Recurrent Neural Networks Notes #1

I: Modelling sequences requires:

(α): deal with variable-length sequences

(β): maintain sequence order

(γ): keep track of long-term dependencies

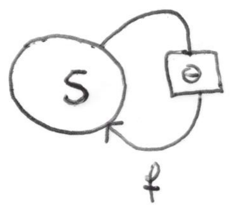
(δ): share parameters across the sequence

II: Classical form of an autonomous dynamical system:

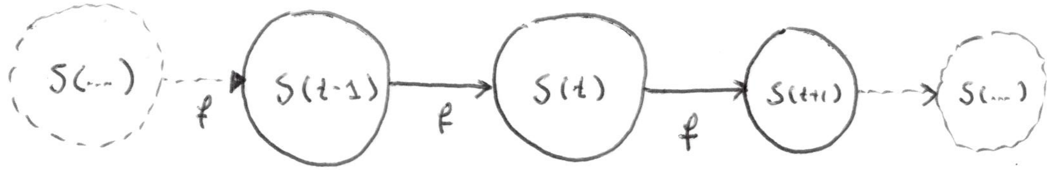
$$\boxed{\underline{s}(t) = f(\underline{s}(t-1); \underline{\theta})} \quad t \geq 1$$

where $\underline{s}(t)$ is the state-vector of the system and $\underline{\theta}$ is a vector of internal constant parameters

III: Computational Graph of Classical Dynamical System:



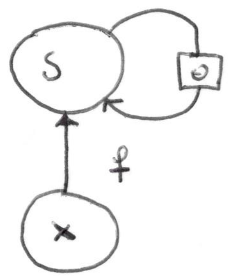
IV: Unfolded Computational Graph of Classical Dynamical System:



V: Autonomous Dynamical System Driven by External Signal $\underline{x}(t)$:

$$\boxed{\underline{s}(t) = f(\underline{s}(t-1), \underline{x}(t); \underline{\theta}) \quad t \geq 1}$$

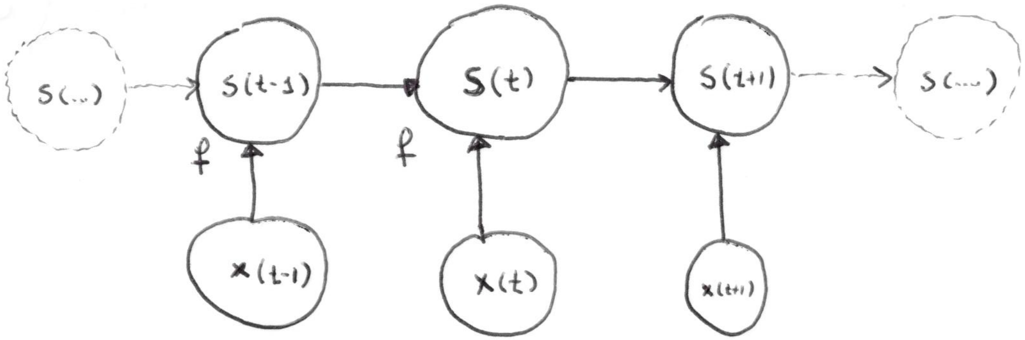
VI: Computational Graph of an Autonomous Dynamical System Driven by External Signal $\underline{x}(t)$:



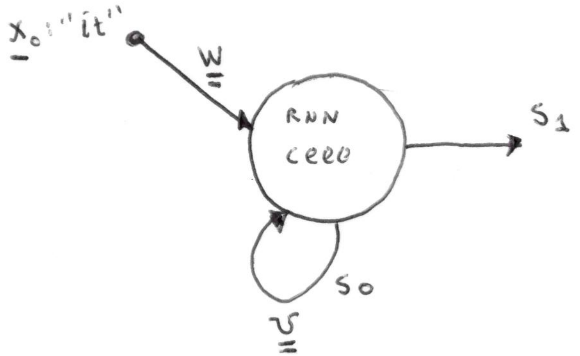
VIII: Many RNNs can be described by a similar equation, such as:

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

VII: Unfolded Computational Graph:



