RNN Regression: Predicting Power Production

Garth Mortensen

November 11, 2018

Table of Contents

Overview

This analysis uses a Recurrent Neural Network (RNN) to predict a wind farm's power production. An RNN is a network specialized for processing a sequence of values.

Purpose

The dataset contains various (X) measurements (i.e. snow, temp., etc) from a power plant. The last column is the power generated on each day (Y). Perform a regression analysis using the **RNN/LSTM/GRU** approach, with **sequence length = 30**.

Recurrent Neural Networks

Recurrent Neural Networks are a family of Neural Networks (NNs) for processing sequential data. Much as a convolutional network is a NN specialized for processing a grid of values...a RNN [is] specialized for processing a sequence of values. Just as convolutional networks can readily scale to large images, and some can process images of variable size, recurrent networks can scale to much longer sequences than would be practical for networks without sequence-based specialization (Goodfellow p. 373).

A recurrent neuron is different from the traditional neuron in that a net composed of the latter merely feeds forward. A RNN, on the other hand, feeds forwards *and* through time steps (frames). Hands-on Machine Learning provides the following visualization of a recurrent neuron, unrolled (**unfolded**) through time.

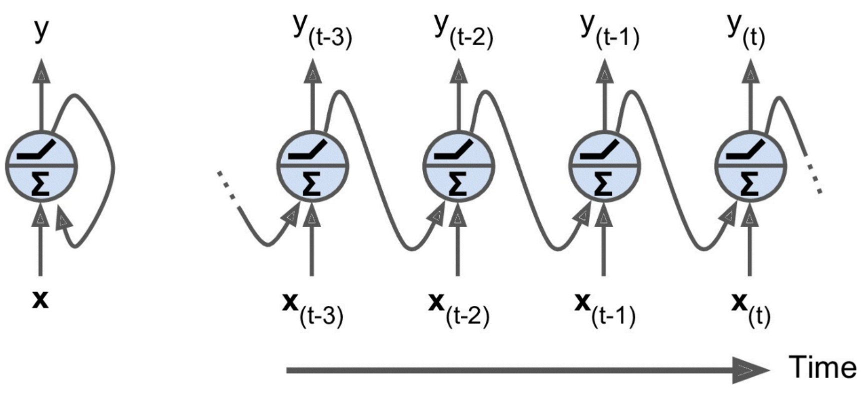


Figure 14-1. A recurrent neuron (left), unrolled through time (right)

(Aurélien Géron, pg 504)

At each time step t, every neuron receives both the input vector x(t) and the output vector from the previous time step y(t–1).

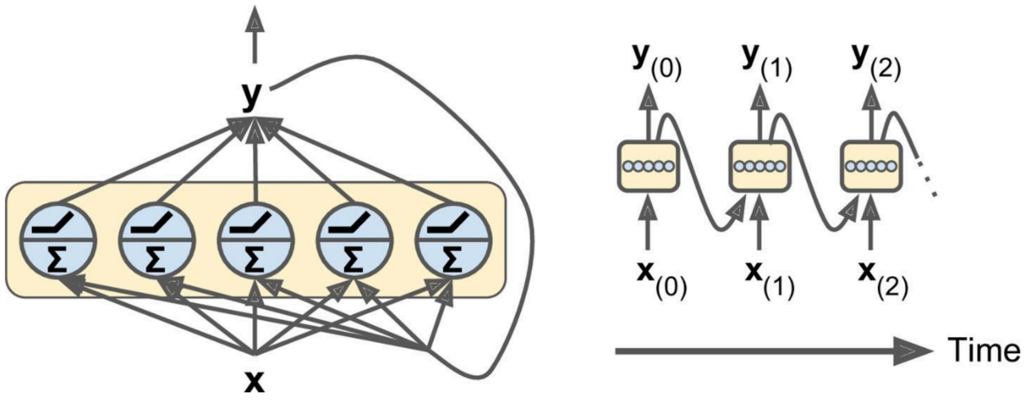


Figure 14-2. A layer of recurrent neurons (left), unrolled through time (right)

(Aurélien Géron, pg 505)

Géron (507) explains that each recurrent neuron has two sets of weights: one for the inputs x (t) and the other for the outputs of the previous time step, y(t–1). The part of a neural network that preserves some state across time steps is called a memory cell (or simply a cell). In general a cell's state at time step t, denoted h(t) (the "h" stands for "hidden"), is a function of some inputs at that time step and its state at the previous time step: h (t) = f(h(t–1) , x(t)). Its output at time step t, denoted y(t), is also a function of the previous state and the current inputs.

