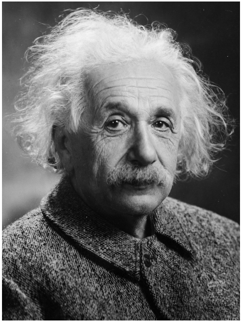
CS-568 Deep Learning

Nazar Khan

PUCIT

Recurrent Neural Networks



Everything should be made as simple as possible, but no simpler.
Albert Einstein

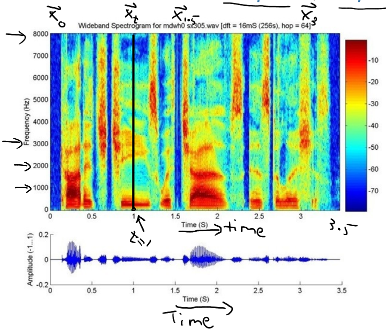
Understanding Recurrent Neural Networks requires some effort and a correct perspective. Do not expect them to be as simple as linear regression.

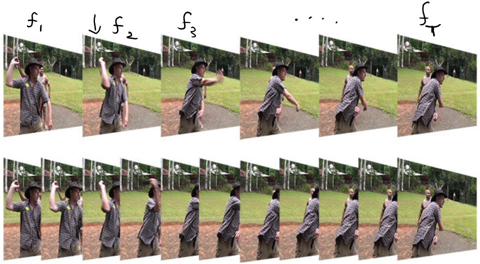
Static vs. Dynamic Inputs

I → label

- *Static* signals, such as an image, do not change over time.
  - Ordered with respect to space.
  - Output depends on current input.
- *Dynamic* signals, such as text, audio, video or stock price change over time.

  hate speech
  - Ordered with respect to time.
  - Output depends on current input as well as past (or even future) inputs.
  - Also called *temporal*, *sequential* or *time-series* data.





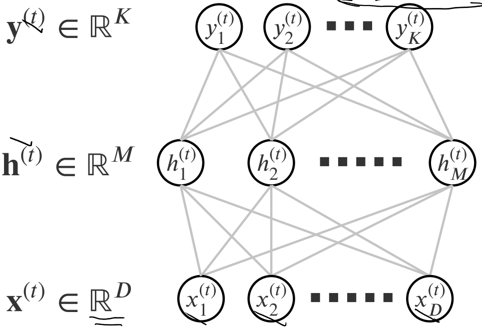
Context in Text

The Taj Mahal *_____ was commissioned by Shah Jahan in 1631, to be built in the memory of* his *___ wife Mumtaz Mahal, who died on 17 June* that *year, giving birth to* their *14th child, Gauhara Begum.* Construction *started in 1632, and the mausoleum was completed* in *___ 1643.*

Time-series Data

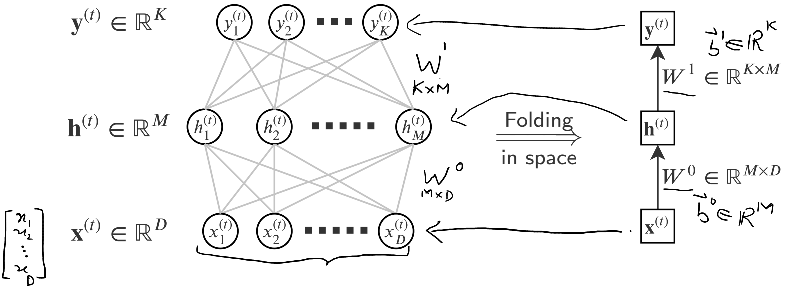
time series
1 input

- A *single* input will be a *series of vectors* $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^T$.



Input component at time t forward propagated through a network.

Representational Shortcut 1 – Space Folding



Each box represents a layer of neurons.

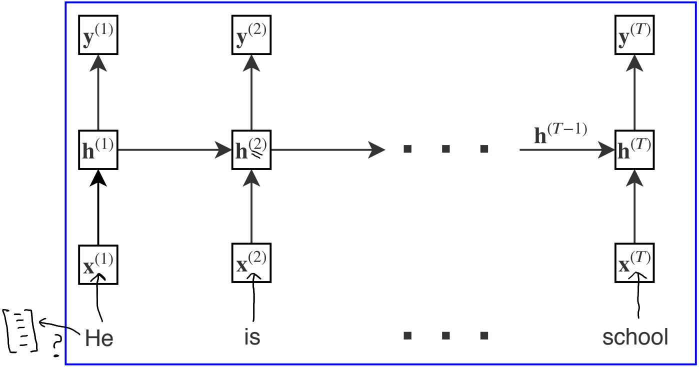
$$\underset{M\times 1}{\vec{h}} = \tanh\Big(\underset{M\times D}{W}\,\underset{D\times 1}{\vec{x}} + \underset{M\times 1}{\vec{b}}\Big)$$

Recurrent Neural Networks

$y^{(t)}$ depends on $\vec{x}^{(t)}, \vec{x}^{(t-1)}, \dots, \vec{x}^{(1)}$

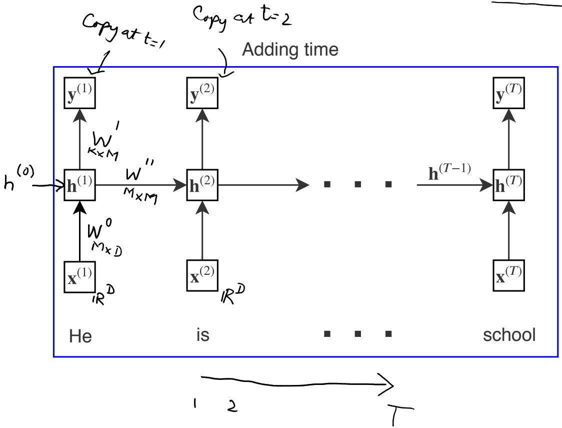
$y^{(2)}$ depends on $x^{(2)}, x^{(1)}$

Adding time

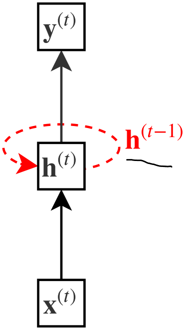


- A recurrent neural network (RNN) makes hidden state at time t directly dependent on the hidden state at time $t-1$ and therefore indirectly *on all previous times*.
- Output $\mathbf{y}_t$ depends on all that the network has already seen so far.

Representational Shortcut 2 – Time Folding



Folding in time

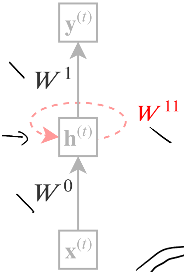


Recurrent Neural Networks

$y = f(a)$

$y = f(z)$

$\boxed{3}$ sets of weights



σ, tanh, I, softmax

pre-activation of layer 1 at time t.

$$\mathbf{y}^{(t)} = f(\overbrace{W^1\mathbf{h}^{(t)} + \mathbf{b}_1}^{\mathbf{z}^{1(t)}})$$

$$\mathbf{h}^{(t)} = \tanh(\underbrace{W^0\mathbf{x}^{(t)} + {\color{red}W^{11}\mathbf{h}^{(t-1)}} + \mathbf{b}_0}_{\mathbf{z}^{0(t)}})$$

Pre-activation of layer 0 at time t.

scalar

$$\underbrace{x_1^{(t)} \quad x_2^{(t)} \quad \cdots \quad x_D^{(t)}}_{\vec{x}^{(t)}}$$

scalar

$$\underbrace{h_1^{(t)} \quad h_2^{(t)} \quad \cdots \quad h_M^{(t)}}_{\vec{h}^{(t)}}$$

Sequence Mappings

$\vec{x}^{(1)}$

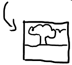
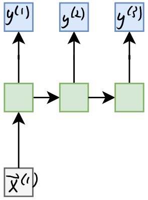


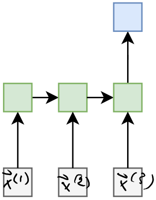
Image captioning

An eagle flying past a tree on green grass.

$y^{(1)}, y^{(2)}, \ldots\ldots y^{(9)}$

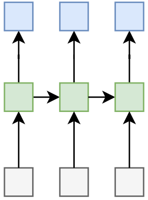


One-to-many

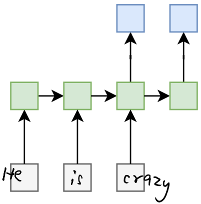


Many-to-one

Video-classification



Many-to-many



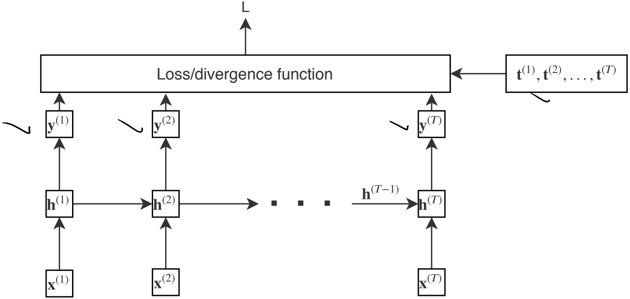
Many-to-many delayed

Language translation

Loss Functions for Sequences

L for 1 sample $\begin{cases} y^{(1)}, t^{(1)} \\ y^{(2)}, t^{(2)} \\ \vdots \\ y^{(T)}, t^{(T)} \end{cases}$ $\vec{y}^{(t)}, \vec{t}^{(t)}$

- For recurrent nets, loss is between *series* of output and target vectors. That is $\mathcal{L}(\{\mathbf{y}_1, \ldots, \mathbf{y}_T\}, \{\mathbf{t}_1, \ldots, \mathbf{t}_T\})$.



Forward propagation in an RNN unfolded in time.

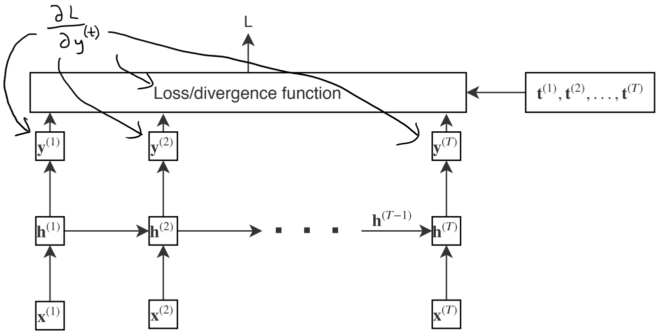
- Notice that loss $\mathcal{L}$ can be computed only after $\mathbf{y}_T$ has been computed.

Loss Functions for Sequences

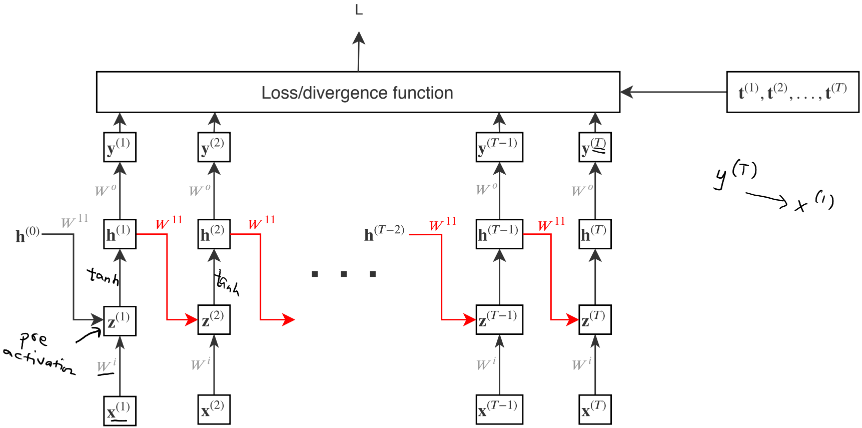
$\mathcal{L} = \mathcal{L}(y^{(1)}, t^{(1)}) + \mathcal{L}(y^{(2)}, t^{(2)}) + \cdots + \mathcal{L}(y^{(T)}, t^{(T)})$

- Loss is *not necessarily* decomposable.
- In the following, we will assume decomposable loss $\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}(\mathbf{y}_t, \mathbf{t}_t)$.
- In both cases, as long as $\frac{\partial L}{\partial \mathbf{y}_t}$ has been computed, backpropagation can proceed.

$y^{(t)}$



Backpropagation Through Time (BPTT)

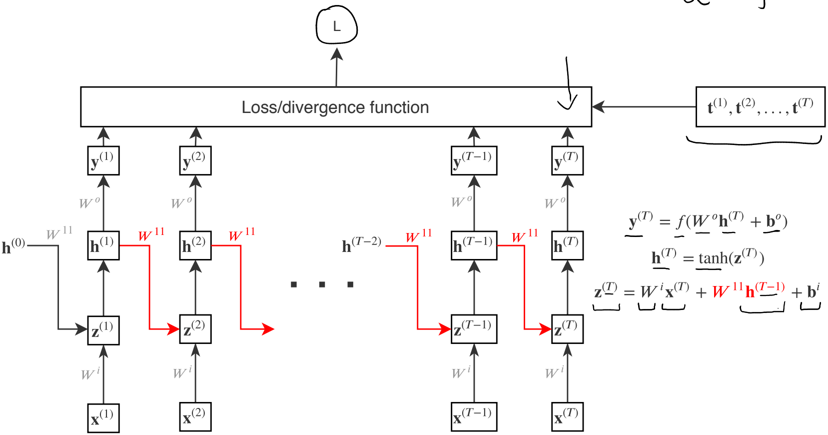


Forward propagation in an RNN unfolded in time. Recurrence between hidden states through pre-activation $\mathbf{z}^{(t)}$ is shown in red.

BPTT

$\boxed{\text{time} = T}$

$\mathcal{L} \quad y^{(T)}$



$$\mathbf{y}^{(T)} = f(W^o \mathbf{h}^{(T)} + \mathbf{b}^o)$$

$$\mathbf{h}^{(T)} = \tanh(\mathbf{z}^{(T)})$$

$$\mathbf{z}^{(T)} = W^i \mathbf{x}^{(T)} + W^{11} \mathbf{h}^{(T-1)} + \mathbf{b}^i$$

Forward propagation in an RNN unfolded in time. Recurrence between hidden states through pre-activation $\mathbf{z}^{(t)}$ is shown in red.

BPTT – Notational Clarity

$x = h^0 \to z^1 \to h^1 \to z^2 \to h^2 \to z^3 \to h^3$

Take-home quiz 4

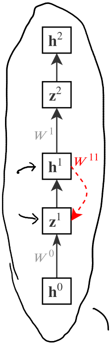
Find Derivatives Formulas

- For notational clarity, at layer l, we will denote the pre-activation by $\mathbf{z}^l$ and activation by $\mathbf{h}^l$.
- So output layer $\mathbf{y}$ will be denoted by $\mathbf{h}^L$ in an L-layer network.
- Input will be denoted by $\mathbf{h}^0$.
- So forward propagation entails

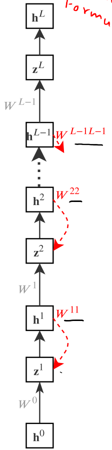
$$\mathbf{h}^0 \to \underbrace{\mathbf{z}^1 \to \mathbf{h}^1}_{\text{layer 1}} \cdots \to \underbrace{\mathbf{z}^{L-1} \to \mathbf{h}^{L-1}}_{\text{layer } L-1} \to \underbrace{\mathbf{z}^L \to \mathbf{h}^L}_{\text{layer } L}.$$

- For 2 layer network

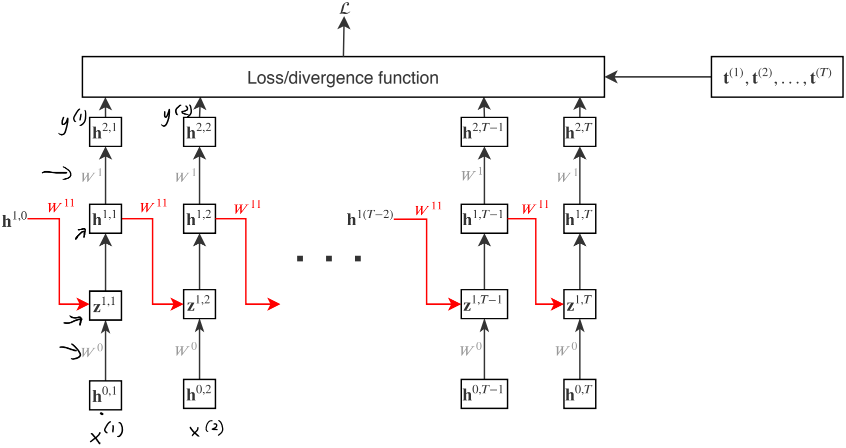
$$\mathbf{h}^{2,T} = f(W^1\mathbf{h}^{1,T} + \mathbf{b}^1)$$
$$\mathbf{h}^{1,T} = \tanh(\mathbf{z}^{1,T})$$
$$\mathbf{z}^{1,T} = W^0\mathbf{h}^{0,T} + W^{11}\mathbf{h}^{1,T-1} + \mathbf{b}^0$$



1 hidden layer.



BPTT – Notational Clarity

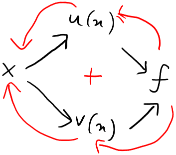


Multivariate Chain Rule

- The chain rule of differentiation states

$$\frac{df(u(x))}{dx} = \frac{df}{du}\frac{du}{dx}$$

when f depends on x through $u()$.

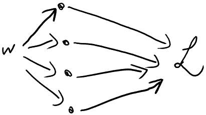


- The *multivariate* chain rule of differentiation states

$$\frac{df(u(x),v(x))}{dx} = \frac{\partial f}{\partial u}\frac{du}{dx} + \frac{\partial f}{\partial v}\frac{dv}{dx}$$

when f depends on x through $u()$ *and* through $v()$.

- Backpropagation is just an application of the multivariate chain rule.



$$\frac{\partial \mathcal{L}}{\partial w} = \sum$$

$$\frac{\partial L}{\partial \vec{y}} = \underbrace{\left[\frac{\partial L}{\partial y_1} \quad \frac{\partial L}{\partial y_2} \quad \cdots \quad \frac{\partial L}{\partial y_K} \right]}_{1\times K}$$

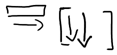
$\frac{s}{M} \leftarrow M^T$ $\quad \frac{s}{v} \leftarrow v^T$ $\quad \frac{v}{s} \leftarrow v$ $\quad \frac{d\vec{v}}{dM}$

$$\underbrace{\frac{dv_1}{dM}}_{M\times K}$$

$$\underbrace{\frac{\partial L^{\text{scalar}}}{\partial W_{\text{matrix}}}}_{M\times K} = \begin{bmatrix} \frac{\partial L}{\partial w_{11}} & \frac{\partial L}{\partial w_{21}} & \cdots & \frac{\partial L}{\partial w_{K1}} \\ & \ddots & & \\ & & & \frac{\partial L}{\partial w_{KM}} \end{bmatrix}$$

$\vec{y} = W\vec{x}$

$\underbrace{y_1} = W'\vec{x}$

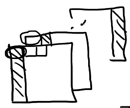


Detour

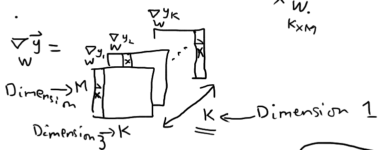
Derivative of a loss function $\underbrace{L}_{\text{scalar}}$ for vector output $\underset{K\times1}{\vec{y}} = \underset{K\times M}{W}\underset{M\times1}{\vec{x}}$ w.r.t matrix $\underset{K\times M}{W}$.

$$\underbrace{\nabla_W L}_{M\times K} = \underbrace{\nabla_{\vec{y}} L}_{1\times K}\ \underbrace{\nabla_W \vec{y}}_{K\times(M\times K)}$$

$\vec{v}^T$



$\nabla_W \vec{y} =$



$\underbrace{\qquad}_{K\times(M\times K)}$

$$M\times K \quad \begin{bmatrix} v_1x_1 & v_2x_1 & \cdots & v_Kx_1 \\ & \ddots & & \\ v_1x_M & \cdots & & v_Kx_M \end{bmatrix} = \underset{M\times1}{\vec{x}}\ \underset{1\times K}{\vec{v}^T} \longrightarrow \underline{\underline{\vec{x}}}\ \underline{\underline{\nabla_{\vec{y}} L}}$$

$\vec{y} = W\vec{x}$

$$\nabla_W y_1 = \begin{bmatrix} \frac{\partial y_1}{\partial w_{11}} & \frac{\partial y_1}{\partial w_{21}} & \cdots \\ \frac{\partial y_1}{\partial w_{12}} & & \ddots \\ \vdots & & \\ \frac{\partial y_1}{\partial w_{1M}} & & \end{bmatrix} \quad =0$$

$$= \left[\boxed{\vec{x}} \ \boxed{0} \right]$$

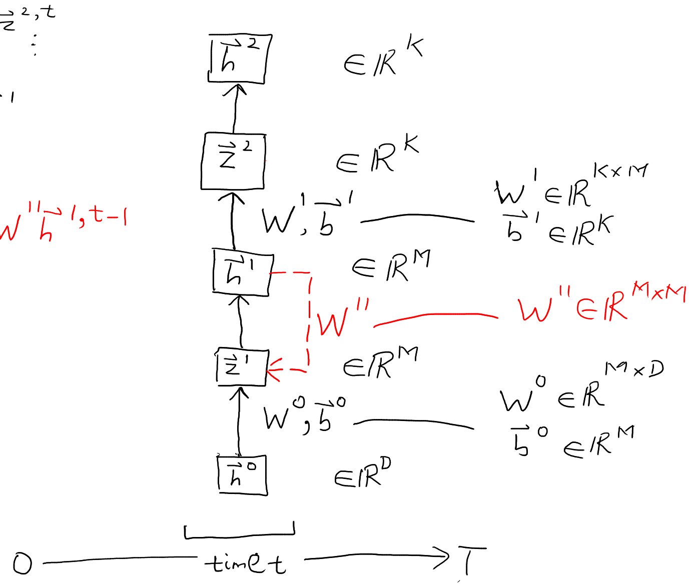
Back propagation through time (BPTT)

$\vec{h}^2 = f(\vec{z}^2)$ $\quad \vec{h}^{2,t}, \vec{z}^{2,t}, \vdots$

$\vec{z}^2 = W^1 \vec{h}^1 + \vec{b}^1$

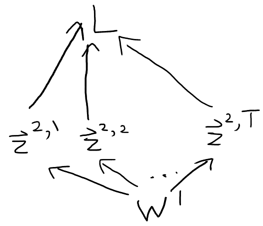
$\vec{h}^1 = \tanh(\vec{z}^1)$

$\vec{z}^1 = W^0 \vec{h}^0 + \vec{b}^0 + W'' \vec{h}^{1,t-1}$



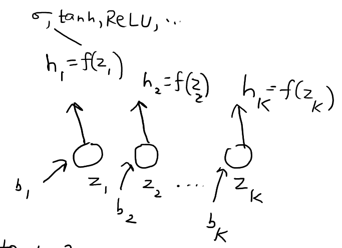
We need 5 derivatives.

1. $\nabla_{W^1} L$
2. $\nabla_{\vec{b}^1} L$
3. $\nabla_{W''} L$
4. $\nabla_{W^0} L$
5. $\nabla_{\vec{b}^0} L$



① $\underline{\underline{\nabla_{W^1} L}}$ $\quad W^1$ affects L through $\vec{z}^2$ at each time step t.

