Multi Layer Multi Objective Extreme Learning Machine

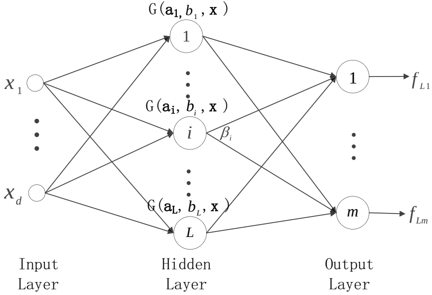
CUI DONGSHUN

NTU, Singapore

19/09/2017

Contents

Extreme Learning Machines (ELMs)



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

- $(\mathbf{a}_i, b_i) \in R^d \times R \ (i = 1, 2, \cdots, L)$: the hidden node parameters which are selected randomly and not tuned
- $\beta_i \in R^m$: the weight vector connecting between the ith hidden node and the output nodes
- $G(\mathbf{a}_i, b_i, \mathbf{x})$: output of the ith hidden node

Extreme Learning Machines (ELMs)

Given the training examples $\{(\mathbf{x}_j, \mathbf{t}_j)\}_{j=1}^{N} \subset R^d \times R^m$.Then we have:

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

or equivalently in matrix form

$$\mathbf{H}\beta = \mathbf{T} \tag{2}$$

where

$$\mathbf{H} = \begin{pmatrix} G(\mathbf{a}_1, b_1, \mathbf{x}_1) & \ldots G(\mathbf{a}_L, b_L, \mathbf{x}_1) \\ G(\mathbf{a}_1, b_1, \mathbf{x}_2) & \ldots G(\mathbf{a}_L, b_L, \mathbf{x}_2) \\ \vdots & \vdots \\ G(\mathbf{a}_1, b_1, \mathbf{x}_N) & \ldots G(\mathbf{a}_L, b_L, \mathbf{x}_N) \end{pmatrix}_{N \times L,} \quad \beta = \begin{pmatrix} \beta_1^T \\ \beta_2^T \\ \vdots \\ \beta_L^T \end{pmatrix}_{L \times m,} \quad \mathbf{T} = \begin{pmatrix} \mathbf{t}_1^T \\ \mathbf{t}_2^T \\ \vdots \\ \mathbf{t}_N^T \end{pmatrix}$$

Extreme Learning Machines (ELMs)

Three-Step Learning Mode

Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^d, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \cdots, N\}$, 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 $\mathbf{H}$.
3. Calculate the output weight $\boldsymbol{\beta}$: $\boldsymbol{\beta} = \mathbf{H}^\dagger \mathbf{T}$.

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

Multi Layer Extreme Learning Machine (ML-ELM)

- Original ELM is a single layer feed-forward neural network
- ML-ELM extends ELM to multi-layer neural networks
- ML-ELM uses ELM-AE to learn the hidden layer parameters
- Experimental results have shown ML-ELM performs better than ELM in computer vision tasks such as classification, object tracking and action recognition

Extreme Learning Machine Auto-Encoder (ELM-AE)

ELM-AE learns features of input data in three different architectures

- Compressed representation: in this representation ELM-AE has less number of hidden neurons than input neurons and performs dimension reduction
- Equal dimension representation: in this representation ELM-AE has the same number of hidden neurons as the input neurons
- Sparse representation: in this representation ELM-AE has larger number of hidden neurons than input neurons

Extreme Learning Machine Auto-Encoder (ELM-AE)

- Compressed $(d > L)$ and equal dimension $(d = L)$ representation representation the ELM feature mapping is calculated as: $\mathbf{h}(\mathbf{x}_j) = g(\mathbf{x}_j\mathbf{A} + \mathbf{b})$ where hidden layer parameters are orthogonal random $\mathbf{A}^T\mathbf{A} = \mathbf{I}$ and $\mathbf{b}^T\mathbf{b} = 1$.
- Sparse $(d < L)$ representation ELM feature napping is calculated as: $\mathbf{h}(\mathbf{x}_j) = g(\mathbf{x}_j\mathbf{A} + \mathbf{b})$ where hidden layer parameters are orthogonal random $\mathbf{A}\mathbf{A}^T = \mathbf{I}$ and $\mathbf{b}\mathbf{b}^T = 1$.

ELM-AE Learning Problems

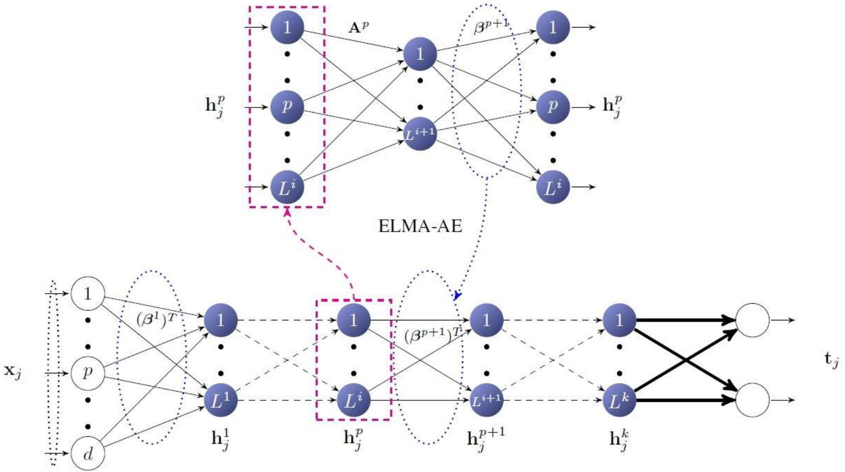
Compressed $(d > L)$ and sparse $(d < L)$ representation ELM-AE solves the following learning problems:

$$\text{Minimize: } ||\boldsymbol{\beta}||_2^2 + C||\mathbf{H}\boldsymbol{\beta} - \mathbf{X}||_2^2 \tag{3}$$

Equal dimension $(d = L)$ representation ELM-AE solves the following learning problems:

$$\begin{aligned} \text{Minimize: } & ||\mathbf{H}\boldsymbol{\beta} - \mathbf{X}||_2^2 \\ \text{Subject to: } & \boldsymbol{\beta}^T\boldsymbol{\beta} = \mathbf{I} \end{aligned} \tag{4}$$

ML-ELM Learning Procedure



Advantage and Disadvantage of ML-ELM

Advantage

- Fast training speed

Disadvantage

- Large number of hidden layer parameters than other multi layer neural networks such as Deep belief Networks (DBN) and Stacked Auto-Encoders (SAE)

Objective

Reduce the number of hidden layer parameters of ML-ELM using multi objective formulation

Multi Layer Multi Objective Extreme Learning Machine (MLMO-ELM)

MLMO-ELM Learning Algorithms

- Multi Objective Extreme Learning Machine Auto-Encoder (MO-ELMAE) learns the hidden layer parameters of Multi Layer Multi Objective Extreme Learning Machine (MLMO-ELM)
- Ridge regression learns the output layer weights of MLMO-ELM

Multi Objective Extreme Learning Machine Auto-Encoder (MO-ELMAE)

MO-ELMAE objective functions

To reduce the number of hidden layer parameters of ELM-AE we can use label information and non-linear information

- Objective function of ELM-AE
- Objective function to learn non-linear weights by using the euclidean distance information of input data
- Objective function to learn weights with label information

MO-ELMAE Objective function

MO-ELMAE objective functions

$$
\begin{aligned}
E = min_{\boldsymbol{\beta}_X, \boldsymbol{\beta}_T} \frac{1}{2} \|\mathbf{H}\boldsymbol{\beta}_X - \mathbf{X}\|_2^2 + \frac{1}{2} \|g(\mathbf{X}\boldsymbol{\beta}_X^T) - \mathbf{X}\mathbf{A}\|_2^2 \\
+ \frac{1}{2} \|g(\mathbf{X}\boldsymbol{\beta}_X^T)\boldsymbol{\beta}_T - \mathbf{T}\|_2^2 \\
+ \frac{C_X}{2} \|\boldsymbol{\beta}_X\|_2^2 + \frac{C_T}{2} \|\boldsymbol{\beta}_T\|_2^2
\end{aligned}
\tag{5}
$$

$\mathbf{H}$ is calculated as:

$$
\begin{aligned}
\mathbf{h}(\mathbf{x}_j) = g(\mathbf{x}_j \mathbf{A} + \mathbf{b}) = [h_1(\mathbf{x}_j), \cdots, h_L(\mathbf{x}_j)] \\
= [\mathbf{a}_1 \cdot \mathbf{x}_j + b_1, \cdots, \mathbf{a}_L \cdot \mathbf{x}_j + b_L]
\end{aligned}
\tag{6}
$$

Where ELM-AE random hidden layer weights and bias $\mathbf{A}$ and $\mathbf{b}$ is calculated as:

$$\text{if } d >= L$$
$$\mathbf{A}^T\mathbf{A} = \mathbf{I}$$
$$\mathbf{b}^T\mathbf{b} = 1 \tag{7}$$

$$\text{if } d < L$$
$$\mathbf{A}\mathbf{A}^T = \mathbf{I}$$
$$\mathbf{b}\mathbf{b}^T = 1 \tag{8}$$

MO-ELMAE Algorithm

There are two learn-able parameters in MO-ELMAE $\boldsymbol{\beta}_T$ $\boldsymbol{\beta}_X$ and can be calculated using alternative optimization as:

- While number of iterations smaller than Maximum iterations
- Calculate $\boldsymbol{\beta}_T$
- Calculate $\boldsymbol{\beta}_X$

Calculating learn-able weights of MO-ELMAE

$$\boldsymbol{\beta}_T = \left(C_T + \mathbf{H}_X^T \mathbf{H}_X\right)^{-1} \mathbf{H}_X^T \mathbf{T} \tag{9}$$

where $\mathbf{H}_X = g(\mathbf{X}\boldsymbol{\beta}_X^T)$.

