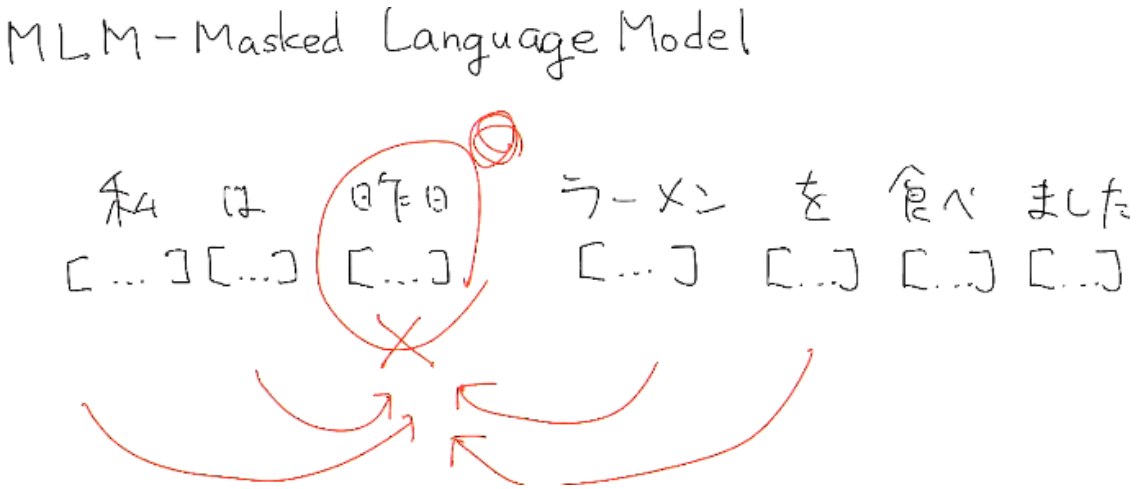
特徴量抽出

機械学習で同じ意味の単語をおなじような embedding 表現にする。



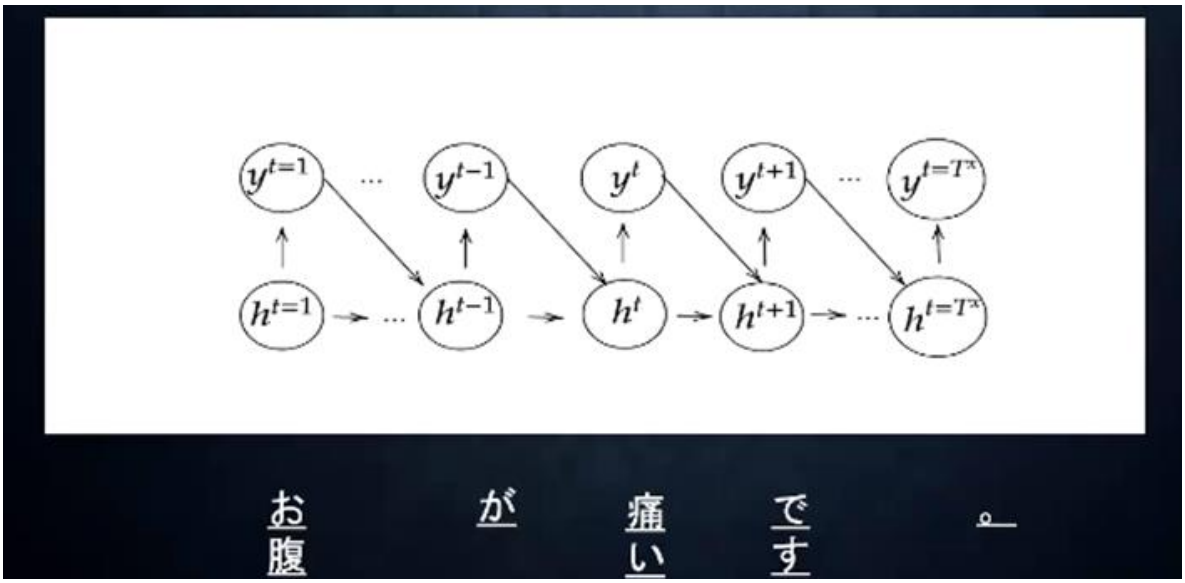
モデルを学習する時に 1 か所を見えなくする。

前後の文にゃ区から予想する。



５．３　DecoderRNN

システムがアウトプットデータを単語等のトークンごとに生成する構造。

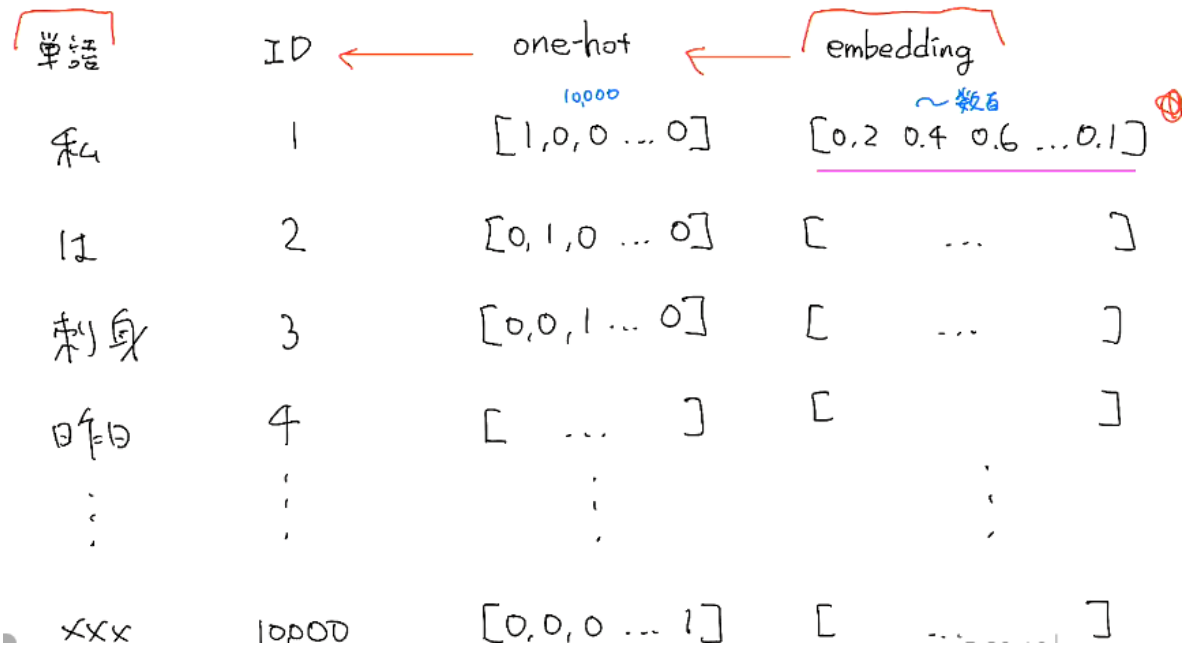


１）DecoderRNN：EncoderRNN の final state(thought vector)から、各 token の生成確率を出力していき、まず final state を DecoderRNN の initial state として設定し、Embedding を入力する。

