Sequences Part I: Recurrent Neural Networks (RNNs)

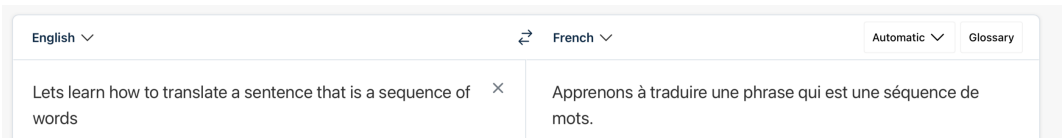
COMP 551 Applied Machine Learning

Isabeau Prémont-Schwarz
School of Computer Science
McGill University

Fall 2024

Deep Neural Networks

- Neural Networks for **Tabular Data**
  - MLP
- Neural Networks for **Images**
  - CNN
- Neural Networks for Sequences
  - The input is a sequence, the output is a sequence, or both are sequences.



Learning Objectives

- Tokenization
- Sequence Learning
  - Seq2Vec
  - Vec2Seq
  - Seq2Seq
- Recurrent Neural Networks (RNNs)
  - Vanilla RNN
  - Gated Recurrent Unit (GRU)
  - Backpropagation through time (BPTT)

Types of Tasks with Sequences

Types of Tasks with Sequences

- Seq2Vec (sequence classification)

Types of Tasks with Sequences

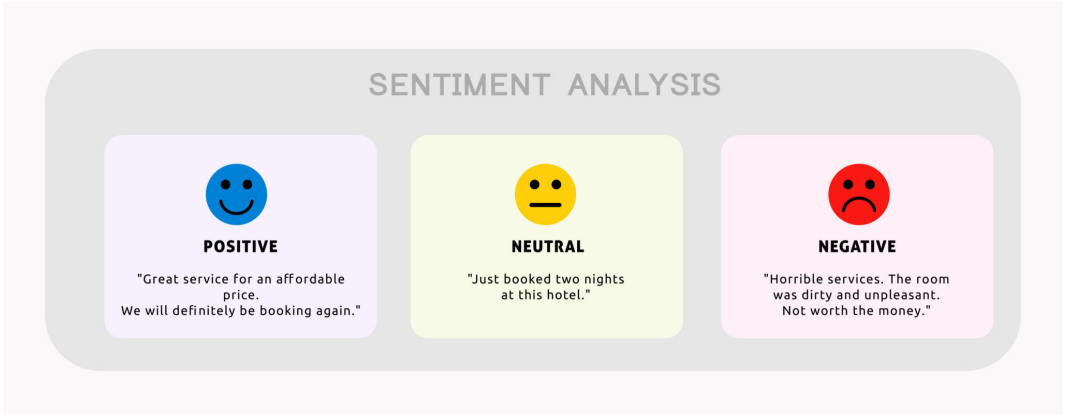
- Seq2Vec (sequence classification)
- Vec2Seq (sequence generation)

Types of Tasks with Sequences

- Seq2Vec (sequence classification)
- Vec2Seq (sequence generation)
- Seq2Seq (sequence translation)

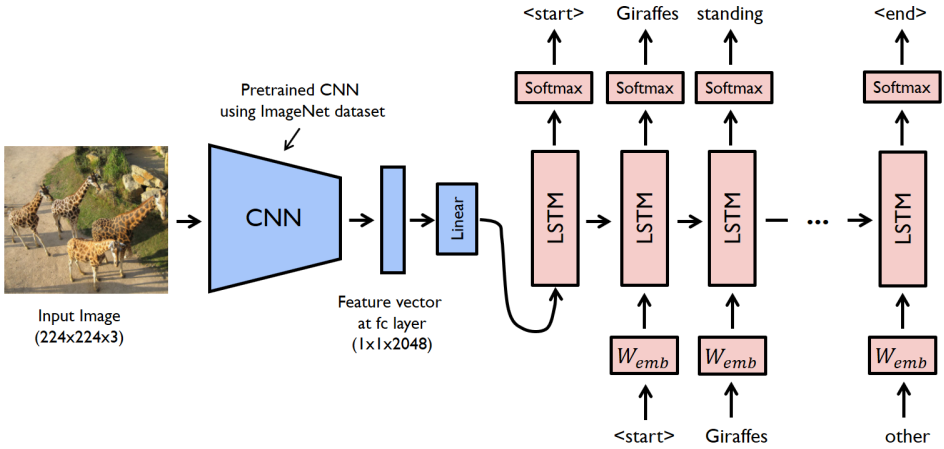
Seq2Vec (sequence classification)

