RNN’s Backpropagation Through Time

Fundamentals of Mathematics for Computer Science [CO5097]_[CH01]

Group 8

- Le Cong Tu - 2570367
- Le Nguyen Gia Nghi - 2570264
- Nguyen Nhat Duy - 2570040
- Tran Thi Van Anh - 2570394

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

BK TP.HCM

1.1 History of Model Development

Autoregressive Models - Latent Autoregressive Models

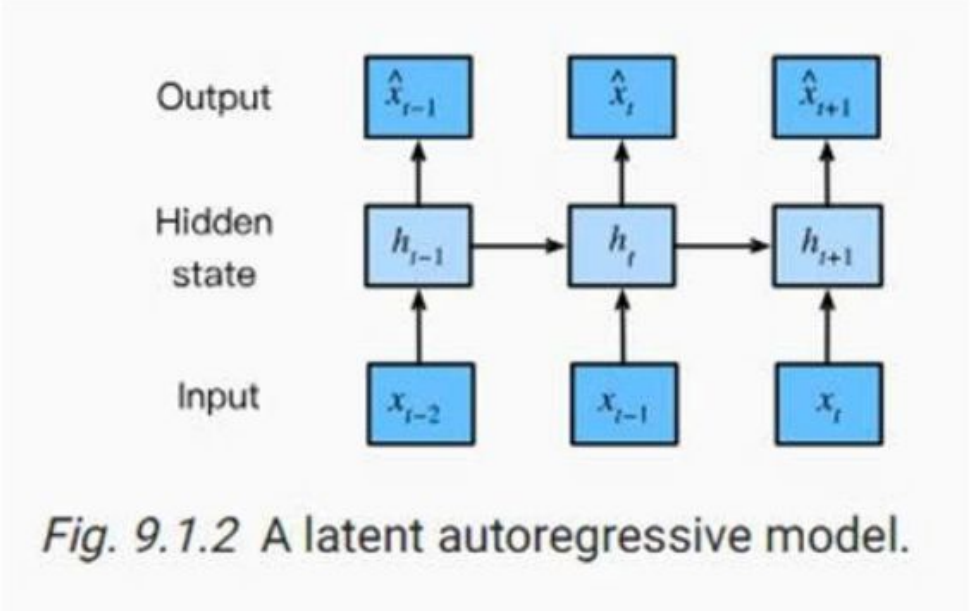


Fig. 9.1.2 A latent autoregressive model.

- **Inputs:** current and past values $(x_t, x_{t-1}, \ldots)$
- **Hidden state:** A summary of the past observations that evolves over time and is updated at each step.
- **Output:** predicted value of signal $\hat{x}_t$

Maintain a hidden state that summarizes the past observations and updates it along with the prediction at each time step.

$$\hat{x}_t = P(x_t \mid h_t)$$

The hidden state is updated using a function:

$$h_t = g(h_{t-1}, x_{t-1})$$

The hidden state is never directly observed

1.1 History of Model Development

Learning Language Model

$$P(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} P(x_t \mid x_1, \ldots, x_{t-1}).$$

$$\begin{aligned} &P(\text{deep}, \text{learning}, \text{is}, \text{fun}) \\ =&P(\text{deep})P(\text{learning} \mid \text{deep})P(\text{is} \mid \text{deep}, \text{learning})P(\text{fun} \mid \text{deep}, \text{learning}, \text{is}). \end{aligned}$$

N-gram models help simulate the occurrence of words in a nearby context without having to compute the entire text sequence

To compute a language model, we must estimate

$$P(x_1, x_2, x_3, x_4) = P(x_1)P(x_2)P(x_3)P(x_4),$$

$$P(x_1, x_2, x_3, x_4) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_2)P(x_4 \mid x_3),$$

$$P(x_1, x_2, x_3, x_4) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2)P(x_4 \mid x_2, x_3).$$

1.1 History of Model Development

Perplexity and Cross-Entropy

Perplexity is an important measurement in natural language processing (NLP), defined as the exponential of cross-entropy

Cross-Entropy Loss

$$\frac{1}{n}\sum_{t=1}^{n} -\log P(x_t \mid x_{t-1}, \ldots, x_1)$$

P is the model's predicted probability for the correct next token

x_t is the actual token in the sequence

Perplexity Definition

$$\exp\left(-\frac{1}{n}\sum_{t=1}^{n}\log P(x_t \mid x_{t-1}, \ldots, x_1)\right).$$

Perplexity measures how unexpected the model is when predicting tokens in the chain.
Models with lower perplexity have more accurate predictions.

1.1 History of Model Development

Neural Networks Without Hidden States

Input: $X \in \mathbb{R}^{n \times d}$

Hidden layer output: $H = \phi(X\mathbf{W}_{xh} + \mathbf{b}_h)$

- $H \in \mathbb{R}^{n \times h}$
- W = weight matrix between input and hidden layer
- b_h = bias vector

Output Layer $\mathbf{O} = \mathbf{H}\mathbf{W}_{hq} + \mathbf{b}_q,$

$\mathbf{W}_{hq} \in \mathbb{R}^{h \times q}$ is the weight parameter,
and $\mathbf{b}_q \in \mathbb{R}^{1 \times q}$ is the bias parameter of the output layer.

1.2 Problem statement

A Recurrent Neural Network (RNN) is a class of artificial neural networks specifically designed to handle sequential data or time series data. Unlike traditional feedforward networks (like MLPs) where the input and output are independent of each other, an RNN leverages the concept of internal memory to process sequences of inputs. This means that the output at any step is dependent not only on the current input but also on the information learned from the previous steps in the sequence.

The Core Mechanism: The Hidden State The fundamental distinguishing feature of an RNN is its recurrent connection, which allows information to cycle back into the network. This forms the hidden state (or context vector), ht, which acts as the network's memory.

1.3 Applications

RNNs are fundamental to modern sequential data processing across Computer Science:

- **Language Modeling**: Predicting the next word in a sequence (e.g., text auto-completion, generating coherent text).
- **Speech Recognition**: Transcribing a sequence of acoustic features into a sequence of words.
- **Time Series Prediction**: Forecasting future values based on historical data (e.g., stock prices, temperature).
- **Video Prediction**: Predicting the subsequent frames in a video sequence.
- **Anomaly Detection**: Identifying unusual patterns or events in sequential data (e.g., detecting fraudulent transactions, sensor malfunctions).

1.4 Motivation

RNNs offer distinct advantages over traditional Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) when dealing with sequences:

- **Contextual Memory**: MLPs and standard CNN's lack a mechanism to retain information processed in previous steps. The RNN's hidden state (h_{t-1}) explicitly provides a dynamic, weighted memory of the entire preceding sequence, allowing it to capture long-range dependencies and context.
- **Variable-Length Sequence Handling**: MLPs require fixed-size inputs. RNNs can easily process sequences of arbitrary or varying lengths because the core computational structure and weights are reused at every time step.
- **Parameter Sharing**: By utilizing the same set of weights (W_h, W_x, W_v) across all time steps, RNNs are highly parameter-efficient and learn generalized sequential patterns, rather than needing a new set of weights for every position in the sequence.

1.5 Conclusion

RNNs are the foundational architecture for sequence processing, leveraging a hidden state as memory. They are trained using Backpropagation Through Time (BPTT) to learn sequential patterns and long-term dependencies in the data.

2.1 RECURRENT NEURAL NETWORK

2.1.1 Overview

An RNN can be visualized as a cycle in a diagram. When "unrolled" over time, it becomes a chain of interconnected modules, one for each time step t.

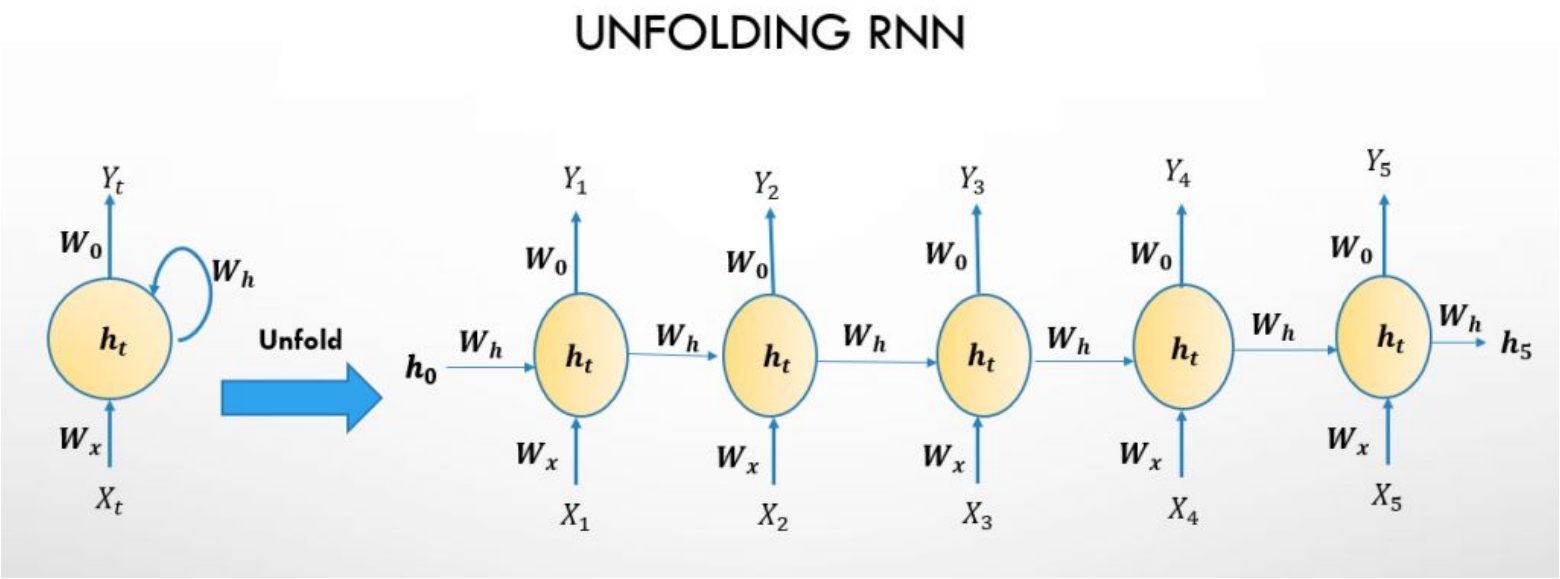
* **Input (x_t)**: The data element at the current time step (e.g., a word vector, a single time-series reading).
* **Hidden State (h_t)**: The network's memory at time t. This vector encapsulates all information learned from x_1 up to x_t. It is computed using both the current input x_t and the previous hidden state h_{t-1}.
* **Output (y_t)**: The output of the network at time . This could be a prediction for x_{t+1} (in language modeling) or a classification/score based on the sequence so far.

For example: We need to build an application to Recognizing actions in a 30s video.

This is a many to one problem in RNN, meaning many inputs and 1 output.