IX: RNNs are capable of remembering their previous state:



x_0: vector representing first word

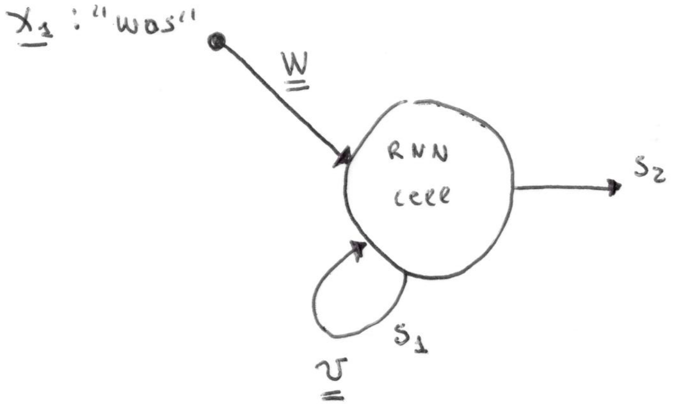
S_0: cell state at $t = \emptyset$ (some initialization is required)

S_1: cell state at $t = 1$

W, U: are weight matrices

(*) Update equation for next cell state:

$$\boxed{S_1 = \tanh(W x_0 + U S_0)}$$



x_1: vector representing second word

S_1: cell state at $t=1$

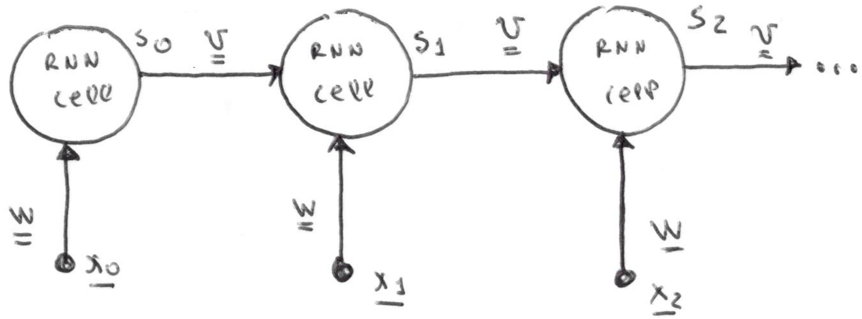
S_2: cell state at $t=2$

$$\boxed{S_2 = \tanh(W x_1 + U S_1)}$$

W, U: weight matrices

X: Unfolding the RNN across time:

(★) In practise, one may have many hidden units and many layers of hidden units.



(α): Notice that the same parameter matrices $\underline{\underline{W}}$, $\underline{\underline{V}}$ are used.

(β): s_n can contain information from all past timesteps.

XI: Training RNN with Back Propagation Through Time (BBTT)

(α): Take the derivative of the loss (gradient) with respect to each parameter.

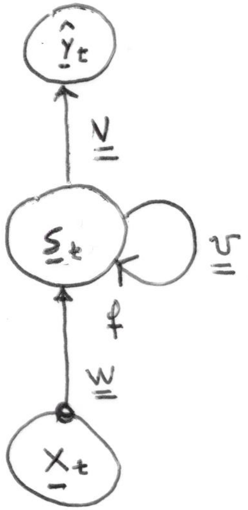
(β): shift parameters to the opposite direction in order to minimize the loss.

} Gradient Descend!!!

XII: Let $\mathcal{X} = \{\underline{x}_1, \underline{x}_2, \dots, \underline{x}_z\}$ a sequence of ℓ-dimensional input vectors $(\underline{x}_t \in \mathbb{R}^{\ell} \; \forall t \in [z])$

Let $\mathcal{Y} = \{\underline{y}_1, \underline{y}_2, \dots, \underline{y}_z\}$ the corresponding sequence of target output vectors $(\underline{y}_t \in \mathbb{R}^m, \forall t \in [z])$

(★) The estimated output of the RNN cell at each time step may be computed as:



(A): Update Hidden State:

$$\underline{s}_t = f(\underline{\underline{W}} \circ \underline{x}_t + \underline{\underline{U}} \circ \underline{s}_{t-1}) \quad (A)$$

