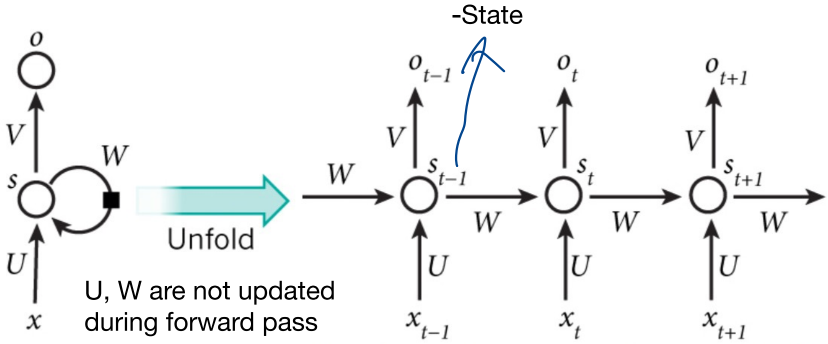
Motivation

- process sequential data
- capture history of inputs/states
- share parameters through a very deep computational graph
  - output is a function of the previous output
  - produced using the same update rule applied to the previous outputs.
- different from convolution across time steps

Recurrent Neural Nets

U, W have same dims

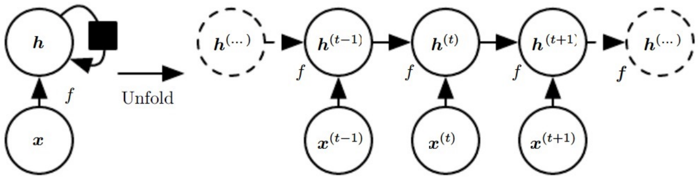
Hidden -State



U, W are not updated during forward pass

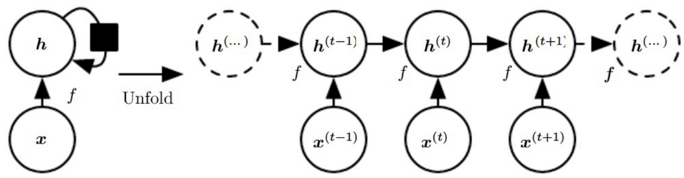
LeCun, Bengio & Hinton. "Deep Learning." *nature* 521.7553 (2015)

Back-Propagation Through Time
(BPTT)



- gradient computation for unfolded loss function w.r.t. parameters very expensive
- O(T) where T is history length
- no parallelization (sequential dependence)

RNN Challenges

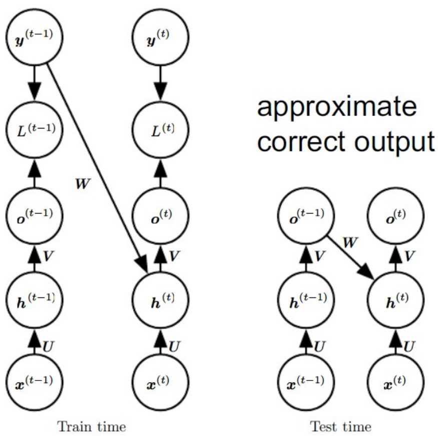


- repeated application of the same operation f
  - exploding gradients — gradient clipping } Remedy
  - vanishing gradients — skip connections }
- long-term dependencies

Teacher Forcing

- use targets as prior outputs
- time steps decoupled
- training parallelizable

approximate correct output



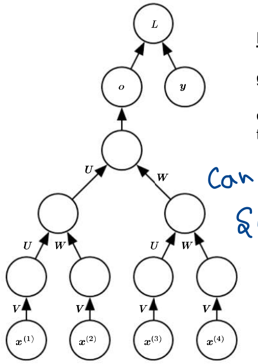
(may also be applied to RNNs with additional hidden-to-hidden connections)

