Big Data Analytic Paradigms - From PCA to Deep Learning

Barnabas K. Tannahill
Aerospace Electronics and Inform. Tech. Division
Southwest Research Institute
San Antonio, TX, USA
barney.tannahill@gmail.com

Mo Jamshidi, Fellow IEEE, A. Fellow AIAA
Department of Electrical and Computer Engineering
University of Texas at San Antonio
San Antonio, TX, USA
moj@wacong.org

Abstract

Large sets of data (numerical, textural and image) have been accumulating in all aspects of our lives for a long time. Advances in sensor technology, the Internet, social networks, wireless communication, and inexpensive memory have all contributed to an explosion of "Big Data". Big data is created in many ways in today's highly inter-connected world. Social networks, system of systems (SoS : complex interoperable) systems and wireless systems are only some of the platforms creating big data. Recent efforts have developed a promising approach, called "Data Analytics", which uses statistical and computational intelligence (CI) tools such as principal component analysis (PCA), clustering, fuzzy logic, neuro-computing, evolutionary computation, Bayesian networks, etc. to reduce the size of "Big Data" to a manageable size and apply these tools to a) extract information, b) build a knowledge base using the derived data, and c) eventually develop a non-parametric model for the "Big Data". This paper attempts to construct a bridge between SoS and Data Analytic to develop reliable models for such systems. *One of the recent most promising data analytic too is "Deep Learning".* Deep learning is the broad term for the recent development and extensions of neural networks in the machine learning community, which has allowed for state of the art results in speech, image, and natural language processing tasks. Hierarchical learning is an area of research which focuses on learning high order representations from low level data. Learning to recognize objects from images, recognizing words or syllables from audio, or recognizing poses and movement from video are all good examples of modern hierarchical learning research, which is a central focus of the "deep learning" movement in the machine learning and computational statistics community. This paper will give a rather comprehensive look at all the BIG data analytic tools – old and new. Data bases of photovoltaic and wind energy data will all be used here.

Keywords: Data Analytics, Big Data, Solar Energy, Clustering, Micro-Grid, Neural Networks, Fuzzy Inference Systems, Fuzzy C-Means, PCA, Deep learning

[1]This work is partially funded by Lutcher Brown Endowed Chair, ACE Laboratory, University of Texas at San Antonio.

Introduction

System of Systems (SoS) are integrated, independently operating systems working in a cooperative mode to achieve a higher performance. A detailed literature survey on definitions to applications of SoS can be found in recent texts by Jamshidi [1, 2]. Application areas of SoS are vast indeed. They are software systems like the Internet, cloud computing, health care, and cyber-physical systems all the way to such hardware dominated cases like military, energy, transportation, etc. Data analytics and its statistical and intelligent tools including clustering, fuzzy logic, neuro-computing, data mining, pattern recognition and post-processing such as evolutionary computations have their own applications in forecasting, marketing, politics, and all domains of SoS.

A typical example of SoS is the future Smart Grid, destined to replace conventional electric grid. A small-scale version of this SoS is a micro-grid designed to provide electric power to a local community. A Micro-Grid is an aggregation of multiple distributed generators (DGs) such as renewable energy sources, conventional generators, in association with energy storage units which work together as a power supply networked in order to provide both electric power and thermal energy for small communities which may vary from one common building to a smart house or even a set of complicated loads consisting of a mixture of different structures such as buildings, factories, etc [2]. Typically, a micro-grid operates synchronously in parallel with the main grid. However, there are cases in which a Micro-Grid operates in islanded mode, or in a disconnected state [3]. Accurate predictions of received solar power can reduce operating costs by influencing decisions regarding buying or selling power from the main grid or utilizing non-renewable energy generation sources.

The object of this paper is to use big data on solar irradiance as an integral system of the micro-grid SoS to extract relevant information for available solar energy in an attempt to derive an unconventional model.

Section II first describes the micro-grid that will be used as the SoS of interest for this paper. Section III then describes the set of environmental data to be used in this paper as the input to the different data analytics tools. Section IV then discusses the application and effectiveness

of different data analytics tools in the generation of models and relations that could be leveraged to better optimize the operation of the micro-grid. Finally, Section V summarizes the exercises discussed in this paper and draws conclusions based on their findings.

