GEFCom2012 Hierarchical Load Forecasting Gradient boosting machines and Gaussian processes

James Robert Lloyd

Department of Engineering, University of Cambridge

Abstract

This report discusses methods for forecasting hourly loads of a US utility as part of the load forecasting track of the Global Energy Forecasting Competition 2012 hosted on Kaggle. The methods described (gradient boosting machines and Gaussian processes) are generic machine learning / regression algorithms and few domain specific adjustments were made. Despite this, the algorithms were able to produce highly competitive predictions and hopefully they can inspire more refined techniques to compete with state-of-the-art load forecasting methodologies.

Keywords: Load forecasting, Gradient boosting machines, Gaussian processes

1. Introduction

This report details the methods I used when competing in the load forecasting track of GEFCom2012[1]. It is split into sections describing each technique used for forecasting temperatures and loads. Within each section I have motivated the particular choice of algorithm, discussed how it was used and described how to replicate the results using the spreadsheets and scripts accompanying this report[2].

I approached the competition in the spirit of a data mining competition. Consequently, some of the choices detailed below were based on intuition to save time and focus on the aspects of the data most likely to give the largest increases in performance. Many of these choices could have been replaced by appropriate searches and cross validation; where more complex techniques would be required I have briefly described how one could perform a more objective analysis.

My methodology for load back / forecasting was to try different general purpose regression algorithms and then ensemble (average) the predictions. The final ensemble comprised predictions from a gradient boosting machine (GBM), Gaussian process (GP) regression and the benchmark solution provided by the competition organisers (a linear model).

This paper is organised as follows. Section 2 discusses data cleansing, section 3 is on temperature forecasting, sections 4 and 5 introduce the GBM and GP forecasting methodologies respectively and section 6 discusses how the final ensemble prediction was formed. A brief introduction to Gaussian processes is provided in Appendix A.

2. Data cleansing

A sensible first step in any prediction task is to look at your data; in particular searching for anomalies. This was performed for both temperature and load data using the spreadsheets `temp/temp.xlsx` and `load/load.xlsx` by plotting the data as time series and using various types of conditional formatting to search for irregularities visually.

Email address: `jrl44@cam.ac.uk` (James Robert Lloyd)

[1] `http://www.gefcom.org/`

[2] Source code available at `https://github.com/jamesrobertlloyd/GEFCOM2012/`

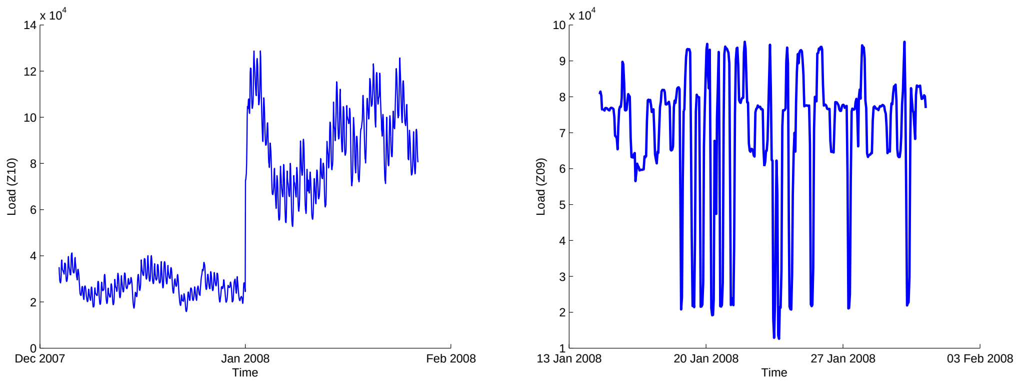

Figure 1: Left: Raw data at zone 10, showing a large discontinuity. Right: Raw data at zone 9, showing atypical load dynamics.

This visual analysis revealed a large discontinuity in load series 10 and atypical dynamics in series 9 (see figure 1 and others for comparison). No other large anomalies were detected during the first inspection of the data and smaller irregularities were not revisited since the algorithms detailed below performed acceptably well without further data cleansing.

No adjustments were made for holidays or other irregular events such as black-outs. Ignoring them was merely a practicality of time constraints rather than a design choice.

