

Introduction to Machine Learning

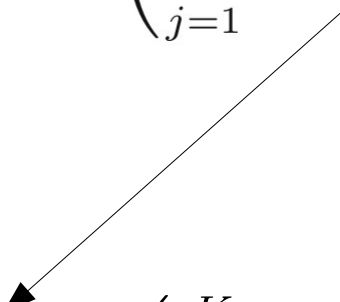
Lecture 14 Deep Learning I Fundamentals

Goker Erdogan
26 – 30 November 2018
Pontificia Universidad Javeriana

(Artificial) Neural networks

- A popular and high-performing approach
 - Represent the input \rightarrow output as a composition of multiple functions
 - Learning adaptive basis functions
 - Representation learning
- Example

$$y(\mathbf{x}, \mathbf{w}) = f \left(\sum_{j=1}^M w_j \phi_j(\mathbf{x}) \right)$$


$$\phi_j(x) = g \left(\sum_{k=1}^K w_k \phi_k(x) \right)$$

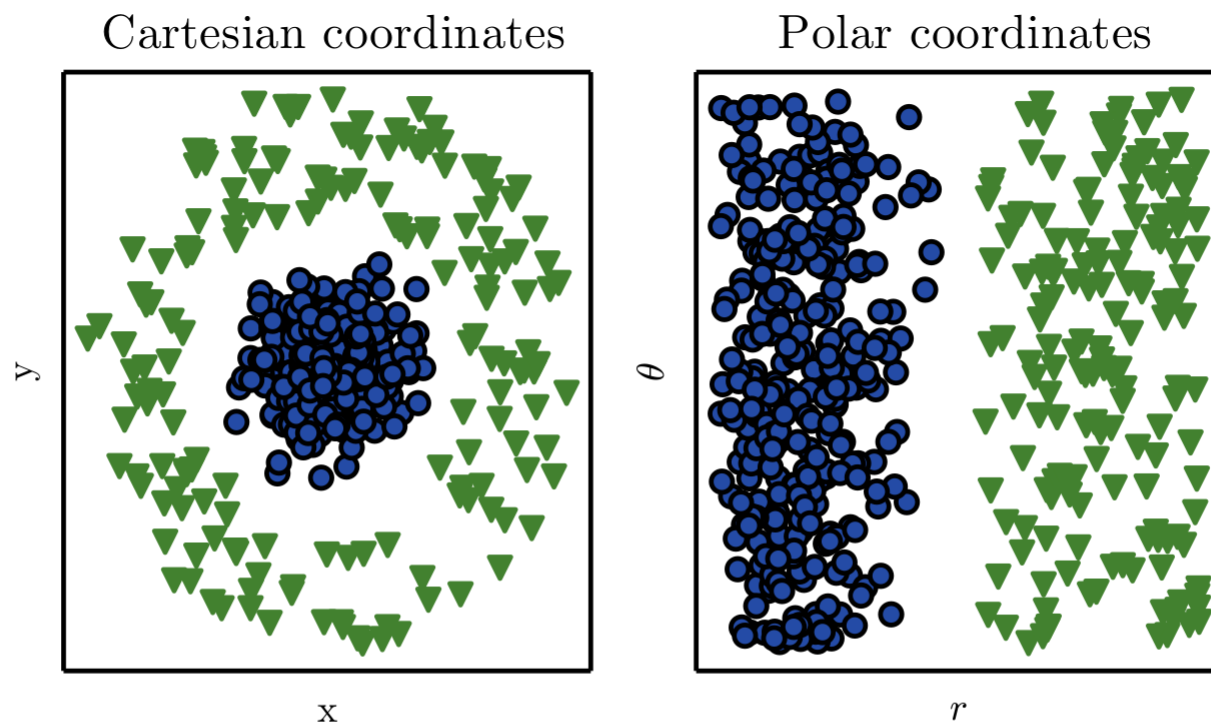
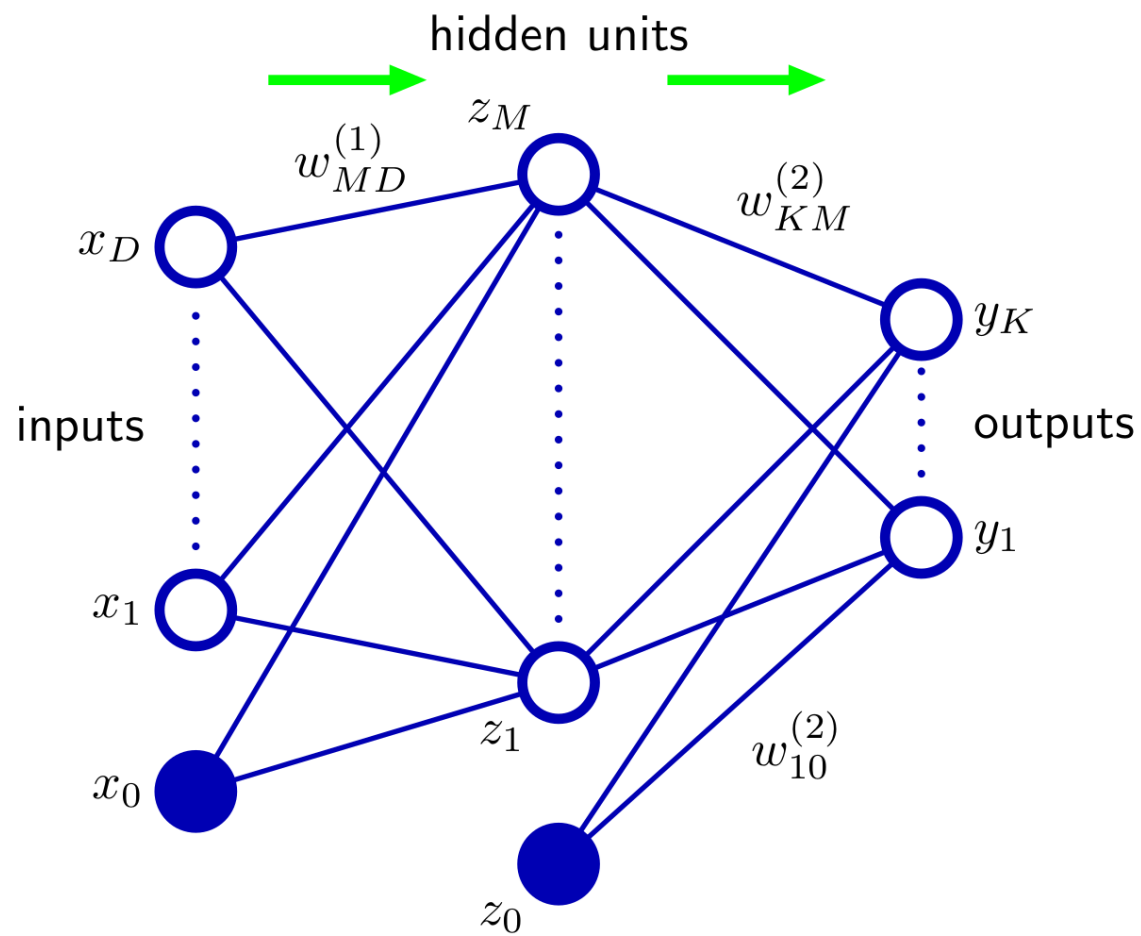


