Deep Learning

13. Recurrent Neural Networks

Dr. Konda Reddy Mopuri
Dept. of AI, IIT Hyderabad
Jan-May 2023

So far...

1. Perceptron, MLP, Gradient Descent (Backpropagation)

So far...

1. Perceptron, MLP, Gradient Descent (Backpropagation)
2. CNNs (visualizing and understanding)

So far...

1. Perceptron, MLP, Gradient Descent (Backpropagation)
2. CNNs (visualizing and understanding)
3. 'Feedforward Neural networks'

Feedforward NNs: some observations

1. Size of the i/p is fixed(?!)

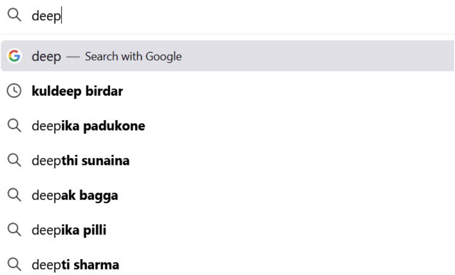
Feedforward NNs: some observations

1. Size of the i/p is fixed(?!)
2. Successive i/p are i.i.d.

Feedforward NNs: some observations

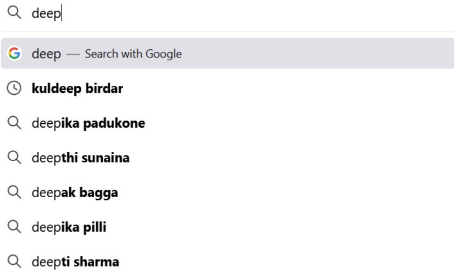
1. Size of the i/p is fixed(?!)
2. Successive i/p are i.i.d.
3. Processing of successive i/p is independent of each other

Consider ‘auto-completion’ task



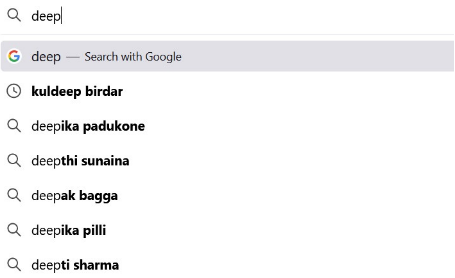
1. Successive i/p are not independent

Consider 'auto-completion' task



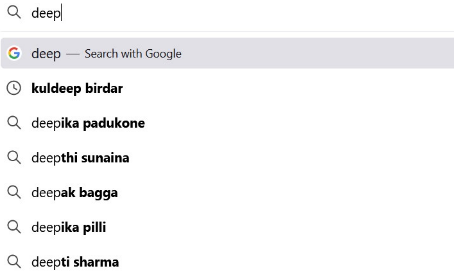
1. Successive i/p are not independent
2. Length of the i/p is not fixed ($\rightarrow$ predictions also)

Consider 'auto-completion' task



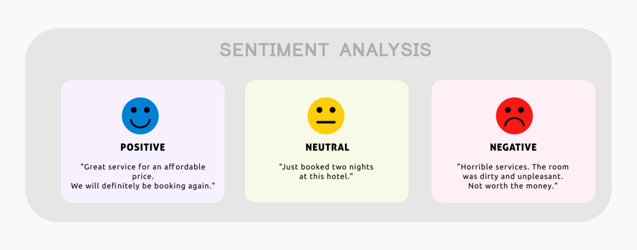
1. Successive i/p are not independent
2. Length of the i/p is not fixed ($\rightarrow$ predictions also)
3. Same underlying task at different 'time instances'

Consider 'auto-completion' task



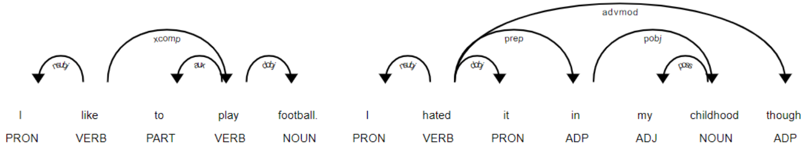
1. Successive i/p are not independent
2. Length of the i/p is not fixed ($\rightarrow$ predictions also)
3. Same underlying task at different 'time instances'
4. Sequence Learning Problems

Sequence Learning Tasks: Example



Sentiment Analysis (Source)

Sequence Learning Tasks: Example



POS-Tagging (Source:Kaggle)

Sequence Learning Tasks: Example

Action Recognition (Source)