3. Temperature forecasting

3.1. Initial analysis and remarks

The error metric used in GEFCom2012 was heavily weighted towards times at which temperatures were unknown (i.e. the load forecast rather than backcasts). Consequently, good temperature predictions seemed crucial for overall success.

When submitting a solution to Kaggle, the error metric was computed on 25% of the held out data and returned to the user. This allowed a user to optimise temperature predictions by optimising the score of resulting load predicitions (i.e. computing load predictions based on different temperature predictions and selecting the temperature prediction with the highest corresponding load prediction score). Depending on how the test data was split (into the 25% test and 75% validation partition) this may have allowed users to come very close to knowing the true future temperatures.

I therefore used a flexible but simple method for forecasting temperatures that could be easily tuned. Figure 2 shows data from temperature station 1 along with various curves used for prediction (described later). The black solid line is the raw data, which shows that temperatures follow a smooth trend with a daily pattern of rising and falling temperatures. For simplicity, I modelled the smooth trend and daily periodicity separately and modelled each temperature station in isolation.

3.2. Methodology

The smooth trend was estimated within the data using local linear regression (e.g. chapter 6 of Hastie et al., 2009) with a bandwidth of one day. I assumed that the smooth trend would probably smoothly return to the historical average. A suitable modelling technique that would give this type of prediction is Gaussian process regression (e.g. Rasmussen and Williams, 2006) (see the appendix for a very brief introduction); the difference between the smoothed temperature and its historical average was regressed against time using a squared exponential kernel and zero mean function.

In Figure 2, the thick blue dashed curve shows the historical average of smoothed temperatures and the thick green dotted line shows the current smoothed temperature and prediction. The parameters of the GP model (length scale

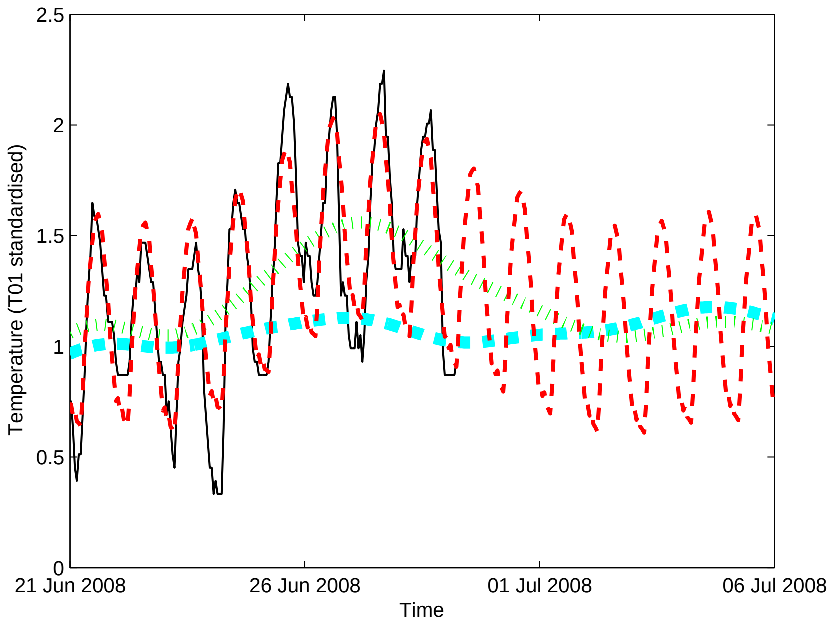

Figure 2: Raw data at temperature station 1 (after removing mean and scaling to have unit standard deviation) (black solid). Historic average of smoothed temperatures (blue dashed thick). Current and predicted smoothed temperatures (green dotted thick) and final temperature predictions (red dashed).

and scale factor of the squared exponential kernel) were tuned by hand, initially choosing sensible values based on plots of the data and then trying to optimise the public test score of derived load predictions.

The difference between the temperature and the smoothed temperature was assumed to be a smooth periodic function with a period of one day. This modelling assumption was implemented using a Gaussian process with a periodic kernel (again regressing the difference against time). Parameters of the GP model were chosen by optimising the marginal likelihood of the model given particular parameters using optimisation methods supplied in the GPML toolbox[3]. Only the last 250 data points (which corresponds approximately to 10 days) were used to fit this part of the model in order to capture recent temperature dynamics.