２）Sampling：生成確率に基づいて token をランダムに選ぶ。

３）Embedding：２で選ばれた token を Embedding して DecoderRNN への次の入力とする。

４）Detokenize：１－３を繰り返し、２で得られた token を文字列に直す。



エンコーダーとは逆になる。

５．４　HRED

１）Seq2Seq の課題

一問一答しかできないーーー＞問に対して文脈も何もなく、ただ応答が行われ続ける。

この問題から HRED が生まれる。

５．４．１　HRED とは

過去 n-1 個の発話から次の発話を生成する。

Seq2Seq では、会話の文脈無視で応答がなされたが、HRED では、前の単語の流れに即して応答されるため、より人間らしい文章が生成される。

例）

システム：インコかわいいよね。

ユーザー：うん

システム：インコかわいいのわかる。

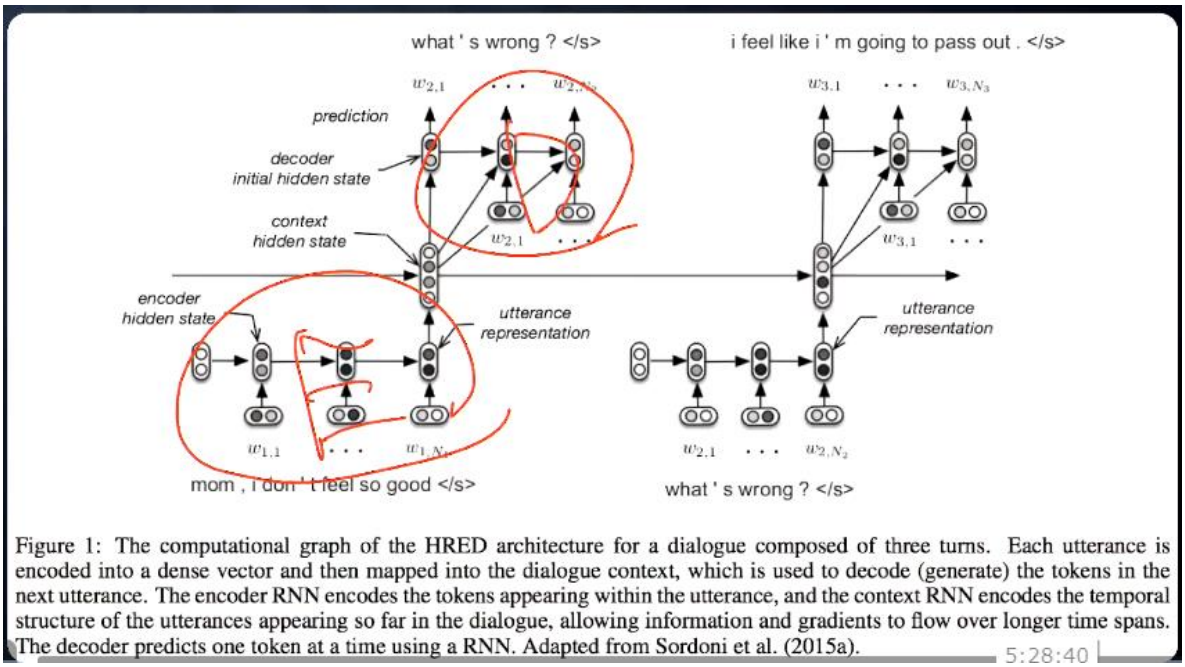


Figure 1: The computational graph of the HRED architecture for a dialogue composed of three turns. Each utterance is encoded into a dense vector and then mapped into the dialogue context, which is used to decode (generate) the tokens in the next utterance. The encoder RNN encodes the tokens appearing within the utterance, and the context RNN encodes the temporal structure of the utterances appearing so far in the dialogue, allowing information and gradients to flow over longer time spans. The decoder predicts one token at a time using a RNN. Adapted from Sordoni et al. (2015a).

