

Neural Network

valentina.corino@polimi.it

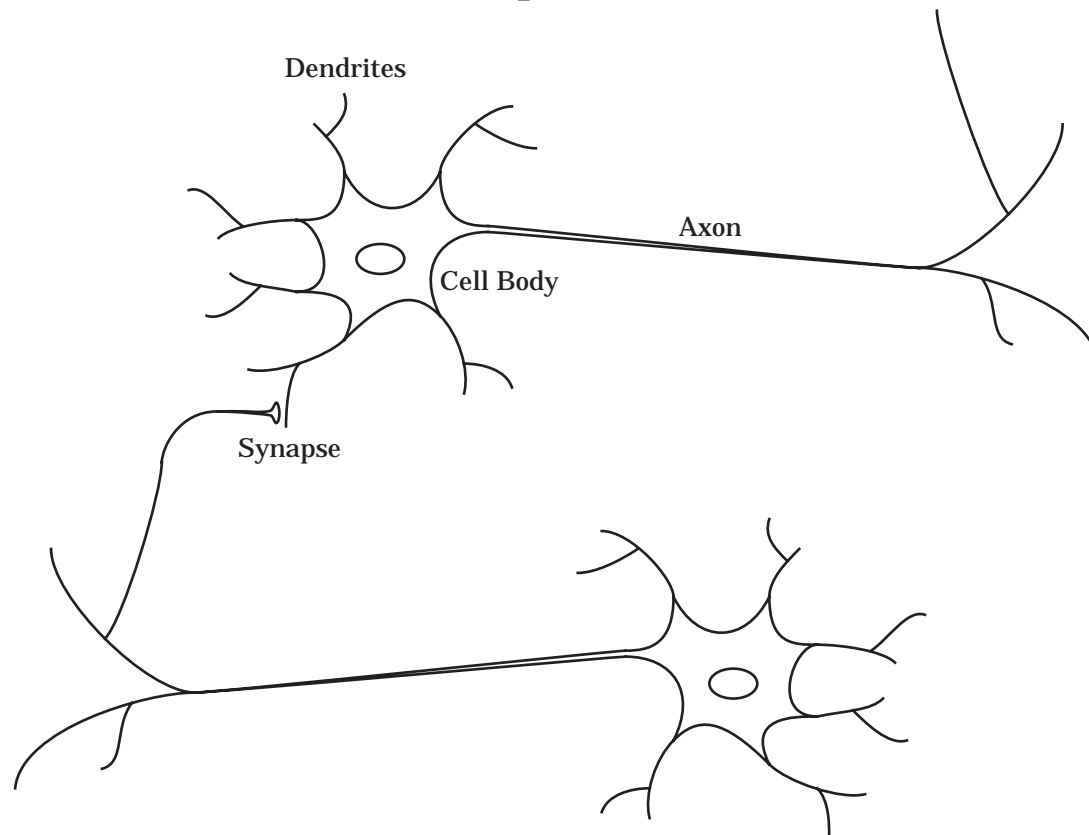
- **Aerospace**
 - High performance aircraft autopilots, flight path simulations, aircraft control systems, autopilot enhancements, aircraft component simulations, aircraft component fault detectors
- **Automotive**
 - Automobile automatic guidance systems, warranty activity analyzers
- **Banking**
 - Check and other document readers, credit application evaluators
- **Defense**
 - Weapon steering, target tracking, object discrimination, facial recognition, new kinds of sensors, sonar, radar and image signal processing including data compression, feature extraction and noise suppression, signal/image identification
- **Electronics**
 - Code sequence prediction, integrated circuit chip layout, process control, chip failure analysis, machine vision, voice synthesis, nonlinear modeling

- Financial
 - Real estate appraisal, loan advisor, mortgage screening, corporate bond rating, credit line use analysis, portfolio trading program, corporate financial analysis, currency price prediction
- Manufacturing
 - Manufacturing process control, product design and analysis, process and machine diagnosis, real-time particle identification, visual quality inspection systems, beer testing, welding quality analysis, paper quality prediction, computer chip quality analysis, analysis of grinding operations, chemical product design analysis, machine maintenance analysis, project bidding, planning and management, dynamic modeling of chemical process systems
- Medical
 - Breast cancer cell analysis, EEG and ECG analysis, prosthesis design, optimization of transplant times, hospital expense reduction, hospital quality improvement, emergency room test advisement

- **Robotics**
 - Trajectory control, forklift robot, manipulator controllers, vision systems
- **Speech**
 - Speech recognition, speech compression, vowel classification, text to speech synthesis
- **Securities**
 - Market analysis, automatic bond rating, stock trading advisory systems
- **Telecommunications**
 - Image and data compression, automated information services, real-time translation of spoken language, customer payment processing systems
- **Transportation**
 - Truck brake diagnosis systems, vehicle scheduling, routing systems

Biology

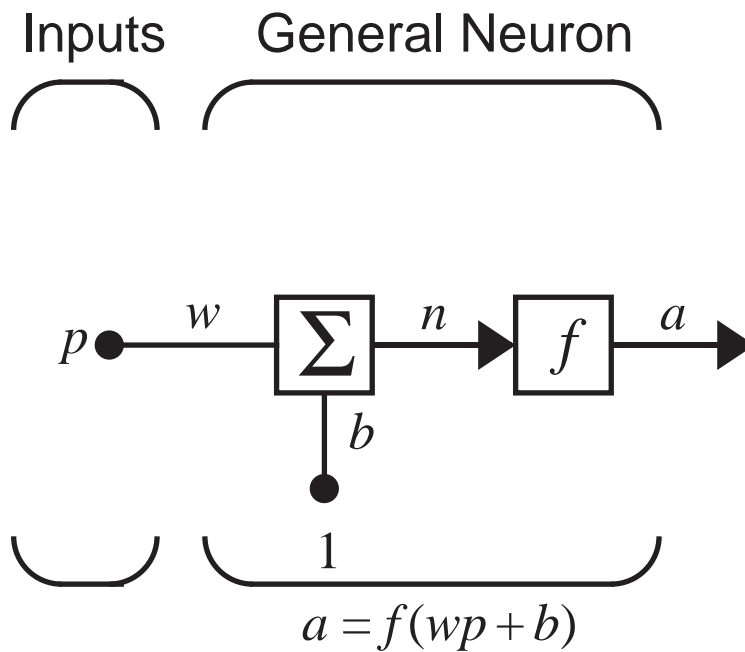
- Neurons respond slowly
 - 10^{-3} s compared to 10^{-9} s for electrical circuits
- The brain uses massively parallel computation
 - $\approx 10^{11}$ neurons in the brain
 - $\approx 10^4$ connections per neuron

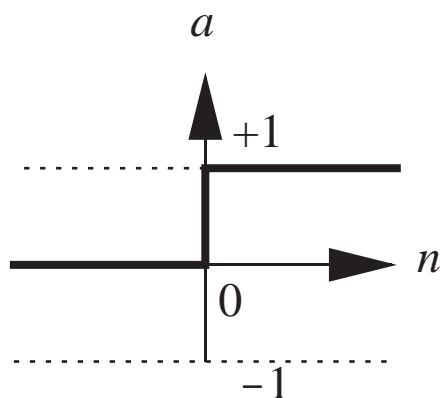




Neuron Model and Network Architectures

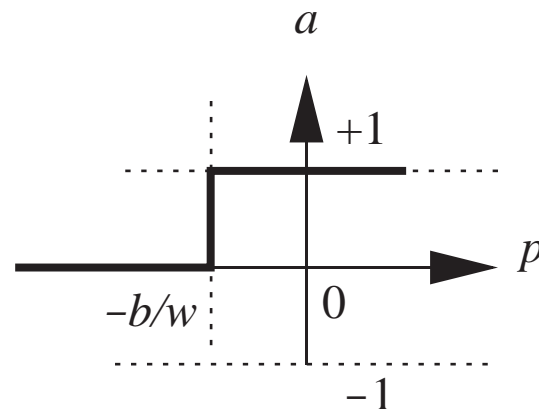
Single-Input Neuron





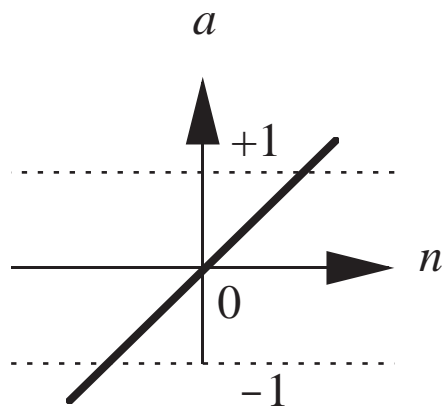
$$a = \text{hardlim}(n)$$

Hard Limit Transfer Function



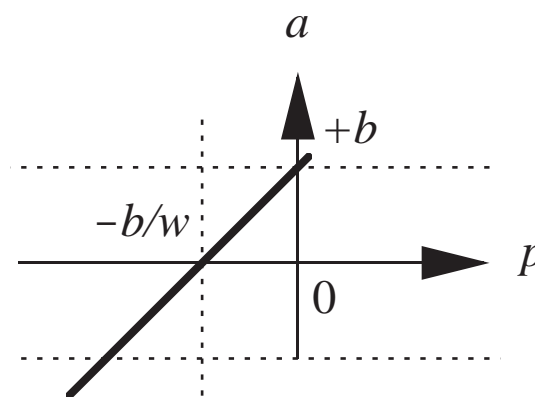
$$a = \text{hardlim}(wp + b)$$

Single-Input *hardlim* Neuron



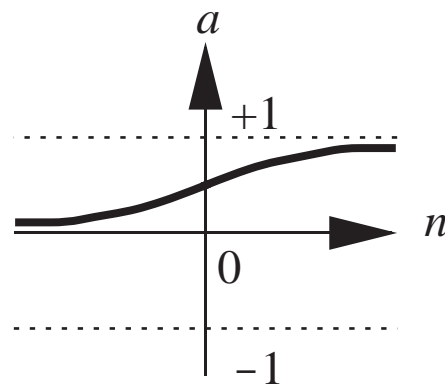
$$a = \text{purelin}(n)$$

Linear Transfer Function



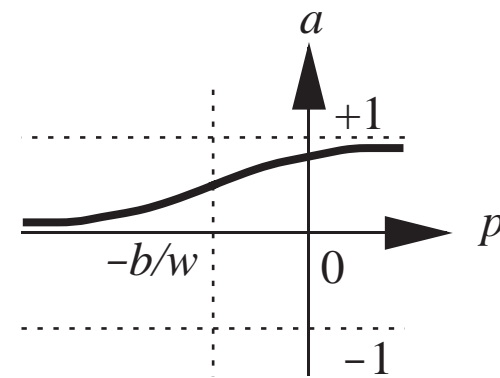
$$a = \text{purelin}(wp + b)$$

Single-Input *purelin* Neuron



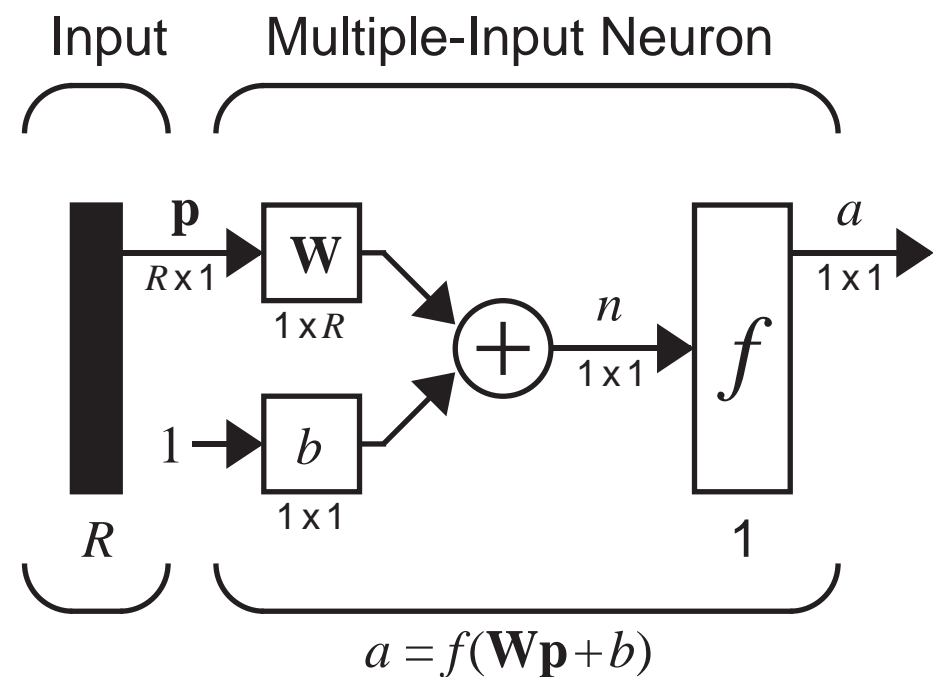
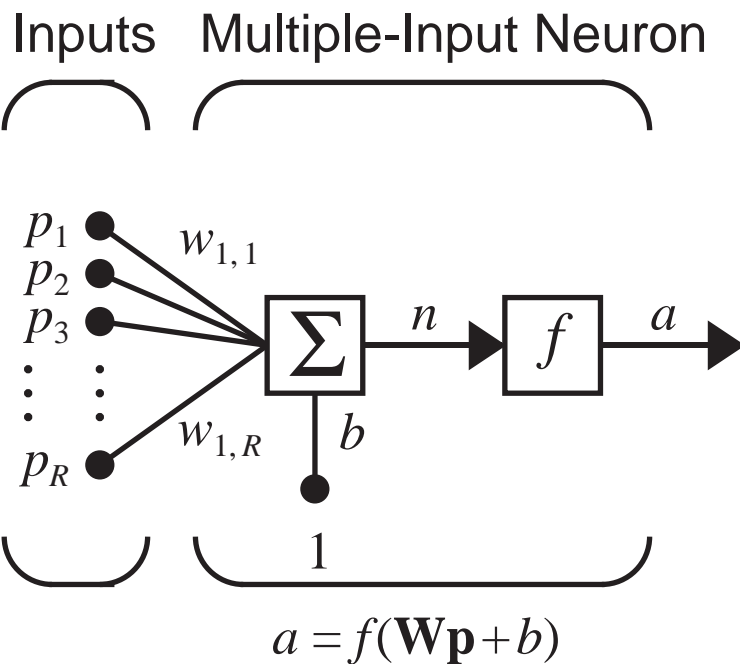
$$a = \text{logsig}(n)$$

