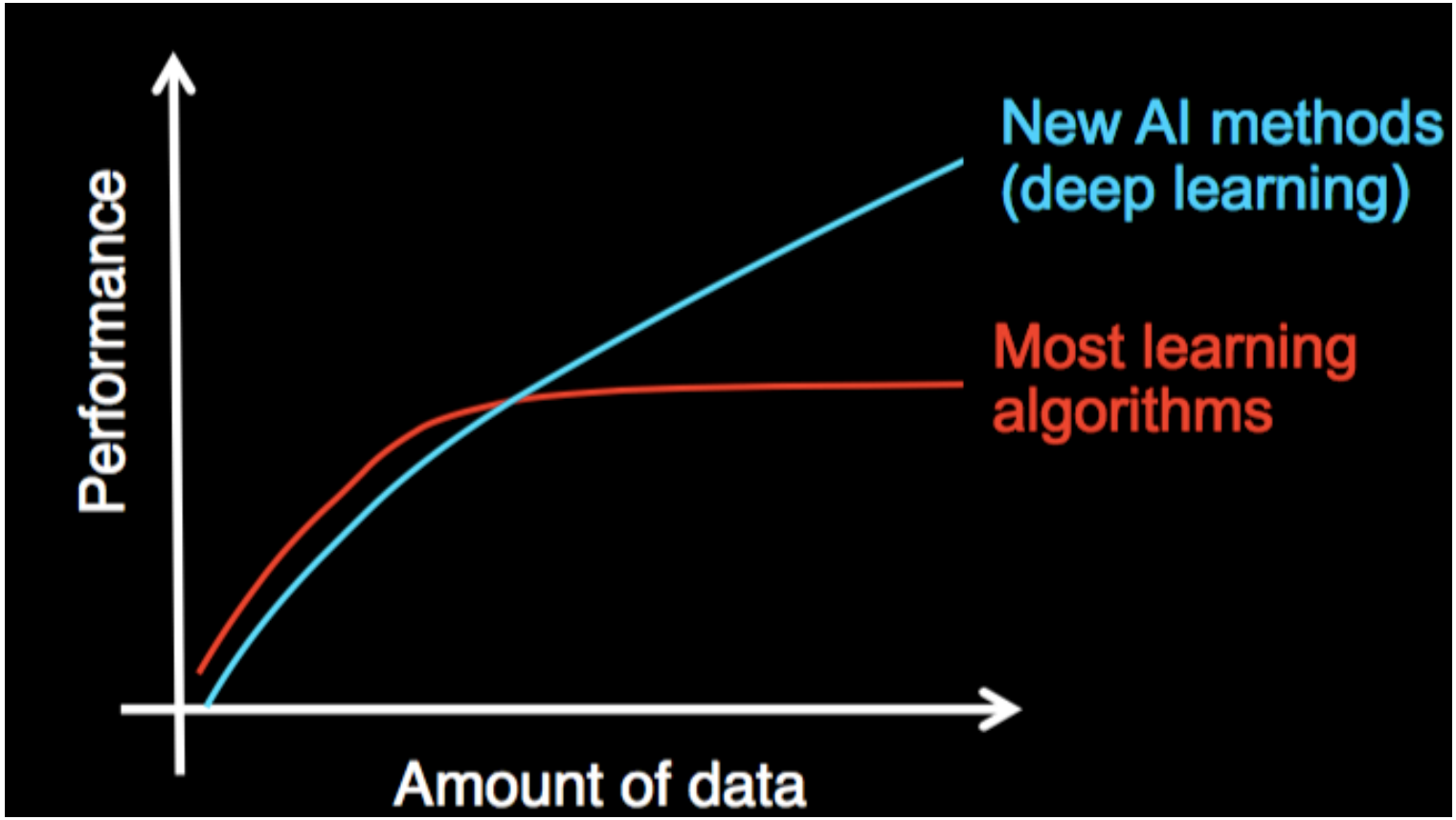
BBM 495
INTRODUCTION TO DEEP LEARNING

LECTURER: BURCU CAN

2019-2020 SPRING

What is Deep Learning?

