# Recurrent Neural Networks

## CS60010: Deep Learning

Abir Das

IIT Kharagpur

Mar 11, 2020

# Agenda

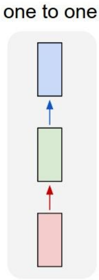
- Get introduced to different recurrent neural architecture *e.g.*, RNNs, LSTMs, GRUs etc.
- Get introduced to tasks involving sequential inputs and/or sequential outputs.

# Resources

- Deep Learning by I. Goodfellow and Y. Bengio and A. Courville. [Link] [Chapter 10]
- CS231n by Stanford University [Link]
- Understanding LSTM Networks by Chris Olah [Link]

# Why do we Need another NN Model?

- So far, we focused mainly on prediction problems with fixedsize inputs and outputs.
- In image classification, input is fixed size image and and output is its class, in video classification, the input is fixed size video and output is its class, in bounding-box regression the input is fixed size region proposal (resized/RoI pooled) and output is bounding box coordinates.



# Why do we Need another NN Model?

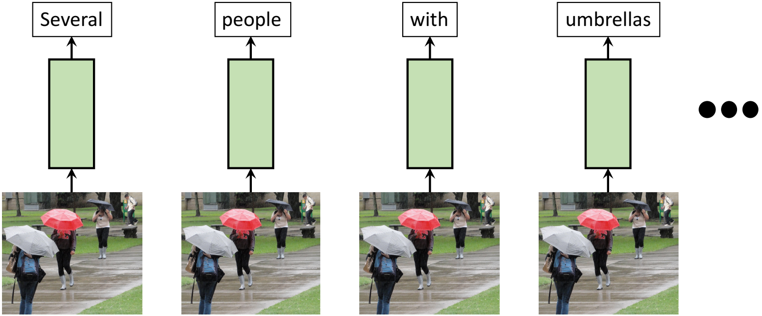
§ Suppose, we want our model to write down the caption of this image.

Figure: Several people with umbrellas walk down a side walk on a rainy day.

Image source: COCO Dataset, ICCV 2015

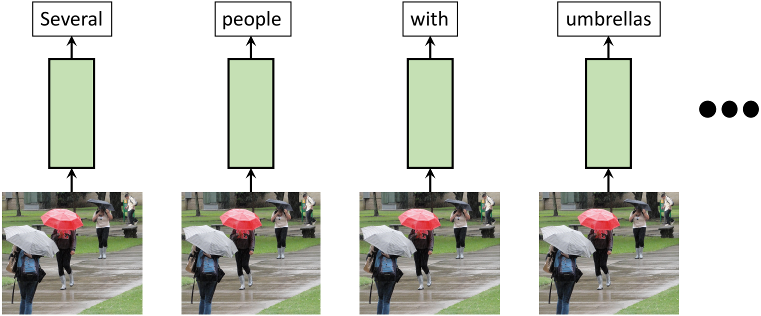
# Why do we Need another NN Model?

§ Will this work?



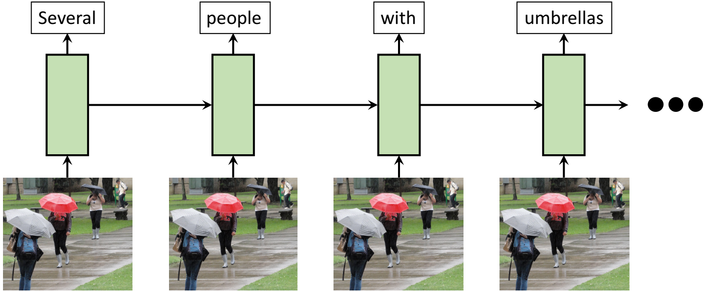
# Why do we Need another NN Model?

§ Will this work?

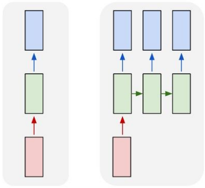



§ When the model generates 'people', we need a way to tell the model that 'several' has already been generated and similarly for the other words.

# Why do we Need another NN Model?




§




e.g. **Image Captioning**

§ image -> sequence of words

# Recurrent Neural Networks: Process Sequences

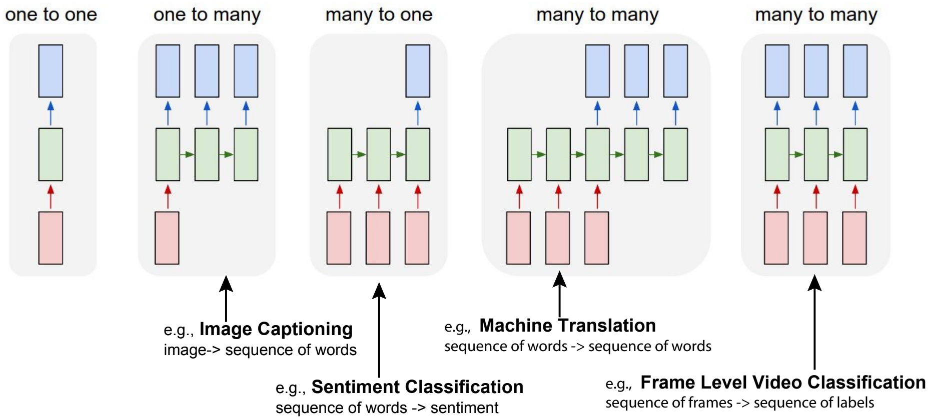
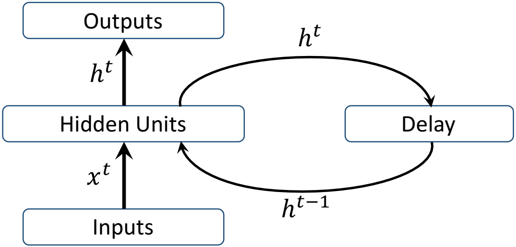