So, $$\underbrace{\nabla_{W^1} L}_{M\times K} = \sum_{t=T}^{1} \overbrace{\underbrace{\nabla_{\vec{z}^{2,t}} L}_{1\times K}}^{\delta\text{-values}} \underbrace{\nabla_{W^1} \vec{z}^{2,t}}_{K\times(M\times K)} = \sum_{t=T}^{1} \underbrace{\vec{h}^{1,t}}_{M\times 1} \underbrace{\nabla_{\vec{z}^{2,t}} L}_{1\times K}$$



where

$$\underbrace{\nabla_{\vec{z}^{2,t}} L}_{1\times K} = \underbrace{\nabla_{\vec{h}^{2,t}} L}_{1\times K} \underbrace{\nabla_{\vec{z}^{2,t}} \vec{h}^{2,t}}_{K\times K} = \nabla_{\vec{h}^{2,t}} L \underbrace{\begin{bmatrix} \frac{\partial h_1}{\partial z_1} & \frac{\partial h_1}{\partial z_2} & \cdots & \frac{\partial h_1}{\partial z_K} \\ \frac{\partial h_2}{\partial z_1} & \frac{\partial h_2}{\partial z_2} & \cdots & \frac{\partial h_2}{\partial z_K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_K}{\partial z_1} & \frac{\partial h_K}{\partial z_2} & \cdots & \frac{\partial h_K}{\partial z_K} \end{bmatrix}^{2,t}}_{K\times K}$$

Jacobian Matrix
↓
Derivatives of outputs w.r.t inputs.
$y = f(x)$
$\frac{dy}{dx}$

← - For σ, tanh, ReLU, Jacobian matrix is diagonal.
- For softmax, Jacobian matrix will be full.

② $\underline{\underline{\nabla_{\vec{b}^1} L}}$ $\quad$ Follow the reasoning for $\nabla_{W^1} L$ above but note that $\nabla_{\vec{b}^1} \vec{z}^{2,t} = I_{K\times K}$

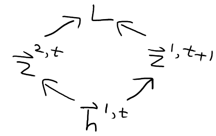
So, $$\nabla_{\vec{b}^1} L = \sum_{t=T}^{1} \nabla_{\vec{z}^{2,t}} L$$

③ $\underline{\underline{\nabla_{W''} L}}$ $\quad W''$ affects L through $\vec{z}^1$ at each time t.

$$\underbrace{\nabla_{W''} L}_{M\times M} = \sum_{t=T}^{1} \underbrace{\nabla_{\vec{z}^{1,t}} L}_{1\times M} \underbrace{\nabla_{W''} \vec{z}^{1,t}}_{M\times(M\times M)} = \sum_{t=T}^{1} \underbrace{\vec{h}^{1,t-1}}_{M\times 1} \underbrace{\nabla_{\vec{z}^{1,t}} L}_{1\times M}$$

where $$\nabla_{\vec{z}^{1,t}} L = \nabla_{\vec{h}^{1,t}} L \underbrace{\nabla_{\vec{z}^{1,t}} \vec{h}^{1,t}}_{\substack{M\times M \\ \text{Jacobian} \\ \text{matrix}}} \quad \underbrace{\begin{bmatrix} \frac{\partial h_1}{\partial z_1} & \frac{\partial h_1}{\partial z_2} & \cdots & \frac{\partial h_1}{\partial z_M} \\ \frac{\partial h_2}{\partial z_1} & \frac{\partial h_2}{\partial z_2} & \cdots & \frac{\partial h_2}{\partial z_M} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_M}{\partial z_1} & \frac{\partial h_M}{\partial z_2} & \cdots & \frac{\partial h_M}{\partial z_M} \end{bmatrix}^{1,t}}_{M\times M}$$

$\vec{h}^{1,t}$ affects L through $\vec{z}^{2,t}$ at each time t and through $\vec{z}^{1,t+1}$ at each time $t+1$.



So,

$$\underbrace{\nabla_{\vec{h}^{1,t}} L}_{1\times M} = \nabla_{\vec{z}^{2,t}} L \;\nabla_{\vec{h}^{1,t}} \vec{z}^{2,t} + \underbrace{\nabla_{\vec{z}^{1,t+1}} L \;\nabla_{\vec{h}^{1,t}} \vec{z}^{1,t+1}}$$

$$= \underbrace{\nabla_{\vec{z}^{2,t}} L}_{1\times K} \underbrace{W^{1}}_{K\times M} + \underbrace{\nabla_{\vec{z}^{1,t+1}} L}_{1\times M} \underbrace{W^{11}}_{M\times M}$$

This term will not be present for $t=T$.

④ $\nabla_{W^0} L$

W^0 affects L through $\vec{z}^{1}$ at each time t.

So,

$$\underbrace{\nabla_{W^0} L}_{D\times M} = \sum_{t=T}^{1} \underbrace{\nabla_{\vec{z}^{1,t}} L}_{1\times M} \underbrace{\nabla_{W^0} \vec{z}^{1,t}}_{M\times(D\times M)} = \sum_{t=T}^{1} \underbrace{\vec{h}^{0,t}}_{D\times 1} \underbrace{\nabla_{\vec{z}^{1,t}} L}_{1\times M}$$

③ $\nabla_{\vec{b}^0} L$

Follow the reasoning for $\nabla_{W^0} L$ above and note that $\nabla_{\vec{b}^0} \vec{z}^{1,t} = I_{M\times M}$

So,

$$\underbrace{\nabla_{\vec{b}^0} L}_{1\times M} = \sum_{t=T}^{1} \underbrace{\nabla_{\vec{z}^{1,t}} L}_{1\times M}$$

Now you have ALL the derivatives required to train an RNN with 1 hidden layer.

Training an RNN

Notice that an RNN is a very very deep network.

