Human Language Technology:
Application to Information Access

# Lesson 4
# Deep learning for NLP:
# Word Representation Learning

*October 20, 2016*

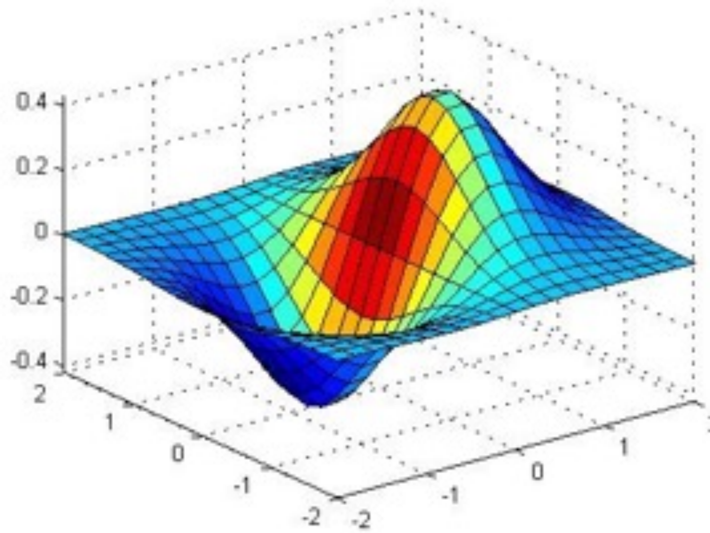
EPFL Doctoral Course EE-724
Nikolaos Pappas
Idiap Research Institute, Martigny

# Outline of the talk

1. Introduction and Motivation
2. Neural Networks - The basics
3. Word Representation Learning
4. Summary and Beyond Words

# Deep learning

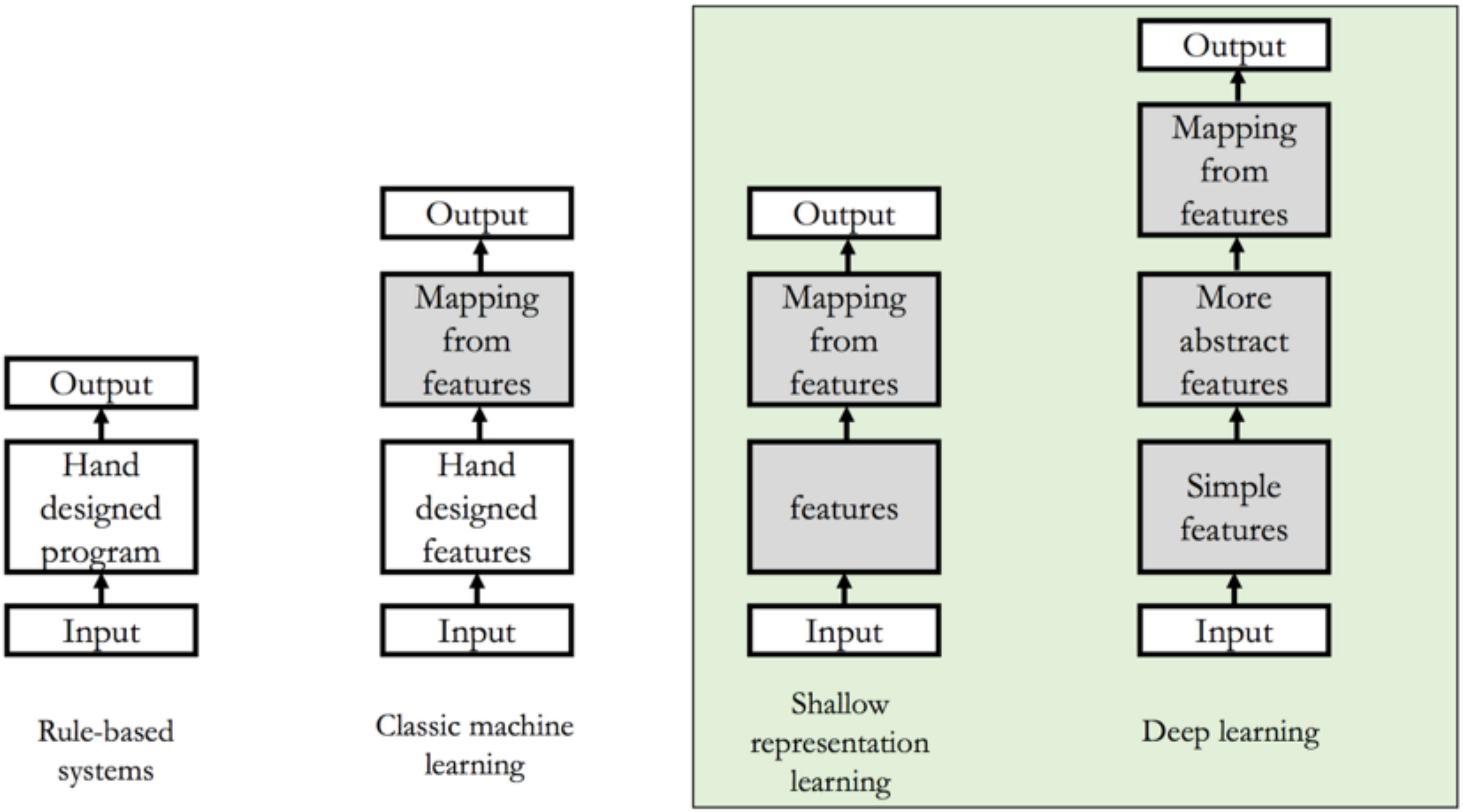
- Machine Learning boils down to minimizing an objective function to increase task performance
  - mostly relies on human-crafted features
  - e.g. topic, syntax, grammar, polarity




➡ **Representation Learning**: attempts to learn automatically good features or representations

➡ **Deep Learning:** machine learning algorithms based on multiple levels of representation or abstraction

# Key point: Learning multiple levels of representation



# Motivation for exploring deep learning: Why care?

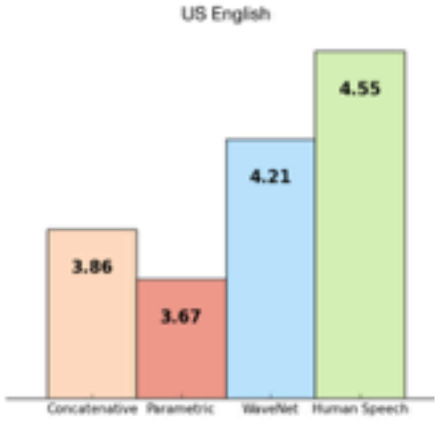
- Human crafted features are time-consuming, rigid, and often incomplete
- Learned features are easy to adapt and learn
- Deep Learning provides a very flexible, unified, and learnable framework that can handle a variety of input, such as vision, speech, and language.
  - **unsupervised** from raw input (e.g. text)
  - **supervised** with labels by humans (e.g. sentiment)

# Motivation for exploring deep learning: Why now?

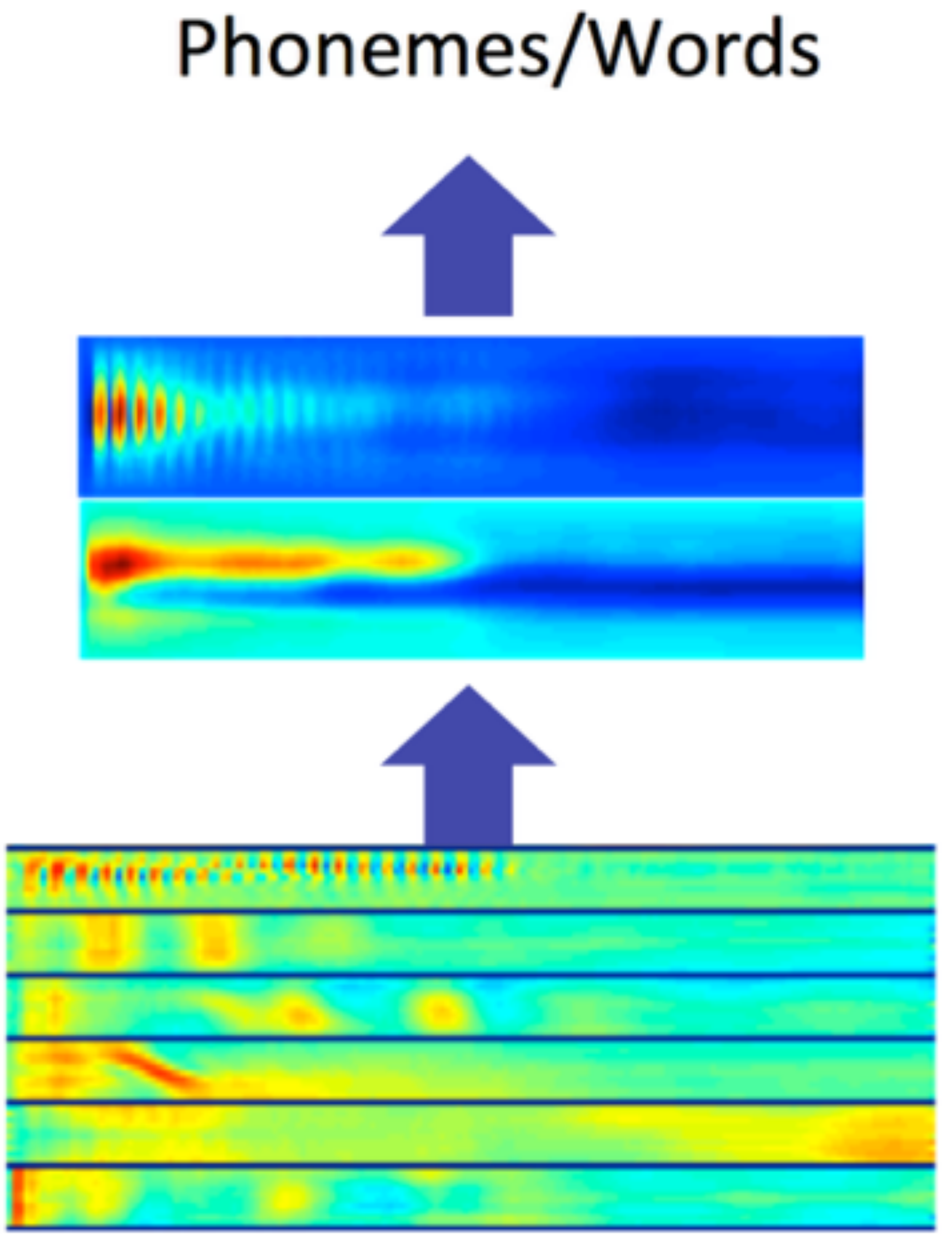
- What enabled deep learning techniques to start outperforming other machine learning techniques since Hinton et al. 2006?
  - Larger amounts of data
  - Faster computers and multicore cpu and gpu
  - New models, algorithms and improvements over "older" methods (**speech**, **vision** and **language**)

# Deep learning for speech: Phoneme detection

- The first breakthrough results of "deep learning" on large datasets by Dahl et al. 2010
    - -30% reduction of error
- Most recently on speech synthesis Oord et al. 2016