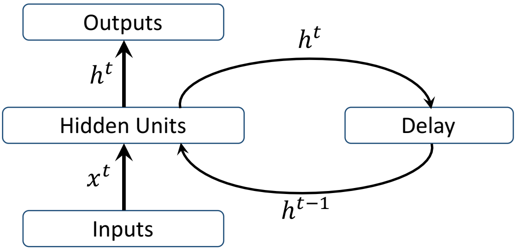

Image source: CS231n from Stanford

# Recurrent Neural Network

- The fundamental feature of a Recurrent Neural Network (RNN) is that the network contains at least one feedback connection so that activation can flow in a loop.
- The feedback connection allows information to persist. Remember the generation of `people` would require the generation of `several` to be remembered.
- The simplest form of RNN has the previous set of hidden unit activations feeding back into the network along with the inputs.

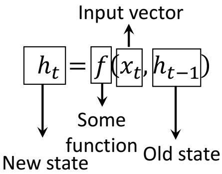


# Recurrent Neural Network



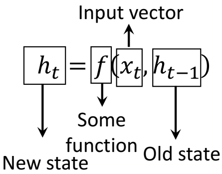

- Note that the concept of '`time`' or sequential processing comes into picture.
- The activations are updated one time-step at a time.
- The task of the delay unit is to simply delay the hidden layer activation until the next time-step.

# Recurrent Neural Network

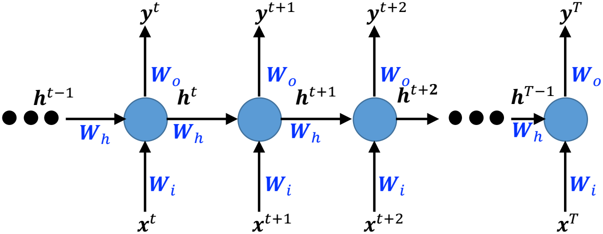



- $f$, in particular, can be a layer of a neural network.

# Recurrent Neural Network

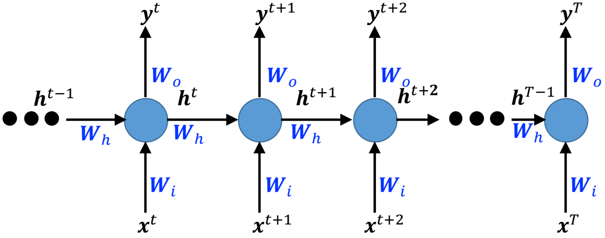



- $f$, in particular, can be a layer of a neural network.
- Lets unroll the recurrent connection.




- Note that all weight matrices are same across timesteps. So, the weights are shared for all the timesteps.

# Recurrent Neural Network: Forward Pass



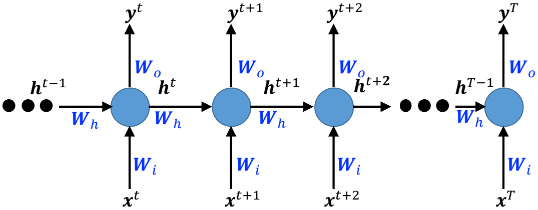

$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t \tag{1}$$

$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t) \tag{2}$$

$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t \tag{3}$$

§ Note that we can have biases too. For simplicity these are omitted.

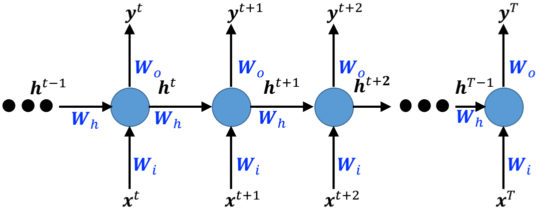
# Recurrent Neural Network: BPTT




$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t$$

- BPTT: Backpropagation through time

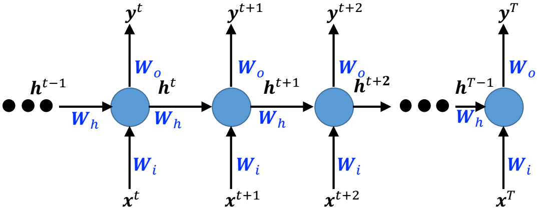
# Recurrent Neural Network: BPTT




$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

- BPTT: Backpropagation through time
- Total loss $L = \sum_{t=1}^{T} L^t$ and we are after $\frac{\partial L}{\partial \mathbf{W}_o}$, $\frac{\partial L}{\partial \mathbf{W}_h}$ and $\frac{\partial L}{\partial \mathbf{W}_i}$

