Recurrent Neural Network - Derivations & Proofs

Paul F. Roysdon, Ph.D.

Contents

1 Mathematical Derivations & Proofs

1.1 Introduction

A **Recurrent Neural Network (RNN)** models sequential data by maintaining a latent *hidden state* that is updated recurrently as new inputs arrive. At each time step, the new hidden state is a nonlinear function of the current input and the previous hidden state, and predictions are produced from the hidden state. Training proceeds by minimizing a sequence loss via *Backpropagation Through Time* (BPTT), which computes exact gradients by unrolling the computation graph over time. We derive the forward equations, the full BPTT gradient recursions, and analyze vanishing/exploding gradients from first principles.

1.2 Data and Notation

Let a sequence be $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T}$, where

$$\mathbf{x}_t \in \mathbb{R}^{d_x} \text{ (input at time } t\text{)}, \qquad \mathbf{y}_t \in \{0,1\}^K \text{ (one-hot class target or a real vector)}.$$

Hidden states $\mathbf{h}_t \in \mathbb{R}^{d_h}$. Parameters:

$$\mathbf{W}_{xh} \in \mathbb{R}^{d_h \times d_x}, \quad \mathbf{W}_{hh} \in \mathbb{R}^{d_h \times d_h}, \quad \mathbf{b}_h \in \mathbb{R}^{d_h}, \qquad \mathbf{W}_{hy} \in \mathbb{R}^{K \times d_h}, \quad \mathbf{b}_y \in \mathbb{R}^{K}.$$

We denote the elementwise nonlinearity by $\phi(\cdot)$ (e.g., tanh or ReLU). The Hadamard (elementwise) product is $\odot$; $\texttt{Diag}(\cdot)$ forms a diagonal matrix from a vector.

1.3 Model Formulation (Vanilla RNN)

Given an initial hidden $\mathbf{h}_0$ (zero or learned), the forward dynamics are

$$\begin{aligned} \text{Pre-activation:} \quad & \mathbf{s}_t = \mathbf{W}_{xh}\,\mathbf{x}_t + \mathbf{W}_{hh}\,\mathbf{h}_{t-1} + \mathbf{b}_h \in \mathbb{R}^{d_h}, && (1) \\ \text{Hidden update:} \quad & \mathbf{h}_t = \phi(\mathbf{s}_t) \in \mathbb{R}^{d_h}, && (2) \\ \text{Output logits:} \quad & \mathbf{z}_t = \mathbf{W}_{hy}\,\mathbf{h}_t + \mathbf{b}_y \in \mathbb{R}^{K}. && (3) \end{aligned}$$

For classification, the predictive distribution at time t is the softmax

$$\hat{\mathbf{p}}_t = \mathrm{softmax}(\mathbf{z}_t), \qquad [\hat{\mathbf{p}}_t]_k = \frac{e^{z_{t,k}}}{\sum_{j=1}^{K} e^{z_{t,j}}}.$$

Sequence topologies. *Many-to-many* (per-time outputs): use all $\mathbf{z}_t$. *Many-to-one* (sequence classification): use only the final (or pooled) hidden $\mathbf{h}_T$ for prediction via Eqn. (3) with $t = T$.

1.4 Training Objective (Empirical Risk)

For sequence classification with per-step cross-entropy, the loss of one sequence is

$$\mathcal{L} \;=\; \sum_{t=1}^{T} \ell_t, \qquad \ell_t \;=\; -\mathbf{y}_t^{\top} \log \hat{\mathbf{p}}_t \;=\; -\sum_{k=1}^{K} y_{t,k} \log[\hat{\mathbf{p}}_t]_k. \tag{4}$$

(For many-to-one, drop the sum and keep only $t = T$.) Optionally add weight decay:

$$\tfrac{\lambda}{2}\left(\|\mathbf{W}_{xh}\|_F^2 + \|\mathbf{W}_{hh}\|_F^2 + \|\mathbf{W}_{hy}\|_F^2\right).$$

1.5 BPTT: Full Derivation

Loss Function

Suppose we have a loss function $\ell(\mathbf{y}_t, \hat{\mathbf{y}}_t)$ at each time step, where $\hat{\mathbf{y}}_t$ is the target output. The total loss over the sequence is given by

$$\mathcal{L} = \sum_{t=1}^{T} \ell(\mathbf{y}_t, \hat{\mathbf{y}}_t).$$

Our goal is to compute the gradients of $\mathcal{L}$ with respect to the network parameters (e.g., $\mathbf{W}_{hh}$, $\mathbf{W}_{xh}$, and $\mathbf{W}_{hy}$).