Recursive NNs

- generalization of RNNs
- computational graph structured as deep tree
  - reduces to sequence in RNNs
- can process complex data structures
  - e.g. parse trees

recursive networks

complex structure to fixed-size vector



recursive neural network

generalization of RNNs

computational graph (given from external tool such as parser) structured as deep tree

Can be used in Sentiment Analysis

generalization

RNNs

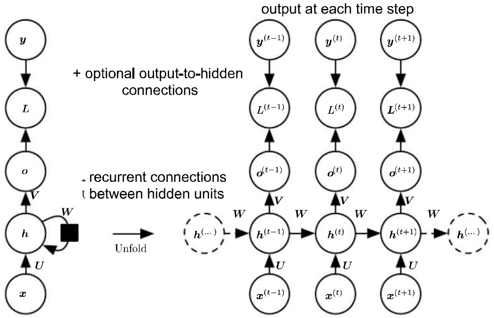
sequence-to-sequence

sequence to sequence (same length)

output at each time step

+ optional output-to-hidden connections

recurrent connections between hidden units

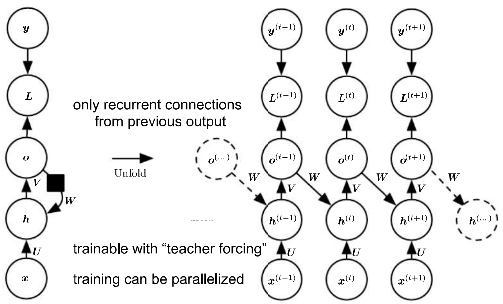


can compute any function computable by a Turing machine (universal function approximator)

sequence to sequence (same length)

output at each time step

only recurrent connections from previous output



trainable with "teacher forcing"

training can be parallelized

lacks important information from past unless o is very high-dimensional & rich

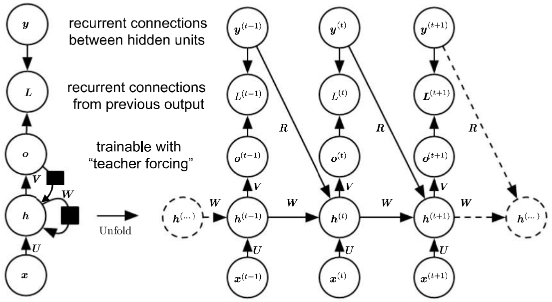
sequence to sequence (same length)

output at each time step

recurrent connections between hidden units

recurrent connections from previous output

trainable with "teacher forcing"



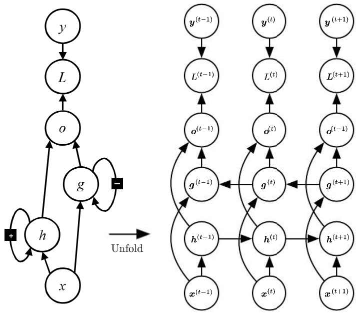
can model arbitrary distribution over sequences of y given sequences of x

bi-directional sequence to sequence (same length)

output at each time step

(extendable to 2D inputs)

recurrent connections between hidden units



g(t) relevant summary of future (backward)

h(t) relevant summary of past (forward),

can model dependencies on both the past and the future

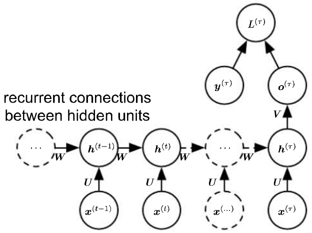
bi-directional

encoder

sequence to fixed-size vector

output after full input sequence has been read

recurrent connections between hidden units



encoder (reader): read input sequence, generate hidden state
(= encoder part of encoder-decoder architecture)

encoder-decoder

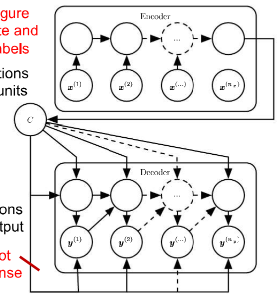
sequence to sequence (variable length)(Async)

