Lecture 22
Neural Networks

Nathan Schneider

(slides from Chris Manning, Yoav Artzi, Greg Durrett)

ANLP | 29 November 2017

Natural Language Processing with Deep Learning

CS224N/Ling284

Christopher Manning and Richard Socher

Lecture 1: Introduction

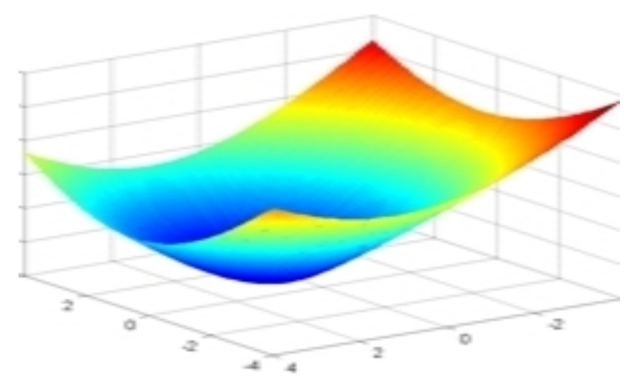
2. What’s Deep Learning (DL)?

- **Deep learning** is a subfield of **machine learning**

- Most machine learning methods work well because of **human-designed representations** and **input features**
  - For example: features for finding named entities like locations or organization names (Finkel et al., 2010):

Feature	**NER**
Current Word	✓
Previous Word	✓
Next Word	✓
Current Word Character n-gram	all
Current POS Tag	✓
Surrounding POS Tag Sequence	✓
Current Word Shape	✓
Surrounding Word Shape Sequence	✓
Presence of Word in Left Window	size 4
Presence of Word in Right Window	size 4

- Machine learning becomes just optimizing weights to best make a final prediction



Machine Learning vs. Deep Learning

Machine Learning in Practice

Describing your data with features a computer can understand

Learning algorithm

Domain specific, requires Ph.D. level talent

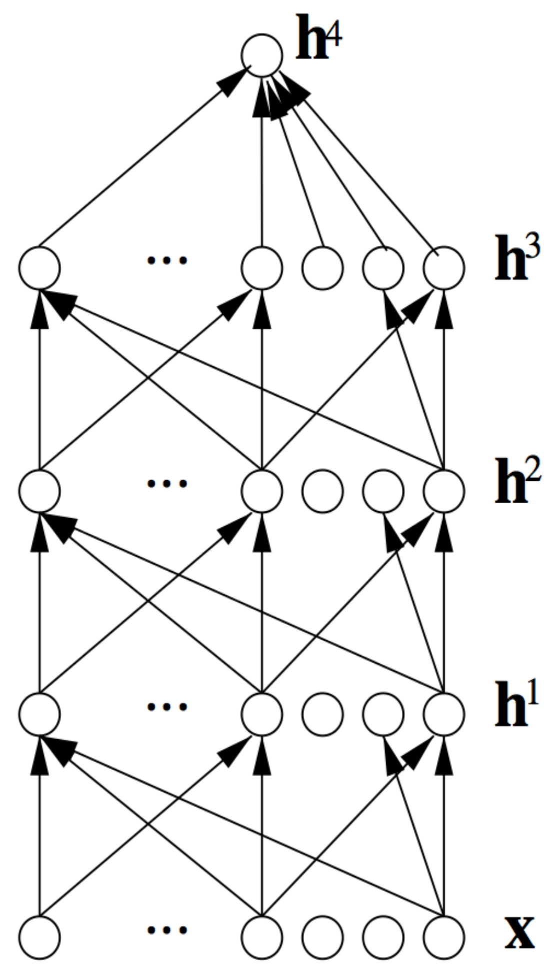
Optimizing the weights on features

What's Deep Learning (DL)?

- Representation learning attempts to automatically learn good features or representations

- Deep learning algorithms attempt to learn (multiple levels of) representation and an output

- From "raw" inputs **x** (e.g., sound, characters, or words)



On the history of and term “Deep Learning”

- We will focus on different kinds of **neural networks**
- The dominant model family inside deep learning

- Only clever terminology for stacked logistic regression units?
  - Maybe, but interesting modeling principles (end-to-end) and actual connections to neuroscience in some cases

- We will not take a historical approach but instead focus on methods which work well on NLP problems now
- For a long (!) history of deep learning models (starting ~1960s), see: Deep Learning in Neural Networks: An Overview by Jürgen Schmidhuber

Reasons for Exploring Deep Learning

- Manually designed features are often over-specified, incomplete and take a long time to design and validate

- **Learned Features** are easy to adapt, fast to learn

- Deep learning provides a very flexible, (almost?) universal, learnable framework for **representing** world, visual and linguistic information.

- Deep learning can learn **unsupervised** (from raw text) and **supervised** (with specific labels like positive/negative)

Reasons for Exploring Deep Learning

- In ~2010 **deep** learning techniques started outperforming other machine learning techniques. Why this decade?

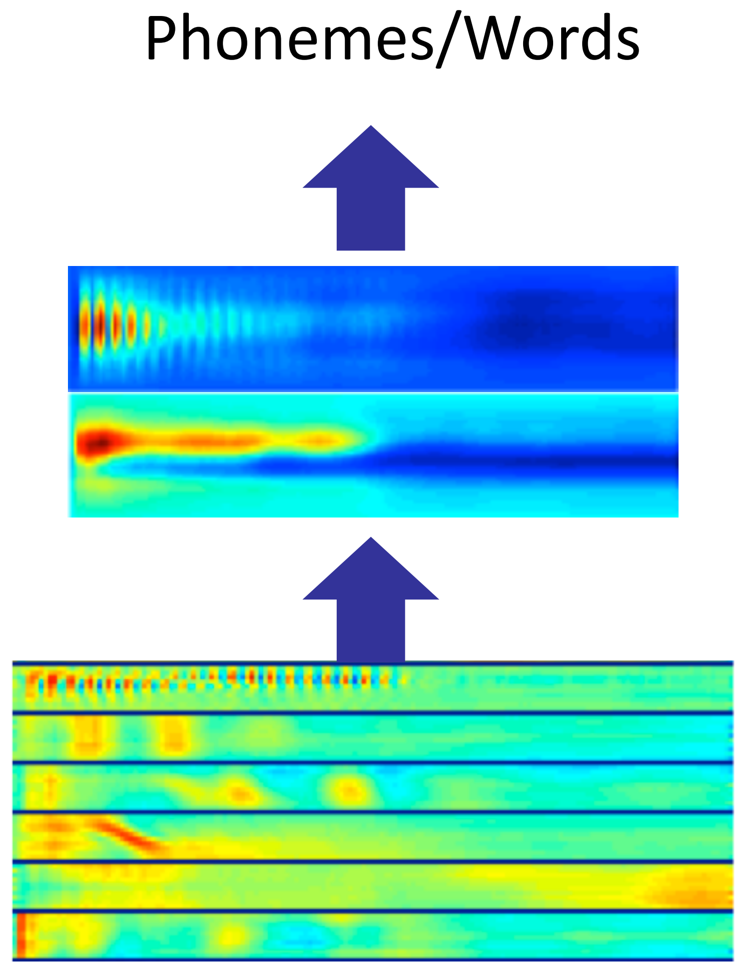
- Large amounts of training data favor deep learning
- Faster machines and multicore CPU/GPUs favor Deep Learning
- New models, algorithms, ideas
  - Better, more flexible learning of intermediate representations
  - Effective end-to-end joint system learning
  - Effective learning methods for using contexts and transferring between tasks

→ **Improved performance** (first in speech and vision, then NLP)

Deep Learning for Speech

- The first breakthrough results of "deep learning" on large datasets happened in speech recognition
- Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition Dahl et al. (2010)

Acoustic model	Recog WER	RT03S FSH	Hub5 SWB
Traditional features	1-pass −adapt	**27.4**	**23.6**
Deep Learning	1-pass −adapt	**18.5** (−33%)	**16.1** (−32%)



Deep Learning for Computer Vision

Most deep learning groups have focused on computer vision (at least till 2 years ago)

The breakthrough DL paper: ImageNet Classification with Deep Convolutional Neural Networks by Krizhevsky, Sutskever, & Hinton, 2012, U. Toronto. 37% error red.

Zeiler and Fergus (2013)

5. Deep NLP = Deep Learning + NLP

Combine ideas and goals of NLP with using representation learning and deep learning methods to solve them

Several big improvements in recent years in NLP with different

- **Levels**: speech, words, syntax, semantics
- **Tools:** parts-of-speech, entities, parsing
- **Applications**: machine translation, sentiment analysis, dialogue agents, question answering

Word meaning as a neural word vector – visualization

$$
expect = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \\ 0.487 \end{pmatrix}
$$

Word similarities

Nearest words to frog:

1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

litoria

leptodactylidae

rana

eleutherodactylus

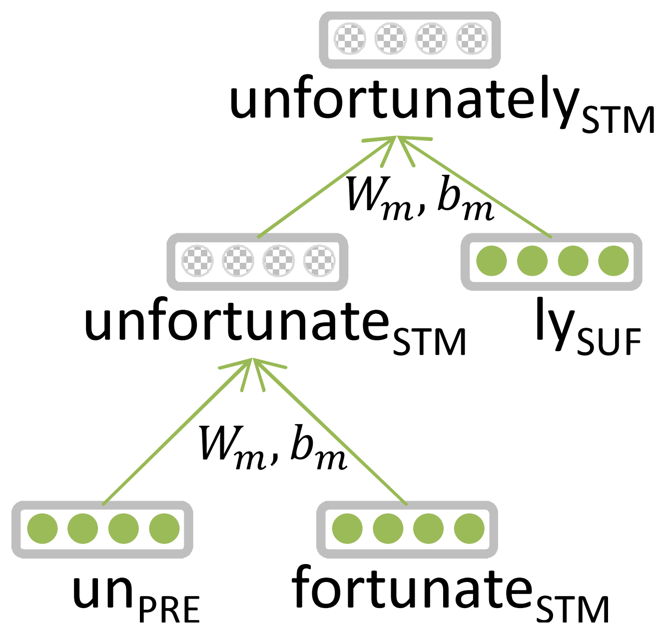
http://nlp.stanford.edu/projects/glove/

Representations of NLP Levels: Morphology

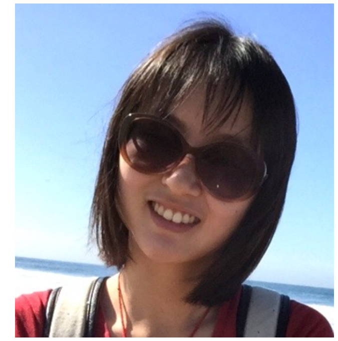
- Traditional: Words are made of morphemes

prefix	stem	suffix
un	interest	ed

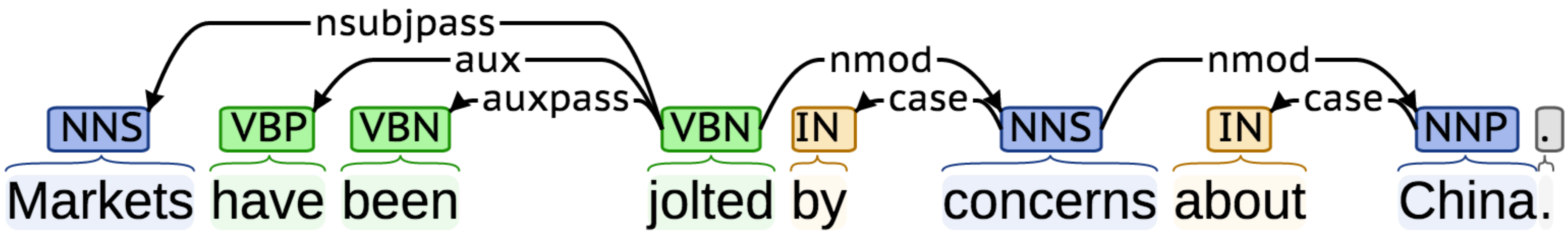
- DL:
  - every morpheme is a vector
  - a neural network combines two vectors into one vector
  - Luong et al. 2013



NLP Tools: Parsing for sentence structure



Neural networks can accurately determine the structure of sentences, supporting interpretation



Softmax layer:

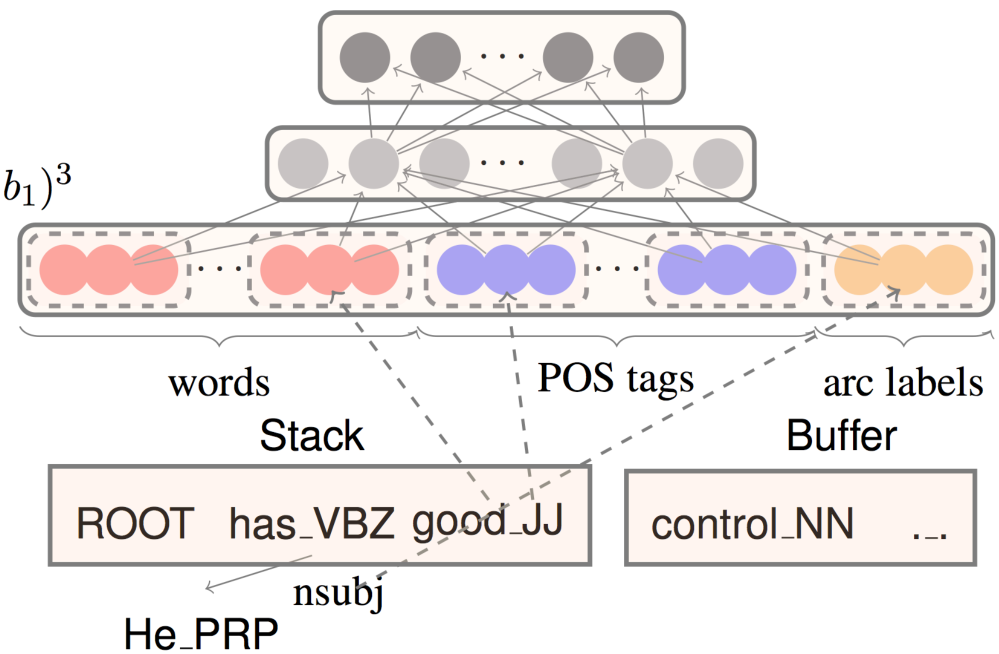
$$p = \texttt{softmax}(W_2 h)$$

Hidden layer:

$$h = (W_1^w x^w + W_1^t x^t + W_1^l x^l + b_1)^3$$

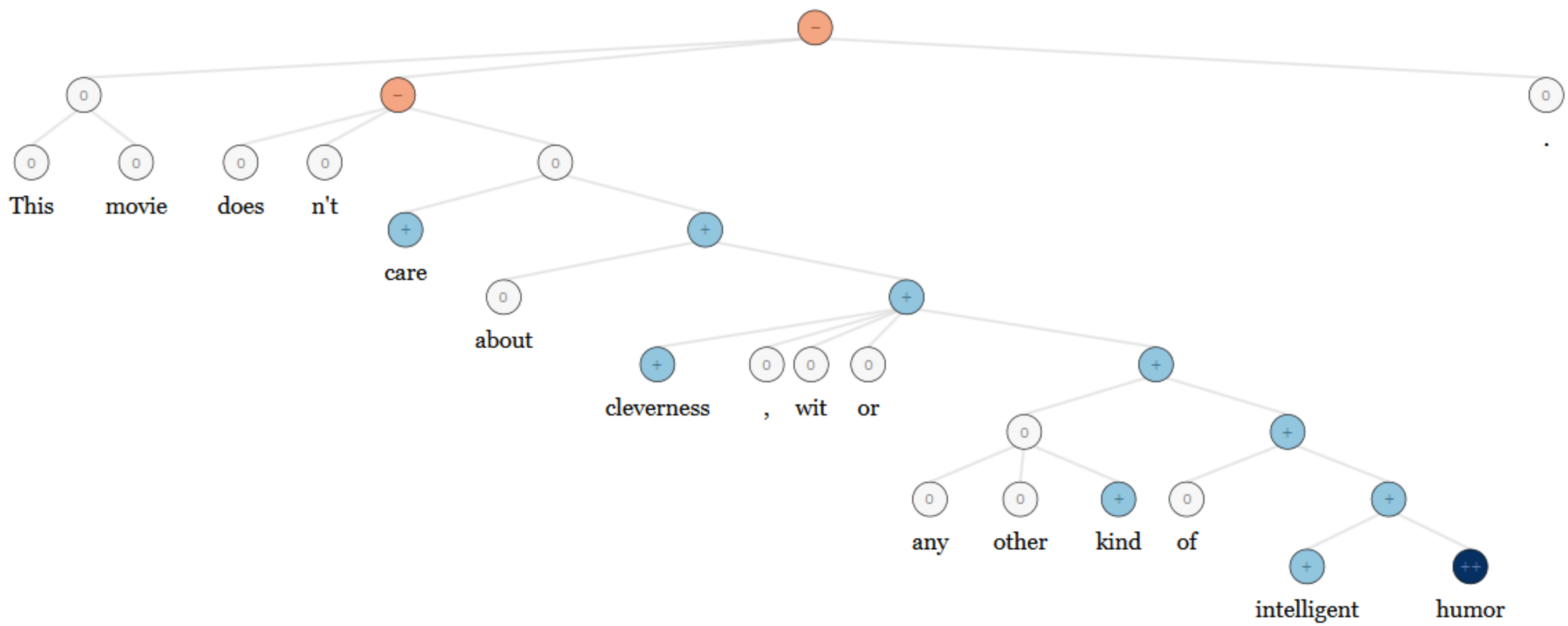
Input layer: $[x^w, x^t, x^l]$

Configuration



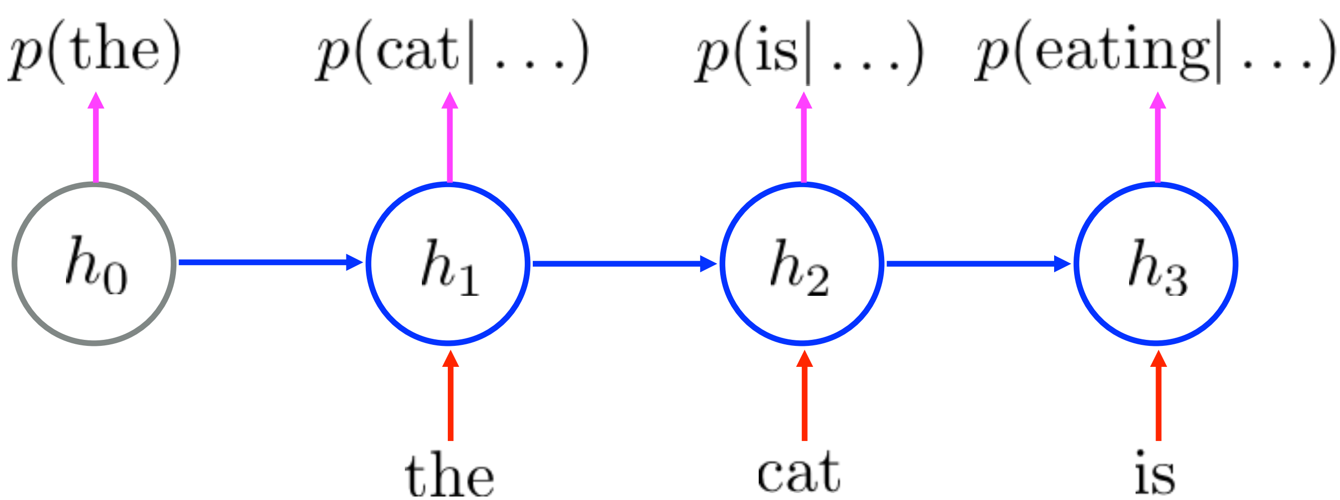
NLP Applications: Sentiment Analysis

- Traditional: Curated sentiment dictionaries combined with either bag-of-words representations (ignoring word order) or hand-designed negation features (ain’t gonna capture everything)
- Same deep learning model that was used for morphology, syntax and logical semantics can be used! → RecursiveNN



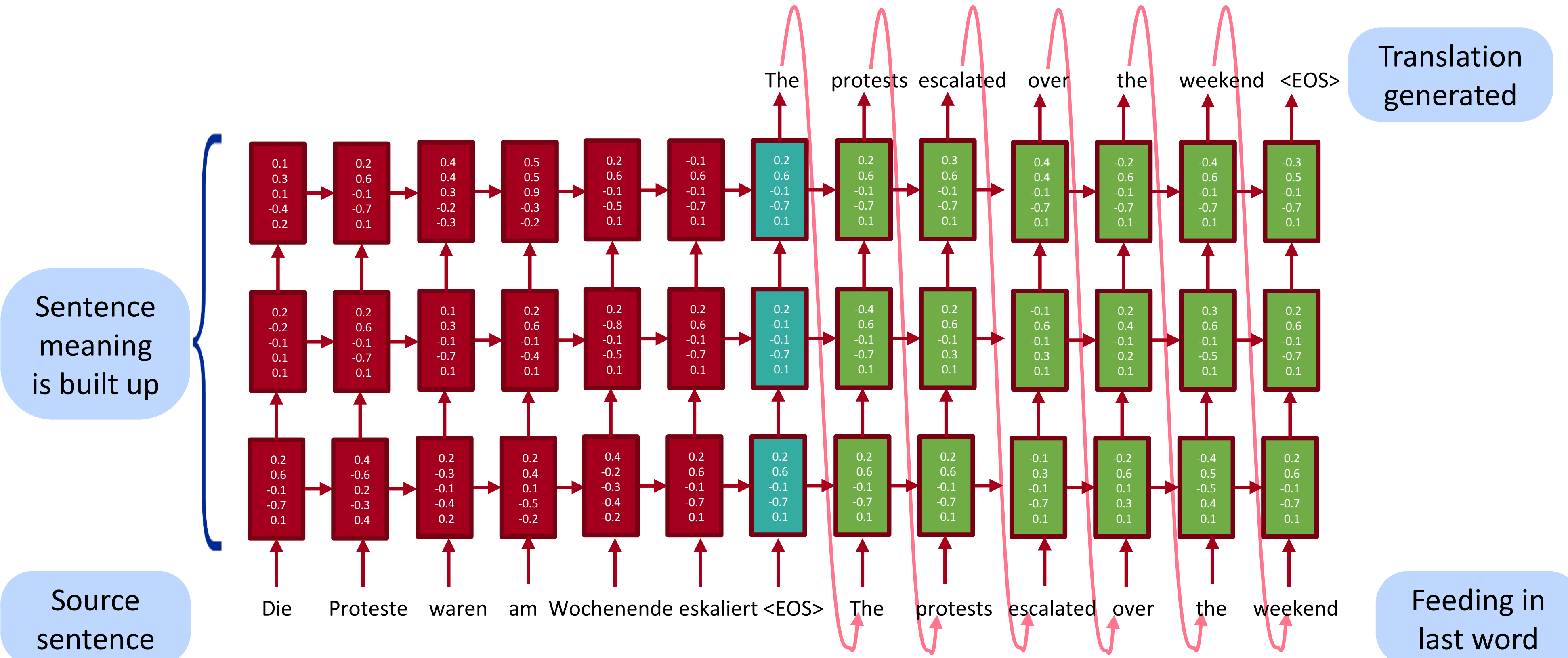
Dialogue agents / Response Generation

- A simple, successful example is the auto-replies available in the Google Inbox app
- An application of the powerful, general technique of **Neural Language Models**, which are an instance of Recurrent Neural Networks



Neural Machine Translation

Source sentence is mapped to **vector**, then output sentence generated
[Sutskever et al. 2014, Bahdanau et al. 2014, Luong and Manning 2016]



Now live for some languages in Google Translate (etc.), with big error reductions!

CS5740: Natural Language Processing
Spring 2017

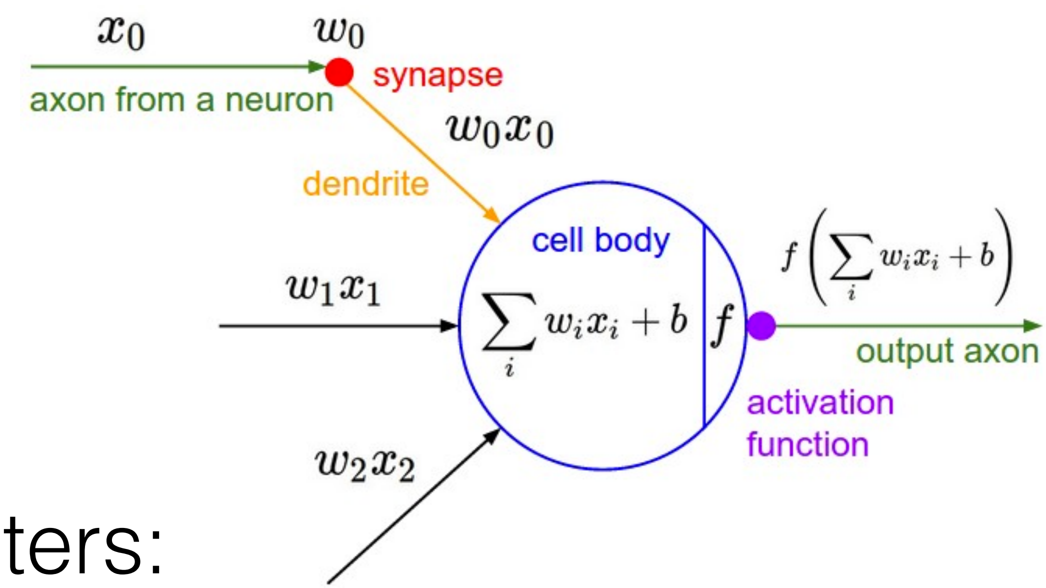
Neural Networks

Instructor: Yoav Artzi

Slides adapted from Dan Klein, Dan Jurafsky, Chris Manning, Michael Collins, Luke Zettlemoyer, Yejin Choi, and Slav Petrov

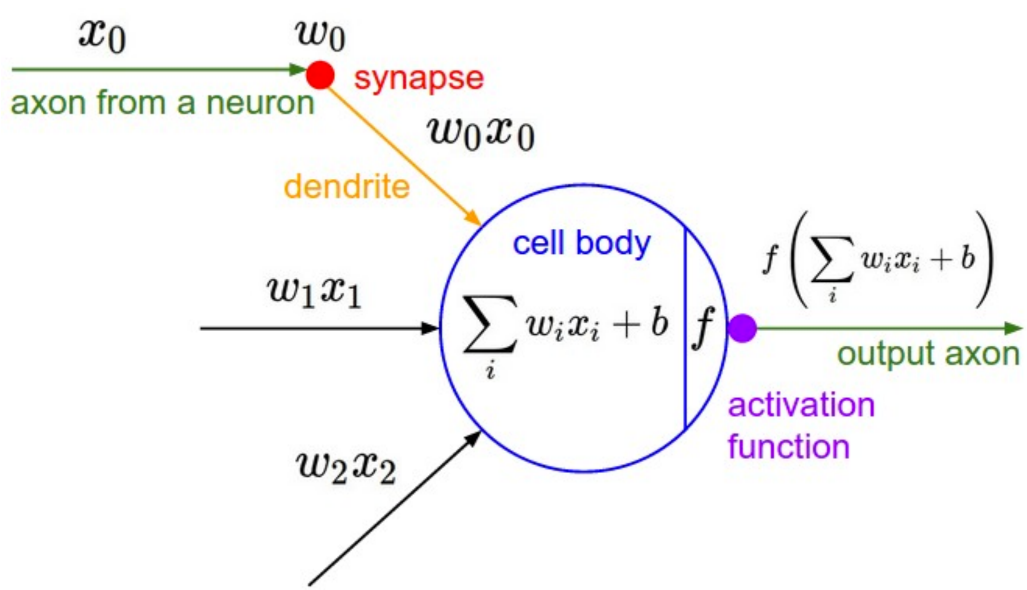
Neuron

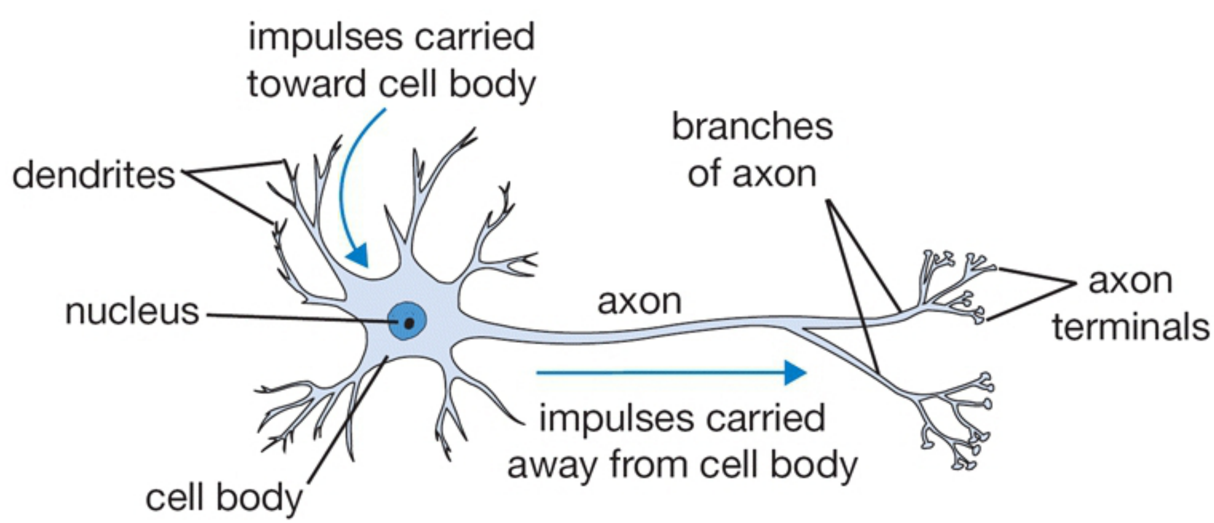
- Neural networks comes with their terminological baggage



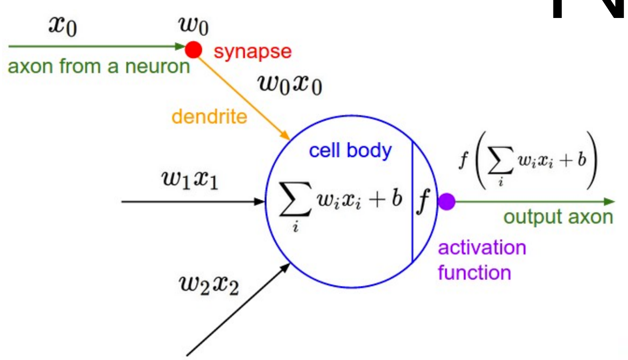
- Parameters:
  - Weights: w_i and b
  - Activation function
- If we drop the activation function, reminds you of something?

