Machine Learning for Time Series Data Analysis in Smart Grid

Yan Zhang

Professor, University of Oslo, Norway

Learning Objectives

Throughout this lecture, it is aimed for the students to be able to

- Understand time series analysis concepts
- Understand how time series data analysis can be converted into a machine problem, e.g., linear regression, multiple linear regression
- Understand how time series data analysis can be converted into deep learning problem, e.g., neural networks, recurrent neural network

Industry Invited Talk Today

- **Speakers**: Vivi Mathiesen, *Head of Section, NVE (Norges vassdrags- og energidirektorat)*

- **Title**: Energy Market and Nord Pool

- **NVE**: NVE is a directorate under the Ministry of Petroleum and Energy and is responsible for the management of Norway's water and energy resources. More information: http://www.nve.no/

NVE

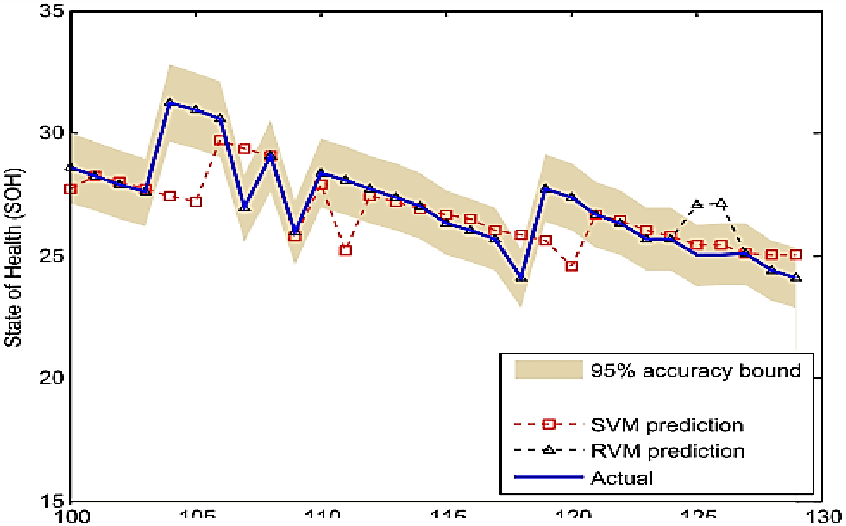
Outline

- **Time series analysis concepts**
- **Time series forecasting is converted into a machine learning problem**
  - **Linear Regression (LR)**
  - **Multiple Linear Regression (MLR)**
- **Time series forecasting is converted into an advanced deep learning problem**
  - **ANN (Artificial Neural Networks)**
  - **RNN (Recurrent Neural Networks)**

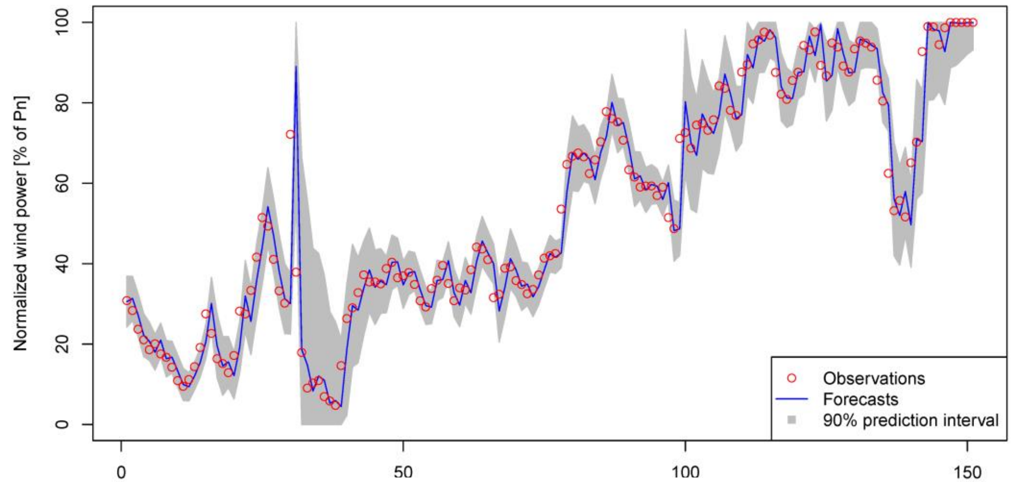
Time series data: definition and applications

- Time series: an ordered sequence of values of a variable at equally spaced time intervals.
- The usage of time series models includes
  - Prediction: the future based on the past
  - Control: the process producing the series
  - Understanding: the mechanism generating the series
  - Description: the main features of the series
- Time Series Analysis is used for many applications such as:
  - Economic forecasting, sales forecasting, budgetary analysis, stock market analysis, quality control, inventory studies, workload projections, energy market analysis

Time series data in smart grid

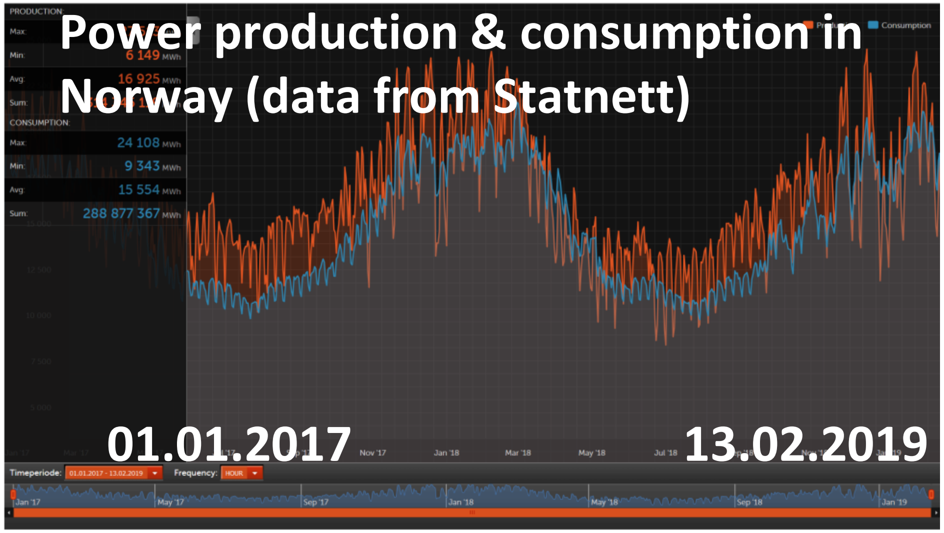


Battery health



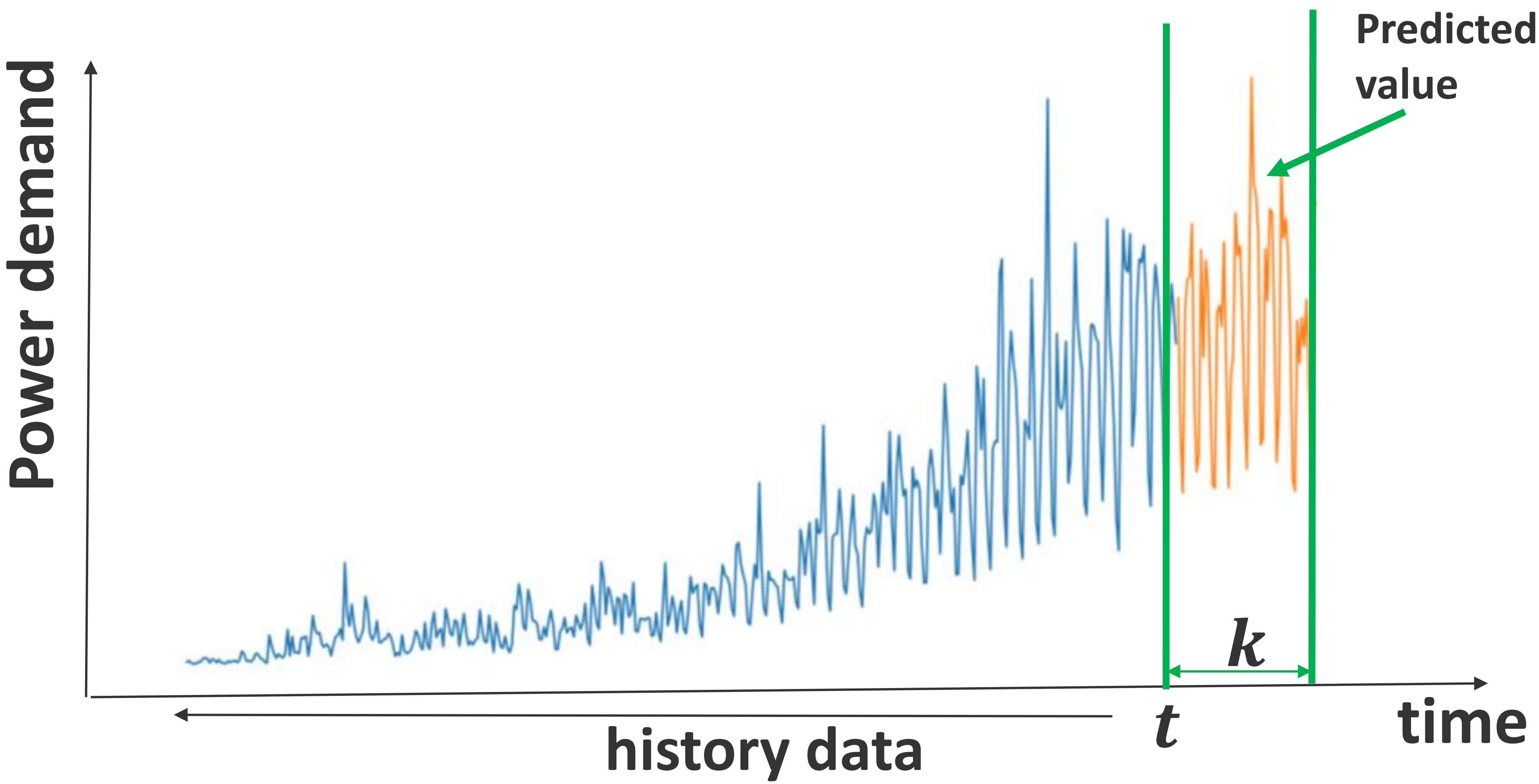
Wind power generation

Electricity prices



https://www.statnett.no/en/for-stakeholders-in-the-power-industry/data-from-the-power-system/#production-and-consumption

