Subject PreDoc2 Part III 2 RNN Date 19

Attendants

RNN (Recurrent Neural Networks)
回帰結合型ニューラルネットワーク

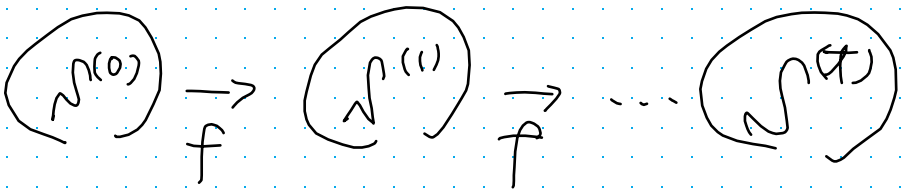
可変長の系列データの処理も可能

計算グラフの展開 (unfold)

回帰の計算を繰り返しを持つ計算グラフで表現

$$s^{(t)} = f(s^{(t-1)}; \theta)$$

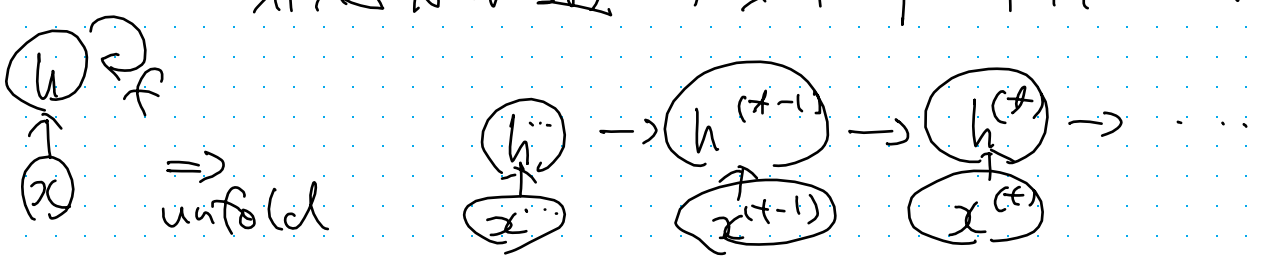
時刻 t における s の定義が、$t-1$ における同じ定義で ref するので回帰的



外部信号 $x^{(t)}$ で作動する動的システム

$$s^{(t)} = f(s^{(t-1)}, x^{(t)}; \theta)$$

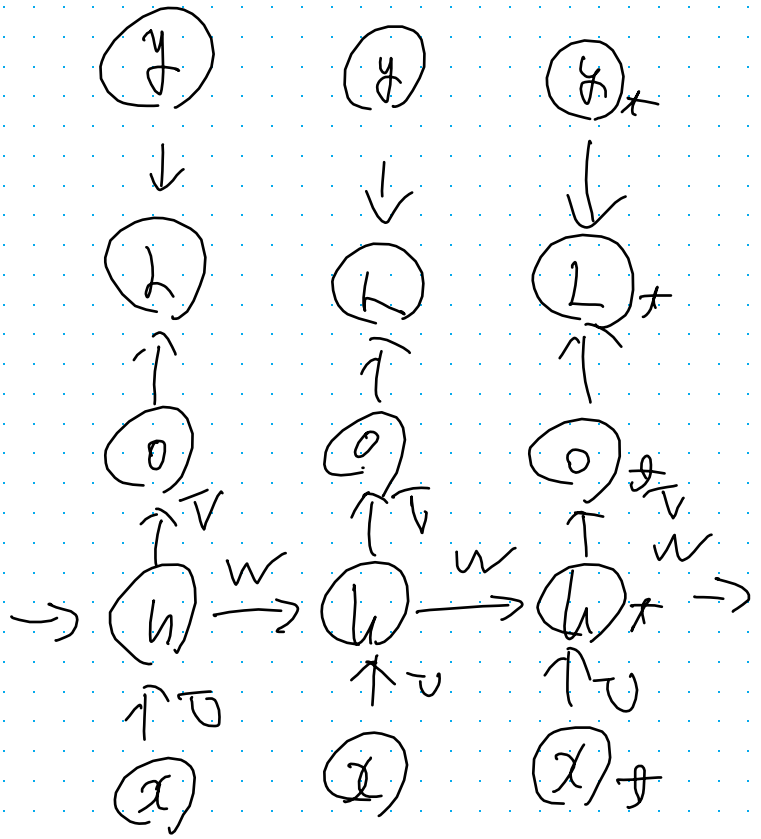
状態 s は過去の全系列の情報を含む



Subject

Date 20

Attendants



$$a^t = b + Wh^{t-1} + Ux^t$$

$$h^t = \tanh(a^t)$$

$$o^t = c + Vh^t$$

$$\hat{y}^t = \mathrm{softmax}(o^t)$$

$$L^{(t)} = \sum_t L^{(t)}$$

$$= -\sum_t \log p_{\mathrm{model}}(y^{(t)} \mid \{x^{(1)}, \dots x^{(t)}\})$$

$x^{(1)} \dots x^{(t)}$ が与えられた下での $y^{(t)}$ の負の対数尤度を $L^{(t)}$ とする

Subject 10.2.1 (DL Book) Date 21

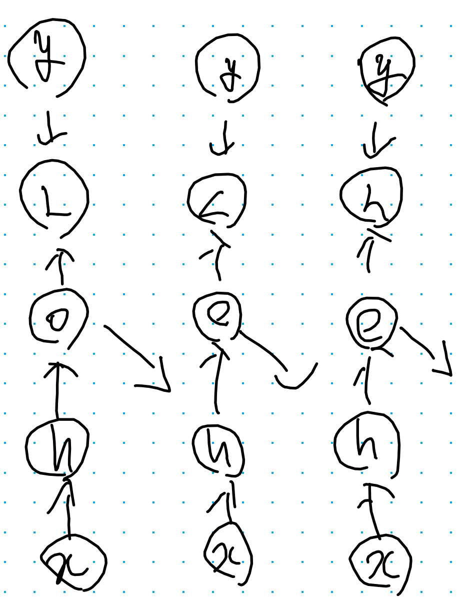
Attendants

BPTT (Backpropagation Through Time)

