A Modified Levenberg Marquardt Algorithm for Simultaneous Learning of Multiple Datasets

Mehmet Önder Efe, *Senior Member, IEEE*, Burak Kürkçü, Coşku Kasnakoğlu, *Member, IEEE*, Zaharuddin Mohamed, and Zhijie Liu, *Member, IEEE*

***Abstract*—Levenberg-Marquardt (LM) algorithm is a powerful approach to optimize the parameters of a neural network (NN). Given a training dataset, the algorithm synthesizes the best path toward the optimum. This brief demonstrates the use of LM optimization algorithm when there are more than one dataset and on/off type switching of NN parameters is allowed. For each dataset a pre-selected set of parameters are allowed for modification and the proposed scheme reformulates the Jacobian under the switching mechanism. The results show that a NN can store information available in different datasets by a simple modification to the original LM algorithm, which is the novelty introduced in this brief. The results are verified on a regression problem.**

***Index Terms*—Levenberg-Marquardt algorithm, multiple dataset learning, masked neural networks.**

NOMENCLATURE

t	Discrete time index
$\mathbf{y}(t)$ ($\mathbf{y}_i(t)$)	Output vector of the neural network at time t (i^{th} entry of $\mathbf{y}$ at time t)
$\tau(t)$ ($\tau_i(t)$)	Target output vector at time t (i^{th} entry of τ at time t)
$\mathbf{u}(t)$ ($\mathbf{u}_i(t)$)	Input vector of the neural network at time t (i^{th} entry of $\mathbf{u}$ at time t)
$\mathbf{w}(t)$ ($\mathbf{w}_i(t)$)	Parameter (weight/bias) vector of neural network at time t (i^{th} entry of $\mathbf{w}$ at time t)
$\mathbf{I}$	Identity matrix
$\mathbf{J}(t)$	Jacobian of the classical LM algorithm at time t
$\mathbf{J}_F(t)$	Jacobian of the proposed LM algorithm at time t
$\mathbf{E}(t)$	Error vector of the classical LM algorithm at time t
$\mathbf{E}_F(t)$	Error vector of the proposed LM algorithm at time t
μ	Step size parameter
$\mathbf{m}^d$ ($\mathbf{m}_i^d$)	Mask vector for the d^{th} dataset (i^{th} entry of $\mathbf{m}^d$)
$\mathbf{e}_i^p(t)$ i^{th}	entry of the error vector for the p^{th} input/output pair at time t
D	Number of different datasets
P_d	Number of input/output pairs in dataset d
P	Number of input/output pairs for classical single dataset learning setting
N	Total number of adjustable parameters in the neural network
L	Loss function for a single dataset
L_F	Combined loss functions for multiple dataset setting
K	Number of NN outputs

Manuscript received 20 October 2023; accepted 15 November 2023. Date of publication 28 November 2023; date of current version 27 March 2024. This brief was recommended by Associate Editor X. Zhang. *(Corresponding author: Mehmet Önder Efe.)*

Mehmet Önder Efe is with the Department of Computer Engineering, Hacettepe University, 06800 Ankara, Turkey (e-mail: onderefe@gmail.com).

Burak Kürkçü is with the Department of Mechanical Engineering, University of California at Berkeley, Berkeley, CA 94720 USA (e-mail: bkurkcu@berkeley.edu).

Coşku Kasnakoğlu is with the Electrical and Electronics Engineering Department, TOBB University of Economics and Technology, 06510 Ankara, Turkey (e-mail: kasnakoglu@etu.edu.tr).

Zaharuddin Mohamed is with the School of Electrical Engineering, Universiti Teknologi Malaysia, Johor Bahru 81310, Malaysia (e-mail: zahar@fke.utm.my).

Zhijie Liu is with the School of Intelligence Science and Technology, University of Science and Technology Beijing, Beijing 100083, China (e-mail: liuzhijie2012@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2023.3335140.

Digital Object Identifier 10.1109/TCSII.2023.3335140

I. INTRODUCTION

ARTIFICIAL neural networks have been studied many times in the past. Given an input output dataset, a NN structure can realize a map accompanied with an error bound. The works of Funahashi [1], Cybenko [2] and Hornik [3] have set two facts: 1) NNs are universal approximators and 2) A single hidden layer NN can realize a given input/output map with a prescribed error bound if the number of neurons in the hidden layer are large enough. The work of Hagan and Menhaj [4] was a pioneer in casting the LM optimization scheme for NNs and since then the algorithm has been successfully implemented in various applications [5], [6], [7] and become a standard tool in several commercial NN optimization software. The works devoted to speeding up the LM algorithm are worthy of consideration as each training cycle in LM scheme requires matrix inversion and utilizing matrix inversion lemma for reducing the complexity may be a remedy [7], [8], [9].