Time series prediction

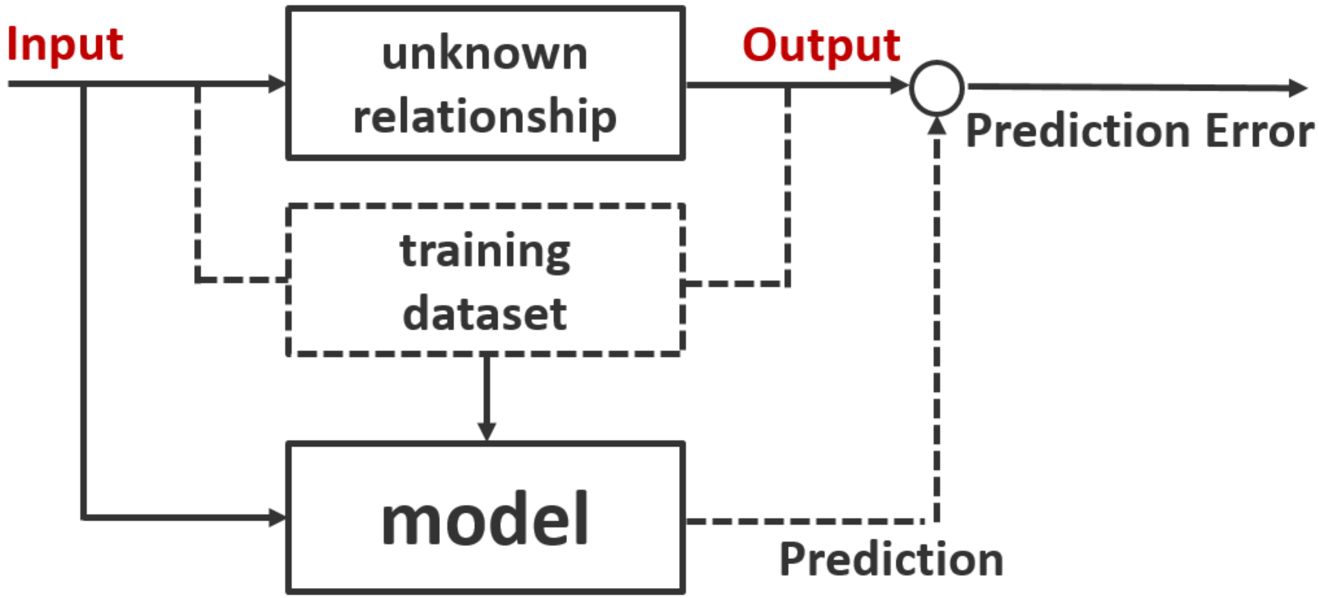


- Time series prediction: an estimate of power demand in the next k hours, conditional to all history data up to time t

- **Note**: at time t+3, the power demand at time t+1 and t+2 are already known. In this case, all data until time t+2 are the history data.

CONVERTING TIME SERIES ANALYSIS INTO MACHINE LEARNING PROBLEM

Supervised learning



- The relationship between output and input is unknown. We need to infer from historical data the possibly dependence between the input and the output.

- We can build the model based on the training dataset to show the relationship between output and input. Then, we can use the built model to make prediction.

Time series prediction: a simple example

4,2,3,6,9,?

- **Q:** we have a series of numbers, what is the number after 9?

- **Sliding window method**: the use of previous time steps to predict the next time step. **Window size**: the number of previous time steps. In this example, window size = 1

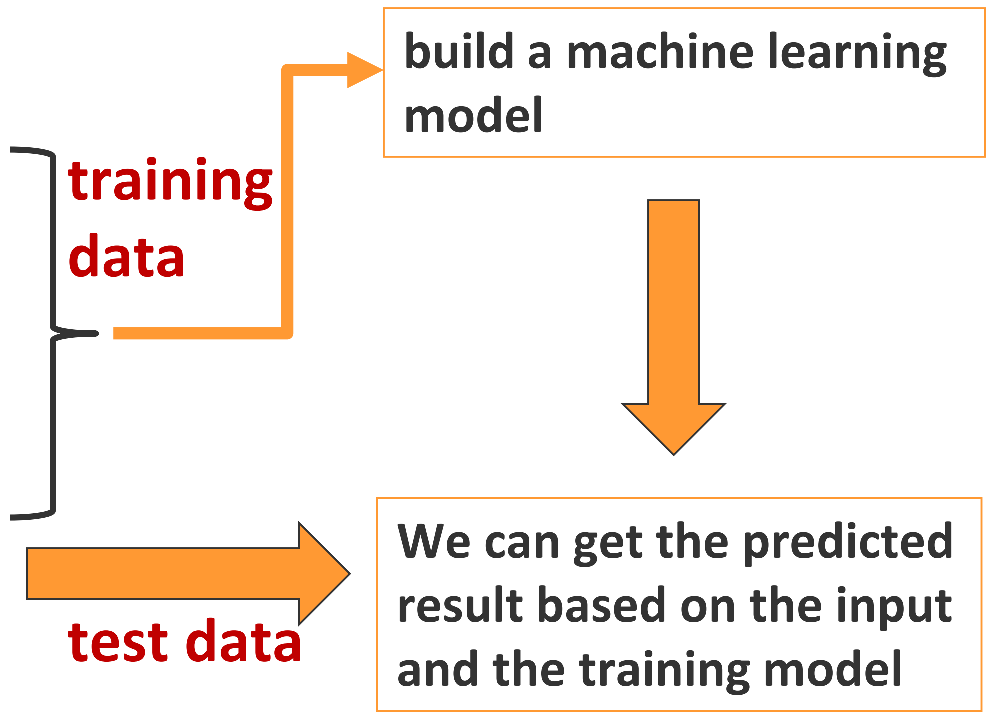
- Time series prediction can be structured as a supervised learning problem.

X (input)	Y (output)
4	2
2	3
3	6
6	9
9	?

Time series prediction is structured as a supervised learning problem

- **The relationship between output and input is unknown. We can use the training data to build the model to show the relationship between output and input. Then, we can use the built model to make prediction.**

X (input)	Y (output)
4	**2**
2	**3**
3	**6**
6	**9**
9	**?**



Main observations

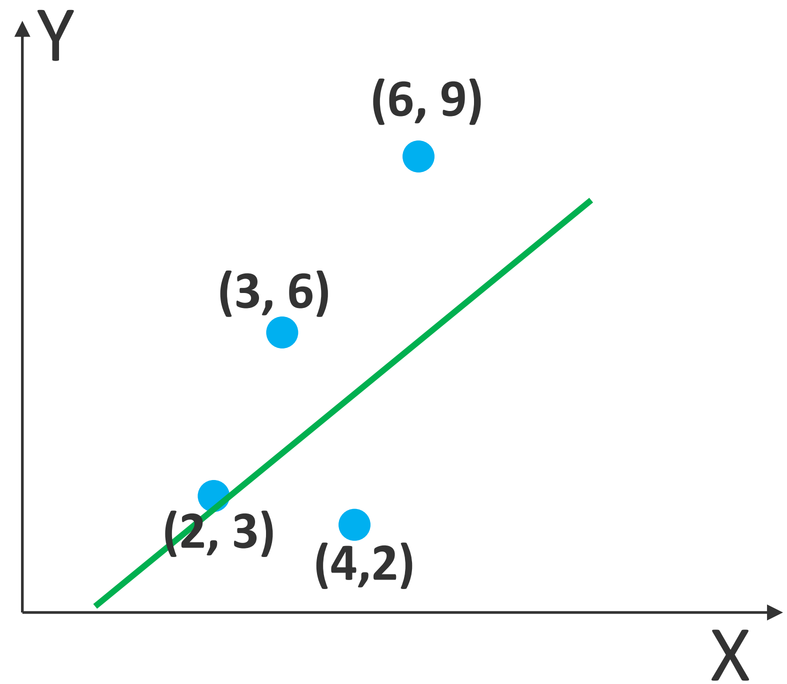
- For the training model: the previous time step is the input **X** and the next time step is the output **Y**
- The order between the observations is preserved, and must continue to be preserved when using this dataset to train a supervised model
- When a time series dataset is prepared this way, any of the standard machine learning algorithms may be applied, as long as the order of the rows is preserved.

- For example, if we assume the linear relationship between **Y** and **X**, then we can use linear regression model $\mathbf{y} = w_0 + w_1 \boldsymbol{x}$

Recall linear regression model

- The linear model is defined as: $y = w_0 + w_1 x$
- Given N observations (x_i, y_i); $(\mathrm{i} = 1, 2, \dots \mathrm{N})$, we need to find w_0 and w_1

```
x <- c(4, 2, 3, 6)
y <- c(2, 3, 6, 9)
#linear regression function
lmOut = lm(y ~ x)
# make the prediction
a <- data.frame(x = 9)
result <-  predict(lmOut, a)
```



Output: 4.322580645

Multiple linear regression

4,2,3,6,9,?

- Multiple linear regression: model the relationship between two or more variables and an output variable by fitting a linear equation to the data.

- The size of sliding window can be increased to include more previous time steps, and hence we can build multiple linear regression

Window size = 2

X1	X2	Y
4	2	**3**
2	3	**6**
3	6	**9**
6	**9**	**?**

$$\mathbf{y} = w_0 + w_1\boldsymbol{x}_1 + w_1 x_2$$

Window size = 3

X1	X2	X3	Y
4	2	3	**6**
2	3	6	**9**
3	**6**	**9**	**?**

$$\mathbf{y} = w_0 + w_1\boldsymbol{x}_1 + w_1 x_2 + w_3 x_3$$

Multiple linear regression

- The linear model is defined as: $\mathbf{y} = w_0 + w_1 \boldsymbol{x}_1 + w_2 \boldsymbol{x}_2$
- Given N observations (x_{1i}, x_{2i}, y_i); $(\mathbf{i} = \mathbf{1}, \mathbf{2}, \ldots \mathbf{N})$, we need to find $\boldsymbol{w}_0$, $\boldsymbol{w}_1$, and $\boldsymbol{w}_2$

```
x1 <- c(4, 2, 3)

x2 <- c(2, 3, 6)

y <- c(3, 6, 9)

#multiple linear regression

lmOut = lm(y ~ x1 + x2)

# make the prediction

x <- c(6, 9)

result <- coef(lmOut)[1] + coef(lmOut)[2]*x[1] + coef(lmOut)[3]*x[2]
```

Output: 10.28

Wind power prediction case

- "POWER": wind power measurement. The file only contains hourly wind power measurements

- **Q1:** what is the power generation at time 10:00?

- **Q2:** what is the power generation at time 11:00?

- At 11:00, we have the power generation value already at 10:00. We can use the power generation value at 10:00 as input.

TIMESTAMP,POWER
20120101 1:00,0.2736781568
20120101 2:00,0.0867959455
20120101 3:00,0.0068114015
20120101 4:00,0.0186459868
20120101 5:00,0.0348118328
20120101 6:00,0.0219168973
20120101 7:00,0.01823263
20120101 8:00,0.0096419971
20120101 9:00,0.0055353869

20120101 10:00	?
20120101 11:00	?

Real wind power data from wind farm in Australia

- **The data is between 2012.01.01-2013.10.01, which has 15336 data records**
- **POWER_i: power generation value in the i^{th} data records (i=1,2…15336)**

TIMESTAMP,POWER	
20120101 1:00,0.2736781568	POWER_1
20120101 2:00,0.0867959455	POWER_2
20120101 3:00,0.0068114015	POWER_3
20120101 4:00,0.0186459868	POWER_4
20120101 5:00,0.0348118328	POWER_5
20120101 6:00,0.0219168973	POWER_6
20120101 7:00,0.01823263	POWER_7
20120101 8:00,0.0096419971	POWER_8
20120101 9:00,0.0055353869	POWER_9
⋮	⋮
20131001 00:00 0.9310143417	POWER_15336

Linear regression for time series prediction (I)

- We need to build the linear function to find the predicted wind power generation in the last two weeks, e.g., 2013.09.17-2013.10.01.

