Gaussian Process Regression with K-means Clustering for Very Short-Term Load Forecasting of Individual Buildings at Stanford [1]

Carol Hsin

***Abstract*—The objective of this project is to return expected electricity load in kiloWatts (kW) given a building and a time interval. The raw data consists the mean kW of 15 minute intervals for two years from 122 meters corresponding to 109 buildings at Stanford resulting in a data set of 70176 observations. The data was cleaned and then split into training and test sets. K-means clustering was used to split the training set into more similar sets. Test dates and test buildings was clustered clusters and clusters were used to make the prediction on test dates from given clusters. The baseline model is an auto regressive model. GPR models with 7 days prior and GPR models based on clustering was also completed. Error analysis was done using Root Mean Squared Error.**

I. Introduction

Unlike other products, electricity isn't easily stored, so producers must anticipate and meet the maximum demand, also known as peak load, at any given time or outages will ensue. Since consuming fuel and running electrical power generators isn't free, producers also need to reduce operational costs by keeping the generation reserve at the minimum required to meet demand. While it would be suboptimal to be producing at peak if most of the electricity generated will be wasted, it would also be problematic for outages to occur due to inadequate supply. Thus, accurate electrical forecasts are essential to energy management, production, transmission, and distribution because these forecasts allow dispatchers to make decisions on the optimal, real-time scheduling of electricity production between power plants or generation units.

A. Gaussian Process Regression (GPR)

A Gaussian process is a collection of random variables of which the joint distribution of any subset of the variables are Gaussian. It can be viewed as a multivariate Gaussian in which we have infinitely many variables, which basically corresponds to a function because we can think of a function as being an infinitely long vector that returns a value based on the input x. Thus, a Gaussian process can be used to describe a distribution over functions.[1]

For clarity, let us explicitly state the differences between a Gaussian distribution and a Gaussian process. A Gaussian *distribution* is fully specified by a mean vector, μ, and a covariance matrix Σ.

$$f = (f_1, \ldots, f_2)^\top \sim \mathcal{N}(\mu, \Sigma) \text{ for indexes } i = 1, \ldots, n$$

In contrast, a Gaussian *process* is fully specified by a mean function $m(x)$ and covariance function $k(x, x')$ with x as the indexes. We can think of the mean function as an infinitely long vector and the covariance function as an infinite-by-infinite dimensional matrix. Gaussian processes with finite index sets would just be Gaussian *distributions*.

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')) \text{ for indexes } x$$

Gaussian distributions are nice to work with because of their special properties, particularly the marginalization property because that allows us to get finite dimensional answers to finite dimensional questions from the infinite dimensional process.[1] Recall that the marginal of a joint Gaussian is Gaussian and that formulating the marginal distribution is just taking the corresponding sub-blocks from the mean and covariance matrix of the joint distribution:

$$\begin{bmatrix} x_A \\ x_B \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \mu_A \\ \mu_B \end{bmatrix}, \begin{bmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{bmatrix})$$

$$p(x_A) = \int_{x_B \in R^n} p(x_A, x_B; \mu, \Sigma)\, \mathrm{d}x$$

$$x_A \sim \mathcal{N}(\mu_A, \Sigma_{AA})$$

This property means that to make a prediction with on a test point x_*, so one can just take the relevant sub-block of the covariance matrix of the $(n+1)$-dimensional joint distribution (n training points and 1 test point).

Another useful key property is that the covariance matrix corresponding to a multivariate Gaussian distribution is positive semidefinite, which is also the necessary and sufficient condition for a matrix K to be a Mercer kernel. Thus, any valid kernel function can be used as a covariance function and we can apply the 'kernel trick' to reduce computation time. Generally, the kernel is the most important aspect of a Gaussian process and choosing the right kernel to capture the correlations in the data is where most of the work will be. Specifically, if we assume zero mean, which is a commonly used assumption, then the Gaussian process is completely defined by the covariance function.