Figure 1.1: Example of different representations: suppose we want to separate two categories of data by drawing a line between them in a scatterplot. In the plot on the left, we represent some data using Cartesian coordinates, and the task is impossible. In the plot on the right, we represent the data with polar coordinates and the task becomes simple to solve with a vertical line. (Figure produced in collaboration with David Warde-Farley.)



$$y_k(\mathbf{x}, \mathbf{w}) = \sigma \left(\sum_{j=1}^M w_{kj}^{(2)} h \left(\sum_{i=1}^D w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

Neural networks basics

- A neural network is built out of **layers**
 - Each layer has multiple **neurons (units)**
 - Each neuron represents a function of the following form

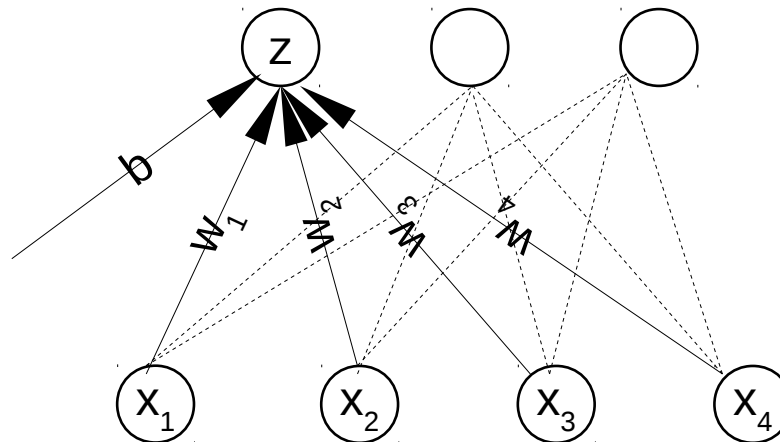
$$z(x) = a \left(\sum_m w_m x_m + b \right)$$