The idea that motivated us to study this problem has its roots in the functionality of biological brain. There could be regions that contain multifunctional biological structures that are triggered chemically via the secretion system. Analogously, we build a NN structure and define a mask that runs over the NN and ask for the realization of multiple tasks for multiple masks.

1549-7747 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: ULAKBIM UASL - Hacettepe Universitesi. Downloaded on April 28,2024 at 09:28:25 UTC from IEEE Xplore. Restrictions apply.

To demonstrate the idea, we adopt single hidden layer NN structures to study the problem of simultaneous learning of multiple datasets. The conventional feedforward NN structures are suitable for gradient based tuning schemes, i.e., differentiable neuronal activation functions are assumed throughout this brief.

The problem of learning multiple datasets have been considered several times in the past. In [10], a NN is trained for a given dataset, the network is then pruned and the process for next dataset considers the previously fixed weights together with the adjustable weight subset, which was the pruned subset of the previous stage. The algorithm in [10] considers the datasets one by one and tries to teach the new dataset to a gradually contracting neural space (free neurons or weights) aided also by the previously fixed weight values. The results focus on deep neural structures with several well known datasets. This is a good example of storing different datasets within different subnetworks of a NN structure, which was studied also in [11]. The work in [12] focuses on two separate online classification settings, which is based on the availability of a number of orthogonal vectors that makes it possible to store the parameter vector as a sum. When the appropriate mask multiplies the weight vector, the weight set associated to the chosen mask, which is called context key, is obtained. The method requires finding orthogonal terms to store the weight and bias set. The main difficulty here is to find the orthogonal terms when there are many different datasets. Further, since the considered neural structure is a deep NN structure containing thousands of neurons and many layers, one cannot make sure that the structure is the smallest possible one that meets the desired learning goals. This issue can be studied by the modification proposed in the current work. In [13], a large NN structure is initialized and without any modification, for a given task, the best weight/bias subset is sought. As long as the NN is large in structure, the diversity that spans the functional space of the NN can be ensured and a best matching mask vector could be inferred for a given arbitrary dataset. The work in [13] solves the reverse problem and defines four continual learning scenarios and describes how a best matching mask could be extracted if a dataset is given. In [14], a deep NN structure is initialized and every weight is multiplied by 1 to keep, 0 to hide and -1 to invert. The optimization problem is to find the best multiplier vector to obtain a good performance. The authors reported that the approach resulted in up to 99% of pruning rate. The cited body of literature focuses largely on the deep NN structures, where the redundancy is excessive and necessity to each one of the adjustable parameters is questionable. Further, the works cited assume the lottery ticket hypothesis proposed by Frankle and Carbin [15], which states that there exists a subnetwork that performs the desired mapping with an acceptable performance and the problem is to identify its elements within the deep NN structure.

A natural problem arising in the cited volume of literature is the catastrophic forgetting, which causes forgetting of the learned information for one dataset while the tuning is carried out for another [16], [17]. The current paper remedies the problem of catastrophic forgetting in an efficient way as all input output pairs are treated simultaneously in the Jacobian matrix.

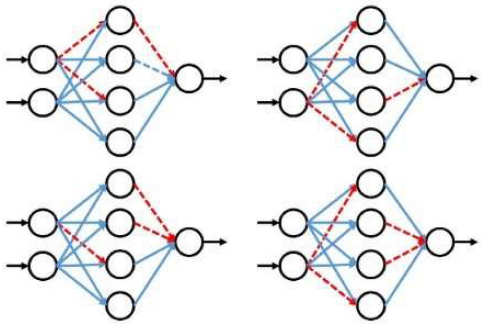

Fig. 1. Four datasets and preselected four subnetworks. Inactive parameters are shown dashed and the weight values are frozen throughout the NN.

The contribution of this brief is to modify the original LM tuning algorithm in such a way that gradient information can be effectively used in learning of multiple and possibly dissimilar datasets under the conditions that are valid for ordinary NN training. This contribution advances the subject area towards effective multifunctional and masked NN applications that may open horizons for reinterpreting federated learning, explainable artificial intelligence and resource sharing hardware applications.