↳ 順伝播での計算をされた状態を逆伝播で再利用されるまで保持される必要

計算量が大きいのと並列化できない

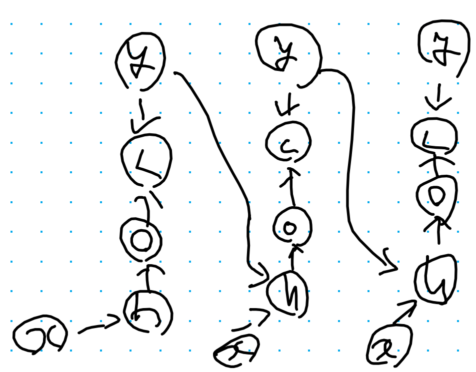
⇒ 回帰的な結合を t-1 stepでの出力と t stepでの状態(隠れユニット)間に持つ



出力ユニットでNNが未来を予測するために使用する過去の情報を全て持っている必要

h→o が次元圧縮されるとダメ

Teacher Forcing



訓練中にモデルがt時刻で真の出力 y_{t-1} を受け取る

⇓

条件付き尤度.

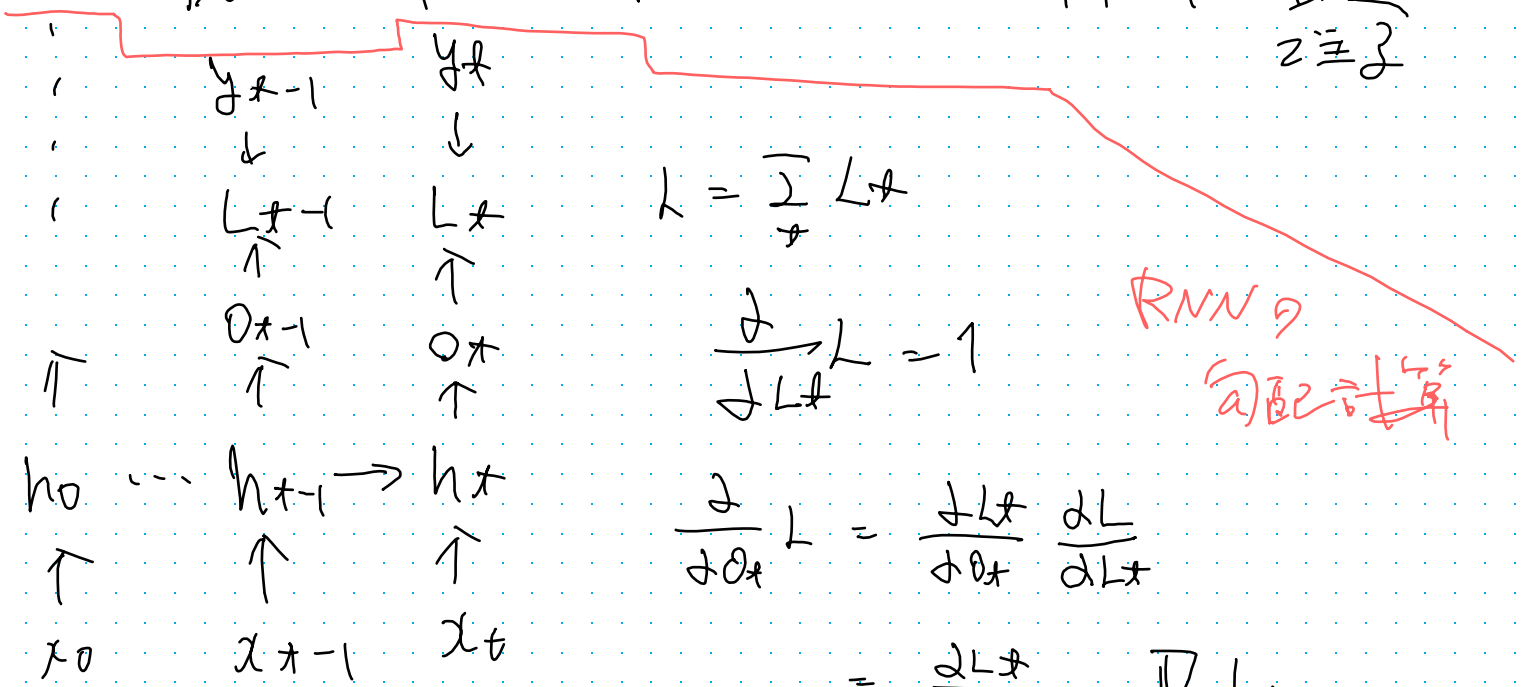
Subject Compute Gradient Date 22

Attendants

Teacher Forcing は

隠れユニット同士の結合がなければ BPTT を回避

できる



$$L = \sum_t L_t$$

$$\frac{\partial}{\partial L_t} L = 1$$

RNN の勾配計算

$$\frac{\partial}{\partial o_t} L = \frac{\partial L_t}{\partial o_t} \frac{\partial L}{\partial L_t}$$

$$= \frac{\partial L_t}{\partial o_t} = \nabla_{o_t} L_t$$

最終 step では h_τ のノードは o_τ のみなので

$$\frac{\partial o_\tau}{\partial h_\tau} = V \Rightarrow \frac{\partial L}{\partial h_\tau} = \frac{\partial o_\tau}{\partial h_\tau} \frac{\partial L_\tau}{\partial o_\tau} \frac{\partial L}{\partial L_\tau}$$