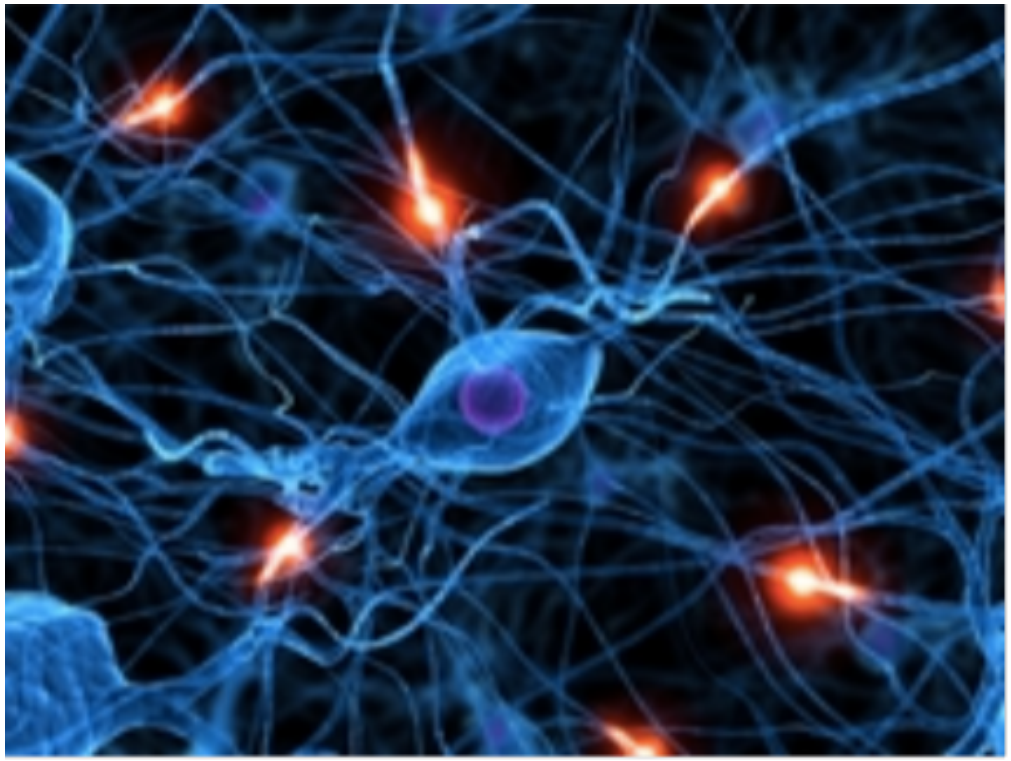
Data and Machine Learning



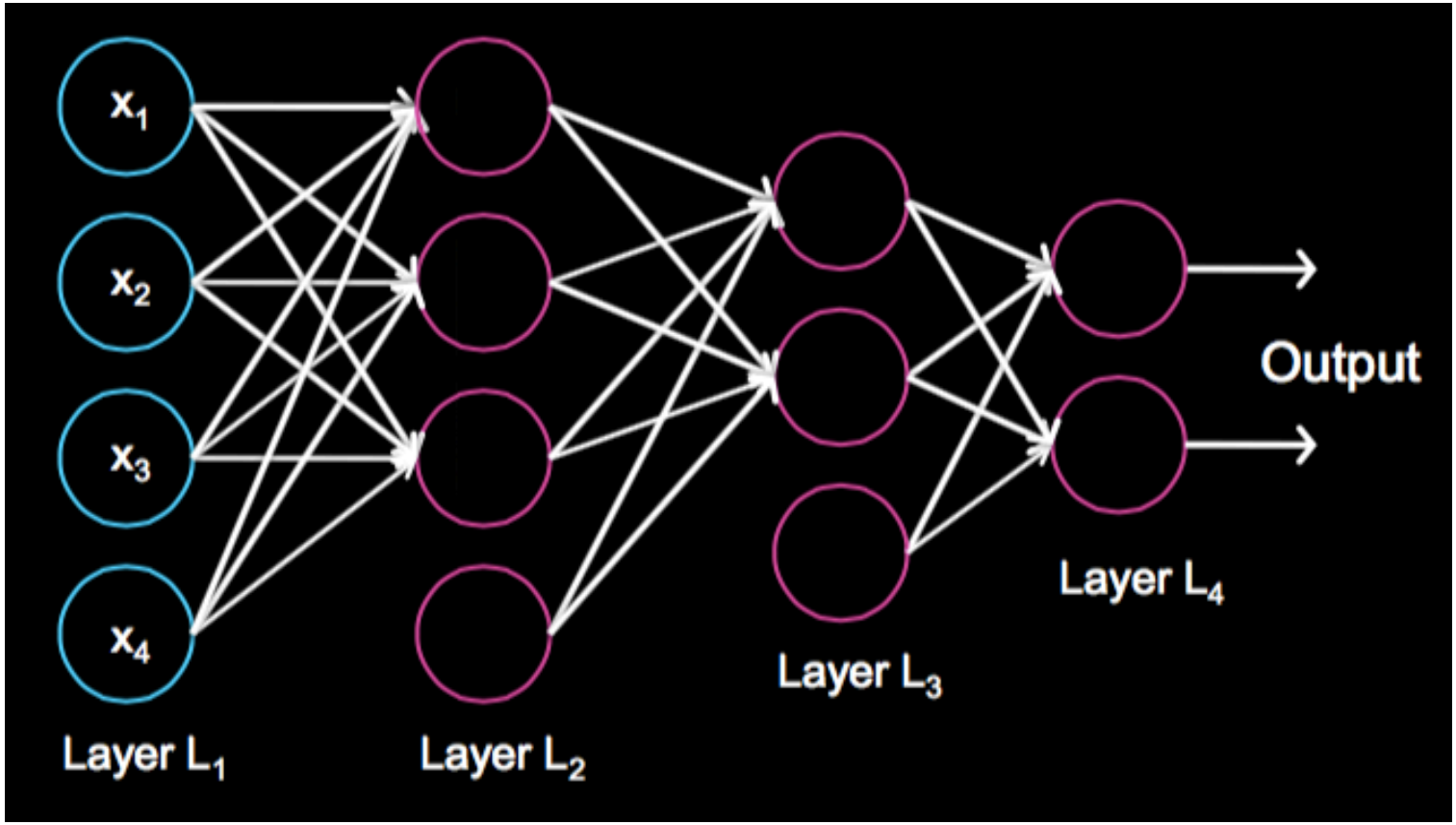
The idea

- **Build learning algorithms that mimic the brain!**

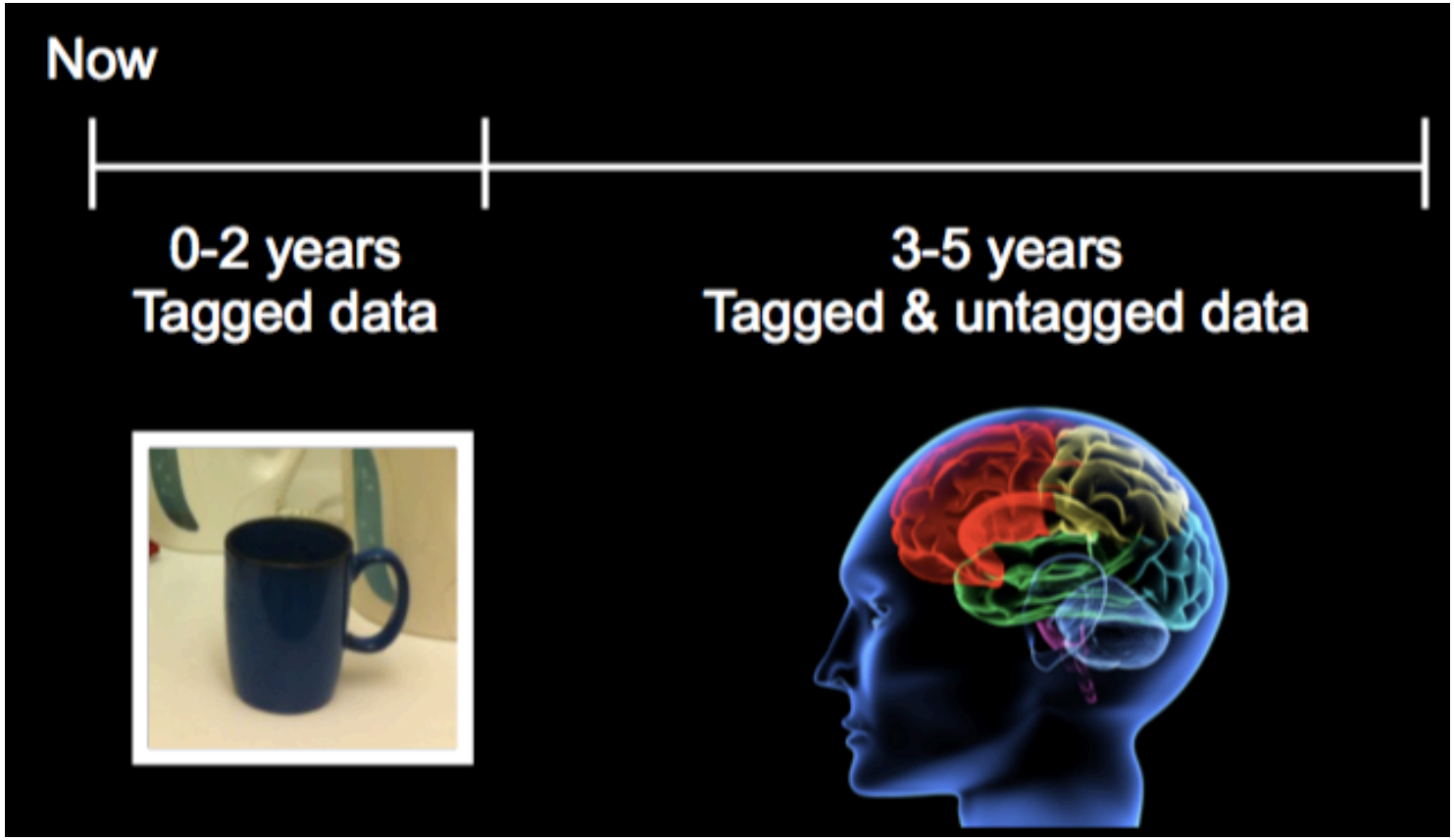
Neurons in the brain



Neural Networks (Deep Learning)



Deep Learning Trends



Learning from Tagged Data

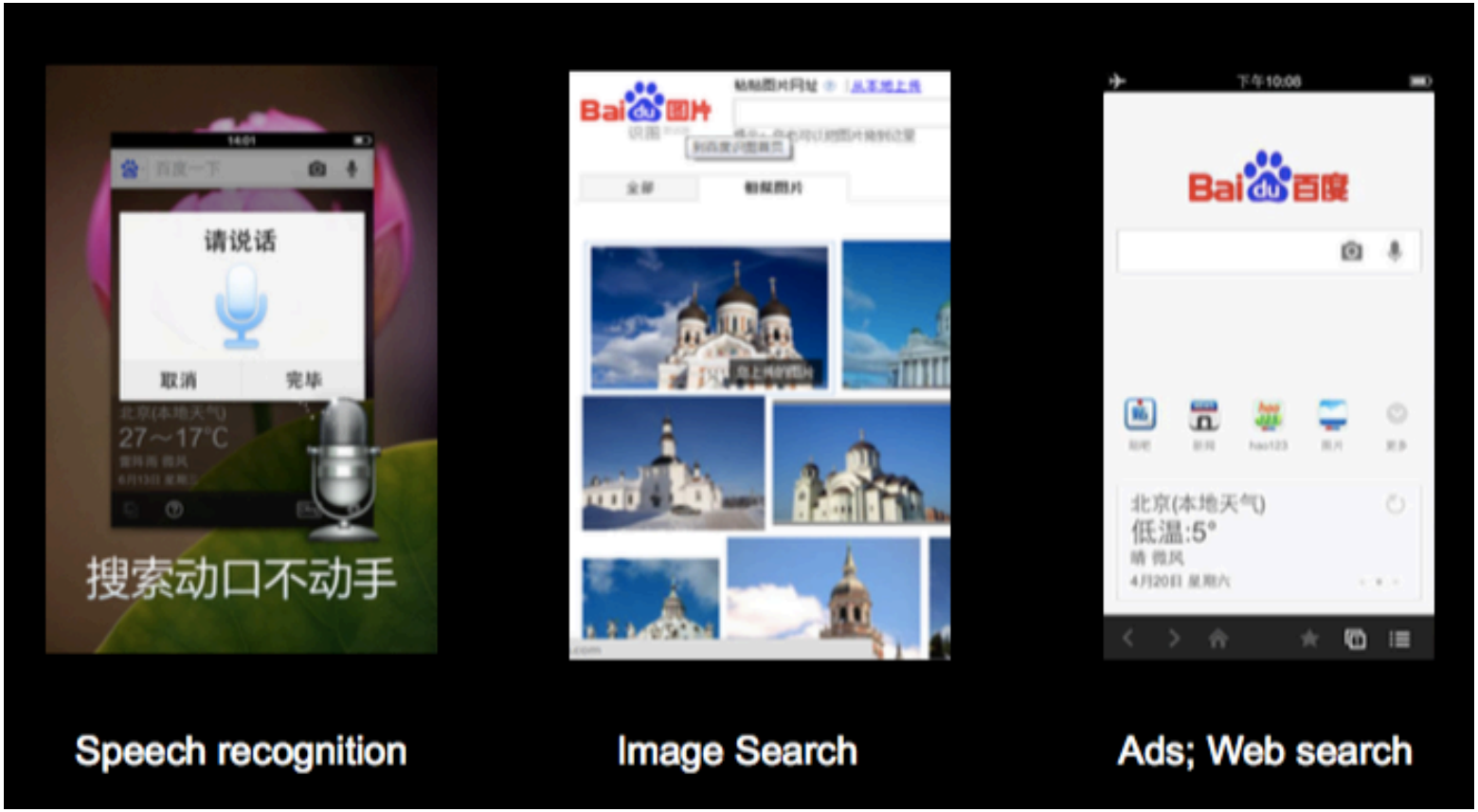
Learning from Tagged Data

In 1917, Einstein applied the general theory of relativity to model the large-scale structure of the universe. He was visiting the United States when Adolf Hitler came to power in 1933 and did not go back to Germany, where he had been a professor at the Berlin Academy of Sciences. He settled in the U.S., becoming an American citizen in 1940. On the eve of World War II, he endorsed a letter to President Franklin D. Roosevelt alerting him to the potential development of "extremely powerful bombs of a new type" and recommending that the U.S. begin similar research. This eventually led to what would become the Manhattan Project. Einstein supported defending the Allied forces, but largely denounced using the new discovery of nuclear fission as a weapon. Later, with the British philosopher Bertrand Russell, Einstein signed the Russell-Einstein Manifesto, which highlighted the danger of nuclear weapons. Einstein was affiliated with the Institute for Advanced Study in Princeton, New Jersey, until his death in 1955.