output

activation function

weights

bias

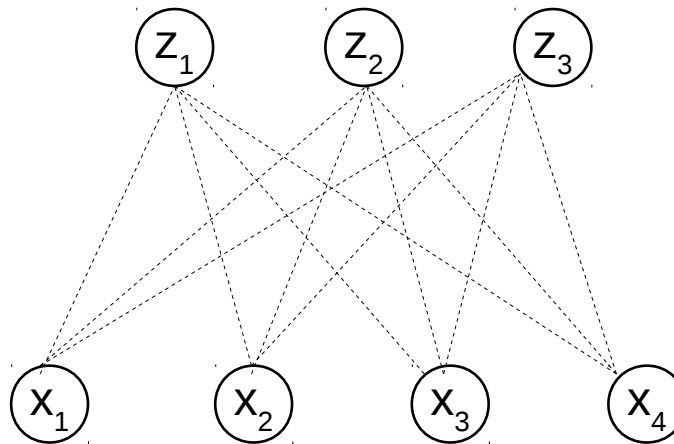


Neural networks basics

- We can represent the computation in a layer as

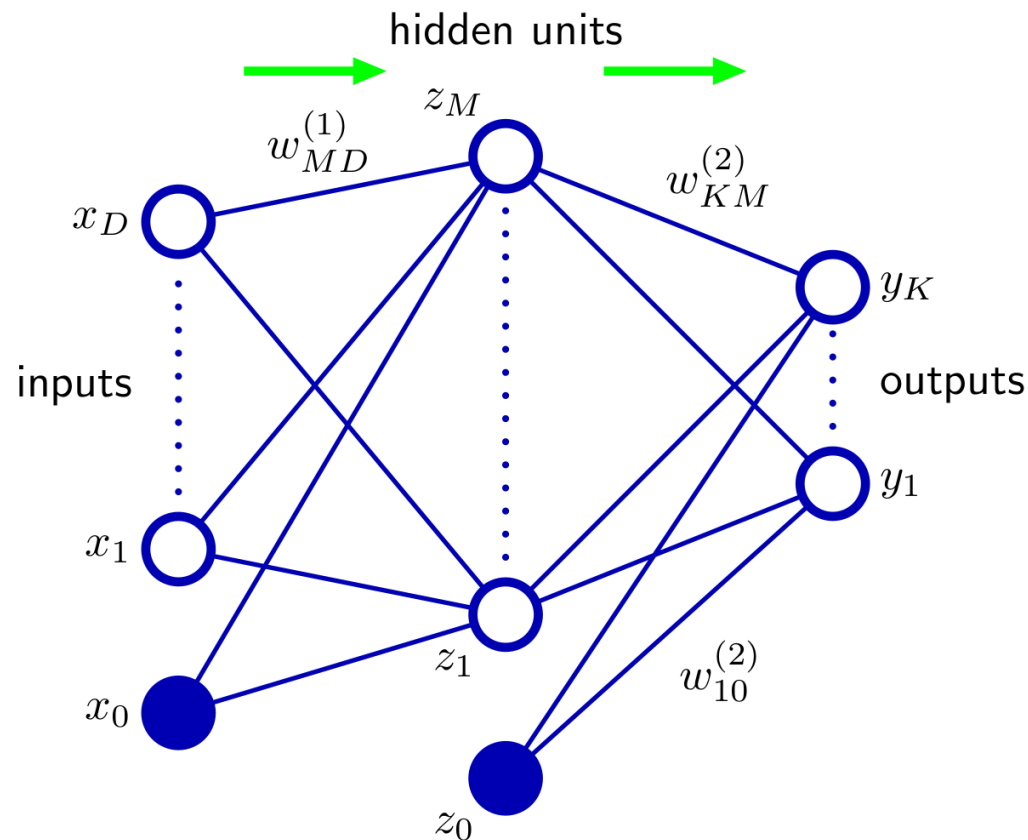
$$z = a(Wx + b)$$

- In the below example
 - 4 input units, 3 output units
 - $W_{3 \times 4}$: weight matrix
 - b_3 : bias vector



Neural networks basics

- More terminology
 - **Depth** of a network: number of layers (not counting the input layer)
 - **Hidden units**
 - **Parameters**: weights and biases



Activation functions

- **Nonlinear function** applied (usually) at each layer

A diagram showing the formula $z(x) = a \left(\sum_m w_m x_m + b \right)$. Arrows point from labels to parts of the formula: 'output' points to $z(x)$, 'activation function' points to a , 'weights' points to w_m , and 'bias' points to b .

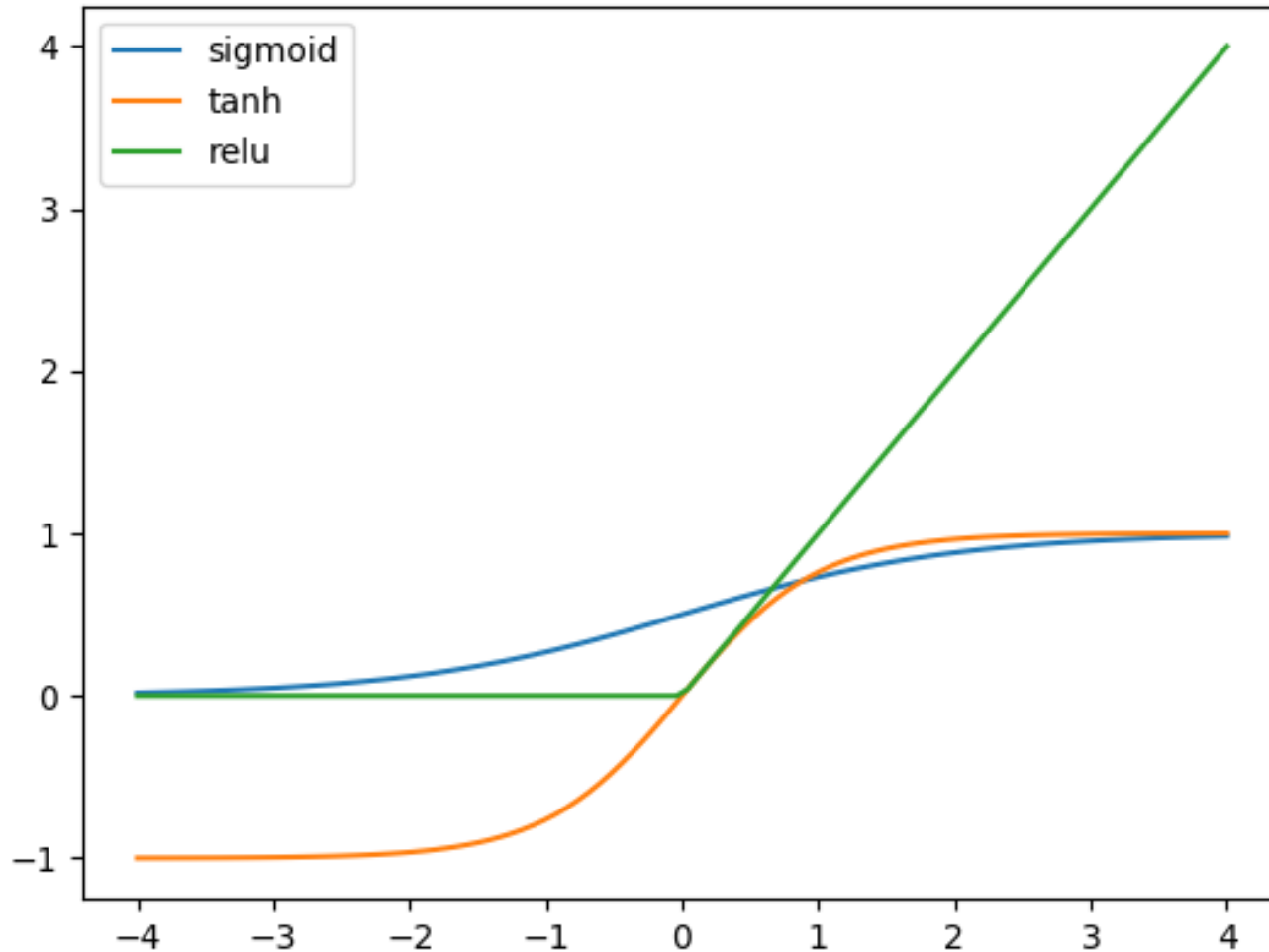
$$z(x) = a \left(\sum_m w_m x_m + b \right)$$

output activation function weights bias

Various choices for activation functions

- Logistic sigmoid $\sigma(a) \equiv \frac{1}{1 + \exp(-a)}$
- tanh (hyperbolic tangent) $\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$
- Rectified linear unit (ReLU) $\text{relu}(a) = \max(0, a)$