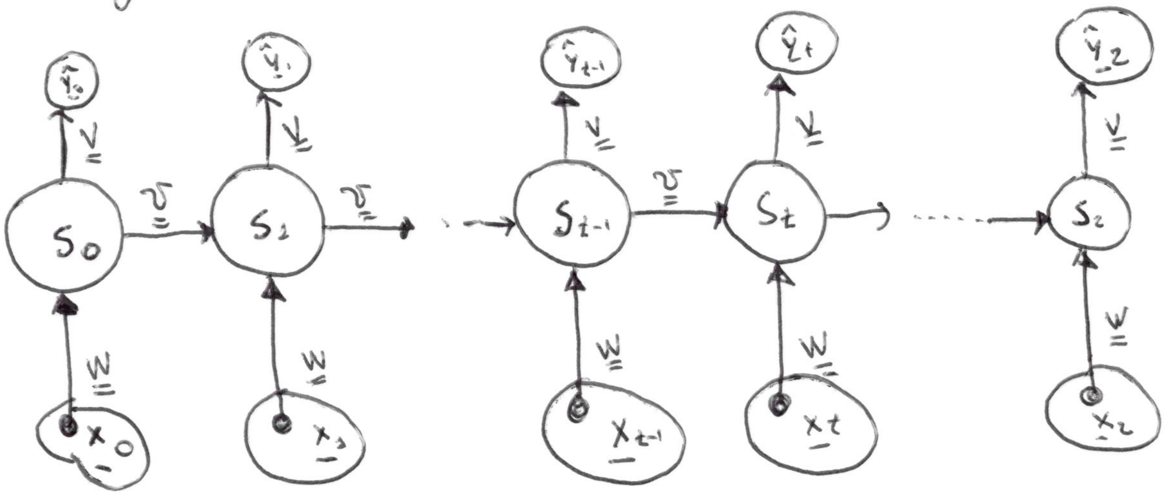
where (★) $f(u) = \tanh(u) = \dfrac{e^{u} - e^{-u}}{e^{u} + e^{-u}}$

(B): Output Vector:

$$\hat{\underline{y}}_t = \underline{\underline{V}} \circ \underline{s}_t \quad (B)$$

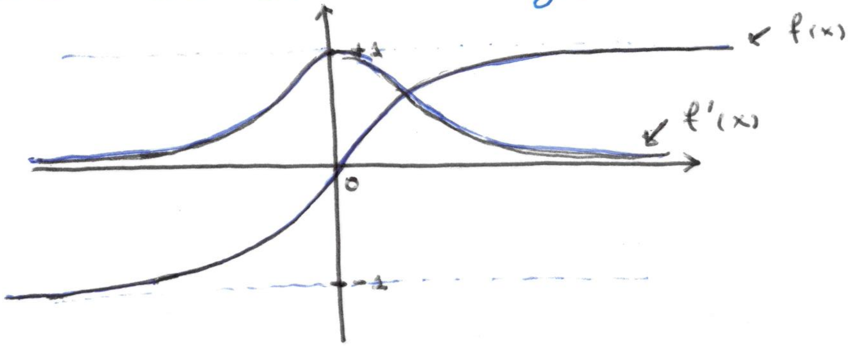
(★) By unfolding the RNN cell for z time steps we get:

Unfolded Description of the RNN cell!!!



★ A very helpful result relates to expressing the derivative of a given activation function as a function of itself. Let's consider the hyperbolic tangent function for example. #6

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$ [SIGMOID FORM]



• Express the derivative $f'(x)$ as a function of $f(x)$?

Let $p(x) = e^{x} - e^{-x}$ and $q(x) = e^{x} + e^{-x}$, so that $f(x) = \frac{p(x)}{q(x)}$.

