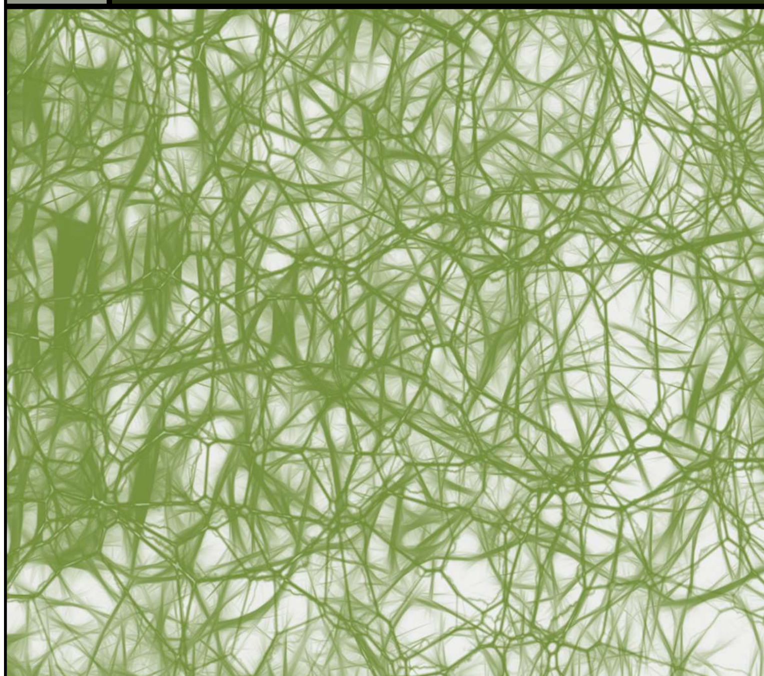
Department of Computer Science
University of Bristol

COMSM0045 – Applied Deep Learning 2020/21

comsm0045-applied-deep-learning.github.io



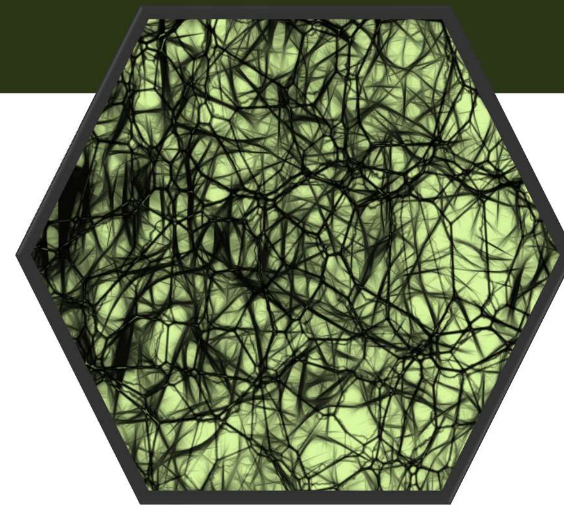
Lecture 01

BASICS OF ARTIFICIAL NEURAL NETWORKS

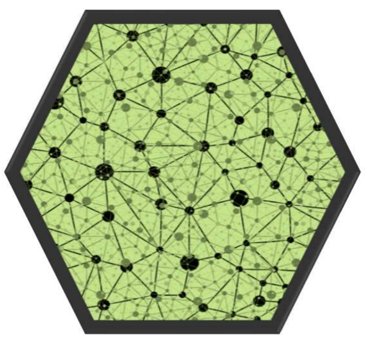
Tilo Burghardt | tilo@cs.bris.ac.uk

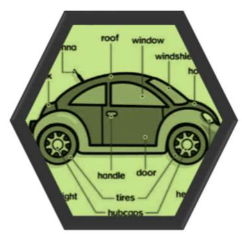
35 Slides

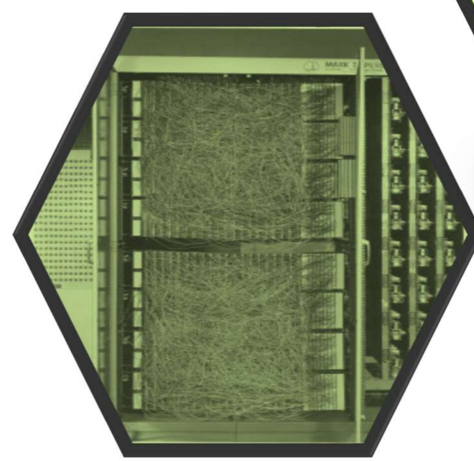
Agenda for Lecture 1



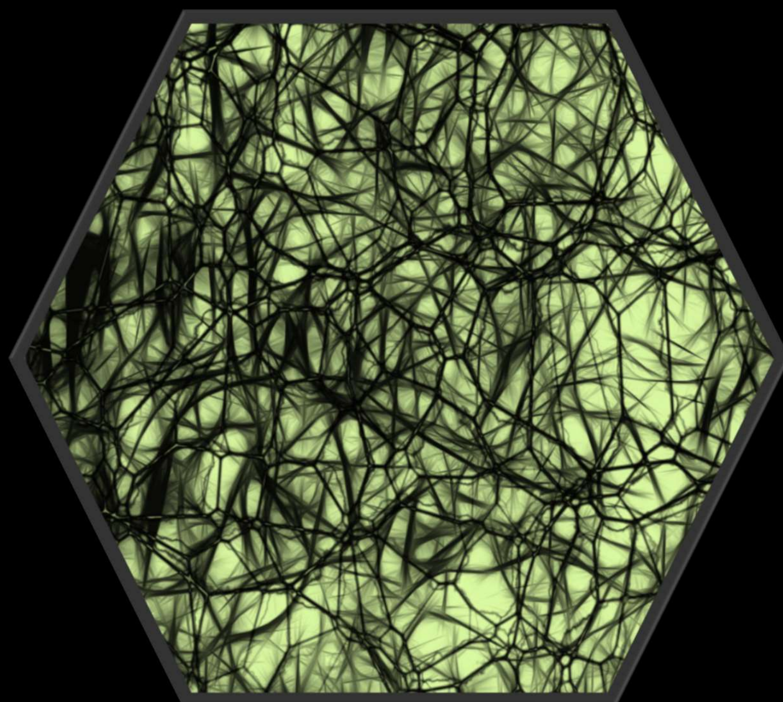
- Neurons and their Structure
- Single & Multi Layer Perceptron
- Basics of Cost Functions
- Gradient Descent and Delta Rule
- Notation and Structure of
  Deep Feed-Forward Networks



