深層学習

系列データのためのネットワーク

会津大学 コンピュータ理工学研究科 コンピュータ情報システム学専攻 髙橋輝

系列データ

個々の要素が順序付きの集まり

$$\mathbf{x}^1, \mathbf{x}^2, \mathbf{x}^3, \cdots, \mathbf{x}^T$$

として与えられるデータを系列データと呼ぶ

約束

- 要素の並びをインデックス$1, 2, 3, \cdots, T$で表し, tを時刻と呼ぶ.
  - 時刻は物理的な時間と対応するとは限らない
- 個々のデータが順序を持っている, つまり, 並びに意味を持っていればよい.

今回扱う問題

1. テキスト to 多クラス (Text to Multi-class)

- レストランの利用客の感想を3段階で評価.
- 文を構成する各単語をベクトルで表現. (下に例示)
  $\mathbf{x}^1 = \text{`}\mathtt{They}\text{`}, \mathbf{x}^2 = \text{`}\mathtt{have}\text{`}, \cdots, \mathbf{x}^{15} = \text{`}\mathtt{better}\text{`}$
- データの最小単位は一つの文$(\mathbf{X}_n = (\mathbf{x}^1, \mathbf{x}^2, \cdots, \mathbf{x}^{T_n}))$.
- 単語数は自由なので, 系列長T_nも自由.

今回扱う問題

2. 音声認識(Speech Recognition)

- 発話を記録した時間信号から発話内容を推定する.
- 信号は一定の周期で標本化され,
  量子化されたデジタルデータ(=一般的な音声データ)
- 前処理
  - 方法の例: $10ms$間隔で$25ms$幅の窓で切り出し, 周波数スペクトルの分布情報を取り出して, 特徴ベクトルの系列$(\mathbf{x}^1, \mathbf{x}^2, \cdots,)$を得る.
- 入力に前処理を行ったデータを取り, 発話を構成する音素(phoneme) or
  発話内容を直接表す文字列を推定する.