We may observe that: (i): $p'(x) = e^{x} + e^{-x} = q(x)$

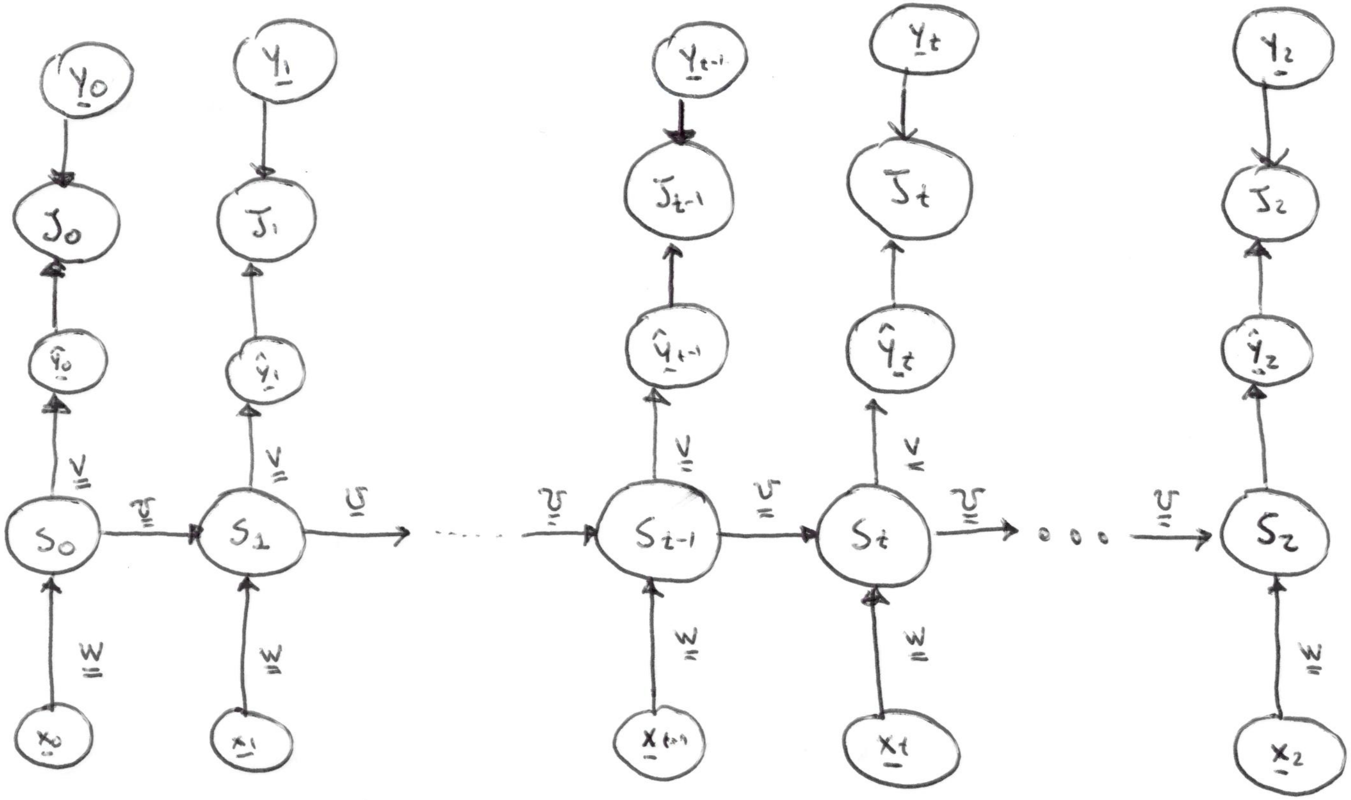
(ii): $q'(x) = e^{x} - e^{-x} = p(x)$

Thus, we have that: $f'(x) = \frac{p'(x)q(x) - p(x)q'(x)}{q^2(x)} \Rightarrow$

$$f'(x) = \frac{q^2(x) - p^2(x)}{q^2(x)} \Rightarrow f'(x) = 1 - \frac{p^2(x)}{q^2(x)} \Rightarrow$$

$$f'(x) = 1 - \left(\frac{p(x)}{q(x)}\right)^2 \Rightarrow \boxed{f'(x) = 1 - f^2(x)}$$

⊛ Since we are making a prediction at each time step, we have to associate a loss at each timestep too;



Unfolded RNN for each time-step $0 \le t \le z$.

