Stream-Based Electricity Load Forecast

João Gama^{1,2} and Pedro Pereira Rodrigues^{1,3}

¹LIAAD - INESC Porto L.A.

²Faculty of Economics, University of Porto

³Faculty of Sciences, University of Porto

Rua de Ceuta, 118 - 6 andar, 4050-190 Porto, Portugal

jgama@fep.up.pt pprodrigues@fc.up.pt

Abstract. Sensors distributed all around electrical-power distribution networks produce streams of data at high-speed. From a data mining perspective, this sensor network problem is characterized by a large number of variables (sensors), producing a continuous flow of data, in a dynamic non-stationary environment. Companies make decisions to buy or sell energy based on load profiles and forecast. We propose an architecture based on an online clustering algorithm where each cluster (group of sensors with high correlation) contains a neural-network based predictive model. The goal is to maintain in real-time a clustering model and a predictive model able to incorporate new information at the speed data arrives, detecting changes and adapting the decision models to the most recent information. We present results illustrating the advantages of the proposed architecture, on several temporal horizons, and its competitiveness with another predictive strategy.

1 Motivation

Electricity distribution companies usually set their management operators on SCADA/DMS products (Supervisory Control and Data Acquisition / Distribution Management Systems). Load forecast is a relevant auxiliary tool for operational management of an electricity distribution network, since it enables the identification of critical points in load evolution, allowing necessary corrections within available time. In SCADA/DMS systems, the load forecast functionality has to estimate, for different horizons, certain types of measures which are representative of system's load: active power, reactive power and current intensity. Given its practical application and strong financial implications, electricity load forecast has been targeted by innumerable works, mainly relying on the non-linearity and generalizing capacities of neural networks (ANN), which combine a cyclic factor and an auto-regressive one to achieve good results [4]. Nevertheless, static iteration-based training, usually applied to train ANN, is not adequate for high speed data streams. On current real applications, data is being produced in a continuous flow at high speed. In this context, faster answers are usually required, keeping an anytime model of the data, enabling better decisions. Moreover, a predictive system may be developed to serve a set of thousands of load

sensors, but the load demand values tend to follow a restricted number of profiles, considerably smaller than the total set of sensors. Clustering of sensors greatly allows the reduction of necessary predictive models. However, most work in data stream clustering has been concentrated on example clustering and less on variable clustering [7].

The paper is organized as follows. In the next section we present the general architecture of the system, main goals and preprocessing problems with sensors data, the clustering module, and the incremental predictive models. Section 3 presents the evaluation methodology and preliminary results using real data from an electricity network. Last section resumes the lessons learned and future work.

2 General Description

The main objective of this work is to present an incremental system to continuously predict in real time the electricity load demand, in huge sensor networks. The system must predict the value of each individual sensor with a given temporal horizon, that is, if at moment t we receive an observation of all network sensors, the system must execute a prediction for the value of each variable (sensor) for the moment $t + k$. In this scenario, each variable is a time series and each new example included in the system is the value of one observation of all time series for a given moment. Our approach is to first cluster the sensors using an online data stream sensor clustering algorithm, and then associate to each cluster a ANN trained incrementally with the centroid of the cluster. Overall, the system predicts all variables in real time, with incremental training of ANN and continuous monitoring of the clustering structure.

Pre-processing Data. The electrical network spreads out geographically. Sensors send information at different time scales and formats: some sensors send information every minute, others send information each hour, etc.; some send the absolute value of the variable periodically, while others only send information when there is a change in the value of the variable. Sensors act in adversary weather and battery conditions. The available information is noisy. To reduce the impact of noise, missing values, and different granularity, data is aggregated and synchronized in time windows of 15 minutes. This is done in a server, in a pre-processing stage, and was motivated by the fact that it allows to instantiate sensor values for around 80% of the sensors. Data comes in the form of tuples: $\langle date, time, sensor, measure, value \rangle$. All pre-processing stages (agglomeration and synchronization) require one single scan over the incoming data.