Activation functions



Activation functions

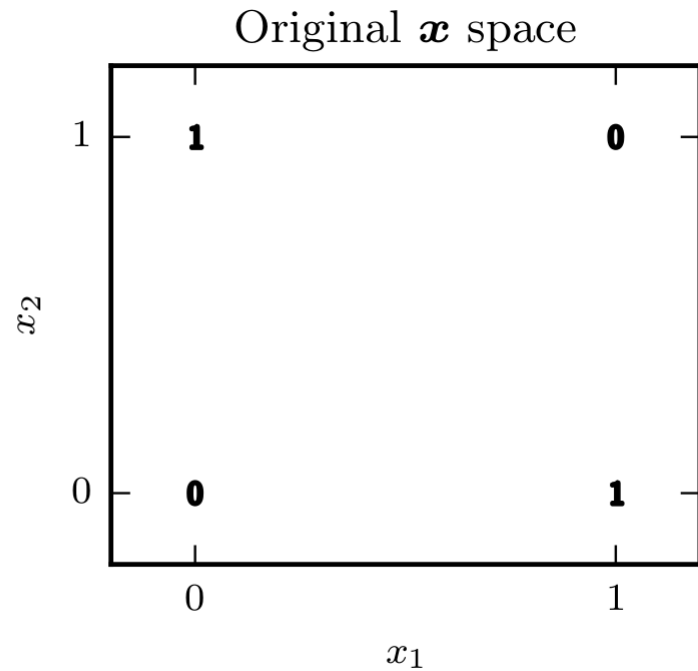
- Has to be nonlinear
 - Otherwise there is **no point in adding more layers**
 - Exercise: show this
- Current wisdom
 - **Use ReLUs**
 - Sigmoid and tanh used to be popular
 - But ReLU outperforms both
- Why ReLUs?
 - Sigmoid and tanh are **squashing** activation functions
 - Outputs **saturate** if input gets large
 - Derivatives of sigmoid and tanh are small in this case
 - Makes it more difficult to do gradient based optimization
 - In general, activation functions that are more linear-like work better

Output layer

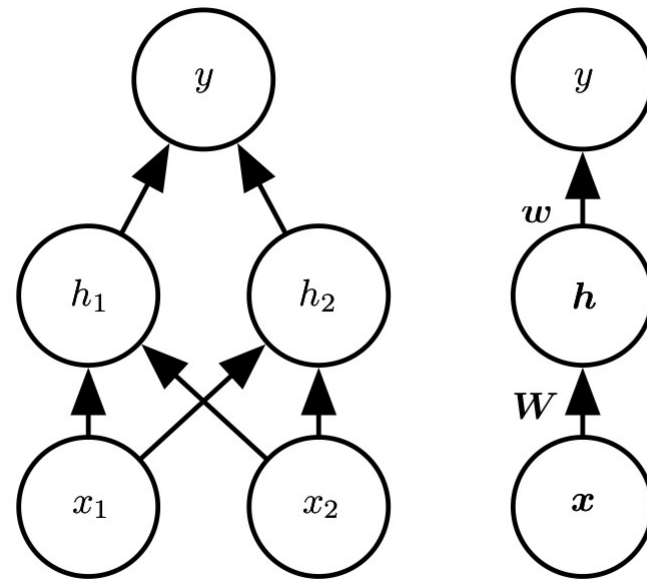
- A general recipe for constructing parametric input → output functions
 - Can be **applied to supervised and unsupervised problems**
 - Classification, regression, dimensionality reduction ...
- Activation function on the output **depends on the problem**
 - **Regression**: No activation function
 - Use output as is
 - **Binary classification**: Use logistic sigmoid
 - Map real numbers to $[0, 1]$
 - **Multi-class classification**: Use softmax
 - Map real numbers to a probability distribution over K classes

Example: XOR function

Problem



Network architecture



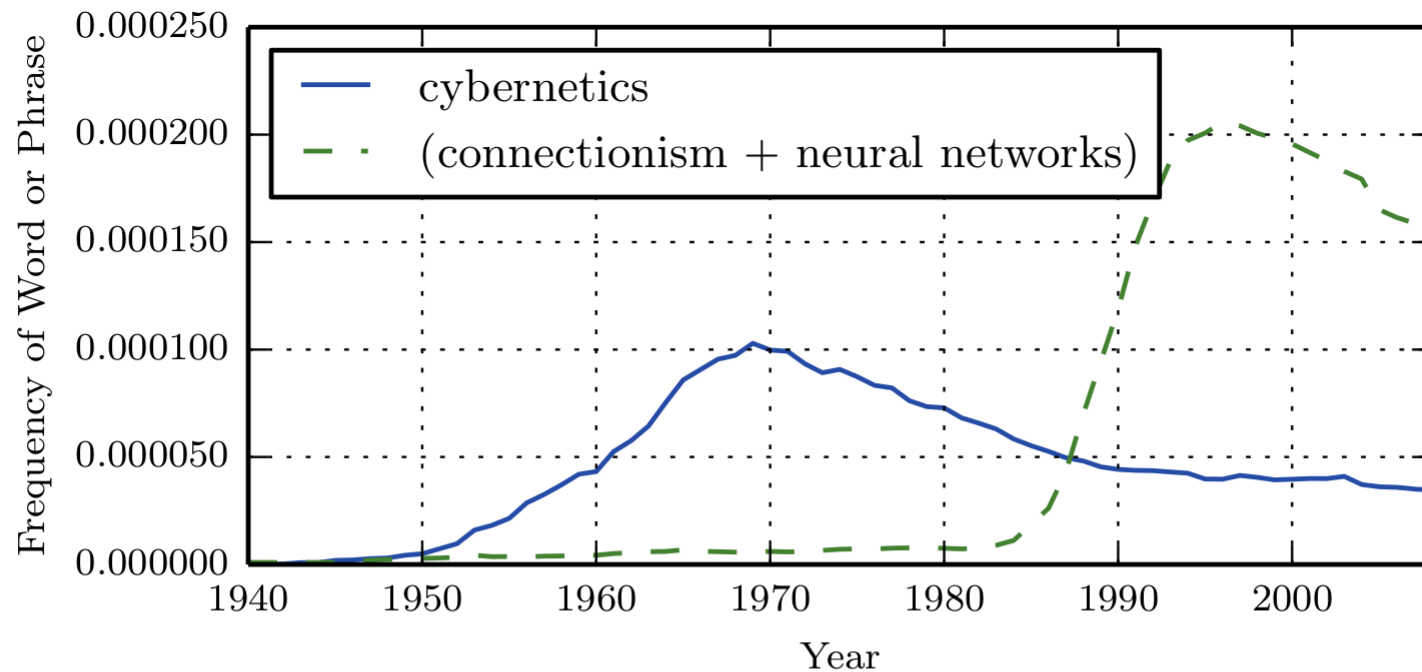
$$\mathbf{W} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} 1 \\ -2 \end{bmatrix},$$

$$f(\mathbf{x}; \mathbf{W}, \mathbf{c}, \mathbf{w}, b) = \mathbf{w}^\top \max\{0, \mathbf{W}^\top \mathbf{x} + \mathbf{c}\} + b.$$