Log-Sigmoid Transfer Function

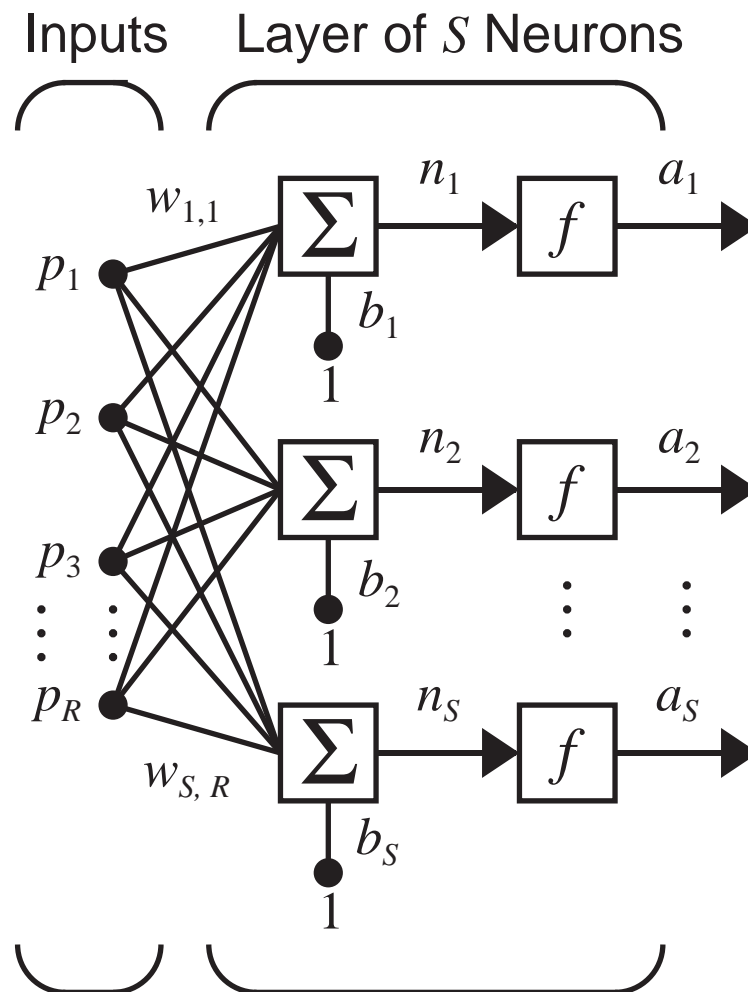


$$a = \text{logsig}(wp + b)$$

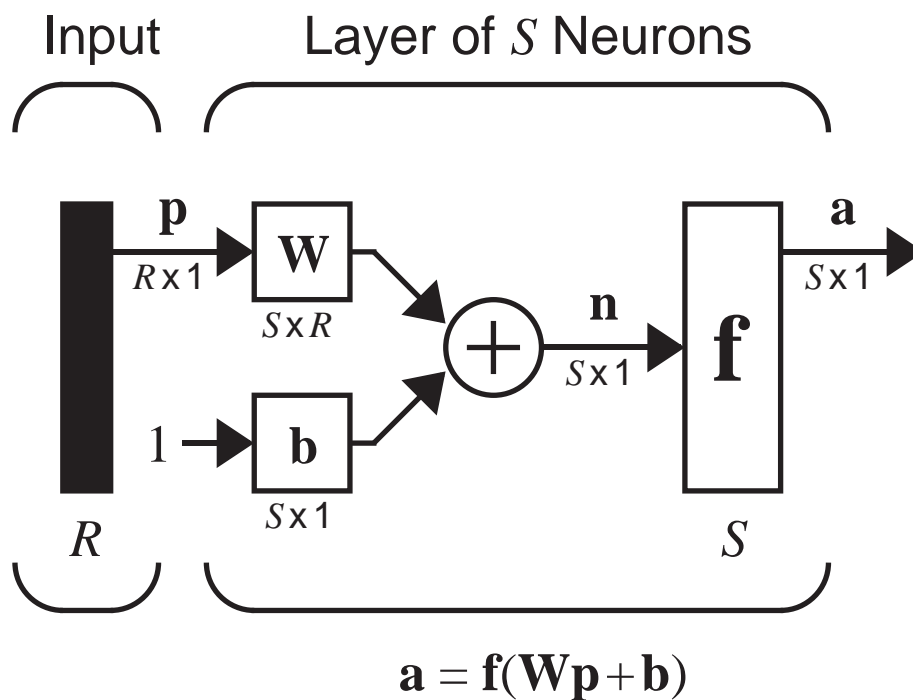
Single-Input *logsig* Neuron



Abbreviated Notation



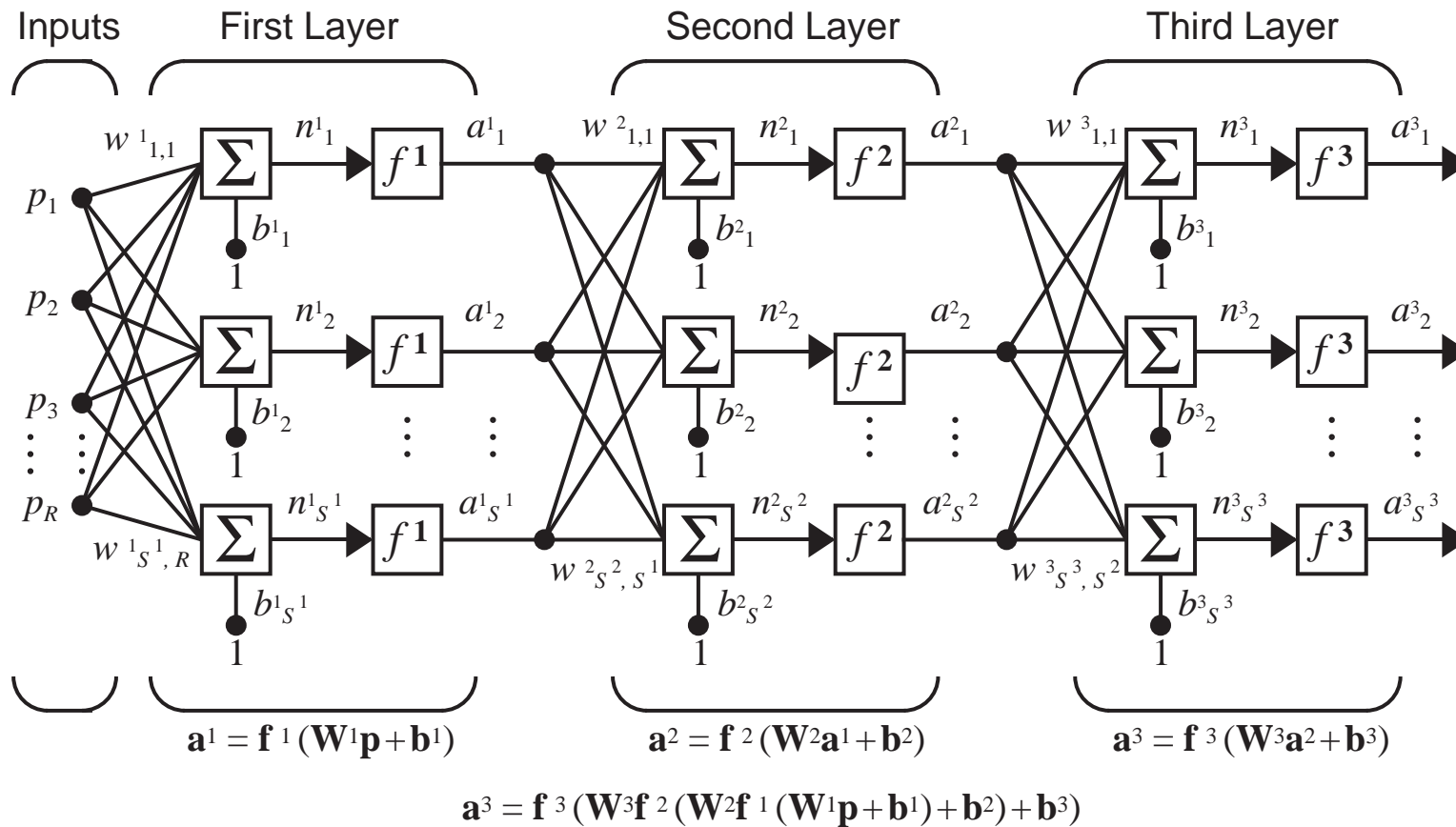
$$\mathbf{a} = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b})$$

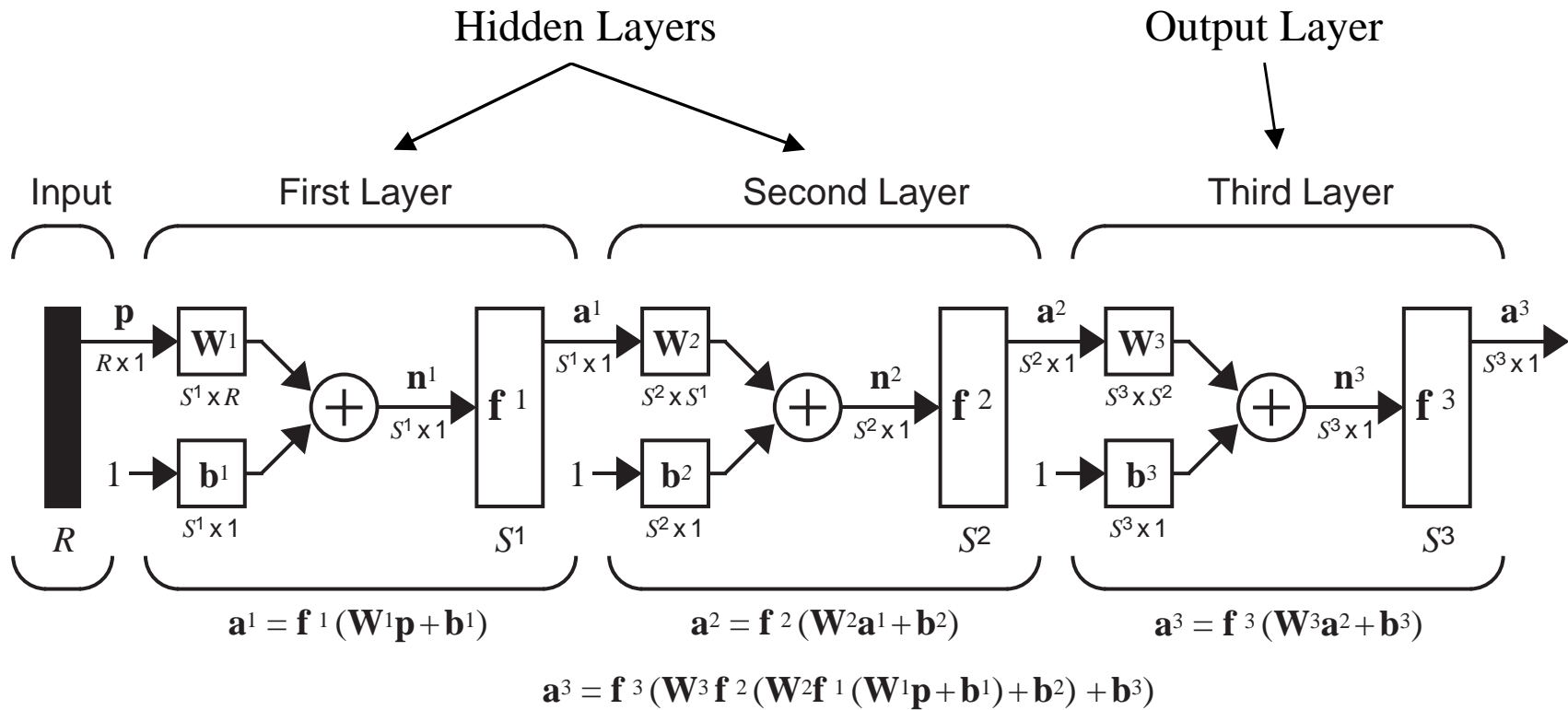


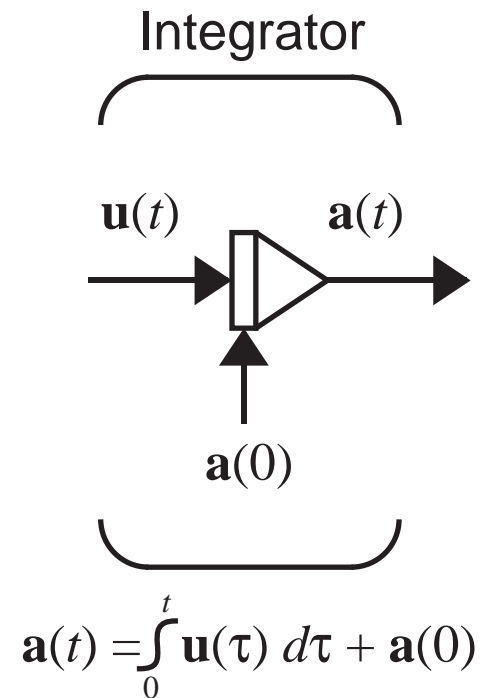
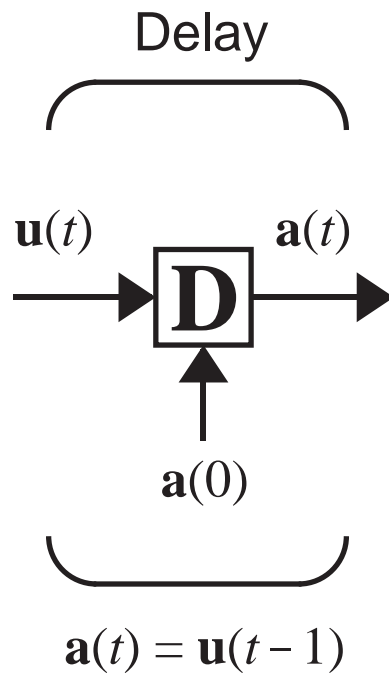
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

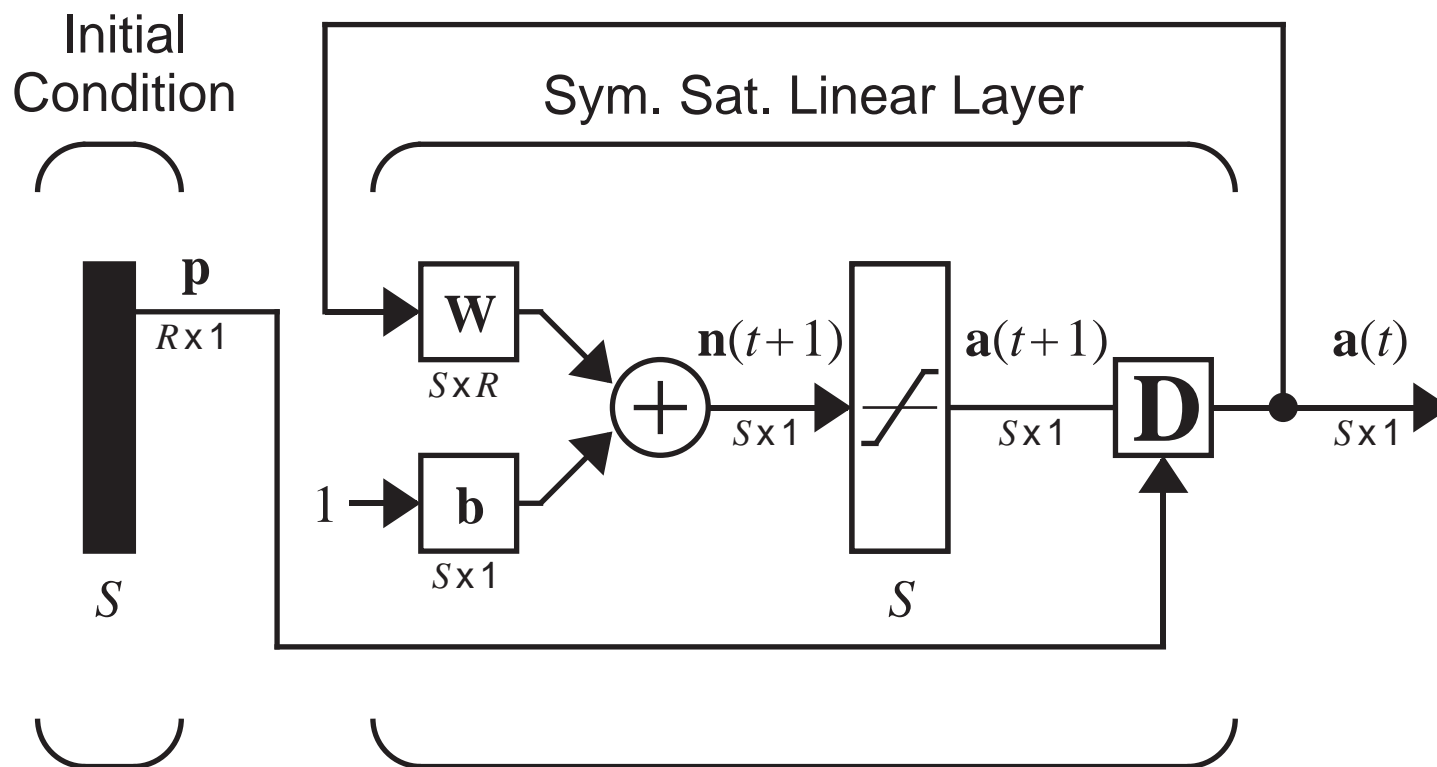
$$\mathbf{p} = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_R \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_S \end{bmatrix} \quad \mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_S \end{bmatrix}$$

Multilayer Network









$$\mathbf{a}(0) = \mathbf{p} \quad \mathbf{a}(t+1) = \text{satlin}(\mathbf{W}\mathbf{a}(t) + \mathbf{b})$$

$$\mathbf{a}(1) = \text{satlins}(\mathbf{W}\mathbf{a}(0) + \mathbf{b}) = \text{satlins}(\mathbf{W}\mathbf{p} + \mathbf{b})$$

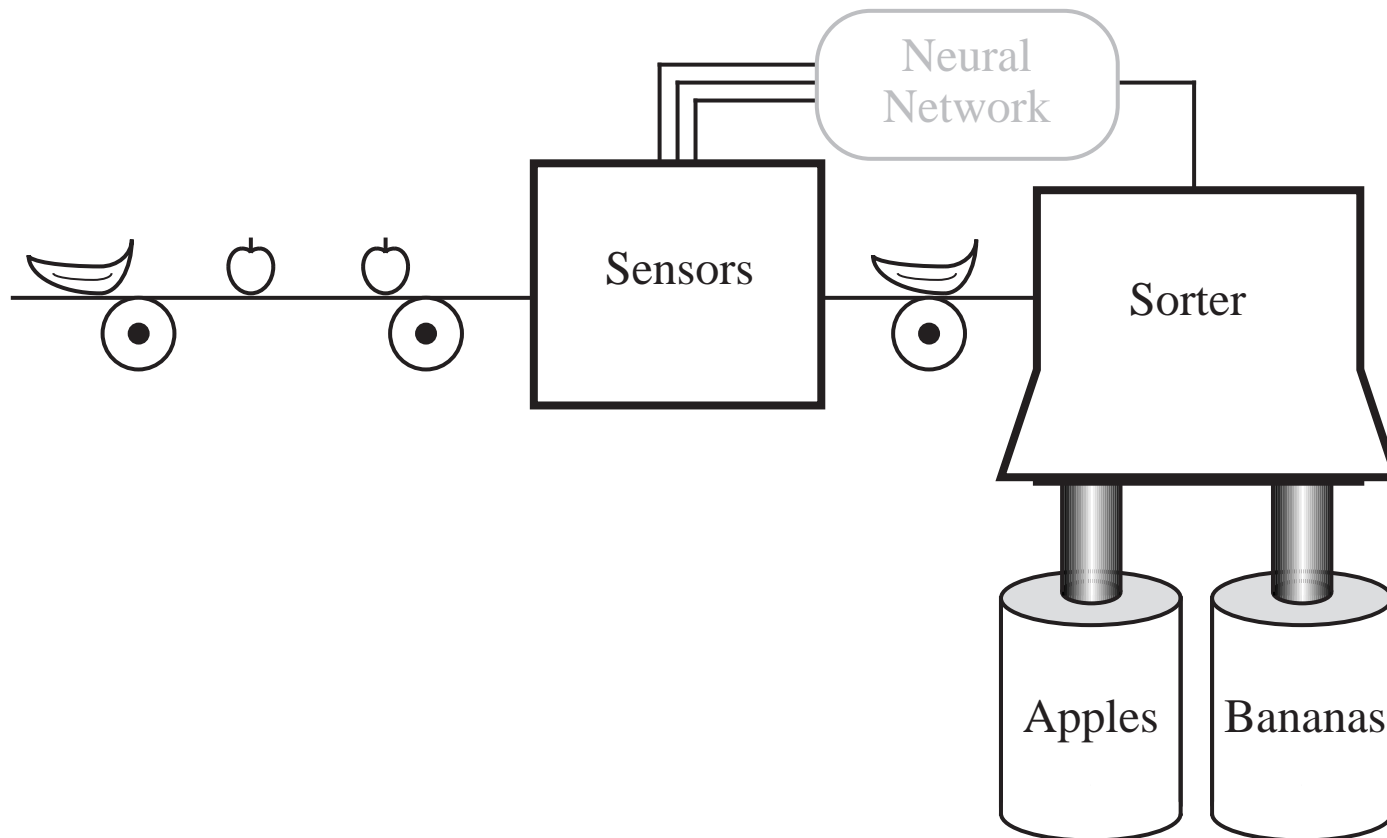
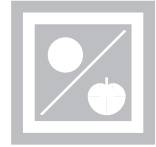
$$\mathbf{a}(2) = \text{satlins}(\mathbf{W}\mathbf{a}(1) + \mathbf{b})$$

