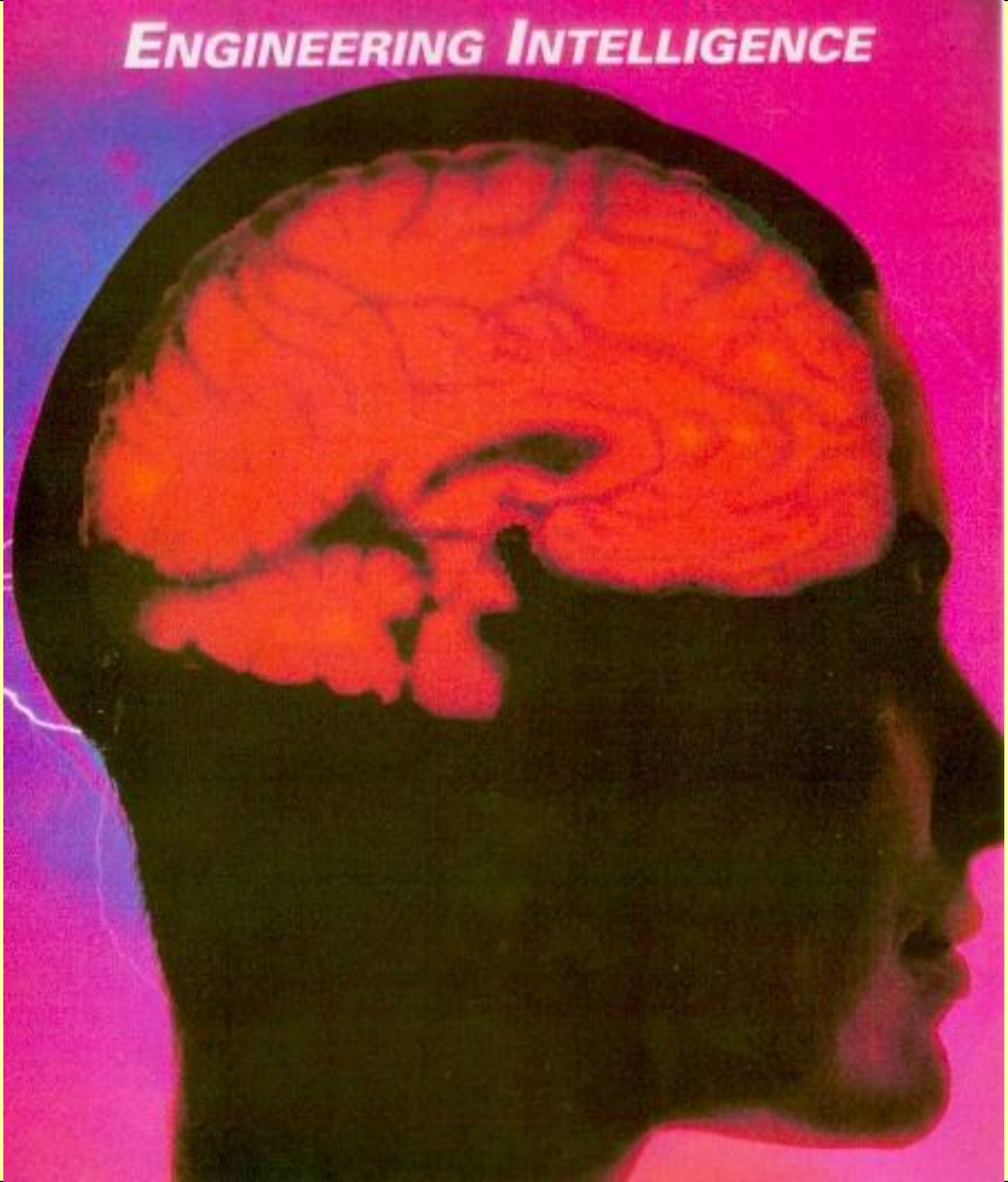
Fuzzy-Neural Networks for Modeling and Intelligent Control

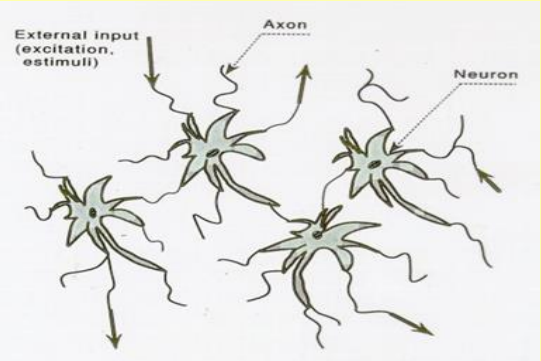
Antonio Moran, Ph.D.

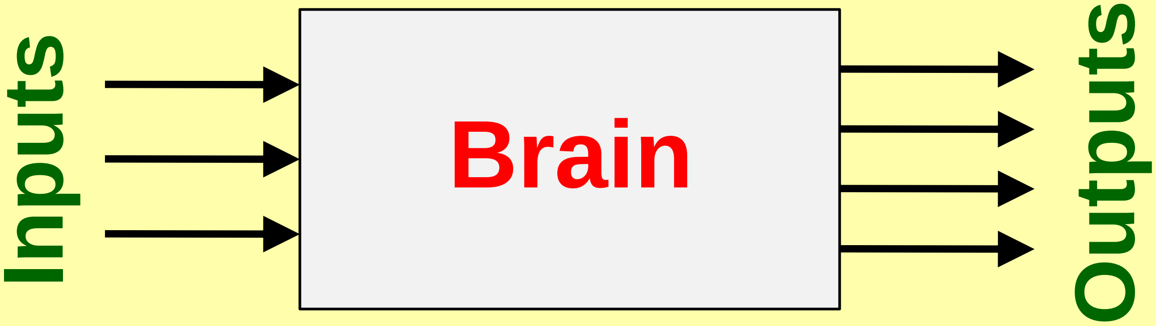
amoran@ieee.org


ENGINEERING INTELLIGENCE

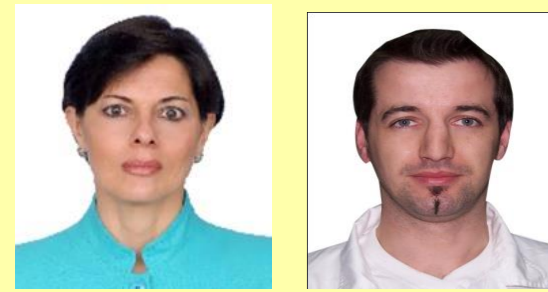
The Brain

Behaves as a System with Inputs and Outputs



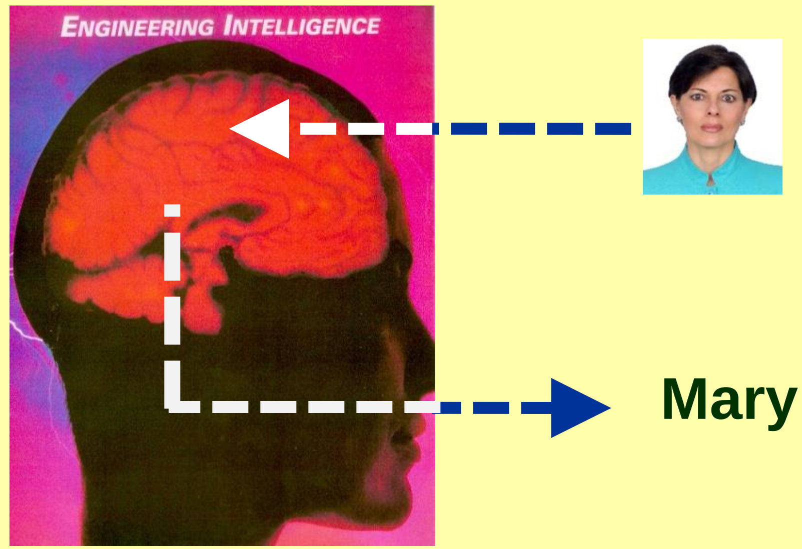


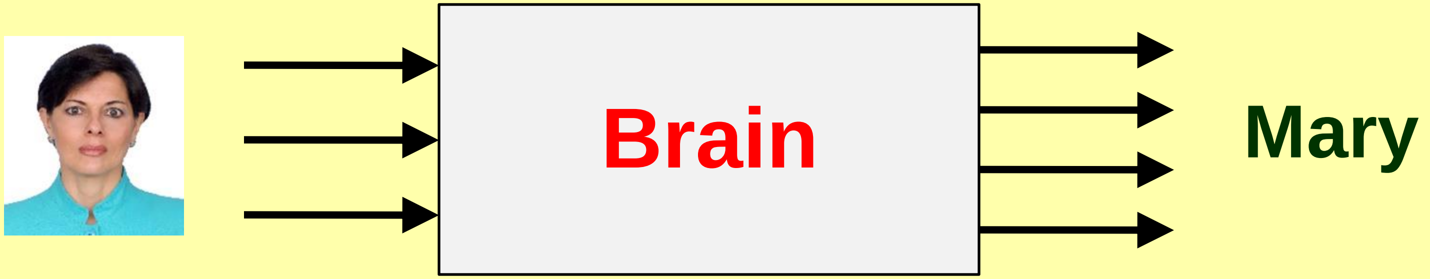
Face Recognition



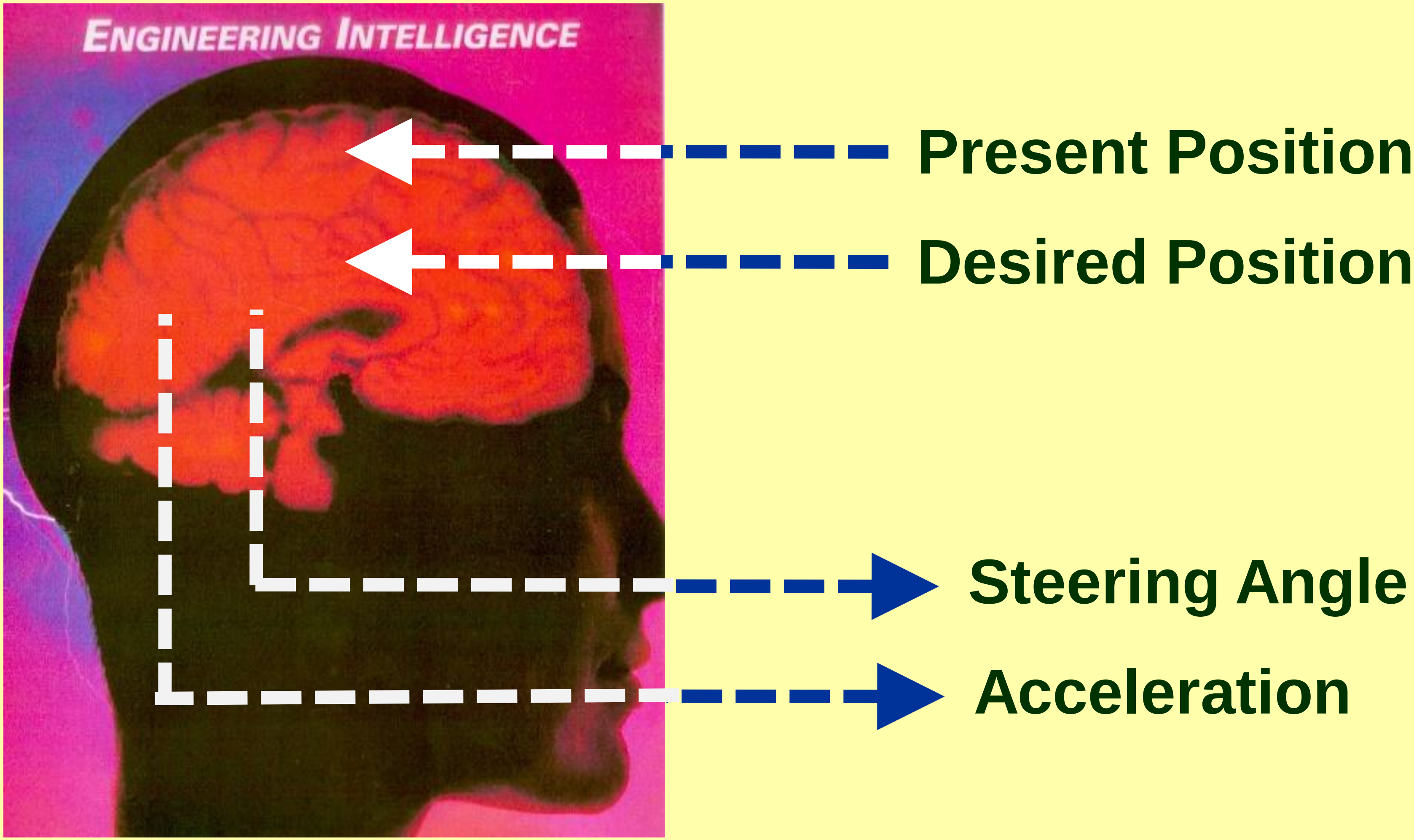
Mary, Peter

Face Recognition



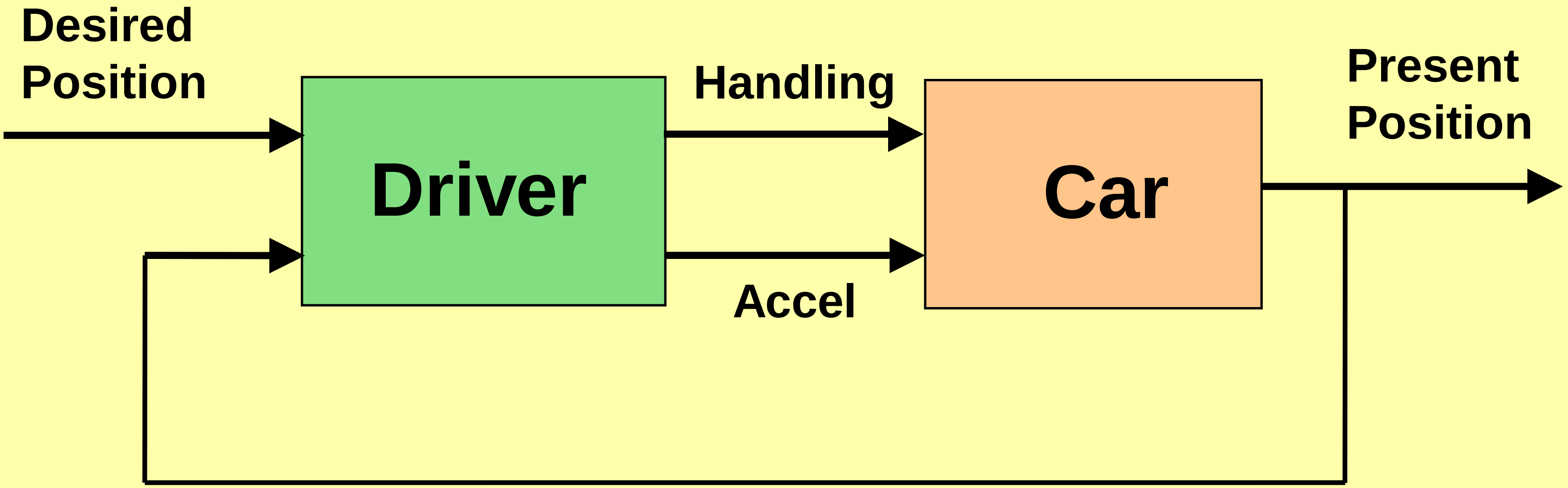


Car Driving


ENGINEERING INTELLIGENCE
Present Position
Desired Position
Steering Angle
Acceleration

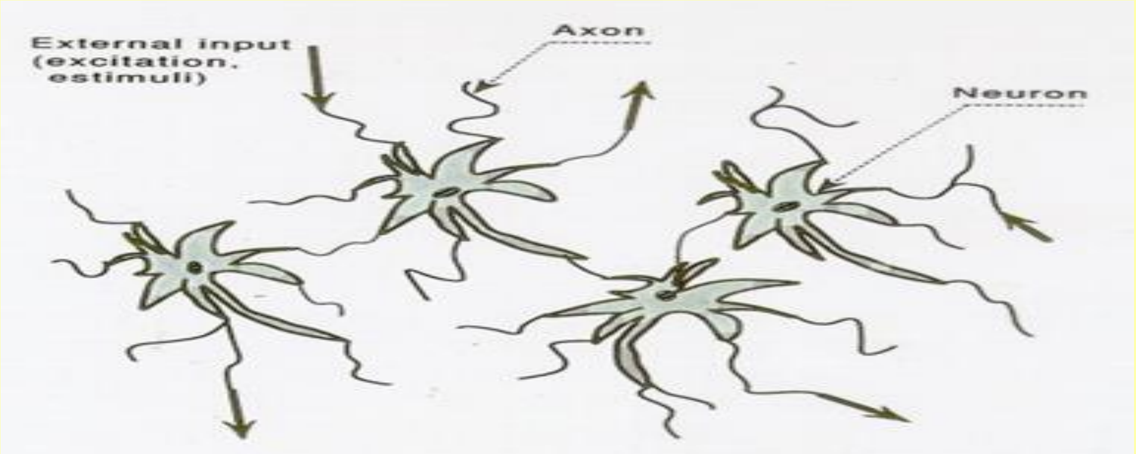
Car Driving

A Control Problem

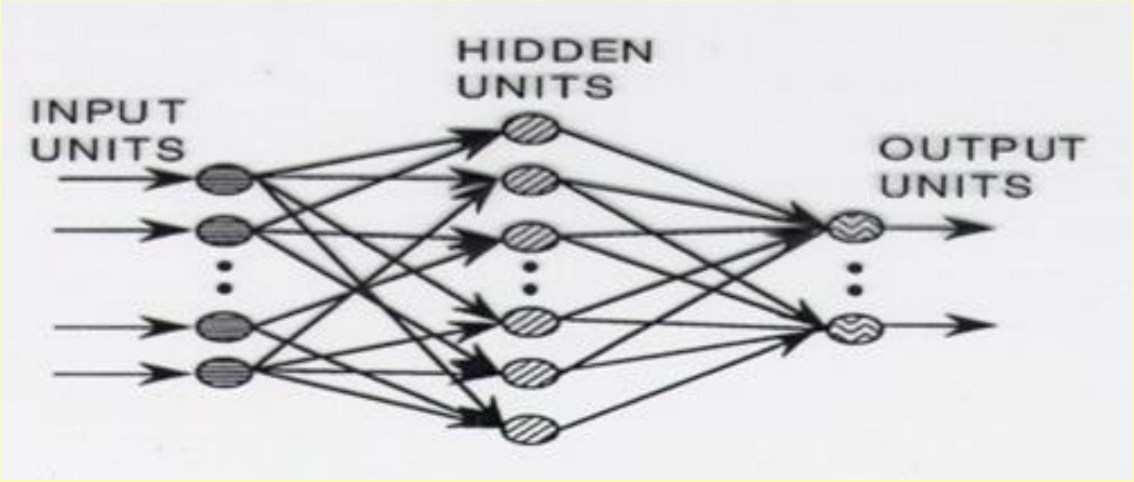


Artificial Neural Network Model

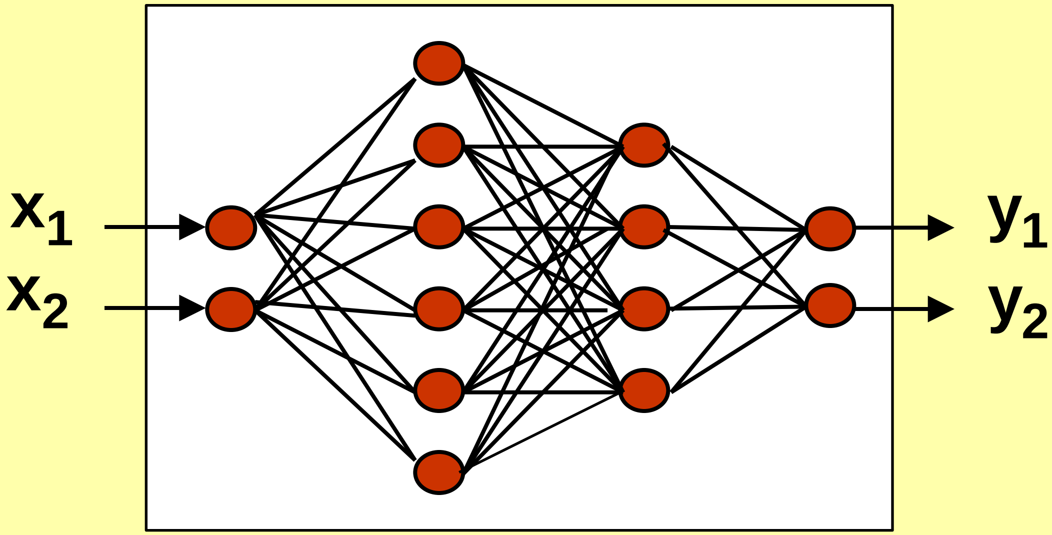
A Natural Neural Network



Multilayer Neural Network Model

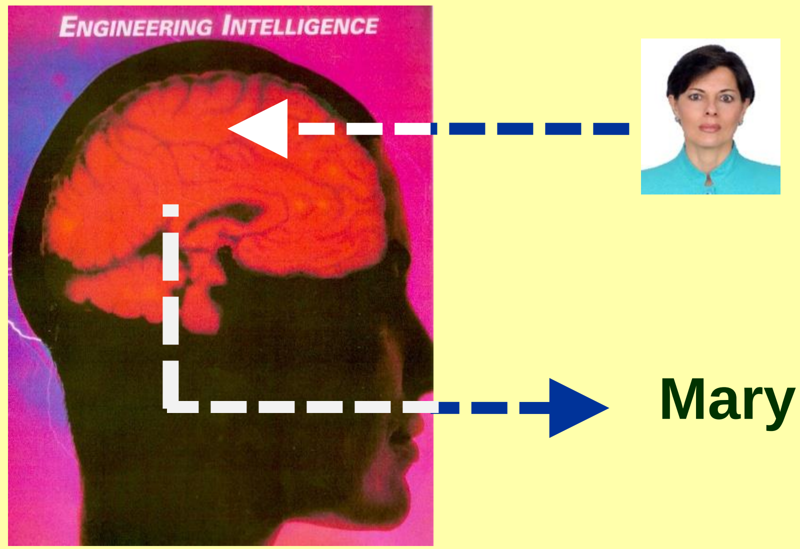


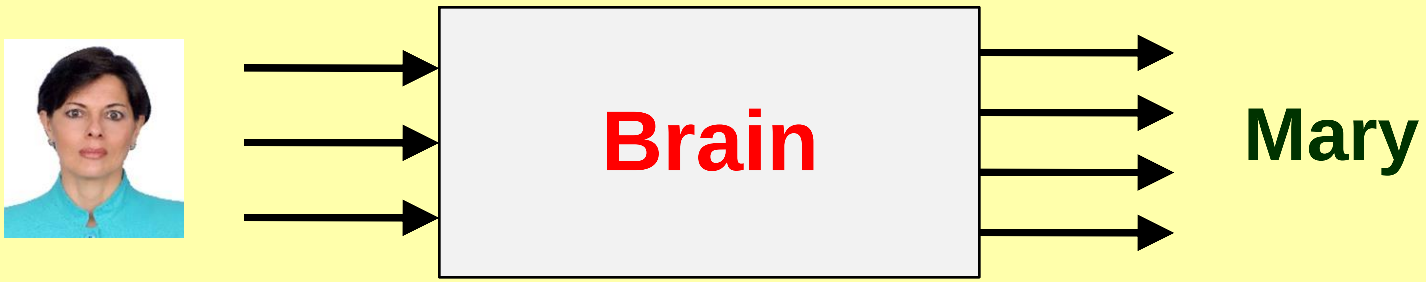
Neural Networks



$$y = \Phi(x)$$

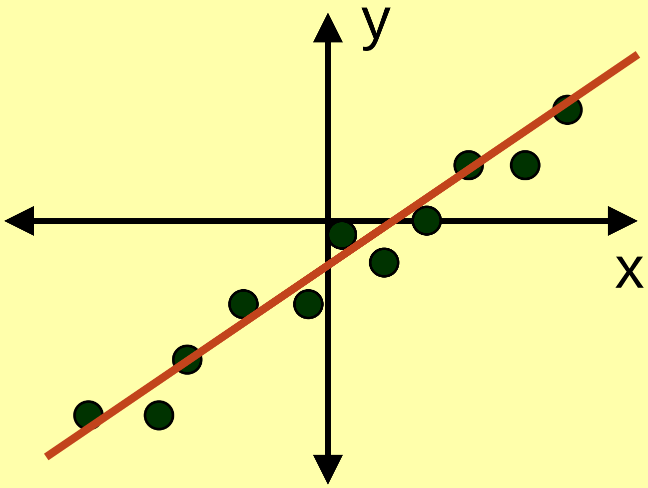
Face Recognition



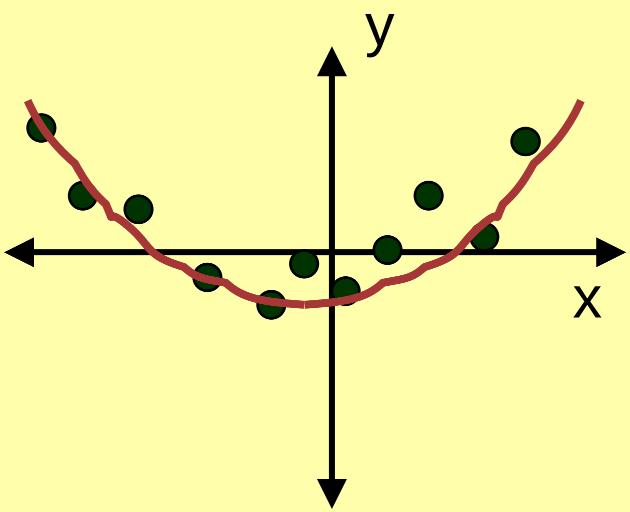


$$y = \Phi(x)$$

Function Estimation

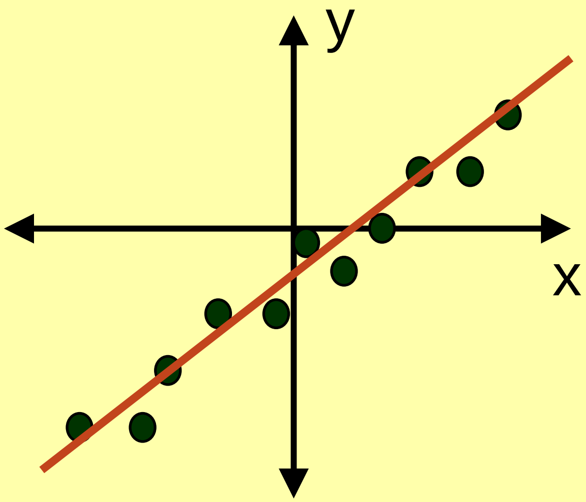


$y = ax + b$



$y = ax^2 + bx + c$

Function Estimation



$$y = ax + b$$

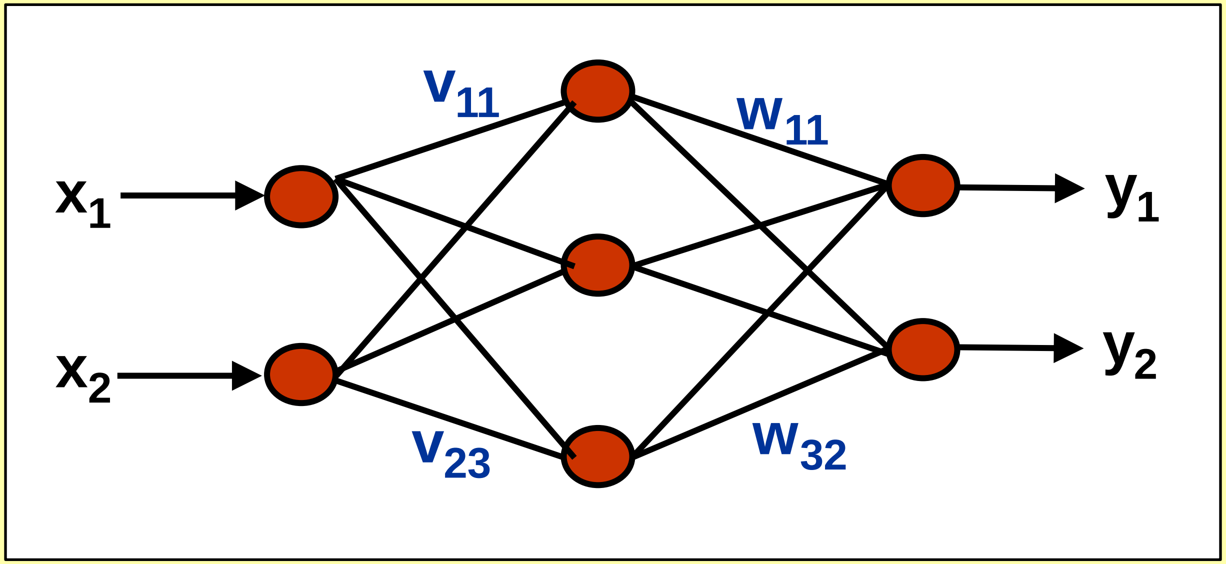
$$J = 0.5\,(y_1-\bar{y}_1)^2 + 0.5\,(y_2-\bar{y}_2)^2 + \cdots\cdots + 0.5\,(y_N-\bar{y}_N)^2$$

Problem: Find a and b that minimize J

Training of Neural Network

Data

x_1	x_2	$\bar{y}_1$	$\bar{y}_2$
*	*	*	*
*	*	*	*
*	*	*	*



Cost function to be minimized:

$$J = 0.5\,(y_{(1)}-\bar{y}_{(1)})^T(y_{(1)}-\bar{y}_{(1)}) + \cdots\cdots + 0.5\,(y_{(N)}-\bar{y}_{(N)})^T\,(y_{(N)}-\bar{y}_{(N)})$$

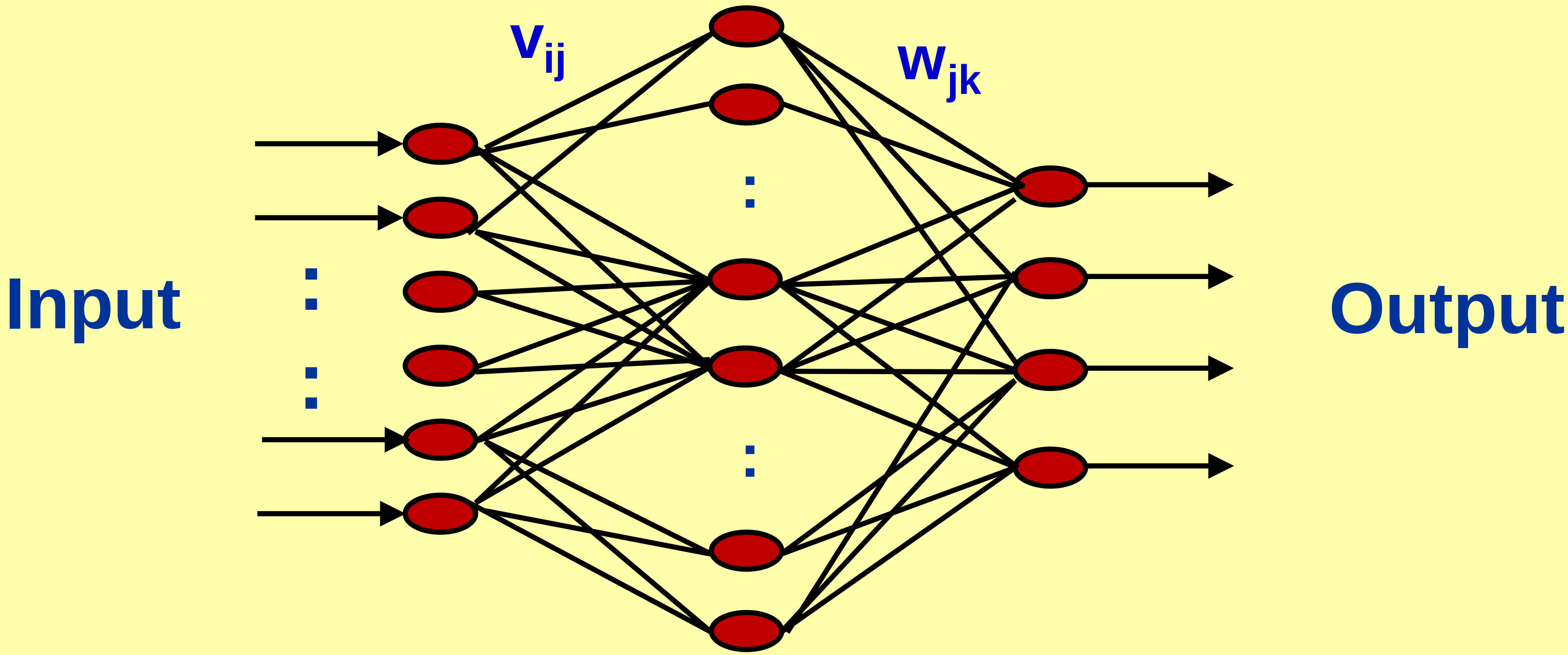
$$y_{(k)} = [\, y_{1(k)} \quad y_{2(k)} \,]^T$$

Problem: Find v_{ij} and w_{jk} that minimize J

Face Recognition

Neural network for recognizing 10 faces

Neural Network for Face Recognition



Input: Face

Output: Code for each face

Face Recognition

Reducing the size of images - Pixeling

Full Color
2808 x 2425

Gray Scale
1826 x 1529
The face occupies the most of the image

Monocromatic
40 x 30
1200 pixels

Neural Network for Face Recognition

Image Preprocessing - Pixeling

1213x1013

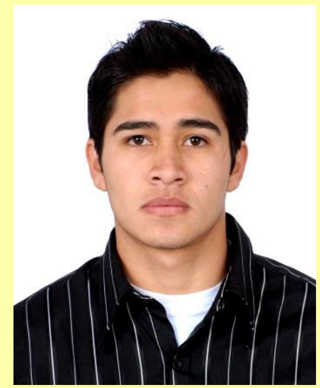
40x30

2644x2106

40x30

2854x2370

40x30



2446x2016

40x30

2507x2190

40x30

Face Recognition

Network Input

Matrix 40x30

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 1 1 1 1 1 1 0 0 0 0 0 0 0

: : : : :

: : : : :

0 0 0 0 1 0 1 1 1 1 1 1 1 0 0 1 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

The matrix should be transformed into vector

Face Recognition

Network Input: Converting 40x30 matrix into 1200x1 vector

$$\left|\begin{array}{cccccccccccccccccccc}
0&0&0&0&0&0&0&0&0&1&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&1&1&1&1&1&1&0&0&0&0&0&0&0\\
\vdots&&&\vdots&&&&&\vdots&&&\vdots&&&&&\vdots&&&\\
0&0&0&0&1&0&1&1&1&1&1&1&1&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{array}\right| \Rightarrow \left|\begin{array}{c}0\\0\\0\\0\\\vdots\\1\\0\\1\\0\\\vdots\\1\\0\\0\\0\end{array}\right|$$

40x30

1200x1

Face Recognition

Network Output

A code assigned to each of the ten faces

(Orthogonal codes)


1 0 0 0 0 0 0 0 0 0

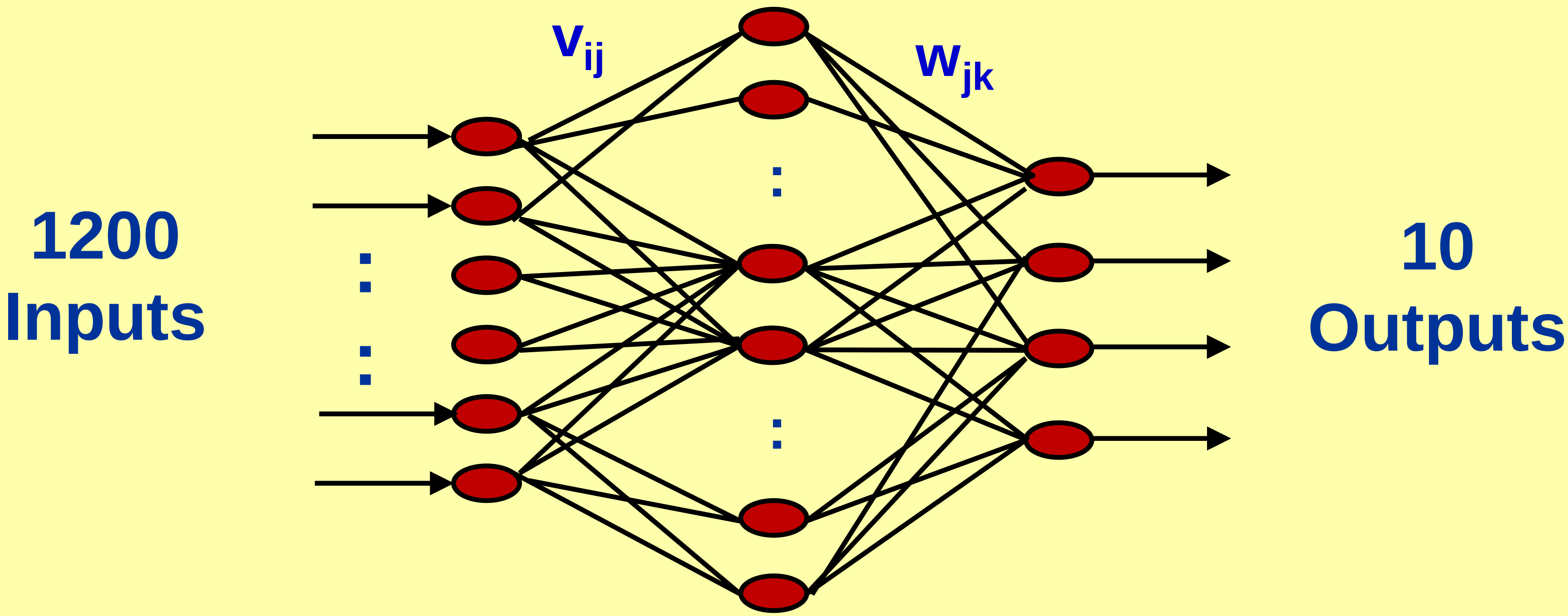

0 1 0 0 0 0 0 0 0 0


0 0 1 0 0 0 0 0 0 0


0 0 0 0 0 0 0 0 1 0

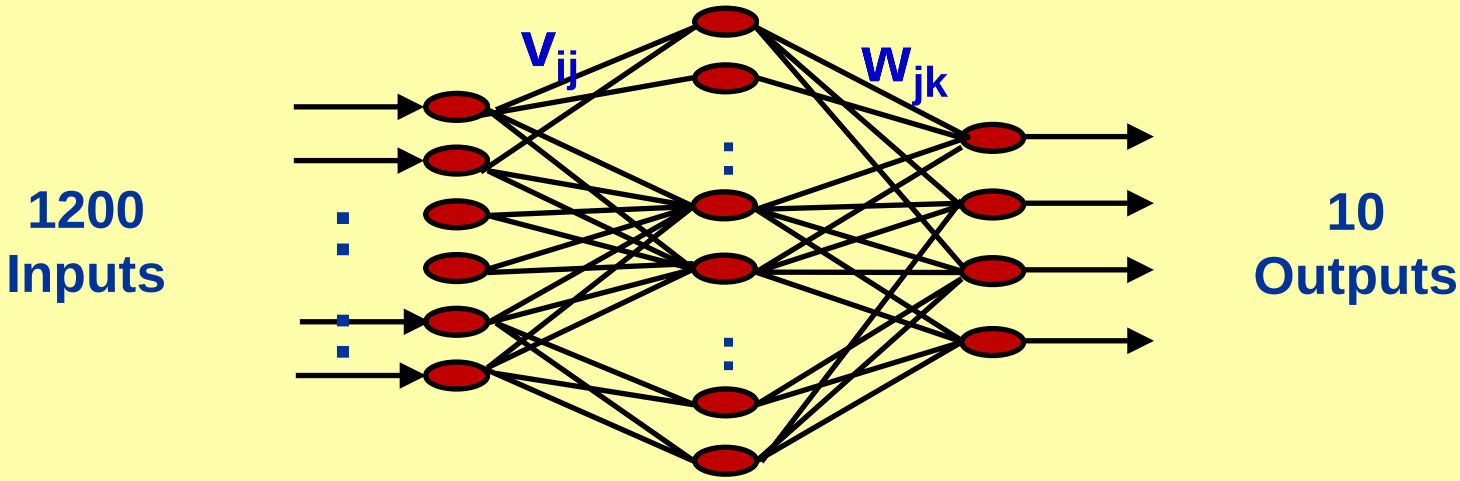

0 0 0 0 0 0 0 0 0 1

Neural Network for Face Recognition



To generate input-output training data, several faces of a person could be considered but all of them with the same output code