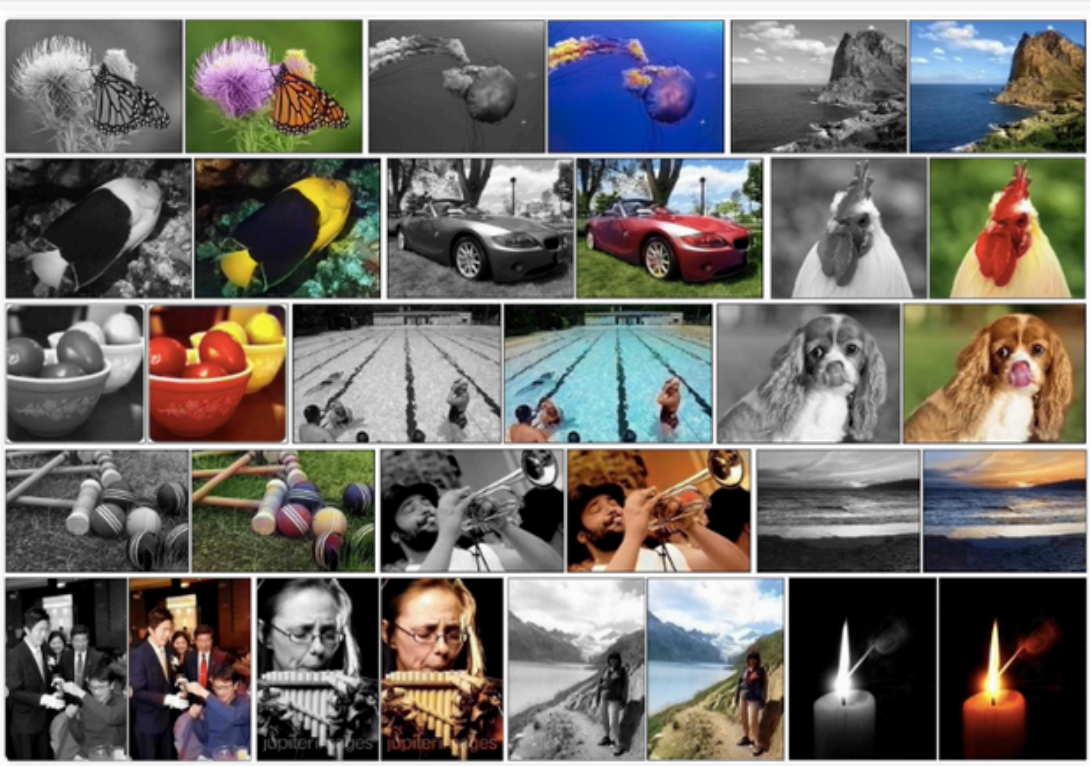
Tag colours:

LOCATION TIME PERSON ORGANIZATION MONEY PERCENT DATE

Deep Learning Applications



Colorization of Black and White Images



Colorization of Black and White Photographs
Image taken from Richard Zhang, Phillip Isola and Alexei A. Efros.

Larsson et al. Learning Representations for Automatic Colorization, 2017

Automatic Machine Translation

Instant Visual Translation

Example of instant visual translation, taken from the Google Blog.

Sutskever et al. Sequence to Sequence Learning with Neural Networks, 2014

Deep Learning for Recommender Systems

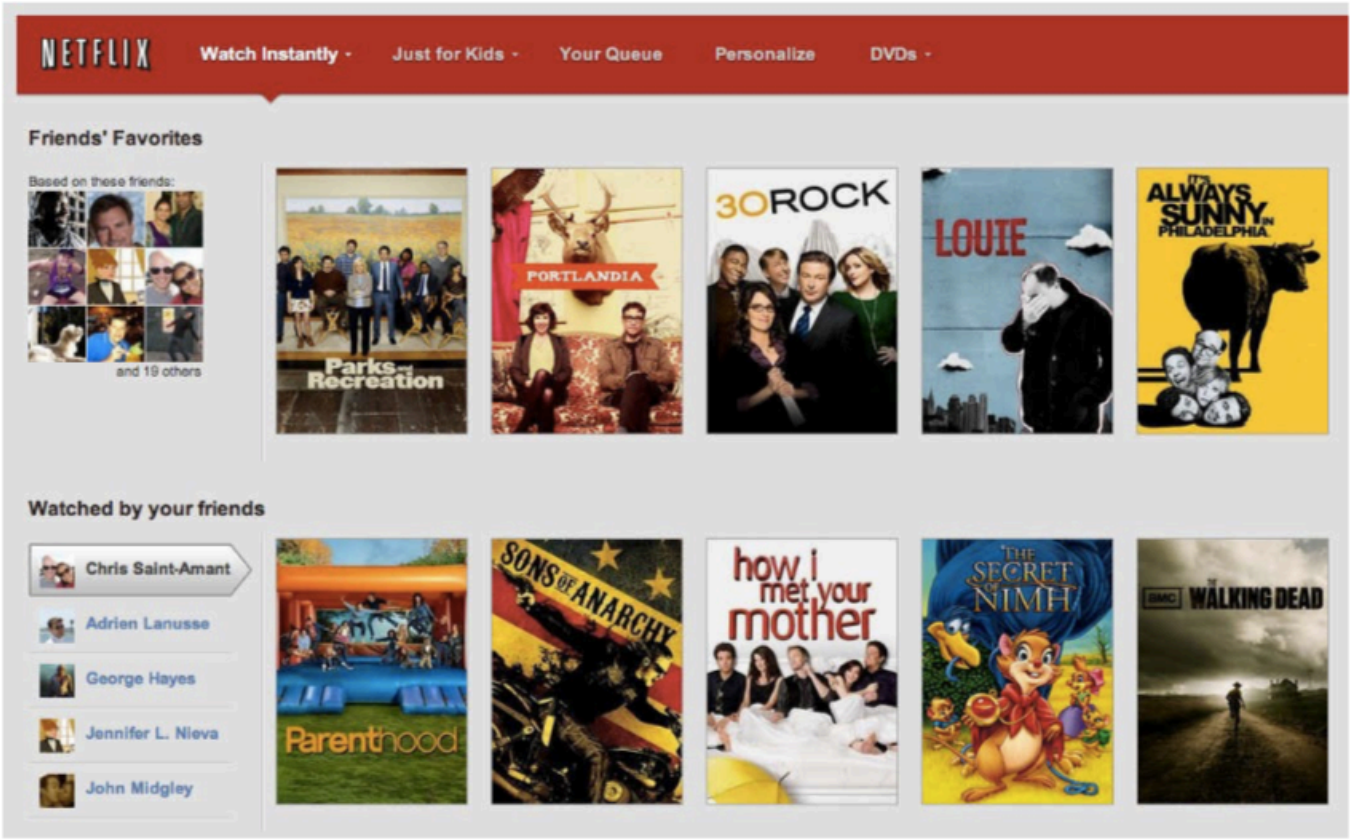


Image courtesy of Netflix

Recommender System

Total price: $40.05

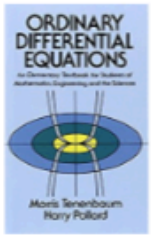
Add all three to Cart

Add all three to List

- **This item:** Numerical Methods for Scientists and Engineers (Dover Books on Mathematics) by R. W. Hamming Paperback $14.31
- Partial Differential Equations for Scientists and Engineers (Dover Books on Mathematics) by Stanley J. Farlow Paperback $10.26
- Ordinary Differential Equations (Dover Books on Mathematics) by Morris Tenenbaum Paperback $15.48

Customers Who Bought This Item Also Bought

Partial Differential Equations for Scientists and Engineers (Dover...
› Stanley J. Farlow
132
Paperback
$10.26 Prime



Ordinary Differential Equations (Dover Books on Mathematics)
› Morris Tenenbaum
115
Paperback
$15.48 Prime

Fourier Series (Dover Books on Mathematics)
Georgi P. Tolstov
61
#1 Best Seller in Functional Analysis...
Paperback
$3.99 Prime

An Introduction to Information Theory: Symbols, Signals and...
› John R. Pierce
90
Paperback
$3.99 Prime

Linear Algebra (Dover Books on Mathematics)
Georgi E. Shilov
45
Paperback
$14.35 Prime

Image courtesy of Amazon

Automatic Text Generation

- Paul Graham essays
- Shakespeare
- Wikipedia articles (including the markup)
- Algebraic Geometry (with LaTeX markup)
- Linux Source Code
- Baby Names

The model is capable of learning how to spell, punctuate, form sentences and even capture the style of the text in the corpus.

Sutskever et al. Generating Text with Recurrent Neural Networks, 2011

AI as a computer systems problem

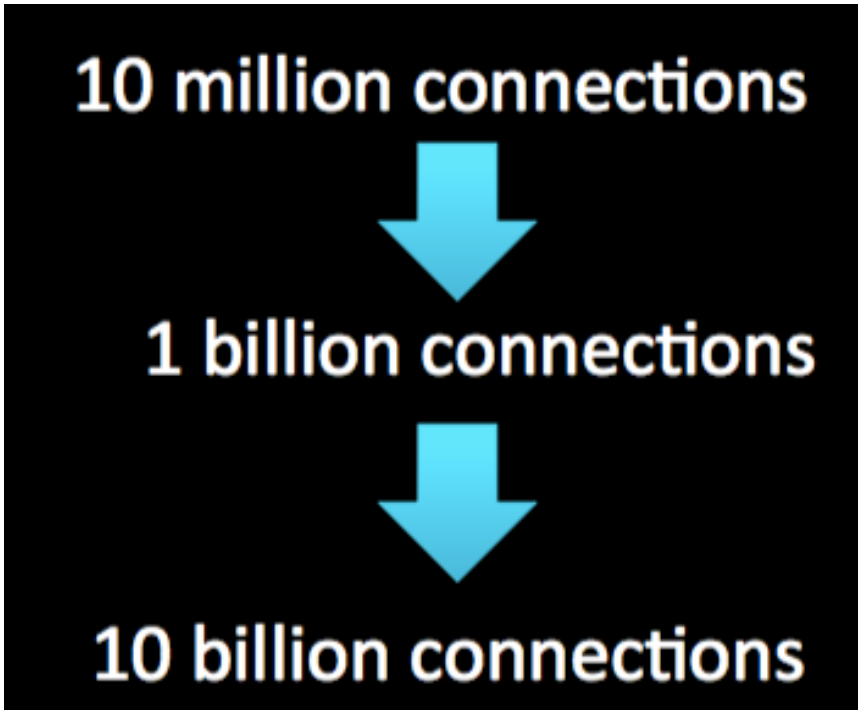
10 million connections

1 billion connections

16,000 CPUs is expensive

GPUs (Graphics Processor Unit)

Building Huge Neural Networks



What is the idea?

