Proceedings of the 6th WSEAS Int. Conf. on FUZZY SYSTEMS, Lisbon, Portugal, June 16-18, 2005 (pp73-78)

A new Soft Computing based Controller for a Biped Robot

SALVATORE PENNACCHIO, ANTONIO LA MALFA,
LILIANA LO PRESTI, MARIANNA ROTOLO
University of Palermo
Viale delle Scienze, 90128 Palermo
ITALY
http://www.dias.unipa.it

Abstract: - This paper presents a new controller, based on a soft computing approach, for a biped robot with two infrared sensors. This control system permits the robot to move in an environment avoiding obstacles. The soft computing approach enables the robot to move under unknown situations. The system has been realized in Matlab environment and it is composed by: a fuzzy controller and an equivalent neural network. Results has been tested creating two applications, one running on PC and the other on the robot, managing serial communication in the two directions.

Key-Words: - Basic Stamp, fuzzy, Matlab, neural network, neuro-fuzzy controller, serial communication

1 Introduction

Control systems based on a fuzzy logic approach differ from the others because trough fuzzy logic it is possoble making inferences even under uncertainty situations and under unknow environmental conditions and accidental causes.
In the last years the fuzzy logic approach has been chosen by several researchers in the field of robotics [1,2,3]. Above all the fuzzy logic controllers appear in those problems with uncertain inputs and with unknown environments [4].
Fuzzy navigation systems are simpler to implement than other navigation systems because they can handle infinite navigation situations with a finite set of rules.
Trough the neural network it is possible, as in human life, a period of learning . During this period the robot can learn the best rules in a new environment [5].
This paper presents a new method, called E@syToddler, which permits to conjugate the versatility of a non-Aristotelian inferential system with the computational power of a distributed system like a neural network . Abandoned Java, not specifically computing-oriented, E@syToddler adopts Matlab development environment by which we realized the neuro-fuzzy controller that permits to manage robot motions; particularly the system determines, beginning from the information detected by sensors, the most adequate behaviour to move the robot in the environment avoiding obstacles.
.

2 Platform and robot

During experimental proofs, we have used Toddler™ biped robot, produced by Parallax Inc.
A couple of servomotors determines motion [10]; one moves the centre of gravity to keep body's balance, the other

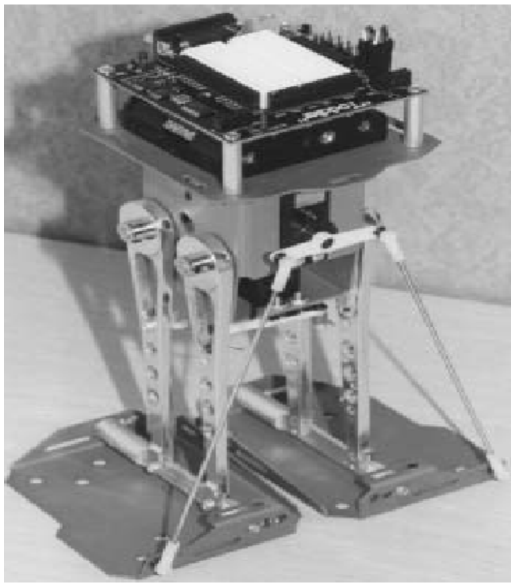


Fig. 1. Toddler™ Robot

moves legs forward and backward keeping feet parallel to the plain.
A BASIC Stamp 2 module assembled in the upper part of Toddler™ consents to control the servomotors and the two infrared sensors.
Programs are written in a BASIC-like language; they are realized on PC in a suitable development environment named BASIC Stamp Editor; on the contrary, they are executed on board after a preliminary storage on the robot's EEPROM.

3 Fuzzy controller

The Fuzzy Controller, realized by the Fuzzy Toolbox available in Matlab, is a Sugeno controller and presents two numerical inputs, correspondent to the distances taken by

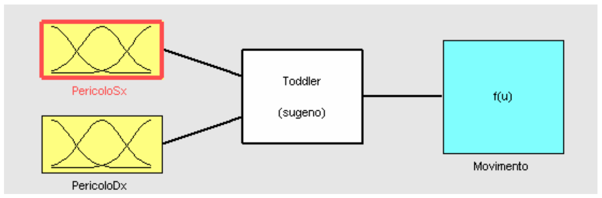


Fig. 2. The Fuzzy Logic Controller

the robot two sensors.
The output is a numerical value between 0 and 5. We have chosen three membership functions in order to evidence the presence of a low, medium or high danger respectively in relation with possible values every input can assume. Input can vary between 0 and 5.