Async meaning the model can handle diff inp seq length and can produce diff out seq length.

simplified figure without state and transition labels

recurrent connections between hidden units

encoder-decoder



encoder (reader): read input sequence, generate hidden state

(bottleneck)

recurrent connections from [previous] output

decoder (writer): generate output sequence from hidden state

does not make sense

loss not shown!

decoder

fixed-size ("context") vector to sequence

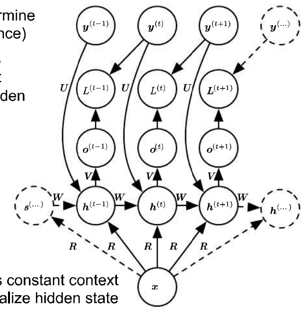
strange indexing (stressing prediction of next output)

(needs to determine end of sequence)

recurrent connections from [previous] output (usually with output-to-hidden connections)

trainable with "teacher forcing"

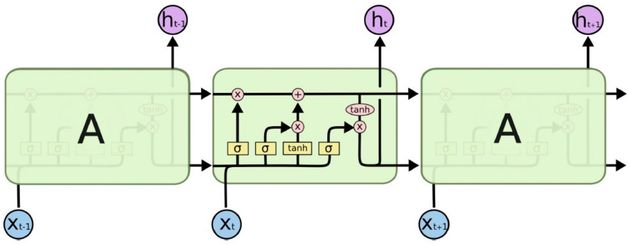
recurrent connections between hidden units



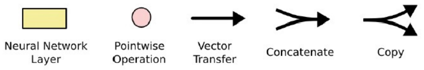
input x serves as constant context or / and to initialize hidden state

decoder (writer): generate output sequence from hidden state
(= decoder part of encoder-decoder architecture)

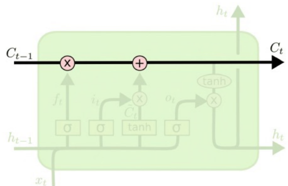
LSTM

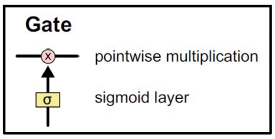


The repeating module in an LSTM contains four interacting layers.

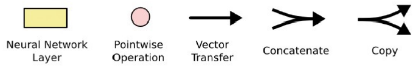


LSTM Cell State

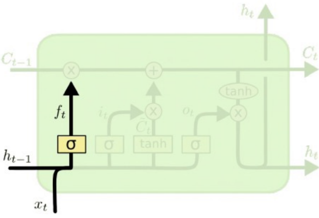




Removing or adding information to the cell state is controlled by gates.

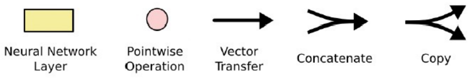


LSTM Forget Gate

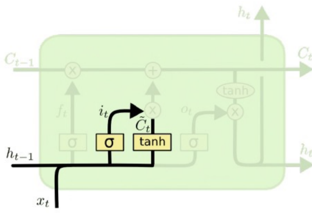


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Decide what information from cell state is deleted (0) or kept (1).

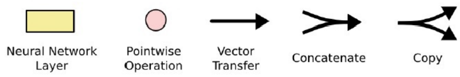


LSTM Input Gate

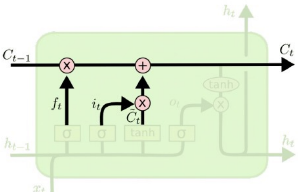


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Decide what new information to store.

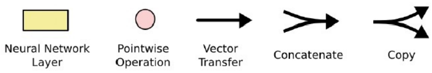


LSTM Cell State Update

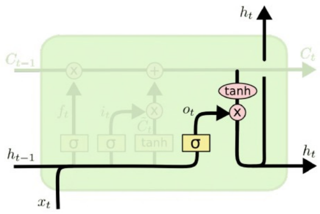


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Delete information and add new one.

