Introduction to Extreme Learning Machines

Guang-Bin HUANG

Assistant Professor
School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore

Hands-on Workshop on Machine Learning for BioMedical Informatics 2006, National University of Singapore
21 Nov 2006

Outline

Outline

Outline

Outline

Feedforward Neural Networks with Additive Nodes

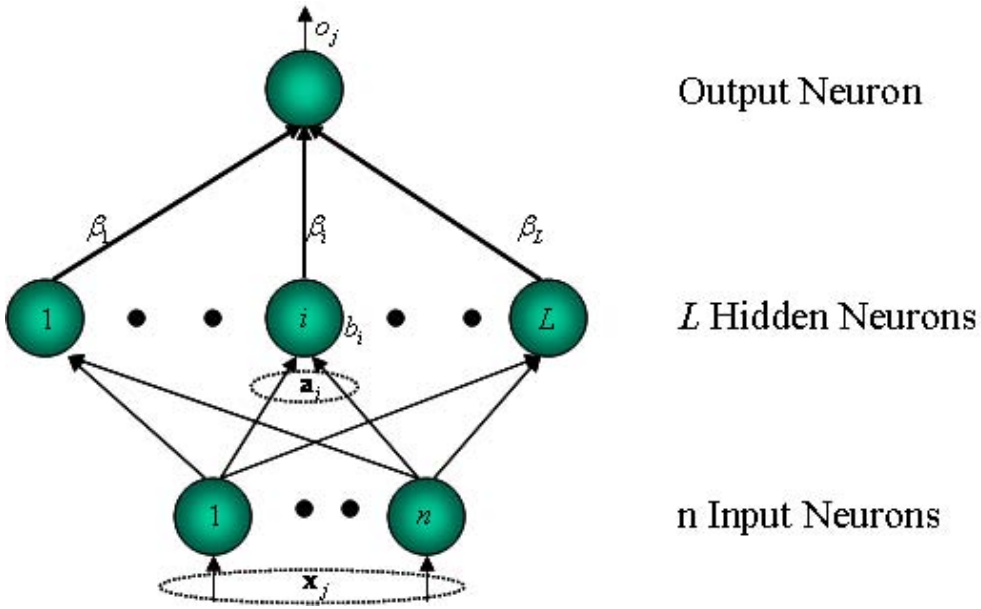


Figure 1: Feedforward Network Architecture: additive hidden nodes

Output of hidden nodes

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i) \quad (1)$$

$\mathbf{a}_i$: the weight vector connecting the ith hidden node and the input nodes.
b_i: the threshold of the ith hidden node.

Output of SLFNs

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \quad (2)$$

β_i: the weight vector connecting the ith hidden node and the output nodes.

Feedforward Neural Networks with Additive Nodes

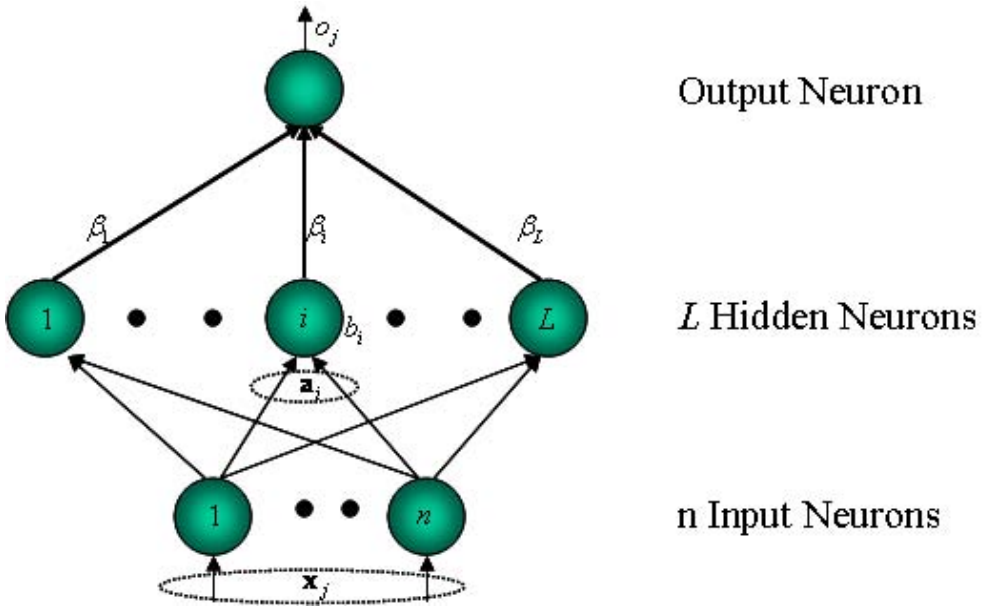


Figure 1: Feedforward Network Architecture: additive hidden nodes

Output of hidden nodes

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i) \quad (1)$$

$\mathbf{a}_i$: the weight vector connecting the ith hidden node and the input nodes.
b_i: the threshold of the ith hidden node.

Output of SLFNs

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \quad (2)$$

β_i: the weight vector connecting the ith hidden node and the output nodes.

Feedforward Neural Networks with Additive Nodes

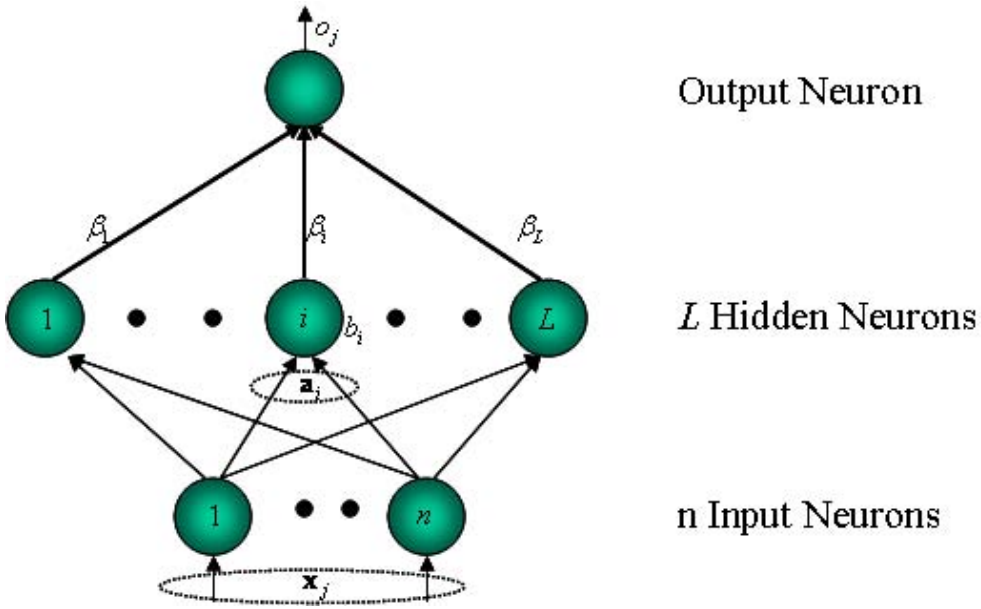


Figure 1: Feedforward Network Architecture: additive hidden nodes

Output of hidden nodes

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i) \quad (1)$$

$\mathbf{a}_i$: the weight vector connecting the ith hidden node and the input nodes.
b_i: the threshold of the ith hidden node.

Output of SLFNs

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \quad (2)$$

β_i: the weight vector connecting the ith hidden node and the output nodes.

Feedforward Neural Networks with Additive Nodes

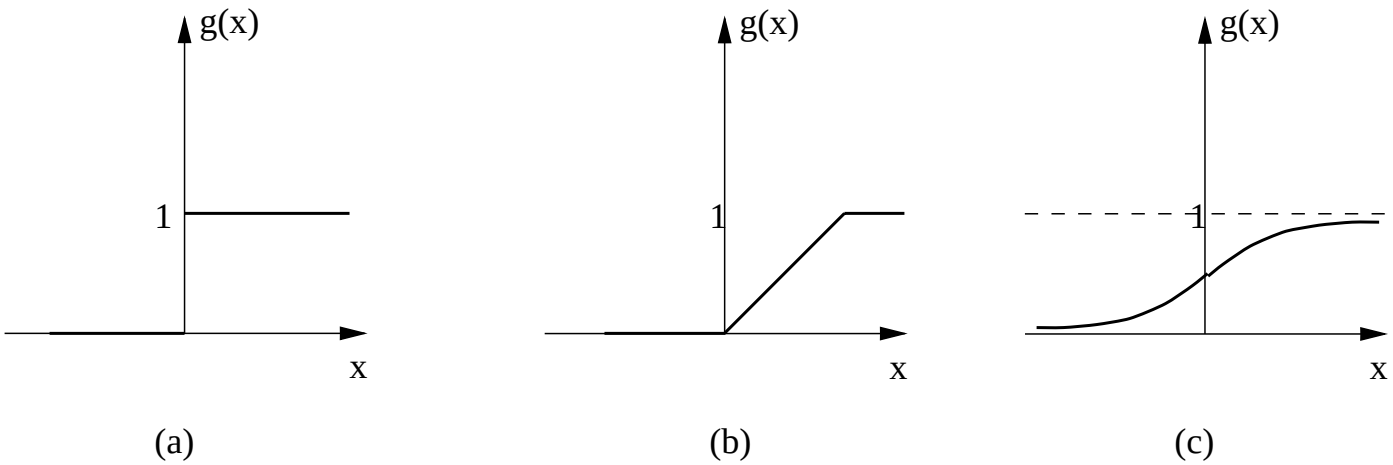


Figure 2: Activation Functions $g(x)$

Feedforward Neural Networks with RBF Nodes

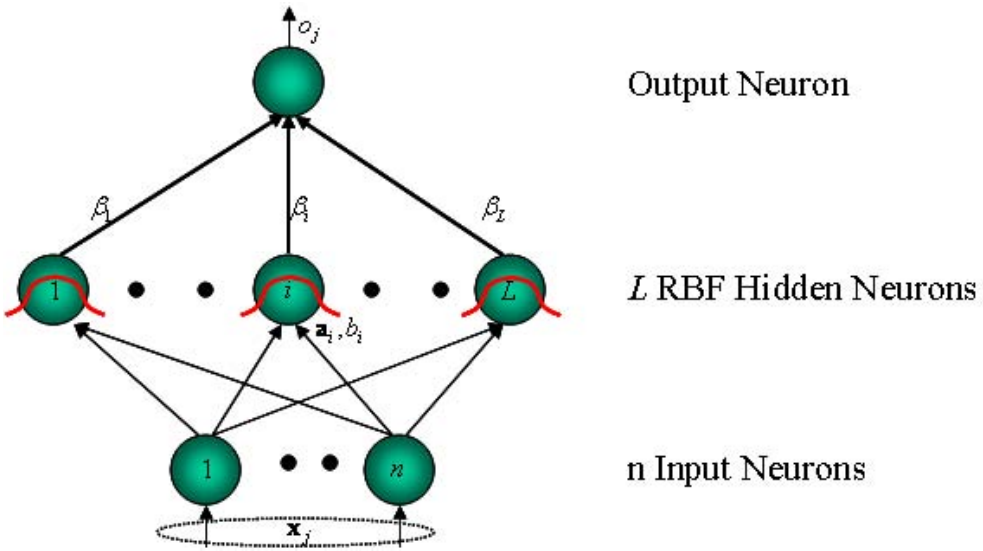


Figure 3: Feedforward Network Architecture: RBF hidden nodes

Output of hidden nodes

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g\left(b_i \|\mathbf{x} - \mathbf{a}_i\|\right) \quad (3)$$

$\mathbf{a}_i$: the center of the ith hidden node.
b_i: the impact factor of the ith hidden node.

