Wind energy production forecasting

Floris Ouwendijk [a] Henk Koppelaar [a] Rutger ter Borg [b]

Thijs van den Berg [b]

[a] Delft University of Technology, PO box 5031, 2600 GA Delft, the Netherlands

[b] NUON Energy Trade and Wholesale, Spaklerweg 20, Amsterdam, the Netherlands

Abstract

This paper presents a study to forecast the total production of some wind parks in The Netherlands. The predictions are based on several forecasts of the wind speed and wind direction. Two techniques are used: k-nearest neighbours and Lasso. These techniques are applied to three versions of the problem. The initial version scales the features. The second version modifies the wind directions to a more natural data format. The third version incorporates lags in the dataset. Each version improves the results somewhat, with the k-nearest neighbour technique having the lowest errors, and Lasso the most stable performance.

1 Introduction to the problem

NUON is the biggest wind energy producer in The Netherlands. Approximately 50% of the 600MW installed wind energy in The Netherlands is managed by NUON. The produced energy is sold 24 hours in advance on the energy market. It is therefore important to know the amount of energy that will probably be produced, for differences between predicted and delivered amounts either cost money or result in a price that is sub optimal. The cost of power imbalance is around 6 Euro/MWh. This means that the energy produced by the wind parks is worth 6 Euro less than that of perfectly predictable energy sources.

For a part of the total wind park, hourly data is available for the last two years. Predictions are based on predicted wind speeds and wind directions for different locations in the Netherlands. These weather predictions are also available 24 hours in advance. Because of the inherent error in these data, it will be impossible to create a perfect forecast.

This paper presents an attempt to improve the current predictions of the energy amount produced by the wind parks.

2 Previous work

There are not many articles concerning the application of machine learning techniques to the prediction of complete wind farms.

Li et al. [3] use neural networks for the prediction of the energy output for one turbine. Their results match the actual performance.
Denison et al. [2] predict the wind energy production on the longer term, based on historical data, by using a Bayesian multivariate adaptive regression spline model.
Nielsen and Nielsen [4] use statistical models to create production forecasts using meteorological data for wind farms. This led to the creation of WPPT.
Beyer et al. [1] use the numerical weather prediction model HIRLAM for wind parks in Germany. Error rates of 12 to 20% are reported.

Unfortunately, because of the lack of articles regarding the same type of problem, it is hard to compare the results. We have been able to measure the error on the same dataset using the current model in use at NUON. These results are described in the section of the current model.

3 Dataset description

The input features are the forecasted wind speed and wind direction of four locations in the Netherlands (DeKooy, Hoogeveen, Leeuwarden and Schiphol), as well as an average of the forecasted wind speed and wind direction at fifteen sites. These values are available 24 hours in advance from the national weather agency. The dataset consists of 18935 examples of data. This is the data for the period February 2001 to March 2003. Each example represents one hour of data in that year. Each example consists of ten input features and one output. The wind speed is measured in meters per second, discretized to 1 m/s intervals. The wind direction is measured in degrees. The locations of the forecasts and the locations of the wind parks are not the same. The output feature is the total energy output of a set of wind parks in the given hour.

Name	Output?	min	max	avg	st.dev
Direction NL	no	0.00	360.00	188.58	91.83
Direction DeKooy	no	0.00	360.00	186.81	95.41
Direction Hoogeveen	no	0.00	360.00	184.82	92.07
Direction Leeuwarden	no	0.00	360.00	185.79	94.49
Direction Schiphol	no	0.00	360.00	184.86	94.60
Speed NL	no	0.00	13.00	4.38	1.91
Speed DeKooy	no	0.00	15.00	5.73	2.42
Speed Hoogeveen	no	0.00	23.00	4.25	1.88
Speed Leeuwarden	no	0.00	21.00	4.63	2.01
Speed Schiphol	no	0.00	14.00	5.19	2.24
Production	yes	−72.00	84235.00	18366.91	19524.47

Correlations of the features show that the wind directions as well as speeds are highly correlated, which was expected, as the Netherlands is not a big country. The output is highly correlated with the wind speeds, and slightly with the wind directions.

Plots projecting each of the features against the another, show a seasonal pattern where the wind speed or energy drops during July and August (figure 1).

The wind speed is highest when the direction is around 230 degrees (figure 2 left). The plots for 'wind speed vs output' show that the wind speed is only an upper bound for the production (figure 2 right). For average wind speeds the production varies between zero and the upper bound. At high wind speeds the turbines have to be shut-down to prevent them from being damaged.

