FORECASTING AVERAGE TOTAL LOAD ON THE ELECTRIC GRID VIA MACHINE LEARNING

LUKE BENNING

April 13, 2015

ABSTRACT. European transparency regulations require transmission system operators to publish data about the electricity market to foster an even playing field between all market makers. The available data allows for the maturation of the electricity market and encourages analysis of data to improve the generation, usage and management of electrical power. Our research specifically will be based upon the Elia grid dataset which gives the total load on the Elia electric grid measured in fifteen minute intervals for the past several years. In particular, our methods will use various discriminative and generative machine learning approaches to forecast the average total load on the Elia grid one day ahead of time. We will demonstrate that applying machine learning methods after preprocessing the load time series can yield a function with a low prediction error, capable of outperforming traditional econometric models such as auto-regression, moving average and exponential smoothing. These results will be practically beneficial as utilities can use the predicted values to generate an adequate amount of energy to avoid grid outages and electrical losses as well as construct dynamic pricing schemes based upon future load.

1. INTRODUCTION

The management of power systems is a complex task for transmission operators and is heavily reliant on knowledge of future energy demand. A model that can accurately forecast load is essential in energy generation as the predicted load can determine which devices should be operated in order to meet demand. Failure to generate an adequate amount of energy can lead to grid failures and conversely oversupply can lead to a waste of energy and resources. With the advent of decentralized electricity markets it is essential to develop accurate pricing protocols based on current demand for which an understanding of future demand is critical. Traditionally methods from classical statistics and econometrics such as regression and auto-regressive moving average have been used to project future electricity demand, yet in this paper we aim to use machine learning algorithms in an attempt to outperform these orthodox approaches.

Specifically the prediction task will be to forecast the average total load for the Elia electric grid one day ahead of time. The total load includes all of the electric loads on the Elia grid, including underlying distribution networks, as well as electrical losses. The power used for exports and energy storage are deducted from this value, and the total load value is therefore an estimation of the actual load on the Elia grid and the underlying networks. The total load value is measured at fifteen minute intervals everyday, yielding ninety-six values per day. The average total load for a particular day is then the average of the ninety-size total load values on that day. The objective is to predict these values before each day with as small an error as possible. To accomplish this we will construct statistical and machine learning based approaches that can forecast the average total load and test their performance for the year of 2014.

Our paper will be structured as follows. The motivation section will detail the importance of accurate forecasts of electric load, while related work will describe past research done with

predicting electricity load and summarize the results that have been obtained so far. The statistical preprocessing section will describe the forecasting problem we are confronted with and provide methods for scaling and de-trending the load time series. The algorithms section will describe three different machine learning algorithms and how they can be used in conjunction with the preprocessing steps to form a prediction model. The econometric benchmarks section will provide three models that will be used to gauge the accuracy of the machine learning approaches. The implementation section will describe the hardware architecture and key software libraries used to build and test our models. In experimental results we will apply our methods to forecast the average total load on the Elia electric grid and gauge the performance of the machine learning approaches against the benchmarks. Future directions will suggest extensions and new directions that can build upon our work, and finally the conclusion will concisely summarize the outcome of our research.

2. MOTIVATION

Load forecasting is a critically important task for utility companies since it is crucial in determining the amount of electricity that the company should provide. Failure to meet the demand can result in the overloading of network components and widespread grid outages, both of which are financially expensive and damaging to a utility company's reputation. Estimating future network flow accurately can allow providers to plan electricity generation, develop grid infrastructure and respond to varying frequencies so as to avoid these outcomes and improve overall network reliability.

An economic benefit of accurate load forecasts is the ability to price electricity in response to the aggregate demand. De-regularization and fierce competition in the energy market have made it vital for a transmission operator to be able to price electricity at a rate that is competitive with other energy providers. These prices fluctuate in response to the demand for electricity, with peak periods being characterized by high electricity prices and likewise off peak periods with lower prices. Dynamically determining when these periods occur and the demand for electricity during these durations can aid in the development of a demand based pricing scheme that is competitive and fair.

In the past various forecasting methods have been employed to predict future load, including econometric approaches, regression based models and statistical learning algorithms. These methods have had varying degrees of success, but there is still significant room for improvement. Given the rapid development of machine learning methods in recent years, it is interesting to gauge their predictive ability against simpler statistical methods and assess whether more complicated models can achieve greater accuracy for the load forecasting problem. We aim to apply machine learning algorithms to the load forecasting problem and determine whether they can outperform these traditional techniques. Moreover we will use well known time series analysis methods for scaling and de-trending the dataset to prepare it for the learning algorithms. This could potentially lead to more accurate load predictions and prove valuable to utilities for the aforementioned reasons.

3. RELATED WORK

As a baseline we can consider Taylor's research in ([12]), where an exponential smoothing method to forecast electricity demand a day ahead of time was constructed. The proposed method achieved fairly good results and was based on the assumption that the demand displayed seasonal patterns of differing frequencies, such as weekly and daily patterns. Our research will statistically test for seasonality patterns in the Elia dataset and use the information for one of the machine learning approaches, which this research has demonstrated can decrease prediction error.

The work of Taylor et al. in ([13]) also found that seasonality patterns exist in electric load time series, and exploited them in their statistical model that was again based on an exponential smoothing model. Perhaps more interestingly, they found that this simple statistical model outperformed neural networks, which performed quite poorly in predicting future load. However we suspect that this is not emblematic of the performance of discriminative learning based classifiers and will empirically demonstrate that this is not the case.

A machine learning approach to load forecasting was also taken in ([3]), which uses support vector machines (SVM) to forecast future electricity load. Their highest performing SVM incorporates time series information by forming feature vectors from the past several days of demand and using them to predict the load on future days. In our research we will use SVM's for prediction like this research, but instead will construct feature vectors from intra-day electricity load rather than from data over multiple days.

