# Extreme Learning Machine: Towards Tuning-Free Learning

## - A Unified Learning Technique for Regression and Multiclass Classification

Guang-Bin HUANG

School of Electrical and Electronic Engineering
Nanyang Technological University, Singapore

# Outline

1. Feedforward Neural Networks
   - Single-Hidden Layer Feedforward Networks (SLFNs)
   - Function Approximation of SLFNs
   - Classification Capability of SLFNs
   - Conventional Learning Algorithms of SLFNs
2. Extreme Learning Machine
   - Generalized SLFNs
   - New Learning Theory: Learning Without Iterative Tuning
   - ELM Algorithm
3. ELM, SVM and LS-SVM
4. Online Sequential ELM

# Outline

# Outline

1. Feedforward Neural Networks
   - Single-Hidden Layer Feedforward Networks (SLFNs)
   - Function Approximation of SLFNs
   - Classification Capability of SLFNs
   - Conventional Learning Algorithms of SLFNs
2. Extreme Learning Machine
   - Generalized SLFNs
   - New Learning Theory: Learning Without Iterative Tuning
   - ELM Algorithm
3. ELM, SVM and LS-SVM
4. Online Sequential ELM

# Outline

1. Feedforward Neural Networks
   - Single-Hidden Layer Feedforward Networks (SLFNs)
   - Function Approximation of SLFNs
   - Classification Capability of SLFNs
   - Conventional Learning Algorithms of SLFNs
2. Extreme Learning Machine
   - Generalized SLFNs
   - New Learning Theory: Learning Without Iterative Tuning
   - ELM Algorithm
3. ELM, SVM and LS-SVM
4. Online Sequential ELM

# Outline

1. Feedforward Neural Networks
   - Single-Hidden Layer Feedforward Networks (SLFNs)
   - Function Approximation of SLFNs
   - Classification Capability of SLFNs
   - Conventional Learning Algorithms of SLFNs
2. Extreme Learning Machine
   - Generalized SLFNs
   - New Learning Theory: Learning Without Iterative Tuning
   - ELM Algorithm
3. ELM, SVM and LS-SVM
4. Online Sequential ELM

# Feedforward Neural Networks with Additive Nodes

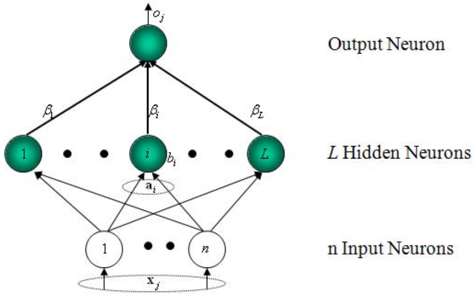



Figure 1: SLFN: additive hidden nodes

**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 $i$th hidden node and the input nodes.
$b_i$: the threshold of the $i$th hidden node.

**Output of SLFNs**

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

$\beta_i$: the weight vector connecting the $i$th hidden node and the output nodes.

# Feedforward Neural Networks with Additive Nodes

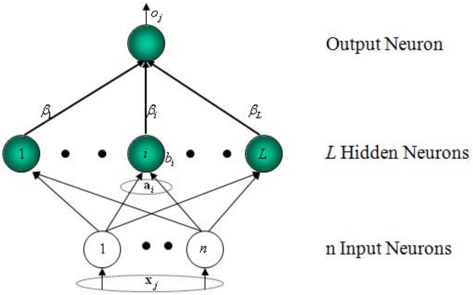



Figure 1: SLFN: additive hidden nodes

**Output of hidden nodes**

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

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

**Output of SLFNs**

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

$\beta_i$: the weight vector connecting the $i$th hidden node and the output nodes.

# Feedforward Neural Networks with Additive Nodes

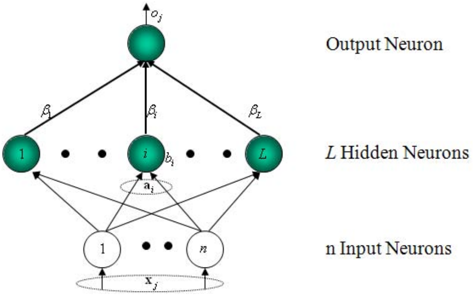



Figure 1: SLFN: additive hidden nodes

**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 $i$th hidden node and the input nodes.
$b_i$: the threshold of the $i$th hidden node.

**Output of SLFNs**

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

$\beta_i$: the weight vector connecting the $i$th hidden node and the output nodes.

# Feedforward Neural Networks with RBF Nodes

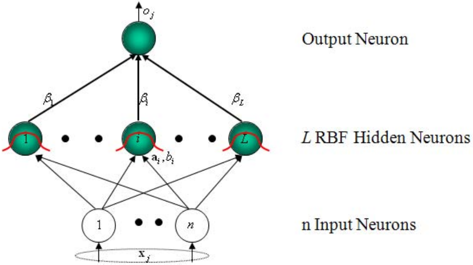



Figure 2: Feedforward Network Architecture: RBF hidden nodes

**Output of hidden nodes**

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

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

**Output of SLFNs**

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

$\beta_i$: the weight vector connecting the $i$th hidden node and the output nodes.

# Feedforward Neural Networks with RBF Nodes

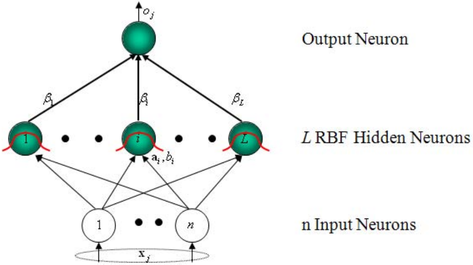



Figure 2: Feedforward Network Architecture: RBF hidden nodes

**Output of hidden nodes**

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

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

**Output of SLFNs**

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

$\beta_i$: the weight vector connecting the $i$th hidden node and the output nodes.

# Feedforward Neural Networks with RBF Nodes

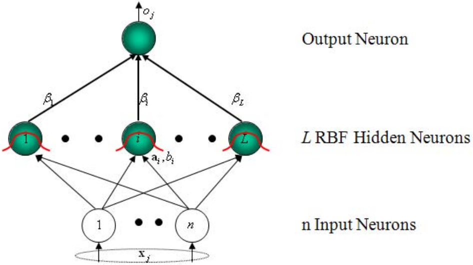



Figure 2: Feedforward Network Architecture: RBF hidden nodes

**Output of hidden nodes**

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

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

**Output of SLFNs**

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

$\beta_i$: the weight vector connecting the $i$th hidden node and the output nodes.

# Outline

1. Feedforward Neural Networks
   - Single-Hidden Layer Feedforward Networks (SLFNs)
   - Function Approximation of SLFNs
   - Classification Capability of SLFNs
   - Conventional Learning Algorithms of SLFNs
2. Extreme Learning Machine
   - Generalized SLFNs
   - New Learning Theory: Learning Without Iterative Tuning
   - ELM Algorithm
3. ELM, SVM and LS-SVM
4. Online Sequential ELM

# Function Approximation of Neural Networks

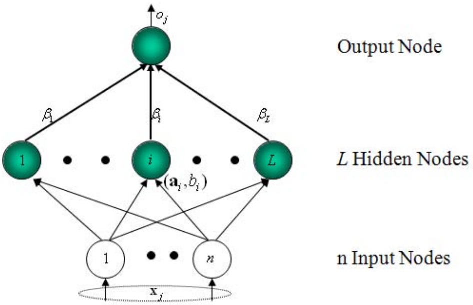



Figure 3: SLFN.

**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 $\epsilon$, for SLFNs with enough number of hidden nodes ($L$) we have

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

**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

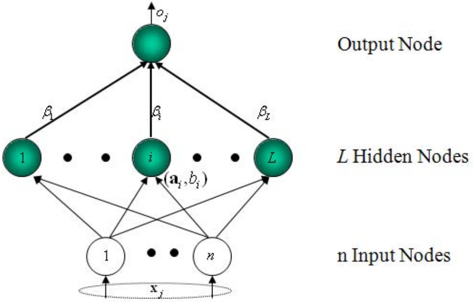



Figure 3: SLFN.