<https://playground.tensorflow.org>

Brief history of neural networks

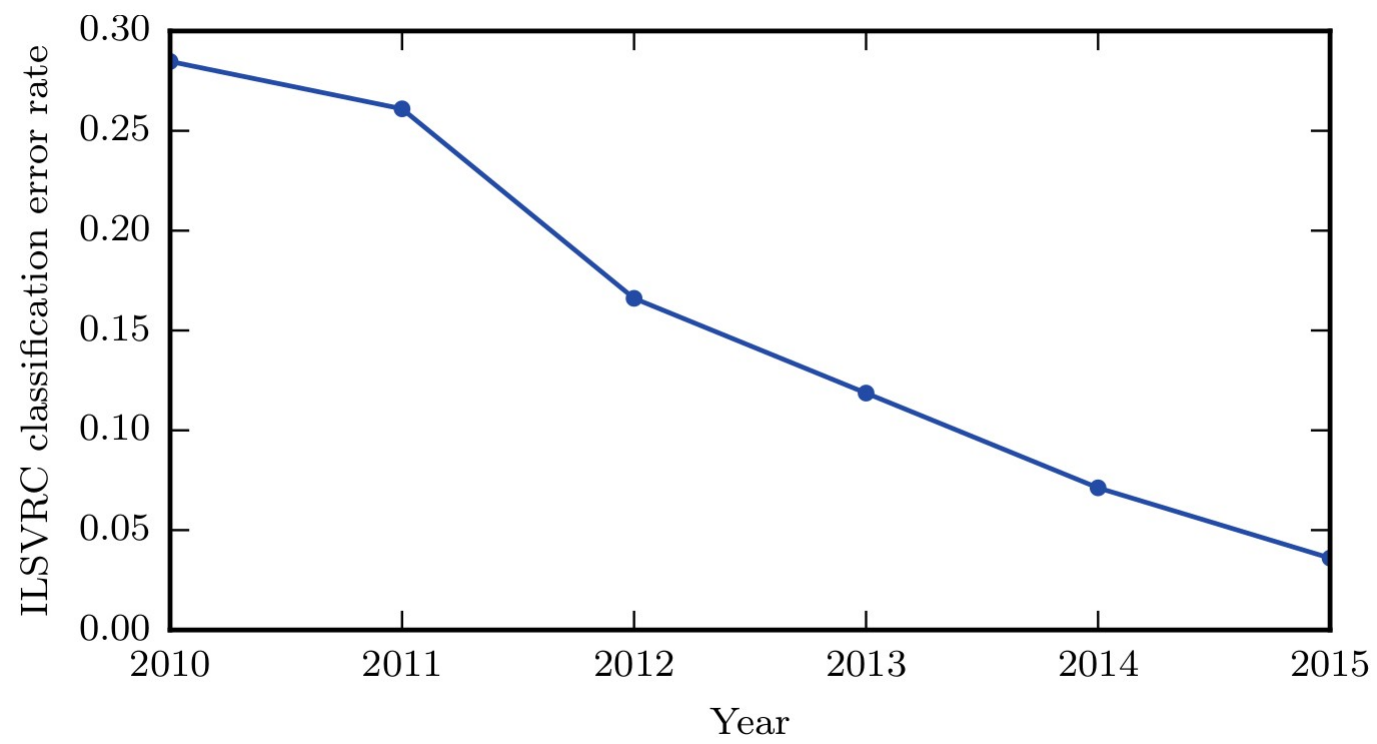
- 1940-1960: Cybernetics
- 1980-1990: Connectionism (neural networks)
- 2006-present: Deep learning



