# Recurrent Neural Networks

## CISC 7026: Introduction to Deep Learning

University of Macau

# Admin

Makeup lecture Saturday October 26, 13:00-16:00

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Agenda

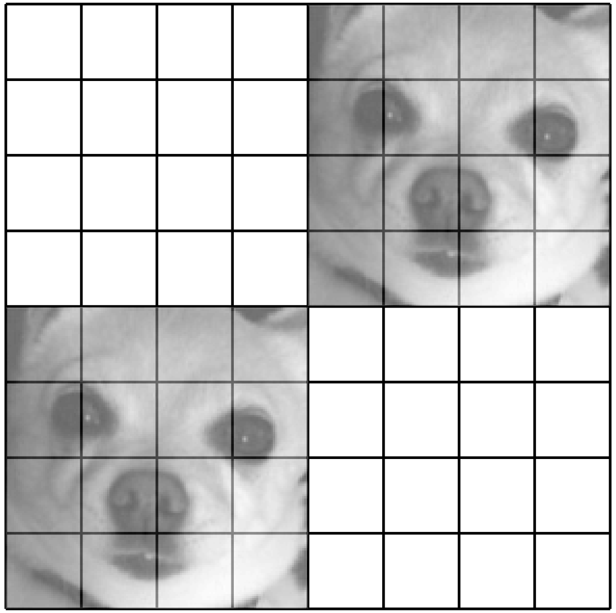
1. **Review**
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Review

In perceptrons, each pixel is an independent neuron

# Review

In perceptrons, each pixel is an independent neuron



# Review

These images are equivalent to a neural network

# Review

These images are equivalent to a neural network

It is a miracle that our neural networks could classify clothing!

# Review

A **signal** represents information as a function of time, space or some other variable

# Review

A **signal** represents information as a function of time, space or some other variable

$$x(t) = 2t + 1$$

# Review

A **signal** represents information as a function of time, space or some other variable

$$x(t) = 2t + 1$$

$$x(u, v) = \frac{u^2}{v} - 3$$

# Review

A **signal** represents information as a function of time, space or some other variable

$$x(t) = 2t + 1$$

$$x(u, v) = \frac{u^2}{v} - 3$$

$x(t), x(u, v)$ represent physical processes that we may or may not know

# Review

A **signal** represents information as a function of time, space or some other variable

$$x(t) = 2t + 1$$

$$x(u, v) = \frac{u^2}{v} - 3$$

$x(t), x(u, v)$ represent physical processes that we may or may not know

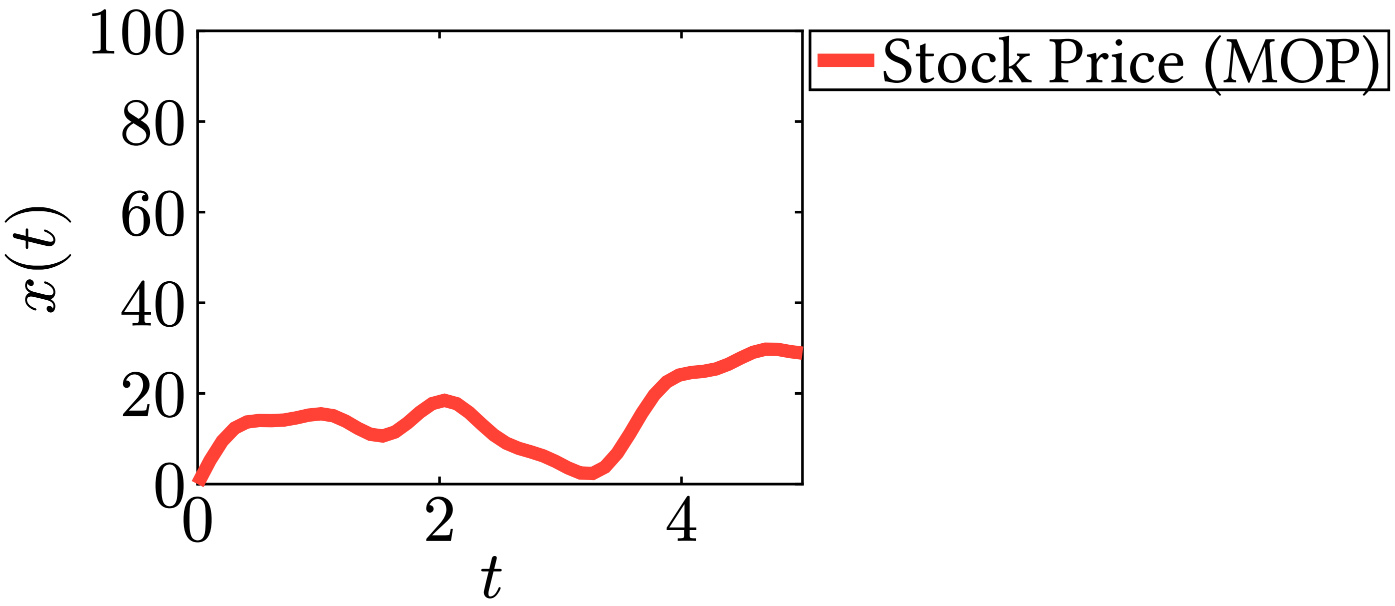
In **signal processing**, we analyze the meaning of signals

# Review

$$x(t) = \text{stock price}$$

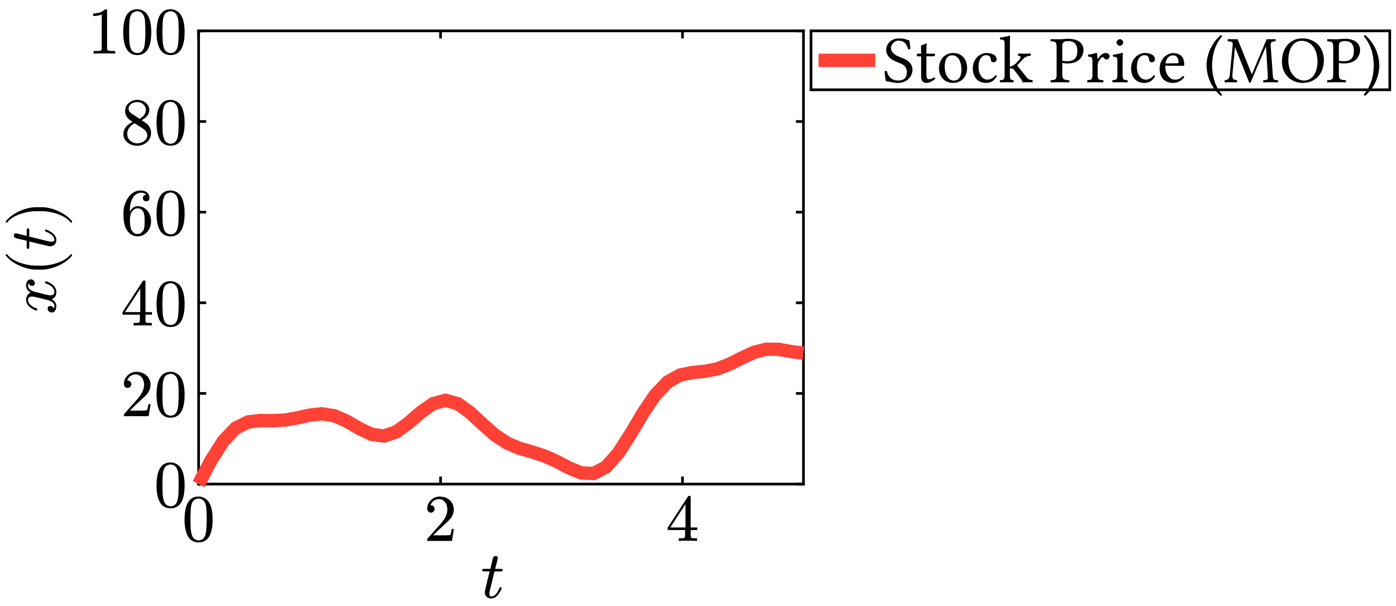
# Review

$$x(t) = \text{stock price}$$



# Review

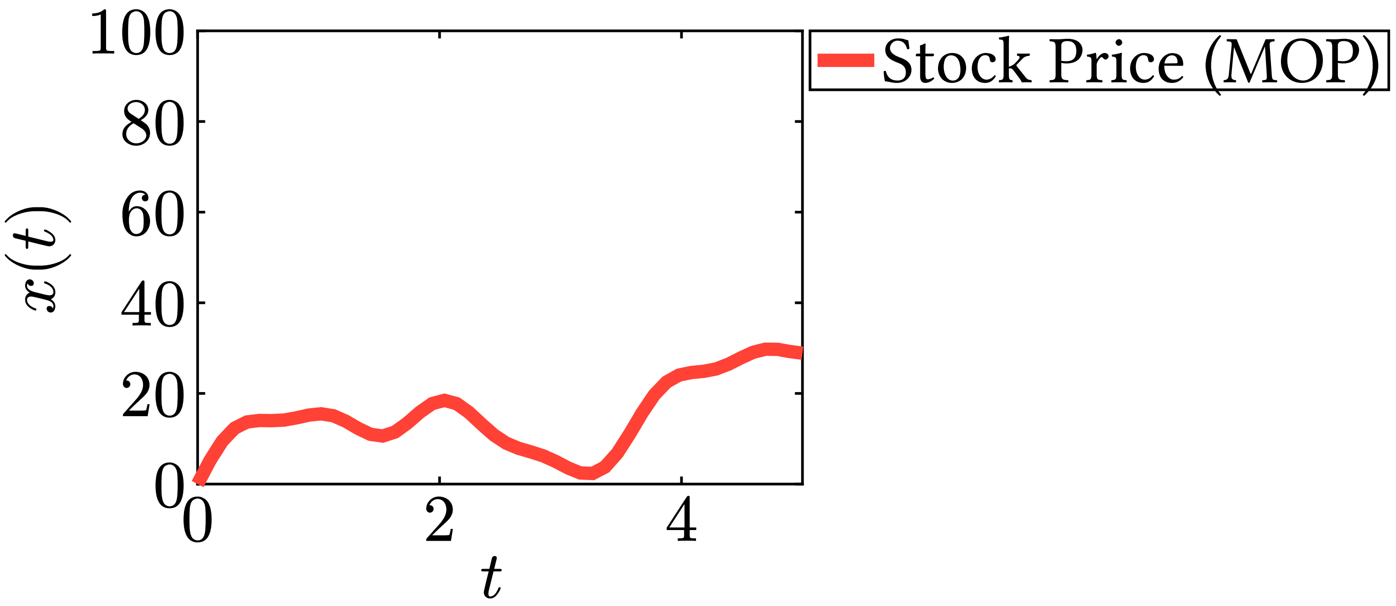
$$x(t) = \text{stock price}$$




There is an underlying structure to $x(t)$

# Review

$$x(t) = \text{stock price}$$




There is an underlying structure to $x(t)$

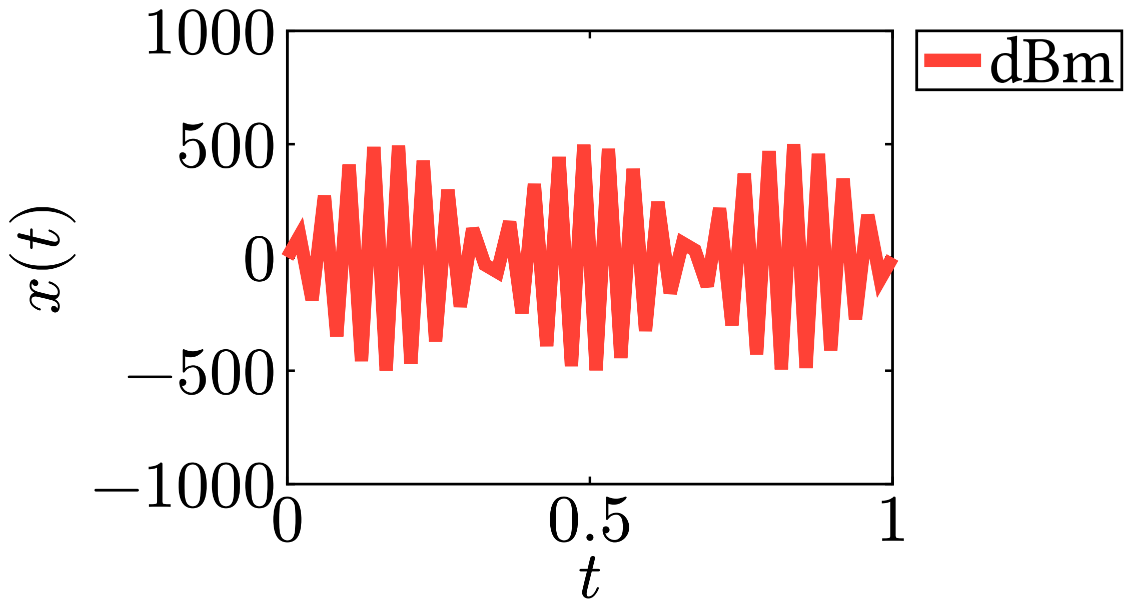
**Structure:** Tomorrow's stock price will be close to today's stock price

