Artificial Neural Networks

Term Project

Objective

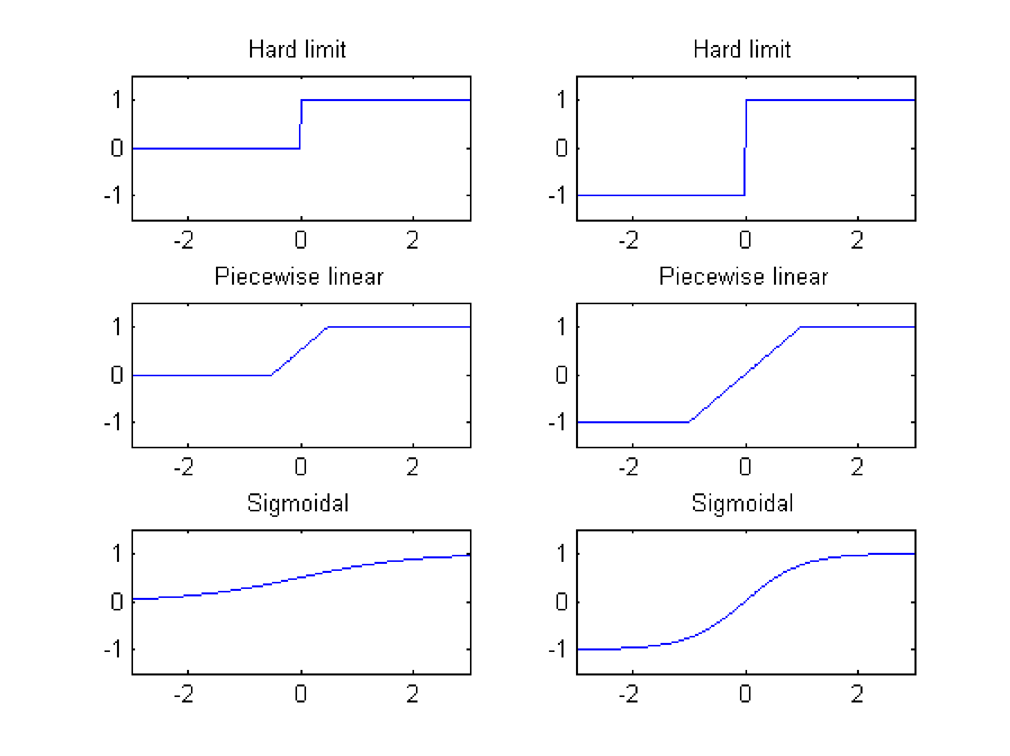
- Activation Function
- Newton's Method
- Steepest Descent Method
- LMS Method
- Pattern Classification
- Rosenblatt's Algorithm
- BPA, ELM and OSELM
- System Indentification and Control

Activation Functions

Types :

- Hard limit function
- Sigmoidal function
- Piecewise linear function

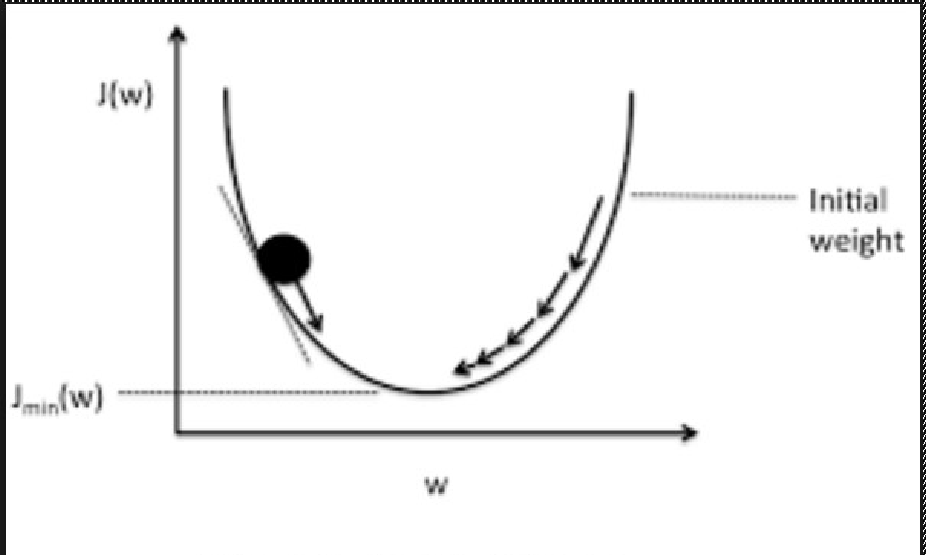
Activation Functions output



Steepest descent method

- Gradient descent tries to find such a minimum x by using information from the first derivative of f: It simply follows the steepest descent from the current point.
- This is like rolling a ball down the graph until it comes to rest (while neglecting inertia).

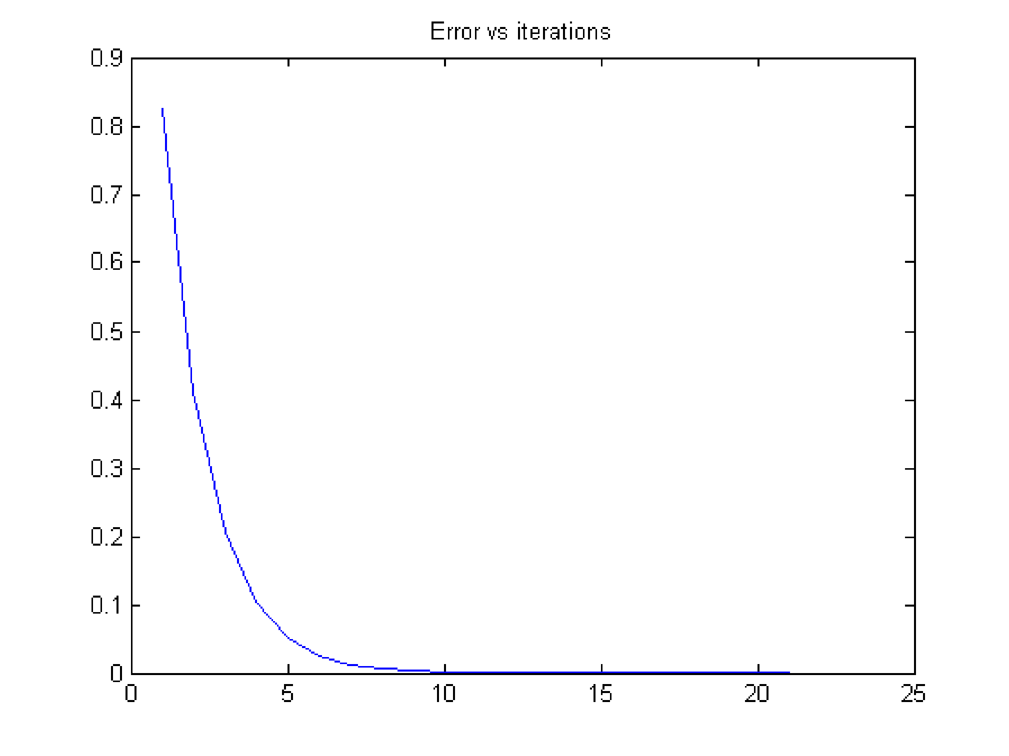
```
x =

     0
     1
     2


W =

    0.4153
    1.2689
    0.8656


y =

    3.0000
```



Schematic of gradient descent.

