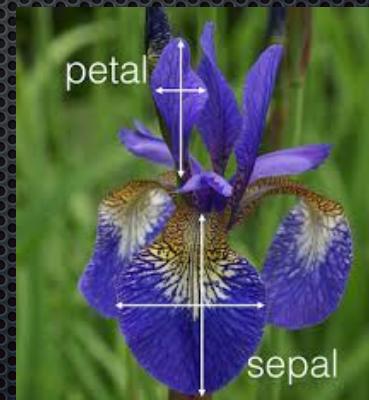


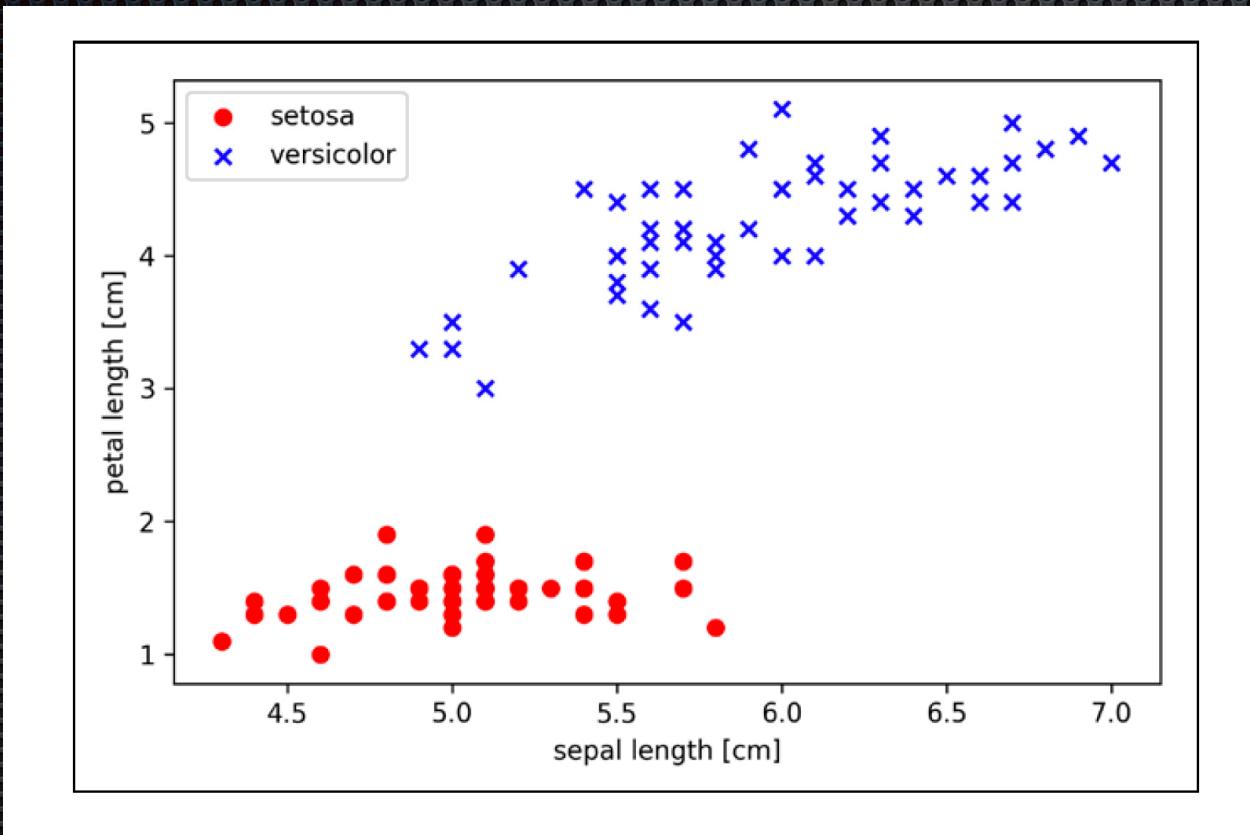
The Iris Classification Problem



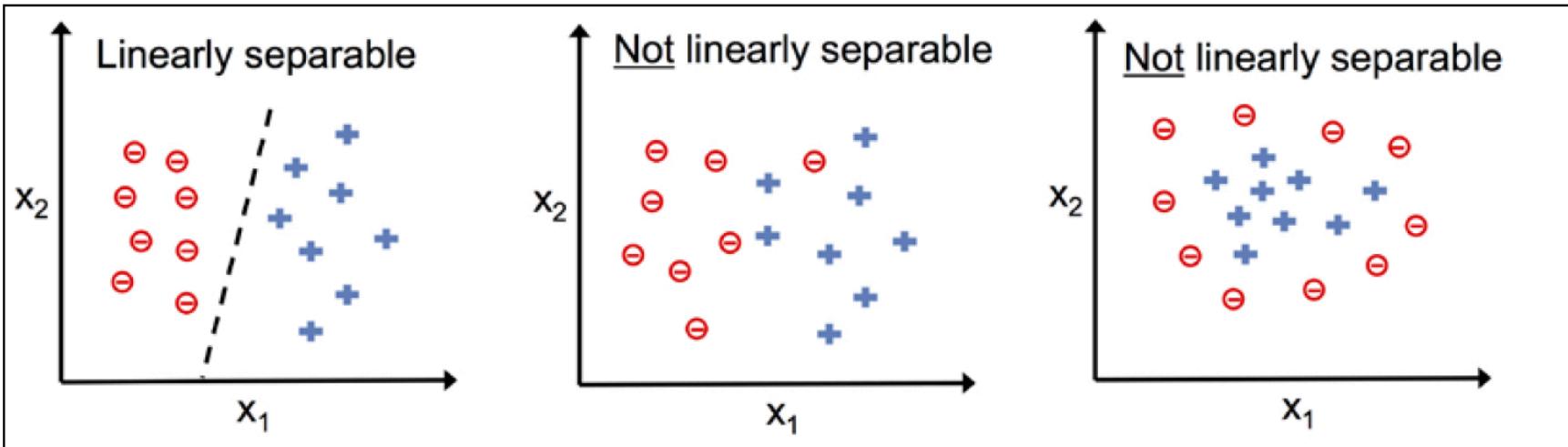