Updating Equations for each time-step $1 \le t \le z$.

$$\underline{s}(t) = f(\underline{\underline{W}} \circ \underline{x}(t) + \underline{\underline{U}} \circ \underline{s}(t-1) + \underline{b})$$

$$\underline{\hat{y}}(t) = \underline{\underline{V}} \circ \underline{s}(t) + \underline{c}$$

⊛ Bias-terms may be associated with each updating equation.

⊛ Form the expression for the total loss of the RNN throughout the complete sequence $\mathcal{X}$:

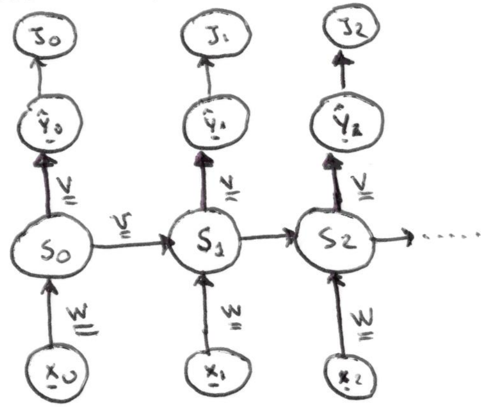
$$\boxed{J(\underline{\theta}) = \sum_{t=0}^{z} J_t(\underline{\theta})}$$

Total Loss

(★) Compute the total gradient by summing the partial derivative w.r.t each parameter of the RNN foreach time-step:

$$\boxed{\frac{\partial J}{\partial \underline{P}} = \sum_t \frac{\partial J_t}{\partial \underline{P}}} \quad (1)$$

KEEP IN MIND THE NETWORK STRUCTURE FOR 2 TIMESTEPS



$$\begin{cases} \underline{s}_t = f(\underline{W} \cdot \underline{x}_t + \underline{U} \cdot \underline{s}_{t-1}) \\ \hat{\underline{y}}_t = \underline{V} \cdot \underline{s}_t \end{cases} \quad \boxed{t \geq 1}$$

(★) Let's initially focus on the weight-matrix $\underline{W}$:

$$\boxed{\frac{\partial J}{\partial \underline{W}} = \sum_t \frac{\partial J_t}{\partial \underline{W}}} \quad (2)$$

(★) Let's focus on a single time step where $t=2$:

$$\boxed{\frac{\partial J_2}{\partial \underline{W}} = \frac{\partial J_2}{\partial \hat{\underline{y}}_2} \cdot \frac{\partial \hat{\underline{y}}_2}{\partial \underline{s}_2} \cdot \frac{\partial \underline{s}_2}{\partial \underline{W}}} \quad (3)$$

CHAIN RULE OF DERIVATIVES.

But, s_2 may be written as $\boxed{\underline{s}_2 = f(\underline{W} \cdot \underline{x}_2 + \underline{U} \cdot \underline{s}_1)}$ (4)

where $\underline{s}_1$ also depends on $\underline{W}$. Therefore, the term $\frac{\partial s_2}{\partial \underline{W}}$ cannot be treated as a constant.

(★) We need to know how exactly the cell-state at time $t=2$ depends on the weight-matrix $\underline{W}$:

(★) In other words, we need to disambiguate the term $\frac{\partial s_2}{\partial \underline{W}}$ for which we need to express its total derivative with respect to $\underline{W}$ as:

$$\boxed{\frac{d s_2}{d \underline{W}} = ?}$$

⊛ Eq. (3) suggests that:

$$\boxed{s_2 = s_2\left(s_1(\underline{\underline{W}}), s_0(\underline{\underline{W}}); \underline{\underline{W}}\right) \quad (5)}$$

⊛ Thus, the total derivative of s_2 w.r.t. $\underline{\underline{W}}$ may be written as: (chain rule for composite functions)

$$\boxed{\frac{d s_2}{d \underline{\underline{W}}} = \frac{\partial s_2}{\partial \underline{\underline{W}}} + \frac{\partial s_2}{\partial s_1} \cdot \frac{\partial s_1}{\partial \underline{\underline{W}}} + \frac{\partial s_2}{\partial s_0} \cdot \frac{\partial s_0}{\partial \underline{\underline{W}}} \quad (6)}$$