3.3. Replication of results

The spreadsheet and scripts used to manipulate the temperature data can be found in the folder `temp`. The MATLAB script makes use of the GPML toolbox which is contained in the folder `GPML`. The temperature data was reshaped into a (time) $\times$ (temperature station) array using the spreadsheet `temp.xlsx` and then saved in `times_series_raw.csv`. The method described above is implemented by the MATLAB script `predict_temp.m` which saves the predicted time series in `GP_pred_temp.csv` and the smoothing of these predictions in `smooth_temp_GP.mat` and `smooth_temp_GP.csv`.

3.4. Discussion

The red curve in Figure 2 shows the result of the methodology applied to temperature station 1. The fit to the data is far from perfect but it has captured the broad features of the data. I am unfamiliar with the temperature / weather forecasting literature but I am quite certain that far better methods exist!

[3]http://www.gaussianprocess.org/gpml/code/matlab/doc/

4. Gradient boosting machines

4.1. Initial analysis and remarks

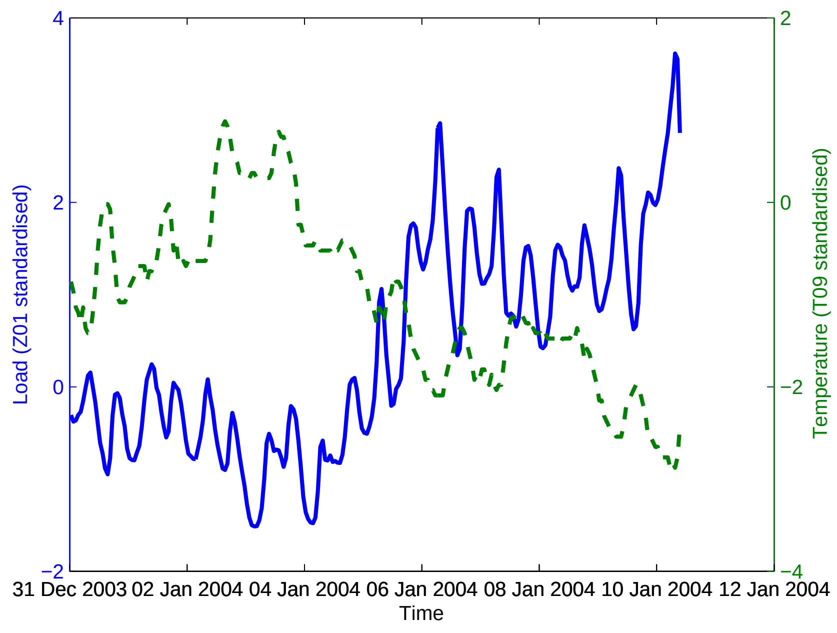

Figure 3: Raw data at load zone 1 (after removing mean and scaling to have unit standard deviation) (blue solid) and raw data at temperature station 9 (after removing mean and scaling to have unit standard deviation) (green dashed).

Figure 3 plots a subset of the data for loads at zone 1 and temperature station 9. We can see that that the load has a smoothly varying trend with roughly periodic daily deviations from this trend. The load also appears to be negatively correlated with temperature in this region of data. It is also reasonable to assume that load dynamics will be different on weekends compared to weekdays.

Based on these observations, I modelled loads as a function of time of day, time within week, temperatures and smoothed temperatures (all stations). All zones were modelled independently for simplicity.

4.2. Methodology

A standard 'black-box' technique for learning an unknown regression function is that of using gradient boosting machines (GBM) (e.g. chapter 10 of Hastie et al., 2009). For each zone I used all of the data to train a GBM and then used that to predict all outputs i.e. I learnt a global function for each zone.

I used a standard R implementation of GBM using most of the default settings[4]. After some brief experimentation with parameter values I used a shrinkage factor of 0.01, 10000 trees and an interaction depth of 3. I experimented with other parameter values, using the public test score as a guide, but nothing I tried improved upon my initial guess and therefore I did not expend much effort trying to optimise parameter values.

4.3. Replication of results

The scripts and data used can be found in the folder `gbm`. The load data was reshaped into a (time) $\times$ (load zone) array using the spreadsheet `load.xlsx` in the folder `load`. This was then concatenated with the temperature predictions and smoothed temperature predictions and saved in `gbm_input.csv`. The R script `basic_gbm.R` then performs the method described above to produce predictions which are saved in `gbm_output.csv`.

