LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Deep Learning for Physicists

Lecture #7: Recurrent neural networks and sequential data

Kosmas Kepesidis

Outline

- Sequential relations
- Recurrent neural networks (RNNs)
- Training of RNNs
- Long short-term memory (LSTM)
- Gated recurrent units (GRU)
- Areas of application
- Applications in physics

Sequential relations

- **Sequential data**: inherently ordered in some way
- Examples include:
  - Time series data
  - Words, sentences in text data
  - Strings of nucleobases that make up DNA
  - Weather/economic forecasts
  - Data from financial markets
  - Audio/video recordings
  - …

Sequential relations

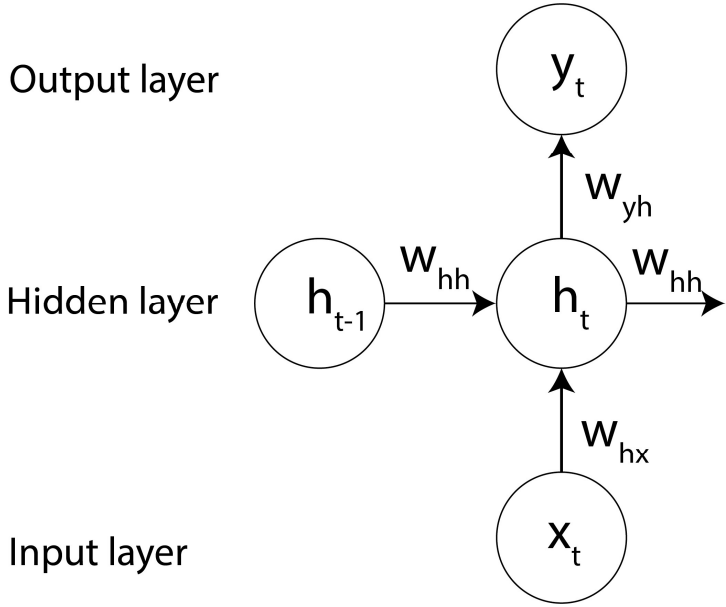
- Data streams in physics:
  - Time evolution is a central theme in physics – sequential data typically come in the form of **temporal sequences**
  - Feedforward neural networks have no capacity to recognize “order” in the input data and thus cannot solve problems where the temporal context is important
    - For example: predicting next step in a sequence
  - It is natural to seek an architecture that takes a-priori into account the temporal character of the data
    - Use information from previously “seen” data when evaluating a new data point

Sequential relations

- Data streams in physics:
  - Problem is solved by an architecture called **recurrent neural networks** (RNNs) that incorporates a type of memory called **internal state**
  - RNNs were invented in the 1980s – but computational issues (e.g. vanishing gradient), prevented their use until the late 1990s
  - Main breakthroughs took place in 2010s, as large amounts of sequential data sets started being available

Recurrent neural networks (RNNs)

- Basic unit of an RNN is show on the right
- Assume we capture d quantities in an experiment, at each time point t
  - Input data $\vec{x}_t$ (at single time point) is a d-dimensional vector
  - **Hidden state** at each timepoint is represented by a p-dimensional vector $\vec{h}_t = f(\vec{x}_t,\ \vec{h}_{t-1})$
  - The output $\vec{y}_t = g\left(\vec{h}_t\right)$, which depend only on the hidden layer, could be a prediction for $\vec{x}_{t+1}$
  - f and g are time invariant: implying that underline characteristics of a time series do not change in time



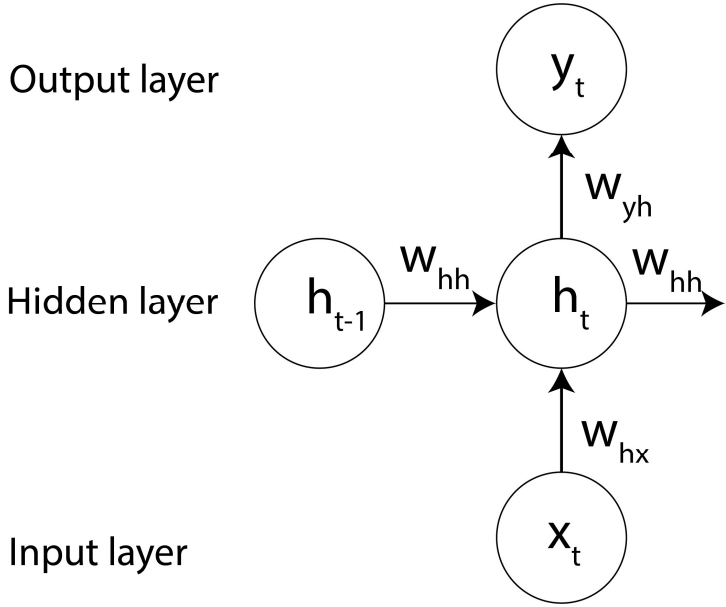
Recurrent neural networks (RNNs)

