

# Introduction to **Deep Learning** for Question Answering

Jens Lehmann, Andreas Both, Ioanna Lytra, Mohnish Dubey, Denis Lukovnikov,  
Kuldeep Singh, Gaurav Maheshwari, Priyansh Trivedi



ESWC 2018, Heraklion

# Outline

Motivation

Neural Networks

Recurrent Neural Networks

Trainings and Optimizations

Hands On

Future Directions

**Who is the father of  
Luke Skywalker?**

What is the best dessert  
in the world?



Name  
one ex  
president  
of United States?



Where is  
ESWC 2018 held?

# Motivation

# Traditional ML based NLP Pipeline



# Feature engineering for NLP is difficult.

Verbs in the sentence forms the relations.

# Feature engineering for NLP is difficult.

- ✓ Name all the movies in which Robert Downey Jr **acted**?
- ✗ Name all the **parents** of Robert Downey Jr?

# Feature engineering for NLP is difficult.

Verbs in the sentence forms the relations.

Words before the noun represents the relations.

# Feature engineering for NLP is difficult.

- ✓ Name all the movies in which Robert Downey Jr **acted**?
- ✓ Name the **parents** of Robert Downey Jr?
- ✗ Name Robert Downey Jr's **parent**?

# Feature engineering for NLP is difficult.

Verbs in the sentence forms the relations.

Words before the noun represents the relations.

Special keywords like “of”, “whose”, “not” etc.

# NLP Ambiguity

Time flies like a arrow

V/S

Fruit flies like a banana

# NLP Ambiguity

"I scream" vs. "ice cream" -

unionized = “union” + “ized” or “un” + “ionized”

# Manual Feature engineering

Double counting.

Extending the rules to different domain is non-trivial

# Traditional ML based NLP Pipeline



## Issues

- ✗ Cost Time
- ✗ Domain Expertise
- ✗ Manual Efforts

**LET ME SPRINKLE SOME**

**DEEP LEARNING MAGIC**

# Deep learning based NLP Pipeline

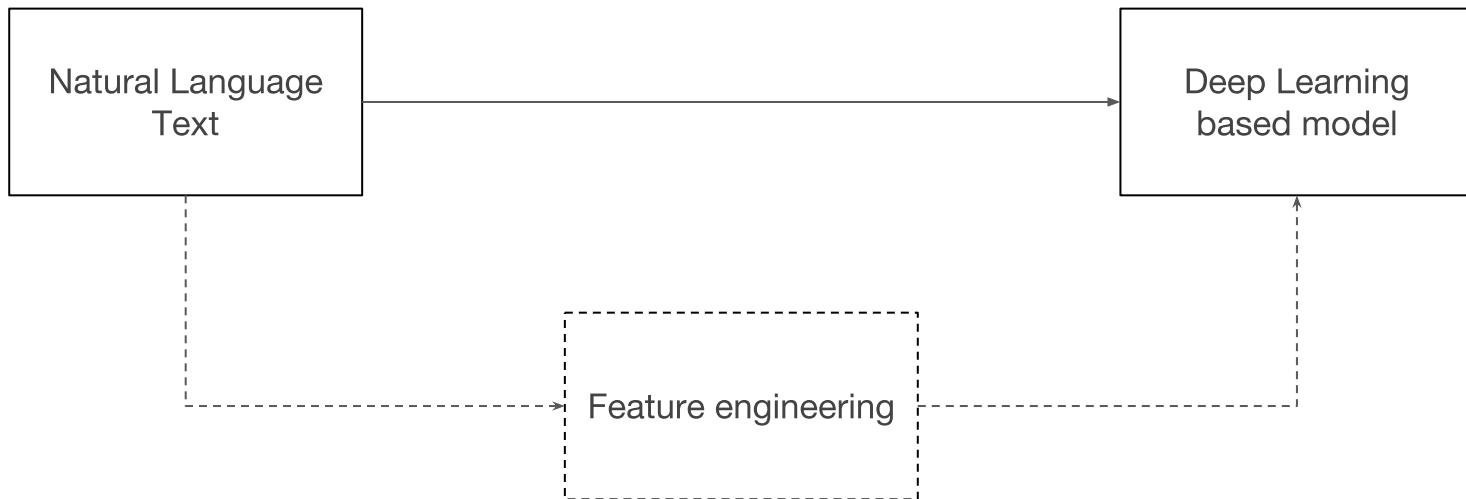


Ideally learns the features automatically a.k.a  
**Representation learning**

More data beats clever algorithms, but better  
data beats more data

- Peter Norvig

# Deep learning based NLP Pipeline



# Empirical Validity

Sentiment Analysis

Image Captioning

Argument Mining

Structured Question Answering

Textual Entailment

Semantic Parsing

Word Embeddings

Language Modeling

Paraphrasing

Text based Question Answering

Named Entity Recognition

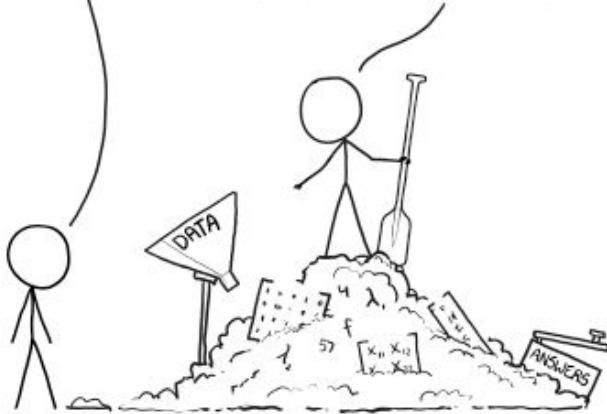
Machine Translation

THIS IS YOUR MACHINE LEARNING SYSTEM?

YUP! YOU POUR THE DATA INTO THIS BIG  
PILE OF LINEAR ALGEBRA, THEN COLLECT  
THE ANSWERS ON THE OTHER SIDE.

WHAT IF THE ANSWERS ARE WRONG?

JUST STIR THE PILE UNTIL  
THEY START LOOKING RIGHT.



# Why now

Faster, cheaper GPU's

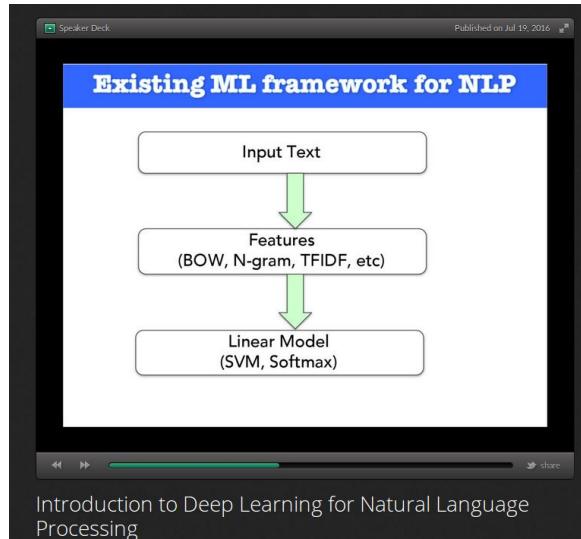
Simple to use libraries

Large amount of data

# Representation learning

Attempts to learn the features automatically, thus reducing the need of domain expertise, manual feature engineering.

Existing ML framework



# Why?

1. Resilience to structural, syntactic variations. Finding implicit patterns.
  - a. Manual Rule making is difficult.
2. Easily keeping up with updated KBs (different KBs?)
3. Distributional Semantics Argument
4. Empirical validity.
  - a. Deep Learning is really effective in NLP tasks
  - b. Do be careful. Its not as simple as throwing deep learning on a jumbled mess of gum, hair and tape and magic happens (xkcd comic)
5. Why now?
  - a. Faster machine more data better regularization

## Learning representations

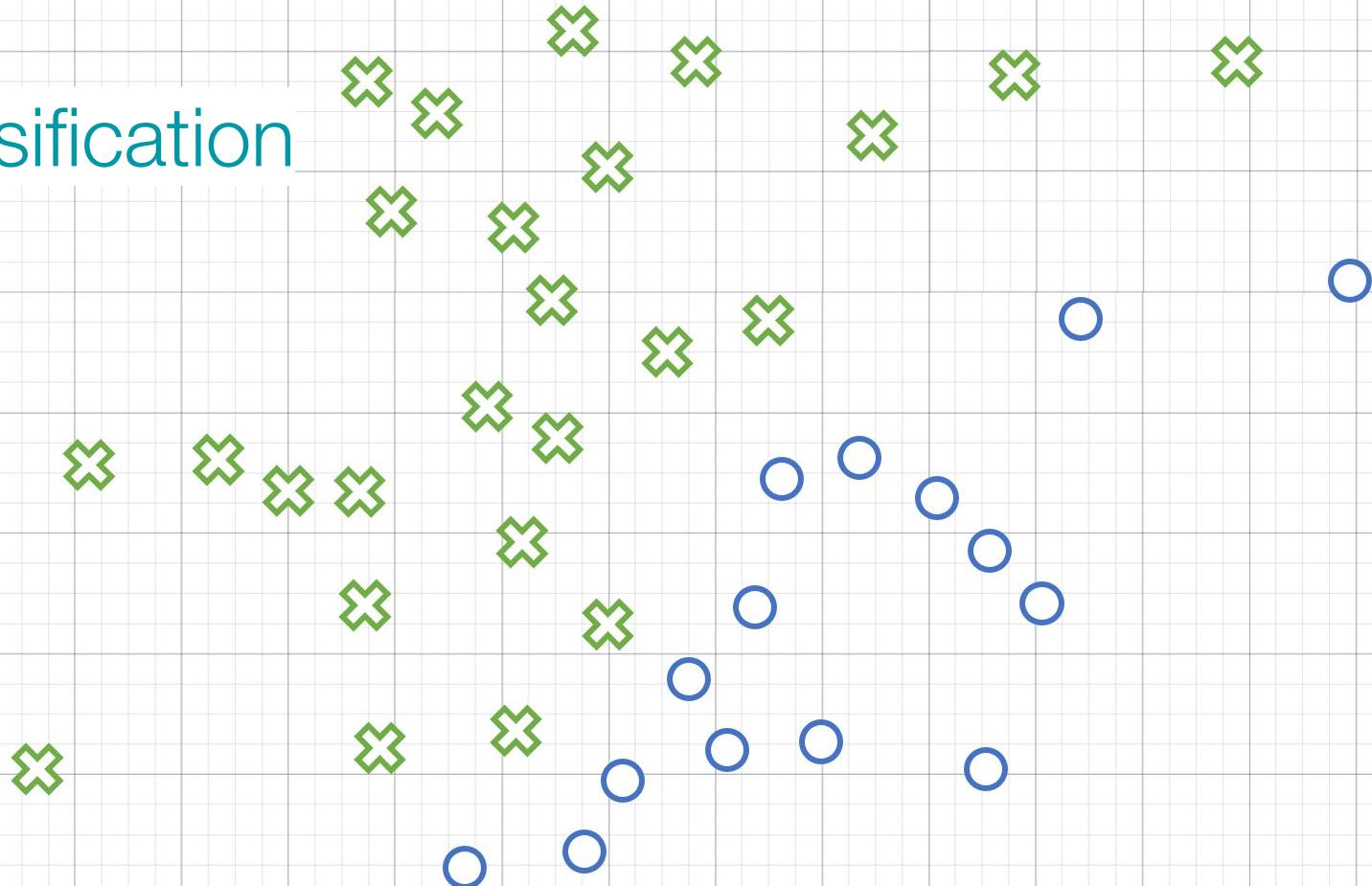
- Handling recursivity of human language.
  - Human sentences are composed of words and phrases, some structure
  - Need compositionality in ML models
  - Recursion: same op applied repeatedly on different components

Rules dont work.

