Homework

Quanxi Guo

23/09/2024

Background

Solar irradiance and ambient temperature have always been two important factors affecting the power generation of solar photovoltaic panels. For my research field, smart microgrid energy management, the renewable energy power generation system is very important to the entire energy management system, so I plan to use SINDy (Sparse Identification of Nonlinear Dynamics) to model the power generation of solar photovoltaic panels with solar irradiance and ambient temperature, and then accurately predict the power generation of solar photovoltaic panels.

Data Source

This homework will use two sets of public data. The first set of data comes from the Australian DKA Solar Center. The data includes solar irradiance, temperature and power, etc. The data website is: Home Page | DKA Solar Centre [1].

Another dataset is derived from measurements of power load and indoor parameters in an office building in Bangkok [2], Thailand. The entire dataset is based on data from 2018-2019 for 7 floors where data was recorded. Each area has a lighting load and a plug load measurement. The measurements include power consumption (kW) of a single air conditioning (AC) unit, lighting load, and plug load, as well as environmental sensor data including indoor temperature

(°C), relative humidity (%), and ambient light (lux), with a granularity of 1440 data points per day. In this homework, data from April 17, 2019, to April 21, 2019, were used, including power consumption, indoor temperature, indoor humidity, and outdoor temperature (obtained from NASA [3])

Code

All codes for this homework are written in Python and hosted online at QuanxiGuo/Modeling-from-measurements-homework (github.com)

PCA (Principal Component Analysis)

Principal Component Analysis (PCA) is a widely used technique for dimensionality reduction, aimed at extracting the most important features from data. There are two common methods to perform PCA: Eigenvalue Decomposition (EVD) and Singular Value Decomposition (SVD). When using EVD, the covariance matrix of the target matrix must first be computed. However, if both the number of samples and the number of features are large, the computational cost can be quite high. On the other hand, using SVD allows us to obtain the principal components without needing to compute the covariance matrix. This approach is particularly efficient when dealing with large sample sizes.

The brief concept of SVD is as follows. For any matrix A there is always a singular value decomposition:

$$\boldsymbol{A = USV^T}$$

If A is an M×N matrix, then the resulting U is an M×M square matrix, and the orthogonal vectors in U are called left singular vectors. S is an M×N matrix, and all elements of S except the diagonal are 0, and the elements on the

diagonal are called singular values. V^T is the transposed matrix of V, which is a N×N matrix, and the orthogonal vectors in it are called right singular value vectors. In PCA, the right singular vectors represent the directions of the principal components, which define the axes of the new feature space. The magnitude of each singular value indicates how much variance is explained by its corresponding principal component. Components with larger singular values explain more of the data's variance and are prioritized during selection. This process allows PCA to efficiently identify and retain the most significant components using SVD.

In the original data of [1], there are nine features related to the output power of solar photovoltaic panels. The number of features is obviously too many. If we want to build a SINDy function library for these nine features, the work is undoubtedly huge. So, I hope to reduce the dimensionality of the original data by PCA.

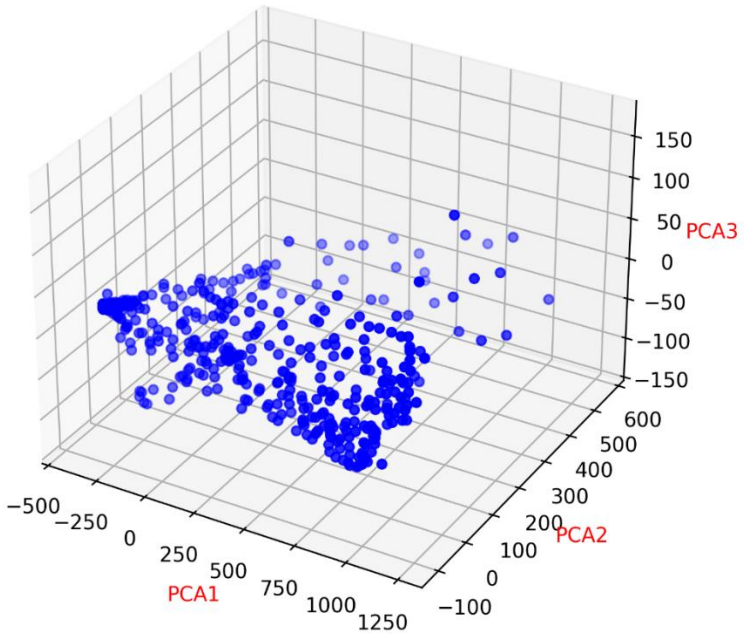

Figure 1. Three-dimensional PCA data point distribution (3%)

