Learning from Data
Lecture 4: Neural Networks I
Perceptrons & Word Embeddings

Hessel Haagsma
hessel.haagsma@rug.nl
room 1311.0411

5th of December 2016

Outline

My Research

Most NLP research focuses on **literal** language.

My Research

Most NLP research focuses on **literal** language.

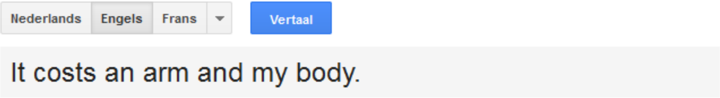
But what about the rest? *metaphor*, *idiom*, *sarcasm*, *humor*, ...

My Research

- Literal translation: *It costs a rib from my body.*
- Correct translation: *It costs me an arm and a leg.*
- Alternative translation: *It is very expensive.*

My Research

- Literal translation: *It costs a rib from my body.*
- Correct translation: *It costs me an arm and a leg.*
- Alternative translation: *It is very expensive.*



My Research

Create an idiom processing pipeline:

1. **Take a sentence:** 'My grandfather kicked the bucket yesterday.'
2. **Detect:** 'My grandfather [kicked the bucket] yesterday.'
3. **Disambiguate:** 'My grandfather [kicked the bucket]$_{idiom}$ yesterday.
4. **Align:** [kick]$_1$ [the]$_2$ [bucket]$_3$ - [to]$_0$ [die]$_1$'
5. **Paraphrase:** 'My grandfather ~~kicked~~ die the bucket yesterday.'
6. **Clean and smooth:** 'My grandfather die*d* ~~the bucket~~ yesterday.'
7. **Result:** 'My grandfather died yesterday.'

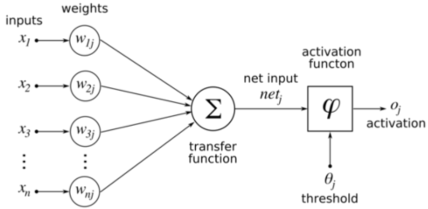
Perceptrons vs. Word Embeddings

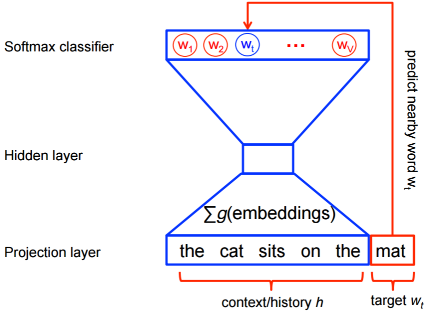
- Ancient (1943-)
- Mostly obsolete
- Relatively simple
- Computationally cheap
- Classifier
- Algorithm

- Recent (1990s/2010s)
- Highly popular in NLP
- Conceptually simple
- Computationally expensive and complex
- Features
- Representations

Perceptrons & Word Embeddings

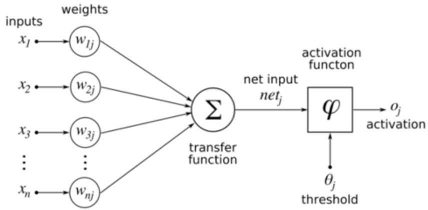
The connection: both are neural network approaches

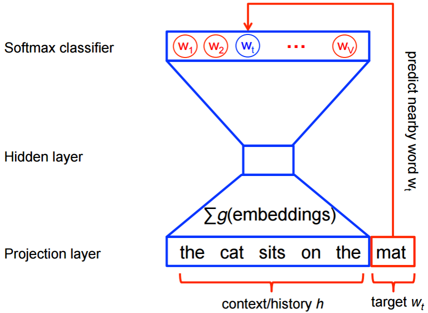




Perceptrons & Word Embeddings

The connection: both are neural network approaches





More about (deep) neural networks next week

What is a perceptron?

- The simplest neural network variant
  - a.k.a. single-layer feed-forward neural network
- Neural networks consist of nodes
  - a.k.a. artificial neurons
  - a.k.a. threshold logic units (TLUs)
- More powerful version: the multi-layer perceptron (next week)