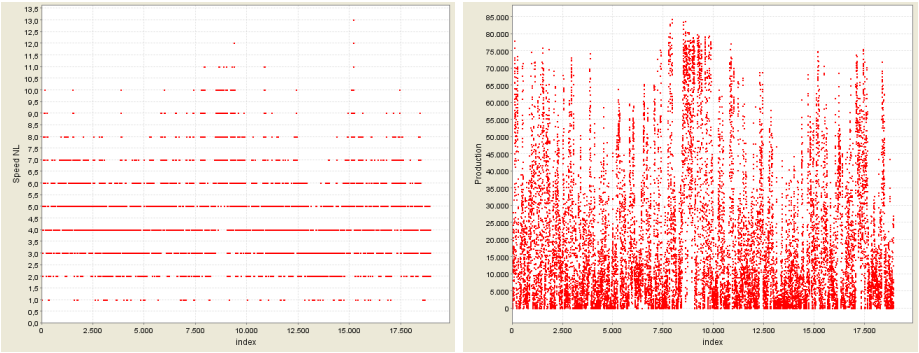

Figure 1: Distribution of wind speed and production over 2 years.

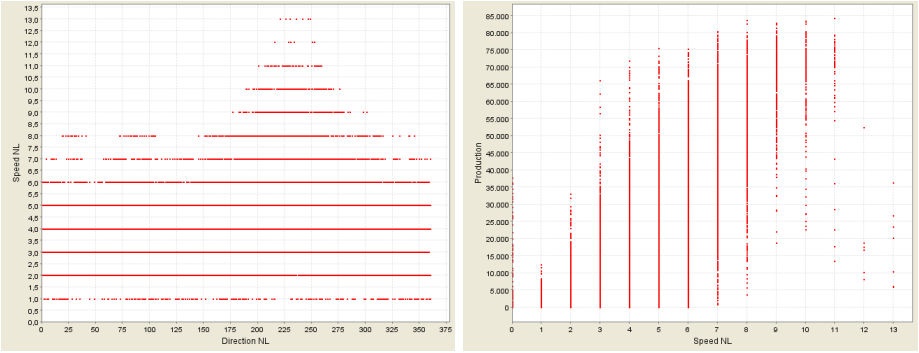

Figure 2: Left: wind direction vs wind speed, right: speed vs production

4 Our approach

This research was performed using the 'Workbench for Machine Learning Techniques' [5]. This is a tool that allows the user to easily create test scripts for testing machine learning techniques and applying these techniques to problems such as this wind energy prediction problem. These scripts consist of modules placed in a stackable architecture.

Each of the modules in the script modifies the dataset. The modified dataset is passed on to subsequent modules, or used to apply a machine learning algorithm to

a train and test set. The user-interface allows the parameters to the modules and learning techniques to be varied. The execution of the techniques is distributed over multiple computer systems. After the execution, different error measures are available. These measures can be set-off against the parameters using visualization modules. This allows us to quickly find optimal parameters.

Two machine learning techniques are used. The first is the k-nearest neighbour algorithm for regression problems. The algorithm is initialised with a set of examples x_i and their corresponding output y_i. The distance to all examples is calculated for each new vector:

$$d(x_i, x_j) = \sqrt{\sum_{m=1}^{N} (x_{im} - x_{jm})^2}$$

The examples are ordered according to this distance, and the k examples with the least distance are selected. The predicted value for the new vector is then either the average of these points, or a weighed sum where examples with a smaller distance contribute more to the predicted output.

The second technique is Lasso [6]. Lasso is based on relevence vector machines, which produce sparse models using kernel functions. It has the advantage that the error distributions for train set and test set are comparable. Predictions are calculated by

$$f(x) = \alpha_0 + \sum_{i=1}^{n} \alpha_i K(x_i, x)$$

in which α is a vector weights for each of the examples, and K is the kernel function. To obtain α, the problem

$$\begin{aligned} \text{minimise} \quad & ||y - 1\alpha_0 - \mathbf{K}\alpha||_2^2 \\ \text{subject to} \quad & ||\alpha||_1 \leq \text{kappa} \end{aligned}$$

is solved. Kappa controls the complexity of the solution. Two kernel functions were used; the radial basis function (RBF) kernel and the polynomial kernel.

$$\begin{aligned} \text{Polynomial} \quad & K(u, v) = (\text{gamma}\ u \cdot v + \text{coef})^d \\ \text{RBF} \quad & K(u, v) = e^{-(u-v)^2/\sigma^2} \end{aligned}$$

Three approaches are taken to come to improve the predictions that are currently obtained (section 4.1). The first approach was to use scaled data. The second modifies the wind directions in the dataset to a more natural system. The third approach extends the dataset even further by adding lags of the input features to the dataset.

