EEVC
Brussels, Belgium, November 19-22, 2012

Energy Demand Prediction in a Charge Station: A Comparison of Statistical Learning Approaches

Hélène Le Cadre[1], Cédric Auliac[2]
[1]*Mines ParisTech, Centre for Applied Mathematics, France, Email: helene.le_cadre@mines-paristech.fr*
[2]*CEA, LIST,* 91191 *Gif-sur-Yvette CEDEX, France, Email: cedric.auliac@cea.fr*

Abstract

In this article, we compare the performances of 5 learning techniques: artificial neural networks, support vector machines, ARIMA processes, regret based methods and a naive approach based on the periodic repeatitions of the predictions. They have been tested over a real database which can be associated with the energy demand generated by electric vehicles wishing to reload, in a specific charge station. Using this generic database, our simulations highlight the fact that regret based methods clearly outperform the other learning approaches. This class of methods is all the more interesting as it enables the introduction of game theory to model the interdependencies between the agents composing the ecosystem and provides economic guidelines.

Keywords: Supervised Learning, Regret algorithms, Electric Vehicle Demand

1 Introduction

The management of Electric Vehicle (EV) charging infrastructure requires both to optimize the location of the various charge stations composing the underlying network and to provide relevant predictions to the EV drivers and to the utilities managing the charge stations. We proposed and tested an algorithm based on the cross-entropy method in [4], to optimize the charging infrastructure, under competition between the utilities. The solving of the non-cooperative geographic location game between the utilities required to make simplifying assumptions about the EV drivers' energy demand. To improve the accuracy of the modelling assumptions regarding the EV drivers' energy demand, it is necessary to develop efficient learning approaches. One major problem is that it is impossible to measure explicitly the real energy demand of the drivers. Indeed, it is the consumption that the utilities like Better Place, monitor in their charge stations. Of course, there exists a non-trivial relation between demand and consumption but it appears as impossible to infer analytically. In the context of the european FP7 project Elvire, we modeled the problem as a partial information game and provided predictions about the energy demand using regret based methods [1], [3]. Following this direction, the goal of this article is to compare the performances of 5 energy demand prediction methods on a real data set.

The prediction system is based on data measurements available online on the machine learning repository PEMS-SF [2]. More precisely, we have downloaded 15 months worth of daily data from the California Department of Transportation website. The data describes the occupancy rate, between 0 and 1, of different car lanes of San Francisco bay area freeways. The measurements cover the period from january 1, 2008 to march 30, 2009 and are sampled every 10 minutes. We consider each day in the database as a single time series of dimension 963 i.e., the number of sensors which functioned consistently throughout the studied period, and length 144. We remove public holidays from the dataset, as well as two days with anomalies (march 8, 2009 and march 9, 2009) where all the sensors were muted between 2 and 3 a.m. This results in a database of 440 time series. Associating a sensor with a charge station, we have kept only the first sensor historical data.

To be more generic, we will consider any data

measurements to improve our predictions. The data probes can measure for example:

- The time series of the number of EVs in the charge stations of the utility managing the charging infrastructure
- The time series of the road occupancy, under the assumption that the road occupancy is measured by a coefficient between 0 (no vehicle) and 1 (maximal congestion)
- The time series of the energy demand in a specific region
- The revenues of an energy producer, etc.

As mentioned above, the data measurements are stored in a database server. They are all synchronized and associated with a time instant. Then, there exists at least two approaches to forecast the energy refuelling demand:

> Approach A: Either, estimate the real demand from the aggregation of the different measures given by all the sensors at past time periods, and through a time series forecasting method, forecast the future values.
>
> Approach B: Or, use directly a forecasting method starting from the data measurements. It is also called a partial feedback forecasting method.

Under Approach A, we will use artificial neural networks, support vector machines and ARIMA models. Under Approach B, we will use the regret based monitoring technique.

In Section 2, we will describe artificial neural networks, support vector machines and ARIMA models, while in Section 3 we will detail the principles of the regret based methods. Finally, in Section 4, we will provide an interpretations of the results that we obtained using the PEMS-SF database.

