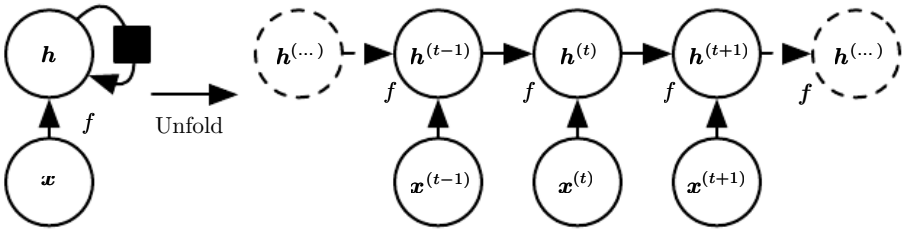
Recurrent Neural Network

CHEN Si

Contents

1 Definition



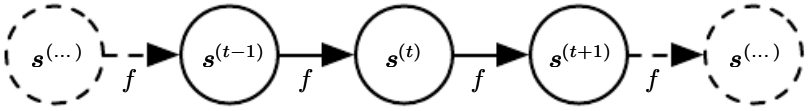
$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

- Hidden units $\boldsymbol{h}$ represents
  1. the state of the dynamical system.
  2. lossy summary of the task-relevant aspects of the past sequence of inputs up to t, when the recurrent network is trained to perform a task that requires predicting the future from the past.
- Black square indicates that an interaction takes place with a delay of a single time step.

1.1 Motivation

Dynamical System

Unfolded Computational Graph



where $\boldsymbol{s}^{(t)}$ is called the **state** of the system.

Recurrent Defintion

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}; \boldsymbol{\theta})$$

Unfolded Definition (up to step 3)

$$\begin{aligned}\boldsymbol{s}^{(3)} &= f(\boldsymbol{s}^{(2)}; \boldsymbol{\theta}) \\ &= f(f(\boldsymbol{s}^{(1)}; \boldsymbol{\theta}); \boldsymbol{\theta})\end{aligned}$$

Dynamical System with External Signal $\boldsymbol{x}^{(t)}$

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

1.2 Advantages

1. Learning just a single model f:
    - The learned model always has the same input size, regardless of the sequence length.
    - Parameter sharing: It is possible to use the same transition function f with the same parameters at every time step.
    - Allows generalization to sequence lengths that does not appear in the training set.
    - Requires much fewer training examples to estimate.

1.3 Disadvantages

1. Optimizing the parameters may be difficult, because of the reduced number of parameters.

1.4 How to determine the length τ

1. Add a special symbol corresponding to the end of a sequence.
2. Introduce an extra Bernoulli output to the model that represents the decision to either contnue generation or halt generation at each time step. (a more general way)
3. Add an extra output to the model that predicts the interger τ itself.

$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}) = P(\tau)P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)} \mid \tau)$$

2 Taxonomy

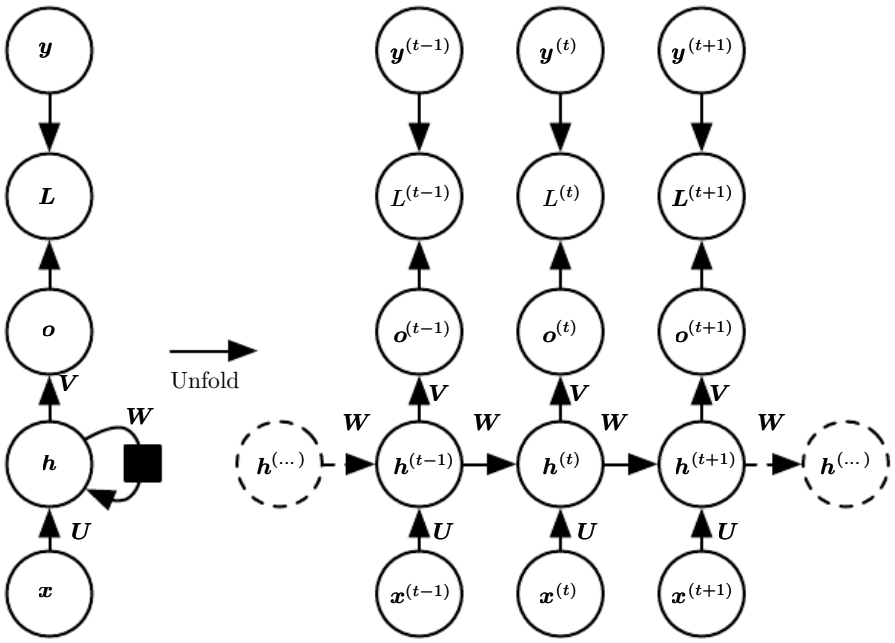
2.1 Sequence to Sequence of the same length

2.1.1

Recurrent networks that produce an output at each time step and have recurrent connections between hidden units.

$$\begin{aligned} \boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \\ \boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\ \boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \\ \hat{\boldsymbol{y}}^{(t)} &= \text{softmax}(\boldsymbol{o}^{(t)}) \end{aligned} \tag{1}$$

with a initial state $\boldsymbol{h}^{(0)}$.



Characteristics

- Maps an input sequence to an output sequence of the same length.

- Loss:

$$\begin{aligned} & L\left(\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\}\right) \\ =& \sum_t L^{(t)} \\ =& -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}\right) \end{aligned} \tag{2}$$

  where $L^{(t)}$ is the negative log-likelihood of $y^{(t)}$ given $\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}$

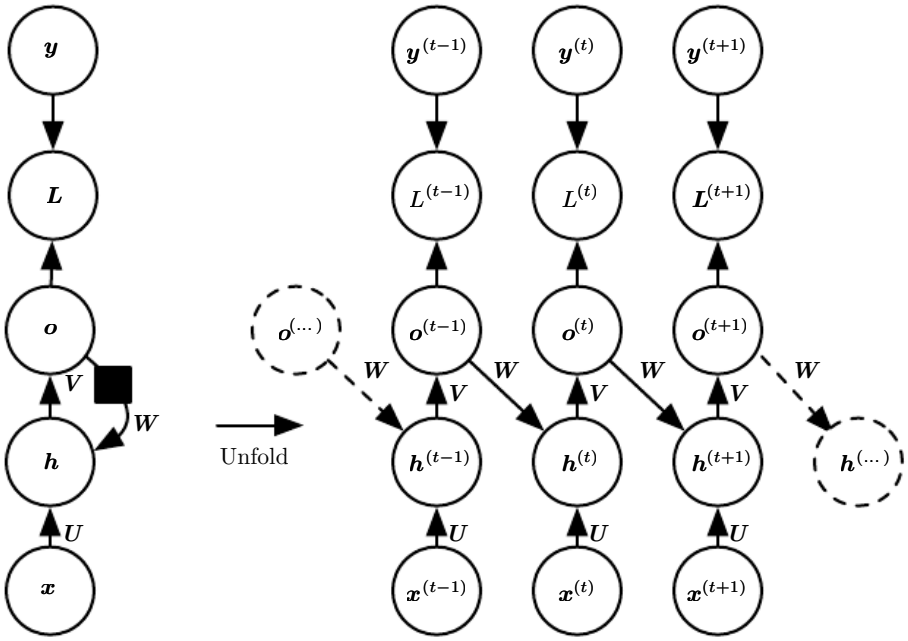
  - Conditional independence assumption: The conditional distribution

$$P(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}) = \prod_t P(\boldsymbol{y}^{(t)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)})$$

- Runtime is $O(\tau)$, memory cost is $O(\tau)$.
- Only able to represent distributins in which the $\boldsymbol{y}$ values are conditionally independent from each other given the $\boldsymbol{x}$ values.

