Lab Course Energy Informatics

LAB SESSION: HOUSEHOLD ENERGY FORECASTING

Group E: Ozgur Akyazi, Oleksandr Puhachov, Hardani Maulana

14 May 2017

Exercise 1. Background and understanding prediction error metrics

During the last decades the attitude of most countries to the power has changed significantly. Along with the general growth of the world economy and rapid spread of technologies, the demand for electricity has been rising, and it is expected to increase by 76% (compared to 2007 levels), according to (Ramchurn et al. 2012). To meet this huge demand, fossil fuels will not be a solution in the future, because of their exhaustibility, the greenhouse effect and its damage on the Earth. As a consequence, there is a constantly growing popularity of renewable energy sources and gradual reduce of energy received from burning fuel fossils.

With the renewable energy being rather intermittent, i.e. its supply depends on various factors: sunshine activity, wind speed, specific local weather conditions, it becomes essential to ensure that customers' demand is met against an available supply. To do this, there are various fields to be enhanced and one of them is smart grids, which will be predicting customers' electricity demand in a timely manner and planning the efficient energy flow from available source to sink, in order to be able to provide energy when needed, especially in peak hours.

Another challenge lies in the low voltage level prediction according to an unstable behavior of household electricity usage. On the other side, distribution network operators need more accurate prediction to help improving their management and planning of the networks. Scenario of storage device solution could be used to ease the circumstance of peak demand. Likewise, the utilization of storage device can be optimized by accurate forecasts to help appropriate planning of charging and discharging.

As discussed, there are cases which could be improved with better accuracy of predictions, and there are multiple measures of a model to be considered, rather than only one. When practical applications of the predictions are integrated into real life scenarios, accuracy, as a dimension, becomes a part of all other dimensions.

(Aman, Simmhan, and Prasanna 2014) present a useful set of metrics to evaluate the performance of forecasts. One of them is *scale independence*, where the error metric is not affected by the magnitude of the scale. While comparing two scale dependent error metrics could be deceiving, the same does not hold for the scale independent errors. An example to scale dependent errors is Root Mean Square Error(RMSE):

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(p_i - o_i)^2}{n}} \tag{1}$$

where o_i is observed/actual value and p_i is model prediction for interval i. As a matter of fact, with different selections for magnitude of scale, this would result in different values. Mean Absolute Percentage Error(MAPE), which is

the normalized version of Mean Absolute Error(MAE), is a widely used scale independent error metric:

$$M = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{o_i - p_i}{o_i} \right| \tag{2}$$

Another performance measure dimension is *Reliability*, which is a measure of whether (or with what percentage) the algorithm/model reports similar outputs. Since the model will be used with new input values, i.e live data from grids, it is expected that the accuracy would be same or similar with the training and test data. So, this is an important metric due to its indication how much one should rely on the model. As an example, Volatility Adjusted Benefit(VAB) is one of the reliability measures.

$$VAB = \frac{\frac{1}{n} \sum_{i=1}^{n} \left(\frac{|b_i - o_i|}{o_i} - \frac{|p_i - o_i|}{o_i} \right)}{std \left(\frac{|b_i - o_i|}{o_i} - \frac{|p_i - o_i|}{o_i} \right)} \tag{3}$$

where b_i is baseline prediction value for interval i. As the differences of observed values from baseline or model predictions are normalized by observed values, this is an scale independent metric. By normalizing the model's improvement with the standard deviation, it creates a volatility measure. As the value increases, the volatility becomes low, and, conversely, becomes high as the value increases. Low VAB may not be preferred because the model does not behave in a consistent manner. Next performance measure is *Application Dependency*. An application independent model could be used in different fields uniformly. On the other side, dependent ones constitute some parameters which are specific to the field of the application. Finally, the last measure is *cost* and it consists of costs in a large spectrum as data collection, effort and time of prediction, training model, reusability. Using these performance measures and, of course, accuracy, one could evaluate and understand the notion of the model or how it will be behaving in a real application.

Above mentioned evaluation measures enable so-called "double penalty" when used for single household prediction. This happens due to volatile and noisy demand nature as well as frequent and irregular peaks, which are absent in middle- and high voltage networks, When forecasting a single household power demand, the feature can be predicted accurately in value, however, it may be displaced in time. Hence, the adjusted p-norm error should be used instead:

$$E_p^w = \min_{p \in \mathcal{P}} ||Pf - x||_p \tag{4}$$

Here f and x - forecast and actual data correspondingly, P - permutation matrix, which allows to describe the match between actual and predicted value. w - is an adjustment limit and is used to restrict the magnitude of displacement. We assume $P_{ij} = 0$ for $|i - j| > w$. Importantly, it is more essential to penalize more peaks that are displaced too late rather than those that are displaced too

early. This allows to make sure a battery is sufficiently charged before a peak demand (Hyndman and Koehler 2006). To summarize, adjustment error helps to "fix" the displacement of predicated values in low-voltage networks which are characterized by volatile and noisy demand.