$\vec{h}^{2,t}$

$\vec{z}^{2,t}$

$\vdots$

$\vec{h}^{2} = f(\vec{z}^{2})$

$\vec{z}^{2} = W^{1}\vec{h}^{1} + \vec{b}^{1}$

$\vec{h}^{1} = \tanh(\vec{z}^{1})$

$\vec{z}^{1} = W^{0}\vec{h}^{0} + \vec{b}^{0} + W^{11}\vec{h}^{1,t-1}$

$$\delta_j = \underbrace{h'(a_j)}_{} \sum_{k=1}^{K} w_{kj} \underbrace{\delta_k}_{\leq 1}$$

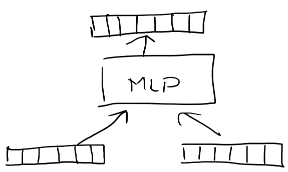
σ: $\leq \frac{1}{4}$ $\quad$ tanh: ≤ 1

Vanishing gradients

Exploding gradients $\leftarrow$ W^{11} evals. λ

Benefit of Recurrent architecture over standard MLP

N-bit addition

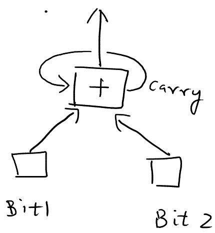


2-bits

$$\begin{array}{r} 11 \\ +\;11 \\ \hline 110 \end{array}$$

Drawbacks

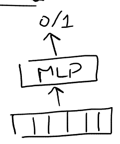
- Lots of training examples.
  Exponentially many in terms of n
- Generalization $\Rightarrow$ Training errors.
- Will work for n-bit numbers only.



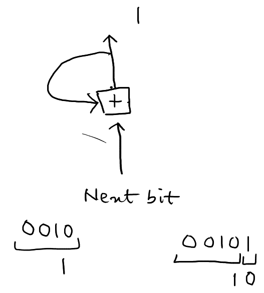
C	B_1	B_2	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

- 8 training examples
- Perfect answers
- Will work for numbers of any size.

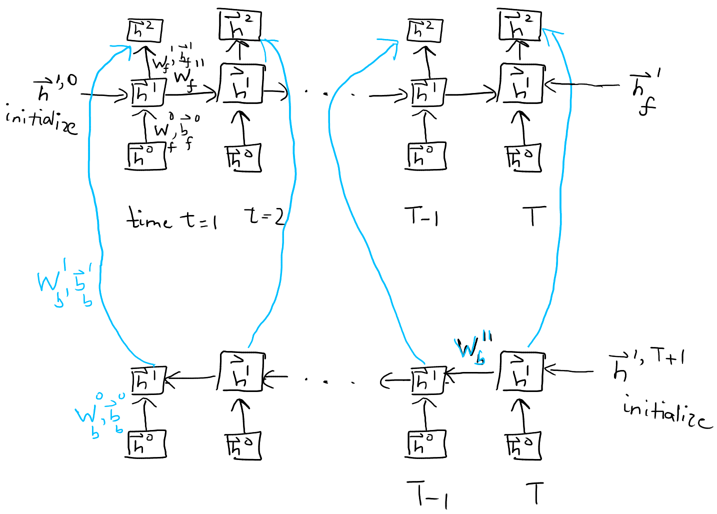
Parity (# 1s even or odd)



Same drawbacks as above.



Bidirectional RNNs



1. Fprop $(\vec h^{0,1}, \vec h^{0,2}, \dots, \vec h^{0,T}, \vec h^{1,0}) \to \vec h_f^{1,1}, \vec h_f^{1,2}, \dots, \vec h_f^{1,T}$

2. Fprop $(\underbrace{\vec h^{0,T}, \vec h^{0,T-1}, \dots, \vec h^{0,1}}_{\text{Flip input seq.}}, \vec h^{1,T+1}) \to \underbrace{\vec h_b^{1,1}, \vec h_b^{1,2}, \dots, \vec h_b^{1,T}}_{\text{after flipping in time.}}$

3. Compute $\vec h^{2,t}$ using $\vec h_f^{1,t}$ and $\vec h_b^{1,t}$.

BPTT

- After dropping L, T for clarity and using the multivariate chain rule

$$\frac{\partial \mathcal{L}}{\partial z_i} = \frac{\partial \mathcal{L}}{\partial h_1}\frac{\partial h_1}{\partial z_i} + \frac{\partial \mathcal{L}}{\partial h_2}\frac{\partial h_2}{\partial z_i} + \cdots + \frac{\partial \mathcal{L}}{\partial h_K}\frac{\partial h_K}{\partial z_i}$$

$$= \underbrace{\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial h_1} & \frac{\partial \mathcal{L}}{\partial h_2} & \cdots & \frac{\partial \mathcal{L}}{\partial h_K} \end{bmatrix}}_{\nabla_{\mathbf{h}}\mathcal{L}} \underbrace{\begin{bmatrix} \frac{\partial h_1}{\partial z_i} \\ \frac{\partial h_2}{\partial z_i} \\ \vdots \\ \frac{\partial h_K}{\partial z_i} \end{bmatrix}}_{\nabla_{z_i}\mathbf{h}} = \nabla_{\mathbf{h}}\mathcal{L}\nabla_{z_i}\mathbf{h}$$

