**BASIC ELM (NON KERNELIZED)**

# DEFINISI

Menurut Huang(Guang-bin Huang, Zhu, & Siew, 2004), *Extreme Learning Machine* (ELM) adalah algoritma baru yang dikembangkan dari *Feedforward Neural Networks* (FNs) dengan *single hidden layer.* Algoritma ELM juga sering disebut sebagai *Single hidden Layer Feedforward Neural Networks* (SLFNs).

Algoritma FNs merupakan salah satu algoritma *neural networks*. FNs terdiri dari beberapa ataupun banyak *neuron* sebagai unit pemproses yang dikelola oleh *layer.* Tiap *neuron* dalam *layer* ini terhubung dengan *neuron* pada *layer* sebelumnya. Hubungan antar *neuron* ini juga memiliki bobot. Pada prinsipnya, data yang dimasukan sebagai *input* diproses oleh *layer* demi *layer* sampai menjadi *output.* Selama proses berlangsung sebagai pengklasifikasi, tidak terjadi *feedback* antar *layer.* Dan karena hal inilah maka algoritma disebut *feedforward* *neural networks*.

Dikembangkannya ELM ini dilatarbelakangi oleh proses pembelajaran dari FNs yang memerlukan waktu yang lama. Hal ini dikarenakan oleh 2 faktor yaitu:

1. Menggunakan algoritma *gradient based* untuk *training*, yang mana algoritma ini memang memerlukan waktu yang lama dalam bekerja.
2. Semua parameter dioptimasi secara iteratif dengan menggunakan beberapa algoritma semisal *backpropagration*

Pembuatan algoritma ELM ini berhasil meningkatkan kinerja yang lebih baik dari FNs. Kinerja yang dimaksud adalah dari sisi waktu komputasi yang jauh lebih cepat maupun generalisasi yang lebih baik jika dibandingkan dengan FNs yang berbasis *backprogration*(Guang Bin Huang, Zhu, & Siew, 2006). Alasan utama atas peningkatan kinerja ELM ini adalah karena ELM dibuat dengan inisialisasi secara random tetapi tanpa dievaluasi secara iteratif.

Namun demikian, ternyata penentuan nilai random pada ELM dalam hal pembobotan pada parameter W dan nilai bias di input layer ini menyebabkan kinerja kurang efektif karena tergantung pada nilai yang dimasukan pada parameter W pada input layer(Y. Wang, Cao, & Yuan, 2011). Selain itu, Wang(D. Wang, Wang, & Ji, 2015) mengatakan bahwa dengan adanya nilai random ini menyebabkan kinerja algoritma menjadi tidak stabil.

Penelitian-penelitian telah dilakukan dalam upaya peningkatkan kinerja algoritma ELM. Dari penelitian yang telah dilakukan itu, ada 2 pendekatan yang digunakan yaitu pendekatan *feature mapping* dan inisialisasi pembobotan input. Penelitian yang berfokus pada *feature mapping* diantaranya adalah K-ELM(Guang-Bin Huang, Hongming Zhou, Xiaojian Ding, & Rui Zhang, 2012), PCA-ELM(Castaño, Fernández-Navarro, & Hervás-Martínez, 2013), PL-ELM(Henríquez & Ruz, 2017)ELM-AE(Kasun, Zhou, Huang, & Vong, 2013), dan GELM-AE(Sun et al. 2017). Sedangkan penelitian sebelumnya yang menggunakan pendekatan pembobotan input adalah: IPSO-ELM(Han, Yao, and Ling 2013), ROLS-ELM(N. Wang, Er, and Han 2014), LDS-ELM(Cervellera and Macciò 2016), dan ELM-RO(W. Wang and Liu 2017)

Dalam makalah ini penyelesaian hanya berfokus pada pendekatan pembobotan input. Dan dari penelitian yang sejauh ini berhasil diketahui sebagai metode *state-of-the-arts* terbaik dijelaskan dibawah ini.

Penelitian ELM-Auto Encoder(ELM-AE)(Kasun et al., 2013) menggunakan *autoencoder* untuk *feature mapping* di ELM. Metode ini merupakan metode terbaik berdasarkan review(Hussain et al., 2017) lebih baik dari beberapa metode lain. Akan tetapi, penggunaan *kernel* menyebabkan komputasi yang berat sehingga memerlukan waktu yang lama.

Kemudian, Wang melakukan penelitian ELM-*Random Orthogonal* (RO)(W. Wang & Liu, 2017). ELM-RO terbukti memiliki kinerja yang lebih baik daripada standar ELM dan metode random lainnya. Akan tetapi, penggunaan teknik ini masih menggunakan metode *random* yang masih kurang efektif karena tergantung pada parameter W.

Permasalahan yang ada pada penelitian-penelitian sebelumnya di atas, perlu diselesaikan dengan memberikan pembobotan W pada input layer dengan cara lain sehingga diharapkan dapat memperoleh kinerja yang lebih baik. *Restricted Boltman Machine* (RBM)dapat mempelajari dengan baik terhadap probabilitas dari distribusi data(Tissera & McDonnell, 2016). Untuk itulah RBM dapat dicoba untuk diterapkan sebagai algoritma pembobotan pada *input* *layer* di ELM.

# ALGORITMA

**Algoritma ELM (Huang, 2004)**

1 Function elm\_training (X, Y, k, W, b):

**Input** : A training set {X, Y};

The number of hidden neurons k;

The input weights W and the bias b;

2 if W and b are empty then

3 Initialize W and b according to U (−1, 1); #uniform random

end

4 Compute the feature map H from equation 2;

5 **Compute** β from equation 3;

6 **Return**: W, H and β

Legends:

X contoh data

Y label data

k jumlah *neuron*

W bobot *input*

U *Uniform random*

b *bias*

H *matrix penrose*

β bobot *output*

**Algoritma untuk input layer**

* **Algoritma Random**

Input:

Number of neurons n, number of attributes c

Step 1: Make a random matrix with size n x c

Step 2: Calculate output step 1 with \* 2 and substract with 1

Return random matrix on step 2

* **Algoritma Random Orthogonal (Wang & Liu, 2017)**

Input:

Input matrix X n × N consisted of N n -dimensional input vectors;

Objective dimension L (define n ̃ = min { L, n } );

Output:

The random orthogonal projection matrix W L × n .

1: Generate a random matrix A L × n ̃ by MC method.

2: Orthogonalize A by column based on Gram–Schmidt orthogonalization

method, and then obtain A orth .

3:

(1) If L < n

Compute W pca , whose rows are consisted of the first L loading vectors

corresponding to the first L principal components of input matrix X , and let

W L ×n = A orth W pca

( 2) Else

Let W L ×n = A orth .

Return: random orthogonal projection matrix W L × n .

# PEMBOBOTAN

**Algoritma RBM**

1 Function rbm\_training (X, k, η, ρ, α, bs, it):

Input : The training data X with N samples;

The number of hidden neurons k;

The parameters η, ρ and α;

The batch size bs;

The maximum number of iterations it

Sample θ from a normal distribution with μ = 0.1 and σ = 1;

3 Split X in batches X b with bs samples;

4 Initialize i = j = 0;

5 while j < it or θ converged do

while j < it or θ converged do

6 for each batch X b do

7 while i < bs do

8 v 0 = X ib Sample d 0 from equation 6;

9 Sample v 1 from equation 7;

10 Sample d 1 from equation 6;

11 Update θ from equations 8, 9 and 10;

12 i = i +1

end

end  
 j = j + 1

end

13 return θ

Legends:

X contoh data

k jumlah neuron

η learning rate

ρ weight decay

α momentum

bs jumlah batch

it jumlah iterasi

θ

**Algoritma RBM-ELM**

1 Function rbm\_elm\_training ({X t , Y t }, k, η, ρ, α):

Input : The training set {X t , Y t } with N samples;

The number of neurons k;

The parameters η, ρ and α;

2 From Algorithm 2 call rbm\_training (X t , k, η, ρ, α) to get W r and c;

3 From Algorithm 1 call elm\_training (X t , Y t , k, W r , c) to get H and β;

4 Return: W, H and β;

**Formula Min Max Scaler**

*(x**i*–*m**i**n*(*x*)) / (max(x) – min(x))

# VALIDASI

# KODE

# DATASETS

Huang, 2004

Wang & Liu, 2017

Pacheco, 2017

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nama | # Contoh | # Atribut | # Label | Permutasi |
| Credit Australia | 690 | 14 | 2 | Ya |
| Diabetic | 1151 | 19 | 2 | Ya |
| **DNA** | **3186** | **180** | **3** | **Tidak** |
| Gisette | 7000 | 5000 | 3 | Tidak |
| Isolet | 7797 | 617 | 26 | Tidak |
| Madelon | 2600 | 500 | 2 | Ya |
| MNIST | 70000 | 784 | 10 | Tidak |
| Spam | 4601 | 57 | 2 | Tidak |
| Urban land cover | 675 | 147 | 9 | Tidak |

# PARAMETERS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |