

Recurrent Neural Networks

Hankui Zhuo

May 17, 2019

<http://xplan-lab.org>

Contents

1. introduction
2. unfolding computational graphs
3. recurrent neural networks structures
4. computing the gradient

Introduction

- RNNs
 - A family of neural networks processing sequential data, such as,
 - a sequence of values $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\tau)}$
 - can scale to much longer sequences, e.g.,
 - text
 - Voice
 - ...
 - can also process sequences of variable length, e.g.,
 - 10 words or 10000 words? -- piece of cake
 - 1min voice or 5000min voice? – piece of cake
 - ...

Overall idea



advantages

Sharing parameters across different parts of a model:

- extend and apply the model to examples of different forms (variable lengths)



disadvantages

Having separate parameters:

- cannot generalize to sequence lengths not seen during training
- cannot share statistical strength across different sequence lengths
- Cannot share statistical strength across across different positions in time

Overall idea

- Particularly important when a specific piece of information can occur at **multiple positions** within the sequence.

“ I went to Nepal in **2009** ” “In **2009**, I went to Nepal.”

Question:

When did the narrator go to Nepal?

Answer:

“**2009**”, no matter it appears in the sixth word
or second word

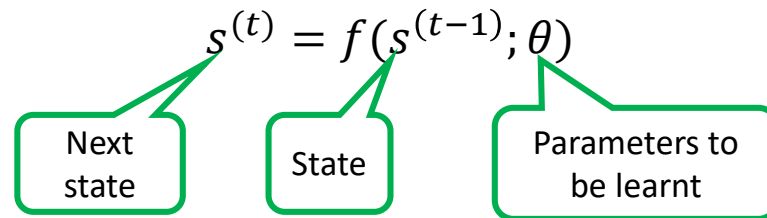
Overall idea

- Suppose that we trained a feedforward network that processes sentences of fixed length.
 - A traditional fully connected feedforward network would have **separate parameters** for each input feature
 - It would need to learn **all of the rules of the language** separately **at each position** in the sentence.



Unfolding computational graphs

- Unfolding a **recursive or recurrent computation** into a **computational graph** that has a repetitive structure
- Consider the classical form of a dynamical system



- It is **recurrent** because the definition of s at time t refers back to the same definition at time $t - 1$

How to unfold the system

- For example, if we unfold the system for $\tau = 3$ time steps, we obtain

$$s^{(3)} = f(s^{(2)}; \theta) = f(f(s^{(1)}; \theta); \theta)$$



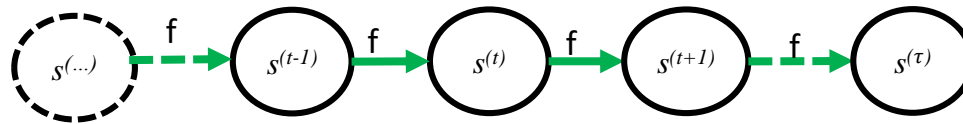
They are
shared

- For a finite number of time steps τ , the graph can be unfolded by applying the definition $\tau - 1$ times

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

How to unfold the system

- Unfolding the equation by **repeatedly** applying the definition in this way has yielded an expression that does **not involve recurrence**.
- Such an expression can now be represented by a traditional directed **acyclic computational graph**.



With external signal

- a dynamical system driven by an external signal $x^{(t)}$

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

- To indicate the state is the hidden units of the network, we use the variable h to represent the state

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

- Typical RNNs will add extra architectural features such that output layers that **read information out of the state h** to make predictions

Lossy summary: hidden state h

- It maps an arbitrary length sequence $(x(t), x(t-1), x(t-2), \dots, x(2), x(1))$ to a fixed length vector $h(t)$

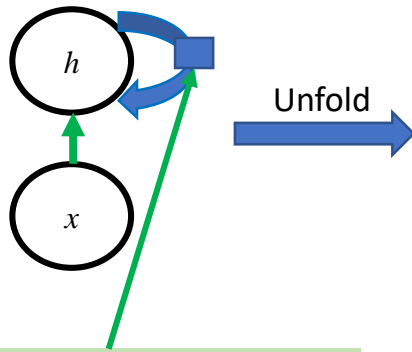
$$\begin{aligned} h^{(t)} &= g^{(t)}(x^{(t)}, x^{(t-1)}, x^{(t-2)}, \dots, x^{(2)}, x^{(1)}) \\ &= f(h^{(t-1)}, x^{(t)}; \theta) \end{aligned}$$

- Depending on the training criterion, this summary might selectively keep **some aspects** of the past sequence with more precision than other aspects.