# Recurrent Neural Network: BPTT



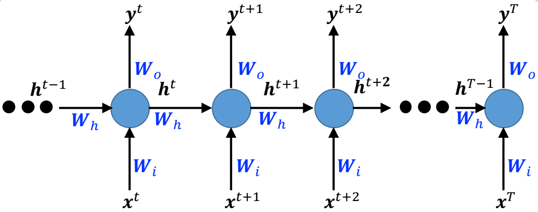

$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

- BPTT: Backpropagation through time
- Total loss $L = \sum_{t=1}^{T} L^t$ and we are after $\frac{\partial L}{\partial \mathbf{W}_o}$, $\frac{\partial L}{\partial \mathbf{W}_h}$ and $\frac{\partial L}{\partial \mathbf{W}_i}$
- Lets compute $\frac{\partial L}{\partial \mathbf{y}^t}$.

$$\frac{\partial L}{\partial \mathbf{y}^t} = \frac{\partial L}{\partial L^t}\frac{\partial L^t}{\partial \mathbf{y}^t} = 1.\boxed{\frac{\partial L^t}{\partial \mathbf{y}^t}} \tag{4}$$

- $\frac{\partial L^t}{\partial \mathbf{y}^t}$ is computable depending on the particular form of the loss function.

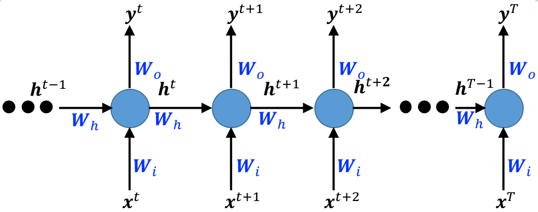
# Recurrent Neural Network: BPTT




$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

§ Lets compute $\frac{\partial L}{\partial \mathbf{h}^t}$. The subtlety here is that all $L^t$ after timestep $t$ are functions of $\mathbf{h}^t$. So, let us first consider $\frac{\partial L}{\partial \mathbf{h}^T}$, where $T$ is the last timestep.

# Recurrent Neural Network: BPTT

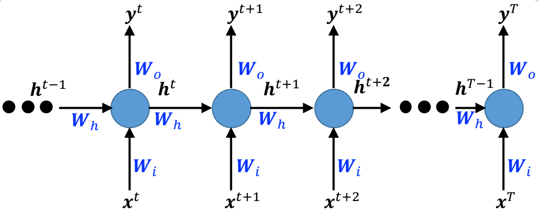



$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t$$

§ Lets compute $\frac{\partial L}{\partial \mathbf{h}^t}$. The subtlety here is that all $L^t$ after timestep $t$ are functions of $\mathbf{h}^t$. So, let us first consider $\frac{\partial L}{\partial \mathbf{h}^T}$, where $T$ is the last timestep.

$$\frac{\partial L}{\partial \mathbf{h}^T} = \frac{\partial L}{\partial \mathbf{y}^T} \frac{\partial \mathbf{y}^T}{\partial \mathbf{h}^T} = \boxed{\frac{\partial L}{\partial \mathbf{y}^T}} \mathbf{W}_o \tag{5}$$

# Recurrent Neural Network: BPTT



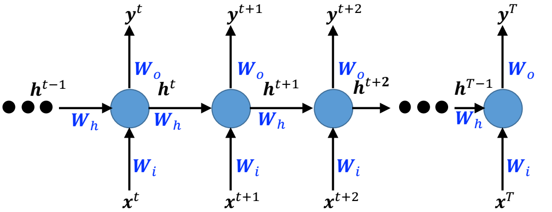

$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t$$

- Lets compute $\frac{\partial L}{\partial \mathbf{h}^t}$. The subtlety here is that all $L^t$ after timestep $t$ are functions of $\mathbf{h}^t$. So, let us first consider $\frac{\partial L}{\partial \mathbf{h}^T}$, where $T$ is the last timestep.

$$\frac{\partial L}{\partial \mathbf{h}^T} = \frac{\partial L}{\partial \mathbf{y}^T}\frac{\partial \mathbf{y}^T}{\partial \mathbf{h}^T} = \boxed{\frac{\partial L}{\partial \mathbf{y}^T}}\mathbf{W}_o \quad (5)$$

- $\frac{\partial L}{\partial \mathbf{y}^T}$, we just computed last slide (eqn. (4)).
- For a generic $t$, we need to compute $\frac{\partial L}{\partial \mathbf{h}^t}$. $\mathbf{h}^t$ affects $\mathbf{y}^t$ and also $\mathbf{h}^{t+1}$. For this we will use something that we used while studying Backpropagation for feedforward networks.

