

Neural Networks – Part II
PHYS 250 (Autumn 2019) – Lecture 16

David Miller

Department of Physics and the Enrico Fermi Institute
University of Chicago

November 29, 2019

Outline

1 *Reminders*

- Reminders from Lecture 15

2 *Historical perspective*

- Brief History of Machine Learning Generally
- Brief History of Neural Networks

3 *Structure of Neural Networks*

- Single layer perceptron
- Training a single layer perceptron
- Training a Multi-Layer Perceptron (MLP)

Reminders from last time

We embarked on a whirlwind introduction to neural networks.

Neural networks and machine learning

- **Context and perspective**

- We discussed the general issue of training computers to **discover, identify, and analyze patterns** of interest in datasets
- Categorized tasks that make use of this idea: **classification, regression, generation, clustering, anomaly detection**

- **Neural networks as a tool**

- Introduced both the **modeling** perspective as well as the **biological** perspective on what a neural network achieves
- Described the **structure and function** of a neuron
- Began discussing the **mathematical properties** of a neural network

Today we will build our own networks! But first, I just wanted to follow-up on some points and questions from last time.

Outline

1 *Reminders*

- Reminders from Lecture 15

2 *Historical perspective*

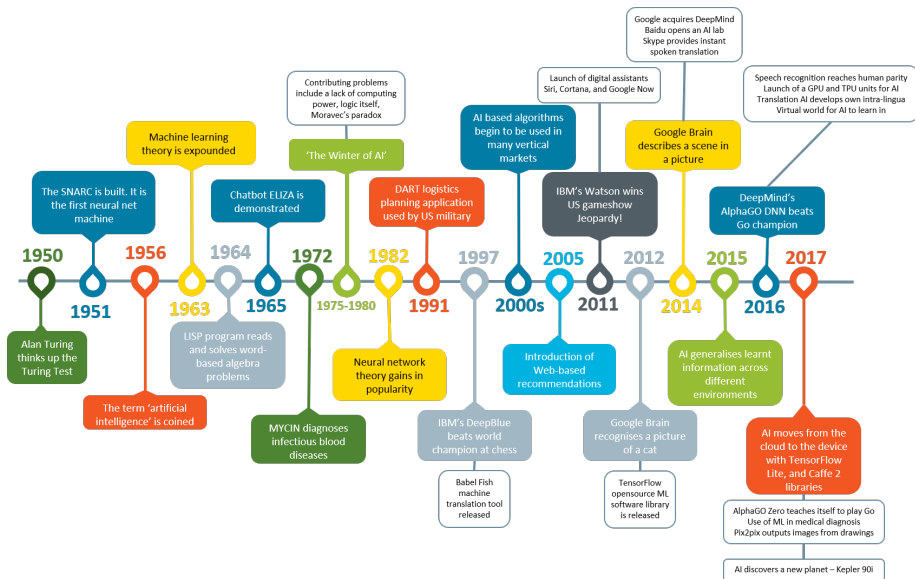
- Brief History of Machine Learning Generally
- Brief History of Neural Networks

3 *Structure of Neural Networks*

- Single layer perceptron
- Training a single layer perceptron
- Training a Multi-Layer Perceptron (MLP)

