DS 4400

Machine Learning and Data Mining I
Spring 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

March 18 2021

Outline

- Deep Learning
  - Motivation
  - Goals
- Deep Learning as representation learning
- Perceptron and its limitations
- Feed-forward neural networks

SVM Classifier

- Support Vector Classifier (SVC, linear SVM)
  - SVC classifier is linear combination of dot product between testing point and support vectors
  - $h(z) = \theta_0 + \sum_{i \in S} \alpha_i < z, x_i >$
  - Two methods to train:
    - Solve directly optimization problem to find maximum margin
    - Gradient descent with hinge loss objective
- SVM classifier
  - Select a kernel function K
  - SVM classifier is linear combination of kernel between testing point and support vectors
  - $h(z) = \theta_0 + \sum_{i \in S} \alpha_i K(z, x_i)$

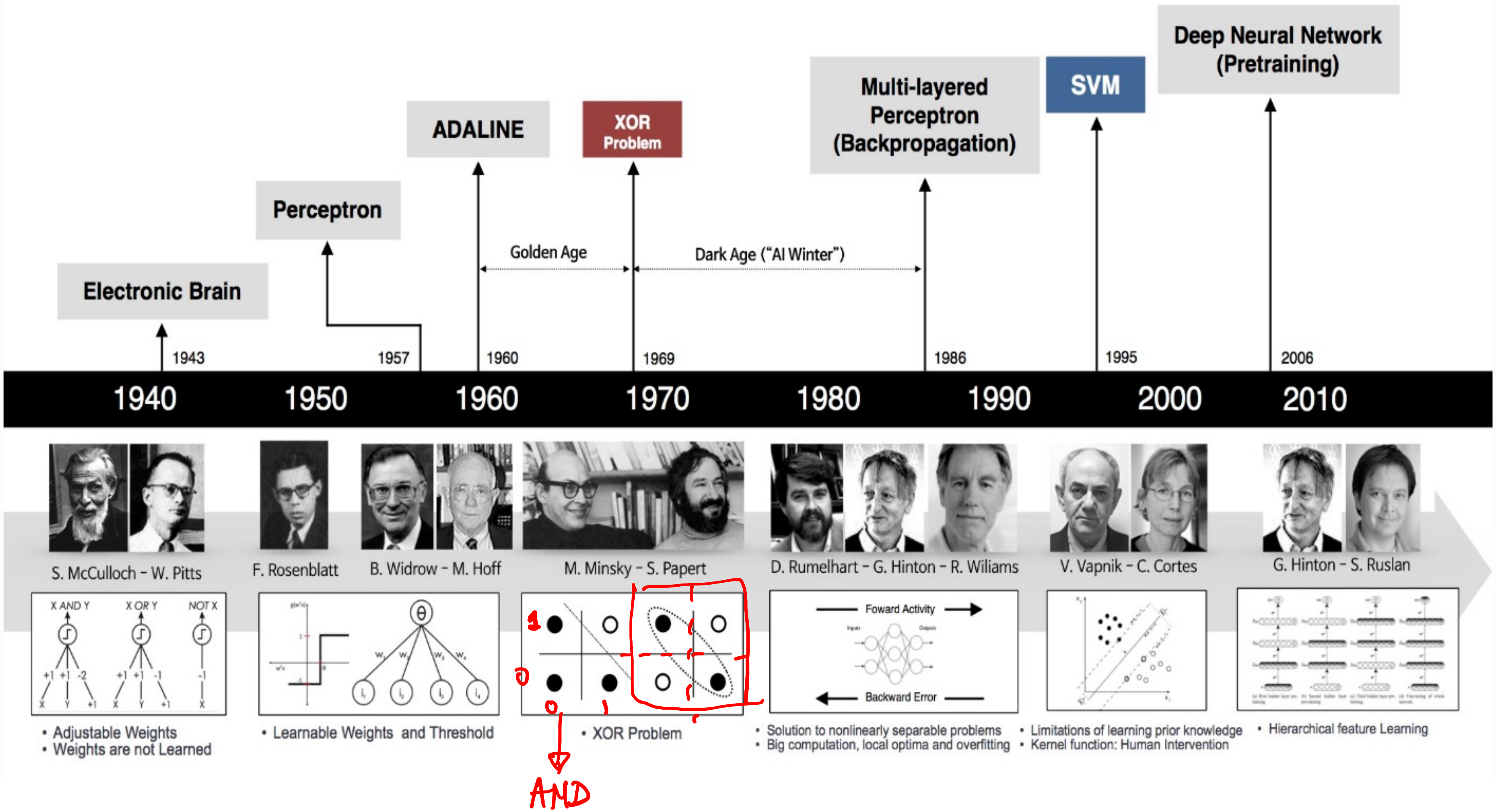
1) Gaussian (RBF) ; 2) Polynomial

σ var. deg p

Comparing SVM with other classifiers

- SVM is resilient to outliers
  - Similar to Logistic Regression
  - LDA or kNN are not
- SVM can be trained with Gradient Descent
  - Hinge loss cost function
- Supports regularization
  - Can add penalty term (ridge or Lasso) to cost function
- Linear SVM is most similar to Logistic Regression

History of Deep Learning

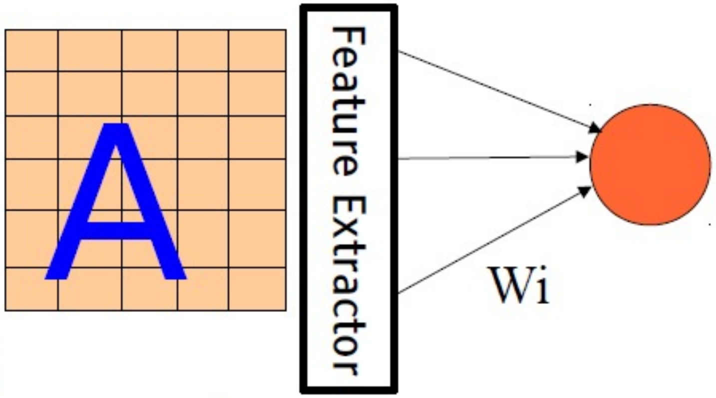


References

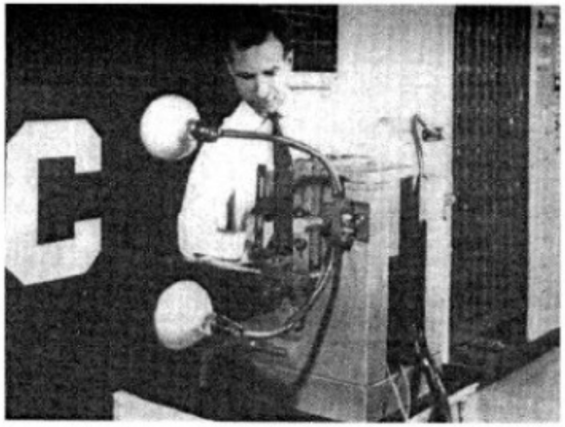
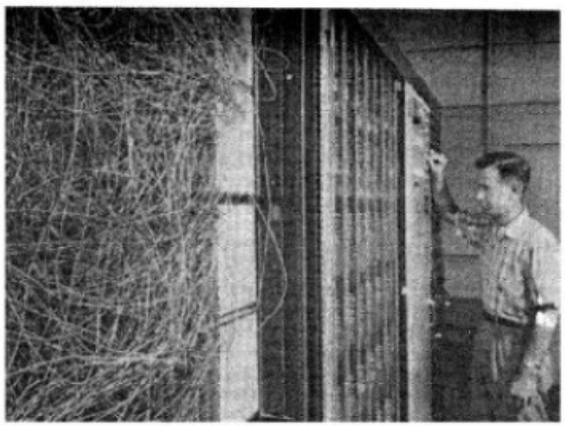
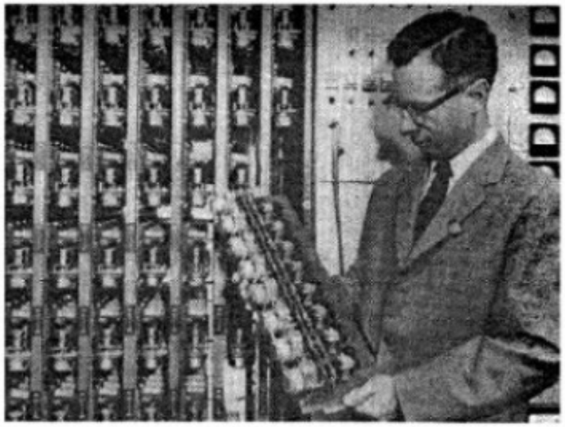
- Deep Learning books
  - https://d2l.ai/ (D2L)
  - https://www.deeplearningbook.org/ (advanced)
- Stanford notes on deep learning
  - http://cs229.stanford.edu/summer2020/cs229-notes-deep_learning.pdf

