DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

October 30 2018

Logistics

- Start working on projects!
- Final exam
  - Tuesday, Dec. 11, 2-5pm in ISEC 655
- Project presentations
  - Monday, Dec. 3rd
  - Exact time TBD (likely 3:00-5:30pm)
  - Class on Tuesday, Dec. 4 is cancelled
- Project report
  - Due Friday, Dec. 7

Review

- Review of traditional learning techniques
  - Linear classifiers (logistic regression, LDA)
  - Decision trees
  - Ensembles (Random Forests, AdaBoost)
  - SVM
  - Naïve Bayes
- Evaluation in machine learning
  - Confusion matrix
  - Metrics: precision, recall, F1, AUC
  - ROC curves

Comparing Supervised Learning

Comparing Supervised Learning Algorithms : Table

Algorithm	Problem Type	Results interpretable by you?	Easy to explain algorithm to others?	Average predictive accuracy	Training speed	Prediction speed
KNN	Either	Yes	Yes	Lower	Fast	Depends on n
Linear regression	Regression	Yes	Yes	Lower	Fast	Fast
Logistic regression	Classification	Somewhat	Somewhat	Lower	Fast	Fast
Naive Bayes	Classification	Somewhat	Somewhat	Lower	Fast (excluding feature extraction)	Fast
Decision trees	Either	Somewhat	Somewhat	Lower	Fast	Fast
Random Forests	Either	A little	No	Higher	Slow	Moderate
AdaBoost	Either	A little	No	Higher	Slow	Fast
Neural networks	Either	No	No	Higher	Slow	Fast

Roadmap to End-of-Semester

- Deep Learning
  - Motivation
  - Feed-Forward Neural Networks
  - Training by backpropagation
  - Convolutional and Recurrent Neural Networks
- Unsupervised learning
  - Principal Component Analysis (PCA)
  - Feature representation (Autoencoders)
  - Clustering (k-means, Hierarchical Clustering)
- Adversarial learning

Today's Outline

- Motivation for Deep Learning
- Deep Learning as representation learning
- Categories of neural networks
- Feed-Forward architectures
  – Activation functions
  – Vectorization
- Representing Boolean functions
  – XOR can be learned with 1 hidden layer

Deep Learning

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

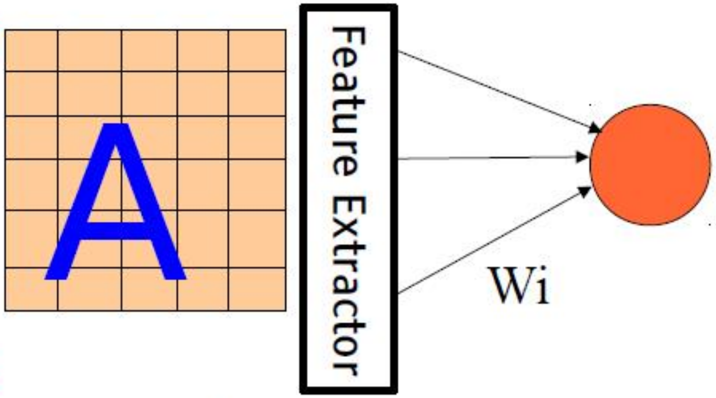
hand-crafted Feature Extractor → "Simple" Trainable Classifier →

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

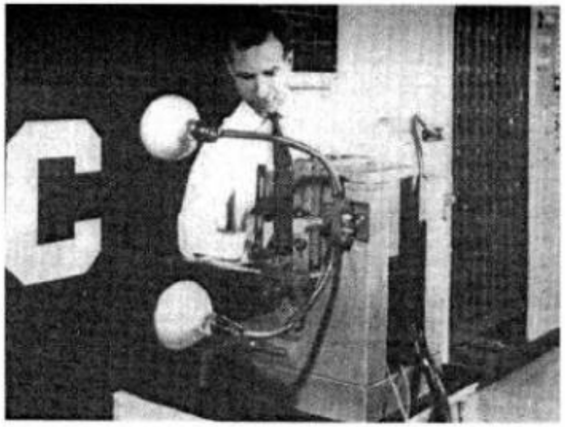
Trainable Feature Extractor → Trainable Classifier →

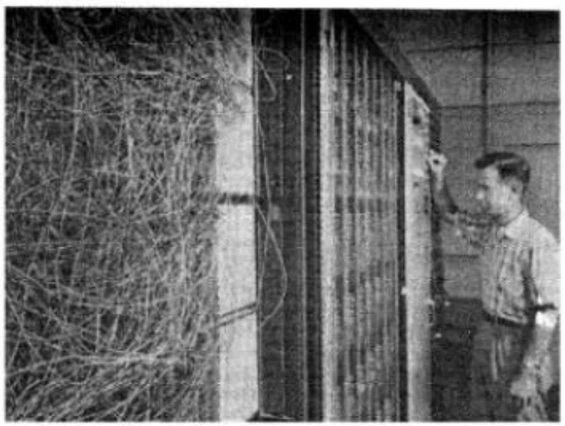
Before 2013

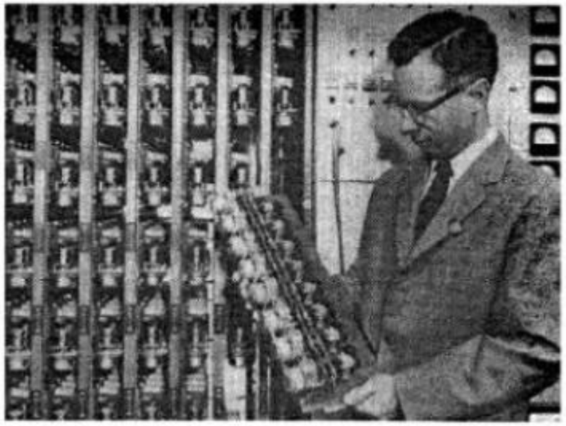
- The first learning machine: the Perceptron
  - Built at Cornell in 1960
- The Perceptron was a linear classifier on top of a simple feature extractor
- The vast majority of practical applications of ML today use glorified linear classifiers or glorified template matching.
- Designing a feature extractor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



