DS 5220

Supervised Machine Learning and Learning Theory

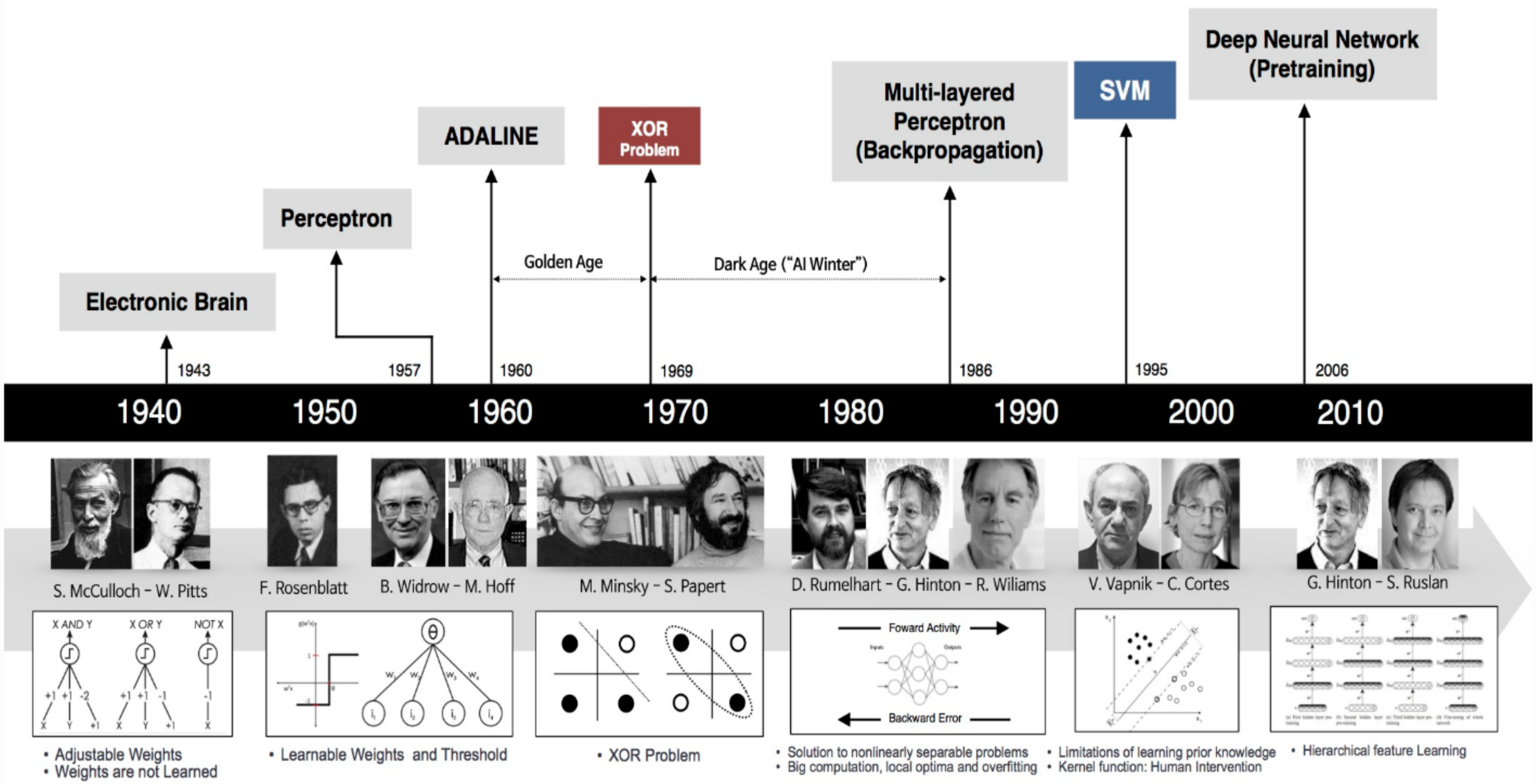
Alina Oprea
Associate Professor, CCIS
Northeastern University

November 6 2019

Comparing classifiers

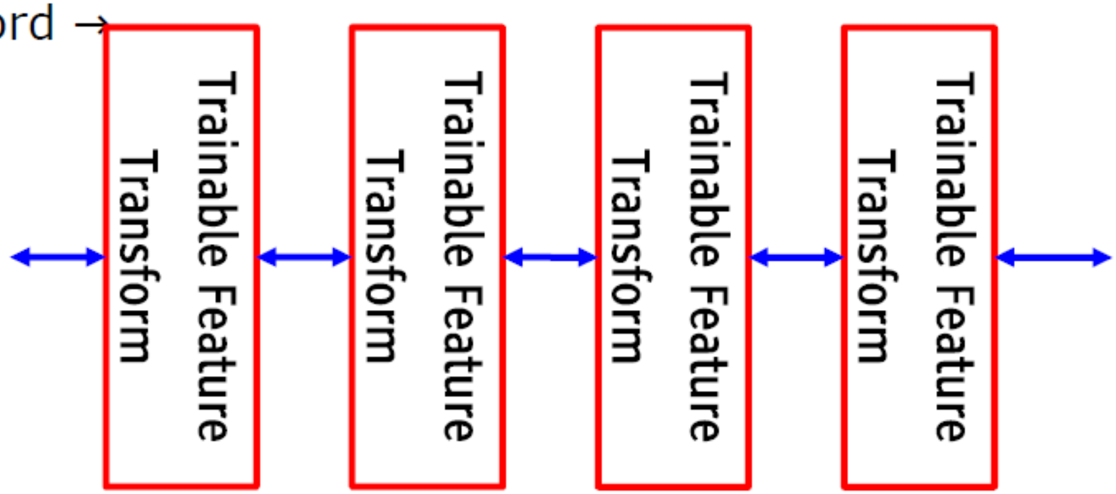
Algorithm	Interpretable	Model size	Predictive accuracy	Training time	Testing time
Logistic regression	High	Small	Lower	Low	Low
kNN	Medium	Large	Lower	No training	High
LDA	Medium	Small	Lower	Low	Low
Decision trees	High	Medium	Lower	Medium	Low
Ensembles	Low	Large	High	High	High
Naïve Bayes	Medium	Small	Lower	Medium	Low
SVM	Medium	Small	High	High	Low
Neural Networks	Low	Large	High	High	Low

History of Deep Learning

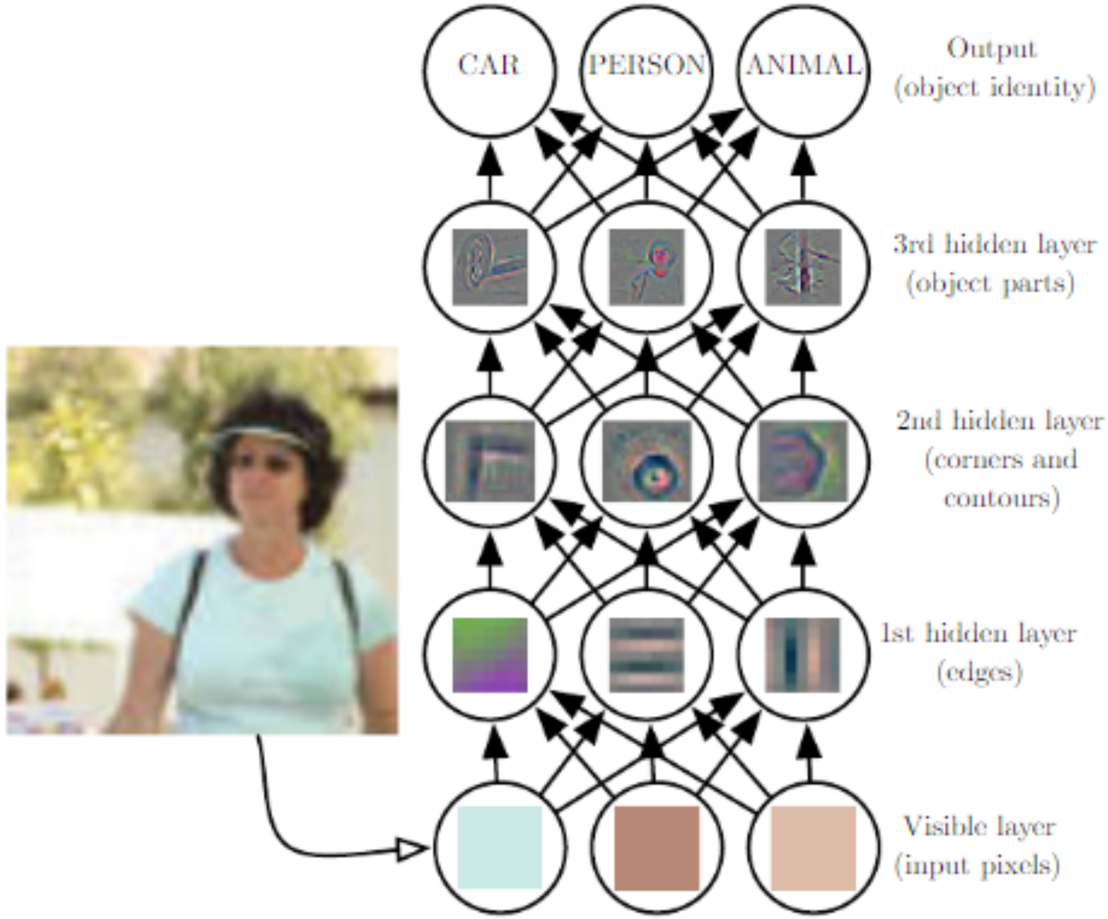


Trainable Feature Hierarchy

- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable feature transform
- Image recognition
  - Pixel → edge → texton → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story
- Speech
  - Sample → spectral band → sound → … → phone → phoneme → word →