X (input)	Y (output)
POWER_1	**POWER_2**
POWER_2	**POWER_3**
POWER_3	**POWER_4**
…	**…**
POWER_14999	**POWER_15000**
POWER_15000	**?**

training data

build a linear regression model

$$\mathbf{y} = w_0 + w_1 \boldsymbol{x}$$

test data

We can get the predicted result based on the input and the linear regression model:

$w_0 + w_1 * \boldsymbol{POWER\_15000}$

Linear regression for time series prediction (II)

```
#read data from CSV file
data <- read.csv("WindPowerData.csv)

#using window sliding method to define input and output in
#the training model
XTrain = data$POWER[1:14999]
YTrain = data$POWER[2:15000]

#linear regression
lmOut = lm(YTrain ~ XTrain)
```

Define input and output data to predict the 15001th power generation on 2013.09.17, 1:00

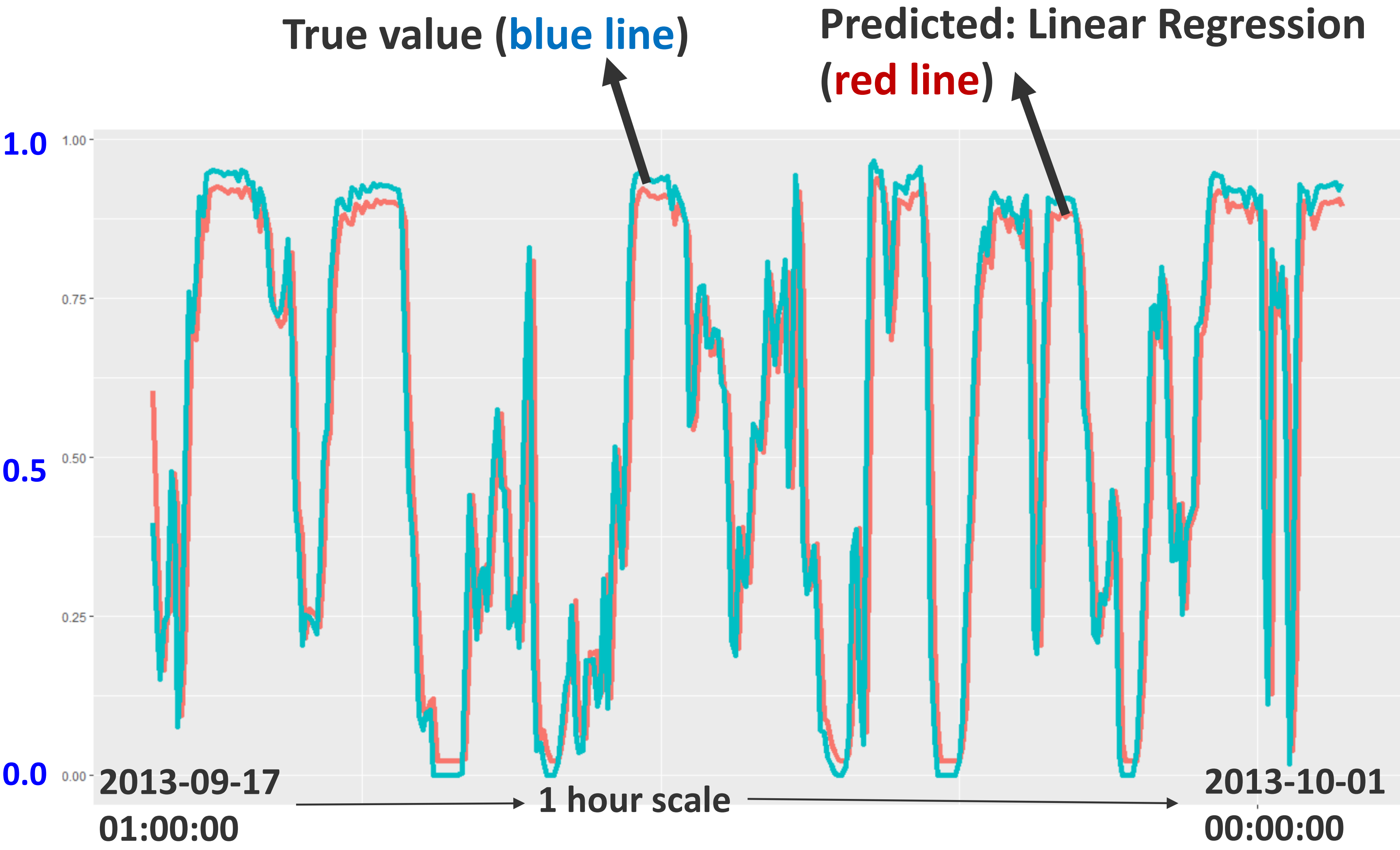
linear regression

```
#using window sliding method to define input and output in
#the training model
XTrain = data$POWER[1:15000]
YTrain = data$POWER[2:15001]

#linear regression
lmOut = lm(YTrain ~ XTrain)
```

Define input and output data to predict the 15002th power generation on 2013.09.17, 2:00

At this time, we have the true value of power generation on 2013.09.17, 1:00. We can use the power generation value at this time as input.

True value & Predicted Wind Power for the Test Data (RMSE= 0.14)



Recall multiple linear regression model

- **Multiple linear regression: model the relationship between two or more variables and an output variable by fitting a linear equation to the observed data.**
- **For example: Wind power generation is not only dependent on wind speed. It is also related to wind direction, temperature, and pressure. If we consider more factors affecting wind power generation, we have Multiple Linear Regression model**

$$\text{Wind Power} = w_0 + w_1 * \text{WindSpeed} + w_2 * \text{WindDirection} + w_3 * \text{Temperature} + w_4 * \text{Pressure}$$

Code:
```
lm(powerTrain ~ wsTrain + winddirection +
temperature + pressure)
```

Multiple linear regression for time series prediction (I)

- The size of sliding window can be increased to include more previous time steps, and we can build multiple linear regression, e.g., **window size = 2**

X1 (input)	X2 (input)	Y (output)
POWER_1	POWER_2	**POWER_3**
POWER_2	POWER_3	**POWER_4**
POWER_3	POWER_4	**POWER_5**
…		…
POWER_14998	POWER_14999	**POWER_15000**
POWER_14999	**POWER_15000**	**?**

training data

build a multiple linear regression model

test data

Get the predicted result based on the input and the training model

Multiple linear regression for time series prediction (II)

```
#read data from CSV file
data <- read.csv("WindPowerData.csv)

#using window sliding method to define input and output in
#the training model
X1Train = data$POWER[1:14998]
X2Train = data$POWER[2:14999]
YTrain = data$POWER[3:15000]

#multiple linear regression
lmOut = lm(YTrain ~ X1Train + X2Train)
```

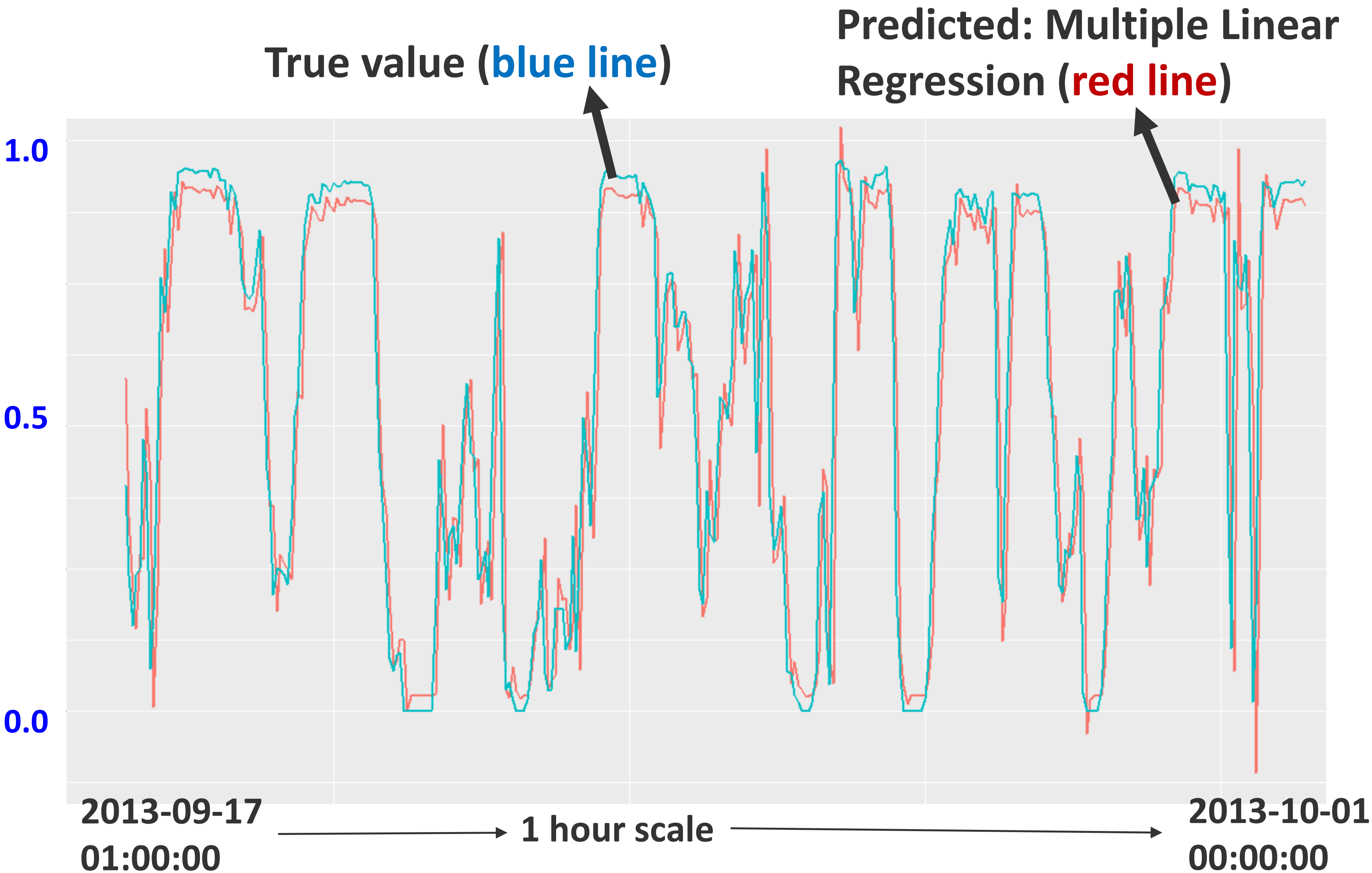
Define input and output to predict the 15001th power generation on 2013.09.17, 1:00

Multiple linear regression

```
#using window sliding method to define input and output in
#the training model
X1Train = data$POWER[1:14999]
X2Train = data$POWER[2:15000]
YTrain = data$POWER[3:15001]

#multiple linear regression
lmOut = lm(YTrain ~ X1train + X2Train)
```