Data: petal width/length
sepal width/length



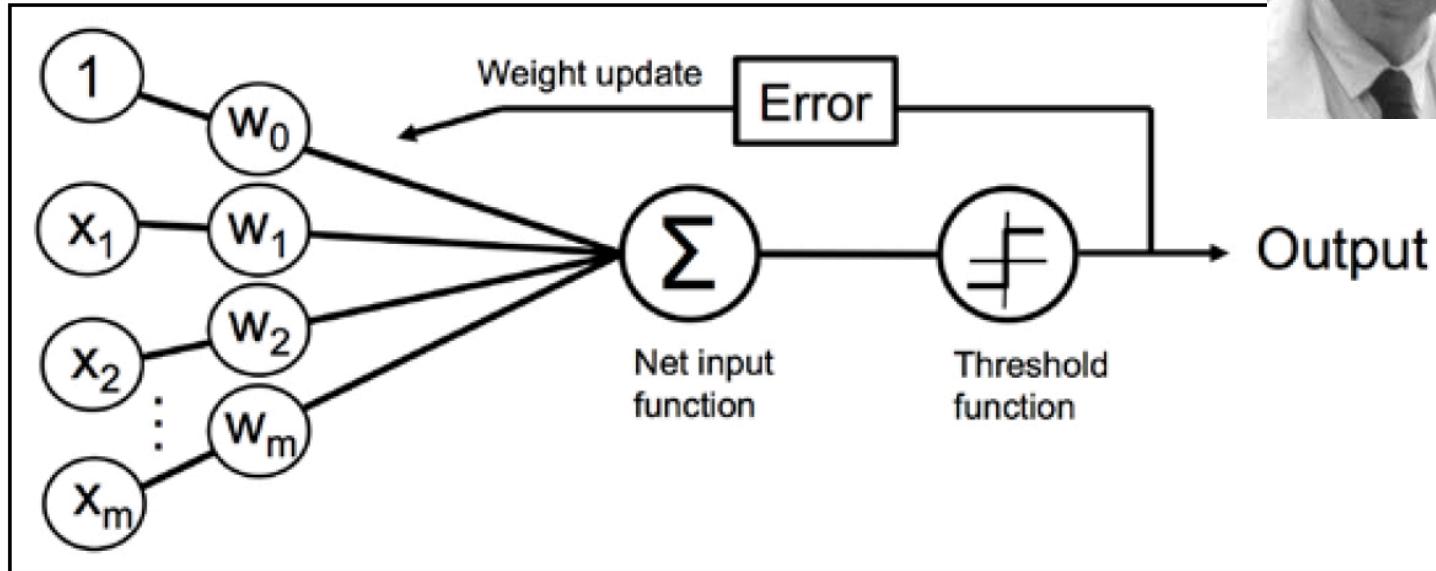
Classification problem



Linearly vs Not linearly separable



Perceptron (Rosenblatt, 1957)



