COMPUTER SCIENCE AND DATA ANALYTICS

Course: Intro to Big Data Analytics

TERM PAPER

The Application of Self Organizing Map Clustering
to
Solar Energy Production Forecasting

Student: G39894949, Narmin Jamalova

Instructor: Dr. Abzatdin Adamov

Baku 2021

The Application of Self-Organizing Map Clustering to Solar Energy Production Forecasting

Narmin Jamalova

ADA University, Azerbaijan / George Washington University, U.S.A.

Email: njamalova2019@ada.edu.az

***Abstract*—Accurate solar energy production forecasting is becoming more relevant as the world is shifting towards greener energy alternatives. Traditional literature focuses mainly on improving model parameters and getting hold of more data to help increase the accuracy of predictions, while some other applicable algorithms, already used and tested in similar contexts, remain not so widely researched in the solar sphere. This paper presents the application of the Self-Organizing Map (SOM) clustering algorithm to the solar energy prediction challenge. The method is tested on a 2016 HI-SEAS weather station dataset of various meteorological observations, including wind speed, humidity and atmospheric pressure. First, SOM is used as a pre-processor to map similar data points to nodes that are then clustered into three distinct units. Then, each cluster is fed into a separate Support Vector Machine regressor to compare the accuracy of the prediction results (SOM-SVR) versus that of not applying SOM at all (noSOM-SVR). Performance is evaluated by comparing the root mean squared error of each model. Results indicate that the application of SOM-SVR increases the accuracy on average across all clusters versus noSOM-SVR. This can offer more insight into how to obtain better prediction accuracies coupled with further research in this field.**

Keywords—solar, energy, machine learning, SVM, SOM, weather forecasting, energy forecasting

I. INTRODUCTION

The importance of green energy solutions is increasing every year as traditional energy resources are dwindling and the fear of global warming is driving on. Many countries are on the lookout for cleaner and easier alternatives. Amongst these, the most convenient, reliable and easy-to-operate alternative source of energy is solar.

According to data sourced from the bp statistical energy review (2019) [1], solar energy generation experienced a relative change of +326% from 2010 to 2019, in terms of TWh. However, solar energy production is unstable, as it is heavily dependent on such factors, as:

- weather conditions (sky cloudiness, atmospheric pressure, humidity);
- geographical locations of power plants (i.e. grids of photovoltaic cells);
- power plant configurations (susceptibility to corrosion and other maintenance faults).

Moreover, maximum solar energy production is achieved during mid-day under clear sky conditions due to the sun's movement to zenith, after which it gradually falls off until the next day. All of these fluctuating variables lead to an unstable supply stream trying to meet much stabler (albeit, gradually increasing) demand.

One of the solutions to an unstable supply stream is to efficiently store the produced surplus energy for future demand. However, the storage cost can be quite expensive (particularly, for poorer countries) and storage release plans still heavily rely on a reliable understanding of upcoming weather conditions. Hence, accurate forecasting of solar energy generation is crucial to meet demand, have efficient energy storage plans and reshape the current energy mix in favor of greener power.

There are several solar forecasting methods in use today and they are broadly classified into the following methods:

a) statistical (artificial neural networks, support vector machines, regressions, etc.);

b) physical (mathematics and physics equations, simulations);

c) hybrid (ensembles of both above).

Regardless of the significant advances in this area, many forecasting algorithms are still underperforming, especially when it comes to forecasting non-clear-sky weather conditions. Most of the models in place are focused on feature selection and improvement to model parameters (such as number of hidden layers, time lags, etc.) as evidenced in the literature review to follow. Yet, some of the some other applicable pre-processing techniques remain not so widely tested in this field.

One of such under-tested mechanisms is the Self-Organizing Map (SOM) clustering algorithm. Unlike the solar energy prediction sphere, this technique has been thoroughly tested in other areas, such as wind energy forecasting and electric load balancing. For instance, in Tan et al. (2016) [2], SOM is applied as a pre-processor to an Extreme Learning Machine model to help improve the accuracy of predictions. Lee et al. (2019) [3] cluster their data into node subsets based on similar characteristics and feed the clustered results to an ensemble model to better predict electric load balancing.

Hence, the purpose of this paper is twofold:

- To review the existing machine learning techniques for solar energy production forecasting and some of the insights gained by researchers about the data.
- To analyze how a yet not widely tested SOM clustering method can contribute to a better prediction accuracy for solar energy forecasting, coupled with a wider variety of features than previously considered.

The structure of the paper is quite simple. The first section discusses why solar energy production forecasting has gained importance in the time of big data and analytics. The second section presents a comprehensive literature review of existing studies and methodologies, including the progress made in applying SOM to the subject field. The third section describes the dataset, methodology applied to analyze SOM clustering method's relevance to the field, as well as the algorithm itself. The fourth section presents the results and discusses them in detail. The final section

concludes this paper with a summary of key points and recommended actions for future research.

II. Background

There is a lot of historically-collected solar energy data that up until now has been difficult to analyze due to technological limitations. The advent of increased processing capabilities for both structured and unstructured data, as well as machine learning techniques presented a great opportunity to extract valuable insights from this data pool and improve on traditional physics-based forecasting equations.

Data science tools facilitate effective data wrangling and machine learning models can learn from data in real (or almost-real) time to keep improving their prediction accuracies. All of these can lead to faster and better decision making in the energy industry, more efficient operations and a better understanding of supply and demand conditions.

Moreover, evermore open-source data is becoming available for researchers to try modeling on. Examples include the data provided by International Renewable Energy Agency [4], Global Solar Atlas [5], Solar Energy Industries Association [6] and more.

III. Literature Review

Broadly, existing research in the area of solar energy forecasting focuses on the following time horizons:

- very short-term (0-6 hours);
- short-term (a day or several days ahead);
- long-term (months or years ahead).

In their research, Gensler et al. (2016) [7] use a numerical weather prediction model as input parameters for training a day-ahead solar energy production model for a German solar farm. The numerical weather prediction model (NWP) provides such meteorological variables, as solar irradiance, temperature, wind speed etc. Min-max normalization and a clear-sky filter are applied to the dataset of produced weather parameters. The clear-sky filter removes the times of the day where the sun is not shining brightly or at all. Auto-encoder is trained and Auto-LSTM (Long-term Short-term Memory Network) is appended with a time lag of two steps. The Auto-encoder learns the compressed representation of features that are then passed on to the LSTM as input sequences. The accuracy of the LSTM depends on how well the resulting model recreates the inputted sequence. The resulting root mean squared error (RMSE) is at 7.1%.

The shortcoming of this research seems to lie in the application of the clear-sky filter to the dataset. Solar energy can still be produced, albeit in lower amounts, in no-clear sky conditions [8]. Hence, this research is too centered on good weather parameters which probably describes the low error result achieved.

Monteiro et al. (2013) [9] use a multi-layer perceptron neural network, with no particular pre-processing applied, to generate an hourly solar energy prediction forecast. The network has seven neurons and two hidden layers. The input features (hourly solar irradiance, temperature, etc.) are also obtained from an NWP. Cross-validation is applied to avoid overfitting and a sixty-forty train-test split is used for modeling. The results are better than that of the numerical weather model, with RMSE at 12%. Yet, the shortcoming is that the network "forgets" about the important data points flowing through its layers as they propagate through deeper parts of it.

Another major finding of different researchers is the fact that the NWP data, in its raw form, is never quite enough for an accurate analysis.

Buwei et al. (2018) [10] propose the usage of remote sensing data, together with traditional NWP inputs and call this combination "data fusion". The extended data source is then fed into a support vector regressor, achieving only 10%-16% RMSE, depending on the predicted month. Although the dataset was increased in its size, the resultant RMSE is still far from ideal. This highlights the importance of using the right pre-processors (e.g., filters), no matter the size of the dataset.

Other researches opted to consider a bigger dataset but with a clever feature selection pre-processor. Aler et al. (2015) [11] suggest using a ReliefF algorithm on an Oklahoma solar plant dataset to select the features that distinguish well between data points or group similar data points together. Yet, the result of this paper indicates no significant benefit gained from such a technique.

Unlike previously mentioned authors, Gao et al. (2019) [12] generate four different models to forecast solar energy generation in four different seasons. The rationale is ruled by the fact that solar irradiance varies at different times of the year and the variance could be acting as noise in the previously applied models. The input features used in this paper are solar irradiance, air temperature, relative humidity and wind speed, amongst others. Data is min-max normalized and fed into an hour-ahead LSTM network with a time lag of three steps. The achieved RMSE varies between 5% and 13%, signifying a relative increase in quality over some of the previous studies.