Sequence Learning Tasks: Example

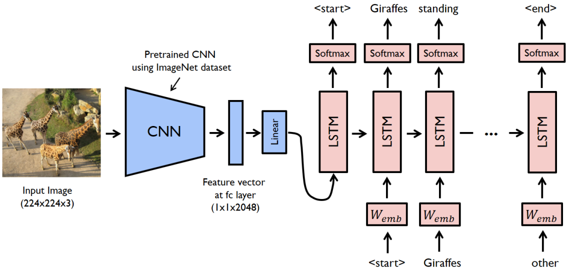
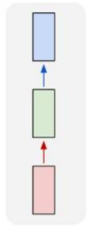


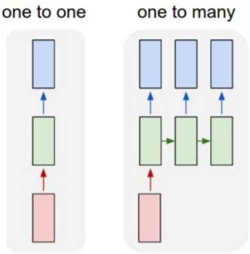
Image Captioning(Source)

Sequence Learning Tasks: Variations



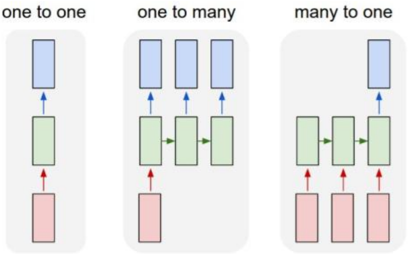
Source

Sequence Learning Tasks: Variations



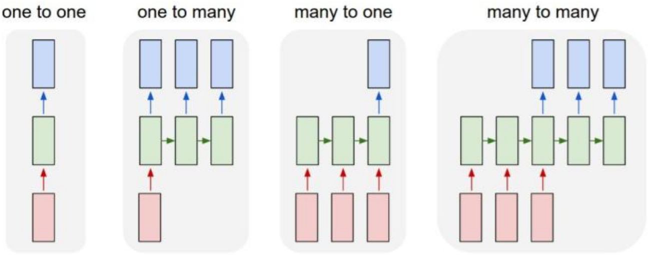
Source

Sequence Learning Tasks: Variations



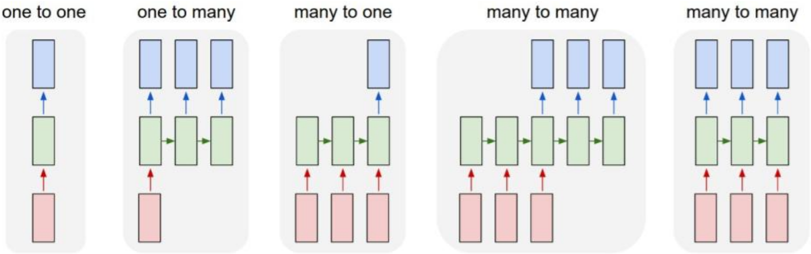
Source

Sequence Learning Tasks: Variations



Source

Sequence Learning Tasks: Variations



Source

Recurrent Neural Networks (RNN)

1. NNs designed to solve sequence learning tasks

Recurrent Neural Networks (RNN)

1. NNs designed to solve sequence learning tasks
2. Characteristics
    1. Model the dependence among the i/p

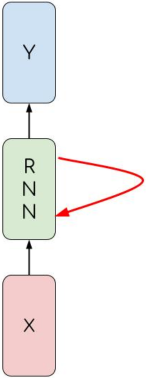
Recurrent Neural Networks (RNN)

1. NNs designed to solve sequence learning tasks
2. Characteristics
    1. Model the dependence among the i/p
    2. Handle variable length of i/p

Recurrent Neural Networks (RNN)

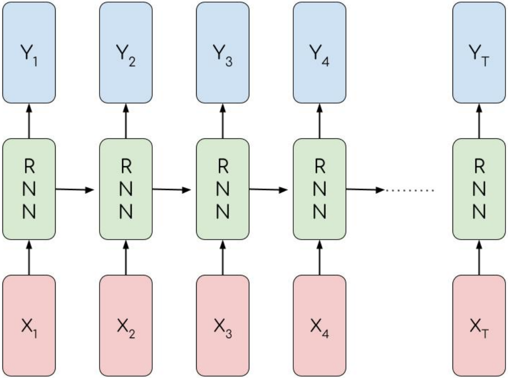
1. NNs designed to solve sequence learning tasks
2. Characteristics
    1. Model the dependence among the i/p
    2. Handle variable length of i/p
    3. Same function applied at all time instances

RNNs: internal state



RNNs have internal state (or, memory) that gets updated with input

RNNs: unfolding



RNNs

1. Apply the same transformation at every time step $\rightarrow$ 'Recurrent' NNs

RNNs

1. Apply the same transformation at every time step $\rightarrow$ 'Recurrent' NNs
2. i/p sequence $x_t \in \mathbb{R}^{\mathbb{D}}$