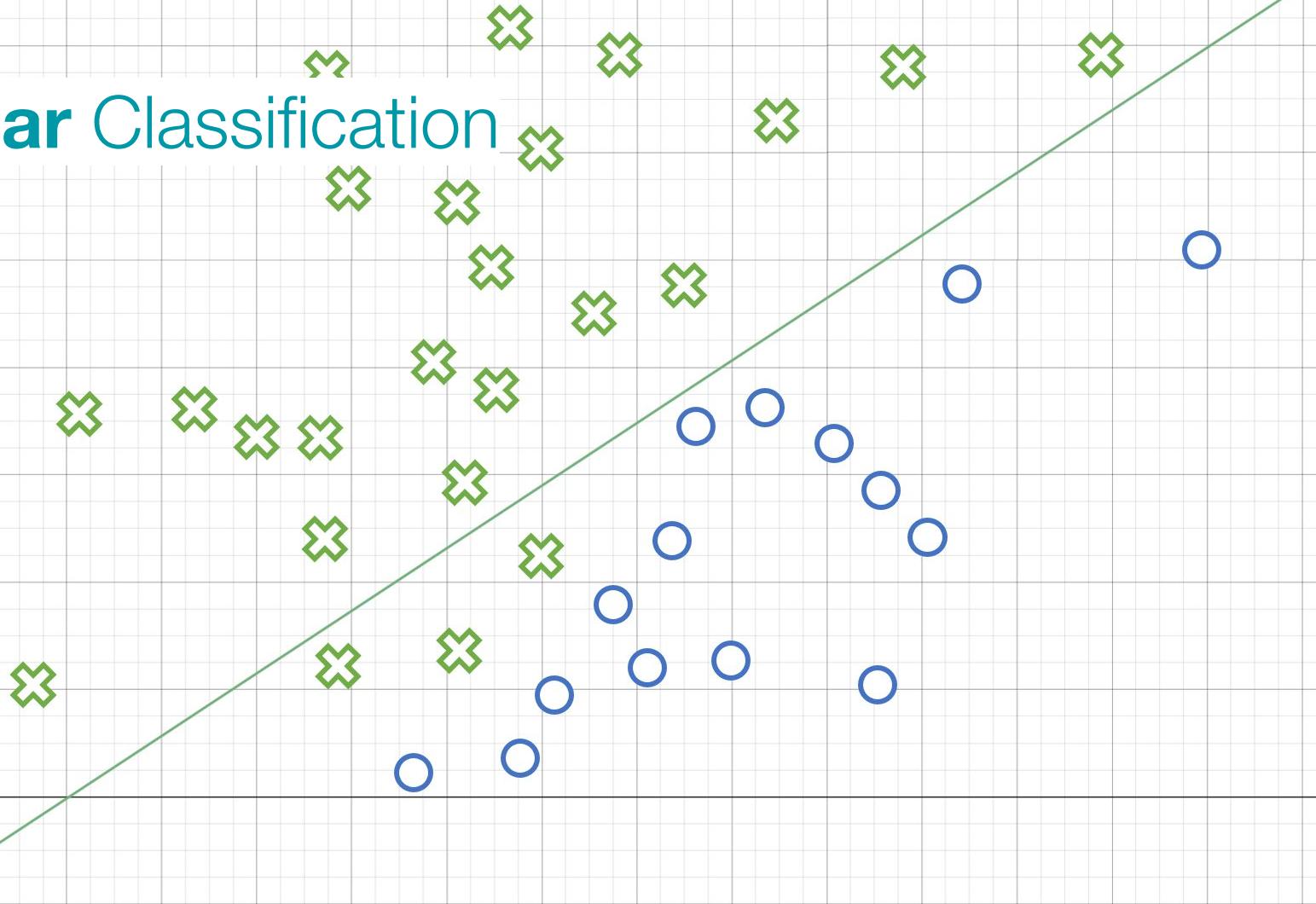
Traditional ML dont work either

# Neural Networks

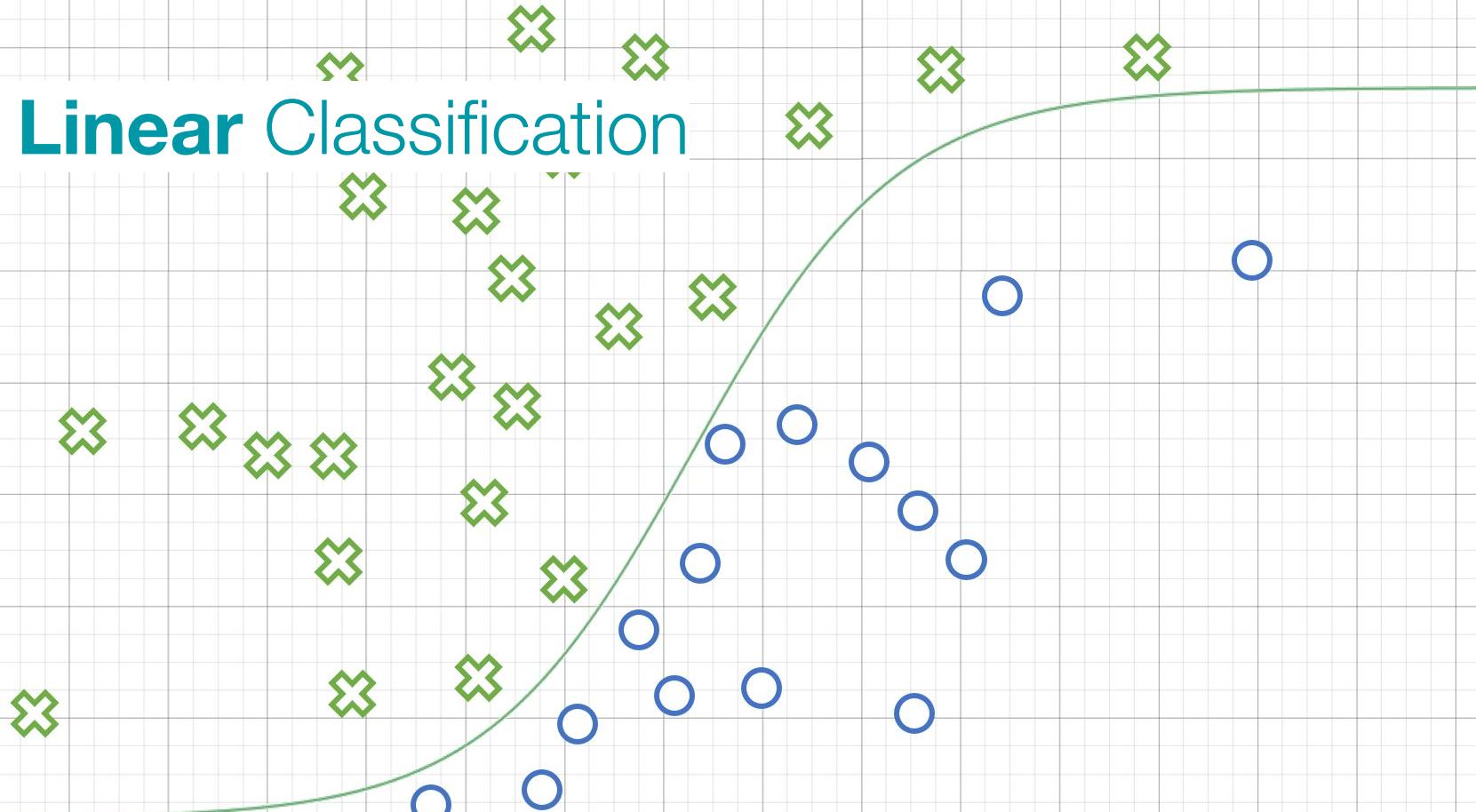
# Classification

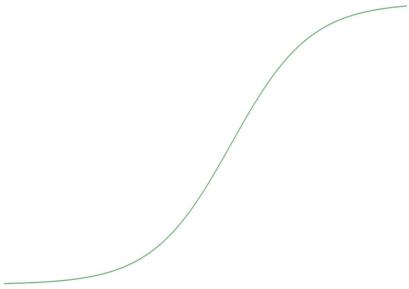


# Linear Classification



# Non Linear Classification





## Non linearity

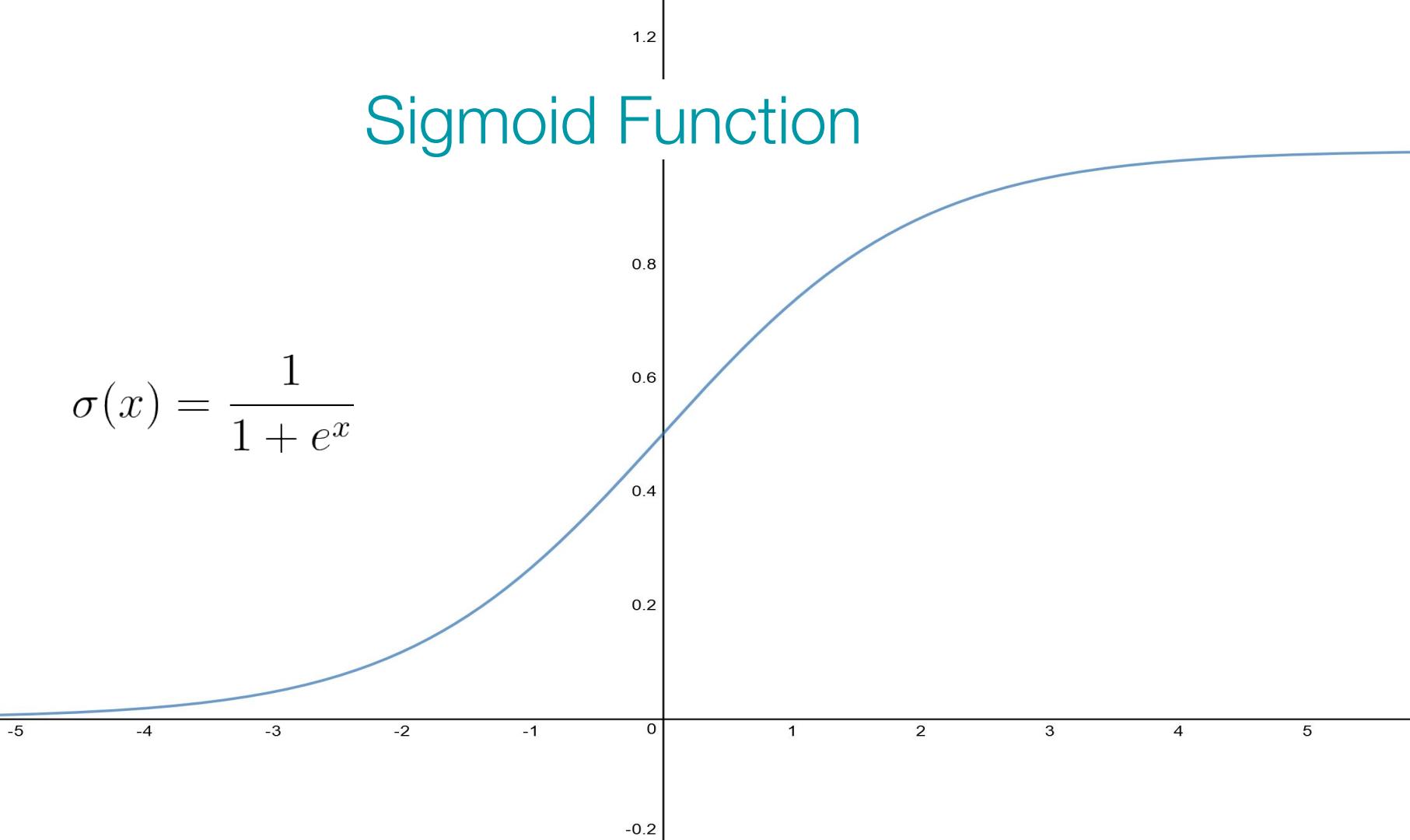
Can be stacked (composite functions) to increase expressivity.

Their gradients are smoother (derivatives of linear transformations are constant, regardless of the input.)

In any case, nonlinear functions are a generalization of linear functions.

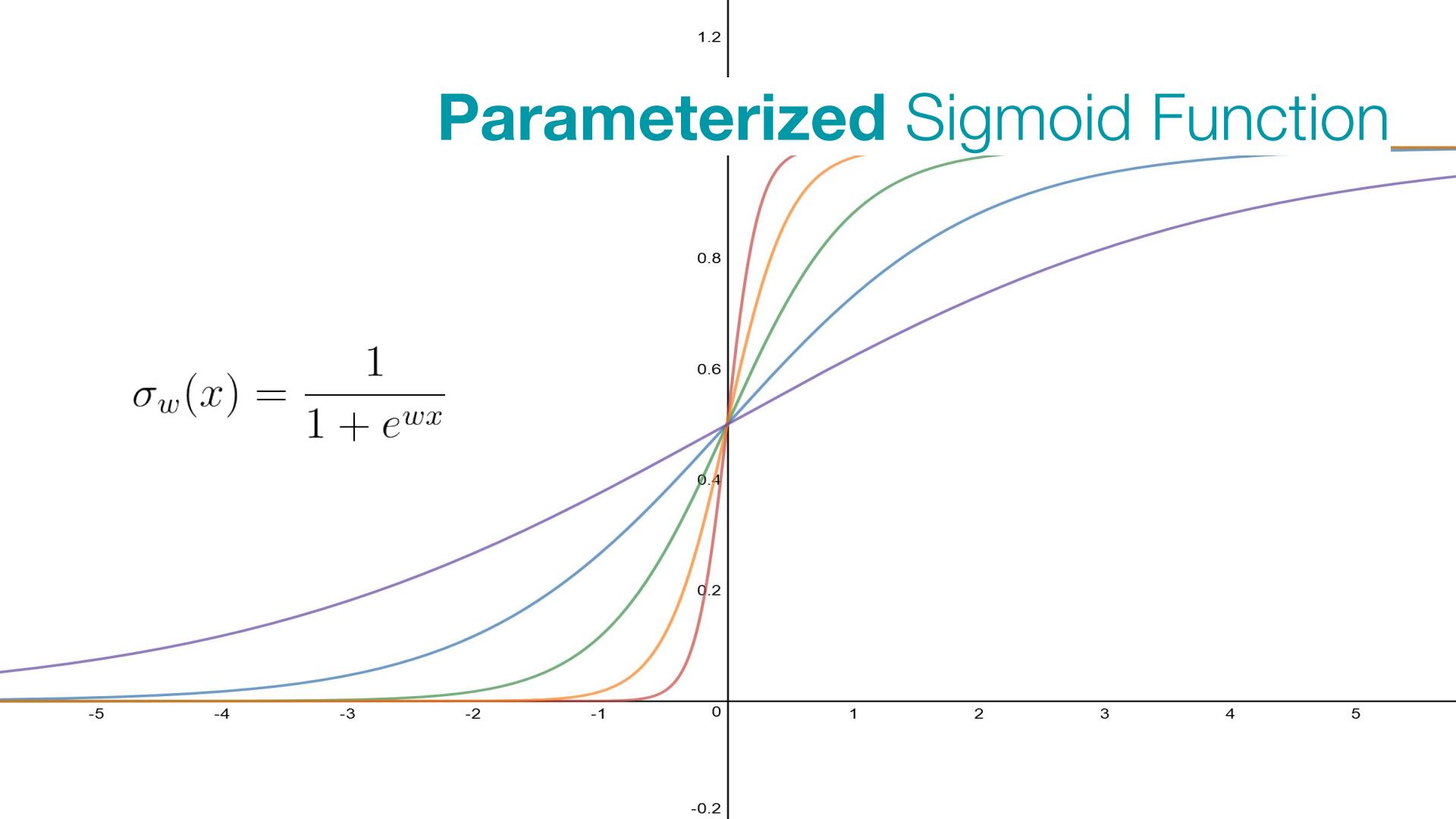
# Sigmoid Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



# Parameterized Sigmoid Function

$$\sigma_w(x) = \frac{1}{1 + e^{wx}}$$



# Parameterized Sigmoid Function

$$\sigma_{w,b}(x) = \frac{1}{1 + e^{wx+b}}$$

-5 -4 -3 -2 -1 0 1 2 3 4 5

1.2

0.8

0.6

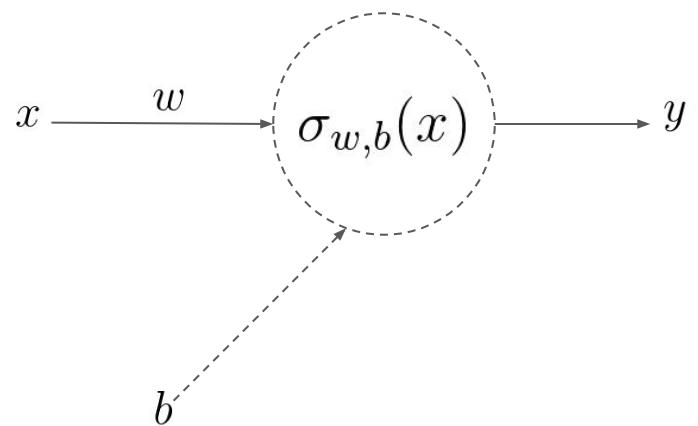
0.4

0.2

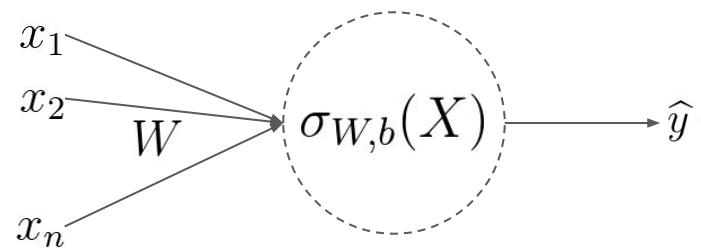
-0.2

0

# Neuron



# Vector Inputs to Neuron

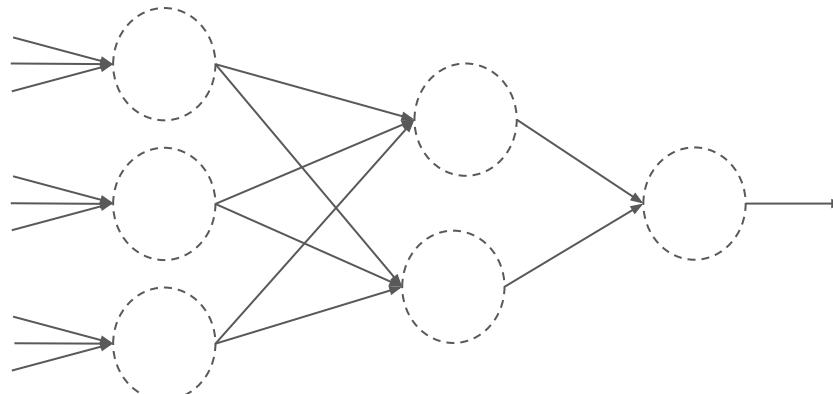


# Stacking (Composite Functions)

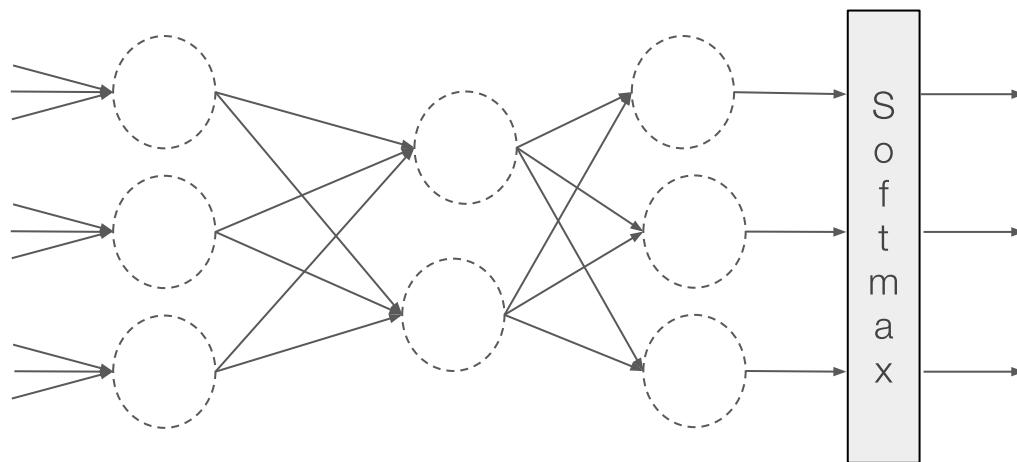


$$\sigma_{W_2, b_2}(\sigma_{W_1, b_1}(X))$$

# Feed Forward Network



# Multi-class Classification



# Activation Functions

Till now we have seen sigmoid function.