Notation and Setup

For compactness, we define the pre-activation (or "net input") at time t as

$$\boldsymbol{z}_t = \mathbf{W}_{hh}\,\mathbf{h}_{t-1} + \mathbf{W}_{xh}\,\mathbf{x}_t + \mathbf{b}_h,$$

so that

$$\mathbf{h}_t = \phi(\boldsymbol{z}_t).$$

Because the network is recurrent, $\mathbf{h}_t$ influences not only the loss at time t (via $\mathbf{y}_t$) but also the losses at future time steps.

Gradient with Respect to $\mathbf{W}_{hh}$

The total gradient with respect to $\mathbf{W}_{hh}$ is given by summing the contributions from each time step:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}}\bigg|_t.$$

Since $\mathbf{W}_{hh}$ affects the hidden state at time t via

$$\mathbf{h}_t = \phi(\boldsymbol{z}_t) \quad \text{with} \quad \boldsymbol{z}_t = \mathbf{W}_{hh}\,\mathbf{h}_{t-1} + \cdots,$$

by the chain rule we have

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \mathbf{h}_t}\,\frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{hh}}.$$

Define the "error" at the hidden state as

$$\boldsymbol{\delta}_t^h \triangleq \frac{\partial \mathcal{L}}{\partial \mathbf{h}_t}.$$

Because $\mathbf{h}_t$ contributes directly to the loss at time t and indirectly to future losses, we obtain the recursive relation:

$$\boldsymbol{\delta}_t^h = \underbrace{\frac{\partial \ell_t}{\partial \mathbf{h}_t}}_{\text{direct contribution}} + \underbrace{\left(\frac{\partial \mathcal{L}}{\partial \mathbf{h}_{t+1}}\frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}\right)}_{\text{future contributions}}.$$

That is, for $t = 1, \ldots, T-1$,

$$\boldsymbol{\delta}_t^h = \frac{\partial \ell_t}{\partial \mathbf{h}_t} + \delta_{t+1}^h\,\frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}.$$

For the final time step $t = T$, we have:

$$\boldsymbol{\delta}_T^h = \frac{\partial \ell_T}{\partial \mathbf{h}_T}.$$

Next, since

$$\mathbf{h}_t = \phi(\boldsymbol{z}_t) \quad \text{with} \quad \boldsymbol{z}_t = \mathbf{W}_{hh}\,\mathbf{h}_{t-1} + \cdots,$$

the chain rule gives:

$$\frac{\partial \mathbf{h}_t}{\partial \boldsymbol{z}_t} = \phi'(\boldsymbol{z}_t).$$

(When ϕ is applied elementwise, $\phi'(\boldsymbol{z}_t)$ is a diagonal matrix with the derivatives on the diagonal.) Also, because

$$\boldsymbol{z}_t = \mathbf{W}_{hh}\,\mathbf{h}_{t-1} + \cdots,$$

we have:

$$\frac{\partial \boldsymbol{z}_t}{\partial \mathbf{W}_{hh}} = \mathbf{h}_{t-1}^T.$$

Thus, the gradient of $\mathbf{h}_t$ with respect to $\mathbf{W}_{hh}$ is:

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{W}_{hh}} = \phi'(\boldsymbol{z}_t)\,\mathbf{h}_{t-1}^T.$$

Hence, the contribution at time t is:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}}\bigg|_t = \boldsymbol{\delta}_t^h\,\phi'(\boldsymbol{z}_t)\,\mathbf{h}_{t-1}^T.$$

Thus, the overall gradient is:

$$\boxed{\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \boldsymbol{\delta}_t^h\,\phi'(\boldsymbol{z}_t)\,\mathbf{h}_{t-1}^T.}$$