Around the same time, Perveen et al. (2019) [13] suggested a more novel approach by introducing forecasts for different weather, as opposed to season, conditions. The studied weather conditions include clear sky, cloudy sky and hazy sky for five different climate zones in India. A long-term time horizon is chosen for predictions with data from 2002 to 2011 used for training and 2012 to 2016 for testing. The tested model is the adaptive neural-fuzzy inference system (ANFIS) which performs preprocessing by converting values to fuzzy numerals (many-valued truth logic). Achieved RMSE varies between 0% and 50%, depending on climate zones and weather conditions. This research indicates that better filtering can help improve accuracy but not all the different weather conditions are captured well enough, hence the higher error rate in some territories. This could be due to sky-filters not necessarily differentiating between drastically different weather conditions (e.g. clear sky with high humidity versus hazy sky with low humidity).

This shortcoming of consideration persists in the research made by Nitisanon (2017) [14], as well. In this paper, SOM algorithm is applied to classify the data into sunny, cloudy and rainy days for solar energy forecasting. Yet, the major disadvantage is that the only variables considered are solar irradiance, time and temperature, with no heed paid to other important parameters, such as humidity and atmospheric pressure. Also, the filtering mask remains questionable, since a different combination of weather parameters (such as sunny day with high wind speed versus cloudy day with low wind speed) could yield similar irradiance profiles. The

resultant RMSE varies between 6% and 38%, most likely for this exact reason.

Tan et al. (2016) [2] propose using a SOM network as a pre-processor for an Extreme Learning Machine (ELM) to predict short-term wind energy generation. The resultant RMSE decreases with the application of SOM as a pre-processor. Yet, again, the input features considered are too few to draw a reliable conclusion.

Hence, it's observed that no weather filters to date have generated the results industry experts could hope for. It seems like the problems lie in :

1. manual categorization of weather conditions into various sky-filters or rainy/sunny/cloudy days, which do not necessarily account for the complexity of weather patterns;

2. too few features used in cases where SOM networks were applied to automatically generate weather condition-clusters.

Hence, the following sections shall focus on using SOM as the automatic weather filter that can account for the complexity of patterns. At the same time, more variables of interest shall be considered to form clusters.

IV. Dataset & Methodology

This section describes the dataset used in this paper and the methodology followed to assess the SOM network preprocessor applicability to solar energy forecasting.

A. Dataset

The dataset for this research was obtained from Kaggle [15] with the original source being the NASA database. Data covers 32,686 meteorological observations made at various time intervals in 2016 at the HI-SEAS (Hawaii Space Exploration Analog and Simulation) weather station.

The variables of interest include date and time, solar radiation (watts per square meter), temperature (degrees Fahrenheit), pressure (Hg) and humidity (%), wind direction (degrees) and wind speed (miles per hour), the time of sunset and sunrise (Hawaii time).

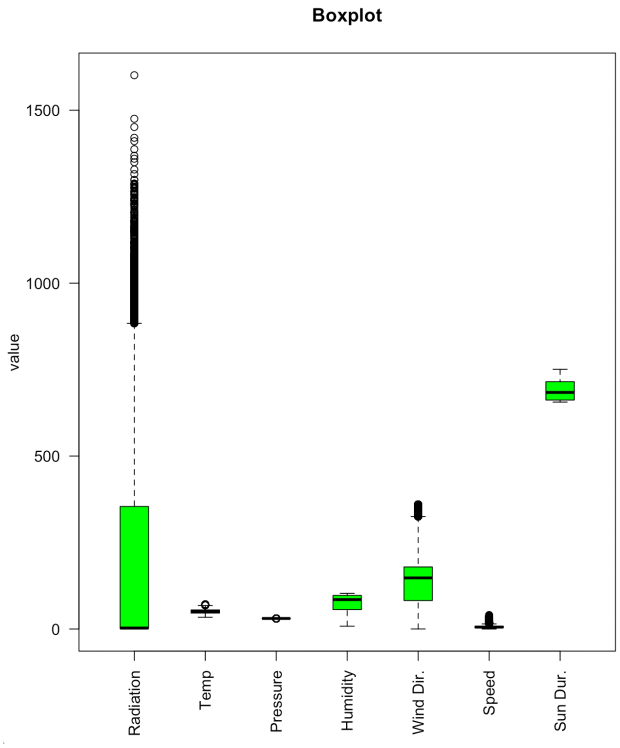

Figure 1: Outlier detection

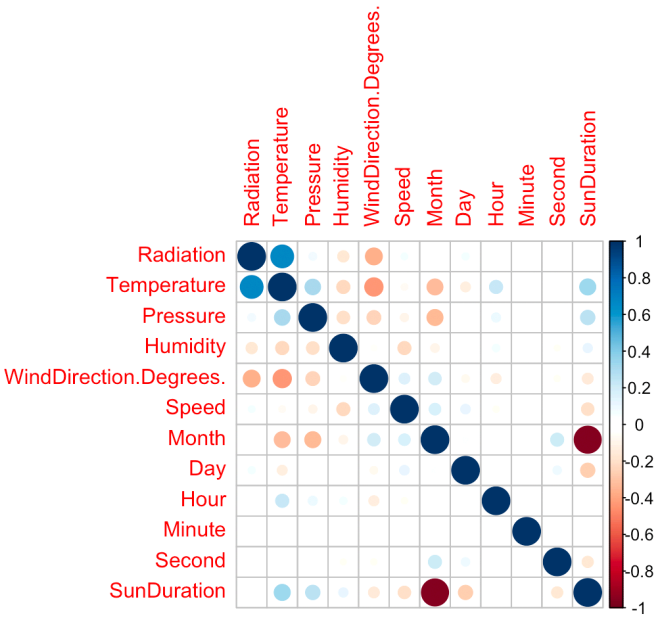

Figure 2: Correlation analysis

Feature engineering and transformation were applied to the dataset as follows:

a) the date and time features were transformed into variables *'Month'*, *'Day'*, *'Hour'*, *'Minute'* and *'Second'*;

b) an engineered feature called *'Sun Duration'* was introduced as the number of minutes between sunset and sunrise.

B. Methodology

1. The transformed dataset was cleansed from outliers in *'Radiation'* and *'Wind Direction'* depicted on Figure 1, leaving 29,365 observations in place.

2. The cleansed dataset was min-max normalized and pair-wise correlations were plotted for better analysis (Figure 2). These plots will be discussed in the next section to follow.

3. Data was split sequentially into train and test sets, using a seventy-thirty split.

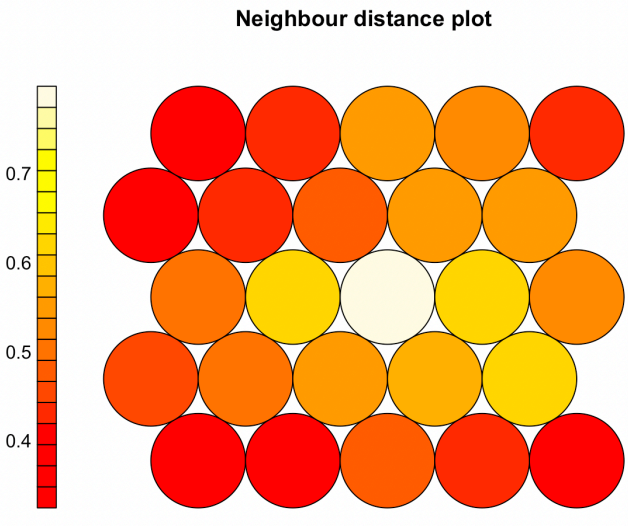

Figure 3: Inter-node distances

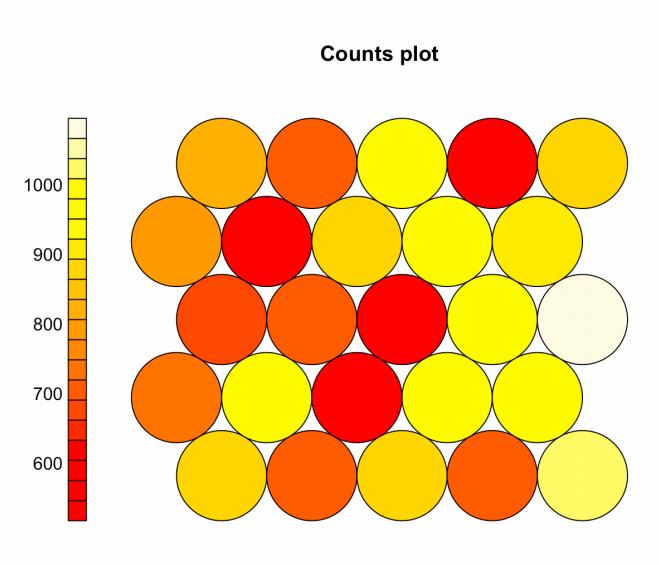

Figure 4: Sample Distribution

4. SOM clustering model was applied to the train dataset to generate cluster information. A 5x5 grid for code book vectors was chosen as it resulted in mid-to-high distances between nodes for many of the nodes considered (Figure 3). Note that mid-to-higher distances indicate dissimilarity of data. The number of iterations was kept at 100, with the learning rate declining linearly from 5% to 1% over the iterations.

