On the Use of Off-the-Shelf Machine Learning Techniques to Predict Energy Demands of Power TAC Consumers

Francisco Natividad, Russell Y. Folk, William Yeoh, and Huiping Cao

Department of Computer Science
New Mexico State University
Las Cruces NM 88003, USA
`{fnativid,rfolk,wyeoh,hcao}@cs.nmsu.edu`

Abstract. The Power Trading Agent Competition (Power TAC) is a feature-rich simulation that simulates an energy market in a smart grid, where software brokers can buy energy in wholesale markets and sell energy in tariff markets to consumers. Successful brokers can maximize their profits by buying energy at low prices in the wholesale market and selling them at high prices to the consumers. However, this requires that the brokers have accurate predictions of the energy consumption of consumers so that they do not end up having excess energy or insufficient energy in the marketplace. In this paper, we conduct a preliminary investigation that uses standard off-the-shelf machine learning techniques to cluster and predict the consumption of a restricted set of consumers. Our results show that a combination of the popular k-means, k-medoids, and DBSCAN clustering algorithm together with an autoregressive lag model can predict, reasonably accurately, the consumption of consumers.

Key words: Smart Grid, Artificial Intelligence, Multi-Agent System, Machine Learning

1 Introduction

With the rise in the production of renewable energy in the residential market as well as the proliferation of electric vehicles, there is a concerted effort to transform the conventional power grid into a "smart grid". A feature of this smart grid is an energy market, where software agents can buy and sell energy. In this energy market, transactions can occur with all players in the current energy grid from conventional energy producers with power plants to conventional energy consumers in the residential market.

Rich simulations such as the *Power Trading Agent Competition* (Power TAC) [1] provide an efficient way for researchers to test different possible characteristics of this market before deployment in the real world. In the Power TAC smart grid simulation, a software agent acts as a broker to buy energy in bulk from a wholesale market and sells energy to consumers in a tariff market. The aim of the broker is to maximize its profits through intelligent bidding strategies in the wholesale market and intelligent tariff designs in the tariff market. This game was developed as a scenario for the annual Trading Agent Competition, a research competition with over a decade of history [2].

For brokers to do well in this competition, one of the key requirements is that they need to be able to predict the energy demands of consumers in the tariff market accurately. An accurate prediction will allow the broker to identify the amount of energy accurately that it needs to purchase in the wholesale market, which can then translate to effective wholesale bidding strategies to purchase energy at the low prices, resulting in larger profits when the energy is sold to the consumers.

In this paper, we report results of our preliminary study, where we use standard off-the-shelf machine learning techniques to identify classes of consumers that have predictable energy requirements. The identification of such classes will allow a broker to design tariffs that specifically target those classes of consumers and exploit their highly predictable energy demands to maximize overall profits. Our results show that a combination of the popular k-means, k-medoids, and DBSCAN clustering algorithm together with an autoregressive lag model can predict, reasonably accurately, the consumption of consumers.

2 Background: Power TAC

Power Trading Agent Competition (Power TAC) [1] is a feature-rich simulation suite available to researchers interested in working on the smart grid problem. Power TAC offers researchers the chance to explore many characteristics of future smart grids by allowing the creation of agents that operate in several different energy markets including the wholesale, the tariff, and the load-balancing markets. The goal of each agent is to acquire energy and sell it at a profit to its customers. This game was designed as a scenario for the annual Trading Agent Competition, a research competition with over a decade of history [2].

The wholesale market attempts to simulate existing energy markets such as the European or North American large energy producer. In Power TAC, the wholesale market is structured as a “day-ahead market,” where the energy is a perishable good, which allows brokers to buy and sell quantities of energy for future delivery. Market structures like this exist across many different types of perishable goods, so finding effective, robust, automated bidding strategies for these markets is a significant research challenge.

The tariff market is where the major portion of energy purchased from the wholesale market is sold. Energy is sold to consumers (e.g., households, offices, etc.) through tariffs offered by the brokers. The overall goal of each broker is to maximize its profit (e.g., by selling energy in the tariff market at a higher price than the purchase price of the energy in the wholesale market). Because of this, the broker wishes to offer competitive tariffs that attract a large pool of consumers.

There are a variety of tariff options available for a broker to publish that allows for consumer and prosumer (e.g., consumers that have access to renewable energy such as solar panels) customers. For example, brokers may structure tariffs that are tiered where energy kept below a given threshold is priced low but the price increases when more is required. Another option is to price energy according to the time of the day or day of the week allowing brokers to sell energy at a higher price during peak hours. Power TAC also models customers with electric vehicles and allows brokers to issue specific tariffs that are specialized to their needs or controllable tariffs that can be interrupted if the energy cost is too great.

