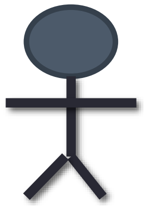
RNN LIBRARY IN R

Asst.Prof.Dr.Supakit Nootyaskool
IT-KMITL

Study map



- 1.Basic programming
  - R-programming
- 2. Perceptron
  - Activity function
- 3. Feed Forward NN
  - Logistic function
- 4. Feed Forward NN
  - XOR gate
  - Multi-layer perceptron
- 5. Example & Library Feed Forward NN
  - N:N, 1:N model
  - iris dataset
- 6. Writing NN Code
  - Data scaling, Confusion matrix
  - Writing NN code
- 7. Recurrent Neural Network
- 8. Apply RNN & Library
- 9. GRU LSTM
- 10. CNN
- 11 Apply GA to NN

Topic

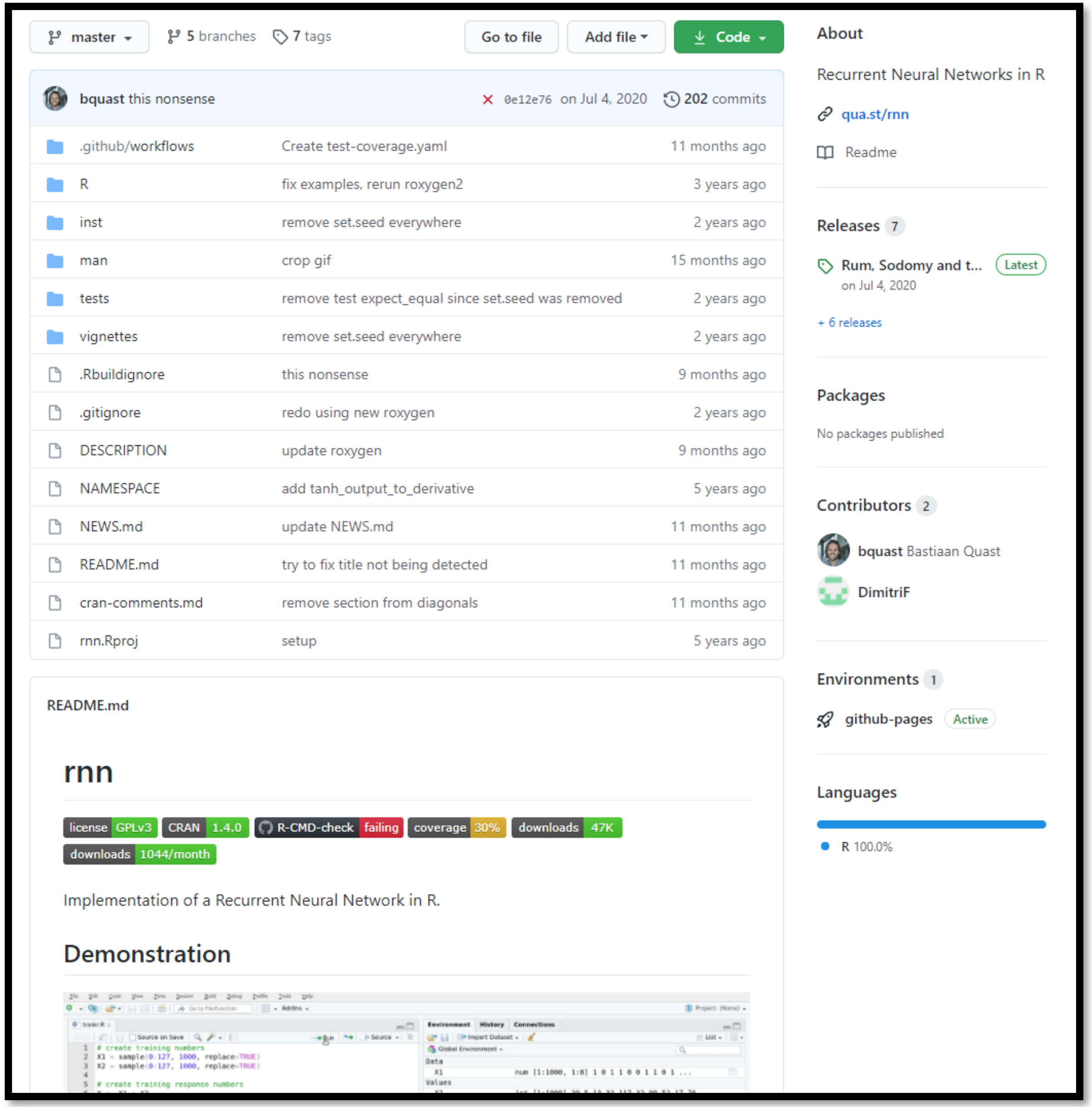
- RNN package
- Introduce RNN to Humidity Forecasting
- Changing data for training WeatherAUS dataset
- Understand 3D dimensions and setting RNN()
- Four inputs RNN
- Sum of Sine Waves