Newton's method

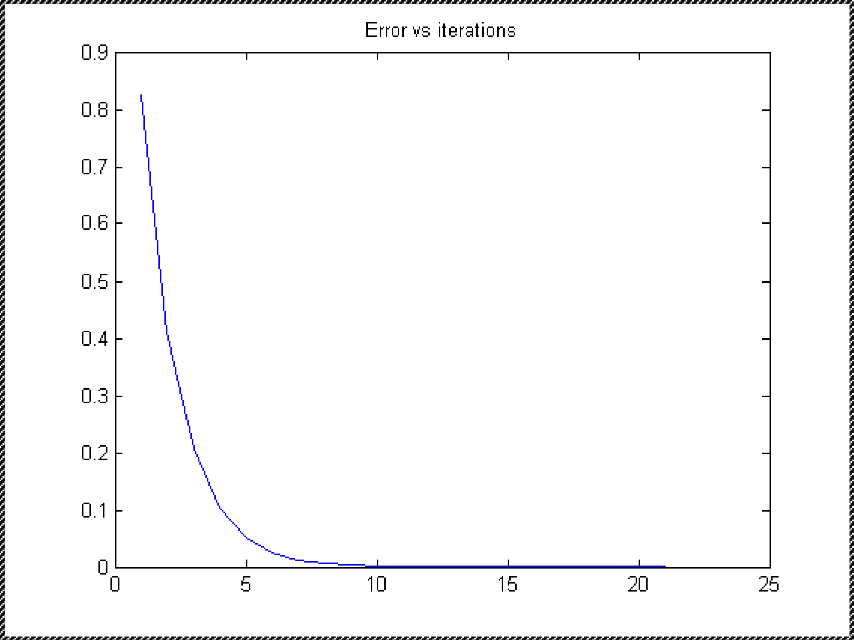
```
x =

     0
     1
     2


W =

    0.0000
    0.6000
    1.2000


y =

    3.0000
```



LMS Method

- The LMS algorithm is based on the idea of gradient descent to search for the optimal (minimum error) condition, with a cost function equal to the mean squared error at the filter output.

```
x =

     0
     1
     2


W =

    0.3098
    0.5006
    1.2497


y =

    3.0000
```



Rosenblatt Algorithm

- Rosenblatt Algorithm states that if the patterns (vectors) used to train the perceptron are drawn from two linearly separable classes, then the perceptron algorithm converges and positions the decision surface in the form of a hyperplane between the two classes.

Rosenblatt Algorithm contd..

- Let $w(1)$ be any initial choice of weight vector and $x(n)$ be any sequence in (C1 U C2) 1.
- If the nth member of the training set, $x(n)$, is correctly classified by the weight vector $w(n)$ computed at the nth iteration of the algorithm, no correction is made to the weight vector of the perceptron in accordance with the rule:
- $w(n + 1) = w(n)$ if $w^T x(n) \geq 0$ and $x(n)$ belongs to class C1
- $w(n + 1) = w(n)$ if $w^T x(n) < 0$ and $x(n)$ belongs to class C2

Rosenblatt Algorithm contd..

2. Otherwise, the weight vector of the perceptron is updated in accordance with the rule

$w(n + 1) = w(n) - \eta(n)x(n)$ if $w^Tx(n) \geq 0$ and $x(n)$ belongs to class C_2

$w(n + 1) = w(n) + \eta(n)x(n)$ if $w^Tx(n) < 0$ and $x(n)$ belongs to class C_1

where the learning-rate parameter $\eta(n)$ controls the adjustment applied to the weight vector at iteration n.

Applications Of Rosenblatts Algorithm

```
W =

  -1.5000
   1.0000
   0.5000
```

```
W =

  -0.5000
   0.5000
   0.5000
```

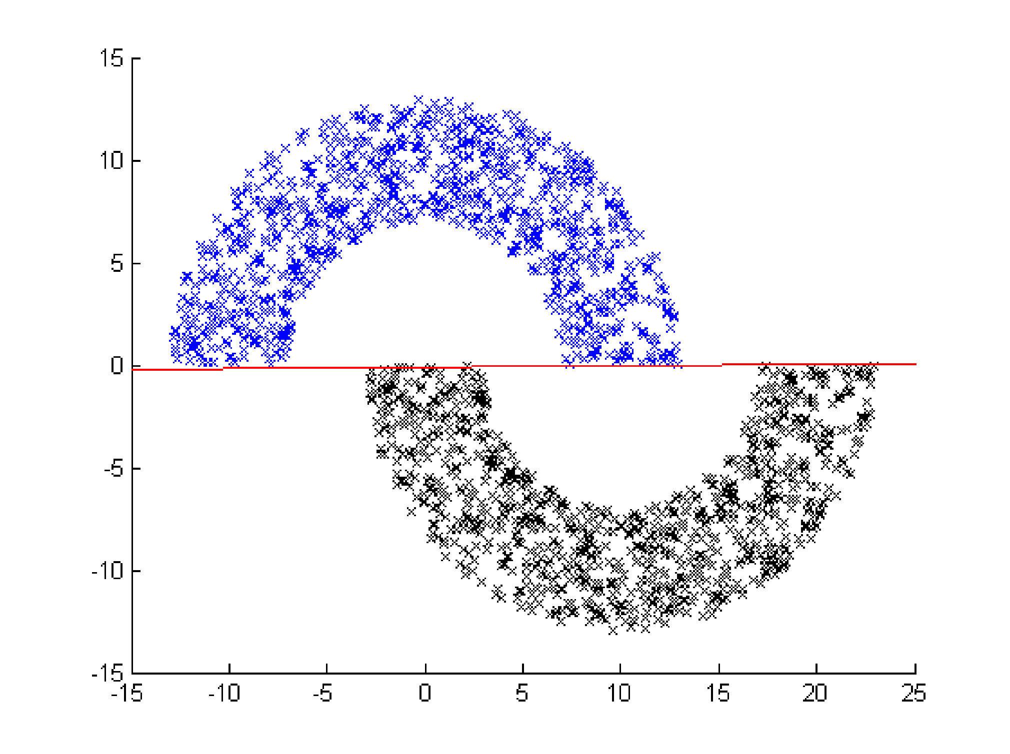
```
W =

        0
  -0.5000
```

AND Gate

OR Gate

NOT Gate

Applications Of Rosenblatts Algorithm



Back Propagation Algorithm

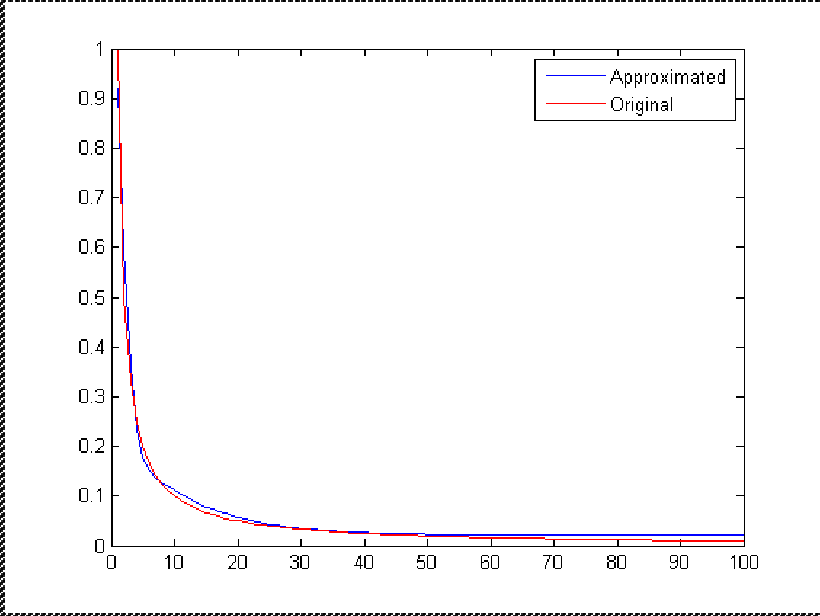
- The backward propagation algorithm is a method of training artificial neural networks and used in conjunction with an optimization method such as gradient descent.
- The algorithm repeats a two phase cycle, propagation and weight update. When an input vector is presented to the network, it is propagated forward through the network, layer by layer, until it reaches the output layer.

BPA contd..

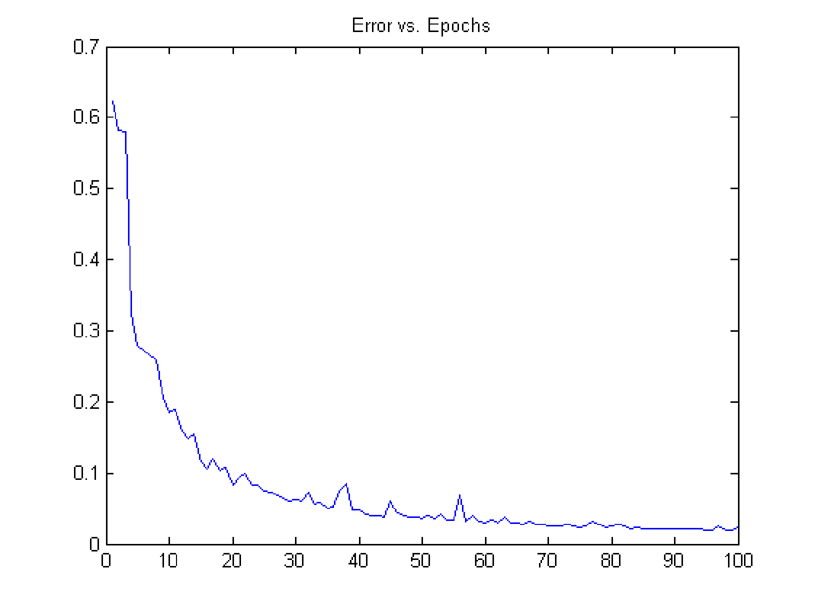
- The output of the network is then compared to the desired output, using an error function, and an error value is calculated for each of the neurons in the output layer. The error values are then propagated backwards, starting from the output, until each neuron has an associated error value which roughly represents its contribution to the original output. The error values are used to calculate the gradient of the loss function with respect to the weights in the network. In the second phase, this gradient is fed to the optimization method, which in turn uses it to update the weights, in an attempt to minimize the loss function.

Function Approximation Using BPA

Function Approximation (y = 1/x)

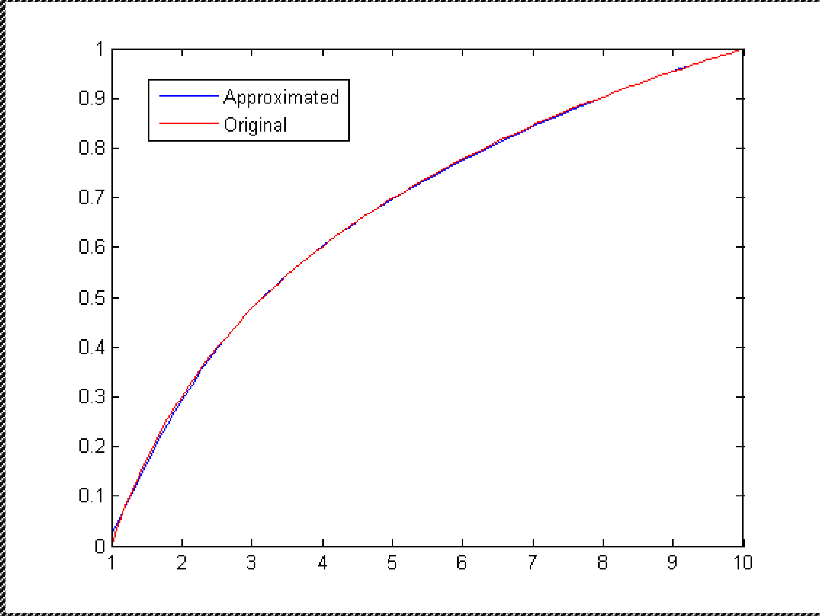


Error vs Epochs

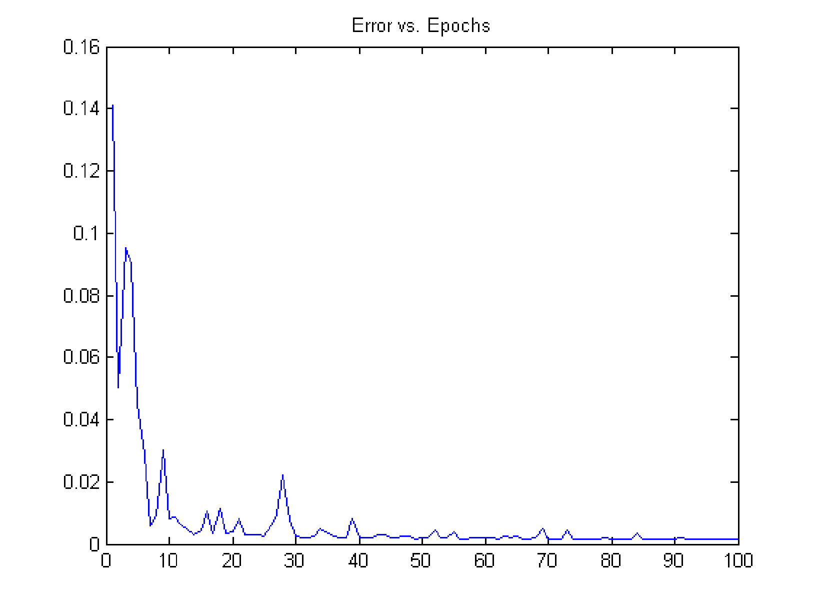


Function Approximation Using BPA

Function Approximation (y = ln x)