The script used in the workbench consists of several modules. The first module in the script is the 'normalize' module. This module scales all inputs and the output to mean 0 and standard deviation 1. The reason to do so is that most techniques, including k-nearest neighbours and Lasso, work better if these values are close to zero. After this scaling all values lie between -1 and 3.

The second module is a 'random subset' module. This module picks a random set of examples from the dataset. The size of this set was set to 30%. The reason to do so is that the dataset contains a large number of examples, of which many are similar. This slows down the training while the results are presumed to be similar when compared to training on the complete dataset. During the tests, this presumption will be tested.

The third module is a cross-validation module that creates four folds. This allows us to see if there are large variations in the results.

The fourth and final module is the machine learning technique. For the first attempt, this is initially the k-nearest neighbour module, where k can be varied, and has two weighing modes. Thereafter the Lasso module is used, which allows kappa to be varied, as well as choosing a kernel function and its parameters.

4.1 Current model

The current model in use at NUON is a second order polynomial. This polynomial is fit on the complete dataset to obtain the least mean squared error solution.

Using this model, predictions for all examples in the dataset were obtained. This results in a mean absolute error of 0.44 and a mean squared error of 0.36 between the actual and predicted values.

4.2 Basic approach

The first attempt to improve the current model is by using k-nearest neighbours. The workbench was given a script that modified the dataset (normalize all features and output), and to search for the best settings. These were determined to be at $k = 6$ with the nearest neighbours being weighted by their distance to the given example and then summed. The mean squared error for these settings was 0.04 and on the test set 0.29. With these settings fixed, the test was repeated with the complete dataset. The error dropped to the score given in table 1.

Error measure	Average	Min	Max
MSE train set	0.03	0.03	0.03
MSE test set	0.23	0.22	0.24
MAE train set	0.11	0.11	0.11
MAE test set	0.32	0.31	0.32

Table 1: Error measures for k-nearest neighbours on scaled dataset

Error measure	Average	Min	Max
MSE train set	0.29	0.29	0.29
MSE test set	0.30	0.29	0.31
MAE train set	0.39	0.38	0.39
MAE test set	0.39	0.39	0.40

Table 2: Error measures for Lasso with polynomial kernel on scaled dataset

The next tests are performed using Lasso. They are based on the same script in the workbench, with the k-nearest neighbour module being replaced by the Lasso module. Two kernels were tried: the RBF and the polynomial kernel.

The RBF kernel was tried for different values of kappa and sigma. In the Lasso algorithm, kappa is the upper bound for the L1 norm of the active set. For

different values for kappa, there were always some values of sigma that performed best. The differences between these pairs however are negligible. For both train and test set, the MSE varied around 0.31.

The polynomial kernel was tried for different degrees, and different values for gamma and the coefficient, in addition to kappa. The polynomial of the fourth degree was the most effective. For parameters kappa=0.7, gamma=0.3 and coefficient=3.0 the best results were obtained (table 2).

When the amount of examples that is allowed in the dataset is increased, these values do not change.

4.3 Modified wind directions

The wind directions varied over 0 to 360 (before scaling them around zero). The jump from 360 to 0 is quite significant, whereas in reality there is no distinction. To clear this problem, the directions were all mapped onto a circle. That way, the values become better comparable. The original wind directions are then removed from the dataset, and sine and cosine values of the mappings are added.

To measure the results, the polynomial kernel was used, again with degree 4. The initial results (with all settings kept the same) improved over the previous test.

Error measure	Average	Min	Max
MSE train set	0.26	0.25	0.28
MSE test set	0.27	0.27	0.27
MAE train set	0.37	0.36	0.38
MAE test set	0.37	0.37	0.38

Table 3: Error measures for Lasso with polynomial kernel on dataset with modified wind directions

Error measure	Average	Min	Max
MSE train set	0.25	0.24	0.26
MSE test set	0.29	0.26	0.31
MAE train set	0.36	0.36	0.37
MAE test set	0.39	0.37	0.39

Table 4: Error measures for Lasso with polynomial kernel on dataset with modified wind directions and a lag of one time step

When the number of examples that was used was increased to 70% the numbers did not change. By varying kappa, gamma and coefficient, it was found that the parameters were still optimal.

4.4 Adding lags

Because of the discretized wind speed values, we assume that better predictions are obtained by incorporating lags of previous wind speeds. The script was modified so that after the mapping of directions onto circles all input features were lagged for periods 1, 2 and 5 time steps. For lag=5, this resulted in a dataset consisting of 90 input features (for each of the five locations, the wind speed and the sine and cosine of the direction are available, and these values are repeated for t-1, t-2, t-3, t-4 and t-5).

The tests were performed with the same settings as for the previous test. The results are summarized in tables 4, 5 and 6.