**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 $\epsilon$, for SLFNs with enough number of hidden nodes ($L$) we have

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

**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

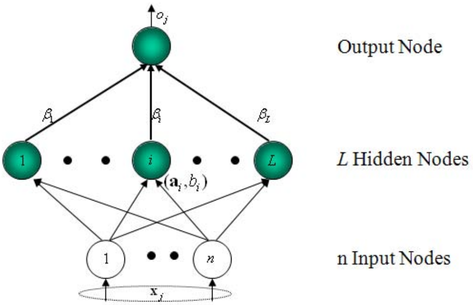



Figure 3: SLFN.

**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 $\epsilon$, 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

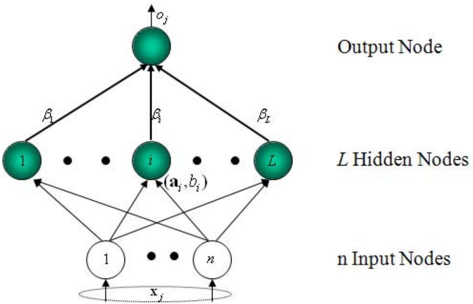



Figure 4: SLFN.

### 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

$$f_L(\mathbf{x}_j) = \mathbf{o}_j, \forall j \qquad (6)$$

- 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: $\beta_i$, $\mathbf{a}_i$, $b_i$.

# Function Approximation of Neural Networks

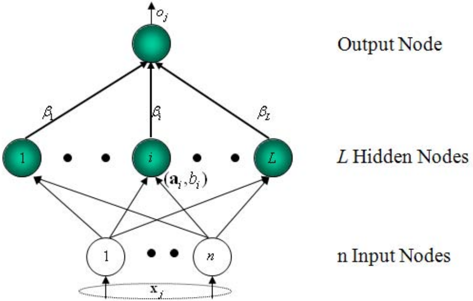



Figure 4: SLFN.

**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

$$f_L(\mathbf{x}_j) = \mathbf{o}_j, \forall j \tag{6}$$

- 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: $\beta_i$, $\mathbf{a}_i$, $b_i$.

# Function Approximation of Neural Networks

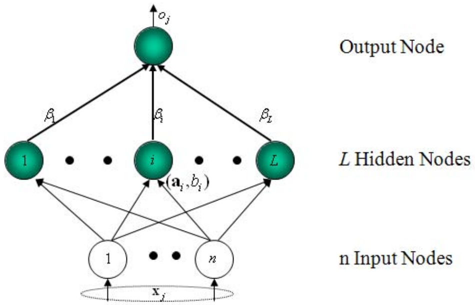



Figure 4: SLFN.

**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

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

- 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: $\beta_i$, $\mathbf{a}_i$, $b_i$.

# Function Approximation of Neural Networks

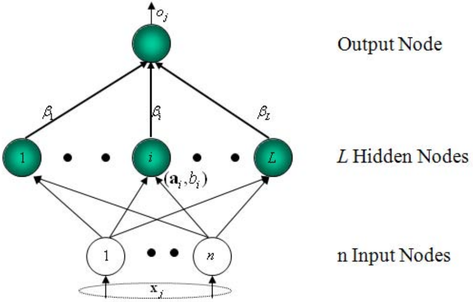



Figure 4: SLFN.

**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

$$f_L(\mathbf{x}_j) = \mathbf{o}_j, \forall j \tag{6}$$

- 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: $\beta_i$, $\mathbf{a}_i$, $b_i$.

# Outline

1. Feedforward Neural Networks
   - Single-Hidden Layer Feedforward Networks (SLFNs)
   - Function Approximation of SLFNs
   - Classification Capability of SLFNs
   - Conventional Learning Algorithms of SLFNs
2. Extreme Learning Machine
   - Generalized SLFNs
   - New Learning Theory: Learning Without Iterative Tuning
   - ELM Algorithm
3. ELM, SVM and LS-SVM
4. Online Sequential ELM

# Classification Capability of SLFNs

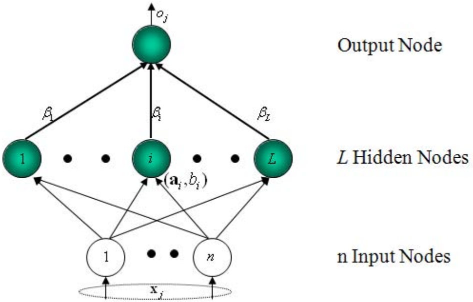



Figure 5: SLFN.

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

1. Feedforward Neural Networks
   - Single-Hidden Layer Feedforward Networks (SLFNs)
   - Function Approximation of SLFNs
   - Classification Capability of SLFNs
   - Conventional Learning Algorithms of SLFNs
2. Extreme Learning Machine
   - Generalized SLFNs
   - New Learning Theory: Learning Without Iterative Tuning
   - ELM Algorithm
3. ELM, SVM and LS-SVM
4. Online Sequential ELM

# Learning Algorithms of Neural Networks

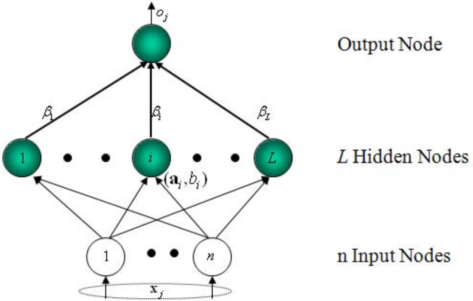



Figure 6: Feedforward Network Architecture.

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

# Learning Algorithms of Neural Networks

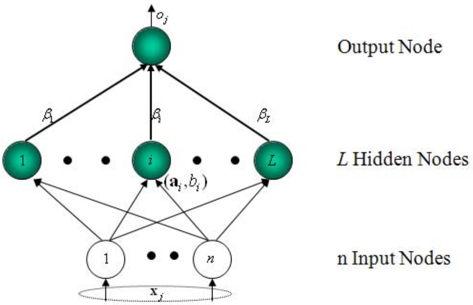



Figure 6: Feedforward Network Architecture.

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

# Learning Algorithms of Neural Networks

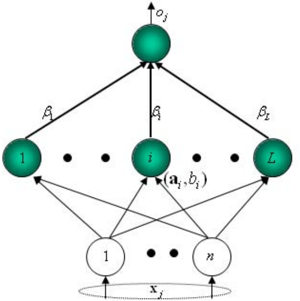



Figure 6: Feedforward Network Architecture.

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

# Learning Algorithms of Neural Networks

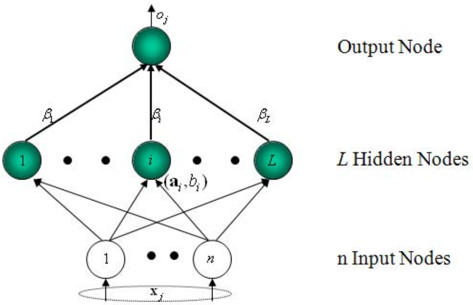



Figure 6: Feedforward Network Architecture.

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

# Support Vector Machine

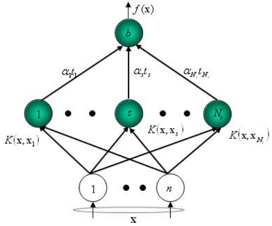



Figure 7: SVM Architecture.

SVM optimization formula:

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

The decision function of SVM is: $f(\mathbf{x}) = \text{sign}\left(\sum_{s=1}^{N_s} \alpha_s t_s K(\mathbf{x}, \mathbf{x}_s) + b\right)$

Q. Liu, et al., "Extreme support vector machine classifier," *LNCS*, vol. 5012, pp. 222-233, 2008.

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.