Exercise 2. Questions

1. What can be used as a benchmark in timeseries prediction?
   (a) ( ) Most common consumption (ZeroR)
   (b) (x) Persistence
   (c) (x) Averaged Profiles
   (d) ( ) Last weeks consumption
2. Check all scale independent error measures which could be used for household energy prediction
   (a) ( ) MAE
   (b) (x) NRMSE
   (c) (x) MAPE
   (d) ( ) RMSE

Exercise 3. Hands on prediction using SciKit Learn

In order to beat the benchmark, we are using two approaches: feature extraction using descriptive statistics and various machine learning algorithms. To be able to compare the benchmark and our algorithm, we are using the same resolution: calculating the average load for the same prediction base and predicting the same interval with that average load data at hand. The results comparison will be explained in the following section.

Feature extraction

We added feature extraction using descriptive statistics which are maximum values, mean, kurtosis, and skew.

```
for key, gmeasures in load_per_device:
        l_measures = list(gmeasures)
    list_loads = list(map(lambda a: a.load, l_measures))
    dc.update({key + "_last": float(l_measures[len(l_measures)-1].load)})
    dc.update({key + "_stdev": np.std(list_loads)})
    dc.update({key + "_max": np.amax(list_loads)}) #new feature
    dc.update({key + "_mean": np.mean(list_loads)}) #new feature
```

```
    dc.update({key + "_kurtosis": kurtosis(list_loads)}) #new feature
    dc.update({key + "_skew": skew(list_loads)}) #new feature

dc.update(target=calc_average_load(pb.actuals))
dc.update({'target_disc': 0})

return dc
```

Machine Learning Algorithms

We have tried all machine learning algorithm available in *scikit* as shown below in the snippet (all classification algorithms are also have been tested, however, they did not perform as well as the regression ones) and in the Figure 1. From all the algorithms, we use SVR from Support Vector Machine since it yields the best result.