Can use several other functions such as

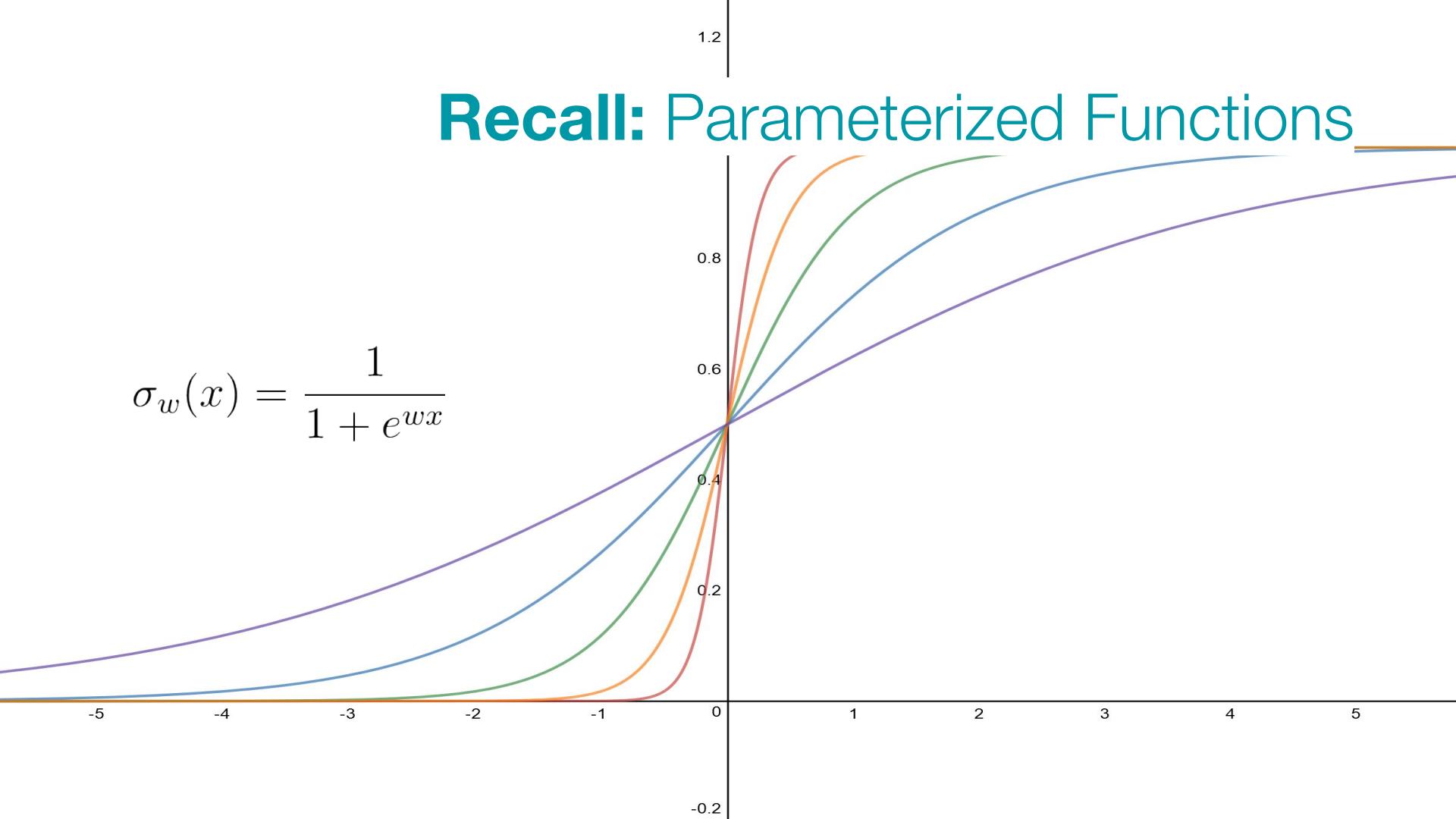
Hyperbolic Tangent

Rectifier

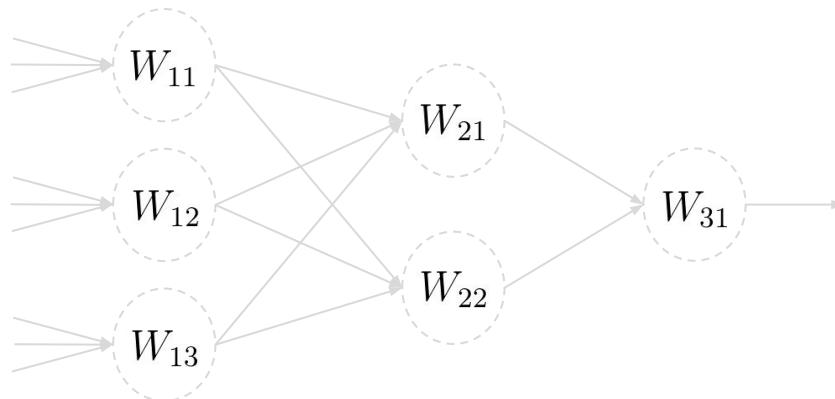
# Training Neural Networks

# Recall: Parameterized Functions

$$\sigma_w(x) = \frac{1}{1 + e^{wx}}$$



# Parameters



\* omitted biases for brevity's sake

# Needles in a Haystack

Can we set them manually?

# Weights (above):

No 

Recommended?

~~Kill a kitten till it works~~

# Needles in a Haystack

Can we set them manually?

# Weights (above):

No 

Recommended?

~~Kill a kitten till it works~~

Backpropagate

Do you even  
*backpropagate?*

# Idea

Find the effect of every parameter on model's performance.

For that we first need a way to calculate model performance:  
**Loss Function** (Error Function)

# Loss Function

Difference between *model output* and *given label* for a data point.

$$\hat{Y} = f_w(X) \qquad Y$$

$$E = L(\hat{Y}, Y)$$

Many ways to calculate loss. Eg. Mean Squared Error

$$L(\hat{Y}, Y) = \frac{1}{2n} \sum_{i=1}^n (\hat{y} - y)^2$$

# Gradient Descent

Gradients of Error w.r.t. model parameters: change the parameter in that direction to *increase* the error.

Update  $w_n = w_n - \eta \frac{dL}{dw_n}$

# Summing it up

- input data:  $x$

$x$

$y$

# Summing it up

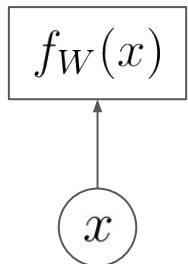
- input data:  $x$
- input data label:  $y$

$x$

$y$

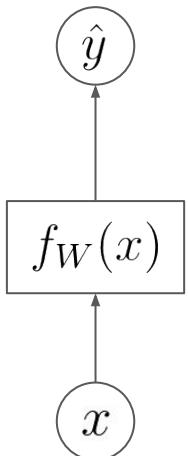
# Summing it up

- input data:  $x$
- input data label:  $y$
- model:  $f_W(x)$



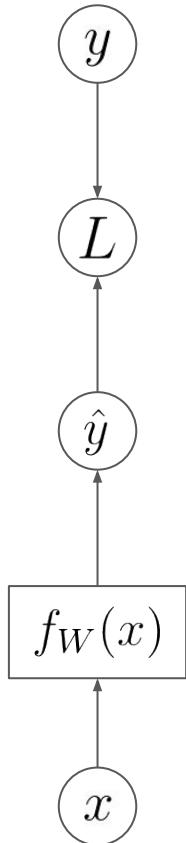
$y$

# Summing it up



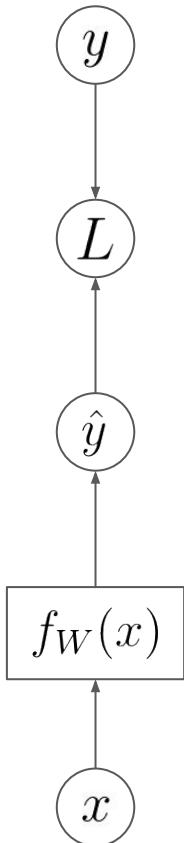
- input data:  $x$
- input data label:  $y$
- model:  $f_W(x)$
- model output:  $\hat{y} = f_W(x)$

# Summing it up



- input data:  $x$
- input data label:  $y$
- model:  $f_W(x)$
- model output:  $\hat{y} = f_W(x)$
- loss:  $L(\hat{y}, y)$

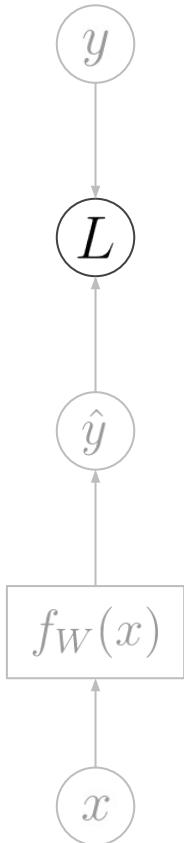
# Summing it up



- input data:  $x$
- input data label:  $y$
- model:  $f_W(x)$
- model output:  $\hat{y} = f_W(x)$
- loss:  $L(\hat{y}, y)$

} Forward Propagation

# Summing it up



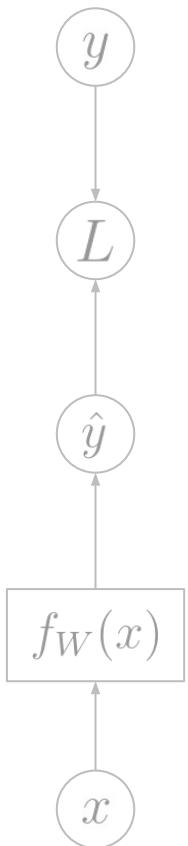
- input data:  $x$
- input data label:  $y$
- model:  $f_W(x)$
- model output:  $\hat{y} = f_W(x)$
- loss:  $L(\hat{y}, y)$

- gradients:  $\frac{dL}{dW_{1:n}}$

}

Forward Propagation

# Summing it up



- input data:  $x$
- input data label:  $y$
- model:  $f_W(x)$
- model output:  $\hat{y} = f_W(x)$
- loss:  $L(\hat{y}, y)$

- gradients:  $\frac{dL}{dW_{1:n}}$

- parameter update:  $w_n = w_n - \eta \frac{dL}{dW_{1:n}}$

}

Forward  
Propagation

}

Backward  
Propagation

# Training Neural Networks

**Supervised Learning:** Each example has an input vector, and a corresponding label.

We train the model to learn a mapping (function) from the input vector to the label.

Done by *optimizing parameters* using backpropagation.

# Neural Networks for **NLP**

# Representing Language

Words have inherent meanings.

Language is sequential in nature.

# Words have inherent meanings

## Desired Properties

- ✓ represent the semantic similarity of related words
- ✓ each word can be distinguished from others

Neural Networks expect vectors as input.

How do we convert text to vector?

# One-hot Vectors

Word tokens are assigned a unique ID (represented by a one-hot vector)

## Desired Properties

- ✓ represent the semantic similarity of related words
- ✓ each word can be distinguished from others

name	=	[ 1, 0, 0, 0, 0 ..., 0]
all	=	[ 0, 0, 0, 1, 0 ..., 0]
movies	=	[ 0, 1, 0, 0, 0 ..., 0]
where	=	[ 0, 0, 1, 0, 0 ..., 0]

# One-hot Vectors

Word tokens are assigned a unique ID (represented by a one-hot vector)

## Desired Properties

- ✓ represent the semantic similarity of related words
- ✓ each word can be distinguished from others

name	=	[ 1, 0, 0, 0, 0 ..., 0]
all	=	[ 0, 0, 0, 1, 0 ..., 0]
movies	=	[ 0, 1, 0, 0, 0 ..., 0]
where	=	[ 0, 0, 1, 0, 0 ..., 0]

# One-hot Vectors **Limitations**

## Desired Properties

- ✓ represent the semantic similarity of related words
- ✓ each word can be distinguished from others

No useful information to the system regarding the relationships that may exist between the words.

One-hot vectors are sparse in nature.

## Distributional Hypothesis

You shall know a word by the  
company it keeps

J. R. Firth,  
1957: 11

government debt problems turning into banking crises as has happened in  
saying that Europe needs unified banking regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

# Word Embeddings

Predictive model:

Given a word, what is the probable context.

## Desired Properties

✓ represent the semantic similarity of related words

or

Given a context, what is the probable word.

✓ ~~each word can be distinguished from others~~

# Word Embeddings

Predictive model:

Given a word, what is the probable context.

$$p(w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2} | w_t)$$

## Desired Properties

- ✓ represent the semantic similarity of related words
- ✓ ~~each word can be distinguished from others~~

or

Given a context, what is the probable word.

$$p(w_t | w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2})$$

# Word Embeddings

Dense Vector Space:

## Desired Properties

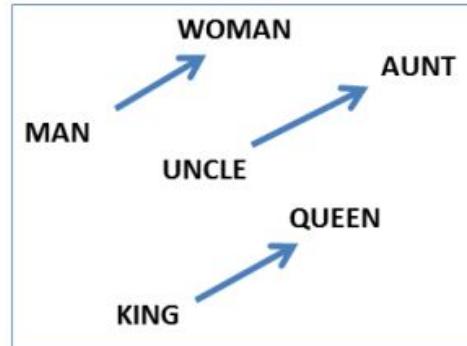
- ✓ represent the semantic similarity of related words
- ✓ ~~each word can be distinguished from others~~

Force a fixed number of dimensions in the vector space.

