Information Propagation in Deep Networks

Generative and Deep Learning (GDL)

Davide Bacciu (davide.bacciu@unipi.it)

UNIVERSITÀ DI PISA

Objectives

- Introduction to sequential data processing
- (Vanilla) Recurrent neural networks RNNs refresher
- Issues with information propagation in RNNs: the exploding and vanishing gradient problem (EVGP)
- Dissecting the causes of gradient vanish in RNNs
- Preliminary directions on how to solve the EVGP
- Analyzing information propagation beyond the gradient

Sequential data processing

Sequential data

A sequence of observations following a complete ordering, where an observation needs to be interpreted in the context of the predecessors

Timeseries

- Observations from processes that evolve in time
- E.g.: healthcare, finance, meteorology, robotics, ICT systems

Sequences

- Information strings that can be read left-to-right or right-to-left (or both)
- E.g.: natural and artificial languages, genomic data, proteomic data

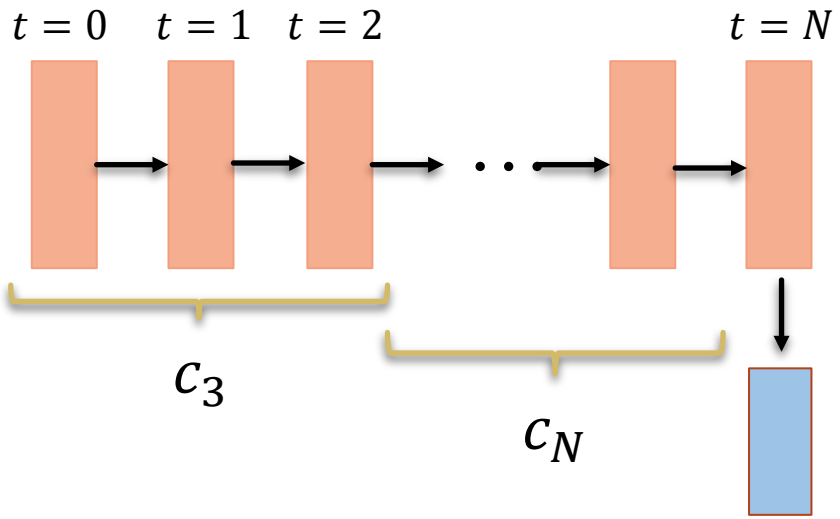
Formalization

- A timeseries $\boldsymbol{x}$ is a sequence of ordered measurements

$$\boldsymbol{x} = x_0, x_1, \dots, x_t, \dots, x_N$$

- Where x_t (or $x(t)$) is the measurement at time t
  - Timeseries observations can be observable at irregular time intervals
- Sequential data is defined similarly, only x_t defines the data in position t in the sequence
- Irrespectively of the type (sequential or timeseries), the single observation x_t can be complex at wish (i.e. a scalar, an image, a graph, …)

Dealing with Sequences in NN



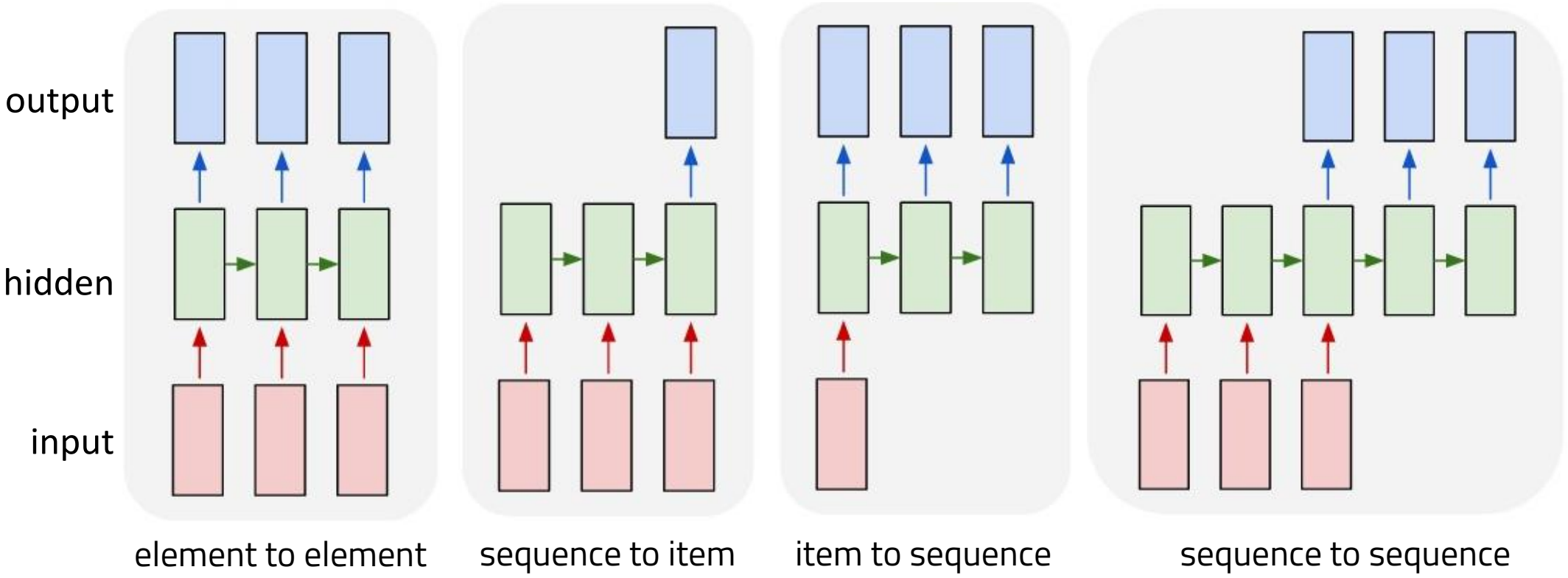
Variable size data describing sequentially dependent information

