Social Media & Text Analysis

lecture 10 - Deep Learning for NLP

Follow @cocoweixu

CSE 5539-0010 Ohio State University

Instructor: Wei Xu

Website: socialmedia-class.org

some slides are adapted from Richard Socher, Greg Durrett, Chris Manning, Chris Dyer, Dan Jurafsky,

Last Class

- December 7 (Review and Q&A)
- No final exam
- Also one-on-one office hour by appointment on December 7 for homework #3 or research projects or Q&A

Word Embeddings

- Skip-gram — predicts surrounding "outside" words given the "center" word

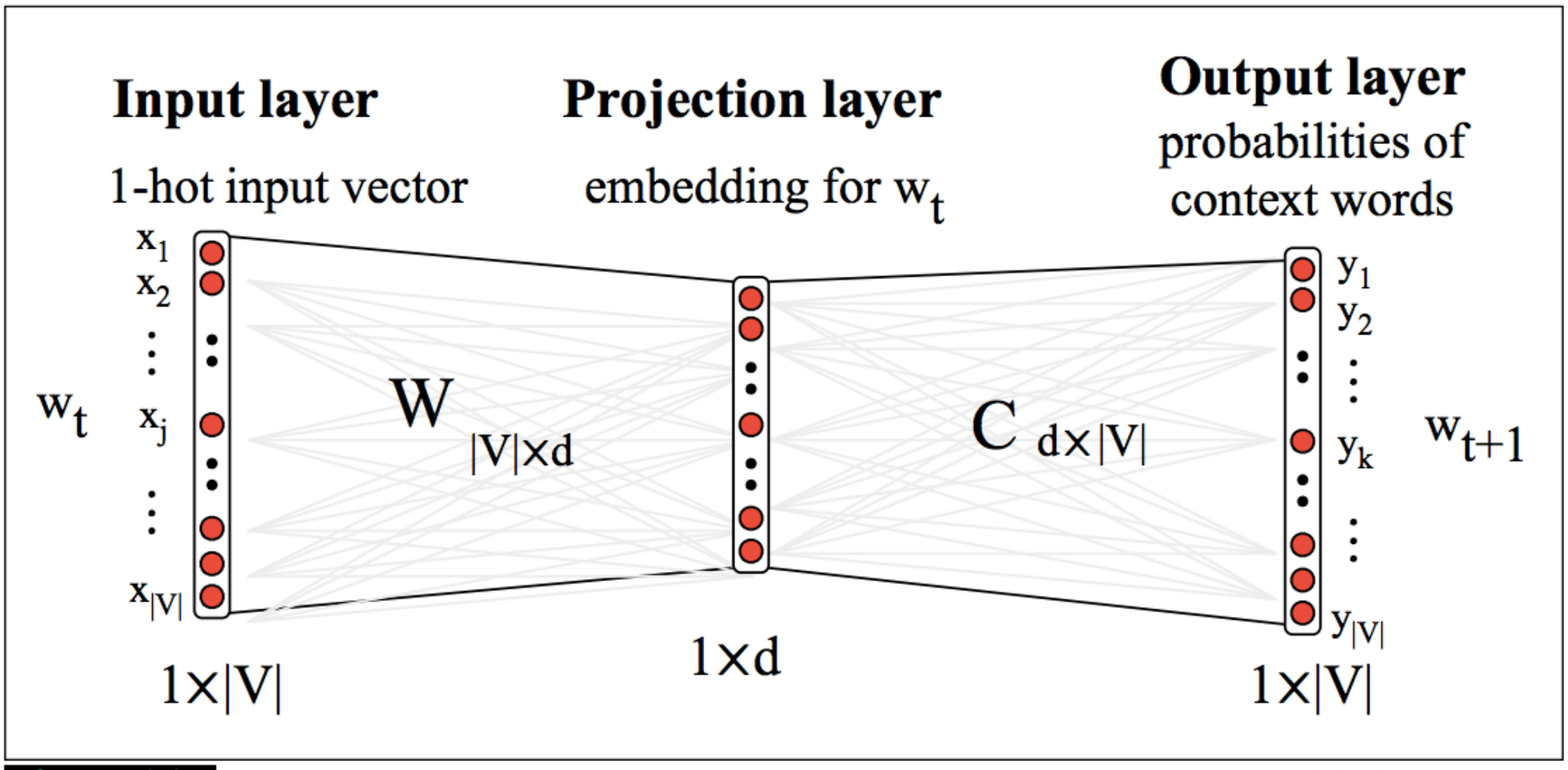


Figure 16.5 The skip-gram model viewed as a network (Mikolov et al. 2013, Mikolov et al. 2013a).

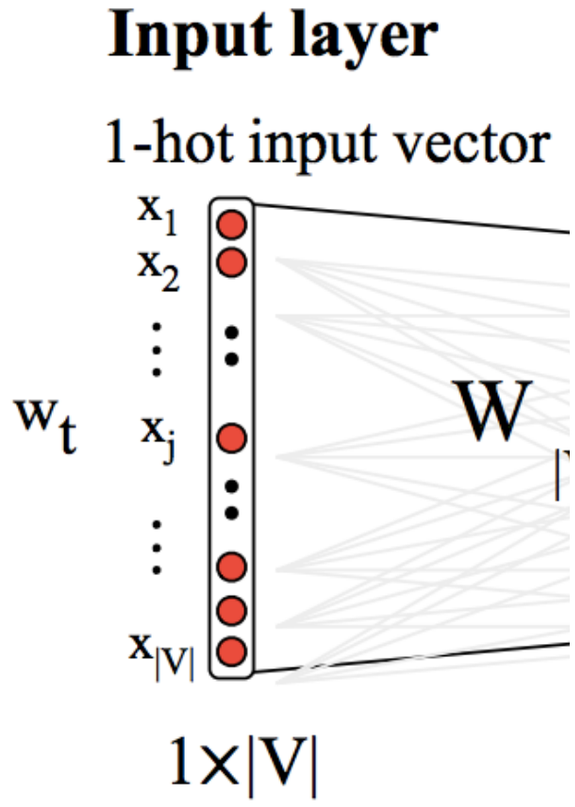
Input Layer

- “one-hot” word vectors

  - a vector of dimension $|\boldsymbol{V}|$ (size of vocabulary)

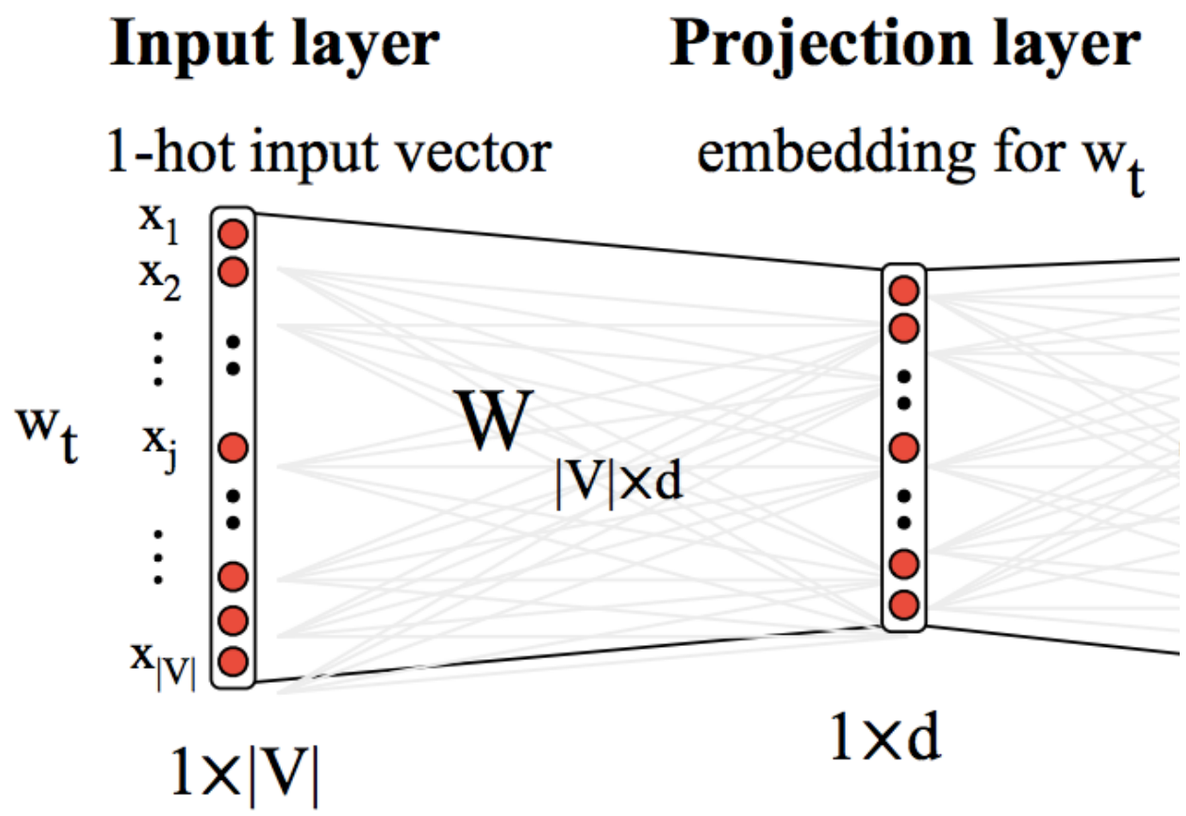
  - all “0”s expect a single “1” in the vector

  - different positions of that “1” represent different words



Hidden (Projection) Layer

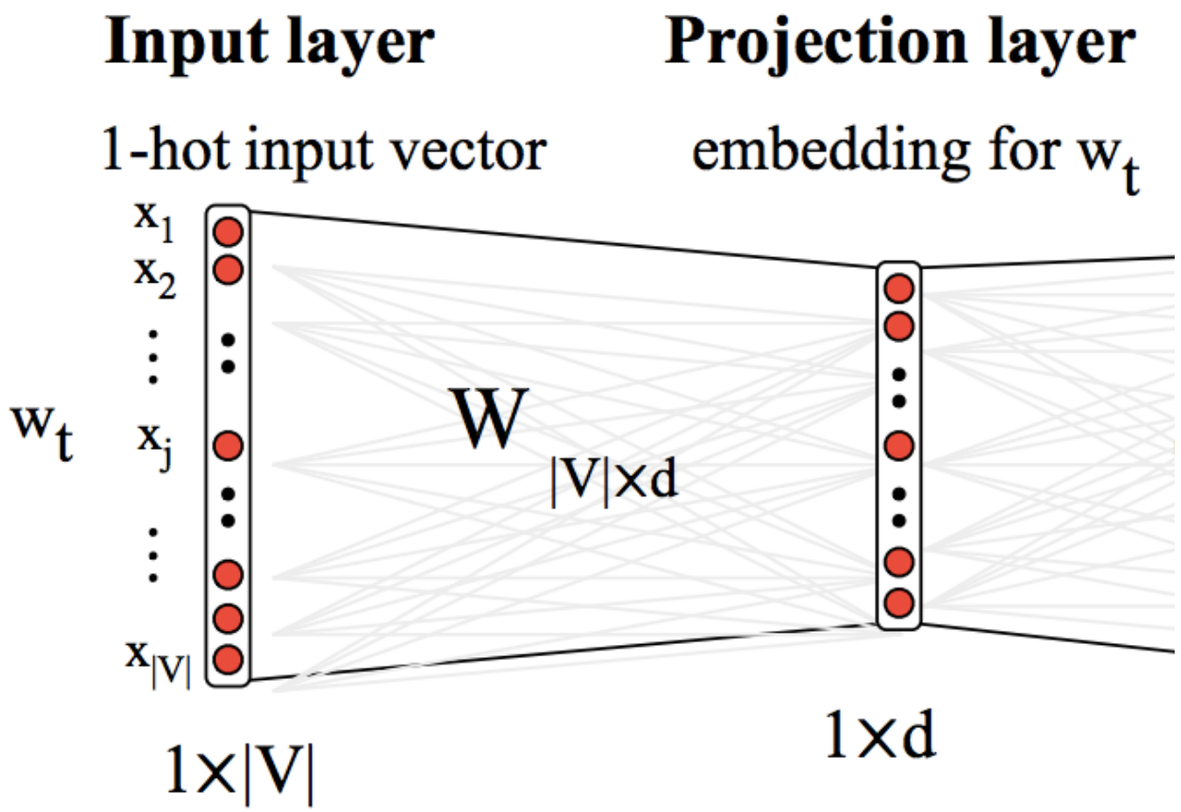
- A simple look up — the rows of this weight matrix are actually “input” word vectors



Hidden (Projection) Layer

- A simple look up — the rows of this weight matrix are actually "input" word vectors