Output of SLFNs

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \quad (4)$$

β_i: the weight vector connecting the ith hidden node and the output nodes.

Feedforward Neural Networks with RBF Nodes

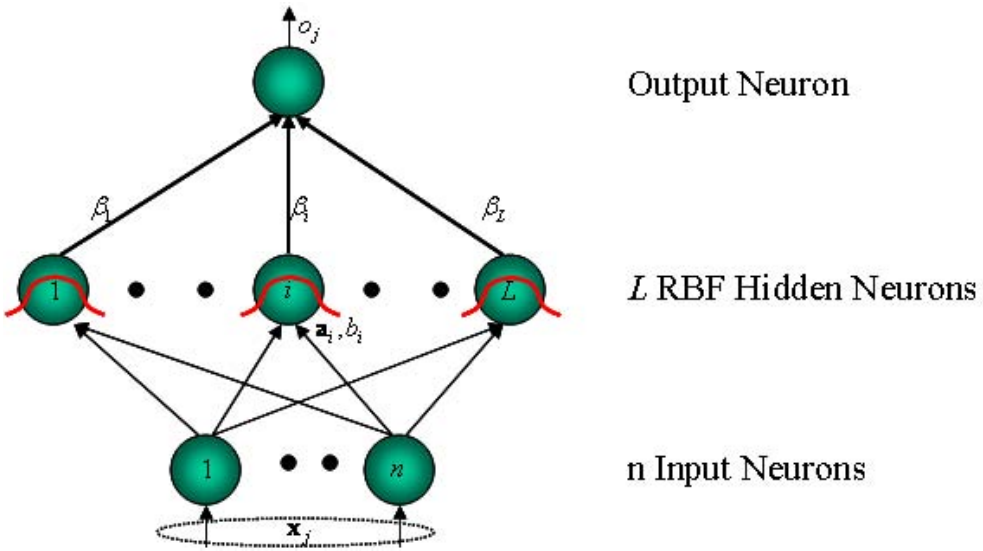


Figure 3: Feedforward Network Architecture: RBF hidden nodes

Output of hidden nodes

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g\left(b_i \|\mathbf{x} - \mathbf{a}_i\|\right) \quad (3)$$

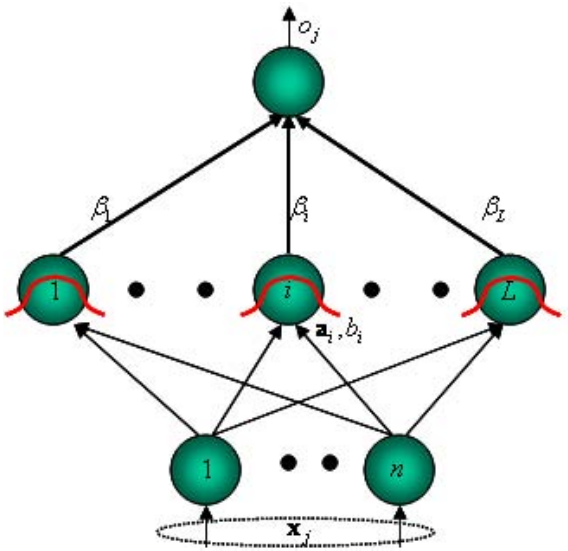
$\mathbf{a}_i$: the center of the ith hidden node.
b_i: the impact factor of the ith hidden node.

Output of SLFNs

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \quad (4)$$

β_i: the weight vector connecting the ith hidden node and the output nodes.

Feedforward Neural Networks with RBF Nodes



Output Neuron

L RBF Hidden Neurons

n Input Neurons

Figure 3: Feedforward Network Architecture: RBF hidden nodes

Output of hidden nodes

$$G(\mathbf{a}_i, b_i, \mathbf{x}) = g\left(b_i \|\mathbf{x} - \mathbf{a}_i\|\right) \quad (3)$$

$\mathbf{a}_i$: the center of the ith hidden node.
b_i: the impact factor of the ith hidden node.

Output of SLFNs

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) \quad (4)$$

β_i: the weight vector connecting the ith hidden node and the output nodes.

Outline

Function Approximation of Neural Networks

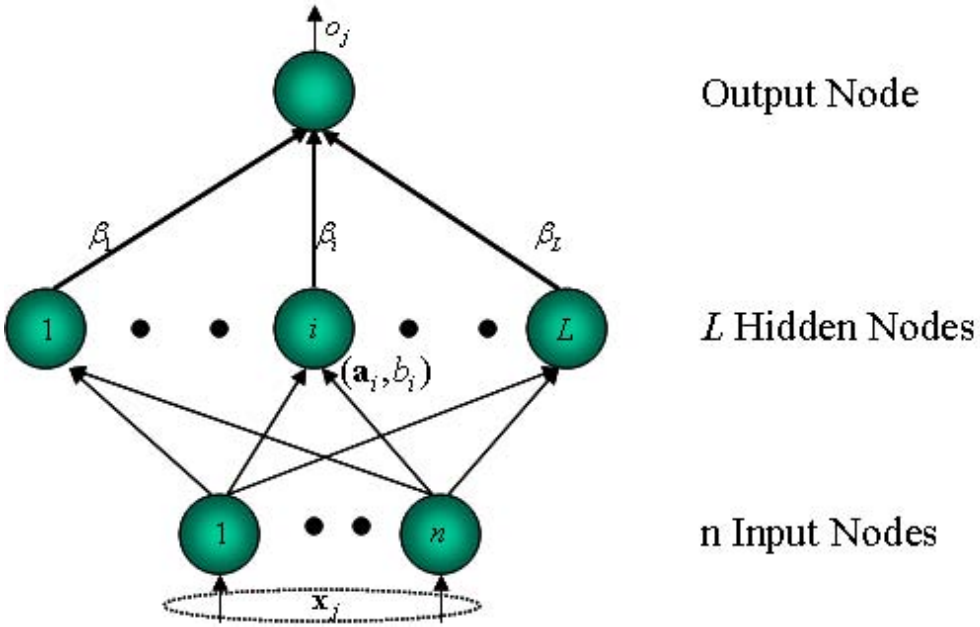


Figure 4: Feedforward Network Architecture

Mathematical Model

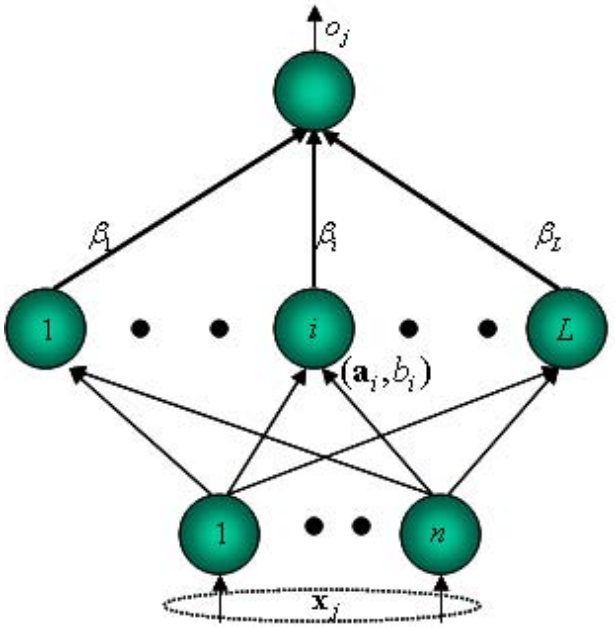
Any continuous target function $f(\mathbf{x})$ can be approximated by SLFNs. In other words, given any small positive value ϵ, for SLFNs with enough number of hidden nodes (L) we have

$$\|f_L(\mathbf{x}) - f(\mathbf{x})\| < \epsilon \quad (5)$$

Learning Issue

In real applications, target function f is usually unknown. One wishes unknown f could be approximated by SLFNs f_L appropriately.

Function Approximation of Neural Networks



Output Node

L Hidden Nodes

n Input Nodes

Figure 4: Feedforward Network Architecture

Mathematical Model

Any continuous target function $f(\mathbf{x})$ can be approximated by SLFNs. In other words, given any small positive value ϵ, for SLFNs with enough number of hidden nodes (L) we have

$$\|f_L(\mathbf{x}) - f(\mathbf{x})\| < \epsilon \quad (5)$$

Learning Issue

In real applications, target function f is usually unknown. One wishes unknown f could be approximated by SLFNs f_L appropriately.