Forces similar words to appear together (across some dimensions)

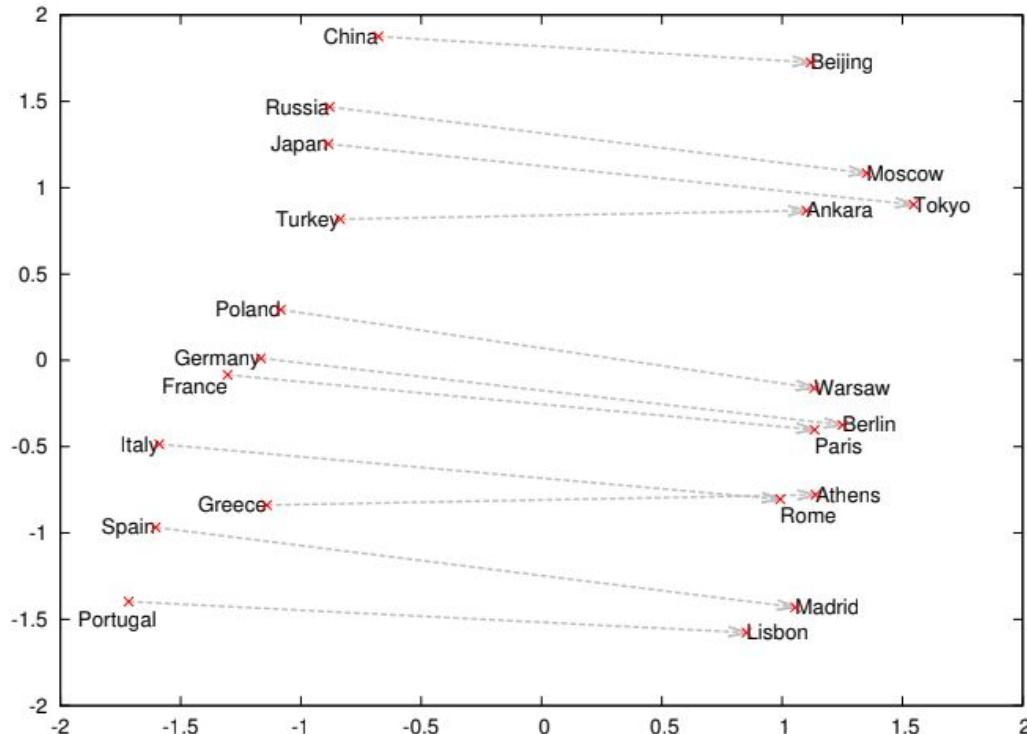
# Word2Vec

“encodes semantic components as linear vector differences”

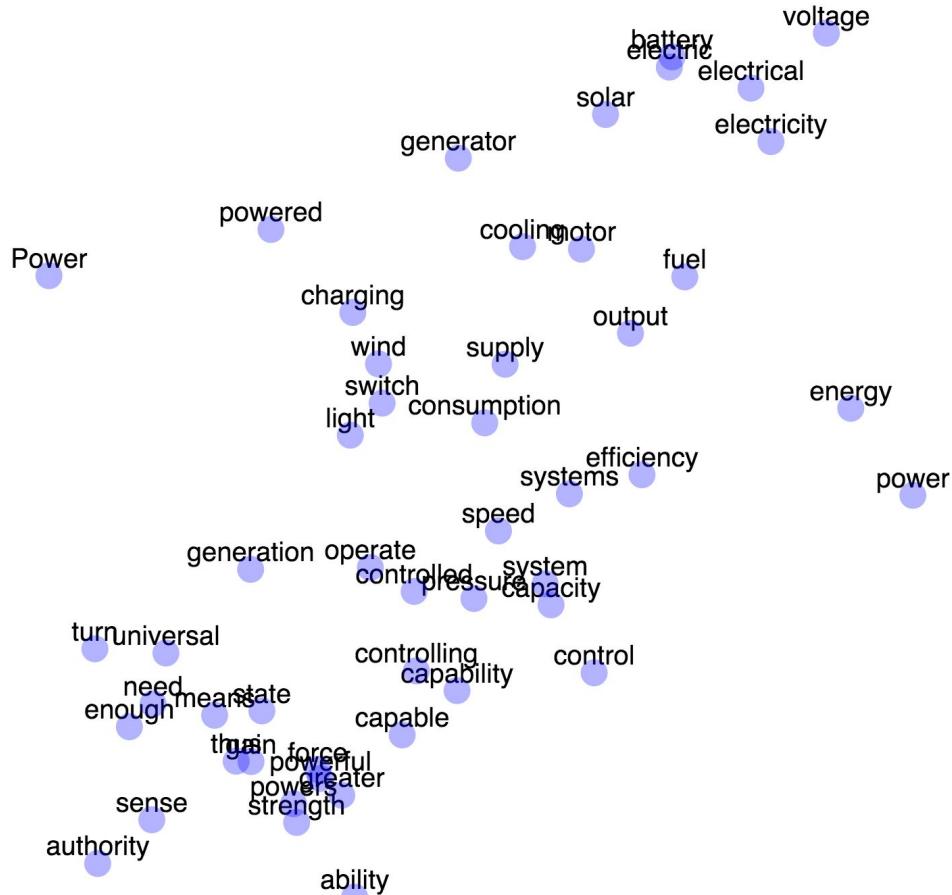


Meaningful  
syntactic and  
semantic  
regularities  
captured

# Word2Vec



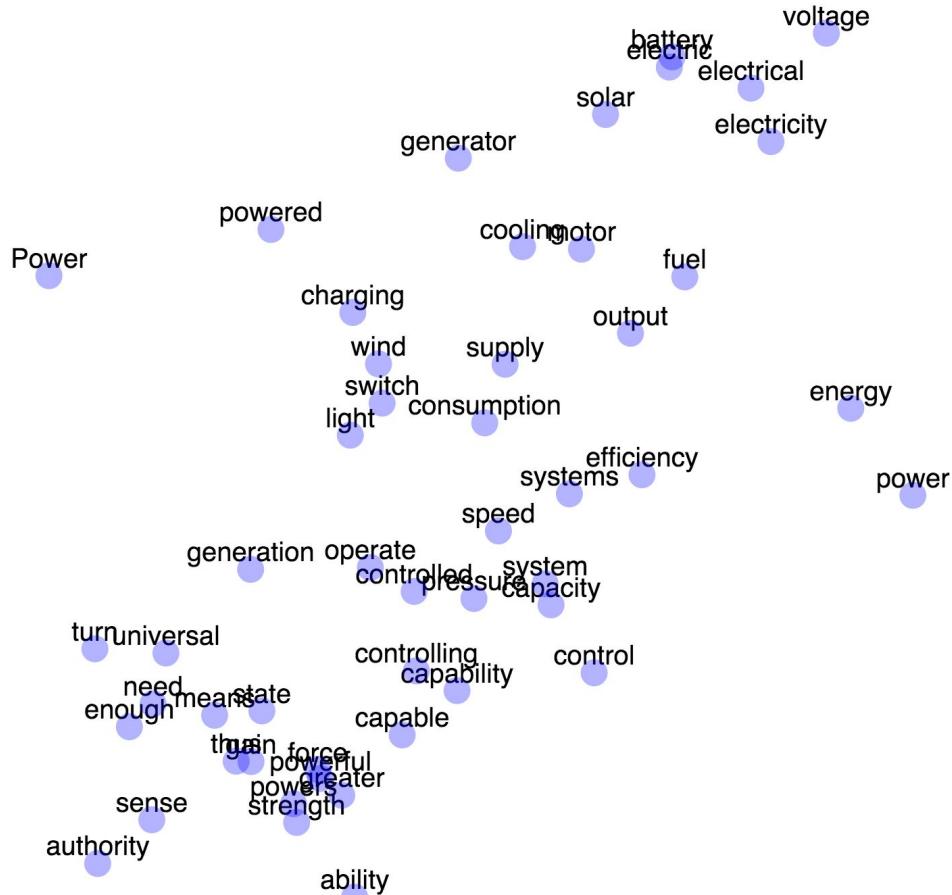
Meaningful  
syntactic and  
semantic  
regularities  
captured



Meaningful  
syntactic and  
semantic  
regularities  
captured

## Desired Properties

- ✓ ~~represent the semantic similarity of related words~~
- ✓ ~~each word can be distinguished from others~~



# Word Embeddings

Widely used for most NLP tasks.

Numerous methods of training:

- word2vec
- GloVe
- fastText embeddings

Pre-trained, off the shelf solutions.

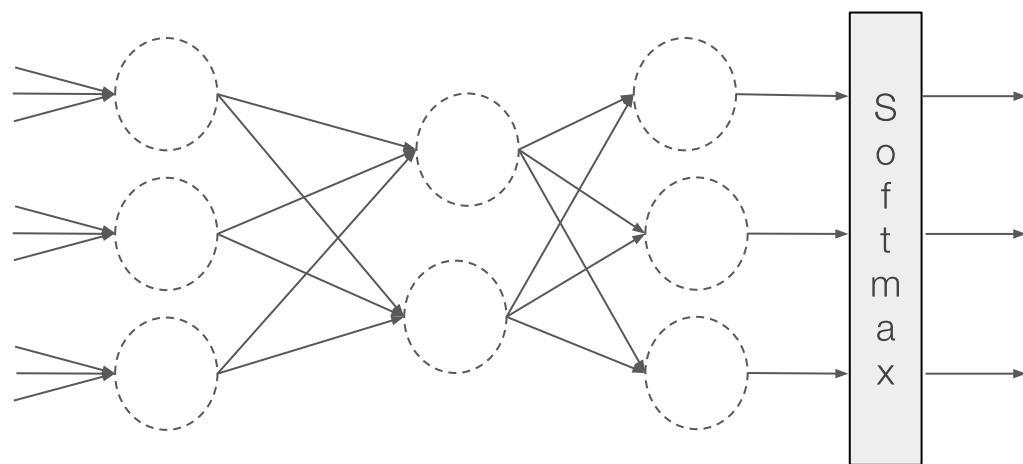
# Word Embeddings are matrices

$$\begin{bmatrix} 0.21 & 0.42 & 0.16 & 0.86 & 0.67 \\ 0.86 & 0.90 & 0.20 & 0.38 & 0.19 \\ 0.49 & 0.24 & 0.87 & 0.95 & 0.88 \\ 0.80 & 0.64 & 0.19 & 0.32 & 0.38 \\ 0.71 & 0.54 & 0.66 & 0.61 & 0.40 \\ 0.39 & 0.17 & 0.19 & 0.54 & 0.41 \\ 0.17 & 0.30 & 0.32 & 0.96 & 0.76 \\ 0.05 & 0.32 & 0.80 & 0.92 & 0.81 \end{bmatrix}$$

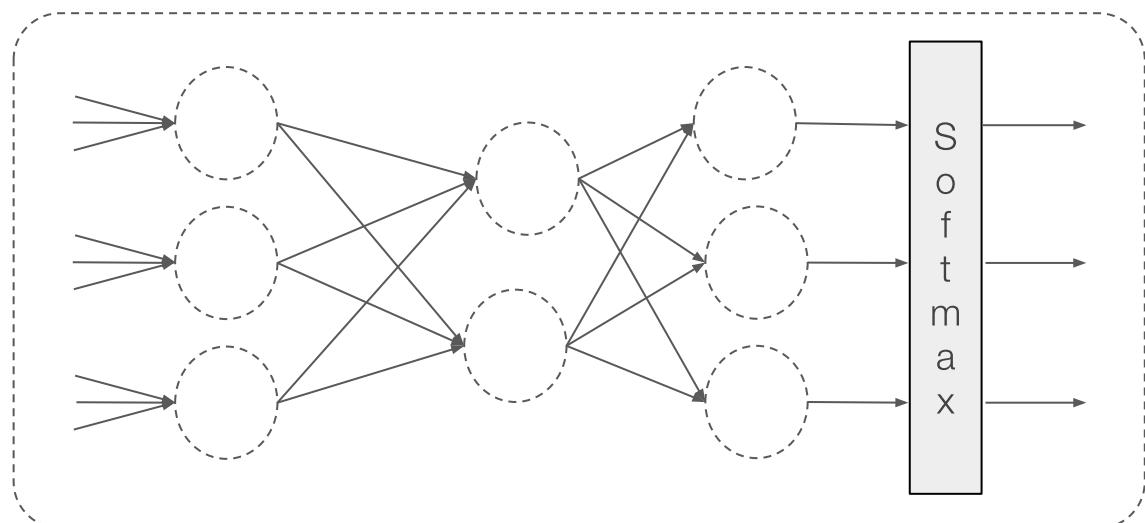
# Word Embeddings are matrices

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} 0.21 & 0.42 & 0.16 & 0.86 & 0.67 \\ 0.86 & 0.90 & 0.20 & 0.38 & 0.19 \\ 0.49 & 0.24 & 0.87 & 0.95 & 0.88 \\ 0.80 & 0.64 & 0.19 & 0.32 & 0.38 \\ 0.71 & 0.54 & 0.66 & 0.61 & 0.40 \\ 0.39 & 0.17 & 0.19 & 0.54 & 0.41 \\ 0.17 & 0.30 & 0.32 & 0.96 & 0.76 \\ 0.05 & 0.32 & 0.80 & 0.92 & 0.81 \end{bmatrix} = \begin{bmatrix} 0.71 & 0.54 & 0.66 & 0.61 & 0.40 \end{bmatrix}$$

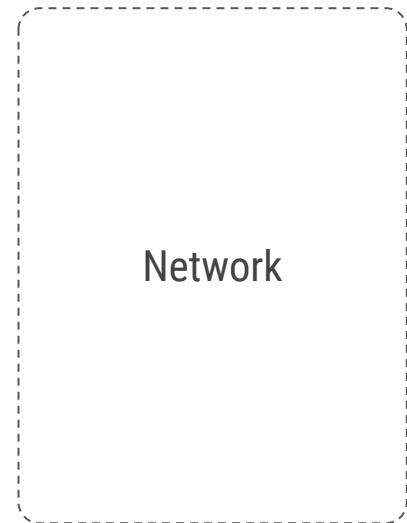
# Using Word Embeddings in NN



# Using Word Embeddings in NN



# Using Word Embeddings in NN



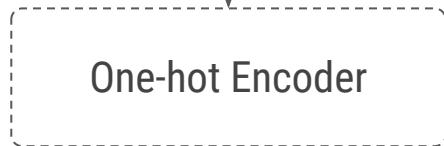
# Using Word Embeddings in NN

$\dots w_{t-1}, w_t, w_{t+1}, w_{t+2} \dots$

Network

# Using Word Embeddings in NN

$\dots w_{t-1}, w_t, w_{t+1}, w_{t+2} \dots$

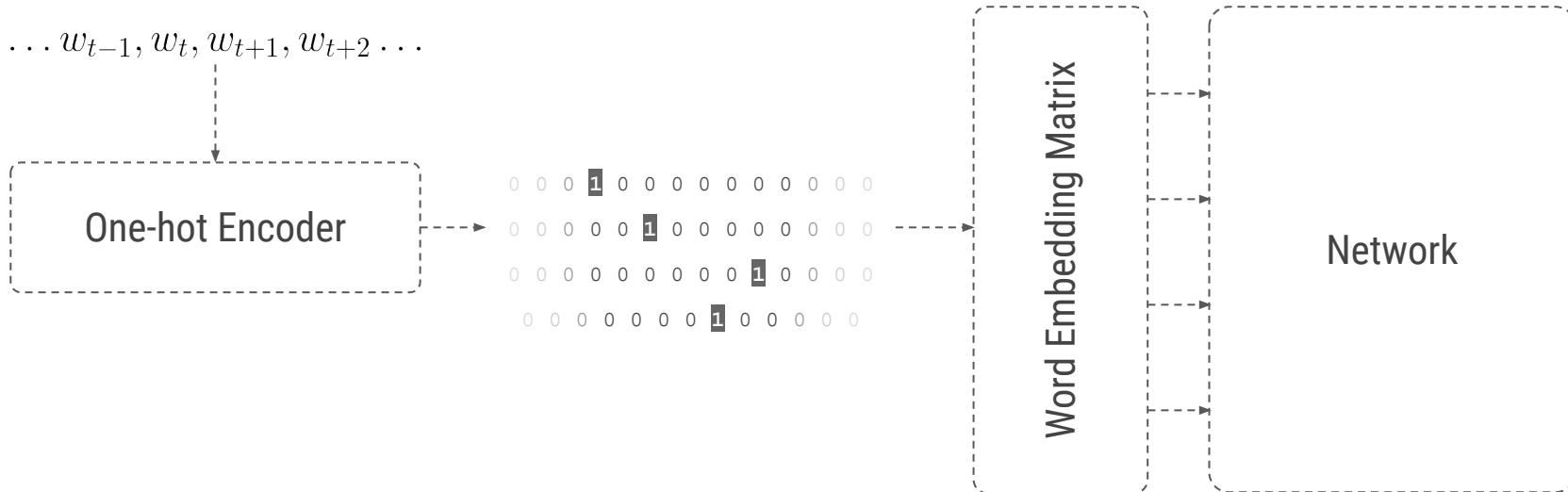


```
0 0 0 1 0 0 0 0 0 0 0 0 0  
0 0 0 0 0 1 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0 0 1 0 0 0  
0 0 0 0 0 0 0 0 1 0 0 0 0
```

