NEURAL NETWORKS APPLIED

David Kauchak
CS159 – Fall 2020

1

Admin

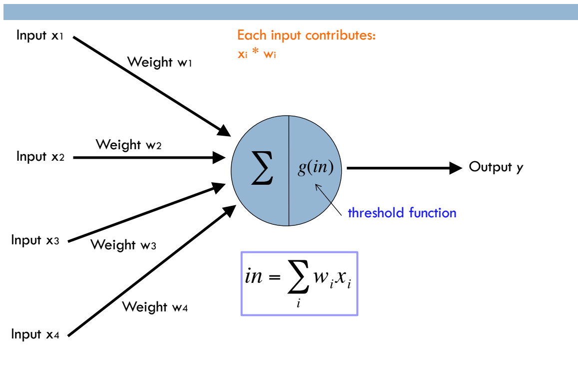
Grading
- Assignment 6a back
- Project proposal back
- Assignment 6b & 7 outstanding

Project status report due Wed

Thursday
- start class with course feedback
- ethics discussion: spend 15 minutes glancing over papers

2

A Single Neuron/Perceptron

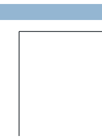


3

Activation functions

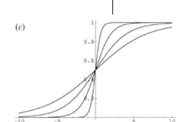
hard threshold:

$$g(in) = \begin{cases} 1 & if\ in \geq T \\ 0 & otherwise \end{cases}$$

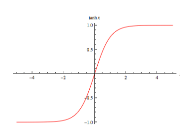


sigmoid

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



tanh x

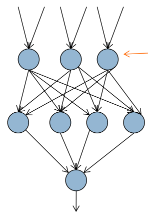


why other threshold functions?

4

Neural network

inputs



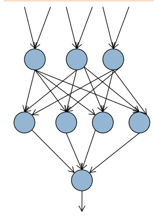
Individual perceptrons/neurons

5

Neural network

inputs

some inputs are provided/entered

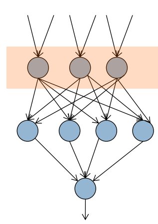


6

Neural network

inputs

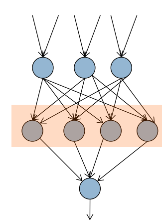
each perceptron computes and calculates an answer



7

Neural network

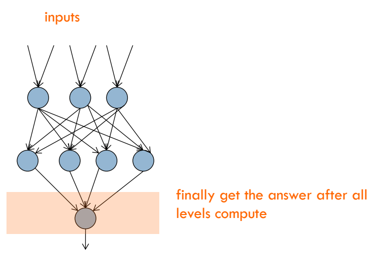
inputs



those answers become inputs for the next level

8

Neural network



9

Training the perceptron

First wave in neural networks in the 1960's

Single neuron

Trainable: its threshold and input weights can be modified

If the neuron doesn't give the desired output, then it has made a mistake

Input weights and threshold can be changed according to a learning algorithm

10

Examples - Logical operators

AND – if all inputs are 1, return 1, otherwise return 0

OR – if at least one input is 1, return 1, otherwise return 0

NOT – return the opposite of the input

XOR – if exactly one input is 1, then return 1, otherwise return 0

11

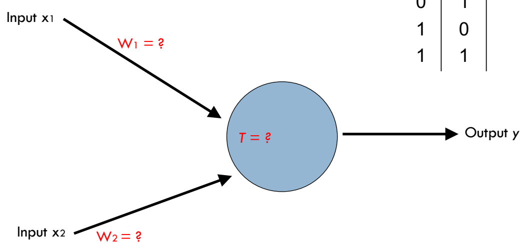
AND

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

12

AND

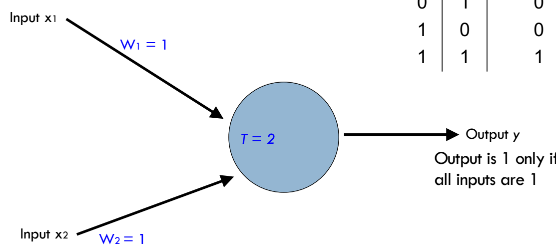
x_1	x_2	x_1 and x_2
0	0	0
0	1	0
1	0	0
1	1	1