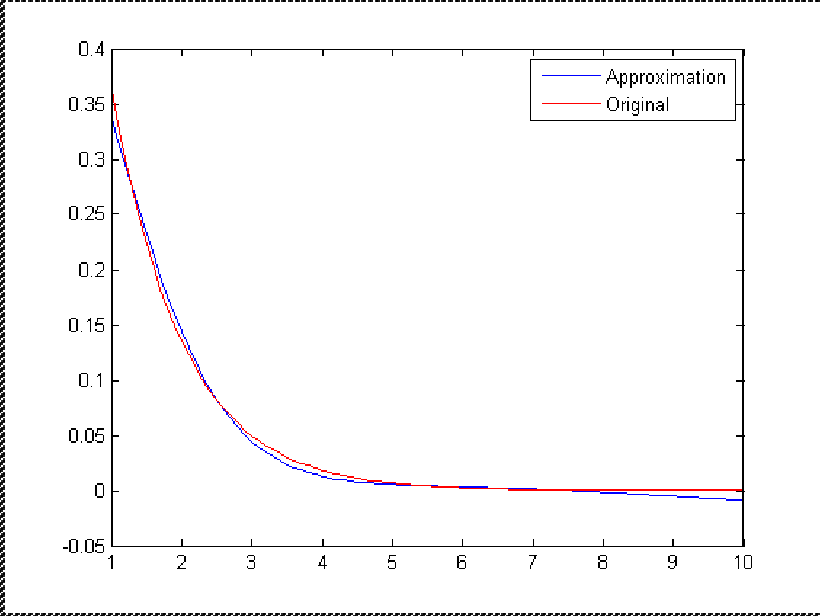


Error vs Epochs

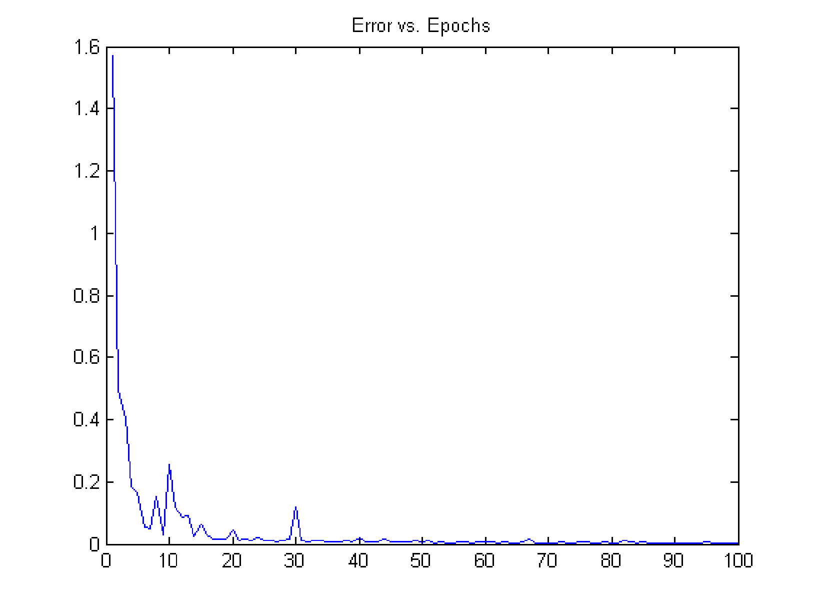


Function Approximation Using BPA

Function Approximation (y = e^-x)

