NLP Deep Learning

JADS

ERIK TROMP - FUTURECLUB

Intro

Household announcements

- Who am I
- Course overview
- Objective
- Important, for contact, e-mail me at erik@futureclub.nl or erik@understandling.com
  not on my uni e-mails

Program

- Basics of Deep Learning (for NLP)
- Vectorization models
- Auto-encoders
- Recurrent neural nets
- Recursive neural nets
- LSTMs
- Attention models
- Deep learning for NLP in practice
- t-SNE
- Google Colab

Basics of deep learning

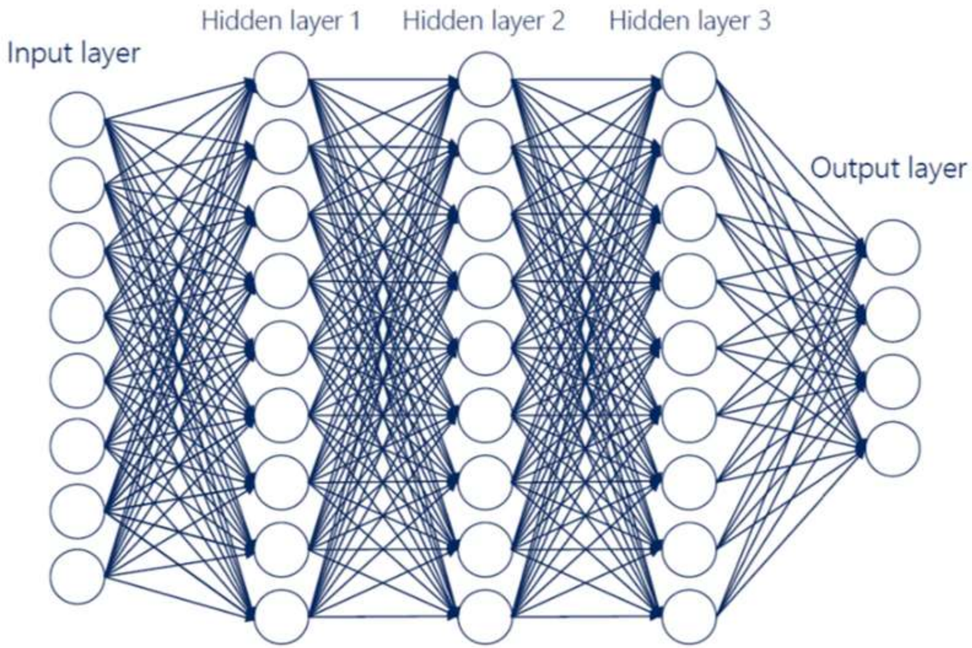
What is deep learning?

Basics of deep learning

Neural networks

Stacked “deepy” – so > x layers

Basics of deep learning



Basics of deep learning

Each layer consists of neurons

But what are those?

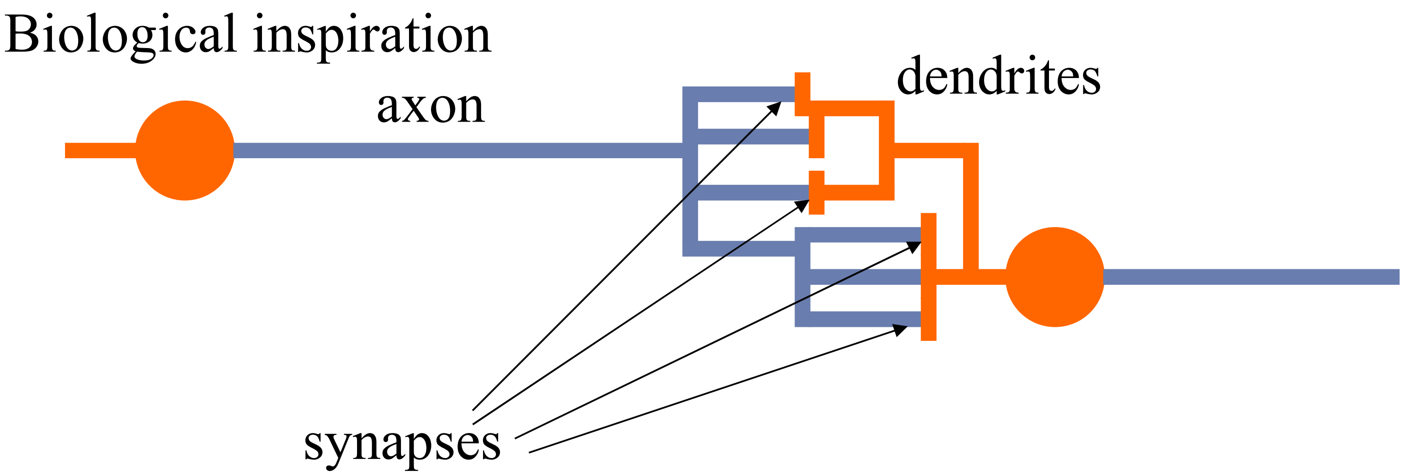
Basics of deep learning

Each layer consists of neurons

But what are those?

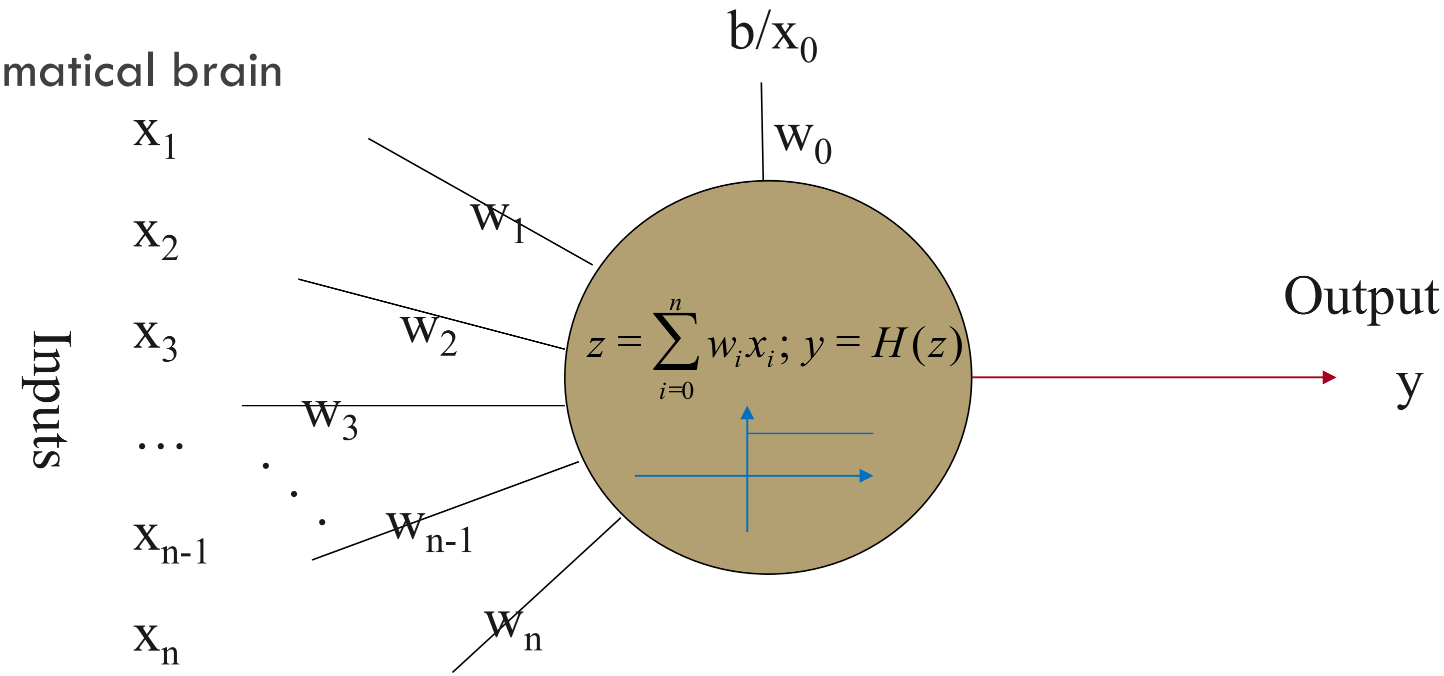
Basics of deep learning

Le brain



Basics of deep learning

Le mathematical brain



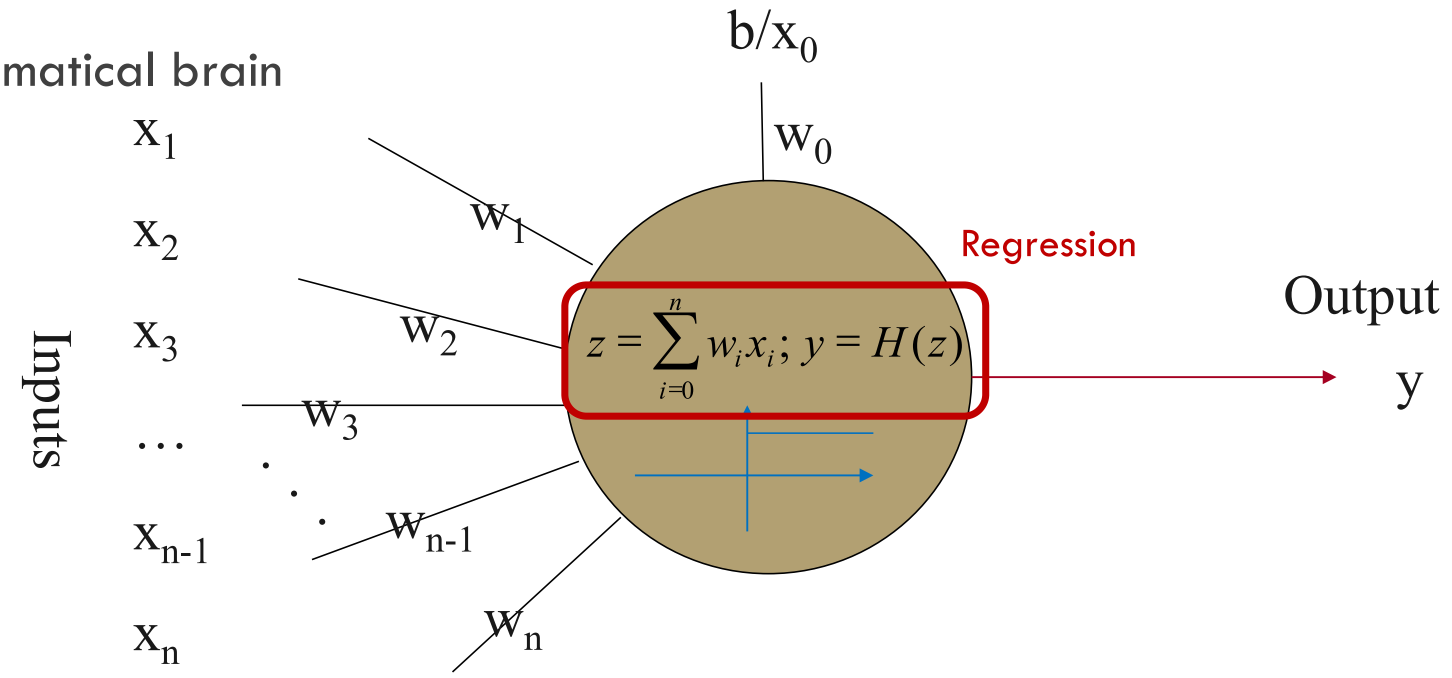
The McCullogh-Pitts model

Basics of deep learning

What does this look like?