This brief is organized as follows: Section II describes the modified LM algorithm, Section III discusses an exemplar learning problem and the last section is devoted to the concluding remarks.

II. Modified Levenberg-Marquardt Algorithm

In the implementation of the LM optimization scheme for such problem, a NN structure is chosen and each dataset is linked with its own mask vector, the components of which enable or disable a particular weight/bias parameter. In Fig. 1, a NN structure specialized for four different datasets is shown. The weights shown by dashed lines are disabled in the corresponding configuration.

According to the figure, we deduce two immediate facts for the masking mechanism.

1. Every single parameter (weight or bias) must be active at least for one dataset.
2. The masking scheme must allow information flow from each input to each output.

Assume we have D different datasets and for each of them we adopt a randomly chosen mask vector that performs an on/off operation on the entries of the NN parameter vector, $\mathbf{w}$. Depending on the content of the mask vector $\mathbf{m}$, some weights are activated for all datasets, some are peculiar to particular subgroups of the datasets. This way creates a NN structure that has a set of real parameter set ($\mathbf{w}$) composed of the weights and biases, and an associated mask vector for each dataset ($\mathbf{m}^d$, d=1, 2,..., D).

Considering there is only one dataset ($D = 1$) and all weights and biases are active, i.e., all entries of the mask vector are 1, we have the following LM based parameter update law:

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \left(\mu \mathbf{I}_{N\times N} + \mathbf{J}(t)^T \mathbf{J}(t)\right)^{-1} \mathbf{J}(t)^T \mathbf{E}(t) \quad (1)$$

Authorized licensed use limited to: ULAKBIM UASL - Hacettepe Universitesi. Downloaded on April 28,2024 at 09:28:25 UTC from IEEE Xplore. Restrictions apply.

where, μ is the step size parameter and the Jacobian ($\mathbf{J}(t)$) and the error vector ($\mathbf{E}(t)$) are given in (2) and (3), respectively. It is well known that for large values of μ, LM algorithm displays the characteristics of Error Backpropagation (EBP) algorithm with step size $1/\mu$ and for small μ, the algorithm behaves like Gauss-Newton (GN) algorithm and becomes vulnerable to matrix inversion problems. LM algorithm is therefore a smooth transition in between EBP and GN exhibiting the positive qualities of both to some extent, [8]. The loop structure of the approach without details of μ adaptation and early stopping is given in Algorithm 1, where the stopping criterion is based on a prescribed performance level.

$$\mathbf{J}(t)=\begin{bmatrix}\begin{bmatrix}\frac{\partial \mathbf{e}_1^1}{\partial \mathbf{w}_1} & \frac{\partial \mathbf{e}_1^1}{\partial \mathbf{w}_2} & \cdots & \frac{\partial \mathbf{e}_1^1}{\partial \mathbf{w}_N}\\ \frac{\partial \mathbf{e}_2^1}{\partial \mathbf{w}_1} & \frac{\partial \mathbf{e}_2^1}{\partial \mathbf{w}_2} & \cdots & \frac{\partial \mathbf{e}_2^1}{\partial \mathbf{w}_N}\\ & \vdots & & \\ \frac{\partial \mathbf{e}_K^1}{\partial \mathbf{w}_1} & \frac{\partial \mathbf{e}_K^1}{\partial \mathbf{w}_2} & \cdots & \frac{\partial \mathbf{e}_K^1}{\partial \mathbf{w}_N}\end{bmatrix}_{K\times N}\\ \begin{bmatrix}\frac{\partial \mathbf{e}_1^2}{\partial \mathbf{w}_1} & \frac{\partial \mathbf{e}_1^2}{\partial \mathbf{w}_2} & \cdots & \frac{\partial \mathbf{e}_1^2}{\partial \mathbf{w}_N}\\ \frac{\partial \mathbf{e}_2^2}{\partial \mathbf{w}_1} & \frac{\partial \mathbf{e}_2^2}{\partial \mathbf{w}_2} & \cdots & \frac{\partial \mathbf{e}_2^2}{\partial \mathbf{w}_N}\\ & \vdots & & \\ \frac{\partial \mathbf{e}_K^2}{\partial \mathbf{w}_1} & \frac{\partial \mathbf{e}_K^2}{\partial \mathbf{w}_2} & \cdots & \frac{\partial \mathbf{e}_K^2}{\partial \mathbf{w}_N}\end{bmatrix}_{K\times N}\\ \vdots\\ \begin{bmatrix}\frac{\partial \mathbf{e}_1^P}{\partial \mathbf{w}_1} & \frac{\partial \mathbf{e}_1^P}{\partial \mathbf{w}_2} & \cdots & \frac{\partial \mathbf{e}_1^P}{\partial \mathbf{w}_N}\\ \frac{\partial \mathbf{e}_2^P}{\partial \mathbf{w}_1} & \frac{\partial \mathbf{e}_2^P}{\partial \mathbf{w}_2} & \cdots & \frac{\partial \mathbf{e}_2^P}{\partial \mathbf{w}_N}\\ & \vdots & & \\ \frac{\partial \mathbf{e}_K^P}{\partial \mathbf{w}_1} & \frac{\partial \mathbf{e}_K^P}{\partial \mathbf{w}_2} & \cdots & \frac{\partial \mathbf{e}_K^P}{\partial \mathbf{w}_N}\end{bmatrix}_{K\times N}\end{bmatrix}_{\mathrm{PK}\times N} \tag{2}$$

$$\mathbf{E}(t)=\begin{bmatrix}\begin{bmatrix}\mathbf{e}_1^1(t)\\ \mathbf{e}_2^1(t)\\ \vdots\\ \mathbf{e}_K^1(t)\end{bmatrix}_{K\times 1}\\ \begin{bmatrix}\mathbf{e}_1^2(t)\\ \mathbf{e}_2^2(t)\\ \vdots\\ \mathbf{e}_K^2(t)\end{bmatrix}_{K\times 1}\\ \vdots\\ \begin{bmatrix}\mathbf{e}_1^P(t)\\ \mathbf{e}_2^P(t)\\ \vdots\\ \mathbf{e}_K^P(t)\end{bmatrix}_{K\times 1}\end{bmatrix}_{\mathrm{PK}\times 1} \tag{3}$$

$$L(t)=\mathbf{E}(t)^T\mathbf{E}(t) \tag{4}$$

Now consider a masked NN structure such that the adjustable parameter vector $\mathbf{w}$ is associated with a separate mask vector ($\mathbf{m}^d$) for each dataset d=1, 2, …, D. Elements of $\mathbf{m}^d$ are composed of zeros and ones, activating and deactivating the corresponding entry in $\mathbf{w}$. Having this picture in the front, the d^{th} component of Jacobian has its new form as in (5). The overall Jacobian ($\mathbf{J}_F$) of the multiple dataset learning problem is given in (6) and the tuning law utilizing $\mathbf{J}_F$ is given in (7). Apparently, the structure of the tuning law is the same as that in (1) and the concatenated error vector used in (7) is given in (8). The update scheme in (7) optimizes the cost in (9) and for each individual mask, the NN approximates the associated dataset. The accuracy that can be accomplished by the NN depends on the size of the masked subnetwork. Arguments of this discussion are the same as in ordinary single dataset NN training schemes.

Algorithm 1 Levenberg-Marquardt Algorithm

```
t=1;
L=Inf;
 μ=1;
PerformanceLevel = ChooseLevel;
SetNeuralNetwork
while L > PerformanceLevel
   E = [];
   J = [];
   % Pattern loop
   for p = 1 to P
      y = ForwardPassPattern#p
      % Append sample error τ – y to E
      E= [E ; τ – y]
      % Loop over each output of the NN
      for k = 1 to K
        Row_of_J = [];
        % Parameter (weight/bias) index
        for i = 1 to N
          Compute ∂e_k^p/∂w_i
          Row_of_J = [Row_of_J ∂e_k^p/∂w_i]
        end
        J = [J ; Row_of_J]
        % One row of J(t) is ready and appended
      end
      % One K×N sub-block of J(t) is ready
   end
   % J(t) is ready, update now
   w(t+1) = w(t) – (μ I+J(t)^T J(t))^-1 J(t)^T E(t)
   DistributeNewValuesToTheirPositions
   L = E^T E
   DisplayCost L
   Update_μ_IfNecessary
   RunEarlyStoppingRoutineIfNecessary
   t = t+1;
end
```

$$\mathbf{J}_d=\begin{bmatrix}\begin{bmatrix}\frac{\partial \mathbf{e}_1^1}{\partial \mathbf{w}_1}\mathbf{m}_1^d & \frac{\partial \mathbf{e}_1^1}{\partial \mathbf{w}_2}\mathbf{m}_2^d & \cdots & \frac{\partial \mathbf{e}_1^1}{\partial \mathbf{w}_N}\mathbf{m}_N^d\\ \frac{\partial \mathbf{e}_2^1}{\partial \mathbf{w}_1}\mathbf{m}_1^d & \frac{\partial \mathbf{e}_2^1}{\partial \mathbf{w}_2}\mathbf{m}_2^d & \cdots & \frac{\partial \mathbf{e}_2^1}{\partial \mathbf{w}_N}\mathbf{m}_N^d\\ & \vdots & & \\ \frac{\partial \mathbf{e}_K^1}{\partial \mathbf{w}_1}\mathbf{m}_1^d & \frac{\partial \mathbf{e}_K^1}{\partial \mathbf{w}_2}\mathbf{m}_2^d & \cdots & \frac{\partial \mathbf{e}_K^1}{\partial \mathbf{w}_N}\mathbf{m}_N^d\end{bmatrix}_{K\times N}\\ \begin{bmatrix}\frac{\partial \mathbf{e}_1^2}{\partial \mathbf{w}_1}\mathbf{m}_1^d & \frac{\partial \mathbf{e}_1^2}{\partial \mathbf{w}_2}\mathbf{m}_2^d & \cdots & \frac{\partial \mathbf{e}_1^2}{\partial \mathbf{w}_N}\mathbf{m}_N^d\\ \frac{\partial \mathbf{e}_2^2}{\partial \mathbf{w}_1}\mathbf{m}_1^d & \frac{\partial \mathbf{e}_2^2}{\partial \mathbf{w}_2}\mathbf{m}_2^d & \cdots & \frac{\partial \mathbf{e}_2^2}{\partial \mathbf{w}_N}\mathbf{m}_N^d\\ & \vdots & & \\ \frac{\partial \mathbf{e}_K^2}{\partial \mathbf{w}_1}\mathbf{m}_1^d & \frac{\partial \mathbf{e}_K^2}{\partial \mathbf{w}_2}\mathbf{m}_2^d & \cdots & \frac{\partial \mathbf{e}_K^2}{\partial \mathbf{w}_N}\mathbf{m}_N^d\end{bmatrix}_{K\times N}\\ \vdots\\ \begin{bmatrix}\frac{\partial \mathbf{e}_1^{P_d}}{\partial \mathbf{w}_1}\mathbf{m}_1^d & \frac{\partial \mathbf{e}_1^{P_d}}{\partial \mathbf{w}_2}\mathbf{m}_2^d & \cdots & \frac{\partial \mathbf{e}_1^{P_d}}{\partial \mathbf{w}_N}\mathbf{m}_N^d\\ \frac{\partial \mathbf{e}_2^{P_d}}{\partial \mathbf{w}_1}\mathbf{m}_1^d & \frac{\partial \mathbf{e}_2^{P_d}}{\partial \mathbf{w}_2}\mathbf{m}_2^d & \cdots & \frac{\partial \mathbf{e}_2^{P_d}}{\partial \mathbf{w}_N}\mathbf{m}_N^d\\ & \vdots & & \\ \frac{\partial \mathbf{e}_K^{P_d}}{\partial \mathbf{w}_1}\mathbf{m}_1^d & \frac{\partial \mathbf{e}_K^{P_d}}{\partial \mathbf{w}_2}\mathbf{m}_2^d & \cdots & \frac{\partial \mathbf{e}_K^{P_d}}{\partial \mathbf{w}_N}\mathbf{m}_N^d\end{bmatrix}_{K\times N}\end{bmatrix}_{P_dK\times N} \tag{5}$$

Authorized licensed use limited to: ULAKBIM UASL - Hacettepe Universitesi. Downloaded on April 28,2024 at 09:28:25 UTC from IEEE Xplore. Restrictions apply.

where

$$\mathbf{J}_F = \begin{bmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \vdots \\ \mathbf{J}_D \end{bmatrix}_{\left(K\sum_{d=1}^{D} P_d\right)\times N} \tag{6}$$

and the modified LM update law can be formulated as below.

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \left(\mu \mathbf{I}_{N\times N} + \mathbf{J}_F(t)^T \mathbf{J}_F(t)\right)^{-1} \mathbf{J}_F(t)^T \mathbf{E}_F(t) \tag{7}$$

where

$$\mathbf{E}_F(t) = \begin{bmatrix} \mathbf{E}^1(t) \\ \mathbf{E}^2(t) \\ \vdots \\ \mathbf{E}^D(t) \end{bmatrix} \tag{8}$$

$$L_F(t) = \sum_{d=1}^{D} \left(\mathbf{E}^d(t)\right)^T \left(\mathbf{E}^d(t)\right) \tag{9}$$

The procedure for the proposed scheme is given in Algorithm 2. The difference between (7) and (1) is the fact that (1) is peculiar to single dataset case and (7) allows teaching multiple datasets each accompanied by a mask vector creating a specific subnetwork.

The modification proposed here is important in the sense that the algorithm is guided towards identifying the maximal overlaps in the neural functional structure that is distributed over the neural network connectivity. In a single dataset case with conventional scheme (1), the optimization algorithm finds a solution among many other possibilities. Imposing masks and obtaining subnetworks make the available optimization problem much more challenging as the solution space is constrained by the training datasets, which are more than one and which are possibly dissimilar. This naturally requires finding local similarities in between the datasets and this is automatically organized utilizing the LM algorithm and the formulation given in (7).

Algorithm 2 Modified Levenberg-Marquardt Algorithm

```
t=1;
L=Inf;
μ=1;
PerformanceLevel = ChooseLevel;
SetNeuralNetwork
SetMaskVectors
while L > PerformanceLevel
  E_F = [];
  J_F = [];
  % Dataset loop
  for d = 1 to D
    % Pattern loop
    for p = 1 to P_d
      y = ForwardPassPattern#p
      % Append sample error τ − y to E
      E_F= [E_F; τ − y]
      % Loop over each output of the NN
      for k = 1 to K
        Row_of_Jd = [];
        % Parameter (weight/bias) index
        for i = 1 to N
          Compute m_i^d × ∂e_k^p/∂w_i
          Row_of_Jd = [Row_of_Jd m_i^d × ∂e_k^p/∂w_i]
        end
        J_d = [J_d; Row_of_Jd]
        % One row of J(t) is ready and appended
      end
      % One K×N sub-block of J(t) is ready
    end
    % J_d is ready. Append to Jacobian
    J_F = [J_F;J_d]
  end
  % Jacobian J_F(t) is ready, update now
  w(t+1) = w(t) − (μ I+J_F(t)^T J_F(t))^-1 J_F(t)^T E_F(t)
  DistributeNewValuesToTheirPositions
  L_F = E_F^T E_F
  DisplayCost L_F
  Update_μ_IfNecessary
  RunEarlyStoppingRoutineIfNecessary
  t = t+1;
end
```

III. Simulations

In the simulations, we study a regression problem and we generate four datasets ($D = 4$). The set of functions generating the datasets are orthonormal over $(x,y) \in [-1, +1]\times[-1, +1]$. This is deliberate as we would like to test the NN's capability of handling the dissimilarities. The NN has 2-100-1 structure, the input vector is $[x\ y]^T$, the hidden neurons have hyperbolic tangent type nonlinearity and the output neuron is a linear one.

The four functions have been sampled linearly, i.e., 21 data points along each axis create a total of 441 grid points. We used the same input data grid for sampling the four functions, which generate different outputs requiring appropriate switching. Since our goal is to demonstrate the use of modified scheme, we have not used a validation set and stopped the training after 2000 iterations. We define the mean squared error (MSE) as in (10).

$$MSE(t) = L_F(t) / \sum_{d=1}^{D} P_d \tag{10}$$

After 2000 iterations of running Algorithm 2, *MSE* decreases to 9.882e-5 and the results shown in Fig. 2 are obtained. According to the figure, the NN is proved capable of learning multiple datasets simultaneously.

Since the training considers all training samples together, the proposed scheme is not vulnerable to the well known catastrophic forgetting phenomenon. A last issue is the initialization of the elements of the mask vector ($\mathbf{m}_i^d$). For this, define a random variable (u) that distributes uniformly between 0 and 1. Based on this, we adopted the following scheme:

$$\mathbf{m}_i^d = \begin{cases} 1 & u \geq 0.3 \\ 0 & u < 0.3 \end{cases}, d = 1, 2, \ldots, D; i = 1, 2, \ldots, N \tag{11}$$

Authorized licensed use limited to: ULAKBIM UASL - Hacettepe Universitesi. Downloaded on April 28,2024 at 09:28:25 UTC from IEEE Xplore. Restrictions apply.