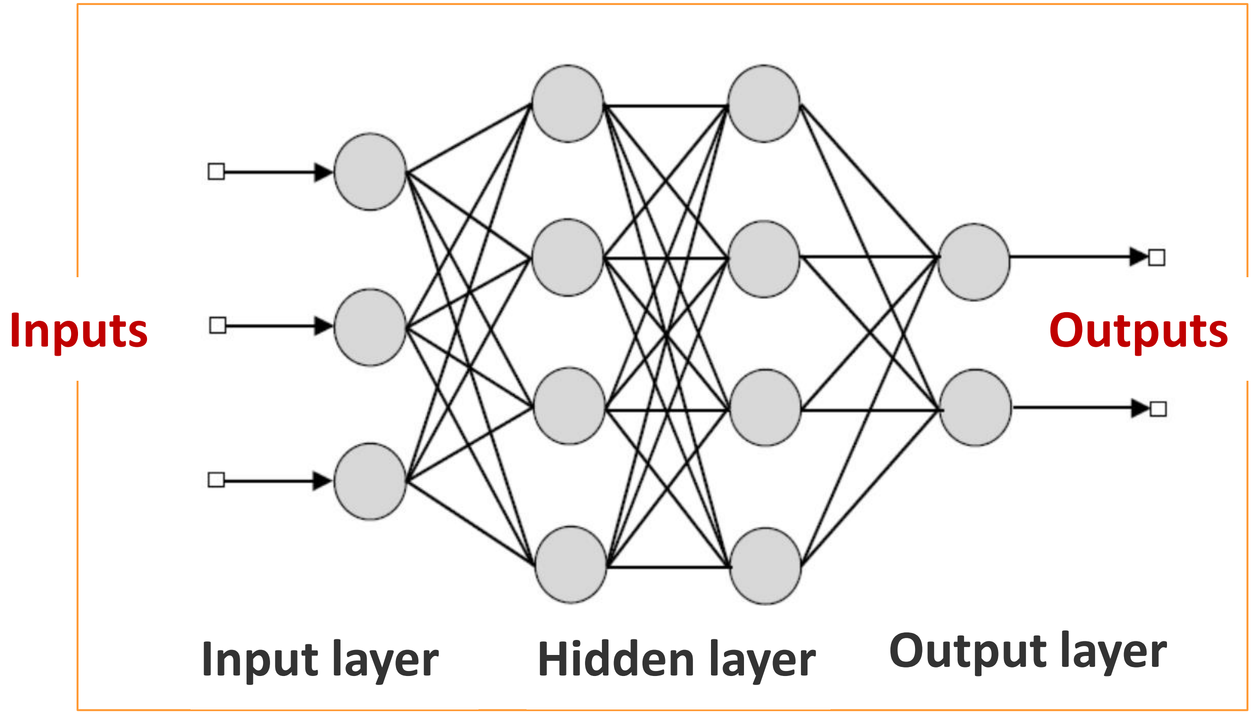
Define input and output data to predict the 15002th power generation on 2013.09.17, 2:00

True value & Predicted Wind Power for the Test Data (RMSE = 0.13)



CONVERTING TIME SERIES ANALYSIS INTO NEURAL NETWORK PROBLEM

Artificial Neural Network (ANN) model



- Input layers: Layers that take inputs based on existing data
- Hidden layers: Layers that use backpropagation to determine the weights in order to improve the prediction accuracy of the model
- Output layers: Output of predictions based on the data from the input and hidden layers

Time series prediction: a simple example

4,2,3,6,9,?

- **Q:** we have a series of numbers, what is the number after 9?

- We can build a neural network by using the previous time step to predict the immediate next time step, i.e., **window size=1**. Then, this neural network model has
  - one node in input layer
  - one node in output layer
  - one or multiple hidden layers

X (input)	Y (output)
4	**2**
2	**3**
3	**6**
6	**9**
9	**?**

Neural Networks for time series prediction

- **The data is between 2012.01.01-2013.10.01, which has 15336 data records. We need to use neural network to predict wind power generation in the last two weeks, e.g., 2013.09.17-2013.10.01.**
- **POWER_i: power generation value in the i^{th} data records (i=1,2…15336)**

X (input)	Y (output)
POWER_1	**POWER_2**
POWER_2	**POWER_3**
POWER_3	**POWER_4**
…	…
POWER_14999	**POWER_15000**
POWER_15000	**?**

training data

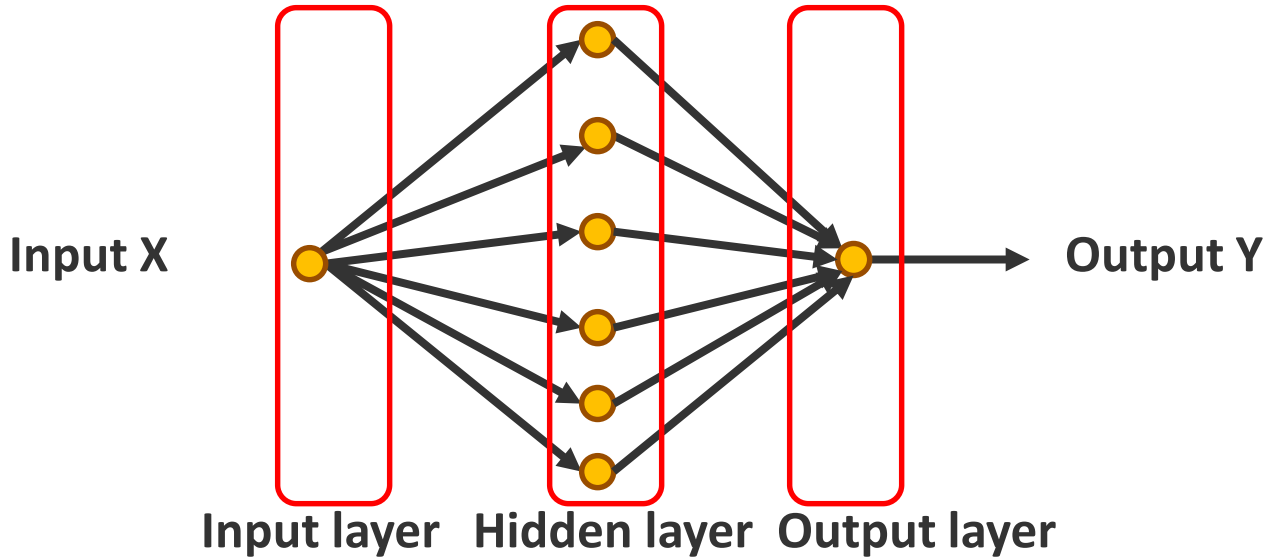
build a neural network model

test data

We can get the predicted result based on the input and the training model

Neural network model for wind energy forecasting

- One input node in the input layer
- One hidden layer, 6 hidden nodes in the hidden layer
- One output node



Source code in R for neural network model

```
#read data from CSV file
data <- read.csv("WindPowerData.csv)
#define input and output in #the training model
XTrain = data$POWER[1:14999]
YTrain = data$POWER[2:15000]
trainingData <- data.frame(XTrain, YTrain)

#neural network model
NNModel = neuralnet(YTrain ~ Xtrain, data=trainingData,
hidden = 6)
```

Predict the 15001th power generation on 2013.09.17, 1:00

Neural network

```
#define input and output in #the training model
XTrain = data$POWER[1:15000]
YTrain = data$POWER[2:15001]
trainingData <- data.frame(XTrain, YTrain)

#neural network model
NNModel = neuralnet(YTrain ~ Xtrain, data=trainingData,
hidden = 6)
```

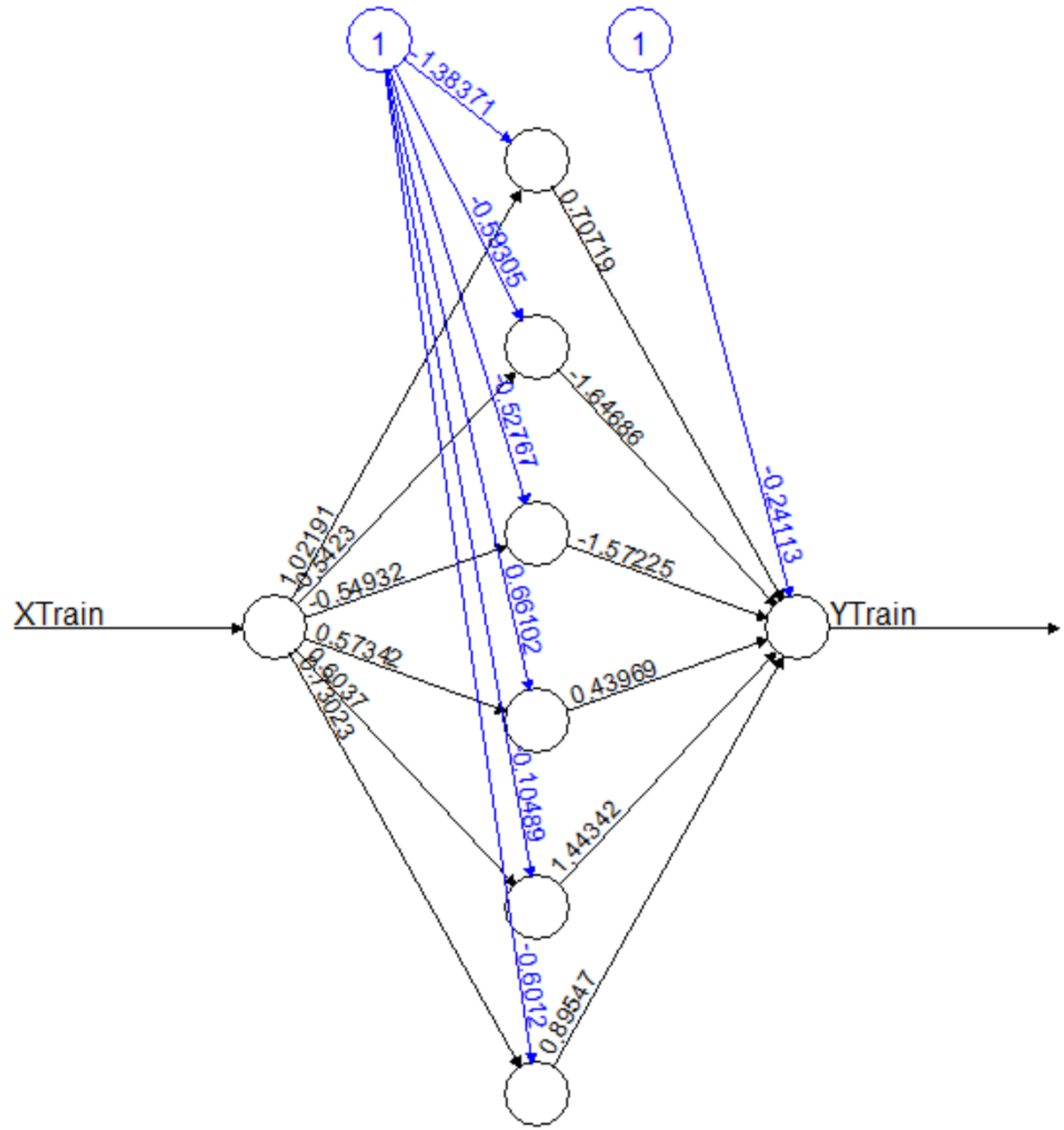
Predict the 15002th power generation on 2013.09.17, 2:00

Note: at 2013.09.17, 2:00, we have the true power generation value already at 2013.09.17, 1:00. We can use the value at 2013.09.17, 1:00 as input.