Before 2013

- **The first learning machine: the Perceptron**
  - Built at Cornell in 1960
- **The Perceptron was a linear classifier on top of a simple feature extractor**
- **The vast majority of practical applications of ML today use glorified linear classifiers or glorified template matching.**
- **Designing a feature extractor requires considerable efforts by experts.**



$$y = sign\left(\sum_{i=1}^{N} W_i F_i(X) + b\right)$$

Deep Learning

- **The traditional model of pattern recognition (since the late 50's)**
  - Fixed/engineered features (or fixed kernel) + trainable classifier

hand-crafted Feature Extractor → "Simple" Trainable Classifier →

Traditional ML

- End-to-end learning / Feature learning / Deep learning

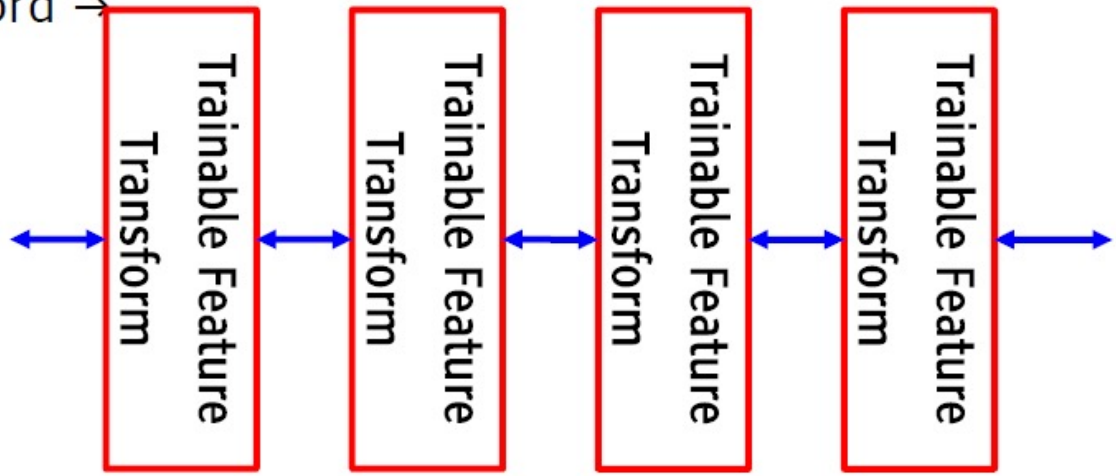
→ Trainable Feature Extractor → Trainable Classifier →

Input.

Automated

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 →



LAYERS

Learning Representations

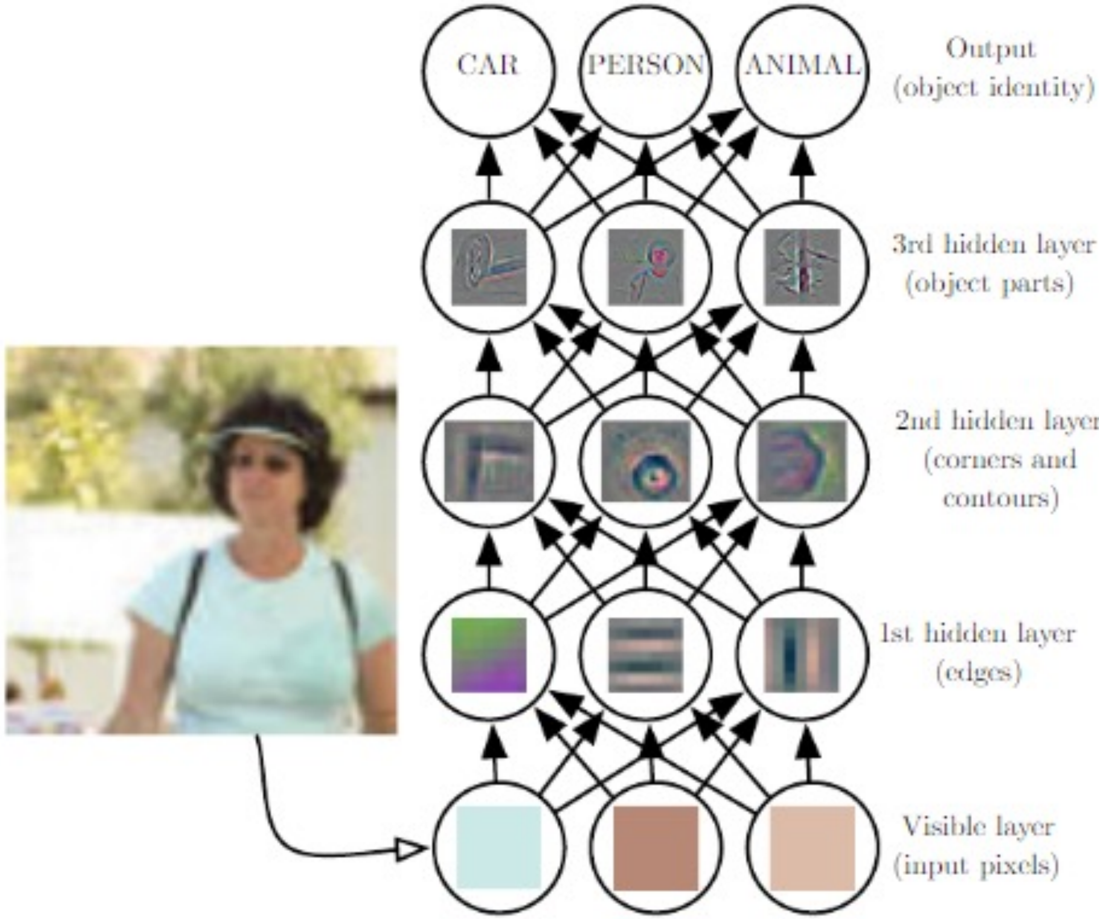
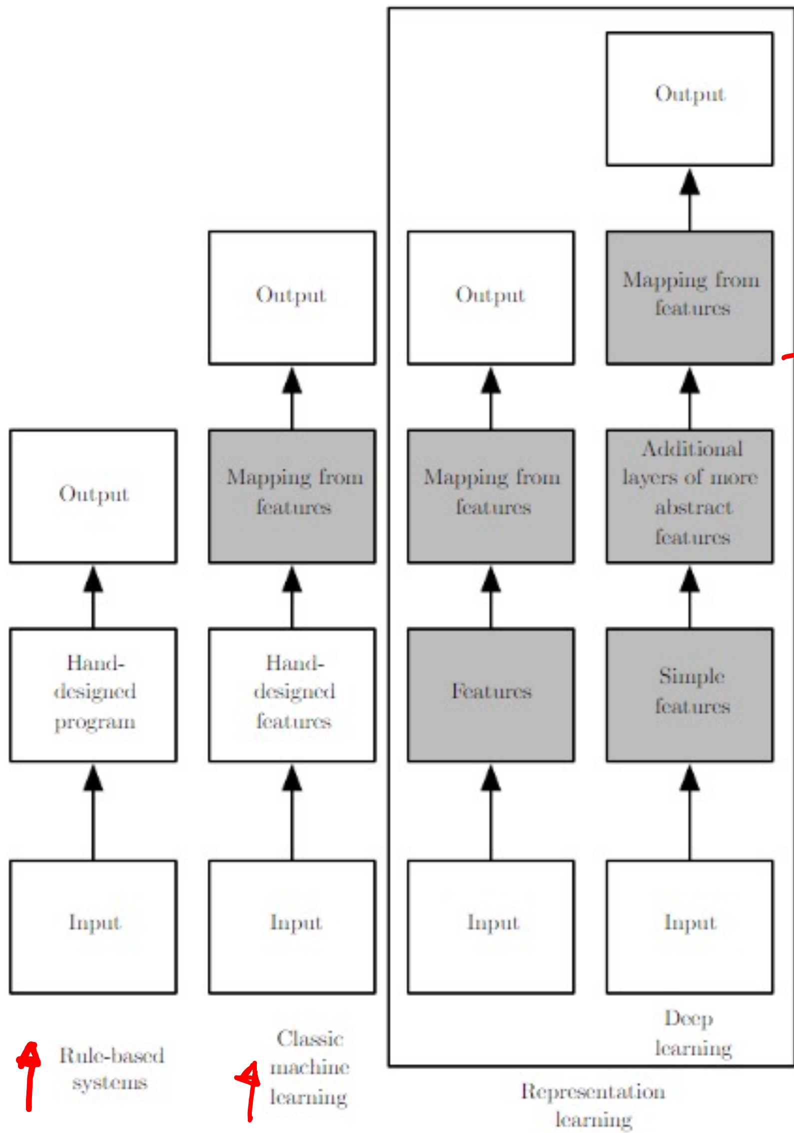


