



Published in Image Processing On Line on YYYY-MM-DD.
 Submitted on YYYY-MM-DD, accepted on YYYY-MM-DD.
 ISSN 2105-1232 © YYYY IPOL & the authors CC-BY-NC-SA
 This article is available online with supplementary materials,
 software, datasets and online demo at
<https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000527>

Double Descent in the case of Multilayer Perceptron

Emett Haddad

ENS Paris-Saclay, 91190, Gif-Sur-Yvette, France
emett.haddad@ens-paris-saclay.fr

PREPRINT June 25, 2024

Abstract

We consider $y^* : x \in \mathcal{X} = \mathbb{R}^D \rightarrow \mathbb{E}_{(x,y) \sim \mathcal{P}}(y|x) \in \mathcal{Y} = \mathbb{R}$, with x and y random variable with joint distribution $(x, y) \hookrightarrow \mathcal{P}$. We have a **dataset** $\mathcal{D} := \{(x_n, y_n) \in \mathcal{X} \times \mathcal{Y}\}_{n=1}^N$ where $y_n := y(x_n)$, and $y = y^* + \epsilon$ where ϵ represents the noise as $\mathbb{E}(\epsilon|x) = 0$, et $\mathbb{V}(\epsilon) = \sigma^2$. We want to create an estimator $\hat{y}_{\mathcal{D}} : \mathcal{X} \rightarrow \mathcal{Y}$ which minimises the empirical risk in order to minimise the true risk $\mathcal{R}(\hat{y}) = \mathbb{E}_{(x,y) \sim \mathcal{P}}((y - \hat{y}(x))^2)$. Working in the context of [1], but with $\epsilon = 0$.

Source Code

The reviewed source code and documentation for this algorithm are available from [the web page of this article](#)¹. Usage instruction are included in the `README.txt` file of the archive.

1 Model

- $X = [x_1, \dots, x_N]^T \in \mathcal{M}_{N,D}(\mathbb{R})$, $Y = [y_1, \dots, y_N]^T$
- $\hat{y}_{\beta}(x) = W_L \circ \sigma \circ \dots \sigma \circ W_1(x)$, $W_l(x_l) = A_l x_l + b_l$
- $\sigma(x) = \max(0, x)$ ReLU function.
- Constant-step gradient descent: $\hat{\beta}_{t+1} = \hat{\beta}_t - \alpha \nabla \hat{\mathcal{R}}(\hat{y}_{\hat{\beta}_t})$, $\alpha \in \mathbb{R}^+$
- Empirical Risk function $\hat{\mathcal{R}}(y) = \frac{1}{2} \|Y - \hat{y}_{\beta}(X)\|_2^2$

¹<https://ipolcore.ipol.im/demo/clientApp/demo.html?id=77777000527>

2 Algorithms

Algorithm 1: Dataset initialisation

```

1 function dataset_initialisation( $f, C, M, \text{ratio\_data}$ )
  Input  $y, C, M, \text{ratio\_data}$ :  $y : \mathbb{R}^D \rightarrow \mathbb{R}, C = [[a_d, b_d], 1 \leq d \leq D], \text{ratio\_data} \in [0, 1]$ 
2    $U = \mathcal{U}([0, 1]^{(M, D)})$ 
3    $X = \text{Diag}(a_d) + U \text{Diag}(b_d - a_d)$  #  $X \hookrightarrow \mathcal{U}(C^M)$ 
4    $Y = y(X)$ 
5    $X_{\text{train}}, X_{\text{test}}, Y_{\text{train}}, Y_{\text{test}} = \text{test\_split}(X, Y, \text{ratio\_data})$ 
6   return  $X_{\text{train}}, X_{\text{test}}, Y_{\text{train}}, Y_{\text{test}}$ 

```

Algorithm 2: MLP Gradient Descent

```

1 function MLP_Gradient_Descent( $P_{\min}, P_{\max}, \text{Epochs}, \alpha, \text{Data}, \text{Data\_global}$ )
2    $X_{\text{train}}, X_{\text{test}}, Y_{\text{train}}, Y_{\text{test}} = \text{Data}$ 
3    $X_{\text{global}}, Y_{\text{global}} = \text{Data\_global}$ 
4   for  $P_{\min} \leq p \leq P_{\max}$  do
5      $MLP = \text{Affine}(1, P) \circ \sigma \circ \text{Affine}(P, D)$  # Creation of the MLP structure
6      $MLP.\text{fit}(X_{\text{train}}, Y_{\text{train}}, \text{Epochs}, \alpha)$ 
7      $\text{Train\_error}[p] = \log(1 + \text{MSE}(Y_{\text{train}}, MLP(X_{\text{train}})))$ 
8      $\text{Test\_error}[p] = \log(1 + \text{MSE}(Y_{\text{test}}, MLP(X_{\text{test}})))$ 
9      $\text{Global\_error}[p] = \log(1 + \text{MSE}(Y_{\text{test}}, MLP(X_{\text{global}})))$ 
10   $\hat{y} = MLP(P_{\max})$ 
11  return  $\text{Train\_error}, \text{Test\_error}, \text{Global\_error}, \hat{y}$ 