Incremental Clustering of Data Streams. Data streams usually consist of variables producing examples continuously over time at high-speed. The basic idea behind clustering time series is to find groups of variables that behave similarly through time. Applying variable clustering to data streams, requires to incrementally compute dissimilarities. The goal of an incremental clustering system for streaming time series is to find (and make available at any time t) a partition of the streams, where streams in the same cluster tend to be more

alike than streams in different clusters. In electrical networks there are clear clusters of demands (like sensors placed near towns or in countryside) which evolve smoothly over time. We believe that a top-down hierarchical approach to the clustering problem is the most appropriate as we do not need to define *a priori* the number of clusters and allow an analysis at different granularity levels.

The system uses the ODAC clustering algorithm which includes an incremental dissimilarity measure based on the correlation between time series, calculated with sufficient statistics gathered continuously over time. There are two main operations in the hierarchical structure of clusters: *expansion* that splits one cluster into two new clusters; and *aggregation* that aggregates two clusters. Both operators are based on the diameters of the clusters, and supported by confidence levels given by the Hoeffding bounds. The main characteristic of the system is the monitoring of those diameters. In ODAC, the dissimilarity between variables a and b is measured by an appropriate metric, the $rnomic(a, b) = \sqrt{(1 - corr(a, b))/2}$, where $corr(a, b)$ is the Pearson's correlation coefficient. More details can be found in [7]. For each cluster, the system chooses two variables that define the diameter of that cluster (those that are less correlated). If a given heuristic condition is met on this diameter, the system splits the cluster in two, assigning each of those variables to one of the two new clusters. Afterwards, the remaining variables are assigned to the cluster that has the closest pivot (first assigned variables). The newly created leaves start new statistics, assuming that only the future information will be useful to decide if the cluster should be split.

A requirement to process data streams is change detection. In electrical networks and for long term conditions, the correlation structure evolves smoothly. The clustering structure must adapt to this type of changes. In a hierarchical structure of clusters, considering that the data streams are produced by a stable concept, the intra-cluster dissimilarity should decrease with each split. For each given cluster C_k , the system verifies if older split decision still represents the structure of data, testing the diameters of C_k , C_k 's sibling and C_k 's parent. If diameters are increasing above parent's diameter, changes have occurred, so the system aggregates the leaves, restarting the sufficient statistics for that group.

The presented clustering procedure is oriented towards processing high speed data streams. The main characteristics of the system are constant memory and constant time in respect to the number of examples. In ODAC, system space complexity is constant on the number of examples, even considering the infinite amount of examples usually present in data streams. An important feature of this algorithm is that every time a split is performed on a leaf with n variables, the global number of dissimilarities needed to be computed at the next iteration diminishes at least $n - 1$ (worst case scenario) and at most $n/2$ (best case scenario). The time complexity of each iteration of the system is constant given the number of examples, and decreases with every split occurrence. Figure 1 presents the resulting hierarchy of the clustering procedure.

Incremental Learning of ANN. In this section we describe the predictive module of our system. Each group defined by the cluster structure has a feed-forward MLP ANN attached, which was initially trained with a time series representing

Fig. 1. ODAC hierarchy in the Electrical Network (~ 2500 sensors)

the global load of the sensor network, using only past data. The ANN is incrementally trained with incoming data, being used to predict future values of all the sensors in the cluster.

At each moment t , the system executes two actions: one is to predict the moment $t + k$; the other is to back-propagate in the model the error, obtained by comparing the current real value with the prediction made at time $t - k$. The error is back-propagated through the network only once, allowing the system to cope with high speed streams. Although the system builds the learning model with the centroid of the group, the prediction is made for each variable independently. Every time a cluster is split, the offspring clusters inherit the parent's model, starting to fit a different copy separately. This way, a specification of the model is enabled, following the specification of the clustering structure. When an aggregation occurs, due to changes in the clustering structure, the new leaf starts a new predictive model.

