energies

MDPI

Article

Chiller Load Forecasting Using Hyper-Gaussian Nets

Manuel R. Arahal *, Manuel G. Ortega and Manuel G. Satué

Systems Engineering and Automation Department, University of Seville, 41092 Seville, Spain; mortega@us.es (M.G.O.); mgarrido16@us.es (M.G.S.)

* Correspondence: arahal@us.es; Tel.: +34-954-48-73-43

Abstract: Energy load forecasting for optimization of chiller operation is a topic that has been receiving increasing attention in recent years. From an engineering perspective, the methodology for designing and deploying a forecasting system for chiller operation should take into account several issues regarding prediction horizon, available data, selection of variables, model selection and adaptation. In this paper these issues are parsed to develop a neural forecaster. The method combines previous ideas such as basis expansions and local models. In particular, hyper-gaussians are proposed to provide spatial support (in input space) to models that can use auto-regressive, exogenous and past errors as variables, constituting thus a particular case of NARMAX modelling. Tests using real data from different world locations are given showing the expected performance of the proposal with respect to the objectives and allowing a comparison with other approaches.

Keywords: energy consumption prediction; time-series forecasting; neural approximation; hyper-gaussian

Citation: Arahal, M.R.; Ortega, M.G.; Satué, M.G. Chiller Load Forecasting Using Hyper-Gaussian Nets. *Energies* **2021**, *14*, 3479. https://doi.org/10.3390/en14123479

Academic Editor: Jae-Weon Jeong

Received: 11 May 2021
Accepted: 9 June 2021
Published: 11 June 2021

1. Introduction

In the context of energy consumption, chillers are becoming a factor of great importance. In medium to large buildings, chillers are responsible for producing cold water to cover the demand for Heating Ventilation and Air Conditioning (HVAC) and other uses [1]; as a result, a significant fraction of global energy consumption is due to chillers. Recently, operation of chillers has been the subject of research with the aim of reducing energy consumption maintaining operational objectives. For instance, efficient sequencing of different units allows the sharing of cooling load among several machines; these can then operate closer to their respective optimum conditions [2]. To this end different control schemes have been proposed. When available, forecasts of the cooling demand [3] can be used to produce control sequences that are feasible and do not pose a significant strain on the chillers (such as frequent switchings on and off). Moreover, the use of thermal energy storage devices can play a significant role in energy savings if coupled with a suitable forecasting system [4].

Predicting the cooling demand can be posed as a multivariate time-series forecasting problem (see [3,5,6] and references therein). For a given building/facility, several variables play a significant role in cooling load: external temperature, wind, humidity, building capacity and occupancy, desired thermal comfort level, etc. For some of these there might be some forecasts available, for instance weather forecasts can be easily accessed from various sources. However, forecasts for the exact particular location of the building might need models specially developed to such end.

In this paper a energy demand forecasting model is developed having in mind the special features arising from chiller optimization. In particular, ease of design/deployment and adaptiveness of the predictors are key for this application. It is remarkable that social changes can produce changes in the use of the building/facility that affect the energy demand, thus adaptiveness is a sought after quality.

Neural networks based on basis function such as RBFNN [7], GRNN [8,9] and Hyper-Gaussian [10] are specially suited for the above posed problem. The basis functions are

such that the response of each base is significative (different from zero) in just a region of the input space. This makes the derivation of the base parameters a simpler task compared with other Neural Networks (NN) approaches [11].

In this paper a methodology for developing predictors for use in the context of chiller optimization is presented. The proposal uses an Hyper-Gaussian net to provide support for local Auto Regressive Moving Average with eXogenous inputs (ARMAX) models. The concept of local linear models is thus combined with that of Hyper-Gaussian. In the proposal, each node is allowed to have a different metric to compute the distance as well as an associate ARMAX model with its own parameters and inputs.

In this context, and regarding the contributions of the present work, please note that, although hyper-gaussians were proposed long ago, in most implementations the weighted metric employed (also referred to as "Mahalanobis-like distance" [10]) is the same for all nodes. In such cases the Hyper-Gaussian NN becomes equivalent to a regular Radial Basis Function Neural Network (RBFNN) where the input data have been subject to a special normalization using the inverse of the experimental covariance matrix. Furthermore, note that the most used basis functions are radial (unlike hyper-gaussians that are not radial).

The proposed methodology aims at the already presented key features, namely low training requirements (in terms of memory, computation, design parameters and data required). In addition, the local nature of the approximation makes it possible to train the network with fewer input–output training data. Finally, the adaptation of the resulting model can easily be performed on-line, meaning that each new observation, acquired during the predictor use, can be used to fine-tune the predictor. This is of great importance for because: 1—many applications for energy management requiring energy demand prediction might not have access to large data-sets to train the NN predictor, and 2—even if the past data have been gathered they might not be that relevant due to social and environmental changes as well as possible changes in the installations.

The methodology is put to the test with various forecasting problems arising in the context of energy consumption forecasting and in particular chiller optimization. In all cases, direct multi-step ahead forecasting is considered, with horizons (number of steps) that are relevant for the chiller optimization problem. A bibliographical review of related works is presented next and commented upon in the light of the objectives sought for chiller optimization. Then, in Section 3, the forecasting problem is detailed followed by Section 4 where the proposed methodology is presented. The tests used for assessment are explained in Section 5 where the obtained results, including comparisons with other approaches, are shown and discussed.

2. Related Works

In this section, a review of neural models used for forecasting energy consumption is presented, paying special attention to those related to the proposed methodology and others used in the tests for comparison. For a thorough review of artificial intelligence based load forecasting the reader is advised to check [6,12,13] and their references.

Energy demand forecasting is a difficult task due to some characteristics such as non-stationarity (in the mean and/or variance), calendar and other social effects, seasonality effects (weekly cycle, annual cycle), high volatility, jumps and outliers, and, in some cases, lack of long historical data sets [5]. This motivates the research on methods that might prove useful for some particular applications such as cooling of large buildings. Arguably, the first factor to take into account is the forecasting horizon. The case of interest for chiller operation optimization is that of short-term forecasting (horizons from a few hours to days [14]). For medium and long-term forecasting see [15] and the references therein.