[4]http://cran.r-project.org/web/packages/gbm/index.html

4.4. *Discussion*

The time series for zone 10 had a very large discontinuity (see left of Figure 1). GBM was used as a global prediction method which means it will have produced very bad predictions for zone 10. I later fixed this by modelling the loads either side of the discontinuity separately. Perversely, this resulted in a worse public test score. However, this change resulted in a much improved private test score; in hindsight I should have opted for the more reasonable model!

5. Gaussian processes

5.1. *Initial analysis and remarks*

The observations made about the load data in section 4.1 (i.e. smooth trends, near periodic variations, dependence on temperature) can be modelled using Gaussian process regression. Specifying the kernel function of a Gaussian process allows one to encode these structural assumptions into a regression model. See the appendix for a brief introduction to the effect of choosing different kernels.

5.2. *Methodology*

Three different kernel functions were used; one for backcasting, one for forecasting and one for backcasting zone 9. The form of the three kernels used were as follows[5]

$$\mathrm{SE}_t + \mathrm{SE}_S + \mathrm{SE}_t \times \mathrm{SE}_T \times \textsc{Per}_t \tag{1}$$

$$\mathrm{SE}_t + \mathrm{SE}_t + \textsc{Per}_t \tag{2}$$

$$\mathrm{SE}_t + \mathrm{SE}_S + \mathrm{SE}_T \times \textsc{Per}_t \tag{3}$$

where SE refers to a squared exponential kernel, Per is a periodic kernel and the subscripts indicate which dimension the kernel acts upon; (t)ime, (T)emperature or (S)moothed temperature. For backcasting, the model was applied to the 1000 surrounding data points; for forecasting the last 500 data points were used. Only one temperature station was used in this method; for each back / forecasting region the marginal likelihood of the model was computed for each temperature station. Bayesian model averaging (e.g. Hoeting et al., 1999) was then used to combine the predictions (in practice, this was usually numerically equivalent to selecting the model / temperature station with the highest marginal likelihood).

Kernel (1) encodes the assumptions that loads can be explained as a smoothly varying trend and an approximately periodic component. The smooth trend is the sum of a smooth function of smoothed temperature and a smooth function of time (to explain any deviations from the trend implied by temperature). The periodic component can vary through time and by being multiplied by SE_t and SE_T the shape of the periodicity will be more similar to recent times and points in time with similar temperatures. This kernel was used for backcasting all zones apart from zone 9. An example of this kernel in action is shown in Figure 4. The left plot shows the load data (black solid) and the prediction (red dashed). On the right is the temperature time series which resulted in the model with the highest marginal likelihood. Comparing the two plots we can see that the deviations of the prediction from the trend through time correspond to fluctuations in the temperature time series.

Kernel (2) encodes the assumption that loads can be explained as the sum of two smooth functions with different length scales and an exactly periodic smooth function. This simpler kernel was used for forecasting since it gave more stable extrapolations of the data.

Kernel (3) is a simpler version of kernel (1) used for backcasting zone 9. The periodic component is now only similar to other times with similar temperatures. This prevented the predictions from being so greatly affected by the large isolated load fluctuations present in zone 9.

The forms and parameters of the kernel functions (see the accompanying code for all of the values) were chosen by trial and error (observing which kernels gave reasonable looking predictions and computing public test scores).

