Assignment 2 - Machine Learning for Wind Energy Forecasting

INF5870 - group 5

Sharanan Kulam - sharanak@ifi.uio.no
Tanusan Rajmohan - tanusanr@ifi.uio.no
Johan Paramanathan - johanpar@ifi.uio.no
Birashanthan Tharmakulasingam - birashat@ifi.uio.no

UNIVERSITY OF OSLO

Spring 2018

Contents

Task 1

Error metrics

Model	MSE	RMSE
Linear Regression	0.04682	0.21638
K-nearest neighbours	0.04678	0.21628
Support vector regression	0.04569	0.21375
Neural network	0.04677	0.21627

(1.1)

From the figure (1.1) we can see that the majority of the models have very small deviations between the RMSE. An important aspect regarding this may be that we are only looking at the correlation between two variables, where the deviation in the prediction accuracy may not differ that much.

Comments

We can also see that the SVR is the optimal method to apply here. One reason for this is that by adjusting the ϵ margin, we set the model to have some slack so that only out-of-margin errors are taken into consideration as errors. Everything within the margin is considered a correct prediction. Comparing this approach to LR, KNN and NN we can see that these models compute the absolute error for all samples, and does not include some slack or range of predictions that are accepted as correct.

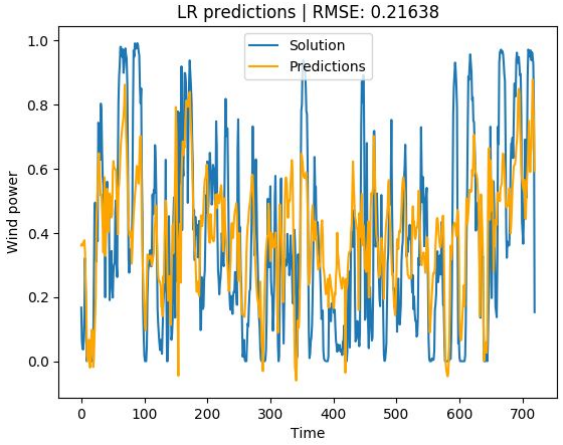
When running the code multiple times, we observed that the results changed for the neural network, with smaller deviations at each run. Compared to the other methods, the function approximation (prediction line) is not deterministic. The primary reason for this is that neural network parameters, such as weights are often initialized randomly or follows some initialization scheme. There may, therefore, be different sets of weights for a given optimal solution.

These metrics were extracted from our initial run without tuning available parameters for each of the models, except KNN where we used cross-validation to find the optimal K. Otherwise, for SVR and NN there are different parameters that can be adjusted that potentially affect the result. For SVR, we may for instance tune the ϵ margin (default 0.1) and for the NN we can adjust parameters such as the learning rate, hidden layers or try different activation functions and optimizers.

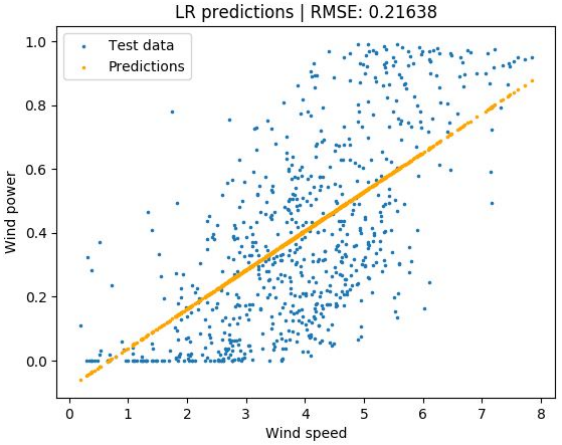
Figures

For task 1 we have generated two figures for each method. As described in the assignment, we have included the comparison figures related to the true wind energy measurements and the predicted results over time. Additionally, we've also included the figures of the regression curve. This way, we can not only observe the accuracy of the figures but also how the model was fit.