# 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

# Generalized SLFNs

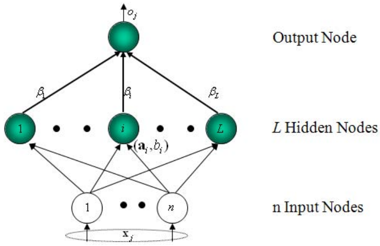



Figure 8: SLFN: any type of piecewise continuous $G(\mathbf{a}_i, b_i, \mathbf{x})$.

**General Hidden Layer Mapping**

- Output function of SLFNs: $f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x})$
- 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 functions of hidden nodes can be but are not limited to:
  Sigmoid: $G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i)$
  RBF: $G(\mathbf{a}_i, b_i, \mathbf{x}) = g(b_i \|\mathbf{x} - \mathbf{a}_i\|)$

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

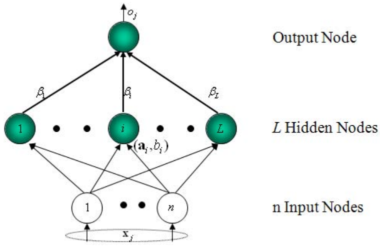



Figure 8: SLFN: any type of piecewise continuous $G(\mathbf{a}_i, b_i, \mathbf{x})$.

**General Hidden Layer Mapping**

- Output function of SLFNs:
  $f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x})$
- 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 functions of hidden nodes can be but are not limited to:
  Sigmoid: $G(\mathbf{a}_i, b_i, \mathbf{x}) = g(\mathbf{a}_i \cdot \mathbf{x} + b_i)$
  RBF: $G(\mathbf{a}_i, b_i, \mathbf{x}) = g(b_i \|\mathbf{x} - \mathbf{a}_i\|)$

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

1. Feedforward Neural Networks
   - Single-Hidden Layer Feedforward Networks (SLFNs)
   - Function Approximation of SLFNs
   - Classification Capability of SLFNs
   - Conventional Learning Algorithms of SLFNs
2. Extreme Learning Machine
   - Generalized SLFNs
   - New Learning Theory: Learning Without Iterative Tuning
   - ELM Algorithm
3. ELM, SVM and LS-SVM
4. Online Sequential ELM

# 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 $\beta_i$ is chosen to minimize $\|f(\mathbf{x}) - f_L(\mathbf{x})\|, \forall i$.

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 $\beta_i$ is chosen to minimize $\|f(\mathbf{x}) - f_L(\mathbf{x})\|, \forall i$.

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

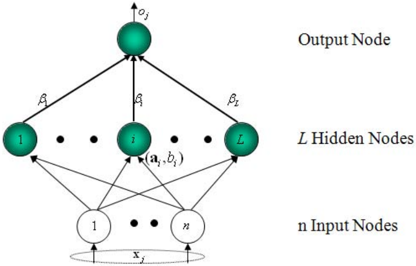



Figure 9: Generalized SLFN: any type of piecewise continuous $G(\mathbf{a}_i, b_i, \mathbf{x})$.

**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

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \quad j = 1, \cdots, N \qquad (8)$$

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

# Unified Learning Platform

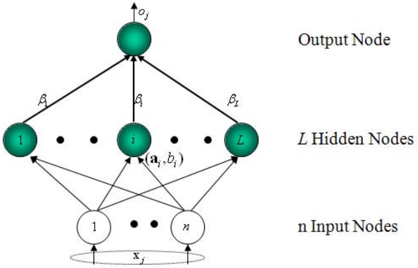



Figure 9: Generalized SLFN: any type of piecewise continuous $G(\mathbf{a}_i, b_i, \mathbf{x})$.

## 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

$$\sum_{i=1}^{L} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}_j) = \mathbf{t}_j, \quad j = 1, \cdots, N \qquad (8)$$

- $(\mathbf{a}_i, b_i)$: hidden node parameters.
- $\beta_i$: the weight vector connecting the $i$th 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

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ \vdots & \cdots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \cdots & G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{bmatrix}_{N \times L} \tag{9}$$

$$\beta = \begin{bmatrix} \beta_1^T \\ \vdots \\ \beta_L^T \end{bmatrix}_{L \times m} \text{ and } \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^T \\ \vdots \\ \mathbf{t}_N^T \end{bmatrix}_{N \times m} \tag{10}$$

$\mathbf{H}$ is called the hidden layer output matrix of the neural network; the $i$th column of $\mathbf{H}$ is the output of the $i$th 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 $\beta$: $\beta = \mathbf{H}^\dagger \mathbf{T}$.

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 $\beta$: $\beta = \mathbf{H}^\dagger \mathbf{T}$.

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 $\beta$: $\beta = \mathbf{H}^\dagger \mathbf{T}$.

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 $\beta$: $\beta = \mathbf{H}^\dagger \mathbf{T}$.

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 $\beta$: $\beta = \mathbf{H}^\dagger \mathbf{T}$.

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 many 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

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

and

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

## Ridge Regression Theory

A positive value $\frac{\mathbf{I}}{C}$ 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.

# Output Functions of Generalized SLFNs

**Ridge regression theory based ELM**

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

and

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

**Ridge Regression Theory**

A positive value $\frac{\mathbf{I}}{C}$ 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.

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

G.-B. Huang, et al., "Extreme learning machine for regression and multiclass classification", *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol. 42, no. 2, pp. 513-529, 2012.
K.-A. Toh, "Deterministic Neural Classification", *Neural Computation*, vol. 20, no. 6, pp. 1565-1595, 2008.
G.-B. Huang, et al., "Extreme Learning Machines: A Survey", *International Journal of Machine Leaning and Cybernetics*, pp. 107-122, vol. 2, no. 2, 2011.

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

G.-B. Huang, et al., "Extreme learning machine for regression and multiclass classification", *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol. 42, no. 2, pp. 513-529, 2012.
K.-A. Toh, "Deterministic Neural Classification", *Neural Computation*, vol. 20, no. 6, pp. 1565-1595, 2008.
G.-B. Huang, et al., "Extreme Learning Machines: A Survey", *International Journal of Machine Leaning and Cybernetics*, pp. 107-122, vol. 2, no. 2, 2011.

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

G.-B. Huang, et al., "Extreme learning machine for regression and multiclass classification", *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol. 42, no. 2, pp. 513-529, 2012.
K.-A. Toh, "Deterministic Neural Classification", *Neural Computation*, vol. 20, no. 6, pp. 1565-1595, 2008.
G.-B. Huang, et al., "Extreme Learning Machines: A Survey", *International Journal of Machine Leaning and Cybernetics*, pp. 107-122, vol. 2, no. 2, 2011.

# ELM Classification Boundaries

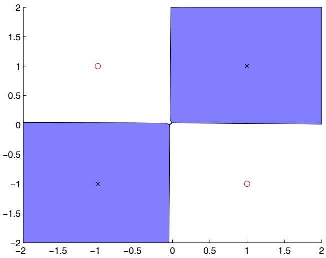



Figure 10: XOR Problem

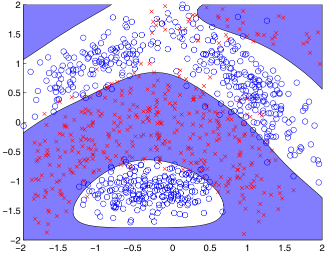



Figure 11: Banana Case

# Performance Evaluation of ELM

| Datasets | # train | # test | # features | # classes | Random Perm |
|---|---|---|---|---|---|
| Letter | 13333 | 6667 | 16 | 26 | Yes |
| Shuttle | 43500 | 14500 | 9 | 7 | No |
| USPS | 7291 | 2007 | 256 | 10 | No |
| MNIST | 60,000 | 10,000 | 784 | 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) |
| 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** | *96.51* | *0* | **80.4** | *96.76* | *0* | *59.1357* | *96.28* | *0.28* | **0.6877** |

| 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) |
| 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** | *96.51* | *0* | **80.4** | *96.76* | *0* | *59.1357* | **98.9** | *0* | *9.2784* |

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

G.-B. Huang, et al., "Extreme learning machine for regression and multiclass classification", *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol. 42, no. 2, pp. 513-529, 2012.

# Handwritten Characters Recognition

| Datasets | SVM (Gaussian Kernel) | | | Deep Learning | | | ELM (Gaussian Kernel) | | |
|---|---|---|---|---|---|---|---|---|---|
| | Testing | | Training Time | Testing | | Training Time | Testing | | Training Time |
| | Rate (%) | Dev (%) | | Rate (%) | Dev (%) | | Rate (%) | Dev (%) | |
| **MNIST** | *98.6*[a] | - | - | *98.8*[a,b] | - | a week | *98.78*[c] | - | 5 mins |

[a] G. E. Hinton, Science, Vol. 313, July 2006; [b] Deep learning method.
[c] Courtesy to Li Deng, Microsoft Research, Redmond, USA, for running ELM in a computer with large memory.

Table 3: Performance comparison of SVM and ELM in MNIST OCR Applications.

Figure 12: Sample images in MNIST dataset (from www.zjucadcg.cn/dengcai/Data/MNIST/images.html)

# Face Recognition

| Methods | YALE | ORL |
|---|---|---|
| Standard eigenface | 76 | 92.2 |
| Waveletface | 83.3 | 92.5 |
| Curveletface | 82.6 | 94.5 |
| Waveletface + PCA | 84 | 94.5 |
| Waveletface + LDA | 84.6 | 94.7 |
| Waveletface + weighted modular PCA | 83.6 | 95 |
| Curveletface + LDA | 83.5 | 95.6 |
| Waveletface + KAM | 84 | 96.6 |
| Curveletface + PCA | 83.9 | 96.6 |
| Curveletface + PCA + LDA | 92 | 97.7 |
| **Curveletface + B2DPCA + ELM** | **99.7** | **99.9** |

Table 4: Testing accuracy (%) of different methods for YALE and ORL face database.

A. A. Mohammed, et al., "Human face recognition based on multidimensional PCA and extreme learning machine," *Pattern Recognition*, vol. 44, pp. 2588-2597, 2011.

Figure 13: Face samples from YALE.

# Face Recognition

| Components | Number of hidden nodes | | | | | | |
|---|---|---|---|---|---|---|---|
| | 35 | 40 | 45 | 50 | 55 | 60 | Dev |
| 5 | 92.56 | 92.92 | 92.97 | 92.95 | 92.62 | 92.55 | 0.2047 |
| 10 | 99.80 | 99.78 | 99.93 | 99.77 | 99.81 | 99.75 | 0.0641 |
| 15 | 99.01 | 99.07 | 99.01 | 98.98 | 99.04 | 98.94 | 0.0454 |
| 20 | 99.97 | 99.95 | 99.96 | 99.97 | 99.89 | 99.95 | 0.0299 |
| 25 | 100 | 100 | 100 | 100 | 100 | 100 | 0 |

Table 5: Average recognition rates(%) for JAFFE database at varying number of hidden nodes: random Sigmoid hidden nodes

# Automatic Object Recognition

| Dataset | ELM Based | AdaBoost Based | Joint Boosting | Scale-Invariant Learning |
|---|---|---|---|---|
| Bikes | 94.6 | 93.4 | 92.5 | 73.9 |
| Planes | 95.3 | 90.0 | 90.2 | 92.7 |
| Cars | 99.0 | 96.0 | 90.3 | 97.0 |
| Leaves | 98.3 | 94.2 | - | 97.8 |
| Faces | 97.9 | 98.0 | 96.4 | - |

Table 6: Accuracy comparison (%) of different approaches

R. Minhas, et al., "A fast recognition framework based on extreme learning machine using hybrid object information," *Neurocomputing*, vol. 73, pp. 1831-1839, 2010.

Figure 14: Sample images from CalTech database.

# Leukocyte Image Segmentation

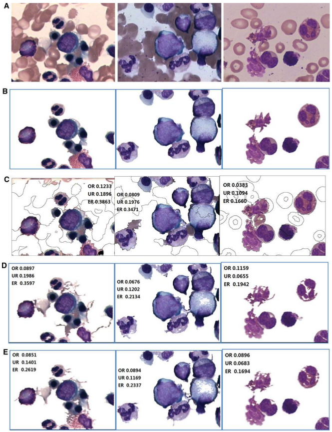



Figure 15: Leukocyte image segmentation

Some examples:

a. Original Leukocyte images from bone marrow smears.
b. Manual segmentation results as ground truth.
c. Segmentation results based on marker-controlled watershed.
d. Segmentation results based on SVM.
e. Segmentation results based on ELM

C. Pan, et al., "Leukocyte image segmentation by visual attention and extreme learning machine," *Neural Computing and Applications*, 2011.

# Image Super-Resolution by ELM

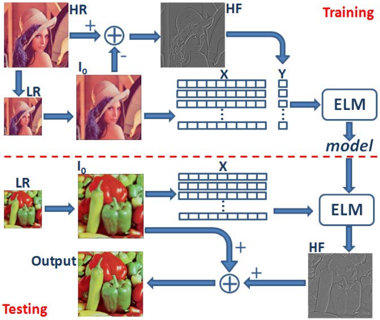



Figure 16: The system diagram of the proposed super-resolution algorithm.

L. An and B. Bhanu, "Image super-resolution by extreme learning machine," *2012 IEEE International Conference on Image Processing*, September 30 - October 3, 2012, Orlando, Florida, USA

# Image Super-Resolution by ELM

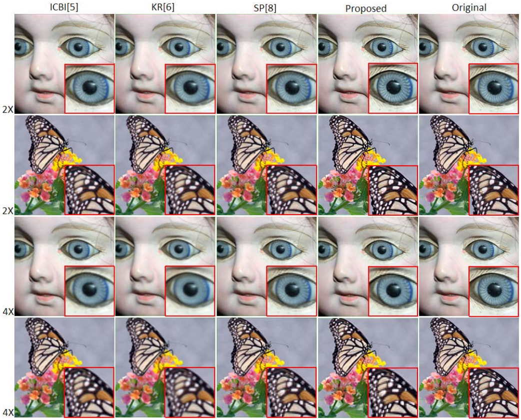



Figure 17: From top to down: super-resolution at 2x and 4x. State-of-the-art methods: iterative curve based interpolation (ICBI), kernel regression based method (KR), compressive sensing based sparse representation method (SR).

# Real-Time Remote Satellite Sensing

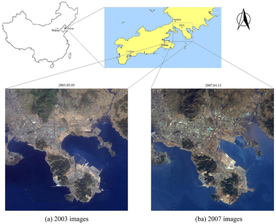



Figure 18: Location of the Dalian Development Area (DDA), China, and the corresponding SPOT-5 (Satellite for earth observation-5) images from 2003 and 2007.

N.-B. Chang, "Satellite-based multitemporal-change detection in urban environments,"

http://spie.org/x44379.xml?pf=true&ArticleID=x44379, Feb 2011.

## Challenging Problems

- Improving land management depends critically on the capacity of (near-)real-time monitoring of land-use/land cover (LULC) change.
- From multitemporal to rapid-change detection, both the resolution of satellite sensors and the computational capacity of the classifiers used for image processing must be well integrated.
- Early remote-sensing image-classification studies employed statistical methods, such as the maximum-likelihood classifier, KNN, and the K-means clustering approach.
- In recent years, methods based on artificial-intelligence and machine-learning techniques have become popular. Approaches based on neural computing, fuzzy logic, evolutionary algorithms, and expert systems are widely used.
- Existing processing techniques using LULC methods are often time-consuming, laborious, and tedious to use, resulting in the unavailability of the results within the designated time window.

