

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

How to learn with text?

How to learn with text

What do you want to do with the text?

Our case: generate query

Simpler example: basic sentiment classification of text

(classification = given text, assign a label to it)

(sentiment classification = is the text positive or negative?)

Text classification

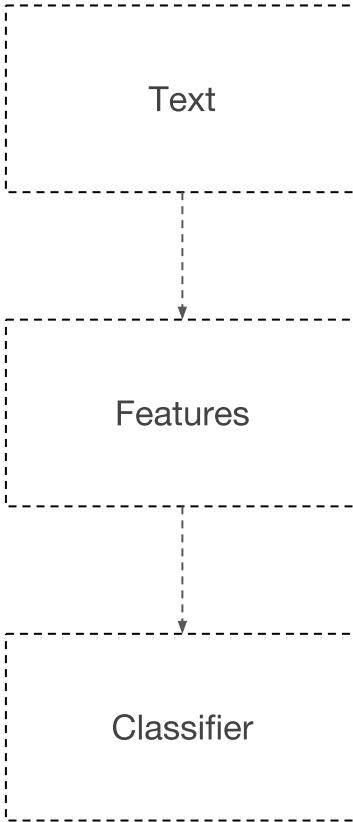
Target classes: positive/neutral

Or: how positive is the sentence?

Example 1: “Definitely, Maybe was so good” → very positive

Example 2: “Definitely, Maybe was not so good” → negative

How to learn this?



How to learn with text

1. Get features of text
 - a. useful properties about the input
2. Classify
 - a. Use the properties to form a decision

Text classification - features?

What features can we use?

- Words in the sentence:
“Definitely, Maybe was so good”
 $\rightarrow \{\text{“Definitely”, “maybe”, “was”, “so”, “good”}\} \leftarrow (\text{a set!})$

⇒ Represent input as a vector:

$$[\ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ \dots]$$

was so good

= bag-of-words vector = sum of one-hot vectors for words

Text classification - classifier?

⇒ Feature vector:

$$x^T = [\quad 0 \quad \dots]$$

was so good

$$w^T = [\quad w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \quad w_7 \quad w_8 \quad w_9 \quad w_{10} \quad w_{11} \quad w_{12} \quad \dots]$$

Classifier has parameter vector w

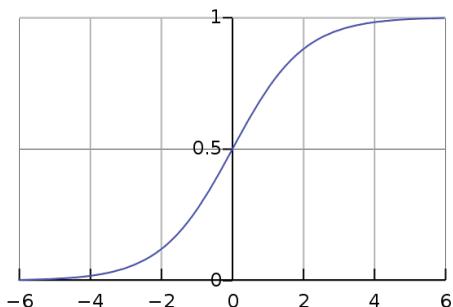
Parameters are changed during training.

Inner product $x^T w$ gives a score.

Text classification - classifier?

Inner product $x^T w$ gives a score.

We want to know probability of text being positive!
⇒ sigmoid

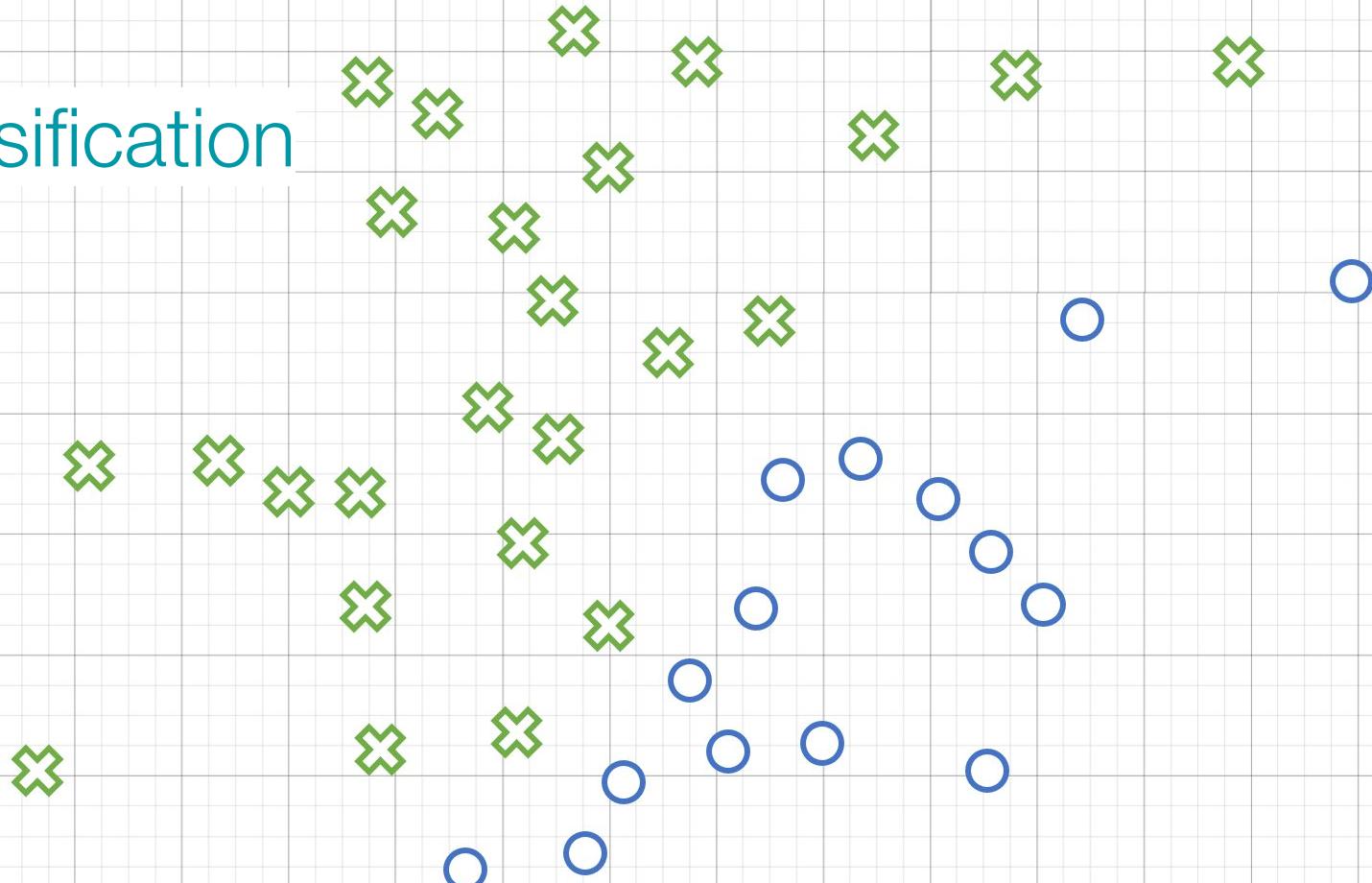


$$P(y = \text{True}|\mathbf{x}) = \sigma(\mathbf{x}^T \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{x}^T \mathbf{w}}}$$

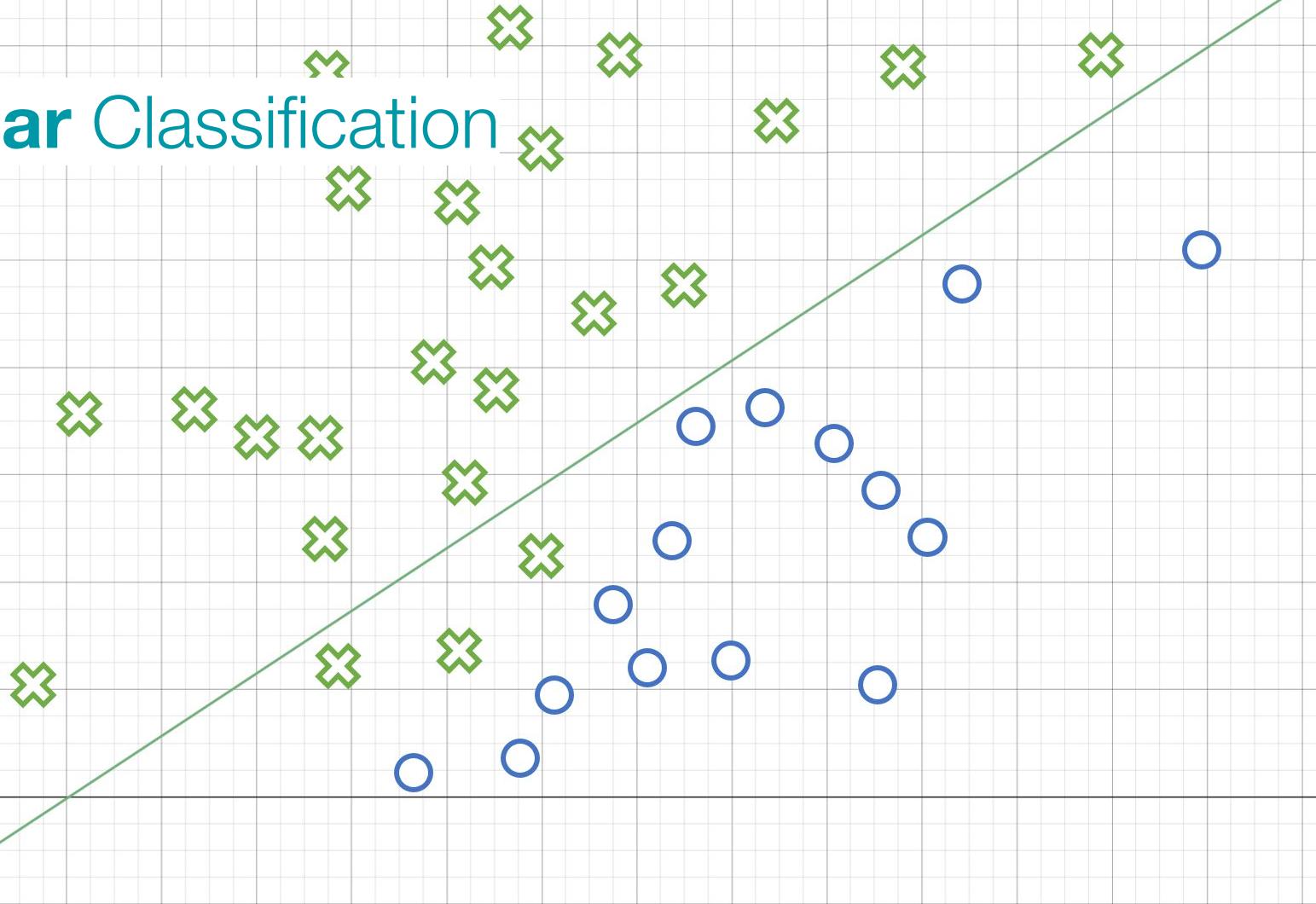
Training = changing parameters to maximize probability of correct class

*other classifiers are possible too

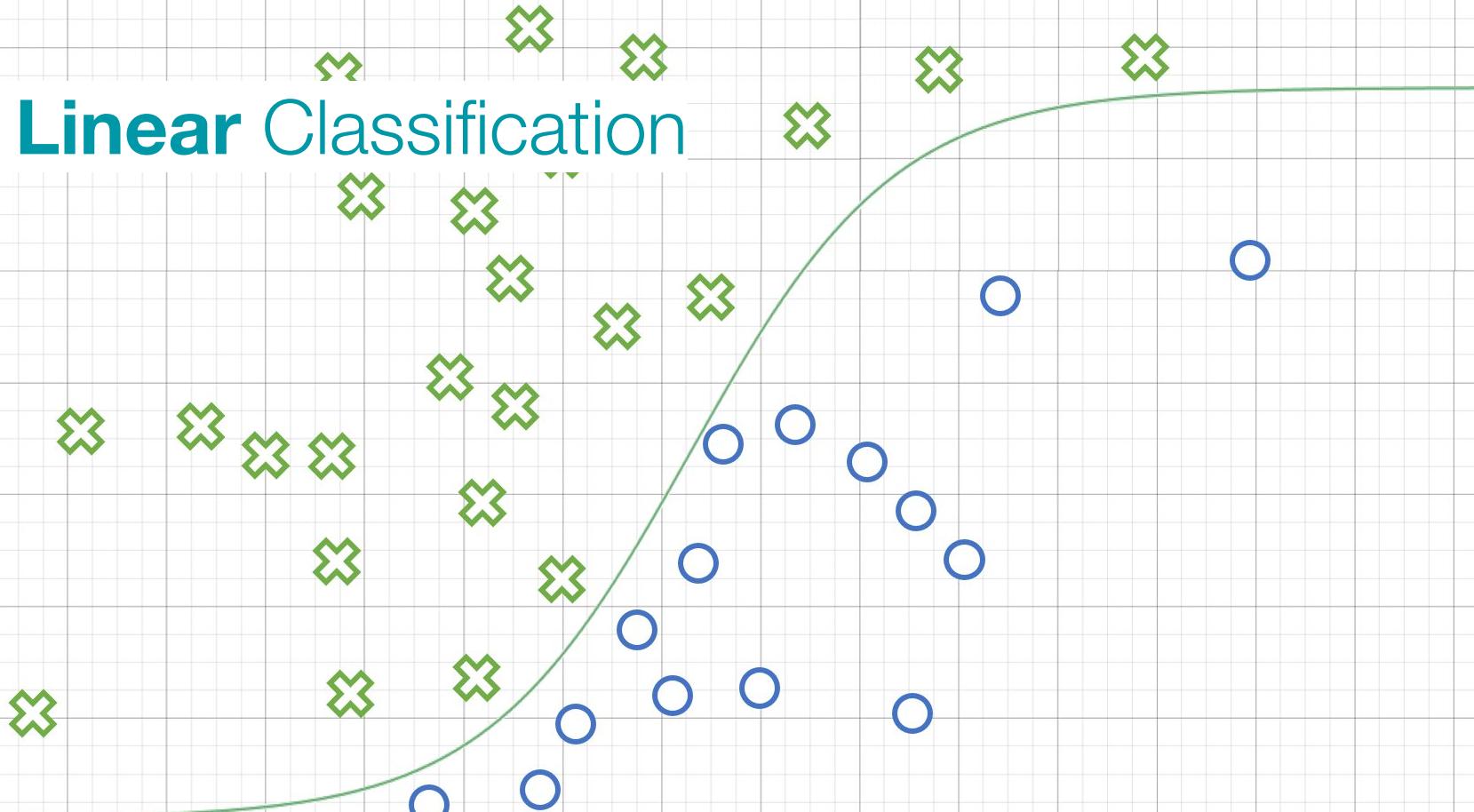
Classification



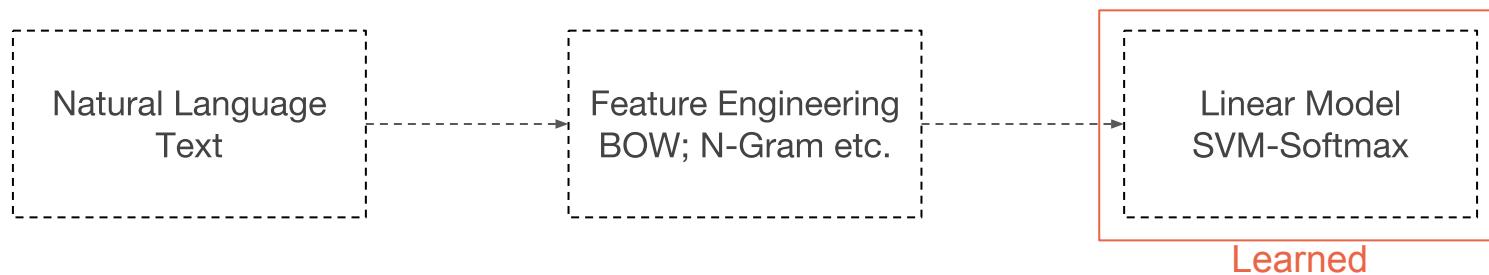
Linear Classification



Non Linear Classification



Traditional ML



Traditional ML - problem

Coming up with good features can be difficult, requires time and expertise. Change of domain can be difficult.

Sentiment classification = learns how positive each word is.

