HW 4: Forecasting Residential Electricity Power Consumption

Due: Friday April 12 at 11:59pm PT

You have been hired as a Building Energy Data Scientist. Congratulations! What an exciting new opportunity! In your first job assignment, you must construct algorithms to forecast residential electricity consumption with data and machine learning (ML). The assignment is organized in a tutorial fashion, thereby allowing you to practice ML on a relevant real-world energy system example.

Reading
[Optional] Read "Gated Ensemble Learning Method for Demand-Side Electricity Load Forecasting" first authored by Eric Burger, available at this URL.

Background
Several years ago we collected Green Button Data <http://www.greenbuttondata.org/> from the CE 295 students. This data includes hourly electricity consumption from over 20 residences in the East Bay. The data has been (mostly) cleansed for your convenience including (i) aligning time-stamps, (ii) filling-in missing data, and (iii) organizing it into an easily readable format.

Problem 1: Exploratory Data Analysis
Download the data file `HW4_Train_Data.csv`. Load the CSV data into Matlab or Python

(a) Create a bar plot of the average hourly energy consumption [kWh] for each building. Make the x-axis the building index number. Make the y-axis the average hourly energy consumption. In red, super impose error bars on your bars that indicate the standard deviation of hourly energy consumption. Provide this plot in your report.

(b) Which building has abnormally high variance? Do any buildings have moments of NEGATIVE power consumption? If so, then we will remove this building from our analysis. This building likely installed solar during the year and the smart meter data aggregates power consumption and solar power generation, so it's un-usable for this homework.

(c) Re-organize your energy data set into a 4-D array. In order, the dimensions correspond to (i) building index, (ii) week-of-year, (iii) day-of-week, and (iv) hour-of-day. For each element in the 4-D vector, store the *normalized* energy. That is, divide kWh by the maximum hourly energy consumption for that building.

Normalize the energy for each building. That is, divide kWh by the maximum hourly energy consumption for that building. In industry and academia, we sometimes refer to these normalized building electricity consumption trajectories as "load shapes". Now, generate the following plots:

- In seven separate figures (one for each day-of-week), plot the hourly energy consumption load shapes vs. hour (x-axis) for each building - all super-imposed.

- In each of the seven figures, plot the *average* hourly energy consumption in a thick black line.

How to do this? In Matlab, you can re-organize your energy data set into a 4-D array. In order, the dimensions can correspond to (i) building index, (ii) week-of-year, (iii) day-of-week, and (iv) hour-of-day. For each element in the 4-D vector, store the *normalized* energy. In Python, you can create a DataFrame with the Pandas library, where the first three columns are week-of-year, day-of-week, and hour-of-day. The next columns would be normalized energy for all the buildings. Then you can use the `groupby` function in Pandas to help you generate the plots.

Provide the seven plots in your report. They should look similar to Fig. 1 below. Comment on any observations you might have.

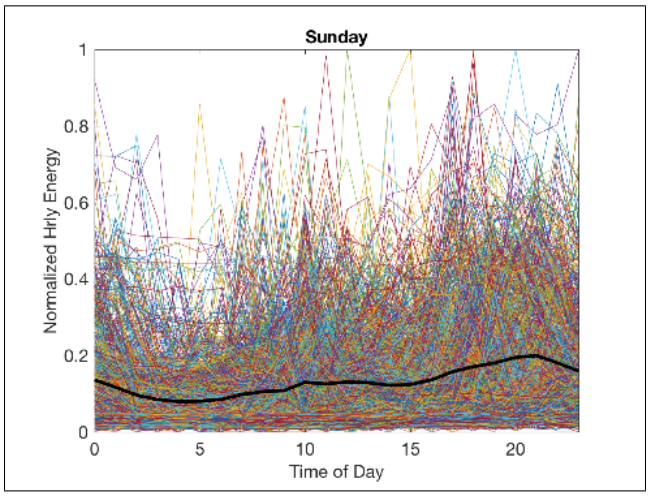


Figure 1: Example of plot you must produce for Problem 1(c).

Problem 2: Average Model

In this problem, we design an extremely simple forecasting model that is often effective. We call it the "Average Model". The average model forecasts the building power to be the average value from historical data at the HoD and DoW. Mathematically:

$$\hat{P}_{avg}(k) = \sum_{i=1}^{7} \sum_{j=1}^{24} \bar{P}_{ij} \cdot U_{ij}(k) \tag{1}$$

where

- $\bar{P} \in \mathbb{R}^{7 \times 24}$ is the average value from historical data, for each (DoW, HoD) pair.
- $U(k) \in \{0,1\}^{7 \times 24}$. All elements are zero, except the one element which corresponds to (DoW,HoD) at time step k. For example,

- if k corresponds to 12am on Sunday, then $U_{1,1}(k) = 1$.
- if k corresponds to 3am on Monday, then $U_{2,4}(k) = 1$.
- if k corresponds to 11pm on Saturday, then $U_{7,23}(k) = 1$.

Answer the following questions.

(a) Download test data file `HW4_Test_Data.csv`. Load the CSV data into Matlab or Python. The test data includes one week of normalized hourly load for Sunday Sept 7, 2014 00:00 – Saturday Sept 13, 2014 23:00. Generate seven plots (one for each DoW) which visualize HoD (x-axis) and the normalized hourly energy for the Test Data and Average Model.

(b) Compute the mean absolute error (MAE)

$$MAE = \frac{1}{m}\sum_{i=1}^{m} |P(k) - \hat{P}(k)| \tag{2}$$

for each DoW and the entire week. Provide these error metrics in your report. Which DoW has the largest and smallest MAE?