# Real-Time Remote Satellite Sensing

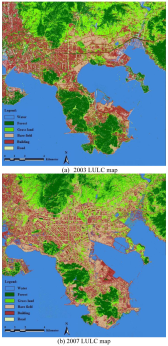



Figure 19: Final classification results produced from (a) the 2003 and (b) 2007 image sets using the PL-ELM.

## ELM Solution

- Classifier: Partial Lanczos extreme-learning machine (PL-ELM, Tang and Han, *Neurcomputing*, 2009).
- Texture features and vegetation indices were extracted.
- More features to the image pixels were added and the “normalized differential vegetation index,” as well as four commonly used texture features (angular second moment, contrast, correlation, and homogeneity) were introduced. The feature-space dimension for all data points/pixels were expanded to eight.
- The LULC features into six major categories, including water bodies, forests, grasslands, bare fields, buildings, and roads.
- PL-ELM classification approach outperforms five other major algorithms, including the BP, maximum-likelihood, KNN and naive Bayes’ algorithms, as well as SVM.
- This case study in the DDA based on images collected in 2003 and 2007 fully supports the monitoring needs and aids in rapid-change detection in terms of both urban expansion and coastal-land reclamation.
- (Near) real-time remotely sensed information can be employed to speed the decision making process for problem resolution.
- ELM based real-time remote sensing can contribute to improved coastal and land management, hazard mitigation, emergency response, and ecosystem-service design.

# EEG Based Epileptic Seizure Detection

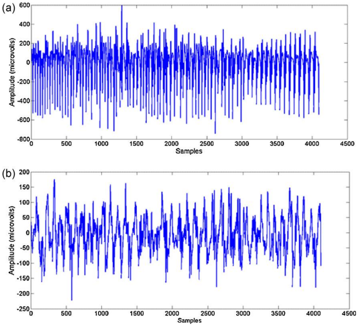



Figure 20: Sample EEG recordings. (a) Ictal EEG (Set S) (b) Interictal EEG (Set F).

Y. Song, et al., "Automatic epileptic seizure detection in EEGs based on optimized sample entropy and extreme learning machine," *Journal of Neuroscience Methods*, vol. 210, pp. 132-146, 2012.

# EEG Based Epileptic Seizure Detection

Figure 21: Intracranial electrode placements.

Y. Song, et al., "Automatic epileptic seizure detection in EEGs based on optimized sample entropy and extreme learning machine," *Journal of Neuroscience Methods*, vol. 210, pp. 132-146, 2012.

# EEG Based Epileptic Seizure Detection

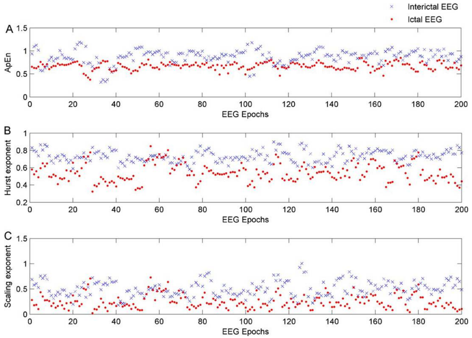



Figure 22: Nonlinear features extracted from EEG signals (approximate entropy (ApEn), Hurst exponent and scaling exponent obtained with detrended fluctuation analysis (DFA)) are employed to characterize interictal and ictal EEGs.

Q. Yuan, et al., "Epileptic EEG classification based on extreme learning machine and nonlinear feature," *Epilepsy Research*, vol. 96, pp. 29-38, 2011.

# Epileptic EEG Classification

| Classifiers | Sensitivity (%) | Specificity (%) | Accuracy (%) | Train Time (s) | Test Time (s) |
|---|---|---|---|---|---|
| ELM | 92.50 $\pm$ 2.00 | 96.00 $\pm$ 2.50 | 96.00 $\pm$ 0.50 | **0.0803** | **0.0135** |
| BP | 91.50 $\pm$ 3.00 | 94.00 $\pm$ 3.50 | 95.50 $\pm$ 0.50 | 1.6363 | 0.0256 |
| SVM | 95.00 $\pm$ 2.00 | 93.75 $\pm$ 0.25 | 95.25 $\pm$ 0.25 | 12.6410 | 3.6406 |

Table 7: Nonlinear features extracted from EEG signals (approximate entropy (ApEn), Hurst exponent and scaling exponent obtained with detrended fluctuation analysis (DFA)) are employed to characterize interictal and ictal EEGs. United features of ApEn, Hurst exponent and scaling exponent were used in this work.

Q. Yuan, et al., “Epileptic EEG classification based on extreme learning machine and nonlinear feature,” *Epilepsy Research*, vol. 96, pp. 29-38, 2011.

Y. Song, et al., “Automatic epileptic seizure detection in EEGs based on optimized sample entropy and extreme learning machine,” *Journal of Neuroscience Methods*, vol. 210, pp. 132-146, 2012.

# Real Operation of Wind Farms

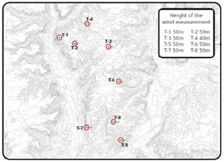



Figure 23: Situation of the wind measuring towers in Spain and within the wind farm.

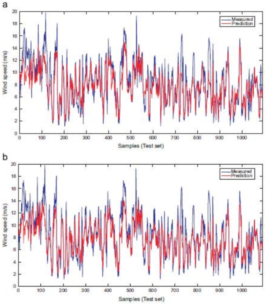



Figure 24: Wind speed prediction in tower 6 of the considered wind farm obtained by the ELM network (prediction using data from 7 towers). (a) Best prediction obtained and (b) worst prediction obtained.

B. Saavedra-Moreno, et al, "Very fast training neural-computation techniques for real measure-correlate-predict wind operations in wind farms," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 116, pp. 49-60, 2013.

# Electricity Price Forecasting

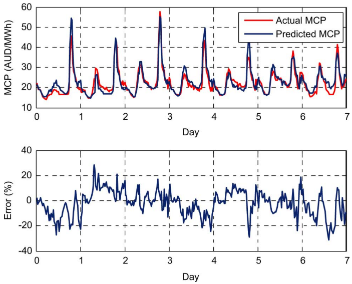



Figure 25: Average results of market clearing prices (MCP) forecast by ELM in winter: Trading in the Australian national electricity market NEM is based on a 30-min trading interval. Generators submit their offers every 5 min each day. Dispatch price is determined every 5 min and 6 dispatch prices are averaged every half-hour to determine the regional MCPs. In order to assist decision-making process for generators, there are totally 48 MCPs needed to be predicted at the same time for the coming trading day.

X. Chen, et al., "Electricity Price Forecasting With Extreme Learning Machine and Bootstrapping," *IEEE Transactions on Power Systems*, vol.27, no. 4, pp. 2055-2062, 2012.

# Remote Control of a Robotic Hand

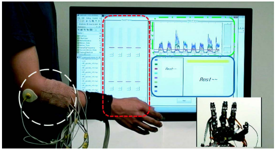



Figure 26: Control of a remote robot hand using sEMG signals on a forearm via TCP-IP communications.

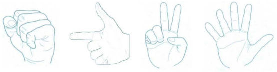


Figure 27: Cue signs of four motions.

- An eight wrist motions offline classification using linear support vector machines with little training time (under 10 minutes).
- This study shows human could control the remote side robot hand in real-time using his or her sEMG signals with less than 50 seconds recorded training data with ELM.

H. Lee, et al., "Online Remote Control of a Robotic Hand Configurations Using sEMG Signals on a Forearm," *the 2011 IEEE International Conference on Robotics and Biomimetics*, December 7-11, 2011.

# Remote Control of a Robotic Hand

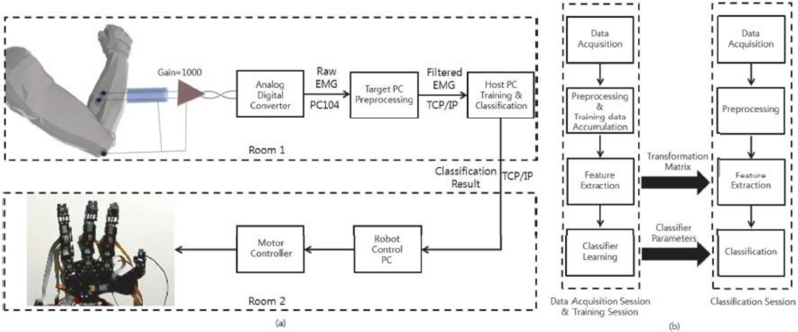



Figure 28: Algorithm and system block diagram

# ELM Based Intrusion Detection

Table 8: 22 Attack Types

| Denial of Service (DoS) | Unauthorized Access to Local Root Privileges | Unauthorized Access from a Remote Machine | Probing |
|---|---|---|---|
| Back<br>Neptune<br>Land<br>Teardrop<br>Ping of Death<br>Smurf | Perl<br>Buffer Overflow<br>Load Module<br>Rootkit | FTP Write<br>Guess Password<br>Imap<br>Multihop<br>Phf<br>Spy<br>Warezclient<br>Warezmaster | IP Sweep<br>Nmap<br>Port Sweep<br>Satan |

The dataset is from the 1998 DAPRA intrusion detection program. During the evaluation program, an environment was set up in Lincoln Labs to record 9 weeks of raw TCP/IP dump data for a network simulating a typical U.S. air force LAN. Then the LAN was operated under a real environment and blasted with multiple attacks. After that, 7 weeks of raw tcpdump data was processed into millions of connection records. Finally, 41 quantitative and qualitative features were extracted using data mining techniques.

C. Cheng, et al, "Intrusion detection using random features: an extreme learning machine approach," *Proceedings of International Joint Conference on Neural Networks (IJCNN2012)*, June 10 - June 15, 2012, Brisbane, Australia

# ELM Based Intrusion Detection

Table 9: Basic Features of Individual TCP Connections

| Feature Names | Description | Types |
|---|---|---|
| duration | length (number of seconds) of the connection | continuous |
| protocol_type | type of the protocol | discrete |
| service | network service on the destination | discrete |
| src_bytes | number of data bytes from source to destination | continuous |
| dst_bytes | number of data bytes from destination to source | continuous |
| flag | normal or error status of the connection | discrete |
| land | 1 if connection is from/to the same host/port, 0 otherwise | discrete |
| wrong_fragment | number of “wrong” fragments | continuous |
| urgent | number of urgent packets | continuous |

# ELM Based Intrusion Detection

Table 10: Content Features Within a Connection Suggested by Domain Knowledge

| Feature Names | Description | Types |
|---|---|---|
| hot | number of “hot” indicators | continuous |
| num_failed_logins | number of failed login attempts | continuous |
| logged_in | 1 if successfully logged in, 0 otherwise | discrete |
| num_compromised | number of “compromised” conditions | continuous |
| root_shell | 1 if root shell is obtained, 0 otherwise | discrete |
| su_attempted | 1 if “su root” command attempted, 0 otherwise | discrete |
| num_root | number of “root” accesses | continuous |
| num_file_creations | number of file creation operations | continuous |
| num_shells | number of shell prompts | continuous |
| num_access_files | number of operations on access control files | continuous |
| num_outbound_cmds | number of outbound commands in an ftp session | continuous |
| is_hot_login | 1 if the login belongs to the “hot” list, 0 otherwise | discrete |
| is_guest_login | 1 if the login is a “guest”login, 0 otherwise | discrete |

# ELM Based Intrusion Detection

Table 11: Traffic Features Computed Using a Two-Second Time Window

| Feature Names | Description | Types |
|---|---|---|
| count | number of connections to the same host as the current connection in the past two seconds | continuous |
| serror_rate | % of connections that have “SYN” errors | continuous |
| rerror_ rate | % of connections that have “REJ” errors | continuous |
| same_srv_rate | % of connections to the same service | continuous |
| diff_srv_rate | % of connections to different services | continuous |
| srv_count | number of connections to the same service as the current connection in the past two seconds | continuous |
| srv_serror_rate | % of connections that have “SYN” errors | continuous |
| srv_rerror_rate | % of connections that have “REJ” errors | continuous |
| srv_diff_host_rate | % of connections to different hosts | continuous |

# ELM Based Intrusion Detection

Table 12: Binary-Class Performance Comparison Results

| Dataset Size | SVM | | Basic ELM (Random Sigmoid Nodes) | | ELM (Gaussian Kernel) | |
|---|---|---|---|---|---|---|
| Training/Testing | Rate (%) | 95% Confidence Interval (%) | Rate (%) | 95% Confidence Interval (%) | Rate (%) | 95% Confidence Interval (%) |
| 1000/1000 | 99.15 | 99.12 - 99.17 | 99.33 | 99.15 - 99.51 | 99.12 | 99.05 - 99.25 |
| 2000/2000 | 99.43 | 99.40 - 99.45 | 99.07 | 98.90 - 99.24 | 99.27 | 99.25 - 98.28 |
| 4000/4000 | 99.77 | 99.76 - 98.78 | 99.58 | 99.50 - 99.66 | 99.63 | 99.61 - 99.65 |

# ELM Based Intrusion Detection

Table 13: Multi-Class Performance Comparison Results

| Dataset Size | SVM | | Basic ELM (Random Sigmoid Nodes) | | ELM (Gaussian Kernel) | |
|---|---|---|---|---|---|---|
| Training/Testing | Rate (%) | 95% Confidence Interval (%) | Rate (%) | 95% Confidence Interval (%) | Rate (%) | 95% Confidence Interval (%) |
| 1000/1000 | 97.58 | 97.52 - 97.64 | 96.83 | 96.40 - 97.23 | 97.78 | 97.72 - 97.83 |
| 2000/2000 | 98.31 | 98.27 - 98.34 | 97.07 | 96.77 - 97.37 | 98.81 | 98.76 - 98.86 |
| 4000/4000 | 98.69 | 98.66 - 98.72 | 97.00 | 96.68 - 97.32 | 98.74 | 98.70 - 98.78 |

# 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 14: Basic ELM: 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.)

G.-B. Huang, et al., “Extreme learning machine: theory and applications,” *Neurocomputing*, vol. 70, pp. 489-501, 2006.

# 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 15: Basic ELM: Performance comparison for learning function: SinC (5000 noisy training data and 5000 noise-free testing data) .

# 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 $\|\beta\|$

# 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)

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

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)**

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

where

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

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

# Scalability: ELM vs LS-SVM

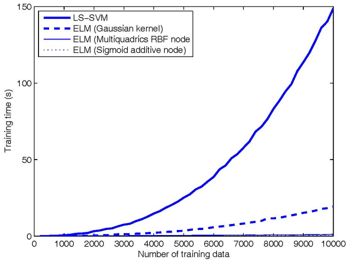



Figure 29: Scalability of different classifiers: Letter dataset.

G.-B. Huang, et al., "Extreme learning machine for regression and multiclass classification", *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol. 42, no. 2, pp. 513-529, 2012.

# Optimization Constraints of ELM and LS-SVM

**ELM: Based on Equality Constraint Conditions**

ELM optimization formula:

$$\text{Minimize: } L_{P_{ELM}} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 \qquad (11)$$

$$\text{Subject to: } \mathbf{h}(\mathbf{x}_i)\beta = \mathbf{t}_i^T - \xi_i^T, \forall i$$

The corresponding dual optimization problem:

$$\text{minimize: } L_{D_{ELM}} = \frac{1}{2}\|\beta\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\xi_i\|^2 - \sum_{i=1}^{N}\sum_{j=1}^{m}\left(\mathbf{h}(\mathbf{x}_i)\beta - \mathbf{t}_i^T + \xi_i^T\right)\alpha_i$$

$$\text{subject to: } \beta = \mathbf{H}^T\alpha, \alpha_i = C\xi_i, \mathbf{h}(\mathbf{x}_i)\beta - \mathbf{t}_i^T + \xi_i^T = 0, \forall i$$

G.-B. Huang, et al., "Extreme learning machine for regression and multiclass classification", *IEEE Transactions on Systems, Man and Cybernetics - Part B*, vol. 42, no. 2, pp. 513-529, 2012.