# Recurrent Neural Network: BPTT

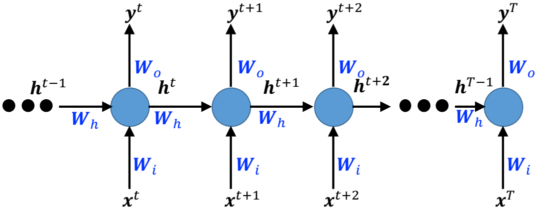



$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t$$

§ If $u = f(x, y)$, where $x = \phi(t), y = \psi(t)$, then $\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t}$

$$\frac{\partial L}{\partial \mathbf{h}^t} = \frac{\partial L}{\partial \mathbf{y}^t}\frac{\partial \mathbf{y}^t}{\partial \mathbf{h}^t} + \frac{\partial L}{\partial \mathbf{h}^{t+1}}\frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t} \quad (6)$$

# Recurrent Neural Network: BPTT



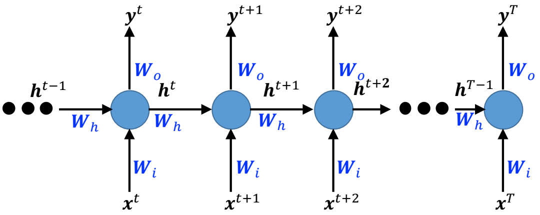

$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

§ If $u = f(x, y)$, where $x = \phi(t), y = \psi(t)$, then $\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t}$

$$\frac{\partial L}{\partial \mathbf{h}^t} = \frac{\partial L}{\partial \mathbf{y}^t}\frac{\partial \mathbf{y}^t}{\partial \mathbf{h}^t} + \frac{\partial L}{\partial \mathbf{h}^{t+1}}\frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t} \tag{6}$$

§ $\frac{\partial L}{\partial \mathbf{y}^t}$ we computed in eqn. (4)

# Recurrent Neural Network: BPTT



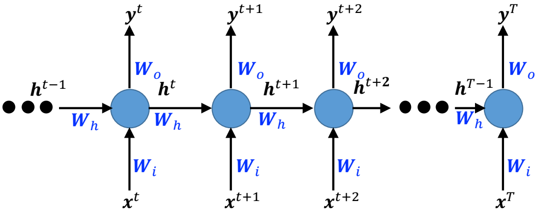

$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

§ If $u = f(x,y)$, where $x = \phi(t), y = \psi(t)$, then $\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t}$

$$\frac{\partial L}{\partial \mathbf{h}^t} = \frac{\partial L}{\partial \mathbf{y}^t}\frac{\partial \mathbf{y}^t}{\partial \mathbf{h}^t} + \frac{\partial L}{\partial \mathbf{h}^{t+1}}\frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t} \tag{6}$$

§ $\frac{\partial \mathbf{y}^t}{\partial \mathbf{h}^t} = \mathbf{W}_o$.

# Recurrent Neural Network: BPTT



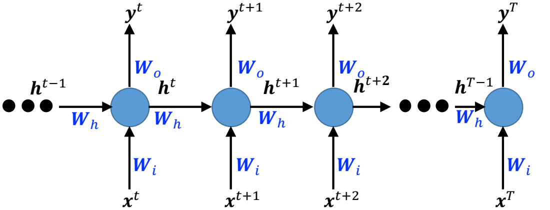

$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

§ If $u = f(x, y)$, where $x = \phi(t), y = \psi(t)$, then $\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t}$

$$\frac{\partial L}{\partial \mathbf{h}^t} = \frac{\partial L}{\partial \mathbf{y}^t}\frac{\partial \mathbf{y}^t}{\partial \mathbf{h}^t} + \frac{\partial L}{\partial \mathbf{h}^{t+1}}\frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t} \tag{6}$$

§ $\frac{\partial L}{\partial \mathbf{h}^{t+1}}$ is almost same as $\frac{\partial L}{\partial \mathbf{h}^t}$. It is just for the next timestep.

# Recurrent Neural Network: BPTT



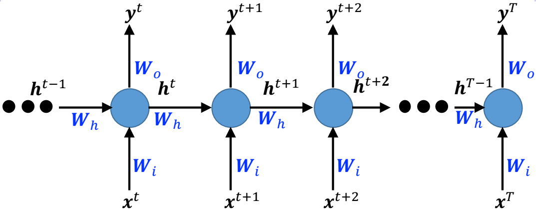

$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

- If $u = f(x, y)$, where $x = \phi(t), y = \psi(t)$, then $\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t}$

$$\frac{\partial L}{\partial \mathbf{h}^t} = \frac{\partial L}{\partial \mathbf{y}^t}\frac{\partial \mathbf{y}^t}{\partial \mathbf{h}^t} + \frac{\partial L}{\partial \mathbf{h}^{t+1}}\frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t} \quad (6)$$

- $\frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t} = \frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{a}^{t+1}}\frac{\partial \mathbf{a}^{t+1}}{\partial \mathbf{h}^t} = \mathbf{g}' \cdot \mathbf{W}_h.$
- Since, g is an elementwise operation, g′ will be a diagonal matrix.
- In particular, if $g$ is $tanh$, then
$\frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t} = diag\left(1 - (h_1^t)^2, 1 - (h_2^t)^2, \cdots, 1 - (h_m^t)^2\right)$