In this project, we are only interested in the case of predicting using noisy observations $y = f(x) + \varepsilon$ where $f(x)$ is unknown and noise is modeled as $\varepsilon \sim \mathcal{N}(0, \sigma_n^2)$. Thus the covariance of y is $cov(y) = K(X, X) + \sigma_n^2 I$

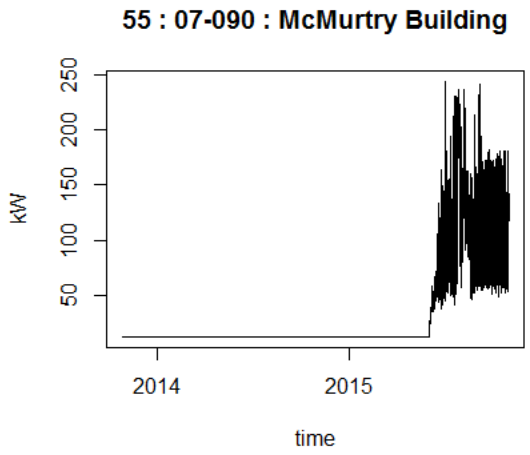


Fig. 1: Plot of a rejected building

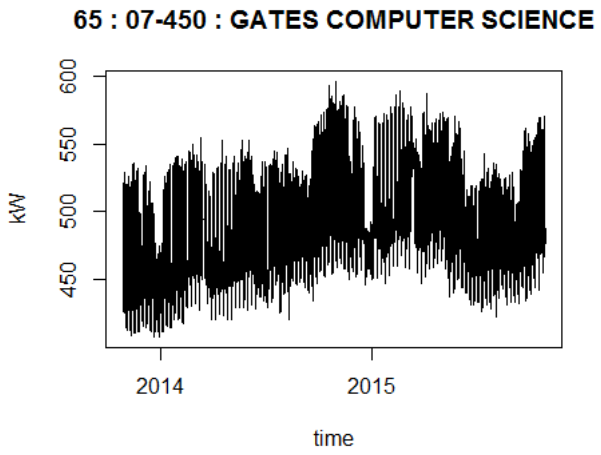


Fig. 2: Comparison to non-rejected data

and after defining $K = K(X, X)$ to simplify notation, the key predictive equations for GPR is:[1]

$$f_*|X, y, X_* \sim \mathcal{N}(\bar{f}_*, cov(f_*))$$

$$\bar{f}_* = E[f_*|X, y, X_*] = K(X_*, X)[K + \sigma_n^2 I]^{-1} y$$

$$cov(f_*) = K(X_*, X_*) - K(X_*, X)[K + \sigma_n^2 I]^{-1} K(X, X_*)$$

II. Data Description

The raw data obtained from Stanford Facilities consists of the mean kW of 15 minute intervals for two years from 122 meters corresponding to 109 buildings at Stanford resulting in a data set of 70176 observations. Each building may be serviced by multiple meters and each meter may be servicing multiple buildings. The data ranges from October 28, 2013 at 12:00:00 AM to October 28, 2015 at 10:30:00 PM.

A. Data Exploration and Cleaning

Preliminary data exploration revealed there are 8 missing data points within the period being examined. This was discovered while formatting the time. After looking at the data file, missing points occurred on March 9, 2014 and March 8, 2015 between 01:45:00 PST and 03:00:00 PDT, so for about one hour starting at 2AM. This corresponds to daylight savings time, so the data for that period is missing because the hours don't actually exist and this was confirmed because Nov 3, 2013 and Nov 2, 2014 has extra data points. The missing points around March were extrapolated using the mean of the points before and after while the extra points were merged by averaging every consecutive two points.

In the data cleaning process, meters servicing the same buildings were merged and any meters serviced more than one building was removed. This process reduced the data from 122 to 93 variables.

All 93 remaining were plotted and analyzed to remove anomalies, especially buildings with missing data that was replaced with a number as a stand-in. This reduced the data to 71 variables for a 70080x71 data matrix (365 days/year * 24 hours/day * 4 15-mins/60-mins * 2 years = 70080).

B. Dividing the Data

The data was divided such that only the last half would be part of the test set. The trick here was that since the plan was to see how well similar buildings could help predict the load of one building, there needed to be a way to group the buildings. K-means would do this, but personal computers do not have enough memory required for k-means on a 35088x71 (70176/2 = 35088). To solve this, the dates were clustered (more in K-Mean section).