# Optimization Constraints of ELM and LS-SVM

**LS-SVM: Based on Equality Constraint Conditions**

LS-SVM optimization formula:

$$\text{minimize: } L_{P_{LS-SVM}} = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 \qquad (12)$$

$$\text{subject to: } t_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b) = 1-\xi_i, \forall i$$

The corresponding dual optimization problem:

$$\text{minimize: } L_{D_{LS-SVM}} = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2 - \sum_{i=1}^{N}\alpha_i\left(t_i\left(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b\right)-1+\xi_i\right)$$

$$\text{subject to: } \mathbf{w} = \sum_{i=1}^{N}\alpha_i t_i\phi(\mathbf{x}_i), \alpha_i = C\xi_i, t_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b)-1+\xi_i = 0, \forall i$$

$$\sum_{i=1}^{N}\alpha_i t_i = 0$$

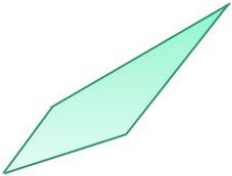


Figure 30: In LS-SVM optimal $\alpha_i$ are found from one hyperplane $\sum_{i=1}^{N}\alpha_i t_i = 0$.

# Optimization Constraints of ELM and SVM

**ELM variant: Based on Inequality Constraint Conditions**

ELM optimization formula:

$$\text{Minimize: } L_P = \frac{1}{2}\|\beta\|^2 + C\sum_{i=1}^{N}\xi_i$$
$$\text{Subject to: } t_i\beta \cdot \mathbf{h}(\mathbf{x}_i) \geq 1 - \xi_i, \forall i$$
$$\xi_i \geq 0, \forall i \quad (13)$$

The corresponding dual optimization problem:

$$\text{minimize: } L_D = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j \mathbf{h}(\mathbf{x}_i)\cdot\mathbf{h}(\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i \quad (14)$$

$$\text{subject to: } 0 \leq \alpha_i \leq C, \forall i$$

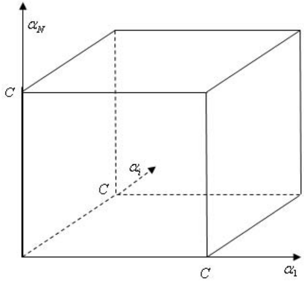



Figure 31: ELM

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

# Optimization Constraints of ELM and SVM

## SVM Constraint Conditions

SVM optimization formula:

$$\begin{aligned}&\text{Minimize: } L_P = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i\\ &\text{Subject to: } t_i(\mathbf{w}\cdot\phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \forall i\\ &\xi_i \geq 0, \quad i=1,\cdots,N\end{aligned} \tag{15}$$

The corresponding dual optimization problem:

$$\begin{aligned}&\text{minimize: } L_D = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} t_i t_j \alpha_i \alpha_j \phi(\mathbf{x}_i)\cdot\phi(\mathbf{x}_j) - \sum_{i=1}^{N}\alpha_i\\ &\text{subject to: } 0 \leq \alpha_i \leq C, \forall i\\ &\sum_{i=1}^{N} t_i\alpha_i = 0\end{aligned} \tag{16}$$

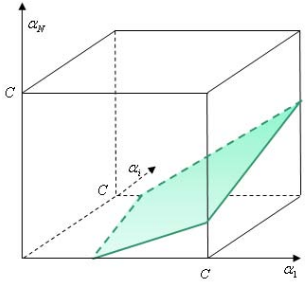



Figure 32: SVM

# Optimization Constraints of ELM and SVM

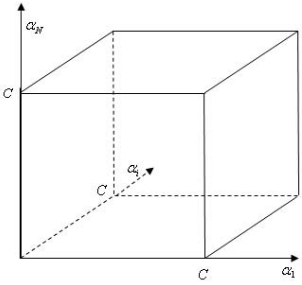



Figure 33: ELM

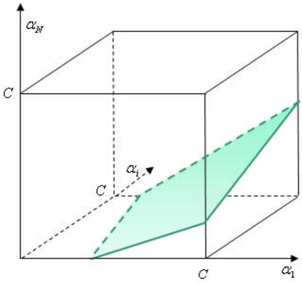



Figure 34: SVM

ELM and SVM have the same dual optimization objective functions, but in ELM optimal $\alpha_i$ are found from the entire cube $[0, C]^N$ while in SVM optimal $\alpha_i$ 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. SVMs may apply same application-oriented constraints to irrelevant applications. 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.

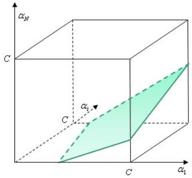



Figure 35: SVM

## 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.
3. LS-SVM inherits such "generosity" from the conventional 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. SVMs may apply same application-oriented constraints to irrelevant applications. 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.

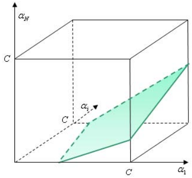



Figure 35: SVM

## 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.
3. LS-SVM inherits such "generosity" from the conventional SVM.

# 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.

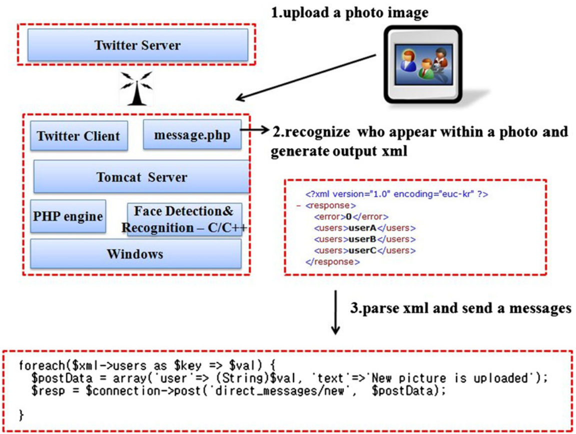
# OS-ELM Algorithm

## 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.

# Intelligent Photo Notification System For Twitter Service




K. Choi, et al., "Incremental face recognition for large-scale social network services", *Pattern Recognition*, vol. 45, pp. 2868-2883, 2012.

# Intelligent Photo Notification System For Twitter Service

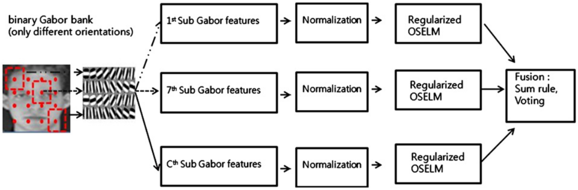



Figure 36: Binary Gabor filter-based OS-ELM (BG-OSELM)

| Methods | Baseline | | Sequential Subspace | | | Sequential Classifiers | | |
|---|---|---|---|---|---|---|---|---|
| Database | PCA | FDA | CCIPCA | IPCA | ILDA | OSELM | BG-OSELM(S) | BG-OSELM(V) |
| AR | 77.0 | 72.3 | 55.0 | 77.3 | 76.6 | 80.3 | 92.0 | 87.6 |
| EYALE | 99.7 | 96.9 | 58.5 | 99.7 | 100.0 | 100.0 | 99.7 | 99.7 |
| BIOID | 98.1 | 97.3 | 91.6 | 97.5 | - | 98.5 | 97.4 | 96.7 |
| ETRI | 95.8 | 95.5 | 86.9 | 95.4 | - | 97.2 | 97.0 | 94.2 |

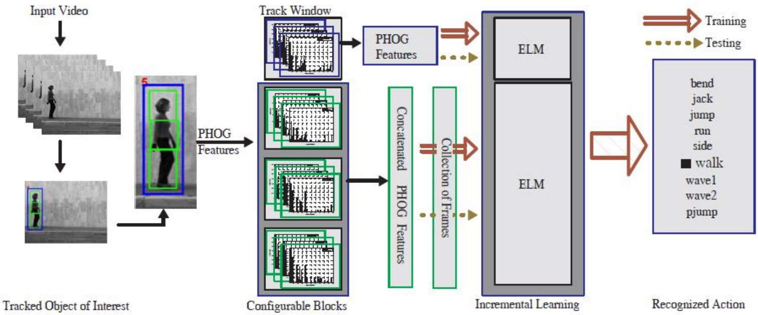
Table 16: Performance comparison of different sequential methods.

## Online Sequential Human Action Recognition

Figure 37: Example frames from top row: Weizmann dataset, middle row: KTH dataset, and bottom row: UCF sports dataset

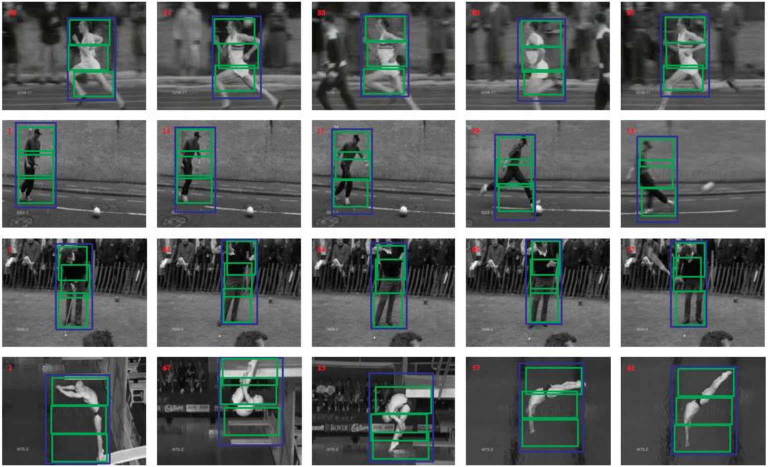
R. Minhas, et al., "Incremental learning in human action recognition based on *Snippets*", *(in press) IEEE Transactions on Circuits and Systems for Video Technology*, 2012.

# Online Sequential Human Action Recognition




R. Minhas, et al., "Incremental learning in human action recognition based on *Snippets*", *(in press) IEEE Transactions on Circuits and Systems for Video Technology*, 2012.

## Online Sequential Human Action Recognition



| Weizmann dataset | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Methods | OS-ELM Based | | | | [32] | [14] | [36] | | | [11] | |
| Frames | 1/1 | 3/3 | 6/6 | 10/10 | 1/12 | 1/9 | 1/1 | 7/7 | 10/10 | 8/8 | 20/20 |
| Accuracy | 65.2 | 95.0 | 99.63 | 99.9 | 55.0 | 93.8 | 93.5 | 96.6 | 99.6 | 97.05 | 98.68 |
| **KTH dataset** | | | | | | | | | | | |
| Methods | OS-ELM Based | | | | [25] | [33] | [43] | [14] | [36] | | [12] |
| Frames | 1/1 | 3/3 | 6/6 | 10/10 | - | - | - | - | 1/1 | 7/7 | 20/20 |
| Accuracy | 74.4 | 88.5 | 92.5 | 94.4 | 91.3 | 90.3 | 83.9 | 91.7 | 88.0 | 90.9 | 90.84 |

Table 17: Classification comparison against different approaches at *snippet*-level.

| Weizmann dataset | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Methods | OS-ELM Based | | | | [2] | [32] | [14] | [36] | [41] | [30] | [11] |
| Frames | 1/1 | 3/3 | 6/6 | 10/10 | - | - | - | - | - | - | - |
| Accuracy | 100.0 | 100.0 | 100.0 | 100.0 | 100.0 | 72.8 | 98.8 | 100.0 | 97.8 | 99.44 | 100.0 |
| **KTH dataset** | | | | | | | | | | | |
| Methods | OS-ELM Based | | | | [14] | [36] | [30] | [21] | [27] | [9] | [44] |
| Frames | 1/1 | 3/3 | 6/6 | 10/10 | - | - | - | - | - | - | - |
| Accuracy | 92.8 | 93.5 | 95.7 | 96.1 | 91.7 | 92.7 | 94.83 | 95.77 | 97.0 | 96.7 | 95.7 |

Table 18: Classification comparison against different approaches at *sequence*-level.

# Summary

- Both G. Hinton and V. Vapnik have made great contributions in neural networks R&D
  - Without Hinton's work on BP in 1982, neural networks might not have revived in 1980's.
  - Without Vapnik's work on SVM in 1995, neural networks might have disappeared although many SVM researchers do not consider SVM a kind of solutions to the traditional neural networks. Without SVM, many applications in pattern recognition, HCI, BCI, computational intelligence and machine learning, etc, may not have appeared.
  - However, both BP and SVM over-emphasize some aspects of learning and overlook the other aspects, and thus, both become incomplete in theory:
    - i) BP gives preference on training but does not consider the stability of the system (consistency of minimum norm of weights in neural networks and matrix theory)
    - ii) SVM confines the research in the maximum margin concept which limits the research in binary classification and does not have direct and efficient solutions to regression and multi-class applications. The consistency between maximum margin, minimum norm of weights in neural networks and matrix theory has been overlooked.
- From learning point of view, ELM theory seems more complete.

# 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 random hidden nodes (random features) and kernels.

# Summary

- Generally speaking, efficient for regression and classification applications. Existing applications:
  - Biometrics
  - Bioinformatics
  - Image processing (image segmentation, image quality assessment, image super-resolution)
  - Signal processing
  - Human action recognition
  - Disease prediction and eHealthCare
  - Location positioning system
  - Brain computer interface
  - Human computer interface
  - Feature selection
  - Time-series
  - Real-time learning and prediction
  - Security and data privacy
  - Big data analytics
  - Internet of Things
- Potential Influence:
  - Machine learning (resulting in second wave of machine learning and artificial intelligence with the increasing demand in handling big data in different applications?)
  - Matrix theory and optimization theory
  - Functioning artificial “brain” (coming out in 10 years?)
  - Robot and automation
  - Data and knowledge discovery
  - Cognitive and reasoning system

# Open Problems

1. As observed in experimental studies, the performance of basic ELM is stable in a wide range of number of hidden nodes. Compared to the BP learning algorithm, the performance of basic ELM is not very sensitive to the number of hidden nodes. However, how to prove it in theory remains open.
2. One of the typical implementations of ELM is to use random nodes in the hidden layer and the hidden layer of SLFNs need not be tuned. It is interesting to see that the generalization performance of ELM turns out to be very stable. How to estimate the oscillation bound of the generalization performance of ELM remains open too.
3. It seems that ELM performs better than other conventional learning algorithms in applications with higher noise. How to prove it in theory is not clear.
4. ELM always has faster learning speed than LS-SVM if the same kernel is used?

# Open Problems

5. ELM provides a batch learning kernel solution which is much simpler than other kernel learning algorithms such as LS-SVM. It is known that it may not be straightforward to have an efficient online sequential implementation of SVM and LS-SVM. However, due to the simplicity of ELM, is it possible to implement the online sequential variant of the kernel based ELM?
6. ELM always provides similar or better generalization performance than SVM and LS-SVM if the same kernel is used (if not affected by computing devices' precision)?
7. ELM tends to achieve better performance than SVM and LS-SVM in multiclasses applications, the higher the number of classes is, the larger the difference of their generalization performance will be?
8. Scalability of ELM with kernels in super large applications.
9. Parallel and distributed computing of ELM.
10. ELM will make real-time reasoning feasible?