Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering
IDP Institut für Datenanalyse
und Prozessdesign

Extreme Learning Machines and Its applications in Fault Detection

Yang Hu

Research Associate

ZHAW Zurich University of Applied Sciences

Institute for Data Analysis and Process Design (IDP)

2015.04.20

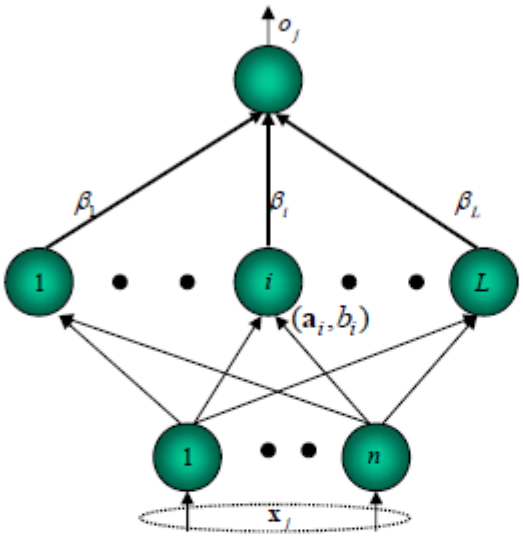
Outline

Zürcher Hochschule
für Angewandte Wissenschaften
zh aw School of Engineering
IDP Institut für Datenanalyse und Prozessdesign

- Extreme Learning Machines Theory
- Comparison with SVM
- Performance Analysis
- Further Discussions

Extreme Learning Machines Theory

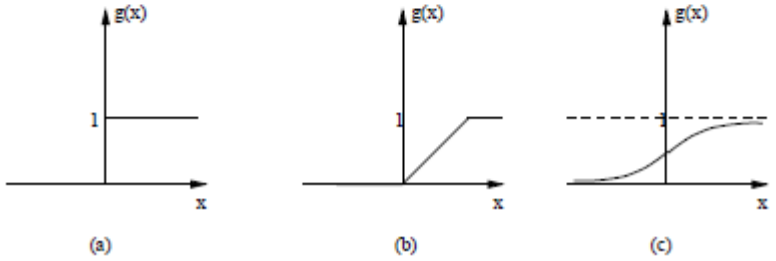
Single Layer Feedforward Neural Networks (SLFN)



Output Node

L Hidden Nodes

n Input Nodes



Output of additive hidden nodes:

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

Output of RBF hidden nodes:

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

The output function of SLFNs is:

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

$\boldsymbol{\beta}_i$: Output weight vector connecting the ith hidden node and the output nodes

Extreme Learning Machines Theory

Single Layer Feedforward Neural Networks (SLFN)

- Approximation capability [Leshno 1993, Park and Sandberg 1991]: 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 $\|f_L(\mathbf{x}) - f(\mathbf{x})\| < \varepsilon$.
- Classification capability [Huang, et al 2000]: As long as SLFNs can approximate any continuous target function $f(\mathbf{x})$, such SLFNs can differentiate any disjoint regions.

Extreme Learning Machines Theory

Single Layer Feedforward Neural Networks (SLFN)

- Many learning methods mainly based on gradient-descent / iterative approaches have been developed over the past three decades.
  - Back-Propagation (BP) [Rumelhart 1986] and its variants are most popular.
- Least-square (LS) solution for RBF network, with single impact factor for all hidden nodes. [Broomhead and Lowe 1988]
- QuickNet (White, 1988) and Random vector functional network (RVFL) [Igelnik and Pao 1995]
- Support vector machines and its variants. [Cortes and Vapnik 1995]
- Deep learning: dated back to 1960s and resurgence in mid of 2000s [wiki 2015]

Zürcher Hochschule
für Angewandte Wissenschaften
zh aw School of Engineering
IDP Institut für Datenanalyse
und Prozessdesign

Single Layer Feedforward Neural Networks (SLFN)

Learning Issues:

- Most effects are focus on how to adjust hidden nodes;
- Learning is somehow inefficiency

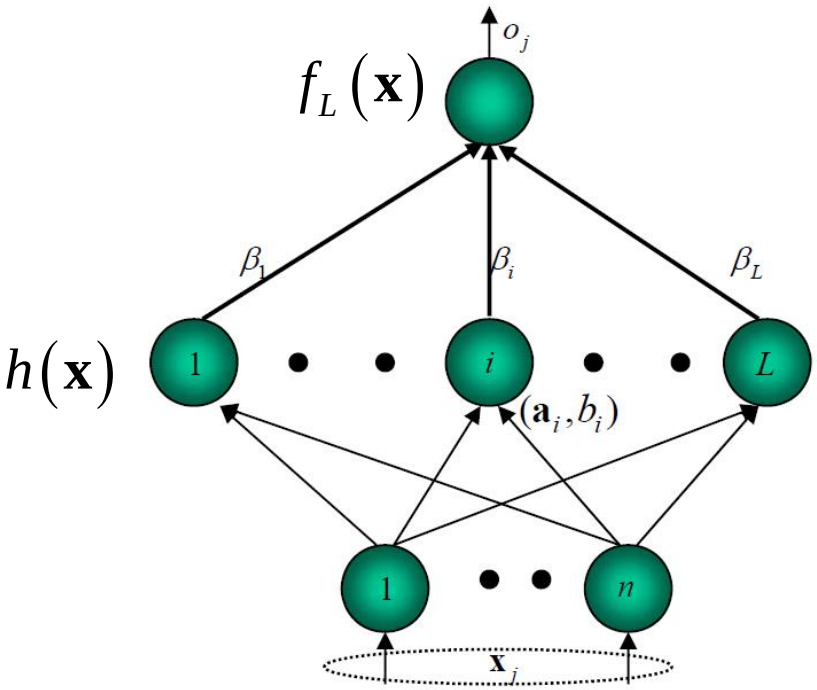
Question:

- Do we really need so many learning methods to iteratively train the SLFN?
- How can we make learning efficient?

Extreme Learning Machines Theory

Extreme Learning Machines (ELM)

-- "Generalized" SLFN



Output function of "generalized" SLFNs:

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

The hidden layer output function (hidden layer mapping, ELM feature space):

$$\boldsymbol{h}(x) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \cdots, G(\mathbf{a}_L, b_L, x)]$$