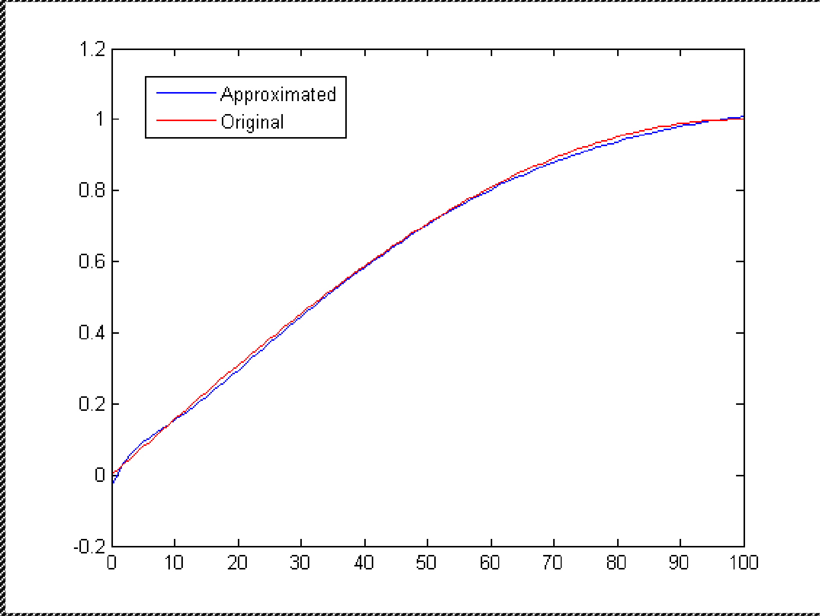


Error vs Epochs

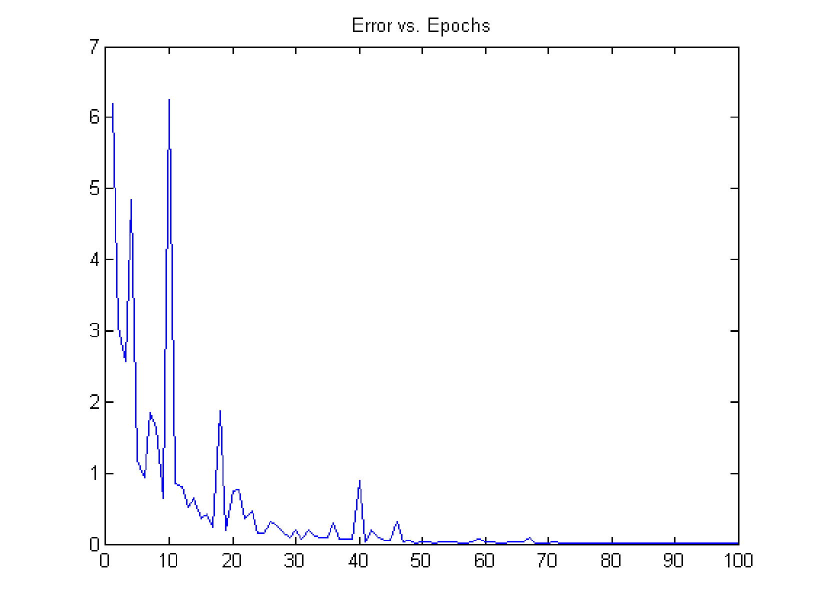


Function Approximation Using BPA

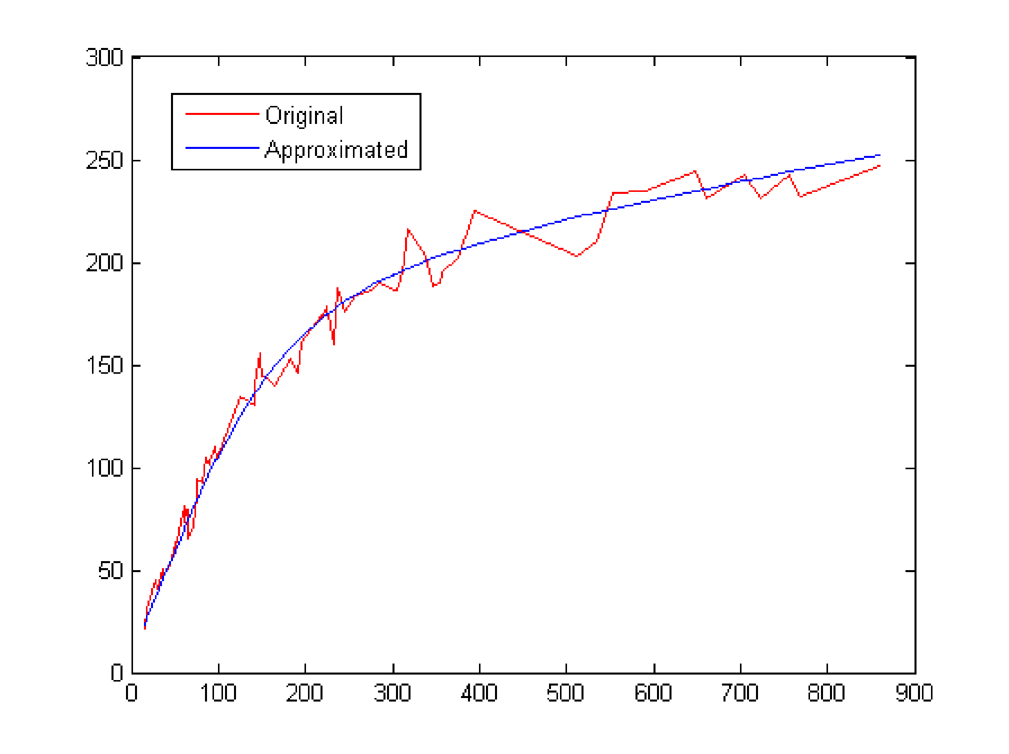
Function Approximation (y = sin x)



Error vs Epochs

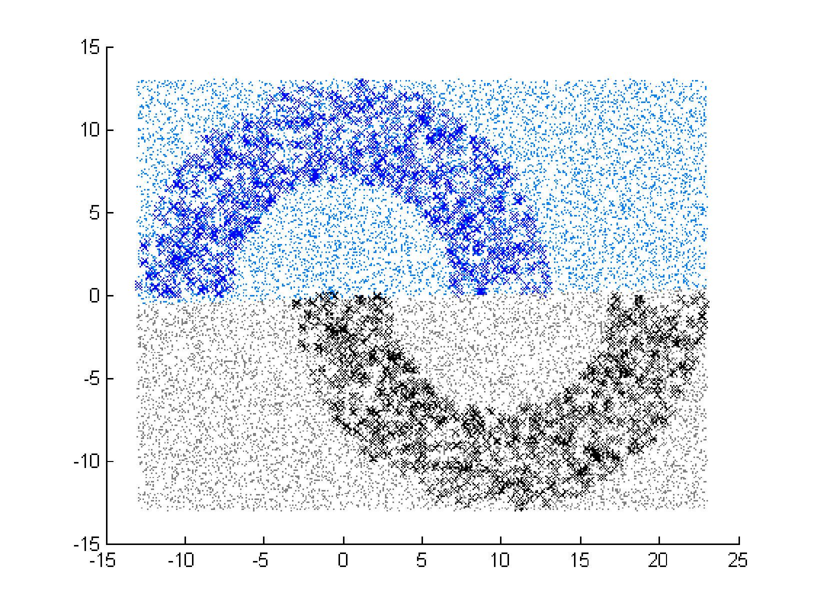


Time Series Prediction Using BPA

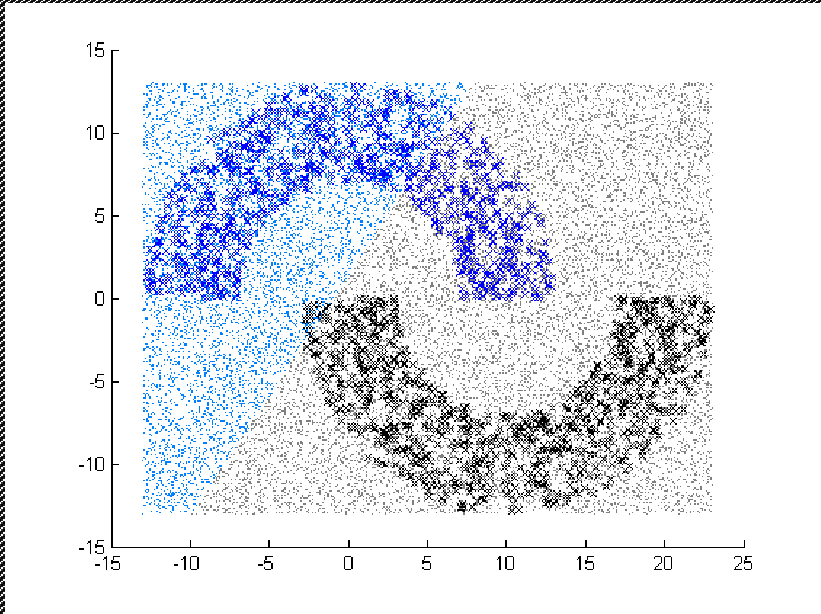


Double Moon Classification Using BPA

Normal



Worst Case



Extreme Learning Machine(ELM) and Online Sequential ELM (OSELM)