```

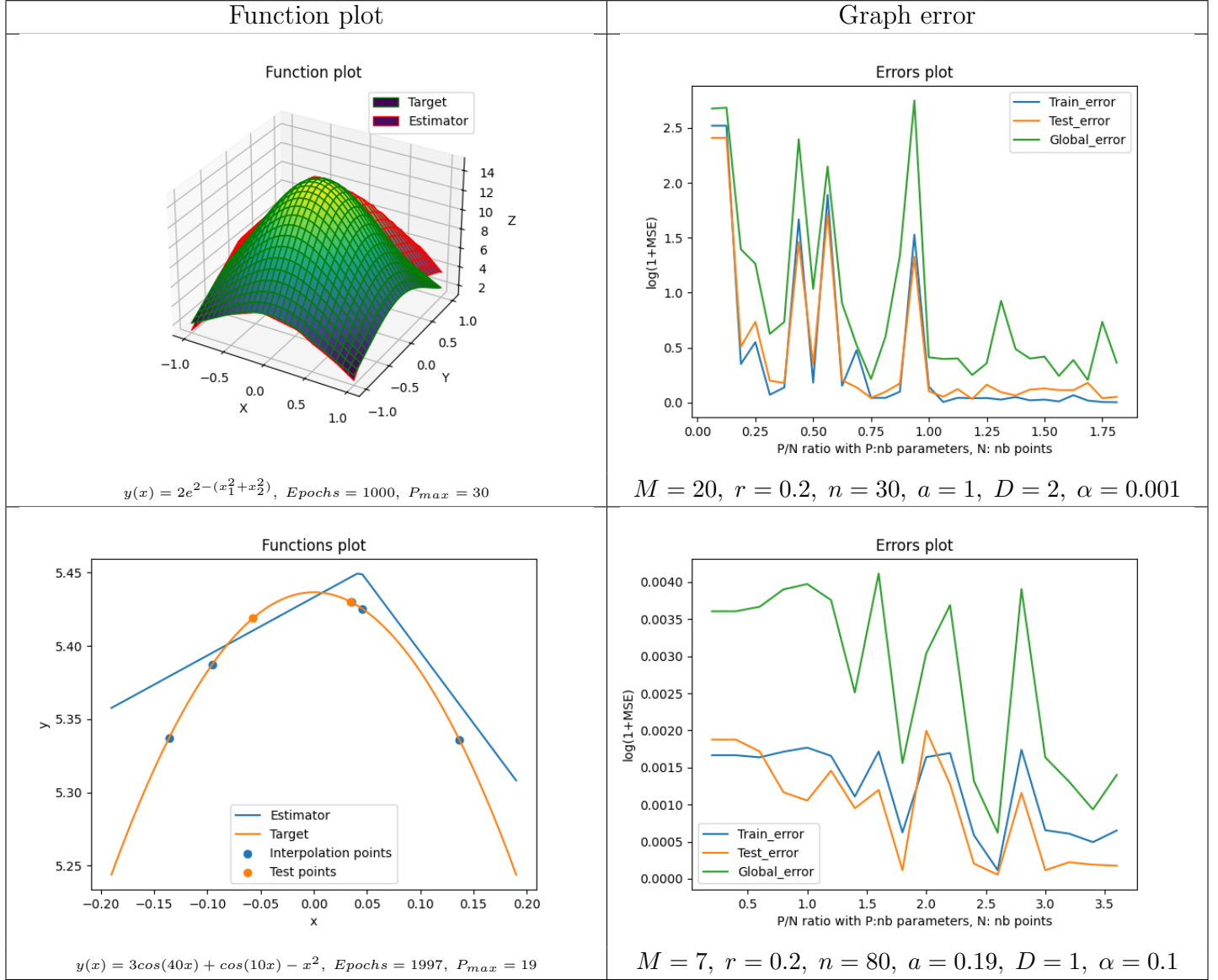
Algorithm 3: MLP Gradient Descent FIT

```

1 function MLP.fit( $X_{\text{train}}, Y_{\text{train}}, \text{Epochs}, \alpha$ )
2   for  $1 \leq \text{epoch} \leq \text{Epochs}$  do
3     for  $(x_{\text{train}}, y_{\text{train}}) \in \mathcal{D}$  do
4        $\text{score\_gradient} = MLP(x_{\text{train}}) - y_{\text{train}}$  # gradient of MSE
5        $\text{backpropagation}(\text{score\_gradient})$ 
6        $\text{update}(\alpha)$  # update weights

```

3 Examples (random seed : 23334)



References

- [1] EMMETT HADDAD, *Github Double Descent*. <https://github.com/EmettGabrielH/Double-descent---Emett-Haddad>, 2024. [Online].