As shown in Figure 1, PCA is used to reduce the original nine eigenvalues to

three, which greatly reduces the amount of data processing. Due to the large amount of data, only 3% of the data points are shown in the figure. It should be noted that the new data obtained after PCA processing has no actual physical meaning, because the main goal of PCA is to reduce the dimension of the data by linearly combining the original features, rather than retaining the original physical meaning.

SINDy

SINDy is an algorithm that automatically constructs a concise dynamic model from data, which is like performing a linear fit on the model. However, unlike ordinary linear fitting problems, SINDy screens the terms in the function library through sparse regression equations to find concise equations that describe the dynamics of the system.

When using SINDy to process data, we first create a function library Θ. For example, for a data with three features x, y and z,

$$\Theta = [x^3, y^3, z^3, x^2y, y^2x, z^2x, \cdots\cdots, xyz]$$

Next, we will use the sparse regression equation ξ to filter the items in the function library theta. This can be achieved through the L0, L1 or L2 norm, or we can set a threshold below which values will not be considered.

$$A = \xi\Theta$$

Where A is the formula of the final model. Based on this formula, we can also know the physical laws of the model. It is worth noting that if the model data is processed by PCA, the terms in the final formula will have no specific physical meaning, so the real physical laws of the model cannot be known.

1. Lasso regression

Lasso regression is a commonly used regression method that effectively implements feature selection and model simplification by introducing L1 regularization, thereby reducing the complexity of the model. I used this sparse regression method to screen a library consisting of PCA1, PCA2, and PCA3 (the regularization parameter is 0.1), aiming to derive a formula for the output power of solar photovoltaic panels based on these three principal components. In Python, this can be easily implemented using the `sklearn.linear_model` module in the Scikit-learn library.

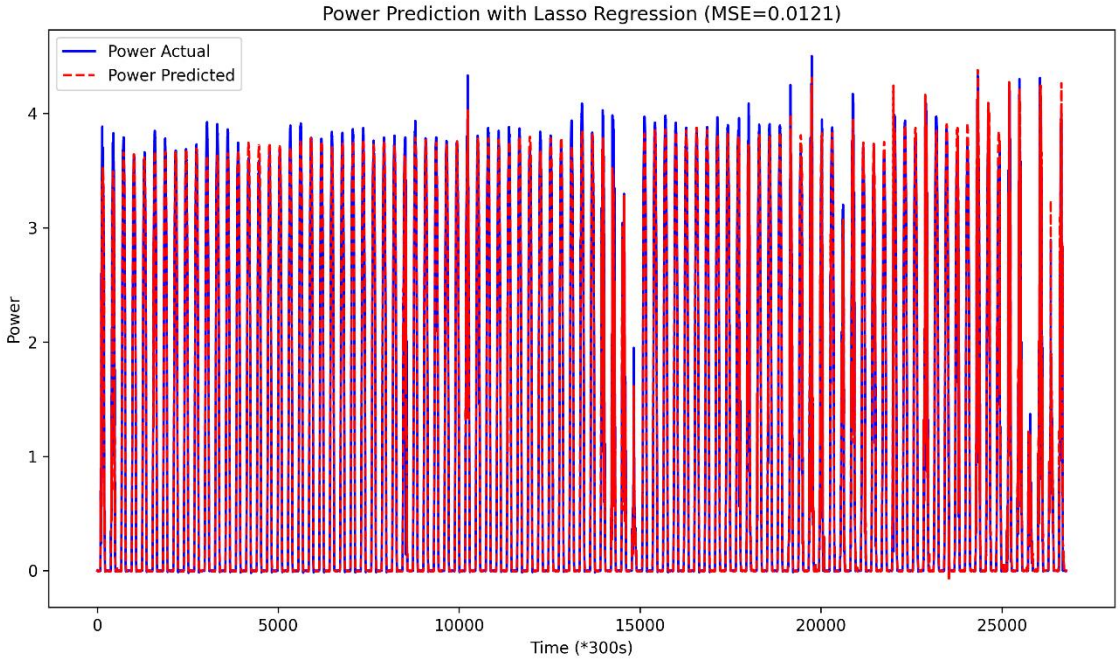

Figure 2. Comparison of true power and model predicted power (PCA_Lasso)

