

Sequence Processing

Moritz Wolter

September 28, 2023

High-Performance Computing and Analytics Lab, Uni Bonn

Recurrent neural networks

- Elman-RNN

- Long Short Term Memory

- Gated recurrent Units

- Orthogonal networks

Applications

Neural Attention and Transformers

Code snippets

Motivation

- Thus far we have never integrated information over time.
- We want the ability to create internal memory.
- Consider the sentence: I live in Paris. I speak ...
- ... French.
- Clearly it is likely for someone in Paris to speak French.
- Memory should help networks taking Paris into account when deciding what language is spoken.

Recurrent neural networks

- Recurrent neural networks are often considered the goto choice for sequences.
- Chapter ten in [GBC16], for example, bears the title "Sequence Modeling: Recurrent and Recursive Nets".

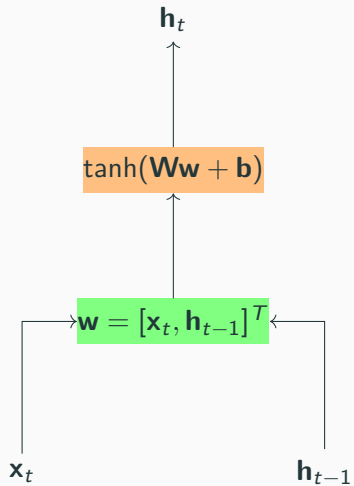
Elman-recurrent neural networks

A simple solution is to add a state to the network and feed this state recurrently back into the network [Elm90],

$$\overline{\mathbf{h}}_t = \mathbf{W}_h \mathbf{h}_t + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}, \quad (1)$$

$$\mathbf{h}_{t+1} = f(\overline{\mathbf{h}}_t). \quad (2)$$

Elman-recurrent neural networks



Unrolling in Time

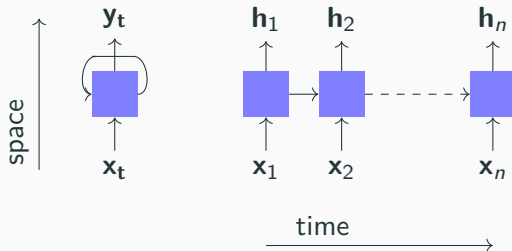


Figure: The rolled (left) cell can be unrolled (right) by considering all inputs it saw during the current gradient computation iteration.

Stability of recurrent connections

For an intuition. Consider a linear network without activations or inputs.

$$\mathbf{h}_{t+1} = \mathbf{W}_h \mathbf{h}_t \quad (3)$$

The evolution of the \mathbf{h} -sequence is guided by its largest eigenvalue. If an eigenvalue larger than one exists. The state explodes. If all eigenvalues are smaller than one the state vanishes [GBC16].

Long Short Term Memory (LSTM)

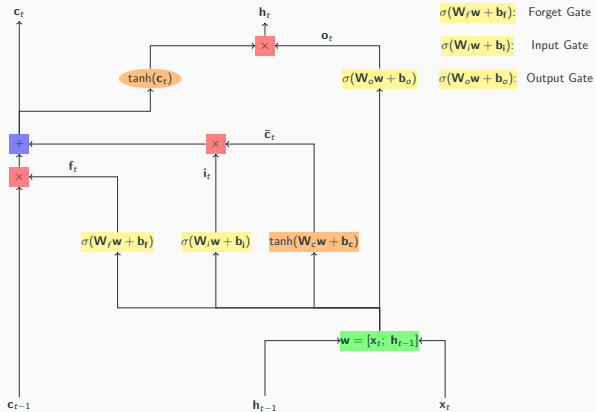


Figure: An LSTM cell as described in [HS97; Gre+16].