But doesn't know:

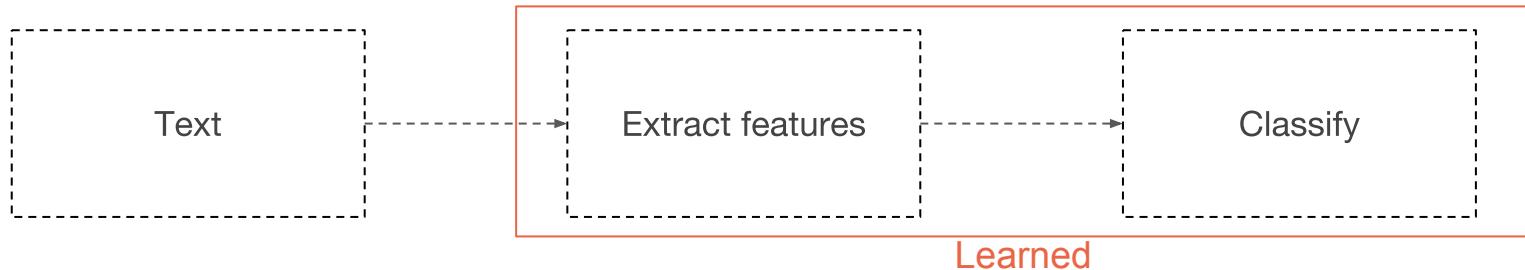
- Compositionality of language → can try n-grams (but more features and might still miss interesting patterns)
- Each word has independent parameters (one-hot representation)

Advantage of traditional features: understandable

How about...

... we let the machine learn how to produce good features?

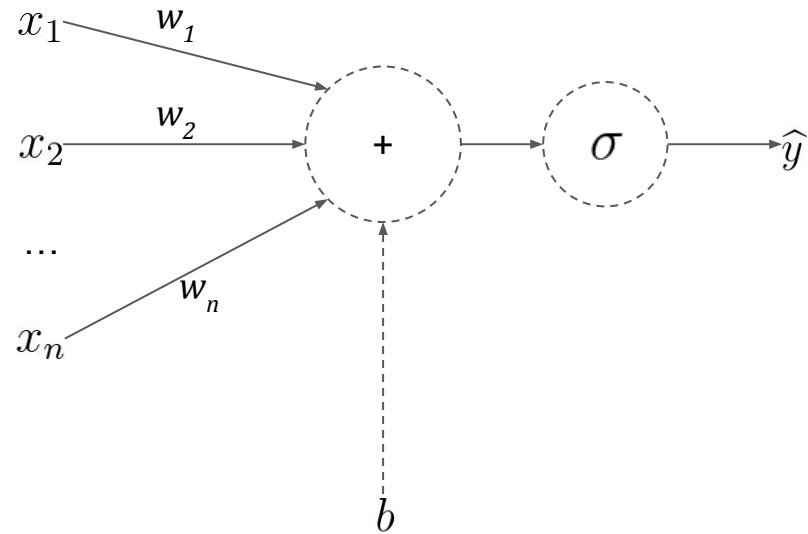
Deep learning



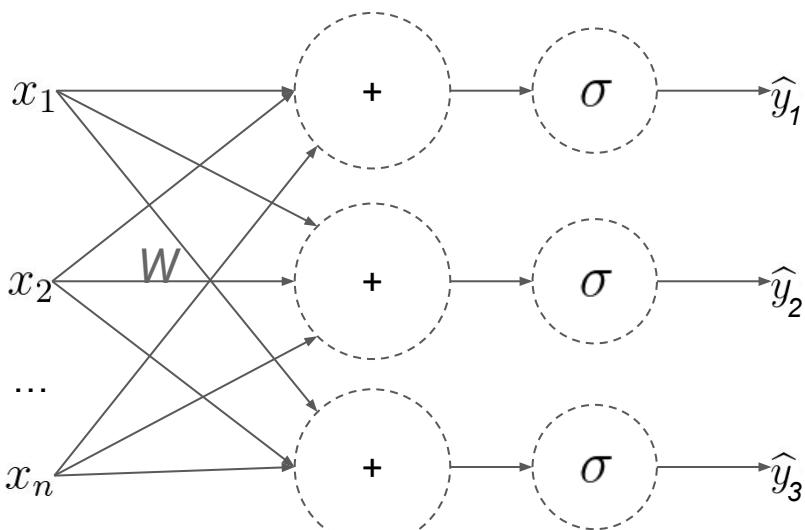
Ideally learns the features automatically a.k.a
Representation learning

Neural Networks

Neuron



A layer of neurons



$$\sigma_{W,b}(X)$$

$$\hat{y} = \sigma(W\vec{x} + \vec{b})$$

$$W \in \mathbb{R}^{d_{out} \times d_{in}}$$

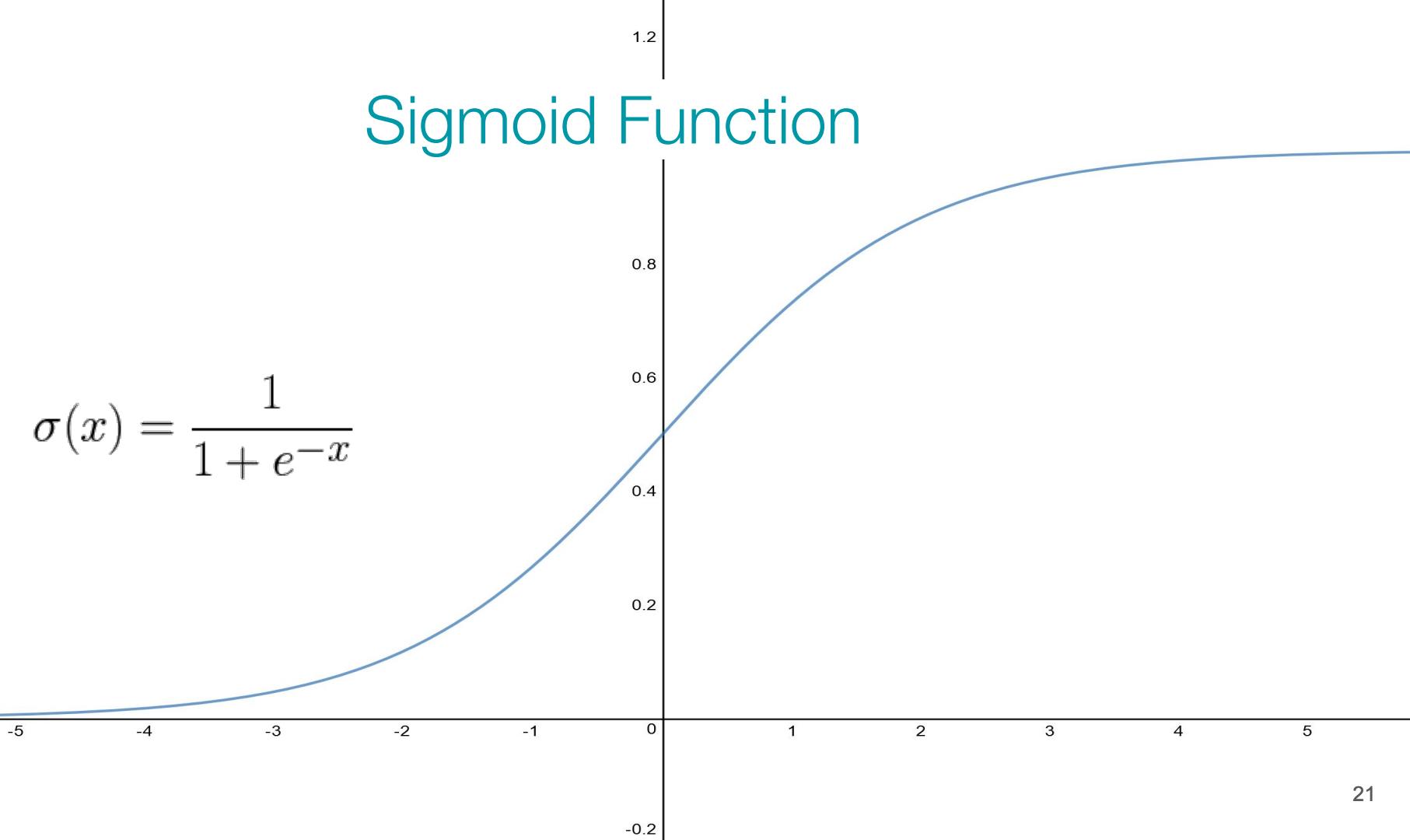
$$\vec{x} \in \mathbb{R}^{d_{in}}$$

$$\vec{b} \in \mathbb{R}^{d_{out}}$$

$$\hat{y} \in \mathbb{R}^{d_{out}}$$

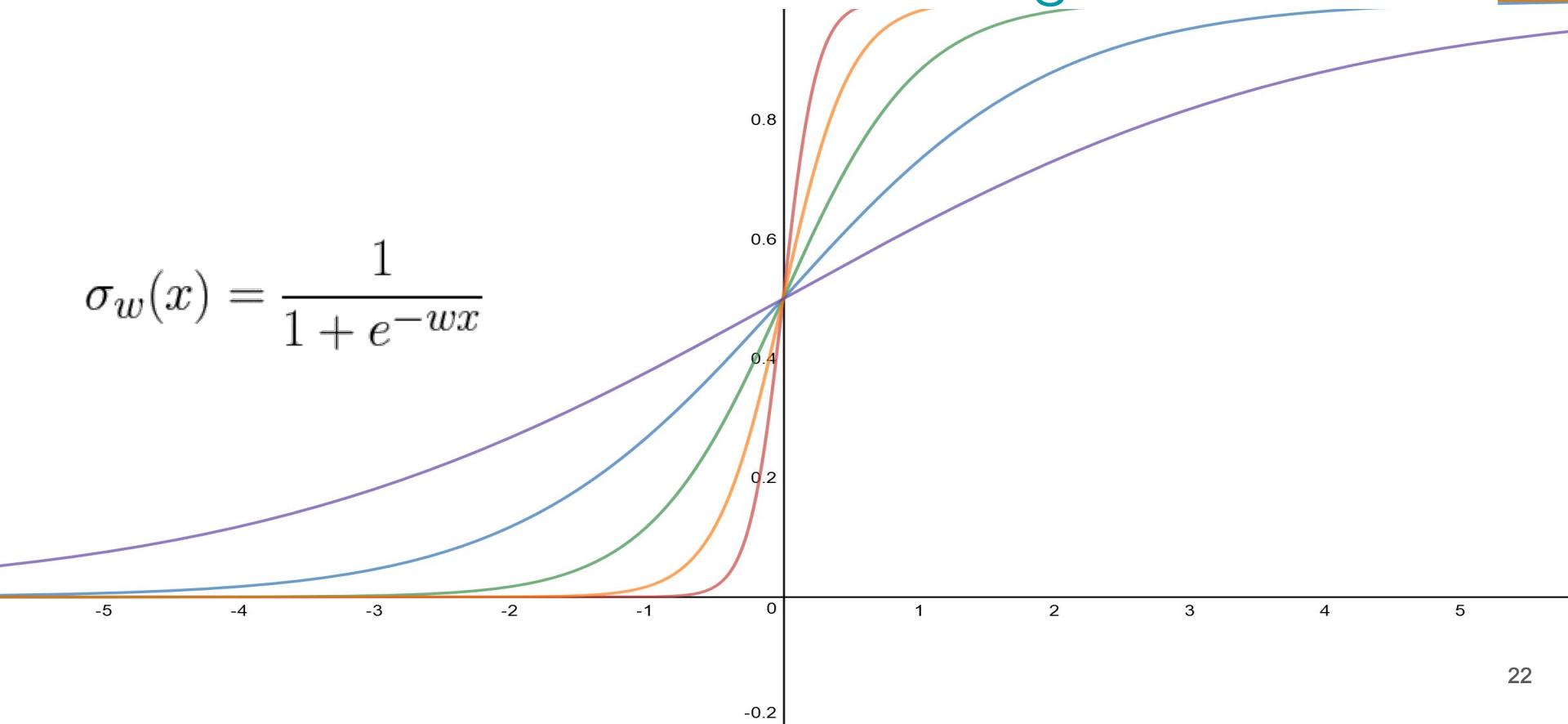
Sigmoid Function

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



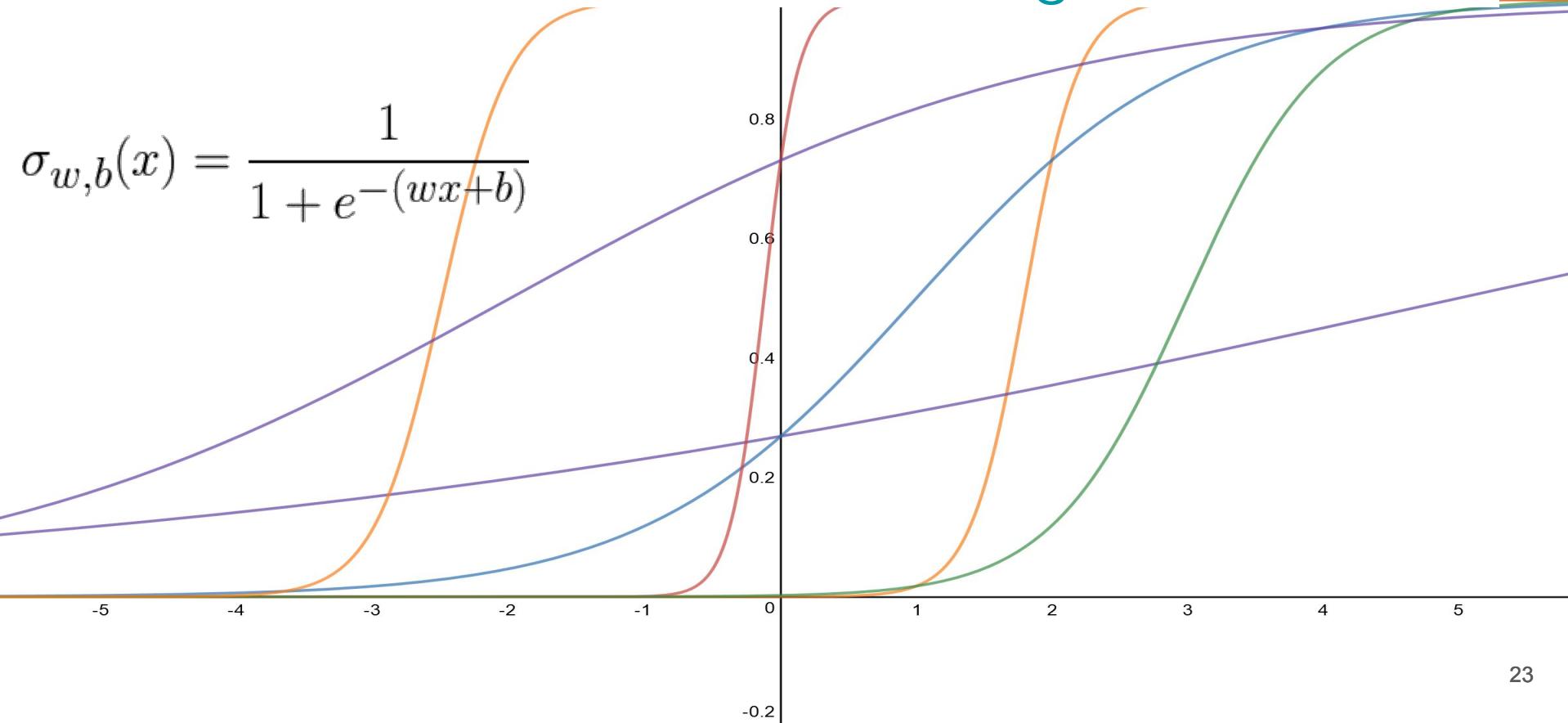
Parameterized Sigmoid Function

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

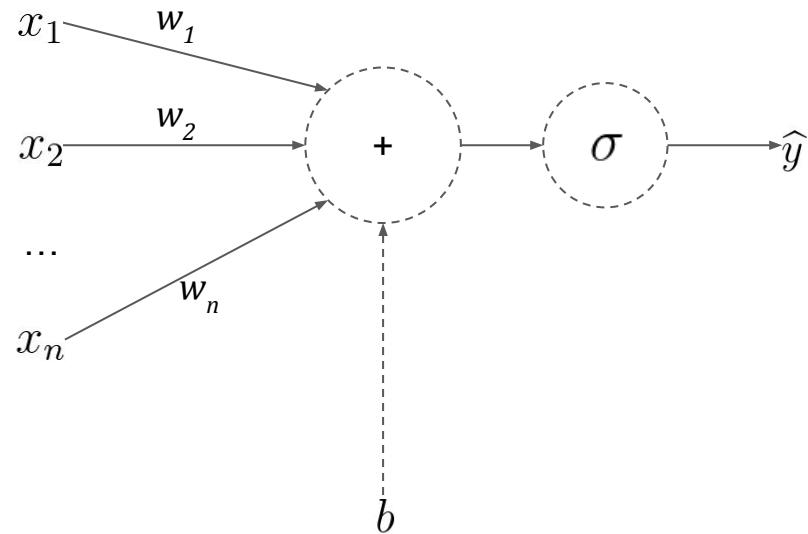


Parameterized Sigmoid Function

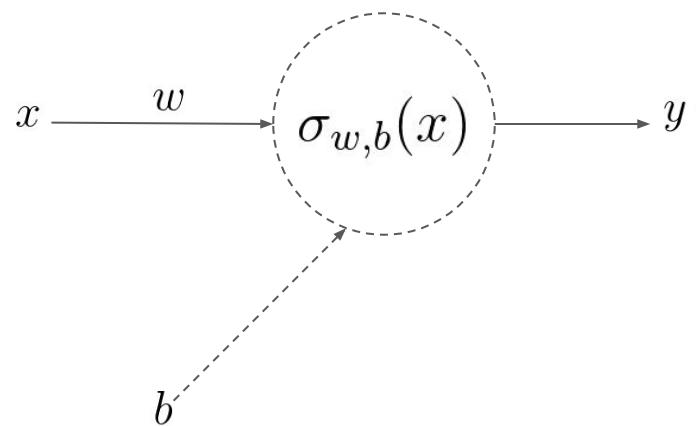
$$\sigma_{w,b}(x) = \frac{1}{1 + e^{-(wx+b)}}$$



