MSBD 5002 Project 2023

Wind Power Forecast with Recurrent Neural Networks

Yi Lin Ye[1]

[1]20922644, ylye@connect.ust.hk

May 22, 2023

Abstract

Accurate forecasting of wind power production is crucial in many ways. In this report, I present my approach to the 2022 KDD Cup Spatial Dynamic Wind Power Forecast, which aims to facilitate progress in data-driven machine learning methods for wind power forecasting. The dataset included 245 days of historical context factors for 134 wind turbines, and students were required to predict future power production for a 2 days interval (288 10-minute period) for each turbine. Two recurrent neural network models, plain RNN and GRU, were adopted. Then the models result are ensembled to achieve high prediction accuracy. In addition, feature engineering and kNN imputation method are applied to the model and described in detail below. This approach achieved an RMSE of 53.58 and an MAE of 44.03.

Keywords
Machine Learning, Wind Power Forecast

1 Introduction

The integration of renewable energy sources such as wind power into the electric grid system poses a significant challenge due to the high variability in their production. Accurate forecasting of wind power production is crucial for maintaining a balance between electricity generation and consumption. The Wind Power Forecasting (WPF) problem has been widely studied in the data mining and machine learning community, but achieving high prediction accuracy remains a challenge due to the complexity of the problem. The 2022 KDD Cup proposed a Spatial Dynamic Wind Power Forecasting Challenge to facilitate progress in data-driven machine learning methods for WPF. The challenge dataset, provided by Longyuan Power Group Corp. Ltd., included over 8 months of historical data for 134 wind turbines, including power production, wind speed, direction, temperature, and other turbine internal status variables. We were tasked with predicting future power production for a two-day interval for each of the 134 turbines with RMSE and MAE as metrics. In this report, I will analyze the dataset, examine my approaches and evaluate the results of the challenge. Through this analysis, I aim to increase the accuracy of WPF with machine learning or deep learning technique. Moreover, through this challenge, I hope to gain insights into the latest trends and techniques in the field of wind power forecasting and their potential applications in the renewable energy industry.

2 Method

In this section, I will introduce feature engineering, model architecture, and model prediction in detail. Section 2.1 outlined the overall structure of the model. Section 2.2 and Section 2.3 further introduce the preprocessing of data, model training, and the model performance.

2.1 Model Overview

Inspired by baseline methodology and several KDD cup winning teams' papers[1] [2][3][4], I adopted two recurrent neural networks (RNN) models, namely the plain RNN and Gated Recurrent Unit (GRU), as the basic models. I chose these two models because they are both effective at capturing the temporal dependencies in sequential data, including wind power data, and have been shown to achieve good performance in wind power forecasting tasks.

The data pre-processing work involved using two different imputation methods, namely linear interpolation and kNN interpolation, to handle missing data before constructing 134 RNN and GRU models, allowing one to compare the performance of the models and determine the optimal imputation method for our dataset and modeling task.

After training the plain RNN and GRU models separately, models are ensembled based on the loss of the validation set to obtain the final prediction scores. Ensembling is a technique that combines the predictions of multiple models to obtain a more accurate and robust forecast. In our case, a 50% weight is assigned to each of the two models in the ensemble.

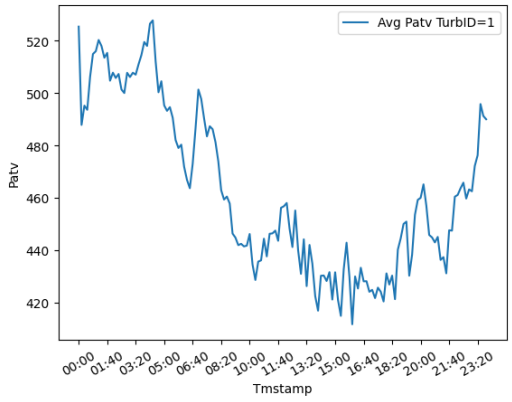

Figure 1: Average Wind Power of Turbine ID 1

Column	Feature	Specification
1	TurbID	Wind Turbine ID
2	Day	Date of the record
3	Tmstamp	Created time of the record
4	Wspd (m/s)	The wind speed recorded by the anemometer
5	Wdir (°)	Angle between wind direction and turbine nacelle
6	Etmp (°C)	Temperature of the surrounding environment
7	Itmp (°C)	Temperature inside the turbine nacelle
8	Ndir (°)	Nacelle direction, i.e., the yaw angle of the nacelle
9	Pab1/2/3(°)	Pitch angle of blade 1/2/3
10	Prtv(kW)	Reactive power
11	Patv(kW)	Active power height

Table 1: Features Specification

2.2 Data Exploration and Preprocessing

The Spatial Dynamic Wind Power Forecasting (SDWPF) dataset used in the challenge was collected from the Supervisory Control and Data Acquisition (SCADA) system of a wind farm. The dataset includes 10-minute samples of critical external features such as wind speed, wind direction, and external temperature, as well as essential internal features such as inside temperature, nacelle direction, and pitch angle of blades. The dataset consists of 134 wind turbines, and each turbine can generate wind power separately. The average wind power in 245 days of an individual turbine (ID 1) for different time periods in a day is plotted in Figure 1. However, the output power of the wind farm is the sum of all the wind turbines' power output at any given time. A detailed introduction of the main features of the data is provided in Table 1.