[5]See source code for parameter values

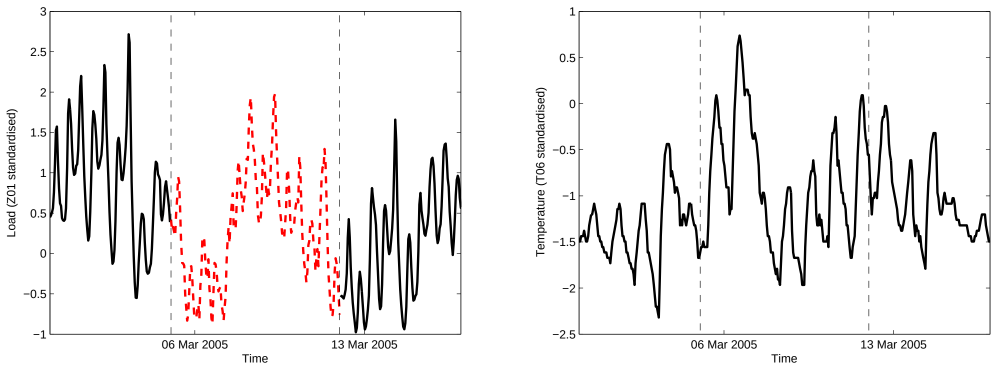

Figure 4: Left: Raw data (after scaling and translation) (black solid) and Gaussian process predictions (red dashed) for zone 1. Right: Corresponding temperature time series used for prediction. Notice that the peak in temperature corresponds to a trough in the load predictions i.e. the model has predicted a negative correlation between temperatures and loads.

5.3. Replication of results

The spreadsheet and scripts used can be found in the folder gp. The load data was reshaped into a (time) × (load zone) array using the spreadsheet `load.xlsx` in the folder `load` and then saved in `load_input.csv` which encodes missing values as zeros. The outputs of the temperature prediction `GP_pred_temp.csv` and `smooth_temp_GP.mat` are used directly.

The MATLAB script `gp_elec.m` performs the methodology described above, saving the predictions to `gp_pred.csv`.

5.4. Discussion

Selecting the form and parameters of the kernels by hand requires a moderate amount of familiarity with Gaussian processes. Automatic selection of kernel functions is an active area of research; recent publications include an automated search over a generative grammar of kernel expressions by Duvenaud et al. (2013), using deep neural networks to learn a kernel function by Salakhutdinov and Hinton (2008) and a new class of kernels designed to have good extrapolation performance by Wilson and Adams (2013). It is likely that better performance could have been achieved by a larger and more principled search over kernel functions.

6. Selecting the final ensemble

6.1. Remarks

The final prediction was formed as the ensemble (weighted average) of predictions from three separate methods; gradient boosting machines, Gaussian process regression and the benchmark solution provided by the competition organisers (a linear model). Ensembling is a very powerful technique for combining predictions from separate models. If two predictions (viewed as random quantities) are statistically uncorrelated then their average will have lower variance than either one separately.

In practice, different algorithms are correlated and ensembling highly correlated predictions will rarely lead to improved performance. In particular, I also tried the random forest algorithm but the predictions were too correlated with gradient boosting machines (they are both based on an ensemble of decision trees) to be a useful addition to the ensemble.

6.2. Methodology

The ensemble weights were chosen by hand, using the public test score as the metric to be optimised. The public test scores did not vary considerably once sensible parameters had been found so more advanced techniques were not used to optimise the ensemble weights.

6.3. *Replication of results*

The spreadsheets used to manipulate model output can be found in the folder `ensemble`. The spreadsheet `sub_creator.xlsx` was used to convert (time) $\times$ (load zone) arrays of model output into the required format for submission. The output of this spreadsheet was then copied in `ensemble_creator.xlsx` which performs the averaging to produce ensembles of the predictions.

6.4. *Discussion*

This method could have been been slightly improved by programmatically selecting the ensemble weights, optimising some form of validation metric. One could also have tried to optimise the public test score directly. Since only 2 submissions could be tested per day, one would want to use an optimisation technique that made very efficient use of data (e.g. Osborne et al., 2009; Snoek et al., 2012).

Larger improvements could likely be achieved by considering a larger number of base predictions. It is likely that the algorithms used by other competitors in GEFCom2012 would provide useful additions to the ensemble of predictions.

7. Conclusion

In this competition I applied general purpose machine learning / regression algorithms and made few adjustments for the specific domain the data arose from. Despite this, the resulting predictions were highly competitive. It is hoped that this report shows that simple techniques can be very effective for predictive modelling and can inspire practical techniques for load forecasting.

Acknowledgments

I would like to thank Alex Davies for guidance into the ways of data mining and David Duvenaud for advice on the practicalities of using Gaussian processes.