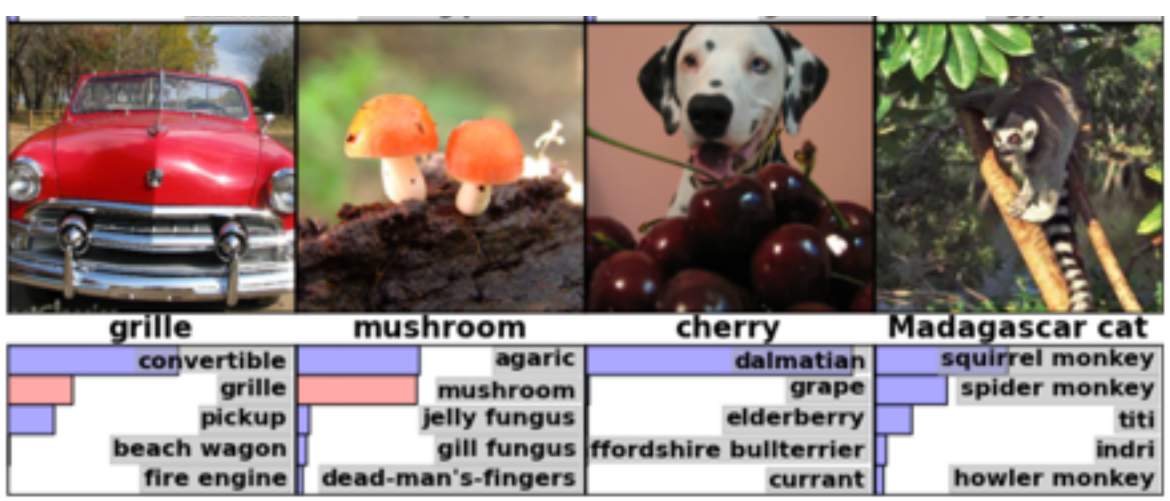


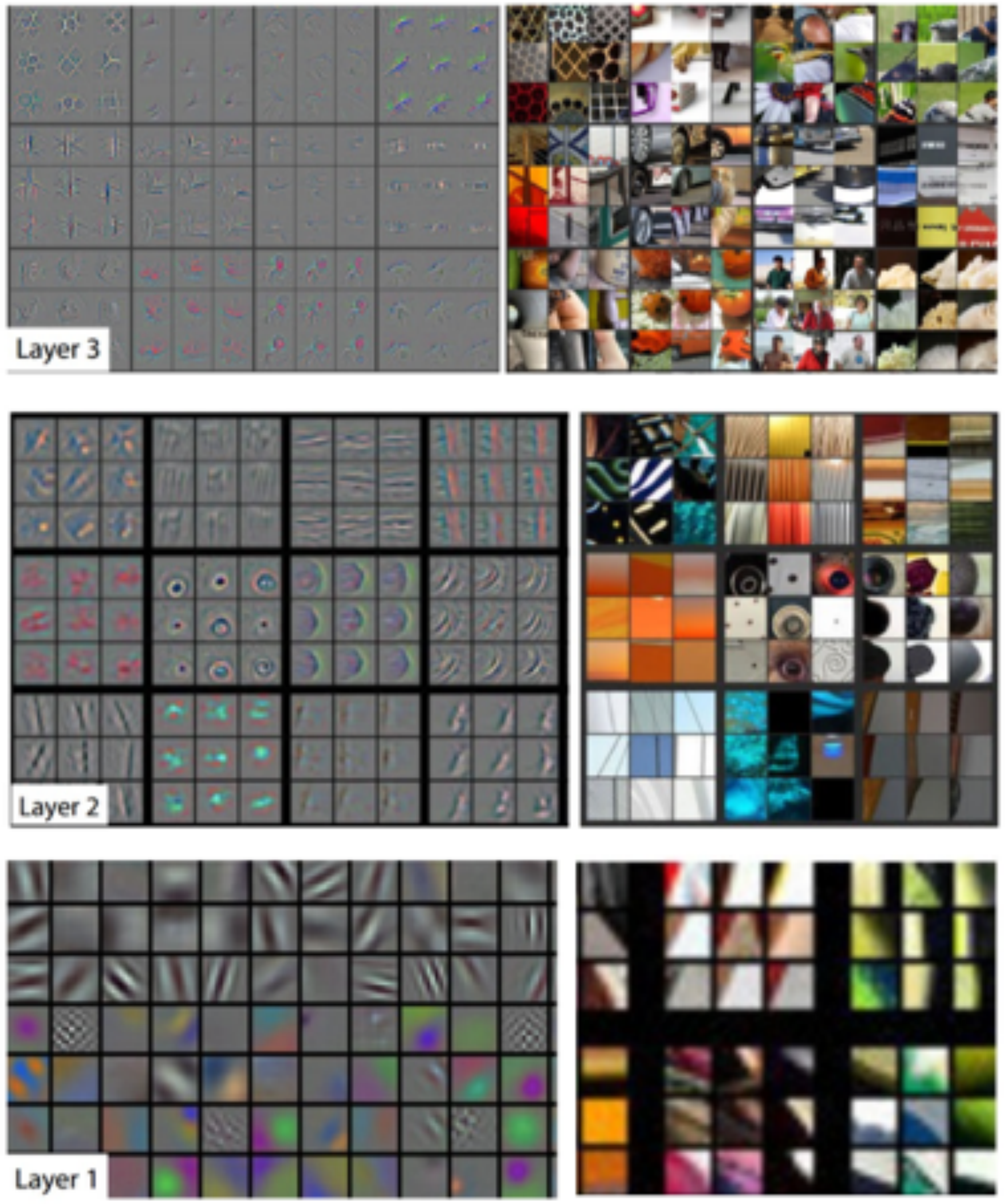




# Deep learning for vision: Object detection

- Popular topic for DL
- Breakthrough on ImageNet by Krizhevsky et al. 2012
  - -21% and -51% error reduction at top 1 and 5

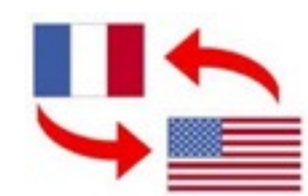



Zeiler and Fergus (2013)

# Deep learning for language: Ongoing

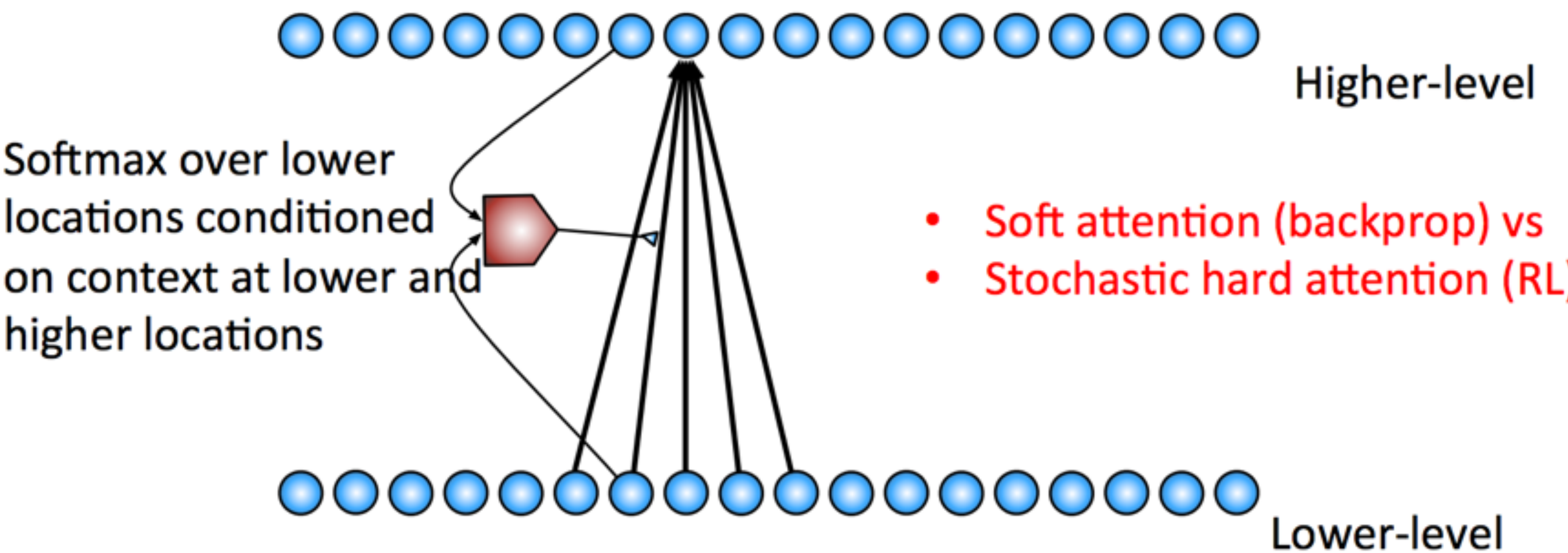
- Significant improvements in recent years across different levels (phonology, morphology, syntax, semantics) and applications in NLP
  - **Machine translation** (most notable)
  - **Question answering**
  - **Sentiment classification**
  - **Summarization**



Still a lot of work to be done… e.g. metrics
(beyond “basic” recognition - attention, reasoning, planning)

# Attention mechanism for deep learning

- Operates on input or intermediate sequence
- Chooses "where to look" or learns to assign a relevance to each input position — essentially parametric pooling



# Deep learning for language: Machine Translation

- Reached the state-of-the-art in one year: Bahdanau et al. 2014, Jean et al. 2014, Gulcehre et al. 2015

(a) **English→French (WMT-14)**

| | NMT(A) | Google | P-SMT |
|---|---|---|---|
| NMT | 32.68 | 30.6* | **37.03•** |
| +Cand | 33.28 | – | |
| +UNK | 33.99 | 32.7° | |
| +Ens | **36.71** | **36.9°** | |

(b) **English→German (WMT-15)**

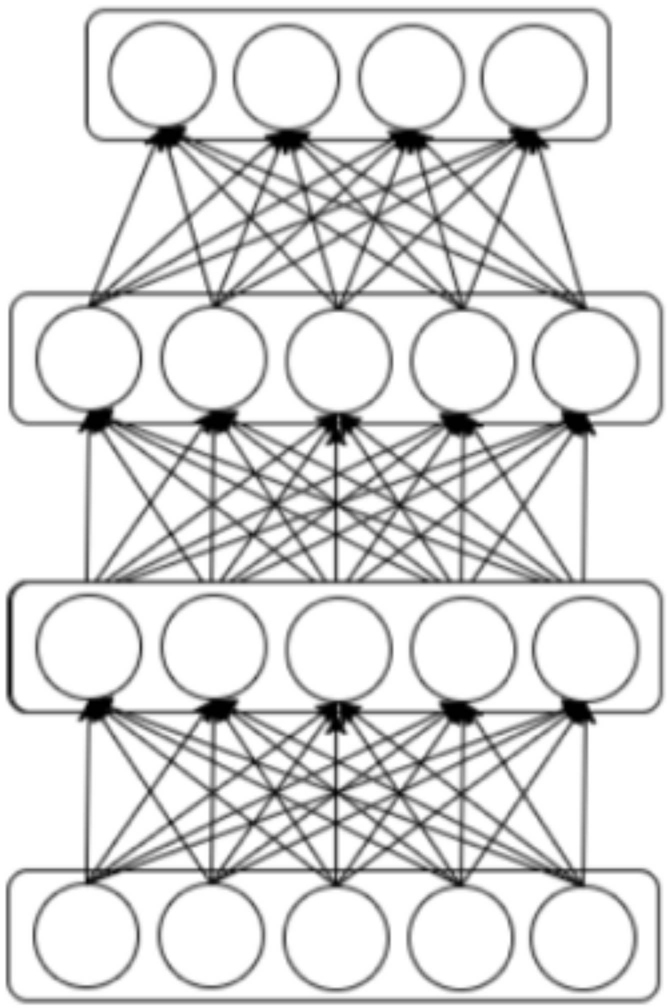
| Model | Note |
|---|---|
| **24.8** | Neural MT |
| 24.0 | U.Edinburgh, Syntactic SMT |
| 23.6 | LIMSI/KIT |
| 22.8 | U.Edinburgh, Phrase SMT |
| 22.7 | KIT, Phrase SMT |

(c) **English→Czech (WMT-15)**

| Model | Note |
|---|---|
| **18.3** | Neural MT |
| 18.2 | JHU, SMT+LM+OSM+Sparse |
| 17.6 | CU, Phrase SMT |
| 17.4 | U.Edinburgh, Phrase SMT |
| 16.1 | U.Edinburgh, Syntactic SMT |

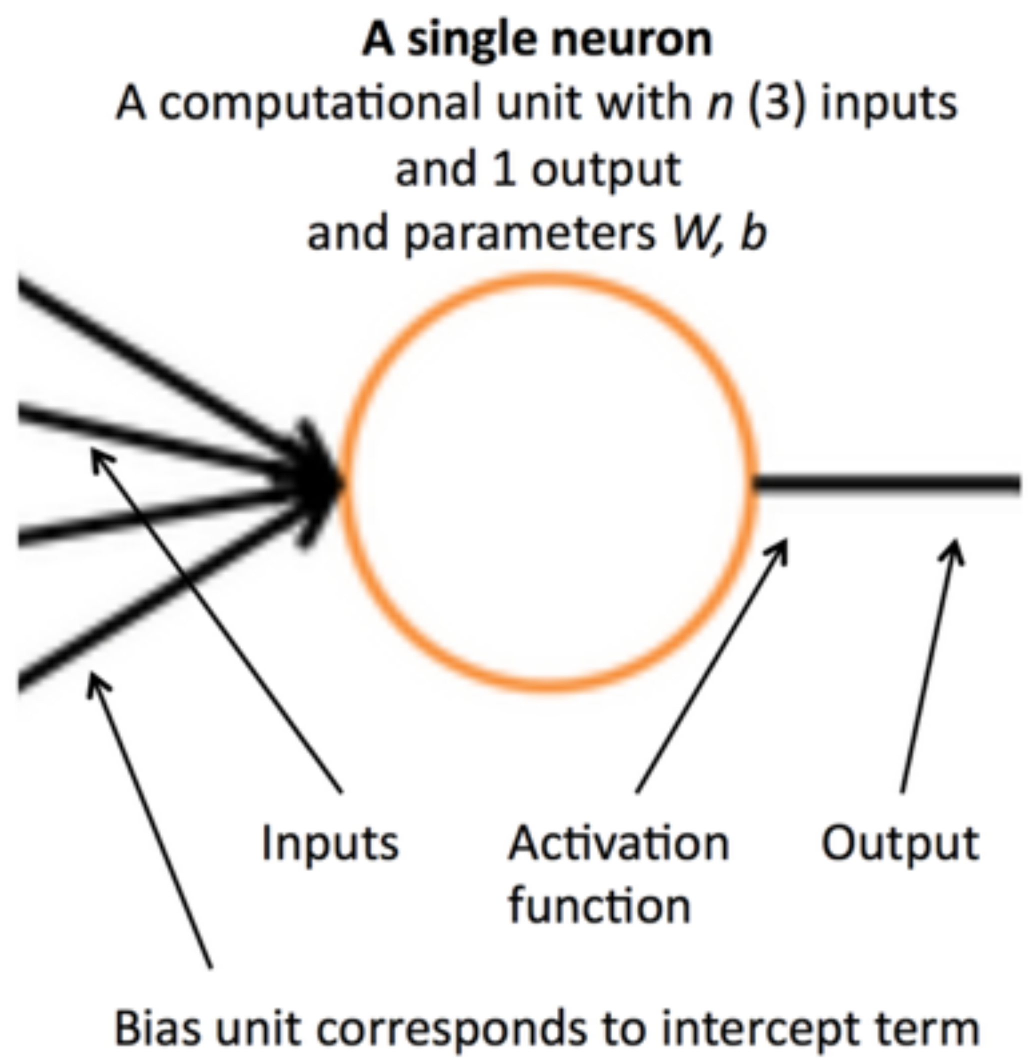
# Outline of the talk

1. Neural Networks
    - Basics: perceptron, logistic regression
    - Learning the parameters
    - Advanced models: spatial and temporal / sequential
2. Word Representation Learning
    - Semantic similarity
    - Traditional and recent approaches
    - Intrinsic and extrinsic evaluation
3. Summary and Beyond



# Introduction to neural networks

- Biologically inspired from how the human brain works
  - Seems to have a generic learning algorithm
  - Neurons activate in response to inputs and produce excite other neurons