Basics of deep learning

Le mathematical brain



The McCullogh-Pitts model

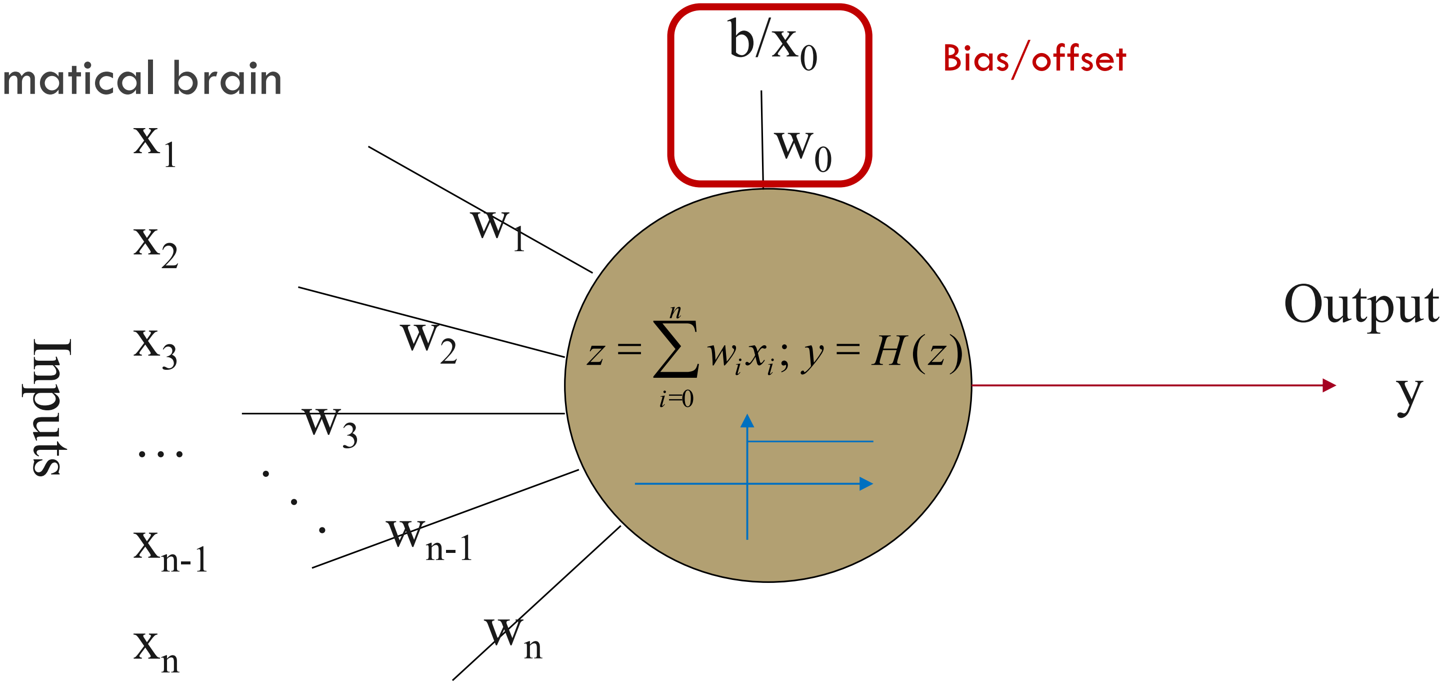
Basics of deep learning

Let’s distill this

FUTURE CLUB

Basics of deep learning

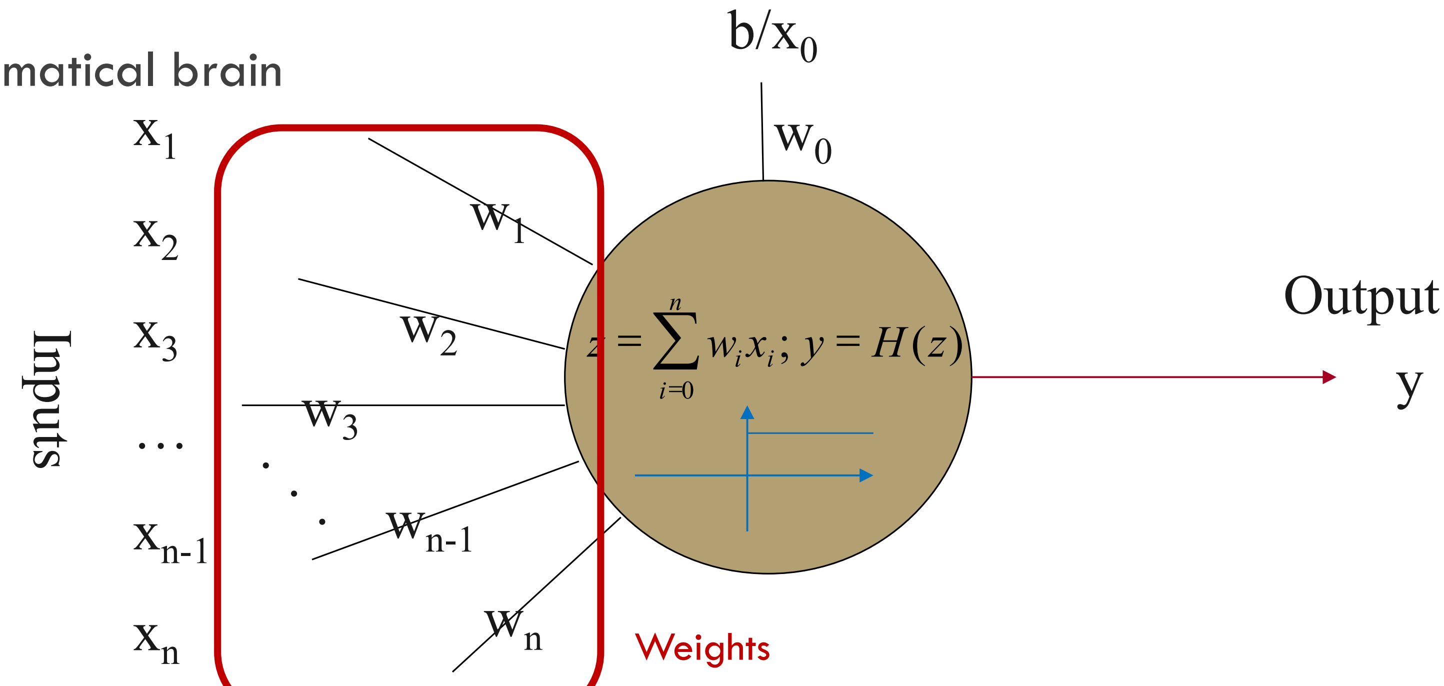
Le mathematical brain



The McCullogh-Pitts model

Basics of deep learning

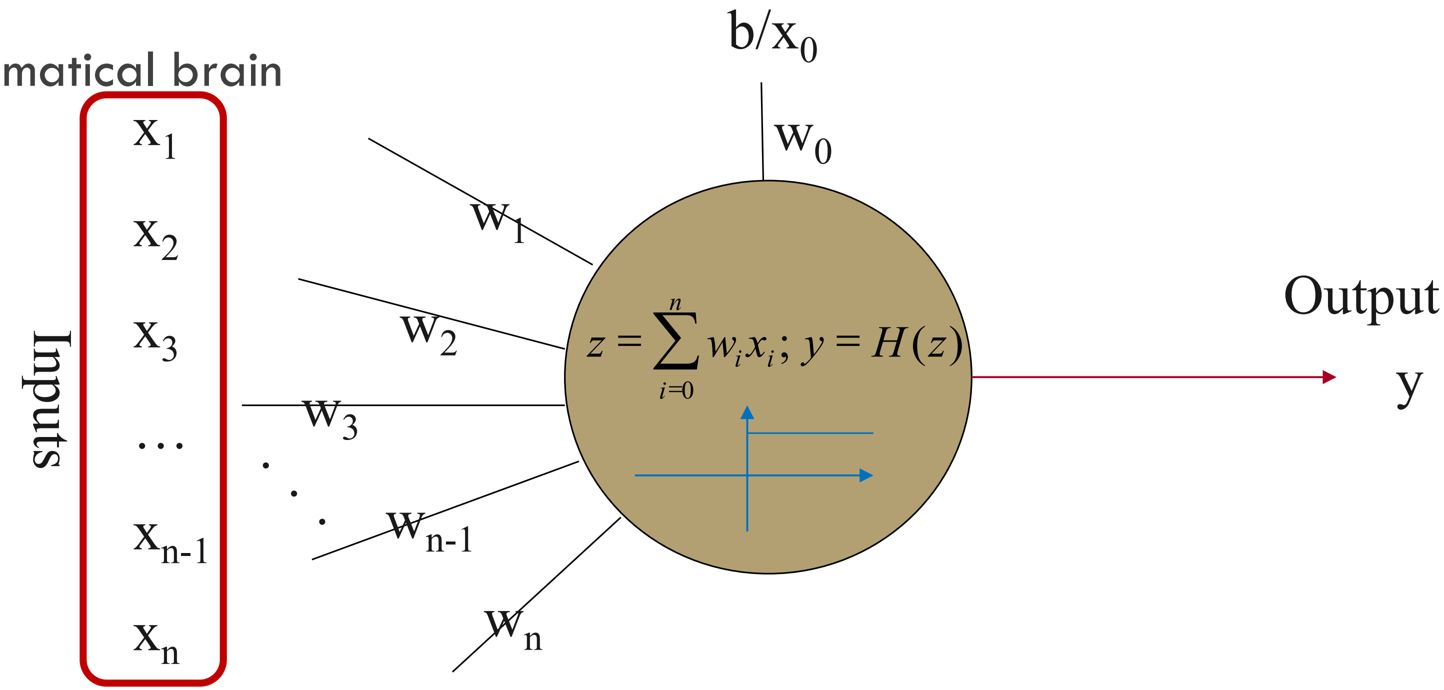
Le mathematical brain



The McCullogh-Pitts model

Basics of deep learning

Le mathematical brain



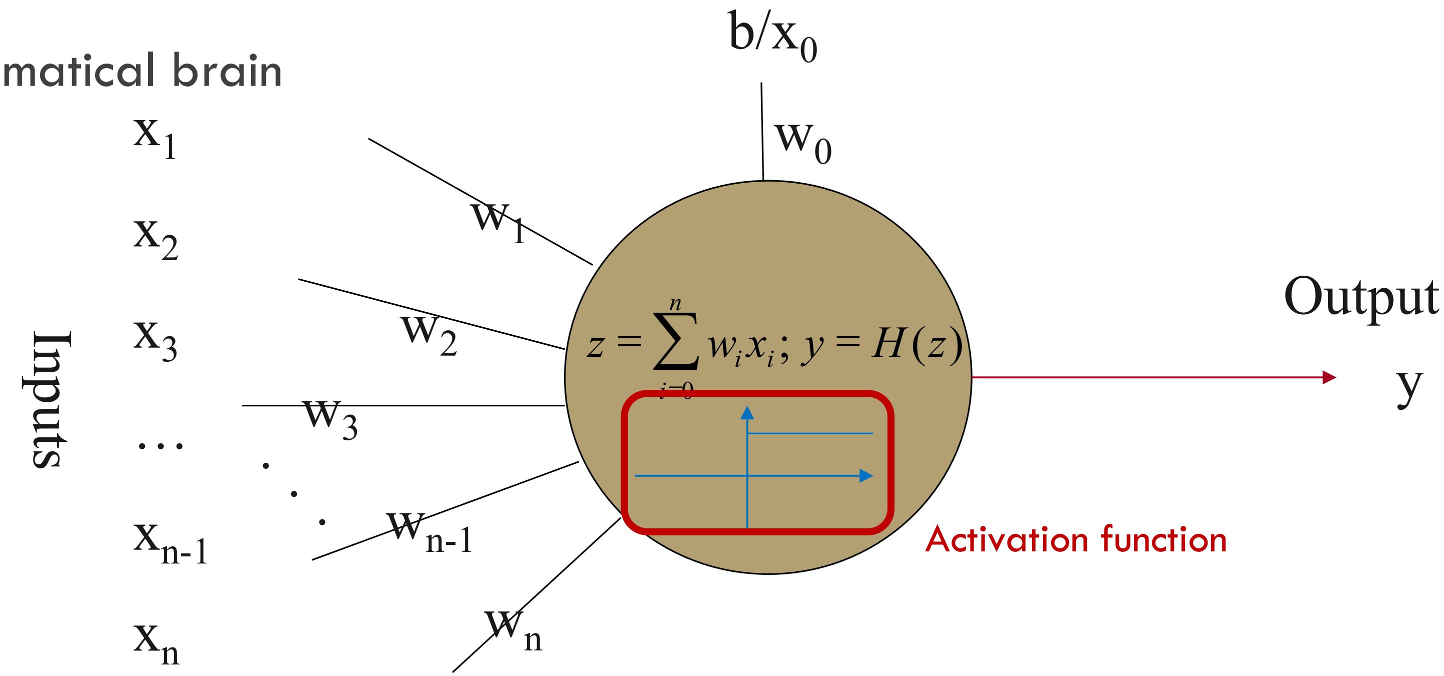
Inputs

The McCullogh-Pitts model

FUTURE CLUB

Basics of deep learning