As $\boldsymbol{\beta}_X$ cannot be calculated analytically, E is differentiated with respect to $\boldsymbol{\beta}_X$ to calculate $\frac{\delta E}{\delta \boldsymbol{\beta}_X}$ as:

$$\begin{aligned}\frac{\delta E}{\delta \boldsymbol{\beta}_X} &= \mathbf{H}^T(\mathbf{H}\boldsymbol{\beta}_X - \mathbf{X}) \\ &+ \mathbf{H}_X.(1 - \mathbf{H}_X).(\mathbf{H}_X \mathbf{A}\mathbf{A}^T - \mathbf{X}\mathbf{A}) \\ &+ \mathbf{H}_X.(1 - \mathbf{H}_X).(\mathbf{H}_X \boldsymbol{\beta}_T {\boldsymbol{\beta}_T}^T - \mathbf{T}\boldsymbol{\beta}_T) \\ &+ C_X\end{aligned} \tag{10}$$

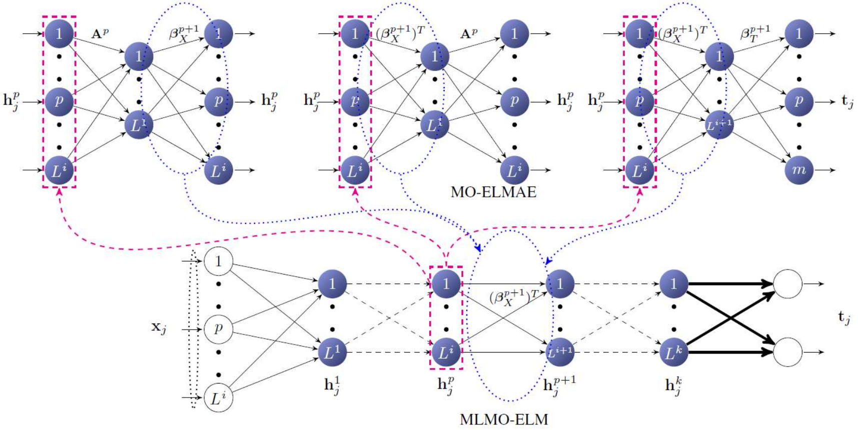
Calculating learn-able weights of MO-ELMAE

$\boldsymbol{\beta}_X$ is calculated iteratively as:

$$\boldsymbol{\beta}_X = \boldsymbol{\beta}_X - \lambda \frac{\delta E}{\delta \boldsymbol{\beta}_X} \tag{11}$$

where λ is the learning rate. We use the off the shelf solver *minfunc* to calculate MO-ELMAE weights $\boldsymbol{\beta}_X$ using Limited memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm. L-BFGS algorithm finds learning rate λ and must not be provided by the user.

MLMO-ELM Learning Procedure



MLMO-ELM Learning Procedure

- MO-ELMAE learns the hidden layer parameters of Multi Layer Multi Objective Extreme Learning Machine (MLMO-ELM)
- MO-ELMAE uses Input data $\mathbf{X}$ to learn the first hidden layer parameters of MLMO-ELM
- MO-ELMAE uses the first hidden layer output $\mathbf{H}^1$ of MLMO-ELM to learn the second hidden layer parameters of MLMO-ELM
- MO-ELMAE uses the p-th hidden layer output $\mathbf{H}^p$ of MLMO-ELM to learn the $p+1$-th hidden later parameters of MLMO-ELM
- Ridge regression learns the output layer parameters of MLMO-ELM

Datasets

- NORB object recognition dataset: Contains stereo images of size $2 \times 96 \times 96$ and for the experiments, NORB images were down-sampled to $2 \times 32 \times 32$. NORB dataset contains 24300 training samples and 24300 testing samples representing 5 object classes
- Optical Character Recognition (OCR) dataset: Contains 42152 training samples and 10000 testing samples representing 26 classes from a-z characters. OCR data are binary pixel images of 16×8.

Experimental Setup

- Experiments were carried out in a workstation with a 2.6 Ghz Xeon E5-2630 v2 processor and 512 GB ram running matlab 2016a
- Average testing accuracy and average training time of ten trials are reported for the MLMO-ELM algorithm

Results

Algorithm	Network Architecture	Testing Accuracy	Training Time
OCR dataset			
DBM	128-2000-2000-26	91.56%	>24h
ML-ELM	128-100-100-15000-26	90.31%(± 0.13)	0.06h
H-ELM	128-200-200-15000-26	90.16%	0.02h
MLMO-ELM (ours)	128-200-3000-26	**91.99%(± 0.06)**	1.7h
NORB dataset			
DBM	2048-4000-4000-4000-5	92.77%	>48h
ML-ELM	2048-2000-2000-4000-5	89.54%(± 0.17)	0.02h
H-ELM	2048-3000-3000-15000-5	91.28%	0.05h
MLMO-ELM (ours)	2048-3000-4000-5	**92.98%(± 0.26)**	5.78h

Discussion

- Results show that porposed MLMO-ELM outperforms ML-ELM, DBM and H-ELM
- Results also show that the number of hidden layer parameters are similar to DBM, but the learning time is significantly lower than DBM
- However, the learning time of MLMO-ELM is higher than ML-ELM and H-ELM

Thank you!