- Biologically inspired model
- Huge amount of training samples
- General and suitable for any input
- Supervised, unsupervised

What has been achieved?

loosely biologically inspired models

huge amount of training samples when available

general and suitable for many kinds of inputs after adaptation

supervised learning in a product

unsupervised and reinforcement learning – work in progress

A bit of history

early 1960s

Alexey Ivakhnenko

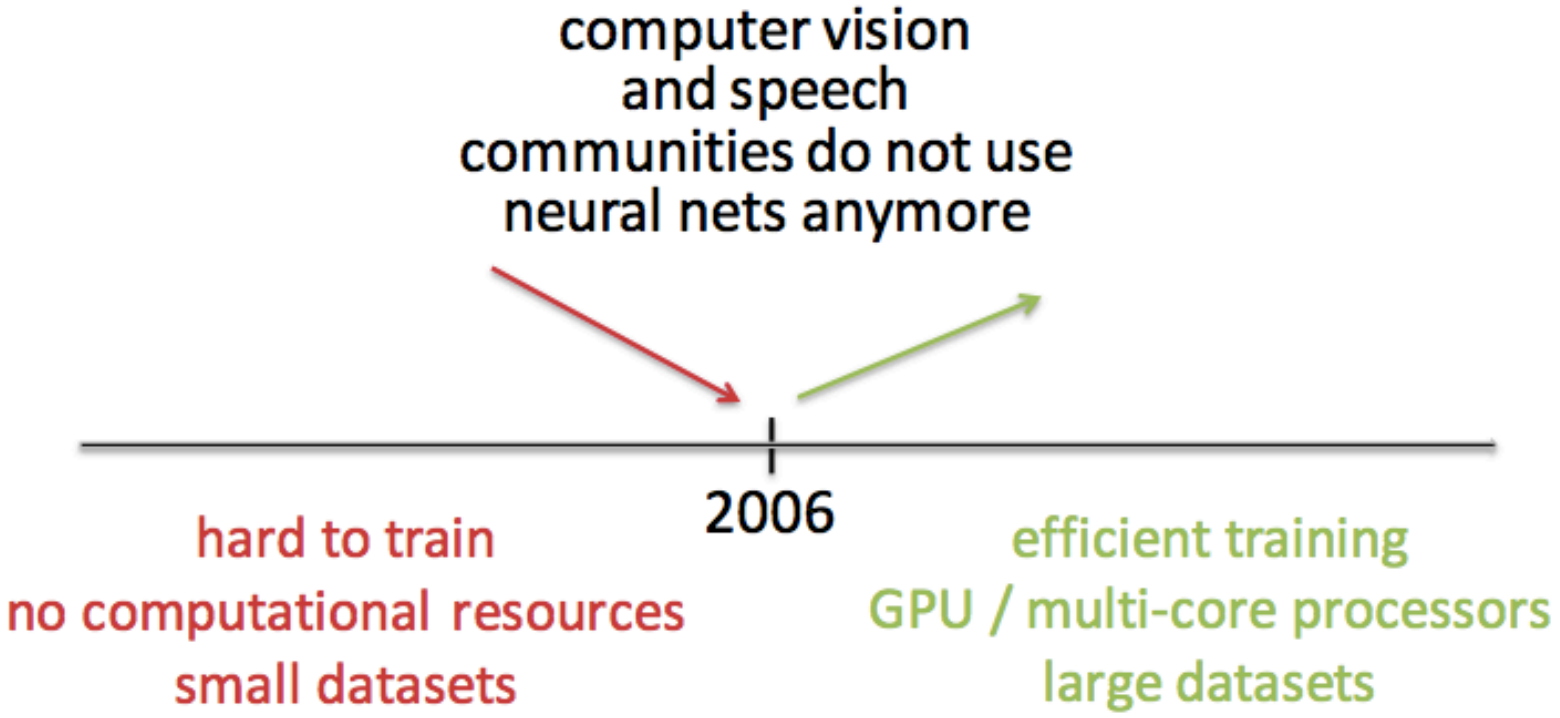
first works on deep neural networks

1986

Geoffrey Hinton

backpropagation algorithm in its current form

A bit of history



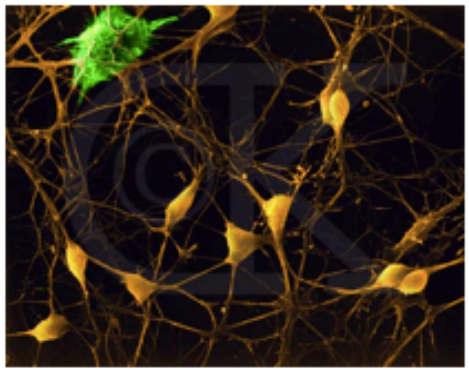
What is a Neural Network like?

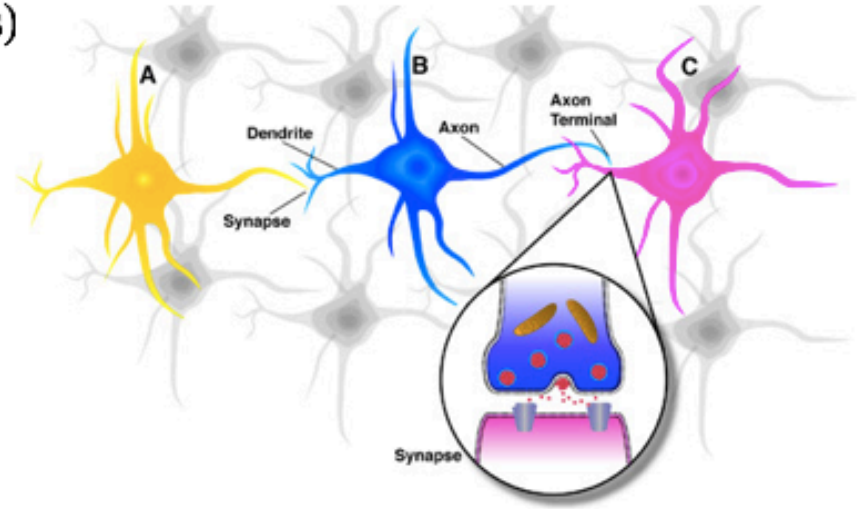
Biological Inspiration

Idea : To make the computer more robust, intelligent, and learn, …
Let's model our computer software (and/or hardware) after the brain

Neurons in the Brain

- Although heterogeneous, at a low level the brain is composed of neurons
  - A neuron receives input from other neurons (generally thousands) from its synapses
  - Inputs are approximately summed
  - When the input exceeds a threshold the neuron sends an electrical spike that travels that travels from the body, down the axon, to the next neuron(s)



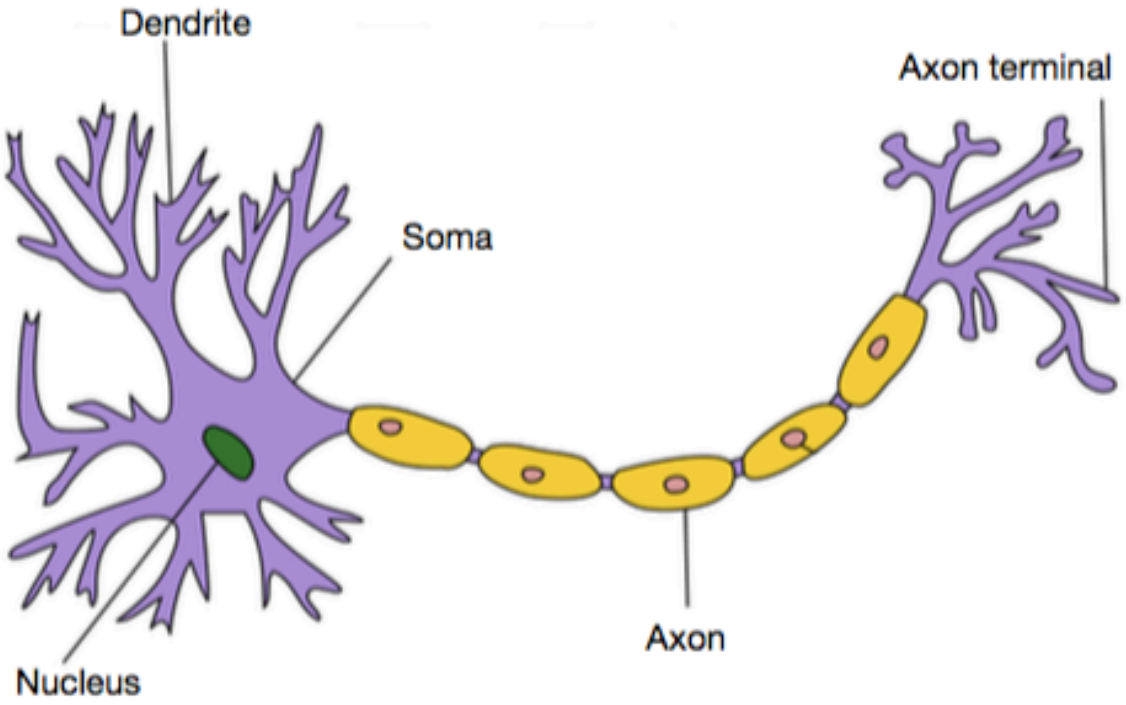


Comparison of Brains and Traditional Computers

	Personal Computer	Human Brain
processing units	1 CPU, 2–10 cores 10^{10} transistors 1–2 graphics cards/GPUs, 10^3 cores/shaders 10^{10} transistors	10^{11} neurons
storage capacity	10^{10} bytes main memory (RAM) 10^{12} bytes external memory	10^{11} neurons 10^{14} synapses
processing speed	10^{-9} seconds 10^9 operations per second	$> 10^{-3}$ seconds < 1000 per second
bandwidth	10^{12} bits/second	10^{14} bits/second
neural updates	10^6 per second	10^{14} per second