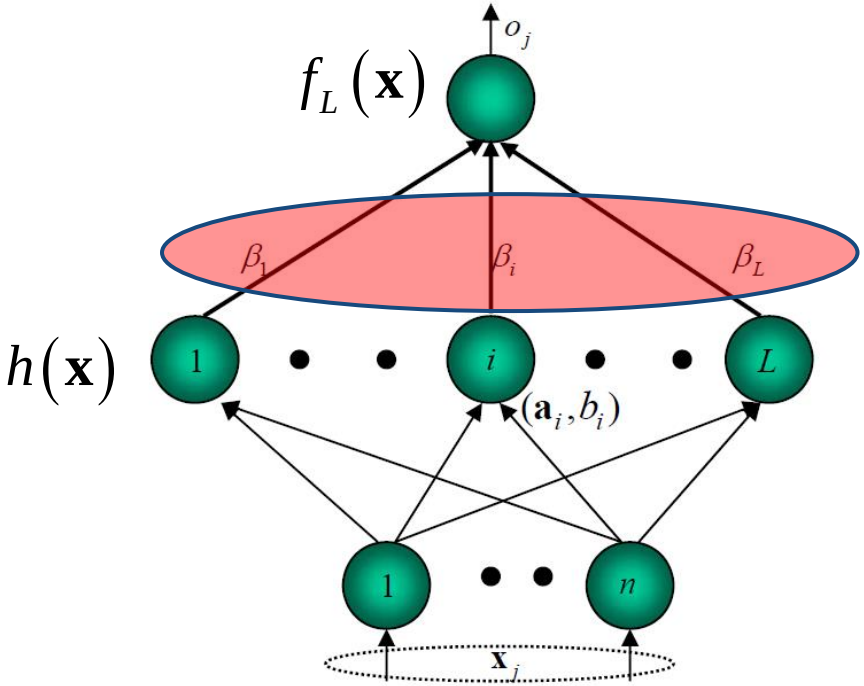
The output functions of hidden nodes can be but are not limited to:

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

RBF: $G(\mathbf{a}_i, b_i, \mathbf{x}) = g(b_i \|\mathbf{x} - \mathbf{a}_i\|)$

Fourier Series: $G(\mathbf{a}_i, b_i, \mathbf{x}) = \cos(\mathbf{a}_i \cdot \mathbf{x} + b_i)$

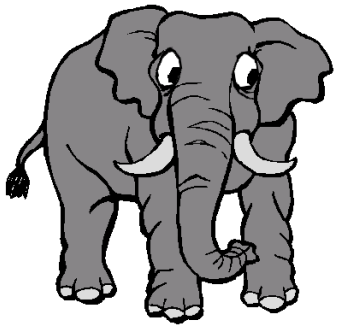
Extreme Learning Machines Theory



How to train ELM?

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, \mathbf{x}) = \mathbf{H}\beta$$

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} h_1(\mathbf{x}_1) & \cdots & h_L(\mathbf{x}_1) \\ \vdots & \vdots & \vdots \\ h_1(\mathbf{x}_N) & \vdots & h_L(\mathbf{x}_N) \end{bmatrix}$$



Extreme Learning Machines Theory

How to train ELM?

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, \mathbf{x}) = \mathbf{H}\beta$$

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

Equivalent ELM optimization formula:

$$\min_{\boldsymbol{\beta} \in \mathbf{R}^{L \times m}} \quad \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\sum_{i=1}^{N}\|\mathbf{e}_i\|^2$$

$$\text{s.t.} \quad \mathbf{h}(\mathbf{x}_i)\boldsymbol{\beta} = \mathbf{t}_i^T - \mathbf{e}_i^T, \quad i = 1, \ldots, N.$$

$$\Rightarrow \quad \min_{\boldsymbol{\beta} \in \mathbf{R}^{L \times m}} \quad L_{\text{ELM}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{C}{2}\|\mathbf{T} - \mathbf{H}\boldsymbol{\beta}\|^2$$

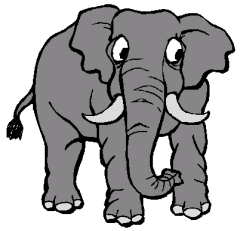
ridge regression or regularized least squares

$$\frac{\partial L_{ELM}}{\partial \beta} = \beta - C\mathbf{H}^T(\mathbf{T} - \mathbf{H}\beta) = 0 \quad \Rightarrow \quad \beta^* = \left(\mathbf{H}^T\mathbf{H} + \frac{\mathbf{I}}{C}\right)^{-1}\mathbf{H}^T\mathbf{T}$$

Extreme Learning Machines Theory

How to train ELM?

$$f_L(\mathbf{x}) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, \mathbf{x}) = \mathbf{H}\beta$$



Simple Math is Enough!

Given a training set $\{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^d, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, 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 $\mathbf{H} = \begin{bmatrix} \boldsymbol{h}(\mathbf{x}_1) \\ \vdots \end{bmatrix}$.

3) Calculate the output weights $\boldsymbol{\beta}$. $\boldsymbol{\beta}^* = \left(\mathbf{H}^T\mathbf{H} + \frac{\mathbf{I}}{C}\right)^{-1} \mathbf{H}^T\mathbf{T}$

Extreme Learning Machines Theory

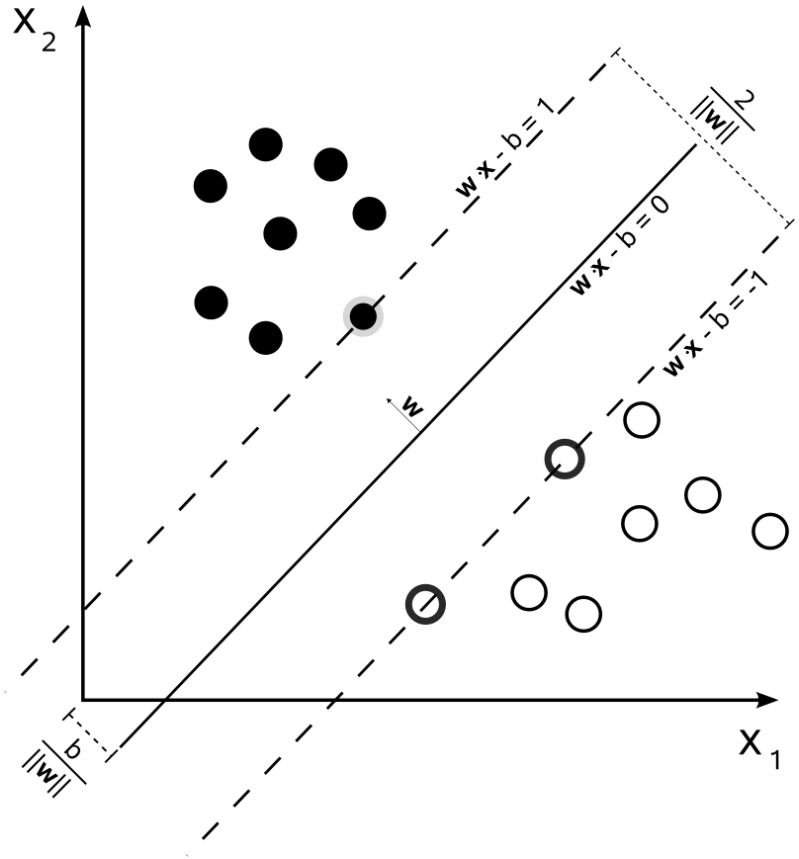
Extreme Learning Machines (ELM)

