ECE 1508: Applied Deep Learning Chapter 6: Recurrent NNs

Ali Bereyhi

ali.bereyhi@utoronto.ca

Department of Electrical and Computer Engineering University of Toronto

Winter 2025

RNN: Dataset and Learning Setting

We are now looking into a supervised learning problem where we are to

learn label from a sequence of data that has generally a temporal correlation

Let's denote the sequence with $\mathbf{x}[1], \dots, \mathbf{x}[T]$

Temporal Correlation

By temporal correlation we mean that entries at other time instances carry information about one particular entry $\mathbf{x}[t]$

- + But how is a label assigned to this sequence?
- Well, that can be of various forms!

We considered a very simple case: many-to-many type I



In this case, every entry has a label

 $\, \, \downarrow \, \,$ speech tagging: $\mathbf{x}[t]$ is a part of speech and $\mathbf{y}[t]$ is its tag

We considered a very simple case: many-to-many type II



In this case, we get a label once after multiple entries

 \downarrow speech recognition: $\mathbf{x}[t]$ is a small part of speech and $\mathbf{y}[t]$ says what is a every couple of minutes about

Applied Deep Learning

We considered a very simple case: many-to-many type III



In this case, we start to get labels after some delay

 \downarrow language translation: $\mathbf{x}[1], \ldots, \mathbf{x}[t-1]$ is a sentence in German and $\mathbf{y}[t-1], \ldots, \mathbf{y}[T]$ is its translation to English

Applied Deep Learning

Chapter 6: RNNs

We considered a very simple case: one-to-many



In this case, we get only one input data and have a sequence of labels

 \downarrow image captioning: x[1] is an image and $y[1], \ldots, y[T]$ is a caption describing what is inside the image

Applied Deep Learning

Chapter 6: RNNs

We considered a very simple case: many-to-one



In this case, we get only one label for a whole input sequence

 \downarrow sequence classification: $\mathbf{x}[1], \ldots, \mathbf{x}[T]$ is a speech and $\mathbf{y}[T]$ says if this speech is constructive or destructive

Applied Deep Learning

7/16



In fact, FNNs are one-to-one RNNs

- → we can think of every data-point as a sequence of length one, or
- → we may think of dataset as a long sequence with no temporal correlation

Applied Deep Learning

Chapter 6: RNNs

RNN: General Form

. . .

To construct an RNN, we can use any module that we have learned:

- we can use a fully-connected layer
- we can use a convolutional layer
- we can use a residual unit
- we may use an inception unit used in GoogLeNet

But, classical implementations use *fully-connected layers*

We can use one layer to make a shallow RNN or multiple to make a deep RNN

- + Don't we do any change to them?
- Not a serious change
 - \downarrow We may change the activation to $tanh(\cdot)$: we will see later why
 - $\, {\scriptstyle {\scriptstyle {\sf I}}} \,$ We expand the input dimension: since we need to also give memory as input

Shallow RNN

Let's break it down a bit: say the yellow box is a fully-connected layer



This layer gets input $\mathbf{x} \in \mathbb{R}^N$ and returns activated feature $\mathbf{h} \in \mathbb{R}^M$

- We replace its activation by $anh{(\cdot)}$
- We modify it to get a new input in \mathbb{R}^{N+M}

 - $\, {\scriptstyle {\scriptstyle {\rm L}}} \, M$ entries for features ${\rm {\bf h}}$ in time t-1

Shallow RNN

Let's break it down a bit: say the yellow box is a fully-connected layer



We show it this way now

- We call $\mathbf{h}[t]$ usually the hidden state
- We pass the hidden state through an output layer
 - → Output is not necessarily corresponding to label

Shallow RNN

We may show our shallow RNN compactly via the following diagram



In this diagram

- Each edge is a set of weights, e.g., a weight matrix
- The return edge also has a delay in time
- + But, isn't that simply Elman Network?
- If we use a fully-connected layer and sigmoid activation; then, Yes! But, Remember that
 - Elman did not train it over time
 - Elman in its model used sigmoid activation

Deep RNN

We can add more layers to make a deep RNN



In this diagram

- Each edge is a set of weights, e.g., a weight matrix
- The return edge also has a delay in time
- And, it's no more Elman Network

Deep RNN

It might be easier to think of it as the following diagram



We can use any module that we like

- We may add skip connection
- We could make dense connections
- We may skip over time
- We may replace hidden layers with convolutions or anything else

Shallow RNN: Elman-Like Network

Let's start with simple RNN: a shallow RNN with fully-connected hidden layer

- + You mean Elman Network?
- Yes, but we now try to address the challenges that Elman did not address

Let's look at the flow of information once again



Shallow RNN: Forward Propagation

Let's specify the learnable parameters with some colors



Say we set the activation to $f\left(\cdot\right)$

- **1** We have $\mathbf{y}[t] = f(\mathbf{W}_2\mathbf{h}[t])$: we can learn \mathbf{W}_2
- 2 We have $\mathbf{h}[t] = f(\mathbf{W}_1 \mathbf{x}[t] + \mathbf{W}_m \mathbf{h}[t-1])$: we can learn \mathbf{W}_1 and \mathbf{W}_m
- $\mathbf{3}$ We can start with any hidden state: this means we can learn $\mathbf{h}[0]$ too