Grant Davis
CS 458
12/13/2022
Predicting Solar Power Generation Using Current Weather Data

Abstract

Renewable energy sources are becoming increasingly common in power grids around the world. Unlike their fossil fuel counterparts, these energy sources cannot increase or decrease their output to meet demand. Instead, they rely on their respective natural supply, such as the sun or wind, and must be managed to provide power to the electrical grid in an optimal way. In the case of solar power, many factors influence total power output. By analyzing current and historical weather forecasts, alongside their respective power outputs, machine learning can be used to predict future power generation. I propose a Support Vector Regressor to accomplish this task using a dataset of two years' worth of hourly solar power generation and associated weather conditions. The model, after extensive tuning, produced accurate predictions for solar power generation 24 hours in the future. There was a small enough margin of error to accomplish the goal of predicting future supply to manage future demand with some degree of certainty. The model could be improved by incorporating future weather forecasts alongside current conditions.

Index Terms—regression analysis, solar power generation, support vector machines.

Introduction

Predicting solar power generation carries several uses. Most importantly, it can be used to minimize the waste of electricity. Power plant operators, with the knowledge of future short-term power generation, can better match consumer power demands. As renewable energy cannot be controlled, it requires more stringent management. With these predictions, the goal is to maximize power output and minimize any waste when the demand is lower than the supply.

Without such predictions, solar power grids cannot decrease revenue loss. Optimization of the power grid is necessary to do so. Traditionally, this has been difficult due to the inherent fluctuations in solar power output. Couple this with variability in power demand, and a difficult control and optimization problem is presented.

Integration of short-term weather forecast data is a key method to predict solar power generation. Using historical data to predict future output, one can reasonably estimate power generation for a given day and time. With this knowledge, the problem is now reduced to managing power demand, since power generation is reasonably known. In this way, one can then realize the gains from an optimized solar power grid as far less power is being wasted. II.

Related Work

Existing literature on the topic of forecasting solar power generation varies in scope and methodology. Some papers were focused on microgrids, while others proposed general methods. Chen et al. [1] implemented a radial basis function network, using both current solar power and a self-organized map of numerical weather predictions. The model performed very well, even with somewhat inaccurate weather predictions, obtaining a mean absolute percentage error of roughly 10%. Sharma et. al [4] discuss a multitude of prediction models, ultimately choosing a Support Vector Machine with a radial basis function kernel as their leading model. Their work also incorporated short-term weather forecasts to further increase accuracy. Rodriguez et al. [3] focus on very short-term (future 10 minutes) energy generation predictions at the microgrid level using an artificial neural network.Their model produced results with a strong correlation to the actual data, but it is unclear as to the usefulness in longer-term prediction applications.

Existing methods seem to be similar in their results, but the goal of the research varied. Predictions 24 hours ahead were less common compared to short-term (less than 3 hours ahead) forecasting. The incorporation of weather forecast data for the location was a novel introduction but was out of scope for my research. However, the existing work found success with this inclusion. Model selection, too, was similar as the most common candidates were random forests, Support Vector Machines, and neural networks. The significant training time of the latter model may have been a reason the former two were more frequently used. Additionally, in some cases, a high number of weather variables were used to train the model [3]. Similarly, data attributes were sometimes created from the given data, such as a ”season” variable. It is interesting to note the lack of ”day of year” attributes, as it seems there would be some effect on the model’s decision. However, the researchers may have felt a generalized attribute (akin to ”season”) was sufficient for this purpose.

Methods

Data Preprocessing

When preprocessing the data, it needed to be determined which attributes to use or ignore. Correlation between variables is explored, as shown in Figure 1, but ultimately this did not weigh heavily on the final variable selection. I chose to include all attributes to train the model with, as each provided some information about the current weather or time of year, which all contributed to the total power output. Some models ignored the day of the year, but I included it as I believe it does play an important role in power output. For example, some date ranges have shorter days (periods in which the sun is shining) and the model should take this into account. However, the issue with the provided timestamps is their inclusion of the year, which is not a useful piece of information for the model. This was removed, and all dates (as the DateTime object requires it) use the year 1900. This method was chosen over an integer value because the data did not have to be cast into an integer type.