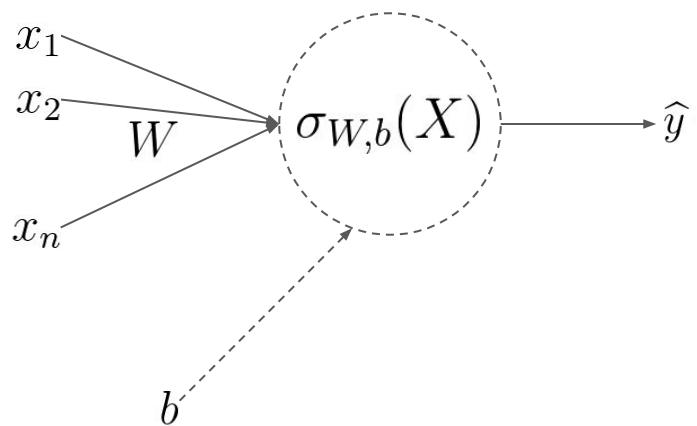
Neuron



Neuron



Vector Inputs to Neuron



$$\hat{y} = \sigma(W\vec{x} + \vec{b})$$

$$W \in \mathbb{R}^{d_{out} \times d_{in}}$$

$$\vec{x} \in \mathbb{R}^{d_{in}}$$

$$\vec{b} \in \mathbb{R}^{d_{out}}$$

$$\hat{y} \in \mathbb{R}^{d_{out}}$$

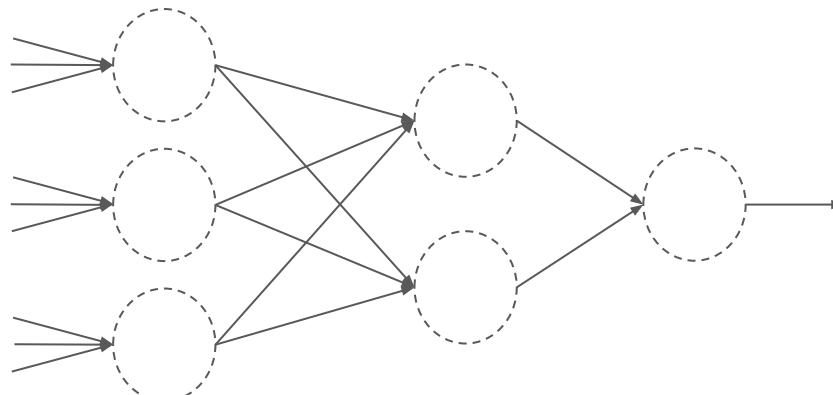
Stacking (Composite Functions)



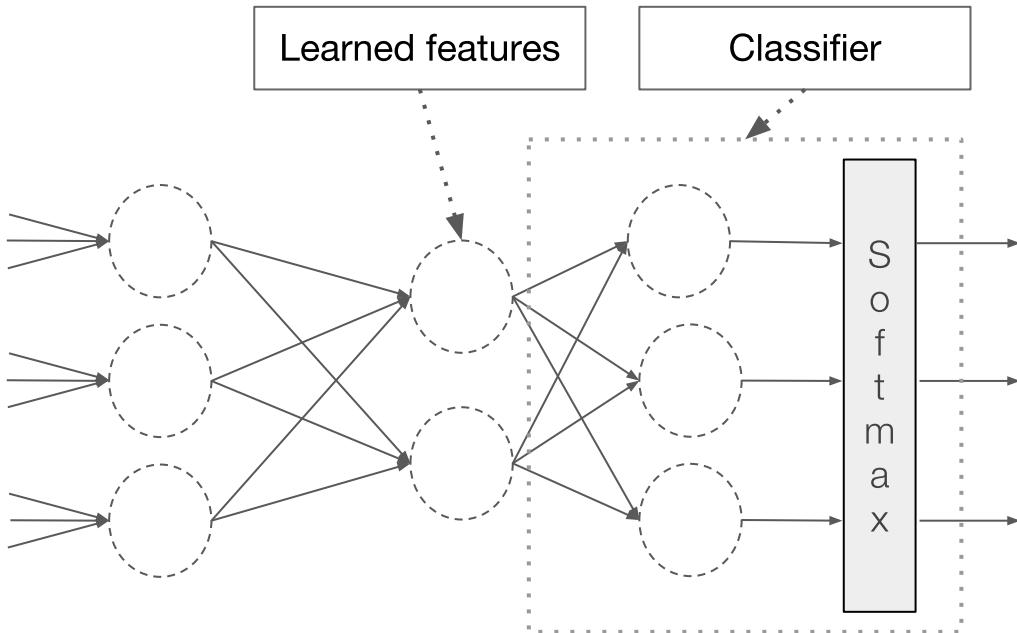
$$\sigma_{W_2, b_2}(\sigma_{W_1, b_1}(X))$$

$$\sigma(W_2 \cdot \sigma(W_1 \cdot \vec{x} + \vec{b}_1) + \vec{b}_2)$$

Feed Forward Network



Multi-class Classification



$$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\top \mathbf{w}_j}}{\sum_{k=1}^K e^{\mathbf{x}^\top \mathbf{w}_k}}$$

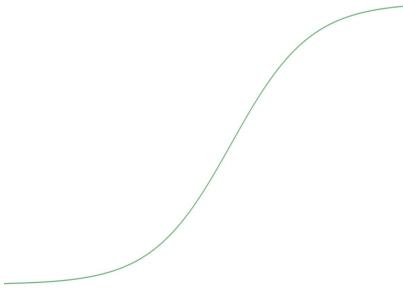
Activation Functions

Till now we have seen sigmoid function.

Can use several other functions such as

Hyperbolic Tangent

Rectifier



Non linearity

Can be stacked (composite functions) to increase expressivity.

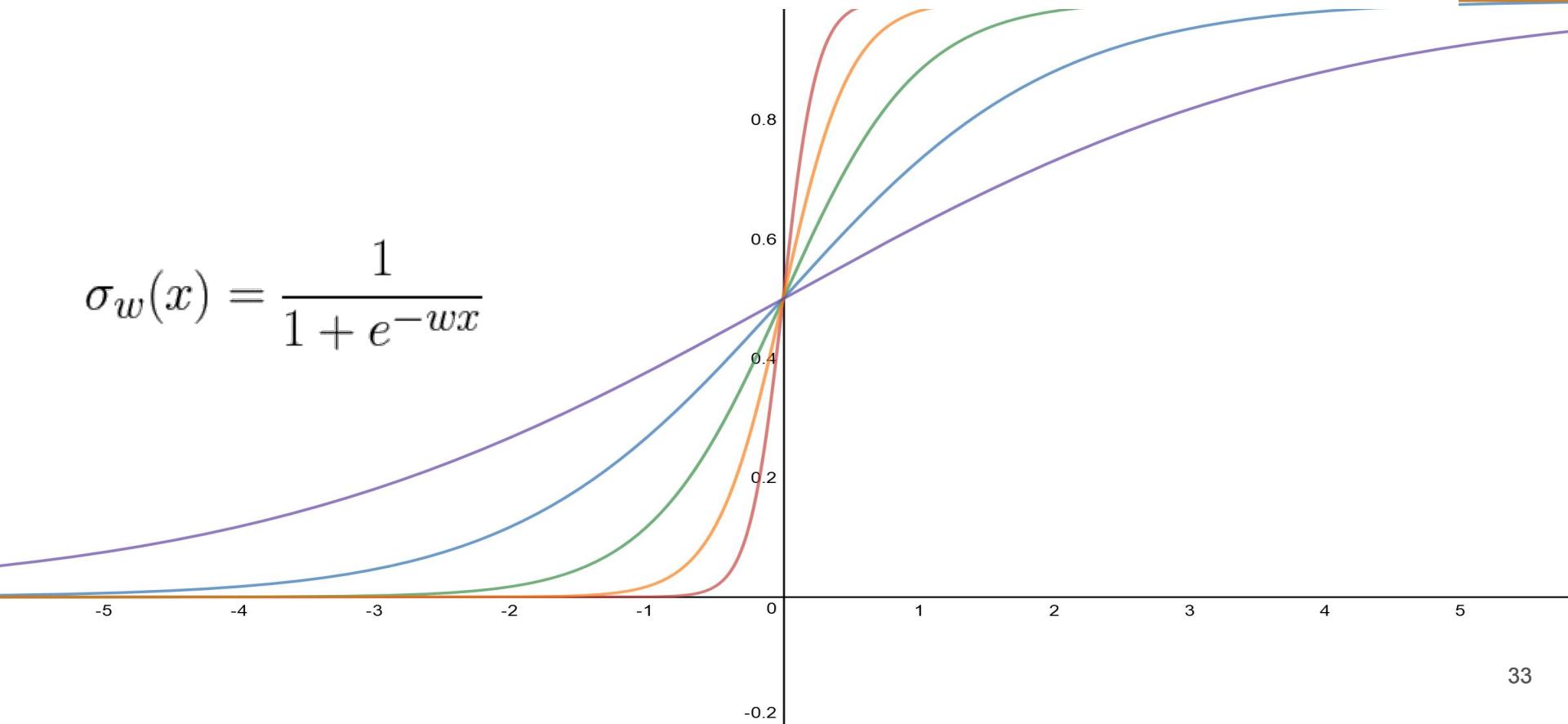
Their gradients are smoother (derivatives of linear transformations are constant, regardless of the input.)

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

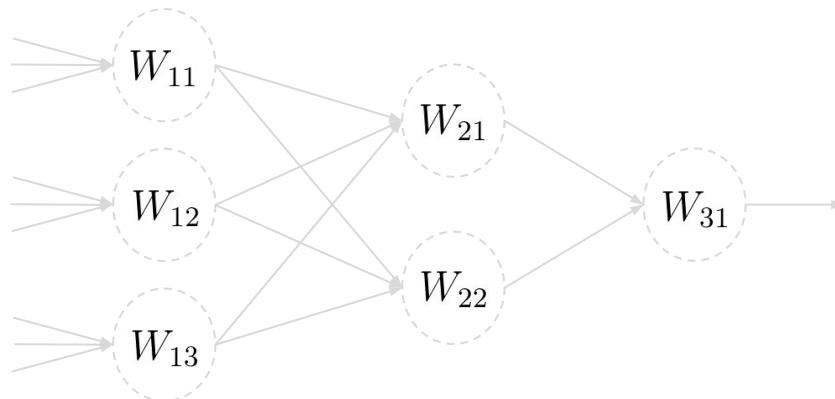
Training Neural Networks

Recall: Parameterized Functions

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

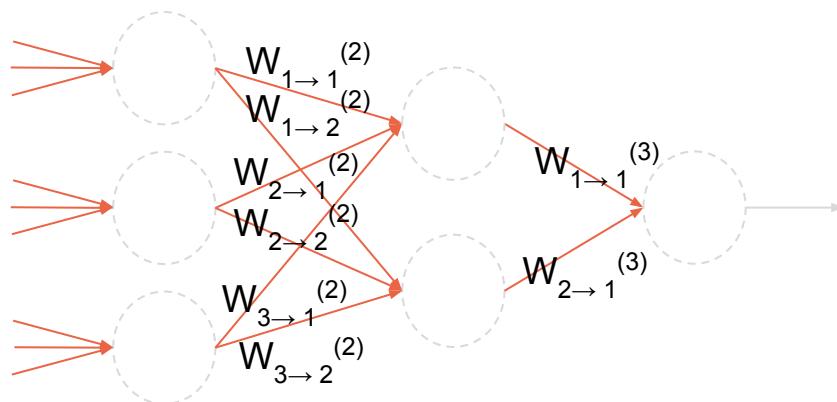


Parameters



* omitted biases for brevity's sake ₃₄

Parameters



* omitted biases for brevity's sake ₃₅

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

Backpropagate?

Idea

Backpropagation = chain rule of differentiation

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 values in that direction to *increase* the error.

⇒ we want to minimize error → go against gradient
(Backpropagation: since we have multiple layers, use chain rule to compute derivative w.r.t. loss)

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

Summing it up

- input data: x

x

y

Summing it up

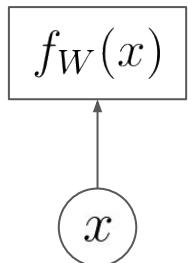
- input data: x
- input data label: y

x

y

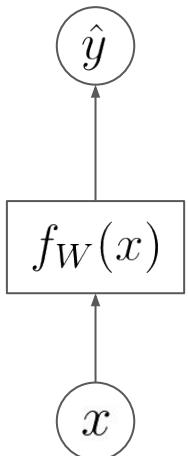
Summing it up

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



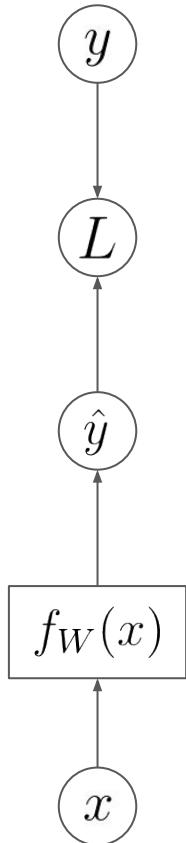
y

Summing it up



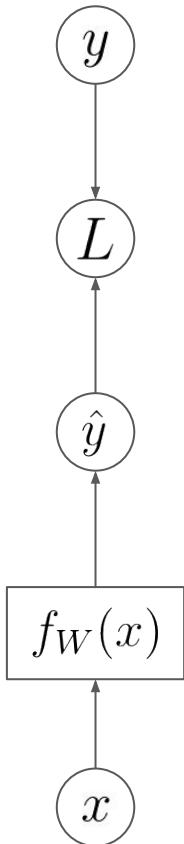
- input data: x
- input data label: y
- model: $f_W(x)$
- model output: $\hat{y} = f_W(x)$

Summing it up



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

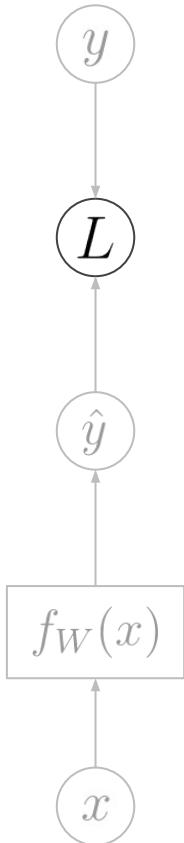
Summing it up



- 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

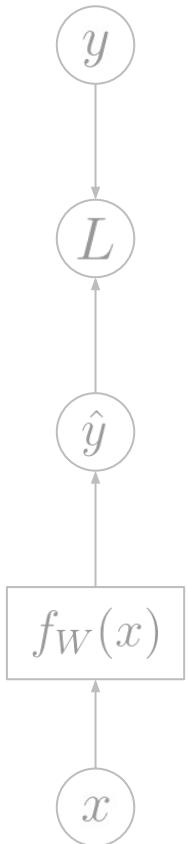


- input data: x
- input data label: y
- model: $f_W(x)$
- model output: $\hat{y} = f_W(x)$
- loss: $L(\hat{y}, y)$

- 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: $L(\hat{y}, y)$

} Forward Propagation

} Backward Propagation

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

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 gradient descent.