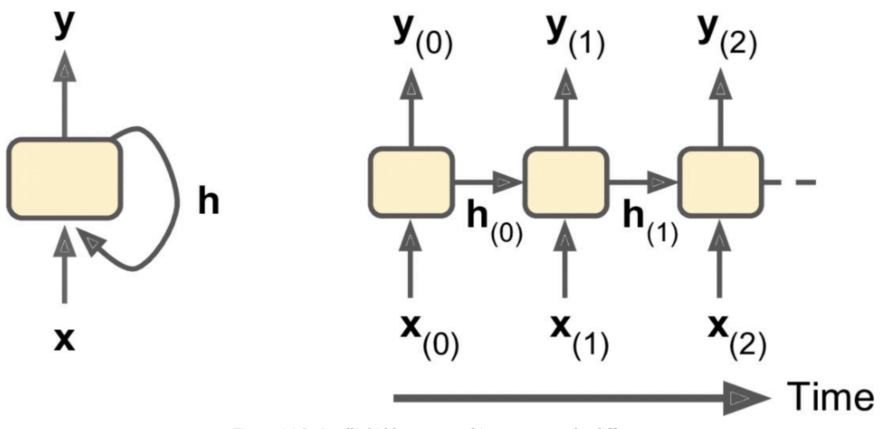


Figure 14-3. A cell's hidden state and its output may be different

(Aurélien Géron, pg 507)

This illustrates the sharing of parameters across a deep structure. Each member of the output is produced using the same update rules applied to the previous outputs. This recurrent formulation results in the sharing of parameters through a very deep computational graph (Goodfellow p. 364).

Long Short-Term Memory

A good summary of LSTMs by Matlab is here.

To train a deep neural network to predict numeric values from time series or sequence data, you can use a long short-term memory (LSTM) network.

Time to build an RNN.

Pre-Processing

Let's first clear variables and close graphs.

```
% Clear the worksapce
clear all
clc
close all
```

We'll measure how long it takes to run this entire worksheet, including training the RNN, using a stopwatch method. This computer is running an i7-6500 with 16GB DDR4-2400 RAM. You can use CPU-Z to determine your RAM speed. In Matlab, we can also look at our GPU specs.