Linear Regression

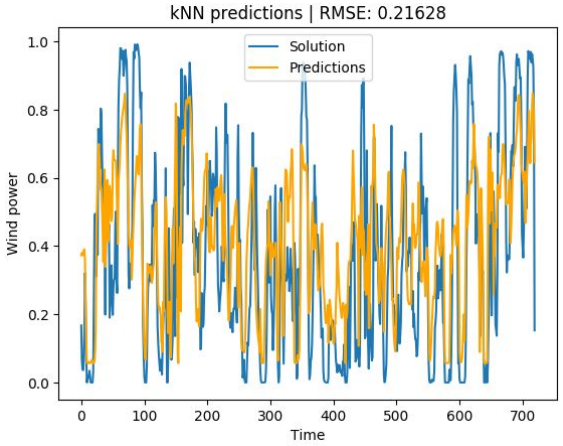


(1.2)

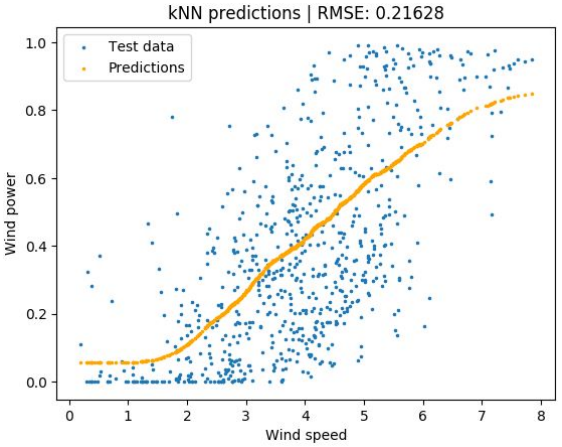


(1.3)

K-nearest neighbors | k = 1634 using cross-validation

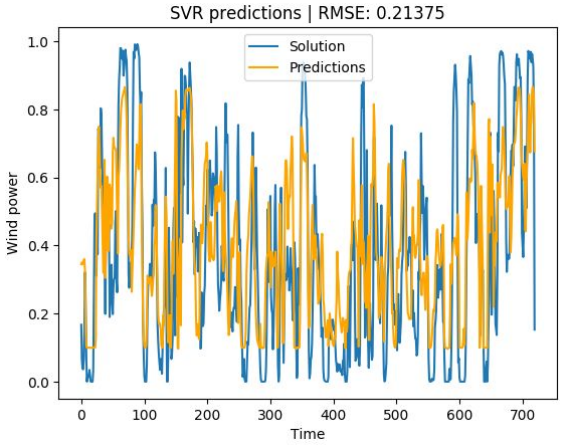


(1.4)

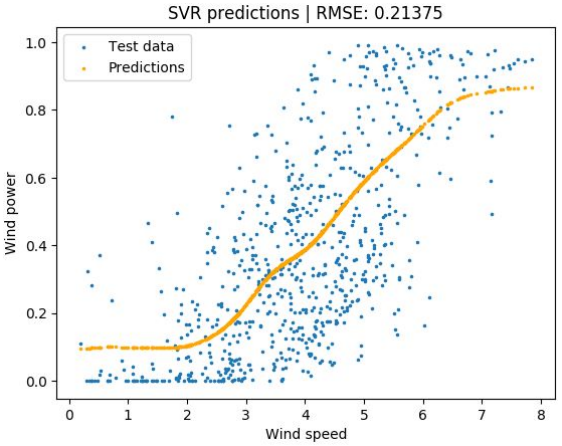


(1.5)

