Predicting daily incoming solar energy from weather data

ROMAIN JUBAN, PATRICK QUACH

Stanford University - CS229 Machine Learning

December 12, 2013

Being able to accurately predict the solar power hitting the photovoltaic panels is a key challenge to integrate more and more renewable energy sources into the grid, as the total power generation needs to match the instantaneous consumption load. The solar power coming to our planet is predictable, but the energy produced fluctuates with varying atmospheric conditions. Usually, numerical weather prediction models are used to make irradiation forecasts. This project focuses on machine learning techniques to produce more accurate predictions for solar power (see figure 1).

Our strategy to make this prediction is:
- collect, understand and process the weather data,
- perform different machine learning techniques to make the prediction,
- perform some feature engineering aside of the forecast features,
- analyze the results and discuss them.

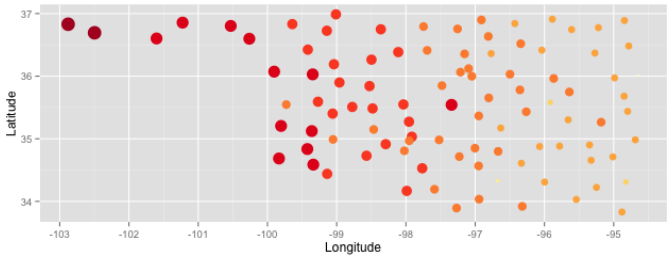

Figure 1: Distribution of the annual insolation at the Mesonet sites (training data).

1 Gathering the data

The data (in netCDF4 format, very popular to manipulate weather data) has been downloaded from the Kaggle website, provided by American Meteorological Society[1] in several files:

1. **weather data**, as the values of 15 weather parameters (such as precipitation, maximum temperature, air pressure, downward/upward short-wave radiative flux,...) forecasted at 5 different hours of the day and provided by 11 different ensemble forecast models. This data is forecasted for a uniform spatial grid (16×9) centered on Oklahoma State and has been collected everyday from 1994 to 2007 (5113 days) for the training dataset and from 2008 to 2012 (1400 days) for the testing dataset.
2. **daily incoming solar energy data**, as the total daily incoming solar energy at 98 Oklahoma Mesonet[1] sites (different from the grid points of the weather data from 1994 to 2007.

With the weather data from 2007 to 2012, for the 144 GEFS[2] grid points, we want to predict the daily solar energy at the 98 Mesonet sites, as shown on figure 2.

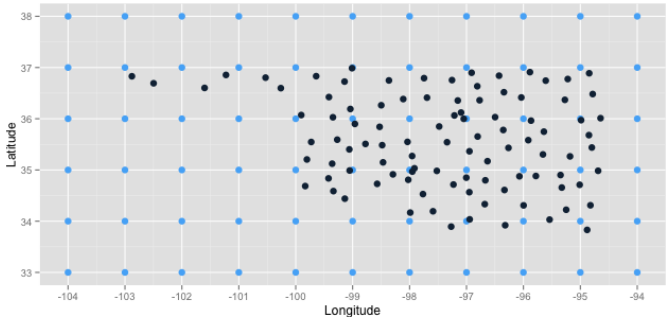

Figure 2: The weather data is known at the GEFS sites (blue), while the solar prediction needs to be done at the Mesonet sites (black).

2 Adapt the data to our needs

For each Mesonet site, we have identified the four closest GEFS weather sites, as shown on figure 3. There are then two possible methods to use the data. For each Mesonet site, we can either interpolate geographically the weather data from the grid to the Mesonet site, or factor all the features in our algorithms to make a prediction.