For example, in language modelling, it prefers to predict the **next word** given previous words

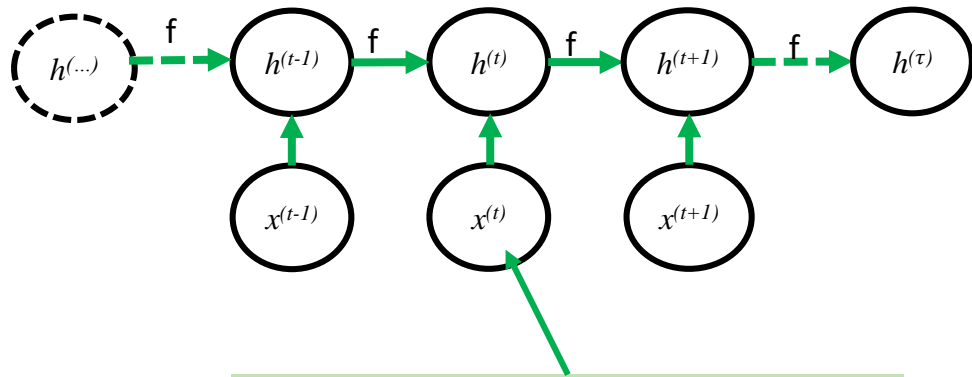
Two ways

Circuit diagram



a delay of a single time step

unfolded computational graph



associated with one particular time instance

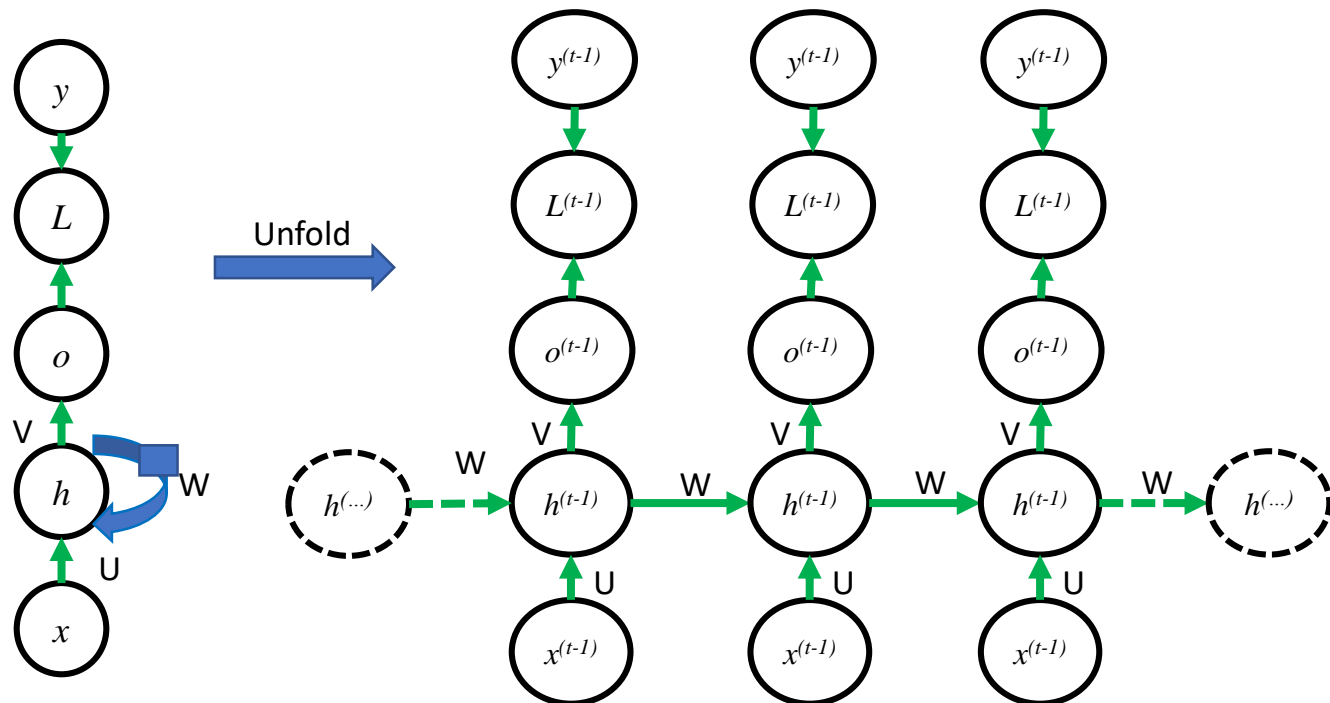
Advantages/disadvantages

- The recurrent graph is **succinct**.
- The unfolded graph:
 - provides an **explicit description** of which computations to perform.
 - helps to **illustrate the idea** of information flow **forward in time** (computing outputs and losses)
 - **backward in time** (computing gradients) by explicitly showing the path along which this information flows

RNNs structure

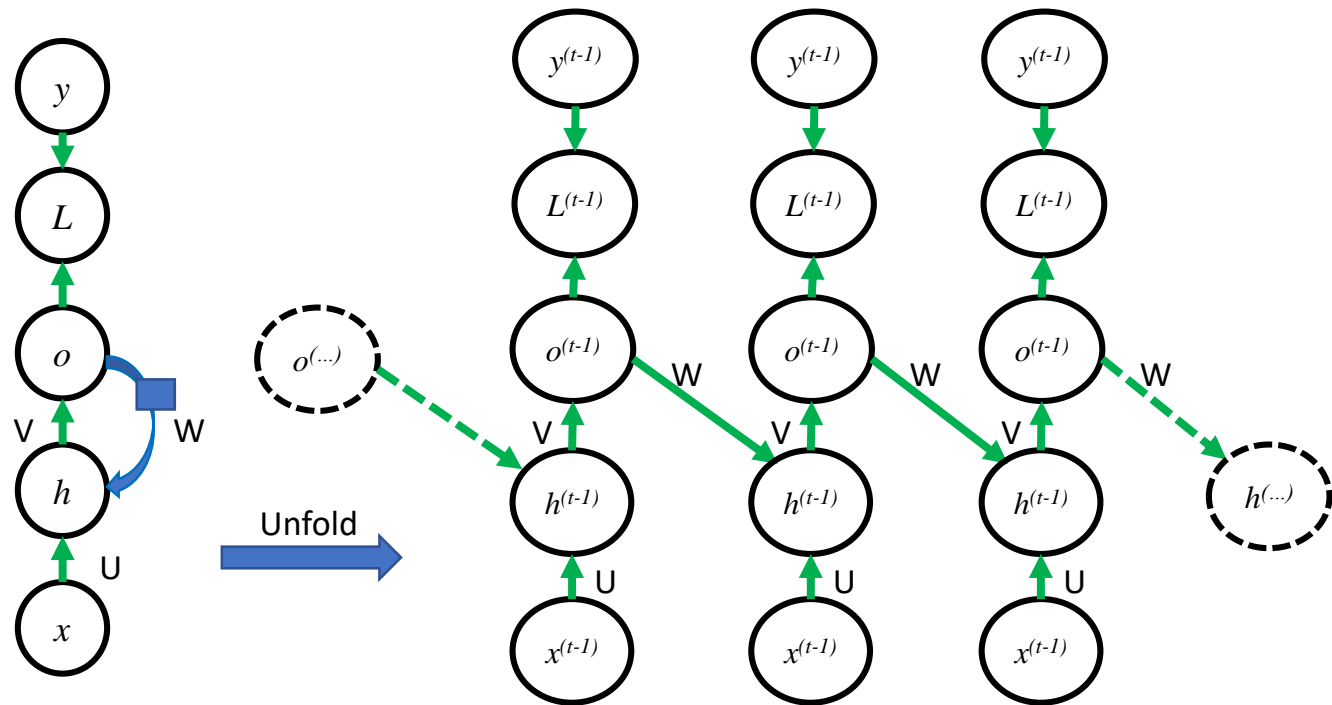
- With the graph unrolling and parameter sharing ideas, we can design a wide variety of recurrent neural networks

