Extreme Learning Machine

Outline

- Extreme Learning Machine Motivation.
- Structure of the proposed Extreme Learning Machine (ELM).
- Basis of Extreme Learning Machine.
- Show an Example of training small data using ELM algorithm.
- Show an Example of predicting small data using ELM algorithm.

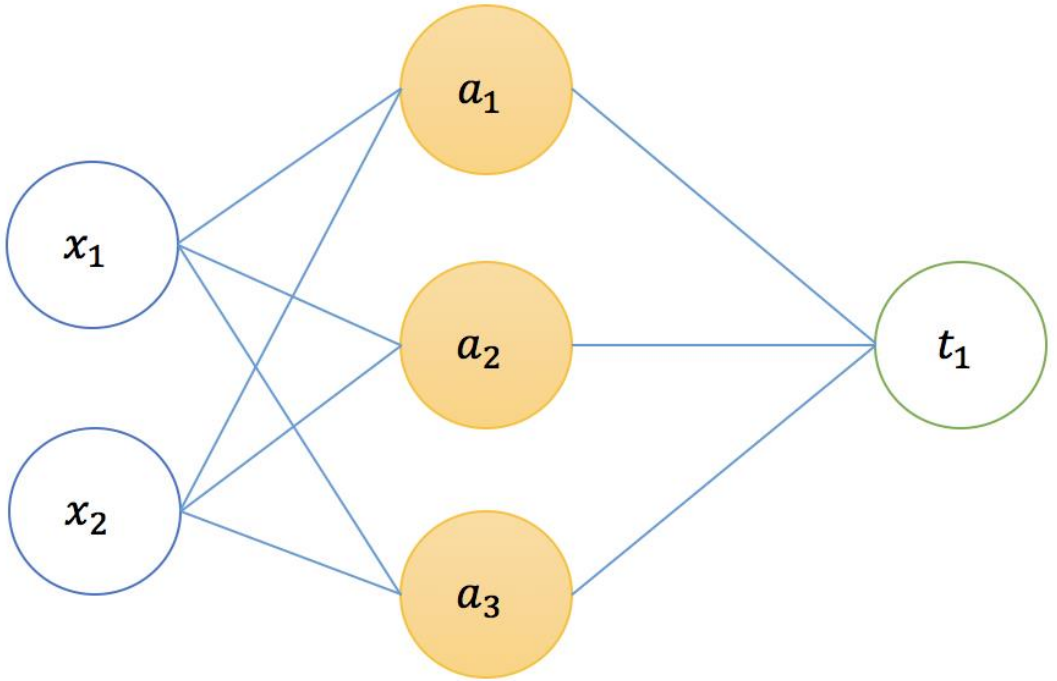
Motivation

- Feedforward Neural Nets such as back-propagation are slow and the reason is:
  - Training using gradient based learning algorithms e.g. many iterative steps required to achieve better performance.
  - The weights are adjusted iteratively using gradient methods.
- Extreme Learning Machine (ELM) was proposed to overcome these issues and offer better generalization performance.

The proposed algorithm - Extreme Learning Machine (ELM)

- Single hidden layer feedforward neural network.
- Randomly chosen input weights and hidden unit biases.
- Weights and biases need not to be adjusted.
- Train the net by finding least square solution $\hat{\beta}$ to the linear system $H\beta = T$
- Analytically determines output weights.
- Provide best generalization performance at extremely fast learning speed.

Structure of Extreme Learning Machine



Initial idea

- $T\ labels, N$ observations, and $\widetilde{N}$ hidden units, single hidden layer feedforward neural net.
- When: $N = \widetilde{N}$
- $H = (w_1, \dots, w_{\widetilde{N}}, b_1, \dots, b_{\widetilde{N}})$, hidden layer output matrix, nonsingular and square matrix.
- Nonlinear infinitely differentiable activation function.
- No updates to input weights and hidden layer's biases.
- linear system $H\beta = T$ has exactly one solution
- $\beta = H^{-1} * T$
- Can learn N distinct observations with 0 error.