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

- Peter Norvig



image from <https://xkcd.com/1838/>

Why now

Faster, cheaper GPU's

Simple to use libraries

Large amount of data

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:

Force a fixed number of dimensions in the vector space.

Desired Properties

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

Forces similar words to appear together (across some dimensions)

Word2Vec

“encodes semantic components as linear vector differences”

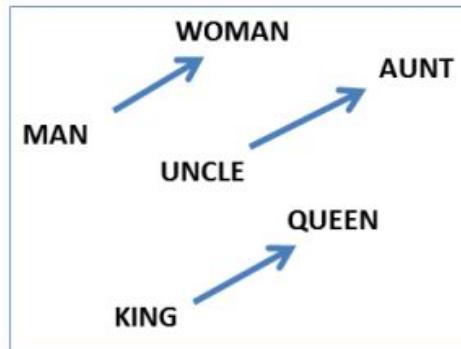
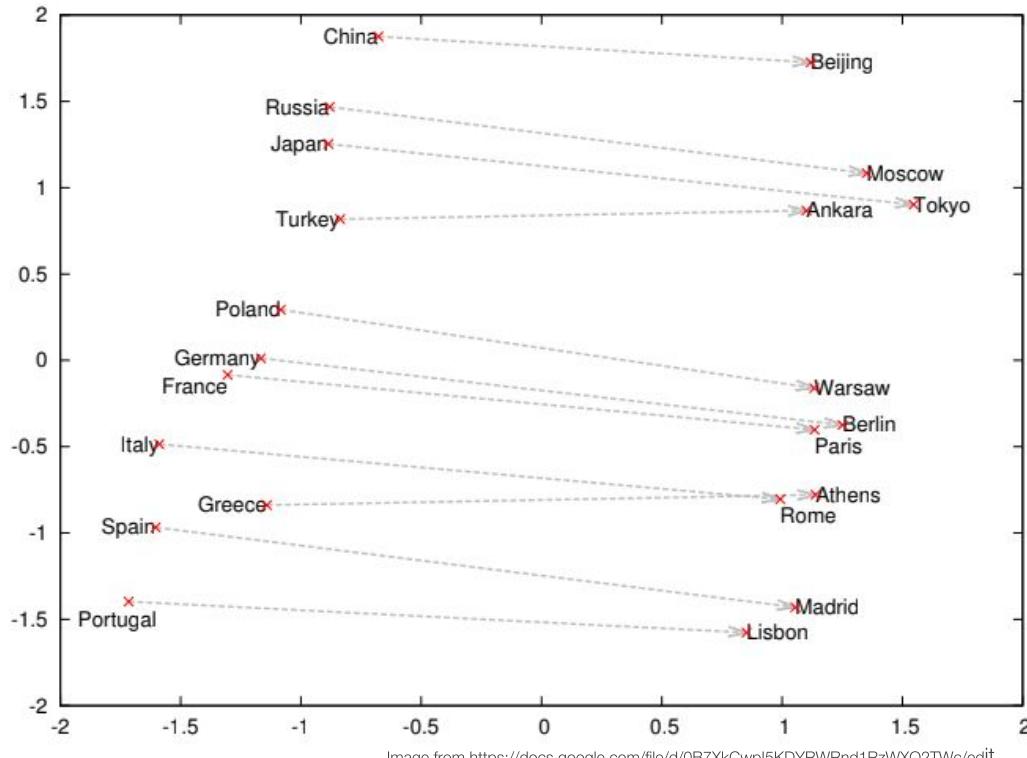


Image from
<https://docs.google.com/file/d/0B7XkCwpl5KDYRWRnd1RzWXQ2TWc/edit>

Meaningful
syntactic and
semantic
regularities
captured

Word2Vec



Meaningful
syntactic and
semantic
regularities
captured

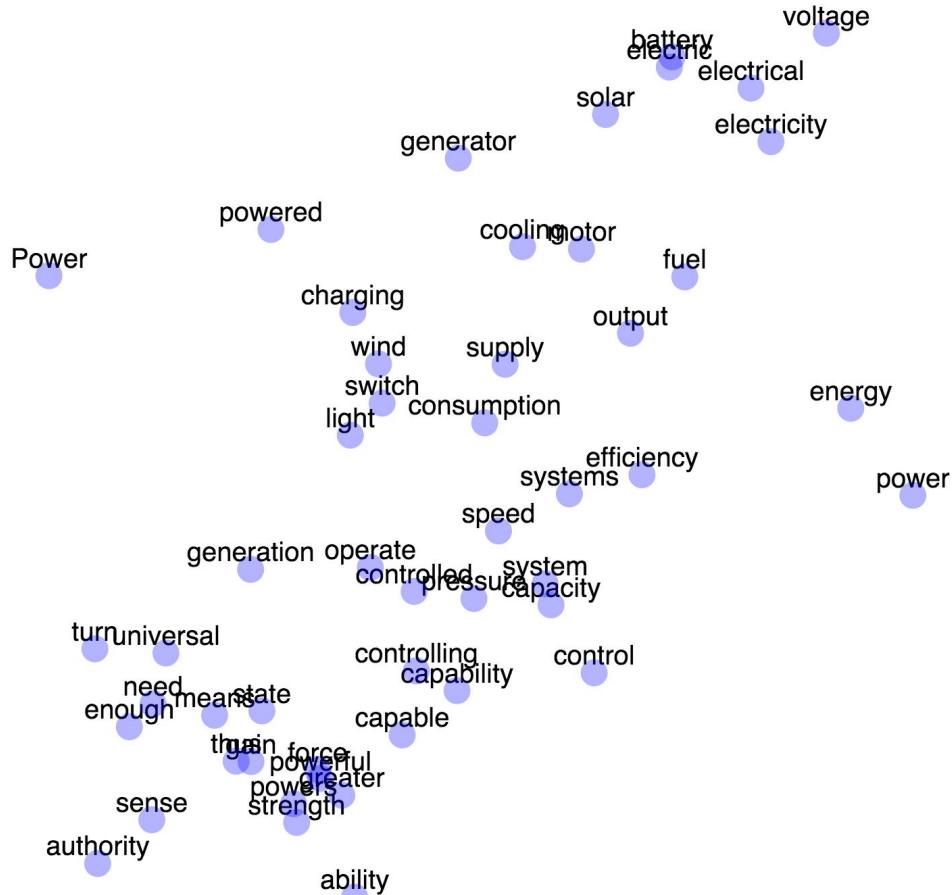


Image from
<https://github.com/danielvarga/word2vec-web-visualization>

Meaningful
syntactic and
semantic
regularities
captured

Desired Properties

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



Image from
<https://github.com/danielvarga/word2vec-web-visualization>

Word Embeddings

Widely used for most NLP tasks.

Different training methods possible:

- Word2vec (Mikolov et al., 2013)
- GloVe (Pennington et al., 2013)

Pre-trained, off the shelf solutions.