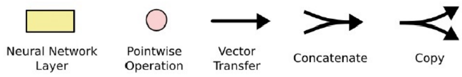


LSTM Output Gate

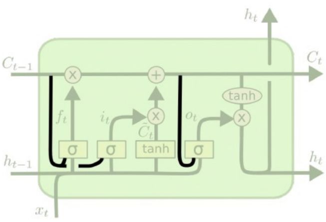


$$o_t = \sigma\left(W_o \ [h_{t-1}, x_t] \ + \ b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Transform state and decide what to output.

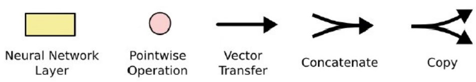


Peep Hole Connections

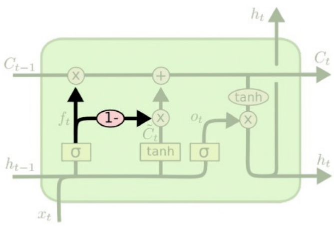


$$f_t = \sigma\left(W_f \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_f\right)$$
$$i_t = \sigma\left(W_i \cdot [\boldsymbol{C_{t-1}}, h_{t-1}, x_t] \ + \ b_i\right)$$
$$o_t = \sigma\left(W_o \cdot [\boldsymbol{C_t}, h_{t-1}, x_t] \ + \ b_o\right)$$

Allow gates to look at cell states.

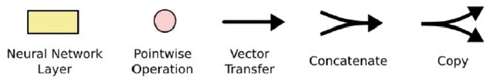


Coupled I/F Gates

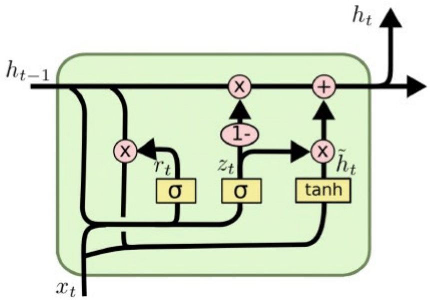


$$C_t = f_t * C_{t-1} + (\mathbf{1} - \boldsymbol{f_t}) * \tilde{C}_t$$

Only input new values to the state when something older gets forgotten.

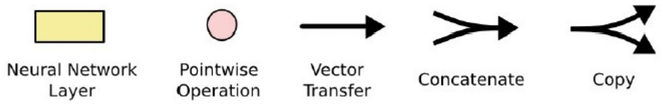


Gated Recurrent Units (GRUs)



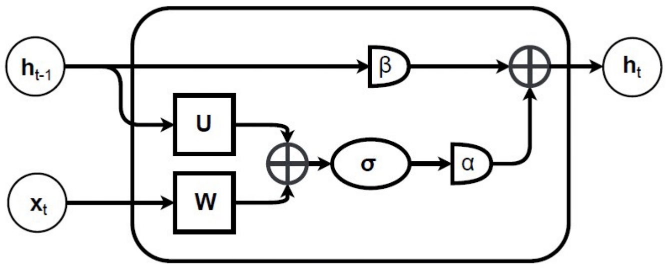
$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Combines forget and input gates into a single update gate.
Merges cell state and hidden state.



FastRNN

adding a weighted residual connection



$$\tilde{\mathbf{h}}_t = \sigma(\mathbf{W}\mathbf{x}_t + \mathbf{U}\mathbf{h}_{t-1} + \mathbf{b})$$
$$\mathbf{h}_t = \alpha\tilde{\mathbf{h}}_t + \beta\mathbf{h}_{t-1}$$

new parameters: $0 \leq \alpha, \beta \leq 1$; typically: $\alpha \ll 1$ and $\beta \approx 1 - \alpha$

RNNs

- work well for sequential data
  - time series (with low sampling rate)
  - texts (translation, discourse, sentiment, ...)
- support variable-length input
  - including long-term dependencies
- are hard to parallelize

Q What makes RNNs stand out from the other network architectures you learned about so far?