- Basic unit of an RNN is show on the right
- Assume we capture d quantities in an experiment, at each time point t
  - Computations are performed in a way similar to feedforward networks

$$\vec{h}_t = \tanh(\mathbf{W}_{hx}\,\vec{x}_t + \mathbf{W}_{\mathrm{hh}}\vec{h}_{t-1} + \vec{b}_h)$$
$$\vec{y}_t = \sigma(\mathbf{W}_{yh}\,\vec{h}_t + \vec{b}_y)$$

  - Matrix dimensions:

$$W_{hx} \to p \times d,\;\; W_{hh} \to p \times p,\;\; W_{yh} \to n \times p$$

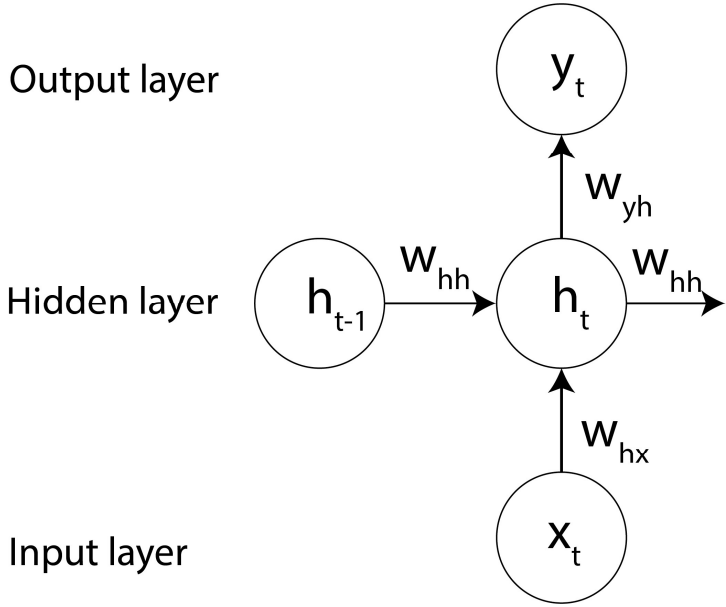


Recurrent neural networks (RNNs)

- Basic unit of an RNN is show on the right
- Assume we capture d quantities in an experiment, at each time point t
  - The hidden vector $\vec{h}_t$ is updated as soon as a new data point is inserted
  - The input vector $\vec{x}_t$ interacts with the hidden state of the previous step and indirectly with all previous time steps

$$\vec{h}_t = f\left(\vec{x}_t, \vec{h}_{t-1}\right) = f\left(\vec{x}_t, f\left(\vec{x}_t, \vec{h}_{t-2}\right)\right) = u(\vec{x}_t, \vec{x}_{t-1}, \vec{x}_{t-2}, \dots)$$

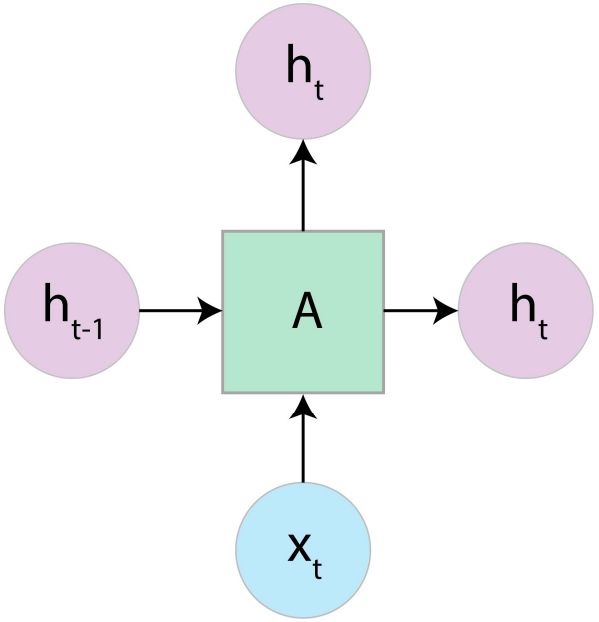
  - These recursive characteristics enable RNNs to process time series data with variable input



Recurrent neural networks (RNNs)