2.1.2

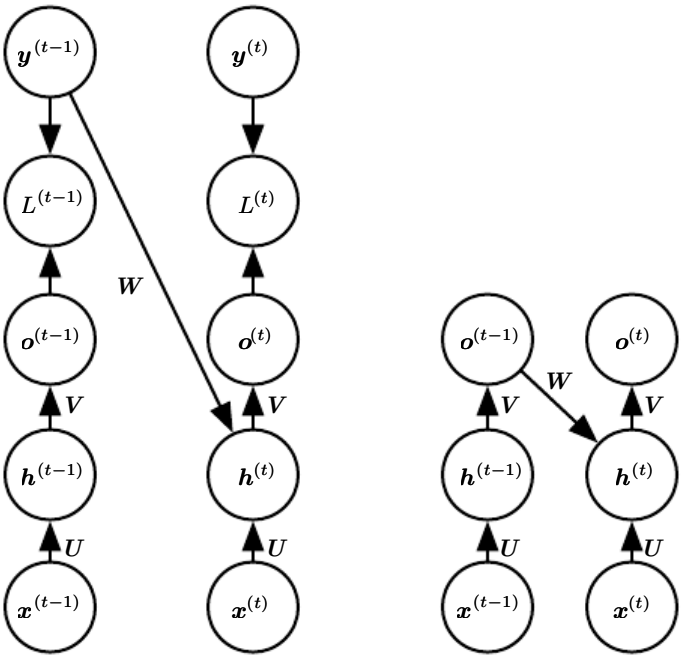
Recurrent networks that produce an output at each time step and have recurrent connections only from the output at one time step to the hidden units at the next time step.



Characteristics

- Strictly less powerful because it lacks hidden-to-hidden recurrent connections. (e.g. it connot simulate a universal Turing machine.)
- Training can be **parallelized** because all the time steps are decoupled (the training set provides the ideal value of previous output).
- Avoids back-propagation through time.
- Teacher forcing: For example, for a sequence with two time steps

$$\begin{aligned}&\log p\left(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right)\\ =&\log p\left(\boldsymbol{y}^{(2)} \mid \boldsymbol{y}^{(1)}, \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right)+\log p\left(\boldsymbol{y}^{(1)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right)\end{aligned}$$

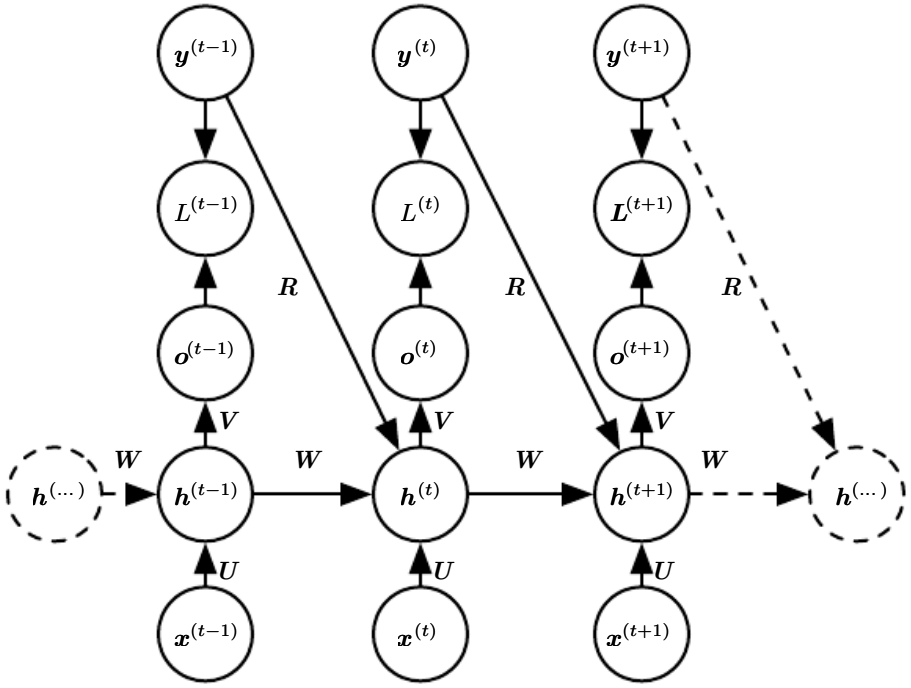


Disadvantages

1. The disadvantages of strict teacher forcing (no BPTT) is that the inputs during training could be quite different from the inputs during testing (**closed-loop** mode).
   Solutions:
   (a) Train with both teacher-forced inputs and free-running inputs

(b) Randomly chooses to use generated values or actual data values as input (curriculum learning strategy: gradially use more of the generated values as input).