1, 2共に, 出力は入力と異なる長さの系列を出力できる必要がある

リカレントニューラルネットワーク

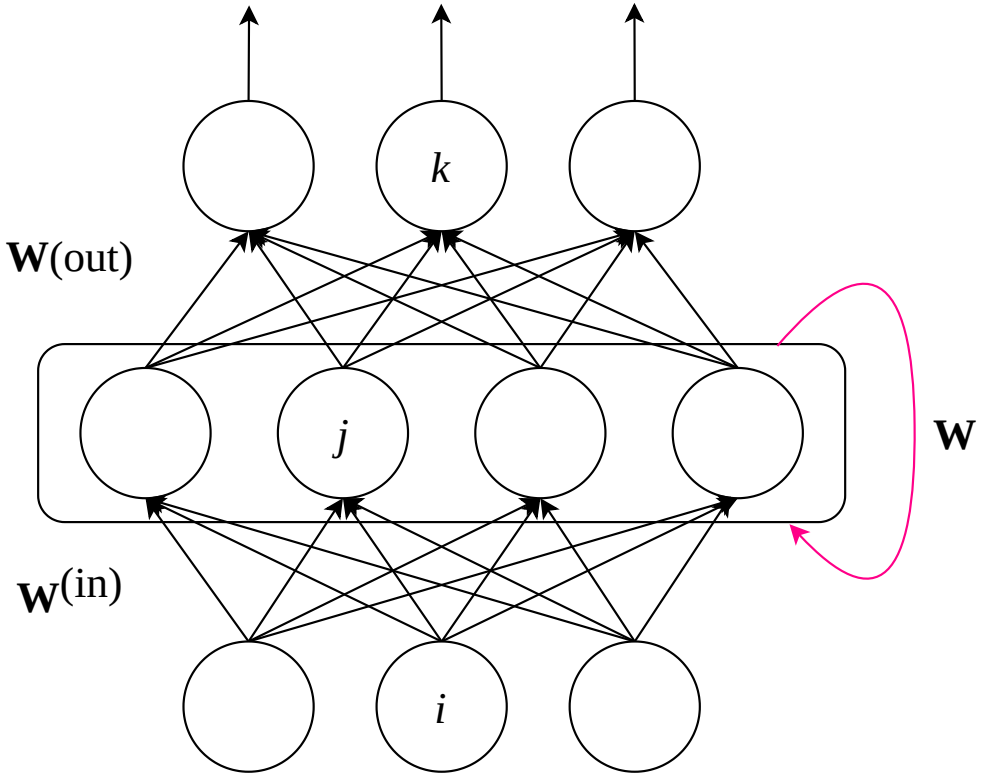
1. リカレントニューラルネットワーク(RNN)とはなんぞや？

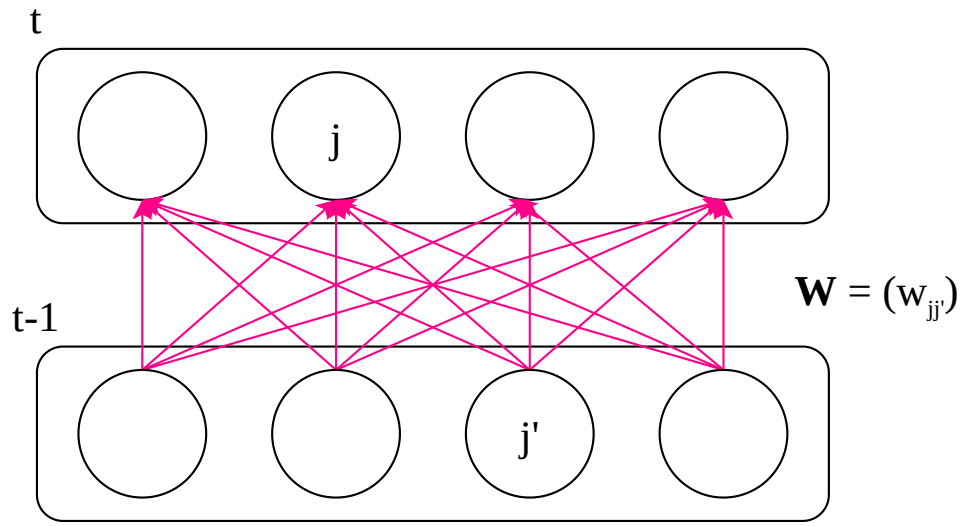
A. リカレントニューラルネットワーク(Recurrent Neural Network)とは,

内部に(有向)閉路を持つニューラルネットワーク　　の総称である.

- 例として,
  - Elman Network
  - Jordan Network
  - Time Delay Network
  - Echo State Network など様々なものがあるが, 始めは単純なものを考える.

シンプルなRNN





前図のように,中間層のユニットの出力が自分自身に重み付きで戻されるRNNを考える. この自分自身に戻ってくるパスを帰還路と呼ぶ.

この構造により, 中間層のユニットは, ひとつ前の状態を**覚える**ことができる.

また, このユニットは, ひとつ前の出力と, 現在の入力の両方を考慮して状態が変わるため, 振る舞いを**動的に変化させる**ことができる.

この二つの特性により, この単純なRNNは系列データ中の"文脈"を捉えることが期待される.

RNNは各時刻tにつき1つの入力$\mathbf{x}^t$を受け取り, 出力$\mathbf{y}^t$を出力する.
これは, 入力と同じ長さの系列を出力することを意味する.

過去に受け取った入力(理論上はすべて)が帰還路を通して出力に影響を与える.

順伝播型ネットワークとの比較

	順伝播型ネットワーク	RNN
帰還路	なし	あり
写像	$\mathbf{x}^t \mapsto y$	$(\mathbf{x^0}, \mathbf{x^1}, \cdots, \mathbf{x^t}) \mapsto \mathbf{y}$

このRNNは, 系列データについて, 順伝播型ネットワークと同じ万能性を持つ.

RNNの順伝播

系列$(\mathbf{x^0}, \mathbf{x^1}, \cdots, \mathbf{x^t})$を順に入力すると, 系列$(\mathbf{y^0}, \mathbf{y^1}, \cdots, \mathbf{y^t})$を出力する.

$$y : (\mathbf{x}^0, \mathbf{x}^1, \cdots, \mathbf{x}^t) \mapsto (\mathbf{y}^0, \mathbf{y}^1, \cdots, \mathbf{y}^t)$$

この計算の詳細を後のスライドで説明するが, その前に定義を行う.

定義

- $\mathbf{x}^t = (x_i^t)$: ネットワークの入力
- $\mathbf{u}^t = (u_j^t), \mathbf{z}^t = (z_j^t)$: 中間層ユニットの入出力
- $\mathbf{v}^t = (v_k^t), \mathbf{y}^t = (y_k^t)$: 出力層ユニットの入出力
- $\mathbf{d}^t = (d_k^t)$: 目標とする出力

続・定義

- $\mathbf{W}^{(in)} = (w_{ji}^{(in)})$: 入力層と中間層のユニット間の重み
- $\mathbf{W} = (w_{j'j})$: 帰還路の結合重み
- $\mathbf{W}^{(out)} = (w_{kj}^{(out)})$: 中間層と出力層のユニット間の重み
- $x_0^t = 1, z_0^t = 1$: 各層の0番目はバイアスを表現するため, 常に1を出力するユニットを配置する.
    - つまり, 中間層のバイアスは, $w_{j0}^{(in)}$, $w_{k0}^{(out)}$ によって表現される.

時刻$t-1$のユニットから時刻tのユニットへの全結合が存在する.
そのため, 中間層への入力は,

$$\mathbf{u}^t = \mathbf{W}^{(in)}\mathbf{x}^t + \mathbf{W}z_{t-1}$$

となり, 中間層の出力は, 活性化関数fを経由して,

$$\mathbf{z}^t = f(\mathbf{u}^t)$$

と表示される.

この中間層の出力を利用して, RNNの出力$\mathbf{y}^t$を計算する.
出力層には, 中間層の重みとは独立に出力層にも重み$\mathbf{W}^{(out)}$が存在するため, 出力層ユニットへの入力$\mathbf{v}^t$は,

$$\mathbf{v}^t = \mathbf{W}^{(out)}\mathbf{z}^t$$

と計算され, 活性化関数$\mathbf{f}^{(out)}$を経由して, 出力$\mathbf{y}^t$が以下のように計算される.

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

なお, 層出力の正規化には, バッチ正規化ではなく, インスタンス正規化を利用する.

問題への適用

RNNの出力層の設計は, 通常のNNと同じ.
以下は, 例としてクラス分類の場合を考える.
出力層は, クラス数(C)と同様の数のユニットを並べ, ソフトマックス関数を活性化関数に選ぶ.

- 出力系列: $\mathbf{y}^1, \cdots, \mathbf{y}^t$
- 目標値: $\mathbf{d}^1, \cdots, \mathbf{d}^t$
- 訓練サンプルを$n = 1, \cdots, N$として, サンプルnの系列長をT_nと書く.
  すると, 損失関数は

$$E(\mathbf{w}) = -\sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{C}\mathbf{d}_{nk}^t \log(\mathbf{y}_k^t; \mathbf{w})$$

となる.

同系列長同士の問題

RNNに系列データを入力すると同じ長さの系列が出力データとして得られるため, この構造を持っている問題であれば, そのままRNNを適用することができる.

- N個の単語からなる文を入力に受け取り, 各単語の品詞を決定する問題.
- 連続稼働する機械設備を常時監視し, 異常検知したり, 事前に発生を予測したりする問題.

出力が単一要素である問題

出力が単一要素である問題であれば, 同様にRNNを適用できる.

- 感情分析問題
  この場合, 例えば, RNNの最後の時刻の出力のみを利用する.

RNNでのリアルタイム推論

要素が逐次, 追加入力されて系列をなすような問題の場合, RNNで実時間推論を行うことができる.

- 例えば, 工業機械の異常検知等

一方, そのような実時間処理が必要とされない場合, 系列データを逆方向に入力することも可能である.
このように逆向きの系列を入力に取るRNNと, 通常の向きのRNNを考え, 両者のRNNの出力層を統合したものが双方向RNN(Bidirectional RNN)である.

