Extreme Learning Machine: Learning Without Iterative Tuning

Guang-Bin HUANG

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

Talks in ELM2010 (Adelaide, Australia, Dec 7 2010), HP Labs (Palo Alto, USA), Microsoft Research (Redmond, USA), UBC (Canada), Institute for InfoComm Research (Singapore), EEE/NTU (Singapore)

Outline

Outline

Outline

Outline

Outline

Feedforward Neural Networks with Additive Nodes

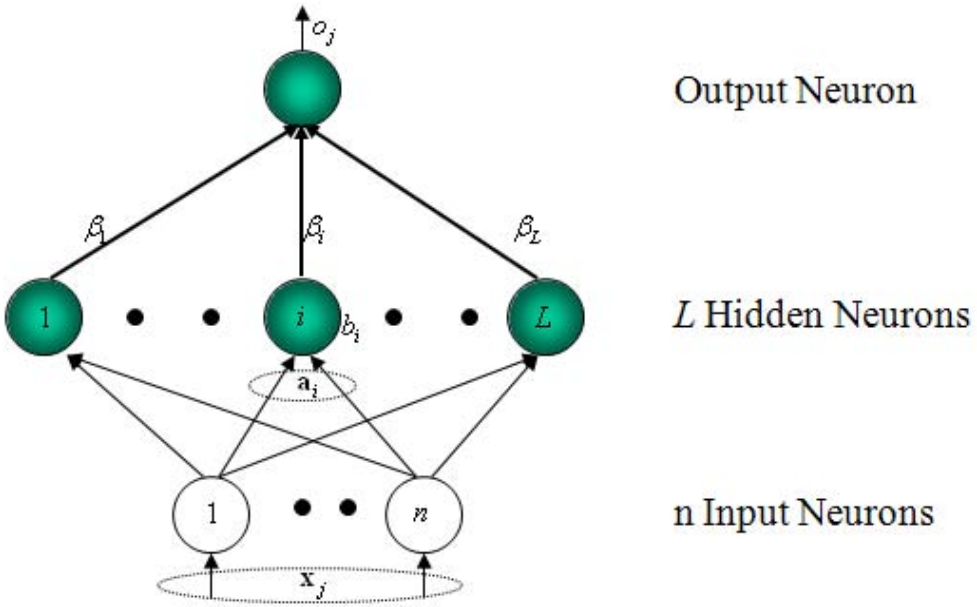


Figure 1: SLFN: additive hidden nodes

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

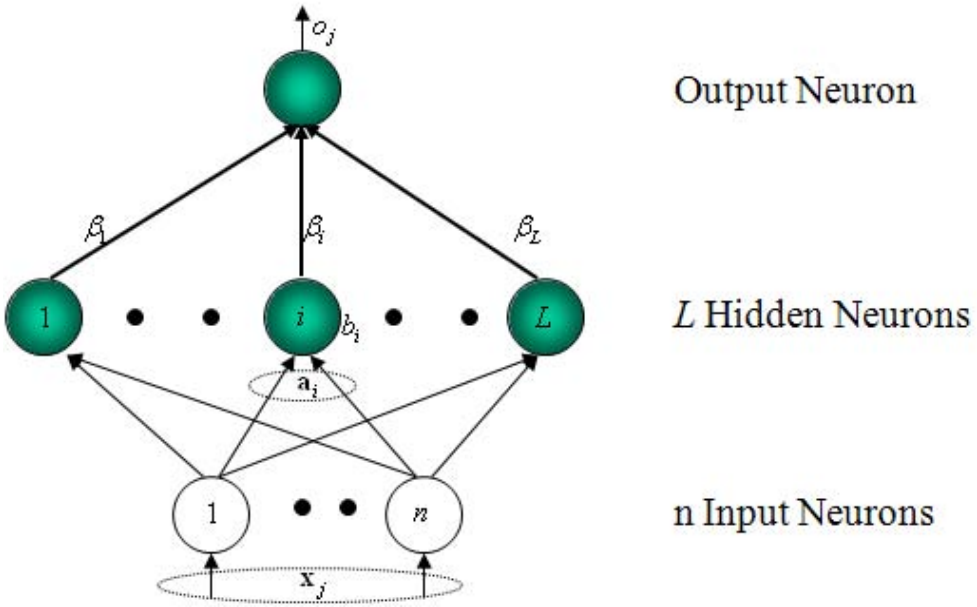


Figure 1: SLFN: additive hidden nodes

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

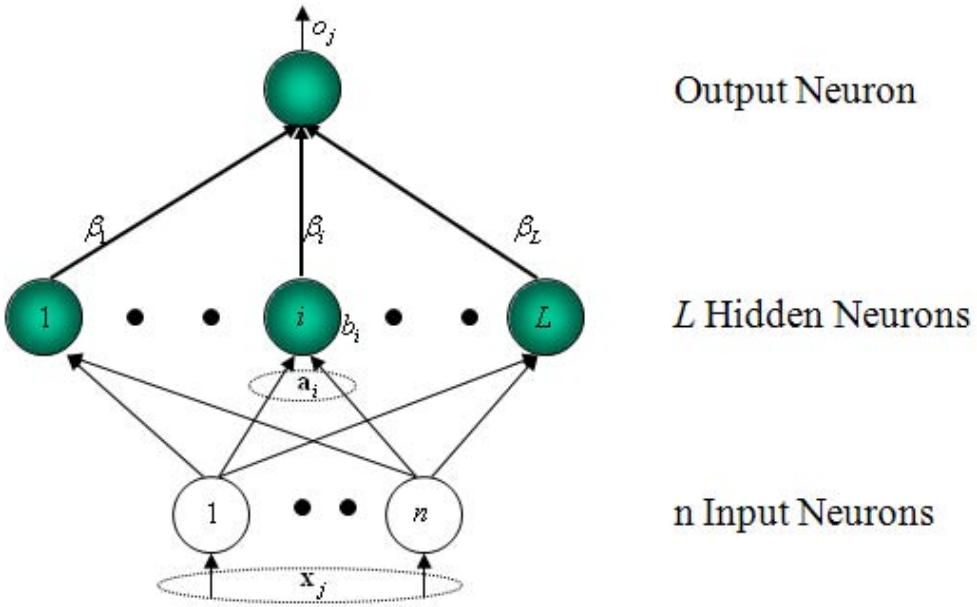


Figure 1: SLFN: additive hidden nodes

Output of hidden nodes

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

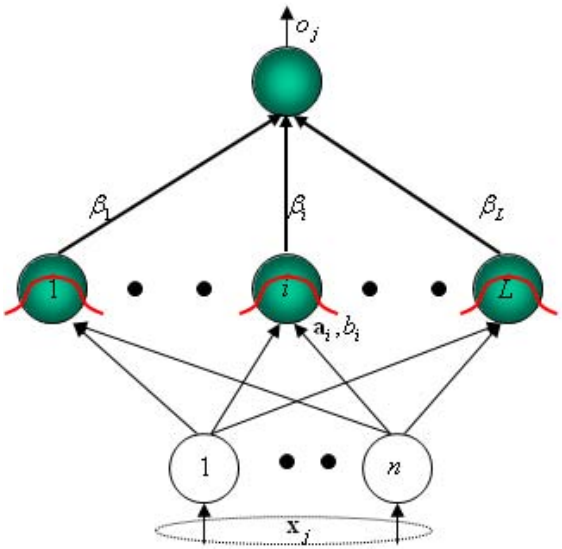
$\mathbf{a}_i$: the weight vector connecting the ith hidden node and the input nodes.
b_i: the threshold of the ith hidden node.

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 RBF Nodes



Output Neuron

L RBF Hidden Neurons

n Input Neurons

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.