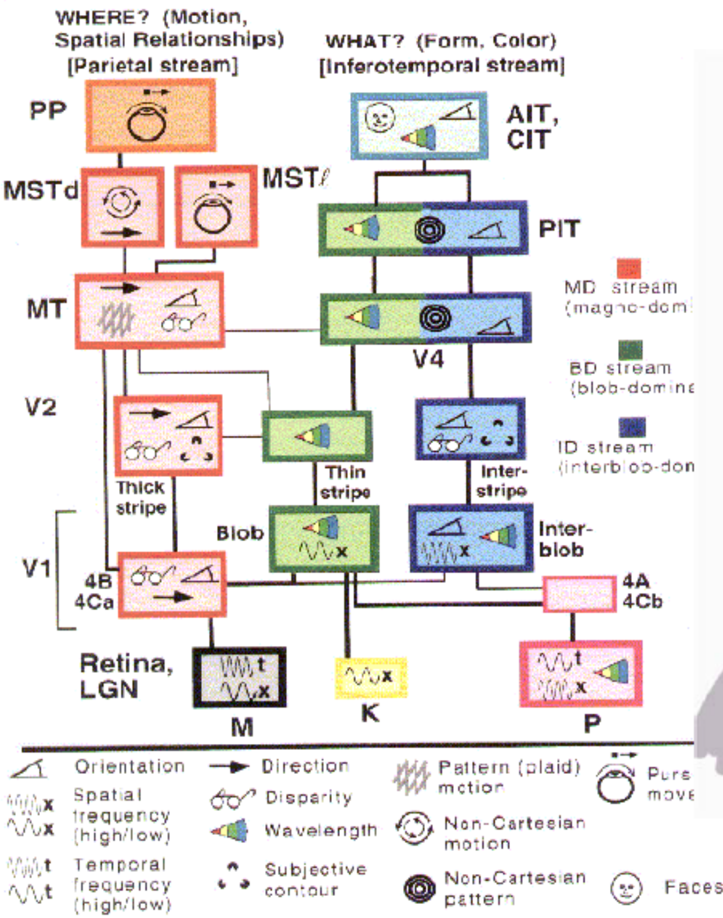
Learning Representations

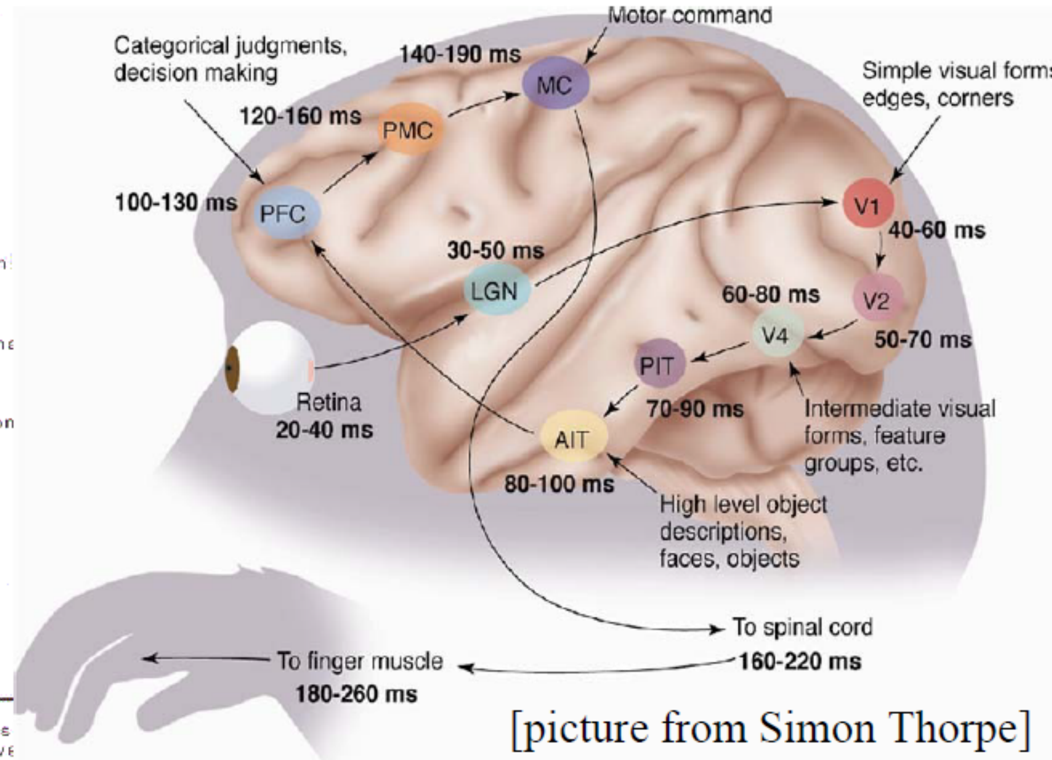


Deep Learning addresses the problem of learning hierarchical representations

The Visual Cortex is Hierarchical

- The ventral (recognition) pathway in the visual cortex has multiple stages
- Retina - LGN - V1 - V2 - V4 - PIT - AIT ....
- Lots of intermediate representations





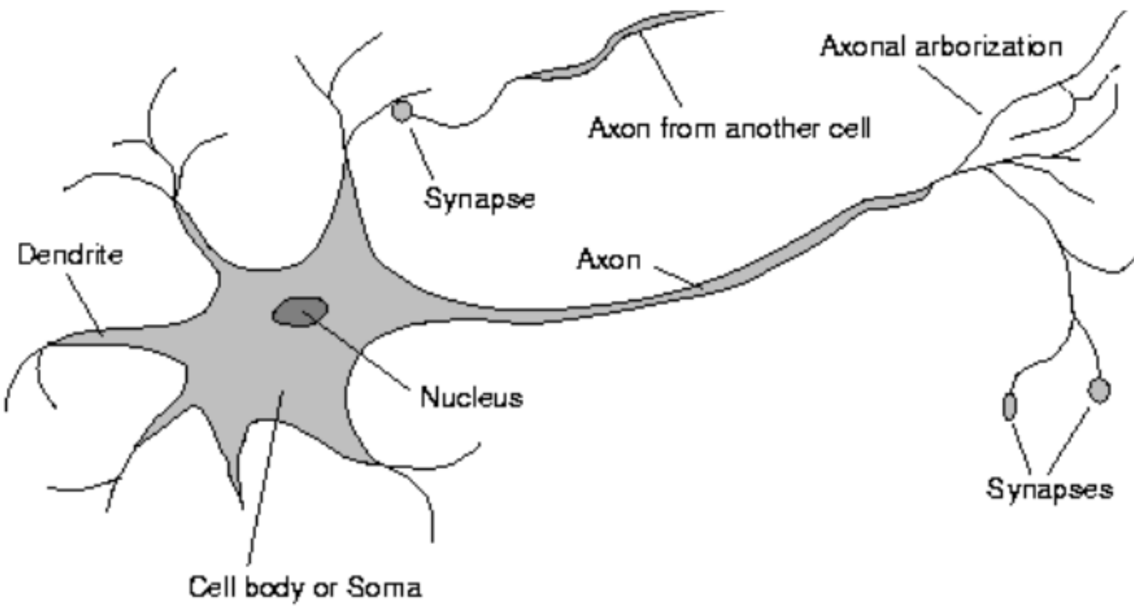
[picture from Simon Thorpe]