- Core building block of an RNN

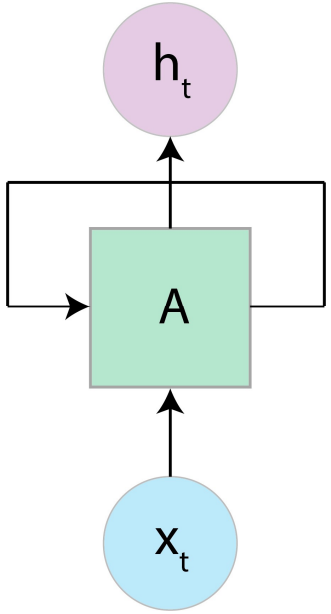
  ➢ The unit **A** performs the computation of $\vec{h}_t = \tanh(\mathbf{W}_{hx}\,\vec{x}_t + \mathbf{W}_{\mathrm{hh}}\vec{h}_{t-1} + \vec{b}_h)$



C. Olah, Understanding LSTM networks, (2015)

Recurrent neural networks (RNNs)

- Core building block of an RNN
  - The unit **A** performs the computation of $\vec{h}_t = \tanh(\mathbf{W}_{hx}\,\vec{x}_t + \mathbf{W}_{\mathrm{hh}}\vec{h}_{t-1} + \vec{b}_h)$
  - Due to its recursive properties can be symbolized by **self-loop**
  - Since information on the hidden state is passed on as additional input from one time point to the next

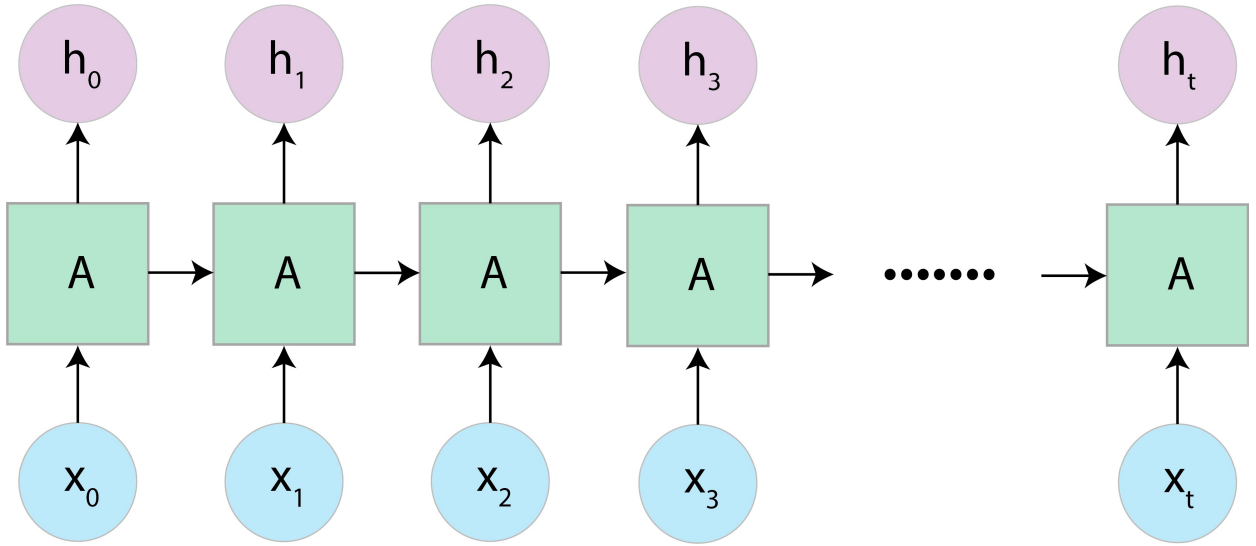


C. Olah, Understanding LSTM networks, (2015)

Recurrent neural networks (RNNs)

- Core building block of an RNN

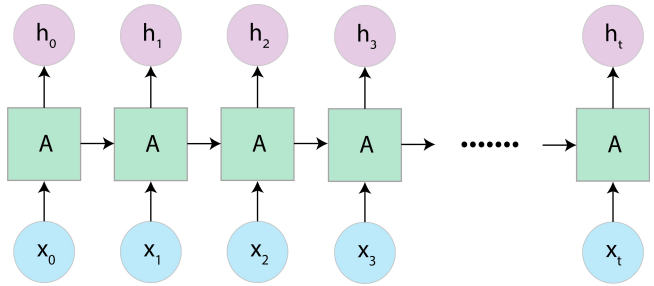
  ➢ Unrolling of a recurrent neural network – horizontal axis is equivalent to a timeline



C. Olah, Understanding LSTM networks, (2015)

Recurrent neural networks (RNNs)

- Core building block of an RNN
  - ➢ Unrolling of a recurrent neural network – horizontal axis is equivalent to a timeline
  - ➢ The recurrent unit can be considered as an internal memory that controlled by the network
  - ➢ Such stored states make up what is known as **gated memory**



C. Olah, Understanding LSTM networks, (2015)

Training of RNNs

- Feedforward networks are trained using **backpropagation**
- For RNNs, a method adapted to sequential data is used: **backpropagation through time** (BPTT)
  - The algorithm is based on the unrolled version of the RNN taking into account that all layers share the same parameters
- Contrast to regular backpropagation:
  - **Regular backpropagation**: gradients are calculated for each input, which is then propagated backwards through the network from the output to the input to update weights
  - **BPTT**: gradients for the network parameters are accumulated over all time steps before passing the result backwards through the network