System Model

The micro-grid SoS is shown in Figure 1. Shown here are solar array, battery storage, DC-AC inverter, load and a controller to manage the entire system. Ultimately, we want to forecast received solar power as a model based on real-time environmental measurements to be used in an energy management system [3] to minimize operating costs.

This micro-grid represents a facility scale Cyber-Physical System (CPS) or a SoS consisting of a building with:

- A fixed (or with tracking system) solar photovoltaic system
- A load demand in the form of overall energy consumption, HVAC and lighting, with bi-directional communications (e.g. bi-directional inverter)
- A reconfigurable control and acquisition system (i.e. with open I/O modules, embedded controller for communication, processing and a user-programmable FPGA)
- A local, off-site or cloud-based computing infrastructure for simulation/computational analysis.

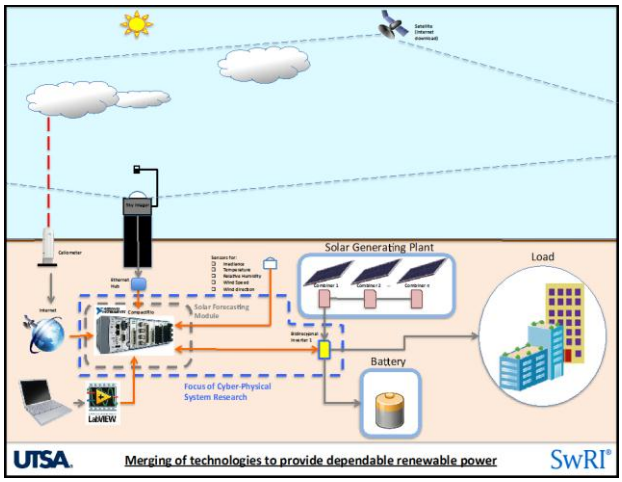

Figure 1. *A PV Forecasting System as a constituent member of a MicroGrid SoS [4]*

PV Data Description

To ensure the Photovoltaic (PV) input data for the different data analysis tools is comprehensive, data from different sources was combined to form the full dataset. This was possible because of the solar research projects occurring in Golden, CO, where the National Renewable Energy Laboratory (NREL) is conducting long term research and data recording to support the growing renewable energy industry.

The first source was the data recorded by the Solar Radiation Research Laboratory (SRRL), which employs over 70 instruments to measure solar conditions and environmental parameters [5]. Also, this data set includes 180° images of the sky that are used to determine current cloud conditions directly.

The second source of data was the SOLPOS data, made available by the Measurement and Instrumentation Data Center (MIDC), which has stations throughout North America to capture information on solar position and available solar energy [6]. Luckily, the MIDC has a station near NREL, so their data can be used in conjunction with the SRRL data.

The final set of data originates from the Iowa Environmental Mesonet (IEM) [7]. Their Automated Surface Observing System (ASOS) station near the Golden, CO site was also included to have current weather data in the set.

Data from the month of October 2012 was combined from the different sources of data. This final set includes one sample for each minute of the month and incorporates measured values for approximately 250 different variables at each data point. The data set was sanitized to only include data points containing valid sensor data prior to the analysis.

Data Analytic of PV Data

In this section, the analysis steps are described, and the results from the different techniques are compared. The goal is to use data analytics tools to generate a useful model from the dataset without needing to resort to parametric analysis and the use of subject matter experts.

Objective Identification

Since the micro-grid would benefit from predicted values of solar irradiance, it was decided that the output of the data analytics should be 60 minute predicted values of three key irradiance parameters (Global Horizontal Irradiance (GHI), Direct Horizontal Irradiance (DHI), and Direct Normal Irradiance (DNI)).

Input Variable Downselection

The input variables were down selected from the full data set to only include cloud levels, humidity, temperature, wind speed, and current irradiance levels. If this exercise was conducted using “Cloud” computing, the number of variables might not need to be down-selected; however, since this effort took place on a single PC, the number of variables was reduced.

Cleanup of the Raw Dataset

Next, the data set was further reduced by removing data points in which GHI, DHI, and DNI levels were very low. The primary reason for this second step was to reduce the amount of time and memory necessary for analysis. Figure2 is a graph containing the measurements of GHI, DHI, and DNI over one day in the cleaned dataset.