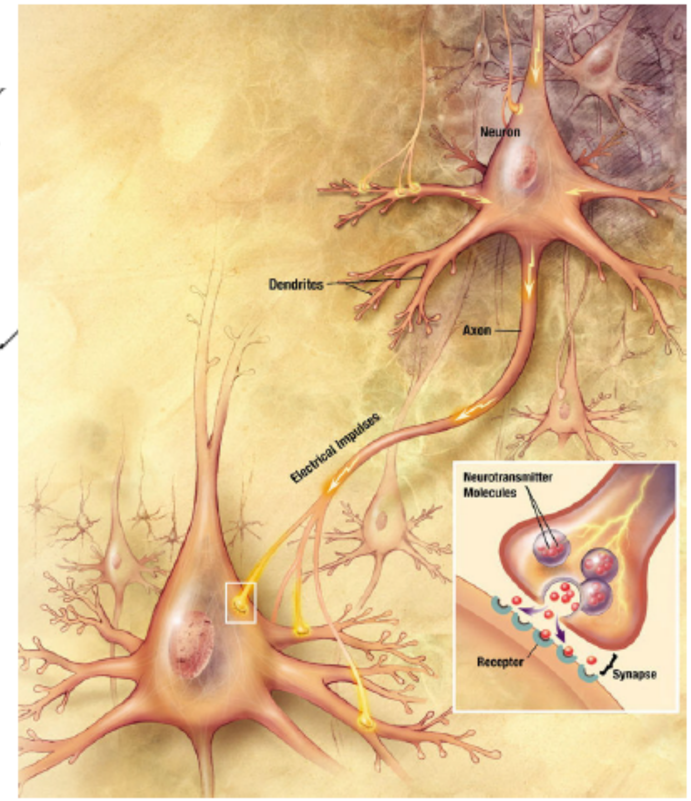
[Gallant & Van Essen]

Neural Function

- Brain function (thought) occurs as the result of the firing of **neurons**
- Neurons connect to each other through **synapses**, which propagate **action potential** (electrical impulses) by releasing **neurotransmitters**
  - Synapses can be **excitatory** (potential-increasing) or **inhibitory** (potential-decreasing), and have varying **activation thresholds**
  - Learning occurs as a result of the synapses' **plasticicity**: They exhibit long-term changes in connection strength
- There are about 10^{11} neurons and about 10^{14} synapses in the human brain!