Simulated consumers can be broken down into two categories: elemental models and factored models. Elemental models define a consumer profile using granular characteristics such as the number of members in the household, the number of working days of the members, and the number of appliances in the household. However, defining elemental models in a simulation might not be efficient in modeling large-population consumers. To alleviate this limitation, factored models are introduced. Factored models can represent profiles of large consumer populations such as hospitals, campuses, apartment complexes, office buildings, etc.

The third major market is the load-balancing market, which functions as an energy equalizer in the Power TAC simulation. The current constraints of the simulation allow for an infinite supply of energy; that is, brokers will never be short on the energy promised to their customers. However, this requires that a broker that is unable to meet energy demands in other markets purchase the remaining energy in the load-balancing market at much higher than average prices. Because of this, it is in a broker's best interest to accurately predict the demand that it is required to fulfill.

3 Power TAC Consumer Demand Prediction

We now describe our approach to better understand if consumers have highly predictable consumption rates that can be exploited in a Power TAC agent. The high-level idea of our approach is as follows:

(1) We generated data for two sets of experiments. In the first smaller controlled experiment, we focused on household elemental models and vary the number of members in the household and the number of working days of the members. In the second larger uncontrolled experiment, we generated data for both factored and elemental models with their default Power TAC configurations.
(2) We used dimensionality reduction techniques to reduce the dimensions of the data points in order to reduce the training time.
(3) We clustered the data using three off-the-shelf clustering algorithms: k-means++ [3], k-medoids [4], and DBSCAN [5].
(4) We predicted the demand of the clusters using two off-the-shelf prediction methods [6]: an autoregressive lag model and a 2-week moving average predictor.

3.1 Data Generation

In Power TAC, a bootstrap file containing bootstrap data is generated as a unique seed for a new game. The bootstrap data is used as the beginning set of consumption patterns per consumer that a broker is allowed to analyze before the start of a game. The bootstrap file contains game parameters and about two weeks or 360 hours of historical information (e.g., consumer data, weather data, etc.). Once the game begins, a simulation file containing game data is generated using a bootstrap file that began the game.

In this paper, we analyzed power consumption patterns using two experiments – a smaller controlled experiment with customized Power TAC configurations and a larger uncontrolled experiment with default Power TAC configurations. The first smaller and controlled experiment used 35 different configurations by manipulating two characteristics of a household consumer; the ranges of members in a household and working days were set between 1 and 5 and between 1 and 7, respectively. Note that a household consumer is represented virtually by two loads (a base load and controllable load) with four different tariff shifting properties (non-shifting, smart-shifting, regularly-shifting, and randomly-shifting). This resulted in eight different types of virtual consumers. For each of the 8 types of virtual consumers, there were 35 configurations with 100 distinct bootstrap files generated per configuration. Each game had about 58 days, or 1,399 hours, of energy consumption. In total, this experiment generated 28,000 data points of consumption information. The second larger and uncontrolled experiment included all consumers in a default Power TAC game. A typical Power TAC game includes 28 elemental and factored consumers. This experiment produced 100 distinct default bootstrap files and associated game data for a total of 2,800 data points.

Once the data had been generated by the Power TAC games, both the bootstrap and simulation files were prepared for our clustering and prediction algorithms. A modified version of the Power TAC Log Tool[1] was used to perform the extraction of data into a comma separated format (CSV).

The CSV files were transformed into a matrix for bootstrap data **B** and game data **G**. **B** contains about two weeks or 360 hours of historical consumption data points per consumer:

$$\mathbf{B} = \begin{bmatrix} c_{1,0} & \cdots & c_{1,359} \\ \vdots & \vdots & \vdots \\ c_{N,0} & \cdots & c_{N,359} \end{bmatrix} \qquad \text{(bootstrap consumption for } N \text{ consumers)}$$

where $c_{i,j}$ is the energy consumption of consumer i at time step j. Also, the bootstrap data for consumer i is indexed by $\mathbf{B}_i$.

For each bootstrap file, there is an associated simulation file with game data $\mathbf{G}$, where the game data for consumer i is indexed by $\mathbf{G}_i$.