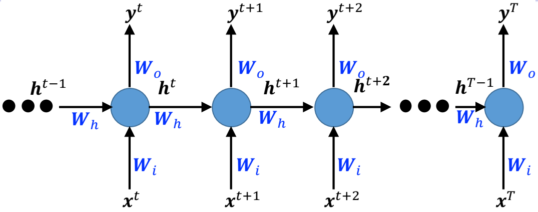
# Recurrent Neural Network: BPTT




$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

- $\frac{\partial L}{\partial \mathbf{h}^t} = \frac{\partial L}{\partial \mathbf{y}^t}\mathbf{W}_o + \frac{\partial L}{\partial \mathbf{h}^{t+1}} diag\big(1-(h_1^t)^2, 1-(h_2^t)^2, \cdots, 1-(h_m^t)^2\big)\mathbf{W}_h$
- All the other things we can compute, but to compute $\frac{\partial L}{\partial \mathbf{h}^t}$ we need $\frac{\partial L}{\partial \mathbf{h}^{t+1}}$.
- From eqn. (5), we get $\frac{\partial L}{\partial \mathbf{h}^T}$, which gives $\frac{\partial L}{\partial \mathbf{h}^{T-1}}$ and so on.

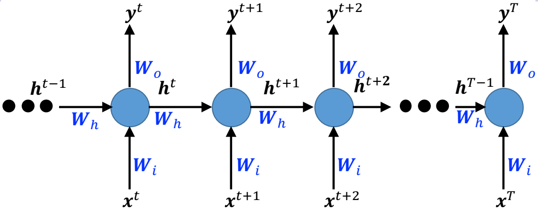
# Recurrent Neural Network: BPTT




$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

- $\frac{\partial L}{\partial \mathbf{h}^t} = \frac{\partial L}{\partial \mathbf{y}^t}\mathbf{W}_o + \frac{\partial L}{\partial \mathbf{h}^{t+1}} diag\big(1-(h_1^t)^2, 1-(h_2^t)^2, \cdots, 1-(h_m^t)^2\big)\mathbf{W}_h$
- All the other things we can compute, but to compute $\frac{\partial L}{\partial \mathbf{h}^t}$ we need $\frac{\partial L}{\partial \mathbf{h}^{t+1}}$.
- From eqn. (5), we get $\frac{\partial L}{\partial \mathbf{h}^T}$, which gives $\frac{\partial L}{\partial \mathbf{h}^{T-1}}$ and so on.
- Now, $\frac{\partial L}{\partial \mathbf{W}_o} = \sum_t \frac{\partial L^t}{\partial \mathbf{W}_o} = \sum_t \frac{\partial L^t}{\partial \mathbf{y}^t}\frac{\partial \mathbf{y}^t}{\partial \mathbf{W}_o} = \sum_t \boxed{\frac{\partial L^t}{\partial \mathbf{y}^t}}\mathbf{h}^t$
- $\frac{\partial L^t}{\partial \mathbf{y}^t}$ is computable depending on the form of the loss function.

# Recurrent Neural Network: BPTT



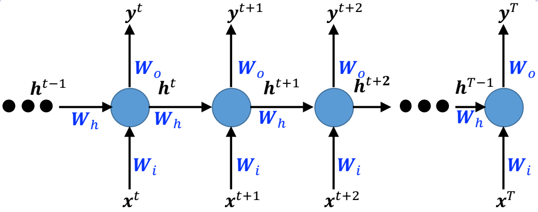

$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

- $\frac{\partial L}{\partial \mathbf{h}^t} = \frac{\partial L}{\partial \mathbf{y}^t}\mathbf{W}_o + \frac{\partial L}{\partial \mathbf{h}^{t+1}} diag\big(1-(h_1^t)^2, 1-(h_2^t)^2, \cdots, 1-(h_m^t)^2\big)\mathbf{W}_h$
- All the other things we can compute, but to compute $\frac{\partial L}{\partial \mathbf{h}^t}$ we need $\frac{\partial L}{\partial \mathbf{h}^{t+1}}$.
- From eqn. (5), we get $\frac{\partial L}{\partial \mathbf{h}^T}$, which gives $\frac{\partial L}{\partial \mathbf{h}^{T-1}}$ and so on.
- Now, $\frac{\partial L}{\partial \mathbf{W}_o} = \sum_t \frac{\partial L^t}{\partial \mathbf{W}_o} = \sum_t \frac{\partial L^t}{\partial \mathbf{y}^t}\frac{\partial \mathbf{y}^t}{\partial \mathbf{W}_o} = \sum_t \boxed{\frac{\partial L^t}{\partial \mathbf{y}^t}}\mathbf{h}^t$
- $\frac{\partial L^t}{\partial \mathbf{y}^t}$ is computable depending on the form of the loss function.
- (Do it yourself) Similarly for $\frac{\partial L}{\partial \mathbf{W}_h}$ and $\frac{\partial L}{\partial \mathbf{W}_i}$.

# Exploding or Vanishing Gradients

- In recurrent nets (also in very deep nets), the final output is the composition of a large number of non-linear transformations.
- The derivatives through these compositions will tend to be either very small or very large.
- If $h = (f \circ g)(x) = f(g(x))$, then $h'(x) = f'(g(x))g'(x)$
- If the gradients are small, the product is small.
- If the gradients are large, the product is large.