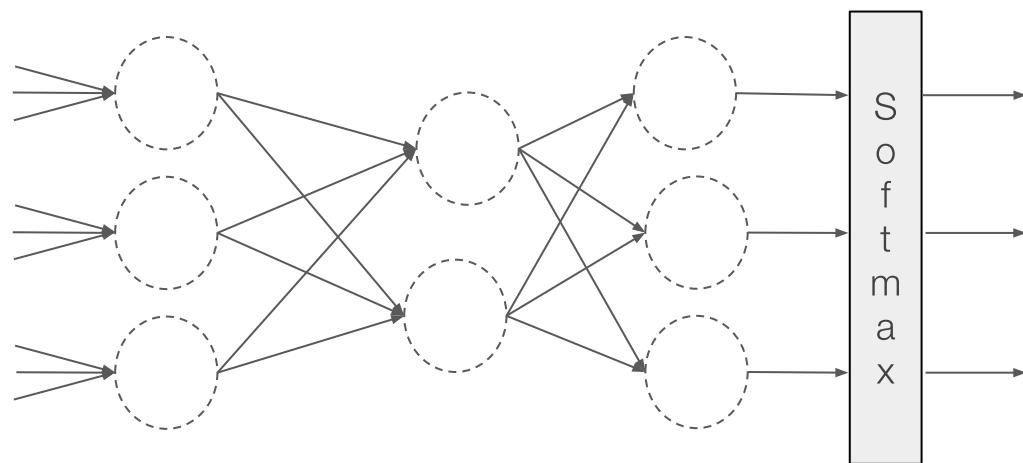
Word Embeddings are matrices

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

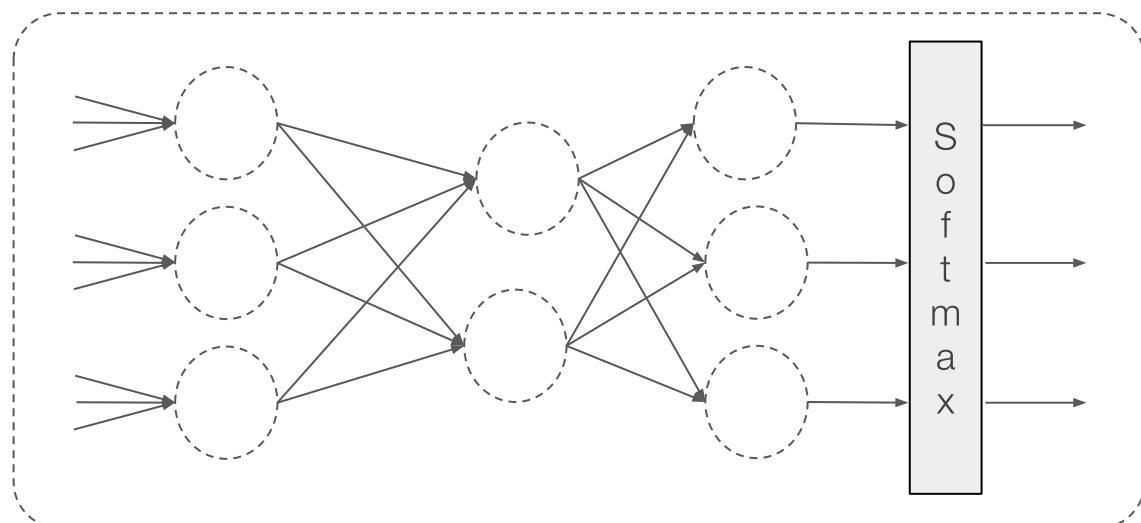
Word Embeddings are matrices

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

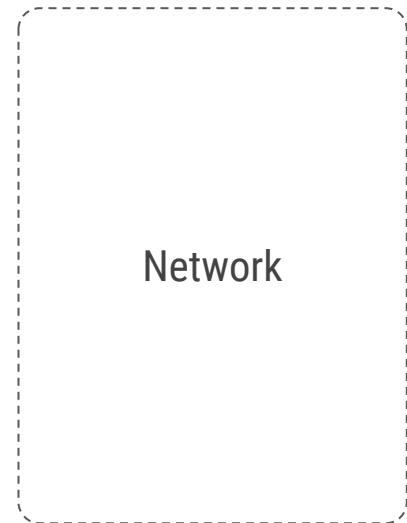
Using Word Embeddings in NN



Using Word Embeddings in NN



Using Word Embeddings in NN



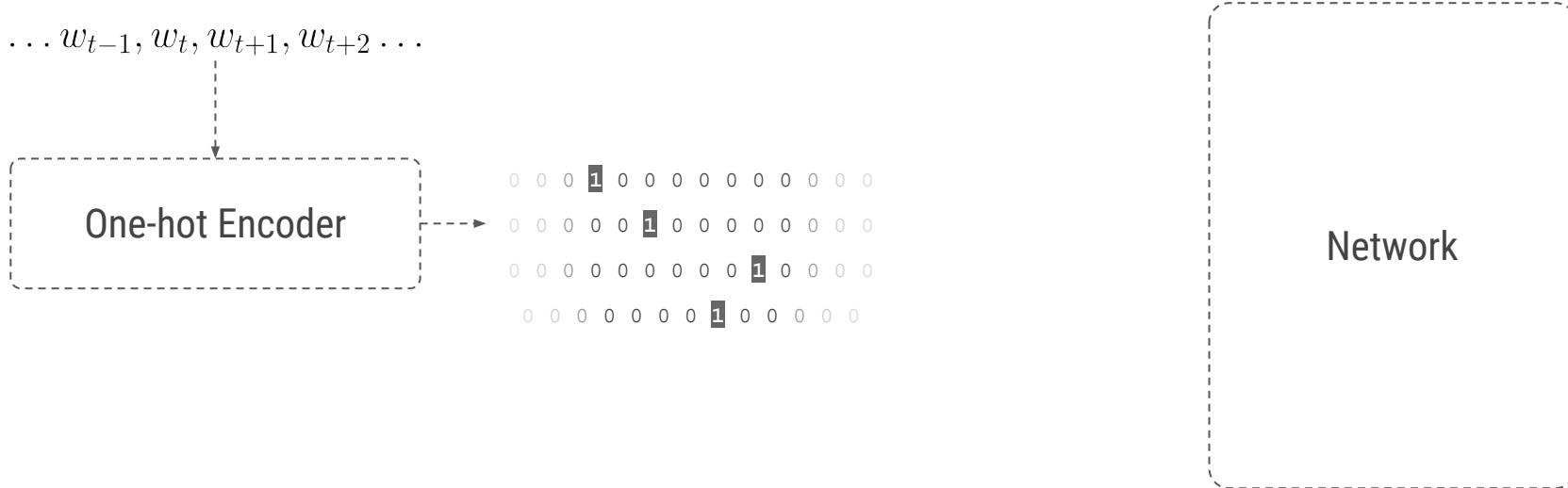
Using Word Embeddings in NN

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

Network

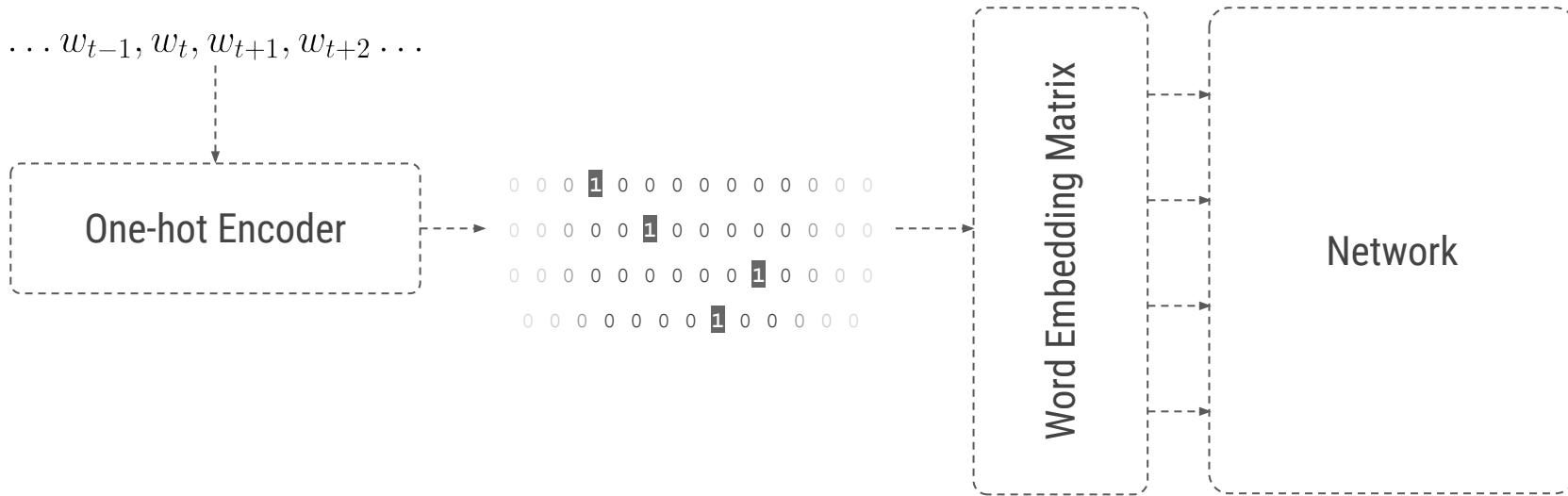
Using Word Embeddings in NN

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

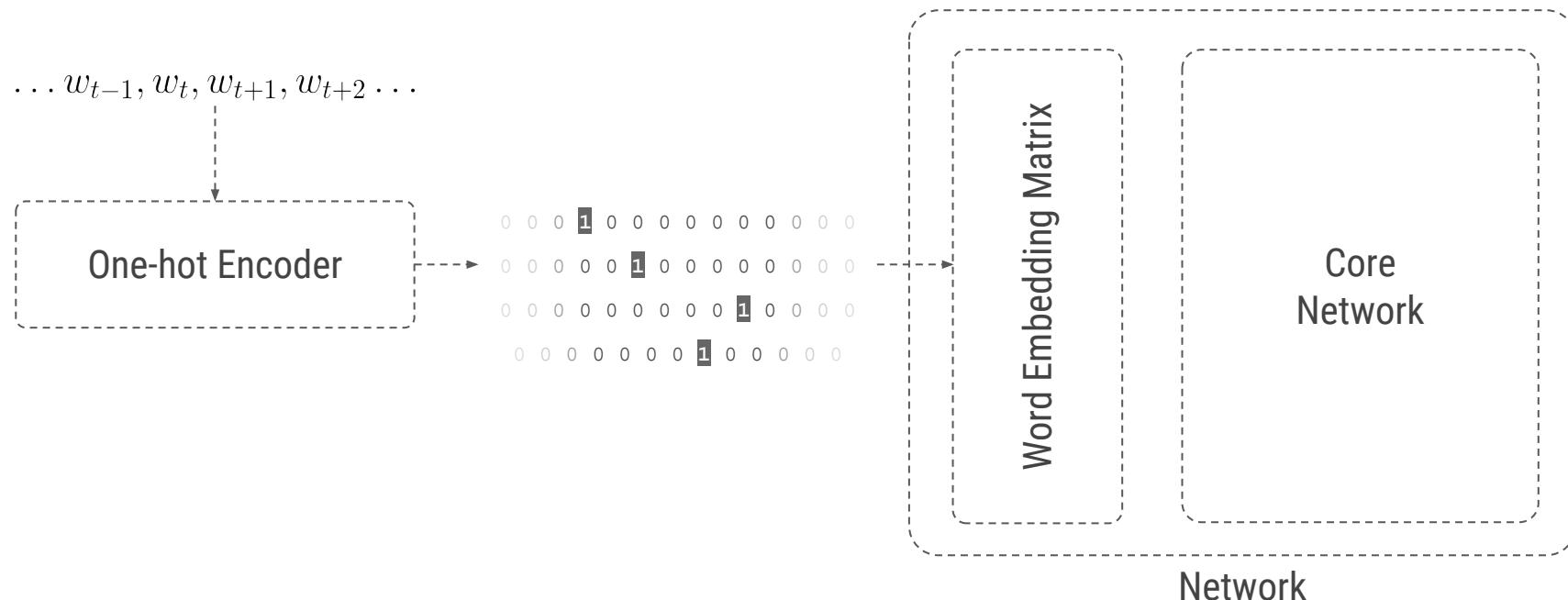


Using Word Embeddings in NN

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



Trainable Embeddings



Representing Language

Words have inherent meanings.

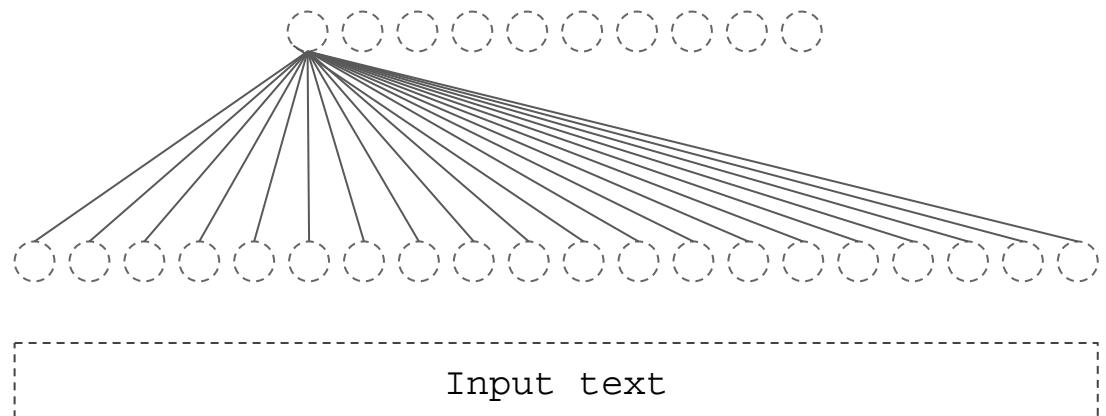


Language is sequential in nature.

Problems with Feedforward Nets

Number of parameters:

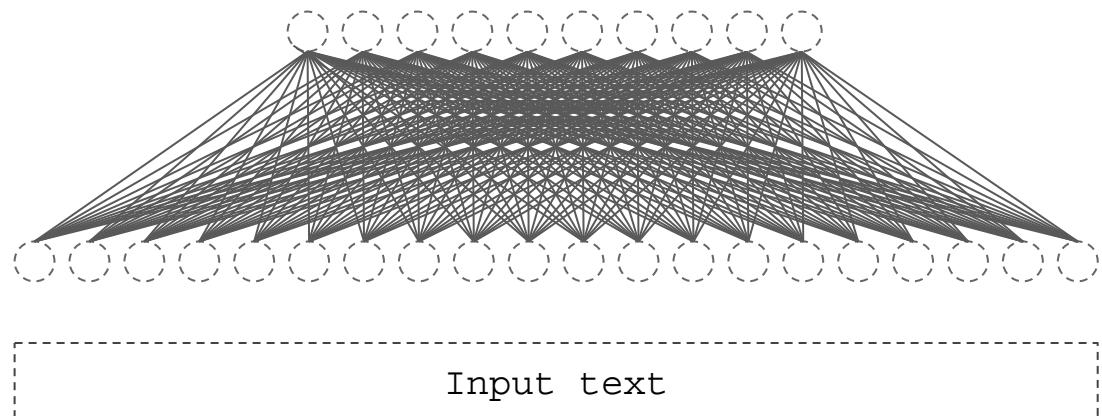
$$20 \times 10$$



Problems with Feedforward Nets

Number of parameters:

$$20 \times 10$$



Problems with Feedforward Nets

Number of parameters:

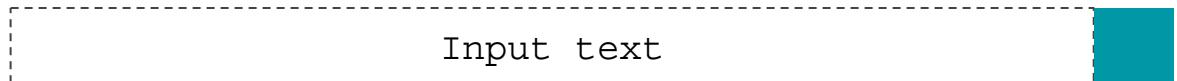
$$20 \times 10$$

Unseen length of text

oooooooooooo

×

oooooooooooooooooooo



Problems with Feedforward Nets

Number of parameters:

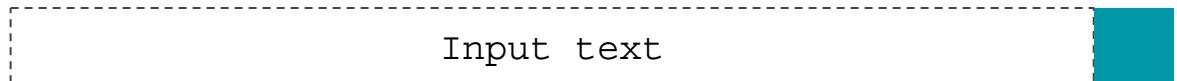
$$20 \times 10$$

Unseen length of text

oooooooooooo



oo●oooooooooooo●ooo



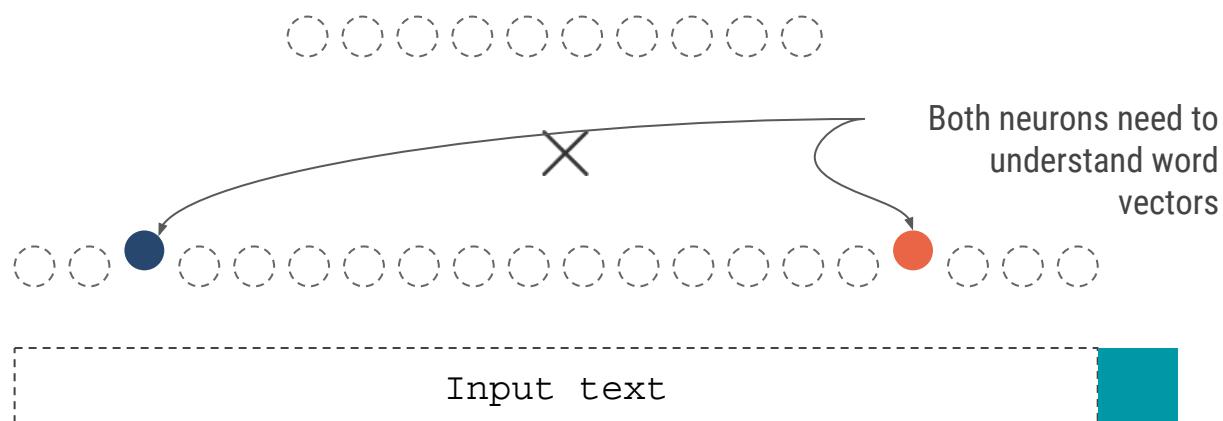
Problems with Feedforward Nets

Number of parameters:

$$20 \times 10$$

Unseen length of text

Independent weights
learning the same thing.



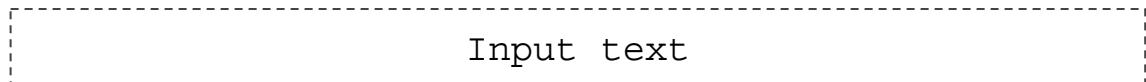
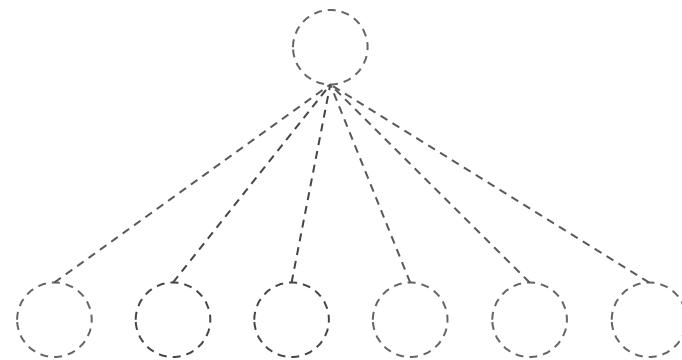
Problems with Feedforward Nets

Number of parameters:

$$20 \times 10$$

Unseen length of text

Independent weights
learning the same thing.



Problems with Feedforward Nets

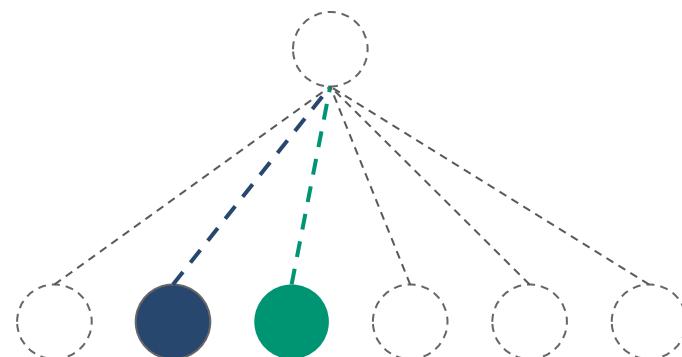
Number of parameters:

$$20 \times 10$$

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

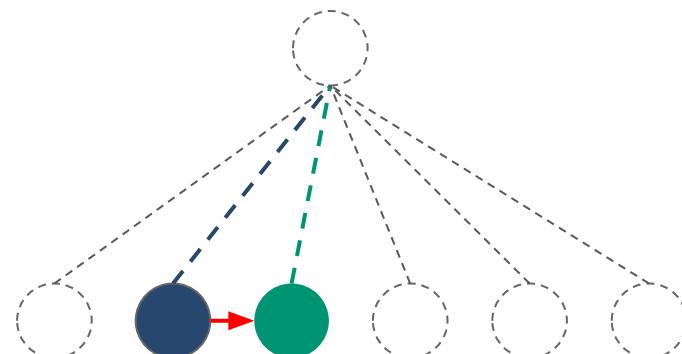
Number of parameters:

$$20 \times 10$$

Unseen length of text

Independent weights
learning the same thing.

Language is sequential.



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

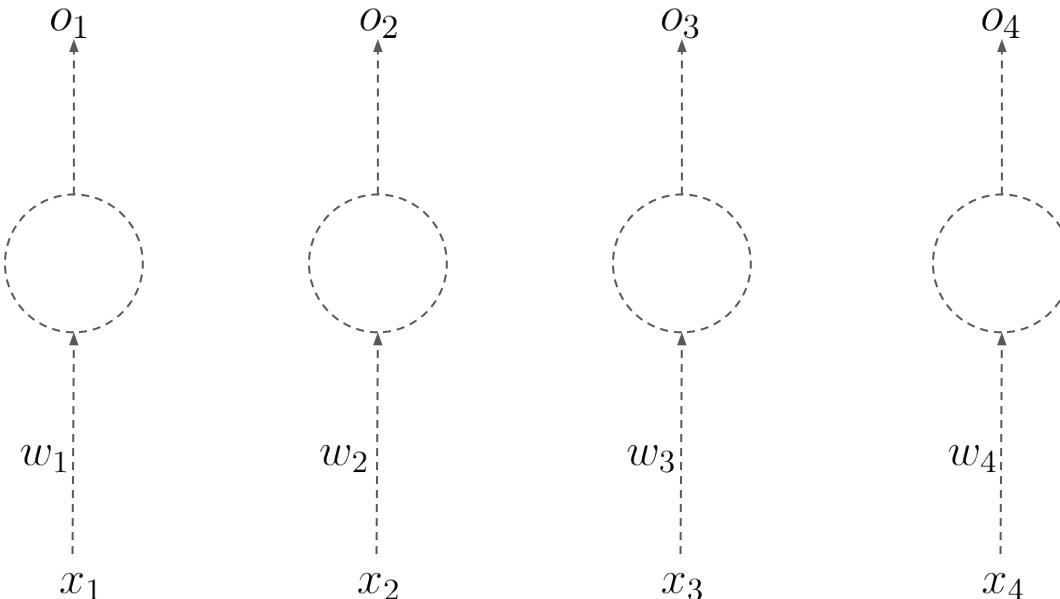
B: United States **isn't** **in** North America

Language is Sequential

Networks for Sequences

How to use NN with sequences?

Standard FF layer

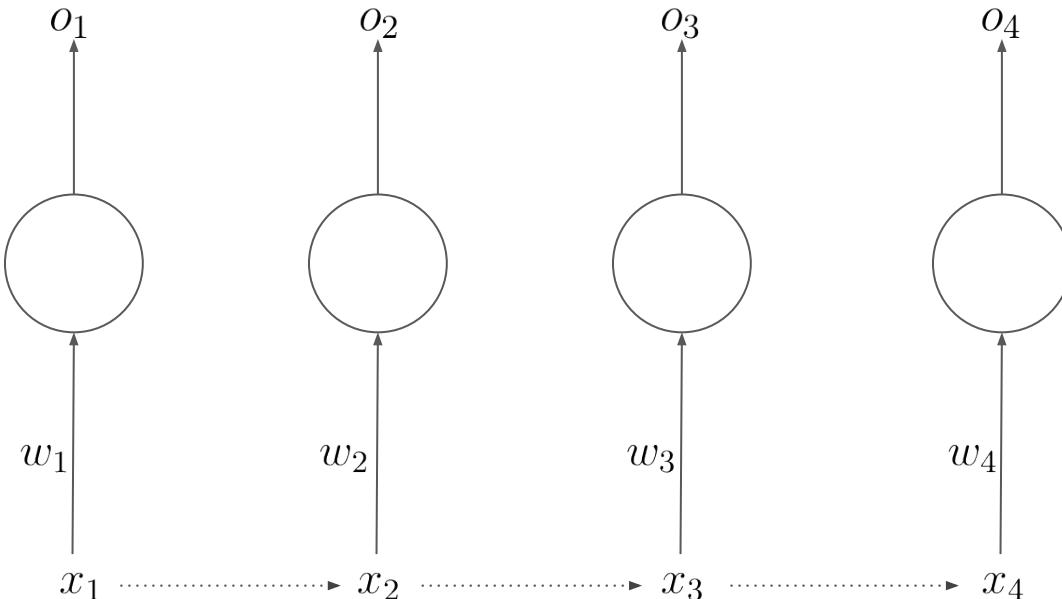


Sequential Networks?

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

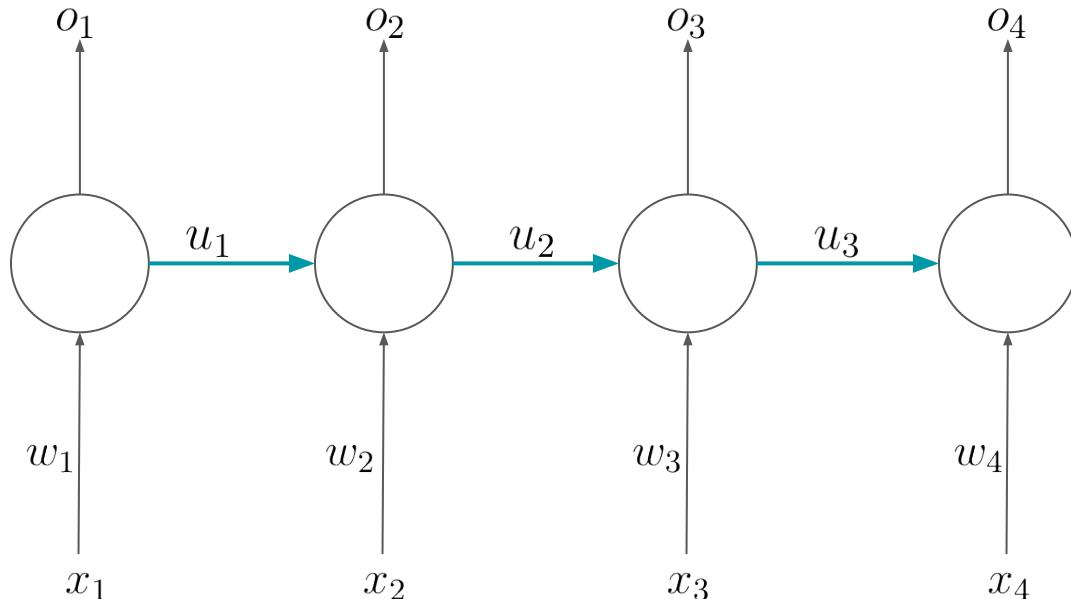
Standard FF layer

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.

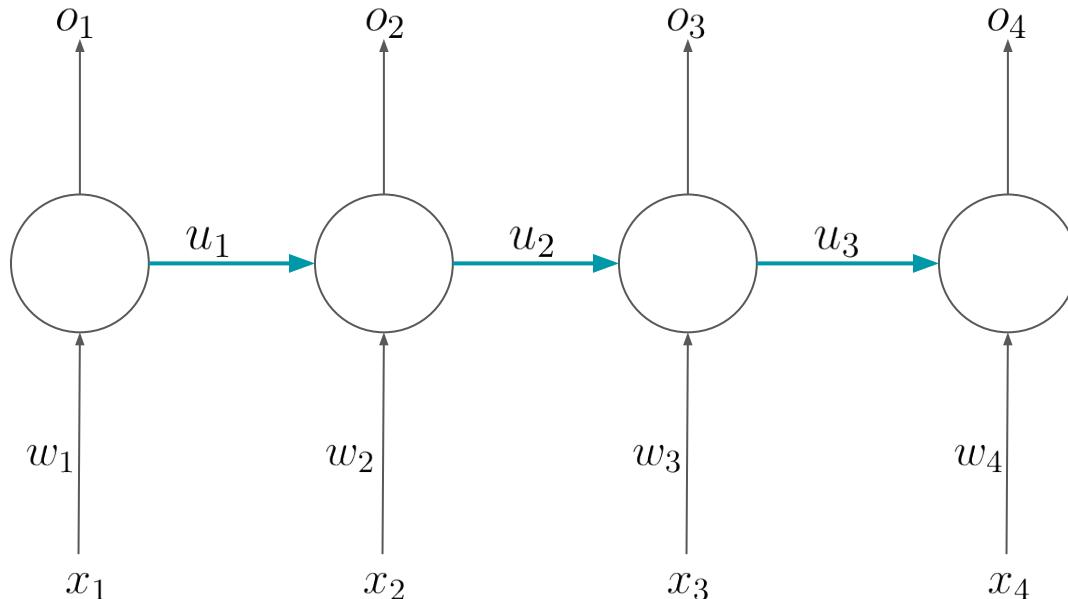
$$p(x_{t-1}, x_t) \neq p(x_{t-1}) \times p(x_t)$$

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

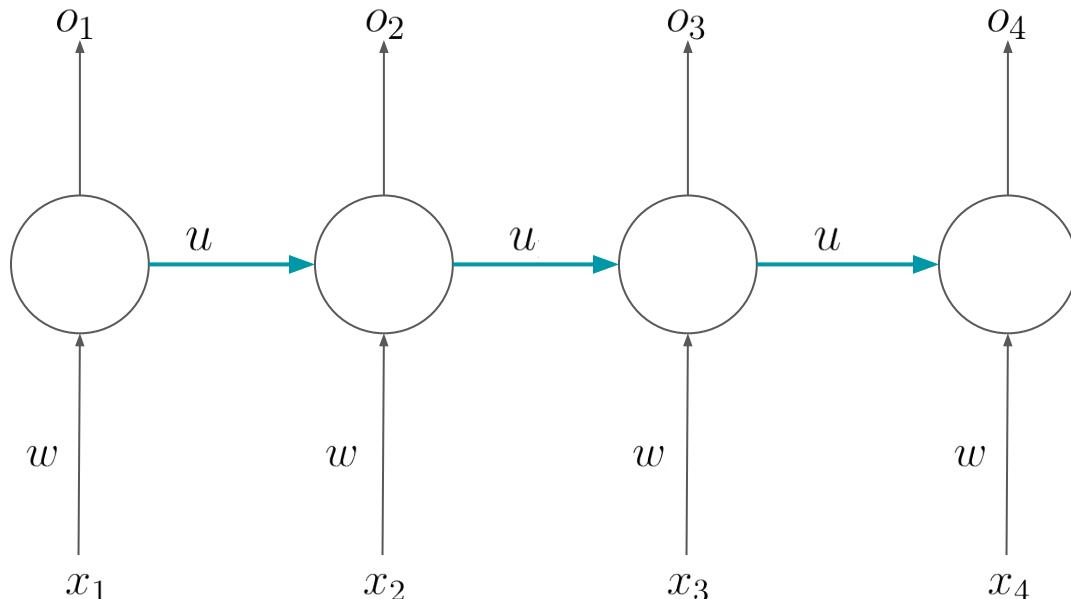
Standard FF layer

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

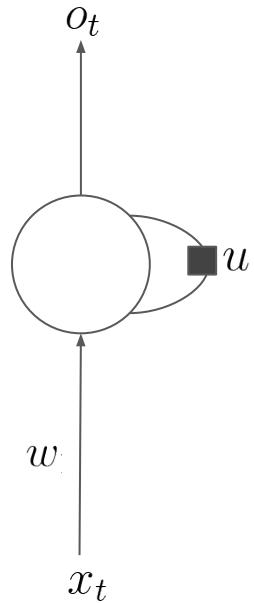
- Time dependent
- Affect the next one
- **Share Weights**



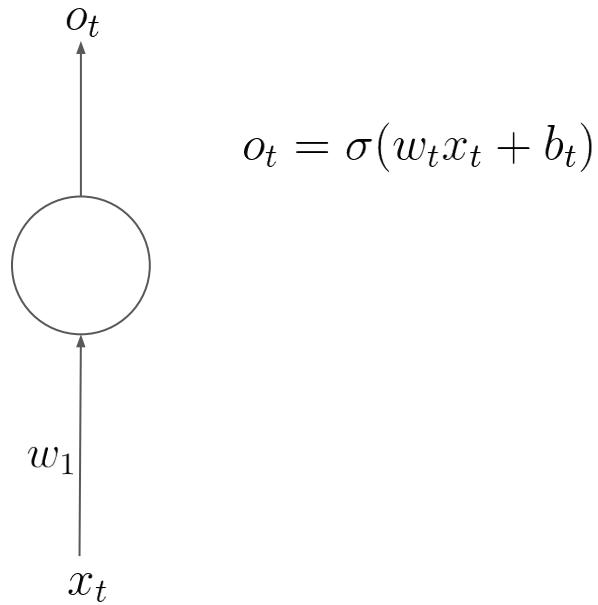
Unfolded Visualization

Neuron in **one layer** are now

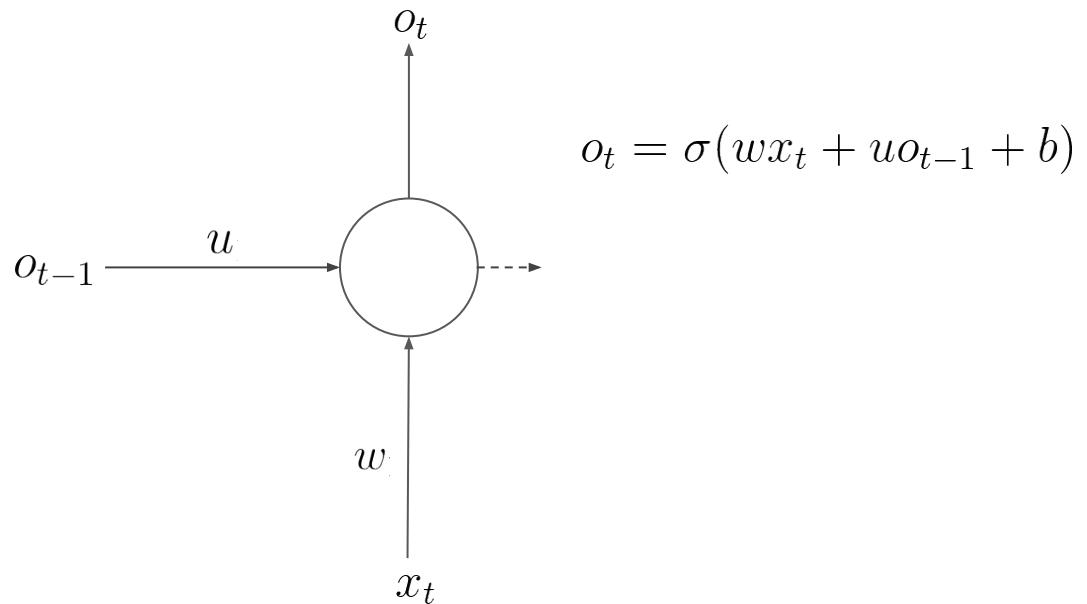
- Time dependent
- Affect the next one
- Share weights



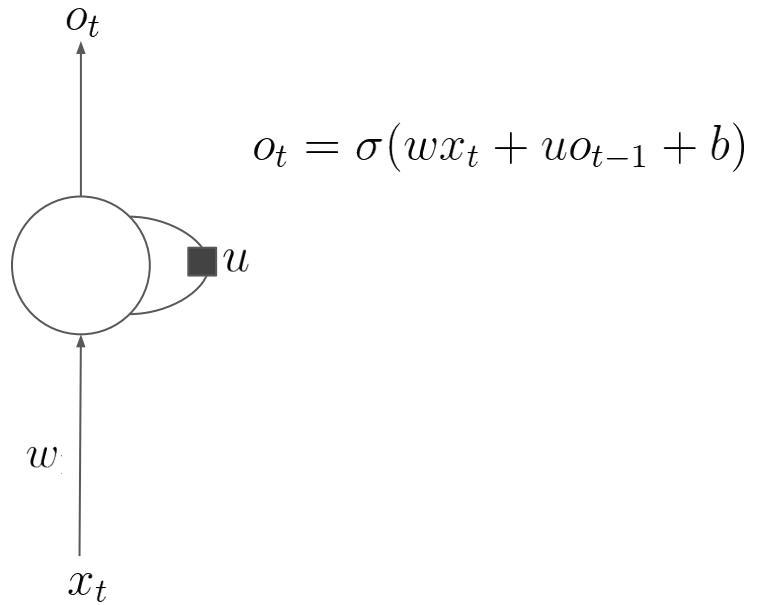
Non-Recurrent Neuron



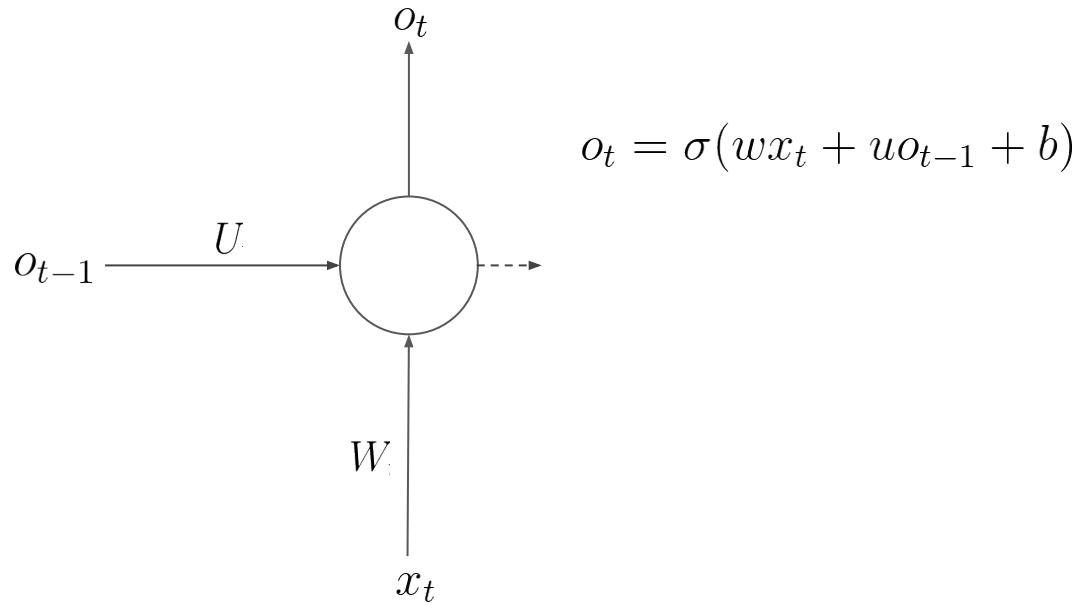
Recurrent Neuron



Recurrent Neuron



Recurrent Neuron

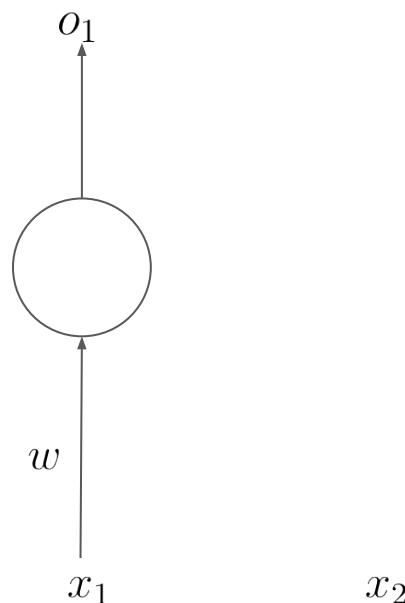


Unfolding Example

x_1

Unfolding Example

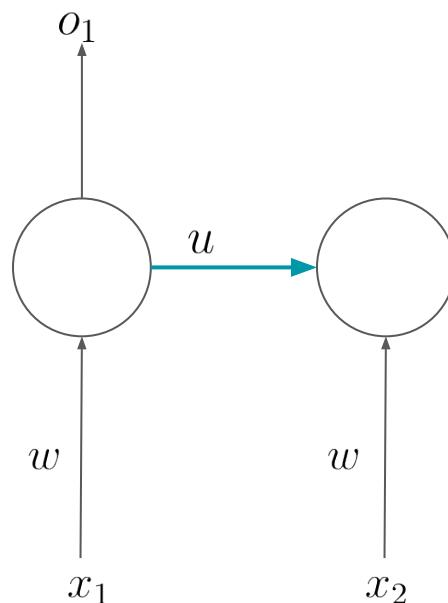
$$o_1 = \sigma(Wx_1 + b)$$



Unfolding Example

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

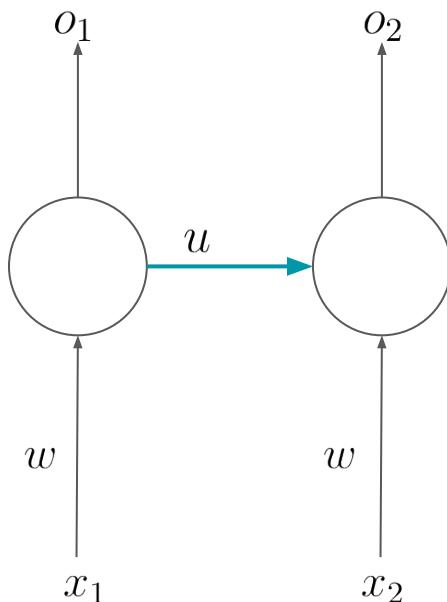
$$Wx_2 + Uo_1 + b$$



Unfolding Example

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

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

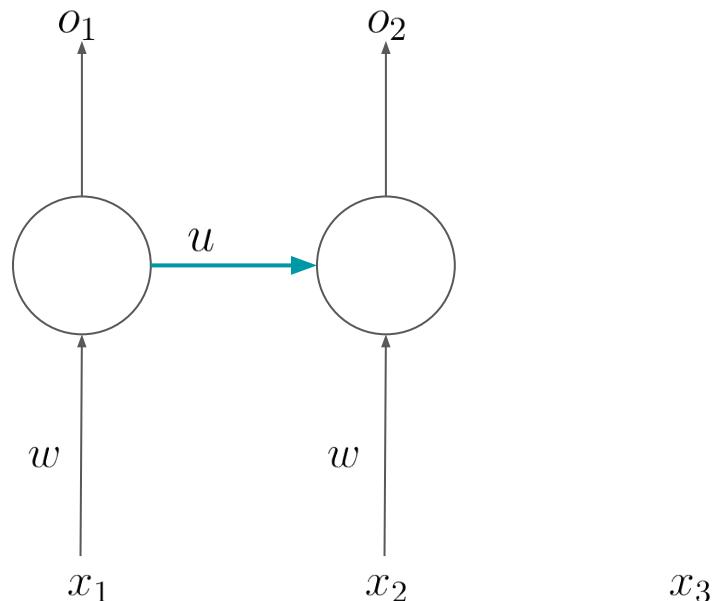


is a lossy summary of input so far. Lossy due to fixed-size vector encoding information about varying sized input. Depending upon training, can learn to discard irrelevant parts.