Support vector regression

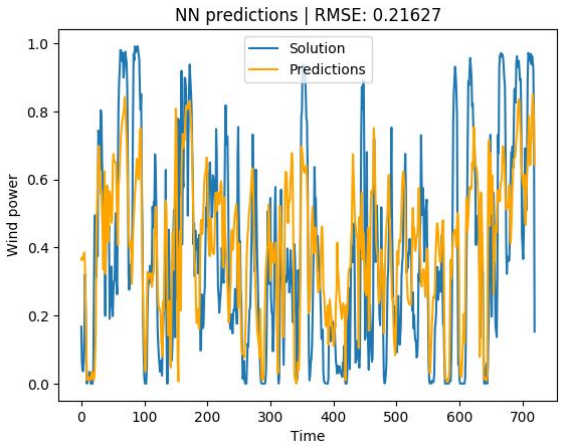


(1.6)

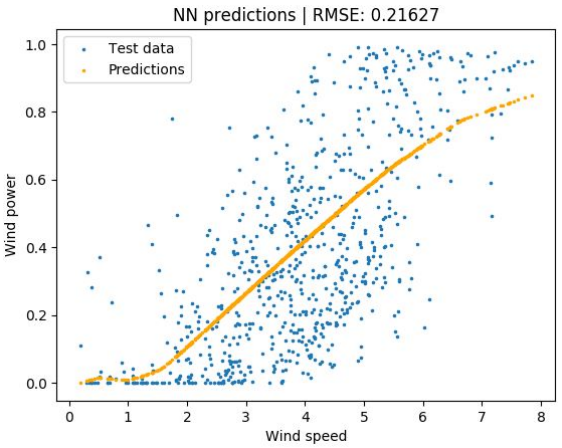


(1.7)

Neural network



(1.8)



(1.9)

Task 2

Error metrics

Comparison of LR and MLR across multiple variables			
Model	**Variables**	**MSE**	**RMSE**
Linear Regression	Wind direction	0.08719	0.29528
Linear Regression (task 1)	Wind speed	0.04682	0.21638
Multiple Linear Regression	Wind direction and wind speed	0.04647	0.21556

(2.1)

Comments

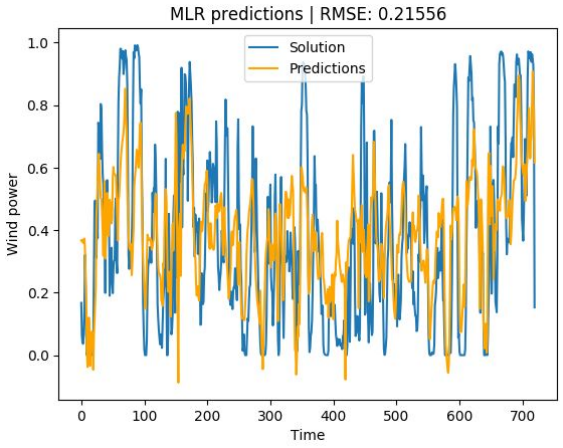
We can observe that the RMSE is higher when only looking at the correlation between wind direction and wind power separately. Compared to wind speed there is a distinct difference. A good way to understand the correlation is to plot the data and analyze the results. When using MLR, wind direction has a small influence on the RMSE, but it is not significant. Further on, we will look at how much wind direction actually influences the power generation.

Figures

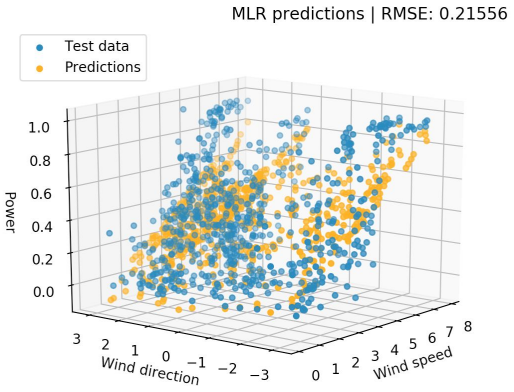
Similar to task 1, we have included two figures, one for the prediction and one to observe the model fit. A good way to illustrate an MLR, especially with 2 predictors, is to analyze the model fit in a 3D space. Regardless, by only observing the 3D plot there is no possibility of extracting useful information from it. Because of this, we have also analyzed the independent relation between wind power/wind direction and wind power/wind speed with linear regression. This way, we can understand which of these variables correlates the most.

