Modeling sequences with Neural Networks

Vlado Menkovski

Eindhoven University of Technology

v.menkovski@tue.nl

April 5, 2018

Modeling sequences with Neural Networks

Overview

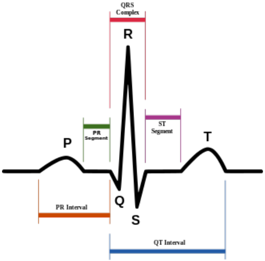
Temporal correlations

Sequential data

- Measurements of processes in time

Example:

- Working of the human hearth:



Should take between .06 - .1s

- Any longer may indicate abnormality

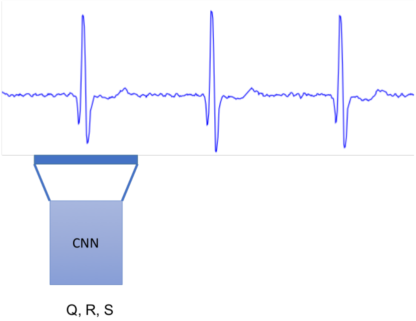
Temporal correlations

Sample of data sequence

Temporal correlations

Feature detector for the Q, R, S

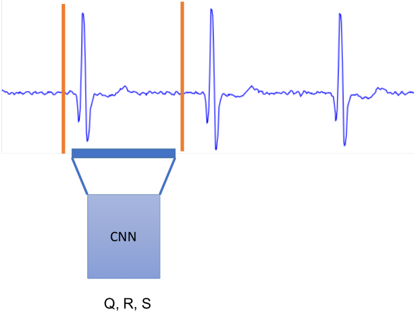
- Can we use a ConvNet?



Temporal correlations

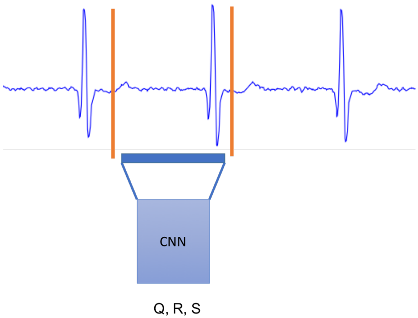
Windowing

- Window size?