Brief history of neural networks

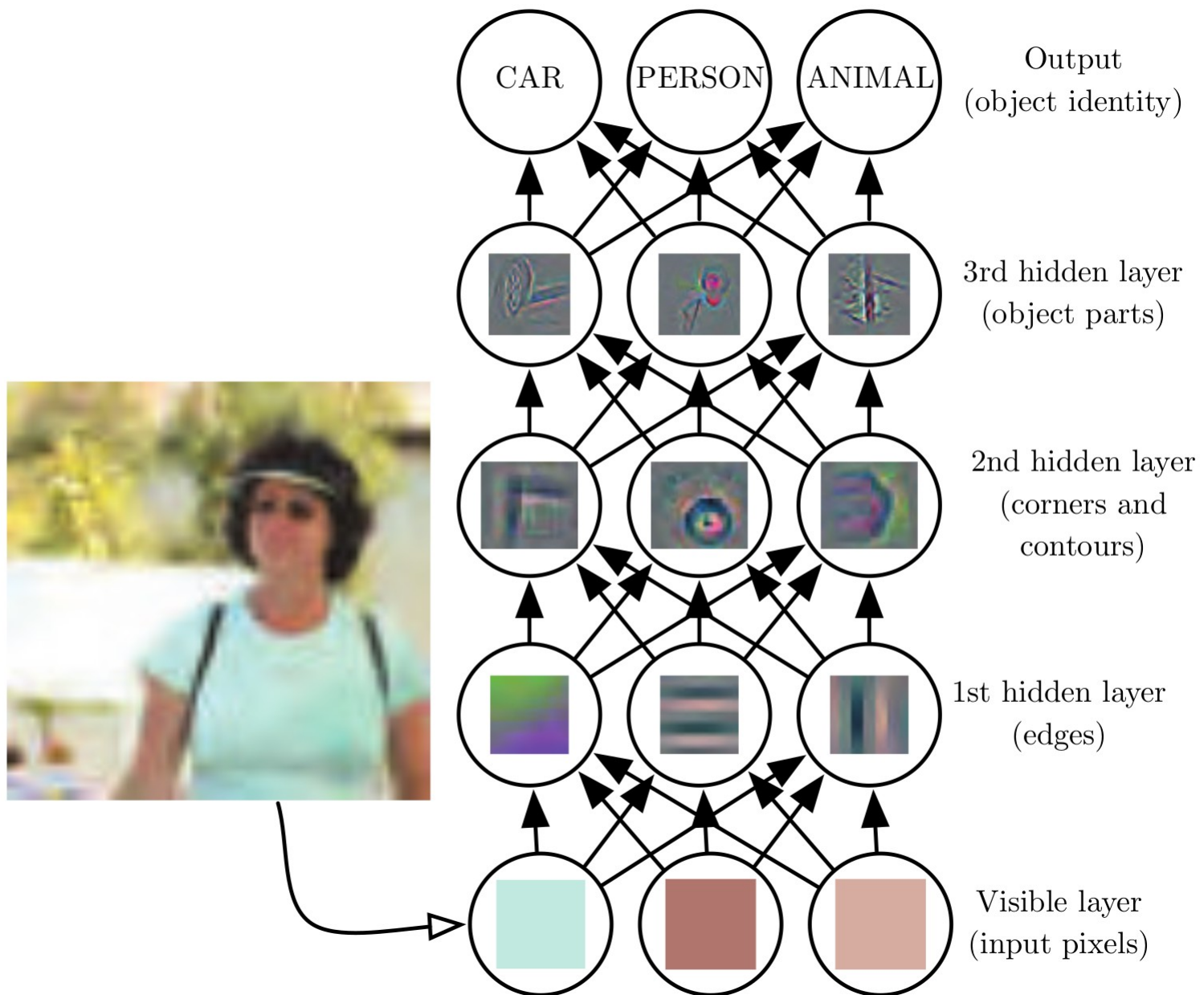
- Inspired by the neurons in the brain
 - McCulloch and Pitts (1943), calculate logical propositions using simple neuron-like units
 - Rosenblatt's perceptron (1958), first algorithm for learning weights
- Resurgence of interest in 1980s
 - Connectionism/parallel distributed processing
 - Backpropagation rediscovered (1986)
 - LeCun's convolutional net (1998)
- Deep learning (2006-present)
 - Hinton (2006), a technique to train deep networks
 - AlexNet (2012), winner of ImageNet challenge





What is deep learning?

- Deep neural networks
 - Many layers
 - Up to 100s
 - Build a hierarchical representation
 - Representations that are expressed in terms of other, simpler representations
- Quite successful in many domains
 - e.g., speech recognition, image segmentation, machine translation
 - Led to significant advances in accuracy