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

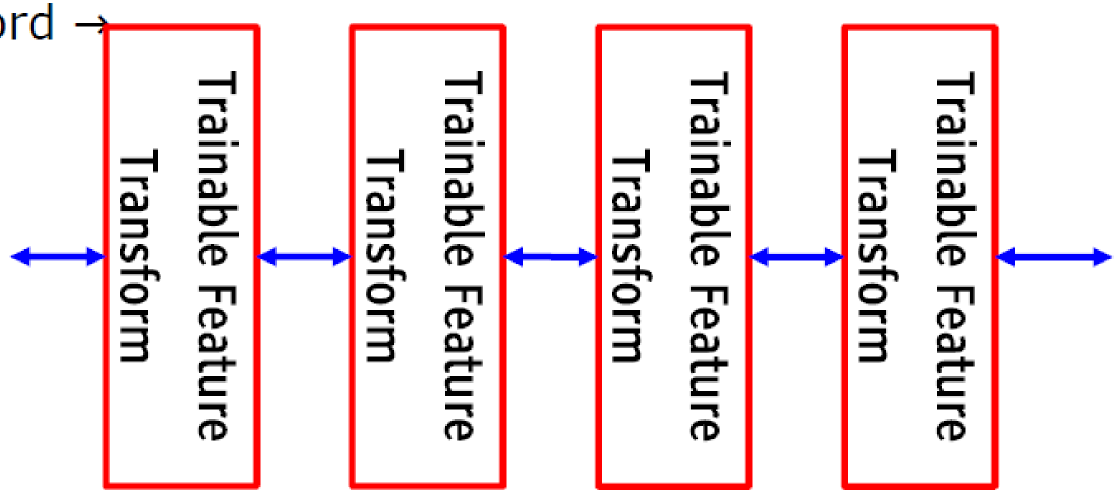




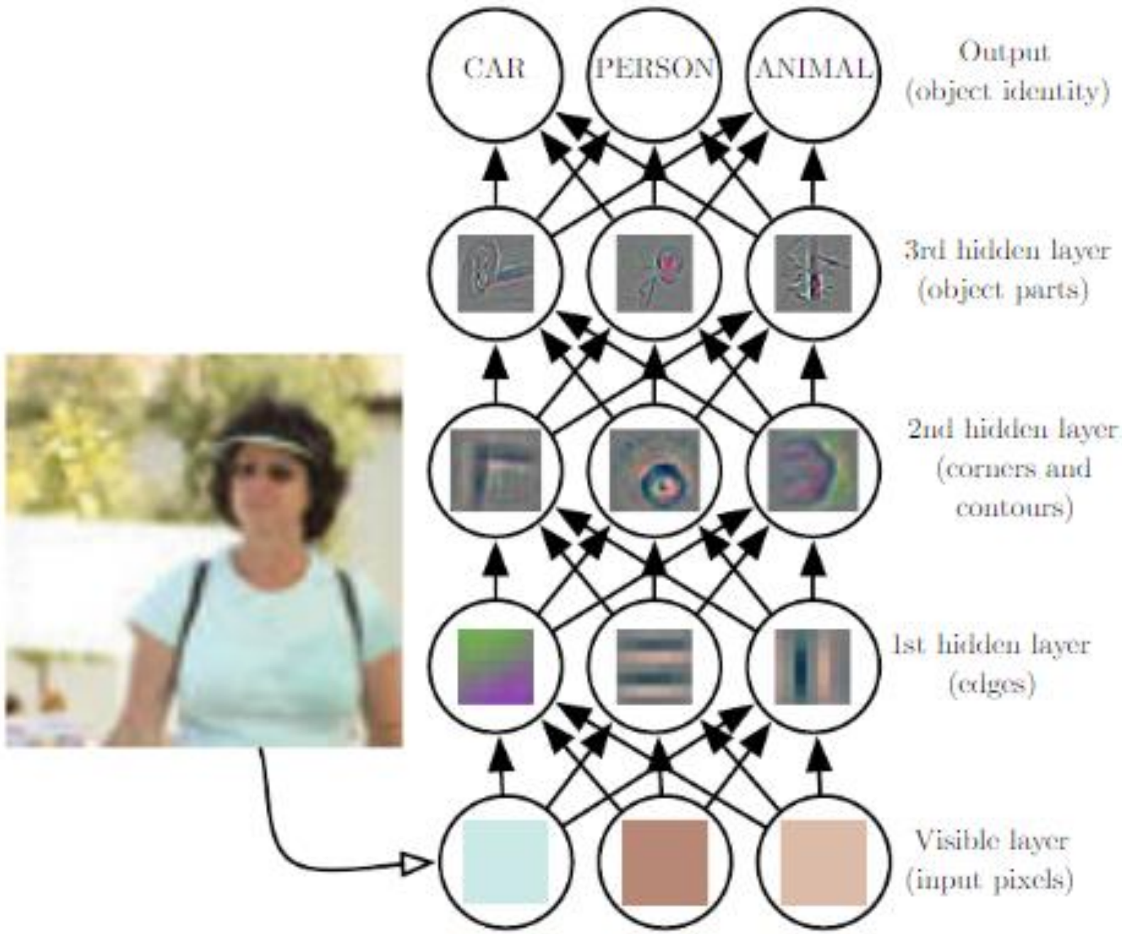


Trainable Feature Hierarchy

- 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 →

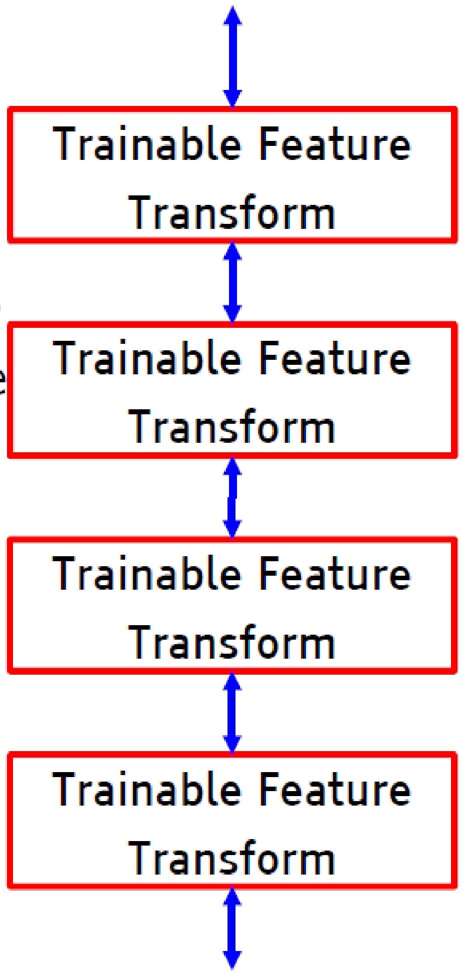


Learning Representations



Learning Representations

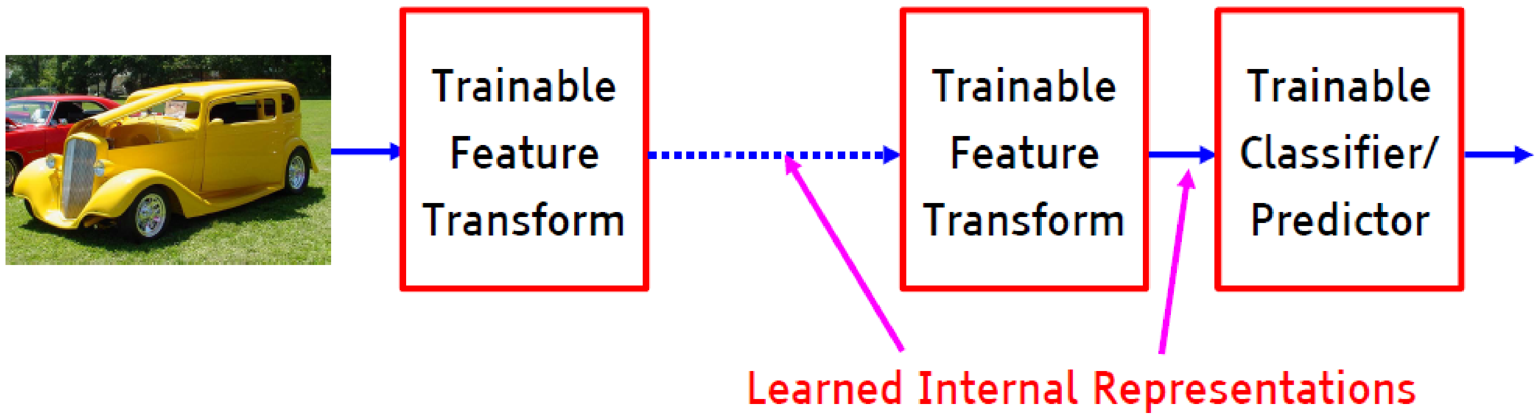
- How do we learn representations of the perceptual world?
  - How can a perceptual system build itself by looking at the world?
  - How much prior structure is necessary
- ML/AI: how do we learn features or feature hierarchies?
  - What is the fundamental principle? What is the learning algorithm? What is the architecture?
- Neuroscience: how does the cortex learn perception?