Neural models need to capture dynamic context c_t to perform predictions

- Recurrent Neural Networks (RNNs) compress the context c_t into a vector of neural activations: the neural network memory/state h_t
- RNN flavors
  - Vanilla: Elman, SRN, …
  - Randomized approaches
  - Gated recurrent networks

Graphics credit
@ karpathy.github.io

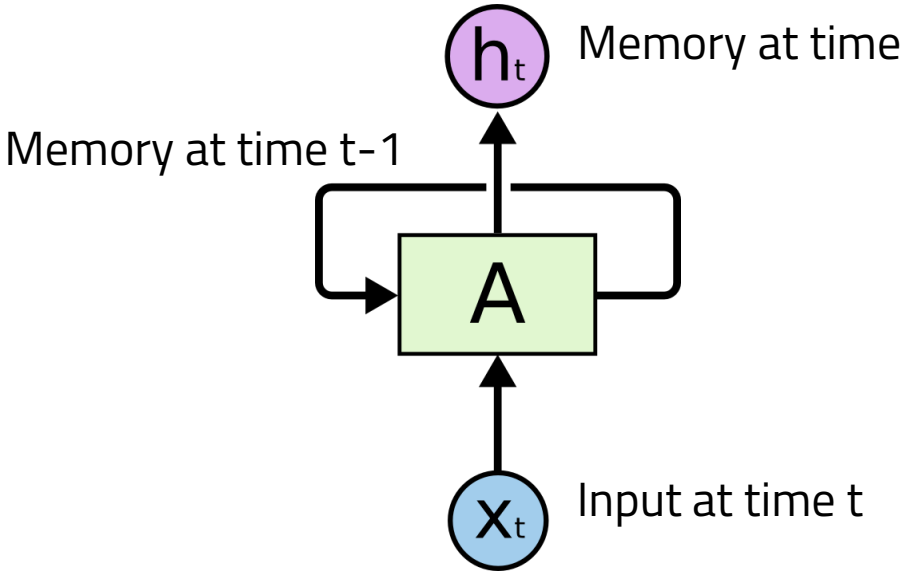
Supervised Tasks on Sequences

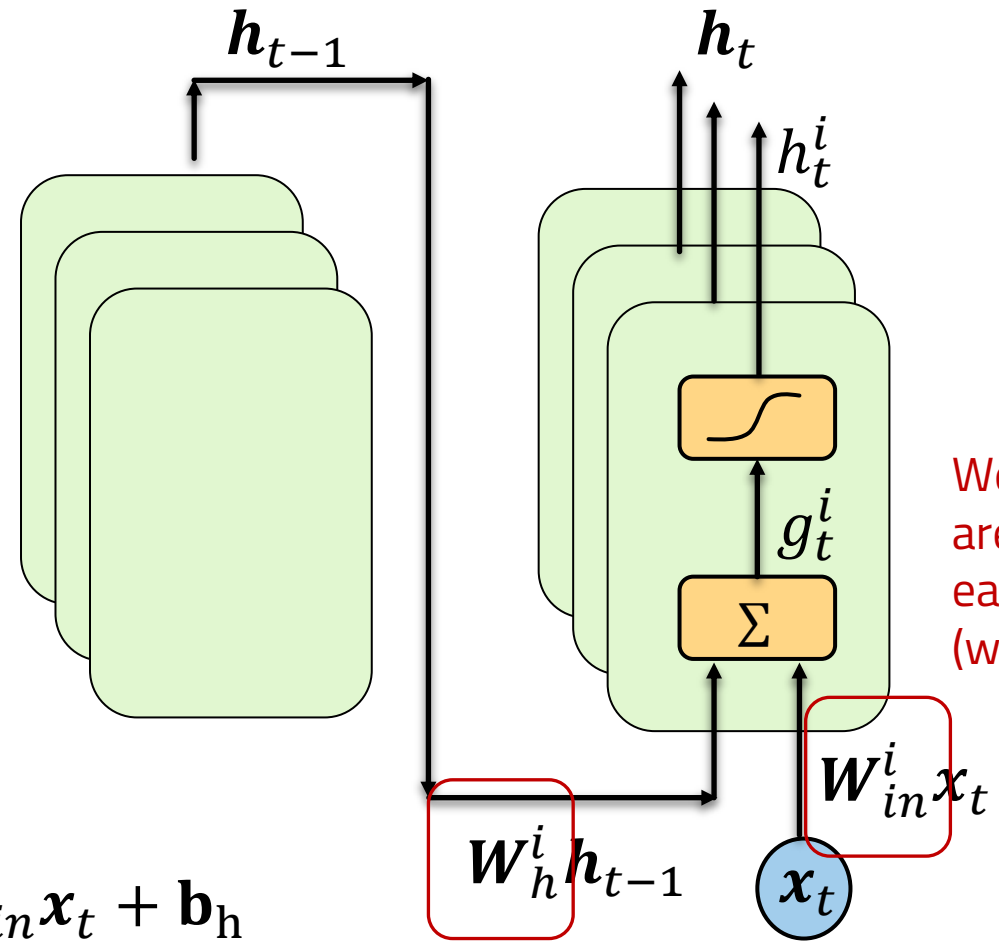


RNNs refresher

A Vanilla RNN

RNN output $\boldsymbol{y}_t = f(\boldsymbol{W}_{out}\boldsymbol{h}_t + \boldsymbol{b}_{out})$





Weight matrices are reused at each time t (weight sharing)

