

Extreme Learning Machine

作者：凯鲁嘎吉 - 博客园 <http://www.cnblogs.com/kailugaji/>

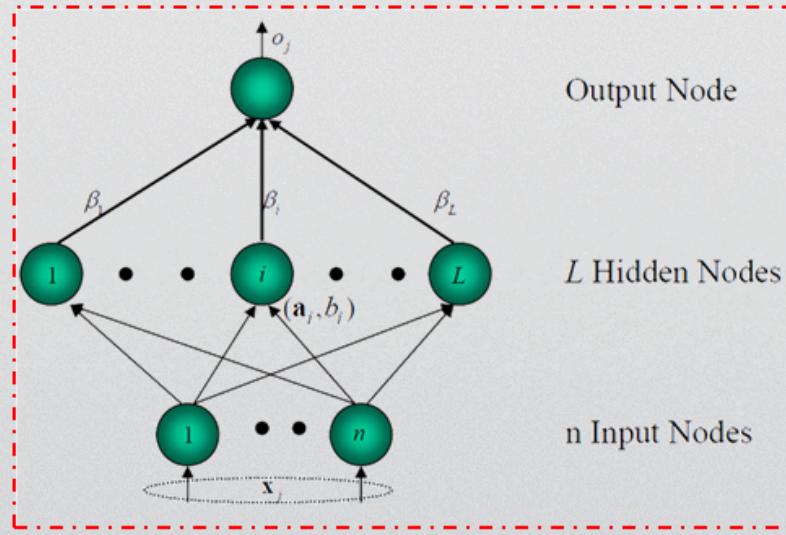
更新：只做科普，别做学术。

1. ELM

2004年南洋理工大学黄广斌提出了ELM算法。极限学习机 (ELM Extreme Learning Machine)是一种快速的单隐层前馈神经网络 (SLFN) 训练算法。

该算法的特点是在网络参数的确定过程中,隐层节点参数(a, b)随机选取,在训练过程中无需调节,只需要设置隐含层神经元的个数,便可以获得唯一的最优解;而网络的外权(即输出权值)是通过最小化平方损失函数得到的最小二乘解(最终化归成求解一个矩阵的 Moore-Penrose 广义逆问题).这样网络参数的确定过程中无需任何迭代步骤,从而大大降低了网络参数的调节时间。与传统的训练方法相比,该方法具有学习速度快、泛化性能好等优点。

➤ Extreme Learning Machine



Algorithm ELM:

Input: Training set $X = \{(x_j, t_j) | x_j \in R^n, t_j \in R^m\}, j = 1, 2, \dots, n$, activation function $g(x)$ and hidden node number \tilde{N} .

Output: Output weights of SLFNs.

Step 1: Randomly assign input weights \mathbf{a}_i and biases $b_i, i = 1, 2, \dots, \tilde{N}$;

Step 2: Calculate the hidden layer output matrix H ;

Step 3: Calculate the output weights as $\hat{\beta} = H^\dagger T$.

✓ Single hidden layer feedforward networks

$$\begin{aligned} & \| \mathbf{H}(\hat{\mathbf{a}}_1, \dots, \hat{\mathbf{a}}_{\tilde{N}}, \hat{b}_1, \dots, \hat{b}_{\tilde{N}}) \hat{\beta} - \mathbf{T} \| \\ &= \min_{\mathbf{a}_i, b_i, \beta} \| \mathbf{H}(\mathbf{a}_1, \dots, \mathbf{a}_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}) \beta - \mathbf{T} \| \end{aligned}$$

✓ Extreme learning machine

$$\begin{aligned} & \| \mathbf{H}(\hat{\mathbf{a}}_1, \dots, \hat{\mathbf{a}}_{\tilde{N}}, \hat{b}_1, \dots, \hat{b}_{\tilde{N}}) \hat{\beta} - \mathbf{T} \| \\ &= \min_{\beta} \| \mathbf{H}(\mathbf{a}_1, \dots, \mathbf{a}_{\tilde{N}}, b_1, \dots, b_{\tilde{N}}) \beta - \mathbf{T} \| \end{aligned}$$