Finally we have the work of Ahmed in ([2]) which empirically demonstrates the benefit of statistically preprocessing time series data before applying machine learning algorithms for forecasting. They demonstrated that data normalization, logarithmic scaling, trend removal and deseasonalization can improve the performance of discriminative learning algorithms such as K-Nearest Neighbours and Support Vector Machines. In our approach we will logarithmically scale the data and remove the trend component before applying machine learning methods, and use the seasonality information for a clustering based learning algorithm.

4. Statistical Preprocessing

We begin by introducing some notation that will be used throughout the paper:
X_{train} - The sequence of vectors that compose the load time series from 2009-2013, in chronological order. Each vector is 96-dimensional since the total load on the electric grid is measured every fifteen minutes.
Y_{train} - The corresponding next day average total load for the examples in X_{train}. By average total load we mean the average over all 96 load values for the next day.
X_{test}, Y_{test} - The same as the training dataset except for the year of 2014.

The relationship between the examples and their target values can be mathematically expressed as follows:

$$Y^i_{train} = \frac{1}{96}\sum_{j=1}^{96} X^{i+1}_{train}[j] \forall i, 1 \leqslant i \leqslant |Y_{train}| \text{ and } Y^i_{test} = \frac{1}{96}\sum_{j=1}^{96} X^{i+1}_{test}[j] \forall i, 1 \leqslant i \leqslant |Y_{test}|$$

The last example in the training and test sets required extracting the load values from the next year which was done as part of the preprocessing. We have plotted the values of Y_{train} over the corresponding time period in Figure 1. From the plot there appears to be significant trend and seasonal components in addition to large numerical values for the load. The following preprocessing steps will address these issues to make the data more tractable for learning.

4.1. **Preprocessing.**

There are 2 steps that compose the preprocessing stage:

• Logarithmic Scaling : The time series contains very large numerical values, which can hinder the ability of classifiers to learn effectively and make the problem computationally intensive. To combat these two problems, the values in X_{train}, Y_{train} and X_{test} will be scaled by the natural logarithm before the learning stage.

• Trend Removal : Eliminating the trend in the target values can improve forecaster accuracy by removing the notion of time. To accomplish this, we first estimate the trend by performing least squares linear regression on the Y_{train} dataset. Each of the values correspond to a particular day, thus

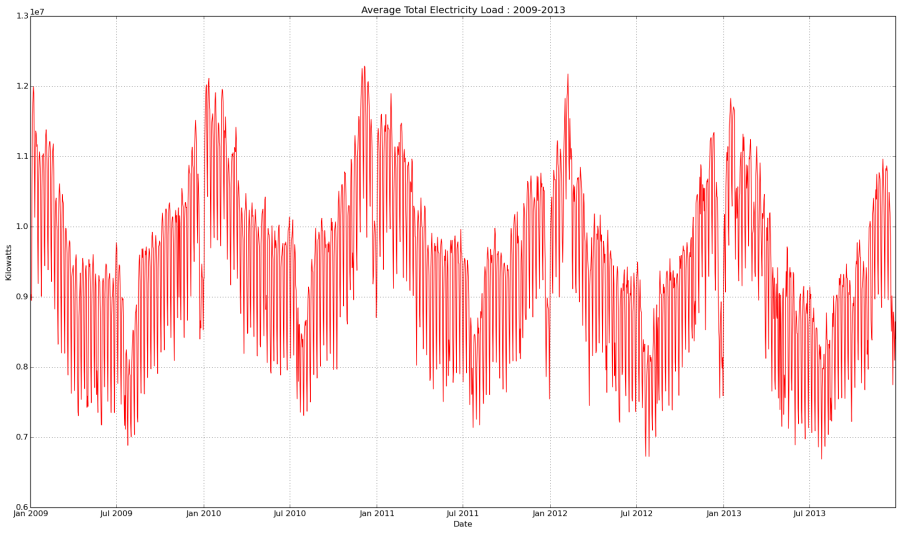


FIGURE 1. Average total load on the Belgian electric grid for 2009-2013

we can think of the values in Y_{train} as a sequence of values in chronological order. The independent variable for the regression is the number of days from the start of the series and the dependent variable is the the value of Y_{train} for that particular day. Once the regression line is computed, we subtract the regression predicted value from each value in Y_{train} to get the residuals which form the de-trended time series. The X_{train} values are left unaffected.

4.2. **Seasonality Analysis.** From previous research and observing the plot of Y_{train} in Figure 1, we suspect that the time series contains a significant seasonal component. To verify our intuition we proceed to statistically test for seasonality behaviour.

To begin we display the lag correlation plot of Y_{train} in Figure 2 which plots the values of Y_{train} against their previous day values. This plot at a high level allows us to determine whether the underlying time series is random or not. If it is random then the lag plot should not display any pattern or meaningful sub-structure. However based on the plot we can observe a significant linear trend, which is indicative of some underlying pattern that can potentially be due to seasonality.

Now that we have evidence for a seasonal component we will use the correlogram to determine the period of the seasons. The correlogram plots autocorrelation factors against their time lags, where the autocorrelation factor is the covariance of Y_{train} with itself under some given time lag divided by the variance of Y_{train}. Mathematically the estimation of the autocorrelation factor r_k with lesser bias (see Box-Jenkins [5]) for Y_{train} with time lag k can be expressed as:

$$r_k = \frac{1}{n-k}\left(\frac{\sum_{j=1}^{n-k}(Y^j_{train}-\mu_Y)(Y^{j+k}_{train}-\mu_Y)}{\sum_{j=1}^{n}(Y^j_{train}-\mu_Y)(Y^j_{train}-\mu_Y)}\right) = \frac{1}{\sigma^2(n-k)}\left(\sum_{j=1}^{n-k}(Y^j_{train}-\mu_Y)(Y^{j+k}_{train}-\mu_Y)\right)$$