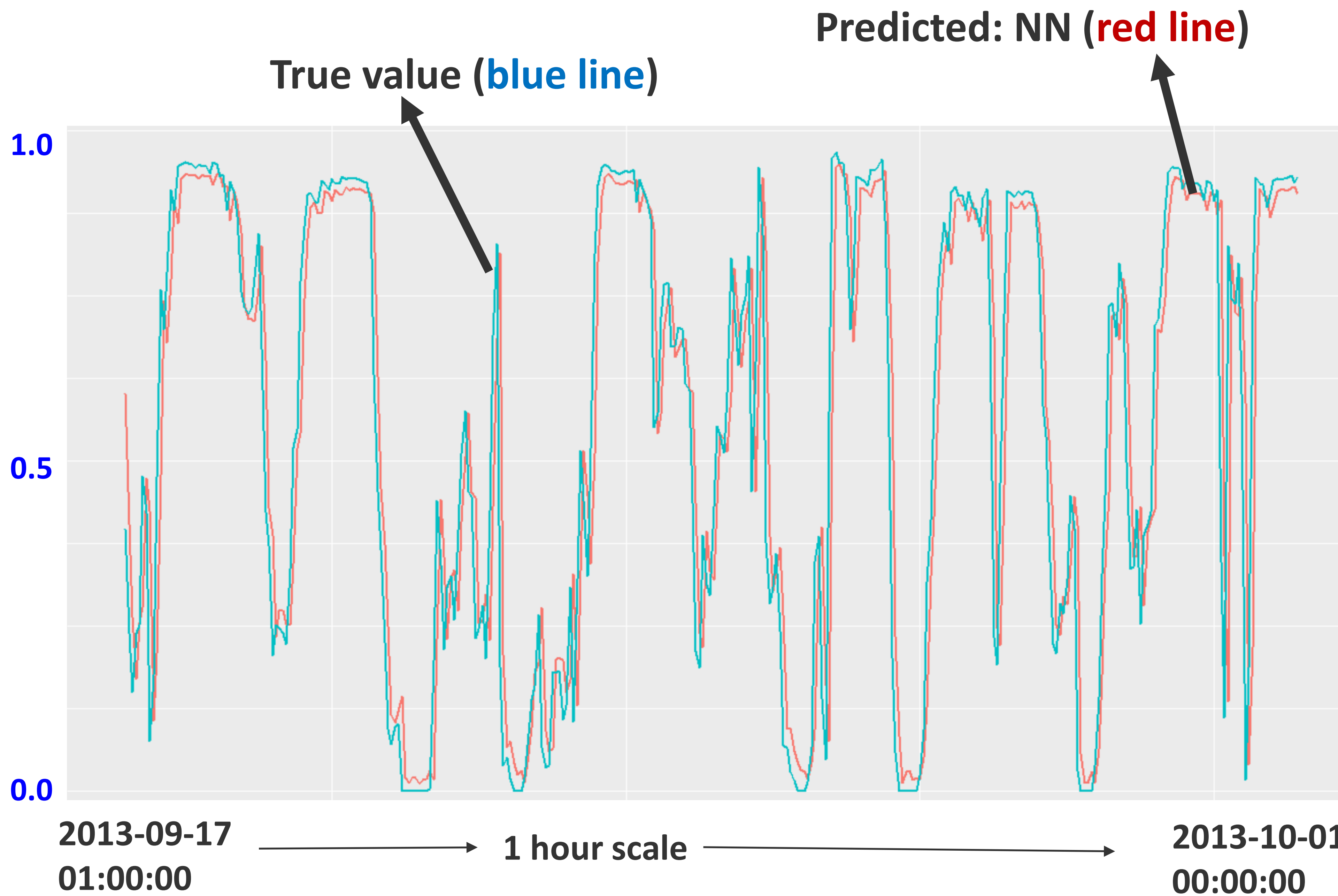
Neural network model with one hidden layer

- **The neural network model: one input, one hidden layer with 6 nodes, one output**

- **This model is used for predicting the 15001th power generation on time: 2013.09.17, 1:00**

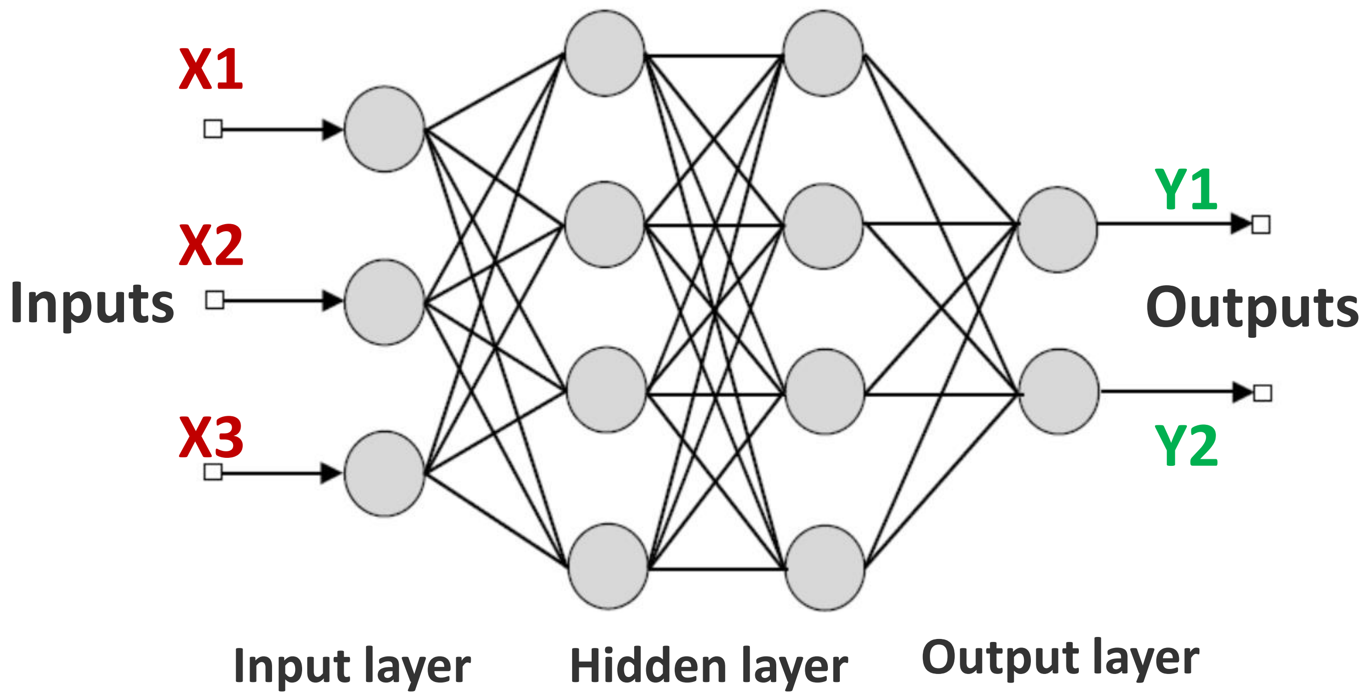


Real & Predicted Wind Power (RMSE=0.13)



Multiple input and multiple output neural network model for time series forecasting (I)

- Q: can we build a neural network model with **3 input** and **2 output** to make the prediction?



Multiple input and multiple output neural network model for time series forecasting (II)

- **Define input and output data**

X1 (input)	X2 (input)	X3 (input)	Y1 (output)	Y2 (output)
POWER_1	POWER_2	POWER_3	**POWER_4**	**POWER_5**
POWER_2	POWER_3	POWER_4	**POWER_5**	**POWER_6**
POWER_3	POWER_4	POWER_5	**POWER_6**	**POWER_7**
…	…	…	**…**	**…**
POWER_14996	POWER_14997	POWER_14998	**POWER_14999**	**POWER_15000**
POWER_14997	**POWER_14998**	**POWER_14999**	**?**	**?**

RECURRENT NEURAL NETWORK (RNN) FOR WIND ENERGY FORECASTING

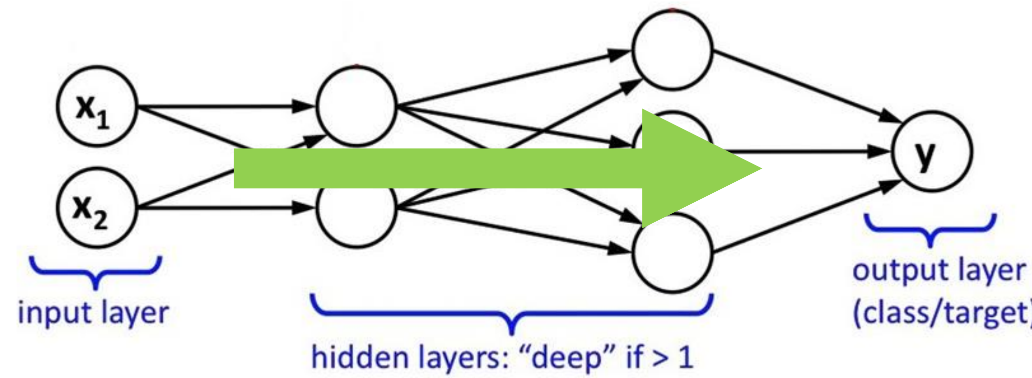
Challenges of traditional feedforward neural networks

- **Feedforward neural networks**
  - input signals unidirectional from the input layer to the output layer.
  - all inputs and outputs are independent of each other.

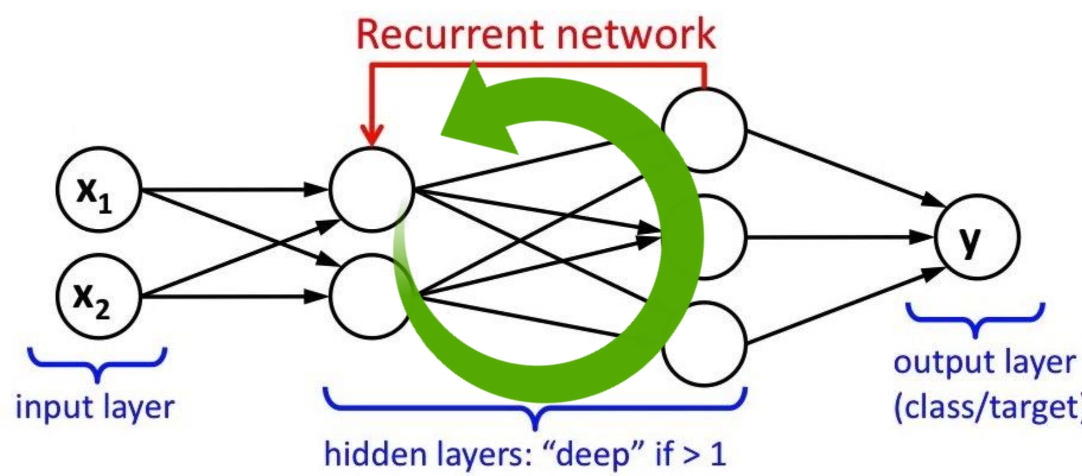
- **Main Challenges**
  - If we want to predict the next word in a sentence, we'd better know which words came before it. ➔ Traditional neural networks have no history information!
  - When we read a book, we understand each word on our understanding of previous words. We don't throw everything away and start thinking from scratch again. We have memory. ➔ Traditional neural networks has no memory!