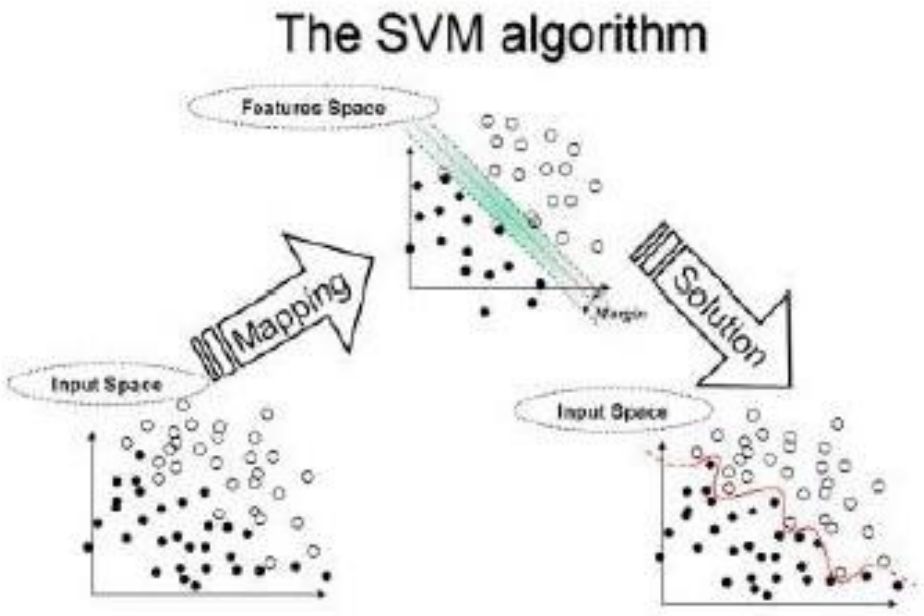
-- "Generalized" SLFN

- **Essence of ELM**
  - Hidden layer need not be tuned.
    - "randomness" is just one of ELM's implementation, but not all
    - Some conventional methods adopted "semi-randomness"
  - Hidden layer mapping $\boldsymbol{h}(\mathbf{x})$ satisfies universal approximation conditions.
  - Minimize: $\|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_p$ and $\|\boldsymbol{\beta}\|_q$
    - (norm p and q could have different values, $q = 1, \frac{1}{2}, 2, \cdots$)
  - It satisfies both ridge regress theory [Hoerl and Kennard 1970] and neural network generalization theory [Bartlett 1998].

Comparison with SVM

Support Vector Machine

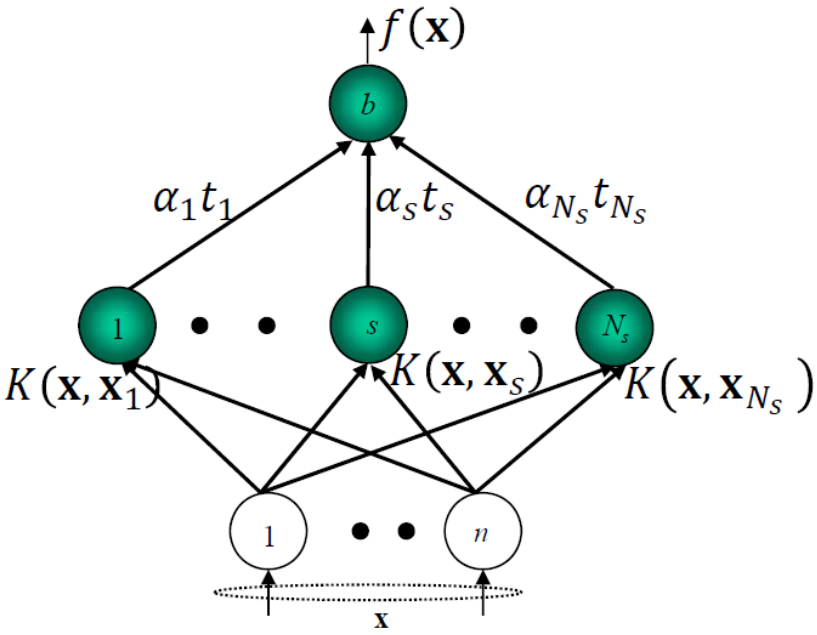




Comparison with SVM

Support Vector Machine

– an Alternative Solution of SLFN



SVM optimization formula

$$\text{minimize: } L_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$
$$\text{subject to: } t_i(\mathbf{w}\cdot\phi(\mathbf{x}) + b) \geq 1 - \xi_i, \forall i$$
$$\xi_i \geq 0, \forall i$$

LS-SVM optimization formula

$$\text{minimize: } L_p = \frac{1}{2}\|\mathbf{w}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2$$
$$\text{subject to: } t_i(\mathbf{w}\cdot\phi(\mathbf{x}) + b) = 1 - \xi_i, \forall i$$

Typical kernel function:

$$K(\mathbf{x}, \mathbf{y}) = \exp(-\gamma\|\mathbf{x} - \mathbf{y}\|^2)$$

The decision function of SVM and LS-SVM is:

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

Comparison with SVM

Support Vector Machine

– an Alternative Solution of SLFN

- **LS-SVM: Based on Equality Constraint Conditions** [Suykens and Vandewalle 1999]
  - LS-SVM optimization formula:

    Minimize: $L_{P_{LS-SVM}} = \frac{1}{2}\|\mathbf{w}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\xi_i^2$

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

  - The corresponding dual optimization problem:

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

    subject to:

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

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

In LS-SVM optimal α_i are found from one hyper plane $\sum_{i=1}^{N}\alpha_i t_i = 0$

Comparison with SVM

Support Vector Machine

– an Alternative Solution of SLFN

- **ELM: Based on Equality Constraint Conditions** [Huang, et al 2012]
  - ELM optimization formula:

    Minimize: $L_{P_{ELM}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2$

    subject to: $\boldsymbol{h}(\mathbf{x}_i)\boldsymbol{\beta} = \boldsymbol{t}_i^T + \boldsymbol{\xi}_i^T, \forall i$
  - The corresponding dual optimization problem:

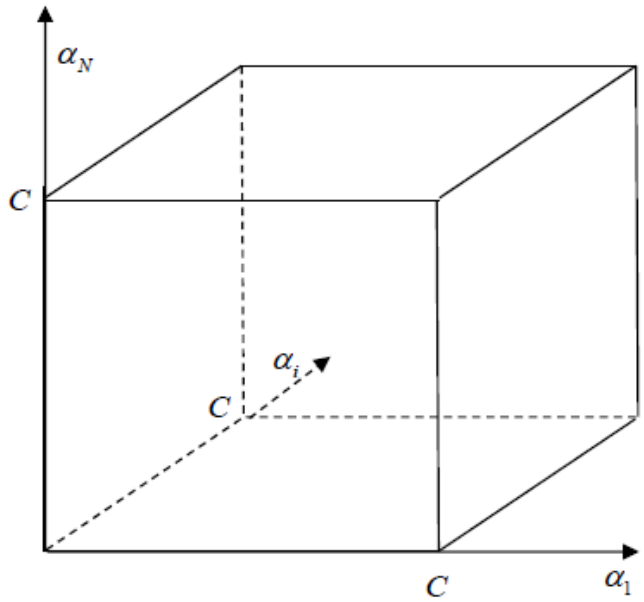
    Minimize: $L_{D_{ELM}} = \frac{1}{2}\|\boldsymbol{\beta}\|^2 + C\frac{1}{2}\sum_{i=1}^{N}\|\boldsymbol{\xi}_i\|^2 - \sum_{i=1}^{N} \left(\boldsymbol{h}(\mathbf{x}_i)\boldsymbol{\beta} - \boldsymbol{t}_i^T + \boldsymbol{\xi}_i^T\right)\boldsymbol{\alpha}_i$