Function Approximation of Neural Networks

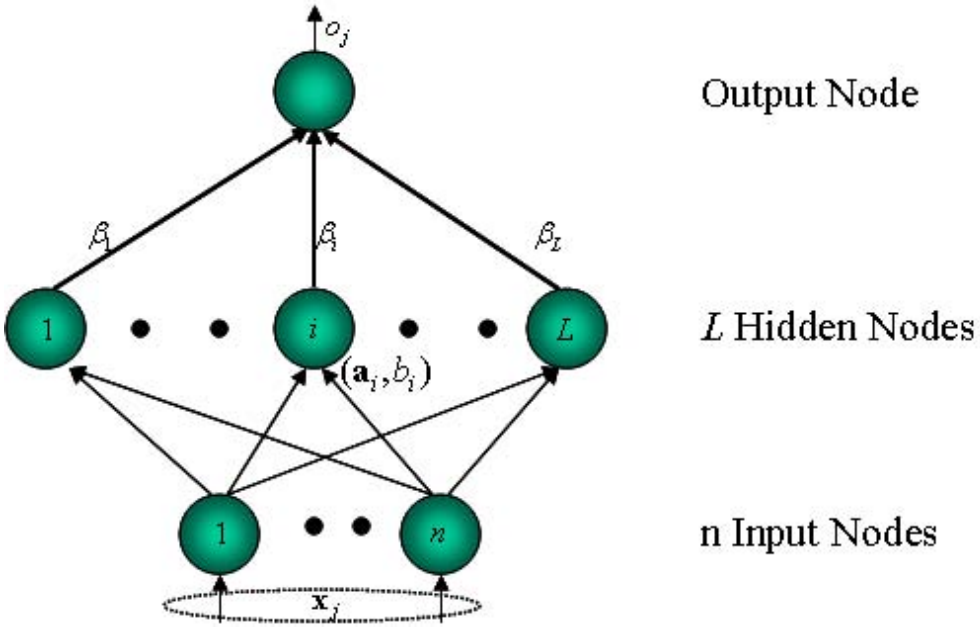


Figure 4: Feedforward Network Architecture

Mathematical Model

Any continuous target function $f(\mathbf{x})$ can be approximated by SLFNs. In other words, given any small positive value ϵ, for SLFNs with enough number of hidden nodes (L) we have

$$\|f_L(\mathbf{x}) - f(\mathbf{x})\| < \epsilon \quad (5)$$

Learning Issue

In real applications, target function f is usually unknown. One wishes unknown f could be approximated by SLFNs f_L appropriately.

Function Approximation of Neural Networks

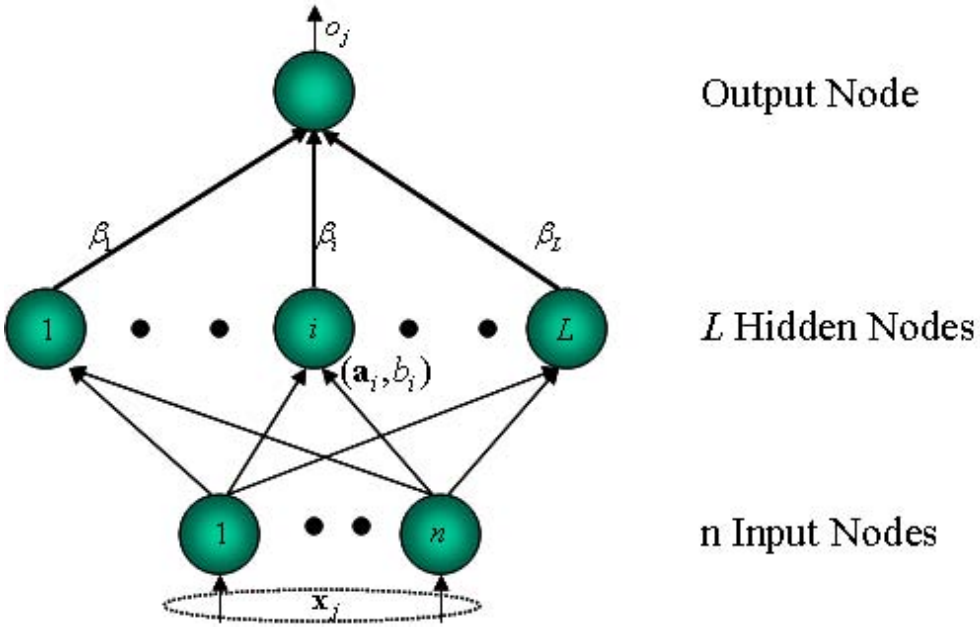


Figure 5: Feedforward Network Architecture

Learning Model

1. For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbb{R}^n \times \mathbb{R}^m$, SLFNs with L hidden nodes and activation function $g(x)$ are mathematically modeled as

$$f_L(\mathbf{x}_j) = \mathbf{o}_j, \quad j = 1, \ldots, N \quad (6)$$

2. Cost function: $E = \sum_{j=1}^{N} \|\mathbf{o}_j - \mathbf{t}_j\|$.
3. Learning is to find $\hat{\beta}_i$, $\hat{\mathbf{a}}_i$, $\hat{b}_i$ such that $E(\hat{\beta}_i, \hat{\mathbf{a}}_i, \hat{b}_i) = \min_{\beta_i, \mathbf{a}_i, b_i} E(\beta_i, \mathbf{a}_i, b_i)$

Function Approximation of Neural Networks

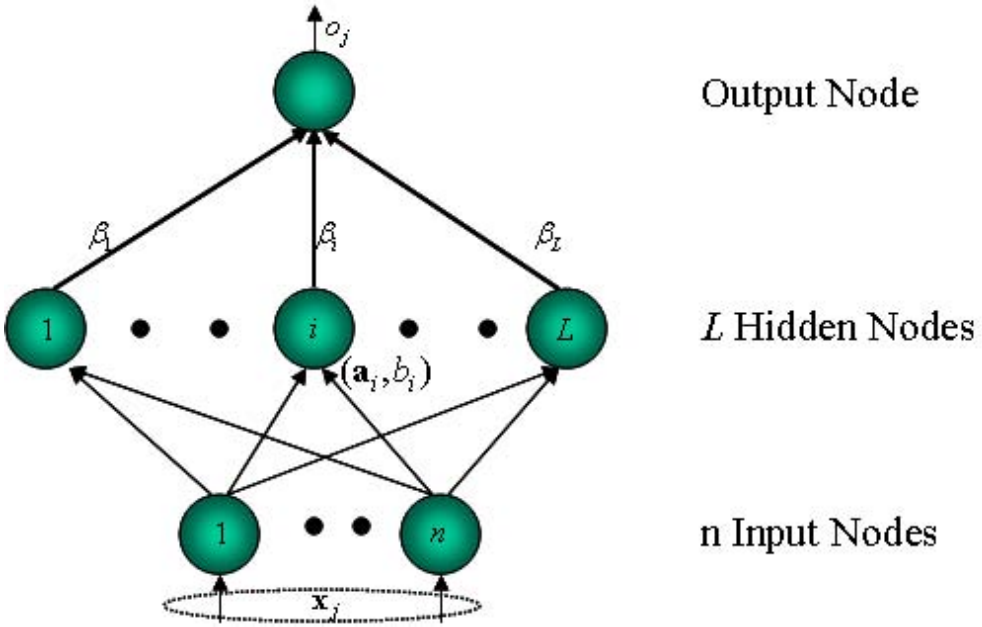


Figure 5: Feedforward Network Architecture

Learning Model

- For N arbitrary distinct samples $(\mathbf{x}_j, \mathbf{t}_j) \in \mathbf{R}^n \times \mathbf{R}^m$, SLFNs with L hidden nodes and activation function $g(x)$ are mathematically modeled as

$$f_L(\mathbf{x}_j) = \mathbf{o}_j, \quad j = 1, \cdots, N \qquad (6)$$

- Cost function: $E = \sum_{j=1}^{N} \left\| \mathbf{o}_j - \mathbf{t}_j \right\|_2$.
- The target is to minimize the cost function E by adjusting the network parameters: β_i, $\mathbf{a}_i$, b_i.

Function Approximation of Neural Networks

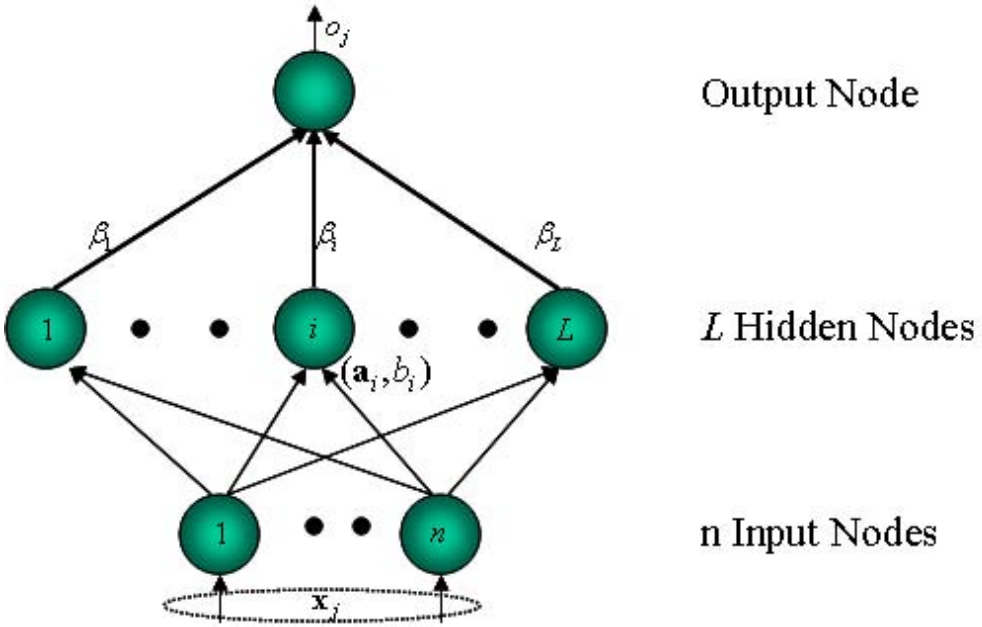


Figure 5: Feedforward Network Architecture

Learning Model

- For N arbitrary distinct samples $(\mathbf{x}_j, \mathbf{t}_j) \in \mathbf{R}^n \times \mathbf{R}^m$, SLFNs with L hidden nodes and activation function $g(x)$ are mathematically modeled as

$$f_L(\mathbf{x}_j) = \mathbf{o}_j, \quad j = 1, \cdots, N \qquad (6)$$

- Cost function: $E = \sum_{j=1}^{N} \left\| \mathbf{o}_j - \mathbf{t}_j \right\|_2$.
- The target is to minimize the cost function E by adjusting the network parameters: β_i, $\mathbf{a}_i$, b_i.