Traditional neural network and Recurrent neural network

- **Traditional artificial neural networks: input signals unidirectional from the input layer to the output layer.**



- **Recurrent neural networks: a kind of neural network which will send current information back to itself. As a result, it has memory and can "remember" the history information.**



Recurrent Neural Networks (RNN)

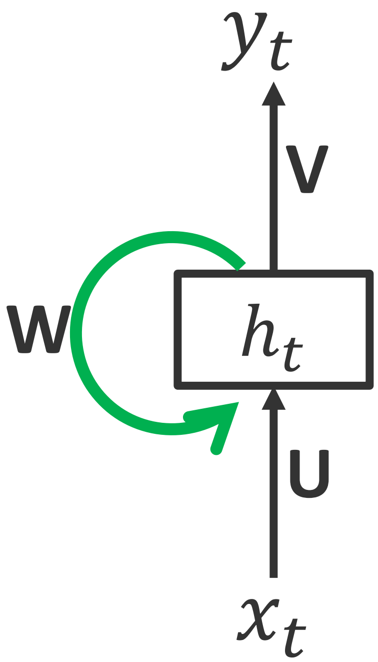
- **Main Features:**
  - **RNN allows signals to travel in both directions. This property mirrors more closely how a biological neural network works.**
  - **RNN has loops, allowing information to persist.**
  - **RNN has a “memory” which captures information about what has been calculated so far.**
  - **RNN can have a very long memory with variants of RNN.**

- **Application:**
  - **RNN is usually very good at predicting sequences or time series data. If your task is to predict a sequence or a periodic signal, then using a RNN might be a good starting point.**

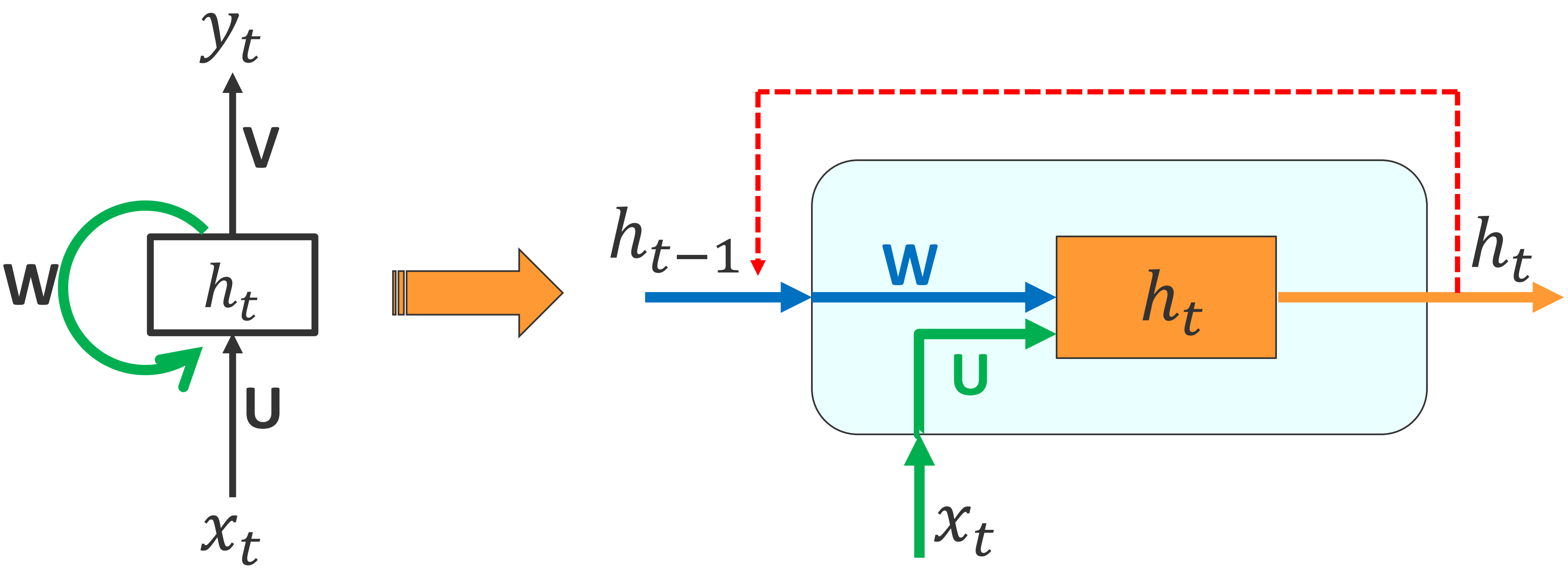
Recurrent Neural Networks: architecture (I)

Recurrent neural networks have

- two inputs at time steps **t (t=1,2,3…):**
  - the present input x_t
  - the hidden state h_t: here, we use "*state*" to refer to a set of values that summarizes all the information about the past behavior of the system. The hidden state h_t captures information about what happened in *all* the previous time steps. It is the memory of the network.
- Output y_t: two sources of input are combined to decide the output
- **U, V, W** weights: U for the input layer, W for the hidden layer and V for the output layer.



Recurrent Neural Networks: architecture (II)



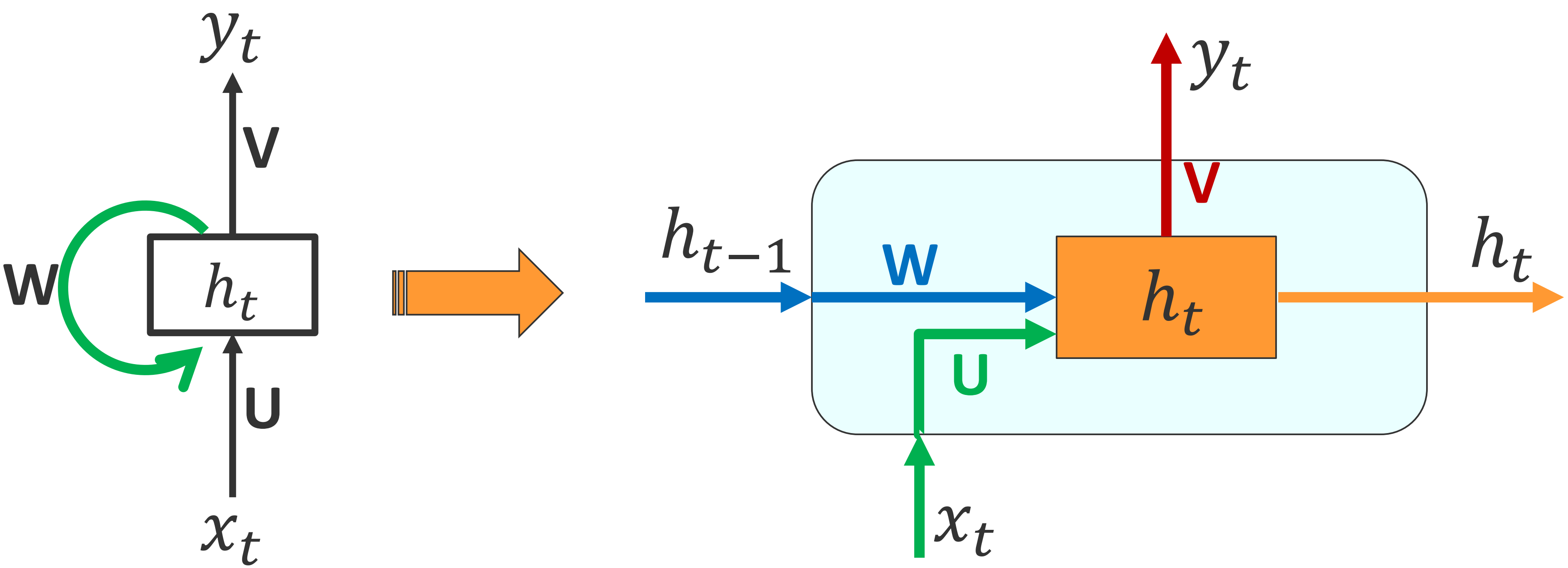
- Time recurrence is introduced by relating state h_t with its past state h_{t-1}. The hidden state is saved and can be used in the next time step.

- The hidden state h_t is calculated based on the previous hidden state h_{t-1} and the input x_t at the current time step.

$$h_t = f(Ux_t + Wh_{t-1});$$

where $f(.)$: activation function

Recurrent Neural Networks: architecture (III)

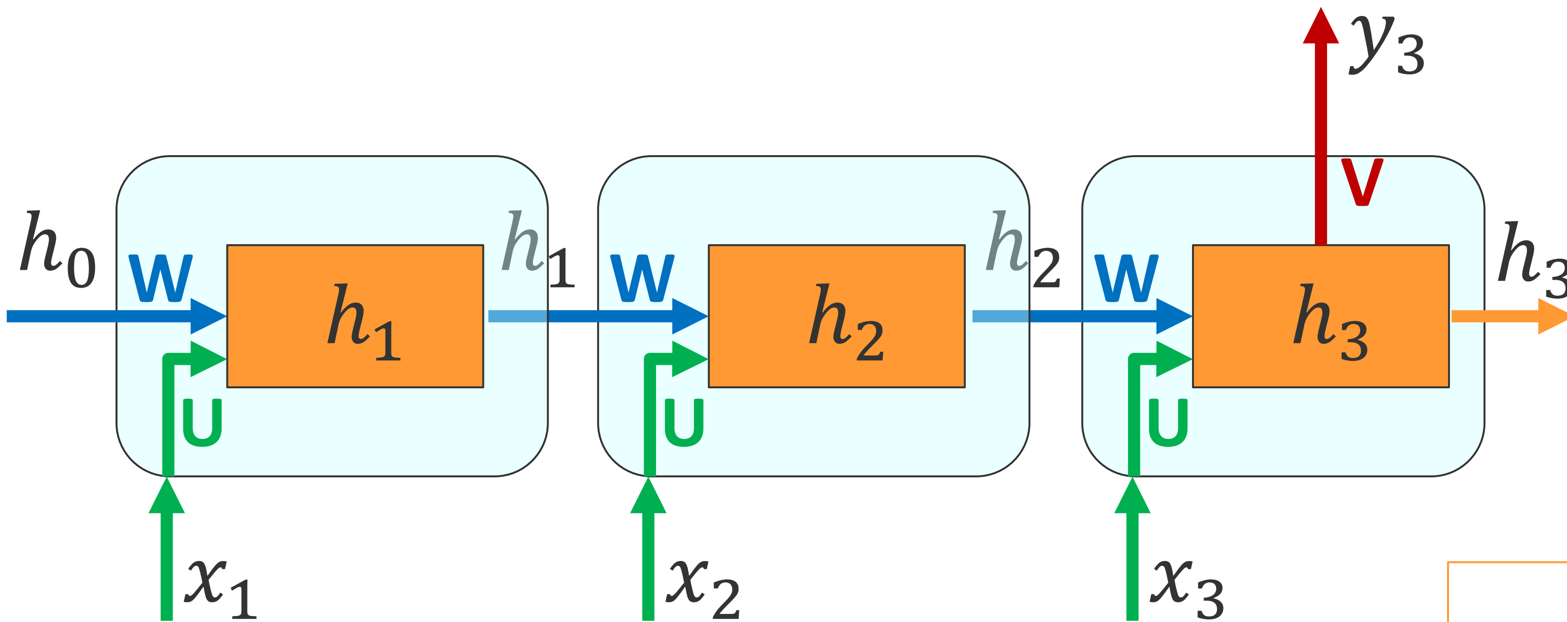


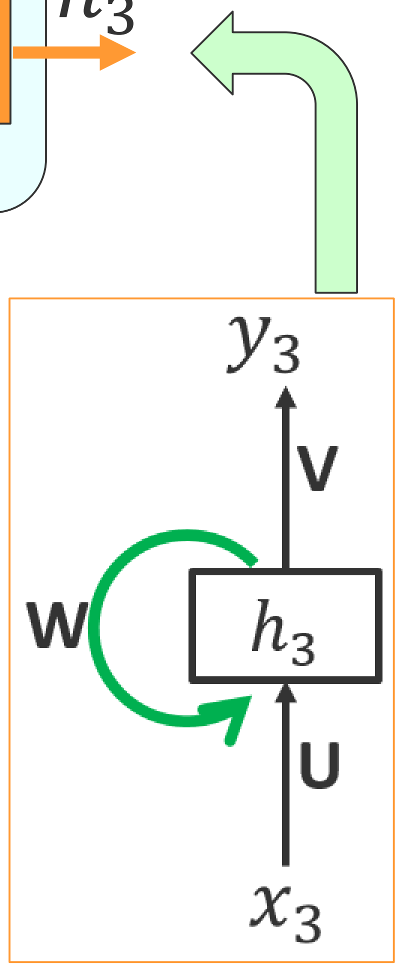
- The output $\boldsymbol{y_t}$ is calculated based on the memory $\boldsymbol{h_t}$ at time $\boldsymbol{t}$.

$$\boldsymbol{y_t = Vf(h_t)};$$

where $f(.)$: activation function

Unfold RNN when t=3 (I)





- The state h_3 depends on the input x_3 and the state h_2 in the previous time step.
- The output y_3 depends on the state h_3.Eventually, y_3 depends on all states (h_3, h_2, h_1, h_0)
- Unfolded RNN can be seen as deep neural network, except that the recurrent weights are same for all time steps.