$\because o_t = c + V h_t$

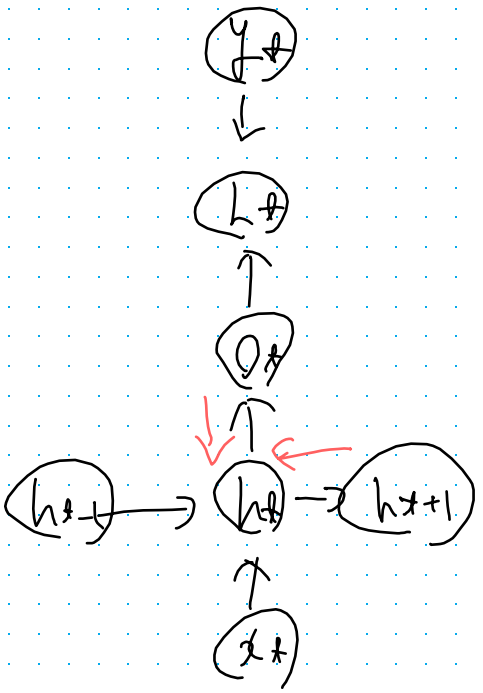
$$\frac{\partial L}{\partial h_\tau} = V^T \nabla_{o_\tau} L$$

Subject

Compute Gradient

Date 23

Attendants



htはノードとして Ot と ht+1 の両方を持つ

$$\frac{\partial h}{\partial h_t} = \frac{\partial o_t}{\partial h_t}\frac{\partial h}{\partial o_t} + \frac{\partial h_{t+1}}{\partial h_t}\frac{\partial h}{\partial h_{t+1}}$$

$$= \frac{\partial o_t}{\partial h_t}\nabla_{o_t} L + \frac{\partial h_{t+1}}{\partial h_t}\nabla_{h_{t+1}} L$$

$$= V^T \nabla_{o_t} L + W^T \mathrm{diag}\left(1-(h_{t+1})^2\right)\nabla_{h_{t+1}} L$$

$o_t = c + V h_t$ より

$h_{t+1} = \tanh(b + W h_t + U x_t)$

ここで $\mathrm{diag}\left(1-h_{t+1}^2\right)$ は要素 $\left(1-(h_{t+1,i})^2\right)$ をもつ対角行列

公式

$$f(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$f'(x) = 1 - (f(x))^2$$

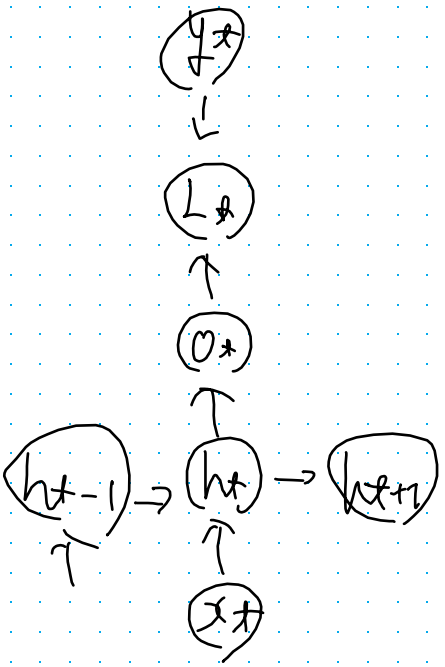
$$\frac{\partial}{\partial h_t}\tanh(b + W h_t + U x_t)$$

$$= 1 - \left(\tanh(b + W h_t + U x_t)\right)^2 W$$

$$= 1 - (h_{t+1})^2 W$$

Subject Compute Gradient Date 24

Attendants

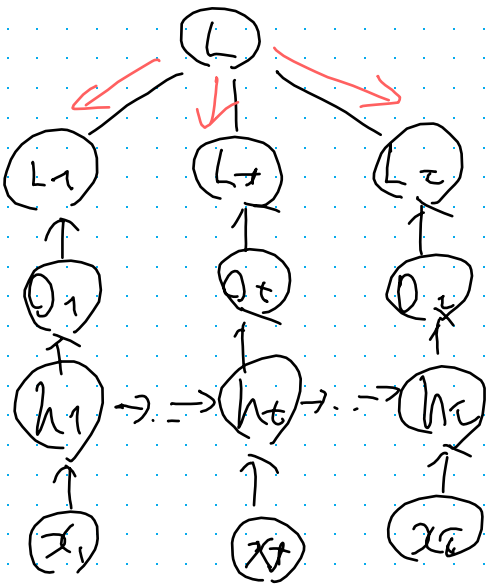


$\hat{y}_t = \mathrm{softmax}(o_t)$

$o_t = c + V h_t$

$h_t = \tanh(a_t)$

$a_t = b + W^T h_{t-1} + U x_t$



$$\frac{\partial L}{\partial c} = \sum_t \frac{\partial o_t}{\partial c} \frac{\partial h_t}{\partial o_t} \frac{\partial L}{\partial L_t}$$

$\frac{\partial o_t}{\partial c} = 1$, $\frac{\partial h_t}{\partial o_t} = \nabla_{o_t} L_t$, $\frac{\partial L}{\partial L_t} = 1$

ここが、h_tとかtの添え字つける

このシグマも

いらないけど、全てのtに関係しているので e.g.

$$\frac{\partial L}{\partial b} = \sum_t \frac{\partial h_t}{\partial b} \frac{\partial L}{\partial h_t}$$

$$= \sum_t \mathrm{diag}\left(1 - (h_t)^2\right) \nabla_{h_t} L$$

$$\frac{\partial L}{\partial V} = \sum_t \frac{\partial o_t}{\partial V} \frac{\partial L}{\partial o_t}$$

$$= \sum_t h_t \nabla_{o_t} L$$

Subject Compute Gradient Date 25

Attendants

$$\frac{\partial L}{\partial W} = \sum_t \frac{\partial h_t}{\partial W} \frac{\partial L}{\partial h_t}$$

$$= \sum_t \mathrm{diag}(1-(h_t)^2)\, h_{t-1} \nabla_{h_t} L$$

$$\frac{\partial L}{\partial U} = \sum_t \frac{\partial h_t}{\partial U} \frac{\partial L}{\partial h_t}$$

$$= \sum_t \mathrm{diag}(1-(h_t)^2)\, x_t \nabla_{h_t} L$$

がんばらなくても算出できる

10.2.3 Directed Graphical Model としての Recurrent Neural Network

$$P(\mathbb{Y}) = \prod_{t=1}^{\tau} P(y^{(t)} \mid y^{(t-1)} \dots y^{(1)})$$

Graphical Model の Edge は他の変数に直接依存しているかどうかを示す e.g マルコフ仮説は

過去の history から $y^{(t)}$ に edge を持たせるのではなく

$\{y^{t-k}, \dots y^{t-1}\}$ から y^{t} へ edge

を持たせる

Subject Date 26

Attendants

① RNNを完全グラフとして解釈

y^tが一次元ベクトル ⇒ Param $O(k^T)$

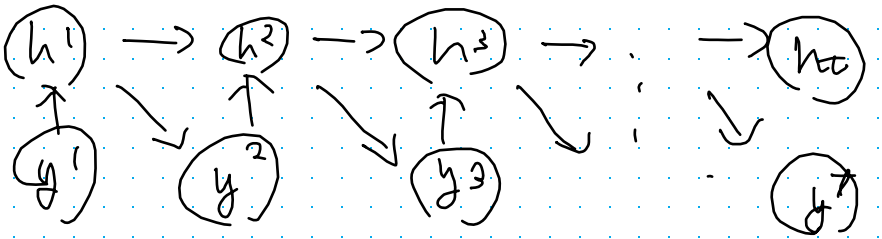
非効率

② 隠れunit h_t is random variableとして扱う

$$h_t = f(h_{t-1}, y_t; \theta)$$

同じパラメータが共有されるので $O(1)$

θ



グラフがしっかりとhを通ることで $y_1 \to h_1 \to y_2$ のように past future がセパレートされる

hは半開の役割 このhを通して $h_1 \to h_t$まで影響をおよぼせる

Subject Date 27

Attendants

Recurrent NN では

① パラメータ共有をしている場合、同じparamが異なる時間 stepで使用できるという前提に基づいている

tにおける変数が予測ステップでの時刻 t+1における変数の conditional dist が stationary である前提

① ② ↙ ↘

② ある stepとその次の stepの関係が常に依存しない

→ 学習が難しい!? LR scheduleの step decayみたいなのは矢張りということかな?

系列の長さの決定

① EOF, EOL (シンボル)を列

② 出力 stopかどうかの判断をするベクトルを出力をモデルに追加

③ 系列の長さて自体を予測 $P(x_1 \dots x_\tau) = P(\tau) P(x_1 \dots x_\tau | \tau)$

任意のRNNにいける のて一般的

Subject Date 2/

Attendants

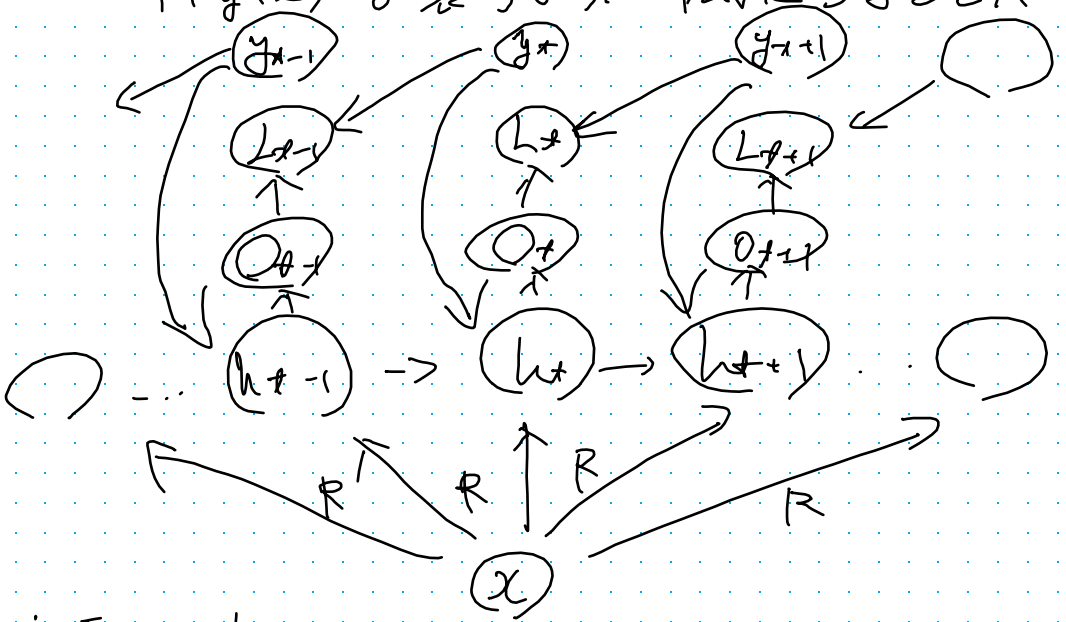
10.2.4 RNNでの系列モデリング

$p(y;\theta)$ を $\theta=w$ として $p(y|w)$

という conditional dist を表すモデルとして解釈できる

$p(y|w)$ のモデルを w が x の関数になるようにして

$p(y|x)$ を表すように拡張することができる



固定長ベクトル

x を系列 y の分布へ写像するRNN

e.g. 画像からの caption 生成

観測された出力系列の各要素 $y^{(t)}$ は、

現stepの入力として

そして 前stepでの正解として

両者の役割を果たす

重み行列Rで

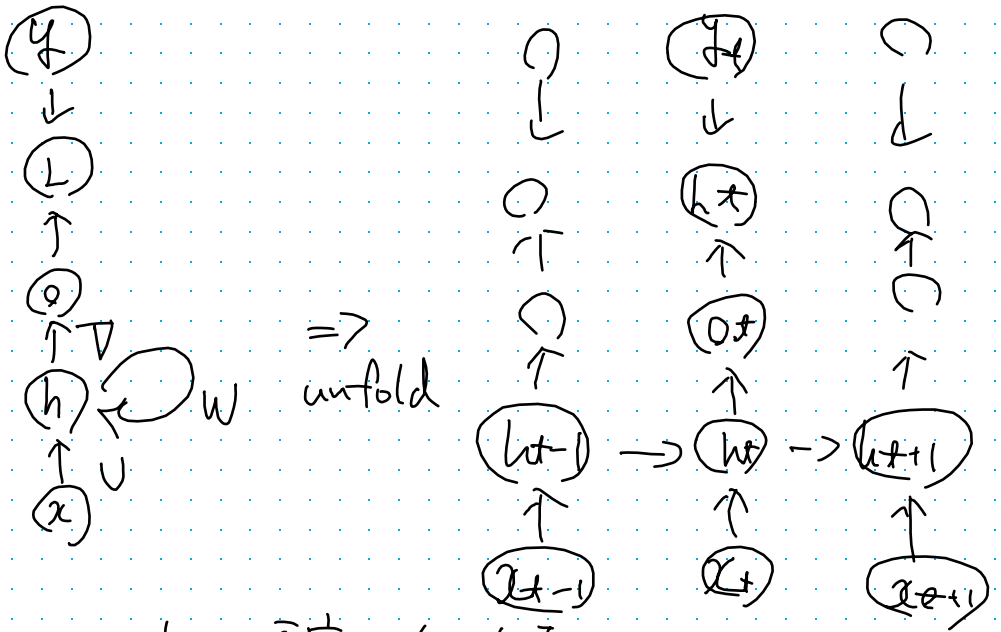
入力と1つ前の相互作用を

parameter化

Subject Date 29

Attendants 条件付き独立性

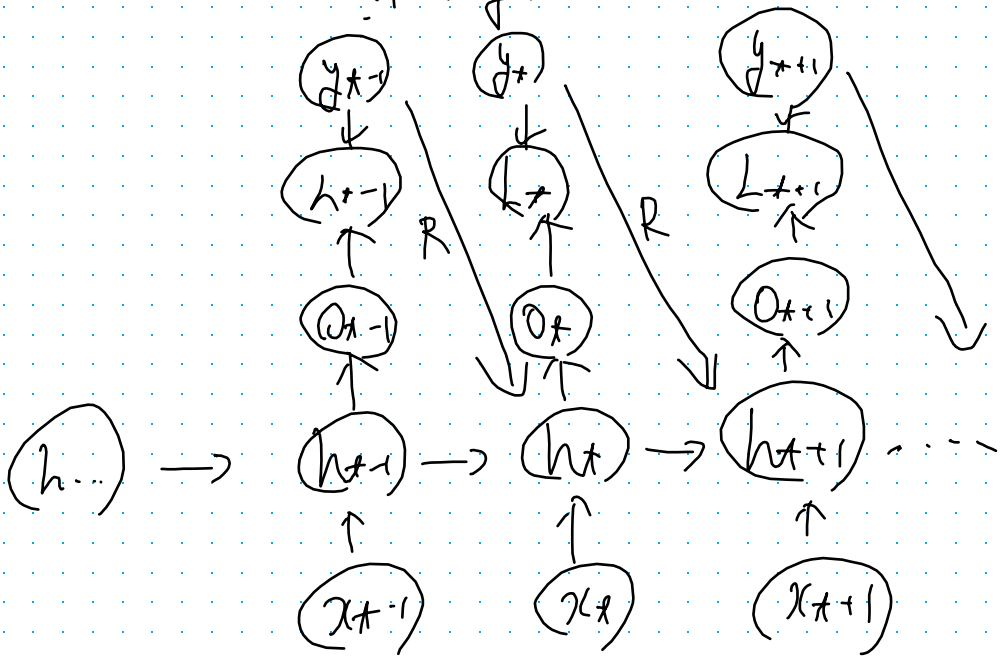
$$P(y_1 \cdots y_\tau \mid x_1 \cdots x_\tau) = \prod_t P(y^t \mid x_1, \cdots x_\tau)$$