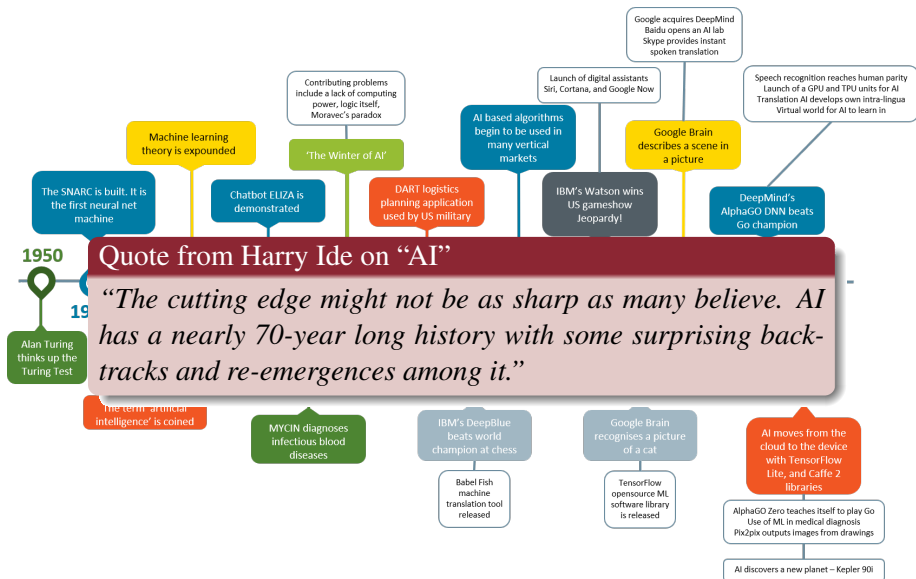
Brief history of machine learning

Taken from **Harry Ide on InnovationLaboratory.com (18 May 2018):**



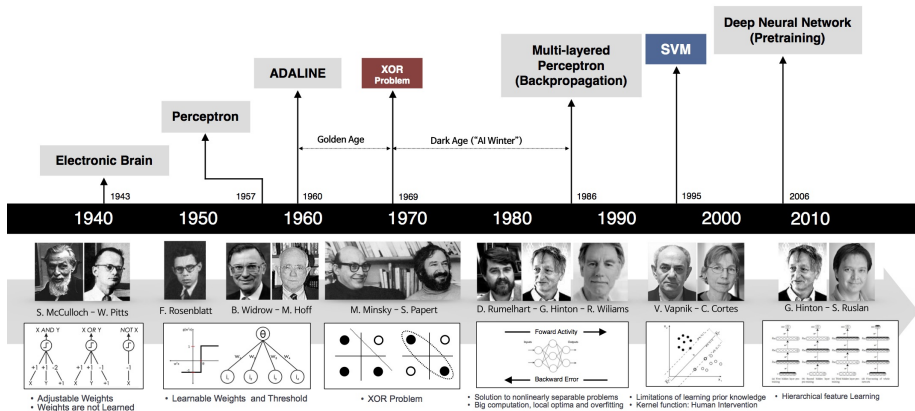
Brief history of machine learning

Taken from **Harry Ide on InnovationLaboratory.com** (18 May 2018):



Brief history of neural networks

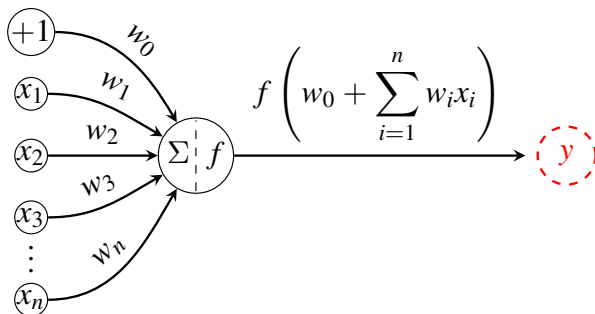
Taken from [this talk on SlideShare](#):



Outline

- 1 *Reminders*
 - Reminders from Lecture 15
- 2 *Historical perspective*
 - Brief History of Machine Learning Generally
 - Brief History of Neural Networks
- 3 *Structure of Neural Networks*
 - Single layer perceptron
 - Training a single layer perceptron
 - Training a Multi-Layer Perceptron (MLP)

