International Journal of Forecasting 30 (2014) 395–401

Contents lists available at ScienceDirect

International Journal of Forecasting

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

A feature engineering approach to wind power forecasting GEFCom 2012

Lucas Silva

DTI Sistemas, Belo Horizonte, Brazil

ARTICLE INFO

Keywords:
GEFCom
Feature engineering
Gradient boosted decision trees
Linear regression
Machine learning
Time series

ABSTRACT

This paper provides detailed information about team Leustagos' approach to the wind power forecasting track of GEFCom 2012. The task was to predict the hourly power generation at seven wind farms, 48 hours ahead. The problem was addressed by extracting time- and weather-related features, which were used to build gradient-boosted decision trees and linear regression models. This approach achieved first place in both the public and private leaderboards.

1. Introduction

The "GEFCom 2012—Wind Forecasting" competition[1] posed the challenge of forecasting the hourly wind power generation for seven wind farms. A dataset containing historical power measurements for these wind farms, as well as meteorological forecasts of the wind components at the level of those wind farms, was provided. A detailed description of the dataset is provided by Hong, Pinson, and Fan (2013).

1.1. Challenges

A big challenge in this task was dealing with the time series nature of the dataset. Since no time series specific model was used, it had to be kept constantly in mind.

Another difficulty was in creating efficient features. Feature creation is one of the most important steps when solving a supervised learning problem. In order to do this, many features were derived from the dataset. The feature creation step had two main guiding principles:

1. Model the wind power generation equation, based on constants, the wind strength and direction, and the air density (surrogated). These features represent the windmill behavior.
2. Discover the relationship between the wind power generation at T and $T \pm n$. The objective here was to reflect the time series nature of the dataset, since the modeling techniques are not time series specific.

1.2. Dataset

As can be seen, the data provided (see Hong et al., 2013) consisted only of a series of hourly wind power generation values for each farm, and the forecasted wind strength and direction, issued every 12 h. This was done to mimic the real operation conditions, and did not include all of the desired information, such as the forecasted temperature and air density. Thus, to make it up for this missing information, many surrogate features were created, as will be explained later. Also, an important step was to build a consistent validation set using only the training period. This validation set allowed the construction of a model that did not overfit the training data.

1.3. Techniques

In this work, three main machine learning techniques have been used, all of them in the R statistical environment. Table 1 gives a brief description of how each algorithm was employed.

1.3.1. Linear regression

The linear regression method has been used in this work as a pre-processing step for combining the wind strength

E-mail address: lucas.eustaquio@gmail.com.

[1] http://www.kaggle.com/c/GEF2012-wind-forecasting.

0169-2070/$ – see front matter
http://dx.doi.org/10.1016/j.ijforecast.2013.07.007

Table 1
Techniques used.

Technique	Used for
Multiple linear regression (Geyer, 2003)	Influence of the wind strength and direction components for each farm Ensemble Post process to smooth the predictions (high frequency filter)
K-MEANS[a]	Similarity model to detect farm and overall behavior
GBM[b]	Models by farm Models by time slot Overall models

[a] R K-Means package, http://stat.ethz.ch/R-manual/R-devel/library/stats/html/kmeans.html.
[b] gbm: Generalized boosted regression models. http://cran.r-project.org/web/packages/gbm/index.html.

and direction components in one single feature. It has also been used to combine the models trained with GBM, and finally as a post-processing step to smoothen the moving average of the predicted values.

The linear regression attempts to model the relationship between two or more explanatory variables and a response variable by fitting a linear equation to the observed data.[2] Eq. (1) shows the formal definition:

$$y_i = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon, \quad (1)$$

where:

- $i = 1, \ldots, n$ is the training instance index;
- β_p are the regression coefficients;
- x_{ip} are the features; and
- ε is the residual error.

1.3.2. K-Means

In this work, K-Means was used to create similarity features. The objective of these features was to cluster instances that should have similar behaviors.

K-Means is a simple learning algorithm for clustering analysis. The goal of the K-Means algorithm is to find the best division of n entities into k groups, so that the total distance between the group's members and its corresponding centroid, representative of the group, is minimized. The pseudo-code for the K-Means algorithm is given in Fig. 1.[3]

1.3.3. GBM—generalized boosted models

GBM was used to train the intermediate models. The GBM package in the R environment with the Gaussian distribution function (Ridgeway, 2013) was used for this.

The R GBM package implements boosting for models, which is commonly used in statistics. The pseudo-code for GBM algorithm is shown in Fig. 2 (see Friedman, 2001). In Fig. 2,

- $F(x)$ is the function to minimize;
- $L(y, p)$ is the loss function;
- y is the output;
- p is the prediction;
- M is the number of trees; and
- $h(x; a)$ is a "weak learner" or "base learner", usually trees.

[2] http://www.stat.yale.edu/Courses/1997-98/101/linmult.htm.
[3] Data Mining Algorithms in R/Clustering/K-Means. http://en.wikibooks.org/wiki/Data_Mining_Algorithms_In_R/Clustering/K-Means.

Algorithm 1: K-Means Algorithm

Input: $E = \{e_1, e_2, \ldots, e_n\}$ (set of entities to be clustered)
k (number of clusters)
$MaxIters$ (limit of iterations)
Output: $C = \{c_1, c_2, \ldots, c_k\}$ (set of cluster centroids)
$L = \{l(e) \mid e = 1, 2, \ldots, n\}$ (set of cluster labels of E)

```
foreach c_i ∈ C do
    c_i ← e_j ∈ E (e.g. random selection)
end
foreach e_i ∈ E do
    l(e_i) ← argminDistance(e_i, c_j) j ∈ {1...k}
end

changed ← false;
iter ← 0;
repeat
    foreach c_i ∈ C do
        UpdateCluster(c_i);
    end
    foreach e_i ∈ E do
        minDist ← argminDistance(e_i, c_j) j ∈ {1...k};
        If minDist ≠ l(e_i) then
            l(e_i) ← minDist;
            changed ← true;
        end
    end
    iter++;
until changed = true and iter ≤ MaxIters ;
```

Fig. 1. K-Means algorithm.

Algorithm 1: Gradient_Boost

1. $F_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^{N} \Psi(y_i, \rho)$
2. For $m = 1$ to M do:
3. $\tilde{y}_i = -\left[\frac{\partial \Psi(y_i, F(\mathbf{x}_i))}{\partial F(\mathbf{x}_i)}\right]_{F(\mathbf{x})=F_{m-1}(\mathbf{x})}$, $i = 1, N$
4. $\mathbf{a}_m = \arg\min_{\mathbf{a},\beta} \sum_{i=1}^{N} [\tilde{y}_i - \beta h(\mathbf{x}_i; \mathbf{a})]^2$
5. $\rho_m = \arg\min_\rho \sum_{i=1}^{N} \Psi(y_i, F_{m-1}(\mathbf{x}_i) + \rho h(\mathbf{x}_i; \mathbf{a}_m))$
6. $F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \rho_m h(\mathbf{x}; \mathbf{a}_m)$
7. endFor

end Algorithm

Fig. 2. Boosting algorithm.

1.4. Approach overview

The basic framework of the approach is shown in Fig. 3. First, in the preprocessing step, features were created. Next, three types of models were trained using the processed data:

- models that were trained for each farm;
- models that were trained for each predicted time slot (1–3 h ahead, 4–6 h ahead, . . . , 45–48 h ahead); and
- overall models trained without splitting the samples (except for the cross-validation).

1.5. Organization of this paper

In this paper, the features and models used are described in detail. Section 2 describes the approach taken, explains the feature creation and modeling techniques