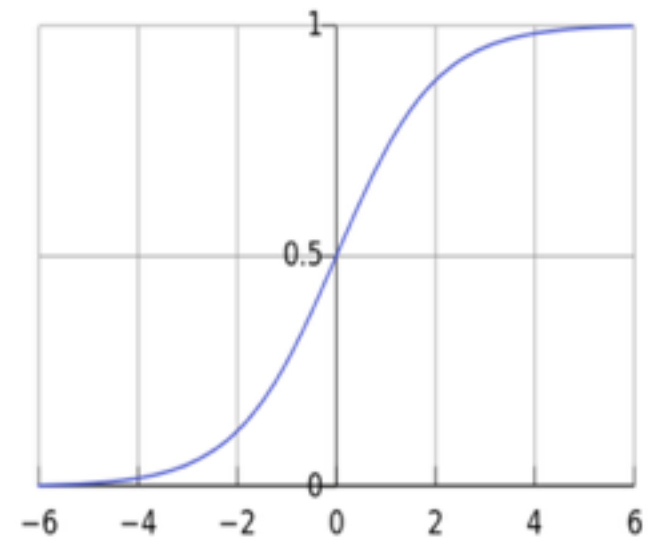


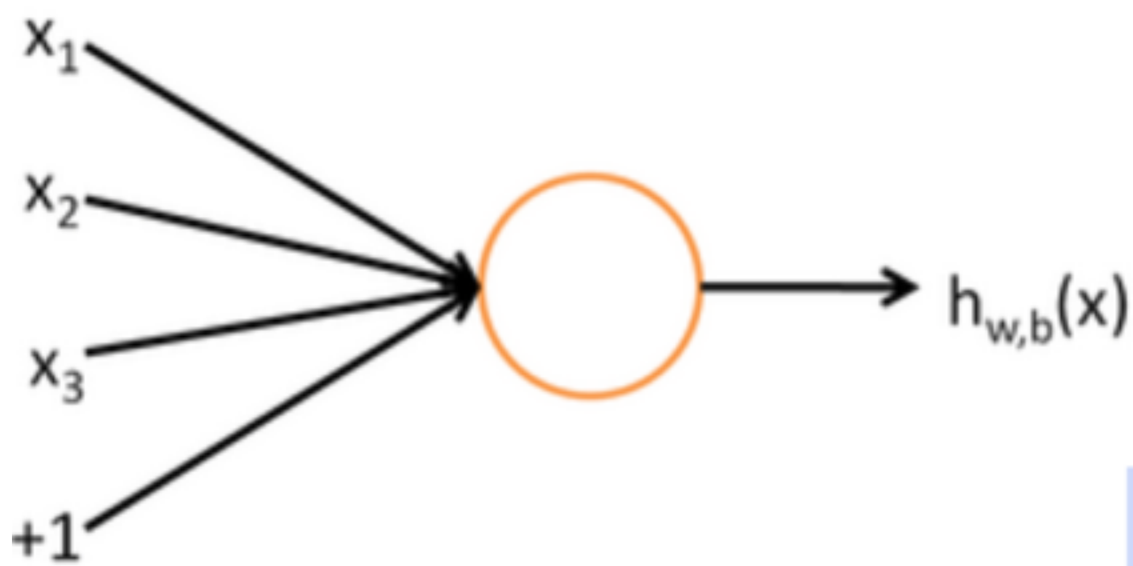
# Artificial neuron or Perceptron

$$h_{w,b}(x) = f(w^{\top}x + b)$$

*b:* We can have an “always on” feature, which gives a class prior, or separate it out, as a bias term

$$f(z) = \frac{1}{1+e^{-z}}$$

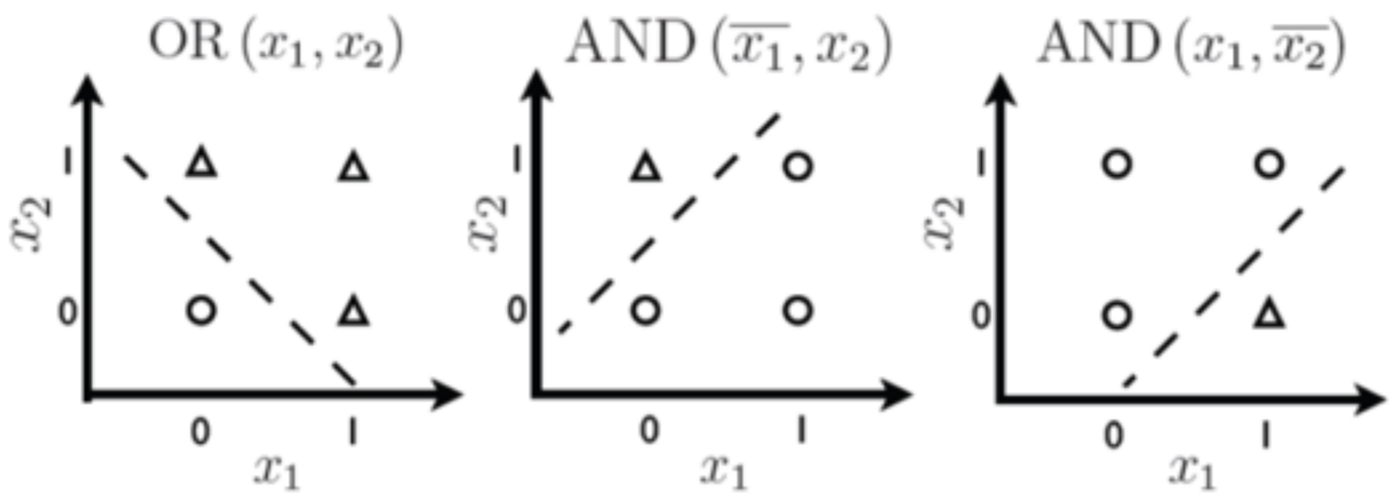


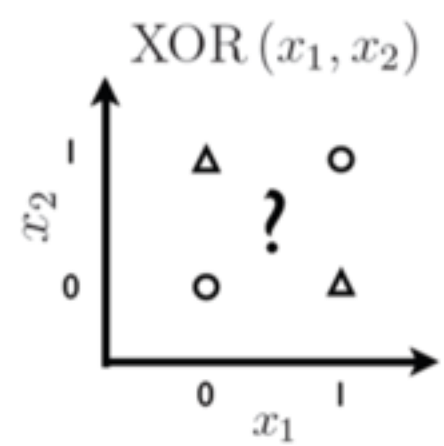

*w*, *b* are the parameters of this neuron i.e., this logistic regression model

# What can a perceptron do?

- Solve linearly separable problems




- … but not non-linearly separable ones.



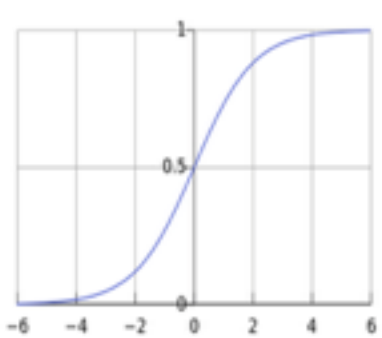
# From logistic regression to neural networks

Vector form: $$P(c \mid d, \lambda) = \frac{e^{\lambda^{\mathsf{T}} f(c,d)}}{\sum_{c'} e^{\lambda^{\mathsf{T}} f(c',d)}}$$

Make two class:

$$P(c_1 \mid d, \lambda) = \frac{e^{\lambda^{\mathsf{T}} f(c_1,d)}}{e^{\lambda^{\mathsf{T}} f(c_1,d)} + e^{\lambda^{\mathsf{T}} f(c_2,d)}} = \frac{e^{\lambda^{\mathsf{T}} f(c_1,d)}}{e^{\lambda^{\mathsf{T}} f(c_1,d)} + e^{\lambda^{\mathsf{T}} f(c_2,d)}} \cdot \frac{e^{-\lambda^{\mathsf{T}} f(c_1,d)}}{e^{-\lambda^{\mathsf{T}} f(c_1,d)}}$$

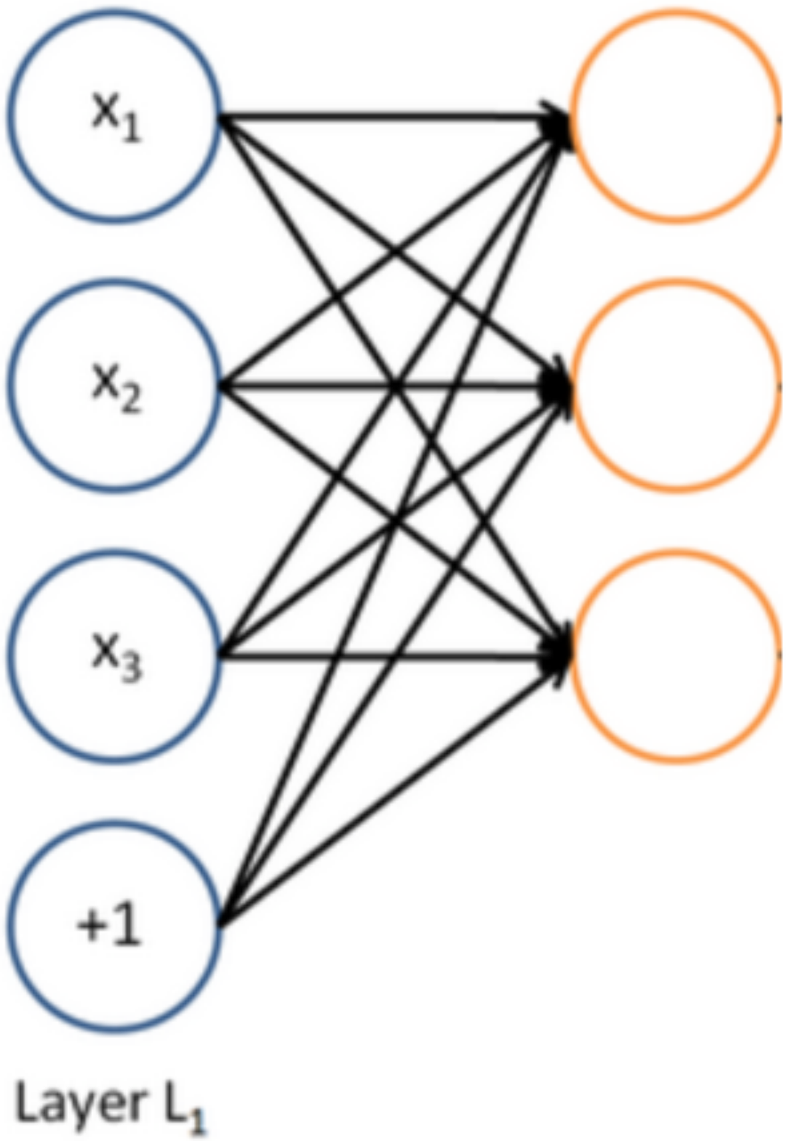
$$= \frac{1}{1 + e^{\lambda^{\mathsf{T}}[f(c_2,d) - f(c_1,d)]}} = \frac{1}{1 + e^{-\lambda^{\mathsf{T}} x}}$$ for $x = f(c_1,d) - f(c_2,d)$




$$= f(\lambda^{\mathsf{T}} x)$$

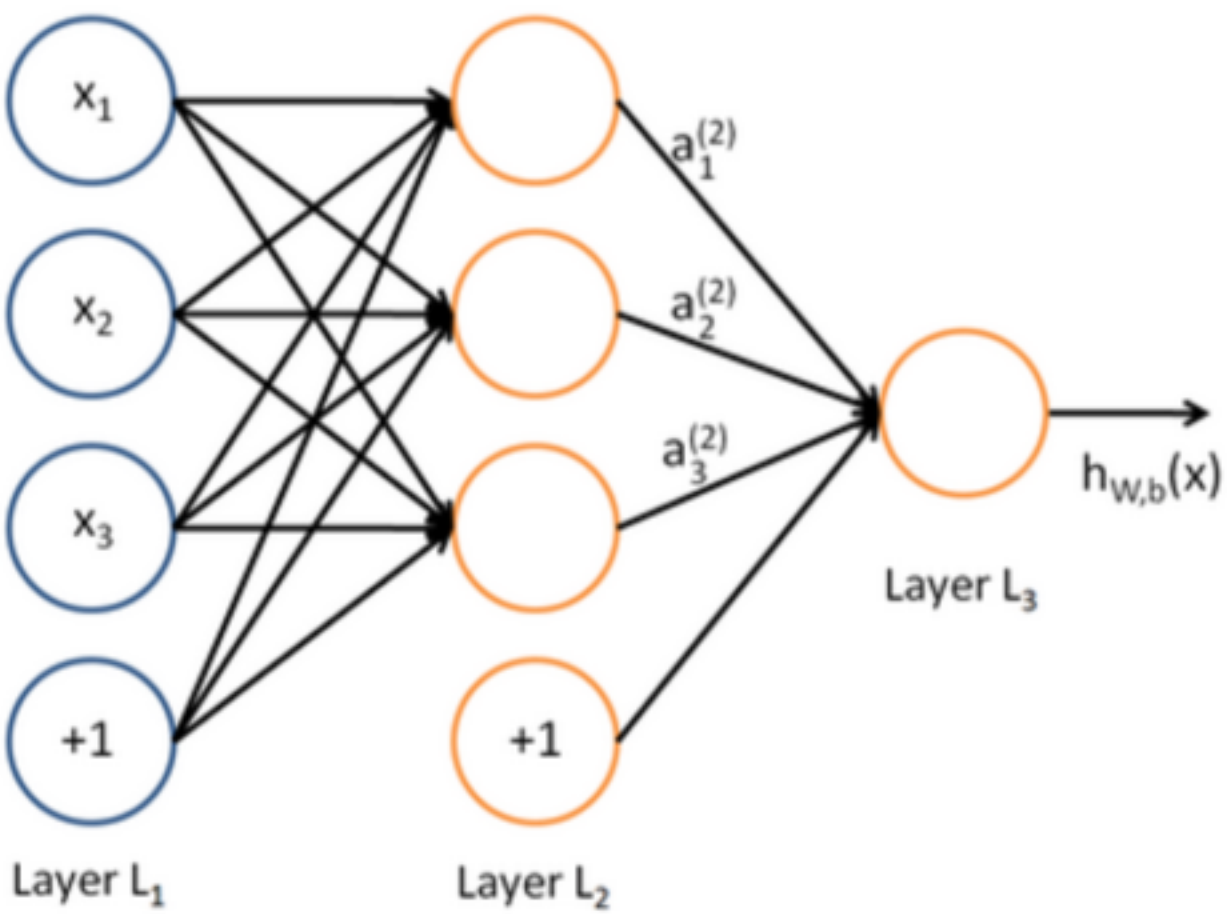
for $f(z) = 1/(1 + \exp(-z))$, the logistic function – a sigmoid non-linearity.