この独立を仮定しないとき

⇒直前の出力から現在の状態への接続を含めること

系列 y における依存の分布をモデル化できる



Subject Bidirectional RNN Date 30

Attendants 10.3

入力系列全体に依存する $y^{(t)}$ の予測を出力したいときがある

e.g. 音声認識

(未来も過去も見る必要性がある場合)

10.4 Encoder - Decoder

可変長 → 可変長 系列への写像を行うRNN

e.g. 音声認識・機械翻訳・QA

Encoderは文脈 C を出力

入力系列 $X = (x^{(1)}, \dots x^{(n_x)})$ を要約したベクトル

or ベクトルの系列

Decoderは出力系列 $Y = (y^{(1)} \dots y^{(n_y)})$ を生成するために

文脈 C により条件付けられる

Subject 10.4 Encoder-Decoder Date 3/

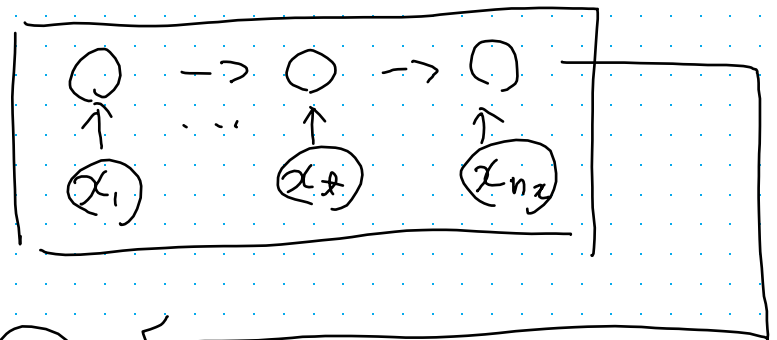
Attendants

Train 系列の x と y について

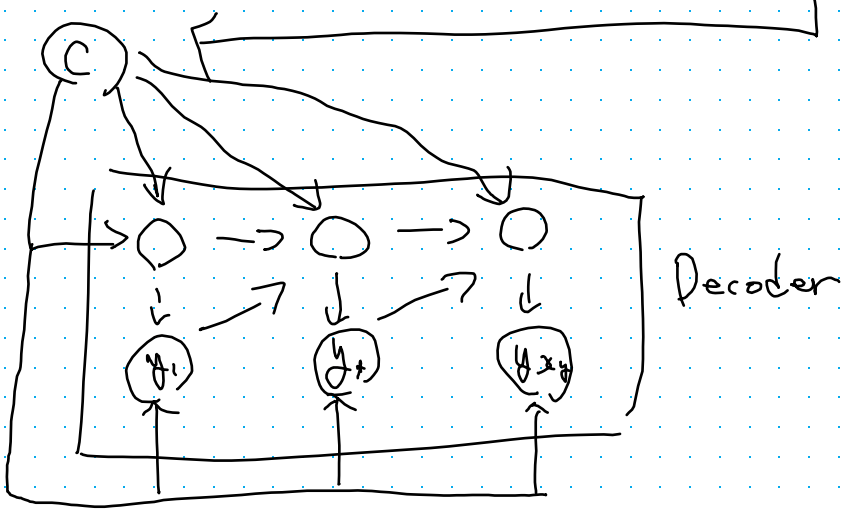
$\log P(y_1 \dots y_{n_y} | x_1 \dots x_{n_x})$ の平均が最大化されるように

Encoder

2つの RNN を学習



通常 Encoder の最後の状態 h_{n_x} を文脈 C として利用



C の次元が低すぎると、長い系列情報を適切に要約できない

solution 1. C を可変長ベクトルにする

2. アテンションメカニズム

Subject 10.5 深層回帰結合ネットワーク Date 32

Attendants

RNNのブロックを深くする

- 複数の層への分解
- ブロックに個別にMLPを持たせる
- 隠れ層→隠れ層の経路にskip接続導入

10.6 再帰型NN

RecursiveNNは RecurrenceNNと違って
深い木構造の計算グラフを持つ

深さをtからlogtに減らせる

→ 長期依存を取り扱うのに有利

最適な木構造自体は未解明

最大のeigenvecを持たないh0の要素は最終的に破棄される

10.7 長期依存性の課題

$h_t = W^T h_{t-1}$

$h_t = (W^t)^T h_0$

ここで $W = Q \Lambda Q^T$

$h_t = Q^T \Lambda^t Q h_0$

固有値は1未満は0に減衰し、1より大きいなら爆発

慎重にステップが選択されたRNNは上手くいかない

Subject　　　　　　　　　　　　　　　　　　Date 33

Attendants

スケーリングが限定されすぎていると、小さな摂動にRobustじゃない
(らしい)