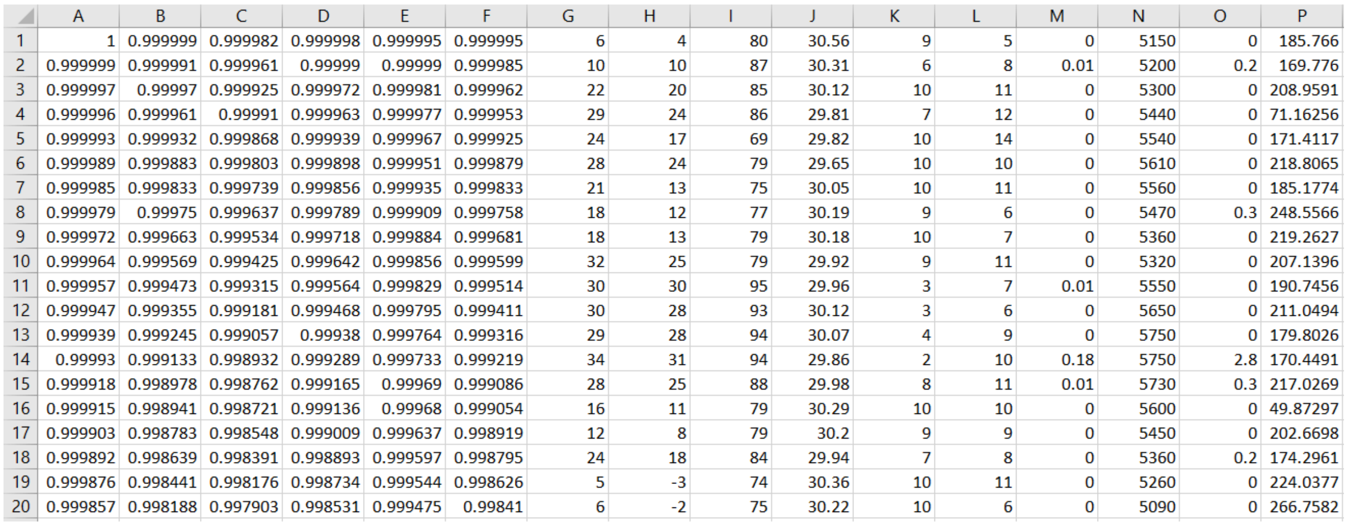
```
% display GPU count
gpuCountstring = ['Total GPUs: ', num2str(gpuDeviceCount)]
disp(gpuCountstring)

% if GPU exists, display properties
if gpuDeviceCount ~= 0
    disp(gpuDevice)
end
```

Start the stopwatch.

```
% Start stopwatch
% tic ENABLE ONCE COMPLETE
```

Load Data

The dataset we will be loading appears as:



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	1	0.999999	0.999982	0.999998	0.999995	0.999995	6	4	80	30.56	9	5	0	5150	0	185.766
2	0.999999	0.999991	0.999961	0.99999	0.99999	0.999985	10	10	87	30.31	6	8	0.01	5200	0.2	169.776
3	0.999997	0.99997	0.999925	0.999972	0.999981	0.999962	22	20	85	30.12	10	11	0	5300	0	208.9591
4	0.999996	0.999961	0.99991	0.999963	0.999977	0.999953	29	24	86	29.81	7	12	0	5440	0	71.16256
5	0.999993	0.999932	0.999868	0.999939	0.999967	0.999925	24	17	69	29.82	10	14	0	5540	0	171.4117
6	0.999989	0.999883	0.999803	0.999898	0.999951	0.999879	28	24	79	29.65	10	10	0	5610	0	218.8065
7	0.999985	0.999833	0.999739	0.999856	0.999935	0.999833	21	13	75	30.05	10	11	0	5560	0	185.1774
8	0.999979	0.99975	0.999637	0.999789	0.999909	0.999758	18	12	77	30.19	9	6	0	5470	0.3	248.5566
9	0.999972	0.999663	0.999534	0.999718	0.999884	0.999681	18	13	79	30.18	10	7	0	5360	0	219.2627
10	0.999964	0.999569	0.999425	0.999642	0.999856	0.999599	32	25	79	29.92	9	11	0	5320	0	207.1396
11	0.999957	0.999473	0.999315	0.999564	0.999829	0.999514	30	30	95	29.96	3	7	0.01	5550	0	190.7456
12	0.999947	0.999355	0.999181	0.999468	0.999795	0.999411	30	28	93	30.12	3	6	0	5650	0	211.0494
13	0.999939	0.999245	0.999057	0.99938	0.999764	0.999316	29	28	94	30.07	4	9	0	5750	0	179.8026
14	0.99993	0.999133	0.998932	0.999289	0.999733	0.999219	34	31	94	29.86	2	10	0.18	5750	2.8	170.4491
15	0.999918	0.998978	0.998762	0.999165	0.99969	0.999086	28	25	88	29.98	8	11	0.01	5730	0.3	217.0269
16	0.999915	0.998941	0.998721	0.999136	0.99968	0.999054	16	11	79	30.29	10	10	0	5600	0	49.87297
17	0.999903	0.998783	0.998548	0.999009	0.999637	0.998919	12	8	79	30.2	9	9	0	5450	0	202.6698
18	0.999892	0.998639	0.998391	0.998893	0.999597	0.998795	24	18	84	29.94	7	8	0	5360	0.2	174.2961
19	0.999876	0.998441	0.998176	0.998734	0.999544	0.998626	5	-3	74	30.36	10	11	0	5260	0	224.0377
20	0.999857	0.998188	0.997903	0.998531	0.999475	0.99841	6	-2	75	30.22	10	6	0	5090	0	266.7582

```
% local dir
% in order to read this file in, i had to in Excel retype cell 1, which had a
% formatting issue.
data = dlmread('C:\Users\grm\Google Drive\aStThomas\7ArtificalIntelligence\Assignments\6 RNN Re

%matlab online dir
%data = csvread('/MATLAB Drive/Data/TS_data_HW_18F.csv', 0, 1);
```

Partition the training and test data. Train on all but lthe last 30 days of the sequence, and test on the last 30

```
numTimeStepsTrain = length(data(1:end - 30, 1:1));

dataTrain = data(1:numTimeStepsTrain, :);
dataTest = data(numTimeStepsTrain + 1:end, :); % +1? Seems right.
```

Define Y Target

We define the Y column, which is power production.

```
YTrain = dataTrain(:, 16);
YTest = dataTest(:, 16);
```

Define X Matrix

We define the X columns, which are the remainder of the data matrix' columns.

```
XTrain = dataTrain(:, 1:end - 1);
XTest = dataTest(:, 1:end - 1);
```

Normalize Sequence Data

Because all measurements in our dataframe are in different units, we need to standardize. Normalize X to have zero mean and unit variance.

Standardize AFTER splitting, because you don't want the training data to leak information about the test data.

"To normalize sequence data, first calculate the per-feature mean and standard deviation of all the sequences. Then, for each training observation, subtract the mean value and divide by the standard deviation." Source

```
mu = mean([XTrain{:}],2);
sigma = std([XTrain{:}],0,2);
XTrain = cellfun(@(X) (X-mu)./sigma,XTrain,'UniformOutput',false);
```

Or...

```
mu = mean(dataTrain);
sig = std(dataTrain);

dataTrainStandardized = (dataTrain - mu) / sig;
```

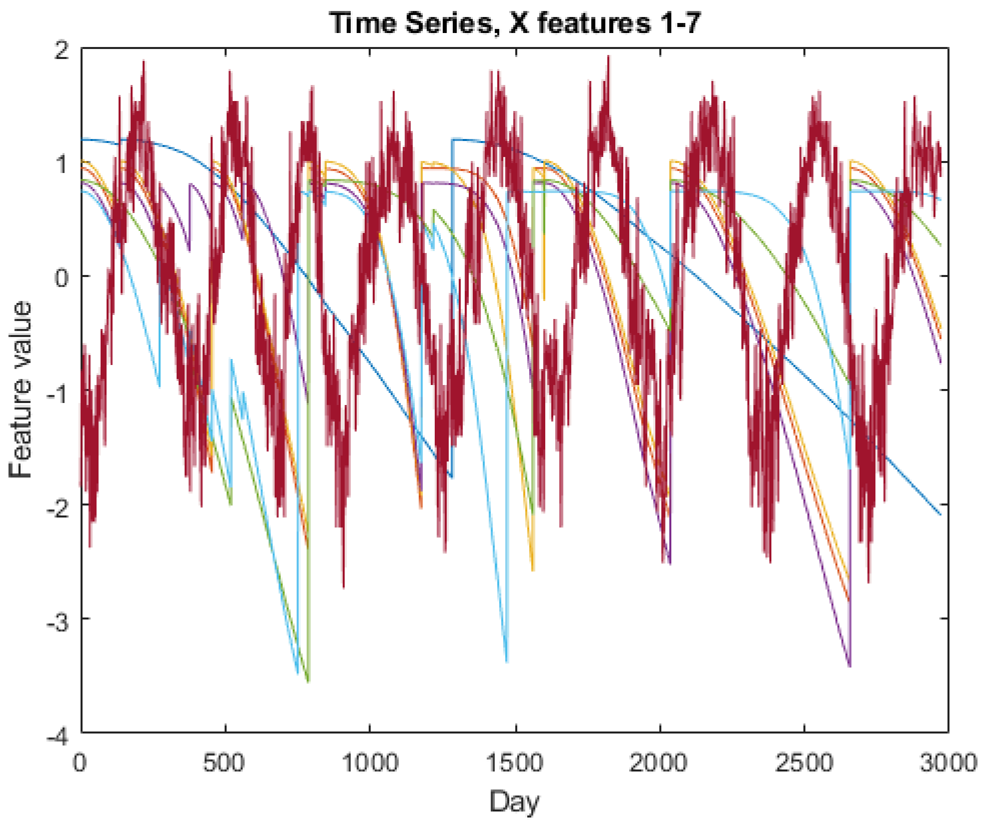
I believe the following provides the same results, but must test.

```
XTrain = zscore(XTrain);
XTest = zscore(XTest);

YTrain = zscore(YTrain);
YTest = zscore(YTest);
```