Figure 2: Feedforward Network Architecture: RBF hidden nodes

Feedforward Neural Networks with RBF Nodes

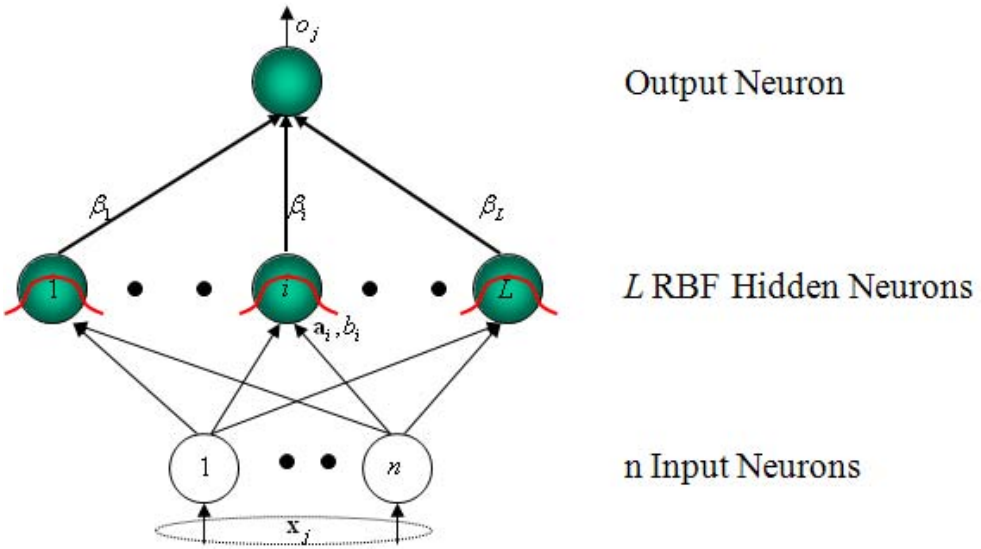


Figure 2: Feedforward Network Architecture: RBF hidden nodes

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

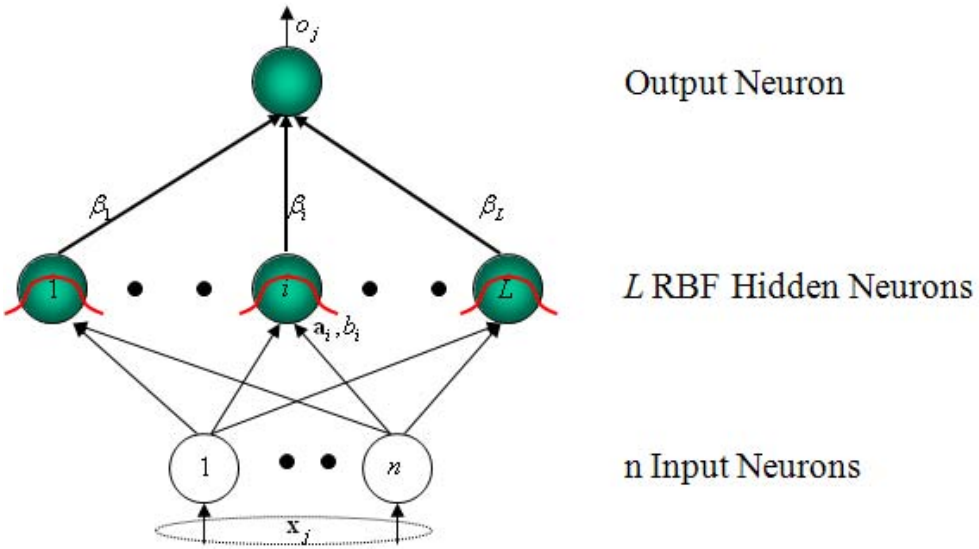


Figure 2: Feedforward Network Architecture: RBF hidden nodes

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

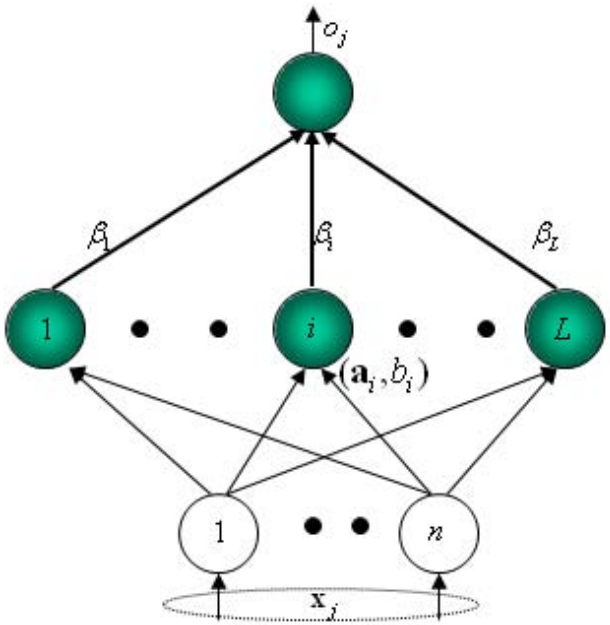


Figure 3: SLFN.

Mathematical Model

Any continuous target function $f(\mathbf{x})$ can be approximated by SLFNs with adjustable hidden nodes. 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 that unknown f could be approximated by SLFNs f_L appropriately.

Function Approximation of Neural Networks

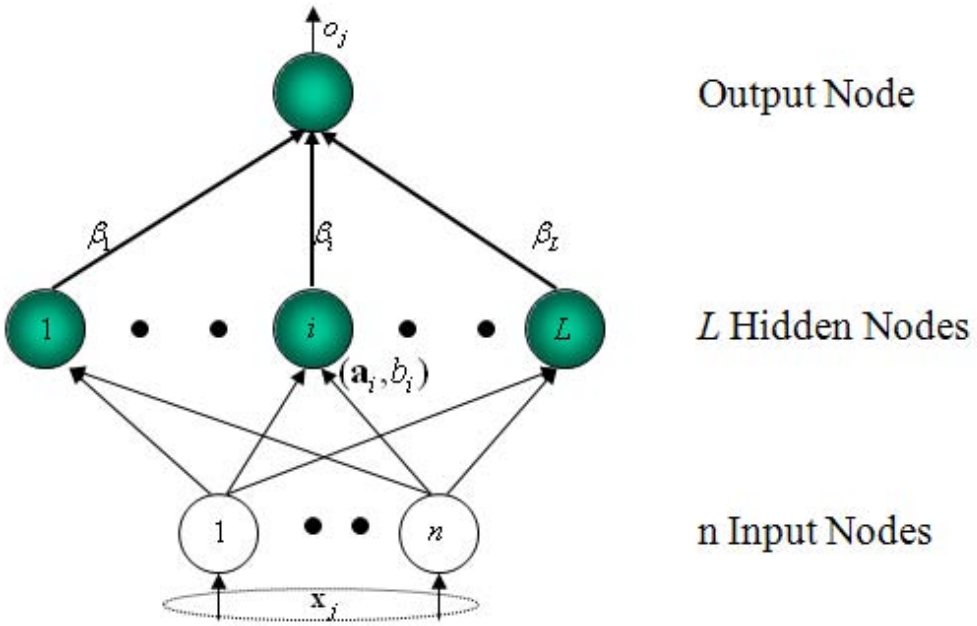


Figure 3: SLFN.

Mathematical Model

Any continuous target function $f(\mathbf{x})$ can be approximated by SLFNs with adjustable hidden nodes. 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 that unknown f could be approximated by SLFNs f_L appropriately.

Function Approximation of Neural Networks

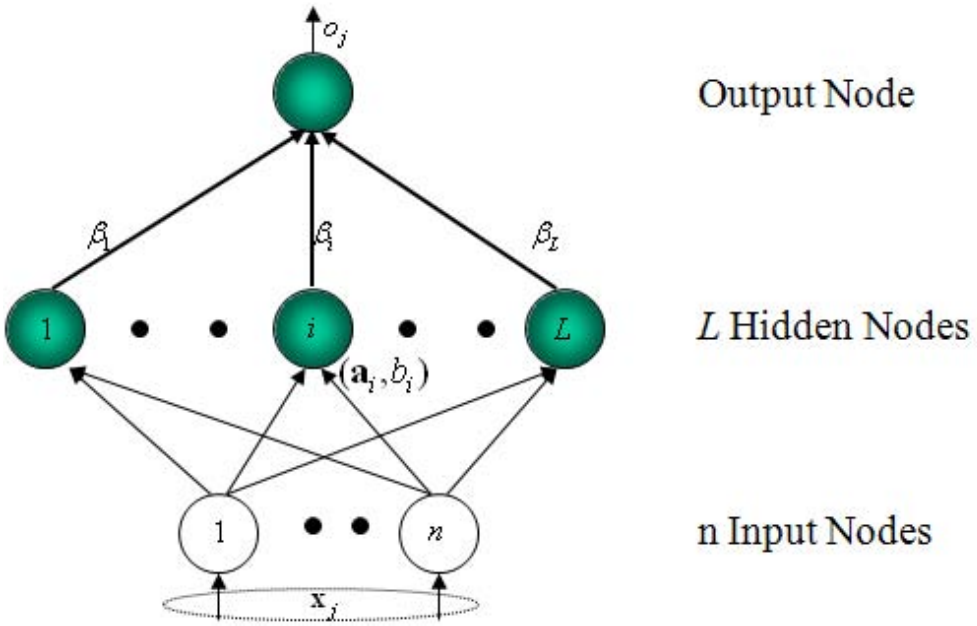


Figure 3: SLFN.

Mathematical Model

Any continuous target function $f(\mathbf{x})$ can be approximated by SLFNs with adjustable hidden nodes. 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 that unknown f could be approximated by SLFNs f_L appropriately.

Function Approximation of Neural Networks

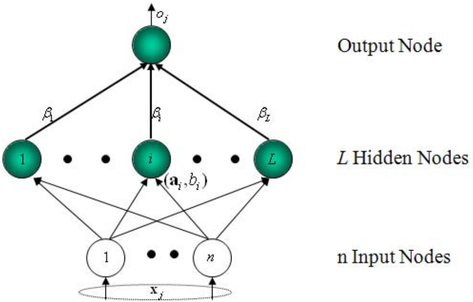


Figure 4: SLFN.

Learning Model

- For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbf{R}^n \times \mathbf{R}^m$, 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

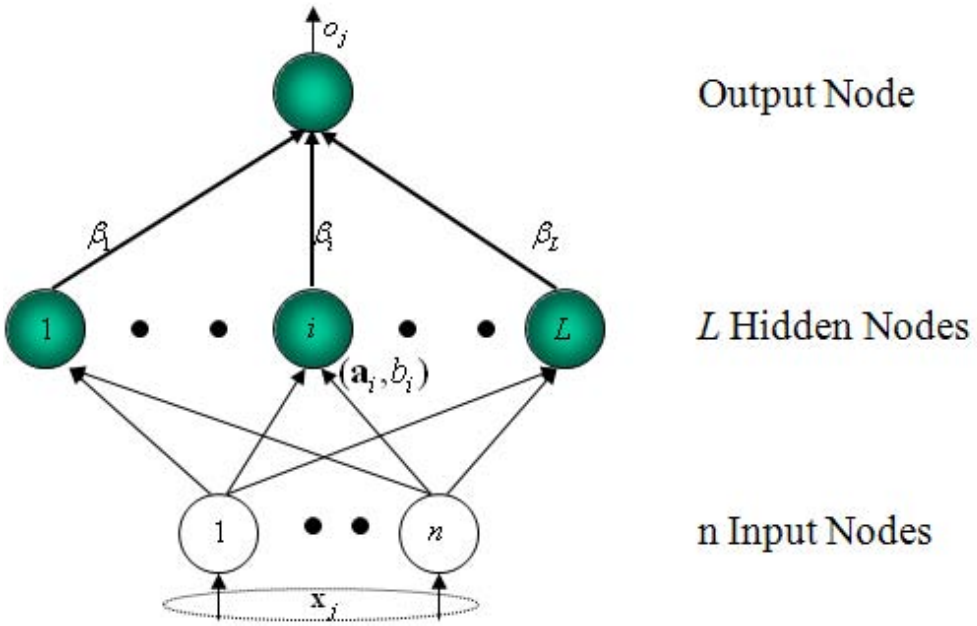


Figure 4: SLFN.

