Multi-Stage Robust Wind Power Forecasting

Chenxu Wang
University of Science and Technology of China
China
wcx123@mail.ustc.edu.cn

Jinda Lu
University of Science and Technology of China
China
jackie64321@gmail.com

Yuan Gao
University of Science and Technology of China
China
928852036@qq.com

ABSTRACT

As a pollution-free and renewable energy, wind power has great potential for development. Some of the largest obstacle that hinders the process of incorporating wind energy into a grid system are noisy data and uncertainty. As the active power output of wind farms is highly dependent on local weather conditions such as wind speed and wind direction which change rapidly, accurate wind power forecasting (WPF) is a promising task that draws great attention. In this paper, we propose a novel framework that ensembles LightGBM model and GRU model. Tree-based model is robust to outliers and missing values, and have good interpretability which helps with feature engineering optimization. These advantages make LightGBM one of the most common methods in data mining competitions. And to handle with time-series feature, we build encoder-decoder architecture based on GRU to capture correlation of different time scales. Some heuristic techniques are adopted to facilitate model learning. Online evaluation shows that our model gives a reliable prediction of the active power. The code is available at https://github.com/injadlu/KDDCUP2022.

KEYWORDS

Machine Learning, Data Mining, Wind Power Forecasting

ACM Reference Format:
Chenxu Wang, Jinda Lu, and Yuan Gao. 2018. Multi-Stage Robust Wind Power Forecasting. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Renewable energy, such as wind energy, plays an increasingly important role in power system, due to its clean and safe nature[5]. Sometimes the electricity generated by the wind turbines is transmitted to large utility grids which have high demand of power quality and stability[6]. Hence wind power forecasting (WPF) is a crucial task, and accurate prediction could possibly help with management strategies like load dispatch planning and maintenance schedule [2].

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

However, WPF is challenging for two reasons: noisy data and high variability. First of all, some critical features such as wind speed and wind direction can be missing or have abnormal values due to the operating status of the sensors, which put forward high requirements on model robustness; additionally, levels of production of wind energy have high variability [4]: this variability comes from its dependence on unstable weather conditions present at the wind farm, from this perspective, summarizing dynamic historical time-series data to predict the future is an intuitive way to handle the problem.

In our work, to address above challenges, we investigate strategies on ensembling LightGBM [3] and GRU [1] model. For robustness, LightGBM can learn from outliers and missing values, and have good interpretability which helps promote the quality of feature engineering. What's more, tree-based models have non-linear capability and are unlikely to overfit. In addition, to alleviate variability issue, we design an encoder-decoder architecture based on GRU, which predict the future data with historical sequence of the same wind farm. Besides, we propose some heuristic techniques including data preprocessing and feature engineering. The experiments show that our solution can achieve relatively low value on the given evaluation metric.

2 METHOD

In this section, we will introduce feature engineering, model architecture, and model prediction in detail. In Section 2.1, we have outlined the overall structure of our model. In Section 2.2 and Section 2.3, we further introduce the pre-process of data, model training, and the post-process of prediction.

2.1 Model Overview

As shown in the Fig 1, our method can be divided into three parts: data preprocessing, model training and prediction result post-processing. The dataset used for the competition collected 245 days of the wind farm. The data is collected in real scenarios and contains a large amount of dirty data. Therefore, we first simply cleaned the data and use the relatively cleaner data to support the model training. To ensure the robustness of the model, we trained two different types of models for each wind turbine which consist of the neural network model GRU and the GBDT model LightGBM. So we will get 134 groups of models. They are often used to solve time series prediction problems. As for the difference of two types of models, we need to generate training data in different formats. While training the model, we also design a new optimization objective for GRU. Finally, We fuse the prediction results of two models and aggregate the prediction results of adjacent wind turbines using the location information of wind turbines.

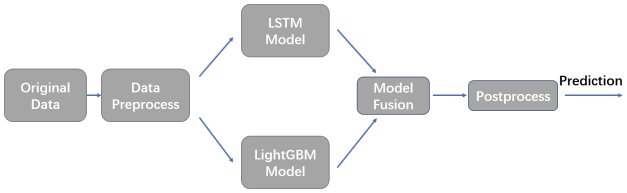
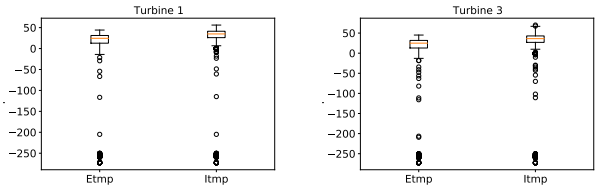


Figure 1: Model overview for our method. It can be divided into three parts: data preprocessing, model training and prediction result postprocessing



(a) Turbine 1 (b) Turbine 3

Figure 2: The box figure of two wind turbines' temperature features.