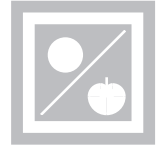


An Illustrative Example

3

Apple/Banana Sorter





Measurement Vector

$$\mathbf{p} = \begin{bmatrix} \text{shape} \\ \text{texture} \\ \text{weight} \end{bmatrix}$$

Shape: { 1 : round ; -1 : elliptical }

Texture: { 1 : smooth ; -1 : rough }

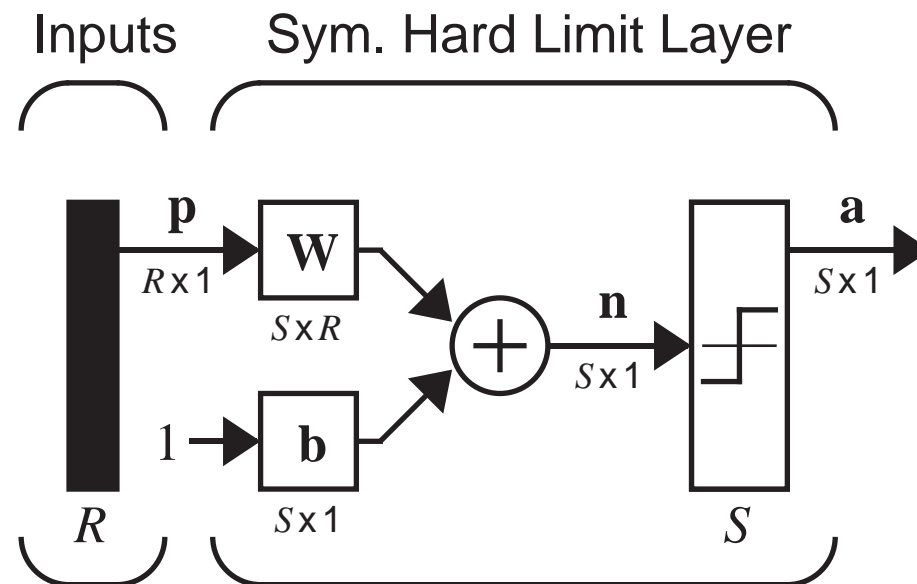
Weight: { 1 : > 1 lb. ; -1 : < 1 lb. }

Prototype Banana

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

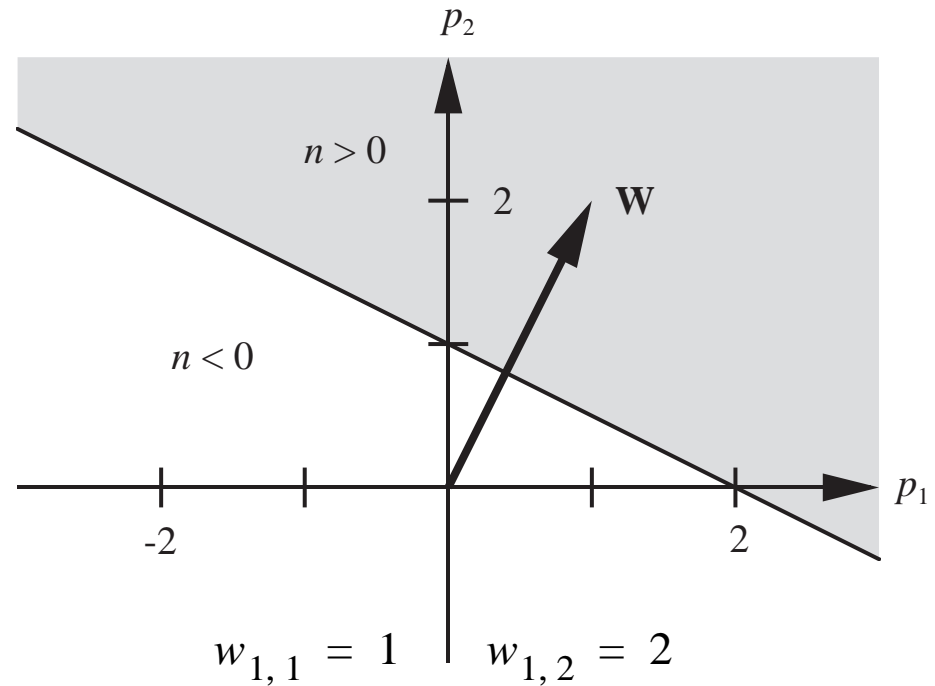
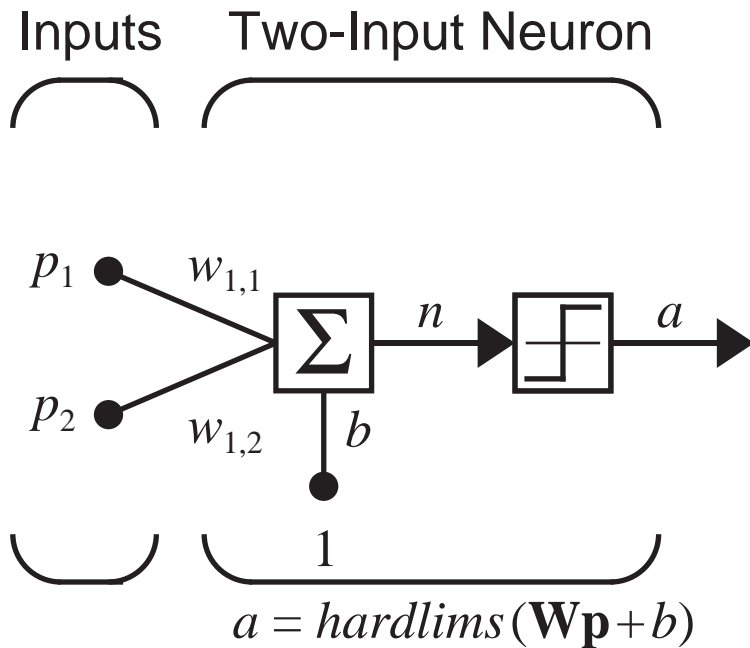
Prototype Apple

$$\mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$



$$\mathbf{a} = \text{hardlims}(\mathbf{W}\mathbf{p} + \mathbf{b})$$

Two-Input Case



$$a = \text{hardlims}(n) = \text{hardlims}\left(\begin{bmatrix} 1 & 2 \end{bmatrix} \mathbf{p} + (-2)\right)$$

Decision Boundary

$$\mathbf{W}\mathbf{p} + b = 0 \quad \begin{bmatrix} 1 & 2 \end{bmatrix} \mathbf{p} + (-2) = 0$$

3

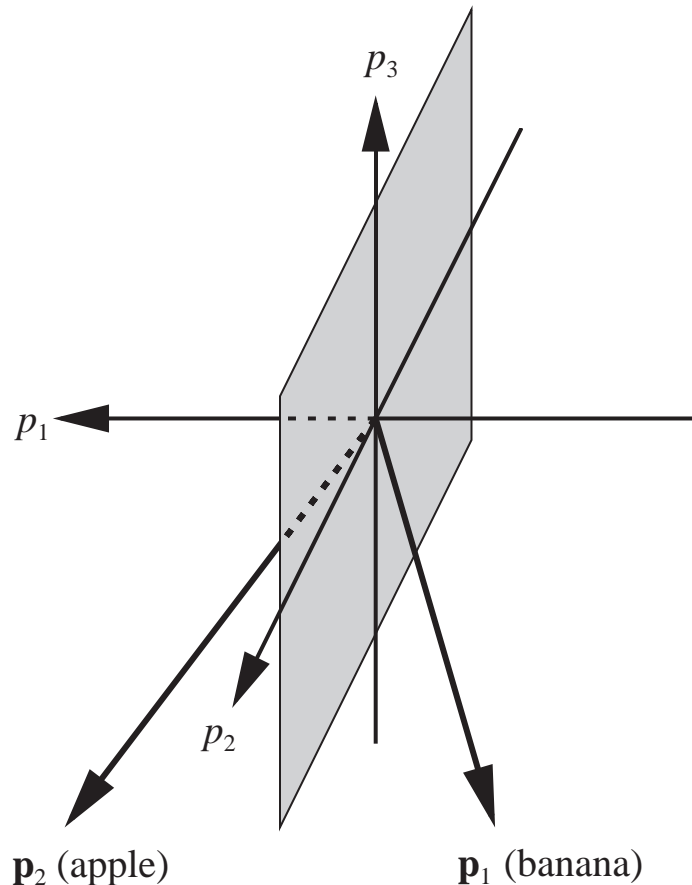
Apple/Banana Example



$$a = \text{hardlims} \left(\begin{bmatrix} w_{1,1} & w_{1,2} & w_{1,3} \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + b \right)$$

The decision boundary should separate the prototype vectors.

$$p_1 = 0$$



The weight vector should be orthogonal to the decision boundary, and should point in the direction of the vector which should produce an output of 1. The bias determines the position of the boundary

$$\begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ p_3 \end{bmatrix} + 0 = 0$$



Banana:

$$a = \text{hardlims} \left(\begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0 \right) = 1(\text{banana})$$

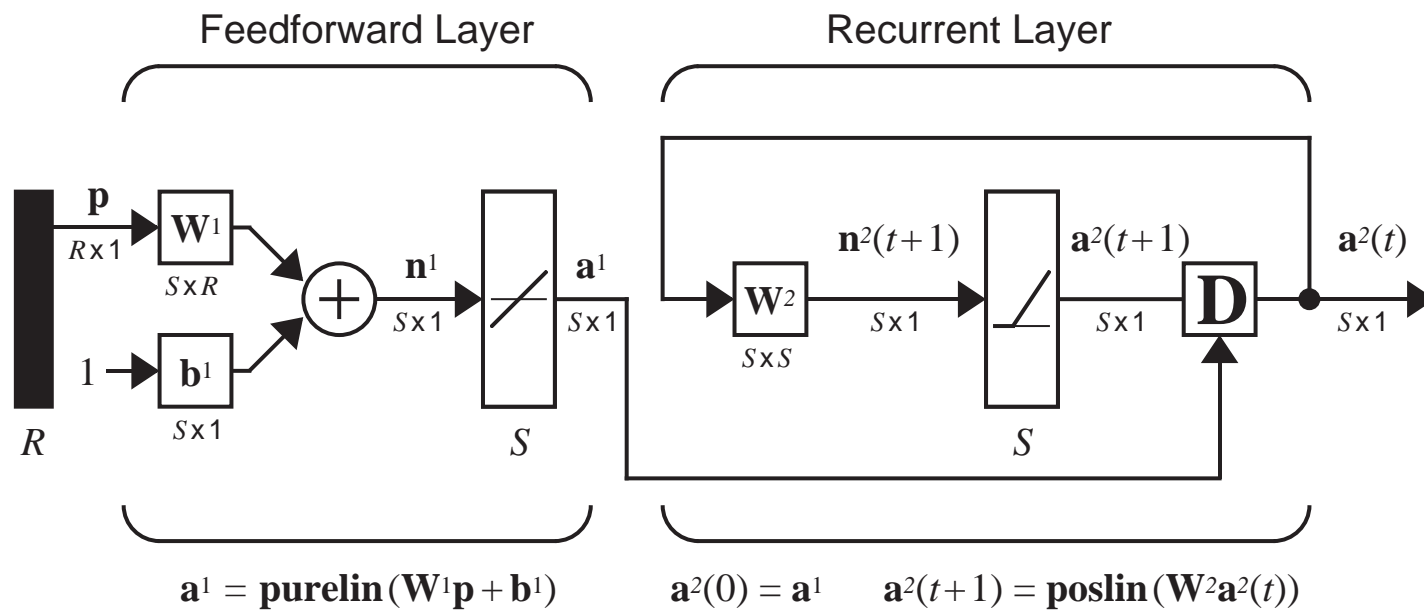
Apple:

$$a = \text{hardlims} \left(\begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0 \right) = -1 \text{ (apple)}$$

“Rough” Banana:

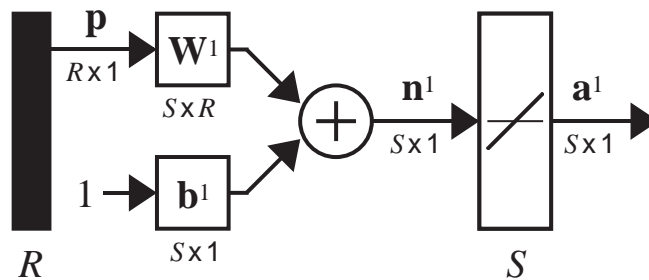
$$a = \text{hardlims} \left(\begin{bmatrix} -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + 0 \right) = 1(\text{banana})$$

Hamming Network





Feedforward Layer



$$\mathbf{a}^1 = \text{purelin}(\mathbf{W}^1 \mathbf{p} + \mathbf{b}^1)$$

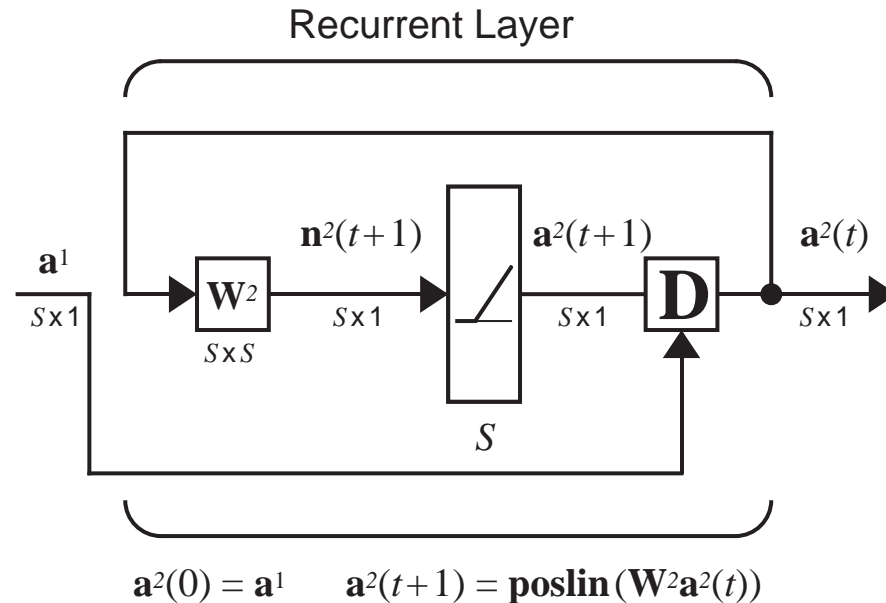
For Banana/Apple Recognition

$$S = 2$$

$$\mathbf{W}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} = \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \end{bmatrix}$$

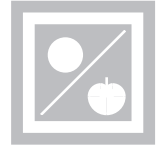
$$\mathbf{b}^1 = \begin{bmatrix} R \\ R \end{bmatrix} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{a}^1 = \mathbf{W}^1 \mathbf{p} + \mathbf{b}^1 = \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \end{bmatrix} \mathbf{p} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1^T \mathbf{p} + 3 \\ \mathbf{p}_2^T \mathbf{p} + 3 \end{bmatrix}$$



$$\mathbf{W}^2 = \begin{bmatrix} 1 & -\epsilon \\ -\epsilon & 1 \end{bmatrix} \quad \epsilon < \frac{1}{S-1}$$

$$\mathbf{a}^2(t+1) = \text{poslin} \left(\begin{bmatrix} 1 & -\epsilon \\ -\epsilon & 1 \end{bmatrix} \mathbf{a}^2(t) \right) = \text{poslin} \left(\begin{bmatrix} a_1^2(t) - \epsilon a_2^2(t) \\ a_2^2(t) - \epsilon a_1^2(t) \end{bmatrix} \right)$$

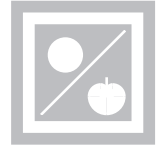


First Layer

Input (Rough Banana)

$$\mathbf{p} = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

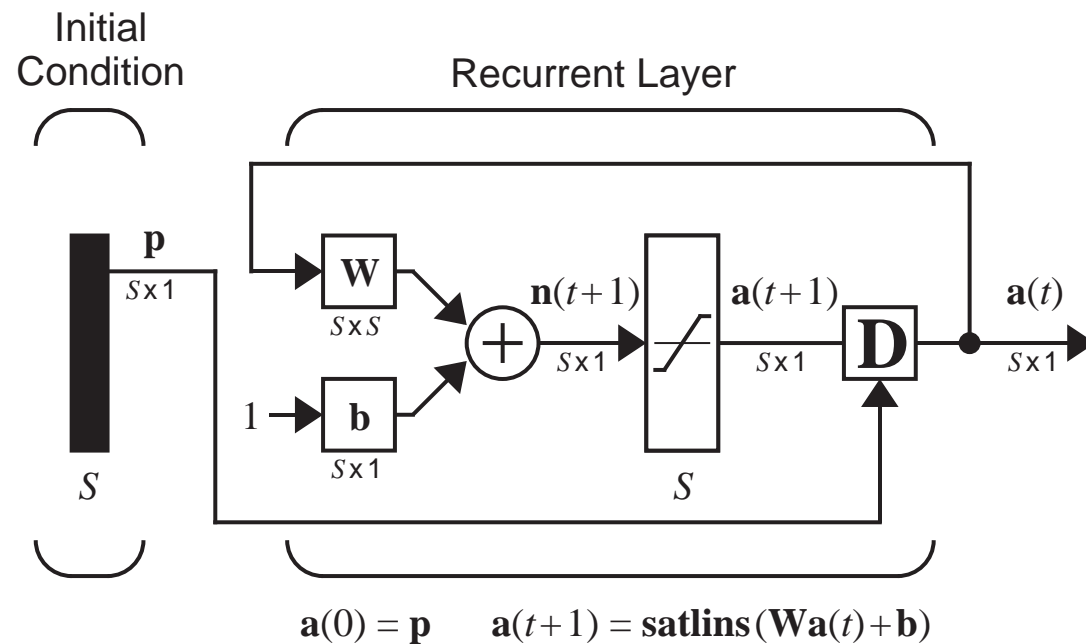
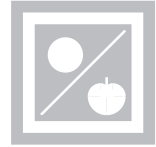
$$\mathbf{a}^1 = \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix} + \begin{bmatrix} 3 \\ 3 \end{bmatrix} = \begin{bmatrix} (1 + 3) \\ (-1 + 3) \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$