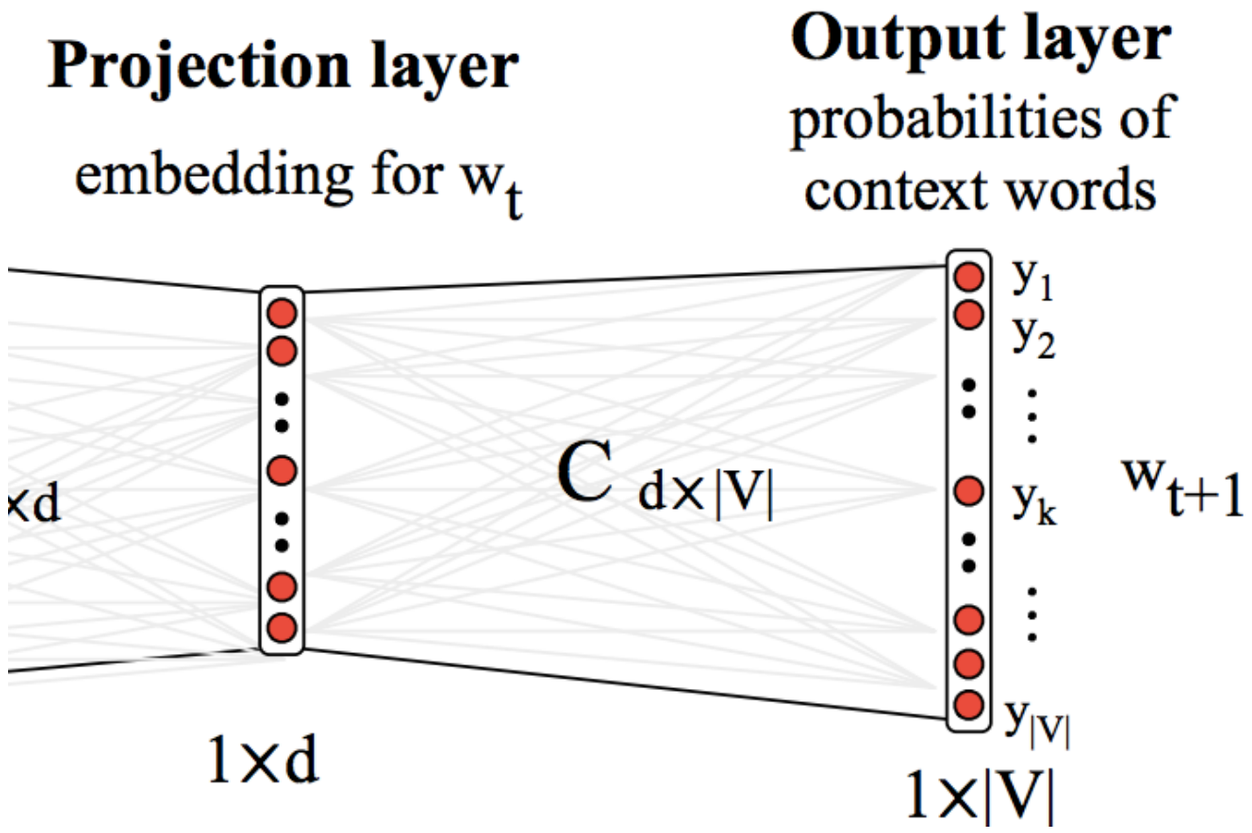
$$[0 \quad 0 \quad 0 \quad 1 \quad 0] \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = [10 \quad 12 \quad 19]$$



Output Layer

- predicts surrounding “outside” (context) words given the “center” word → A classification problem!

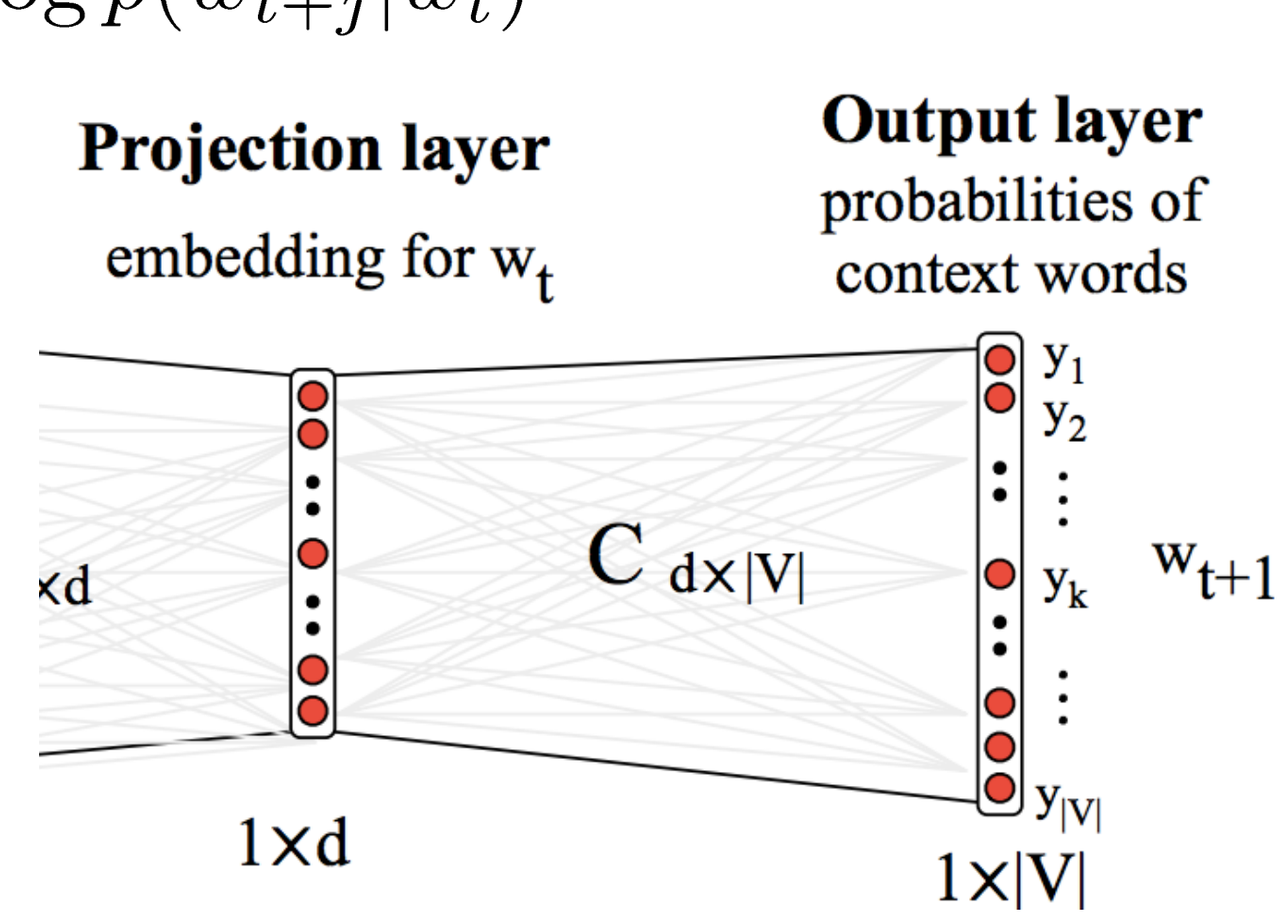
- Softmax Regression = Multi-class Logistic Regression



Output Layer

- Objective function: maximize the log probability of any "outside" (context) word given the "center" word

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

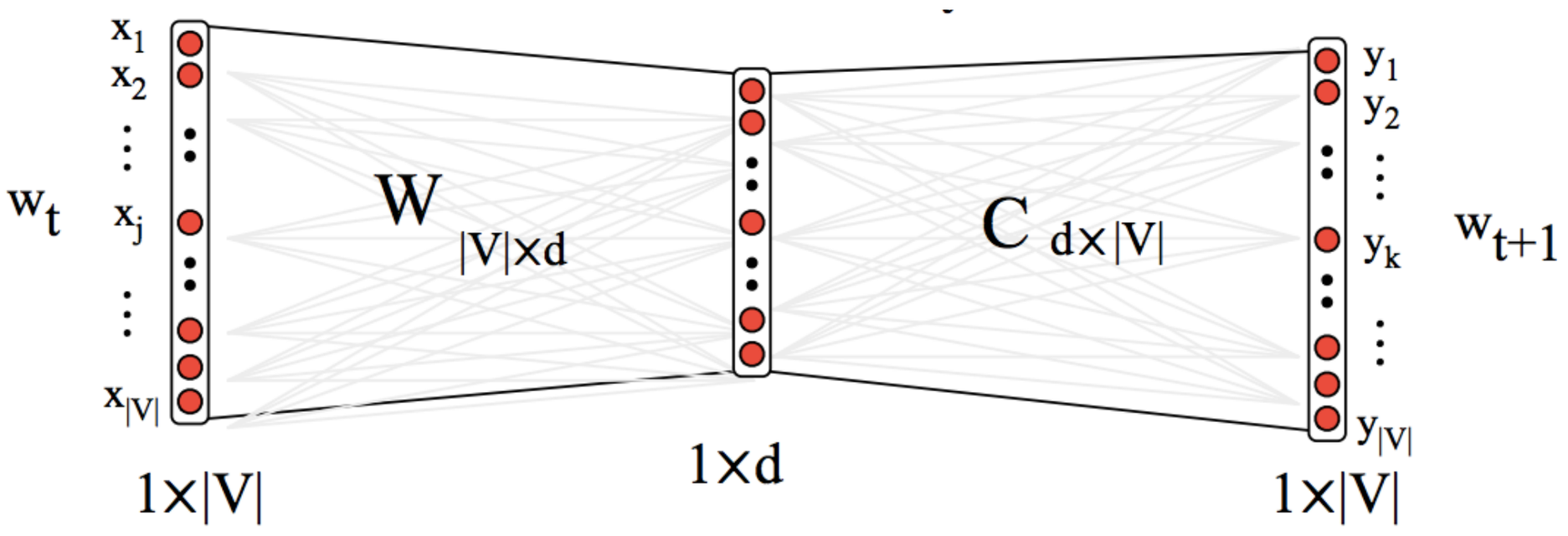


Output Layer

- predicts surrounding "outside" (context) words given the "center" word

$$p(o|c) = \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}$$

- so, every word has two vectors!



Gradient Descent

- Cost/Objective function:

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

- For a “center” word and an “outside” word:

$$\log p(o|c) = \log \frac{\exp\left(u_o^T v_c\right)}{\sum_{w=1}^{W} \exp\left(u_w^T v_c\right)}$$

Gradient Descent

- Basics:

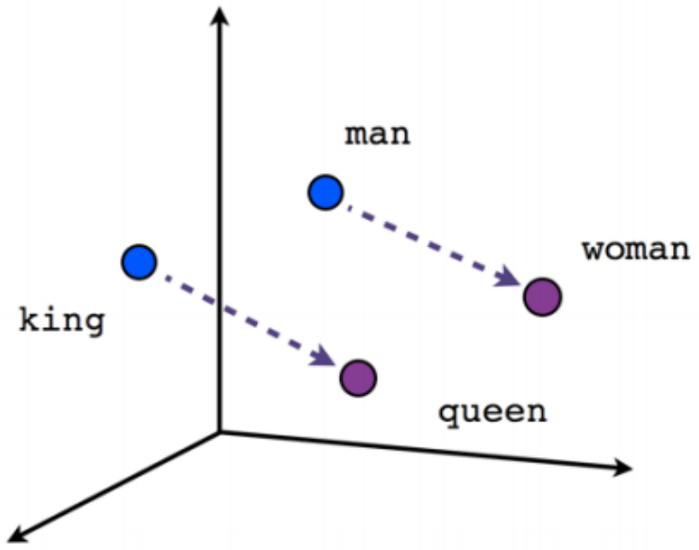
$$\frac{\partial \mathrm{x}^T a}{\partial \mathrm{x}} = \frac{\partial a^T \mathrm{x}}{\partial \mathrm{x}} = a$$

$$\frac{\partial e^{\mathrm{x}}}{\partial \mathrm{x}} = e^{\mathrm{x}} \qquad \frac{\partial \log \mathrm{x}}{\partial \mathrm{x}} = \frac{1}{\mathrm{x}}$$

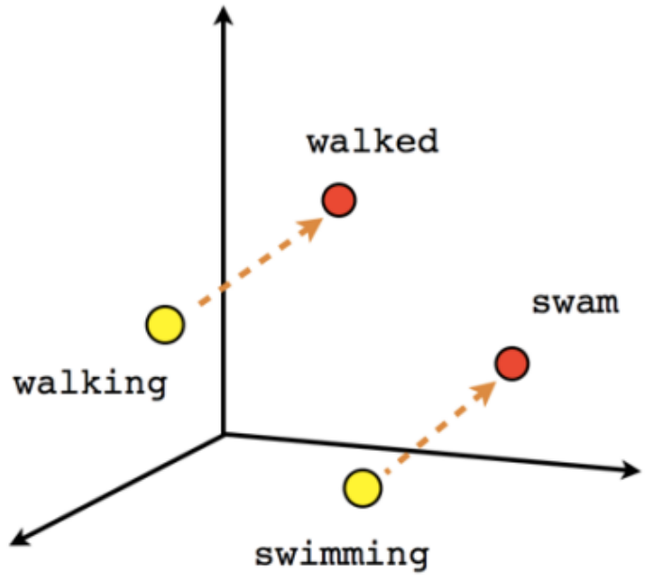
- Chain Rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial x} = \frac{\partial f(g)}{\partial g}\frac{\partial g(x)}{\partial x}$$

