RNN

岡田崇

2026 年 6 月 14 日

目次

1 RNN とは何か

RNN（Recurrent Neural Network）は、**系列データ**を扱うためのニューラルネットワークである。系列データとは、データ点の**順序**に意味があるデータを指す。

系列データの例

データ	系列の単位
文章	単語、文字、サブワード
音声	時刻ごとの音響特徴量
株価	日ごと、分ごとの価格や出来高
動画	フレーム列
センサーデータ	時刻ごとの測定値

通常の全結合ニューラルネットワークは、入力を基本的には独立したベクトルとして扱う。一方で、RNN は**過去の入力の情報**を内部状態として持ち続けながら、系列を 1 ステップずつ処理する。

Remark: RNN の直感

RNN の本質は、**「今の判断に、過去の文脈を使う」**ことである。現在の入力だけでは意味が決まらない場合でも、過去の入力を隠れ状態に蓄積しておけば、より文脈に沿った予測ができる。

2 RNN の基本アイデア

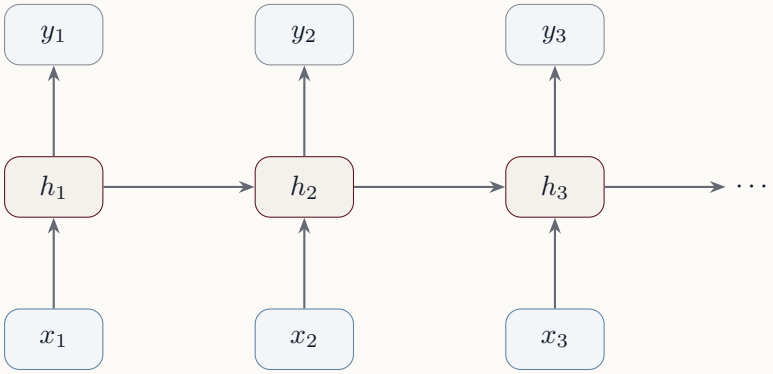
時刻 t における RNN は、現在の入力 x_t と、直前の隠れ状態 h_{t-1} を受け取る。そして、新しい隠れ状態 h_t を計算する。

Simple RNN の基本式

$$h_t = \phi\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right), \tag{1}$$

$$y_t = g\left(W_{hy}h_t + b_y\right). \tag{2}$$

ここで、ϕ は tanh や ReLU などの活性化関数、g はタスクに応じた出力関数である。分類問題では $g = \mathrm{softmax}$ を使うことが多い。