Example:



Vec2Seq (sequence generation)

Example: Caption Generation:



Seq2Seq (sequence translation)

Examples:

Seq2Seq (sequence translation)

Examples:

- Language translation (eg. Fr $\rightarrow$ En)

Seq2Seq (sequence translation)

Examples:

- Language translation (eg. Fr $\rightarrow$ En)
- Protein folding (Sequence of proteins $\rightarrow$ Sequence of angles)

Seq2Seq (sequence translation)

Examples:

- Language translation (eg. Fr $\rightarrow$ En)
- Protein folding (Sequence of proteins $\rightarrow$ Sequence of angles)
- Time series (eg. Stock price prediction)

Seq2Seq (sequence translation)

Examples:

- Language translation (eg. Fr $\rightarrow$ En)
- Protein folding (Sequence of proteins $\rightarrow$ Sequence of angles)
- Time series (eg. Stock price prediction)

Note: We also distinguish between aligned and non-aligned Seq2Seq models.

1D Convolution

How could we generate a sequence of unknown length?

How could we generate a sequence of unknown length?

- Have a state which keeps track of past information.

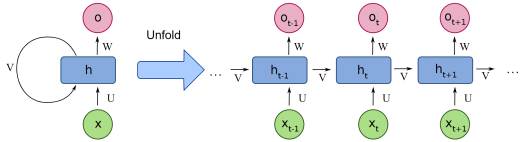
How could we generate a sequence of unknown length?

- Have a state which keeps track of past information.
- Have an special token $< EOS >$ which designates the end of sequence.

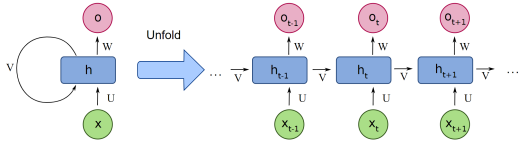
How could we generate a sequence of unknown length?

- Have a state which keeps track of past information.
- Have an special token $< EOS >$ which designates the end of sequence.
- Potentially also a token $< SOS >$ which designates the start of sequence.

Vanilla RNN

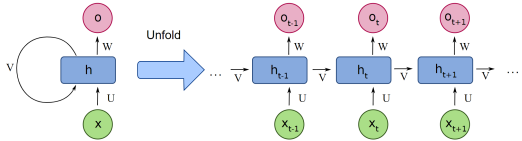


Vanilla RNN



$$x_t \in \mathbb{R}^d, h_t \in \mathbb{R}^m, o_t \in \mathbb{R}^n$$

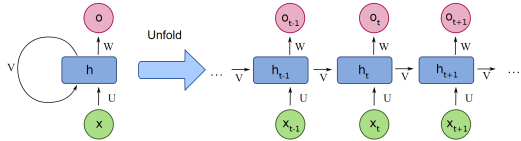
Vanilla RNN



$$x_t \in \mathbb{R}^d, h_t \in \mathbb{R}^m, o_t \in \mathbb{R}^n$$

Matrix shapes: $U : m \times d, V : m \times m, W : n \times m$

Vanilla RNN



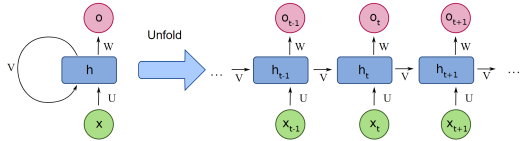
$$x_t \in \mathbb{R}^d, h_t \in \mathbb{R}^m, o_t \in \mathbb{R}^n$$

Matrix shapes: $U : m \times d, V : m \times m, W : n \times m$

$$h_t = f\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$

where f is the non-linear activation function (tanh is often used, but ReLU is also possible)

Vanilla RNN



$$x_t \in \mathbb{R}^d, h_t \in \mathbb{R}^m, o_t \in \mathbb{R}^n$$

Matrix shapes: $U : m \times d, V : m \times m, W : n \times m$

$$h_t = f\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$

where f is the non-linear activation function (tanh is often used, but ReLU is also possible)

$$o_t = g\left(W \cdot h + b_o\right)$$

Vanilla RNN: example

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

Vanilla RNN: example

$$h_t = \mathrm{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, b_o = 0$$

Vanilla RNN: example

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, b_o = 0$$

$$h_{t-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, x_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

What is h_t?

Vanilla RNN: example

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, b_o = 0$$

$$h_{t-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, x_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

What is h_t?

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$

Vanilla RNN: example

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, b_o = 0$$

$$h_{t-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, x_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

What is h_t?

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t\right)$$

Vanilla RNN: example

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, b_o = 0$$

$$h_{t-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, x_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

What is h_t?

$$h_t = \text{ReLU}\left(\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + U \cdot x_t\right)$$

Vanilla RNN: example

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, b_o = 0$$

$$h_{t-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, x_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

What is h_t?

$$h_t = \text{ReLU}\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + U \cdot x_t\right)$$

Vanilla RNN: example

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, b_o = 0$$

$$h_{t-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, x_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

What is h_t?

$$h_t = \text{ReLU}\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right)$$

Vanilla RNN: example

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, b_o = 0$$

$$h_{t-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, x_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

What is h_t?

$$h_t = \text{ReLU}\left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}\right)$$

Vanilla RNN: example

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, b_o = 0$$

$$h_{t-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, x_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

What is h_t?

$$h_t = \text{ReLU}\left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

Vanilla RNN: example

$$h_t = \text{ReLU}\left(V \cdot h_{t-1} + U \cdot x_t + b_h\right)$$
$$o_t = \left(W \cdot h_t + b_o\right), \qquad x_t \in \mathbb{R}^2, h_t \in \mathbb{R}^3, o_t \in \mathbb{R}^1$$

$$U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \end{bmatrix}, V = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, W = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}, b_h = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, b_o = 0$$

$$h_{t-1} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, x_t = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

What is o_t?

$$o_t = W \cdot h_t = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = 1$$

Recurrent neural networks (RNNs)

- Vec2Seq (sequence generation)
  - output, $y_{1:T}$ is generated one token at a time
  - at each step we sample y_t from the hidden state h_t and then feed it back to the model to get h_{t+1}

arbitrary-length sequence of vectors

$$f_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^{N_\infty C}$$

D: input vector size

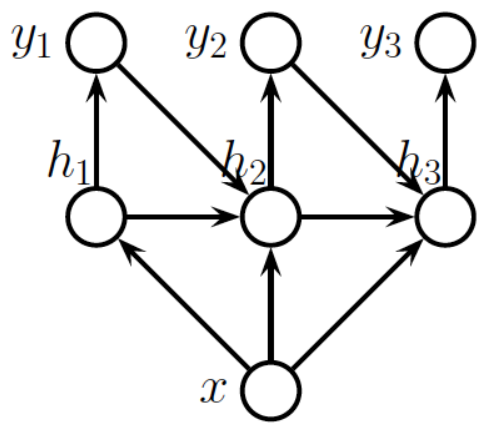
N_∞: *arbitrary-length sequence of vectors of length C*

C: each output vector size

conditional generative model:

$$p(y_{1:T}|x) = \sum_{h_{1:T}} p(y_{1:T}, h_{1:T}|x) = \sum_{h_{1:T}} \prod_{t=1}^{T} p(y_t|h_t)p(h_t|h_{t-1}, y_{t-1}, x)$$

with the initial hidden state$p(h_1|h0, y0, x) = p(h_1|x)$



Recurrent neural networks (RNNs)

- Vec2Seq (sequence generation) $f_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^{TC}$

conditional generative model:

$$p(y_{1:T}|x) = \sum_{h_{1:T}} p(y_{1:T}, h_{1:T}|x) = \sum_{h_{1:T}} \prod_{t=1}^{T} p(y_t|h_t) p(h_t|h_{t-1}, y_{t-1}, x)$$