Training of RNNs

- Issues with training:
  - Theoretically, RNNs can process arbitrarily long sequences and their long-range dependences
  - In practice, severe numerical instabilities arise due to the **vanishing gradient** problem
    - Due to the N-fold multiplication of $\mathbf{W}_{\mathrm{hh}}$ with itself (in sequences with N time points)
    - Leading to poor long-term memory for $N \gg 1$
    - As a result, critical temporal correlations escape the learning process of basic RNNs (bad long-term memory)
  - Solutions:
    1. **Long short-term memory** (LSTM)
    2. **Gated recurrent unit**

Long short-term memory (LSTM)

- **Long short-term memory** (LSTM) was developed for tackling the problem of long-term memory (vanishing gradient)
- **LSTM units** have dramatically improved the performance of RNNs and is responsible for their practical successes
- They are able to remember data over arbitrarily-long time scales:
  - Due to internal memory – called **cell state**
  - And **gates** that control the **cell state**
- Idea is to modify the recursive formula for $\vec{h}_t = f(\vec{x}_t,\ \vec{h}_{t-1})$ in a way that allows control over its updates
  - This is done with the introduction of the p-dimensional **cell-state vector**: $\vec{C}_t$

S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Long short-term memory (LSTM)

- $\vec{C}_t$ can selectively preserve or delete stored elements and “decide” whether to take in new information

- LSTM unit is compose of three elements (gates), which control the information flow:
  1. **Forget gate**: decides whether to delete or keep elements stored in the cell
  2. **Input gate**: decides whether new information flows into the cell
  3. **Output gate**: decides whether the updated cell value contributes to the hidden state

  - Above decisions are not binary (0, 1)
  - Layers that perform gate operations use an **activation function** such as the sigmoid which return values between 0 and 1
  - Decisions concern the degree to which a value is further processed

S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Long short-term memory (LSTM)

- Intermediate vectors:
  - For simplicity, we restrict ourselves to one-dimensional input (also drop vector notation)

$$
\begin{aligned}
&\text{forget layer}: && f_t = \sigma\left(W_f\, x_t + U_f\, h_{t-1} + b_f\right)\\
&\text{input layer}: && i_t = \sigma\left(W_i\, x_t + U_i\, h_{t-1} + b_i\right)\\
&\text{output layer}: && o_t = \sigma\left(W_o\, x_t + U_o\, h_{t-1} + b_o\right)\\
&\text{cell input layer}: && \tilde{C}_t = \tanh\left(W_c\, x_t + U_c\, h_{t-1} + b_c\right)
\end{aligned}
$$

  - Above variables are set for each time step using weights **W**, **U** and biases **b**

S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Long short-term memory (LSTM)

- Cell-state calculation:
  - The new cell state C_t is calculated by the weighted average between the previous cell state C_{t-1} and the intermediate vector $\tilde{C}_t$
    - $C_t = f_t \, C_{t-1} + i_t \, \tilde{C}_t$
    - f_t decides how much of the previous state will be "forgotten"
    - i_t decides how much of new input will be added to the cell state