10.8 Echo State Networks (ESNs)

reservoir computing

① 回帰重みと入力重みの学習の難しさを回避するアプローチ

liquid state machine (ESNの連続値unitを隠れ層の二値出力のspikingニューロンを使う)
違いがある

Problem

RNNにおいて $h_t \to h_{t+1}$, $x_t \to h_t$ への写像重みは学習することが難しい

②
ヤコビ行列のspectral半径に固定

③ 重みをうまく初期化する

Subject Date 34

Attendants

10.9 複数時間スケールのための Leaky ユニットとその他手法

1. 時間方向に Skip 接続を追加

2. Leaky ユニット

u_t の移動平均 μ_t

$$\mu_t \leftarrow \alpha \mu_{t-1} + (1-\alpha) u_t$$

α が1に近いと μ_t は過去の情報を長期間記憶し、

α が0に近いと過去の情報は急速に破棄される

3. 接続の削除

重みが1のを削除して、それより長い接続に replace

10.10 Long Short Term Memory & RNN w/ Gate

gated RNN

e.g. LSTM

gated recurrent unit (GRU)

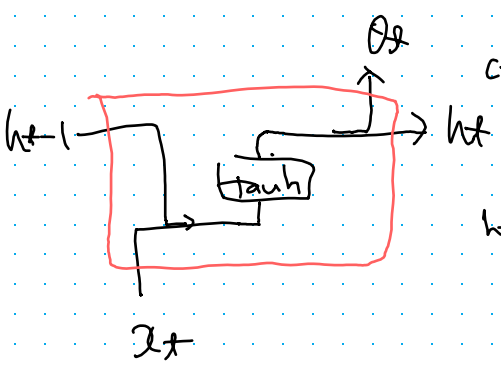
これを行うために、手動で選択した定数または、パラメータによる接続重みを使った。
GRUでは、これを時間stepごとに変更可能な接続重みに一般化した

Leaky Unit と同様にゲート付きRNNは消失も爆発もしない勾配を持ち、それを伝播させる

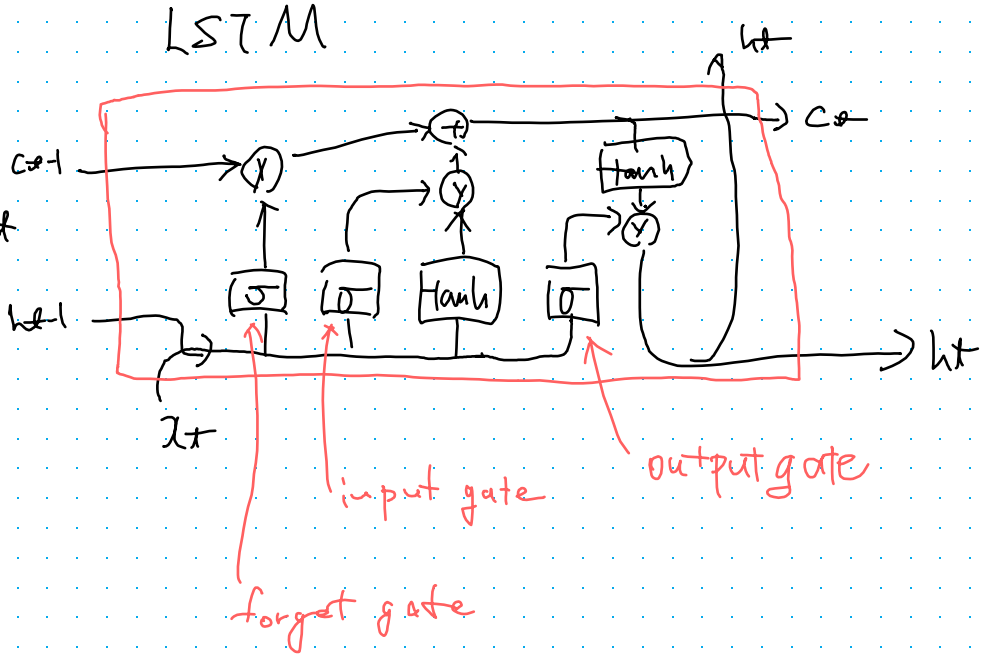
Subject LSTM (10.11) Date 35

Attendants

RNN



LSTM



LSTM 回帰結合型ネットワークは、

入力と回帰ユニットのアフィン変換に要素ごとの非線形変換を実施に単純に適用するユニットのかわりに、通常のRNNの回帰に加えて、内部回帰を持つセルをLSTMセルと言う.

GRU (Gated Reccurence Unit) と呼ばれる

ユニットを持ったゲートつきRNNが最近提案されている。

↓

LSTMとの大きな違いは単一のゲートユニットが forged states のUnitの要素の決定と同時に制御する点である

Subject Date 36

Attendants 10.12 長期依存性の optim

grad clipping

↳ RNNなどの極めて非線形なfuncは
非常に大きい or 小さい gradを持つ
傾向がある

10.11.2 情報の流れを促進するTimeの
normalize

↙

grad 消失への対応

gradの積が1になるように

↳ LSTMの利用が1つのアプローチ
(self loop と gate)

10.13 明示的メモリ.

DONE

Subject PreDoc2 Part III [2 RNN] Date 9/7

Attendants Autumn 2021

Sequence parity function

input: sequence (binary)

output: whether num of 1's seen so far is odd or even

odd → 1

even → 0

e.g. input [0, 1, 0, 1, 1, 0]

the parity sequence is [0, 1, 1, 0, 1, 1]

Design vanilla RNN to implement the parity func

Your implementation should include the equations of your hidden units and the details about derivations w/ diff input of x_t (0 or 1)

Subject Date 3/8

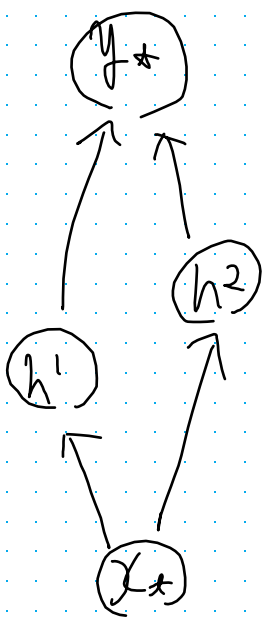
Attendants

y_t Parity Bits 0 1 1 0 1 1 →

x_t input 0 1 0 1 1 0

Each parity bit is the XOR of input and the previous parity bit

Parity is a classic example of a problem that's hard to solve with shallow feed-forward net but easy to solve w/ RNN



XOR

y_{t-1}	x_t	y_t
0	0	0
0	1	1
1	0	1
1	1	0

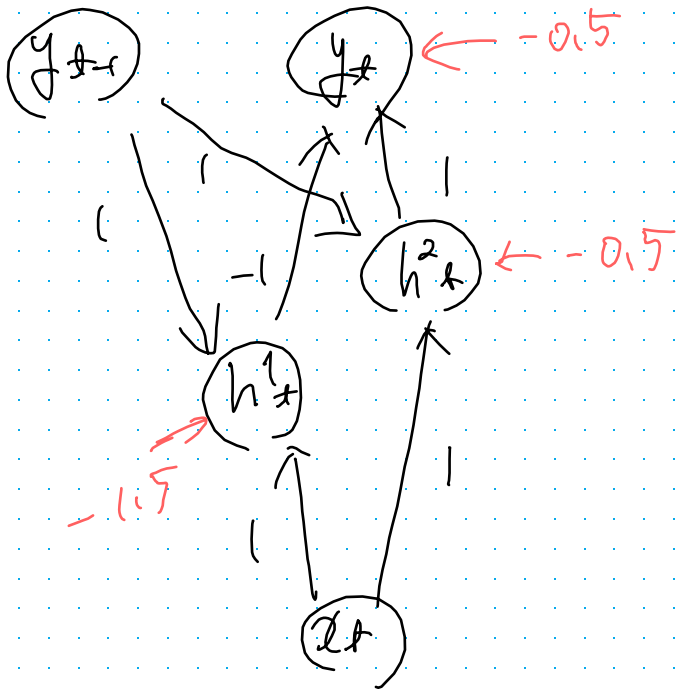
linear model cannot compute XOR

↳ then use hidden layer

Subject All hidden and output units are binary threshold unit

Date 39

Attendants



$h^1_t = f(y_{t-1} + x_t - 1.5)$

$h^2_t = f(y_{t-1} + x_t - 0.5)$

$y_t = f(h^2_t - h^1_t - 0.5)$

f is hard threshold activation function

AND OR

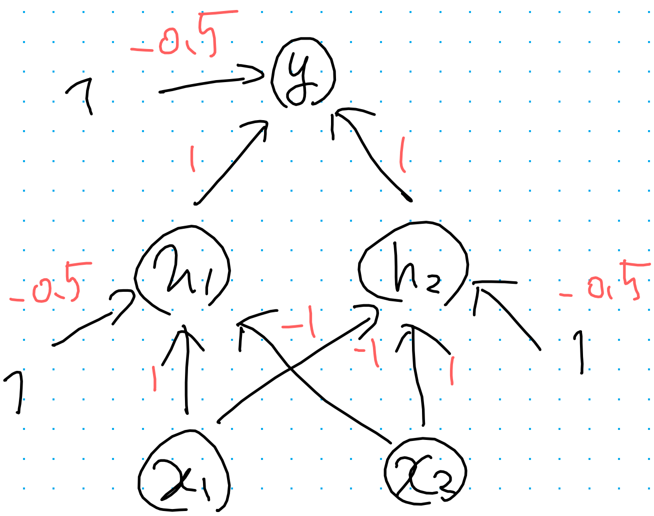
y_{t-1}	x_t	h^1_t	h^2_t	y_t
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

AND

OR

Subject CSC321 Lecture 5 Date 40

Attendants Designing a network to compute XOR



1 : bias term

$h_1 = f(x_1 - x_2 - 0.5)$

$h_2 = f(x_2 - x_1 - 0.5)$

$y = f(h_1 + h_2 - 0.5)$

x_1	x_2	h_1	h_2	y
0	0	0 0	0 0	0
0	1	0 0	1 1	1
1	0	1 0	0 1	1
1	1	0 1	0 1	0

← この赤い文字は fを計算した結果.

↑ AND ではない ↑ OR ではない ↑ XOR

$$f(z) = \begin{cases} z > 0 \rightarrow 1 \\ z \leq 0 \rightarrow 0 \end{cases}$$

End of Note.

Subject

Date

Attendants