Visualize Time Series

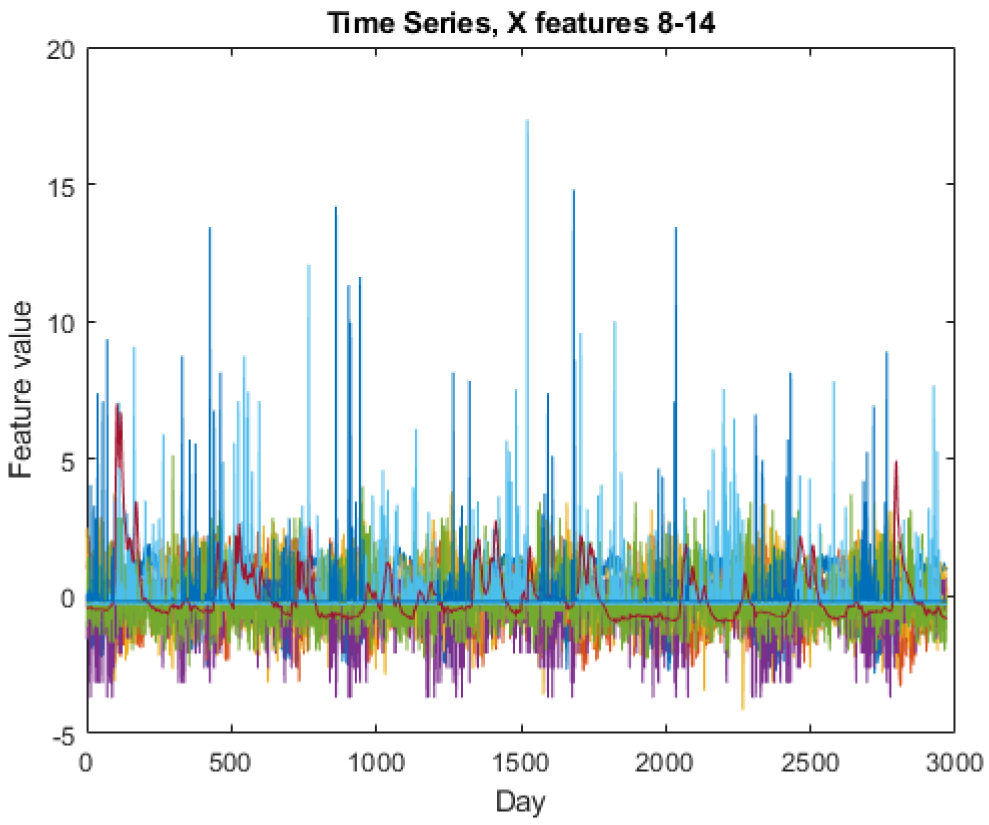
Plot X features 1 to 7.

```
figure
plot(XTrain(1:end, 1:7))
xlabel("Day")
ylabel("Feature value")
title("Time Series, X features 1-7")
```