BPTT

- This allows us to write

$$\underbrace{\nabla_{\mathbf{z}}\mathcal{L}}_{1\times K} = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial z_1} & \frac{\partial \mathcal{L}}{\partial z_2} & \dots & \frac{\partial \mathcal{L}}{\partial z_K} \end{bmatrix}$$

$$= \begin{bmatrix} \nabla_{\mathbf{h}}\mathcal{L}\nabla_{z_1}\mathbf{h} & \nabla_{\mathbf{h}}\mathcal{L}\nabla_{z_2}\mathbf{h} & \dots & \nabla_{\mathbf{h}}\mathcal{L}\nabla_{z_K}\mathbf{h} \end{bmatrix}$$

$$= \underbrace{\nabla_{\mathbf{h}}\mathcal{L}}_{1\times K} \underbrace{\begin{bmatrix} \nabla_{z_1}\mathbf{h} & \nabla_{z_2}\mathbf{h} & \dots & \nabla_{z_K}\mathbf{h} \end{bmatrix}}_{K\times K} = \nabla_{\mathbf{h}}\mathcal{L}\nabla_{\mathbf{z}}\mathbf{h}$$

  where

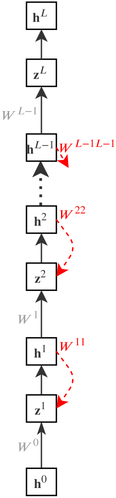
$$\nabla_{\mathbf{z}}\mathbf{h} = \begin{bmatrix} \frac{\partial h_1}{\partial z_1} & \frac{\partial h_1}{\partial z_2} & \dots & \frac{\partial h_1}{\partial z_K} \\ \frac{\partial h_2}{\partial z_1} & \frac{\partial h_2}{\partial z_2} & \dots & \frac{\partial h_2}{\partial z_K} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial h_K}{\partial z_1} & \frac{\partial h_K}{\partial z_2} & \dots & \frac{\partial h_K}{\partial z_K} \end{bmatrix}$$

  is the $K \times K$ *Jacobian matrix* of $\mathbf{h}$ with respect to $\mathbf{z}$.
- For simple, per-component activation functions (tanh, ReLU), Jacobian will be diagonal. For softmax, it will be dense.

BPTT

Derivative of loss $\mathcal{L}$ for any time t w.r.t weights of any layer l can be computed as

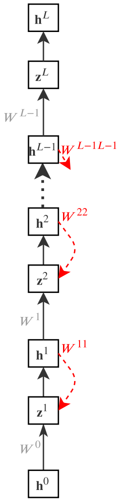
$$\underbrace{\nabla_{W^{l-1}}\mathcal{L}}_{K\times M}\Big|_t = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial W^{l-1}_{11}} & \frac{\partial \mathcal{L}}{\partial W^{l-1}_{12}} & \cdots & \frac{\partial \mathcal{L}}{\partial W^{l-1}_{1M}} \\ \frac{\partial \mathcal{L}}{\partial W^{l-1}_{21}} & \frac{\partial \mathcal{L}}{\partial W^{l-1}_{22}} & \cdots & \frac{\partial \mathcal{L}}{\partial W^{l-1}_{2M}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}}{\partial W^{l-1}_{K1}} & \frac{\partial \mathcal{L}}{\partial W^{l-1}_{K2}} & \cdots & \frac{\partial \mathcal{L}}{\partial W^{l-1}_{KM}} \end{bmatrix}$$



BPTT

Loss $\mathcal{L}$ for time t depends on W^{l-1} through $\mathbf{z}^{l,t}$

$$\underbrace{\nabla_{W^{l-1}}\mathcal{L}}_{K\times M}\Big|_t = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial z_1^{l,t}}\frac{\partial z_1^{l,t}}{\partial W_{11}^{l-1}} & \frac{\partial \mathcal{L}}{\partial z_1^{l,t}}\frac{\partial z_1^{l,t}}{\partial W_{12}^{l-1}} & \cdots & \frac{\partial \mathcal{L}}{\partial z_1^{l,t}}\frac{\partial z_1^{l,t}}{\partial W_{1M}^{l-1}} \\ \frac{\partial \mathcal{L}}{\partial z_2^{l,t}}\frac{\partial z_2^{l,t}}{\partial W_{21}^{l-1}} & \frac{\partial \mathcal{L}}{\partial z_2^{l,t}}\frac{\partial z_2^{l,t}}{\partial W_{22}^{l-1}} & \cdots & \frac{\partial \mathcal{L}}{\partial z_2^{l,t}}\frac{\partial z_2^{l,t}}{\partial W_{2M}^{l-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}}{\partial z_K^{l,t}}\frac{\partial z_K^{l,t}}{\partial W_{K1}^{l-1}} & \frac{\partial \mathcal{L}}{\partial z_K^{l,t}}\frac{\partial z_K^{l,t}}{\partial W_{K2}^{l-1}} & \cdots & \frac{\partial \mathcal{L}}{\partial z_K^{l,t}}\frac{\partial z_K^{l,t}}{\partial W_{KM}^{l-1}} \end{bmatrix}$$



BPTT

$$\underbrace{\nabla_{W^{l-1}}\mathcal{L}}_{K\times M}\Big|_t = \begin{bmatrix} \frac{\partial \mathcal{L}}{\partial z_1^{l,t}} h_1^{l-1,t} & \frac{\partial \mathcal{L}}{\partial z_1^{l,t}} h_2^{l-1,t} & \dots & \frac{\partial \mathcal{L}}{\partial z_1^{l,t}} h_M^{l-1,t} \\ \frac{\partial \mathcal{L}}{\partial z_2^{l,t}} h_1^{l-1,t} & \frac{\partial \mathcal{L}}{\partial z_2^{l,t}} h_2^{l-1,t} & \dots & \frac{\partial \mathcal{L}}{\partial z_2^{l,t}} h_M^{l-1,t} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial \mathcal{L}}{\partial z_K^{l,t}} h_1^{l-1,t} & \frac{\partial \mathcal{L}}{\partial z_K^{l,t}} h_2^{l-1,t} & \dots & \frac{\partial \mathcal{L}}{\partial z_K^{l,t}} h_M^{l-1,t} \end{bmatrix}$$

$$= \underbrace{\begin{bmatrix} \frac{\partial \mathcal{L}}{\partial z_1^{l,t}} \\ \frac{\partial \mathcal{L}}{\partial z_2^{l,t}} \\ \vdots \\ \frac{\partial \mathcal{L}}{\partial z_K^{l,t}} \end{bmatrix}}_{\nabla^{Tr}_{\mathbf{z}^{l,t}}\mathcal{L}} \underbrace{\begin{bmatrix} h_1^{l-1,t} & h_2^{l-1,t} & \dots & h_M^{l-1,t} \end{bmatrix}}_{\nabla^{Tr}_{W^{l-1}}\mathbf{z}^{l,t}} = \left(\nabla^{Tr}_{\mathbf{z}^{l,t}}\mathcal{L}\right)\left(\nabla^{Tr}_{W^{l-1}}\mathbf{z}^{l,t}\right)$$

