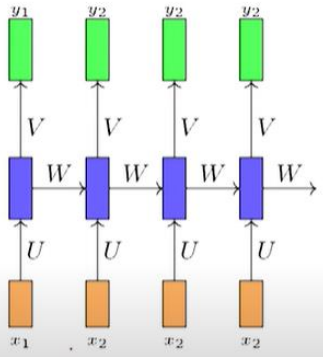


- Before proceeding let us look at the dimensions of the parameters carefully

$x_i \in \mathbb{R}^n$ (n-dimensional input)

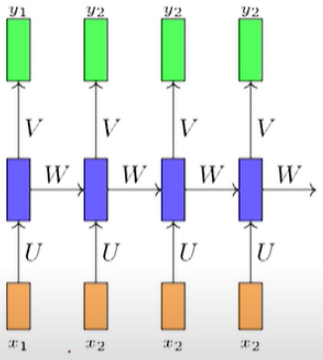
$s_i \in \mathbb{R}^d$ (d-dimensional state)

$y_i \in \mathbb{R}^k$ (say k classes)

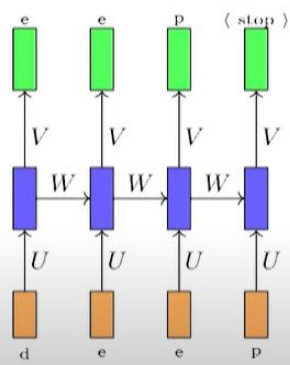
$U \in \mathbb{R}^{n \times d}$

$V \in \mathbb{R}^{d \times k}$

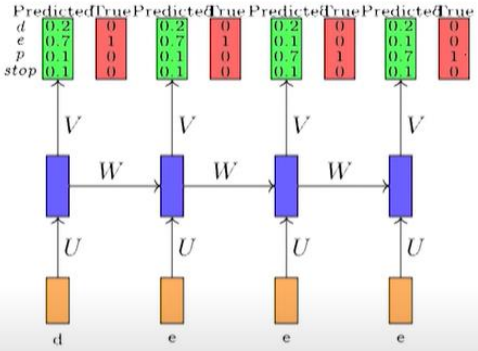
$W \in \mathbb{R}^{d \times d}$



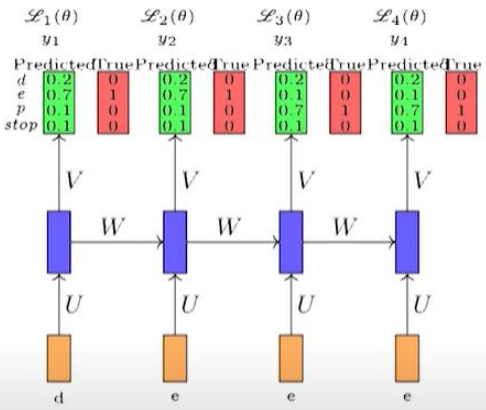
- How do we train this network ? (**Ans**: using backpropagation)



- Suppose we consider our task of auto-completion (predicting the next character)
- For simplicity we assume that there are only 4 characters in our vocabulary (d,e,p, <stop>)
- At each timestep we want to predict one of these 4 characters
- What is a suitable output function for this task ?



- Suppose we initialize U, V, W randomly and the network predicts the probabilities as shown
- And the true probabilities are as shown
- We need to answer two questions
- What is the total loss made by the model ?
- How do we backpropagate this loss and update the parameters ($\theta = \{U, V, W\}$) of the network ?



- The total loss is simply the sum of the loss over all time-steps

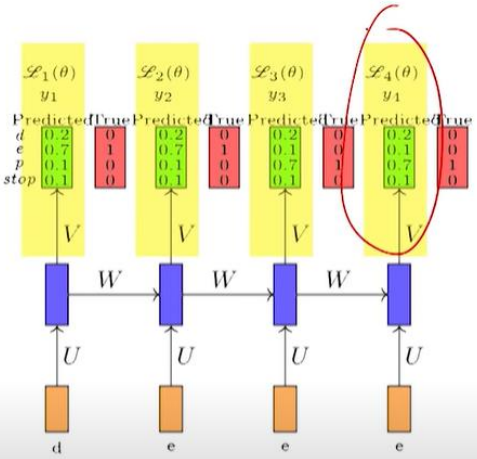
$$\mathscr{L}(\theta) = \sum_{t=1}^{T} \mathscr{L}_t(\theta)$$