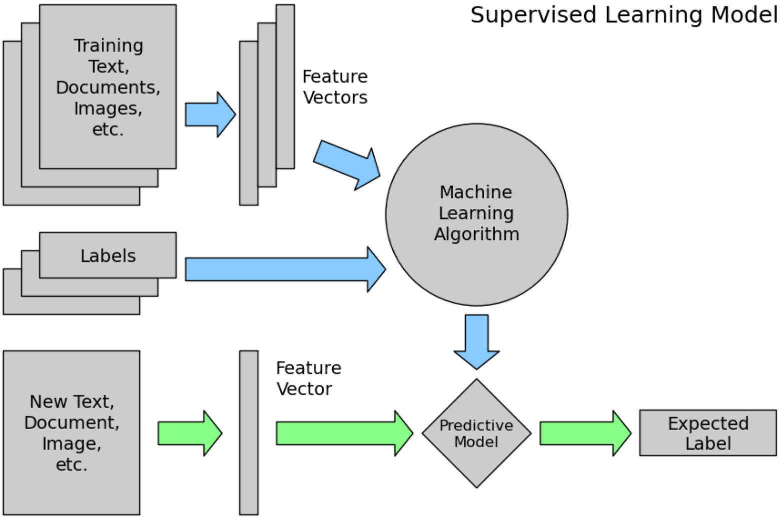
What is a perceptron?

- The simplest neural network variant
  - a.k.a. single-layer feed-forward neural network
- Neural networks consist of nodes
  - a.k.a. artificial neurons
  - a.k.a. threshold logic units (TLUs)
- More powerful version: the multi-layer perceptron (next week)

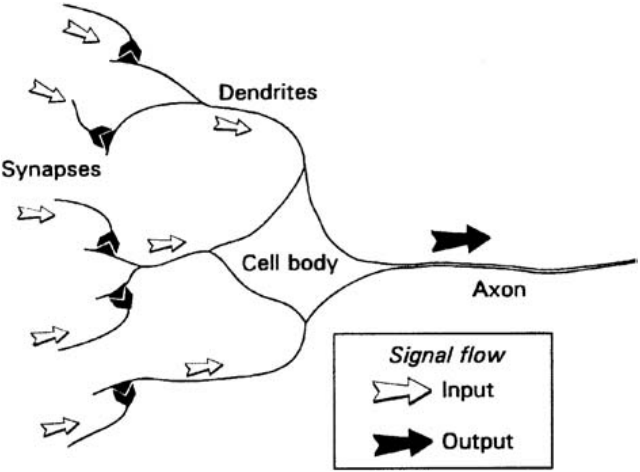
What is a perceptron?

- The simplest neural network variant
    - a.k.a. single-layer feed-forward neural network
- Neural networks consist of nodes
    - a.k.a. artificial neurons
    - a.k.a. threshold logic units (TLUs)
- More powerful version: the multi-layer perceptron (next week)

Supervised Learning

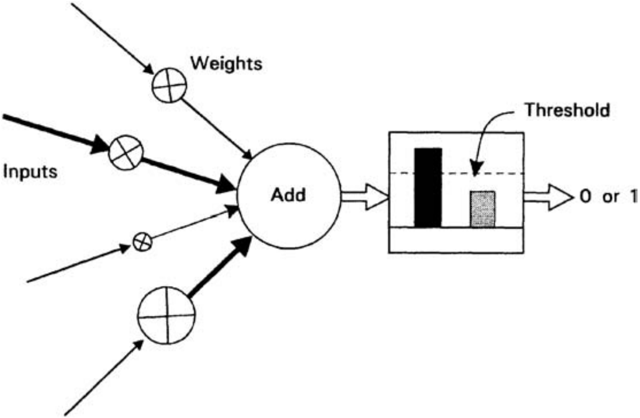


The Human Neuron



Source: Kevin Gurney - An Introduction to Neural Networks (1997)

The Artificial Neuron



Source: Kevin Gurney - An Introduction to Neural Networks (1997)

Definition

A *neural network* is an interconnected assembly of simple processing elements, *units* or *nodes*, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or *weights*, obtained by a process of adaptation to, or *learning from, a set of training patterns*.

- neural network
- units/nodes
- weights
- (supervised) learning

Definition

A **neural network** is an interconnected assembly of simple processing elements, *units* or *nodes*, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or *weights*, obtained by a process of adaptation to, or *learning from, a set of training patterns*.

- neural network
- units/nodes
- weights
- (supervised) learning

Definition

A *neural network* is an interconnected assembly of simple processing elements,**units** or **nodes**, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or *weights*, obtained by a process of adaptation to, or *learning from, a set of training patterns*.

- neural network
- units/nodes
- weights
- (supervised) learning

Definition

A *neural network* is an interconnected assembly of simple processing elements, *units* or *nodes*, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or **weights**, obtained by a process of adaptation to, or *learning from, a set of training patterns*.