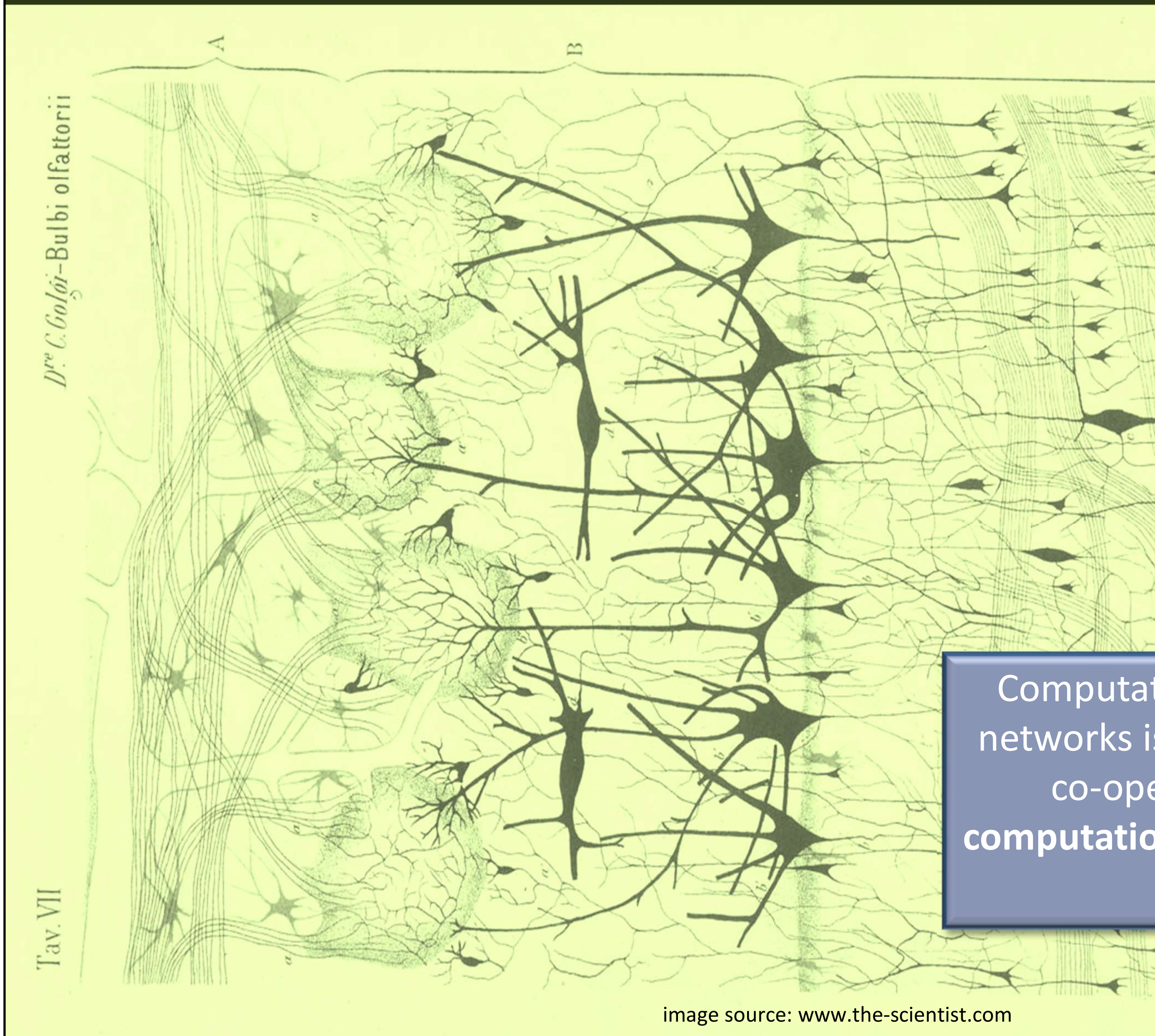


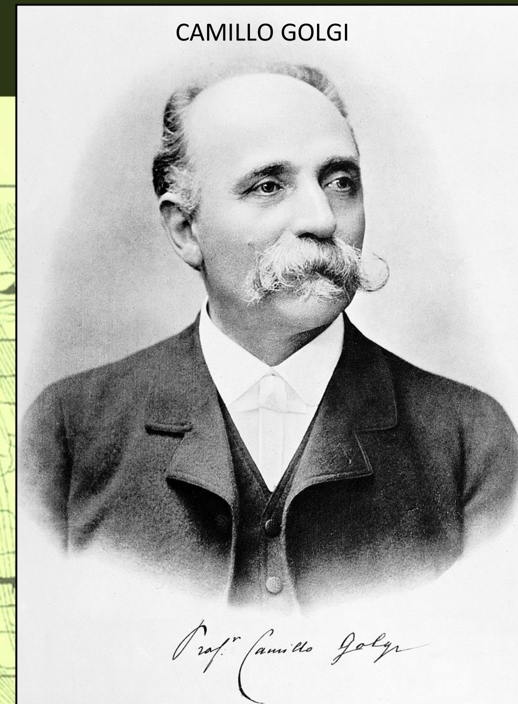


Biological Inspiration



Golgi’s first Drawings of Neurons

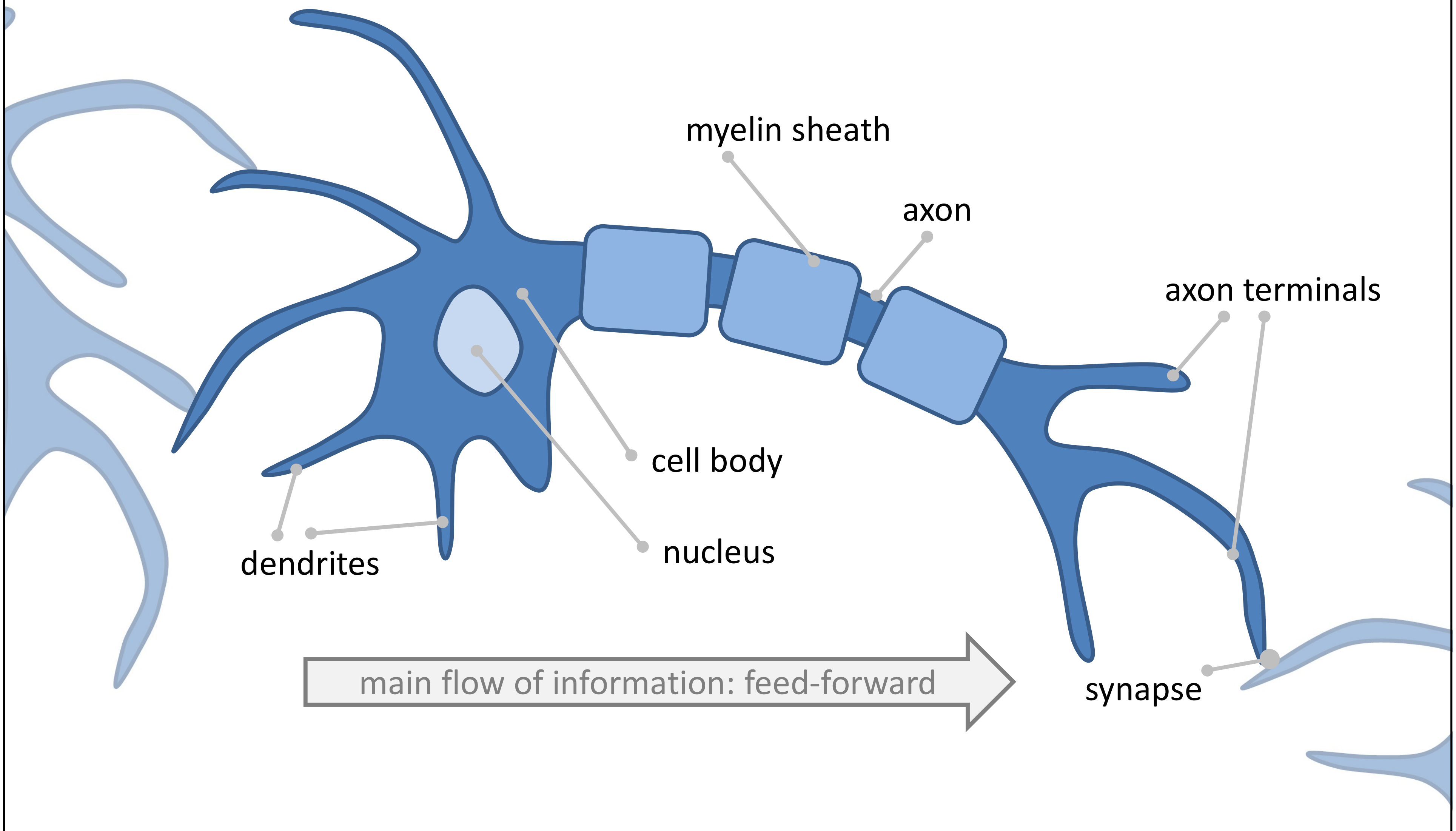




Computation in biological neural networks is delivered based on the co-operation of **individual computational components**, namely neuron cells.

image source: www.the-scientist.com

Schematic Model of a Neuron



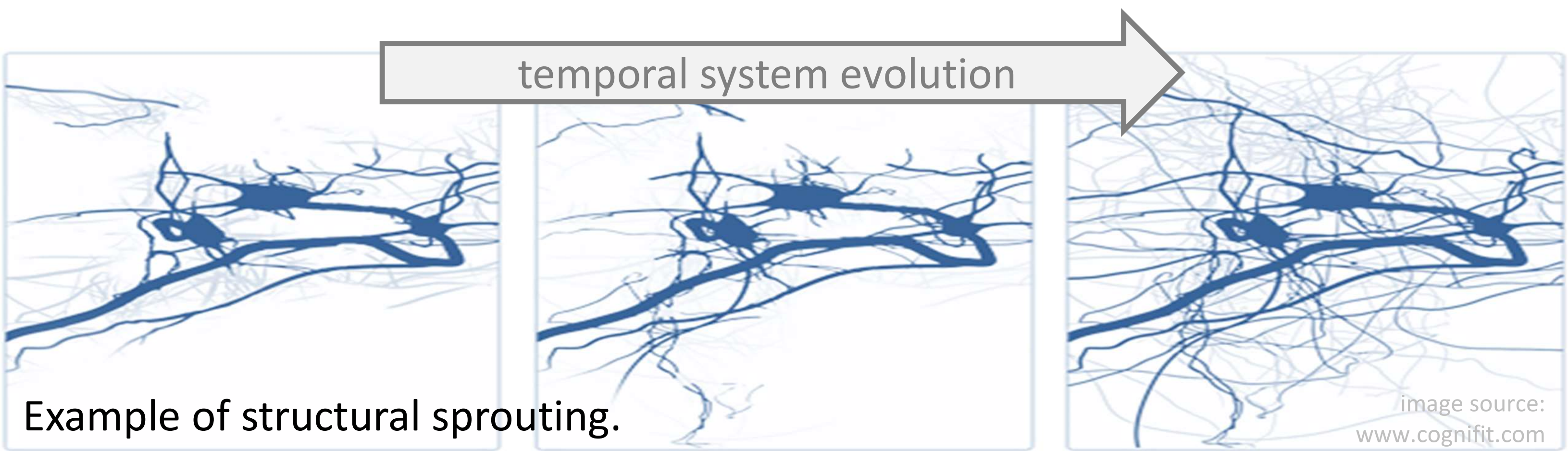
Pavlov and Assistant Conditioning a Dog

An environment can **condition** the behaviour of biological neural networks leading to the incorporation of new information.

image source: www.psysci.co

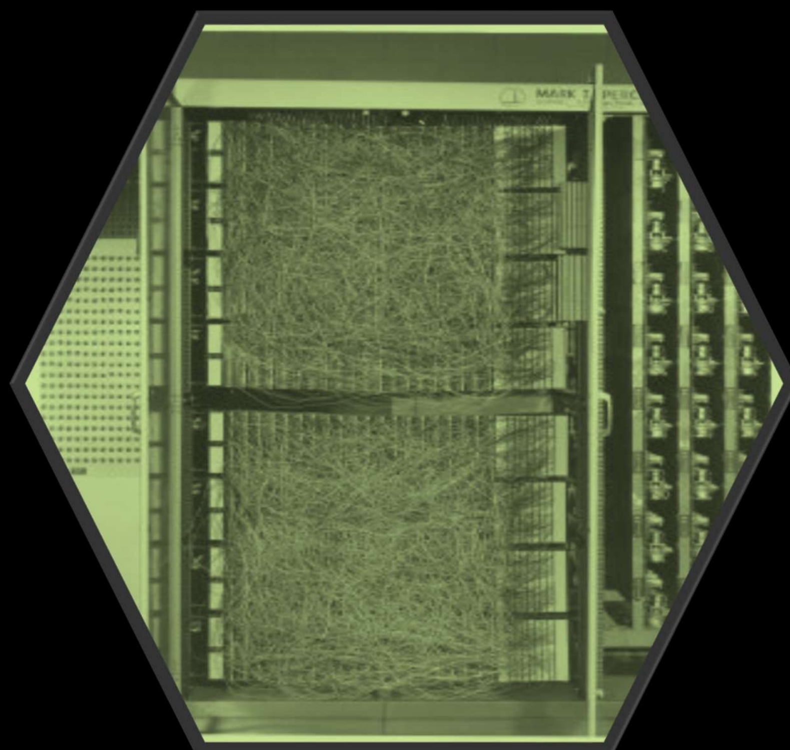
Neuro-Plasticity

- plasticity refers to a system's ability to adapt structure and/or behaviour to accommodate new information
- the brain shows various forms of plasticity:
  - natural forms include synaptic plasticity (mainly chemical), structural sprouting (growth), rerouting (functional changes), and neurogenesis (new neurons)

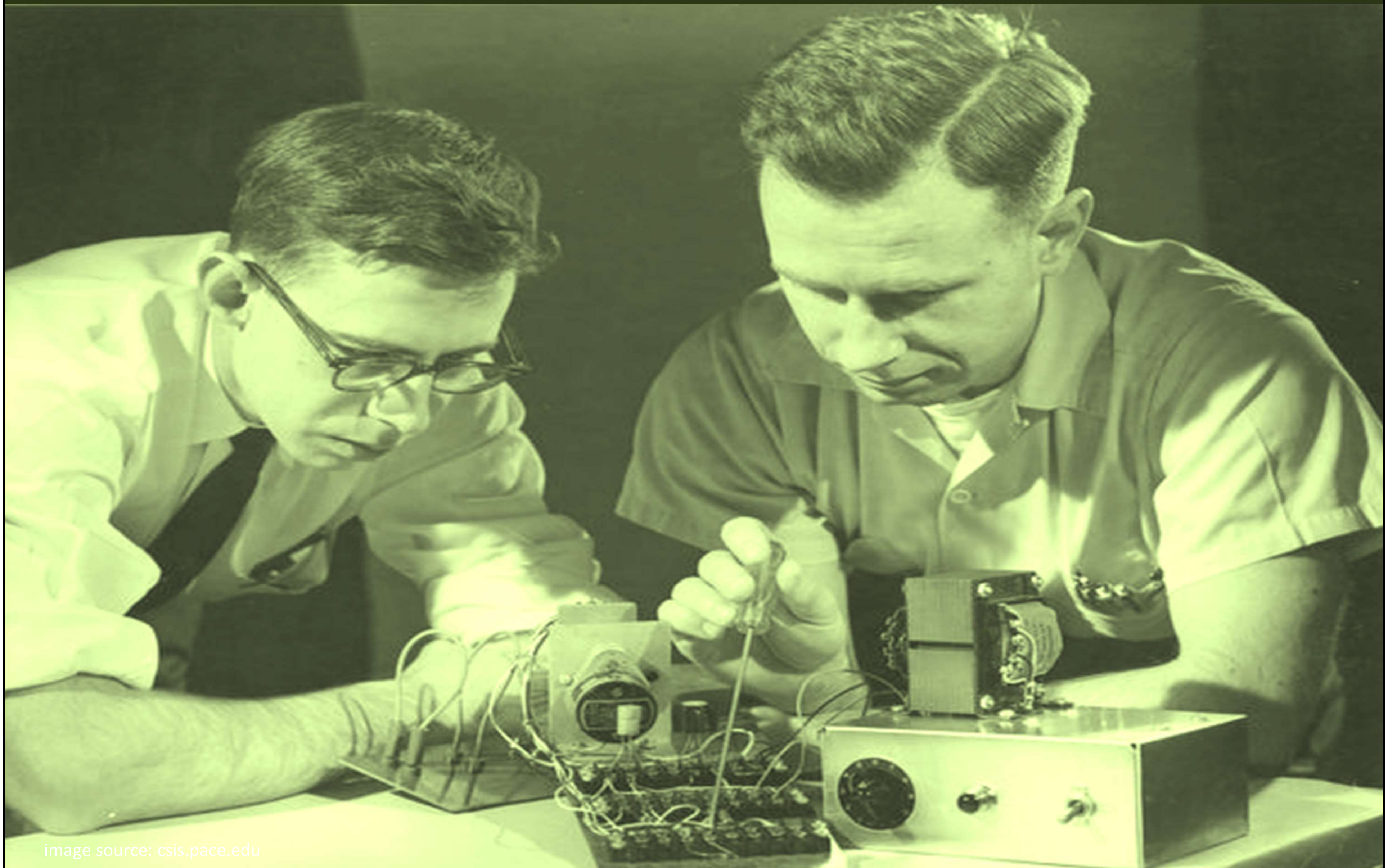


Example of structural sprouting.

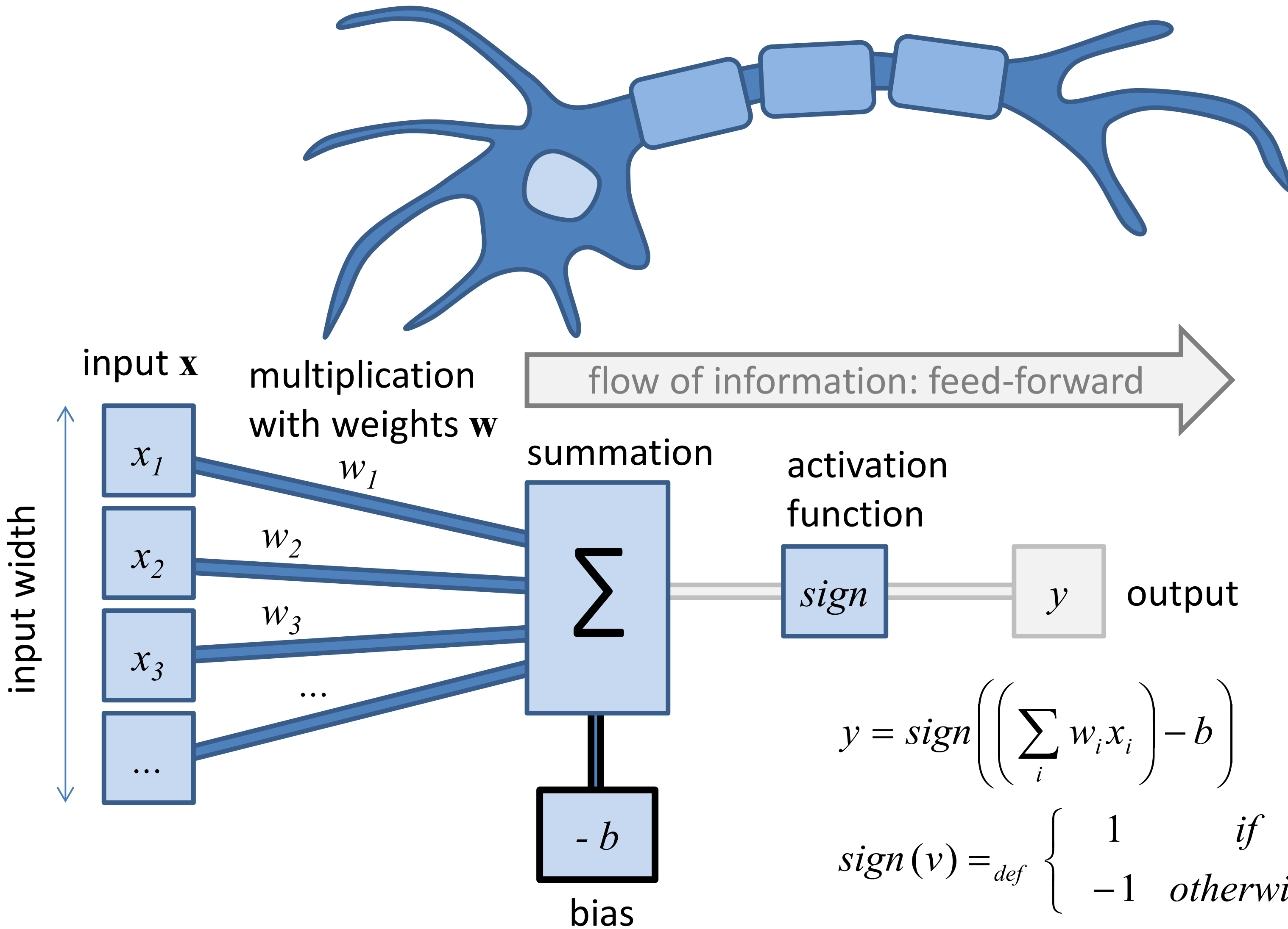
Artificial Feed-forward Networks



Rosenblatt's (left) development of the Perceptron (1950s)



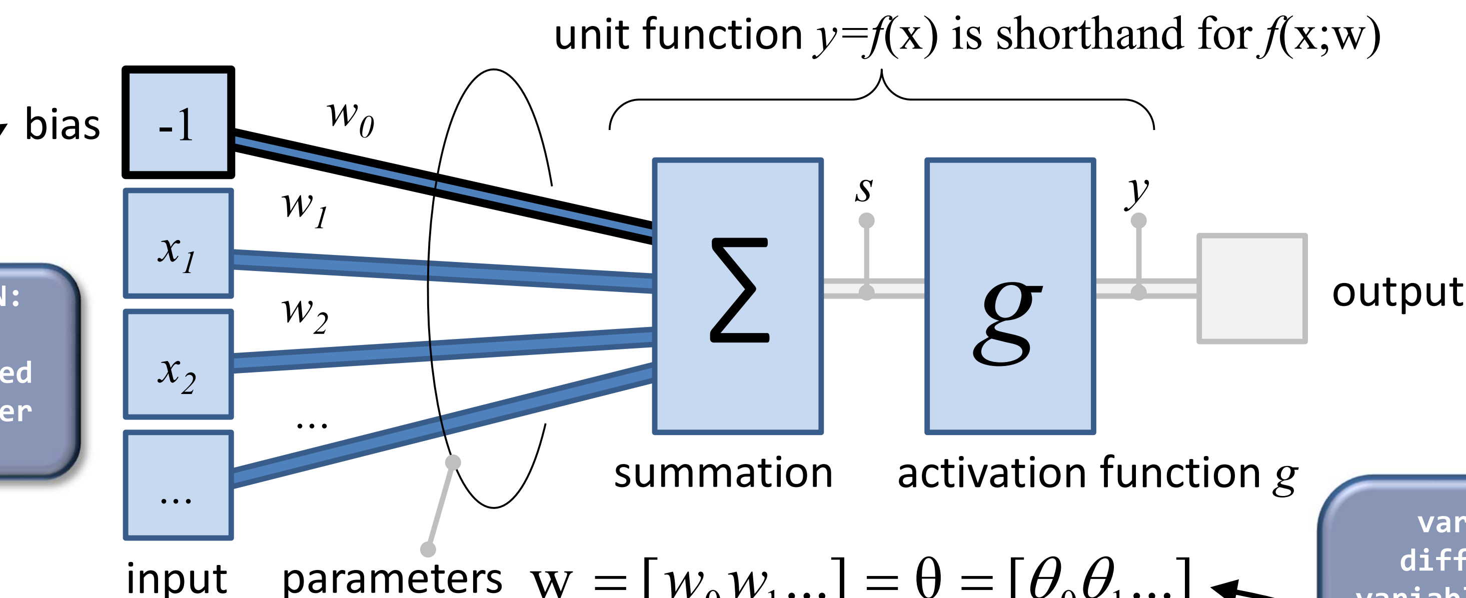
Simplification of a Neuron to a Computational Unit



$$y = sign\left(\left(\sum_i w_i x_i\right) - b\right)$$

$$sign(v) =_{def} \begin{cases} 1 & if \quad v \geq 0 \\ -1 & otherwise \end{cases}$$

Notational Details for the Perceptron



$$\mathrm{w} = [w_0 w_1 \ldots] = \theta = [\theta_0 \theta_1 \ldots]$$

CONVENTION: bias is incorporated in parameter vector

various different variable names are used for parameters, most often we will use w

$$\underbrace{y}_{\textit{unit output}} = \underbrace{sign}_{\textit{activation function}} (\underbrace{\mathrm{w}^T}_{\textit{parameters}} \underbrace{\mathrm{x}}_{\textit{input}}) = g(\mathrm{w}^T \mathrm{x})$$

$$= \underbrace{f}_{\textit{unit function}} (\underbrace{\mathrm{x}}_{\textit{input}} ; \underbrace{\mathrm{w}}_{\textit{parameters}})$$

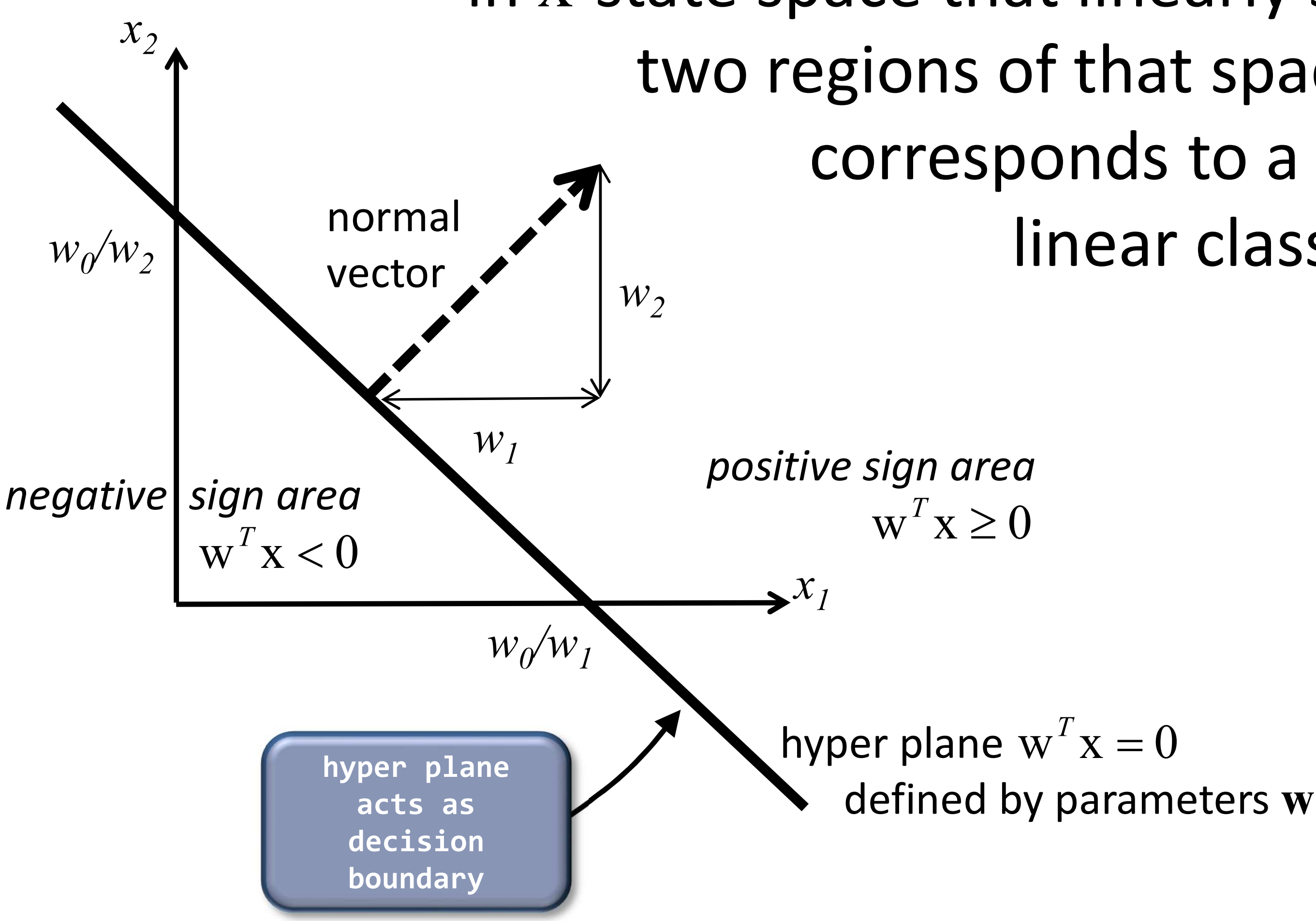
NOTATION: italic font refers to scalars

NOTATION: semicolon separates input (left) from parameters (right)

NOTATION: a minor letter in non-italic font refers to a vector, a capital letter in non-italic font would refer to a matrix or vector set

Geometrical Interpretation of the State Space

The basic Perceptron defines a hyper plane $0 = \mathrm{w}^T\mathrm{x}$ in x-state space that linearly separates two regions of that space (which corresponds to a two-class linear classification)



Basic Perceptron (Supervised) Learning Rule

- **Idea:** whenever the system produces a misclassification with current weights, adjust weights by $\Delta \mathrm{w}$ towards a better performing weight vector:

$$\underbrace{\Delta \mathrm{w}}_{update} = \begin{cases} \eta \, f^*(\mathrm{x}) \, \mathrm{x} & if \quad \overbrace{f^*(\mathrm{x})}^{ground\,truth} \neq \overbrace{f(\mathrm{x})}^{actual\,output} \\ 0 & otherwise \end{cases}$$

... where η is the learning rate.

Training a Single-Layer Perceptron

Compare Output and Ground Truth

$$f^*(\mathrm{x}) \neq f(\mathrm{x})\,?$$

Compute Output

$$f(\mathrm{x}) = sign(\mathrm{w}^T \mathrm{x})$$

Adjust Weights

$$\Delta w_i = \begin{cases} \eta\, f^*(\mathrm{x})\, x_i & if \quad f^*(\mathrm{x}) \neq f(\mathrm{x}) \\ 0 & otherwise \end{cases}$$

Consider Next (Training) Input Pair

$$\left(\mathrm{x}, f^*(\mathrm{x})\right)$$

Perceptron Learning Example: OR

Perceptron Training Attempt of **OR** using

$$\Delta \mathrm{w} = \eta\,(f^{*}(\mathrm{x}) - f(\mathrm{x}))\,\mathrm{x}; \quad \eta = 0.5$$

learning progress sampling some $(\mathbf{x}, f^*)$

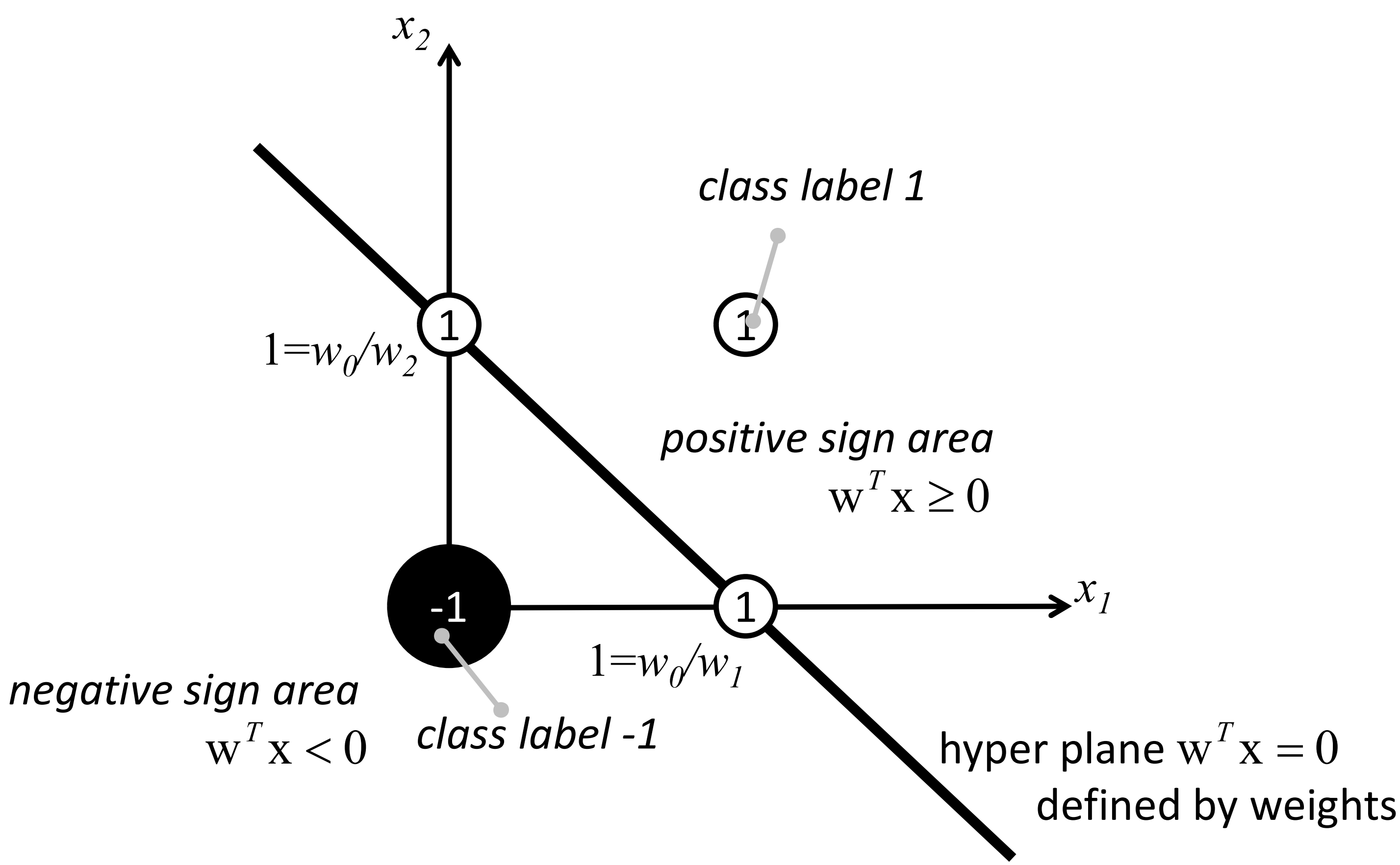
x_0	x_1	x_2	*parameters* w	f	f^*	*update* Δw
-1	0	0	(0,0,0)	1	-1	(1,0,0)
-1	1	0	(1,0,0)	-1	1	(-1,1,0)
-1	0	0	(0,1,0)	1	-1	(1,0,0)
-1	0	1	(1,1,0)	-1	1	(-1,0,1)
-1	0	0	(0,1,1)	1	-1	(1,0,0)
-1	0	1	(1,1,1)	1	1	(0,0,0)
-1	1	0	(1,1,1)	1	1	(0,0,0)
-1	1	1	(1,1,1)	1	1	(0,0,0)
-1	0	0	(1,1,1)	-1	-1	(0,0,0)
...	...	...	...	...	...	...