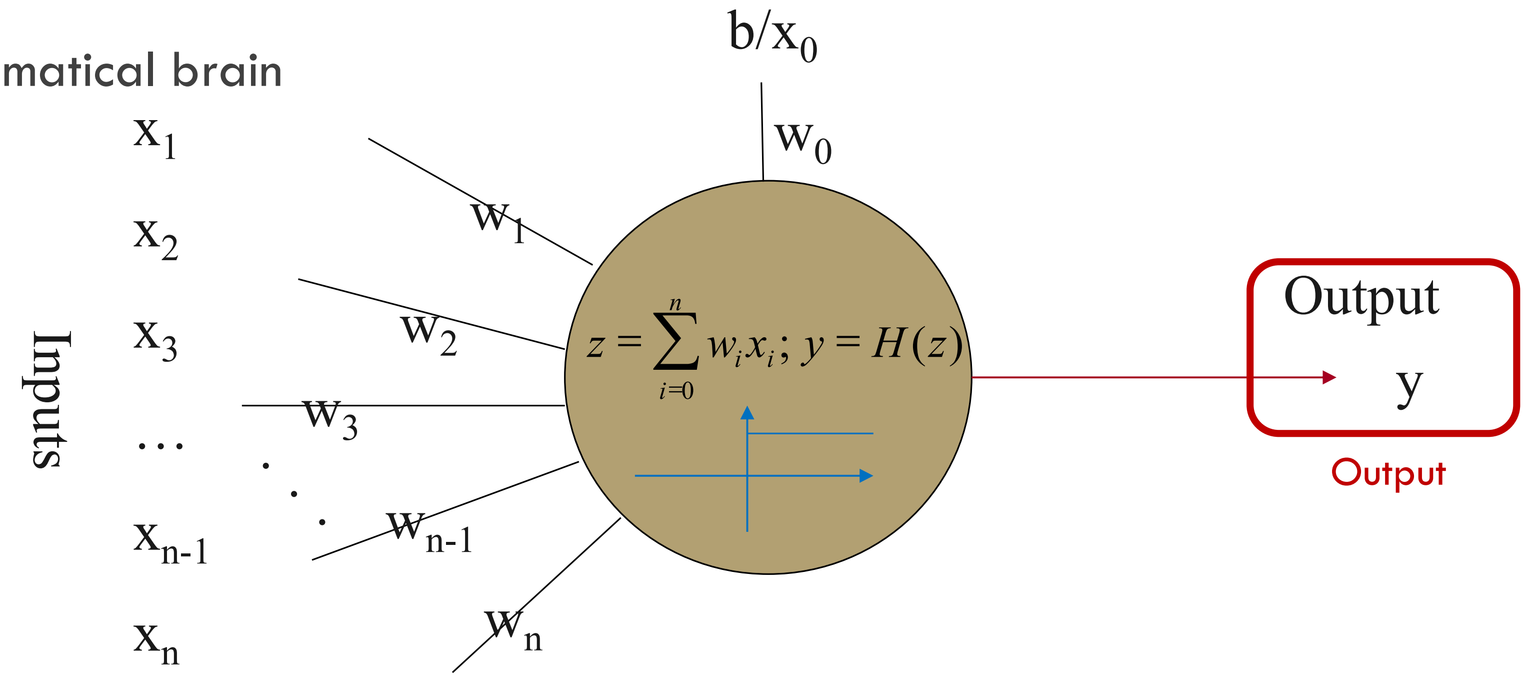
Le mathematical brain



The McCullogh-Pitts model

Basics of deep learning

Le mathematical brain



The McCullogh-Pitts model

Basics of deep learning

Important: these are all continuous numbers

Basics of deep learning

Important: these are all continuous numbers

Sometimes, forcibly, even in [0..1] or even binary {0/1}

Basics of deep learning

But we have text!

Not numbers

Basics of deep learning

So what do we do?

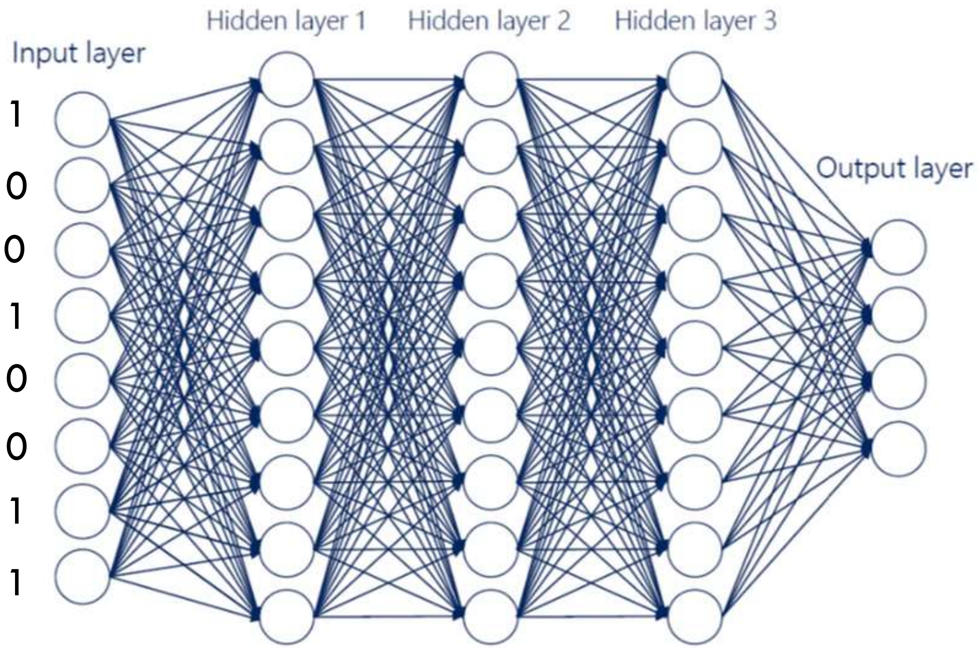
Basics of deep learning

We vectorize the inputs… but how?