IMAGE PIXEL

Deep Learning addresses the problem of learning hierarchical representations

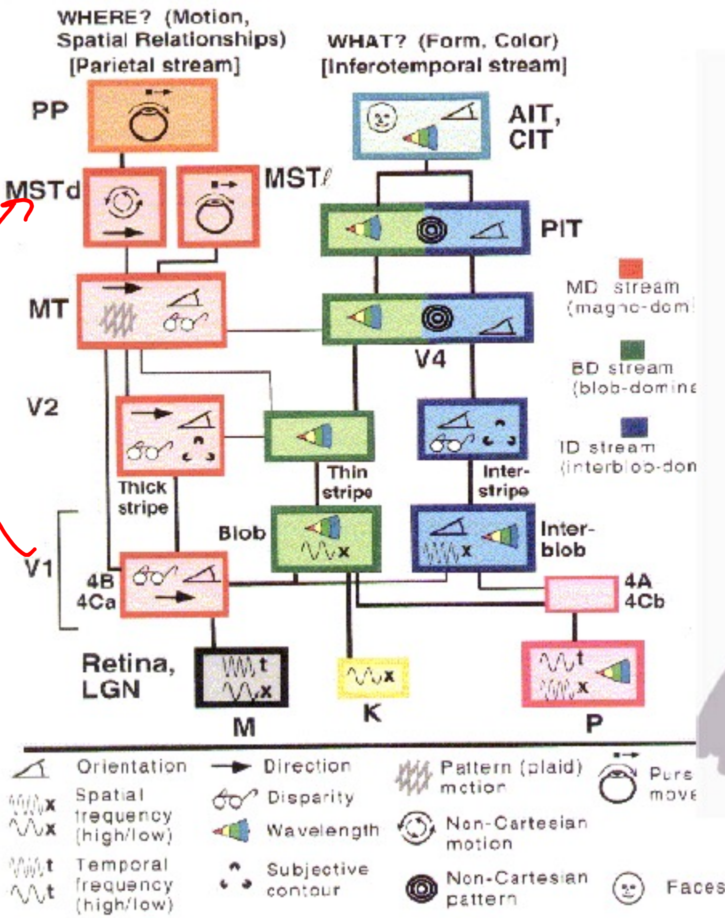
Deep Learning vs Traditional Learning

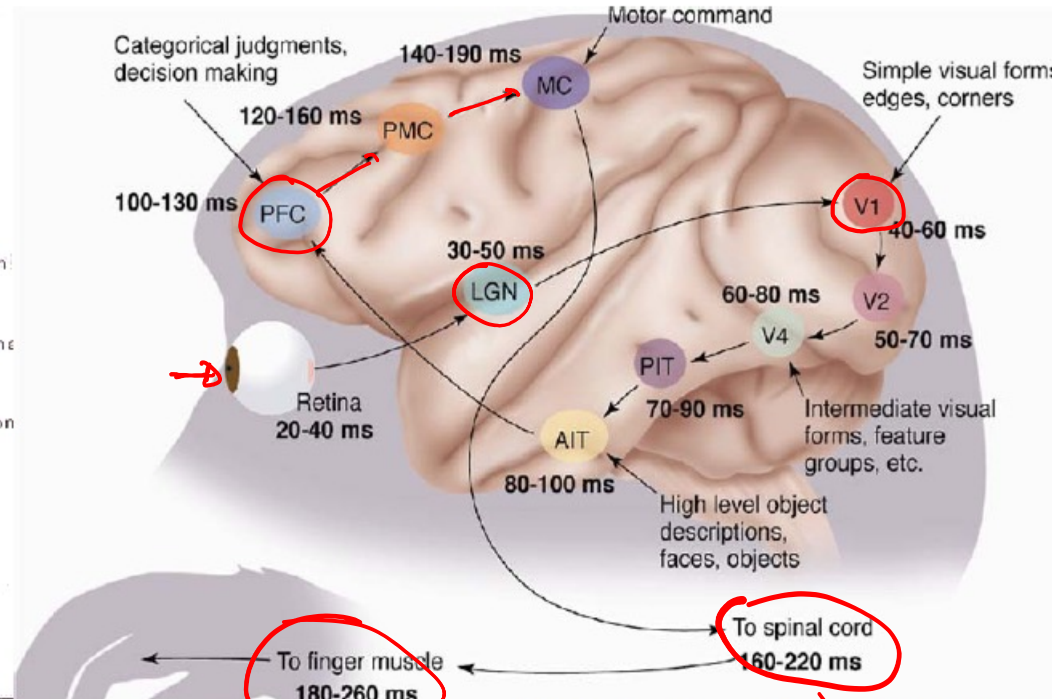


DEEP LEARNING

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]

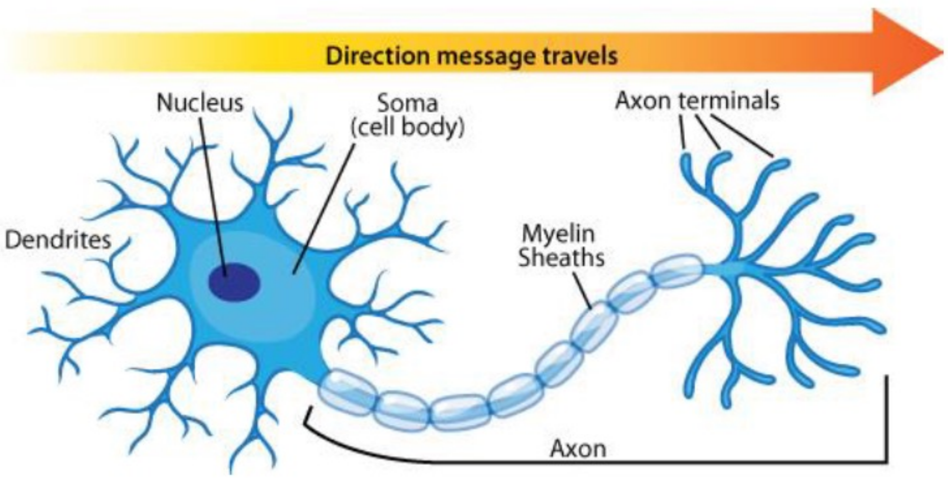
[Gallant & Van Essen]

Neural Function

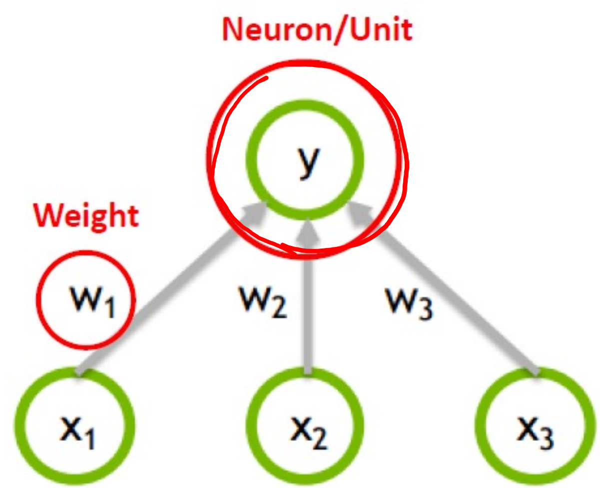
- Brain function (thought) occurs as the result of the firing of **neurons**
- Neurons connect to each other through **synapses**, which propagate **action potential** (electrical impulses) by releasing **neurotransmitters**
  - Synapses can be **excitatory** (potential-increasing) or **inhibitory** (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!