  - Does the cortex "run" a single, general learning algorithm? (or a small number of them)
- CogSci: how does the mind learn abstract concepts on top of less abstract ones?



- Deep Learning addresses the problem of learning hierarchical representations with a single algorithm

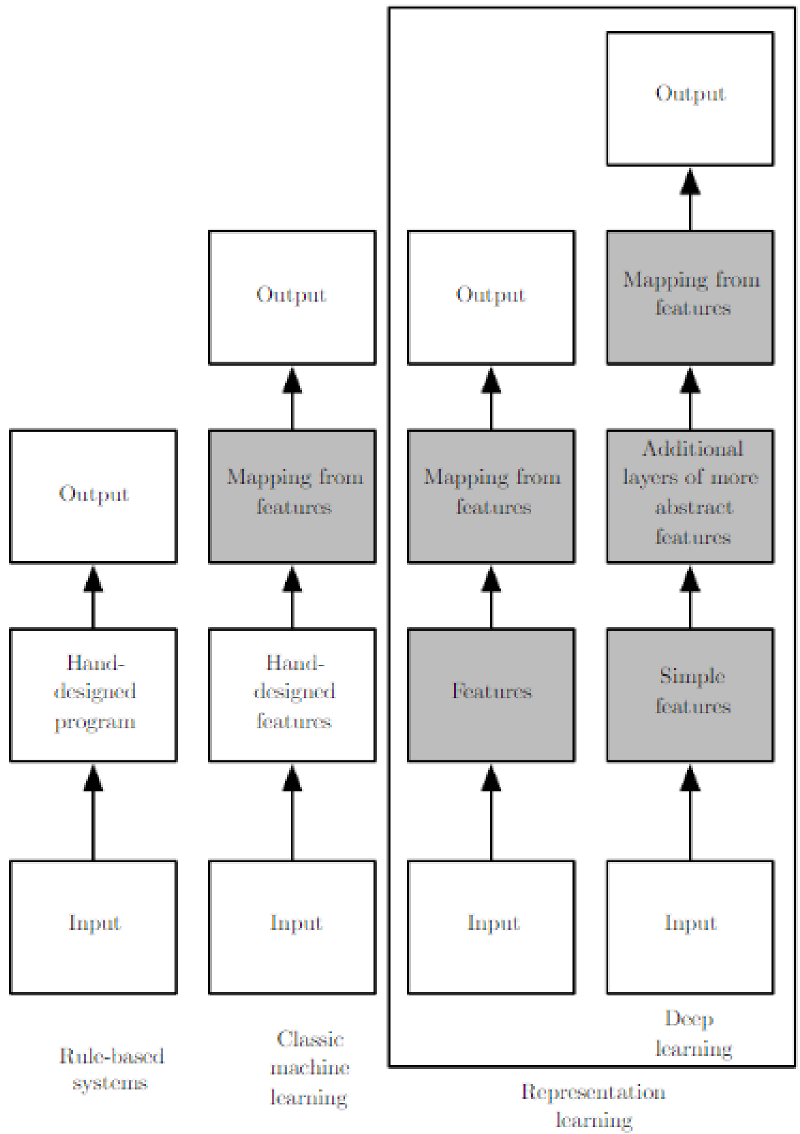
End-to-end learning

- **A hierarchy of trainable feature transforms**
  - Each module transforms its input representation into a higher-level one.
  - High-level features are more global and more invariant
  - Low-level features are shared among categories



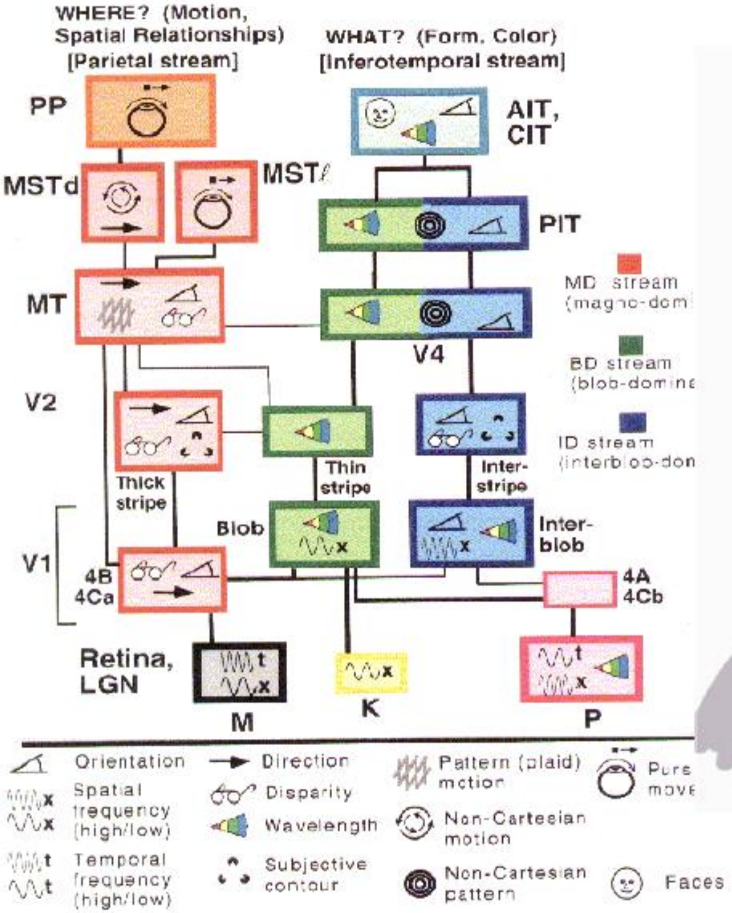
- **How can we make all the modules trainable and get them to learn appropriate representations?**