5. The distribution of samples per each node is relatively uniform with the exception of several nodes (Figure 4). This first step assigns the observations to map units, i.e. neural nodes that encompass similar data points. The map units are then clustered into three, using a hierarchical clustering algorithm, according to the optimal number of clusters on the elbow-point graph (Figure 5).

6. SVR (Support Vector Regressor) is then fitted to each generated cluster, in turn, and the performance of the resulting SOM-SVR model is compared to that of not using SOM at all (noSOM-SVR).

The reason for choosing SVR as the regressor is due to its high popularity and relative simplicity in various solar

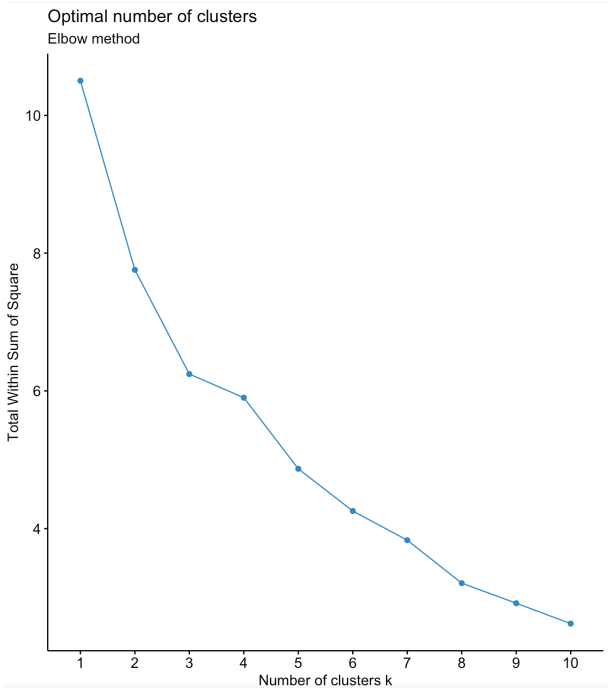

Figure 5: Optimal Number of Clusters

energy forecast-centered researches (Buwei et al. (2018) [10], Yen et al. (2018) [16], Zeng et al. (2012) [17]). To reiterate, the purpose of this paper is to analyze the contribution of SOM as a pre-processor when it's based on many features and not to dive into the applications of various ML models.

The overall methodology is depicted on Figure 6. The implementation was performed in R and the code repository is available here.

C. *Self-Organizing Maps*

The SOM is an unsupervised neural learning algorithm that learns a mapping from a high dimensional continuous feature space onto a lower dimensional discrete space of neurons (Barreto, 2007) [18].

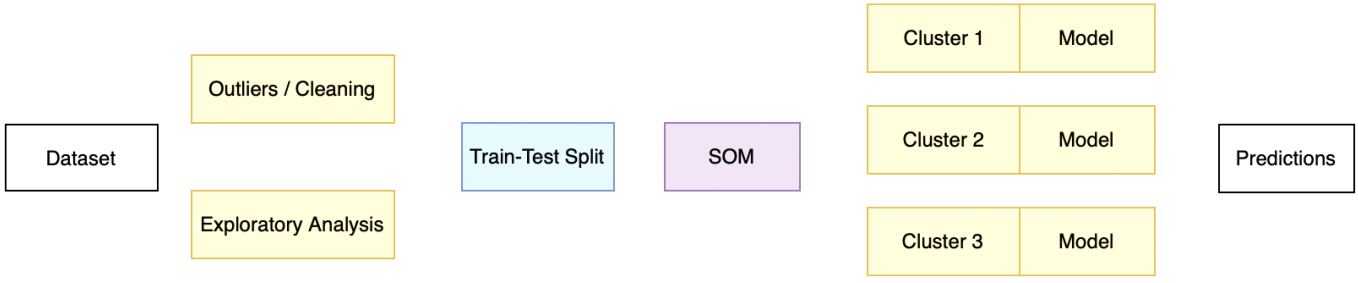

Figure 6: Methodology framework

The neurons are arranged in a two-dimensional rectangular array, as observed on Figures 3 and 4. Multiple iterations are performed until clusters with a high density of neurons are formed.

Since a 5x5 grid was selected for this research (based on the relatively low inter-node distances and a more or less uniform sample distribution), there are 25 neurons in total. They gradually take up the positions on the grid and approximate the shape of the original data as the number of iterations increases (Figure 7).

Neurons that are significantly different from each other have a mid-to-high inter-distance (as observed on Figure 3). Radar plots, like the one on Figure 8 for this research, show the aggregate profile for each neuron, its data points and what makes it different from or similar to the other neurons.

The way SOM algorithm works is as follows, explained in plain English:

1. Neurons are randomly placed in data. In other words, the neurons' weights are randomly initialized.

2. A random data point, i.e. random vector of features, is selected from the training set.

3. The neuron closest to this data point, called the Best Matching Unit (BMU), is found by comparing its weight to the weight of the selected data point.

4. The BMU is moved closer to the data point with each iteration at the learning rate. In other words, it is rewarded by making its weight become more like that of the selected data point.

5. The BMU's neighbors, identified by using a radius around the BMU, are moved closer to that data point, as well. The value of the radius decreases with each iteration.