2.1.3



Characteristics

- Because of connections from the output at time t to the hidden unit at time $t+1$, the model can represent arbitrary probability distributions over the $\boldsymbol{y}$ sequence (no conditional independence assumption).

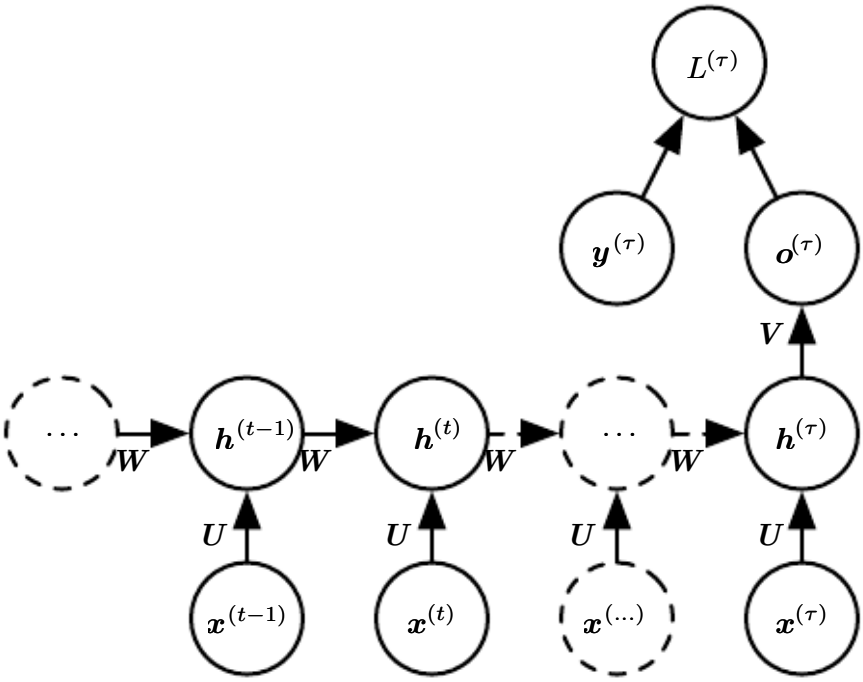
Restrictions

1. The length of both sequences must be the same.

2.2 Sequence to Fixed-Size Vector

2.2.1

Recurrent networks with recurrent connections between hidden units, that read an entire sequence and then produce a single output.



Characteristics

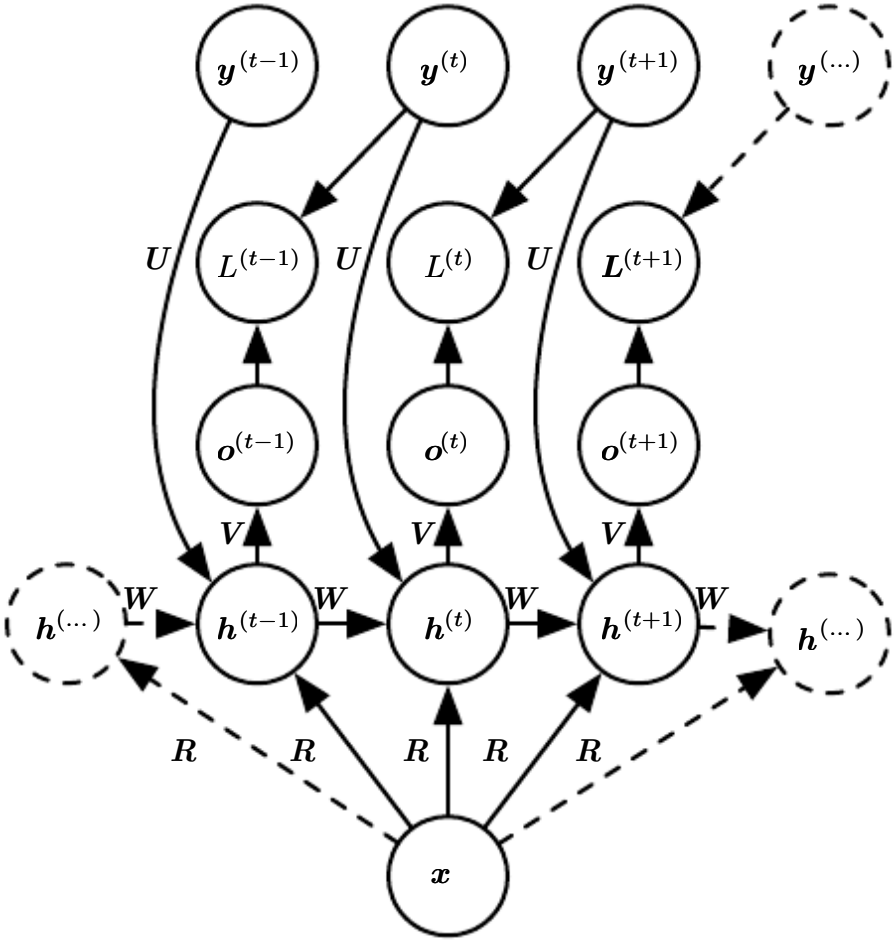
- There might be a target right at the end, or the gradient on the ouput $\boldsymbol{o}^{(t)}$ can be obtained by back-propagating from further downstream modules.