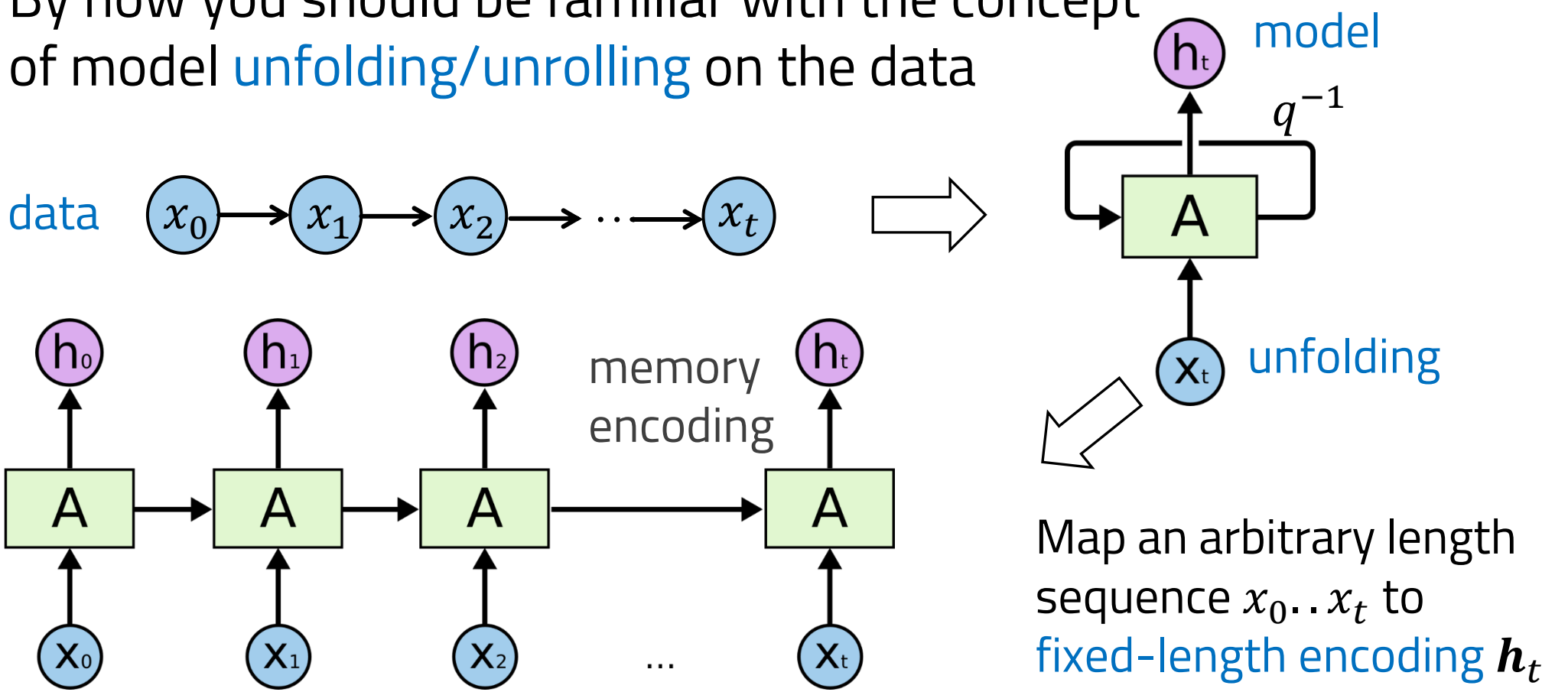
$$\boldsymbol{h}_t = tanh(\boldsymbol{g}_t)$$

$$\boldsymbol{g}_t(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t) = \boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_{in}\boldsymbol{x}_t + \mathbf{b}_\mathrm{h}$$

Graphics credit
@ colah.github.io

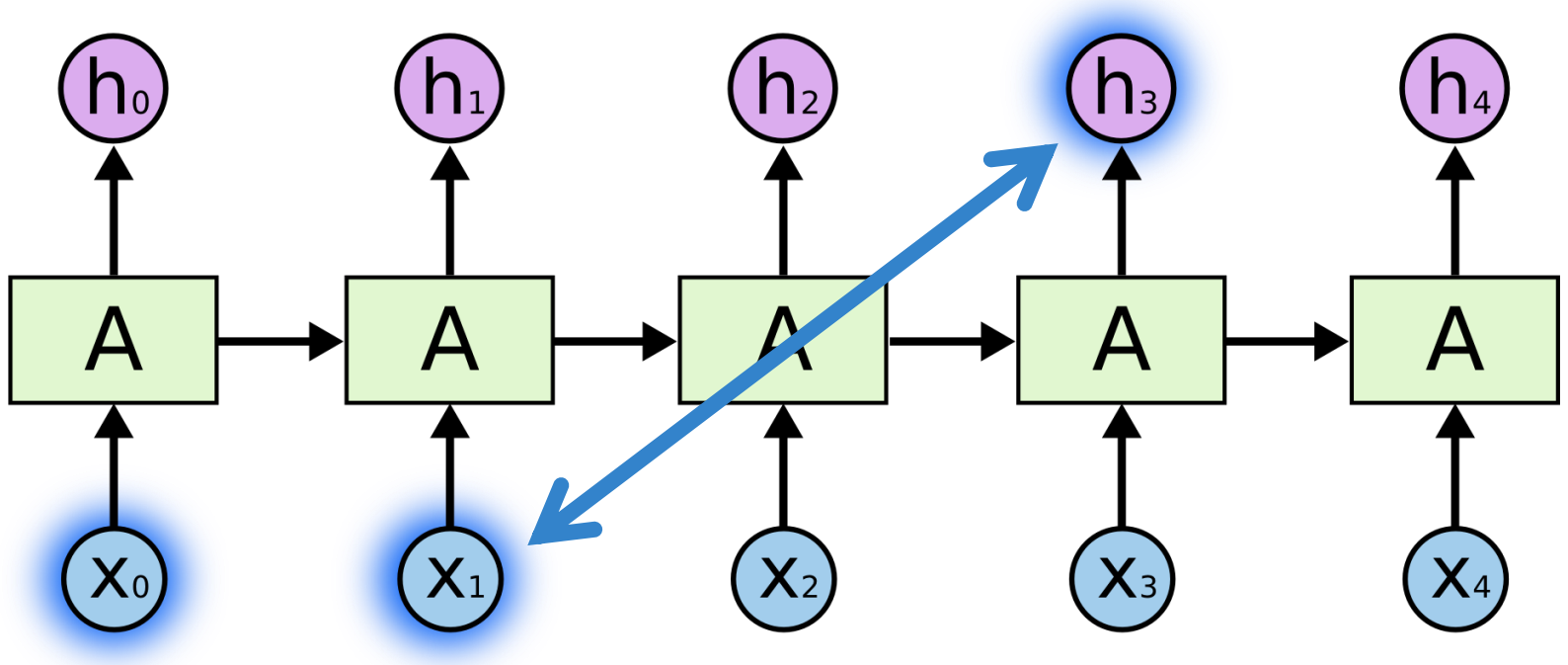
Unfolding RNN (Forward Pass)