The problem of chiller plant optimization has been tackled in many research works. Optimization is usually carried out in static terms (based on pre-specified load profiles) or dynamically by using appropriate control techniques. In the second case, having access to accurate predictions for the variables of interest makes the control techniques more capable of delivering energy saving schemes. Variables such as temperature/humidity/solar

radiation and building/facility occupancy are usually considered. The cooling demand prediction was tackled in [16] using Generalized Regression Neural Networks (GRNN), Ref. [17] using support vector machines, Ref. [5] using Multi Layer Perceptrons (MLP), Ref. [18] using genetic methods that provide simple models by means of feature selection.

GRNN belong to a class of nonparametric estimation methods that can be posed as a special case of basis functions networks. The use of the Parzen Kernel with a set of input–output training pairs produces the well known GRNN method that uses as many bases as training pairs. Training of GRNN is not iterative and just a single parameter (node spread) is usually considered [8,9].

If the Parzen Kernel is replaced with a local linear model and the number of basis functions is restricted to some pairs (or averaged values) then a fusion of linear local models is obtained. In the literature the term local linear RBFNN has been used for such a construct [19]. Unsupervised center location and global spread parameters were used in most cases. For instance, in [20] a recurrent self-organizing map is used to cluster input training data, local autoregressive (AR) models are then developed for each cluster. The case of different spreads for each node is less common but it is treated with either global optimization techniques such as particle swarm optimization [19] or gradient descent [21].

Other means of using linear models have been proposed, in particular the work of [22] is somehow reciprocal of this proposal as the authors use a functional representation for the parameters of an AR model that becomes state-dependent. This has been extended to RBFNN producing the so called RBF-AR model of [23] and other variants [24]. The idea has also been extended to wavelets as in [25] where the local linear models are combined using a wavelet NN trained using Particle Swarm Optimization.

Regarding the use of non Euclidean metrics for the basis function, the higher flexibility of Hyper-gaussians (HG) compared with RBFNN can decrease the problem posed by correlation between components of the input vector. Furthermore, carefully selecting the nodes helps to reduce the number of bases needed for a given approximation. Although many papers remark the possibility of basis functions using a weighted metric, few actually use them. Please notice that in some previous works, the Hyper-Gaussian NN is referred to as "generalized RBF" [26]. This terminology is confusing since there are a handful of other meanings given to the term (e.g., [27]). Another important aspect is that of training of the Hyper-Gaussians. In the seminal paper [10], regularization is introduced as a means to achieve better generalization. A similar approach also appears in later works [28].

On-line adaptation of RBFNN has also been proposed in other papers such as [29] where the basis functions centers and their diagonal covariance matrix are updated by means of a multi-innovation recursive least square. Please notice that schemes for the on-line adaptation of MLP are not that straightforward.

Summarizing the pros and cons of the main NN forecasting models, and in the context of chiller optimization, it is fair to state the following

- MLP. The pro-side is the ability to deal with high dimensional spaces (i.e., large number of inputs). The main drawback is the difficulty to use on-line adaptation.
- RBFNN. The main advantage comes from the existence of constructive algorithms for concurrent training/design and from the possibility of on-line training/adaptation. The main drawback stems from the possible growth on the number of nodes as the input space dimension increases.

The proposed model aims at retaining the good traits of RBFNN avoiding its drawbacks. To this end, Hyper-gaussian nodes are used in conjunction with local models, reducing the possible growth on the number of nodes.

3. Forecasting Problem

To proceed orderly, the forecasting problem of interest for chiller optimization is first presented. A number of time-series are involved, mainly external temperature and building occupancy. Hourly values are assumed in order to be of use for the control system, so the temporal index t represents hours. For each time-series the aim is to compute (at time

$t = h$) a prediction for $t = h + H$ based on measurements available at time $t = h - 1$. A discrete-time control approach is being used here in the notation for ease of connecting the predictor with the rest of the control system driving the optimized operation of the energy system [30].

A direct multi-step forecast strategy is adopted thus the needed prediction will be computed as:

$$\hat{y}(t + H) = N(x(t - 1)), \quad (1)$$

where N is a function (realized by a neural model) of an input vector x whose components are derived from past values of the time-series or from other available information. The selection of the input vector components is discussed later in this Section; such selection takes into account candidate components that are presented in the following.

3.1. Candidate Input Variables

The components of vector x in (1) receive various denominations: regressors, input elements, input variables, features, etc. For the problem at hand, it is possible to make a list of signals that are candidates to be used as actual components of x. The signals in the list will be referred to as candidate input variable or candidate regressors. Please note that the list does not pursue exhaustiveness.

- Type I. Pure autoregressive values. Signals in this group have the general form $y(t - L)$ for some integer lag value $L \geq 1$. Please note that in classical (linear) model identification, lags are consecutive. For the multi-step problem using non-linear approximators it might be advisable to use non-consecutive values of L.
- Type II. Averaged autoregressive values. Signals in this group are derived from y taking the mean (or median) value starting at time $t - L_1$ up to time $t - L_2$. This type of regressor provides smoothed out values and also some information coming from larger lags ($t - L_2$, $t - L_2 + 1$, etc.)
- Type III. Extreme values from a window of past values. The maximum, minimum and combinations such as maximum–minimum values are considered in this group. The window is defined using two lags: $L_1 \geq 1$ and $L_2 > L1$.
- Type IV. Differences of autoregressive values. Signals in this group have the general form: $y(t - L_2) - y(t - L_1)$. This group is specially relevant for the direct multi-step problem.
- Type V. Averaged differences. This group uses averages of signals of type IV.
- Type VI. Temporal indices. Signals in this group are variables of the day within the year, day within the week, hour within the day, etc. Often it is interesting to encode the information in a way that reflects the observed periodicities. For instance, instead of d_y (day within the year) with $d_y \in [1, 365]$ one may encode it as two cyclic variables $z_1(d_y) = \sin 2\pi d_y/365$, $z_2(d_y) = \cos 2\pi d_y/365$. Signals z_1 and z_2 are continuous over years unlike d_y.
- Type VII. Weather forecasts. As stated in the introduction, the meteorological forecast might be available even if for a large area (city, province, state) in which the building/facility is located.
- Type VIII. Social indicators. Signals in this group are useful to encode a priori knowledge about energy demand stemming from social considerations such as the day of the week, festivities, etc.

3.2. Data Preprocessing

As is usually done in forecasting applications, first data should be pre-processed to take care of some traits such as trend, seasonality and outliers. Such pre-processing must be designed having in mind the specific application short-term forecasting of variables relevant for cooling demand. It must also follow the directrices presented in the introduction: ease of design/deployment and adaptiveness. To this end the following steps are taken.

1. Outlier detection. Outliers can be the product of erroneous measurements made by sensors or incorrect treatment of the information (for instance misplaced points in the data-set). Spectral methods can be of use since the basic periods are known (annual cycle, 24 h cycle) and an estimate of the harmonic content for each one can be obtained from data. A simpler approach, however, is to use the point-like value $y(t)$ of the hourly series and compare it with the mean (or median) over a window of data points $\mu = \text{mean}(y(t-L), \ldots, y(t-1), y(t+1), \ldots, y(t+L))$. In this way, if $|y(t) - \mu| > U$ then $y(t)$ is considered an outlier. This requires the selection of just two parameters: L and U. The detected outliers can then be substituted by filtered values [31].
2. Trend removal. There are many reports about the necessity of detrending data prior to its use with NN. One of the simplest approaches is global linear fit performed on the data available for training [31].
3. Seasonality treatment. The kind of time series that are of interest in this application are linked to meteorological condition that show an annual cycle with more or less marked seasons (depending on location). Seasonality can be accommodated by NN models if one includes temporal indications as inputs. Nevertheless a simple procedure can be applied since (in this case) the periodicity is known in advance. Thus, the seasonal average can be computed and its value substracted from all data points.
4. Normalization. Linear scaling to the $[-1, \ +1]$ interval is performed. Please note that in the proposed approach a weighted metric is used and this step is not that important. Nevertheless it will be used to provide a fair ground for comparison with other approaches.

These steps are applied to all data used in the paper for model design, testing and comparisons with other approaches.

3.3. Selection of Regressors

The problem of composing an input vector out of the possible choices given the candidate input variables is referred to as "input variable selection" (IVS) and "variable/feature subset selection". It has been widely reported that it is not an easy task, yet it is a step of paramount importance. In particular, in the general case of NN models, there are no assumptions regarding the structure of the model (unlike the classical linear regression problem) and that couples the IVS problem with the model selection problem. At any rate, factors that difficult the selection process are: the large number of candidate variables, the correlations among them that create redundancy and existence of variables that have little or no predictive power but can correlate with some patterns in the historical data.

There are in the literature different types of algorithms to perform IVS for NN models. Filters, wrappers and embedded methods have been widely used for time-series forecasting [32]. Each one has its own merits and drawbacks. For the application at hand (direct multi-step forecasting for chiller optimization) and the proposed model that will be presented later, it is possible to perform IVS applying a linear correlation time-series analysis in the vein of Box and Jenkins. This is so because, in the proposed model, localicity is exploited so there is no need for wrappers and embedded methods that arise in the general context of NN models. So, step-wise regression will be used to determine which variables (from groups I to VIII above) will form the input vector x used by the predictor given by (1).

Please note that, for typical energy-related forecasting, the filter approach is much less computationally intensive than any other IVS. It also provides insight into the problem as the selected variables can be interpreted through correlations or by means of fitted linear models. These traits are in accordance with the directing lines adopted for this particular application, as stated in the introduction. Moreover, note that some dependency (or correlation) among variables is to be expected. Removal of the redundant ones does reduce the dimension of the candidate list. In this way the computation burden associated with the input variable selection is greatly reduced. This procedure is most advisable in many practical situations. However, just to make sure of the results, the results shown

are based on the complete list using step-wise regression. Please bear in mind that in the comparisons performed later, all models share the same input vector to provide a fair comparison.

4. Proposed Model

Neural models of the MLP type have some positive traits that have made them a choice for forecasting. In particular, the ability to establish complex relationships between variables due to their flexibility and training procedure. However there are implicit drawbacks that need attention: risk of over-fitting producing low generalization, black box nature of the resulting model, difficulties on short datasets, difficulties in on-line adapting an already trained network to incoming new observations [14] and the need for elaborate methods for input variable selection.

On the other hand, RBFNN are three-layer nets similar to MLP with just one hidden layer. In a RBFNN the input vector x is not weighted, rather it is received by each node and a kernel-like function $\Phi()$ is computed based on it. Typical basis functions are Gaussians defined as

$$\Phi(x) = \exp(-d(x,c)^2/\sigma^2), \tag{2}$$

where σ defines the spread of the Gaussian and d is the distance of input vector x from the base center c taken in most cases as the Euclidean distance

$$d(x,c)^2 = (x-c)^T(x-c). \tag{3}$$

The output of a RBFNN for input vector x is computed aggregating the activation of each base in a weighted manner:

$$N(x) = \sum_{n=1}^{Nn} \omega_n \Phi_n(x) = \sum_{n=1}^{Nn} \omega_n \exp(-d(x,c_n)^2/\sigma_n^2), \tag{4}$$

where ω_n is the output weight of node n and Nn is the number of nodes.

In addition to Gaussians, the basis functions can take other forms such as $\Phi(x) = (d(x,c)^2 + \sigma^2)^{1/2}$ (multiquadrics), $\Phi(x) = (d(x,c)^2 + \sigma^2)^{-1/2}$ (inverse multiquadrics). In all cases, the response (activation) of each base is significative (notably different from zero) in just a region of the input space close to its center. This makes the derivation of the base parameters (centers and spreads) an easy task amenable to constructive procedures [11].

In addition, to tune basis output weights (ω_n) fast non-iterative methods such as the pseudo-inverse can be used [33]. This feature also makes it possible to train the network with fewer input–output training data (compared with feedforward MLP, recurrent NN and other approaches). Furthermore, the adaptation of an already trained set of basis functions can easily be performed on-line using each new observation to fine-tune the predictor during its use.

However, a large number of nodes in the hidden layer might be needed to produce the same approximation as an MLP, compromising generalization. Several variations of the basic RBFNN scheme have been proposed to enhance the positive traits and diminish the negative ones. In particular, using local linear models that use the nodes for spatial support. Then each node provides a weight factor to be applied to a local model. The output of the net is the weighted sum for all local models. These can, in principle, be of any type, although simple (linear) models are used in most cases.

The output of a local linear RBFNN (with Gaussian nodes) for input vector x is computed as:

$$N(x) = \sum_{n=1}^{Nn} \omega_n(x) \Lambda_n(x), \tag{5}$$

where $\Lambda_n(x)$ is the output of the n-th local model and ω_n is defined in terms of the GRNN basis as:

$$\omega_n(x) = \exp(-d(x, c_n)^2/\sigma_n^2). \tag{6}$$

With an adequate choice of basis placement (c_n), basis spreads (σ_n) and local models, the needed number of hidden nodes needed can drastically be reduced compared with RBFNN, while retaining the positive traits of RBFNN. In particular, on-line adaptation is possible and an interpretation of the input–output relationship is also available through the local models.

Another early modification of the basic RBFNN scheme is the Hyper-Gaussian of HyperBFNN [10]. The Hyper-Gaussian nodes are similar to those of a Gaussian RBFNN but using a weighted norm defined as

$$d_W(x, c)^2 = ||(x - c)||_W^2 = (x - c)^T W(x - c), \tag{7}$$

where W is a definite positive matrix. For some choices of W, the resulting base is radial (e.g., a W diagonal matrix). In the general case the basis are Gaussians with iso-lines defining ellipses in the input space with an arbitrarily rotated axis.

In this paper the local linear RBFNN concept is used with two important new features: the local models are chosen as ARMAX linear regression models and the supporting basis are Hyper-Gaussians. Moreover, the weighted norm W is independently chosen for each node. The resulting structure receives the name Local Linear Hyper-Gaussian Model (LLHGM).

According to the previous definitions, the local models are defined as:

$$\Lambda_n(x) = \theta_n x, \tag{8}$$

where θ is a vector of model parameters. The activation of each base is calculated as:

$$\omega_n(x) = \exp((x - c_n)^T W_n(x - c_n)), \tag{9}$$

and the output of the LLHGM as:

$$N(x) = \sum_{n=1}^{Nn} \omega_n(x)\Lambda_n(x). \tag{10}$$

4.1. Training of the LLHGM

The design of the predictor includes the selection of the parameters of the Hyper-Gaussians (centers and weighting matrices) and the local model parameters for each HG node. Again, the local nature of the approximation produced by a LLHGM makes it possible to attack each task more or less separately. It is also customary that a last phase of supervised training in which gradient-descent can fine-tune some or all of the parameters.

A typical realization of NN also includes some hyper-parameters (or structural parameters) to be selected. In the case of RBFNN these include maximum number of bases, extreme values for spread factors and heuristics for center placement using the distance to previous nodes. These are usually set by testing using a validation set.

4.1.1. Node Placement

Node placement is the selection of a value for the centers c_n for each n to be used in (9). In the proposed model, as happens with RBFNN and related approaches, the centers can be obtained using different strategies, such as: self-organizing, supervised training, and constructive methods such as RAN [11], GAP [34], C-RBF [26] and their variants.

In principle any method could be used for the proposed model. However, it is important to recall that, in the proposed model, the Hyper-Gaussian basis are used to provide support (in the Mathematical sense of localicity) to the local models. So node placement is not tightly linked to shape in the output space as happens with RBFNN.

This almost rules out supervised training and favors self-organizing over constructive methods. In addition, self-organizing methods such as K-Means are in line with the desired trait of ease of implementation. Finally, boot-strapping K-Means yields placements that are robust (in the sense of being less sensitive to random choice of initial solution) than constructive algorithms.

4.1.2. Weighted Metric

Once the center have been chosen, matrices W_n for each node n are easily derived. Several paths can then be followed, perhaps the simplest one is to consider all data points x_i belonging to the n-th Voronoi region produced by the K-Means phase, arrange them in a matrix $X = (x_i)$ and compute the covariance matrix $C_n = \mathrm{cov}X$ from which W_n is derived simply as

$$W_n = \gamma_n C_n^{-} 1. \tag{11}$$

Parameter γ_n is necessary because the covariance matrix contains information about relative importance (for the weighted metric) of each component of vector x in relation to the others, but it does not provide the width or spread for the Hyper-Gaussian node. This parameter can be set as a global value for all nodes or it can be derived for each node as a function of the distance to its nearest nodes. Whatever the case, it is customary (in RBFNN) to allow some overlapping of basis support, otherwise the input space would not be totally covered up by the set of all nodes. Then, instead of a fixed value for γ, the designer has to come up with a desired value for the overlapping (which might be easier to set as it admits a geometrical interpretation). Different values can be tested and chosen using a test set without much burden.

4.1.3. Local Models

So far just the input space has been used to determine parameters. For the local models it is necessary to include the output values as well. The task is an easy one and can be solved performing a least squares fit $\Lambda_n(x_i) = y_i + \epsilon_i$.

Pseudo-inverse methods can be used with minimum computer power (in comparison with a typical ANN training) to obtain the vector of parameters θ_n for each n. Please note that, as in the previous case, one might just use the Voronoi regions from the K-Means phase to decide which data points x_i are used for each node.

4.1.4. Final Adjustment and Adaptation

The previous steps allow the setting all parameters of the proposed model. It is interesting to remark that just some iterations on the K-Means algorithm and straightforward computations have been needed. With this, the model could be put to work and the adaptation would take care of minor corrections. However, in many cases regarding RBFNN models, one might fine-tune the parameters using gradient-descent.

The chain-rule of derivation allows to back-propagate the output error. In other words, compute the derivative of the squared error with respect to the parameters of the model and use the derivative to slightly modify each parameter seeking a minimum of the error.

Learning rates must be selected keeping in mind how much a given error level is allowed to modify each parameter. This is specially important for the adaptation phase. Moreover, bear in mind that if centers are allowed to move, care has to be taken to prevent the eventual superposition of nodes. Finally, since the design of the model involves little computation it is possible to re-run the whole process if one has doubts about the performance of the adaptation process.