S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

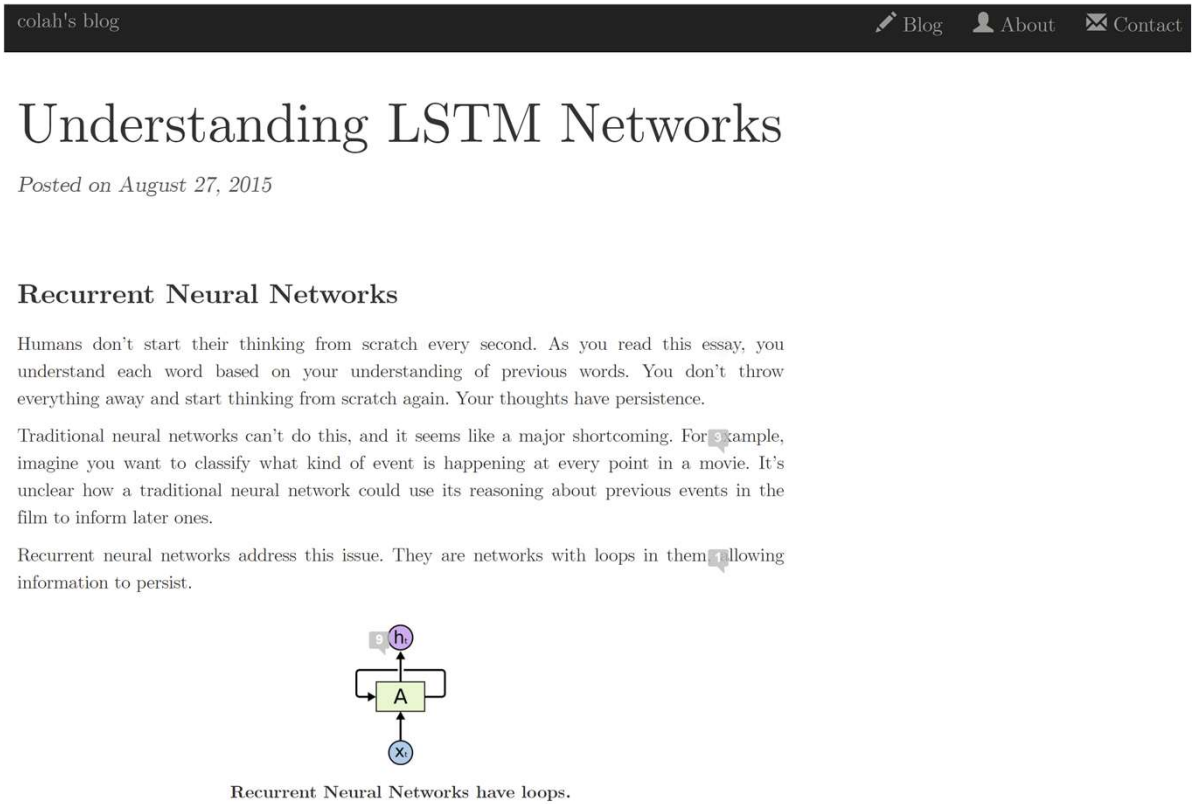
Long short-term memory (LSTM)

- Hidden-state calculation:
  - $h_t = o_t \tanh(C_t)$
  - o_t indicates the extend to which information flows from the cell state into the hidden state
  - $\frac{\partial C_t}{\partial C_{t-1}} = f_t$ and since o_t depends only on W_0, U_0 and b_0, the partial derivatives of h_t with respect to the weights require only the derivative of $\tanh()$

S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Long short-term memory (LSTM)

- A great blog post about “LSTMs”: https://colah.github.io/posts/2015-08-Understanding-LSTMs/



colah's blog

Blog About Contact

Understanding LSTM Networks

Posted on August 27, 2015

Recurrent Neural Networks

Humans don't start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don't throw everything away and start thinking from scratch again. Your thoughts have persistence.

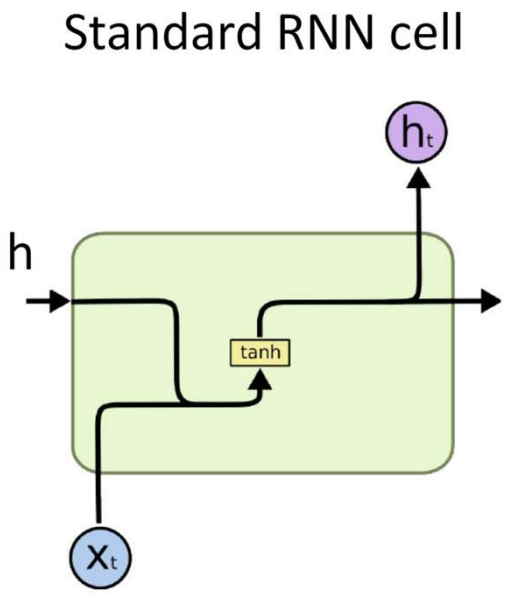
Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.

Recurrent Neural Networks have loops.

Long short-term memory (LSTM)

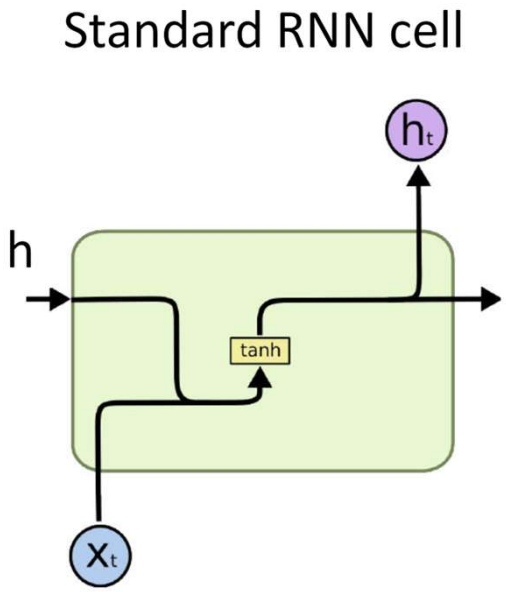
- Comparison of basic RNN architecture and that of LSTM



S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Long short-term memory (LSTM)