Analogy to Human Brain

Human Brain



Biological Neuron



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

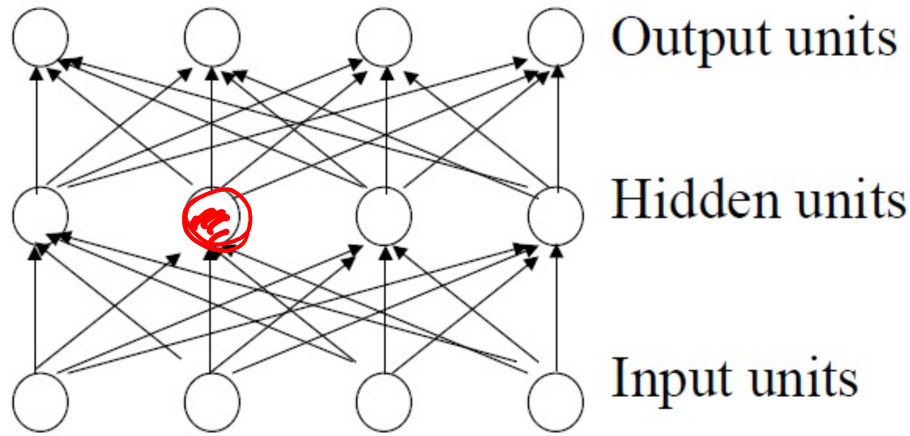
$$\boxed{F(x)=\max(0,x)}$$ ReLU

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

Neural Networks



LABELS

LEARNED DURING TRAINING

DATA (PIXELS)

Layered feed-forward network

- 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**

Example

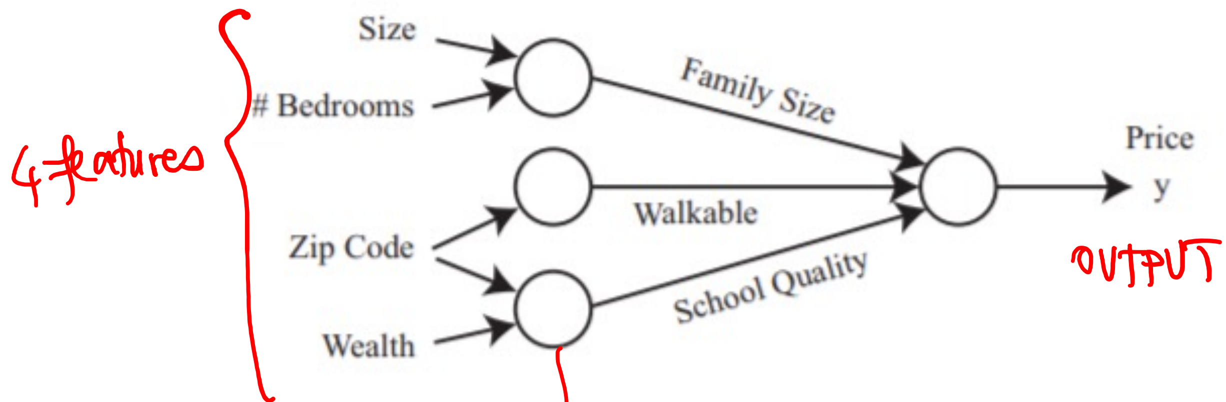


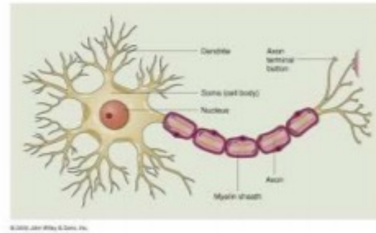
Figure 2: Diagram of a small neural network for predicting housing prices.

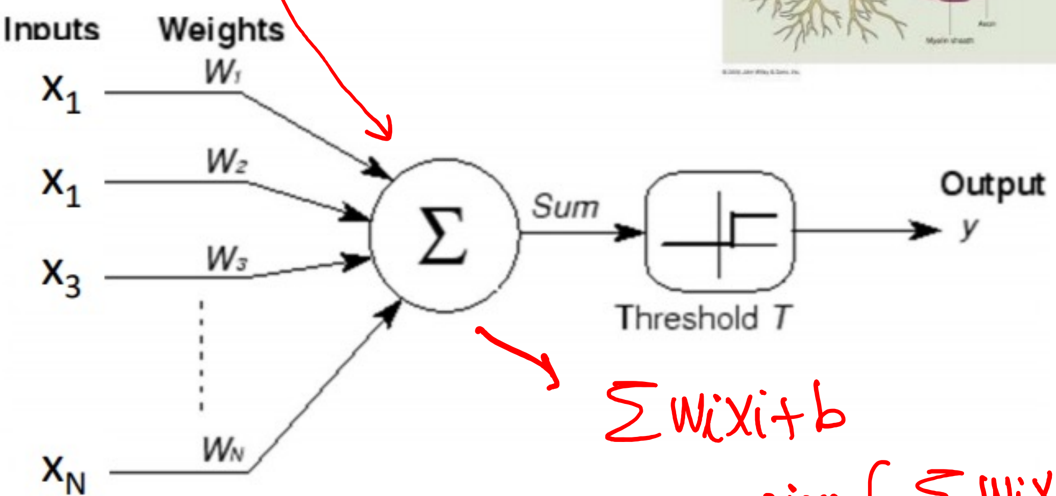
Intermediate Features: Family size, Walkable, School quality

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

Perceptron

θ b





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

$\sum w_i x_i + b$

$sign(\sum w_i x_i + b) \geq 0 \to y=1$

$\to y=0$

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

$$\theta = \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_d \end{bmatrix}$$

$\theta^T x > 0$

The Perceptron

• θ random

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

$\in \{-1, 1\}$

- The perceptron uses the following update rule each time it receives a new training instance (x_i, y_i) ; $y_i \in \{-1, 1\}$

$$\boxed{\theta_j \leftarrow \theta_j - \frac{1}{2}(h_\theta(x_i) - y_i)x_{ij}}$$

for $j = 1, \dots, d$.

  – If the prediction matches the label, make no change

  – Otherwise, adjust $\boldsymbol{\theta}$

1) LINEAR REG: $h_\theta(x) = \theta^T x$; 2) LOGISTIC REG $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$

The Perceptron

$h_\theta(x_i) = \mathrm{sign}(\theta^T x_i)$

- The perceptron uses the following update rule each time it receives a new training instance (x_i, y_i)

$$\boxed{\theta_j \leftarrow \theta_j - \frac{1}{2}\left(h_\theta(x_i) - y_i\right)x_{ij}}$$

1) $h_\theta(x_i) = y_i \Rightarrow$ NO UPDATE

2) $h_\theta(x_i) = 1,\ y_i = -1 \Rightarrow \theta_j \leftarrow \theta_j - x_{ij}$

3) $h_\theta(x_i) = -1,\ y_i = 1 \Rightarrow \theta_j \leftarrow \theta_j + x_{ij}$

$\theta_j \leftarrow \theta_j + y_i x_{ij}, \quad j = \overline{1,d}$

$$\boxed{\theta \leftarrow \theta + y_i x_i}$$

Online Perceptron

Let $\theta \leftarrow$[0,0,…,0]
Repeat:
  Receive training example (x_i, y_i)
  If $y_i\theta^T x_i \leq 0$ // prediction is incorrect
    $\theta \leftarrow \theta + y_i\, x_i$
Until stopping condition

Online learning – the learning mode where the model update is performed each time a single observation is received

Batch learning – the learning mode where the model update is performed after observing the entire training set

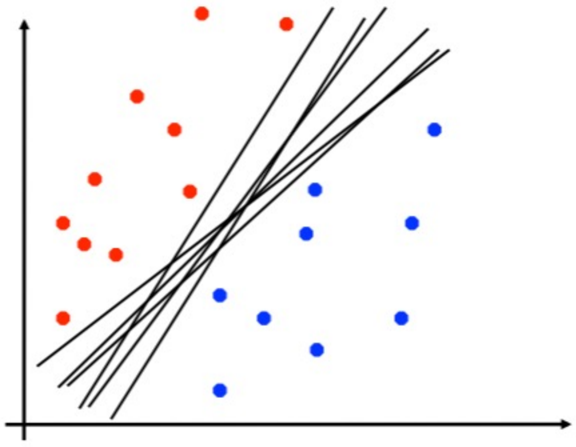
Batch Perceptron

Let $\theta \leftarrow$[0,0,…,0]
Repeat:
    $\Delta = [0,0,\ldots,0]$
    For $i = 1$ to N // Consider all training examples
        If $y_i\theta^T x_i \leq 0$ // Prediction is incorrect
        $\Delta \leftarrow \Delta + y_i\, x_i.$ // Accumulate all errors
    $\theta \leftarrow \theta + \boxed{\frac{\Delta}{N}}$ // Parameter update rule
Until stopping condition