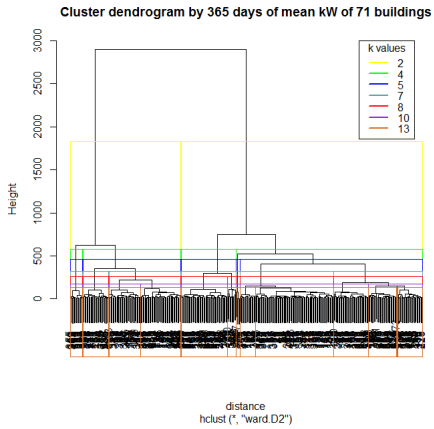
For each experiment, new models were trained per test date per test building and only data from past wrt the test date/time was used to obtain predictions. Basically, for one building chosen from each of the building cluster and one day from each of the date clusters, we got a total of 4 test buildings and 8 test dates, so 32 models per experiment. There are 3 experiments (see Experiments section).

III. K-means Clustering

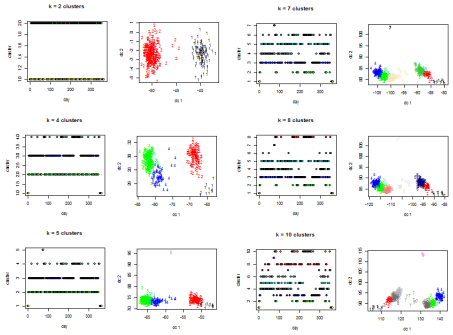
K-means clustering was performed to get similar dates with which to cluster the buildings. R's kmeans and plotting functions were used in this section.

A. *Clustering Dates*

To cluster dates, the 35040x71 matrix was reduced to a 35040x1 mean vector (mean across the 71 buildings). Then the vector was reshaped so that each row was a date (1:365) and the columns were the time intervals (1:96 for 24 hours * 4 15-min/60-min).



The resulting matrix was fed into R's kmeans function to cluster the rows using the euclidean distance (each point seen as a 96-dimensional vector). The resulting hierarchical cluster dendrogram revealed 7 possible K-values based on euclidean distances calculated. While each point is in 96 dimensional space, a visualization using usual discriminant coordinates was created for better analysis. For some K-values, it was clear which days were being clustered, e.g. winter vacation in K=4 was cluster 4. A k value of 8 was picked because the distances were reasonable and the goal was to get as many clusters as the data would allow.



B. *Clustering Buildings*

The dates clusters were used to cluster the buildings. The clusters were used to turn the 35040x71 data matrix into a 71x768 matrix by splitting dates of the building vector (35040x1) into the cluster groups and then taking the mean of each group and then reshapping that into a vector to be inserted into the k-means clustering building matrix.

$$\begin{bmatrix} 1 \\ \vdots \\ 35040 \end{bmatrix} \rightarrow [365\text{x}96] \rightarrow [8\text{x}96] \rightarrow \begin{bmatrix} 1 \\ \vdots \\ 768 \end{bmatrix} \rightarrow [71\text{x}768]$$

The resulting 71x768 matrix was clustered by rows and the following cluster dendrogram was used to determine the appropriate K-value. See figure on page 4 for building cluster dendrogram. A k-value of 4 was chosen by the usual distance metric because it was the highest number of clusters that was reasonable given the data.