2.2 LightGBM Model

2.2.1 Data Preprocessing. LightGBM is one of the most common methods in data mining competitions which is very convenient to fit the tabular data. However, there is a lot of noise in the data. For example, as shown in the Fig 2, the environment temperature and the internal temperature of wind turbine in the data have many outliers, which make the data harder to fit. First of all, we fill the missing data with the mean value of the corresponding feature. And then, we calculate the median of temperature in groups of the day across different wind turbines t_m. For the temperature feature $t_i$in the dataset, if t_i is far from the corresponding t_m, we replace the t_i with t_m. Formally, if $|t_i - t_m| > 10$, we treat t_i as an outlier. Considering that some sensors are aging, the collected data may be biased.

We decided to reduce the uncertainty in the data by smoothing the data from adjacent wind turbines. We define the position of the wind turbine T_i is (x_i, y_i). For each turbine pair (i, j), we calculate the Euclidean Distance $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ and choose four nearest wind turbines as neighbors for each one. For turbine T_i, assume that the distance and features of T_i with itself and its neighbors are $d = (0, d_{i,i_1}, d_{i,i_2}, d_{i,i_3}, d_{i,i_4})$ and $f = (f_i, f_{i1}, f_{i2}, f_{i3}, f_{i4})$. We obtain the proportion of aggregation $a = softmax(\frac{d}{\rho})$. Finally, the aggregated feature of turbine T_i is $f_{T_i} =< f, a >$. Furthermore, according to the value of a, we define the adjacency matrix $S \in \mathcal{R}^{134\times134}$. For each line of S, there are five nonzero values which are equal to the output of the softmax function. The entire data preprocessing is shown in Figure 3.

2.2.2 Feature Engineering. We construct training data mainly by the lag features. First of all, We compress the amount of data. We replace the features of the adjacent timestamps with the mean of their features As a result, the number of features per 24 hours

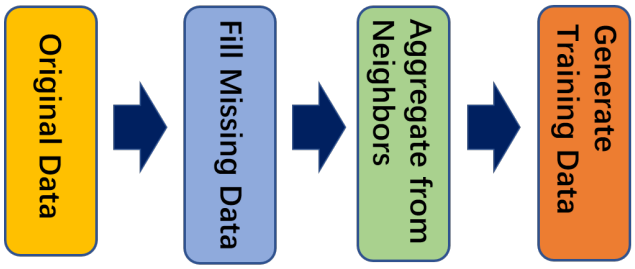


Figure 3: Data preprocessing for LightGBM training.

dropped from 144 to 72. Since we have 10 columns of features, the total number of features is still huge, and we want to sift through the most important ones. We hope to select the regular and meaningful features. After carefully selecting, we keep $Wspd, Etmp, Itmp, Patv$ as the feature for LightGBM training. We will discuss the feature selection in Section 3.1.2. Assume that recent timestamp is t, we want to predict the power of the subsequent 288 timestamps, which can be formulated as $P_{t+1}, P_{t+2}, ..., P_{t+288}$. As mentioned above, we construct new features using the mean values of adjacent timestamp features. We will also use one model to predict the power of two timestamp. That is $\hat{P_{t+1}} = \hat{P_{t+2}} = LGB_1(f^t)$. $\hat{P_{t+1}}$ means the predicted power of timestamp $t+1$ and f^t means the feature of timestamp t. f^t is constructed by lag features with 108 order. It can be formulated as

$$f^t = (Wspd_{t-107}, Etmp_{t-107}, Itmp_{t-107}, Patv_{t-107}, Wspd_{t-106}, Etmp_{t-106}, Itmp_{t-106}, Patv_{t-1067}, ..., Wspd_t, Etmp_t, Itmp_t, Patv_t) \quad (1)$$

. And, the label of training LGB_i is $\frac{Patv_{2i-1}+Patv_{2i}}{2}, i = 1, 2, ..., 144$.

2.3 GRU Model

GRU is a popular model in handing time series data. In this competition, we adopt the GRU as a baseline model. Followings are the detailed description of our GRU method.

2.3.1 Data Preprocessing. The abnormal conditions is shown in Table 1.The whole dataset contains 134 turbines with 35280 timestamps, hence the total number of data points is 4727520. However, the total number of abnormal points is 1403543, which means almost 30% of the training data is noisy. To tackle the noisy data problem, we block the noisy data, and use the pure clean data for GRU model optimization. We calculate mean and std statistics on the clean data for each turbine. before put into GRU model, the data is normalized by z-score normalization with clean mean and std. Each data point contains 13 attributes, we select the normalized $Wspd, Wdir, Etmp, Itmp, Ndir, Pab1, Pab2, Pab3, Prtv, Patv$ as the feature for each data point.

2.3.2 Model Architecture. As shown in Fig 4, our GRU model is an encoder-decoder architecture, and the GRU encoder shares hidden states with the GRU decoder.The GRU encoder architecture is designed to capture the time correlation information in input sequence. And the GRU decoder is adopted to generate the output sequence. Considering to improve the responsiveness for short time series of the model, we propose 2 GRU model, 1 for the first half

length output, namely GRU-FH(First Half), and another for total length output, namely GRU-ALL. In this competition, for GRU-ALL encoder, the length of input sequence is 144, the input sequence is sampled from the dataset, which indicates the features of the most recent day. And for GRU-ALL decoder, the length of input sequence is 288, the input sequence is all-0. The output of decoder is our prediction for the next 2 days. The input sequence of our GRU-FH encoder is same as the GRU-ALL. For GRU-FH decoder, the length of input sequence is 144, which indicates that our GRU-FH decoder only predict for the next day.

2.3.3 Optimization. As the competition calculates the *rmse* and *mae* distance as final score of the model, we adopt a combination of *rmse* and *mae* loss. Our loss function can be formalized as :

$$Loss = \alpha * rmse + (1 - \alpha) * mae \tag{2}$$

For a turbine with 288 length sequence output, the loss can be formalized as :

$$Loss = \alpha * \sum_{i=0}^{288} \sqrt{(y^i - f(g(x))^i)^2} + (1 - \alpha) * \sum_{i=0}^{288} |(y^i - f(g(x))^i)| \tag{3}$$

In Eq3, y indicates the 288 length sequence ground truth, x refers to the 144 length sequence input. g is the encoder, and g takes x as input, and saves the information of input sequence into hidden states R. f is the decoder, f takes 288 length all-zero sequence as input, with hidden states initialized as R, and f outputs the 288 length sequence prediction.

To capture the time correlation information in training phase, we keep the noisy data. However, optimizing with noisy data makes the GRU model biased. To alleviate the issue above, when calculate loss, we construct a clean data index set S, and calculate loss with indexed clean data. For a training sample(x, y), x indicates the input sequence with 144 length, y refers to the ground truth sequence with 288 length. As discussed above, to keep the time series information, we keep the noisy x, remove the noisy information in y. For a sample (x, y), S can be formalized as :

$$S = (i | y^i \notin Abnormal\ Set) \tag{4}$$

Abnormal Set contains all data fits the abnormal conditions, the abnormal conditions is shown in Table1. Thus, for a turbine training, the Loss can be formalized as :

$$Loss = \alpha * \sum_{i \in S} \sqrt{(y^i - f(g(x))^i)^2} + (1 - \alpha) * \sum_{i \in S} |(y^i - f(g(x))^i)| \tag{5}$$

The definition in Eq5 is the same as Eq3 and Eq4. Compared to Eq3, in Eq5, we do not choose all y^i for loss calculation, but select the clean y^i for loss calculation and model optimizing. This indicates that our GRU model can not just capture the time correlation information, but more crucially, optimizing on the right step.

2.4 Model Fusion

To provide the robust prediction, we adapt the model fusion to guarantee the performance around different test phases. $LGB[a : b]$ denotes the prediction from *LightGBM* on timestamp $[a, b)$. We design different fusion strategies for different timestamps. This is because different types of models have different strengths. Then, the prediction after model fusion can be formulated as follow:

index	Abnormal conditions
1	NaN in raw data
2	[Patv] < 0
3	[Patv] = 0 & [Wspd] > 2.5
4	[Pab1] >89 | [Pab2] >89 | [Pab3] >89
5	[Wdir] > 180 | [Wdir] < -180
6	[Ndir] > 720 | [Ndir] <-720

Table 1: Abnormal conditions.

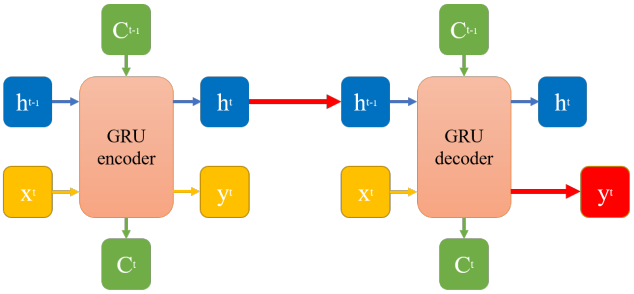


Figure 4: GRU Model overview. Left is the encoder part, right is the decoder part

$$Prediction[0:96] = S^2 \cdot (0.8 * LGB[0:96] + 0.1 * GRU - FH[0:96] + 0.1 * GRU - ALL[0:96]) + b_1 \tag{6}$$

$$Prediction[96:144] = 0.5 * GRU - FH[96:144] + 0.5 * GRU - ALL[96:144] + b_2 \tag{7}$$

$$Prediction[144:288] = GRU - ALL[144:288] + b_3 \tag{8}$$

S denotes the adjacency matrix defined in Section 2.2. We propagate information twice from the neighbor nodes. b_1, b_2, b_3 denotes the constant bias in our method. We adjust them both in the offline and online test data. In the following paper, we define the **Postprocessing** as adding the bias term and multiplying the adjacency matrix.

3 EXPERIMENTS

In this section, we will add more details about model training and serving to improve reproducibility. What is more, we also show the empirical results of offline and online testing to explain the motivation of each component of our method.

3.1 Training Details

We use the proposed SDWPF dataset[7] as our training set. SDWPF dataset contains 134 wind turbines with 35280 data points for each turbine. The power of each wind turbine is predicted by an independent group of models. T his is to say, we have 134 groups of models. Each group contains three models, which contains two GRU models and one LightGBM model.

3.1.1 LightGBM Training. For LightGBM model training, we divided the training/validation dataset on a scale of 0.8/0.2. After generating the training and validation dataset, we grid search the

hyperparameters according to the best performance on validation dataset. The metric we choose for LightGBM is the L2 Loss Function. The learning rate is set as 0.05. The *maxdepth* and *maxleaves* are 3 and 8. The *maxbin* is set as 30. When the validation performance does not improve in the 200 iterations, we stopped training early.

3.1.2 Feature Selection for LightGBM. As discussed in Section 2, we select 4 different columns of features to construct the lag feature. We treat all the numerical features in one timestamp as the input of a LightGBM model to fit the power. In detail, the input features are $TurbID, x, y, Wspd, Wdir, Etmp, Itmp, Ndir, Pab1, Pab2, Pab3, Prtv$. We also split the whole dataset as 0.8/0.2 for training/validation. The results are shown on the Table 3.1.3. We design a feature ablation study to seek the most important features for forecasting the power of wind turbine. When we take out the direction feature $(Wdir, Ndir)$, the validation performance decreases a little. That's because wind turbines automatically align themselves with the wind direction. Therefore, we do not need there features. When we remove the *Pab* features, performance decreases a lot. *Pab* determines the area of contact between the fan blade and the wind. However, because the *Pab* of a wind turbine is adaptive, it is difficult to predict. As for the *Prtv*, getting rid of it doesn't have much effect either. If we remove the temperature features $(Etmp, Itmp)$, the performance decreases significantly. The temperature features also have some regularity. Therefore, we finally select $(Wspd, Etmp, Itmp, Patv)$ to construct the lag feature.

3.1.3 GRU Training. We have 2 GRU models for each turbine, we have 134 GRU-FH models and 134 GRU-ALL GRU-models in total. The configs of the GRU-FH and GRU-ALL are the same. The encoder is a 1-layer GRU with hidden size equals 96, decoder is the same as encoder. the decoder is followed by a linear projection head to transform the GRU output into output space.

For each turbine,we divide the dataset into 214/31 days, the first 214 days as the training set, and the last 31 days as the validation set. As discussed in Section 2, we calculate mean and std statistics among the whole clean data for each turbine. Our GRU models are trained with the normalized data. To capture the time correlation information in training phase, we keep the noisy data. To prevent the model biased by the noisy data, when optimizing, we construct a clean index set S, select clean data indexes from S, and compare the corresponding prediction with ground truth. For each data point,we select the normalized $Wspd, Wdir, Etmp, Itmp, Ndir, Pab1, Pab2, Pab3, Prtv, Patv$ as the feature. We train our each GRU model with 10 epoches, and use early stopping strategy by the validation loss.As discussed in Section 2, the loss function can be formalized as $\alpha * rmse + (1 - \alpha) * mae$, in training phase, we make the α equals 0.5. The optimizer is Adam with weight decay equals 1e-4. The original learning rate is 1e-4, and the learning rate is divided by 0.5 for each 2 epoches.

3.2 Performance Comparison

In this section, we compare several variants of our methods to show the effectiveness of each component. We also construct offline evaluation. We use data from the last 31 days to generate $test_x$ and $test_y$. The metric and test data filtering of offline test are the same as online test. The results are shown on Table 3.1.3. We choose

Feature Ablation	Train's L2	Valid's L2
All Feature	3111.79	4839.4
-[Ndir,Wdir]	3900.84	5089.14
-[Ndir,Wdir,Pab1,Pab2,Pab3]	20765.7	9691.74
-[Ndir,Wdir,Pab1,Pab2,Pab3,Prtv]	39849.8	9867.13
-[Ndir,Wdir,Pab1,Pab2,Pab3,Prtv,Etmp,Itmp]	31378.7	16915.2

Table 2: Feature ablation study for LightGBM model.

Model Variants	Offline Evaluation	Online Evaluation
Baseline GRU Model	37.75	45.52
GRU w/o dp and new loss	37.10	45.16
GRU-Ours	36.50	44.62
LightGBM w/o dp	37.85	45.05
LightGBM-Ours	37.40	45.01
Model Fusion w/o pp and first 144	36.38	44.50
Model Fusion-Ours	36.30	44.48

Table 3: Model variants of our methods. The w/o means *without*, *dp* means *datapreprocessing*, *pp* means *predictionpostprocessing* and $first144$ means whether fusion the GRU model which only predicts the first 144 timestamps.

results on Phase 2 as our Online Evaluation. According to the table, we have following observations:

First, we observe that new loss works on GRU model for both offline and online evaluation, which indicates the effectiveness of clean dataset S: for offline evaluation, the performance gain is 1.72%, and it gets 0.79% improvement in online evaluation. Additionally, for both GRU and LightGBM model, data pre-processing boosts the performance of the original model, which verifies that appropriate feature engineering can help with the prediction: for LightGBM, the performance gain is 1.19%, and it gets 1.62% improvement for GRU model. Comparing fused model and single model, we conclude that model fusion could bring about great improvement, and proper post-processing is also helpful. The final score we got in online evaluation and offline evaluation is 36.30 and 44.48 respectively.

REFERENCES

[1] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).

[2] Guoqiang Ji, Wenchuan Wu, and Boming Zhang. 2016. Robust generation maintenance scheduling considering wind power and forced outages. *IET Renewable Power Generation* 10, 5 (2016), 634–641.

[3] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems* 30 (2017).

[4] Paul Veers, Katherine Dykes, Eric Lantz, Stephan Barth, Carlo L Bottasso, Ola Carlson, Andrew Clifton, Johney Green, Peter Green, Hannele Holttinen, et al. 2019. Grand challenges in the science of wind energy. *Science* 366, 6464 (2019), eaau2027.

[5] Katya Vladislavleva, Tobias Friedrich, Frank Neumann, and Markus Wagner. 2011. Predicting the Energy Output of Wind Farms Based on Weather Data: Important Variables and their Correlation. *CoRR* abs/1109.1922 (2011).

[6] Eddie Yatiyana, Sumedha Rajakaruna, and Arindam Ghosh. 2017. Wind speed and direction forecasting for wind power generation using ARIMA model. In *2017 Australasian Universities Power Engineering Conference (AUPEC)*. 1–6.

[7] Jingbo Zhou, Xinjiang Lu, Yixiong Xiao, Jiantao Su, Junfu Lyu, Yanjun Ma, and Dejing Dou. 2022. SDWPF: A Dataset for Spatial Dynamic Wind Power Forecasting Challenge at KDD Cup 2022. *Techincal Report* (2022).