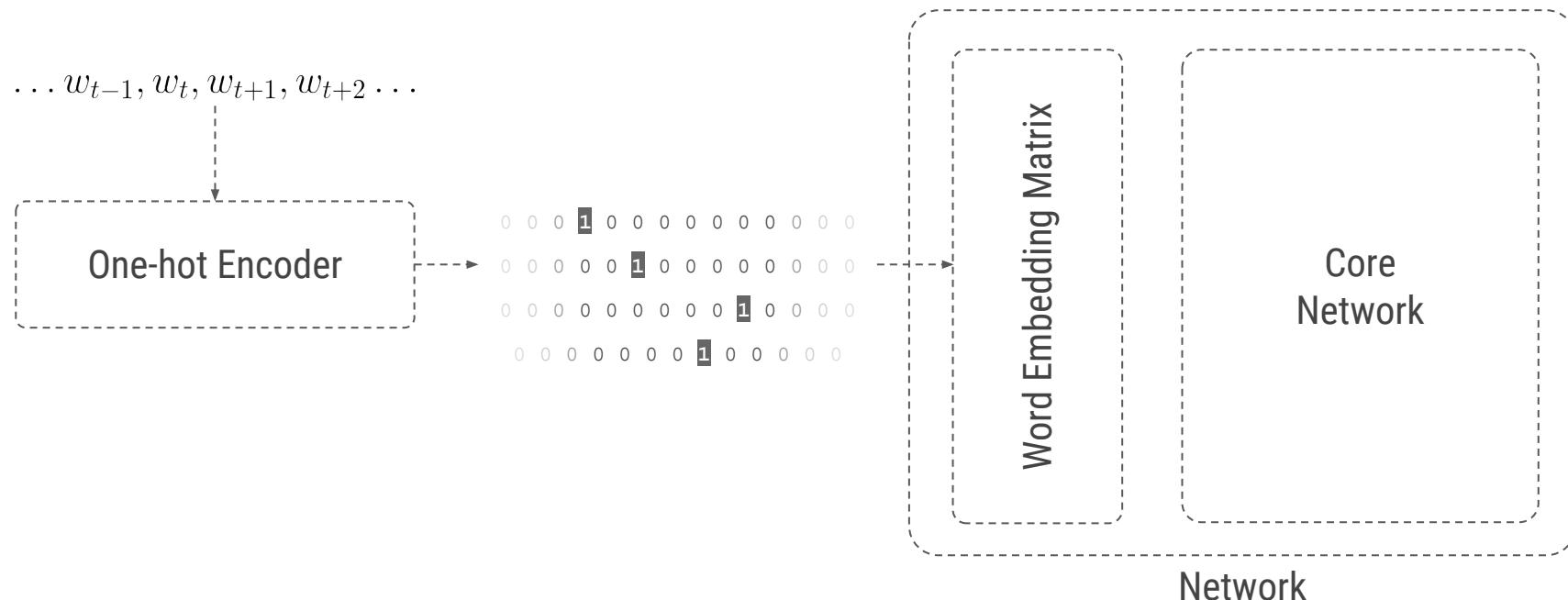
Network

# Using Word Embeddings in NN

$\dots w_{t-1}, w_t, w_{t+1}, w_{t+2} \dots$



# Trainable Embeddings



# Representing Language

Words have inherent meanings.

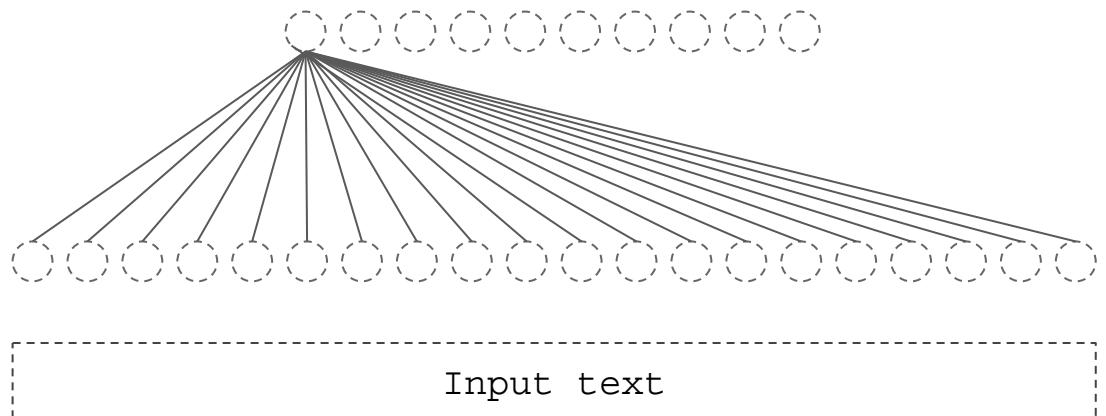


Language is sequential in nature.

# Problems with Feedforward Nets

Number of parameters:

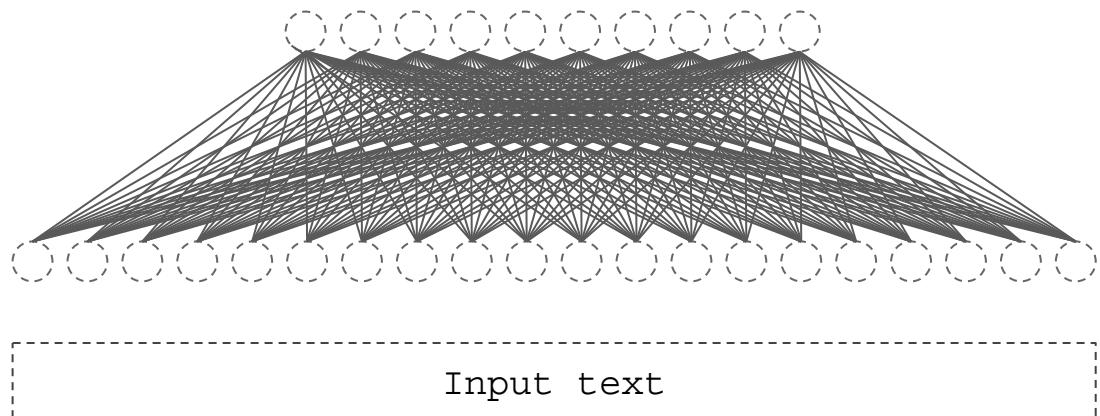
$$20 \times 10$$



# Problems with Feedforward Nets

Number of parameters:

$$20 \times 10$$



# Problems with Feedforward Nets

Number of parameters:

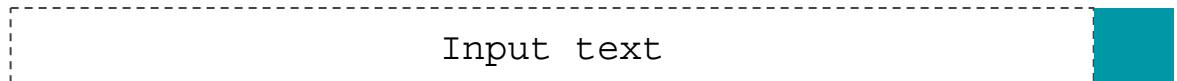
$$20 \times 10$$

Unseen length of text

oooooooooooo

×

oooooooooooooooooooo



# Problems with Feedforward Nets

Number of parameters:

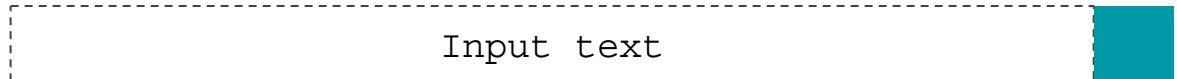
$$20 \times 10$$

Unseen length of text

oooooooooooo

×

oo●oooooooooooo●ooo



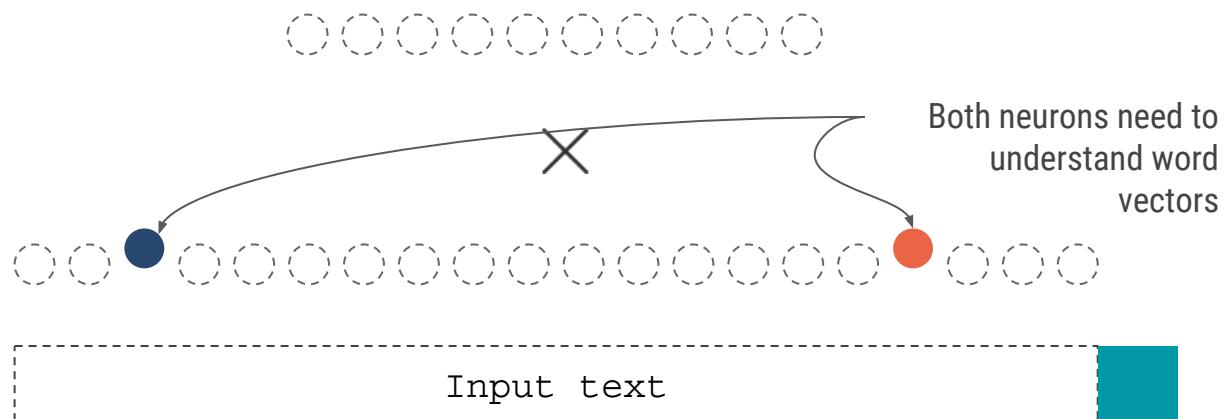
# Problems with Feedforward Nets

Number of parameters:

$$20 \times 10$$

Unseen length of text

Independent weights  
learning the same thing.



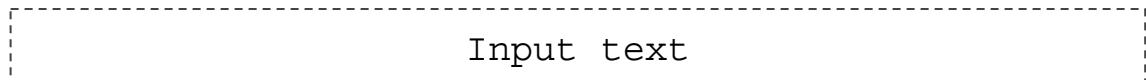
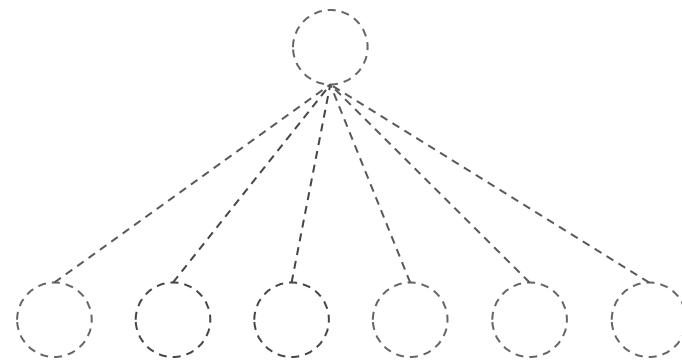
# Problems with Feedforward Nets

Number of parameters:

$$20 \times 10$$

Unseen length of text

Independent weights  
learning the same thing.



# Problems with Feedforward Nets

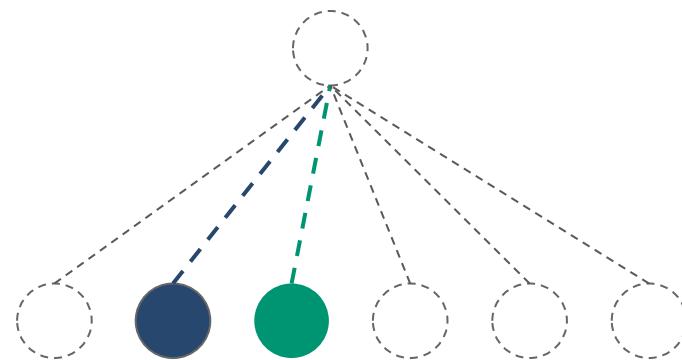
Number of parameters:

$$20 \times 10$$

Unseen length of text

Independent weights  
learning the same thing.

Language is sequential.



A:      United States **is** **in** North America

B:      United States **isn't** **in** North America

# Problems with Feedforward Nets

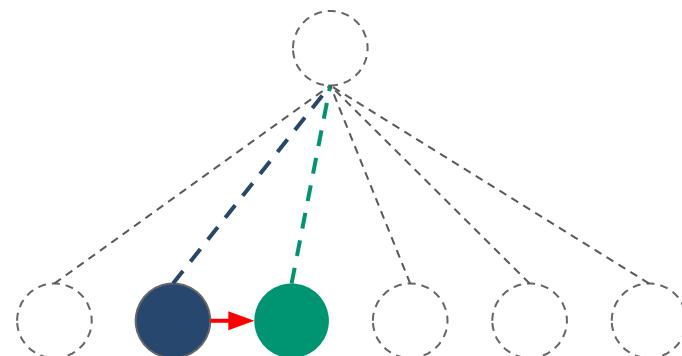
Number of parameters:

$$20 \times 10$$

Unseen length of text

Independent weights  
learning the same thing.

Language is sequential.



A:      United States **is** **in** North America

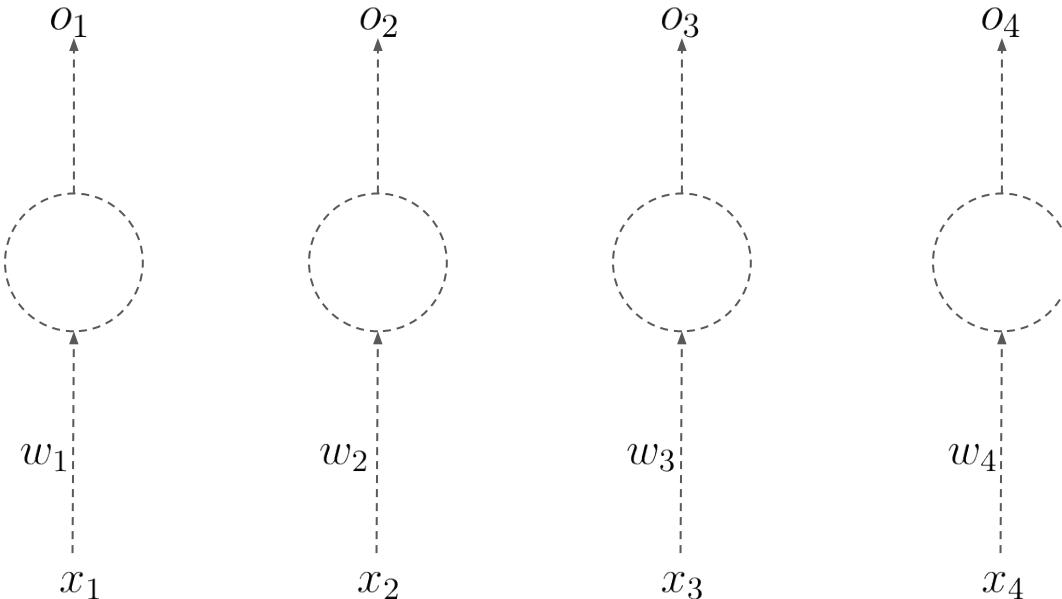
B:      United States **isn't** **in** North America

Language is Sequential

Cue **Sequential**  
Networks

# Sequential Networks?

Standard FF layer

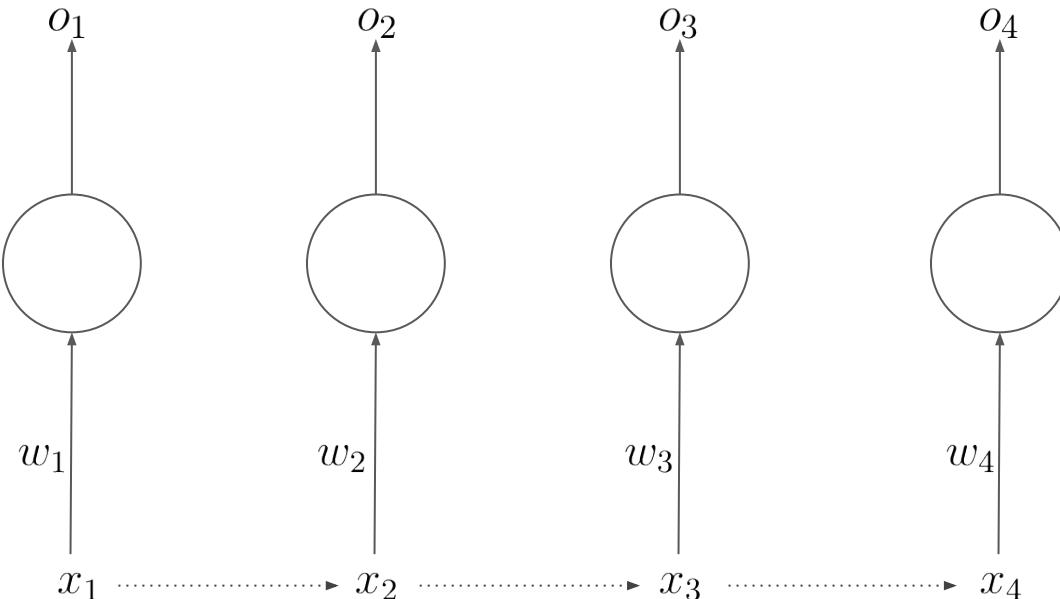


# Sequential Networks?

Standard FF layer

But inputs might depend upon each other.