Unfold RNN when t=3 (II)

$$y_3 = Vf(h_3)$$

$$h_3 = f(Ux_3 + Wh_2)$$

where $f(.)$: activation function

The output y_3 depends on h_3, which further depends on the input x_3 and the hidden state h_2 in the previous time step.

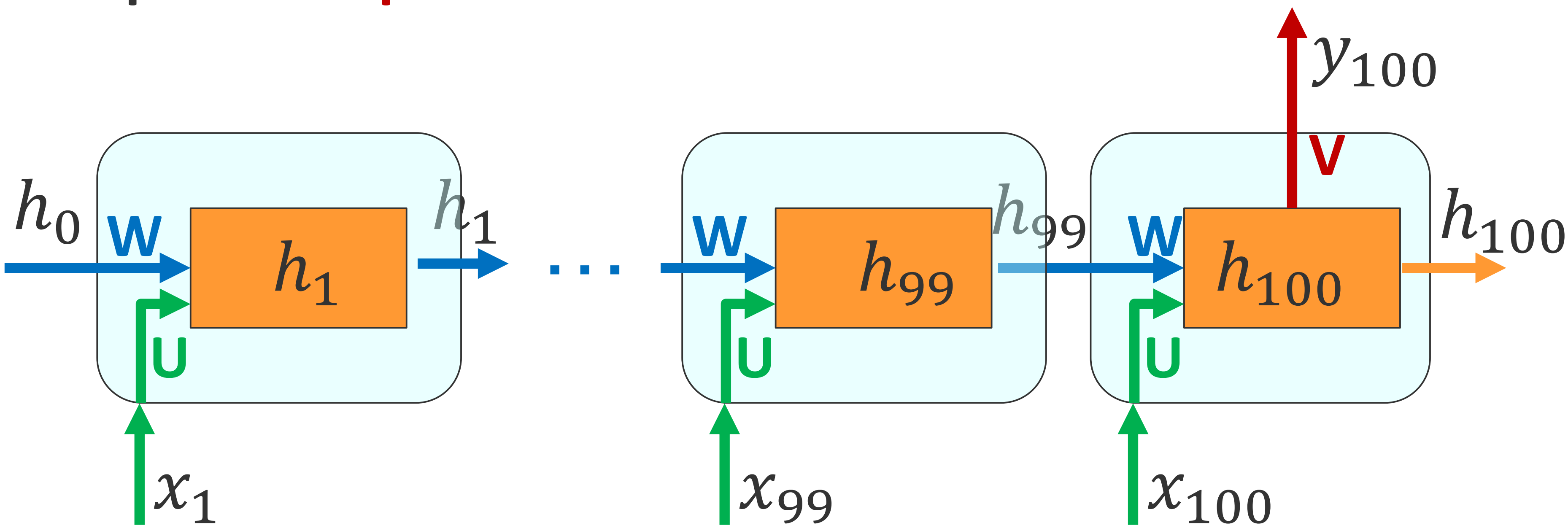
$$h_2 = f(Ux_2 + Wh_1)$$

h_2 further depends on the input x_2 and the hidden state h_1 in the previous time step.

$$h_1 = f(Ux_1 + Wh_0)$$

h1 further depends on both the input x1 and the hidden state in the previous time h0(=0)

RNNs connect previous information to the present time step: for deep RNN when t=100



$$h_{100} = f(Ux_{100} + Wh_{99})$$
$$y_{100} = Vf(h_{100})$$

→ h_{100} depends on both x_{100} and the hidden state in the previous epoch h_{99}

$$h_{99} = f(Ux_{99} + Wh_{98})$$

→ h_{99} depends on both x_{99} and the hidden state in the previous epoch h_{98}

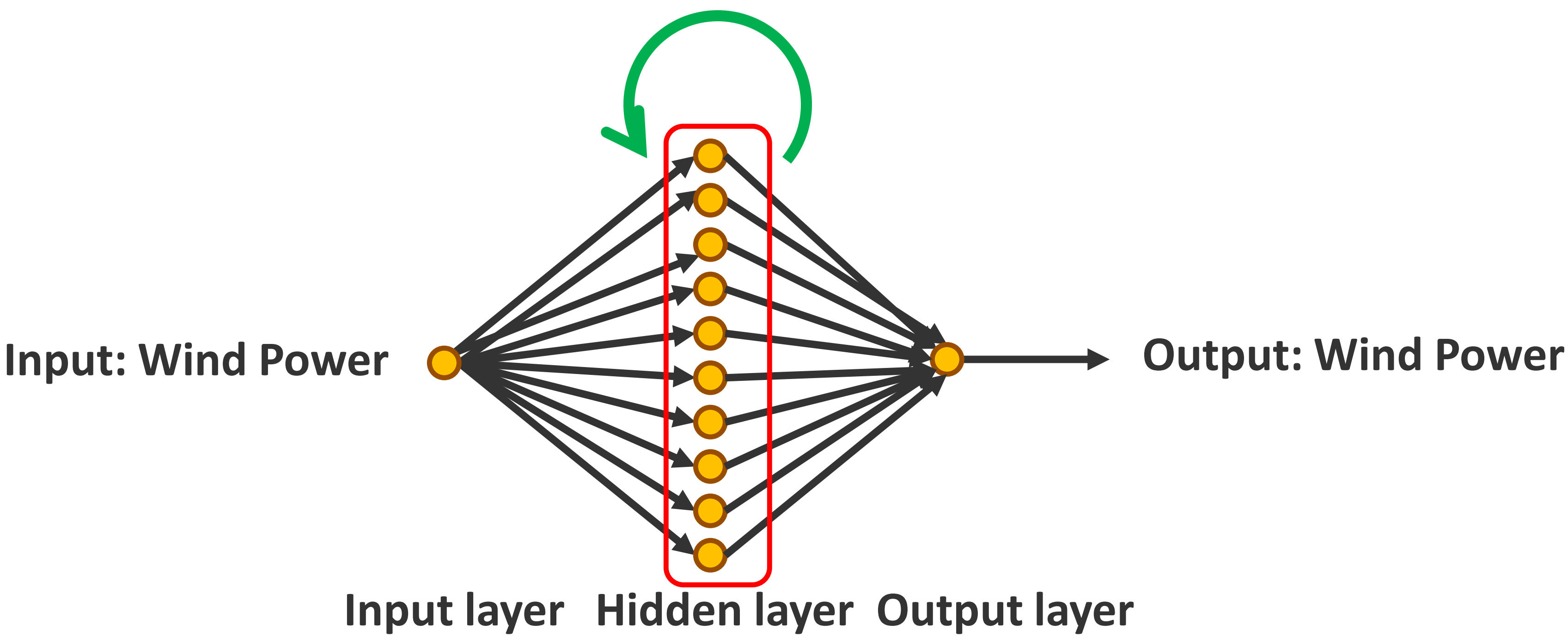
⋮

$$h_1 = f(Ux_1 + Wh_0)$$

→ h1 depends on both x1 and the hidden state in the previous time h0(=0)

Recurrent Neural Network for Wind Energy Forecasting

- One input node in the input layer
- One recurrent hidden layer, 10 hidden nodes in the hidden layer
- One output node
- Input: wind power; output: wind power → time series forecasting



Recurrent neural Networks for time series prediction

- The data is between 2012.01.01-2013.10.01, which has 15336 data records.
- We need to build the recurrent neural network model to find the predicted wind power generation in the last two weeks, e.g., 2013.09.17-2013.10.01.
- We use **2000 data records** as training data due to high computation load

X (input)	Y (output)
POWER_13001	**POWER_13002**
POWER_13002	**POWER_13003**
POWER_13003	**POWER_13004**
…	**…**
POWER_14999	**POWER_15000**
POWER_15000	**?**

training data

build a recurrent neural network model

test data

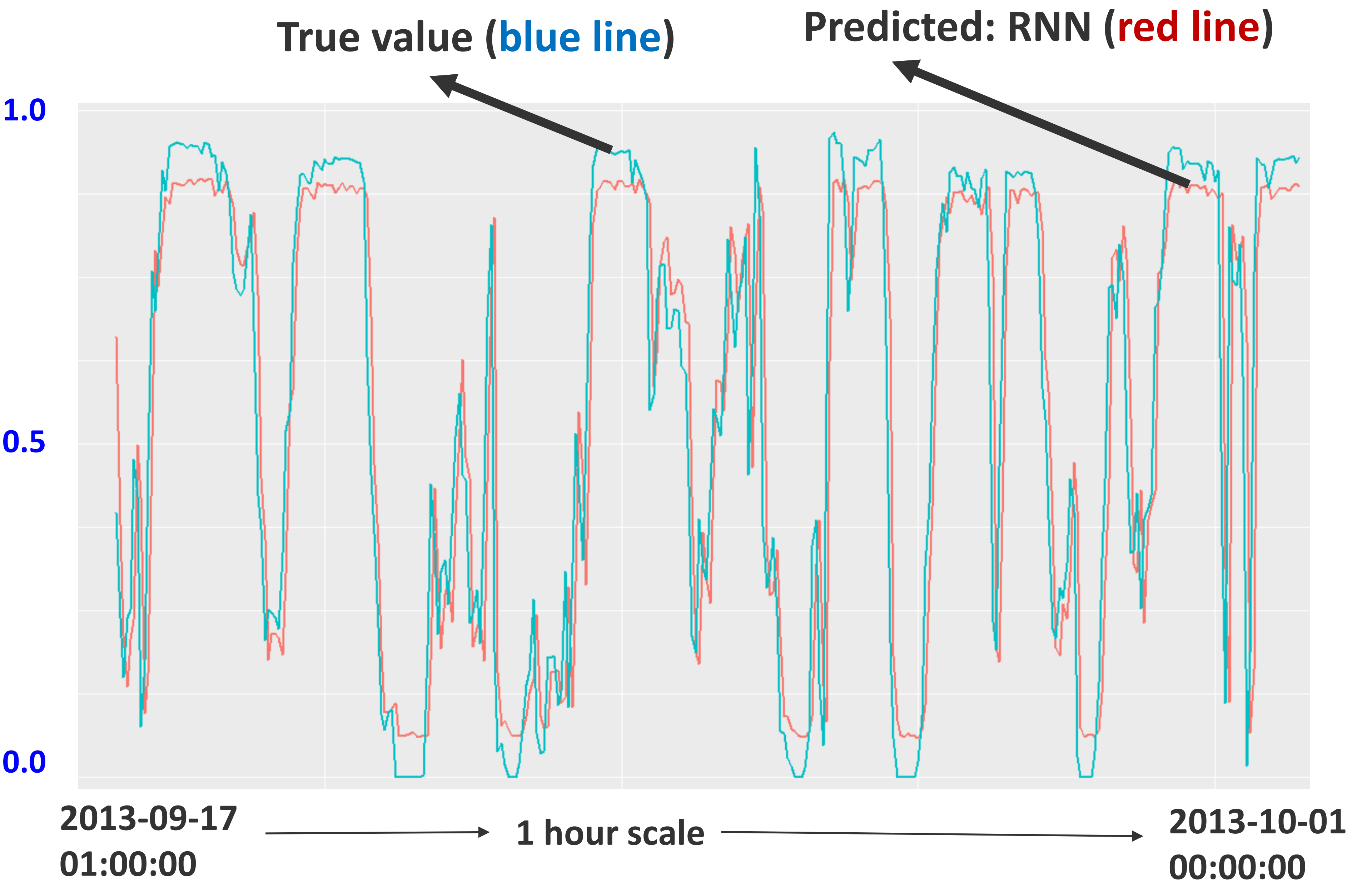
We can get the predicted result based on the input and the training model