- Recurrent networks that produce an output at each time step and have recurrent connections between hidden units



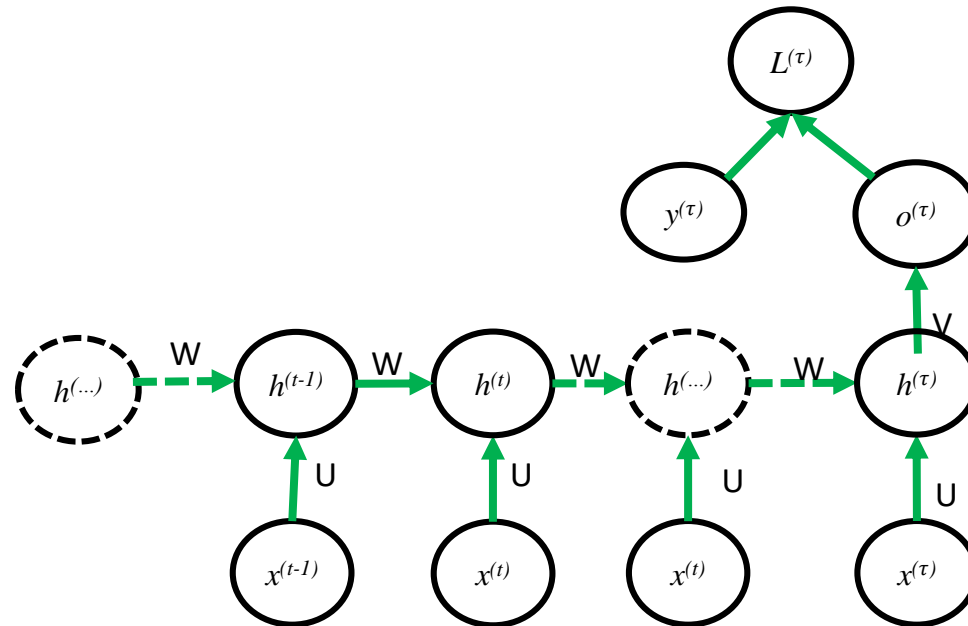
RNNs structure

2. Recurrent networks that produce an output at each time step and have recurrent connections only from **the output** at one time step to **the hidden units** at the next time step



RNNs structure

3. Recurrent networks with recurrent connections between hidden units, that read an **entire sequence** and then produce a **single output**



RNNs update equations

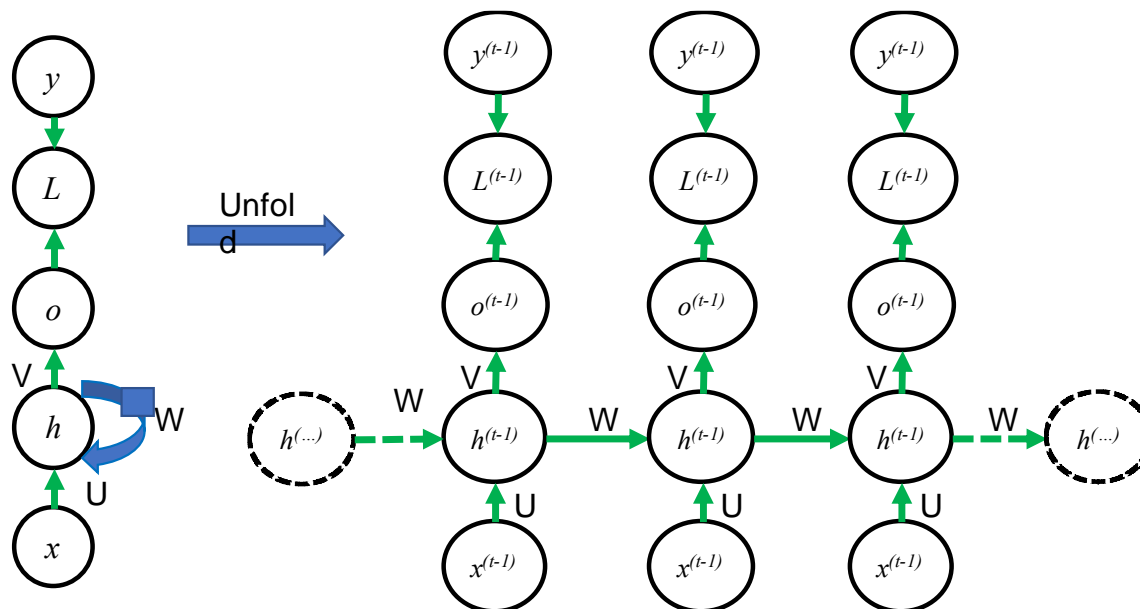
- The forward propagation equations regarding the first type of RNNs