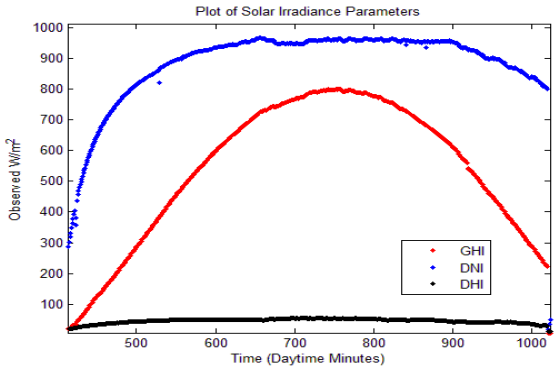

Figure 2. *Three Key Irradiance Parameter Plot for a Clear Day*

Non-Parametric Model Generation Tools

After cleaning took place, the data could be fed into either of the two non-parametric model generating tools, the Fuzzy Inference System Generator and Back-Propagation Neural Network training tools included in the Matlab Fuzzy Logic Toolbox and the Neural Network Toolbox.

Non-Parametric Model Generation Tools

The Fuzzy Logic Toolbox function used in this exercise, *genfis3* uses Fuzzy C-Means clustering to cluster values for each variable which produces fuzzy membership functions for each of the variables in the input matrix and output matrix. It then determines the rules necessary to map each of the fuzzy inputs to the outputs to best match the training data set. These membership functions and rules can be viewed using the Matlab FIS GUI tools such as *ruleview*. When run with default parameters, the *genfis3* function ran significantly slower and performed worse than Matlab's Neural Network fitting function.

Note in 3, differences in the observed and predicted data points generally corresponds to the presence of clouds or other anomalies that could not be predicted an hour in advance using the variables input to the function.

Neural Network Fitting Tool

The second model generating method was the Matlab Neural Network Training tool. By default, this tool uses the Levenberg-Marquardt back propagation method to train the network to minimize its mean squared error performance. Results from training one sample set are shown in Figure 4-6.