Long Short Term Memory (LSTM)

Like a differentiable memory chip [Gra12] LSTM-memory can store n_h numbers. Gates govern all changes to the cell state. Gate and state equations are defined as [HS97; Gre+16]

$$\mathbf{z}_t = \tanh(\mathbf{W}_z \mathbf{x}_t + \mathbf{R}_z \mathbf{h}_{t-1} + \mathbf{b}_z), \quad (4)$$

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{R}_i \mathbf{h}_{t-1} + \mathbf{p}_i \odot \mathbf{c}_{t-1} + \mathbf{b}_i), \quad (5)$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{R}_f \mathbf{h}_{t-1} + \mathbf{p}_f \odot \mathbf{c}_{t-1} + \mathbf{b}_f), \quad (6)$$

$$\mathbf{c}_t = \mathbf{z}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (7)$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{R}_o \mathbf{h}_{t-1} + \mathbf{p}_o \odot \mathbf{c}_t + \mathbf{b}_o), \quad (8)$$

$$\mathbf{h}_t = \tanh(\mathbf{c}_t) \odot \mathbf{o}_t. \quad (9)$$

Potential new states \mathbf{z}_t are called block input. \mathbf{i} is called the input gate. The forget gate is \mathbf{f} and \mathbf{o} denotes the output gate. $\mathbf{p} \in \mathbb{R}^{n_h}$ are peephole weights, $\mathbf{W} \in \mathbb{R}^{n_i \times n_h}$ denotes input, $\mathbf{R} \in \mathbb{R}^{n_o \times n_h}$ are the recurrent matrices. \odot indicates element-wise products.

Long Short Term Memory (LSTM)

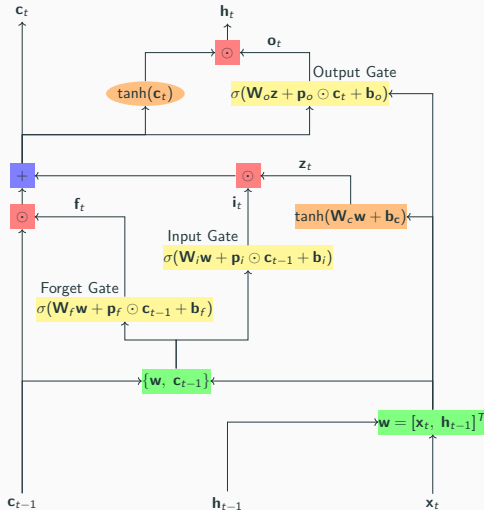


Figure: An LSTM-cell with peephole connections as described in [HS97; Gre+16]

Gated recurrent Units

$$\mathbf{r}_t = \sigma(\mathbf{W}_r \mathbf{h}_{t-1} + \mathbf{V}_r \mathbf{x}_t + \mathbf{b}_r), \quad (10)$$

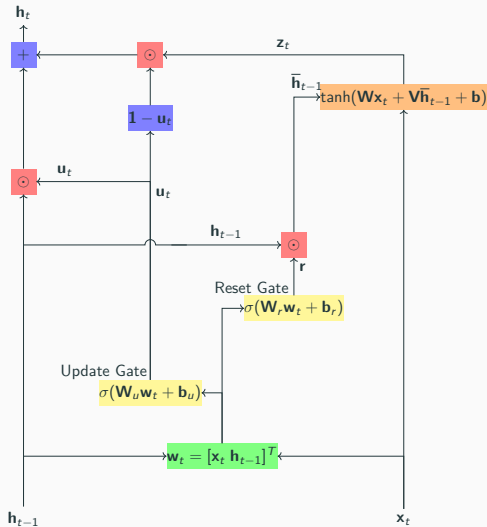
$$\mathbf{u}_t = \sigma(\mathbf{W}_u \mathbf{h}_{t-1} + \mathbf{V}_u \mathbf{x}_t + \mathbf{b}_u) \quad (11)$$

$$\mathbf{z}_t = \tanh(\mathbf{W}(\mathbf{r}_t \odot \mathbf{h}_{t-1}) + \mathbf{V} \mathbf{x}_t + \mathbf{b}), \quad (12)$$

$$\mathbf{h}_t = \mathbf{u}_t \odot \mathbf{z}_t + (1 - \mathbf{u}_t) \odot \mathbf{h}_{t-1}. \quad (13)$$

$\mathbf{h}_t \in \mathbb{R}^{n_h}$ denotes the cell state and output at time t . The block input is called $\mathbf{z}_t \in \mathbb{R}^{n_h}$. The reset $\mathbf{r} \in \mathbb{R}^{n_h}$ and update gates $\mathbf{u} \in \mathbb{R}^{n_h}$ take care of memory management. $\mathbf{W} \in \mathbb{R}^{n_i \times n_h}$ denote input matrices, $\mathbf{V} \in \mathbb{R}^{n_h \times n_h}$ is used for recurrent weight matrices.