$$p(x_{t-1}, x_t) \neq p(x_{t-1}) \times p(x_t)$$



# Sequential Networks

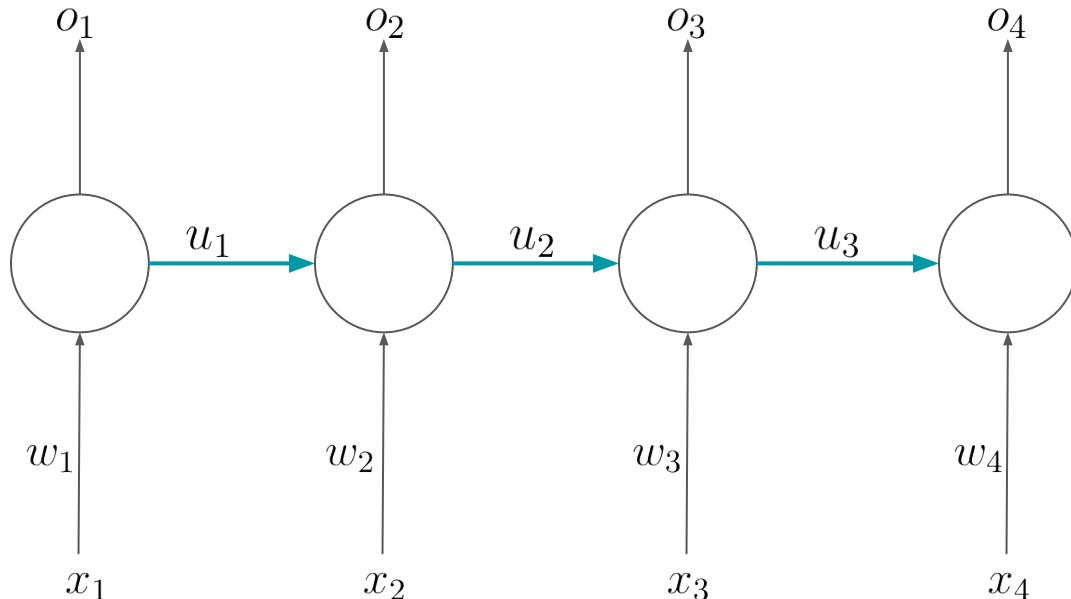
Standard FF layer

But inputs might depend upon each other.

$$p(x_{t-1}, x_t) \neq p(x_{t-1}) \times p(x_t)$$

Neuron in **one layer** are now

- **Time dependent**
- Affect the next one



# Sequential Networks

Standard FF layer

But inputs might depend upon each other.

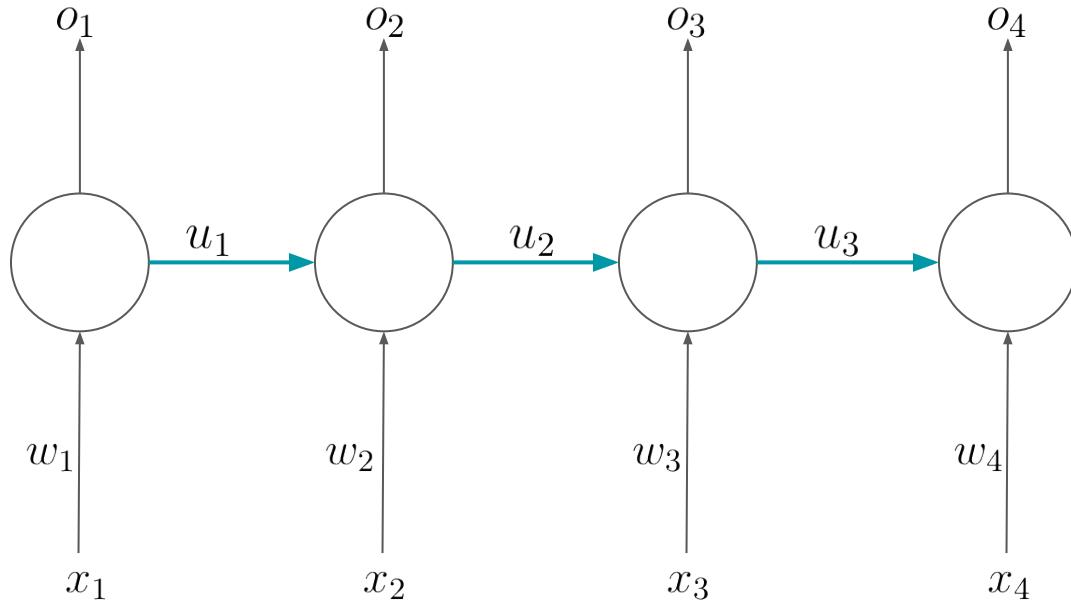
$$p(x_{t-1}, x_t) \neq p(x_{t-1}) \times p(x_t)$$

Neuron in **one layer** are now

- **Time dependent**
- Affect the next one

More parameters!

Every neuron must learn language individually, as well as take into account prev. input?



# Sequential Networks

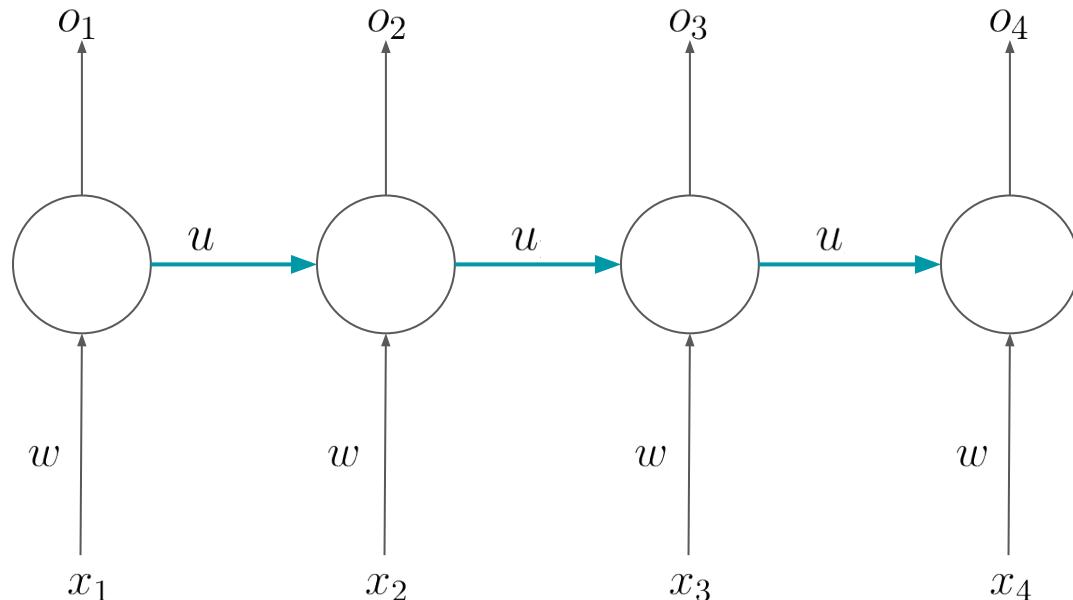
Standard FF layer

But inputs might depend upon each other.

$$p(x_{t-1}, x_t) \neq p(x_{t-1}) \times p(x_t)$$

Neuron in **one layer** are now

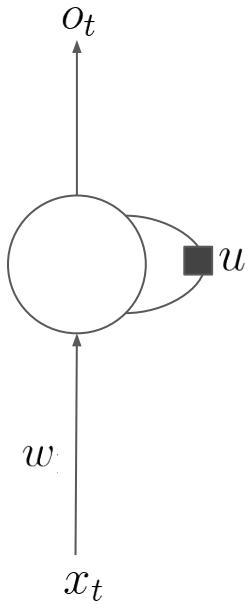
- Time dependent
- Affect the next one
- **Share Weights**



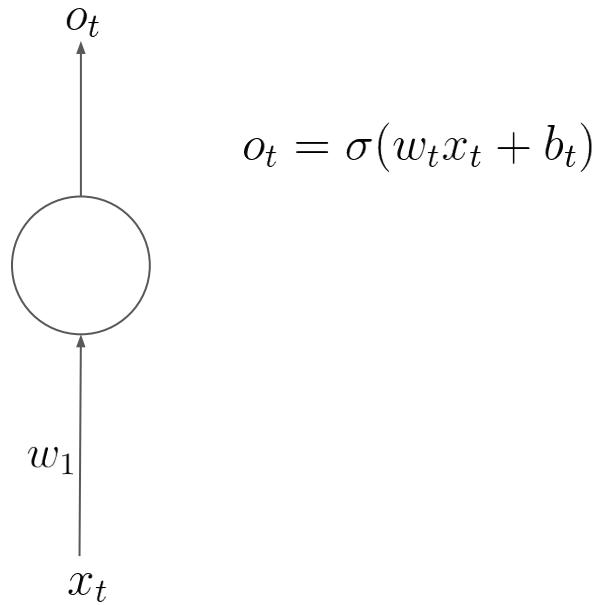
# Unfolded Visualization

Neuron in **one layer** are now

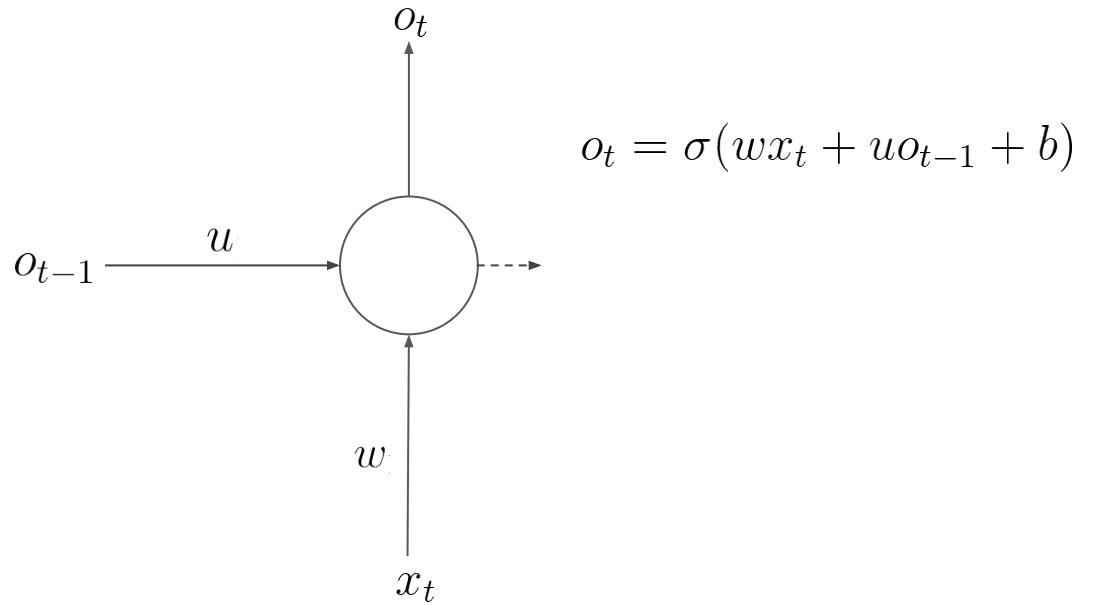
- Time dependent
- Affect the next one
- Share weights