Basics of deep learning

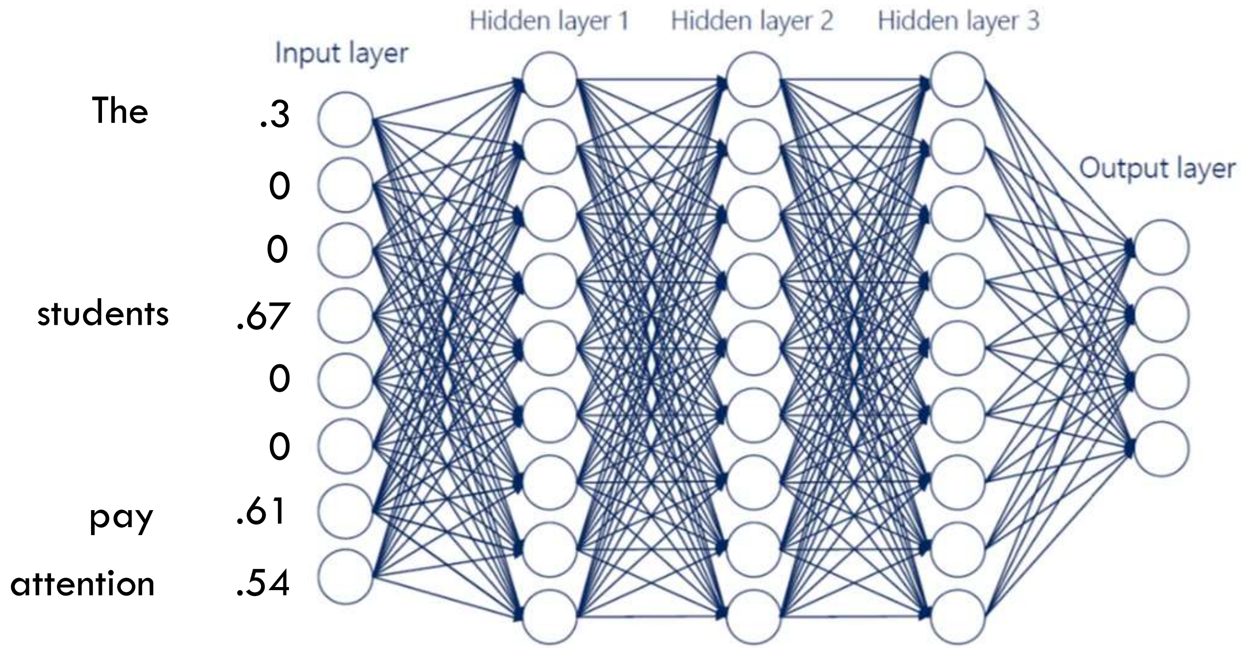
Bag-of-words

The 1
0
0
students 1
0
0
pay 1
attention 1



Basics of deep learning

TF-IDF



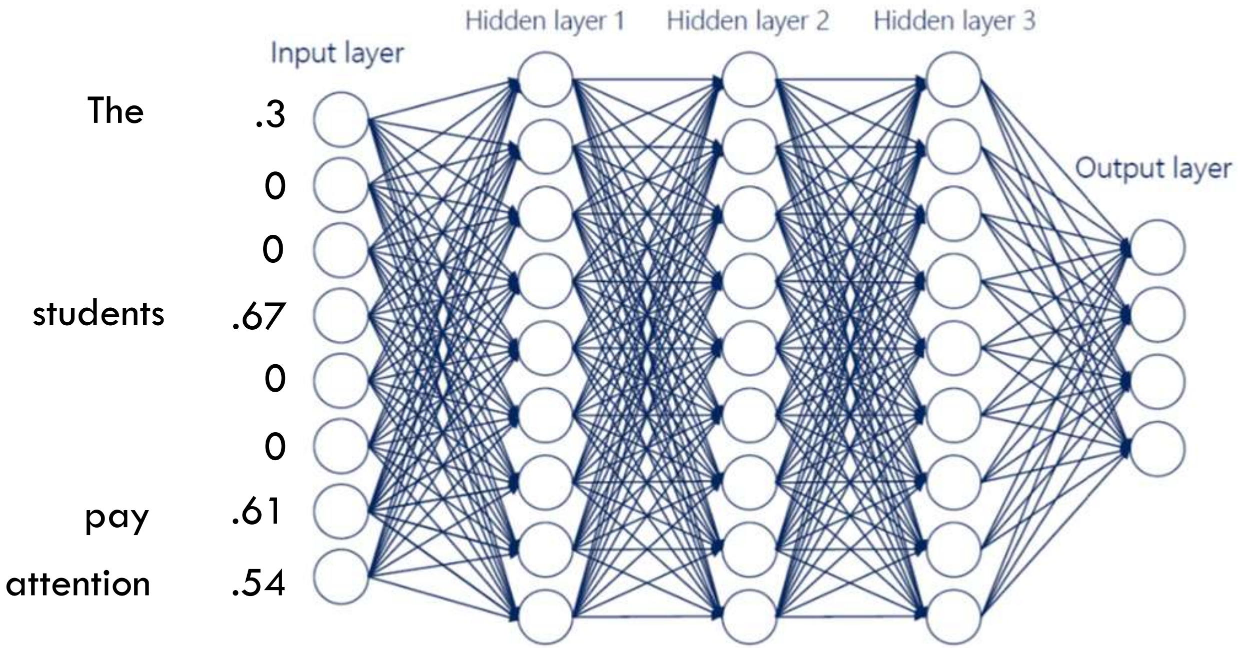
Basics of deep learning

And then we train

using backpropagation

with mini-batches

and gradient (as)descent



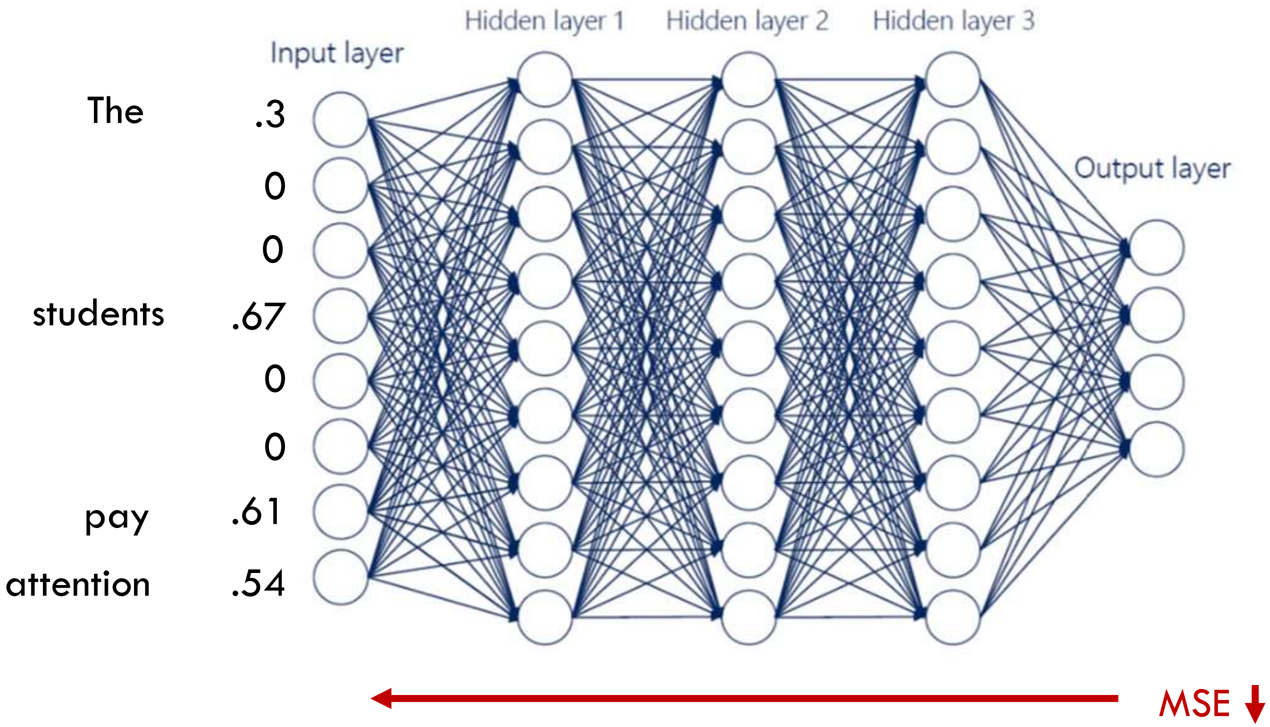
Basics of deep learning

And then we train

using backpropagation

with mini-batches

and gradient (as)descent



Basics of deep learning

But what about order?

Basics of deep learning

What about syntax?

Basics of deep learning

What about semantics?

Basics of deep learning

What about memory?

Basics of deep learning

We need vectorization models!

Basics of deep learning

And some way to store memory in a neural net?!

Program

- ~~Basics of Deep Learning (for NLP)~~
- Vectorization models
- Auto-encoders
- Recurrent neural nets
- Recursive neural nets
- LSTMs
- Attention models
- Deep learning for NLP in practice
- t-SNE
- Google Colab

Vectorization models

How do we go from this?

“dog”

Vectorization models

To this

0.8, -0.2, 0.78, -1.23, 0.98, 0.012, 0.54, -0.32

Vectorization models

And what does this mean?

0.8, $\boxed{-0.2,}$ 0.78, -1.23, 0.98, 0.012, 0.54, -0.32

Vectorization models

We will look into multiple vectorization methods

Word2vec

GloVe

FastText

Doc2Vec

Vectorization models

But we start off with word2vec

Word2vec

T. Mikolov – Google

"Efficient Estimation of Word Representations in Vector Space"

https://arxiv.org/abs/1301.3781

Word2vec

Important note: word2vec is unsupervised!

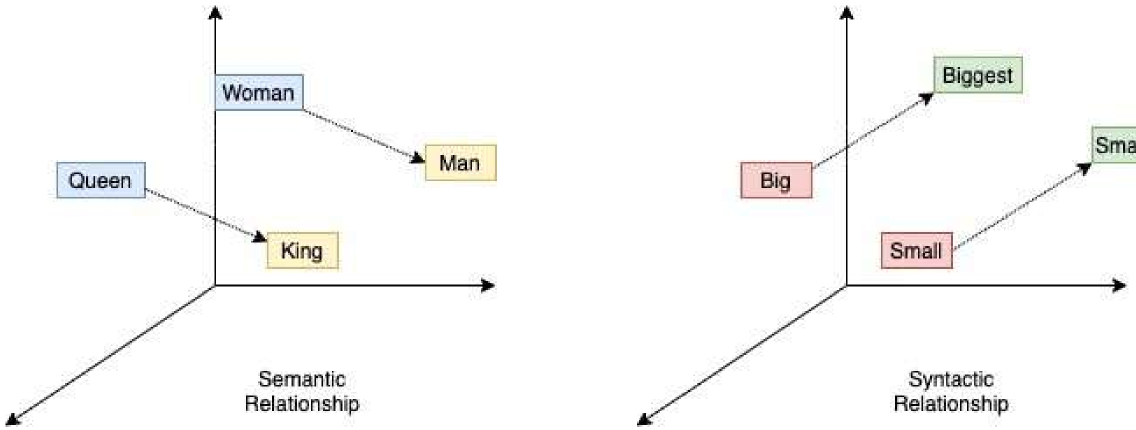
Word2vec

Intuition

1. Start with random word vectors
2. Try to explain a word by its surrounding OR
3. Try to explain a word’s surrounding by that word
4. Update the vectors
5. Rinse & repeat

Word2vec

Result



© Mikolov et al.

Word2vec

Intuition

1. Start with random word vectors
2. Try to explain a word by its surrounding OR
3. Try to explain a word's surrounding by that word
4. Update the vectors
5. Rinse & repeat

Continuous Bag-Of-Words (CBOW)

Skip-gram

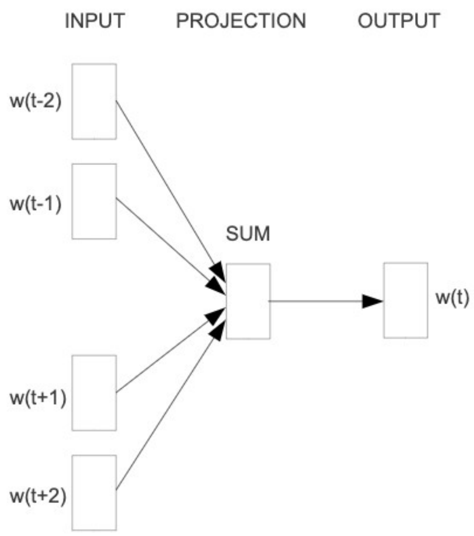
Word2vec

Small sidenote

While very important in deep learning for NLP, word2vec on its own is, in essence, not **deep** learning (it's shallow!)

CBOW

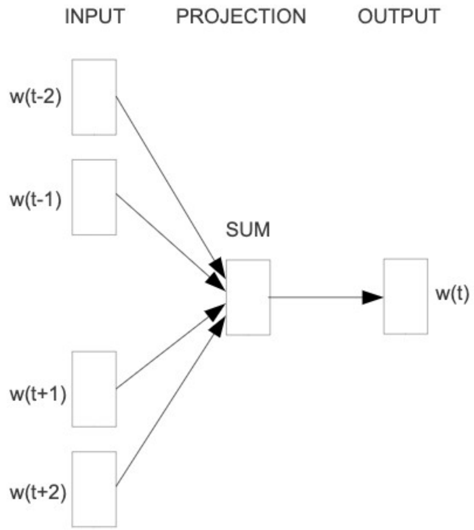
"This course on NLP is awesome"



CBOW

CBOW

"This course on NLP is awesome"



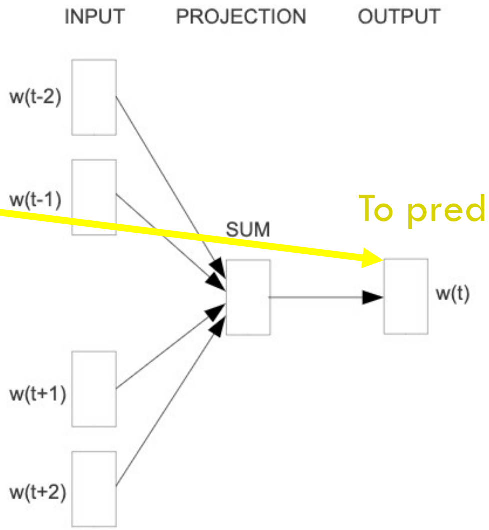
CBOW

CBOW

“This course on NLP is awesome”

Use these as input (concatenated)

To predict this



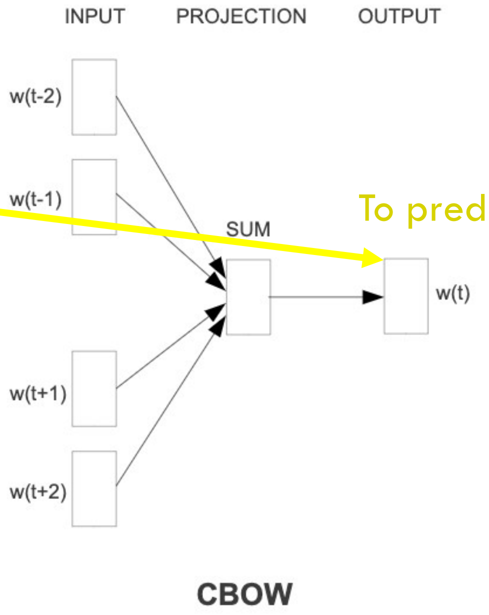
CBOW

CBOW

"This course on NLP is awesome"

Use these as input (concatenated)

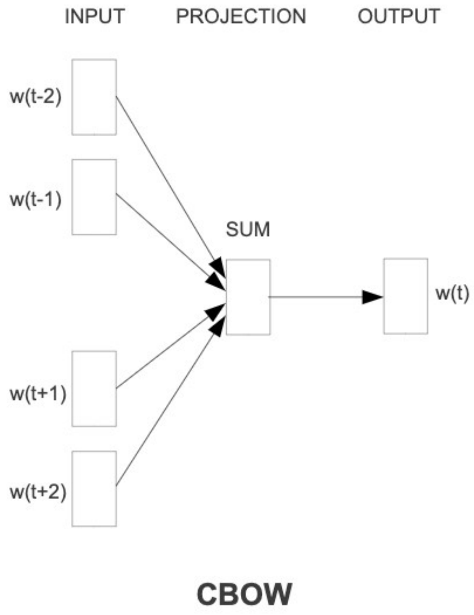
To predict this



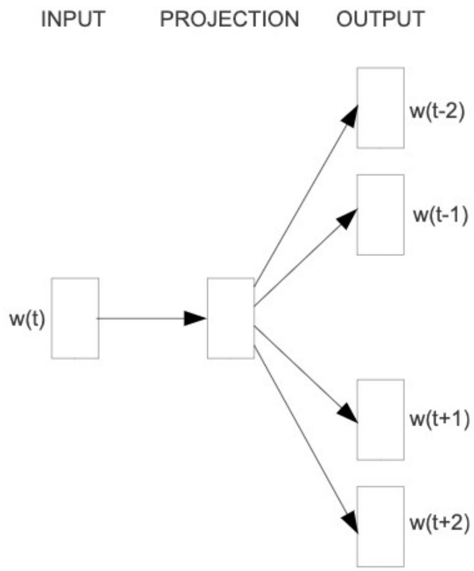
With a softmax
activation function

CBOW

"This course on NLP is awesome"