Figure 2 shows the features correlation heatmap. It suggests that the variable Patv has the highest correlation with wind speed. This indicates that wind speed could be a valuable input for longer-term power prediction using Patv.

With 10 columns of features in the original dataset, it is still huge for our model to run efficiently, and thus we need to select the most relevant ones for training our model. After a careful selection process, 5 key features have been identified *Day*, *Tmstamp*, *Wspd*, *Wdir*, and *Etmp* to be used for training our model.

There are several caveats to consider when using the

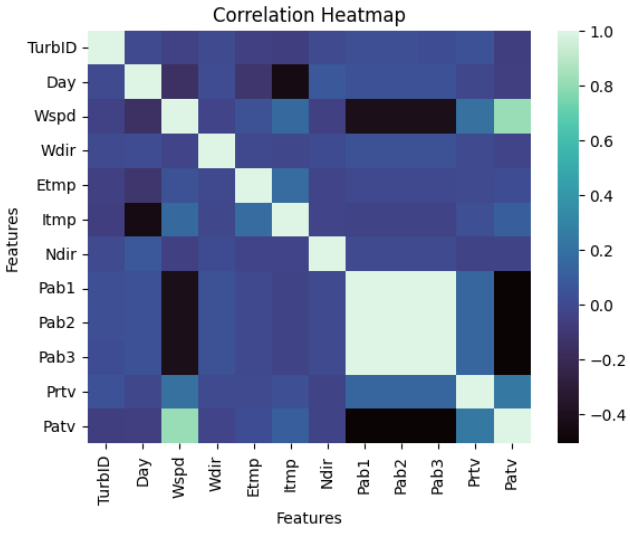

Figure 2: Heatmap for all the features

SDWPF dataset for training and evaluating models, including zero, missing, unknown, and abnormal values. These are all defined and explained in [1]. A significant portion of the SDWPF dataset contains missing and abnormal data. To address this issue and facilitate our wind power forecasting task, two approaches are used to clean up the dataset.

- **Linear interpolation.** Forward and backward linear interpolation are conducted on missing and unknown values while the remaining NAN are set to zeros.

- **KNN interpolation.** K-nearest neighbor (kNN) interpolation method which uses distance measurements to identify neighboring points and estimates missing values using the full value of neighboring observations is employed to address the data issues. The choice of the number of neighbors (k) is a critical parameter in the kNN algorithm, as a large k can result in filling missing values with the mean value of the entire dataset, while a small k may introduce noise from nearby neighbors. For this model, **10** nearest neighbor values were used in the kNN interpolation to strike a balance between these two considerations.

After interpolation, standard normalization provided by the baseline code [1] was applied to *Wspd*, *Wdir*, *Etmp*, and *Patv*. To encode the day and time stamp of a time series data, traditional feature engineering techniques were employed using sine and cosine functions. Specifically, the angles representing time were transformed into their respective sine and cosine components.

2.3 Models: RNN & GRU

Recurrent Neural Networks (RNNs) are a type of neural network that can handle sequential data by maintaining a hidden state that is updated at each time step. The key idea behind RNNs is to use the previous hidden state and the current input to compute the next hidden state, which is then used to make a prediction for the next time step. This allows RNNs to capture the temporal dependencies in the data and model the dynamics of the underlying process[5].

In the context of wind power forecasting, an RNN can take into account historical wind power measurements and weather conditions to make predictions for future time steps. For example, the input at time step t includes the wind speed, wind direction, and other relevant weather variables at time t, as well as the wind power output at previous time steps. The RNN can then be trained to predict the wind power output for the next time step based on this input[6].

However, one limitation of plain RNNs is the vanishing gradient problem, which can occur when the gradients become very small as they are backpropagated through time, making it difficult to train the network effectively[6]. To address this issue, Gated Recurrent Units (GRU) were introduced as a variant of RNNs that use gating mechanisms to control the flow of information into and out of the hidden state.

GRU uses two gates, an update gate and a reset gate, which regulate the flow of information into and out of the hidden state. The update gate controls how much of the previous hidden state should be retained and how much of the current input should be added to the new hidden state, while the reset gate determines how much of the previous hidden state should be disregarded. The update gate and reset gate are both sigmoid functions that take the current input and the previous hidden state as inputs.