By now you should be familiar with the concept of model unfolding/unrolling on the data



Map an arbitrary length sequence $x_0..x_t$ to fixed-length encoding $\boldsymbol{h}_t$

Learning to Encode Input History

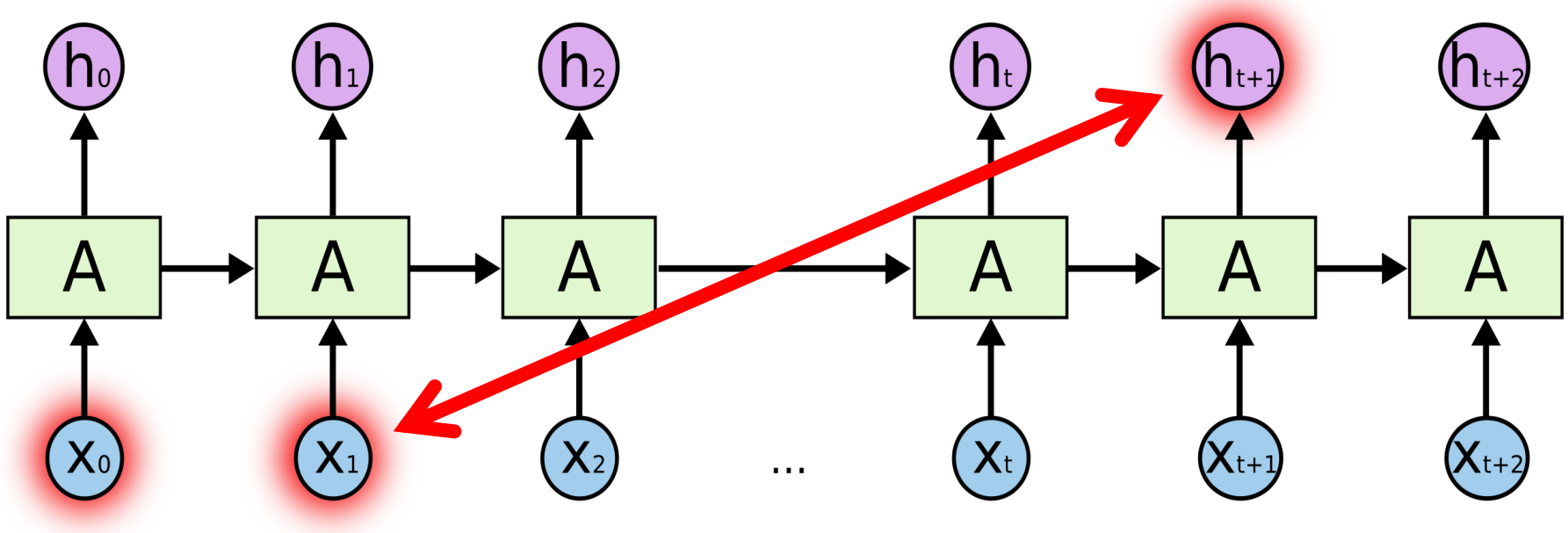


Hidden state $\boldsymbol{h}_t$ summarizes information on the history of the input signal up to time t

The Issue

Learning Long-Term Dependencies is Difficult

It is difficult to assign credit to an input a time l for an error generated a time t when $t - l$ is large



J. Hochreiter. Untersuchungen zu dynamischen neuronalen Netzen, TUM, 1991

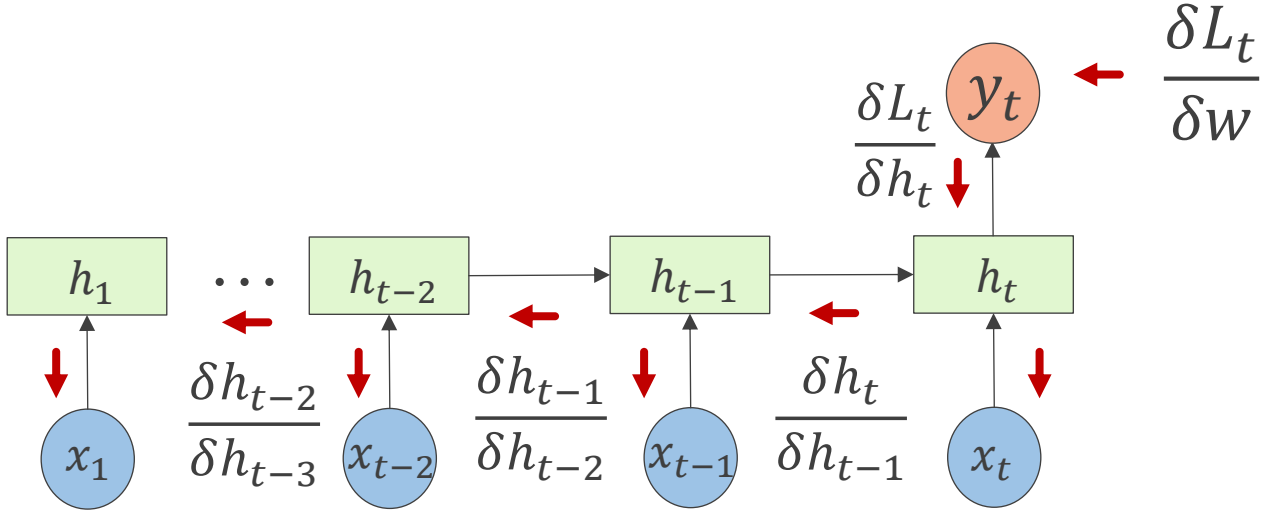
What is the cause?