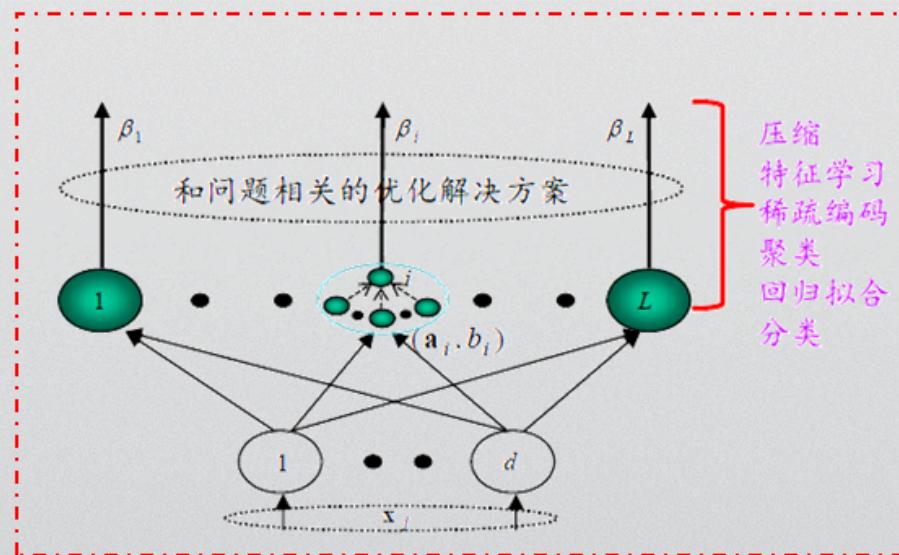
$$\hat{\beta} = \arg \min_{\beta} \| \mathbf{H}\beta - \mathbf{T} \|_F$$

$$\hat{\beta} = \mathbf{H}^\dagger \mathbf{T}$$

若 $\mathbf{H}^\top \mathbf{H}$ 非奇异, $\mathbf{H}^\dagger = (\mathbf{H}^\top \mathbf{H} + \lambda \mathbf{I})^{-1} \mathbf{H}^\top$

若 $\mathbf{H}\mathbf{H}^\top$ 非奇异, $\mathbf{H}^\dagger = \mathbf{H}^\top (\mathbf{H}^\top \mathbf{H} + \lambda \mathbf{I})^{-1}$

➤ Extreme Learning Machine



L 随机隐层节点（几乎任何非线性阶段连续函数）：
 $h_i(\mathbf{x}) = G_i(\mathbf{a}_i, b_i, \mathbf{x})$

虽然生物神经元的输出函数的数学模型未知，但是几乎所有生物神经元可以看成是非线性阶段连续函数，为 ELM 理论所覆盖。

$$\text{ELM 网络输出: } f_L(x) = \sum_{i=1}^L \beta_i G(\mathbf{a}_i, b_i, \mathbf{x})$$

$$\text{ELM 特征映射: } \mathbf{h}(x) = [G(\mathbf{a}_1, b_1, \mathbf{x}), \dots, G(\mathbf{a}_L, b_L, \mathbf{x})]$$