OR

x_1	x_2	f^*
0	0	-1
0	1	1
1	0	1
1	1	1

encoding could be changed to traditional value *0* by adjusting the output of the sign function to *0;* training algorithm still valid

Geometrical Interpretation of **OR** Space Learned



Larger Example Visualisation

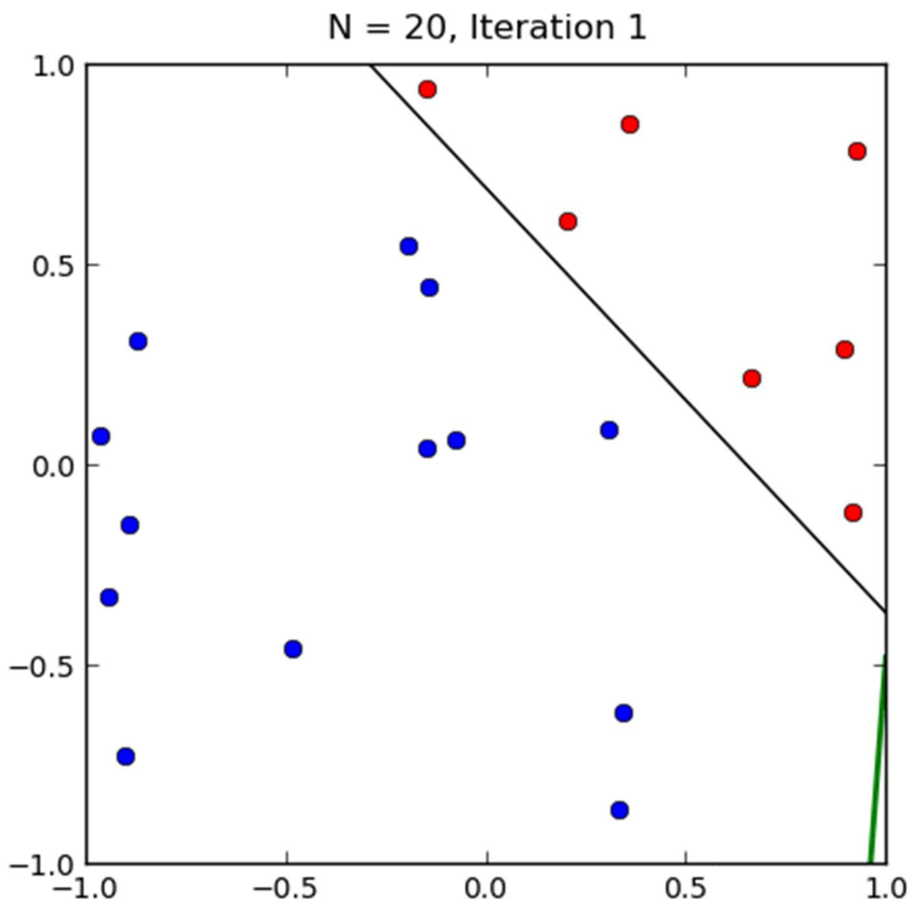


image source: datasciencelab.wordpress.com

Cost Functions

Cost (or Loss) Functions

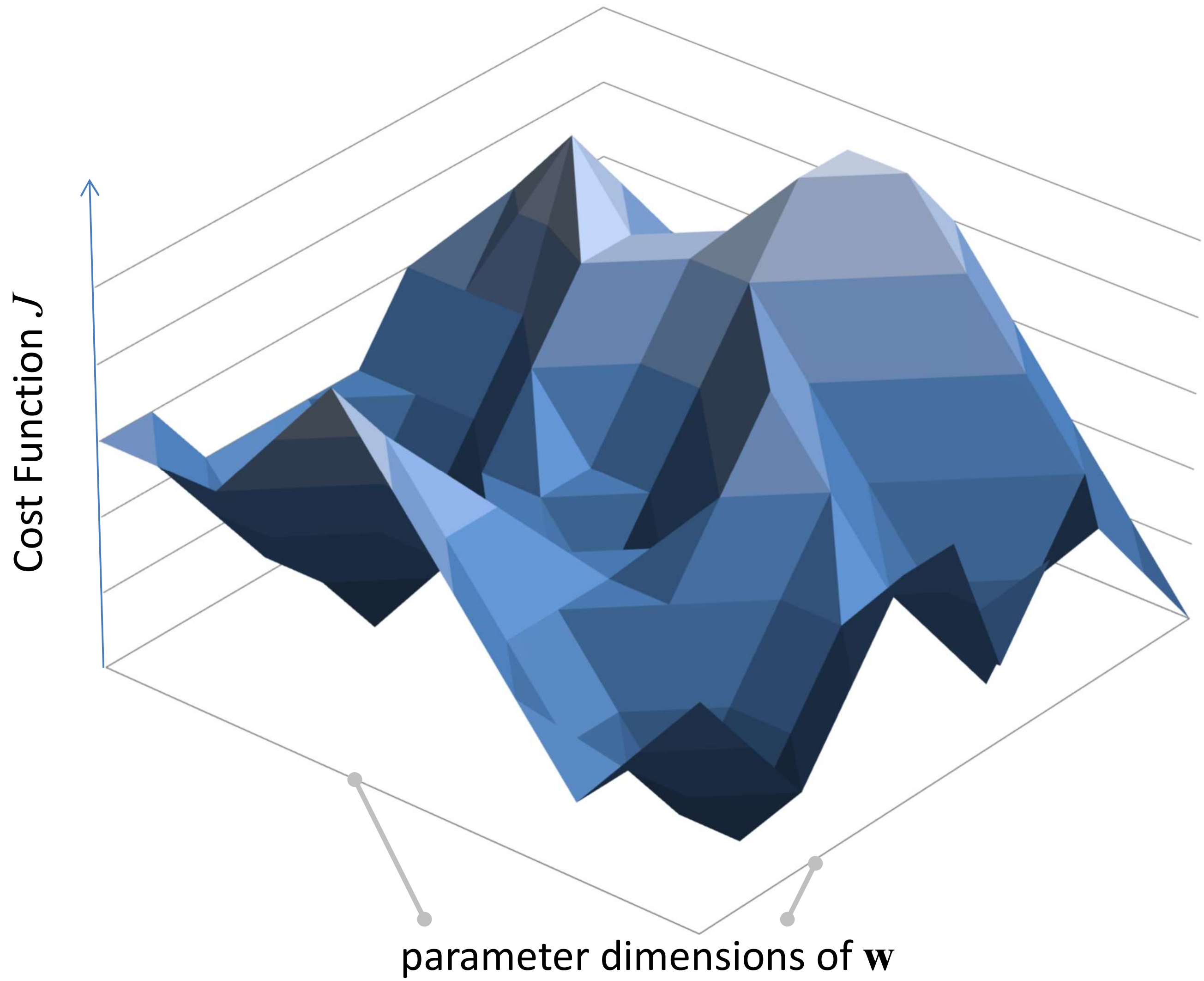
Idea: Given a set $\mathbf{X}$ of input vectors $\mathbf{x}$ of one or more variables and a parameterisation $\mathbf{w}$, a Cost Function is a map $\boldsymbol{J}$ onto a real number representing a cost or loss associated with the input configurations. (Negatively related to 'goodness of fit'.)

Expected Loss: $J(\mathrm{X};\mathrm{w}) = \mathrm{E}_{(\mathrm{x}, f^*(\mathrm{x})) \sim p} L(f(\mathrm{x};\mathrm{w}), f^*(\mathrm{x}))$

Empirical Risk: $J(\mathrm{X};\mathrm{w}) = \dfrac{1}{|\mathrm{X}|} \sum_{\mathrm{x} \in \mathrm{X}} L(f(\mathrm{x};\mathrm{w}), f^*(\mathrm{x}))$

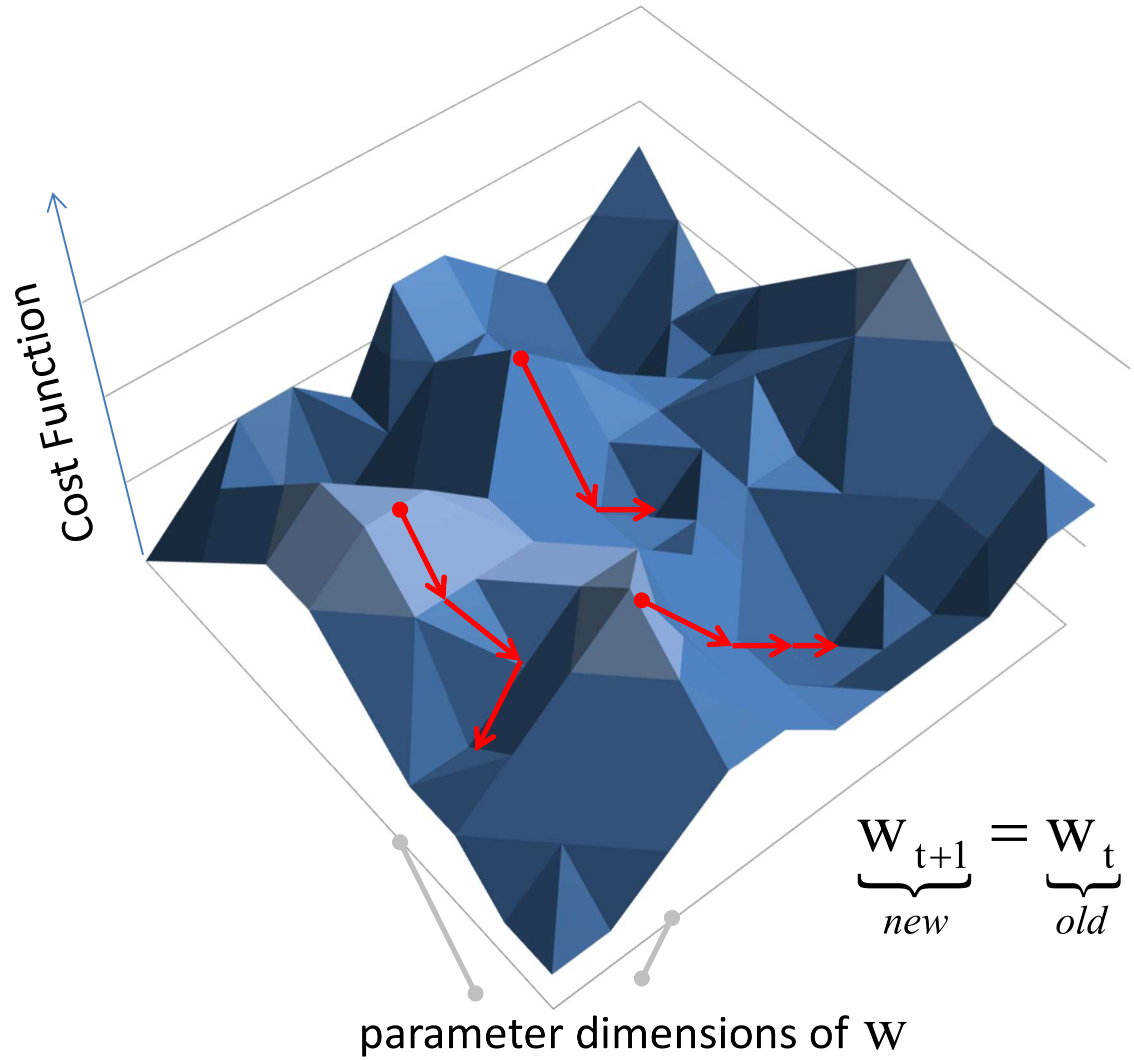
MSE Example: $MSE_{loss} = \underbrace{J(\mathrm{X};\mathrm{w})}_{loss\ function} = \dfrac{1}{|\mathrm{X}|} \sum_{\mathrm{x} \in \mathrm{X}} \underbrace{\left(f(\mathrm{x};\mathrm{w}) - f^*(\mathrm{x})\right)^2}_{per-example\ loss\ function}$