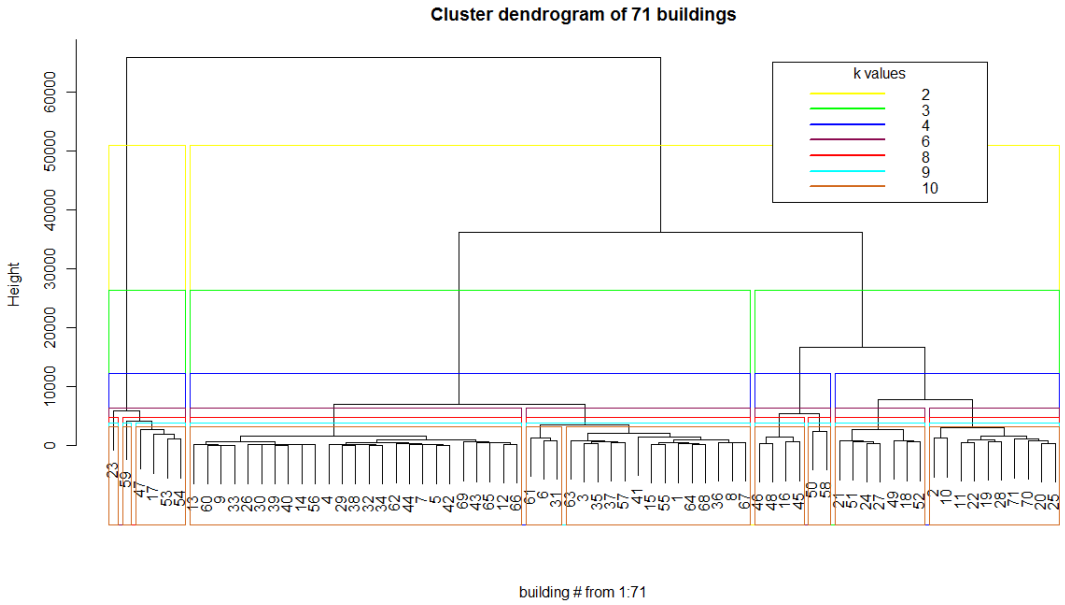
IV. Experiments and Models

A. *Baseline Model: Autoregressive Linear Model*

For the 32 baseline models, the data point is predicted based only on data from the past. An AR model was completed for each building and for each test date based on 30 days prior to the test date. R's auto.arima was used to process each data vector. An input vector of 30*96=2880 points prior to a test date was used to predict the output vector of 96 for the next 96 time points in the test date. This was done using a for loop, so only one time point (of the 96) was predicted per run and the prediction saved into a vector, then the next test point would be predicted with the same 30 days but also with the previous time point from the data.

AR Building 27		
date	trainRMSE	testRMSE
001	3.11562406136734	5.92908086302396
243	2.80726350414573	27.1727170432979
107	3.57288754308325	25.0050035248015
337	3.60722058939789	19.6730977413011
341	3.64815270136514	6.34749946969175
298	3.17871587441246	13.7552846364742
076	3.00282373966765	16.4055959407633
259	2.86685920192124	22.5292421013259

AR Building 41		
date	trainRMSE	testRMSE
001	11.3137171108784	1.63099169210129
243	1.74704263141441	16.5290857191283
107	15.8048459682757	68.6245047572578
337	16.1212470934128	71.9370965560381
341	16.0362318459376	53.1374392548315
298	16.6870934592130	40.6767609910458
076	16.1701762503520	37.6412048858712
259	5.48500264308082	50.8484470522624



AR Building 50		
date	trainRMSE	testRMSE
001	3.38363367869690	11.8094774861282
243	3.73769993365999	17.0612639134373
107	4.57362220540115	37.5793096579895
337	3.94042377491766	48.1890094398847
341	3.88495198175363	10.8304380342995
298	4.21250373061003	14.1253591604769
076	3.97807505678671	17.8563097039936
259	3.56124275581864	36.8535230182934

AR Building 53		
date	trainRMSE	testRMSE
001	15.9652386786304	25.3668404088917
243	10.8554679708035	54.8878136779890
107	16.6909585241200	61.2952412778726
337	13.8077246997190	59.4570004960659
341	15.3236243694047	51.9888134588392
298	12.7511734577707	20.4150748457536
076	11.3992393682145	36.7204665273363
259	11.7719471437517	107.842217763288

B. GPR with 7 days prior

The 32 models used an input vector of size 7*96 = 672 to predict the output vector of size 96. Like in AR, a for loop was used and a vector was used to keep track of each time point predicted. For this and all models using GPR, the covariance matrix used was Squared Error and the hyperparameters were tuned in part by using the GPML Matlab toolkit. [2]

Building27errs.csv		
date	trainRMSE	testRMSE
001	1.8478	3.2777
243	2.0239	4.1934
107	2.5364	4.1437
337	3.0882	9.7364
341	2.9625	4.8741
298	1.8401	3.1328
076	1.2381	6.0168
259	2.1532	3.5589