Applications

1. Summarize a sequence and produce a fixed-size representation used as input for further processing.

2.3 Fixed-Size Vector to Sequence

2.3.1



Characteristics

- $\boldsymbol{x}^\top \boldsymbol{R}$ is effectively a new bias parameter.
- Each element $\boldsymbol{y}^{(t)}$ of the observed output sequence serves both as input (for the current time step) and, during training, as target (for the previous time step).

Applications

1. Image captioning, where a single image is used as input to a model that then produces a sequence of words describing the image.

3 Back Propagation Through Time (BPTT)

Here we talk about the BPTT of equation 1 as forward propagation and equation 2 as loss function. (Note: here $\hat{\boldsymbol{y}}^{(t)} = \text{softmax}(\boldsymbol{o}^{(t)})$)

1. $\dfrac{\partial L}{\partial L^{(t)}} = 1$

2. $(\nabla_{\boldsymbol{o}^{(t)}} L)_i = \dfrac{\partial L}{\partial o_i^{(t)}} = \dfrac{\partial L}{\partial L^{(t)}} \dfrac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbb{1}_{i=y^{(t)}}$

3. For $t = \tau$

$$\nabla_{\boldsymbol{h}^{(\tau)}} L = \boldsymbol{V}^\top \nabla_{\boldsymbol{o}^{(\tau)}} L$$