“the first machine which is capable of having an original idea,”

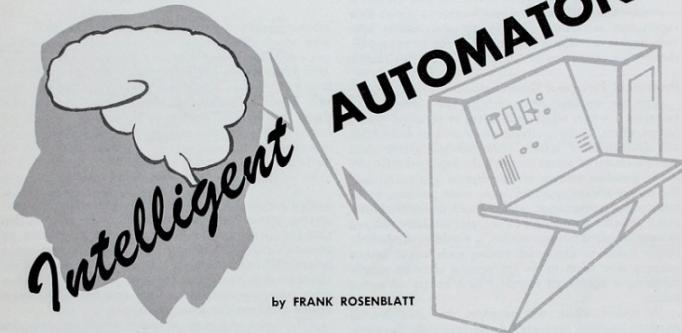
Vol. VI, No. 2, Summer 1958

research trends

CORNELL AERONAUTICAL LABORATORY, INC., BUFFALO 21, NEW YORK



The Design of an



Introducing the perceptron — A machine which senses, recognizes, remembers, and responds like the human mind.

STORIES about the creation of machines having human qualities have long been a fascinating province in the realm of science fiction. Yet we are now about to witness the birth of such a machine — a machine capable of perceiving, recognizing, and identifying its surroundings without any human training or control.

Development of that machine has stemmed from a search for an understanding of the physical mechanisms which underlie human experience and intelligence. The question of the nature of these processes is at least as ancient as any other question in western science and philosophy, and, indeed, ranks as one of the greatest scientific challenges of our time.

Our understanding of this problem has gone perhaps as far as had the development of physics before Newton. We have some excellent descriptions of the phenomena to be explained, a number of interesting hypotheses, and a little detailed knowledge about events in the nervous system. But we lack agreement on any integrated set of principles by which the functioning of the nervous system can be understood.

We believe now that this ancient problem is about to yield to our theoretical investigation for three reasons:

First, in recent years our knowledge of the functioning of individual cells in the central nervous system has vastly increased.

Second, large numbers of engineers and mathematicians are, for the first time, undertaking serious study of the mathematical basis for thinking, perception, and the handling of information by the central nervous system, thus providing the hope that these problems may be within our intellectual grasp.

Third, recent developments in probability theory and in the mathematics of random processes provide new tools for the study of events in the nervous system, where only the gross statistical organization is known and the precise cell-by-cell "wiring diagram" may never be obtained.

Receives Navy Support

In July, 1957, Project PARA (Perceiving and Recognizing Automaton), an internal research program which had been in progress for over a year at Cornell Aeronautical Laboratory, received the support of the Office of Naval Research. The program had been concerned primarily with the application of probability theory to

①

A. Perceptron (Rosenblatt, 1957)

Firing of neuron is binary classification task. Use step function as activation function $\phi(\tilde{z})$, i.e.

$$\phi(\tilde{z}) = \begin{cases} 1 & \tilde{z} \geq 0 \\ -1 & \tilde{z} < 0 \end{cases}$$

where

$$\tilde{z} = \sum_{j=1}^m w_j x_j$$

x_1, \dots, x_m : input data

w_1, \dots, w_m : weights

and suppose there are N data samples ($i=1, \dots, N$)

Define

$$\begin{cases} w_0 = -\theta & \text{(bias)} \\ x_0 = 1 \end{cases}$$

then $z = \sum_{j=0}^m w_j x_j = -\theta + \tilde{z}$

$$\phi(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$$

Perceptron algorithm :

- initialise weights randomly $\epsilon(0, 1)$
- for each training sample (x_1^i, \dots, x_m^i) with true output $y^i \in \{-1, 1\}$
 1. compute : $\hat{y}^i = \phi(z^i)$
 2. update : $\hat{w}_j = w_j + \Delta w_j$

3

$$\Delta w_j = \eta \left(\hat{y}^i - y^i \right) x_j^i \quad j=1, \dots, m$$

η : learning rate

Note : all weights updated simultaneously for each i

Cases :

1. $\hat{y}^i - y^i = 0$ correct prediction of output
 $\rightarrow \Delta w_j = 0$

2. $\hat{y}^i - y^i \neq 0$ incorrect

- a. $\hat{y}^i = 1, y^i = -1$

$$\rightarrow \Delta w_j = -2\eta x_j^i \rightarrow \begin{matrix} \text{more} \\ w \end{matrix}$$

- b. $\hat{y}^i = -1, y^i = 1$

$$\rightarrow \Delta w_j = 2\eta x_j^i \rightarrow \begin{matrix} \text{more} \\ w \end{matrix}$$

26. assume $\eta = 1$ & $x_j^i = \frac{1}{2}$ [4]

(sample misclassified with
 $y_i^i = -1$ instead of 1.)