- neural network
- units/nodes
- weights
- (supervised) learning

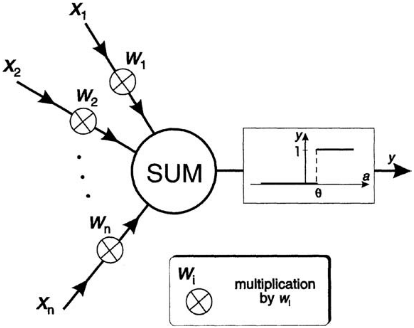
Definition

A *neural network* is an interconnected assembly of simple processing elements, *units* or *nodes*, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the interunit connection strengths, or *weights*, obtained by a process of adaptation to, or **learning from, a set of training patterns**.

- neural network
- units/nodes
- weights
- (supervised) learning

Components

- neural network
- inputs
- weights
- node(s)
- activation
- output

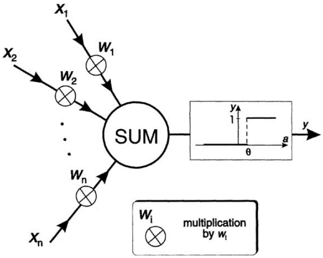


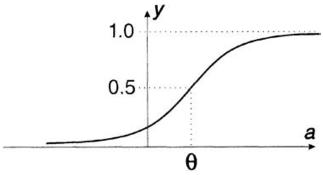
From Activation to Output

- Activation $= a = \sum_{i=1}^{n} *w_i * x_i$
- Output $= y = h(a)$
- Activation function $h(a)$:
  - Threshold θ:

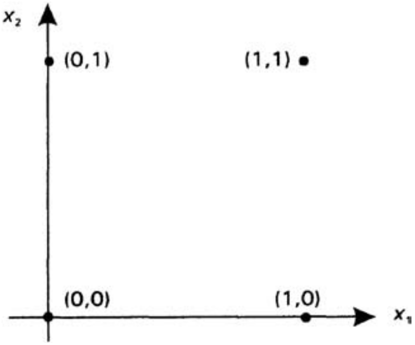
$$y = \begin{cases} 1 \ \ if \ \ a \geq \theta \\ 0 \ \ if \ \ a < \theta \end{cases}$$

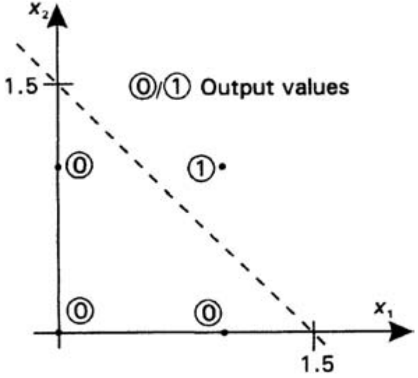
  - Continuous squashing (e.g. sigmoid): $y = \sigma(a) = \dfrac{1}{1 + e^{-(a-\theta)/\rho}}$