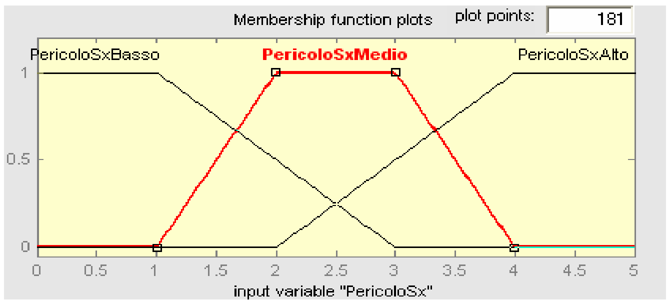


Fig. 3 Input Membership Functions

The first membership function is “PericoloSxBasso” (a low danger on the left) which assumes value 1, when the input value is between 0 and 1, and decreases until reaching the value 0 if input is greater than or equal to 3.
The second membership function is “PericoloSxMedio” (a medium danger on the left); it is a symmetrical trapezoidal function that assumes value 0 in the ranges (0,1) and (4,5), value 1 in the range (2,3) and intermediate values in the remaining ranges.
Finally, the third membership function is “PericoloSxAlto” (a high danger on the left) that assumes value 0 if input is lower than 2, has an increasing linear trend between 2 and 4 and assumes value 1 if input is greater than 4.
These functions are the same for both inputs; so in fig. 3, we only present membership functions of the “PericoloSx” input (input from the left sensor).
According to the definition of Sugeno fuzzy logic controllers, output membership functions are simple horizontal straight lines [2]; in this case they are nine, one for each possible combination of inputs. Only a single membership function is activated when the elaboration of the controller, executing the inference described in the following, terminates.
First, numerical inputs are fuzzified and then used in the *antecedent* part of fuzzy rules that are in the form:

if *antecedent1* ***and*** *antecedent2* ***then*** *consequent*

In the *consequent* part there are fuzzy values that output can assume: go on (Avanza), a little turn to the left (GiraPocoSx), a turn to the left (GiraSx), a little turn to the right (GiraPocoDx), a turn to the right (GiraDx) and draw back (Indietreggia). The nine chosen rules for the inference phase are those in Table I.
The nine values yielded by the inference phase are finally defuzzified, i.e. become numerical values belonging to the range [0,5], and returned by the controller

AND	PERICOLO BASSOSX	PERICOLO MEDIOSX	PERICOLOA LTOSX
PERICOLO BASSODX	*AVANZA*	*GIRA POCODX*	*GIRADX*
PERICOLO MEDIODX	*GIRA POCOSX*	*GIRA POCODX*	*GIRADX*
PERICOLO ALTODX	*GIRASX*	*GIRASX*	*INDIETREG-GIA*

Table 1. Motion Table

4 Neural Network

After defining the fuzzy logic controller, we have realized the equivalent neural network (Fig. 4) which reproduces the behaviour of the controller [9].
In order to do this, we made use of the Neural Network Toolbox, available in Matlab, and chose a feed-forward structure characterized by two inputs, six different neural layers and an only output.
Inputs are named “PericoloSx” (input from the left sensor) and “PericoloDx” (input from the right sensor) and correspond to data perceived by IR sensors during robot motion; inputs are discrete and assume integer values between 0, which corresponds to the absence of danger, and 5, corresponding to the highest danger.
Layers process inputs contributing towards the calculation of the network global output which still is a value between 0 and 5.
The first three layers are directly connected to the inputs by weighted unary connections and make fuzzification, that is simulate the behaviours of the inputs membership functions of the fuzzy controller [6].
The first layer is composed by two neurons characterized by a trapezoidal activation function corresponding to the “PericoloBasso” membership function; so it assumes the value 1 if the weighted sum of inputs is between 0 and 1 while it produces the value 0 if this sum is greater than 3; moreover, if the input is between 1 and 3 the function has a decreasing linear trend.
Also the second layer is composed by two neurons characterized by a trapezoidal activation function which

