Load forecasting exercise

Richard Li

**Data Integrity**

We first check that the data contains contiguous 15 min intervals, as any gaps would complicate the lagged load inputs. My initial check led me to realize I wasn't properly accounting for daylight savings time/time zones. After correcting for this, I found no time gaps in the dataset.

Unfortunately, load data was lost at the end of DST in Nov 2012 and 2013, presumably because the data at the "old" 1:15am was overwritten by data at the "new" 1:15am. I initially tried interpolating these missing load values, but ended up removing them altogether as this produced a better training result.

In [**276**]: df2 = df[(df['datetime'] >= datetime(2012,11,2)) & (df['datetime'] < datetime(2013,12,2))]

In [**277**]: df2[df2['actual\_kwh'].isnull()]['datetime']

Out[**277**]:

92 2012-11-02 00:00:00

289 2012-11-04 01:15:00

290 2012-11-04 01:30:00

291 2012-11-04 01:45:00

292 2012-11-04 01:00:00

35233 2013-11-03 01:15:00

35234 2013-11-03 01:30:00

35235 2013-11-03 01:45:00

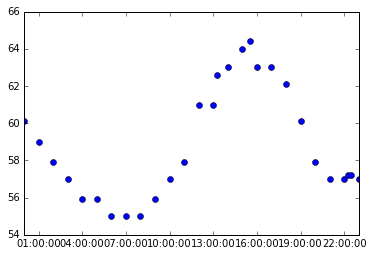
35236 2013-11-03 01:00:00

Name: datetime, dtype: datetime64[ns]

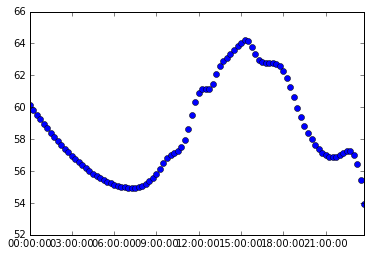
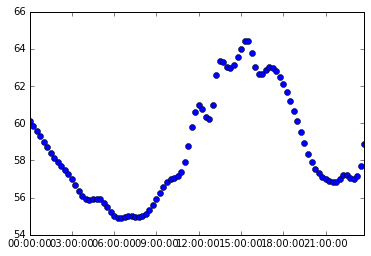
**Interpolating Temperatures:**

I chose to use spline interpolation to fill in the missing temperature values, as they can accommodate the micro-fluctuations of temperature from cloud cover, pressure changes, etc. I suspect the interpolation is most accurate in monotonic regions, and less accurate near local maxima/minima and inflection points.

The spline parameters were chosen to avoid overfitting, where each data point becomes a knot, while maintaining the integrity of the actual hourly temperature readings. [why iterate over smaller samples instead of fit a spline for the entire data sample? explain boundary effects?] For cubic splines on 200-step samples, I experimentally chose this smoothing factor to be s=0.2.



Actual load: 7/5/2013



Spline k=3, s=0: overfitted, too many knots Spline k=3, s= 0.2: good balance

For certain days with significant fluctuations in temperature, such a small smoothing factor led to artifacs, creating erratic jumps in temperature or raising a RuntimeWarning. For these segments, the smoothing factor was incrementally increased until the spline stabilized. This adjustment was required for 18 segments (7% of the data), requiring smoothing factors up to 2.6.

**Selection of Model:**

I chose to use a neural network (MLP) as it is a nonlinear regression model, which can accommodate the nonlinear interactions of load demand. Given more time, I would like to compare performance to a bagged regression tree model, which may offer advantages in efficiency.

I considered using a time-series NN, but decided that the time component can be adequately represented by the predictor variables below.

**Predictors:** 8 predictor variables were implemented:

Short-term forecasting only:

– temperature

– 24 hr lagged load

– 7 day lagged load

– previous day average load

Short- and long-term forecasting:

– time of day

– day of week

– day of year

– work day or weekend/holiday

Humidity measures (dew point) are often included in load forecasting, but assuming this dataset is from California, humidity should not be a strong predictor for load.

