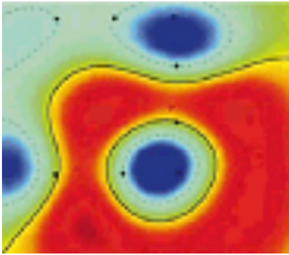
WiSe 2024/25

Deep Learning 1

Technische Universität Berlin



Lecture 9

Recurrent Neural Networks

Outline

Stateless vs. Stateful Models

- Characterization
- Examples

Recurrent Neural Networks (RNNs)

- General formulation of a RNN
- Examples of practical RNNs
  (e.g. standard, bidirectional, encoder-decoder)
- Choosing the initial state

The Difficulty of Training RNNs

- The vanishing/exploding gradient problem

LSTM Architecture for RNNs

Applications of RNNs

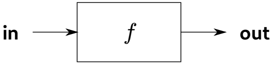
Part 1

Stateless vs. Stateful Models

Stateless vs. Stateful Models

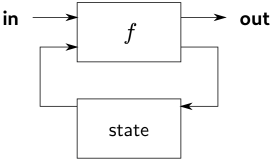
Stateless Predictor

- The prediction is simply a function of the data given as input.
- The data given as input could be e.g. an simple vector of measurement, or a sequence of such vectors (a time series).



Stateful Predictor

- The prediction is a function that takes produces a prediction from the input and the current state of the system. The function also outputs the future state of the system.



Stateless vs. Stateful Models

Example: Stateless model for moving average

- Equation:
$$y_t = \alpha \cdot x_t + \beta x_{t-1} + \gamma x_{t-2}$$
- This model can be interpreted as a sliding window through the input sequence, and has a finite horizon.
- Assuming an input time series $(x_1, x_2, \dots, x_T)$, values $y_3, y_4, \dots$ can be predicted directly.

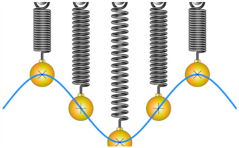
Example: Stateful model for moving average

- Equation:
$$\begin{bmatrix} y_t \\ h_t \end{bmatrix} = \begin{bmatrix} h_{t-1} \\ \gamma h_{t-1} + (1-\gamma) x_t \end{bmatrix}$$
- This model has an infinite horizon.
- Assuming an input time series $(x_1, x_2, \dots, x_T)$ one needs to specify an initial state h_0 to compute any of the predicted values y_t.
- Potentially less parameters to tune.

Representing an Oscillator

表示振荡器

- 一个谐振子让粒子以加速度运动，该加速度与粒子偏离原点的距离成正比，但方向相反：



- An harmonic oscillator lets a particle accelerate in a way that is proportional but of opposite sign to its offset from the origin:

$$\overbrace{\underbrace{(r_t - r_{t-1})}_{\approx v_t} - \underbrace{(r_{t-1} - r_{t-2})}_{\approx v_{t-1}}}^{\approx a_t} = -\gamma r_t$$

- The same equation can be rewritten as:

$$r_t = 2\beta r_{t-1} - \beta r_{t-2}$$

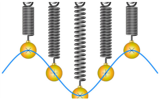
where $\beta = 1/(1+\gamma)$, and defining the states as:

$$\boldsymbol{h}_t = \begin{bmatrix} r_t \\ r_{t-1} \end{bmatrix} \quad \boldsymbol{h}_{t-1} = \begin{bmatrix} r_{t-1} \\ r_{t-2} \end{bmatrix} \quad \ldots$$

we get the following equations update equation:

$$\boldsymbol{h}_t = \begin{pmatrix} 2\beta & -\beta \\ 1 & 0 \end{pmatrix} \cdot \boldsymbol{h}_{t-1}$$

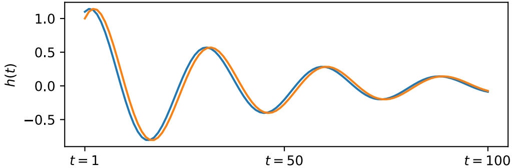
Representing an Oscillator



- The stateful model

$$\boldsymbol{h}_t = \begin{pmatrix} 2\beta & -\beta \\ 1 & 0 \end{pmatrix} \cdot \boldsymbol{h}_{t-1}$$

with initial conditions $\boldsymbol{h}_1 = [1.1, 1.0]$ and with parameter $\gamma = 0.05$ gives the sequence of states:



from which we can clearly see the oscillating effect.