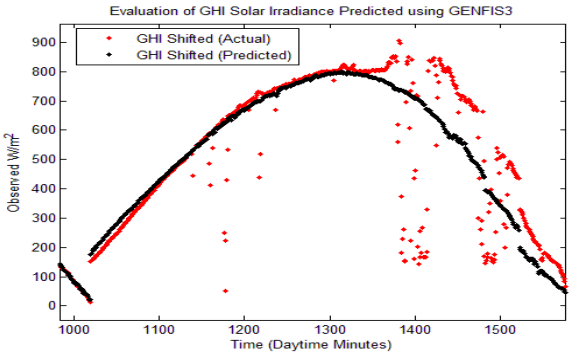

Figure 3. *Data Generated Using GENFIS3 Based on 13 Input Variables*

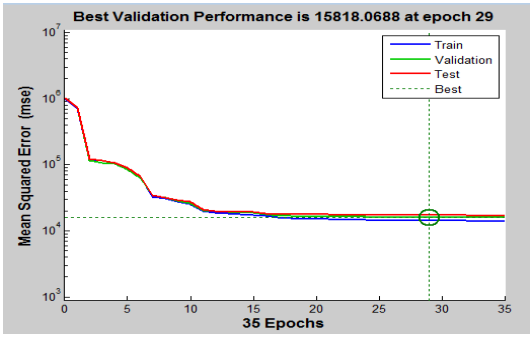

Figure 4. *Back-propagation Performance Curve*

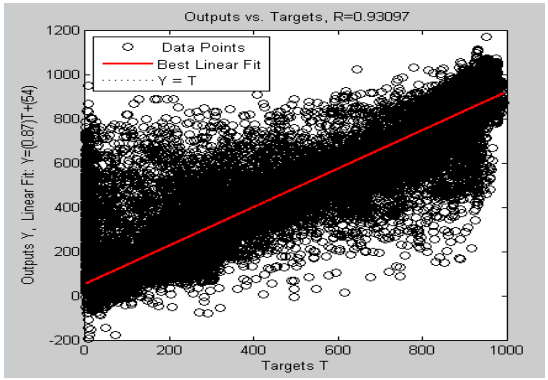

Figure 5. *Post Training Network Regression Performance*

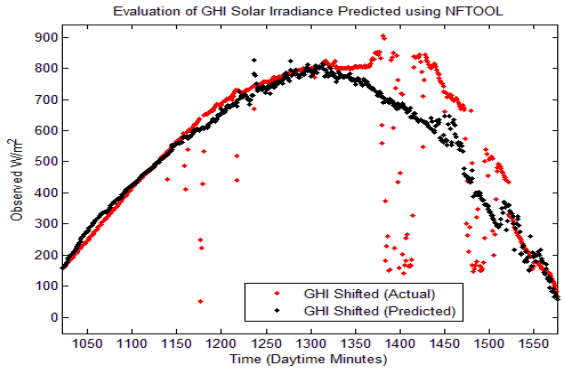

Figure 6. *Data Generated Using NFTOOL Based on 13 Input Variables and 10 Hidden Neurons*

Additional Pre-Processing Discussion

Once the initial performance of these two tools was evaluated, it was decided that further effort should go into including a greater number of original input variables and including additional preprocessed parameters in the training data in an effort to enhance the performance of the derived model. This effort took three paths, the calculation of nonlinear input parameters, the inclusion of a greater number of input parameters, and the reduction of input data dimension when necessary in order to support the execution requirements of the two model generation tools.

Nonlinear Data Set Expansion

In an effort to derive additional useful input parameters from the existing dataset, each variable included in the dataset generated several additional variables based on nonlinear functions and past values of the variable itself. Inclusion of these parameters in the training data set greatly improved the performance of the training tools. A subject matter expert would be useful in this step to identify useful derived parameters such as these to add to the training data set.

Large Data Sets and Principal Component Analysis

Models were generated using different sets of input variables to try to assess the impact of incorporating increasing numbers of variables in the training data set. In general, the trained model performed better when more variables were included in the training data set; however, as the number of variables increased, the training execution time became excessive and out-of-memory errors occurred when the data sets became too large.

In order to combat this issue, the dimension of the training data set was reduced to a manageable size using PCA. PCA can be used to compress the information from a large number of variables to a smaller dataset while minimizing the information lost during this process [8, 9]. This can be performed directly on a dataset using the *princomp* function in Matlab.

The columns of the SCORE matrix returned by the *princomp* function represent the columns of the input data transformed to place the majority of the information in the data set in the first few principal components. The information distribution among the principal components is illustrated in Figure . The higher eigenvalues represent the principal components with the most information. Incorporating principal components past 10 provides minimal additional information.

In this application PCA was primarily useful because it allowed the reduction of very high dimension data sets to smaller, more manageable data sets that could be used as training inputs to the model generation tools.

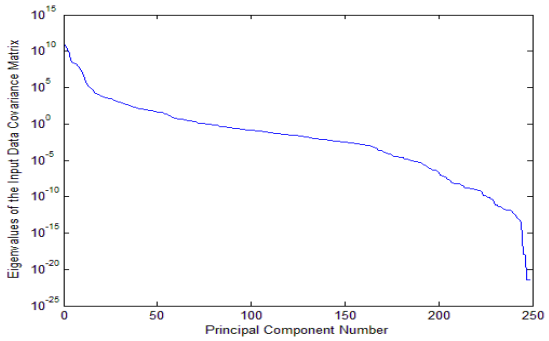

Figure 7. *Principal Component Information Graph*

Figure 8 below shows the quality of information recovery if transforming back to the original basis using only information from the first 50 principal components.