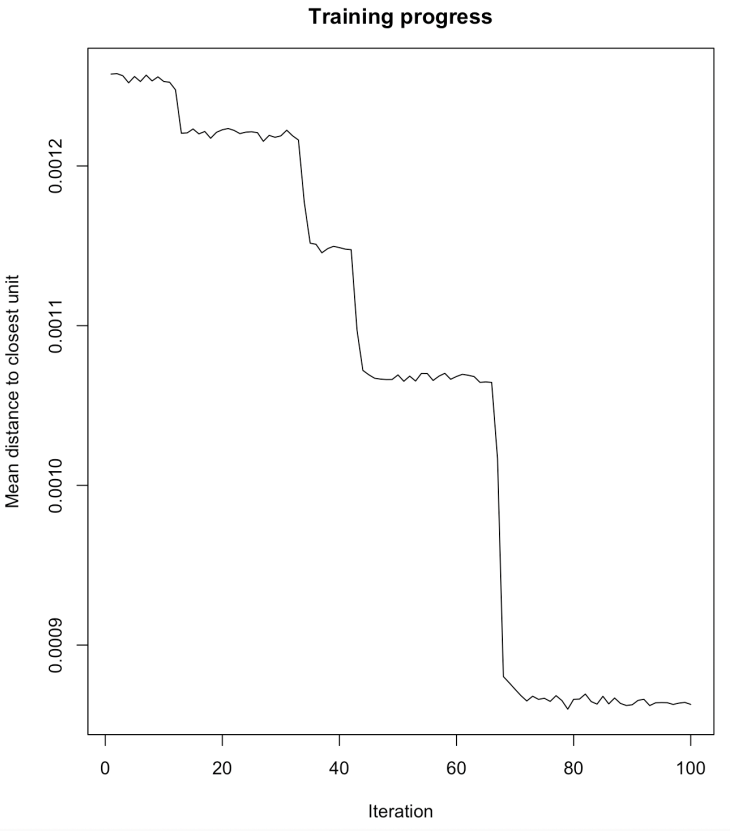

Figure 7: Approximating the data shape

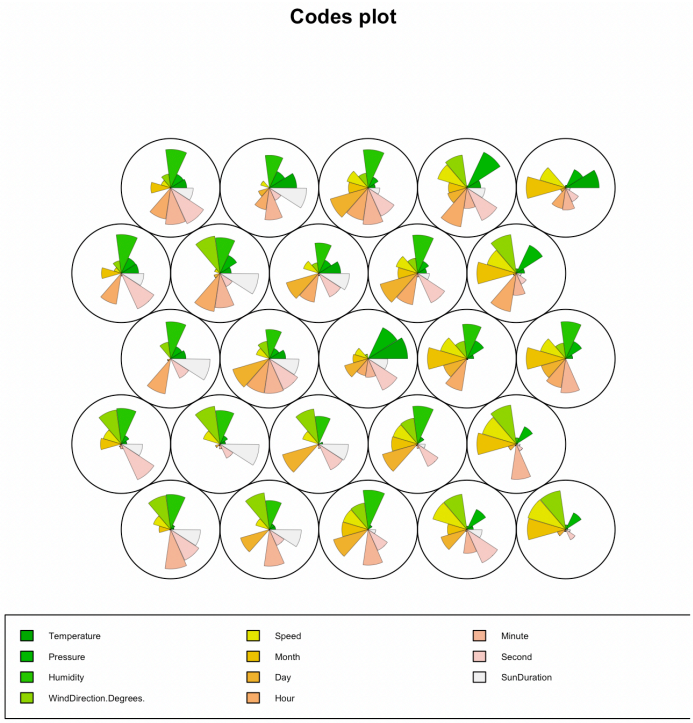

Figure 8: Codes plot

6. The learning rate and radius values are updated and iterations continue a given number of times until neuron positions stabilize.

7. At the end of this process, the result is a list of code book vectors with weights that reduced dimensionality of the original data. Each node's weight vector represents the data samples mapped to that node.

8. Clustering is performed hierarchically, using either Euclidian or Manhattan distance metrics, on the resultant SOM nodes. The cluster labels can then be assigned back to the original dataset by using the computed unit classification attribute for each original data point.

$$d_{k,j} = \sqrt{\prod (x_{ki} - w_{ji})^2}$$

Equation 1

$$w_{ji}(t+1) = w_{ji}(t) + \theta(t)L(t)(x_{ki} - wji(t))$$

Equation 2

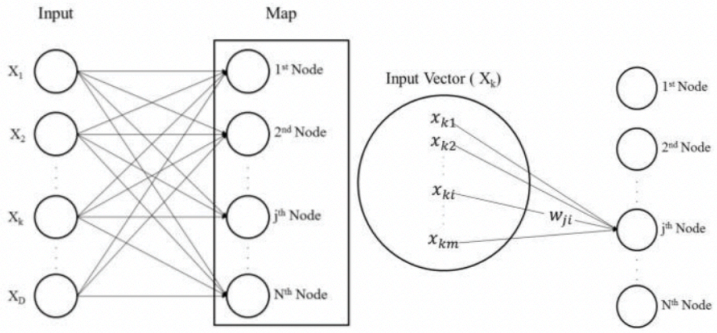

Figure 9: SOM visualized
Source: Nitisanon et al., 2017 [14]

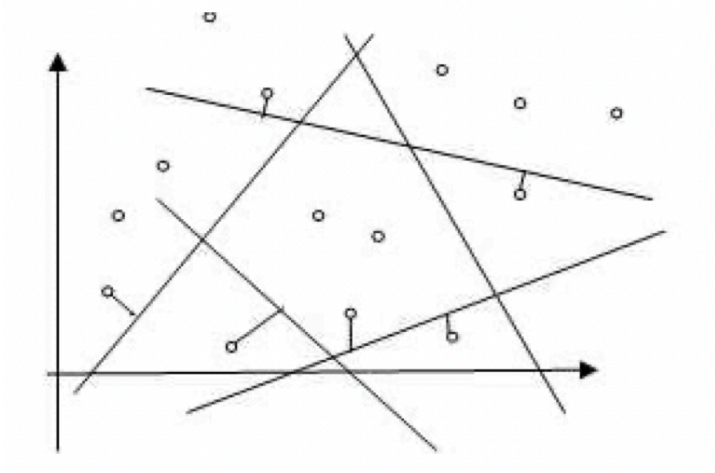

Figure 10: Hyperplane separation
Source: Premalatha et al., 2013 [19]

The visualization of the algorithm is as per Figure 9.

In mathematical terms, the BMU (or the winning node) is mapped to the vector using the equation (1) where x is the input vector and w is the weight vector for a random data group, j.

The input vector x is mapped to the node whose distance d is the smallest (Nitisanon et al., 2017) [14].

With each time step, i.e. iteration, the weights get updated at the learning rate $L(t)$ with an influence rate θ, as per equation (2) above. Hence, each node competes to be the winner.

D. Support Vector Regression

Support Vector Machines (SVMs) are supervised learning models that can be used for classification (SVC) and regression analyses (SVR).

The SVM tries to find a clear separating line (in the case of two dimensions) or hyperplane (in the case of more dimensions) to predict the target variable or classes (Figure 10). The objective is to maximize the distance between points *(x,y)* and the dividing line or hyperplane, at a margin of tolerance.

Equation (3) displays the distance function used for maximization. Note that to maximize this equation, the weight parameter w needs to be minimized.

The hyperplane is represented "… in terms of support vectors, which are training samples that lie outside the boundary …" (Awad et al., 2015) [20]. In other words, the hyperplane is propped up by a list of data points that segregate the space into a number of "data zones". The hyperplane can take on any shape, including the example on Figure 12.

$$\frac{|wx - y + b|}{\sqrt{w^2 + 1}}$$

Equation 3

SVMs utilize different kernel functions to transform linearly inseparable data into data that can be separated with a hyperplane. The said transformation takes place geometrically because kernel functions return the dot product between any two vectors (i.e. an inner product in the feature space).

There are many types of kernel functions used, but the more common ones include:

1. Linear kernel: $k(x, y) = x^T y + c$
2. Polynomial kernel: $k(x, y) = (a\, x^T y + c)^d$
3. Sigmoid kernel: $k(x, y) = tanh(a\, x^T y + c)$
4. Gaussian kernel: $k(x, y) = exp(-\frac{||x - y||^2}{2\sigma^2})$
5. Gaussian radial basis: $k(x, y) = exp(-\gamma\, ||x - y||^2)$

The choice of the kernel depends on the complexity of the data being analyzed. For example, radial basis kernel functions allow to pick out a hypersphere and are generally used when there's no prior knowledge of data, while linear kernels create a simple linear separation. Polynomial kernels are heavily used in image processing, while hyperbolic tangent kernels are used in neural networks.

The loss function for an SVM regression is called the ε-insensitive loss function, ruled by the Lagrange multiplier, as per equation (4). The SVM uses this alongside trying to maximize the distance in equation (3).

$$L_k(y, f(x, w)) = |y - f(x, w) - \varepsilon|, if > \varepsilon, else : 0$$

Equation 4

$$\begin{aligned} \textbf{minimize} \quad & \tfrac{1}{2}\|w\|^2 + C \sum_{i=1}^{\ell} (\xi_i + \xi_i^*) \\ \textbf{subject to} \quad & \begin{cases} y_i - \langle w, x_i \rangle - b & \leq \quad \varepsilon + \xi_i \\ \langle w, x_i \rangle + b - y_i & \leq \quad \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq \quad 0 \end{cases} \end{aligned}$$