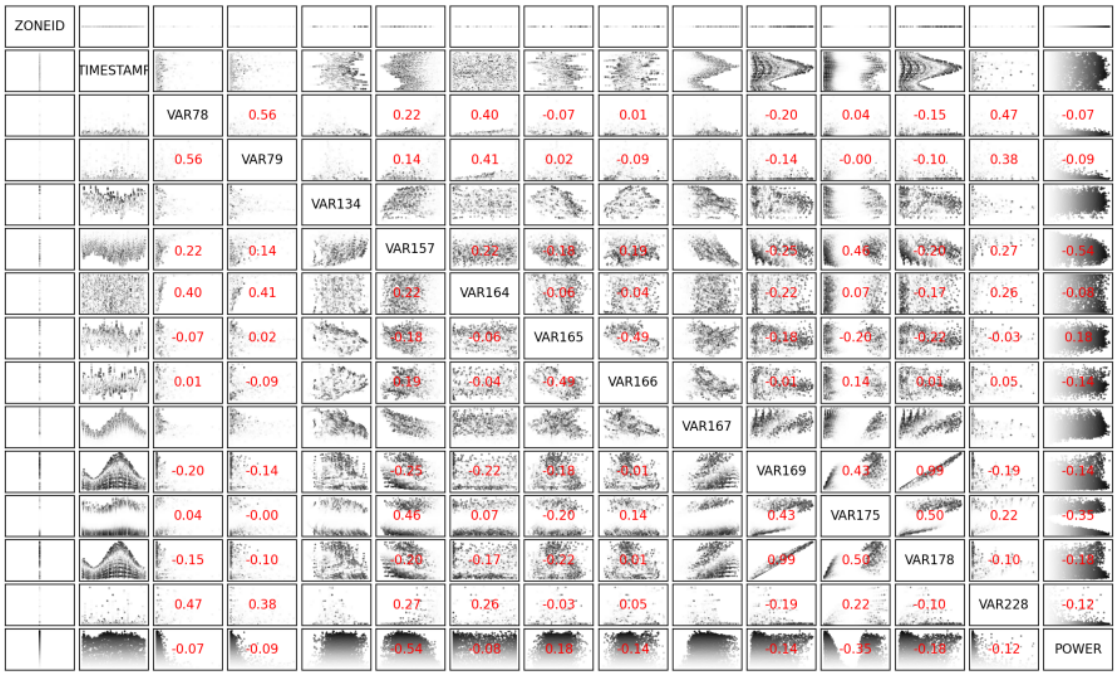

Fig. 1. Variables are plotted against each other as a scatterplot to investigate correlation

After all the attribute data was loaded, it was normalized between the values of zero and one. This was done so no single attribute would have more weight than any of the others in the model. With the normalization, I also chose to perform this on a per-zone basis. Zones represent solar power plants in separate locations. With training data, this did not pose an issue as it was all from zone one, but when importing the test data, the zones were split up and normalized individually. This had the unintended side effect of allowing for the use of loops when running the model against multiple zones, which made life easier. Power was not normalized in this project. Additional preprocessing occurred outside the application as well. The provided file was split into test and training files. These two files were then split up into two files each, with a 24-hour offset between the two. When loading the data, the Solar class, which holds the data, is created four times, and the model is run against each pair of training and test data.

Model Selection

Multiple regression models were available for this project. Linear Regression did not seem suitable for this application, and after some deliberation, the choice was made to use a Support Vector Regressor. With this model, it may have benefitted from selecting some n number of best features, but I did not do this. Other papers implemented this regressor as well, however, a Random Forest Regressor model was popular too.

The model is trained on zone one. It is then used on the testing data. Some additional preprocessing occurred here, as the 24-hour offset files did not account for different zones in the middle of the file. The application checks the number of samples for both data and power, and if there is a discrepancy, it will create a 24-hour offset on the offending array. After the predicted values have been created, they are compared against the actual values. Two scoring metrics are used: root mean squared error and mean absolute error. With these two scores, the objective is to receive a number as close to zero as possible. The inverse of this number could be interpreted as accuracy.