Deep Learning vs Traditional Learning

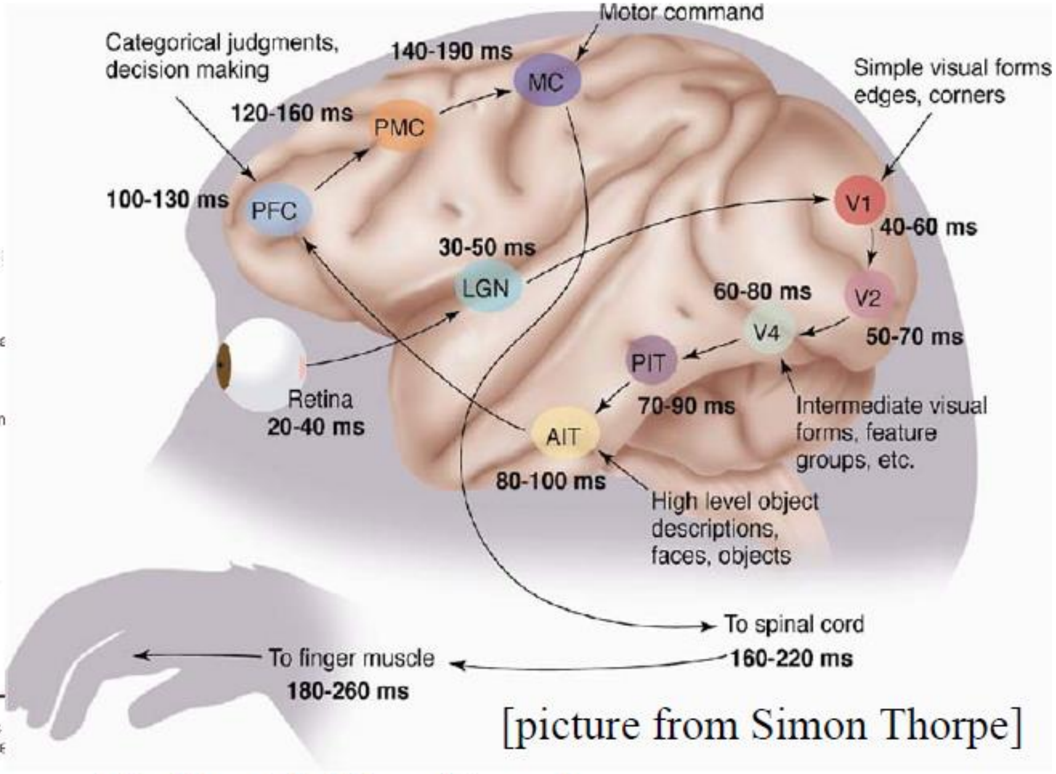


The Visual Cortex is Hierarchical

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



[Gallant & Van Essen]

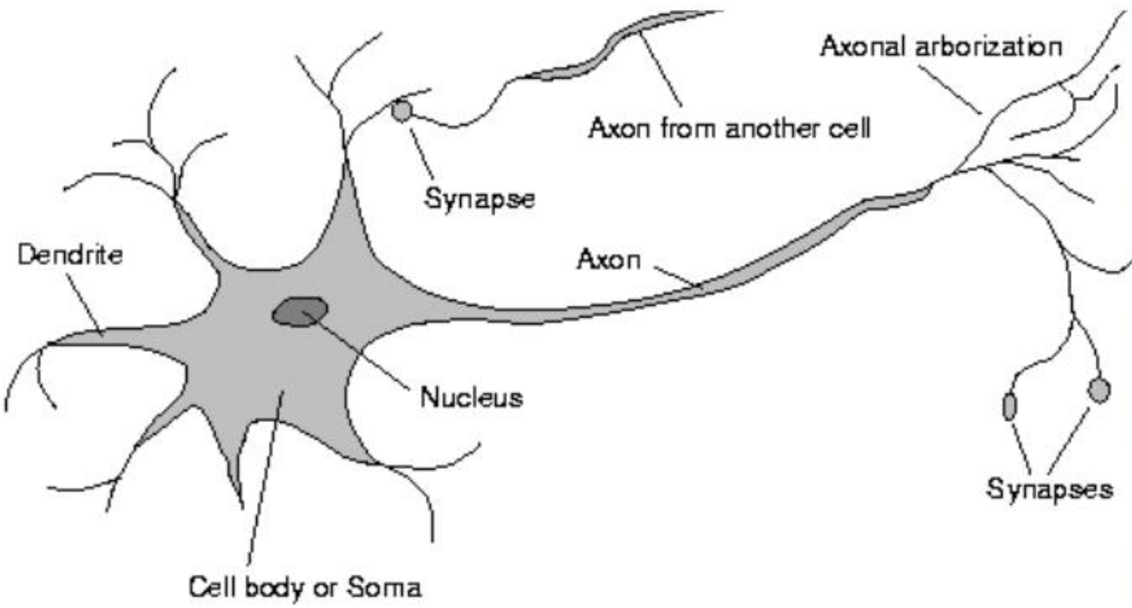


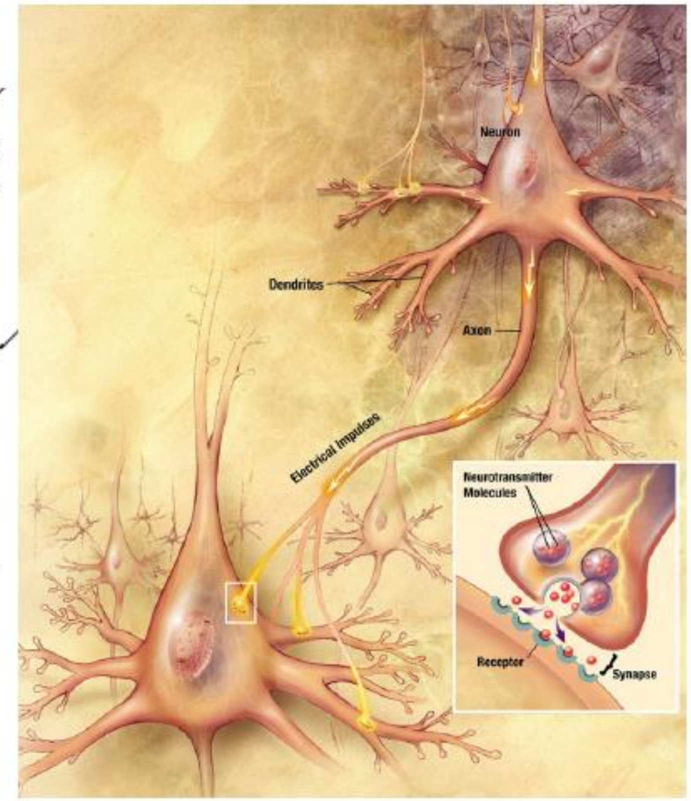
[picture from Simon Thorpe]

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!