Unfolding Example

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

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

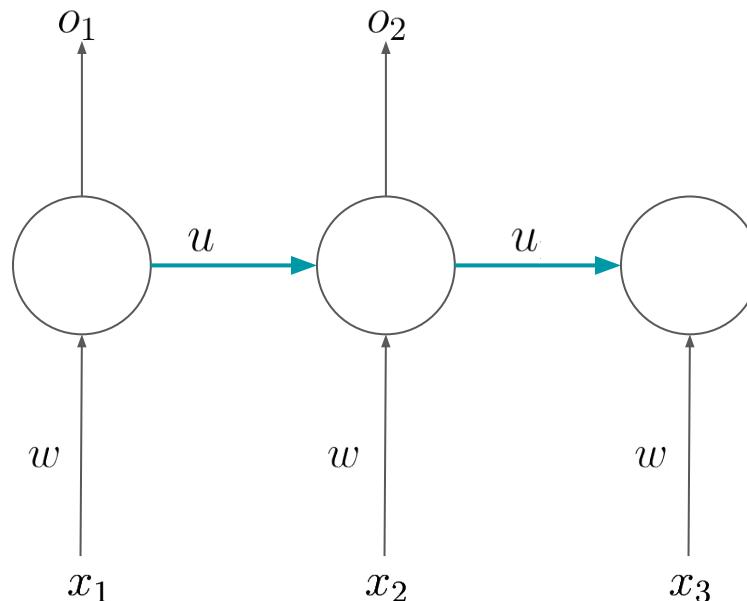


Unfolding Example

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

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

$$Wx_3 + Uo_2 + b$$



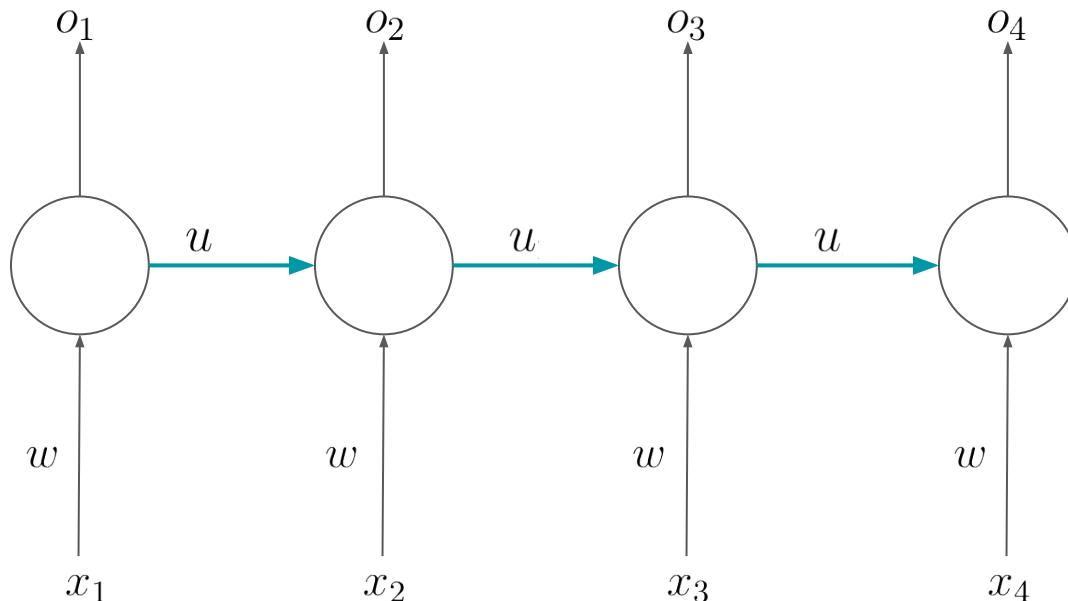
Unfolding Example

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

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

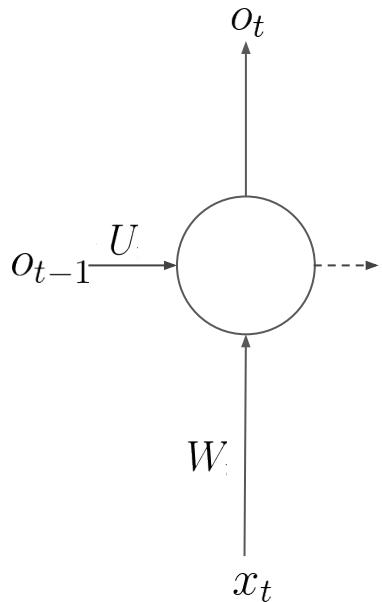
$$o_3 = \sigma(Wx_3 + Uo_2 + b)$$

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



Advantages

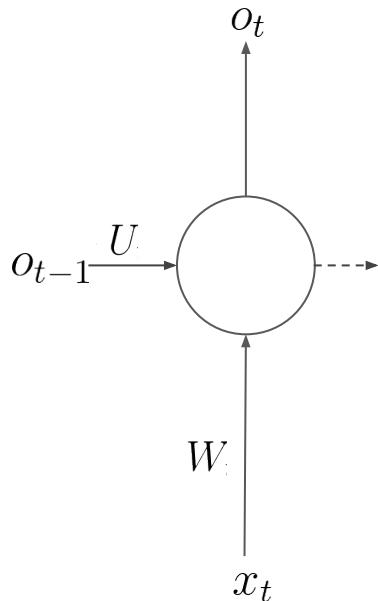
$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



- Specialized for processing sequential input.

Advantages

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

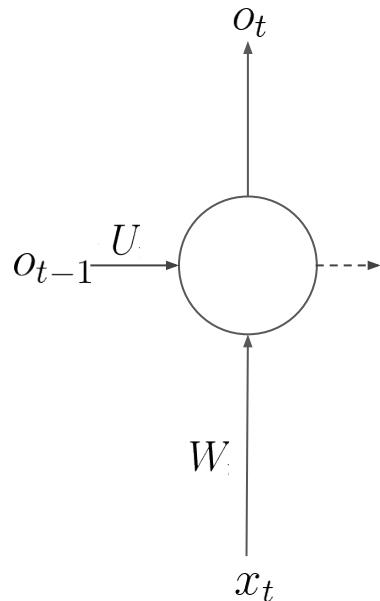


- Specialized for processing sequential input.

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.

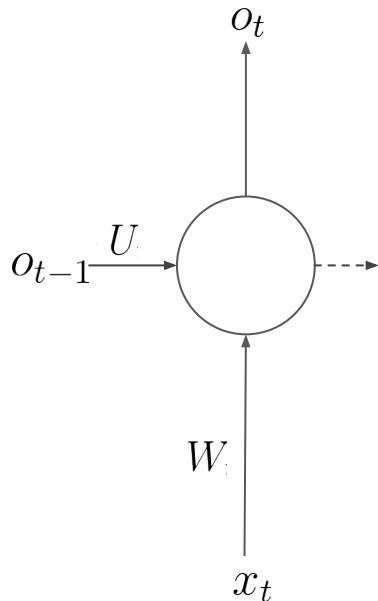
is ready Can differentiate b/w two cases
is not ready while processing “ready”

Can't we use n-grams? **TODO**

- 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

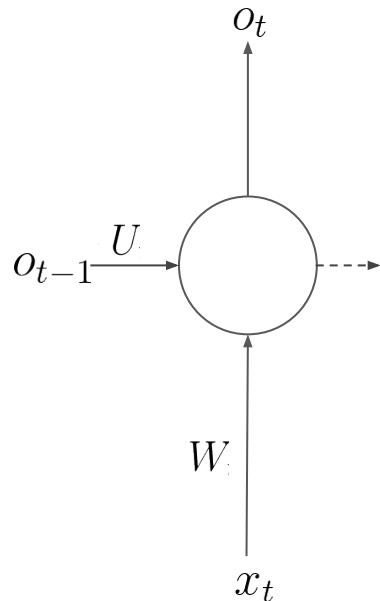
$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



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

Advantages

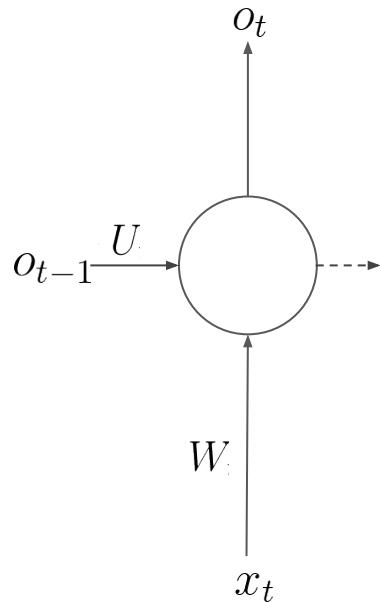
$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



- 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

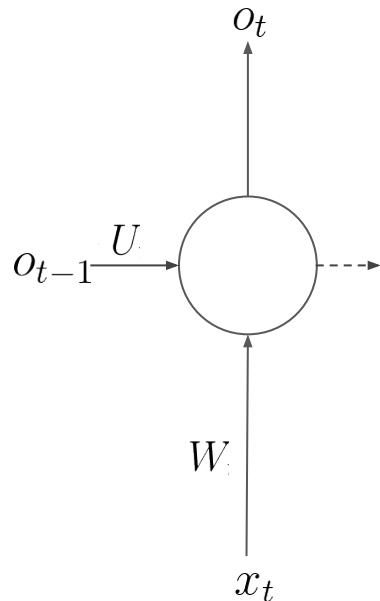
$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



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

Advantages

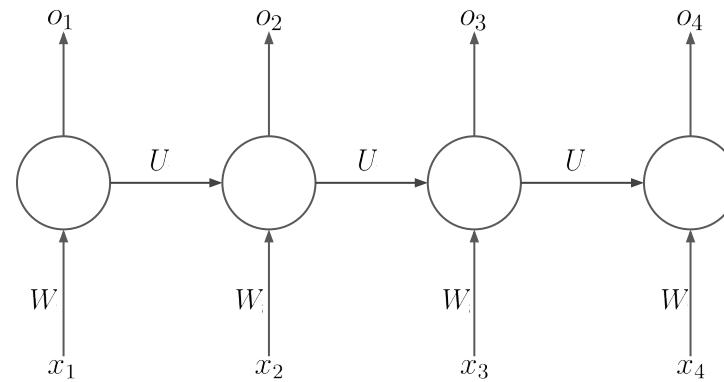
$$o_t = \sigma(wx_t + uo_{t-1} + b)$$



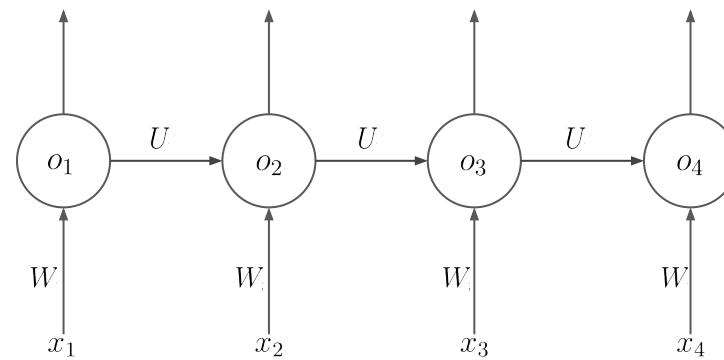
- Specialized for processing sequential input.
- Can generalize for sequences of unseen length
- Reduced parameters
 - Need fewer examples.
 - Less prone to overfit.