H is non-square or singular

- Finding a solution to linear system $H\beta = T$ when H is non-square, or singular is:
- $\hat{\beta} = H^{+}T$

To calculate the pseudo inverse

- In case H_{nXm} is full rank matrix:
- Fat Matrix (Case $m < n$):
  - $H^{+} = H^{T}(HH^{T})^{-1}$
- Skinny Matrix (Case $m > n$):
  - $H^{+} = (H^{T}H)^{-1}H^{T}$
- If H is not in full rank the pseudoinverse can be computed using Singular Value Decomposition (SVD) or other method.

To calculate the pseudo inverse

- The Computation of Moore-Penrose Pseudo inverse through SVD (Singular Value Decomposition) is discussed in Class.
- Please refer class note

How to compute the pseudoinverse in case matrix H is not invertible i.e. non-square and H not full rank?

- First construct the Singular Value Decomposition (SVD)
- $H = USV^T$
- You can solve SVD and find matrices U, S, and V by using two equations:
- $H^T H = VS^T SV^T$ and $HH^T = USS^T U^T$
- The col of V are the eigenvectors of $H^T H$, The col of U are the eigenvectors of HH^T, and the Eigenvalues are the entries of S.
- Then find the pseudoinverse as follow
- $H^+ = V^T S^+ U$

Important properties of solution $\widehat{\boldsymbol{\beta}} = \boldsymbol{H}^{+}\boldsymbol{T}$

- **Minimum training error:**

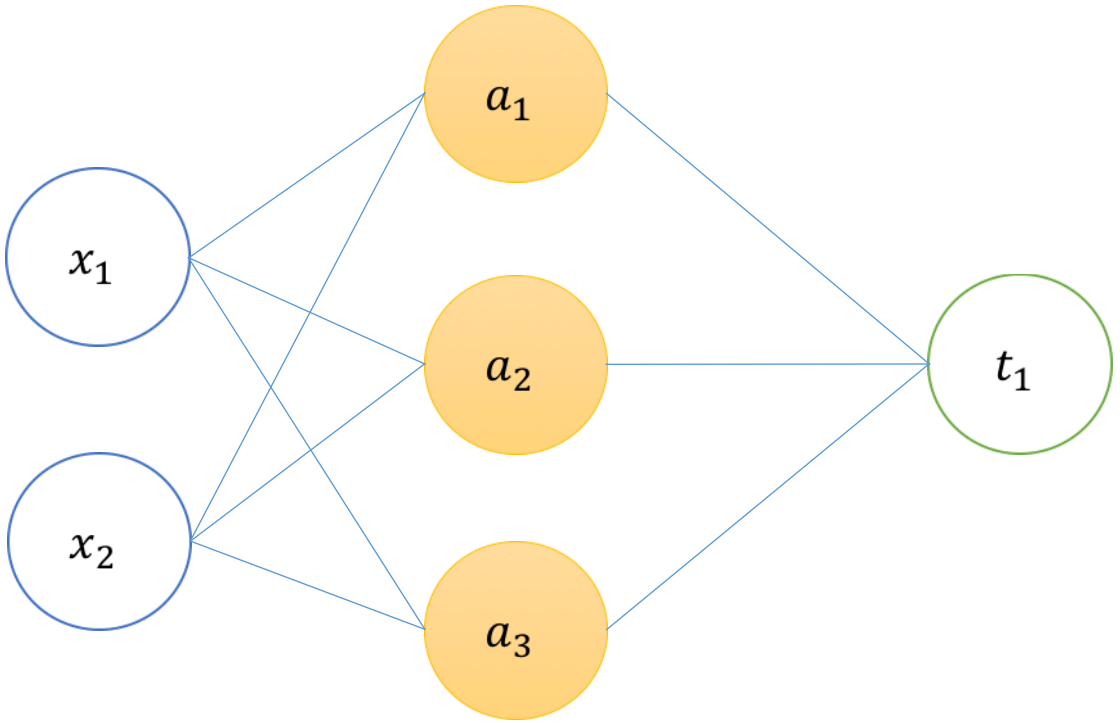
$$\|\mathbf{H}\hat{\beta} - \mathbf{T}\| = \|\mathbf{H}\mathbf{H}^{\dagger}\mathbf{T} - \mathbf{T}\| = \min_{\beta} \|\mathbf{H}\beta - \mathbf{T}\|.$$