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

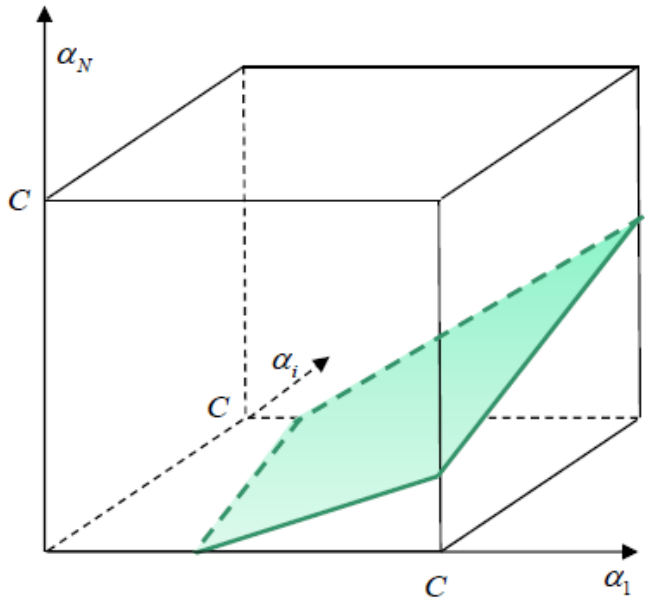
Comparison with SVM

Support Vector Machine

– an Alternative Solution of SLFN



ELM's inequality constraint variant [Huang, et al 2010]



SVM

ELM (based on inequality constraint conditions) 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 $\sum_{i=1}^{N} \alpha_i t_i = 0$ within the cube $[0, C]^N$. SVM always provides a suboptimal solution, so does LS-SVM.

Performance Analysis

TABLE II
SPECIFICATION OF BINARY CLASSIFICATION PROBLEMS

Datasets	# train	# test	# features	Random Perm
Diabetes	512	256	8	Yes
Australian Credit	460	230	6	Yes
Liver	230	115	6	Yes
Banana	400	4900	2	No
Colon	30	32	2000	No
Colon (Gene Sel)	30	32	60	No
Leukemia	38	34	7129	No
Leukemia (Gene Sel)	38	34	60	No
Brightdata	1000	1462	14	Yes
Dimdata	1000	3192	14	Yes
Mushroom	1500	6624	22	Yes
Adult	4781	27780	123	No

TABLE III
SPECIFICATION OF MULTICLASS CLASSIFICATION PROBLEMS

Datasets	# train	# test	# features	# classes	Random Perm
Iris	100	50	4	3	Yes
Glass	142	72	9	6	Yes
Wine	118	60	13	3	Yes
Ecoli	224	112	7	8	Yes
Vowel	528	462	10	11	No
Vehicle	564	282	18	4	Yes
Segment	1540	770	19	7	Yes
Satimage	4435	2000	36	6	No
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 IV
SPECIFICATION OF REGRESSION PROBLEMS

Datasets	# train	# test	# features	Random Perm
Baskball	64	32	4	Yes
Cloud	72	36	9	Yes
Autoprice	106	53	9	Yes
Strike	416	209	6	Yes
Pyrim	49	25	27	Yes
Bodyfat	168	84	14	Yes
Cleveland	202	101	13	Yes
Housing	337	169	13	Yes
Balloon	1334	667	2	Yes
Quake	1452	726	3	Yes
Space-ga	2071	1036	6	Yes
Abalone	2784	1393	8	Yes

Performance Analysis

TABLE VI
PERFORMANCE COMPARISON OF SVM, LS-SVM, AND ELM: BINARY CLASS DATA SETS

Datasets	SVM			LSSVM			Extreme Learning Machine								
							Gaussian Kernel			Sigmoid Additive Node			Multiquadrics RBF Node		
	Testing		Training	Testing		Training	Testing		Training	Testing		Training	Testing		Training
	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)
							$f(\mathbf{x}) = \text{sign}\left(\mathbf{h}(\mathbf{x})\mathbf{H}^T\left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1}\mathbf{T}\right)$								
Diabete	76.97	2.70	0.6759	77.18	2.07	0.1406	**77.52**	2.46	0.0528	77.95	2.18	0.2075	78.09	2.17	0.2306
Australian	85.79	2.03	0.7042	85.91	1.85	0.1250	**86.29**	1.43	0.0403	86.18	1.80	0.1709	86.70	1.90	0.1691
Credit															
Liver	72.65	3.61	0.5616	71.98	3.37	0.0625	72.14	3.74	0.0066	73.01	3.78	0.0528	71.57	0.04	0.0531
Banana	89.84	0	0.8120	89.63	0	0.0620	89.83	0	0.0469	89.61	0.05	0.1350	89.30	0.01	0.1416
Colon	84.38	0	0.1617	81.25	0	0.4531	**84.38**	0	0.0031	81.63	3.32	0.1103	82.13	1.55	0.1472
Colon (Gene Sel)	84.38	0	0.0462	87.50	0	0.0469	**90.63**	0	0.0010	89.62	2.85	0.0075	87.50	0	0.0072
Leukemia	82.34	0	1.007	**85.29**	0	1.703	82.35	0	0.0309	80.08	3.85	0.4288	83.47	3.41	0.5550
Leukemia (Gene Sel)	100	0	0.0494	100	0	0.0625	100	0	0.0003	100	0	0.0075	100	0	0.0106
							$f(\mathbf{x}) = \text{sign}\left(\mathbf{h}(\mathbf{x})\left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{T}\right)$								
Brightdata	99.46	0.23	1.289	**99.24**	0.17	0.5413	98.91	0.25	0.2984	99.31	0.2	0.7573	99.31	0.17	0.7401
Dimdata	95.85	0.27	0.8908	95.19	0.35	0.5781	**95.89**	0.34	0.2734	95.75	0.29	0.749	95.67	0.26	0.75
Mushroom	89.88	0.43	46.56	88.87	0.41	1.531	88.84	0.36	0.8133	88.91	0.36	1.038	88.88	0.33	1.047
* Adult	84.51	0	29.382	84.79	0	5.5703	84.58	0	3.3116	84.59	0.05	0.4246	84.51	0.02	0.5682

Performance Analysis

TABLE VII
PERFORMANCE COMPARISON OF SVM, LS-SVM, AND ELM: MULTICLASS DATA SETS

Datasets	SVM			LSSVM			Extreme Learning Machine								
							Gaussian Kernel			Sigmoid Additive Node			Multiquadrics RBF Node		
	Testing		Training	Testing		Training	Testing		Training	Testing		Training	Testing		Training
	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)	Rate (%)	Dev (%)	Time (s)
							$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{T}$								
Iris	95.12	2.45	0.075	96.28	2.36	0.0021	96.04	2.37	0.0022	97.6	2.29	0.0156	97.33	2.12	0.0161
Glass	67.83	4.67	0.2871	67.22	5.04	0.0097	**68.41**	4.81	0.0026	67.12	4.99	0.0262	66.89	4.97	0.0264
Wine	98.37	1.41	0.075	97.63	1.82	0.0043	**98.48**	1.7	0.0019	98.47	1.81	0.0222	98.57	1.26	0.0206
Ecoli	86.56	3.65	0.2469	85.93	2.82	0.0244	**87.48**	2.8	0.008	87.23	2.88	0.053	87.79	2.74	0.054
Vowel	56.28	0	2.172	52.81	0	0.3290	**58.66**	0	0.0688	53.73	0.91	0.2187	54.75	0.89	0.2231
Vehicle	84.37	1.71	1.5144	83.19	1.93	0.2029	83.16	1.89	0.0831	83.48	1.78	0.2557	83.95	1.85	0.2547
							$\mathbf{f}(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{T}$								
Segment	96.53	0.64	14.30	96.12	0.75	4.302	**96.53**	0.54	0.9272	96.07	0.69	1.889	95.54	0.41	1.070
Satimage	89.75	0	698.4	90.05	0	82.67	**92.35**	0	15.70	89.8	0.32	2.808	89.06	0.38	2.803
DNA	92.86	0	7732	93.68	0	6.359	**96.29**	0	2.156	93.81	0.24	1.586	94.81	0.32	1.597
* Letter	92.87	0.26	302.9	93.12	0.27	335.838	**97.41**	0.13	41.89	93.51	0.15	0.7881	93.96	0.15	1.4339
* **shuttle**	*99.74*	*0*	*2864.0*	*99.82*	*0*	**24767.0**	*99.91*	*0*	*4029.0*	*99.64*	*0.01*	**3.3379**	*99.65*	*0.02*	**5.5455**
* **USPS**	*96.14*	*0*	**12460**	*96.76*	*0*	*59.1357*	**98.9**	*0*	*9.2784*	*96.28*	*0.28*	**0.6877**	*97.25*	*0.24*	**0.9008**

Performance Analysis

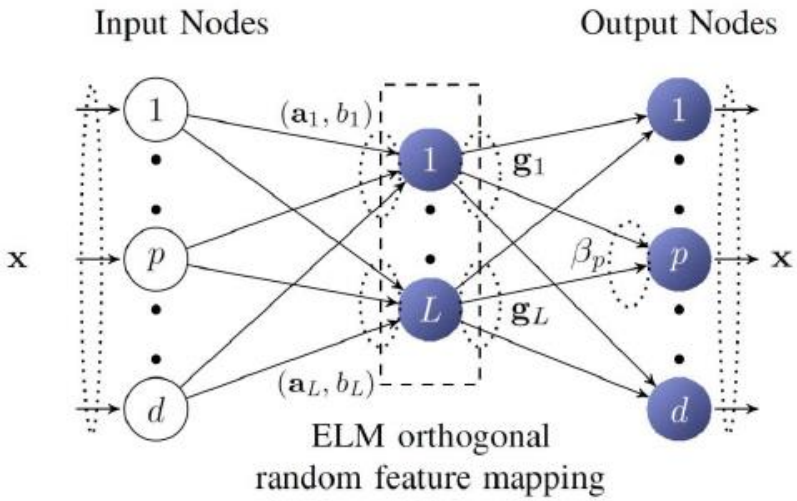
TABLE VIII
PERFORMANCE COMPARISON OF SVM, LS-SVM, AND ELM: REGRESSION DATA SETS

Datasets	SVR			LSSVR			Extreme Learning Machine								
							Gaussian Kernel			Sigmoid Additive Node			Multiquadrics RBF Node		
	Testing		Training	Testing		Training	Testing		Training	Testing		Training	Testing		Training
	RMSE	Dev	Time(s)	RMSE	Dev	Time(s)	RMSE	Dev	Time(s)	RMSE	Dev	Time(s)	RMSE	Dev	Time(s)
										$f(\mathbf{x}) = \mathbf{h}(\mathbf{x})\mathbf{H}^T \left(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T\right)^{-1} \mathbf{T}$					
Baskball	0.162	0.0138	0.0160	**0.1564**	0.0165	0.0010	0.1617	0.0175	0.0005	0.1629	0.0169	0.0066	0.1630	0.0155	0.0094
Cloud	0.3262	0.0668	0.0151	0.3049	0.0203	0.0008	0.3061	0.0239	0.0013	0.3165	0.0178	0.0063	0.2969	0.0243	0.0130
Autoprice	0.1776	0.0179	0.0620	**0.1601**	0.0217	0.0031	0.1697	0.015	0.0020	0.1710	0.0165	0.0188	0.1678	0.0189	0.0247
Strike	0.2282	0.0078	1.578	0.2479	0.0106	0.0531	**0.2322**	0.0128	0.0358	0.2985	0.0053	0.0791	0.2646	0.0186	0.1651
Pyrim	0.128	0.0315	0.0160	0.1272	0.0388	0.0006	**0.0921**	0.0167	0.0003	0.1194	0.0311	0.0059	0.1486	0.0369	0.0083
Bodyfat	0.0279	0.0081	0.0470	0.0280	0.0129	0.0059	0.0242	0.0131	0.0028	0.0286	0.0083	0.0231	0.0262	0.0113	0.0354
Cleveland	0.1646	0.0067	0.0930	0.1605	0.0071	0.0075	0.1618	0.0088	0.0053	0.1603	0.0070	0.0284	0.1609	0.0093	0.0474
Housing	0.0976	0.0085	0.2040	0.0704	0.0068	0.0343	0.0744	0.0106	0.0197	0.0805	0.0077	0.0616	0.0779	0.0084	0.1005
										$f(\mathbf{x}) = \mathbf{h}(\mathbf{x}) \left(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H}\right)^{-1} \mathbf{H}^T\mathbf{T}$					
Balloon	0.059	0.0034	11.30	0.0536	0.0044	1.072	0.0516	0.0043	0.6084	0.0553	0.0013	0.7688	0.0588	0.0121	1.344
Quake	0.1797	0.0068	19.59	**0.1446**	0.0079	1.387	0.1712	0.0099	0.6972	0.1649	0.0061	0.7625	0.1696	0.0098	1.607
Space-ga	0.0648	0.0016	52.75	0.0330	0.0008	3.510	0.0338	0.0018	1.770	0.0624	0.0021	1.220	0.0335	0.0013	1.369
Abalone	0.0764	0.0015	113.1	0.0746	0.0021	7.674	0.0768	0.0021	4.112	0.0761	0.0019	1.324	0.0761	0.0023	1.793