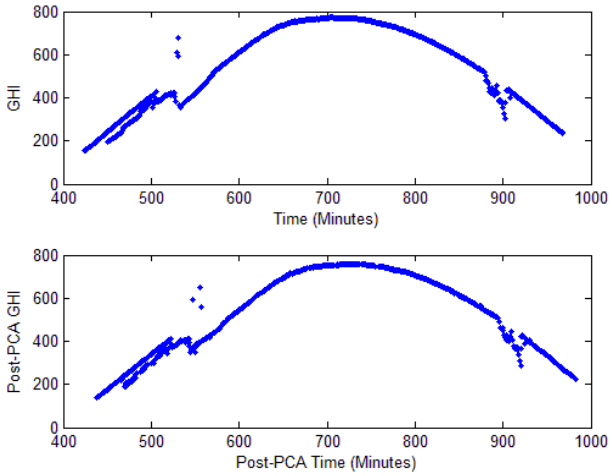

Figure 8. *Data Recovery Demonstration Using First 50 Principal Components*

Results

In order to generate the best non-parametric model possible, different combinations of data inputs to the *GENFIS3* and *NFTOOL* were considered. Different implementations of the options discussed above were evaluated during this analysis.

The best performing *NFTOOL* generated model used all 244 original variables, which were then expanded the dimension to 1945 using the nonlinear variable derivation calculations. Next, the dimension of the data was shrunk to 150 so that the training function had sufficient memory to train the network. The resulting network was the best of all the generated models.

The best performing *GENFIS3* generated model evaluated during this effort used the same input data set as mentioned in the paragraph above with the exception that the dimension was shrunk down to 50 using PCA. It was observed during this effort that effectiveness of the

GENFIS3 tool appears to be less tolerant of high dimension training data sets than the *NFTOOL*.

Table 1 and 2 describe the performance of the models generated using these tools. Note that these performance numbers should be compared qualitatively since the different input parameter configurations can yield different numbers of training data points.

A sub-optimal predictor was constructed in order to show its performance relative to that of the non-parametric models. This predictor was based on the average GHI, DHI, and DNI values for each time bin in the data set. Table 3 shows the improvement of the non-parametric models when

