Deep Learning for Sequential Data: Mathematical Foundations of RNNs and LSTMs

Week 8: FinTech Course
Mathematical Version

November 5, 2025

Abstract

This document provides a rigorous mathematical treatment of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks. We derive the forward propagation equations, backpropagation through time (BPTT) algorithm, gradient flow analysis, and the architectural innovations that enable LSTMs to capture long-term dependencies in sequential data. Applications to financial time series forecasting are discussed with emphasis on theoretical guarantees and convergence properties.

Contents

1 Introduction to Sequential Data

1.1 Mathematical Framework

Consider a sequence of observations $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(T)}$ where each $\mathbf{x}^{(t)} \in \mathbb{R}^{d_x}$ represents a d_x-dimensional input at time t. In financial applications, this could represent:

- Price returns: $r_t = \frac{P_t - P_{t-1}}{P_{t-1}}$
- Trading volumes: $V_t \in \mathbb{R}^+$
- Technical indicators: $\mathbf{I}_t \in \mathbb{R}^k$

Definition 1.1 (Time Series)**.** A time series is a stochastic process $\{X_t : t \in \mathcal{T}\}$ where $\mathcal{T} \subseteq \mathbb{Z}$ or $\mathcal{T} \subseteq \mathbb{R}$, and each X_t is a random variable.

1.2 Temporal Dependencies

The joint probability distribution over a sequence can be factorized as:

$$P(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(T)}) = \prod_{t=1}^{T} P(\mathbf{x}^{(t)} \mid \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t-1)}) \tag{1}$$

For a Markov model of order τ:

$$P(\mathbf{x}^{(t)} \mid \mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(t-1)}) = P(\mathbf{x}^{(t)} \mid \mathbf{x}^{(t-\tau)}, \ldots, \mathbf{x}^{(t-1)}) \tag{2}$$

2 Recurrent Neural Networks

2.1 Architecture and Forward Propagation

Definition 2.1 (Vanilla RNN)**.** A recurrent neural network maintains a hidden state $\mathbf{h}^{(t)} \in \mathbb{R}^{d_h}$ that is updated recurrently according to:

$$\mathbf{h}^{(t)} = \tanh(\mathbf{W}_{hh}\mathbf{h}^{(t-1)} + \mathbf{W}_{xh}\mathbf{x}^{(t)} + \mathbf{b}_h) \tag{3}$$

$$\mathbf{y}^{(t)} = \mathbf{W}_{hy}\mathbf{h}^{(t)} + \mathbf{b}_y \tag{4}$$

where:

- $\mathbf{W}_{hh} \in \mathbb{R}^{d_h \times d_h}$: hidden-to-hidden weight matrix
- $\mathbf{W}_{xh} \in \mathbb{R}^{d_h \times d_x}$: input-to-hidden weight matrix
- $\mathbf{W}_{hy} \in \mathbb{R}^{d_y \times d_h}$: hidden-to-output weight matrix
- $\mathbf{b}_h \in \mathbb{R}^{d_h}$, $\mathbf{b}_y \in \mathbb{R}^{d_y}$: bias vectors
- $\tanh(\cdot)$: hyperbolic tangent activation function

2.2 Activation Functions

The hyperbolic tangent function is defined as:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = \frac{e^{2z} - 1}{e^{2z} + 1} \tag{5}$$

Its derivative is:

$$\frac{d}{dz}\tanh(z) = 1 - \tanh^2(z) = \operatorname{sech}^2(z) \tag{6}$$

Key properties:

- Range: $\tanh(z) \in (-1, 1)$
- Zero-centered: $\tanh(0) = 0$
- Bounded gradient: $\frac{d}{dz}\tanh(z) \in (0, 1]$

2.3 Loss Function

For sequence-to-sequence prediction with target sequence $\{\mathbf{y}^{*(1)}, \ldots, \mathbf{y}^{*(T)}\}$, the loss function is:

$$\mathcal{L}(\boldsymbol{\theta}) = \sum_{t=1}^{T} L(\mathbf{y}^{(t)}, \mathbf{y}^{*(t)}) \tag{7}$$

For regression (financial forecasting):

$$L(\mathbf{y}^{(t)}, \mathbf{y}^{*(t)}) = \frac{1}{2}\|\mathbf{y}^{(t)} - \mathbf{y}^{*(t)}\|^2 = \frac{1}{2}\sum_{i=1}^{d_y}(y_i^{(t)} - y_i^{*(t)})^2 \tag{8}$$

2.4 Backpropagation Through Time (BPTT)

Theorem 2.1 (Gradient Flow in RNNs)**.** The gradient of the loss with respect to the hidden state at time k is given by:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(k)}} = \frac{\partial L^{(k)}}{\partial \mathbf{h}^{(k)}} + \sum_{t=k+1}^{T} \frac{\partial L^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \tag{9}$$

Proof. By the chain rule, the gradient flows backward through the computational graph:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(k)}} = \sum_{t=k}^{T} \frac{\partial L^{(t)}}{\partial \mathbf{h}^{(k)}} \tag{10}$$

$$= \frac{\partial L^{(k)}}{\partial \mathbf{h}^{(k)}} + \sum_{t=k+1}^{T} \frac{\partial L^{(t)}}{\partial \mathbf{h}^{(t)}} \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \tag{11}$$

The temporal derivative can be expanded as:

$$\frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} = \prod_{i=k+1}^{t} \frac{\partial \mathbf{h}^{(i)}}{\partial \mathbf{h}^{(i-1)}} = \prod_{i=k+1}^{t} \mathbf{W}_{hh}^{\top} \operatorname{diag}(1 - \tanh^2(\mathbf{h}^{(i)})) \tag{12}$$

□

2.5 The Vanishing Gradient Problem

Theorem 2.2 (Gradient Decay)**.** For an RNN with weight matrix $\mathbf{W}_{hh}$, the gradient magnitude decays exponentially with the temporal distance $t - k$:

$$\left\| \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \right\| \leq \|\mathbf{W}_{hh}\|^{t-k} \prod_{i=k+1}^{t} \|\text{diag}(1 - \tanh^2(\mathbf{h}^{(i)}))\| \tag{13}$$

Since $|\tanh'(z)| \leq 1$, if $\|\mathbf{W}_{hh}\| < 1$:

$$\left\| \frac{\partial \mathbf{h}^{(t)}}{\partial \mathbf{h}^{(k)}} \right\| \leq \gamma^{t-k}, \quad \gamma < 1 \tag{14}$$

This exponential decay makes it difficult to capture long-term dependencies.

Proposition 2.1 (Spectral Radius Condition)**.** If $\rho(\mathbf{W}_{hh}) < 1$ where ρ denotes the spectral radius, gradients vanish exponentially. If $\rho(\mathbf{W}_{hh}) > 1$, gradients explode exponentially.

3 Long Short-Term Memory Networks

3.1 Architecture

The LSTM solves the vanishing gradient problem through a gating mechanism and an explicit memory cell.

Definition 3.1 (LSTM Cell)**.** An LSTM cell maintains two states:

- Cell state: $\mathbf{c}^{(t)} \in \mathbb{R}^{d_h}$ (long-term memory)
- Hidden state: $\mathbf{h}^{(t)} \in \mathbb{R}^{d_h}$ (short-term working memory)

And employs three gates, all with sigmoid activation $\sigma(z) = \frac{1}{1+e^{-z}}$:

$$\mathbf{f}^{(t)} = \sigma(\mathbf{W}_f[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_f) \quad \text{(forget gate)} \tag{15}$$

$$\mathbf{i}^{(t)} = \sigma(\mathbf{W}_i[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_i) \quad \text{(input gate)} \tag{16}$$

$$\mathbf{o}^{(t)} = \sigma(\mathbf{W}_o[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_o) \quad \text{(output gate)} \tag{17}$$

where $[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] \in \mathbb{R}^{d_h + d_x}$ denotes concatenation.

3.2 Forward Propagation

The LSTM forward pass consists of:

Step 1: Compute candidate cell state

$$\tilde{\mathbf{c}}^{(t)} = \tanh(\mathbf{W}_c[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}] + \mathbf{b}_c) \tag{18}$$

Step 2: Update cell state

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)} \tag{19}$$

where $\odot$ denotes element-wise (Hadamard) product.

Step 3: Compute hidden state

$$\mathbf{h}^{(t)} = \mathbf{o}^{(t)} \odot \tanh(\mathbf{c}^{(t)}) \tag{20}$$

3.3 Gradient Flow Analysis

Theorem 3.1 (LSTM Gradient Propagation)**.** The gradient of the loss with respect to the cell state exhibits additive structure:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{c}^{(k)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{c}^{(k+1)}} \odot \mathbf{f}^{(k+1)} + \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(k)}} \odot \mathbf{o}^{(k)} \odot (1 - \tanh^2(\mathbf{c}^{(k)})) \tag{21}$$

Proof. From equation (19), taking the derivative:

$$\frac{\partial \mathbf{c}^{(t)}}{\partial \mathbf{c}^{(t-1)}} = \text{diag}(\mathbf{f}^{(t)}) \tag{22}$$

Therefore:

$$\frac{\partial \mathbf{c}^{(t)}}{\partial \mathbf{c}^{(k)}} = \prod_{i=k+1}^{t} \text{diag}(\mathbf{f}^{(i)}) \tag{23}$$

This product of diagonal matrices preserves gradient magnitude better than the matrix product in vanilla RNNs. □

Proposition 3.1 (Constant Error Carousel)**.** When the forget gate is close to 1 ($\mathbf{f}^{(t)} \approx \mathbf{1}$), the gradient flows unimpeded through time:

$$\left\| \frac{\partial \mathbf{c}^{(t)}}{\partial \mathbf{c}^{(k)}} \right\| \approx 1 \tag{24}$$

This property, known as the constant error carousel (CEC), enables learning of long-term dependencies.

3.4 Detailed Backpropagation Derivations

3.4.1 Output Gate Gradients

$$\frac{\partial \mathcal{L}}{\partial \mathbf{o}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t)}} \odot \tanh(\mathbf{c}^{(t)}) \tag{25}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_o} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \mathbf{o}^{(t)}} \odot \mathbf{o}^{(t)} \odot (1 - \mathbf{o}^{(t)}) \cdot [\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}]^\top \tag{26}$$

3.4.2 Cell State Gradients

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{c}^{(t)}} = &\frac{\partial \mathcal{L}}{\partial \mathbf{h}^{(t)}} \odot \mathbf{o}^{(t)} \odot (1 - \tanh^2(\mathbf{c}^{(t)})) \\ &+ \frac{\partial \mathcal{L}}{\partial \mathbf{c}^{(t+1)}} \odot \mathbf{f}^{(t+1)} \end{aligned} \tag{27}$$

3.4.3 Forget Gate Gradients

$$\frac{\partial \mathcal{L}}{\partial \mathbf{f}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{c}^{(t)}} \odot \mathbf{c}^{(t-1)} \tag{28}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_f} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \mathbf{f}^{(t)}} \odot \mathbf{f}^{(t)} \odot (1 - \mathbf{f}^{(t)}) \cdot [\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}]^\top \tag{29}$$

3.4.4 Input Gate Gradients

$$\frac{\partial \mathcal{L}}{\partial \mathbf{i}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{c}^{(t)}} \odot \tilde{\mathbf{c}}^{(t)} \tag{30}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_i} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \mathbf{i}^{(t)}} \odot \mathbf{i}^{(t)} \odot (1 - \mathbf{i}^{(t)}) \cdot [\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}]^\top \tag{31}$$

3.4.5 Candidate Cell State Gradients

$$\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{c}}^{(t)}} = \frac{\partial \mathcal{L}}{\partial \mathbf{c}^{(t)}} \odot \mathbf{i}^{(t)} \tag{32}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_c} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{c}}^{(t)}} \odot (1 - \tilde{\mathbf{c}}^{(t)} \odot \tilde{\mathbf{c}}^{(t)}) \cdot [\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}]^\top \tag{33}$$

4 Training Dynamics

4.1 Optimization Algorithm

Training uses gradient descent with the update rule:

$$\boldsymbol{\theta}^{(n+1)} = \boldsymbol{\theta}^{(n)} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^{(n)}) \tag{34}$$

where $\eta > 0$ is the learning rate.

4.2 Adam Optimizer

The Adaptive Moment Estimation (Adam) algorithm maintains first and second moment estimates:

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\mathbf{g}_t \tag{35}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)\mathbf{g}_t^2 \tag{36}$$

Bias-corrected estimates:

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{1 - \beta_1^t} \tag{37}$$

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{1 - \beta_2^t} \tag{38}$$

Parameter update:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{\sqrt{\hat{\mathbf{v}}_t} + \epsilon} \hat{\mathbf{m}}_t \tag{39}$$

Typical hyperparameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $\eta = 0.001$.

4.3 Gradient Clipping

To prevent exploding gradients, we apply gradient clipping:

$$\mathbf{g} \leftarrow \begin{cases} \mathbf{g} & \text{if } \|\mathbf{g}\| \leq \theta \\ \frac{\theta \mathbf{g}}{\|\mathbf{g}\|} & \text{if } \|\mathbf{g}\| > \theta \end{cases} \tag{40}$$

where θ is the clipping threshold (typically $\theta = 5$).

4.4 Regularization

4.4.1 Dropout

During training, randomly set activations to zero with probability p:

$$\tilde{\mathbf{h}}^{(t)} = \mathbf{h}^{(t)} \odot \mathbf{m}^{(t)}, \quad m_i^{(t)} \sim \text{Bernoulli}(1-p) \tag{41}$$

At test time, scale by $(1-p)$:

$$\mathbf{h}_{\text{test}}^{(t)} = (1-p)\mathbf{h}^{(t)} \tag{42}$$

4.4.2 Weight Decay (L2 Regularization)

Add penalty term to loss:

$$\mathcal{L}_{\text{reg}}(\boldsymbol{\theta}) = \mathcal{L}(\boldsymbol{\theta}) + \frac{\lambda}{2}\sum_i \theta_i^2 \tag{43}$$

Gradient update becomes:

$$\frac{\partial \mathcal{L}_{\text{reg}}}{\partial \theta_i} = \frac{\partial \mathcal{L}}{\partial \theta_i} + \lambda\theta_i \tag{44}$$

5 Theoretical Analysis

5.1 Universal Approximation

Theorem 5.1 (RNN Universal Approximation)**.** RNNs with sufficient hidden units can approximate any measurable sequence-to-sequence mapping to arbitrary accuracy.

5.2 Expressiveness

Proposition 5.1 (LSTM vs RNN Expressiveness)**.** LSTMs are strictly more expressive than vanilla RNNs due to:

1. Separate cell and hidden states: $(\mathbf{c}^{(t)}, \mathbf{h}^{(t)})$ vs $\mathbf{h}^{(t)}$
2. Multiplicative gating allowing dynamic routing
3. Linear additive cell state updates preserving information

5.3 Sample Complexity

Theorem 5.2 (Generalization Bound)**.** With probability $1-\delta$, the generalization error of an LSTM satisfies:

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[\mathcal{L}(\mathbf{y}, f(\mathbf{x}))] \leq \hat{\mathcal{L}}_n + \mathcal{O}\left(\sqrt{\frac{d\log(n/\delta)}{n}}\right) \tag{45}$$

where $\hat{\mathcal{L}}_n$ is the empirical loss, n is sample size, and d is the number of parameters.

6 Computational Complexity

6.1 Time Complexity

For a sequence of length T with d_x-dimensional input and d_h-dimensional hidden state:

Vanilla RNN:

$$\mathcal{O}(T \cdot d_h \cdot (d_h + d_x)) \tag{46}$$

LSTM:

$$\mathcal{O}(4T \cdot d_h \cdot (d_h + d_x)) \tag{47}$$

The factor of 4 accounts for the four gates (forget, input, output, cell candidate).