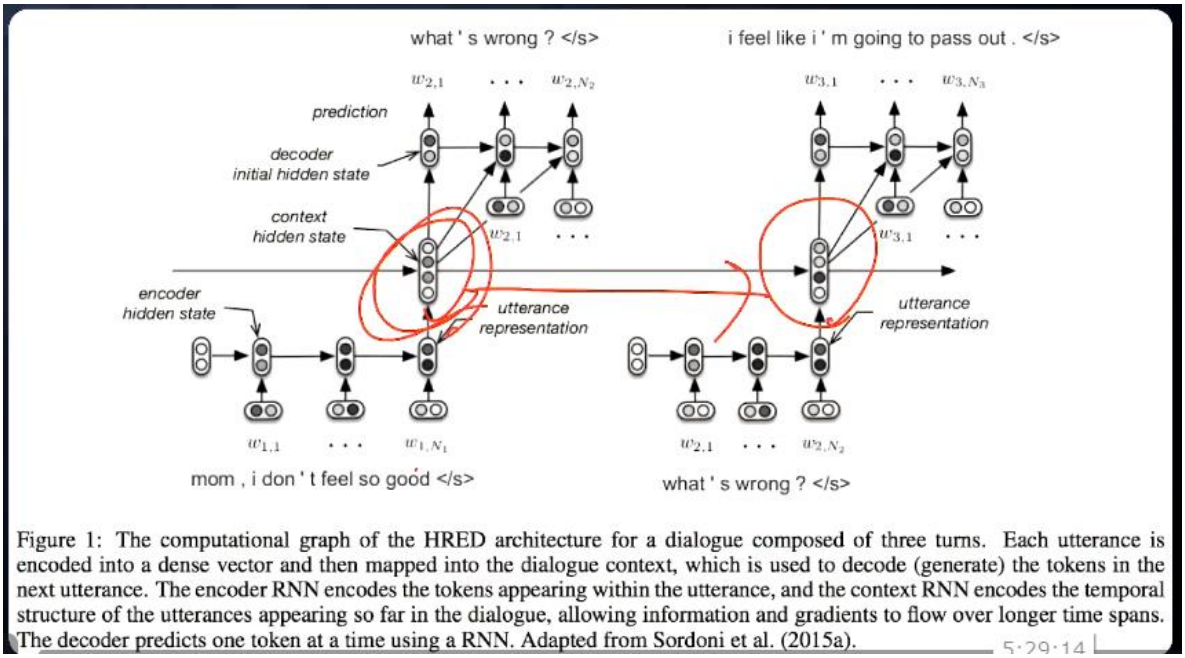


Figure 1: The computational graph of the HRED architecture for a dialogue composed of three turns. Each utterance is encoded into a dense vector and then mapped into the dialogue context, which is used to decode (generate) the tokens in the next utterance. The encoder RNN encodes the tokens appearing within the utterance, and the context RNN encodes the temporal structure of the utterances appearing so far in the dialogue, allowing information and gradients to flow over longer time spans. The decoder predicts one token at a time using a RNN. Adapted from Sordoni et al. (2015a).