# Review

$$x(t) = \text{audio}$$

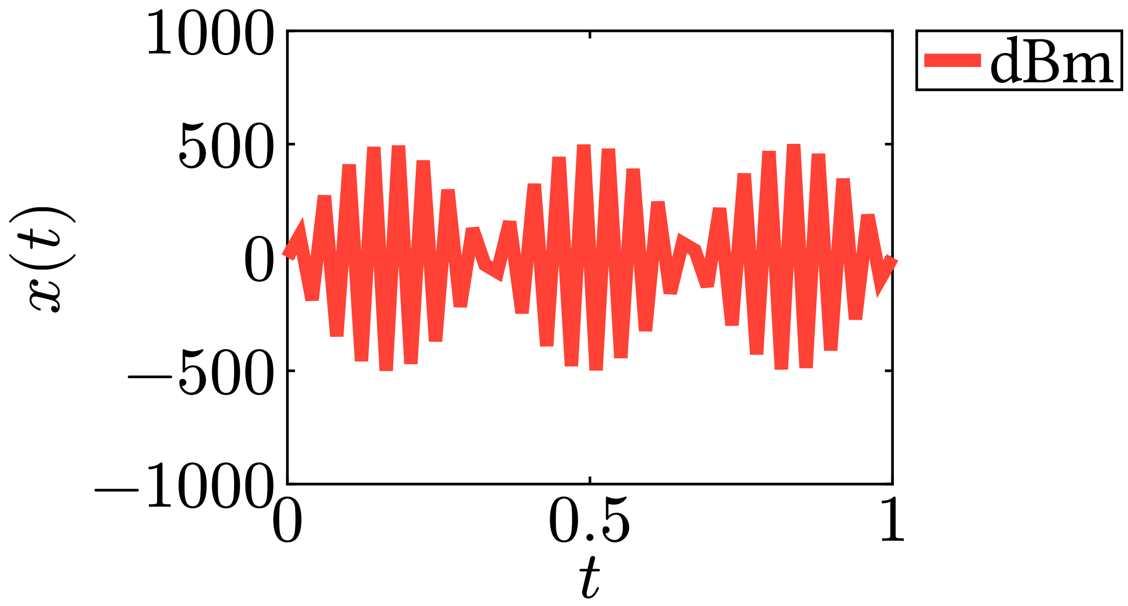
# Review

$$x(t) = \text{audio}$$



# Review

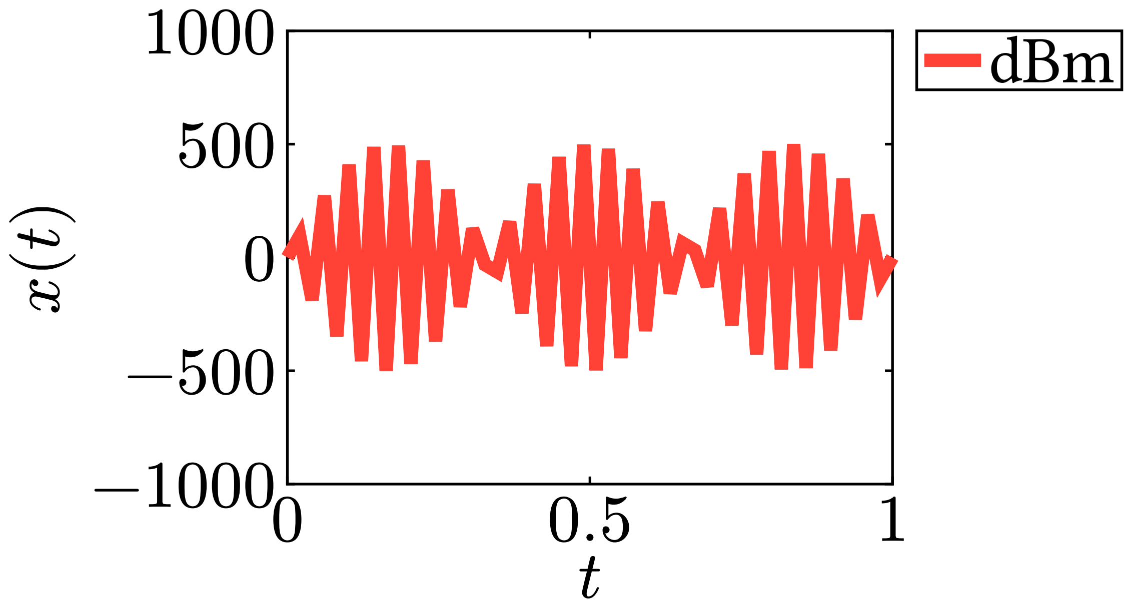
$$x(t) = \text{audio}$$




**Structure:** Nearby waves form syllables

# Review

$$x(t) = \text{audio}$$




**Structure:** Nearby waves form syllables

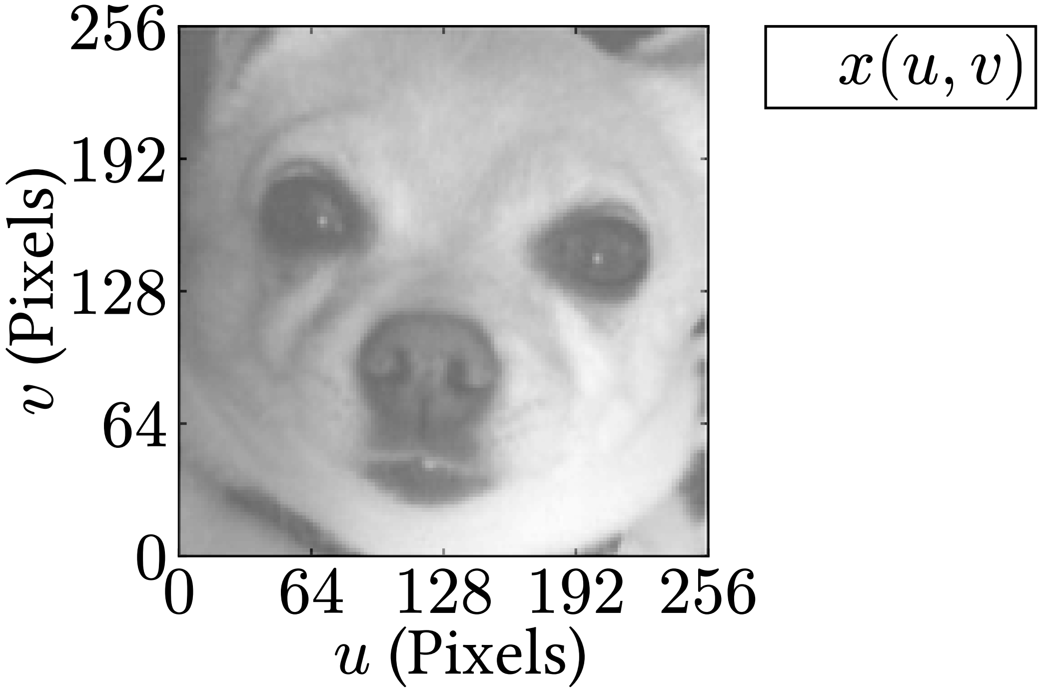
**Structure:** Nearby syllables combine to create meaning

# Review

$$x(u, v) = \text{image}$$

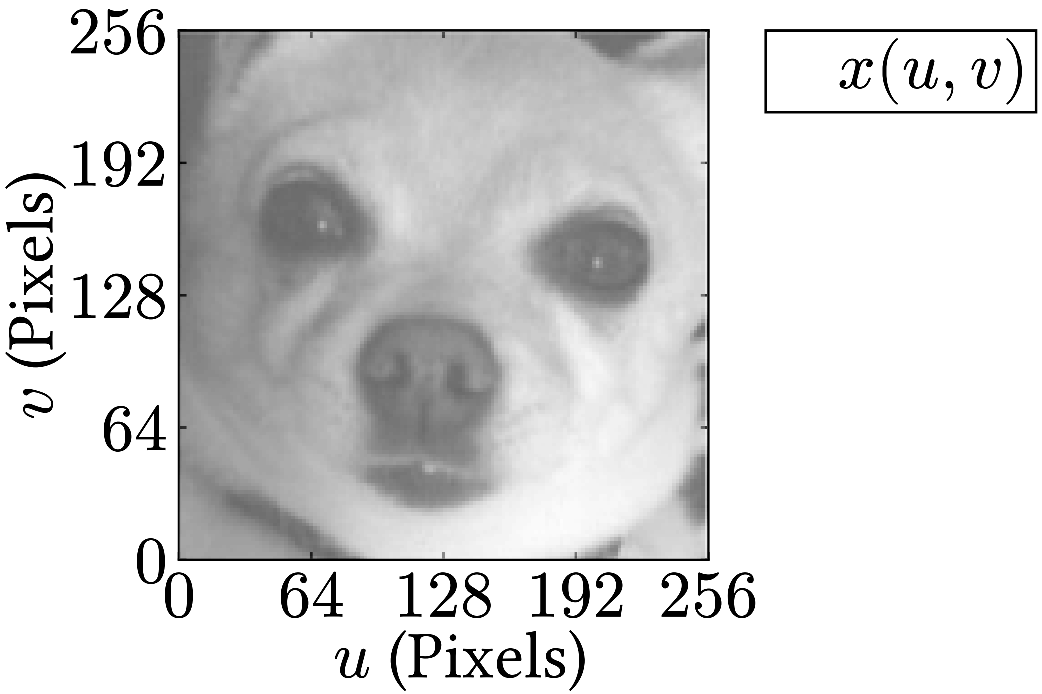
# Review

$$x(u, v) = \text{image}$$



# Review

$$x(u, v) = \text{image}$$




**Structure:** Repeated components (circles, symmetry, eyes, nostrils, etc)

# Review

**Locality:** Information concentrated over small regions of space/time

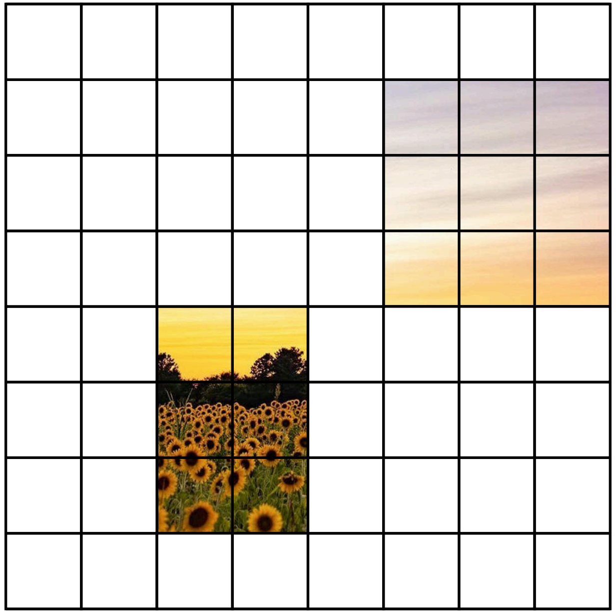
# Review

**Locality:** Information concentrated over small regions of space/time

**Translation Equivariance:** Shift in signal results in shift in output

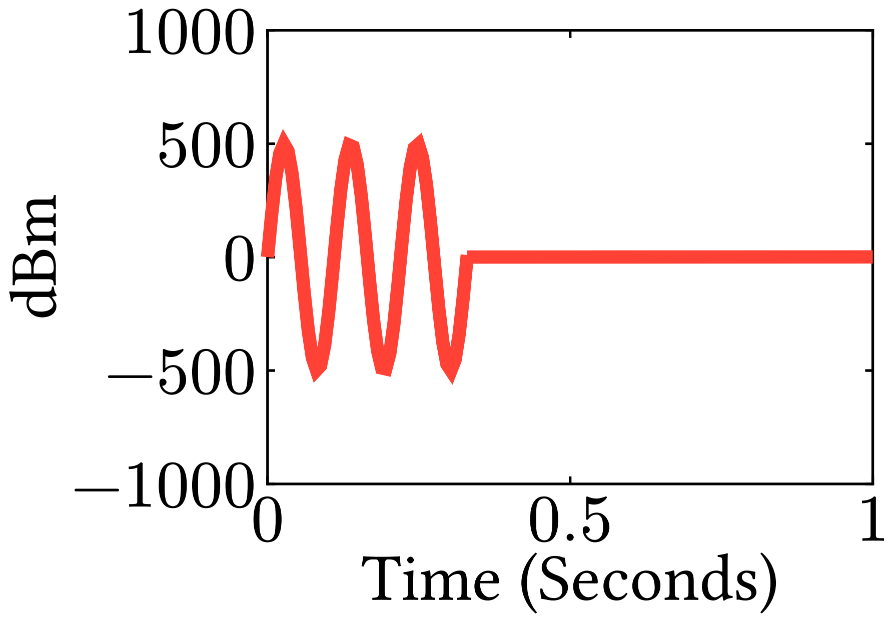
# Review

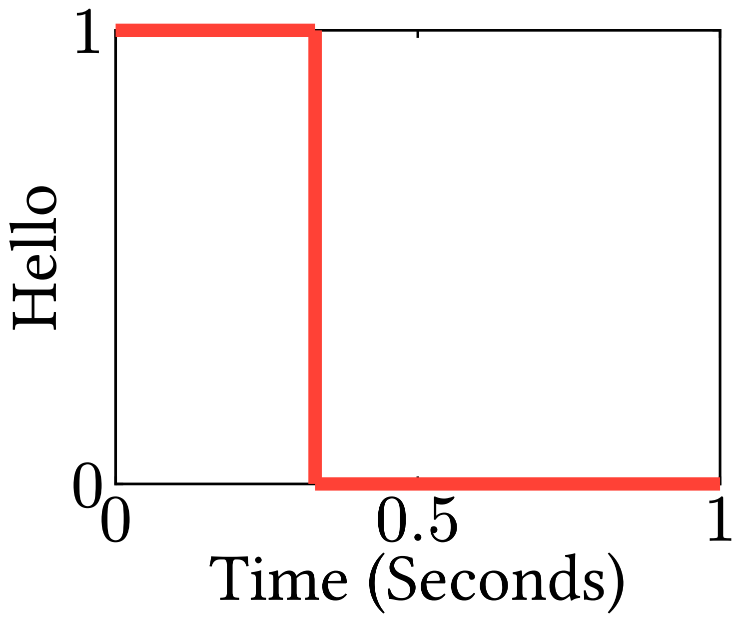
A more realistic scenario of locality and translation equivariance