One shortcoming of the 3D plots is that the model fit is plotted as points rather than a plane, which would be more intuitive and interpretable. However, due to limited time, we decided this was the best option.

Multiple Linear Regression with wind direction and wind speed as predictors



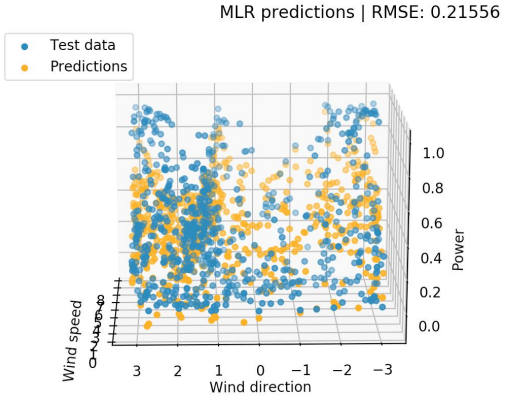
(2.2)



(2.3)

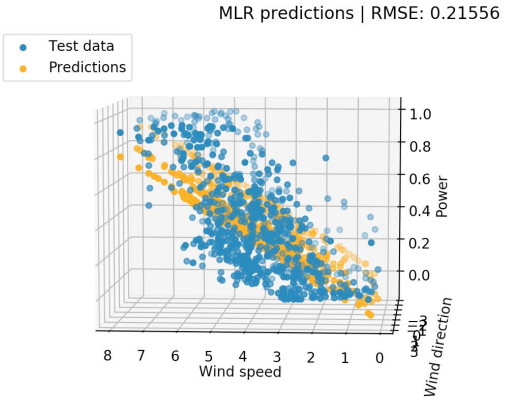
We can see here, that regardless of the direction in wind, it does not affect the power generation significantly. The wind speed has a better influence as a predictor, and we can see an increasing trend in power generation that is positively correlated with wind speed. Figure 2.4 and 2.5 gives a better overview when observing the 3D plots from different angles.

Observing correlation between wind direction and wind power



(2.4)

Observing correlation between wind speed and wind power

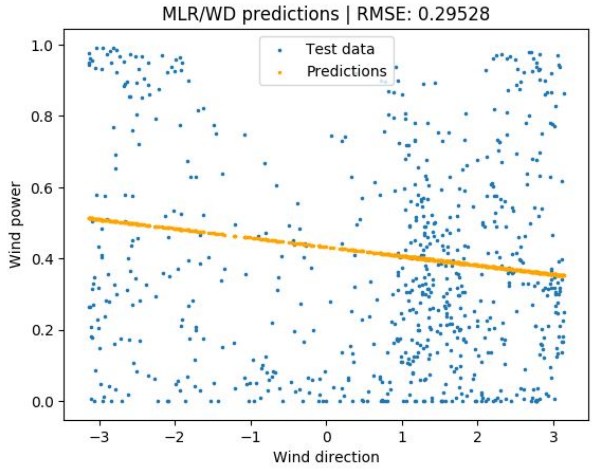


(2.5)

As described above, a better way of understanding the correlations is to look at each of the independent variables in relation to the wind power. We already know how the regression line fit for wind speed as predictor (derived from task 1). By looking at figure 2.6 however, we can also look at how wind direction correlates to the wind power.