５．４．２　HRED の構造

HRED とは Seq2Seq＋Context RNN

Context RNN : Encoder のまとめた各文章の系列をまとめて、これまでの会話コンテキスト全体を表すベクトルに変換する構造。

———＞過去の発話の履歴を加味した返答をできる。

５．４．３　HRED の課題

HRED は確率的な多様性が字面にしかなく、会話の「流れ」のような多様性が無い。

—＞同じコンテキスト（発話リスト）を与えられても、答の内容が毎回会話の流れとしは同じものしか出せない。

HRED は短く情報量に乏しい答えをしがちである。

—＞短いよくある答えを学ぶ傾向がある。

例）「うん」「そうだね」「・・・」など。

５．５　VRED

５．５．１　VRED とは

HRED に、VAE の潜在変数の概念を追加したもの

———＞HRED の課題を VAE の潜在変数の概念を追加することで解決した構造。

５．６　VAE

５．６．１　オートエンコーダ

１）オートエンコーダとは

教師なし学習の一つ。

そのため学習時の入力データは訓練データのみで、教師データは利用しない。

２）オートエンコーダの具体例

MNIST の場合、２８×２８の数字の画像を入れて、同じ画像を出力するニューラルネットワークとなる。

３）オートエンコーダの構造

入力データから潜在変数 Z に変換するニューラルネットワークを Encoder、逆に潜在変数 Z をインプットよし

て元画像を復元するニューラルネットワークを Decoder。

４）メリット

次元削減が行える。

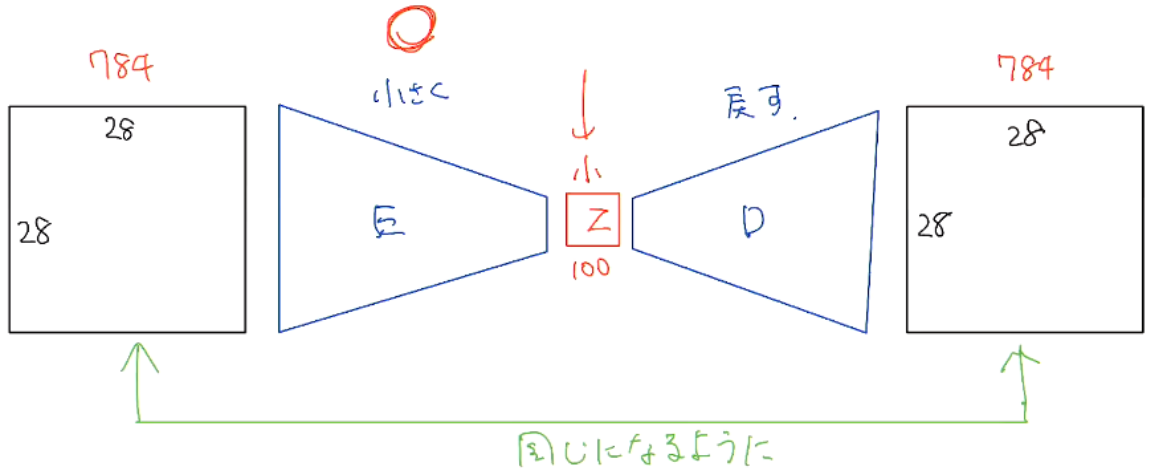