The mean square error (MSE) between the predicted model power and the true power is 0.0121, indicating that the predicted model has good tracking performance for the real data. At the same time, it can be seen from Figure 2 that the performance of the model for peak prediction is not very ideal, which may be due to the error caused by the PCA processing of the data.

The model polynomial and its coefficients are expressed in matrix form as follows (filter items with coefficient greater than 1×10^{-4}):

$$\begin{bmatrix} 2.65 \times 10^{-3} \\ 7.49 \times 10^{-4} \\ -4.18 \times 10^{-3} \end{bmatrix} \times \begin{bmatrix} PCA\ 1 \\ PCA\ 2 \\ PCA\ 3 \end{bmatrix}$$

So, the final power formula is:

$$Power = 2.65 \times 10^{-3} PCA\ 1 + 7.49 \times 10^{-4} PCA\ 2$$
$$- 4.18 \times 10^{-3} PCA\ 3$$

This formula shows the functional relationship between the solar photovoltaic panel power and the three principal components (PCA1, PCA2 and PCA3).

2. Ridge regression

Ridge regression is also a linear regression method, which mainly screens features by introducing L2 regularization. I used this regression method to process a new set of data. Unlike the previous PCA data, this set of data only contains solar irradiance, temperature and power, and the number of sample points has been reduced from 26000 to 6000. I want to use the data to establish a functional relationship between power and solar irradiance and temperature.

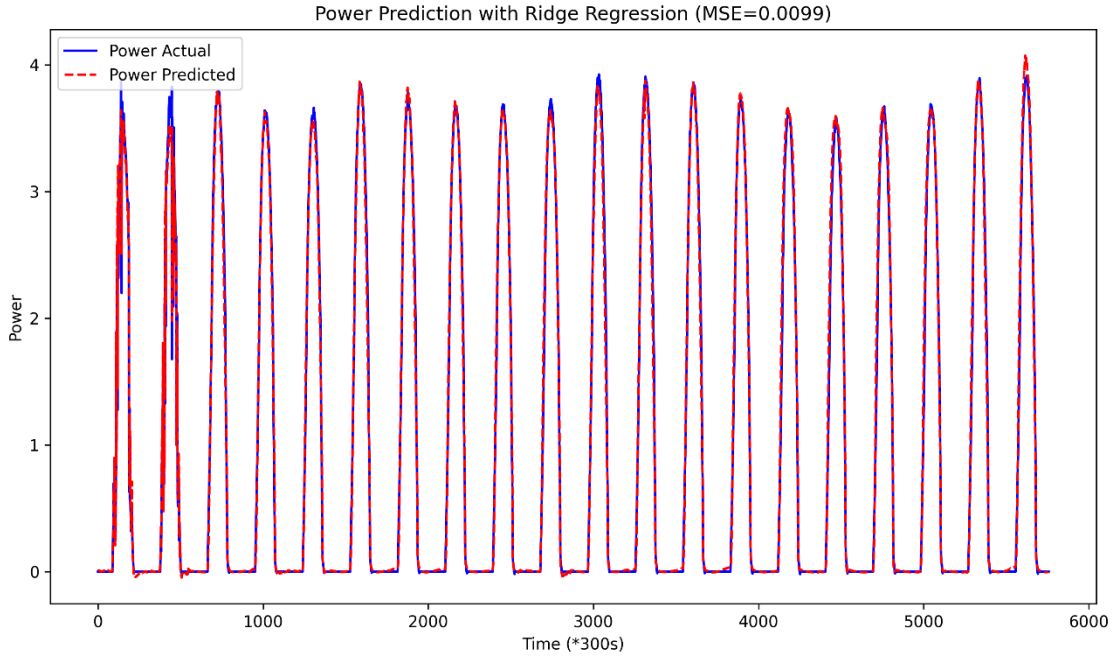

Figure 3. Comparison of true power and model predicted power (Ridge)

When the number of sampling points is reduced, the tracking performance of the prediction model can be more clearly seen, and the prediction accuracy is also improved (MSE=0.0099).

The model polynomial and its coefficients are expressed in matrix form as follows (filter items with coefficient greater than 1×10^{-5}):

$$\begin{bmatrix} -9.57 \times 10^{-3} \\ 3.40 \times 10^{-4} \\ 1.02 \times 10^{-3} \\ 9.26 \times 10^{-5} \\ -3.08 \times 10^{-5} \end{bmatrix} \times \begin{bmatrix} T \\ R \\ T^2 \\ TR \\ T^3 \end{bmatrix}$$

So, the final power formula is:

$$P = -9.57 \times 10^{-3}T + 3.40 \times 10^{-4}R + 1.02 \times 10^{-3}T^2$$

$$+ 9.26 \times 10^{-5}TR - 3.08 \times 10^{-5}T^3$$

Where T is temperature, R is irradiance, and P is power. This also gives the functional relationship between power P and temperature T and irradiance R. This formula has physical significance and explains the extent to which the power of solar photovoltaic panels is affected by temperature and irradiance.

In the above data modeling process, I found that I input all the data into the model. The model only fits the known data and outputs the formula. This has limited application in real life because in most cases, we cannot obtain future data. All we can do is predict the future based on current data. For example, for 6000 data points, I only know the first 3000, and for the last 3000, I only know the temperature and irradiance. Based on this situation, can I predict the power output of the last 3000 data?

Based on this idea, I modified the Python code and divided the 6000 data into two parts. The first 3000 data are the first part, which is a complete data set consisting of temperature, irradiance and power, and is used to train the model. The last 3000 data are the second part, which is an incomplete data set consisting of only temperature and irradiance. The purpose is to test whether the trained model can output accurate power values based on temperature and irradiance.

The results of running the code are as follows:

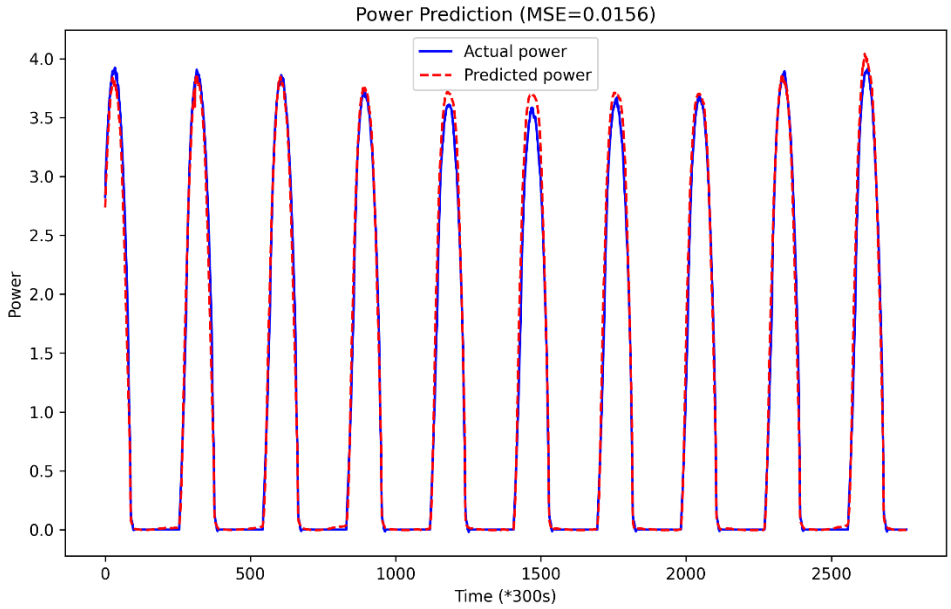

Figure 4. Predicted power vs. actual power

Figure 4 shows the difference between the model prediction and the true value. The first 3000 data have a good training effect on the model. The trained model can accurately output power according to the input temperature and irradiance (MSE=0.0156).