A Biological Neuron

- The human brain is made up of about 100 billion neurons



- Neurons receive electric signals at the dendrites and send them to the axon

The Brain vs Artificial Neural Networks

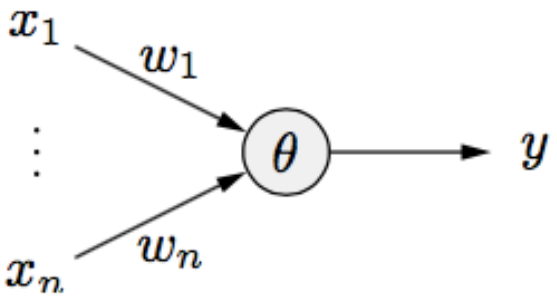
- Similarities
  - Neurons, connections between neurons
  - Learning = change of connections, not change of neurons
  - Massive parallel processing

- But artificial neural networks are much simpler
  - computation within neuron vastly simplified
  - discrete time steps
  - typically some form of supervised learning with massive number of stimuli

An Artificial Neuron

A **Threshold Logic Unit (TLU)** is a processing unit for numbers with n inputs $x_1, \ldots, x_n$ and one output y. The unit has a **threshold** θ and each input x_i is associated with a **weight** w_i. A threshold logic unit computes the function

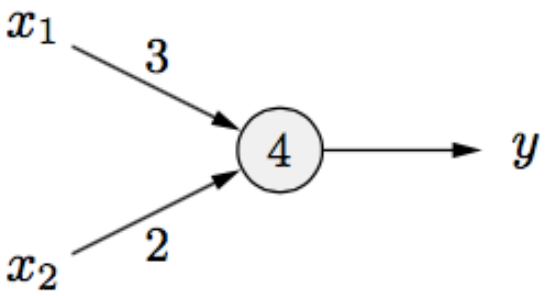
$$y = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} w_i x_i \geq \theta, \\ 0, & \text{otherwise.} \end{cases}$$



TLUs mimic the thresholding behavior of biological neurons in a (very) simple fashion.

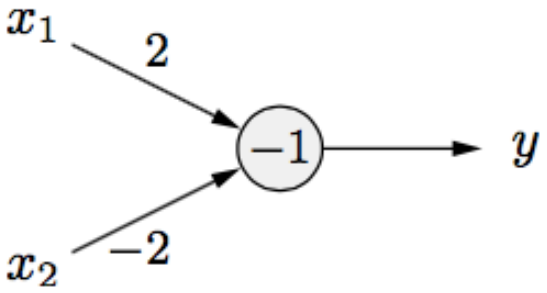
The Neuron Computation

Threshold logic unit for the conjunction $x_1 \wedge x_2$.



x_1	x_2	$3x_1 + 2x_2$	y
0	0	0	0
1	0	3	0
0	1	2	0
1	1	5	1

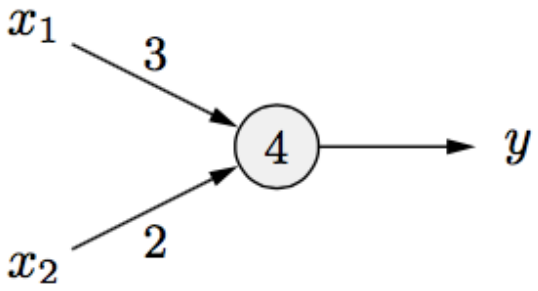
Threshold logic unit for the implication $x_2 \rightarrow x_1$.

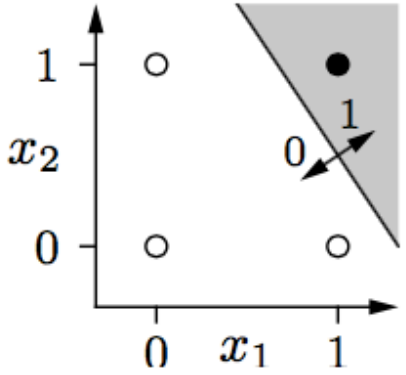


x_1	x_2	$2x_1 - 2x_2$	y
0	0	0	1
1	0	2	1
0	1	-2	0
1	1	0	1

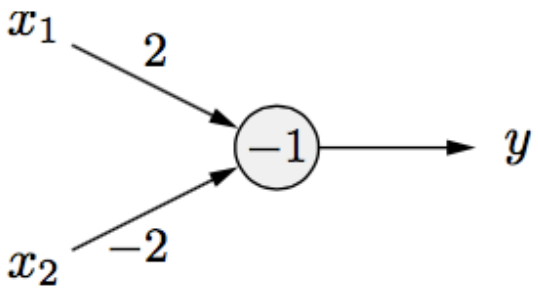
Geometric Interpretation

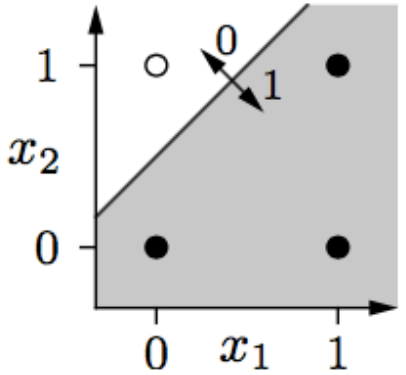
Threshold logic unit for $x_1 \wedge x_2$.





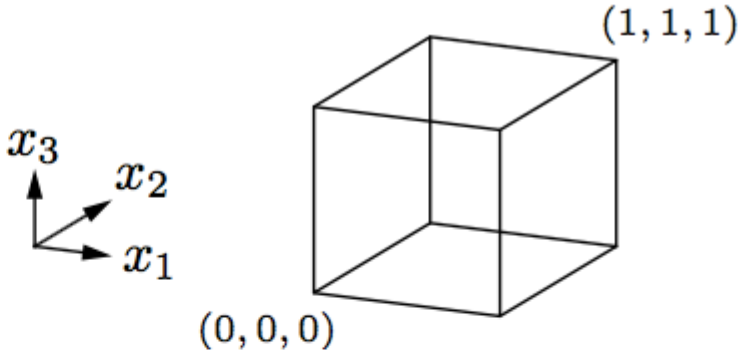
Threshold logic unit for $x_2 \rightarrow x_1$.



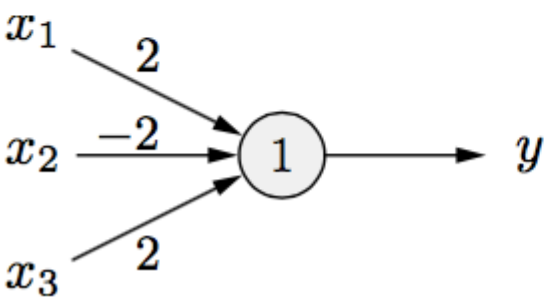


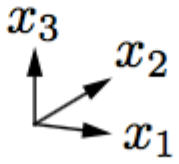
Geometric Interpretation

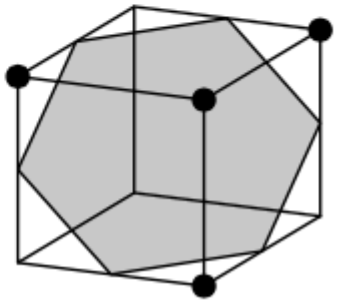
Visualization of 3-dimensional Boolean functions:



Threshold logic unit for $(x_1 \wedge \overline{x_2}) \vee (x_1 \wedge x_3) \vee (\overline{x_2} \wedge x_3)$**.**

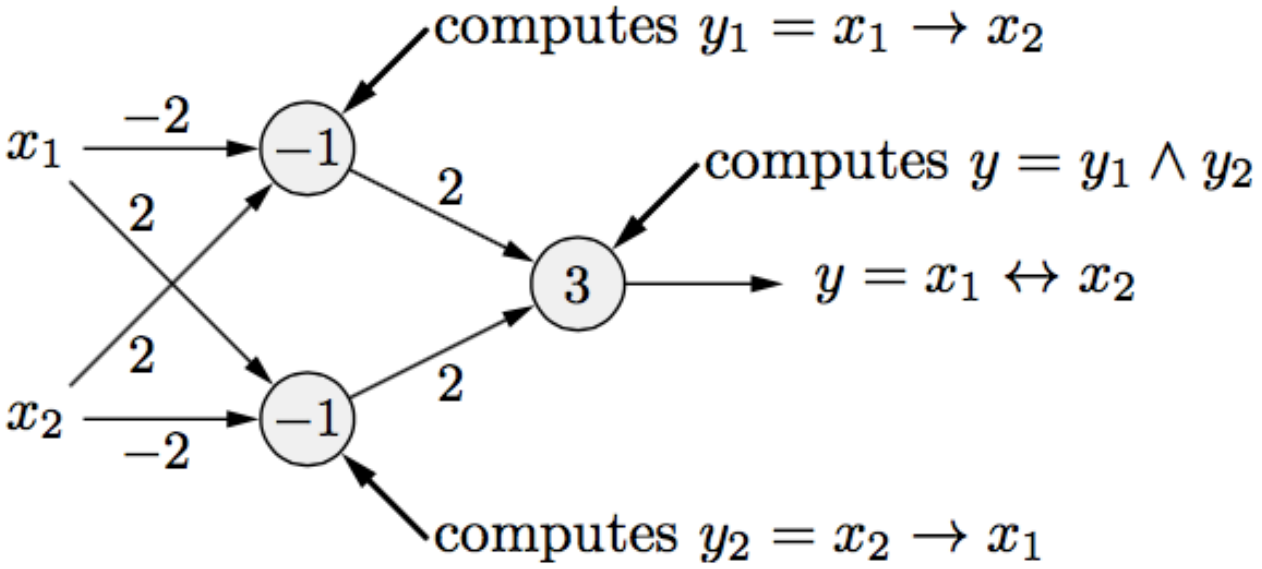






Networks of Neurons

Idea: logical decomposition $\quad x_1 \leftrightarrow x_2 \equiv (x_1 \rightarrow x_2) \wedge (x_2 \rightarrow x_1)$