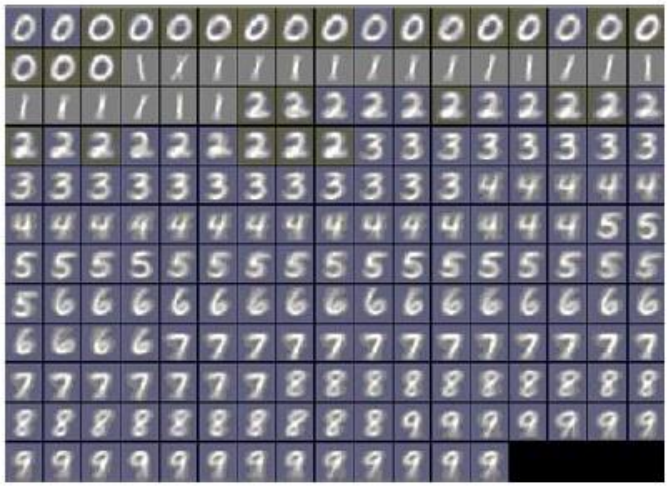
ELM can always achieve comparable performance as SVM and LS-SVM with much faster learning speed, especially for the large dataset

Further Discussion

ELM-auto encoder for feature learning



$d > L$: Compressed Representation
$d = L$: Equal Dimension Representation
$d < L$: Sparse Representation

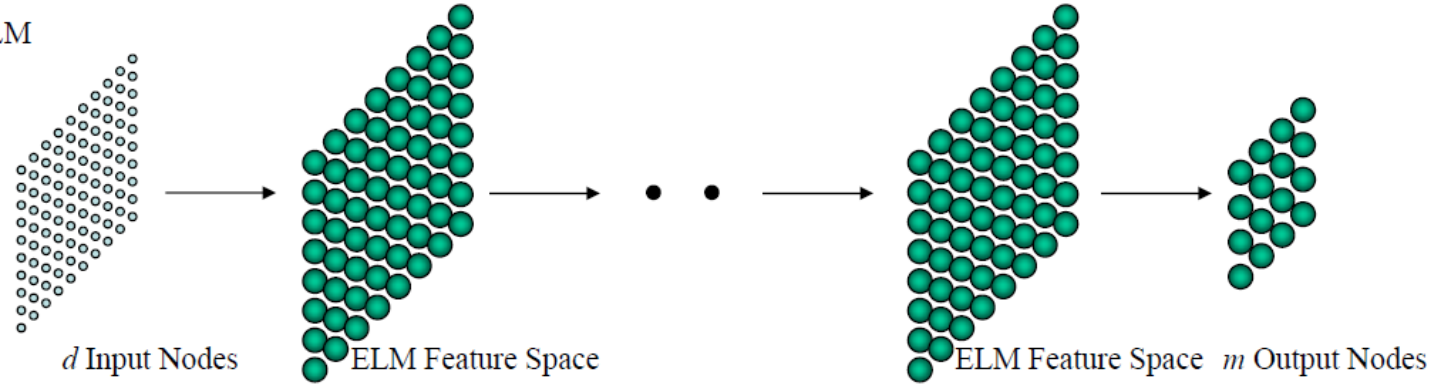


Features represented by the output weights of ELM-AE of MNIST OCR Datasets (with 60000 training samples and 10000 testing samples)

Further Discussion

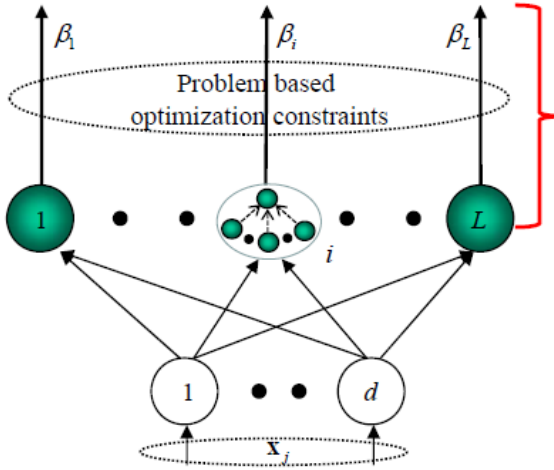
Deep ELM structure

(a) ELM



Different from Deep Learning, All the hidden neurons in ELM as a whole are not required to be iteratively tuned

(b) ELM subnetwork



Feature learning
Clustering
Regression
Classification

Hidden nodes need not be tuned. A hidden node can be a subnetwork of several nodes.