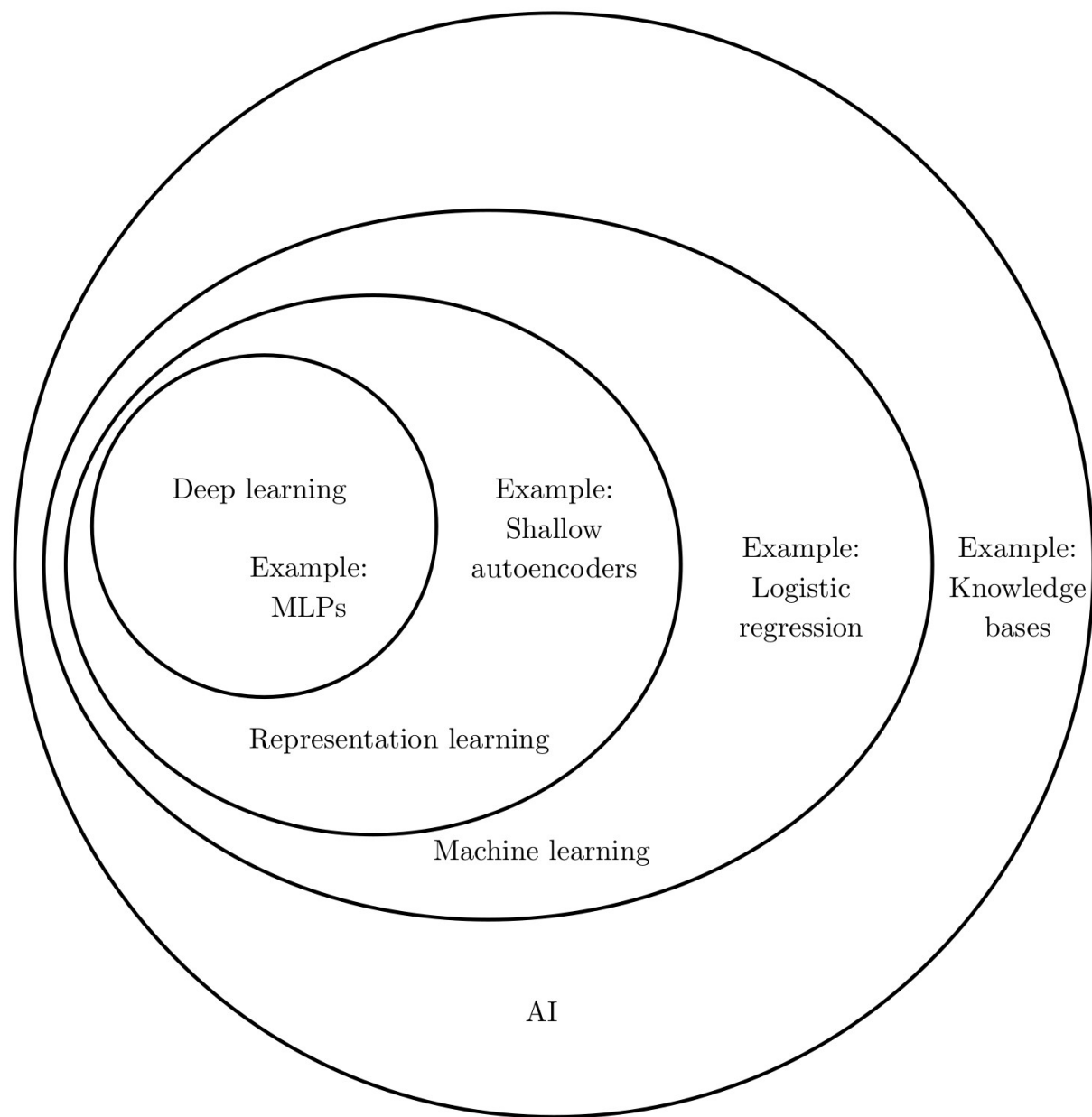
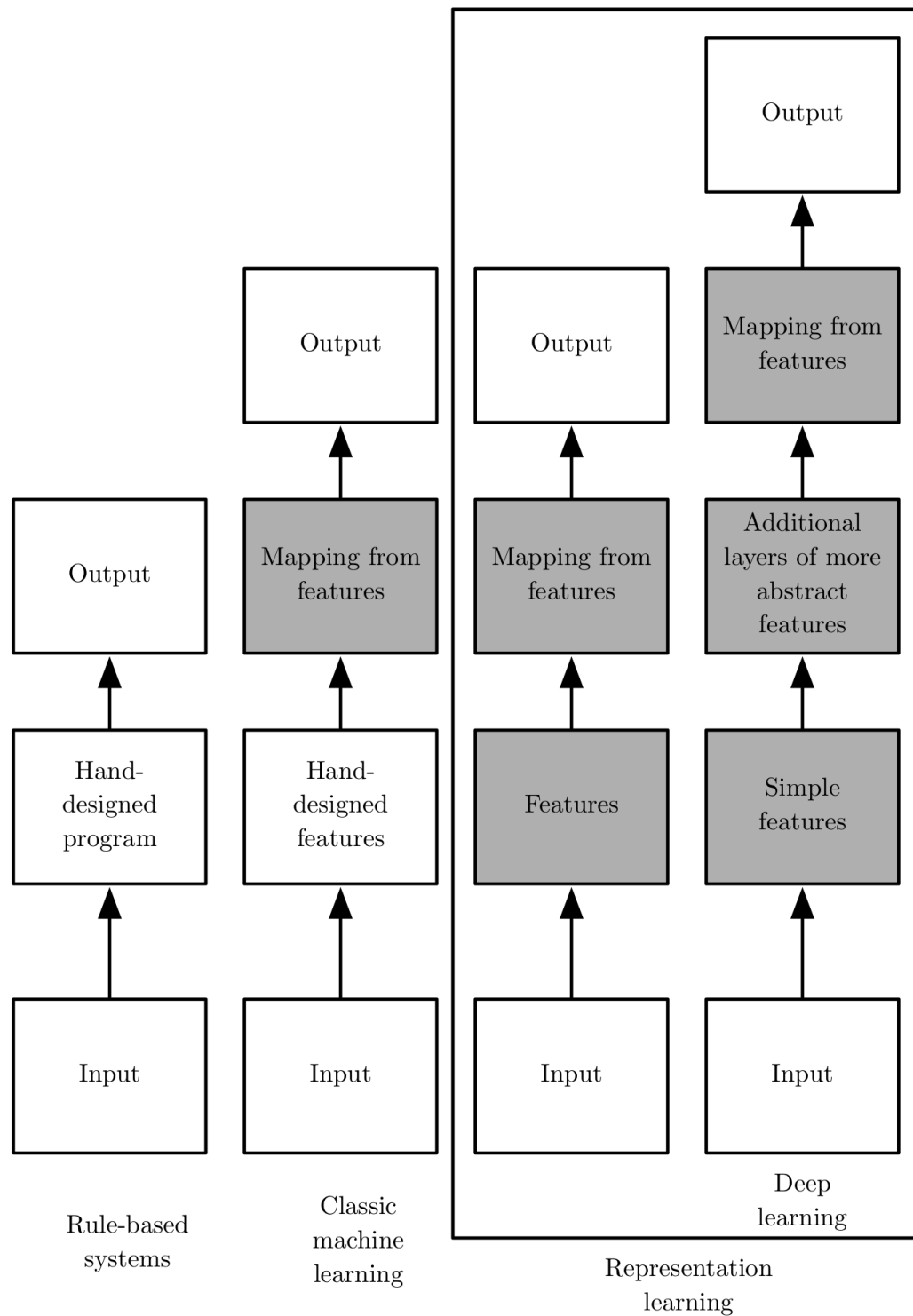


Figure 1.4: A Venn diagram showing how deep learning is a kind of representation learning, which is in turn a kind of machine learning, which is used for many but not all approaches to AI. Each section of the Venn diagram includes an example of an AI technology.



What made this possible?

- There hasn't been huge theoretical advances in neural network theory
 - However, deep learning emerged as a powerful technique
 - Why?
- Various factors
 - More **data**
 - More **computational power**
 - GPUs
 - Better **activation functions and initialization techniques**
 - ReLU
 - Better **optimization techniques**
 - Variants of stochastic gradient descent like RMSProp, Adam