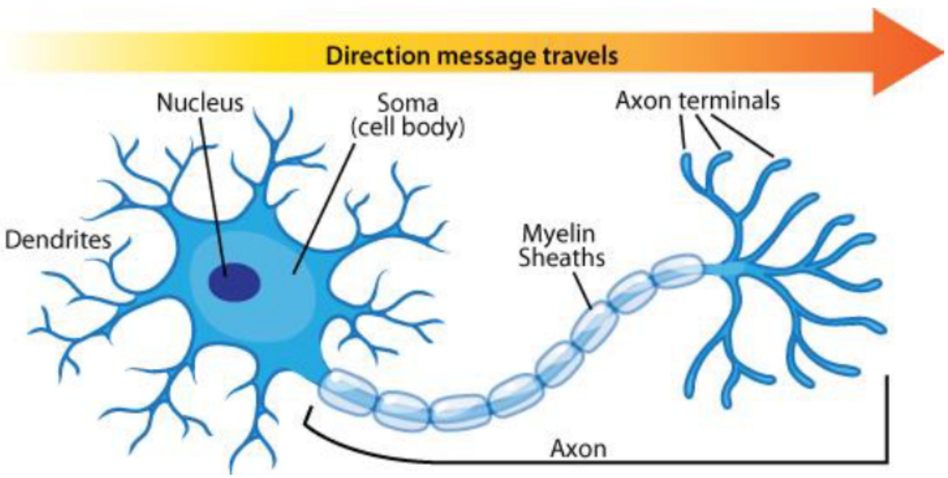
Biology of a Neuron



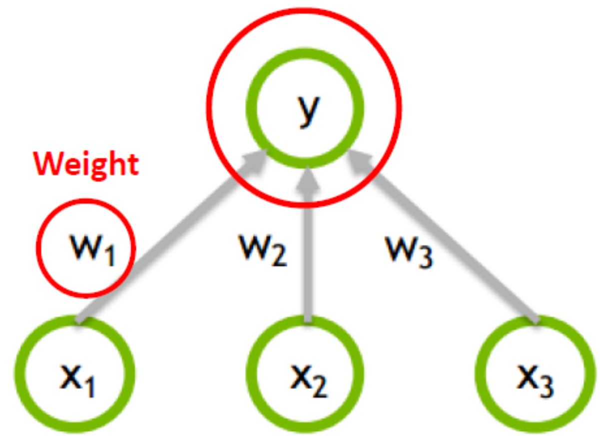


Analogy to Human Brain

Human Brain



Biological Neuron



$$y=F(w_1x_1+w_2x_2+w_3x_3)$$

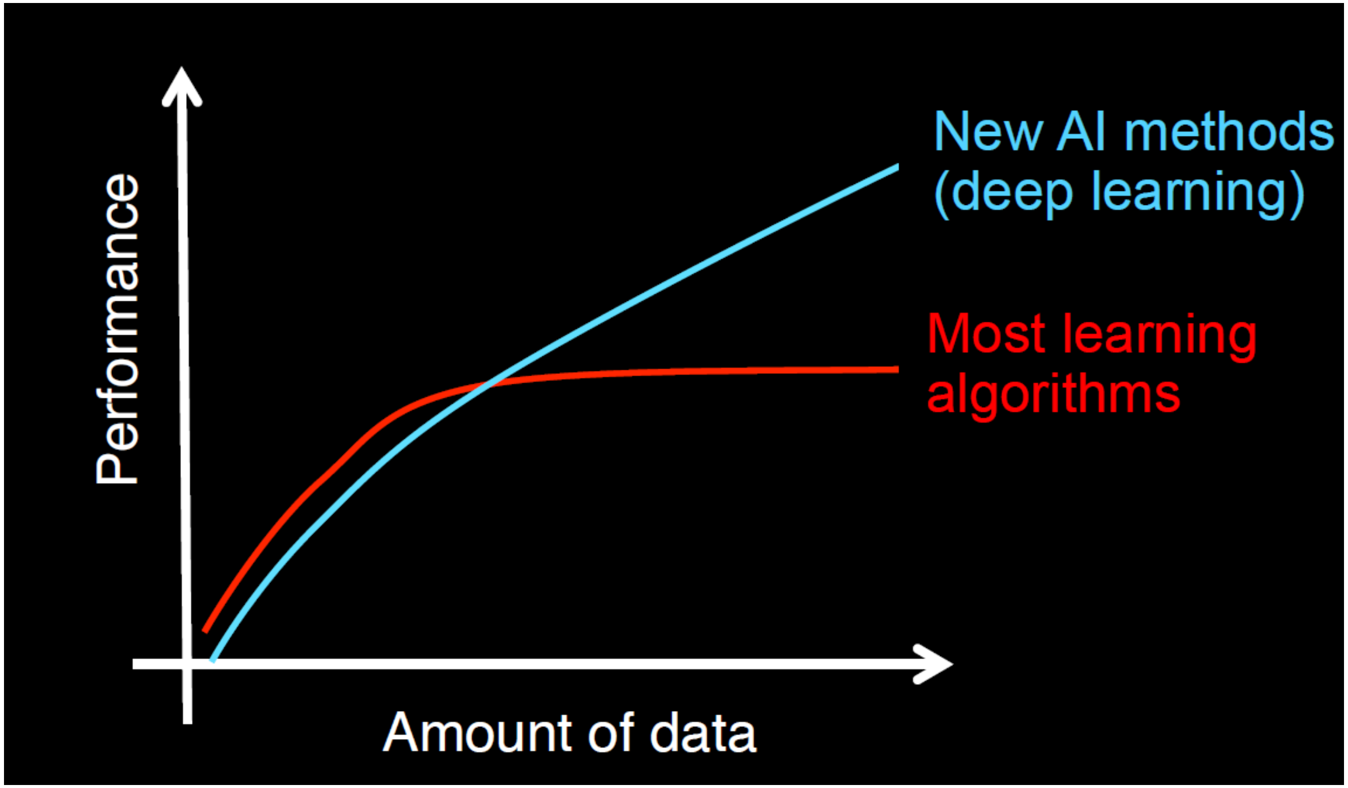
$$F(x)=\max(0,x)$$

Artificial Neuron

Neural Networks

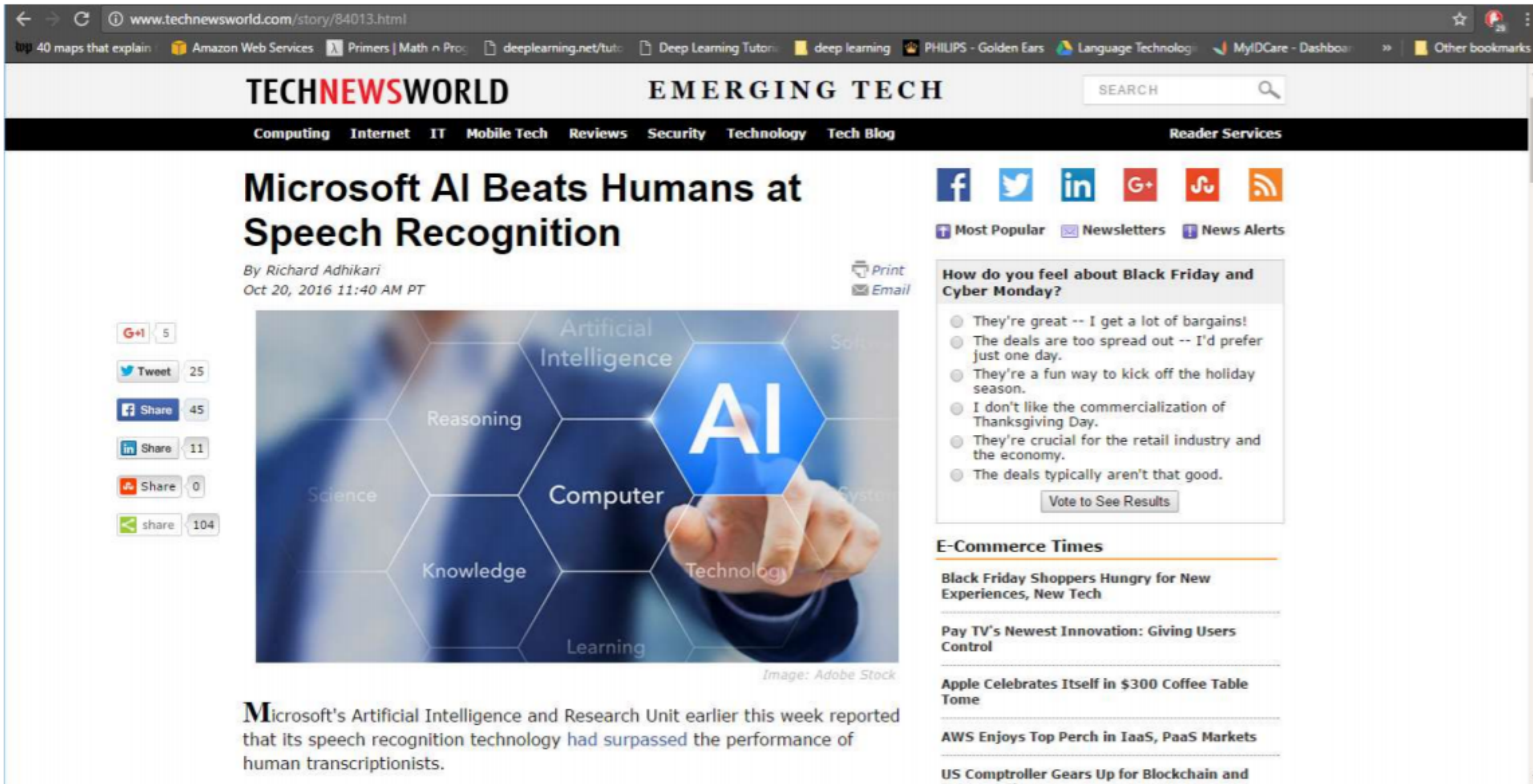
- Origins: Algorithms that try to mimic the brain.
- Very widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications
- Artificial neural networks are not nearly as complex or intricate as the actual brain structure