Building41errs.csv		
date	trainRMSE	testRMSE
001	2.1173	2.1642
243	2.1173	2.8415
107	15.4112	25.6011
337	10.4088	25.5864
341	15.2884	18.6595
298	14.2461	20.6533
076	7.9567	19.6042
259	7.4125	22.0174

Building50errs.csv		
date	trainRMSE	testRMSE
001	1.9757	4.0314
243	2.7139	6.2222
107	2.7835	5.4576
337	2.9771	6.8797
341	3.1093	4.7846
298	3.6101	4.5342
076	1.6541	5.5567
259	3.0225	7.3988

Building53errs.csv		
date	trainRMSE	testRMSE
001	13.0682	26.4726
243	6.0205	14.1705
107	6.4943	12.7095
337	11.4501	26.5387
341	15.8296	16.2554
298	6.4616	10.8375
076	3.5744	10.2883
259	6.4255	12.1282

C. GPR with similar building

The 32 models used an input matrix of size (7*96 = 672)x(number of buildings in cluster) to predict the output vector of size 96. Each of the test buildings were from a building cluster, so their respective building clusters were used as the input. Thus, a vector of the time period (672) from each building in the test building's cluster that is not the test building was used and the input matrices' sizes differed depending on the number of buildings in the cluster.

Building27errs.csv		
date	trainRMSE	testRMSE
001	2.2500	2.8123
243	0.6901	7.9039
107	0.8641	6.9301
337	1.0063	17.3021
341	0.7230	6.7764
298	2.1044	4.9508
076	0.1623	6.5125
259	0.0086	6.5019

Building41errs.csv		
date	trainRMSE	testRMSE
001	0.2706	1.5223
243	0.0758	3.2870
107	0.0972	17.0242
337	0.0653	24.6378
341	0.0016	16.3858
298	0.0005	19.8603
076	0.0041	20.5427
259	0.0018	19.8467

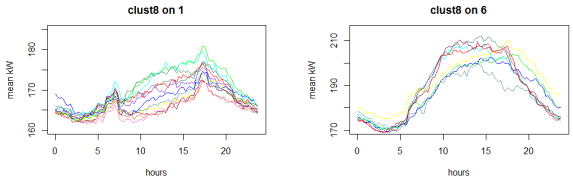
Building41errs.csv		
date	trainRMSE	testRMSE
001	2.3962	4.9584
243	3.8441	7.2553
107	5.2072	9.3673
337	3.9320	10.9341
341	4.6400	6.8721
298	3.6479	7.3121
076	0.8671	9.5621
259	3.8320	11.0443

Building50errs.csv		
date	trainRMSE	testRMSE
001	16.2173	17.9709
243	7.3066	19.9443
107	9.6251	14.6105
337	13.0713	21.4529
341	9.3861	19.3867
298	9.2154	9.7162
076	0.0250	16.5415
259	8.7231	13.4898

V. Analysis

The models using the clustering did not perform as well as expected compared to the models using just the historical data of that particular building being analyzed. This may be that a building's profile is quite unique and surround buildings may not be good features for predictions. It might also be the case that the clustering could be better.

Further plots and analysis of the clustered dates were completed. The data to suggests that the days being clustered do follow a logic in that each has a pattern. The plots of cluster k=8 with cluster 1 is displayed. The mean Kw corresponds to the mean vector of the matrix from the first half of the data, as described in the k-means section.



Errors might also be due to having only 7 days. The author wanted to use more than 7 days, but GPR is a non-parameteric model, which means it is very memory intensive and Matlab gave errors that 'the array exceeds maximum array size preference'. More data points also slows the for loop from 1:96 that retrains a model per run to be slow and given 32 models to per experiment.

VI. Conclusion

While the results from the experiments seem to suggest that similar days are not quite that important, it might be due to the clustering of the buildings, so more work needs to be done to understand why similar buildings do not have a high predictive value.

References

[1] Carl Edward Rasmussen and Chris Williams. *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts: The MIT Press, 2006.

[2] Carl Edward Rasmussen and Chris Williams. *The GPML Toolbox version 3.5*. manual for GPML Matlab Code version 3.6. July 2015.