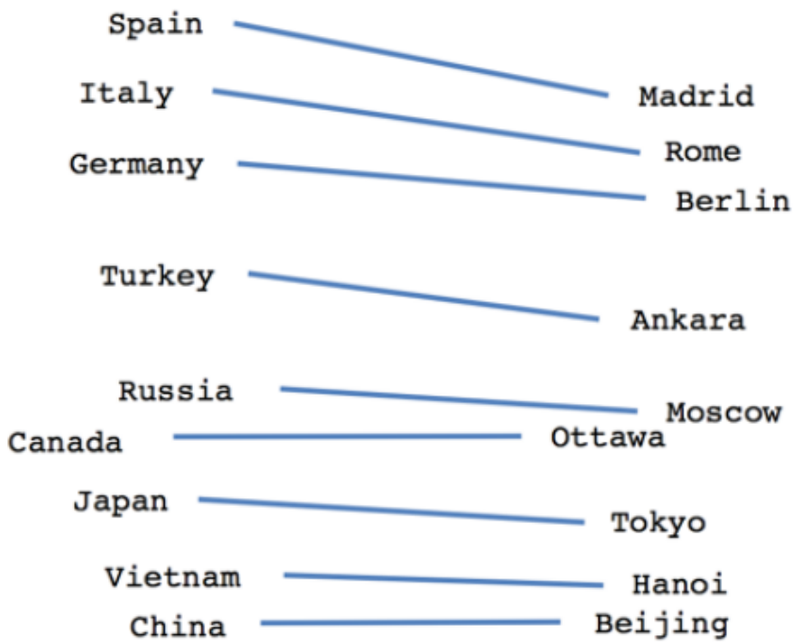
Visualization



Male-Female



Verb tense



Country-Capital

Source: tensorflow.org

Visualization

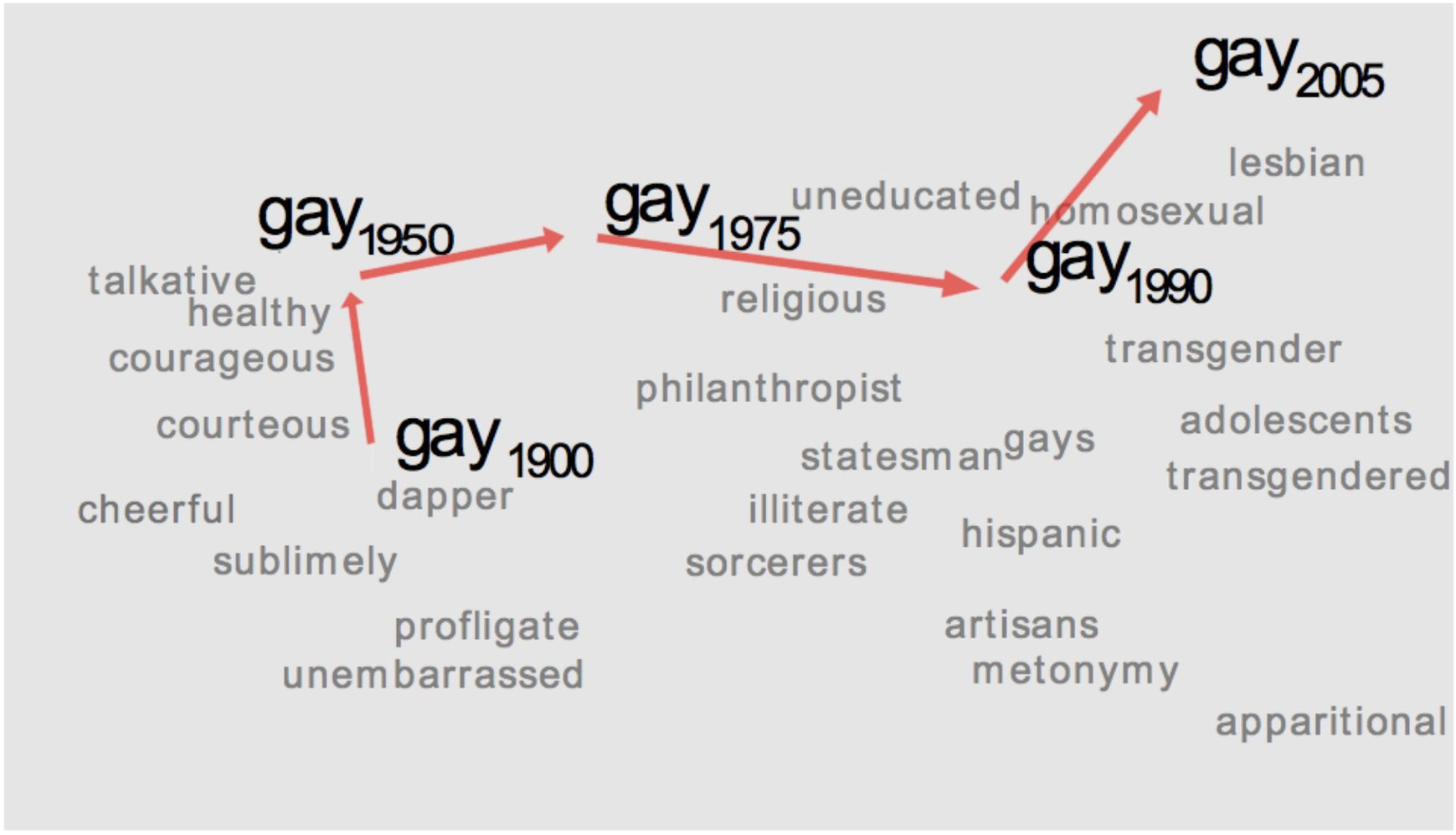


Figure 1: A 2-dimensional projection of the latent semantic space captured by our algorithm. Notice the semantic trajectory of the word **gay** transitioning meaning in the space.

Source: Kulkarni et al. (WWW 2015)
Statistically Significant Detection of Linguistic Change

DeepNLP

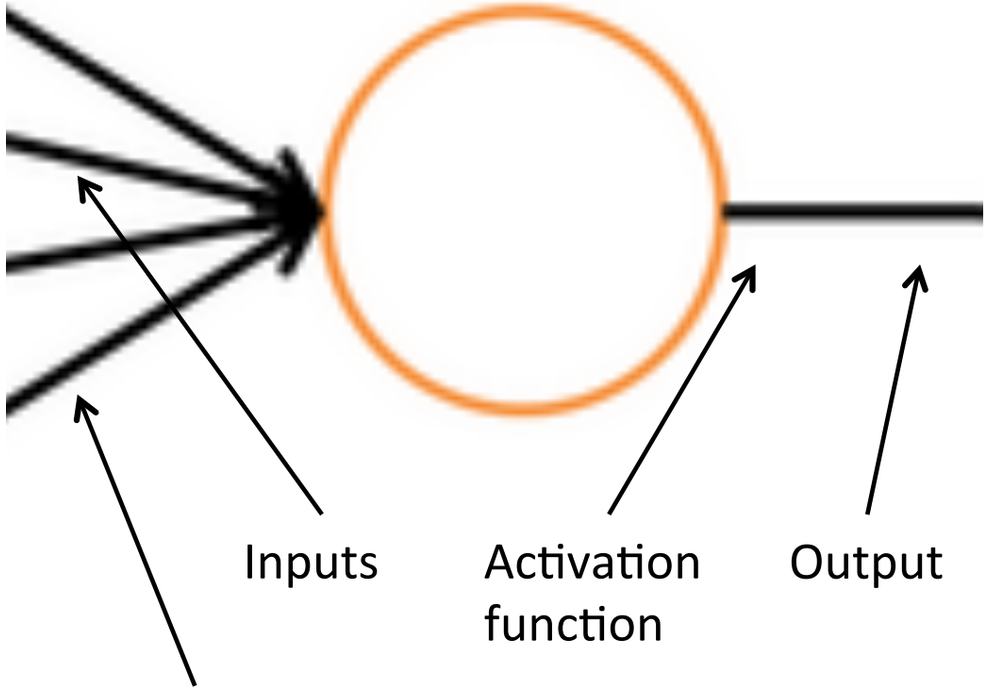
- Word vector and Recurrent Neural Networks (RNN) are the two most important concepts for deepNLP

- also, Convolutional Neural Networks (CNN), Long Short-Term Memory Networks (LSTM) …

A Neuron

- If you know Logistic Regression, then you already understand a basic neural network neuron!

A single neuron
A computational unit with *n* (3) inputs
and 1 output
and parameters *W, b*



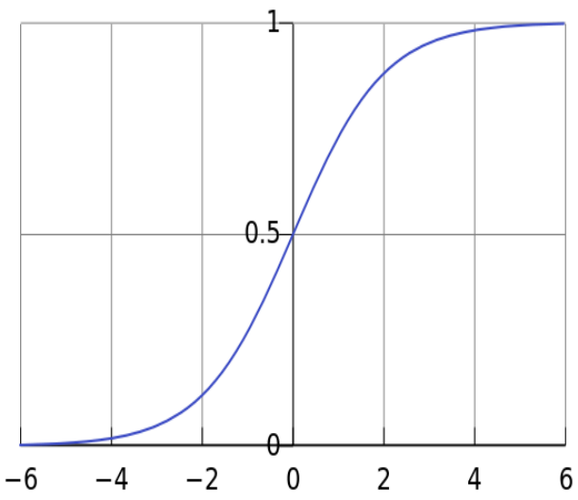
A Neuron

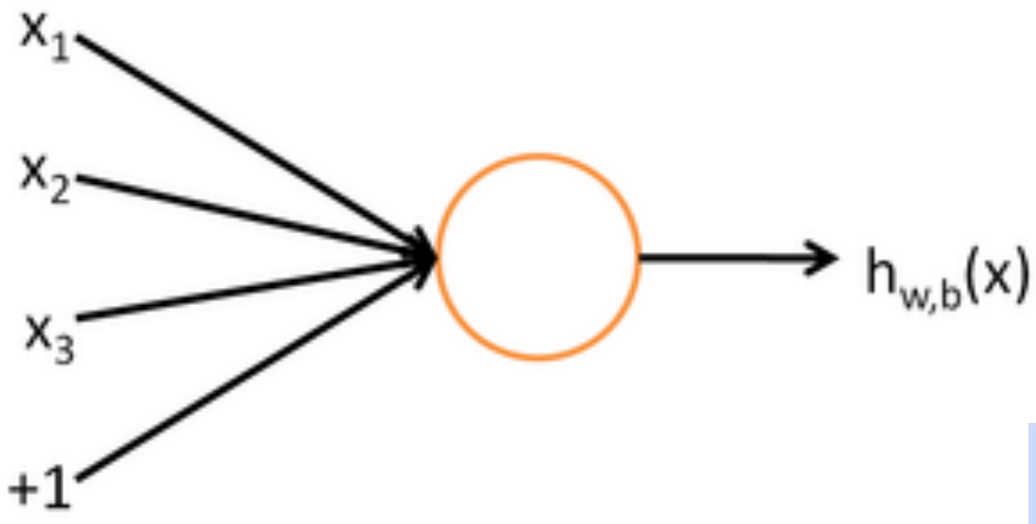
is essentially a binary logistic regression unit

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

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

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



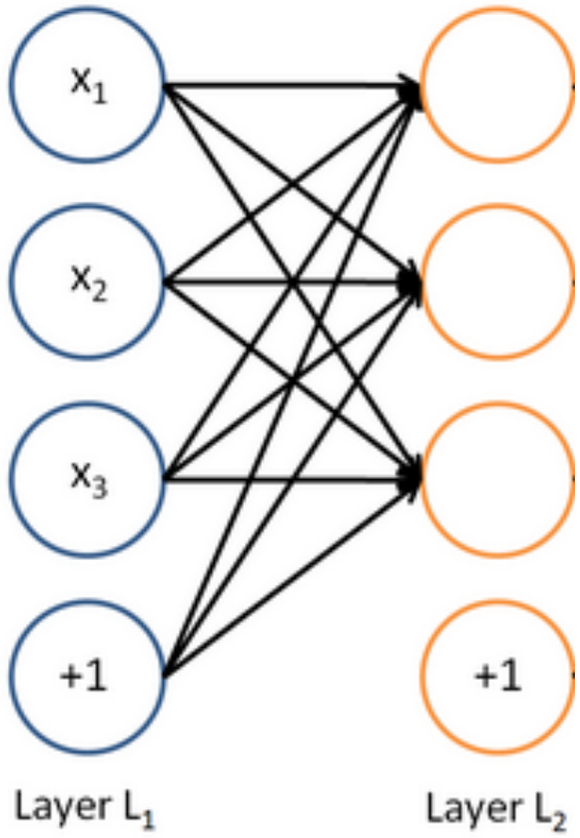


w, *b* are the parameters of this neuron
i.e., this logistic regression model

A Neural Network

= running several logistic regressions at the same time

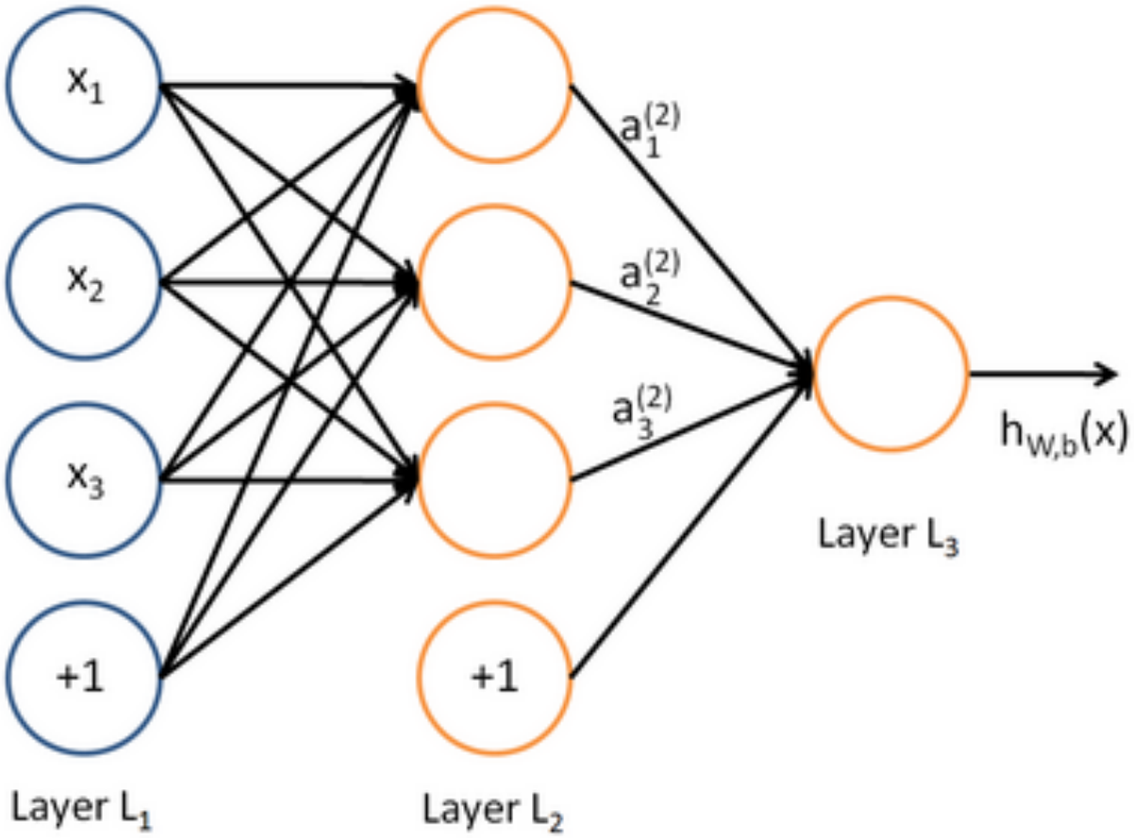
If we feed a vector of inputs through a bunch of logistic regression functions, then we get a vector of outputs …



A Neural Network

= running several logistic regressions at the same time

… which we can feed into another logistic regression function

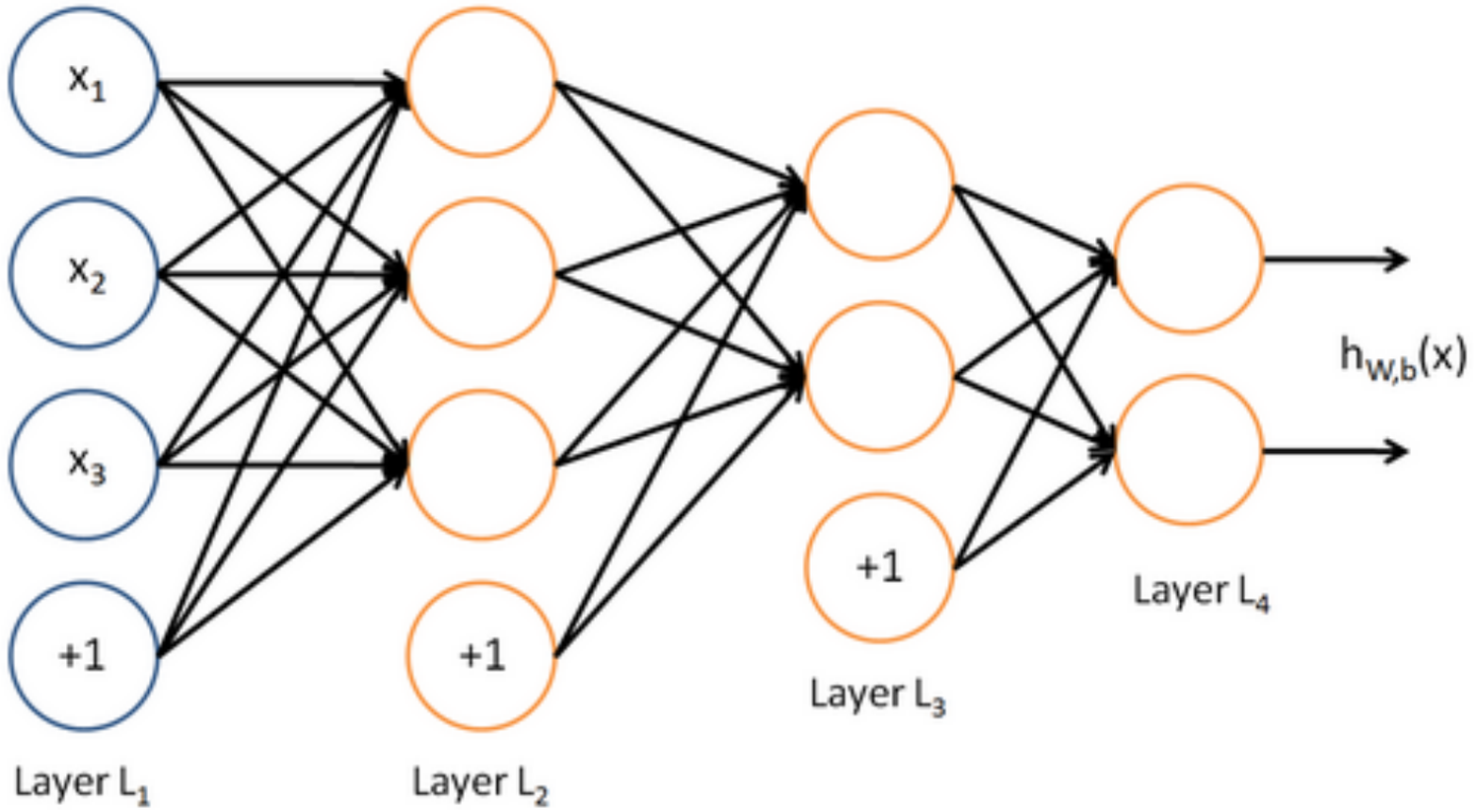


It is the loss function that will direct what the intermediate hidden variables should be, so as to do a good job at predicting the targets for the next layer, etc.

A Neural Network

= running several logistic regressions at the same time

Before we know it, we have a multilayer neural network….



f: Activation Function

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.

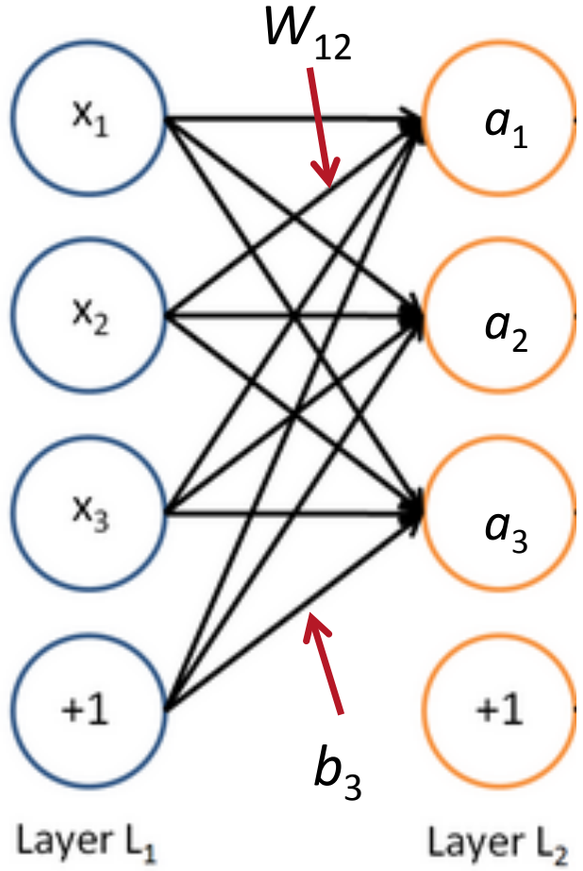
In matrix notation

$$z = Wx + b$$

$$a = f(z)$$

where f is applied element-wise:

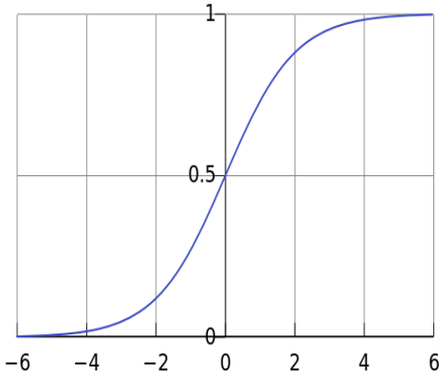
$$f([z_1, z_2, z_3]) = [f(z_1), f(z_2), f(z_3)]$$



Activation Function

logistic ("sigmoid")

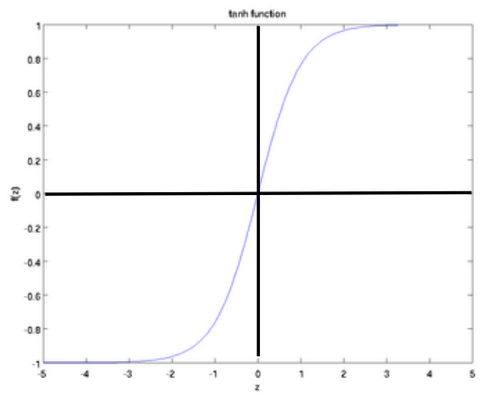
$$f(z) = \frac{1}{1 + \exp(-z)}.$$



$$f'(z) = f(z)(1 - f(z))$$

tanh

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



$$f'(z) = 1 - f(z)^2$$

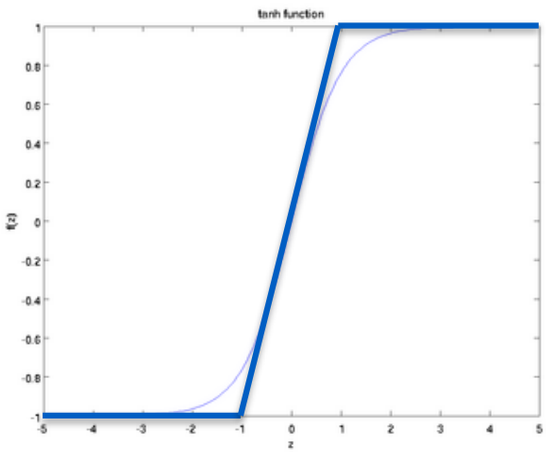
tanh is just a rescaled and shifted sigmoid

$$\tanh(z) = 2\text{logistic}(2z) - 1$$

Activation Function

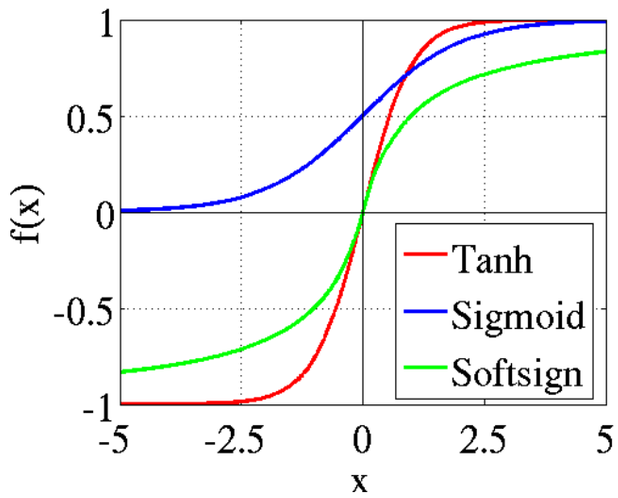
hard tanh

$$\text{HardTanh}(x) = \begin{cases} -1 & \text{if } x < -1 \\ x & \text{if } -1 <= x <= 1 \\ 1 & \text{if } x > 1 \end{cases}$$



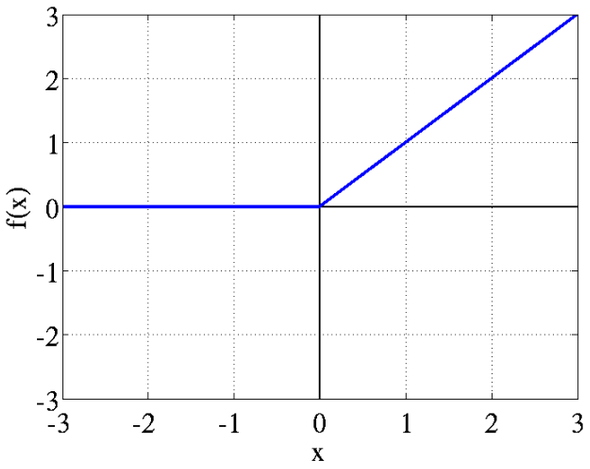
soft sign

$$\text{softsign}(z) = \frac{a}{1+|a|}$$



rectified linear (ReLu)

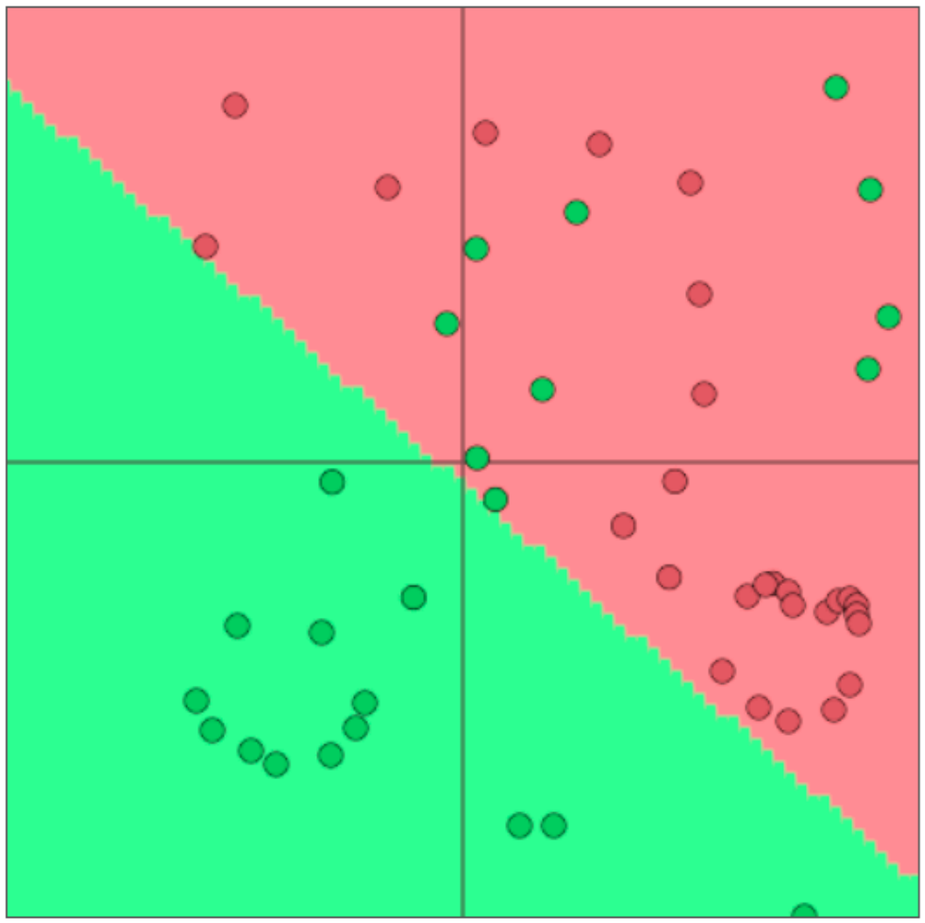
$$\text{rect}(z) = \max(z, 0)$$



- hard tanh similar but computationally cheaper than tanh and saturates hard.
- Glorot and Bengio, *AISTATS* 2011 discuss softsign and rectifier

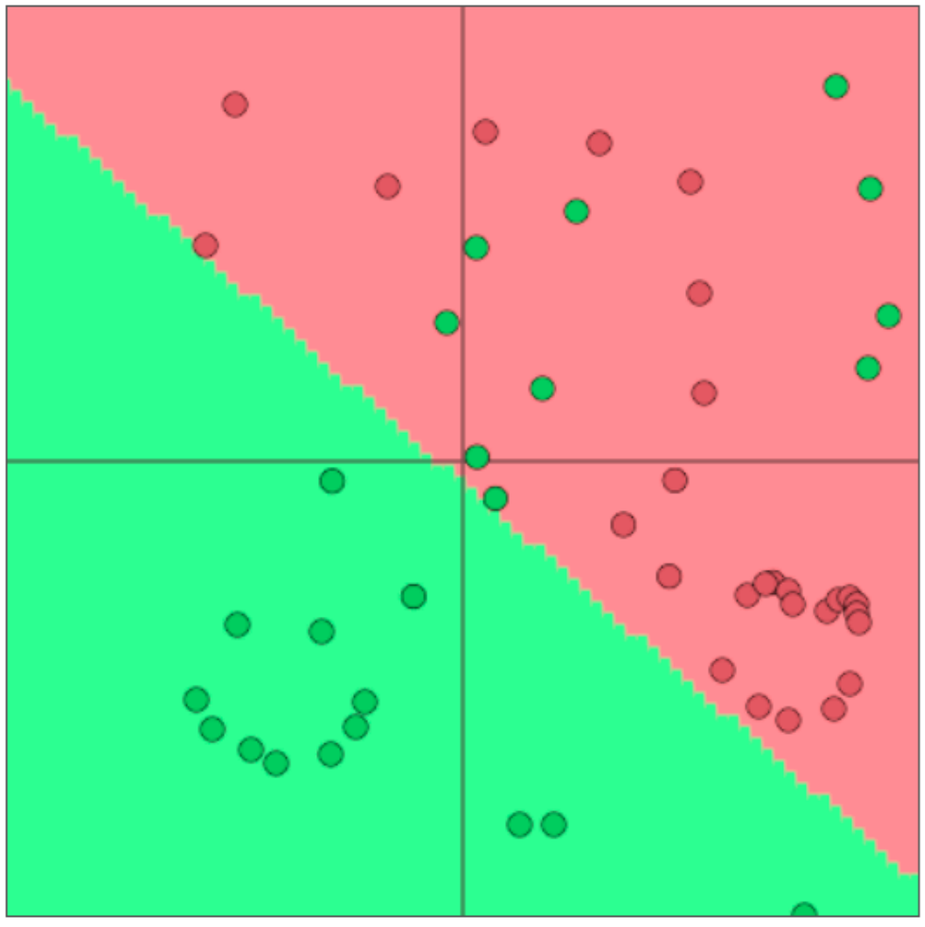
Non-linearity

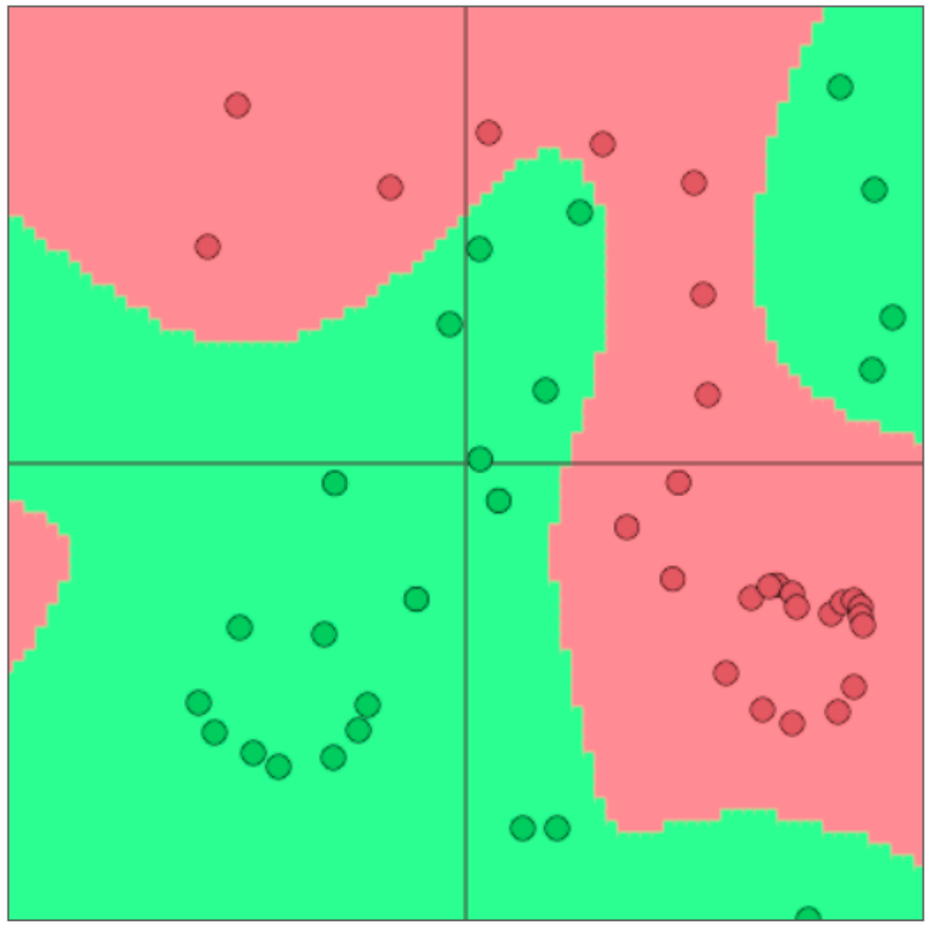
- Logistic (Softmax) Regression only gives linear decision boundaries



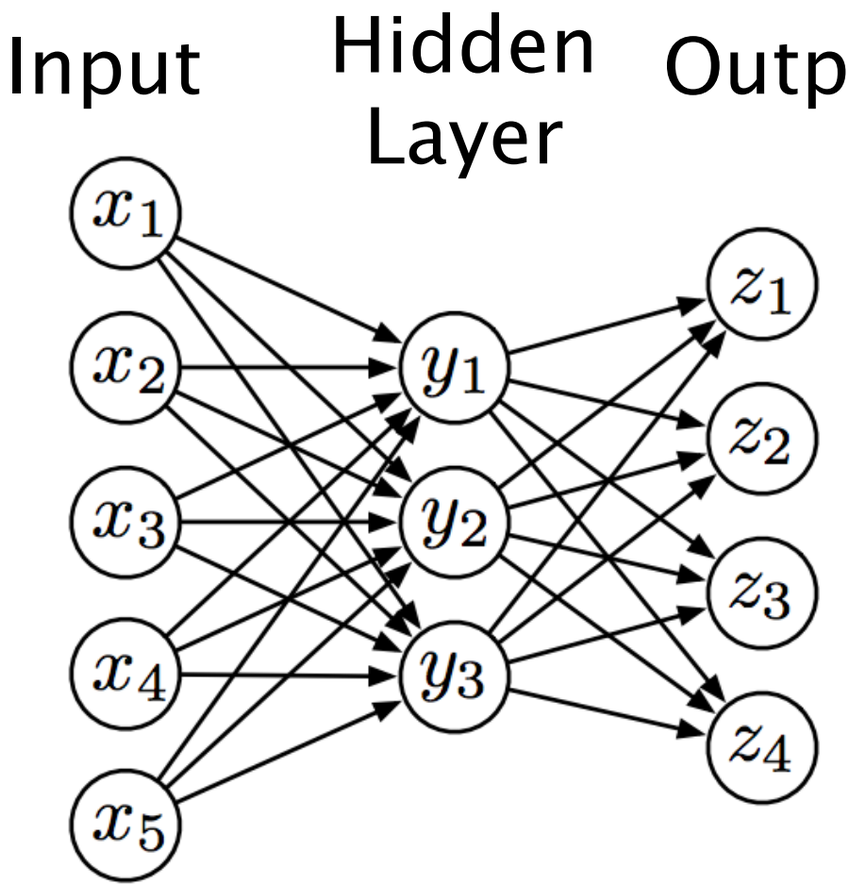
Non-linearity

- Neural networks can learn much more complex functions and nonlinear decision boundaries!





Non-linearity



$$\boldsymbol{y} = g(\mathbf{W}\boldsymbol{x} + \boldsymbol{b})$$

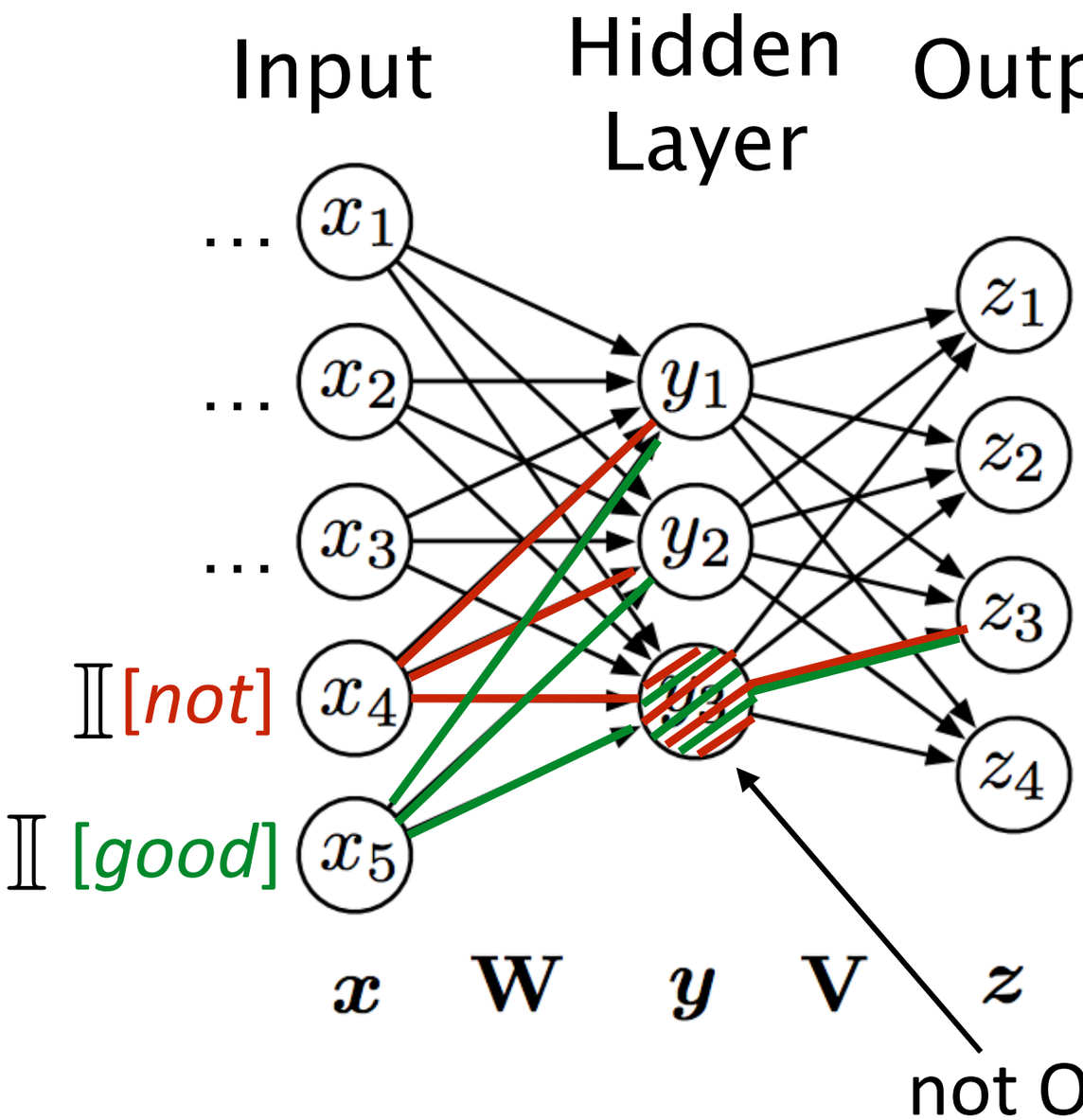
$$\mathbf{z} = g(\mathbf{V}\underbrace{g(\mathbf{W}\mathbf{x} + \mathbf{b})}_{\text{output of first layer}} + \mathbf{c})$$

With no nonlinearity:

$$\mathbf{z} = \mathbf{V}\mathbf{W}\mathbf{x} + \mathbf{V}\mathbf{b} + \mathbf{c}$$

Equivalent to $\mathbf{z} = \mathbf{U}\mathbf{x} + \mathbf{d}$

Non-linearity



Nodes in the hidden layer can learn interactions or conjunctions of features

$$y = -2x_1 - x_2 + 2\tanh(x_1 + x_2)$$

What about Word2vec (Skip-gram and CBOW)?

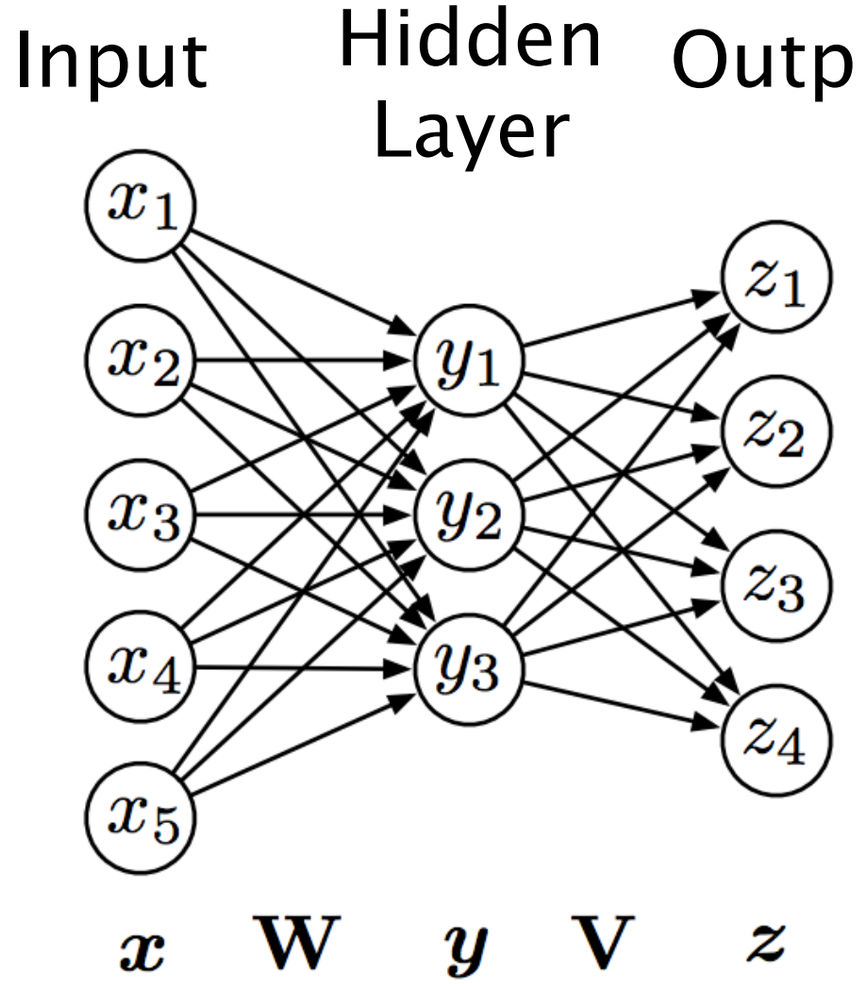
So, what about Word2vec (Skip-gram and CBOW)?

It is not deep learning — but “shallow” neural networks.

It is — in fact — a log-linear model (softmax regression).

So, it is faster over larger dataset yielding better embeddings.

Learning Neural Networks



change in output w.r.t. hidden

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

change in hidden w.r.t. input

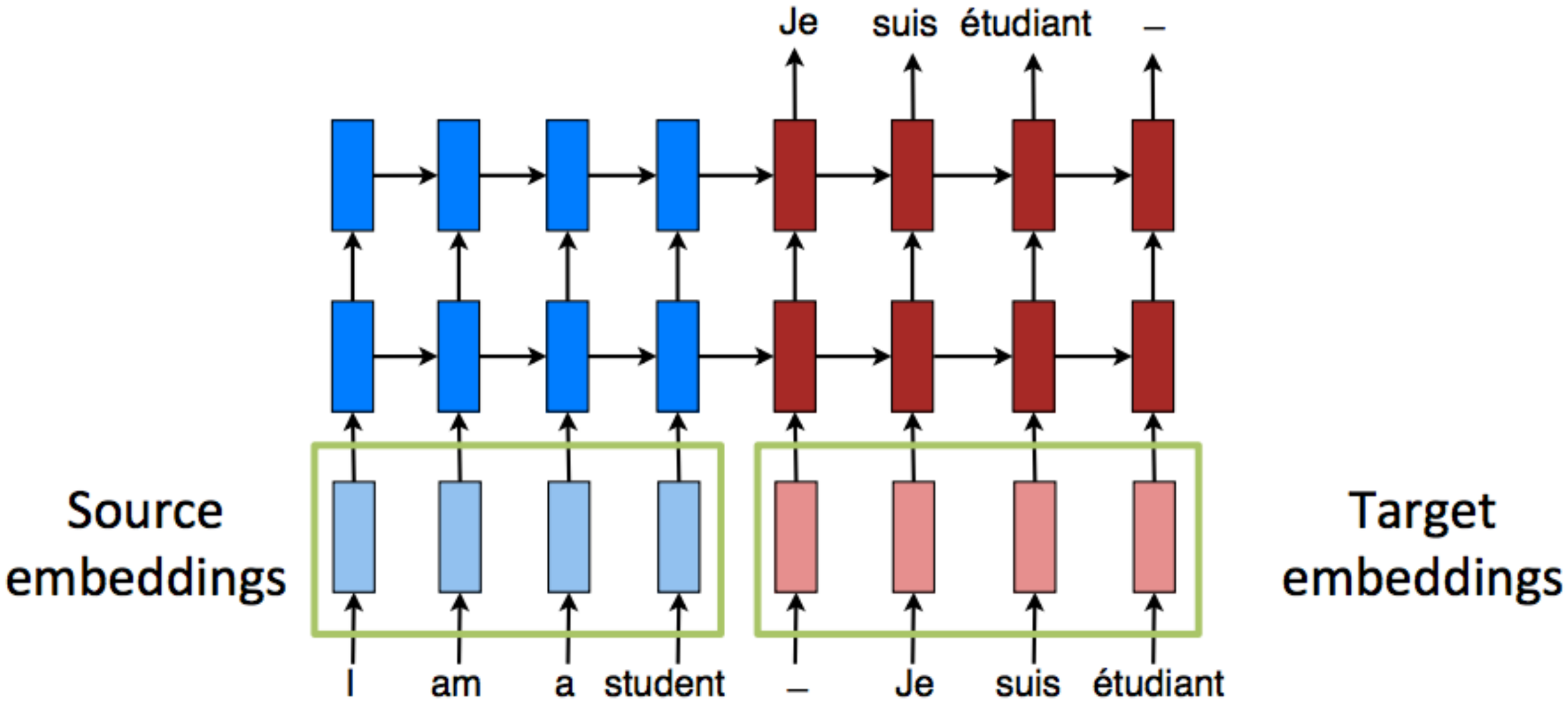
change in output w.r.t. input

Computing these looks like running this network in reverse (backpropagation)

Strategy for Successful NNs

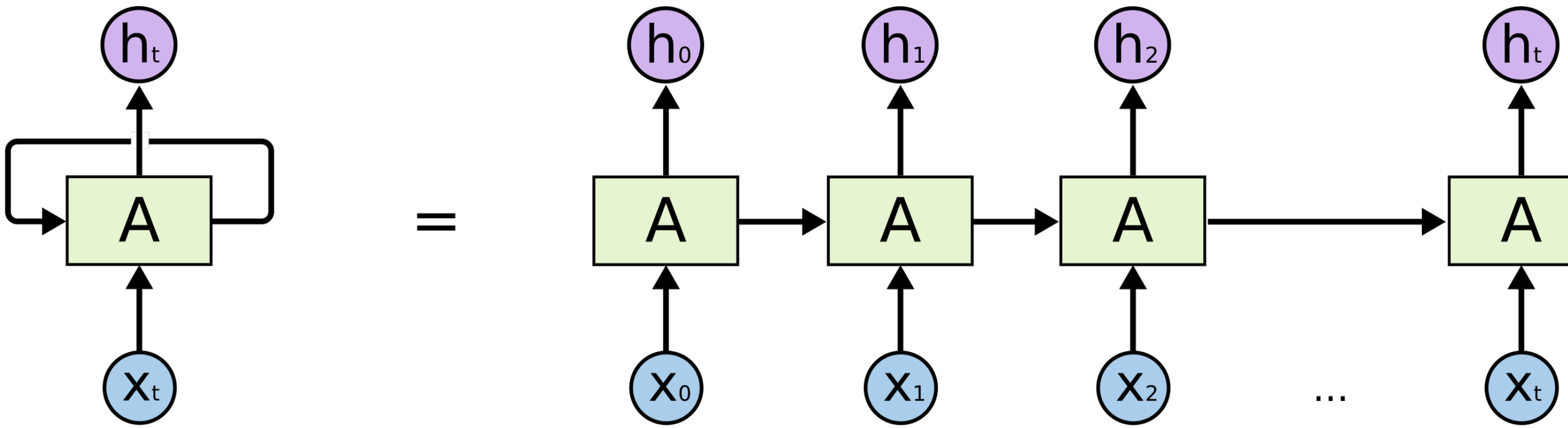
- Select network structure appropriate for problem
  - Structure: Single words, fixed windows, sentence based, document level; bag of words, recursive vs. recurrent, CNN, …
  - Nonlinearity
- Check for implementation bugs with gradient checks
- Parameter initialization
- Optimization tricks
- Check if the model is powerful enough to overfit
  - If not, change model structure or make model “larger”
  - If you can overfit: regularize

Neural Machine Translation



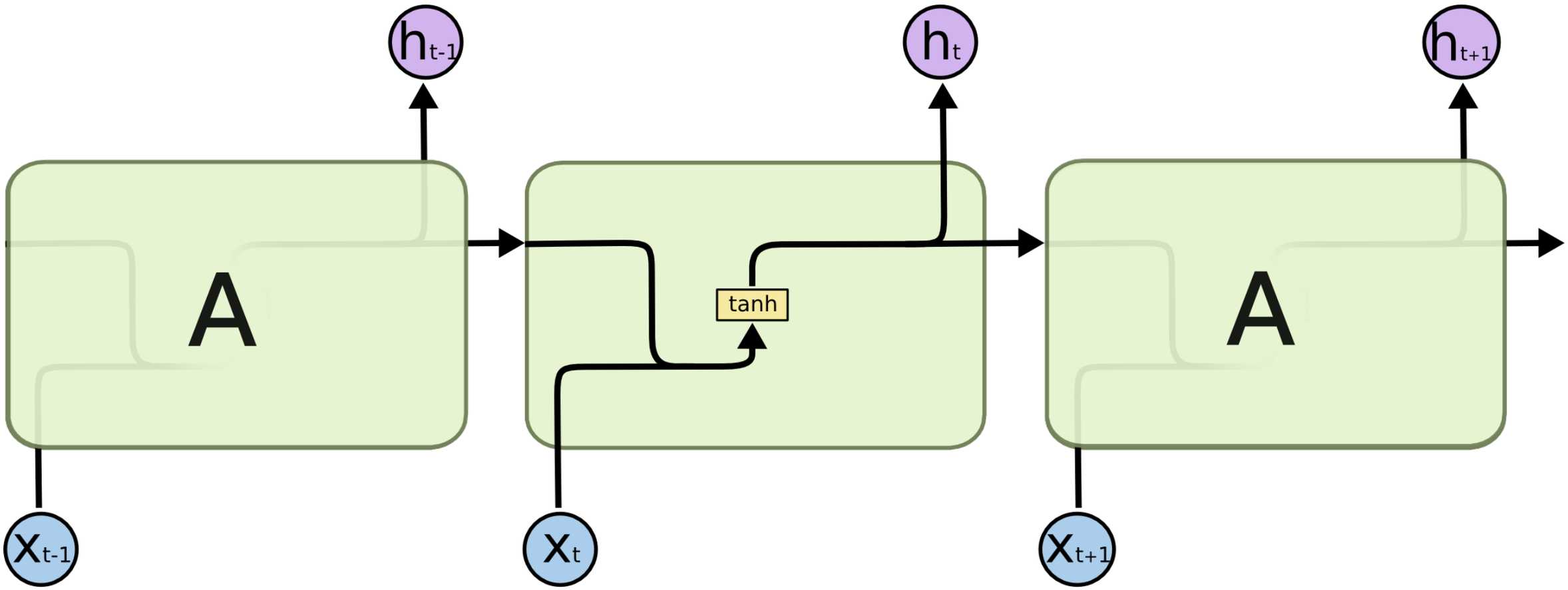
Source: Sequence to Sequence Learning with Neural Networks, Sutskever et al. (NIPS 2014)

Recurrent Neural Network (RNN)



Source: Colah's Blog

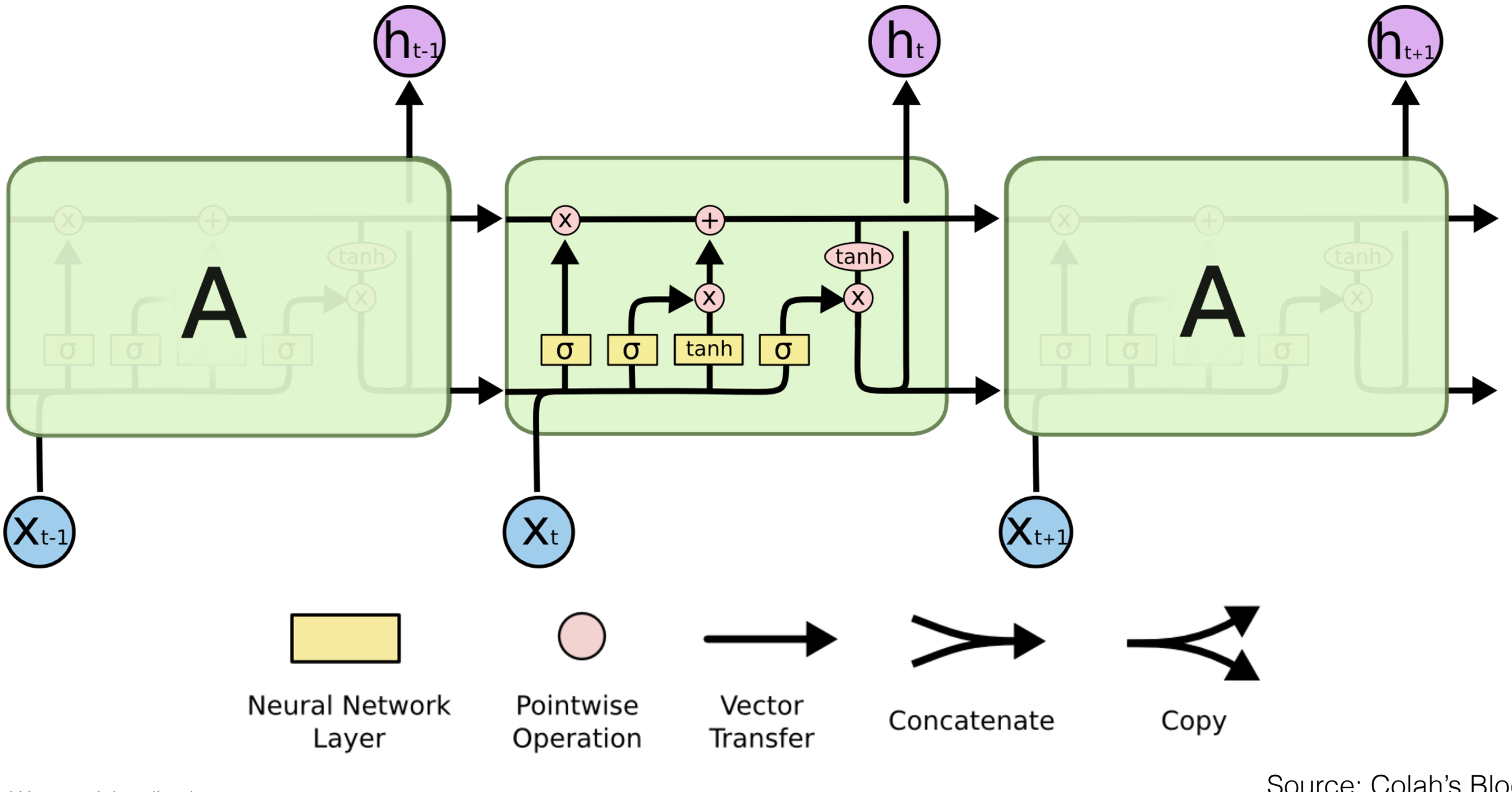
Recurrent Neural Network (RNN)



Source: Colah's Blog

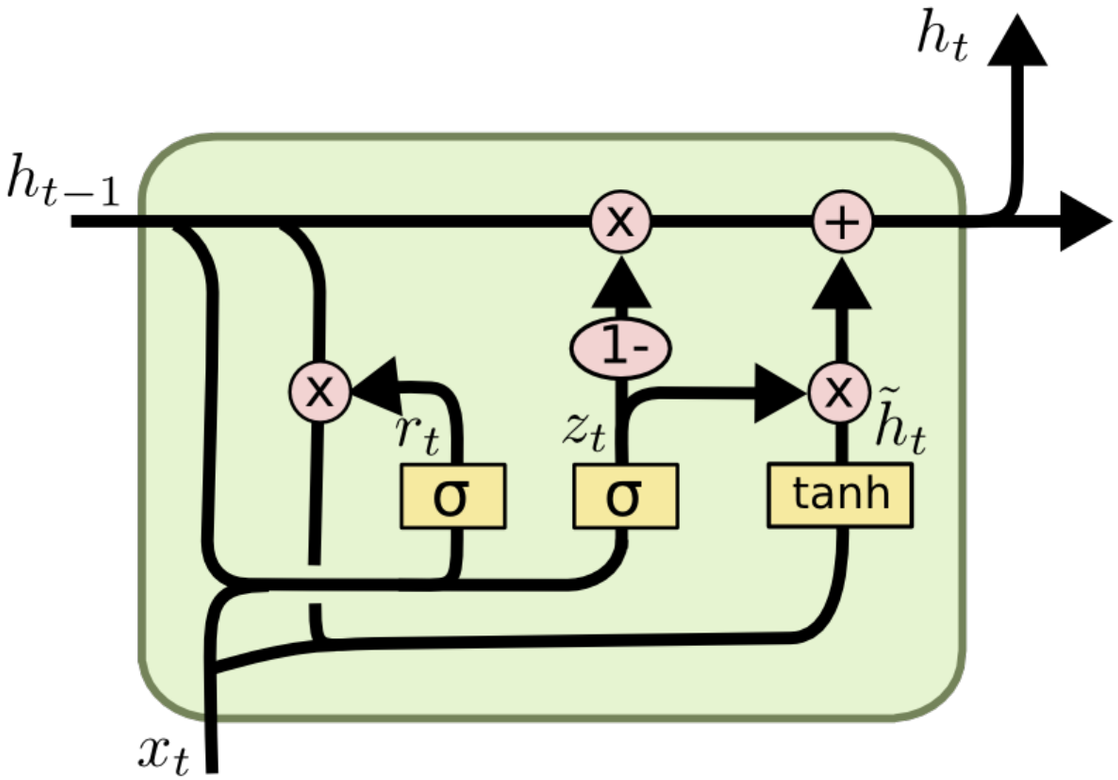
Long Short-Term Memory Networks (LSTM)

(Hochreiter & Schmidhuber, 1997)



Source: Colah's Blog

Long Short-Term Memory Networks (LSTM)

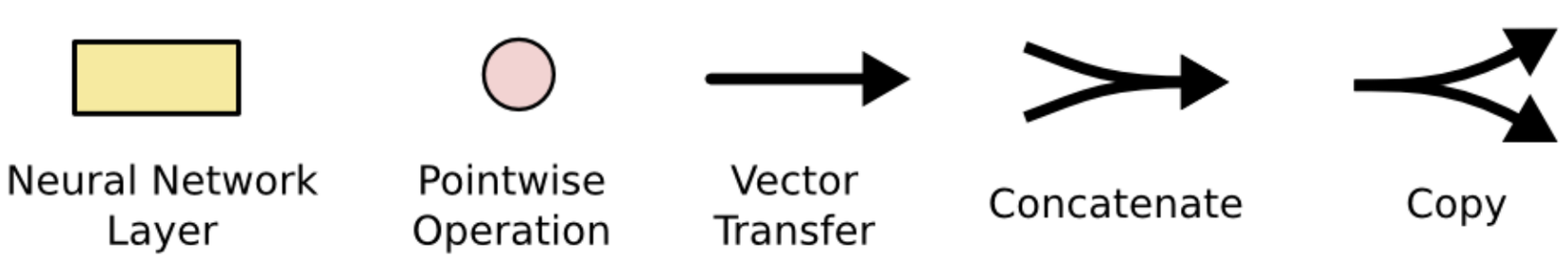


$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

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

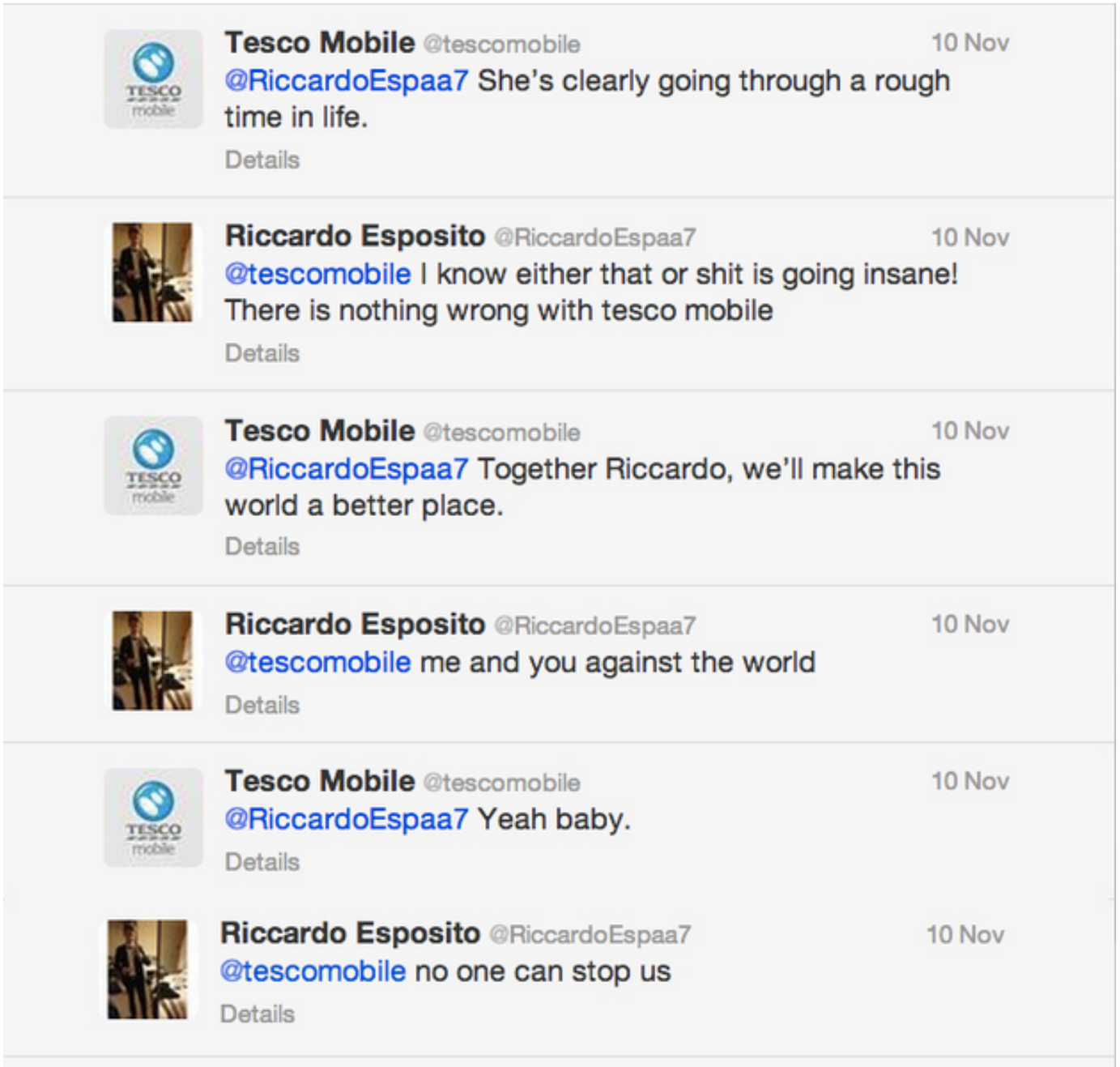
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

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

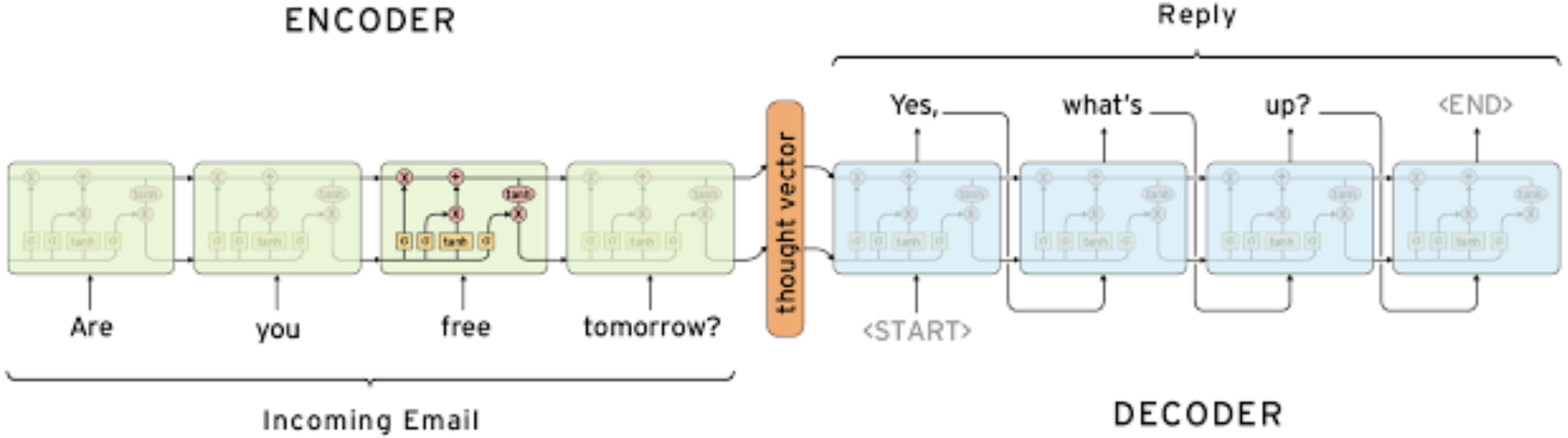


Source: Colah's Blog

Twitter Conversation Data



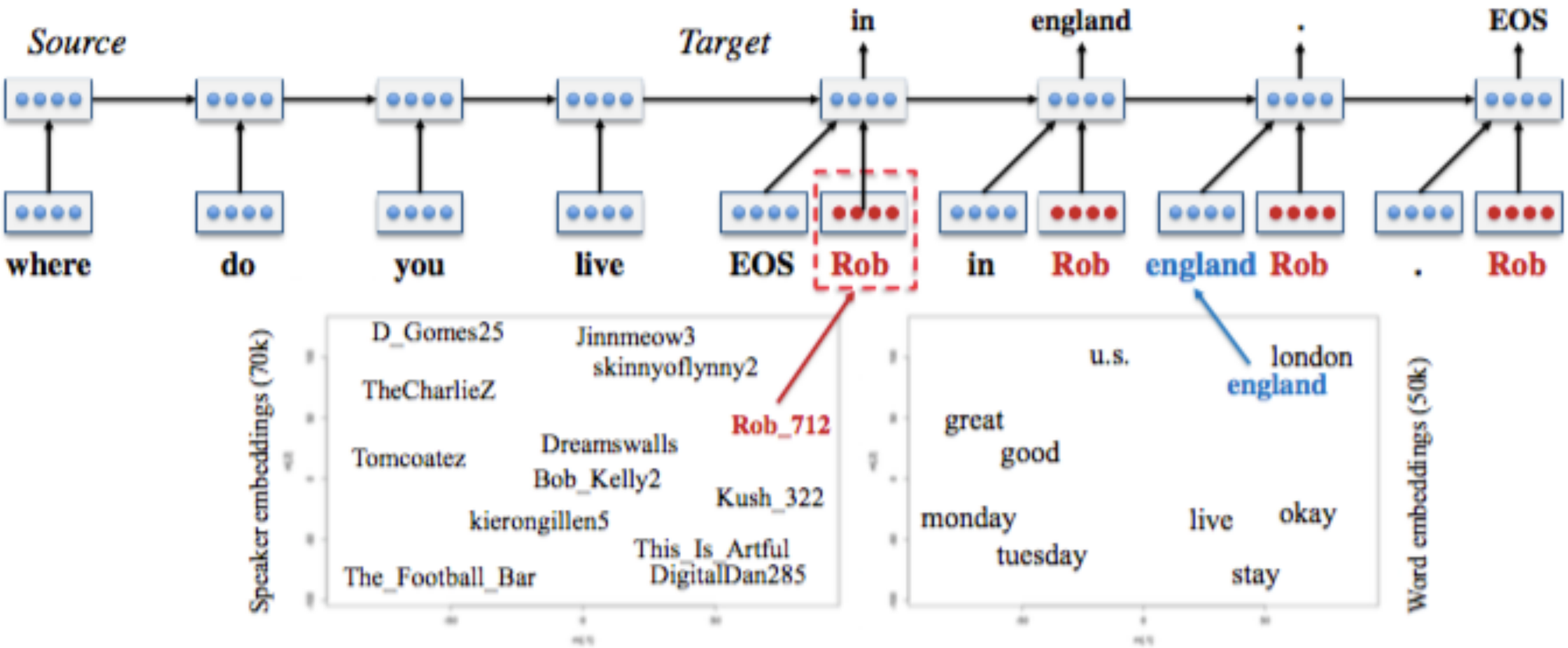
Neural Conversation



Source: Google's blog

Neural Conversation

modeling speakers

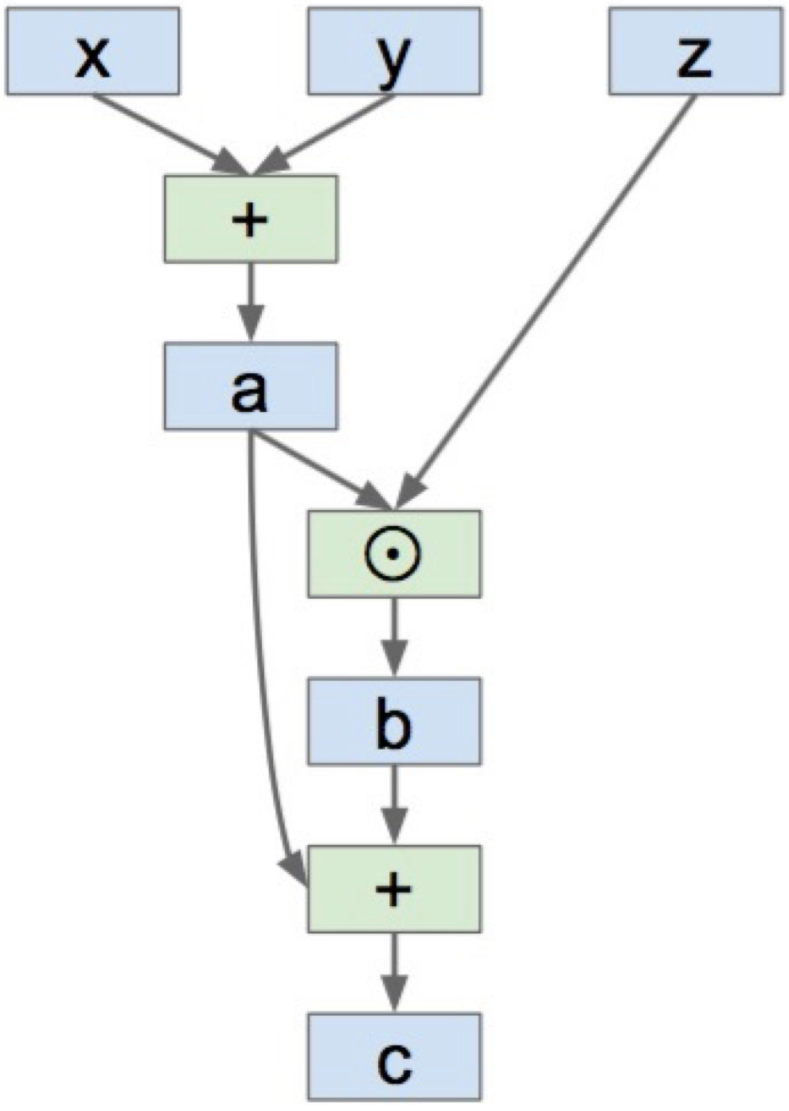


Source: A Persona-Based Neural Conversation Model, Li et al. (ACL 2016)

Neural Network Toolkits

- **Tensorflow**: https://www.tensorflow.org/
  - By Google, actively maintained, bindings for many languages
- **Theano**: http://deeplearning.net/software/theano
  - University of Montreal, less and less maintained
- **Torch**: http://torch.ch/
  - Facebook AI Research, Lua
- **DyNet**: https://github.com/clab/dynet
  - CMU and others, dynamic structures that change for every training instance
- **Caffe**: http://caffe.berkeleyvision.org/
  - UC Berkeley, for vision

Neural Network Toolkits




```python
import theano
import theano.tensor as T

# Define symbolic variables
x = T.matrix('x')
y = T.matrix('y')
z = T.matrix('z')

# Compute some other values symbolically
a = x + y
b = a * z
c = a + b

# Compile a function that computes c
f = theano.function(
      inputs=[x, y, z],
      outputs=c
    )

# Evaluate the compiled function
# on some real values
xx = np.random.randn(4, 5)
yy = np.random.randn(4, 5)
zz = np.random.randn(4, 5)
print f(xx, yy, zz)

# Repeat the same computation
# explicitly using numpy ops
aa = xx + yy
bb = aa * zz
cc = aa + bb
```

Compile a function that produces c from x, y, z (generates code)

Source: http://tmmse.xyz/content/images/2016/02/theano-computation-graph.png

Follow @cocoweixu

Instructor: Wei Xu

www.cis.upenn.edu/~xwe/

Course Website: socialmedia-class.org