上の図は、RNN を時間方向に展開したものである。各時刻で同じ重み W_{xh}, W_{hh}, W_{hy} を共有している点が重要である。

重み共有の意味

RNN は、すべての時刻で**同じ変換規則**を使う。したがって、系列長が変わっても同じモデルを適用できる。

3 RNN で扱える入出力形式

RNN は、入力系列と出力系列の形に応じて、いくつかのパターンで使われる。

3.1 Many-to-One

系列全体から 1 つの出力を得る形式である。

- 文の感情分類
- 動画分類
- 時系列データからの異常検知

典型的には最後の隠れ状態 h_T を使って、

$$\hat{y} = \mathrm{softmax}(W h_T + b)$$

のように分類する。

3.2 One-to-Many

1 つの入力から系列を生成する形式である。

- 画像から説明文を生成する画像キャプション

- 初期条件から音楽や文字列を生成するタスク

3.3 Many-to-Many

系列を入力し、系列を出力する形式である。

- 機械翻訳
- 品詞タグ付け
- 音声認識
- 文字列変換

Remark: 出力のタイミング

RNN は、**各時刻で出力する**ことも、**最後だけ出力する**こともできる。タスク設計では、「どの隠れ状態を使って予測するか」を明確にする必要がある。

4 RNN の学習と損失関数

RNN も通常のニューラルネットワークと同じく、損失関数を定義し、勾配降下法で重みを更新する。

時刻ごとに出力がある場合、全体の損失は各時刻の損失の和として書ける。

系列全体の損失

$$L = \sum_{t=1}^{T} L_t\big(y_t, \hat{y}_t\big).$$

たとえば品詞タグ付けのような系列ラベリングでは、各時刻の単語に対してラベルを予測し、それぞれにクロスエントロピー損失を計算する。

学習で更新されるパラメータ

RNN では、主に次のパラメータを更新する。

$$W_{xh},\quad W_{hh},\quad W_{hy},\quad b_h,\quad b_y.$$

特に W_{hh} は、過去の状態を次の状態へ伝える**再帰重み**であり、RNN らしさを生む中心的なパラメータである。

5 BPTT: Backpropagation Through Time

RNN の学習では、通常の誤差逆伝播を時間方向に展開して適用する。これを **BPTT**（Backpropagation Through Time）と呼ぶ。

BPTT の見方

RNN を時間方向に展開すると、次のような深いニューラルネットワークとして見なせる。

$$(x_1, h_0) \to h_1 \to h_2 \to \cdots \to h_T.$$

この展開された計算グラフに対して、通常の誤差逆伝播を行う。

Remark: RNN は時間方向に深い

系列長 T が大きいほど、RNN は時間方向に深いネットワークになる。つまり、長い系列では**学習が不安定になりやすい**。

BPTT の概念的な流れ

```
forward:
  h_0 = initial_state
  for t = 1, ..., T:
      h_t = RNNCell(x_t, h_{t-1})
      y_t = Output(h_t)
      accumulate loss L_t

backward:
  backpropagate from L_T, ..., L_1 through the unrolled graph
  update shared parameters W_xh, W_hh, W_hy
```

6 勾配消失と勾配爆発

RNN の大きな問題は、長い系列を扱うときに**勾配消失**や**勾配爆発**が起きやすいことである。

6.1 勾配消失問題

RNN では、遠い過去の隠れ状態が現在の損失にどの程度影響するかを計算すると、次のようなヤコビアンの積が現れる。

時間方向の勾配の積

$$\frac{\partial h_T}{\partial h_t} = \frac{\partial h_T}{\partial h_{T-1}} \frac{\partial h_{T-1}}{\partial h_{T-2}} \cdots \frac{\partial h_{t+1}}{\partial h_t}.$$

この積の各要素の大きさが 1 より小さいと、何度も掛け算するうちに勾配が非常に小さくなる。その結果、遠い過去の入力に対する学習信号が消えてしまう。

Remark: 長期依存が苦手になる理由

単純な RNN が長い文脈を苦手とする主な理由は、**遠い過去へ誤差信号が届きにくい**ことである。これは単に「記憶容量が小さい」という話ではなく、学習時の勾配そのものが弱くなる問題である。

6.2 勾配爆発問題

逆に、ヤコビアンの積の大きさが 1 を大きく超える場合、勾配が急激に大きくなる。これを**勾配爆発**という。

勾配爆発が起きると、重みの更新量が極端に大きくなり、学習が不安定になる。実務では**勾配クリッピング**がよく使われる。

勾配クリッピング

勾配ノルムがしきい値 c を超えた場合、次のようにスケールする。

$$\nabla \leftarrow \frac{c}{\|\nabla\|}\nabla \qquad \text{if } \|\nabla\| > c.$$

7 LSTM: Long Short-Term Memory

LSTM は、単純な RNN の勾配消失問題を緩和するために広く使われるモデルである。LSTM では、通常の隠れ状態 h_t に加えて、**セル状態** c_t を持つ。

LSTM の中心アイデア

LSTM は、情報を**忘れる**、**書き込む**、**出力する**という操作をゲートで制御する。これにより、必要な情報を長く保持しやすくなる。

7.1 3 つのゲート

ゲート	役割
忘却ゲート f_t	過去のセル状態 c_{t-1} のうち、どの情報を残すかを決める。
入力ゲート i_t	新しい候補記憶 $\tilde{c}_t$ のうち、どの情報を書き込むかを決める。
出力ゲート o_t	セル状態から、どの情報を隠れ状態 h_t として出すかを決める。

7.2 LSTM の計算式

LSTM cell

$$f_t = \sigma\left(W_f[h_{t-1} \,\|\, x_t] + b_f\right), \tag{3}$$

$$i_t = \sigma\left(W_i[h_{t-1} \,\|\, x_t] + b_i\right), \tag{4}$$

$$\tilde{c}_t = \tanh\left(W_c[h_{t-1} \,\|\, x_t] + b_c\right), \tag{5}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{6}$$

$$o_t = \sigma\left(W_o[h_{t-1} \,\|\, x_t] + b_o\right), \tag{7}$$

$$h_t = o_t \odot \tanh(c_t). \tag{8}$$

ここで、σ はシグモイド関数、$\odot$ は要素ごとの積である。シグモイド関数の出力は 0 から 1 なので、ゲートは情報を通す量を調整する**バルブ**として解釈できる。