2.1 First attempt: Interpolate weather data from the grid to the Mesonet sites

For this, we have different options of spatial interpolation to estimate the value of each weather data from the four GEFS stations to the Mesonet site (see figure 3). We chose the inverse distance weighted average, with the distance being cal-

[1] http://www.mesonet.org/
[2] Global Ensemble Forecast System

culated between two points on a sphere from their longitude and latitude (harvesine distance). For each weather data type:

$$X^{(i)}_{Mesonet_j} = \sum_{k \in GEFS} \frac{w_k}{\sum_{k'=1}^{4} w_{k'}} \times X^{(i)}_k$$

with $w_k = 1/d_k = distance[Mesonet - GEFS_k]$

After this step, we could have weather predictions at our 98 Mesonet sites and could perform supervised learning methods, knowing the daily incoming solar energy at those sites. The main challenge was the very large CPU time needed to run the interpolation task. But it was performed only once for all.

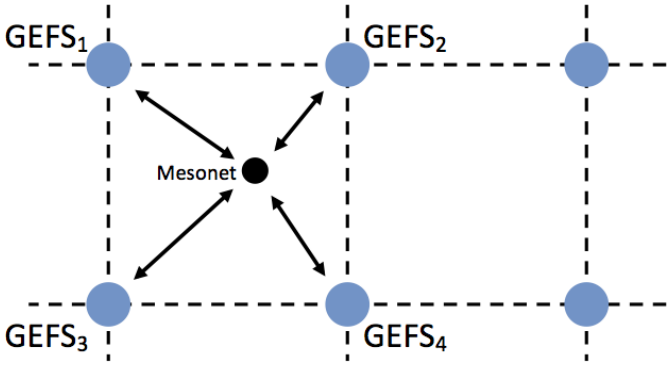

Figure 3: An interpolation of the weather data from the GEFS sites to the Mesonet site is needed.

We were afraid that this pre-algorithm interpolation would turn out to be inconclusive as the aggregation of data would reduce the information we eventually had for the predictions. For example, correlations between the features within each weather station would be lost.

So, we decided to try another method by keeping the interpolation for later in the analysis: for each Mesonet site, we would treat all the data available from the four closest weather stations.

2.2 Second attempt: Factor weather data from the four nearest GEFS grid points as features for each Mesonet site

For each Mesonet site, we then had the four sets of data from the GEFS stations. Two ways were then possible. We could use the four sets of data to make one solar prediction for each of them (four in total) and then interpolate them to the Mesonet site. Or we could make a single prediction for the Mesonet site from the four stations.

Here again, there may not be a direct relationship between the incoming solar radiation at the weather stations and the one at the Mesonet site. That is why we opted for the most conservative method: the interpolation would be indirectly done by adding predictors such as the distance between each station and the central site.

2.3 Verdict

After training our models with these two methods, we observed that pre-algorithm interpolation gave much inaccurate predictions with training and testing errors both unacceptably high compared to the unaggregated model, while being much faster to process (the data handled was then 4 times smaller, which is significant when the lower bound for the total data loaded is around 2GB).

Therefore we decided to go on with an unaggregated model. At this stage, we have weather data for: 98 sites $\times 4$ stations $\times 5113$ days $\times 5$ hours $\times 15$ parameters $\times 11$ models, plus the distances between the Mesonet and each of the four closest GEFS sites.

3 Selecting the predictors

Because of the multiple dimensions of the weather data available, we have decided to make some grouping for the data. As the output should be the daily expected solar power, for each day, site and weather model, we have composed an array of the 15 weather parameters, taken for the 5 different timestamps of the day and the 4 closest stations, which gave an array of 300 predictors for each given day, site and weather model.

After that, we we able to run algorithms on the data for each day, site and model. In the weather prediction industry, these models are usually equally weighted when running forecast softwares. So, we have decided to average the power forecasts from the different models to estimate their combined prediction. However, all the weather parameters are forecasted using the same model, so, not to loose the correlation that we have within the same model (11 models, 300 predictors), we could not work simultaneously with all the models together (300×11 predictors). Therefore the steps chosen to run the algorithms:

1. take the average of each parameter on the 11 models
2. train one model over all the days and sites
3. for each site/day: estimate the incoming solar energy

At this stage, we had weather data for: 98 sites $\times 5113$ days $\times (75+1)$ parameters (distance Mesonet-GEFS) $\times 4$ stations.