R code to build recurrent neural network model

```r

#read CSV file
data <- read.csv("WindPowerDataAU.csv")
Power <- data$POWER

TotalDataLength <- length(Power)
TotalTrainLength <- 15000
ActualTrainLength <- 2000

#use x_i to predict x_(i+1)
X <- Power[(TotalTrainLength - ActualTrainLength) : (TotalTrainLength -1)]
X <- array(X,dim=c(NROW(X),NCOL(X),1))

Y <- Power[(TotalTrainLength - ActualTrainLength + 1) : TotalTrainLength]
Y <- array(Y,dim=c(NROW(Y),NCOL(Y),1))

# Train model.
model <- trainr(Y = Y, X = X, learningrate = 0.1, hidden_dim = 10, numepochs = 150)

```

Building training model

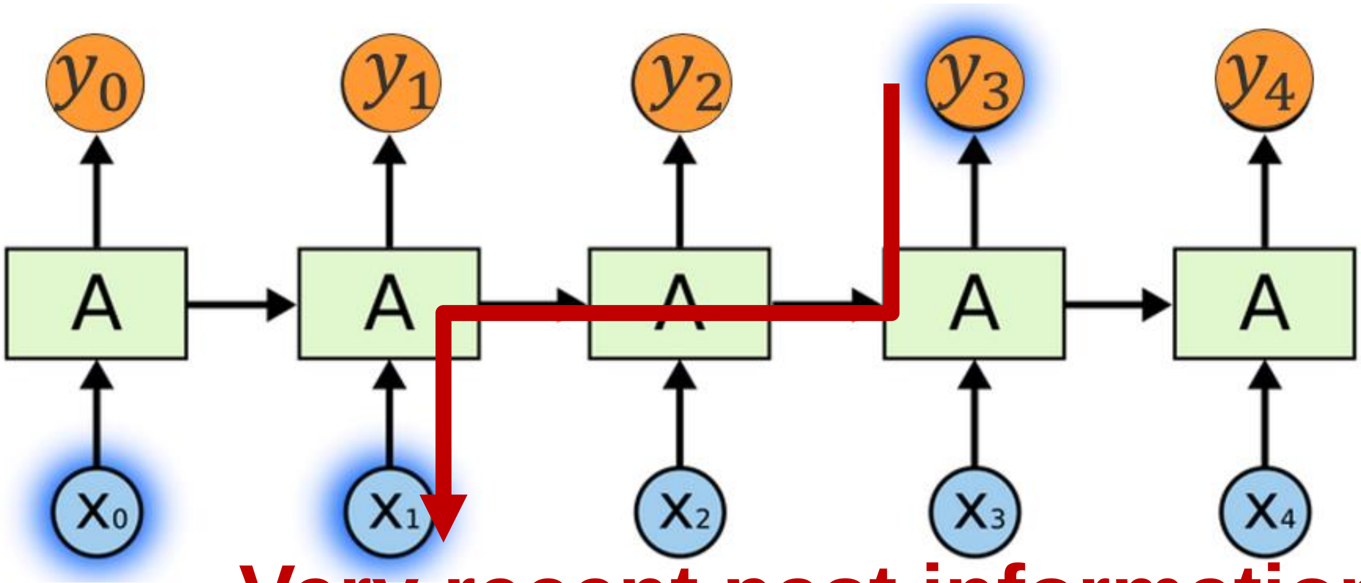
True value & Predicted Wind Power (RMSE = 0.14)



MORE CONSIDERATIONS…

Prediction with only recent previous information

- In theory, RNNs can make use of history information in arbitrarily long sequences, but in practice they may be limited to looking back a few steps
- To predict the last word of "The clouds are in the ________" we don't need any further context. It is obvious that the word is "sky"

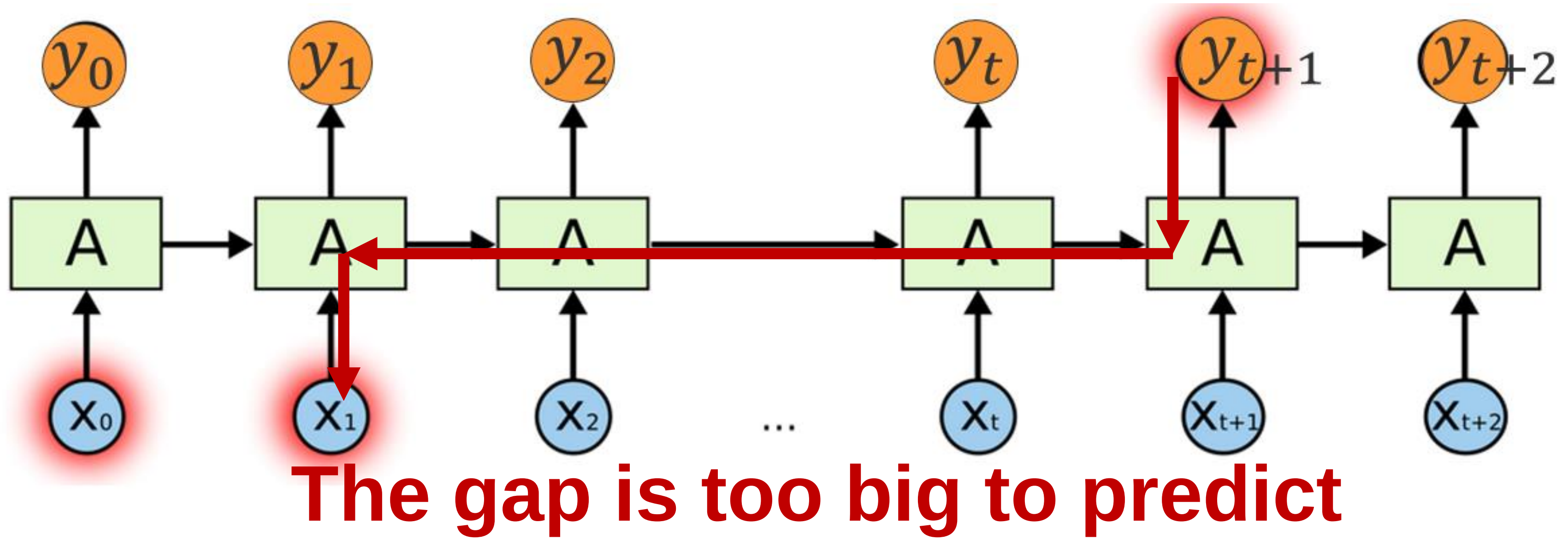


Very recent past information

Problem of Long-term dependency

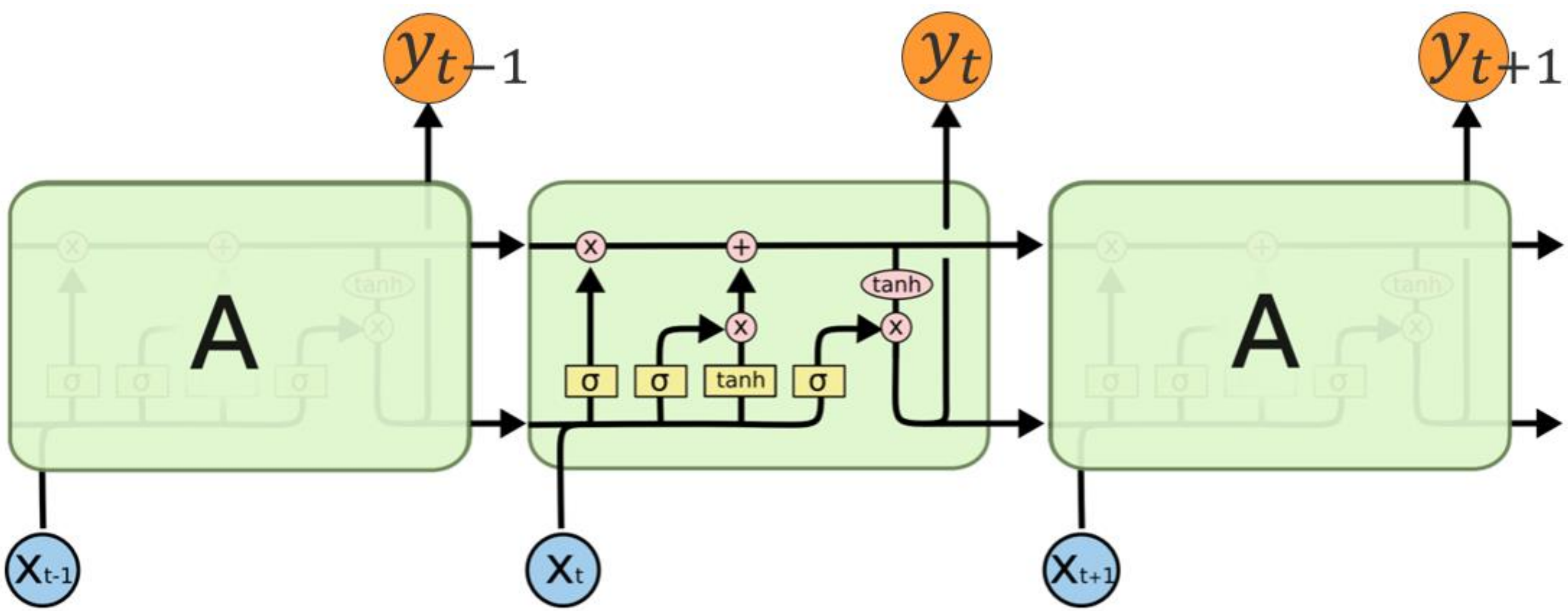
- We predict the last word in the sentence
  - "I grew up in Norway……I speak fluent ________". Using only recent information suggests that the last word is the name of a language. But, more distant past indicates that it is **Norwegian**.

- The gap between the relevant information and where it is needed is very large. As that gap grows, RNNs become unable to learn to connect the information.



Long Short Term Memory (LSTM)

- **Explicitly designed to avoid the long-term dependency problem.**
- **LSTM is one kind of the most promising variant of RNN. The main difference is the hidden layer. Some gates are introduced to help the neuron to choose when to forget and when to remember things.**



references

- **Recurrent Neural Network:** http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/
- **Long Short Term Memory (LSTM):** http://colah.github.io/posts/2015-08-Understanding-LSTMs/