Information Propagation in Deep Networks

Handout Notes - Generative and Deep Learning (GDL)

Davide Bacciu - University of Pisa

Notation. Sequential data are written as $\mathbf{x} = (x_1, \ldots, x_T)$, where x_t is the input at position or time t. The hidden state of a recurrent model is $h_t \in \mathbb{R}^m$, the pre-activation is g_t, and the output is y_t. The recurrent weight matrix is W_h, the input-to-hidden matrix is W_{in}, and the output matrix is W_{out}. For a vector-valued function f, its Jacobian at h is denoted by $J_f(h) = \frac{\partial f}{\partial h}$. A dataset is denoted by $\mathcal{D}$ with size $N = |\mathcal{D}|$. We use $\ell(\theta) = \log p(\mathcal{D} \mid \theta)$ for log-likelihood when probabilistic notation is relevant, and $\mathcal{L}$ for a generic supervised loss. The spectral radius of a square matrix A is denoted by $\rho(A)$, while singular values are used when discussing norm growth and contraction.

1 Why information propagation matters in deep sequential models

Deep models are not only difficult because they contain many parameters; they are difficult because information must be propagated across many computational steps. In feedforward networks these steps correspond to layers. In recurrent neural networks (RNNs), they correspond to time steps. In both cases, training and inference depend on whether signals can travel through the network without disappearing or exploding.

Sequential data make this issue especially visible. A sequence is an ordered collection

$$\mathbf{x} = (x_1, \ldots, x_T),$$

where the interpretation of x_t depends on its context, that is, on what comes before (and sometimes after) it. This includes time series, natural language, biological sequences, event logs, and many other structured signals. A sequential model must therefore summarize relevant history in some internal representation that evolves over time.

In a recurrent model, this representation is the hidden state h_t. The central promise of recurrence is that the state acts as a learned memory:

$$h_t \text{ should summarize the relevant information in } x_1, \ldots, x_t.$$

The central difficulty is that this memory must be maintained through repeated transformations. If those transformations are too contractive, information disappears. If they are too expansive, the dynamics become unstable. This is the core propagation problem in deep recurrent networks.

2 Sequential data and recurrent processing

A sequence model processes inputs one step at a time. At each step, it receives a new input x_t and updates its memory state. This makes recurrent networks naturally suited to variable-length data, unlike fixed-size feedforward architectures that expect a vector of predetermined dimensionality.

Sequential learning tasks come in several forms:

- **sequence-to-sequence:** map an input sequence to an output sequence,
- **sequence-to-item:** produce one prediction after seeing the whole sequence,

- **item-to-sequence:** generate a sequence from a single input or code,
- **item-to-item over time:** output one prediction per input step.

What all these settings share is the need to propagate information through time.

A recurrent network is therefore not just a classifier applied repeatedly. It is a dynamical system whose state evolves under the influence of both the past state and the current input.

3 Vanilla recurrent neural networks

A standard vanilla RNN updates its hidden state according to

$$g_t = W_h h_{t-1} + W_{\text{in}} x_t + b_h, \qquad h_t = \tanh(g_t),$$

and computes an output

$$y_t = f(W_{\text{out}} h_t + b_{\text{out}}),$$

where f depends on the task (for example, softmax for classification or the identity for regression).

There are two structural ideas here.

First, the hidden state h_t compresses the past into a fixed-dimensional vector. Thus the network attempts to encode arbitrarily long input histories into a learned memory representation.

Second, the same matrices W_h and W_{in} are reused at every time step. This is weight sharing across time, and it imposes a form of time-stationary dynamics.

If we unroll the model across time, the recurrence becomes a deep computation graph:

$$h_1 \to h_2 \to \cdots \to h_T.$$

This is why recurrent training exhibits the same core phenomenon as very deep feedforward training: repeated nonlinear transformations can severely distort both forward signal propagation and backward gradient propagation.

Worked example — A vanilla RNN as a dynamical system

Consider the recurrence

$$h_t = \tanh(W_h h_{t-1} + W_{\text{in}} x_t + b_h).$$

Suppose that $x_t = 0$ for all t after some time t_0. Then the future state evolves only according to

$$h_t = \tanh(W_h h_{t-1} + b_h), \qquad t > t_0.$$

Thus, even without new inputs, the hidden state follows an autonomous nonlinear dynamical system. This highlights an important fact: the memory of an RNN is not a passive storage unit. It is an evolving state shaped by recurrent dynamics, and those dynamics determine whether past information is retained, distorted, or forgotten.

4 Learning long-term dependencies

Suppose a model makes an error at time t and we want learning to assign credit to an input that occurred at time $k \ll t$. If the gap $t - k$ is large, then the effect of x_k on the loss at time t must be transmitted through many recurrent transitions.

This is the long-term dependency problem. Conceptually, it asks whether the model can keep useful information alive for many steps. Algorithmically, it asks whether gradient-based learning can trace an error signal back far enough in time to update the responsible parameters.

These two views are closely related but not identical:

- the **backward view** studies whether gradients can propagate through time,

- the **forward view** studies whether information about past inputs remains present in the current state.

Both ultimately depend on the same Jacobian products.

5 Backpropagation through time and the exploding/vanishing gradient problem

Training an RNN by gradient descent uses *backpropagation through time* (BPTT), which applies the chain rule over the unrolled computation graph. Because the same recurrent parameters are used at every step, the total gradient is a sum of contributions from all time positions.

If a loss is incurred at time t, then the gradient with respect to recurrent parameters has the form

$$\frac{\partial \mathcal{L}_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial \mathcal{L}_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}.$$

The key term is

$$\frac{\partial h_t}{\partial h_k},$$

because it measures how a perturbation in the hidden state at time k influences the hidden state at time t. By repeated application of the chain rule,

$$\frac{\partial h_t}{\partial h_k} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \cdots \frac{\partial h_{k+1}}{\partial h_k}.$$

Thus the gradient is governed by a *product of many Jacobians*.

If those Jacobians are norm-contracting, the product shrinks exponentially with time distance and the gradient vanishes. If they are norm-expanding, the product can grow exponentially and the gradient explodes. This is the exploding/vanishing gradient problem (EVGP).

Worked example — Why repeated products create vanishing or exploding gradients

Suppose, for intuition, that each recurrent Jacobian is approximately a scalar multiple of the identity:

$$\frac{\partial h_{\ell+1}}{\partial h_\ell} \approx \lambda I.$$

Then

$$\frac{\partial h_t}{\partial h_k} \approx (\lambda I)^{t-k} = \lambda^{t-k} I.$$

If $|\lambda| < 1$, then

$$\|\lambda^{t-k} I\| \to 0 \qquad \text{as } t - k \to \infty,$$

so gradients vanish exponentially. If $|\lambda| > 1$, then

$$\|\lambda^{t-k} I\| \to \infty,$$

so gradients explode. Only the critical regime $|\lambda| \approx 1$ supports stable propagation over long time gaps. Although real RNN Jacobians are matrices rather than scalars, the same logic applies through their spectral properties.

6 The Jacobian in recurrent networks

To analyze the propagation more precisely, consider the state update

$$h_{t+1} = f(h_t, x_{t+1}) = \phi(W_h h_t + W_{\text{in}} x_{t+1} + b_h),$$

where ϕ is applied elementwise. The Jacobian with respect to the previous hidden state is

$$\frac{\partial h_{t+1}}{\partial h_t} = D_{t+1} W_h,$$

or equivalently $D_{t+1} W_h^\top$ depending on whether gradients are written in row or column convention. Here

$$D_{t+1} = \operatorname{diag}\big(\phi'(g_{t+1,1}), \ldots, \phi'(g_{t+1,m})\big)$$

is a diagonal matrix containing the derivatives of the activation function at time $t+1$.

For the common choice $\phi = \tanh$,

$$\phi'(u) = 1 - \tanh^2(u),$$

so

$$D_{t+1} = \operatorname{diag}\big(1 - h_{t+1,1}^2, \ldots, 1 - h_{t+1,m}^2\big)\,.$$

Since $0 \le 1 - \tanh^2(u) \le 1$, the activation Jacobian is always contractive in norm unless units stay exactly in the linear regime near zero.

This is already a warning sign: even before considering W_h, the nonlinearity itself tends to shrink gradients.

7 Bounding gradient propagation

Using the recurrent Jacobian, the backward signal from time t to time k becomes

$$\frac{\partial \mathcal{L}_t}{\partial h_k} = \frac{\partial \mathcal{L}_t}{\partial h_t} \prod_{\ell=k}^{t-1} \frac{\partial h_{\ell+1}}{\partial h_\ell} = \frac{\partial \mathcal{L}_t}{\partial h_t} \prod_{\ell=k}^{t-1} D_{\ell+1} W_h.$$

Taking norms and using submultiplicativity,

$$\left\| \frac{\partial \mathcal{L}_t}{\partial h_k} \right\| \le \left\| \frac{\partial \mathcal{L}_t}{\partial h_t} \right\| \prod_{\ell=k}^{t-1} \|D_{\ell+1}\| \, \|W_h\|.$$

If, over many steps, the dominant singular values of these Jacobians are consistently below 1, then the product shrinks toward zero. If they are above 1, the product grows rapidly.