2 Approach A: Description of the time series forecasting algorithms

There are three major learning processes, each corresponding to a particular abstract learning task. These are *supervised* learning, *unsupervised* learning and *reinforcement* learning.

In *supervised* learning, we are given a set of example pairs $(x, y) \in X \times Y \subset \mathbb{R}^2$ and the aim is to find a function $f : X \to Y$ in the allowed class of functions that matches the examples. In other words, we wish to infer the mapping implied by the data. The cost function is related to the mismatch between our mapping and the data and it implicitly contains prior knowledge about the problem domain. A commonly used cost is the mean-squared error. With this performance measure, the forecaster will try to minimize the average squared error between the network's output $f(x)$, and the target value y, over all the example pairs. Tasks that fall within the paradigm of supervised learning are pattern recognition, also known as classification, and regression, also known as function approximation.

In *unsupervised* learning, some data x is given and the cost function to be minimized can be any function of the data x and the network's output. Tasks that fall within the paradigm of unsupervised learning are in general, estimation problems. The applications include clustering, the estimation of statistical distributions, compression and filtering.

In *reinforcement* learning, data x are usually not given, but generated by an agent's interactions with the environment. The aim is to discover a policy for selecting actions that minimizes some measure of a long-term cost i.e., the expected cumulative cost. The environment's dynamics and the long-term cost for each policy are usually unknown, but can be estimated. Tasks that fall within the paradigm of reinforcement learning are control problems, games and other sequential decision making tasks.

In this section, we will concentrate on the supervised learning paradigm, assuming that we have some real historical data on the electric refuelling demand to predict the future, even if in practice, we will most probably not get data on the exact value of the demand but on some indirect measures of it, like cars stopping at the station or the quantity of battery exchanged in a day.

2.1 Artificial Neural Networks (ANNs)

Artifical Neural Networks (ANNs) are non linear statistical data modeling tools. They are usually used to model complex relationships between inputs and outputs, or, to find patterns in data. It is inspired by the structure and/or functional aspects of biological networks.

The network pictured in Figure 1, has three layers. The first layer has input neurons, which send data via synapses to the second layer of neurons, and then via more synapses to the third layer of the output neuron. More compex systems will have more layers of neurons with some having increased layers of input neurons and output neurons. The synapses store parameters called weights that manipulate the data in the computations. An ANN is typically defined by three types of parameters:

- The interconnection pattern between the different layers of neurons
- The learning process for updating the weights of the interconnections
- The activation function that converts a neuron's weighted input to its output activation

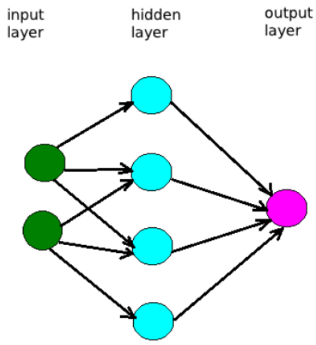


Figure 1: Example of a three layer neural network.

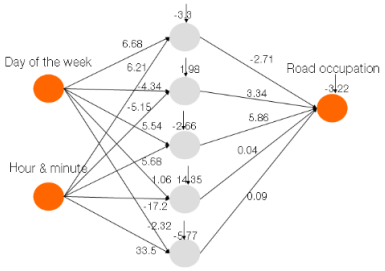
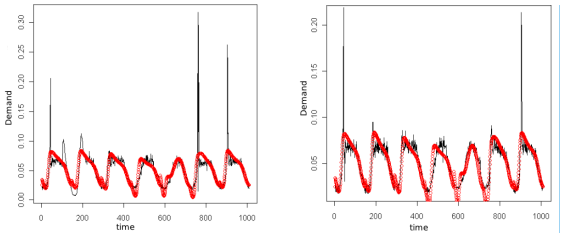


Figure 2: The experimentally optimized three layer ANN used in our study.



(a) (b)

Figure 3: Learning with ANN over a week.