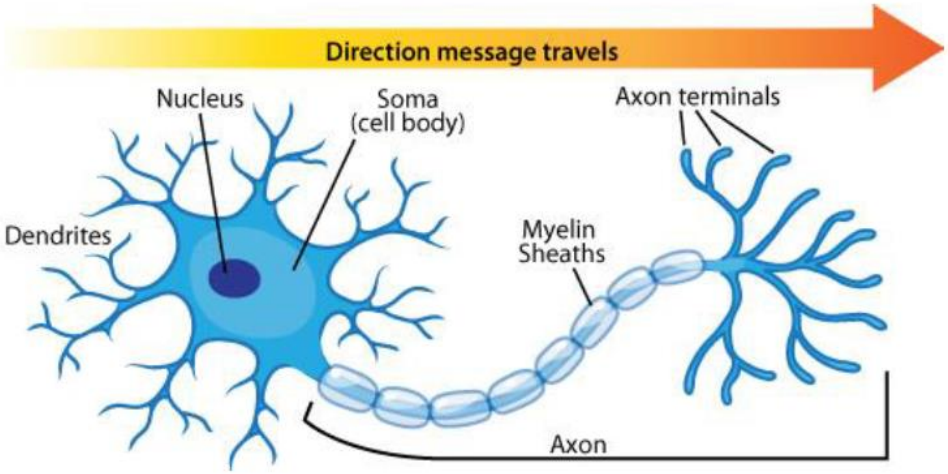
Biology of a Neuron

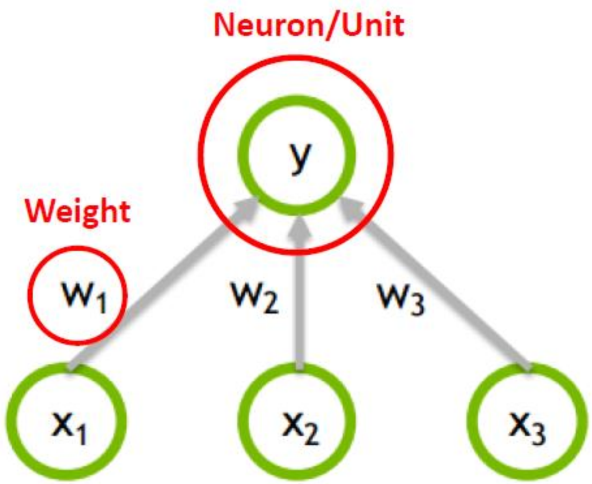




Analogy to Human Brain

Human Brain





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

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

Biological Neuron

Artificial Neuron

Comparison of computing power

INFORMATION CIRCA 2012	**Computer**	**Human Brain**
Computation Units	10-core Xeon: 10^9 Gates	10^{11} Neurons
Storage Units	10^9 bits RAM, 10^{12} bits disk	10^{11} neurons, 10^{14} synapses
Cycle time	10^{-9} sec	10^{-3} sec
Bandwidth	10^9 bits/sec	10^{14} bits/sec

- Computers are way faster than neurons…
- But there are a lot more neurons than we can reasonably model in modern digital computers, and they all fire in parallel
- Neural networks are designed to be massively parallel
- The brain is effectively a billion times faster

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

Example

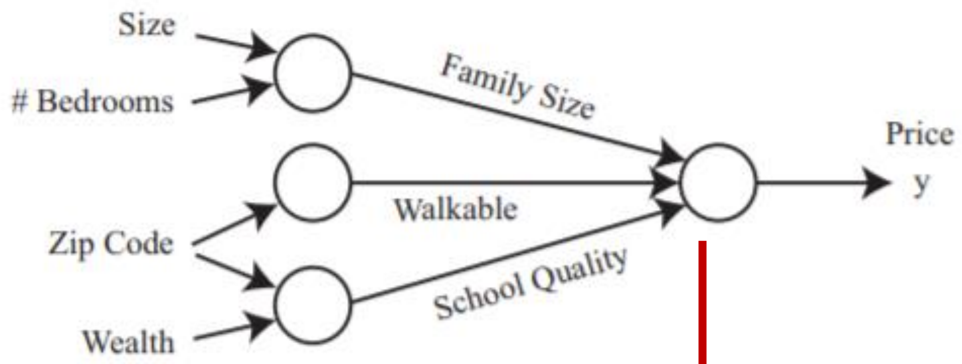
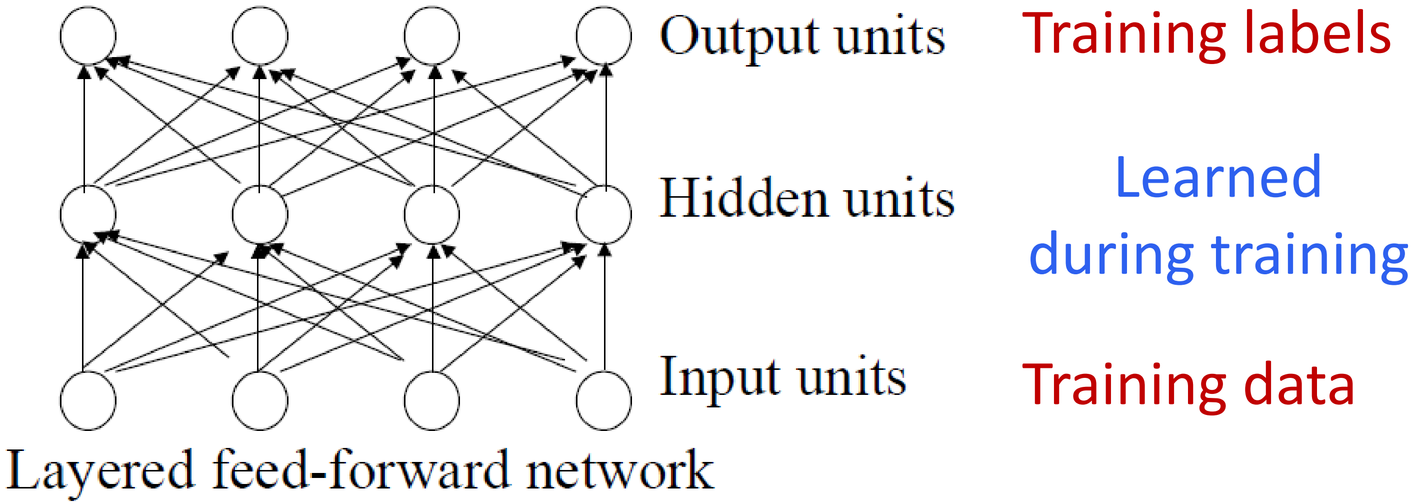


Figure 2: Diagram of a small neural network for predicting housing prices.