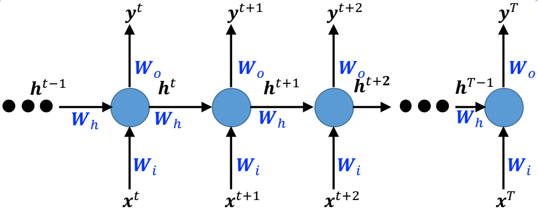
# Exploding or Vanishing Gradients




$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t$$

- Let us see what happens with one learnable weight matrix $\boldsymbol{\theta} = \mathbf{W}_h$

# Exploding or Vanishing Gradients

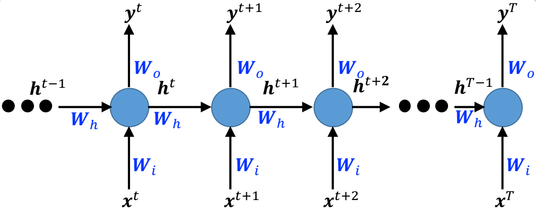



$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t$$

§ Let us see what happens with one learnable weight matrix $\boldsymbol{\theta} = \mathbf{W}_h$

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T} \frac{\partial L^t}{\partial \boldsymbol{\theta}}$$

# Exploding or Vanishing Gradients

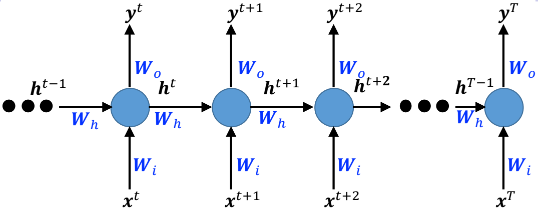



$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t$$

§ Let us see what happens with one learnable weight matrix $\boldsymbol{\theta} = \mathbf{W}_h$

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T} \frac{\partial L^t}{\partial \boldsymbol{\theta}}$$
$$\frac{\partial L^t}{\partial \boldsymbol{\theta}} = \frac{\partial L^t}{\partial \mathbf{y}^t} \frac{\partial \mathbf{y}^t}{\partial \boldsymbol{\theta}}$$

# Exploding or Vanishing Gradients

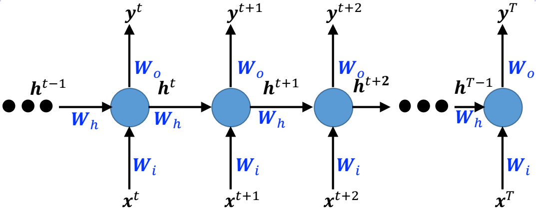



$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

§ Let us see what happens with one learnable weight matrix $\boldsymbol{\theta} = \mathbf{W}_h$

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T} \frac{\partial L^t}{\partial \boldsymbol{\theta}}$$
$$\frac{\partial L^t}{\partial \boldsymbol{\theta}} = \frac{\partial L^t}{\partial \mathbf{y}^t}\frac{\partial \mathbf{y}^t}{\partial \boldsymbol{\theta}}$$
$$\frac{\partial \mathbf{y}^t}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{y}^t}{\partial \mathbf{h}^t}\frac{\partial \mathbf{h}^t}{\partial \boldsymbol{\theta}}$$

# Exploding or Vanishing Gradients



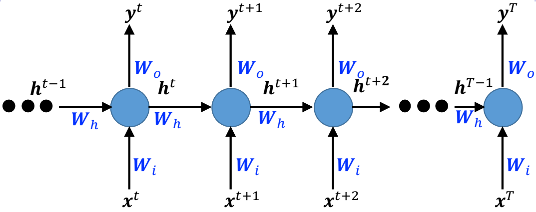

$$\mathbf{a}^t = \mathbf{W}_h\mathbf{h}^{t-1} + \mathbf{W}_i\mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o\mathbf{h}^t$$

- Let us see what happens with one learnable weight matrix $\boldsymbol{\theta} = \mathbf{W}_h$

$$\frac{\partial L}{\partial \boldsymbol{\theta}} = \sum_{t=1}^{T} \frac{\partial L^t}{\partial \boldsymbol{\theta}}$$
$$\frac{\partial L^t}{\partial \boldsymbol{\theta}} = \frac{\partial L^t}{\partial \mathbf{y}^t}\frac{\partial \mathbf{y}^t}{\partial \boldsymbol{\theta}}$$
$$\frac{\partial \mathbf{y}^t}{\partial \boldsymbol{\theta}} = \frac{\partial \mathbf{y}^t}{\partial \mathbf{h}^t}\frac{\partial \mathbf{h}^t}{\partial \boldsymbol{\theta}}$$

- But, $\mathbf{h}^t$ is a function of $\mathbf{h}^{t-1}, \mathbf{h}^{t-2}, \cdots, \mathbf{h}^2, \mathbf{h}^1$ and each of these is a function of $\boldsymbol{\theta}$.