Parameter Tuning

The first run of the application used default parameters for epsilon (0.1), regularization (1.0), and tolerance (0.001). These provided baseline results for the model, which were not bad, to begin with. However, some tuning was necessary to provide even more optimization.

The initial test used the following variables:
- Tolerance = [0.1, 0.00001]
- Regularization = [0.6, 1.4] skipping every other
- Epsilon = [0.04, 0.14] skipping every other

After analyzing the results, it was clear a lower epsilon provided the most benefit. Regularization was not found to have a significant impact but was later optimized further. Tolerance showed the best results at one specific level, so this parameter was set.

The second test followed with new ranges for the parameters:
- Tolerance = [0.0001]
- Regularization = [1.3, 1.5]
- Tolerance = [0.02, 0.04]

These results, while close, showed that the ideal parameters for this application were a tolerance of 0.0001, a regularization parameter of 1.3, and an epsilon of 0.03. With these parameters, there was some improvement in model performance, particularly for Mean Average Error, as shown in Table I. The improvement can further be put in context with a scatter plot of the actual values subtracted by the predicted values, as shown in Figure 2. Elements closer to the red line (representing 0.0) are more accurate. Fig. 2. Zone 3 differences in power results after parameter tuning.

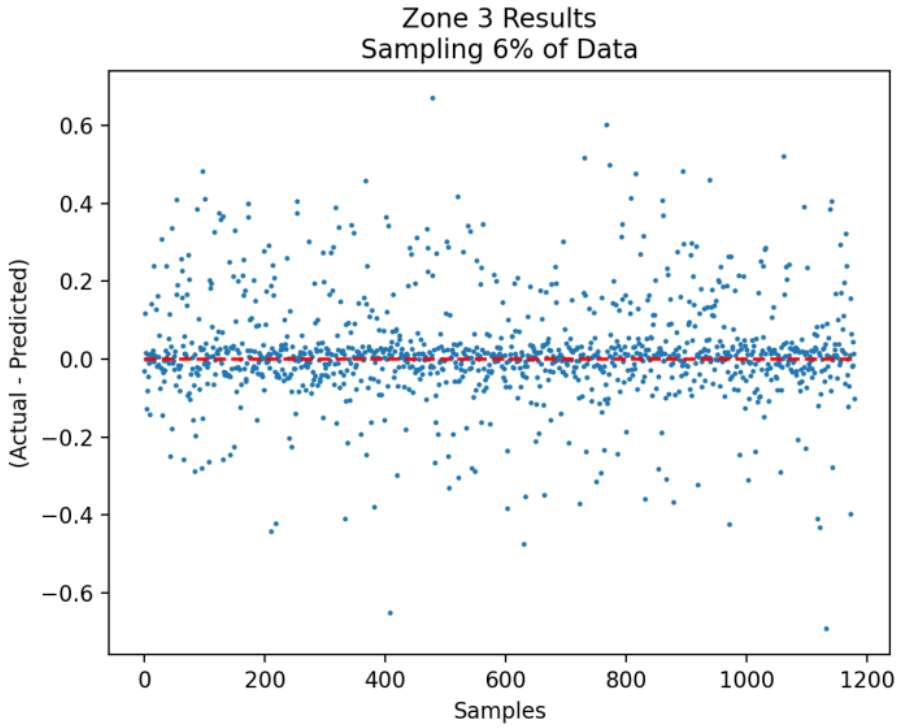

Fig. 2. Zone 3 differences in power results after parameter tuning

Table I Hyperparameter Tuning Results

Metric	Scores (Average of Three Zones)		
	Baseline	***Tuned***	***Improvement***
RMSE	0.140737	0.137754	2.1%
MAE	0.099321	0.084556	14.9%

Results

Verification and Testing

The two evaluation metrics in use for the project provide a fairly accurate representation of the model's performance. The model can additionally be verified by sampling tests and predicted data against each other, comparing the differences. It is expected they will be low, or near the level of the scoring metrics.

At certain points during application development, the program would output which values it was examining when training the model. This verified that the input data was correct. Verification of the values came in particularly handy when splitting test data into multiple zones and creating a 24- hour offset. By comparing the data outputted to the file itself, I could verify the application was segmenting the data correctly.

Experiment Results

The model performed decently well when run against the test set. The root mean squared error and mean average error was low, but not amazing, as listed in Table]II. Considering the power data was normalized between zero and one, and with the two scores averaging about 10%, we have reasonably found our error rate. Flip the error rate, and that indicates the model has an accuracy of about 90%.

Table II Support Vector Regressor Scores

Scoring Metric	**Results**			
	Zone 1	***Zone 2***	***Zone 3***	***Overall***
RMSE	0.129920	0.148652	0.134690	0.137754
MAE	0.077510	0.092160	0.083999	0.084556

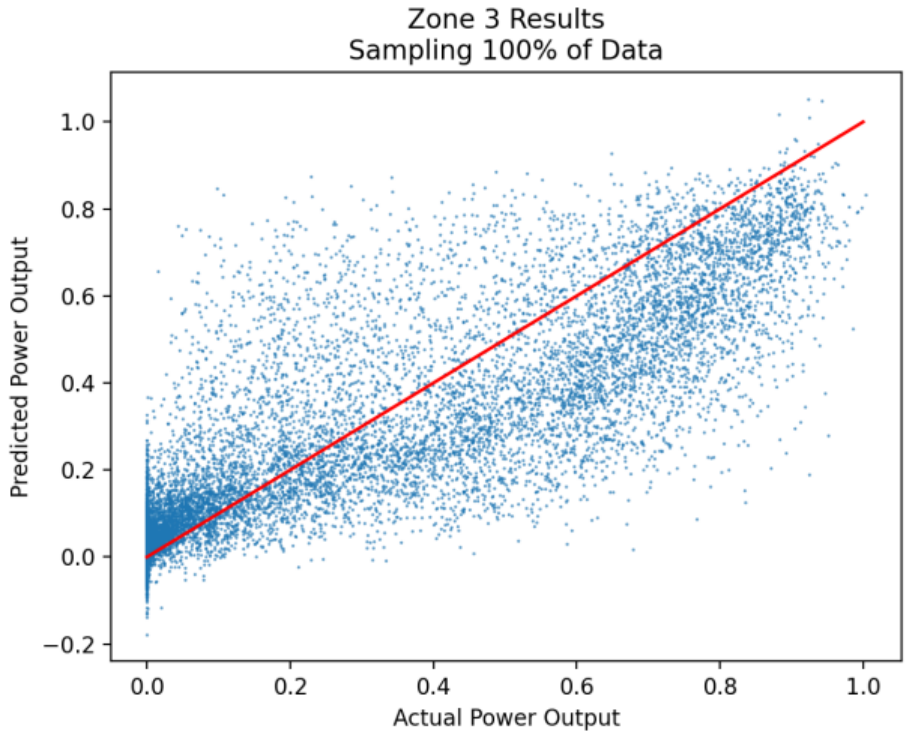

Fig. 3. Zone 3 actual values against expected values.

Examining results for days during seasons of the year also provides a good look at the model's fit. For this test, four months were selected, with a day from about mid-way through each being picked. The

results for January, April, July, and October are shown in Figure 4 and demonstrate good fits for each. It is clear the model prefers to underfit its predictions. This is further apparent in Figure 3, a plot of the actual generation against the predicted generation. There is a clear trend of underfitting, even to the point of some negative amounts when actual power generation is near zero, indicating some issues with the model.

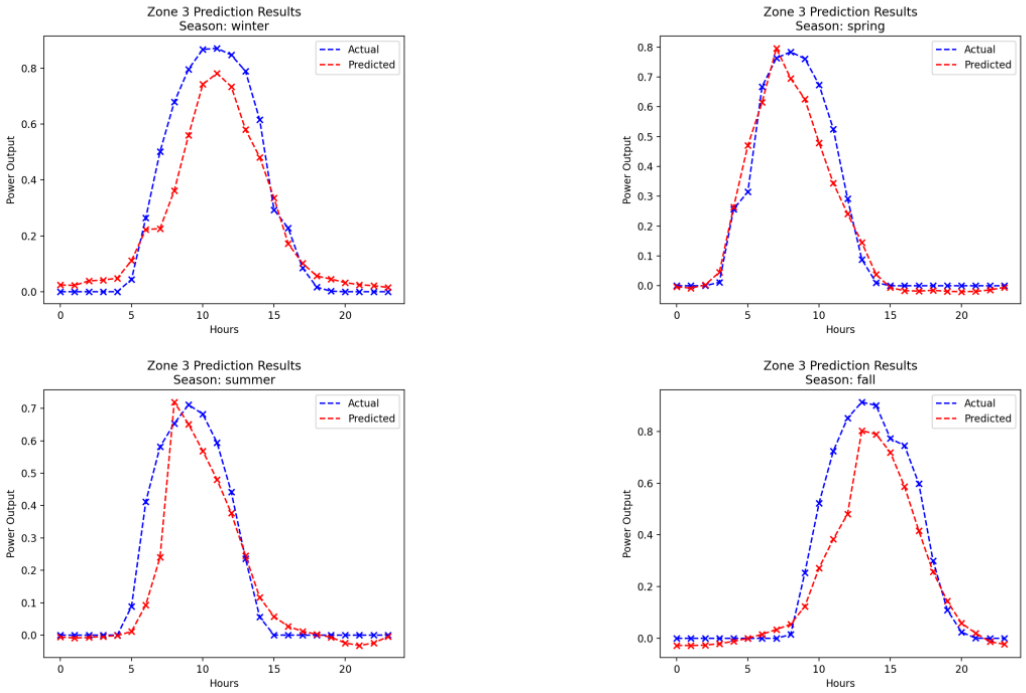

Fig. 4. Zone 3 curves of expected power and actual power on days during four seasons.

A Support Vector Regressor may not have been the ideal choice for this application, but its results were still far from abhorrent. There was enough flexibility with parameter tuning to obtain an increase in accuracy. Visually comparing the results to actual power output demonstrates a good fit, despite the underfitting. This model does not take a significant amount of time to train and test, demonstrating an advantage over other models, such as neural networks, used in [3]. It seems to be robust if there are outliers and has a fairly high accuracy "out of the box." However, the model I implemented is not as suited to large datasets compared to a Linear Support Vector Regressor. Additionally, the model could have benefitted from future weather conditions, in the form of short-term forecasts, to further increase accuracy.

Conclusion

Solar power generation can be accurately predicted 24 hours ahead given current power output and weather data. While the model cannot provide completely precise measurements, it does fulfill the goal of matching consumer demands. Since the model predicts power with an acceptable margin of error, the grid can be optimized one day ahead. With the world moving towards more renewable energy sources and attaching them to the existing power grid, models like the one proposed can provide a level of

management not traditionally available with fluctuating generations. This research, alongside plenty more, will hopefully accelerate the world's transition to sustainable energy.

References

[1] C. Chen, S. Duan, T. Cai, and B. Liu, "Online 24-H solar power forecasting based on weather type classification using artificial neural network," Solar Energy, vol. 85, no. 11, pp. 2856-2870, Nov. 2011.

[2] J. Cochran, M. Miller, O. Zinaman, M. Milligan, D. Arent, B. Palmintier, M. O'Malley, S. Mueller, E. Lannoye, A. Tuohy, B. Kujala, M. Sommer, H. Holttinen, J. Kiviluoma, and S. K. Soonee, "Flexibility in 21st Century power systems," National Renewable Energy Lab, May 2014.

[3] F. Rodr´ıguez, A. Fleetwood, A. Galarza, and L. Fontan, "Predicting ´ solar energy generation through artificial neural networks using weather forecasts for microgrid control," Renewable Energy, vol. 126, pp. 855- 864, Oct. 2018.

[4] N. Sharma, P. Sharma, D. Irwin, and P. Shenoy, "Predicting solar generation from weather forecasts using machine learning," 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm), Oct. 2011.