2.1 RECURRENT NEURAL NETWORK

2.1.1 Overview

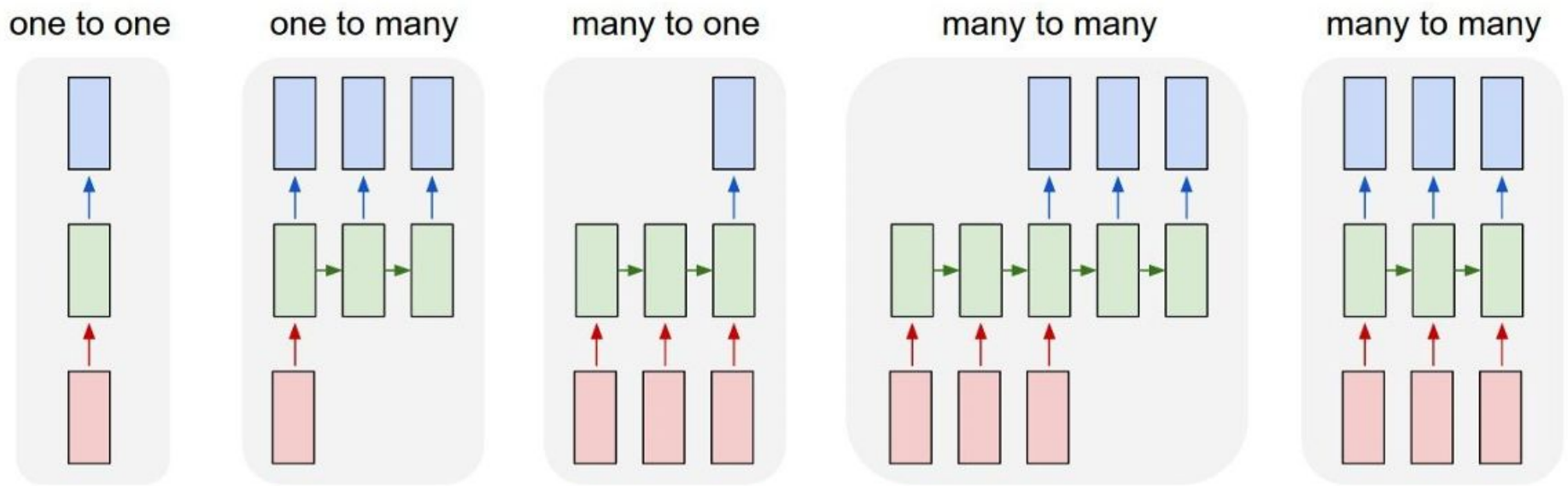


Be aware that the diagram above only illustrates how the RNN is unfolded through time. At each time step t, the RNN functions like a neural network layer: the hidden state contains multiple neurons, each with its own activation function. Both X(t) and Y(t) are column vectors, and their output dimensions can be larger than a single node.

2.1.2 RNN problem classification

- One to one: One single input leads to one single output. This is the standard mapping for most basic tasks. Example: Image Classification.
- One to many: One single input leads to a sequence of outputs. The network generates multiple output steps based on a single initial input context. Example: Image Captioning.
- Many to one: A sequence of inputs leads to one single output. The network processes the entire sequence to arrive at a final classification or prediction. Example: Video Action Classification.
- Many to many: A sequence of inputs leads to a sequence of outputs. Example: Machine Translation.

2.1.2 RNN problem classification



Without loss of generality, we focuses on analyzing the final many-to-many architecture, because in its earlier versions, when there is no input, we can set the input to 0 at those time steps. Similarly, for the many-to-one case, at time steps where no target is required, we do not propagate gradients or update weights for those losses.

2.2 Forward Propagation

2.2.1 Hidden states

The hidden state at the current time step t, H_t is calculated using both the current input X_t and the previous hidden state H_{t-1} .

$$H_t = \phi(W_{hx}X_t + W_{hh}H_{t-1} + b_h).$$

$$net_t = W_{hx}X_t + W_{hh}H_{t-1} + b_h .$$

Where:

- X_t: Input at time t.
- W_{hx}: Weights for the current input.
- H_{t-1}:Hidden state from the previous time step.
- W_{hh}: Weights for the previous hidden state (the recurrent weight).
- b_h: Bias for the hidden layer.
- ϕ: Activation function.

2.2 Forward Propagation

2.2.1 Hidden states

Case study: Build 1 RNN to predict the next character in the string

$$d \rightarrow e \rightarrow m \rightarrow o$$

RNN Configuration

2.2.1 Hidden states

Input → Hidden:

$$W_{hx} = \begin{bmatrix} 0.5 & -0.3 & 0.1 & 0.2 \\ -0.2 & 0.4 & 0.3 & -0.1 \end{bmatrix}$$

Hidden → Hidden

$$W_{hh} = \begin{bmatrix} 0.1 & 0.2 \\ 0.0 & 0.3 \end{bmatrix}$$

Hidden bias

$$b_h = \begin{bmatrix} 0.05 \\ -0.02 \end{bmatrix}$$

Hidden → Output

$$W_{qh} = \begin{bmatrix} 0.3 & 0.1 \\ -0.2 & 0.4 \\ 0.1 & -0.3 \\ 0.2 & 0.2 \end{bmatrix}$$

Output bias

$$b_q = \begin{bmatrix} 0.01 \\ -0.03 \\ 0.02 \\ 0.00 \end{bmatrix}$$

Activation Function: tanh

2.2.1 Hidden states

Sequence

t	Input x_t	Target y_t
1	d	e
2	e	m
3	m	o

- Input x_t : current character.
- Target y_t : next character.

2.2.1 Hidden states

Encoding one-hot

Rule: Vector will have dimension = number of characters = 4.

Character	One-hot
d	1000
e	0100
m	0010
o	0001

2.2.1 Hidden states

- Calculate Hidden State h_1:

$$H_1 = \phi(W_{hx} X_1 + W_{hh} H_0 + b_h).$$

$$=> net_1 = \begin{bmatrix} 0.5 & -0.3 & 0.1 & 0.2 \\ -0.2 & 0.4 & 0.3 & -0.1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 \\ 0.0 & 0.3 \end{bmatrix} \cdot \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix} + \begin{bmatrix} 0.05 \\ -0.02 \end{bmatrix}$$

$$= \begin{bmatrix} 0.5 \\ -0.2 \end{bmatrix} + \begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix} + \begin{bmatrix} 0.05 \\ -0.02 \end{bmatrix} = \begin{bmatrix} 0.55 \\ -0.22 \end{bmatrix}$$

$$H1 = \phi(net1) = \phi(\begin{bmatrix} 0.55 \\ -0.22 \end{bmatrix}) = \begin{bmatrix} 0.5005 \\ -0.2165 \end{bmatrix}.$$

2.2.1 Hidden states

- Calculate Hidden State h_1:

$$\Rightarrow o_1 = Wqh \cdot H1 + bq = \begin{bmatrix} 0.3 & 0.1 \\ -0.2 & 0.4 \\ 0.1 & -0.3 \\ 0.2 & 0.2 \end{bmatrix} \begin{bmatrix} 0.500 \\ -0.217 \end{bmatrix} + \begin{bmatrix} 0.01 \\ -0.03 \\ 0.02 \\ 0.00 \end{bmatrix} = \begin{bmatrix} 0.1385 \\ -0.2167 \\ 0.1350 \\ 0.0568 \end{bmatrix}$$

$$\hat{y}_1 = \text{Softmax}(o_1)$$

$$\Rightarrow \hat{y}_1 \approx \begin{bmatrix} 0.2763 \\ 0.1937 \\ 0.2753 \\ 0.2546 \end{bmatrix}$$

2.2.1 Hidden states

- Calculate Hidden State h_2 :

$$=> H_2 = \phi(\begin{bmatrix} 0.5 & -0.3 & 0.1 & 0.2 \\ -0.2 & 0.4 & 0.3 & -0.1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 \\ 0.0 & 0.3 \end{bmatrix} \cdot \begin{bmatrix} 0.500 \\ -0.217 \end{bmatrix} + \begin{bmatrix} 0.05 \\ -0.02 \end{bmatrix}) :$$

$$= \phi(\begin{bmatrix} -0.2433 \\ 0.315 \end{bmatrix}) = \begin{bmatrix} -0.2386 \\ 0.3050 \end{bmatrix}$$

$$=> o_2 = \begin{bmatrix} 0.3 & 0.1 \\ -0.2 & 0.4 \\ 0.1 & -0.3 \\ 0.2 & 0.2 \end{bmatrix} \begin{bmatrix} -0.2386 \\ 0.3050 \end{bmatrix} + \begin{bmatrix} 0.01 \\ -0.03 \\ 0.02 \\ 0.00 \end{bmatrix} = \begin{bmatrix} -0.0311 \\ 0.1397 \\ -0.0954 \\ 0.0133 \end{bmatrix} \quad => \hat{y}_2 \approx \begin{bmatrix} 0.2398 \\ 0.2845 \\ 0.2249 \\ 0.2507 \end{bmatrix}$$

2.2.1 Hidden states

- Calculate Hidden State h_3 :

$$=> H_3 = \phi(\begin{bmatrix} 0.1 \\ 0.3 \end{bmatrix} + \begin{bmatrix} 0.1 & 0.2 \\ 0.0 & 0.3 \end{bmatrix} \begin{bmatrix} -0.2386 \\ 0.3050 \end{bmatrix} + \begin{bmatrix} 0.05 \\ -0.02 \end{bmatrix})$$

$$= \phi(\begin{bmatrix} 0.1871 \\ 0.3715 \end{bmatrix}) = \begin{bmatrix} 0.1850 \\ 0.3553 \end{bmatrix}$$

$$=> o_3 = \begin{bmatrix} 0.3 & 0.1 \\ -0.2 & 0.4 \\ 0.1 & -0.3 \\ 0.2 & 0.2 \end{bmatrix} \begin{bmatrix} 0.1850 \\ 0.3553 \end{bmatrix} + \begin{bmatrix} 0.01 \\ -0.03 \\ 0.02 \\ 0.00 \end{bmatrix} = \begin{bmatrix} 0.1010 \\ 0.0751 \\ -0.0681 \\ 0.1081 \end{bmatrix} => \hat{y}_3 \approx \begin{bmatrix} 0.2614 \\ 0.2547 \\ 0.2207 \\ 0.2632 \end{bmatrix}$$

2.2.2 Output Layer

Next, the hidden layer output H is used as input of the output layer, which is given by

$$O_t = W_{qh} H_t + b_q .$$

- O_t: Output Vector at time step t.
- H_t: Hidden State Vector at time step t.
- W_{qh}: Weight Matrix connecting the hidden layer to the output layer.
- b_q: Bias Vector for the output layer.

.

2.2.3 Prediction

For classification tasks (like character prediction), the logits are converted into a probability distribution $\hat{y}_t$ using the Softmax function:

$$\hat{\mathbf{y}}_t = \mathrm{softmax}(\mathbf{o}_t)$$

In which each element of o_t is converted into a probability:

$$\hat{y}_{t,i} = \frac{\exp(o_{t,i})}{\sum_{j=1}^{K} \exp(o_{t,j})}$$

Unlike argmax, which only picks the largest score, Softmax converts all outputs into probabilities (sum = 1), allowing us to see how confident the model is for every class.