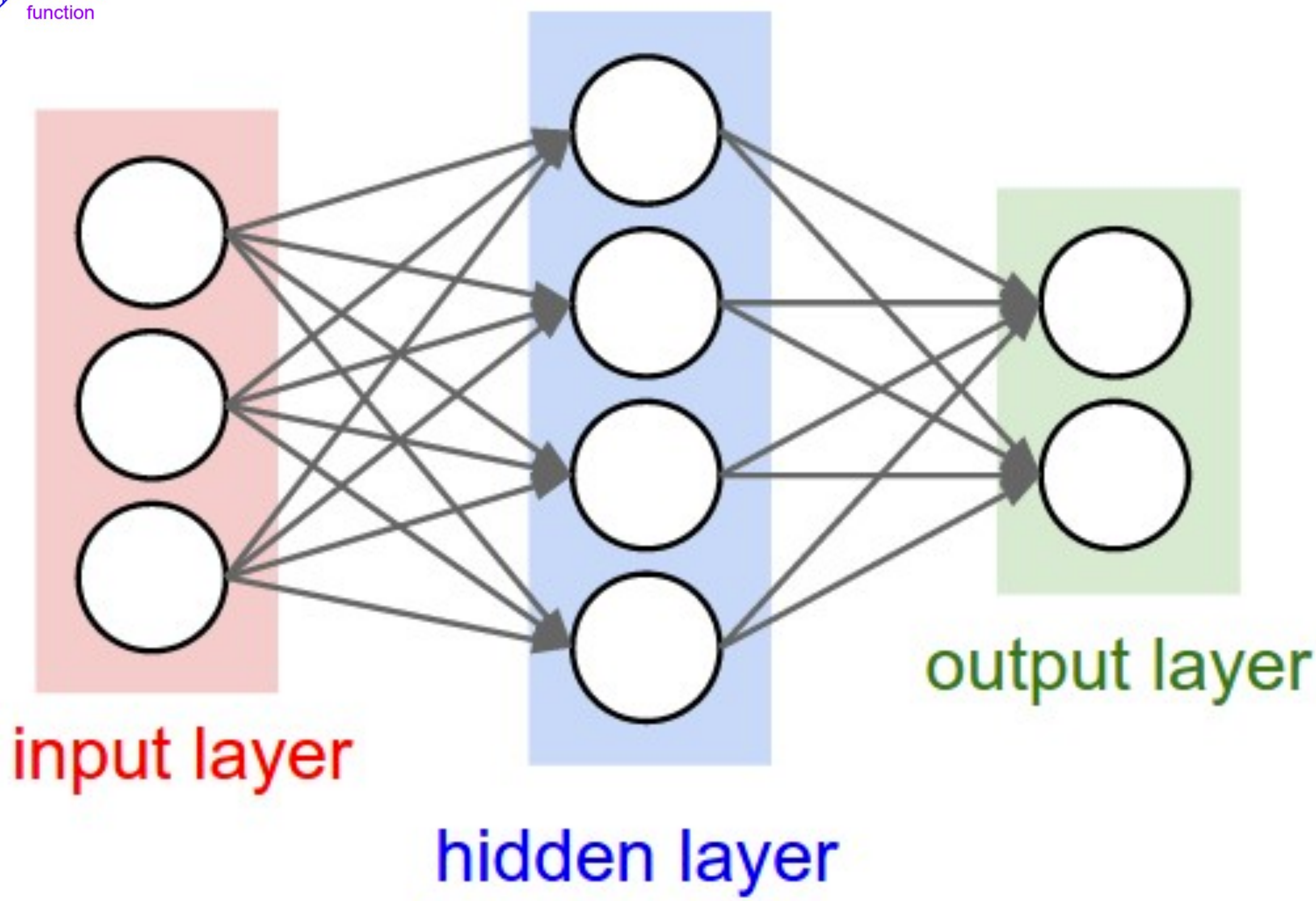
Biological “Inspiration”



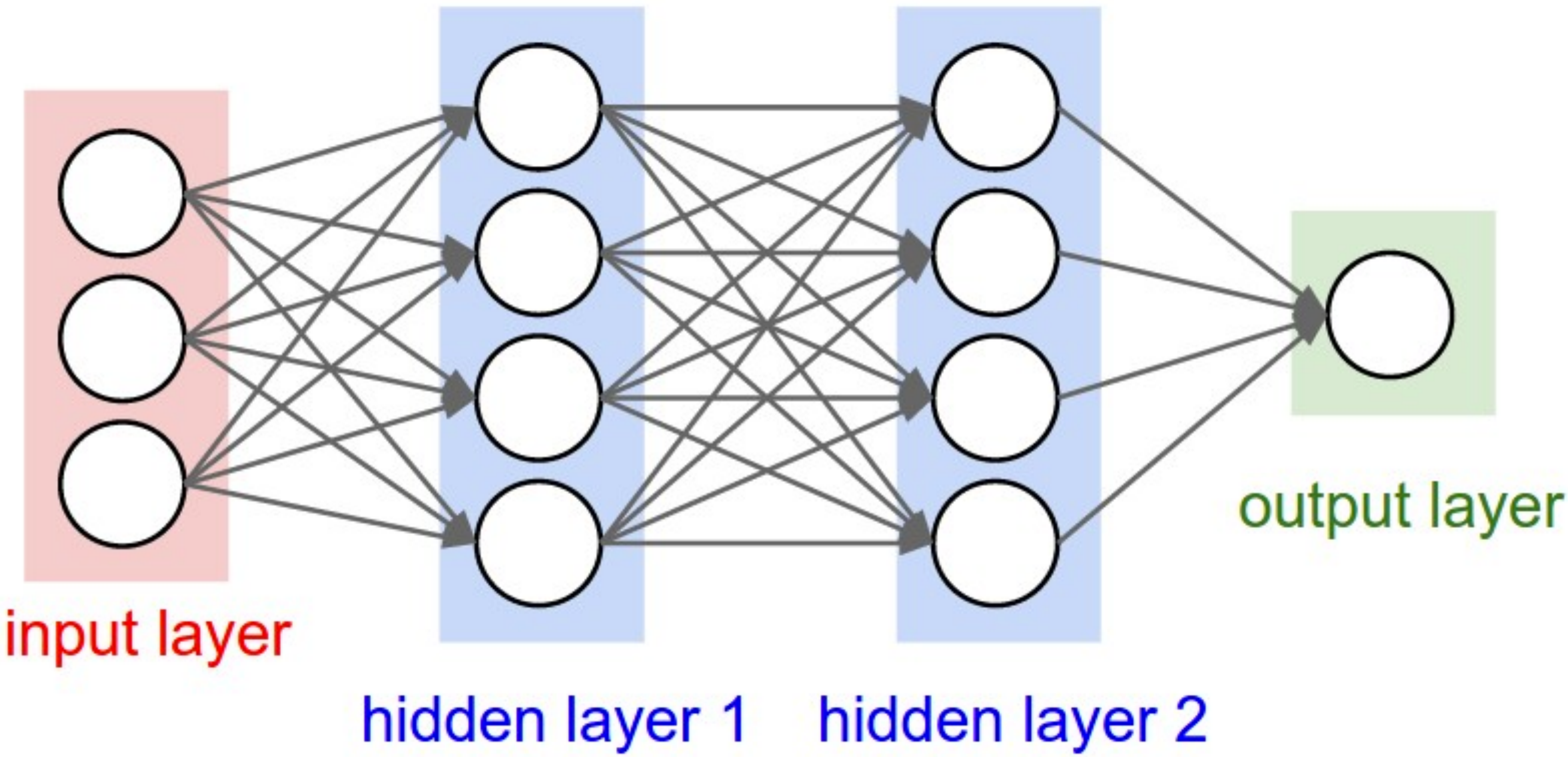


Neural Network





Neural Network

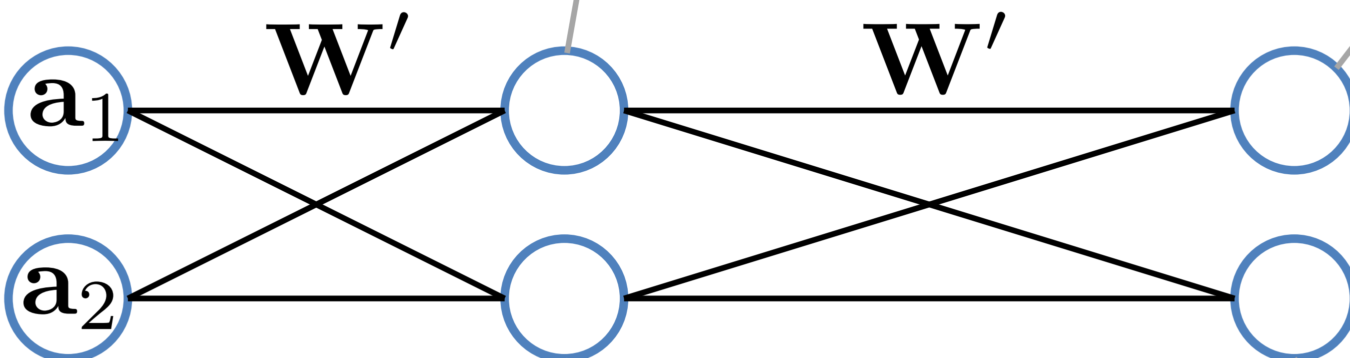


Matrix Notation

$$\mathbf{W}''(\mathbf{W}'\mathbf{a} + \mathbf{b}') + \mathbf{b}''$$

$$\mathbf{o}_1 = \mathbf{W}''_{11}\mathbf{h}_1 + \mathbf{W}''_{12}\mathbf{h}_2 + \mathbf{b}''_1$$

$$\mathbf{h}_1 = \mathbf{W}'_{11}\mathbf{a}_1 + \mathbf{W}'_{12}\mathbf{a}_2 + \mathbf{b}'_1$$



$$\mathbf{h}_2 = \mathbf{W}'_{21}\mathbf{a}_1 + \mathbf{W}'_{22}\mathbf{a}_2 + \mathbf{b}'_2$$

$$\mathbf{o}_2 = \mathbf{W}''_{21}\mathbf{h}_1 + \mathbf{W}''_{22}\mathbf{h}_2 + \mathbf{b}''_2$$

Word Representations

- One-hot vectors:

  hotel = [0 0 0 0 … 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0]
  conference = [0 0 0 0 … 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0]
  hotels = [0 0 0 0 … 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]

  – Problems?
  – Information sharing?
    - "hotel" vs. "hotels"

Word Embeddings

- Each word is represented using a dense low-dimensional vector
  - Low-dimensional << vocabulary size
- If trained well, similar words will have similar vectors
- How to train? What objective to maximize?
  - Soon …

Word Embeddings as Features

- Example: sentiment classification
  – very positive, positive, neutral, negative, very negative
- Feature-based models: bag of words
- Any good neural net architecture?
  – Concatenate all the vectors
    - Problem: different document → different length
  – Instead: sum, average, etc.

Neural Networks for NLP

Greg Durrett

i256: Applied Natural Language Processing

October 24, 2016

Slides at http://www.cs.utexas.edu/~gdurrett/lectures/

Sentiment Analysis

the movie was very good 👍

Sentiment Analysis with Linear Models

Example	Label	Feature	Type
the movie was very good	👍	$\mathbb{I}[\textit{good}]$	Unigrams
the movie was very bad	👎	$\mathbb{I}[\textit{bad}]$	Unigrams
the movie was not bad	👍	$\mathbb{I}[\textit{not bad}]$	Bigrams
the movie was not very good	👎	$\mathbb{I}[\textit{not very good}]$	Trigrams
the movie was not really very enjoyable			4-grams!

Drawbacks

- More complex features capture interactions but scale badly (13M unigrams, 1.3B 4-grams in Google *n*-grams)

- Can we do better than seeing every *n*-gram once in the training data?

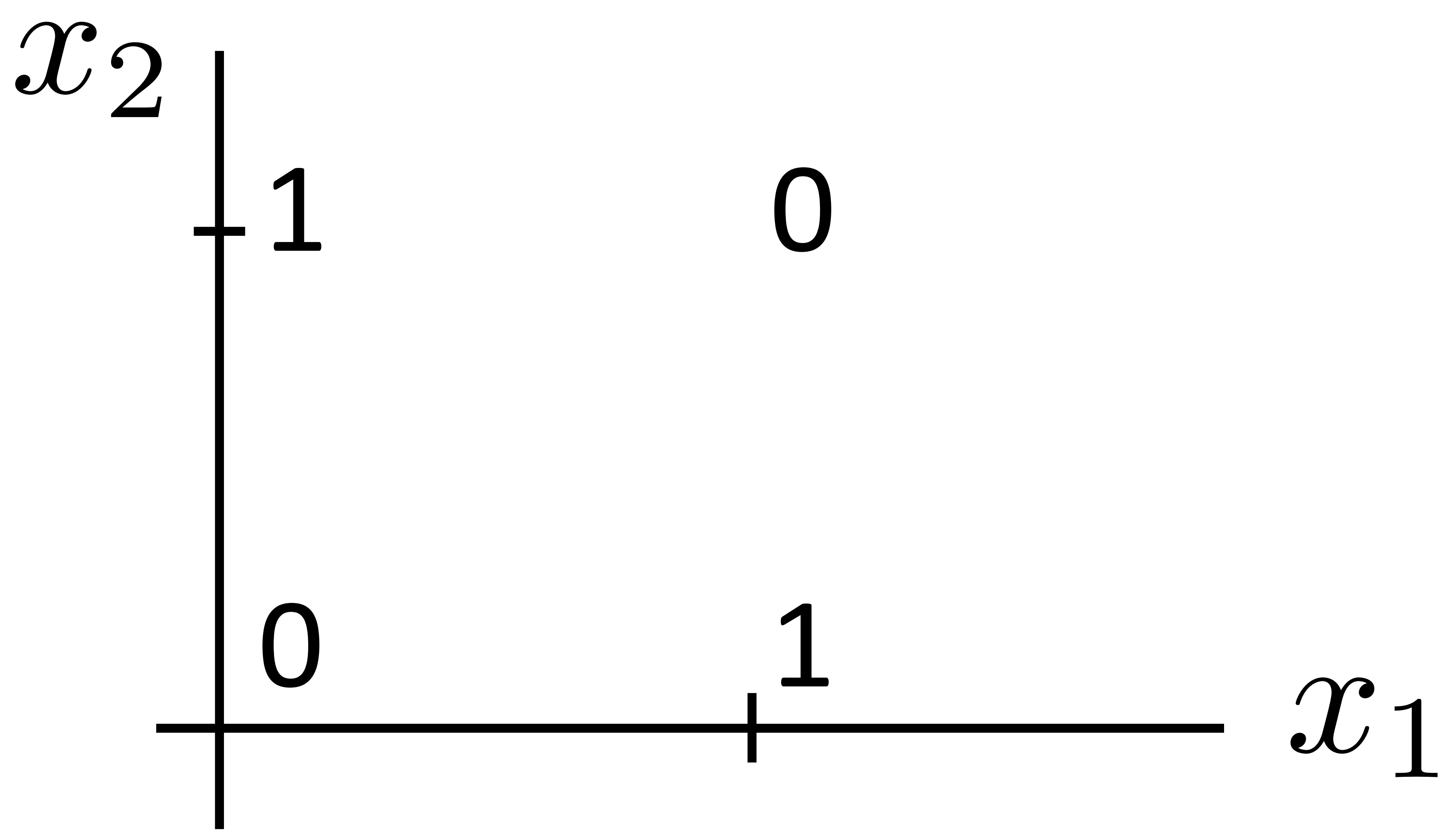
  not very good *not so great*

- Instead of more complex linear functions, let's use *simpler nonlinear functions*, namely neural networks