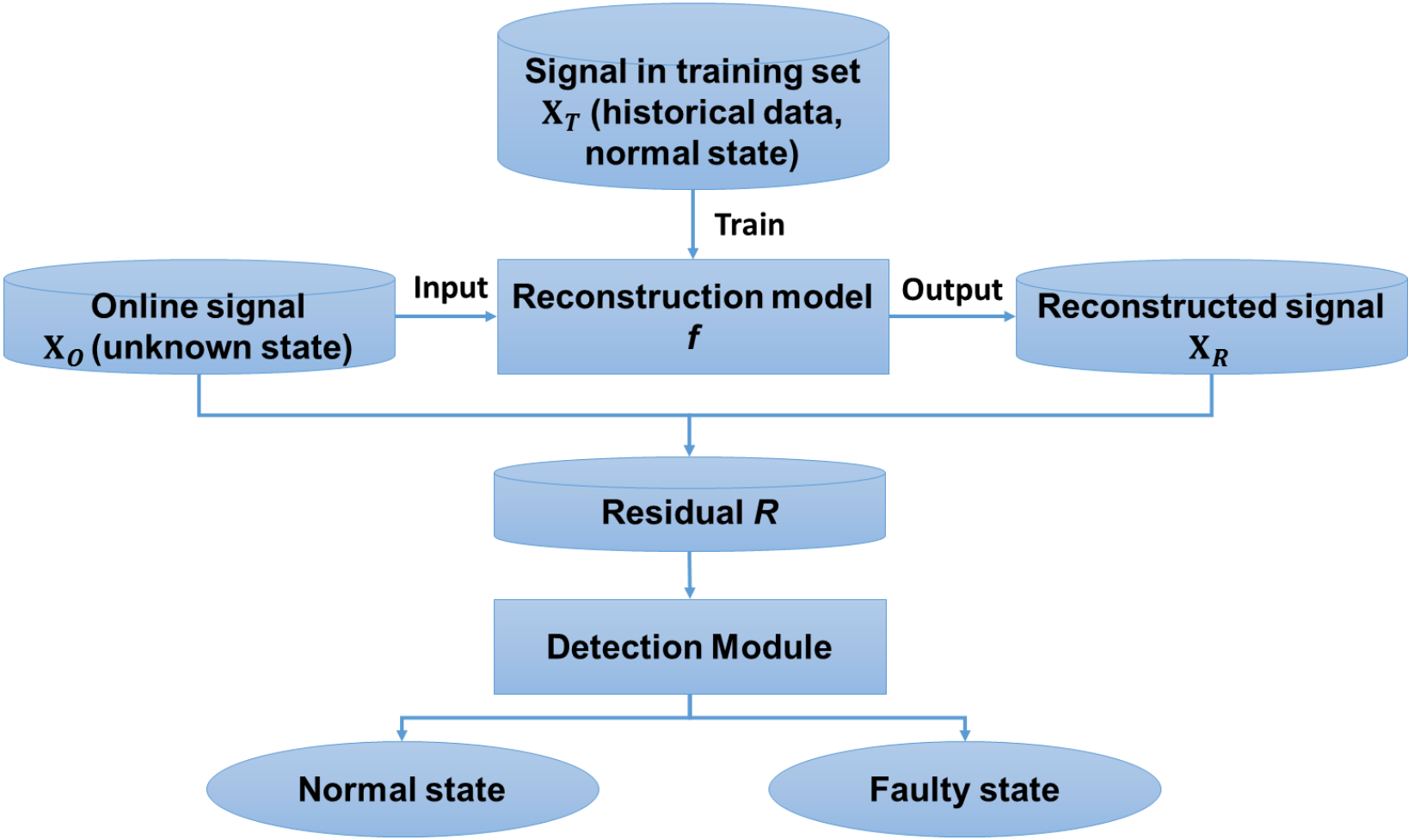
Further Discussion

ELM vs Deep Learning on MINST dataset

Learning Methods	Testing Accuracy	Training Time
H-ELM [Chenwei Deng, et al, 2015]	**99.14**	**281.37s**
Multi-Layer ELM (784-700-700-15000-10) [Huang, et al 2013]	**99.03±0.04**	**444.7s**
Deep Belief Networks (DBN) (748-500-500-2000-10)	98.87	20580s **(5.7 hours)**
Deep Boltzmann Machines (DBM) (784-500-1000-10)	99.05	68246s **(19 hours)**
Stacked Auto Encoders (SAE)	98.6	**> 17 hours**
Stacked Denoising Auto Encoders (SDAE)	98.72	**> 17 hours**
[Huang, et al 2013]		