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

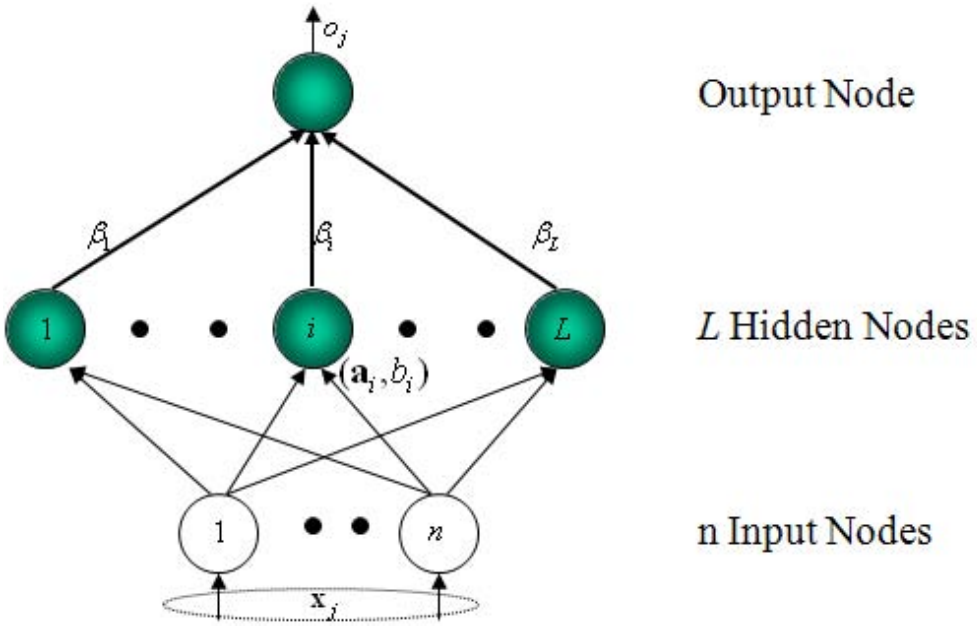


Figure 4: SLFN.

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_j, $\mathbf{a}_j$, b_j.

Function Approximation of Neural Networks

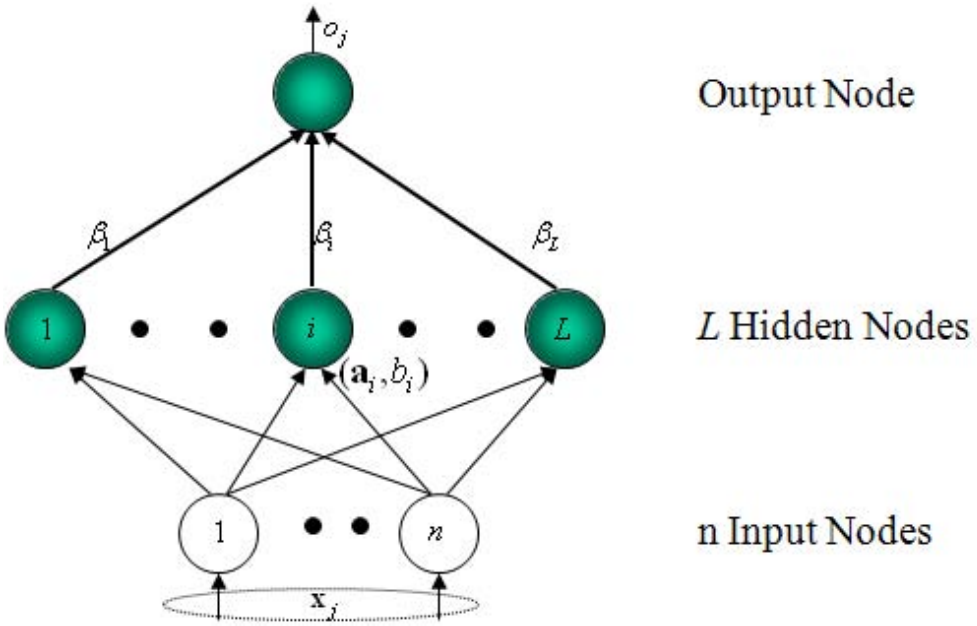


Figure 4: SLFN.

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: $\boldsymbol{\beta}_i$, $\mathbf{a}_i$, b_i.

Outline

Classification Capability of SLFNs

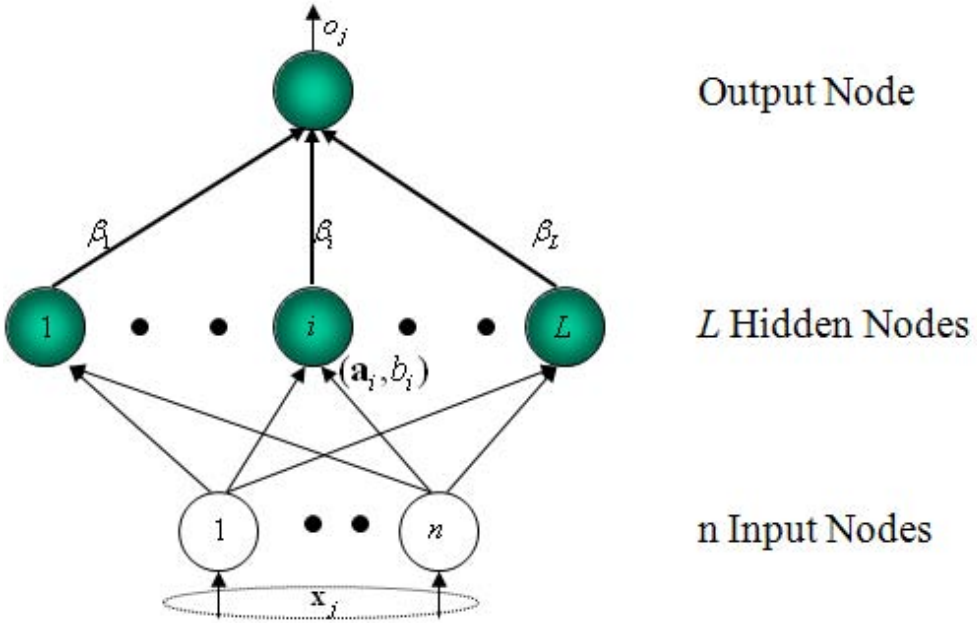


Figure 5: SLFN.

As long as SLFNs can approximate any continuous target function $f(\mathbf{x})$, such SLFNs can differentiate any disjoint regions.

G.-B. Huang, et al., "Classification Ability of Single Hidden Layer Feedforward Neural Networks," *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 799-801, 2000.

Outline

Learning Algorithms of Neural Networks

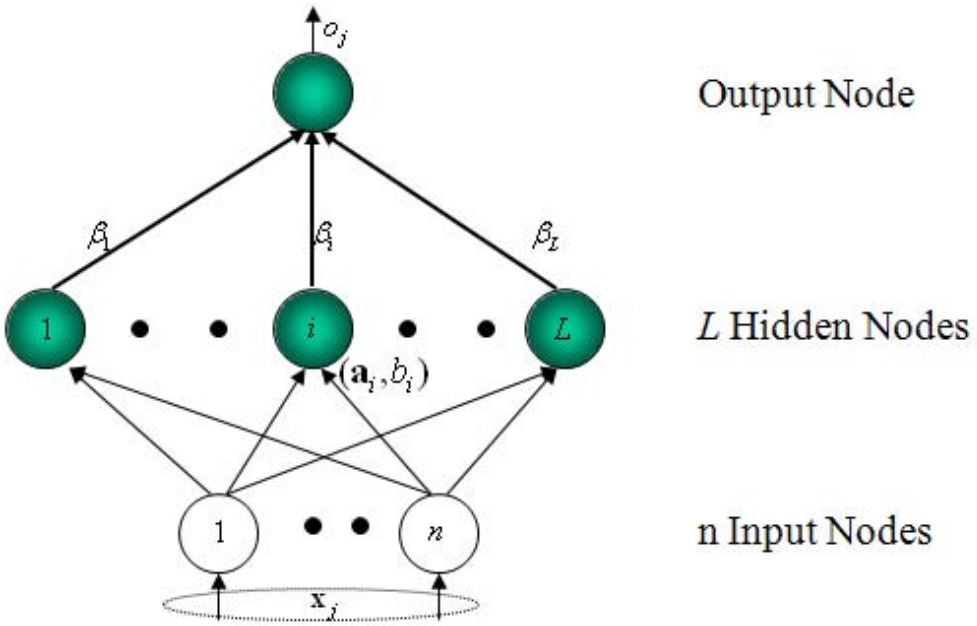
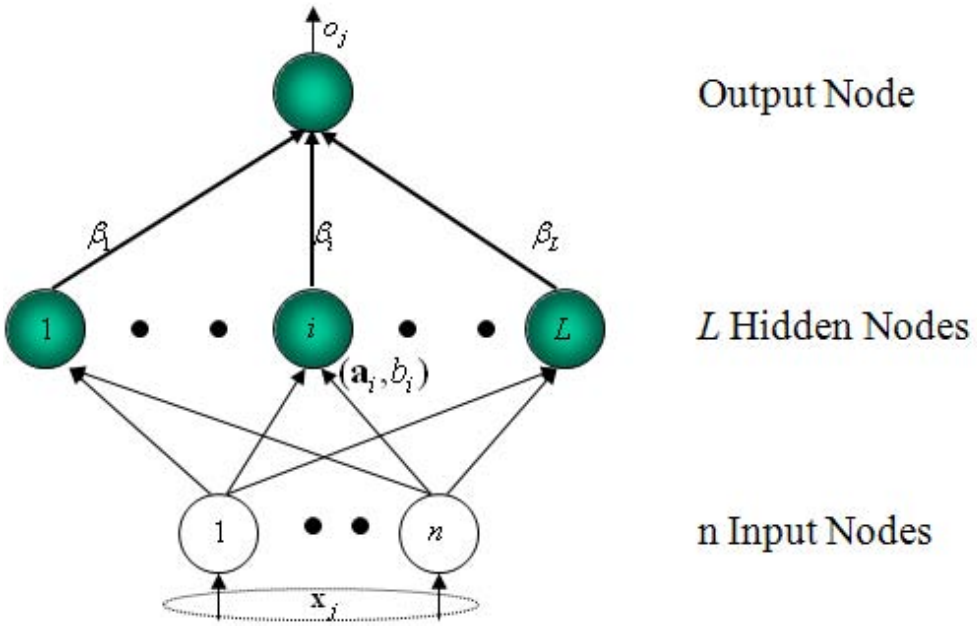


Figure 6: Feedforward Network Architecture.

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.
- Local minima may exist
- Stopping criteria
- Learning rate
- Learning epochs
- ...