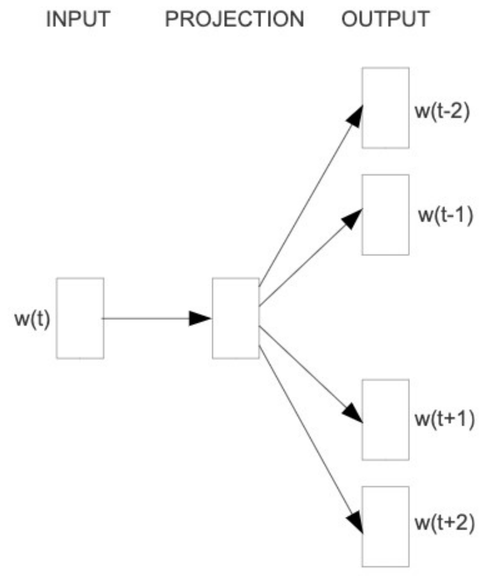
Skip-gram



Skip-gram

Skip-gram

"This course on NLP is awesome"

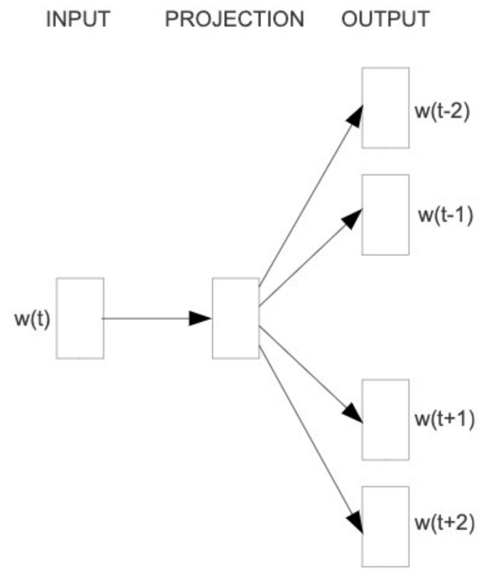


Skip-gram

FUTURE CLUB

Skip-gram

"This course on NLP is awesome"



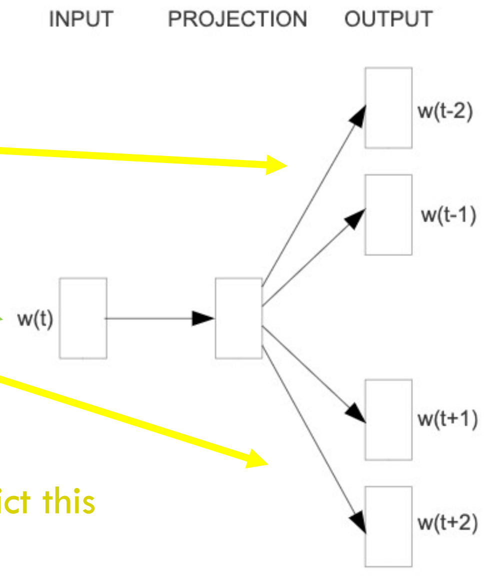
Skip-gram

FUTURE CLUB

Skip-gram

"This course on NLP is awesome"

Use this as input



To predict this

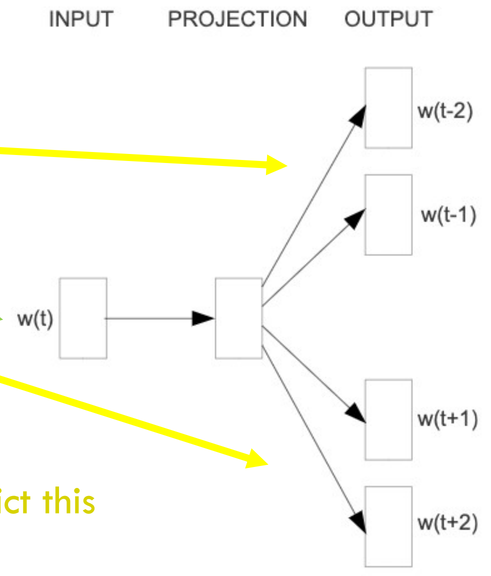
Skip-gram

Skip-gram

"This course on NLP is awesome"

Use this as input

To predict this

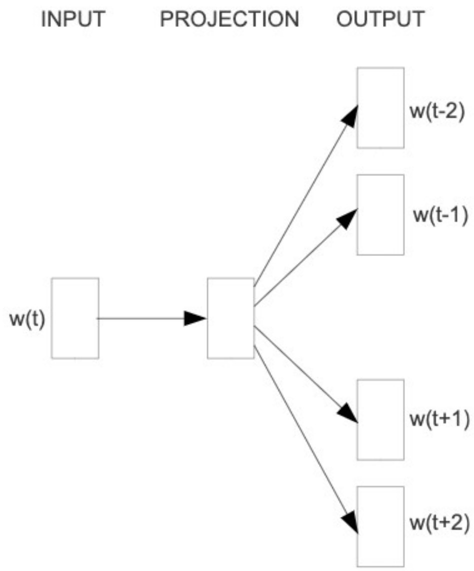


Skip-gram

With a softmax activation function

Skip-gram

"This course on NLP is awesome"



Skip-gram

Word2vec

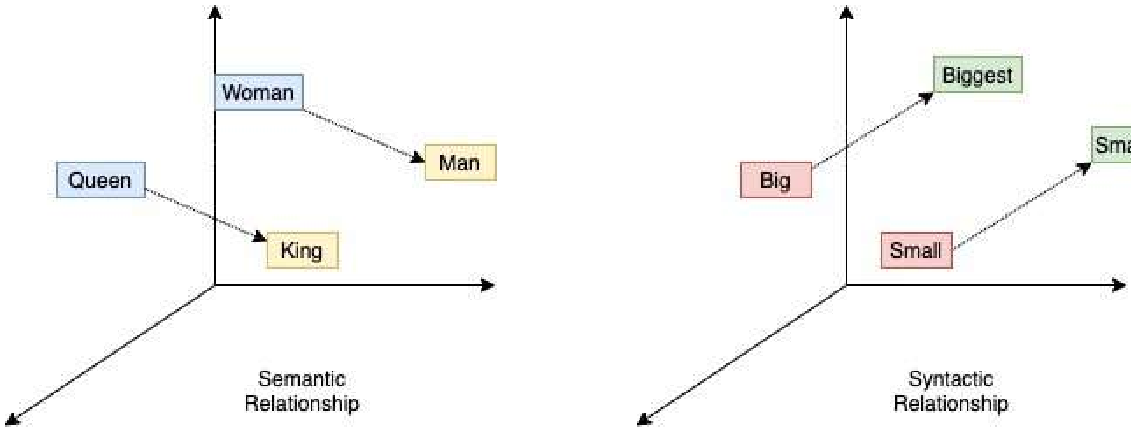
So what is the difference?

(Apart from the inner workings)

Word2vec

Observation 1

Result

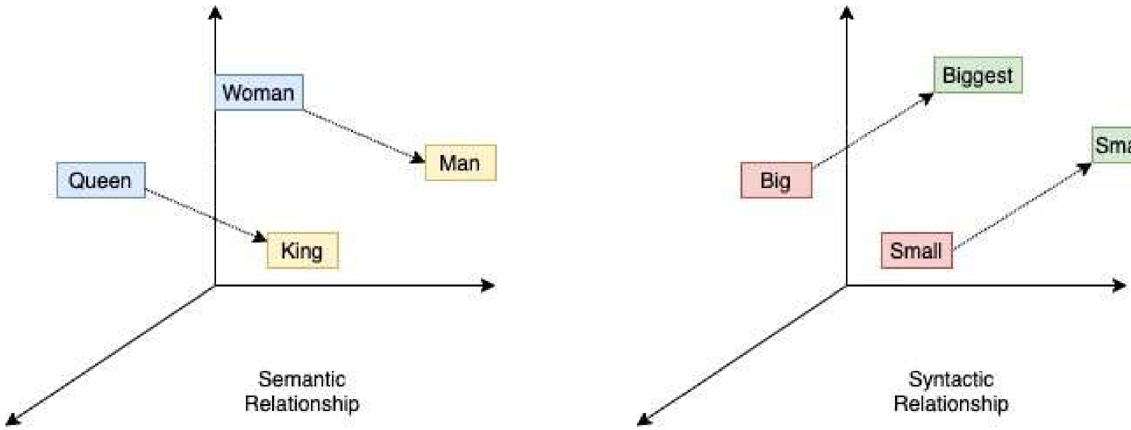


Word2vec

Observation 1

Result

CBOW is better at syntax



Skip-gram is better at semantics

© Mikolov et al.

Word2vec

Observation 2

Since CBOW tries to predict 1 vector from inputs whereas Skip-gram tries to predict multiple vectors from inputs

CBOW is faster… a lot faster

Word2vec

Observation 2.1

To remedy, for Skip-gram we can use **negative sampling:**

Sometimes we “remove” a word, with a probability depending on its frequency

Word2vec

Observation 3

Skip-gram uses single words as inputs, so less overfitting to frequent words (just 1 vector)

CBOW will have many permutations with frequent words

Word2vec

Observation 4

Both methods modify the input vectors!

This is not standard in deep learning

Word2vec

So how do we use this in practice?

Word2vec

Train yourself vs. pre-trained vectors

FUTURE CLUB

Word2vec

Gensim (Python)

See https://rare-technologies.com/word2vec-tutorial/

Word2vec

Gensim (Python)

See https://rare-technologies.com/word2vec-tutorial/

Command-line (C, kinda old)

See https://github.com/tmikolov/word2vec

Word2vec

Gensim (Python)

See https://rare-technologies.com/word2vec-tutorial/

Command-line (C, kinda old)

See https://github.com/tmikolov/word2vec

Many other implementations exist, most combine w2v with GloVe and FastText

Like Gensim, but we'll see more later

Word2vec

Example!

Vectorization models

We will look into multiple vectorization methods

~~Word2vec~~

GloVe

FastText

Doc2Vec

GloVe

Now that we know how word2vec works, the others are easy ☺

GloVe

GloVe

"GloVe: Global Vectors for Word Representation"

From Stanford (Richard Socher, we'll see him later)

https://nlp.stanford.edu/pubs/glove.pdf

GloVe

Main difference with word2vec:

It’s global (duh)

GloVe

Main difference with word2vec:

It’s global (duh)

It looks at word co-occurrences in the entire corpus

GloVe

Hey! This is something that we do in LSA/LDA too!

GloVe

"The cat sat on the mat"

Is <u>the</u> important for <u>cat</u> and <u>mat</u> – or just a random stopword we see frequently?

GloVe

Use a co-occurrence frequency matrix

	the	cat	sat	on	mat
the	0	1	0	1	1
cat	1	0	1	0	0
sat	0	1	0	1	0
on	1	0	1	0	0
mat	1	0	0	0	0

GloVe

And pick two words to look at (fixed)

Then iteratively a third (called the *probe-word*)

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

FUTURE CLUB

GloVe

And pick two words to look at (fixed)

Then iteratively a third (called the *probe-word*)

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

FUTURE CLUB

GloVe

Ice co-occurs more with solid than with gas

Steam co-occurs more with gas than with solid

Both roughly equal with water (related) and fashion (unrelated)

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

GloVe

Ice co-occurs more with solid than with gas

Steam co-occurs more with gas than with solid

Both roughly equal with water (related) and fashion (unrelated)

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

GloVe

Objective

Learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence

GloVe

This results in very strong performance in analogy tasks

GloVe

So how do we use this in practice?