Neural Network for Face Recognition



Training: Five faces of the same person

Validation after training: Different faces of each person

Validación de la Red Neuronal

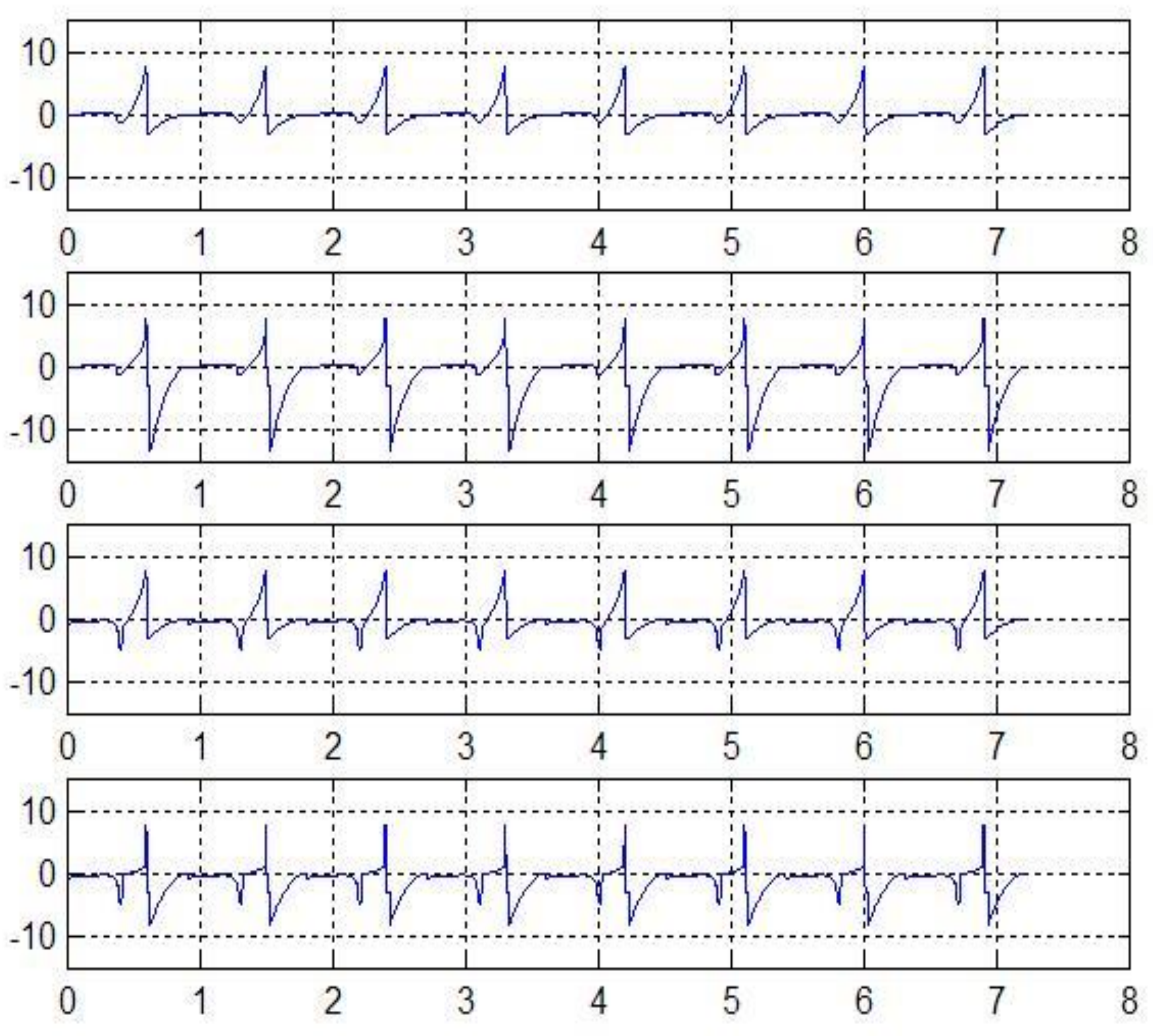
Cara de entrenamiento

0
0
1 ←
0
0
0
0
0
0
0

Cara de validación

0.1
0.1
0.9 ←
0.1
0.3
0.1
0.2
0.1
0.3
0.1

Detection of Cardiac Anomalies



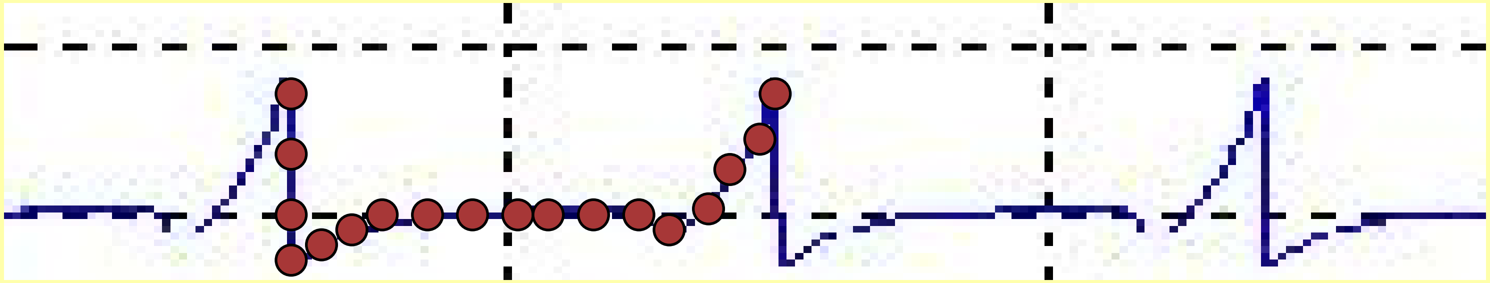
Normal

Anomaly 1

Anomaly 2

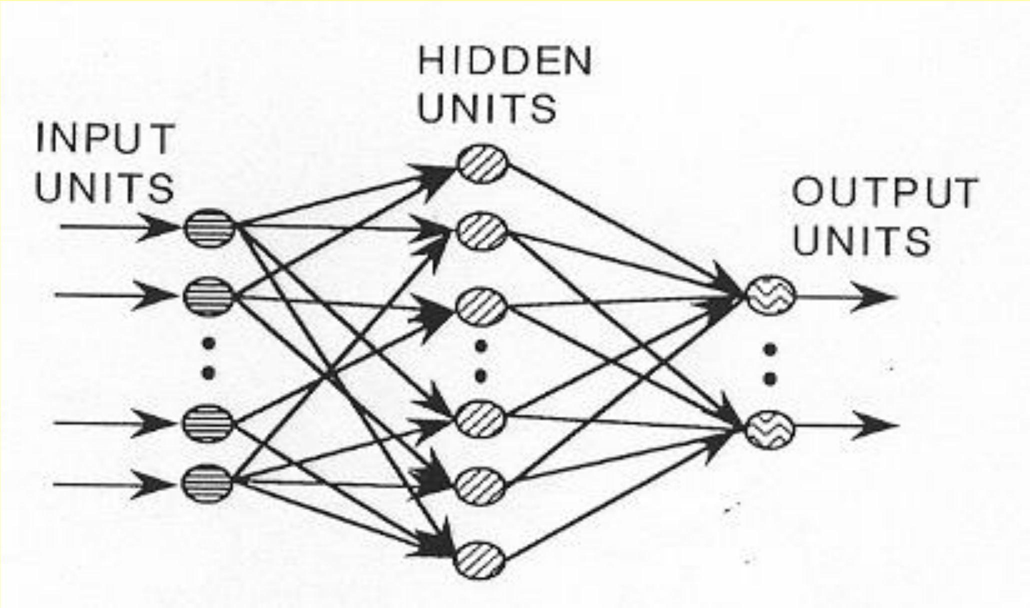
Anomaly 3

Training of Neural Network



600 samples in a period

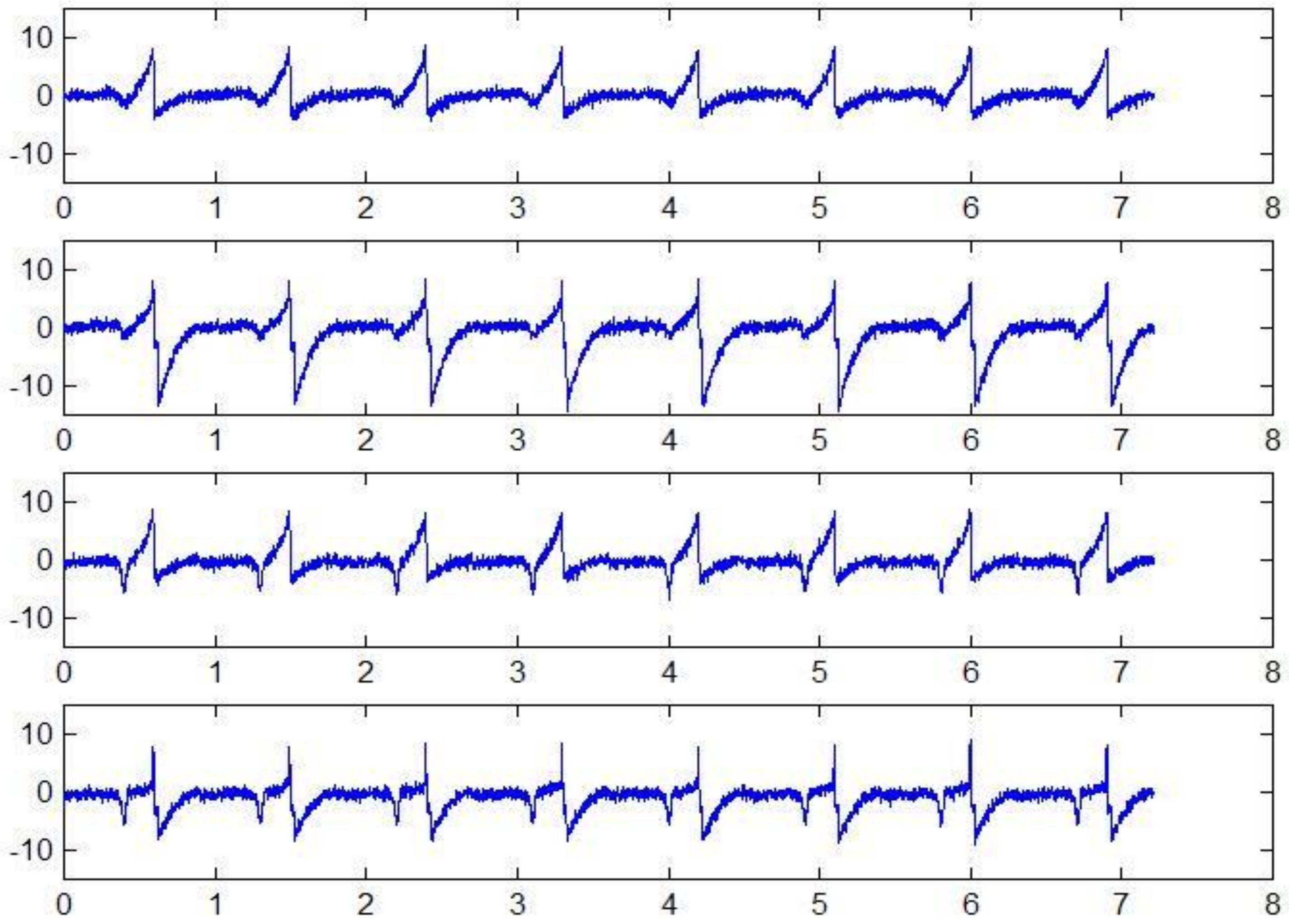
600
Inputs



1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1

4 Outputs

Validation with Noisy Signals

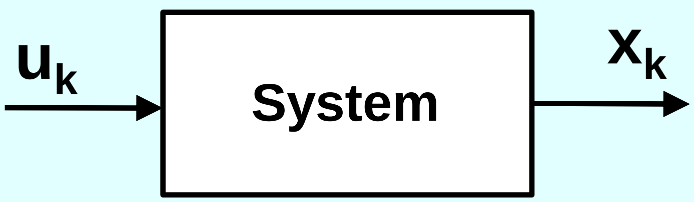


Dynamic Neural Networks

Modeling of Dynamical Systems

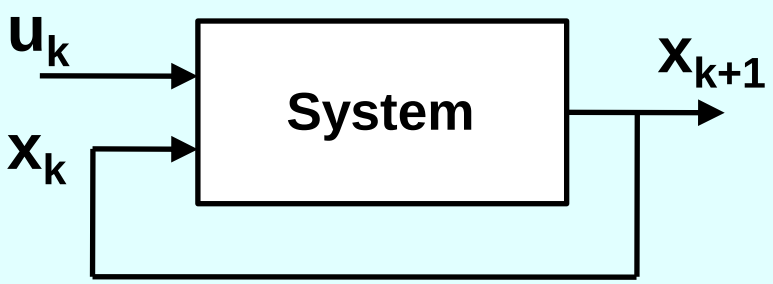
Modeling of Dynamical Systems

Static System



$$x_k = \Phi(u_k)$$

Dynamic System

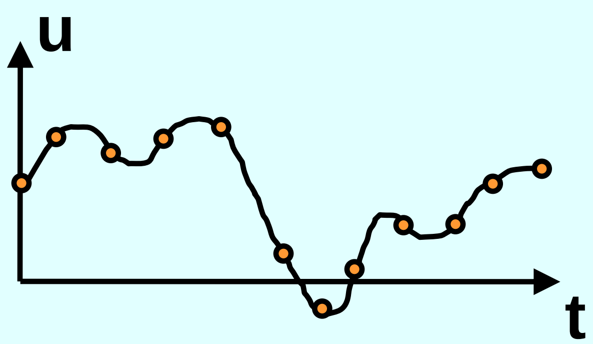


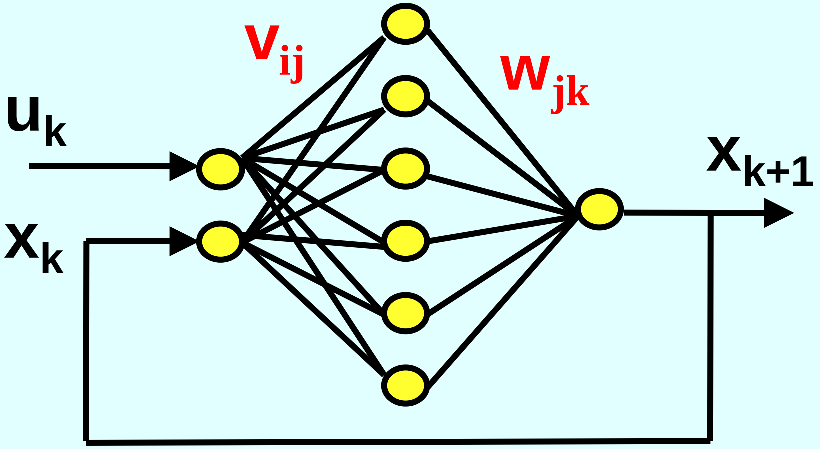
$$x_{k+1} = \Phi(x_k, u_k)$$

Output becomes input in the next step

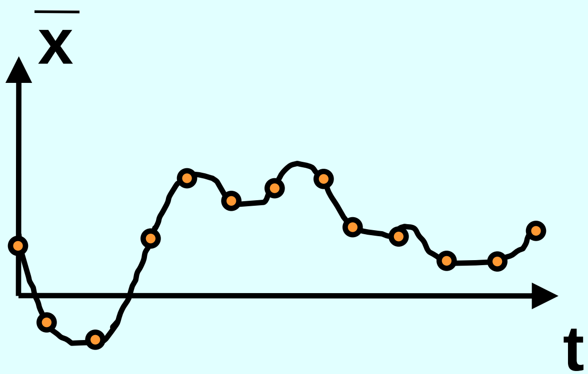
Modeling of Dynamical Systems

Input u





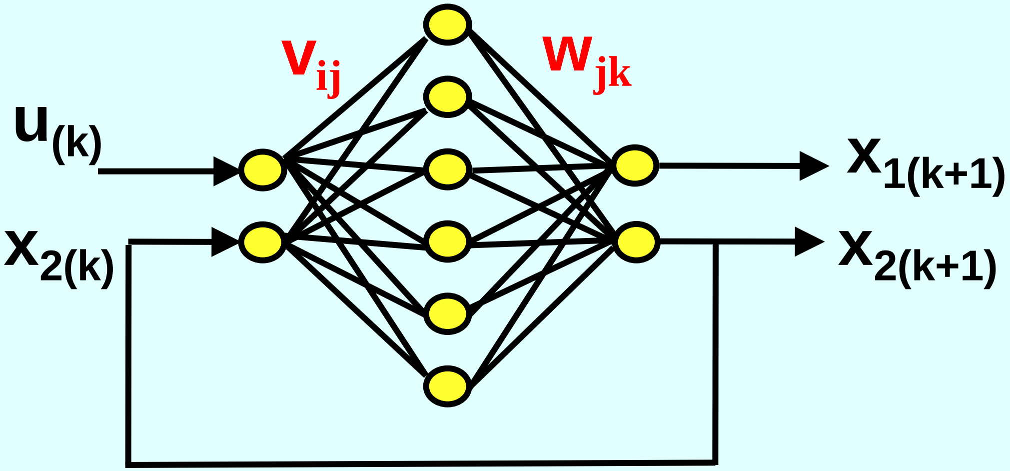
Desired Ouput $\bar{x}$



$$x_{k+1} = \Phi(x_k, u_k)$$

$$x_0\,,\ u_0 \longrightarrow x_1$$
$$x_1\,,\ u_1 \longrightarrow x_2$$
$$x_2\,,\ u_2 \longrightarrow x_3$$
$$\vdots \qquad \vdots$$
$$x_N\,,\ u_N \longrightarrow x_N$$

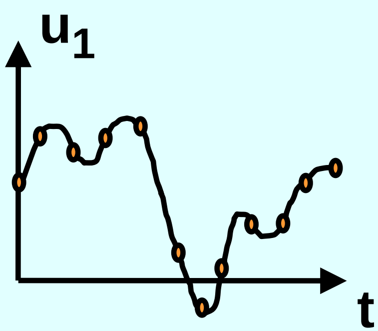
Modeling of Dynamical Systems

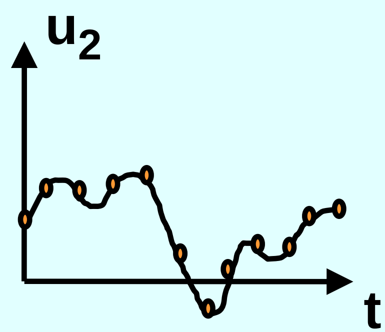


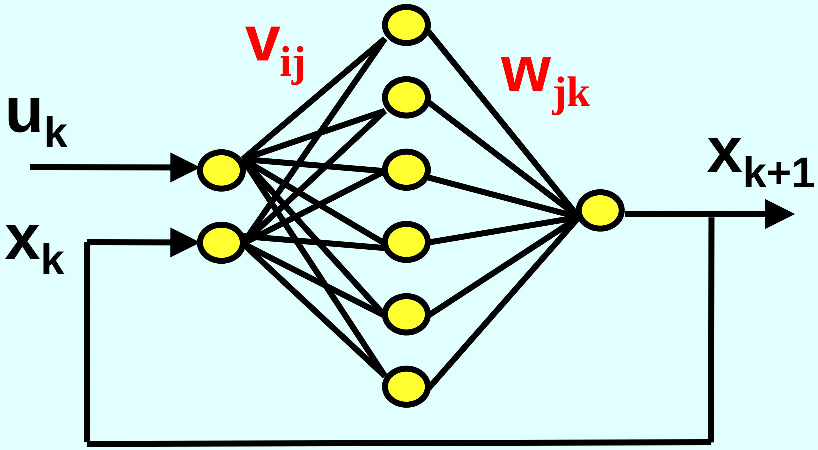
$$x_{k+1} = \Phi(x_k, u_k)$$

Modeling of Dynamical Systems

Input $u = \begin{vmatrix} u_1 \\ u_2 \end{vmatrix}$

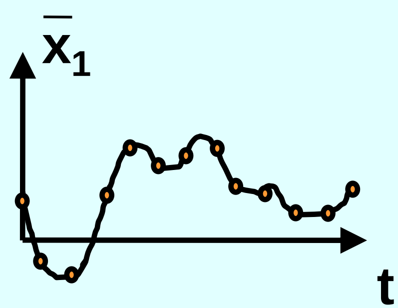


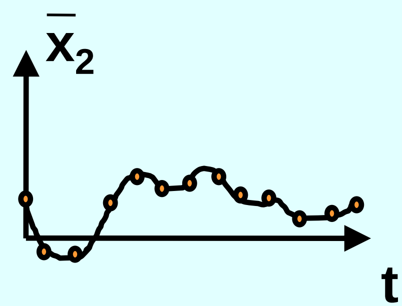




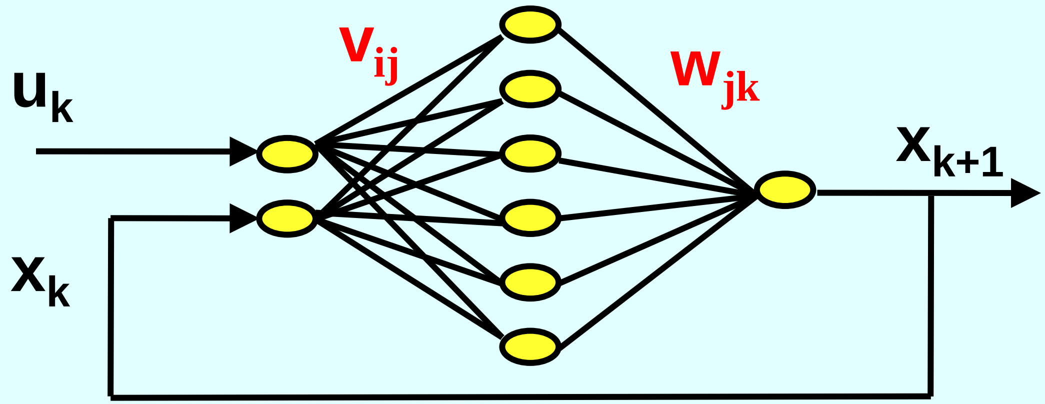
$$x_{k+1} = \Phi(x_k, u_k)$$

Desired Ouput $\overline{x} = \begin{vmatrix} \overline{x}_1 \\ \overline{x}_2 \end{vmatrix}$





Training of Dynamical Neural Networks



Cost Function to be Minimized

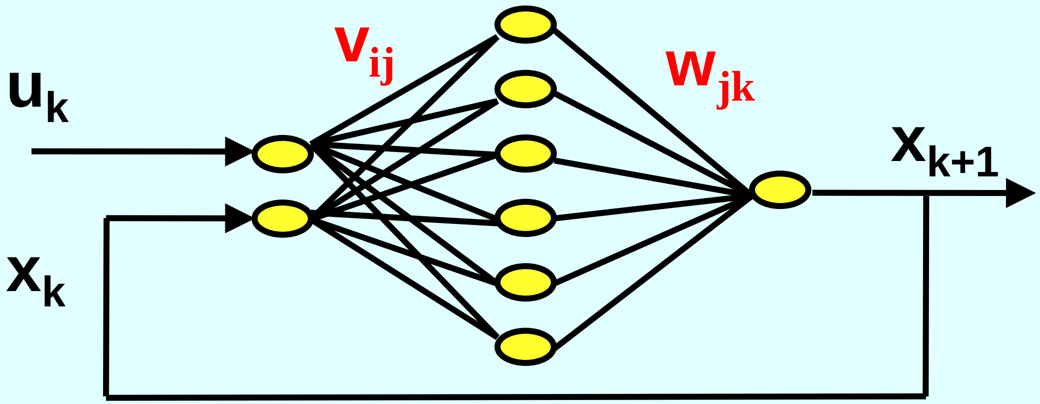
$$J = 0.5\,(x_1-\bar{x}_1)^2 + 0.5\,(x_2-\bar{x}_2)^2 + \cdots + 0.5\,(x_N-\bar{x}_N)^2$$

$$J = 0.5 \sum_{k=1}^{k=N} (x_k - \bar{x}_k)^2$$

$x_k \rightarrow$ Estado (Salida) de la red

$\bar{x}_k \rightarrow$ Salida deseada (data)

Training of Dynamical Neural Networks

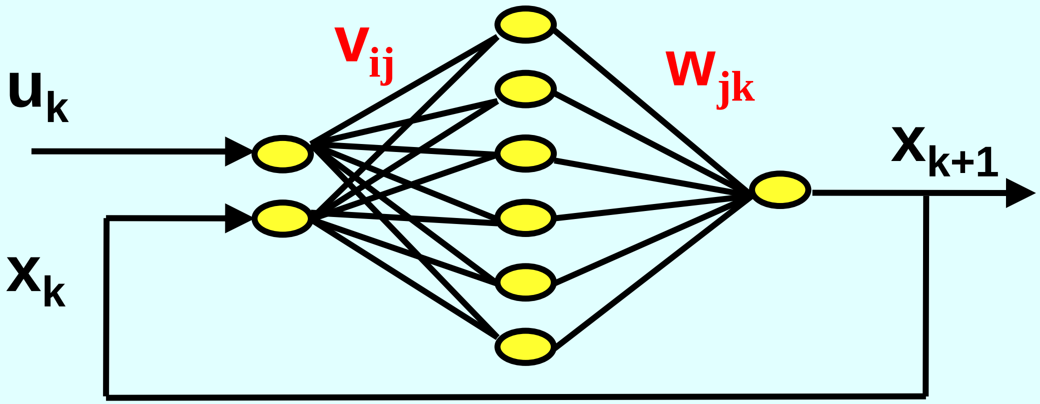


If x is a vector $\quad x = \begin{vmatrix} x_1 \\ x_2 \end{vmatrix}$

Cost Function to be Minimized

$$J = 0.5 \sum_{k=1}^{k=N} (x_k - \overline{x}_k)^T (x_k - \overline{x}_k)$$

Training of Dynamical Neural Networks



Cost Function to be Minimized

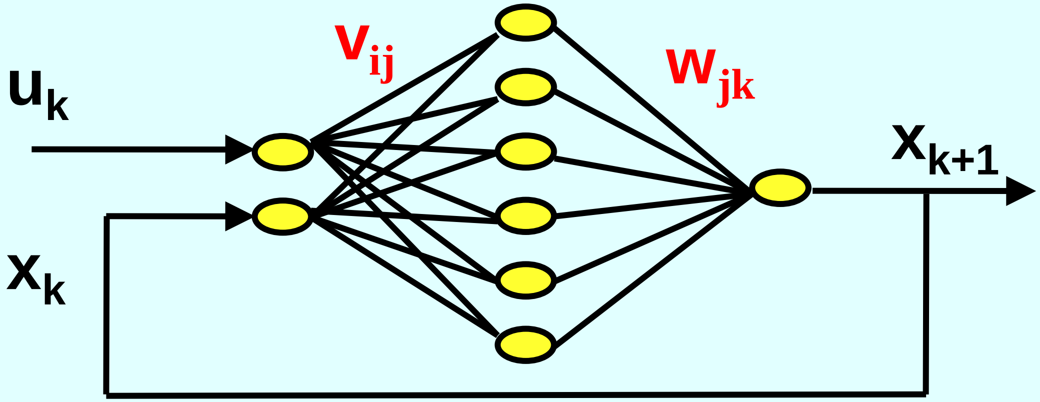
$$J = 0.5\,(x_1 - \bar{x}_1)^2 + 0.5\,(x_2 - \bar{x}_2)^2 + \cdots\cdots + 0.5\,(x_N - \bar{x}_N)^2$$

$$v_{ij} = v_{ij} - \eta\,\frac{\overline{\partial J}}{\overline{\partial v}_{ij}}$$

$$w_{jk} = w_{jk} - \eta\,\frac{\overline{\partial J}}{\overline{\partial w}_{jk}}$$

Total partial derivatives

Training of Dynamical Neural Networks



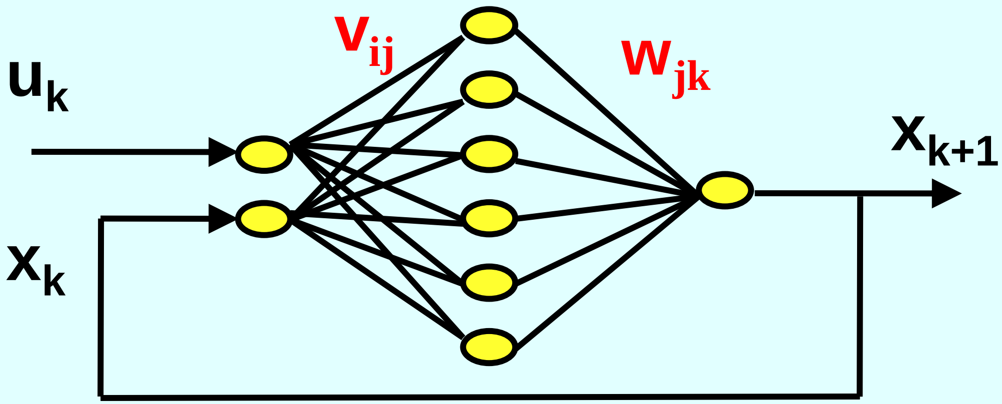
Cost Function to be Minimized $J = 0.5 \sum_{k=1}^{k=N} (x_k - \bar{x}_k)^T (x_k - x_k)$

$$\frac{\overline{\partial J}}{\overline{\partial v}} = \sum_{k=1}^{k=N} (x_k - \bar{x}_k)^T \frac{\overline{\partial x_k}}{\overline{\partial v}}$$

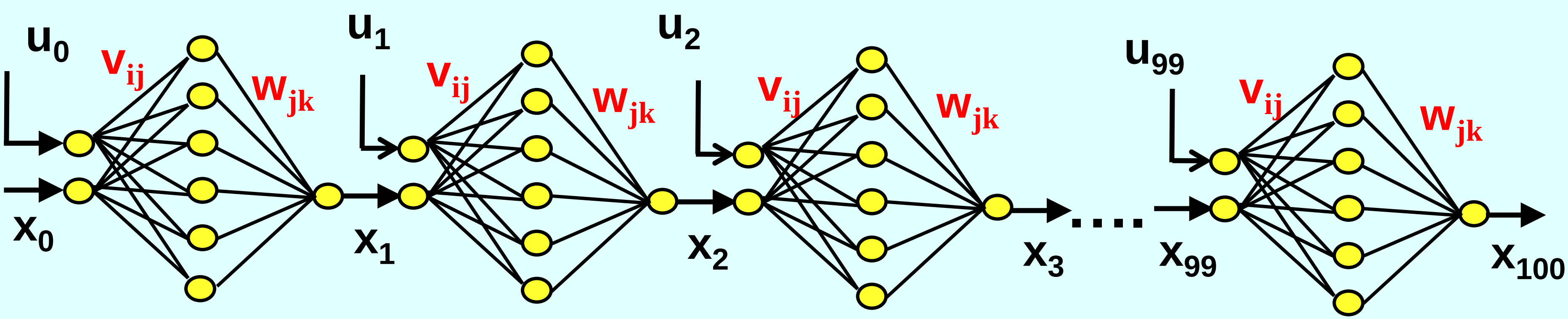
$$\frac{\overline{\partial J}}{\overline{\partial w}} = \sum_{k=1}^{k=N} (x_k - \bar{x}_k)^T \frac{\overline{\partial x_k}}{\overline{\partial w}}$$

Total partial derivative of x_k

Training of Dynamic Neural Networks



Unfolding the Network Along Time



Training of Dynamical Neural Networks

$$v_{ij} = v_{ij} - \eta \frac{\overline{\partial J}}{\overline{\partial v_{ij}}}$$

$$w_{jk} = w_{jk} - \eta \frac{\overline{\partial J}}{\overline{\partial w_{jk}}}$$

Total partial derivatives

$z = 3y + 2x$

$y = 4x + 5r$

$r = 2x + 6s$

Simple Derivative

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial x} = 2$$

Total Derivative

$$\frac{\overline{\partial z}}{\overline{\partial x}} = \frac{\partial z}{\partial x} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial x} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial r}\frac{\partial r}{\partial x}$$

Training of Dynamical Neural Networks

Computation of Total Partial Derivatives

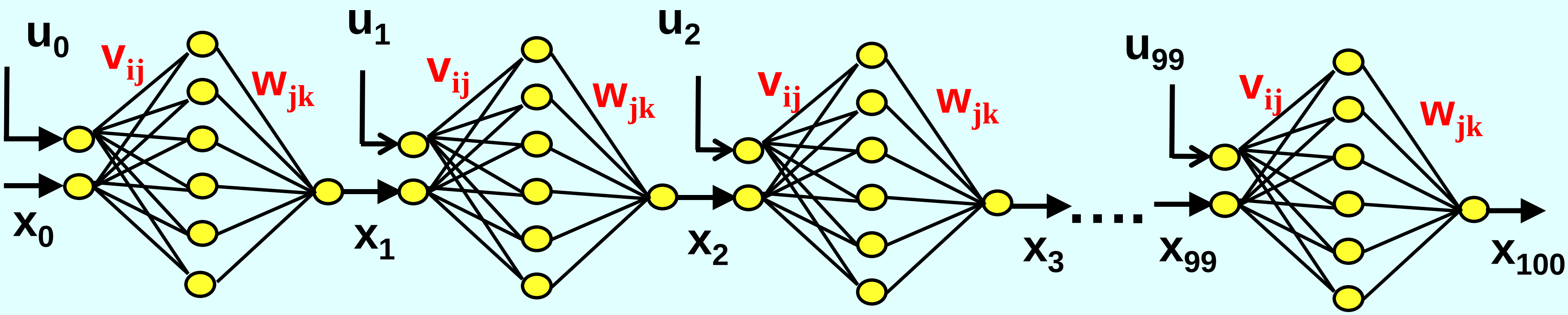
- **Back Propagation Through Time BPTT**

  Paul Werbos, 1972

- **Dynamic Back Propagation DBP**

  Kumpati Narendra, 1989

Dynamic Back Propagation

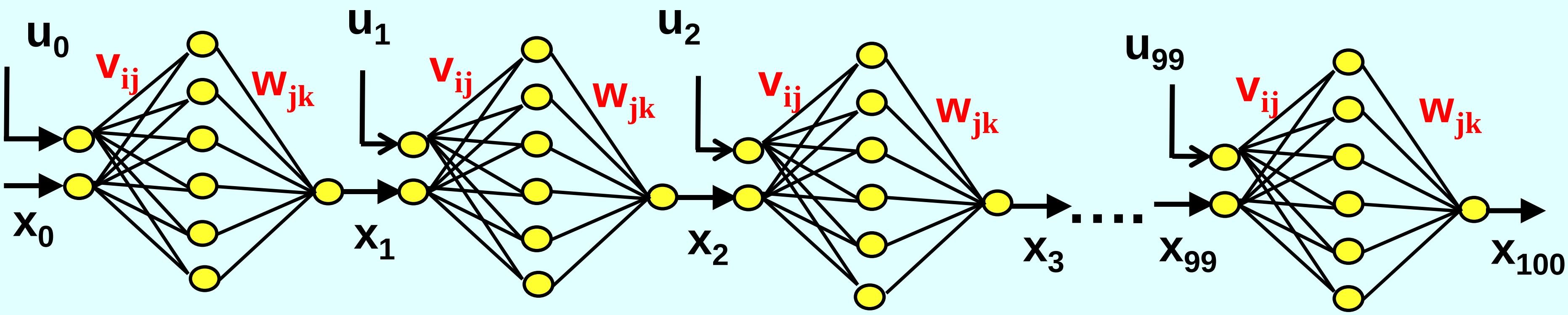


$$\frac{\overline{\partial x_1}}{\overline{\partial v}} = \frac{\partial x_1}{\partial v}$$

$$\frac{\overline{\partial x_2}}{\overline{\partial v}} = \frac{\partial x_2}{\partial v} + \frac{\partial x_2}{\partial x_1}\frac{\overline{\partial x_1}}{\overline{\partial v}}$$

$$\frac{\overline{\partial x_3}}{\overline{\partial v}} = \frac{\partial x_3}{\partial v} + \frac{\partial x_3}{\partial x_2}\frac{\overline{\partial x_2}}{\overline{\partial v}}$$

Dynamic Back Propagation



$$\frac{\overline{\partial x_{k+1}}}{\overline{\partial v}} = \frac{\partial x_{k+1}}{\partial v} + \frac{\partial x_{k+1}}{\partial x_k}\,\frac{\overline{\partial x_k}}{\overline{\partial v}}$$

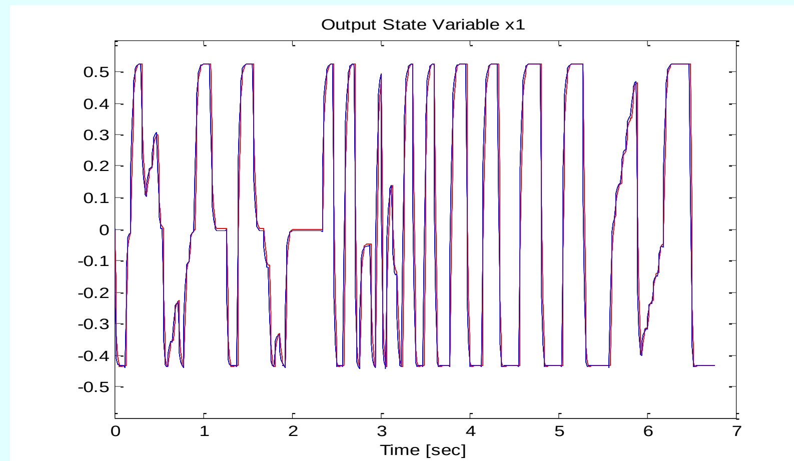
Recursive expression for computation of total partial derivatives