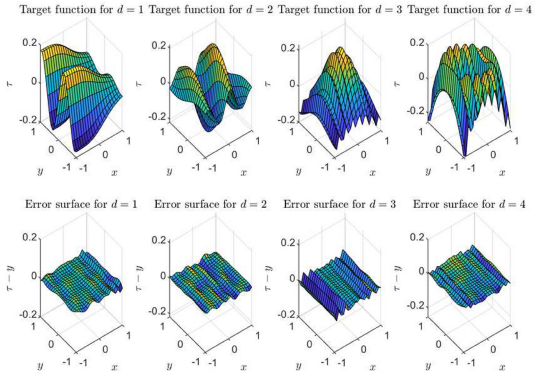

Fig. 2. Top: Target datasets (τ). Bottom: The difference between NN response and the target values ($\tau - y$).

The chosen threshold above is 0.3. One can increase or decrease this threshold to obtain similar results. As the threshold approaches 0, it becomes difficult to learn any of the datasets as the degrees of freedom allocated for each dataset reduces and the subnetwork activates few parameters. As the threshold approaches 1, it becomes impossible to learn multiple datasets as most parameters are kept active for all datasets. Depending on the size of the NN, one should be aware of this and choose a reasonable value that generates a good learning performance.

IV. Conclusion

This brief proposes a modification to the Levenberg Marquardt algorithm. The approach allows learning of multiple datasets simultaneously and catastrophic forgetting is prevented. Each dataset is accompanied by a mask vector that activates a particular parameter set resulting in a subnetwork. The obtained subnetwork realizes the information in the selected dataset. Simulations results prove the solvability of the multiple optimization problems within the Levenberg Marquardt algorithm's setup.

References

[1] K. I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Netw.*, vol. 2, no. 3, pp. 183–192, 1989.
[2] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals Syst.*, vol. 2, no. 4, pp. 303–314, 1989.
[3] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.
[4] M. T. Hagan and M. B. Menhaj, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Netw.*, vol. 5, no. 6, pp. 989–993, Nov. 1994.
[5] C. Lv et al., "Levenberg–Marquardt backpropagation training of multilayer neural networks for state estimation of a safety-critical cyber-physical system," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3436–3446, Aug. 2018.
[6] S. Kimdabi and A. Antoniou, "Design of minimum-phase filters using optimization," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 4, pp. 472–476, Apr. 2017.
[7] B. M. Wilamowski, S. İplikçi, O. Kaynak, and M. O. Efe, "An algorithm for fast convergence in training neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, Washington, DC, USA, 2001, pp. 1778–1782.
[8] J. D. J. Rubio, "Stability analysis of the modified Levenberg-Marquardt algorithm for the artificial neural network training," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3510–3524, Aug. 2021.
[9] S. Ur. Rehman, S. Tu, O. Ur. Rahman, Y. Huang, C. M. S. Magurawalage, and C.-C. Chang, "Optimization of CNN through novel training strategy for visual classification problems," *Entropy*, vol. 20, no. 4, p. 290, 2018.
[10] A. Mallya and S. Lazebnik, "PackNet: Adding multiple tasks to a single network by iterative pruning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2018, pp. 7765–7773.
[11] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: Adapting a single network to multiple tasks by learning to mask weights," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 72–88.
[12] B. Cheung, A. Terekhov, Y. Chen, P. Agrawal, and B. Olshausen "Superposition of many models into one," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2019, pp. 1–10.
[13] M. Wortsman et al., "Supermasks in superposition," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2020, pp. 15173–15184.
[14] N. Koster, O. Grothe, and A. Rettinger, "Signing the Supermask: Keep, hide, invert," 2022, *arXiv:2201.13361*.
[15] J. Frankle and M. Carbin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," 2018, *arXiv:1803.03635*.
[16] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends Cogn. Sci.*, vol. 3, no. 4, pp. 128–135, 1999.
[17] J. Kickpatrick et al., "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci.*, vol. 114, no. 13, pp. 3521–3526, 2017.

Authorized licensed use limited to: ULAKBIM UASL - Hacettepe Universitesi. Downloaded on April 28,2024 at 09:28:25 UTC from IEEE Xplore. Restrictions apply.