- Note that it is not a perfect oscillator. The dampening effect is caused by the crude numerical approximation of the velocity v_t and acceleration a_t (cf. previous slide).

Oscillator with External Force (Input)

- Let us modify the original oscillator as:

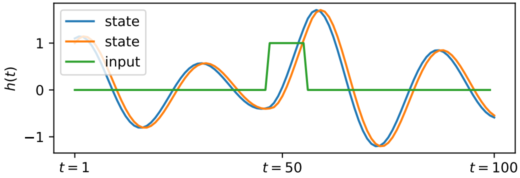
$$\overbrace{\underbrace{(r_t - r_{t-1})}_{\approx v_t} - \underbrace{(r_{t-1} - r_{t-2})}_{\approx v_{t-1}}}^{\approx a_t} = -\gamma r_t + \gamma x_t$$

  where x_t can be interpreted as an external force. The model becomes:

$$\boldsymbol{h}_t = \begin{pmatrix} \alpha & 2\beta & -\beta \\ 0 & 1 & 0 \end{pmatrix} \cdot \begin{bmatrix} x_t \\ \boldsymbol{h}_{t-1} \end{bmatrix}$$

  where $\alpha = \gamma/(1-\gamma)$ and $\beta = 1/(1+\gamma)$.

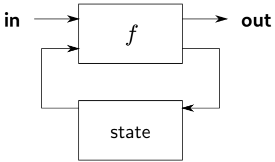
- With the same initial condition and parameters as before, and input time series shown below, we get:



Stateful Models

Stateful Models

- Can be used to induce complex internal dynamics (averaging, dampening, oscillations).
- These dynamics serve as a useful prior for a broad range of prediction tasks involving time series.



Questions:

- Can the equations in the previous slide be generalized to contain a richer set of dynamics?
- Can these equations be learned form the data?

有状态模型（Stateful Models）

- 可以用于引入复杂的内部动态（平均、阻尼、振荡）。
- 这些动态作为一种有用的先验知识，适用于涉及时间序列的广泛预测任务。

问题（Questions）：

- 先前幻灯片中的方程是否可以被推广，以包含更丰富的动态?
- 这些方程是否可以从数据中学习到?

Part 2

Recurrent Neural Networks

Towards a General Formulation: RNNs

矩阵 A, B, C, D 可以从数据中学习，例如通过最小化输出时间序列 y 和某些真实时间序列 t 之间的偏差。

- 矩阵可以参数化或正则化，以引导系统动态朝向特定行为，例如系统状态的缓慢演化、阻尼、振荡等。
- 上述模型可以进一步推广为：

- The model studied above can be generalized by the equation:

$$\begin{bmatrix} \boldsymbol{y}_t \\ \boldsymbol{h}_t \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{h}_{t-1} \end{bmatrix}.$$

  The matrices A, B, C, D can be learned from the data, e.g. to minimize the divergence between the output time series $\boldsymbol{y}$ and some ground-truth time series $\boldsymbol{t}$.

- Matrices can be parameterized or regularized in a way that they steer the system dynamics towards a specific behavior (e.g. slow evolution of the system's state, dampening, oscillations, etc.).
- The model above can further generalized to:

$$\begin{bmatrix} \boldsymbol{y}_t \\ \boldsymbol{h}_t \end{bmatrix} = f_\theta\left(\begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{h}_{t-1} \end{bmatrix}\right)$$

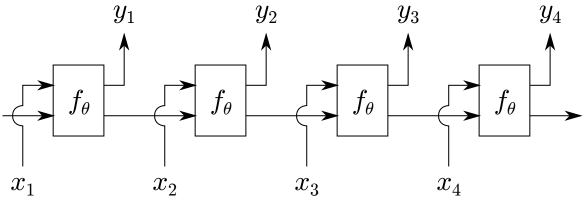
  where f_θ can be any function, e.g. a neural network, with a set of parameters θ to be learned.