Unrolling the Error Back in Time

Because the hidden state at time t influences not only the loss at time t but also losses at later times, the total error δ_t^h can be written explicitly in terms of the losses at all future time steps. Unrolling the recursion yields:

$$\boldsymbol{\delta}_t^h = \frac{\partial \ell_t}{\partial \mathbf{h}_t} + \frac{\partial \ell_{t+1}}{\partial \mathbf{h}_{t+1}} \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t} + \frac{\partial \ell_{t+2}}{\partial \mathbf{h}_{t+2}} \frac{\partial \mathbf{h}_{t+2}}{\partial \mathbf{h}_t} + \cdots,$$

or, more compactly,

$$\boldsymbol{\delta}_t^h = \sum_{k=t}^{T} \frac{\partial \ell_k}{\partial \mathbf{h}_k} \left(\prod_{j=t+1}^{k} \frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}} \right).$$

Since each Jacobian $\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}}$ is given by

$$\frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}} = \phi'(\boldsymbol{z}_j)\, \mathbf{W}_{hh},$$

we have:

$$\prod_{j=t+1}^{k} \frac{\partial \mathbf{h}_j}{\partial \mathbf{h}_{j-1}} = \prod_{j=t+1}^{k} \left[\phi'(\boldsymbol{z}_j)\, \mathbf{W}_{hh}\right],$$

with the convention that when $k = t$, the product is the identity.

Thus, the full expression for the gradient of the loss with respect to $\mathbf{W}_{hh}$ becomes:

$$\boxed{\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \left[\left(\sum_{k=t}^{T} \frac{\partial \ell_k}{\partial \mathbf{h}_k} \prod_{j=t+1}^{k} \left[\phi'(\boldsymbol{z}_j)\, \mathbf{W}_{hh}\right] \right) \phi'(\boldsymbol{z}_t)\, \mathbf{h}_{t-1}^T \right]}.$$

A similar derivation applies for the gradients with respect to $\mathbf{W}_{xh}$ and the bias $\mathbf{b}_h$.

1.6 BPTT: Summary

We unroll Eqns. (1)–(3) over $t = 1{:}T$ and apply reverse-mode differentiation.

Local derivatives. Define the output error at time t (softmax with cross-entropy):

$$\boldsymbol{\delta}_t^{(z)} \triangleq \frac{\partial \ell_t}{\partial \mathbf{z}_t} = \hat{\mathbf{p}}_t - \mathbf{y}_t \in \mathbb{R}^K. \tag{5}$$

Propagating to hidden $\mathbf{h}_t$ through the linear head:

$$\mathbf{g}_t \triangleq \frac{\partial \ell}{\partial \mathbf{h}_t} = \mathbf{W}_{hy}^\top \boldsymbol{\delta}_t^{(z)} + \underbrace{\left(\frac{\partial \ell}{\partial \mathbf{h}_t}\right)_{\text{from future}}}_{\text{to be accumulated}}. \tag{6}$$

Through the nonlinearity at time t:

$$\boldsymbol{\delta}_t^{(s)} \triangleq \frac{\partial \ell}{\partial \mathbf{s}_t} = \mathbf{g}_t \odot \phi'(\mathbf{s}_t) \in \mathbb{R}^{d_h}. \tag{7}$$

Temporal recursion (error accumulation). Because $\mathbf{h}_t$ influences *all* future losses via $\mathbf{h}_{t+1}, \mathbf{h}_{t+2}, \ldots$ through $\mathbf{W}_{hh}$, the gradient $\left(\frac{\partial \ell}{\partial \mathbf{h}_t}\right)_{\text{from future}}$ satisfies the backward recurrence

$$\left(\frac{\partial \ell}{\partial \mathbf{h}_t}\right)_{\text{from future}} = \mathbf{W}_{hh}^\top\, \boldsymbol{\delta}_{t+1}^{(s)} \quad (\text{with } \boldsymbol{\delta}_{T+1}^{(s)} = \mathbf{0}). \tag{8}$$

Combining Eqns. (6)–(8), running $t = T, T-1, \ldots, 1$,

$$\mathbf{g}_t = \mathbf{W}_{hy}^\top \boldsymbol{\delta}_t^{(z)} + \mathbf{W}_{hh}^\top \boldsymbol{\delta}_{t+1}^{(s)}, \qquad \boldsymbol{\delta}_t^{(s)} = \mathbf{g}_t \odot \phi'(\mathbf{s}_t). \tag{9}$$