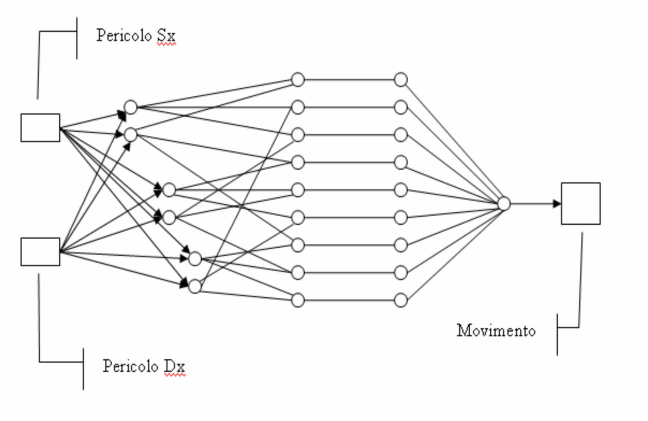


Fig. 4 The neural network

corresponds to the “PericoloMedio” membership function; but this time the function is symmetrical since it has to assume the value 1 if the weighted sum of inputs is between 2 and 3, and the value 0 if this sum is lower than 1 and greater than 4. Instead, in the rest of the range, the function is between 0 and 1 and has a linear trend, increasing between 1 and 2 and decreasing between 3 and 4. Finally, inputs are also connected to the two neurons of the third layer which activate only if the weighted sum is greater than 4. The output assumes the value 0 if this sum is lower than 2 while it assumes values between 0 and 1 if input values belong to the range (2,4). This time too, we have a trapezoidal activation function which corresponds to the “PericoloAlto” membership function. A step function characterizes all nine neurons belonging to the fourth layer which implements the fuzzy rules system so performing the inference. Indeed, each neuron corresponds to one different rule in the controller and produces a binary value: inputs of each neuron are composed by a couple of values coming from the previous layer and the output is 1 if the weighted sum of these values is greater than 0, otherwise it is 0. The correspondence between the neurons and the couples of inputs is summarized in the table II in which each input is named Iij, with i being the layer and j being the neuron; neurons are numbered from top to bottom from 1 to 9 in fig 4.

Neuron	Input Couple
1	(I_{11}, I_{12})
2	(I_{11}, I_{32})
3	(I_{11}, I_{22})
4	(I_{21}, I_{12})
5	(I_{21}, I_{22})
6	(I_{21}, I_{32})
7	(I_{31}, I_{12})
8	(I_{31}, I_{22})
9	(I_{31}, I_{32})

Table 2. Couple of inputs to fourth layer neurons

Last two layers perform defuzzification, that is they simulate the behaviours of the output membership functions. In particular, the fifth layer is a simple transducer and each neuron in it is only connected to the processing unit that occupies the same position in the previous layer;
all connections have unary weights and all neurons have the same activation function which is a step function centred in 0. The weighted sum of the outputs of the fifth layer is the input of the single neuron in the last layer: this neuron is characterized by a purely linear activation function and, as said, yields a value which is the network global output and is between 0 and 5. After the conclusion of network design, we have performed the training, i.e. we have put in input the desired values and modified the weights of connections in order to minimize the error between the desired output and the values we got. We make use of the Widrow–Hoff learning function, also called minimum squared error (MSE), which calculates the variation of the given neuron weights beginning from the weighted sum of inputs, the error and the learning rate.
In particular, the learning rate is chosen in order to optimize the learning process which is realized by the gradient descent algorithm. We needed 5000 learning epochs and, after their conclusion, we have obtained the minimum value of error (fig. 5), which is equal to 0.1/5, i.e. about to 2%, quite acceptable for prefixed purposes.

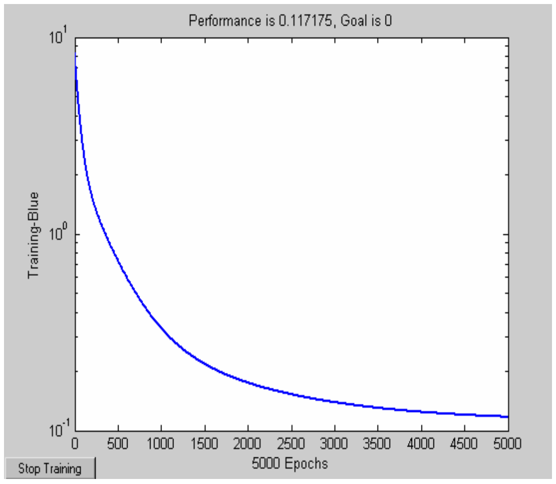


Fig. 5 The error function after 5000 learning epochs

5 Experiments and results

To move in the environment avoiding obstacles, the robot has to cyclically repeat a definite sequence of actions, that is the robot has to:

1. use the sensors to read its distance from obstacle,
2. transmit the obtained readings to the neural network,
3. wait for the result of network computation,
4. select the behaviour corresponding to the obtained result,
5. execute the requested movement.