- Guaranteed to find separating hyperplane if data is linearly separable

Perceptron Limitations

- Is dependent on starting point
- It could take many steps for convergence
- Perceptron can overfit
  - Move the decision boundary for every example



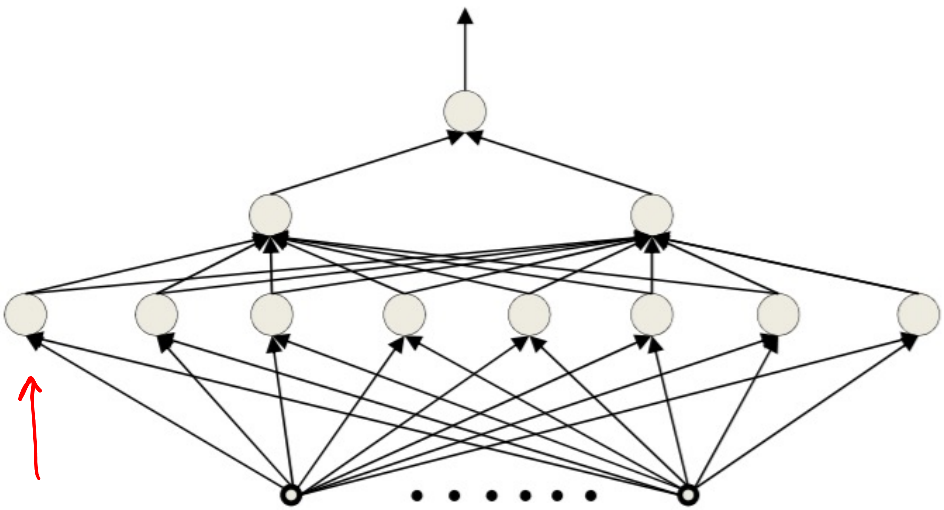
Which of this is optimal?

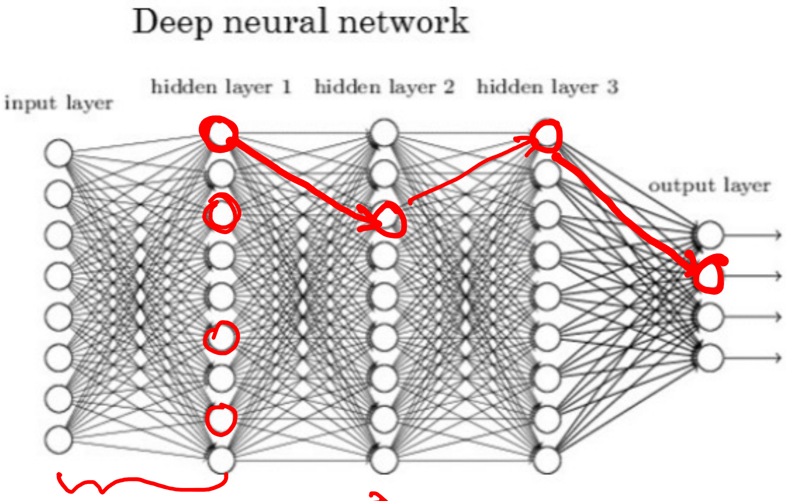
History of Perceptrons

- They were popularised by Frank Rosenblatt in the early 1960's.
  - They appeared to have a very powerful learning algorithm.
  - Lots of grand claims were made for what they could learn to do.
- In 1969, Minsky and Papert published a book called "Perceptrons" that analysed what they could do and showed their limitations.
  - Many people thought these limitations applied to all neural network models.
- The perceptron learning procedure is still widely used today for tasks with enormous feature vectors that contain many millions of features.

They are the basic building blocks for
Deep Neural Networks

Multi-Layer Perceptron



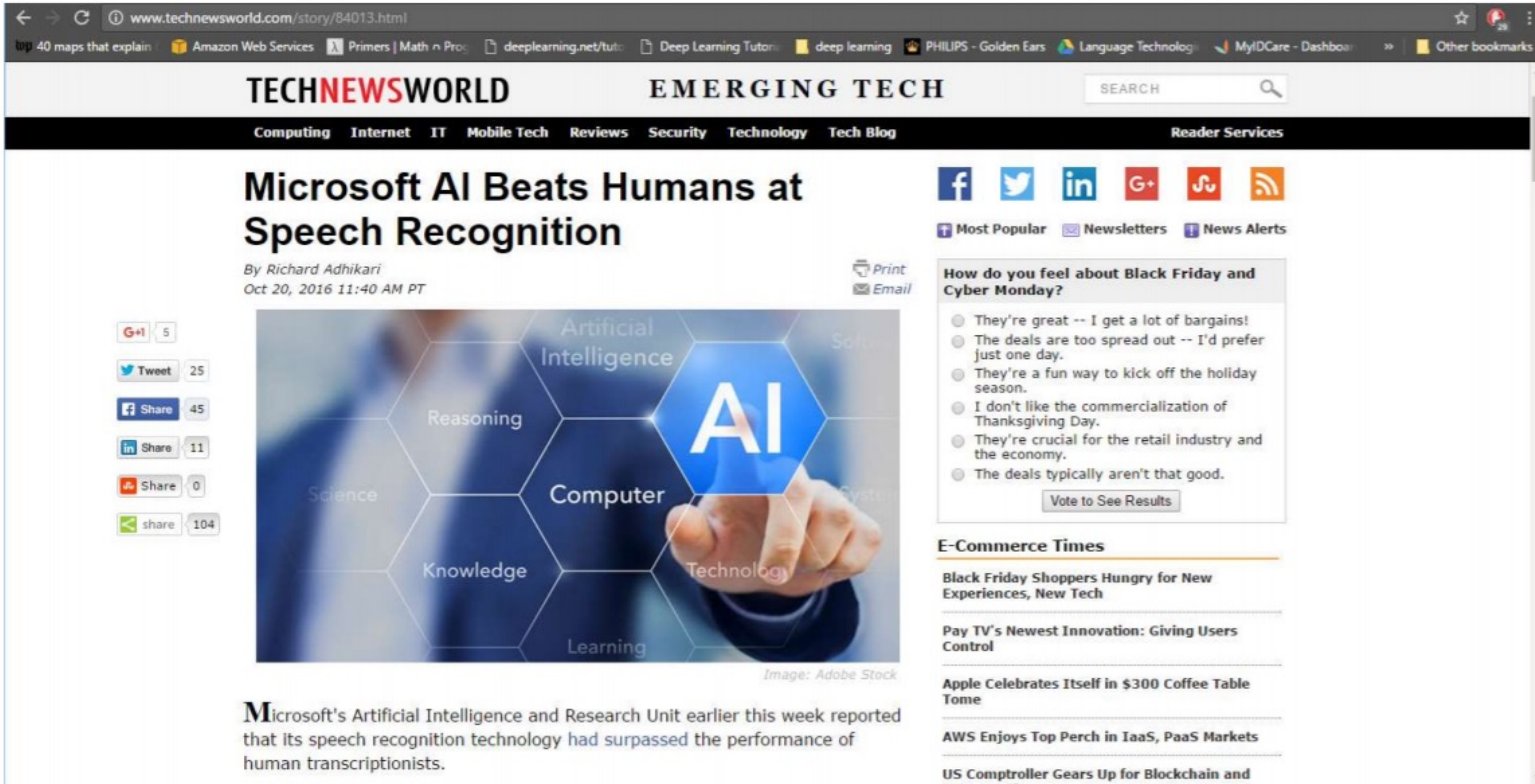


LINEAR + NON-LINEAR

- A network of perceptrons
  - Generally "layered"

Deep Learning Applications

Success stories: Speech recognition



www.technewsworld.com/story/84013.html

40 maps that explain · Amazon Web Services · Primers | Math ∩ Proc · deeplearning.net/tuto · Deep Learning Tutori · deep learning · PHILIPS - Golden Ears · Language Technolog · MyIDCare - Dashboar · Other bookmarks

TECHNEWSWORLD EMERGING TECH SEARCH

Computing · Internet · IT · Mobile Tech · Reviews · Security · Technology · Tech Blog · Reader Services

Microsoft AI Beats Humans at Speech Recognition

By Richard Adhikari
Oct 20, 2016 11:40 AM PT

Print
Email

G+1 5 · Tweet 25 · Share 45 · Share 11 · Share 0 · share 104

Artificial Intelligence · AI · Reasoning · Computer · Science · Knowledge · Technology · Learning

Image: Adobe Stock

Microsoft's Artificial Intelligence and Research Unit earlier this week reported that its speech recognition technology had surpassed the performance of human transcriptionists.