**Training the Model**

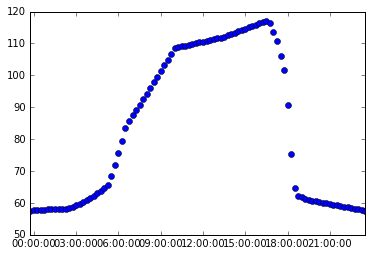
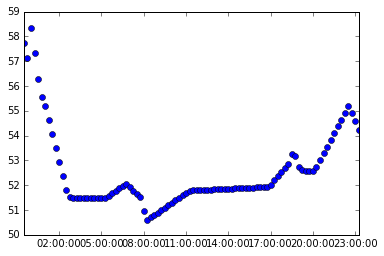
Initially I used sklearn's randomized train/test split, which yielded promising R^2 values, but that sort of shuffling is probably not appropriate for a time series such as this. Our data cannot be considered independent and fairly distributed, since the load at any given time is influenced by that immediately preceeding it. I ultimately chose to retain consecutive data points and use a 25% test split, roughly corresponding to a training set of Nov 2012–Aug 2013, and a test set from Aug–Dec 2013.

Evaluating out-of-sample performance of this trained model using the test set resulted in R^2 values of 0.75 for short-term forecasting (8 predictors), and 0.72 for long-term forecasting (4 predictors).

**Forecasting 24 hr period**

The model is equipped to perform long-term forecasting for a 24-hour period, given a string input of the starting time, i.e. "6-30-2013 14:00." Currently only 4 predictor variables have been implemented for forecasting (time, day of week/year, and workday). Assuming that day-ahead forecasting will be performed on future dates, it is unlkely that the lagged load inputs will be utilized. However, hourly temperature data from weather forecasts would likely improve the model performance.

Relying on just these 4 predictors, the forecast does not provide reliable results. The same day from two consecutive years is shown below; in 2016, the load remains flat throughout day, suggesting that there is a weak relationship between the predictor variables and load. It's likely that the model is overfitted, or additional variables (i.e. temperature) are required to properly predict predict load.



6/30/2017 6/30/2016

**Cross Validation**

Our data cannot be considered independent and identically distributed due to the underlying time dependence of load. Most of the k-folds methods involved randomized shuffling and thus did not seem appropriate. I decided to use a time series split to test the performance of my model, which should preserve the temporal order of the data points.

tscv = TimeSeriesSplit(n\_splits=5)

rcross\_val\_score(mlp,predictors,loads, cv=tscv, scoring = scoring\_system)

**R^2 values** [scoring\_system=r2]

([ 0.768166 , 0.89551232, 0.78033706, 0.82411351, 0.86451467])

# near-term forecasting with 8 predictors

([-3.9023946 , 0.1822153 , -0.1284859 , 0.49114066, 0.44155218])

# far-term forecasting with 4 predictors

**Negative MAE** [scoring\_system= mean\_absolute\_error]

([-4.29499536, -5.97659871, -9.24597929, -8.41778215, -6.90001804])

# near-term forecasting with 8 predictors

([-33.50653901, -16.23860276, -21.61272909, -17.09547857, -10.24015834])

# far-term forecasting with 4 predictors

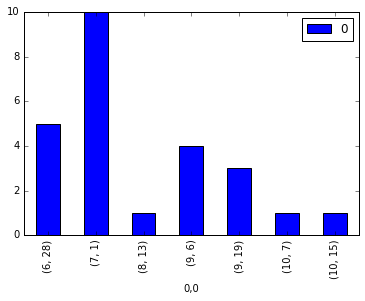
In the far-term forecasting model, the first split is consistently poor, generally improving throughout. This is reasonable considering that the training set is so small early on, and that the predictors rely heavily on seasonality (time of year) – the first training set is essentially winter and the test set is spring.

The near-term model, which incorporates temperature, seems to be less dependent on the size of the training set. This leads me to believe that the far-term model is overfitted

**Implications of Forecasting Error**

It is important to note that not all errors are equally costly for a utility operator – a small underestimation in peak load on a summer day (or winter night, for high VRE penetration grids) could have disastrous consequences in terms of having to switch on highly expensive (and emitting) peaker plants, compared to a low peak load period that can be accommodated by flexible hydro, etc. Presumably, demand response could provide the additional generation at a significantly reduced marginal cost, but accuracy for these high-sensitivity regions is still important.

As a result, I evaluate the forecast error for the highest peak loads of the year, where load exceeds 150. As shown below, the majority of these events occurred on 6/28 and 7/1, where load consistently remained above 140 for 8+ hours on each day.



# of time intervals where load >150

**Next steps:** to investigate forecast errors at peak load:

– retrain my model so that these two dates do not fall within my training set

– mlp.predict() to forecast load for those two days

– compare with actual load graphically and quantitatively

– identify largest errors by absolute & % error

– propose adding a % "buffer" to ensure that potential load underestimations would not significantly impact grid dispatch during these critical load days

– propose ways to train forecast models to prioritize accuracy on high load days, even if it means sacrificing accuracy in other contexts