- Comparison of basic RNN architecture and that of LSTM

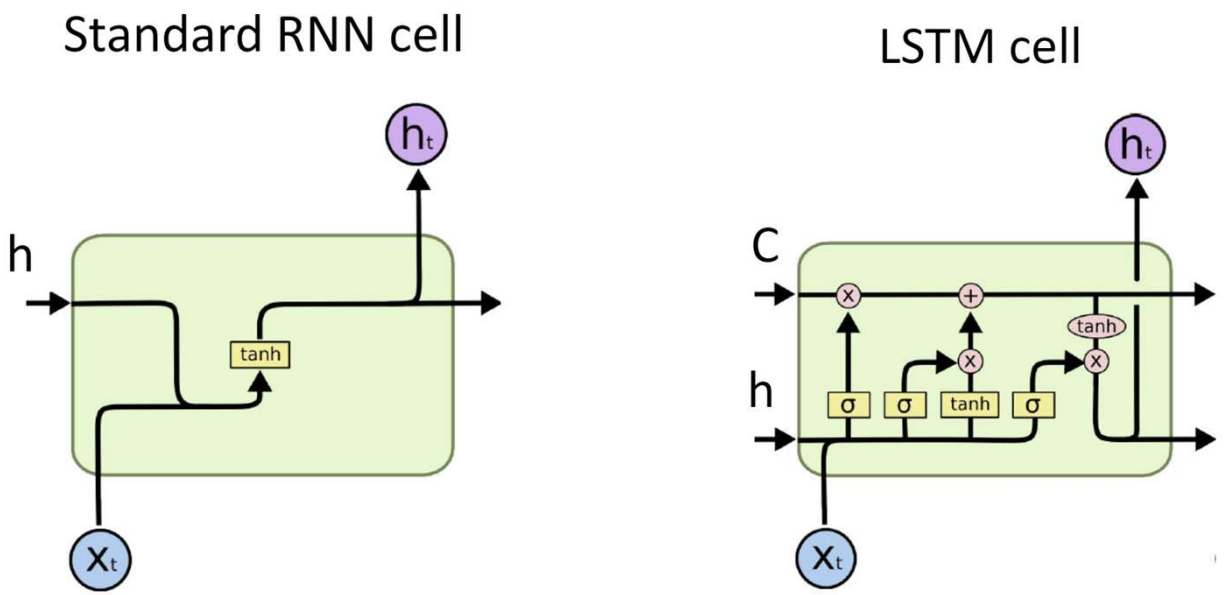


$$h_t = \tanh\left(W_f\, x_t + U_f\, h_{t-1} + b_f\right)$$

S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Long short-term memory (LSTM)

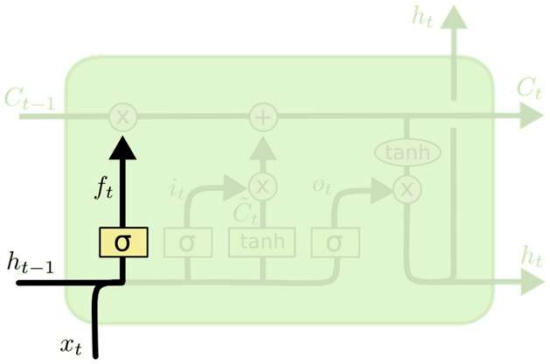
- Comparison of basic RNN architecture and that of LSTM



S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Long short-term memory (LSTM)

- LSTM cell operations, step by step

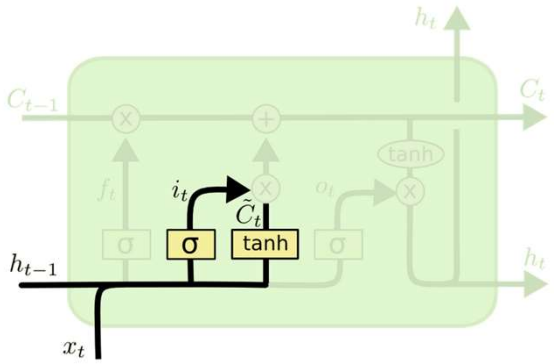


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Long short-term memory (LSTM)

- LSTM cell operations, step by step



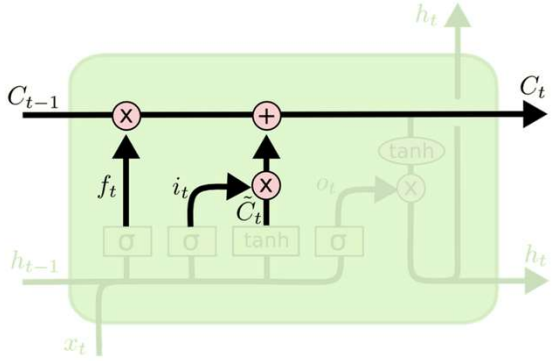
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Long short-term memory (LSTM)