13

AND

x_1	x_2	x_1 and x_2
0	0	0
0	1	0
1	0	0
1	1	1

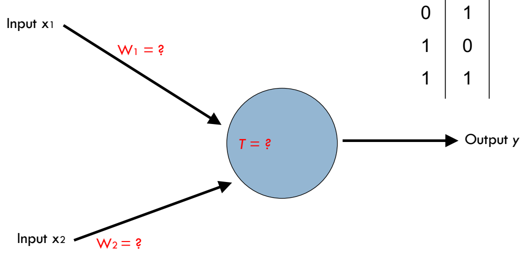


Inputs are either 0 or 1

14

OR

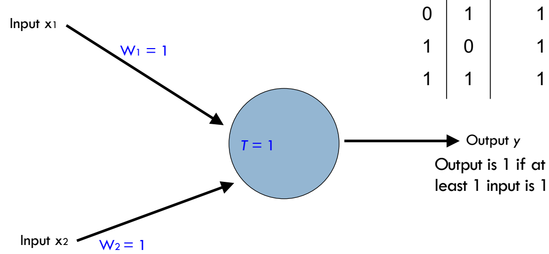
x_1	x_2	x_1 or x_2
0	0	0
0	1	1
1	0	1
1	1	1



15

OR

x_1	x_2	x_1 or x_2
0	0	0
0	1	1
1	0	1
1	1	1

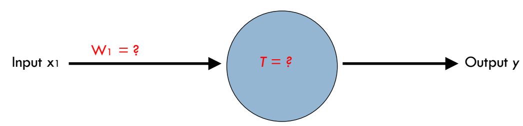


Inputs are either 0 or 1

16

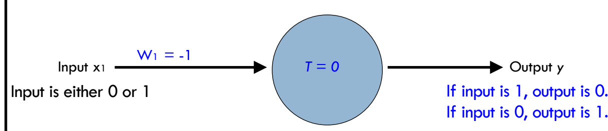
NOT

x_1	not x_1
0	1
1	0



17

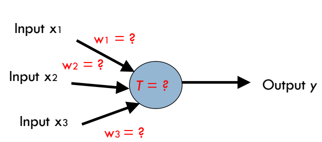
NOT



18

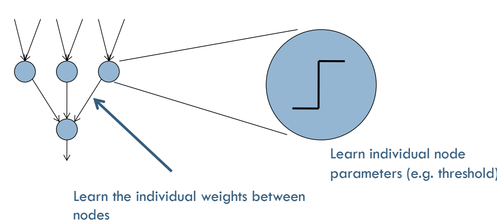
How about…

X_1	X_2	X_3	Output
0	0	0	1
0	1	0	0
1	0	0	1
1	1	0	0
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	0



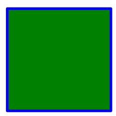
19

Training neural networks



20

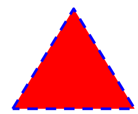
Positive or negative?



NEGATIVE

21

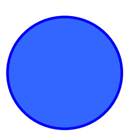
Positive or negative?



NEGATIVE

22

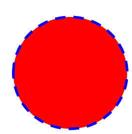
Positive or negative?



POSITIVE

23

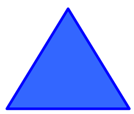
Positive or negative?



NEGATIVE

24

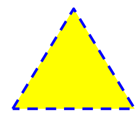
Positive or negative?



POSITIVE

25

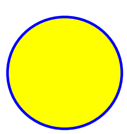
Positive or negative?



POSITIVE

26

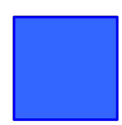
Positive or negative?



NEGATIVE

27

Positive or negative?



POSITIVE

28

A method to the madness

blue = positive

yellow triangles = positive

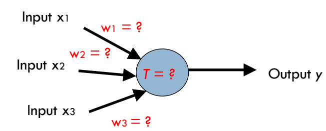
all others negative

How did you figure this out (or some of it)?