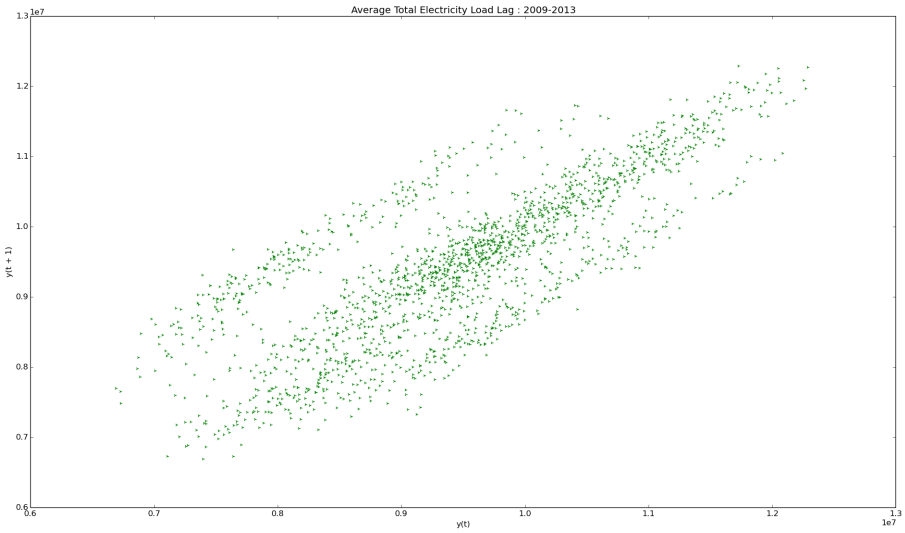
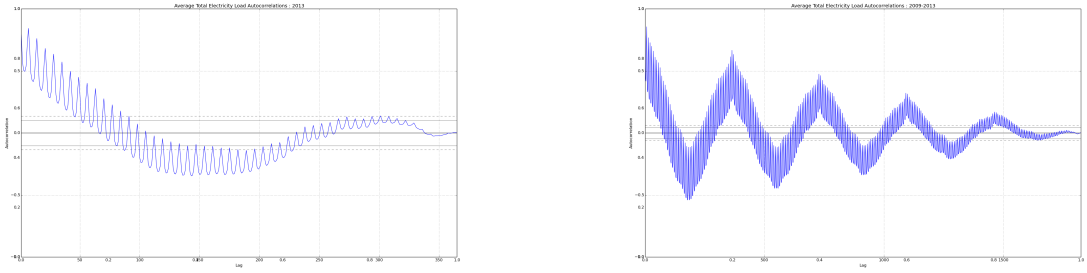


FIGURE 2. Lag plot for average total load during 2009-2013



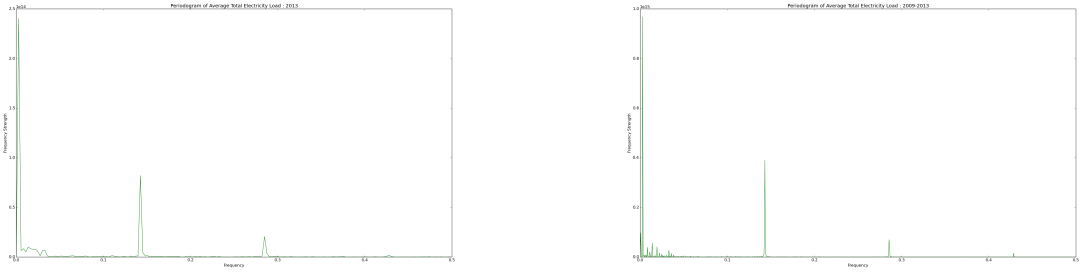
(A) Correlogram for 2013

(B) Correlogram for 2009-2013

FIGURE 3. Correlograms displaying autocorrelation factors at various time lags for the average total load

where we let $n = |Y_{train}|$ and define the mean of Y_{train} as $u_Y = \frac{1}{n}\sum_{i=1}^{n} Y^i_{train}$. Assuming that the values Y_{train} are i.i.d. the mean and variance of the autocorrelation factors is $\frac{-1}{n}$ and $\frac{1}{n}$, respectively. These can be used to find the 95% and 99% confidence bands, which are displayed as horizontal lines on the correlograms.

From the correlograms we can observe that a lag of one year on the $2009-2013$ plot and a lag of one week on the 2013 plot both yield autocorrelation factors exceeding the 99% confidence band. In addition we construct periodogram plots which can be used to identify significant periods of a time series in the frequency domain. From inspecting the periodograms we observe spikes at weekly and yearly frequencies, which is indicative of a strong correlation of the time series at these time lags. From the results of these statistical tests we can conclude that there is a high chance that the time series contains a significant seasonality component. We will use the weekly and yearly



(A) Periodogram for 2013 (B) Periodogram for 2009-2013

FIGURE 4. Periodograms of average total load for analysis in the frequency domain

seasonality periods discovered for the clustering based machine learning algorithm that will be described in the next section.

5. ALGORITHMS

5.1. **Support Vector Regression.** Our first machine learning based forecasting approach will use support vector machine regression (SVR). Traditionally support vector machines are used for binary classification tasks and at a high level find an optimal hyperplane in euclidean space that separates the data into two classes. The algorithm has since been modified for regression tasks and we can therefore apply it to the problem at hand. Specifically, SVR will solve the following optimization problem, presented below in primal form:

$$Minimize \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i}^{n}(\xi_i + \xi_i^*) s.t. \forall i, 1 \leqslant i \leqslant n,$$
$$w^T x_i + b - y_i \leqslant \varepsilon + \xi_i$$
$$-w^T x_i - b + y_i \leqslant \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \geqslant 0$$

Since SVR is known to excel at discriminative learning tasks and work well in high dimensions, it makes for a worthwhile approach towards our prediction task. However given the above formulation we are restricted to linear classification in the original vector space which could be suboptimal for learning. To overcome this restriction we take the dual of the primal formulation to obtain:

$$Maximize -\frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)x_i^T x_j \ \ and$$
$$\sum_{i=1}^{n} -\varepsilon(\alpha_i + \alpha_i^*) + y_i(\alpha_i - \alpha_i^*) \ \ s.t.$$
$$\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$$
$$\alpha_i, \alpha_i^* \in [0, C) \forall i, 1 \leqslant i \leqslant n$$