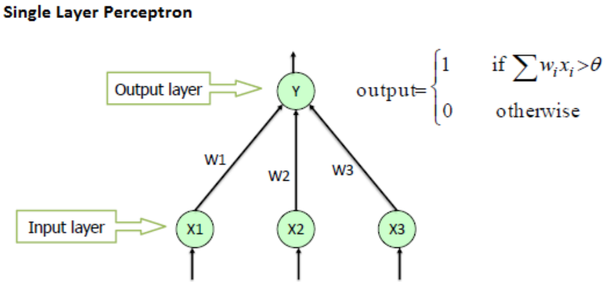


Perceptron Classification Example



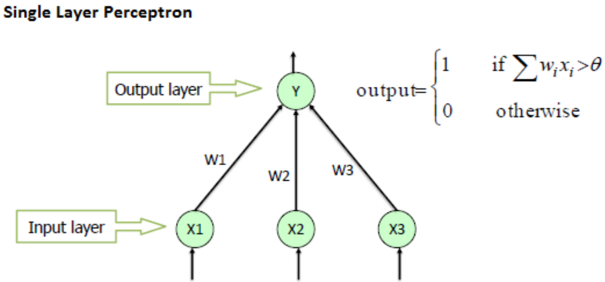


Perceptron Classification Example



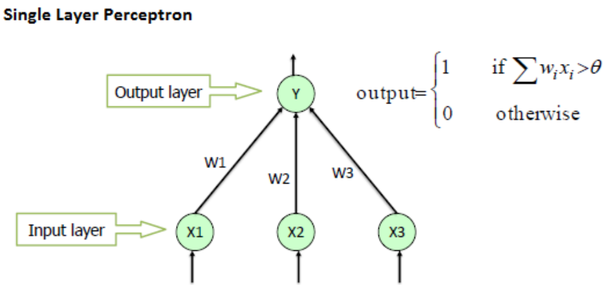
x_1	w_1	x_2	w_2	a	θ	y

Perceptron Classification Example



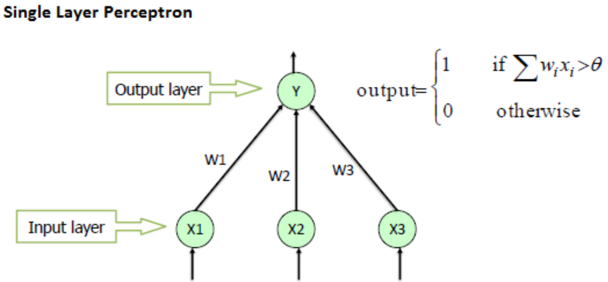
x_1	w_1	x_2	w_2	a	θ	y
0	1	0	1	0	1.5	0

Perceptron Classification Example



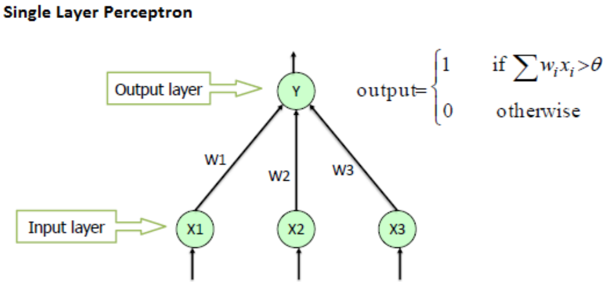
x_1	w_1	x_2	w_2	a	θ	y
0	1	0	1	0	1.5	0
0	1	1	1	1	1.5	0

Perceptron Classification Example



x_1	w_1	x_2	w_2	a	θ	y
0	1	0	1	0	1.5	0
0	1	1	1	1	1.5	0
1	1	0	1	1	1.5	0

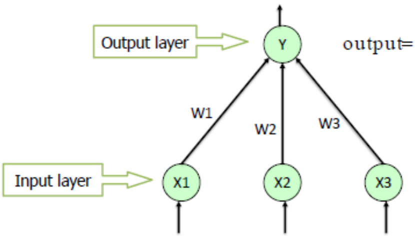
Perceptron Classification Example



x_1	w_1	x_2	w_2	a	θ	y
0	1	0	1	0	1.5	0
0	1	1	1	1	1.5	0
1	1	0	1	1	1.5	0
1	1	1	1	2	1.5	1

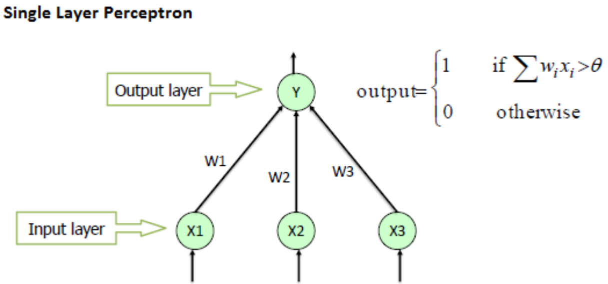
Training the Neural Network Model

Single Layer Perceptron



$$\text{output} = \begin{cases} 1 & \text{if } \sum w_i x_i > \theta \\ 0 & \text{otherwise} \end{cases}$$

Training the Neural Network Model



Where is the model?

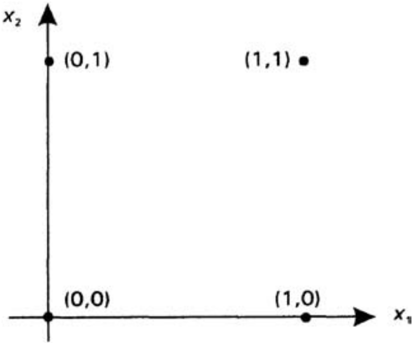
The Perceptron Rule

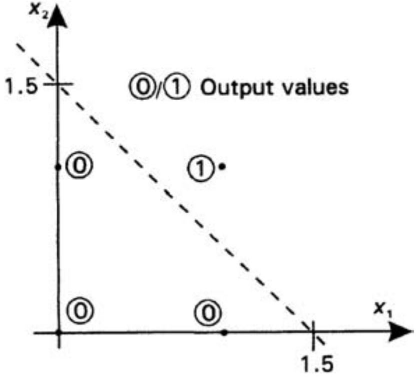
- **The intuition:** figure out who's to blame
  - Learning a task == Learning the weights
  - How? Initialize randomly, then adapt
  - How? Compare predicted and gold values, play the blame game
  - Bigger contribution to error $\rightarrow$ Bigger change

The Perceptron Rule

- **The intuition:** figure out who's to blame
  - Learning a task == Learning the weights
  - How? Initialize randomly, then adapt
  - How? Compare predicted and gold values, play the blame game
  - Bigger contribution to error $\rightarrow$ Bigger change
- **The rule:** $\Delta w_i = \alpha * (t - y) * x_i$
  - Δw: change in weight
  - α: learning rate
  - t: target (gold label)
  - y: output (predicted label)
  - x: input