Function Approximation of Neural Networks

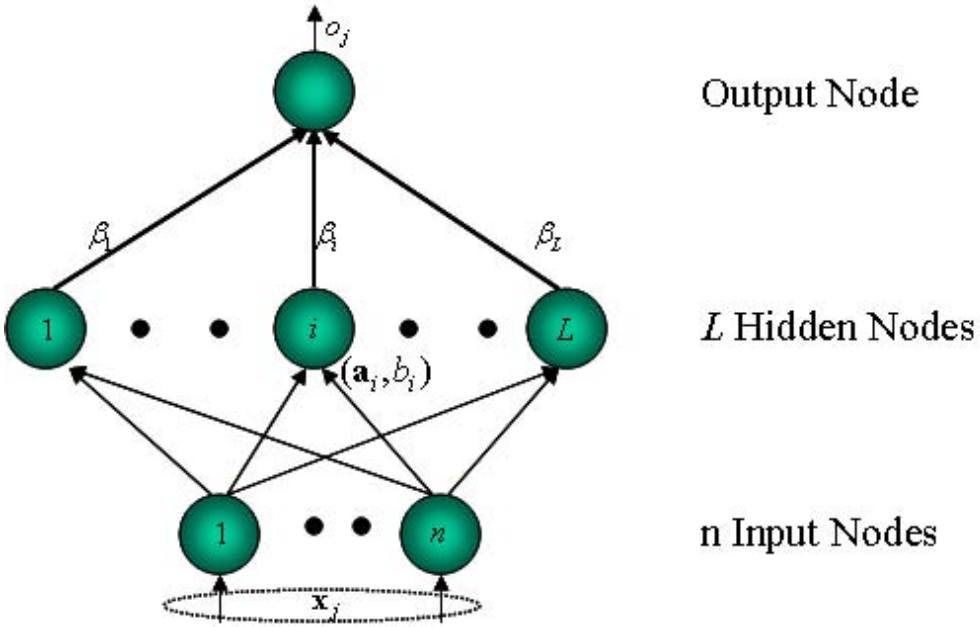


Figure 5: Feedforward Network Architecture

Learning Model

- For N arbitrary distinct samples $(\mathbf{x}_j, \mathbf{t}_j) \in \mathbf{R}^n \times \mathbf{R}^m$, SLFNs with L hidden nodes and activation function $g(x)$ are mathematically modeled as

$$f_L(\mathbf{x}_j) = \mathbf{o}_j, \quad j = 1, \cdots, N \qquad (6)$$

- Cost function: $E = \sum_{j=1}^{N} \left\| \mathbf{o}_j - \mathbf{t}_j \right\|_2$.
- The target is to minimize the cost function E by adjusting the network parameters: β_i, $\mathbf{a}_i$, b_i.

Outline

Learning Algorithms of Neural Networks

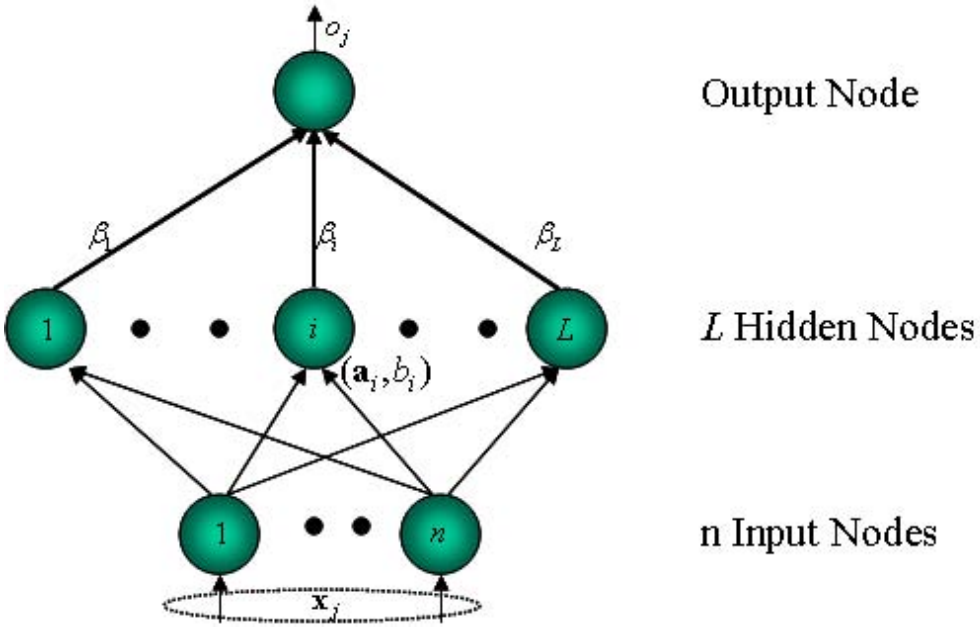


Figure 6: Feedforward Network Architecture

Learning Methods

- Many learning methods mainly based on gradient-descent/iterative approaches have been developed over the past two decades.
- Back-Propagation (BP) and its variants are most popular.

Learning Algorithms of Neural Networks

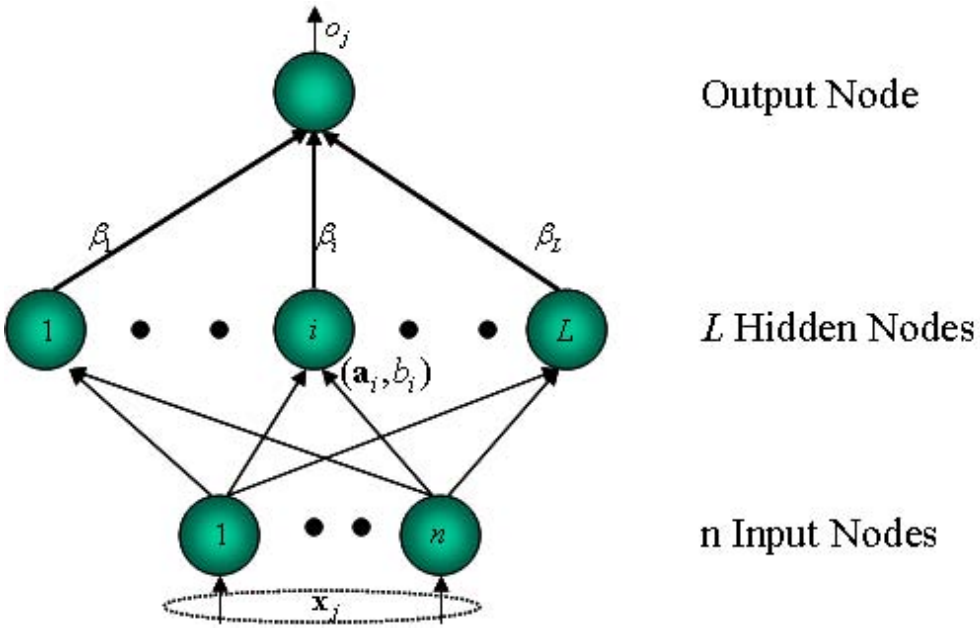


Figure 6: Feedforward Network Architecture

Learning Methods

- Many learning methods mainly based on gradient-descent/iterative approaches have been developed over the past two decades.
- Back-Propagation (BP) and its variants are most popular.

Learning Algorithms of Neural Networks

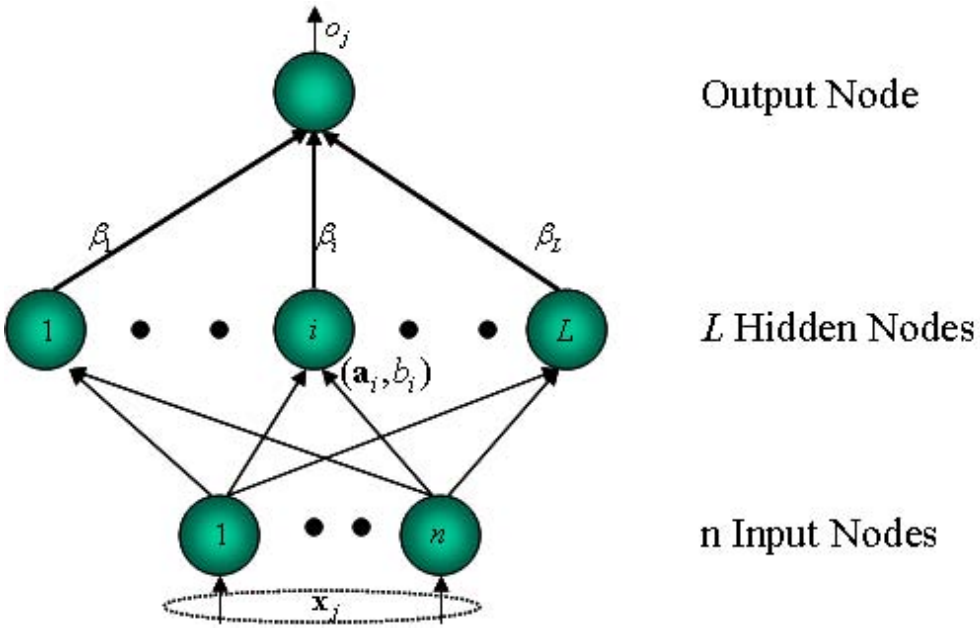


Figure 6: Feedforward Network Architecture

Learning Methods

- Many learning methods mainly based on gradient-descent/iterative approaches have been developed over the past two decades.
- Back-Propagation (BP) and its variants are most popular.

Advantagnes and Disadvantages

Popularity

- Widely used in various applications: regression, classification, etc.

Limitations

- Usually different learning algorithms used in different SLFNs architectures.
- Some parameters have to be tuned mannually.
- Overfitting. (animation: www.ntu.edu.sg/home/egbhuang/NUS-Workshop/generalization.exe)
- Local minima.
- Time-consuming.

Advantagnes and Disadvantages

Popularity

- Widely used in various applications: regression, classification, etc.

Limitations

- Usually different learning algorithms used in different SLFNs architectures.
- Some parameters have to be tuned mannually.
- Overfitting. (animation: www.ntu.edu.sg/home/egbhuang/NUS-Workshop/generalization.exe)
- Local minima.
- Time-consuming.

Generalization capability of SLFNs

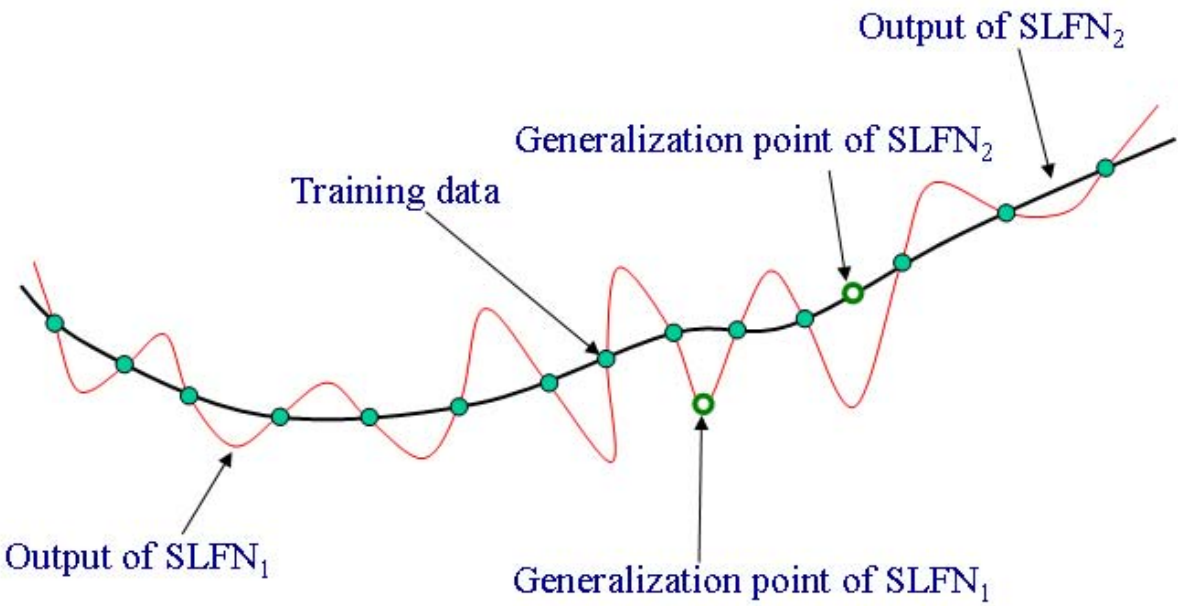


Figure 7: SLFN$_1$ has poor generalization while SLFN$_2$ has good generalization

Local minima issue of conventional learning methods

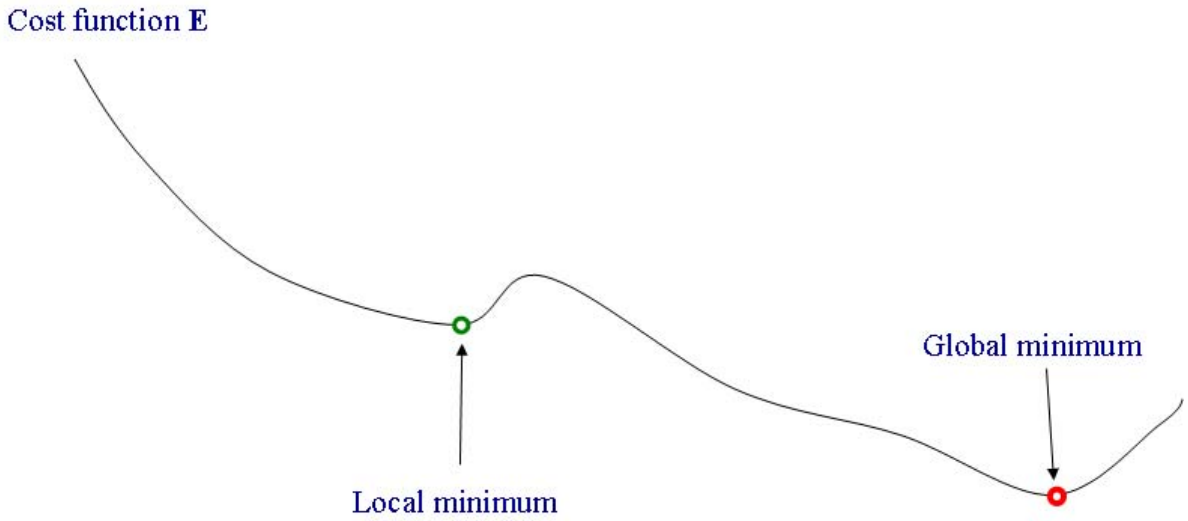
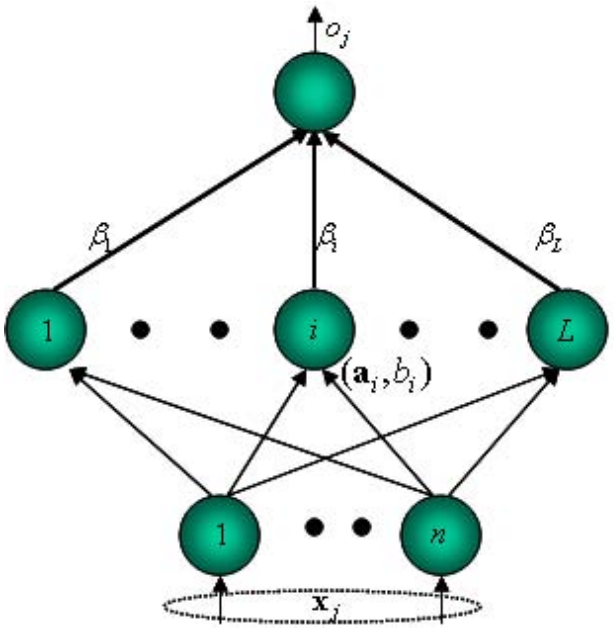


Figure 8: Conventional SLFN learning methods usually stuck Local minima

Outline

Extreme Learning Machine (ELM)



Output Node

L Hidden Nodes

n Input Nodes

New Learning Theory

Given any bounded nonconstant piecewise continuous function g (integrable for RBF nodes), for any continuous target function f and any randomly generated sequence $\{(\mathbf{a}_L, b_L)\}$,

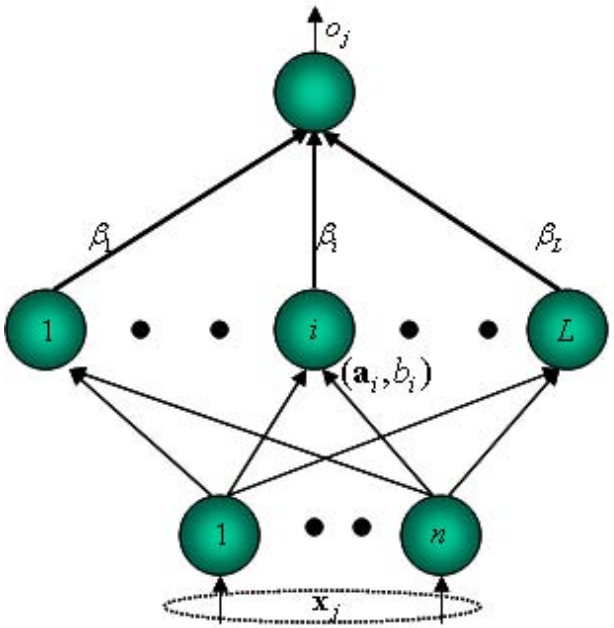
$$\lim_{L \to \infty} \|f(\mathbf{x}) - f_L(\mathbf{x})\| = 0$$

holds with probability one if β_i is chosen to minimize the $\|f(\mathbf{x}) - f_L(\mathbf{x})\|$, $i = 1, \cdots, L$.

Figure 9: Feedforward Network Architecture

G.-B. Huang, et al., "Universal Approximation Using Incremental Networks with Random Hidden Nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.

Extreme Learning Machine (ELM)



Output Node

L Hidden Nodes

n Input Nodes

New Learning Theory

Given any bounded nonconstant piecewise continuous function g (integrable for RBF nodes), for any continuous target function f and any randomly generated sequence $\{(\mathbf{a}_L, b_L)\}$,

$$\lim_{L \to \infty} \|f(\mathbf{x}) - f_L(\mathbf{x})\| = 0$$

holds with probability one if β_i is chosen to minimize the $\|f(\mathbf{x}) - f_L(\mathbf{x})\|$, $i = 1, \cdots, L$.

Figure 9: Feedforward Network Architecture

G.-B. Huang, et al., "Universal Approximation Using Incremental Networks with Random Hidden Nodes," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.

Unified Learning Platform

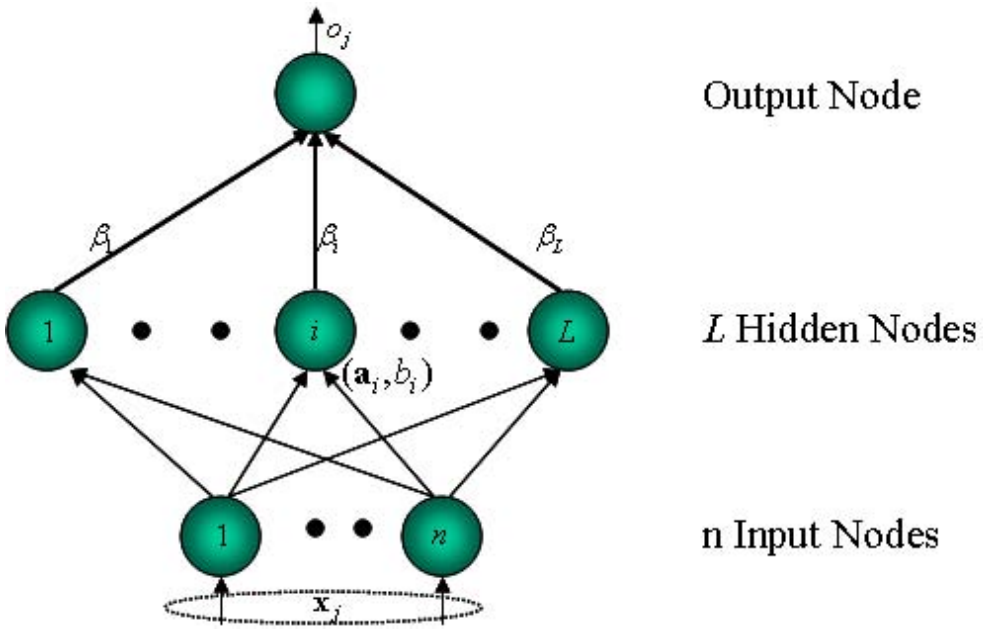


Figure 10: Feedforward Network Architecture

Mathematical Model

- For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbf{R}^n \times \mathbf{R}^m$, standard SLFNs with L hidden nodes and activation function $g(x)$ are mathematically modeled as

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \quad j = 1, \cdots, N \tag{7}$$

- $\mathbf{a}_i$: the input weight vector connecting the ith hidden node and the input nodes or the center of the ith hidden node.
- β_i: the weight vector connecting the ith hidden node and the output node.
- b_i: the threshold or impact factor of the ith hidden node.