The goal of our system is to continuously maintain a prediction for three time horizons: next hour, one day ahead, and one week ahead. This means that after a short initial period, we have three groups of predictions: prediction for the next hour, 24 predictions for the next 24 hours, and 168 predictions for the next week. For the purposes of this application in particular, all predictions are hourly based. For all the horizon forecasts, the clustering hierarchy is the same but the predictive model at each cluster may be different.

The structure of the MLP consists of 10 inputs, 4 hidden neurons (tanh-activated) and a linear output. The input vector for next hour prediction at time t is t minus $\{1, 2, 3, 4\}$ hours and t minus $\{7, 14\}$ days. As usual [6], we consider also 4 cyclic variables, for hourly and weekly periods ($\sin$ and $\cos$). The choice of the networks topology and inputs was mainly motivated by experts suggestion, autocorrelation analysis and previous work with batch approaches [4]. One implication of the chosen inputs is that we no longer maintain the property of processing each observation once¹. Thus, we introduce a buffer (window with the most recent values) strategy. The size of the buffer depends on the horizon forecast and data granularity and is at most two weeks. Figure 2 presents a general description of the procedure executed at each new example.

ANNs are powerful models that can approximate any continuous function [2] with arbitrary small error with a three layer network. The *mauvaise reputation*

¹ A property that the clustering algorithm satisfies.

Fig. 2. Buffered Online Predictions: 1. new real data arrives (r) at time stamp i , substituting previously made prediction (o); 2. define the input vector to predict time stamp i ; 3. execute prediction (t) for time stamp i ; 4. compute error using predicted (t) and real (r) values; 5. back-propagate the error one single time; 6. define input vector to predict time stamp i plus one hour; 7. execute prediction of next hour (p); 8. discard oldest real data (d).

of ANNs comes from slower learning times. Two other known problems of the generalization capacity of neural networks are overfitting and large variance. In our approach the impact of overfitting is reduced due to two main reasons. First we use a reduced number of neurons in the hidden layer. Second, each training example is propagated and the error backpropagated through the network only once, as data is abundant and flow continuously. This is a main advantage, allowing the neural network to process an infinite number of examples at high speed. Another advantage is the smooth adaptation in dynamic data streams where the target function evolves over time. Craven and Shavlik [2] argue that the inductive bias of neural networks is the most appropriate for sequential and temporal prediction tasks. However, this flexibility implies an increase on error variance. In stationary data streams the variance shrinks when the number of examples goes to infinity. In dynamic environments where the target function changes, the variance of predictions is problematic. An efficient variance reduction method is the *dual perturb and combine* [3] algorithm. It consists on perturbing each test example several times, adding white noise to the attribute-values, and predicting each perturbed version of the test example. The final prediction is obtained by aggregating (usually by averaging) the different predictions. The method is directly applicable in the stream setting because multiple predictions only involves test examples. We use the *dual perturb and combine* algorithm to reduce the variance exhibited by neural networks and to estimate a confidence for predictions. For continuous and derivable functions over time one simple prediction strategy, reported elsewhere to work well, consists of predicting for time t the value observed at time $t - k$. A study on the autocorrelation in the time series used to train the scratch neural network reveals that for next hour forecasts $k = 1$ is the most autocorrelated value, while for next day and next week the most autocorrelated one is the corresponding value one week before ($k = 168$). The Kalman filter is widely used in engineering for two main purposes: for

combining measurements of the same variables but from different sensors, and for combining an inexact forecast of system's state with an inexact measurement of the state [5]. We use Kalman filter to combine the neural network forecast with the observed value at time $t - k$, where k depends on the horizon forecast as defined above. The one dimensional Kalman filters works by considering $\hat{y}_i = \hat{y}_{i-1} + K(y_i - \hat{y}_{i-1})$, where $\sigma_i^2 = (1 - K)\sigma_{i-1}^2$ and $K = \frac{\sigma_{i-1}^2}{\sigma_{i-1}^2 + \sigma_r^2}$.