RNN PACKAGE

RNN package

Package features

- Native package in R (Development for R)
- https://github.com/bquast/rnn



`trainr(Y, X, ..)`

rnn

The variable X and Y are 3D array,

- dim 1 is sample
- dim 2 is time
- dim 3 is variables.

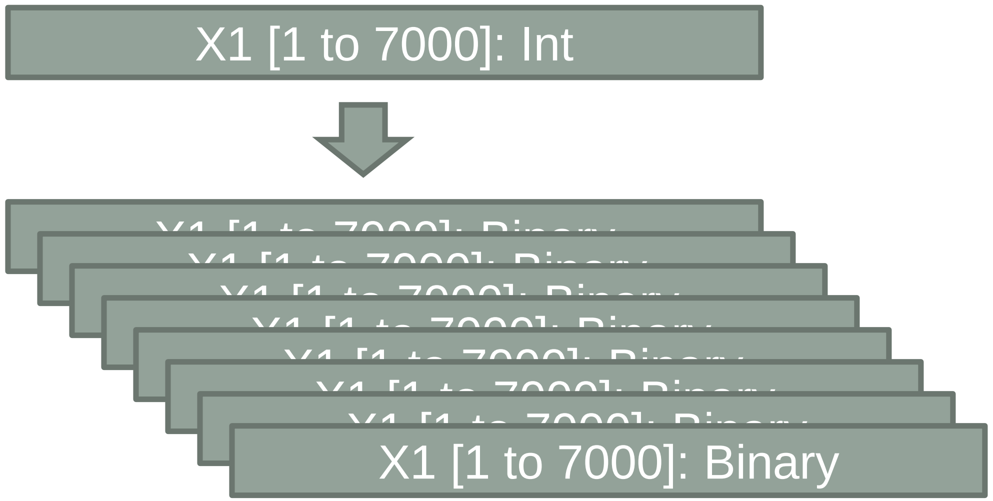
Install packages

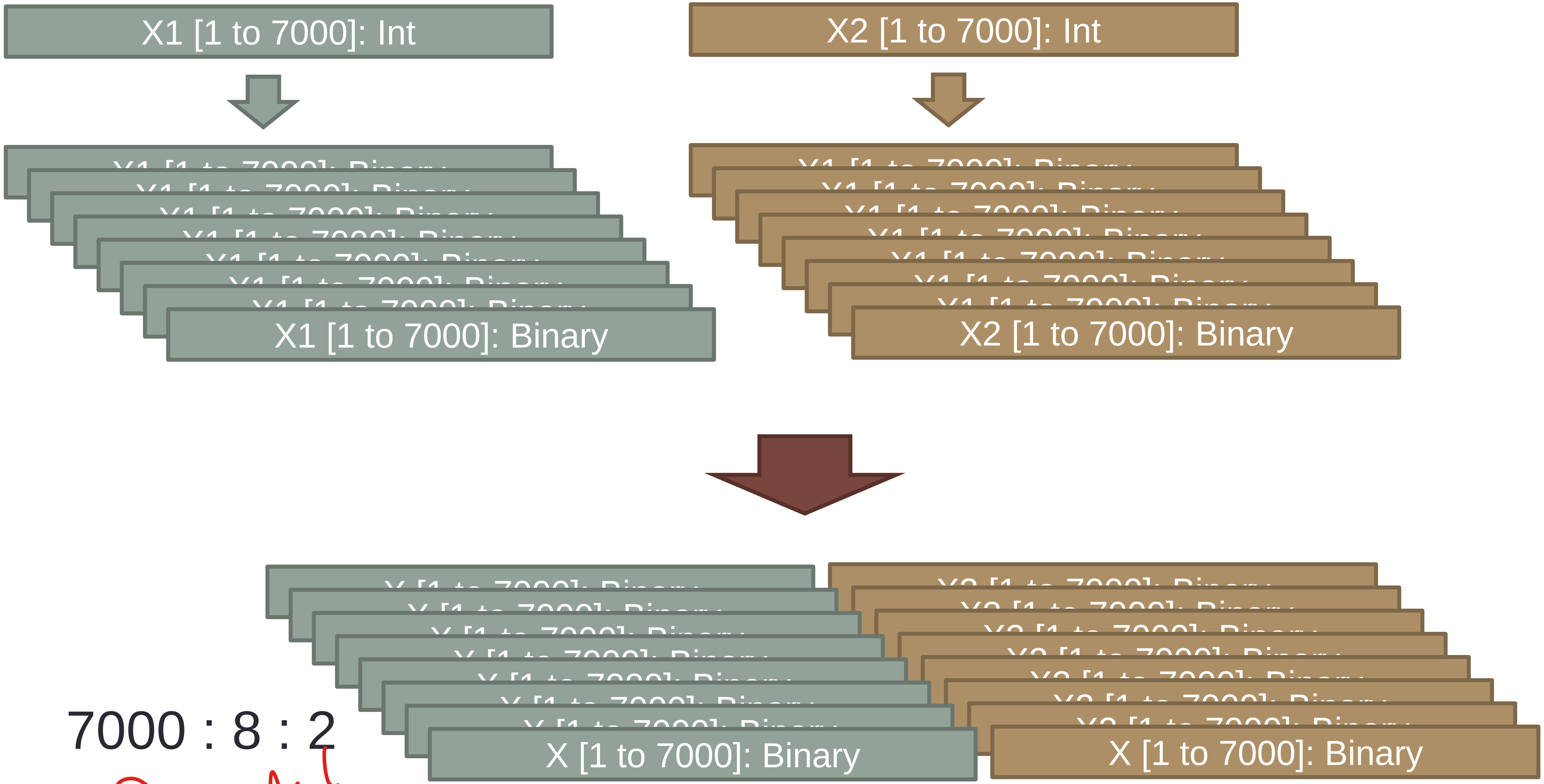
```
install.packages("rnn")

install.packages("digest")
```

Activity 8.1 Binary Addition

```
##############################################
#Activity8.1 Binary addition
##############################################
#install.packages("rnn")
library("rnn")
#Create a set of random numbers in X1 and X2
X1=sample(0:127, 7000, replace=TRUE)
X2=sample(0:127, 7000, replace=TRUE)

#Create training response numbers
Y=X1 + X2

# Convert to binary
X1=int2bin(X1)
X2=int2bin(X2)
Y=int2bin(Y)

# Create 3d array: dim 1: samples; dim 2: time; dim
3: variables.
X=array( c(X1,X2), dim=c(dim(X1),2) )

dim(X)    #Show dimension
head(X)   #Print header of the array
```

```
# Train the model
model <- trainr(Y=Y[,dim(Y)[2]:1],
                X=X[,dim(X)[2]:1,],
                learningrate = 0.1,
                hidden_dim = 10,
                batch_size = 100,
                numepochs = 100)
```