29

Training a neuron (perceptron)

x_1	x_2	x_3	**Output**
0	0	0	1
0	1	0	0
1	0	0	1
1	1	0	0
0	0	1	1
0	1	1	1
1	0	1	1
1	1	1	0



1. start with some initial weights and thresholds
2. show examples repeatedly to NN
3. update weights/thresholds by comparing NN output to actual output

30

Perceptron learning algorithm

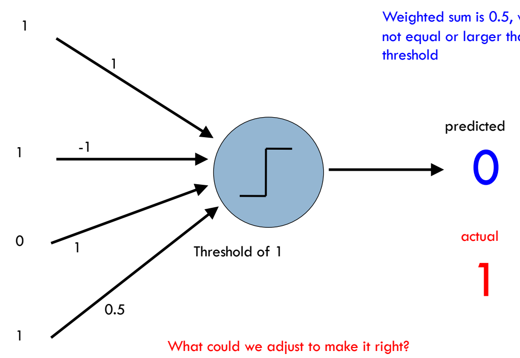
repeat until you get all examples right:

- for each "training" example:
  - calculate current prediction on example
  - if *wrong*:
    - update weights and threshold towards getting this example correct

31

Perceptron learning

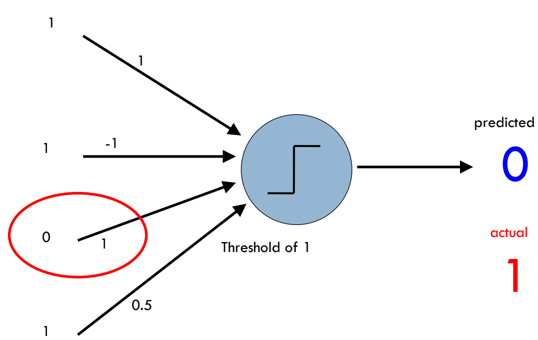
Weighted sum is 0.5, which is not equal or larger than the threshold



What could we adjust to make it right?

32

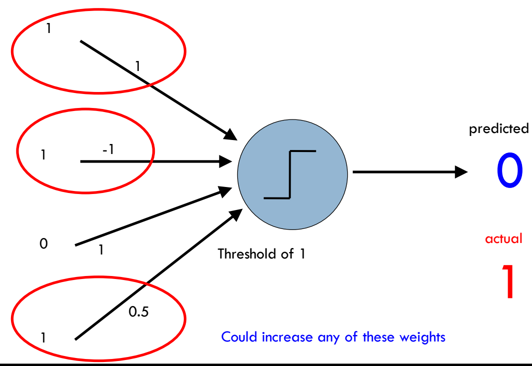
Perceptron learning



This weight doesn't matter, so don't change

33

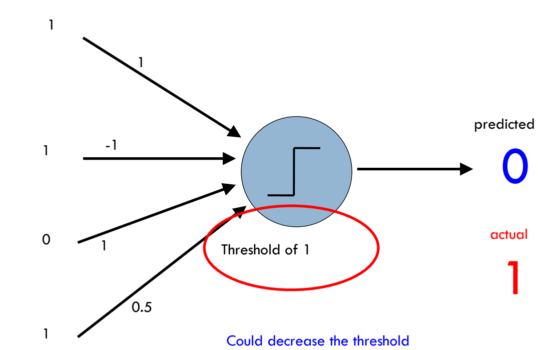
Perceptron learning



Could increase any of these weights

34

Perceptron learning



Could decrease the threshold

35

Perceptron learning

A few missing details, but not much more than this

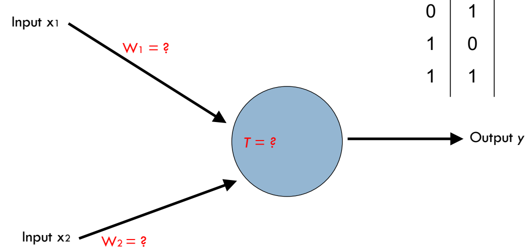
Keeps adjusting weights as long as it makes mistakes

Run through the training data multiple times until convergence, some number of iterations, or until weights don't change (much)