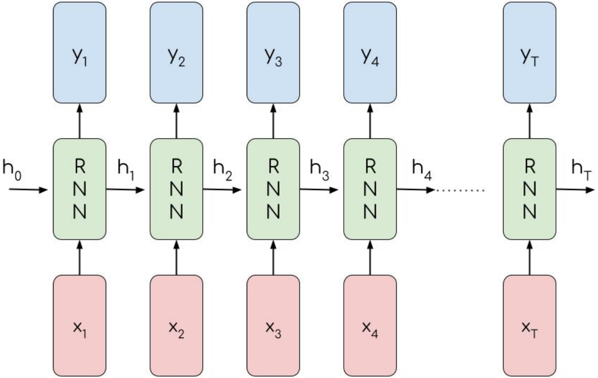
RNNs

1. Apply the same transformation at every time step $\rightarrow$ 'Recurrent' NNs
2. i/p sequence $x_t \in \mathbb{R}^{\mathbb{D}}$
3. Initial recurrent state $h_0 \in \mathbb{R}^{\mathbb{Q}}$

RNNs

1. Apply the same transformation at every time step $\rightarrow$ 'Recurrent' NNs
2. i/p sequence $x_t \in \mathbb{R}^{\mathbb{D}}$
3. Initial recurrent state $h_0 \in \mathbb{R}^{\mathbb{Q}}$
4. RNN model computes sequence of recurrent states iteratively
   $h_t = \phi(x_t, h_{t-1}; w)$

RNNs



Elmon RNN (1990)

1. Start with $h_0 = 0$

Elmon RNN (1990)

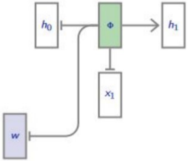
1. Start with $h_0 = 0$
2. $h_t = tanh(W_{xh}.x_t + W_{hh}.h_{t-1} + b_h)$

Elmon RNN (1990)

1. Start with $h_0 = 0$
2. $h_t = tanh(W_{xh}.x_t + W_{hh}.h_{t-1} + b_h)$
3. $y_t = softmax(W_{hy}.y_t + b_y)$

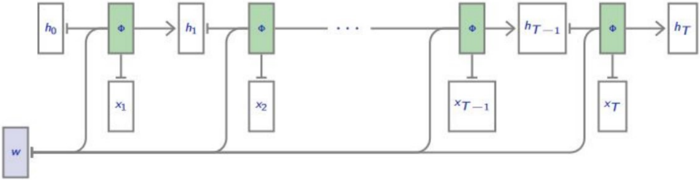
RNNs as computational graph

1. Use the same set of parameters at each time step



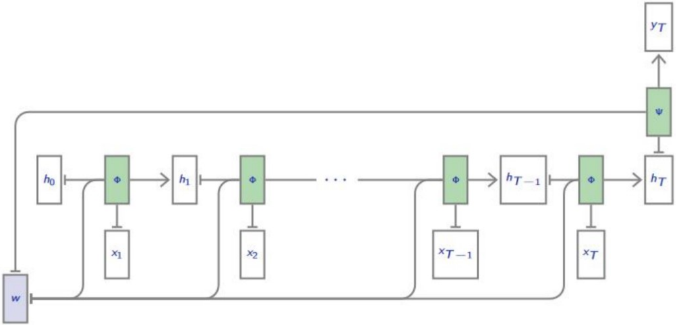
RNNs as computational graph

1. Use the same set of parameters at each time step



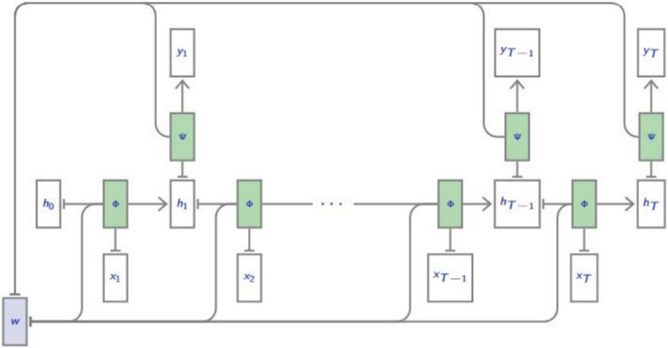
RNNs as computational graph

1. Use the same set of parameters at each time step



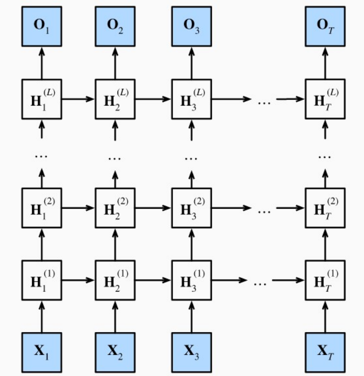
RNNs as computational graph

1. Use the same set of parameters at each time step
2. Flexible to realize different variants (with some tricks!)



Multi-layered RNNs

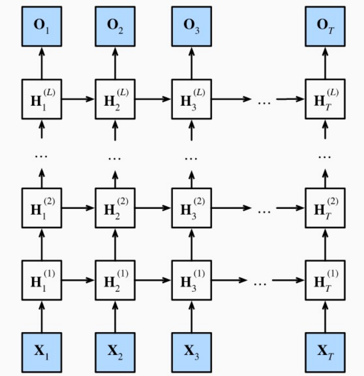
1. Stack multiple RNNs between i/p and o/p layers



Source

Multi-layered RNNs

1. Stack multiple RNNs between i/p and o/p layers
2. $H_t^{(l)} = W_{xh}^{(l)} \cdot H_t^{(l-1)} + W_{hh}^{(l)} \cdot H_{t-1}^{(l)} + b_h^{(l)}$



Source

Backpropagation Through Time (BPTT)

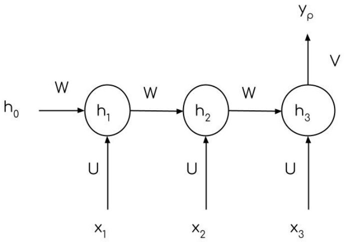
1. Consider a many-to-one variant RNN (e.g. sentiment analysis)

Backpropagation Through Time (BPTT)

1. Consider a many-to-one variant RNN (e.g. sentiment analysis)
2. Let's separate the parameters into U, V, and W

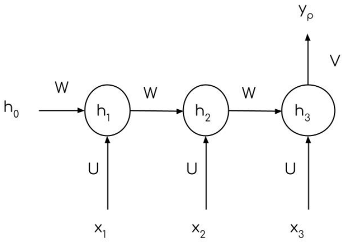
Backpropagation Through Time (BPTT)

1. Consider a many-to-one variant RNN (e.g. sentiment analysis)



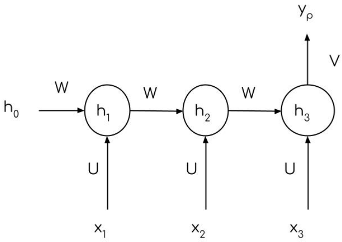
Backpropagation Through Time (BPTT)

1. Consider a many-to-one variant RNN (e.g. sentiment analysis)
2. Let's separate the parameters into U, V, and W



Backpropagation Through Time (BPTT)

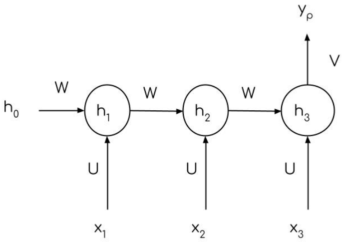
1. Let's now perform SGD (assume loss L is formulated on y_p)



Backpropagation Through Time (BPTT)

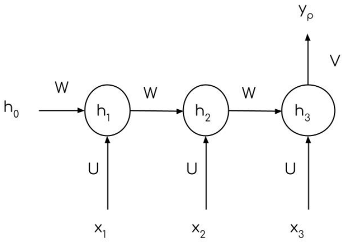
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

1. Let's now perform SGD (assume loss L is formulated on y_p)
2. $\rightarrow$ we need to compute $\frac{\partial L}{\partial V}$, $\frac{\partial L}{\partial W}$, and $\frac{\partial L}{\partial U}$



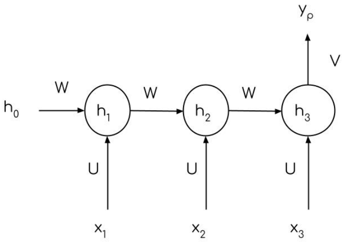
Backpropagation Through Time (BPTT)

1. $\frac{\partial L}{\partial V} = \frac{\partial L}{\partial y_p} \frac{\partial y_p}{\partial V} = \frac{\partial L}{\partial y_p} \cdot \frac{\partial y_p}{\partial z_3} \cdot \frac{\partial z_3}{\partial V}$



Backpropagation Through Time (BPTT)

1. $\frac{\partial L}{\partial V} = \frac{\partial L}{\partial y_p}\frac{\partial y_p}{\partial V} = \frac{\partial L}{\partial y_p} \cdot \frac{\partial y_p}{\partial z_3} \cdot \frac{\partial z_3}{\partial V}$
2. Since we know that $z_3 = V \cdot h_3 + b_y$ and h_3, b_y are independent of V, we can compute $\frac{\partial L}{\partial V}$ in a single step



Backpropagation Through Time (BPTT)

1. Let's now consider $\frac{\partial L}{\partial W}$