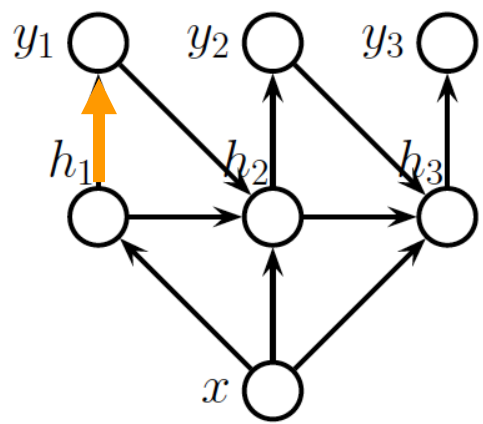
hidden-to-output weights

- real-valued output: $\hat{y}_t = W_{hy} h_t$

$$p(y_t|h_t) = \mathcal{N}(y_t|\hat{y}_t, \mathbf{I})$$

- categorical output: $\hat{y}_t = \mathrm{softmax}(W_{hy} h_t)$

$$p(y_t|h_t) = \mathrm{Categorical}(y_t|\hat{y}_t)$$



** per usual dropping bias terms for simplicity

Recurrent neural networks (RNNs)

- Vec2Seq (sequence generation)

$$f_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^{TC}$$

conditional generative model:

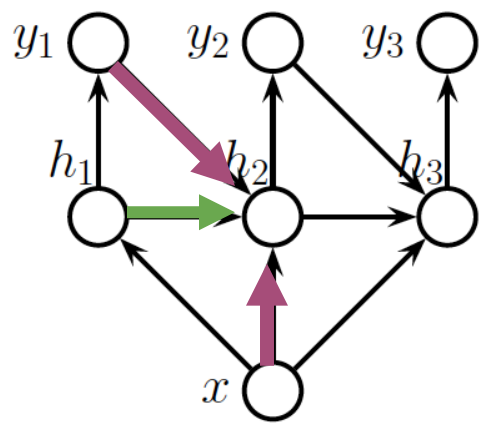
$$p(y_{1:T}|x) = \sum_{h_{1:T}} p(y_{1:T}, h_{1:T}|x) = \sum_{h_{1:T}} \prod_{t=1}^{T} p(y_t|h_t) p(h_t|h_{t-1}, y_{t-1}, x)$$

hidden state:

$$p(h_t|h_{t-1}, y_{t-1}, x) = \mathbb{I}(h_t = f(h_{t-1}, y_{t-1}, x))$$

input-to-hidden weights hidden-to-hidden weights

$$h_t = \varphi(W_{xh}[x; y_{t-1}] + W_{hh} h_{t-1})$$



Recurrent neural networks (RNNs)

- Vec2Seq (sequence generation)

$$f_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^{TC}$$

model

hidden-to-output weights

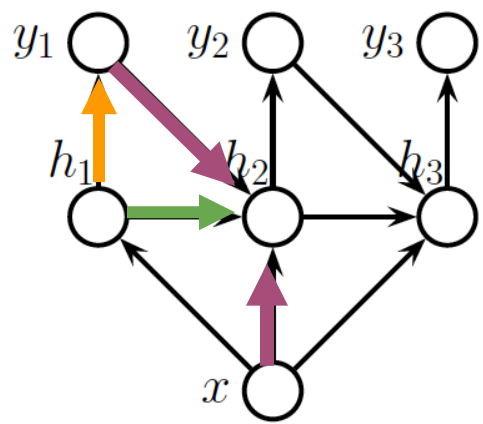
$$\hat{y}_t = g(W_{hy} h_t)$$

input-to-hidden weights hidden-to-hidden weights

$$h_t = \varphi(W_{xh}[x; y_{t-1}] + W_{hh} h_{t-1})$$

RNNs are powerful

- In theory can have unbounded memory and are as powerful as a Turing machine
- In practice, memory size is determined by the size of the latent space and strength of the parameters



Recurrent neural networks (RNNs)

- Vec2Seq (sequence generation)

conditional generative model:

$$p(y_{1:T}|x) = \sum_{h_{1:T}} p(y_{1:T}, h_{1:T}|x) = \sum_{h_{1:T}} \prod_{t=1}^{T} p(y_t|h_t) p(h_t|h_{t-1}, y_{t-1}, x)$$

language modelling: generating sequences unconditionally (by setting $x = \emptyset$) which is learning joint probability distributions over sequences of discrete tokens, i.e., $p(y_1, \ldots, y_T)$

Example:

character level RNN trained on the book The Time Machine by H. G. Wells (32,000 words and 170k character)