36

XOR

x_1	x_2	x_1 **xor** x_2
0	0	0
0	1	1
1	0	1
1	1	0



37

Perceptron learning

A few missing details, but not much more than this

Keeps adjusting weights as long as it makes mistakes

Run through the training data multiple times until convergence, some number of iterations, or until weights don't change (much)

If the training data is linearly separable the perceptron learning algorithm is guaranteed to converge to the "correct" solution (where it gets all examples right)

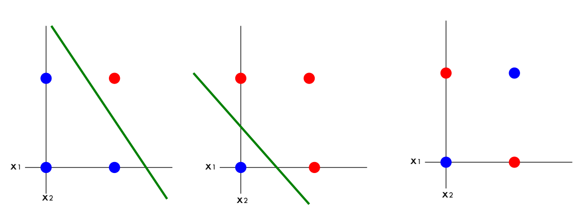
38

Linearly Separable

x_1	x_2	x_1 **and** x_2
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	x_1 **or** x_2
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	x_1 **xor** x_2
0	0	0
0	1	1
1	0	1
1	1	0



39

Perceptrons

1969 book by Marvin Minsky and Seymour Papert

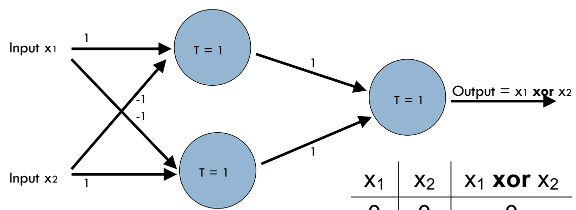
The problem is that they can only work for classification problems that are linearly separable

Insufficiently expressive

"Important research problem" to investigate multilayer networks although they were pessimistic about their value

40

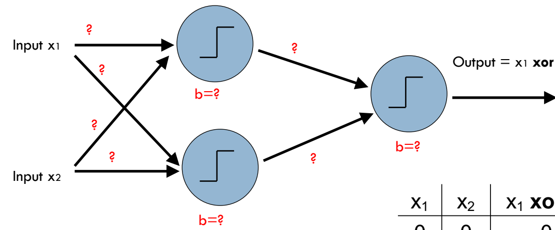
XOR



x_1	x_2	x_1 **xor** x_2
0	0	0
0	1	1
1	0	1
1	1	0

41

Training



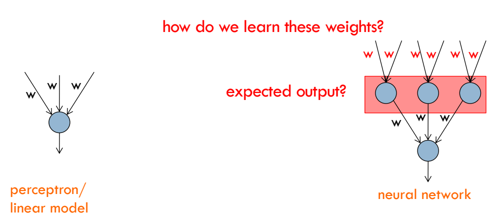
How do we learn the weights?

x_1	x_2	x_1 **xor** x_2
0	0	0
0	1	1
1	0	1
1	1	0

42

Learning in multilayer networks

Challenge: for multilayer networks, we don't know what the expected output/error is for the internal nodes!



43

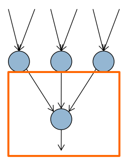
Backpropagation: intuition

Gradient descent method for learning weights by optimizing a loss function

1. calculate output of all nodes
2. calculate the weights for the output layer based on the error
3. "backpropagate" errors through hidden layers

44

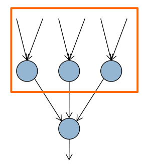
Backpropagation: intuition



We can calculate the actual error here

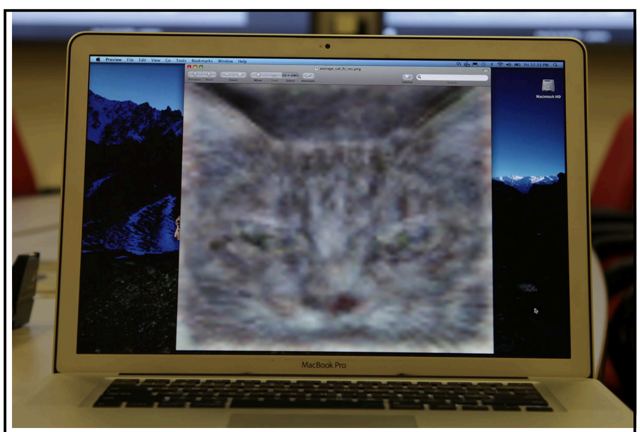
45

Backpropagation: intuition



Key idea: propagate the error back to this layer

46



Trained a NN with 1B connections on 10M snapshots from youtube on 16,000 processors