Energy Landscapes over Parameter Space



Steepest Gradient Descent

Idea of ‘Steepest’ Gradient Descent



$$\underbrace{\mathrm{w}_{t+1}}_{\textit{new}} = \underbrace{\mathrm{w}_{t}}_{\textit{old}} - \underbrace{\eta}_{\textit{learning rate}} \underbrace{\nabla J(\mathrm{X};\mathrm{w}_{t})}_{\textit{steepest gradient}}$$

The Delta Rule

$$J(\mathrm{X};\mathrm{w}) = \frac{1}{2|\mathrm{X}|}\sum_{\mathrm{x}\in\mathrm{X}}\left(\mathrm{w}^{\mathrm{T}}\mathrm{x} - f^{*}(\mathrm{x})\right)^{2}$$

MSE-type cost function with identity function as activation function

$$\Delta\mathrm{w} = -\eta\nabla J(\mathrm{X};\mathrm{w})$$

weight vector change is modelled as a move along the steepest descent

change for a single weight w_k

$$\Delta w_k = -\eta\frac{\partial J(\mathrm{X};\mathrm{w})}{\partial w_k} = -\frac{\eta}{|\mathrm{X}|}\sum_{\mathrm{x}\in\mathrm{X}} x_k\left(\mathrm{w}^{\mathrm{T}}\mathrm{x} - f^{*}(\mathrm{x})\right)$$

...and for a single sample...

$$\Delta w_k = -\eta\, x_k\left(\mathrm{w}^{\mathrm{T}}\mathrm{x} - f^{*}(\mathrm{x})\right)$$

$$\Delta\mathrm{w} = -\eta\,\mathrm{x}\underbrace{\left(\mathrm{w}^{\mathrm{T}}\mathrm{x} - f^{*}(\mathrm{x})\right)}_{\delta\text{ is the error derivative}}$$

this term looks similar to the Perceptron learning rule

$$\Delta\mathrm{w} = -\eta\,\mathrm{x}\,\delta$$

also known as The Delta Rule (Widrow & Hoff, 1960)

Linear Separability

Basic Learning Example: XOR

Perceptron Training Attempt of **XOR** using

$$\Delta \mathrm{w} = \eta\,(f^*(\mathrm{x}) - f(\mathrm{x}))\,\mathrm{x}; \quad \eta = 0.5$$

learning progress sampling some (x, f^*)

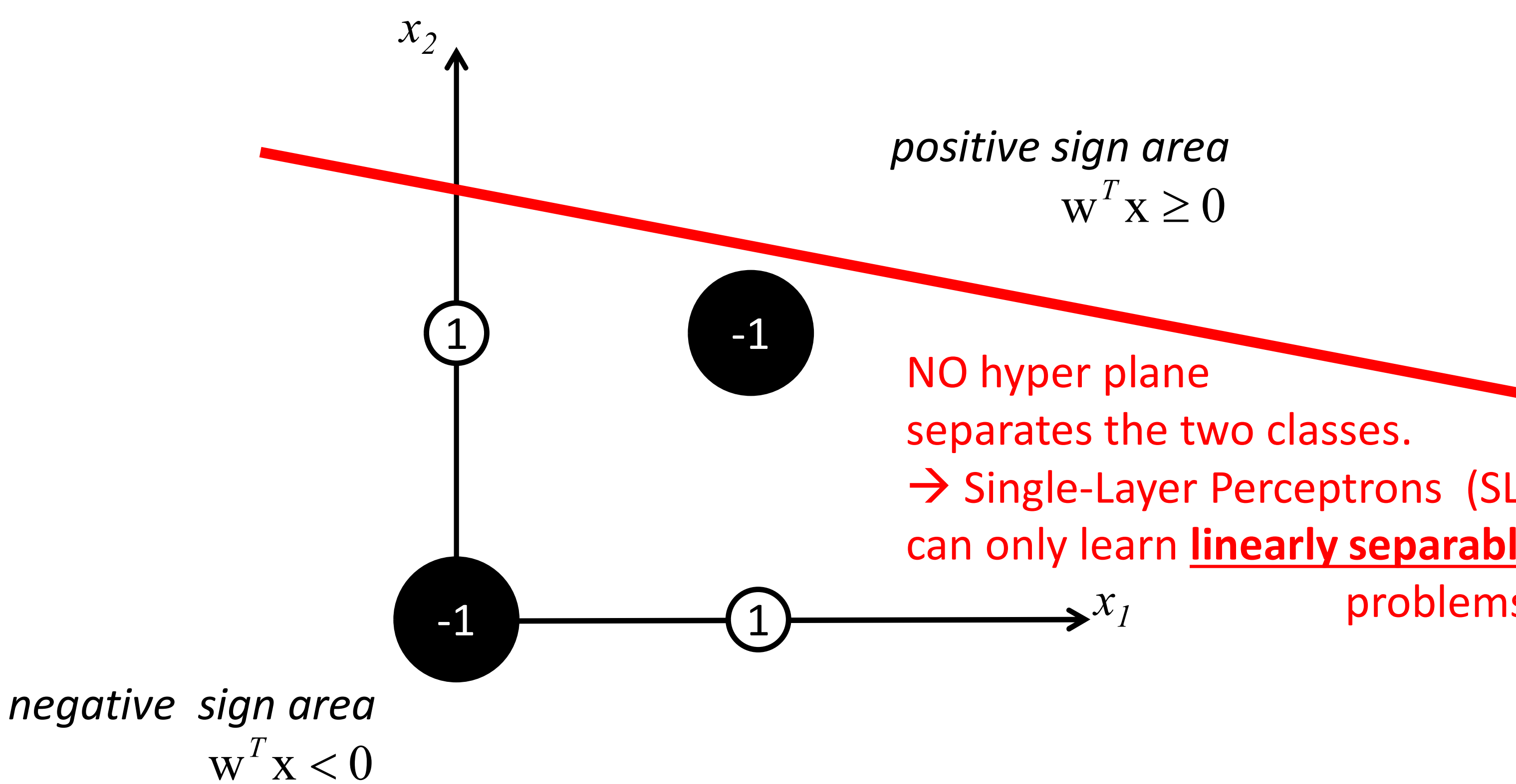
x_0	x_1	x_2	*parameters*	f	f^*	*update*
-1	0	0	(0,0,0)	1	-1	(1,0,0)
-1	1	0	(1,0,0)	-1	1	(-1,1,0)
-1	0	0	(0,1,0)	1	-1	(1,0,0)
-1	0	1	(1,1,0)	-1	1	(-1,0,1)
-1	0	0	(0,1,1)	1	-1	(1,0,0)
-1	0	1	(1,1,1)	1	1	(0,0,0)
-1	1	0	(1,1,1)	1	1	(0,0,0)
-1	1	1	(1,1,1)	1	-1	(1,-1,-1)
-1	1	0	(1,0,0)	-1	1	(-1,1,0)
-1	0	1	(1,1,0)	-1	1	(-1,0,1)
...	...	...	...	...	...	...

XOR

x_1	x_2	f^*
0	0	-1
0	1	1
1	0	1
1	1	-1

Will the learning process ever produce a solution?