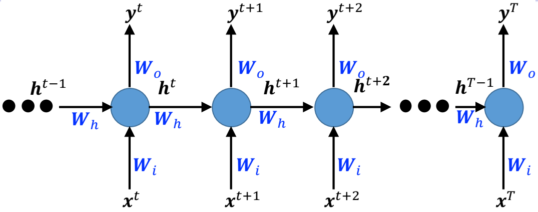
# Exploding or Vanishing Gradients




$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t$$

§ Now we will resort to our friend again - If $u = f(x, y)$, where $x = \phi(t), y = \psi(t)$, then $\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t}$

# Exploding or Vanishing Gradients

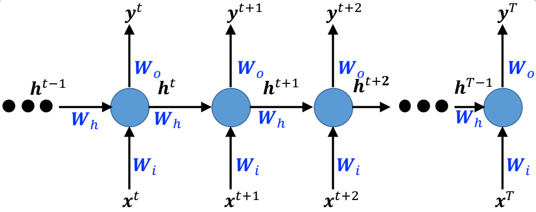



$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t$$

§ Now we will resort to our friend again - If $u = f(x, y)$, where $x = \phi(t), y = \psi(t)$, then $\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t}$

$$\frac{\partial \mathbf{h}^t}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{t-1} \frac{\partial \mathbf{h}^t}{\partial \mathbf{h}^k} \frac{\partial \mathbf{h}^k}{\partial \boldsymbol{\theta}}$$

# Exploding or Vanishing Gradients




$$\mathbf{a}^t = \mathbf{W}_h \mathbf{h}^{t-1} + \mathbf{W}_i \mathbf{x}^t$$
$$\mathbf{h}^t = \mathbf{g}(\mathbf{a}^t)$$
$$\mathbf{y}^t = \mathbf{W}_o \mathbf{h}^t$$

- Now we will resort to our friend again - If $u = f(x, y)$, where $x = \phi(t), y = \psi(t)$, then $\frac{\partial u}{\partial t} = \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t}$

$$\frac{\partial \mathbf{h}^t}{\partial \boldsymbol{\theta}} = \sum_{k=1}^{t-1} \frac{\partial \mathbf{h}^t}{\partial \mathbf{h}^k} \frac{\partial \mathbf{h}^k}{\partial \boldsymbol{\theta}}$$

- And

$$\frac{\partial \mathbf{h}^t}{\partial \mathbf{h}^k} = \frac{\partial \mathbf{h}^t}{\partial \mathbf{h}^{t-1}} \frac{\partial \mathbf{h}^{t-1}}{\partial \mathbf{h}^{t-2}} \cdots \frac{\partial \mathbf{h}^{k+1}}{\partial \mathbf{h}^k}$$
$$= diag\big[\{1-(h_1^{t-1})^2\}\{1-(h_1^{t-2})^2\}\cdots, \{1-(h_2^{t-1})^2\}\{1-(h_2^{t-2})^2\}\cdots,\big]$$

# Exploding or Vanishing Gradients

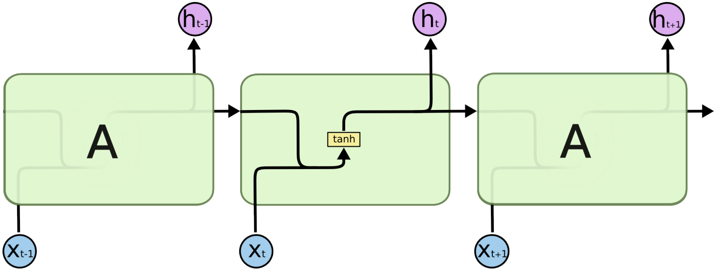
- Exploding Gradients
  - Easy to detect
  - Clip the gradient at a threshold
- Vanishing Gradients
  - More difficult to detect
  - Architectures designed to combat the problem of vanishing gradients. Example: LSTMs by Schmidhuberet *et. al.*

# Long Short Term Memory (LSTM)

- Hochreiter & Schmidhuber(1997) solved the problem of getting an RNN to remember things for a long time (*e.g.*, hundreds of time steps).
- They designed a memory cell using logistic and linear units with multiplicative interactions.
- Information is handled using three gates, namely - forget, input and output.

# Recall: Vanilla RNNs

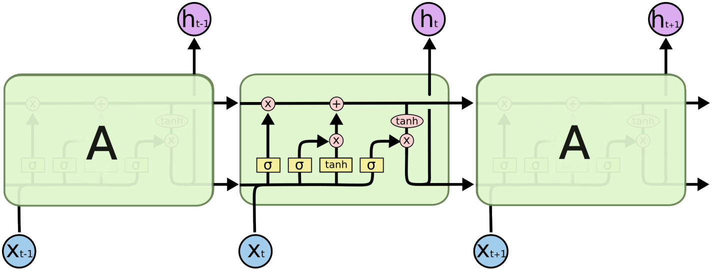
§ In a standard RNN the repeating module has a simple structure.



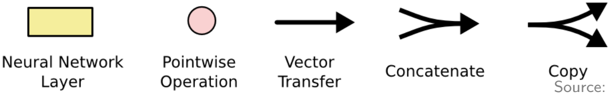

Source: Chris Olah's blog

# LSTMs

- LSTMs also have this chain like structure, but the repeating module has a different structure.
- Instead of having a single neural network layer, there are four, interacting in a very special way.