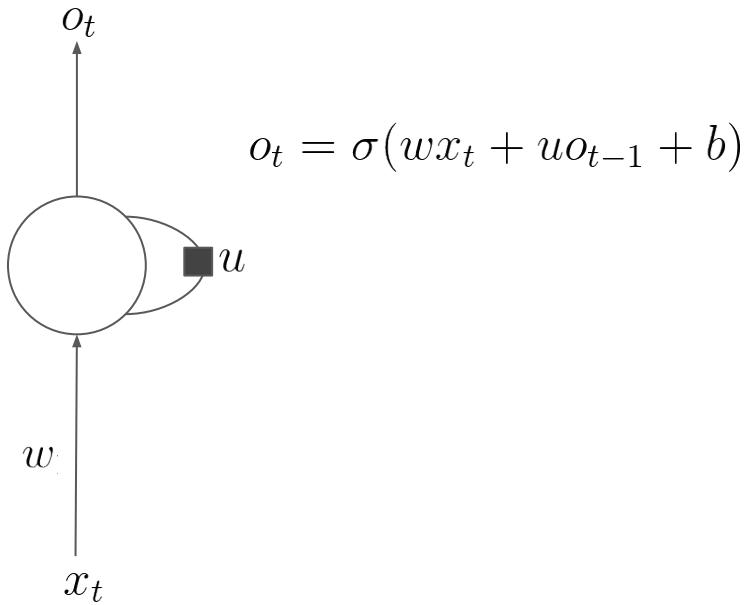
# Non-Recurrent Neuron



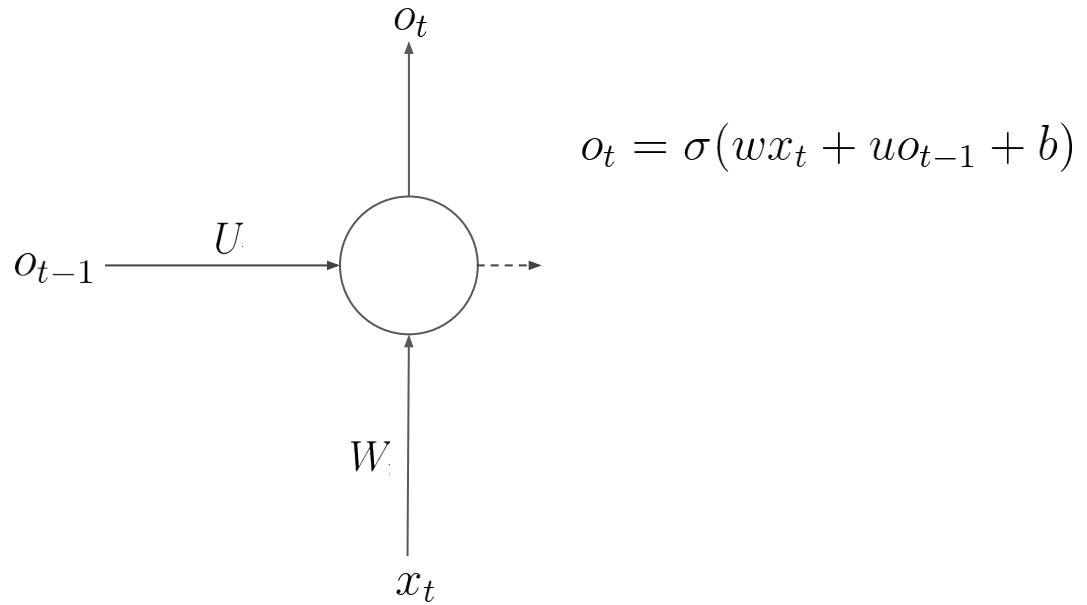
# Recurrent Neuron



# Recurrent Neuron



# Recurrent Neuron

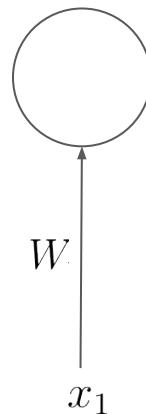


# Unfolding Example

$x_1$

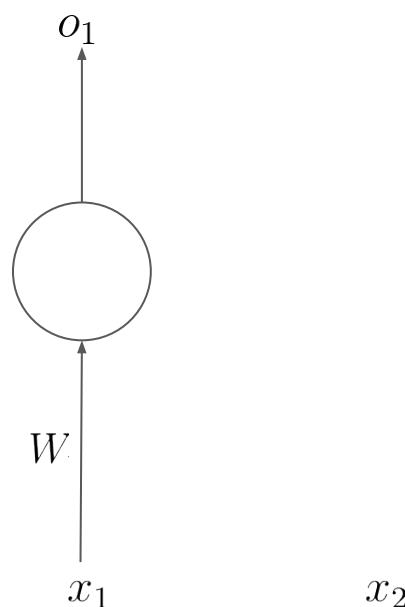
# Unfolding Example

$$o_1 = \sigma(Wx_1 + b)$$



# Unfolding Example

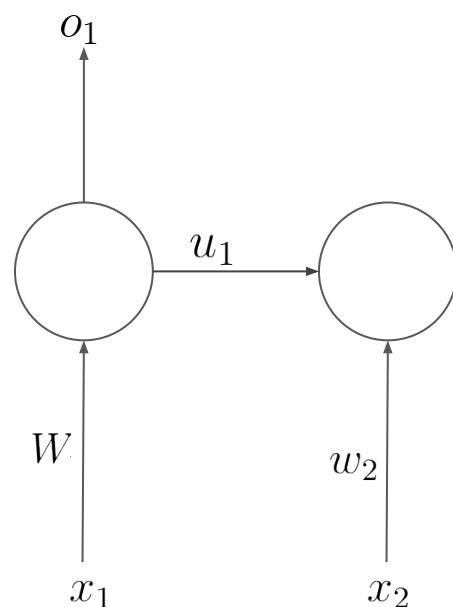
$$o_1 = \sigma(Wx_1 + b)$$



# Unfolding Example

$$o_1 = \sigma(Wx_1 + b)$$

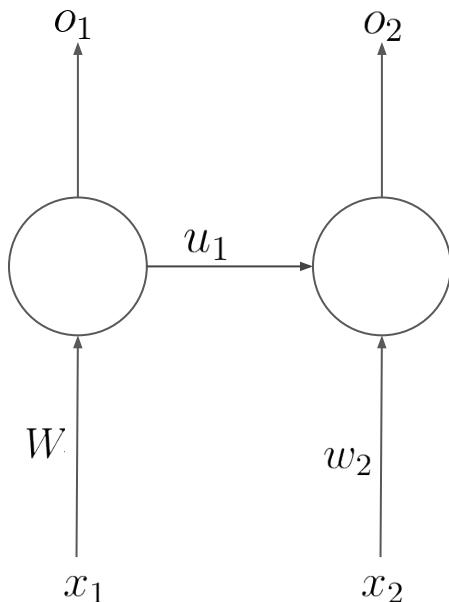
$$Wx_2 + Uo_1 + b$$



# Unfolding Example

$$o_1 = \sigma(Wx_1 + b)$$

$$o_2 = \sigma(Wx_2 + Uo_1 + b)$$

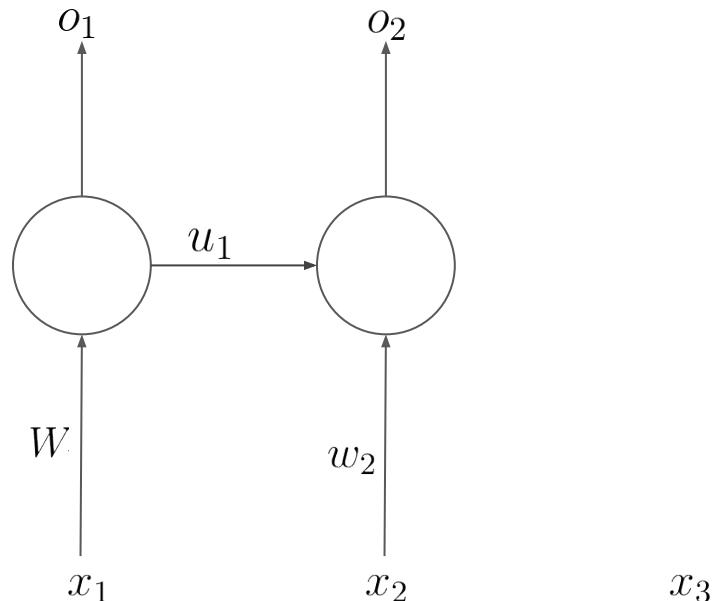


$o_t$  is a lossy summary of input so far. Lossy due to fixed-size vector encoding information about varying sized input. Depending upon training, can learn to discard irrelevant parts.

# Unfolding Example

$$o_1 = \sigma(Wx_1 + b)$$

$$o_2 = \sigma(Wx_2 + Uo_1 + b)$$

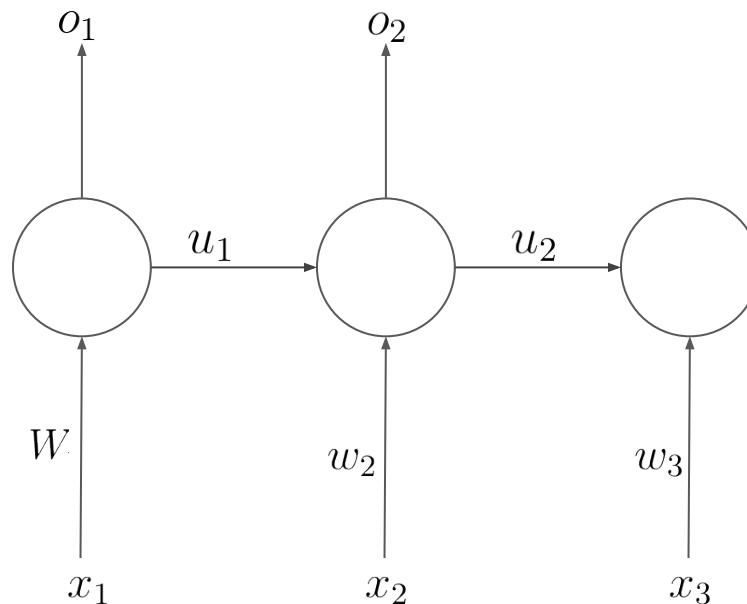


# Unfolding Example

$$o_1 = \sigma(Wx_1 + b)$$

$$o_2 = \sigma(Wx_2 + Uo_1 + b)$$

$$Wx_3 + Uo_2 + b$$



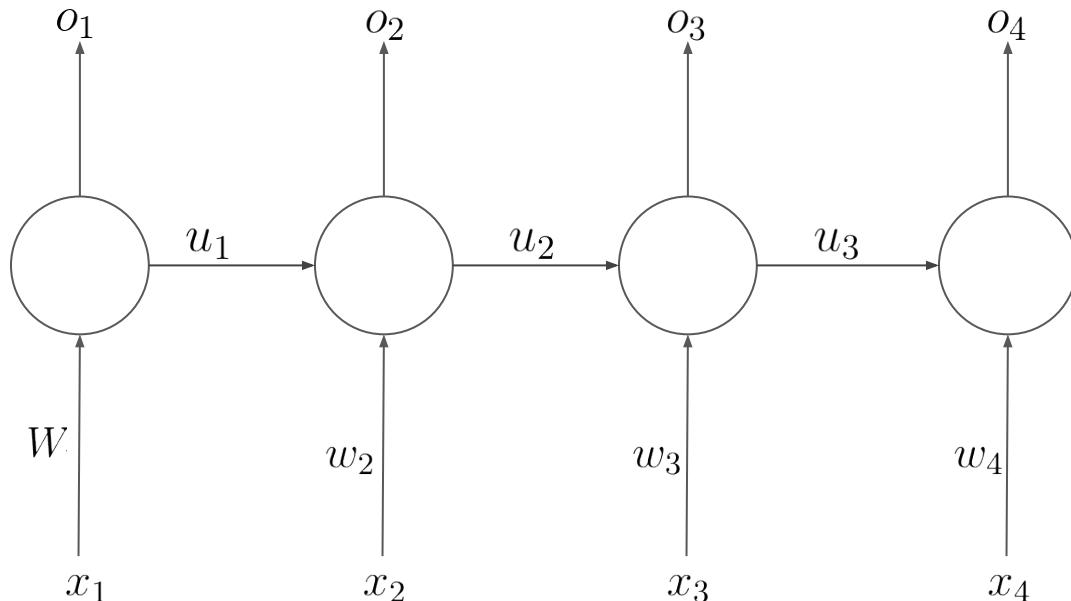
# Unfolding Example

$$o_1 = \sigma(Wx_1 + b)$$

$$o_2 = \sigma(Wx_2 + Uo_1 + b)$$

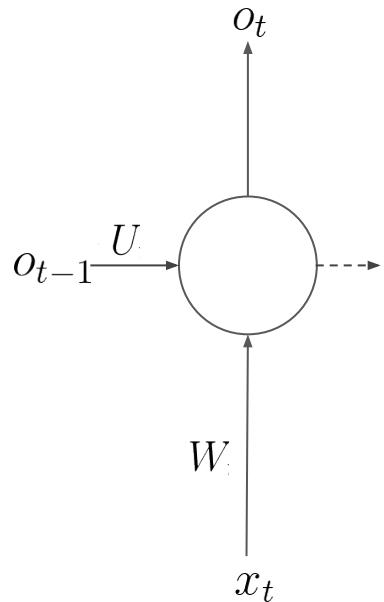
$$o_3 = \sigma(Wx_3 + Uo_2 + b)$$

$$o_4 = \sigma(Wx_4 + Uo_3 + b)$$



# Advantages

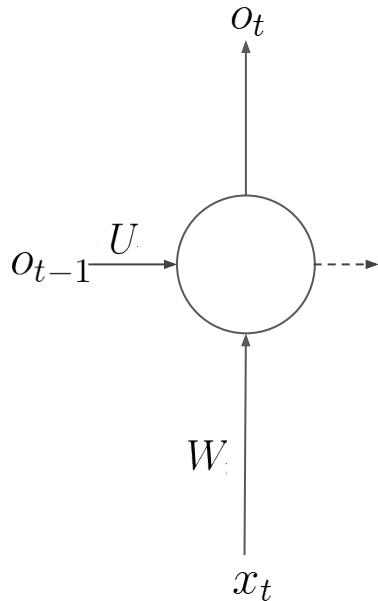
$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



- Specialized for processing sequential input.

# Advantages

$$o_t = \sigma(wx_t + uo_{t-1} + b)$$

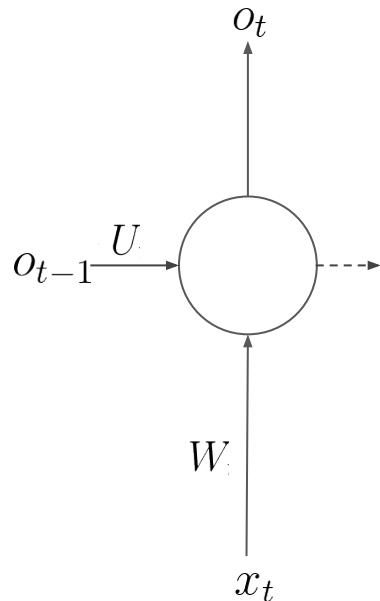


- Specialized for processing sequential input.

**is** ready      Can differentiate b/w two cases  
**isn't** ready      while processing “ready”

# Advantages

$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



- Specialized for processing sequential input.

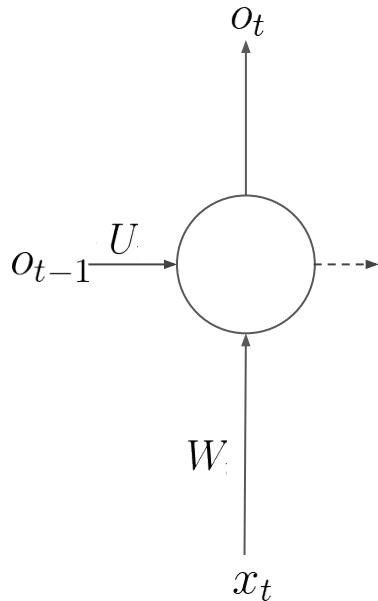
**is** ready      Can differentiate b/w two cases  
**isn't** ready      while processing “ready”

Can't we use n-grams?

- effect limited to only  $d=\frac{n}{2}$  words
- parameters would increase linearly with  $n$
- RNN can (in theory) have effects across entire sequence.

# Advantages

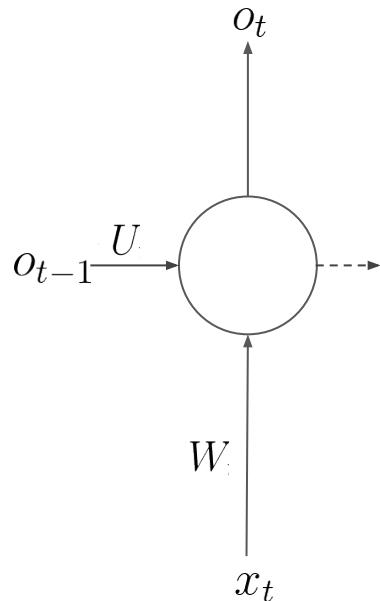
$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



- Specialized for processing sequential input.
- Can generalize for sequences of unseen length

# Advantages

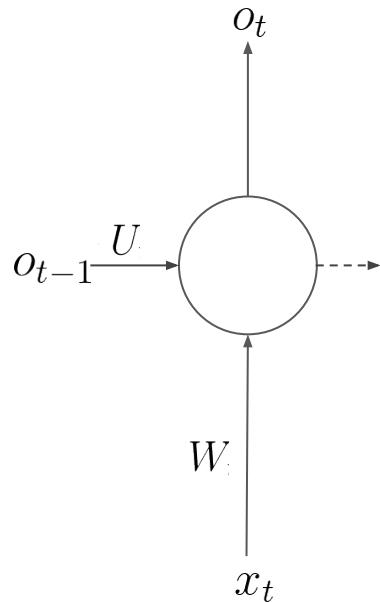
$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



- Specialized for processing sequential input.
- Can generalize for sequences of unseen length
  - Shared parameters -> same update rule. Can unroll ad-infinitum.
  - Parameters learn to parse language. Can parse language ad-infinitum
- Although in practice, effect of a word diminishes exponentially with time (has workarounds e.g. LSTM/GRU/Attention)

# Advantages

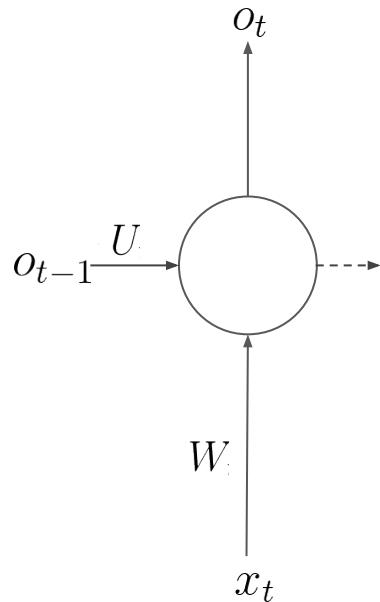
$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



- Specialized for processing sequential input.
- Can generalize for sequences of unseen length
- Reduced parameters

# Advantages

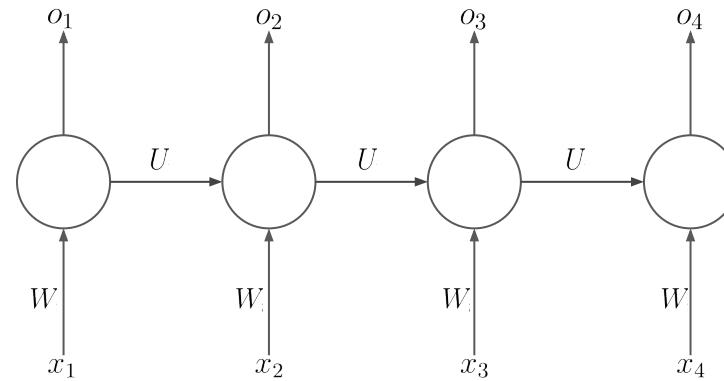
$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



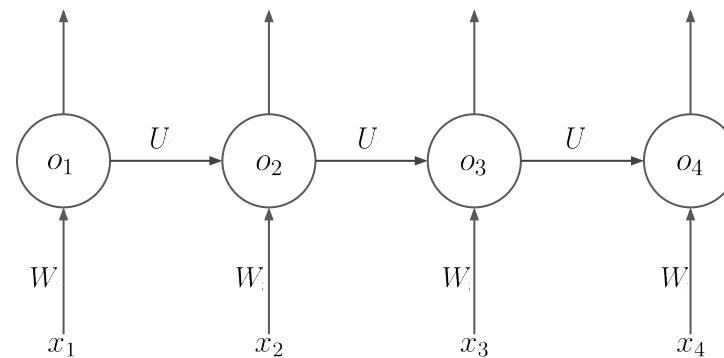
- Specialized for processing sequential input.
- Can generalize for sequences of unseen length
- Reduced parameters
  - Need lesser examples.
  - Less prone to overfit.