Exploding/Vanishing Gradient

Short story: gradients propagated over many stages tend to

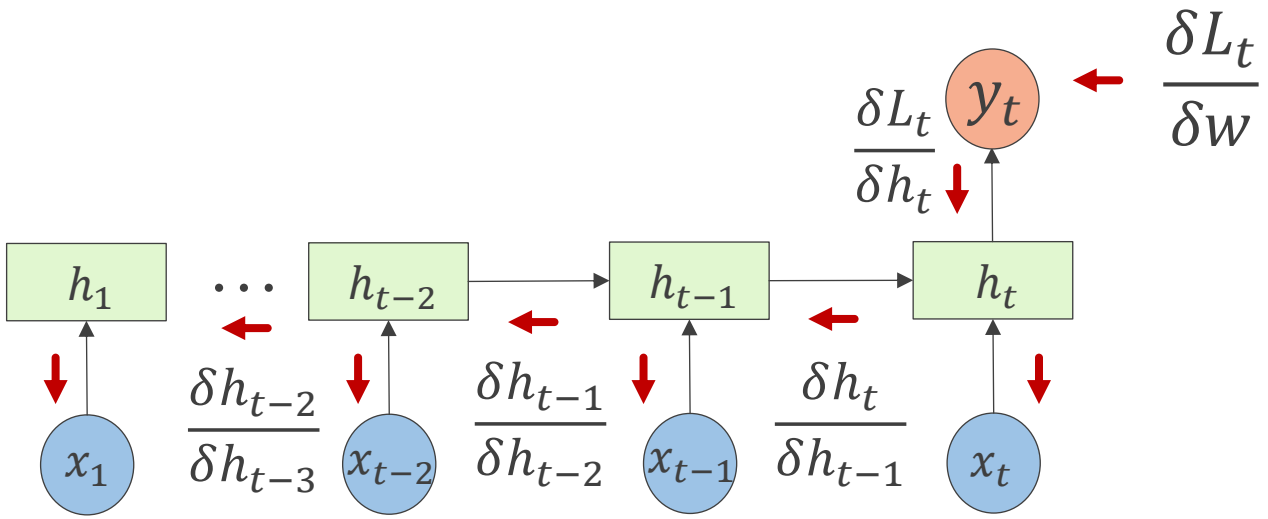
- Vanish (often) $\Longrightarrow$ No learning
- Explode (rarely) $\Longrightarrow$ Instability and oscillations

Weights are shared between time steps $\Longrightarrow$ sum gradient contributions through time



Bengio, Simard and Frasconi, Learning long-term dependencies with gradient descent is difficult. TNN, 1994

Backward propagation



A Closer Look at the Gradient

$$\frac{\delta L_t}{\delta \boldsymbol{W}} = \sum_{k=1}^{t} \frac{\delta L_t}{\delta h_t} \frac{\delta h_t}{\delta h_k} \frac{\delta h_k}{\delta \boldsymbol{W}}$$

This is a parameter matrix
$\Longrightarrow$ we have a Jacobian

Inside here you have the chain rule

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

$$\frac{\delta L_t}{\delta \boldsymbol{W}} = \sum_{k=1}^{t} \frac{\delta L_t}{\delta h_t} \left(\prod_{l=k}^{t-1} \frac{\delta h_{l+1}}{\delta h_l} \right) \frac{\delta h_k}{\delta \boldsymbol{W}}$$

The gradient is a recursive product of hidden activation gradients

The Jacobian

- Given a vectorial function $f(\boldsymbol{x})$, i.e. the activations of a layer of m neurons, the Jacobian of f at $\boldsymbol{x}$ is the matrix of all first-order partial derivatives w.r.t. the n variables

$$J_f(\boldsymbol{x}) = \frac{\partial f}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

- In our setting $\boldsymbol{h}_{t+1} = f(\boldsymbol{h}_t)$, hence

$$J_f(\boldsymbol{h}_t) = \frac{\delta f}{\delta \boldsymbol{h}_t} = \boldsymbol{D}_t \boldsymbol{W}_h^T$$

where $\boldsymbol{D}_t$ is a diagonal matrix holding the derivatives of the activation function

Bounding the Gradient (I)

Given $\boldsymbol{h}_l = tanh(\boldsymbol{W}_{hl}\boldsymbol{h}_{l-1} + \boldsymbol{W}_{in}\boldsymbol{x}_l)$ then $\frac{\delta \boldsymbol{h}_{l+1}}{\delta \boldsymbol{h}_l} = \boldsymbol{D}_{l+1}\boldsymbol{W}_{hl}^T$ where the activation Jacobian is

$$\boldsymbol{D}_{l+1} = diag(1 - \tanh^2(\boldsymbol{W}_{hl}\boldsymbol{h}_l + \boldsymbol{W}_{in}\boldsymbol{x}_{l+1}))$$