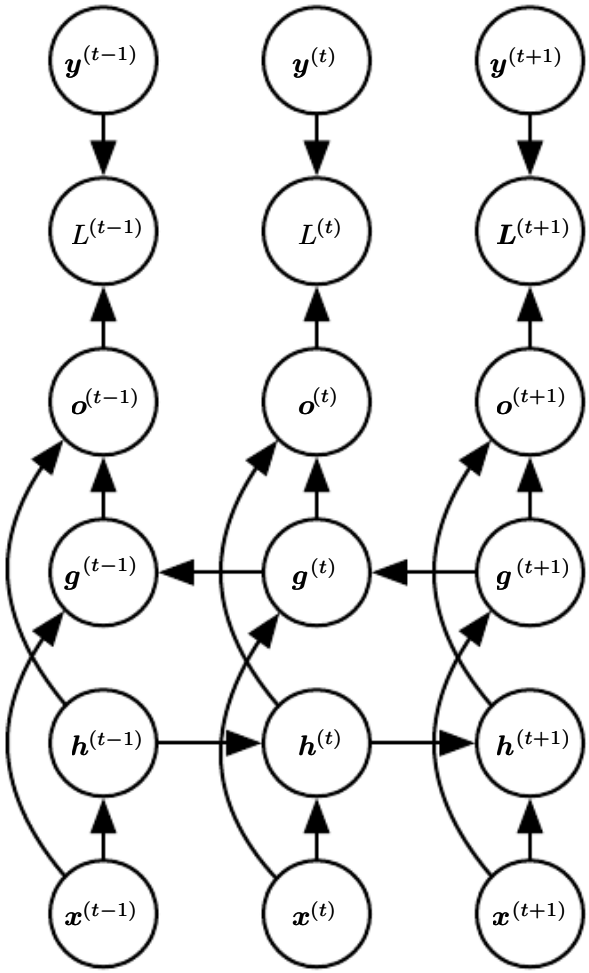
$\forall t = 1, \ldots, \tau - 1$

$$\begin{aligned}
\nabla_{\boldsymbol{h}^{(t)}} L &= \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^\top (\nabla_{\boldsymbol{h}^{(t+1)}} L) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^\top (\nabla_{\boldsymbol{o}^{(t)}} L) \\
&= \boldsymbol{W}^\top \text{diag}\left(1 - \left(\boldsymbol{h}^{(t+1)}\right)^2\right) (\nabla_{\boldsymbol{h}^{(t+1)}} L) + \boldsymbol{V}^\top (\nabla_{\boldsymbol{o}^{(t)}} L)
\end{aligned}$$

4.

$$\begin{aligned}
\nabla_{\boldsymbol{c}} L &= \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}^{(t)}}\right)^\top \nabla_{\boldsymbol{o}^{(t)}} L = \sum_t \nabla_{\boldsymbol{o}^{(t)}} L \\
\nabla_{\boldsymbol{b}} L &= \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}}\right)^\top \nabla_{\boldsymbol{h}^{(t)}} L = \sum_t \text{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) \nabla_{\boldsymbol{h}^{(t)}} L \\
\nabla_{\boldsymbol{V}} L &= \sum_t \sum_i \frac{\partial L}{\partial o_i^{(t)}} \nabla_{\boldsymbol{V}^{(t)}} o_i^{(t)} = \sum_t (\nabla_{\boldsymbol{o}^{(t)}} L) \boldsymbol{h}^{(t)\top} \\
\nabla_{\boldsymbol{W}} L &= \sum_t \sum_i \frac{\partial L}{\partial h_i^{(t)}} \nabla_{\boldsymbol{W}^{(t)}} h_i^{(t)} \\
&= \sum_t \text{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{h}^{(t-1)\top} \\
\nabla_{\boldsymbol{U}} L &= \sum_t \sum_i \frac{\partial L}{\partial h_i^{(t)}} \nabla_{\boldsymbol{U}^{(t)}} h_i^{(t)} \\
&= \sum_t \text{diag}\left(1 - \left(\boldsymbol{h}^{(t)}\right)^2\right) (\nabla_{\boldsymbol{h}^{(t)}} L) \boldsymbol{x}^{(t)\top}
\end{aligned}$$

4 Bidirectional RNNs



- $\boldsymbol{h}^{(t)}$: state of the sub-RNN that moves forward through time
- $\boldsymbol{g}^{(t)}$: state of the sub-RNN that moves backward through time
- $\boldsymbol{o}^{(t)}$: compute a representation that depends on both the past and the future but is most sensitive to the input values around time t

Motivation

- The prediction $\boldsymbol{y}^{(t)}$ may depend on the whole input sequence.

Applications

1. In speech recognition, handwriting recognition and many other sequence-to-sequence learning tasks:
   The correct interpretation of the current sound as a phoneme may depend on the next few phonemes because of co-articulation and may even depend on the next few words because of the linguistic dependencies between nearby words.