FUTURE CLUB

GloVe

Training yourself (GloVe-Python)

See https://github.com/maciejkula/glove-python

From the authors (C code)

See https://github.com/stanfordnlp/GloVe

Gensim (Python), turns GloVe into w2v format (they are both vectors anyway)

See https://radimrehurek.com/gensim/scripts/glove2word2vec.html

Vectorization models

We will look into multiple vectorization methods

~~Word2vec~~

~~GloVe~~

FastText

Doc2Vec

FastText

FastText

"Enriching Word Vectors with Subword Information"

From Facebook AI Research (FAIR, again T. Mikolov)

https://arxiv.org/abs/1607.04606

FastText

This (luckily) is stupid simple!

FastText

This (luckily) is stupid simple!

Don't just use words and N-grams but also use characters (N-grams)!

FastText

Not much more to say on it other than that

It uses Skip-gram by default

(Because CBOW makes little sense)

	anarchy	chy	<anar	narchy
	monarchy	monarc	chy	<monar
	kindness	ness>	ness	kind
	politeness	polite	ness>	eness>
EN	unlucky	<un	cky>	nlucky
	lifetime	life	<life	time
	starfish	fish	fish>	star
	submarine	marine	sub	marin
	transform	trans	<trans	form

FastText

(Oh and it does doc2vec too)

FastText

But what is really cool about FastText…

FastText

But what is really cool about FastText…

They trained models (word vectors) on 157 languages!

https://fasttext.cc/docs/en/references.html

FastText

So how do we use this in practice?

FastText

Build and use their CLI

https://fasttext.cc/docs/en/references.html

Download one of the pre-trained models

https://fasttext.cc/docs/en/crawl-vectors.html

Use Gensim (Python)

https://radimrehurek.com/gensim/models/fasttext.html

FastText

Example!

Vectorization models

We will look into multiple vectorization methods

~~Word2vec~~

~~GloVe~~

~~FastText~~

Doc2Vec

Doc2Vec

So now we can get vectors for words. Great.

Doc2Vec

How do we tackle this?

"I liked the new Star Wars movie a lot, great plot!", 1

"The plot was average, the actors even worse", -1

"Did you know there is a new Star Wars movie out in cinemas right now?", 0

Doc2Vec

Sliding window?

"I liked the new Star Wars movie a lot, great plot!", 1

"The plot was average, the actors even worse", -1

"Did you know there is a new Star Wars movie out in cinemas right now?", 0

Doc2Vec

Sliding window?

“I liked the new Star Wars movie a lot, great plot!”, 1

“The plot was average, the actors even worse”, -1

“Did you know there is a new Star Wars movie out in cinemas right now?”, 0

Doc2Vec

We can just sum, average or weigh all word vectors, but…

Doc2Vec

What about long-term dependencies?

Doc2Vec

What about the labels?

Doc2Vec

We can use doc2vec!

FUTURE CLUB

Doc2Vec

"Distributed Representations of Sentences and Documents"

Quoc Le, T. Mikolov (again!)

https://arxiv.org/abs/1405.4053

Doc2Vec

Note: this is the same thing as paragraph vectors (paragraph2vec) and sentence vectors (sent2vec, sentence2vec)

But **NOT** the same as sense2vec

Still with me? ☺

Doc2Vec

The idea is **super** simple:

Use word2vec but assign a paragraph (or sentence) ID to the word vectors and model that too

Doc2Vec

Note that this ID can be a class label (supervised) or a unique one (unsupervised)

Doc2Vec

We then have 2 models:

Distributed memory model (PV-DM)

Distributed bag of words (PV-DBOW)

Doc2Vec

We then have 2 models:

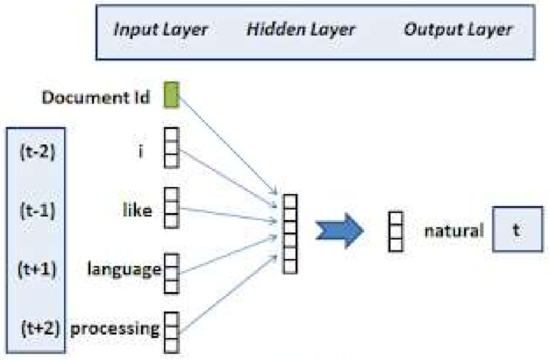
Continuous Bag-Of-Words (CBOW)

Distributed memory model (PV-DM)

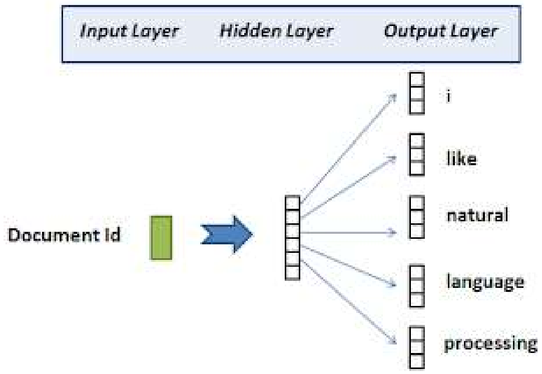
Distributed bag of words (PV-DBOW)

Skip-gram

Doc2Vec



Distributed Memory Model



Distributed Bag of Words Model

Doc2Vec

Note that nowadays, there are many variations to this paradigm

Doc2Vec

So how do we use this in practice?

Doc2Vec

Use Gensim (Python)

https://radimrehurek.com/gensim/models/doc2vec.html

Use FastText (C code, CLI)

https://github.com/facebookresearch/fastText#text-classification

Use Transformers – we will see how later

Program

- ~~Basics of Deep Learning (for NLP)~~
- ~~Vectorization models~~
- Auto-encoders
- Recurrent neural nets
- Recursive neural nets
- LSTMs
- Attention models
- Deep learning for NLP in practice
- t-SNE
- Google Colab

Auto-encoders

Question

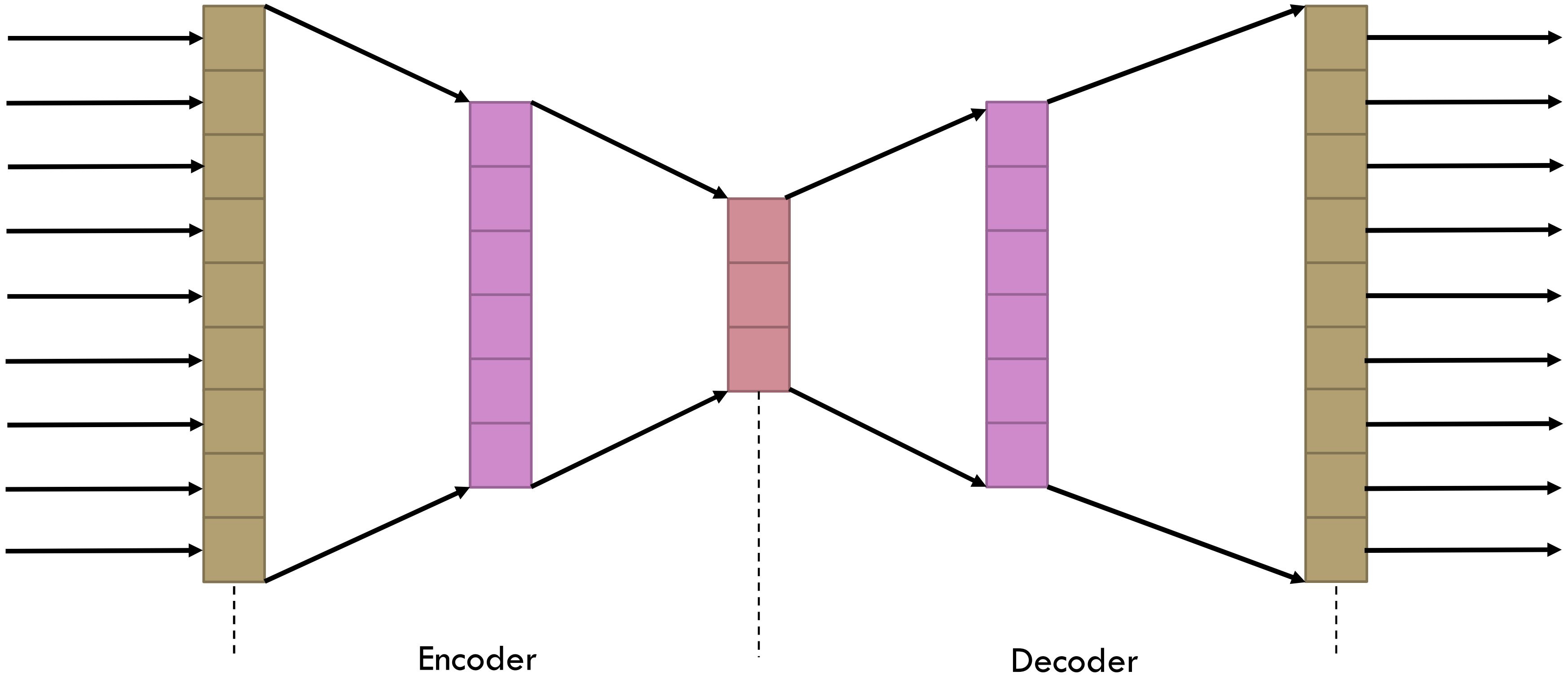
What are auto-encoders?

Auto-encoders

Question

What can we use them for?

Auto-encoders



Auto-encoders

These things are used for feature-reduction

Also of word vectors!

Auto-encoders

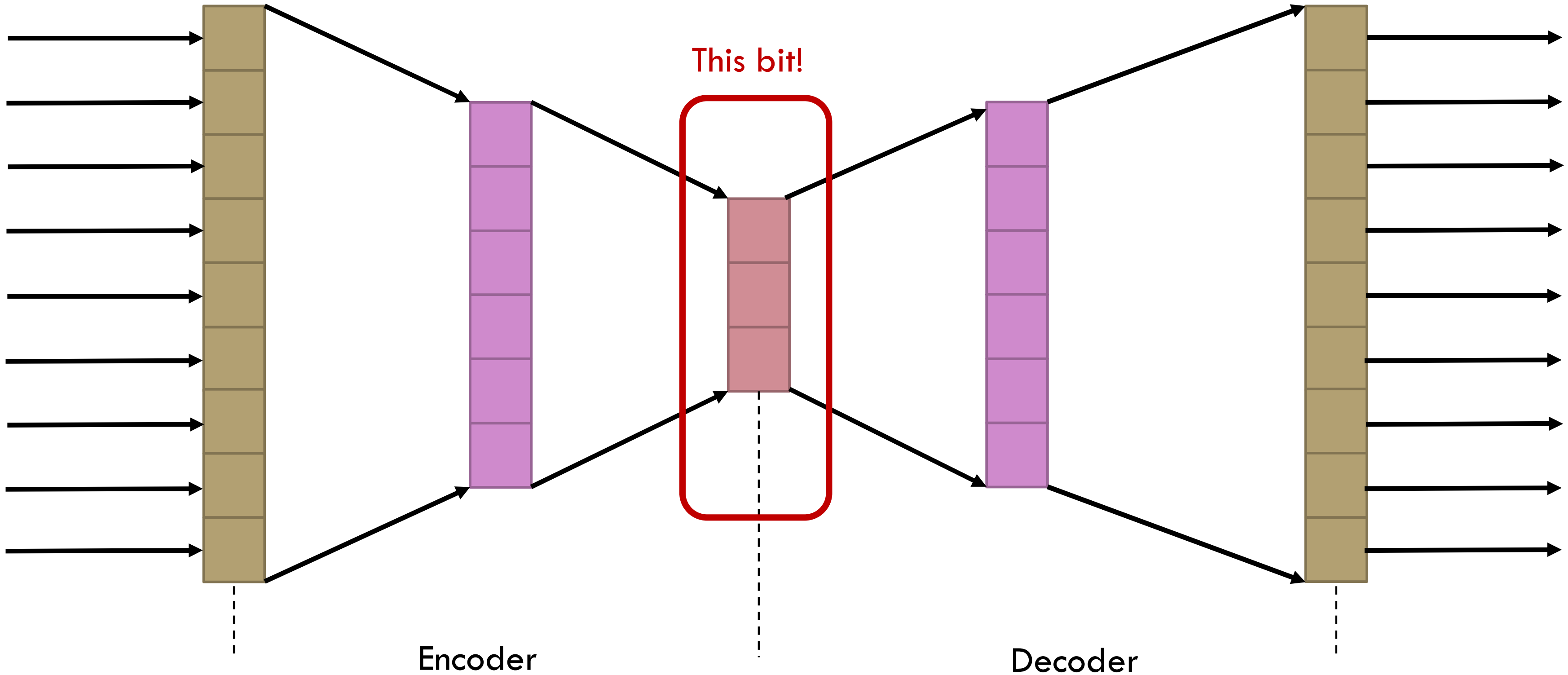
Surprisingly, they improve word vector performance