Most Popular · Newsletters · News Alerts

How do you feel about Black Friday and Cyber Monday?

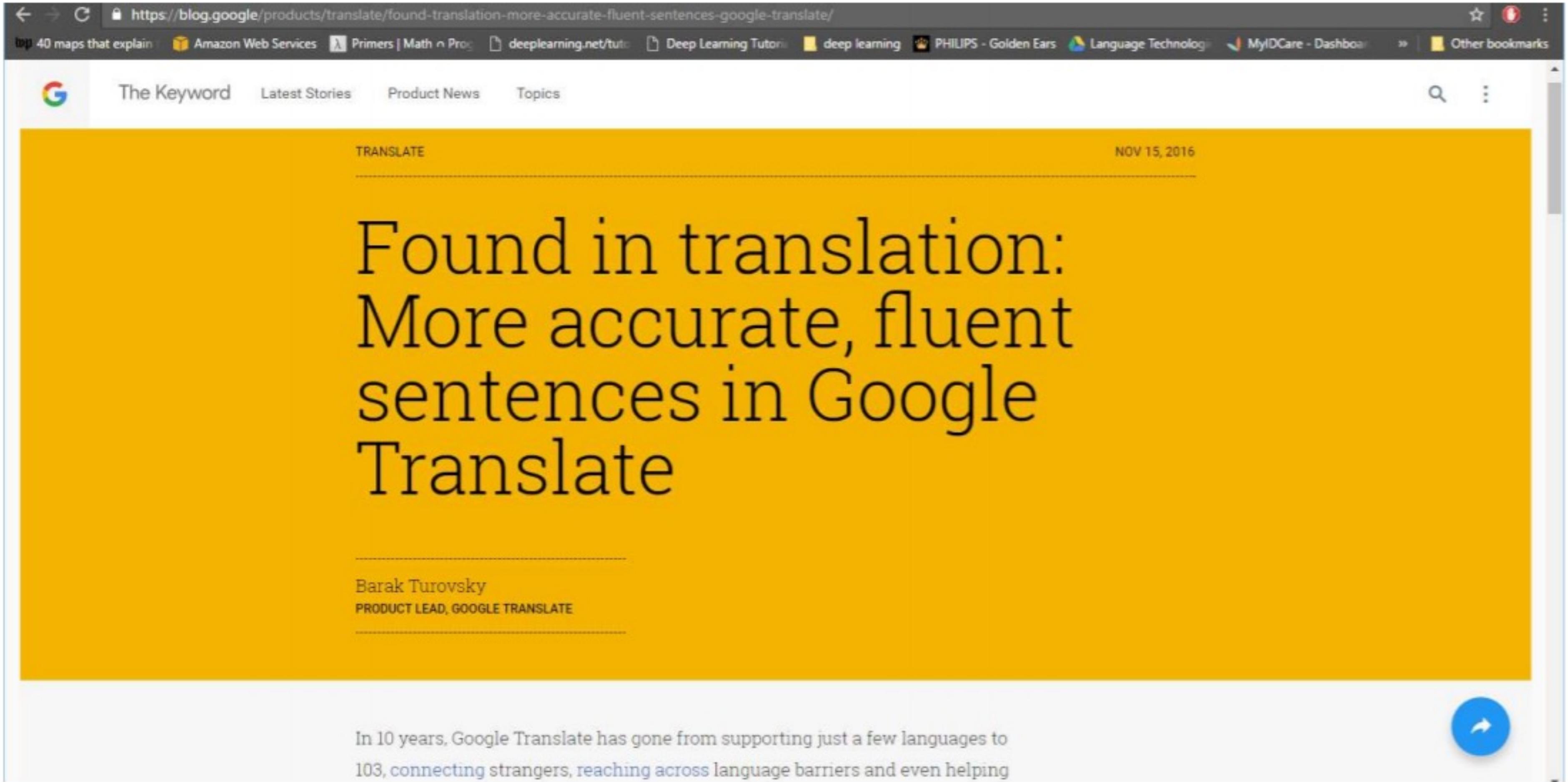
- They're great -- I get a lot of bargains!
- The deals are too spread out -- I'd prefer just one day.
- They're a fun way to kick off the holiday season.
- I don't like the commercialization of Thanksgiving Day.
- They're crucial for the retail industry and the economy.
- The deals typically aren't that good.

Vote to See Results

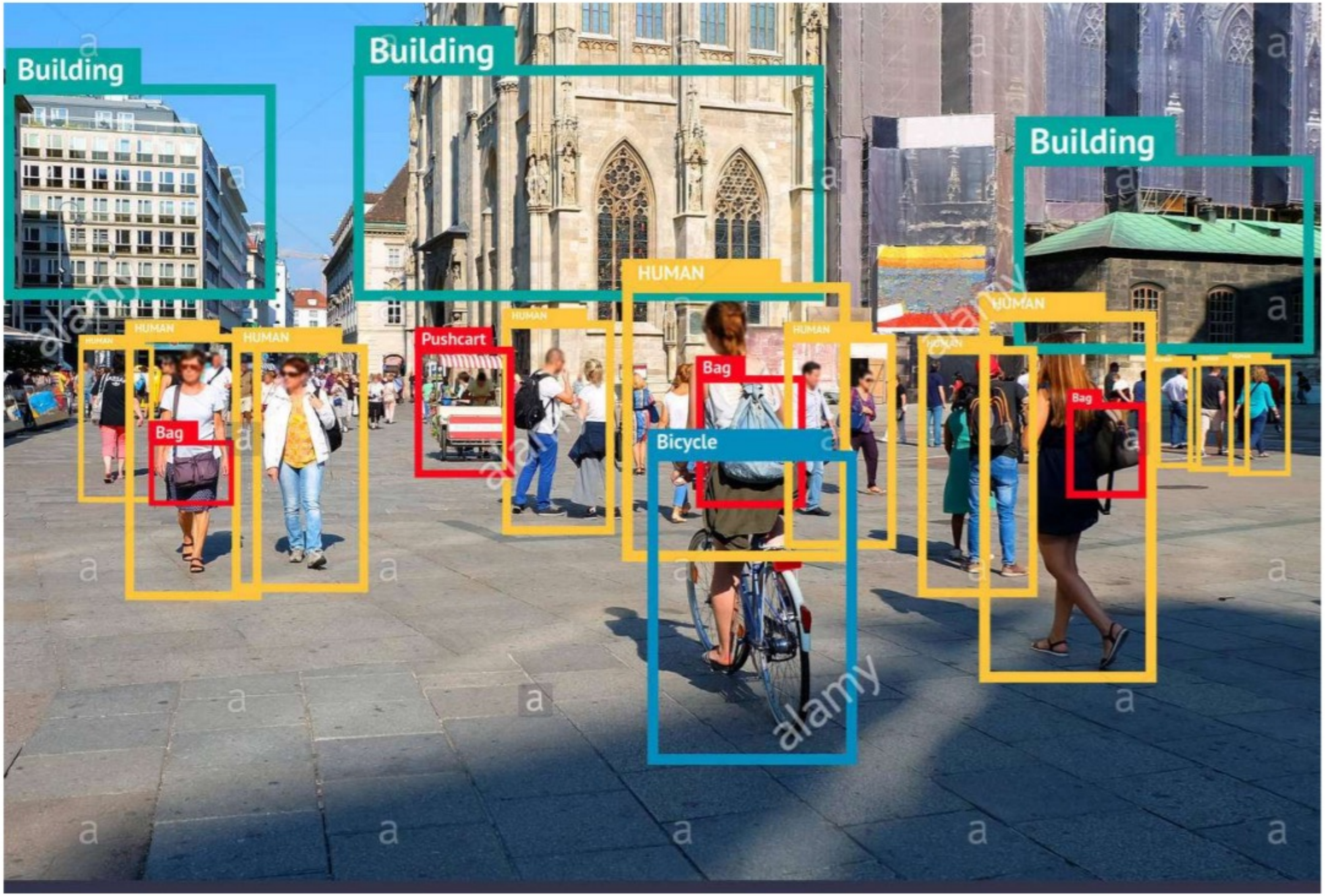
E-Commerce Times

- Black Friday Shoppers Hungry for New Experiences, New Tech
- Pay TV's Newest Innovation: Giving Users Control
- Apple Celebrates Itself in $300 Coffee Table Tome
- AWS Enjoys Top Perch in IaaS, PaaS Markets
- US Comptroller Gears Up for Blockchain and