$$\mathscr{L}_t(\theta) = -log(y_{tc})$$

y_{tc} = predicted probability of true character at time-step i

T = number of timesteps

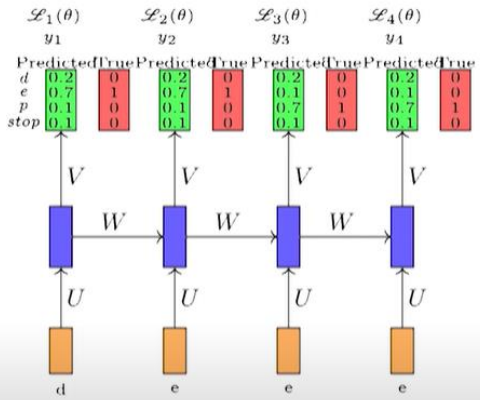
- For backpropagation we need to compute the gradients w.r.t. W, U, V



- Let us consider $\frac{\partial \mathscr{L}(\theta)}{\partial V}$ (V is a matrix so ideally we should write $\nabla_v \mathscr{L}(\theta)$)

$$\frac{\partial \mathscr{L}(\theta)}{\partial V} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}_t(\theta)}{\partial V}$$

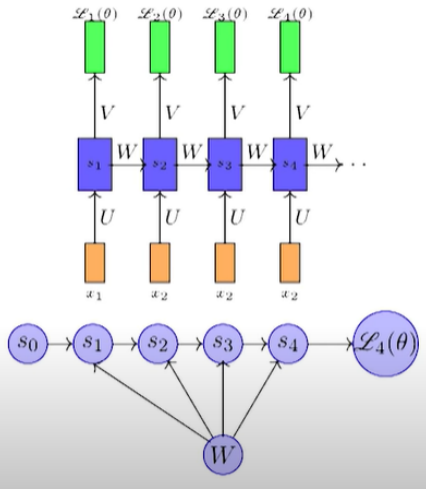
- Each term is the summation is simply the derivative of the loss w.r.t. the weights in the output layer
- We have already seen how to do this when we studied backpropagation



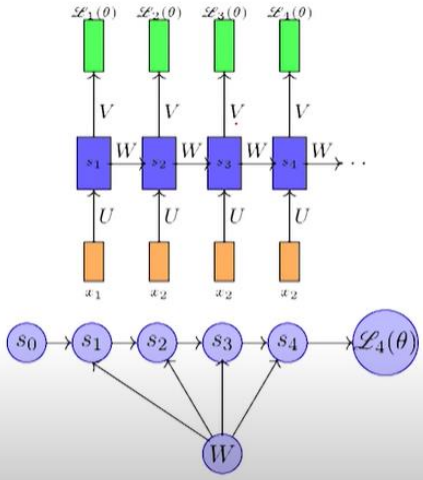
- Let us consider the derivative $\frac{\partial \mathscr{L}(\theta)}{\partial W}$

$$\frac{\partial \mathscr{L}(\theta)}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathscr{L}_t(\theta)}{\partial W}$$

- By the chain rule of derivatives we know that $\frac{\partial \mathscr{L}_t(\theta)}{\partial W}$ is obtained by summing gradients along all the paths from $\mathscr{L}_t(\theta)$ to W
- What are the paths connecting $\mathscr{L}_t(\theta)$ to W ?
- Let us see this by considering $\mathscr{L}_4(\theta)$



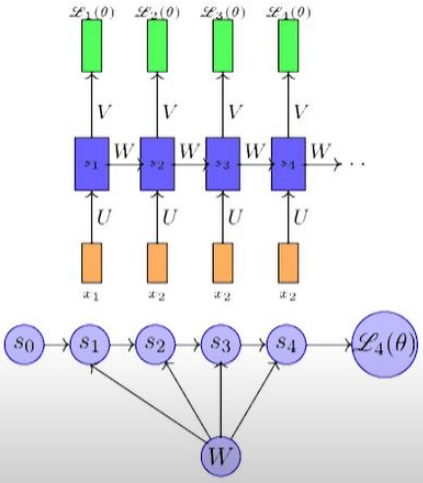
- $\mathscr{L}_4(\theta)$ depends on s_4
- s_4 in turn depends on s_3 and W
- s_3 in turn depends on s_2 and W
- s_2 in turn depends on s_1 and W
- s_1 in turn depends on s_0 and W where s_0 is a constant starting state.