- LSTM cell operations, step by step

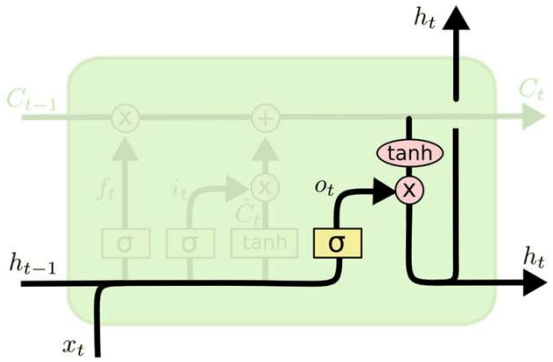


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Long short-term memory (LSTM)

- LSTM cell operations, step by step



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9, pp. 1735–1785 (1997)

Gated recurrent units (GRU)

- **Gated recurrent units** (GRU) were introduced in 2014 and are widely used nowadays
- GRU aim at simplifying the LSTM weight-update process
- In GRU there exist no cell state but only two gates to control the update and the hidden state at each time point
- Instead of having separate *forget*, *input* and *output* gates, here we have a *reset* and an *update* gate

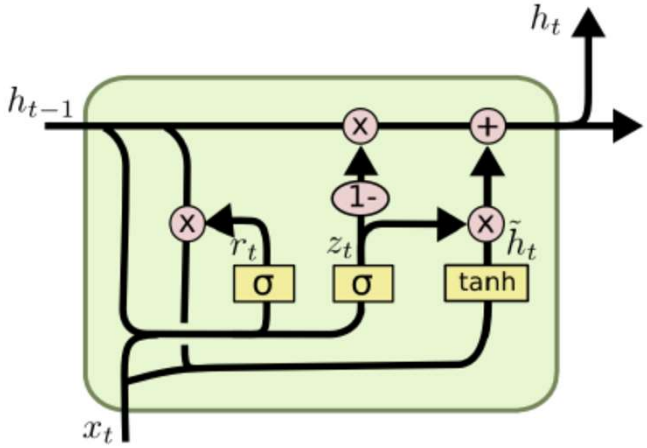
Gated recurrent units (GRU)

- Updates are performed directly on the hidden state for each time step in a two-step process
- First, the intermediate variables are set up using specific weights W, U and biases b
- An intermediate variable $\tilde{h}_t$ is calculated as follows:

$$
\begin{aligned}
&\text{reset layer :} && r_t = \sigma\left(W_r\, x_t + U_r\, h_{t-1} + b_r\right)\\
&\text{update layer :} && z_t = \sigma\left(W_z\, x_t + U_z\, h_{t-1} + b_z\right)\\
&\text{candidate layer :} && \tilde{h}_t = \tanh\left(W_h\, x_t + U_h\left(r_t \cdot h_{t-1}\right) + b_h\right)
\end{aligned}
$$

Gated recurrent units (GRU)

- Update of hidden variable: $h_t = (1 - z_t) \cdot h_{t-1} + z_t \cdot \tilde{h}_t$
- Note: new information is stored in the hidden state only on the expense of old information



J. Chung et al., Empirical evaluation of gated recurrent neural networks on sequence modeling (2014)

Gated recurrent units (GRU)

- GRUs are simpler to implement than LSTM and require fewer parameters, which explains their growing popularity
- But LSTM have been used for much longer and considered the “safer” option

J. Chung et al., Empirical evaluation of gated recurrent neural networks on sequence modeling (2014)

Areas of application

- RNNs find applications in a broad range of fields – when data is sequential

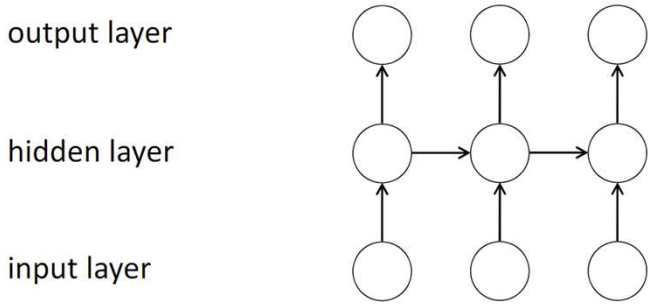
  - Natural Language Processing (NLP):
    - Conversion of phrases into machine-interpretable format
    - Can simplify human-device interaction, e.g. voice assistants
    - Convert speech into text
    - Language translation

  - Processing protein sequence data in bioinformatics:
    - Different protein sequences have different length: RNNs are flexible to this regard

Areas of application

- Different “flavors”, different applications

  - Standard mapping of RNNs:

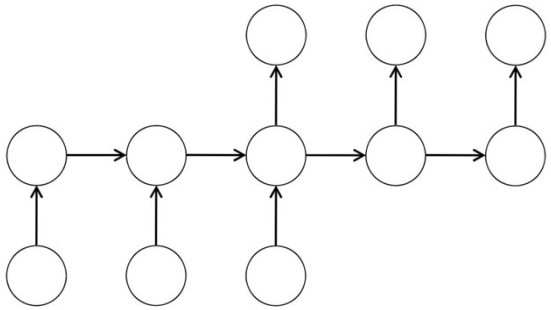


  - Applications:
    1. Transform the input sequence (e.g. Temperature readings)
    2. Annotate video frame by frame

Areas of application

- Different “flavors”, different applications

  - Delayed sequence processing:

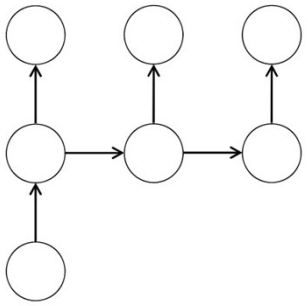


  - Applications:
    1. Predicting future time steps of a time series
    2. Autocorrect in text applications
    3. Automatic translation: initial words are used for creating further translation

Areas of application

- Different “flavors”, different applications

  - Sequence reduced to a single element:

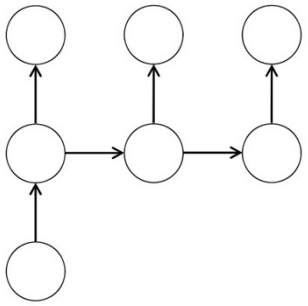


  - Applications:
    1. Automatic captioning of images, with a single image as input

Areas of application

- Different “flavors”, different applications

  - Single event to sequence:

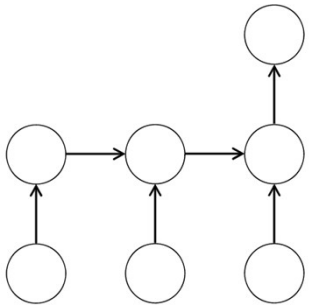


  - Applications:
    1. Automatic captioning of images, with a single image as input

Areas of application

- Different “flavors”, different applications

  - Sequence to single event:



  - Applications:
    1. Text classification

Applications in physics

- Large Hadron Collider (LHC) monitoring:
  - LHC depends on superconducting magnets to keep the circulating protons in the vacuum tube on the correct trajectories
  - Stability of the magnets is crucial for proper operation and therefore their function should be monitored
  - So-called “quench” corresponds to unintentional loss of superconducting state of a magnet which has the capacity to destroy the magnet unless energy is dissipated in a controlled way

M. Wielgosz et al., Using LSTM recurrent neural networks for monitoring the LHC superconducting magnets, Nuclear Inst. and Methods in Physics Research (2017)

Applications in physics

- Large Hadron Collider (LHC) monitoring:
  - Magnets are equipped with an monitoring system that records voltage signals (time traces) that can be generated when the superconducting state is lost
  - Several configurations of RNNs with 128 LSTM cells were tested
  - The networks were trained with real voltage-time data for a commonly used magnets
  - Data from 425 quench events were monitored between 2008 and 2016 and used for training (including 24 hours before each event)
  - RNNs proved to be able to predict magnet failure!!!

M. Wielgosz et al., Using LSTM recurrent neural networks for monitoring the LHC superconducting magnets, Nuclear Inst. and Methods in Physics Research (2017)

Applications in physics

- Crack propagation in simulated materials
  - ➢ RNNs with LSTM cells were used to track the time evolution of stress and damage in a simulated material
  - ➢ Result: prediction of summary statistics for quantities of interest from initial conditions to the failure point

A. Mehta, et al., Physics-informed spatiotemporal deep learning for emulating coupled dynamical systems, Proceedings of the AAAI 2020 Spring Symposium on Combining Artificial Intelligence and Machine Learning with Physical Sciences (2020)

Applications in physics

- Another application in this weeks tutorial: cosmic rays
  - Large arrays of radio antennas can be used to measure cosmic rays by recording the electromagnetic radiation generated in the atmosphere
  - These radio signals are strongly contaminated by galactic noise as well as signals from human origin
  - Since these signals appear to be similar to the background, the discovery of cosmic-ray events can be challenging

Summary

- Recurrent neural networks (RNNs) are a special type of artificial neural network suitable for sequential data
- RNN building blocks are called cells
- RNNs are trained by a backpropagation algorithm adapted to sequential data, called backpropagation through time

Summary

- The classical realization of an RNN cell is the long short-term memory (LSTM)
- LSTM cells have a further internal memory called cell state
- Another important realization of an RNN cell is the gated recurrent unit (GRU)
- GRUs have no cell state and only two gates to control the update of the hidden state