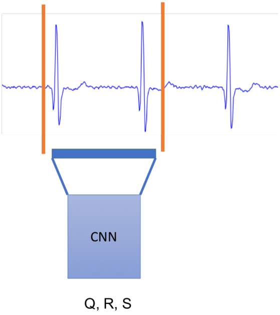
Temporal correlations

Location #2



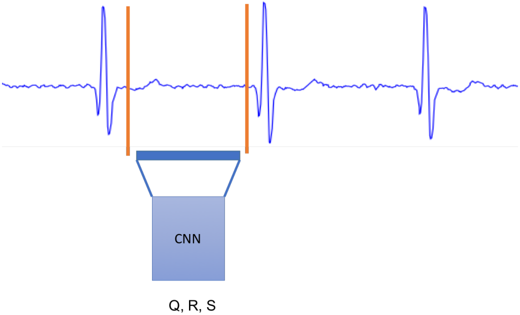
Temporal correlations

Faster sequence



Temporal correlations

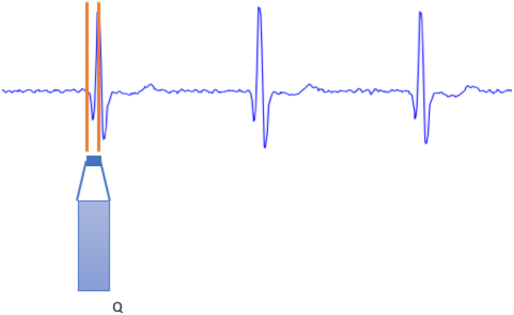
Location #3



Temporal correlations

Sequential processing

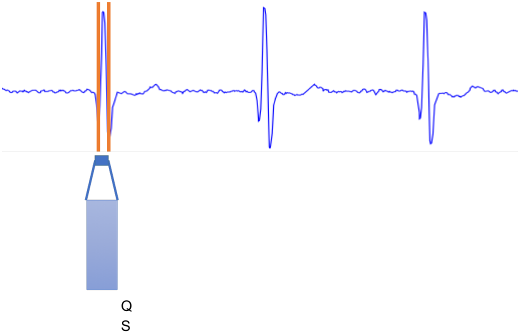
- Step 1



Temporal correlations

Sequential processing

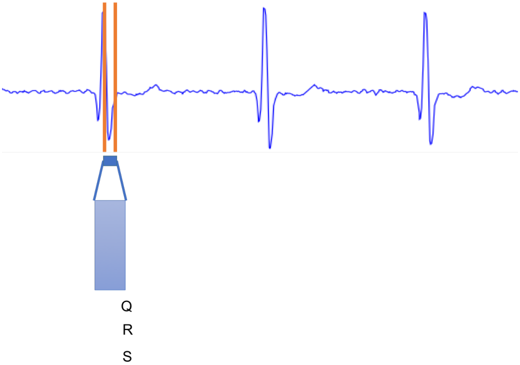
- Step 2



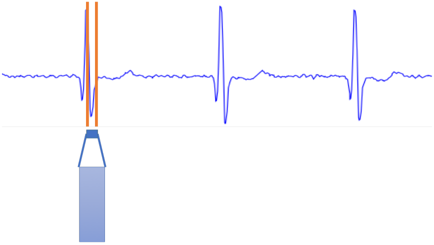
Temporal correlations

Sequential processing

- Step 3



Sequence processing



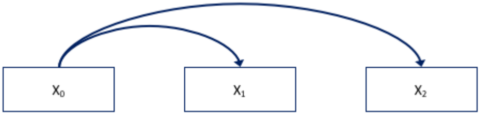
- Feature detector for the Q, and R, and S
- Remember the the point of Q
- Remember the point of R
- Remember the point of S
- Count the distance from Q to R to S

Sequential Data

- Sequence of datapoints is not IID
- A datapoint is a sequence of measurements
- Example:
  - Words in a sentence
  - Sentences in a paragraph
- Modeling sequences with CNN is less efficient
- CNN needs to have feature detectors for all combinations of symbols

Auto-regressive models

Simple prediction model based on previous datapoints



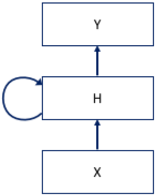
$$x_t = w_0 x_{t-1} + w_1 x_{t-2} + w_2$$

$$x_t = G_\theta(x_{t-1}, x_{t-2}, ...)$$

- Fixed size input
- Fixed size output
- No parameter re-use when processing symbols in the sequence

Recurrent neural network

- Ideally, the model sees each input only once
- Can store information in memory
- Can map correlations over time

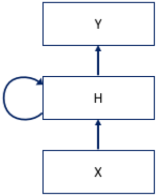


$$h_t = W\phi(h_{t-1}) + Ux_t$$

$$y_t = V\phi(h_t)$$

Recurrent neural network

- Describes a program
- With certain inputs and some internal variables



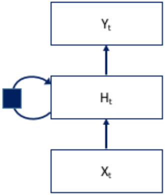
$$h_t = W\phi(h_{t-1}) + Ux_t$$

$$y_t = V\phi(h_t)$$

- Input x, output y
- The output y is conditioned on x and previous values of x

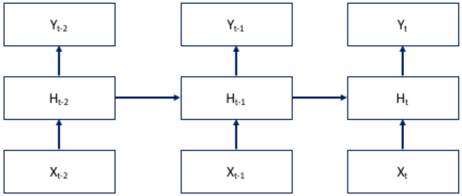
Recurrent neural network

- Delay edge



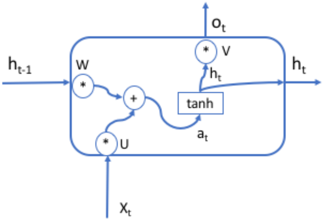
Recurrent neural network

- Diagram unrolled



Recurrent neural network

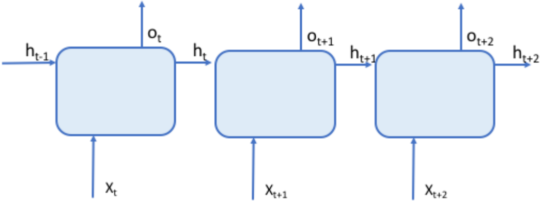
- Recurrent neural network model



$$a^{(t)} = b + Wh^{(t-1)} + Ux^{(t)}$$
$$h^{(t)} = \tanh(a^{(t)})$$
$$o^{(t)} = C + Vh^{(t)}$$
$$\hat{y}^{(t)} = softmax(o^{(t)})$$