5. Experimental Results

The proposed model is trained on different data-sets to show its adequacy for the objectives that motivate the research. Comparison with other strategies will be presented on the basis of hold-out data posterior in time to that used for training.

5.1. Data-Sets for Design and Comparisons

Three data-sets are considered. T1 consists of hourly temperature values from Atlanta (USA). E1 are hourly energy consumption from Dominion Virginia Power (DOM) with no temperature data linked to them. In these cases the data were obtained from the Kaggle website [35]. E2 consists of hourly energy consumption from a chiller in Seville Spain for which historical temperature values are available. Please note that several locations of the world are represented, thus different climates and social patterns are considered. Furthermore, the data sets correspond to situations of interest for chiller consumption forecasting, including temperature (T1), energy with no attached temperature (E1) and energy with associated temperature (E2).

Please note that, in data-driven forecasting of chiller load, meteorological forecasts might be available even for a large area (city, province, state) in which the building/facility is located. Yet the data-driven forecast for a particular location might be more accurate than the large area forecasts since local factors can be included. This is one reason why set T1 is included. The other reason is that, in order to use set E2, a forecast of temperature is needed. Regarding E1, it is included to be able to compare results in the (perhaps not too common) case where temperature forecasts are not available. The same methodology is applied and for the sake of completeness the results are given further validating the method.

In order to compare different models using different data sets it is convenient to define a figure of merit such as the the Mean Squared Error (MSE) of the predictions. However, the physical units or scale used by the output variable can obscure the interpretation of results. A dimensionless variable that is often used is the MSE of the predictions divided by the MSE of the lazy man's prediction. This is the same as dividing the MSE by the observed variance of the variable of interest. This measure provides an idea of the inherent difficulty to predict a time-series. Consider the following definition:

$$s^2(F, L) = \frac{1}{T}\sum_t (y(t+F) - y(t-L))^2, \tag{12}$$

where F is the prediction horizon, T the number of data points and L is an appropriate lag. For the one-step ahead prediction it makes sense to consider $H = 1$, $L = 1$ (adopting the discrete-time control approach), then $s(1,1)^2$ is a measure of the variability of the time-series when going from a known datum $y(t-1)$ to a future one $y(t+1)$. This figure of merit $s(1,1)$ can be interpreted as the mean squared error that a lazy-predictor would produce. The lazy predictor in this case takes the form $\hat{y}(t+1) = y(t-1)$. Any other prediction should produce less variance and the amount by which the variance is reduced is a measure of the goodness of the predictor. In other words, $s(1,1)$ provides a base-line for comparing predictors.

Now, for a direct H-step ahead predictor and given the fact that a 24 h cycle is involved, it is natural to consider the cases a) ($F = H$, $L = 1$) because $y(t-1)$ is the moment closest to the actual one for which measurements are available, and b) ($F = H$, $L = 24 - H$) because $y(t-24+H)$ corresponds to the same hour as $y(t+F)$ but on the previous day. In fact, the lazy man's prediction for $y(t+F)$ would be $\hat{y}(t+F) = y(t-L)$. Then, the following figure of merit emerges:

$$E = \frac{100}{s^2(F, L)}\sqrt{\frac{1}{T}\sum_{t=1}^{T}(y(t+F) - \hat{y}(t+F))^2}, \tag{13}$$

where y is the real value and $\hat{y}$ the prediction. This measure (E) establishes the degree in which the observed variance has been reduced by the predictor allowing a comparison of results in different data-sets (with possibly different variables and/or units). Please notice that the lazy man's prediction would generate a value of $E = 100$, any other predictor should produce lesser values, in fact, the lower the better.

The $s(F, L)$ statistics are shown in Table 1 for each data set and for each of the values of H considered in the tests. Other characteristics of the data-sets are also included.

Table 1. Main characteristics of the data-sets used in the tests.

Set ID	Scope (Years)	Range	$s^2(6,1)$	$s^2(6,18)$	$s^2(3,1)$	Unit
T1	5	$[-13.7, 37.6]$	5.70	3.91	3.86	(ºC)
E1	5	$[1.2, 21]$	33.6	14.4	23.2	(MW)
E2	5	$[0, 357]$	81.7	27.9	63.5	(KW)

The chiller used at the installation producing the E2 data-set has the following characteristics: 400 (kW) cooling power, $1.5 \times 5 \times 2.2$ (m) in size (including casing), and $4 \cdot 10^3$ (Kg) in weight. It operates producing chilled water at about 5 (ºC) that is sent to the installation from where it returns at a higher temperature depending on the thermal load. In addition, a laboratory version is available that allows inspection of its elements for research purposes as illustrated in Figure 1.

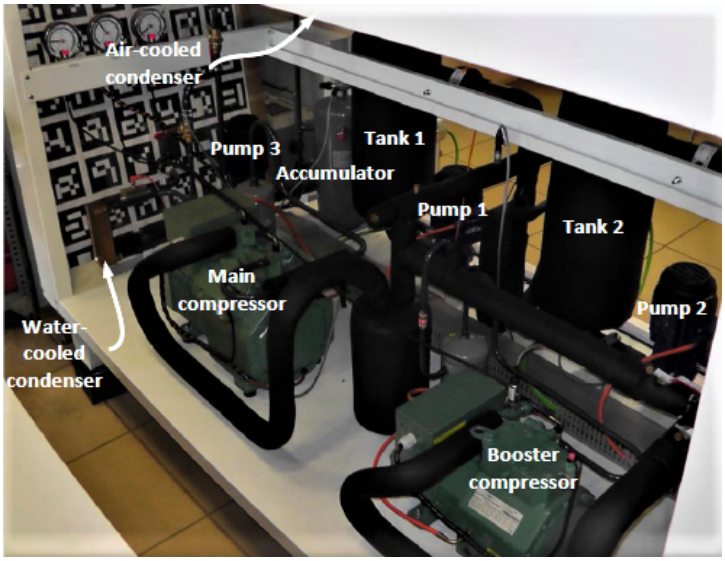

Figure 1. Laboratory version of the chiller used at the installation producing the E2 data-set.