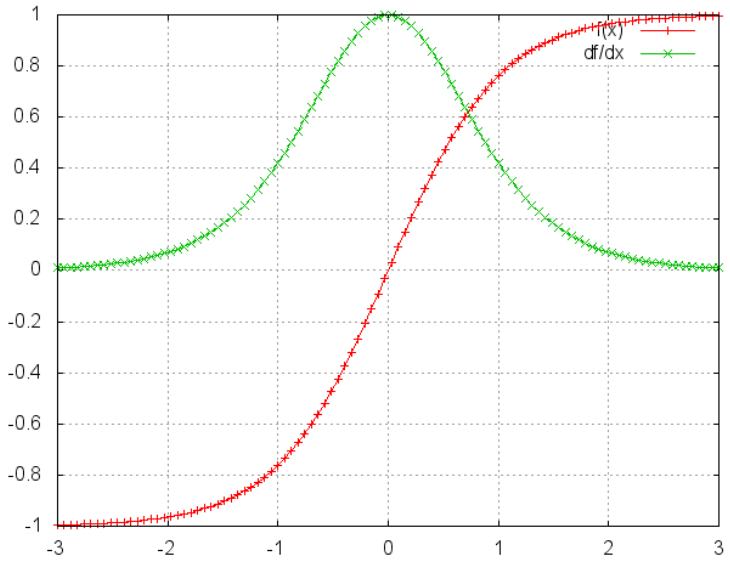
$$\frac{\delta L_t}{\delta \boldsymbol{h}_k} = \frac{\delta L_t}{\delta \boldsymbol{h}_t}\left(\prod_{l=k}^{t-1}\frac{\delta \boldsymbol{h}_{l+1}}{\delta \boldsymbol{h}_l}\right) = \frac{\delta L_t}{\delta \boldsymbol{h}_t}\prod_{l=k}^{t-1}\boldsymbol{D}_{l+1}\boldsymbol{W}_{hl}^T$$

We are interested in the gradient magnitude $\left\|\frac{\delta L_t}{\delta \boldsymbol{h}_k}\right\|$

Bounding the Gradient (II)

$$\left\|\frac{\delta L_t}{\delta \boldsymbol{h}_k}\right\| = \left\|\frac{\delta L_t}{\delta \boldsymbol{h}_t}\prod_{l=k}^{t-1}\boldsymbol{D}_{l+1}\boldsymbol{W}_{hl}^T\right\| \leq \left\|\frac{\delta L_t}{\delta \boldsymbol{h}_t}\right\|\prod_{l=k}^{t-1}\|\boldsymbol{D}_{l+1}\|\|\boldsymbol{W}_{hl}^T\| \approx \left\|\frac{\delta L_t}{\delta \boldsymbol{h}_t}\right\|\|\boldsymbol{D}\|^{k-1}\|\boldsymbol{W}_h^T\|^{k-1} \approx \left\|\frac{\delta L_t}{\delta \boldsymbol{h}_t}\right\|\rho(\boldsymbol{D})^{k-1}\rho\left(\boldsymbol{W}_h^T\right)^{k-1}$$

Bounded by the spectral radius ρ for some norm and k large enough



Can shrink to zero or increase exponentially depending on the spectral properties

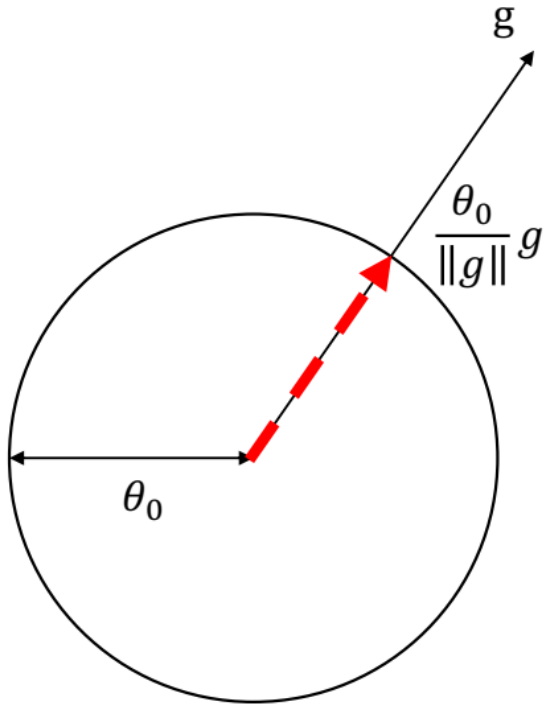
- $\rho < 1 \Longrightarrow$ vanishish
- $\rho > 1 \Longrightarrow$ exploding

Addressing the EVGP

Gradient Clipping for Exploding Gradients

- Take $g = \frac{\delta L_t}{\delta \boldsymbol{W}}$
- If $\|g\| > \theta_0$ then $g = \frac{\theta_0}{\|g\|} g$

Rescaling does not work for gradient vanish as total gradient is a sum of time dependent gradients (preserving relative contribution from each time makes it exponentially decay)



$$\frac{\delta L_t}{\delta \boldsymbol{W}} = \sum_{k=1}^{t} \frac{\delta L_t}{\delta h_t} \frac{\delta h_t}{\delta h_k} \dots$$

Pursuing constant error propagation

Solution seems to be having the Jacobian ensuring that $\rho = 1$

- Select an appropriate activation function (A)
- Choose a recurrent weight matrix with the right spectral properties (B)

$$\frac{\delta \boldsymbol{h}_{l+1}}{\delta \boldsymbol{h}_l} = \boldsymbol{D}_{l+1} \boldsymbol{W}_h^T$$

(A) Activation Function

- Popular choices (sigmoid, tanh) are always contractive ($\rho < 1$)
- Alternatives: modReLU
- Much simpler alternative: no activation function (identity)

Tanh activation

$$\boldsymbol{h}_{l+1} = \tanh\left(\boldsymbol{W}_h^T \boldsymbol{h}_l + \boldsymbol{W}\boldsymbol{x}_{l+1}\right)$$

$$\frac{\delta \boldsymbol{h}_{l+1}}{\delta \boldsymbol{h}_l} = \boldsymbol{D}_{l+1}\boldsymbol{W}_h^T$$

Linear activation

$$\boldsymbol{h}_{l+1} = \boldsymbol{W}_h^T \boldsymbol{h}_l + \boldsymbol{W}\boldsymbol{x}_{l+1}$$

$$\frac{\delta \boldsymbol{h}_{l+1}}{\delta \boldsymbol{h}_l} = \boldsymbol{I}\boldsymbol{W}_h^T = \boldsymbol{W}_h^T$$