Success stories: Machine Translation



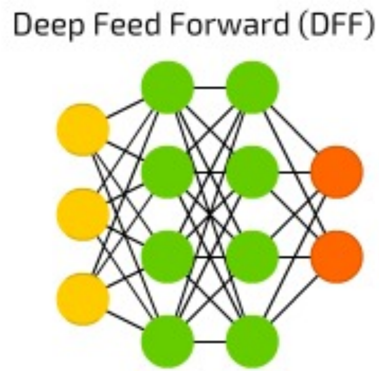
Success stories: Image segmentation



Neural Network Architectures

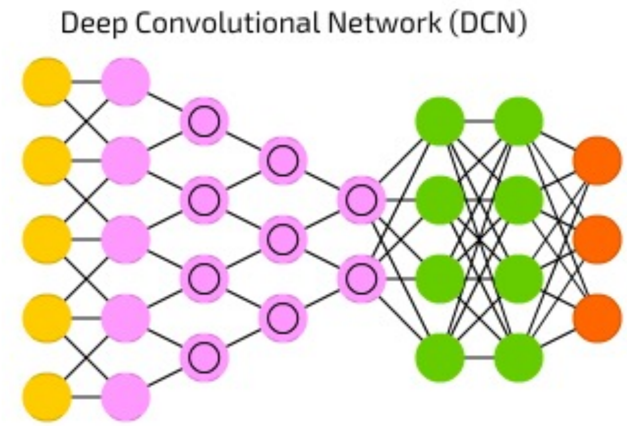
Feed-Forward Networks

- Neurons from each layer connect to neurons from next layer



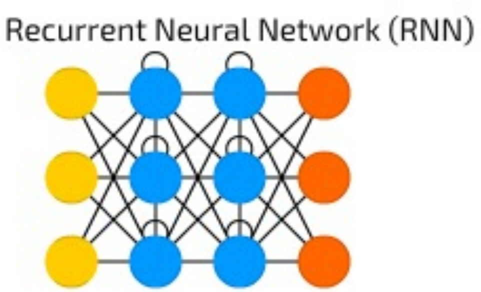
Convolutional Networks

- Includes convolution layer for feature reduction
- Learns hierarchical representations