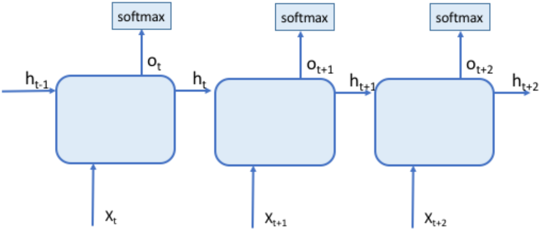
Recurrent neural network

- Recurrent neural network model



Recurrent neural network

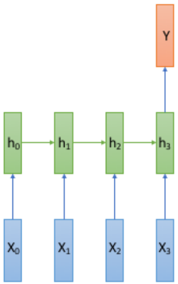
- Recurrent neural network model



Application of RNN

Applications

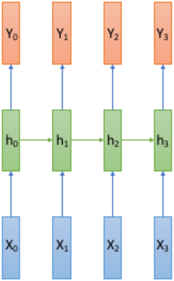
- **sequence classification**
- sequence to sequence mapping (seq2seq)
- sequence generation



Application of RNN

Applications

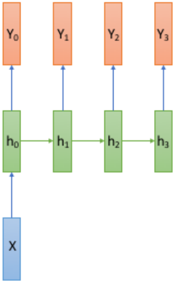
- sequence classification
- **sequence to sequence mapping (seq2seq)**
- sequence generation



Application of RNN

Applications

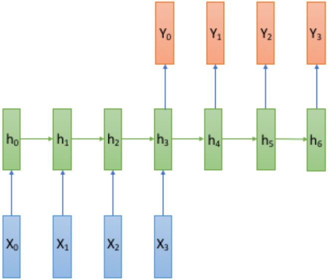
- sequence classification
- sequence to sequence mapping (seq2seq)
- **sequence generation**



Application of RNN

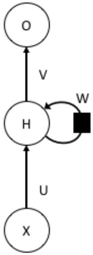
Applications

- What is possible reason for this architecture?

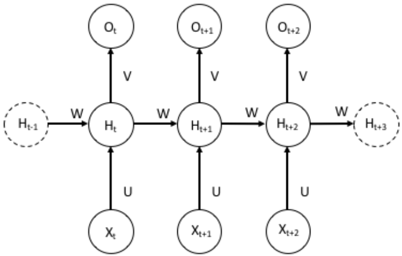


Training

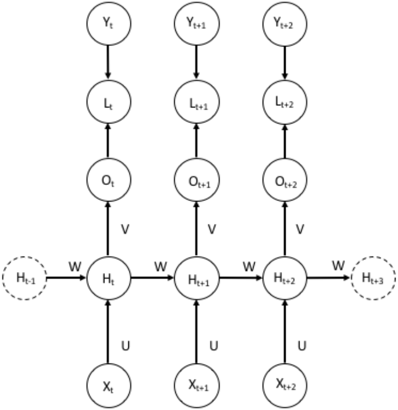
- Stochastic Gradient Descent
- Backpropagation through time
- Unroll the computation graph



Training



Training

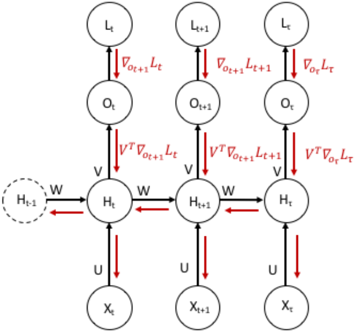


$$L = \sum_t L_t$$

$$\frac{\partial L}{\partial L_t} = 1$$

$$\nabla_{O_t} L = \frac{\partial L}{\partial O_t} = \frac{\partial L}{\partial L_t} \frac{\partial L_t}{\partial O_t}$$

Training



$$\nabla_{h_\tau} L = V^\top \nabla_{O_\tau} L_\tau$$

$$\nabla_{h_t} L = \left(\frac{\partial h_{t+1}}{\partial h_t}\right)^\top (\nabla_{h_{t+1}} L) + \left(\frac{\partial O_t}{\partial h_t}\right)^\top \nabla_{O_t} L$$

$$\frac{\partial h_{t+1}}{\partial h_t} = W^\top diag(\phi'(h_{t+1}))$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} W^\top diag(\phi'(h_{i-1}))$$

Vanilla RNN drawbacks

- Long term dependencies [1,2]
- Vanishing Gradient
- Exploding Gradient
- Jacobian terms multiply many time

[1] Hochreiter, Sepp, and Jrgen Schmidhuber. "Long short-term memory." Neural computation 9.8 (1997): 1735-1780.