the movie was not really very enjoyable

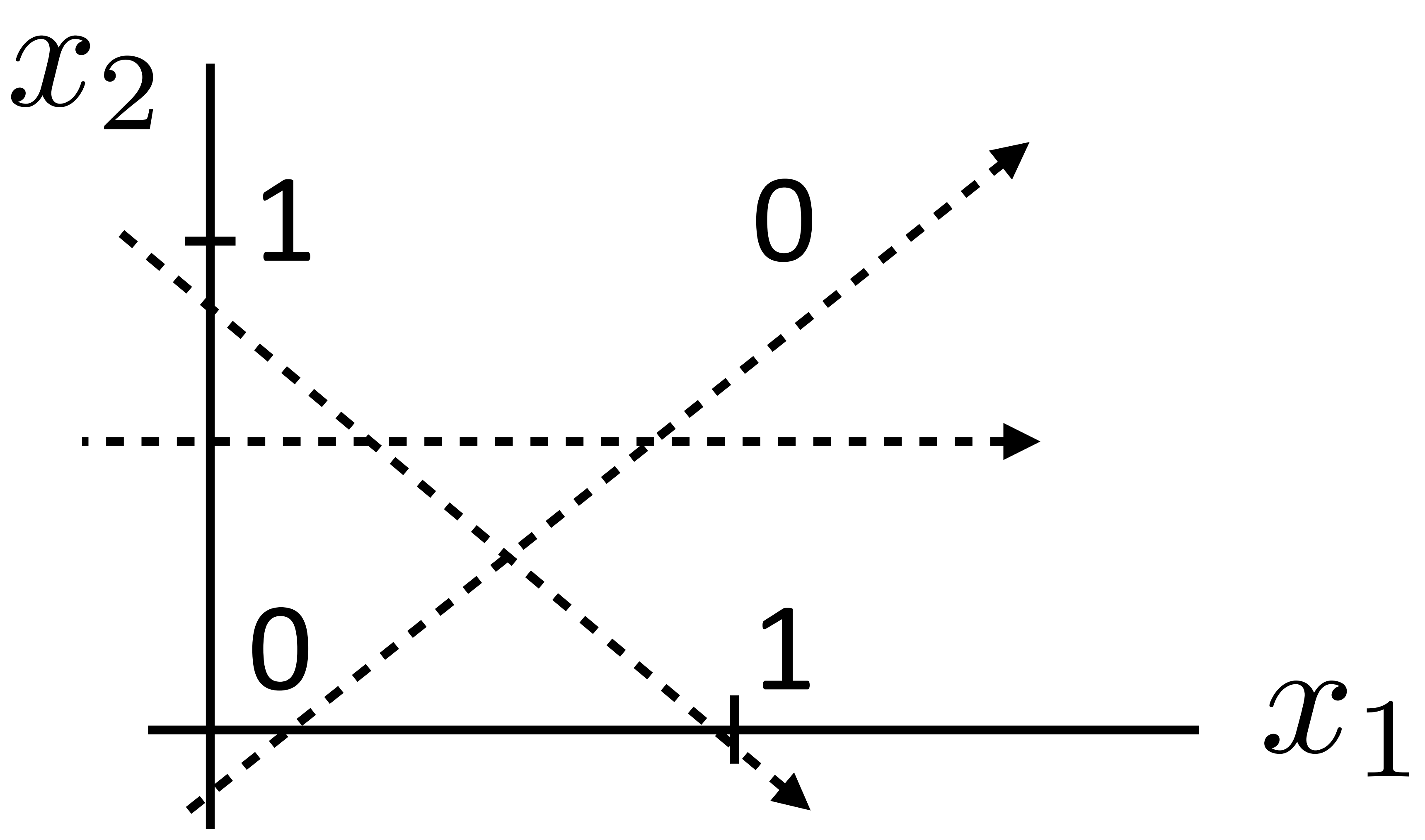
Neural Networks: XOR

- Let's see how we can use neural nets to learn a simple nonlinear function
- Inputs $x_1,\ x_2$
  (generally $\mathbf{x} = (x_1, \ldots, x_m)$)
- Output y
  (generally $\mathbf{y} = (y_1, \ldots, y_n)$)



x_1	x_2	$y = x_1 \text{ XOR } x_2$
0	0	0
0	1	1
1	0	1
1	1	0

Neural Networks: XOR



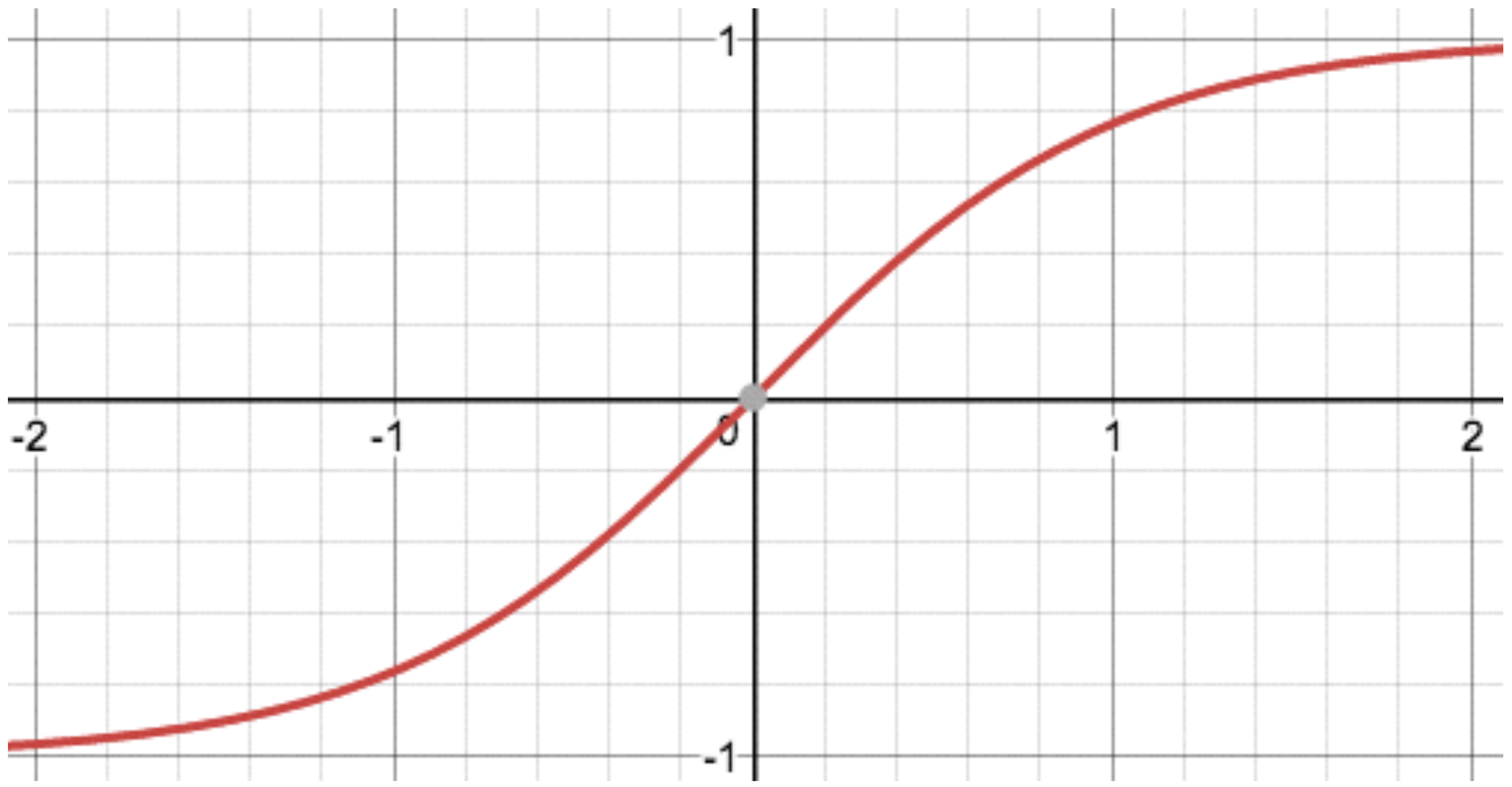
$$y = a_1 x_1 + a_2 x_2$$ X

$$y = a_1 x_1 + a_2 x_2 + a_3 \tanh(x_1 + x_2)$$ ✓

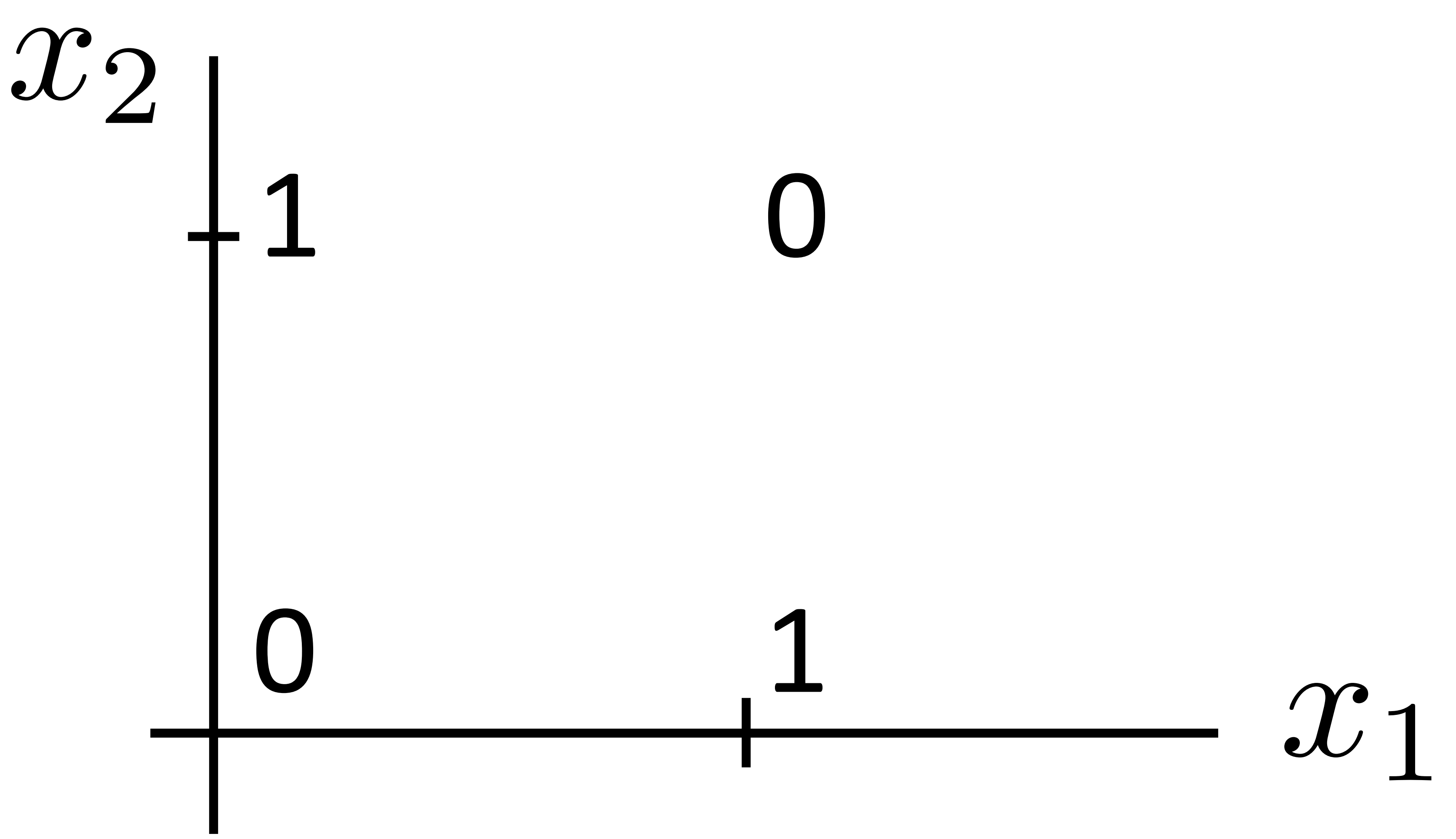
"or"