Perceptron Classification Example





Perceptron Training Example

x_1	w_1	x_2	w_2	a	θ	y	t	$\alpha * (t - y)$	Δw_1	Δw_2	$\Delta \theta$
0	0.0	0	0.4	0	0.3	0	0	0	0	0	0

- **The Rule:** $\Delta w_i = \alpha * (t - y) * x_i$
- **Learning Rate:** 0.25
- **Threshold Input:** -1

Perceptron Training Example

x_1	w_1	x_2	w_2	a	θ	y	t	$\alpha * (t - y)$	Δw_1	Δw_2	$\Delta\theta$
0	0.0	0	0.4	0	0.3	0	0	0	0	0	0
0	0.0	1	0.4	0.4	0.3	1	0	-0.25	0	-0.25	+0.25

- **The Rule:** $\Delta w_i = \alpha * (t - y) * x_i$
- **Learning Rate:** 0.25
- **Threshold Input:** -1

Perceptron Training Example

x_1	w_1	x_2	w_2	a	θ	y	t	$\alpha * (t - y)$	Δw_1	Δw_2	$\Delta\theta$
0	0.0	0	0.4	0	0.3	0	0	0	0	0	0
0	0.0	1	0.4	0.4	0.3	1	0	-0.25	0	-0.25	+0.25
1	0.0	0	0.15	0	0.55	0	0	0	0	0	0

- **The Rule:** $\Delta w_i = \alpha * (t - y) * x_i$
- **Learning Rate:** 0.25
- **Threshold Input:** -1

Perceptron Training Example

x_1	w_1	x_2	w_2	a	θ	y	t	$\alpha * (t - y)$	Δw_1	Δw_2	$\Delta\theta$
0	0.0	0	0.4	0	0.3	0	0	0	0	0	0
0	0.0	1	0.4	0.4	0.3	1	0	-0.25	0	-0.25	+0.25
1	0.0	0	0.15	0	0.55	0	0	0	0	0	0
1	0.0	1	0.15	0.15	0.55	0	1	0.25	+0.25	+0.25	-0.25

- **The Rule:** $\Delta w_i = \alpha * (t - y) * x_i$
- **Learning Rate:** 0.25
- **Threshold Input:** -1

Perceptron Training Example

x_1	w_1	x_2	w_2	a	θ	y	t	$\alpha * (t - y)$	Δw_1	Δw_2	$\Delta\theta$
0	0.0	0	0.4	0	0.3	0	0	0	0	0	0
0	0.0	1	0.4	0.4	0.3	1	0	-0.25	0	-0.25	+0.25
1	0.0	0	0.15	0	0.55	0	0	0	0	0	0
1	0.0	1	0.15	0.15	0.55	0	1	0.25	+0.25	+0.25	-0.25
-	0.25	-	0.40	-	0.3	-	-	-	-	-	-

- **The Rule:** $\Delta w_i = \alpha * (t - y) * x_i$
- **Learning Rate:** 0.25
- **Threshold Input:** -1

Perceptron Training Properties & Parameters

Weight Initialization: how to set starting weights

Stochastic Learning: update weights after each training example

Batch Learning: update weights after a batch of training examples

Learning Rate: update the weights a lot, or just a little?

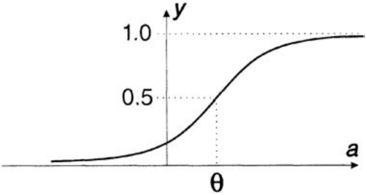
Stopping Criterion: when have we trained enough?

Threshold/Bias: in practice, a weight with fixed input of -1

Error Function: how to define the size of the mistake

Perceptron Capabilities

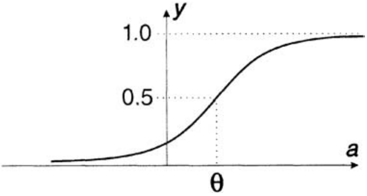
- **So far:** binary classification, with categorical output
- Threshold function $\rightarrow$ categorical output
- **Also possible:** regression, with continous output



- Sigmoid function $\rightarrow$ continuous output
- **Also possible:** multiple outputs, for multi-class classification
- Get probability for each class $\rightarrow$ select highest

Perceptron Capabilities

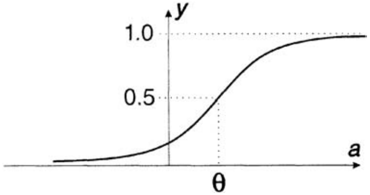
- **So far:** binary classification, with categorical output
- Threshold function → categorical output
- **Also possible:** regression, with continous output