6.2 Space Complexity

Parameters:

- RNN: $(d_h + d_x) \times d_h + d_h + d_y \times d_h + d_y$
- LSTM: $4[(d_h + d_x) \times d_h + d_h] + d_y \times d_h + d_y$

Activations (for BPTT):

$$\mathcal{O}(T \cdot d_h) \tag{48}$$

7 Financial Applications

7.1 Return Prediction

Let $r_t = \log(P_t/P_{t-1})$ be the log-return. The prediction problem is:

$$\hat{r}_{t+1} = f_{\boldsymbol{\theta}}(r_1, \ldots, r_t) \tag{49}$$

7.2 Volatility Forecasting

Predict conditional variance:

$$\hat{\sigma}^2_{t+1} = \mathbb{E}[r^2_{t+1} \mid r_1, \ldots, r_t] \tag{50}$$

The LSTM can capture volatility clustering: $\mathbb{E}[\sigma^2_{t+1} \mid \sigma^2_t] > \mathbb{E}[\sigma^2_{t+1}]$ when σ^2_t is high.

7.3 Portfolio Optimization

Given predicted returns $\hat{\mathbf{r}}_{t+1}$ and covariance matrix $\hat{\boldsymbol{\Sigma}}_{t+1}$:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \left\{ \mathbf{w}^\top \hat{\mathbf{r}}_{t+1} - \frac{\gamma}{2} \mathbf{w}^\top \hat{\boldsymbol{\Sigma}}_{t+1} \mathbf{w} \right\} \tag{51}$$

subject to $\mathbf{1}^\top \mathbf{w} = 1$ and $\mathbf{w} \geq \mathbf{0}$.

7.4 Risk Metrics

7.4.1 Value at Risk (VaR)

The α-VaR is the α-quantile of the loss distribution:

$$\text{VaR}_\alpha = \inf\{l : P(L \leq l) \geq \alpha\} \tag{52}$$

7.4.2 Expected Shortfall (CVaR)

$$\text{CVaR}_\alpha = \mathbb{E}[L \mid L \geq \text{VaR}_\alpha] \tag{53}$$

LSTMs can model the full distribution $P(r_{t+1} \mid r_1, \ldots, r_t)$ to estimate these risk measures.

8 Convergence Analysis

8.1 Convergence of SGD

Theorem 8.1 (SGD Convergence for LSTMs)**.** Under assumptions of Lipschitz continuity and bounded variance of stochastic gradients, SGD converges to a stationary point. For a loss function $\mathcal{L}$ that is L-smooth:

$$\mathbb{E}\left[\min_{t \leq T} \|\nabla \mathcal{L}(\boldsymbol{\theta}_t)\|^2\right] \leq \frac{2(\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}^*)}{\eta T} + \frac{L \eta \sigma^2}{B} \tag{54}$$

where B is batch size and σ^2 is gradient variance.

8.2 Local Minima and Saddle Points

Proposition 8.1 (Landscape Properties)**.** For overparameterized LSTMs ($d_h \gg$ required), most critical points are saddle points rather than local minima. The Hessian at a saddle point has at least one negative eigenvalue, allowing gradient descent to escape.

9 Advanced Topics

9.1 Gated Recurrent Units (GRU)

A simplified alternative to LSTM with two gates:

Update gate:

$$\mathbf{z}^{(t)} = \sigma(\mathbf{W}_z[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}]) \tag{55}$$

Reset gate:

$$\mathbf{r}^{(t)} = \sigma(\mathbf{W}_r[\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}]) \tag{56}$$

Candidate hidden state:

$$\tilde{\mathbf{h}}^{(t)} = \tanh(\mathbf{W}[\mathbf{r}^{(t)} \odot \mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}]) \tag{57}$$

Final hidden state:

$$\mathbf{h}^{(t)} = (1 - \mathbf{z}^{(t)}) \odot \mathbf{h}^{(t-1)} + \mathbf{z}^{(t)} \odot \tilde{\mathbf{h}}^{(t)} \tag{58}$$

GRUs have $3(d_h + d_x) \times d_h$ parameters vs LSTM's $4(d_h + d_x) \times d_h$.

9.2 Bidirectional LSTMs

Process sequences in both directions:

$$\overrightarrow{\mathbf{h}}^{(t)} = \text{LSTM}_{\text{forward}}(\mathbf{x}^{(t)}, \overrightarrow{\mathbf{h}}^{(t-1)}) \tag{59}$$

$$\overleftarrow{\mathbf{h}}^{(t)} = \text{LSTM}_{\text{backward}}(\mathbf{x}^{(t)}, \overleftarrow{\mathbf{h}}^{(t+1)}) \tag{60}$$

$$\mathbf{h}^{(t)} = [\overrightarrow{\mathbf{h}}^{(t)}, \overleftarrow{\mathbf{h}}^{(t)}] \tag{61}$$

Useful for sequence labeling but not for forecasting (can't use future information).

9.3 Attention Mechanisms

Instead of fixed hidden state, compute weighted average:

$$\mathbf{c}_t = \sum_{i=1}^{T} \alpha_{ti} \mathbf{h}_i \tag{62}$$

where attention weights are:

$$\alpha_{ti} = \frac{\exp(e_{ti})}{\sum_{j=1}^{T} \exp(e_{tj})}, \quad e_{ti} = \text{score}(\mathbf{h}_t, \mathbf{h}_i) \tag{63}$$

Common scoring functions:

- Dot product: $e_{ti} = \mathbf{h}_t^\top \mathbf{h}_i$
- Bilinear: $e_{ti} = \mathbf{h}_t^\top \mathbf{W} \mathbf{h}_i$
- Additive: $e_{ti} = \mathbf{v}^\top \tanh(\mathbf{W}_1 \mathbf{h}_t + \mathbf{W}_2 \mathbf{h}_i)$

10 Numerical Stability

10.1 Initialization Strategies

Xavier/Glorot Initialization:

$$W_{ij} \sim \mathcal{U}\left(-\sqrt{\frac{6}{d_{\text{in}} + d_{\text{out}}}}, \sqrt{\frac{6}{d_{\text{in}} + d_{\text{out}}}}\right) \tag{64}$$

He Initialization (for ReLU):

$$W_{ij} \sim \mathcal{N}\left(0, \frac{2}{d_{\text{in}}}\right) \tag{65}$$

LSTM-specific:

- Forget gate bias: $b_f = 1$ (start with remembering)
- Other biases: $b = 0$
- Weights: Xavier initialization

10.2 Batch Normalization for RNNs

Apply batch normalization to hidden states:

$$\hat{\mathbf{h}}^{(t)} = \frac{\mathbf{h}^{(t)} - \boldsymbol{\mu}_{\mathcal{B}}}{\sqrt{\boldsymbol{\sigma}_{\mathcal{B}}^2 + \epsilon}} \tag{66}$$

However, this requires careful handling of temporal statistics. Layer normalization is often preferred:

$$\hat{\mathbf{h}}^{(t)} = \frac{\mathbf{h}^{(t)} - \mu_t}{\sqrt{\sigma_t^2 + \epsilon}} \tag{67}$$

where μ_t and σ_t^2 are computed over feature dimension for each time step.

11 Conclusion

We have presented a comprehensive mathematical treatment of RNNs and LSTMs, covering:

1. Forward and backward propagation with complete derivations
2. Analysis of vanishing/exploding gradients and solutions
3. LSTM architecture with gating mechanisms
4. Training dynamics and optimization algorithms
5. Theoretical guarantees and complexity analysis
6. Applications to financial time series prediction

The key insight is that LSTMs solve the vanishing gradient problem through:

- Additive cell state updates (constant error carousel)
- Multiplicative gates controlling information flow
- Separation of long-term (cell) and short-term (hidden) memory

These properties enable LSTMs to capture long-term dependencies essential for financial forecasting, where patterns may span hundreds of time steps.

References

1. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735-1780.

2. Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *ICML*.

3. Graves, A. (2012). Supervised sequence labelling with recurrent neural networks. *Springer*.

4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

5. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *ICLR*.