(looks like action potential in neuron)

x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0



Neural Networks: XOR



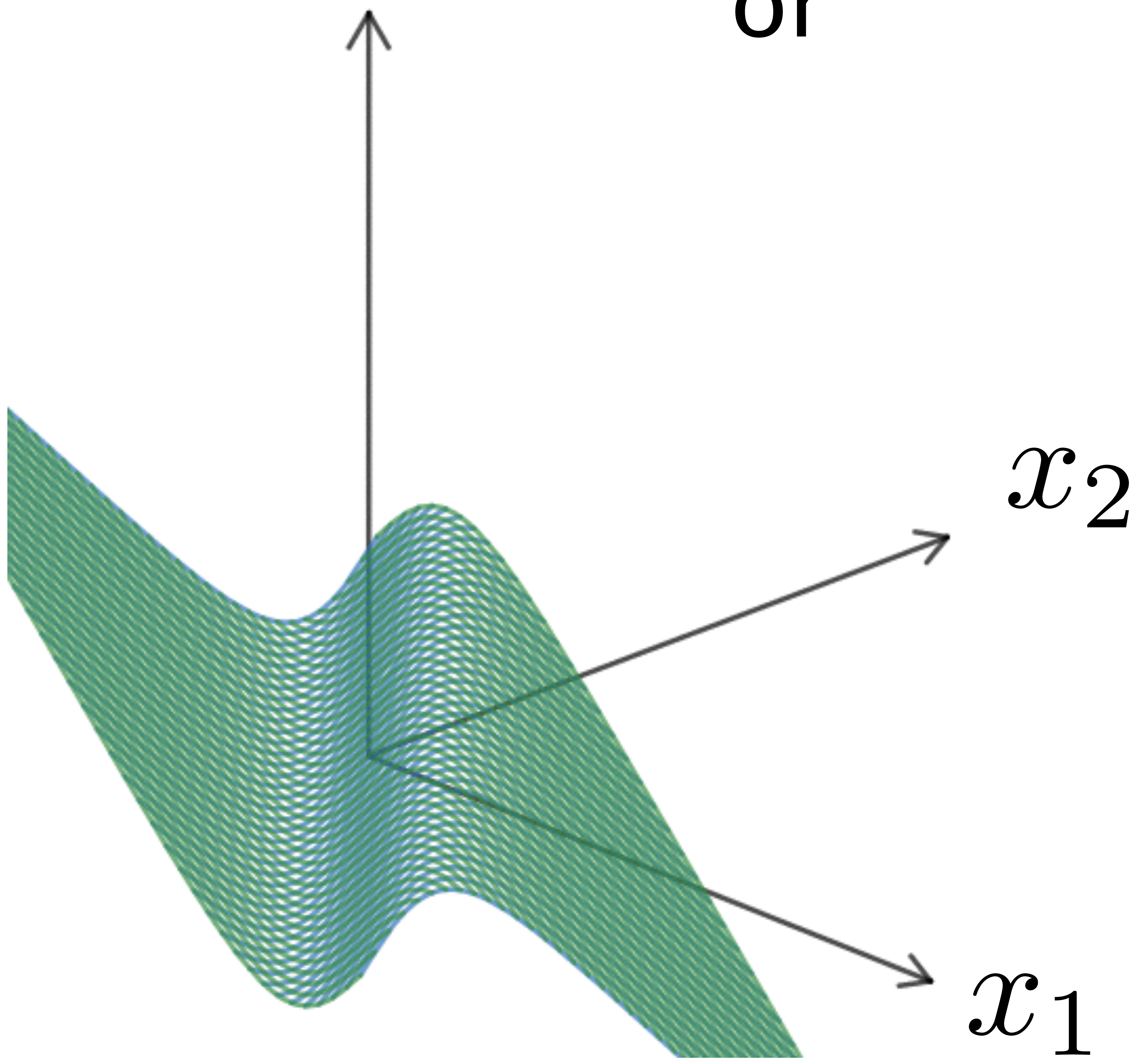
x_1	x_2	x_1 XOR x_2
0	0	0
0	1	1
1	0	1
1	1	0

$$y = a_1 x_1 + a_2 x_2$$

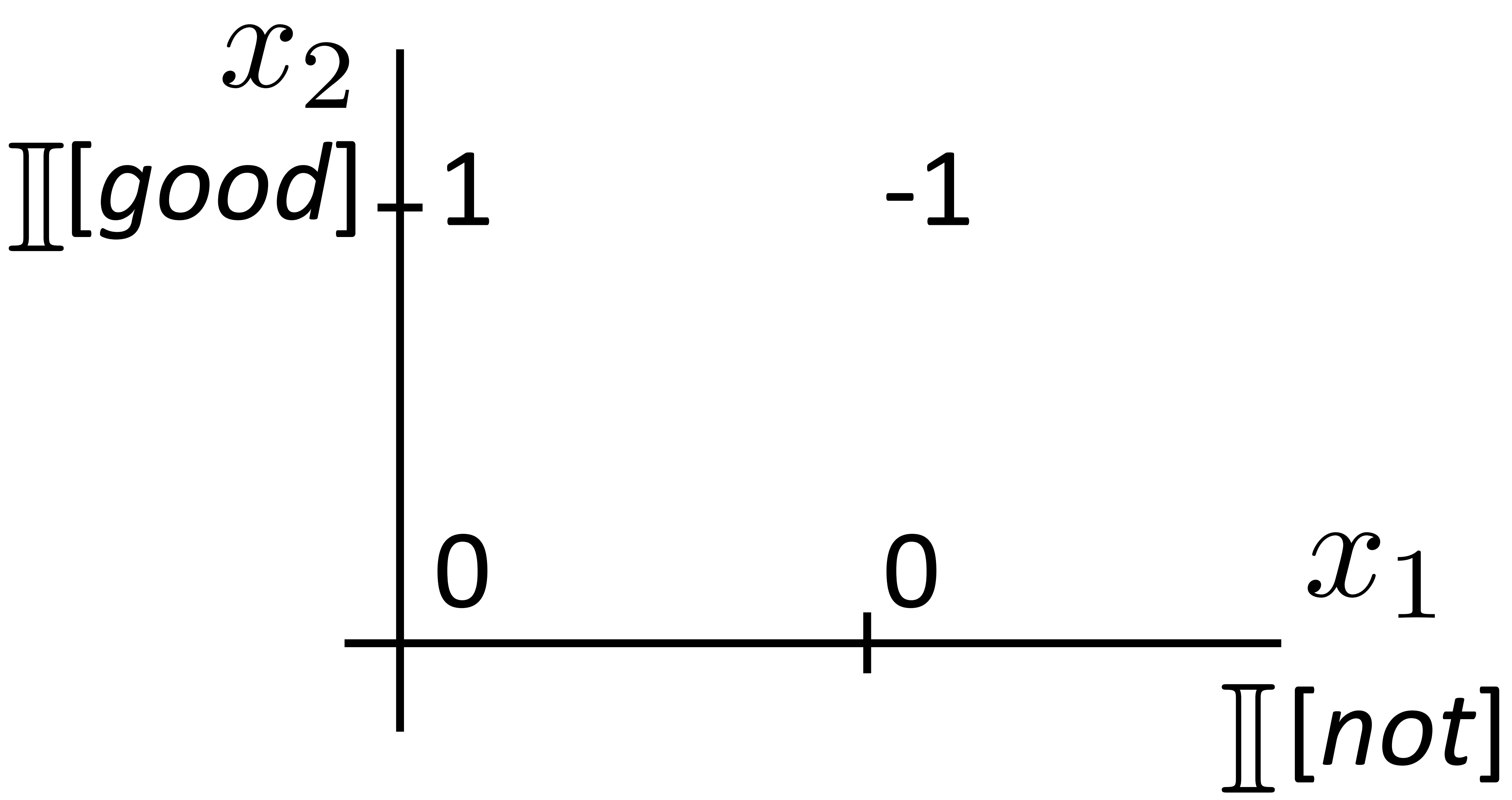
$$y = a_1 x_1 + a_2 x_2 + a_3 \tanh(x_1 + x_2)$$

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

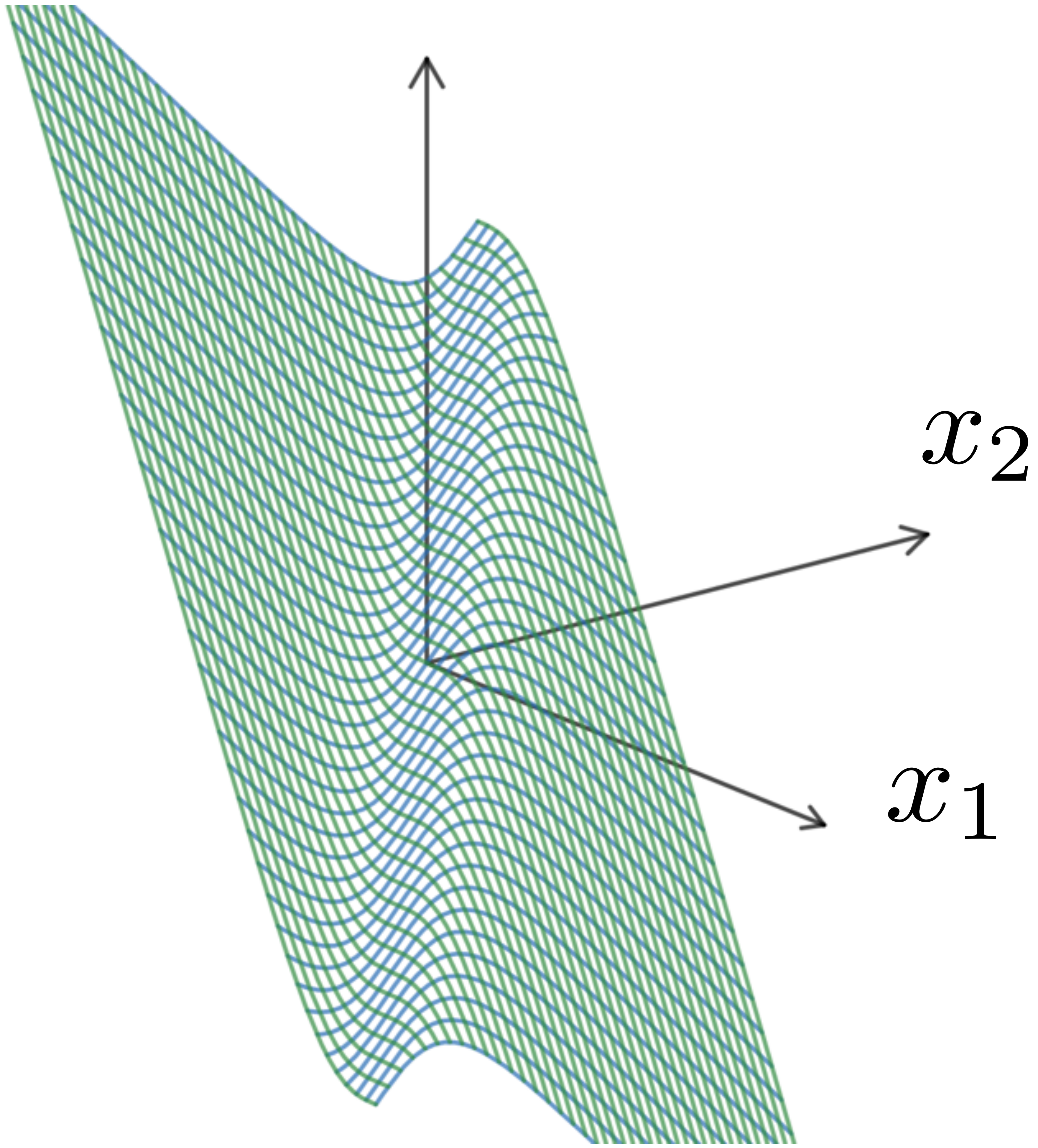
"or"



Neural Networks: XOR



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



Neural Networks

(Linear model: $y = \mathbf{w} \cdot \mathbf{x} + b$)

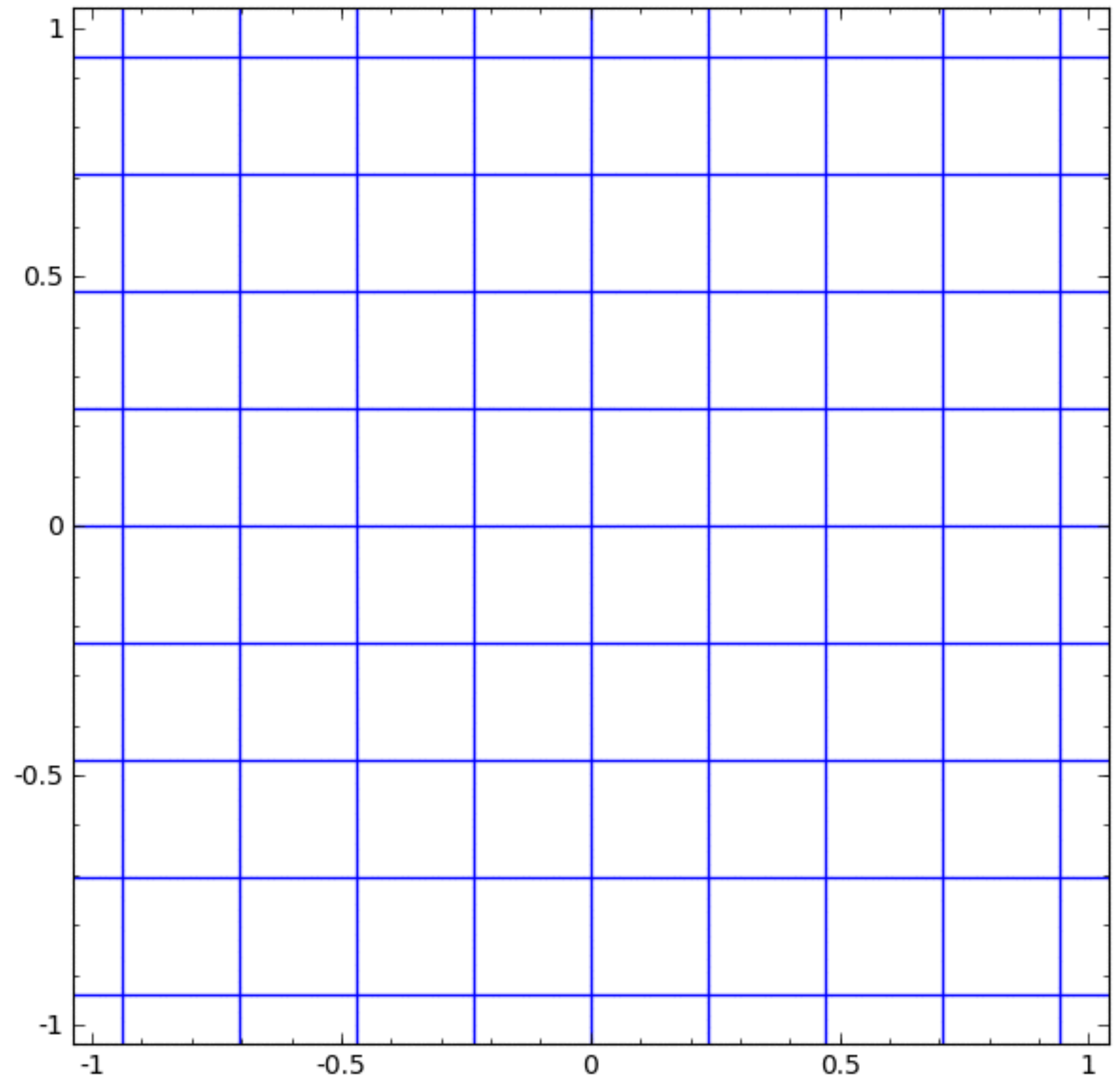
$$y = g(\mathbf{w} \cdot \mathbf{x} + b)$$