⊛ Eq. (6), suggests the following compact form:

$$\boxed{\frac{d s_2}{d \underline{\underline{W}}} = \sum_{k=0}^{2} \frac{\partial s_2}{\partial s_k} \cdot \frac{\partial s_k}{\partial \underline{\underline{W}}} \quad (7)}$$

⊛ Apparently, the summation term for $k=2$ yields
$$\boxed{\frac{\partial s_2}{\partial s_2} \cdot \frac{\partial s_2}{\partial \underline{\underline{W}}} = \frac{\partial s_2}{\partial \underline{\underline{W}}}}$$

⊛ Finally, Eq. (3) becomes:

$$\boxed{\frac{\partial J_2}{\partial \underline{\underline{W}}} = \sum_{k=0}^{2} \frac{\partial J_2}{\partial \hat{y}_2} \cdot \frac{\partial \hat{y}_2}{\partial s_2} \cdot \frac{\partial s_2}{\partial s_k} \cdot \underbrace{\frac{\partial s_k}{\partial \underline{\underline{W}}}}_{\substack{\text{Contributions of } \underline{\underline{W}} \\ \text{in previous time-steps} \\ \text{to the error at time-step } t=2}} \quad (8)}$$

⊛ In this setting, the general form of Eq. (8) at any given time-step t may be given as:

$$\boxed{\frac{\partial J_t}{\partial \underline{\underline{W}}} = \sum_{k=0}^{t} \frac{\partial J_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \underbrace{\frac{\partial s_t}{\partial s_k} \cdot \frac{\partial s_k}{\partial \underline{\underline{W}}}}_{\substack{\text{Contribution of } \underline{\underline{W}} \\ \text{in previous timesteps to} \\ \text{the error at timestep } t.}} \quad (9)}$$

⊛ What about $\frac{\partial J}{\partial \underline{\underline{U}}}$ and $\frac{\partial J}{\partial \underline{\underline{V}}}$?

⊛ $$\boxed{\frac{\partial J}{\partial \underline{\underline{V}}} = \sum_{t} \frac{\partial J_t}{\partial \underline{\underline{V}}} \quad (10)}$$

⊛ $$\boxed{\frac{\partial J}{\partial \underline{\underline{U}}} = \sum_{t} \frac{\partial J_t}{\partial \underline{\underline{U}}} \quad (11)}$$

⊛ We have to aknowledge that

(i): S_t does depend on $\underline{\underline{U}}$

(ii): S_t does not depend on $\underline{\underline{V}}$

- Thus, we may write that:

$$\boxed{\frac{\partial J_t}{\partial \underline{\underline{V}}} = \frac{\partial J_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial \underline{\underline{V}}} \quad (12)}$$

- For the case of the weight-matrix $\underline{\underline{U}}$ we may write that:

$$\frac{\partial J_t}{\partial \underline{\underline{U}}} = \frac{\partial J_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial \underline{\underline{U}}} \quad (13)$$

- We need to take into consideration the contribution of $\underline{\underline{U}}$ in the previous timesteps to the error at timestamp t.

- To do so, we have to formulate the explicit dependence of each s_t on $\underline{\underline{U}}$ alongside with the implicit dependence of each s_t on the previous state vectors $s_{t-1}, s_{t-2}, \ldots, s_1, s_0$.

- Therefore, we may write that:

$$\boxed{s_t = s_t\left(s_{t-1}(\underline{\underline{U}}), s_{t-2}(\underline{\underline{U}}), \ldots, s_0(\underline{\underline{U}}); \underline{\underline{U}}\right) \quad (14)}$$

- We need the total derivative counterpart of the quantity, $\frac{d s_t}{d \underline{\underline{U}}}$

- Eq.(14) suggests that we may write:

$$\frac{d s_t}{d \underline{\underline{U}}} = \sum_{k=0}^{t} \frac{\partial s_t}{\partial s_k} \cdot \frac{\partial s_k}{\partial \underline{\underline{U}}} \quad (15)$$

- Finally, by combining Eqs.(13) and (15) we get:

$$\boxed{\frac{\partial J_t}{\partial \underline{\underline{U}}} = \sum_{k=0}^{t} \frac{\partial J_t}{\partial \hat{y}_t} \cdot \frac{\partial \hat{y}_t}{\partial s_t} \cdot \frac{\partial s_t}{\partial s_k} \cdot \frac{\partial s_k}{\partial \underline{\underline{U}}} \quad (16)}$$

- Next Question: "Why RNNs are so hard to train?"