Regardless of the vectorization method used

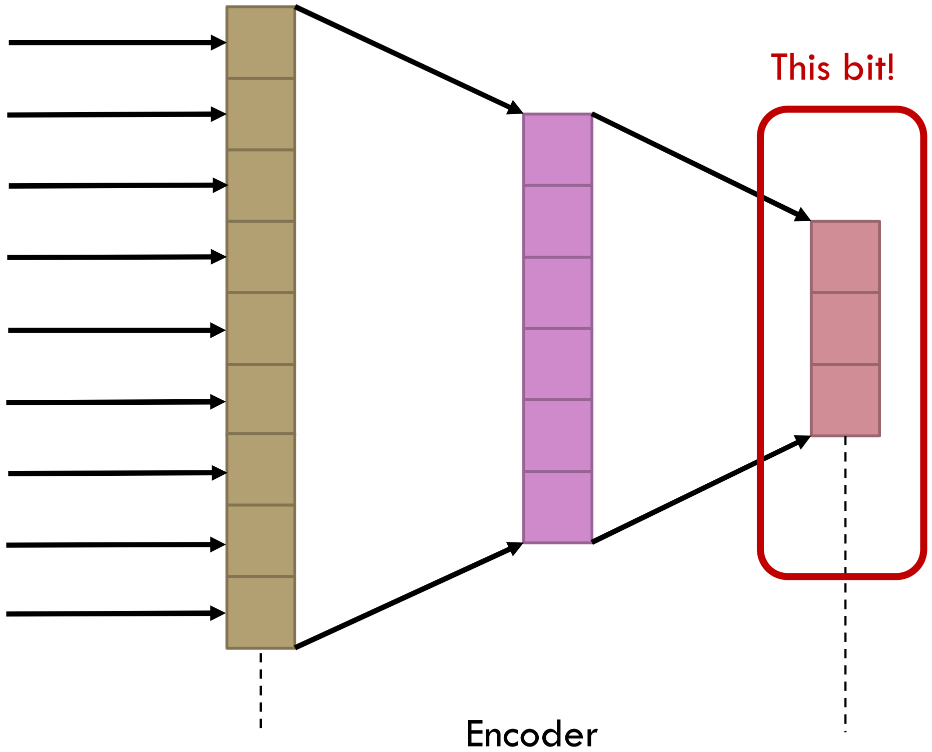
Auto-encoders

But you can also do topic modeling with this

Auto-encoders



Auto-encoders



Auto-encoders

This is the basics

You can stack them, one AE on top of another

Use denoisizing, variational, etc.

Auto-encoders

So how do we use this in practice?

Auto-encoders

Basically, pick your deep learning framework and use the layers present

My favourite is Keras – this is an auto-encoder:

```
import keras
from keras import layers

encoding_dim = 32

input_img = keras.Input(shape=(784,))
encoded = layers.Dense(encoding_dim, activation='relu')(input_img)
decoded = layers.Dense(784, activation='sigmoid')(encoded)
autoencoder = keras.Model(input_img, decoded)
```

Auto-encoders

A note

Auto-encoders vs word2vec

Auto-encoders

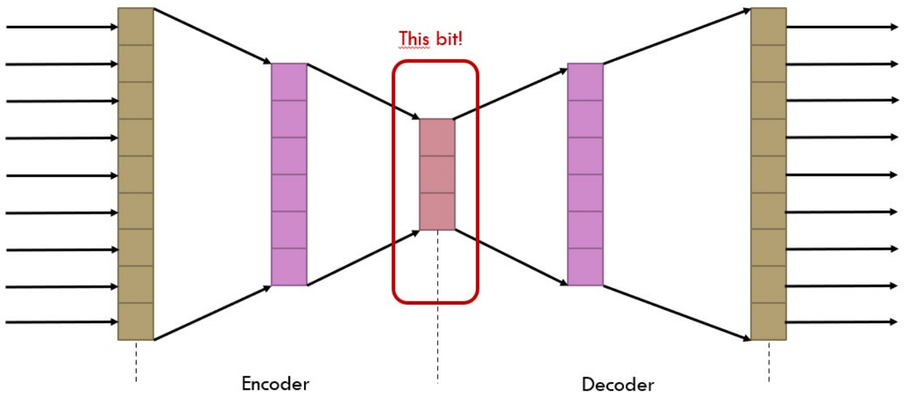
A note

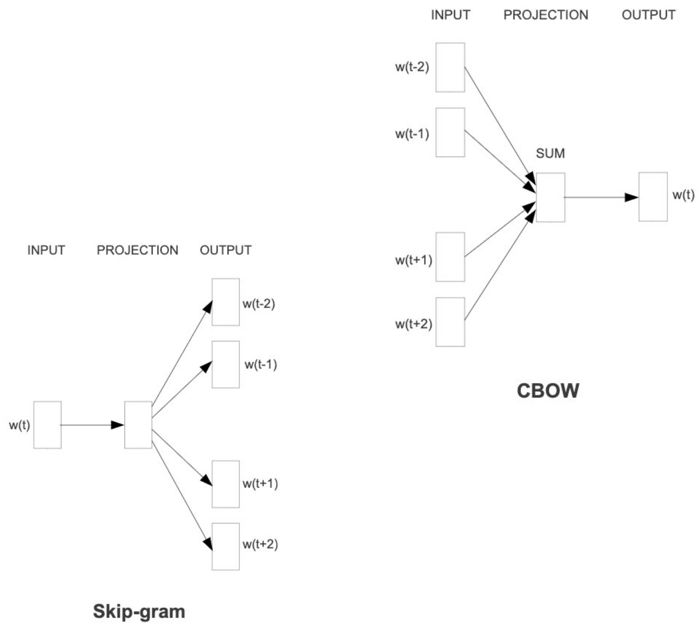
Auto-encoders vs word2vec

Different mechanisms sharing some similarities

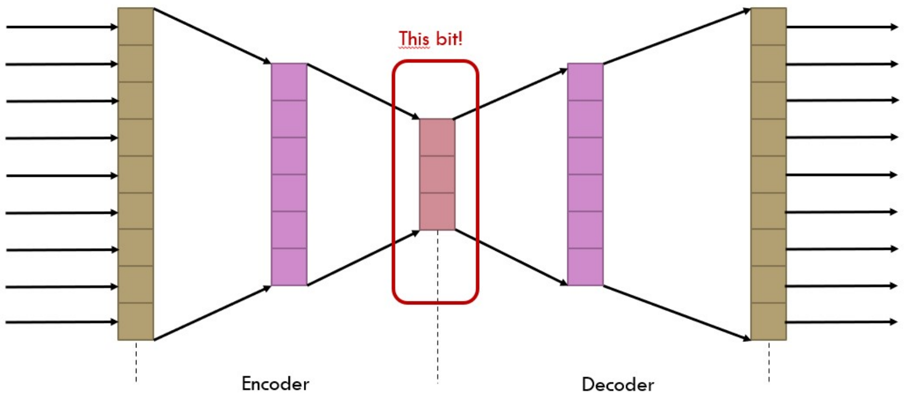
FUTURE CLUB

Auto-encoders

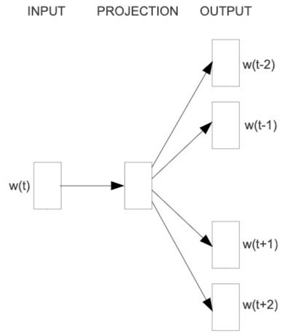




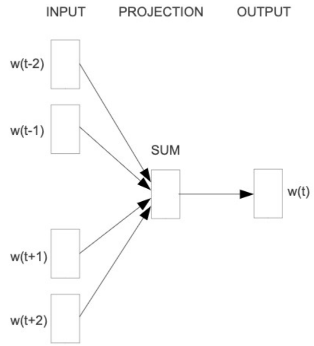
Auto-encoders



This backpropagates the reconstruction error of the inputs
(Potentially) using any activation function and loss