Modeling of Nonlinear Dynamic System

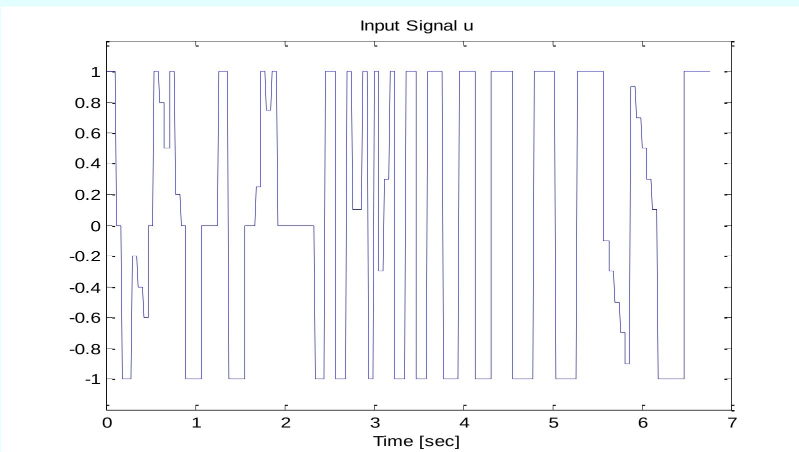
One Input and Two Outputs

Network Training

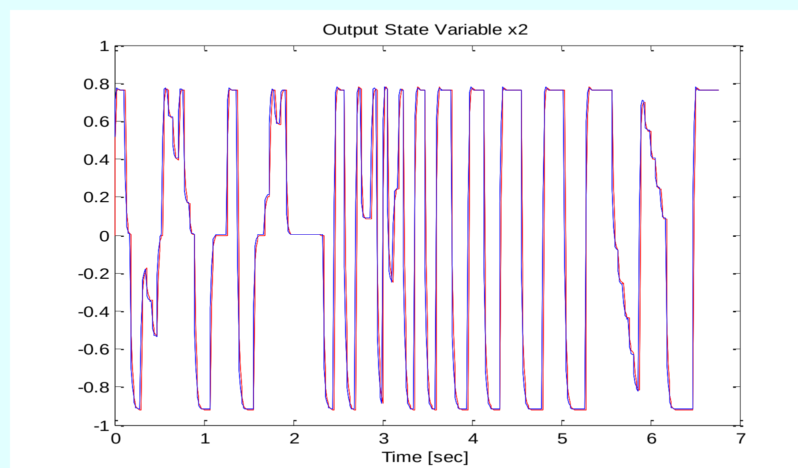
Output Signal x1



Input Signal u



Output Signal x2



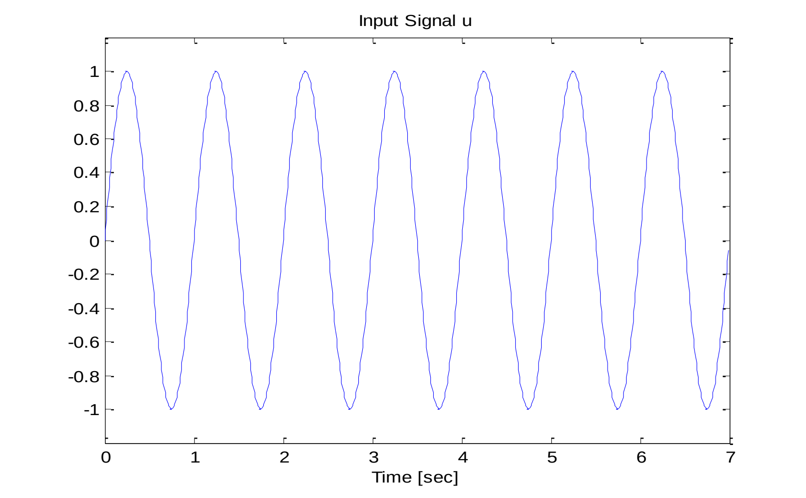
Training Signal

Model Output

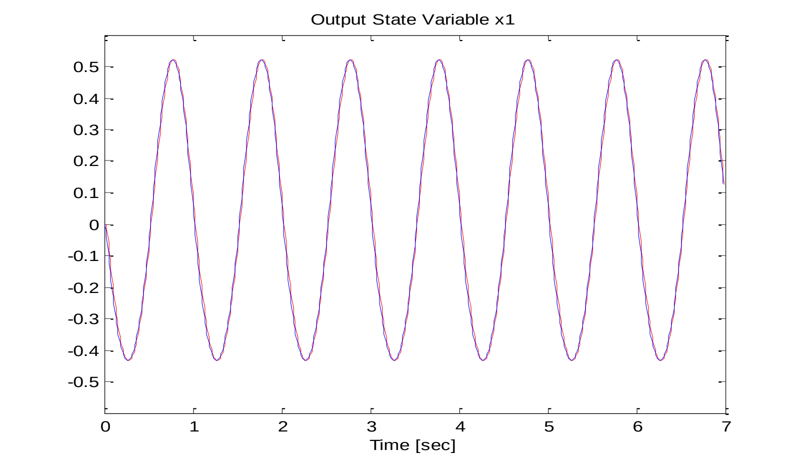
Modeling of Nonlinear Dynamic System

Validation: Input-Output Signals

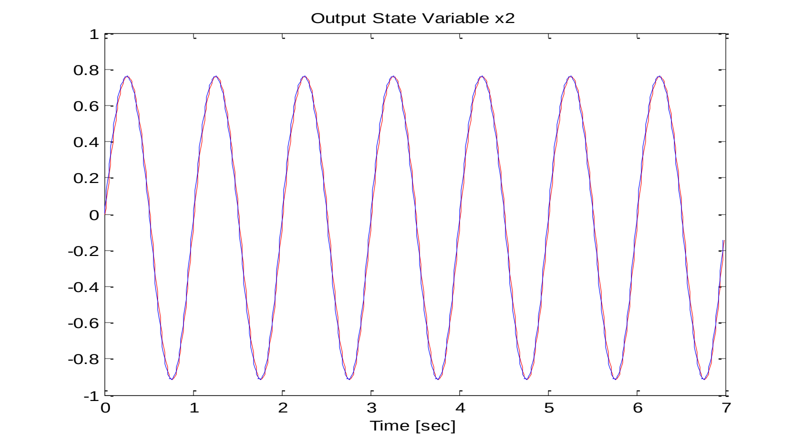
Input Signal u



Output Signal x1



Output Signal x2



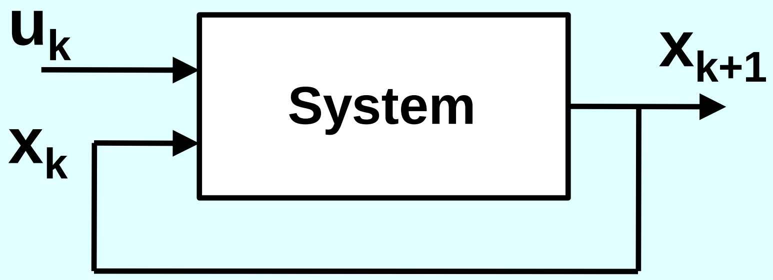
Training Signal

Model Output

Modeling of Nonlinear Dynamic System

Matlab Simulation

Dynamical system with 1 input and 3 outputs



$$x = \begin{vmatrix} x_1 \\ x_2 \\ x_3 \end{vmatrix}$$

Nonlinear system

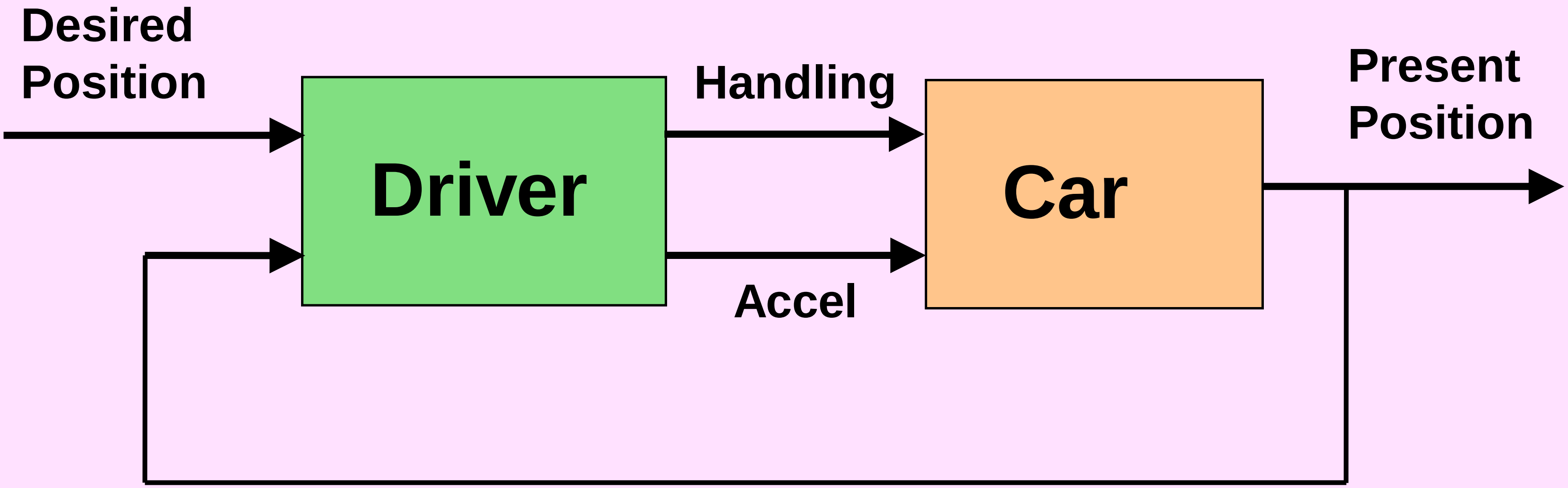
$$x_k = Ax_k + Bu_k + Gx_k u_k$$

Dynamic Neural Networks

Control of Dynamical Systems

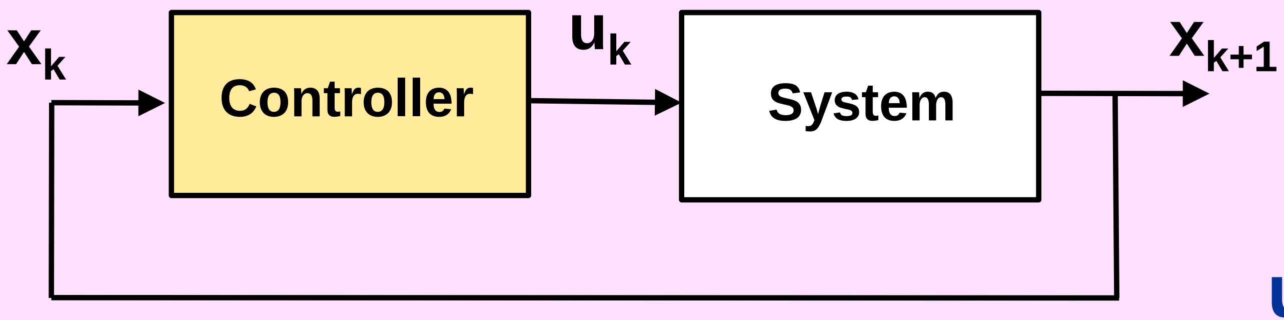
Car Driving

A Control Problem



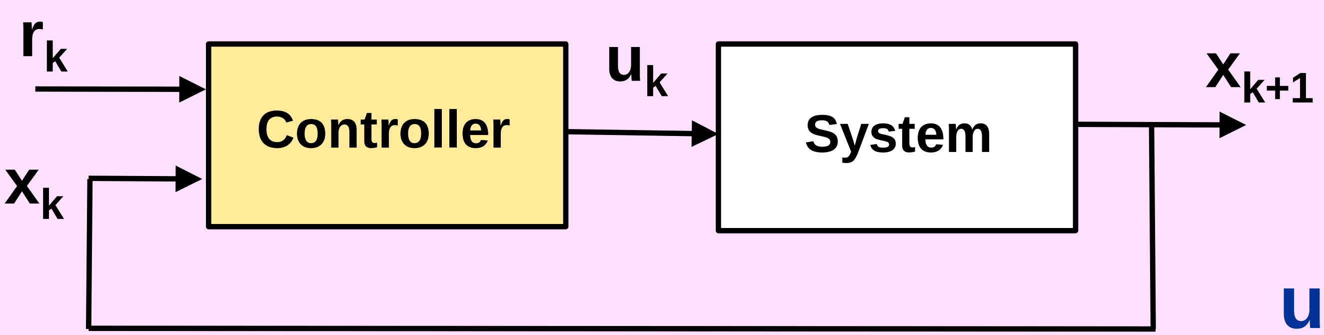
Control of Dynamical Sytems

Stabilization



$$u_k = \Omega(x_k)$$

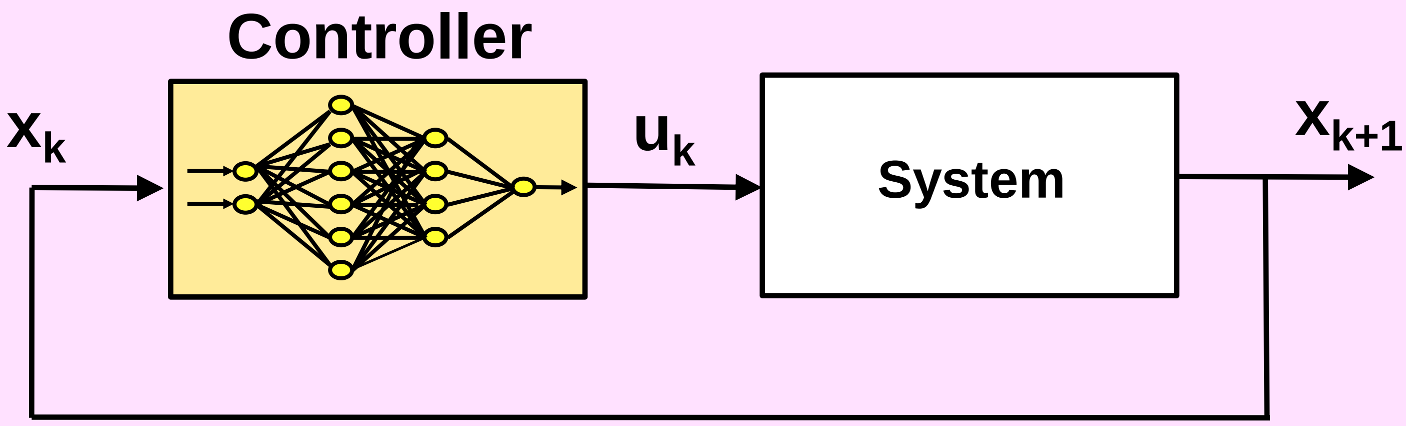
Tracking



$$u_k = \Omega(x_{k,}\, r_k)$$

Control of Dynamical Sytems

Stabilization



Controller

$$u_k = \Omega(x_k)$$

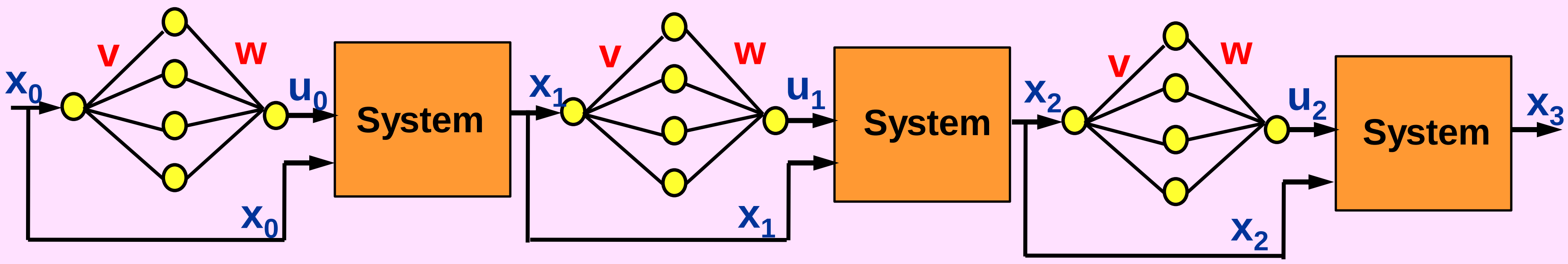
System

$$x_{k+1} = \Phi(x_k, u_k)$$

Represented by:

- Neural Network
- State Equation

Training of Neuro-Controller

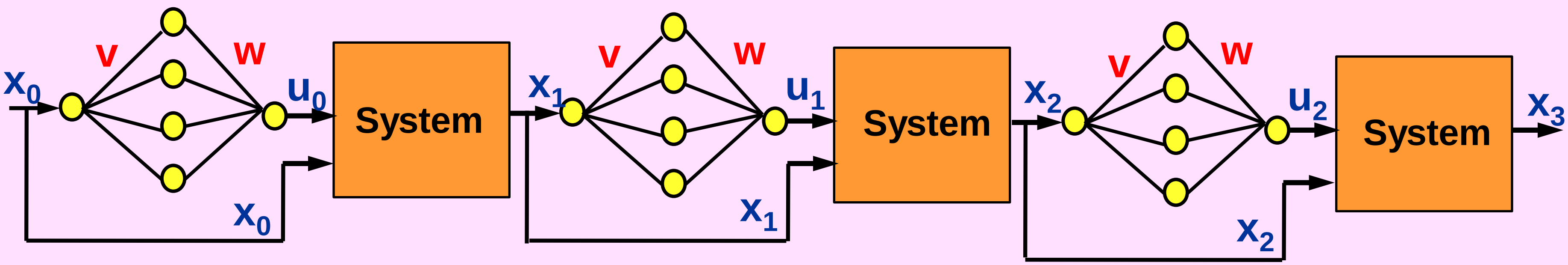


If x is a vector $x = \begin{vmatrix} x_1 \\ x_2 \end{vmatrix}$

Cost Function to be Minimized

$$J = 0.5 \sum_{k=1}^{k=N} (x_k - \overline{x}_k)^T (x_k - x_k)$$

Training of Neuro-Controller



Cost Function to be Minimized

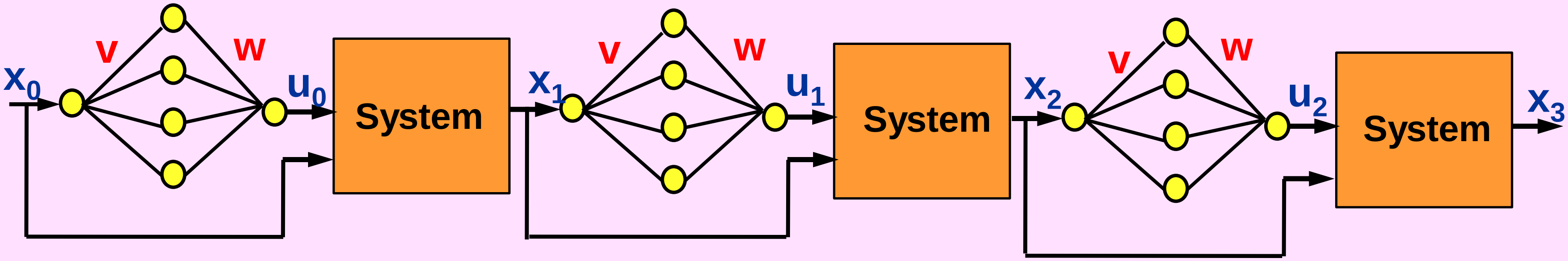
$$J = 0.5\,(x_1-\bar{x}_1)^2 + 0.5\,(x_2-\bar{x}_2)^2 + \cdots\cdots + 0.5\,(x_N-\bar{x}_N)^2$$

$$J = 0.5 \sum_{k=1}^{k=N} (x_k - \bar{x}_k)^2$$

$x_k \rightarrow$ Estado (Salida)

$\bar{x}_k \rightarrow$ Salida deseada

Training of Neuro-Controller



Cost Function to be Minimized

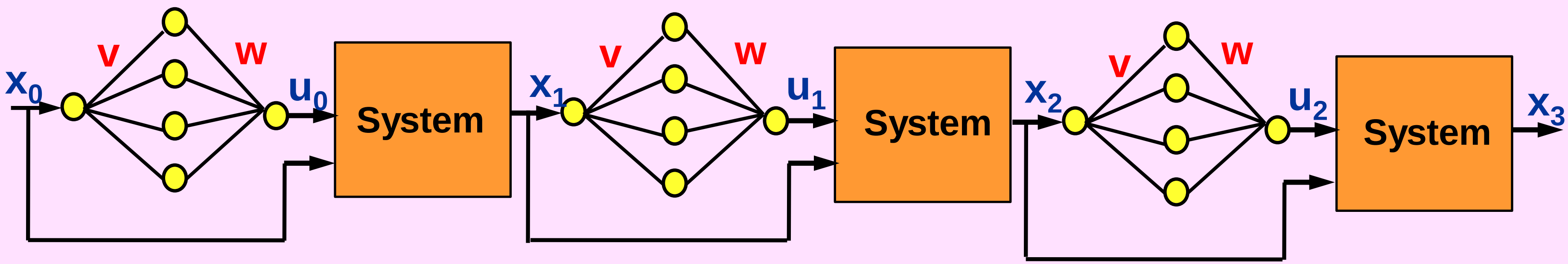
$$J = 0.5\,(x_1-\overline{x}_1)^2 + 0.5\,(x_2-\overline{x}_2)^2 + \cdots\cdots + 0.5\,(x_N-\overline{x}_N)^2$$

$$v_{ij} = v_{ij} - \eta\,\frac{\overline{\partial J}}{\overline{\partial v_{ij}}}$$

$$w_{jk} = w_{jk} - \eta\,\frac{\overline{\partial J}}{\overline{\partial w_{jk}}}$$

Total partial derivatives

Training of Neuro-Controller



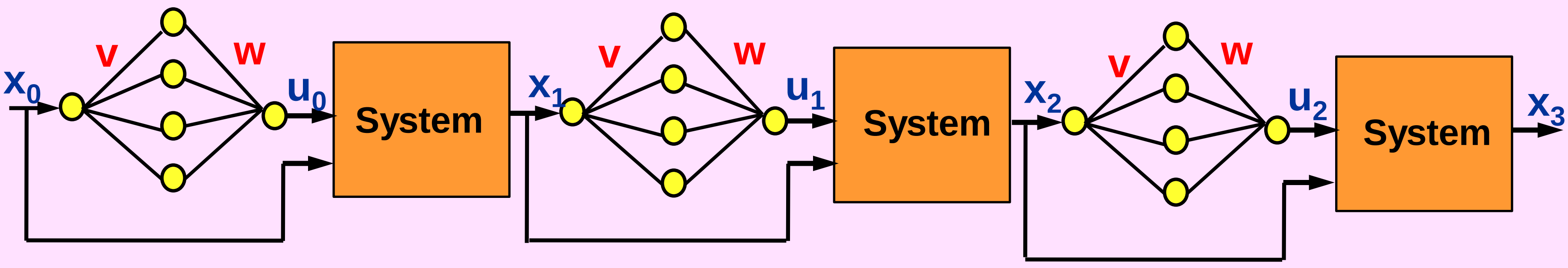
Cost Function to be Minimized $J = 0.5 \sum_{k=1}^{k=N} (x_k - \bar{x}_k)^T (x_k - x_k)$

$$\frac{\overline{\partial J}}{\overline{\partial v}} = \sum_{k=1}^{k=N} (x_k - \bar{x}_k)^T \frac{\overline{\partial x_k}}{\overline{\partial v}}$$

$$\frac{\overline{\partial J}}{\overline{\partial w}} = \sum_{k=1}^{k=N} (x_k - \bar{x}_k)^T \frac{\overline{\partial x_k}}{\overline{\partial w}}$$