In Eqs.(9) and (16) share a common factor $\frac{\partial s_t}{\partial s_k}$ which may be written as:

$$\boxed{\frac{\partial s_t}{\partial s_k} = \prod_{m=k+1}^{t} \frac{\partial s_m}{\partial s_{m-1}} \quad (17)}$$

⊛ The problem of vanishing gradient:

According to Eq. (16), the gradient of the loss function at time (t) requires the computation of the term

$$\boxed{\frac{\partial \underline{h}_t}{\partial \underline{h}_k}.} \quad [17]$$

For example for $t=2$ and $k=0$, we would have to compute $\frac{\partial \underline{h}_2}{\partial \underline{h}_0}$ which according to the interdependence of the various hidden states of the RNN can be written as:

$$\boxed{\frac{\partial \underline{h}_2}{\partial \underline{h}_0} = \frac{\partial \underline{h}_2}{\partial \underline{h}_1} \cdot \frac{\partial \underline{h}_1}{\partial \underline{h}_0}} \quad [19]$$

But, what if we are focusing on a timestep very far in the future?

$$\boxed{\frac{\partial \underline{h}_t}{\partial \underline{h}_0} = \frac{\partial \underline{h}_t}{\partial \underline{h}_{t-1}} \cdot \frac{\partial \underline{h}_{t-1}}{\partial \underline{h}_{t-2}} \cdot \dots \cdot \frac{\partial \underline{h}_3}{\partial \underline{h}_2} \cdot \frac{\partial \underline{h}_2}{\partial \underline{h}_1} \cdot \frac{\partial \underline{h}_1}{\partial \underline{h}_0}} \quad [19]$$

⊛ The general form of the term $\frac{\partial \underline{h}_t}{\partial \underline{h}_k}$ may be expressed as:

$$\boxed{\frac{\partial \underline{h}_t}{\partial \underline{h}_k} = \prod_{m=k+1}^{m=t} \frac{\partial \underline{h}_m}{\partial \underline{h}_{m-1}} \quad [20]}$$

which in turn reveals the necessity of computing the following term:

$$\boxed{\frac{\partial \underline{h}_t}{\partial \underline{h}_{t-1}} \quad [21]}$$

For the case of a trivial vanilla RNN with a single neuron, we may assume that:

(1): $\underline{x}_t \in \mathbb{R}^{\ell \times 1}, \quad 1 \leq t \leq 2$

(2): $y_t \in \mathbb{R}, \quad 1 \leq t \leq 2$

(3): $h_t \in \mathbb{R}, \quad 1 \leq t \leq 2 \quad (\underline{h}_t \equiv h_t)$

(4): $\underline{\underline{W}}_{xh} \in \mathbb{R}^{\ell \times 1}, \quad 1 \leq t \leq 2 \quad (\underline{\underline{W}}_{xh} \equiv \underline{W}_{xh})$

(5): $W_{hh} \in \mathbb{R}, \quad 1 \leq t \leq 2 \quad (\underline{\underline{W}}_{hh} \equiv W_{hh})$

(6): $W_{hy} \in \mathbb{R}, \quad 1 \leq t \leq 2 \quad (\underline{\underline{W}}_{hy} \equiv W_{hy})$

(*) According to the previous definitions, we may re-write the updating equations for the forward pass of the information within the network as:

$$\boxed{\begin{aligned} h_t &= f(\underline{W}_{xu}^T \cdot \underline{x}_t + W_{hh} \cdot h_{t-1}) \; [22] \\ y_t &= W_{hy} \cdot h_t \; [23] \end{aligned}}$$

(*) At this point, we must mention the content of Lecture Page #6 concerning the hyperbolic tangent and its derivative.