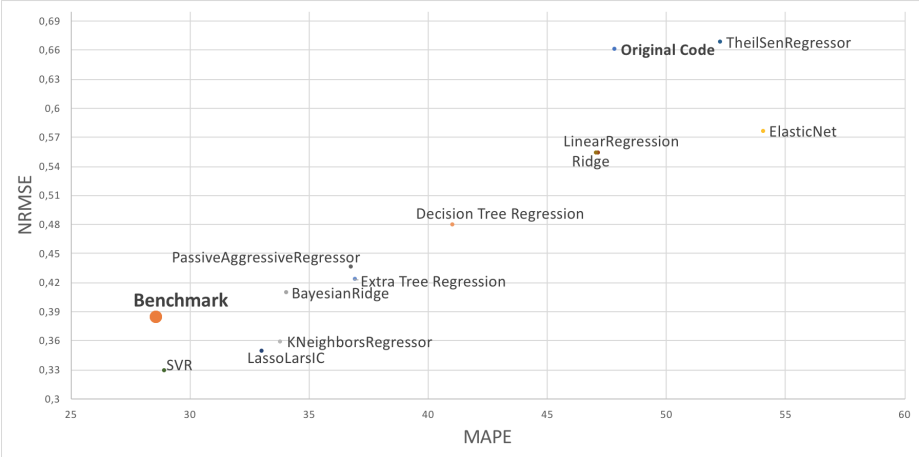


Figure 1: Algorithm comparison

```
solution_algo_ensemble = [
                    ensemble.ExtraTreesRegressor(),
                   ensemble.GradientBoostingRegressor(),
                    ensemble.RandomForestRegressor(),
                   ]
solution_algo_gaus = [
    gaussian_process.GaussianProcessRegressor(),

]
sol_alg_kernrig = [
    kernel_ridge.KernelRidge()
]
```

```
sol_lin_model = [
    linear_model.ARDRegression(),
    linear_model.BayesianRidge(),
    linear_model.ElasticNet(),
    linear_model.HuberRegressor(),
    linear_model.Lars(),
    linear_model.Lasso(),
    linear_model.LassoLars(),
    linear_model.LassoLarsIC(),
    linear_model.LinearRegression(),
    linear_model.PassiveAggressiveRegressor(),
    linear_model.RANSACRegressor(),
    linear_model.Ridge(),
    linear_model.SGDRegressor(),
    linear_model.TheilSenRegressor()
]
sol_neigh = [
    neighbors.KNeighborsRegressor(),
    neighbors.RadiusNeighborsRegressor(),
]
sol_nn = [
    neural_network.MLPRegressor()
]
sol_neigh_class = [
    neighbors.KNeighborsClassifier()
]
sol_svm = [
    svm.LinearSVR(),
    svm.NuSVR(),
    svm.SVR()
]
sol_tree = [
    tree.DecisionTreeRegressor(),
    tree.ExtraTreeRegressor()
]
sol_alg = svm.SVR(kernel="rbf") #choosen alogrithm
```

Hyperparameters

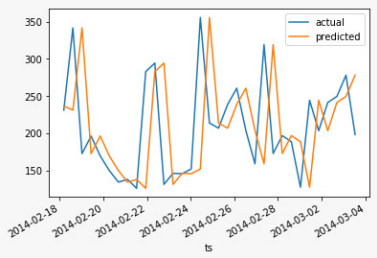
After we had found the most appropriate algorithm for our goal, we tried to optimize the hyperparameters. By setting different C values, kernel functions, number of degrees of the kernel function and gamma values, we have tried a wide range of combinations, but the default settings have yielded always the best results.

The next optimization point was about the horizon, increment and history

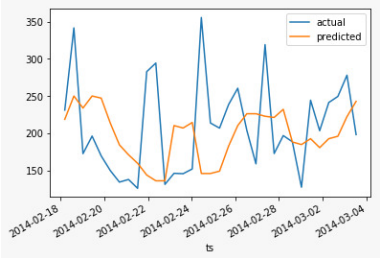
length. Comparing the different algorithms results and new feature extraction, we realized that we are not progressing as expected. However, we still had a room for improvements. Changing the resolution of our predictions helped essentially. Our experiments shown that as the horizon and increment increased, we achieved better results. Particularly, horizon set to 1000 minutes and increment of 600 minutes gave the best results. This can be explained by the following statement: increasing the horizon helps with more data to use when predicting.

After we have a model, using the information from the horizon we predict the next chunk. However, after a certain point this approach does not work well, since we are using data from e.g. 1200 minutes before, which is not very reasonable to do. On the other hand, as increment is increased, the algorithm has to make less predictions for the same period (2 weeks in our case). This leads to predicting only the average of the next increment duration. As in the horizon parameter, this approach gets to its top since too much averaging yields to larger errors. Our experiments also showed that the history length did not affect the results much, hence we decided to use it as 2 days, the smallest value which reports favorable results.

We concluded that 1000 minutes of horizon and 600 minutes of increment with history length 2 days yields the best results 28% MAPE and 0.33 NRMSE, whereas PERSISTENCE reports 28% MAPE and 0.39 NRMSE. Since we were trying to beat the benchmark, 1000, 1000 and 2, horizon increment and history length, respectively, yields 34% MAPE, 0.38 NRMSE while the benchmark reports 37% MAPE, 0.45 NRMSE, which beats the benchmark definitely. Even though this second set of hyperparameters surpasses the benchmark, we are able to get better results with first set (see Figure 2).



(a) Benchmark Prediction Using PERSISTENCE



(b) Result Prediction Using SVR

Figure 2: Comparison of Prediction

Discussion

As a result, we have experimented various machine learning algorithms and tried to optimize the algorithm metrics. This helped a lot about reasoning why some changes would work some others would not. Even though we have slightly beaten the benchmark, still it is not a straightforward task to do, since it is obvious that the results could be improved a lot.

References

Aman, Saima, Yogesh Simmhan, and Viktor K. Prasanna (2014). "Holistic Measures for Evaluating Prediction Models in Smart Grids". In: *CoRR* abs/1406.0223. arXiv: 1406.0223. URL: http://arxiv.org/abs/1406.0223.

Hyndman, Rob J and Anne B Koehler (2006). "Another look at measures of forecast accuracy". In: *International Journal of Forecasting*, pp. 679–688.

Ramchurn, Sarvapali D. et al. (2012). "Putting the 'smarts' into the smart grid: a grand challenge for artificial intelligence". In: *Commun. ACM* 55.4, pp. 86–97. DOI: 10.1145/2133806.2133825. URL: http://doi.acm.org/10.1145/2133806.2133825.