Both of these formulations are described by Smola in [11]. We can replace the inner product term in the dual formulation with a kernel k, which measures similarity as an inner product in a high dimensional vector space and allows SVR to learn non-linear patterns with respect to the original vector space. This could potentially lead to a more accurate predictor which will be investigated further in the experiments. The regression function f is then:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*) * k(x, x_i) + b$$

We now define the SVR based forecasting algorithm. The input to the algorithm consists of a kernel function k and misclassification penalty C along with the training and test datasets. The algorithm applies the scaling and de-trending steps to X_{train} and Y_{train} as detailed in the preprocessing section to obtain X'_{train} and Y'_{train}. Then SVR is used with kernel k and penalty C with X'_{train} and Y'_{train} to obtain a predictor f. Then for each example $x \in X_{test}$, the algorithm scales x to obtain x' and applies f to obtain a prediction $y' = f(x')$. The trend component is then added back to y' and then rescaled by the inverse natural logarithm to obtain the final value y.

5.2. **Clustering.** The next machine learning approach is based on clustering and uses the seasonality information examined in the preprocessing section. We can think of the season length being the number of unique classes of examples, where each class corresponds to an offset into any given season. In terms of clustering, we would like to construct a cluster for each of these classes such that examples within a cluster exhibit similarities while being dissimilar to examples in other clusters. To accomplish this we can cluster the examples and then given a test example, predict its value based on those it is most similar to, which would be the examples corresponding to the same seasonal offset as the test example (e.g. the examples in the same cluster as the test example).

We will try two different clustering algorithms. The first is a bottom-up type of hierarchical clustering called agglomerative clustering, which is an unsupervised learning algorithm that given a set of vectors and a cluster count k, finds a set of k clusters $S = \phi_1, ..., \phi_k$ such that every vector belongs to exactly one of the clusters and the inter-cluster distances are maximized (see Johnson - [6]). The distances between clusters will be computed using euclidean distances and Ward's linkage, whose objective is to merge clusters so as to minimize the total within cluster variance.

The second clustering approach is the same as the first except the clustering algorithm is K-Means. The K-Means algorithm will cluster a set of vectors into k clusters using the euclidean distance metric so as to minimize the within cluster sum of squares. Details of the K-Means algorithm can be found in Hartigan's paper ([4]).

The clustering based algorithm works as follows. The input to the algorithm consists of a season length s, a choice of clustering algorithm (either K-Means or Agglomerative), as well as the training and test datasets. Begin by logarithmically scaling and de-trending X_{train} and Y_{train} to obtain X'_{train} and Y'_{train}. Then run either the clustering algorithm with cluster count s on X'_{train} to obtain a set of clusters $S = \phi_1, ..., \phi_s$ with corresponding centroids $\mu_1, ..., \mu_s$. Then for each example $x \in X_{test}$, apply the logarithmic scaling step to obtain x'. Find the cluster $\phi^* \in S$ with centroid μ^* such that the euclidean distance between μ^* and x' is minimal, which can be expressed as follows:

$$\phi^* = argmin_{\phi_i \in S} d(\mu_i, x')$$

The target value y' of x' is then a weighted sum of each $y_t \in Y'_{train}$ for the $x_t \in X'_{train}$ that are members of the cluster ϕ^* multiplied by the euclidean distance between x_t and x', which is then

normalized by the sum of the euclidean distances between each of the x_t and x'. The prediction can then be expressed compactly as:

$$y' = \frac{\sum_{\{x_t \in \phi^*\}} d(x_t, x') * y_t}{\sum_{\{x_t \in \phi^*\}} d(x_t, x')}$$

Then add the trend to y' and rescale with the inverse natural logarithm to obtain the prediction y.

5.3. **Neural Networks.** The third learning approach we propose is based upon neural networks, which have been used for discriminative learning tasks for several decades and seen a large amount of success. A feed-forward neural network is an acyclic directed graph structure $G = (V, E)$ where the nodes V are neurons and are connected to each other by the set of directed edges E. There is a set of input neurons which take in an example and output the values of the example vector to subsequent hidden neurons. These hidden neurons take linear combinations of their inputs, potentially adding a bias value, and then apply an activation function to the computed sum and output the obtained value along the outgoing edges to other neurons. In this way values propagate through the network from the input layer until they reach an output layer, where an error function is applied to the value output from the output layer neurons. The network is then updated via the famous Back-propagation algorithm which attempts to minimize the obtained error via Gradient Descent by adjusting the network weights. The mechanics of the neural network we use is as follows:

The input z_j and output y_j of the neurons can be expressed as:

$$z_j = \sum_{i|e=(i,j)\in E} y_i w_{ij}$$

$$y_j = \frac{1}{1 + exp(-z_j)}$$

The Squared Loss for a predicted value y_j against a true value $\hat{y}$ is:

$$\zeta(y_j, \hat{y}) = \frac{1}{2}(y_j - \hat{y})^2$$

Define an error term ε_j which will make the subsequent gradient expression simpler.

The term ε_j for an output neuron j is:

$$\varepsilon_j = (y_j - \hat{y})(\frac{exp(-z_j) - 1}{(1 + exp(-z_j))^2})$$

The term ε_j for an input neuron j is:

$$\varepsilon_j = (\sum_{k|(j,k)\in E} \varepsilon_k)(\frac{exp(-z_j) - 1}{(1 + exp(-z_j))^2})$$

The gradient for the weight of an edge to an output neuron is:

$$\begin{aligned}\frac{\partial \zeta}{\partial w_{ij}} &= \frac{\partial \zeta}{\partial y_j}\frac{\partial y_j}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}} \\ &= (y_j - \hat{y})(\frac{exp(-z_j) - 1}{(1 + exp(-z_j))^2})(y_i) = \varepsilon_j y_i\end{aligned}$$

Likewise the gradient for the weight of an edges to an input neuron is:

$$\frac{\partial \zeta}{\partial w_{ij}} = \frac{\partial \zeta}{\partial y_j}\frac{\partial y_j}{\partial z_j}\frac{\partial z_j}{\partial w_{ij}}$$
$$= (\sum_{k|(j,k)\in E} \varepsilon_k)(\frac{exp(-z_j)-1}{(1+exp(-z_j))^2})(y_i) = \varepsilon_j y_i$$

The neural network will be trained using stochastic gradient descent (SGD), which estimates the gradient from a subset of the examples rather than the entire dataset. Updating the weights in this manner can lead to faster convergence. The weight update rule can then be written succinctly as:

$$w_{ij}^{t+1} = w_{ij}^{t} - \lambda(\sum_{\alpha=1}^{N} \varepsilon_j^{\alpha} y_i^{\alpha})$$

where SGD observes N examples before updating, and ε_j^{α} and y_i^{α} are the errors and neuron outputs condition on the example x_α. The value t stands for the current iteration or epoch, and the value λ is the learning rate which controls the speed at which the weights are updated.

Finally, unlike the other machine learning algorithms, we will reduce the dimensionality of the feature vectors with Principal Component Analysis (PCA). The reason for this is that the training set contains about 2000 examples which is far too few a number to train a network with 96 input neurons (consequently the hidden layers would also have to be larger) due to the number of parameters. Hence we use PCA to project the examples into a lower dimensional subspace while maintaining the maximum amount of variance, which should retain the information needed to distinguish between examples. A reference to PCA and how it works can be found in ([1]).

The forecasting algorithm based on neural networks works as follows. The input to the algorithm consists of the training and test datasets, a positive integer h for the number of hidden neurons and the dimension d for PCA (the activation and error functions are fixed and there is a single hidden layer, as we observed alternates perform much worse). Begin by logarithmically scaling and de-trending Y_{train} to obtain Y'_{train}. Logarithmically scale and apply PCA with target dimension d to X_{train} to obtain X'_{train}, retaining the projection matrix W used for PCA. Then construct a neural network with input layer of d neurons, a fully connected hidden layer of h neurons and a single output neuron with incoming edges from all the hidden neurons. The neuron activation functions are sigmoid and the loss is the squared loss, as detailed previously. For each example $x \in X_{test}$, logarithmically scale x to get $\hat{x}$ then obtain x' by projecting using the formula $x' = W\hat{x}$. Apply the neural network to x' to get an output y', then add the trend component and rescale with the inverse natural logarithm to obtain the predicted value y.

5.4. **Gaussian Process Regression.** Our final machine learning approach will use Gaussian Process Regression (GPR), which is a discriminative learning algorithm used primarily for regression tasks. Let $X = x_1, ..., x_n$ be training examples with corresponding target values $y = y_1, ..., y_n$. Then GPR will attempt to learn a model of the form:

$$y(x) = f(x) + \varepsilon$$

where $f(x)$ is a regression component and ε is a gaussian noise term with zero mean and variance σ_n^2, i.e. $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$ (see Radmussen [9]). Additionally there is a fully stationary covariance matrix Σ for the examples X under some covariance function k such that $\Sigma_{ij} = k(x_i, x_j)$. This implies that y follows the multivariate gaussian distribution $y \sim \mathcal{N}(0, \Sigma)$. Inference in GPR can now be

derived by the maximum likelihood principle using Bayes theorem for some unknown example x^*. Let $\Sigma_{x^*} = [k(x_1, x^*), k(x_2, x^*), ..., k(x_n, x^*)]$ be a covariance vector between the unknown and given examples and $N \in R^{nXn}$ be a diagonal matrix where the i^{th} diagonal value is the ε noise for the i^{th} example, and all other entries zero. Then the predicted value y^* can be expressed as:

$$y(x^*) = \Sigma_{x^*}[\Sigma^{-1} + N]y$$

The GPR forecasting algorithm is then as follows. The input to the algorithm consists of a covariance function k as well as the training and test datasets. The algorithm applies the logarithmic scaling and de-trending operations to X_{train} and Y_{train} to obtain X'_{train} and Y'_{train}. A predictor f is then computed using GPR on X'_{train} and Y'_{train} with covariance measure k. Then for each example $x \in X_{test}$ apply logarithmic normalization to obtain x' and compute $y' = f(x')$. The trend component is then added back to y' and the inverse natural logarithm applied to obtain the prediction y.

6. Econometric Benchmarks

The aforementioned machine learning approaches will be benchmarked against several econometric models that are commonly used in time series forecasting.

An Autoregressive (AR) model is a classical econometric model that expresses values in a time varying process as a linear combination of previously observed values. Specifically, an AR model of order p assumes that a value x_t at time t in the process can be written as a sum of a constant, noise term, and p previously observed values. Compactly, this is expressed as follows:

$$x_t = c + \varepsilon_t + \sum_{j=1}^{k} x_{t-j} w_j$$

where ε_t is a white-noise error term, e.g. $\varepsilon_t \sim N(0, \sigma^2)$, and c is a constant.

A Moving Average (MA) model is another econometric model that assumes values in a time varying process can be expressed as linear combinations of error terms and a constant mean. In particular, a MA model of order q assumes that a value x_t at time t can be expressed as the sum of a constant mean and a weighted sum of previous q errors.

$$x_t = \mu + \sum_{j=1}^{q} \varepsilon_{t-j} w_j$$

The ε_t error terms are assumed to be mutually independent white noise error terms. An MA model assumes that the process is generated by a MA process, and since a MA process is always strongly stationary the process is assumed to have this characteristic as well. This implies that the MA model works well on processes with the properties of constant mean, finite variance and that the cumulative distribution function of the joint probability distribution for a sequential set of values in the process is invariant to time shifting.

$$\frac{1}{|X|} \sum_{x \in X} x = c$$

$$\sum_{x \in X} (x - \frac{1}{|X|} \sum_{x \in X} x) < \infty$$

$$F_X(x_i, ..., x_{i+k}) = F_X(x_{i+j}, ..., x_{i+j+k}) \forall j \in \mathbb{N}$$

The final benchmarking model is Double Exponential Smoothing (DES), otherwise known as Holt-Smoothing after its founder. A DES model makes the assumption that values in a time varying process can be expressed as a sum of a trend and level component. A DES model is characterized by the following two equations:

$$s_t = \alpha y_t + (1 - \alpha)(s_{t-1} + b_{t-1}), 0 < \alpha < 1$$
$$b_t = \gamma(s_t - s_{t-1}) + (1 - \gamma)b_{t-1}, 0 < \gamma < 1$$

The S_t term is the smoothed prediction for the process at time t, and b_t is the trend component at time t. The first equation is used to continually update the smoothed predictions, and the second equation is to update the trend throughout the process. The parameters α is a tradeoff factor between the current value in the process and the previous smooth and trend values, and the γ factor is a tradeoff between the current trend and previous trend components. Forecasting m time-steps ahead in the DES model is then done with the following equation:

$$x_{t+k} = s_t + k * b_t$$

Careful observation will indicate that the DES model does not account for a seasonal component, which is considered in the Triple Exponential Smoothing (TES) model. However in our tests we found that TES performs worse than DES, and so choose to omit a description of TES.

The parameters of the aforementioned econometric models can be derived in a myriad of different ways. For simplicity we chose to use the default *R* implementations of these methods which can derive the parameters automatically.

In the context of the load forecasting problem, we wish to predict each value Y^i_{test} for $1 \leqslant i \leqslant |Y_{test}|$. For each econometric model, we predict Y^i_{test} by fitting the model to the entire Y_{train} time series and $Y^1_{test}, ..., Y^{i-1}_{test}$, then predict one day ahead using the model. This necessarily requires reconstructing the model for every day we wish to predict, however this is computationally feasible as the dataset is relatively small. Also observe that while we do use part of Y_{test} for model construction, we only use the values that have been observed so far in time and therefore have already been revealed.

7. Implementation

. The project was implemented in Python, with heavy use of the scikit-learn ([8]) machine learning library, Numpy numerical computing library, Pandas ([7]) data analysis library and Pybrain ([10]) machine learning library. The system architecture is as follows:

- Operating System - Ubuntu 13.10
- Processor - Intel Core i7-3610QM Processor (6M Cache, up to 3.30 GHz)
- RAM - 12GB DDR3 @ 1600 MHZ

8. Experimental Results

We now apply the four machine learning based algorithms along with the baseline econometric models to the task of forecasting average total load for 2014. For each algorithm we tried several different parameters and displayed the results for each of the configurations on a single plot. Figures 5,6,7,8 and 9 display the econometric benchmark, SVR, clustering, GPR and neural network results on the original scale, respectively.

We chose to measure performance according to the normalized root-mean square error (NRMSE) since it is scaled and allows for a fair comparison between the algorithms. If we let Y_{pred} and Y_{test} be

n prediction and corresponding test values, respectively, the function for NRMSE can be expressed as:

$$NRMSE(Y_{pred}, Y_{test}) = \frac{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y^i_{pred} - Y^i_{test})^2}}{Y^{MAX}_{pred} - Y^{MIN}_{pred}}$$

We measured the NRMSE for all of the algorithms along with their various configurations and displayed the results and parameter choices in Table 1.

We immediately notice that our results show strong evidence that machine learning algorithms can outperform the econometric approaches for load forecasting. All four of the machine learning approaches had at least one parameter configuration where the NRMSE was lower than all of the benchmark NRMSE's. The lowest NRMSE rate of 0.096 was achieved by GPR with linear, cubic and absolute exponential kernels, with square exponential GPR and clustering methods achieving low NRMSE rates as well. This is is significantly better than the baseline econometric methods where the lowest NRMSE was 0.128 using an autoregressive model of order seven. The fact that GPR surpassed the clustering, SVR and neural network algorithms demonstrates the well known ability of GPR to achieve high accuracy rates on discriminative regression learning tasks when the dataset is not too large (e.g. at most a few thousand examples and several dozen dimensions). However our results are somewhat surprising in that aside of the preprocessing steps, no information regarding seasonality was used in the GPR model, even though previous research and our initial analysis showed that the load time series contained a significant seasonality component. This is suggestive that seasonality information might not be necessary to achieve high accuracy rates for forecasting the average total load. Nevertheless we found that the normalization and de-trending preprocessing steps were crucial for the machine learning methods to be successful, thus showing that the ideas from time series analysis can be used in combination with machine learning to develop strong prediction models.

9. Future Directions

Based on our results we have strong evidence that machine learning based algorithms have the ability to achieve high accuracy rates for the load forecasting problem, and consequently believe it would be rewarding to explore this area further. One immediate direction would be to try other discriminative machine learning algorithms such as Bayesian neural networks, k-nearest neighbours, regression methods (Ridge, Logistic, Bayesian) and decision trees and measure their performance against the results found in this paper. Also since our algorithms commonly had numerous parameters (SVR had the kernel function and penalty for error, GPR had the covariance function), another direction might be to try varying the parameters to see if another choice yields lower error rates. Moreover multiple algorithms could be used together with ensemble methods such as boosting or bootstrap aggregating, which historically have been demonstrated in certain cases to outperform the constituent algorithms for the ensemble.

Furthermore it seems worthwhile to investigate methods from time series analysis and how they can be used in conjunction with machine learning. At this point there is a wide variety of time series methods that can be used to alter and transform datasets which allows for a deeper understanding and analysis of the underlying process responsible for generating the data. Some examples include noise detection and removal, de-seasonalization (although we found additive decomposition to

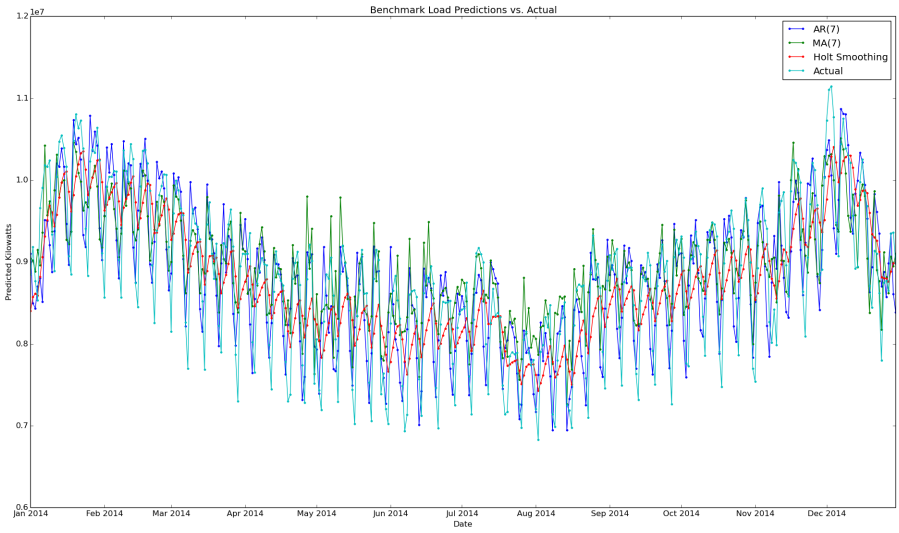


FIGURE 5. Benchmark predictions for average total load in 2014

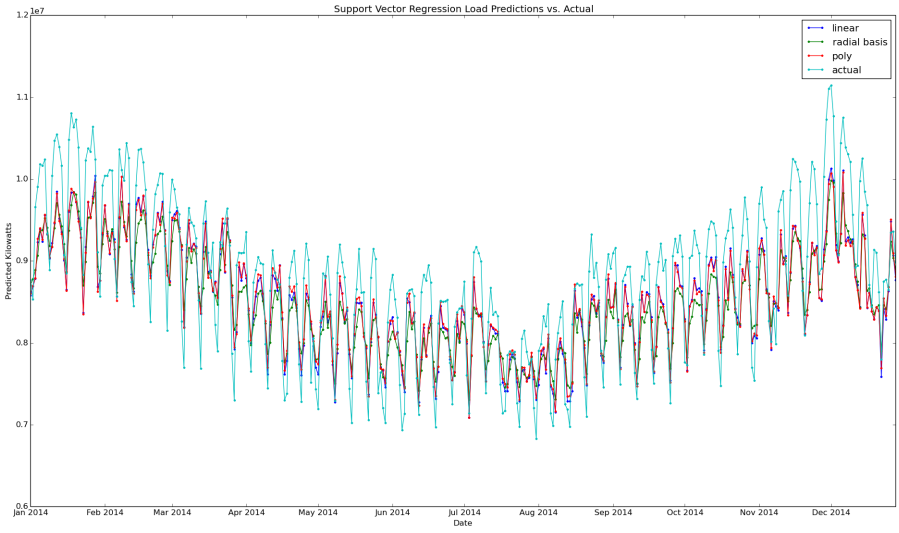


FIGURE 6. SVR predictions for average total load in 2014

be ineffective) and smoothing (Kalman filtering, Laplacian). Using these techniques along with learning algorithms might lead to robust and accurate predictors that could potentially outperform the use of pure econometric or machine learning approaches.

Finally it would be interesting to gather time series data for other electric load time series from other transmission operators and apply the techniques used in this paper. We believe that electric load time series share certain commonalities such as seasonality effects and trends, and that the forecasting methods presented in this paper should achieve at least a moderate amount of success on

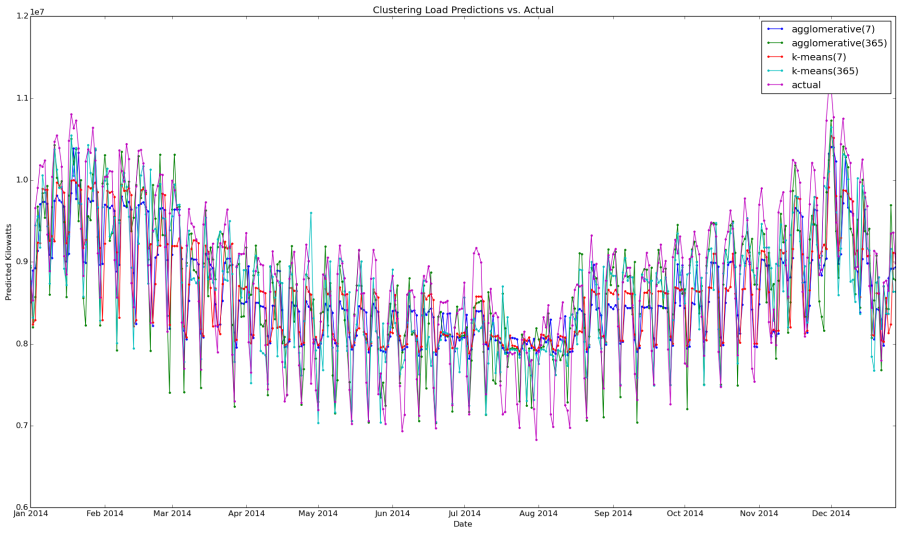


FIGURE 7. Clustering predictions for average total load in 2014

FIGURE 8. GPR predictions for average total load in 2014

such datasets. Weather data could also be collected and used for forecasting, which other researchers have demonstrated is strongly correlated with electricity usage. Unfortunately we did not have access to Belgium weather data and could not pursue this direction in our research. In any case, the results obtained could then be compared to those that are used by the utilities to see if they are more accurate than what is being used in industry, which could have a profound impact if the predictions are consistently better. As this is still an active area of research, it is worthwhile to

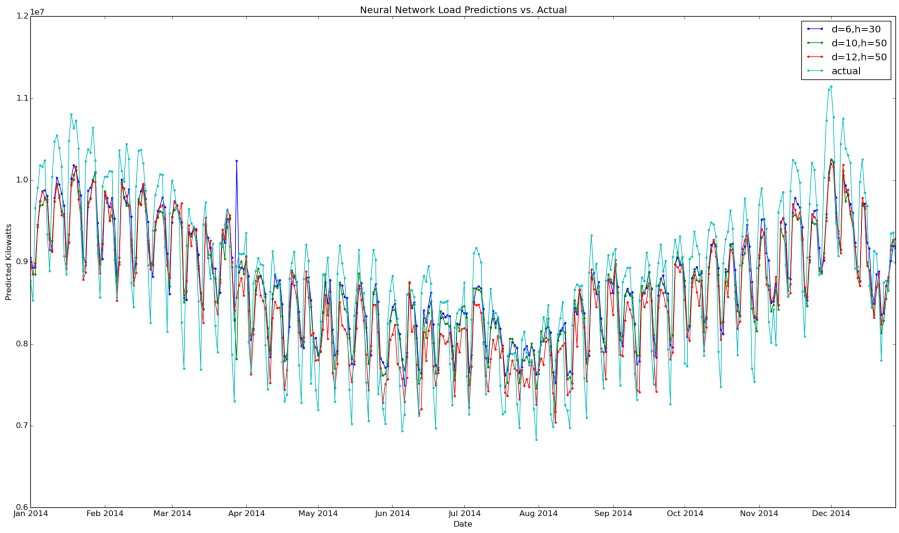


FIGURE 9. Neural Network predictions for average total load in 2014

Algorithm	Parameters	NRMSE
Autoregressive	Order $p = 7$	0.128
Moving Average	Order $q = 7$	0.132
Double Exponential Smoothing	$\alpha = 0.5$, $\gamma = 0.5$	0.131
Support Vector Regression	Linear Kernel : $k(x, x') = x^T x'$	0.123
Support Vector Regression	Radial Basis Kernel : $k(x, x') = exp(\|\|x - x'\|\|^2)$	0.139
Support Vector Regression	Polynomial Kernel : $k(x, x') = (x^T x' + 1)^{0.95}$	0.123
Gaussian Process Regression	Square Kernel : $k(x, x') = \sigma^2 exp(\frac{\|\|x-x'\|\|}{2l^2})$	0.097
Gaussian Process Regression	Absolute Kernel : $k(x, x') = \sigma^2 exp(\frac{\|x-x'\|}{l})$	0.096
Gaussian Process Regression	Cubic Kernel $k(x, x') = Max(0, 1 - 3(\theta\|\|x - x'\|\|^2) + 2(\theta\|\|x - x'\|\|)^3$:	0.096
Gaussian Process Regression	Linear Kernel : $k(x, x') = \sigma^2 x^T x'$, $\sigma = 0.1$	0.096
Clustering	Agglomerative, Ward Linkage, $s = 7$	0.150
Clustering	Agglomerative, Ward Linkage, $s = 365$	0.123
Clustering	K-Means, $s = 7$, 10 runs	0.149
Clustering	K-Means, $s = 365$, 10 runs	0.129
Neural Network	$h = 30, d = 6$ (40 epochs, $\lambda = 0.015$)	0.127
Neural Network	$h = 50, d = 10$ (40 epochs, $\lambda = 0.015$)	0.122
Neural Network	$h = 50, d = 12$ (40 epochs, $\lambda = 0.015$)	0.128

TABLE 1. Comparison of Algorithms for 2014 Load Forecasting

compare approaches to real world standards to determine whether the forecasting methodology currently being used by utility companies could be improved.

10. Conclusion

Based upon our empirical results we have successfully demonstrated that machine learning techniques can result in accurate predictors for forecasting the average total load on the Elia electric grid one day ahead of time. Moreover we compared these approaches against orthodox econometric models such as auto-regression and moving average and discovered that the learning based algorithms can outperform these traditional approaches. Unfortunately seasonality information did not help improve the performance of the most accurate classifier which used Gaussian Process Regression, although it was integral to a clustering based approach that also achieved fairly low error rates. However we discovered that logarithmic scaling and de-trending the load time series significantly increased the accuracy of the learning based models. Overall we believe that our approaches clearly demonstrate the value in utilizing both time series and learning methods which we conjecture will be integral in future work on load forecasting. The problem remains an active area of research and is far from being a solved case, and we hope that transmission operators continue to pursue improved forecasting methods for their significant practical value.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, 2010. 9

[2] Nesreen K Ahmed, Amir F Atiya, Neamat El Gayar, and Hisham El-Shishiny. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews*, 29(5-6):594–621, 2010. 3

[3] Bo-Juen Chen, Ming-Wei Chang, and Chih-Jen Lin. Load forecasting using support vector machines: A study on eunite competition 2001. *Power Systems, IEEE Transactions on*, 19(4):1821–1830, 2004. 3

[4] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, pages 100–108, 1979. 7

[5] Gwilym M Jenkins and Donald G Watts. Spectral analysis. 1968. 4

[6] Stephen C Johnson. Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254, 1967. 7

[7] Wes McKinney. pandas: a foundational python library for data analysis and statistics. 11

[8] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011. 11

[9] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006. 9

[10] Tom Schaul, Justin Bayer, Daan Wierstra, Yi Sun, Martin Felder, Frank Sehnke, Thomas Rückstieß, and Jürgen Schmidhuber. PyBrain. *Journal of Machine Learning Research*, 11:743–746, 2010. 11

[11] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004. 7

[12] James W Taylor. Short-term electricity demand forecasting using double seasonal exponential smoothing. *Journal of the Operational Research Society*, 54(8):799–805, 2003. 2

[13] James W Taylor, Lilian M De Menezes, and Patrick E McSharry. A comparison of univariate methods for forecasting electricity demand up to a day ahead. *International Journal of Forecasting*, 22(1):1–16, 2006. 3