[1] `https://github.com/powertac/powertac-tools`

$$\mathbf{G} = \begin{bmatrix} c_{1,360} & \cdots & c_{1,1758} \\ \vdots & \vdots & \vdots \\ c_{N,360} & \cdots & c_{N,1758} \end{bmatrix} \qquad \text{(simulation consumption for } N \text{ consumers)}$$

All bootstrap and game data are paired and represented by matrix **D**.

$$\mathbf{D} = \begin{bmatrix} \mathbf{B}_1 & \mathbf{G}_1 \\ \vdots & \vdots \\ \mathbf{B}_N & \mathbf{G}_N \end{bmatrix} \qquad \text{(paired consumption data)}$$

The rows in matrix **D** are then shuffled and split in half into a training dataset and a test dataset. Training bootstrap data is represented by $\mathbf{B}_{train}$ and training game data is represented by $\mathbf{G}_{train}$. Similarly, test bootstrap data is represented by $\mathbf{B}_{test}$ and test game data is represented by $\mathbf{G}_{test}$.

3.2 Dimensionality Reduction

Principal Component Analysis (PCA) is a technique that is widely used for applications such as dimensionality reduction, lossy data compression, feature extraction, and data visualization [7]. PCA can be defined as the orthogonal projection of a dataset onto a lower dimensional linear space, known as the principal subspace, such that the variance of the projected data is maximized [8]. Principal components were calculated using *Singular Value Decomposition* (SVD) on the covariance matrix of a training bootstrap dataset. SVD creates three matrices: left singular vectors represented as a matrix **U**, where each column is a unit vector representing a principal component; a singular values matrix **V** that has the variance represented by each principal component; and a right singular matrix, which was ignored. Using the singular values **V**, one can select P principal components from M dimensions to retain a certain percentage of the total variance R using the following equation:

$$R = \frac{\sum_{i=1}^{P} \mathbf{V}_i}{\sum_{i=1}^{M} \mathbf{V}_i} \tag{1}$$

Before applying PCA, the training bootstrap dataset $\mathbf{B}_{\text{train}}$ should be standardized. In other words, the training bootstrap dataset should be rescaled to have zero mean and unit variance using the z-score normalization in Equation (2). Calculating the z-score requires the mean $\mathbb{E}[\mathbf{B}_{\text{train}}]$ and standard deviation $\sigma_{\mathbf{B}_{\text{train}}}$ [9].

$$\text{z-score} = \frac{\mathbf{B}_{\text{train}} - \mathbb{E}[\mathbf{B}_{\text{train}}]}{\sigma_{\mathbf{B}_{\text{train}}}} \tag{2}$$

Once $\mathbf{B}_{train}$ is standardized, SVD was applied to generate the principal components. The principal components were selected by solving Equation (1) with $R \geq 0.9$, representing ninety-percent retained variance. Then, $\mathbf{B}_{train}$ was projected onto a lower dimensional subspace defined by the selected principal components.

3.3 Clustering

We now describe how we clustered the training bootstrap dataset $\mathbf{B}_{\text{train}}$. We used the following off-the-shelf clustering algorithms: k-means++, k-medoids, and DBSCAN. k-means++ is based on a well known partitioning based algorithm called k-means [3]. k-means++ adds a heuristic when initializing the cluster centroids used in k-means, then uses the original k-means algorithm. A known problem with the k-means algorithm is its weakness with the presence of noise, which can cause it to fail to converge [9]. Algorithm 1 presents the pseudocode of k-means++. In the k-means pseudocode, $D(x)$ denotes the shortest distance from a data point to the closest center we have already chosen [3].

Algorithm 1 k-means++ algorithm

```
1: procedure k-MEANS++($X$, $k$)
2:     arbitrarily choose a data point from $X$ as centroid $\mathcal{C}_1$.
3:     for each $i \in \{2, \dots, k\}$, choose a data point from $X$ with probability $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$ as centroid $\mathcal{C}_i$.
4:     repeat
5:         For each $i \in \{1, \dots, k\}$, set the clusters $\mathcal{C}_i$ to be the set of points in $X$ that are closer to $c_i$ than those points that are to $c_j$ for all $j \neq i$.
6:         For each $i \in \{1, \dots, k\}$, set $c_i$ to be the center of mass of all points in $\mathcal{C}_i : c_i = \frac{1}{|\mathcal{C}_i|} \sum_{x \in \mathcal{C}_i} x$.
7:     until $\mathcal{C}$ no longer changes
8: end procedure
```

We also explored the partitioning-based algorithm *Partitioning Around Medoids* (PAM), a popular approximation algorithm of k-medoids designed by Kaufman *et al.* [4]. PAM uses two phases: a build phase, where initial k medoids are selected arbitrarily, and a swap phase, where the algorithm attempts to find a substitution for a current medoid with a non-medoid that reduces the within-cluster distance. PAM is shown in Algorithm 2 and in this paper is referred to as k-medoids.

Algorithm 2 k-medoids PAM algorithm

```
1: procedure k-MEDOIDS($X$, $k$)
2:     arbitrarily choose $k$ data points from $X$ as the initial medoids.
3:     repeat
4:         for each non-medoid data point, (re)assign it to the cluster with the nearest medoid.
5:         select a non-medoid data point, swap with a current medoid that reduces the total within-cluster distance.
6:     until no change
7: end procedure
```

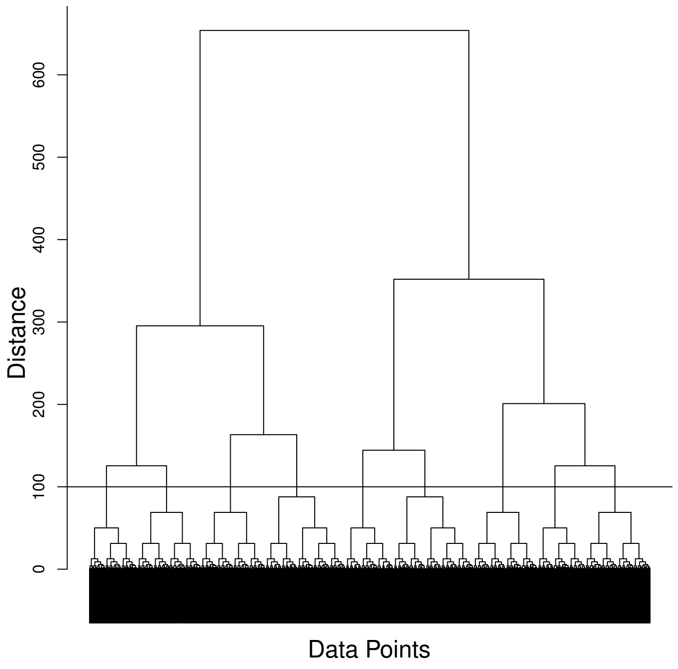


Fig. 1. Dendrogram of our dataset

To define the number of clusters k for k-means and k-medoids, we used two methods. The first is through hierarchical clustering, which creates a hierarchy that can be presented as a dendrogram. Figure 1 shows an example of a dendrogram. The dendrogram represents related data, and each successive relation creates the hierarchy. Therefore, it provides a visual tradeoff between the number of clusters and the size of each cluster. The larger the *distance* value for which a cut is made, the fewer the number of clusters and the larger the size of the clusters. For example, if we chose to cut at $distance = 100$, we will have $k = 9$ clusters, which are about equal size.

The second method to choose k is the elbow method, which is based on increasing the number of clusters to help reduce the sum of within-cluster distances of each data point to its cluster representative. We first select a small k and then slowly increment it until $\sqrt{\frac{N}{2}}$ [9], where N is the number of data points in the dataset. Figure 2 shows an example of the total within-cluster distance as a function of the number of clusters k for the k-means algorithm. The goal is to choose k at the "elbow," which is when increasing k does not significantly reduce the within-cluster distances. A reasonable "elbow" for our figure is at $k = 12$, which is indicated by an arrow.

Finally, *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN) is a density-based clustering algorithm that uses a similarity heuristic to find groups that contain a defined minimum number of data points δ within

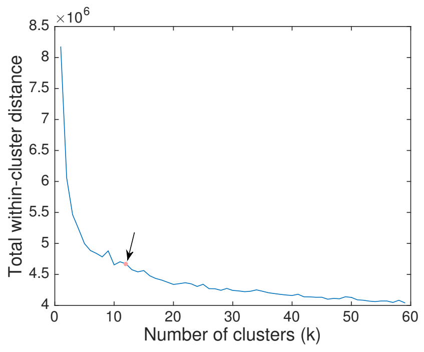


Fig. 2. Tradeoff between within-cluster distance and number of clusters k

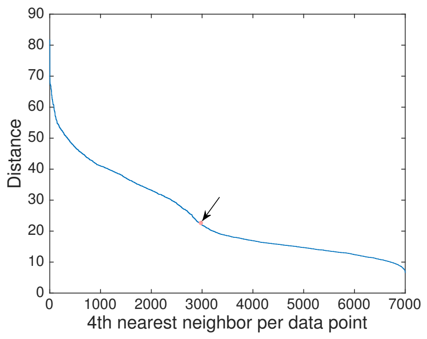


Fig. 3. Tradeoff between distance ϵ and number of data points whose 4th nearest neighbor is within ϵ

Algorithm 3 DBSCAN algorithm

```
1: procedure DBSCAN($X$, $\epsilon$, $\delta$)
2:     mark all data points in $X$ as unvisited
3:     repeat
4:         randomly select an unvisited object $x$;
5:         mark $x$ as visited;
6:         if the $\epsilon$-neighborhood of $x$ has at least $\delta$ points then
7:             create a new cluster $C$ and add $x$ to $C$;
8:             set $\Pi$ as the set of points in the $\epsilon$-neighborhood of $x$;
9:             for each point $x'$ in $\Pi$ do
10:                if $x'$ is unvisited then
11:                    mark $x'$ as visited
12:                    if $\epsilon$-neighborhood of $x'$ has at least $\delta$ points then
13:                        add those points to $\Pi$
14:                    end if
15:                end if
16:                if $x'$ is not yet a member of any cluster then
17:                    add $x'$ to $C$
18:                end if
19:            end for
20:            output $C$
21:        else
22:            mark $x$ as noise
23:        end if
24:    until no object is unvisited
25: end procedure
```

a defined ϵ-distance. The algorithm selects a data point at random and greedily adds data points that reside within ϵ of the start data point. Once the minimum number of data points is obtained, it will attempt to expand the cluster

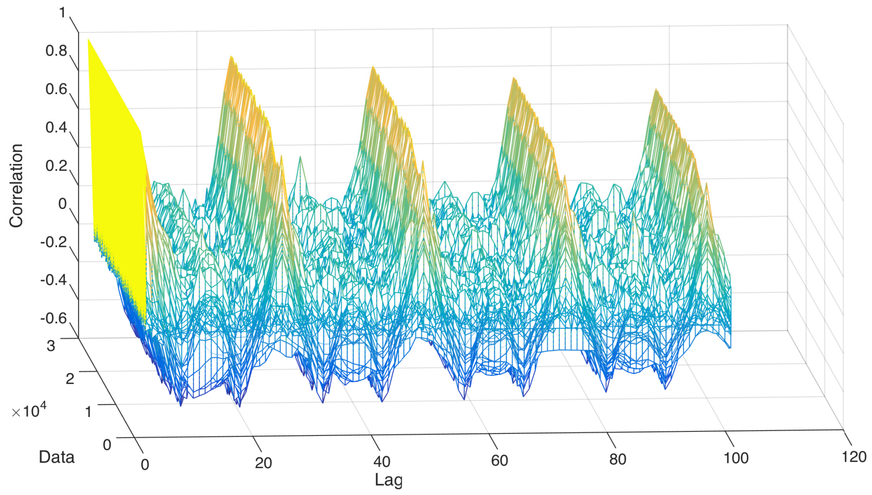


Fig. 4. Sample autocorrelation as a function of the lag

by continuously clustering more data points within ϵ from any data point in the cluster. Algorithm 3 shows the pseudocode of DBSCAN.

DBSCAN is different from k-means and k-medoids in that it requires a minimum number of data points δ to define a cluster and a maximum distance ϵ to associate dense neighbors. These parameters can be estimated using methods devised by Ester *et al.* [5]. For example, in Figure 3, $\delta = 4$ and $\epsilon = 22.49$.

3.4 Prediction

We now describe how we learn the parameters of prediction methods using the game training datasets $\mathbf{G}_{\text{train}}$. We used two off-the-shelf methods to predict the energy consumption of consumers. The first is a *moving average* with a two week or 336 hour window defined by:

$$x_t = \mathbb{E}[x_{t-336} + \cdots + x_{t-1}] \tag{3}$$

which takes two weeks of the known consumption in the past and averaging for an estimated next consumption value, x_t, at consumption hour t.

The second is a variant of the *classical autoregressive model* (AR) [6] defined as:

$$x_t = w_1 \cdot x_{t-h} + w_0 \tag{4}$$

where x_t is the predicted future energy consumption; w_1 and w_0 are weights and the model uses the value from a fixed "lag" in the past x_{t-h}; where h is the lag value. This variant equation is used because it performed well with the periodic consumption behavior of the household consumers. To determine the best lag value, we attempted to find consumption patterns that may exist in the time series data. Using the equation:

$$\hat{\gamma}(h) = \frac{1}{m} \sum_{i=1}^{m-h} (x_{i+h} - \bar{x})(x_i - \bar{x}) \tag{5}$$

we found the sample autocovariance of the time series data [6], where h is the lag and m is the number of time steps in the time series. Then, using the equation below:

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)} \tag{6}$$

we can compute the sample autocorrelation with the sample autocovariance of the original time series shifted by h hours $\hat{\gamma}(h)$ over the sample autocovariance of the original time series $\hat{\gamma}(0)$. Figure 4 plots an example sample autocorrelation and one can visualize that there is a peak correlation at 24-hour intervals.

Therefore, the variant autoregression lag model in equation (4) uses the lag value of 24 simulated hours in the past.

4 Experimental Results

Recall that we have controlled and uncontrolled experiments. We ran our experiments with the three clustering algorithms described in Section 3.3 and used the two prediction algorithms described in Section 3.4 to understand the advantages of clustering. To evaluate our algorithms on the test datasets, we associate the test bootstrap data for each consumer to the most similar cluster and used the prediction model of that cluster to predict the consumption in the test game data associated to that test bootstrap data.

We used $k = \{6, 10\}$ for the controlled experiments of base and controllable loads defined by the hierarchical clustering and elbow methods for the k-means and k-medoids clustering algorithms, and we set $\delta = 4$ and $\epsilon = \{11.77, 16.34\}$ for base and controllable loads, respectively, for the DBSCAN algorithm. Similarly, we used $k = 2$ for the uncontrolled experiments of mixed both load types, and we set $\delta = 4$ and $\epsilon = 0.75$ for the mixed load types. We also used the lower and upper limits of k being 1 cluster and N clusters, a cluster per training bootstrap data point. This provided us an idea of a possible lower and upper bound for predictive error using partitioning based algorithms k-means and k-medoids.

Note, in our experiments we found k-means++ and k-medoids to have very similar results during prediction. Hence, we only discuss k-means++ because it had a slightly better result than k-medoids. We used the *Mean Squared Error* (MSE) to analyze the prediction error of both prediction methods. The MSE was computed using:

$$\text{MSE} = \frac{1}{n \cdot m} \sum_{i=1}^{n} \sum_{j=1}^{m} (\hat{c}_{i,j} - c_{i,j})^2 \tag{7}$$

where $\hat{c}_{i,j}$ is predicted energy consumption of consumer i during hour j, $c_{i,j}$ is the actual energy consumption of consumer i during hour j, n is the number of

Table 1. Controlled Experiment Prediction MSE with k-means++

k	**Load Type**	**Moving Average**	**Autoregressive Lag Model**
1	Base	1.4148e-05 kWh	5.8561e-06 kWh
6	Base	1.4148e-05 kWh	5.8545e-06 kWh
10	Base	1.4148e-05 kWh	5.8425e-06 kWh
7000	Base	1.4148e-05 kWh	5.9506e-06 kWh
1	Controllable	1.8476e-05 kWh	1.8691e-05 kWh
6	Controllable	1.8476e-05 kWh	1.8229e-05 kWh
10	Controllable	1.8476e-05 kWh	1.8201e-05 kWh
7000	Controllable	1.8476e-05 kWh	1.7819e-05 kWh

Table 2. Controlled Experiment Prediction MSE with DBSCAN

ϵ	**Load Type**	**Moving Average**	**Autoregressive Lag Model**
11.77	Base	1.4148e-05 kWh	5.8536e-06 kWh
16.34	Controllable	1.8476e-05 kWh	1.8685e-05 kWh

Table 3. Uncontrolled Experiment Prediction MSE with k-means++

k	**Load Type**	**Moving Average**	**Autoregressive Lag Model**
1	Both	1.9750 kWh	0.5161 kWh
2	Both	1.9750 kWh	0.4999 kWh
1400	Both	1.9750 kWh	0.4928 kWh

Table 4. Uncontrolled Experiment Prediction MSE with DBSCAN

ϵ	**Load Type**	**Moving Average**	**Autoregressive Lag Model**
0.75	Both	1.9750 kWh	0.4925 kWh

consumers in the test dataset, and m is the number of time steps in the time series.

In the smaller controlled experiments, Tables 1 and 2 tabulate the prediction MSE for k-means and DBSCAN, respectively. For both clustering algorithms, the autoregressive lag model outperformed the moving average for base loads. The reason is that it has a more predictable consumption pattern. On the other hand, the autoregressive lag model and the moving average model performed similarly for controllable loads. The reason is that the consumption of controllable loads is more erratic. A more granular view of per-cluster MSE is presented in Figures 5 to 8.

In the larger uncontrolled experiments, Tables 3 and 4 tabulate the prediction MSE for k-means and DBSCAN, respectively. In both clustering algorithms, the autoregressive model also outperformed the moving average. The per-cluster view is presented in Figures 9 and 10, where the y-axis is in log scale.

5 Related Work

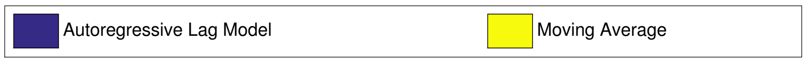
While there is a large number of Power TAC brokers that have competed in the past Power TAC competitions including AgentUDE15[2], Maxon15[3], Mertacor[4], COLDPower [10], TacTex14 [11], and CWIBroker14 [12], many of the approaches used by the brokers are not published publicly. As such, it is difficult to accurately identify the types of learning approaches taken by the agents. We describe below a sample of brokers that *do* publish their approaches and describe how we differ from them in our learning methods.

Parra Jr. and Kiekitveld [13] investigated the use of a large number of algorithms including linear regressions, decision trees, and k-nearest neighbors, all implemented on WEKA, to predict customer energy usage patterns in Power TAC. Their analysis used weather and energy consumption to perform analysis on different types of consumers in a Power TAC simulation. The main difference between their work and ours is that we used dimensionality reduction techniques as well as clustering prior to using prediction algorithms. Unfortunately, their results show that their techniques were not successful in finding a good model without a high error.

Urieli and Stone [11] also use learning algorithms in their TacTex14 broker, where they cluster consumers not by their energy usage but by their type. For example, office complex consumers are all clustered together independent of the number of occupants in the office complex, which is not known to the broker. They then use a locally weighted linear regression model to predict the energy consumption of those clustered consumers.

Finally, the approach taken by Wang *et al.* [14] is the most similar to ours, where they too cluster customers according to their energy usage using the k-means algorithm. However, their prediction methods are different, where they propose two methods. The first predicts the future consumption based on a weighted sum of the current consumption and the historical consumption and the second uses logistic regression based on historical usage data and weather data. As they also discussed the strategies of their broker for the other parts of the competition (i.e., the wholesale, imbalance, and tariff markets), they show empirical results on how their broker performed overall. As such, it is not known how effective their prediction algorithms are. In contrast, we show that our

[2] `http://www.powertac.org/wiki/index.php/AgentUDE15`
[3] `http://www.powertac.org/wiki/index.php/Maxon15`
[4] `http://www.powertac.org/wiki/index.php/Mertacor2015`



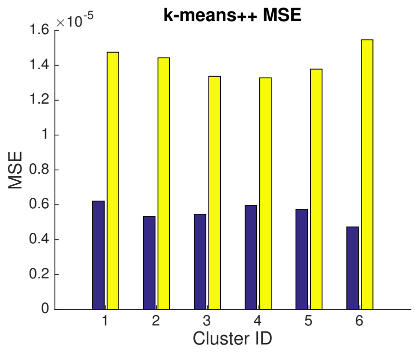


Fig. 5. Prediction MSE: Base Load in Controlled Experiment with k-means++ ($k = 6$)

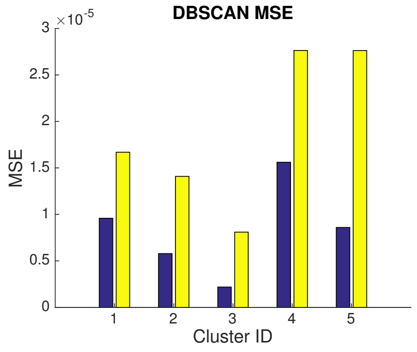


Fig. 6. Prediction MSE: Base Load in Controlled Experiment with DBSCAN

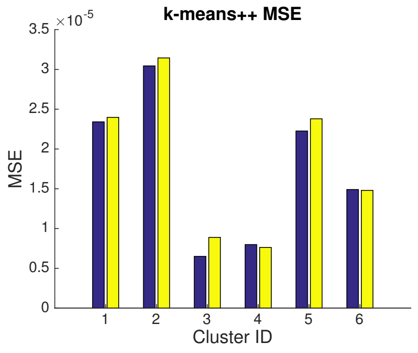


Fig. 7. Prediction MSE: Controllable Load in Controlled Experiment with k-means++ ($k = 6$)

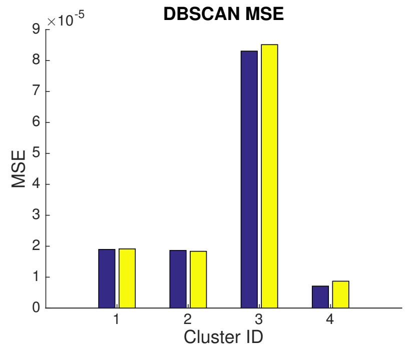


Fig. 8. Prediction MSE: Controllable Load in Controlled Experiment with DBSCAN

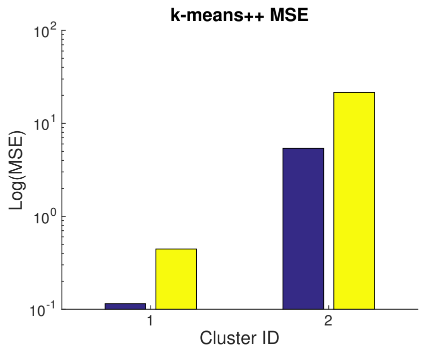


Fig. 9. Prediction MSE: Uncontrolled Experiment with k-means++ ($k = 2$)

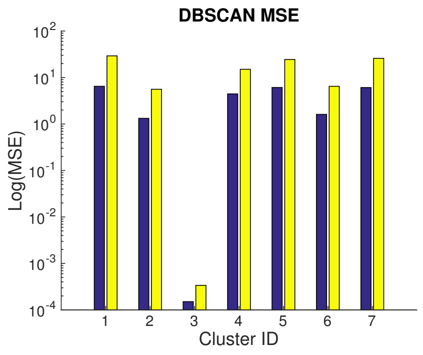


Fig. 10. Prediction MSE: Uncontrolled Experiment with DBSCAN

autoregressive lag model, which is significantly simpler and computationally efficient, has a small error and illustrate the underlying reason for this behavior, which is the high correlation in energy consumption in 24-hour intervals.

6 Conclusions and Future Work

In this preliminary study, we show that off-the-shelf clustering and prediction algorithms can be effectively used to classify consumers based on the predictability of their energy consumptions. We show that the k-means, k-medoids, and DBSCAN clustering algorithms coupled with an autoregressive lag model can predict energy consumption of consumers with reasonable accuracy. These results show that there is a strong temporal structure to the energy consumptions. Finally, we also plan to exploit the strong temporal correlations and integrate the clustering and prediction algorithms into an actual Power TAC broker agent for the competition.

Acknowledgments. This research is partially supported by NSF grants 1345232 and 1337884. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, or the U.S. government.

References

1. Ketter, W., Peters, M., Collins, J.: Autonomous agents in future energy markets: the 2012 power trading agent competition. In: Proceedings of the AAAI Conference on Artificial Intelligence. (2013) 1298–1304
2. Wellman, M.P., Greenwald, A., Stone, P., Wurman, P.R.: The 2001 trading agent competition. Electronic Markets **13**(1) (2003) 4–12
3. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. In: Proceedings of the ACM-SIAM Symposium on Discrete Algorithms. (2007) 1027–1035
4. Kaufman, L., Rousseeuw, P.: Clustering by means of medoids. North-Holland (1987)
5. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Conference on Knowledge Discovery and Data Mining. (1996) 226–231
6. Shumway, R., Stoffer, D.: Time Series Analysis and Its Applications: With R Examples. Springer Texts in Statistics. Springer (2010)
7. Jolliffe, I.: Principal Component Analysis. Springer Series in Statistics. Springer (2002)
8. Bishop, C.: Pattern Recognition and Machine Learning. Information science and statistics. Springer (2013)
9. Han, J., Pei, J., Kamber, M.: Data Mining: Concepts and Techniques. The Morgan Kaufmann Series in Data Management Systems. Elsevier Science (2011)

10. Serrano, J., de Cote, E.M., Rodríguez, A.Y.: Fixing energy tariff prices through reinforcement learning. In: Proceedings of the International Workshop on Agent based Complex Automated Negotiations. (2015)
11. Urieli, D., Stone, P.: Tactex'13: a champion adaptive power trading agent. In: Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems. (2014) 1447–1448
12. Liefers, B., Hoogland, J., La Poutr, H.: A successful broker agent for power tac. In Ceppi, S., David, E., Podobnik, V., Robu, V., Shehory, O., Stein, S., Vetsikas, I.A., eds.: Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets. Volume 187 of Lecture Notes in Business Information Processing. Springer International Publishing (2014) 99–113
13. Parra Jr, J., Kiekintveld, C.: Initial exploration of machine learning to predict customer demand in an energy market simulation. In: Proceedings of Workshop on Trading Agent Design and Analysis. (2013)
14. Wang, X., Zhang, M., Ren, F., Ito, T.: Gongbroker: A broker model for power trading in smart grid markets. In: International Conference on Web Intelligence and Intelligent Agent Technology. (2015) 21–24