○ RNNs are a family of neural networks for processing sequential data. recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization.

○ a recurrent neural network is a neural network that is specialized for processing a sequence of values x(1), . . . , x(τ)

○ A traditional fully connected feedforward network would have separate parameters for each input feature, so it would need to learn all of the rules of the language separately at each position in the sentence. By comparison, a recurrent neural network shares the same weights across several time steps.

Q What is the difference between (a) applying (1D-)convolution along the sequence dimension and (b) using an RNN to process the sequence?

○ The convolution operation allows a network to share parameters across time but is shallow. The output of a convolution is a sequence where each member of the output is a function of a small number of neighbouring members of the input. The idea of parameter sharing manifests in the application of the same convolution kernel at each time step.

○ Recurrent networks share parameters in a different way. Each member of the output is a function of the previous members of the output. Each member of the output is produced using the same update rule applied to the previous outputs. This recurrent formulation results in the sharing of parameters through a very deep computational graph.

Ans: CNN's are stateless, RNNs depending on the previous state.
parallelize in CNN's not in RNNs.

③ what is "back-propagation the time"?

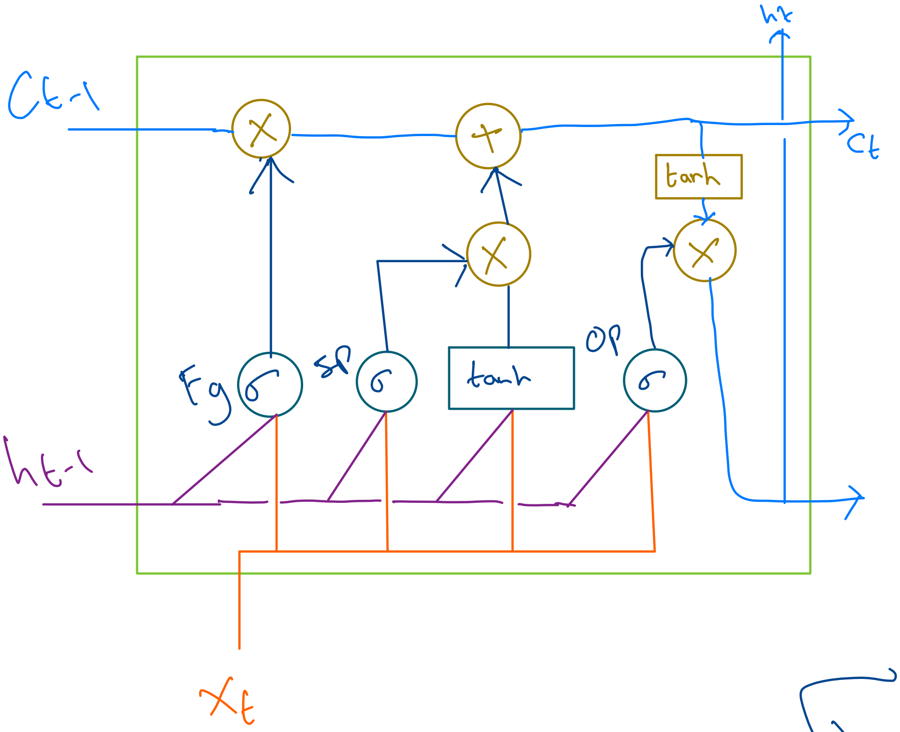
Ans: Compute gradient for an unrolled graph at time t

⑦ what is an LSTM

Ans:- A memory cell will be there in every hidden state.

⑧ In LSTM memory cells the first cell uses sigmoid? any alternatives?

Ans:- uses any function that has range of (0,1) which is differentiable.



Forget gate $= \sigma(W_f [h_{t-1}, x_t]) + b_f$

In gate $= \sigma(W_f [h_{t-1}, x_t])$

o/p gate $= \sigma(W_f [h_{t-1}, x_t])$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$$

$$h_t = o_t * \tanh(C_t)$$