Learning Algorithms of Neural Networks

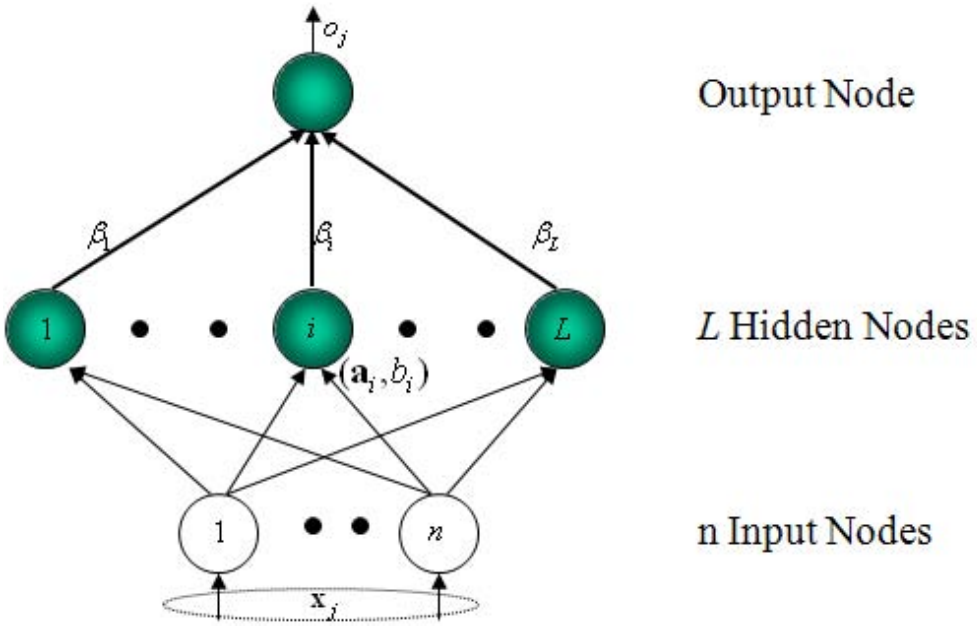


Learning Methods

- 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.
- Least-square (LS) solution for RBF network, with single impact factor for all hidden nodes.

Figure 6: Feedforward Network Architecture.

Learning Algorithms of Neural Networks

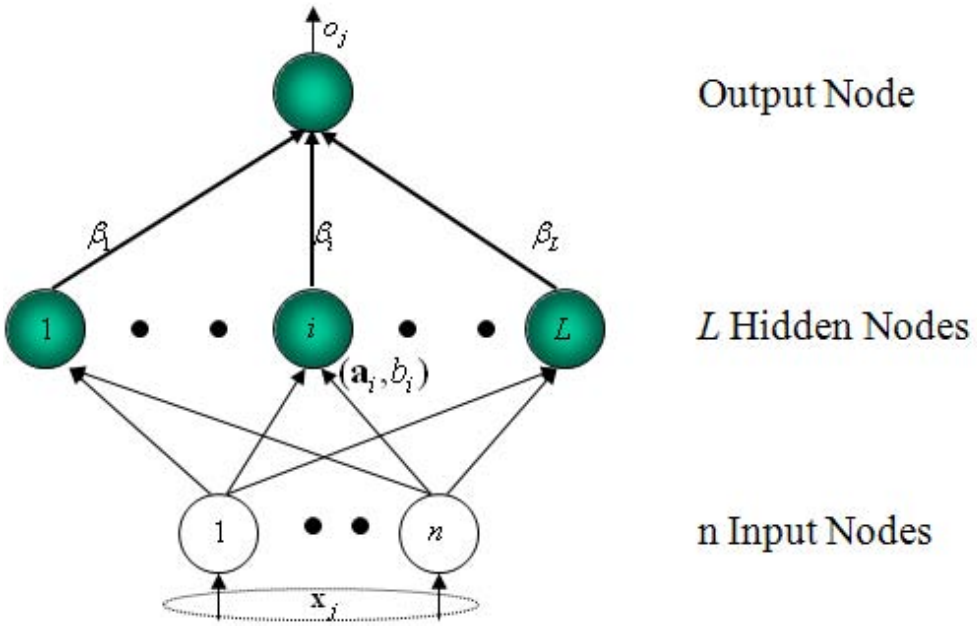


Learning Methods

- 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.
- Least-square (LS) solution for RBF network, with single impact factor for all hidden nodes.

Figure 6: Feedforward Network Architecture.

Learning Algorithms of Neural Networks



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.
- Least-square (LS) solution for RBF network, with single impact factor for all hidden nodes.

Figure 6: Feedforward Network Architecture.

Advantages 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 manually.
- Overfitting.
- Local minima.
- Time-consuming.

Advantages 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 manually.
- Overfitting.
- Local minima.
- Time-consuming.

Outline

Support Vector Machine

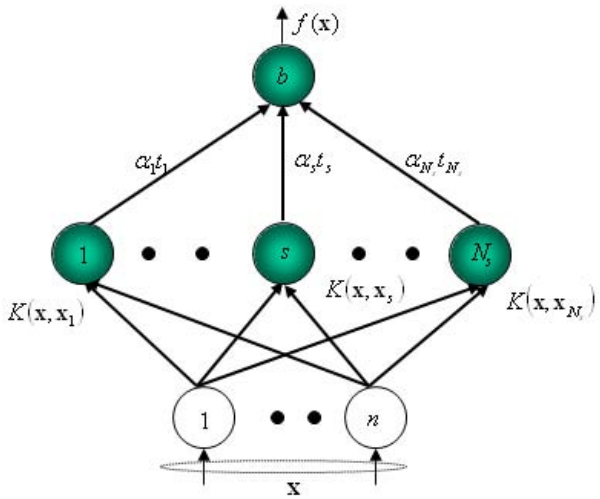


Figure 7: SVM Architecture.

SVM optimization formula:

$$\begin{aligned} \text{Minimize: } & L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{Subject to: } & t_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \quad i = 1, \cdots, N \\ & \xi_i \geq 0, \quad i = 1, \cdots, N \end{aligned} \tag{7}$$

The decision function of SVM is:

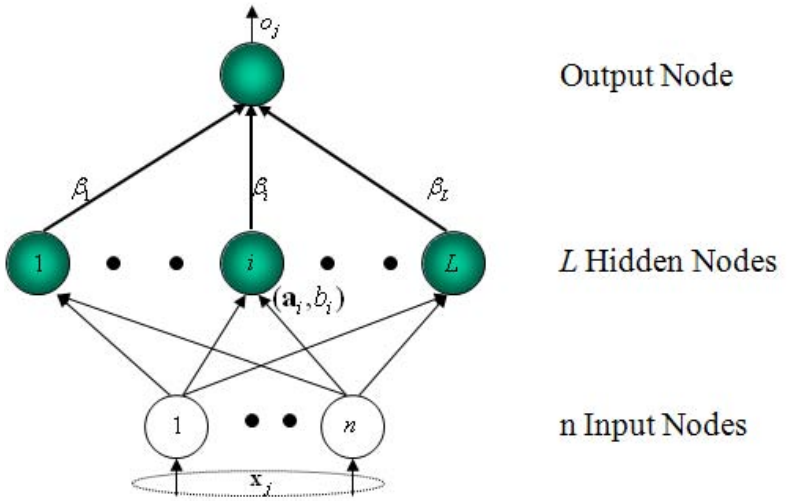
$$f(\mathbf{x}) = \text{sign}\left(\sum_{s=1}^{N_S}\alpha_s t_s K(\mathbf{x}, \mathbf{x}_s) + b\right) \tag{8}$$

B. Frénay and M. Verleysen, "Using SVMs with Randomised Feature Spaces: an Extreme Learning Approach," *ESANN*, Bruges, Belgium, pp. 315-320, 28-30 April, 2010.

G.-B. Huang, et al., "Optimization Method Based Extreme Learning Machine for Classification," *Neurocomputing*, vol. 74, pp. 155-163, 2010.

Outline

Generalized SLFNs



General Hidden Layer Mapping

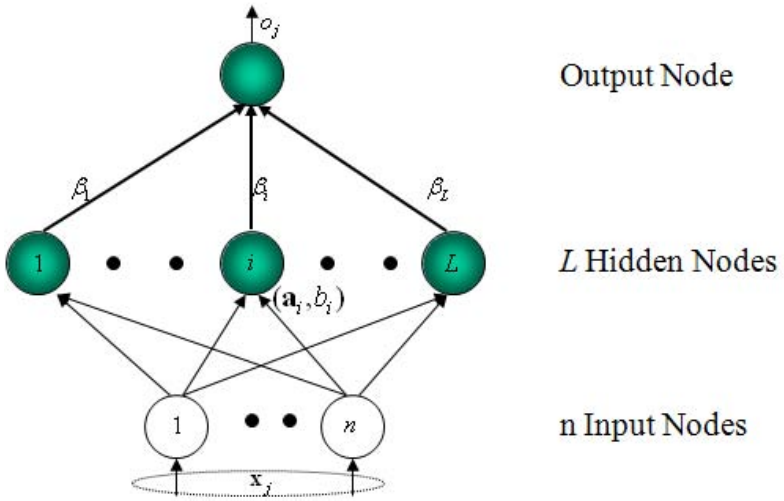
- Output function: $f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x})$
- The hidden layer output function (hidden layer mapping):
  $h(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \cdots, G(\mathbf{a}_L, b_L, \mathbf{x})]$
- The output function needn't be:
  Sigmoid: $G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i)$
  RBF: $G(\mathbf{a}_i, b_i, \mathbf{x}) = g(b_i \|\mathbf{x} - \mathbf{a}_i\|)$

Figure 8: SLFN: any type of piecewise continuous $G(\mathbf{a}_i, b_i, \mathbf{x})$.

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

G.-B. Huang, et al., "Convex Incremental Extreme Learning Machine," *Neurocomputing*, vol. 70, pp. 3056-3062, 2007.

Generalized SLFNs



General Hidden Layer Mapping

- Output function: $f(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x})$
- The hidden layer output function (hidden layer mapping):
  $h(\mathbf{x}) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \cdots, G(\mathbf{a}_L, b_L, \mathbf{x})]$
- The output function needn't be:
  Sigmoid: $G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i)$
  RBF: $G(\mathbf{a}_i, b_i, \mathbf{x}) = g\left(b_i \|\mathbf{x} - \mathbf{a}_i\|\right)$

Figure 8: SLFN: any type of piecewise continuous $G(\mathbf{a}_i, b_i, \mathbf{x})$.

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

G.-B. Huang, et al., "Convex Incremental Extreme Learning Machine," *Neurocomputing*, vol. 70, pp. 3056-3062, 2007.