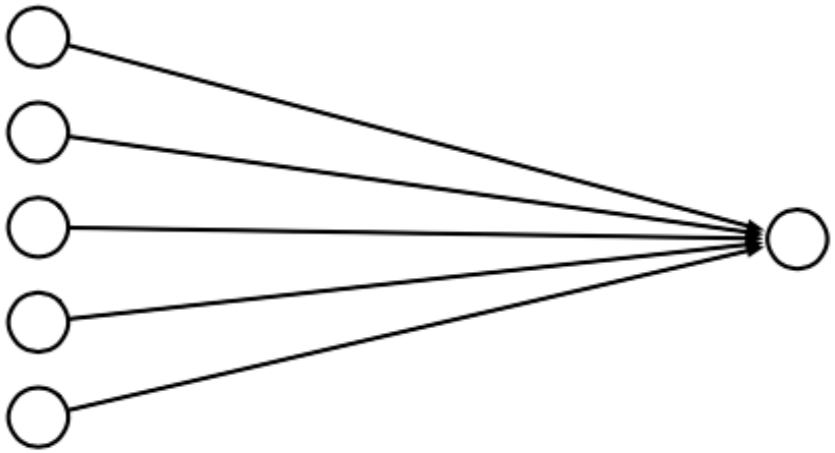


Linear Models

- We used before weighted linear combination of feature values h_j and weights λ_j

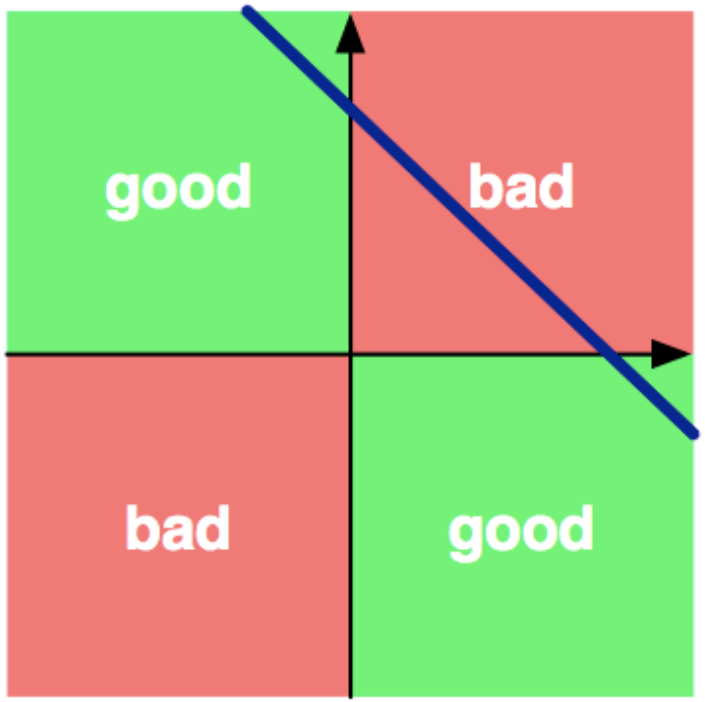
$$\text{score}(\lambda, \mathbf{d}_i) = \sum_j \lambda_j \; h_j(\mathbf{d}_i)$$

- Such models can be illustrated as a "network"



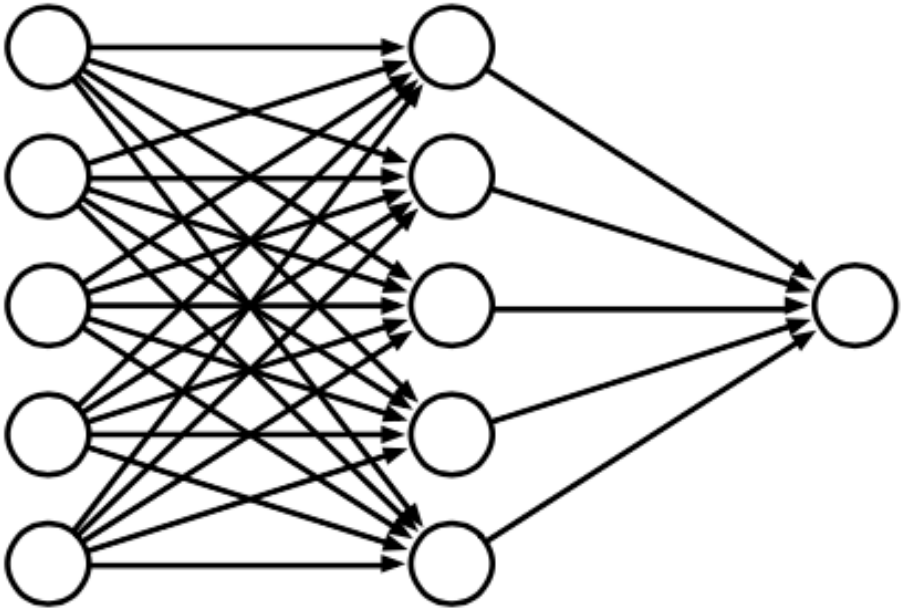
XOR

- Linear models cannot model XOR



Multiple Layers

- Add an intermediate ("hidden") layer of processing (each arrow is a weight)



- Have we gained anything so far?

Non-Linearity

- Instead of computing a linear combination

$$\text{score}(\lambda, \mathbf{d}_i) = \sum_j \lambda_j \, h_j(\mathbf{d}_i)$$

- Add a non-linear function

$$\text{score}(\lambda, \mathbf{d}_i) = f\left(\sum_j \lambda_j \, h_j(\mathbf{d}_i)\right)$$

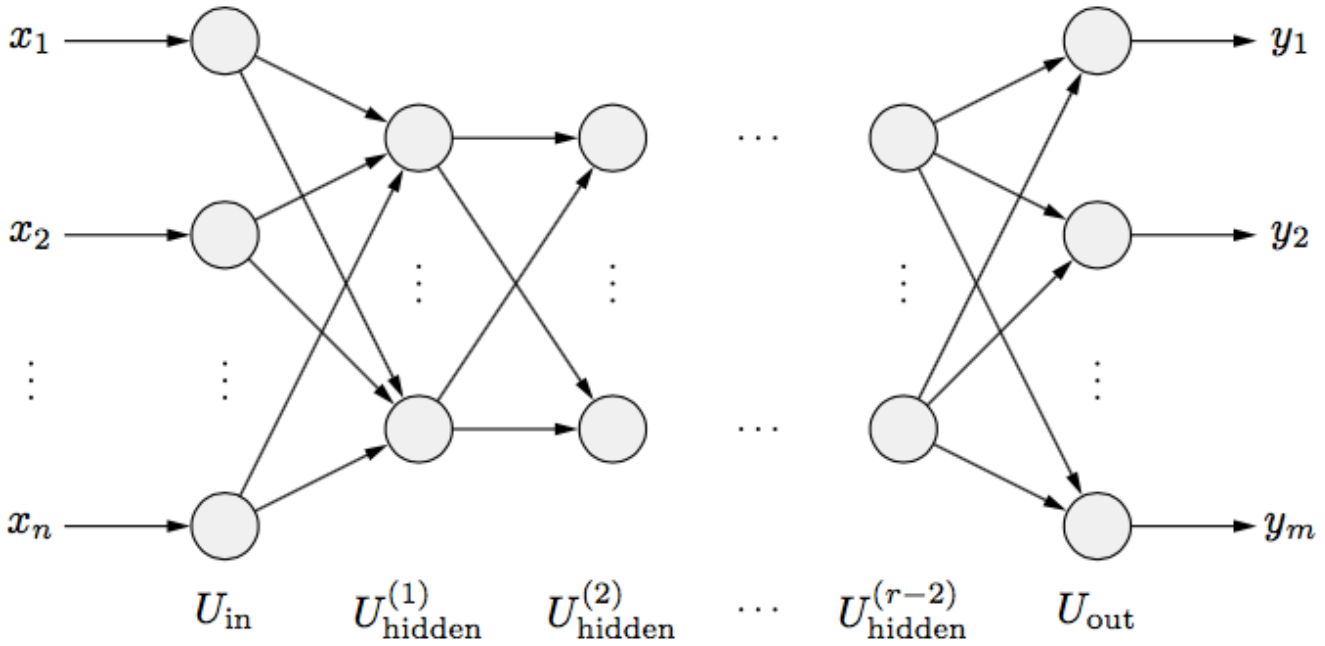
- Popular choices

tanh(x) sigmoid(x) = $\frac{1}{1+e^{-x}}$ relu(x) = max(0,x)