Remark: セル状態が重要

LSTM の強さは、c_t が時間方向に比較的直接伝わることにある。これにより、単純な RNN よりも**長期依存関係**を学習しやすい。

8 GRU: Gated Recurrent Unit

GRU は、LSTM を簡略化した RNN である。LSTM よりパラメータ数が少なく、計算も軽いことが多い。

GRU には主に 2 つのゲートがある。

ゲート	役割
更新ゲート z_t	過去の隠れ状態をどれだけ残すかを決める。
リセットゲート r_t	候補隠れ状態を作るとき、過去の情報をどれだけ無視するかを決める。

GRU cell

$$z_t = \sigma\left(W_z[h_{t-1} \,\|\, x_t] + b_z\right), \tag{9}$$

$$r_t = \sigma\left(W_r[h_{t-1} \,\|\, x_t] + b_r\right), \tag{10}$$

$$\tilde{h}_t = \tanh\left(W_h[(r_t \odot h_{t-1}) \,\|\, x_t] + b_h\right), \tag{11}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t. \tag{12}$$

LSTM と GRU の使い分け

精度を最優先して比較する場合は、LSTM と GRU の両方を試す価値がある。計算量やモデルサイズを抑えたい場合は、GRU が有力な候補になる。

9 Bidirectional RNN

通常の RNN は、入力を左から右、つまり過去から未来へ処理する。しかし、自然言語処理では後ろの文脈が前の単語の意味を決めることも多い。

Remark: 未来の文脈も使いたい

「銀行に行ってお金をおろした」という文では、「銀行」が金融機関を意味することは後ろの語句から分かる。したがって、系列全体が事前に与えられているタスクでは、未来方向の情報も使うと有利である。

Bidirectional RNN は、前向き RNN と後ろ向き RNN を同時に使う。

双方向隠れ状態

$$\overrightarrow{h_t} = \mathrm{RNN}_{\mathrm{fwd}}(x_1, \ldots, x_t), \tag{13}$$

$$\overleftarrow{h_t} = \mathrm{RNN}_{\mathrm{bwd}}(x_T, \ldots, x_t), \tag{14}$$

$$h_t = [\overrightarrow{h_t} \,\|\, \overleftarrow{h_t}]. \tag{15}$$

前後両方の文脈を使えるため、品詞タグ付け、固有表現抽出、音声認識などで有効である。ただし、未来の入力が必要なので、リアルタイム処理には不向きな場合がある。

10 Encoder-Decoder と Attention

RNN は、系列から系列への変換にも使われる。代表例が **Encoder-Decoder** モデルである。

10.1 Encoder

Encoder は入力系列を順に読み込み、文全体の情報を隠れ状態へ圧縮する。

$$(x_1, x_2, \ldots, x_T) \longrightarrow h_T.$$

10.2 Decoder

Decoder は Encoder が作った表現をもとに、出力系列を 1 ステップずつ生成する。

$$h_T \longrightarrow (y_1, y_2, \ldots, y_S).$$

Encoder-Decoder の例

機械翻訳では、Encoder が原文を読み込み、Decoder が翻訳文を生成する。

「私は学生です」$\longrightarrow$ “I am a student.”

10.3 Attention

固定長ベクトル h_T だけに入力文全体を押し込めると、長い系列では情報が失われやすい。そこで導入されるのが **Attention** である。

Attention は、Decoder が各時刻で出力を生成するときに、入力系列のどの部分を参照すべきかを動

的に決める仕組みである。

Remark: Attention の直感

Attention は、翻訳中に元文の重要な単語へ**視線を向ける**仕組みと考えられる。RNN の隠れ状態だけに全情報を詰め込む必要がなくなるため、長い系列に強くなる。

11 実装上の注意

RNN を実装するときは、数式だけでなく、データの扱いにも注意が必要である。

11.1 Padding と Mask

実際の系列データは長さがばらばらである。ミニバッチでまとめて処理するためには、短い系列に **padding** を入れて長さをそろえる。

> **Padding の例**
>
> ```
> I like cats <PAD> <PAD>
> I really like small cats
> ```

ただし、padding 部分を損失計算に含めると、モデルが意味のない ⟨PAD⟩ を学習してしまう。そのため、**mask** を用いて padding 部分を無視する。