- Sigmoid function → continuous output
- **Also possible:** multiple outputs, for multi-class classification
- Get probability for each class → select highest

Perceptron Capabilities

- **So far:** binary classification, with categorical output
- Threshold function $\rightarrow$ categorical output
- **Also possible:** regression, with continous output

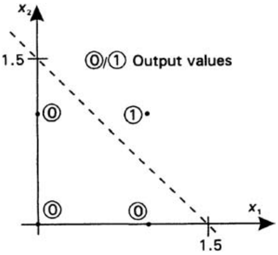


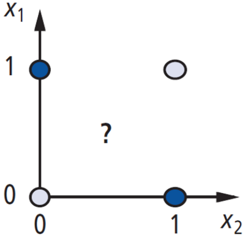
- Sigmoid function $\rightarrow$ continuous output
- **Also possible:** multiple outputs, for multi-class classification
- Get probability for each class $\rightarrow$ select highest

Perceptron Limitations

Limitations:

- Only handles linearly separable problems
- AND, OR, but not XOR

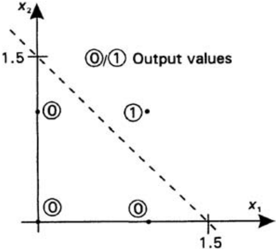


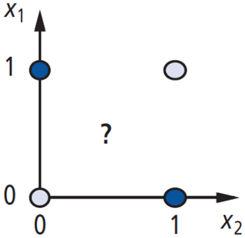


Perceptron Limitations

Limitations:

- Only handles linearly separable problems
- AND, OR, but not XOR





Solution:

- Use multi-layer networks (next week)

Word Embeddings

Intuition

"You shall know a word by the company it keeps"

J.R. Firth, 1957

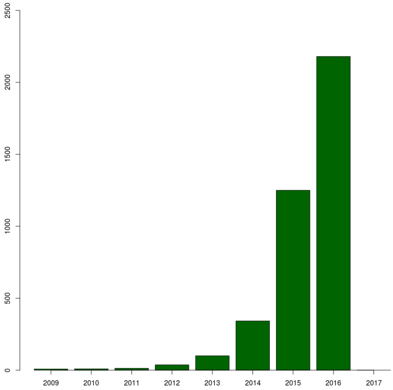
Terminology

- **Word Embeddings:** words, embedded in a vector space
- A distributional semantic representation
- Dense, real-valued, high-dimensional vectors
- Also known as:
  - (semantic) word vectors
  - neural (word) embeddings
  - semantic vector space models

The Rise and Rise of Embeddings

Google Scholar hits for "word embeddings":

- 2009: 8
- 2010: 9
- 2011: 13
- 2012: 37
- 2013: 100
- 2014: 342
- 2015: >1250
- 2016: >2180
- 2017: ?



http://aclanthology.info/events/acl-2016

Early Days

The idea of embedding words in a vector space is **not** new
Earliest ideas from the **50s** and **60s**
From 1990 onwards, **precursors** arise:

- Take count-based statistics, apply dimensionality reduction
- *LSA, topic models, LDA, PCA*
- Early neural language model approaches
- *SOM, SRN*

All **distributional semantic models**

Early Days

The idea of embedding words in a vector space is **not** new

Earliest ideas from the **50s** and **60s**

From 1990 onwards, **precursors** arise:

- Take count-based statistics, apply dimensionality reduction
- *LSA, topic models, LDA, PCA*
- Early neural language model approaches
- *SOM, SRN*

All **distributional semantic models**

Recent Development

Why now?

- Idea of *pre-training* (Collobert and Weston, 2008)
- Effective toolkit for pre-training (Mikolov, 2013)
- Impressive performance on different tasks sparks hype
- Also: availability of big data
- Also: increase computational power (GPUs)

Recent Development

Why now?

- Idea of *pre-training* (Collobert and Weston, 2008)
- Effective toolkit for pre-training (Mikolov, 2013)
- Impressive performance on different tasks sparks hype
- Also: availability of big data
- Also: increase computational power (GPUs)

Recent Development

Why now?

- Idea of *pre-training* (Collobert and Weston, 2008)
- Effective toolkit for pre-training (Mikolov, 2013)
- Impressive performance on different tasks sparks hype
- Also: availability of big data
- Also: increase computational power (GPUs)

Recent Development

Why now?

- Idea of *pre-training* (Collobert and Weston, 2008)
- Effective toolkit for pre-training (Mikolov, 2013)
- Impressive performance on different tasks sparks hype
- Also: availability of big data
- Also: increase computational power (GPUs)

Recent Development

Why now?