5.2. Input Variables

A thorough search of the combination of variables was performed to obtain the most promising input vector for each data-set. In order to produce a fair comparison, the same input variable is used for all methods under comparison. In Table 2 the chosen input vector is presented for each data-set. The variables come from the list of candidate variables given in Section 3.1. The particular values of the lags and other parameters are presented in Table 3.

Table 2. Composition of the input vector for each test.

Set	Input Variables	$\dim(x)$
T1	$[v_1, v_4, v_5, v_6, v_8, v_9]$	6
E1	$[v_1, v_4, v_5, v_6, v_8, v_9]$	6
E2	$[v_1, v_2, v_4, v_5, v_6, v_8, v_9, v_{10}, v_{11}]$	9

Table 3. Variables considered for the input vector.

Variable	Type	Parameters
v_1	I	$L = 1$
v_2	I	$L = 24 - 6$
v_3	II	$L_1 = 1, L_2 = 4$
v_4	II	$L_1 = 24 - 6, L_2 = 24 - 6 + 3$
v_5	III	$L_1 = 1, L_2 = 24$
v_6	IV	$L_1 = 25, L_2 = 18$
v_7	V	$L_1 = 25, L_2 = 18$
v_8	VI	z_1
v_9	VI	z_2
v_{10}	VII	temperature forecast
v_{11}	VIII	festivity

5.3. Results

The proposed predictor has been tested with the data-sets of Table 1. For comparison purposes, a feed-forward NN was also trained on each data-set. In all cases, 4 years of data were used for model design (structure and parameter estimation) and one additional year (posterior in time) for comparison. The first set is termed historical (H) and was in turn divided into training/testing/validations parts as usual. The second set is termed new (N) and was not used except for the final comparison.

The number of nodes was obtained using the hold-out set technique. The structure of the resulting models is presented in Table 4. Two variants of the proposed algorithm are presented in Table 5, the one without adaptation (LLHGM) and the one with adaptation (LLHGM-a), in both cases the number of nodes is the same.

Table 4. Structure of the compared models.

Model	Type	Nodes T1	Nodes E1	Nodes E2
MLP	Feed-forward NN	12	18	19
LLHGM	Local Linear HG	32	39	47

The forecasting errors are reported in Table 5 using the figure of merit (E) previously introduced and for the particular values $F = 6$, $L = 24 - 6$ since a lead of 6 hours in the forecast is considered. Please bear in mind that number (E) is a dimensionless figure of merit that is independent of scale and allows comparing different models on different data sets. A value of $E = 100$ is obtained by the lazy man's forecast that simply produces $\hat{y}(t + F)$ as $y(t - L)$. Hence, any value less than 100 is an improvement over that simple forecast. It can be seen that the MLP is able to reduce E from 14% to 26% depending on the data-set. The proposed model has a slightly better reduction specially in the (N) section meaning that the generalization is better. The adaptive version of the proposed model achieves even better generalization results as expected.

Table 5. Results obtained in the tests.

Model	T1 (H)	T1 (N)	E1 (H)	E1 (N)	E2 (H)	E2 (N)
MLP	71.16	76.33	63.68	68.31	64.39	86.09
LLHGM	71.54	72.13	63.54	65.42	64.04	82.29
LLHGM-a	71.54	70.42	63.54	65.01	64.04	79.15

Figure 2 shows the trajectories of the predicted variable y and the forecasts $\hat{y}$ for the MLP and the proposed model. The temporal span is a a week and the obtained corresponds to the (N) section of each data-set. It can be seen that the worst predictions correspond to the E1 data-set. Please recall that no meteorological information is used for this case, just

auto-regressive values. It can also be seen that in data-set E2 the use of a temperature and social indicator allows the creation of better predictions compared with the E1 case.

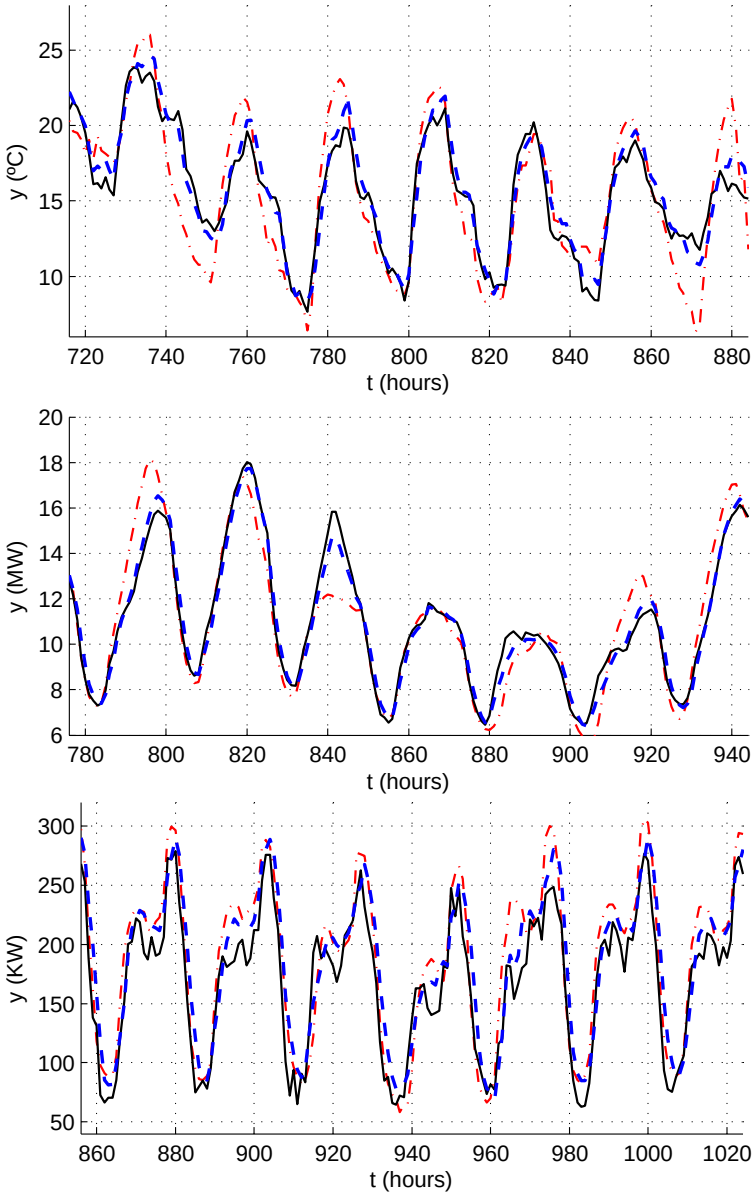

Figure 2. Forecasting results in 7 days of new data for the three datasets (**Top**) T1 (**Middle**) E1 (**Bottom**) E2. In all cases the predicted variable is presented with a dash-dotted red line, the forecast given by the MLP is the black solid line and the LLHGM forecast is shown as a blue dashed line.