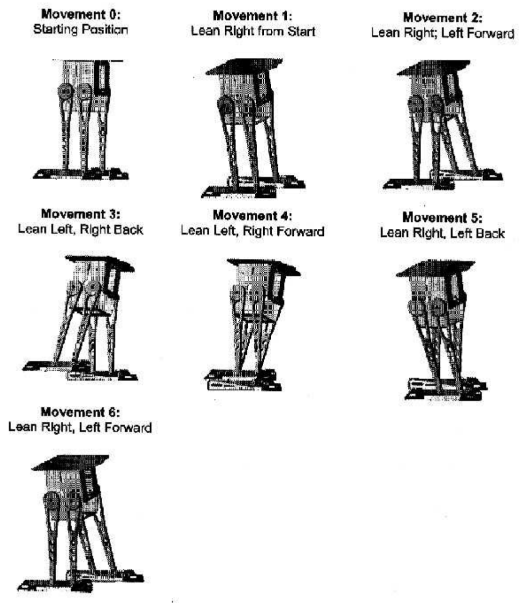
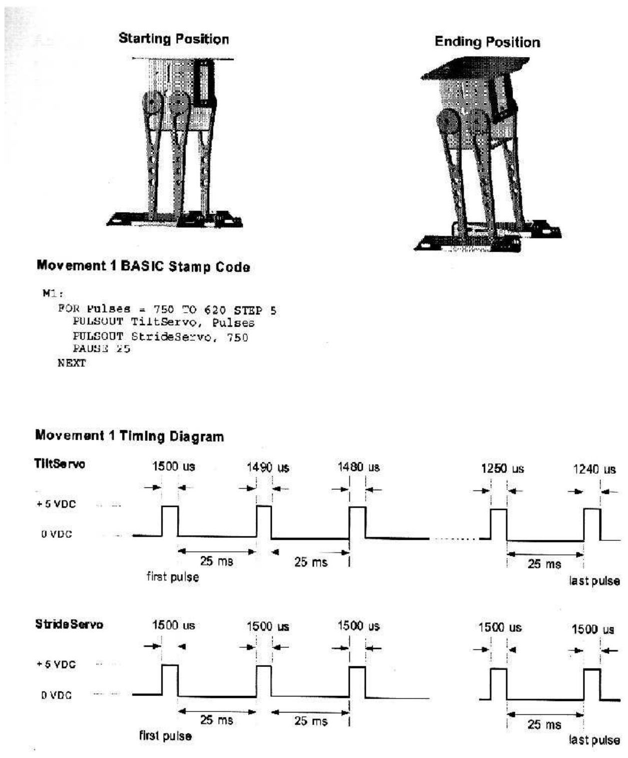


Fig.6 Basic walking movements of the robot

Obstacle detection is obtained using the readings of the right and the left sensors at five different frequencies that are 37.500 KHz, 38.250 KHz, 39.500 KHz, 40.500 KHz and 41.500 KHz [10].
Indeed, the use of an only frequency supplies no information about the distance of the robot from the object since the sensor output is high or low and it can only indicate the presence or the absence of an object. Instead, valuing the sensor output at 5 different frequencies, it is possible to know how far the obstacle is from the robot and to act consequently; indeed, we have observed that sensors are more sensible to low frequencies and then if the sensor output is high for all the frequencies the object is very near, if it is high only for lower frequencies the object is on average distant, if it is low for all the frequencies there are no obstacles in the neighbourhood.
Detection of each sensor produces a value between 0 and 5 representing both the number of frequencies to which sensor as detected the object and a qualitative measure of the distance of the obstacle from the robot.
This value is given in input to the neural network establishing on the COM port, by a RS 232 cable, a serial communication between PC and robot[10]. The network processes the given value and supplies an output that is transmitted to the robot which selects the motion correspondent to these new data. We have chosen to move the robot in the direction where danger is minimum, so Toddler goes on in absence of obstacles, goes towards the direction opposite to that where it has detected an object or it draws back in front of too near obstacles.
Each behaviour is composed of a sequence of elementary motions realized by sending a pulse to the servomotor we want to control and leaving the other disconnected.
The elementary motions are 6 (Fig. 6) and are:

1. move the body to the right
2. move the body to the left
3. move the body to the central position
4. move the right leg
5. move the left leg
6. move the leg to the central position

We have chosen the correspondence between output of the network and the behaviour to implement. In Fig. 7 we show Movement 1 as an example.



```
M1:
  FOR Pulses = 750 TO 620 STEP 5
    PULSOUT TiltServo, Pulses
    PULSOUT StrideServo, 750
    PAUSE 25
  NEXT
```

Fig. 7 Example of Movement 1

6 Conclusions

In this paper has been presented a new controller for a biped robot with a neuro fuzzy approach. Through a serial communication between the Pc and the robot it has been possible to optimize the control system because it is on the PC. Experiments and results demonstrate the validation of our E@sytoddler system. We believe that this kind of robots will be used in the future for helping disable persons and then our research will continue for a better performance.

References:

[1] S. Pennacchio, F.M. Raimondi, E. Autiello, A. Senia, "Fuzzy Logic Controller for Behaviour-Based Navigation Systems", *WSEAS TRANSACTIONS on CIRCUITS and SYSTEMS,* pp. 379-383, April 2004.

[2] S. Pennacchio, F.M. Raimondi, "Fuzzy Motor Schema for Mobile Robots", *WSEAS TRASACTIONS on SYSTEMS*, pp. 2211-2214, July 2004.

[3] T. E. Mora, E. N. Sanchez, "Fuzzy Logic Based Real Time Navigation Controller for a Mobile Robot", IEEE Int. Conf.Intelligent Robots and Systems,Ottobre 1998.

[4] C. Barret, M. Benreguieg, H. Maaref, "Fuzzy Agents for Reactive Navigation of a Mobile Robot", *IEEE* Int. Conf. Knowledge Based Intelligent Electronic System, 21-23 Maggio 1997.

[5] T. L. Lee, C. J. Wu, "Fuzzy Motion Planning of Mobile Robots in Unknown Enviroments", *Journal of Intelligent and Robotic Systems* 37, 2003.

[6] Li, Harari, Wong, Kapila, "Matlab-Based Graphical User Interface Development for Basic Stamp 2 Microcontroller Projects", Settembre 2003

[7] Insop Song", Fuzzy Logic and Neural Network Controller", SD558 Project, Aprile 2002

[8] AA.VV.," Advanced Robotics with the Toddler", Student Guide, version 1.2, www.parallax.com

[9] Howard Demuth, Mark Beale, "Neural Network Toolbox", The MathWorks, Inc., Gennaio 1998

[10] AA.VV.," Fuzzy Logic Toolbox", The MathWorks, Inc., Settembre 2002

Appendix a: Matlab code for the fuzzy logic controller

ToddlerFLC.fis

```
[System]
Name='Toddler'
Type='sugeno'
Version=2.0
NumInputs=2
NumOutputs=1
NumRules=9
AndMethod='prod'
OrMethod='probor'
ImpMethod='min'
AggMethod='max'
DefuzzMethod='wtaver'
[Input1]
Name='PericoloSx'
Range=[0 5]
NumMFs=3
MF1='PericoloSxBasso':'trapmf',[0 0 1 3]
MF2='PericoloSxMedio':'trapmf',[1 2 3 4]
MF3='PericoloSxAlto':'trapmf',[2 4 5 5]
[Input2]
Name='PericoloDx'
Range=[0 5]
NumMFs=3
MF1='PericoloDxBasso':'trapmf',[0 0 1 3]
MF2='PericoloDxMedio':'trapmf',[1 2 3 4]
MF3='PericoloDxAlto':'trapmf',[2 4 5 5]

[Output1]
Name='Movimento'
Range=[0 1]
NumMFs=9
MF1='Indietreggia':'constant',[5]
MF2='Avanza':'constant',[0]
MF3='GiraPocoDx':'constant',[1]
MF4='GiraDx':'constant',[2]
MF5='GiraPocoSx':'constant',[3]
MF6='GiraSx':'constant',[4]
MF7='GiraSx2':'constant',[4]
MF8='GiraPocoDx2':'constant',[1]
MF9='GiraDx2':'constant',[2]
[Rules]
1 1, 2 (1) : 1
1 3, 6 (1) : 1
1 2, 5 (1) : 1
2 1, 3 (1) : 1
2 2, 8 (1) : 1
2 3, 7 (1) : 1
3 1, 4 (1) : 1
3 2, 9 (1) : 1
3 3, 1 (1) : 1
```