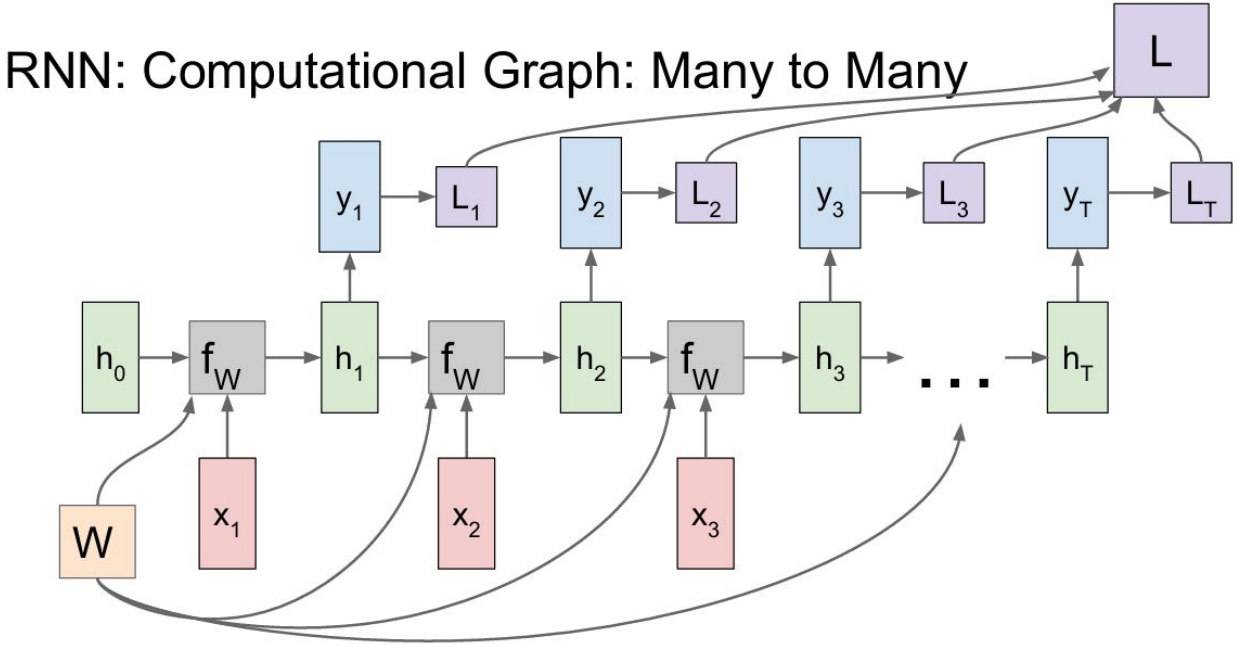
2.3 LOSS FUNCTION

- In Supervised Learning, the Loss Function (also called the Cost Function or Objective Function) is the compass that guides the training process.
- It quantifies the "distance" or error between the model's predictions and the actual ground truth.

$$L(\mathbf{x}_1, \ldots, \mathbf{x}_T, \mathbf{y}_1, \ldots, \mathbf{y}_T, \mathbf{w}_h, \mathbf{w}_o) = \frac{1}{T} \sum_{t=1}^{T} l(\mathbf{y}_t, \hat{\mathbf{y}}_t).$$

- y_t: The ground truth (actual target) at time t.
- $\hat{\mathbf{y}}_t$: The network's prediction at time t.
- ℓ: The specific error function.
- T: The total length of the sequence.

2.3 LOSS FUNCTION



Although an unfolded RNN may resemble a CNN, since each time step can be viewed as a layer, but the crucial point is that all these “layers” share the same weights across time. Another major difference is that each time step produces its own loss, and the total loss is the sum over all time steps. As a result, RNNs involve both local and global derivatives, not just spatially but also temporally.

2.4 BACKPROPAGATION THROUGH TIME

Training a neural network requires more than just defining its architecture and computing output predictions. The essential challenge lies in determining how to adjust the network’s parameters so that its predictions gradually become more accurate. Backpropagation is the algorithm that enables this learning process.

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient at the Output Layer $(\frac{\partial L_t}{\partial o_t})$

$$\frac{\partial L_t}{\partial o_t} = \sum_i \frac{\partial L}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial o_t}$$

Activation function: tanh

Softmax (*):

$$\hat{y}_t = \frac{e^{o_t}}{\sum_{i=1}^{T} e^{o(i)}}$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient at the Output Layer $(\frac{\partial L_t}{\partial o_t})$

$$\hat{y}_t = \frac{e^{o_t}}{\sum_{i=1}^{T} e^{o(i)}}$$

$$=> \frac{\partial \hat{y}_t}{\partial o_t} = \frac{\left(e^{o_t} \cdot \sum_{t=1}^{T} e^{o_t}\right) - \left(e^{o(i)} \cdot e^{o_t}\right)}{\left(\sum_{i=1}^{T} e^{o(i)}\right)^2}$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient at the Output Layer $(\frac{\partial L_t}{\partial o_t})$

Case 1: i = t (Correct character)

$$=> \frac{\partial \hat{y}_t}{\partial o_t} = \frac{e^{o_t}}{\sum_{i=1}^{T} e^{o(i)}} - \frac{e^{o_t}}{\sum_{i=1}^{T} e^{o(i)}} \cdot \frac{e^{o_t}}{\sum_{i=1}^{T} e^{o(i)}}$$

$$=> \frac{\partial \hat{\mathbf{y}}_\mathbf{t}}{\partial \mathbf{o_t}} = \hat{\mathbf{y}}_\mathbf{t} - \hat{\mathbf{y}}_\mathbf{t} \cdot \hat{\mathbf{y}}_\mathbf{t} = \hat{\mathbf{y}}_\mathbf{t}(\mathbf{1} - \hat{\mathbf{y}}_\mathbf{t})$$

Case 2: i != t (Wrong character)

$$=> \frac{\partial \hat{y}_t}{\partial o_t} = -\frac{e^{o(i)} \cdot e^{o_t}}{\left(\sum_{i=1}^{T} e^{o(i)}\right)^2}$$

$$=> \frac{\partial \hat{y}_t}{\partial o_t} = -\left(\frac{e^{o(i)}}{\sum_{i=1}^{T} e^{o(i)}}\right) \cdot \left(\frac{e^{o_t}}{\sum_{i=1}^{T} e^{o(i)}}\right)$$

$$=> \frac{\partial \hat{\mathbf{y}}_\mathbf{t}}{\partial \mathbf{o_t}} = -\hat{\mathbf{y}}_\mathbf{i}\hat{\mathbf{y}}_\mathbf{t}$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient at the Output Layer $(\frac{\partial L_t}{\partial o_t})$

Cross-enropy loss (**):

$$L = -\sum_{i=1}^{T} y_i \ln(\hat{y}_i)$$

$$=> \frac{\partial L}{\partial \hat{y}_t} = \frac{\partial}{\partial \hat{y}_t}\left[-\sum_{i=1}^{T} y_i \ln(\hat{y}_i)\right]$$

$$=> \frac{\partial L_t}{\partial \hat{y}_t} = \frac{\partial}{\partial \hat{y}_t}[-y_t \ln(\hat{y}_t)]$$

$$==> \frac{\partial L_t}{\partial \hat{y}_t} = -y_t \cdot \frac{1}{\hat{y}_t}$$

$$==> \frac{\partial L_t}{\partial \hat{y}_t} = -\frac{y_t}{\hat{y}_t}$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient at the Output Layer $(\frac{\partial L_t}{\partial o_t})$

From (*) and (**):

$$\frac{\partial L_t}{\partial o_t} = \sum_i \frac{\partial L}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial o_t}$$

$$=> \frac{\partial L_t}{\partial o_t} = -\frac{y_t}{\hat{y}_t} \cdot \hat{y}_t(1-\hat{y}_t) + \sum_{i \neq t} \left(-\frac{y_t}{\hat{y}_t}\right) \cdot (-\hat{y}_i \hat{y}_t)$$

$$=> \frac{\partial L_t}{\partial o_t} = -y_t \cdot (1-\hat{y}_t) + \sum_{i \neq t} \left(-\frac{y_t}{\hat{y}_t}\right) \cdot (-\hat{y}_i \hat{y}_t)$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient at the Output Layer $(\frac{\partial L_t}{\partial o_t})$

Cause we one-hot encoding, y_t at wrong character equal 0

$$=> \frac{\partial L_t}{\partial o_t} = -y_t \cdot (1 - \hat{y}_t)$$

$$=> \frac{\partial L_t}{\partial o_t} = \hat{y}_t - y_t$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient for Bias $(\frac{\partial L}{\partial b_q})$

$$\frac{\partial L}{\partial b_q} = \frac{1}{T} \sum_{t=1}^{T} \frac{\partial L_t}{\partial o_t}.$$

Gradient for Output Weights (W_{qh})

The overall gradient is the average of the gradients at each time step:

$$\frac{\partial L}{\partial W_{qh}} = \frac{1}{T} \sum_{t=1}^{T} \frac{\partial L_t}{\partial W_{qh}}$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient for Output Weights (W_{qh})

Apply Chain rule in t:

$$\frac{\partial L_t}{\partial W_{qh}} = \frac{\partial L_t}{\partial o_t} \cdot \frac{\partial o_t}{\partial W_{qh}}$$

$$=> \frac{\partial L}{\partial W_{qh}} = \frac{1}{T} \sum_{t=1}^{T} \left(\frac{\partial L_t}{\partial o_t} h_t^T \right)$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient for Output Weights (W_{qh})

Example:

$$\frac{\partial L}{\partial W_{qh}} = \frac{1}{3}\sum_{t=1}^{3}\left(\frac{\partial L_t}{\partial o_t}\cdot h_t^T\right)$$

$$\frac{\partial L_1}{\partial o_1}\cdot h_1^T = \begin{bmatrix} 0.2763 \\ -0.8063 \\ 0.2753 \\ 0.2546 \end{bmatrix} \cdot \begin{bmatrix} 0.5005 & -0.2165 \end{bmatrix} = \begin{bmatrix} 0.1383 & -0.0598 \\ -0.4036 & 0.1746 \\ 0.1378 & -0.0596 \\ 0.1274 & -0.0551 \end{bmatrix}$$

$$\frac{\partial L_2}{\partial o_2}\cdot h_2^T = \begin{bmatrix} 0.2398 \\ 0.2845 \\ -0.7751 \\ 0.2507 \end{bmatrix} \cdot \begin{bmatrix} -0.2386 & 0.3050 \end{bmatrix} = \begin{bmatrix} -0.0572 & 0.0731 \\ -0.0679 & 0.0868 \\ 0.1849 & -0.2364 \\ -0.0598 & 0.0765 \end{bmatrix}$$

$$\frac{\partial L_3}{\partial o_3}\cdot h_3^T = \begin{bmatrix} 0.2614 \\ 0.2547 \\ 0.2207 \\ -0.7368 \end{bmatrix} \cdot \begin{bmatrix} 0.1850 & 0.3553 \end{bmatrix} = \begin{bmatrix} 0.0483 & 0.0922 \\ 0.0471 & 0.0899 \\ 0.0408 & 0.0779 \\ -0.136 & -0.26 \end{bmatrix}$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient for Output Weights (W_{qh})

Example:

$$\frac{\partial L}{\partial W_{qh}} = \frac{1}{3}\sum_{t=1}^{3}\left(\frac{\partial L_1}{\partial o_1}\cdot h_1^T + \frac{\partial L_2}{\partial o_2}\cdot h_2^T + \frac{\partial L_3}{\partial o_3}\cdot h_3^T\right)$$

$$\frac{\partial L}{\partial W_{qh}} = \frac{1}{3}\begin{bmatrix} 0.1294 & 0.1055 \\ -0.4244 & 0.3513 \\ 0.3635 & -0.218 \\ -0.068 & -0.238 \end{bmatrix}$$

$$\frac{\partial L}{\partial W_{qh}} \approx \begin{bmatrix} 0.0428 & 0.0351 \\ -0.141 & 0.1171 \\ 0.1211 & -0.072 \\ -0.022 & -0.079 \end{bmatrix}$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient for Output Weights (W_{qh})

Example:

$$\frac{\partial L}{\partial b_q} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L_t}{\partial o_t} = \frac{1}{T}\sum_{t=1}^{T}(\hat{y}_t - y_t)$$

$$\frac{\partial L_3}{\partial \mathbf{o}_3} = (\hat{y}_3 - y_3) = \begin{bmatrix} 0.2614 \\ 0.2547 \\ 0.2207 \\ 0.2632 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = [0.2614, 0.2547, 0.2207, -0.7368]^\top.$$

$$\frac{\partial L_2}{\partial \mathbf{o}_2} = (\hat{y}_2 - y_2) = \begin{bmatrix} 0.2398 \\ 0.2845 \\ 0.2249 \\ 0.2507 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = [0.2398, 0.2845, -0.7751, 0.2507]^\top.$$

$$\frac{\partial L_1}{\partial \mathbf{o}_1} = (\hat{y}_1 - y_1) = \begin{bmatrix} 0.2763 \\ 0.1937 \\ 0.2754 \\ 0.2546 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = [0.2763, -0.8063, 0.2754, 0.2546]^\top.$$

2.4.1 Gradients of the Output Layer (W_{qh}, W_{bq})

Gradient for Output Weights (W_{qh})

Example:

$$\frac{\partial L}{\partial b_q} = \frac{1}{3}\left(\frac{\partial L_1}{\partial o_1} + \frac{\partial L_1}{\partial o_2} + \frac{\partial L_3}{\partial o_3}\right)$$

$$\Rightarrow \frac{\partial L}{\partial b_q} = \frac{1}{3}\begin{bmatrix} 0.2763 \\ -0.8063 \\ 0.2754 \\ 0.2546 \end{bmatrix} + \begin{bmatrix} 0.2398 \\ 0.2845 \\ -0.7751 \\ 0.2507 \end{bmatrix} + \begin{bmatrix} 0.2614 \\ 0.2547 \\ 0.2207 \\ -0.7368 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} 0.7775 \\ -0.267 \\ -0.279 \\ -0.231 \end{bmatrix} = \begin{bmatrix} 0.2591 \\ -0.089 \\ -0.093 \\ -0.077 \end{bmatrix}$$

2.4.2 Gradients of the Recurrent Layer(W_{hh}, W_{hx}, W_{bh})

Output derivative with respect to the Recurrent Weights $(\frac{\partial o_t}{\partial h_t})$

$$o_t = W_{qh} h_t + b_q$$

$$=> \frac{\partial o_t}{\partial h_t} = \frac{\partial(\mathbf{W}_{qh} h_t + b_q)}{\partial h_t} = \mathbf{W}_{qh}^T.$$

2.4.2.1 Forward recursion

$$
\begin{aligned}
\frac{\partial L}{\partial W_{hx}} &= \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L_t}{\partial \text{net}_t}\cdot\frac{\partial \text{net}_t}{\partial W_{hx}} \\
&= \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L_t}{\partial \text{net}_t}\left(\frac{\partial \text{net}_t}{\partial W_{hx}}+\sum_{i=1}^{t-1}\frac{\partial \text{net}_t}{\partial \text{net}_i}\cdot\frac{\partial \text{net}_i}{\partial W_{hx}}\right) \\
&= \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L_t}{\partial \text{net}_t}\left(\frac{\partial \text{net}_t}{\partial W_{hx}}+\sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t}\frac{\partial \text{net}_j}{\partial \text{net}_{j-1}}\right)\cdot\frac{\partial \text{net}_i}{\partial W_{hx}}\right).
\end{aligned}
$$

Explanation:

- **First equality:** Shows that the gradient of the total loss with respect to W_{hx} is the sum of the gradients at each time step. Here, the chain rule is applied, multiplying the derivative of the loss with respect to net_t by the derivative of net_t with respect to W_{hx}.
- **Second equality:** Expands the chain rule for $\frac{\partial \text{net}_t}{\partial W_{hx}}$ over all previous time steps from t down to 1. It has two components: the first term is the local derivative at time t, and the second term sums the derivatives through the influence of net_t on all previous states net_i (Where $i = 1, \ldots, t-1$).
- **Third equality:** Expresses the second term as a product of successive chain-rule derivatives along the time steps: $\frac{\partial \text{net}_t}{\partial \text{net}_{t-1}} \cdot \frac{\partial \text{net}_{t-1}}{\partial \text{net}_{t-2}} \cdots \frac{\partial \text{net}_{i+1}}{\partial \text{net}_i}$. This clearly illustrates the **Backpropagation Through Time (BPTT)** computation.

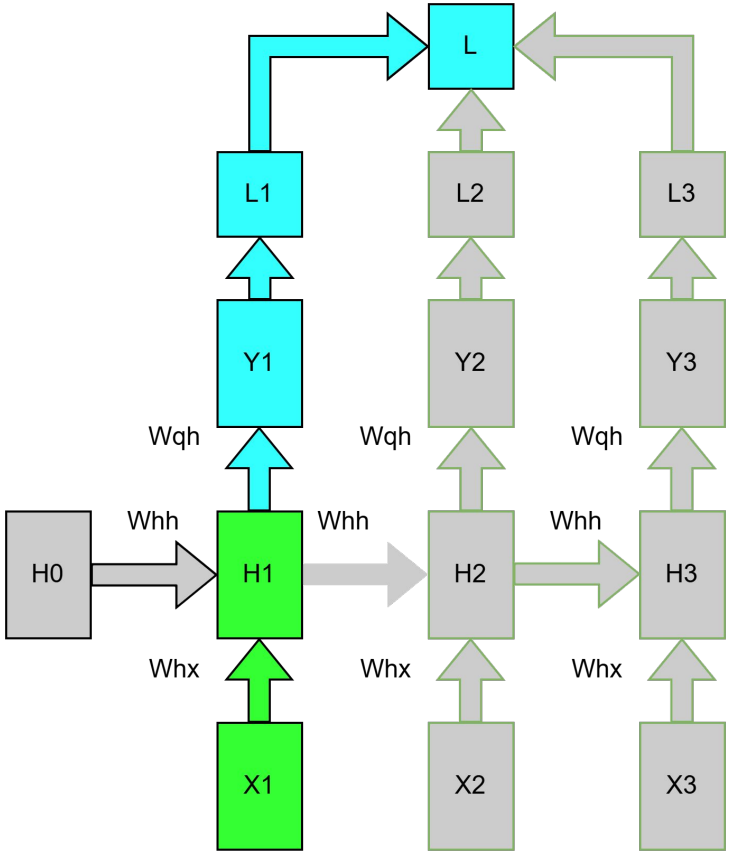
Note: The main idea of the above formula is that, at each time step t, L_t is influenced by which states at the current time and in the past. From there, at each time step t, we expand the chain rule from t to 1.

2.4.2.1 Forward recursion

For example, T=3:

- At t = 1:

$$\frac{\partial L_1}{\partial W_{hx}} = \frac{\partial L_1}{\partial \text{net}_1} \cdot \frac{\partial \text{net}_1}{\partial W_{hx}}$$

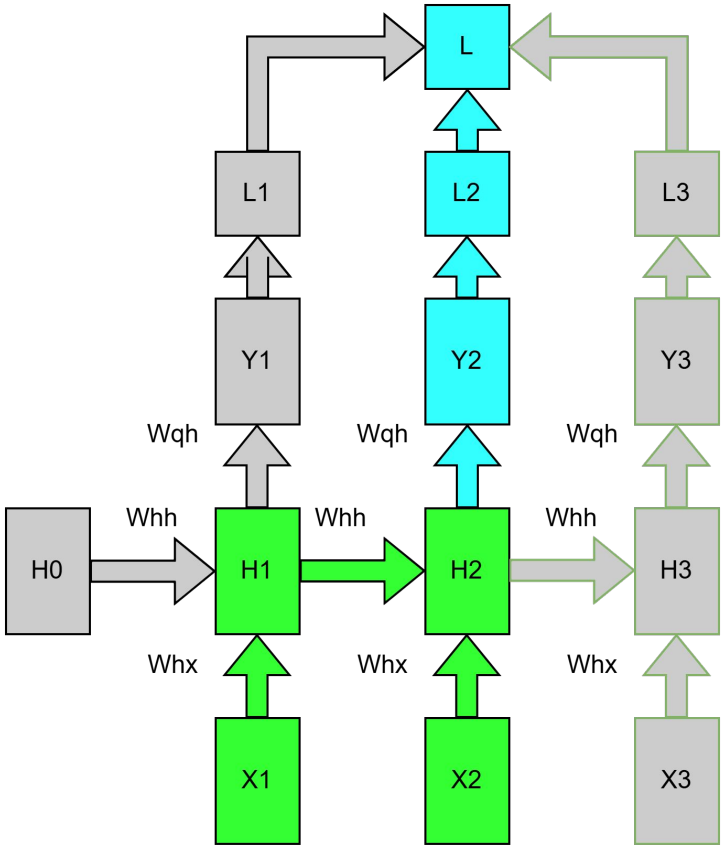


2.4.2.1 Forward recursion

For example, T=3:

- At t = 2:

$$\frac{\partial L_2}{\partial W_{hx}} = \frac{\partial L_2}{\partial \text{net}_2} \cdot \left(\frac{\partial \text{net}_2}{\partial W_{hx}} + \frac{\partial \text{net}_2}{\partial net_1} \cdot \frac{\partial \text{net}_1}{\partial W_{hx}} \right)$$

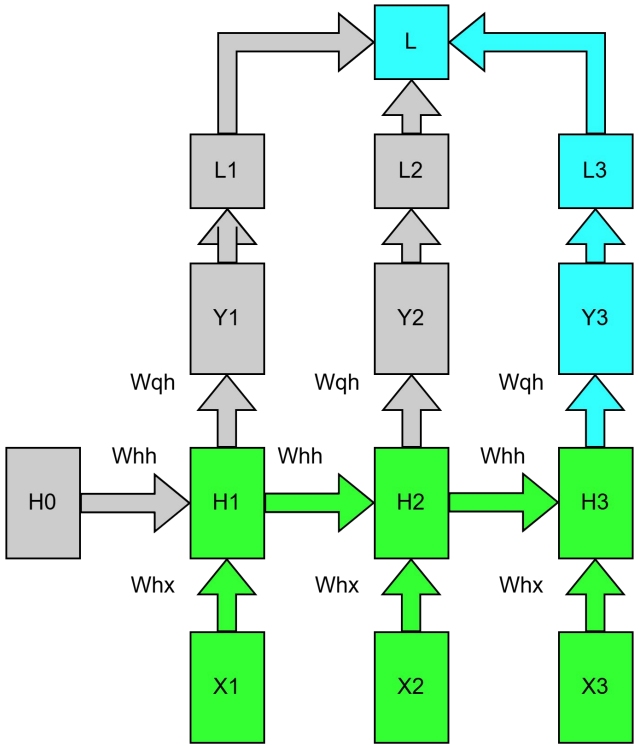


2.4.2.1 Forward recursion

For example, T=3:

- At t = 3:

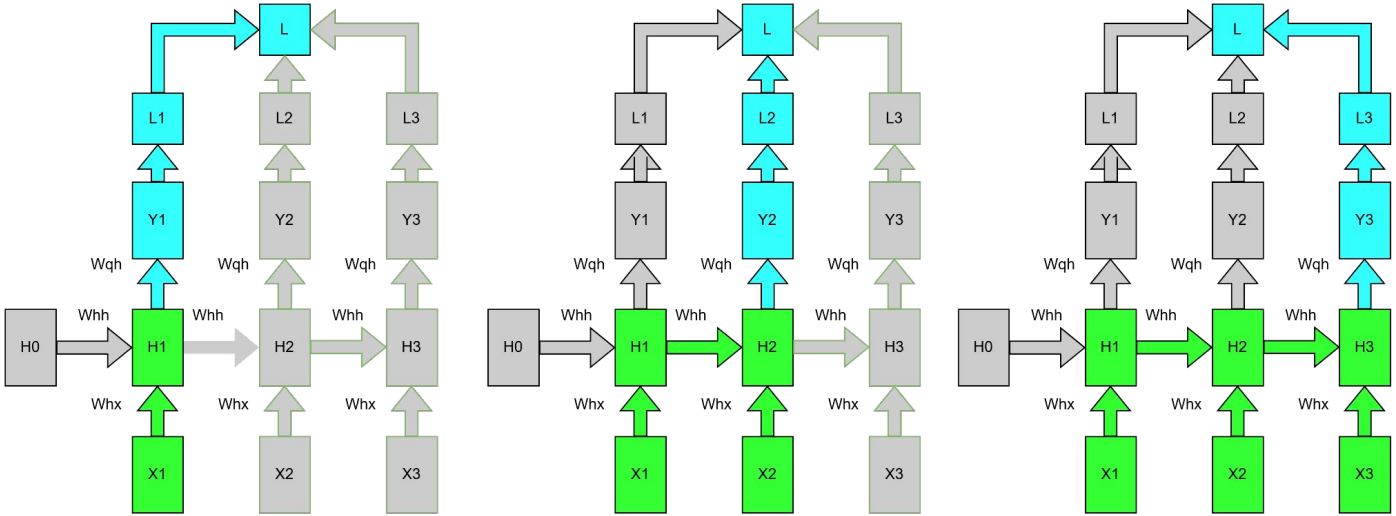
$$\frac{\partial L_3}{\partial W_{hx}} = \frac{\partial L_3}{\partial \text{net}_3} \cdot \left(\frac{\partial \text{net}_3}{\partial W_{hx}} + \frac{\partial \text{net}_3}{\partial net_2} \cdot \frac{\partial \text{net}_2}{\partial W_{hx}} + \frac{\partial \text{net}_3}{\partial net_2} \cdot \frac{\partial \text{net}_2}{\partial \text{net}_1} \cdot \frac{\partial \text{net}_1}{\partial W_{hx}} \right)$$



2.4.2.1 Forward recursion

- Sum up:

$$\frac{\partial L}{\partial W_{hx}} = \frac{1}{T}\frac{\partial L_1}{\partial \text{net}_1}\cdot\frac{\partial \text{net}_1}{\partial W_{hx}} + \frac{1}{T}\frac{\partial L_2}{\partial \text{net}_2}\cdot\left(\frac{\partial \text{net}_2}{\partial W_{hx}} + \frac{\partial \text{net}_2}{\partial net_1}\cdot\frac{\partial \text{net}_1}{\partial W_{hx}}\right) + \frac{1}{T}\frac{\partial L_3}{\partial \text{net}_3}\cdot\left(\frac{\partial \text{net}_3}{\partial W_{hx}} + \frac{\partial \text{net}_3}{\partial net_2}\cdot\frac{\partial \text{net}_2}{\partial W_{hx}} + \frac{\partial \text{net}_3}{\partial net_2}\cdot\frac{\partial \text{net}_2}{\partial \text{net}_1}\cdot\frac{\partial \text{net}_1}{\partial W_{hx}}\right).$$



2.4.2.1 Forward recursion

- The $\frac{\partial \text{net}_t}{\partial W_{hx}}$ component can be calculated recursively as follows:

$$\frac{d\text{net}_t}{dW_{hx}} = \underbrace{\frac{\partial \text{net}_t}{\partial W_{hx}}}_{\text{direct}} + \sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t} \frac{\partial \text{net}_j}{\partial \text{net}_{j-1}} \right) \frac{\partial \text{net}_i}{\partial W_{hx}}$$

$$= \frac{\partial \text{net}_t}{\partial W_{hx}} + \frac{\partial \text{net}_t}{\partial \text{net}_{t-1}} \cdot \left[\sum_{i=1}^{t-1} \left(\prod_{j=i+1}^{t-1} \frac{\partial \text{net}_j}{\partial \text{net}_{j-1}} \right) \frac{\partial \text{net}_i}{\partial W_{hx}} \right]$$

$$= \frac{\partial \text{net}_t}{\partial W_{hx}} + \frac{\partial \text{net}_t}{\partial \text{net}_{t-1}} \cdot \left(\underbrace{\frac{\partial \text{net}_{t-1}}{\partial W_{hx}}}_{\text{direct at } t-1} + \sum_{i=1}^{t-2} \left(\prod_{j=i+1}^{t-1} \frac{\partial \text{net}_j}{\partial \text{net}_{j-1}} \right) \frac{\partial \text{net}_i}{\partial W_{hx}} \right)$$

$$= \frac{\partial \text{net}_t}{\partial W_{hx}} + \frac{\partial \text{net}_t}{\partial \text{net}_{t-1}} \cdot \underbrace{\frac{d\text{net}_{t-1}}{dW_{hx}}}_{\text{Total derivative at } t-1}$$

2.4.2.1 Forward recursion

we can also calculate for $\frac{\partial L}{\partial W_{hh}}$ and $\frac{\partial L}{\partial b_h}$

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L_t}{\partial \text{net}_t}\left(\frac{\partial \text{net}_t}{\partial W_{hh}} + \sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t}\frac{\partial \text{net}_j}{\partial \text{net}_{j-1}}\right)\cdot\frac{\partial \text{net}_i}{\partial W_{hh}}\right)$$

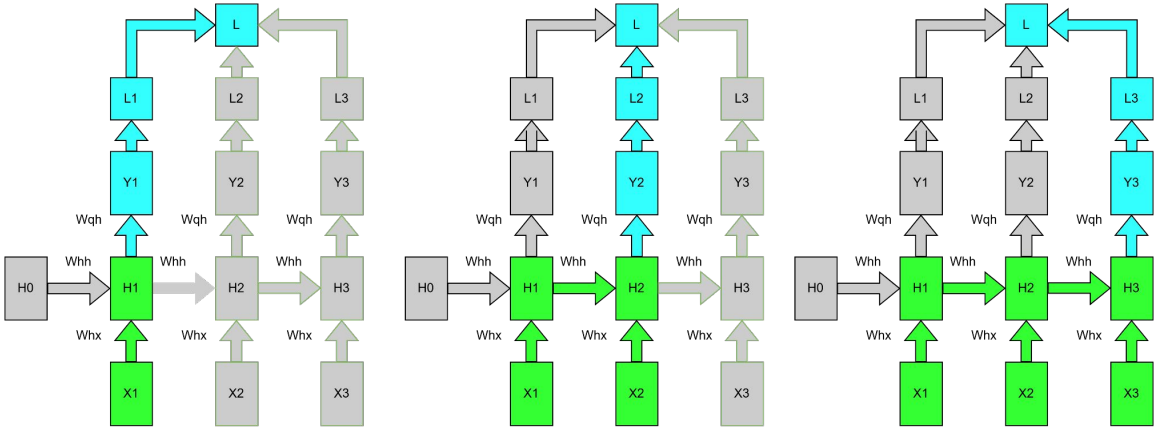
$$\frac{\partial L}{\partial b_h} = \frac{1}{T}\sum_{t=1}^{T}\frac{\partial L_t}{\partial \text{net}_t}\left(\frac{\partial \text{net}_t}{\partial b_h} + \sum_{i=1}^{t-1}\left(\prod_{j=i+1}^{t}\frac{\partial \text{net}_j}{\partial \text{net}_{j-1}}\right)\cdot\frac{\partial \text{net}_i}{\partial b_h}\right)$$

it can be seen that at any time step t, we always have to maintain recursion over three different quantities, namely $\frac{d\text{net}_t}{dW_{hx}}$, $\frac{d\text{net}_t}{dW_{hz}}$ and $\frac{d\text{net}_t}{dW_{bh}}$

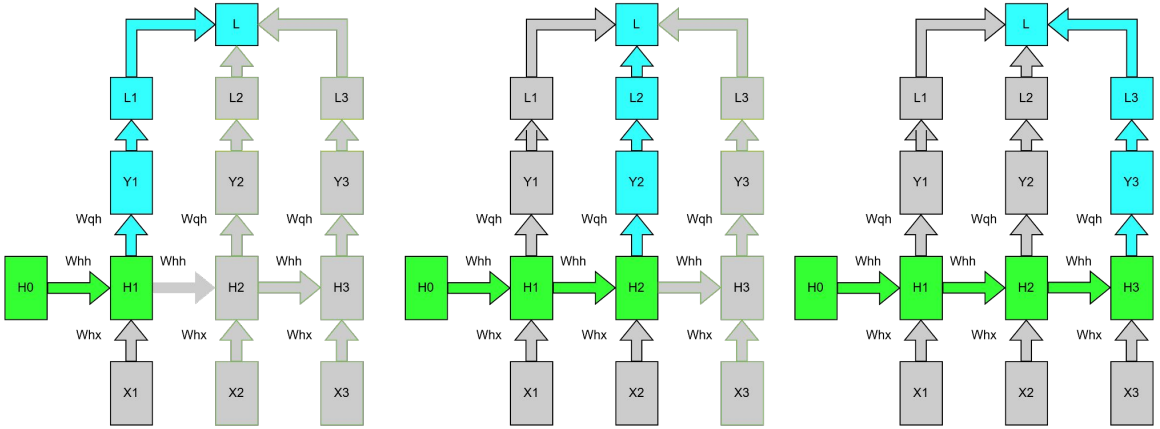
Keeping these recursive components, they do not share much information with each other and cannot reuse any parameters.

2.4.2.1 Forward recursion

$$\frac{\partial L}{\partial W_{hx}}$$



$$\frac{\partial L}{\partial W_{hh}}$$



2.4.2.2 Backward recursion

$$\frac{\partial L}{\partial W_{hx}} = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{\partial L}{\partial net_t}\right)\left(\frac{\partial net_t}{\partial W_{hx}}\right)$$

$$= \frac{1}{T}\sum_{t=1}^{T}\sum_{i=t}^{T}\left(\frac{\partial L_i}{\partial net_t}\right)\left(\frac{\partial net_t}{\partial W_{hx}}\right)$$

$$= \frac{1}{T}\sum_{t=1}^{T}\left[\frac{\partial L_t}{\partial net_t} + \sum_{i=t+1}^{T}\left(\left(\prod_{j=t}^{i-1}\frac{\partial net_{j+1}}{\partial net_j}\right)^{T}\frac{\partial L_i}{\partial net_i}\right)\right]\left(\frac{\partial net_t}{\partial W_{hx}}\right).$$

The main idea for the expansion presented below is to reverse the question compared to forward recursion. Instead of asking, at each time step, the current loss is affected by which states in the present and past. (L_t is affected by h_i , with i from t down to 1), we query in the opposite direction: at each time step t, the current state can influence which losses, namely the current h_t affecting current and all future L_i (with i running from t to T).

2.4.2.2 Backward recursion

The main idea for the expansion presented below is to reverse the question compared to forward recursion. Instead of asking, at each time step, which current and past states affect L_t (L_t is affected by h_i , with i from t down to 1), we query in the opposite direction: at each time step t, which components of the Loss can the current state influence, namely the current h_t affecting current and all future L_i (with i running from t to T).

$$\begin{aligned}\frac{\partial L}{\partial W_{hx}} &= \frac{1}{T}\sum_{t=1}^{T}\left(\frac{\partial L}{\partial net_t}\right)\left(\frac{\partial net_t}{\partial W_{hx}}\right) \\ &= \frac{1}{T}\sum_{t=1}^{T}\sum_{i=t}^{T}\left(\frac{\partial L_i}{\partial net_t}\right)\left(\frac{\partial net_t}{\partial W_{hx}}\right) \\ &= \frac{1}{T}\sum_{t=1}^{T}\left[\frac{\partial L_t}{\partial net_t} + \sum_{i=t+1}^{T}\left(\left(\prod_{j=t}^{i-1}\frac{\partial net_{j+1}}{\partial net_j}\right)^{T}\frac{\partial L_i}{\partial net_i}\right)\right]\left(\frac{\partial net_t}{\partial W_{hx}}\right). \end{aligned} \tag{16}$$

Explanation of the formula:

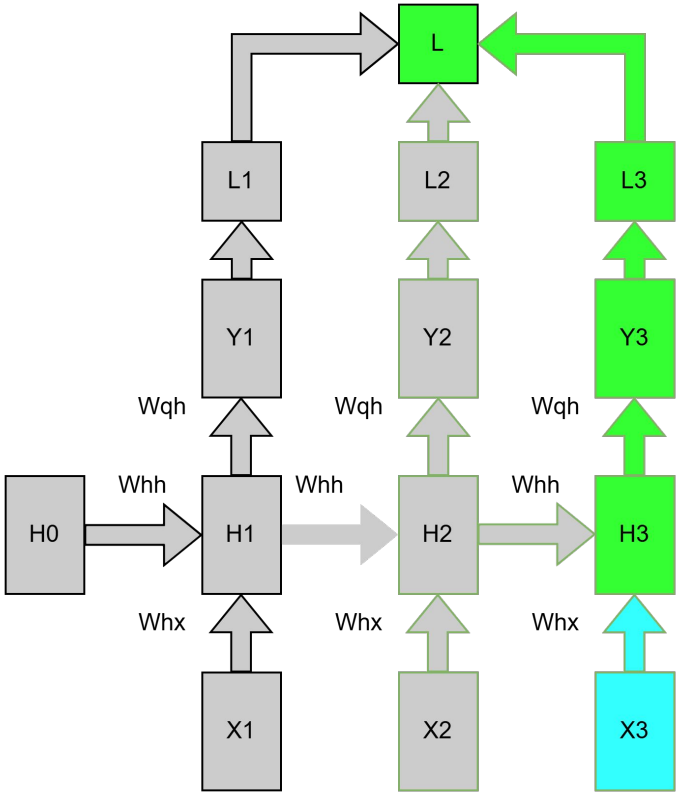
- First equality: At each time step t, we expand the full-time derivative of the Loss with respect to the current net net_t, and then multiply it by the local derivative over time between net_t and the weight W_{hx}.
- Next equality: At each time step t, net_t can only influence the current Loss and future Losses, which explains why i runs from t to T.
- Final equality: This expands the full-time chain rule between L_i and net_t, in other words, and more explicitly L_i affects net_t through a chain $(\frac{\partial net_i}{\partial net_{i-1}}), (\frac{\partial net_{i-1}}{\partial net_{i-2}}), \ldots, (\frac{\partial net_{t+1}}{\partial net_t})$.