- **Smallest norm of weights :**

- $$\|\hat{\beta}\| = \|\mathbf{H}^{\dagger}\mathbf{T}\| \leqslant \|\beta\|,$$
  $$\forall \beta \in \left\{\beta : \|\mathbf{H}\beta - \mathbf{T}\| \leqslant \|\mathbf{H}\mathbf{z} - \mathbf{T}\|, \forall \mathbf{z} \in \mathbf{R}^{\tilde{N}\times N}\right\}$$

- **Unique minimum norm Least-squares solution of $\boldsymbol{H\beta = T, which\ is\ \widehat{\beta} = H^{+}T}$**

Extreme Learning Machine



X W b β T

$$\mathbf{H}\beta = \mathbf{T},$$

where

$$\mathbf{H}(\mathbf{w}_1, \ldots, \mathbf{w}_{\tilde{N}}, b_1, \ldots, b_{\tilde{N}}, \mathbf{x}_1, \ldots, \mathbf{x}_N) = \begin{bmatrix} g(\mathbf{w}_1 \cdot \mathbf{x}_1 + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_1 + b_{\tilde{N}}) \\ \vdots & \cdots & \vdots \\ g(\mathbf{w}_1 \cdot \mathbf{x}_N + b_1) & \cdots & g(\mathbf{w}_{\tilde{N}} \cdot \mathbf{x}_N + b_{\tilde{N}}) \end{bmatrix}_{N \times \tilde{N}},$$

$$\beta = \begin{bmatrix} \beta_1^{\mathrm{T}} \\ \vdots \\ \beta_{\tilde{N}}^{\mathrm{T}} \end{bmatrix}_{\tilde{N} \times m} \quad \text{and} \quad \mathbf{T} = \begin{bmatrix} \mathbf{t}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{t}_N^{\mathrm{T}} \end{bmatrix}_{N \times m}.$$

$\tilde{N} = \#\ Hidden\ units$
$N = \#\ of\ observation$

ELM Algorithm

Algorithm ELM: Given a training set $\aleph = \{(\mathbf{x}_i, \mathbf{t}_i) | \mathbf{x}_i \in \mathbf{R}^n, \mathbf{t}_i \in \mathbf{R}^m, i = 1, \ldots, N\}$, activation function $g(x)$, and hidden node number $\tilde{N}$,

Step 1: Randomly assign input weight $\mathbf{w}_i$ and bias b_i, $i = 1, \ldots, \tilde{N}$.

Step 2: Calculate the hidden layer output matrix $\mathbf{H}$.

Step 3: Calculate the output weight β

$$\beta = \mathbf{H}^{\dagger}\mathbf{T}, \tag{16}$$

where $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_N]^{\mathrm{T}}$.