This boils down to: $76 \times 4 = 304$ features, and $98 \times 5113 = 501074$ samples, for $98 \times 1796 = 176008$ predictions to make.

4 Understanding the data

Before running any algorithm on the massive dataset, we wanted to get a grasp on the kind of influence some of the features had on the output. So, we took the weather parameters that seemed the most meaningful to us and plotted heat maps. On figures 4 and 5, we can qualitatively distinguish the areas where there is more clouds from those having clearer skies, and also where the solar flux is the highest. Naturally, when

overlapping figure 5 and figure 1, we can notice that shortwave flux has a great impact on the eventual output. It is not the only factor though. The West-East distribution of clouds with the clouds being more frequent in the East, will also probably have high negative correlation with the output. We will verify later these correlations.

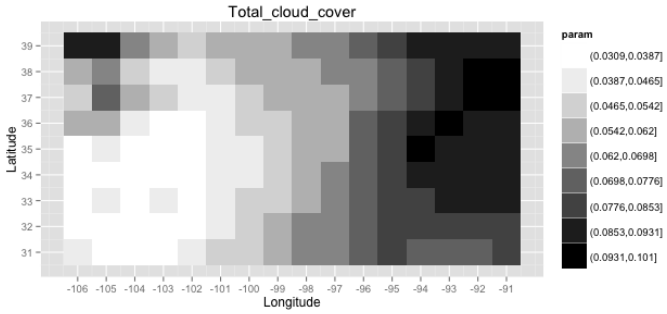

Figure 4: The cloud cover varies a lot along the West-East (clear-cloudy) direction.

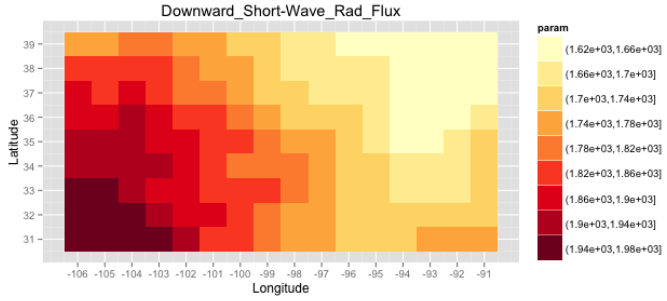

Figure 5: The downward short wave radiation flux represents the radiation coming from the Sun.

Then, we also wanted to have some more quantitative pre-analysis, by measuring the correlation between the factors and the response. Scatterplots were useful in giving a visual estimate of the kind of correlation between them: linear, polynomial, inverse... For example, on figure 6 we can observe that when there have been more than a certain amount of clouds in a day, then the solar energy is very low. This relationship is unlikely to be only linear, but it could be linear piecewise (solar energy decreasing until zero).

5 Regression methods

To be able to compare our approach to others' (Kaggle leaderboard), we have used the MAE[3] formula to calculate the error.