In our application for the energy demand forecast, we have decided to stick to three layers of neurons. The input layer encodes the time of the measure. The output layer is a single neuron representing the demand. However, the number of hidden neurons is not fixed. We consider that all input neurons send their information to all hidden neurons and that all hidden neurons send their information to the output neuron. This structure is sometimes called a feed-forward neural network. We also add to each neuron a specific input which is a white noise. It enables some probabilistic behavior.

The activation function is linear. It equals the weighted sum sent by the input neurons of a given neuron. The output has values in $[0; 1]$. To make less computation errors, we will need to scale the inputs of the neural network to have values in $[0; 1]$. This scaling is needed in most neural network applications but may result in quite large absolute errors if the considered domain for the demand is large.

Referring to the supervised learning framework described at the beginning of the section, we will learn the following function: demand $= f(\text{month}, \text{day}, \text{hour}, \text{minute})$ represented as a graph of neurons from n historical data points and then predict the future demand by applying this function on some future dates.

We fit a neural network with 5 hidden neurons on the data of occupancy of a California road for 60 days with 144 measures per day. It gives us the neural network pictured in Figure 2 where each value on a synapse is the weight of the synapse and the activation function is linear.

We can check the good fit of the ANN by showing the output which was produced by the input data of the second week of the learning data in Figure 3 (a) and of the data points corresponding to the week between day 67 and 74 in Figure 3 (b), and by comparing it with the true values. The black line represents the true measurements while the red points are the estimated ones, generated by the ANN.

2.2 Support Vector Regression (SVR)

A support vector machine (SVM) is a supervised learning method that analyzes data and recognizes patterns, used for classification and regression analysis. In that sense, it is very close to a neural network and can even be interpreted as an extension of a specific neural network named the perceptron. However, there is no a priori on the dependency structure between inputs and output and on the activation functions forms in the case of SVM contrary to neural networks. In addition, during the learning phase, a SVM model tries to separate output categories by a clear gap that is as wide as possible and not just big enough to classify the learning data correctly. As a result, it is less sensitive to learning data isolated errors (outliers) than a neural network.

The standard SVM is a non-probabilistic binary linear classifier. It takes a set of input data and predicts, for each given input, which of two possible classes forms the input by defining two parallel hyperplanes as far as possible from each other, separating the learning data. In Figure 4, we separate green points on one side of a first hyperplane from white points on the other side of a second hyperplane. These two hyperplanes are the most distant possible hyperplanes of the considered space separating the two types of points. In 1992, Boser et al. suggested a way to create nonlinear classifiers by applying the kernel trick [5] to maximum-margin hyperplanes. This allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The transformation may be nonlinear and the transformed space high dimensional. Thus,

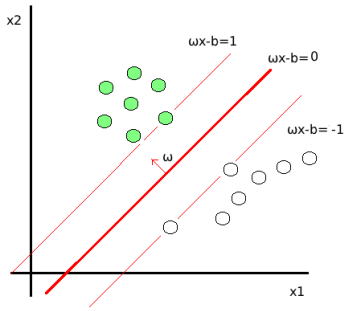


Figure 4: SVM as a binary classifier.

though the classifier is a hyperplane in the high-dimensional feature space, it may be nonlinear in the original input space. Some common kernels include: polynomial, hyperbolic tangent and Gaussian radial basis kernels. The separation line between the two classes which is a hyperplane in the feature space may be a very complex curve in the original space as depicted in Figure 5.

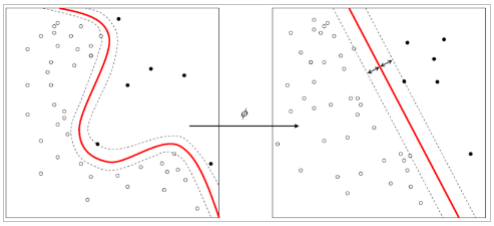


Figure 5: Illustration of the kernel trick (source: Wikipedia).

