Multi Layer Multi Objective Extreme Learning Machine

CUI DONGSHUN

NTU, Singapore

19/09/2017
Contents

1 Background
   Extreme Learning Machine (ELM)
   Multi Layer Extreme Learning Machine (ML-ELM)
   Extreme Learning Machine Auto-Encoder (ELM-AE)
   ML-ELM Learning Algorithm
   Advantage and Disadvantage of ML-ELM
   Objective

2 Multi Layer Multi Objective Extreme Learning Machine (MLMO-ELM)
   Multi Objective Extreme Learning Machine Auto-Encoder
      (MO-ELMAE)
   MLMO-ELM Learning Algorithm
   Experiments
Extreme Learning Machines (ELMs)

The hidden node parameters which are selected randomly and not tuned:

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

\[
\mathbf{f}_L(x) = \sum_{i=1}^{L} \beta_i G(a_i, b_i, x). 
\]
Extreme Learning Machines (ELMs)

Given the training examples \( \{(x_j, t_j)\}_{j=1}^{N} \subset \mathbb{R}^d \times \mathbb{R}^m \). Then we have:

\[
\sum_{i=1}^{L} \beta_i G(a_i, b_i, x) = t_j, \ j = 1, \ldots, N
\]

or equivalently in matrix form

\[
H\beta = T \tag{2}
\]

where

\[
H = \begin{pmatrix}
G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\
G(a_1, b_1, x_2) & \cdots & G(a_L, b_L, x_2) \\
\vdots & \ddots & \vdots \\
G(a_1, b_1, x_N) & \cdots & G(a_L, b_L, x_N)
\end{pmatrix}
\]

\[
\beta = \begin{pmatrix}
\beta_1^T \\
\beta_2^T \\
\vdots \\
\beta_L^T
\end{pmatrix}
\]

\[
T = \begin{pmatrix}
t_1^T \\
t_2^T \\
\vdots \\
t_N^T
\end{pmatrix}
\]
Extreme Learning Machines (ELMs)

Three-Step Learning Mode

Given a training set \( \mathcal{X} = \{(x_i, t_i) | x_i \in \mathbb{R}^d, t_i \in \mathbb{R}^m, i = 1, \ldots, N\} \), hidden node output function \( G(a, b, x) \), and the number of hidden nodes \( L \),

1. Assign randomly hidden node parameters \((a_i, b_i), i = 1, \ldots, L\).

2. Calculate the hidden layer output matrix \( H \).

3. Calculate the output weight \( \beta \): \( \beta = H^\dagger T \).

where \( H^\dagger \) is the Moore-Penrose generalized inverse of hidden layer output matrix \( H \).
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: the ELM feature mapping is calculated as:
  \[ h(x_j) = g(x_jA + b) \]
  where hidden layer parameters are orthogonal random
  \[ AA^T = I \text{ and } b^Tb = 1. \]

- Sparse ($d < L$) representation: ELM feature mapping is calculated as:
  \[ h(x_j) = g(x_jA + b) \]
  where hidden layer parameters are orthogonal random
  \[ AA^T = I \text{ and } b^Tb = 1. \]
ELM-AE Learning Problems

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

Minimize: $\|\beta\|_2^2 + C\|H\beta - X\|_2^2$  \hspace{1cm} (3)

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

Minimize: $\|H\beta - X\|_2^2$  \hspace{1cm} (4)

Subject to: $\beta^T \beta = I$
ML-ELM Learning Procedure

\[
\begin{align*}
&x_j \\ &\vdots \\ &d \\
\end{align*}
\]

\[
\begin{align*}
&h_j^1 \\
&\vdots \\
&h_j^p \\
\end{align*}
\]

\[
\begin{align*}
&h_j^{p+1} \\
&\vdots \\
&h_j^k \\
\end{align*}
\]

\[
\begin{align*}
&L^1 \\
&\vdots \\
&L^i \\
&\vdots \\
&L^{i+1} \\
&\vdots \\
&L^k \\
\end{align*}
\]

\[
\begin{align*}
&1 \\
&\vdots \\
&1 \\
&\vdots \\
&1 \\
\end{align*}
\]

\[
\begin{align*}
&\mathbf{A}^p \\
&\vdots \\
&\beta_{p+1} \\
\end{align*}
\]

\[
\begin{align*}
&t_j \\
\end{align*}
\]
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 functions

\[ E = \min_{\beta_X, \beta_T} \frac{1}{2} \| H\beta_X - X \|_2^2 + \frac{1}{2} \| g(X\beta_X^T) - XA \|_2^2 \]
\[ + \frac{1}{2} \| g(X\beta_X^T)\beta_T - T \|_2^2 \]
\[ + \frac{C_X}{2} \| \beta_X \|_2^2 + \frac{C_T}{2} \| \beta_T \|_2^2 \]

(5)

H is calculated as:

\[ h(x_j) = g(x_jA + b) = [h_1(x_j), \cdots, h_L(x_j)] \]
\[ = [a_1 \cdot x_j + b_1, \cdots, a_L \cdot x_j + b_L] \]

(6)
Where ELM-AE random hidden layer weights and bias $\mathbf{A}$ and $\mathbf{b}$ is calculated as:

\[
\begin{align*}
\text{if } d \geq L & \quad \mathbf{A}^T \mathbf{A} = \mathbf{I} \\
\text{if } d < L & \quad \mathbf{b}^T \mathbf{b} = 1
\end{align*}
\] (7)

\[
\begin{align*}
\mathbf{A} \mathbf{A}^T &= \mathbf{I} \\
\mathbf{b} \mathbf{b}^T &= 1
\end{align*}
\] (8)
MO-ELMAE Algorithm

There are two learn-able parameters in MO-ELMAE $\beta_T \beta_X$ and can be calculated using alternative optimization as:

- While number of iterations smaller than Maximum iterations
- Calculate $\beta_T$
- Calculate $\beta_X$
Calculating learn-able weights of MO-ELMAE

\[ \beta_T = \left( C_T + H_X^T H_X \right)^{-1} H_X^T T \]  

(9)

where \( H_X = g(X\beta_X^T) \).

As \( \beta_X \) cannot be calculated analytically, \( E \) is differentiated with respect to \( \beta_X \) to calculate \( \frac{\delta E}{\delta \beta_X} \) as:

\[
\frac{\delta E}{\delta \beta_X} = H^T (H\beta_X - X) \\
+ H_X (1 - H_X) (H_X A A^T - X A) \\
+ H_X (1 - H_X) (H_X \beta_T \beta_T^T - T \beta_T) \\
+ C_X
\]

(10)
Multi Layer Multi Objective Extreme Learning Machine (MLMO-ELM)  

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

Calculating learn-able weights of MO-ELMAE

\( \beta_X \) is calculated iteratively as:

\[
\beta_X = \beta_X - \lambda \frac{\delta E}{\delta \beta_X}
\]  \( (11) \)

where \( \lambda \) is the learning rate. We use the off the shelf solver minfunc to calculate MO-ELMAE weights \( \beta_X \) using Limited memory Broyden Fletcher Goldfarb Shanno (L-BFGS) algorithm. L-BFGS algorithm finds learning rate \( \lambda \) 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 layer 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 $16 \times 8$. 
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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Network Architecture</th>
<th>Testing Accuracy</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OCR dataset</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBM</td>
<td>128-2000-2000-26</td>
<td>91.56%</td>
<td>&gt;24h</td>
</tr>
<tr>
<td>ML-ELM</td>
<td>128-100-100-15000-26</td>
<td>90.31% (± 0.13)</td>
<td>0.06h</td>
</tr>
<tr>
<td>H-ELM</td>
<td>128-200-200-15000-26</td>
<td>90.16%</td>
<td>0.02h</td>
</tr>
<tr>
<td>MLMO-ELM (ours)</td>
<td>128-200-3000-26</td>
<td><strong>91.99% (± 0.06)</strong></td>
<td>1.7h</td>
</tr>
<tr>
<td><strong>NORB dataset</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBM</td>
<td>2048-4000-4000-4000-5</td>
<td>92.77%</td>
<td>&gt;48h</td>
</tr>
<tr>
<td>ML-ELM</td>
<td>2048-2000-2000-4000-5</td>
<td>89.54% (± 0.17)</td>
<td>0.02h</td>
</tr>
<tr>
<td>H-ELM</td>
<td>2048-3000-3000-15000-5</td>
<td>91.28%</td>
<td>0.05h</td>
</tr>
<tr>
<td>MLMO-ELM (ours)</td>
<td>2048-3000-4000-5</td>
<td><strong>92.98% (± 0.26)</strong></td>
<td>5.78h</td>
</tr>
</tbody>
</table>
Discussion

- Results show that proposed 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!