$$MAE = \frac{1}{98N} \sum_{j=1}^{98} \sum_{i=1}^{N} |F_j^{(i)} - O_j^{(i)}|$$

[3]Mean Absolute Error

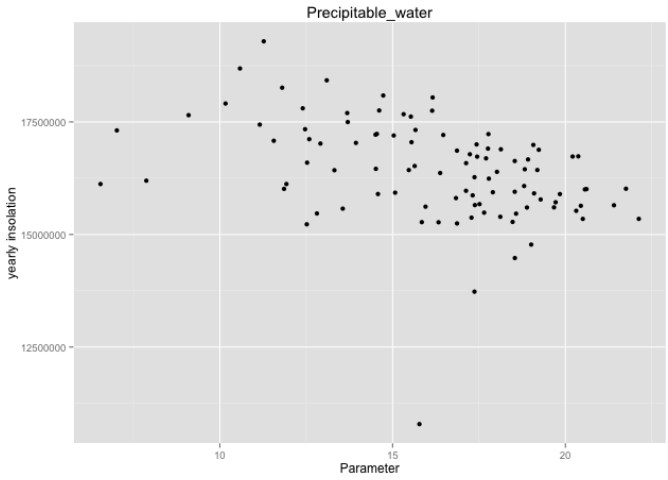

Figure 6: The scatterplot provides us a way to use our scientific intuition.

The mean absolute error is commonly used by the renewable energy industry to compare forecast performance. It does not excessively punish extreme forecasts.

5.1 Simple linear regression

We started with a simple linear regression to make our first predictions. Hence the forecasted daily incoming solar energy for each day and Mesonet site was:

$$F^{(i)}_{Mesonet_j} = \sum_{k=1}^{304} \theta_k \times X^{(i)}_{j,k}$$

To determine the coefficients θ_k we have trained our model by minimizing

$$RSS(\theta) = \sum_{j=1}^{98} \sum_{i=1}^{5113} (F_j^{(i)} - O_j^{(i)})^2$$

To assess the bias variance trade-off, we have divided the training set into two subsets: the first one with the data from 1994 to 2006 (12 years) used as the training set, and the other one with the data from 2007 to 2008 (2 years) used as the validation set. We have trained different models by varying the size of the training dataset and computed the corresponding cross-validation error on the validation test. Then, we have plotted both the training and test MAE of each model to show the "learning curve", on figure 7.

The learning curve for the linear regression model shows the evolution of the learning and testing errors. For 304 predictors, the sample does not seem to be large enough below 10 years. From 10 to 12 years of training samples, for a testing set of 2 years, MAEs converge and it seems that we train a model without too much bias nor too much variance. So, it seems that 12 years of data should be enough to train a model of 304 features.

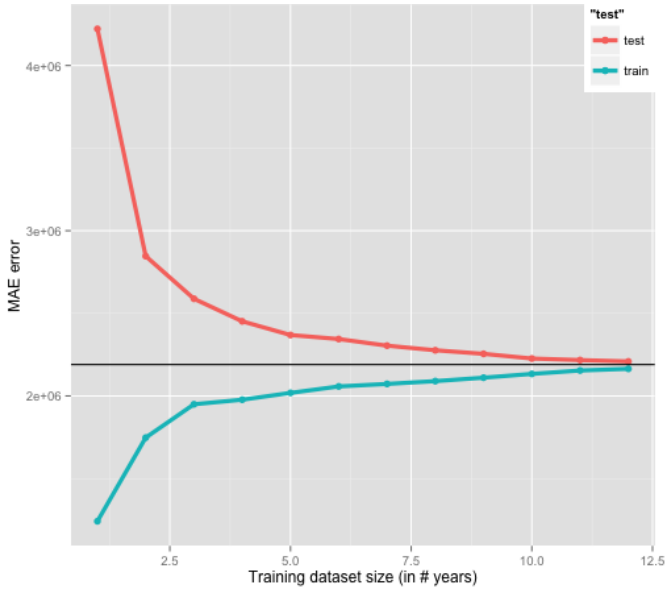

Figure 7: Learning curve for a simple linear regression: training set within 1994-2006 (1 to 12 years), testing set from 2007 to 2008 (2 years).

We could then apply our model to the full training set to get predictions for the Kaggle testing set. It took about 20 minutes to run on the *corn.stanford.edu* machines. With the first submission, we reached the $80^{th}/160$ position in the competition.

If we did not have the test error calculation module provided by Kaggle to evaluate our next models, we would have calculated the evolution of the test learning curve for the different models adopted. On figure 8, we can see that the models get more and more accurate with an increasing sample size.