RNN Visualization

The RNN defined by the equation:

$$\begin{bmatrix} \boldsymbol{x}_t \\ \boldsymbol{h}_{t-1} \end{bmatrix} \xmapsto{f_\theta} \begin{bmatrix} \boldsymbol{y}_t \\ \boldsymbol{h}_t \end{bmatrix}$$

can be visualized as:

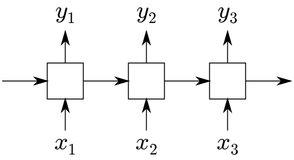


Observation:

- A RNN can be seen as a big neural network composed of a large number of sub neural networks with shared parameters. The whole architecture can be trained via backprop.
- The function f_θ is composed multiple times. If f_θ is a neural network of depth L, the RNN becomes a network of depth $L \cdot T$.

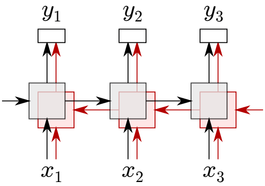
RNN Architectures for Sequence-to-Sequence

Standard (unidirectional) RNNs:



- Generate the output at the same time as the input is received $\rightarrow$ enable a strong coupling between the two sequences.
- Cannot use information about later time steps when generating the output sequence (problem for e.g. translation).

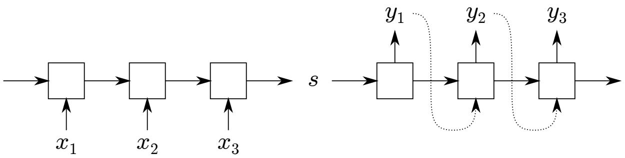
Bidirectional RNNs



- Add a RNN in reverse direction in order to incorporate information from future values in the sequence.

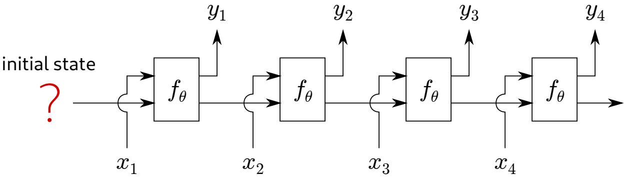
RNN Architectures

Encoder-Decoder RNNs:



- Instead of generating the output sequence at the same time as we process the input sequence, first create a global representation of the input sequence s, and then, generate the output sequence from s.
- This ability to read throught the whole sequence before generating is useful for tasks such as machine translation.
- However, the RNN architecture has fundamentally difficulties to retain long-term dependencies in a sequence. Attention-based / transformer architectures (Lecture 8) are often working better.

The Problem of Initial States



Problem:

- Unlike the input data, the RNN's initial state (at time $t = 0$) is not given and must be initialized to some value.

Possible approaches:

- Set it to some arbitrary value (e.g. $\boldsymbol{h}_0 = \mathbf{0}$).
- Set it at random (the RNN will then learn to desensitize itself to the initial state).
- Use one of the two approaches above **and** simulate the RNN for a few time steps in order to generate an intial state that is more plausible.

