Improving Load Forecasting Performance via Sample Reweighting

Chenxi Wang, *Student Member, IEEE,* Yangze Zhou, *Student Member, IEEE,* Qingsong Wen, *Senior Member, IEEE,* Yi Wang, *Member, IEEE*

***Abstract*—Load forecasting techniques have been well developed and almost all typical models are trying to minimize the error over all equally weighted data samples. However, treating all data equally in the training process is unreasonable and even harmful to model generalization. This letter investigates the issue of data inequality which is crucial but is overlooked in the field of load forecasting. To this end, we propose a novel load forecasting method via sample reweighting to tackle the data inequality issue. The proposed method determines the weights of samples using their gradients and influence functions in the process of training. Case studies on a real-world load dataset using linear models, artificial neural networks, and long short term memory in both deterministic and probabilistic situations all reveal that assigning different weights to various samples can improve forecasting accuracy.**

***Index Terms*—Data inequality, Sample reweighting, Influence function, Load forecasting**

I. Introduction

THE thriving of machine learning and massive data collected by sensors and smart meters, have greatly contributed the booming development of load forecasting technology [1]. Current works on load forecasting mainly focus on developing or applying effective feature extraction methods and advanced regression models to enhance forecasting accuracy. People also explore new topics, such as probabilistic forecasting [2], interpretable forecasting [3], robust forecasting [4], multi-energy forecasting [5], etc.

When developing regression models, we usually treat all the data samples fairly and the goal of the model training process is to minimize the total loss of all equally weighted training data samples. However, treating all data equally in the training process may be unreasonable and not conducive to model generalization. For example, summer tends to lead to larger and more volatile loads, which may cause more significant forecasting errors. It implies that the load data collected in such a situation may be more influential and ought to be attached more attention. Thus, data inequality, which means that the degrees of influence of various samples are different, should have been considered in load forecasting. Despite its importance, it has not received sufficient attention. Hence, this letter wants to address the following question: *When training a load forecasting model, can we attach different weights to different samples to enhance forecasting performance?*

Actually, similar problems have been investigated beyond load forecasting but in the general machine learning area. Specifically, to achieve robust classifiers, [6] proposed a meta-learning algorithm that learns to reweight reasonable training samples. [7] developed an unlabeled data samples reweighting method to balance the loss of labeled and unlabeled samples in semi-supervised learning tasks. Nevertheless, these efforts all focus on the image recognition problem, where the data inequality concept has been well accepted and studied. To the best of our knowledge, the thought of reweighting has hardly been explored in the field of load forecasting.

Hence, to deal with data inequality, this letter studies a sample reweighting strategy to improve load forecasting performance. The proposed method assigns different weights to various samples by utilizing the gradients and influence function in the training process. It is noted that our method is model-agnostic and can be applied easily to any gradient descent-based model. This letter makes the following contributions:

1) Study a novel load forecasting method via sample reweighting to tackle the data inequality issue, which has been rarely touched in the load forecasting field;
2) Propose a gradient descent and influence function-based algorithm to efficiently determine the weights of training samples. The proposed algorithm can be used for any model that can be trained using gradient descent methods.

The remainder of this letter is organized as follows: Section II describes the proposed sample reweighting strategy. Section III applies the proposed method to the real-world dataset with both deterministic and probabilistic forecasting. Conclusions and future works are reached in Section IV.

II. Proposed Methodology

A. Problem Statement

Typically, to train a load forecasting model f parameterized by θ, we try to minimize the empirical and structural risk in the training dataset D_{train}. The corresponding optimal model parameters are:

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{N} \sum_{z_i \in \mathcal{D}_{train}} \ell\left(f(x_i, \theta), y_i\right) + \frac{\lambda}{2}\|\theta\|_2^2, \quad (1)$$

The work was supported in part by the National Key R&D Program of China (2022YFE0141200), in part by the National Natural Science Foundation of China (72242105), and in part by the Alibaba Innovative Research Programme. (*Corresponding author: Yi Wang.*)

Chenxi Wang, Yangze Zhou, and Yi Wang are with the Department of Electrical and Electronic Engineering, The University of Hong Kong, Hong Kong SAR, China, and are also with The University of Hong Kong Shenzhen Institute of Research and Innovation, Shenzhen, 518057, China.