隐层节点输出函数（单节点）：

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

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

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

隐层复合节点输出函数（子网络节点）

2. H-ELM

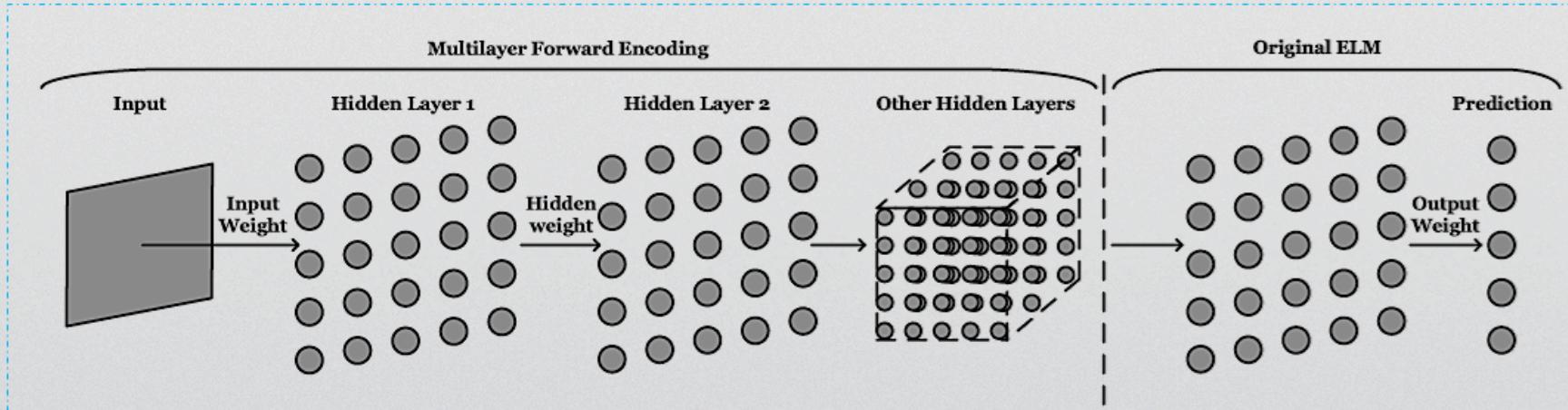
Hierarchical Extreme Learning Machine (H-ELM) 是 2015 年提出的一个 ELM 的改进算法。分为两个阶段：

1. 非监督式的特征编码。在第一阶段，基于 ELM 的稀疏自编码器被设计用来从输入数据中抽取多层的稀疏特征。通过逐层自编码获得每层的权重矩阵，权重确定后，无需微调。
2. 监督式的特征分类。在第二阶段，第一阶段获得的高层次特征（维度可能比原始数据大）将被一个随机矩阵打散，将打散后的数据作为原始 ELM 的输入，最后用原始 ELM 来解决

分类或者回归问题。

与ELM的区别在于自编码阶段，权重矩阵的惩罚项用的是L1范数，参数的更新公式用Fast Iterative Shrinkage-thresholding (FISTA)求解，而ELM的权重惩罚项用的L2范数，用岭回归求解。

➤ Hierarchical Extreme Learning Machine



- 1. 非监督式的特征编码。在第一阶段，基于ELM的稀疏自编码器被设计用来从输入数据中抽取多层的稀疏特征。
- 2. 监督式的特征分类。在第二阶段，第一阶段获得的高层次特征将被一个随机矩阵分散，最后用原始ELM来解决分类或者回归问题。

Tang J, Deng C, Huang G B. Extreme learning machine for multilayer perceptron[J]. IEEE transactions on neural networks and learning systems, 2015, 27(4): 809-821.

➤ Hierarchical Extreme Learning Machine

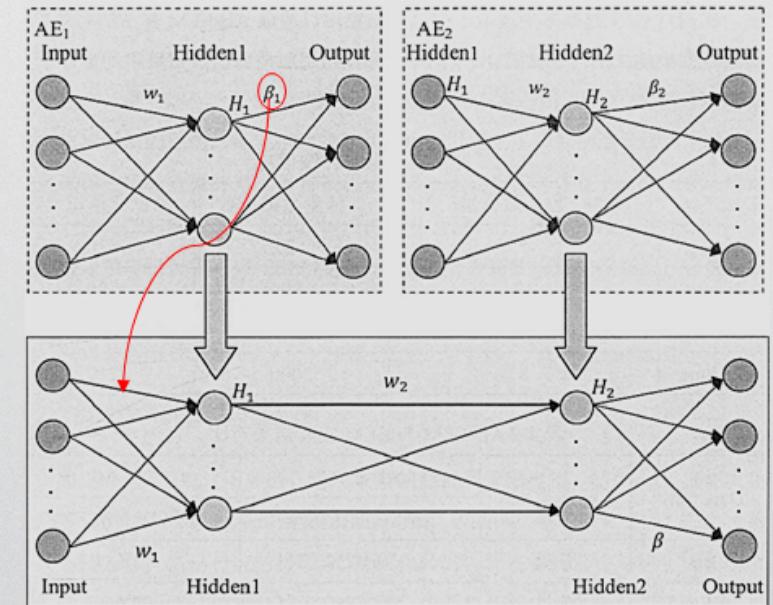
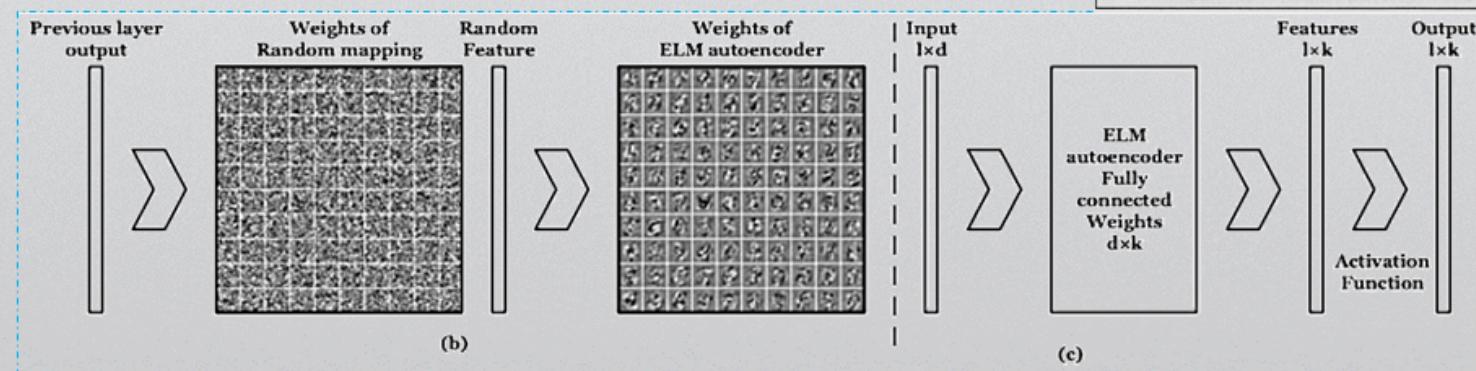
1. 非监督式的特征编码

$$\min L_{ELM} = \|\mathbf{H}\boldsymbol{\beta} - \mathbf{T}\|_2^2 + \lambda \|\boldsymbol{\beta}\|_2^2$$

$$\min L_{H-ELM} = \|\mathbf{H}\boldsymbol{\beta} - \mathbf{X}\|_2^2 + \|\boldsymbol{\beta}\|_1$$

$$\mathbf{H}_i = g(\mathbf{H}_{i-1}\boldsymbol{\beta})$$

2. 监督式的特征分类



Tang J, Deng C, Huang G B. Extreme learning machine for multilayer perceptron[J]. IEEE transactions on neural networks and learning systems, 2015, 27(4): 809-821.

➤ Hierarchical Extreme Learning Machine

$$O_{\beta} = \arg \min_{\beta} \{ \|H\beta - X\|^2 + \|\beta\|_L\} \quad (2.19)$$

为了明确表示L1优化问题的优化方法，上式可以重写成如下公式：

$$O_{\beta} = p(\beta) + q(\beta) \quad (2.20)$$

其中 $p(\beta) = \|H\beta - X\|^2$, $q(\beta) = \|\beta\|_L$, H-ELM中采用基于L1范数的FISTA快速解此类问题，FISTA具体步骤如下：

1. 计算光滑凸函数P的利普希茨常量L;
2. 初始化参数 $y_1 = \beta_0$, $t_1 = 1$, 然后对于k($k \geq 1$)进行如下迭代:

(a) $\beta_k = p_L(y_k)$, 其中 p_L 为

$$p_L = \arg \min_{\beta} \left\{ \frac{L}{2} \|\beta - (\beta_{k-1} - \frac{1}{L} \nabla f(\beta_{k-1}))\|^2 + g(\beta) \right\}$$

(b) $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$

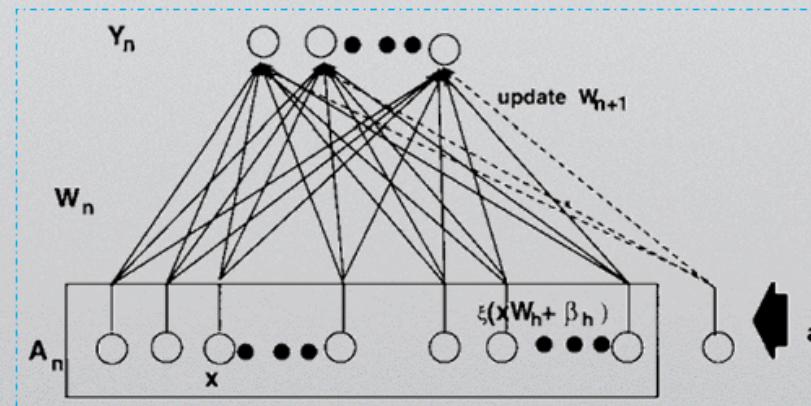
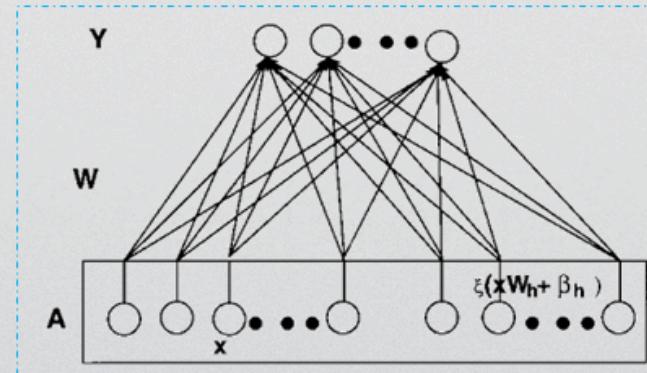
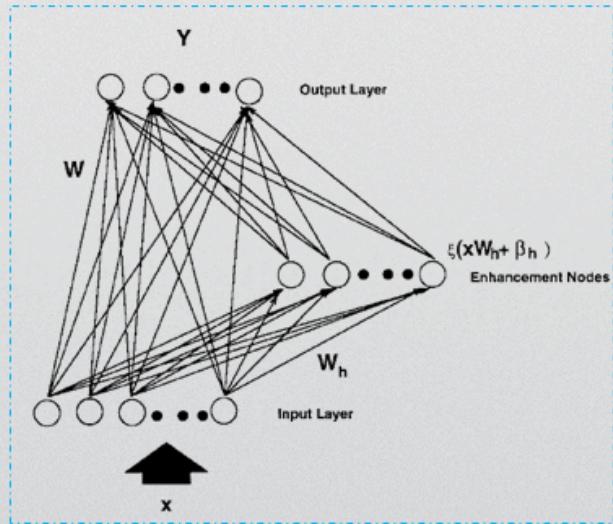
(c) $y_{k+1} = \beta_k + \left(\frac{t_{k-1}}{t_{k+1}} \right) (\beta_k - \beta_{k-1})$

Tang J, Deng C, Huang G B. Extreme learning machine for multilayer perceptron[J]. IEEE transactions on neural networks and learning systems, 2015, 27(4): 809-821.

3. RVFL

Random Vector Functional-link Network (RVFL)算法是1994年提出的算法，与ELM相比，它增加了从输入层到输出层的连接权重。输入层到隐层的权重与隐层的偏置还是随机赋权，只有输入层到输出层与隐层到输出层的权重需要用最小二乘法或者其他方法求解。

► Random Vector Functional-link Network (RVFL)



4. 参考

[1] ELM官方网址: [Extreme Learning Machines](#)

[2] Huang G B, Zhu Q Y, Siew C K. [Extreme learning machine: theory and applications](#)[J]. Neurocomputing, 2006, 70(1-3): 489-501.

- [3] [MATLAB程序：ELM极速学习机](#)
- [4] [论战Yann LeCun：谁能解释极限学习机（ELM）牛X在哪里？](#)
- [5] Tang J, Deng C, Huang G B. [Extreme learning machine for multilayer perceptron](#)[J]. IEEE transactions on neural networks and learning systems, 2015, 27(4): 809-821.
- [6] Pao Y H, Park G H, Sobajic D J. [Learning and generalization characteristics of the random vector functional-link net](#)[J]. Neurocomputing, 1994, 6(2): 163-180.
- [7] [伪逆总结 - CSDN](#)
- [8] 对ELM的质疑：[Extreme Learning Machine: Duplicates Others' Papers from 1988-2007](#)
- [9] 王常飞. 融合分层极限学习机算法研究[D]. 湘潭大学, 2018.
- [10] [软阈值迭代算法 \(ISTA\) 和快速软阈值迭代算法 \(FISTA\)](#)