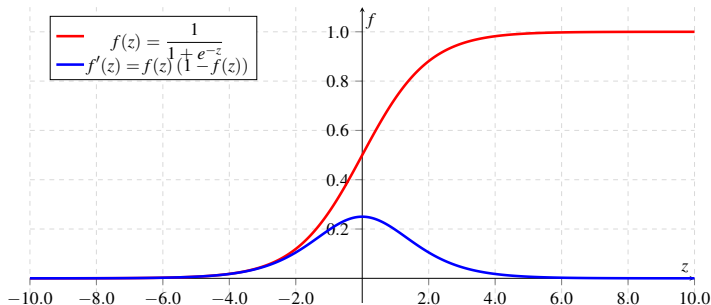
Single layer perceptron



- $\vec{x} = (x_1, x_2, \dots, x_n)$ is an input feature vector of length n
i.e. the attributes of the data, e.g. voltages
- $\vec{w} = (w_1, w_2, \dots, w_n)$ is the weight vector with w_0 reserved as a bias
 - becomes a matrix for multiple layers
- Σ indicates summation (or matrix mult.): $z = \sum w_i x_i$ ($x_0 = 1$)
- f is the activation function, or non-linearity: $f(z)$
- $y = f(z)$ is the output

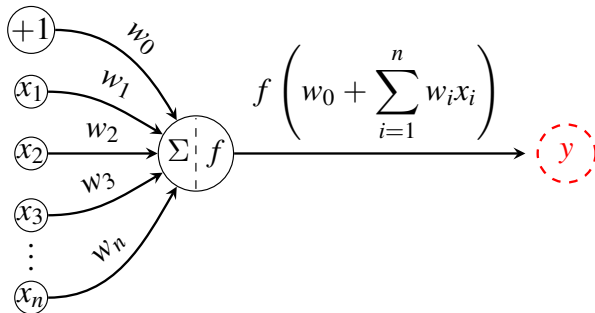
Sigmoid as activation function

As we discussed, a typical function for a **single layer perceptron** is the **sigmoid**.



Here, we plot both the function itself, as well as its derivative, since that will be important when evaluating the **backpropagation** of weights in order to update the neural network.

Training a single layer perceptron

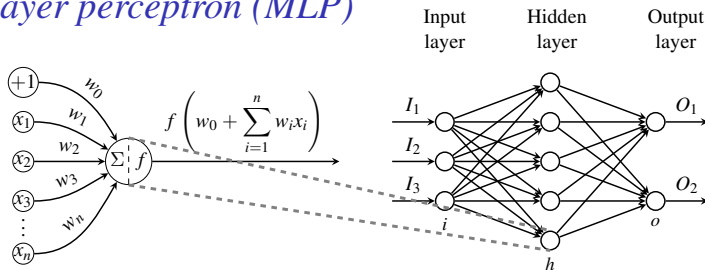


Given j objects \vec{x}_j in dataset, each with **known values of f** , d_j

- **Calculate the output:** $y_j = f(\vec{w} \cdot \vec{x}_j)$
- **Determine the error:** $\epsilon_j = d_j - y_j$
- **Update the weights:** $w_i^{\text{new}} = w_i + r(\epsilon_j \cdot \vec{x}_j)_i$

Choosing the learning rate r is where the derivative is used. It's not important for the single-layer perceptron, but is **essential** for a network.

Multi-layer perceptron (MLP)



Given j objects \vec{I}_j in dataset, each with features $\vec{I} = (I_1, I_2, \dots, I_n)$ and **known outputs \vec{d}_j at each output node o** , $\vec{d} = (d_1, d_2, \dots, d_o)$

- **Calculate the h outputs of hidden layer:** $v_h = f\left(\sum_i w_{ih} I_i\right)$
- **Calculate the o outputs of output layer:** $y_o = f\left(\sum_h w_{ho} v_h\right)$
- **Determine the error at output each node o :** $\epsilon_o = d_o - y_o$
- **Determine the total error for data object j :** $\mathcal{E}_j = \frac{1}{2} \sum_o \epsilon_o^2$
- **Determine change in weights for output neuron y_o :** $\Delta w_{oh} = -\eta \frac{\partial \mathcal{E}}{\partial z_o} v_h = \eta \epsilon_o f'(z_o)$

LeCun, Bengio, Hinton, "Deep learning"

Nature volume 521, pages 436-444 (28 May 2015)