Gradients w.r.t. parameters. Summing contributions over time gives

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hy}} = \sum_{t=1}^{T} \boldsymbol{\delta}_t^{(z)}\, \mathbf{h}_t^\top, \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{b}_y} = \sum_{t=1}^{T} \boldsymbol{\delta}_t^{(z)}, \tag{10}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}_{xh}} = \sum_{t=1}^{T} \boldsymbol{\delta}_t^{(s)}\, \mathbf{x}_t^\top, \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \boldsymbol{\delta}_t^{(s)}\, \mathbf{h}_{t-1}^\top, \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{b}_h} = \sum_{t=1}^{T} \boldsymbol{\delta}_t^{(s)}. \tag{11}$$

The gradient w.r.t. the initial hidden (useful when it is learned) is $\frac{\partial \mathcal{L}}{\partial \mathbf{h}_0} = \mathbf{W}_{hh}^\top \boldsymbol{\delta}_1^{(s)}$.

Proof (chain rule over time). Differentiate $\ell = \sum_t \ell_t(\mathbf{z}_t(\mathbf{h}_t), \ldots)$; apply chain rule

$$\frac{\partial \ell}{\partial \mathbf{h}_t} = \frac{\partial \ell_t}{\partial \mathbf{h}_t} + \left(\frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}\right)^\top \frac{\partial \ell}{\partial \mathbf{h}_{t+1}} = \mathbf{W}_{hy}^\top \boldsymbol{\delta}_t^{(z)} + \left(\mathtt{Diag}(\phi'(\mathbf{s}_{t+1}))\, \mathbf{W}_{hh}\right)^\top \boldsymbol{\delta}_{t+1}^{(s)},$$

which yields Eqn. (9); parameter gradients follow from local Jacobians $\partial \mathbf{s}_t / \partial \mathbf{W}_{xh} = (\cdot)\, \mathbf{x}_t^\top$, $\partial \mathbf{s}_t / \partial \mathbf{W}_{hh} = (\cdot)\, \mathbf{h}_{t-1}^\top$, etc. ■

1.7 Vanishing/Exploding Gradients: Spectral Analysis

Consider the linearized recurrence around a trajectory:

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \mathtt{Diag}(\phi'(\mathbf{s}_t))\, \mathbf{W}_{hh} \;\triangleq\; \mathbf{J}_t.$$

The backprop factor from time $t+k$ to t is the product $\mathbf{J}_{t+k}^\top \cdots \mathbf{J}_{t+1}^\top$. Hence, for any consistent matrix norm,

$$\left\| \frac{\partial \ell}{\partial \mathbf{h}_t} \right\| \le \sum_{k=0}^{T-t} \|\mathbf{W}_{hy}^\top\|\, \|\mathbf{J}_{t+1}^\top \cdots \mathbf{J}_{t+k}^\top\|\, \|\boldsymbol{\delta}_{t+k}^{(z)}\|.$$

If $\|\mathbf{J}_\tau\| \le \rho < 1$ uniformly (e.g., $\|\mathbf{W}_{hh}\| \max \|\phi'\| < 1$), products decay $\mathcal{O}(\rho^k)$ (*vanishing gradients*). If $\rho > 1$, they grow (*exploding gradients*). In the linear RNN with $\phi(\cdot) = \mathrm{id}$, $\mathbf{J}_t = \mathbf{W}_{hh}$ and $\partial \ell / \partial \mathbf{h}_t$ contains $(\mathbf{W}_{hh}^\top)^k$; behavior is governed by the spectral radius $\rho(\mathbf{W}_{hh})$.

Mitigations. Gradient clipping (e.g., rescale when $\|\nabla\|_2 > c$), orthogonal/identity initialization of $\mathbf{W}_{hh}$, gated architectures (LSTM/GRU) that modify $\mathbf{J}_t$ to keep eigenvalues near 1, careful choice of ϕ (e.g., tanh vs. ReLU) and normalization.

1.8 Optimization and Practical Variants

- **Truncated BPTT.** For long sequences, accumulate gradients over windows of length $L \ll T$ by periodically detaching the computation graph: preserve local dependencies while controlling memory and compute.
- **Teacher forcing / scheduled sampling.** For auto-regressive outputs, during training feed ground-truth past outputs; at inference, feed model outputs.
- **Regularization.** Weight decay, dropout on inputs/hidden states, gradient clipping, early stopping.
- **Initialization.** $\mathbf{W}_{hh}$ orthogonal or scaled identity; $\mathbf{b}_h$ sometimes positive for ReLU to avoid dead states; $\mathbf{h}_0$ zeros or learned.