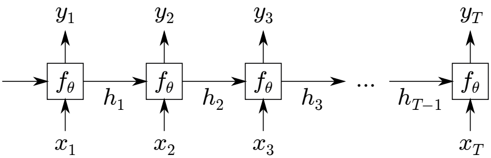
可能的方法（Possible approaches）：

- 将其设为某个任意值（例如 $h_0 = 0$）。
- 随机设定（RNN 将学习如何对初始状态去敏感化）。
- 采用上述两种方法之一 **并** 运行 RNN 若干时间步，以生成更合理的初始状态。

Part 3

Difficulty of RNN Training

RNN Optimization: Pathological Gradients
病理学的



The objective to optimize for a RNN is typically expressed as:

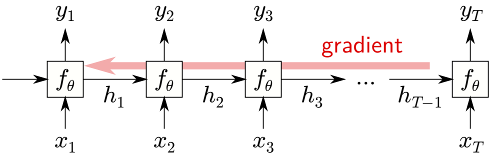
$$\mathcal{E} = \ell(y_1, t_1) + \cdots + \ell(y_T, t_T)$$

The gradient of the objective w.r.t. the parameter vector θ can be expressed via the chain rule:

$$\frac{\partial \mathcal{E}}{\partial \theta} = \sum_{t=1}^{T} \frac{\partial \mathcal{E}}{\partial y_t} \cdot \left(\frac{\partial^+ y_t}{\partial \theta} + \frac{\partial y_t}{\partial h_{t-1}} \frac{\partial h_{t-1}}{\partial \theta} \right)$$

$$\frac{\partial h_{t-1}}{\partial \theta} = \frac{\partial^+ h_{t-1}}{\partial \theta} + \sum_{s=2}^{t-1} \underbrace{\left(\prod_{i=s}^{t-1} \frac{\partial h_i}{\partial h_{i-1}} \right)}_{P_{s,t}} \frac{\partial^+ h_{s-1}}{\partial \theta}$$

RNN Optimization: Pathological Gradients



Observation:

- In the previous slide, we could express the error gradient $\partial\mathcal{E}/\partial\theta$ as a sum over indices $t = 1 \dots T$, and $s = 2 \dots t-1$, where each summand contains a product structure of the type.

$$P_{s,t} = \Big(\prod_{i=s}^{t-1} \frac{\partial h_i}{\partial h_{i-1}}\Big)$$

- On one extreme, the summand corresponding to indices $s = 2$ and $t = T$ features a very large product structure of $T-2$ terms.
- On the other extreme, for summands where $s = t-1$ the product structure totally vanishes (and just becomes an identity matrix I).

RNN Optimization: Pathological Gradients

Analysis for the Linear Model:

- Recall that the linear model is given by the equations:

$$\begin{bmatrix} y_i \\ h_i \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \cdot \begin{bmatrix} x_i \\ h_{i-1} \end{bmatrix}$$

- For such a model, the matrix $P_{s,t}$ can be computed in closed form:

$$P_{s,t} = \left(\prod_{i=s}^{t-1} \frac{\partial h_i}{\partial h_{i-1}} \right) = D^{t-s}$$

hence, $P_{2,T} = D^{T-2}$.

Eigenvalue Decomposition

If D is diagonalizable, the matrix can be rewritten as $D = Q \Lambda Q^{-1}$ with Λ containing the eigenvalues of D, then

$$D^2 = Q\Lambda \underbrace{Q^{-1}Q}_{I} \Lambda Q^{-1} = Q\Lambda^2 Q^{-1}$$

and after a few steps, $D^{T-2} = Q\Lambda^{T-2}Q^{-1}$.

RNN Optimization: Pathological Gradients

Two cases for the Linear RNNs:

$\max_k \lambda_k > 1$ The norm of the matrix D^{T-2} will keep increasing as T becomes large $\rightarrow$ gradients tend to explode.

$\max_k \lambda_k < 1$ The norm of the matrix D^{T-2} will keep decreasing as T becomes large $\rightarrow$ gradients tend to vanish.

In both cases, this creates a disbalance between the different terms of the gradient.

However, these results do not generalize exactly to nonlinear RNNs. We are left with heuristics, such as:

- Training the RNN with an appropriate level of noise applied to the hidden states so that the model has incentive to desentisize itself from the lower layers (thereby avoiding the exploding gradient problem).
- Choosing a particular class of functions for the RNN that is shown to be more robust to the vanishing/exploding gradient problem.

