A Critical Review of Recurrent Neural Networks for Sequence Learning

Sung Eun Lee

August 01, 2024

Seoul National University

Contents

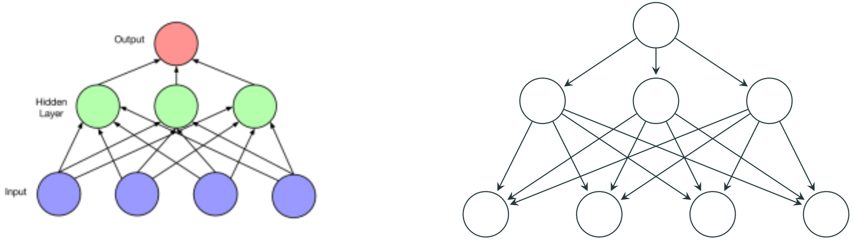
1. Introduction
2. RNN
3. Summary

Introduction

Notation

- $x^{(t)}$: Input vector at time step t.
- $h^{(t)}$: Hidden state vector at time step t.
- $\hat{y}^{(t)}$: Output vector at time step t.
- W^{hx}: Weight matrix between the input layer and the hidden layer.
- W^{hh}: Weight matrix for the hidden layer, connecting it to itself at adjacent time steps.
- W^{yh}: Weight matrix between the hidden layer and the output layer.
- b_h: Bias vector for the hidden layer.
- b_y: Bias vector for the output layer.

Feedforward Neural Network and Backpropagation



(a) Feedforward Neural Network.

(b) Backpropagation

Figure 1: Feedforward Neural Network and Backpropagation

- Learning is accomplished by iteratively updating each of the weights to minimize a loss funton $\mathcal{L}(\hat{y}, y)$, which penalizes the distance between the output $\hat{y}$ and the target y.
- Backpropagation uses the chain rule to calculate the derivative of the loss function $\mathcal{L}$ with respect to each parameter in the network.

Why Use RNNs?

- Many learning tasks require dealing with sequential data.
  e.g. Image Captioning, Time Series prediction, Video Analysis.

Challenge

- Standard Neural Networks have limitations.
  They often assume independence among the training and test samples.

Soultion

- RNNs have a recurrent structure that allows them to remember information from previous time steps and use it as input.
- RNNs are connectionist models that capture the dynamics of sequences via cylces in the network of nodes.

RNN

Recurrent Neural Network

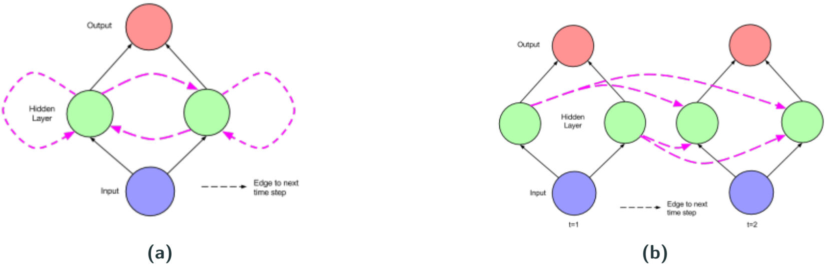


Figure 2: Simple Recurrent Network

- **Recurrent Neural Networks** are feedforward neural networks augmented by the inclusion of edges that span adjacent time stpes, introducing a notion of time to time model.
- It is then clear that the unfolded network can be trained across many time steps using backpropagation. This algorithm, called backpropagation through time (BPTT).

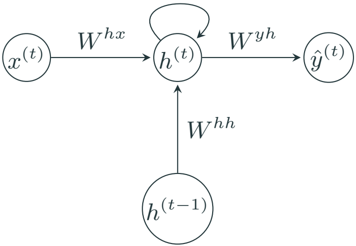


Figure 3: Detailed Structure of a Simple Recurrent Network.

- $h^{(t)} = f(W^{hx}x^{(t)} + W^{hh}h^{(t-1)} + b_h)$
  $\hat{y}^{(t)} = g(W^{yh}h^{(t)} + b_y)$, where f and g are activation functions.
- A distinctive feature of the RNN architecture is the sharing of weights across time steps.

Challenge

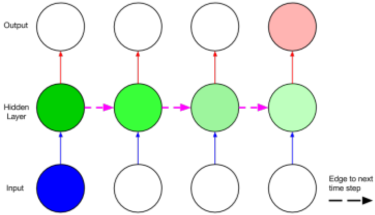


Figure 4: A Visualization of the long-term dependency problem.

- In RNNs, vanishing and exploding gradients occur when the recurrent weight is less than or greater than 1, respectively
- **Exploding Gradients**
  The value of the loss function can become extremely large. This can cause the optimization algorithm to diverge, leading to instability in the learning process.
- **Vanishing Gradients**
  Information from earlier time stpes may not be effectively passed to the present of future, leading to long-term dependency problems.

Solution

Note

- According to [Bengio et al.(1994)], it is difficult to address the long-term dependency problem in RNNs.

Exploding Gradients

- **Truncated Backpropagation Through Time(TBPTT)** calculates the gradients by performing backpropagation only over a fixed segment of time steps from the past.
- **Gradient Clipping** is a method that limits the magnitude of the gradient to a specific threshold.

Vanishing Gradients

- **Long short-term memory(LSTM)** is a special structure of RNNs capable of learning long-term dependencies.

Summary

Summary

1. The RNN structure is suitable for sequence data and time series data.
2. RNNs encounter issues with long-term dependencies.
3. LSTM was introduced in 1997 to address the long-term dependency problem caused by gradient vanishing in RNNs.