At each time step t, a GRU computes the candidate activation h_t as a function of the current input x_t and the previous hidden state h_{t-1} :

$$
\begin{aligned}
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \\
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\
\tilde{h}t &= \tanh(Wxhx_t + r_t * (W_{hh}h_{t-1}) + b_h) \\
h_t &= (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t
\end{aligned}
$$

where *sigma* is the sigmoid activation function, *tanh* is the hyperbolic tangent activation function, and * denotes element-wise multiplication. W and b are the parameters of the GRU, which are learned during training.

In wind power forecasting, GRUs have been shown to be effective at capturing complex temporal patterns in wind power data. The gating mechanism in GRUs enables them to selectively update the hidden state and control the amount of information that is retained from previous time steps, which can help to alleviate the vanishing gradient problem and capture long-term dependencies in the data[2].

Model	Hyperparameter	Attempt 1	Attempt 2
RNN	Input Length	36	144
	Input Size	6	6
	Hidden Units	12	12
	Layers	4	4
GRU	Input Length	36	144
	GRU Layers	4	2
	Hidden Units	12	8
	Learning Rate	0.25e-4	5e-4
	Batch Size	64	64

Table 2: Model Hyperparameter

2.4 Model Training & Experiment

The RNN and GRU models are trained individually for each of the 134 turbines using mean square error (MSE) as the training loss. Datasets with two different imputation methods, namely linear interpolation and kNN interpolation, are used separately in constructing 134 RNN and GRU models with the same structure. The data was split into a training set consisting of the first 214 days and a validation set consisting of the remaining 31 days.

To examine the difference and optimize the forecast, I take two attempts to test the model performance with different hyperparameters. The optimized hyperparameters include input length, number of layers, number of hidden nodes, and learning rates. The hyperparameter values for attempt 2 were referenced from CSU team's work[2]. The hyperparameters for RNN and GRU in different attempts are listed in Table 2.

In addition, the ensemble method is applied to the plain RNN and GRU methods. An equal weight is assigned to each of the models to mitigate the weaknesses of each model.

Ensembling the two models can help to mitigate the weaknesses of each model and exploit their strengths. For example, the plain RNN may be better at capturing short-term dependencies, while the GRU might be better at capturing long-term dependencies. By combining the predictions of the two models, we can obtain a more comprehensive and accurate forecast.

3 Results

The detailed experimental results of the models are listed in Table 3.

Based on our experimental results, the best score of my model over the tested time series of 50 predictions is: RMSE of 53.58, MAE of 44.03, and an overall score of 48.81. The training time is around 5.5 hours for each model and the evaluation time for 50 predictions is around 22 minutes on a Linux machine with Nvidia Tesla T4 GPU.

Data Preprocess	Model	Overall Score	RMSE	MAE	Train Time	Eval Time	GPU
Linear Interpolation	RNN Attempt 1	49.68	54.84	44.52	302 min	22 min	V100
	RNN Attempt 2	51.32	57.04	45.59	330 min	22 min	T4
	GRU Attempt 1	49.69	54.91	44.48	306 min	21 min	T4
	GRU Attempt 2	50.27	55.05	45.50	270 min	22 min	T4
	Ensemble 1	49.62	54.80	44.45	-	21 min	T4
	Ensemble 2	50.27	55.38	45.10	-	22 min	T4
kNN Imputation	RNN Attempt 2	49.45	54.77	44.13	321 min		T4
	GRU Attempt 2	48.95	53.67	44.23	309 min	22 min	T4
	Ensemble 2	48.81	53.58	44.03	-	21 min	T4

Table 3: Model Performance

Acknowledgements

The model ideas are inspired by baseline and paper from a few different teams in KDD cup 2022[1][3][2]. The code is built on top of baseline code and code from Team Zealen in KDD cup[1] [4].

References

[1] Jingbo Zhou, Weiwei Sun, Yu Liu, and Jie Yang. Sdwpf: A dataset for spatial dynamic wind power forecasting challenge at kdd cup 2022. Technical report, Technical Report, 2022.

[2] Zhi Liu, Min Li, Baichuan Yang, and He Wei. Spatial wind power forecasting using a gru-based model: Windteam csu123. In *Proceedings of ACM Conference (Conference'17)*, page 3, New York, NY, USA, 2022. ACM.

[3] Fangquan Lin, Wei Jiang, Hanwei Zhang, and Cheng Yang. Kdd cup 2022 wind power forecasting team 88vip solution. In *Proceedings of KDD CUP 2022 (SIGKDD'22)*, page 6, New York, NY, USA, 2022. ACM.

[4] Ming Li, Lin Zhang, Yizhou Wang, Kai Zhang, Yifan Zhang, Yudong Liu, and Xuejiao Yang. Spatial-temporal recurrent neural network for wind power forecasting in baidu kdd cup 2022. *arXiv preprint arXiv:2205.08733*, 2022.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[6] Xianyong Wang, Weizhou Zhang, and Jianhua Wang. Wind power forecasting based on deep learning algorithms: A review. *Energy*, 172:130–147, 2019.