Output when given prefix "the":

the *githa some thong the time traveller held in his hand was a glitteringmetallic framework scarcely larger than a small clock and verydelicately made there was ivory in it and the latter than s bettyre tat howhong s ie time thave ler simk you a dimensions le ghat dionthat shall travel indifferently in any direction of space and timeas the driver determinesfilby contented himself with laughterbut i have experimental verification said the time travellerit would be remarkably convenient for the histo*

See the code here, read more here

Recurrent neural networks (RNNs)

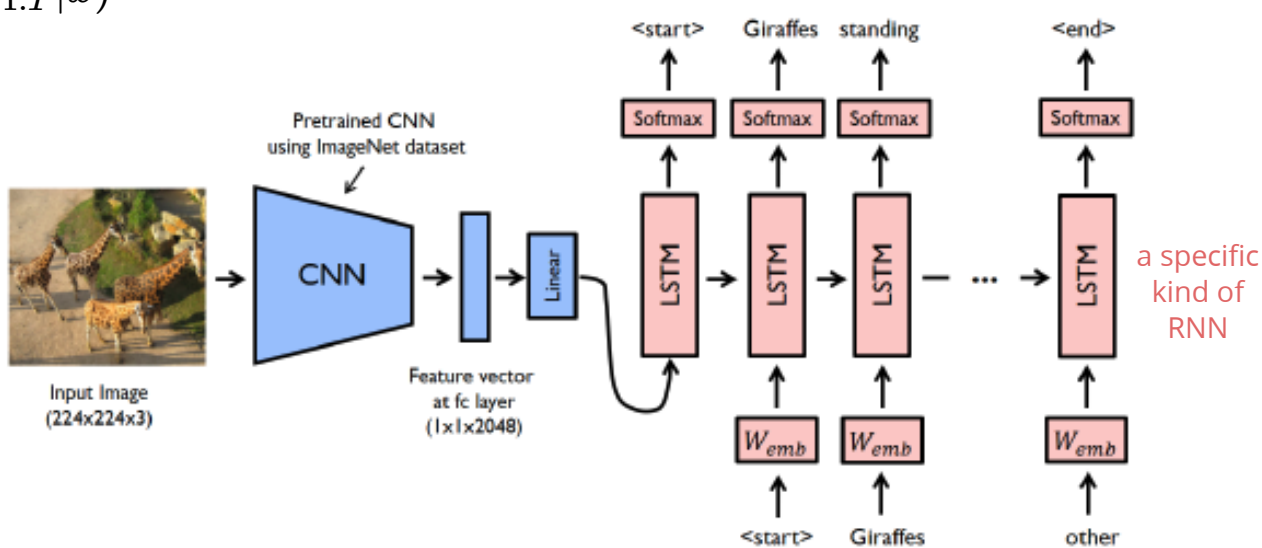
- Vec2Seq (sequence generation)

conditional generative model:

$$p(y_{1:T}|x) = \sum_{h_{1:T}} p(y_{1:T}, h_{1:T}|x)$$

Example:

CNN-RNN model for image captioning when x is embedding by a CNN



See more here

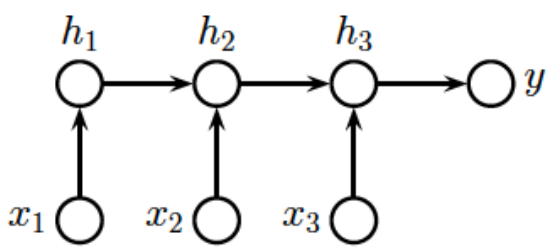
Recurrent neural networks (RNNs)

- Seq2Vec (sequence classification) $f_\theta : \mathbb{R}^{TD} \rightarrow \mathbb{R}^C$
  - predict a single fixed-length output vector given a variable length sequence as input $y \in \{1, \ldots, C\}$

use the final state:

$$\hat{y} = \text{softmax}(W h_T)$$

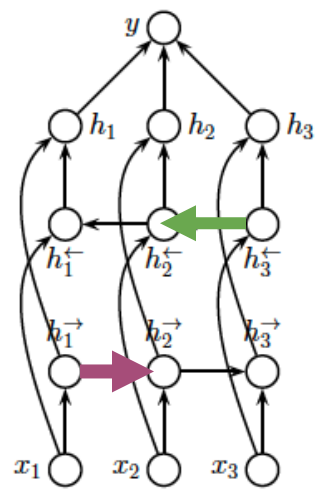
$$p(y|x_{1:T}) = \text{Categorical}(y|\hat{y})$$



Bi-directional RNN:

the hidden states of the RNN depend on the past and future context

gives better results



Recurrent neural networks (RNNs)

- Seq2Vec (sequence classification) $f_\theta : \mathbb{R}^{TD} \rightarrow \mathbb{R}^C$
  - predict a single fixed-length output vector given a variable length sequence as input

$$h_t^{\rightarrow} = \varphi\left(W_{xh}^{\rightarrow} x_t + W_{hh}^{\rightarrow} h_{t-1}^{\rightarrow}\right)$$

$$h_t^{\leftarrow} = \varphi\left(W_{xh}^{\leftarrow} x_t + W_{hh}^{\leftarrow} h_{t+1}^{\leftarrow}\right)$$

$$h_t = [h_t^{\rightarrow}, h_t^{\leftarrow}]$$

$$\overline{h} = \frac{1}{T} \sum_{t=1}^{T} h_t$$

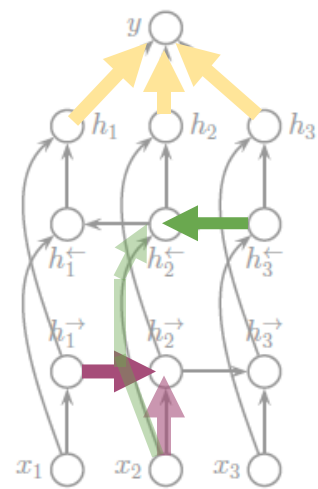
$$\hat{y} = \mathrm{softmax}(W\overline{h})$$

$$p(y|x_{1:T}) = \mathrm{Categorical}(y|\hat{y})$$

Bi-directional RNN:

the hidden states of the RNN depend on the past and future context

gives better results



Recurrent neural networks (RNNs)

- Seq2Vec (sequence classification) $f_\theta : \mathbb{R}^{TD} \rightarrow \mathbb{R}^C$
  - predict a single fixed-length output vector given a variable length sequence as input

Example:

Sentiment classification with word level **bidirectional** LSTM trained on a subset of the Internet Movie Database (IMDB) reviews. (20k positive and 20k negative examples)

IMDb

Prediction examples for two inputs:

'this movie is so great' $\Rightarrow$ 'positive'

'this movie is so bad' $\Rightarrow$ 'negative'

see the code here, and read more here

Recurrent neural networks (RNNs)

- Seq2Seq (sequence translation)
  - aligned: $T = T'$
  - unaligned: $T \neq T'$

$$f_\theta : \mathbb{R}^{TD} \rightarrow \mathbb{R}^{T'C}$$

Recurrent neural networks (RNNs)

- Seq2Seq (sequence translation)
  - aligned: $T = T'$

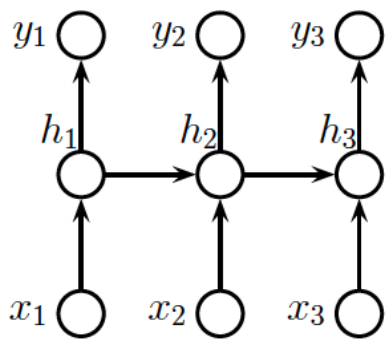
$$f_\theta : \mathbb{R}^{TD} \rightarrow \mathbb{R}^{TC}$$

modify the RNN as:

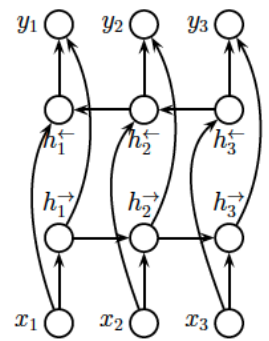
$$p\left(y_{1:T} \mid x_{1:T}\right) = \sum_{h_{1:T}} \prod_{t=1}^{T} p\left(y_t \mid h_t\right) \mathbb{I}\left(h_t = f\left(h_{t-1}, x_t\right)\right)$$

initial state: $h_1 = f\left(h_0, x_1\right) = f_0\left(x_1\right)$

dense sequence labeling:
predict one label per location



Bi-directional



Recurrent neural networks (RNNs)

- Seq2Seq (sequence translation)
  - aligned: $T = T'$

$$f_\theta : \mathbb{R}^{TD} \rightarrow \mathbb{R}^{TC}$$

modify the RNN as:

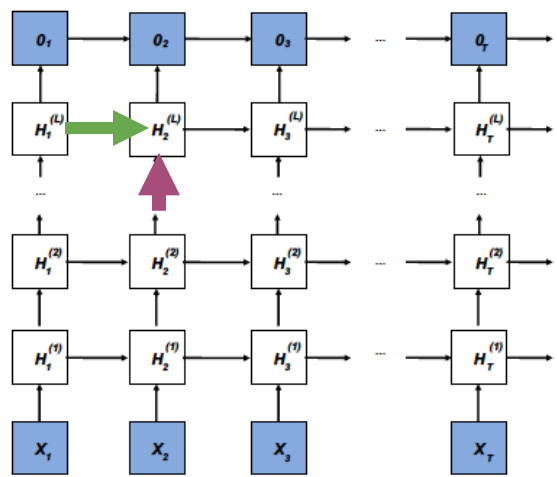
$$p\left(y_{1:T} \mid x_{1:T}\right) = \sum_{h_{1:T}} \prod_{t=1}^{T} p\left(y_t \mid h_t\right) \mathbb{I}\left(h_t = f\left(h_{t-1}, x_t\right)\right)$$

more depth to be more expressive

input-to-hidden weights　　hidden-to-hidden weights

$$h_t^l = \varphi_l\left(W_{xh}^l h_t^{l-1} + W_{hh}^l h_{t-1}^l\right)$$

$$y_t = W_{hy} h_t^L$$



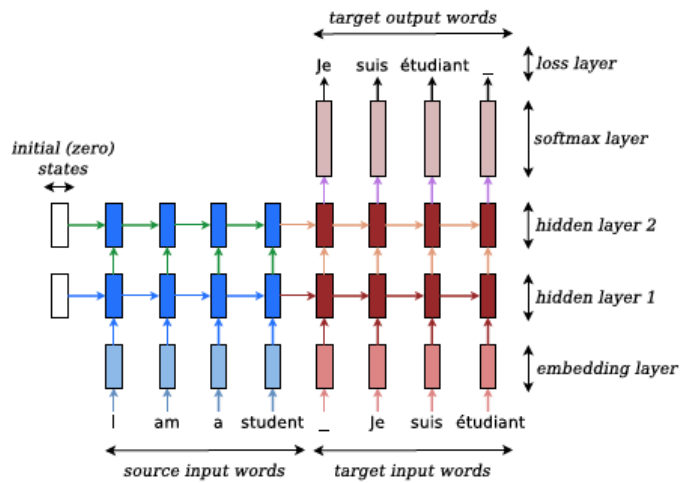
Recurrent neural networks (RNNs)

- Seq2Seq (sequence translation)
  - unaligned: $T \neq T'$

$$f_\theta : \mathbb{R}^{TD} \rightarrow \mathbb{R}^{T'C}$$

- encode the input sequence to get the context vector, the last state of an RNN, $c = f_e(x_{1:T})$
- generate the output sequence using an RNN decoder, $y_{1:T'} = f_d(c)$

Example: translating English to French



see code here

Training: Backpropagation through time (BPTT)

unroll the computation graph, then apply the backpropagation

Example:

model

$$h_t = W_{hx} x_t + W_{hh} h_{t-1}$$

$$\hat{y}_t = W_{hy} h_t$$

loss

$$L = \frac{1}{T} \sum_{t=1}^{T} \ell\left(y_t, \hat{y}_t\right)$$

derivatives

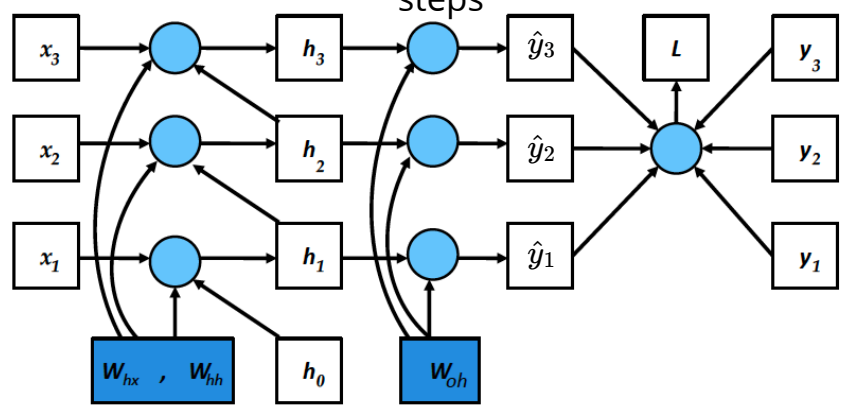
$$\frac{\partial L}{\partial W_{hx}}$$

$$\frac{\partial L}{\partial W_{hh}}$$

$$\frac{\partial L}{\partial W_{hy}}$$

Example:

An RNN unrolled (vertically) for 3 time steps



Training: Backpropagation through time (BPTT)

unroll the computation graph, then apply the backpropagation

Example:

model

$$h_t = W_{hx}x_t + W_{hh}h_{t-1} = f\left(x_t, h_{t-1}, w_h\right)$$

$[\mathrm{vec}(W_{hx}); \mathrm{vec}(W_{hh})]$

$$\hat{y}_t = W_{hy}h_t = g(h_t, w_y)$$

loss

$$L = \frac{1}{T}\sum_{t=1}^{T} \ell\left(y_t, \hat{y}_t\right)$$

derivatives

$$\left.\begin{matrix}\frac{\partial L}{\partial W_{hx}} \\ \frac{\partial L}{\partial W_{hh}} \\ \frac{\partial L}{\partial W_{hy}}\end{matrix}\right\} \frac{\partial L}{\partial w_h} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial \ell(y_t,\hat{y}_t)}{\partial w_h} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial \ell(y_t,\hat{y}_t)}{\partial \hat{y}_t} \frac{\partial g(h_t,w_y)}{\partial h_t} \frac{\partial h_t}{\partial w_h}$$

$$\frac{\partial f(x_t,h_{t-1},w_h)}{\partial w_h} + \frac{\partial f(x_t,h_{t-1},w_h)}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial w_h}$$

expand this recursively

$$\frac{\partial h_t}{\partial w_h} = \frac{\partial f(x_t,h_{t-1},w_h)}{\partial w_h} + \sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t} \frac{\partial f(x_j,h_{j-1},w_h)}{\partial h_{j-1}}\right) \frac{\partial f(x_i,h_{i-1},w_h)}{\partial w_h}$$

see code here

Gating and long term memory

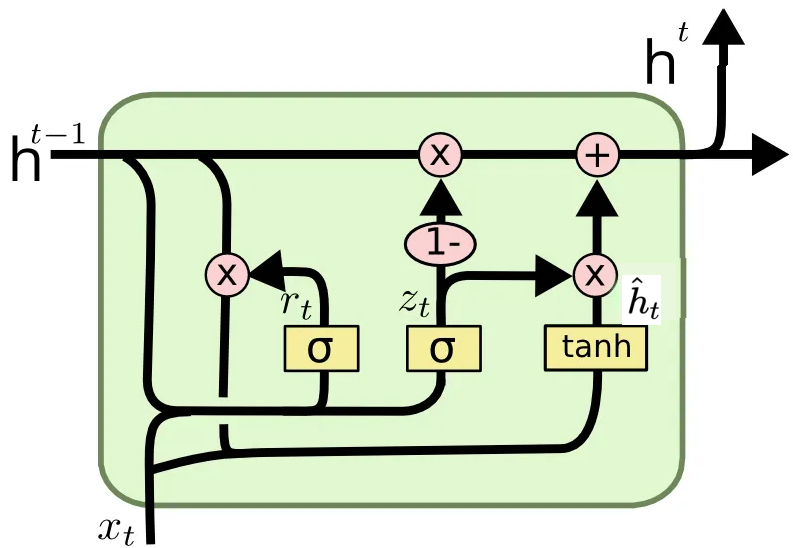
Vanishing and exploding gradients

activations can decay or explode as we go forwards and backwards in time

RNN variations that circumvent this:

- Gated recurrent units (GRU)
  - learns when to update the hidden state, by using a gating unit
- Long short term memory (LSTM)
  - augments the hidden state with a memory cell

The Gated Recurrent Unit (GRU):



$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r)$$
$$\hat{h}_t = \phi(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h)$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t$$

Summary

- Recurrent neural networks (RNNs)
  - Vec2Seq (sequence generation)
  - Seq2Vec (sequence classification)
  - Seq2Seq (sequence translation)
  - training with back propagation through time