Outline

New Learning Theory: Learning Without Iterative Tuning

New Learning View

- Learning Without Iterative Tuning: Given any nonconstant piecewise continuous function g, if continuous target function $f(\mathbf{x})$ can be approximated by SLFNs with adjustable hidden nodes g then the hidden node parameters of such SLFNs needn't be tuned.
- All these hidden node parameters can be randomly generated without the knowledge of the training data. That is, for any continuous target function f and any randomly generated sequence $\{(\mathbf{a}_i, b_i)_{i=1}^{L}\}$, $\lim_{L\to\infty} \|f(\mathbf{x}) - f_L(\mathbf{x})\| = \lim_{L\to\infty} \|f(\mathbf{x}) - \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x})\| = 0$ holds with probability one if β_i is chosen to minimize $\|f(\mathbf{x}) - f_L(\mathbf{x})\|, i = 1, \cdots, L$.

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

G.-B. Huang, et al., "Convex Incremental Extreme Learning Machine," *Neurocomputing*, vol. 70, pp. 3056-3062, 2007.

G.-B. Huang, et al., "Enhanced Random Search Based Incremental Extreme Learning Machine," *Neurocomputing*, vol. 71, pp. 3460-3468, 2008.

New Learning Theory: Learning Without Iterative Tuning

New Learning View

- Learning Without Iterative Tuning: Given any nonconstant piecewise continuous function g, if continuous target function $f(\mathbf{x})$ can be approximated by SLFNs with adjustable hidden nodes g then the hidden node parameters of such SLFNs needn't be tuned.
- All these hidden node parameters can be randomly generated without the knowledge of the training data. That is, for any continuous target function f and any randomly generated sequence $\{(\mathbf{a}_i, b_i)_{i=1}^{L}\}$, $\lim_{L\to\infty} \|f(\mathbf{x}) - f_L(\mathbf{x})\| = \lim_{L\to\infty} \|f(\mathbf{x}) - \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x})\| = 0$ holds with probability one if β_i is chosen to minimize $\|f(\mathbf{x}) - f_L(\mathbf{x})\|$, $i = 1, \cdots, L$.

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

G.-B. Huang, et al., "Convex Incremental Extreme Learning Machine," *Neurocomputing*, vol. 70, pp. 3056-3062, 2007.

G.-B. Huang, et al., "Enhanced Random Search Based Incremental Extreme Learning Machine," *Neurocomputing*, vol. 71, pp. 3460-3468, 2008.

Unified Learning Platform

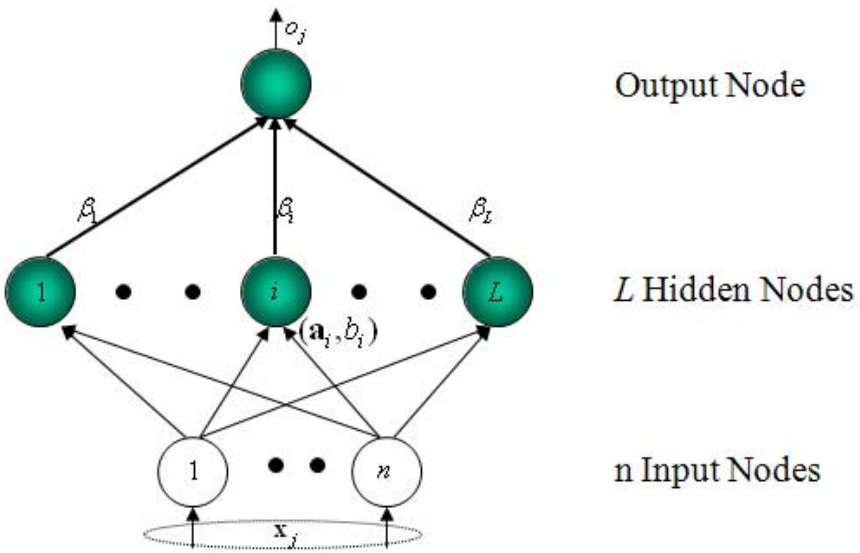


Figure 9: Generalized SLFN: any type of piecewise continuous $G(\mathbf{a}_i, b_i, \mathbf{x})$.

Mathematical Model

- For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbf{R}^n \times \mathbf{R}^m$, SLFNs with L hidden nodes each with output function $G(\mathbf{a}_i, b_i, \mathbf{x})$ are mathematically modeled as

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \quad j = 1, \cdots, N \qquad (9)$$

- $(\mathbf{a}_i, b_i)$: hidden node parameters.
- β_i: the weight vector connecting the ith hidden node and the output node.

Unified Learning Platform

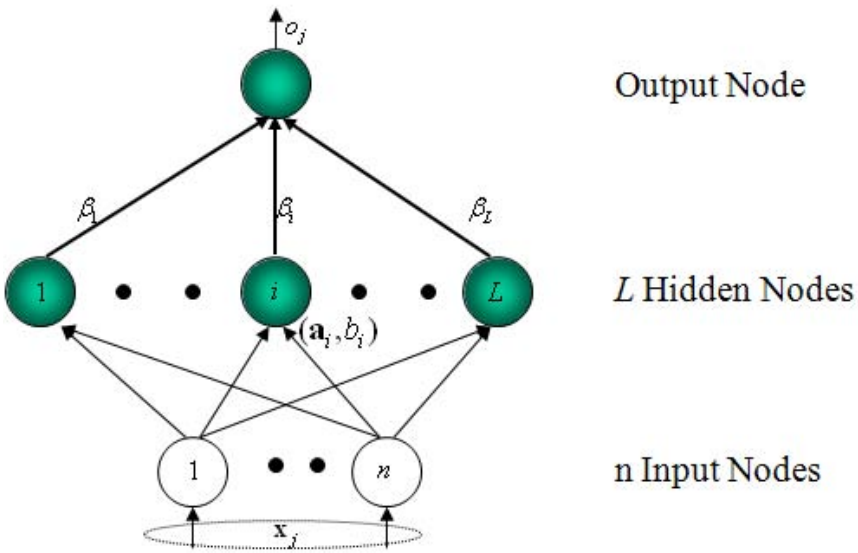


Figure 9: Generalized SLFN: any type of piecewise continuous $G(\mathbf{a}_i, b_i, \mathbf{x})$.

Mathematical Model

- For N arbitrary distinct samples $(\mathbf{x}_i, \mathbf{t}_i) \in \mathbf{R}^n \times \mathbf{R}^m$, SLFNs with L hidden nodes each with output function $G(\mathbf{a}_i, b_i, \mathbf{x})$ are mathematically modeled as

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \quad j = 1, \cdots, N \qquad (9)$$

- $(\mathbf{a}_i, b_i)$: hidden node parameters.
- β_i: the weight vector connecting the ith hidden node and the output 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

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L} \tag{10}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{11}$$

$\mathbf{H}$ 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 $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, hidden node output function $G(\mathbf{a}, b, \mathbf{x})$, and the number of hidden nodes L,

1. Assign randomly hidden node parameters $(\mathbf{a}_i, b_i)$, $i = 1, \cdots, L$.
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.extreme-learning-machines.org

Extreme Learning Machine (ELM)

Three-Step Learning Model

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

1. Assign randomly hidden node parameters $(\mathbf{a}_i, b_i)$, $i = 1, \cdots, L$.
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.extreme-learning-machines.org

Extreme Learning Machine (ELM)

Three-Step Learning Model

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

1. Assign randomly hidden node parameters $(\mathbf{a}_i, b_i)$, $i = 1, \cdots, L$.
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.extreme-learning-machines.org

Extreme Learning Machine (ELM)

Three-Step Learning Model

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

1. Assign randomly hidden node parameters $(\mathbf{a}_i, b_i)$, $i = 1, \cdots, L$.
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.extreme-learning-machines.org

Extreme Learning Machine (ELM)

Three-Step Learning Model

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

1. Assign randomly hidden node parameters $(\mathbf{a}_i, b_i)$, $i = 1, \cdots, L$.
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.extreme-learning-machines.org

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.
- The hidden node parameters $\mathbf{a}_i$ and b_i are not only independent of the training data but also of each other.
- Unlike conventional learning methods which MUST see the training data before generating the hidden node parameters, ELM could generate the hidden node parameters before seeing the training data.
- Unlike traditional gradient-based learning algorithms which only work for differentiable activation functions, ELM works for all bounded nonconstant piecewise continuous activation functions.
- Unlike traditional gradient-based learning algorithms facing several issues like local minima, improper learning rate and overfitting, etc, ELM tends to reach the solutions straightforward without such trivial issues.
- The ELM learning algorithm looks much simpler than other popular learning algorithms: neural networks and support vector machines.

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.
M.-B. Li, et al., "Fully Complex Extreme Learning Machine" *Neurocomputing*, vol. 68, pp. 306-314, 2005.

Output Functions of Generalized SLFNs

Ridge regression theory based ELM

$$\mathbf{f(x)} = \mathbf{h(x)}\beta = \mathbf{h(x)H}^T\left(\mathbf{HH}^T\right)^{-1}\mathbf{T} \Longrightarrow \mathbf{h(x)H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{HH}^T\right)^{-1}\mathbf{T}$$

and

$$\mathbf{f(x)} = \mathbf{h(x)}\beta = \mathbf{h(x)}\left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} \Longrightarrow \mathbf{h(x)}\left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}$$

Ridge Regression Theory

A positive value I/C can be added to the diagonal of $\mathbf{H}^T\mathbf{H}$ or $\mathbf{HH}^T$ of the Moore-Penrose generalized inverse $\mathbf{H}$ the resultant solution is stabler and tends to have better generalization performance.

A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems", *Technometrics*, vol. 12, no. 1, pp. 55-67, 1970.

Citation: G.-B. Huang, H. Zhou, X. Ding and R. Zhang, "Extreme Learning Machine for Regression and Multi-Class Classification", *submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2010.

Output Functions of Generalized SLFNs

Ridge regression theory based ELM

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T} \Longrightarrow \mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}$$

and

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} = \mathbf{h}(\mathbf{x})\left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T} \Longrightarrow \mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}$$

Ridge Regression Theory

A positive value I/C can be added to the diagonal of $\mathbf{H}^T\mathbf{H}$ or $\mathbf{H}\mathbf{H}^T$ of the Moore-Penrose generalized inverse $\mathbf{H}$ the resultant solution is stabler and tends to have better generalization performance.

A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems", *Technometrics*, vol. 12, no. 1, pp. 55-67, 1970.

Citation: G.-B. Huang, H. Zhou, X. Ding and R. Zhang, "Extreme Learning Machine for Regression and Multi-Class Classification", *submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2010.