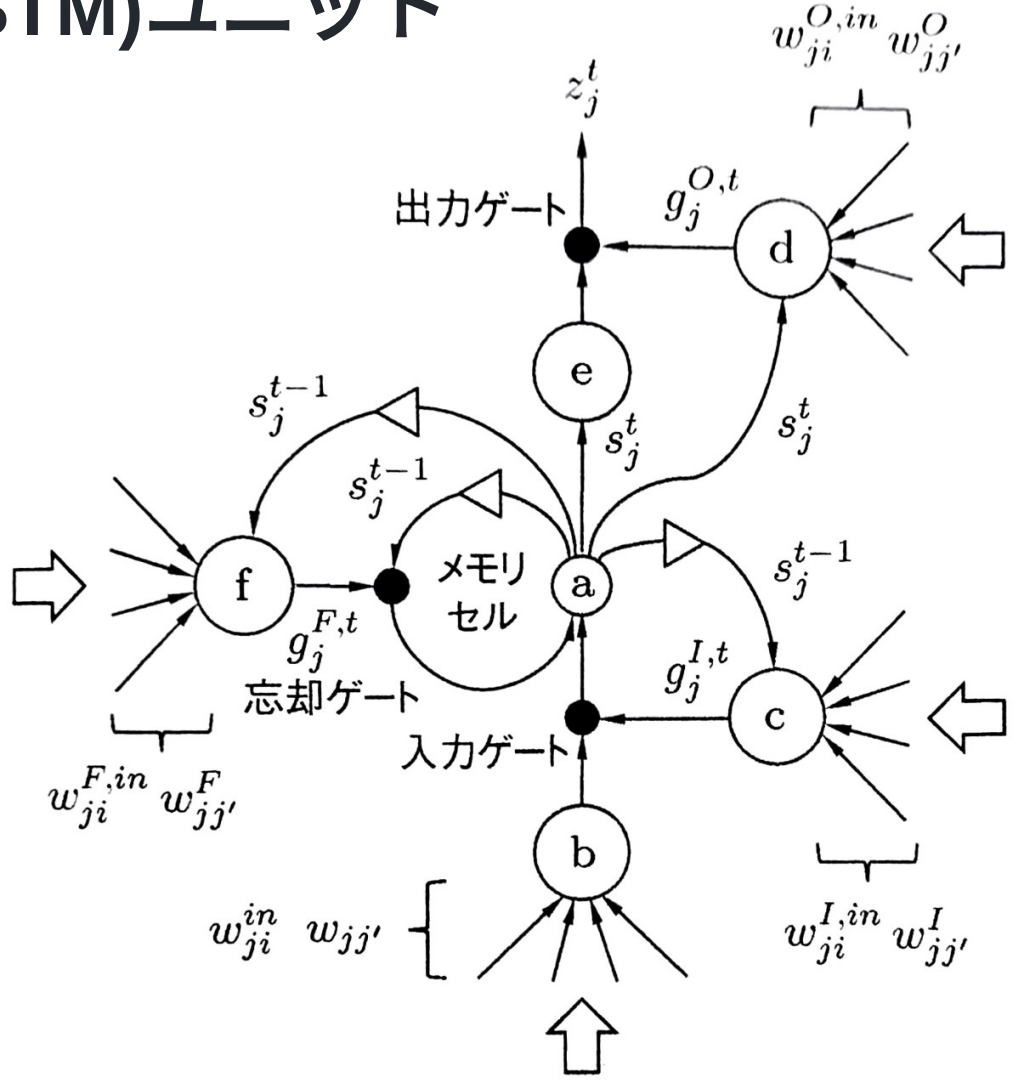
- 言語処理モデル等で前後の文脈を考慮する際に利用されることがある.

ゲート機構

RNNと勾配消失問題

RNNは過去の入力の履歴を保持できるが, 出力に反映できるのはせいぜい過去10時刻分の入力である.
これは, 図6.4のように考えるとわかりやすく, RNNの見かけの総数は, 高々3層であるが, 逆誤差伝播を行う際は, さかのぼる時間の分だけ深いネットワークになる.
$\rightarrow$ 勾配消失が発生する.