Figure 11: SVM regression problem

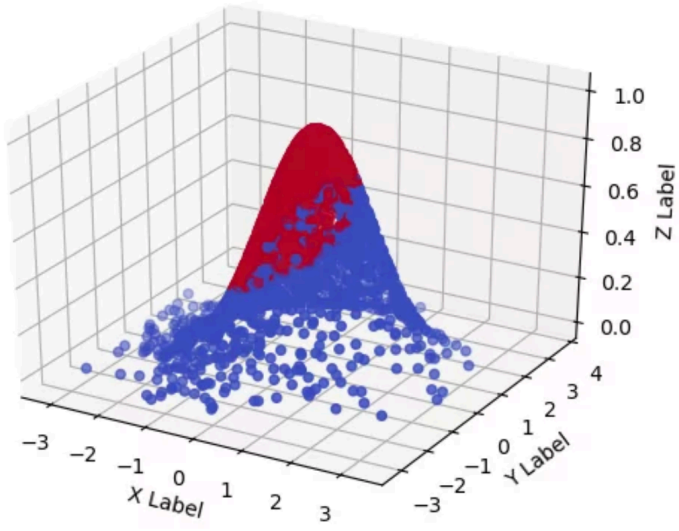

Figure 12: SVM radial basis kernel
Source: link

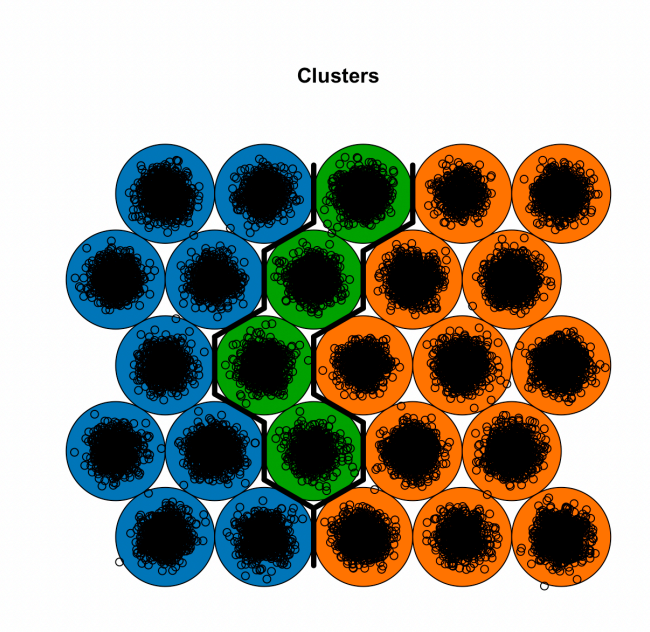

Figure 13: Clustered nodes

Hence, the SVM regression problem is formulated as per Figure 11, where *C* defines the trade-off between exact accuracy and the tolerated margin. The source of the figure is Smola et al. (2003) [21] and the equation was formulated by Vapnik in the 90s.

The kernel function used for this research is the radial basis kernel function, the example of which is depicted on Figure 12.

IV. Results & Discussion

The plot analysis on Figure 2 indicates that there are correlations between the following attributes:

a) *'Sun Duration'* and *'Month'* : high negative correlation, indicating that the time between sunset and sunrise decreases closer to the end of the year.

b) *'Temperature'* and *'Radiation':* high positive correlation, indicating that solar irradiance increases with an increase in temperature.

c) *'Wind Direction'* and *'Pressure':* slightly positive correlation, indicating that the direction winds blow in is positively influenced by the atmospheric pressure of the area.

Although excluding highly correlated non-dependent variables can improve prediction results (i.e. by removing multi-collinearity), none of the variables was dropped due to the following reasons:

- for simplicity and better interpretability of this analysis;
- some non-linear algorithms, including SVRs, can indirectly benefit from the presence of these features due to the amount of important information they carry;
- the purpose of this research is centered around assessing the precision of predictions, rather than focusing on the values of coefficients for each independent variable.

Features with lower correlation were also kept in the model to allow for a better SOM clustering result, whereby the presence of many features can help differentiate the nodes better.

The results of the SOM model on Figure 3 show that the distances between nodes are mostly in the mid-to-high range, with the exception of several nodes that seem to be close to others (highlighted in dark color tone). This margin of error is tolerated for the sake of this analysis at this stage.

The trained SOM model approximated the shape of the original data once the number of iterations exceeded sixty (Figure 7). As displayed on the codes plot in Figure 8, most nodes are different to others in the amount of importance, i.e. magnitude, placed on various attributes. For example, the first node is mostly focused around such attributes as humidity and time in terms of their contribution to the node's value, while the last node displays patterns through wind direction and speed variables, too. There are a few nodes that look similar to each other in terms of the relative magnitude of different variables and this is related to the accepted margin of error, observed previously on Figure 3.

The mapped nodes are then clustered into three, as displayed on Figure 13. This allows to subset the train data into three pieces on which a separate SVR is then trained.

The reason for training the models separately is ruled by the fact that subsetting incoming data into clusters can help leverage better performance through a more accurately fitted model, undisturbed by the noise of other clusters. This is in line with the methodologies used by Gao et al. (2019) [12], Perveen et al. (2019) [13] and Tan et al. (2016) [2], in which the researchers chose to train on separate data subsets to capture the effects of different weather conditions and seasons.

The resultant SOM-SVR RMSE achieved on each cluster and noSOM-SVR RMSE is displayed in Table 1. SOM-SVR helped to achieve an RMSE of 23% on average across all clusters, while noSOM-SVR RMSE is at 34%. The error results per cluster are quite comparable to some of the literature considered. The focal point of these outputs lies in the fact that SOM clustering methodology helped to achieve a better prediction accuracy, on par with the already reviewed literature, as compared to not using SOM at all.

model	no clustering	cluster 1	cluster 2	cluster 3
SOM-SVR	-	14.03%	37.74%	16.70%
noSOM-SVR	33.75%	-	-	-

Table 1: RMSE for considered models

The average error rate for SOM-SVR is dominated by the high RMSE achieved on the second cluster. Naturally, one

cluster	number of observations (train)
1	6,844
2	10,532
3	3,179

Table 2: Number of observations

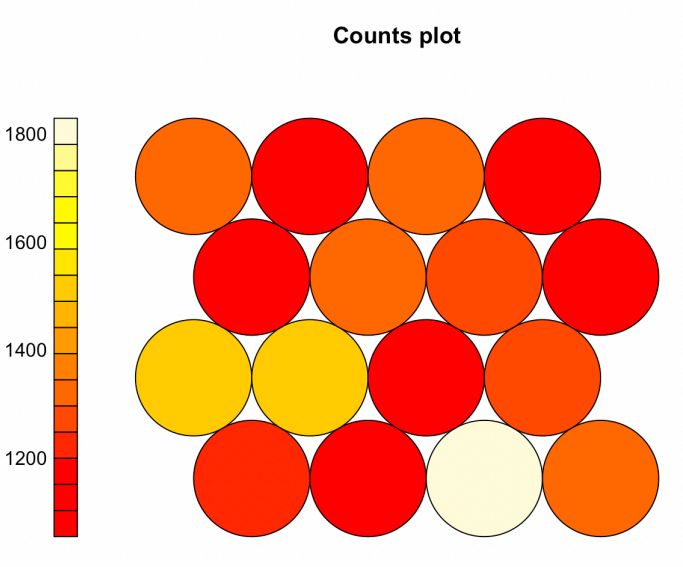

Figure 14: 4x4 sample distribution

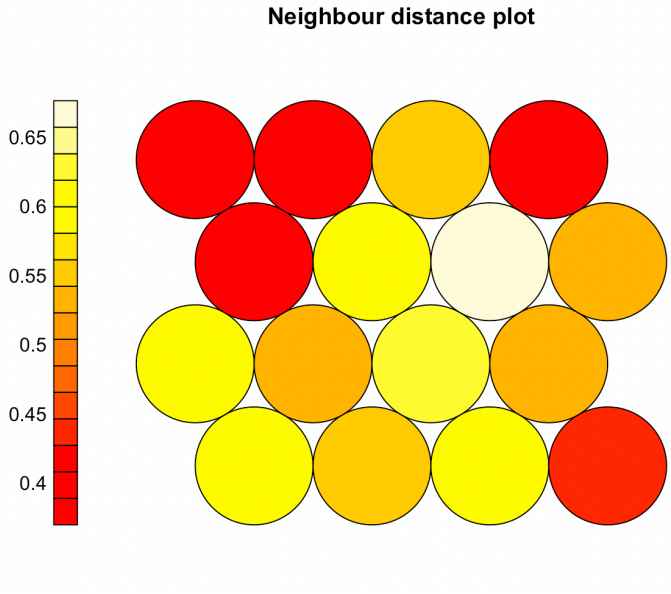

Figure 15: 4x4 inter-node distances

can assume that the error rate is driven by data imbalance, but more formally, it can be due to any of the following:

- the original chosen grid size being too small for the considered data;
- natural imbalance in the dataset representations;
- an overfitting model.

Indeed, comparing the number of observations across all three clusters in Table 2, one can observe that the second cluster has the highest number of observations, despite a higher inaccuracy. This is an indication that the model overfitted on this cluster more so than on others, hence the higher RMSE. Overall, though, the transformed dataset is imbalanced.

Although the results appear promising, the following factors have to be taken into account as disadvantages of SOM-SVR approach:

- The process of mapping observations into neurons is not intuitive and hence, requires extra interpretation despite the fact that SOM preserves the data's topological shape.
- Neuron weights are randomly initialized and this can affect the final results indirectly.
- The number of neurons is fixed and defined in advance by observing the inter-distance and count plots for various-sized grids. In other words, there is no direct way of determining the optimal grid size.
- The algorithm heavily relies on the quality of the dataset and selected features. It also requires nearby observations to behave similarly, which does not always hold true in reality.

As a sensitivity to minimize the margin of error tolerated previously, a lower grid size of 4x4 was also tested. This resulted in a much more uniform distribution of samples per node than before, as depicted on Figure 14. At the same time, the inter-node distance increased for almost all nodes but five (Figure 15).

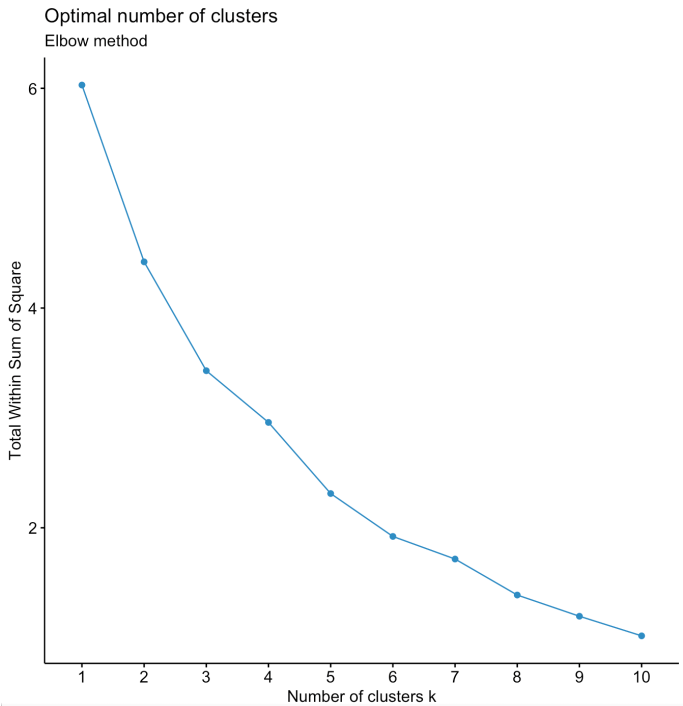

Figure 16: 16 node-clustering

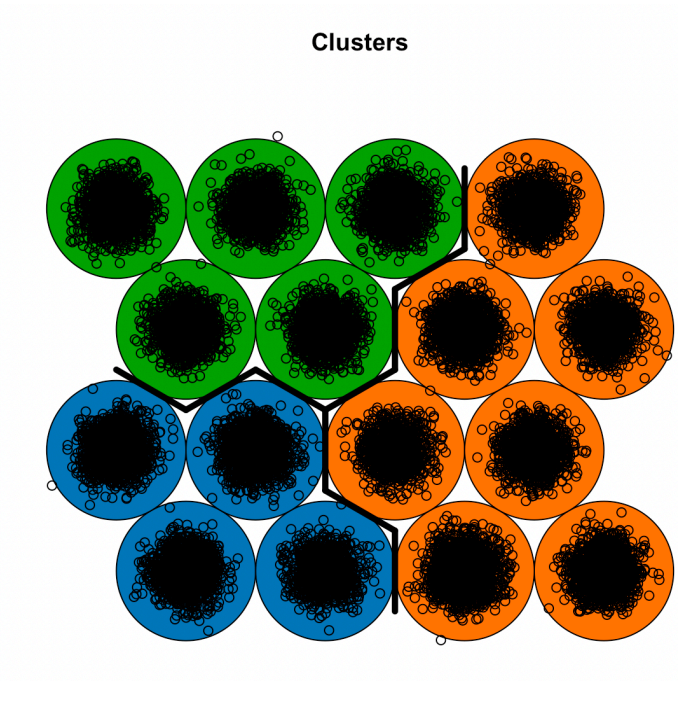

Figure 17: 4x4 node clustering

model	no clustering	cluster 1	cluster 2	cluster 3
SOM-SVR	-	35.41%	37.87%	NaN
noSOM-SVR	33.75%	-	-	-

Table 3: RMSE for considered models on 4x4 grid

In total, sixteen code book vectors, i.e. 16 nodes, were generated as compared to 25 nodes in the original run. The optimal number of clusters remained at three, as before (see Figure 16). Yet, this time the sizes of clusters, in terms of the number of nodes, are more comparable than before (compare Figure 17 with Figure 13).

The test dataset appeared to contain data representations of only two clusters. The resulting RMSEs, however, are higher than before for each cluster (see Table 3).

The reason for higher error rates, despite a seemingly better grid mapping, could lie in the following:

- Although SOM nodes differentiate between sample representations better, the resulting combinations of features in each cluster do not necessarily contribute to a more accurate SVR score. In other words, there could be more noise in the data of each cluster (represented by weakly correlated features which have now become even more dense, resulting in more noise per cluster).
- Each cluster's solar irradiance profile could be explained by a slightly different set of features.

This conclusion naturally leads towards applying a feature selection technique for either the SVR alone or the combined SOM-SVR model, as another sensitivity.

First, only the most correlated features are selected for the SVR regressor based on Figure 2. The variables considered include temperature, wind direction, month and sun duration. However, the obtained results output an RMSE of 39% for cluster one and 32% for cluster two. This is an indication that, although the selected features explain cluster two better than the previously considered model, the same is not true for cluster one. Indeed, by observing the codes plot on Figure 18, one can see that wind direction and month appear less frequently in terms of their contribution to nodes' values in the upper left side, which corresponds to cluster one mapped on Figure 17. On the other hand, features like humidity, minute and second appear to explain cluster one better. Adjusting for this factor brings the error rate for cluster one back to circa 35.8%, which is quite comparable to the error rate achieved on Table 3.

Hence, applying the aforementioned feature selection to a 5x5 grid size, which resulted in a lower average error score originally, provides the following results for cluster two:

- The error rate decreases from 37.74% (Table 1) to 32.59%, which is quite a considerable change.
- This signifies the importance of distinguishing which factors are noise and which ones carry important information per cluster, as opposed to considering the whole dataset.

Yet, this has not fixed the problem of a relatively high error score in cluster two, indicating a need to create a more balanced dataset and/or a less overfitting model.

Despite the already listed disadvantages of SOM, it does house more potential than some of the more traditional

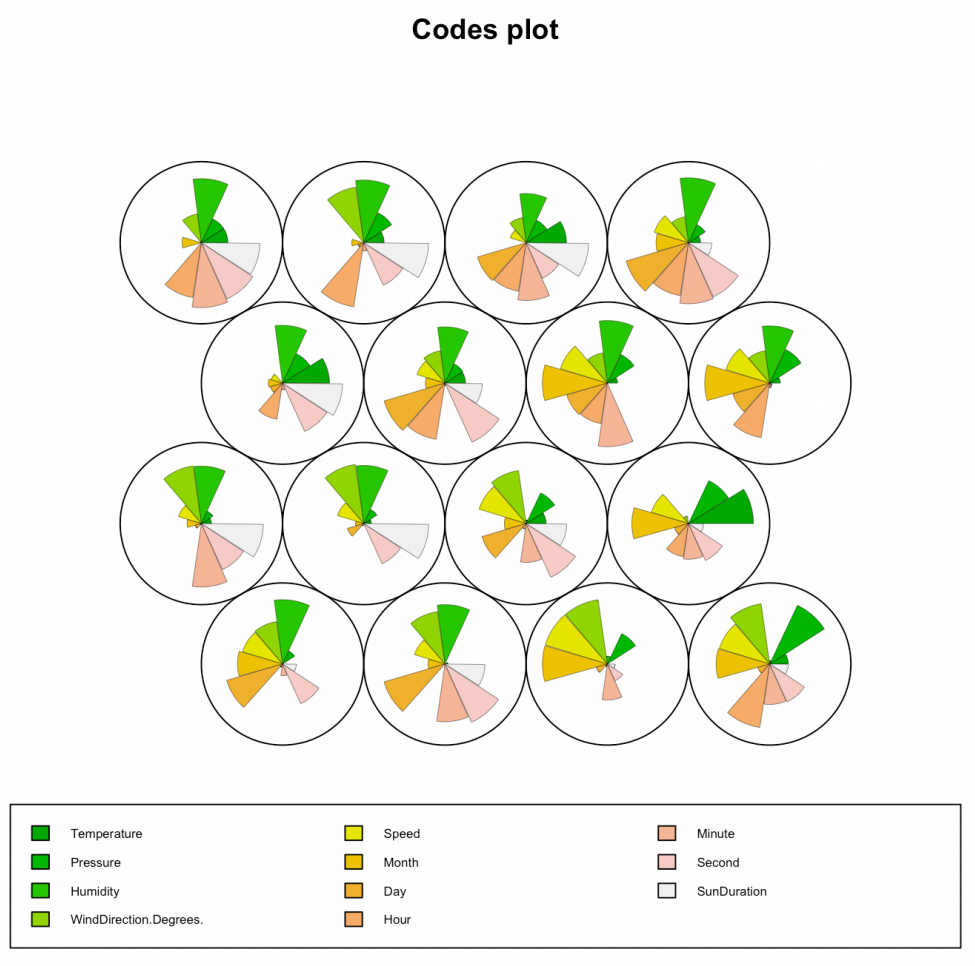

Figure 18: 4x4 codes plot

clustering algorithms, like k-means, for future research. In particular:

• Traditional clustering methods are harder to visualize and hence, interpret compared to SOM, which guarantees the preservation of the topological shape.

• In k-means the nodes do not have a direct relationship with each other and hence, their movement is more free-form. The cases where a similar data point ends up in a separate cluster happen not so rarely. SOM, on the other hand, guarantees that similar samples end up together (provided correctly chosen grid parameters) by "pulling" neighbors to the BMU.

• SOMs are particularly relevant to problems where the set of features can be reduced hierarchically to smaller specialized problems. In this sense, SOMs act like more simplified PCAs, removing the need for additional feature filtering.

• In SOM, nodes "compete" to become the winners to represent a data sample, as opposed to k-means, which is more general.

The shortcomings of the SOM method in this context can be potentially overcome with better SOM-based algorithms, including time adaptive SOM (TASOM) and growing SOM (GSOM), the architectures of which are described below.

The traditional SOM algorithms rely heavily on the data's topological shape which is not necessarily stable. In fact, Hosseini et al. (2003) [22] claim that the learning of the topological shape is the reason for why SOM underperforms under changing conditions. "The speed of change in the environment is not guaranteed to be constant." [22] Hence, the authors propose using adaptive learning parameters.

Instead of using radius values for identifying the neighbors of the BMU, TASOM uses neighborhood functions or sets. In particular, the algorithm takes the following steps [22]:

1. Initialize weight vectors and number of neurons randomly, set the learning rate close to 1.

2. For each neuron, identify its neighborhood set NH. For example, in a two-dimensional grid $NH_{i,j} = set((i-1,j),(i+1,j),(i,j-1),(i,j+1))$.

3. Randomly select an input vector x from the dataset. At this point, TASOM does not yet know anything about the structure of the data it's trying to model.

4. Find the BMU for the selected input vector x by comparing Euclidian distances between weight vectors and x: $argmin||x(n) - w_j(n)||$.

5. Update the size of the neighborhood for the winning neuron at a constant rate β. This is performed by comparing the distances between weight vectors of the winning neuron versus those of its neighbors: $min||w_i(n) - w_j(n)||$.

6. The learning rate parameter is then updated in the neighborhood of the winning neuron n using the following formula:

$$\eta_j(n+1) = \eta_j(n) + \alpha f(||x(n) - w_j(n)||_s / s_f) - \eta_j(n))$$

7. Adjust scaling and continue iterations until node positions stabilize.

Hosseini et al. (2003) [22] showcase that for a stationary environment, the normalized error of a neuron reduces with every iteration until zero, which leads to node weight convergence.

When the environment is non-stationary, however, the normalized errors increase which increases the learning rates in turn. Thus, the network is forced to learn faster until the weights finally converge. The non-stationary environment contributes to the peak observed on graph (c) on Figure 19, whereby the quantization error increases given non-stationarity at time step 15 - yet, that causes the learning rate to increase as well (graph (a)) at that time step and neighbors to be reset as per graph (b). This continues until weights converge.

GSOM was developed specifically to solve the issue of trying to find the optimal map size empirically.

The nodes in GSOM are connected triangularly and the distances between neighbors get actively analyzed and updated. The way this algorithm works is as follows (Zhou et al., 2005) [23]:

1. Each node has error variables attached to it. When the node becomes the BMU, the distance between itself and the data point it's vying to represent gets added to this variable. Hence, nodes can be easily compared as to how closely they match a data sample.

2. After each such allocation the node with the maximum error value - m - is located and its farthest neighbor is determined. The edge between them is split and a new node is inserted halfway.

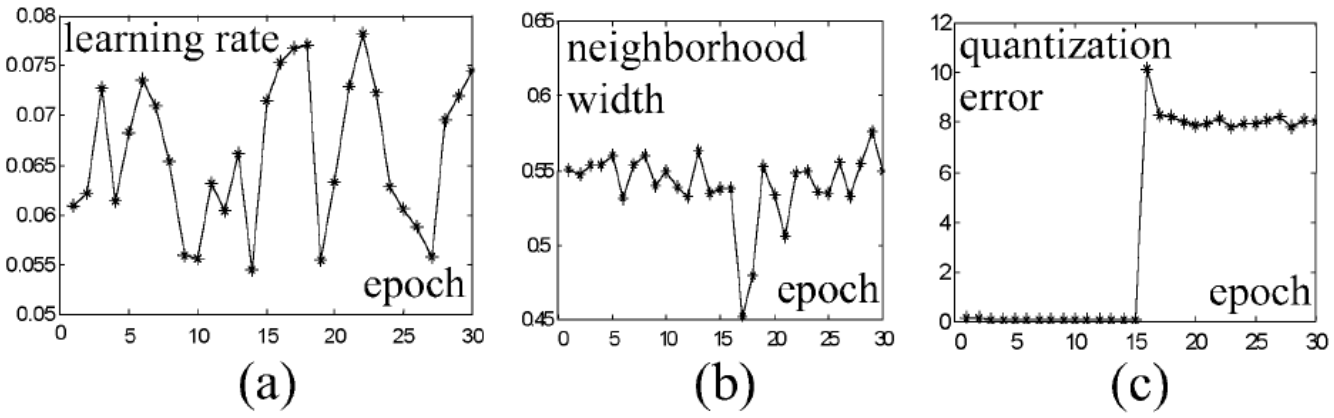

Figure 19: TASOM visualized
Source: Hosseini et al. (2003) [22]

3. This insertion leads to a better mapping mechanism whereby the initially thrown away node - *m* - has a lower chance of becoming the node with the maximum error value in the next iteration.

4. This, seemingly less radical, segregation helps to better visualize the areas with lower probability density.

5. More formally, equation (5) is computed at each iteration for a given number of nodes, *n.* If the delta between the chosen data vector *x* and the selected node exceeds *k,* then the node and its adjacent edges are removed.

$$k_{remove} > n_{distribution} n(1-(1-p^{\frac{1}{n}})^{\frac{1}{n_{distribution}}})$$

Equation 5

where

• p - the probability distribution of an area x,

• $n_{distribution}$ - the input vectors distribution,

• k_{remove} - removed nodes that had high maximum errors.

Hence, the above-mentioned extensions of the traditional SOM can lead to a more optimal learning in non-stationary environments, better adaptation to the dataset and an optimal grid size selection, without the need to confirm it empirically as performed in this research.

V. Conclusion & Future Work

This paper describes the various algorithms in place for solar energy production forecasting and presents an experiment using a less widely tested SOM clustering algorithm based on a larger variety of features than previously considered.

Solar energy forecasting has gained wide attention in the context of the climate change battle, whereby countries and corporations are increasingly taking steps to reshape the current energy mix. However, this transition requires robust predictions of solar radiation profiles to optimally balance supply and demand. Significant advances have been made in the solar energy prediction sphere, starting from applications of models, such as LSTMs, to raw datasets down to the application of various pre-processing techniques to help increase accuracy scores. In particular, the importance of filtering based on weather and climate conditions has been recognized.

Yet, a major shortcoming of such filtering techniques lies in either of the following:

• a rather small amount of features considered for some of the clustering pre-processors,

• categorizing weather conditions into clusters that do not account for the complexity of weather patterns.

Thus, this research attempts to remedy the aforementioned by proposing the usage of SOM clustering tool as a pre-processor to machine learning models that capture the intricacies of individual clusters.

SOM model focuses on reducing the dimensionality of the original dataset by mapping similar data points to nodes in a neural network, using the concept of BMUs. Neighboring data points are moved to the same nodes based on the radius value to the node. Nodes preserve the topological shape of the original data and are easier to visualize. Ideally, the number of nodes selected leaves high distances between nodes, allowing for greater dissimilarity.

The resultant nodes, also called code book vectors, are clustered hierarchically into an optimal number of units. An SVR model is then applied to each cluster separately to generate more accurate predictions.

In this paper, SOM is applied to the 2016 HI-SEAS dataset of meteorological observations, that includes features like humidity, atmospheric pressure and wind speed. The results of applying SOM as a pre-processor to SVRs (SOM-SVR) are compared with those of not applying SOM at all (noSOM-SVR). Comparing the average RMSE of two models suggests that SOM-SVR increases the accuracy of predictions on average (23% error rate) versus noSOM-SVR (34% error rate).

However, the results per cluster are less homogenous. In a 5x5 grid selection, cluster two ends up having a relatively higher error rate. The reason for it is said to be due to an overfitting model and/or data imbalance.

A lower grid size of 4x4 is also tested for sensitivity. However, the outputs are far from satisfactory with error rates increasing to 30%-s for all clusters. This indicates either:

• that the resulting feature combinations in each cluster do not necessarily define the solar radiation profile (i.e. they are just noise), or

• each cluster's solar radiation profile is defined by a different set of features - in particular, by those that contribute the most in their magnitude on the codes plots.

Feature selection is performed as the next step sensitivity. Only the defining features for cluster two are selected for its SVR model. RMSE decreases by circa 6%, which is a considerable change. The remaining error rate could be due to the imbalance of data representations that result in an overfitting model.

All the results are largely at par with the considered literature and point to a big potential of using SOM as a pre-processor in the study of solar energy forecasting, coupled with other pre-processing techniques that can be of relevance (feature selection, encoding etc.).

The advantages of SOM over traditional clustering mechanisms lie in its preservation of the topological data shape, a guaranteed pull-together of neighboring data points and a "compete"-type inherent optimization.

The disadvantages of SOM, however, are no less important to consider:

• SOM is not generative, i.e. it does not learn to recreate the data and hence, new incoming data does not necessarily map well to the nodes obtained during training.

• The optimality of the results depends on the chosen hyper parameters, such as grid size (or number of nodes), learning rate, its decay and the number of iterations. There is no direct way to find the best parameters without trial and error for the traditional SOM algorithm. It does become clear that the algorithm needs an optimization mechanism for a better non-manual selection of hyper-parameters, including grid size, for a more robust computation.

• Random weight initializations can indirectly be influencing the final results.

To address these obvious shortcomings, better SOM-based models are available for use in various fields.

For example, the time-adaptive SOM (TASOM) network has an adaptive learning rate and different neighborhood size for each node. This partially helps solve the hyper-parameter problem and can achieve better mapping results, when the data samples are quite distinct.

The growing SOM (GSOM) starts with a minimum number of nodes and increases the size of the grid based on the computed heuristic. The growth rate can be controlled through the spread factor. This helps solve the problem of finding the optimal grid size parameters.

The oriented and scalable map (OS-Map) is an extension of SOM that narrows the data point down to many best matching units on the map and hence, different types of map orientations can be tested.

In the future, this work can be further improved by considering the following:

• Testing SOM as a pre-processor for LSTM networks to analyze whether such pre-processing techniques hold true in the context of models beyond SVMs.

• Balancing the dataset and cluster representations.

• Using data fusion, i.e. data beyond NWP, for a better clustering opportunity.

• Comparing the usage of SOM in combination with other pre-processing techniques, such as the clear-/hazy-sky filters versus no-SOM models.

• Applying more optimal SOM models to this research, including TASOM, GSOM and/or OS-Map to optimally decide on the number of nodes and learning rates.

References

1. Our World in Data based on bp Statistical Review of World Energy, 2019. Annual Change in Solar Energy Generation, 2019. Retrieved from https://ourworldindata.org/grapher/annual-change-solar?tab=table&time=2011..latest
2. Tan K., Logenthiran T., Woo W., 2016. Forecasting of Wind Energy Generation Using Self-Organizing Maps and Extreme Learning Machines. 2016 IEEE Region 10 Conference (TENCON), pp. 451-454.
3. Lee J., Kim J., Ko W., 2019. Day-Ahead Electric Load Forecasting for the Residential Building with a Small-Size Dataset Based on a Self-Organizing Map and a Stacking Ensemble Learning Method. MDPI, Applied Sciences, 2019, 9, 1231.
4. International Renewable Solar Agency, IRENA. Retrieved from https://www.irena.org
5. Global Solar Atlas. Retrieved from https://globalsolaratlas.info/map
6. Solar Energy Industries Association, SEIA. Retrieved from https://www.seia.org
7. Gensler A., Henze J., Sick B., 2016. Deep Learning for Solar Power Forecasting - An Approach Using Autoencoder and LSTM Neural Networks. 2016 IEEE International Conference on Systems, Man, and Cybernetics, October 9-12, 2016.
8. Chrobak P., Skovajsa J., Zalesak M., 2016. Effect of cloudiness on the production of electricity by photovoltaic panels. MATEC Web of Conferences 76, 2016.
9. Monteiro C., Fernandez-Jimenez A., Ramirez-Rosado I., Munoz-Jimenez A., Lara-Santillan P., 2013. Short-term Forecasting Models for Photovoltaic Plants: Analytical versus Soft-Computing Techniques. Hindi Publishing Corporation, Mathematical Problems in Engineering, Volume 13, Article ID 767284.
10. Buwei W., Jianfeng C., Bo W., Shuanglei F., 2018. A Solar Power Prediction Using Support Vector Machines Based on Multi-source Data Fusion. POWERCON 2018, International Conference on Power System Technology.
11. Aler R., Martin R., Valls J., Galvan I., 2015. A Study of Machine Learning Techniques for Daily Solar Energy Forecasting Using Numerical Weather Models. Springer International Publishing, Studies in Computational Intelligence.
12. Gao M., Li J., Hong F., Long D., 2019. Short-Term Forecasting of Power Production in a Large-Scale Photovoltaic Plant Based on LSTM. MDPI, Applied Sciences.
13. Perveen G., Rizwan M., Goel N., 2019. An ANFIS-based model for solar energy forecasting and its smart grid application. Wiley, Engineering Reports.
14. Nitisanon, S., Hoonchareon, N., 2017. Solar Power Forecast with Weather Classification using Self-Organizing Map. IEEE, 2019.
15. HI-SEAS 2016 Solar Radiation Database, NASA Hackathon. Retrieved from https://www.kaggle.com/dronio/SolarEnergy
16. Yen C., Su K., Yu M., 2018. Solar Power Prediction via Support Vector Machine and Random Forest. E3S Web of Conferences 69(2):01004.
17. Zeng, J., Qiao W., 2013. Short-term solar power prediction using a support vector machine. Elsevier, Volume 52, pp. 118-127.
18. Barreto, W., 2007. Time Series Prediction with the Self-Organizing Map: A Review. Springer, Perspectives of Neural-Symbolic Integration, pp. 135-158.
19. Premalatha M., Lakshmi C., 2013. SVM Trade-off between Maximize the Margin and Minimize the Variables Used for Regression. International Journal of Pure and Applied Mathematics, Volume 87, No. 6, pp. 741-750.
20. Awad M., Khanna R., 2015. Support Vector Regression. Springer, Efficient Learning Machines, pp. 67-80.
21. Smola A., Scholkopf B., 2003. A Tutorial on Support Vector Regression. Retrieved from https://alex.smola.org/papers/2003/SmoSch03b.pdf
22. Hosseini H., Safabakhsh R., 2003. TASOM: A New Time Adaptive Self-Organizing Map. IEEE 2003, Transactions on Systems, Man, and Cybernetics.
23. Zhou J., Fu Y., 2005. Clustering High-Dimensional Data Using Growing SOM. Springer-Verlag Berlin Heidelberg 2005.