# Convolution

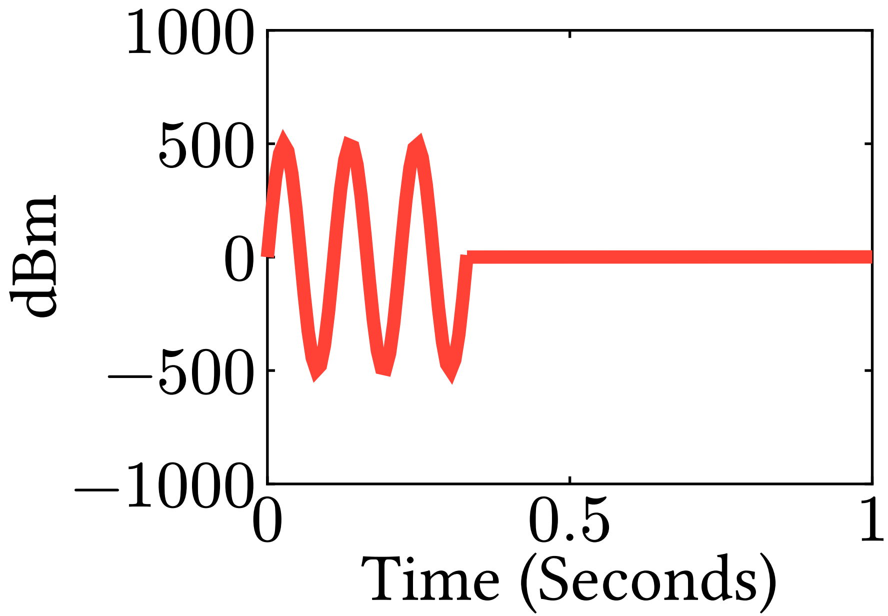
We use convolution to turn signals into useful signals

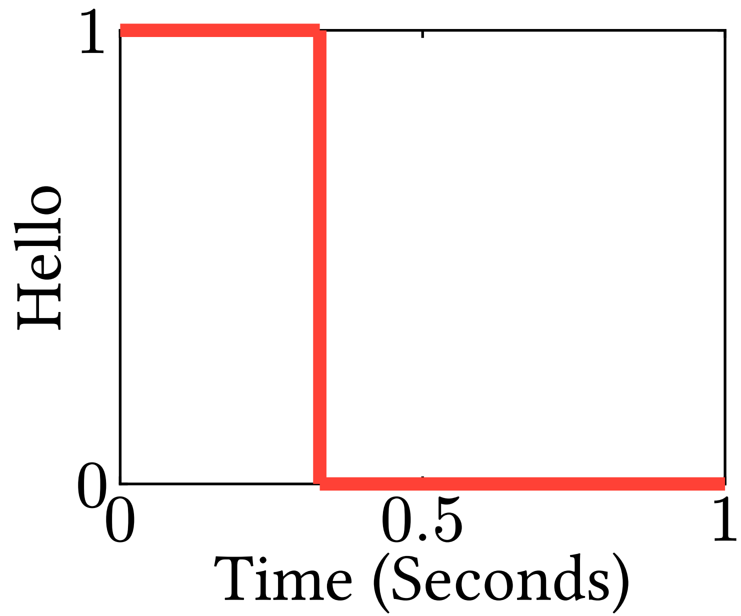


# Convolution

We use convolution to turn signals into useful signals




Convolution is translation equivariant and local

# Convolution

Convolution is the sum of products of a signal $x(t)$ and a **filter** $g(t)$

# Convolution

Convolution is the sum of products of a signal $x(t)$ and a **filter** $g(t)$

If the t is continuous in $x(t)$

$$x(t) * g(t) = \int_{-\infty}^{\infty} x(t - \tau) g(\tau) d\tau$$

# Convolution

Convolution is the sum of products of a signal $x(t)$ and a **filter** $g(t)$

If the t is continuous in $x(t)$

$$x(t) * g(t) = \int_{-\infty}^{\infty} x(t - \tau) g(\tau) d\tau$$

If the t is discrete in $x(t)$

$$x(t) * g(t) = \sum_{\tau=-\infty}^{\infty} x(t - \tau) g(\tau)$$

# Convolution

Convolution is the sum of products of a signal $x(t)$ and a **filter** $g(t)$

If the t is continuous in $x(t)$

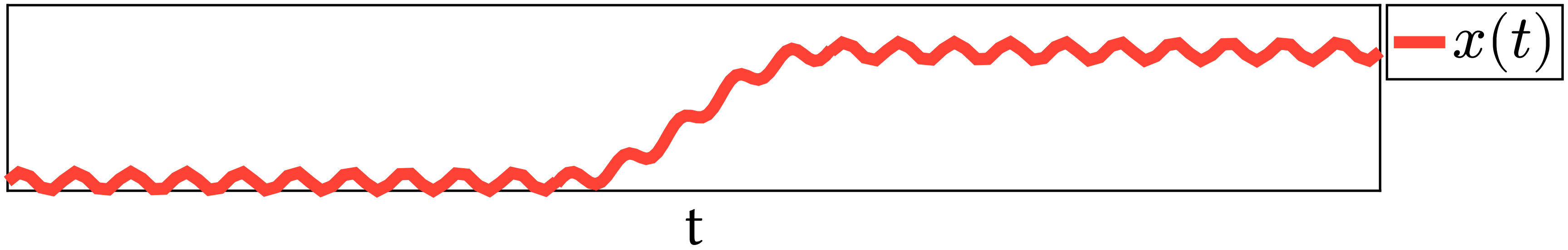
$$x(t) * g(t) = \int_{-\infty}^{\infty} x(t - \tau) g(\tau) d\tau$$

If the t is discrete in $x(t)$

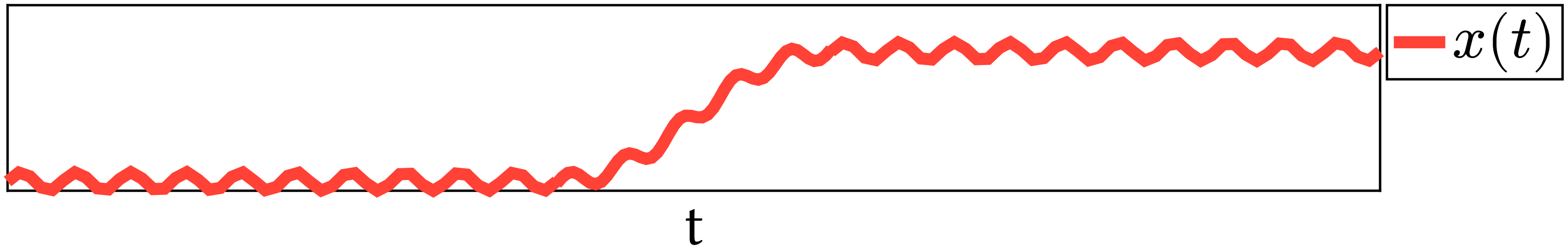
$$x(t) * g(t) = \sum_{\tau=-\infty}^{\infty} x(t - \tau) g(\tau)$$

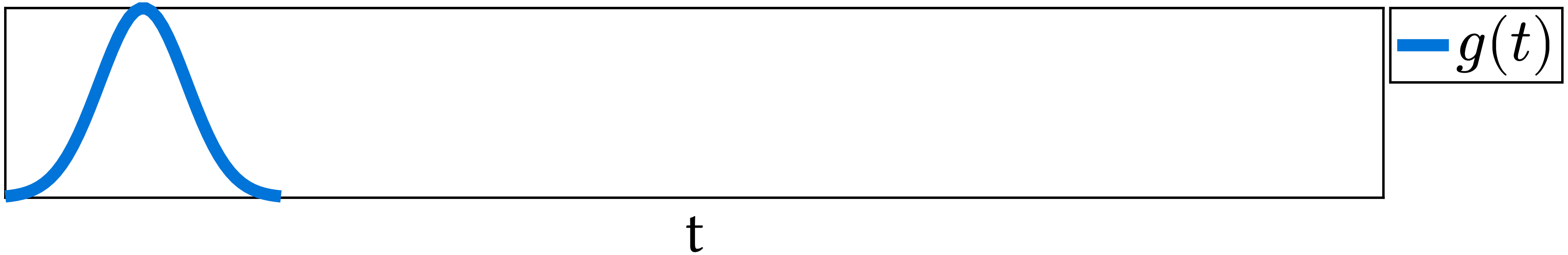
We slide the filter $g(t)$ across the signal $x(t)$

# Convolution

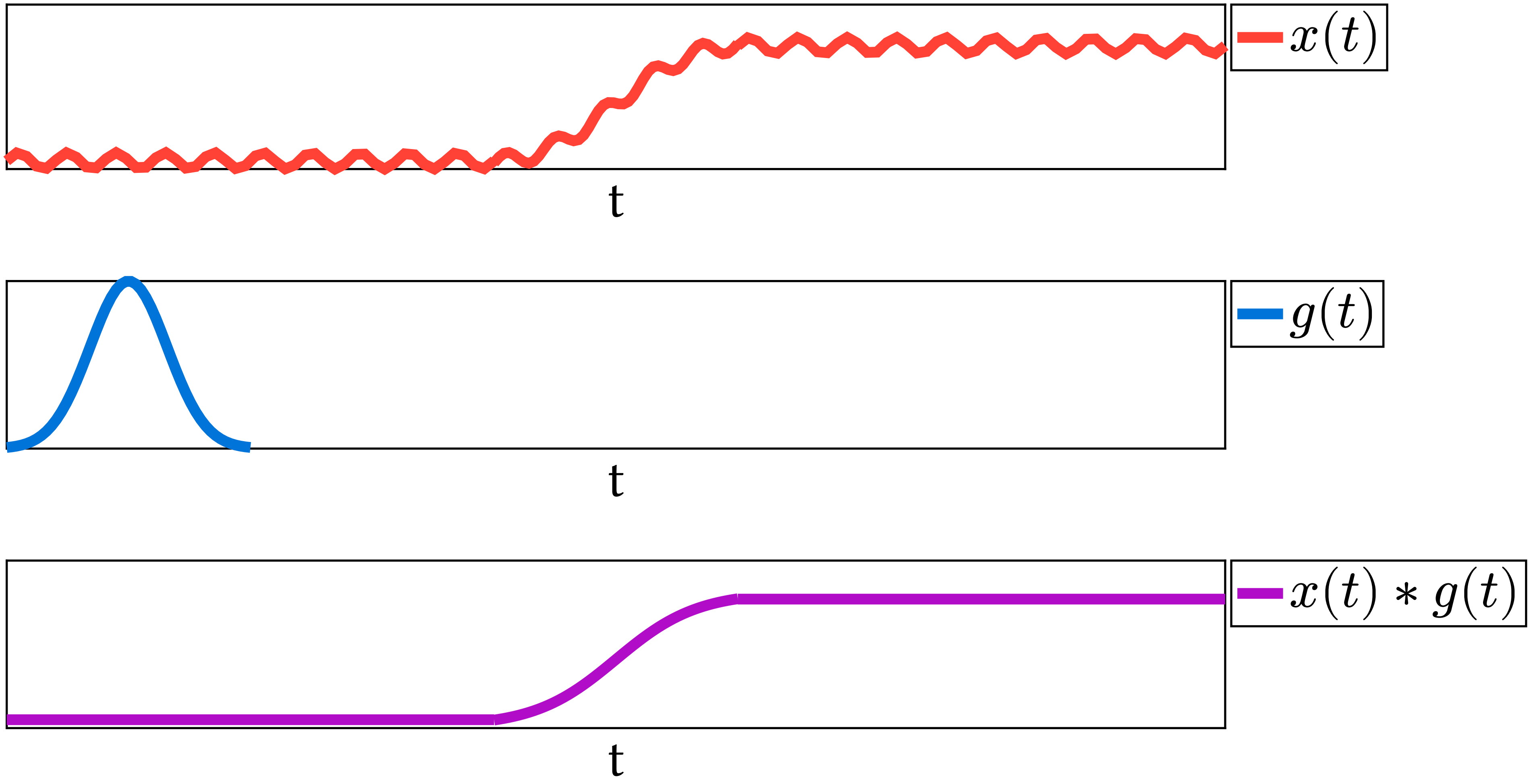


# Convolution



# Convolution



# Convolution

$$\begin{bmatrix} x(t) \\ g(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & & & \\ & & & & \end{bmatrix}$$

# Convolution

$$\begin{bmatrix} x(t) \\ g(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} \color{red}{1} & \color{red}{2} & 3 & 4 & 5 \\ \color{red}{2} & \color{red}{1} & & & \\ \color{red}{4} & & & & \end{bmatrix}$$

# Convolution

$$\begin{bmatrix} x(t) \\ g(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ & 2 & 1 & & \\ 4 & 7 & & & \end{bmatrix}$$

# Convolution

$$\begin{bmatrix} x(t) \\ g(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ & & 2 & 1 & \\ 4 & 5 & 10 & & \end{bmatrix}$$

# Convolution

$$\begin{bmatrix} x(t) \\ g(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & \color{red}{4} & \color{red}{5} \\ & & & \color{red}{2} & \color{red}{1} \\ 4 & 5 & 10 & \color{red}{13} & \end{bmatrix}$$

# Convolution

$$\begin{bmatrix} x(t) \\ g(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & & & \\ 4 & 5 & 10 & 13 & \end{bmatrix}$$

# Convolution

$$\begin{bmatrix} x(t) \\ g(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & & & \\ 4 & 5 & 10 & 13 & \end{bmatrix}$$

To make a convolution layer, we make the filter with trainable parameters

# Convolution

$$\begin{bmatrix} x(t) \\ g(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & & & \\ 4 & 5 & 10 & 13 & \end{bmatrix}$$

To make a convolution layer, we make the filter with trainable parameters

$$\begin{bmatrix} x(t) \\ g(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ \theta_2 & \theta_1 & & & \\ & & & & \end{bmatrix}$$

# Review

We can write both a perceptron and convolution in vector form

$$f(x(t), \boldsymbol{\theta}) = \sigma\left( \boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix} \right)$$

# Review

We can write both a perceptron and convolution in vector form

$$f(x(t), \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}\right) \qquad f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) \\ \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) \\ \vdots \end{bmatrix}$$

A convolution layer applies a “mini” perceptron to every few timesteps

# Review

We can write both a perceptron and convolution in vector form

$$f(x(t), \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \\ \vdots \end{bmatrix}\right) \qquad f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) \\ \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) \\ \vdots \end{bmatrix}$$

A convolution layer applies a "mini" perceptron to every few timesteps

The output size depends on the signal length

# Review

If we want a single output, we should **pool**

# Review

If we want a single output, we should **pool**

$$z(t) = f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left( \boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix} \right) & \sigma\left( \boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix} \right) & \dots \end{bmatrix}^\top$$