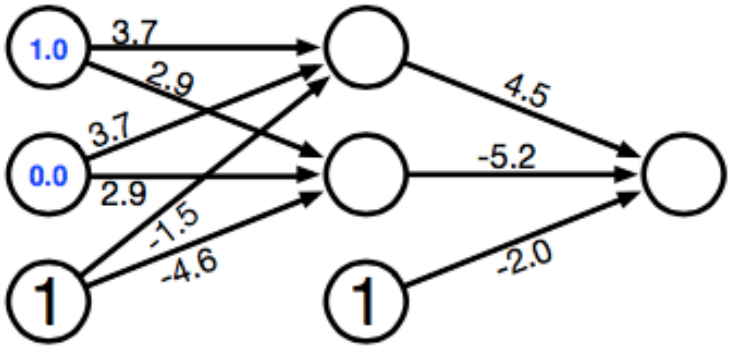
(sigmoid is also called the "logistic function")

More Layers = Deep Learning

General structure of a multi-layer perceptron



Example

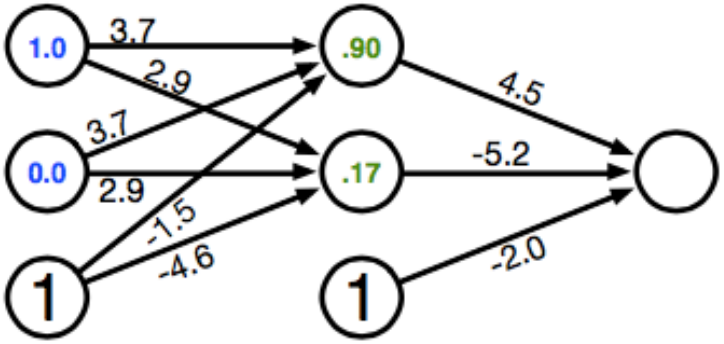


- Try out two input values
- Hidden unit computation

$$\text{sigmoid}(1.0 \times 3.7 + 0.0 \times 3.7 + 1 \times -1.5) = \text{sigmoid}(2.2) = \frac{1}{1+e^{-2.2}} = 0.90$$

$$\text{sigmoid}(1.0 \times 2.9 + 0.0 \times 2.9 + 1 \times -4.5) = \text{sigmoid}(-1.6) = \frac{1}{1+e^{1.6}} = 0.17$$

Example

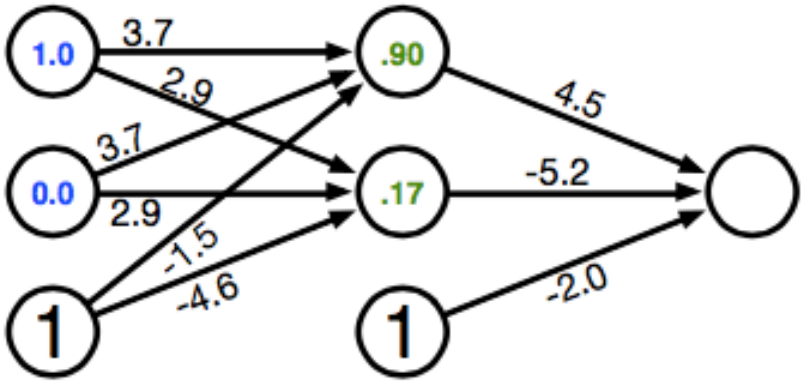


- Try out two input values
- Hidden unit computation

$$\text{sigmoid}(1.0 \times 3.7 + 0.0 \times 3.7 + 1 \times -1.5) = \text{sigmoid}(2.2) = \frac{1}{1+e^{-2.2}} = 0.90$$

$$\text{sigmoid}(1.0 \times 2.9 + 0.0 \times 2.9 + 1 \times -4.5) = \text{sigmoid}(-1.6) = \frac{1}{1+e^{1.6}} = 0.17$$

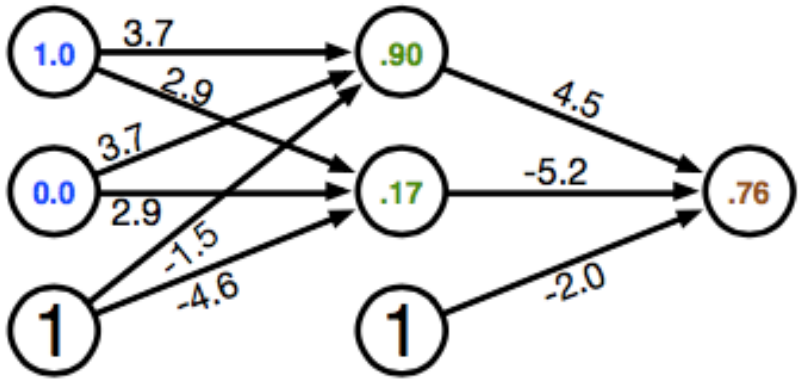
Example



- Output unit computation

$$\text{sigmoid}(.90 \times 4.5 + .17 \times -5.2 + 1 \times -2.0) = \text{sigmoid}(1.17) = \frac{1}{1+e^{-1.17}} = 0.76$$

Example

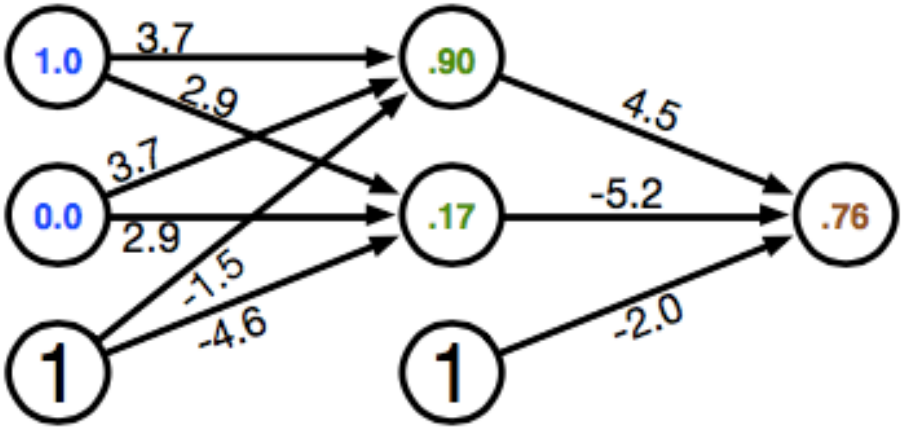


- Output unit computation

$$\text{sigmoid}(.90 \times 4.5 + .17 \times -5.2 + 1 \times -2.0) = \text{sigmoid}(1.17) = \frac{1}{1 + e^{-1.17}} = 0.76$$

Backpropagation in Training

Error



- Computed output: $y = .76$
- Correct output: $t = 1.0$

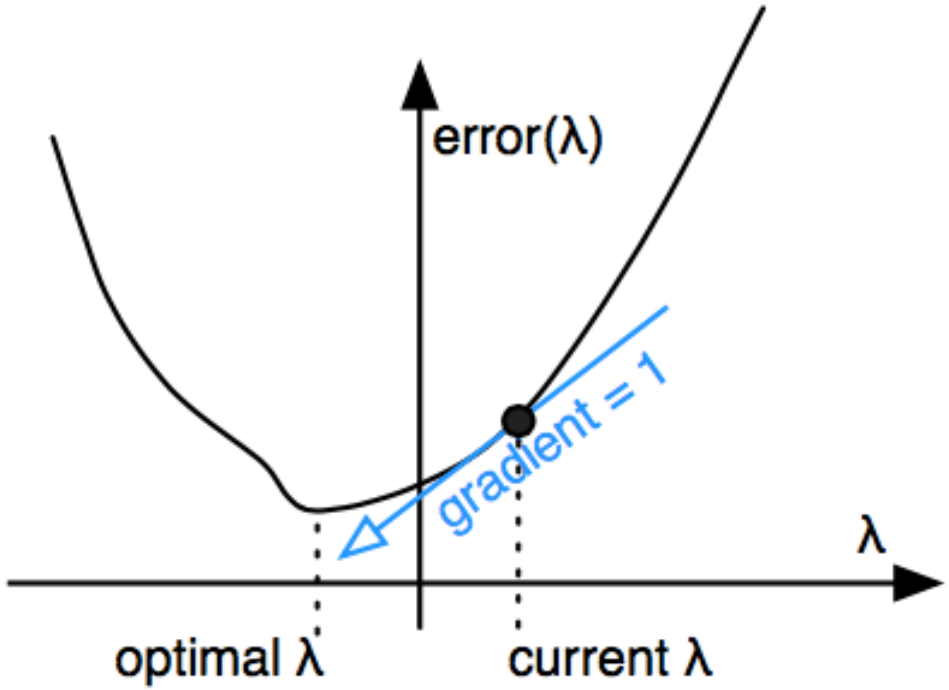
$\Rightarrow$ How do we adjust the weights?