Error measure	Average	Min	Max
MSE train set	0.26	0.26	0.27
MSE test set	0.31	0.29	0.32
MAE train set	0.37	0.36	0.37
MAE test set	0.39	0.38	0.40

Table 5: Error measures for Lasso with polynomial kernel on dataset with modified wind directions and a lag of two time steps

Error measure	Average	Min	Max
MSE train set	0.25	0.24	0.26
MSE test set	0.30	0.28	0.32
MAE train set	0.36	0.36	0.37
MAE test set	0.39	0.38	0.40

Table 6: Error measures for Lasso with polynomial kernel on dataset with modified wind directions and a lag of five time steps

Because of the good initial results with k-nearest neighbours and the varying results with Lasso, an additional test was performed with the kNN module, with lag=5 on 30% of the examples. This resulted in the measures in table 7.

Error measure	Average	Min	Max
MSE train set	0.05	0.04	0.05
MSE test set	0.24	0.23	0.24
MAE train set	0.14	0.14	0.14
MAE test set	0.34	0.33	0.34

Table 7: Error measures for kNN on dataset with modified wind directions and a lag of five time steps

Error measure	Average	Min	Max
MSE train set	0.03	0.03	0.04
MSE test set	0.20	0.19	0.20
MAE train set	0.12	0.12	0.12
MAE test set	0.29	0.29	0.30

Table 8: Error measures for kNN on dataset with modified wind directions and a lag of one time step on the complete dataset

As results for kNN improved as more examples were used for training, the test is repeated with all examples, and a lag of one time step. These results are summarized in table 8.

5 Conclusion

Two techniques were applied to the NUON wind speed dataset. These techniques were k-nearest neighbours and Lasso. Successive improvements were made to the dataset to obtain better results.

The k-nearest neighbour technique was able to obtain very good results. This is probably due to the huge amount of examples. Good examples are always near and the pattern is obviously stable.

Lasso is a sophisticated technique. From the comparison with the kNN technique, it seems that modelling the results of the predictions on the energy output

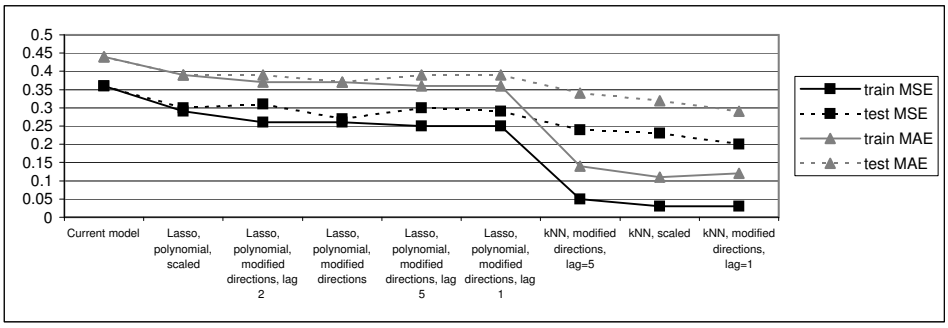

Figure 3: Results of the tests

is hard. The advantage is that the performance is more stable that that of the kNN technique.

Conversion of the wind directions to a more natural format improved the predictions. Results for adding lags vary. Apparently the addition adds less information to the data than the increase in complexity is able to justify. Adding a small amount of lag improves the results (minimum average MSE and MAE drop slightly), but the average errors on the test set increase.

The use of Lasso with a fourth degree polynomial kernel results in average errors of 9-10%. By using k-nearest neighbours with distance based weighing of the 6 nearest neighbours the average error is expected to be in the range 3-8%.

References

[1] H.G. Beyer, D. Heinemann, H. Mellinghoff, K. Monnich, and H. Waldl. Forecast of regional power output of wind turbines, 1999.

[2] D.G.T. Denison, P. Dellaportas, and B.K. Mallick. Wind speed prediction in a complex terrain, 2001.

[3] S. Li, D.C. Wunsch, E.A. O'Hair, and M.G. Giesselmann. Using neural networks to estimate wind turbine power generation energy conversion. *IEEE Transaction on energy conversion*, 16(3):276–282, September 2001.

[4] H. A. Nielsen and T. S. Nielsen. Using meteorological forecasts in on-line predictions of wind power. chapter 5: Estimation methods, 1999. ISBN: 87-90707-18-4.

[5] F.A. Ouwendijk. Workbench for machine learning techniques. Master's thesis, Delft University of Technology, june 2003.

[6] V. Roth. Sparse kernel regressors. In Georg Dorffner, Horst Bischof, and Kurt Hornik, editors, *Proceedings of the International Conference Vienna (ICANN'01)*, volume 2130 of *Lecture Notes in Computer Science*, pages 339–346. Springer-Verlag, 2001.