Total partial derivative of x_k

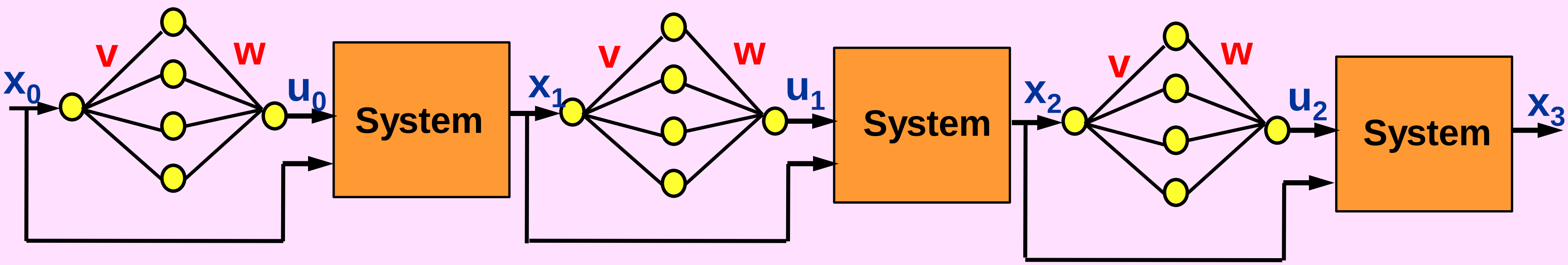
Dynamic Back Propagation



$$\frac{\overline{\partial x_{k+1}}}{\overline{\partial v}} = \frac{\partial x_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial v} + \left(\frac{\partial x_{k+1}}{\partial x_k} + \frac{\partial x_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial x_k}\right)\frac{\overline{\partial x_k}}{\overline{\partial v}}$$

Recursive expression for computation of total partial derivatives

Dynamic Back Propagation

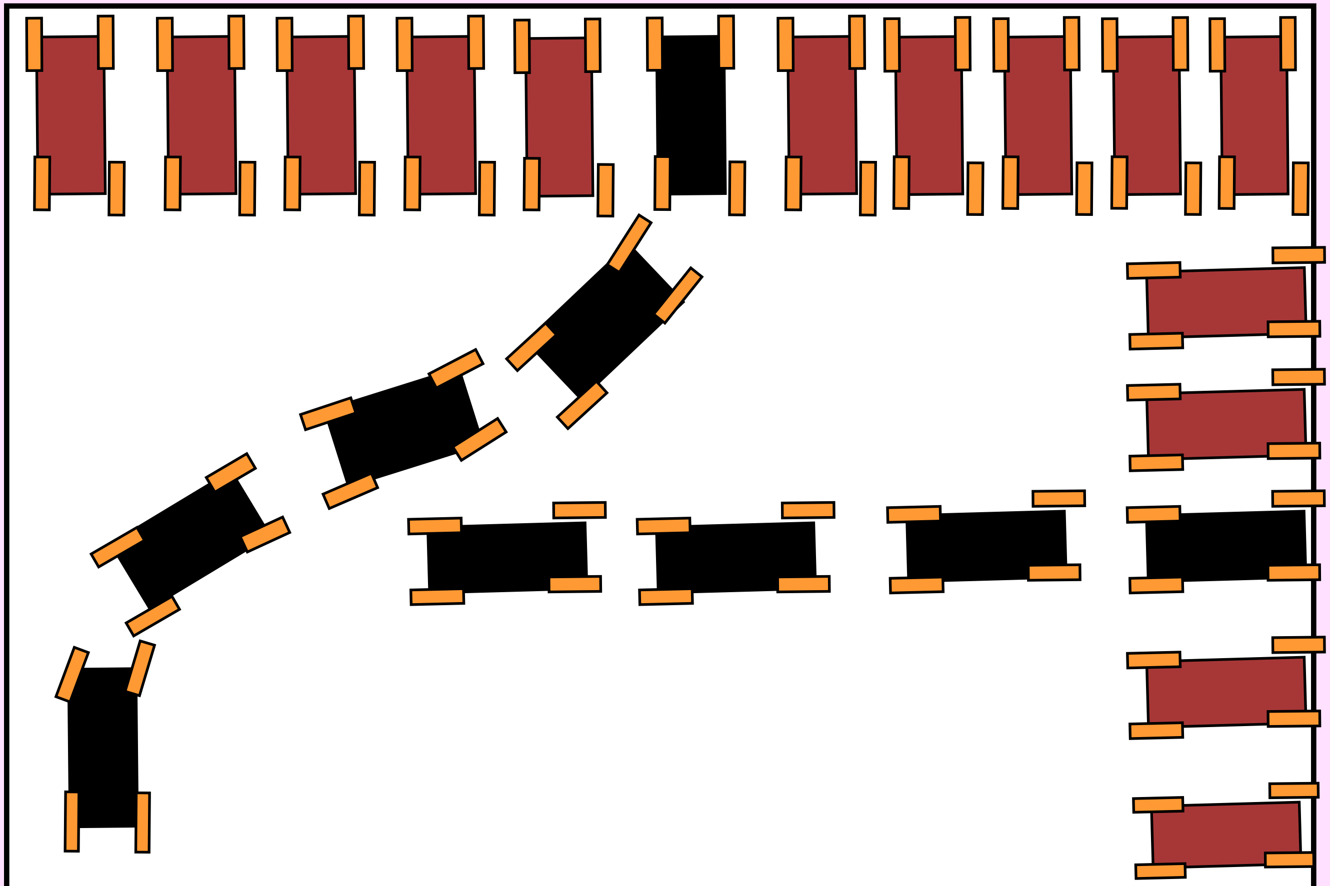


$$\frac{\overline{\partial x_{k+1}}}{\overline{\partial v}} = \frac{\partial x_{k+1}}{\partial u_k}\,\frac{\partial u_k}{\partial v} + \left(\frac{\partial x_{k+1}}{\partial x_k} + \frac{\partial x_{k+1}}{\partial u_k}\,\frac{\partial u_k}{\partial x_k}\right)\frac{\overline{\partial x_k}}{\overline{\partial v}}$$

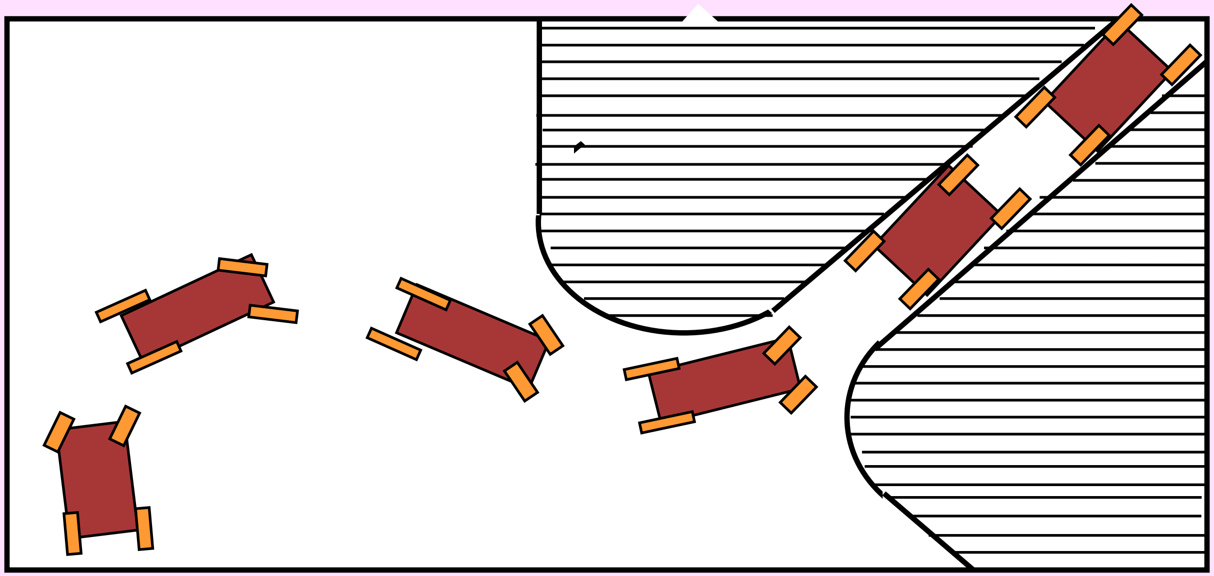
Computed with the system model

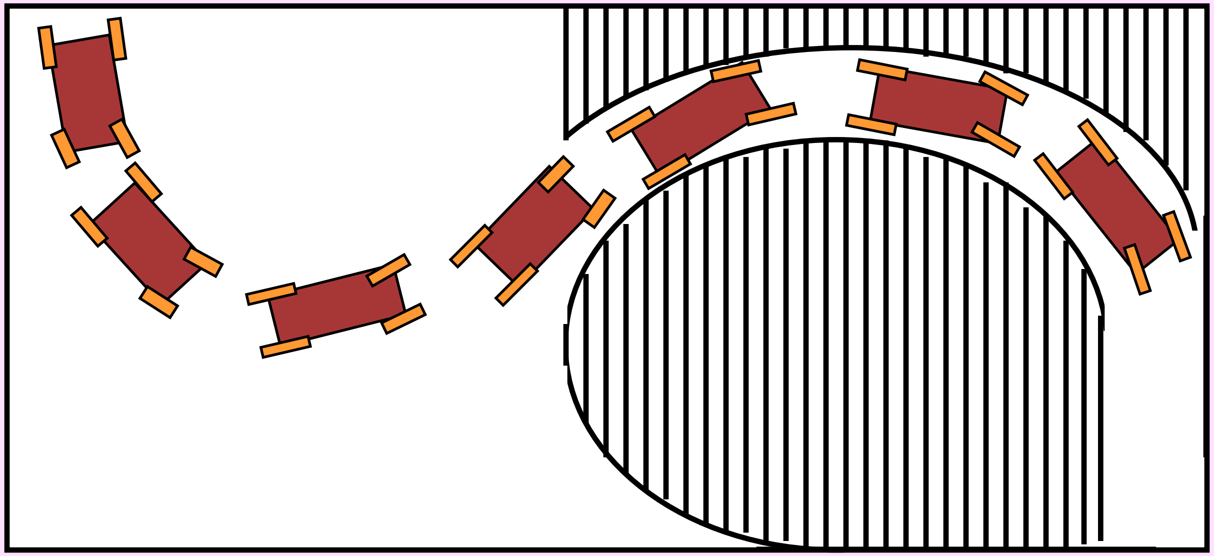
Computed with the neural network

Positioning of Mobile Robots

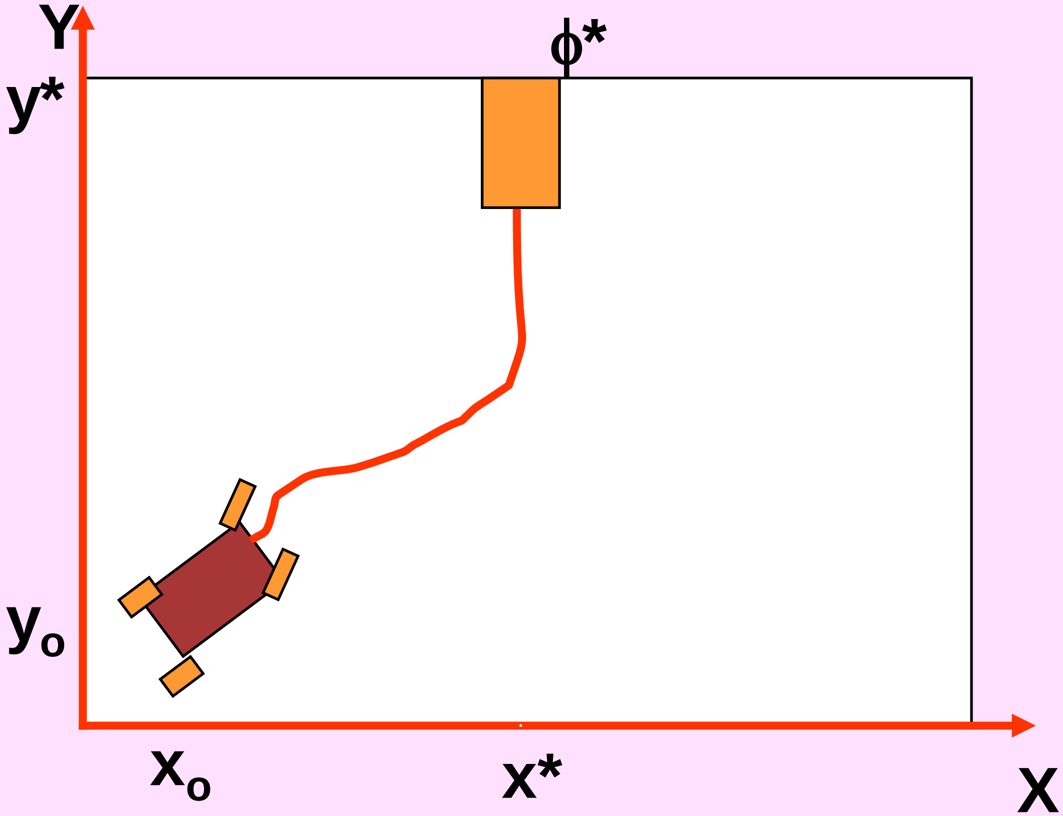


Mobile Robot Following a Road





Control Problem



How to compute steer angle δ

Initial Position

x_o

y_o

ϕ_o

δ

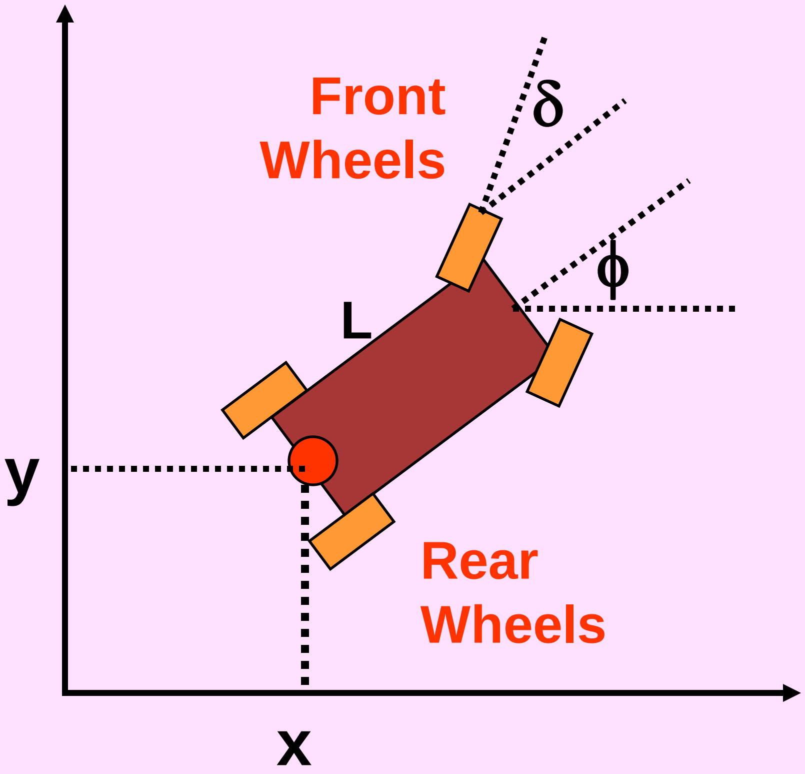
$x^* = 50$

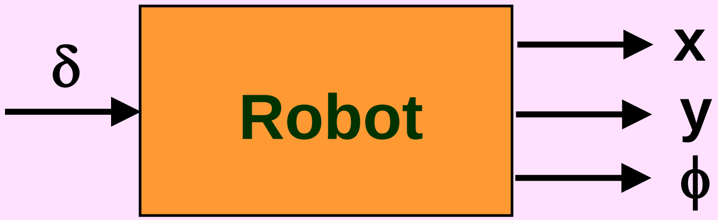
$y^* = 100$

$\phi^* = \pi/2$

Desired Position

Robot Model





$$x(k+1) = x(k) + v\Delta t \cos(\phi(k))$$

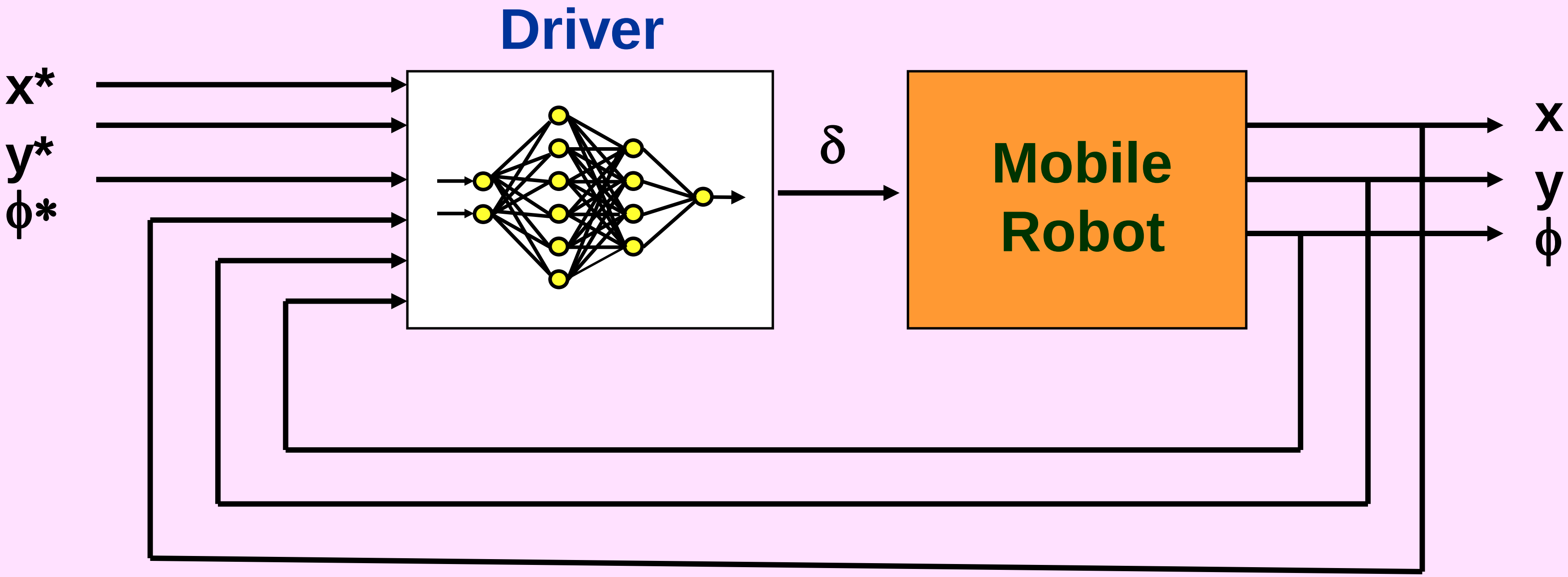
$$y(k+1) = y(k) + v\Delta t \,\mathrm{sen}(\phi(k))$$

$$\phi(k+1) = \phi(k) - v\Delta t / L \tan(\delta(k))$$

- Backward motion
- Constant speed
- No slipping – No skidding

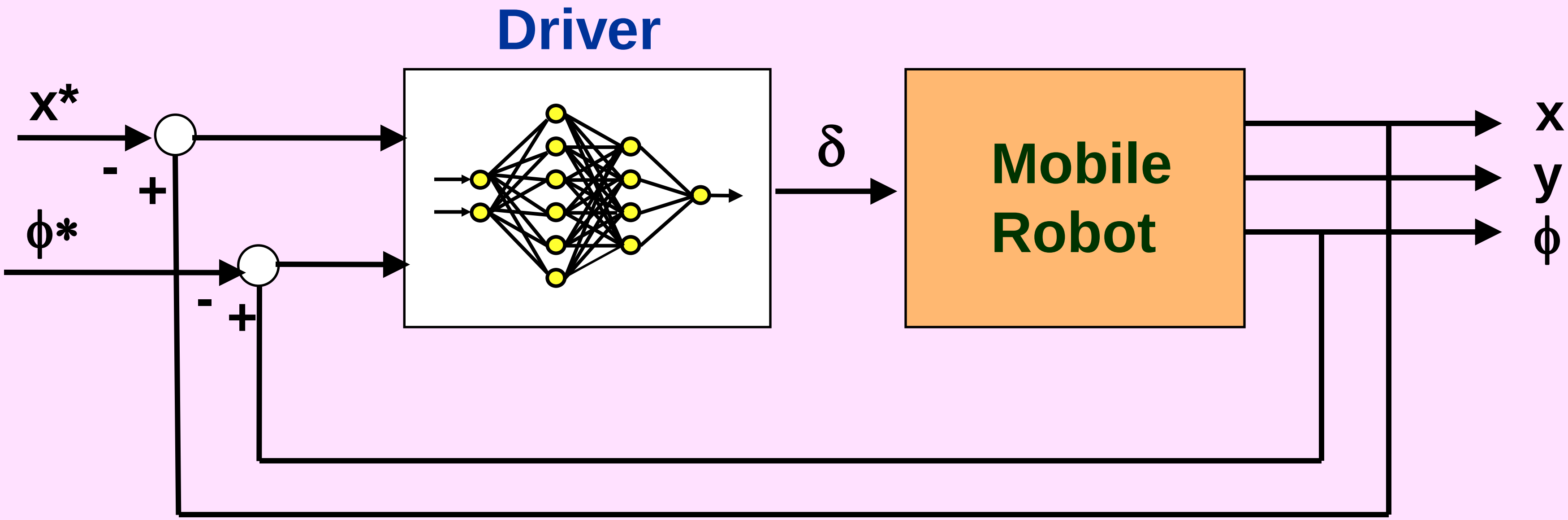
Positioning of Mobile Robot

Control Structure



Positioning of Mobile Robot

Control Structure



Given problem characteristics, coordinate y is not used for control

Dynamic Back Propagation

Robot Model

$$x(k+1) = x(k) + v\Delta t \cos(\phi(k))$$

$$\phi(k+1) = \phi(k) - v\Delta t / L \tan(\delta(k))$$

$$x_k = \begin{vmatrix} x(k) \\ \phi(k) \end{vmatrix} \qquad u_k = \tan(\delta(k))$$