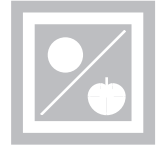
Second Layer

$$\mathbf{a}^2(1) = \mathbf{poslin}(\mathbf{W}^2 \mathbf{a}^2(0)) = \begin{cases} \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix}\right) \\ \mathbf{poslin}\left(\begin{bmatrix} 3 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \end{cases}$$

$$\mathbf{a}^2(2) = \mathbf{poslin}(\mathbf{W}^2 \mathbf{a}^2(1)) = \begin{cases} \mathbf{poslin}\left(\begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 0 \end{bmatrix}\right) \\ \mathbf{poslin}\left(\begin{bmatrix} 3 \\ -1.5 \end{bmatrix}\right) = \begin{bmatrix} 3 \\ 0 \end{bmatrix} \end{cases}$$



Apple/Banana Problem



$$\mathbf{W} = \begin{bmatrix} 1.2 & 0 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 0 \\ 0.9 \\ -0.9 \end{bmatrix}$$

$$a_1(t+1) = \text{satlins}(1.2a_1(t))$$

$$a_2(t+1) = \text{satlins}(0.2a_2(t) + 0.9)$$

$$a_3(t+1) = \text{satlins}(0.2a_3(t) - 0.9)$$

Test: “Rough” Banana

$$\mathbf{a}(0) = \begin{bmatrix} -1 \\ -1 \\ -1 \end{bmatrix}$$

$$\mathbf{a}(1) = \begin{bmatrix} -1 \\ 0.7 \\ -1 \end{bmatrix}$$

$$\mathbf{a}(2) = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$$

$$\mathbf{a}(3) = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \text{ (Banana)}$$



- Perceptron
 - Feedforward Network
 - Linear Decision Boundary
 - One Neuron for Each Decision
- Hamming Network
 - Competitive Network
 - First Layer – Pattern Matching (Inner Product)
 - Second Layer – Competition (Winner-Take-All)
 - # Neurons = # Prototype Patterns
- Hopfield Network
 - Dynamic Associative Memory Network
 - Network Output Converges to a Prototype Pattern
 - # Neurons = # Elements in each Prototype Pattern



Perceptron Learning Rule



- Supervised Learning

Network is provided with a set of examples of proper network behavior (inputs/targets)

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

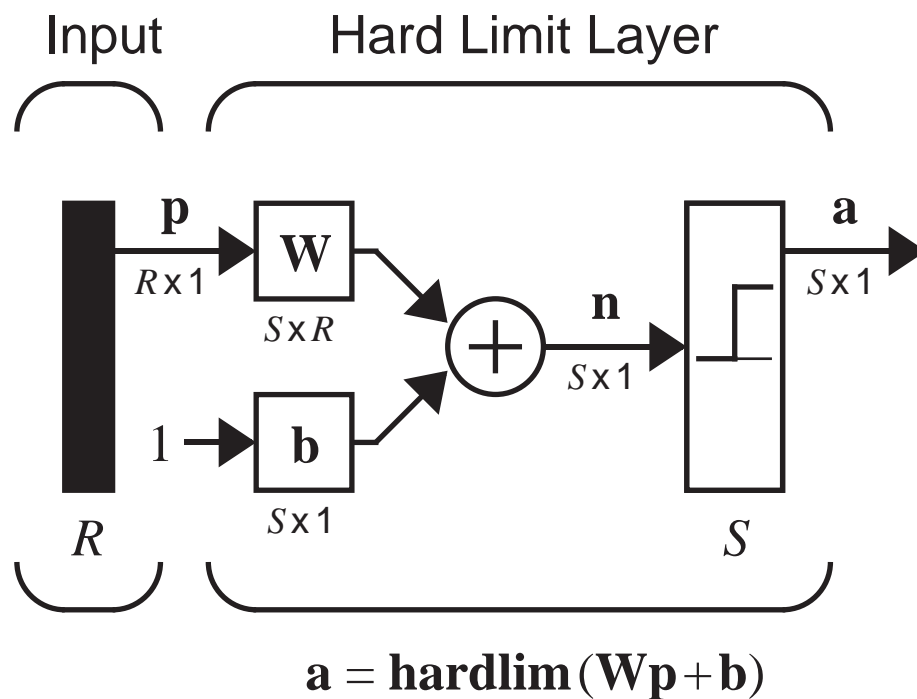
- Reinforcement Learning

Network is only provided with a grade, or score, which indicates network performance

- Unsupervised Learning

Only network inputs are available to the learning algorithm. Network learns to categorize (cluster) the inputs.

Perceptron Architecture



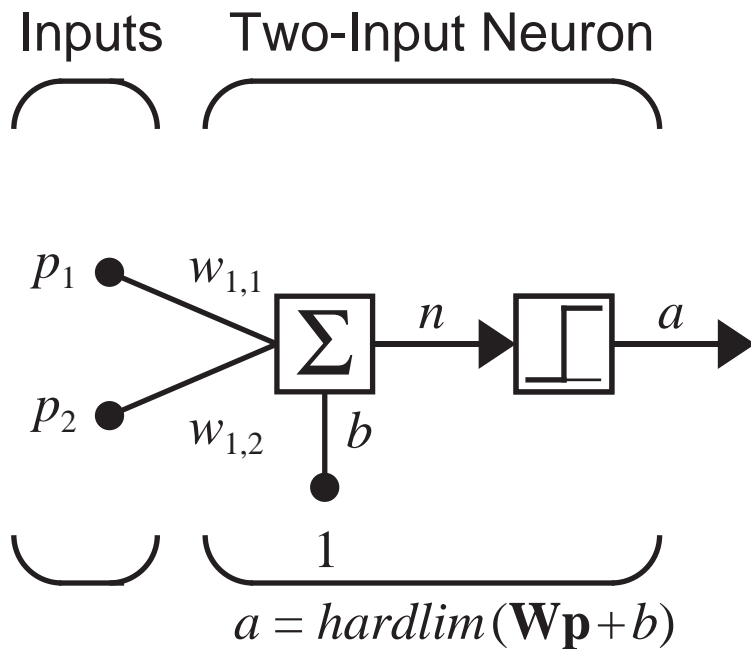
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

$${}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$

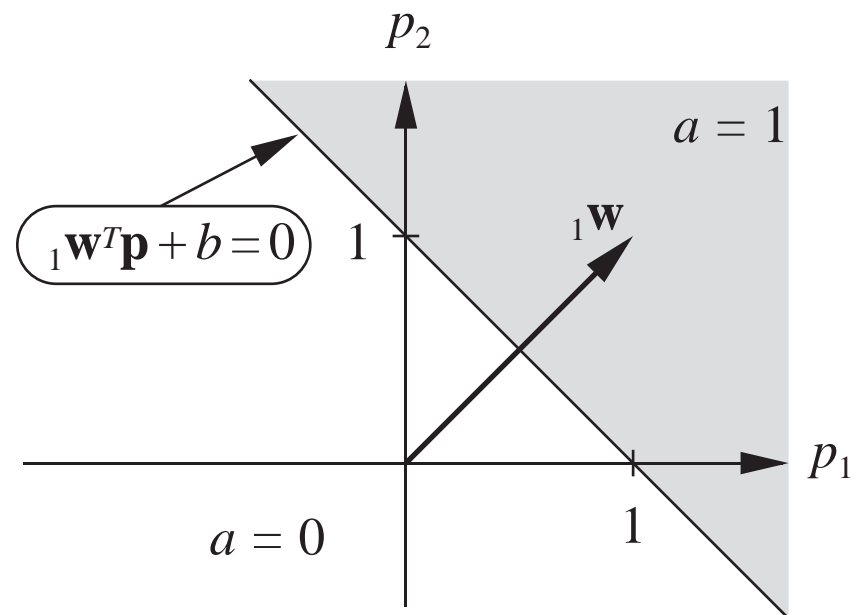
$$\mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

$$a_i = \text{hardlim}(n_i) = \text{hardlim}({}_i\mathbf{w}^T \mathbf{p} + b_i)$$

Single-Neuron Perceptron



$$w_{1,1} = 1 \quad w_{1,2} = 1 \quad b = -1$$



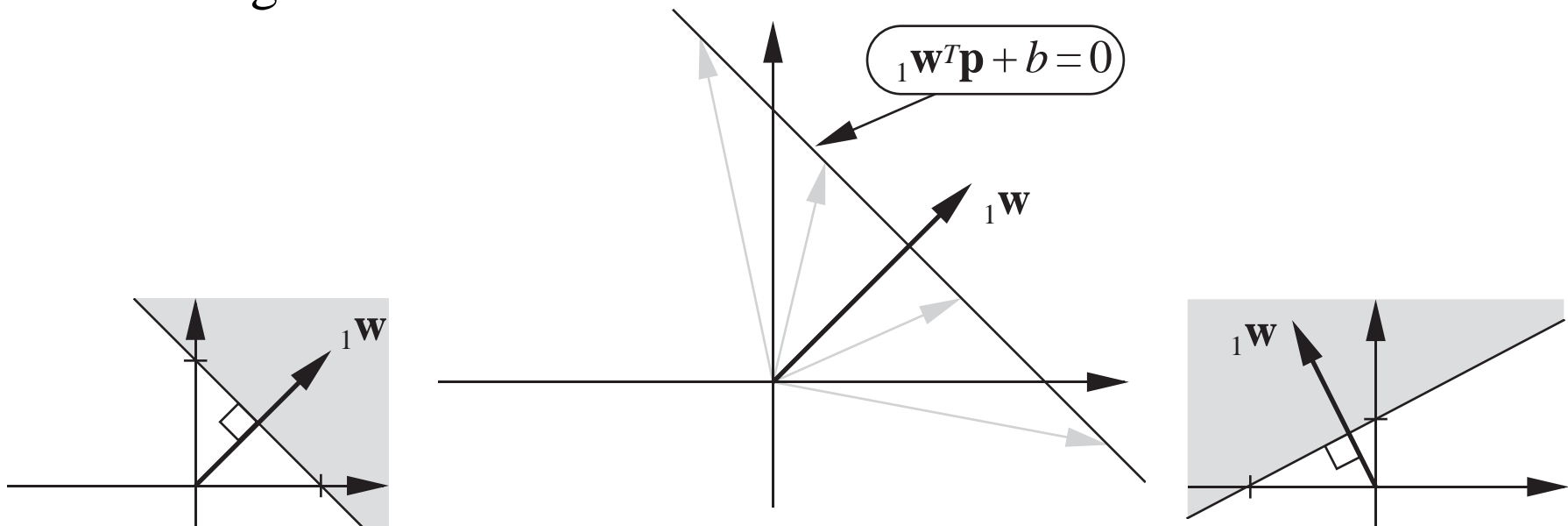
$$a = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p} + b) = \text{hardlim}(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

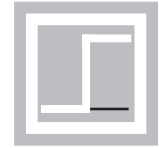
Decision Boundary

$${}_1\mathbf{w}^T \mathbf{p} + b = 0$$

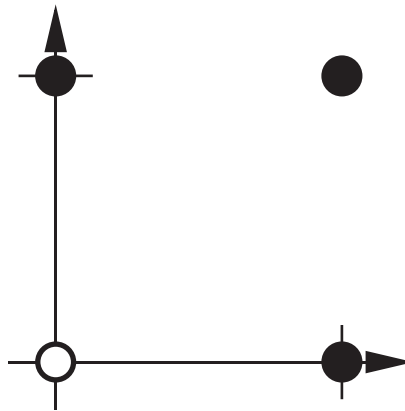
$${}_1\mathbf{w}^T \mathbf{p} = -b$$

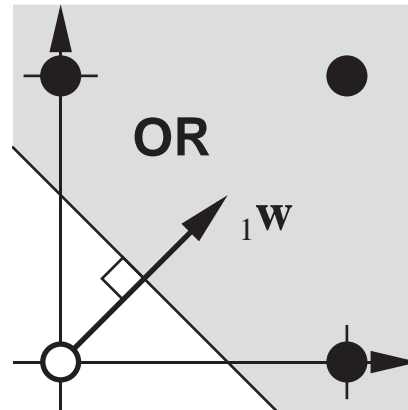
- All points on the decision boundary have the same inner product with the weight vector.
- Therefore they have the same projection onto the weight vector, and they must lie on a line orthogonal to the weight vector





$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, t_1 = 0 \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, t_2 = 1 \right\} \quad \left\{ \mathbf{p}_3 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, t_3 = 1 \right\} \quad \left\{ \mathbf{p}_4 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, t_4 = 1 \right\}$$





Weight vector should be orthogonal to the decision boundary.

$${}_1\mathbf{w} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

Pick a point on the decision boundary to find the bias.

$${}_1\mathbf{w}^T \mathbf{p} + b = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} 0 \\ 0.5 \end{bmatrix} + b = 0.25 + b = 0 \quad \Rightarrow \quad b = -0.25$$



Each neuron will have its own decision boundary.

$${}_i\mathbf{w}^T \mathbf{p} + b_i = 0$$

A single neuron can classify input vectors
into two categories.

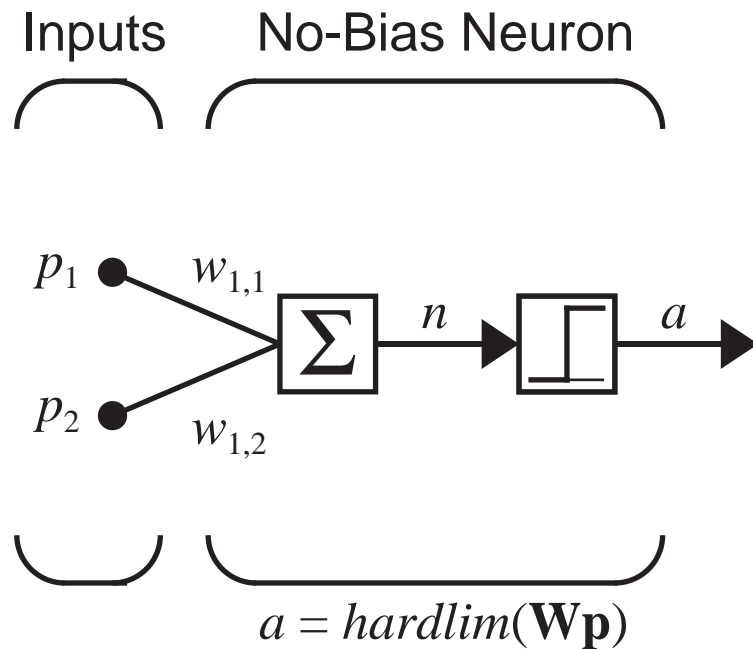
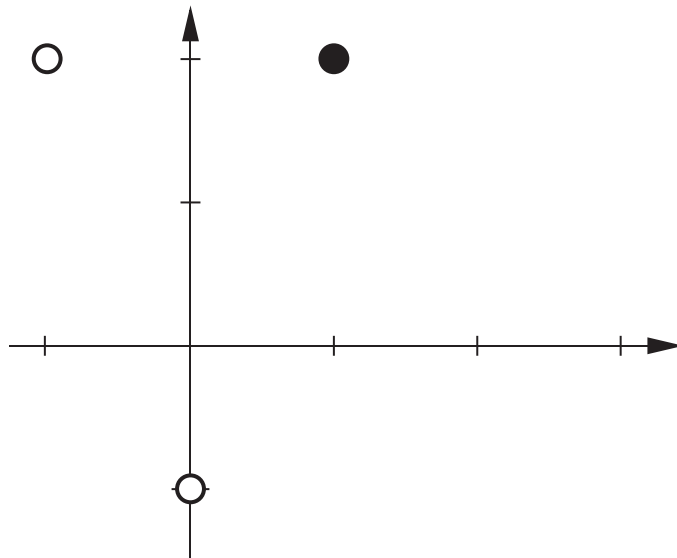
A multi-neuron perceptron can classify
input vectors into 2^S categories.

4

Learning Rule Test Problem

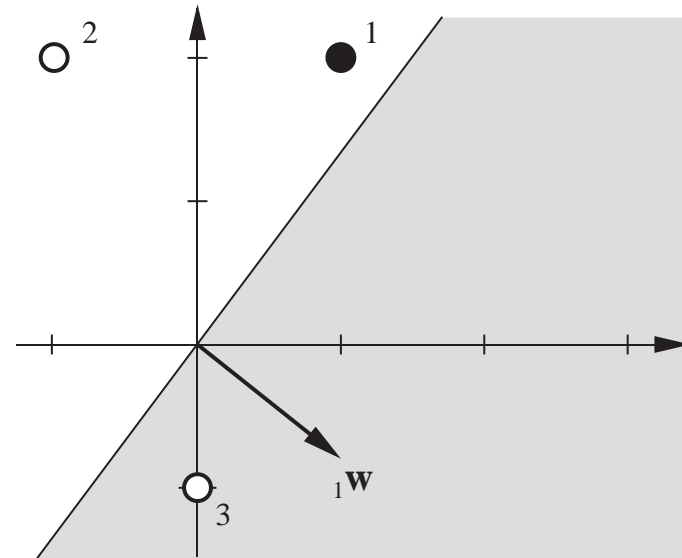
$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, t_1 = 1 \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, t_2 = 0 \right\} \quad \left\{ \mathbf{p}_3 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, t_3 = 0 \right\}$$



Random initial weight:

$${}_1\mathbf{w} = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix}$$



Present \mathbf{p}_1 to the network:

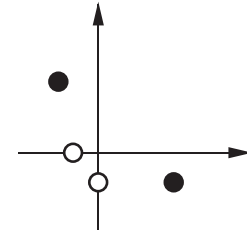
$$a = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p}_1) = \text{hardlim}\left(\begin{bmatrix} 1.0 & -0.8 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$$

$$a = \text{hardlim}(-0.6) = 0$$

Incorrect Classification.

Tentative Learning Rule

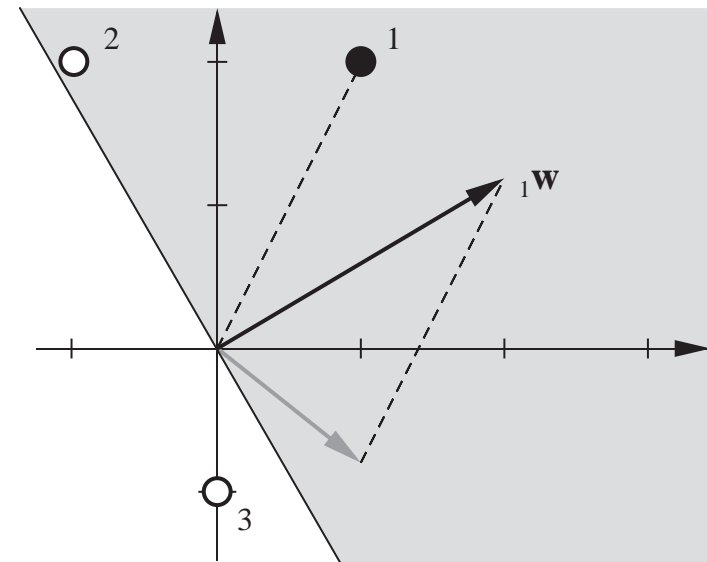
- Set ${}_1\mathbf{w}$ to \mathbf{p}_1
– Not stable \times



- Add \mathbf{p}_1 to ${}_1\mathbf{w}$ \checkmark

Tentative Rule: If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$



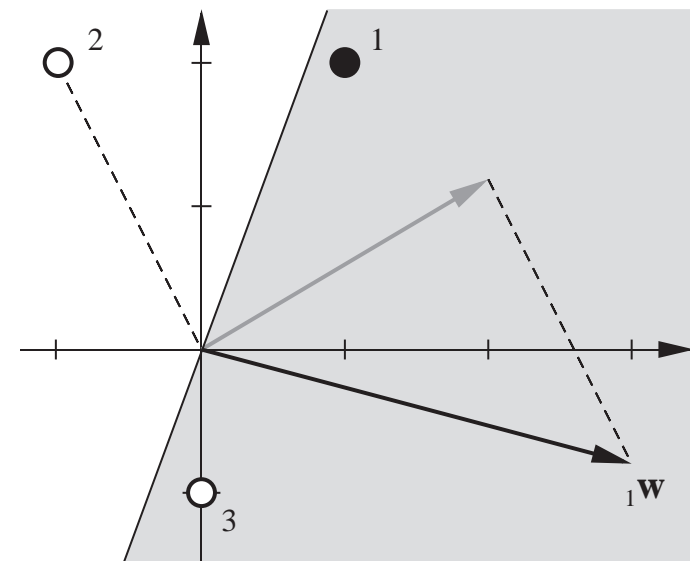
Second Input Vector

$$a = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p}_2) = \text{hardlim}\left(\begin{bmatrix} 2.0 & 1.2 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix}\right)$$

$$a = \text{hardlim}(0.4) = 1 \quad (\text{Incorrect Classification})$$

Modification to Rule: If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix} - \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix}$$

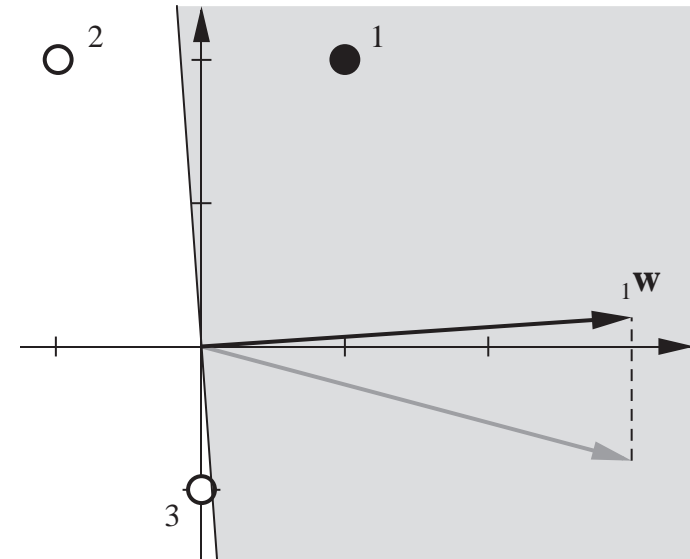


Third Input Vector

$$a = \text{hardlim}({}_1\mathbf{w}^T \mathbf{p}_3) = \text{hardlim}\left(\begin{bmatrix} 3.0 & -0.8 \end{bmatrix} \begin{bmatrix} 0 \\ -1 \end{bmatrix}\right)$$

$$a = \text{hardlim}(0.8) = 1 \quad (\text{Incorrect Classification})$$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix} 3.0 \\ -0.8 \end{bmatrix} - \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 3.0 \\ 0.2 \end{bmatrix}$$



Patterns are now correctly classified.

$$\text{If } t = a, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}.$$

Unified Learning Rule

If $t = 1$ and $a = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $t = 0$ and $a = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $t = a$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

$$e = t - a$$

If $e = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $e = -1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $e = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t - a)\mathbf{p}$$

$$b^{new} = b^{old} + e$$

A bias is a weight with an input of 1.



To update the i th row of the weight matrix:

$${}_i\mathbf{w}^{new} = {}_i\mathbf{w}^{old} + e_i\mathbf{p}$$

$$b_i^{new} = b_i^{old} + e_i$$

Matrix form:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T$$

$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}$$

Apple/Banana Example

Training Set

$$\left\{ \mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, t_1 = \boxed{1} \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, t_2 = \boxed{0} \right\}$$

Initial Weights

$$\mathbf{W} = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \quad b = 0.5$$

First Iteration

$$a = \text{hardlim}(\mathbf{W}\mathbf{p}_1 + b) = \text{hardlim}\left(\begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0.5\right)$$

$$a = \text{hardlim}(-0.5) = 0 \quad e = t_1 - a = 1 - 0 = 1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} + (1)\begin{bmatrix} -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix}$$

$$b^{new} = b^{old} + e = 0.5 + (1) = 1.5$$



$$a = \text{hardlim}(\mathbf{W}\mathbf{p}_2 + b) = \text{hardlim}\left(\begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + (1.5)\right)$$

$$a = \text{hardlim}(2.5) = 1$$

$$e = t_2 - a = 0 - 1 = -1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix} + (-1)\begin{bmatrix} 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -1.5 & -1 & -0.5 \end{bmatrix}$$

$$b^{new} = b^{old} + e = 1.5 + (-1) = 0.5$$



$$a = \text{hardlim}(\mathbf{W}\mathbf{p}_1 + b) = \text{hardlim}\left(\begin{bmatrix} -1.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0.5\right)$$

$$a = \text{hardlim}(1.5) = 1 = t_1$$

$$a = \text{hardlim}(\mathbf{W}\mathbf{p}_2 + b) = \text{hardlim}\left(\begin{bmatrix} -1.5 & -1 & -0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} + 0.5\right)$$

$$a = \text{hardlim}(-1.5) = 0 = t_2$$



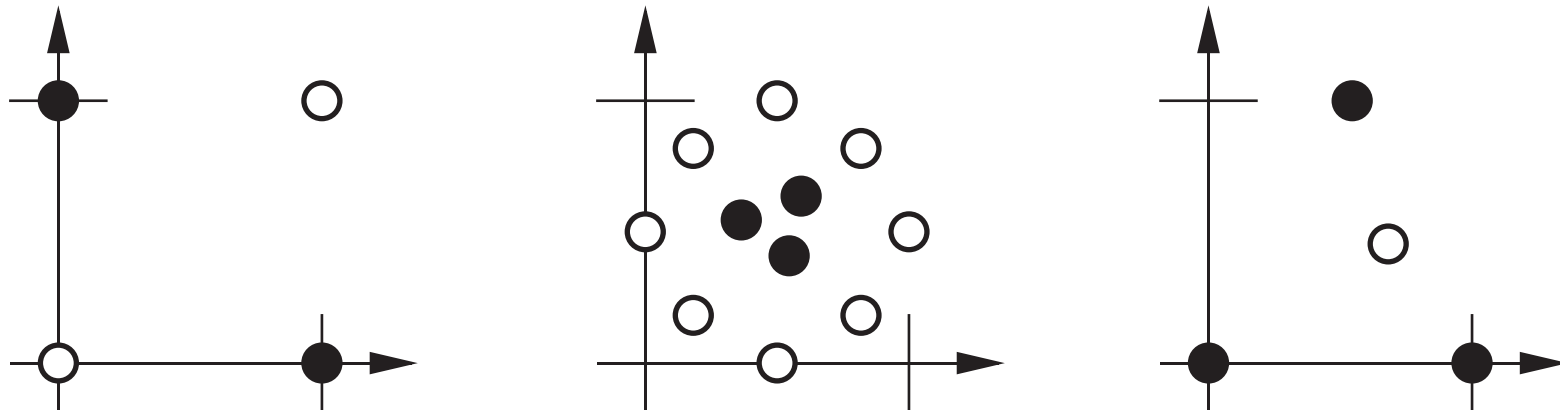
The perceptron rule will always converge to weights which accomplish the desired classification, assuming that such weights exist.



Linear Decision Boundary

$$\mathbf{w}^T \mathbf{p} + b = 0$$

Linearly Inseparable Problems



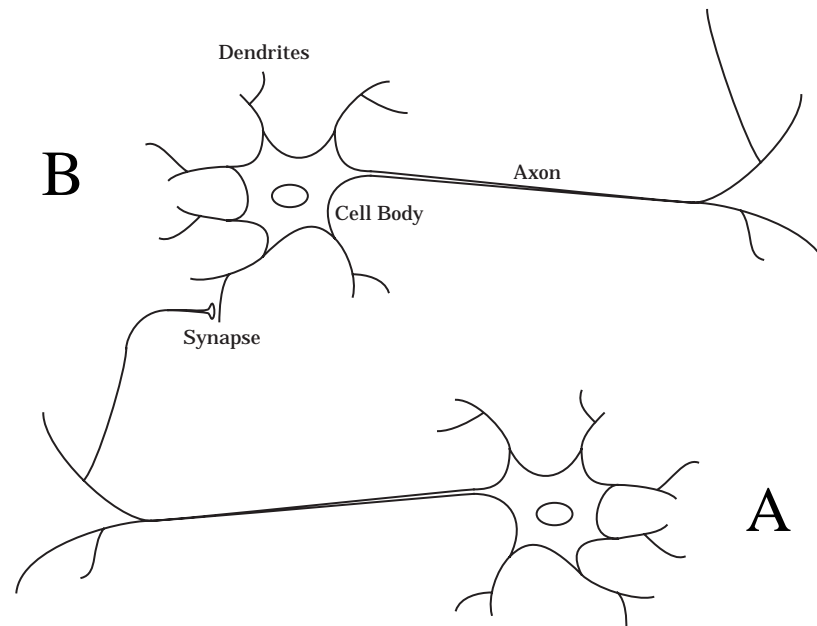


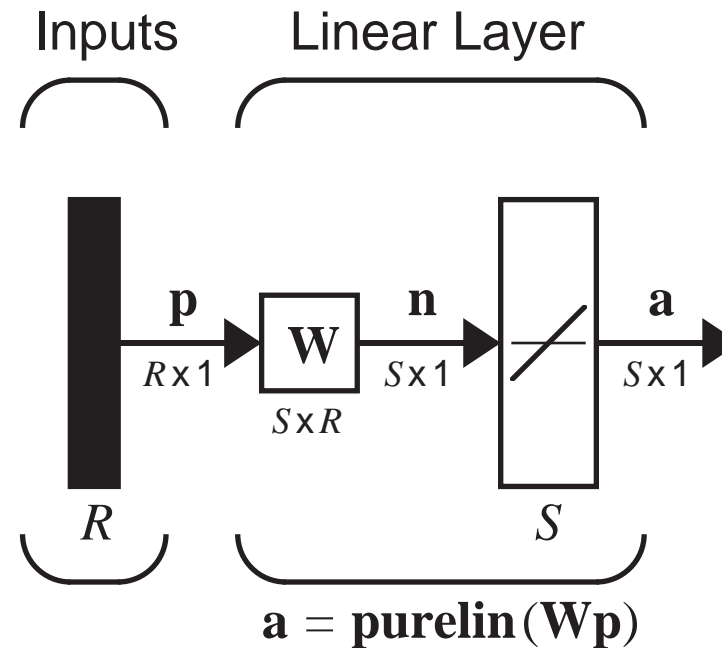
Supervised Hebbian Learning



“When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency, as one of the cells firing B, is increased.”

D. O. Hebb, 1949





$$\mathbf{a} = \mathbf{Wp} \qquad a_i = \sum_{j=1}^R w_{ij} p_j$$

Training Set:

$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \dots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$



Hebb Rule

$$w_{ij}^{new} = w_{ij}^{old} + \alpha f_i(a_{iq}) g_j(p_{jq})$$

↑ Postsynaptic Signal ↑ Presynaptic Signal

Simplified Form:

$$w_{ij}^{new} = w_{ij}^{old} + \alpha a_{iq} p_{jq}$$

Supervised Form:

$$w_{ij}^{new} = w_{ij}^{old} + t_{iq} p_{jq}$$

Matrix Form:

$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T$$

Batch Operation



$$\mathbf{W} = \mathbf{t}_1 \mathbf{p}_1^T + \mathbf{t}_2 \mathbf{p}_2^T + \dots + \mathbf{t}_Q \mathbf{p}_Q^T = \sum_{q=1}^Q \mathbf{t}_q \mathbf{p}_q^T \quad (\text{Zero Initial Weights})$$

Matrix Form:

$$\mathbf{W} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \dots & \mathbf{t}_Q \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_Q^T \end{bmatrix} = \mathbf{T} \mathbf{P}^T$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_Q \end{bmatrix}$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \dots & \mathbf{t}_Q \end{bmatrix}$$



$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \left(\sum_{q=1}^Q \mathbf{t}_q \mathbf{p}_q^T \right) \mathbf{p}_k = \sum_{q=1}^Q \mathbf{t}_q (\mathbf{p}_q^T \mathbf{p}_k)$$

Case I, input patterns are orthogonal.

$$\begin{aligned} (\mathbf{p}_q^T \mathbf{p}_k) &= 1 & q &= k \\ &= 0 & q &\neq k \end{aligned}$$

Therefore the network output equals the target:

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k$$

Case II, input patterns are normalized, but not orthogonal.

$$\mathbf{a} = \mathbf{W}\mathbf{p}_k = \mathbf{t}_k + \underbrace{\sum_{q \neq k} \mathbf{t}_q (\mathbf{p}_q^T \mathbf{p}_k)}_{\text{Error}}$$



Banana

Apple

Normalized Prototype Patterns

$$\mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} \quad \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} \quad \left\{ \mathbf{p}_1 = \begin{bmatrix} -0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix}, \mathbf{t}_1 = [-1] \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix}, \mathbf{t}_2 = [1] \right\}$$

Weight Matrix (Hebb Rule):

$$\mathbf{W} = \mathbf{TP}^T = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} -0.5774 & 0.5774 & -0.5774 \\ 0.5774 & 0.5774 & -0.5774 \end{bmatrix} = \begin{bmatrix} 1.1548 & 0 & 0 \end{bmatrix}$$

Tests:

$$\text{Banana} \quad \mathbf{Wp}_1 = \begin{bmatrix} 1.1548 & 0 & 0 \end{bmatrix} \begin{bmatrix} -0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix} = \begin{bmatrix} -0.6668 \end{bmatrix}$$

$$\text{Apple} \quad \mathbf{Wp}_2 = \begin{bmatrix} 0 & 1.1548 & 0 \end{bmatrix} \begin{bmatrix} 0.5774 \\ 0.5774 \\ -0.5774 \end{bmatrix} = \begin{bmatrix} 0.6668 \end{bmatrix}$$



Performance Index: $\mathbf{W}\mathbf{p}_q = \mathbf{t}_q \quad q = 1, 2, \dots, Q$

$$F(\mathbf{W}) = \sum_{q=1}^Q \|\mathbf{t}_q - \mathbf{W}\mathbf{p}_q\|^2$$

Matrix Form:

$$\mathbf{W}\mathbf{P} = \mathbf{T}$$

$$\mathbf{T} = [\mathbf{t}_1 \ \mathbf{t}_2 \ \dots \ \mathbf{t}_Q] \quad \mathbf{P} = [\mathbf{p}_1 \ \mathbf{p}_2 \ \dots \ \mathbf{p}_Q]$$

$$F(\mathbf{W}) = \|\mathbf{T} - \mathbf{W}\mathbf{P}\|^2 = \|\mathbf{E}\|^2$$

$$\|\mathbf{E}\|^2 = \sum_i \sum_j e_{ij}^2$$



$$\mathbf{W}\mathbf{P} = \mathbf{T}$$

Minimize:

$$F(\mathbf{W}) = \|\mathbf{T} - \mathbf{W}\mathbf{P}\|^2 = \|\mathbf{E}\|^2$$

If an inverse exists for \mathbf{P} , $F(\mathbf{W})$ can be made zero:

$$\mathbf{W} = \mathbf{T}\mathbf{P}^{-1}$$

When an inverse does not exist $F(\mathbf{W})$ can be minimized using the pseudoinverse:

$$\mathbf{W} = \mathbf{T}\mathbf{P}^+$$

$$\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T$$



Hebb Rule

$$\mathbf{W} = \mathbf{T}\mathbf{P}^T$$

Pseudoinverse Rule

$$\mathbf{W} = \mathbf{T}\mathbf{P}^+$$

$$\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T$$

If the prototype patterns are orthonormal:

$$\mathbf{P}^T\mathbf{P} = \mathbf{I}$$

$$\mathbf{P}^+ = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T = \mathbf{P}^T$$



$$\left\{ \mathbf{p}_1 = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{t}_1 = [-1] \right\} \quad \left\{ \mathbf{p}_2 = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}, \mathbf{t}_2 = [1] \right\} \quad \mathbf{W} = \mathbf{TP}^+ = [-1 \ 1] \left(\begin{bmatrix} -1 & 1 \\ 1 & 1 \\ -1 & -1 \end{bmatrix} \right)^+$$

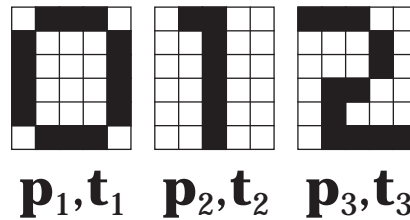
$$\mathbf{P}^+ = (\mathbf{P}^T \mathbf{P})^{-1} \mathbf{P}^T = \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}^{-1} \begin{bmatrix} -1 & 1 & -1 \\ 1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0.25 & -0.25 \\ 0.5 & 0.25 & -0.25 \end{bmatrix}$$

$$\mathbf{W} = \mathbf{TP}^+ = [-1 \ 1] \begin{bmatrix} -0.5 & 0.25 & -0.25 \\ 0.5 & 0.25 & -0.25 \end{bmatrix} = [1 \ 0 \ 0]$$

$$\mathbf{Wp}_1 = [1 \ 0 \ 0] \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} = [-1]$$

$$\mathbf{Wp}_2 = [1 \ 0 \ 0] \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix} = [1]$$

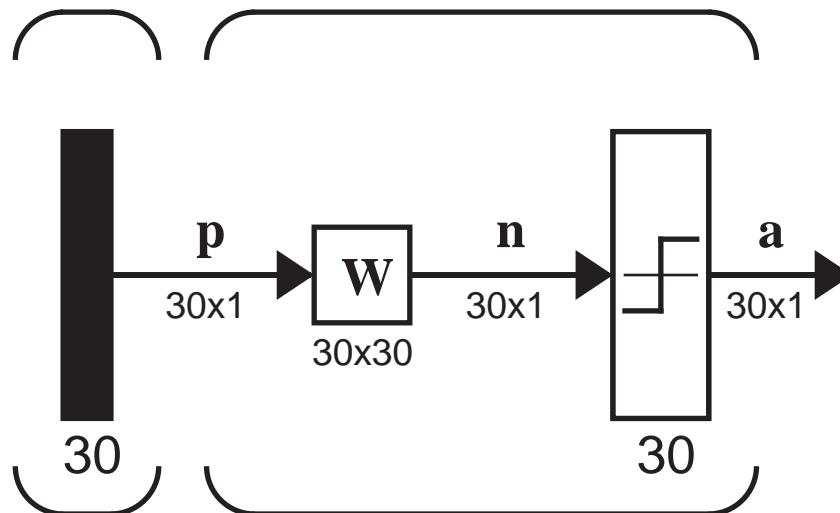
Autoassociative Memory



$$\mathbf{p}_1 = [-1 \ 1 \ 1 \ 1 \ 1 \ -1 \ 1 \ -1 \ -1 \ -1 \ -1 \ 1 \ 1 \ -1 \ \dots \ 1 \ -1]^T$$

Inputs

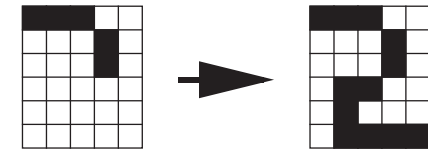
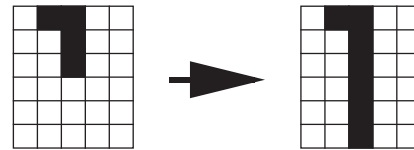
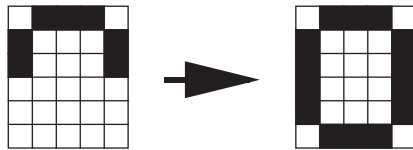
Sym. Hard Limit Layer



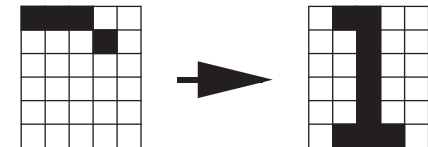
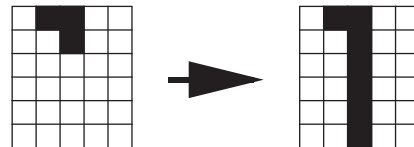
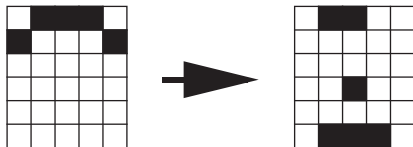
$$\mathbf{W} = \mathbf{p}_1 \mathbf{p}_1^T + \mathbf{p}_2 \mathbf{p}_2^T + \mathbf{p}_3 \mathbf{p}_3^T$$

$$\mathbf{a} = \text{hardlims}(\mathbf{W}\mathbf{p})$$

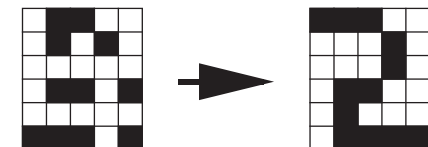
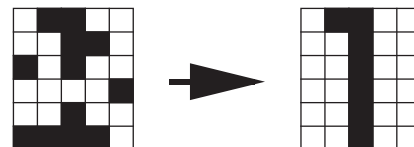
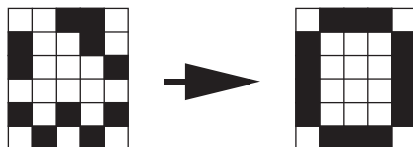
50% Occluded



67% Occluded



Noisy Patterns (7 pixels)





Basic Rule: $\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{t}_q \mathbf{p}_q^T$

Learning Rate: $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T$

Smoothing: $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T - \gamma \mathbf{W}^{old} = (1 - \gamma) \mathbf{W}^{old} + \alpha \mathbf{t}_q \mathbf{p}_q^T$

Delta Rule: $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha (\mathbf{t}_q - \mathbf{a}_q) \mathbf{p}_q^T$

Unsupervised: $\mathbf{W}^{new} = \mathbf{W}^{old} + \alpha \mathbf{a}_q \mathbf{p}_q^T$

Examples



ELSEVIER

journal homepage: www.intl.elsevierhealth.com/journals/cmpb

Long-term heart rate variability as a predictor of patient age

Valentina D.A. Corino^{a,*}, Matteo Matteucci^b, Luca Cravello^c,
Ettore Ferrari^c, Antonio A. Ferrari^d, Luca T. Mainardi^a

^a Department of Biomedical Engineering, Polytechnic University of Milan, Milan, Italy

^b Department of Electronic and Information, Polytechnic University of Milan, Milan, Italy

^c Department of Internal Medicine and Medical Therapy, University of Pavia, Italy

^d IRCCS "S. Matteo" Polyclinic, Preside of Belgioioso, Pavia, Italy

- **biological** age is different from the **chronological** one
 - biological markers of aging could be of great interest
 - cardiovascular system is affected by age, increasing age seems to be associated with a reduction of both the overall HRV and the complexity of interbeat dynamics
 - could the healthy subject age be estimated from his cardiovascular conditions??
- AIM:** to predict subject (biological) age from HRV parameters

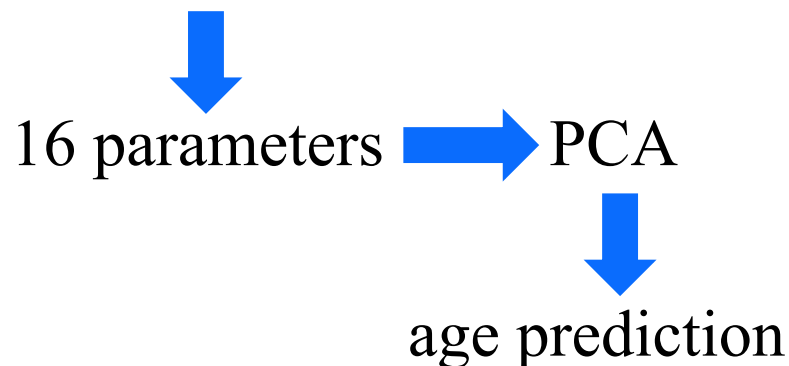
- 131 healthy subjects of different age (20–85 years old)


Table 1 – Mean age and number of subjects in the four selected classes

Class	# Subjects		Age range (min–max)	Age (mean \pm S.D.)
	Male	Female		
Young	15	24	20–40	30.7 \pm 4.6
Adult	12	14	41–60	51.3 \pm 6.4
Elderly	18	28	61–70	65.0 \pm 2.5
Senescent	9	11	71–85	76.8 \pm 4.6

- long-term analysis (24-h ECG recordings)

- traditional linear time parameters (M, SDNN, SDANN, SDNNi, pNN50, rMSSD, poincarè analysis)
- frequency domain parameters (ULF, VLF, LF, HF power, alpha slope)
- non-linear metrics (ApEn, DFA parameters)



- the entire dataset was divided, sampling uniformly, in three subsets:
 1. group I (training set) data used to train the network (67%);
 2. group II (validation set) data used to validate the network and to choose the best network parameters (22%);
 3. group III (test set) data used to test the final network and examine its behaviour with never-seen data (11%).
- the selection of the data for the three subsets may influence the performance of the neural networks  3 different uniform samplings then the results averaged
- 3, 4, 6 and all PCs corresponding to 90%, 95%, 99% and 100% of kept variance

- Classifiers:
 1. *Robust linear regression*
 2. *Feedforward neural network*
 3. *Radial basis function neural network*

Robust linear regression

- uses an iteratively reweighted least-squares algorithm, in which weights are recalculated, at each iteration, by applying the bisquare function to the residuals of the previous iteration

Feedforward neural network

- Levenberg–Marquardt training algorithm
- **Early stopping technique** used to improve the generalization of the network and to avoid overtraining (the error on the validation set was monitored during the training phase and when the error began to increase, training was stopped)
- two-layer network with three neurons in the first layer and one neuron in the second (output) layer (the transfer function was a hyperbolic tangent in the first layer and linear in the second one)

Radial basis functions neural network

- the network is created by adding one neuron at a time. Neurons are added to the network until the sum-squared error falls beneath an error goal or a maximum number of neurons has been reached.
- One hidden radial basis layer and a linear output layer
- Generally, RBF tend to have more neurons than a comparable FFNN because radial basis neurons only respond to relatively small regions of the input space

8

Ex 1: Results

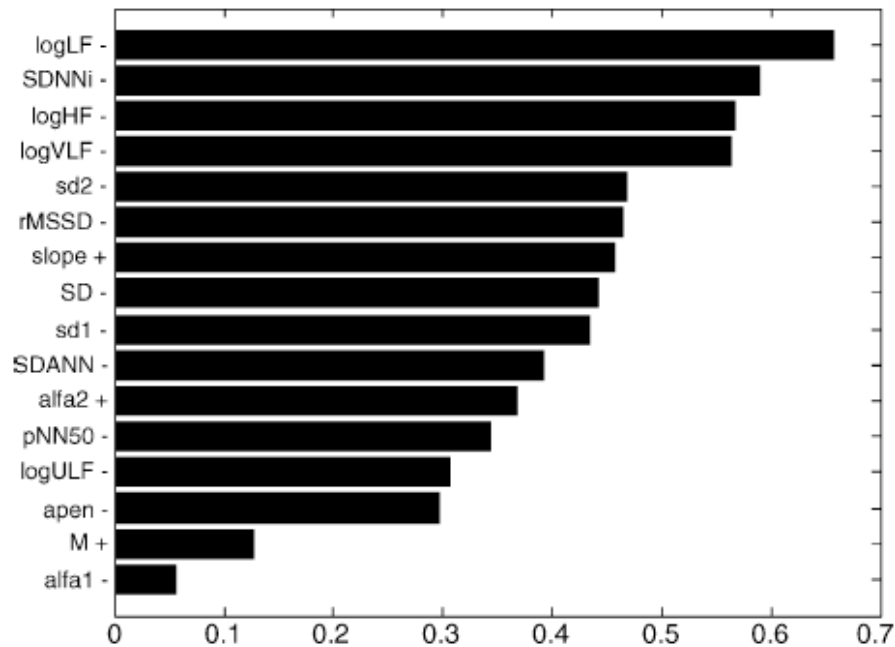


Fig. 3 – Correlation between HRV parameters and age. Pearson correlation coefficient computed between each HRV parameter and real age in the whole set of subjects. (–) Negative correlation; (+) positive correlation.

Table 2 – Factor loadings between the first four PCs and the significantly correlated variables are reported

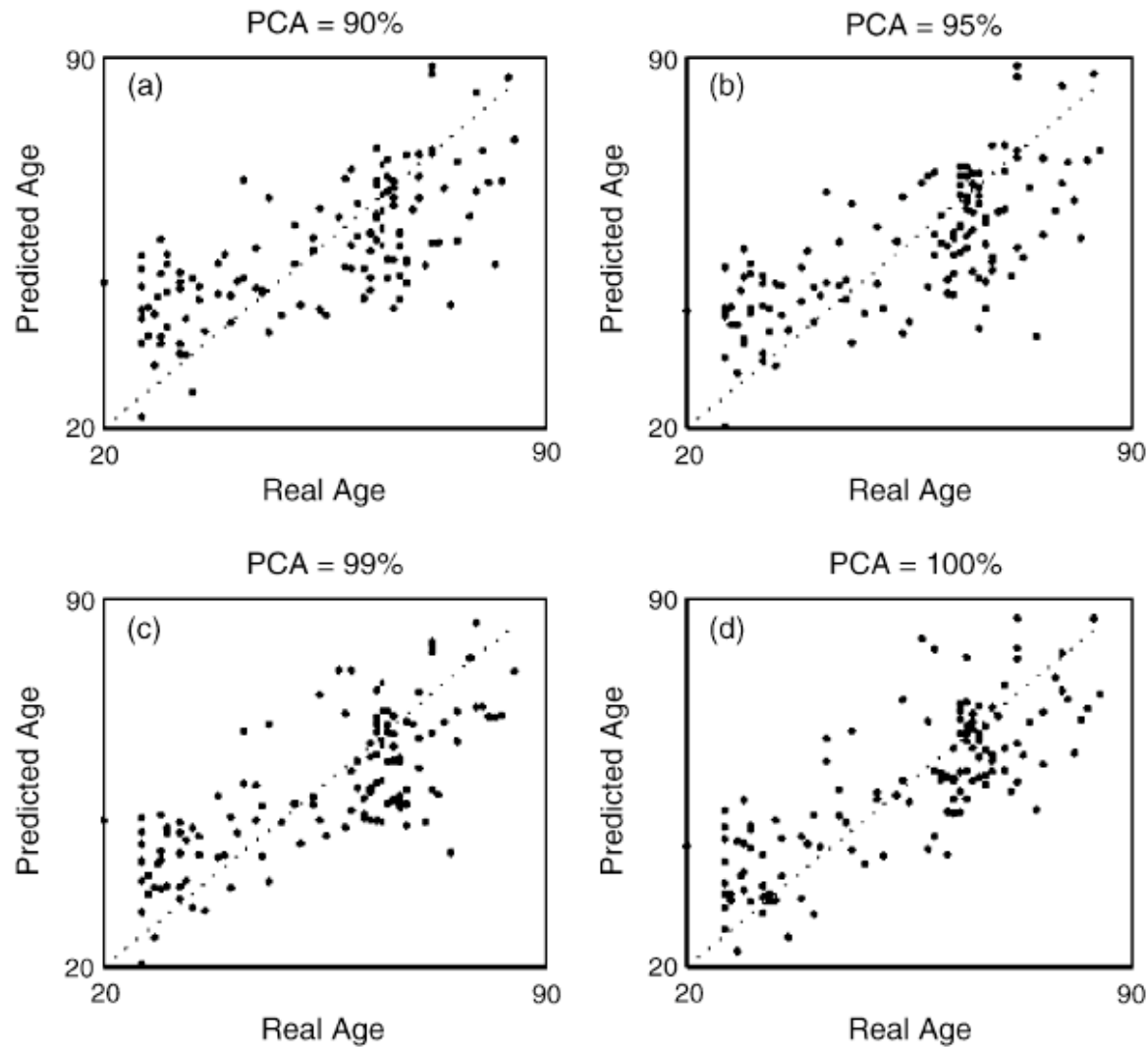
	PC1	PC2	PC3	PC4
logHF	0.314	–0.190	–0.020	0.082
logLF	0.307	–0.044	0.304	0.010
logVLF	0.313	0.100	0.115	–0.261
SDNNi	0.336	–0.010	0.053	–0.109
SDANN	0.249	0.346	–0.026	0.269
ApEn	0.155	–0.494	–0.062	–0.177
α_1	–0.048	0.343	0.552	–0.299
α_2	–0.117	0.221	–0.531	–0.303
Slope β	–0.146	0.230	–0.342	0.370
M	0.149	0.102	–0.283	–0.602

In bold the most relevant variables in each component.

8

Ex 1: Results

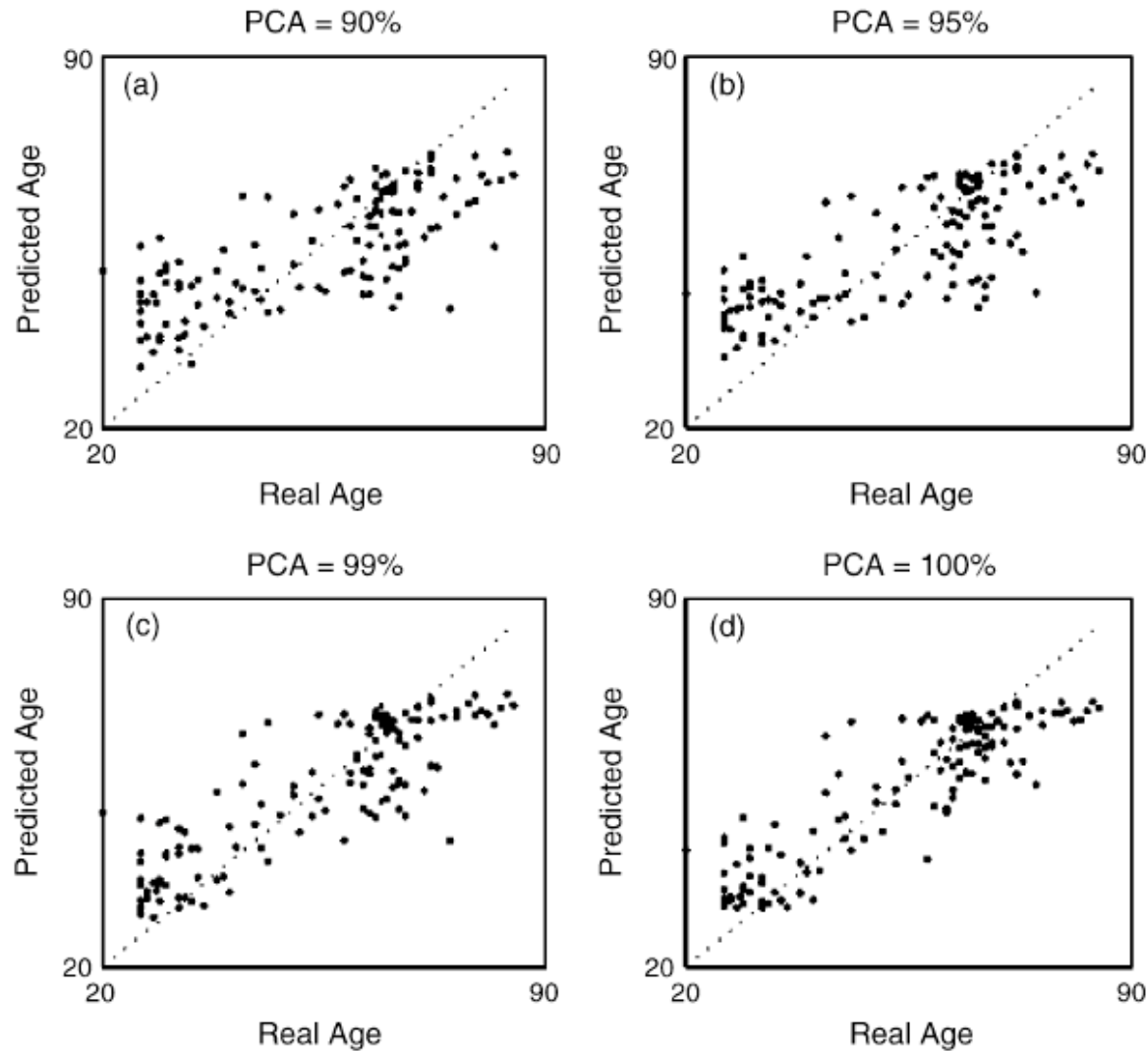
Robust Linear Regression



8

Ex 1: Results

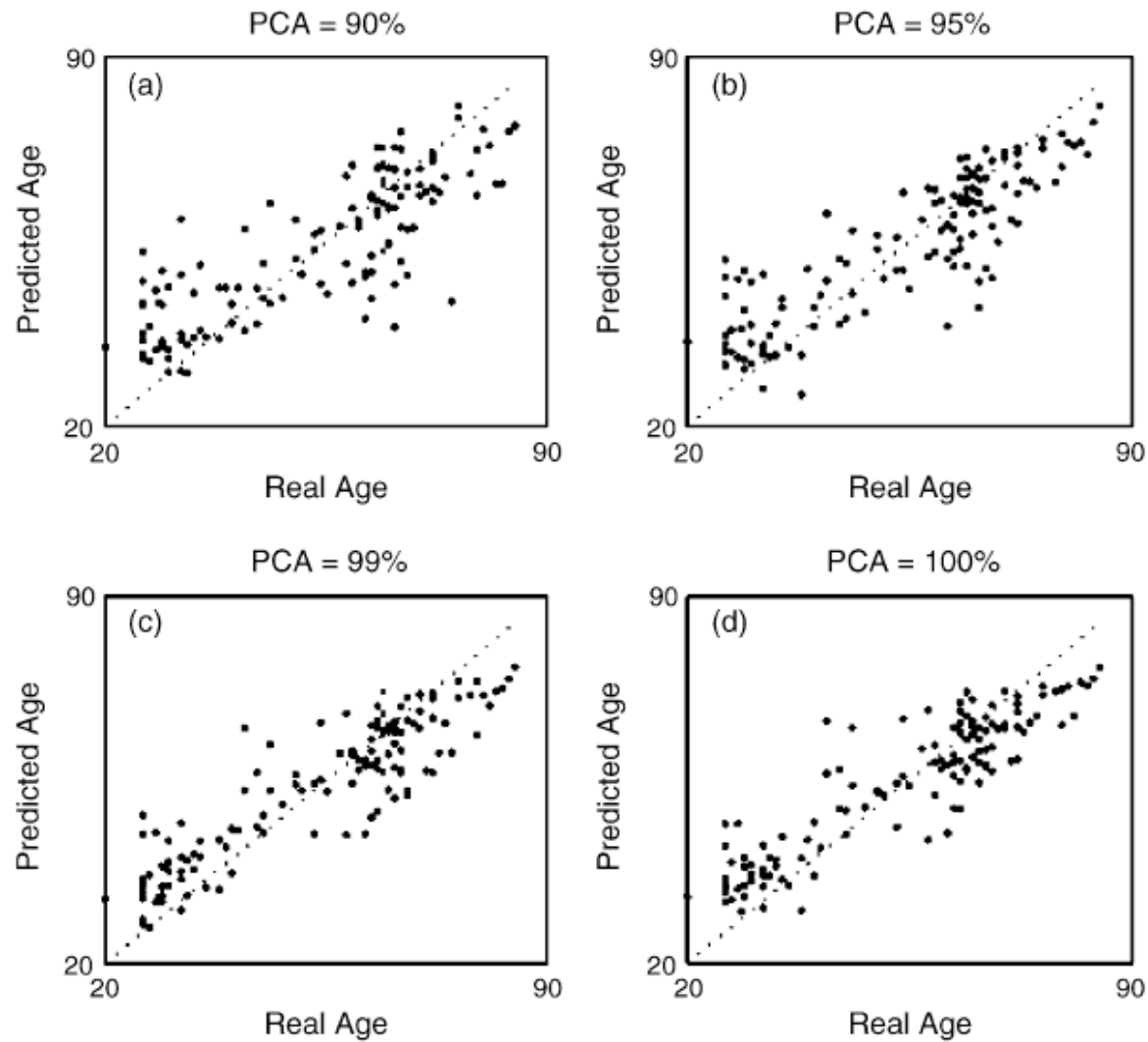
Feedforward NN




8

Ex 1: Results

Radial Basis Function NN



- NN were able to learn how to approximate the complex function relating HRV and age, without losing generalisation capability (good performance even in test set)  the underlying relation is highly non-linear
- age in younger subjects is overestimated whereas it is underestimated in older ones

Example 2

Automatic classification of the Cyclic
Alternating Pattern (CAP)

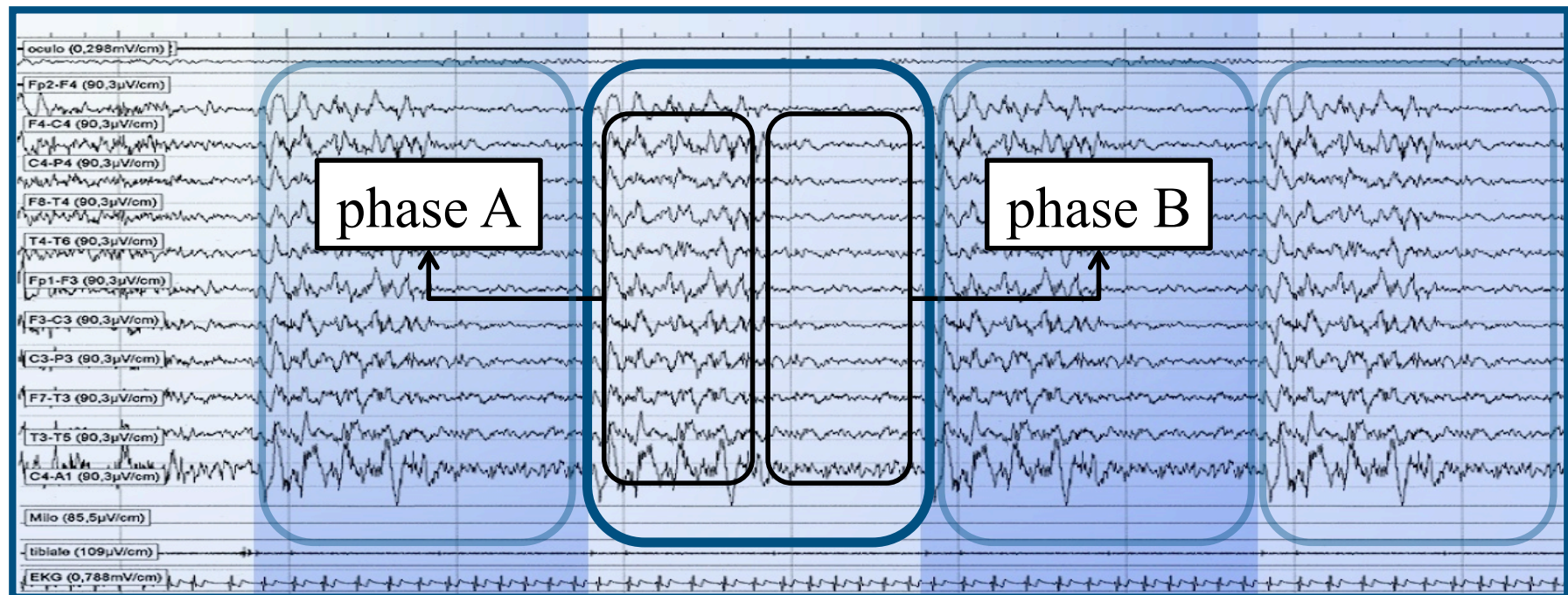
8

Ex 2: Introduction

CAP is a periodic activity on the EEG during NREM sleep made of

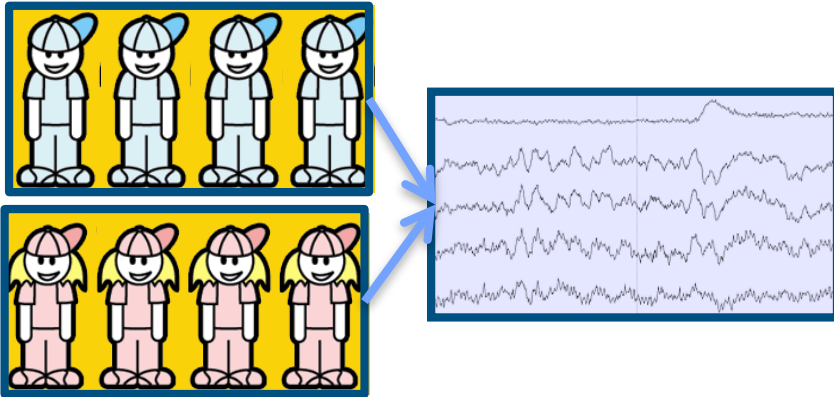
An activation phase: **phase A**

A second phase between two phases A where only the background is visible: **phase B**



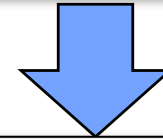
8

Ex 2: Dataset



From the EEG we extracted **7 features** capable of distinguishing CAP A phases from CAP B phases:

- 8 healthy, young subjects
- A single 8-hour EEG trace for each subject (C3-A2 or C4-A1)

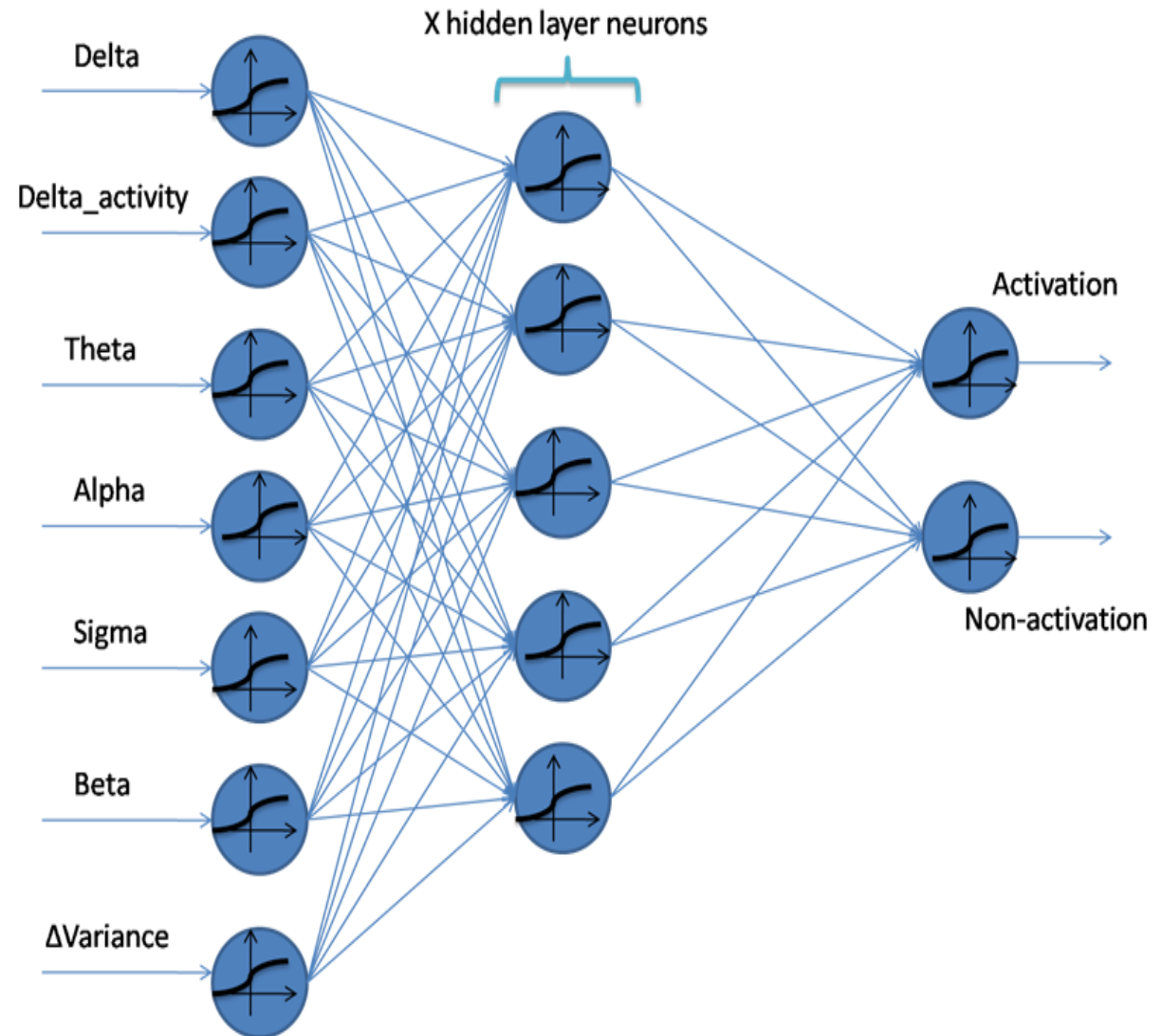


1. Delta descriptor
2. Theta descriptor
3. Alpha descriptor
4. Sigma descriptor
5. Beta descriptor
6. Delta Hjorth activity
7. Differential EEG variance

8

Ex 2: Structure of the NN

- 3-layer Automatic Neural Network
- Labels: 1 for phase A, 0 for phase B
- Training algorithm: backpropagation with gradient descent
- Activation function: *logsig*



- The data were classified with the Leave One Out method: for each patient to be classified, the remaining seven were divided into training, testing and validation set.
- The number x of the hidden layer neurons varied from 2 to 30. The classification error was minimized.
- Each training was restarted 10 times in order to avoid local minima problems.
- The different partitionings of the data lead to 7 “optimal” neural networks for each patient
- The 7 networks were simulated on the patient to classify, and the final classification was obtained as the rounded mean of the 7 results.