Intermediate Features

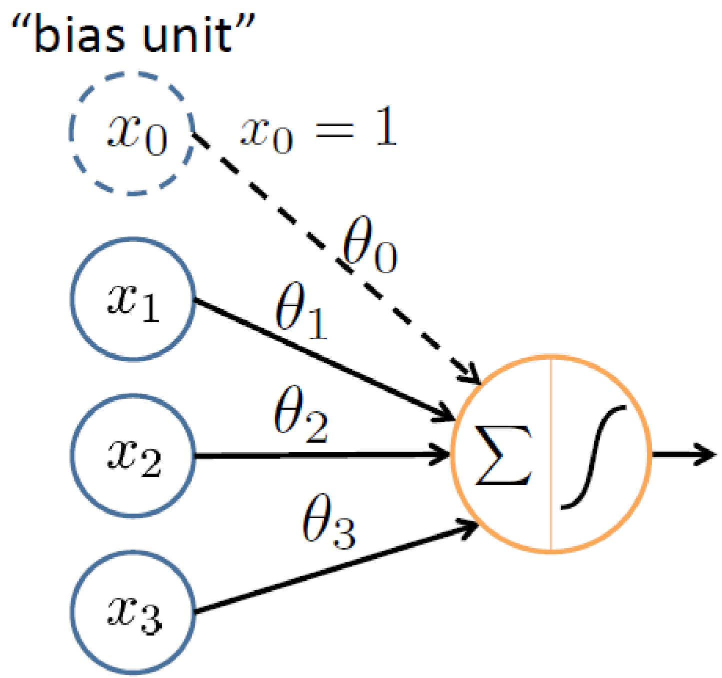
- Provide as input only training data: input and label
- Neural Networks automatically learn intermediate features!

Neural Networks



- 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**

Logistic Unit



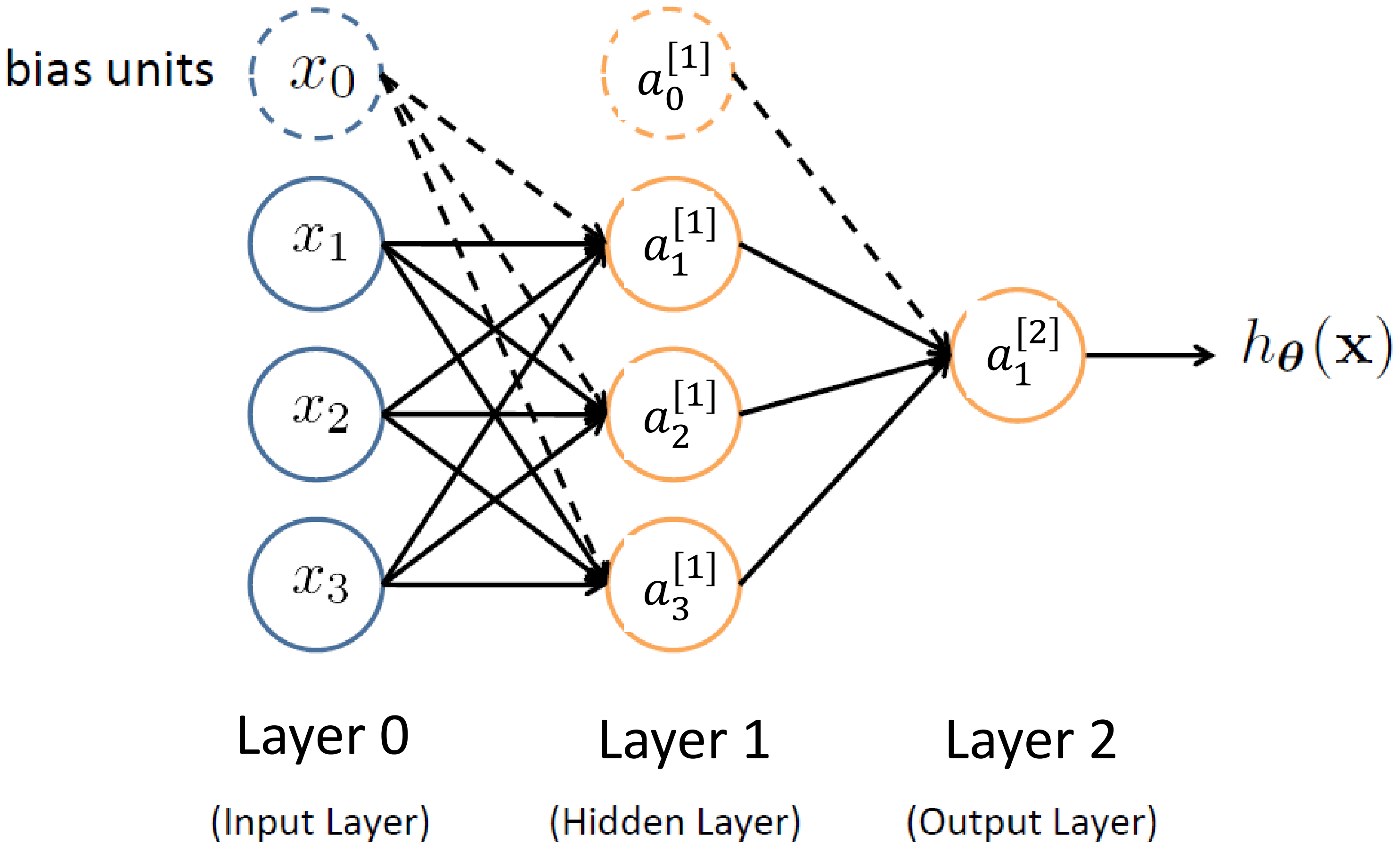
$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