In addition, I also used another set of data [2] [3] to conduct data-based modeling. The method used was still SINDy. This set of data included air conditioning power, indoor temperature, indoor humidity and outdoor temperature. My goal was to use the change of outdoor temperature to predict the change of air conditioning power, indoor temperature and indoor humidity; compared with the strong correlation between power and temperature and irradiance in the above prediction model, the impact of outdoor temperature on air conditioning power is indirect, and indoor temperature and indoor humidity are affected by both air conditioning power and outdoor temperature at the same time. Therefore, the system is more complex, and the various features are coupled with each other.

Like the previous model, this time we still use 6,000 models for experiments. The first 3,000 data models are used to train the model, and the last 3,000 data are used to test the accuracy of the model. The results of running the code are as follows:

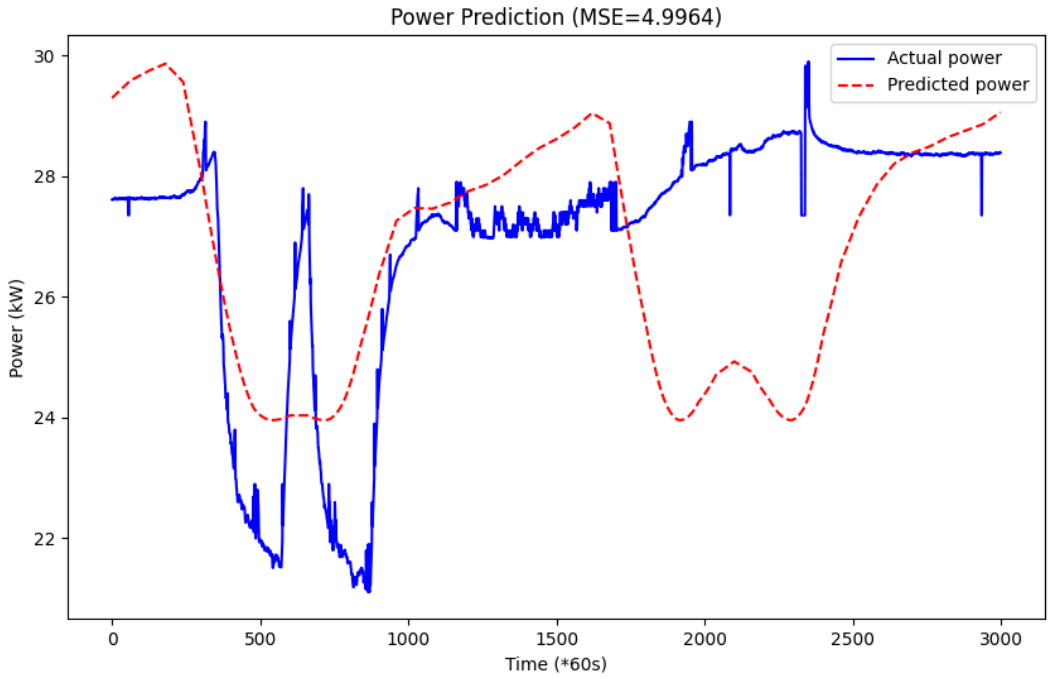

Figure 5. Air conditioning power prediction

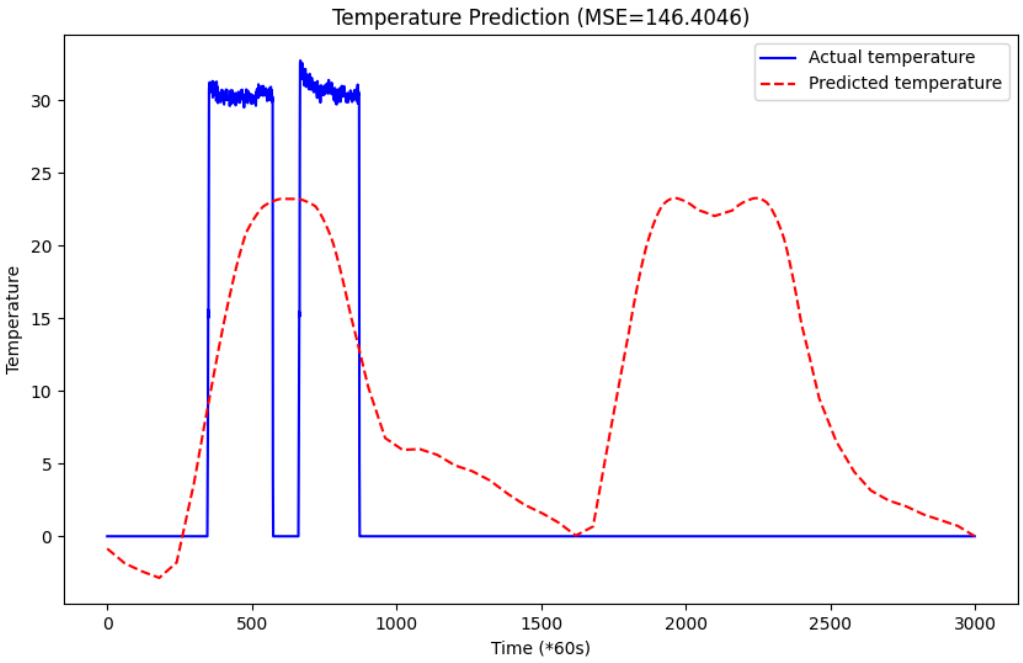

Figure 6. Indoor temperature prediction

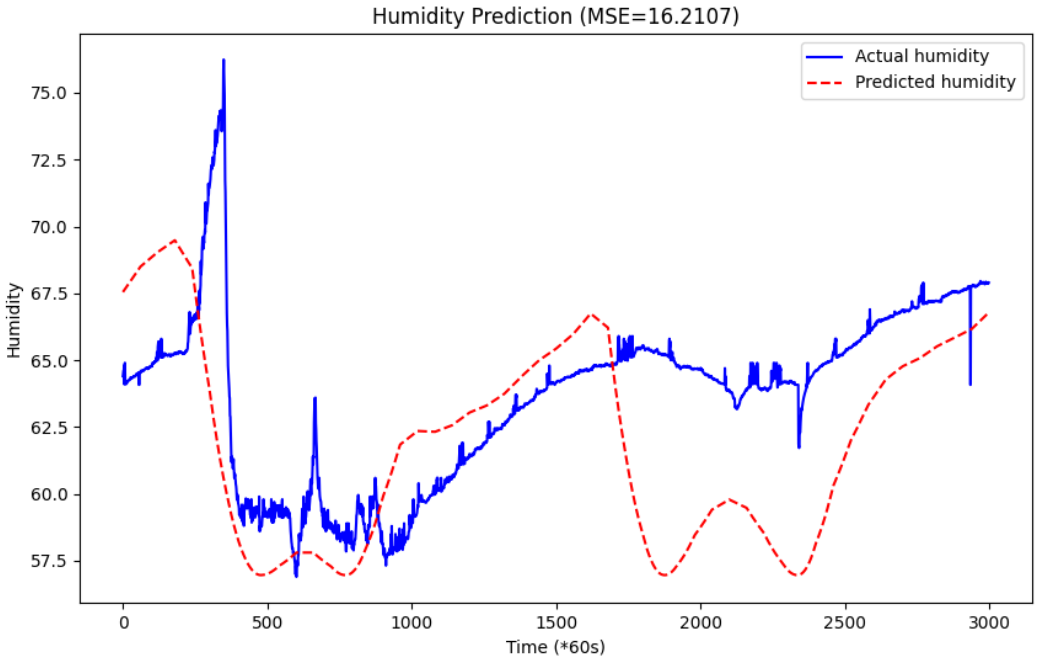

Figure 7. Indoor humidity prediction

As can be seen from the three figures above, the prediction results are not very accurate. The root mean square errors of different features are relatively large, especially the prediction model of indoor temperature (Fig. 6), with a root mean square error of 146.40. The reason for this phenomenon may be that the indoor temperature is affected by the outdoor temperature and the air conditioning power. If the indoor temperature is predicted based solely on the outdoor temperature, the predicted value may have a large deviation.

In addition, for air conditioning power, the increase in outdoor temperature will lead to an increase in indoor temperature, thereby indirectly increasing the power of the air conditioner. Therefore, the accuracy of using outdoor temperature to predict air conditioning power is the highest among the three (MSE=4.9964, Figure 5), but this does not mean that this model is competent for the task of air conditioning power prediction. I believe that the lack of training

data is also one of the reasons for the poor performance of the training model. This experiment only used 3,000 sets of data to train the model, which may be due to insufficient training volume, resulting in inaccurate model predictions.

Besides, the set temperature value of the air conditioner and its control mode will also affect the output power of the air conditioner, but these data are missing, so this may also be a reason for the error.

NN (Neural Networks)

I hope to use the neural network model to process the data used in the Ridge regression above again. Like the previous example, the first 3,000 samples are used to train the model, and the last 3,000 samples are used to test the model.

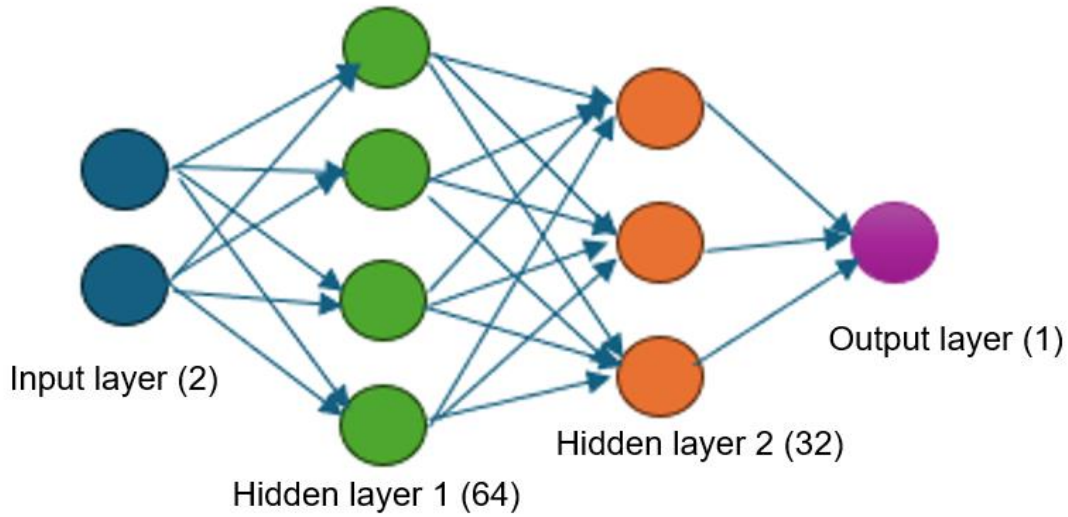

Figure 8. Neural network architecture

Figure 8 shows the brief structure of the neural network constructed in this experiment. I use the PyTorch library to build the neural network. The input has 2 neurons (2 features), the first hidden layer contains 64 neurons, with ReLU as the activation function, and 20% of the neurons are discarded to prevent

overfitting; the second hidden layer is similar, but the number of neurons is reduced to 32, and the output layer has one neuron to output the result. The training cycle (Epoch) is set to 1000.