# Recurrent Networks in the Wild

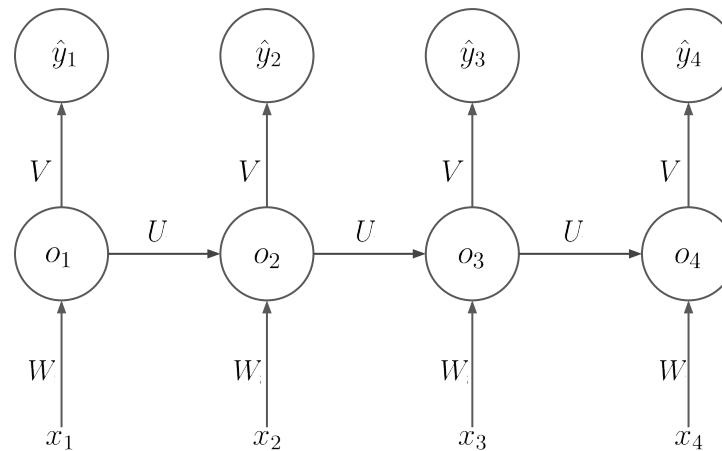
# The complete Picture



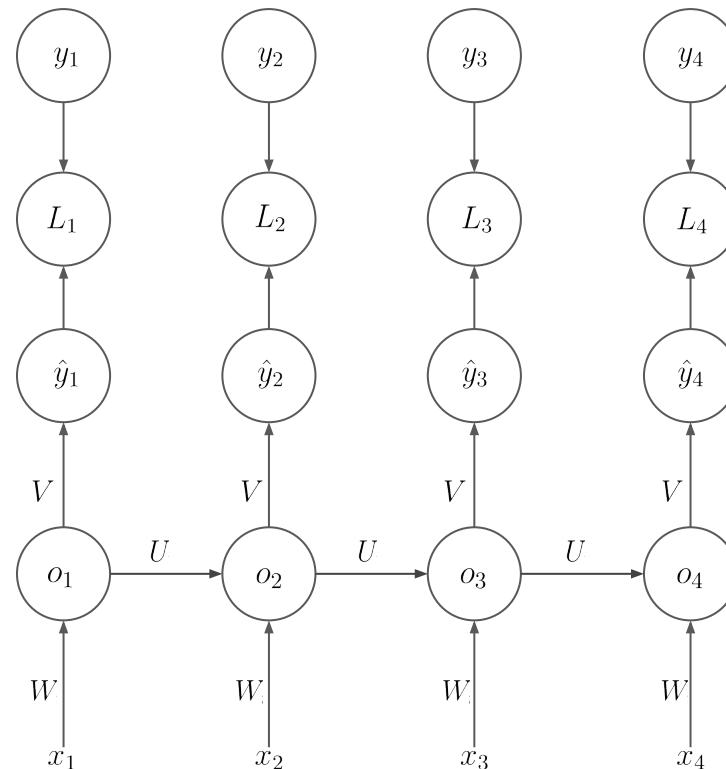
# The complete Picture



# The complete Picture

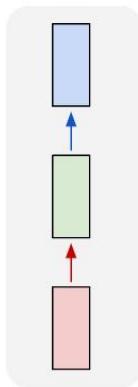


# The complete Picture

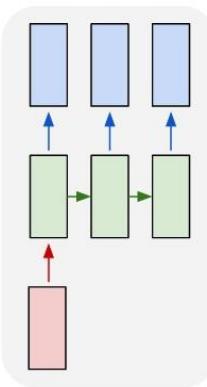


# Alternatives

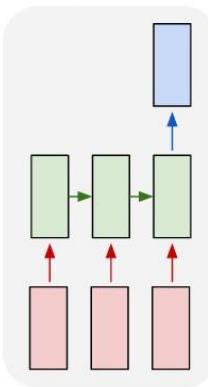
one to one



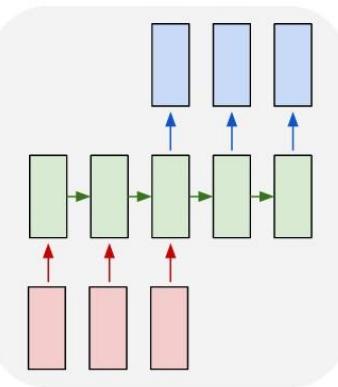
one to many



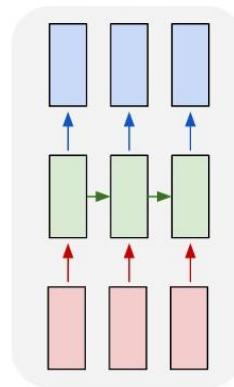
many to one



many to many



many to many



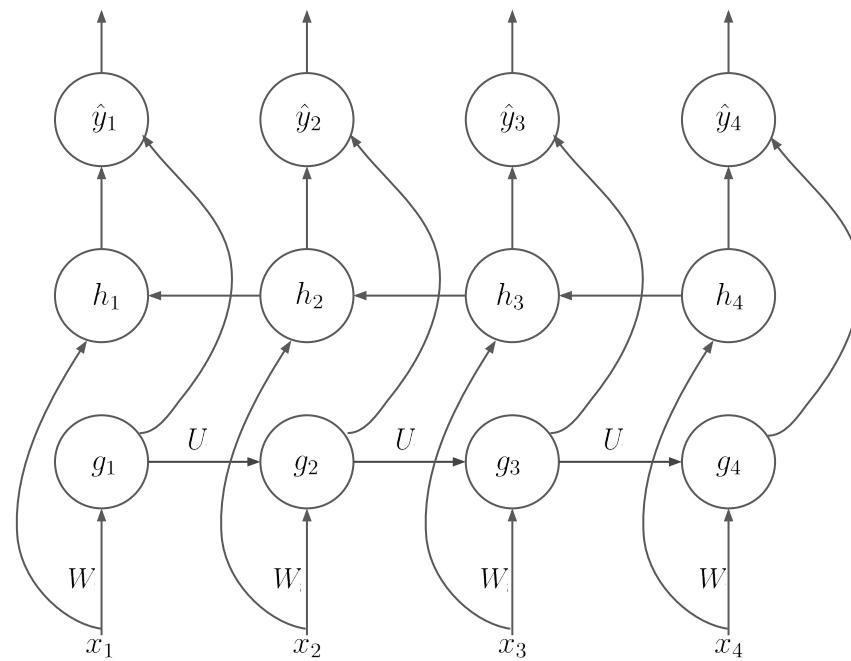
# Limitations

# Bidirectional RNN

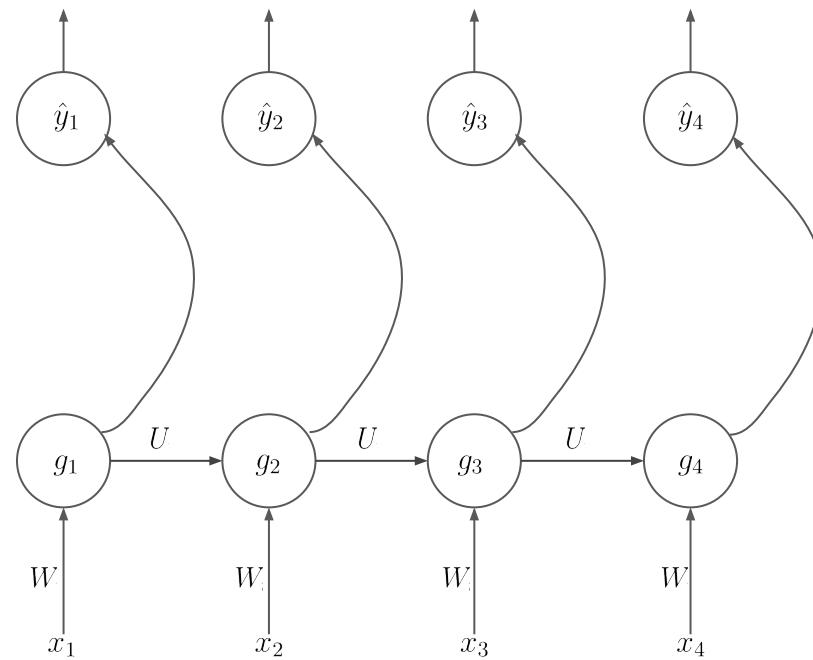
In some applications the information about *next word* might be useful for the current one.

Influence of start of sequence might diminish towards the end.

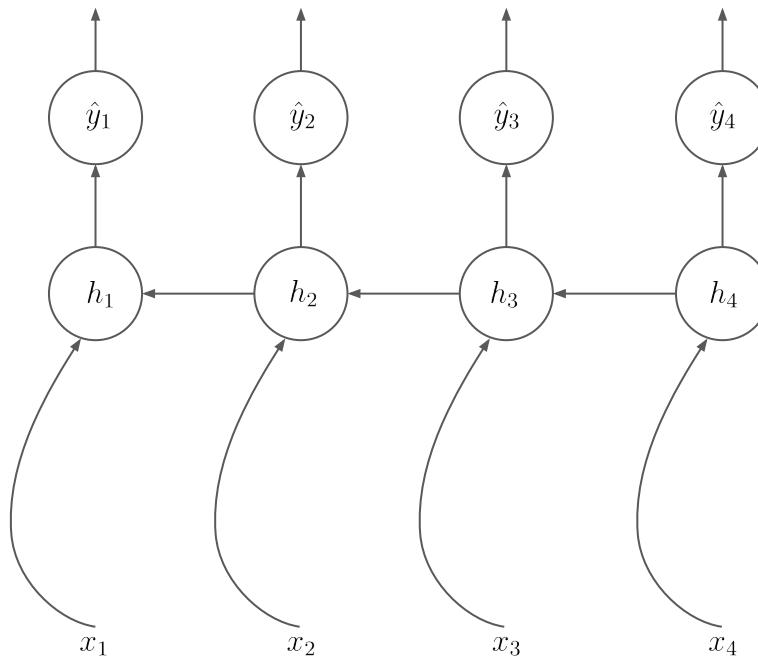
# Bidirectional Network



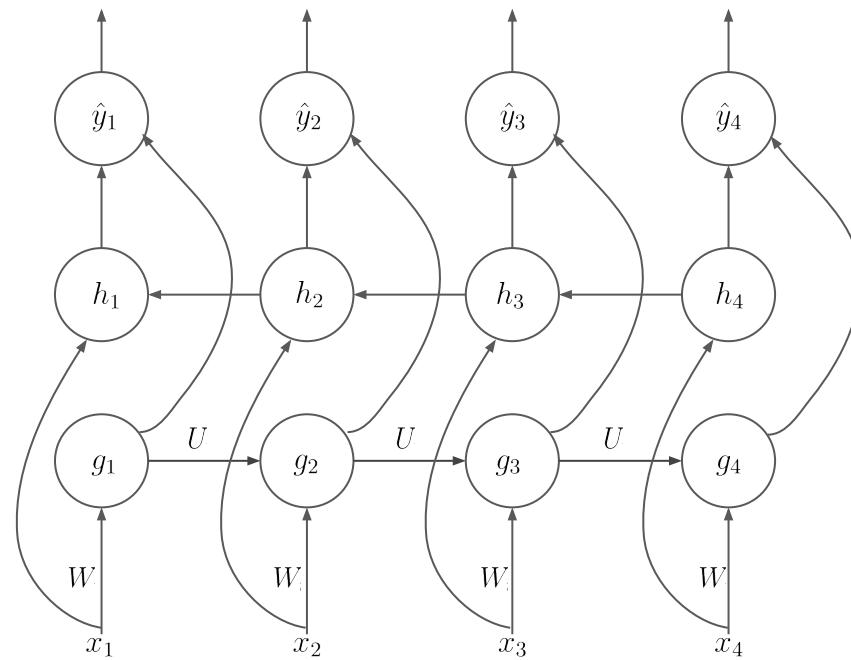
# Bidirectional Network



# Bidirectional Network



# Bidirectional Network

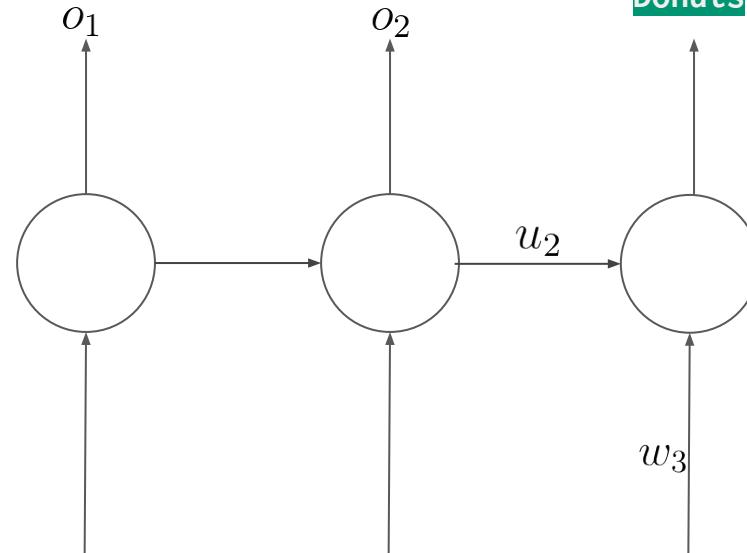




# RNN Shortcoming

Humans love **Donuts**

**Donuts**



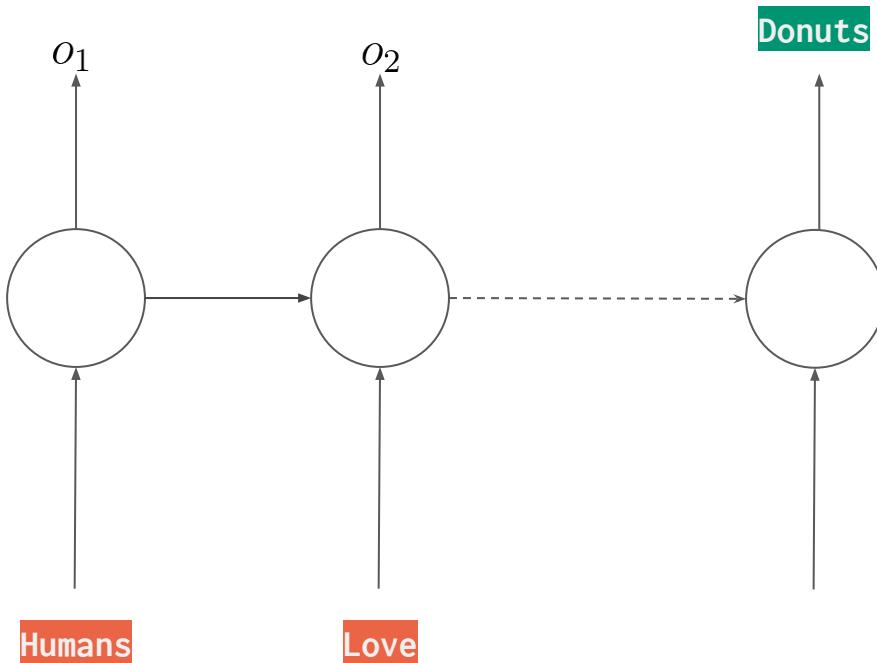
Consider a language model  
predicting next word.



# Shortcoming

Humans love sweet things and **Donuts** are the best

Long range information is difficult to handle.



# RNN shortcomings

- Lossy summary.
- Information is passed via gradients and if the gradients are small, information might be lost.

# Long Short Term Memory Cell

- Cell has a memory to store information.
- Gates which decide whether to “forget” some of the memory, given new input.
- Learns when and what to “forget”, “input” and “output”