A commonly used shorthand is to relate this behavior to the spectral properties of the recurrent transformation. In the linear regime, the long-range behavior is controlled by the eigenvalues or singular values of W_h. If the effective Jacobian has spectral radius below one, gradients vanish. If it exceeds one, gradients explode.

Worked example — Why tanh is naturally contractive

For a scalar activation $h = \tanh(g)$,

$$\frac{dh}{dg} = 1 - \tanh^2(g).$$

Since $\tanh(g) \in (-1, 1)$ for all finite g, we have

$$0 < 1 - \tanh^2(g) \leq 1.$$

The derivative is close to 1 only when g is near 0. As $|g|$ grows, $\tanh(g)$ saturates to ± 1 and the derivative approaches 0. Therefore, even if W_h were norm-preserving, repeated multiplication by activation derivatives can still drive gradients toward zero whenever the hidden units enter saturation. This is one of the main reasons why tanh-based vanilla RNNs struggle with long-term credit assignment.

8 Exploding gradients and gradient clipping

Exploding gradients are numerically dangerous because parameter updates become unstable and optimization oscillates or diverges. A standard practical countermeasure is *gradient clipping*.

Let $g = \nabla_\theta \mathcal{L}$ be the full gradient vector. If its norm exceeds a threshold τ, replace it by

$$g \leftarrow \frac{\tau}{\|g\|} g.$$

This preserves the gradient direction but limits its magnitude.

Gradient clipping is effective for explosions because the problem is excessive scale. It does not solve vanishing gradients, because there the difficulty is not an oversized gradient but an exponentially weak signal from distant time steps. Rescaling the final gradient cannot restore information that was lost differentially across time contributions.

9 Seeking constant error propagation

The ideal recurrent system for long-term learning would propagate gradients without exponential growth or decay. At a rough linearized level, this means that the recurrent Jacobian should behave like a norm-preserving transformation:

$$\left\| \frac{\partial h_{\ell+1}}{\partial h_\ell} \right\| \approx 1.$$

There are two main levers for approaching this regime:

1. choose an activation function whose derivative is not strongly contractive,
2. choose a recurrent matrix with favorable spectral properties.

9.1 Activation-function perspective

Sigmoid and tanh are contractive almost everywhere, so they naturally encourage gradient decay. A much simpler extreme alternative is to use the identity activation:

$$h_{t+1} = W_h h_t + W_{\text{in}} x_{t+1} + b_h.$$

Then the Jacobian becomes

$$\frac{\partial h_{t+1}}{\partial h_t} = W_h,$$

without an extra diagonal shrinkage term from the nonlinearity.

9.2 Recurrent-weight perspective

If W_h is orthogonal, then

$$W_h^\top W_h = I,$$

and its singular values are all exactly 1. Thus, with identity activation and orthogonal recurrence, the state transition is norm-preserving in the linear regime. Special cases include:

- orthogonal matrices,
- unitary matrices in the complex domain,
- the identity matrix itself.

These choices motivate several families of recurrent architectures designed to maintain stable propagation.

Worked example — The constant-error idealized recurrence

Suppose we choose identity activation and identity recurrent matrix:

$$h_t = h_{t-1} + c(x_t),$$

where $c(x_t)$ denotes some input encoding. Then

$$\frac{\partial h_t}{\partial h_{t-1}} = I,$$

and therefore

$$\frac{\partial h_t}{\partial h_k} = I \qquad \text{for all } k < t.$$

Backward errors are propagated unchanged: there is neither vanishing nor exploding gradient. However, this solution is too naive in practice. Because the state simply accumulates incoming contributions, memory can saturate with redundant or irrelevant information. The model lacks an effective mechanism for selective forgetting or controlled update. This is the conceptual reason why gated architectures such as LSTMs and GRUs are needed: they aim to preserve stable propagation while also managing memory content.

10 Beyond gradients: forward information propagation

Gradient analysis studies the *backward* flow of learning signals. But propagation problems also exist in the *forward* direction. Even if training were numerically stable, the model would still fail at long-term reasoning if information about a past input faded out of the hidden state as time progressed.