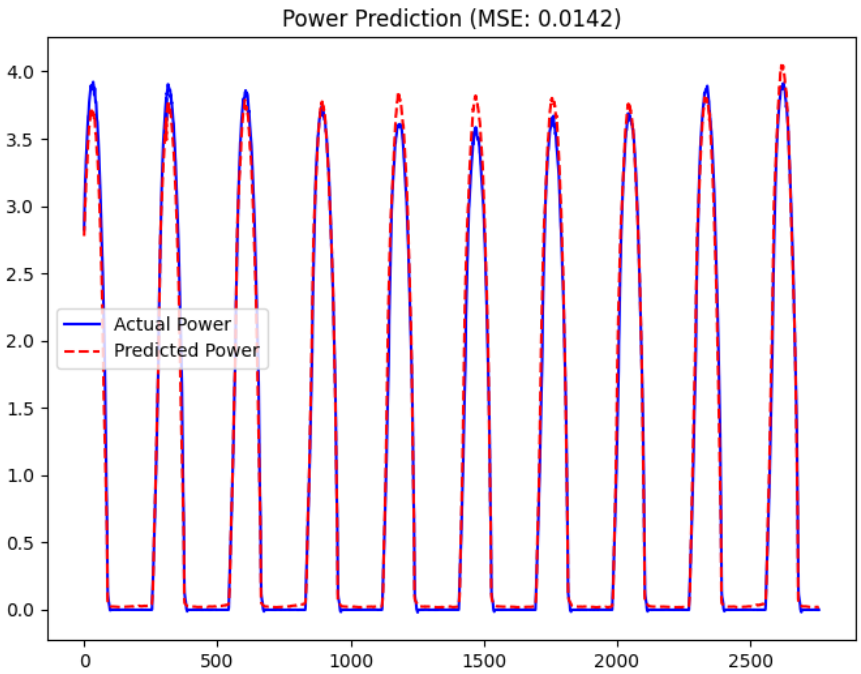

Figure 9. Predicted power vs. actual power (NN)

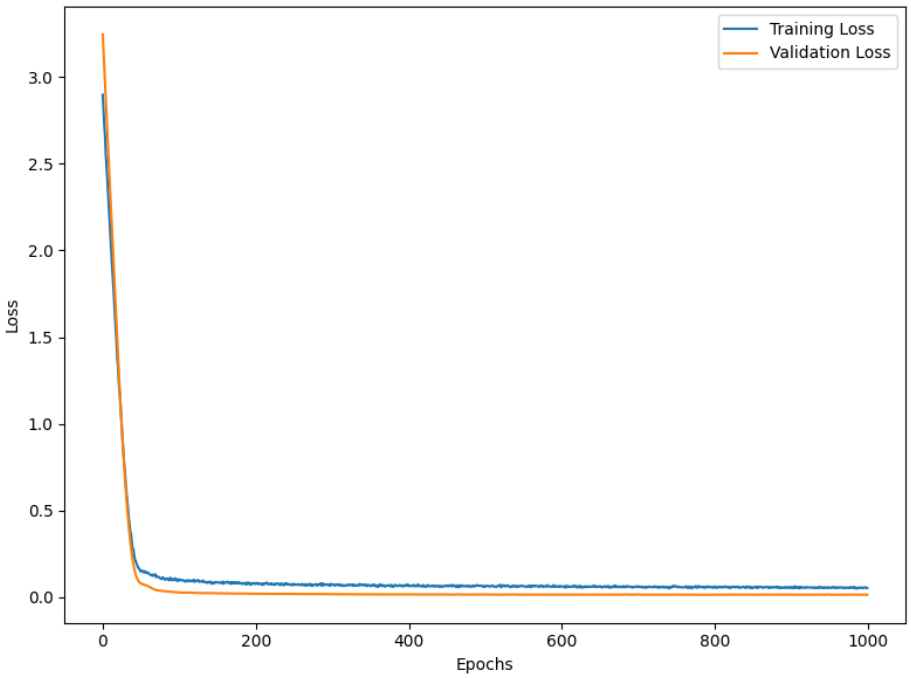

Figure 10. Training loss (NN)

Figures 9 and 10 show that the neural network has trained well, and the model predictions show good tracking of the true values. When the training cycle reaches 700, the training loss almost stops decreasing, which seems to indicate that the model has reached its optimal state, and further training will not bring performance improvement. In fact, in the first 70 data training, the validation loss of the model dropped from 3 to 0.1, indicating that the model not only performs well on the training data, but also generalizes well to new data.

For the same set of data, SINDy and NN seem to show similar prediction accuracy (MSE: 0.0156 vs 0.0142). The advantage of SINDy is that it can provide a clear and easy-to-interpret system dynamics model; once the equations are fitted through the data, it can directly use these equations for rapid prediction. The advantage of NN is that it can handle some complex systems with unclear physical mechanisms, because NN is essentially a black box that makes predictions by learning a large amount of data.

Reference

[1] "Home Page | DKA Solar Centre." https://dkasolarcentre.com.au/.

[2] Pipattanasomporn, M., Chitalia, G., Songsiri, J., Aswakul, C., Pora, W., Suwankawin, S., Audomvongseree, K. and Hoonchareon, N., 2020. CU-BEMS, smart building electricity consumption and indoor environmental sensor datasets. *Scientific Data*, *7*(1), p.241.

[3] LAADS S&O [2]: VJ121A1N--5200/2019-04-17..2019-04-30/DB/Site:118 (nasa.gov)