ELM Training Example

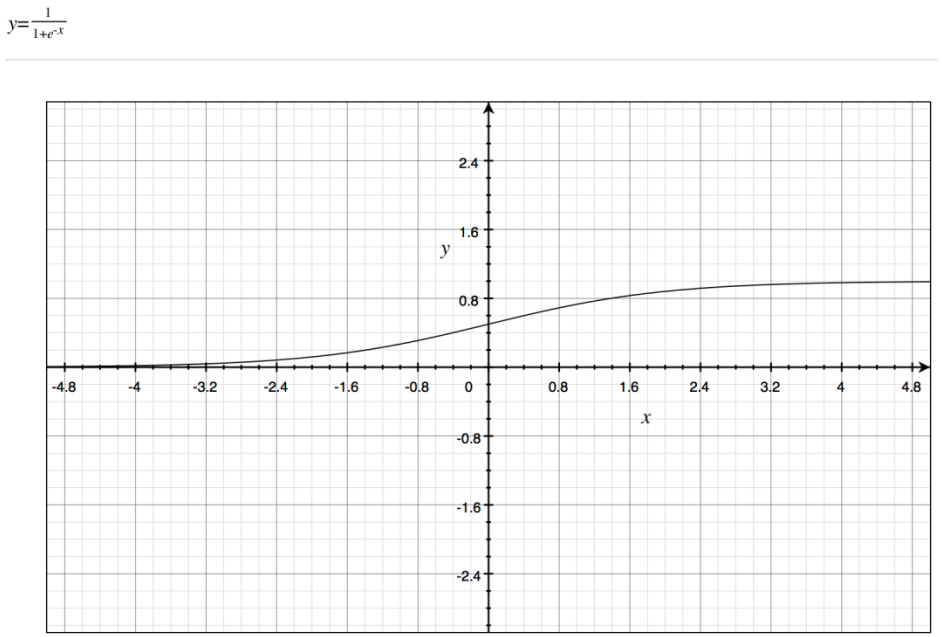
- $X_1 = (-2.2, 0), (1.8, 0),\ X_2 = (-1.2, 1), (-1.4, 1), X_3 = (2.6, 0), (-1.7, 0),\ X_4 = (1.9, 1), (2.1, 1)$
- $W_1 = (-0.3776, 0.2040), W_2 = (0.0571, -0.4741), W_3 = (-0.6687, 0.3082),$
- $b_1 = (0.6892), b_2 = (0.7482), b_3(0.4505)$
- $X = \begin{bmatrix} -2.2 & 1.8 \\ -1.2 & -1.4 \\ 2.6 & -1.7 \\ 1.9 & 2.1 \end{bmatrix} . W = \begin{bmatrix} -0.3776 & 0.0571 & -0.6687 \\ 0.2040 & -0.4741 & 0.3082 \end{bmatrix} + b = \begin{bmatrix} 0.6892 & 0.7482 & 0.4505 \\ 0.6892 & 0.7482 & 0.4505 \\ 0.6892 & 0.7482 & 0.4505 \\ 0.6892 & 0.7482 & 0.4505 \end{bmatrix}$

ELM Training Example (continued)

- Since we have tempH matrix, now feed it to an activation function e.g. sigmoid.

- $\text{tempH} = \begin{bmatrix} 1.8870 & -0.2306 & 2.4763 \\ 0.8567 & 1.3434 & 0.8215 \\ -0.6392 & 1.7025 & -1.8120 \\ 0.4001 & -0.1389 & -0.1729 \end{bmatrix}$

- $H = \begin{bmatrix} \frac{1}{1+e^{-tempH_{1,1}}} & \frac{1}{1+e^{-tempH_{1,2}}} & \frac{1}{1+e^{-tempH_{1,3}}} \\ \frac{1}{1+e^{-tempH_{2,1}}} & \frac{1}{1+e^{-tempH_{2,2}}} & \frac{1}{1+e^{-tempH_{2,3}}} \\ \frac{1}{1+e^{-tempH_{3,1}}} & \frac{1}{1+e^{-tempH_{3,2}}} & \frac{1}{1+e^{-tempH_{3,3}}} \\ \frac{1}{1+e^{-tempH_{4,1}}} & \frac{1}{1+e^{-tempH_{4,2}}} & \frac{1}{1+e^{-tempH_{4,3}}} \end{bmatrix}$



ELM Training Example (continued)

- $H = \begin{bmatrix} 0.8684 & 0.4426 & 0.9225 \\ 0.7020 & 0.7930 & 0.6946 \\ 0.3454 & 0.8459 & 0.1404 \\ 0.5987 & 0.4653 & 0.4569 \end{bmatrix}$

- $H\beta = T$

- $\hat{\beta} = (H^T H)^{-1} H^T T^T$

ELM Training Example (continued)