This suggests another quantity of interest: sensitivity of the current memory to a past input. Formally, for an input received at time ℓ and a state observed at time $t > \ell$, we study

$$\frac{\partial h_t}{\partial x_\ell}.$$

Applying the chain rule gives

$$\frac{\partial h_t}{\partial x_\ell} = \frac{\partial h_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial h_{t-2}} \cdots \frac{\partial h_{\ell+1}}{\partial h_\ell} \frac{\partial h_\ell}{\partial x_\ell}.$$

Thus

$$\frac{\partial h_t}{\partial x_\ell} = \left(\prod_{i=\ell+1}^{t} J_i \right) \frac{\partial h_\ell}{\partial x_\ell},$$

where $J_i = \frac{\partial h_i}{\partial h_{i-1}}$ is the recurrent Jacobian.

This is mathematically the same structure as in backward gradient propagation. Consequently, the same spectral logic applies:

- singular values < 1 imply contraction of forward signals, so information fades,
- singular values > 1 imply amplification and instability,
- singular values ≈ 1 support stable information transmission.

11 Memory quality, stability, and dissipation

The hidden state of an RNN can be viewed as a memory system. A good memory should be sensitive enough to preserve meaningful traces of past inputs, but not so unstable that tiny perturbations explode into uncontrollable dynamics.

This leads to two undesirable regimes.

11.1 Dissipative memory

If the recurrent dynamics are strongly contractive, then small differences in the past are rapidly forgotten. The state becomes insensitive to older inputs:

$$\left\| \frac{\partial h_t}{\partial x_\ell} \right\| \to 0 \qquad \text{as } t - \ell \to \infty.$$

This is a memory that leaks information too quickly.

11.2 Unstable memory

If the recurrent dynamics amplify perturbations, then tiny changes in the past can lead to large deviations in the state:

$$\left\| \frac{\partial h_t}{\partial x_\ell} \right\| \to \infty \text{ or becomes erratic.}$$

This is not useful memory either, because representations become fragile and difficult to control.

The desirable regime lies between these extremes: memory should preserve information in a stable way without either dissipating it or amplifying noise.

Worked example — Forward memory sensitivity and Jacobian products

Let

$$h_{t+1} = \phi(W_h h_t + W_{\text{in}} x_{t+1} + b_h).$$

Then

$$\frac{\partial h_t}{\partial x_\ell} = \left(\prod_{i=\ell+1}^{t} D_i W_h \right) \frac{\partial h_\ell}{\partial x_\ell}.$$

Take norms:

$$\left\| \frac{\partial h_t}{\partial x_\ell} \right\| \leq \left(\prod_{i=\ell+1}^{t} \|D_i\| \, \|W_h\| \right) \left\| \frac{\partial h_\ell}{\partial x_\ell} \right\|.$$

If the dominant singular values of the Jacobians are below one, then this norm decays with the time gap $t - \ell$. Hence the present state becomes progressively less informative about past inputs. This formalizes the intuitive notion of a "leaky" neural memory.

12 Spectral viewpoint on memory capacity

The repeated-Jacobian perspective suggests a unifying message: both trainability and memory retention are governed by the spectrum of the recurrent dynamics.

In a local linear approximation, the hidden-state dynamics are controlled by the recurrent matrix and the activation derivatives. Therefore:

- **gradient behavior** depends on how backward Jacobian products evolve,
- **memory behavior** depends on how forward Jacobian products evolve.

In both cases, the decisive quantities are singular values and spectral radii.

This spectral viewpoint is powerful because it links deep learning to classical dynamical-systems ideas such as stability, contraction, and dissipation. An RNN is not just a function approximator; it is a learned discrete-time dynamical system. Its success depends on whether its dynamics occupy the right regime for the task.

13 Implications for recurrent architecture design

The analysis above explains why vanilla RNNs are often inadequate for long-range dependencies and why more sophisticated architectures were introduced.

Three broad design principles follow naturally:

1. **Control the Jacobian spectrum.** Orthogonal or unitary parametrizations, identity-like initializations, and careful recurrent design aim to keep propagation close to norm-preserving.
2. **Avoid permanently contractive nonlinearities.** If the activation derivative is always much smaller than one, long-range gradients disappear almost inevitably.
3. **Separate memory preservation from memory update.** The naive identity-recurrence solution preserves gradients but has poor memory management. Gated models solve this by learning when to write, keep, or forget information.

This is the conceptual bridge to gated recurrent architectures such as LSTMs and GRUs, and also to other approaches that explicitly shape memory dynamics.