47

http://www.nytimes.com/2012/06/26/technology/in-a-big-network-of-computers-evidence-of-machine-learning.html

48

Deep learning

WIKIPEDIA

Deep learning is a branch of machine learning based on a set of algorithms that attempt to model high level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and non-linear transformations.

Deep learning is part of a broader family of machine learning methods based on learning representations of data.

49

Deep learning

Key: learning better features that abstract from the "raw" data

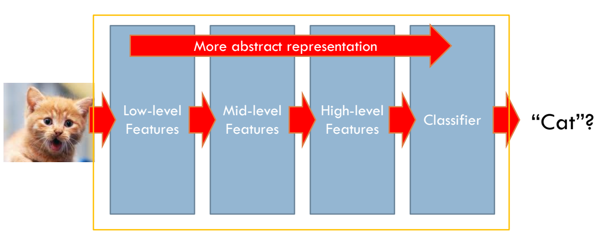
Using learned feature representations based on large amounts of data, generally unsupervised

Using classifiers with multiple layers of learning

50

Deep learning

- Train *multiple layers* of features/abstractions from data.
- Try to discover *representation* that makes decisions easy.

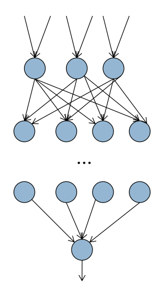


Deep Learning: train layers of features so that classifier works well.

Slide adapted from: Adam Coates

51

Deep learning for neural networks



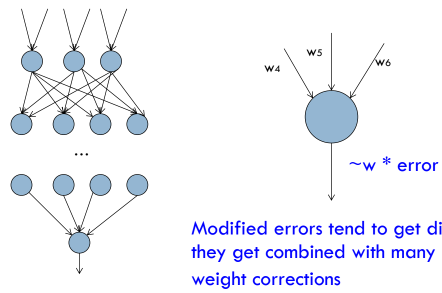
Traditional NN models: 1-2 hidden layers

Deep learning NN models: 3+ hidden layers

52

Challenges

What makes "deep learning" hard for NNs?



Modified errors tend to get diluted as they get combined with many layers of weight corrections

53

Deep learning

Growing field

Driven by:
- Increase in data availability
- Increase in computational power
- Parallelizability of many of the algorithms

Involves more than just neural networks (though, they're a very popular model)

54

word2vec

How many people have heard of it?

What is it?

55

Word representations generalized

Project words into a multi-dimensional "meaning" space

word ➡ $[x_1, x_2, \ldots, x_d]$

What was our projection for assignment 5?

56

Word representations generalized

Project words into a multi-dimensional "meaning" space

word ➡ $[w_1, w_2, \ldots, w_d]$

Each dimension is the co-occurrence of *word* with w_i

57

Word representations

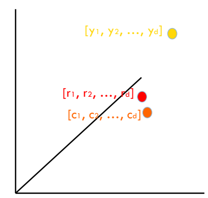
Project words into a multi-dimensional "meaning" space

word ➡ $[x_1, x_2, \ldots, x_d]$

red ➡ $[r_1, r_2, \ldots, r_d]$

crimson ➡ $[c_1, c_2, \ldots, c_d]$

yellow ➡ $[y_1, y_2, \ldots, y_d]$



58

Word representations

Project words into a multi-dimensional "meaning" space

word ➡ $[x_1, x_2, \ldots, x_d]$

The idea of word representations is not new:

- Co-occurrence matrices
- Latent Semantic Analysis (LSA)

New idea: learn word representation using a task-driven approach

59

A prediction problem

I like to eat bananas with cream cheese

Given a context of words

Predict what words are likely to occur in that context

60

A prediction problem

Given text, can generate lots of examples:

I like to eat bananas with cream cheese

input	prediction
___ like to eat	I
I ___ to eat bananas	like
I like ___ eat bananas with	to
I like to ___ bananas with cream	eat
…	…

61

A prediction problem

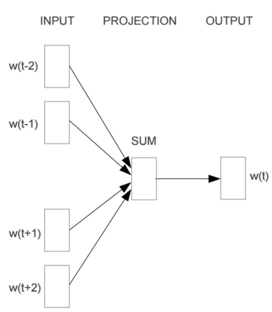
Use data like this to learn a distribution:

$$p(word \mid context)$$

$$p(w_i \mid \underbrace{w_{i-2}w_{i-1}}_{\text{words before}}\underbrace{w_{i+1}w_{i+2}}_{\text{words after}})$$

62

Train a neural network on this problem

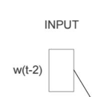


https://arxiv.org/pdf/1301.3781v3.pdf

63

Encoding words

How can we input a "word" into a network?

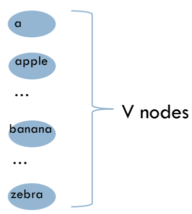


64

"One-hot" encoding

For a vocabulary of V words, have V input nodes

All inputs are 0 except the for the one corresponding to the word

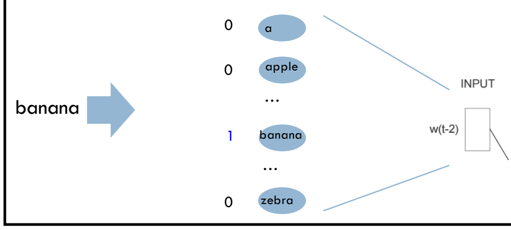


65

"One-hot" encoding

For a vocabulary of V words, have V input nodes

All inputs are 0 except the for the one corresponding to the word

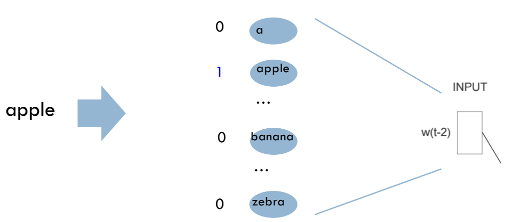


66

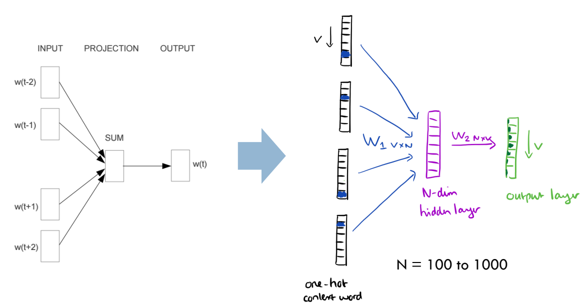
"One-hot" encoding

For a vocabulary of V words, have V input nodes

All inputs are 0 except the for the one corresponding to the word



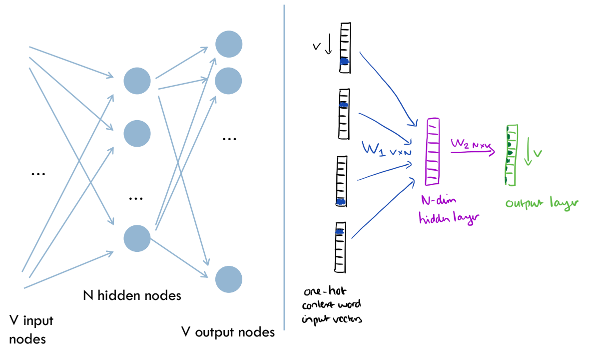
67



https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

68

Another view



69

Training: backpropagation

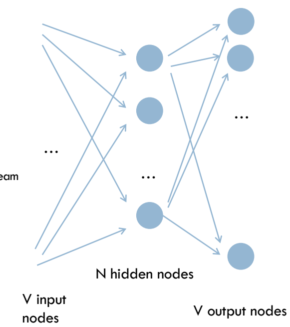
___ like to eat

I ___ to eat bananas

I like ___ eat bananas with

I like to ___ bananas with cream

…



I

like

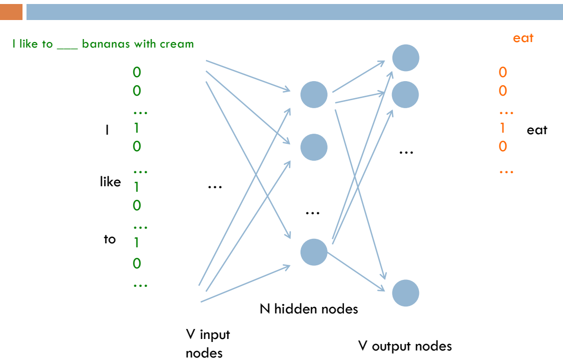
to

eat

…

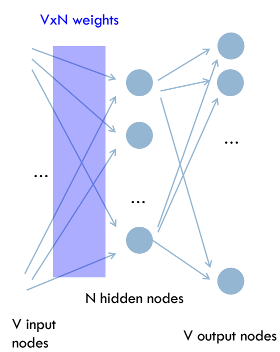
70

Training: backpropagation



71

Word representation



The weights for each word provide an N dimensional mapping of the word

Words that predict similarly should have similar weights

72