Gated recurrent units



Stiefel Manifold Weight Updates [Wisdom2016]

$$\mathbf{h}_t = \text{ReLU}(\mathbf{W}_h \mathbf{h}_t + \mathbf{W}_x \mathbf{x}_t + \mathbf{b}) \quad (14)$$

$$\mathbf{W}_{k+1} = (\mathbf{I} + \frac{\lambda}{2} \mathbf{A}_k)^{-1} (\mathbf{I} - \frac{\lambda}{2} \mathbf{A}_k) \mathbf{W}_k, \quad (15)$$

$$\text{where } \mathbf{A} = \mathbf{W} \overline{\nabla_{\mathbf{w}} F}^T - \overline{\mathbf{W}}^T \nabla_{\mathbf{w}} F. \quad (16)$$

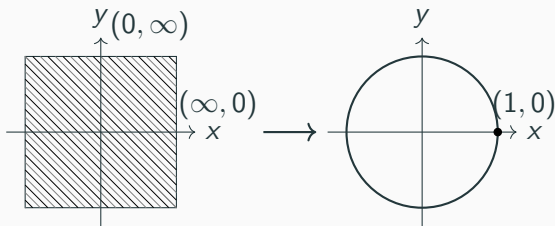


Figure: Fix the optimized matrix eigenvalues onto the unit circle.

- LSTM works like a differentiable memory chip.
- When in doubt, use LSTM.

Applications

Time-series forecasting

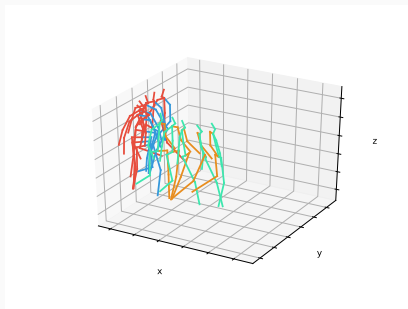
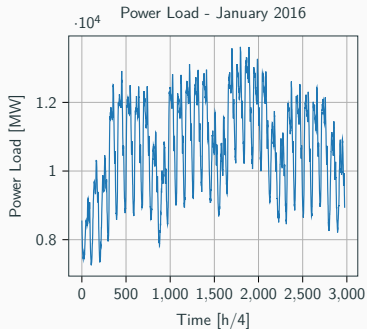
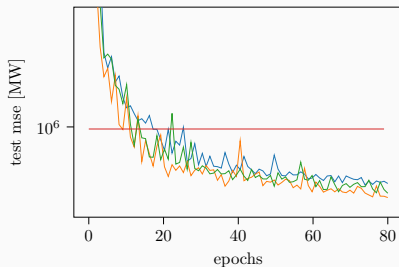
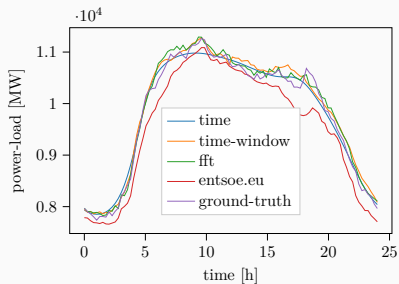


Figure: Monovariate power-load and multivariate motion-capture time series data.

Day-ahead power-load

Day-ahead power load forecasting using European Network of Transmission system operators for electricity data: [WGY20]



One hot encoding for letters. A possible encoding looks for all characters in a dataset. The number of occurring characters determines the length of every one-hot character vector. A system that accepts text and produces text, therefore, maps one-hot encoded sequences onto each other.

Given a sequence of input letters or words LSTM, for example, can model the probability of the next letter or word.

$$p_n(y_i|y_1, y_2, \dots, y_{i-1}) = LSTM(y_{i-1}, c_{i-1}, h_{i-1}) \quad (17)$$

This could, for example, help users type.

- RNNs are versatile and suitable for many different sentence processing tasks.
- But, there's more!

Example: Machine Translation

[BCB15] used RNN for the task of machine translation.