Performance of Deep Learning

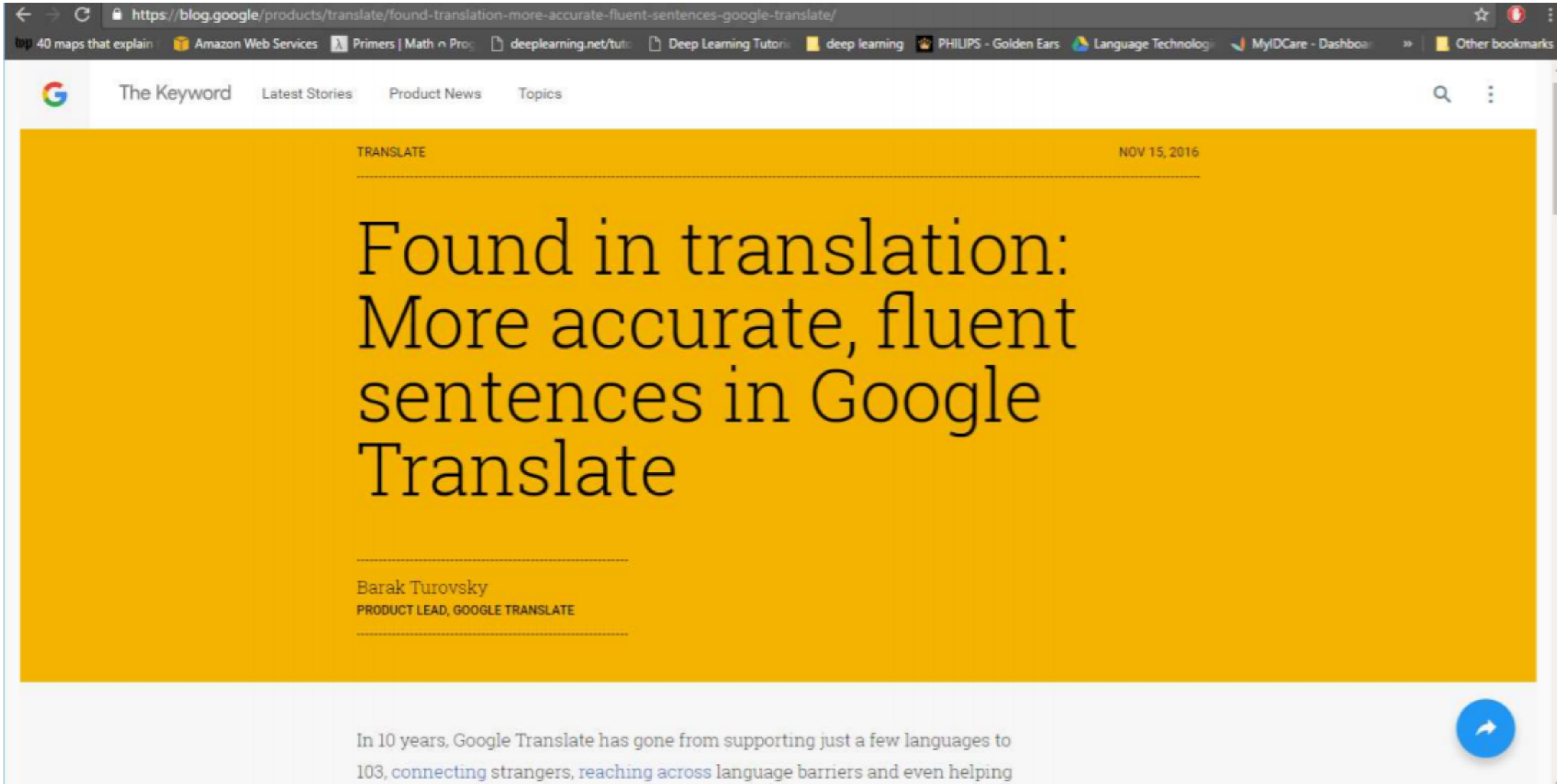


Deep Learning Applications

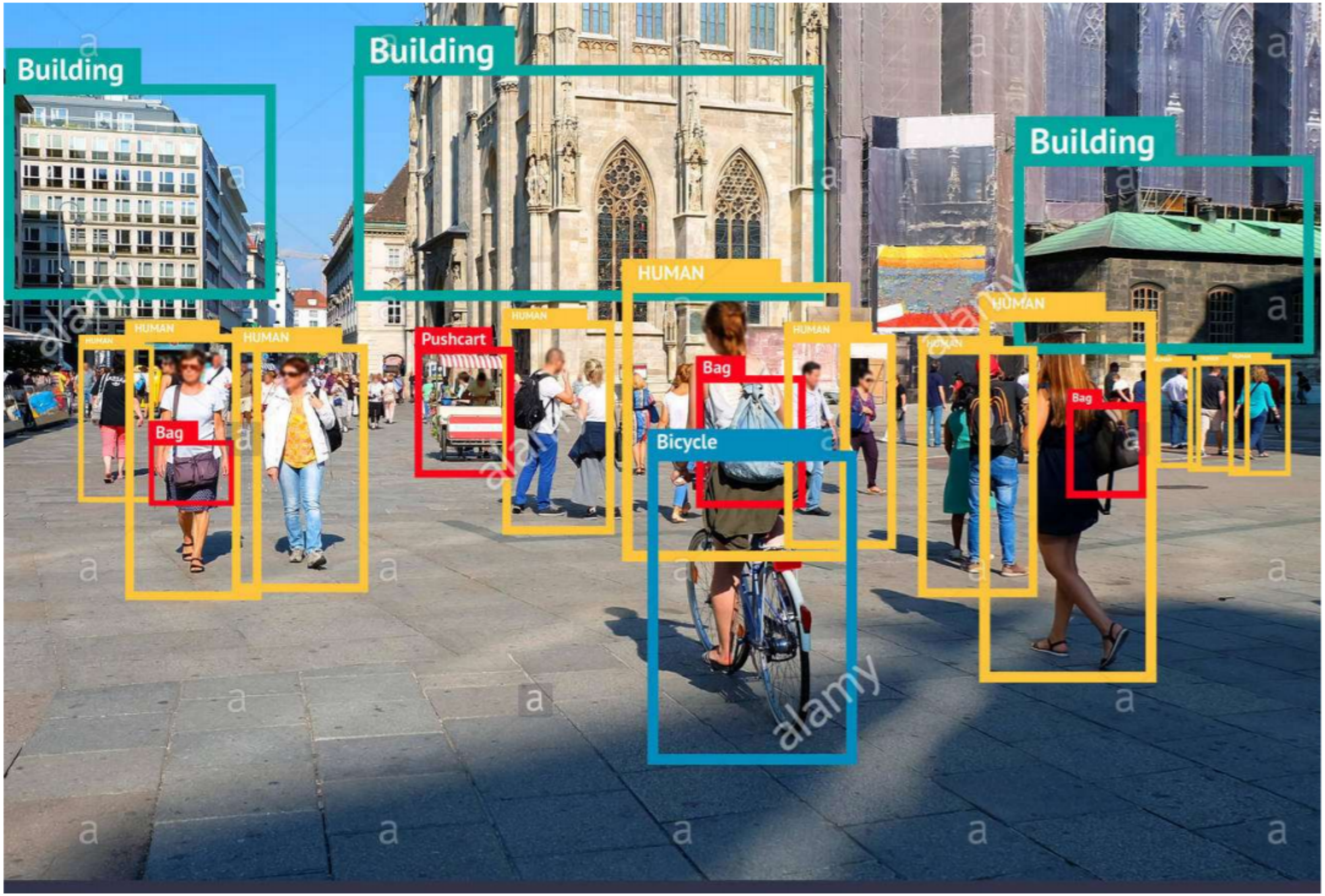
Success stories: Speech recognition



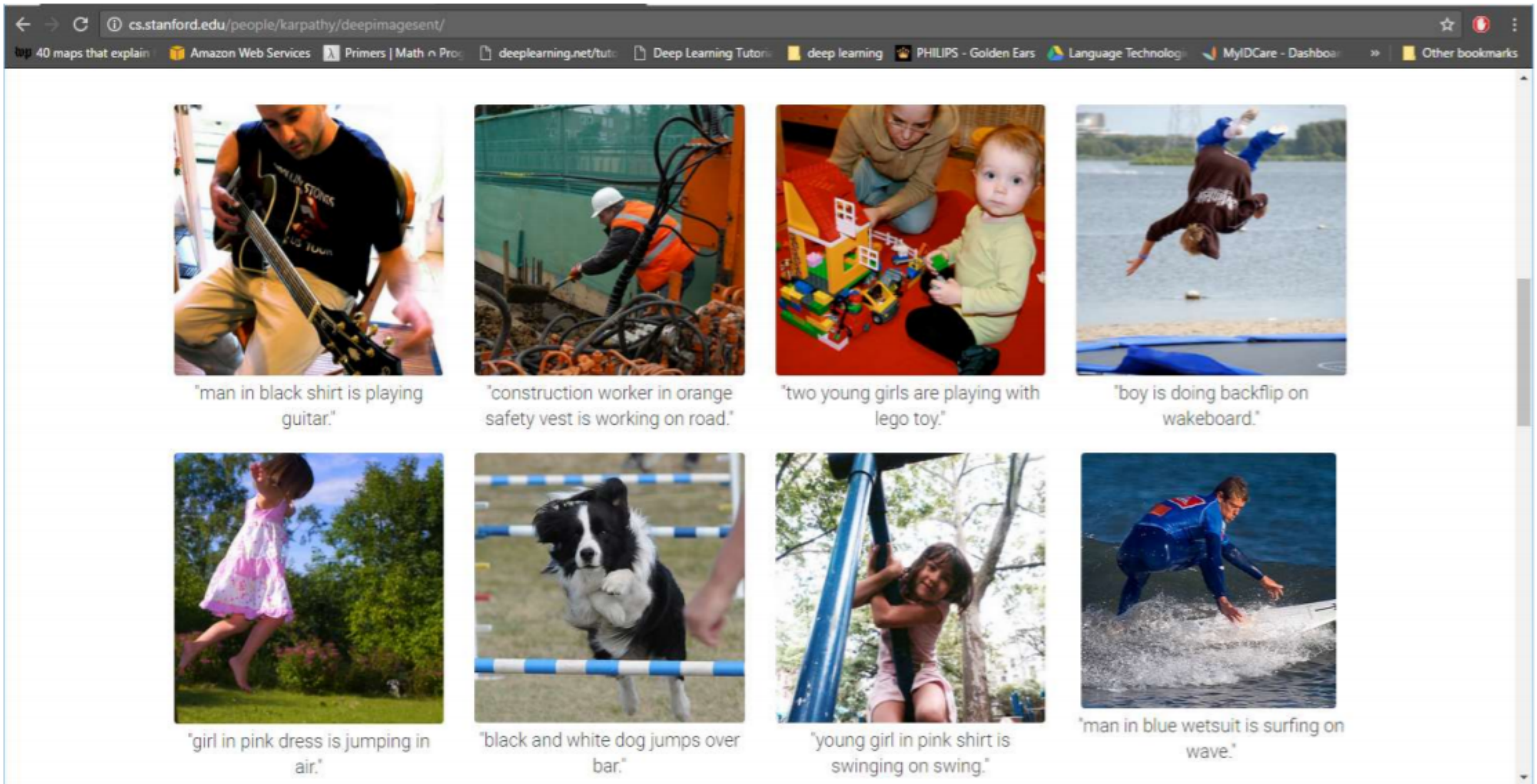
Success stories: Machine Translation



Success stories: Image segmentation



Success stories: Image captioning



References

- Deep Learning books
  - https://d2l.ai/ (D2L)
  - https://www.deeplearningbook.org/ (advanced)
- Stanford notes on deep learning
  - http://cs229.stanford.edu/notes/cs229-notes-deep_learning.pdf
- History of Deep Learning
  - https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html

Example

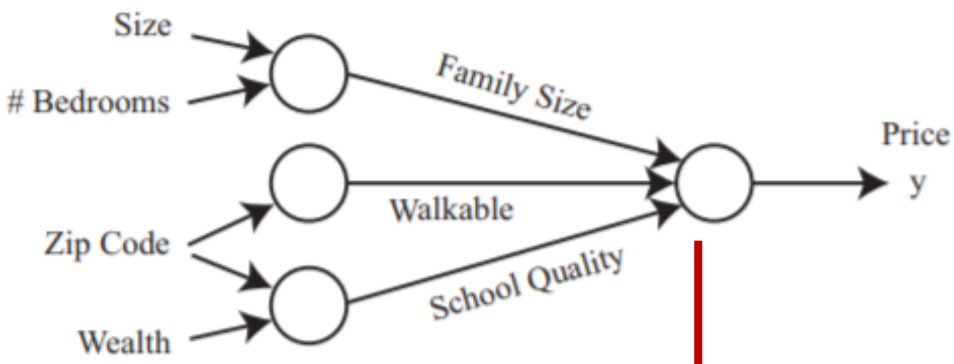
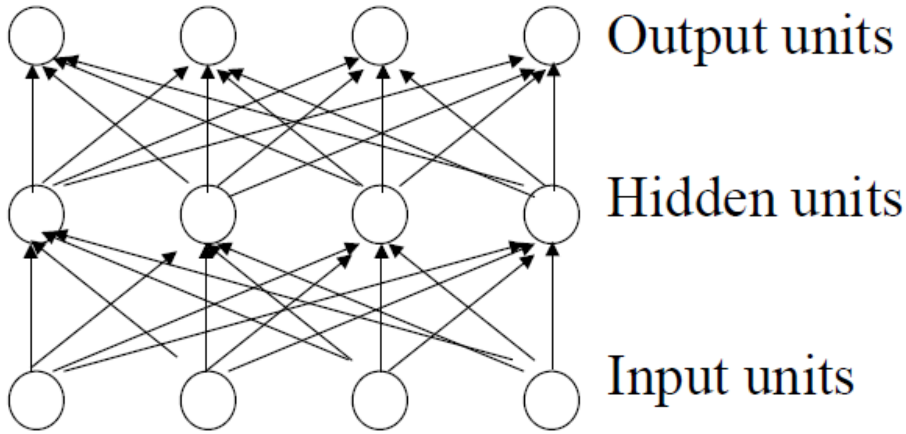


Figure 2: Diagram of a small neural network for predicting housing prices.

Intermediate Features

- Provide as input only training data: input and label
- Neural Networks automatically learn intermediate features!