(B) Recurrent Weights

It is possible to achieve $\rho = 1$

- Orthogonal matrices: $\boldsymbol{W}^T\boldsymbol{W} = \boldsymbol{I}$
- Unitary matrices (complex domain): $\boldsymbol{W}^H\boldsymbol{W} = \boldsymbol{I}$
- Identity matrix: $\boldsymbol{W} = \boldsymbol{I}$

Orthogonal Matrix + linear activation:

$$\boldsymbol{h}_{l+1} = \boldsymbol{W}_h^T\boldsymbol{h}_l + \boldsymbol{W}\boldsymbol{x}_{l+1}$$

$$\frac{\delta \boldsymbol{h}_{l+1}}{\delta \boldsymbol{h}_l} = \boldsymbol{I}\boldsymbol{W}_h^T = \boldsymbol{W}_h^T$$

$$\left\|\frac{\delta \boldsymbol{h}_{l+1}}{\delta \boldsymbol{h}_l}\right\| = \|\boldsymbol{I}\boldsymbol{W}_h^T\| = 1$$

Constant Error Propagation

Lets now set

- Identity activation function
- Identity weight matrix

$$\boldsymbol{h}_t = \boldsymbol{h}_{t-1} + \hat{c}(\boldsymbol{x}_t)$$

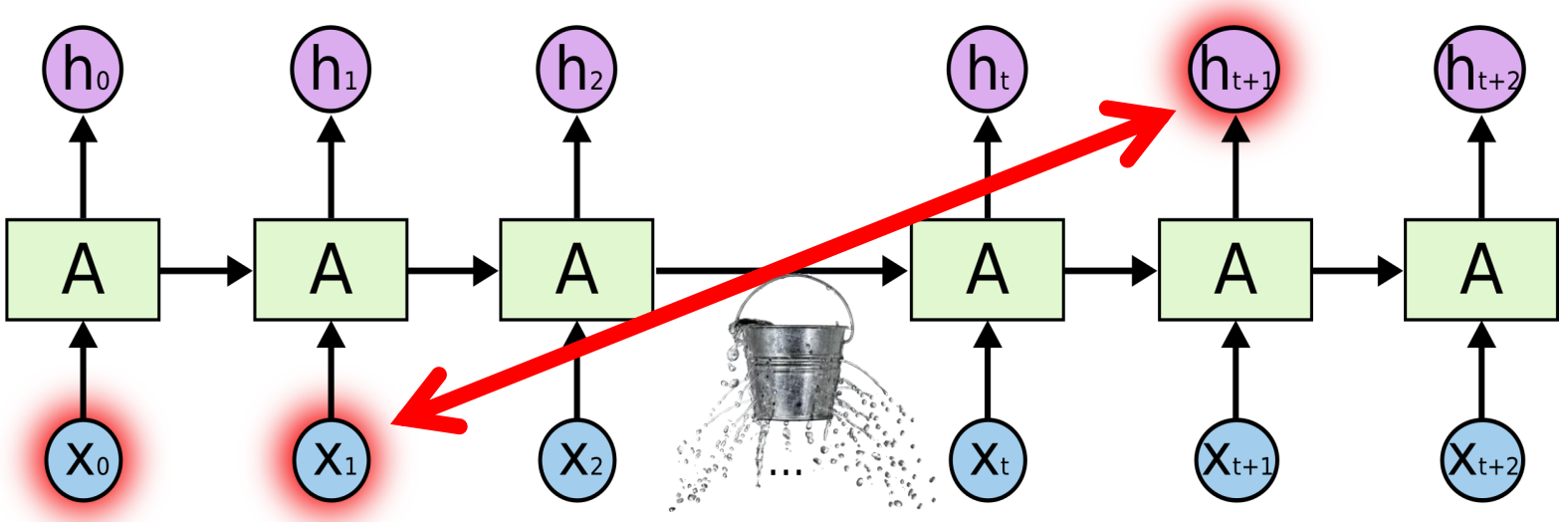
The formulation above has the desired spectral properties but does not work in practice as it quickly saturates memory (e.g. with replicated/non-useful inputs and states).

⇒ We want to be able to "control the forgetting" (**next lecture**)

A broader view

Looking at propagation problems in the forward direction

When the time gap between the observation and the state grows there may be little residual information of the input inside of the memory



How can we study the efficiency of the propagation/retainment of information in memory?