And then, we have tried to use other more advanced models.

5.2 Lasso and Ridge

As an alternative to a Simple Linear Regression, we can fit a model using techniques that shrink the coefficient estimates towards zero, reducing their variance and making more stable predictions. We have selected two different methods, Ridge and Lasso. Ridge regression minimizes

$$RSS(\theta) + \lambda \sum_{k=1}^{304} \theta_k^2$$

Lasso regression in addition to constraining the coefficient estimates, performs feature selection by setting certain coefficients exactly to 0. It minimizes

$$RSS(\theta) + \lambda \sum_{k=1}^{304} |\theta_k|$$

λ is a tuning parameter that controls the shrinkage of the coefficients. The optimal value of shrinking was obtained by cross-validation on the training dataset.

5.3 Random Forests

Then, we wanted to try more complex methods that could handle the large number of features and the highly non-linear and complex relationship between the features and the response of the data, that has been observed during the first visualizations of the data. Tree-based methods (decision trees) seemed to be a good match.

Random Forests builds a large number of decision trees by generating different bootstrapped training data sets and averages all the predictions. But when building these trees, each time a split in a tree is considered, a random sample of m predictors is chosen from the full set of p predictors. The classifiers may be weak predictors when used separately, but much stronger when combined with other predictors. Randomness allows weak predictors to be taken into account and uncorrelates the trees.

Two tuning parameters are needed to build a Random Forests algorithm: the total number of trees generated and the number of features randomly selected at each split when building the trees. To determine optimal values for those two parameters, we have run several cross validation models and selected those which gave us the best results. We have started with values around those given in the litterature[2] (Typically a good value for m is $\sqrt{p}$ which is around 17 in our case) and explored the different learning curves given by models. Eventually, we ran our Random Forests algorithm with 15 predictors on 3000 trees.

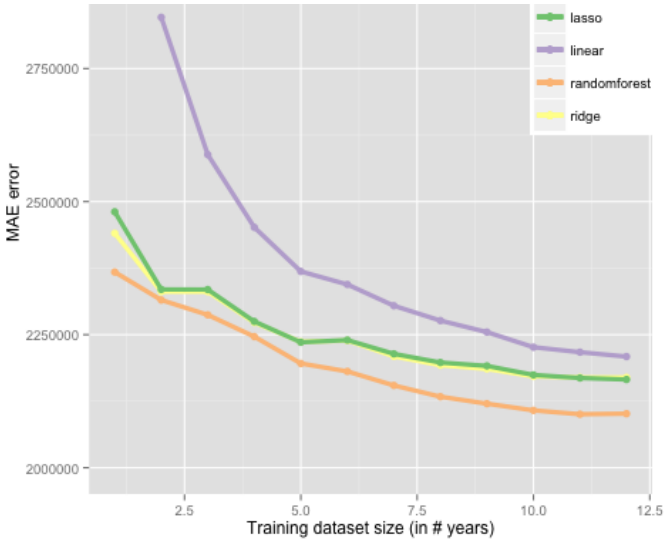

Figure 8: Learning errors (MAE) for different models.

5.4 Models comparison

On figure 8, we can see that the most successful models are given by using Random Forest methods (6% more accurate than linear regression, 54^{th} on the Kaggle board).

6 Additional features

To help our predictions, we have tried other features: as Environmental Engineers, we know that the incoming solar radiation to the Earth heavily depends on two main parameters, the time of the year and the location. But we have also added other parameters:

1. time: the incoming solar energy high relies on the spatial position of the Sun, which depends on the seasons, so we have added a categorical feature to factor the month of the prediction day.
2. location: the solar incident varies with the latitude, but we have also added the longitude of the Mesonet site as we have observed graphically that the irradiation also relies on the longitude, even though this may not apply to other places (see figure 9)
3. altitude: we have found a high correlation between longitude and altitude (and irradiation) in Oklahoma, so we have also added the altitudes of the sites and GEFS stations.

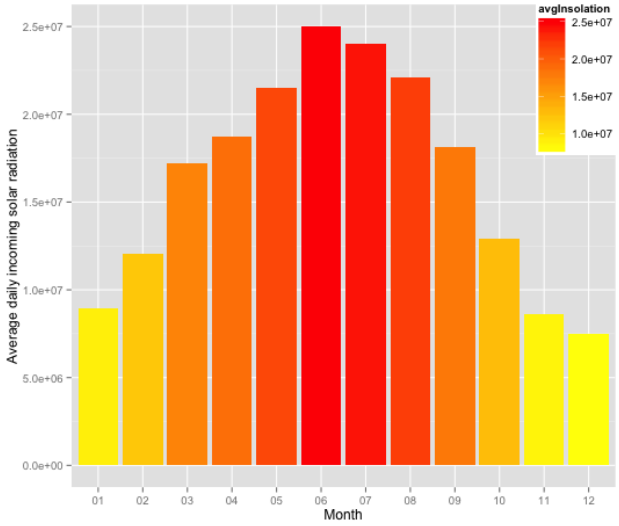

Figure 9: Average monthly insolation

7 Results analysis

With these additional features, we have submitted another prediction file to Kaggle. It reduced our MAE by 9% (40th on Kaggle), compared to predictions made by the Random Forests algorithm on the raw data alone.

On figure 10, a visualization of the relative errors shows an accuracy of our predictions of 9 to 15%, which is quite satisfying. We can notice that the error values tend to be larger on the eastern part of the plot, whereas the smallest error values are located on the western side. Keeping in mind the 3-hour sampling of the weather data, very sudden changes in the weather parameters cannot be noticed in averaged datasets. Therefore, the prediction of solar power can be less accurate in "wet" climates (such as the East of Oklahoma) due to the higher variability of weather (clouds, precipitation), than for "arid" climates (such as the West of Oklahoma).

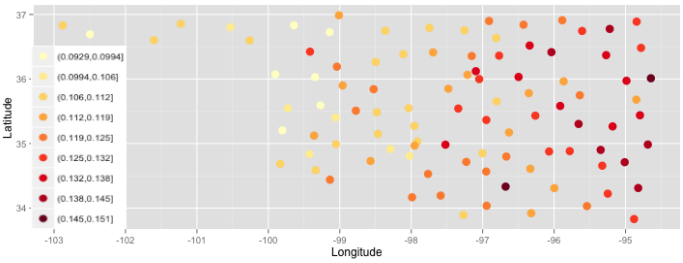

Figure 10: Relative MAE plot for each site

Our last task was about identifying the main features of the model: it turned out that the upward solar flux features were much more informative than the downward ones. Indeed, the upward flux directly reflects the amount of energy that is reflected back to the atmosphere, that is a fraction of the incident irradiation that actually hits the ground.

8 Conclusion

This machine learning project was our first hands-on experience with real big data. The project was about data preparation for a big part: it involved data understanding, sorting and reframing. Then, we had to think about ways to run our algorithm, as we needed machines capable of handling about 2GB of input data at once. It was challenging, but that made us really think about ways to save time and resources: how to reduce the computational load of our code, how relevant it is to make backups of intermediate files, how useful it is to run calibration test algorithms before launching codes that would run for tens of hours.

It was definitely challenging to work with this big data. We have also learnt a lot about implementing algorithms in real life, as we were not working in a fully academic environmental anymore.

And finally, as we tried to understand the different correlation relationships between the parameters and the forecasts, we surprisingly also got a better understanding of solar prediction from an energy engineer point of view.

References

[1] AMS *2013-2014 Solar Energy Prediction Contest*, `http://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest/`.

[2] James, G., Witten, D., Hastie, T., Tibshirani, R, 2013. An introduction to statistical learning. Springer. pp. 303-320.