Neural Networks



Layered feed-forward network

Training labels

Learned during training

Training data

- Neural networks are made up of **nodes** or **units**, connected by **links**
- Each link has an associated **weight** and **activation level**
- Each node has an **input function** (typically summing over weighted inputs), an **activation function**, and an **output**

Recall: The Perceptron

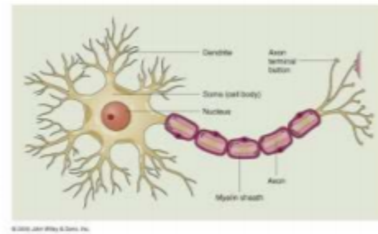
$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}) \text{ where } \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

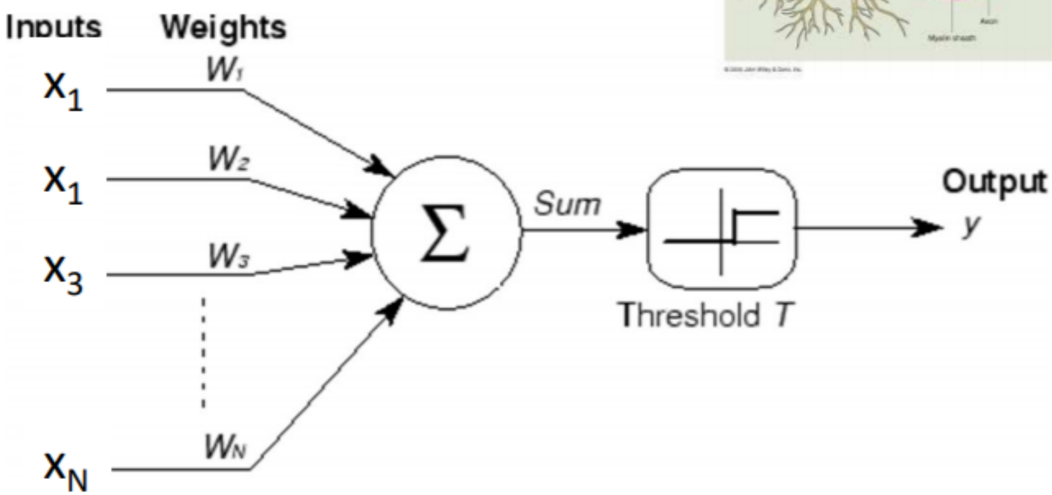
- The perceptron uses the following update rule each time it receives a new training instance $(\boldsymbol{x}^{(i)}, y^{(i)})$

$$\theta_j \leftarrow \theta_j - \frac{1}{2} \underbrace{\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right) - y^{(i)}\right)}_{\text{either 2 or -2}} x_j^{(i)}$$

  - If the prediction matches the label, make no change
  - Otherwise, adjust $\boldsymbol{\theta}$

Perceptron

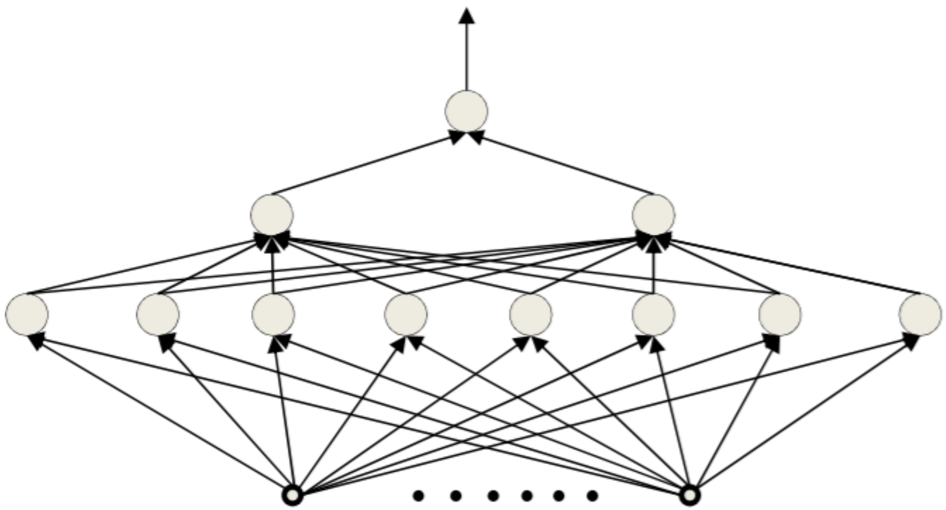


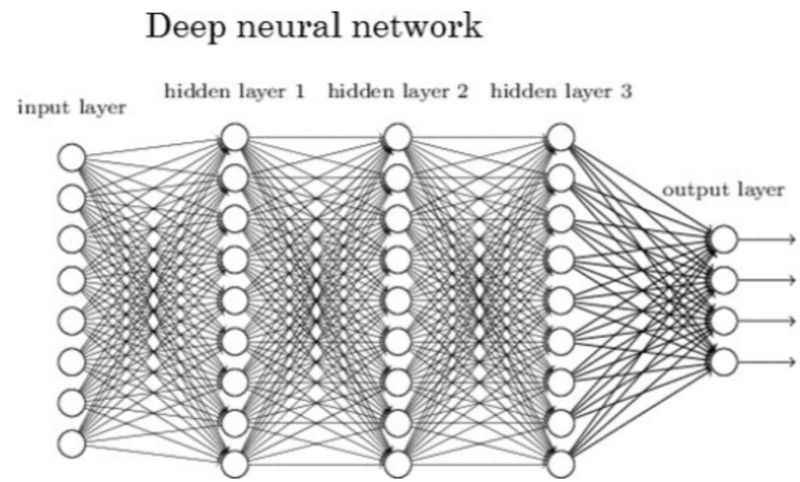


$$y = \begin{cases} 1 \; if \; \sum_i w_i x_i \geq T \\ 0 \; else \end{cases}$$

- A threshold unit
  - "Fires" if the weighted sum of inputs exceeds a threshold

Multi-Layer Perceptron



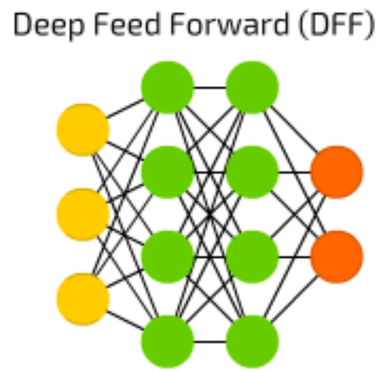


- A network of perceptrons
  - Generally “layered”

Neural Network Architectures

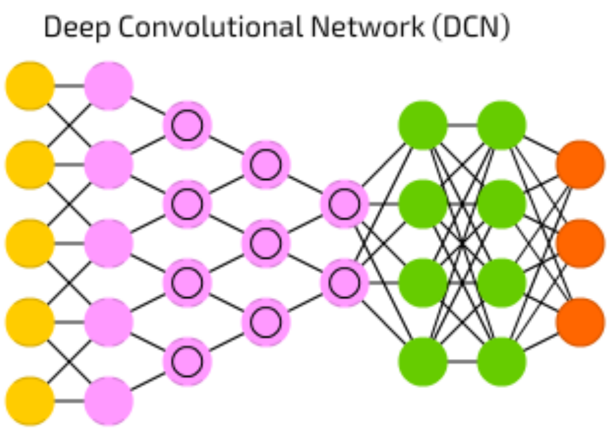
Feed-Forward Networks

- Neurons from each layer connect to neurons from next layer



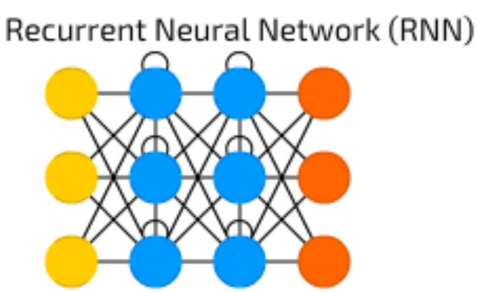
Convolutional Networks

- Includes convolution layer for feature reduction
- Learns hierarchical representations