- $H^T H = \begin{bmatrix} 0.8684 & 0.7020 & 0.3454 & 0.5987 \\ 0.4426 & 0.7930 & 0.8459 & 0.4653 \\ 0.9225 & 0.6946 & 0.1404 & 0.4569 \end{bmatrix} * \begin{bmatrix} 0.8684 & 0.4426 & 0.9225 \\ 0.7020 & 0.7930 & 0.6946 \\ 0.3454 & 0.8459 & 0.1404 \\ 0.5987 & 0.4653 & 0.4569 \end{bmatrix}$
- $\hat{\beta} = \left(\begin{bmatrix} 1.7247 & 1.5118 & 1.6107 \\ 1.5118 & 1.7568 & 1.2904 \\ 1.6107 & 1.2904 & 1.5618 \end{bmatrix} \right)^{-1} * \left(\begin{bmatrix} 0.8684 & 0.7020 & 0.3454 & 0.5987 \\ 0.4426 & 0.7930 & 0.8459 & 0.4653 \\ 0.9225 & 0.6946 & 0.1404 & 0.4569 \end{bmatrix} . \begin{bmatrix} -1 \\ +1 \\ -1 \\ +1 \end{bmatrix} \right)$
- $\hat{\beta} = \begin{bmatrix} 61.8730 & -16.2165 & -50.4104 \\ -16.2165 & 5.6983 & 12.0158 \\ -50.4104 & 12.0158 & 42.7002 \end{bmatrix} * \begin{bmatrix} 0.0869 \\ -0.0301 \\ 0.0886 \end{bmatrix}$
- $\hat{\beta} = \begin{bmatrix} 1.3965 \\ -0.5156 \\ -0.9575 \end{bmatrix}$

- $H = \begin{bmatrix} 0.8684 & 0.4426 & 0.9225 \\ 0.7020 & 0.7930 & 0.6946 \\ 0.3454 & 0.8459 & 0.1404 \\ 0.5987 & 0.4653 & 0.4569 \end{bmatrix}$ $\beta = \begin{bmatrix} 1.3965 \\ -0.5156 \\ -0.9575 \end{bmatrix}$

- $T = H\beta$

- $T = \begin{bmatrix} 0.8684 & 0.4426 & 0.9225 \\ 0.7020 & 0.7930 & 0.6946 \\ 0.3454 & 0.8459 & 0.1404 \\ 0.5987 & 0.4653 & 0.4569 \end{bmatrix} * \begin{bmatrix} 1.3965 \\ -0.5156 \\ -0.9575 \end{bmatrix} = \begin{bmatrix} -0.1013 \\ 0.0936 \\ 0.0881 \\ -0.1587 \end{bmatrix}$

ELM Classify Example

- Based on the model we built we are going to classify a test instance
- $\mathrm{X}_{test} = (2.5, 0), (-1.3, 0)$
- $tempH_{test} = \mathrm{X_{test}} * \mathrm{W} + \mathrm{b}\ (\mathrm{W}, \text{and b from learned model})$
- $\left([2.5 \quad -1.3] * \begin{bmatrix} -0.3776 & 0.0571 & -0.6687 \\ 0.2040 & -0.4741 & 0.3082 \end{bmatrix}\right) + [0.6892 \quad 0.7482 \quad 0.4505\ ]$
- $\mathrm{tempH_{test}} = [-0.5199 \quad 1.5071 \quad -1.6218\ ]$
- $\mathrm{H_{test}} = \left[\frac{1}{1+\mathrm{e}^{-\left(\mathrm{tempH_{test_{1,1}}}\right)}} \quad \frac{1}{1+\mathrm{e}^{-\left(\mathrm{tempH_{test_{1,2}}}\right)}} \quad \frac{1}{1+\mathrm{e}^{-\left(\mathrm{tempH_{test_{1,3}}}\right)}}\right]$

ELM Classify Example

- $H_{test} = [0.3729 \quad 0.8186 \quad 0.1650\ ]$
- $T_{test} = H_{test} * \beta$
- $T_{test} = [0.3729 \quad 0.8186 \quad 0.1650\ ] * \begin{bmatrix} 1.3965 \\ -0.5156 \\ -0.9575 \end{bmatrix} = -0.0593$

ELM have better generalization compared to gradient based algorithm such as BP

Extreme Learning Machine (ELM)	Gradient based Algorithms e.g. Backpropagation
Unique minimum solution	Prone to local minima convergence trap
Reach both smallest training error and smallest weight norm.	Minimizes Error alone.
Does not need a stopping methods.	Overtraining when objective function doesn't have proper stopping methods and validation.