Key Concepts

- Gradient descent
  - error is a function of the weights
  - we want to reduce the error
  - gradient descent: move towards the error minimum
  - compute gradient → get direction to the error minimum
  - adjust weights towards direction of lower error

- Back-propagation
  - first adjust last set of weights
  - propagate error back to each previous layer
  - adjust their weights

Gradient Descent



Derivative of Sigmoid

- Sigmoid

$$\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$$

- Reminder: quotient rule

$$\left(\frac{f(x)}{g(x)}\right)' = \frac{g(x)f'(x) - f(x)g'(x)}{g(x)^2}$$

- Derivative

$$\begin{aligned}
\frac{d\,\text{sigmoid}(x)}{dx} &= \frac{d}{dx}\frac{1}{1+e^{-x}} \\
&= \frac{0 \times (1-e^{-x}) - (-e^{-x})}{(1+e^{-x})^2} \\
&= \frac{1}{1+e^{-x}}\left(\frac{e^{-x}}{1+e^{-x}}\right) \\
&= \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) \\
&= \text{sigmoid}(x)(1 - \text{sigmoid}(x))
\end{aligned}$$

Final Layer Update

- Linear combination of weights $s = \sum_k w_k h_k$
- Activation function $y = \text{sigmoid}(s)$
- Error (L2 norm) $E = \frac{1}{2}(t - y)^2$
- Derivative of error with regard to one weight w_k

$$\frac{dE}{dw_k} = \frac{dE}{dy}\frac{dy}{ds}\frac{ds}{dw_k}$$

Final Layer Update (1)

- Linear combination of weights $s = \sum_k w_k h_k$
- Activation function $y = \text{sigmoid}(s)$
- Error (L2 norm) $E = \frac{1}{2}(t - y)^2$
- Derivative of error with regard to one weight w_k

$$\frac{dE}{dw_k} = \frac{dE}{dy}\frac{dy}{ds}\frac{ds}{dw_k}$$

- Error E is defined with respect to y

$$\frac{dE}{dy} = \frac{d}{dy}\frac{1}{2}(t - y)^2 = -(t - y)$$

Final Layer Update (2)

- Linear combination of weights $s = \sum_k w_k h_k$
- Activation function $y = \text{sigmoid}(s)$
- Error (L2 norm) $E = \frac{1}{2}(t - y)^2$
- Derivative of error with regard to one weight w_k

$$\frac{dE}{dw_k} = \frac{dE}{dy}\frac{dy}{ds}\frac{ds}{dw_k}$$

- y with respect to x is $\text{sigmoid}(s)$

$$\frac{dy}{ds} = \frac{d\,\text{sigmoid}(s)}{ds} = \text{sigmoid}(s)(1 - \text{sigmoid}(s)) = y(1 - y)$$

Final Layer Update (3)

- Linear combination of weights $s = \sum_k w_k h_k$
- Activation function $y = \text{sigmoid}(s)$
- Error (L2 norm) $E = \frac{1}{2}(t - y)^2$
- Derivative of error with regard to one weight w_k

$$\frac{dE}{dw_k} = \frac{dE}{dy}\frac{dy}{ds}\frac{ds}{dw_k}$$

- x is weighted linear combination of hidden node values h_k

$$\frac{ds}{dw_k} = \frac{d}{dw_k}\sum_k w_k h_k = h_k$$

Putting it All Together

- Derivative of error with regard to one weight w_k

$$\frac{dE}{dw_k} = \frac{dE}{dy}\frac{dy}{ds}\frac{ds}{dw_k}$$
$$= -(t-y) \quad y(1-y) \quad h_k$$

  - error
  - derivative of sigmoid: y'

- Weight adjustment will be scaled by a fixed learning rate μ

$$\Delta w_k = \mu\ (t-y)\ y'\ h_k$$

Multiple Output Nodes

- Our example only had one output node
- Typically neural networks have multiple output nodes
- Error is computed over all j output nodes

$$E = \sum_j \frac{1}{2}(t_j - y_j)^2$$

- Weights $k \rightarrow j$ are adjusted according to the node they point to

$$\Delta w_{j \leftarrow k} = \mu (t_j - y_j)\, y'_j\, h_k$$

Hidden Layer Update

- In a hidden layer, we do not have a target output value
- But we can compute how much each node contributed to downstream error
- Definition of error term of each node

$$\delta_j = (t_j - y_j)\, y'_j$$

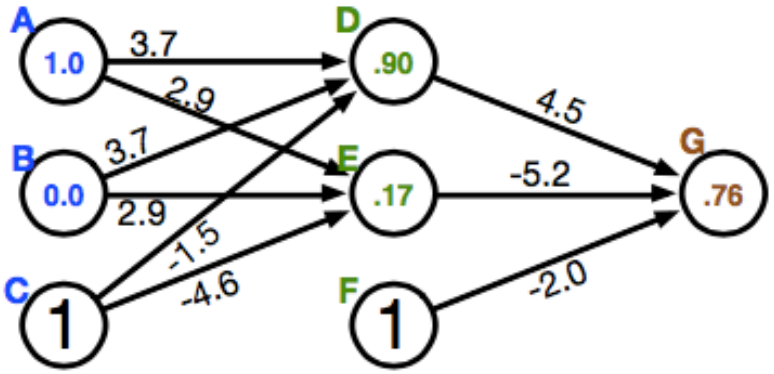
- Back-propagate the error term
  (why this way? there is math to back it up...)

$$\delta_i = \left(\sum_j w_{j \leftarrow i} \delta_j \right) y'_i$$

- Universal update formula

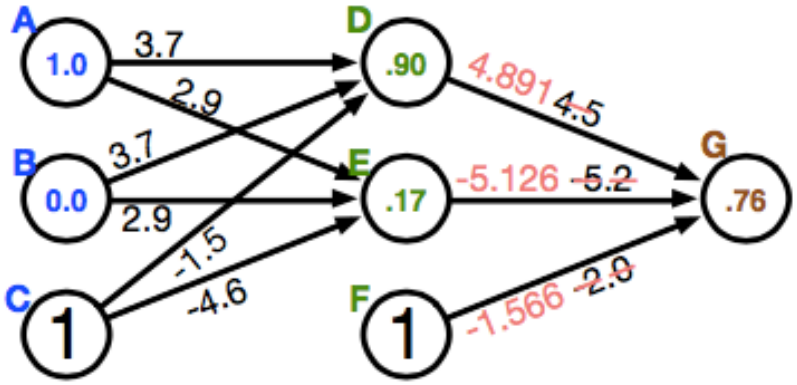
$$\Delta w_{j \leftarrow k} = \mu\, \delta_j\, h_k$$

Our Example



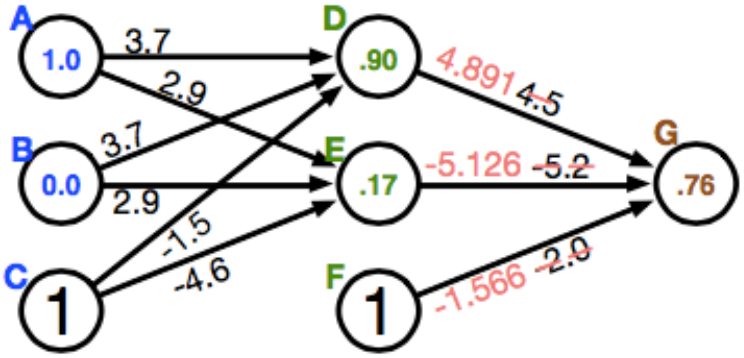
- Computed output: $y = .76$
- Correct output: $t = 1.0$
- Final layer weight updates (learning rate $\mu = 10$)
  - $\delta_G = (t - y)\, y' = (1 - .76)\, 0.181 = .0434$
  - $\Delta w_{GD} = \mu\, \delta_G\, h_D = 10 \times .0434 \times .90 = .391$
  - $\Delta w_{GE} = \mu\, \delta_G\, h_E = 10 \times .0434 \times .17 = .074$
  - $\Delta w_{GF} = \mu\, \delta_G\, h_F = 10 \times .0434 \times 1 = .434$

Our Example



- Computed output: $y = .76$
- Correct output: $t = 1.0$
- Final layer weight updates (learning rate $\mu = 10$)
  - $\delta_G = (t - y)\, y' = (1 - .76)\, 0.181 = .0434$
  - $\Delta w_{GD} = \mu\, \delta_G\, h_D = 10 \times .0434 \times .90 = .391$
  - $\Delta w_{GE} = \mu\, \delta_G\, h_E = 10 \times .0434 \times .17 = .074$
  - $\Delta w_{GF} = \mu\, \delta_G\, h_F = 10 \times .0434 \times 1 = .434$

Hidden Layer Updates



- Hidden node **D**
  - $\delta_{\mathsf{D}} = \left(\sum_j w_{j \leftarrow i} \delta_j\right) y'_{\mathsf{D}} = w_{\mathsf{GD}}\, \delta_{\mathsf{G}}\, y'_{\mathsf{D}} = 4.5 \times .0434 \times .0898 = .0175$
  - $\Delta w_{\mathsf{DA}} = \mu\, \delta_{\mathsf{D}}\, h_{\mathsf{A}} = 10 \times .0175 \times 1.0 = .175$
  - $\Delta w_{\mathsf{DB}} = \mu\, \delta_{\mathsf{D}}\, h_{\mathsf{B}} = 10 \times .0175 \times 0.0 = 0$
  - $\Delta w_{\mathsf{DC}} = \mu\, \delta_{\mathsf{D}}\, h_{\mathsf{C}} = 10 \times .0175 \times 1 = .175$
- Hidden node **E**
  - $\delta_{\mathsf{E}} = \left(\sum_j w_{j \leftarrow i} \delta_j\right) y'_{\mathsf{E}} = w_{\mathsf{GE}}\, \delta_{\mathsf{G}}\, y'_{\mathsf{E}} = -5.2 \times .0434 \times 0.2055 = -.0464$
  - $\Delta w_{\mathsf{EA}} = \mu\, \delta_{\mathsf{E}}\, h_{\mathsf{A}} = 10 \times -.0464 \times 1.0 = -.464$
  - etc.

Additional Aspects

GPU

- Neural network layers may have, say, 200 nodes
- Computations such as $W\vec{h}$ require $200 \times 200 = 40,000$ multiplications
- Graphics Processing Units (GPU) are designed for such computations
  - image rendering requires such vector and matrix operations
  - massively mulit-core but lean processing units
  - example: NVIDIA Tesla K20c GPU provides 2496 thread processors
- Extensions to C to support programming of GPUs, such as CUDA

Toolkits

- Theano
- Tensorflow (Google)
- PyTorch (Facebook)
- MXNet (Amazon)
- DyNet

References

- Introduction to Neural Networks, Philipp Koehn, 3 October 2017
- Artificial Neural Networks and Deep Learning, Christian Borgelt, University of Konstanz, Germany