# Review

If we want a single output, we should **pool**

$$z(t) = f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) & \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) & \dots \end{bmatrix}^\top$$

$$\mathrm{SumPool}(z(t)) = \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) + \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) + \dots$$

# Review

If we want a single output, we should **pool**

$$z(t) = f(x(t), \boldsymbol{\theta}) = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) & \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) & \dots \end{bmatrix}^\top$$

$$\mathrm{SumPool}(z(t)) = \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.1) \\ x(0.2) \end{bmatrix}\right) + \sigma\left(\boldsymbol{\theta}^\top \begin{bmatrix} 1 \\ x(0.2) \\ x(0.3) \end{bmatrix}\right) + \dots$$

$$\mathrm{MeanPool}(z(t)) = \frac{1}{T - k + 1} \mathrm{SumPool}(z(t)); \quad \mathrm{MaxPool}(z(t)) = \max(z(t))$$

# Review

$$\begin{bmatrix}0&1&1&1&1&1&0&1\\0&1&1&1&1&1&1&1\\1&0&0&1&1&0&0&1\\1&0&0&1&1&0&0&1\\1&1&0&0&0&1&1&1\\1&1&0&0&0&1&1&1\\0&1&0&0&0&1&1&0\\1&0&1&1&1&1&1&1\end{bmatrix} * \begin{bmatrix}2&0\\0&1\end{bmatrix} + \begin{bmatrix}0&1&1&1&1&1&0&1\\0&1&1&1&1&1&1&1\\1&0&0&1&1&0&0&1\\1&0&0&1&1&0&0&1\\1&1&0&0&0&1&1&1\\1&1&0&0&0&1&1&1\\0&1&0&0&0&1&1&0\\1&0&1&1&1&1&1&1\end{bmatrix} * \begin{bmatrix}1&1\\0&1\end{bmatrix} + \begin{bmatrix}0&1&1&1&1&1&0&1\\0&1&1&1&1&1&1&1\\1&0&0&1&1&0&0&1\\1&0&0&1&1&0&0&1\\1&1&0&0&0&1&1&1\\1&1&0&0&0&1&1&1\\0&1&0&0&0&1&1&0\\1&0&1&1&1&1&1&1\end{bmatrix} * \begin{bmatrix}1&0\\0&0\end{bmatrix}$$

# Review

One last thing, **stride** allows you to “skip” cells during convolution

# Review

One last thing, **stride** allows you to "skip" cells during convolution

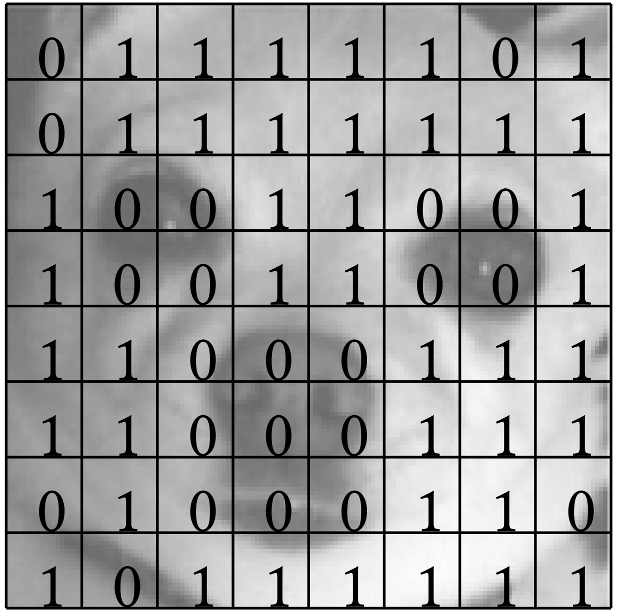
This can decrease the size of image without pooling

# Review

One last thing, **stride** allows you to “skip” cells during convolution

This can decrease the size of image without pooling

**Padding** adds zero pixels to the image to increase the output size




$$*$$

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

# Agenda

1. **Review**
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Agenda

1. Review
2. **Sequence Modeling**
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Sequence Modeling

We previously used convolution to model signals

# Sequence Modeling

We previously used convolution to model signals

Convolution is an electrical engineering approach to modeling sequences

# Sequence Modeling

We previously used convolution to model signals

Convolution is an electrical engineering approach to modeling sequences

Now, we will discuss a psychological approach to sequence modeling

# Sequence Modeling

We previously used convolution to model signals

Convolution is an electrical engineering approach to modeling sequences

Now, we will discuss a psychological approach to sequence modeling

We call these models **recurrent neural networks** (RNNs)

# Sequence Modeling

We previously used convolution to model signals

Convolution is an electrical engineering approach to modeling sequences

Now, we will discuss a psychological approach to sequence modeling

We call these models **recurrent neural networks** (RNNs)

You can solve temporal tasks using either convolution or RNNs

# Sequence Modeling

Convolution works over inputs of any variables (time, space, etc)

# Sequence Modeling

Convolution works over inputs of any variables (time, space, etc)

Recurrent neural networks only work with time

# Sequence Modeling

Convolution works over inputs of any variables (time, space, etc)

Recurrent neural networks only work with time

Convolution makes use of locality and translation equivariance properties

# Sequence Modeling

Convolution works over inputs of any variables (time, space, etc)

Recurrent neural networks only work with time

Convolution makes use of locality and translation equivariance properties

Recurrent models do not assume locality or equivariance

# Sequence Modeling

Convolution works over inputs of any variables (time, space, etc)

Recurrent neural networks only work with time

Convolution makes use of locality and translation equivariance properties

Recurrent models do not assume locality or equivariance

Equivariance and locality make learning more efficient, but not all problems have this structure

# Sequence Modeling

**Example 1:** You like dinosaurs as a child, you grow up and study dinosaurs for work

# Sequence Modeling

**Example 1:** You like dinosaurs as a child, you grow up and study dinosaurs for work

**Question:** Is this local?

# Sequence Modeling

**Example 1:** You like dinosaurs as a child, you grow up and study dinosaurs for work

**Question:** Is this local?

**Answer:** No, two events separated by 20 years

# Sequence Modeling

**Example 1:** You like dinosaurs as a child, you grow up and study dinosaurs for work

**Question:** Is this local?

**Answer:** No, two events separated by 20 years

# Sequence Modeling

**Example 2:** Your parent changes your diaper

# Sequence Modeling

**Example 2:** Your parent changes your diaper

**Question:** Translation equivariant?

# Sequence Modeling

**Example 2:** Your parent changes your diaper

**Question:** Translation equivariant?

No! Ok if you are a baby, different meaning if you are an adult!

# Sequence Modeling

**Example 2:** Your parent changes your diaper

**Question:** Translation equivariant?

No! Ok if you are a baby, different meaning if you are an adult!

# Sequence Modeling

**Example 3:** You hear a gunshot then see runners

# Sequence Modeling

**Example 3:** You hear a gunshot then see runners

**Question:** Translation equivariant?

# Sequence Modeling

**Example 3:** You hear a gunshot then see runners

**Question:** Translation equivariant?

**Answer:** No! (1) gunshot, (2) see runners, enjoy the race. (1) see runners, (2) hear gunshot, you start running too!

# Sequence Modeling

**Example 3:** You hear a gunshot then see runners

**Question:** Translation equivariant?

**Answer:** No! (1) gunshot, (2) see runners, enjoy the race. (1) see runners, (2) hear gunshot, you start running too!

**Question:** Any other examples?

# Sequence Modeling

Problems without locality and translation equivariance are difficult to solve with convolution

# Sequence Modeling

Problems without locality and translation equivariance are difficult to solve with convolution

For these problems, we need something else!

# Sequence Modeling

Problems without locality and translation equivariance are difficult to solve with convolution

For these problems, we need something else!

Humans experience time and process temporal data

# Sequence Modeling

Problems without locality and translation equivariance are difficult to solve with convolution

For these problems, we need something else!

Humans experience time and process temporal data

Can we design a neural network based on human perceptions of time?

# Agenda

1. Review
2. **Sequence Modeling**
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Agenda

# Composite Memory

How do humans experience time?

# Composite Memory

How do humans experience time?

Humans create memories

# Composite Memory

How do humans experience time?

Humans create memories

We experience time when we reason over our memories

# Composite Memory

John Locke (1690) believed that conciousness and identity arise from memories

# Composite Memory

John Locke (1690) believed that conciousness and identity arise from memories

If all your memories were erased, you would be a different person

# Composite Memory

John Locke (1690) believed that conciousness and identity arise from memories

If all your memories were erased, you would be a different person

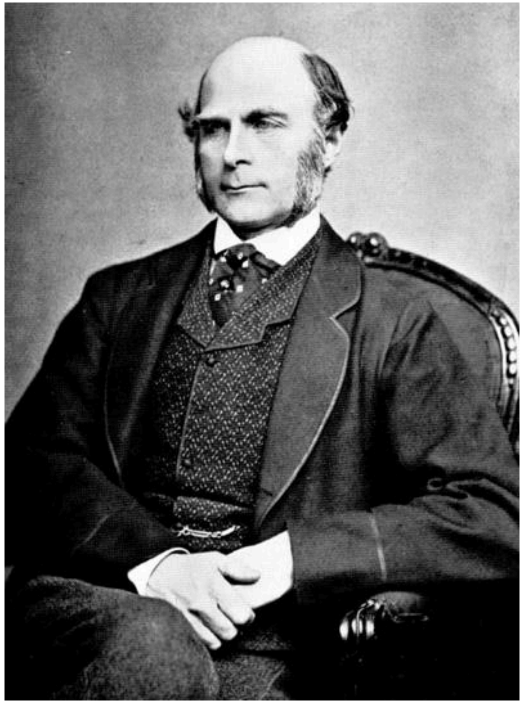
Without the ability to reason over memories, we would simply react to stimuli like bacteria

# Composite Memory

So how do we model memory in humans?

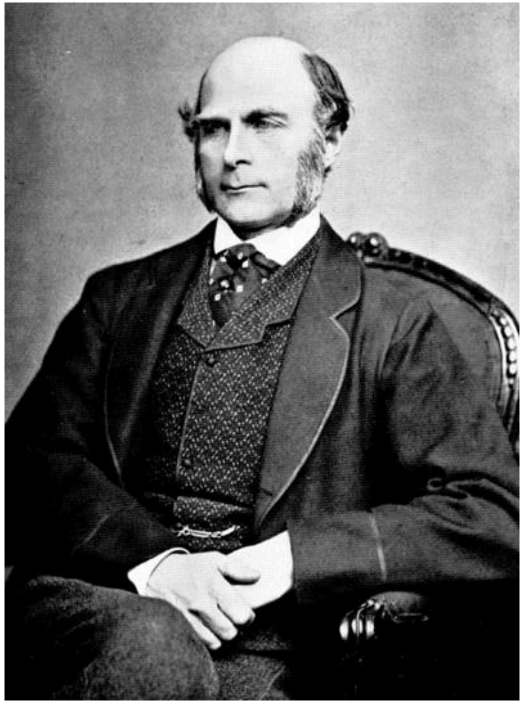
# Composite Memory

Francis Galton (1822-1911),
composite memory

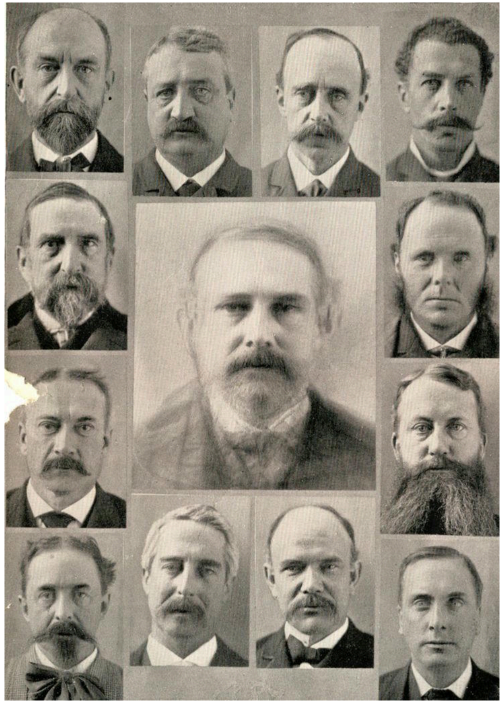


# Composite Memory

Francis Galton (1822-1911), composite memory



Composite photo of members of a party



# Composite Memory

**Task:** Model how our mind represents memories

# Composite Memory

**Task:** Model how our mind represents memories

$$X : \mathbb{R}^{T \times h \times w} \quad \text{Faces you see at the club}$$

# Composite Memory

**Task:** Model how our mind represents memories

$$X : \mathbb{R}^{T \times h \times w} \quad \text{Faces you see at the club}$$

$$Y : \mathbb{R}^{h \times w} \quad \text{The image in your mind}$$

$$f : X \times \Theta \mapsto Y$$

# Composite Memory

**Task:** Model how our mind represents memories

$$X : \mathbb{R}^{T \times h \times w} \quad \text{Faces you see at the club}$$

$$Y : \mathbb{R}^{h \times w} \quad \text{The image in your mind}$$

$$f : X \times \Theta \mapsto Y$$

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

# Composite Memory

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

# Composite Memory

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

**Question:** What if we see a new face?

# Composite Memory

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

**Question:** What if we see a new face?

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \left( \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

# Composite Memory

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

**Question:** What if we see a new face?

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \left( \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

**Question:** And another new face?

# Composite Memory

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

**Question:** What if we see a new face?

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \left( \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

**Question:** And another new face?

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \left( \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{newnew}}$$

# Agenda

1. Review
2. Sequence Modeling
3. **Composite Memory**
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. **Linear Recurrence**
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Linear Recurrence

We can rewrite this function **recurrently**

# Linear Recurrence

We can rewrite this function **recurrently**

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \underbrace{\left( \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right)}_{\boldsymbol{h}} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

# Linear Recurrence

We can rewrite this function **recurrently**

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \underbrace{\left( \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i \right)}_{\boldsymbol{h}} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{\text{new}}$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h}$$

# Linear Recurrence

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h}$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

# Linear Recurrence

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h} \qquad f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

# Linear Recurrence

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h} \qquad f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

# Linear Recurrence

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h} \qquad\qquad f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

$$\boldsymbol{h}_3 = f(\boldsymbol{h}_2, \boldsymbol{x}_3, \boldsymbol{\theta}) = \boldsymbol{h}_2 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$$

# Linear Recurrence

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h} \qquad f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

$$\boldsymbol{h}_3 = f(\boldsymbol{h}_2, \boldsymbol{x}_3, \boldsymbol{\theta}) = \boldsymbol{h}_2 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$$

$$\vdots$$

$$\boldsymbol{y} = \boldsymbol{h}_T = f(\boldsymbol{h}_{T-1}, \boldsymbol{x}_T, \boldsymbol{\theta}) = \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

# Linear Recurrence

$$\boldsymbol{x} \in \mathbb{R}^{d_x}, \quad \boldsymbol{h} \in \mathbb{R}^{d_h} \qquad f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

$$\boldsymbol{h}_3 = f(\boldsymbol{h}_2, \boldsymbol{x}_3, \boldsymbol{\theta}) = \boldsymbol{h}_2 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$$

$$\vdots$$

$$\boldsymbol{y} = \boldsymbol{h}_T = f(\boldsymbol{h}_{T-1}, \boldsymbol{x}_T, \boldsymbol{\theta}) = \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

# Linear Recurrence

**Question:** What is the meaning of $\boldsymbol{h}$ in humans?

# Linear Recurrence

**Question:** What is the meaning of $h$ in humans?

# Linear Recurrence

Right now, our model remembers everything

# Linear Recurrence

Right now, our model remembers everything

https://www.youtube.com/watch?v=IQ8Aak-k5Yc

# Linear Recurrence

Right now, our model remembers everything

https://www.youtube.com/watch?v=IQ8Aak-k5Yc

Humans cannot remember everything!