Output functions of Generalized SLFNs

Valid for both kernel and non-kernel learning

1. Non-kernel based:

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{T}$$

and

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{T}$$

2. Kernel based: (if $\mathbf{h}(\mathbf{x})$ is unknown)

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \Omega_{ELM}\right)^{-1} \mathbf{T}$$

where $\Omega_{ELM\,i,j} = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$

Citation: G.-B. Huang, H. Zhou, X. Ding and R. Zhang, "Extreme Learning Machine for Regression and Multi-Class Classification", *submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2010.

Output functions of Generalized SLFNs

Valid for both kernel and non-kernel learning

1. Non-kernel based:

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{T}$$

and

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{T}$$

2. Kernel based: (if $\mathbf{h}(\mathbf{x})$ is unknown)

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \Omega_{ELM}\right)^{-1} \mathbf{T}$$

where $\Omega_{ELM\,i,j} = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$

Citation: G.-B. Huang, H. Zhou, X. Ding and R. Zhang, "Extreme Learning Machine for Regression and Multi-Class Classification", *submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2010.

Output functions of Generalized SLFNs

Valid for both kernel and non-kernel learning

1. Non-kernel based:

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{T}$$

and

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{T}$$

2. Kernel based: (if $\mathbf{h}(\mathbf{x})$ is unknown)

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \Omega_{ELM}\right)^{-1} \mathbf{T}$$

where $\Omega_{ELM\,i,j} = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$

Citation: G.-B. Huang, H. Zhou, X. Ding and R. Zhang, "Extreme Learning Machine for Regression and Multi-Class Classification", *submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2010.

Outline

Differences between ELM and LS-SVM

ELM (unified for regression, binary/multi-class cases)

1. Non-kernel based:

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{T}$$

and

$$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{T}$$

2. Kernel based: (if $\mathbf{h}(\mathbf{x})$ is unknown)

$$\mathbf{f}(\mathbf{x}) = \begin{bmatrix} K(\mathbf{x}, \mathbf{x}_1) \\ \vdots \\ K(\mathbf{x}, \mathbf{x}_N) \end{bmatrix}^T \left(\frac{\mathbf{I}}{C} + \Omega_{ELM}\right)^{-1} \mathbf{T}$$

where $\Omega_{ELM\,i,j} = \mathbf{h}(\mathbf{x}_i) \cdot \mathbf{h}(\mathbf{x}_j) = K(\mathbf{x}_i, \mathbf{x}_j)$

LS-SVM (for binary class case)

$$\begin{bmatrix} 0 & \mathbf{T}^T \\ \mathbf{T} & \frac{\mathbf{I}}{C} + \Omega_{LS-SVM} \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 & \mathbf{T}^T \\ \mathbf{T} & \frac{\mathbf{I}}{C} + \mathbf{Z}\mathbf{Z}^T \end{bmatrix} \begin{bmatrix} b \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 \\ \vec{1} \end{bmatrix}$$

where

$$\mathbf{Z} = \begin{bmatrix} t_1\phi(\mathbf{x}_1) \\ \vdots \\ t_N\phi(\mathbf{x}_N) \end{bmatrix}$$

$$\Omega_{LS-SVM} = \mathbf{Z}\mathbf{Z}^T$$

For details, refer to: G.-B. Huang, H. Zhou, X. Ding and R. Zhang, "Extreme Learning Machine for Regression and Multi-Class Classification", *submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence*, October 2010.

ELM Classification Boundaries

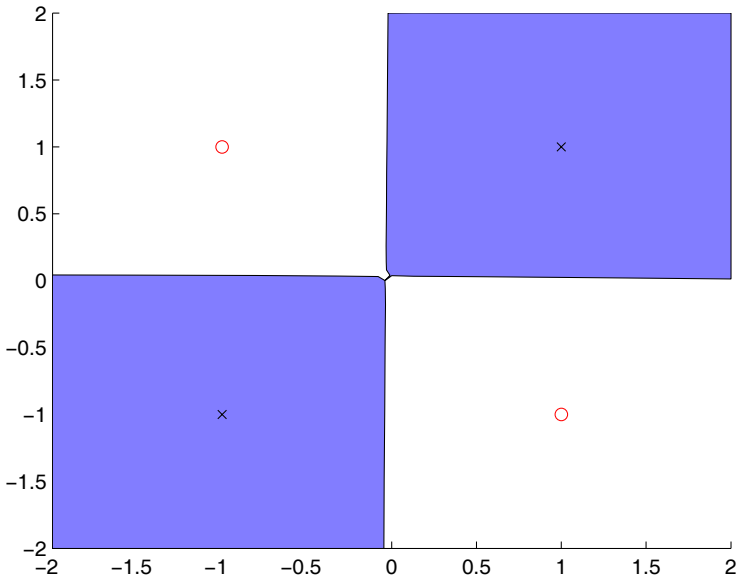


Figure 10: XOR Problem

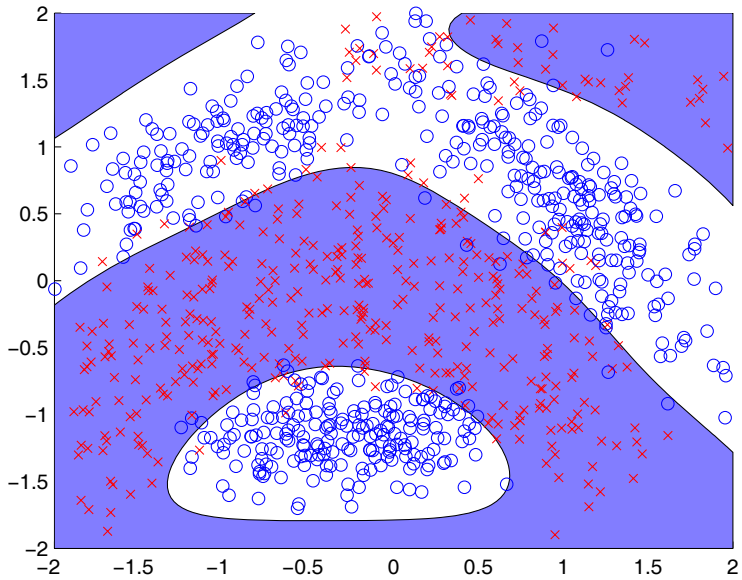


Figure 11: Banana Case

Performance Evaluation of ELM

Datasets	# train	# test	# features	# classes	Random Perm
DNA	2000	1186	180	3	No
Letter	13333	6667	16	26	Yes
Shuttle	43500	14500	9	7	No
USPS	7291	2007	256	10	No

Table 1: Specification of multi-class classification problems

Performance Evaluation of ELM

Datasets	SVM (Gaussian Kernel)			LS-SVM (Gaussian Kernel)			ELM (Sigmoid hidden node)		
	Testing		Training	Testing		Training	Testing		Training
	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)
DNA	92.86	0	7732	93.68	0	6.359	93.81	0.24	1.586
Letter	92.87	0.26	302.9	93.12	0.27	335.838	93.51	0.15	0.7881
shuttle	*99.74*	*0*	*2864.0*	*99.82*	*0*	**24767.0**	*99.64*	*0.01*	**3.3379**
USPS	*96.14*	*0*	**12460**	*96.76*	*0*	*59.1357*	*96.28*	*0.28*	**0.6877**

ELM: non-kernel based

Datasets	SVM (Gaussian Kernel)			LS-SVM (Gaussian Kernel)			ELM (Gaussian Kernel)		
	Testing		Training	Testing		Training	Testing		Training
	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)
DNA	92.86	0	7732	93.68	0	6.359	**96.29**	0	2.156
Letter	92.87	0.26	302.9	93.12	0.27	335.838	**97.41**	0.13	41.89
shuttle	*99.74*	*0*	*2864.0*	*99.82*	*0*	**24767.0**	*99.91*	*0*	*4029.0*
USPS	*96.14*	*0*	**12460**	*96.76*	*0*	*59.1357*	**98.9**	*0*	*9.2784*

ELM: kernel based

Table 2: Performance comparison of SVM, LS-SVM and ELM: multi-class datasets.

Artificial Case: Approximation of 'SinC' Function

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 3: 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.

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 4: 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 5: Comparison of training and testing RMSE of SVR and ELM.

Real-World Regression Problems

Datasets	BP # nodes	SVR (C, γ)	SVR # SVs	ELM # 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 6: 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 7: 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 8: 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.)

Essence of ELM

Key expectations

1. Hidden layer need not be tuned.
2. Hidden layer mapping $\mathbf{h}(\mathbf{x})$ satisfies universal approximation condition.
3. Minimize: $\|\mathbf{H}\beta - \mathbf{T}\|$ and $\|\beta\|$

Essence of ELM

Key expectations

1. Hidden layer need not be tuned.
2. Hidden layer mapping $\mathbf{h}(\mathbf{x})$ satisfies universal approximation condition.
3. Minimize: $\|\mathbf{H}\beta - \mathbf{T}\|$ and $\|\beta\|$

Essence of ELM

Key expectations

1. Hidden layer need not be tuned.
2. Hidden layer mapping $\mathbf{h}(\mathbf{x})$ satisfies universal approximation condition.
3. Minimize: $\|\mathbf{H}\beta - \mathbf{T}\|$ and $\|\beta\|$

Optimization Constraints of Different Methods

ELM variant: Based on Inequality Constraint Conditions

ELM optimization formula:

$$\begin{aligned} &\text{Minimize: } L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{Subject to: } t_i\beta \cdot \mathbf{h}(\mathbf{x}_i) \geq 1 - \xi_i, \quad i = 1, \cdots, N \\ &\qquad\qquad \xi_i \geq 0, \quad i = 1, \cdots, N \end{aligned} \tag{12}$$

The corresponding dual optimization problem:

$$\begin{aligned} &\text{minimize: } L_D = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j \mathbf{h}(\mathbf{x}_i)\cdot\mathbf{h}(\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \\ &\text{subject to: } 0 \leq \alpha_i \leq C, i = 1, \cdots, N \end{aligned} \tag{13}$$

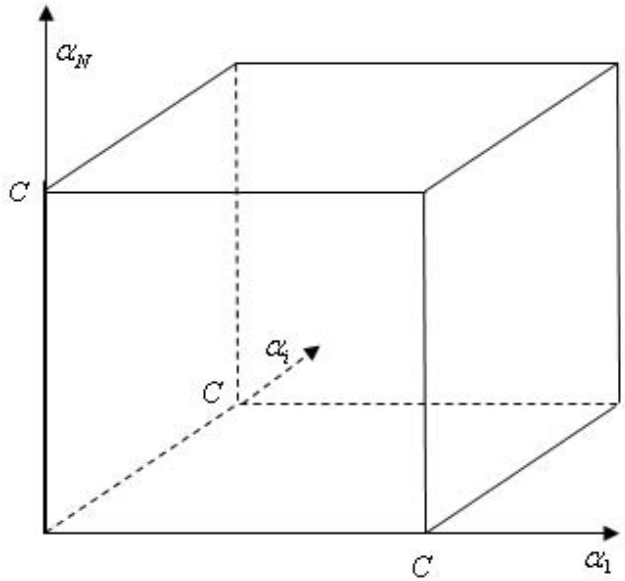


Figure 12: ELM

G.-B. Huang, et al., "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, pp. 155-163, 2010.