- What we have here is an ordered network
- In an ordered network each state variable is computed one at a time in a specified order (first s_1, then s_2 and so on)
- Now we have

$$\frac{\partial \mathscr{L}_1(\theta)}{\partial W} = \frac{\partial \mathscr{L}_1(\theta)}{\partial s_4} \frac{\partial s_4}{\partial W}$$

- We have already seen how to compute $\frac{\partial \mathscr{L}_1(\theta)}{\partial s_4}$ when we studied backprop
- But how do we compute $\frac{\partial s_4}{\partial W}$



- Recall that

$$s_4 = \sigma(W s_3 + b)$$

- In such an ordered network, we can't compute $\frac{\partial s_4}{\partial W}$ by simply treating s_3 as a constant (because it also depends on W)
- In such networks the total derivative $\frac{\partial s_4}{\partial W}$ has two parts
- **Explicit** : $\frac{\partial^+ s_4}{\partial W}$, treating all other inputs as constant
- **Implicit** : Summing over all indirect paths from s_4 to W

$$\frac{\partial s_4}{\partial W} = \underbrace{\frac{\partial^+ s_4}{\partial W}}_{\text{explicit}} + \underbrace{\frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial W}}_{\text{implicit}}$$

$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\Big[\underbrace{\frac{\partial^+ s_3}{\partial W}}_{\text{explicit}} + \underbrace{\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial W}}_{\text{implicit}}\Big]$$

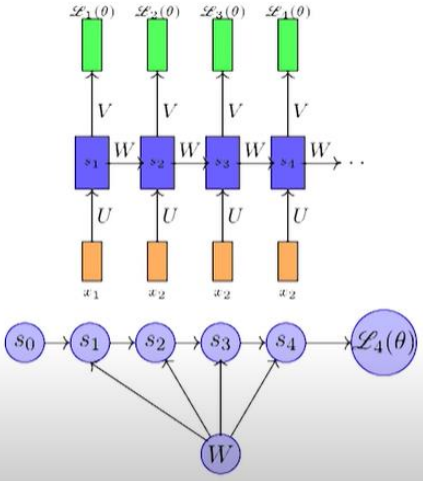
$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial^+ s_3}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial s_2}\Big[\frac{\partial^+ s_2}{\partial W} + \frac{\partial s_2}{\partial s_1}\frac{\partial s_1}{\partial W}\Big]$$

$$= \frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial^+ s_3}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial s_2}\frac{\partial^+ s_2}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial s_1}\Big[\frac{\partial^+ s_1}{\partial W}\Big]$$

For simplicity we will short-circuit some of the paths

$$\frac{\partial s_4}{\partial W} = \frac{\partial s_4}{\partial s_4}\frac{\partial^+ s_4}{\partial W} + \frac{\partial s_4}{\partial s_3}\frac{\partial^+ s_3}{\partial W} + \frac{\partial s_4}{\partial s_2}\frac{\partial^+ s_2}{\partial W} + \frac{\partial s_4}{\partial s_1}\frac{\partial^+ s_1}{\partial W} = \sum_{k=1}^{4}\frac{\partial s_4}{\partial s_k}\frac{\partial^+ s_k}{\partial W}$$

11:10 / 14:23



- Finally we have

$$\frac{\partial \mathcal{L}_4(\theta)}{\partial W} = \frac{\partial \mathcal{L}_4\theta}{\partial s_4}\frac{\partial s_4}{\partial W}$$

$$\frac{\partial s_4}{\partial W} = \sum_{k=1}^{4}\frac{\partial s_4}{\partial s_k}\frac{\partial^+ s_k}{\partial W}$$

$$\therefore \frac{\partial \mathcal{L}_t(\theta)}{\partial W} = \frac{\partial \mathcal{L}_t(\theta)}{\partial s_t}\sum_{k=1}^{t}\frac{\partial s_t}{\partial s_k}\frac{\partial^+ s_k}{\partial W}$$

- This algorithm is called backpropagation through time (BPTT) as we backpropagate over all previous time steps