# Linear Recurrence

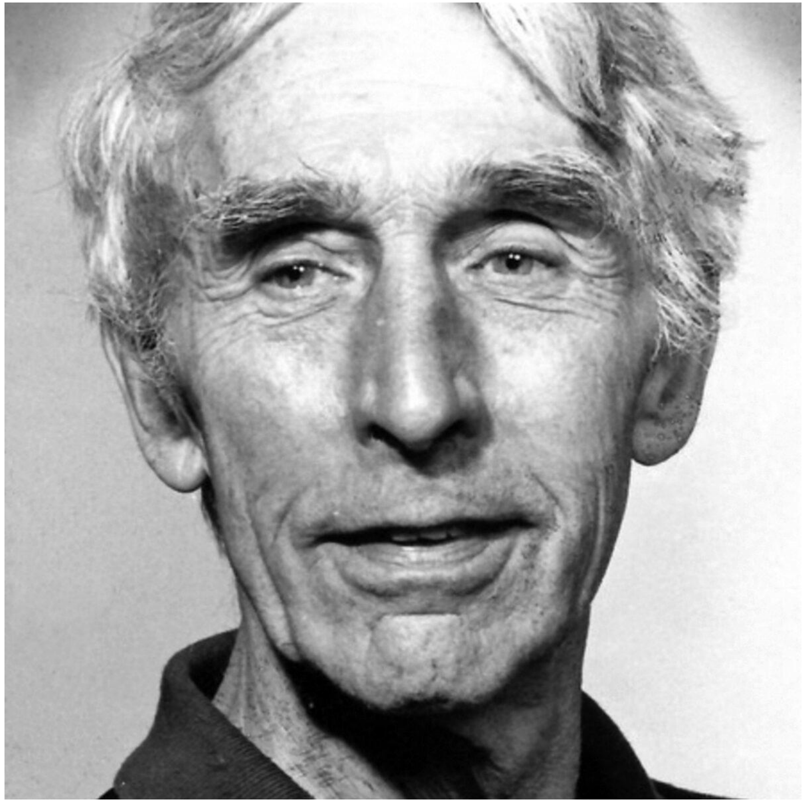
Right now, our model remembers everything

https://www.youtube.com/watch?v=IQ8Aak-k5Yc

Humans cannot remember everything!

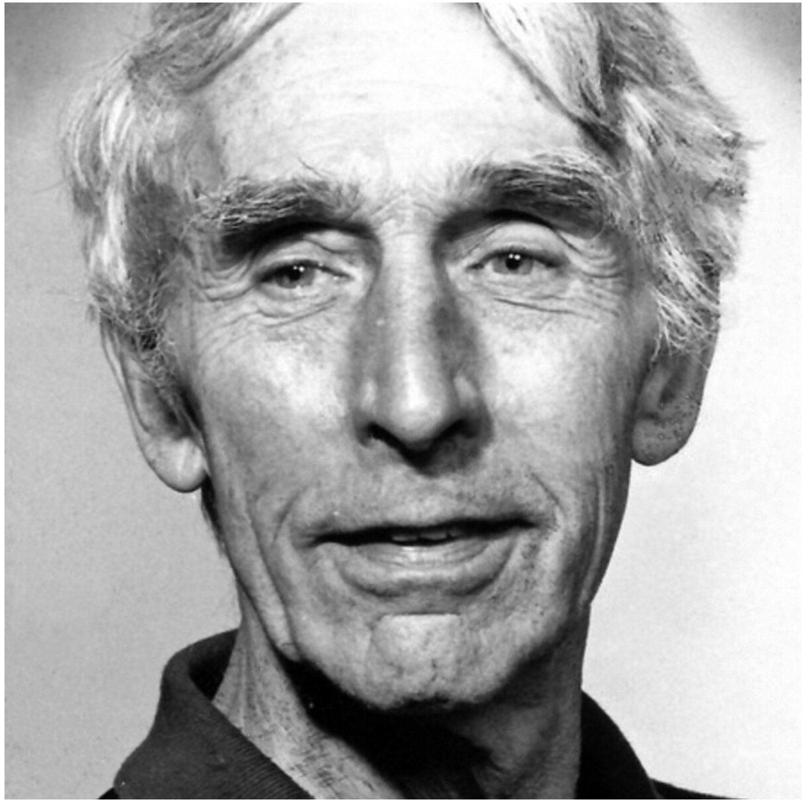
# Linear Recurrence

Murdock (1982)



# Linear Recurrence

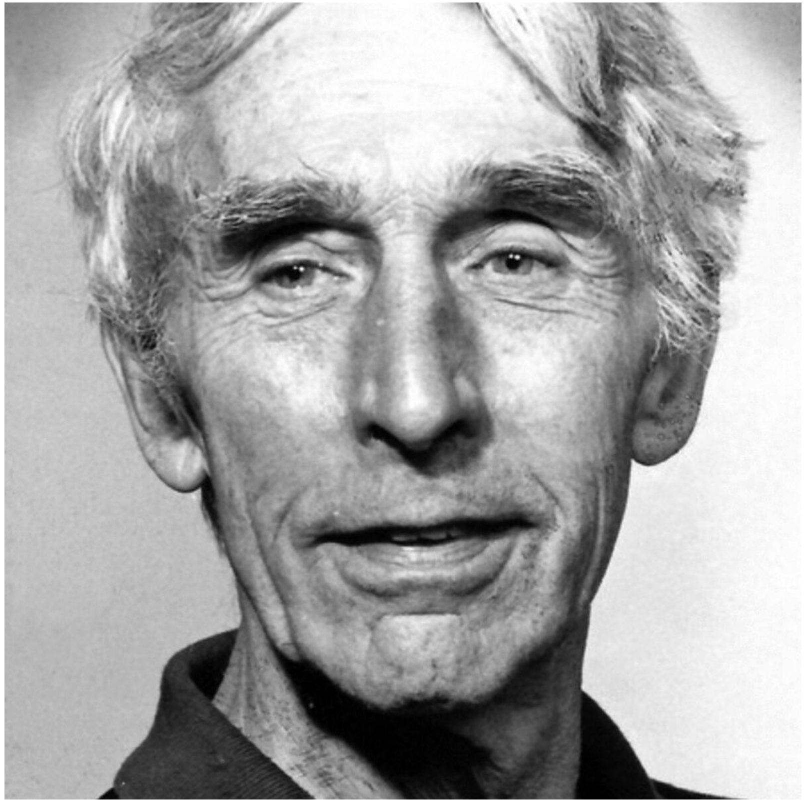
Murdock (1982)



$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

# Linear Recurrence

Murdock (1982)

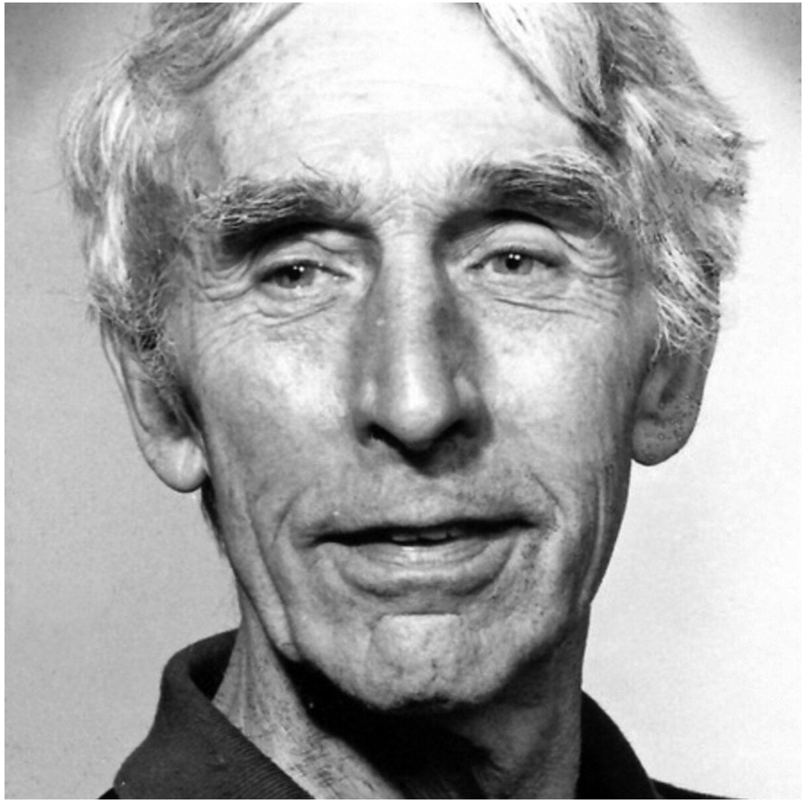


$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

**Key Idea:** $\lim_{T \rightarrow \infty} \gamma^T = 0$

# Linear Recurrence

Murdock (1982)



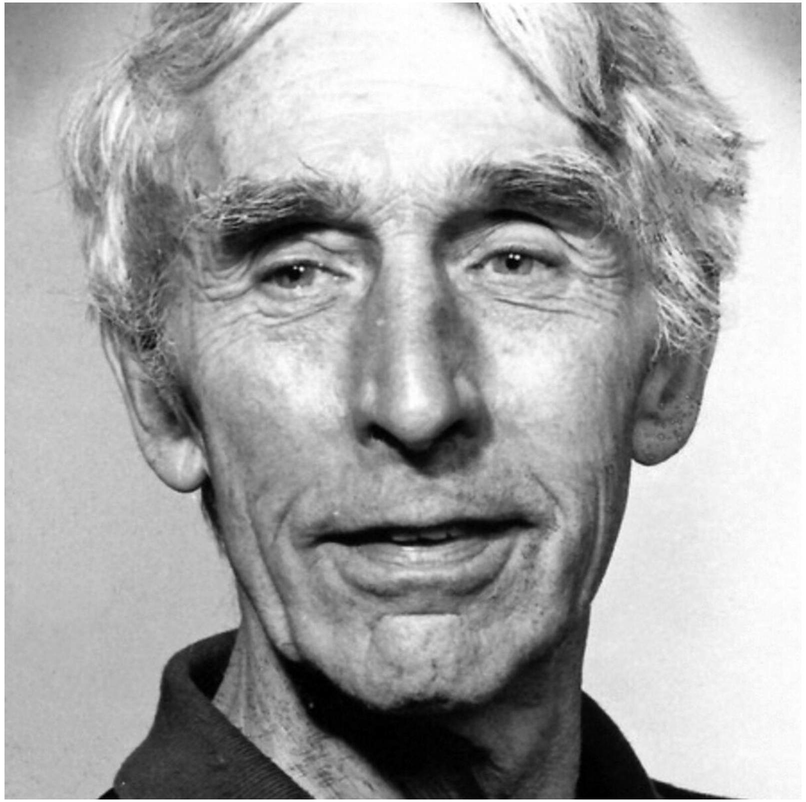
$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

**Key Idea:** $\lim_{T \rightarrow \infty} \gamma^T = 0$

Let $\gamma = 0.9$

# Linear Recurrence

Murdock (1982)



$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

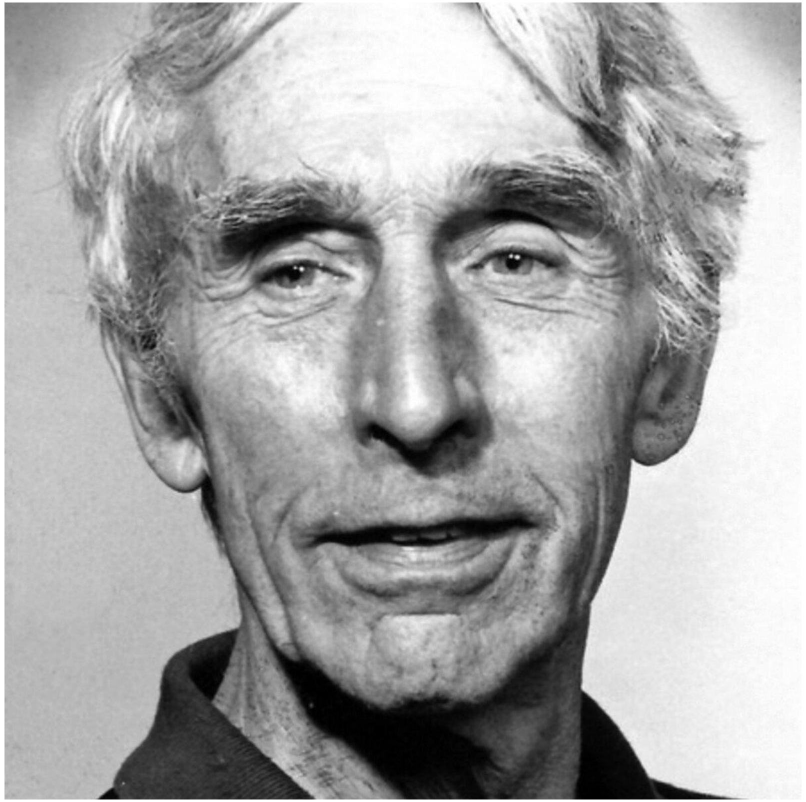
**Key Idea:** $\lim_{T \to \infty} \gamma^T = 0$