# A neural network: several logistic regressions at the same time



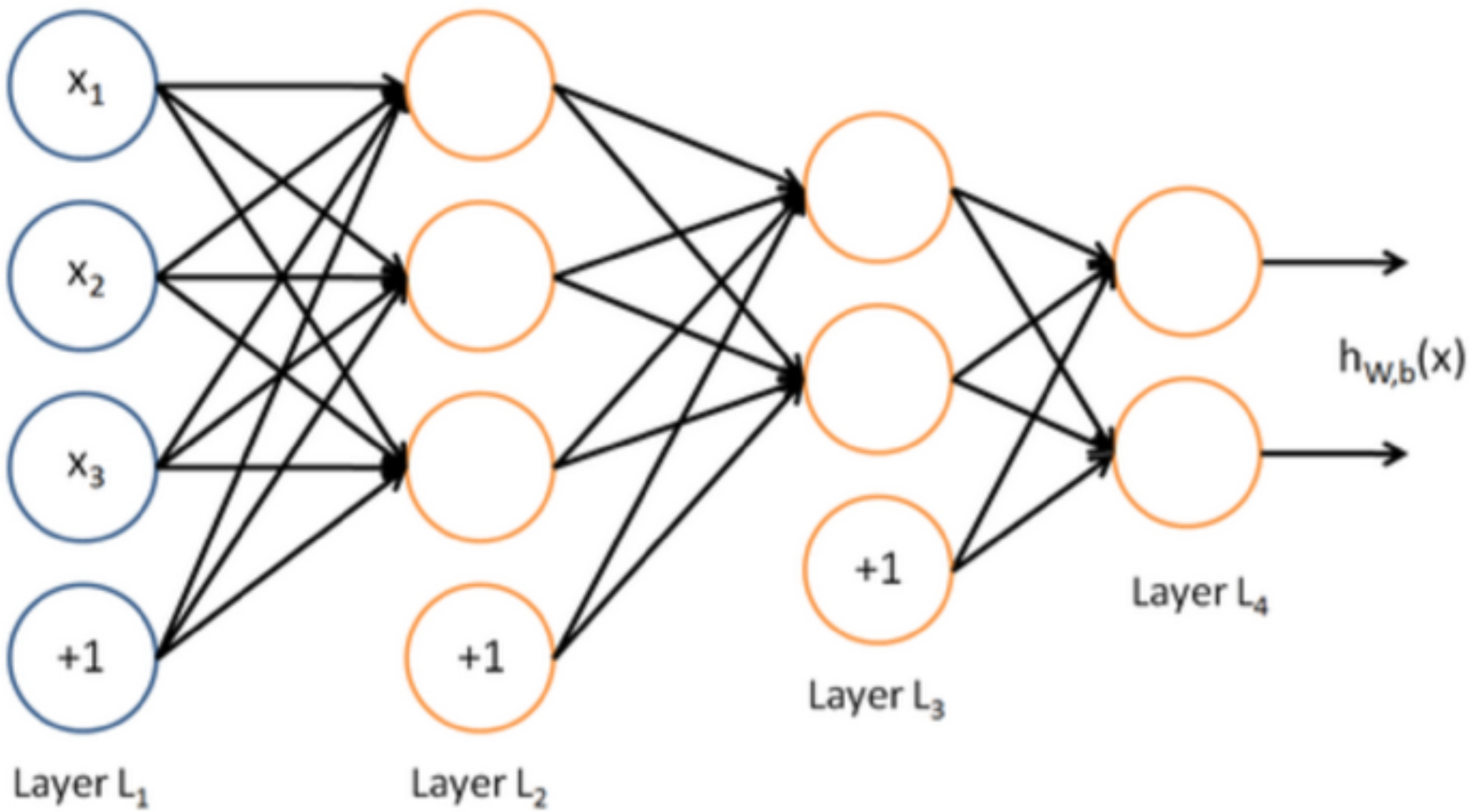

- Apply several regressions to obtain a vector of outputs
- The values of the outputs are initially unknown
    - <u>No need to specify</u> ahead of time what values the logistic regressions are trying to predict

# A neural network: several logistic regressions at the same time




- The intermediate variables are learned directly based on the training objective
- This makes them do a good job at predicting the target for the next layer
- Result: able to model non-linearities in the data!

# A neural network: extension to multiple layers



# A neural network: Matrix notation for a layer

We have

$$a_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1)$$
$$a_2 = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + b_2)$$

etc.

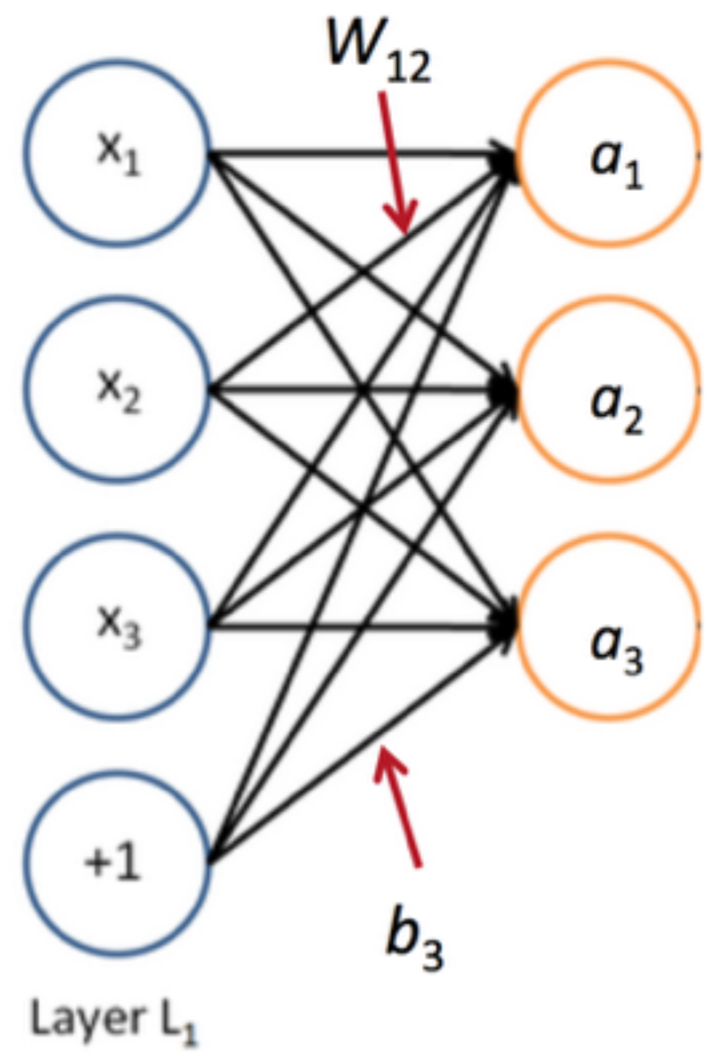
In matrix notation

$$z = Wx + b$$

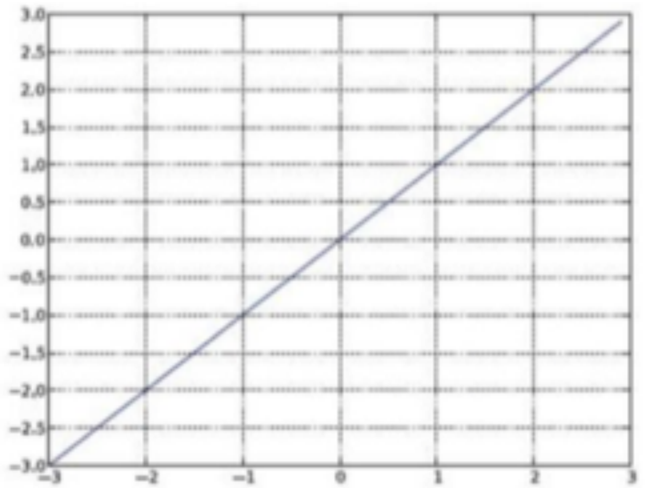
$$a = f(z)$$

where $f$ is applied element-wise:

$$f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$$

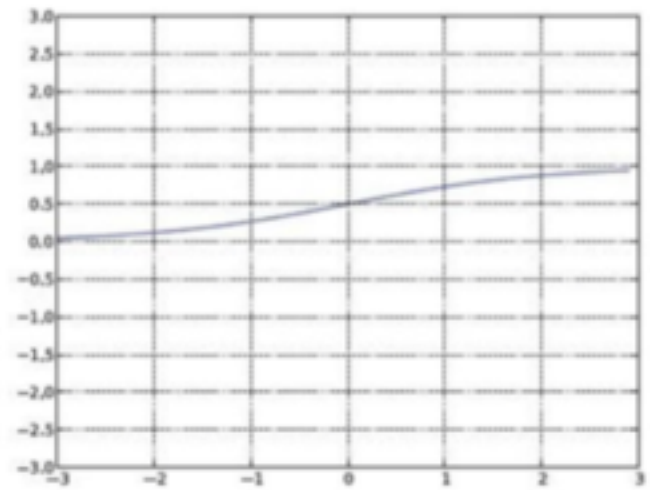


# Several activation functions to choose from



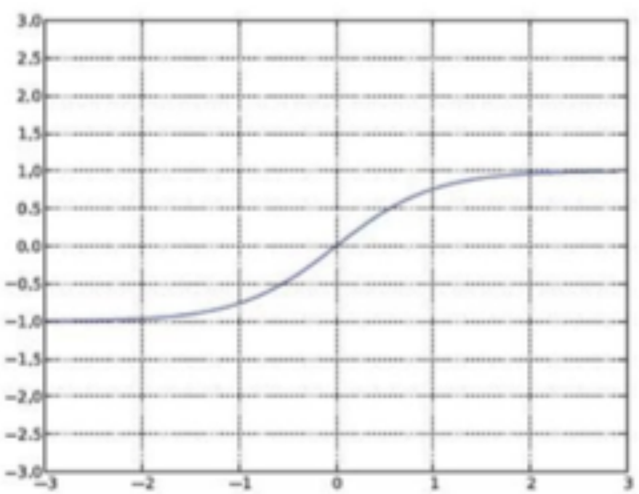
Identity

$$g(a) = a$$



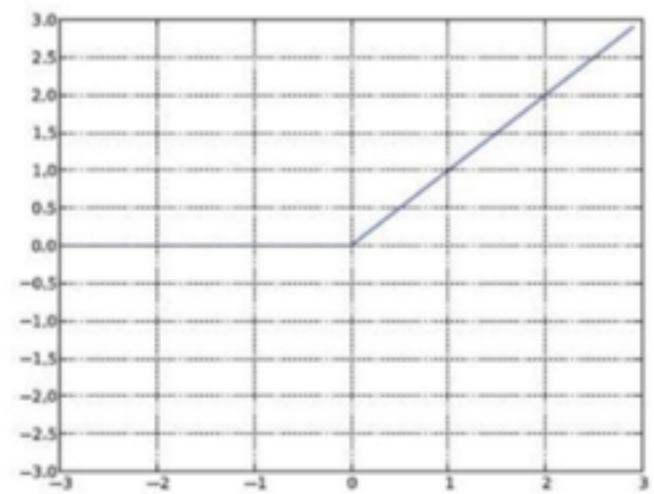
sigmoid

$$g(a) = sigm(a) = \frac{1}{1 + exp(a)}$$



tanh

$$g(a) = tanh(a) = \frac{exp(2a) - 1}{exp(2a) + 1}$$



relu

$$g(a) = relu(a) = max(0, a)$$

14

***Images from Hugo Larochelle's course.**

# Learning parameters using gradient descend

- Given training data $\mathcal{D} = \{x^{(i)}, y^{(i)}\}_{i=1}^{N}$ find $W$ and $b$ that minimizes loss with respect to these parameters

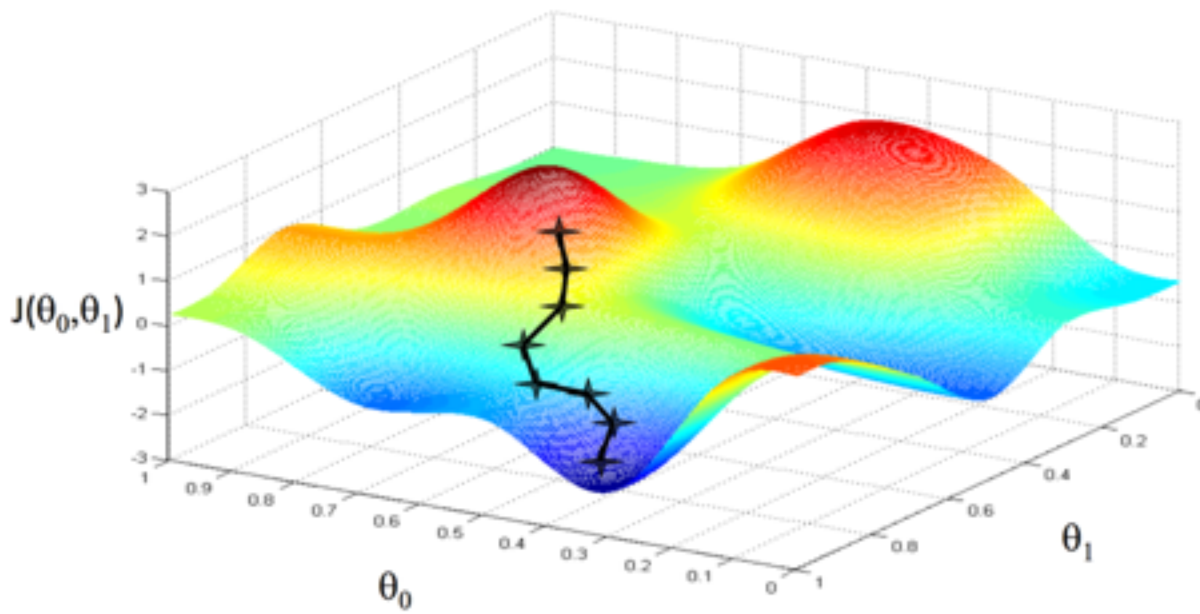
$$\mathcal{J}(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(g(a(x^{(i)})) - y^{(i)})^2$$

- Compute gradient with respect to parameters and make small step towards the direction of the negative gradient

$$W = W - \alpha \frac{\partial \mathcal{J}}{\partial W}$$

$$b = b - \alpha \frac{\partial \mathcal{J}}{\partial b}$$

$\alpha$ - learning rate or step size.