[2] Bengio, Yoshua, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." IEEE transactions on neural networks 5.2 (1994): 157-166.

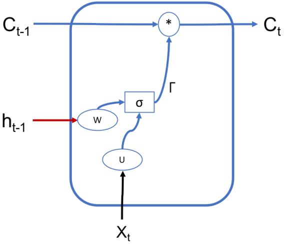
RNN Gating

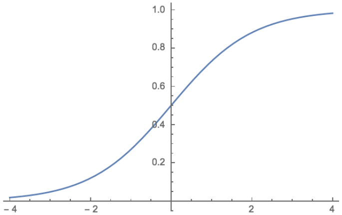
Protect the state of the RNN

Rather than updating the state with each datapoint

- Learn when to update, given the input and the previous hidden state
- What to update given the input and the previous state
- Even more so, what to remove (forget)
- What to add into the memory

Gating mechanism

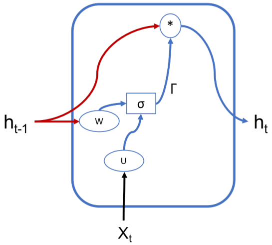




$$\Gamma = \sigma(Wh_{t-1} + Ux_t + b)$$

$$C_t = \Gamma * C_{t-1}$$

Gating mechanism

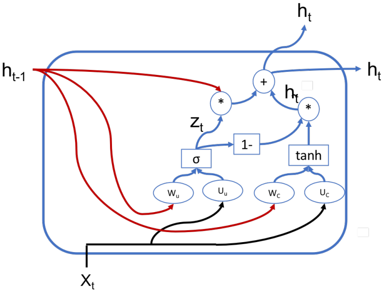


$$\Gamma = \sigma(Wh_{t-1} + Ux_t + b)$$

$$h_t = \Gamma * h_{t-1}$$

Gated Recurrent Unit

- Memory cell h (c)



$$\tilde{h}_t = \tanh(W_c h_{t-1} + U_c x_t + b_c)$$

$$z_t = \sigma(W_u h_{t-1} + U_u x_t + b_u)$$

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t$$

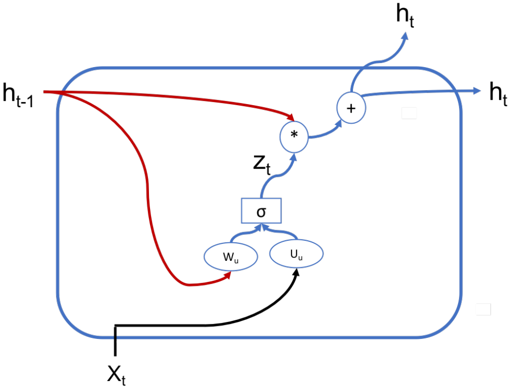
$$o_t = C + V h_t$$

$$\hat{y}_t = softmax(o_t)$$

[3]

[3] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." arXiv preprint arXiv:1406.1078 (2014).

Gated Recurrent Unit (forget)



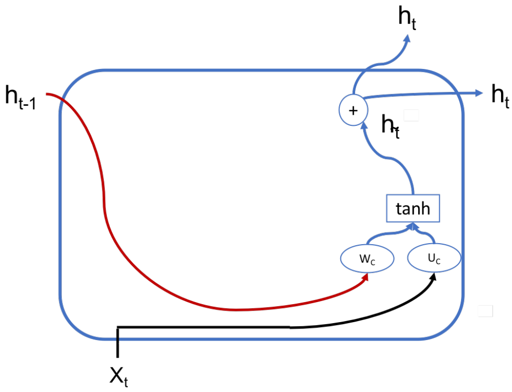
$$z_t = \sigma(W_u h_{t-1} + U_u x_t + b_u)$$

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t$$

$$o_t = C + V h_t$$

$$\hat{y}_t = softmax(o_t)$$

Gated Recurrent Unit (add)