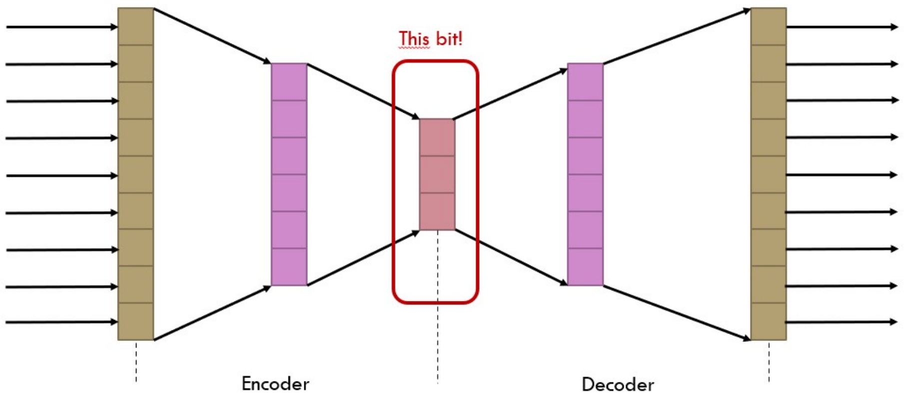


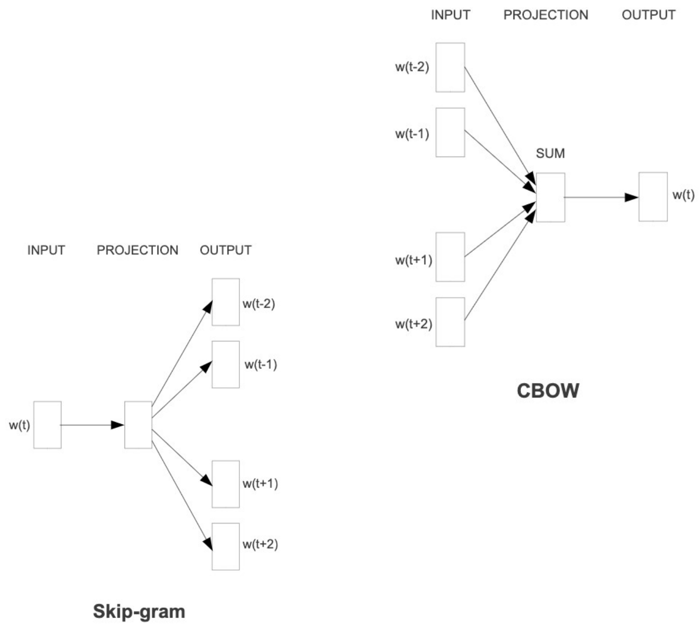
Skip-gram



CBOW

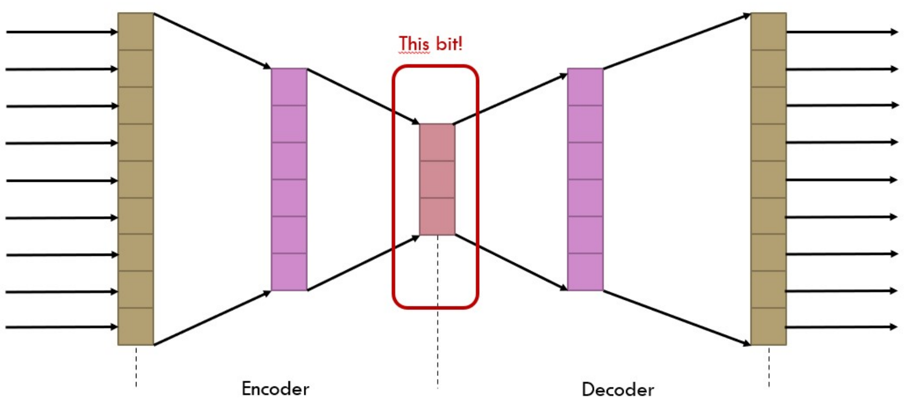
Auto-encoders



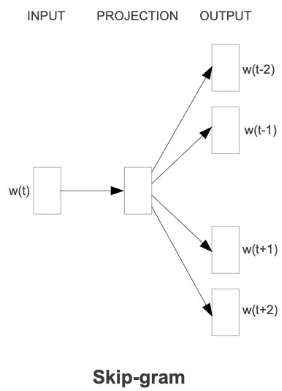


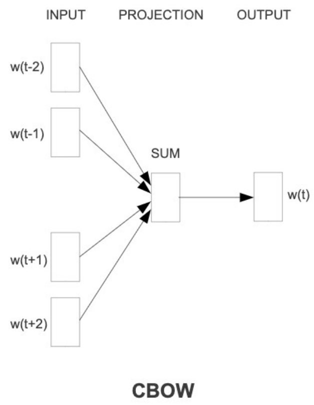
Here we backpropagate the gradient of a softmax classifier to the input word vectors such that we minimize crossentropy loss

Auto-encoders



Dimensionality reduction





Context

FUTURE CLUB

Program

- ~~Basics of Deep Learning (for NLP)~~
- ~~Vectorization models~~
- ~~Auto-encoders~~
- Recurrent neural nets
- Recursive neural nets
- LSTMs
- Attention models
- Deep learning for NLP in practice
- t-SNE
- Google Colab

Recurrent neural nets

What do we do with this?

My forte in early life was playing violin, but towards adolescence I started to drift away from that hobby

Recurrent neural nets

What do we do with this?

My forte in early life was playing violin, but towards adolescence I started to drift away from that hobby

Recurrent neural nets

Observation

Dependencies in text are not always left-to-right (English), nor right-to-left (Arabic) only…

Recurrent neural nets

Even more so if we model multiple languages with one model

Recurrent neural nets

So if we move words through a neural net in a sliding window, we might miss out on dependencies

Recurrent neural nets

Also… consider machine translation

Recurrent neural nets

EN: “The bread is very tasty”

FR:

Recurrent neural nets

EN: “The bread is very tasty”

FR: le/la/les

Recurrent neural nets

EN: “The bread is very tasty”

FR: le/la/les pain

Recurrent neural nets

EN: “The bread is very tasty”

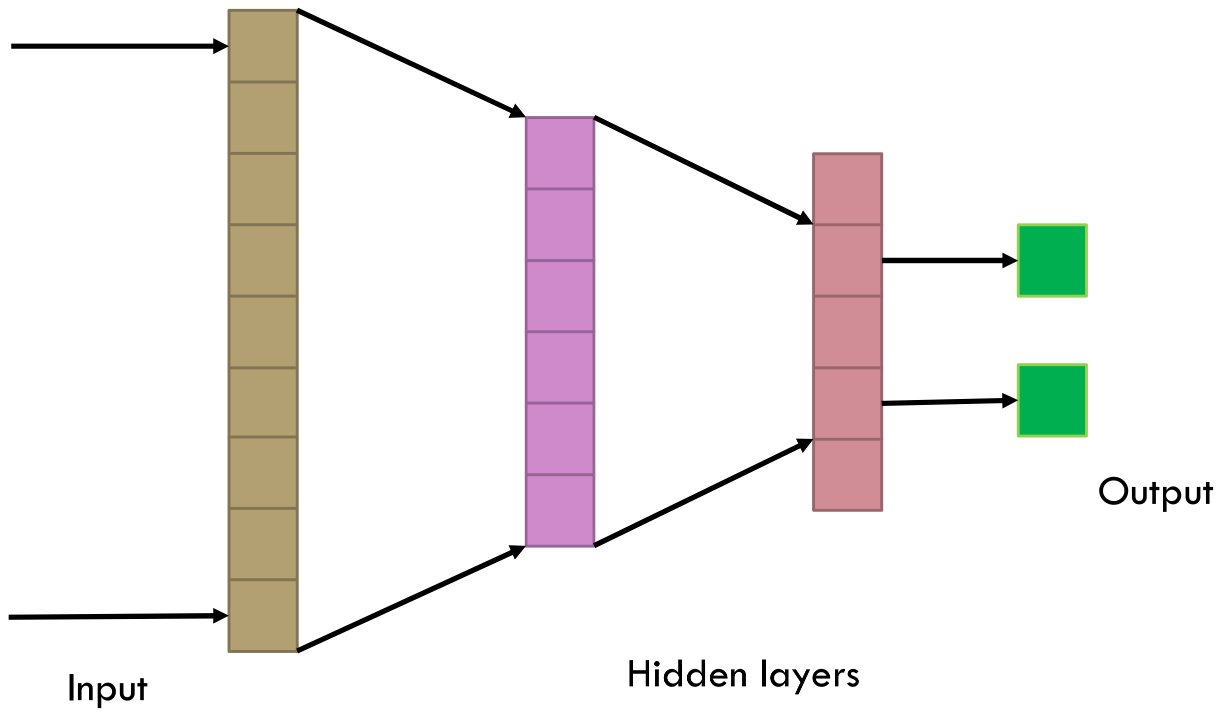
FR: le pain

Recurrent neural nets

We use recurrent neural nets for this

Recurrent neural nets

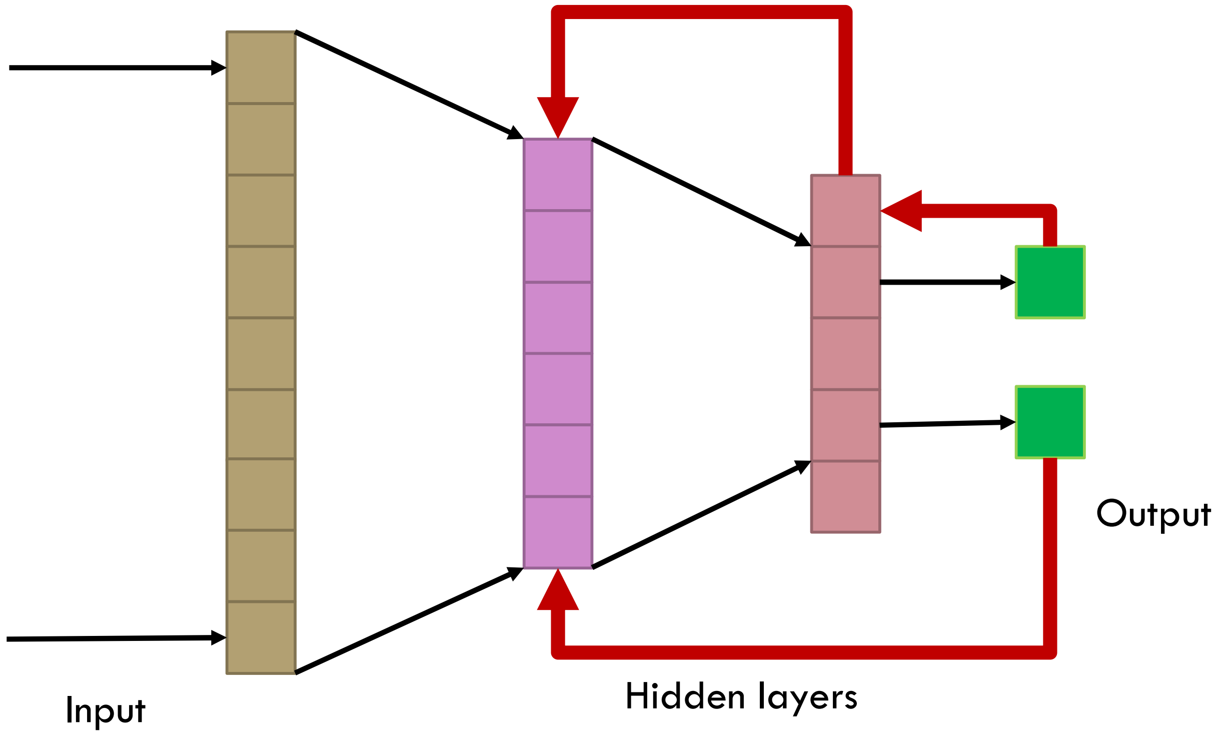
Standard feed-forward neural net



FUTURE CLUB

Recurrent neural nets

Recurrent neural net



FUTURE CLUB

Recurrent neural nets

Backpropagation through time (BPTT)

1. Sequence unrolling
   - BPTT: unrolls the RNN over time: expand the network across time steps, creating a deep feedforward network where each time step corresponds to a layer
   - BP: compute gradients for each layer and update weights using the chain rule

Recurrent neural nets

Backpropagation through time (BPTT)

2. Compute gradients over time
   - BPTT: compute gradients for each time step, starting from the last, working backward in time. Loss at each time step computes gradients for the corresponding layers
   - BP: compute gradients for each layer based on the loss at the final layer and the forward pass

Recurrent neural nets

Backpropagation through time (BPTT)

3. Update weights
   - BPTT: accumulate gradients at each time step for entire sequence. Once you've computed gradients for all time steps, update weights using accumulated gradients
   - BP: update weights using the gradients computed at each layer

Recurrent neural nets

Backpropagation through time (BPTT) – main differences with BP

- Temporal/sequential by nature: information from timestep t (sliding window of words at position X) to t+1 is explicitly captured
- Length of sequence matters. The longer, the harder to train with vanishing/exploding gradients because of accumulation

Recurrent neural nets

Neural networks work well on sequential inputs, like text

Especially when the output/target is also a sequence.

Recurrent neural nets

Such as machine translation

Such as text generation

Recurrent neural nets

RNNs capture information over time, a sort of memory

Recurrent neural nets

RNNs capture information over time, a sort of memory

And they are really powerful!

Recurrent neural nets

But they come with some challenges

Recurrent neural nets

But they come with some challenges

- The RNN "remembers history" but only to its capacity – information decays
  - Both the inputs and the history are "stored" in the hidden layer
- Vanishing gradient is particularly important here because of the feedback
- Exploding gradient is the opposite and might also occur because of the feedback

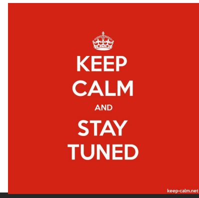
Recurrent neural nets

So what now?

FUTURE CLUB

Recurrent neural nets

LSTMs ☺



Recurrent neural nets

So how do we use this in practice?

Recurrent neural nets

Basically, pick your deep learning framework and use the layers present

My favourite is Keras – this is an RNN:

```
from keras.models import Sequential
from keras.layers import Dense, SimpleRNN
from sklearn.metrics import mean_squared_error

hidden_units = 12
input_shape = (512,)
activation = "linear"

model = Sequential()
model.add(SimpleRNN(hidden_units, input_shape=input_shape,
activation=activation))
model.add(Dense(units=dense_units, activation=activation))
model.compile(loss='mean_squared_error', optimizer='adam')
```