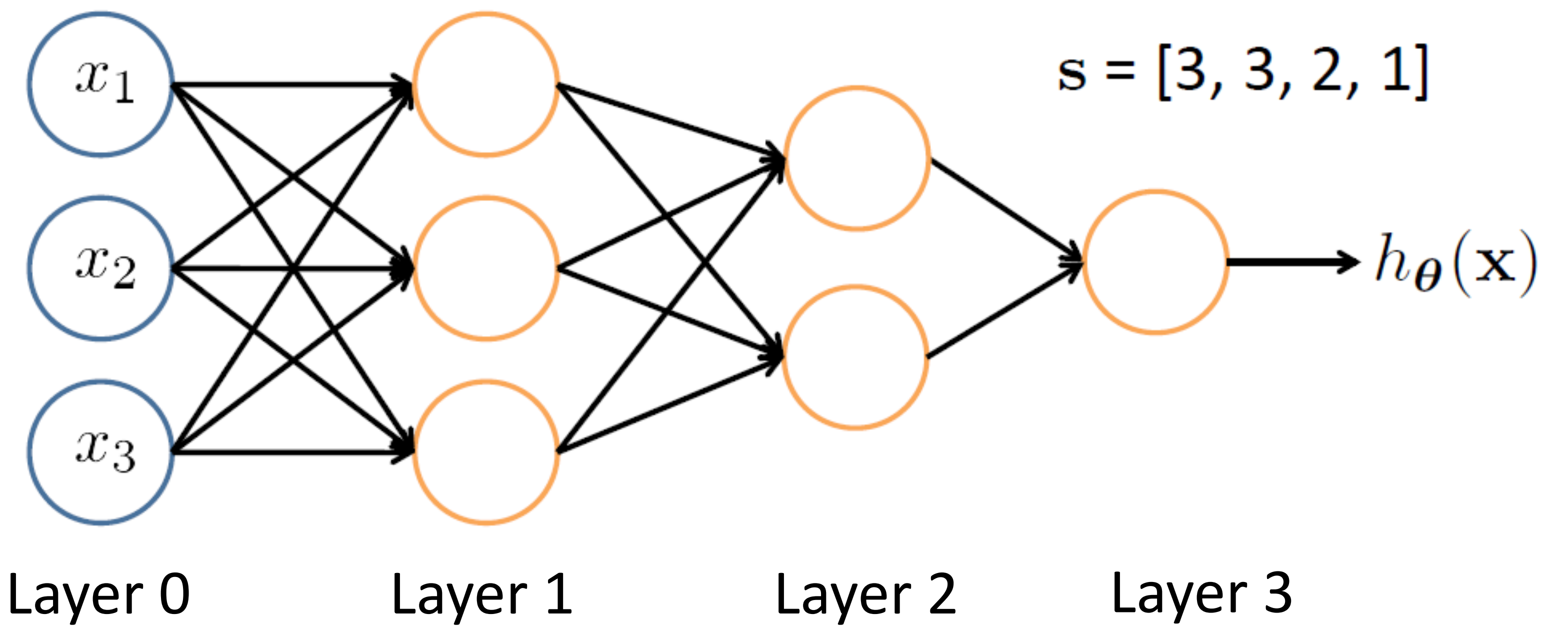


Recurrent Networks

- Keep hidden state
- Have cycles in computational graph



Feed-Forward Networks



L denotes the number of layers

$\mathbf{s} \in {\mathbb{N}^+}^L$ contains the numbers of nodes at each layer

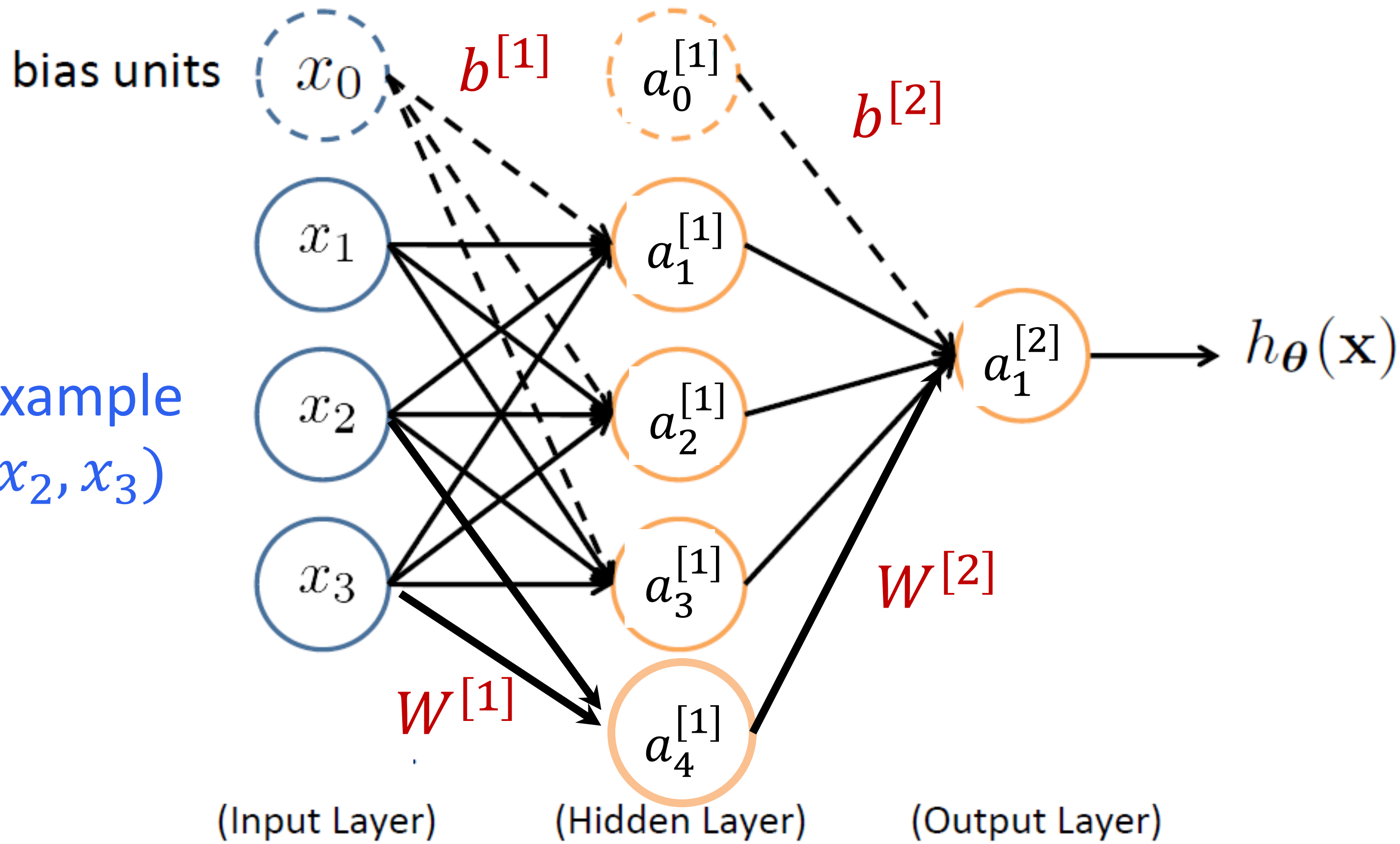
- Not counting bias units
- Typically, $s_0 = d$ (# input features) and $s_{L\text{-}1} = K$ (# classes)

Feed-Forward NN

- Hyper-parameters
  - Number of layers
  - Architecture (how layers are connected)
  - Number of hidden units per layer
  - Number of units in output layer
  - Activation functions
- Other
  - Initialization
  - Regularization

Feed-Forward Neural Network

Training example
$x = (x_1, x_2, x_3)$



Layer 0 Layer 1 Layer 2

No cycles

$\theta = (b^{[1]}, W^{[1]}, b^{[2]}, W^{[2]})$

Vectorization

$$z_1^{[1]} = W_1^{[1]T} x + b_1^{[1]} \quad \text{and} \quad a_1^{[1]} = g(z_1^{[1]})$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$z_4^{[1]} = W_4^{[1]T} x + b_4^{[1]} \quad \text{and} \quad a_4^{[1]} = g(z_4^{[1]})$$

$$\underbrace{\begin{bmatrix} z_1^{[1]} \\ \vdots \\ \vdots \\ z_4^{[1]} \end{bmatrix}}_{z^{[1]} \in \mathbb{R}^{4\times 1}} = \underbrace{\begin{bmatrix} — & W_1^{[1]T} & — \\ — & W_2^{[1]T} & — \\ & \vdots & \\ — & W_4^{[1]T} & — \end{bmatrix}}_{W^{[1]} \in \mathbb{R}^{4\times 3}} \underbrace{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}}_{x \in \mathbb{R}^{3\times 1}} + \underbrace{\begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ \vdots \\ b_4^{[1]} \end{bmatrix}}_{b^{[1]} \in \mathbb{R}^{4\times 1}}$$

$$z^{[1]} = W^{[1]} x + b^{[1]}$$

Linear

$$a^{[1]} = g(z^{[1]})$$

Non-Linear

Vectorization

Output layer

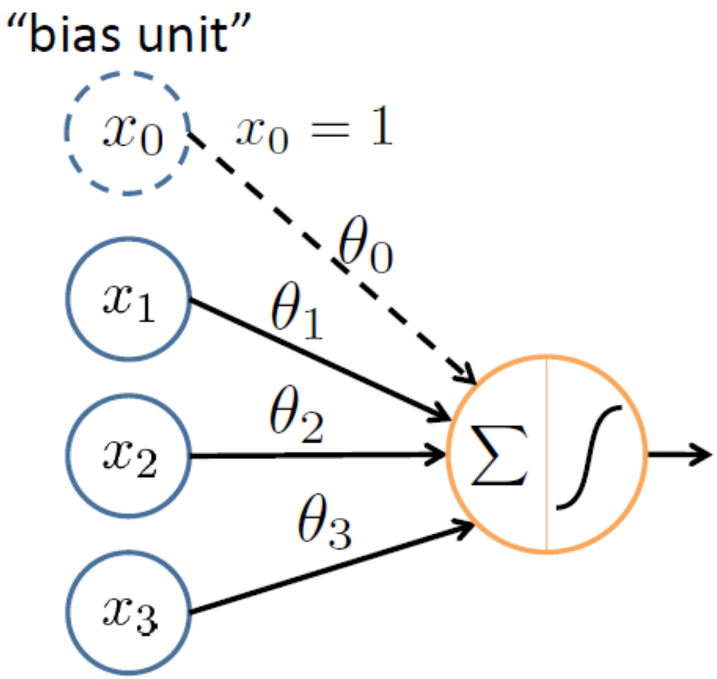
$$z_1^{[2]} = {W_1^{[2]}}^T a^{[1]} + b_1^{[2]} \quad \text{and} \quad a_1^{[2]} = g(z_1^{[2]})$$

$$\underbrace{z^{[2]}}_{1\times 1} = \underbrace{W^{[2]}}_{1\times 4}\underbrace{a^{[1]}}_{4\times 1} + \underbrace{b^{[2]}}_{1\times 1} \quad \text{and} \quad \underbrace{a^{[2]}}_{1\times 1} = g(\underbrace{z^{[2]}}_{1\times 1})$$