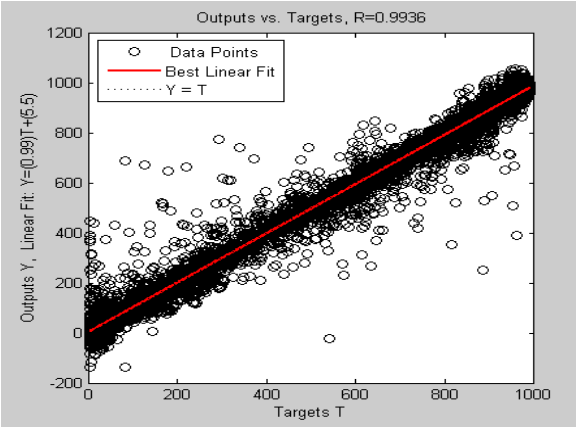

Figure 9. *Best Neural Network Linear Regression Performance*

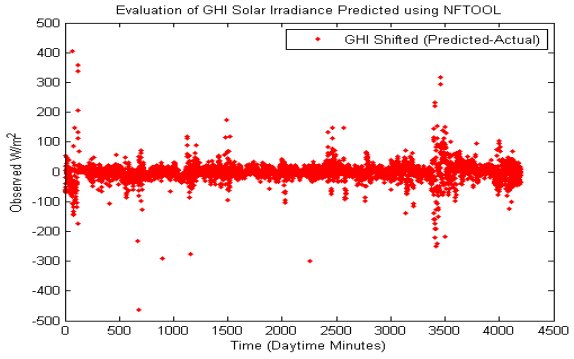

Figure 20. *Best Neural Network GHI Error*

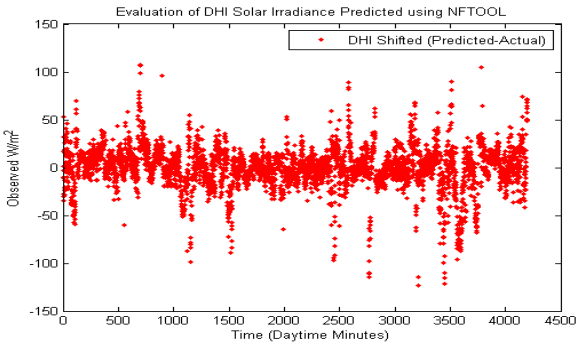

Figure 31. *Best Neural Network DHI Error*

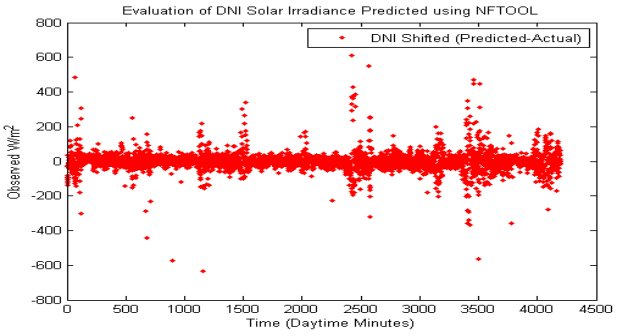

Figure 42. *Best Neural Network DNI Error*

Table 1. *Performance Comparison of the Generated Non-Parametric Models (GENFIS3)*

Model Type	Input Params	Add Nonlin. Params	PCA?	Final Dim	MSE GHI	MSE DNI	MSE GHI	R
FIS	13	N	N	13	8.81E+03	4.98E+04	2.03E+03	0.906
FIS	32	N	N	32	8.68E+03	4.90E+04	1.99E+03	0.907
FIS	32	Y	Y	10	6.64E+03	3.58E+04	1.70E+03	0.938
FIS	32	Y	Y	50	6.35E+03	3.12E+04	1.85E+03	0.945
FIS	13	Y	N	97	4.13E+03	2.31E+04	1.14E+03	0.961
FIS	244	N	Y	50	1.06E+04	5.00E+04	2.40E+03	0.903
FIS	244	Y	Y	50	2.96E+03	2.08E+04	1.05E+03	0.967
FIS	244	Y	Y	150	1.82E+04	5.50E+04	3.35E+03	0.901

Table 2. *Performance Comparison of the Generated Non-Parametric Models (NN10)*

Model Type	Input Params	Add Nonlin. Params	PCA?	Final Dim	MSE GHI	MSE DNI	MSE GHI	R
NN10	13	N	N	13	7.01E+03	3.41E+04	1.65E+03	0.934
NN10	32	N	N	32	6.94E+03	3.40E+04	1.96E+03	0.934
NN10	32	Y	Y	249	1.22E+03	3.83E+03	5.28E+02	0.992
NN10	32	Y	Y	10	4.17E+03	1.74E+04	1.14E+03	0.969
NN10	32	Y	Y	50	2.21E+03	7.14E+03	9.74E+02	0.986
NN10	13	Y	N	97	1.72E+03	4.57E+03	6.58E+02	0.991
NN10	32	Y	N	249	1.20E+03	3.52E+03	4.20E+02	0.993
NN10	244	N	Y	50	5.26E+03	1.63E+04	1.57E+03	0.966
NN10	244	Y	Y	50	1.46E+03	4.91E+03	5.44E+02	0.991
NN10	244	Y	Y	150	9.97E+02	2.95E+03	4.57E+02	0.994

compared to this sub-optimal predictor, named "Time Bin Mean" in the table below.

Table 3. *Performance of Best Non-Parametric to Mean Time Bin Sub-Optimal Predictor*

	Model Type		
Parameter	Time Bin Mean	Best FIS	Best NN10
MSE GHI	1.17E+04	2.96E+03	9.97E+02
% MSE GHI Improvement	0%	294%	1070%
MSE DNI	8.44E+04	2.08E+04	2.95E+03
% MSE DNI Improvement	0%	306%	2765%
MSE GHI	4.51E+03	1.05E+03	4.57E+02
% MSE GHI Improvement	0%	331%	886%
R	8.39E-01	9.67E-01	9.94E-01
% R Improvement	100%	115%	119%

During this analysis, the aspect of the scalability of the *GENFIS3* and *NFTOOL* tools was evaluated. The model

generation time for *NFTOOL* was always shorter than *GENFIS3* for the same data sets. The relationship of *NFTOOL* execution time to dataset length and dimension was generally linear for the test cases evaluated. The relationship of *GENFIS3* execution time to dataset length was also linear; however, its relationship between dataset dimension and execution time was exponential. This is shown in Figure 5.