然而，这些结果并不能完全推广到非线性 RNNs（循环神经网络）。我们需要依赖一些启发式方法，例如：

- 在训练 RNN 时对隐藏状态施加适当程度的噪声，使得模型有动力从较低层的影响中去敏感化（从而避免梯度爆炸问题）。
- 为 RNN 选择某一特定类别的函数，该函数已被证明对梯度消失或梯度爆炸问题更具鲁棒性。

Part 4

Long Short-Term Memory

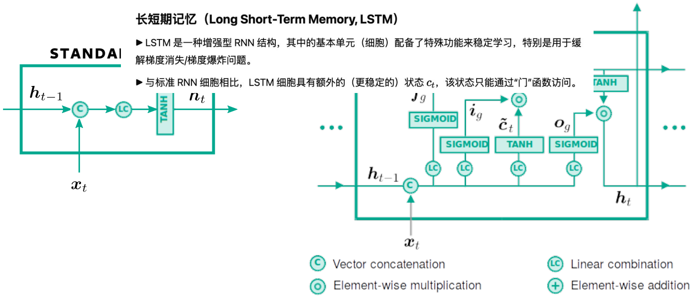
Long Short-Term Memory

长短期记忆（Long Short-Term Memory, LSTM）

▶ LSTM 是一种增强型 RNN 结构，其中的基本单元（细胞）配备了特殊功能来稳定学习，特别是用于缓解梯度消失/梯度爆炸问题。

▶ 与标准 RNN 细胞相比，LSTM 细胞具有额外的（更稳定的）状态 c_t，该状态只能通过"门"函数访问。

- The LSTM is an enhanced RNN architecture where the building blocks (cells) are equiped with special functions to stabilize learning, specifically, mitigate the vanishing/exploding gradient problem.
- The LSTM cell, in comparison to a standard RNN cell, has an additional (more stable) state $\boldsymbol{c}_t$, that is only accessed through 'gates' functions.



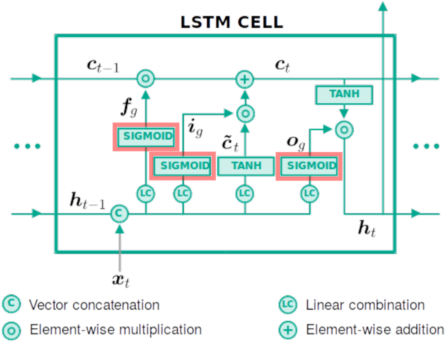
长短期记忆（Long Short-Term Memory, LSTM）

▶ LSTM 是一种增强型 RNN 结构，其中的基本单元（细胞）配备了特殊功能来稳定学习，特别是用于缓解梯度消失/梯度爆炸问题。

▶ 与标准 RNN 细胞相比，LSTM 细胞具有额外的（更稳定的）状态 c_t，该状态只能通过"门"函数访问。

Long Short-Term Memory

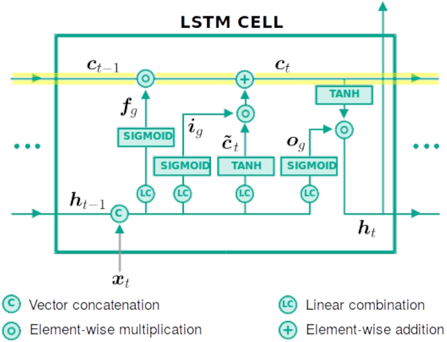
- 状态 c 仅通过三个门控（一个门控是一个 sigmoid 运算的乘法）访问。
  - “遗忘门” (f_g) 执行“擦除”操作。
  - “输入门” (i_g) 执行“写入”操作。
  - “输出门” (o_g) 执行“读取”操作。



Observation:

- The state $\boldsymbol{c}$ is only accessed through three gates (a gate is a multiplication by a sigmoid). The '*forget gate*' $\boldsymbol{f}_g$ performs an 'erase' operation. The '*input gate*' $\boldsymbol{i}_g$ performs an 'write' operation. The '*output gate*' $\boldsymbol{o}_g$ performs an 'read' operation.