3 Experimental Evaluation

The electrical network we are studying contains more than 2500 sensors spreaded out over the network, although some of them have no predictive interest. The measure of interest is current intensity (I). There are 565 High Tension (HT) sensors, 1629 Mean Tension (MT) sensors, and 299 Power Transformers (PT) sensors. We consider around three years of data, aggregated in an hourly basis, unless a fault was detected. The analysis of results were aggregated by month. The system makes forecasts for next hour, one day ahead, and one week ahead. At each time point t , the user can consult the forecast for next hour, next 24 hours and all hours for the next week. The design of the experimental evaluation in streams is not an easy task. For each point in time and sensor we have an estimate of the error. This estimate evolves over time. To have insights about the quality of the model these estimates must be aggregated. In this particular application, there are natural time windows for aggregations: week windows and month windows. For all time horizons, we aggregate the error estimates by month and type of sensor, for a one year test period. The quality measure usually considered in electricity load forecast is the MAPE (*Mean Absolute Percentage Error*) defined as $MAPE = \sum_{i=1}^n \frac{|(\hat{y}_i - y_i)/y_i|}{n}$, where y_i is the real value of variable y at time i and $\hat{y}_i$ is the corresponding predicted value. In this work, we prefer to use as quality measure the MEDAPE (*Median Absolute Percentage Error*) to reduce sensibility to outliers [1]. Table 1 presents global results for predicting the load over all horizons and on all sensors. We can stress that the system is stable over time, with acceptable performance. All experiments reported here ran in a AMD Athlon(tm) 64 X2 Dual Core Processor 3800+ (2GHz). The system processes around 30000 points per second with a total running time for all the experiments reported here of about 1 hour. For a 24 hours forecast, electricity load demand has a clear daily pattern, where we can identify day and night, lunch and dinner time. For a single forecast at time t , the historical inputs are: $t - \{24h, (168 - 24)h, 168h, 169h, (168 + 24)h, 336h\}$. The results for the 24 hours ahead forecast are also presented in Table 1. In comparison with the one hour forecast, the level of degradation in the predictions is around 2-3%. For the one week ahead load forecast, the standard profile is also well defined: five quite similar week days, followed by two weekend days. As for the 24 hours forecast, several strategies could be designed for one week ahead forecast. Our lab experiments pointed out consistent

Table 1. Median of MEDAPE for all sensors by month, for three different horizons

1 Hour Ahead					1 Day Ahead					1 Week Ahead				
I, %	HT	MT	PT	All	I, %	HT	MT	PT	All	I, %	HT	MT	PT	All
Jan	4.34	4.98	4.60	4.63	Jan	6.61	6.52	6.98	6.44	Jan	5.95	6.10	6.50	5.95
Feb	4.24	5.07	4.74	4.73	Feb	6.97	6.83	7.14	6.73	Feb	6.81	6.87	6.95	6.68
Mar	4.24	5.01	4.60	4.66	Mar	7.23	7.09	7.74	7.03	Mar	7.14	7.49	7.69	7.27
Apr	4.46	5.38	5.08	4.98	Apr	8.05	7.71	9.12	7.75	Apr	7.13	7.31	8.10	7.17
May	3.90	4.77	4.45	4.38	May	6.79	6.42	7.46	6.41	May	5.57	6.17	6.32	5.97
Jun	3.93	4.91	4.58	4.55	Jun	7.23	7.21	8.56	7.21	Jun	6.22	6.79	7.02	6.58
Jul	3.87	4.62	4.25	4.26	Jul	7.13	6.98	7.77	6.95	Jul	7.02	7.38	7.40	7.11
Aug	3.68	4.30	3.89	3.98	Aug	6.97	6.20	7.06	6.22	Aug	7.99	8.11	9.10	7.96
Sep	4.33	4.93	4.42	4.59	Sep	6.99	6.83	7.46	6.80	Sep	6.14	6.69	6.86	6.46
Oct	4.50	5.19	4.67	4.84	Oct	8.03	7.38	8.25	7.41	Oct	6.41	6.40	6.94	6.31
Nov	3.89	4.66	4.32	4.37	Nov	7.17	6.87	7.86	6.87	Nov	6.39	5.97	6.49	5.91
Dec	4.34	5.18	4.65	4.84	Dec	8.62	8.02	8.73	7.96	Dec	9.02	8.58	8.85	8.48

advantages using the simplest strategy of a single forecast using the historical inputs $t - \{168h, 169h, (336 - 24)h, 336h, (336 + 24)h, (336 + 168)h\}$. The results for the one week ahead forecast are also presented in Table 1. Again, when comparing these results with one hour ahead forecast, one can observe a degradation of around 2%. At this point we can state that our strategy roughly complies with the requirements presented by the experts. To assess the quality of prediction, we have compared with another predictive system. For the given year, the quality of the system in each month is compared with Wavelets [8]² on two precise variables each month, chosen as relevant predictable streams (by an expert) but exhibiting either low or high error. Results are shown on Table 2, for the 24 variables, over the three different horizons. The Wilcoxon signed ranks test was applied to compare the error distributions, and the corresponding *p-value* is shown (we consider a significance level of 5%). The relevance of the incremental system using neural networks is exposed, with lower error values on the majority of the studied variables. Moreover, it was noticed an improvement on the performance of the system, compared to the predictions made using Wavelets, after failures or abnormal behavior in the streams. Nevertheless, weaknesses arise that should be considered by future work.

4 Conclusions and Future Issues

This paper introduces a system that gathers a predictive model for a large number of sensors data within specific horizons. The system incrementally constructs a hierarchy of clusters and fits a predictive model for each leaf. Experimental results show that the system is able to produce acceptable predictions for different horizons. Focus is given by experts on overall performance of the complete system. The main contribution of this work is the reduction of the human effort needed to maintain the predictive models over time, eliminating the batch cluster analysis and the periodic ANN training, while keeping the forecast quality at competitive levels. Directions for future work are the inclusion of background knowledge such as temperature, holiday, and special events, into the learning process. Moreover, sensor network data is distributed in nature, suggesting the study of ubiquitous and distributed computation.

² Wavelets are the standard method used in the company we are working with.

Table 2. MEDAPE for selected variables of current intensity (I), exhibiting low or high error. Comparison with Wavelets is considered for the three different horizons.

1 Hour Ahead					1 Day Ahead					1 Week Ahead				
Wav	NN	NN-Wav			Wav	NN	NN-Wav			Wav	NN	NN-Wav		
%	%	%	p-value		%	%	%	p-value		%	%	%	p-value	
Low Error					Low Error					Low Error				
Jan	1.69	2.72	1.03	<0.001	Jan	3.50	3.98	0.48	<0.001	Jan	7.07	3.81	-3.26	<0.001
Feb	2.99	2.79	-0.20	0.196	Feb	7.89	5.62	-2.27	<0.001	Feb	7.18	4.28	-2.90	<0.001
Mar	3.63	2.75	-0.88	<0.001	Mar	6.11	6.38	0.27	<0.001	Mar	5.33	3.99	-1.34	<0.001
Apr	2.05	2.58	0.53	0.002	Apr	8.04	5.45	-2.60	<0.001	Apr	14.68	4.74	-9.94	<0.001
May	2.69	2.28	-0.41	<0.001	May	19.47	7.63	-11.84	<0.001	May	14.16	4.05	-10.11	<0.001
Jun	2.33	2.52	0.29	0.051	Jun	3.68	4.26	0.58	0.002	Jun	4.41	3.40	-1.01	<0.001
Jul	2.14	2.12	-0.02	0.049	Jul	5.83	5.61	-0.22	<0.001	Jul	7.13	4.45	-2.68	<0.001
Aug	2.59	2.54	-0.05	0.537	Aug	6.14	3.64	-2.50	<0.001	Aug	4.73	5.96	1.23	0.008
Sep	2.65	2.64	-0.01	0.374	Sep	7.57	7.65	0.08	0.835	Sep	10.03	3.73	-6.30	<0.001
Oct	2.28	2.36	0.08	0.127	Oct	7.05	8.77	1.73	0.001	Oct	6.28	7.34	1.06	0.010
Nov	2.41	2.14	-0.27	0.085	Nov	4.08	4.52	0.44	0.047	Nov	3.15	4.06	0.91	0.003
Dec	3.56	2.97	-0.59	0.029	Dec	9.92	5.70	-4.23	<0.001	Dec	14.02	7.02	-7.00	<0.001
High Error					High Error					High Error				
Jan	9.04	10.34	1.30	<0.001	Jan	9.04	10.34	1.30	<0.001	Jan	19.73	14.91	-4.82	<0.001
Feb	8.51	9.82	1.31	0.002	Feb	8.51	9.82	1.31	0.002	Feb	9.95	10.54	0.59	0.053
Mar	11.52	11.28	-0.24	0.166	Mar	11.52	11.28	-0.24	0.166	Mar	32.18	28.95	-3.23	<0.001
Apr	9.36	12.74	1.38	<0.001	Apr	9.36	12.74	1.38	<0.001	Apr	18.22	17.93	-0.30	0.074
May	12.89	10.54	-2.35	0.035	May	12.89	10.54	-2.35	0.035	May	14.65	10.43	-4.22	<0.001
Jun	6.68	8.10	1.42	<0.001	Jun	6.68	8.10	1.42	<0.001	Jun	8.96	8.11	-0.86	0.373
Jul	14.52	10.68	-3.84	<0.001	Jul	14.52	10.68	-3.84	<0.001	Jul	32.68	21.12	-11.56	<0.001
Aug	11.11	12.27	1.16	0.034	Aug	11.11	12.27	1.16	0.034	Aug	13.19	14.28	1.09	0.062
Sep	10.52	9.81	-0.71	0.656	Sep	10.52	9.81	-0.71	0.656	Sep	30.58	21.71	-8.87	<0.001
Oct	12.45	11.25	-1.20	0.002	Oct	12.45	11.25	-1.20	0.002	Oct	29.44	24.65	-4.79	0.009
Nov	8.85	7.71	-1.14	0.356	Nov	8.85	7.71	-1.14	0.356	Nov	17.19	12.46	-4.72	<0.001
Dec	11.76	10.91	-0.85	0.040	Dec	11.76	10.91	-0.85	0.040	Dec	38.26	45.08	6.82	0.056

Acknowledgement. Thanks to Paulo Santos from EFACEC, the financial support of projects ALES II (POSC/EIA/55340/2004) and RETINAE (PRIME/IDEIA/70/00078). PPR thanks FCT for a PhD Grant (SFRH/BD/29219/2006).

References

1. Armstrong, J.S., Collopy, F.: Error measures for generalizing about forecasting methods: empirical comparisons. *Int. Journal of Forecasting* 8, 69–80 (1992)
2. Craven, M., Shavlik, J.W.: Understanding time-series networks: a case study in rule extraction. *International Journal of Neural Systems* 8(4), 373–384 (1997)
3. Geurts, P.: Dual perturb and combine algorithm. In: *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics* pp. 196–201 (2001)
4. Hippert, H.S., Pedreira, C.E., Souza, R.C.: Neural networks for short-term load forecasting: a review and evaluation. *IEEE Transactions on Power Systems* 16(1), 44–55 (2001)
5. Kalman, R.E.: A new approach to linear filtering and prediction problems. In: *Transaction of ASME - Journal of Basic Engineering*, pp. 35–45 (1960)
6. Khotanzad, A., Afkhami-Rohani, R., Lu, T.-L., Abaye, A., Davis, M., Maratukulam, D.J.: ANNSTLF – A neural-network-based electric load forecasting system. *IEEE Transactions on Neural Networks* 8(4), 835–846 (1997)
7. Rodrigues, P.P., Gama, J., Pedroso, J.P.: ODAC: Hierarchical clustering of time series data streams. In: *SIAM Int. Conf. Data Mining* pp. 499–503 (2006)
8. Rauschenbach, T.: Short-term load forecast using wavelet transformation. In: *Proceedings of Artificial Intelligence and Applications* (2002)