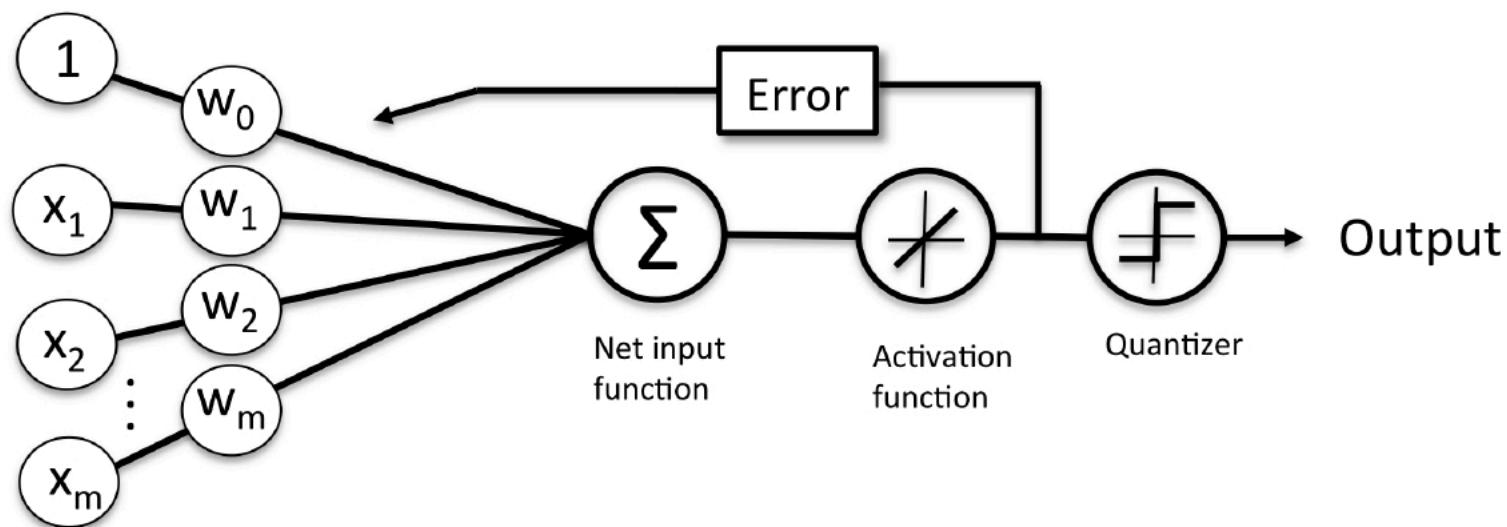
$$\rightarrow \Delta w_j^i = 1$$

pushed such that z is
larger, so that it is more
likely that $z^i \geq 0$ and
 $y_i^i = \phi(z^i) = +1$

Remarks :- the output is also
called a class label

- classes should be
separable by a
linear decision
boundary.

Adaptive Linear Neuron (AdaLiNe)



B. Adaptive linear neuron

In the perceptron, the weight were updated based on the step function. In the Adaline, the weight are updated using the linear activation function

$$\phi(z) = z$$

where $z = \sum_{j=0}^m w_j x_j$

After this, a quantizer (e.g. a step function) can be used to predict the class labels,

i.e. $q(z) = \begin{cases} 1 & z \geq 0 \\ -1 & z < 0 \end{cases}$

6

where again the bias is taken into account through w_0 and x_0 .

To determine the weights, we define a cost function

$$\mathcal{J}(\underline{w}) = \frac{1}{z} \sum_i (y^i - \phi(z^i))^2$$

where the sum is over the samples.

The mathematical problem to solve is to determine \underline{w}_* as

$$\underline{w}_* = \min_{\underline{w}} \mathcal{J}(\underline{w})$$

Mathematical problem

Input

$$\mathbf{x} = (x_0, \dots, x_m)^T$$

Weights

$$\mathbf{w} = (w_0, \dots, w_m)^T$$

data samples: $i = 1, \dots, N$

Activation:

$$\mathbf{z} = \mathbf{w}^T \mathbf{x}$$

$$\phi(\mathbf{z}) = \mathbf{z}$$

Cost function:

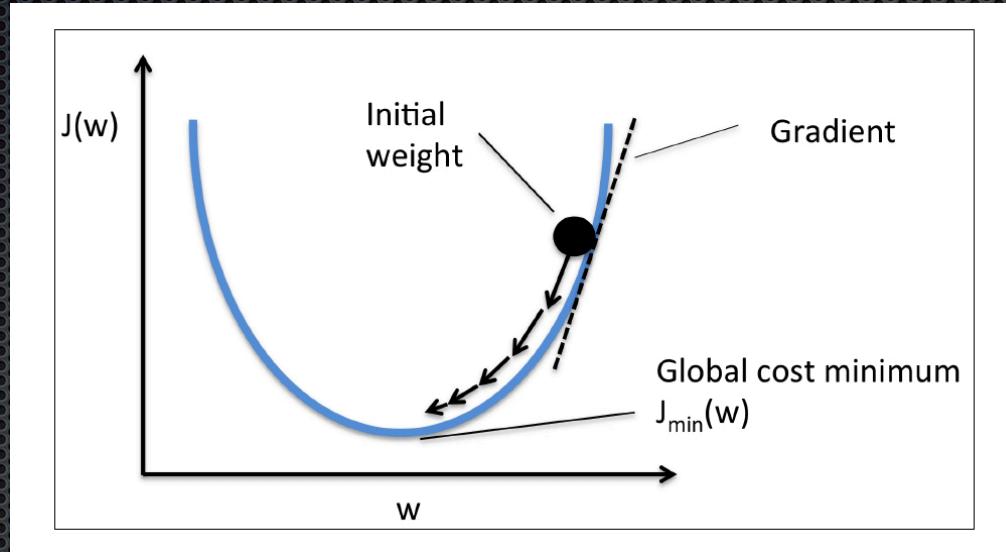
$$J(\mathbf{w}) = \frac{1}{2} \sum_i (y^i - \phi(z^i))^2$$

Task:

$$\min_{\mathbf{w}} J(\mathbf{w})$$

Method: Gradient Descent

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \Delta \mathbf{w}_{k+1}$$



$$\Delta \mathbf{w}_{k+1} = -\eta \nabla J(\mathbf{w}_k)$$

7

Method 1. Gradient descend

iteration index k , $\underline{w}_0 = \underline{w}_{\text{initial}}$

then

$$\left| \begin{array}{l} \underline{w}_{k+1} = \underline{w}_k + \Delta \underline{w}_{k+1} \\ \Delta \underline{w}_{k+1} = -\eta \nabla \mathcal{E}(\underline{w}_k) \end{array} \right. \begin{array}{l} \text{learning rate} \\ \text{down gradient} \end{array}$$

Now $(\nabla \mathcal{E})_j$ can be computed from

$$\begin{aligned} (\nabla \mathcal{E})_j &= \frac{\partial}{\partial w_j} \left(\frac{1}{2} \sum_i \left(y^i - \phi(z^i) \right)^2 \right) \\ &= \sum_i (y^i - \phi(z^i)) \frac{\partial}{\partial w_j} (y^i - \phi(z^i)) \\ \text{use } \phi(z^i) &= \sum_{j=0}^m x_j^i w_j \\ &= \sum_i (y^i - \phi(z^i)) (-x_j^i) \end{aligned}$$

Hence :

$$(\Delta w_{t+1})_j = \eta \equiv (y^i - q(z^i)) x_j^i$$

Remarks : - convergence can be slow when small values of η are needed. In this case, it often helps to rescale the input

data (with $x_j^i = (x_j^1, \dots, x_j^N)$)

$$\tilde{x}_j^i = \frac{x_j^i - \mu_j}{\sigma_j}$$

where $\mu_j^i = \frac{1}{N} \sum_{i=1}^N x_j^i$ (mean)

and $\sigma_j^2 = \frac{1}{N} \sum_{i=1}^N (x_j^i - \mu_j)^2$

9

Method 2 : Stochastic gradient
descend

Update of weights :

$$(\Delta w_{k+1})_j = \eta (y^i - \phi(z^i)) x_j^i$$

Advantages : - cheaper
- more easily to
hop over 'local'
minima of J .

Variants : - use varying η
- shuffle training
data for every
iteration (epoch)

Exercise B2

https://github.com/hadijkstra/CriticalEarth_April2022