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

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

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

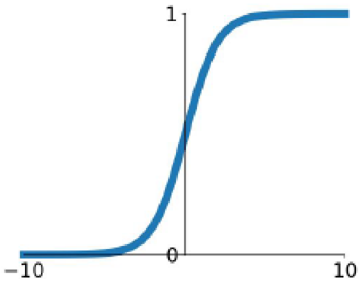
Neural Network



Activation Functions

Sigmoid

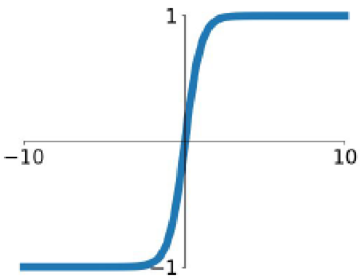
$\sigma(x) = \frac{1}{1+e^{-x}}$



Positive Classification

tanh

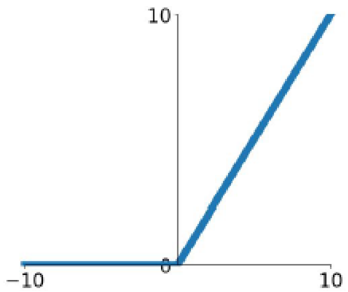
$\tanh(x)$



Regression

ReLU

$\max(0, x)$

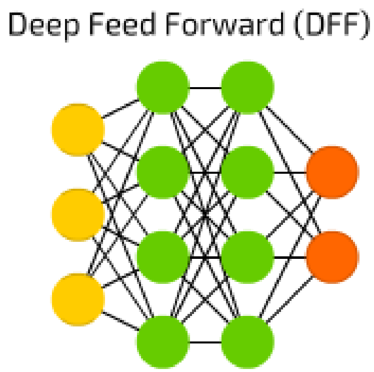


Positive Classification

Neural Network Architectures

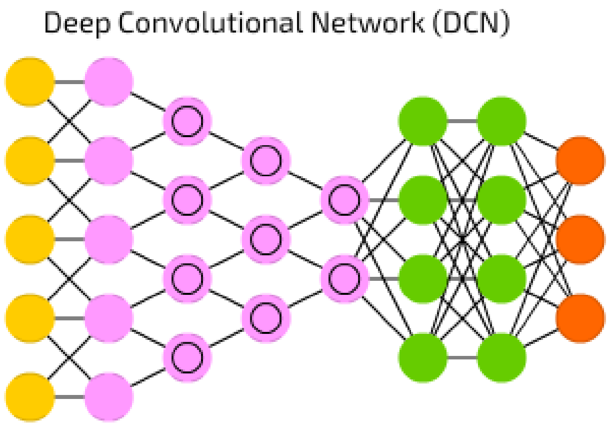
Feed-Forward Networks

- Neurons from each layer connect to neurons from next layer



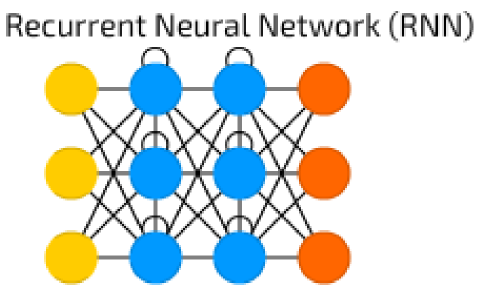
Convolutional Networks

- Includes convolution layer for feature reduction
- Learns hierarchical representations



Recurrent Networks

- Keep hidden state
- Have cycles in computational graph



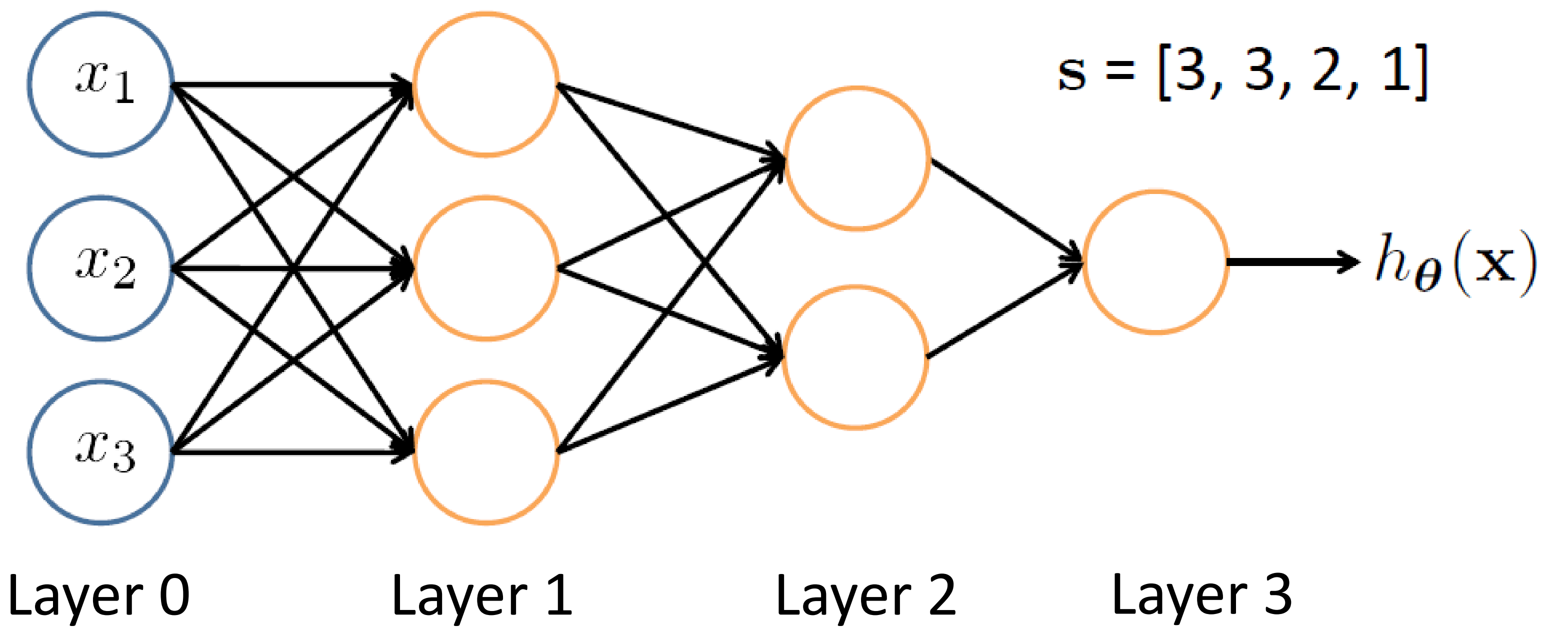
Feed-Forward Process

- Input layer units are set by some exterior function (think of these as **sensors**), which causes their output links to be **activated** at the specified level

- Working forward through the network, the **input function** of each unit is applied to compute the input value
  - Usually this is just the weighted sum of the activation on the links feeding into this node

- The **activation function** transforms this input function into a final value
  - Typically this is a **nonlinear** function, often a **sigmoid** function corresponding to the "threshold" of that node