Figure 3 shows the histogram of errors obtained for data-set E2 using the proposed model with and without adaptation. The data points used in the histogram belong to the (N) section of the data-set and comprise 3 months of hourly data. This allows the validation of the model using data not previously used in the design and training of the model.

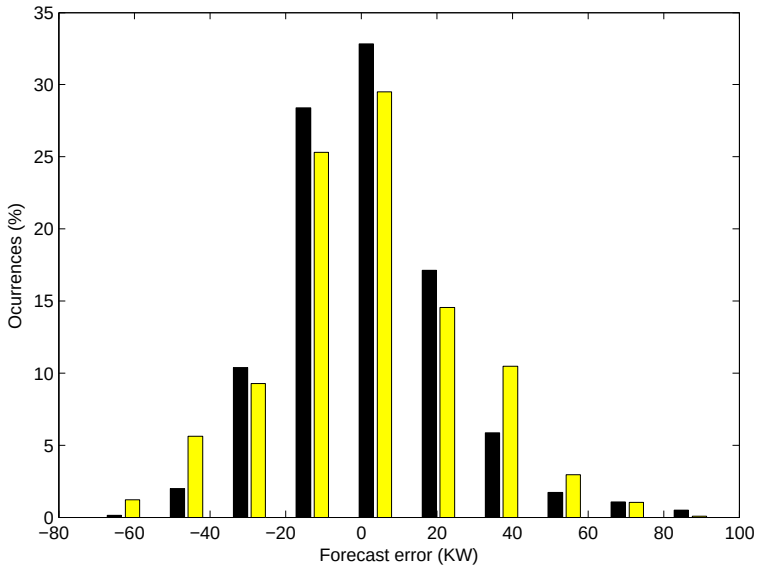

Figure 3. Histogram of errors obtained for data-set E2 using the proposed model with (black bars) and without adaptation (yellow bars).

6. Discussion

The results show how the proposed scheme compares with a classical approach such as a Multi Layer Perceptron. Please notice that the features sought by this proposal are easy implementation and on-line adaptation capability. Yet the results show that the proposal can also compete in terms of forecasting accuracy, especially with new data (i.e., data encountered during the use of the predictor after training). The authors believe that this result comes from two sources: a) the local nature of the approximation made by the model and b) the low complexity of the local linear models. This prevents (or limits the possibility of) the LLHGM from taking the course of other approaches (such as MLP) to bad generalization.

The ease of on-line adaptation provides a further reduction in forecasting error which is potentially a relief for the changing of load patterns due to socio-economic effects. These changes are likely to appear in connection with energy consumption and, in particular, with building usage and the associated HVAC needs.

Author Contributions: Conceptualization and methodology M.R.A. and M.G.O.; software, M.G.S.; validation, M.R.A., M.G.O. and M.G.S.; formal analysis and investigation, M.R.A., M.G.O. and M.G.S.; resources and data curation M.R.A. and M.G.S.; writing—original draft preparation, writing—review and editing, and visualization, M.R.A., M.G.O. and M.G.S.; supervision, project administration and funding acquisition, M.G.O. and M.R.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research (including the APC) was funded as Proyecto RTI2018-101897-B-I00 by FEDER/Ministerio de Ciencia e Innovación–Agencia Estatal de Investigación.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data-sets T1 and E1 can be found at the Kaggle website [35].

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AR	Auto Regressive
ARMA	Auto Regressive Moving Average
ARMAX	Auto Regressive Moving Average with eXogenous inputs
GRNN	Generalized Regression Neural Network
HG	Hyper Gaussian
HVAC	Heating, Ventilation and Air Conditioning
LLHGM	Local Linear Hyper-Gaussian Model
MLP	Multi Layer Perceptron
RBFNN	Radial Basis Function Neural Network
NN	Neural Network
SVM	Support Vector Machine