Measuring the quality of neural memory

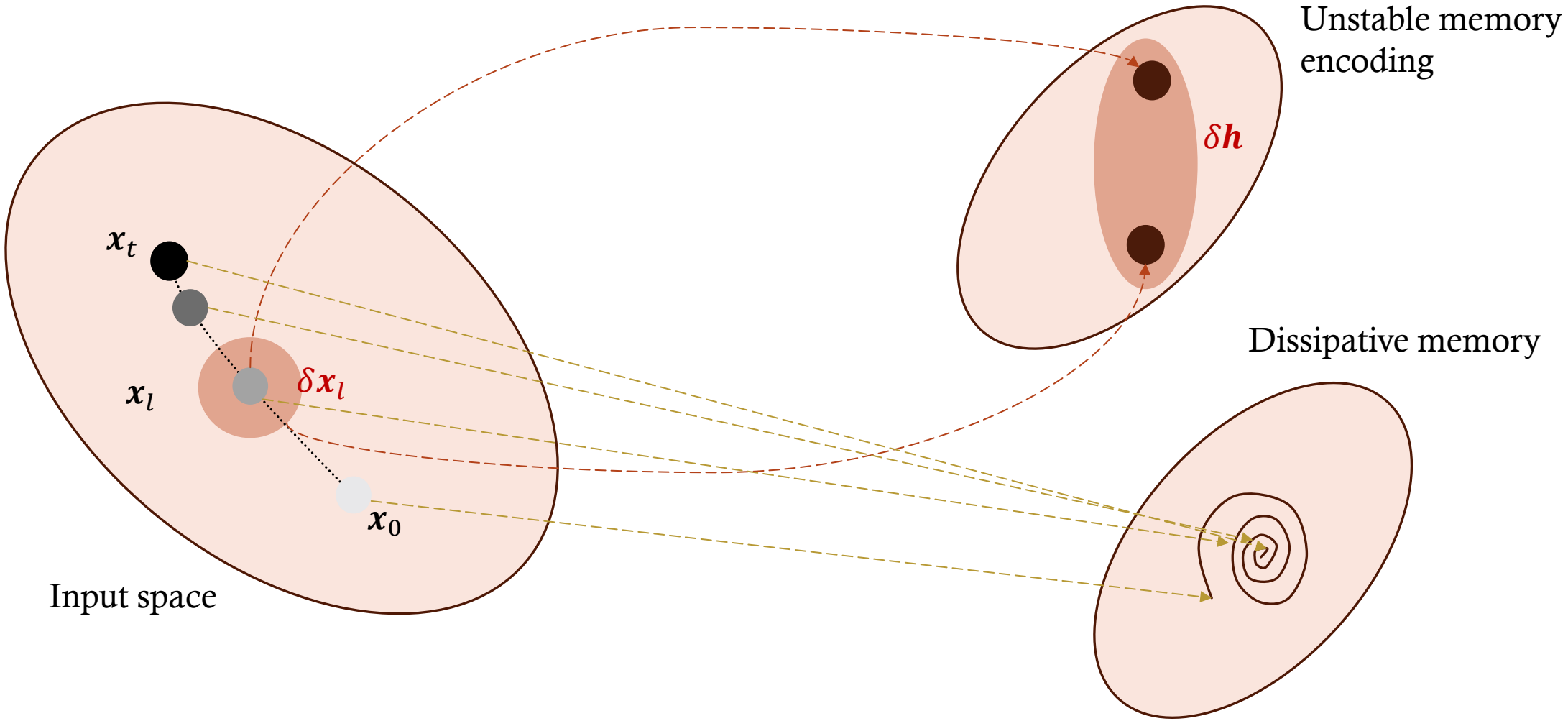
- Memory efficiency can be related to sensitivity analysis
- The ability to retain traces in $\boldsymbol{h}_t$ of small changes in some input information $\boldsymbol{x}_l$ at time l
- More formally, we study the behaviour of $\left\|\frac{\delta \boldsymbol{h}_t}{\delta \boldsymbol{x}_l}\right\|$ as $t - l$ grows
- Guess who is back?

$$\left\|\frac{\delta \boldsymbol{h}_t}{\delta \boldsymbol{x}_l}\right\| = \left\|\frac{\delta \boldsymbol{h}_t}{\delta \boldsymbol{h}_{t-1}}\frac{\delta \boldsymbol{h}_{t-1}}{\delta \boldsymbol{x}_l}\right\| = \left\|\frac{\delta \boldsymbol{h}_t}{\delta \boldsymbol{h}_{t-1}}\frac{\delta \boldsymbol{h}_{t-1}}{\delta \boldsymbol{h}_{t-2}}\cdots\frac{\delta \boldsymbol{h}_l}{\delta \boldsymbol{x}_l}\right\| = \left\|\prod_{i=l+1}^{t}\frac{\delta \boldsymbol{h}_i}{\delta \boldsymbol{h}_{i-1}}\frac{\delta \boldsymbol{h}_l}{\delta \boldsymbol{x}_l}\right\| = \left\|\prod_{i=l+1}^{t}\boldsymbol{J}_i\frac{\delta \boldsymbol{h}_l}{\delta \boldsymbol{x}_l}\right\|$$

Our Jacobian friend **J**

- Since we are interested in the norms, memory sensitivity is regulated again by the spectral properties of the RNN Jacobian
  - It is not by chance that these properties connect with stability and dissipation in physical systems

On the effects of stability and dissipation



Jacobian Spectrum and Memory Capacity

- The behavior of sensitivity depends on the singular values of $\boldsymbol{J}_i$.

$$\frac{\delta \boldsymbol{h}_t}{\delta \boldsymbol{x}_l} = \prod_{i=l+1}^{t} \boldsymbol{J}_i \frac{\delta \boldsymbol{h}_l}{\delta \boldsymbol{x}_l}$$

- If singular values
  - $< 1 \rightarrow$ forward signals contract $\rightarrow$ information fades
  - $> 1 \rightarrow$ forward signals amplify $\rightarrow$ instability
  - $\approx 1 \rightarrow$ stable information propagation
- When analysed in the linear regime, the behaviour above can be controlled via the spectral properties of the recurrent weight matrix $\rho(\boldsymbol{W}_h)$
  - We will see how we can leverage this intuition to design efficient RNNs (**next lecture**)

Wrap-Up

Take home messages

- Sequential data are compound ordered information
  - Recurrent neural models are based on weight sharing (time stationarity)
- Learning long-term dependencies can be difficult due to gradient vanish/explosion
  - Produced by numerical issues in repeated multiplications
  - Vanishing and instability linked to spectral properties of the activation Jacobian
  - Appropriate choices of the activation functions and recurrent weights can control the issue
- Naïve solution uses identity function and weights in the recurrence
  - Representation power issues
  - Memory management issues
- Propagation issues do not affect only the backward pass: they produce leaky/unstable neural memories

Next Lecture

RNNs addressing/controlling gradient propagation issues

- Gated approaches
  - Gating neurons
  - Long-Short Term Memories (LSTM)
  - Gated Recurrent Units (GRU)
- Randomized approaches
  - Controlling memory properties alone
  - Echo state network
- Advanced topics
  - Autoregressive modelling with RNNs
  - Advanced RNN architectures