# Extreme Learning Machine

## Background

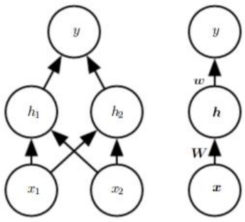
For deep learning area, how to fix the best set of hyperparameters has always been a tough problem. In this paper, Huang (2006) proposed a new way to treat the weight and intercept in each hidden layer fixed, which are generated through some probability density function. In this way, we can use least square estimation during the output layer to fix the data, which will avoid the traditional gradient-based methods and saves lots of computation.

## Simulated Data

In this project, I am going to reproduce the model with some simulated data:

$$y = f^{(2)}(\boldsymbol{h}; \boldsymbol{w}, b)$$

$$\boldsymbol{h} = f^{(1)}(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c})$$




$$f(\boldsymbol{x}; \boldsymbol{W}, \boldsymbol{c}, \boldsymbol{w}, b) = f^{(2)}(f^{(1)}(\boldsymbol{x})).$$

Here,

$$f^{(1)}(x; W, c) = g(W^T x + c), f^{(2)}(h; \omega, b) = \omega^T h + b$$

$$f(x; W, c, \omega, b) = \omega^T \max\{0, W^T x + c\} + b$$

We choose the function truncated at 0 to destroy the linear properties of the whole function f. It is called activation function. So my idea is to first generate $x_{1i}$ and $x_{2i}$ following some distribution, the distribution doesn't matter but I want them to be positive so maybe half-normal is the best choice. Then according to the generated $x_{1i}$ and $x_{2i}$, I will generate $y_i$, which is our simulated response variable.

$$y_i = \beta_1 x_{1i} + \beta_2 x_{1i}^2 + \beta_3 x_{1i}^3 + \beta_4 x_{2i}^{-1} + \beta_4 x_{2i}^{-2} + \beta_4 x_{2i}^{-3} + \varepsilon_i, \varepsilon_i \sim^{i.i.d.} N(0, \sigma^2)$$

## ELM

First, I will fix the number of hidden layers and the number of hidden units in each hidden layer. I will need cross validation to explore the best set of combinations. After fixing these two important indices, I will generate the $W$ and $c$ (i.e., Slope and intercept) in each hidden layer randomly and calculated the $h$ of the last hidden layer, then use this output $h$ to do the least square estimate and then calculate the fitted output, compare the output distribution with the true data and evaluate whether the model has learned the properties of the data generating process.

## Reference


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