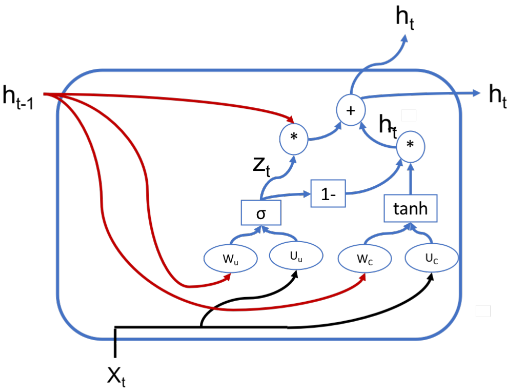


$$\tilde{h}_t = \tanh(W_c h_{t-1} + U_c x_t + b_c)$$

$$o_t = C + Vh_t$$

$$\hat{y}_t = softmax(o_t)$$

Gated Recurrent Unit



$$\tilde{h}_t = \tanh(W_c h_{t-1} + U_c x_t + b_c)$$

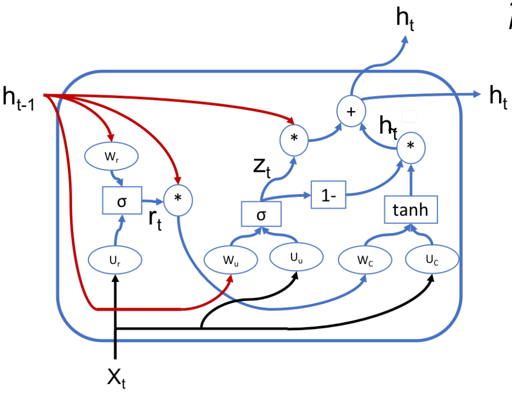
$$z_t = \sigma(W_u h_{t-1} + U_u x_t + b_u)$$

$$h_t = z_t \cdot h_{t-1} + (1 - z_t) \cdot \tilde{h}_t$$

$$o_t = C + V h_t$$

$$\hat{y}_t = softmax(o_t)$$

Gated Recurrent Unit (Full)



$$\tilde{h}_t = \tanh(W_c r_t \cdot h_{t-1} + U_c x_t + b_c)$$

$$z_t = \sigma(W_c h_{t-1} + U_c x_{(t)} + b_c)$$

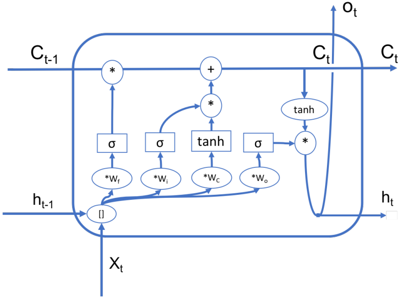
$$r_t = \sigma(W_r h_{t-1} + U_r x_t + b_r)$$

$$h_t = z_t \cdot \tilde{h}_t + (1 - z_t) \cdot h_{t-1}$$

$$o_t = C + V h_t$$

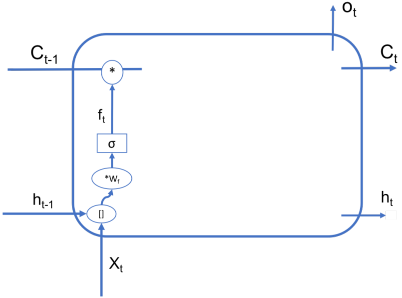
$$\hat{y}_t = softmax(o_t)$$

Long short term memory



Long short term memory

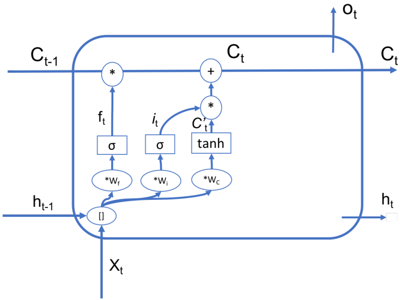
Forget gate



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

Long short term memory

Add gate

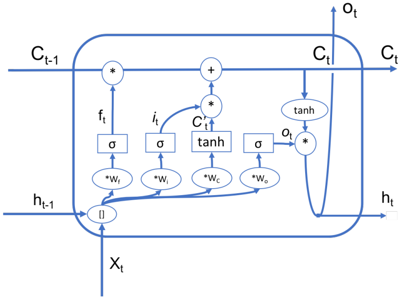


$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$C'_t = \sigma(W_C \cdot [h_{t-1}, x_t] + b_C)$$

Long short term memory

Output gate



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

The output is:

- based on the cell content
- filtered by the output gate
- activation of the hidden state and the input

the output is propagate to the next step as well