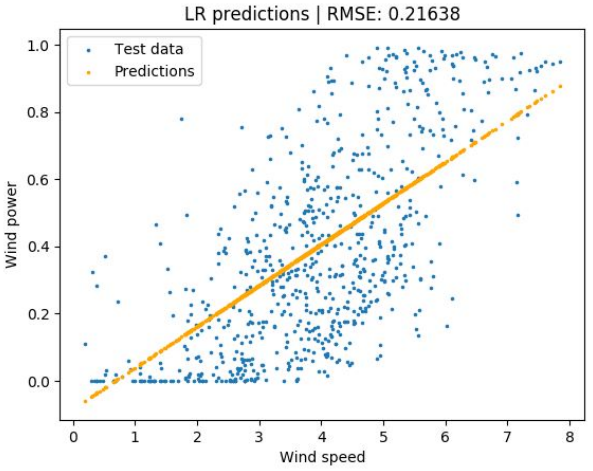
There are two clear distinctions here, and we can see that the deviation in power is very small when using wind direction as predictor, compared to wind speed in figure 2.7, thus minimal overall influence.

Linear Regression with wind direction as the predictor



(2.6)

Linear Regression with wind speed as comparison (task 1)



(2.7)

Comments on correlations

We have analyzed both variables independently using linear regression and collectively using multiple linear regression, and concluded that wind power has little or no influence on the power generation at all. However, another approach to understanding the correlations between variables is numerical, when computing what is known as the correlation coefficient. The coefficient ranges from -1 to 1, for strongest possible agreement and disagreement (positive and negative correlation) respectively[1]. The *Pandas* library in Python provides the opportunity to compute these coefficients, and generating the correlation matrix and extracting the coefficients for wind direction and wind speed, we get the following values:

Variables	Correlation with power
Wind Speed (W10)	0.727077
Wind Direction	-0.135124

(2.8)

As we can see here wind speed has a very positive correlation (strong agreement), compared to the wind direction, which has a slightly negative correlation or almost no correlation at all.

[1] See definition on correlation coefficient from NCME (National Council on Measurement in Education) in the references.

Task 3

Error metrics

Model	MSE	RMSE
Recurrent Neural Network	0.08837	0.29728
Linear Regression	0.08273	0.28763

(3.1)

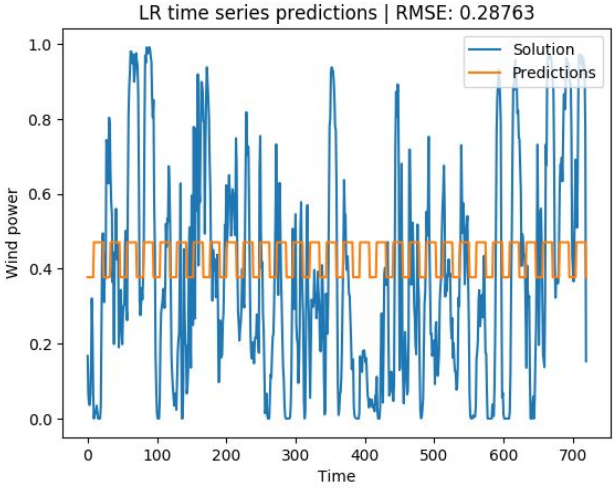
Comments

As seen in 3.1, we were not able to get the best MSE or RMSE values. We only use it to find a coefficient, that overall can predict the course of the values. Taking this into account and assuming that the 24-hour progression of time, there was bound to some deviation. With trial and error in the implementation, we found out that the dates resulted in larger deviation because the dates were not encoded numerically. To solve this we implemented a function to encode the date format. From the initial error metrics the RMSE results have been lowered for both LR and RNN around 0.1.

The differences between the RMSE values for LR and RNN is marginal. LR might perform better because it takes less variables into account when doing the calculations, compared to RNN. RNN does several tunings in the hidden layer when calculating. With this in mind, having less data to use (compared to task 1 and 2) gives a greater probability for deviations. The difference between LR and RNN can also be explained by the way we encode and format the data. RNN is generally for time series and sequence modeling to find an optimal way to discover patterns and correlations over time. However, different sequences will result in different accuracies and we have justified our formatting in the section below.

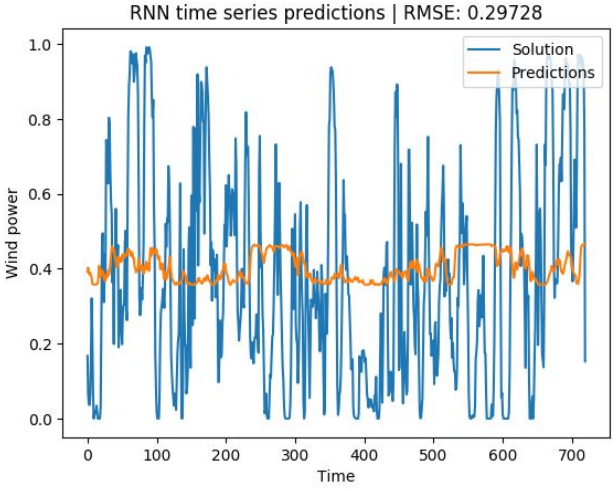
Figures

Time series with LR



(3.2)

Time series with RNN



(3.3)

Comments on data formatting

Time series with LR

We used the whole training set as input, because LR does not require a lot of computing compared to RNN. The frequency of the predictions seems to try to replicate the changes in different days, thus making a wave like constant pattern.

In LR we read the *TrainData.csv* file and the *ForecastTemplate.csv* for input and output. We used the *ForecastTemplate.csv* to write the predicted values for each of the methods (LR and RNN). The *TrainData.csv* file was used to actually compute the predictions, in LR we converted the timestamps, just to make sure it was in the correct format. Without the numeric format the linear model would not be created. When we took the data raw from the csv files, the program just kept running for a long time without any output. We reconstructed the timestamps to a numeric value before using them in the linear model to predict the values for the given month, November. We also tried to use different time periods from the *TrainData.csv* file but did not get any better results or lower RMSE, which is why we chose the method and RMSE value we got.

In RNN the format was not changed, here we only used the power values as we read them from the *TrainData.csv* file.

Time series with RNN

For RNN the data would be pre-processed in terms of finding an optimal sequence, and afterwards processed by the network over total epochs which is given. With this in mind, the size of each sequence should not be larger than necessary. As overdoing it in terms of too much data or dimensions, or even epochs resulted in excruciating long runtimes. Additionally, it would also be run with specified number of hidden dimensions. Based on the number of tunable parameters, we experienced a lot of different values and bad predictions. To then try to predict best as possible, we sat out to try different sets of the data we had. Through extensive trial and error, we managed to find a good range of data that responded well with the input we had.

The parameters that seemed optimal, was one hidden dimension as we did not see a decrease of RMSE when using more hidden units (there were some deviations in the predictions, but the RMSE was higher). What we assumed was the cause of this is that minor deviations or errors from a prediction from one timestep to the next had larger implications. This was concluded after discussing with the group teachers as chaos theory and butterfly effect is considered to be reasons for deviation.

The learning rate was set to 0.01 which was found through cross validation. The same approach was used for finding the number of epochs, which was set to 20.

Sources

- LR in R: https://machinelearningmastery.com/linear-regression-in-r/
- RNN documentation: https://cran.r-project.org/web/packages/rnn/vignettes/rnn.html
- Correlation definition: http://www.ncme.org/ncme/NCME/Resource_Center/Glossary/NCME/Resource_Center/Glossary1.aspx?hkey=4bb87415-44dc-4088-9ed9-e8515326a061#anchorC
- Linear Model used for time series http://stat.ethz.ch/R-manual/R-devel/library/stats/html/predict.lm.html
- Machine Learning for Renewable Energy Forecasting lecture: http://folk.uio.no/yanzhang/INF5870-2018/windsim.pdf
- Deep Learning for Renewable Energy Forecasting lecture: http://folk.uio.no/yanzhang/INF5870-2018/DeepLearningforEnergyForecasting-Lecture10.pdf