Appendix b: Matlab code for the neural network

ToddlerNN.m

```
function [net] = ToddlerNN
close all;
% Definition of the network structure
net = network;
net.numInputs = 2;
net.inputs{1}.size = 1;
net.inputs{1}.range = [0 5];
net.inputs{2}.size = 1;
net.inputs{2}.range = [0 5];
net.numLayers = 6;
net.biasConnect = [0;0;0;0;0;0];
net.layers{1}.size = 2;
net.layers{1}.transferFcn = 'basso';
net.layers{2}.size = 2;
net.layers{2}.transferFcn = 'medio';
net.layers{3}.size = 2;
net.layers{3}.transferFcn = 'alto';
net.layers{4}.size = 9;
net.layers{4}.transferFcn = 'ToddlerHardlim';
net.layers{5}.size = 9;
net.layers{5}.transferFcn = 'ToddlerHardlim';
net.layers{6}.size = 1;
net.layers{6}.transferFcn = 'purelin';
net.inputConnect = [1 1; 1 1; 1 1; 0 0; 0 0; 0 0];
net.layerConnect = [0 0 0 0 0 0; 0 0 0 0 0 0; 0 0 0 0 0 0; 1 1
1 0 0 0 ; 0 0 0 1 0 0 ; 0 0 0 0 1 0];
net.outputConnect = [0 0 0 0 0 1];
net.targetConnect = [0 0 0 0 0 1];
net.IW{1,1}= [1; 0];
net.IW{1,2}= [0; 1];
net.IW{2,1}= [1; 0];
net.IW{2,2}= [0; 1];
net.IW{3,1}= [1; 0];
net.IW{3,2}= [0; 1];
net.LW{4,1}=[1 1; 1 0; 1 0; 0 1; 0 0; 0 0; 0 1; 0 0; 0 0];
net.LW{4,2}=[0 0; 0 0; 0 1; 1 0; 1 1; 1 0; 0 0; 0 1; 0 0];
net.LW{4,3}=[0 0; 0 1; 0 0; 0 0; 0 0; 0 1; 1 0; 1 0; 1 1];
net.LW{5,4}=[1 0 0 0 0 0 0 0 0;0 1 0 0 0 0 0 0 0;0 0 1 0 0 0
0 0 0;0 0 0 1 0 0 0 0 0;0 0 0 0 1 0 0 0 0;0 0 0 0 0 1 0 0 0;0 0
0 0 0 0 1 0 0;0 0 0 0 0 0 0 1 0;0 0 0 0 0 0 0 0 1];
net.LW{6,5}= [1 1 1 1 1 1 1 1 1];
net.layerWeights{4,1}.learnFcn = 'learnwh';
net.layerWeights{4,2}.learnFcn = 'learnwh';
net.layerWeights{4,3}.learnFcn = 'learnwh';
net.layerWeights{5,4}.learnFcn = 'learnwh';
net.layerWeights{6,5}.learnFcn = 'learnwh';
%Definizione delle funzioni della rete
net.trainFcn='traingd';
net.performFcn='mse';
% Choice of the training parameters
net.trainParam.epochs = 5000;
%Training set
X = [0 0 0 0 0 0 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4
4 5 5 5 5 5 5;
   0 1 2 3 4 5 0 1 2 3 4 5 0 1 2 3 4 5 0 1 2 3 4 5 0 1 2 3 4 5
0 1 2 3 4 5];
Y = [0 0 1 1 2 2 0 0 1 1 2 2 3 3 3 1 2 2 3 3 3 1 2 2 4 4 4 4 5
5 4 4 4 4 5 5];
net.trainParam.lr = maxlinlr(X,'bias');
net = train(NET,X,Y);
```