Appendix A. A very brief introduction to Gaussian processes

For a much richer introduction, see Rasmussen and Williams (2006). A Gaussian process, $(f(x))_{x \in X}$ is a set of random variables such that any finite collection has a jointly Gaussian distribution. A Gaussian process can be completely specified by a mean function $m(x)$ and kernel function $\kappa(x, x')$ such that

$$\text{Expectation}(f(x)) = m(x) \quad \text{(A.1)}$$

$$\text{Covariance}(f(x), f(x')) = \kappa(x, x'). \quad \text{(A.2)}$$

Gaussian processes can be used to specify a prior distribution on functions and then be used in a Bayesian analysis. They are nonparametric and can learn any function given sufficient data.

The kernel function encodes the structure of a Gaussian process. The left plot in Figure A.5 shows samples (random draws) from a Gaussian process distribution using squared exponential kernels; this function encodes an assumption of smoothness. Intuitively, these plots show the types of functions that will most easily / quickly be learned using Gaussian process regression. One parameter of a squared exponential kernel is a length scale which controls how quickly the value of the Gaussian process varies. The blue line in the left plot has a short length scale which results in quick variations of the function. The green line has a long length scale and consequently the Gaussian process is visually much smoother.

The right plot in Figure A.5 shows samples from a Gaussian process with a more complicated kernel function. The kernel function is the product of a squared exponential kernel and a periodic kernel. This encodes the assumption that the Gaussian process is roughly periodic, and that the shape of the periodicity changes smoothly.

By constructing more advanced kernel functions one can specify more complex structures as desired. Kernel functions depend on parameters and they are typically chosen by maximising the marginal likelihood of training data given the parameter choices (all gradients can be computed analytically). One can also perform a fully Bayesian analysis by integrating over parameter values; this typically requires approximate inference techniques.

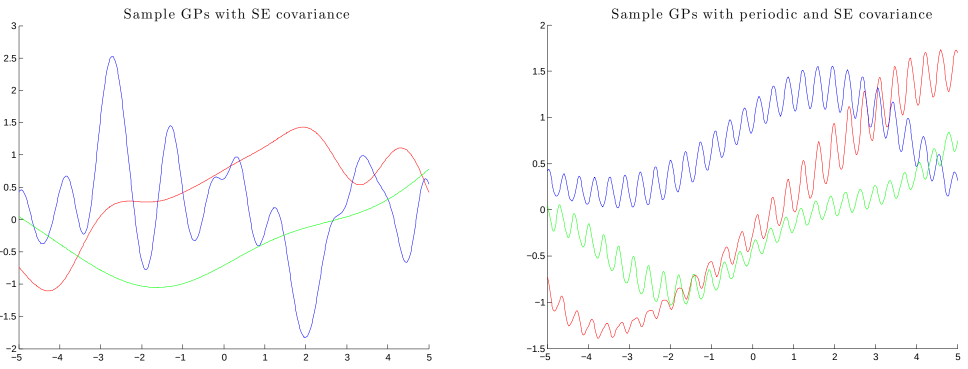

Figure A.5: Left: Draws form a Gaussian process with squared exponential kernel with differing length scales. Right: Draws using a squared exponential and periodic product kernel.

References

Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J. B., Ghahramani, Z., 2013. Structure discovery in nonparametric regression through compositional kernel search. In: Proceedings of the 30th International Conference on Machine Learning.

Hastie, T., Tibshirani, R., Friedman, J. H., 2009. The Elements of Statistical Learning. Springer.

Hoeting, J. A., Madigan, D., Raftery, A. E., Volinsky, C. T., 1999. Bayesian Model Averaging: A Tutorial. Statistical Science 14 (4), 382–401.

Osborne, M. A., Garnett, R., Roberts, S. J., 2009. Gaussian processes for global optimization. In: 3rd International Conference on Learning and Intelligent Optimization (LION3).

Rasmussen, C. E., Williams, C. K. I., 2006. Gaussian Processes for Machine Learning. The MIT Press.

Salakhutdinov, R., Hinton, G., 2008. Using deep belief nets to learn covariance kernels for Gaussian processes. Advances in Neural information processing systems 20, 1249–1256.

Snoek, J., Larochelle, H., Adams, R. P., 2012. Practical bayesian optimization of machine learning algorithms. In: Advances in Neural Information Processing Systems.

Wilson, A. G., Adams, R. P., 2013. Gaussian process covariance kernels for pattern discovery and extrapolation. In: Proceedings of the 30th International Conference on Machine Learning.