2.4.2.2 Backward recursion

Similar to forward recursion, we expand the example:

With (t = 3):

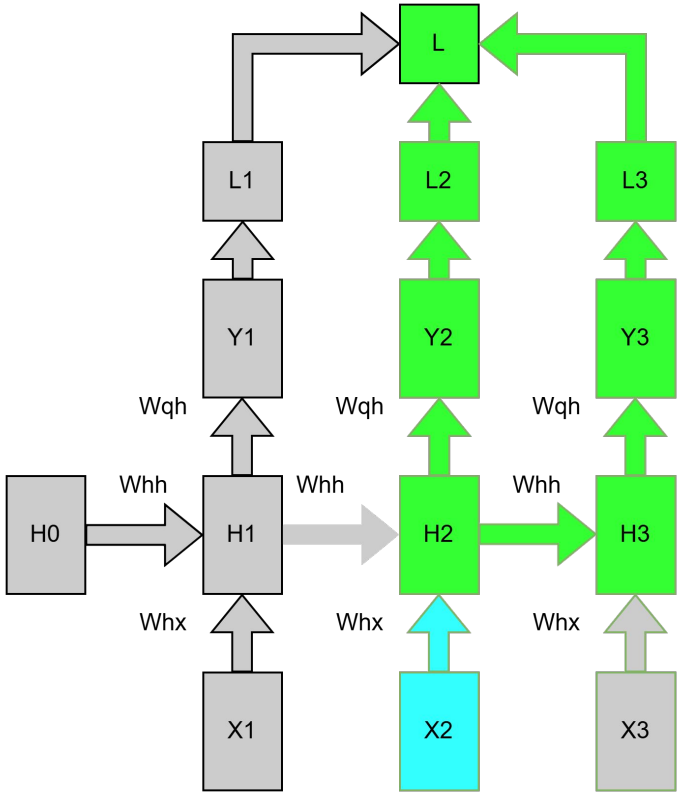
$$\frac{1}{T}\frac{\partial L}{\partial net_3}\frac{\partial net_3}{\partial W_{hx}} = \frac{1}{T}\frac{\partial L_3}{\partial net_3}\frac{\partial net_3}{\partial W_{hx}}$$



2.4.2.2 Backward recursion

With (t = 2):

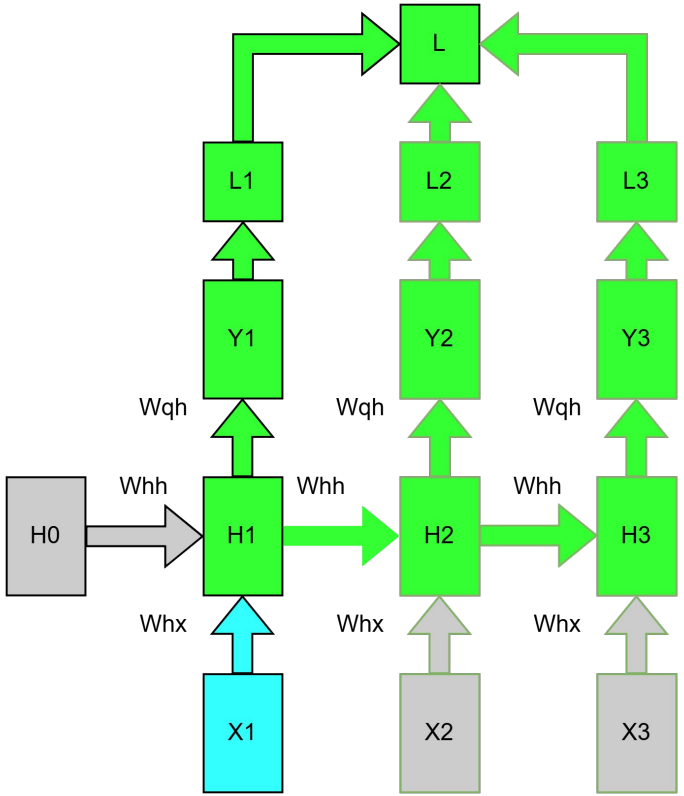
$$\frac{1}{T}\frac{\partial L}{\partial net_2}\frac{\partial net_2}{\partial W_{hx}} = \frac{1}{T}\left(\frac{\partial L_2}{\partial net_2} + \left(\frac{\partial net_3}{\partial net_2}\right)^T \frac{\partial L_3}{\partial net_3}\right)\frac{\partial net_2}{\partial W_{hx}}.$$



2.4.2.2 Backward recursion

With (t = 1):

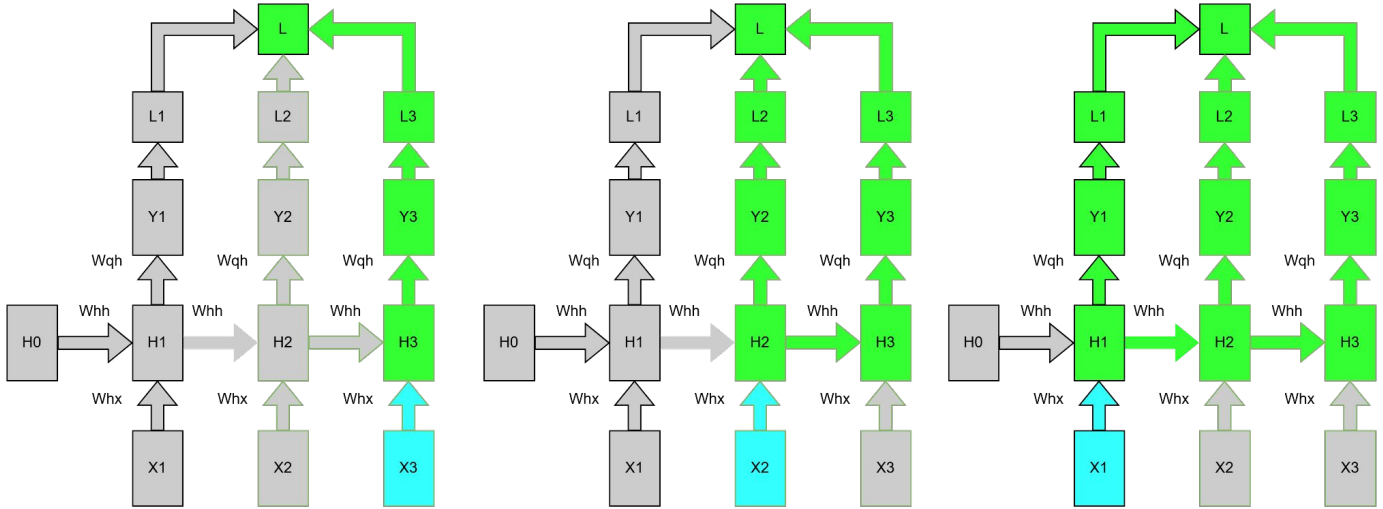
$$\frac{1}{T}\frac{\partial L}{\partial W_{hx}}\bigg|_{t=1} = \frac{1}{T}\left(\frac{\partial L_1}{\partial net_1} + \left(\frac{\partial net_2}{\partial net_1}\right)^T \frac{\partial L_2}{\partial net_2} + \left(\frac{\partial net_3}{\partial net_2}\frac{\partial net_2}{\partial net_1}\right)^T \frac{\partial L_3}{\partial net_3}\right)\frac{\partial net_1}{\partial W_{hx}}.$$



2.4.2.2 Backward recursion

So the total gradient for W_{hx} in the above example will be:

$$\frac{\partial L}{\partial W_{hx}} = \frac{1}{T}\frac{\partial L_3}{\partial net_3}\frac{\partial net_3}{\partial W_{hx}} + \frac{1}{T}\left(\frac{\partial L_2}{\partial net_2} + \left(\frac{\partial net_3}{\partial net_2}\right)^T \frac{\partial L_3}{\partial net_3}\right)\frac{\partial net_2}{\partial W_{hx}} + \frac{1}{T}\left(\frac{\partial L_1}{\partial net_1} + \left(\frac{\partial net_2}{\partial net_1}\right)^T \frac{\partial L_2}{\partial net_2} + \left(\frac{\partial net_3}{\partial net_2}\frac{\partial net_2}{\partial net_1}\right)^T \frac{\partial L_3}{\partial net_3}\right)\frac{\partial net_1}{\partial W_{hx}}$$



2.4.2.2 Backward recursion

So in general, the above calculation method gives the same results as the forward recursion method above.

Backward recursion

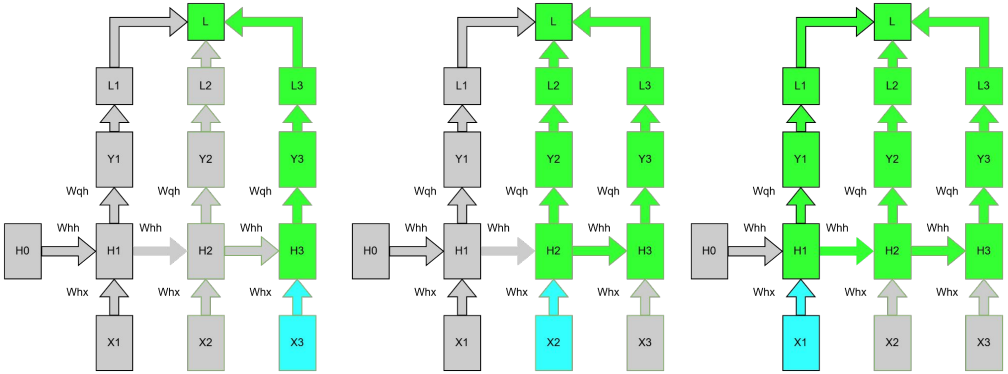
$$\begin{aligned}\frac{\partial L}{\partial W_{hx}} &= \frac{1}{T}\frac{\partial L_3}{\partial net_3}\frac{\partial net_3}{\partial W_{hx}} \\ &+ \frac{1}{T}\left(\frac{\partial L_2}{\partial net_2} + \left(\frac{\partial net_3}{\partial net_2}\right)^T \frac{\partial L_3}{\partial net_3}\right)\frac{\partial net_2}{\partial W_{hx}} \\ &+ \frac{1}{T}\left(\frac{\partial L_1}{\partial net_1} + \left(\frac{\partial net_2}{\partial net_1}\right)^T \frac{\partial L_2}{\partial net_2} + \left(\frac{\partial net_3}{\partial net_2}\frac{\partial net_2}{\partial net_1}\right)^T \frac{\partial L_3}{\partial net_3}\right)\frac{\partial net_1}{\partial W_{hx}}\end{aligned}$$

Forward recursion

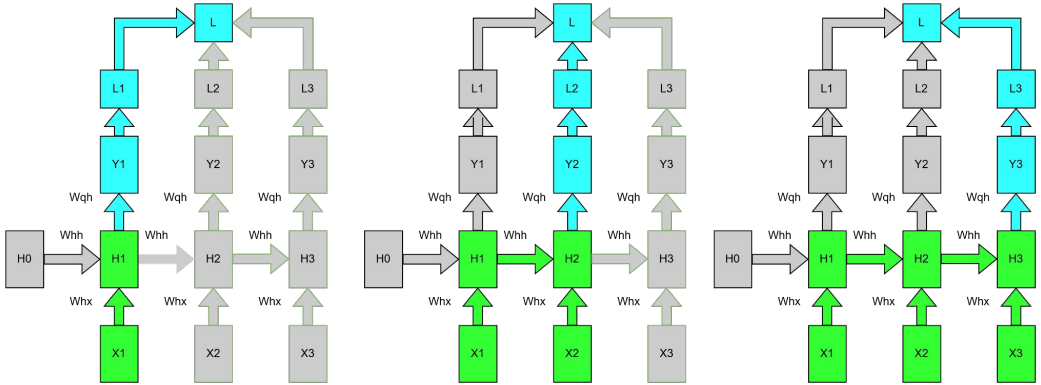
$$\begin{aligned}\frac{\partial L}{\partial W_{hx}} &= \frac{1}{T}\frac{\partial L_1}{\partial \mathrm{net}_1}\cdot\frac{\partial \mathrm{net}_1}{\partial W_{hx}} \\ &+ \frac{1}{T}\frac{\partial L_2}{\partial \mathrm{net}_2}\cdot\left(\frac{\partial \mathrm{net}_2}{\partial W_{hx}} + \frac{\partial \mathrm{net}_2}{\partial net_1}\cdot\frac{\partial \mathrm{net}_1}{\partial W_{hx}}\right) \\ &+ \frac{1}{T}\frac{\partial L_3}{\partial \mathrm{net}_3}\cdot\left(\frac{\partial \mathrm{net}_3}{\partial W_{hx}} + \frac{\partial \mathrm{net}_3}{\partial net_2}\cdot\frac{\partial \mathrm{net}_2}{\partial W_{hx}} + \frac{\partial \mathrm{net}_3}{\partial net_2}\cdot\frac{\partial \mathrm{net}_2}{\partial \mathrm{net}_1}\cdot\frac{\partial \mathrm{net}_1}{\partial W_{hx}}\right).\end{aligned}$$

2.4.2.2 Backward recursion

Backward recursion



Forward recursion



2.4.2.2 Backward recursion

$\frac{\partial L}{\partial net_t}$ can be calculated recursively. Analyzing this derivative carefully, we can see:

$$
\begin{aligned}
\frac{\partial L}{\partial net_t} &= \frac{\partial L_t}{\partial net_t} + \sum_{i=t+1}^{T} \frac{\partial L_i}{\partial net_t} \\
&= \frac{\partial L_t}{\partial net_t} + \sum_{i=t+1}^{T} \left(\left(\prod_{j=t}^{i-1} \frac{\partial net_{j+1}}{\partial net_j} \right)^T \frac{\partial L_i}{\partial net_i} \right) \qquad \text{(each term } h \times 1\text{)} \\
&= \frac{\partial L_t}{\partial net_t} + \left(\sum_{i=t+1}^{T} \left(\prod_{j=t+1}^{i-1} \frac{\partial net_{j+1}}{\partial net_j} \right)^T \frac{\partial L_i}{\partial net_i} \right) \left(\frac{\partial net_{t+1}}{\partial net_t} \right)^T \\
&= \frac{\partial L_t}{\partial net_t} + \left(\frac{\partial net_{t+1}}{\partial net_t} \right)^T \left(\sum_{i=t+1}^{T} \left(\prod_{j=t+1}^{i-1} \frac{\partial net_{j+1}}{\partial net_j} \right)^T \frac{\partial L_i}{\partial net_i} \right) \\
&= \frac{\partial L_t}{\partial net_t} + \left(\frac{\partial net_{t+1}}{\partial net_t} \right)^T \frac{\partial L}{\partial net_{t+1}} \qquad \text{(since the innter sum equals } \frac{\partial L}{\partial net_{t+1}}\text{)} \\
&= \left(\frac{\partial h_t}{\partial net_t} \right)^T \frac{\partial L_t}{\partial h_t} + \left(\frac{\partial net_{t+1}}{\partial net_t} \right)^T \frac{\partial L}{\partial net_{t+1}} \qquad \left(\frac{\partial h_t}{\partial net_t} : h \times h \right) \\
&= \left(\frac{\partial h_t}{\partial net_t} \right)^T \left(\frac{\partial L_t}{\partial h_t} + \left(\frac{\partial net_{t+1}}{\partial h_t} \right)^T \frac{\partial L}{\partial net_{t+1}} \right).
\end{aligned}
$$

2.4.2.2 Backward recursion

Expanding the chain rule similarly for W_{hh} and W_{bh}, we obtain the following formula:

$$\frac{\partial L}{\partial W_{hh}} = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{\partial L}{\partial net_t}\right)\left(\frac{\partial net_t}{\partial W_{hh}}\right)$$

$$\frac{\partial L}{\partial W_{bh}} = \frac{1}{T}\sum_{t=1}^{T}\left(\frac{\partial L}{\partial net_t}\right)\left(\frac{\partial net_t}{\partial W_{bh}}\right)$$

$$\delta_t := \frac{\partial L}{\partial net_t}.$$

$$\boxed{\delta_t = \left(\frac{\partial h_t}{\partial net_t}\right)^T \left(\frac{1}{T}\frac{\partial L_t}{\partial h_t} + \left(\frac{\partial net_{t+1}}{\partial h_t}\right)^T \delta_{t+1}\right)}$$

$$\boxed{\frac{\partial L}{\partial W_{hx}} = \frac{1}{T}\sum_{t=1}^{T}\delta_t\, x_t^T}$$

$$\boxed{\frac{\partial L}{\partial W_{hh}} = \frac{1}{T}\sum_{t=1}^{T}\delta_t\, h_{t-1}^T}$$

$$\boxed{\frac{\partial L}{\partial W_{bh}} = \frac{1}{T}\sum_{t=1}^{T}\delta_t}$$

2.4.2.2 Backward recursion

Calculate $(\frac{\partial L}{\partial \mathbf{b}_h})$, $(\frac{\partial L}{\partial \mathbf{W}_{hh}})$, $(\frac{\partial L}{\partial \mathbf{W}_{hx}})$

At t = 3

$$\frac{\partial L}{\partial \mathbf{net}_3} = \frac{\partial L_3}{\partial \mathbf{net}_3} = \frac{\partial L_3}{\partial \mathbf{o}_3}\frac{\partial o_3}{\partial \mathbf{h}_3}\frac{\partial h_3}{\partial \mathbf{net}_3} = (W_{qh}^{\top}(\hat{y}_3 - y_3))\phi'(net_3)$$

$$\frac{\partial L_3}{\partial \mathbf{o}_3} = (\hat{y}_3 - y_3) = \begin{bmatrix} 0.2614 \\ 0.2547 \\ 0.2207 \\ 0.2632 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = [0.2614, 0.2547, 0.2207, -0.7368]^{\top}.$$

$$W_{qh}^{\top}\frac{\partial L_3}{\partial \mathbf{o}_3} = \begin{bmatrix} 0.3 & -0.2 & 0.1 & 0.2 \\ 0.1 & 0.4 & -0.3 & 0.2 \end{bmatrix} \begin{bmatrix} 0.2614 \\ 0.2547 \\ 0.2207 \\ -0.7368 \end{bmatrix} = \begin{bmatrix} -0.09781 \\ -0.08555 \end{bmatrix}.$$

2.4.2.2 Backward recursion

$$1-\tanh(a_3)^2 = 1-h_3^2 \approx \begin{bmatrix} 1-0.1850^2 \\ 1-0.3553^2 \end{bmatrix} \approx [0.9658, 0.8738]^\top.$$

$$\delta^3 = (W_{qh}^\top \frac{\partial L_3}{\partial \mathbf{o}_3} + W_{qh}^\top \delta^4) \odot (1-\tanh(a_3)^2) = \begin{bmatrix} -0.09781 \\ -0.08555 \end{bmatrix} \odot \begin{bmatrix} 0.9658 \\ 0.8738 \end{bmatrix} \approx \begin{bmatrix} -0.09446 \\ -0.07475 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{net}_3}\frac{\partial net_3}{\partial \mathbf{W}_{hh}} = \delta^3 . {h_2}^\top = \begin{bmatrix} -0.09446 \\ -0.07475 \end{bmatrix} \begin{bmatrix} -0.2385 & 0.3050 \end{bmatrix} = \begin{bmatrix} 0.022538 & -0.028810 \\ 0.017835 & -0.022799 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{net}_3}\frac{\partial net_3}{\partial \mathbf{W}_{hx}} = \delta^3 . {x_3}^\top = \begin{bmatrix} -0.09446 \\ -0.07475 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -0.09446 & 0 \\ 0 & 0 & -0.07475 & 0 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{net}_3}\frac{\partial net_3}{\partial \mathbf{b}_h} = \delta^3 = \begin{bmatrix} -0.09446 \\ -0.07475 \end{bmatrix}.$$

2.4.2.2 Backward recursion

At t = 2

$$\frac{\partial L}{\partial \mathbf{net}_2} = (\frac{\partial L_2}{\partial \mathbf{h}_2} + (W_{hh}^\top)\frac{\partial L}{\partial \mathbf{net}_3})\frac{\partial h_2}{\partial \mathbf{net}_2} = (\frac{\partial L_2}{\partial \mathbf{o}_2}\frac{\partial o_2}{\partial \mathbf{h}_2} + (W_{hh}^\top)\frac{\partial L}{\partial \mathbf{net}_3}) \odot \phi'(net_2)$$

$$= (W_{qh}^\top(\hat{y}_2 - y_2) + (W_{hh}^\top)(\frac{\partial L}{\partial \mathbf{net}_3}) \odot \phi'(net_2)$$

$$\frac{\partial L_2}{\partial \mathbf{o}_2} = (\hat{y}_2 - y_2) = \begin{bmatrix} 0.2398 \\ 0.2845 \\ 0.2249 \\ 0.2507 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = [0.2398, 0.2845, -0.7751, 0.2507]^\top.$$

$$W_{qh}^\top \frac{\partial L_2}{\partial \mathbf{o}_2} = \begin{bmatrix} 0.3 & -0.2 & 0.1 & 0.2 \\ 0.1 & 0.4 & -0.3 & 0.2 \end{bmatrix} \begin{bmatrix} 0.2398 \\ 0.2845 \\ -0.7751 \\ 0.2507 \end{bmatrix} = \begin{bmatrix} -0.01233 \\ 0.42045 \end{bmatrix}.$$

$$W_{hh}^\top \delta^3 = \begin{bmatrix} 0.1 & 0.0 \\ 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} -0.09446 \\ -0.07475 \end{bmatrix} = \begin{bmatrix} -0.009446 \\ -0.041317 \end{bmatrix}$$

2.4.2.2 Backward recursion

$$1-\tanh(a_2)^2 = 1-h_2^2 \approx \begin{bmatrix} 1+0,2386^2 \\ 1-0.3050^2 \end{bmatrix} \approx [0.9431, 0.9070]^\top.$$

$$\delta^2 = (W_{qh}^\top \frac{\partial L_2}{\partial \mathbf{o}_2} + W_{hh}^\top \delta^3) \odot (1-tanh(a_2)^2) = (\begin{bmatrix} -0.01233 \\ 0.42045 \end{bmatrix} + \begin{bmatrix} -0.009446 \\ -0.041317 \end{bmatrix}) \odot \begin{bmatrix} 0.9431 \\ 0.9070 \end{bmatrix} \approx \begin{bmatrix} -0.021776 \\ 0.379133 \end{bmatrix}$$

$$\odot \begin{bmatrix} 0.9431 \\ 0.9070 \end{bmatrix} \approx \begin{bmatrix} -0.0205 \\ 0.3439 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{net}_2} \frac{\partial net_2}{\partial \mathbf{W}_{hh}} = \delta^2 . {h_1}^\top \begin{bmatrix} -0.0205 \\ 0.3439 \end{bmatrix} [0.5005 \quad -0.2165] = \begin{bmatrix} -0.010260 & 0.004438 \\ 0.172122 & -0.074454 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{net}_2} \frac{\partial net_2}{\partial \mathbf{W}_{hx}} = \delta^2 . {x_2}^\top = \begin{bmatrix} -0.0205 \\ 0.3439 \end{bmatrix} [0 \quad 1 \quad 0 \quad 0] = \begin{bmatrix} 0 & -0.0205 & 0 & 0 \\ 0 & 0.3439 & 0 & 0 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{net}_2} \frac{\partial net_2}{\partial \mathbf{b}_h} = \delta^2 = \begin{bmatrix} -0.0205 \\ 0.3439 \end{bmatrix}.$$

2.4.2.2 Backward recursion

At t = 1

$$\frac{\partial L_1}{\partial \mathbf{net}_1} = (\frac{\partial L_1}{\partial \mathbf{h}_1} + (W_{hh}^\top)\frac{\partial L}{\partial \mathbf{net}_2})\frac{\partial h_1}{\partial \mathbf{net}_1}$$

$$= (\frac{\partial L_1}{\partial \mathbf{o}_1} + (W_{hh}^\top(\frac{\partial L}{\partial \mathbf{net}_2}) \odot \phi'(net_1)) = (W_{qh}^\top(\hat{y}_1 - y_1) + (W_{hh}^\top)(\frac{\partial L}{\partial \mathbf{net}_2}) \odot \phi'(net_1)$$

$$\frac{\partial L_1}{\partial \mathbf{o}_1} = (\hat{y}_1 - y_1) = \begin{bmatrix} 0.2763 \\ 0.1937 \\ 0.2754 \\ 0.2546 \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = [0.2763, -0.8063, 0.2754, 0.2546]^\top.$$

$$W_{qh}^\top \frac{\partial L_2}{\partial \mathbf{o}_2} = \begin{bmatrix} 0.3 & -0.2 & 0.1 & 0.2 \\ 0.1 & 0.4 & -0.3 & 0.2 \end{bmatrix} \begin{bmatrix} 0.2763 \\ -0.8063 \\ 0.2754 \\ 0.2546 \end{bmatrix} = \begin{bmatrix} 0.32261 \\ -0.32659 \end{bmatrix}.$$

$$W_{hh}^\top \delta^2 = \begin{bmatrix} 0.1 & 0.0 \\ 0.2 & 0.3 \end{bmatrix} \begin{bmatrix} -0.0205 \\ 0.3439 \end{bmatrix} = \begin{bmatrix} -0.00205 \\ 0.09907 \end{bmatrix}.$$

2.4.2.2 Backward recursion

$$1-\tanh(a_1)^2 = 1-h_1^2 \approx \begin{bmatrix} 1-0.5005^2 \\ 1+0.2165^2 \end{bmatrix} \approx [0.7495, 0.9531]^\top.$$

$$\delta^1 = (W_{qh}^\top \frac{\partial L_1}{\partial \mathbf{o}_1} + W_{hh}^\top \delta^2) \odot (1-tanh(a_1)^2) = (\begin{bmatrix} 0.32261 \\ -0.32659 \end{bmatrix} + \begin{bmatrix} -0.00205 \\ 0.09907 \end{bmatrix}) \odot [0.7495, 0.9531] \approx \begin{bmatrix} 0.32056 \\ -0.22752 \end{bmatrix} \odot \begin{bmatrix} 0.7495 \\ 0.9531 \end{bmatrix} \approx \begin{bmatrix} 0.2403 \\ -0.2168 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{net}_1} \frac{\partial net_1}{\partial \mathbf{W}_{hh}} = \delta^1 . {h_0}^\top = \begin{bmatrix} 0.2403 \\ -0.2168 \end{bmatrix} [0 \quad 0] = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{net}_1} \frac{\partial net_1}{\partial \mathbf{W}_{hx}} = \delta^1 . {x_1}^\top = \begin{bmatrix} 0.2403 \\ -0.2168 \end{bmatrix} [1 \quad 0 \quad 0 \quad 0] = \begin{bmatrix} 0.2403 & 0 & 0 & 0 \\ -0.2168 & 0 & 0 & 0 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{net}_1} \frac{\partial net_1}{\partial \mathbf{b}_h} = \delta^1 = \begin{bmatrix} 0.2403 \\ -0.2168 \end{bmatrix}.$$

2.4.2.2 Backward recursion

$$\frac{\partial L}{\partial \mathbf{W}_{hx}} = \frac{1}{3}\left(\frac{\partial L}{\partial \mathbf{net}_3} X_3^\top + \frac{\partial L}{\partial \mathbf{net}_2} X_2^\top + \frac{\partial L}{\partial \mathbf{net}_1} X_1^\top\right) = \frac{1}{3}\left(\begin{bmatrix} 0 & 0 & -0.09446 & 0 \\ 0 & 0 & -0.07475 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -0.0205 & 0 & 0 \\ 0 & 0.3439 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0.2403 & 0 & 0 & 0 \\ -0.2168 & 0 & 0 & 0 \end{bmatrix}\right) = \frac{1}{3}\begin{bmatrix} 0.2403 & -0.0205 & -0.09446 & 0 \\ -0.2168 & 0.3439 & -0.07475 & 0 \end{bmatrix} \approx \begin{bmatrix} 0.0801 & -0.00683 & -0.0315 & 0 \\ -0.0723 & 0.1146 & -0.0249 & 0 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{W}_{hh}} = \frac{1}{3}\left(\frac{\partial L}{\partial \mathbf{net}_3} h_2^\top + \frac{\partial L}{\partial \mathbf{net}_2} h_1^\top + \frac{\partial L}{\partial \mathbf{net}_1} h_0^\top\right) = \frac{1}{3}\left(\begin{bmatrix} 0.022538 & -0.028810 \\ 0.017835 & -0.022799 \end{bmatrix} + \begin{bmatrix} -0.010260 & 0.004438 \\ 0.172122 & -0.074454 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\right) \approx \begin{bmatrix} 0.0040927 & -0.008124 \\ 0.06332 & -0.03242 \end{bmatrix}.$$

$$\frac{\partial L}{\partial \mathbf{b}_h} = \frac{1}{3}\left(\frac{\partial L}{\partial \mathbf{net}_3} + \frac{\partial L}{\partial \mathbf{net}_2} + \frac{\partial L}{\partial \mathbf{net}_1} = \frac{1}{3}\left(\begin{bmatrix} -0.09446 \\ -0.07475 \end{bmatrix} + \begin{bmatrix} -0.0205 \\ 0.3439 \end{bmatrix} + \begin{bmatrix} 0.2403 \\ -0.2168 \end{bmatrix}\right) = \begin{bmatrix} 0.04178 \\ 0.01745 \end{bmatrix}.$$

2.4.3 Vanishing/Exploding Gradient Analysis

Restating: in backward recursion, the term $\frac{\partial L}{\partial \text{net}_t}$ is used to perform the global recursive computation over time from $T \to 1$.

This term can be expanded into:

$$\frac{\partial L}{\partial \text{net}_t} = \frac{\partial L_t}{\partial \text{net}_t} + \sum_{i=t+1}^{T} \frac{\partial L_i}{\partial \text{net}_i} \left(\prod_{j=t}^{i-1} \frac{\partial \text{net}_{j+1}}{\partial \text{net}_j} \right)$$

$$= \frac{\partial L_t}{\partial \text{net}_t} + \sum_{i=t+1}^{T} \left(\prod_{j=t}^{i-1} \frac{\partial \text{net}_{j+1}}{\partial \text{net}_j} \right)^{\top} \frac{\partial L_i}{\partial \text{net}_i}$$

$$= \frac{\partial L_t}{\partial \text{net}_t} + \sum_{i=t+1}^{T} \left(\prod_{j=t}^{i-1} \frac{\partial \text{net}_{j+1}}{\partial h_j} \frac{\partial h_j}{\partial \text{net}_j} \right)^{\top} \frac{\partial L_i}{\partial \text{net}_i}$$

$$= \frac{\partial L_t}{\partial \text{net}_t} + \sum_{i=t+1}^{T} \left(\prod_{j=t}^{i-1} W_{hh} \right)^{\top} \left(\prod_{j=t}^{i-1} \varphi'(\text{net}_j) \right)^{\top} \frac{\partial L_i}{\partial \text{net}_i}$$

$$= \frac{\partial L_t}{\partial \text{net}_t} + \sum_{i=t+1}^{T} (W_{hh}^{i-t})^{\top} \left(\prod_{j=t}^{i-1} \varphi'(\text{net}_j) \right)^{\top} \frac{\partial L_i}{\partial \text{net}_i}$$

Exercise 1

Let $\mathbf{M} \in \mathbb{R}^{n \times n}$ be a **symmetric** matrix with eigenvalues λ_i and corresponding eigenvectors $\mathbf{v}_i$, ordered as $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.

1.1

Show that $\mathbf{M}^k$ has eigenvalues λ_i^k.

1.2

Show that for a random vector $\mathbf{x}$, with high probability, $\mathbf{M}^k \mathbf{x}$ becomes almost aligned with the principal eigenvector $\mathbf{v}_1$ as $k \to \infty$.

1.3

What does the above result imply for **gradients in RNNs**?

Exercise 2

Besides **gradient clipping**, propose **other methods** to cope with **exploding gradients** in recurrent neural networks.

2.5 Truncation

In RNN, processing very long sequences (e.g., thousands of time steps) presents two critical challenges: memory exhaustion (storing the entire computation graph) and numerical instability (vanishing or exploding gradients due to repeated matrix multiplication).

To address this, standard implementations utilize Truncated Backpropagation Through Time (TBPTT). This technique decouples the forward pass from the backward pass dynamics.

1. The Mechanism
    - Forward Pass (Continuous): The network processes the data in smaller chunks (e.g., window size k_1). Crucially, the final hidden state of the previous chunk is carried over as the initial state of the next chunk. This ensures the model retains "memory" of the entire sequence context.
    - Backward Pass (Truncated): During backpropagation, the gradient flow is explicitly cut off after a fixed number of steps (k_2). We "detach" the history, effectively assuming that the gradient contribution from time steps prior to the current window is zero.
2. Mathematical Implication
    - Idea: Do not backprop for entire time,but only backprop for a small time of the k most recent steps.
    - Instead of calculating:

$$\frac{\partial L}{\partial W_{hx}} = \sum_{t=1}^{T} \frac{\partial L_t}{\partial W}$$

    - We calculate:

$$\frac{\partial L}{\partial W_{hx}} = \sum_{t=T-k+1}^{T} \frac{\partial L_t}{\partial W}$$

2.5 Truncation

3. Two common types of truncation
    - Fixed Truncation

    We choose fixed k.

    For example: k=20 -> We backdrop for 20 steps only.

    - Randomized Truncation

    k is not fixed but random for each batch.

4. The Trade-off

While truncation makes training computationally feasible and stabilizes gradients, it introduces a limitation: the model loses the ability to learn long-term dependencies that exceed the truncation window length.

THANK YOU FOR LISTENING

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

BK TP.HCM