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \quad \hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{o}^{(t)})$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)})$$


$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}$$

$$\hat{y}_i^{(t)} = [\text{softmax}(\mathbf{o}^{(t)})]_i = \frac{\exp(o_i^{(t)})}{\sum_j \exp(o_j^{(t)})}$$



Total loss

The total loss for a given sequence of x values paired with a sequence of y values would then be just the sum of the losses over all the time steps. For example, if $L(t)$ is the negative log-likelihood of $y^{(t)}$ given $x(1), \dots, x(t)$, then

$$L(\{x^{(1)}, \dots, x^{(\tau)}\}, \{y^{(1)}, \dots, y^{(\tau)}\}) \\ = \sum_t L^{(t)} = - \sum_t \log p_{\text{model}}(y^{(t)} | \{x^{(1)}, \dots, x^{(\tau)}\})$$


is given by reading the entry for $y^{(t)}$ from the model's output vector $\hat{y}^{(t)}$

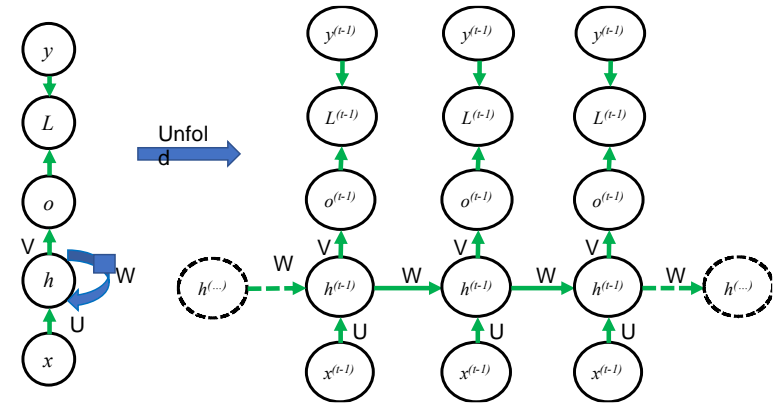
Computing the gradient

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \quad \hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{o}^{(t)})$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)})$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}$$

$$\hat{y}_i^{(t)} = [\text{softmax}(\mathbf{o}^{(t)})]_i = \frac{\exp(o_i^{(t)})}{\sum_j \exp(o_j^{(t)})}$$



$$(\nabla_{o^{(t)}} L)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i,y^{(t)}}$$

$$\nabla_{h^{(t)}} L = V^T \nabla_{o^{(t)}} L$$

$$\begin{aligned} \nabla_{h^{(t)}} L &= \left(\frac{\partial h^{(t+1)}}{\partial h^{(t)}} \right)^T (\nabla_{h^{(t+1)}} L) + \left(\frac{\partial o^{(t)}}{\partial h^{(t)}} \right)^T (\nabla_{o^{(t)}} L) \\ &= W^T (\nabla_{h^{(t+1)}} L) \text{diag}(1 - (h^{(t+1)})^2) + V^T (\nabla_{o^{(t)}} L) \end{aligned}$$