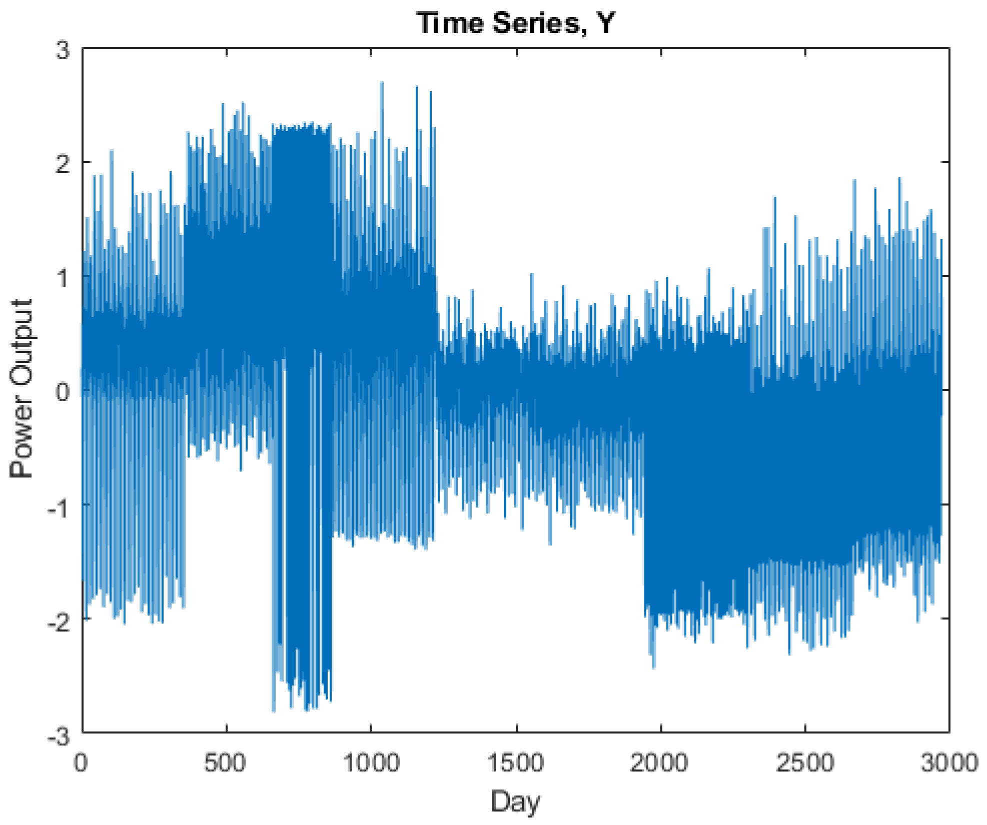
Plot X features 8 to 15.

```
figure
plot(XTrain(1:end, 8:15))
xlabel("Day")
ylabel("Feature value")
title("Time Series, X features 8-14")
```