Let $\gamma = 0.9$

$$0.9 \cdot 0.9 \cdot \boldsymbol{h} = 0.81 \boldsymbol{h}$$

# Linear Recurrence

Murdock (1982)



$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = {\color{red}\gamma} \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

**Key Idea:** $\lim_{T \to \infty} \gamma^T = 0$

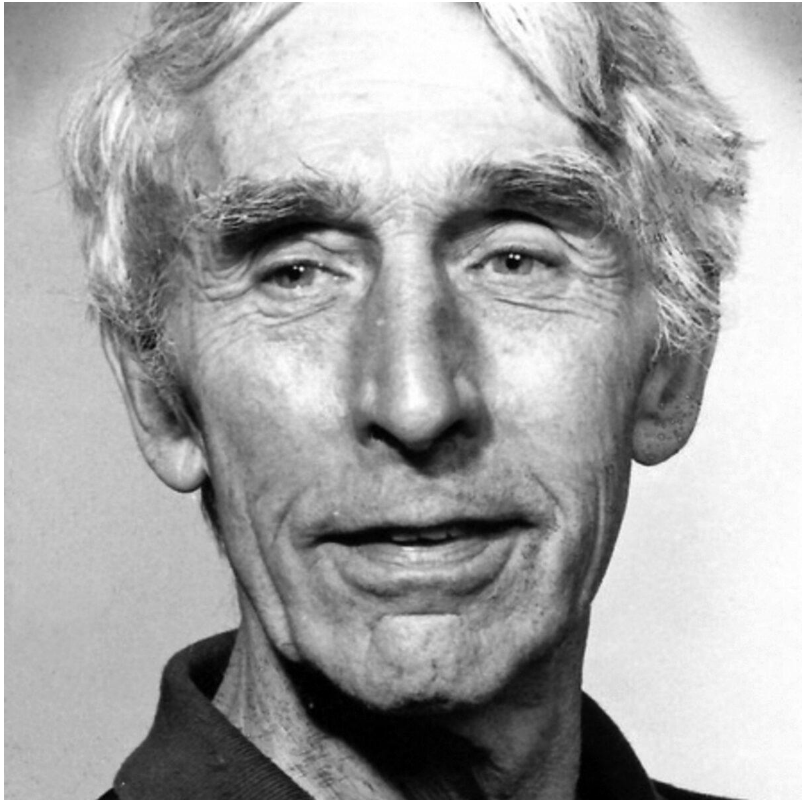
Let $\gamma = 0.9$

$$0.9 \cdot 0.9 \cdot \boldsymbol{h} = 0.81 \boldsymbol{h}$$

$$0.9 \cdot 0.9 \cdot 0.9 \cdot \boldsymbol{h} = 0.72 \boldsymbol{h}$$

# Linear Recurrence

Murdock (1982)



$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

**Key Idea:** $\lim_{T \to \infty} \gamma^T = 0$

Let $\gamma = 0.9$

$$0.9 \cdot 0.9 \cdot \boldsymbol{h} = 0.81 \boldsymbol{h}$$

$$0.9 \cdot 0.9 \cdot 0.9 \cdot \boldsymbol{h} = 0.72 \boldsymbol{h}$$

$$0.9 \cdot 0.9 \cdot 0.9 \cdot \ldots \cdot \boldsymbol{h} = 0$$

# Linear Recurrence

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

# Linear Recurrence

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

$$\boldsymbol{h}_T = \gamma \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

# Linear Recurrence

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

$$\boldsymbol{h}_T = \gamma \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

**Question:** What is $\boldsymbol{h}_{T-1}$ in terms of $\boldsymbol{h}_{T-2}$?

# Linear Recurrence

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

$$\boldsymbol{h}_T = \gamma \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

**Question:** What is $\boldsymbol{h}_{T-1}$ in terms of $\boldsymbol{h}_{T-2}$?

$$\gamma \boldsymbol{h}_{T-2} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1}$$

# Linear Recurrence

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

$$\boldsymbol{h}_T = \gamma \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

**Question:** What is $\boldsymbol{h}_{T-1}$ in terms of $\boldsymbol{h}_{T-2}$?

$$\gamma \boldsymbol{h}_{T-2} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1}$$

$$\boldsymbol{h}_T = \gamma(\gamma \boldsymbol{h}_{T-2} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1}) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

# Linear Recurrence

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

$$\boldsymbol{h}_T = \gamma \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

**Question:** What is $\boldsymbol{h}_{T-1}$ in terms of $\boldsymbol{h}_{T-2}$?

$$\gamma \boldsymbol{h}_{T-2} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1}$$

$$\boldsymbol{h}_T = \gamma(\gamma \boldsymbol{h}_{T-2} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1}) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

$$\boldsymbol{h}_T = \gamma(\gamma(\gamma \boldsymbol{h}_{T-3} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-2}) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1}) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

# Linear Recurrence

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}; \quad 0 < \gamma < 1$$

$$\boldsymbol{h}_T = \gamma \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

**Question:** What is $\boldsymbol{h}_{T-1}$ in terms of $\boldsymbol{h}_{T-2}$?

$$\gamma \boldsymbol{h}_{T-2} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1}$$

$$\boldsymbol{h}_T = \gamma(\gamma \boldsymbol{h}_{T-2} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1}) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

$$\boldsymbol{h}_T = \gamma(\gamma(\gamma \boldsymbol{h}_{T-3} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-2}) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1}) + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

$$\boldsymbol{h}_T = \gamma^3 \boldsymbol{h}_{T-3} + \gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-2} + \gamma \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

# Linear Recurrence

$$\boldsymbol{h}_T = \gamma^3 \boldsymbol{h}_{T-3} + \gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-2} + \gamma \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

# Linear Recurrence

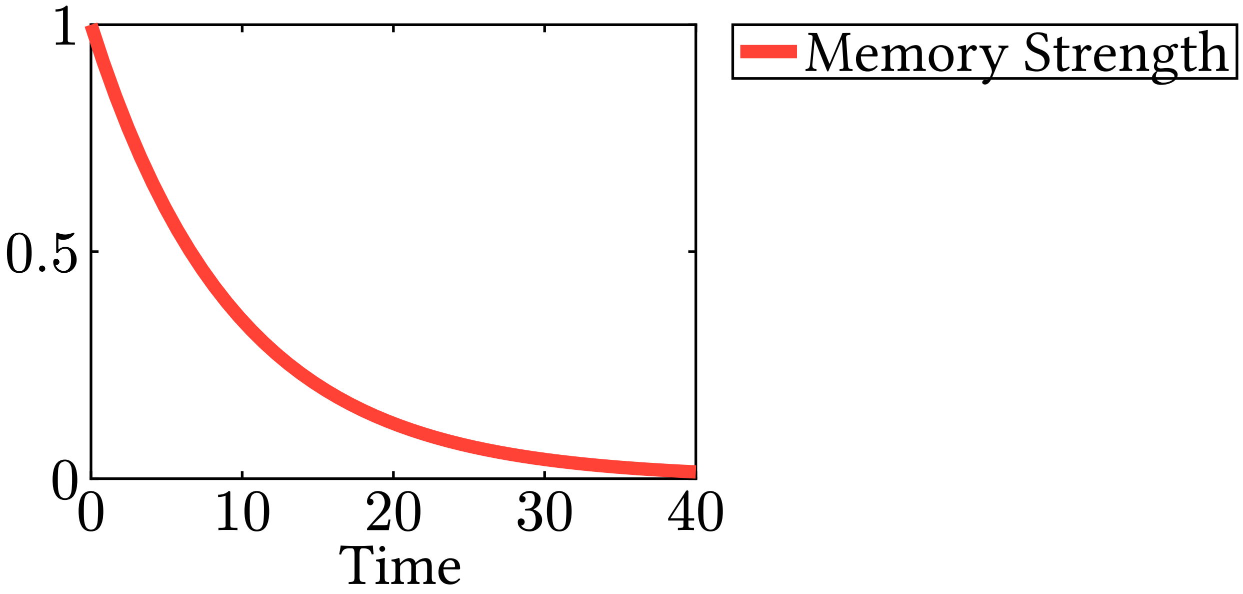
$$\boldsymbol{h}_T = \gamma^3 \boldsymbol{h}_{T-3} + \gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-2} + \gamma \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

$$\boldsymbol{h}_T = \sum_{i=1}^{T} \gamma^{T-i-1} \boldsymbol{\theta}^\top \boldsymbol{x}_i$$

# Linear Recurrence

$$\boldsymbol{h}_T = \gamma^3 \boldsymbol{h}_{T-3} + \gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-2} + \gamma \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

$$\boldsymbol{h}_T = \sum_{i=1}^{T} \gamma^{T-i-1} \boldsymbol{\theta}^\top \boldsymbol{x}_i$$



# Linear Recurrence

As $T$ increases, we add new information

# Linear Recurrence

As $T$ increases, we add new information

As $T$ increases, we slowly forget old information

# Linear Recurrence

As $T$ increases, we add new information

As $T$ increases, we slowly forget old information

The memory decay is smooth and differentiable

# Linear Recurrence

As $T$ increases, we add new information

As $T$ increases, we slowly forget old information

The memory decay is smooth and differentiable

We can learn the parameters using gradient descent

# Linear Recurrence

Morad et al., *Reinforcement Learning with Fast and Forgetful Memory.* Neural Information Processing Systems. (2024).

# Linear Recurrence

Morad et al., *Reinforcement Learning with Fast and Forgetful Memory.* Neural Information Processing Systems. (2024).

$$\boldsymbol{H}_t = \gamma \odot \boldsymbol{H}_{t-1} + g(\boldsymbol{x}_t)$$

# Linear Recurrence

Morad et al., *Reinforcement Learning with Fast and Forgetful Memory.* Neural Information Processing Systems. (2024).

$$\boldsymbol{H}_t = \gamma \odot \boldsymbol{H}_{t-1} + g(\boldsymbol{x}_t)$$

Our models learn to play board games and computer games

# Linear Recurrence

Morad et al., *Reinforcement Learning with Fast and Forgetful Memory.* Neural Information Processing Systems. (2024).

$$\boldsymbol{H}_t = \boldsymbol{\gamma} \odot \boldsymbol{H}_{t-1} + g(\boldsymbol{x}_t)$$

Our models learn to play board games and computer games

Outperforms other recurrent models (LSTM, GRU, etc)

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. **Linear Recurrence**
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. **Scans**
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Scans

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \gamma \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \gamma \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

$$\boldsymbol{h}_3 = f(\boldsymbol{h}_2, \boldsymbol{x}_3, \boldsymbol{\theta}) = \gamma \boldsymbol{h}_2 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$$

$$\vdots$$

$$\boldsymbol{y} = \boldsymbol{h}_T = f(\boldsymbol{h}_{T-1}, \boldsymbol{x}_T, \boldsymbol{\theta}) = \gamma \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

# Scans

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \gamma \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \gamma \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

$$\boldsymbol{h}_3 = f(\boldsymbol{h}_2, \boldsymbol{x}_3, \boldsymbol{\theta}) = \gamma \boldsymbol{h}_2 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$$

$$\vdots$$

$$\boldsymbol{y} = \boldsymbol{h}_T = f(\boldsymbol{h}_{T-1}, \boldsymbol{x}_T, \boldsymbol{\theta}) = \gamma \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

How do we compute $\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T$ on a computer?

# Scans

$$\boldsymbol{h}_1 = f(\boldsymbol{0}, \boldsymbol{x}_1, \boldsymbol{\theta}) = \gamma \boldsymbol{0} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

$$\boldsymbol{h}_2 = f(\boldsymbol{h}_1, \boldsymbol{x}_2, \boldsymbol{\theta}) = \gamma \boldsymbol{h}_1 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$$

$$\boldsymbol{h}_3 = f(\boldsymbol{h}_2, \boldsymbol{x}_3, \boldsymbol{\theta}) = \gamma \boldsymbol{h}_2 + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$$

$$\vdots$$

$$\boldsymbol{y} = \boldsymbol{h}_T = f(\boldsymbol{h}_{T-1}, \boldsymbol{x}_T, \boldsymbol{\theta}) = \gamma \boldsymbol{h}_{T-1} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_T$$

How do we compute $\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T$ on a computer?

We use an algebraic operation called a **scan**

# Scans

Our function $f$ is just defined for a single $X$

# Scans

Our function $f$ is just defined for a single $X$

$$f : H \times X \times \Theta \mapsto H$$

# Scans

Our function $f$ is just defined for a single $X$

$$f : H \times X \times \Theta \mapsto H$$

To extend $f$ to sequences, we scan $f$ over the inputs

$$\text{scan}(f) : H \times X^T \times \Theta \mapsto H^T$$

# Scans

Our function $f$ is just defined for a single $X$

$$f : H \times X \times \Theta \mapsto H$$

To extend $f$ to sequences, we scan $f$ over the inputs

$$\text{scan}(f) : H \times X^T \times \Theta \mapsto H^T$$

$$\text{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_T \end{bmatrix}$$

# Scans

$$f : H \times X \times \Theta \mapsto H, \quad \text{scan}(f) : H \times X^T \times \Theta \mapsto H^T$$

# Scans

$$f : H \times X \times \Theta \mapsto H, \quad \text{scan}(f) : H \times X^T \times \Theta \mapsto H^T$$

$$\text{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_T \end{bmatrix}$$

# Scans

$$f : H \times X \times \Theta \mapsto H, \quad \text{scan}(f) : H \times X^T \times \Theta \mapsto H^T$$

$$\text{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_T \end{bmatrix}$$

$$\text{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} f(h_0, x_1, \boldsymbol{\theta}) \\ f(h_1, x_2, \boldsymbol{\theta}) \\ \vdots \\ f(h_{T-1}, x_T, \boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} f(h_0, x_1) \\ f(f(h_0, x_1), x_2) \\ \vdots \\ f(\ldots f(h_0, x_1) \ldots, x_T) \end{bmatrix}$$

# Scans

```python
import jax
import jax.numpy as jnp

T, d_h = 10, 4
xs, h0 = jnp.ones((T, d_h)), jnp.zeros((d_h,))
theta = (jnp.ones((d_h,)), jnp.ones((d_h, d_h))) # (b, W)

def f(h, x):
    b, W = theta
    result = h + W.T @ x + b
    return result, result # return one, return all

_, hs = jax.lax.scan(f, init=h0, xs=xs) # Scan f over x
```

# Scans

`torch` does NOT have built-in scans, and is very slow compared to `jax`

# Scans

`torch` does NOT have built-in scans, and is very slow compared to `jax`

We will write our own scan

# Scans

`torch` does NOT have built-in scans, and is very slow compared to `jax`

We will write our own scan

```
def scan(f, h, xs):
    # h shape is (d_h,)
    # xs shape is (T, d_x)
    hs = []
    for x in xs:
        h = f(h, x, theta)
        hs.append(h)
    # output shape is (T, d_h)
    return hs
```

# Scans

```python
import torch
T, d_h = 10, 4
xs, h0 = torch.ones((T, d_h)), torch.zeros((d_h,))
theta = (torch.ones((d_h,)), torch.ones((d_h, d_h)))
def f(h, x):
    b, W = theta
    result = h + W.T @ x + b
    return result # h

hs = scan(f, h0, xs)
```

# Scans

Many deep learning courses do not teach scans

# Scans

Many deep learning courses do not teach scans

I teach scans because they are an important part of future LLMs

# Scans

Many deep learning courses do not teach scans

I teach scans because they are an important part of future LLMs

OpenAI, Google, etc are currently experimenting with LLMs that use **associative scans**

# Scans

Many deep learning courses do not teach scans

I teach scans because they are an important part of future LLMs

OpenAI, Google, etc are currently experimenting with LLMs that use **associative scans**

Standard LLM: ~8000 words, LLM + associative scan: 1M+ words

# Scans

Many deep learning courses do not teach scans

I teach scans because they are an important part of future LLMs

OpenAI, Google, etc are currently experimenting with LLMs that use **associative scans**

Standard LLM: ~8000 words, LLM + associative scan: 1M+ words

An associative scan is a very fast scan we use when $f$ obeys the associative property

# Scans

Many deep learning courses do not teach scans

I teach scans because they are an important part of future LLMs

OpenAI, Google, etc are currently experimenting with LLMs that use **associative scans**

Standard LLM: ~8000 words, LLM + associative scan: 1M+ words

An associative scan is a very fast scan we use when $f$ obeys the associative property

**Question:** Does $f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$ obey the associative property?

# Scans

Many deep learning courses do not teach scans

I teach scans because they are an important part of future LLMs

OpenAI, Google, etc are currently experimenting with LLMs that use **associative scans**

Standard LLM: ~8000 words, LLM + associative scan: 1M+ words

An associative scan is a very fast scan we use when $f$ obeys the associative property

**Question:** Does $f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$ obey the associative property?

**Answer:** Yes, linear operations obey the associative property

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. **Scans**
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. **Output Modeling**
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Output Modeling

We are almost done defining recurrent models

# Output Modeling

We are almost done defining recurrent models

There is one more step we must consider, **memory recall**

# Output Modeling

We are almost done defining recurrent models

There is one more step we must consider, **memory recall**

$\boldsymbol{h}$ represents all memories, but humans only access a few memories at once

# Output Modeling

We are almost done defining recurrent models

There is one more step we must consider, **memory recall**

$\boldsymbol{h}$ represents all memories, but humans only access a few memories at once

**Example:** I ask you your favorite ice cream flavor

# Output Modeling

We are almost done defining recurrent models

There is one more step we must consider, **memory recall**

$\boldsymbol{h}$ represents all memories, but humans only access a few memories at once

**Example:** I ask you your favorite ice cream flavor

You recall previous times you ate ice cream, but not your phone number

# Output Modeling

We are almost done defining recurrent models

There is one more step we must consider, **memory recall**

$\boldsymbol{h}$ represents all memories, but humans only access a few memories at once

**Example:** I ask you your favorite ice cream flavor

You recall previous times you ate ice cream, but not your phone number

# Output Modeling

We model recall using a function $g$

# Output Modeling

We model recall using a function $g$

$$g : H \times X \times \Theta \mapsto Y$$

# Output Modeling

We model recall using a function $g$

$$g : H \times X \times \Theta \mapsto Y$$

$g$ searches your memories $h$ using the input $x$, to produce output $y$

# Output Modeling

We model recall using a function $g$

$$g : H \times X \times \Theta \mapsto Y$$

$g$ searches your memories $h$ using the input $x$, to produce output $y$

$\boldsymbol{x}$ : What is your favorite ice cream flavor?

# Output Modeling

We model recall using a function $g$

$$g : H \times X \times \Theta \mapsto Y$$

$g$ searches your memories $h$ using the input $x$, to produce output $y$

$\boldsymbol{x}$ : What is your favorite ice cream flavor?

$\boldsymbol{h}$ : Everything you remember (hometown, birthday, etc)

# Output Modeling

We model recall using a function $g$

$$g : H \times X \times \Theta \mapsto Y$$

$g$ searches your memories $h$ using the input $x$, to produce output $y$

$\boldsymbol{x}$ : What is your favorite ice cream flavor?

$\boldsymbol{h}$ : Everything you remember (hometown, birthday, etc)

$g$ : Searches your memories for ice cream memories, and responds "chocolate"

# Output Modeling

We model recall using a function $g$

$$g : H \times X \times \Theta \mapsto Y$$

$g$ searches your memories $h$ using the input $x$, to produce output $y$

$\boldsymbol{x}$ : What is your favorite ice cream flavor?

$\boldsymbol{h}$ : Everything you remember (hometown, birthday, etc)

$g$ : Searches your memories for ice cream memories, and responds "chocolate"

Now, combine $f$ and $g$

# Output Modeling

**Step 1:** Perform scan to find recurrent states

# Output Modeling

**Step 1:** Perform scan to find recurrent states

$$\begin{bmatrix} \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_T \end{bmatrix} = \mathrm{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_f\right)$$

# Output Modeling

**Step 1:** Perform scan to find recurrent states

$$\begin{bmatrix} \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_T \end{bmatrix} = \text{scan}(f)\left( \boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_f \right)$$

**Step 2:** Perform recall on recurrent states

# Output Modeling

**Step 1:** Perform scan to find recurrent states

$$\begin{bmatrix} \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_T \end{bmatrix} = \mathrm{scan}(f) \left( \boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_f \right)$$

**Step 2:** Perform recall on recurrent states

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_T \end{bmatrix} = \begin{bmatrix} g(\boldsymbol{h}_1, \boldsymbol{x}_1, \boldsymbol{\theta}_g) \\ \vdots \\ g(\boldsymbol{h}_T, \boldsymbol{x}_T, \boldsymbol{\theta}_g) \end{bmatrix}$$

# Output Modeling

**Step 1:** Perform scan to find recurrent states

$$\begin{bmatrix} \boldsymbol{h}_1 \\ \vdots \\ \boldsymbol{h}_T \end{bmatrix} = \operatorname{scan}(f)\left( \boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_f \right)$$

**Step 2:** Perform recall on recurrent states

$$\begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_T \end{bmatrix} = \begin{bmatrix} g(\boldsymbol{h}_1, \boldsymbol{x}_1, \boldsymbol{\theta}_g) \\ \vdots \\ g(\boldsymbol{h}_T, \boldsymbol{x}_T, \boldsymbol{\theta}_g) \end{bmatrix}$$

Questions? This is on the homework

# Output Modeling

We defined:

- Recurrent function $f$

# Output Modeling

We defined:

- Recurrent function $f$
- Scanned recurrence $\text{scan}(f)$

# Output Modeling

We defined:

- Recurrent function $f$
- Scanned recurrence $\text{scan}(f)$
- Output function $g$

To run:

# Output Modeling

We defined:

- Recurrent function $f$
- Scanned recurrence $\text{scan}(f)$
- Output function $g$

To run:

- Execute $\text{scan}(f)$ over inputs to make recurrent states

# Output Modeling

We defined:

- Recurrent function $f$
- Scanned recurrence $\text{scan}(f)$
- Output function $g$

To run:

- Execute $\text{scan}(f)$ over inputs to make recurrent states
- Execute $g$ over recurrent states to make outputs

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. **Output Modeling**
7. Recurrent Loss Functions
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. **Recurrent Loss Functions**
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Recurrent Loss Functions

Let us examine some example tasks:

# Recurrent Loss Functions

Let us examine some example tasks:

- Clock

# Recurrent Loss Functions

Let us examine some example tasks:

- Clock
- Explaining a video

# Recurrent Loss Functions

**Task:** Clock – keep track of time

# Recurrent Loss Functions

**Task:** Clock – keep track of time

Every second, the second hand ticks

# Recurrent Loss Functions

**Task:** Clock – keep track of time

Every second, the second hand ticks

Every minute, the minute hand ticks

# Recurrent Loss Functions

**Task:** Clock – keep track of time

Every second, the second hand ticks

Every minute, the minute hand ticks

Add up the ticks to know the time

# Recurrent Loss Functions

**Task:** Clock – keep track of time

Every second, the second hand ticks

Every minute, the minute hand ticks

Add up the ticks to know the time

$$X \in \{0, 1\}^2, \quad Y \in \mathbb{R}^2$$

# Recurrent Loss Functions

Example input sequence:

$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 1 \end{bmatrix}$$

# Recurrent Loss Functions

Example input sequence:

$$\begin{bmatrix} 0 & 1 \\ 0 & 1 \\ \vdots & \vdots \\ 1 & 1 \end{bmatrix}$$

Desired output sequence

$$\begin{bmatrix} 0 & 1 \\ 0 & 2 \\ \vdots & \vdots \\ m & s \end{bmatrix}$$

# Recurrent Loss Functions

We have a ground truth for each input $y_i$

# Recurrent Loss Functions

We have a ground truth for each input $y_i$

Can use square error

# Recurrent Loss Functions

We have a ground truth for each input $y_i$

Can use square error

First, scan $f$ over the inputs to find $h$

$$\boldsymbol{h}_{[i],j} = \text{scan}(f)\left(\boldsymbol{h}_0, \boldsymbol{x}_{[i]}, \boldsymbol{\theta}_f\right)$$

# Recurrent Loss Functions

We have a ground truth for each input $y_i$

Can use square error

First, scan $f$ over the inputs to find $h$

$$\boldsymbol{h}_{[i],j} = \text{scan}(f)\left(\boldsymbol{h}_0, \boldsymbol{x}_{[i]}, \boldsymbol{\theta}_f\right)$$

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{T} \left[g\left(\boldsymbol{h}_{[i],j}, \boldsymbol{x}_{[i],j}, \boldsymbol{\theta}_g\right) - \boldsymbol{y}_{[i],j}^2\right]^2$$

# Recurrent Loss Functions

We have a ground truth for each input $y_i$

Can use square error

First, scan $f$ over the inputs to find $h$

$$\boldsymbol{h}_{[i],j} = \operatorname{scan}(f)\left(\boldsymbol{h}_0, \boldsymbol{x}_{[i]}, \boldsymbol{\theta}_f\right)$$

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{T} \left[g\left(\boldsymbol{h}_{[i],j}, \boldsymbol{x}_{[i],j}, \boldsymbol{\theta}_g\right) - \boldsymbol{y}_{[i],j}^2\right]^2$$

Onto the next task

# Recurrent Loss Functions

**Task:** Watch a video, then explain it to me

$$X \in \mathbb{Z}^{3 \times 32 \times 32}, \quad Y \in \{\text{comedy show}, \text{action movie}, \dots\}$$

# Recurrent Loss Functions

**Task:** Watch a video, then explain it to me

$$X \in \mathbb{Z}^{3 \times 32 \times 32}, \quad Y \in \{\text{comedy show}, \text{action movie}, ...\}$$

Example input sequence: $\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_T \end{bmatrix}$

Example output: "dancing dog"

# Recurrent Loss Functions

**Task:** Watch a video, then explain it to me

$$X \in \mathbb{Z}^{3 \times 32 \times 32}, \quad Y \in \{\text{comedy show}, \text{action movie}, \ldots\}$$

Example input sequence: $\begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_T \end{bmatrix}$

Example output: "dancing dog"

Unlike before, we have many inputs but just one output!

# Recurrent Loss Functions

We will use the classification loss

# Recurrent Loss Functions

We will use the classification loss

We scan $f$ over the sequence, the compute $g$ for the final timestep

# Recurrent Loss Functions

We will use the classification loss

We scan $f$ over the sequence, the compute $g$ for the final timestep

$$h_{[i],j} = \text{scan}(f)\left(\boldsymbol{h}_0, \boldsymbol{x}_{[i]}, \boldsymbol{\theta}_f\right)$$

# Recurrent Loss Functions

We will use the classification loss

We scan $f$ over the sequence, the compute $g$ for the final timestep

$$h_{[i],j} = \mathrm{scan}(f)\left(\boldsymbol{h}_0, \boldsymbol{x}_{[i]}, \boldsymbol{\theta}_f\right)$$

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_y} y_{[i],j} \log g\left(\boldsymbol{h}_{[i],T}, \boldsymbol{x}_{[i],T}\right)_j$$

# Recurrent Loss Functions

We will use the classification loss

We scan $f$ over the sequence, the compute $g$ for the final timestep

$$h_{[i],j} = \text{scan}(f)\left(\boldsymbol{h}_0, \boldsymbol{x}_{[i]}, \boldsymbol{\theta}_f\right)$$

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{\theta}) = \sum_{i=1}^{n} \sum_{j=1}^{d_y} y_{[i],j} \log g\left(\boldsymbol{h}_{[i],T}, \boldsymbol{x}_{[i],T}\right)_j$$

We only care about the $\boldsymbol{h}_T$

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. **Recurrent Loss Functions**
8. Backpropagation through Time
9. Recurrent Neural Networks
10. Coding

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. **Backpropagation through Time**
9. Recurrent Neural Networks
10. Coding

# Backpropagation through Time

1. We created the model

# Backpropagation through Time

1. We created the model
2. We found the loss function

# Backpropagation through Time

1. We created the model
2. We found the loss function
3. Now we need to find the parameters!

# Backpropagation through Time

1. We created the model
2. We found the loss function
3. Now we need to find the parameters!

Just like all other neural networks, we train recurrent models using gradient descent

# Backpropagation through Time

1. We created the model
2. We found the loss function
3. Now we need to find the parameters!

Just like all other neural networks, we train recurrent models using gradient descent

**Step 1:** Compute gradient

# Backpropagation through Time

1. We created the model
2. We found the loss function
3. Now we need to find the parameters!

Just like all other neural networks, we train recurrent models using gradient descent

**Step 1:** Compute gradient

**Step 2:** Update parameters

# Backpropagation through Time

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

# Backpropagation through Time

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

**Question:** What is $\nabla_{\boldsymbol{\theta}} f$?

# Backpropagation through Time

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \gamma \boldsymbol{h} + \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}$$

**Question:** What is $\nabla_{\boldsymbol{\theta}} f$?

$$\nabla_{\boldsymbol{\theta}} f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \overline{\boldsymbol{x}}^\top$$

# Backpropagation through Time

$$\operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} f(h_0, x_1, \boldsymbol{\theta}) \\ f(h_1, x_2, \boldsymbol{\theta}) \\ \vdots \\ f(h_{T-1}, x_T, \boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} f(h_0, x_1) \\ f(f(h_0, x_1), x_2) \\ \vdots \\ f(\ldots f(h_0, x_1) \ldots, x_T) \end{bmatrix}$$

**Question:** What is $\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)$?

**Hint:** Chain rule

# Backpropagation through Time

$$\mathrm{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} f(h_0, x_1, \boldsymbol{\theta}) \\ f(h_1, x_2, \boldsymbol{\theta}) \\ \vdots \\ f(h_{T-1}, x_T, \boldsymbol{\theta}) \end{bmatrix} = \begin{bmatrix} f(h_0, x_1) \\ f(f(h_0, x_1), x_2) \\ \vdots \\ f(...f(h_0, x_1)..., x_T) \end{bmatrix}$$

**Question:** What is $\nabla_{\boldsymbol{\theta}} \mathrm{scan}(f)$? **Hint:** Chain rule

$$\nabla_{\boldsymbol{\theta}} \mathrm{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \nabla_{\boldsymbol{\theta}}[f](f(h_0, x_1), x_2) \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \vdots \\ \nabla_{\boldsymbol{\theta}}[f](...f(h_0, x_1)..., x_T) \cdot ... \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \end{bmatrix}$$

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. **Backpropagation through Time**
9. Recurrent Neural Networks
10. Coding

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. **Recurrent Neural Networks**
10. Coding

# Recurrent Neural Networks

If $f$ is a neural network, then we have a recurrent neural network

# Recurrent Neural Networks

The first recurrent neural network was the **Elman Network**

# Recurrent Neural Networks

The first recurrent neural network was the **Elman Network**

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

# Recurrent Neural Networks

The first recurrent neural network was the **Elman Network**

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$g(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}_3^\top \overline{\boldsymbol{h}}$$

# Recurrent Neural Networks

The first recurrent neural network was the **Elman Network**

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$g(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}_3^\top \overline{\boldsymbol{h}}$$

$\boldsymbol{h}$ grows large and causes exploding gradients, $\sigma$ should be sigmoid!

# Recurrent Neural Networks

The first recurrent neural network was the **Elman Network**

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$g(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}_3^\top \overline{\boldsymbol{h}}$$

$\boldsymbol{h}$ grows large and causes exploding gradients, $\sigma$ should be sigmoid!

**Question:** Max value of $\sigma(\boldsymbol{h})$?

# Recurrent Neural Networks

The first recurrent neural network was the **Elman Network**

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$g(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}_3^\top \overline{\boldsymbol{h}}$$

$\boldsymbol{h}$ grows large and causes exploding gradients, $\sigma$ should be sigmoid!

**Question:** Max value of $\sigma(\boldsymbol{h})$? 1!

# Recurrent Neural Networks

The first recurrent neural network was the **Elman Network**

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$g(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}_3^\top \overline{\boldsymbol{h}}$$

$\boldsymbol{h}$ grows large and causes exploding gradients, $\sigma$ should be sigmoid!

**Question:** Max value of $\sigma(\boldsymbol{h})$? 1!

**Question:** Anything missing from our linear model?

# Recurrent Neural Networks

The first recurrent neural network was the **Elman Network**

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$g(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \boldsymbol{\theta}_3^\top \overline{\boldsymbol{h}}$$

$\boldsymbol{h}$ grows large and causes exploding gradients, $\sigma$ should be sigmoid!

**Question:** Max value of $\sigma(\boldsymbol{h})$? 1!

**Question:** Anything missing from our linear model?

**Answer:** Forgetting!

# Recurrent Neural Networks

Add forgetting

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{h}}\right)$$

Add forgetting

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{h}}\right)$$

**Question:** $\sigma$ is sigmoid. What is range/codomain of $f_{\text{forget}}$?

# Recurrent Neural Networks

Add forgetting

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{h}}\right)$$

**Question:** $\sigma$ is sigmoid. What is range/codomain of $f_{\text{forget}}$?

**Answer:** $[0, 1]$

# Recurrent Neural Networks

Add forgetting

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{h}}\right)$$

**Question:** $\sigma$ is sigmoid. What is range/codomain of $f_{\text{forget}}$?

**Answer:** $[0, 1]$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_3^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_4^\top \overline{\boldsymbol{h}} \odot f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta})\right)$$

# Recurrent Neural Networks

Add forgetting

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{h}}\right)$$

**Question:** $\sigma$ is sigmoid. What is range/codomain of $f_{\text{forget}}$?

**Answer:** $[0, 1]$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_3^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_4^\top \overline{\boldsymbol{h}} \odot f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta})\right)$$

When $f_{\text{forget} < 1}$, we forget!

**Minimal gated unit** (MGU)

**Minimal gated unit** (MGU)

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{h}}\right)$$

**Minimal gated unit** (MGU)

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{h}}\right)$$

$$f_2(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_3^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_4^\top \left(f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h}\right)\right)$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h} + \left(1 - f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta})\right) \odot f_2(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta})$$

**Minimal gated unit** (MGU)

$$f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{h}}\right)$$

$$f_2(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_3^\top \overline{\boldsymbol{x}} + \boldsymbol{\theta}_4^\top \left(f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h}\right)\right)$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h} + \left(1 - f_{\text{forget}}(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta})\right) \odot f_2(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta})$$

Left term forgets old, right term replaces forgotten memories

# Recurrent Neural Networks

There are even more complicated models

# Recurrent Neural Networks

There are even more complicated models

- Long Short-Term Memory (LSTM)

# Recurrent Neural Networks

There are even more complicated models

- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

# Recurrent Neural Networks

There are even more complicated models

- Long Short-Term Memory (LSTM)
- Gated Recurrent Unit (GRU)

LSTM has 6 different functions! Too complicated to review

# Recurrent Neural Networks

Elman network $f$: $$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

# Recurrent Neural Networks

Elman network $f$: $$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

**Question:** What is the gradient for scan$(f)$?

# Recurrent Neural Networks

Elman network $f$: $$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

**Question:** What is the gradient for $\text{scan}(f)$?

$$\nabla_{\boldsymbol{\theta}} \text{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \nabla_{\boldsymbol{\theta}}[f](f(h_0, x_1), x_2) \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \vdots \\ \nabla_{\boldsymbol{\theta}}[f](\ldots f(h_0, x_1)\ldots, x_T) \cdot \ldots \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \end{bmatrix}$$

# Recurrent Neural Networks

Elman network $f$: $$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

**Question:** What is the gradient for $\mathrm{scan}(f)$?

$$\nabla_{\boldsymbol{\theta}} \mathrm{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \nabla_{\boldsymbol{\theta}}[f](f(h_0, x_1), x_2) \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \vdots \\ \nabla_{\boldsymbol{\theta}}[f](...f(h_0, x_1)..., x_T) \cdot ... \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \end{bmatrix}$$

# Recurrent Neural Networks

Elman network $f$: $$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

# Recurrent Neural Networks

Elman network $f$: $$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \nabla_{\boldsymbol{\theta}}[f](f(h_0, x_1), x_2) \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \vdots \\ \nabla_{\boldsymbol{\theta}}[f](\ldots f(h_0, x_1)\ldots, x_T) \cdot \ldots \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \end{bmatrix}$$

# Recurrent Neural Networks

Elman network $f$: $$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \nabla_{\boldsymbol{\theta}}[f](f(h_0, x_1), x_2) \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \vdots \\ \nabla_{\boldsymbol{\theta}}[f](\ldots f(h_0, x_1) \ldots, x_T) \cdot \ldots \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \end{bmatrix}$$

$$\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1) \overline{\boldsymbol{x}}_1^\top \\ \nabla_{\boldsymbol{\theta}}[\sigma]\left(\boldsymbol{\theta}_2 \overline{\boldsymbol{h}}_1, x_2\right) \nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1) \overline{\boldsymbol{x}}_1^\top \\ \vdots \end{bmatrix}$$

# Recurrent Neural Networks

Elman network $f$: $$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{h}} + \boldsymbol{\theta}_2^\top \overline{\boldsymbol{x}}\right)$$

$$\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \nabla_{\boldsymbol{\theta}}[f](f(h_0, x_1), x_2) \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \\ \vdots \\ \nabla_{\boldsymbol{\theta}}[f](\dots f(h_0, x_1) \dots, x_T) \cdot \dots \cdot \nabla_{\boldsymbol{\theta}}[f](h_0, x_1) \end{bmatrix}$$

$$\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1) \overline{\boldsymbol{x}}_1^\top \\ \nabla_{\boldsymbol{\theta}}[\sigma]\left(\boldsymbol{\theta}_2 \overline{\boldsymbol{h}}_1, x_2\right) \nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1) \overline{\boldsymbol{x}}_1^\top \\ \vdots \end{bmatrix}$$

# Recurrent Neural Networks

$$\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1)\overline{\boldsymbol{x}}_1^\top \\ \nabla_{\boldsymbol{\theta}}[\sigma]\left(\boldsymbol{\theta}_2 \overline{\boldsymbol{h}}_1, \boldsymbol{x}_2\right)\nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1)\overline{\boldsymbol{x}}_1^\top \\ \vdots \end{bmatrix}$$

# Recurrent Neural Networks

$$\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1)\overline{\boldsymbol{x}}_1^\top \\ \nabla_{\boldsymbol{\theta}}[\sigma]\left(\boldsymbol{\theta}_2 \overline{\boldsymbol{h}}_1, \boldsymbol{x}_2\right)\nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1)\overline{\boldsymbol{x}}_1^\top \\ \vdots \end{bmatrix}$$

**Question:** What's the problem?

# Recurrent Neural Networks

$$\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1)\overline{\boldsymbol{x}}_1^\top \\ \nabla_{\boldsymbol{\theta}}[\sigma]\left(\boldsymbol{\theta}_2 \overline{\boldsymbol{h}}_1, \boldsymbol{x}_2\right)\nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1)\overline{\boldsymbol{x}}_1^\top \\ \vdots \end{bmatrix}$$

**Question:** What's the problem?

**Answer:** Vanishing gradient

$$\nabla[\sigma] \cdot \nabla[\sigma] \cdot \ldots = 0$$

# Recurrent Neural Networks

$$\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1)\overline{\boldsymbol{x}}_1^\top \\ \nabla_{\boldsymbol{\theta}}[\sigma]\left(\boldsymbol{\theta}_2 \overline{\boldsymbol{h}}_1, \boldsymbol{x}_2\right)\nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1)\overline{\boldsymbol{x}}_1^\top \\ \vdots \end{bmatrix}$$

**Question:** What's the problem?

**Answer:** Vanishing gradient

$$\nabla[\sigma] \cdot \nabla[\sigma] \cdot \ldots = 0$$

**Question:** What can we do?

# Recurrent Neural Networks

$$\nabla_{\boldsymbol{\theta}} \operatorname{scan}(f)\left(\boldsymbol{h}_0, \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \begin{bmatrix} \nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1)\overline{\boldsymbol{x}}_1^\top \\ \nabla_{\boldsymbol{\theta}}[\sigma]\left(\boldsymbol{\theta}_2 \overline{\boldsymbol{h}}_1, \boldsymbol{x}_2\right)\nabla_{\boldsymbol{\theta}}[\sigma](\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1)\overline{\boldsymbol{x}}_1^\top \\ \vdots \end{bmatrix}$$

**Question:** What's the problem?

**Answer:** Vanishing gradient $\nabla[\sigma] \cdot \nabla[\sigma] \cdot \ldots = 0$

**Question:** What can we do?

All RNNs suffer from either exploding gradient (ReLU) or vanishing gradient (sigmoid). Active area of research!

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. **Backpropagation through Time**
9. Recurrent Neural Networks
10. Coding

# Agenda

1. Review
2. Sequence Modeling
3. Composite Memory
4. Linear Recurrence
5. Scans
6. Output Modeling
7. Recurrent Loss Functions
8. Backpropagation through Time
9. **Recurrent Neural Networks**
10. Coding

# Recurrent Neural Networks

Jax RNN https://colab.research.google.com/drive/147z7FNGyERV8oQ_4gZmxDVdeoNt0hKta#scrollTo=TUMonlJ1u8Va

Homework https://colab.research.google.com/drive/1CNaDxx1yJ4-phyMvgbxECL8ydZYBGQGt?usp=sharing

Makeup lecture Saturday October 26, 13:00-16:00