Recurrent Networks in the Wild

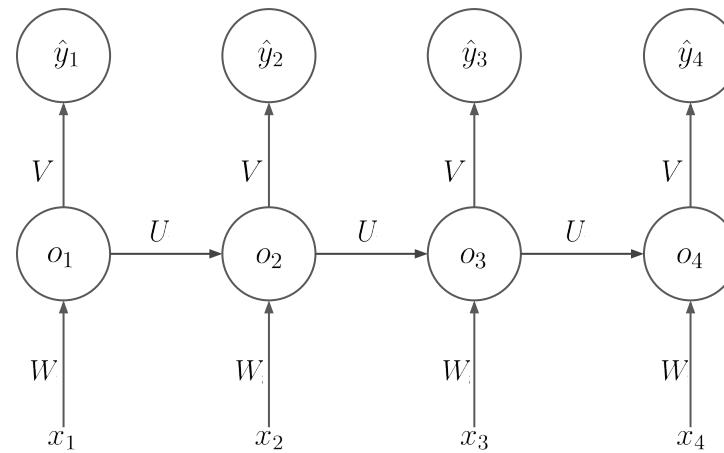
The complete Picture



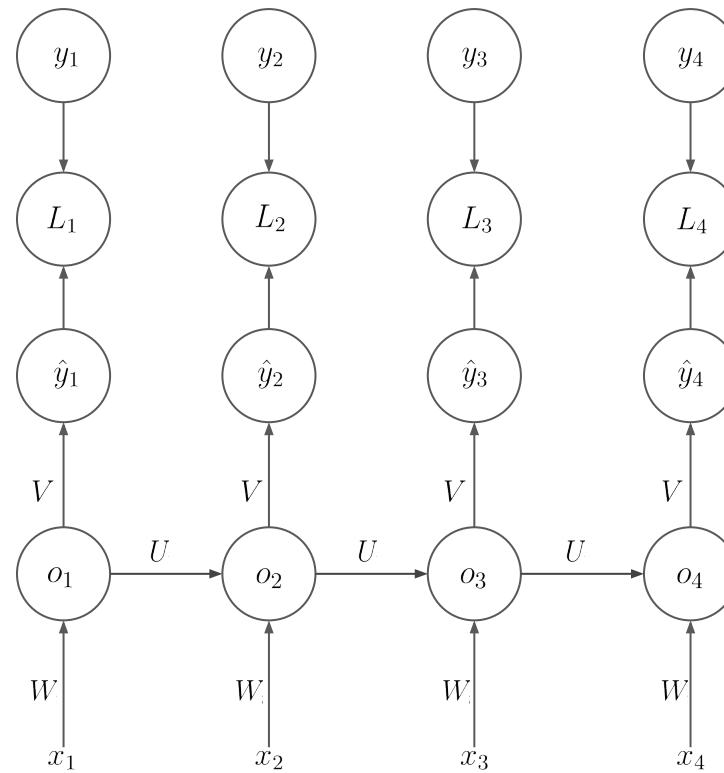
The complete Picture



The complete Picture

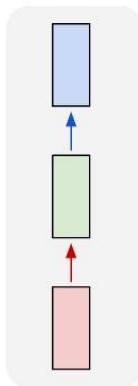


The complete Picture

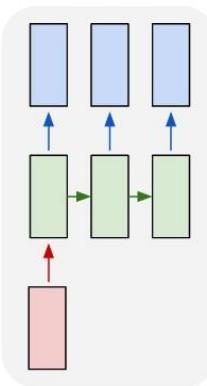


Alternatives

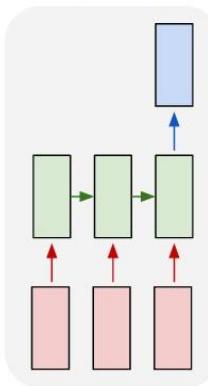
one to one



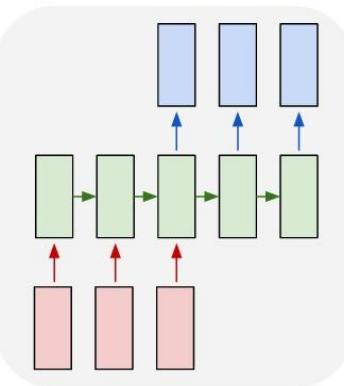
one to many



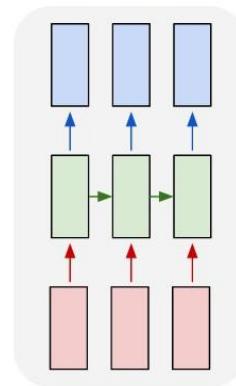
many to one



many to many



many to many

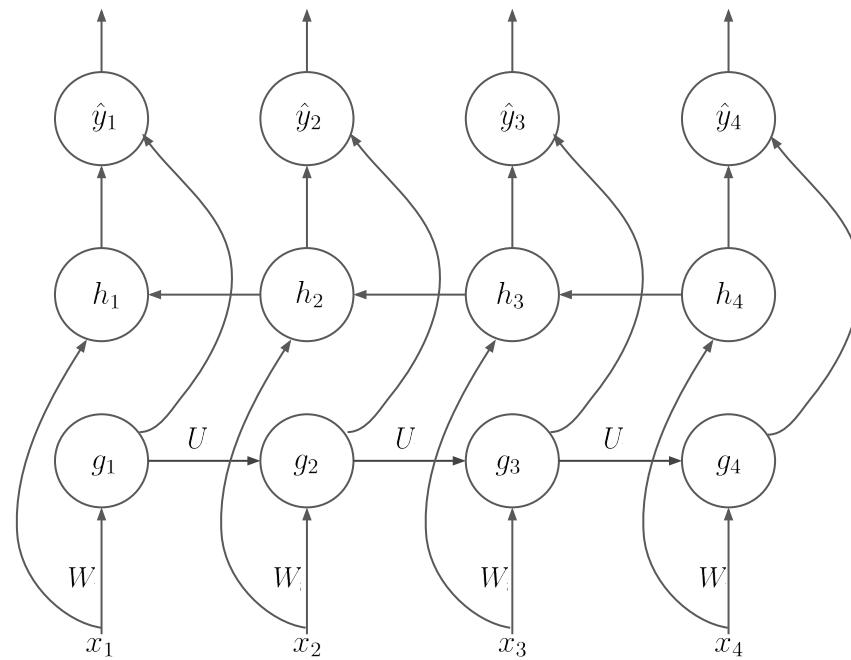


Bidirectional RNN

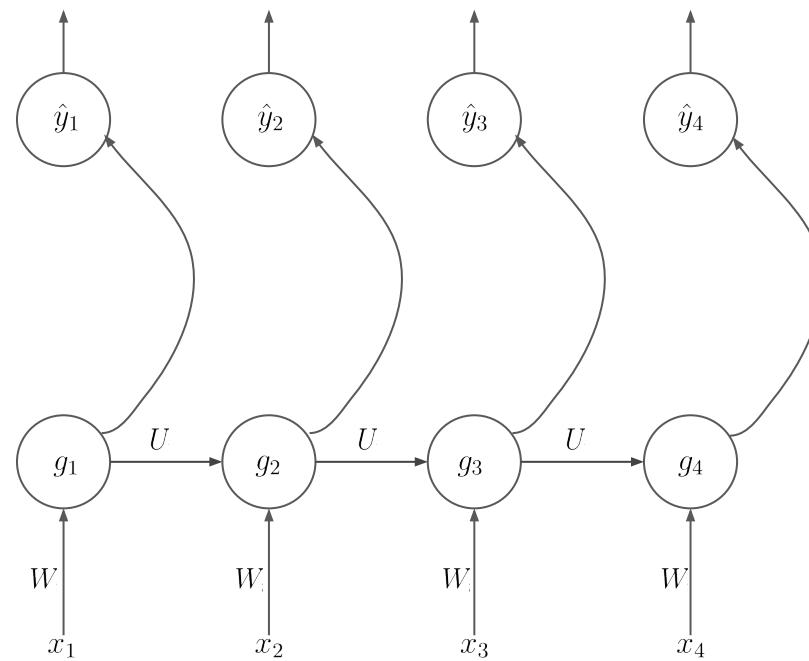
In some applications the information about *next word* might be useful for the current one.

Influence of start of sequence might diminish towards the end.

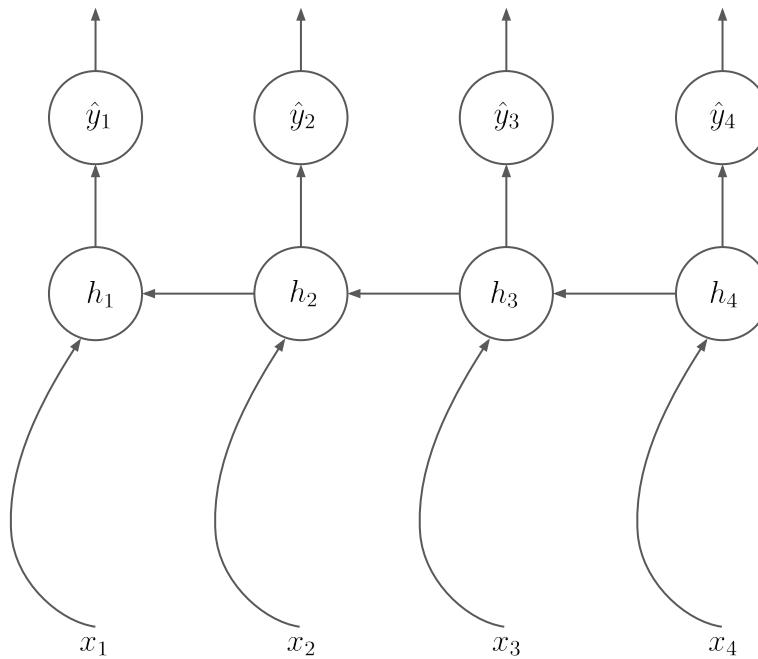
Bidirectional Network



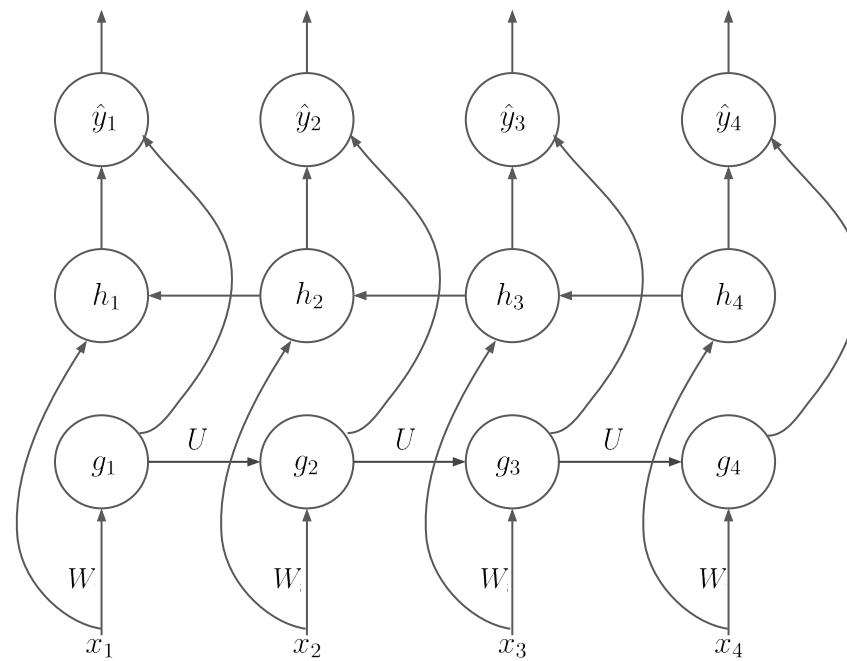
Bidirectional Network



Bidirectional Network



Bidirectional Network

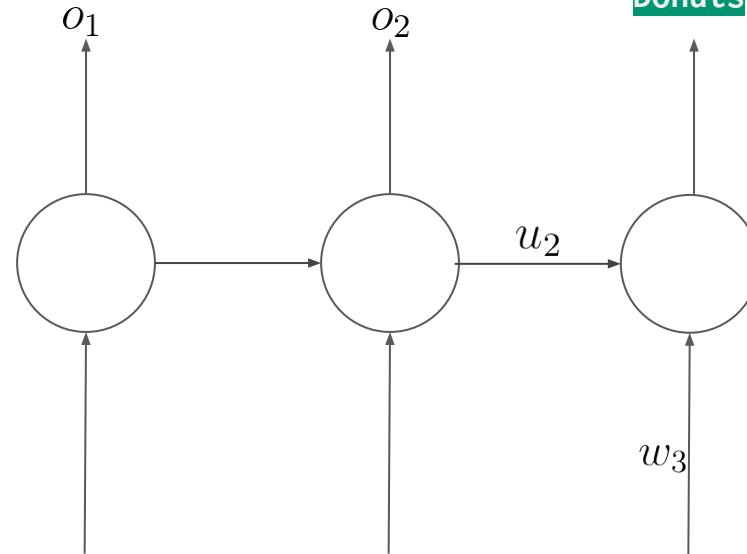




RNN Shortcoming

Humans love **Donuts**

Donuts



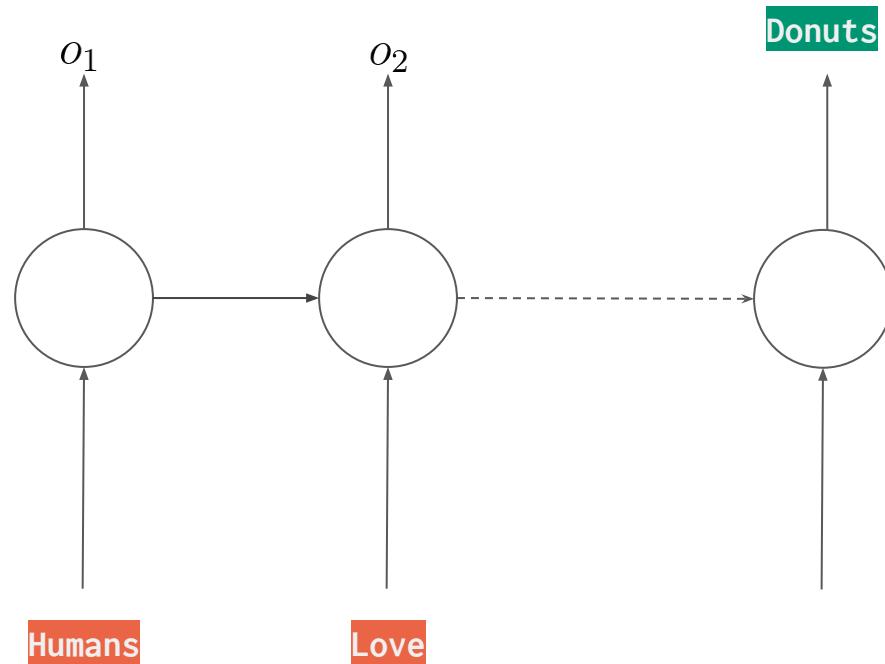
Consider a language model
predicting next word.



Shortcoming

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

Long range information is difficult to handle.

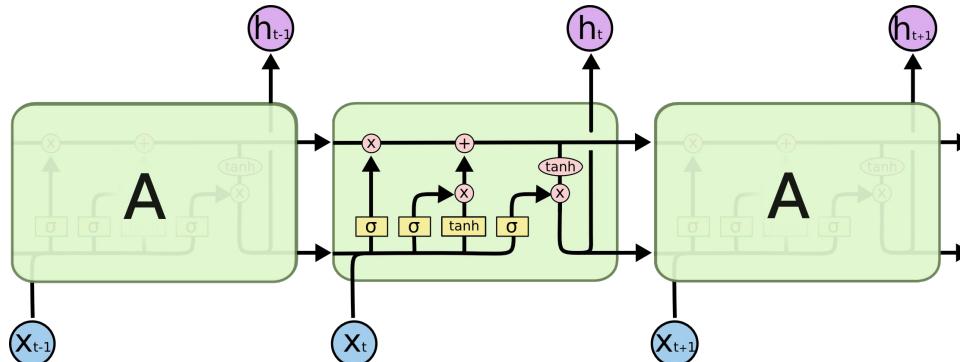


Vanilla RNN shortcomings

- Lossy summary.
- Gradient vanishes while backpropagating across time
→ learning signal to distant past is weak

Long Short Term Memory Cell

- Addresses vanishing gradient of Vanilla RNN
- 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”



References

- Mikolov, Tomas, et al. "**Efficient estimation of word representations in vector space.**" arXiv preprint arXiv:1301.3781 (2013).
- Pennington, Jeffrey, Richard Socher, and Christopher Manning. "**Glove: Global vectors for word representation.**" Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

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



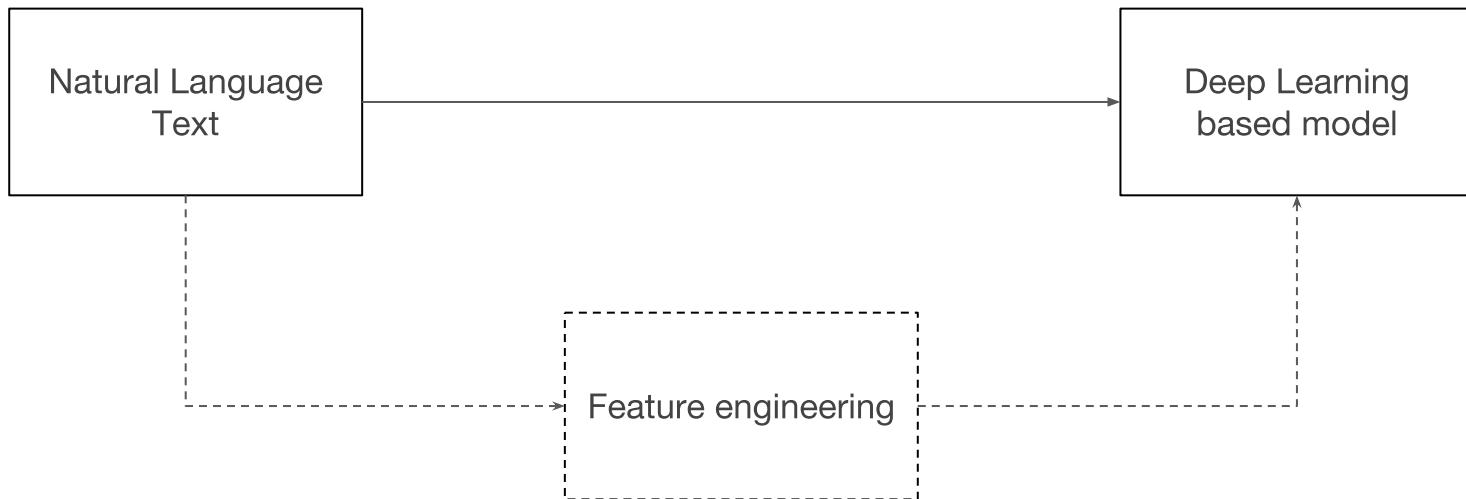
Issues

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

LET ME SPRINKLE SOME

DEEP LEARNING MAGIC

What is “deep learning”?



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