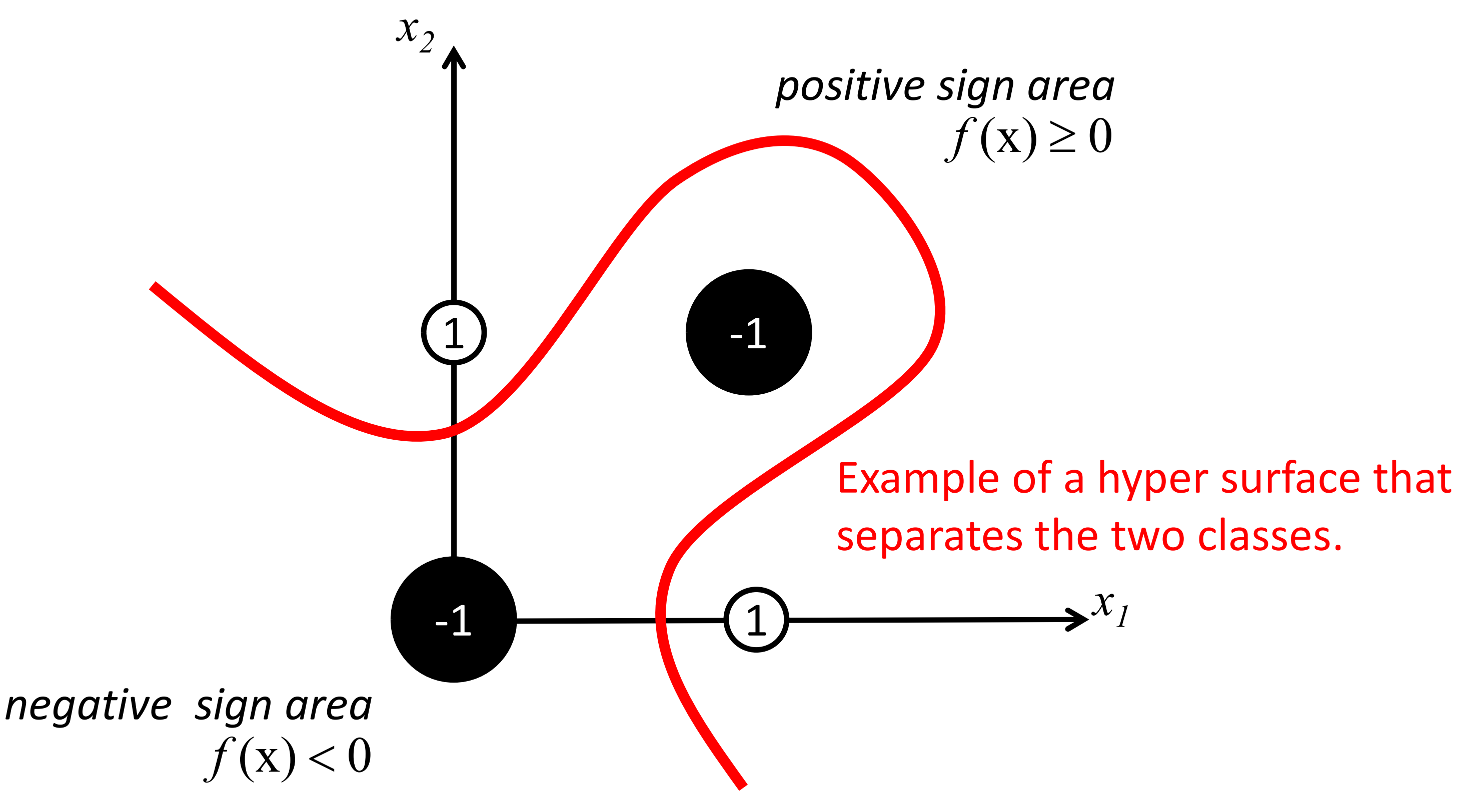
Geometrical Interpretation of XOR Space



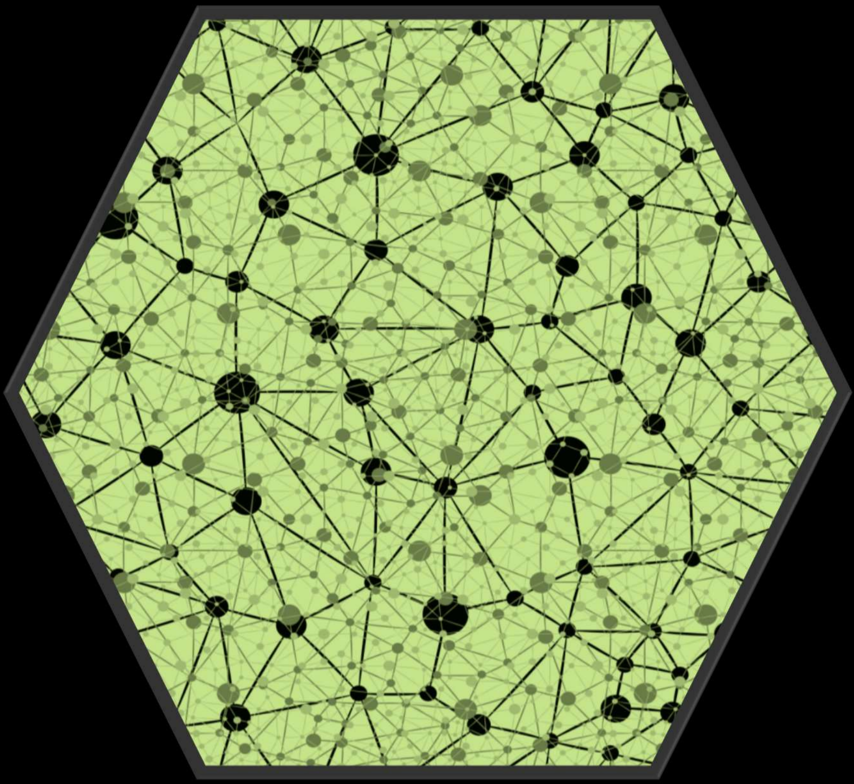
NO hyper plane
separates the two classes.
→ Single-Layer Perceptrons (SLPs)
can only learn **linearly separable** problems.

Encoding Arbitrary Decision Boundaries

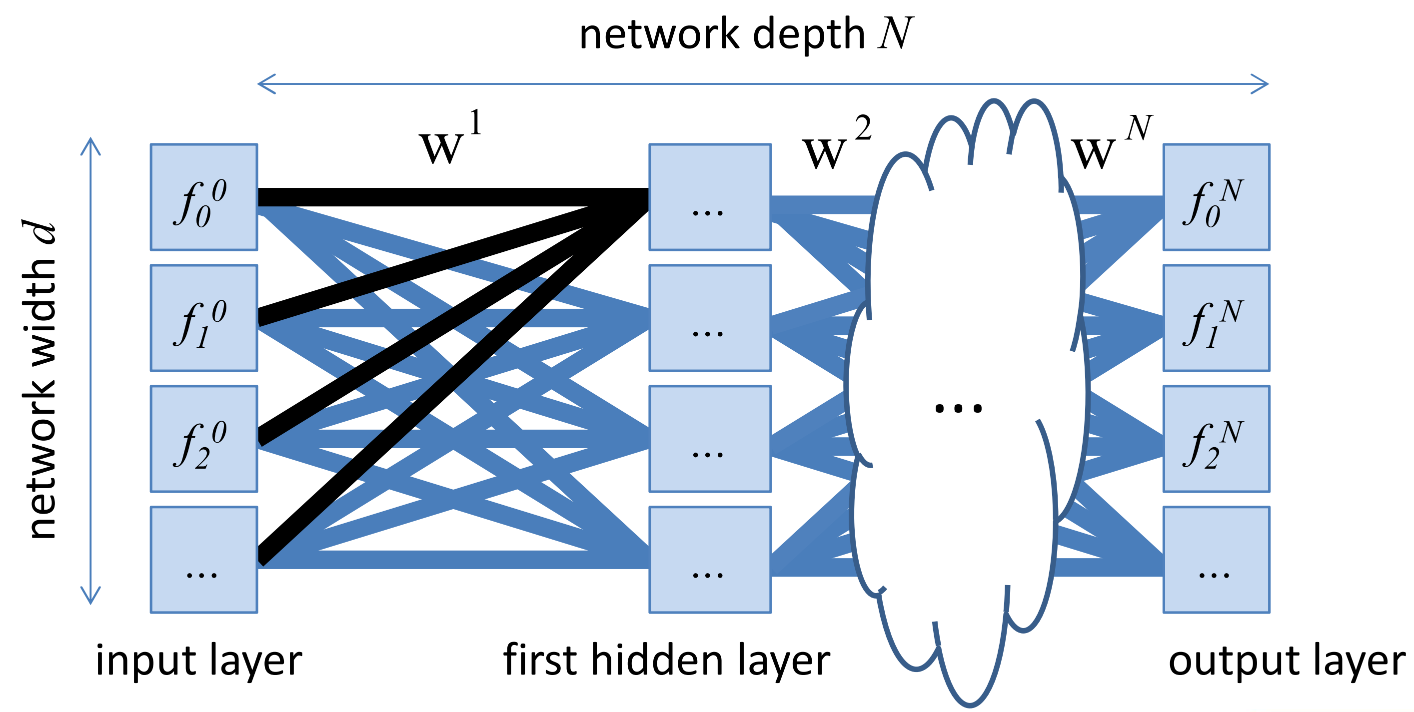
- **Idea:** use of a Multi-Layer Perceptron (MLP) with non-linear activation functions



Multi-Layer Architectures



Structure and Notation for Deep Architectures



$$\underbrace{\mathbf{f}^N}_{\textit{output layer}} = \overbrace{\mathbf{f}}^{\textit{depth}}{}^{N}(\ldots\mathbf{f}^2(\mathbf{f}^1(\underbrace{\mathbf{f}^0}_{\textit{input layer}};\mathbf{W}^1);\mathbf{W}^2)\ldots;\mathbf{W}^N)$$

NOTATION: bold math-script used for tensors of order above 2 (basically 3D arrays or higher)

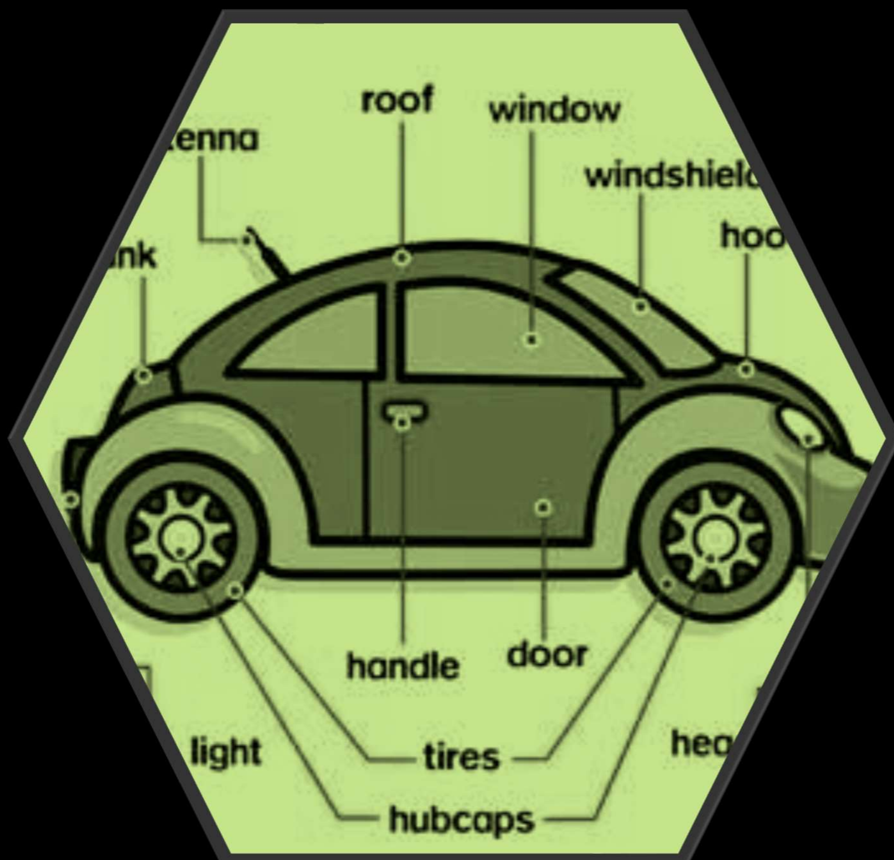
$$\underbrace{\mathbf{W}}_{\textit{all parameters}} = \left[\mathbf{W}^1\mathbf{W}^2\ldots\mathbf{W}^N\right] = \begin{bmatrix} \ldots & \ldots & \ldots \\ \ldots & w_{ij}^{\;l} & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix}$$