Qingsong Wen is with the DAMO Academy, Alibaba Group (U.S.) Inc., Bellevue, WA 98004, USA.

where the first term is the empirical risk and the second one is the structural risk. Each sample $z_i = (x_i, y_i)$ is equally weighted as 1 and N is the size of train dataset. In each sample, the input vector x_i includes historical loads, temperature, calendar features, etc, while y_i represents the actual load. The loss function ℓ will guide the model fitting process, which can be Mean Squared Error (MSE) for deterministic load forecasting or pinball loss function for probabilistic load forecasting. The L2 regularization coefficient λ helps to alleviate the overfitting problem.

Considering the data inequality problem introduced before, we propose to assign a different weight for each training sample in this letter. Our goal is to minimize the weighted training loss and consequently, we can rewrite (1) into:

$$\hat{\theta}(\boldsymbol{w}) = \arg\min_{\theta} \sum_{z_i \in \mathcal{D}_{train}} w_i \ell\left(f(x_i, \theta), y_i\right) + \frac{\lambda}{2}\|\theta\|_2^2, \quad (2)$$

where $\boldsymbol{w} = \{w_1, w_2, \cdots w_n\}$; w_i denotes the weight of sample z_i. For typical load forecasting, $\boldsymbol{w}$ equals to $\mathbf{1}/N$.

The weigh $\boldsymbol{w}$ can be explained as a type of training hyperparameter. It can be adjusted based on the performance of the validation set $\mathcal{L}_{val}$, which is used for approximating the test set:

$$\mathcal{L}_{val}\left(\hat{\theta}(\boldsymbol{w})\right) = \frac{1}{M} \sum_{z_i \in \mathcal{D}_{val}} \ell\left(f(z_i, \hat{\theta}(\boldsymbol{w})), y_i\right), \quad (3)$$

where M is the size of the validation set. Therefore, our proposed reweighting strategy aims to minimize the validation loss while the model parameters are achieved by minimizing the weighted training loss. This implies a bilevel optimization problem:

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \mathcal{L}_{val}\left(\hat{\theta}(\boldsymbol{w})\right) \\ \text{s.t.} \quad & \hat{\theta}(\boldsymbol{w}) = \arg\min_{\theta} \mathcal{L}_{train}\left(\theta, \boldsymbol{w}\right), \end{aligned} \quad (4)$$

where $\mathcal{L}_{train}$ represents the weighted training loss with L2 regularization term in (2).

B. Sample Reweighting Algorithm

To handle the above problem, we extend the thoughts for semi-supervised learning in [7] into load forecasting. Since large load datasets are likely to be involved, we use a gradient-descent method, which shows efficiency in large-scale training, in both inner and outer optimization in (4).

For the inner optimization, with the fixed load sample weights, we can update the model parameters θ to decrease the training loss in a common way. For the outer optimization, with fixed model $\theta(\boldsymbol{w})$, we have to calculate the validation loss gradient w.r.t sample weights $\boldsymbol{w}$, i.e., $\partial \mathcal{L}_{val}(\hat{\theta}(\boldsymbol{w}))/\partial \boldsymbol{w}$.

To acquire this gradient, we can use the chain rule to decompose the gradient into $\nabla_\theta \mathcal{L}_{val}(\hat{\theta}(\boldsymbol{w}))^\top$ and $\partial \hat{\theta}(\boldsymbol{w})/\partial \boldsymbol{w}$. While the former one is relatively simple to obtain, the latter is difficult to compute precisely.

Then, to estimate such a challenging gradient, we start with the thoughts of influence function [8]. The core idea is to calculate the model parameters' change $\Delta\theta$ when we upweight a load sample for a small ϵ from original w_i into $w_i + \epsilon$. Thus, the corresponding optimal model parameters in (2) can be reformulated into:

$$\hat{\theta}(\boldsymbol{w}_{i,\epsilon}) = \arg\min_{\theta} \mathcal{L}_{train}(\theta, \boldsymbol{w}) + \epsilon \ell\left(f(x_i, \theta), y_i\right). \quad (5)$$

For simplicity, we denote $\hat{\theta}(\boldsymbol{w}_{i,\epsilon})$ as $\hat{\theta}_{i,\epsilon}$ and $\hat{\theta}(\boldsymbol{w})$ as $\hat{\theta}$. Since $\hat{\theta}_{i,\epsilon}$ minimize the upweighted loss, we can obtain from the optimal condition:

$$\nabla_{\hat{\theta}_{i,\epsilon}} \mathcal{L}_{train}(\hat{\theta}_{i,\epsilon}, \boldsymbol{w}) + \epsilon \nabla_{\hat{\theta}_{i,\epsilon}} \ell(z_i, \hat{\theta}_{i,\epsilon}) = 0. \quad (6)$$

Then, we conduct Taylor first-order expansion at $\hat{\theta}$ and only keep the $\mathcal{O}(\epsilon)$ terms:

$$\nabla_{\hat{\theta}} \mathcal{L}_{train}(\hat{\theta}, \boldsymbol{w}) + \nabla^2_{\hat{\theta}} \mathcal{L}_{train}(\hat{\theta}, \boldsymbol{w}) \Delta\theta + \epsilon \nabla_{\hat{\theta}} \ell(z_i, \hat{\theta}) \approx 0. \quad (7)$$

Since $\hat{\theta}$ also minimizes $\mathcal{L}_{train}(\hat{\theta}, \boldsymbol{w})$ and we assume that the loss function is second-order differentiable w.r.t θ, we can obtain the estimation of model parameters change:

$$\Delta\theta \approx -\nabla^2_{\hat{\theta}} \mathcal{L}_{train}(\hat{\theta}, \boldsymbol{w})^{-1} \nabla_{\hat{\theta}} \ell(z_i, \hat{\theta}) \epsilon. \quad (8)$$

With the estimation of model parameters change, we can further obtain the following approximation:

$$\begin{aligned} \frac{\partial \hat{\theta}(\boldsymbol{w})}{\partial w_i} &= \lim_{\epsilon \to 0} \frac{\hat{\theta}_{i,\epsilon} - \hat{\theta}}{w_i + \epsilon - w_i} = \frac{\partial \Delta\theta}{\partial \epsilon} \\ &\approx -H_{\hat{\theta}}^{-1} \nabla_{\hat{\theta}} \ell(z_i, \hat{\theta}), \end{aligned} \quad (9)$$

where $H_{\hat{\theta}} = \nabla^2_{\hat{\theta}} \mathcal{L}_{train}(\hat{\theta}, \boldsymbol{w})$ denotes the Hessian matrix of the training loss w.r.t model parameters.

So far, we can consequently write down the validation loss gradient w.r.t w_i through the chain rule:

$$\begin{aligned} \frac{\partial \mathcal{L}_{val}(\hat{\theta})}{\partial w_i} &= \nabla_{\hat{\theta}} \mathcal{L}_{val}(\hat{\theta})^\top \frac{\partial \hat{\theta}(\boldsymbol{w})}{\partial w_i} \\ &= -\nabla_{\hat{\theta}} \mathcal{L}_{val}(\hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_{\hat{\theta}} \ell(z_i, \hat{\theta}) \end{aligned} \quad (10)$$

In this equation, the validation loss gradient w.r.t θ and the Hessian matrix are non-varying with training samples, while $\nabla_{\hat{\theta}} \ell(z_i, \hat{\theta})$ is varying with each training sample.

In practice, the load forecasting dataset may include several years of training samples, which makes it time-consuming to calculate each sample's weight w_i in $\boldsymbol{w}$. Fortunately, with the support of the established automatic differentiation libraries such as Tensorflow, we can vectorize the loss function ℓ, form the training loss Jacobian matrix over all training samples $J_{\hat{\theta}} = \left(\nabla_{\hat{\theta}} \ell(z_1, \hat{\theta}), \nabla_{\hat{\theta}} \ell(z_2, \hat{\theta}), \cdots, \nabla_{\hat{\theta}} \ell(z_n, \hat{\theta})\right)$, and extend it into:

$$\frac{\partial \mathcal{L}_{val}(\hat{\theta})}{\partial \boldsymbol{w}} = -\nabla_{\hat{\theta}} \mathcal{L}_{val}(\hat{\theta})^\top H_{\hat{\theta}}^{-1} J_{\hat{\theta}} \quad (11)$$

If deep models with millions of trainable parameters are involved, the computation burden of the inverse hessian is manifestly heavy as well. This problem can be alleviated through approximating the $H_{\hat{\theta}}$ in [7], [8].

With the above derivation, we can form the eventual sample reweighting algorithm, as shown in **Algorithm 1**:

In general, it can be seen from the above derivation that our proposed algorithm has the following two advantages:

- The proposed strategy does not rely on a specific model and can be used for any parameterized model that can be trained using gradient descent methods.

Algorithm 1 Sample Reweighting Algorithm

Require:
Initial sample weights $\boldsymbol{w} = \mathbf{1}/N$, initial model parameters θ, learning rate η_θ, η_w, outer epoch N_{out}, inner epoch N_{in};

for $n_{out} = 1, 2, ..., N_{out}$ **do**
  for $n_{in} = 1, 2, ..., N_{in}$ **do**
    Update θ: $\theta \leftarrow \theta - \eta_\theta \cdot \nabla_\theta \mathcal{L}_{train}(\theta, \boldsymbol{w})$;
  end for
  Set $\hat{\theta} \leftarrow \theta$;
  Calculate the validation loss gradient w.r.t $\hat{\theta}$: $\nabla_{\hat{\theta}} \mathcal{L}_{val}(\hat{\theta})$;
  Calculate the inverse Hessian matrix w.r.t $\hat{\theta}$: $H_{\hat{\theta}}^{-1}$;
  Calculate the Jacobian matrix w.r.t $\hat{\theta}$: $J_{\hat{\theta}}$;
  Estimate the gradient w.r.t $\boldsymbol{w}$ using (11);
  Update $\boldsymbol{w}$: $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta_w \cdot \frac{\partial \mathcal{L}_{val}(\hat{\theta})}{\partial \boldsymbol{w}}$;
end for

- The two-order differentiable assumption on the loss function is not hard to meet. For deterministic forecasting, the MSE loss function is satisfied manifestly; for the probabilistic one, the pinball loss function can be smoothed by the Huber loss function with ease to fulfill the requirement.

III. Case Studies

A. Experiment Setups

We conduct case studies on the open load dataset from GEFCom2014 [2], which contains the hourly load data and the corresponding temperature data. Specifically, we train, validate, and test our model on the data from 2011 to 2012, from 2013 and from 2014, respectively.

We adopt three popular load forecasting methods, which are linear regression (LR), Artificial Neural Network (ANN), and Long Short Term Memory (LSTM), to be our base models. Additionally, we also include the state-of-the-art boosting model, Gradient Boosting Regression Trees (GBRT), as the benchmark. Boosting has also achieved performance improvements by paying more attention to certain training samples during fitting. For all these models, we perform 24-hour ahead load forecasting in both deterministic and probabilistic ways. All models undergo the same feature engineering with inputs of historical load (last seven days), temperature, and calendar variables (month, day, weekday, hour). For probabilistic load forecasting, the models are guided by smoothed-pinball loss used in [9] and produce 9 quantiles of future loads (from 0.1 to 0.9).

In the process of model training, at the end of each epoch, the validation set will be used to reweight samples in our proposed method while the benchmark method will utilize the same set to early stop fitting if the validation loss does not drop for certain epochs. The settings of the models are listed in Table I.

TABLE I: Experiment Settings

Model	Model Design	Fitting Setting	SR	Early Stop Benchmark
LR	-	optimizer:Adam η_θ: 1e-3 epochs: 300 batchsize: 64 λ: 1e-4	optimizer: SGD η_w: 5e-3	patience: 15 epochs
ANN	FC1: 8 units FC2: 8 units			
LSTM	LSTM: 8 units FC2: 8 units			

TABLE II: Results of Deterministic Load Forecasting

Method	MAPE	RMSE	LMAPE	LRMSE
LR	3.90%	174.50	3.27%	181.75
LR+SR	3.83%	172.75	3.24%	177.15
ANN	2.76%	122.66	2.55%	125.87
ANN+SR	2.58%	113.42	2.32%	115.01
LSTM	2.72%	118.32	2.56%	131.14
LSTM+SR	2.64%	115.50	2.31%	123.20
GBRT	2.68%	120.83	3.00%	137.19

B. Experimental Results

The performance of deterministic load forecasting is evaluated by Mean Absolute Percentage Error (MAPE) and Root Mean Squared Error (RMSE). In addition, we calculate the performance in the winter and summer with large load values (denoted as LMAPE and LRMSE). Table II shows the deterministic load forecasting performance. It is obvious that compared to the equally weighted strategy, three base models enjoy improvement on MAPE and RMSE with the proposed sample reweighting (SR) strategy. Although the improvement on LR was not significant, the MAPE improvement on ANN and LSTM reached 6.5% and 3.0%, respectively, and the average MAPE improvement on the three models exceeds 3.5%. As for GBRT, although it achieves fairly decent performance on overall MAPE and RMSE, our proposed ANN+SR and LSTM+SR methods still outperform it. In addition, on the metrics of LMAPE and LRMSE, our proposed methods are manifestly superior to GBRT, which means that our SR strategy can more effectively improve forecasting performance during seasonal peak periods.

Besides, we calculate the final weight of each load data point by averaging the weights of the samples which use this load data during fitting. Fig. 1 shows the final learned weights in 2011 (on LR as an example). It can be seen that in general, the weights of most load data fluctuate around 1, which is the initial weight. The weights for summer loads, however, are assigned obviously larger than others. It allows the model to focus more on the relevant training samples, which is the possible reason that decreases the LMAPE and LRMSE.

For probabilistic load forecasting, our proposed strategy is also able to provide a solid improvement in forecasting performance. The probabilistic forecasting performance is evaluated by the average pinball loss, Winkler score of 80% prediction interval (PI), and 60% PI. Table III shows the comparison of forecasting performance. Despite the slight performance lag of WS (60%PI) on LR, our proposed SR strategy improves the average pinball loss of these three models by more than 4%, demonstrating its effectiveness in probabilistic forecasting as well. Compared with GBRT, the better performance of our SR-assisted models can also be observed for probabilistic forecasting.

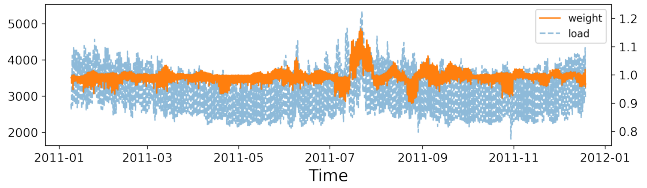

Fig. 1: Load Curve and the Corresponding Sample Weights in 2011 on LR model (weights are multiplied by N)

TABLE III: Results of Probabilistic Load Forecasting

Method	Pinball Loss	WS (80%PI)	WS (60%PI)
LR	50.280	609.763	462.990
LR+SR	50.229	607.124	462.920
ANN	38.658	467.163	354.953
ANN+SR	34.771	416.573	319.439
LSTM	36.072	439.336	329.696
LSTM+SR	35.129	413.291	319.917
GBRT	37.223	446.959	338.857

IV. Conclusions and Future Works

In this letter, we proposed a sample reweighting strategy to improve the performance of load forecasting. The proposed method is model-independent and can be applied to both deterministic and probabilistic load forecasting with ease. Furthermore, in the case study, we showed that our proposed strategy could lead the model to focus on critical samples during the fitting process and thus improve the forecasting performance. In the future, we will explore the influence of the validation sets and the online learning extension to update the sample weights in a regular manner.

References

[1] T. Hong, P. Pinson, Y. Wang, R. Weron, D. Yang, and H. Zareipour, "Energy forecasting: A review and outlook," *IEEE Open Access Journal of Power and Energy*, vol. 7, pp. 376–388, 2020.

[2] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, "Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond," *International Journal of Forecasting*, vol. 32, no. 3, pp. 896–913, 2016.

[3] C. Li, Z. Dong, L. Ding, H. Petersen, Z. Qiu, G. Chen, and D. Prasad, "Interpretable memristive LSTM network design for probabilistic residential load forecasting," *IEEE Trans. Circuits and Systems I: Regular Papers*, vol. 69, no. 6, pp. 2297–2310, 2022.

[4] Y. Zhou, Z. Ding, Q. Wen, and Y. Wang, "Robust load forecasting towards adversarial attacks via bayesian learning," *IEEE Trans. Power Systems*, 2022.

[5] C. Wang, Y. Wang, Z. Ding, T. Zheng, J. Hu, and K. Zhang, "A transformer-based method of multienergy load forecasting in integrated energy system," *IEEE Trans. Smart Grid*, vol. 13, no. 4, pp. 2703–2714, 2022.

[6] M. Ren, W. Zeng, B. Yang, and R. Urtasun, "Learning to reweight examples for robust deep learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4334–4343.

[7] Z. Ren, R. Yeh, and A. Schwing, "Not all unlabeled data are equal: Learning to weight data in semi-supervised learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 21 786–21 797, 2020.

[8] P. W. Koh and P. Liang, "Understanding black-box predictions via influence functions," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1885–1894.

[9] Y. Wang, D. Gan, M. Sun, N. Zhang, Z. Lu, and C. Kang, "Probabilistic individual load forecasting using pinball loss guided LSTM," *Applied Energy*, vol. 235, pp. 10–20, 2019.