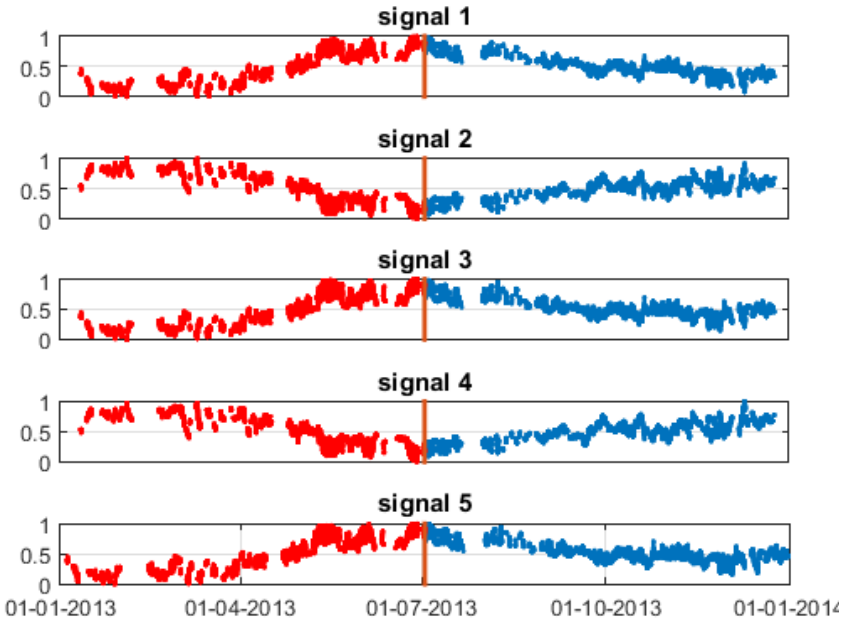
ELM in fault detection

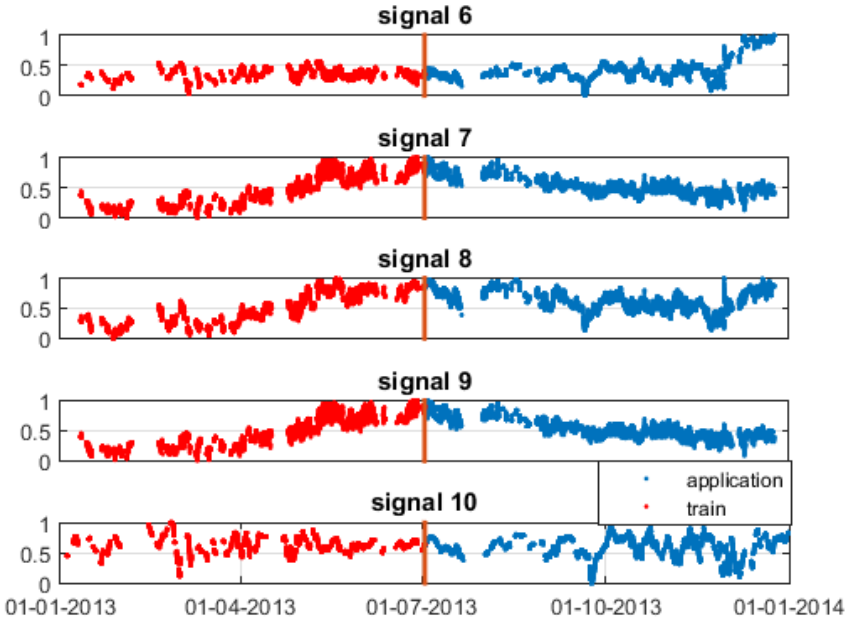
Signal reconstruction method



ELM in fault detection

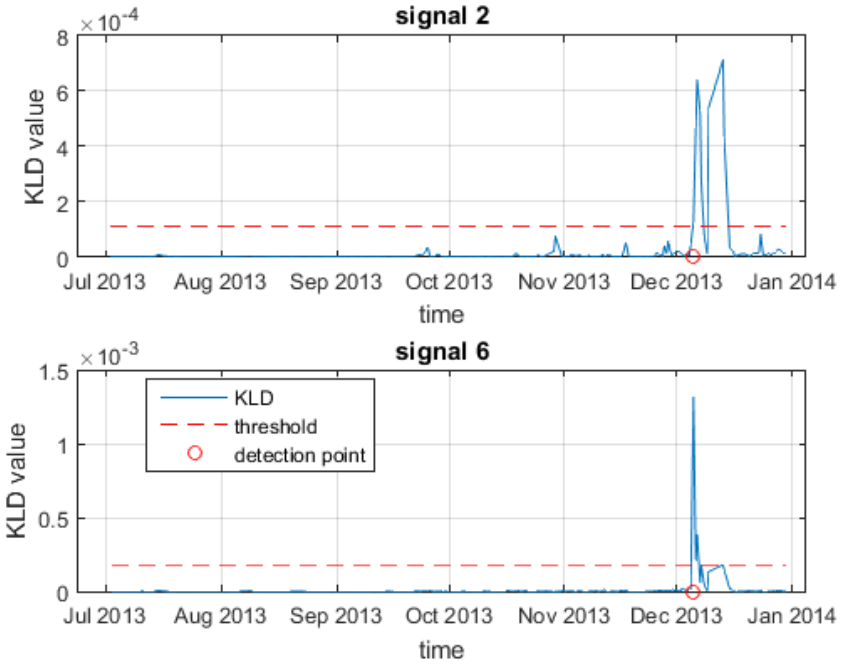
Signal reconstruction method



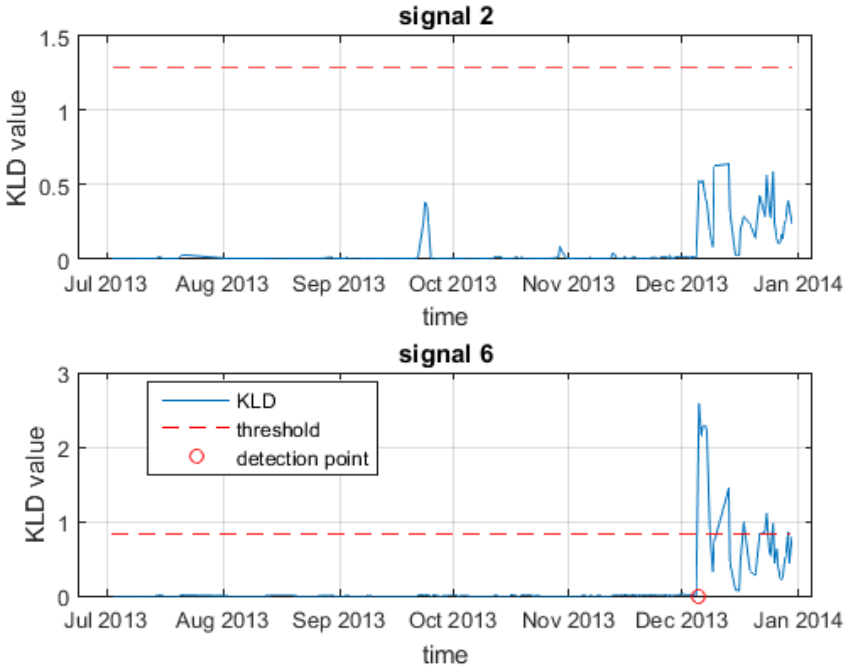


ELM in fault detection

Residual plot using ELM and AAKR

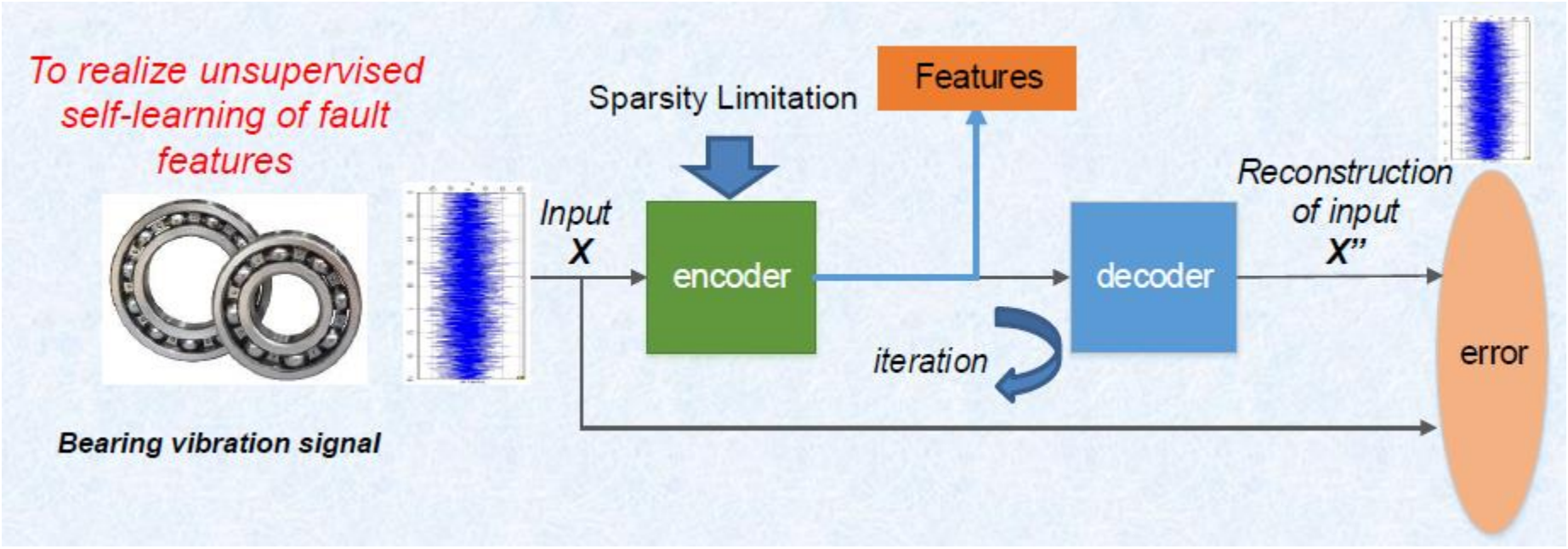


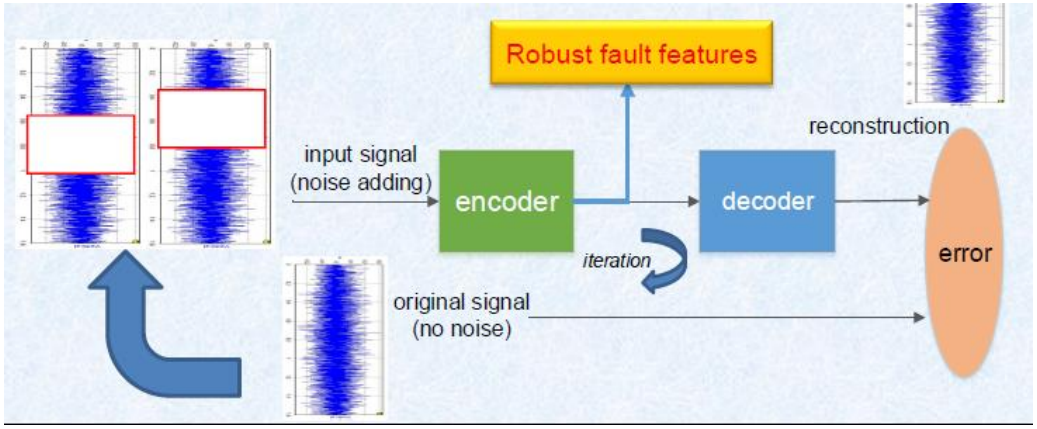
ELM



AAKR

ELM in fault detection





Summary

Extreme Learning Machines (ELM)

-- "Generalized" SLFN

- **Salient Features**
  - "Simple Math is Enough." ELM is a simple tuning-free three-step algorithm.
  - The learning speed of ELM is extremely fast.
  - Unlike conventional existence theories, *the hidden node parameters are not only independent of the training data but also of each other.* Although hidden nodes are important and critical, they need not be tuned.
  - 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.*

Summary

Extreme Learning Machines (ELM)

-- “Generalized” SLFN

Guang-Bin Huang
Associate Professor
Nanyang Technological University, Singapore
Email: EGBHUANG@NTU.EDU.SG
Phone: (+65)6790 4489
Office: S2.1-B2-06

http://www.ntu.edu.sg/home/egbhuang/
All materials in this PPT related with ELM are coming from this website.