長・短期記憶(LSTM)ユニット



前述の勾配消失問題の解決を狙い, 長期にわたる記憶を可能にすべく考案されたのが,
長・短期記憶(Long Short-Term Memory, LSTM).
入力(I), 出力(O), 忘却(F)の3種のゲートと, メモリセルと呼ばれる状態記憶素子を持つ.
各ゲートのコントロールには, 現在の入力と中間層の出力を帰還させたものを利用し,
各ゲートごとに,異なる重みをかけたのちに, ロジスティックシグモイド関数で出力された値$g_j^{*,t} \in [0,1]$を利用する. 0に近いほど, 信号がブロックされ, 1に近いほど信号が素通りする.
(なおわかると思うけど, $* \in \{I, O, F\}$)

定式化

まずはメモリセルの状態遷移を考える. 時刻tにおけるj番目のメモリユニットの記憶状態を, s_j^tとする. このとき, 現在の状態は, 以前の状態に忘却ゲート出力を乗じたものと, 入力に入力ゲート出力を乗じたものとなるため,

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

となる.

なお, $u_j^t, g_j^{F,t}, g_j^{I,t}$は, それぞれメモリユニットへの入力と, 忘却, 入力各ゲートの出力のため, 以下のように表される.

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

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

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

σはロジスティックシグモイド関数. 定義そのままやね

w_j^F, w_j^Iはメモリセルからの出力をゲート開閉の決定に利用するための結合係数.
「のぞき穴(peephole)」結合と呼ばれることもある.
性能向上への貢献が大きくないこともあり, 省略されることもある.([135]論文にはある種のゲートとのぞき穴結合を省略しても, 顕著な性能低下がないことが示されている)

メモリユニットからの出力は, メモリセル出力に活性化関数($\tanh$)を適用したものに出力ゲートを乗じたものなので, 形はほぼ同じで以下のようになる.

$$z_j^t = g_j^{O,t} f(s_j^t)$$

ゲートも形は同じで添え字が違うだけ(ほぼ). 覗き穴結合が$t-1$でなくtの時を参照しているくらい

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

まとめると

$$\mathbf{u}^t = \mathbf{W}^{in}\mathbf{x}^t + \mathbf{W}\mathbf{z}^{t-1}$$

$$\mathbf{s}^t = \mathbf{g}^{F,t} \odot \mathbf{s}^{t-1} + \mathbf{g}^{I,t} \odot f(\mathbf{u}^t)$$

$$\mathbf{z}^t = \mathbf{g}^{O,t} \odot f(\mathbf{s}^t)$$

また, 各ゲートの出力は

$$\mathbf{g}^{F,t} = \sigma(\mathbf{W}^{F,in}\mathbf{x}^t + \mathbf{W}^F\mathbf{z}^{t-1})$$

$$\mathbf{g}^{I,t} = \sigma(\mathbf{W}^{I,in}\mathbf{x}^t + \mathbf{W}^I\mathbf{z}^{t-1})$$

$$\mathbf{g}^{O,t} = \sigma(\mathbf{W}^{O,in}\mathbf{x}^t + \mathbf{W}^O\mathbf{z}^{t-1})$$

とあらわされる.

その他のゲート付き機構

忘却ゲートが最も重要なゲートと考えられているので, それ以外のゲートを削減した構造が提案されている.

更新ゲートRNN(Update Gate RNN, UGRNN)

忘却ゲートと入力ゲートを合わせたようなゲート, 更新ゲート(update gate)を採用し, 出力ゲートを除いたもの. LSTMでは忘却ゲートと入力ゲートは独立に制御されていたが, 更新ゲートでは排他的に制御するようになっている.

$$\mathbf{u}^t = \mathbf{W}^{in}\mathbf{x}^t + \mathbf{W}\mathbf{z}^{t-1}$$

$$\mathbf{s}^t = \mathbf{g}^{U,t} \odot \mathbf{s}^{t-1} + \left(1 - \mathbf{g}^{U,t}\right) \odot f\left(\mathbf{u}^t\right)$$

$$\mathbf{z}^t = \mathbf{s}^t$$

$$\mathbf{g}^{U,t} = \sigma(\mathbf{W}^{U,in}\mathbf{x}^t + \mathbf{W}^U\mathbf{z}^{t-1})$$

ゲート付きRNN(Gated Recurrent Unit, GRU)

UGRNNを拡張したような形を持っている.
帰還路に初期化ゲートを追加しており, 過去の状態をどの程度伝播させるのか制御できるようになっている.