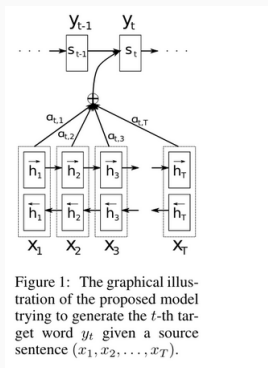


Figure: An RNN-based translation system. Figure from [BCB15].

Neural attention in machine translation

Attention weights group related inputs together, allowing a decoder to find a suitable translation.

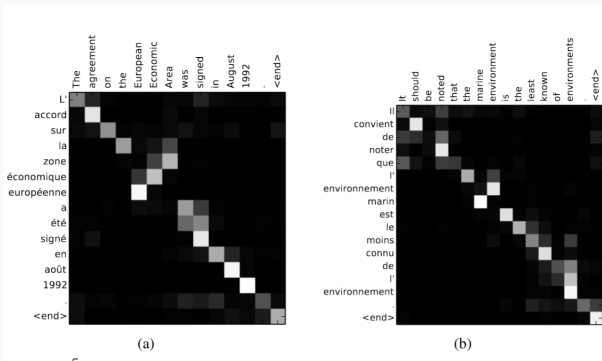
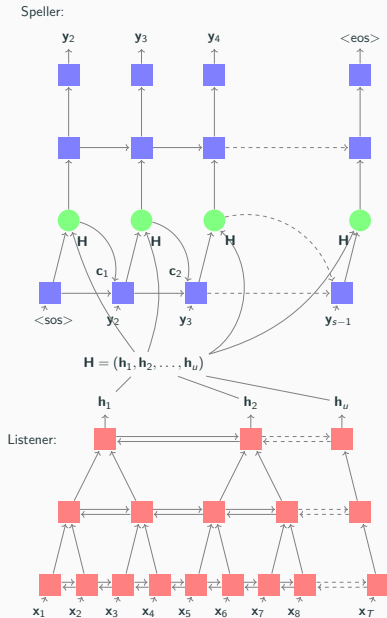


Figure: Attention plots as observed by [BCB15].

Speech Processing [Cha+15]



Attention weights

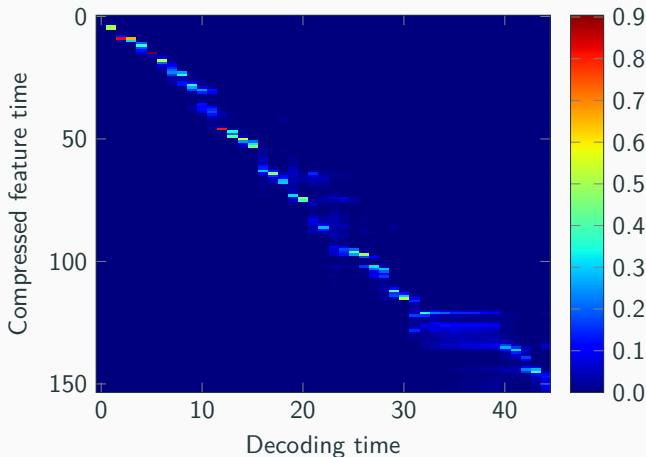


Figure: Attention weights for the speech processing example. On a TIMIT-recording.

Neural Attention and Transformers

Bahadanau attention

Proposed in [BCB15],

$$\mathbf{c}_i = \sum_{j=1}^{T_x} \alpha_{ij} \mathbf{h}_j \quad (18)$$

The idea is to find new α s for every decoding time step i . These are computed using a softmax

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})} \quad (19)$$

if the alignment model outputs $e_{ij} = a(s_{i-1}, h_j)$. Finally, a denotes a feedforward network function of the decoder state \mathbf{s}_{i-1} and annotation \mathbf{h}_j .