1.9 Algorithm (Vanilla RNN + BPTT)

1. **Input:** sequence $\{(\mathbf{x}_t, \mathbf{y}_t)\}_{t=1}^{T}$, learning rate η, (optional) truncation length L.
2. **Forward:** set $\mathbf{h}_0$; for $t = 1{:}T$ compute Eqns. (1)–(3), $\hat{\mathbf{p}}_t$, and accumulate $\mathcal{L}$ via Eqn. (4).
3. **Backward:** initialize $\boldsymbol{\delta}_{T+1}^{(s)} = \mathbf{0}$. For $t = T{:}1$:
   (a) $\boldsymbol{\delta}_t^{(z)} = \hat{\mathbf{p}}_t - \mathbf{y}_t$ (Eqn. (5))
   (b) $\mathbf{g}_t = \mathbf{W}_{hy}^{\top}\boldsymbol{\delta}_t^{(z)} + \mathbf{W}_{hh}^{\top}\boldsymbol{\delta}_{t+1}^{(s)}$; $\boldsymbol{\delta}_t^{(s)} = \mathbf{g}_t \odot \phi'(\mathbf{s}_t)$ (Eqn. (9))
   (c) Accumulate gradients using Eqns. (10)–(11).
4. **Update:** $\Theta \leftarrow \Theta - \eta\nabla\mathcal{L}$ with SGD/Adam (with clipping if needed).

1.10 Computational Aspects

Per-step costs: forward $\mathcal{O}(d_h d_x + d_h^2 + K d_h)$; backward comparable. Memory $\mathcal{O}(T d_h)$ to store $\{\mathbf{s}_t, \mathbf{h}_t\}$ unless using checkpointing or truncation.

1.11 Summary of Variables and Their Dimensions

- $\mathbf{x}_t \in \mathbb{R}^{d_x}$: input at time t; $\mathbf{y}_t \in \{0,1\}^K$ (one-hot) or $\mathbb{R}^K$ (real target).
- $\mathbf{h}_t \in \mathbb{R}^{d_h}$: hidden state; $\mathbf{s}_t \in \mathbb{R}^{d_h}$: pre-activation; $\mathbf{z}_t \in \mathbb{R}^K$: logits.
- $\hat{\mathbf{p}}_t \in \mathbb{R}^K$: softmax probabilities.
- $\mathbf{W}_{xh} \in \mathbb{R}^{d_h \times d_x}$, $\mathbf{W}_{hh} \in \mathbb{R}^{d_h \times d_h}$, $\mathbf{b}_h \in \mathbb{R}^{d_h}$.
- $\mathbf{W}_{hy} \in \mathbb{R}^{K \times d_h}$, $\mathbf{b}_y \in \mathbb{R}^K$.
- Backprop errors: $\boldsymbol{\delta}_t^{(z)} \in \mathbb{R}^K$, $\mathbf{g}_t \in \mathbb{R}^{d_h}$, $\boldsymbol{\delta}_t^{(s)} \in \mathbb{R}^{d_h}$.

1.12 Summary

From first principles, a vanilla RNN iterates the nonlinear state update $\mathbf{h}_t = \phi(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)$ and produces logits $\mathbf{z}_t = \mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y$. Unrolling over time and applying the chain rule yields exact BPTT: the hidden-state adjoint obeys the backward recursion $\mathbf{g}_t = \mathbf{W}_{hy}^{\top}(\hat{\mathbf{p}}_t - \mathbf{y}_t) + \mathbf{W}_{hh}^{\top}\boldsymbol{\delta}_{t+1}^{(s)}$, with $\boldsymbol{\delta}_t^{(s)} = \mathbf{g}_t \odot \phi'(\mathbf{s}_t)$, and the parameter gradients are simple outer-product accumulations over time. A spectral analysis of the Jacobian product explains vanishing/exploding gradients and motivates practical remedies (clipping, orthogonal init, gating). This provides a complete, dimensionally explicit foundation for implementing and training vanilla RNNs with BPTT.