Long Short-Term Memory

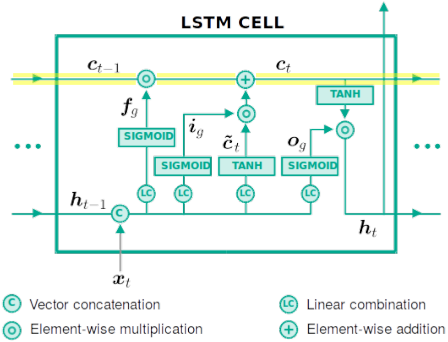


Observation (2):

- The state c stays stable over time (it is only erased or updated when the input gate is open), and there are no weight matrices or nonlinearities transforming c over different time steps, i.e. by default it stays constant.

- 状态 c 随时间保持稳定（仅当输入门打开时才会被擦除或更新）。
  - 不存在权重矩阵或非线性变换作用于 c 的不同时间步，这意味着默认情况下 c 保持不变。

Long Short-Term Memory



Observation (3):

- The gradient flows well and predictably along the path $c_{t-1}, c_t, \ldots$. In particular, the addition operation does not change the gradient. The gradient can then only be dampened by the forget gate, and *never* amplified.

梯度沿着路径 $c_{t-1}, c_t, \ldots$ 流动良好且可预测。

- 特别地，加法运算不会改变梯度。梯度只可能被遗忘门抑制，而不会被放大。

Part 5

RNN Applications

RNNs for Machine Translation

Google Neural Machine Translation:

- Encoder-Decoder architecture with input word vectors in the source language, and output in the target language.
- Stack of LSTMs with residual connections through the stack for better gradient flow. First layer is bidirectional.
- Many more details (attention mechanisms, few-shot learning procedure, etc.)

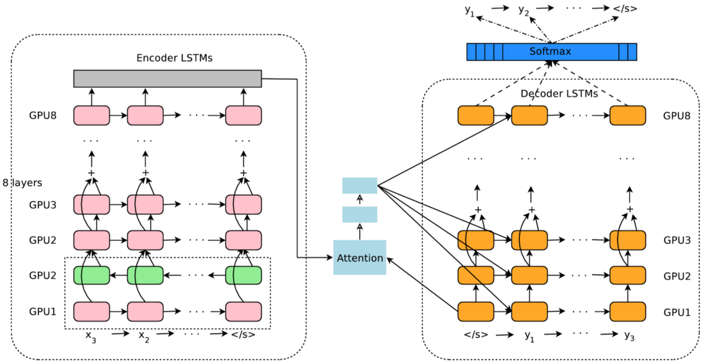


Image source: Wu et al. https://arxiv.org/abs/1609.08144

RNNs for Modeling Motion

Idea:

- Learn a recurrent neural network model of motion (e.g. of a salamander) from observed behavior.运动
- The motion can then be steered by forcing certain neurons or input of the RNN to take specific values.
- The model can be analyzed for insights into the mechanisms of locomotion.

Locomotion（移动/行走）

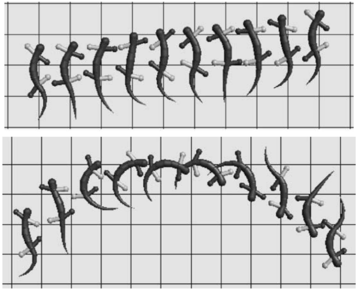


Image Source:
Ijspeert. Biol Cybern 84, 331–348 (2001)

Summary

Summary

- Recurrent neural networks (RNNs) are a special type of neural networks where the internal representation depends both on the input and of the neural network's state.
- RNNs are therefore time-dependent. This makes them natural architectures for modeling processes over time such as the evolution of dynamical systems or more generally sequential data.
- RNNs can be unfolded in time, resulting in deep neural networks with a number of layers proportional to the number of time steps, and shared parameters between the multiple layers.
- In practice, RNNs are hard to train due to the vanishing/exploding gradient problem. A powerful extension of RNNs that exhibits higher stability is the LSTM.