(*) In the light of the previous declarations we may write that:

(i): Let $u \in \mathbb{R}$ be the expression that forms the input argument for the transfer function $f(\cdot)$ as:

$$\boxed{\begin{aligned} u &= \underline{W}_{xu}^T \cdot \underline{x}_t + W_{hh} \cdot h_{t-1} \\ &\underbrace{[1 \times e] \cdot [e \times 1]} \; [1 \times 1][1 \times 1] \\ &[1 \times 1] + [1 \times 1] = [1] \end{aligned}}$$

(ii): $$\frac{\partial h_t}{\partial h_{t-1}} = \frac{\partial f}{\partial u} \cdot \frac{\partial u}{\partial h_{t-1}} \quad [24]$$

(iii) $$\frac{\partial u}{\partial h_{t-1}} = W_{hh} \quad [25]$$

(*) Combining Eqs. (24) and (25) yields:

$$\boxed{\frac{\partial h_t}{\partial h_{t-1}} = f'(u) \cdot W_{hh} \quad [26]}$$

, which gives:

$$\frac{\partial h_t}{\partial h_{t-1}} = \left(1 - f^2(\underline{W}_{xu}^T \underline{x}_t + W_{hh} \cdot h_{t-1})\right) W_{hh} \quad [27]$$

(*) According to Eq. (26), we may identify that:

(i): Since, $f(u) = \tanh(u) \Rightarrow f'(u) \in [0, 1]$

(ii): W_{hh} are sampled from the standard normal distribution, (for the 1-d case, $\mu = 0$, $\sigma^2 = 1$) so that $W_{hh} < 1$

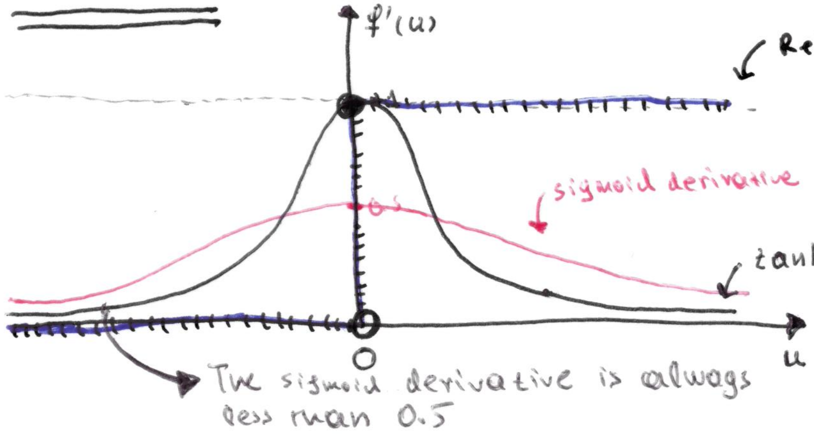
(*) In other words, we are multiplying a lot of small numbers together.

(I): Errors due to further back timesteps have increasingly smaller gradients. (We are actually multiplying long sequences of decimals which are smaller than 1)

(II): Weight-parameters become biased towards capturing shorter-term dependencies.

⊛ Addressing the Problem of Vanishing Gradients:

Solution I: Choice of Activation Functions:



⊛ ReLU Activation Function:

$$f(x) = \max(0, x) = \begin{cases} x, & x > 0; \\ 0, & x \leq 0. \end{cases} \quad [28]$$

$$f'(x) = \begin{cases} 1, & x > 0; \\ 0, & x \leq 0. \end{cases} \quad [29]$$

⊛ Thus, choosing ReLU as the activation function prevents f' from shrinking!!!

Solution II: Initialize Weight Matrices:

(i): Weight-matrices initialized to the Identity Matrix (I), at least prevents vanishing the derivative at the first steps of the calculation

(ii): Biases-vectors could be initialized to zero.

Solution III:

Rather than each node being just a simple RNN cell, we could make each cell a more complex unit with gates controlling what information is passed through.

RNNs vs {LSTMs, GRUs, etc...}

Long Short Term Memory cells are able to keep track of information throughout many timesteps.