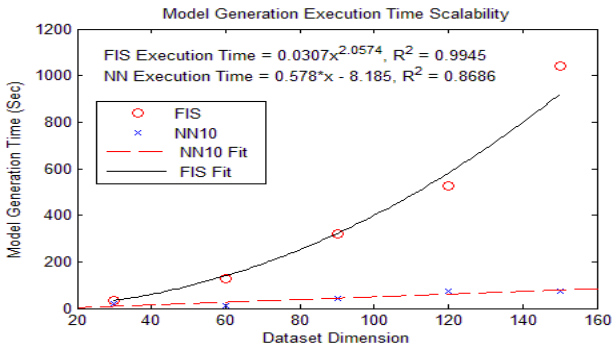

Figure 53. *Model Generation Execution Time Relationship with Dataset Dimension*

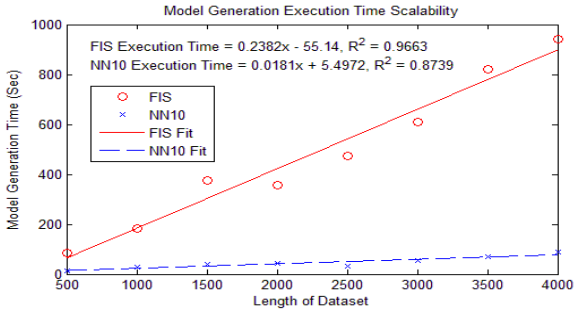

Figure14. *Model Generation Execution Time Relationship with Dataset Length*

Conclusion

This paper presents a high level look at some of the tools available in the Matlab toolset that enable the user to extract information from "Big Data" sources in order to draw useful conclusions. As described in Section 2, the specific application discussed in this paper is the prediction of the amount of solar power generated by a micro-grid. Section 3 then discusses the data that was gathered to support this exercise. Section 4 discusses the steps and techniques considered while trying to generate the best solar irradiance prediction model. Techniques discussed included dataset sanitation, training input parameter selection, model generation via Fuzzy C-Means Clustering and Rule Inference (*GENFIS3*), Neural Network training using back propagation (*NFTOOL*), Pre-Processing nonlinear variables to add to the training data set, and the use of PCA to reduce the dimension of the training data while maximizing the information retained in the data set.

It was observed in the results presented in Section 5 that the best model predicting solar irradiance was one utilizing the maximum number of original and preprocessed variables, which was then reduced to a manageable dimension using PCA prior to use in training the model. The results in this section also showed that the non-parametric model generation methods discussed in this paper performed significantly better than a sub-optimal predictor. Finally, the results describing the model generation times for the two techniques showed that *NFTOOL* provides significantly better training times, especially when the dimension of the dataset is high.

Future work on this topic is planned to address deep learning in place of ANN and using Genetic Programming to optimally reduce the dimension of the dataset, the use of cloud computing to generate models for larger data sets, and the design and evaluation of a controller to buy or sell money from the grid based on demand and predictions of received solar energy.

References

[1] M. Jamshidi (ed.), Systems of Systems Engineering – Principles and Applications, CRC – Taylor & Francis Publishers, London, UK, 2008.

[2] M. Jamshidi (ed.), System of Systems Engineering – Innovations for the 21st Century, John Wiley & Sons, Publishers, New York, NY, 2009.

[3] Y. S. Manjili, A. Rajaee, M. Jamshidi, B. Kelley, "Fuzzy Control of Electricity Storage Unit for Energy Management of Micro-Grids1", World Automation Congress (WAC), Mexico, 2012.

[4] Texas Sustainable Energy Research Institute, Proposal to National Science Foundation on PV Forecasting and Energy Management, M. Jamshidi PI, February 2013, San Antonio, Texas.

[5] National Renewable Energy Laboratory. (2012). Current Irradiance and Meteorological Conditions Data Set. Retrieved from http://www.nrel.gov/midc/srrl_bms/

[6] National Renewable Energy Laboratory. (2012). SOLPOS Data Set. Retrieved from http://www.nrel.gov/midc/solpos/ solpos.html

[7] Iowa Environmental Mesonet. (2012). Automated Surface Observing System Data Set. Retrieved from http://mesonet.agron.iastate.edu/ASOS/

[8] L. I. Smith. (2002, Feb. 26). "A Tutorial on Principal Components Analysis", [Online]. Available: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf