Reinforcement Learning For Peak Shaving with Batteries

Logan Spear
Department of Electrical Engineering
Stanford University
lspear@stanford.edu

Jim Best-Devereux
Department of Electrical Engineering
Stanford University
jimbestd@stanford.edu

Abstract

As the world continues to encounter increasing electrification, an increasingly important topic for users of utility power to explore will be that of peak shaving. Currently, grid operators use a pricing scheme which charges certain energy consumers a large fee based off of how much energy they used in the hour of the month with the highest average demand [1]. This hour is determined retroactively, at the end of each month. To avoid these peak charges, some entities try to switch off of grid power (often to battery power) during probable peak hours, in a practice called peak shaving. This presents an interesting potential application for reinforcement learning, where an agent could be trained to control the battery pack so it is used most efficiently to address demand during peak hours. Guan et al.[2] has shown it is possible to incorporate residential-level photovoltaic energy generation and energy storage systems to reduce electric bills for the residential energy consumer using a TD(λ)-learning algorithm. We attempt to use a simpler approach, using value iteration to learn best actions to control our battery. While our results were less than ideal it highlights the promise of this work and the potential for an independent agent to implement peak shaving. Our code is available at https://github.com/logan-spear/aa228_project.

1 Problem Introduction

Electric power is becoming more and more essential to everyday life. As this trend continues utilities and grid operators have to attempt to balance the grid, especially during time of peak consumption moments when the grid is taxed the most. In order to discourage high peak demand, a special pricing structure is used. At the end of each month, the grid operator looks back at the total demand on the grid throughout that month and identifies the hour which had the highest demand. Then, each user is charged a hefty fee proportional to the amount of energy they were using during that peak. Thus, the goal of each individual is to minimize their energy use (i.e. switching to battery power) during any hour which they believe will be the eventual peak hour. This pricing structure is what motivates our work.

One potential challenge of this work is that the peak is labeled retroactively, so even if a moment is the highest peak so far into a day, it could still be less than a later peak. Additionally, an hour with demand that sets a new peak is often immediately followed by even more hours which set new peaks. If we are constrained to a finite supply battery then we have to be able to predict when the true peak is and not discharge solely when the demand begins to exceed the observed peak-so-far. Another consideration has to be that if the battery is discharged in an attempt to avoid the peak it will have to recharge at some point to be able to handle additional future peaks. Many commercially available systems, such as a Tesla Power Wall, only have the capacity to cover roughly two hours of demand. The optimization of the act of charging and discharging is what we study and attempt to improve upon.

Preprint. Under review.

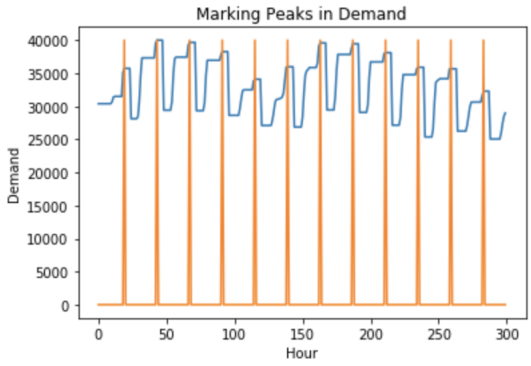

Figure 1: Labeling and Recognizing the Peaks for Each Day.

2 Methods

2.1 Related Works

Using the tools available to us thanks to reinforcement learning we attempt to improve peak shaving with the use of a dispatchable battery that can cover our demand. This is all in efforts of avoiding the peak and its associated cost. Vázquez-Canteli and Nagy endeavored to explore this area of work and came up with a summary of the models and methods that have been used in the past[3]. They found that the most common methods used were Q-Learning and Batch Reinforcement Learning (BRL). For both of these methods their action selection is motivated by either an ϵ-greedy approach or by using a softmax method. An example of the existing work would be Henze and Dodier's work on PV-Battery optimization[4]. In their work Henze and Dodier employed Q-learning and a softmax action selection to optimize how a PV-Solar system is used in coordination with a battery system to minimize the cost to a user in the grid. They trained on 30 years of data and found promise that their optimal control learns the benefits of saving available capacity to meet later critical loads rather than myopically discharging to meet current non-critical loads. Hoping to create a simple yet effective method, we modeled the problem as a partially observable Markov decision process (POMDP).

2.2 Implementation

At a high level, we cast the problem as a POMDP by training a Ridge regression model to forecast the next three intervals of demand, and then treated that forecast as our observation. Then, we further simplified the problem by making the certainty-equivalent assumption, in which we treat our observation as our state and learn a policy as a function of our observation, without a belief vector [6]. Below, we describe each step of the process. This description will closely mirror the order of cells in the Jupyter notebook in our github, where the code has been combined and documented

2.2.1 Regression model

The state we're interested in involves the future demand on the grid, which we cannot observe directly. Thus, we must estimate it using a forecasting model, and treat that forecast as our observation. To create a forecaster, we trained a simple Ridge regression model (linear regression with L2 norm regularization on the regressors). The regularization strength was determined using 5-fold cross validation, where the dataset was our entire dataset. For implementation, we used the `RidgeCV` model from `sklearn` [5].

2.2.2 Action, State, and Observation spaces

Action space: the action space is simply the three actions we can take with the battery: charge, discharge, or do nothing. For this problem instance, we are assuming we either charge or discharge our battery at maximum power for the entire hour. In practice, this problem can be further extended to include charge or discharge rates not at full power, and actions chosen more frequently (say, every

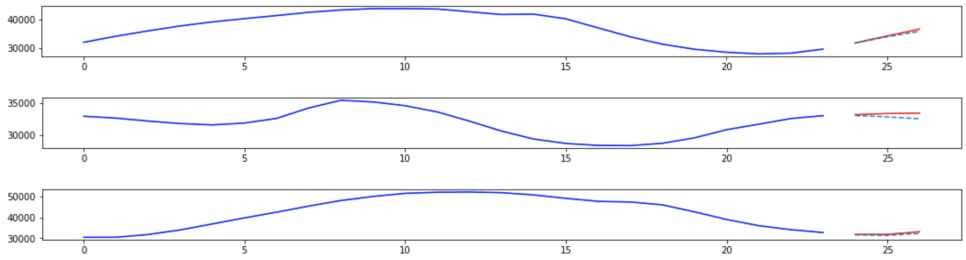

Figure 2: The forecasting used in our model to create an expectation for our electric load.

15 minutes instead of 1 hour). In our case, we limited ourselves to the simpler case of full power actions that last for the entire hour.

State space: the state space consists of three things. The first is a vector of the future demand. In our case, we chose to consider a time horizon of three hours, so the vector is a 3-vector, containing demand for the next three hours. The second thing in the state vector is the binary indicator of whether each future hour is the eventual peak hour. Again, this is a 3-vector, corresponding to the next three hours. Lastly, the third element in the state vector is the current battery charge. We chose to model our battery as a 2-hour battery (can fully drain or charge battery in two hours), and so this battery charge is simply a number in {0,1,2} indicating how many hours of energy is left in the battery. Interestingly, the battery charge is something that we *can* observe directly, and so we don't need to account for it in our observation space.

Observation space: because we cannot directly observe future demand, or whether the future hour is the eventual peak hour, we also have an observation space. As mentioned before, we use our regression model to forecast demand for the next 3 hours. In addition, the only information regarding the peak we have is the highest peak *so far* that we have seen. Thus, an additional element in the observation vector keeps track of the "peak so far," and that value is compared against our forecasts to ultimately help the model decide whether the next hour may be the true peak or not.

Because the model only really needs to care about the value of the forecasted demand relative to the peak-so-far value, we compute that difference and use that as our observation vector, rather than the directly forecasted values. However, because working with continuous values for the observation space was extremely difficult, we simplified the model by only considering the sign of the difference between our forecast and the peak-so-far. Thus, our observation is a binary vector where a 1 corresponds to an hour in which our forecast exceeds the known peak-so-far, and 0 otherwise. Thus, since there are two possible values for the three elements in the observation vector, our observation space was reduced to 8 possible observations (note that this does not include the state of the battery). Taking into consideration battery charge (3 possible values), in total we have $3 \times 8 = 24$ possible observed states we could be in.

2.2.3 Reward function

Technically, the reward for a certain execution of an action can only be calculated once the eventual peak has been determined, which doesn't happen until the end of the month (or end of the day, in our case, since we changed the problem). However, because we know the true peaks within the data, we could simply check whether the model discharged on a true peak or not, and reward it if it did and don't reward it otherwise. However, that would result in extremely sparse rewards, which may make training to convergence more difficult, and would also not incentivize other behaviors we would like the model to exhibit. Thus, we created our own, heuristic reward function which rewards the model whenever it properly discharges on any hour in which the peak gets increased, even if that hour is not the eventual peak for that day.

Furthermore, we also need to set rewards related to charging, otherwise the model may not learn to charge at all. As such, we reward the model for "safely" charging (charging during an hour in which a new peak is not set), and heavily penalize the model for charging during an hour in which a new peak is set (since this added demand would actually *increase* the demand charges). Details of the reward

function are available and documented in the code. We refer to our heuristic reward as "myopic" because, in some sense, it encourages shortsighted, greedy behavior. This is because it rewards the model equally for discharging on an earlier peak and a later peak, when in reality only the later peak matters (since only the last, greatest peak will be considered in practice).

2.2.4 Transition model

The transition function returns the probability of transitioning from one state to another, given a certain action. In our problem, this can effectively be broken up into two parts. The first part is the transition from a binary 3-vector representing whether a new peak is set in each future interval or not, to the next 3-vector representing the same thing. This does not depend on our action, and it is stochastic (must be estimated). The second part is the transition of the battery's charge from one period to the next. This does depend on our action, and is in fact a deterministic function of our action. Thus, if you examine the code, you will see we separate these two estimations into helper functions, then combine them for a single transition function. We estimate the stochastic aspect of the transition using maximum likelihood on the training data.

2.2.5 Observation model

Our observation function simply returns the probability of observing a particular observation vector given the true state vector. We estimate this using maximum likelihood over the training data by, for each interval in the training data, generating our forecast and associated observation, and then counting the number of times each particular observation occurred in the same interval as each particular state.

2.2.6 Value Iteration

With our state, action, and observations spaces and reward function defined and our transition and observation models estimated, we implemented value iteration to estimate the optimal policy under our assumptions. As a heuristic to check for convergence, we looked at the change in the norm of our estimated reward matrix. At the end, the change in norm per iteration was less than 10^{-7}, which we considered to be converged.

3 Results

Unfortunately, with our approach, the learned policy only hit 8% of peaks on our test data. However, it properly never charged when the battery was full or discharged when the battery was empty.

In our analysis, we found that a majority of peaks occur in intervals in which the forecasting model does not predict any new peaks in the next hours. In some sense, this represents how noisy our observations of the true state are, limiting the model's ability to hit true peaks. Other than when the forecaster predicts no peaks, the significant remainder of peaks occur when we predict a new peak only 1 hour from now, 1 and 2 hours from now, or in all three future hours. Interestingly, when the model predicts a new peak 1 and 3 hours from now (but not 2 hours from now), there is almost never a true peak in those instances. This is summarized in the graph in figure 3. In order to leverage this, we tried boosting the reward associated with discharging under observations 4, 6, and 7. Unfortunately, that didn't affect our performance.

Considering the poor performance, we expect that the discretization of the observation space over-simplified the model. A more expressive discretization would be to perhaps bin the amounts that the forecast exceeds the peak-so-far by, rather than making the observation binary. This allows the observation space to convey a rough measure of confidence in predicting a new peak, where the confidence is proportional to amount the forecast exceeds the peak by. This could perhaps enable the policy to learn roughly how noisy the forecasting model is, and not discharge without sufficient confidence in the probability of a peak in the next hour.

Other improvements to the model could likely be found by separating the data by month or by season. Energy demand patterns are strongly dependent on weather, especially temperature, which varies from month to month. Thus, using a separate forecaster per month, each one trained on data from that specific month, could possibly improve performance. This separation of data may also improve the

estimates of the transition and observation models, since their distributions may be related to time of year as well.

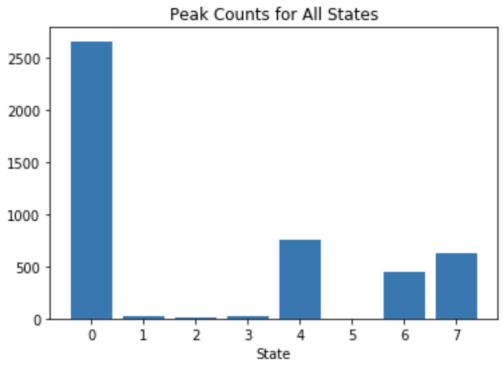

Figure 3: Showing the Number of Peaks Occuring in the Different States.

4 Team member contributions

Both team members contributed significantly to the project. In the early stages, both members researched possible applications and datasets, and wrote the proposal together. Similarly, both members worked together to cast the problem as a POMDP and brainstorm what values belonged in which vectors (state, action, observation, etc) and attended office hours together to consult TAs as well. Both also did research on various algorithms that may apply well to this particular application, and wrote the update together.

In coding up the project, Logan ended up being able to accomplish a significant amount of the actual coding in one sitting. However, most of the discussion on framing the problem had been done at that point, and it was simply putting in code what had been done on a whiteboard.

Regarding the report, both members contributed significantly, but Jim handled a larger portion - drafting, editing, adding discussion of sources, etc.

References

[1] M. H. Albadi, E. F. El-Saadany, *Demand Response in Electricity Markets: An Overview*. 2007 IEEE Power Engineering Society General Meeting, June 2007.

[2] C. Guan, Y. Wang, X. Lin, S. Nazarian, and M. Pedram, *Reinforcement learning-based control of residential energy storage systems for electric bill minimization*. 2015 12th Annual IEEE Consumer Communications and Networking Conference, Jan. 2015, pp. 637–642.

[3] J.R. Vázquez-Canteli, Z. Nagy *Reinforcement learning for demand response: a review of algorithms and modeling techniques*. Appl. Energy, 2019, pp. 1072-1089

[4] Gregor P. Henze , Robert H. Dodier *Adaptive Optimal Control of a Grid-Independent Photovoltaic System*. J. Sol. Energy Eng. Feb 2003, pp 34-42

[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research, 2011, vol 12 pp 2825-2830

[6] Lall, S. 2014 *EE365: Stochastic Control, Stanford University*. Model Predictive Control lecture slides.

[7] https://www.kaggle.com/robikscube/hourly-energy-consumption