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

Nonlinear transformation

Warp space

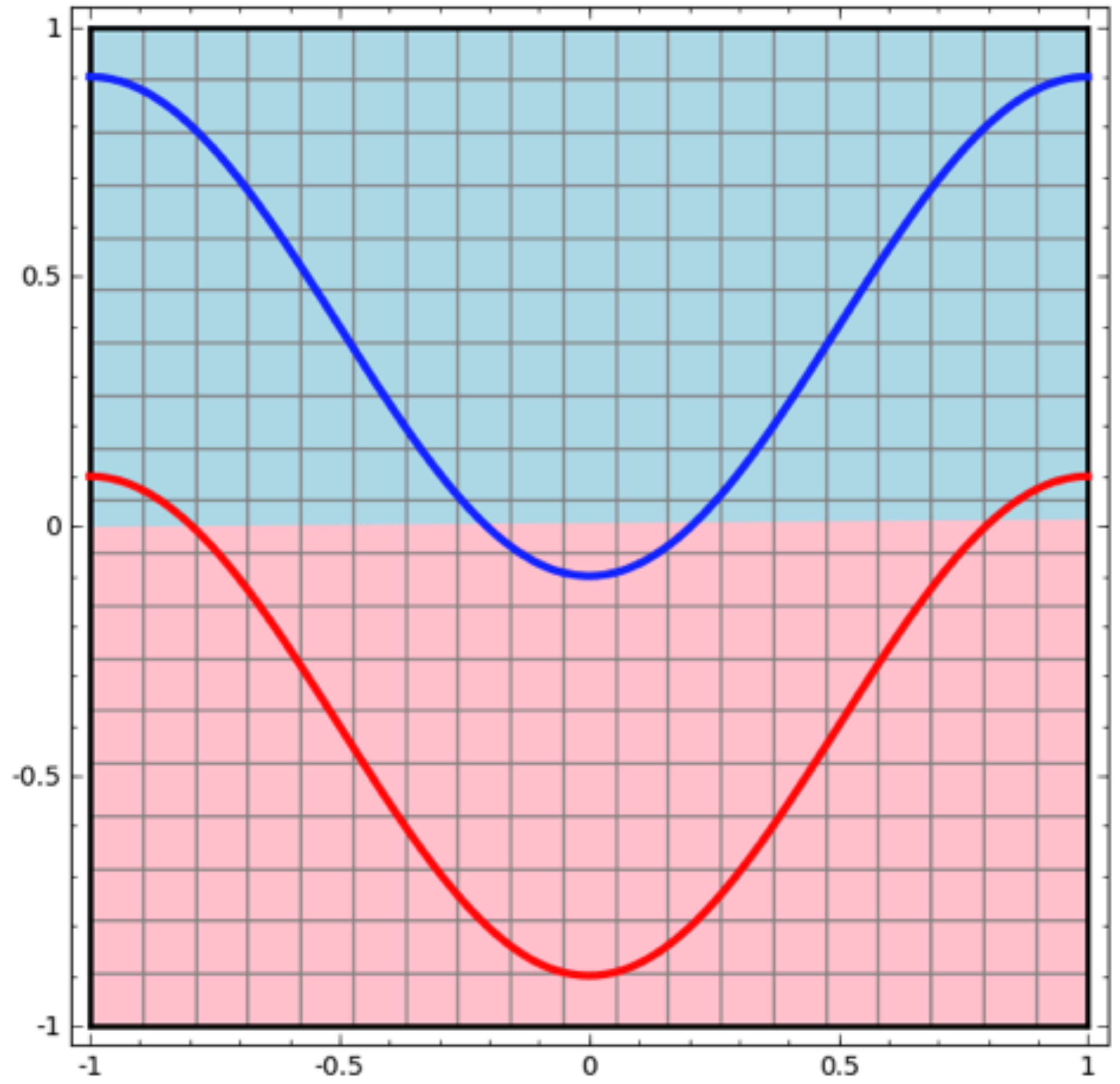
Shift



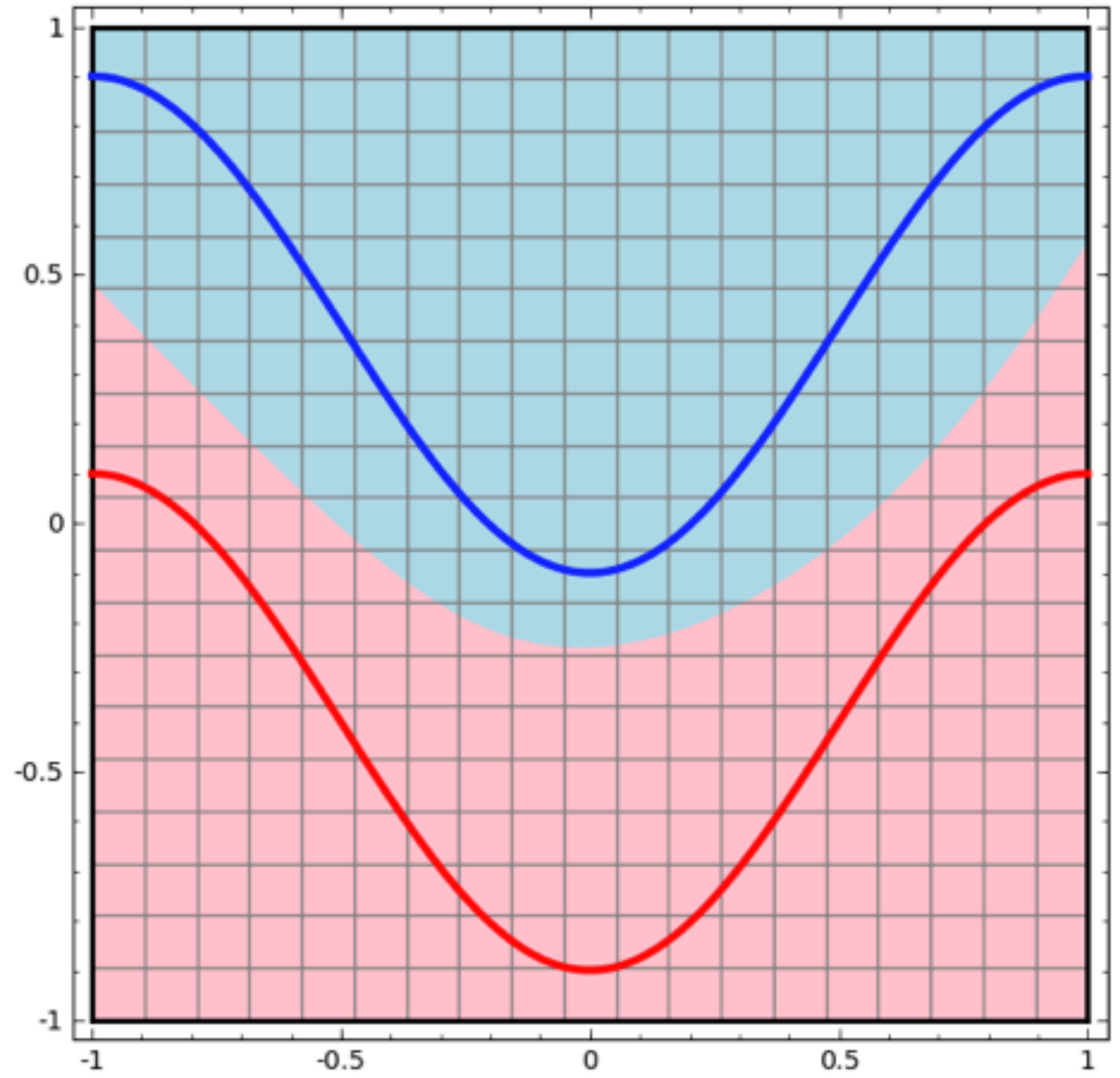
Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Neural Networks

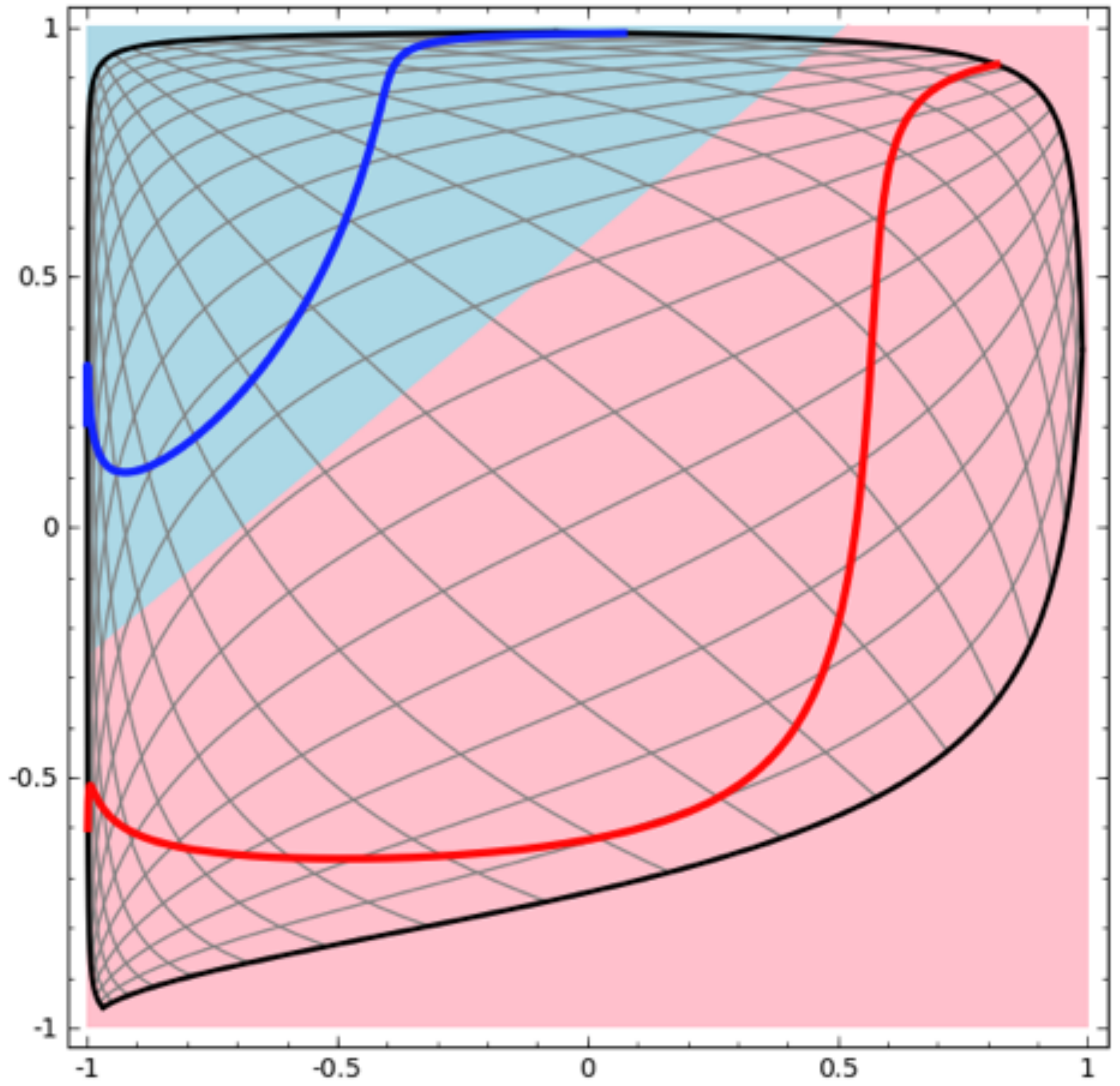
Linear classifier



Neural network



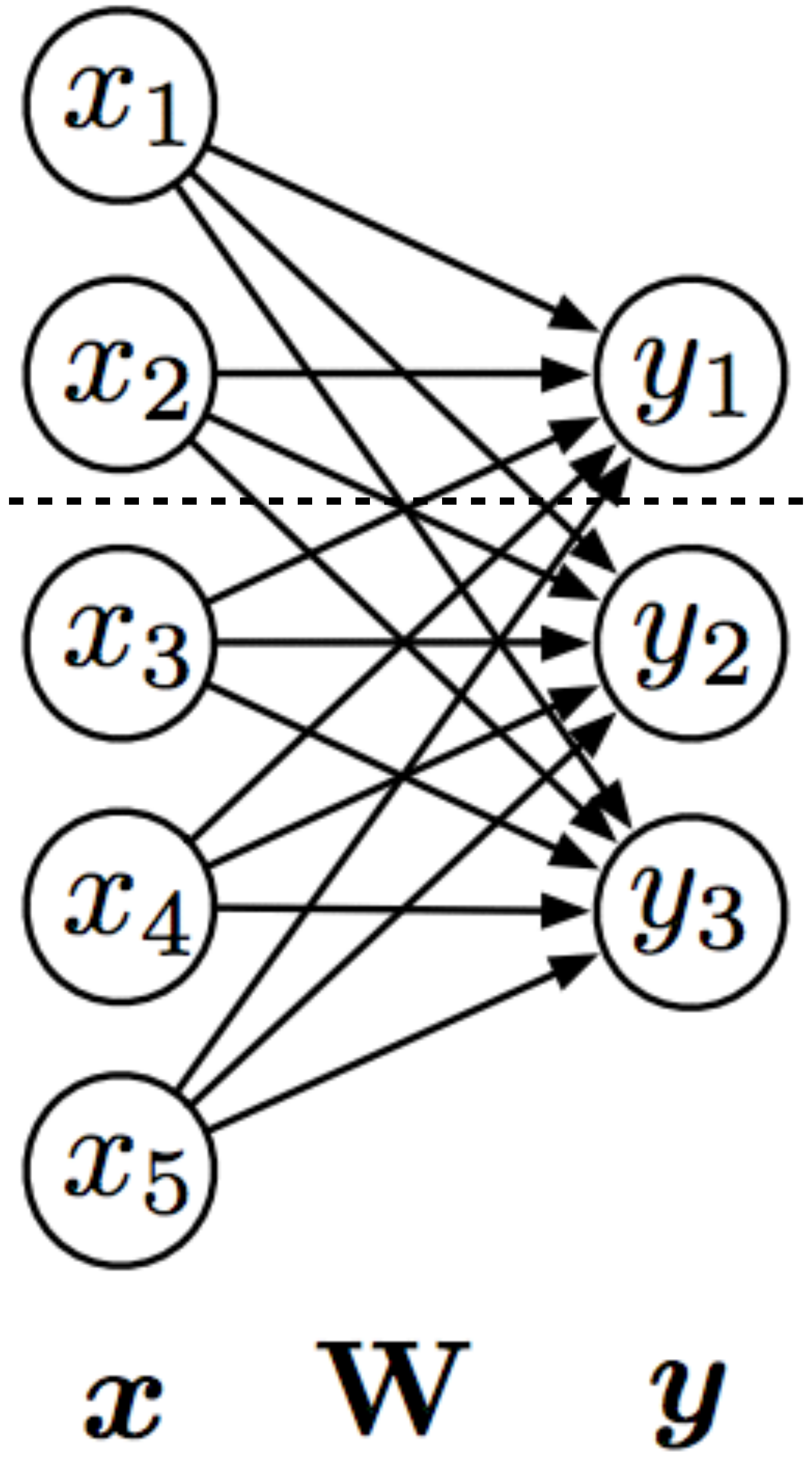
…possible because we transformed the space!