Hidden Units

- Layer 1
  - First hidden unit:
    - Linear: $z_1^{[1]} = \mathrm{W}_1^{[1]\,\mathrm{T}} x + \mathrm{b}_1^{[1]}$
    - Non-linear: $a_1^{[1]} = \mathrm{g}(\mathrm{z}_1^{[1]})$
  - …
  - Fourth hidden unit:
    - Linear: $z_4^{[1]} = \mathrm{W}_4^{[1]\,\mathrm{T}} x + \mathrm{b}_4^{[1]}$
    - Non-linear: $a_4^{[1]} = \mathrm{g}(\mathrm{z}_4^{[1]})$
- Terminology
  - $a_i^{[j]}$ - Activation of unit i in layer j
  - g - Activation function
  - $W^{[j]}$ - Weight vector controlling mapping from layer j-1 to j
  - $b^{[j]}$ - Bias vector from layer j-1 to j

Logistic Unit: A simple NN



$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

$$h_{\boldsymbol{\theta}}(\mathbf{x}) = g\left(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}\right) \;\; = g(w^T x + b)$$

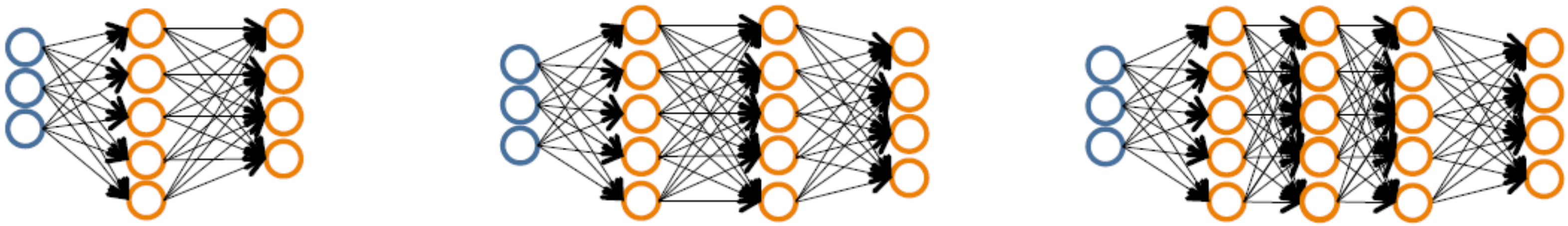
$$= \frac{1}{1 + e^{-\boldsymbol{\theta}^{\mathsf{T}}\mathbf{x}}}$$

Sigmoid (logistic) activation function: $g(z) = \dfrac{1}{1 + e^{-z}}$

No hidden layers

How to pick architecture?

Pick a network architecture (connectivity pattern between nodes)



- # input units = # of features in dataset
- # output units = # classes

Reasonable default: 1 hidden layer

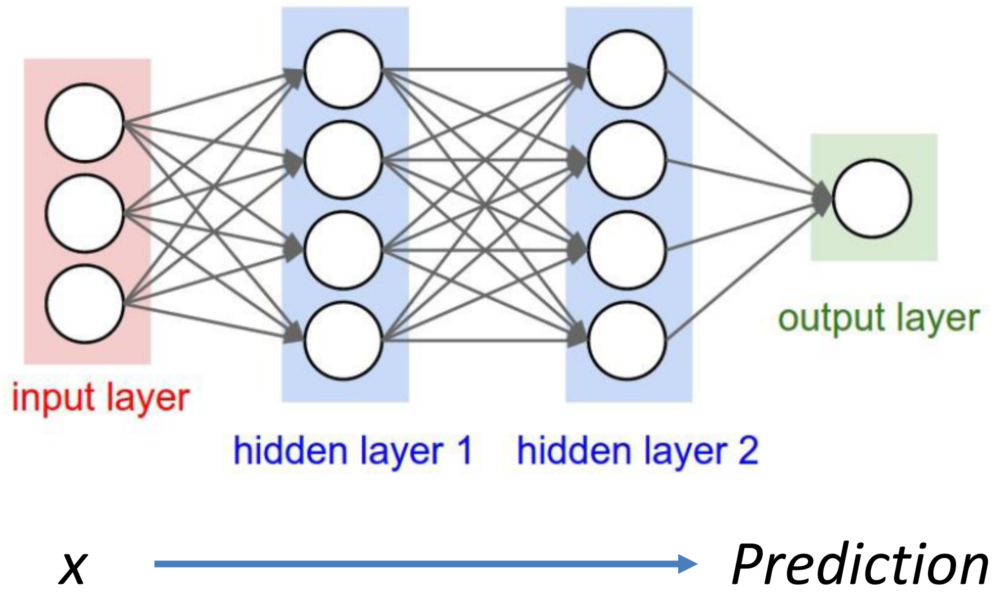
- or if >1 hidden layer, have same # hidden units in every layer (usually the more the better)

Training Neural Networks

- Input training dataset D
  - Number of features: d
  - Labels from K classes
- First layer has d+1 units (one per feature and bias)
- Output layer has K units
- Training procedure determines parameters that optimize loss function
  - Backpropagation
  - Learn optimal $W^{[i]}, b^{[i]}$ at layer *i*
- Testing done by forward propagation

Forward Propagation

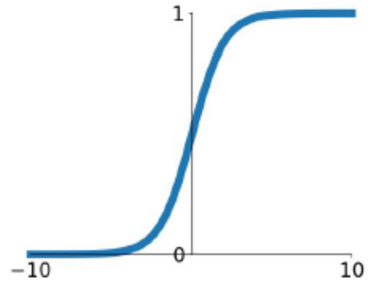
- The input neurons first receive the data features of the object. After processing the data, they send their output to the first hidden layer.
- The hidden layer processes this output and sends the results to the next hidden layer.
- This continues until the data reaches the final output layer, where the output value determines the object's classification.
- This entire process is known as Forward Propagation, or Forward prop.



Activation Functions

Sigmoid

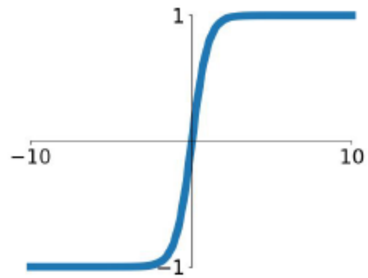
$\sigma(x) = \frac{1}{1+e^{-x}}$



Binary Classification

tanh

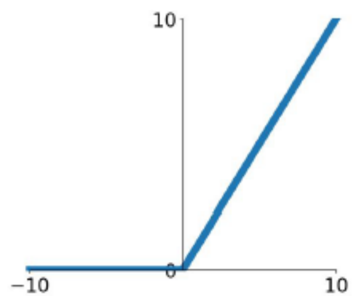
$\tanh(x)$



Regression

ReLU

$\max(0, x)$



Intermediary layers

Why Non-Linear Activations?

- Assume g is linear: $\mathrm{g}(z) = Uz$
  - At layer 1: $z^{[1]} = \mathrm{W}^{[1]}x + \mathrm{b}^{[1]}$
  - $a^{[1]} = \mathrm{g}\left(\mathrm{z}^{[1]}\right) = \mathrm{Uz}^{[1]} = \mathrm{UW}^{[1]}\mathrm{x} + \mathrm{Ub}^{[1]}$
- Layer 2:
  - $a^{[2]} = \mathrm{g}\left(\mathrm{z}^{[2]}\right) = \mathrm{Uz}^{[2]} = \mathrm{UW}^{[2]}\mathrm{a}^{[1]} + \mathrm{Ub}^{[2]} =$

  $= \mathrm{UW}^{[2]}\,\mathrm{UW}^{[1]}x + \mathrm{UW}^{[2]}\mathrm{Ub}^{[1]} + \mathrm{Ub}^{[2]}$
- Last layer
  - Output is linear in input!
  - Then NN will only learn linear functions

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
  - Andrew Moore
  - Yann Lecun
- Thanks!