Optimization Constraints of Different Methods

SVM Constraint Conditions

SVM optimization formula:

$$\begin{aligned} &\text{Minimize: } L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ &\text{Subject to: } t_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \quad i = 1, \cdots, N \\ &\qquad\qquad \xi_i \geq 0, \quad i = 1, \cdots, N \end{aligned} \tag{14}$$

The corresponding dual optimization problem:

$$\begin{aligned} &\text{minimize: } L_D = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \\ &\text{subject to: } \sum_{i=1}^{N} t_i \alpha_i = 0 \\ &\qquad\qquad 0 \leq \alpha_i \leq C, i = 1, \cdots, N \end{aligned} \tag{15}$$

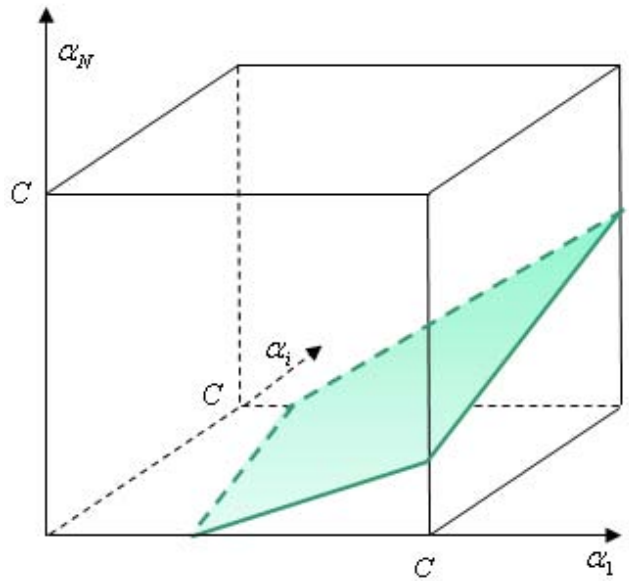


Figure 13: SVM

Optimization Constraints of Different Methods

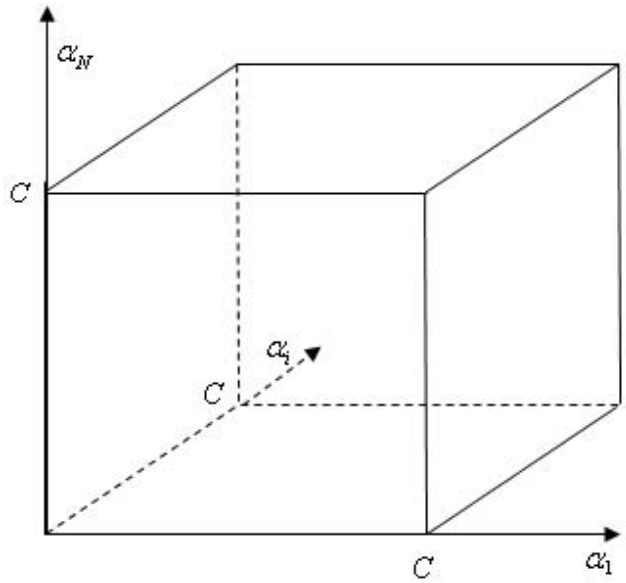


Figure 14: ELM

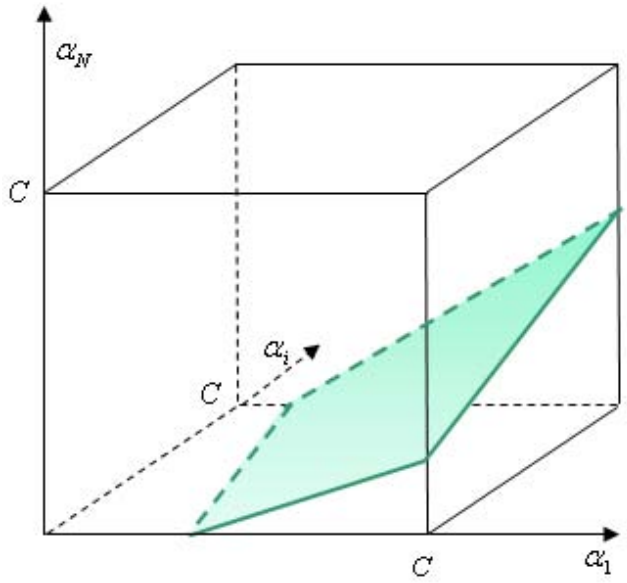


Figure 15: SVM

ELM and SVM have the same dual optimization objective functions, but in ELM optimal α_i are found from the entire cube $[0, C]^N$ while in SVM optimal α_i are found from one hyperplane $\sum_{i=1}^{N} t_i \alpha_i = 0$ within the cube $[0, C]^N$. SVM always provides a suboptimal solution, so does LS-SVM.

Flaws in SVM Theory?

Flaws?

1. SVM is great! Without SVM computational intelligence may not be so successful! Many applications and products may not be so successful either! However ...
2. SVM always searches for the optimal solution in the hyperplane $\sum_{i=1}^{N} \alpha_i t_i = 0$ within the cube $[0, C]^N$ of the SVM feature space.
3. Irrelevant applications may be handled similarly in SVMs. Given two training datasets $\{(\mathbf{x}_i^{(1)}, t_i^{(1)})\}_{i=1}^{N}$ and $\{(\mathbf{x}_i^{(2)}, t_i^{(2)})\}_{i=1}^{N}$ and $\{(\mathbf{x}_i^{(1)}\}_{i=1}^{N}$ and $\{(\mathbf{x}_i^{(2)}\}_{i=1}^{N}$ are totally irrelevant/independent, if $[t_1^{(1)}, \cdots, t_N^{(1)}]^T$ is similar or close to $[t_1^{(2)}, \cdots, t_N^{(2)}]^T$ SVM may have similar search areas of the cube $[0, C]^N$ for two different cases.

G.-B. Huang, et al., "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, pp. 155-163, 2010.

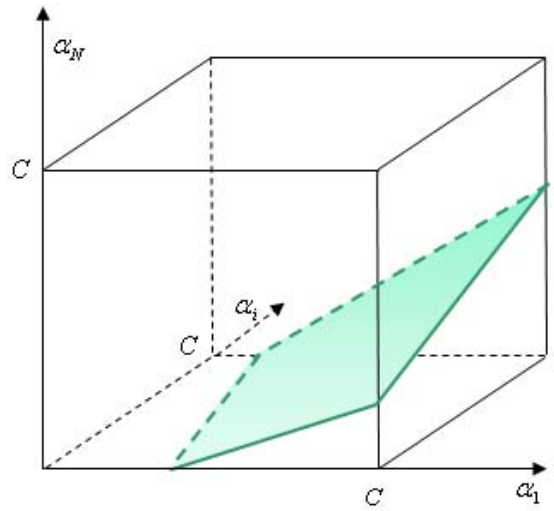


Figure 16: SVM

Reasons

SVM is too "generous" on the feature mappings and kernels, almost condition free except for Mercer's conditions.

1. As the feature mappings and kernels need not satisfy universal approximation condition, b must be present.
2. As b exists, contradictions are caused.

Flaws in SVM Theory?

Flaws?

1. SVM is great! Without SVM computational intelligence may not be so successful! Many applications and products may not be so successful either! However ...
2. SVM always searches for the optimal solution in the hyperplane $\sum_{i=1}^{N} \alpha_i t_i = 0$ within the cube $[0, C]^N$ of the SVM feature space.
3. Irrelevant applications may be handled similarly in SVMs. Given two training datasets $\{(\mathbf{x}_i^{(1)}, t_i^{(1)})\}_{i=1}^{N}$ and $\{(\mathbf{x}_i^{(2)}, t_i^{(2)})\}_{i=1}^{N}$ and $\{(\mathbf{x}_i^{(1)}\}_{i=1}^{N}$ and $\{(\mathbf{x}_i^{(2)}\}_{i=1}^{N}$ are totally irrelevant/independent, if $[t_1^{(1)}, \cdots, t_N^{(1)}]^T$ is similar or close to $[t_1^{(2)}, \cdots, t_N^{(2)}]^T$ SVM may have similar search areas of the cube $[0, C]^N$ for two different cases.

G.-B. Huang, et al., "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, pp. 155-163, 2010.

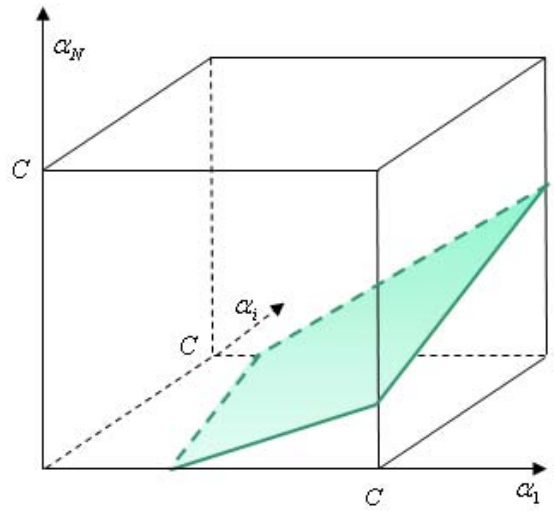


Figure 16: SVM

Reasons

SVM is too "generous" on the feature mappings and kernels, almost condition free except for Mercer's conditions.

1. As the feature mappings and kernels need not satisfy universal approximation condition, b must be present.
2. As b exists, contradictions are caused.