Unified Learning Platform

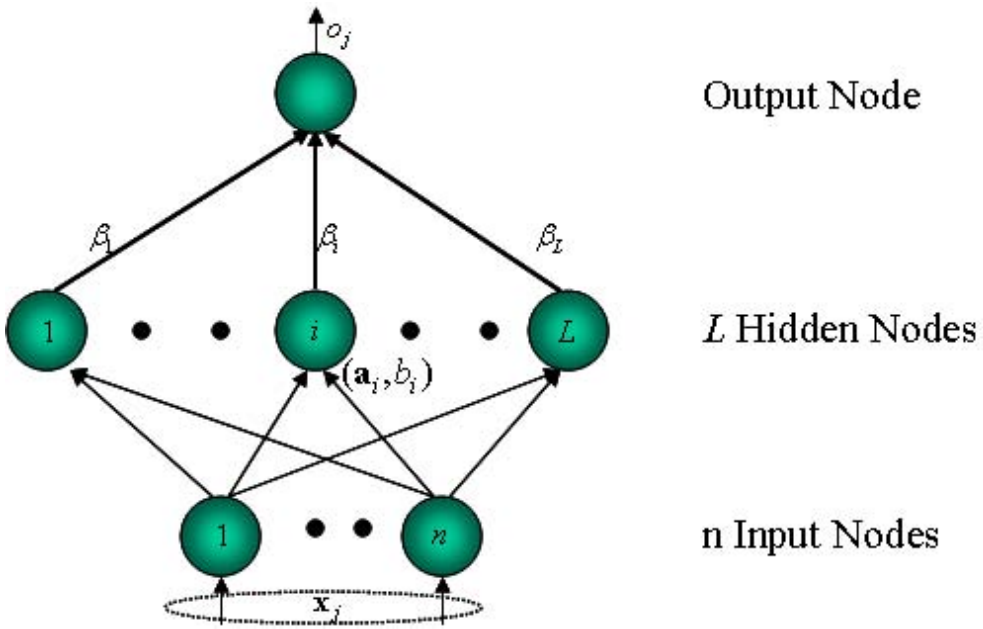


Figure 10: Feedforward Network Architecture

Mathematical Model

- For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbf{R}^n \times \mathbf{R}^m$, standard SLFNs with L hidden nodes and activation function $g(x)$ are mathematically modeled as

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \quad j = 1, \cdots, N \tag{7}$$

- $\mathbf{a}_i$: the input weight vector connecting the ith hidden node and the input nodes or the center of the ith hidden node.
- β_i: the weight vector connecting the ith hidden node and the output node.
- b_i: the threshold or impact factor of the ith hidden node.

Extreme Learning Machine (ELM)

Mathematical Model

- $\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, j = 1, \cdots, N$ is equivalent to $\mathbf{H}\beta = \mathbf{T}$, where

$$\mathbf{H}(\mathbf{a}_1, \cdots, \mathbf{a}_L, b_1, \cdots, b_L, \mathbf{x}_1, \cdots, \mathbf{x}_N) = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L} \tag{8}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{9}$$

$\mathbf{H}$ is called the hidden layer output matrix of the neural network; the ith column of $\mathbf{H}$ is the output of the ith hidden node with respect to inputs $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$.

Outline

Extreme Learning Machine (ELM)

Three-Step Learning Model

Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, activation function g, and the number of hidden nodes L,

1. Assign randomly input weight vectors or centers $\mathbf{a}_i$ and hidden node bias or impact factor b_i, $i = 1, \cdots, L$.
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Calculate the output weight β: $\beta = \mathbf{H}^\dagger \mathbf{T}$.

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of hidden layer output matrix $\mathbf{H}$.

Source Codes of ELM

http://www.ntu.edu.sg/home/egbhuang/

Extreme Learning Machine (ELM)

Three-Step Learning Model

Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, activation function g, and the number of hidden nodes L,

1. Assign randomly input weight vectors or centers $\mathbf{a}_i$ and hidden node bias or impact factor b_i, $i = 1, \cdots, L$.
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Calculate the output weight β: $\beta = \mathbf{H}^\dagger \mathbf{T}$.

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of hidden layer output matrix $\mathbf{H}$.

Source Codes of ELM

http://www.ntu.edu.sg/home/egbhuang/

Extreme Learning Machine (ELM)

Three-Step Learning Model

Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, activation function g, and the number of hidden nodes L,

1. Assign randomly input weight vectors or centers $\mathbf{a}_i$ and hidden node bias or impact factor b_i, $i = 1, \cdots, L$.
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Calculate the output weight β: $\beta = \mathbf{H}^\dagger \mathbf{T}$.

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of hidden layer output matrix $\mathbf{H}$.

Source Codes of ELM

http://www.ntu.edu.sg/home/egbhuang/

Extreme Learning Machine (ELM)

Three-Step Learning Model

Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, activation function g, and the number of hidden nodes L,

1. Assign randomly input weight vectors or centers $\mathbf{a}_i$ and hidden node bias or impact factor b_i, $i = 1, \cdots, L$.
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Calculate the output weight β: $\beta = \mathbf{H}^\dagger \mathbf{T}$.

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of hidden layer output matrix $\mathbf{H}$.

Source Codes of ELM

http://www.ntu.edu.sg/home/egbhuang/

Extreme Learning Machine (ELM)

Three-Step Learning Model

Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, activation function g, and the number of hidden nodes L,

1. Assign randomly input weight vectors or centers $\mathbf{a}_i$ and hidden node bias or impact factor b_i, $i = 1, \cdots, L$.
2. Calculate the hidden layer output matrix $\mathbf{H}$.
3. Calculate the output weight β: $\beta = \mathbf{H}^\dagger \mathbf{T}$.

where $\mathbf{H}^\dagger$ is the Moore-Penrose generalized inverse of hidden layer output matrix $\mathbf{H}$.

Source Codes of ELM

http://www.ntu.edu.sg/home/egbhuang/

ELM Learning Algorithm

Salient Features

- "Simple Math is Enough." ELM is a simple tuning-free three-step algorithm.
- The learning speed of ELM is extremely fast.
- Unlike the traditional classic gradient-based learning algorithms which only work for differentiable activation functions.
- Unlike the traditional classic gradient-based learning algorithms facing several issues like local minima, improper learning rate and overfitting, etc, the ELM tends to reach the solutions straightforward without such trivial issues.
- The ELM learning algorithm looks much simpler than many learning algorithms: neural networks and support vector machines.

Example

- Artificial Case
- Real-World Regression Problems
- Real-World Very Large Complex Applications
- Real Medical Diagnosis Application: Diabetes
- Protein Sequence Classification

Artificial Case: Approximation of 'SinC' Function

$$f(x) = \begin{cases} \sin(x)/x, & x \neq 0 \\ 1, & x = 0 \end{cases} \tag{10}$$

Algorithms	Training Time (seconds)	Training		Testing		# SVs/ nodes
		RMS	Dev	RMS	Dev	
ELM	0.125	0.1148	0.0037	0.0097	0.0028	20
BP	21.26	0.1196	0.0042	0.0159	0.0041	20
SVR	1273.4	0.1149	0.0007	0.0130	0.0012	2499.9

Table 1: Performance comparison for learning function: SinC (5000 noisy training data and 5000 noise-free testing data).

G.-B. Huang, et al., "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.