$$\mathbf{u}^t = \mathbf{W}^{in}\mathbf{x}^t + \mathbf{W}\mathbf{g}^{R,t} \odot \mathbf{z}^{t-1}$$

$$\mathbf{s}^t = \mathbf{g}^{U,t} \odot \mathbf{s}^{t-1} + (1 - \mathbf{g}^{U,t}) \odot f(\mathbf{u}^t)$$

$$\mathbf{z}^t = \mathbf{s}^t$$

$$\mathbf{g}^{U,t} = \sigma(\mathbf{W}^{U,in}\mathbf{x}^t + \mathbf{W}^U\mathbf{z}^{t-1})$$

$$\mathbf{g}^{R,t} = \sigma(\mathbf{W}^{R,in}\mathbf{x}^t + \mathbf{W}^R\mathbf{z}^{t-1})$$

あと, 交差RNN(Intersection RNN, +RNN)とかいろいろあるけど, 論文[138]読んでない.

自己回帰モデル

線形自己回帰モデル

ある時系列データ$\mathbf{x}$の時刻tにおける値$\mathbf{x}^t$を, 今までのデータ$\mathbf{x}^0, \cdots, \mathbf{x}^{t-1}$の線形和とバイアス, ランダムノイズの和で表すことでモデル化したものを(線形)自己回帰モデルと呼び, 以下のように表される.

$$\mathbf{x}^t = \phi_0 + \sum_{i=t-\rho}^{t-1} \phi \mathbf{x}^i + \epsilon^t$$

これは例で, 線形自己回帰モデルの表現力はたかが知れているが, このようなモデルをRNNを使って作ることを考える.
具体的には, $\mathbf{y}\left(\mathbf{x}^{t-1}\right) = \mathbf{x}^t$なるモデル$\mathbf{y}$を作る.

RNNによる自己回帰モデル

単語の系列としての文をRNNで学習させたもの $\rightarrow$ 言語モデル

モデルの生成

英単語のうち, 代表的なものをK個選んで, 1-of-K Encodingしてベクトル$\mathbf{x}$を作る.
RNNはこのベクトルを入力にとり, また同じK語を他クラス分類として予測できるように出力層を設計する. 出力$\mathbf{y}$は, その要素y_iがi番目の語である予測確率を表す.
このRNNが学習するタスクは, 1つの文の単語の系列を途中まで順に入力したときに, 次の単語を予測するタスク. 文の始まりと終わりは, `<start>`, `<end>`なる単語として扱う.
このRNNの与える写像$\left(\mathbf{x}^0, \cdots, \mathbf{x}^{t-2}, \mathbf{x}^{t-1}\right) \rightarrow \mathbf{y}^{t-1} \sim \mathbf{x}^t$は, 以下の条件付確率を表現すると考えられる.

$$p\left(\mathbf{x}^t \middle| \mathbf{x}^{t-1}, \mathbf{x}^{t-2}, \cdots, \mathbf{x}^0\right)$$

系列データ生成

生成した言語モデルを利用して, 系列データを生成することができる.
具体的には, 系列を途中まで外部から入力し, そのあとをRNNに自動生成させることができる.
ただし, 出力を思い通りに制御するためには, 外部から追加情報を与える必要がある.

- 時刻$t = 1$において, 入力の代わりに(もしくは追加で), RNNの内部情報$\mathbf{z}^1$を与える.
- $+\alpha$ 各時刻で外部からの追加入力を与え, 自己回帰本来の入力と統合する.

各時刻でK個の単語のうち1つを選ぶ必要があるが, その選び方は複数ある.

- $argmax_k(y_k^t)$を選ぶ
- 出力された確率で単語を選ぶ
- びーむさーちする.

系列変換

系列データを受け取って, 異なる長さの系列データを出力する問題を考える.
このような問題にRNNを適用する方法の一つとして, 系列データ変換(Sequence to Sequence, Seq2Seq)がある.
Seq2Seqは, エンコーダ, デコーダの2種類のRNNを持つ.

エンコーダ

エンコーダ側RNNには入力系列を与え, 与え終わった後の内部状態は, 入力系列をコンパクトに表現したものであると考える.

デコーダ

エンコーダRNNの内部状態を受け取り, 自己回帰RNNとして, 出力系列を生成する.

References

1. https://qiita.com/mochimochidog/items/ca04bf3df7071041561a
2. Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. Neural networks, 2(5), 359-366.
3. https://qiita.com/t_Signull/items/21b82be280b46f467d1b