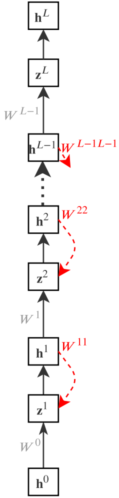
BPTT

- Start from final output $\mathbf{h}^{L,T} = f\left(\mathbf{z}^{L,T}\right)$. Assuming $\nabla_{\mathbf{h}^{L,T}}\mathcal{L}$ has been computed,

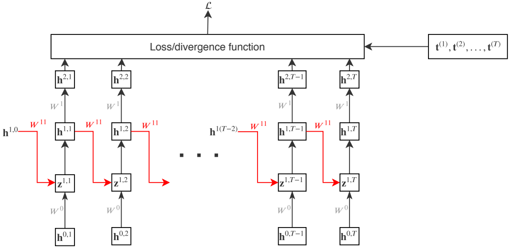
$$\underbrace{\nabla_{\mathbf{z}^{L,T}}\mathcal{L}}_{1\times K} = \underbrace{\nabla_{\mathbf{h}^{L,T}}\mathcal{L}}_{1\times K}\underbrace{\nabla_{\mathbf{z}^{L,T}}\mathbf{h}^{L,T}}_{K\times K}$$

- Derivative of loss w.r.t weights of final layer *at time T* can now be computed as

$$\underbrace{\nabla_{W^{L-1}}\mathcal{L}}_{K\times M_{L-1}}\Big|_T = \left(\nabla^{Tr}_{\mathbf{z}^{L,T}}\mathcal{L}\right)\left(\mathbf{h}^{L-1,T}\right)^{Tr}$$



BPTT



- Viewed as a function of W^{L-1}, loss can be written over time as

$$\mathcal{L}\left(\mathbf{z}^{L,1}(W^{L-1}),\ \mathbf{z}^{L,2}(W^{L-1}),\ \ldots\ ,\ \mathbf{z}^{L,T}(W^{L-1})\right)$$

- Therefore, using multivariate chain rule

$$\nabla_{W^{L-1}}\mathcal{L} = \left(\left(\nabla^{Tr}_{\mathbf{z}^{L,T}}\mathcal{L}\right)\left(\mathbf{h}^{L-1,T}\right)^{Tr} + \left(\nabla^{Tr}_{\mathbf{z}^{L,T-1}}\mathcal{L}\right)\left(\mathbf{h}^{L-1,T-1}\right)^{Tr} + \ldots + \left(\nabla^{Tr}_{\mathbf{z}^{L,1}}\mathcal{L}\right)\left(\mathbf{h}^{L-1,1}\right)^{Tr}\right)$$

$$= \sum_{1}^{t=T}\left(\nabla^{Tr}_{\mathbf{z}^{L,t}}\mathcal{L}\right)\left(\mathbf{h}^{L-1,t}\right)^{Tr}$$

BPTT

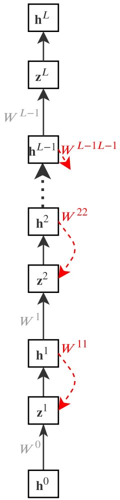
- Similarly, for each layer l from 1 to L

$$\nabla_{W^{l-1}}\mathcal{L} = \sum_{1}^{t=T} \left(\nabla^{Tr}_{\mathbf{z}^{l,t}}\mathcal{L}\right)\left(\mathbf{h}^{l-1,t}\right)^{Tr} \tag{1}$$

BPTT

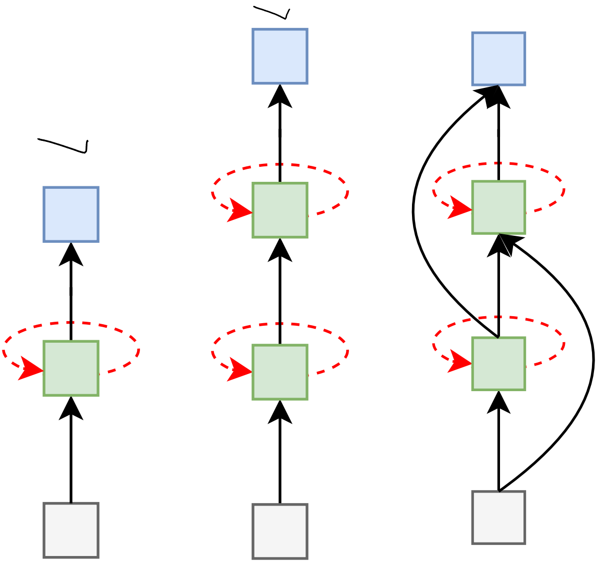
$$\underbrace{\nabla_{\mathbf{h}^{L-1}}\mathcal{L}\Big|_T}_{1\times M_{L-1}} = \underbrace{\nabla_{\mathbf{z}^L}\mathcal{L}\Big|_T}_{1\times M_L} \underbrace{W^{L-1}}_{M_L\times M_{L-1}}$$

$$\underbrace{\nabla_{\mathbf{h}^{L-1}}\mathcal{L}\Big|_t}_{1\times M_{L-1}} = \underbrace{\nabla_{\mathbf{z}^L}\mathcal{L}\Big|_t}_{1\times M_L} \underbrace{W^{L-1}}_{M_L\times M_{L-1}} + \underbrace{\nabla_{\mathbf{z}^{L-1}}\mathcal{L}\Big|_{t+1}}_{1\times M_{L-1}} \underbrace{W^{L-1,L-1}}_{M_{L-1}\times M_{L-1}}$$



RNN Variations

THQ 4



1 hidden state 2 hidden states Skip connections