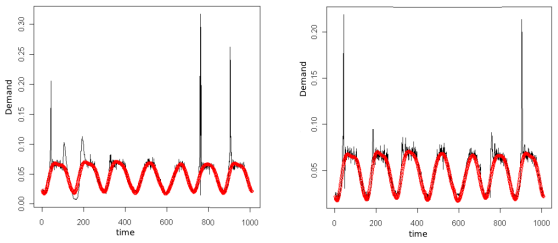
In our application, we will use the radial basis kernel which means that the formula $\exp(-\gamma|u-v|^2)$ is used instead of the scalar product of u and v. We will choose γ as the inverse of the number of observations.

A version of SVM for regression was proposed in 1996 by Vapnik et al. [6]. This method is called support vector regression (SVR). It performs linear regression in the high-dimension feature space using ε insensitive loss and, at the same time, tries to reduce model complexity by minimizing the linear regression factor ω. This can be described by introducing (non-negative) slack variables ξ_i, ξ_i^*, to measure the deviation of training samples outside ε insensitive zone. Thus, SVM regression is formulated as minimization of the following functional:

$$\min \frac{1}{2}\|\omega\|^2 + c\sum_{i=1}^{n}(\xi_i + \xi_i^*)$$
$$y_i - f(x_i, \omega) \leq \varepsilon + \xi_i^*$$
$$f(x_i, \omega) - y_i \leq \varepsilon + \xi_i$$
$$\xi_i, \xi_i^* \geq 0, \ \forall i = 1, ..., n$$

In our application, we will choose $\varepsilon = 0.1$ and $c = 1$ for our computations and $f(.)$ is the radial basis kernel.

We run the support vector machine algorithm on the PEMS-SF database. The algorithm gives us more than 4000 support vectors which means that the space of days of the week and hours of the day is very fragmented in terms of road occupancy.
We can check the good fit of the SVR model by showing the output produced by the input data of the second week of the learning data in Figure 6 (a) and of the data points corresponding to the week between day 67 and 74 in Figure 6 (b), and by comparing it with the true values. The black line represents the true measurements while the red points are the estimated ones, given by the SVR method.



(a) (b)

Figure 6: Learning based on SVR over a week.

2.3 ARIMA processes

An autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. They are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the *integrated* part of the model) can be applied to remove the non-stationarity.

The model is generally referred to as an ARIMA(p, d, q) model where p, d, and q are non-negative integers that refer to the order of the autoregressive, integrated, and moving average parts of the model respectively. ARIMA models are used for observable non-stationary processes x_t that have some clearly identifiable trends:

- A constant trend i.e., zero average, is modeled by $d = 0$
- A linear trand i.e., linear growth behavior, is modeled by $d = 1$
- A quadratic trend i.e., quadratic growth behavior, is modeled by $d = 2$

Sometimes a seasonal effect is suspected in the model. For example, if we consider a model of daily road traffic volumes. Weekends clearly exhibit different behavior from weekdays. In such cases, it is often considered better to use a

SARIMA (seasonal ARIMA) model than to increase the order of the AR or MA parts of the model.

The ARMA model consists of two parts, an autoregressive (AR) part and a moving average (MA) part. The autoregressive part of the model of order p is written:

$$x_t = \bar{c} + \sum_{i=1}^{p} \bar{\varphi}_i x_{t-i} + \varepsilon_t$$

where the $\bar{\varphi}_i, i = 1, ..., p$ are the parameters of the model, $\bar{c}$ is a constant and ε_t is a white noise. The moving average part of the model of order q is written:

$$x_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$$

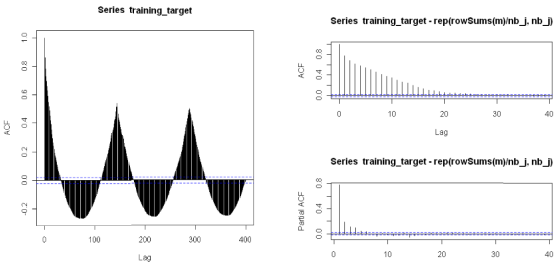
where the $\theta_i, i = 1, ..., q$ are the parameters of the model, μ is the expecation of x_t. It is often assumed to equal 0 and the $\varepsilon_t, \varepsilon_{t-1}, ..., \varepsilon_{t-q}$ are white noise error terms.