Feed-Forward Networks



L denotes the number of layers

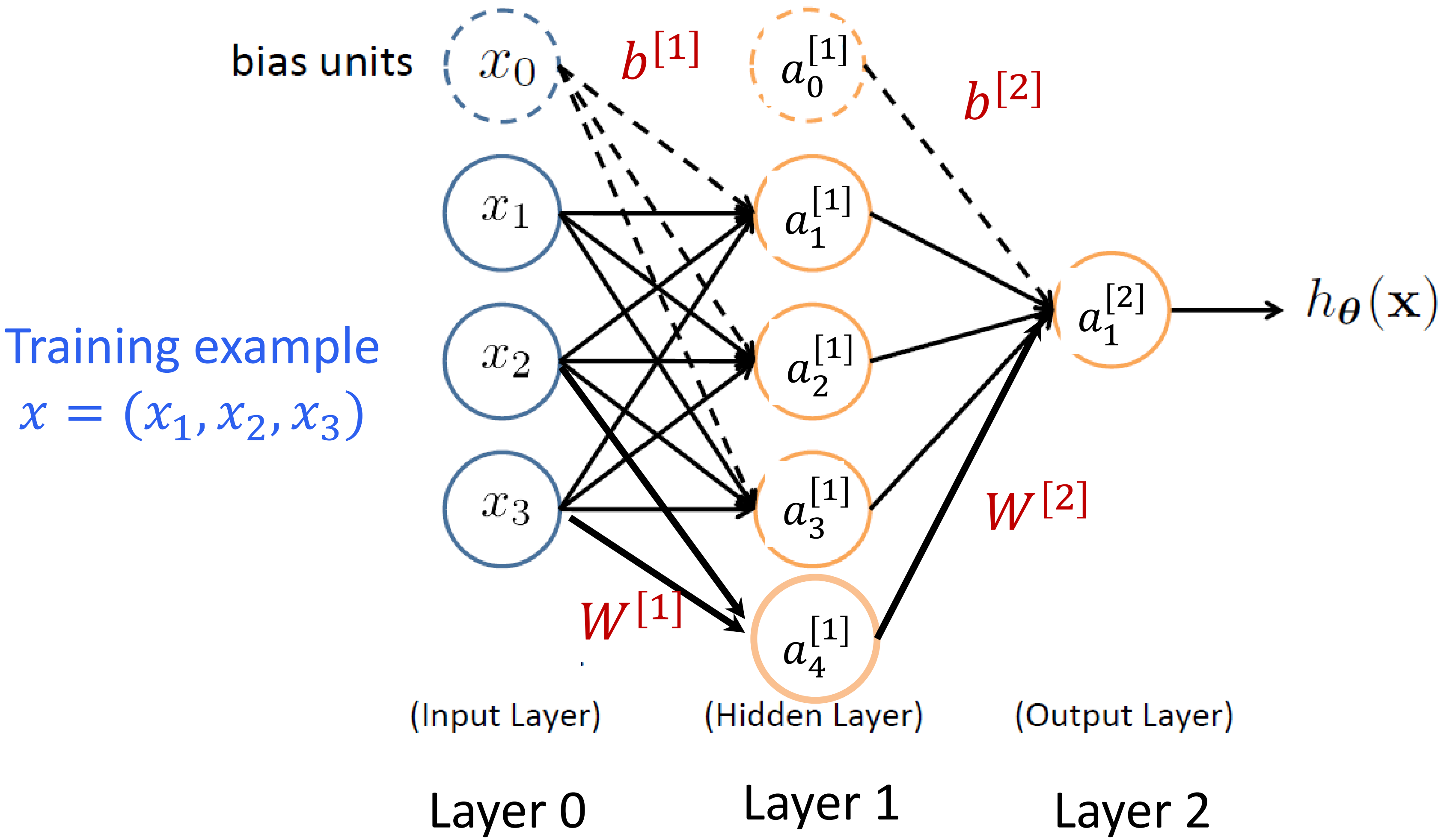
$\mathbf{s} \in {\mathbb{N}^+}^L$ contains the numbers of nodes at each layer

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

Feed-Forward NN

- 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



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

Vectorization

$$
\begin{array}{ccc}
z_1^{[1]} = W_1^{[1]T} x + b_1^{[1]} & \text{and} & a_1^{[1]} = g(z_1^{[1]}) \\
\vdots & \vdots & \vdots \\
z_4^{[1]} = W_4^{[1]T} x + b_4^{[1]} & \text{and} & a_4^{[1]} = g(z_4^{[1]})
\end{array}
$$

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

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

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

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

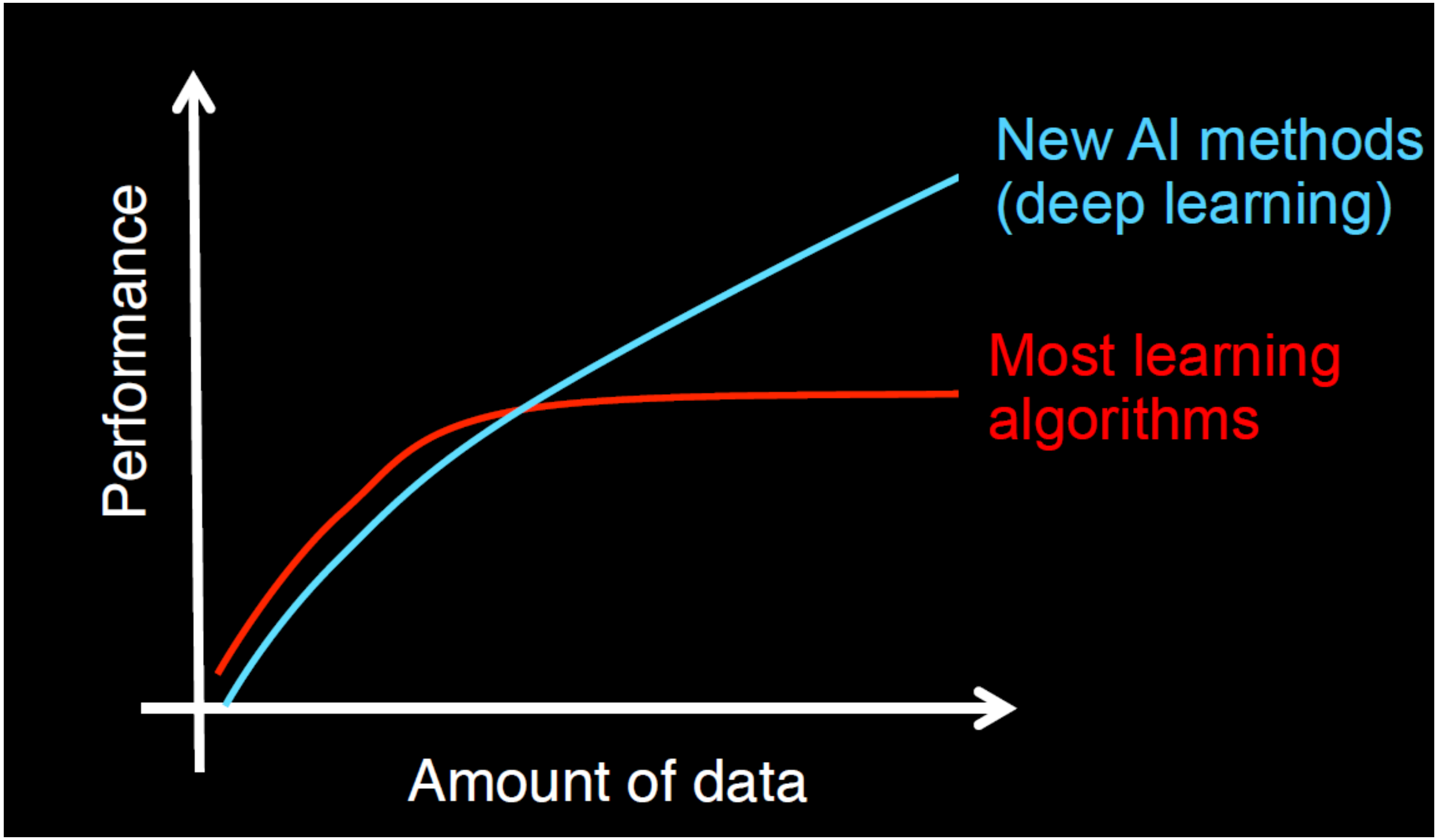
Vectorization

Output layer

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

$$\boxed{\underbrace{z^{[2]}}_{1\times1} = \underbrace{W^{[2]}}_{1\times4} \underbrace{a^{[1]}}_{4\times1} + \underbrace{b^{[2]}}_{1\times1} \quad \text{and} \quad \underbrace{a^{[2]}}_{1\times1} = g(\underbrace{z^{[2]}}_{1\times1})}$$

Performance of Deep Learning



Deep Learning Applications

Acknowledgements

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