Renewable Energy 50 (2013) 236–243

Contents lists available at SciVerse ScienceDirect

Renewable Energy

journal homepage: www.elsevier.com/locate/renene

Predicting the energy output of wind farms based on weather data: Important variables and their correlation

Ekaterina Vladislavleva [a], Tobias Friedrich [b], Frank Neumann [c,*], Markus Wagner [c]

[a] Evolved Analytics Europe BVBA, Veldstraat 37, 2110 Wijnegem, Belgium and Faculty of Computer Science and Engineering Science, Cologne University of Applied Sciences, Steinmüllerallee 1, 51643 Gummersbach, Germany
[b] Friedrich-Schiller-Universität Jena, Ernst-Abbe-Platz 2, 07743 Jena, Germany
[c] School of Computer Science, University of Adelaide, Adelaide, SA 5005, Australia

ARTICLE INFO

Article history:
Received 26 October 2011
Accepted 18 June 2012
Available online 24 July 2012

Keywords:
Wind energy
Prediction
Genetic programming
DataModeler

ABSTRACT

Wind energy plays an increasing role in the supply of energy world wide. The energy output of a wind farm is highly dependent on the weather conditions present at its site. If the output can be predicted more accurately, energy suppliers can coordinate the collaborative production of different energy sources more efficiently to avoid costly overproduction. In this paper, we take a computer science perspective on energy prediction based on weather data and analyze the important parameters as well as their correlation on the energy output. To deal with the interaction of the different parameters, we use symbolic regression based on the genetic programming tool DataModeler. Our studies are carried out on publicly available weather and energy data for a wind farm in Australia. We report on the correlation of the different variables for the energy output. The model obtained for energy prediction gives a very reliable prediction of the energy output for newly supplied weather data.

1. Introduction

Renewable energy, such as wind and solar energy, plays an increasing role in the supply of energy world wide. This trend will continue because global energy demand is increasing, and the use of nuclear power and traditional sources of energy such as coal and oil is either considered unsafe or leads to a large amount of CO_2 emission.

Wind energy is a key player in the field of renewable energy. The capacity of wind energy production has been substantially increased during the last years. In Europe, for example, the capacity of wind energy production has doubled from 2005 to 2007 [13]. However, levels of production of wind energy are hard to predict as they rely on potentially unstable weather conditions present at the wind farm. In particular, wind speed is crucial for energy production based on wind, and it may vary drastically over time. Energy suppliers are interested in accurate predictions, as they can avoid overproduction by coordinating the collaborative production of traditional power plants and weather-dependent energy sources.

Our aim is to map weather data to energy production. We wish to show that even data that is publicly available for weather stations close to wind farms can be used to give a good prediction of the energy output. Furthermore, we examine the impact of different weather conditions on the energy output of wind farms. We are particularly, interested in the correlation of different components that characterize weather conditions such as wind speed, pressure, and temperature.

A good overview on the different methods that were recently applied in forecasting of wind power generation can be found in [3]. Statistical approaches use historical data to predict the wind speed on an hourly basis or to predict energy output directly. On the other hand, short term prediction is often done based on meteorological data, and learning approaches are applied. Kusiak, Zheng, and Song [9] have shown how wind speed data may be used to predict the power output of a wind farm based on time-series prediction modeling. Neural networks are a very popular learning approach for wind power forecasting based on given time series. They provide an implicit model of the function that maps the given weather data to an energy output.

Jursa and Rohrig [4] have used particle swarm optimization and differential evolution to minimize the prediction error of neural networks for short-term wind power forecasting. Kramer and Gieseke [8] used support vector regression for short-term energy forecast and kernel methods and neural networks to analyze wind energy time series [7]. These studies are all based on wind data and do not take other weather conditions into account. Furthermore, neural networks have the disadvantage that they give an implicit model of the function predicting the output, and these models are rarely accessible to a human expert. Usually, one is also interested in

* Corresponding author.
E-mail address: frank.neumann@adelaide.edu.au (F. Neumann).

0960-1481/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.renene.2012.06.036

the function itself and the impact of the different variables that determine the output. We aim to study the impact of different variables on the energy output of the wind farm. Surely, the wind speed available at the wind farm is a crucial parameter [1,12]. Other factors that influence the energy output are, for example, air pressure, temperature and humidity. Our goal is to study the impact and correlation of these parameters with respect to the energy output.

Genetic programming (GP) (see [10] for a detailed presentation) is a type of evolutionary algorithm that can be used to search for functions that map input data to output data. It has been widely used in the field of symbolic regression and the goal of this paper is to show how it can be used for the important real-world problem of predicting energy outputs of wind farms from weather data. The advantage of this method is that it comes up with an *explicit* expression mapping weather data to energy output. This expression can be further analyzed to study the impact of the different variables that determine the output. To compute such an expression, we use the tool DataModeler [2], which is the state of the art tool for doing symbolic regression based on genetic programming. We will also use DataModeler to carry out a sensitivity analysis which studies the correlation between the different variables and their impact on the accuracy of the prediction.

We proceed as follows. In Section 2, we give a basic introduction into the field of genetic programming and symbolic regression, and describe the DataModeler. Section 3 describes our approach of predicting energy output based on weather data and in Section 4 we report on our experimental results. Finally, we finish with some concluding remarks and topics for future research.

2. Genetic programming and DataModeler

Genetic programming [6] is a type of evolutionary algorithm that is used in the field of machine learning. Motivated by the evolution process observed in nature, computer programs are evolved to solve a given task. Such programs are usually encoded as syntax expression trees. Starting with a given set of trees called the population, new trees called the offspring population are created by applying variation operators such as crossover and mutation. Finally, a new parent population is selected from among the previous parents and the offspring based on how well these trees perform for the given task.

Genetic programming has its main success stories in the field of symbolic regression. Given a set of input output vectors, the task is to find a function that maps the input to the output as best as possible, while avoiding over fitting. The resulting function is later often used to predict the output for a newly given input. Syntax trees represent functions in this case, and the functions are changed by crossover and mutation to produce new functions. The quality of a syntax tree is determined by how well it maps the given set of inputs to their corresponding outputs.

The task in symbolic regression can be stated as follows. Given a set of data vectors $(x_{1i}, x_{2i}, \cdots, x_{ki}, y_i) \in \mathbb{R}^{k+1}$, $1 \leq i \leq n$, find a function $f : \mathbb{R}^k \to \mathbb{R}$ such that the approximation error, e.g. the root mean square error

$$\sqrt{\frac{\sum_{i=1}^{n}(y_i - f(x_i))^2}{n}}$$

with $x_i = (x_{1i}, x_{2i}, \cdots, x_{ki})$, is minimized.

We chose to use a tool called DataModeler for our investigations. It is based on genetic programming and designed for solving symbolic regression problems.

2.1. DataModeler

Evolved Analytics' DataModeler is a complete data analysis and feature selection environment running under Wolfram Mathematica 8. It offers a platform for data exploration, data-driven model building, model analysis and management, response exploration and variable sensitivity analysis, model-based outlier detection, data balancing and weighting.

Data-driven modeling in DataModeler is done by symbolic regression via genetic programming. The Symbolic Regression function offers several evolutionary strategies which differ in the applied selection schemes, elitism, reproduction strategies, and fitness evaluation strategies. An advanced user can take full control over symbolic regression and introduce new function primitives, new fitness functions, selection and propagation schemes, etc., by specifying appropriate options in the function call. However, we used the default settings and the default evolution strategy, which in DataModeler is called ClassicGP.[1]

In the symbolic regression performed here, a population of individuals (syntax trees) evolves over a variable number of generations at the Pareto front in the three dimensional objective space of model complexity, model error, and model age [5,11].

Model error in the default setting ranges between 0 and 1, with the best value being 0. It is computed as $1-R^2$, where R is a scaled correlation coefficient. The correlation coefficient of the predicted output is scaled to have the same mean and standard deviation as the observed output.

The model complexity is the expressional complexity of models, and it is computed as the total sum of nodes in all subtrees of the given GP tree. The model age is computed as the number of generations that the model survived in the population. The age of a child individual is computed by incrementing the age of the parent contributing to the root node of the child. We use the age as a secondary optimization objective, as it is used only internally for evolution. At the end of symbolic regression runs, results are displayed in the two-objective space of user-selected objectives, in our case, these objectives are model expressional complexity and $1-R^2$.

The population-specific parameters of our genetic programming system are chosen as follows. The default population size is 300. The default elite set size is 50 individuals from the 'old' population closest to the 3-dimensional Pareto front in the objective space. These individuals are copied to the 'new' population of 300 individuals, after which the size of the new population is decreased down to the necessary 300. This is done by selecting models from the Pareto layers until the initially specified amount of models is found.

The selection of individuals for propagation is done by means of Pareto tournaments. By default, 30 models are randomly sampled from the current population, and Pareto-optimal individuals from this sample are determined as winners to undergo variation until a necessary number of new individuals are created.

Models are coded as parse trees using the GPmodel structure, which contains placeholders for information about model quality, data variables and ranges used to develop the model, and some settings of symbolic regression. For example, the internal GPmodel representation of the first Pareto front model from a set of models from Fig. 3 with an expression $-25.2334 + 3.21666\ \text{windGust}_2$ is presented in Table 1. Note that the first vector inside the GPmodel structure represents model quality. Model complexity is 11, model error is 0.300409. The parse tree of the same model is plotted in Fig. 1.

When a specified execution threshold of a run in seconds is reached, the independent evolution run terminates and a vector of model objectives in the final population is re-evaluated to contain only model complexity and model error. The set of models can

[1] All models reported in this paper were generated using two calls of Symbolic Regression with only the following arguments: input matrix, response vector, execution time, number of independent evolutions, an option to archive models with a certain prefix-name, and a template specification.