Artificial Case: Approximation of 'SinC' Function

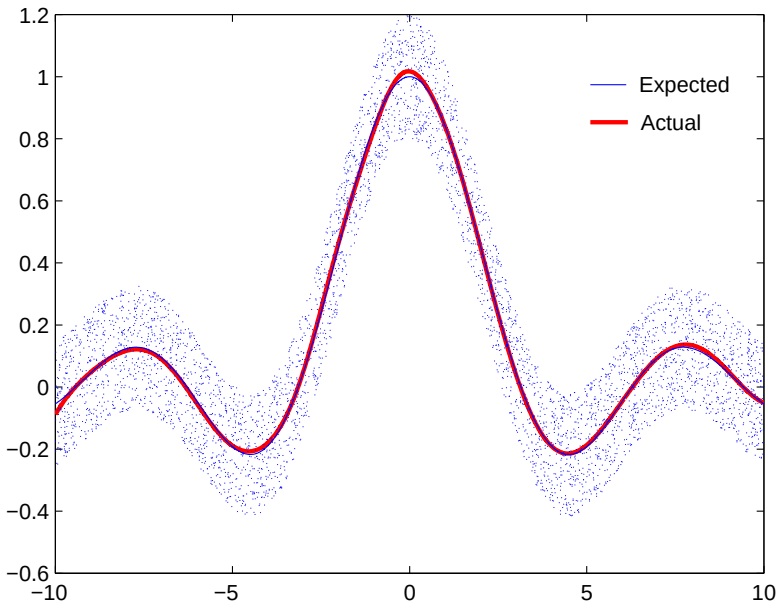


Figure 11: Output of ELM

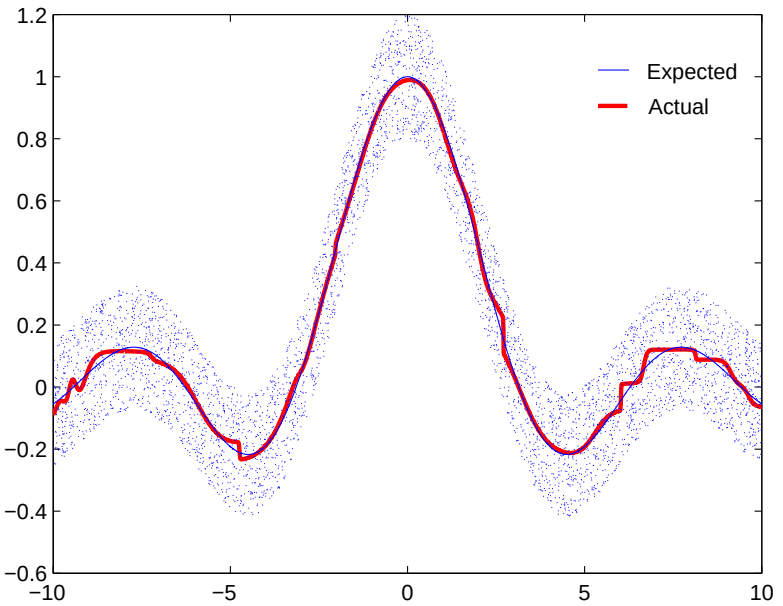


Figure 12: Output of BP

Artificial Case: Approximation of 'SinC' Function

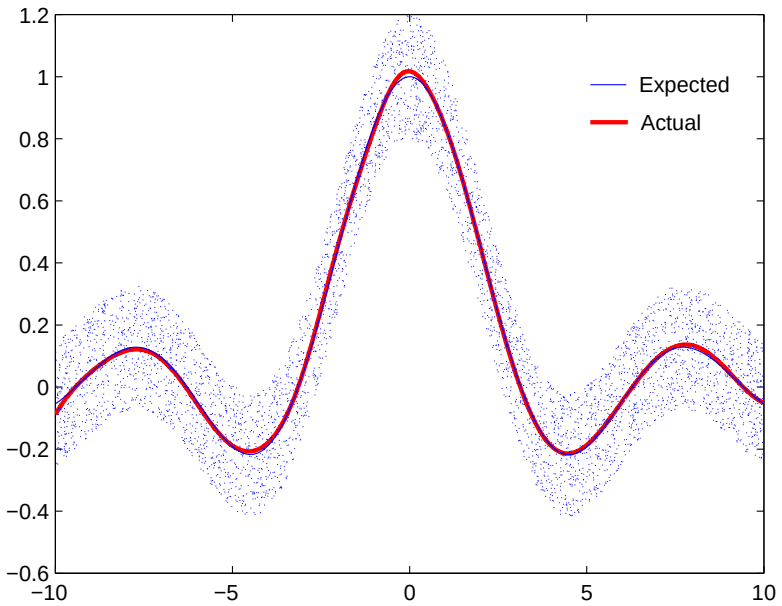


Figure 13: Output of ELM

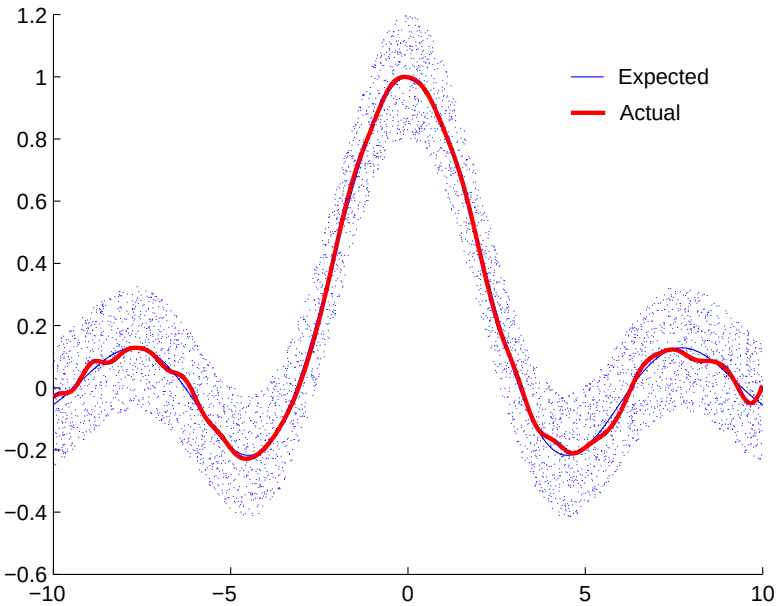


Figure 14: Output of SVM

Real-World Regression Problems

Datasets	BP		ELM	
	training	testing	training	testing
Abalone	0.0785	0.0874	0.0803	0.0824
Delta Ailerons	0.0409	0.0481	0.0423	0.0431
Delta Elevators	0.0544	0.0592	0.0550	0.0568
Computer Activity	0.0273	0.0409	0.0316	0.0382
Census (House8L)	0.0596	0.0685	0.0624	0.0660
Auto Price	0.0443	0.1157	0.0754	0.0994
Triazines	0.1438	0.2197	0.1897	0.2002
Machine CPU	0.0352	0.0826	0.0332	0.0539
Servo	0.0794	0.1276	0.0707	0.1196
Breast Cancer	0.2788	0.3155	0.2470	0.2679
Bank domains	0.0342	0.0379	0.0406	0.0366
California Housing	0.1046	0.1285	0.1217	0.1267
Stocks domain	0.0179	0.0358	0.0251	0.0348

Table 2: Comparison of training and testing RMSE of BP and ELM.

Real-World Regression Problems

Datasets	SVR		ELM	
	training	testing	training	testing
Abalone	0.0759	0.0784	0.0803	0.0824
Delta Ailerons	0.0418	0.0429	0.0423	0.0431
Delta Elevators	0.0534	0.0540	0.0545	0.0568
Computer Activity	0.0464	0.0470	0.0316	0.0382
Census (House8L)	0.0718	0.0746	0.0624	0.0660
Auto Price	0.0652	0.0937	0.0754	0.0994
Triazines	0.1432	0.1829	0.1897	0.2002
Machine CPU	0.0574	0.0811	0.0332	0.0539
Servo	0.0840	0.1177	0.0707	0.1196
Breast Cancer	0.2278	0.2643	0.2470	0.2679
Bank domains	0.0454	0.0467	0.0406	0.0366
California Housing	0.1089	0.1180	0.1217	0.1267
Stocks domain	0.0503	0.0518	0.0251	0.0348

Table 3: Comparison of training and testing RMSE of SVR and ELM.

Real-World Regression Problems

Datasets	BP	SVR		ELM
	# nodes	(C, γ)	# SVs	# nodes
Abalone	10	$(2^4, 2^{-6})$	309.84	25
Delta Ailerons	10	$(2^3, 2^{-3})$	82.44	45
Delta Elevators	5	$(2^0, 2^{-2})$	260.38	125
Computer Activity	45	$(2^5, 2^{-5})$	64.2	125
Census (House8L)	10	$(2^1, 2^{-1})$	810.24	160
Auto Price	5	$(2^8, 2^{-5})$	21.25	15
Triazines	5	$(2^{-1}, 2^{-9})$	48.42	10
Machine CPU	10	$(2^6, 2^{-4})$	7.8	10
Servo	10	$(2^2, 2^{-2})$	22.375	30
Breast Cancer	5	$(2^{-1}, 2^{-4})$	74.3	10
Bank domains	20	$(2^{10}, 2^{-2})$	129.22	190
California Housing	10	$(2^3, 2^1)$	2189.2	80
Stocks domain	20	$(2^3, 2^{-9})$	19.94	110

Table 4: Comparison of network complexity of BP, SVR and ELM.

Real-World Regression Problems

Datasets	BP[a]	SVR[b]	ELM[a]
Abalone	1.7562	1.6123	0.0125
Delta Ailerons	2.7525	0.6726	0.0591
Delta Elevators	1.1938	1.121	0.2812
Computer Activity	67.44	1.0149	0.2951
Census (House8L)	8.0647	11.251	1.0795
Auto Price	0.2456	0.0042	0.0016
Triazines	0.5484	0.0086	$< 10^{-4}$
Machine CPU	0.2354	0.0018	0.0015
Servo	0.2447	0.0045	$< 10^{-4}$
Breast Cancer	0.3856	0.0064	$< 10^{-4}$
Bank domains	7.506	1.6084	0.6434
California Housing	6.532	74.184	1.1177
Stocks domain	1.0487	0.0690	0.0172

[a] run in MATLAB environment.
[b] run in C executable environment.

Table 5: Comparison of training time (seconds) of BP, SVR and ELM.

Real-World Very Large Complex Applications

Algorithms	Time (minutes)		Success Rate (%)				# SVs/ nodes
	Training	Testing	Training		Testing		
			Rate	Dev	Rate	Dev	
ELM	1.6148	0.7195	92.35	0.026	90.21	0.024	200
SLFN	12	N/A	82.44	N/A	81.85	N/A	100
SVM	693.6000	347.7833	91.70	N/A	89.90	N/A	31,806

Table 6: Performance comparison of the ELM, BP and SVM learning algorithms in Forest Type Prediction application. (100, 000 training data and 480,000+ testing data, each data has 53 attributes.)

Real Medical Diagnosis Application: Diabetes

Algorithms	Time (seconds)		Success Rate (%)				# SVs/ nodes
	Training	Testing	Training		Testing		
			Rate	Dev	Rate	Dev	
ELM	0.0118	0.0031	78.68	1.18	77.57	2.85	20
BP	3.0116	0.0035	86.63	1.7	74.73	3.2	20
SVM	0.1860	0.0673	78.76	0.91	77.31	2.35	317.16

Table 7: Performance comparison: ELM, BP and SVM.

Algorithms	Testing Rate (%)
ELM	77.57
SVM	76.50
SAOCIF	77.32
Cascade-Correlation	76.58
AdaBoost	75.60
C4.5	71.60
RBF	76.30
Heterogeneous RBF	76.30

Table 8: Performance comparison: ELM and other popular methods.

Protein Sequence Classification

Algorithms	Training Time		Testing (%)		# SVs/ nodes
	(seconds)	Speedup	Rate	Dev	
ELM (Sigmoid)	0.0998	17417	96.738	0.8628	160
ELM (RBF Kernel)	9.4351	184.1528	96.498	0.6768	485
SVM (RBF Kernel)	0.3434	5060	98.056	0.6819	306.42
BP	1737.5	1	96.037	1.2132	35

Table 9: Performance comparison of different classifiers: Protein Sequence Classification.

D. Wang, et al., "Protein Sequence Classification Using Extreme Learning Machine," *Proceedings of International Joint Conference on Neural Networks (IJCNN2005)*, (Montreal, Canada), 31 July - 4 August, 2005.

Protein Sequence Classification

Training Time (seconds)	Training (%)		Testing (%)		# Nodes
	Rate	Dev	Rate	Dev	
66.604	88.35	1.9389	85.685	2.8345	5
171.02	98.729	1.276	94.524	2.2774	10
374.12	99.45	0.8820	94.757	1.7648	15
624.89	99.6	0.5356	95.558	1.4828	20
843.68	99.511	1.0176	95.551	1.6258	25
1228.4	99.576	1.2518	95.378	1.9001	30
1737.5	**99.739**	**0.5353**	**96.037**	**1.2132**	**35**

Table 10: Performance of BP classifier: Protein Sequence Classification.

Sensitivity of number of hidden nodes

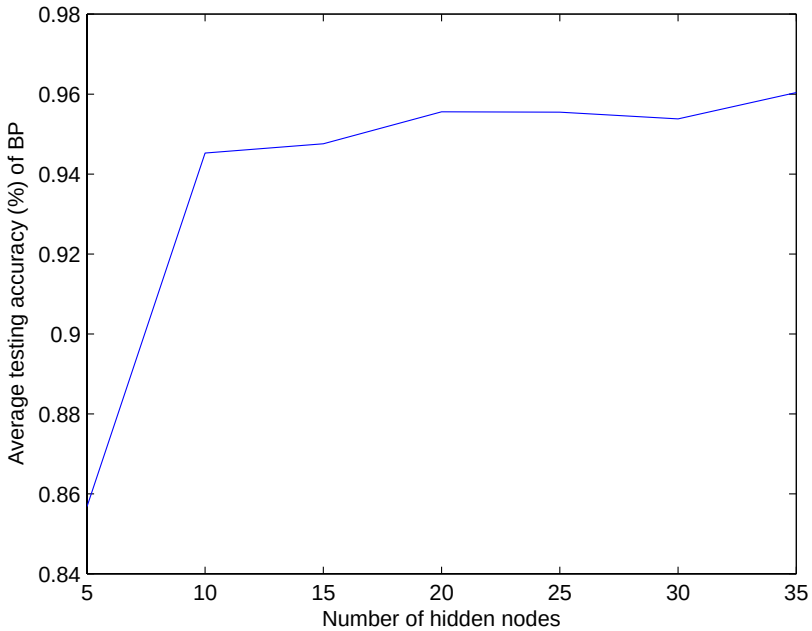


Figure 15: BP

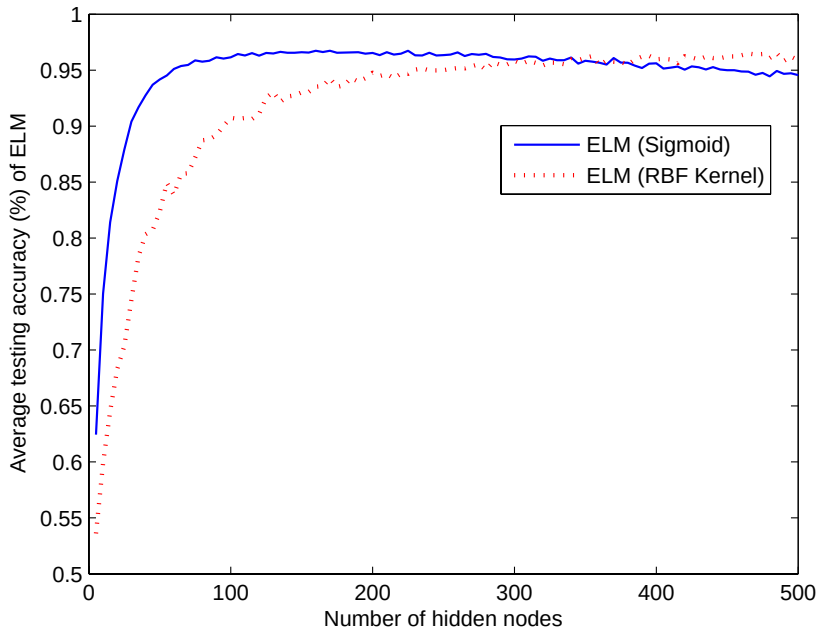


Figure 16: ELM

Summary

- ELM needs much less training time compared to popular BP and SVM/SVR.
- The prediction accuracy of ELM is usually slightly better than BP and close to SVM/SVR in many applications.
- Compared with BP and SVR, ELM can be implemented easily since there is no parameter to be tuned except an insenstive parameter L.
- It should be noted that many nonlinear activation functions can be used in ELM.
- ELM needs more hidden nodes than BP but much less nodes than SVM/SVR, which implies that ELM and BP have much shorter response time to unknown data than SVM/SVR.

Summary

- ELM needs much less training time compared to popular BP and SVM/SVR.
- The prediction accuracy of ELM is usually slightly better than BP and close to SVM/SVR in many applications.
- Compared with BP and SVR, ELM can be implemented easily since there is no parameter to be tuned except an insenstive parameter L.
- It should be noted that many nonlinear activation functions can be used in ELM.
- ELM needs more hidden nodes than BP but much less nodes than SVM/SVR, which implies that ELM and BP have much shorter response time to unknown data than SVM/SVR.

Summary

- ELM needs much less training time compared to popular BP and SVM/SVR.
- The prediction accuracy of ELM is usually slightly better than BP and close to SVM/SVR in many applications.
- Compared with BP and SVR, ELM can be implemented easily since there is no parameter to be tuned except an insenstive parameter L.
- It should be noted that many nonlinear activation functions can be used in ELM.
- ELM needs more hidden nodes than BP but much less nodes than SVM/SVR, which implies that ELM and BP have much shorter response time to unknown data than SVM/SVR.

Summary

- ELM needs much less training time compared to popular BP and SVM/SVR.
- The prediction accuracy of ELM is usually slightly better than BP and close to SVM/SVR in many applications.
- Compared with BP and SVR, ELM can be implemented easily since there is no parameter to be tuned except an insenstive parameter L.
- It should be noted that many nonlinear activation functions can be used in ELM.
- ELM needs more hidden nodes than BP but much less nodes than SVM/SVR, which implies that ELM and BP have much shorter response time to unknown data than SVM/SVR.

Summary

- ELM needs much less training time compared to popular BP and SVM/SVR.
- The prediction accuracy of ELM is usually slightly better than BP and close to SVM/SVR in many applications.
- Compared with BP and SVR, ELM can be implemented easily since there is no parameter to be tuned except an insenstive parameter L.
- It should be noted that many nonlinear activation functions can be used in ELM.
- ELM needs more hidden nodes than BP but much less nodes than SVM/SVR, which implies that ELM and BP have much shorter response time to unknown data than SVM/SVR.

For Further Reading


G.-B. Huang, et al., "Universal Approximation Using Incremental Networks with Random Hidden Computational Nodes", *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.

G.-B. Huang, et al., "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.

N.-Y. Liang, et al., "A Fast and Accurate On-line Sequential Learning Algorithm for Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411-1423, 2006.

Q.-Y. Zhu, et al., "Evolutionary Extreme Learning Machine", *Pattern Recognition*, vol. 38, no. 10, pp. 1759-1763, 2005.

R. Zhang, et al., "Multi-Category Classification Using Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (in press), 2006.

G.-B. Huang, et al., "Can Threshold Networks Be Trained Directly?" *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 3, pp. 187-191, 2006.

Source Codes and references of ELM: http://www.ntu.edu.sg/home/egbhuang/

For Further Reading


G.-B. Huang, et al., "Universal Approximation Using Incremental Networks with Random Hidden Computational Nodes", *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.


G.-B. Huang, et al., "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.

N.-Y. Liang, et al., "A Fast and Accurate On-line Sequential Learning Algorithm for Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411-1423, 2006.

Q.-Y. Zhu, et al., "Evolutionary Extreme Learning Machine", *Pattern Recognition*, vol. 38, no. 10, pp. 1759-1763, 2005.

R. Zhang, et al., "Multi-Category Classification Using Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (in press), 2006.

G.-B. Huang, et al., "Can Threshold Networks Be Trained Directly?" *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 3, pp. 187-191, 2006.

Source Codes and references of ELM: http://www.ntu.edu.sg/home/egbhuang/

For Further Reading


G.-B. Huang, et al., "Universal Approximation Using Incremental Networks with Random Hidden Computational Nodes", *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.


G.-B. Huang, et al., "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.


N.-Y. Liang, et al., "A Fast and Accurate On-line Sequential Learning Algorithm for Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411-1423, 2006.

Q.-Y. Zhu, et al., "Evolutionary Extreme Learning Machine", *Pattern Recognition*, vol. 38, no. 10, pp. 1759-1763, 2005.

R. Zhang, et al., "Multi-Category Classification Using Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (in press), 2006.

G.-B. Huang, et al., "Can Threshold Networks Be Trained Directly?" *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 3, pp. 187-191, 2006.

Source Codes and references of ELM: http://www.ntu.edu.sg/home/egbhuang/

For Further Reading


G.-B. Huang, et al., "Universal Approximation Using Incremental Networks with Random Hidden Computational Nodes", *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.


G.-B. Huang, et al., "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.


N.-Y. Liang, et al., "A Fast and Accurate On-line Sequential Learning Algorithm for Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411-1423, 2006.


Q.-Y. Zhu, et al., "Evolutionary Extreme Learning Machine", *Pattern Recognition*, vol. 38, no. 10, pp. 1759-1763, 2005.

R. Zhang, et al., "Multi-Category Classification Using Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (in press), 2006.

G.-B. Huang, et al., "Can Threshold Networks Be Trained Directly?" *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 3, pp. 187-191, 2006.

Source Codes and references of ELM: http://www.ntu.edu.sg/home/egbhuang/

For Further Reading


G.-B. Huang, et al., "Universal Approximation Using Incremental Networks with Random Hidden Computational Nodes", *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.


G.-B. Huang, et al., "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.


N.-Y. Liang, et al., "A Fast and Accurate On-line Sequential Learning Algorithm for Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411-1423, 2006.


Q.-Y. Zhu, et al., "Evolutionary Extreme Learning Machine", *Pattern Recognition*, vol. 38, no. 10, pp. 1759-1763, 2005.


R. Zhang, et al., "Multi-Category Classification Using Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (in press), 2006.

G.-B. Huang, et al., "Can Threshold Networks Be Trained Directly?" *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 3, pp. 187-191, 2006.

Source Codes and references of ELM: http://www.ntu.edu.sg/home/egbhuang/

For Further Reading


G.-B. Huang, et al., "Universal Approximation Using Incremental Networks with Random Hidden Computational Nodes", *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.


G.-B. Huang, et al., "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.


N.-Y. Liang, et al., "A Fast and Accurate On-line Sequential Learning Algorithm for Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411-1423, 2006.


Q.-Y. Zhu, et al., "Evolutionary Extreme Learning Machine", *Pattern Recognition*, vol. 38, no. 10, pp. 1759-1763, 2005.


R. Zhang, et al., "Multi-Category Classification Using Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (in press), 2006.


G.-B. Huang, et al., "Can Threshold Networks Be Trained Directly?" *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 3, pp. 187-191, 2006.

Source Codes and references of ELM: http://www.ntu.edu.sg/home/egbhuang/

For Further Reading


G.-B. Huang, et al., "Universal Approximation Using Incremental Networks with Random Hidden Computational Nodes", *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879-892, 2006.


G.-B. Huang, et al., "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, vol. 70, pp. 489-501, 2006.


N.-Y. Liang, et al., "A Fast and Accurate On-line Sequential Learning Algorithm for Feedforward Networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411-1423, 2006.


Q.-Y. Zhu, et al., "Evolutionary Extreme Learning Machine", *Pattern Recognition*, vol. 38, no. 10, pp. 1759-1763, 2005.


R. Zhang, et al., "Multi-Category Classification Using Extreme Learning Machine for Microarray Gene Expression Cancer Diagnosis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics* (in press), 2006.


G.-B. Huang, et al., "Can Threshold Networks Be Trained Directly?" *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 3, pp. 187-191, 2006.

Source Codes and references of ELM: http://www.ntu.edu.sg/home/egbhuang/