Results

vector(word1) – vector(word2) = vector(word3) - X

word1 is to word2 as word3 is to X

Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter

73

Results

vector(word1) – vector(word2) = vector(word3) - X

word1 is to word2 as word3 is to X

Type of relationship	Word Pair 1		Word Pair 2	
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

74

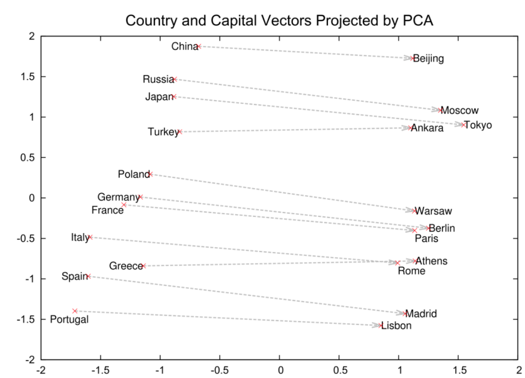
Results

vector(word1) – vector(word2) = vector(word3) - X

word1 is to word2 as word3 is to X

Newspapers			
New York	New York Times	Baltimore	Baltimore Sun
San Jose	San Jose Mercury News	Cincinnati	Cincinnati Enquirer
NHL Teams			
Boston	Boston Bruins	Montreal	Montreal Canadiens
Phoenix	Phoenix Coyotes	Nashville	Nashville Predators
NBA Teams			
Detroit	Detroit Pistons	Toronto	Toronto Raptors
Oakland	Golden State Warriors	Memphis	Memphis Grizzlies
Airlines			
Austria	Austrian Airlines	Spain	Spainair
Belgium	Brussels Airlines	Greece	Aegean Airlines
Company executives			
Steve Ballmer	Microsoft	Larry Page	Google
Samuel J. Palmisano	IBM	Werner Vogels	Amazon

75



2-Dimensional projection of the N-dimensional space

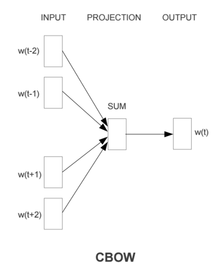
76

Visualized

https://projector.tensorflow.org/

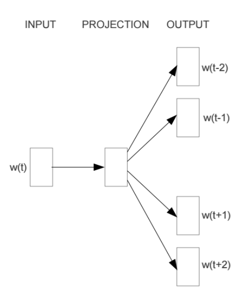
77

Continuous Bag Of Words



78

Other models: skip-gram



79

word2vec

A model for learning word representations from large amounts of data

Has become a popular pre-processing step for learning a more robust feature representation

Models like word2vec have also been incorporated into other learning approaches (e.g. translation tasks)

80

word2vec resources

https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/

https://code.google.com/archive/p/word2vec/

https://deeplearning4j.org/word2vec

https://arxiv.org/pdf/1301.3781v3.pdf

81