11.2 Teacher Forcing

系列生成では、Decoder の次の入力として、前の時刻の出力を使う。しかし学習初期の予測は不安定なので、正解系列の前の単語を入力することがある。これを **Teacher Forcing** という。

Remark: 学習時と推論時のずれ

Teacher Forcing は学習を安定させるが、推論時には正解単語を使えない。このため、学習時と推論時の入力分布がずれることがある。

11.3 Truncated BPTT

長い系列に完全な BPTT を行うと、計算量とメモリ使用量が大きくなる。そこで、一定の長さごとに系列を区切って逆伝播する。これを **Truncated BPTT** という。

例

全系列長が 1000 で、BPTT 長を 50 に設定した場合、50 ステップごとに逆伝播を行う。これにより、計算資源を抑えながら長い系列を扱える。

11.4 勾配クリッピングと初期化

RNN では勾配爆発が起きやすいため、**gradient clipping** を設定することが多い。また、再帰重み W_{hh} の初期化や活性化関数の選択も学習安定性に影響する。

12 例: 感情分類タスク

入力文を 1 つ受け取り、ポジティブかネガティブかを分類する問題を考える。

入力例

$$\text{「この映画はとても面白かった」}$$

まず文をトークンに分割し、各トークンを埋め込みベクトルに変換する。

$$x_1, x_2, \dots, x_T.$$

RNN はこれらを順番に読み込み、最後の隠れ状態 h_T を文全体の表現として使う。

> **文分類の出力**
>
> $$\hat{y} = \mathrm{softmax}(W h_T + b).$$

Remark: 最後の隠れ状態の意味

Many-to-One の分類では、h_T は「系列全体を読んだ後の要約表現」と見なされる。ただし、長い文では最後の状態だけでは情報が不足することもある。その場合、Attention や Pooling を組み合わせることがある。

13 RNN・LSTM・GRU の比較

モデル	特徴	長所	短所
Simple RNN	最も基本的な再帰モデル。隠れ状態のみを持つ。	構造が単純で理解しやすい。	長期依存関係が苦手。勾配消失が起きやすい。
LSTM	セル状態と 3 つのゲートを持つ。	長期依存を学習しやすい。	パラメータ数が多く、計算が重い。
GRU	LSTM を簡略化し、2 つのゲートを持つ。	軽量で実装しやすく、多くのタスクで高性能。	タスクによっては LSTM の方がよい場合がある。

実務的な判断

短い系列や単純なベースラインなら Simple RNN でもよい。**長期依存**が重要なら LSTM や GRU を使う。計算量を抑えたい場合は GRU がよい候補になる。

14 RNN の長所と短所

長所

- 可変長の系列を扱える。
- 過去の情報を状態として保持できる。
- 時系列データや言語データに自然に適用できる。
- ストリーミング処理に向いている。

短所

- 長期依存関係の学習が難しい。
- 勾配消失・勾配爆発が起きやすい。
- 時刻方向に逐次計算するため、並列化しにくい。
- 長い系列では計算が遅くなりやすい。

15 まとめ

本講義の要点

- RNN は、**過去の情報を隠れ状態 h_t に保存**しながら系列を読むモデルである。
- 基本式は $h_t = \phi(W_{xh}x_t + W_{hh}h_{t-1} + b_h)$ である。
- 学習では **BPTT** を用い、時間方向に展開した計算グラフで逆伝播する。
- 単純な RNN は**勾配消失**と**勾配爆発**に弱い。
- **LSTM** と **GRU** は、ゲートにより長期依存関係を扱いやすくする。
- **Bidirectional RNN** は前後の文脈を利用できる。
- **Encoder-Decoder** と **Attention** は、系列変換タスクで重要な拡張である。

Remark: RNN の一言まとめ

RNN の本質は、**「状態を持つニューラルネットワーク」**である。系列を順番に読み、過去の情報を現在の判断に反映する。

付録: 記号一覧

記号	意味
x_t	時刻 t の入力ベクトル。
h_t	時刻 t の隠れ状態。過去の情報を含む。
y_t	時刻 t の出力。
W_{xh}	入力から隠れ状態への重み。
W_{hh}	前の隠れ状態から次の隠れ状態への再帰重み。
W_{hy}	隠れ状態から出力への重み。
c_t	LSTM のセル状態。長期記憶の役割を持つ。
f_t, i_t, o_t	LSTM の忘却ゲート、入力ゲート、出力ゲート。
z_t, r_t	GRU の更新ゲート、リセットゲート。
$\odot$	要素ごとの積。