# Going large scale: Stochastic gradient descent (SGD)

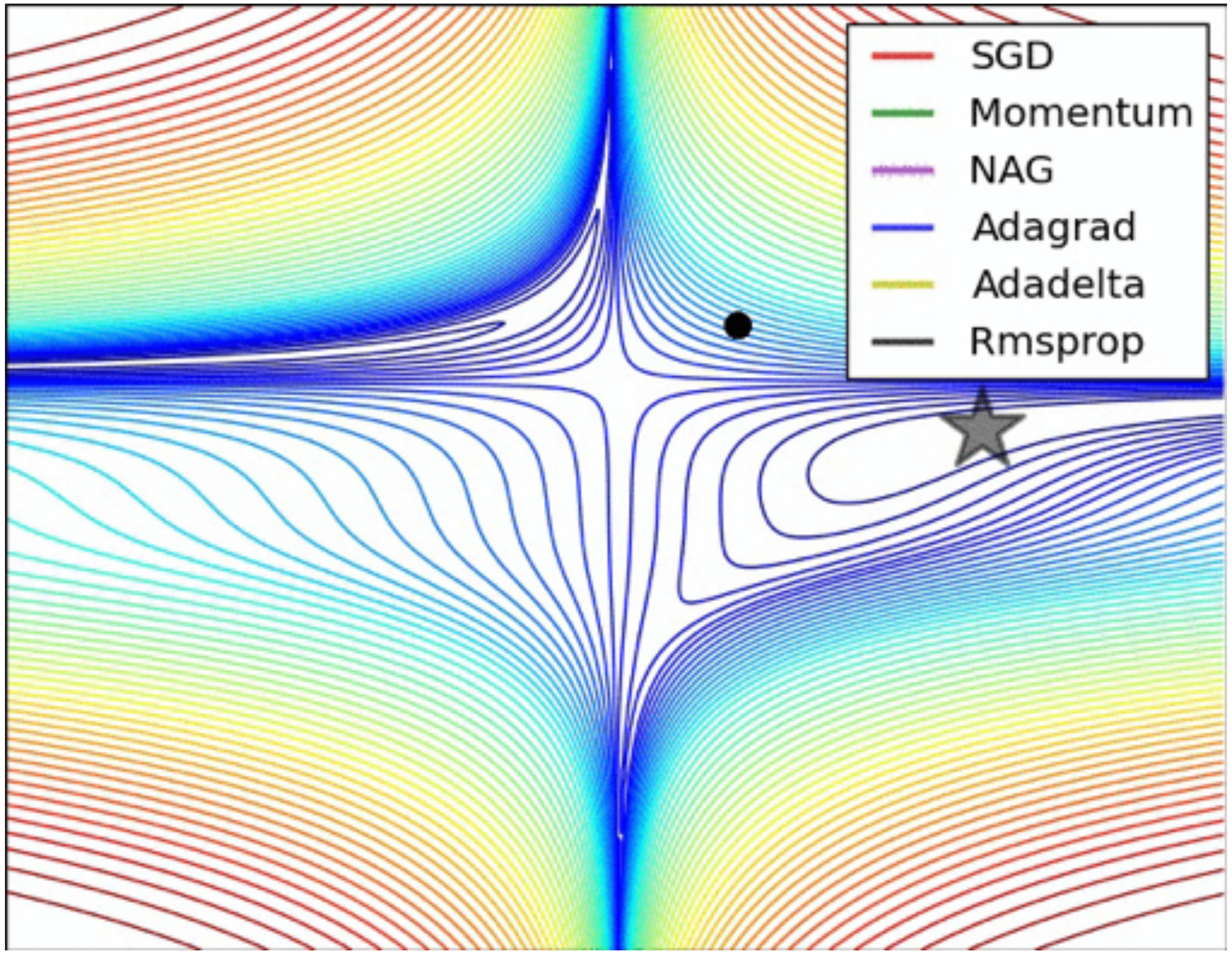
- Approximate the gradient using a mini-batch of examples instead of entire training set
- Online SGD when mini batch size is one
- Most commonly used when compared to GD

$$w_k = w_k - \alpha \; . \; (g(a(x^{(i)}) - y^{(i)}) \; . \; g'(a(x^{(i)})) \; . \; x_k^{(i)} \qquad for\ k = 1, .., d$$

$$b = b - \alpha \; . \; (g(a(x^{(i)}) - y^{(i)}) \; . \; g'(a(x^{(i)}))$$

# Learning parameters using gradient descend

- Several out-of-the-box strategies for decaying learning rate of an objective function:
    - Select the best according to validation set performance



# Training neural networks with arbitrary layers: Backpropagation

- We still minimize the objective function but this time we "backpropagate" the errors to all the hidden layers

- Chain rule: If *y* = *f*(*u*) and *u* = *g*(*x*), i.e. y=f(g(x)), then:

$$\frac{dy}{dx} = \frac{dy}{du}\frac{du}{dx} = \frac{df(u)}{du}\frac{dg(x)}{dx}$$

- Useful basic derivatives:

$$\frac{\partial \mathbf{x}^T\mathbf{a}}{\partial \mathbf{x}} = \frac{\partial \mathbf{a}^T\mathbf{x}}{\partial \mathbf{x}} = \mathbf{a}$$

Typically, backprop computation is implemented in popular libraries: **Theano**, **Torch**, **Tensorflow**

# Training neural networks with arbitrary layers: Backpropagation

Simple example: $$\frac{dy}{dx} = \frac{d}{dx}5(x^3+7)^4$$

$$y = f(u) = 5u^4 \qquad u = g(x) = x^3 + 7$$

$$\frac{dy}{du} = 20u^3 \qquad \frac{du}{dx} = 3x^2$$

$$\frac{dy}{dx} = 20(x^3+7)3x^2$$

# Advanced neural networks

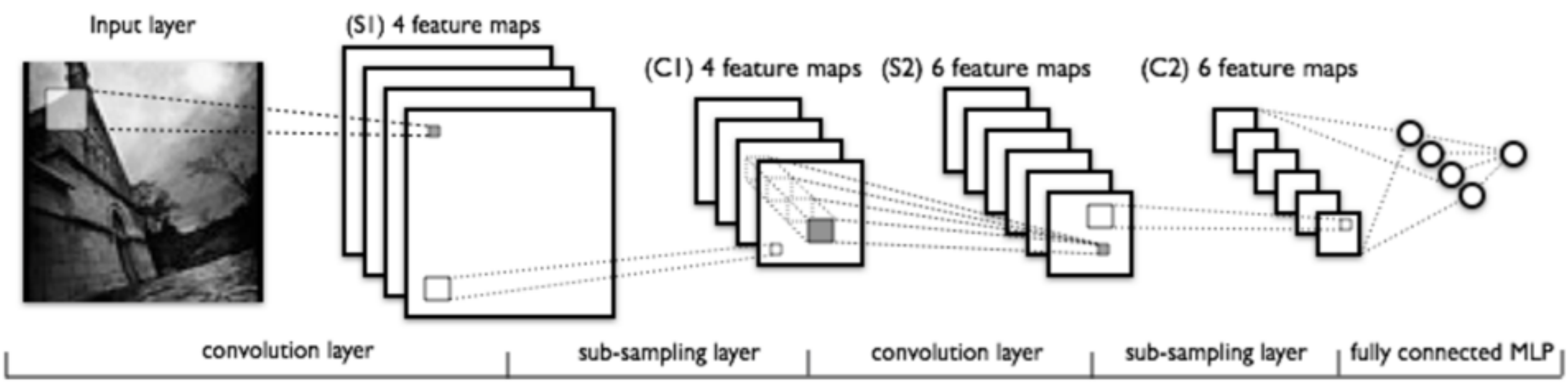
- Essentially, now we have all the basic “ingredients” we need to build deep neural networks
    - More layers more non-linear the final projection
    - Augmentation with new properties

➡ Advanced neural networks are able to deal with different arrangements of the input

- **Spatial**: convolutional networks
- **Sequential**: recurrent networks

# Spatial Modeling: Convolutional neural networks

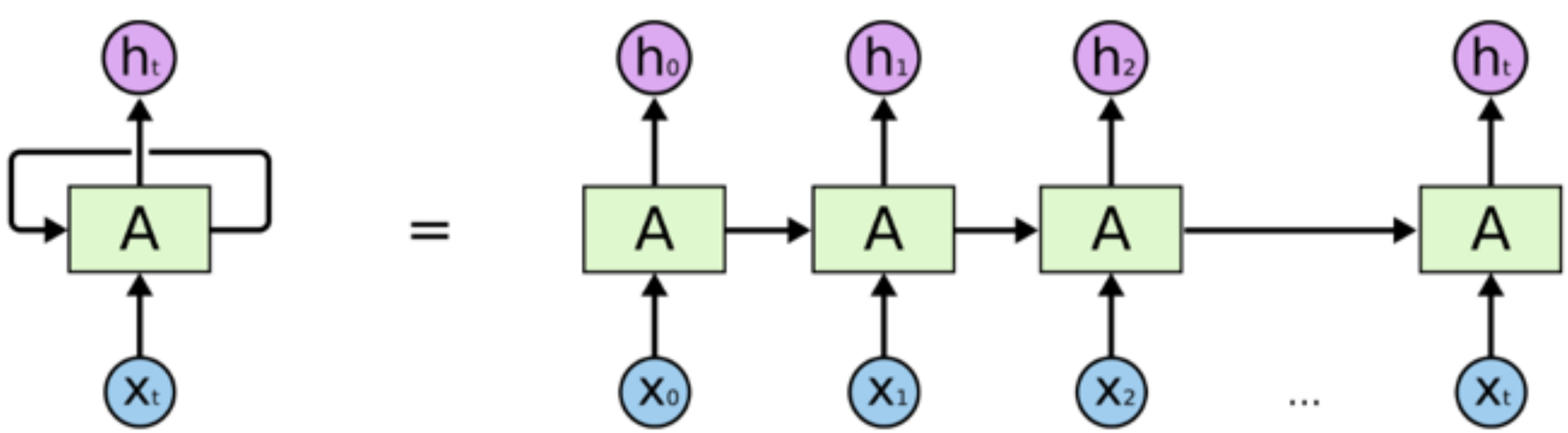
- Fully connected network to input pixels is not efficient
- Inspired by the organization of the animal visual cortex
  - assumes that the inputs are images
  - connects each neuron to a local region



# Sequence modeling: Recurrent neural networks

- Traditional networks can't model sequence information
    - lack of information persistence
- Recursion: Multiple copies of the same network where each one passes on information to its successor

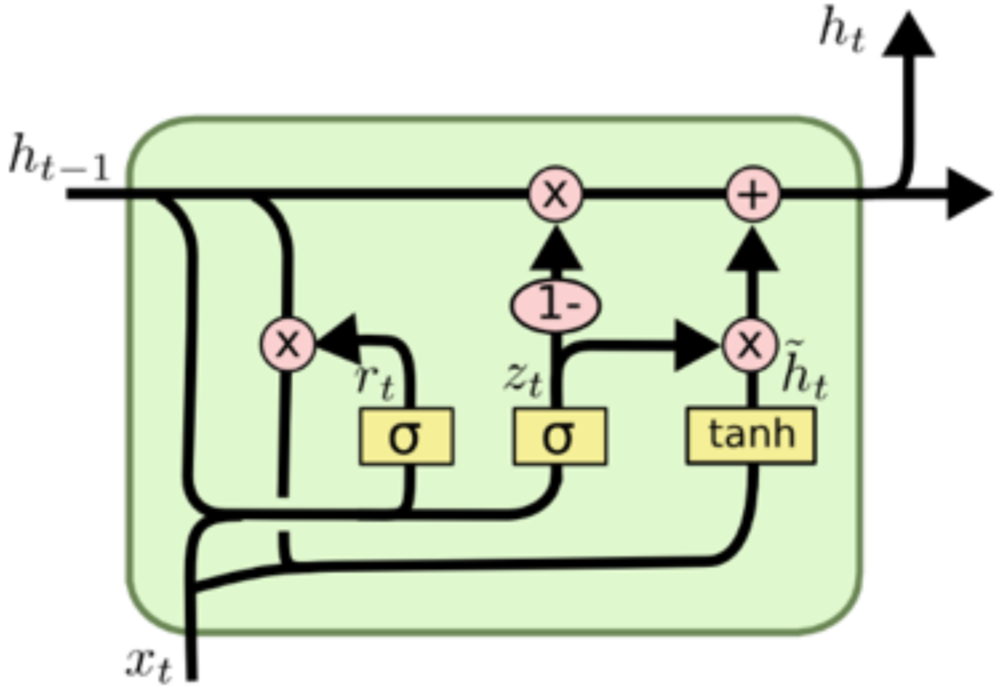
$$h_t = f_W(h_{t-1}, x_t)$$




* Diagram from Christopher Olah's blog.

# Sequence modeling: Gated recurrent networks

- Long-short term memory nets are able to learn long-term dependencies: Hochreiter and Schmidhuber 1997
- Gated RNN by Cho et al 2014 combines the forget and input gates into a single "update gate."