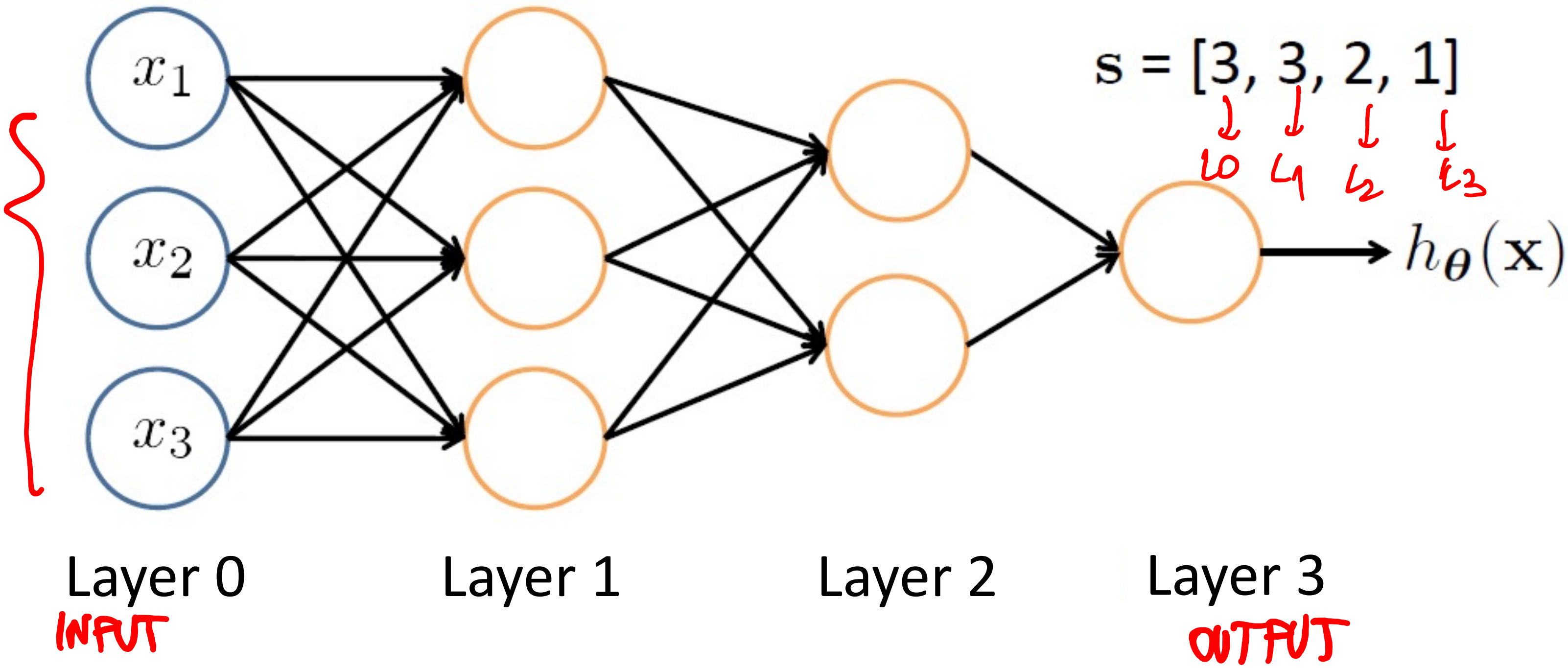


Recurrent Networks

- Keep hidden state
- Have cycles in computational graph



Feed-Forward Networks



L denotes the number of layers

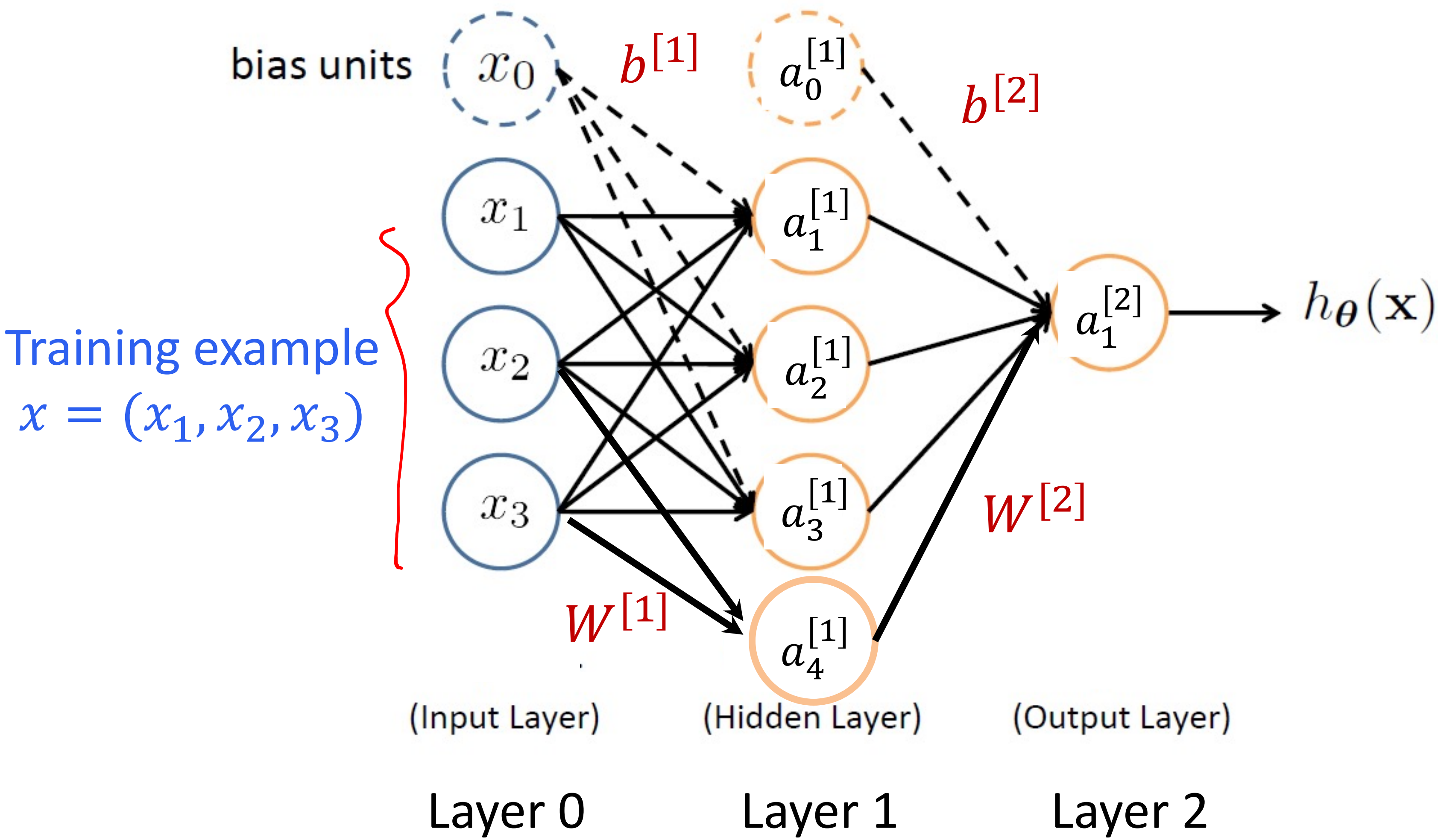
$\mathbf{s} \in {\mathbb{N}^+}^L$ contains the numbers of nodes at each layer

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

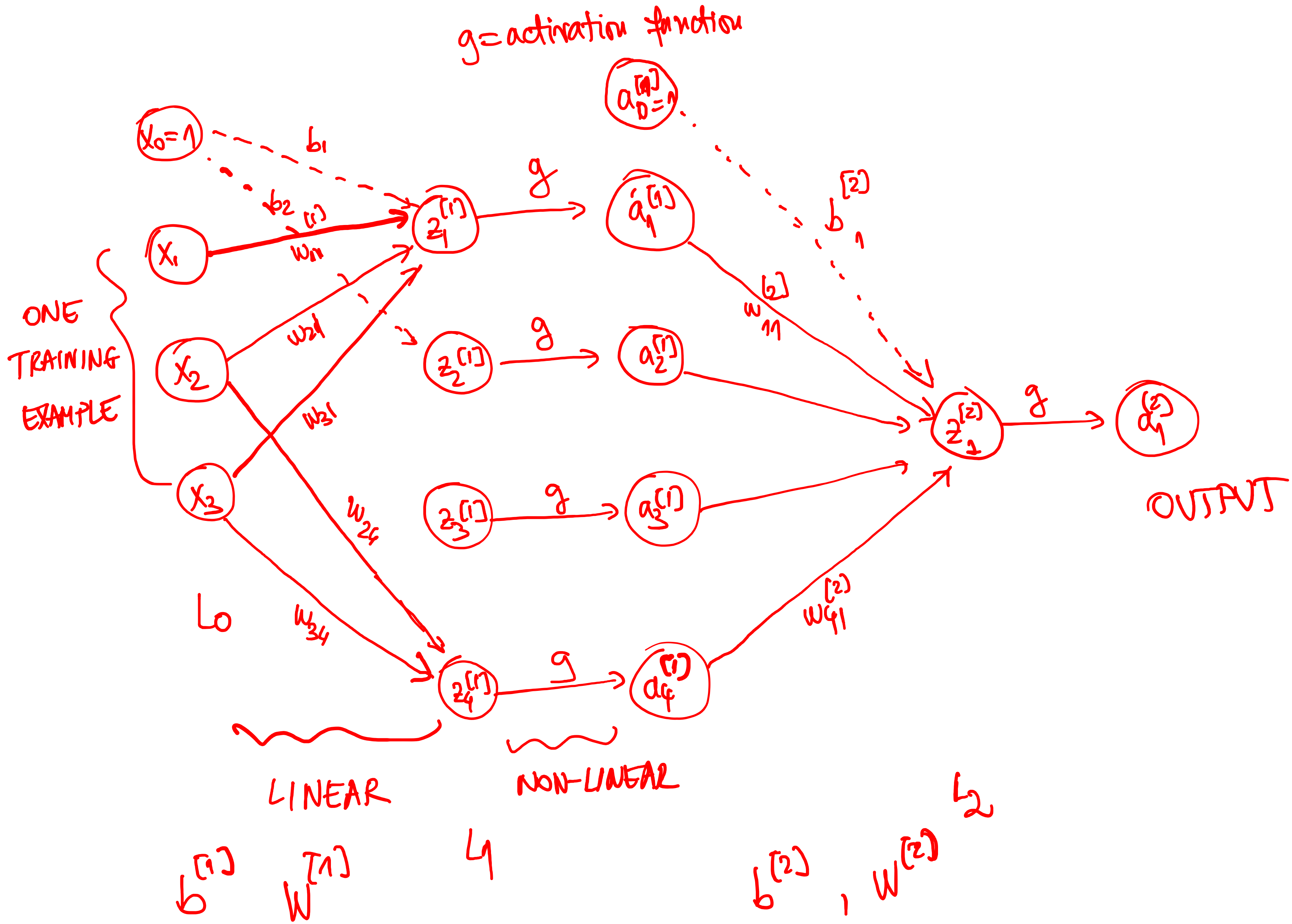
Feed-Forward NN

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

Feed-Forward Neural Network



No cycles $\theta = (b^{[1]}, W^{[1]}, b^{[2]}, W^{[2]})$



g=activation function
$x_0=1$
b_1
b_2
$w_{11}^{[1]}$
x_1
x_2
x_3
w_{21}
w_{31}
w_{24}
w_{34}
ONE TRAINING EXAMPLE
$z_1^{[1]}$
$z_2^{[1]}$
$z_3^{[1]}$
$z_4^{[1]}$
g
$a_0^{[1]}=1$
$a_1^{[1]}$
$a_2^{[1]}$
$a_3^{[1]}$
$a_4^{[1]}$
$b_1^{[2]}$
$w_{11}^{[2]}$
$w_{41}^{[2]}$
$z_1^{[2]}$
$a_1^{[2]}$
OUTPUT
L_0
LINEAR
NON-LINEAR
L_1
L_2
$b^{[1]}$ $W^{[1]}$
$b^{[2]}$, $W^{[2]}$

VECTORIZED FORM

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

$z^{[1]}$ ↓ size 4×1

$$W^{[1]} = \underbrace{\begin{bmatrix} w_{11} & w_{21} & w_{31} \\ & & \\ & & \end{bmatrix}}_{\text{size}(4\times3)} \qquad x = \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}}_{(3\times1)} \qquad b^{[1]} = \begin{bmatrix} b_1^{[1]} \\ \cdot \\ \cdot \\ b_4^{[1]} \end{bmatrix}$$

size 4×1

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

$a^{[1]}$ ↓ Size 4×1

$W^{[1]}$ = weight matrix $L_0 \to L_1$

$b^{[1]}$ = bias vector $L_0 \to L_1$

Vectorization

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

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

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

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

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

Linear

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

Non-Linear

Review

- Perceptrons are limited in their ability to learn
- Feed-Forward Neural Networks are the common neural networks architectures
  - Fully connected networks are called Multi-Layer Perceptron
- Input, output, and hidden layers
  - Linear matrix operations followed by non-linear activations at every layer
- Activations:
  - Examples: ReLU, sigmoid
  - Non-linear functions

Acknowledgements

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