Transformers

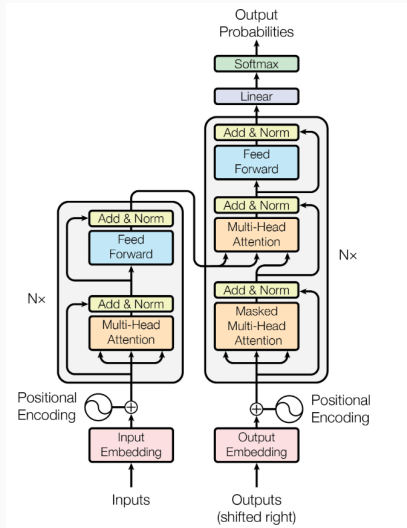


Figure: The transformer architecture as shown in [Vas+17]

[Vas+17] defines dot product attention as,

$$\mathbf{C} = \sigma_s\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \quad (20)$$

With context $\mathbf{C} \in \mathbb{R}^{t,d_k}$, queries $\mathbf{Q} \in \mathbb{R}^{t,d_k}$, keys $\mathbf{K} \in \mathbb{R}^{t,d_k}$, and values $\mathbf{V} \in \mathbb{R}^{t,d_k}$. σ_s denotes the softmax.

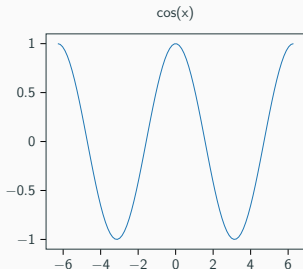
Matrix multiplication and dot products

We can express matrix multiplication as dot products.

$$\mathbf{QK} = \begin{pmatrix} \mathbf{q}_{1,1\dots d_k} \cdot \mathbf{k}_{1\dots d_k,1} & \mathbf{q}_{1,1\dots d_k} \cdot \mathbf{k}_{1\dots d_k,2} & \dots \\ \mathbf{q}_{2,1\dots d_k} \cdot \mathbf{k}_{1\dots d_k,1} & \mathbf{q}_{2,1\dots d_k} \cdot \mathbf{k}_{1\dots d_k,2} & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \quad (21)$$

Alternatively the dot product of two vectors can be written as:

$$\mathbf{q} \cdot \mathbf{k} = |\mathbf{q}||\mathbf{k}|\cos(\theta) \quad (22)$$



Denoising Diffusion Probabilistic Models



Figure 3: LSUN Church samples. FID=7.89

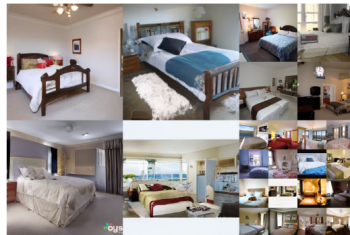


Figure 4: LSUN Bedroom samples. FID=4.90

Figure: Diffusion models rely on a combination of unets and self attention [HJA20].

Conclusion

- Transformers dominate large parts of modern deep learning.
- Their versatility comes at the cost of an enormous data hunger.
- CNN and RNN are still often the better choice on smaller data-sets.
- In today's exercise you can choose to train a generative RNN or a generative transformer.

References

- [BCB15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. “Neural Machine Translation by Jointly Learning to Align and Translate.” In: *CoRR* abs/1409.0473 (2015).
- [Cha+15] William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals. “Listen, attend and spell.” In: *arXiv preprint arXiv:1508.01211* (2015).
- [Elm90] Jeffrey L Elman. “Finding structure in time.” In: *Cognitive science* 14.2 (1990), pp. 179–211.

- [GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [Gra12] Alex Graves. “Supervised sequence labelling.” In: *Supervised sequence labelling with recurrent neural networks*. Springer, 2012, pp. 5–13.
- [Gre+16] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. “LSTM: A search space odyssey.” In: *IEEE transactions on neural networks and learning systems* 28.10 (2016), pp. 2222–2232.

- [HJA20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models.” In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.
- [HS97] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory.” In: *Neural computation* 9.8 (1997), pp. 1735–1780.
- [Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. “Attention is all you need.” In: *Advances in neural information processing systems* 30 (2017).

- [WGY20] Moritz Wolter, Juergen Gall, and Angela Yao.
“Sequence Prediction using Spectral RNNs.” In: *29th
International Conference on Artificial Neural Networks*.
2020.

Code snippets

Sequence coding with dictionaries

```
for int_seq in sequences:
    char_seq = []
    for int_char in int_seq:
        char_seq.append(
            inv_vocab[int(int_char)])
    res.append(char_seq)
```