$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

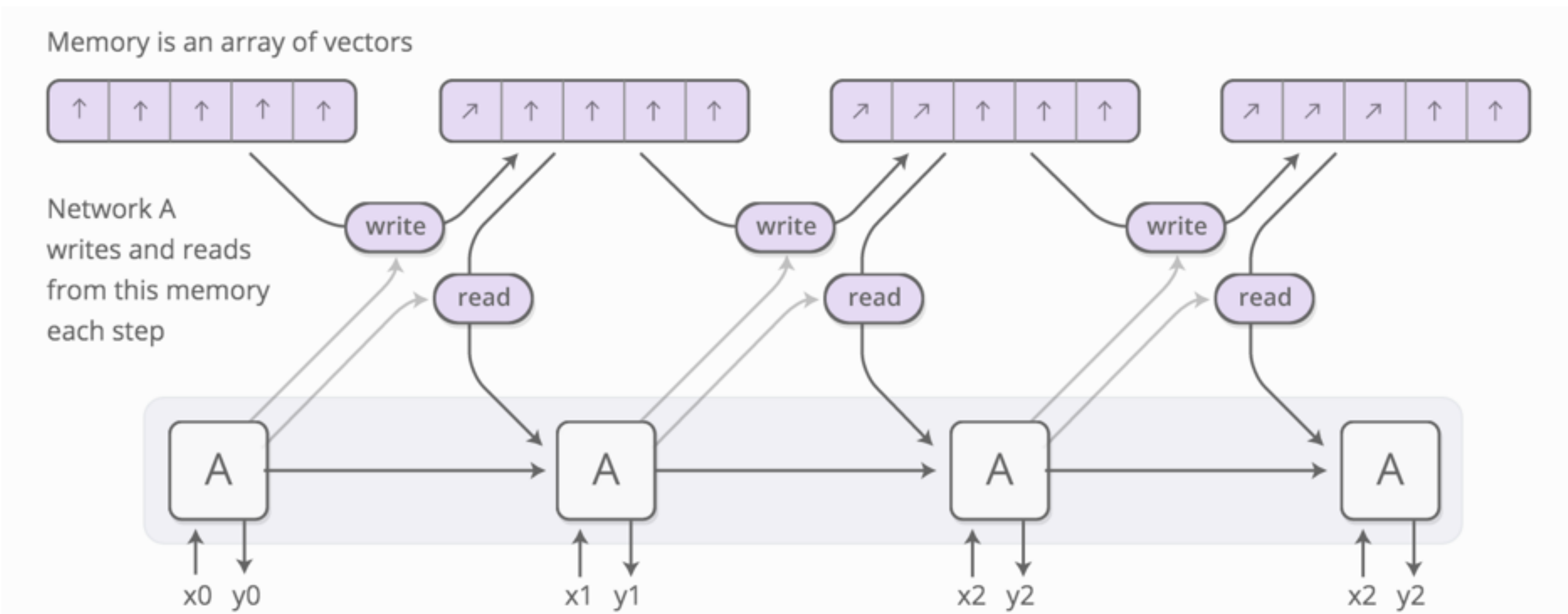
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

* Diagram from Christopher Olah's blog.

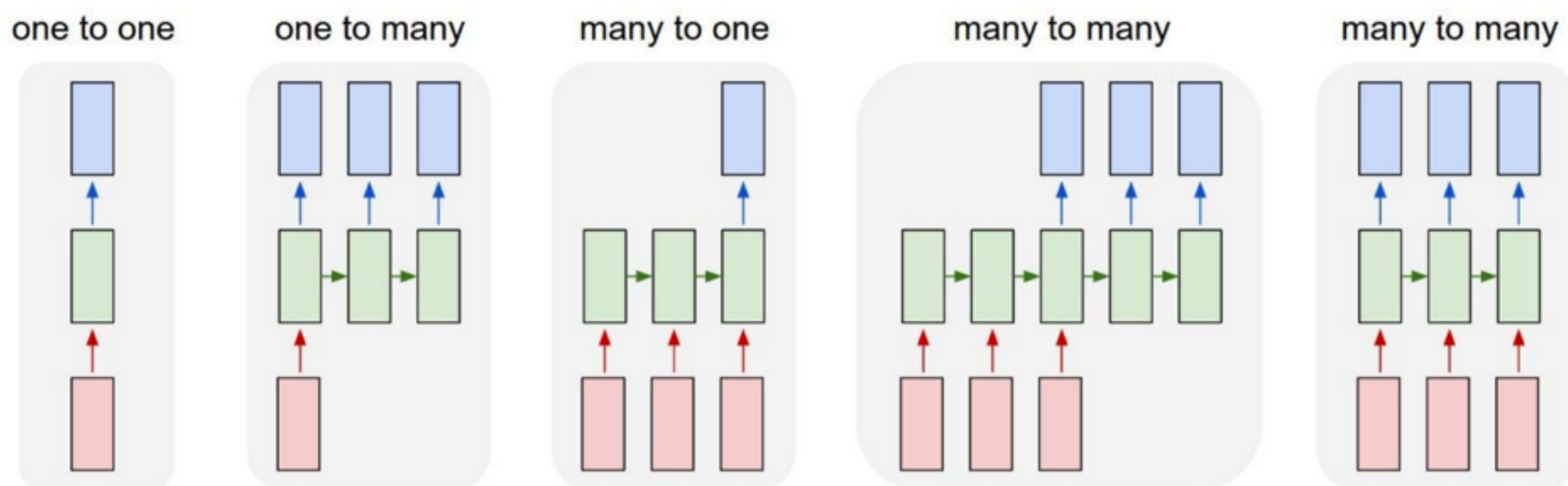
# Sequence modeling: Neural Turing Machines or Memory Networks

- Combination of recurrent network with external memory bank: Graves et al. 2014, Weston et.al 2014




* Diagram from Christopher Olah's blog.

# Sequence modeling: Recurrent neural networks are flexible




* Diagram from Karpathy's Stanford CS231n course.

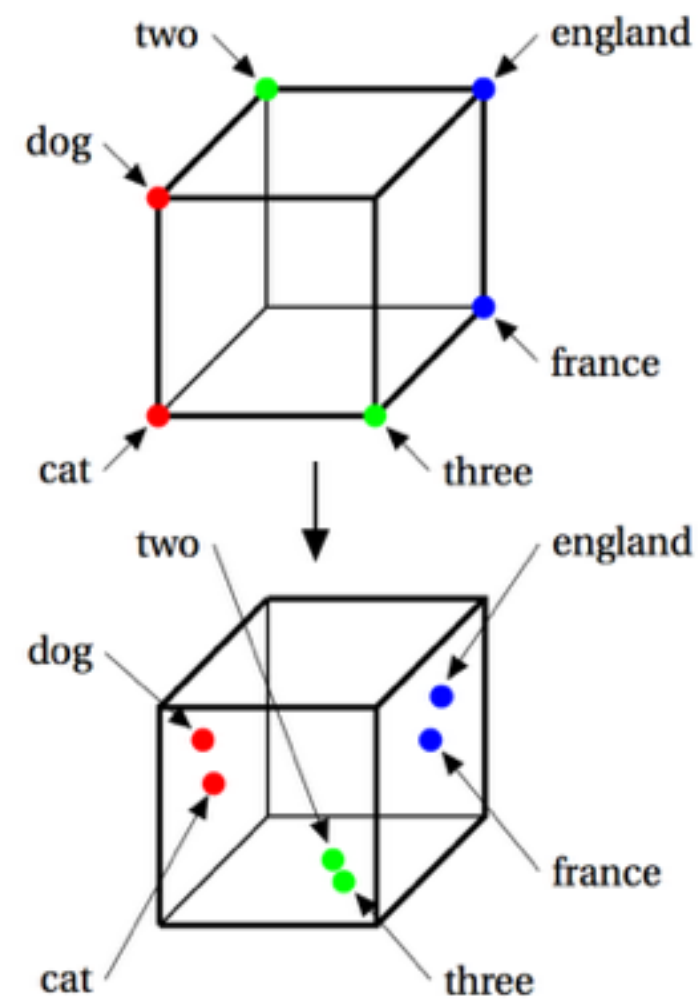
- Vanilla nns
- Image captioning
- Sentiment classification
- Topic detection
- Machine translation
- Summarization
- Speech recognition
- Video classification

# Outline of the talk

1. Neural Networks
    - Basics: perceptron, logistic regression
    - Learning the parameters
    - Advanced models: spatial and temporal / sequential

2. Word Representation Learning
    - Semantic similarity
    - Traditional and recent approaches
    - Intrinsic and extrinsic evaluation

3. Summary and Beyond




* image from Lebret's thesis (2016).

# Semantic similarity: How similar are two linguistic items?

- **Word level**

  *screwdriver —?—> wrench* very similar
  *screwdriver —?—> hammer* little similar
  *screwdriver —?—> technician* related
  *screwdriver —?—> fruit* unrelated

- **Sentence level**

  *The boss fired the worker*
  *The supervisor let the employee go* very similar
  *The boss reprimanded the worker* little similar
  *The boss promoted the worker* related
  *The boss went for jogging today* unrelated

# Semantic similarity: How similar are two linguistic items?

- Defined in many levels
  - words, word senses or concepts, phrases, paragraphs, documents
- Similarity is a specific type of relatedness
  - **related**: topically or via relation
    *heart vs surgeon*
    *wheel vs bike*
  - **similar**: synonyms and hyponyms
    *doctor vs surgeon*
    *bike vs bicycle*

# Semantic similarity: Numerous attempts to answer that


*Image from D. Jurgens' NAACL 2016 tutorial.

# Semantic similarity: Numerous attempts to answer that

Allison and Dix (1986)
Gusfield (1997)
Wise (1996)
Keselj et al. (2003)
50+ Approaches from SemEval 2012, 2013, 2014

**Not to mention word embeddings...**

Sussna (1993, 1997)
Wu and Palmer (1994)
Resnik (1995)
Jiang and Conrath (1997)
Lin (1998)
Hirst and St-Onge (1998)
eacock and Chodorow (1998)
Patwardan (2003)
anerjee and Pederson (2003)

Gat 7)

Ramage et al. (2009)
Yeh et al. (2009)
Radinsky et al. (2011)

We refer to these as Linguistic Levels

Sentence Word Sense

# Semantic similarity: Why do we have so many methods?

- New resources or methods
  - new datasets reveal weakness in previous methods
  - state-of-the-art is moving target
- Task-specific similarity functions
- Performance in new tasks not satisfactory

➡ Semantic similarity is **not** the end-task
  - Pick the one which yields best results
  - Need for methods to quickly adapt similarity

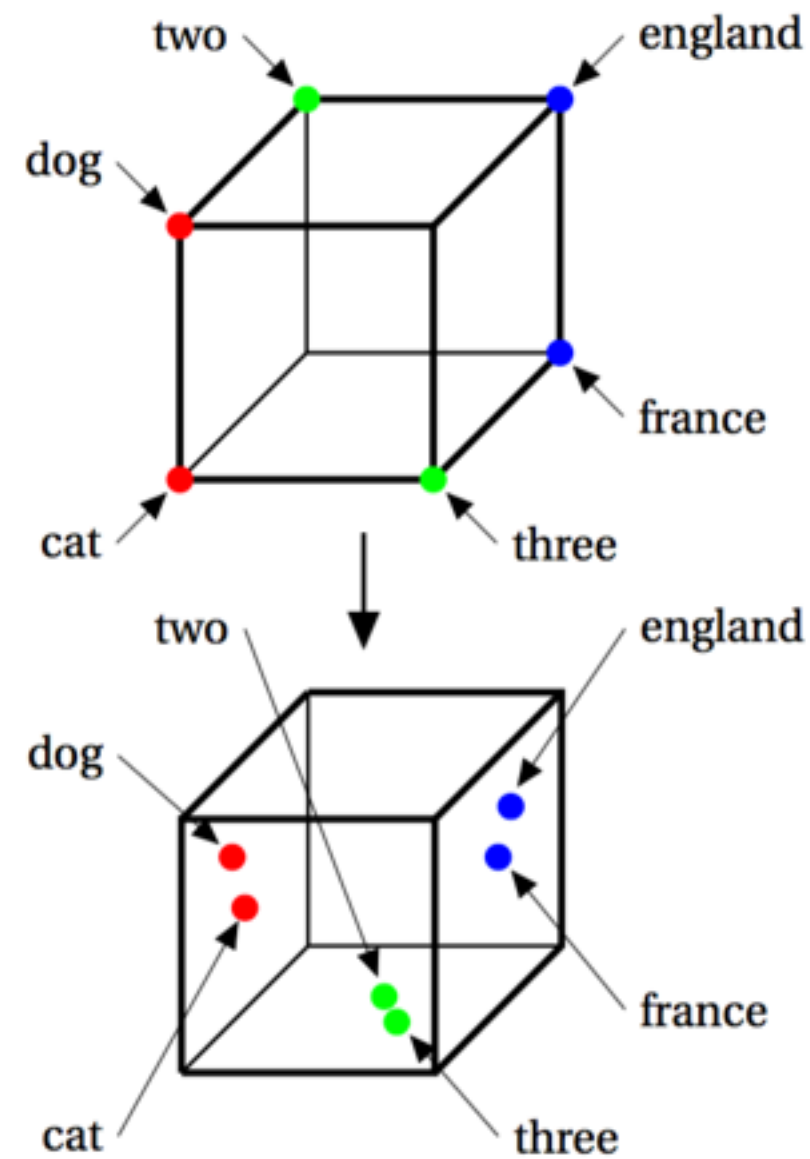
# Two main sources for measuring similarity

**Massive text corpora**

**Semantic resources and knowledge bases**

# How to represent semantics? Vector space models

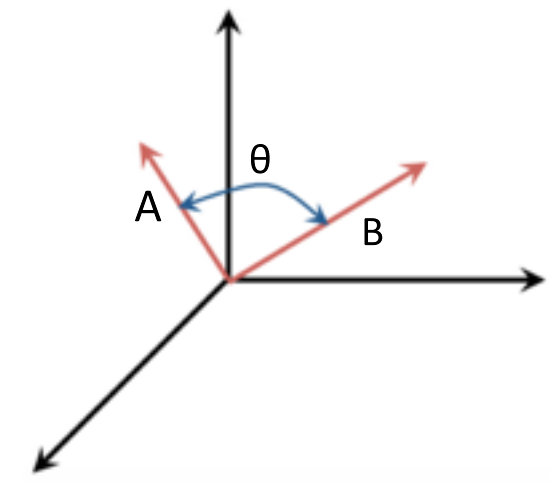
- **Explicit**: each dimension denotes specific linguistic items
  - interpretable dimensions
  - high dimensionality
- **Continuous**: dimensions are not tied to explicit concepts
  - enable comparison between represented linguistic items
  - low dimensionality



# How to compare two linguistic items in the vector space

- Cosine of the angle θ between A and B:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$




- Explicit models have a serious sparsity problem due to their discrete or “k-hot” vector representations

*france* = [0, 0, 0, 1, 0, 0]
*england* = [0, 1, 0, 0, 0, 0]
*france is near spain* = [1, 0, 0, 1, 1, 1]

  - cos(*france*, *england*) = 0.0
  - cos(*france*, *france is near spain*) = 0.57

# Learning word vector representations from text

- Limitations of knowledge-based methods
  - out-of-context despite validity of resources
  - most lack of evaluation on practical tasks
- What if we do not know anything about words? Follow the distributional hypothesis:

"You shall know a word by the company it keeps", Firth 1957

**financial institution**

The value of the central **bank** increased by 10%.