Optimization Constraints of Different Methods

Datasets	# Attributes	# Training data	# Testing data
Breast-cancer	10	300	383
liver-disorders	6	200	145
heart	13	70	200
ionosphere	34	100	251
Pimadata	8	400	368
Pwlinear	10	100	100
Sonar	60	100	158
Monks Problem 1	6	124	432
Monks Problem 2	6	169	432
Splice	60	1000	2175
A1a	123	1605	30956
leukemia	7129	38	34
Colon	2000	30	32

Table 9: Specification of tested binary classification problems

Optimization Constraints of Different Methods

Datasets	SVM (Gaussian Kernel)				ELM (Sigmoid hidden nodes)			
	(C, γ)	Training Time (s)	Testing Rate (%)	Testing Dev (%)	C	Training Time (s)	Testing Rate (%)	Testing Dev (%)
Breast cancer	(5, 50)	0.1118	94.20	0.87	10^{-3}	0.1423	**96.32**	0.75
Liver disorders	(10, 2)	0.0972	68.24	4.58	10	0.1734	**72.34**	2.55
Heart	$(10^4, 5)$	0.0382	76.00	3.85	50	0.0344	**76.25**	2.70
Ionosphere	(5, 5)	0.0218	**90.58**	1.22	10^{-3}	0.0359	89.48	1.12
Pimadata	$(10^3, 50)$	0.2049	76.43	1.57	0.01	0.2867	**77.27**	1.33
Pwlinear	$(10^4, 10^3)$	0.0357	84.35	3.28	10^{-3}	0.0486	**86.00**	1.92
Sonar	(20, 1)	0.0412	**83.33**	3.55	10^4	0.0467	81.53	3.78
Monks Problem 1	(10, 1)	0.0424	**95.37**	0	10^4	0.0821	95.19	0.41
Monks Problem 2	$(10^4, 5)$	0.0860	83.80	0	10^4	0.0920	**85.14**	0.57
Splice	(2, 20)	2.0683	84.05	0	0.01	3.5912	**85.50**	0.54
A1a	(2, 5)	5.6275	84.25	0	0.01	5.4542	**84.36**	0.79

Table 10: Comparison between the conventional SVM and ELM for classification.

Optimization Constraints of Different Methods

Datasets	SVM (Gaussian Kernel)				ELM (Sigmoid hidden nodes)			
	(C, γ)	Training Time (s)	Testing Rate (%)	Testing Dev (%)	(C, L)	Training Time (s)	Testing Rate (%)	Testing Dev (%)
Before Gene Selection								
leukemia	$(10^3, 10^3)$	3.2674	**82.35**	0	(0.01, 3000)	0.2878	81.18	2.37
Colon	$(10^3, 10^3)$	0.2391	81.25	0	(0.01, 3000)	0.1023	**82.50**	2.86
After Gene Selection (60 Genes Obtained for Each Case)								
leukemia	(2, 20)	0.0640	**100**	0	(10, 3000)	0.0199	**100**	0
Colon	(2, 50)	0.0166	87.50	0	(0,001, 500)	0.0161	**89.06**	2.10

Table 11: Comparison between the conventional SVM and ELM in gene classification application.

Online Sequential ELM (OS-ELM) Algorithm

Learning Features

1. The training observations are *sequentially* (one-by-one or chunk-by-chunk with varying or fixed chunk length) presented to the learning algorithm.
2. At any time, only the *newly* arrived single or chunk of observations (instead of the entire past data) are seen and learned.
3. A single or a chunk of training observations is *discarded* as soon as the learning procedure for that particular (single or chunk of) observation(s) is completed.
4. The learning algorithm has *no prior* knowledge as to how many training observations will be presented.

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.

Online Sequential ELM (OS-ELM) Algorithm

Learning Features

1. The training observations are *sequentially* (one-by-one or chunk-by-chunk with varying or fixed chunk length) presented to the learning algorithm.
2. At any time, only the *newly* arrived single or chunk of observations (instead of the entire past data) are seen and learned.
3. A single or a chunk of training observations is *discarded* as soon as the learning procedure for that particular (single or chunk of) observation(s) is completed.
4. The learning algorithm has *no prior* knowledge as to how many training observations will be presented.

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.

Online Sequential ELM (OS-ELM) Algorithm

Learning Features

1. The training observations are *sequentially* (one-by-one or chunk-by-chunk with varying or fixed chunk length) presented to the learning algorithm.
2. At any time, only the *newly* arrived single or chunk of observations (instead of the entire past data) are seen and learned.
3. A single or a chunk of training observations is *discarded* as soon as the learning procedure for that particular (single or chunk of) observation(s) is completed.
4. The learning algorithm has *no prior* knowledge as to how many training observations will be presented.

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.

Online Sequential ELM (OS-ELM) Algorithm

Learning Features

1. The training observations are *sequentially* (one-by-one or chunk-by-chunk with varying or fixed chunk length) presented to the learning algorithm.
2. At any time, only the *newly* arrived single or chunk of observations (instead of the entire past data) are seen and learned.
3. A single or a chunk of training observations is *discarded* as soon as the learning procedure for that particular (single or chunk of) observation(s) is completed.
4. The learning algorithm has *no prior* knowledge as to how many training observations will be presented.

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.

Online Sequential ELM (OS-ELM) Algorithm

Learning Features

1. The training observations are *sequentially* (one-by-one or chunk-by-chunk with varying or fixed chunk length) presented to the learning algorithm.
2. At any time, only the *newly* arrived single or chunk of observations (instead of the entire past data) are seen and learned.
3. A single or a chunk of training observations is *discarded* as soon as the learning procedure for that particular (single or chunk of) observation(s) is completed.
4. The learning algorithm has *no prior* knowledge as to how many training observations will be presented.

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.

OS-ELM Algorithm

Two-Step Learning Model

1. Initialization phase: where batch ELM is used to initialize the learning system.
2. Sequential learning phase: where recursive least square (RLS) method is adopted to update the learning system sequentially.

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.

OS-ELM Algorithm

Two-Step Learning Model

1. Initialization phase: where batch ELM is used to initialize the learning system.
2. Sequential learning phase: where recursive least square (RLS) method is adopted to update the learning system sequentially.

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.

OS-ELM Algorithm

Two-Step Learning Model

1. Initialization phase: where batch ELM is used to initialize the learning system.
2. Sequential learning phase: where recursive least square (RLS) method is adopted to update the learning system sequentially.

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.

Summary

- For generalized SLFNs, learning can be done without iterative tuning.
- ELM is efficient for batch mode learning, sequential learning, incremental learning.
- ELM provides a unified learning model for regression, binary/multi-class classification.
- ELM works with different hidden nodes including kernels.
- Real-time learning capabilities.
- ELM always provides better generalization performance than SVM and LS-SVM if the same kernel is used?
- ELM always has faster learning speed than LS-SVM if the same kernel is used?
- ELM is the simplest learning technique for generalized SLFNs?

Summary

- For generalized SLFNs, learning can be done without iterative tuning.
- ELM is efficient for batch mode learning, sequential learning, incremental learning.
- ELM provides a unified learning model for regression, binary/multi-class classification.
- ELM works with different hidden nodes including kernels.
- Real-time learning capabilities.
- ELM always provides better generalization performance than SVM and LS-SVM if the same kernel is used?
- ELM always has faster learning speed than LS-SVM if the same kernel is used?
- ELM is the simplest learning technique for generalized SLFNs?

Summary

- For generalized SLFNs, learning can be done without iterative tuning.
- ELM is efficient for batch mode learning, sequential learning, incremental learning.
- ELM provides a unified learning model for regression, binary/multi-class classification.
- ELM works with different hidden nodes including kernels.
- Real-time learning capabilities.
- ELM always provides better generalization performance than SVM and LS-SVM if the same kernel is used?
- ELM always has faster learning speed than LS-SVM if the same kernel is used?
- ELM is the simplest learning technique for generalized SLFNs?

Summary

- For generalized SLFNs, learning can be done without iterative tuning.
- ELM is efficient for batch mode learning, sequential learning, incremental learning.
- ELM provides a unified learning model for regression, binary/multi-class classification.
- ELM works with different hidden nodes including kernels.
- Real-time learning capabilities.
- ELM always provides better generalization performance than SVM and LS-SVM if the same kernel is used?
- ELM always has faster learning speed than LS-SVM if the same kernel is used?
- ELM is the simplest learning technique for generalized SLFNs?

Summary

- For generalized SLFNs, learning can be done without iterative tuning.
- ELM is efficient for batch mode learning, sequential learning, incremental learning.
- ELM provides a unified learning model for regression, binary/multi-class classification.
- ELM works with different hidden nodes including kernels.
- Real-time learning capabilities.
- ELM always provides better generalization performance than SVM and LS-SVM if the same kernel is used?
- ELM always has faster learning speed than LS-SVM if the same kernel is used?
- ELM is the simplest learning technique for generalized SLFNs?

Summary

- For generalized SLFNs, learning can be done without iterative tuning.
- ELM is efficient for batch mode learning, sequential learning, incremental learning.
- ELM provides a unified learning model for regression, binary/multi-class classification.
- ELM works with different hidden nodes including kernels.
- Real-time learning capabilities.
- ELM always provides better generalization performance than SVM and LS-SVM if the same kernel is used?
- ELM always has faster learning speed than LS-SVM if the same kernel is used?
- ELM is the simplest learning technique for generalized SLFNs?

Summary

- For generalized SLFNs, learning can be done without iterative tuning.
- ELM is efficient for batch mode learning, sequential learning, incremental learning.
- ELM provides a unified learning model for regression, binary/multi-class classification.
- ELM works with different hidden nodes including kernels.
- Real-time learning capabilities.
- ELM always provides better generalization performance than SVM and LS-SVM if the same kernel is used?
- ELM always has faster learning speed than LS-SVM if the same kernel is used?
- ELM is the simplest learning technique for generalized SLFNs?

Summary

- For generalized SLFNs, learning can be done without iterative tuning.
- ELM is efficient for batch mode learning, sequential learning, incremental learning.
- ELM provides a unified learning model for regression, binary/multi-class classification.
- ELM works with different hidden nodes including kernels.
- Real-time learning capabilities.
- ELM always provides better generalization performance than SVM and LS-SVM if the same kernel is used?
- ELM always has faster learning speed than LS-SVM if the same kernel is used?
- ELM is the simplest learning technique for generalized SLFNs?