We can see from these definitions that a (S)ARIMA model assumes that the time series we are studying are mainly stationary or have a constant trend in their evolution. If there is a shift in car drivers behaviors, due for example to tax increase or promotion or other external events, it will not be taken into account. In addition, to catch all the trends in the evolution, we need quite a long period of observed data to detect the autocorrelation. For example, if there is a seasonal effect such as a big increase in transportation during summer holidays we should need more than one year of data to detect it. This is too long to ensure an optimal mining of the data.

The estimation of the ARIMA model corresponding to some learning data is done through the Box and Jenkins method [7]. The original model uses an iterative three-stage modeling approach:

- Model identification and model selection: making sure that the variables are stationary, identifying seasonality in the dependent series (seasonally differencing it if necessary), and using plots of the autocorrelation and partial autocorrelation functions of the dependent time series to decide which (if any) autoregressive or moving average component should be used in the model.
- Parameter estimation using computation algorithms to arrive at coefficients which best fit the selected ARIMA model. The most common methods use maximum likelihood estimation or non-linear least-squares estimation.
- Model checking by testing whether the estimated model conforms to the specifications of a stationary univariate process. In particular, the residuals should be independent of each other and constant in mean and variance over time. (Plotting the mean and variance of residuals over time and performing a Ljung-Box test or plotting autocorrelation and partial autocorrelation of the residuals are helpful to identify misspecification.) If the estimation is inadequate, we have to return to step one and attempt to build a better model.

We follow the Box and Jenkins method [7] to decompose the PEMS-SF database. First, we run an autocorrelation diagram. Its output is represented in Figure 7 (a). We see that there is a strong seasonality of 144 time periods (which corresponds to one day). We extract the seasonal part of the historical data and we draw again an autocorrelation diagram and a partial autocorrelation diagram which outputs are drawn in Figure 7 (b). We see that the remaining part of the

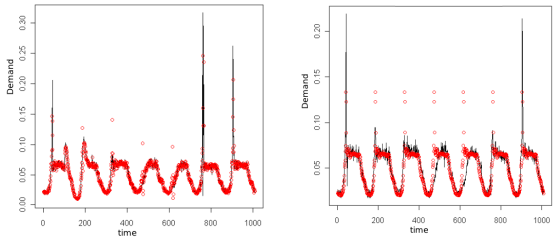


(a) (b)

Figure 7: Autocorrelation diagrams for the PEMS-SF data.

historical data out of the seasonal part is an auto regressive model of order 2. We estimate the corresponding coefficients and we obtain a full model of the road occupancy.

We can check the good fit of the ARIMA process by showing the output produced by the input data of the second week of the learning data in Figure 8 (a) and of the data points corresponding to the week between day 67 and 74 in Figure 8 (b), and by comparing it with the true values. The black line represents the true measurements while the red points are the estimated ones, given by the ARIMA process.



(a) (b)

Figure 8: Learning with an ARIMA process over a week.

3 Approach B: Description of the regret based forecasting algorithms

3.1 Time series forecasting

We can imagine from the previous section that there is a lot of possible techniques to model and predict the energy refuelling demand of electric car vehicles; for example, we can think about all the possible kernel functions when considering support vector machines. But how can we select the best candidate model among all the possible? In many cases, it is a long process of iterative selection which last a few years until the industry reaches a consensus on the best model. It is even more difficult in our case as the refuelling technologies are not stabilized, the available measures to make the prediction are not so clear, and the car drivers' behavior is in definition. One possible solution is to use the regret based techniques to select dynamically the best forecasting technique. Indeed, the starting idea of regret based techniques is to select the best forecasting expert among a finite set by estimating the regret of not having followed that expert at each time period.

The advantages of regret-based techniques are:

- It does not require any a priori on the data that we are forecasting
- It is an online learning algorithm in the sense that it can accept any fresh data measures at each time period to improve its accuracy without needing to rerun the algorithm on all the learning phase contrary to ANNs, SVMs or ARIMA estimation methods
- The error that we make in our predictions are bounded with a computable theoretical expression and under certain circumstances this bound is the best that we can obtain for any forecaster
- They extend the SVM and ANN based techniques

For the regret based forecasting, we take away the seasonal part of the PEMS-SF dataset like in the ARIMA process case. Having removed the data seasonality, we use the Vovk-Azoury-Warmuth forecaster to model the resulting data [1]. Formally, let x_t be a vector containing the time index and the day index corresponding to each time period t and y_t be the value of the road occupancy rate for each time period t, the Vovk-Azoury-Warmuth forecaster p_t is defined by:

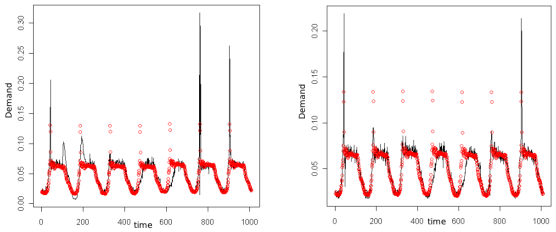
$$\begin{aligned} p_t &= < w_t, x_t > \\ A_t &= I + \sum_{s=1}^{t} x_s x_s^T \\ w_t &= A_t^{-1} \sum_{s=1}^{t-1} y_s x_s \end{aligned}$$

A specificity of the regret based algorithm is that it can benefit from new data points after the learning step to improve its forecasts. Contrary to ANN, SVR or ARIMA process for which the learning process has to be repeated to take into account recent data, the regret based techniques is an online forecasting tool. To be more precise, it is made of two phases:

- Exploration: the algorithm gathers data so as to extract the most reliable information. The forecaster explores all the possible prediction values and keeps in memory its regret performance.
- Exploitation: the algorithm exploits the information that it has already acquired and optimizes its estimates by selecting the best forecaster.

This two stage process explains why the regret algorithm might need some time to learn over the training set and why its learning capacity might be observed only after a sufficiently large number of iterations.

We can check the good fit of the regret based algorithm by showing the output produced by the input data of the second week of the learning data in Figure 9 (a) and of the data points corresponding to the week between day 67 and 74 in Figure 9 (b), and by comparing it with the true values. The black line represents the true measurements while the red points are the estimated ones, given by the regret based algorithm. We



(a) (b)

Figure 9: Learning with a regret based algorithm over a week.

can notice that the algorithm takes benefit from data points after the learning phase to improve its forecasts contrary to neural networks, support vector machines and ARIMA processes as it is an online forecasting tool.
We compare the performance of the regret-based technique to the support vector machine method

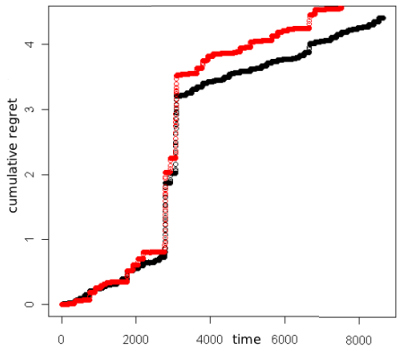


Figure 10: Comparison of the cumulative regret of regret based (black) and SVM (red) methods.

by plotting the cumulative regret obtained by these two techniques in Figure 10. The cumulative regret is the sum of the prediction errors made from time period 1 to the considered time period t and the good property of regret-based technique is that they minimize this value in the long run.

One possible extension of our work could be to use the regret based technique to select dynamically the best forecasting period between support vector machines, ARIMA and ANNs.