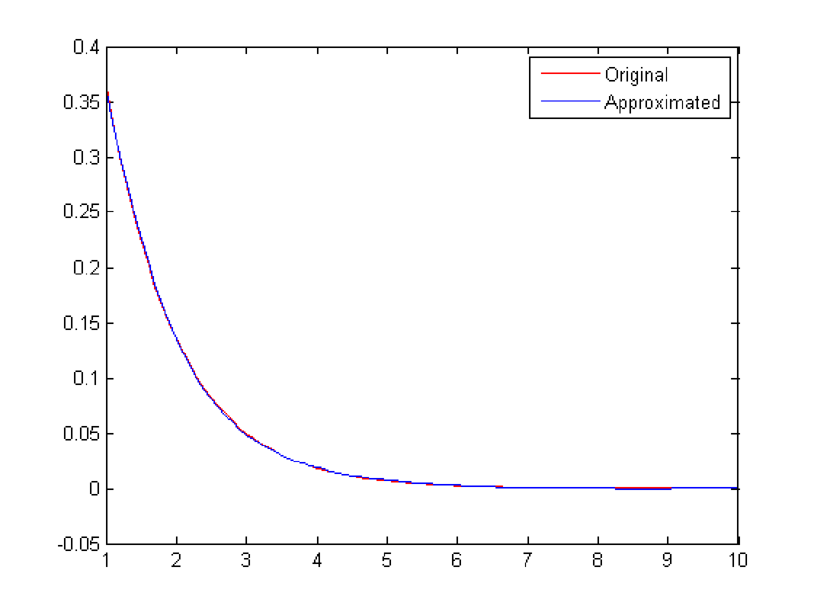
- The name "extreme learning machine (ELM) was given by Guang Bin Huang.
- Extreme learning machines are feedforward neural network for classification or regression with a single layer of hidden nodes, where the weights connecting inputs to hidden nodes are randomly assigned and never updated.
- The weights between hidden nodes and outputs are learned in a single step, which essentially amounts to learning a linear model.

Algorithm for ELM

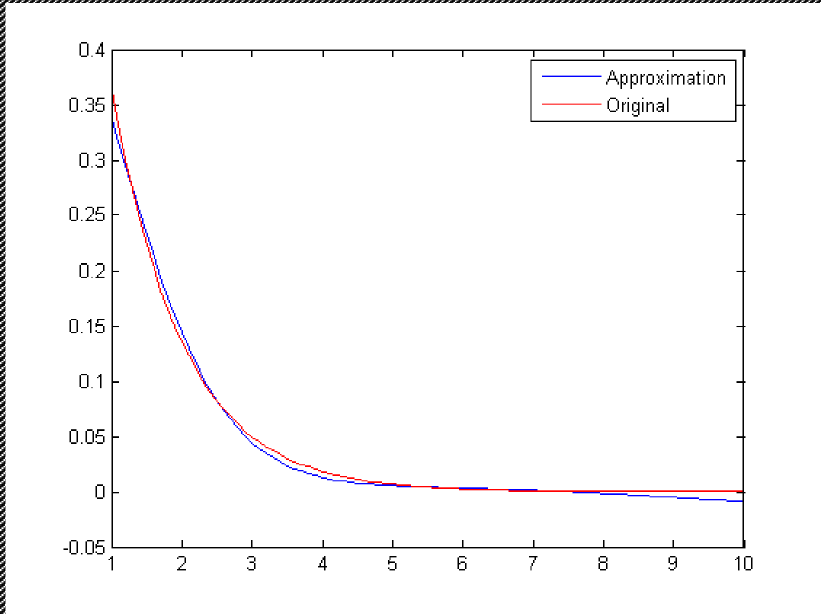
- The algorithm for ELM is as follows,
- Step 1: Assign input weights W1 and hidden nodes bias with random values.
- Step 2: Compute the error function 'J' and find the least square solution 'W' which gives $W = (Y_d Y_1^T)(Y_1 Y_1^T)^{-1}$
- When the input data is very large, the computational cost increases for ELM and therefore OSELM is used which trains network one-by-one in a sequential manner since ELM is batch learning in nature.

ELM vs BPA

ELM (y = e^-x)

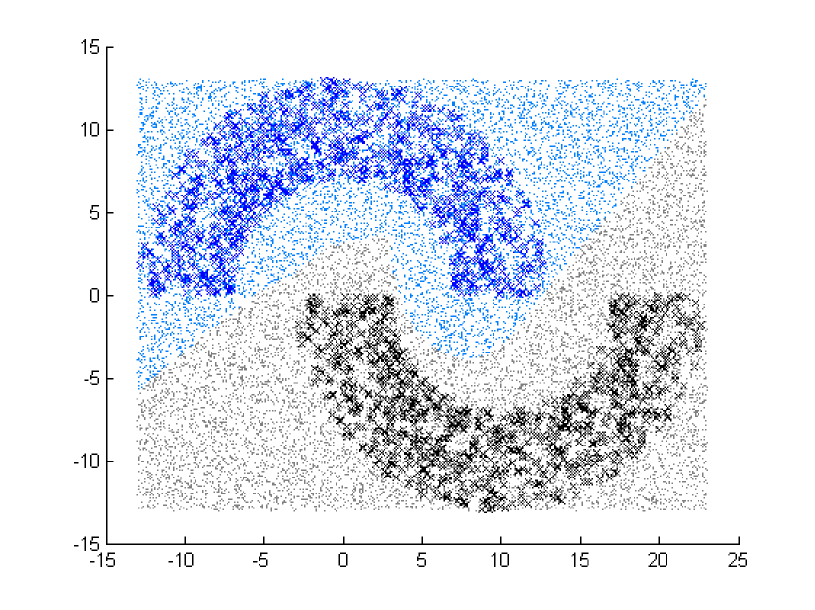


BPA (y = e^-x)

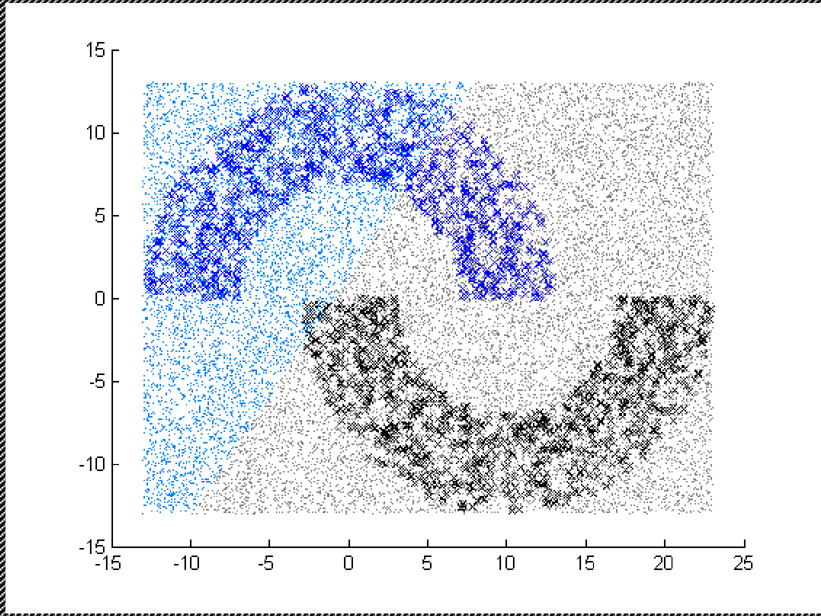


ELM vs BPA

ELM



BPA (worst case)



OSELM Algorithm

- The Algorithm for OSELM is as follows,
- Step 1: Assign the input weight and bias randomly within the range [-1,1].
- Step 2: Calculate the initial hidden layer output matrix Y0.
- Step 3: Estimate the initial output weight

$$W_0 = P_0 H_0^T Y_0, \text{where } P_0 = (H_0^T H_0)^{-1} \text{ and } Y_0 = \{y_1, .., y_{N0}\}^T$$

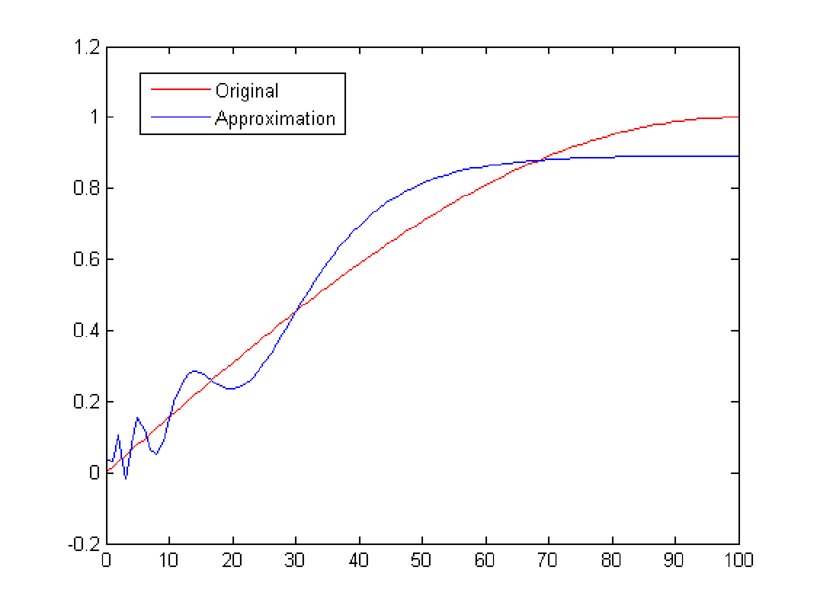
- Step 4: Relabel the subsequent data and find the weight vector for kth chunk of data which gives expressions as follows, for k = 1,2,..

$$P_k = P_{k-1} - P_{k-1} Y_k^T (I + Y_k P_{k-1} Y_k^T)^{-1} Y_k P_{k-1}$$

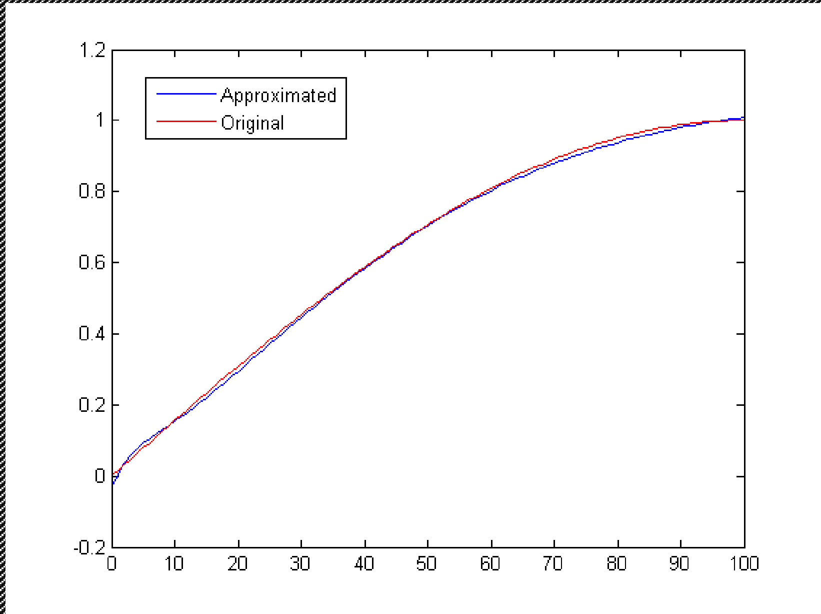
$$W_k = W_{k-1} + P_k Y_k^T (Y_d - Y_k W_{k-1})$$

OSELM vs BPA

OSELM (y = sin x, worst case)

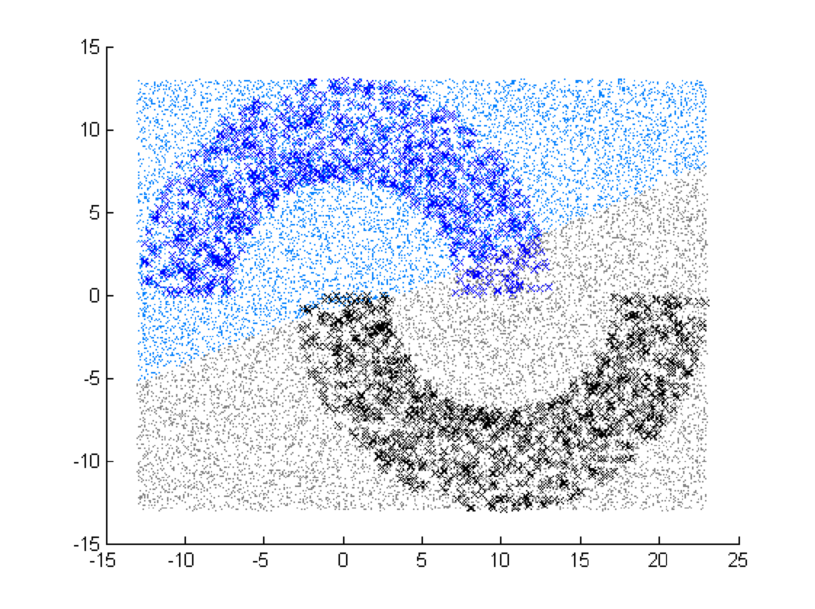


BPA (y = sin x)

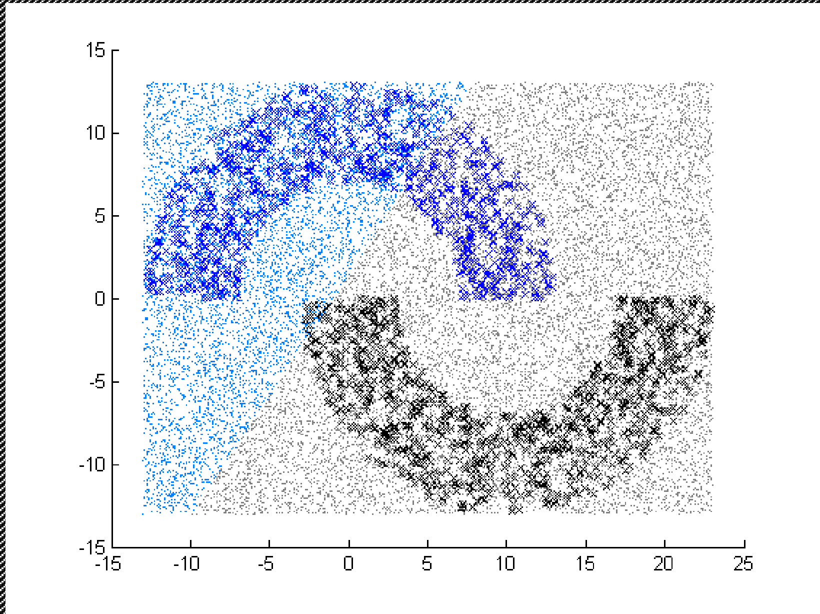


OSELM vs BPA

OSELM (worst case)



BPA (worst case)



Comparison between BPA and ELM

- 1.ELM needs much less training time compared to popular BPA.
- 2. The prediction accuracy of ELM is usually slightly better than BPA in many applications.
- 3. Compared with BPA, ELM can be implemented easily since there is no parameter to be tuned except an insensitive parameter L.
- 4. It should be noted that many nonlinear activation functions can be used in ELM. ELM needs more hidden nodes than BP which implies that ELM and BP have much shorter response time to unknown data.
- 5. Number of epochs in BPA are more compared to ELM.
- 6. BPA is applicable to any number of layers whereas ELM can be applied to only 3-layered network.

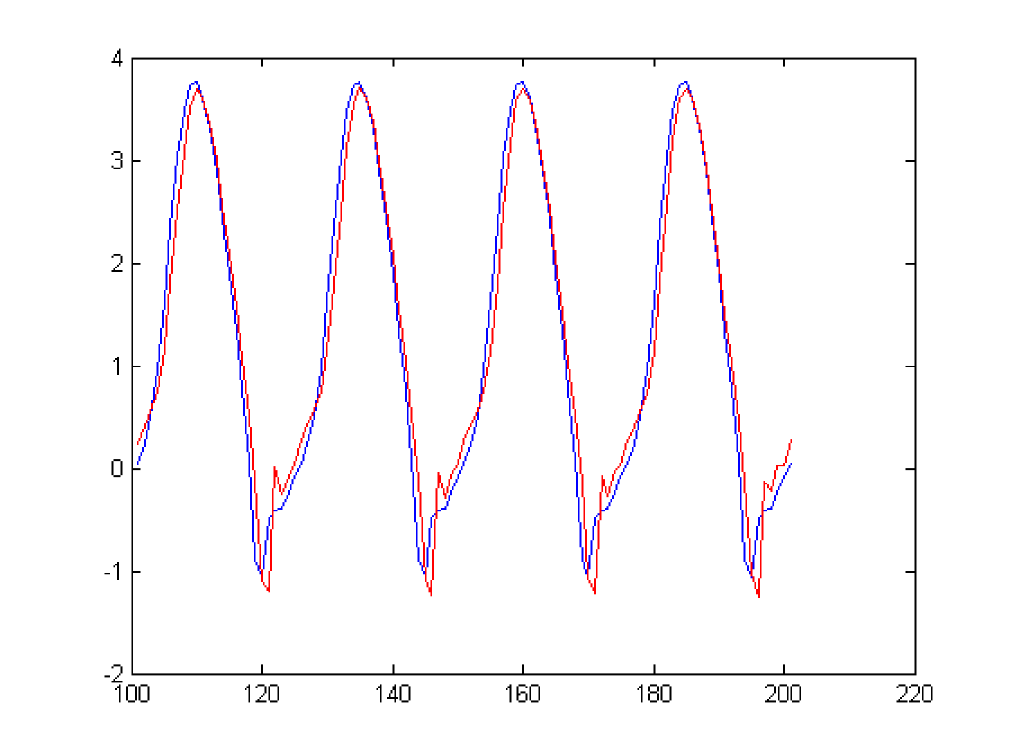
Applications - BPA

- Backpropagation Algorithm
- a. Classification - In this classification problem, the goal is to identify whether a certain "data point" belongs to Class 1, 2, or 3. Random points are assigned to a certain class, and the BPA network is trained to find the pattern. When training is complete, it will use what it has learned to accurately classify new points.
- b. Function Approximation - In this problem, the network tries to approximate the value of a certain function. It is fed with noisy data, and the goal is to find the true pattern. After training, the network successfully estimates the value of the function(function maybe sinc function gaussian function etc..).
- c. Time-series prediction - In this problem, the goal is to design a BPA network to predict a value based on a given time-series data (i.e. stock market prediction based on given trends). To approach this problem, the inputs to the neural network have to be refactored in chunks, and the resulting output will be the next data item directly following that chunk .

Applications - ELM

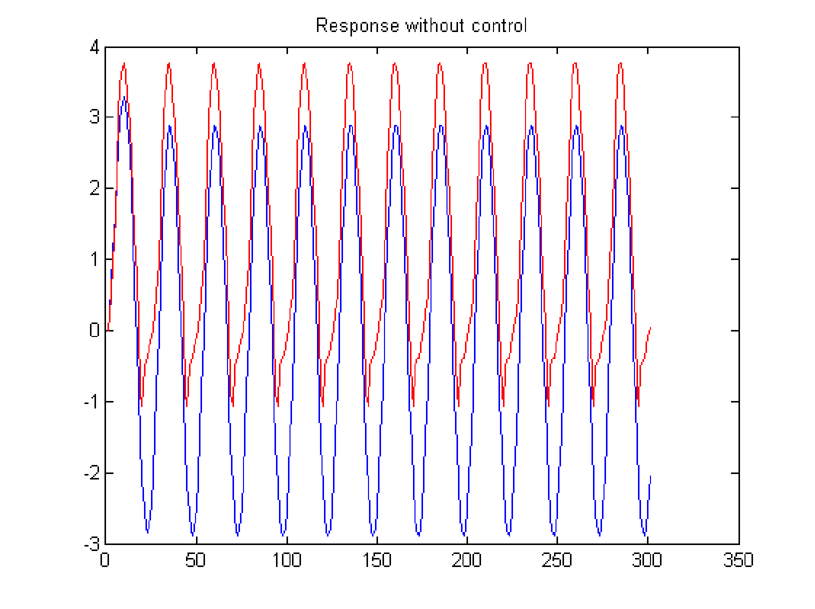
- Extreme Learning Machine
- a. The ELM is used to solve the function approximation problem like that of back propagation algorithm.
- b. ELM is used to solve the classification problems.
- c. ELM is used to solve the Real World Regression problems.
- d. ELM is used to solve Real-World Very Large Complex Applications.
- e. ELM is used to solve Real Medical Diagnosis Application for example Diabetes.

System Identification using BPA



System Control using Neural Network

Without Control



With Control