- Idea of *pre-training* (Collobert and Weston, 2008)
- Effective toolkit for pre-training (Mikolov, 2013)
- Impressive performance on different tasks sparks hype
- Also: availability of big data
- Also: increase computational power (GPUs)

Recent Development

Why now?

- Idea of *pre-training* (Collobert and Weston, 2008)
- Effective toolkit for pre-training (Mikolov, 2013)
- Impressive performance on different tasks sparks hype
- Also: availability of big data
- Also: increase computational power (GPUs)

Words as Vectors - Principle

Previously: count vectors, tf-idf vectors
Why represent documents as vectors?

Words as Vectors - Principle

Previously: count vectors, tf-idf vectors
Why represent documents as vectors?

Simple: math doesn't work with words!
So, represent words as vectors as well

Words as Vectors - Example

- Represent four adjectives: *great*, *terrible*, *good*, *bad*
- Two main dimensions: *intensity*, *positivity*

Word	Intensity	Positivity
great	0.8	1
terrible	0.9	0
good	0.5	1
bad	0.5	0

Words as Vectors - Scaling Up

- More dimensions $\rightarrow$ more information
- Not just *intensity, positivity*, but also *simplicity, comparative, colour-related, rudeness, register, etc.*
- Adjectives $\rightarrow$ all words
- Represent all words in all dimensions
- Manual $\rightarrow$ automatic
- Labelled $\rightarrow$ unlabelled
- Optimal semantic representation, meaningless dimensions
- Alternatively: interpretable, non-distributional, 'linguistic' representations (Faruqui & Dyer, 2015)

Words as Vectors - Scaling Up

- More dimensions → more information
- Not just *intensity, positivity*, but also *simplicity, comparative, colour-related, rudeness, register, etc.*
- Adjectives → all words
- Represent all words in all dimensions
- Manual → automatic
- Labelled → unlabelled
- Optimal semantic representation, meaningless dimensions
- Alternatively: interpretable, non-distributional, 'linguistic' representations (Faruqui & Dyer, 2015)

Words as Vectors - Scaling Up

- More dimensions $\rightarrow$ more information
- Not just *intensity, positivity*, but also *simplicity, comparative, colour-related, rudeness, register, etc.*
- Adjectives $\rightarrow$ all words
- Represent all words in all dimensions
- Manual $\rightarrow$ automatic
- Labelled $\rightarrow$ unlabelled
- Optimal semantic representation, meaningless dimensions
- Alternatively: interpretable, non-distributional, 'linguistic' representations (Faruqui & Dyer, 2015)

Words as Vectors - Scaling Up

- More dimensions $\rightarrow$ more information
- Not just *intensity, positivity*, but also *simplicity, comparative, colour-related, rudeness, register, etc.*
- Adjectives $\rightarrow$ all words
- Represent all words in all dimensions
- Manual $\rightarrow$ automatic
- Labelled $\rightarrow$ unlabelled
- Optimal semantic representation, meaningless dimensions
- Alternatively: interpretable, non-distributional, 'linguistic' representations (Faruqui & Dyer, 2015)

Applications

Word Embeddings can (and have) been used as features for almost every NLP task:

- Tokenization
- Part-of-speech tagging
- Sentiment analysis
- Syntactic parsing
- Paraphrase detection
- Machine translation
- Named entity recognition

Can be used with various models, from 'traditional' SVMs to very deep neural networks

Why does it work?

- Prior Knowledge
- Generalization
- Unlabelled Big Data
- Beyond Training Data

Training

Don't count, predict! (Baroni et al., 2014)

Training

Don't count, predict! (Baroni et al., 2014)

Nico Rosberg quits Formula One after **xxxxx** world title

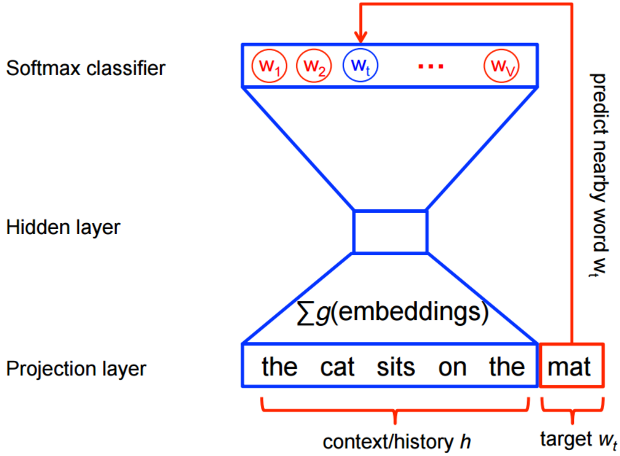
Training

Don't count, predict! (Baroni et al., 2014)