4 Comparison of the prediction method performances

To compare the performances of the various forecasting methods over the datasets, we use as criterion, the expectation of the mean of the absolute errors:

$$\bar{\mathcal{L}} = \frac{\sum_{t\in\mathcal{S}}\sum_{y\in\mathcal{Y}}|y_t - y|p_t(y)}{|\mathcal{S}|}$$

where the set $\mathcal{S}$ can represent either the training or the test set, $0 < |\mathcal{S}| < +\infty$ denotes its cardinal and $\mathcal{Y}$ is the set containing all the possible road occupancy values over the dataset. At time period t, predictor p_t is a density function over space $\mathcal{Y}$ for the regret based learning method whereas it is a real belonging to $\mathcal{Y}$ in case where ANN, ARIMA process or SVR are used. As a result, in this latter case, the performance criterion can be sligthly modified to give: $\bar{\mathcal{L}} = \frac{\sum_{t\in\mathcal{S}}|y_t - p_t|}{|\mathcal{S}|}$. The methods are ranked from 1 to 5 over both sets. Smaller is the rank, smaller is the mean absolute error i.e., better is the method.

In the last row of Table 1, we have tested a naive learning approach based on the repetition of the road occupancy rates over two consecutive weeks belonging to one learning set and then, to the other. We observe that it is worth using elaborate predictions tools over the considered dataset. Indeed over the training set, the regret based approach, ARIMA process and SVR perform better than the naive approach and over the test set, solely, the ANN performs worse that the naive approach. Over the test set i.e., to perform the prediction task, the regret based algorithm generates the smallest value for the absolute loss followed closely by the SVR.

Learning method v.s. Dataset	Training set	Test set
ANN	5	5
ARIMA	1	3
SVR	3	2
Regret	4	1
Naive	1	4

Table 1: Ranking of the learning algorithms based on the mean absolute errors.

In the last row of Table 1, we have tested a naive learning approach based on the repetition of the road occupancy rates over two consecutive weeks belonging to one learning set and then, to the other. We observe that it is worth using elaborate predictions tools over the considered dataset. Indeed over the training set, the regret based approach, ARIMA process and SVR perform better than the naive approach and over the test set, solely, the ANN performs worse than the naive approach. Over the test set i.e., to perform the prediction task, the regret based algorithm generates the smallest value for the absolute loss followed closely by the SVR.

5 Conclusion

In this article, we compared the performances of 5 learning methods: artificial neural networks, support vector machine methods, ARIMA process and regret based algorithm, over the PEMS-SF database which is a freely available machine learning repository [2]. Over the test set i.e., to perform the prediction task, the regret based algorithm generates the better performances in terms of absolute loss minimization, followed closely by the support vector machine method. As a result, energy demand forecasters should favor one of these techniques.

One possible extension could be to use the regret based technique to select dynamically the best forecasting period between support vector machines, ARIMA and artificial neural networks.

It is worth mentioning that regret methods are all the more interesting since they enable the introduction of game theory to model interdependences between the agents composing the

ecosystem and might provide economic guidelines.

Acknowledgements

This work was supported by the European FP7 project Elvire and the Atomic Energy and Alternative Energies Commission (CEA).

References

[1] Cesa-Bianchi N., Lugosi G., Prediction, Learning, And Games, Cambridge university Press, 2006

[2] Frank, A., Asuncion, A., UCI Machine Learning Repository, [http://archive.ics.uci.edu/ml], university of California Irvine, School of Information and Computer Science, 2010

[3] Le Cadre, H., Potarusov, R., Auliac, C., Energy Demand Prediction: A Partial Information Game Approach, in proc. 1-st European Electric Vehicle Congress, 2011

[4] Le Cadre H., Infrastructure Topology Optimization using the Cross-Entropy Method, to appear in the Journal of the Operational Research Society, doi:10.1057/jors.2012.96

[5] Aizerman M., Braverman E., Rozonoer L., Theoretical foundations of the potential function method in pattern recognition learning, Automation and Remote Control, vol.25, pp.821–837, 1964

[6] Drucker H., Burges C. J. C. Kaufman L., Smola A., Vapnik V., Support Vector Regression Machines, Advances in Neural Information Processing Systems, the MIT Press, vol.9, pp.155–161, 1997

[7] Newbold P., The Principles of the Box-Jenkins Approach, Operational Research Quaterly, vol.26, pp.397–412, 1977