NOTATION: superscript usually refers to layer number, subscript to position in layer

weight $w_{ij}^{\;l}$ connects the i^{th} neuron in layer l-1 to the j^{th} neuron in layer l

Outlook:

Learning Representations

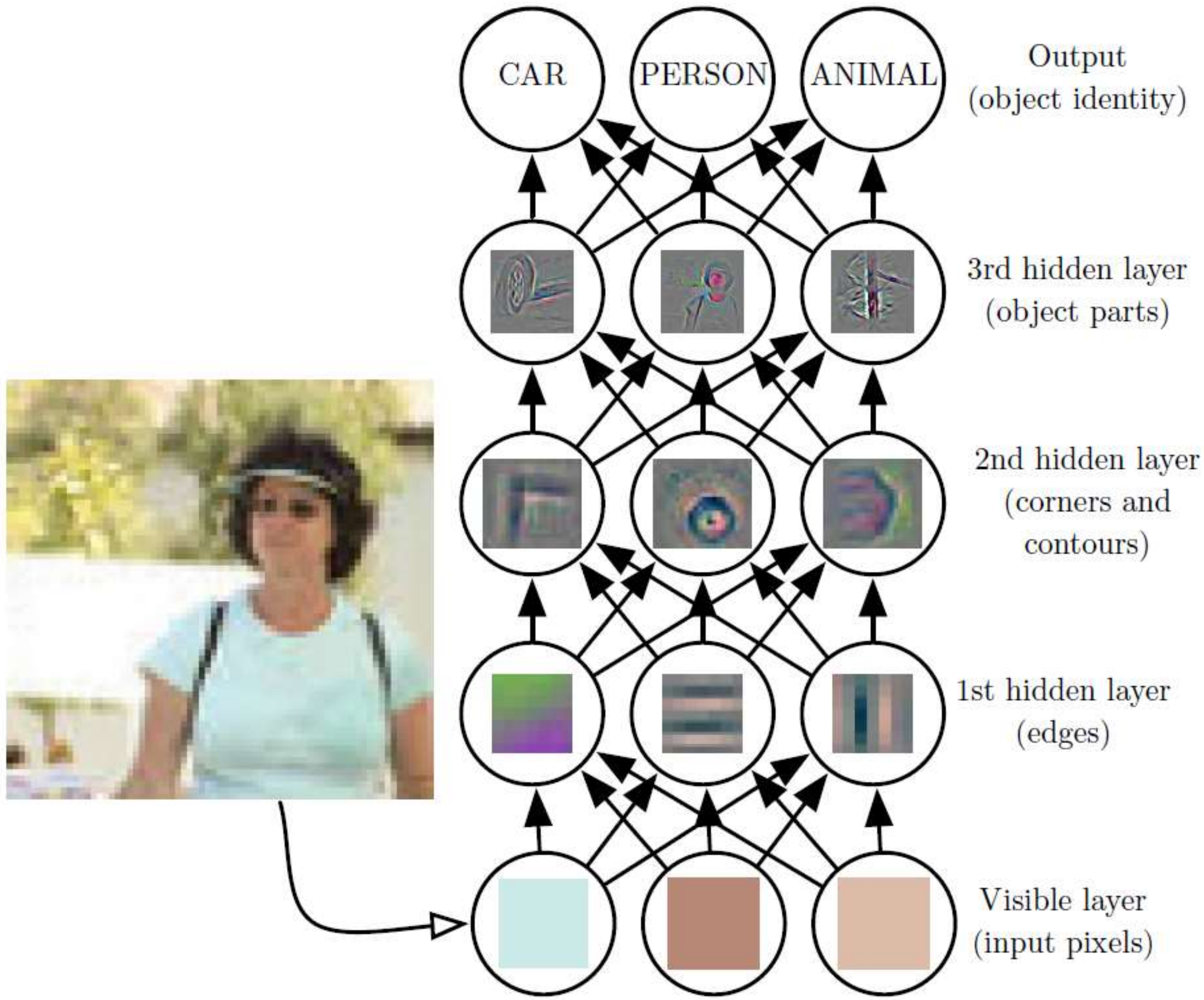


Representational Power of Feedforward Networks

- The basic Perceptron represents a linear classifier.
- Boolean functions can be represented by layered networks with one hidden layer (networks may be very wide requiring an exponential number of hidden neurons compared to input).
- Layered networks with one hidden layer can also represent any continuous function [Cybenko 1989; Hornik et al. 1989].
- Layered networks with two hidden layers can represent any mathematical function [Cybenko 1988].

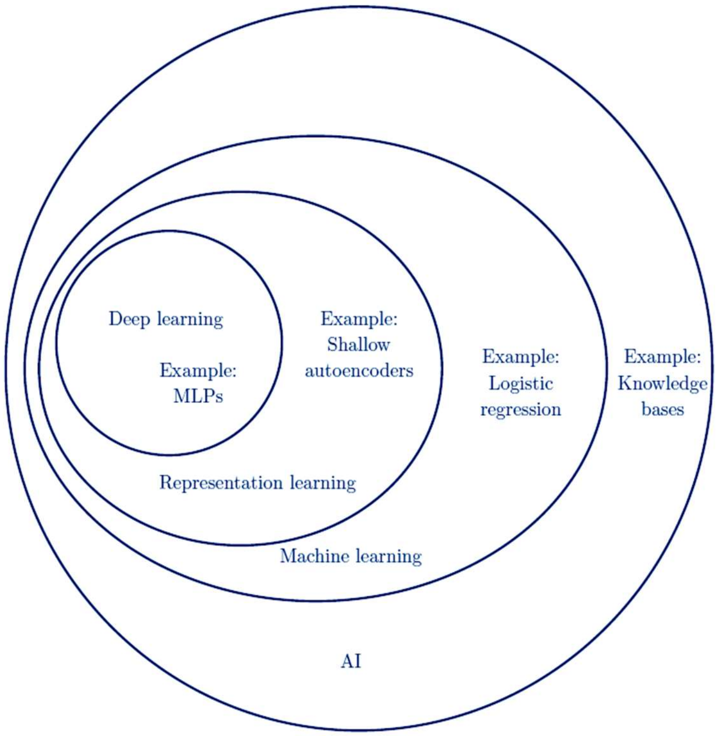
→ long-standing optimism about the potential of neural networks to model learning and intelligent systems
→ question arises: why use more than two hidden layers – why is `deep' advantageous at all? (see Lecture 4)

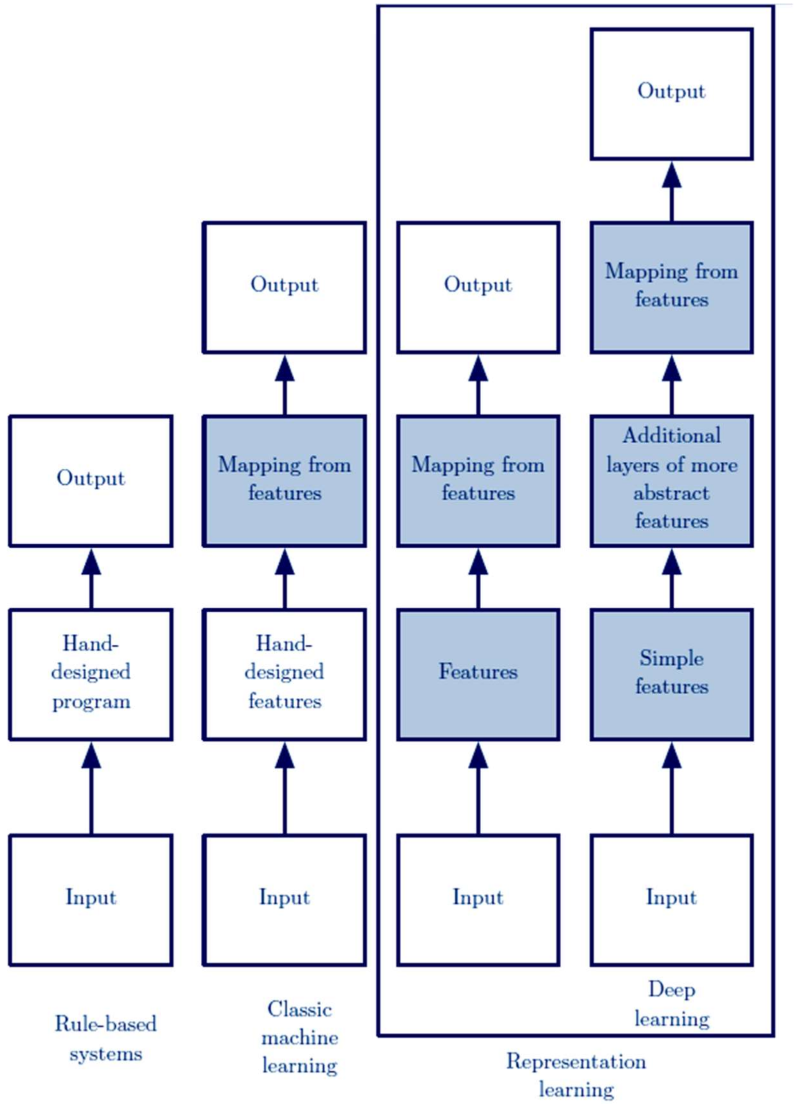
Deep Composition



source: Ian Goodfellow, www.deeplearningbook.org

The Concept of Deep Representation Learning





source: Ian Goodfellow, www.deeplearningbook.org

"It is only after much hesitation that the writer has reconciled himself to the addition of the term "neurodynamics" to the list of such recent linguistic artifacts as "cybernetics", "bionics", "autonomics", "biomimesis", "synnoetics", "intelectronics", and "robotics". It is hoped that by selecting a term which more clearly delimits our realm of interest and indicates its relationship to traditional academic disciplines, the underlying motivation of the perceptron program may be more successfully communicated."

--- *Frank Rosenblatt*

from "Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms", Spartan Books, 1962

Next Time: Towards Training Deep Architectures

- Computational Graphs
- Reverse Auto-Differentiation