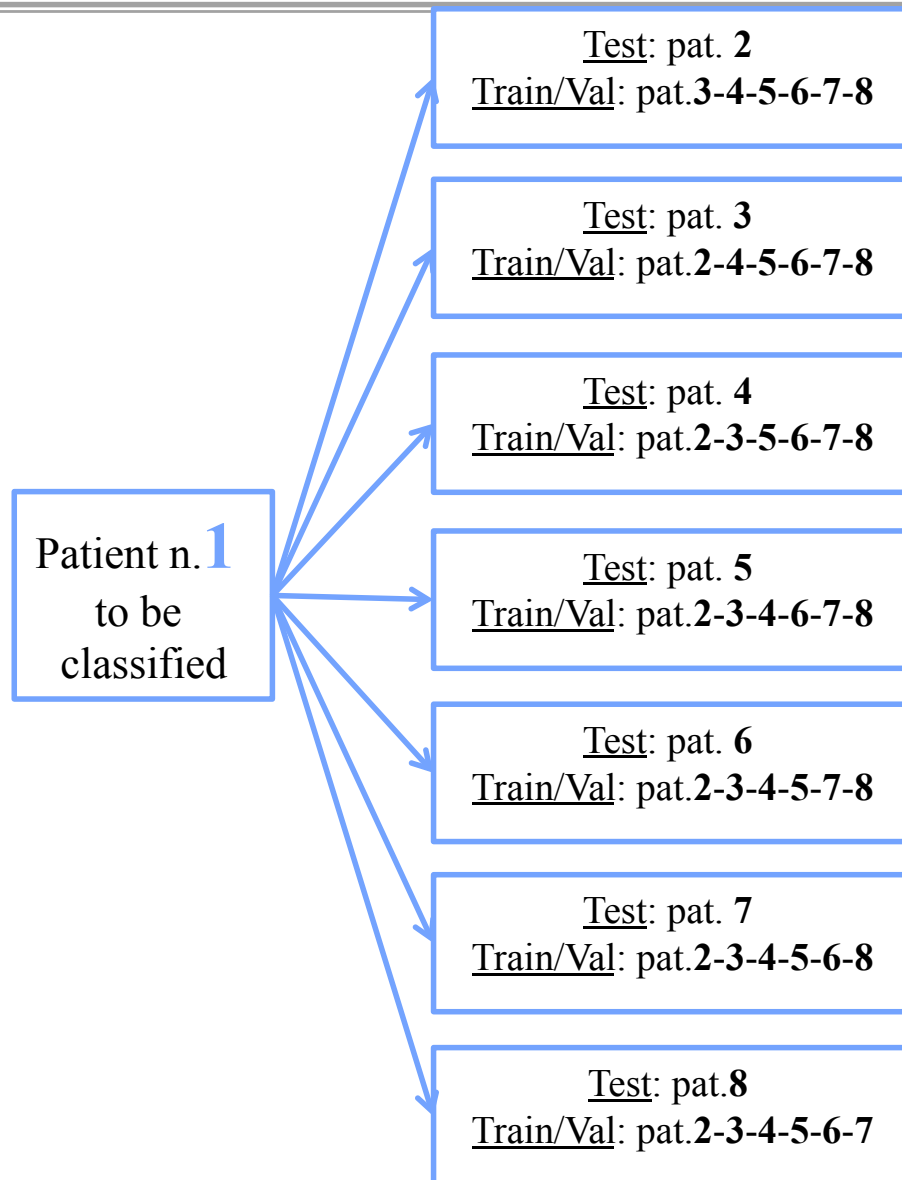
8

Ex 2: Methods

Patient n. **1**
to be
classified

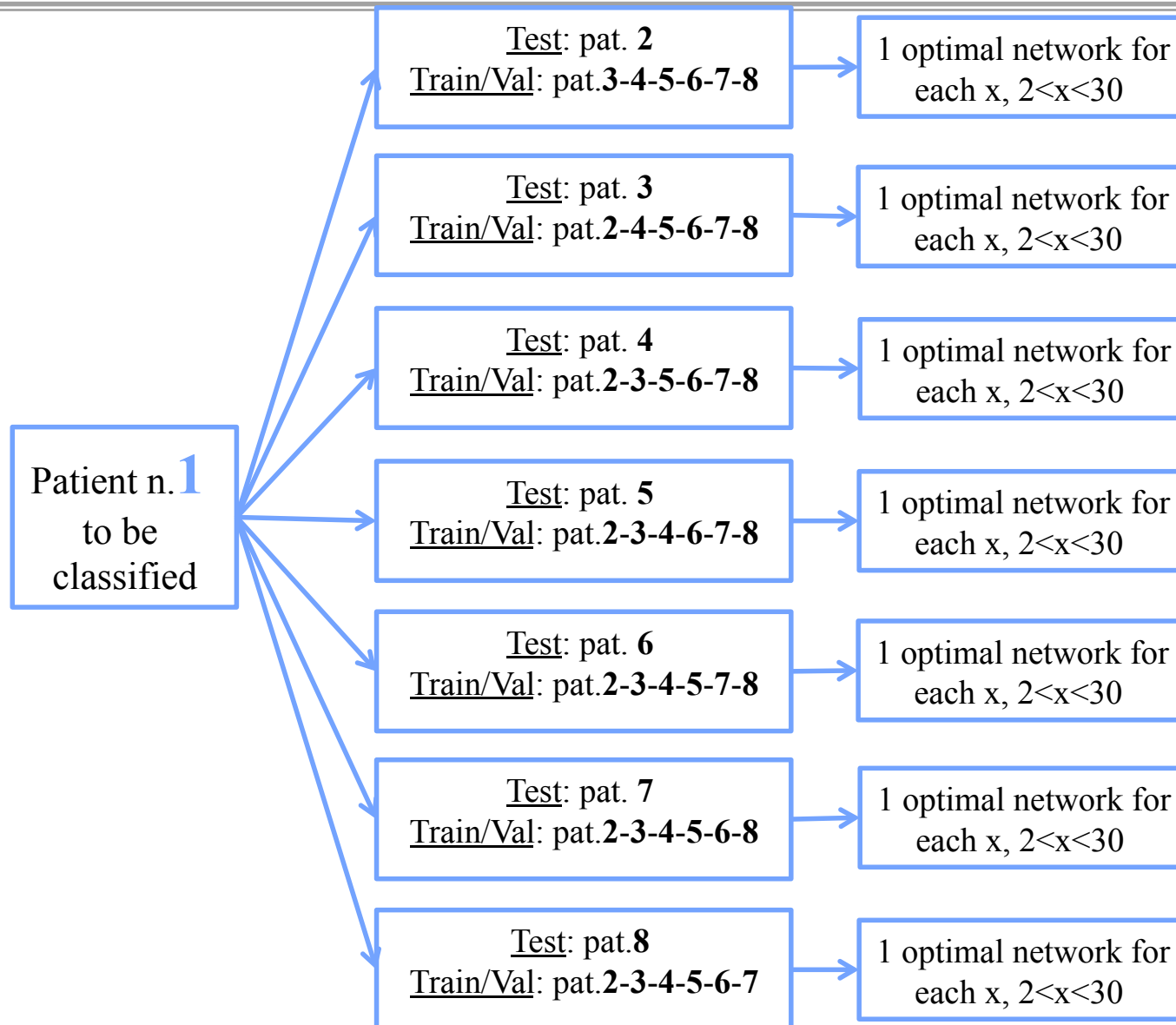
8

Ex 2: Methods



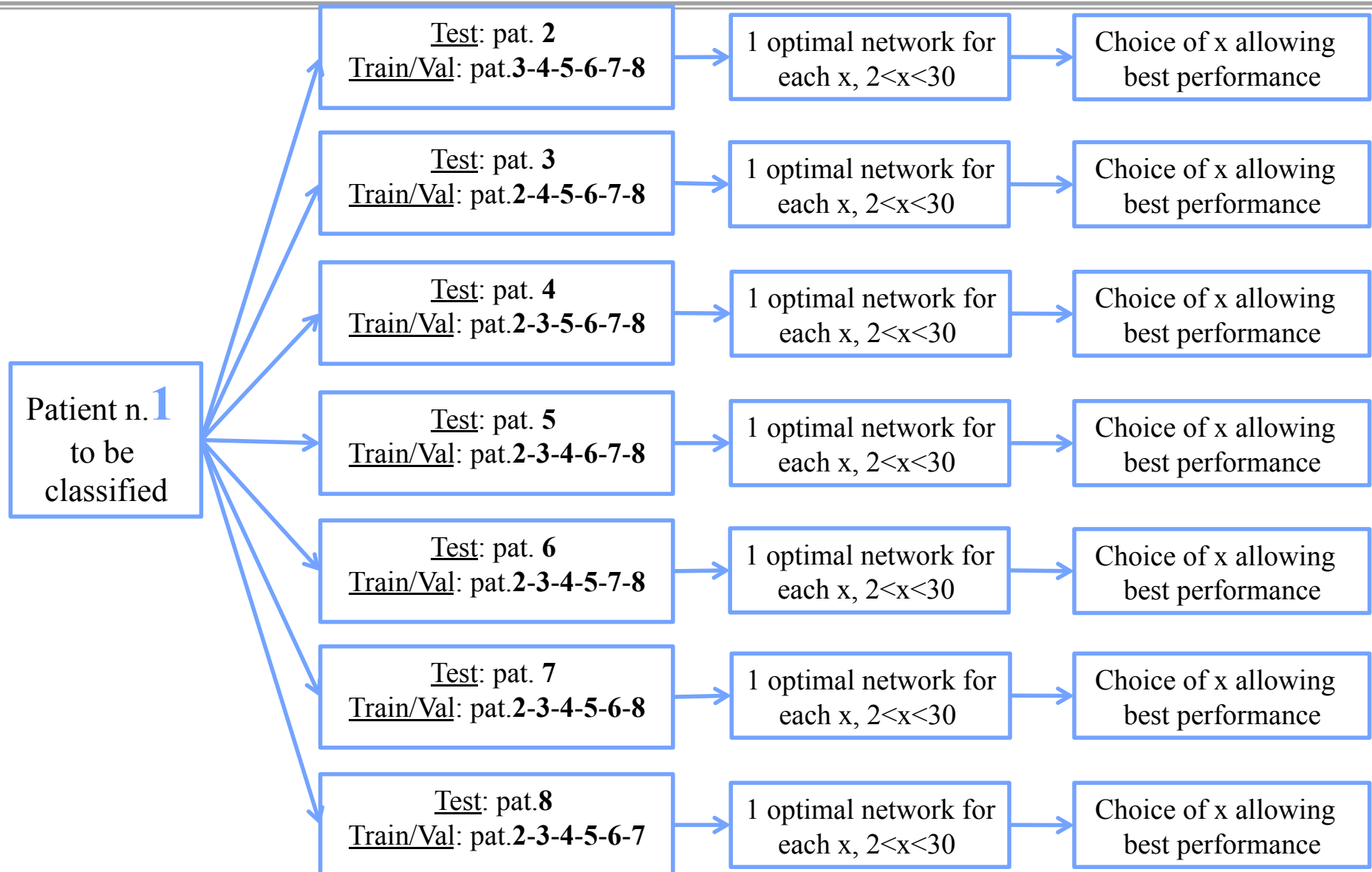
8

Ex 2: Methods



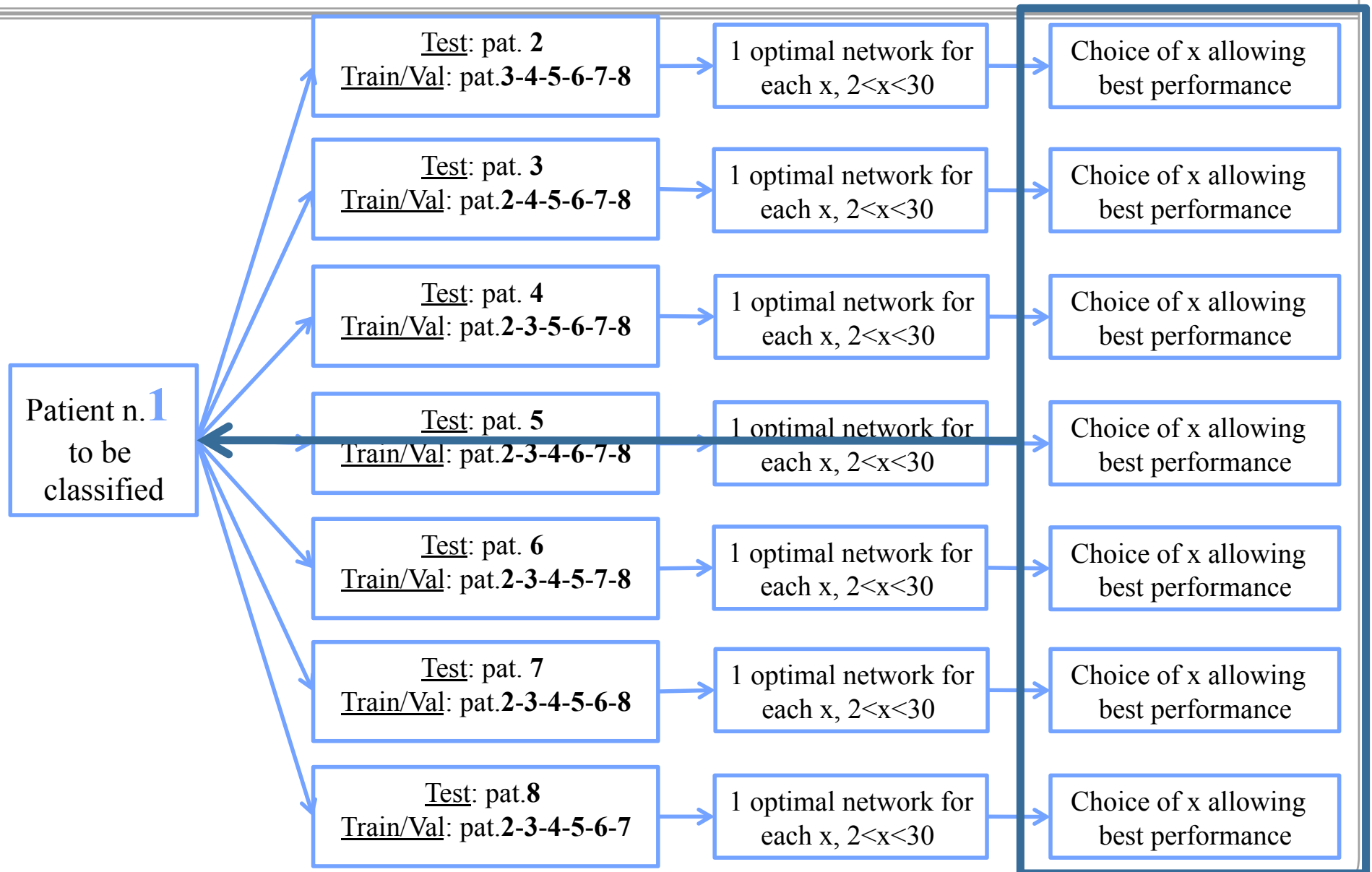
8

Ex 2: Methods



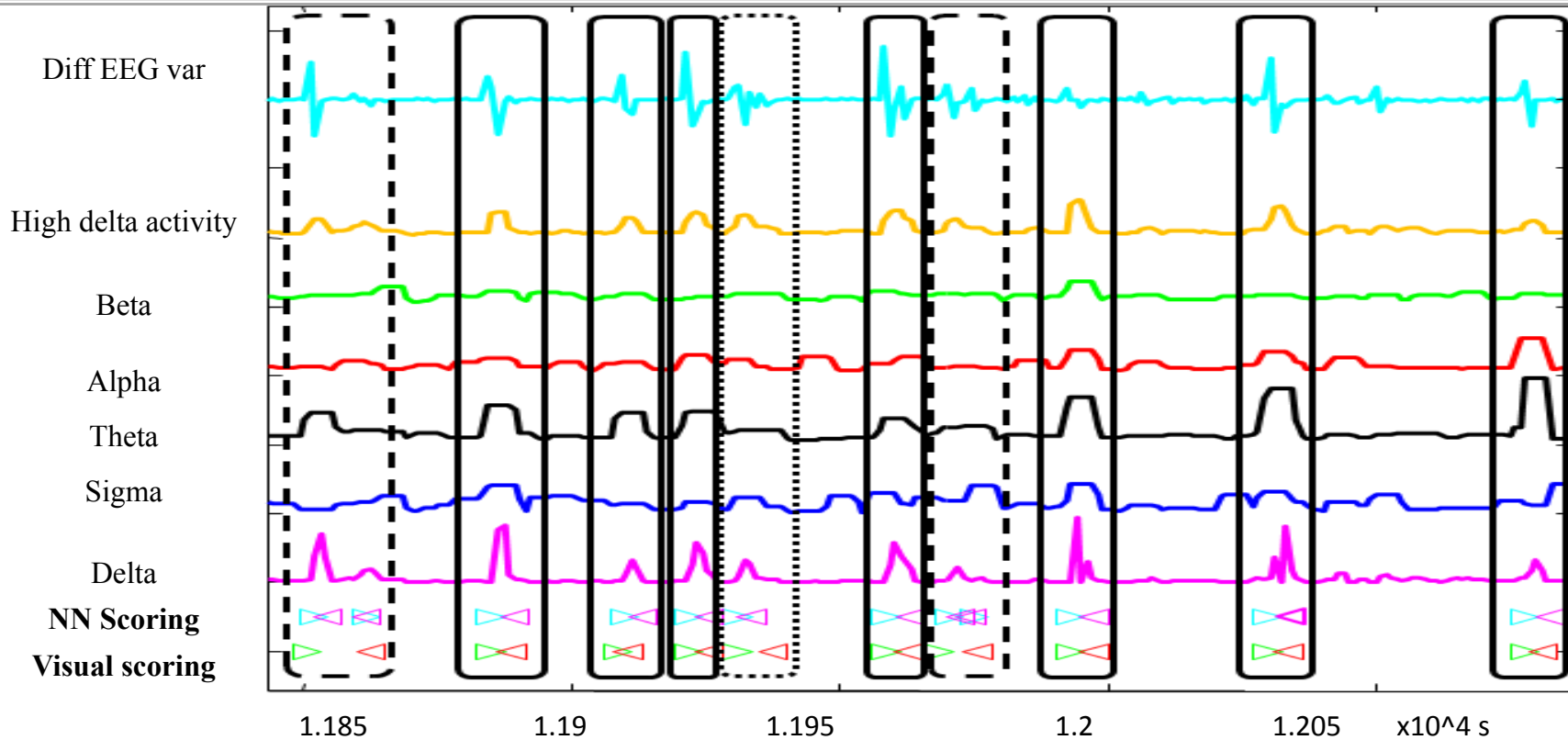
8

Ex 2: Methods



8

Ex 2: Results



	Sensitivity (%)	Specificity (%)	Accuracy (%)
MEAN	74.3	82.6	80.7

Automatic Detection of A phases of the Cyclic Alternating Pattern during Sleep, S. Mariani, E. Manfredini, M. O. Mendez, A. M. Bianchi, M. G. Terzano, S. Cerutti – under publication

- Good accuracy, specificity and sensitivity
- From a clinical point of view, the automatic classification obtained through the NN is very promising (now it is done *manually!!*)