Arsenal dealt massive blow as midfield maestro out for three **xxxxxx**

One method: Continuous Bag-of-Words (CBoW)

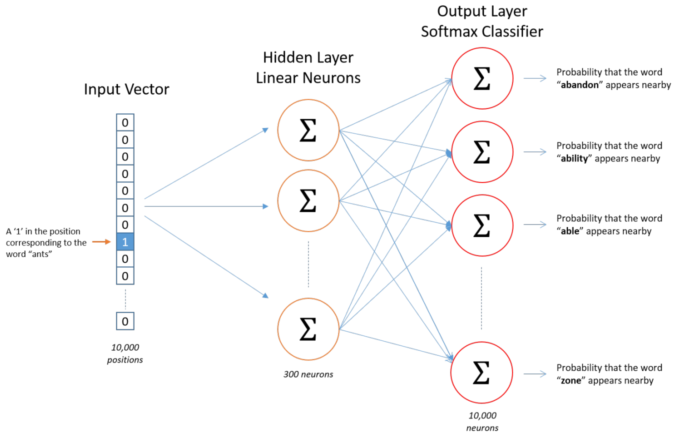
Given a context, predict a word



Source: `https://www.tensorflow.org/tutorials/word2vec/index.html`

Other method: Skip-Gram

Given a word, predict its context



Source: `http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/`

But: many different variants exist!

Extrinsic Evaluation vs. Intrinsic Evaluation

How to evaluate word embeddings?

Extrinsic Evaluation vs. Intrinsic Evaluation

Extrinsic: evaluate model as part of a bigger system

Intrinsic: evaluate a model by itself, stand-alone

Extrinsic: assess effect of new tokenizer on parsing accuracy

Intrinsic: compare new tokenizer output to gold tokenization

Evaluating Embeddings

Many different ways to train embeddings, which is better?

Extrinsic Evaluation

- Use embeddings in parser, PoS-tagger, etc.
- **Pro:** useful, realistic, meaningful
- **Con:** selection of tasks, datasets, systems, expensive

Intrinsic Evaluation

- Use embeddings to measure similarity, analogies, etc.
- **Pro:** cheap, fewer confounding factors
- **Con:** too specific, imperfect human judgements, different datasets

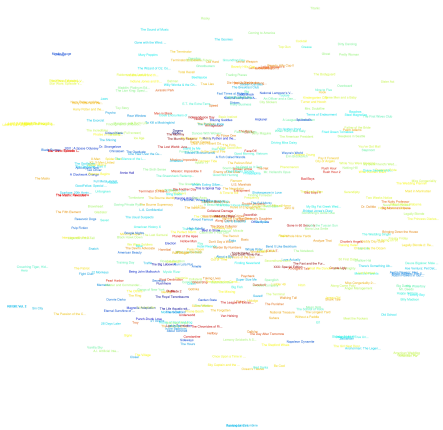
Similarities

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

Source: Collobert et al., *Natural Language Processing (Almost) from Scratch* (2011)

Similarities

- Visualizing embeddings: 2D/3D plot using t-SNE (`https://lvdmaaten.github.io/tsne/`)
- Plot: Common English Words
- Plot: Netflix (3D as 2D + color)
- Demo: similarity querying (`/net/shared/word2vec/distance`)



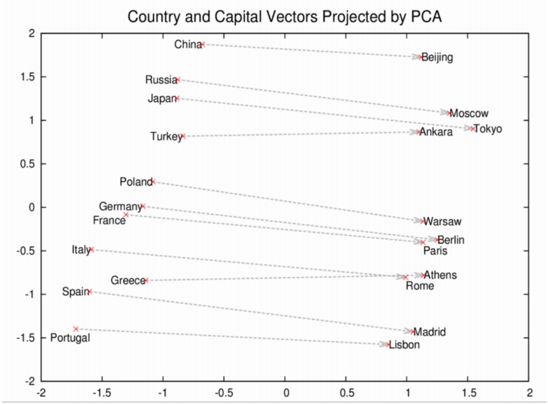
Relations

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Source: Collobert et al., *Natural Language Processing (Almost) from Scratch* (2011)

Relations

Demo: analogy querying (`/net/shared/word2vec/word-analogy`)



Source: `https://deeplearning4j.org/word2vec`

Assignment Overview

Assignment Week 4

Part 1: explore pre-trained word embeddings

Part 2: improve and analyse perceptron performance

Deadline: December 14, 23.00
Note: use LWP **or** use SSH **or** install word2vec