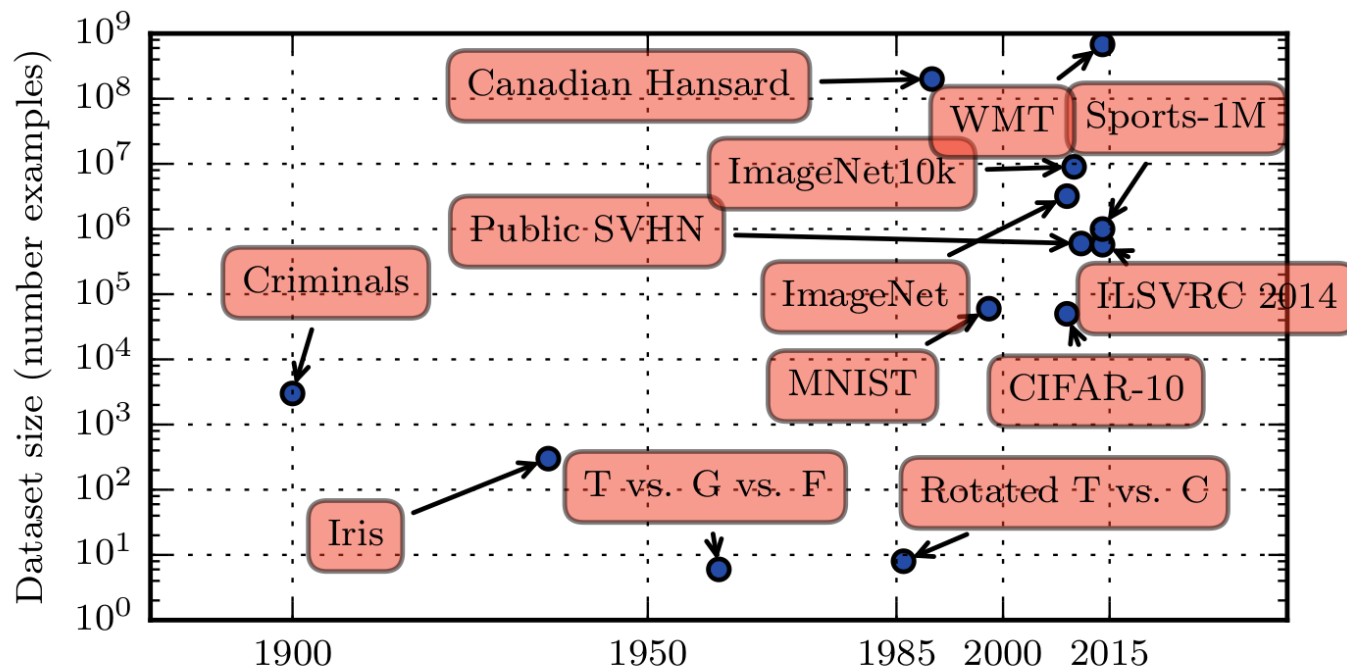


Figure 1.8: Increasing dataset size over time.

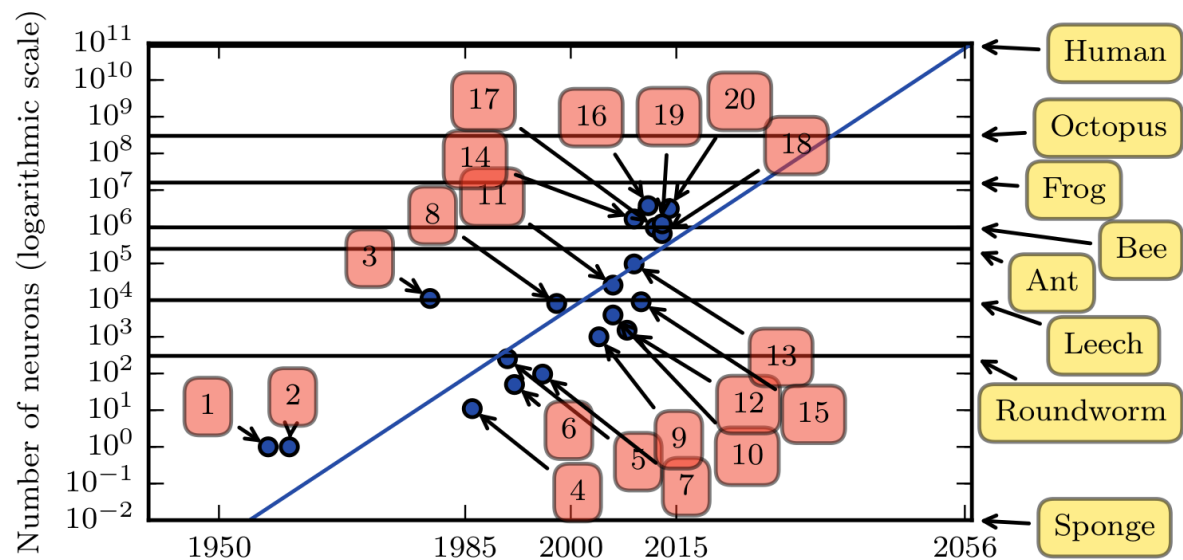


Figure 1.11: Increasing neural network size over time. Since the introduction of hidden units, artificial neural networks have doubled in size roughly every 2.4 years. Biological neural network sizes from Wikipedia (2015).

1. Perceptron (Rosenblatt, 1958, 1962)
2. Adaptive linear element (Widrow and Hoff, 1960)
3. Neocognitron (Fukushima, 1980)
4. Early back-propagation network (Rumelhart et al., 1986b)
5. Recurrent neural network for speech recognition (Robinson and Fallside, 1991)
6. Multilayer perceptron for speech recognition (Bengio et al., 1991)
7. Mean field sigmoid belief network (Saul et al., 1996)
8. LeNet-5 (LeCun et al., 1998b)
9. Echo state network (Jaeger and Haas, 2004)
10. Deep belief network (Hinton et al., 2006)
11. GPU-accelerated convolutional network (Chellapilla et al., 2006)
12. Deep Boltzmann machine (Salakhutdinov and Hinton, 2009a)
13. GPU-accelerated deep belief network (Raina et al., 2009)
14. Unsupervised convolutional network (Jarrett et al., 2009)
15. GPU-accelerated multilayer perceptron (Ciresan et al., 2010)
16. OMP-1 network (Coates and Ng, 2011)
17. Distributed autoencoder (Le et al., 2012)
18. Multi-GPU convolutional network (Krizhevsky et al., 2012)
19. COTS HPC unsupervised convolutional network (Coates et al., 2013)
20. GoogLeNet (Szegedy et al., 2014a)

Summary

- Neural networks
 - Formulation
 - Terminology
 - Activation function
 - Example: XOR
- Brief history
 - Deep Learning
- Exercises
 - Show that without activation functions a multilayer network is equivalent to a single layer network

References

- [1] Goodfellow I., Bengio Y., Courville A. Deep Learning.
<https://www.deeplearningbook.org/>
- [2] Bishop C. Pattern Recognition and Machine Learning. Chapter 5.
- [3] http://www.rutherfordjournal.org/images/TAHC_rosenblatt-sepia.jpg