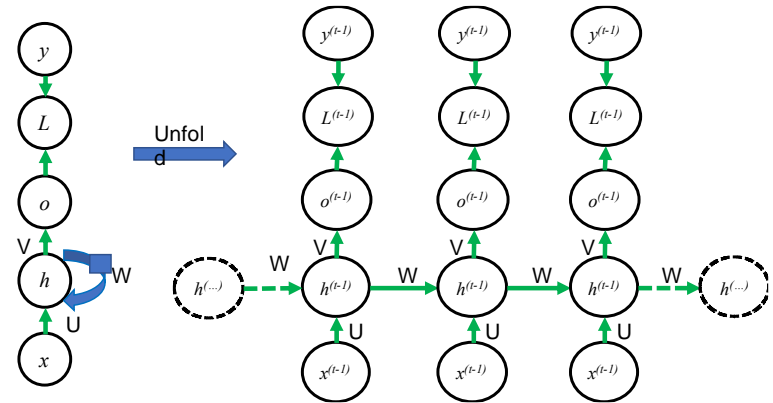
Computing the gradient

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \quad \hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{o}^{(t)})$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)})$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}$$

$$\hat{y}_i^{(t)} = [\text{softmax}(\mathbf{o}^{(t)})]_i = \frac{\exp(o_i^{(t)})}{\sum_j \exp(o_j^{(t)})}$$



$$\nabla_c L = \sum_t \left(\frac{\partial o^{(t)}}{\partial c} \right) \quad \nabla_{o^{(t)}} L = \sum_t \nabla_{o^{(t)}} L$$

$$\nabla_b L = \sum_t \left(\frac{\partial h^{(t)}}{\partial b^{(t)}} \right)^T \nabla_{h^{(t)}} L = \sum_t \text{diag} \left(1 - (h^{(t)})^2 \right) \nabla_{h^{(t)}} L$$

$$\nabla_V L = \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}} \right) \nabla_V o_i^{(t)} = \sum_t (\nabla_{o^{(t)}} L) h^{(t)T}$$

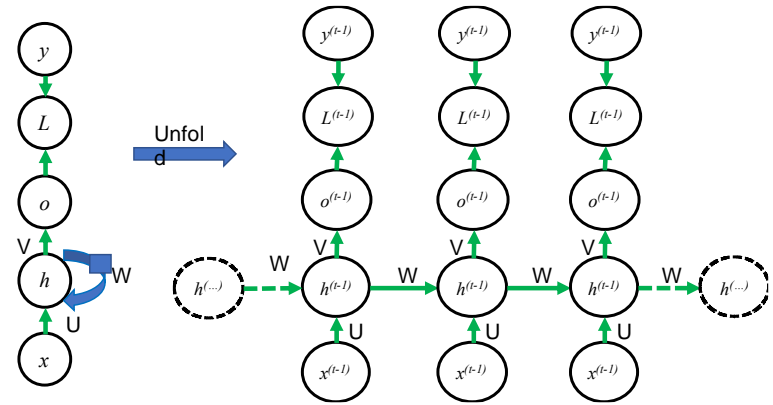
Computing the gradient

$$\mathbf{a}^{(t)} = \mathbf{b} + \mathbf{W}\mathbf{h}^{(t-1)} + \mathbf{U}\mathbf{x}^{(t)} \quad \hat{\mathbf{y}}^{(t)} = \text{softmax}(\mathbf{o}^{(t)})$$

$$\mathbf{h}^{(t)} = \tanh(\mathbf{a}^{(t)})$$

$$\mathbf{o}^{(t)} = \mathbf{c} + \mathbf{V}\mathbf{h}^{(t)}$$

$$\hat{y}_i^{(t)} = [\text{softmax}(\mathbf{o}^{(t)})]_i = \frac{\exp(\mathbf{o}_i^{(t)})}{\sum_j \exp(\mathbf{o}_j^{(t)})}$$



$$\begin{aligned} \nabla_W L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{W^{(t)}} h_i^{(t)} \\ &= \sum_t \text{diag} \left(1 - (h^{(t)})^2 \right) (\nabla_{h^{(t)}} L) h^{(t-1)T} \end{aligned}$$

$$\begin{aligned} \nabla_U L &= \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{U^{(t)}} h_i^{(t)} \\ &= \sum_t \text{diag} \left(1 - (h^{(t)})^2 \right) (\nabla_{h^{(t)}} L) x^{(t)T} \end{aligned}$$

To be continued!