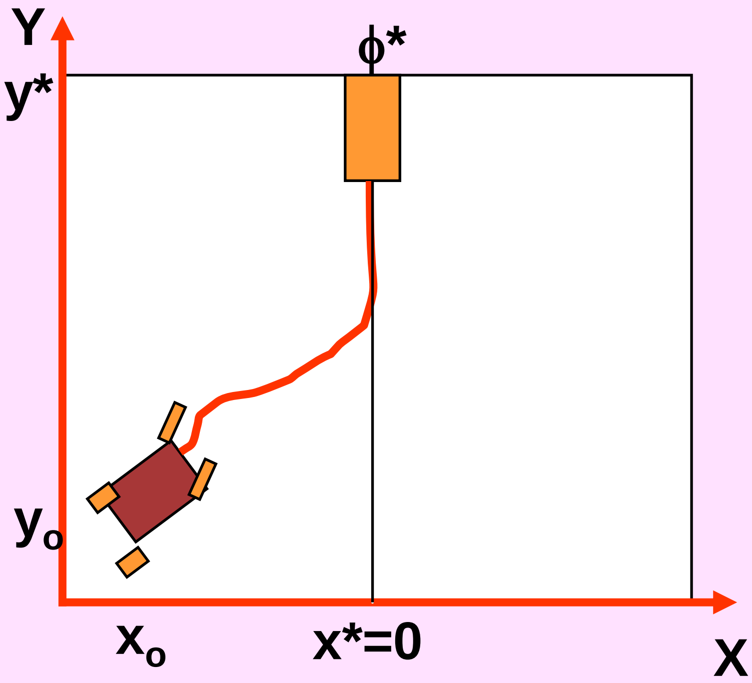
$$\frac{\overline{\partial x_{k+1}}}{\overline{\partial v}} = \frac{\partial x_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial v} + \left(\frac{\partial x_{k+1}}{\partial x_k} + \frac{\partial x_{k+1}}{\partial u_k}\frac{\partial u_k}{\partial x_k}\right)\frac{\overline{\partial x_k}}{\overline{\partial v}}$$

Computed with the system model

$$\frac{\partial x_{k+1}}{\partial x_k} = \begin{vmatrix} 1 & -v\Delta t \sin(\phi(k)) \\ 0 & 1 \end{vmatrix}$$

$$\frac{\partial x_{k+1}}{\partial u_k} = \begin{vmatrix} 0 \\ v\Delta t/L \end{vmatrix}$$

Incremental Learning



Train the neural network for positions close to x*=0 (four positions)

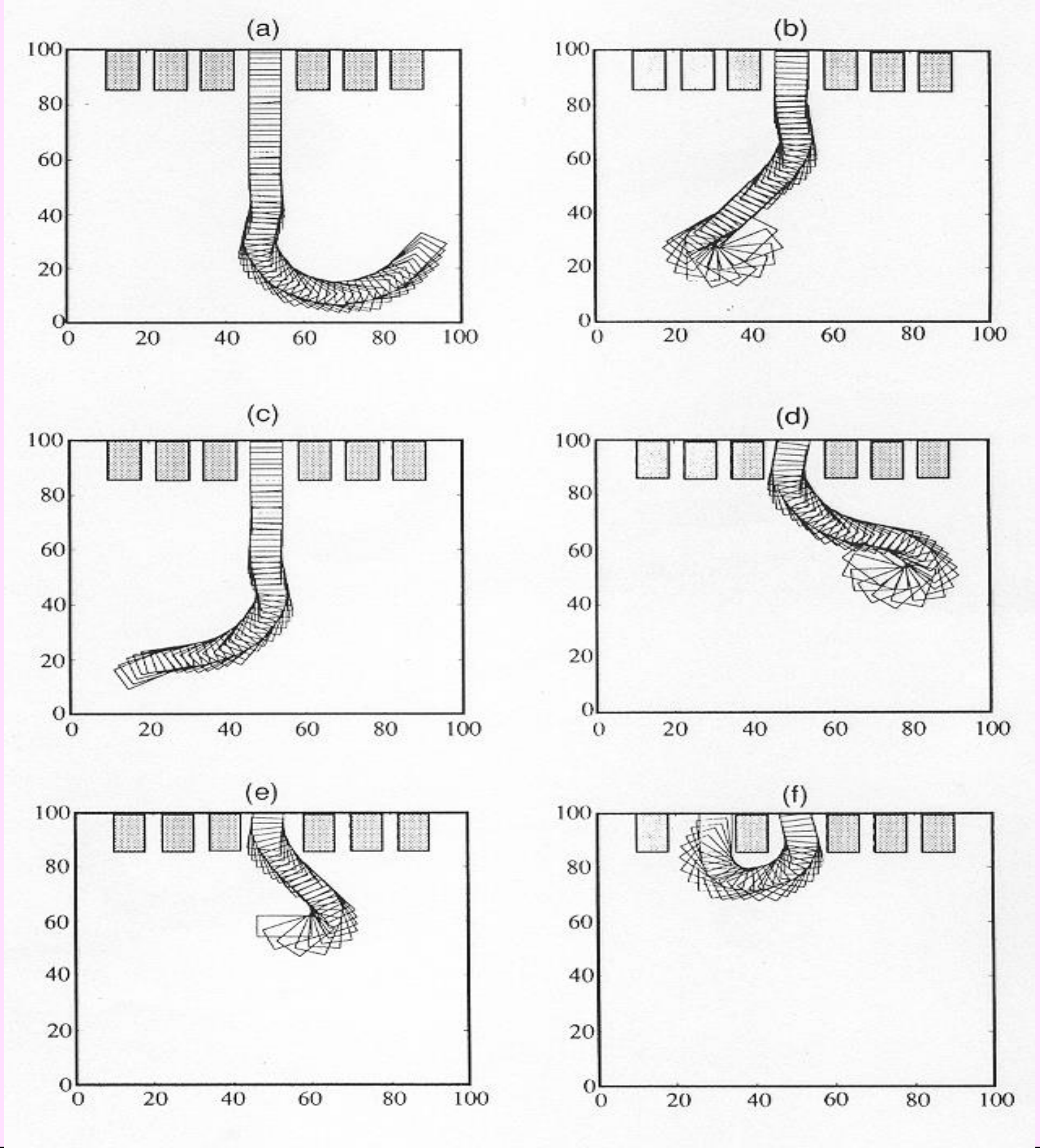
x =	-2	-2	2	2
ϕ =	$-\pi/2$	$\pi/2$	$-\pi/2$	$\pi/2$

Train the neural network for far away positions

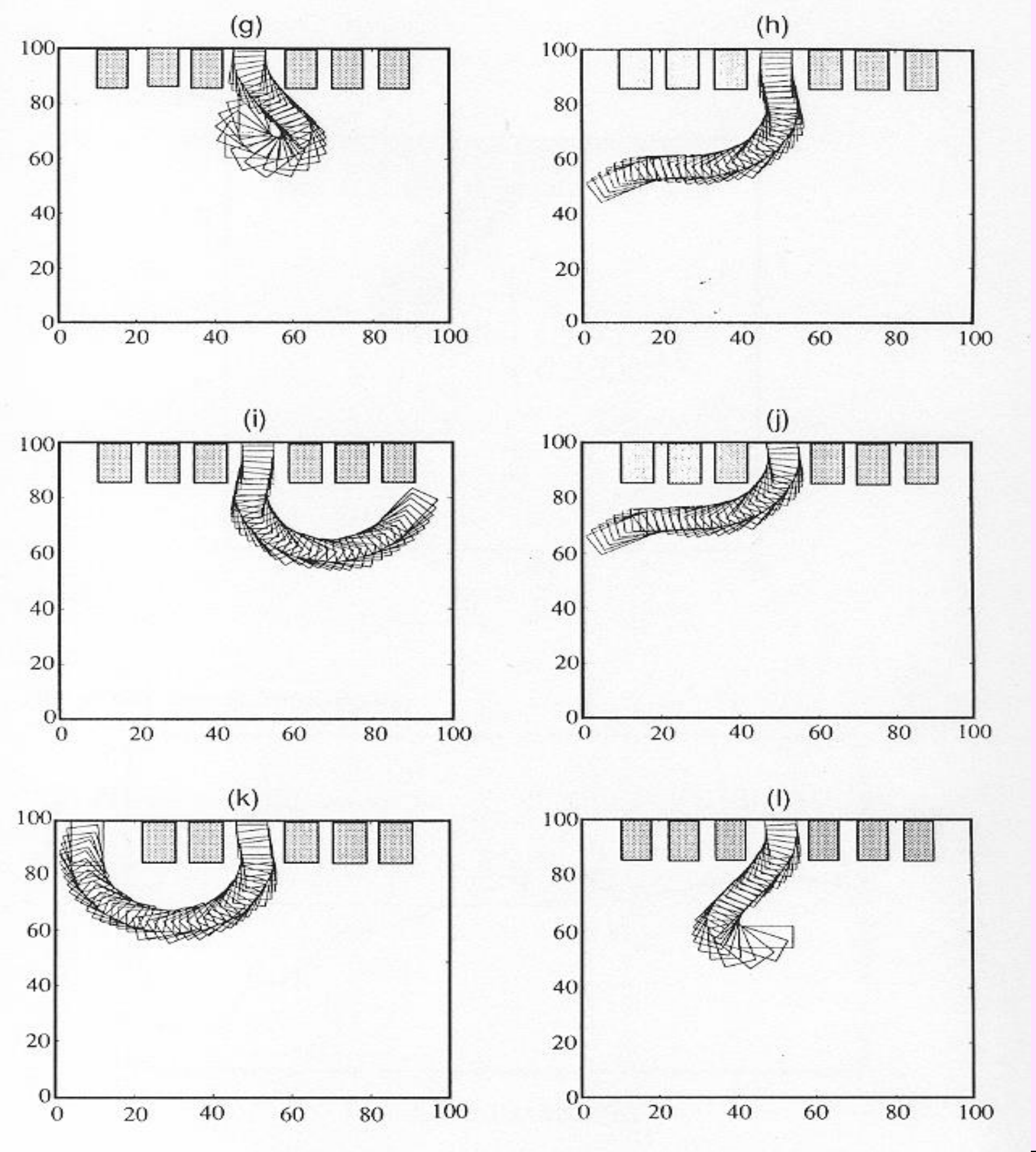
x =	-4	-4	4	4
ϕ =	$-\pi/2$	$\pi/2$	$-\pi/2$	$\pi/2$

x =	-6	-6	6	6
ϕ =	$-\pi/2$	$\pi/2$	$-\pi/2$	$\pi/2$

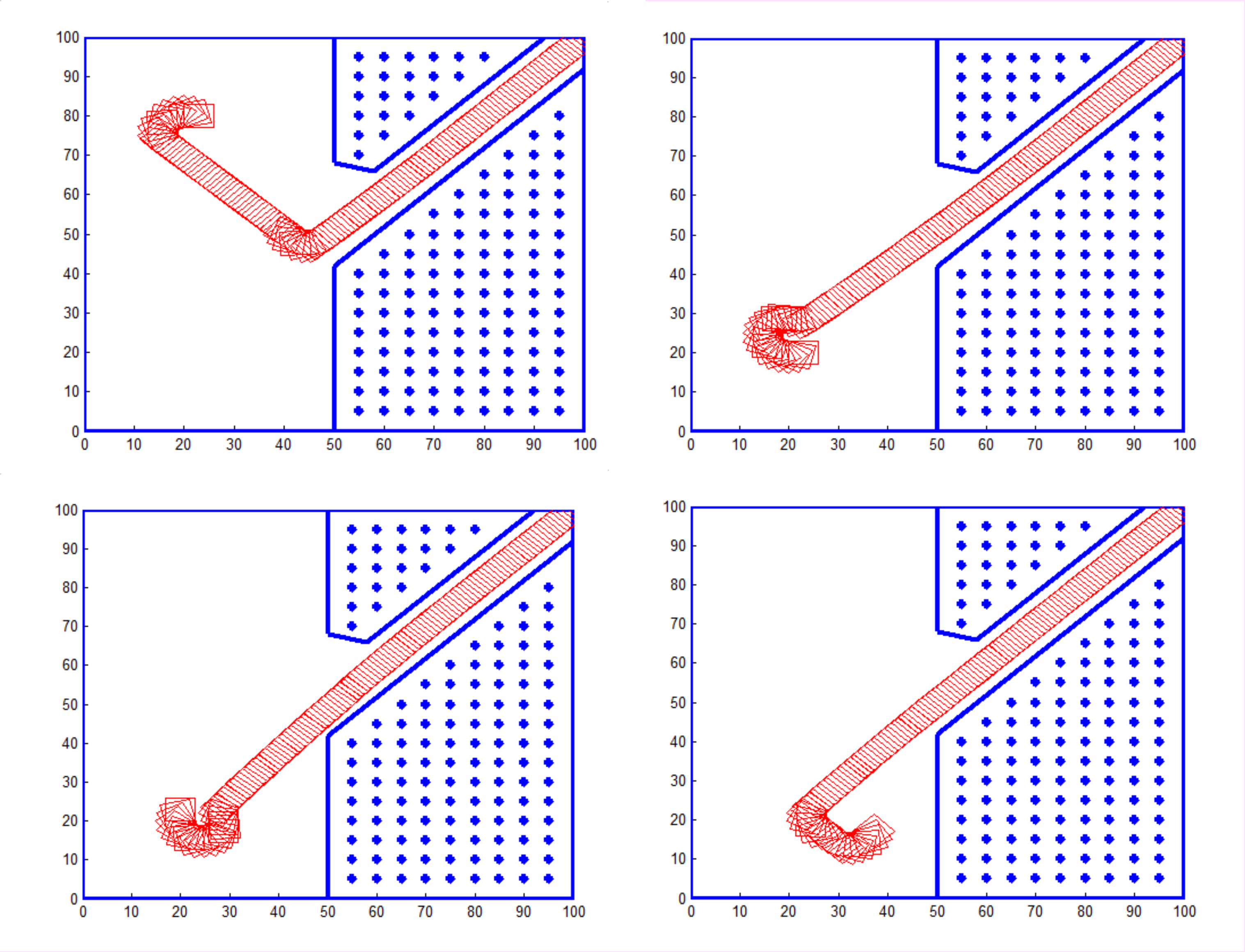
Trajectories of Mobile Robot to Achieve a Final Desired Position



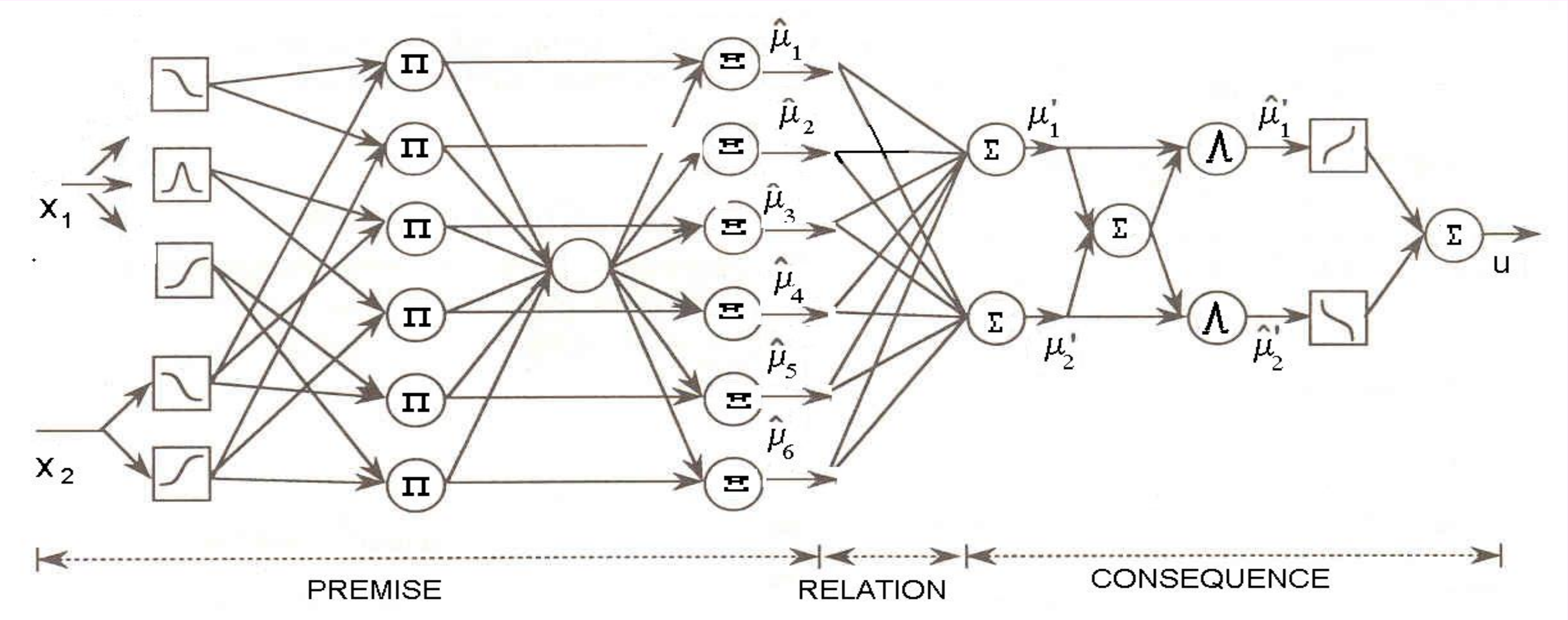
Trajectories of Mobile Robot to Achieve a Final Desired Position



Trajectories of Mobile Robot to Follow a Road



Fuzzy Neural Network

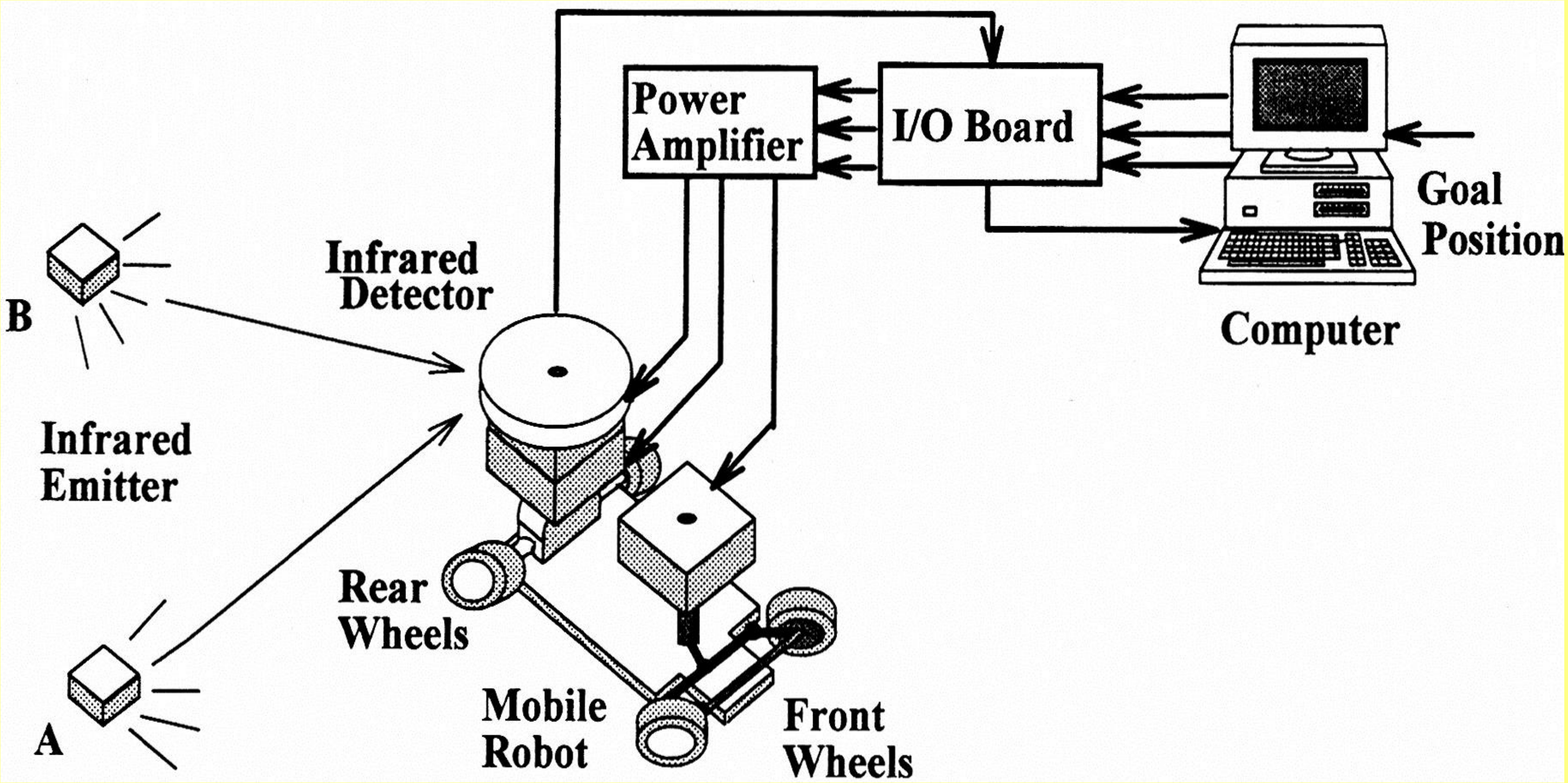


Integrates:

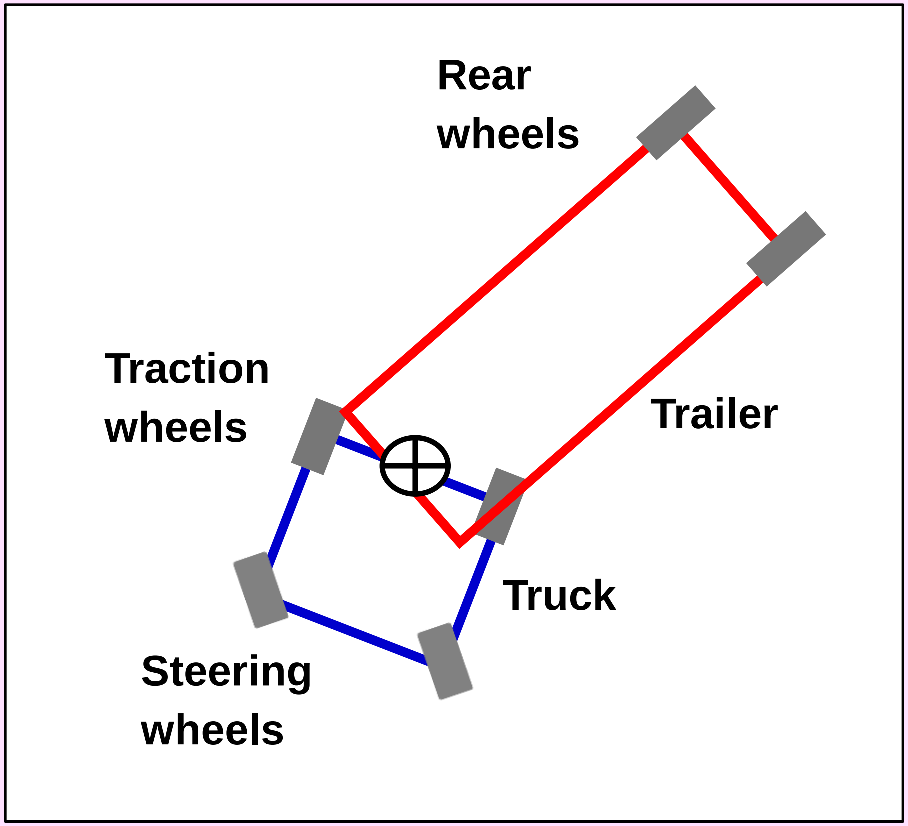
Knowledge → **IF -THEN Rules (Fuzzy)**

Data → **Training (Neural Network)**

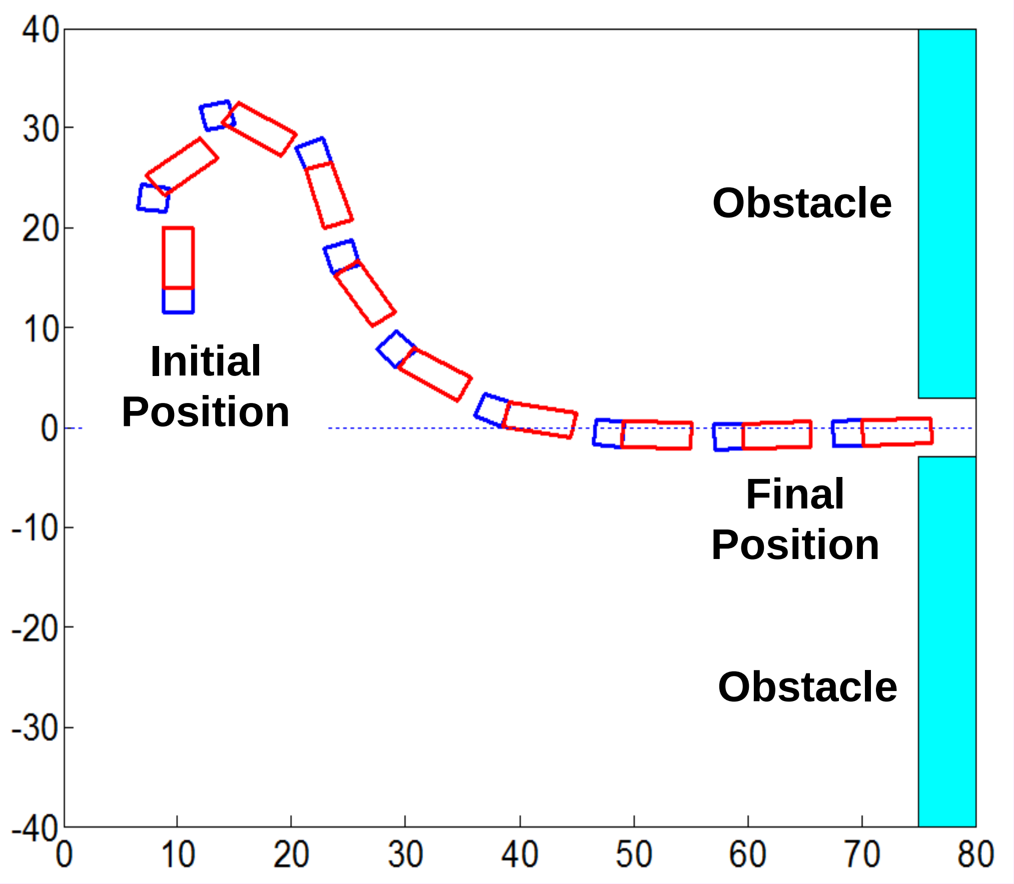
Experimental Mobile Robot



Control of a Truck-Trailer Mobile Robot

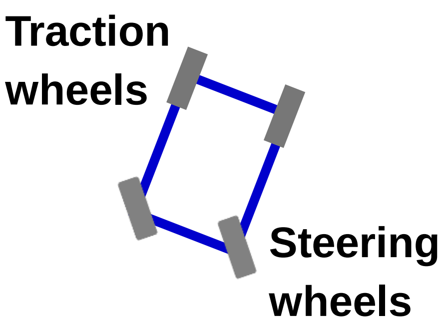


Control of a Truck-Trailer Mobile Robot

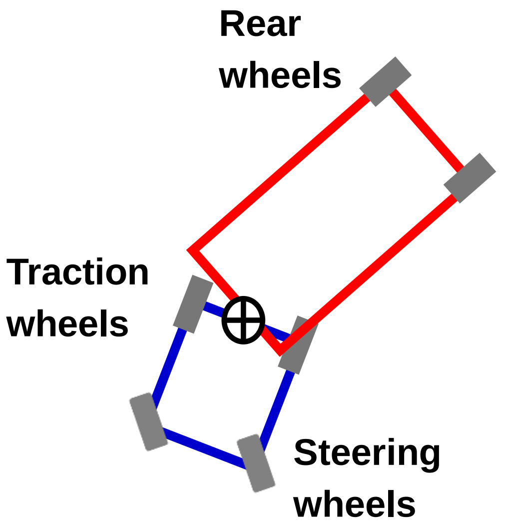


Incremental Learning

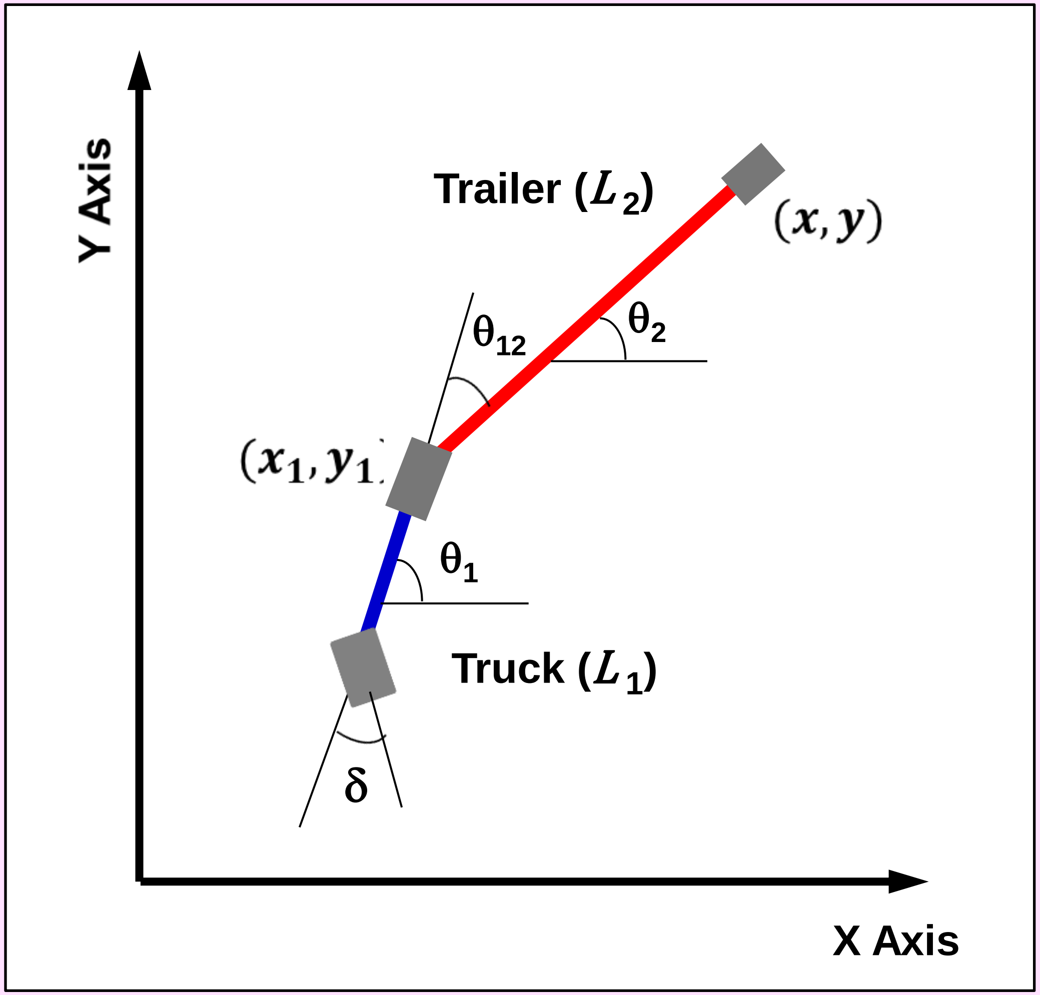
- Train the neural network for controlling a car $\theta_{12} = 0$
  - Close to the desired position
  - Away from the desired position



- Train the neural network for controlling a truck-trailer $\theta_{12} \neq 0$
  - Small values of θ_{12}
  - Higher values of $\theta_{12} < \pi/2$



Control of a Truck-Trailer Mobile Robot



$$\dot{x} = v \cos\theta_{12} \cos\theta_2$$

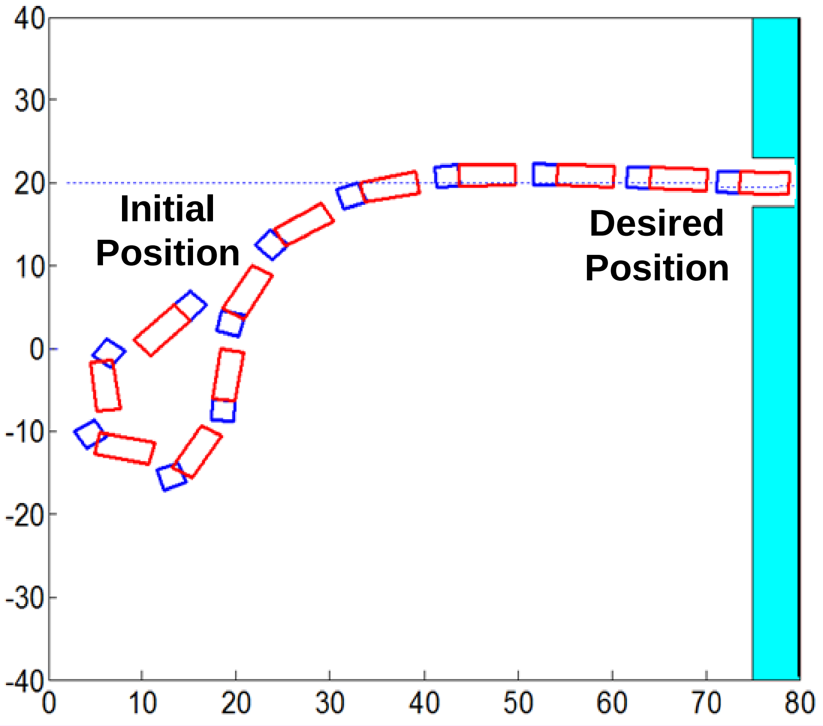
$$\dot{y} = v \cos\theta_{12} \sin\theta_2$$

$$\dot{\theta}_1 = -\frac{v}{L_1} \tan\delta$$

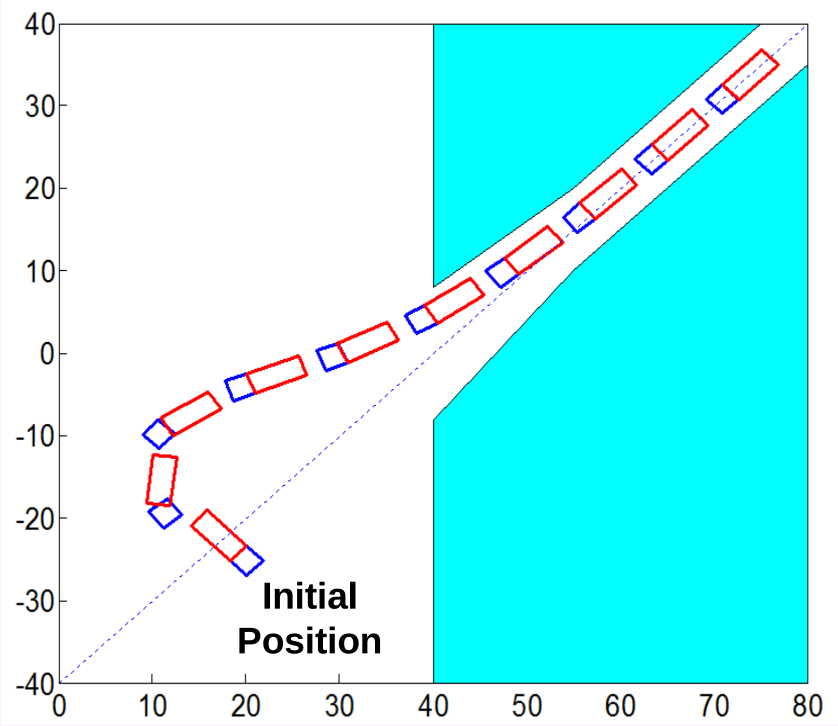
$$\dot{\theta}_2 = -\frac{v}{L_2} \sin\theta_{12}$$

Control of a Truck-Trailer Mobile Robot

Achieving a Goal Position

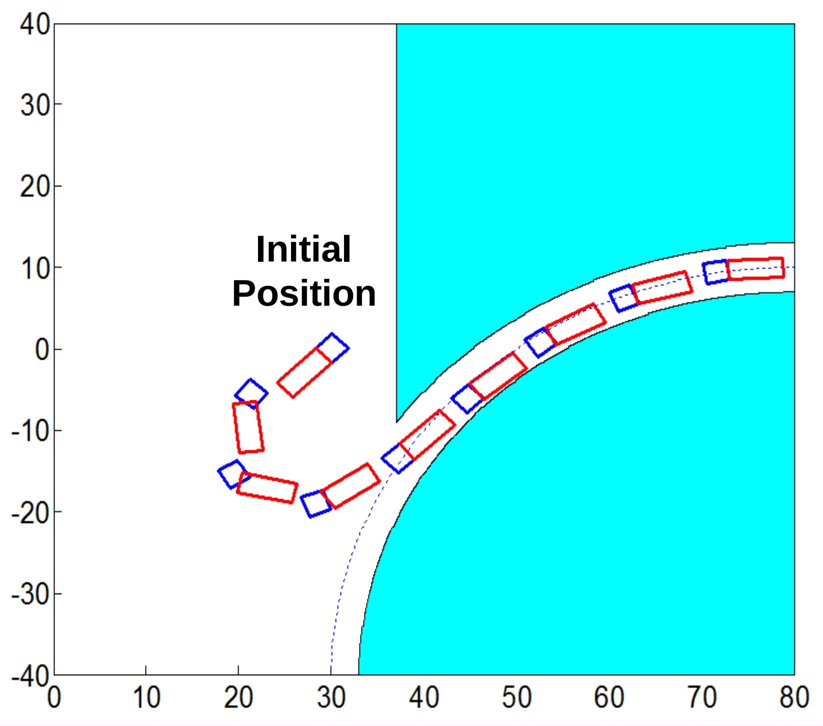


Following a Straight Line

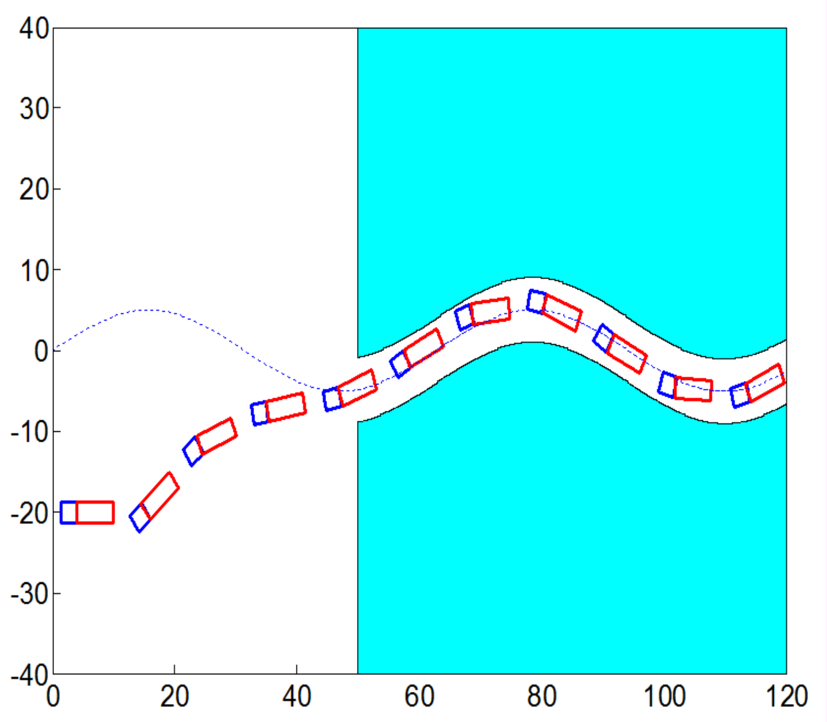


Control of a Truck-Trailer Mobile Robot

Following a Curved Path



Following a Sinusoidal Path



Thank you for your attention!

Antonio Moran, Ph.D.

amoran@ieee.org