Sequence modelling with RNNs: Main ideas

Dr. Benjamin Roth

CIS LMU München

Recurrent Neural Networks (RNNs): Representation Learning for Sequences

- Family of neural networks for processing **sequential data** $\mathbf{x}^{(1)} \dots \mathbf{x}^{(\tau)}$.
  $\Rightarrow$ Sequences of words, characters, ...
- Simplest case: for each time step t there is a representation $\mathbf{h}^{(t)}$ computed from current input $\mathbf{x}^{(t)}$ and previous representation $\mathbf{h}^{(t-1)}$.
  Extensions:
  - Representation at time t can be complex, e.g. several layers.
  - Representation $\mathbf{h}^{(t)}$ can depend on more than one previous $\mathbf{h}$.
  - A context window of inputs can be used for computation of $\mathbf{h}^{(t)}$, e.g. $\mathbf{x}^{(t-2)}, \mathbf{x}^{(t-1)}, \mathbf{x}^{(t)}, \mathbf{x}^{(t+1)}\mathbf{x}^{(t+2)}$.

Recurrent Neural Networks (RNNs): Parameter Sharing

- Parameter sharing: going from a time step $t-1$ to t is parameterized the same for all t.

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)}; \theta)$$

- Representation $\mathbf{h}^{(t)}$ contains (some) information from all previous time steps.

Recurrent Neural Networks (RNNs): Output

- The output of a standard RNN is computed from the hidden representation at time t:

$$\mathbf{o}^{(t)} = f(\mathbf{h}^{(t)}; \theta)$$

- Some RNNs architectures provide output $\mathbf{o}^{(t)}$ at every time step, other architectures only at the last time step ($\mathbf{o}^{(\tau)}$)
  - Every time step: Tagging (POS Tagging, NER, ...)
  - Last time step: Sentence classification (Sentiment polarity, ...)

Recurrent Neural Networks (RNNs): Loss Function

- Loss function that drives training:
  - Every time step: $\mathcal{L}(y^{(1)}, y^{(2)} \dots y^{(\tau)}; o^{(1)}, o^{(2)} \dots o^{(\tau)})$
  - Last time step: $\mathcal{L}(y; o^{(\tau)})$
- Example: Tagging
  - Output **o** (for training) is predicted distribution over tags
    - $\mathbf{o}^{(t)} = P(\text{tag} = ?|\mathbf{h}^{(t)}; \theta)$
    - Can be softmax, for example
  - Loss at time t is e.g. negative log-likelihood (NLL) of training label $y^{(t)}$

$$\mathcal{L}^{(t)} = -\log P(\text{tag} = y^{(t)}|\mathbf{h}^{(t)}; \theta)$$

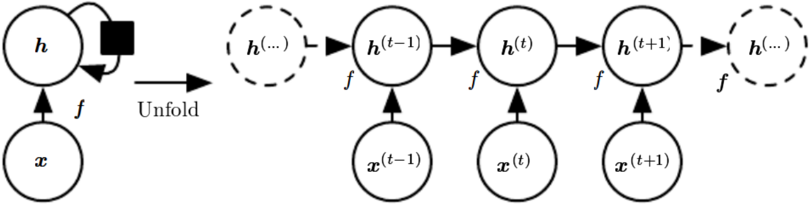
  - Overall NLL-loss:

$$\mathcal{L} = \sum_{t=1}^{\tau} \mathcal{L}^{(t)}$$

  - At test time, output and loss are different
    - Output: most likely label at t
    - Loss: Accuracy, F1-score, ...

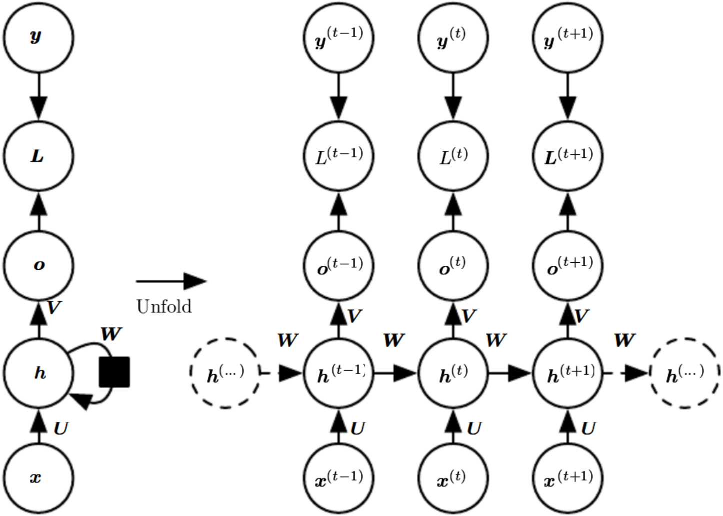
Graphical Notation

- Nodes indicate input data (**x**) or function outputs (otherwise).
- Arrows indicate functions arguments.
- Compact notation (left):
  - All time steps conflated.
  - Black square ■ indicates *"delay"* of 1 time unit.



Source: Goodfellow et al.: Deep Learning.

Graphical Notation: Including Output and Loss Function



Source: Goodfellow et al.: Deep Learning.