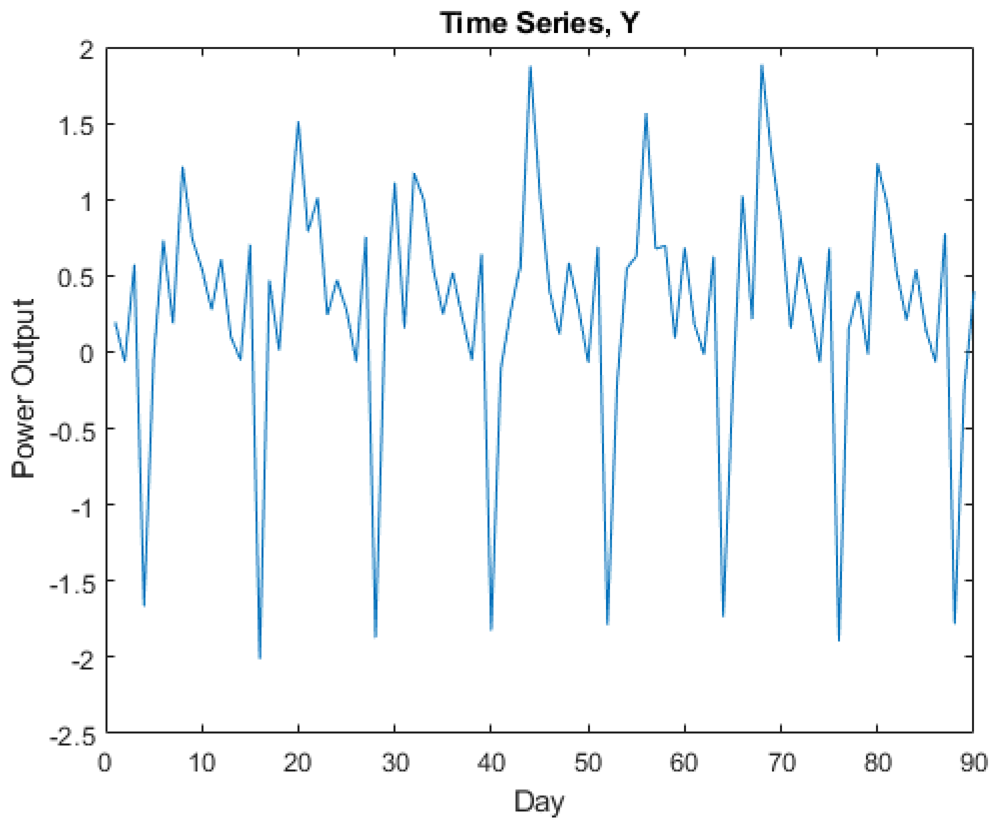
Plot Y.

```
figure
plot(YTrain(1:end, 1))
xlabel("Day")
ylabel("Power Output")
title("Time Series, Y")
```



Plot Y slide.

```
figure
plot(YTrain(1:90, 1))
xlabel("Day")
ylabel("Power Output")
title("Time Series, Y")
```



```
% XTrain = XTrain.';
% XTest = XTest.';
% YTrain = YTrain.';
% YTest = YTest.';

XTrain = num2cell(XTrain);
XTest = num2cell(XTest);
```

Modeling

Matlab Example here - https://www.mathworks.com/help/deeplearning/ug/long-short-term-memory-networks.html

Possible example to work from slide 37 LSTM - https://www.mathworks.com/help/deeplearning/examples/time-series-forecasting-using-deep-learning.html

Specify the size of the sequence input layer to be the **number of features** of the input data.

Specify the size of the fully connected layer to be the **number of responses**. You do not need to specify the sequence length.

```
numFeatures = 15;
```

```
numResponses = 1;
numHiddenUnits = 50;

layers = [ ...
    sequenceInputLayer(numFeatures) % RNN
    lstmLayer(numHiddenUnits,'OutputMode','sequence') % LSTM seq-to-seq, M:M
    fullyConnectedLayer(numResponses), % seq response must have same len as predictors
    regressionLayer];
```

Set options.

```
options = trainingOptions('adam', ...
'MaxEpochs',100, ...
'GradientThreshold',1, ...
'InitialLearnRate',0.005, ...
'LearnRateSchedule','piecewise', ...
'LearnRateDropPeriod',125, ...
'LearnRateDropFactor',0.2);

net = trainNetwork(XTrain, YTrain, layers, options); % X.' ?
```

```
Error using trainNetwork (line 150)
Invalid training data. Predictors must be a N-by-1 cell array of sequences, where N is the number of
sequences. All sequences must have the same feature dimension and at least one time step.
```

```
% net = predictAndUpdateState(net, XTrain); %2
%
% [net, YPred] = predictAndUpdateState(net, Y(end)); %3
%
% for i = 2 : numel(Xtest) %4
%     [net, YPred(:,i)] = predictAndUpdateState(net, YPred(:,i-1));
% end
```