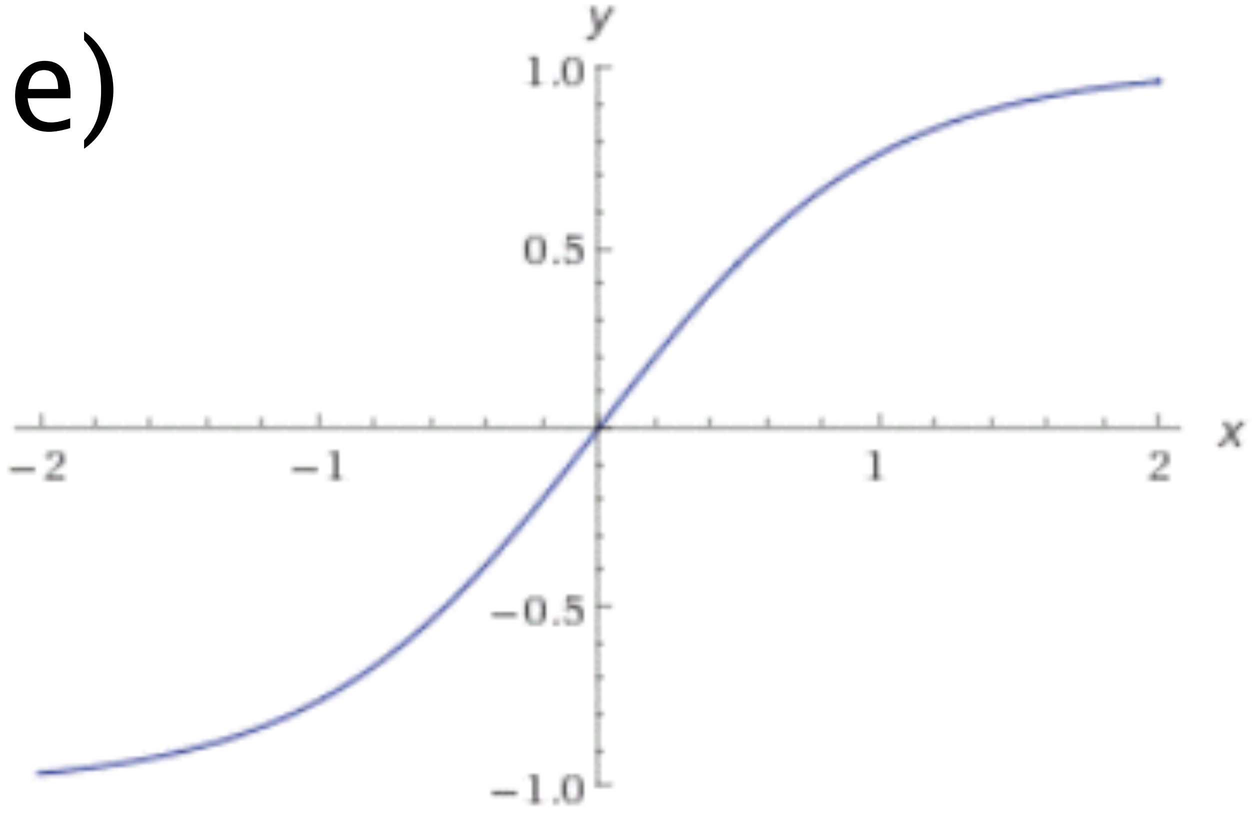
Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

Deep Neural Networks

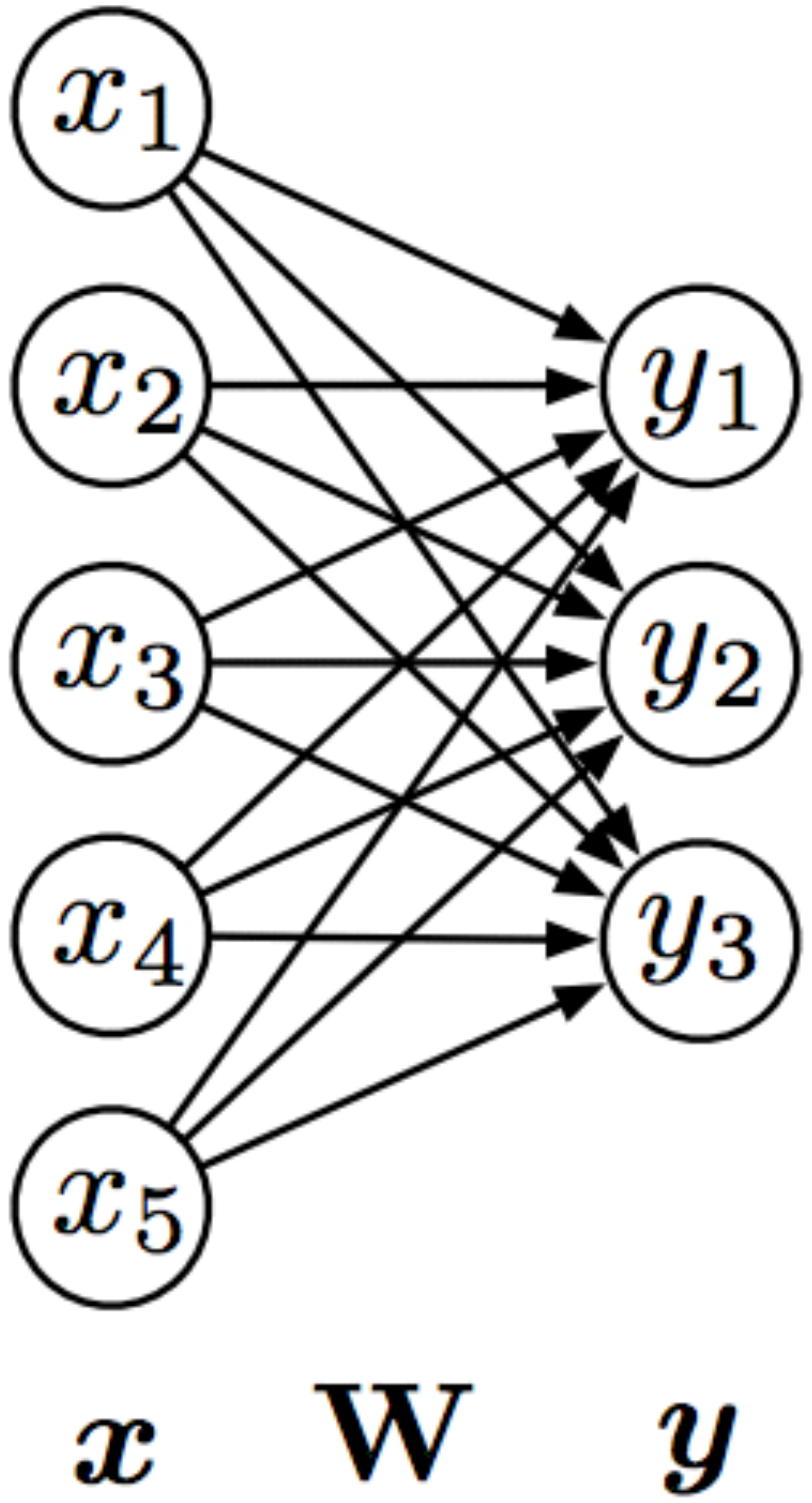
(this was our neural net from the XOR example)



$$y_1 = g(\mathbf{w}_1 \cdot \mathbf{x} + b_1)$$

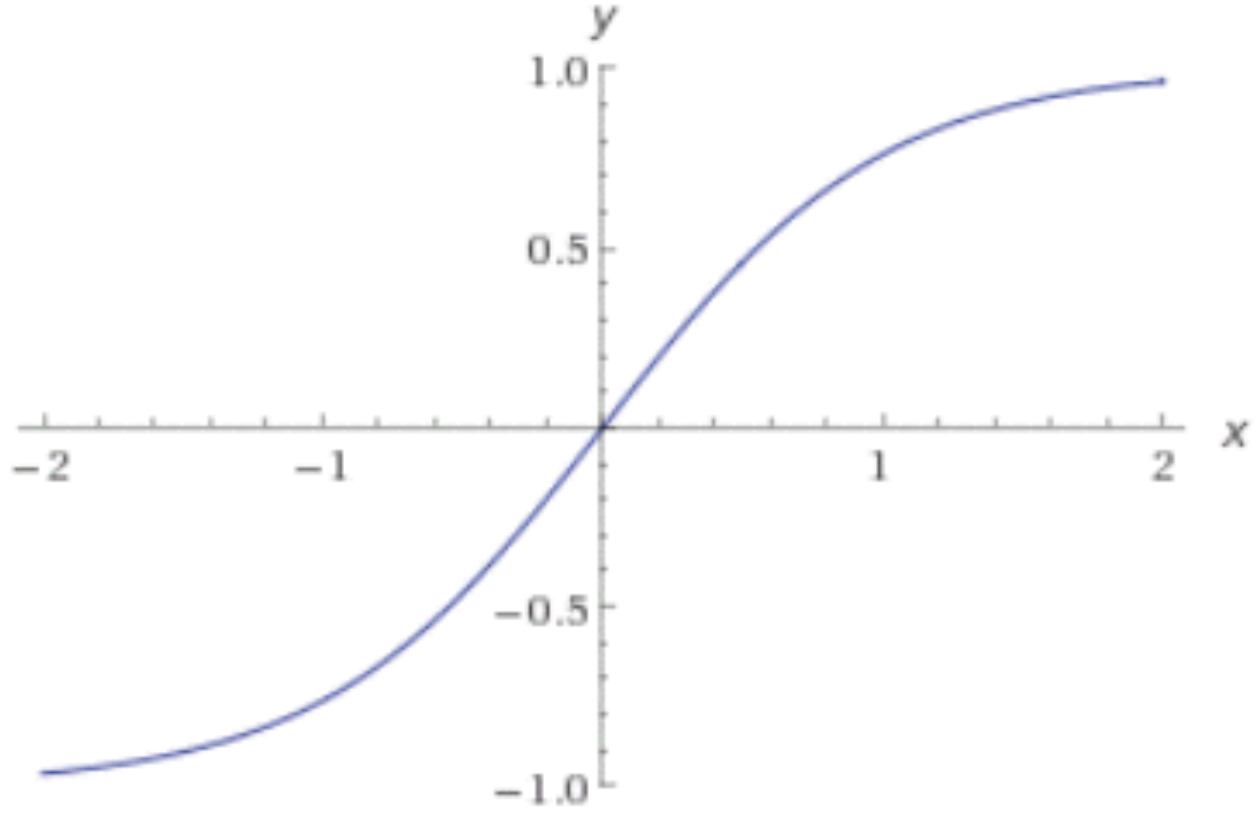


Deep Neural Networks



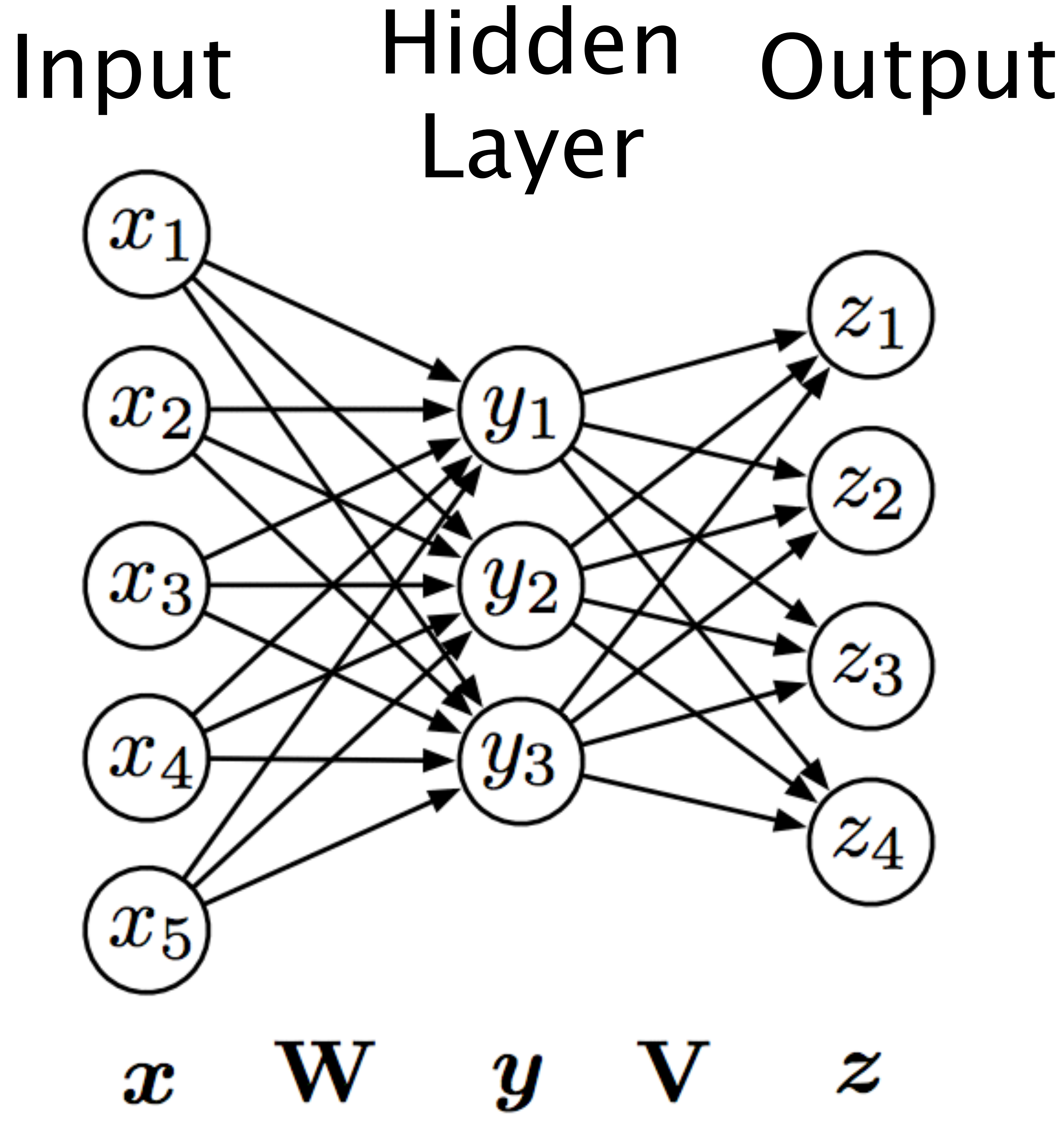
$$y_1 = g(\mathbf{w}_1 \cdot \mathbf{x} + b_1)$$

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



Adopted from Chris Dyer

Deep Neural Networks



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

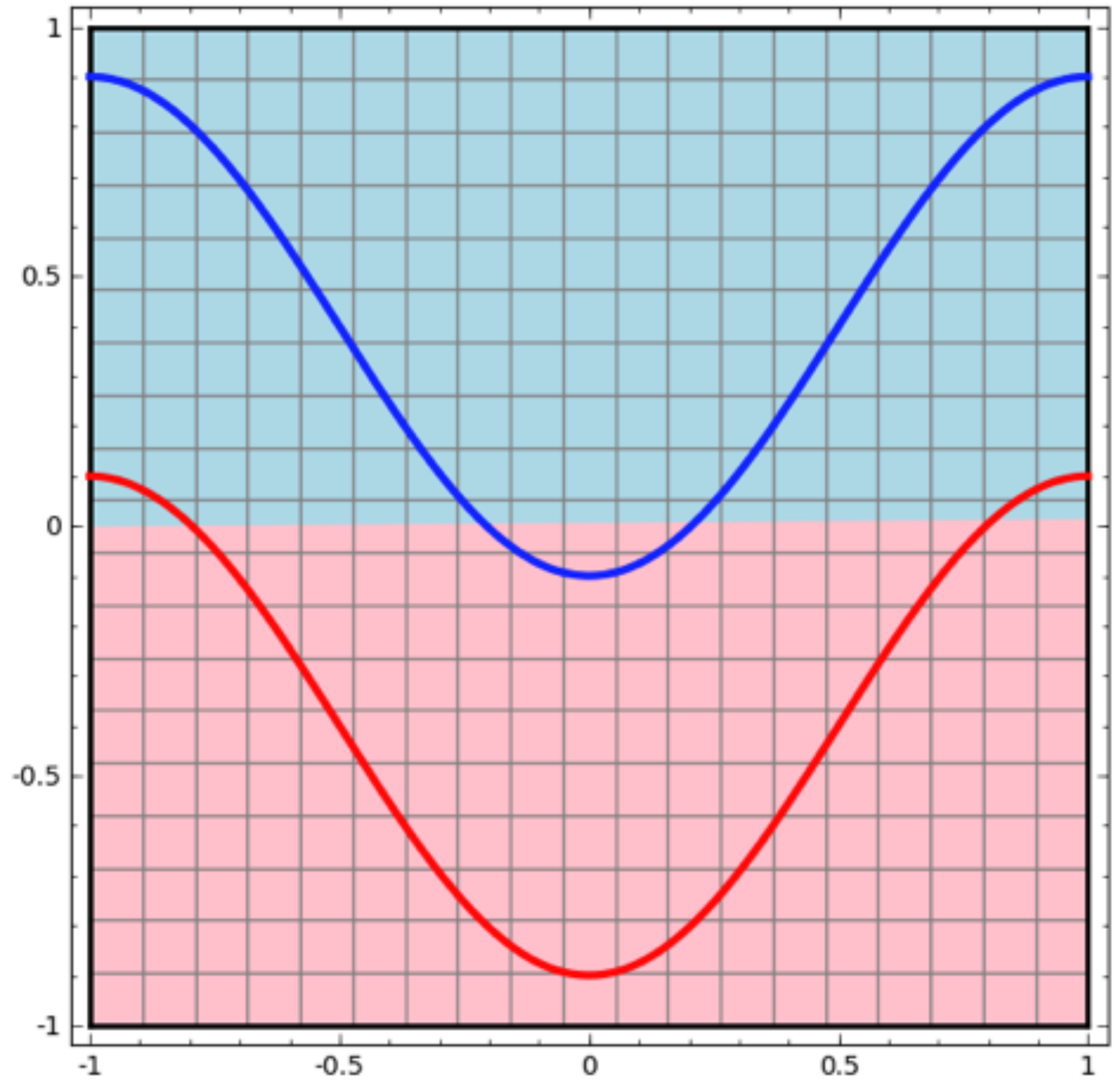
$$\mathbf{z} = g(\mathbf{V}\underbrace{g(\mathbf{W}\mathbf{x} + \mathbf{b})}_{\text{output of first layer}} + \mathbf{c})$$

$$\mathbf{z} = g(\mathbf{V}\mathbf{y} + \mathbf{c})$$

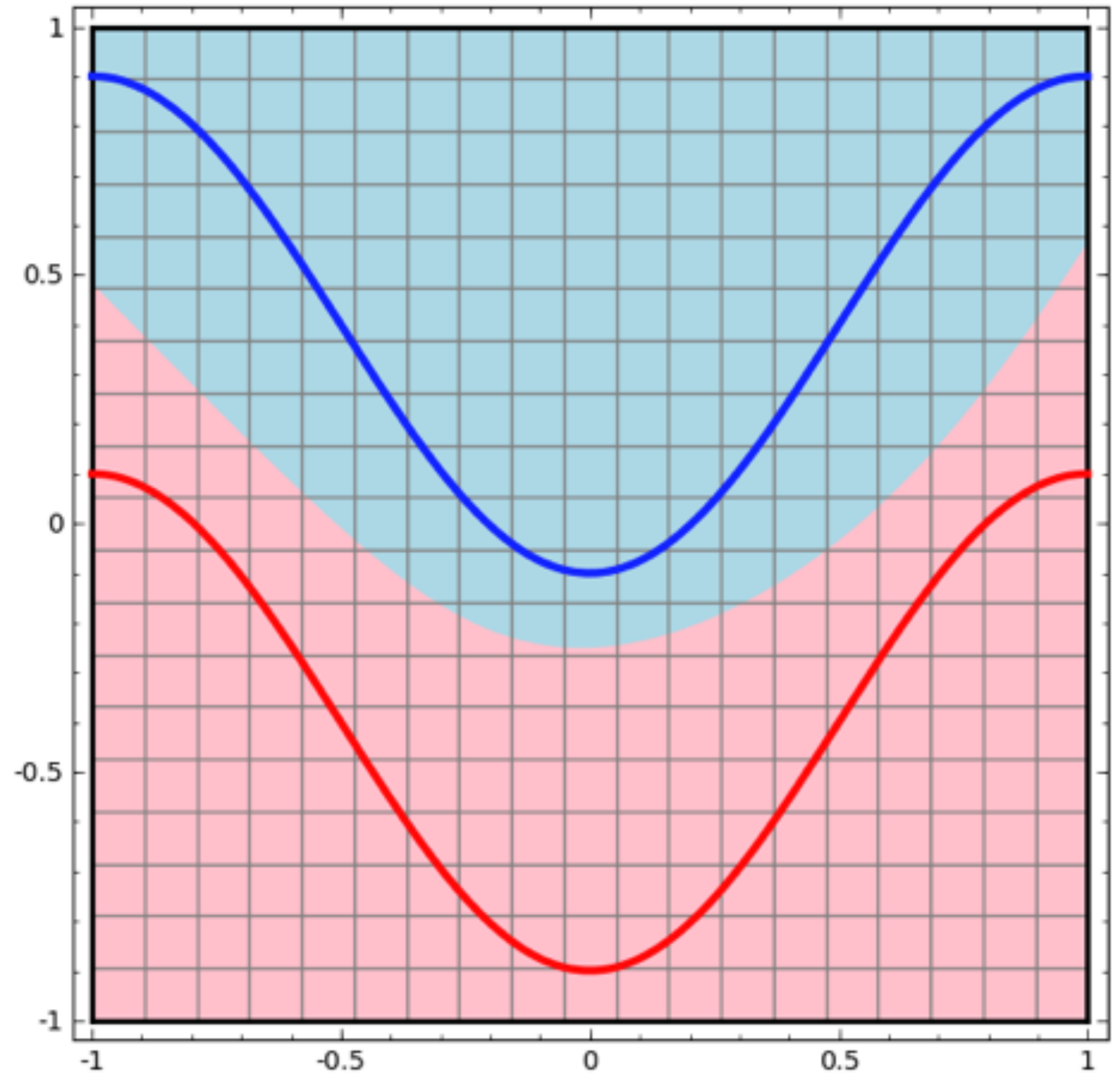
Adopted from Chris Dyer

Neural Networks

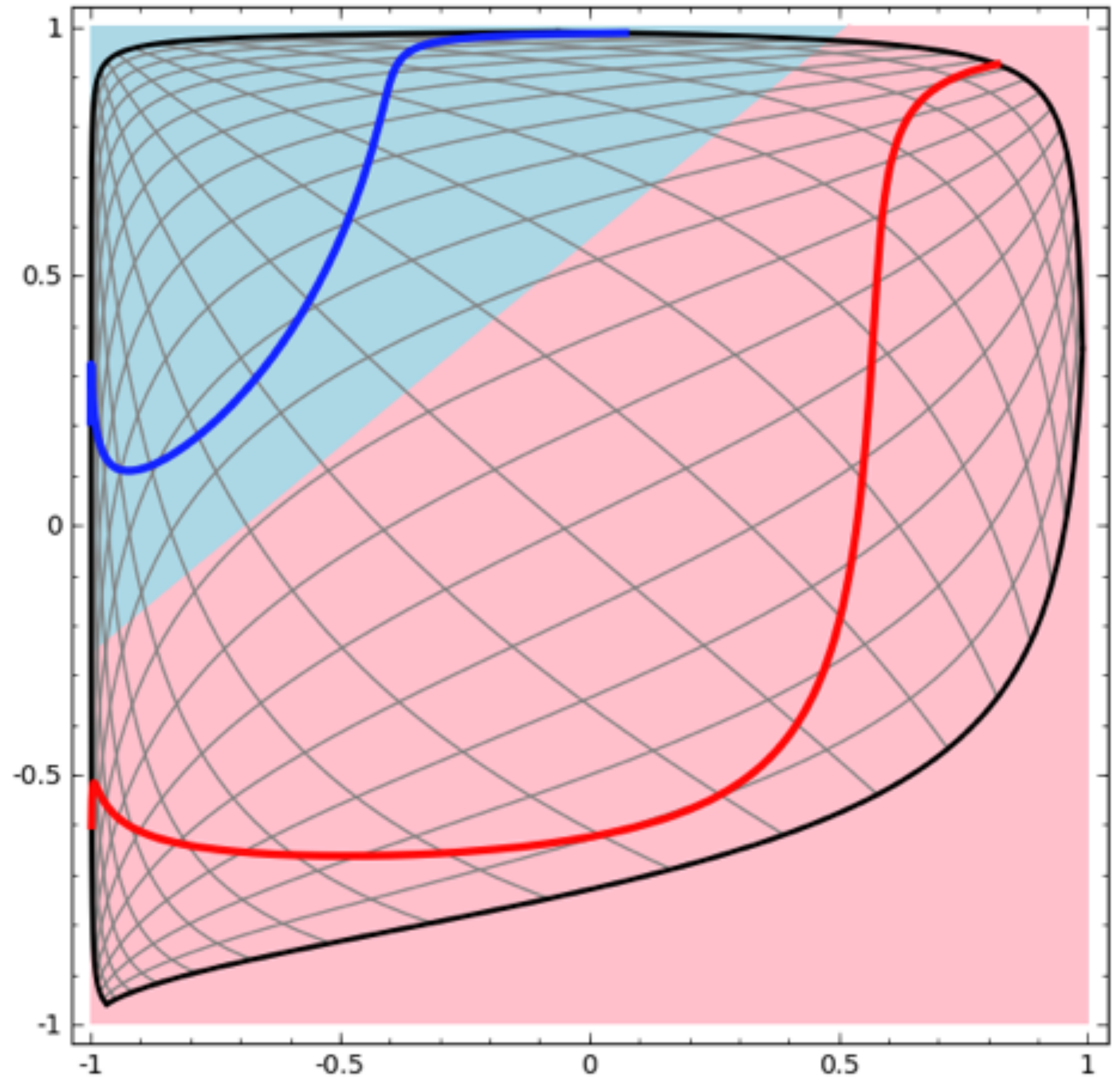
Linear classifier



Neural network



…possible because we transformed the space!

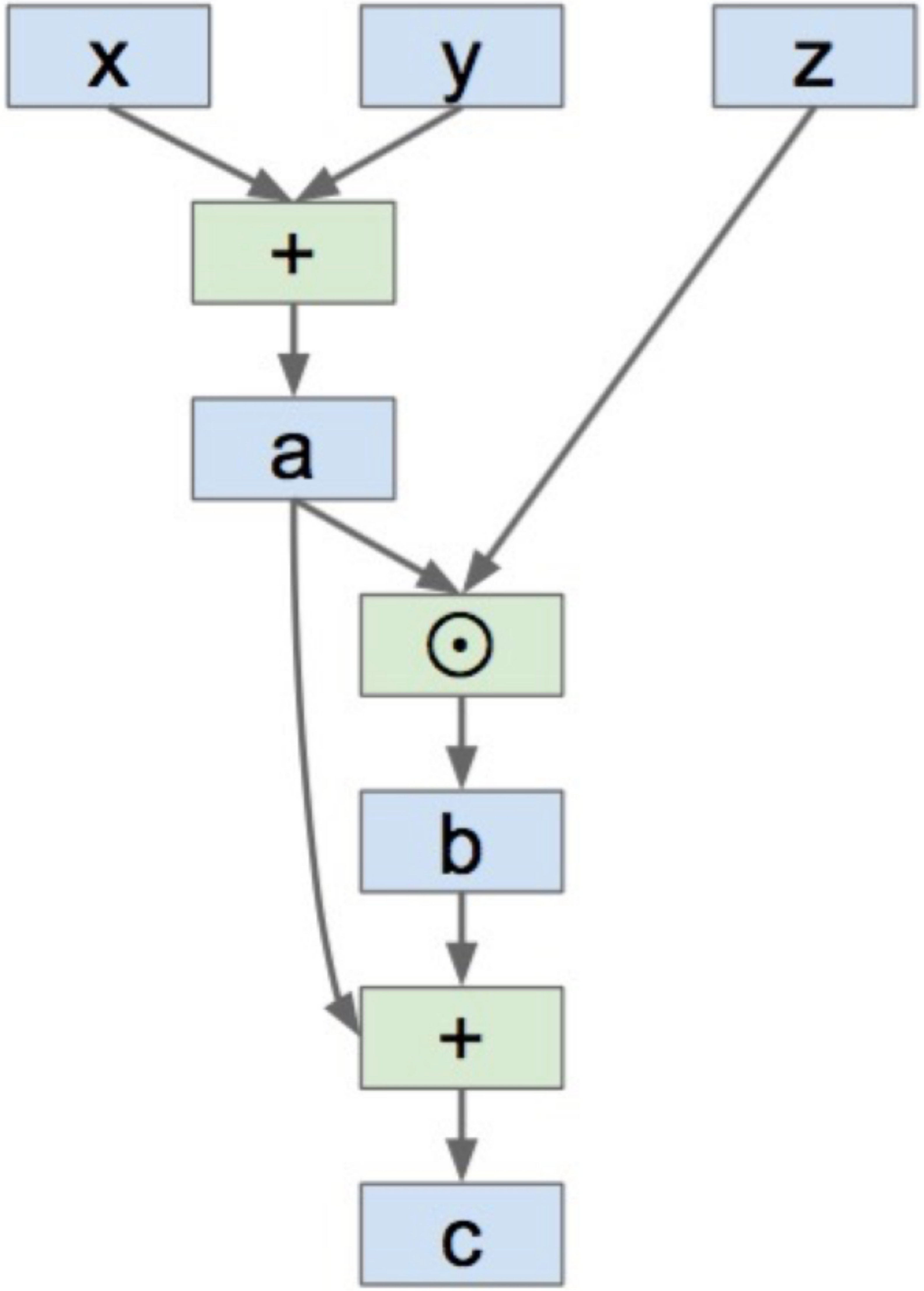


Taken from http://colah.github.io/posts/2014-03-NN-Manifolds-Topology/

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

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)

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

Word Vector Tools

- Word2Vec: https://radimrehurek.com/gensim/models/word2vec.html

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

  - Python code, actively maintained

- GLoVe: http://nlp.stanford.edu/projects/glove/
  - Word vectors trained on very large corpora

Convolutional Networks

- CNNs for sentence class.: https://github.com/yoonkim/CNN_sentence
  - Based on tutorial from: http://deeplearning.net/tutorial/lenet.html
  - Python code
  - Trains very quickly

Takeaways

- Neural networks have several advantages for NLP:
  - We can use *simpler nonlinear functions* instead of more complex linear functions
  - We can take advantage of word similarity
  - We can build models that are both position-dependent (feedforward neural networks) and position-independent (convolutional networks)
- NNs have natural applications to many problems
- While conventional linear models often still do well, neural nets are increasingly the state-of-the-art for many tasks