Problem 3: Autoregressive with eXogeneous Inputs Model (ARX)
In this problem, we will design a forecasting model based on the Autoregressive with eXogenous inputs model (ARX). We consider the ARX model given by:

$$\hat{P}_{arx}(k) = \sum_{\ell=1}^{L} \alpha_\ell \cdot P(k-\ell) + \hat{P}_{avg}(k) \tag{3}$$

where $\hat{P}(k)$ is the normalized hourly energy consumption we aim to forecast in the target hour. The inputs include the previous hourly energy consumption: $P(k-1), P(k-2), \cdots, P(k-L)$. We will also consider an exogenous term corresponding to the average power: $\hat{P}_{avg}(k)$. Consequently, the autoregressive terms adjust the average model based upon temporally local power, over a short retrospective horizon. In total, we must fit L parameters. Answer the following questions.

(a) Write the ARX model in linear-in-the-parameters form: $Y = \Phi\theta$ as described in the video lectures. Define the vector Y and matrix Φ.

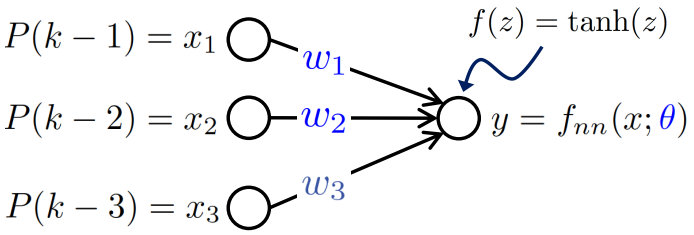
(b) Formulate a least squares optimization problem to find the optimal parameters $\alpha_\ell,\ \ell = 1, 2, \cdots L$ which fit your data. Write down the objective function. Define your notation. Is this a convex program? Why?

(c) Solve your optimization problem with $L = 3$. In your report, give the optimal values of $\alpha_1^\star, \alpha_2^\star, \alpha_3^\star$.

(d) Test your ARX model on the test data set. Generate seven plots (one for each DoW) which visualize HoD (x-axis) and the normalized hourly energy for the Test Data, the Average Model, and the ARX model. Report the MAE for each DoW, and the entire week.

Problem 4: Neural Network Model

Next, we consider a neural network (NN) model, and more specifically a single neuron model. As detailed below, the NN model is just a nonlinear function with many parameters (a.k.a. "weights" in the jargon of ML). Our NN model is simply: $\hat{P}_{nn}(k) = f_{nn}(x;\theta) + \hat{P}_{avg}(k)$ where the NN makes adjustments to the power forecast relative to the mean.



$$z = \sum_{\ell=1}^{3} w_\ell x_\ell = w^T x \qquad (4)$$

$$y = f(z) = \tanh(z) \qquad (5)$$

(a) Consider the least squares error:

$$J = \frac{1}{2}\sum_{i=1}^{m} \delta_i^2 = \frac{1}{2}\sum_{i=1}^{m} \left[y^{(i)} - y(w^T x^{(i)})\right]^2 = \frac{1}{2}\sum_{i=1}^{m} \left[y^{(i)} - \tanh(w^T x^{(i)})\right]^2 \qquad (6)$$

Our objective is to fit neural network weights w to the example training data,

$$x^{(i)} = [P(k-1), P(k-2), P(k-3)]^T, \qquad y^{(i)} = P(k) - \bar{P}^T U(k) \qquad (7)$$

where training example index $(i) = 1, 2, \cdots, m$ corresponds to time index $k = 4, 5, \cdots, K$. We do this by solving a nonlinear least squares optimization problem:

$$\text{minimize} \qquad J = \frac{1}{2}\sum_{i=1}^{m} \left[y^{(i)} - \tanh(w^T x^{(i)})\right]^2 \qquad (8)$$

which is a nonlinear program because the output y is nonlinear w.r.t. weights w. What are the optimization variables?

(b) A simple fitting/optimization algorithm is called "gradient descent":

$$w^{k+1} = w^k - \gamma \cdot \frac{\partial J}{\partial w}\left(w^k\right) \qquad (9)$$

where superscript $(\cdot)^k$ represents the iterations. Armed with the gradient descent method, our next goal is to compute the gradient in (9). Use chain rule to expand $\partial J/\partial w$, as done in the lecture videos. After expanding with chain rule, then express each term analytically.

(c) Implement the gradient descent algorithm yourself. Select an initial guess of $w^0 = 0 \in \mathbb{R}^3$. Use a small step size/learning rate, such as $\gamma = 10^{-3}$ or 10^{-4}. Perform at least three steps of gradient descent. In

your report, give the final values of $w_1^\star, w_2^\star, w_3^\star$.

(d) Test your NN model on the test data set. Generate seven plots (one for each DoW) which visualize HoD (x-axis) and the normalized hourly energy for the Test Data, the Average Model, the ARX model, and the NN model. Report the MAE for each DoW, and the entire week.

Interesting Remarks

- Stochastic gradient descent (SDG) is a popular alternative to the gradient descent algorithm in (9) in ML. In gradient descent, we update weights by computing the gradient with ALL training examples. This can be a computationally heavy task. In SDG, we update a weights based on a RANDOM SUBSET (e.g. 10%) of training examples within each iteration. This is computationally lighter, since we significantly reduce the number of forward propagations and backward propagations through the neural network.

Deliverables

Submit the following on bCourses. Be sure that the function files are named exactly as specified (including spelling and case). Please do NOT zip files.

`LASTNAME_FIRSTNAME_HW4.PDF`
`LASTNAME_FIRSTNAME_HW4.xyz` which contains your respective Matlab or Python files, i.e. `xyz` $\in$ {`m, ipynb`}.