注）Z の次元が入力データより小さい場合、次元削減とみなすことができる。

潜在変数 Z

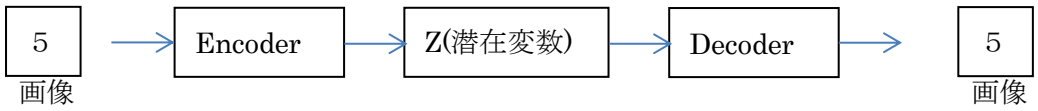
MNIST データ

入力と出力が同じになるように学習する

７８４－＞１００：次元削減



５）オートエンコーダの構造図



５．６．２　VAE

オートエンコーダを工夫する。

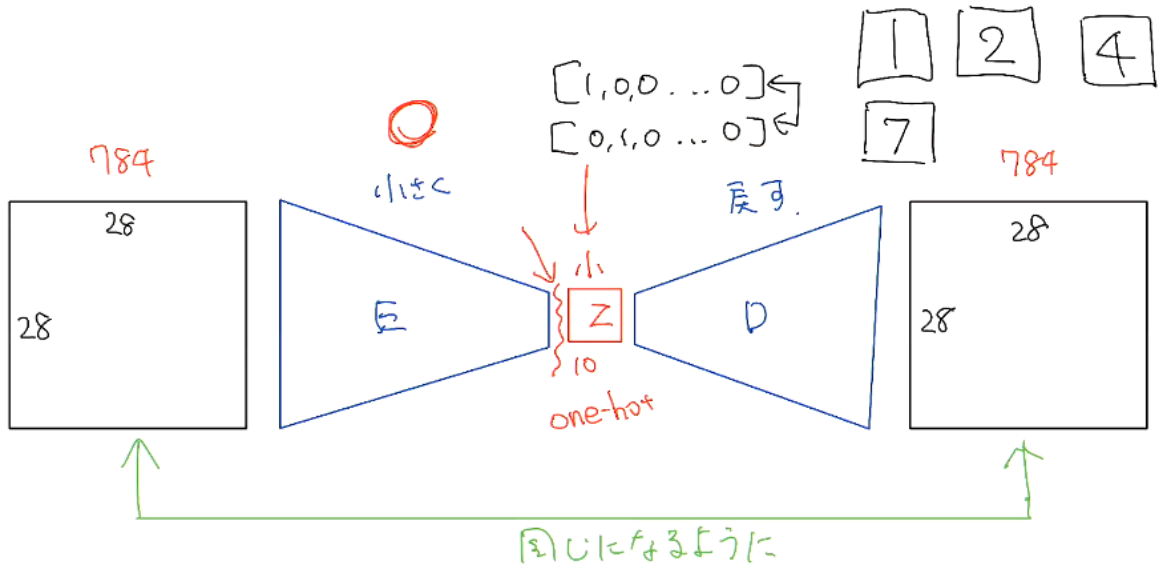
通常のオートエンコーダの場合、何かしら潜在変数 Z にデータを押し込めているものの、その構造がどのようなものかわからない。―＞VAE はこの潜在変数 Z に確率分布 Z-N(0,1)を仮定したもの。

―――＞VAE は、データを潜在変数 Z の確率分布という構造に押し込めることを可能にする。

Encoder の出力に対して、Decoder の入力にランダム性を持たせる。－＞ノイズを加える。

１０この数字を One　Hot　Vector を学習―＞元のデータの近さは無視される。

近いと同じようなベクトルになってほしい



ノイズを加える。

デコーダーは、より汎用的になる。

１）問題

VAE とは自己符号化器の潜在変数に確率分布を導入したもの。

６．　section6_Word2vec

課題：RNN では、単語のような可変長の文字列を RNN に与えることはできない。

―――＞固定長形式で単語を表す必要がある。

学習データからボキャブラリを作成

注）わかりやすく７語のボキャブラリを作成したら、本来は辞書の単語数だけ出来上がる。

例）I want to eat a pples.I like apples.

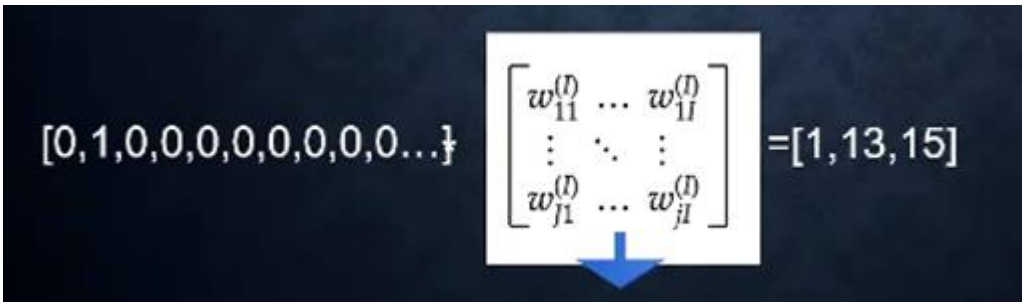
　｛aples,eat,I,like,to,want｝

１）メリット

　大規模データの分散表現の学習が、現実的な計算速度とメモリ量で実現可能にした。

―――＞X：ボキャブラリ×ボキャブラリだけの重み行列が誕生。

　　○：ボキャブラリ×任意の単語ベクトル次元で重み行列が誕生。



ボキャブラリ数×単語ベクトルの次元数の重み行列

Embedding 表現すること

７．　section7_Attention Mechanism

課題：Seq2Seq の問題は、長い文章への対応が難しい。

　Seq2Seq では、２単語でも、１００単語でも固定次元ベクトルの中に入力しなければならない。

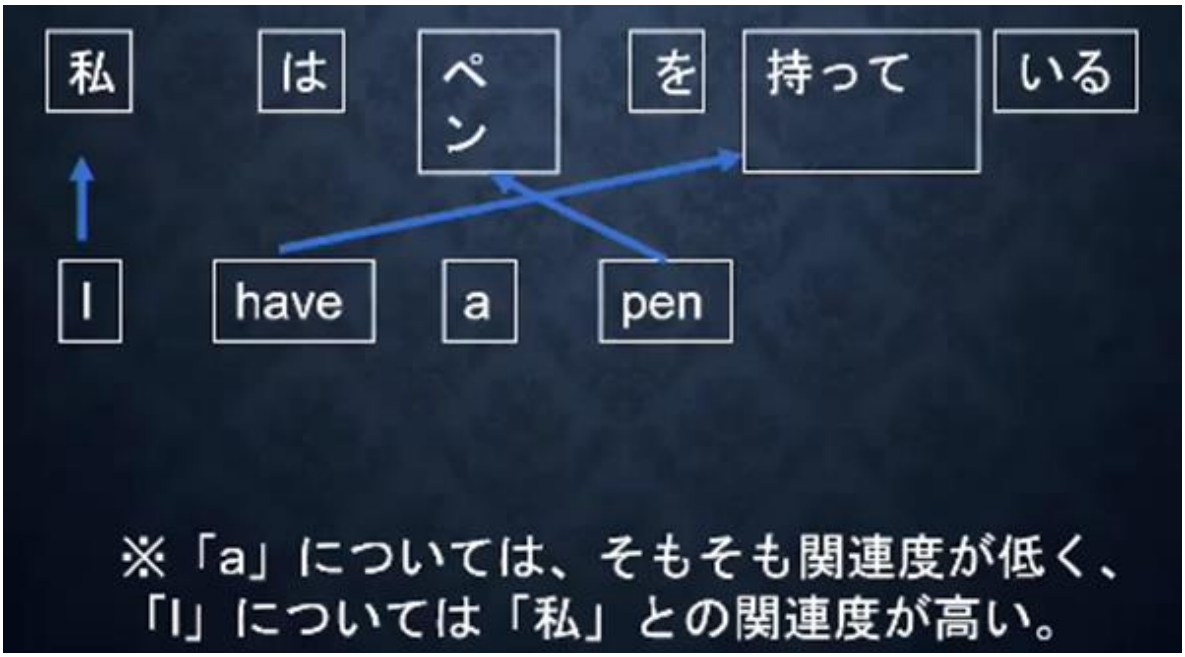
解決策：文章が長くなるほどそのシーケンスの内部表現の次元も大きくなっていく仕組みが必要になる。

―――＞Attention Mechanism

↓

「入力と出力のどの単語が関連しているのか」の関連度を学習する仕組み。

１）具体例



２）問題１

　RNN と word2vec、seq2seq と Attention の違い

RNN は時系列データを処理するのに適したもの

ｗｏｒｄ２ｖｅｃ：

Ａｔｔｅｎｔｉｏｎ

それぞれの間連系に重みを付ける手法

３）問題２

　Seq2seq と HRED、HRED と VHRED の違い

ｓｅｑ２ｓｅｑ：１文の１問１答に対して処理ができる

　ある時系列データからある時系列データを作り出すネットワークのこと。

ＨＲＥＤ：

　Ｓｅｑ２Ｓｅｑに文脈の解釈を付け加えたもの

　文脈の意味をくみ取った変換

ＶＨＲＥＤ：

　ＨＲＥＤは当たり障りのない回答しかできなかったのを、当たり障りのない単語以上の出力をできるようにしたもの。