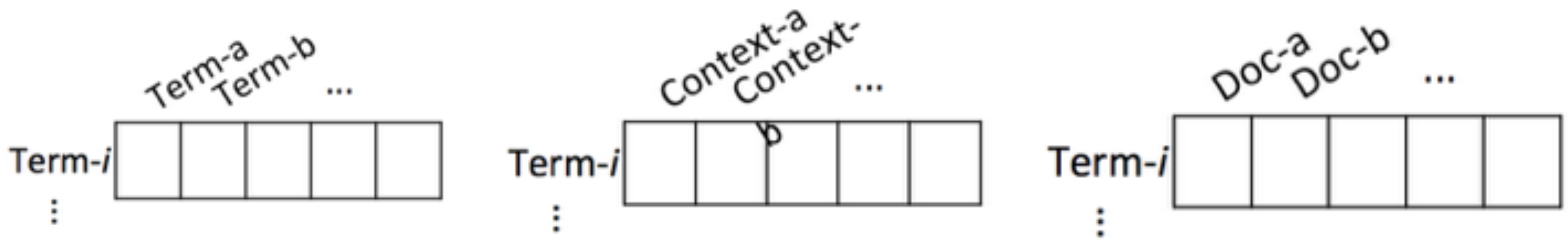
She often goes to the **bank** to withdraw cash.

She went to the river **bank** to have picnic with her child.

**geographical term**

# Simple approach: Compute a word-in-context co-occurence matrix

- Matrix of counts between words and contexts




words context document

- **Limitations** of this method:
  - all words have equal importance (imbalance)
  - vectors are very high dimensional (storage issue)
  - infrequent words have overly sparse vectors (make subsequent models less robust)

# The most standard approach: Dimensionality Reduction

- Perform singular value decomposition (SVD) of the word co-occurence matrix that we saw previously
    - typically, U*Σ is used as the vector space

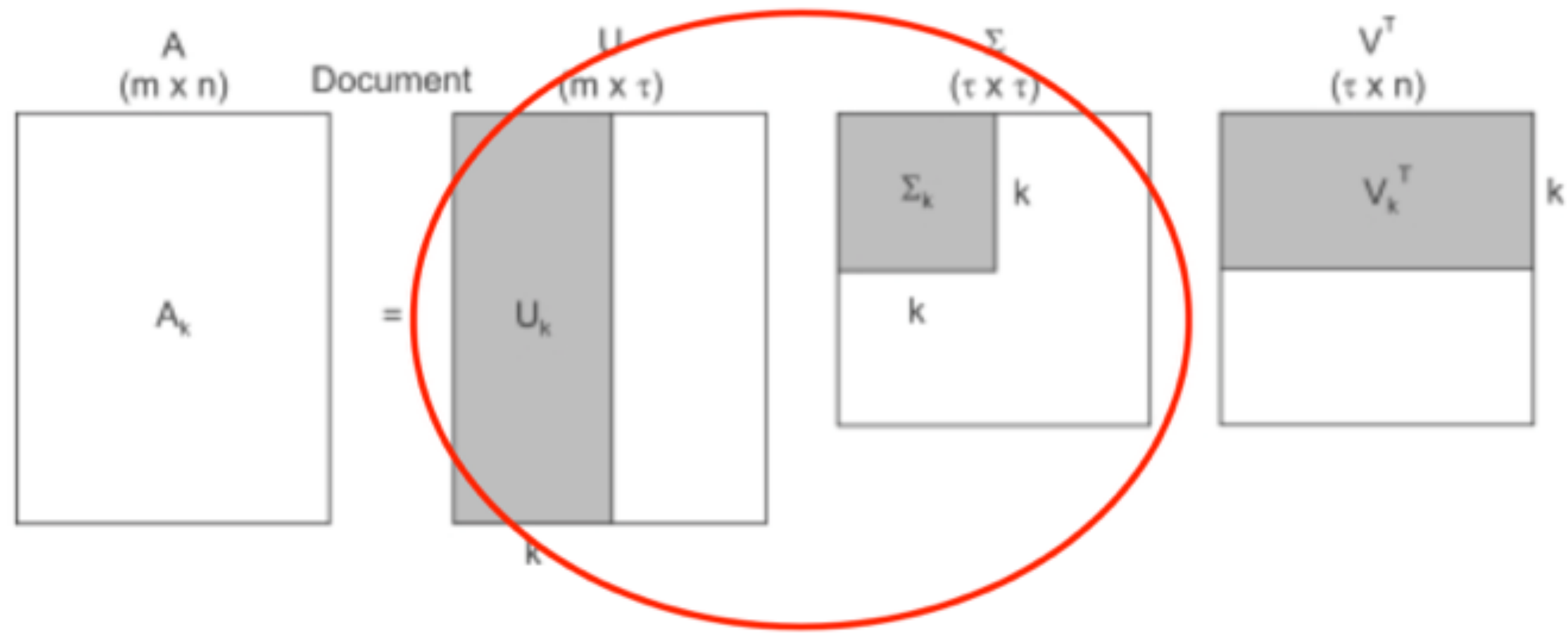
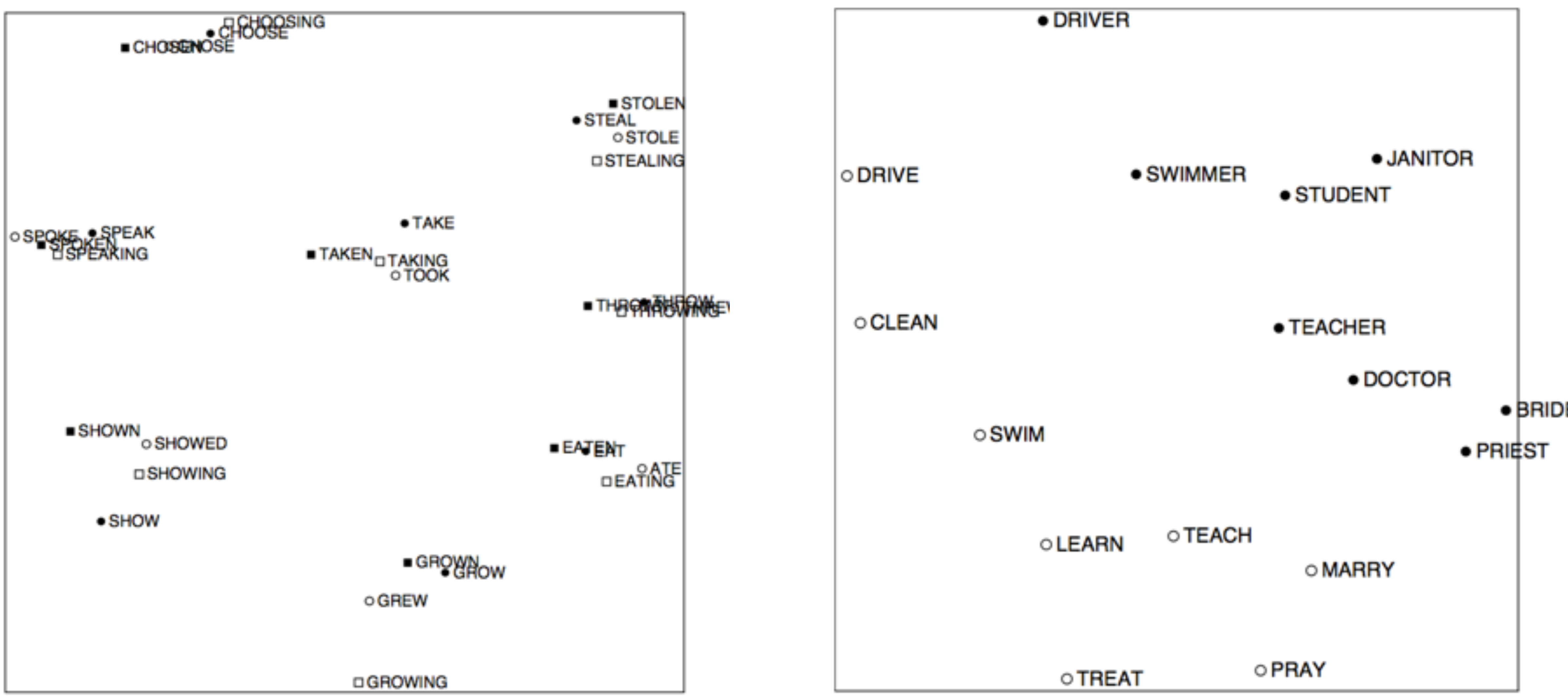



*Image from D. Jurgens' NAACL 2016 tutorial.

# The most standard approach: Dimensionality Reduction

- Syntactically and semantically related words cluster together




*Plots from Rohde et al. 2005

# Dimensionality reduction with Hellinger PCA

- Perform PCA with Hellinger distance on the word co-occurence matrix: Lebret and Collobert 2014
  - Well suited for discrete probability distributions (P, Q)

$$H(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2},$$

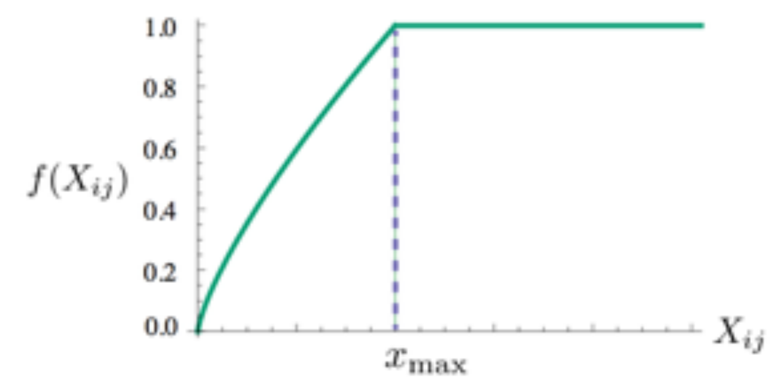
- Neural approaches are time-consuming (tuning, data)
  - instead compute word vectors efficiently with PCA
  - fine-tuning them on specific tasks! Better than neural
- **Limitations**: hard to add new words, not scalable $O(mn^2)$

https://github.com/rlebret/hpca

# Dimensionality reduction with weighted least squares

- Glove vectors by Pennington et al 2014. Factorizes the log of the co-occurence matrix:

$$J(\theta) = \frac{1}{2}\sum_{i,j=1}^{W} f(P_{ij})(u_i^T v_j - \log P_{ij})$$




- Fast training, scalable to huge corpora but still hard to incorporate new words
- Much better results than neural embedding, however under equivalent tuning it is not the case: Levy and Goldberg 2015

http://nlp.stanford.edu/projects/glove/

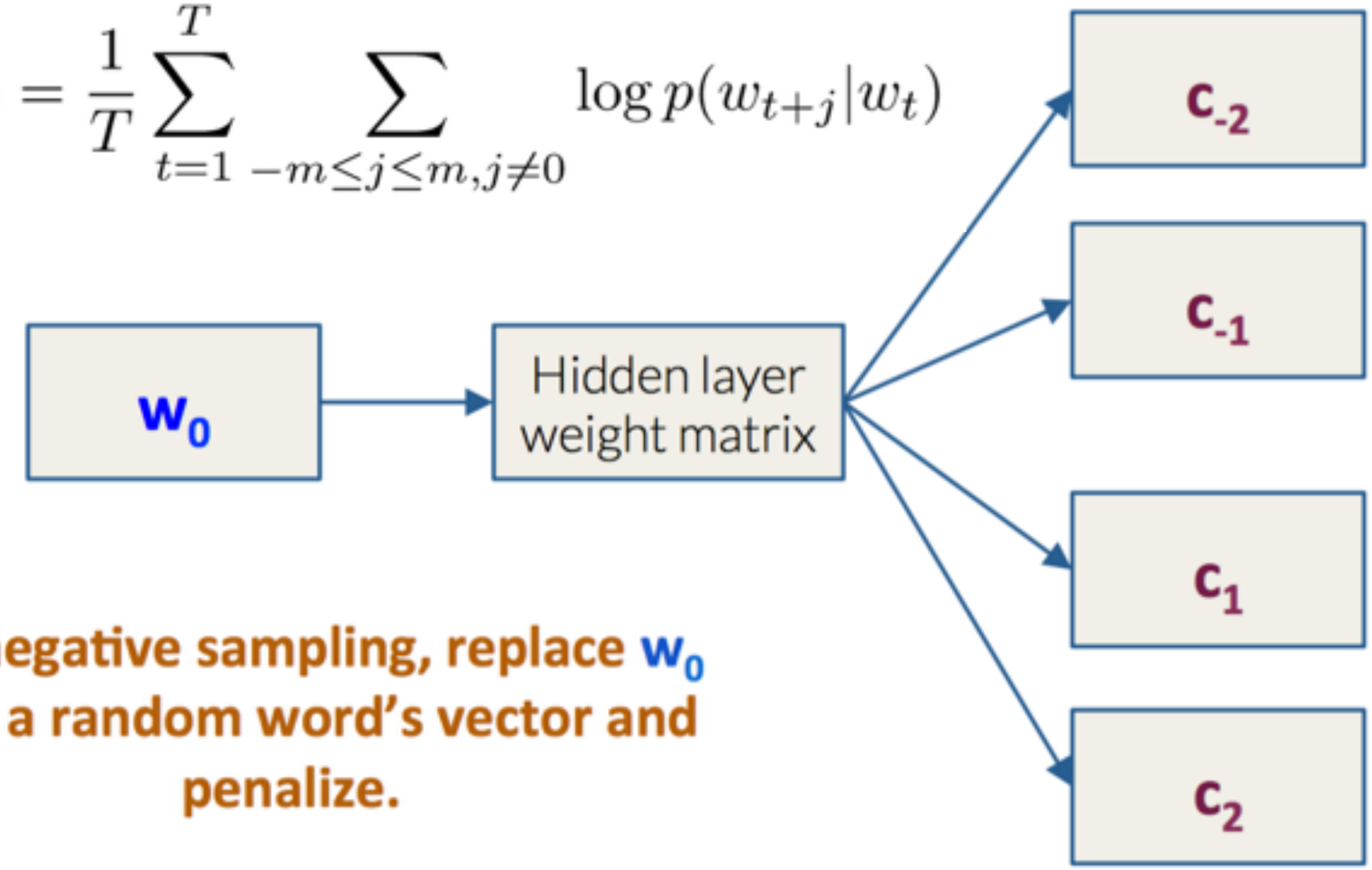
# Dimensionality reduction with neural networks