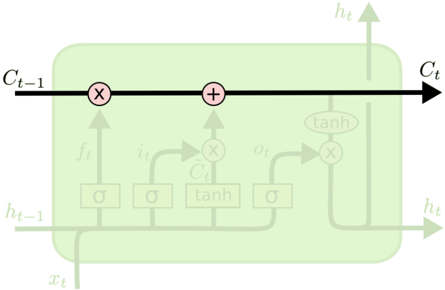

- The notations mean




Source: Chris Olah's blog

# LSTM Memory/Cell State

- The key to LSTMs is the cell state, the horizontal line running through the top of the diagram.
- The cell state is kind of like a conveyor belt. Its very easy for information to just flow along it unchanged.

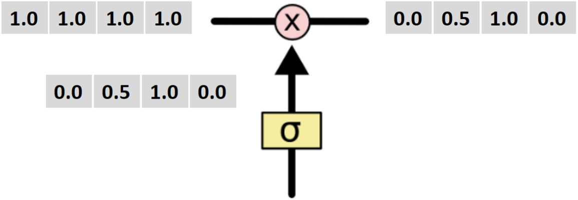



- The LSTM does have the ability to remove or add information to the cell state, carefully regulated by gates.

Source: Chris Olah's blog

# Gate

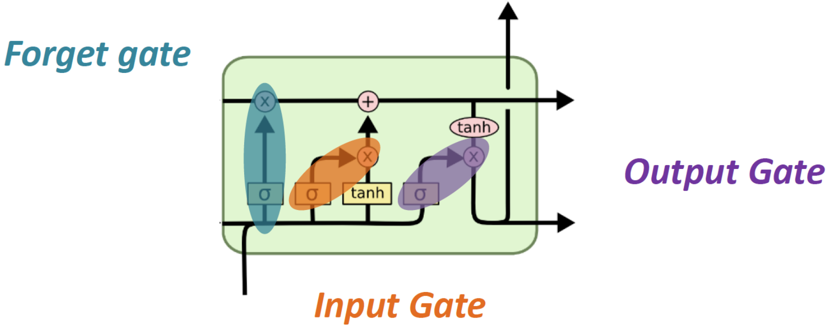
- Composed of a sigmoid neural net layer and a pointwise multiplication operation.
- Sigmoid: outputs numbers between
  - Zero: "let nothing through" and
  - One: "let everything through"




Source: Chris Olah's blog

# Gate

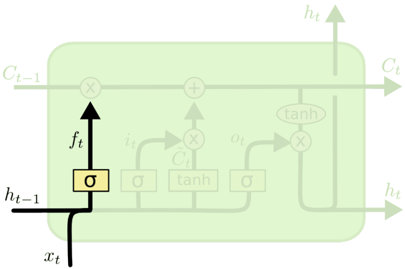
§ And LSTM has three such gates.




Source: Chris Olah's blog

# Forget Gate

- The first step is to decide what information is going to be throw away from the cell state. This decision is made by the "forget gate layer".
- It looks at $h_{t-1}$ and $x_t$, and outputs a number between $0$ and $1$ for each number in the cell state $C_{t-1}$. A $1$ represents "completely keep this" while a $0$ represents "completely get rid of this".

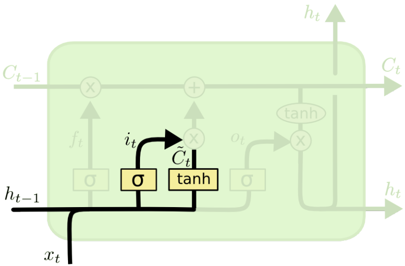



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Source: Chris Olah's blog

# Input Gate

- The next step is to decide what new information is going to be stored in the cell state. This has two parts.
- First, a sigmoid layer called the "input gate layer" decides which values to update. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state.




$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

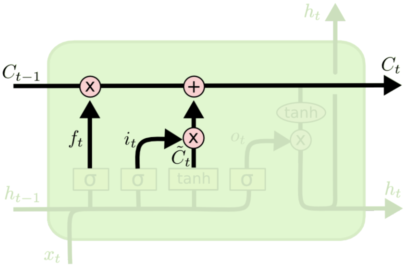
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

- Next step combines these two to create an update to the state.

Source: Chris Olah's blog

# Input Gate

- Its now time to update the old cell state, $C_{t-1}$, into the new cell state $C_t$.
- This is done by multiplying the old state by $f_t$, forgetting the things that were decided to forget earlier and adding $i_t * \tilde{C}_t$.

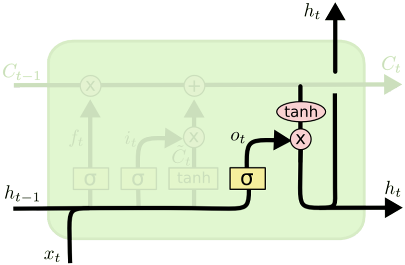



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Source: Chris Olah's blog

# Output Gate

- Finally, we need to decide what is going to be the output. This output will be based on the cell state.
- First a sigmoid layer is run which decides what parts of the cell state are going to be output.
- Then, the cell state is put through tanh (to push the values to be between 1 and 1) and this is multiplied by the output of the sigmoid gate.




$$o_t = \sigma\left(W_o\ [h_{t-1}, x_t]\ +\ b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Source: Chris Olah's blog