References

1. Bejarano, G.; Vivas, C.; Ortega, M.G.; Vargas, M. Suboptimal hierarchical control strategy to improve energy efficiency of vapour-compression refrigeration systems. *Appl. Therm. Eng.* **2017**, *125*, 165–184. [CrossRef]
2. Thangavelu, S.R.; Myat, A.; Khambadkone, A. Energy optimization methodology of multi-chiller plant in commercial buildings. *Energy* **2017**, *123*, 64–76. [CrossRef]
3. Kim, J.H.; Seong, N.C.; Choi, W. Modeling and optimizing a chiller system using a machine learning algorithm. *Energies* **2019**, *12*, 2860. [CrossRef]
4. Rodríguez, D.; Bejarano, G.; Vargas, M.; Lemos, J.M.; Ortega, M.G. Modelling and cooling power control of a TES-backed-up vapour-compression refrigeration system. *Appl. Therm. Eng.* **2020**, *164*, 114415. [CrossRef]
5. Mena, R.; Rodríguez, F.; Castilla, M.; Arahal, M.R. A prediction model based on neural networks for the energy consumption of a bioclimatic building. *Energy Build.* **2014**, *82*, 142–155. [CrossRef]
6. Runge, J.; Zmeureanu, R. Forecasting energy use in buildings using artificial neural networks: A review. *Energies* **2019**, *12*, 3254. [CrossRef]
7. Powell, M.J. Radial basis functions for multivariable interpolation: A review. In *Algorithms for Approximation*; ACM: New York, NY, USA, 1987.
8. Nadaraya, E.A. On estimating regression. *Theory Probab. Appl.* **1964**, *9*, 141–142. [CrossRef]
9. Specht, D.F. A general regression neural network. *IEEE Trans. Neural Netw.* **1991**, *2*, 568–576. [CrossRef] [PubMed]
10. Poggio, T.; Girosi, F. Networks for approximation and learning. *Proc. IEEE* **1990**, *78*, 1481–1497. [CrossRef]
11. Platt, J. A resource-allocating network for function interpolation. *Neural Comput.* **1991**, *3*, 213–225. [CrossRef]
12. Ruiz, L.G.B.; Cuéllar, M.P.; Calvo-Flores, M.D.; Jiménez, M.D.C.P. An application of non-linear autoregressive neural networks to predict energy consumption in public buildings. *Energies* **2016**, *9*, 684. [CrossRef]
13. Raza, M.Q.; Khosravi, A. A review on artificial intelligence based load demand forecasting techniques for smart grid and buildings. *Renew. Sustain. Energy Rev.* **2015**, *50*, 1352–1372. [CrossRef]
14. Rodríguez, F.; Martín, F.; Fontán, L.; Galarza, A. Very Short-Term Load Forecaster Based on a Neural Network Technique for Smart Grid Control. *Energies* **2020**, *13*, 5210. [CrossRef]
15. Bu, S.J.; Cho, S.B. Time Series Forecasting with Multi-Headed Attention-Based Deep Learning for Residential Energy Consumption. *Energies* **2020**, *13*, 4722. [CrossRef]
16. Ben-Nakhi, A.E.; Mahmoud, M.A. Cooling load prediction for buildings using general regression neural networks. *Energy Convers. Manag.* **2004**, *45*, 2127–2141. [CrossRef]
17. Li, Q.; Meng, Q.; Cai, J.; Yoshino, H.; Mochida, A. Applying support vector machine to predict hourly cooling load in the building. *Appl. Energy* **2009**, *86*, 2249–2256. [CrossRef]
18. Dulce-Chamorro, E.; Javier Martinez-de Pison, F. Parsimonious Modelling for Estimating Hospital Cooling Demand to Improve Energy Efficiency. *Logic J. IGPL* **2021**. [CrossRef]
19. Nekoukar, V.; Beheshti, M.T.H. A local linear radial basis function neural network for financial time-series forecasting. *Appl. Intell.* **2010**, *33*, 352–356. [CrossRef]
20. Koskela, T.; Varsta, M.; Heikkonen, J.; Kaski, K. Time series prediction using recurrent SOM with local linear models. *Int. J. Knowl. Based Intell. Eng. Syst.* **1998**, *2*, 60–68.
21. Patra, A.; Das, S.; Mishra, S.; Senapati, M.R. An adaptive local linear optimized radial basis functional neural network model for financial time series prediction. *Neural Comput. Appl.* **2017**, *28*, 101–110. [CrossRef]
22. Chen, R.; Tsay, R.S. Functional-coefficient autoregressive models. *J. Am. Stat. Assoc.* **1993**, *88*, 298–308.
23. Gan, M.; Philip Chen, C.; Chen, L.; Zhang, C.Y. Exploiting the interpretability and forecasting ability of the RBF-AR model for nonlinear time series. *Int. J. Syst. Sci.* **2016**, *47*, 1868–1876. [CrossRef]
24. Gan, M.; Peng, H.; Peng, X.; Chen, X.; Inoussa, G. A locally linear RBF network-based state-dependent AR model for nonlinear time series modeling. *Inf. Sci.* **2010**, *180*, 4370–4383. [CrossRef]

25. Chen, Y.; Yang, B.; Dong, J. Time-series prediction using a local linear wavelet neural network. *Neurocomputing* **2006**, *69*, 449–465. [CrossRef]
26. Zhu, Q.; Cai, Y.; Liu, L. A global learning algorithm for a RBF network. *Neural Netw.* **1999**, *12*, 527–540. [CrossRef]
27. Fernández-Navarro, F.; Hervás-Martínez, C.; Sanchez-Monedero, J.; Gutiérrez, P.A. MELM-GRBF: A modified version of the extreme learning machine for generalized radial basis function neural networks. *Neurocomputing* **2011**, *74*, 2502–2510. [CrossRef]
28. Mahdi, R.N.; Rouchka, E.C. Reduced HyperBF networks: Regularization by explicit complexity reduction and scaled Rprop-based training. *IEEE Trans. Neural Netw.* **2011**, *22*, 673–686. [CrossRef]
29. Chen, H.; Gong, Y.; Hong, X. Online modeling with tunable RBF network. *IEEE Trans. Cybern.* **2013**, *43*, 935–947. [CrossRef]
30. Martell, M.; Rodríguez, F.; Castilla, M.; Berenguel, M. Multiobjective control architecture to estimate optimal set points for user comfort and energy saving in buildings. *ISA Trans.* **2020**, *99*, 454–464. [CrossRef]
31. Adya, M.; Collopy, F.; Armstrong, J.S.; Kennedy, M. Automatic identification of time series features for rule-based forecasting. *Int. J. Forecast.* **2001**, *17*, 143–157. [CrossRef]
32. May, R.; Dandy, G.; Maier, H. Review of input variable selection methods for artificial neural networks. *Artif. Neural Netw. Methodol. Adv. Biomed. Appl.* **2011**, *10*, 16004.
33. Chen, S.; Billings, S.A.; Cowan, C.F.; Grant, P.M. Practical identification of NARMAX models using radial basis functions. *Int. J. Control* **1990**, *52*, 1327–1350. [CrossRef]
34. Huang, G.B.; Saratchandran, P.; Sundararajan, N. An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks. *IEEE Trans. Syst. Man Cybern. Part B* **2004**, *34*, 2284–2292. [CrossRef] [PubMed]
35. Mulla, R. Hourly Energy Consumption-Dominion Virginia Power (DOM), v3. 2018. Available online: https://www.kaggle.com/ (accessed on 30 December 2020).