- The main idea is to directly learn low-dimensional word representations from data
  - Learning representations: Rumelhart et al 1986
  - Neural probabilistic language model: Bengio et al 2003
  - NLP (almost) from scratch: Collobert and Weston 2008
- Recent methods are faster and more simple
  - Continuous Bag-Of-Words (CBOW)
  - Skip-gram with Negative Sampling (SGNS)
  - word2vec toolkit: Mikolov et al. 2013

# word2vec: Skip-gram with negative sampling (SGNS)

- Given the middle word predict surrounding ones in a fixed window of words (maximize log likelihood)

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-m \leq j \leq m, j \neq 0} \log p(w_{t+j} | w_t)$$




**For negative sampling, replace $w_0$ with a random word's vector and penalize.**

# word2vec: Skip-gram with negative sampling (SGNS)

- How is the $P(w_t|h)$ probability implemented?

$$P(w_t|h) = \text{softmax}(\text{score}(w_t, h)) = \frac{\exp\{\text{score}(w_t, h)\}}{\sum_{\text{Word w' in Vocab}} \exp\{\text{score}(w', h)\}}.$$

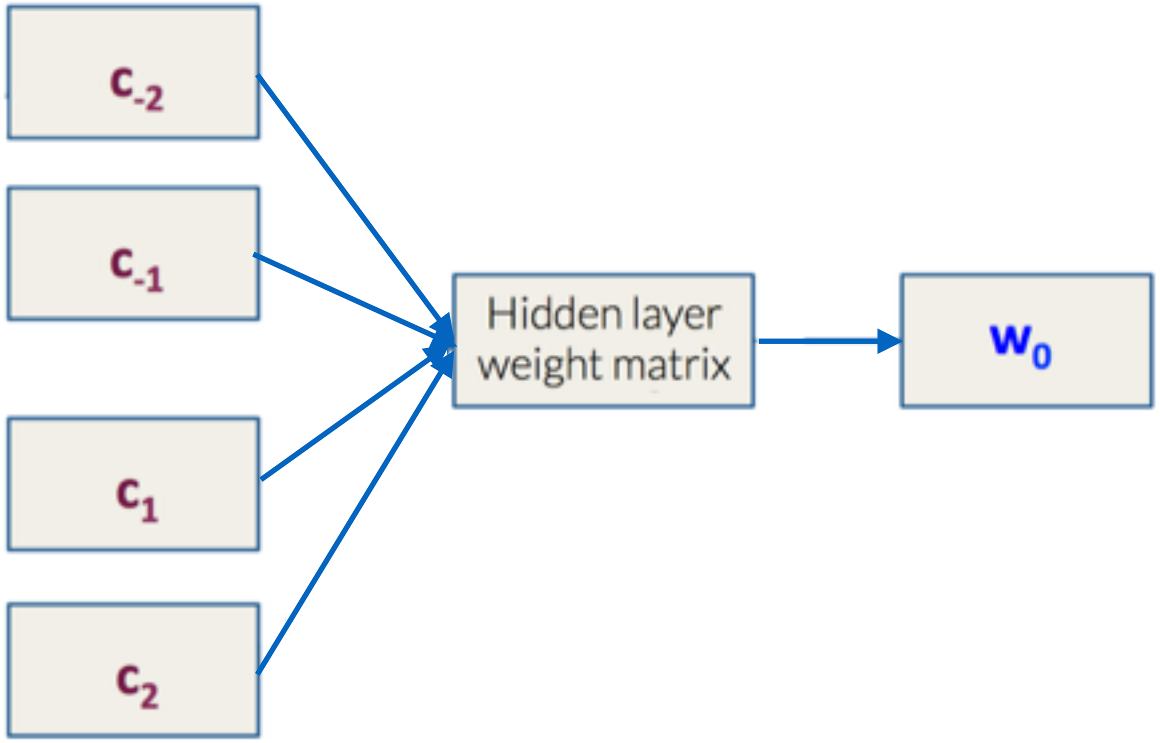
- Denominator is very inefficient for big vocabulary!
- Instead it uses a more scalable objective, $\log Q_\theta$ is a binary logistic regression of word w and history h:

$$J_{\text{NEG}} = \log Q_\theta(D = 1|w_t, h) + k \mathop{\mathbb{E}}_{\tilde{w} \sim P_{\text{noise}}} [\log Q_\theta(D = 0|\tilde{w}, h)]$$

# word2vec: Continuous Bag-Of-Words with negative sampling (CBOW)

- More efficient but the ordering information of the words does not influence the projection

- Factorizes a PMI word-context matrix: Levy and Goldberg 2014
    - builds upon existing methods (new decomp.)
    - improvements on a variety of intrinsic tasks such as <u>relatedness</u>, <u>categorization</u> and <u>analogy</u>: Baroni et al 2014, Schnabel et al 2015

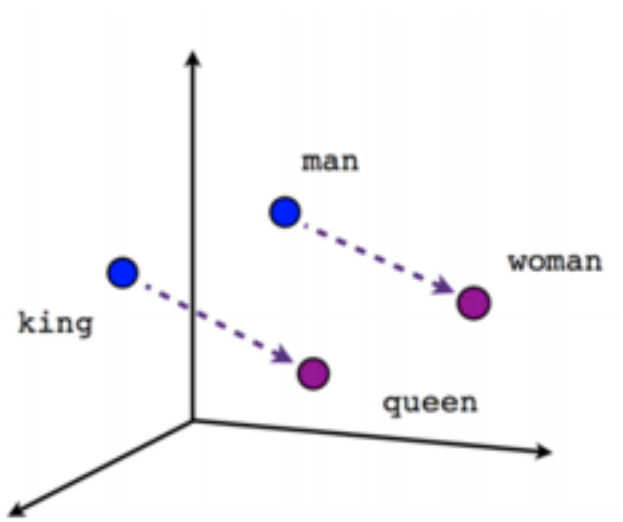



| | RG | WordSim | MEN | TOEFL |
|---|---|---|---|---|
| PMI+SVD | .70 | .70 | .72 | .76 |
| word2vec | .83 | .78 | .80 | .86 |

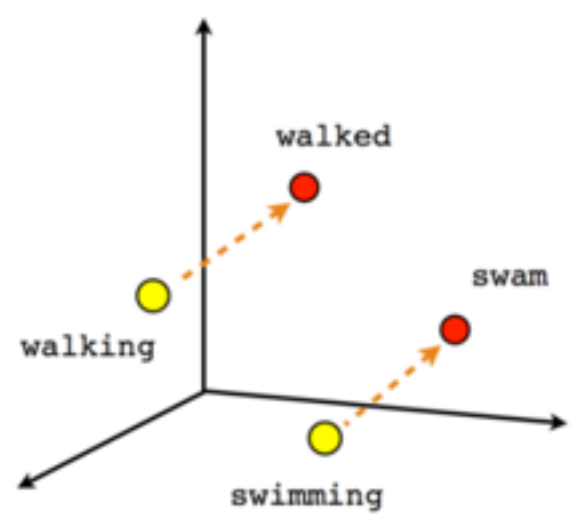
# word2vec: Learns meaningful linear relationships of words

- Word vector dimensions capture several meaningful relations between words: present—past tense, singular—plural, male—female, capital—country
- **Analogy** between words can be efficiently computed using basic arithmetic operations between vectors (+, -)

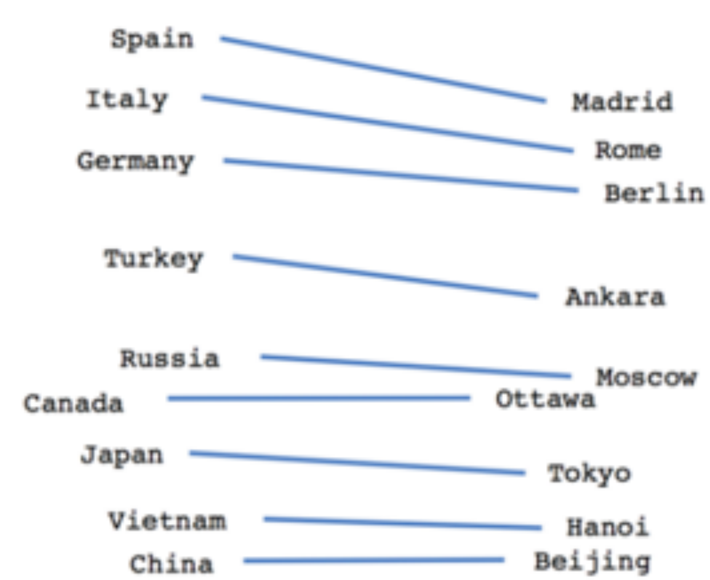
*king - man + woman ≈ queen*




Male-Female




Verb tense




Country-Capital

# Learning word representations from text: Recap

- Most methods are ∗similar∗ to SVD over PMI matrix however word2vec has the edge over alternatives
  - scales well on massive text corpora and new words
  - yields top results in most tasks
- On extrinsic tasks it is essential to fine-tune (for beating BOW)

➡ **Several extensions**
  - dependency-based embeddings: Levy and Goldberg 2014
  - retrofitted-to-lexicons embeddings: Faruqui et al. 2014
  - sense-aware embeddings: Li and Jurafsky 2015
  - visually-grounded embeddings: Lazaridou et al. 2015
  - multilingual embeddings: Gouws et al 2015

# Open problems in semantic similarity research

- Irregular language

can i watch 4od bbc iplayer etc with 10GB useage allowence?

- Multi-word expressions

We need to sort out the problem

We need to sort the problem out

- Syntax and punctuations

Man bites dog | Dog bites man

A woman: without her, man is nothing.

# Open problems in semantic similarity research

- Variable-size input

Prius

A fuel-efficient hybrid car

An automobile powered by both an internal combustion (...)

- Ambiguity when lacking context

The boss **fired** his worker.

- Subjectivity versus objectivity

This was a **good** day. | This was a **bad** day.

- Out-of-vocabulary words: slang, hash-tags, neologisms

# Beyond words

- Word vectors are also useful for building semantic vectors of phrases, sentences and documents
  - input or output space for several practical tasks
  - basis for multilingual or multimodal transfer (via alignment)
  - **<u>interpretability</u>**: do we care about what each word vector dimension means? It depends. We may need to compromise.

- Next course:
  - learning representations of word sequences
  - more details on sequence models

# References


- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." In NIPS 2013.
- Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." In EMNLP, 2014.
- Remi Lebret, Ronan Collobert. "Word Embeddings through Hellinger PCA." In EACL, 2014
- Quoc V. Le, and Tomas Mikolov. "Distributed Representations of Sentences and Documents." In ICML, 2014.
- Manaal Faruqui, Jesse Dodge, Sujay K. Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith. "Retrofitting word vectors to semantic lexicons.", In ACL 2014.
- Omer Levy and Yoav Goldberg. "Dependency-Based Word Embeddings." In ACL 2014.
- Tobias Schnabel, Igor Labutov, David Mimno, and Thorsten Joachims. "Evaluation methods for unsupervised word embeddings." In EMNLP, 2015.
- Omer Levy, Yoav Goldberg, and Ido Dagan. "Improving distributional similarity with lessons learned from word embeddings." TACL, 2015.
- Manaal Faruqui, Yulia Tsvetkov, Pushpendre Rastogi, and Chris Dyer. "Problems With Evaluation of Word Embeddings Using Word Similarity Tasks." In RepEval 2016.
- Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah Smith. "Sparse overcomplete word vector representations." ACL 2015.
- Yoav Goldberg. "A primer on neural network models for natural language processing" arXiv preprint: 1510.00726, 2015.
- Ian Goodfellow, Aaron Courville, and Joshua Bengio. "Deep learning". Book in preparation for MIT Press., 2015.

# Resources (1/2)

➡ **Online courses**

- Coursera course on "Neural networks for machine learning" by Geoffrey Hinton
  https://www.coursera.org/learn/neural-networks
- Coursera course on "Machine learning"  by Andrew Ng
  https://www.coursera.org/learn/machine-learning
- Stanford CS224d "Deep learning for NLP" by Richard Socher
  http://cs224d.stanford.edu/

➡ **Conference tutorials**

- Richard Socher and Christopher Manning, "Deep learning for NLP", EMNLP 2013 tutorial.
  http://nlp.stanford.edu/courses/NAACL2013/
- David Jurgens and Mohammad Taher Pilehvar, "Semantic Similarity Frontiers: From Concepts to Documents", EMNLP 2015 tutorial.
  http://www.emnlp2015.org/tutorials.html#t1
- Mitesh M Kharpa, Sarath Chandar, "Multilingual and Multimodal Language Processing", NAACL 2016 tutorial.
  http://naacl.org/naacl-hlt-2016/t2.html

# Resources (2/2)

➡ **Deep learning toolkits**

- Theano http://deeplearning.net/software/theano
- Torch http://www.torch.ch/
- Tensorflow http://www.tensorflow.org/
- Keras http://keras.io/

➡ **Pre-trained word vectors and codes**

- Word2vec toolkit and vectors
  https://code.google.com/p/word2vec/
- GloVe code and vectors
  http://nlp.stanford.edu/projects/glove/
- Hellinger PCA
  https://github.com/rlebret/hpca
- Online word vector evaluation
  http://wordvectors.org/