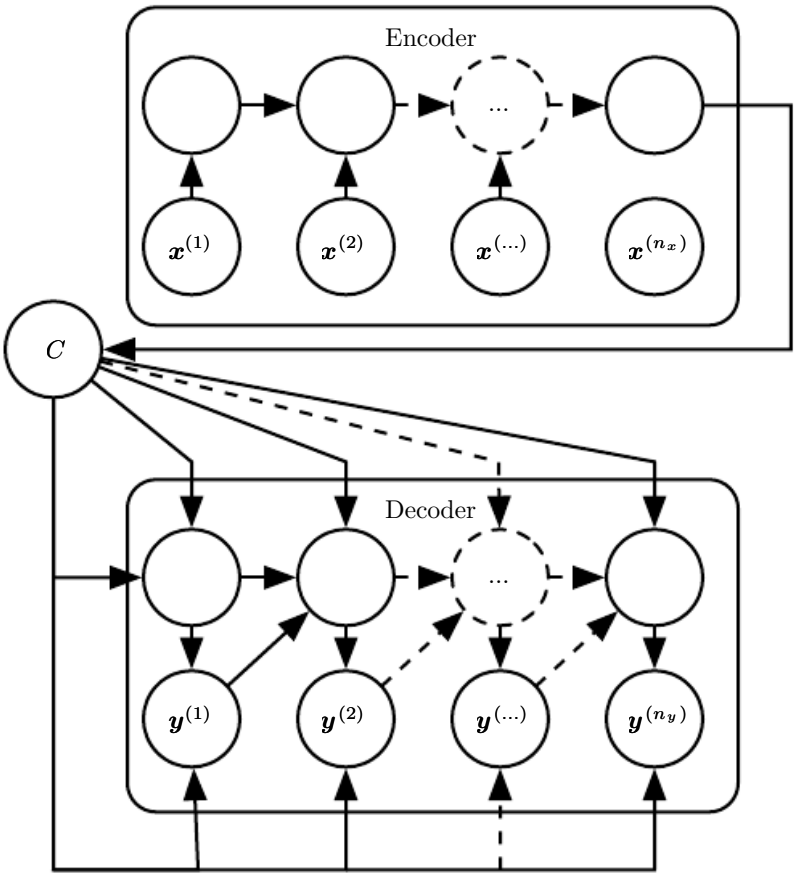
Variants

1. For two-dimensional input (such as images):
   - Having four RNNs, each on going in one of the four directions
   - At each point (i, j) of a $2-D$ grid, an output $O_{i,j}$ could then compute a representation that would capture mostly local information but could also depend on long-range inputs.
   - Restrictions:
     (a) Typically more expensive compared to a convolutional network.

5 Encoder-Decoder or Sequence-to-Sequence Architectures

The input to RNN is often called the "context". We want to produce a representation of this context, C. The context C might be a vector or sequence of vectors that summarize the input sequence $\boldsymbol{X} = (\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n_x)})$. The idea is as follow

- An **encoder** or **reader** or **input** RNN processes the input sequence. The encoder emits the context C, usually as a simple function of its final hidden state $\boldsymbol{h}_{n_x}$.
- A **decoder** or **writer** or **output** RNN is conditioned on that fixed-length vector to generate the ouput sequence $\boldsymbol{Y} = (\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n_y)})$.



Motivation

- Map an input sequence to an output sequence not necessarily of the same length.

Characteristics

- Loss: The two RNNs are trained jointly to maximize the average of $\log P(\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(n_y)} \mid \boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(n_x)})$ over all the pairs of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Applications

1. Speech recognition, machine translation, question answering.

Restrictions

- If the context C has a dimension that is too small to properly summarize a long sequence. (see **attention mechanisme** for solutions)

6 Deeper RNNs

A RNN can be deeper in many ways:

1. The hidden recurrent state can be broken down into groups organized hierarchically.

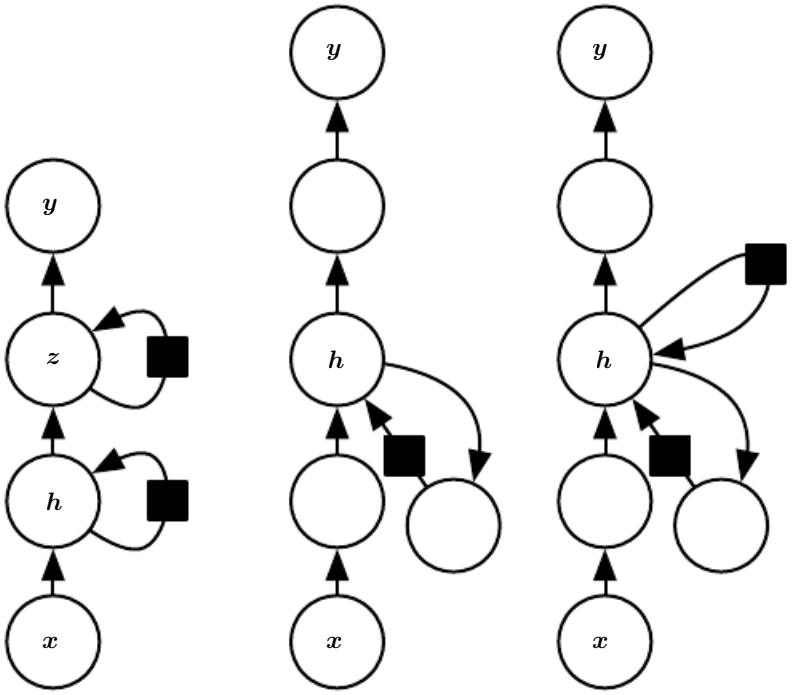
2. Deeper computation (e.g., an MLP) can be introduced in three parts:

   (a) input-to-hidden

   (b) hidden-to-hidden

   (c) hidden-to-output

   This may lengthen the shortest path linking different time steps. The path-lengthening effect can be mitigated by introducing skip connections.



(a) (b) (c)

7 Attention Mechanism

Read the whole sentence or paragraph (to get the context and the gist of what if being expressed), then produce the translated words one at a time,

each time focusing on a different part of the input sentence to gather the semantic details required to produce the next output word.
The attention-based system used to focus on specific parts of the input sequence at each time step has three components:

1. A process that reads raw data (such as source words in a source sentence) and converts them into distributed representations, with one feature vector associated with each word position.

2. A list of feature vectors storing the ouput of the reader. This can be understood as a memory containing a sequence of facts, which can be retrieved later, not necessarily in the same order, without having to visit all of them.

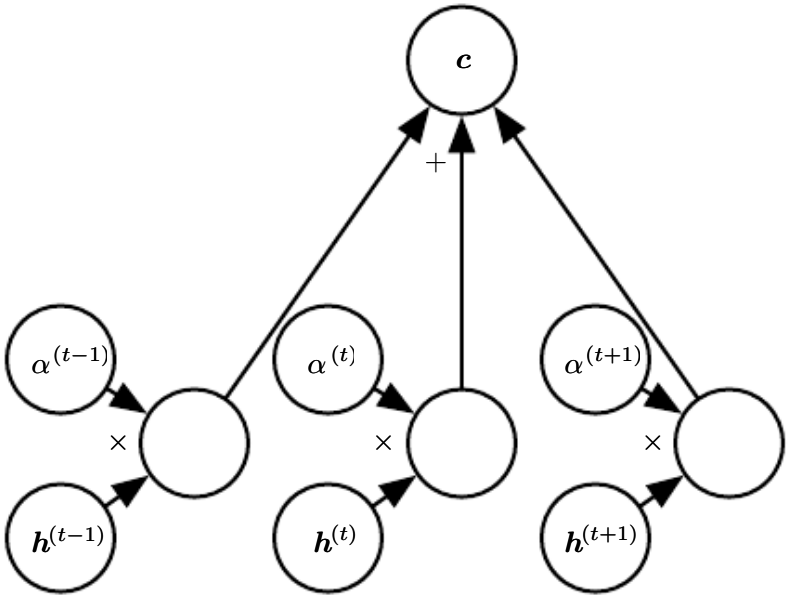
3. A process that exploits the content of the memory to sequentially perform a task, at each time step having the ability put attention on the content of one memory element (or a few, with a different weight).

The third component generates the translated sentence.

Motivation

- Using a fixed-size representation to capture all the semantic details of a very long sentence of, say, 60 words is very difficult.
  - It can be achieved by training a sufficiently large RNN well enough and for long enough.

A modern attention mechanism



- A context vector $\boldsymbol{c}$ is formed by taking aweighted average of feature vectors $\boldsymbol{h}^{(t)}$ with weights $\alpha^{(t)}$.
- The feature vectors vh may be hidden units of a neural network or raw input to the model.
- The weights $\alpha^{(t)}$ are produced by the model iteself. They are usually values in the interval $[0, 1]$ and are intended to concentrate around just one $\boldsymbol{h}^{(t)}$ so that the weighted average approximates reading that one specific time step precisely. The weight $\alpha^{(t)}$ are usually produced by applying a softmax function to relevance scores emitted by another portion of the model.
- More expensive computationally than directly indexing the desired $\boldsymbol{h}^{(t)}$, but direct indexing cannot be trained with gradient descent.