7000 : 8 : 2

ผลิตการใช้ข้อมูล ซ้อน ๆ มิติ

Activity 8.1 Binary Addition

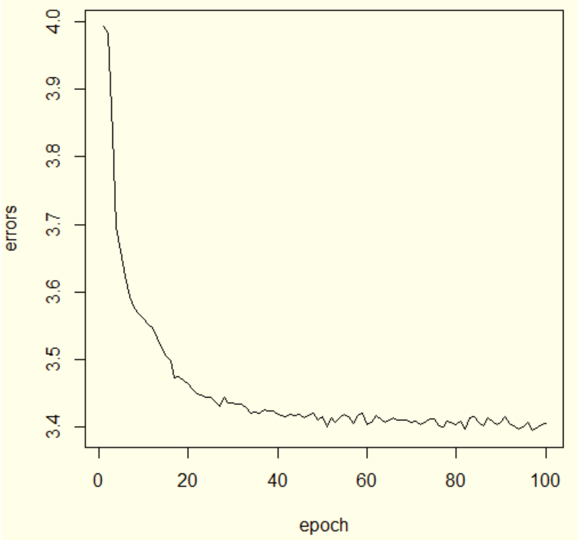
```
############################################
#Activity8.1 Binary addition
############################################
#install.packages("rnn")
library("rnn")
#Create a set of random numbers in X1 and X2
X1=sample(0:127, 7000, replace=TRUE)
X2=sample(0:127, 7000, replace=TRUE)

#Create training response numbers
Y=X1 + X2

# Convert to binary
X1=int2bin(X1)
X2=int2bin(X2)
Y=int2bin(Y)

# Create 3d array: dim 1: samples; dim 2: time; dim 3: variables.
X=array( c(X1,X2), dim=c(dim(X1),2) )

dim(X)   #Show dimension
head(X)  #Print header of the array
```

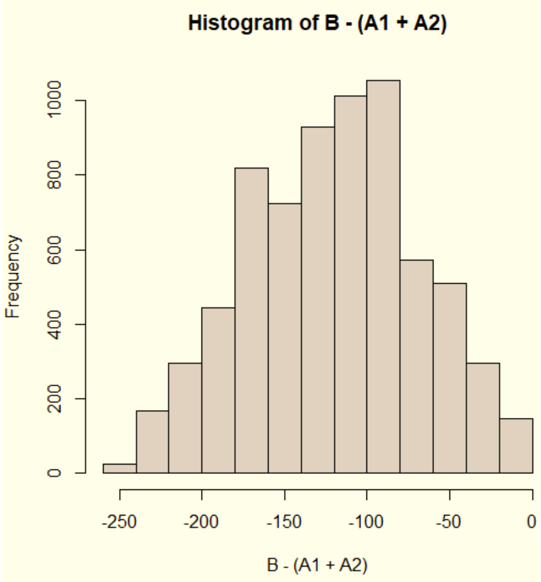
```
# Train the model
model <- trainr(Y=Y[,dim(Y)[2]:1],
                X=X[,dim(X)[2]:1,],
                learningrate = 0.1,
                hidden_dim = 10,
                batch_size = 100,
                numepochs = 100)
```

```
Trained epoch: 94 - Learning rate: 0.1
Epoch error: 3.3968492897109
Trained epoch: 95 - Learning rate: 0.1
Epoch error: 3.40042502883762
Trained epoch: 96 - Learning rate: 0.1
Epoch error: 3.40746443050068
Trained epoch: 97 - Learning rate: 0.1
Epoch error: 3.39498673612038
Trained epoch: 98 - Learning rate: 0.1
Epoch error: 3.39945470008261
Trained epoch: 99 - Learning rate: 0.1
Epoch error: 3.40408891327449
Trained epoch: 100 - Learning rate: 0.1
```

```
plot(colMeans(model$error),type='l',xlab='epoch',ylab='errors')

# Create test inputs
A1=int2bin(sample(0:127, 7000, replace=TRUE))
A2=int2bin(sample(0:127, 7000, replace=TRUE))

# Create 3d array: dim 1: samples; dim 2: time; dim 3: variables
A=array( c(A1,A2), dim=c(dim(A1),2) )

# Now, let us run prediction for new A
B = predictr(model,
        A[,dim(A)[2]:1,] )
        B=B[,dim(B)[2]:1]
```

```
# Convert back to integers
A1=bin2int(A1)
A2=bin2int(A2)
B=bin2int(B)

# Plot the differences as histogram
hist( B-(A1+A2) )
```





INTRODUCE RNN TO HUMIDITY FORECASTING

ปัญหา จรอง

Humidity Forecasting ค.ชื้นในอากาศ

- To learn prediction data with RNN
- Data from Australian weather stations.
- Large data set by 145460 observations from 45 stations
- The source dataset is copyrighted
- A CSV version of this dataset is at https://rattle.togaware.com/weatherAUS.csv app. 21MB

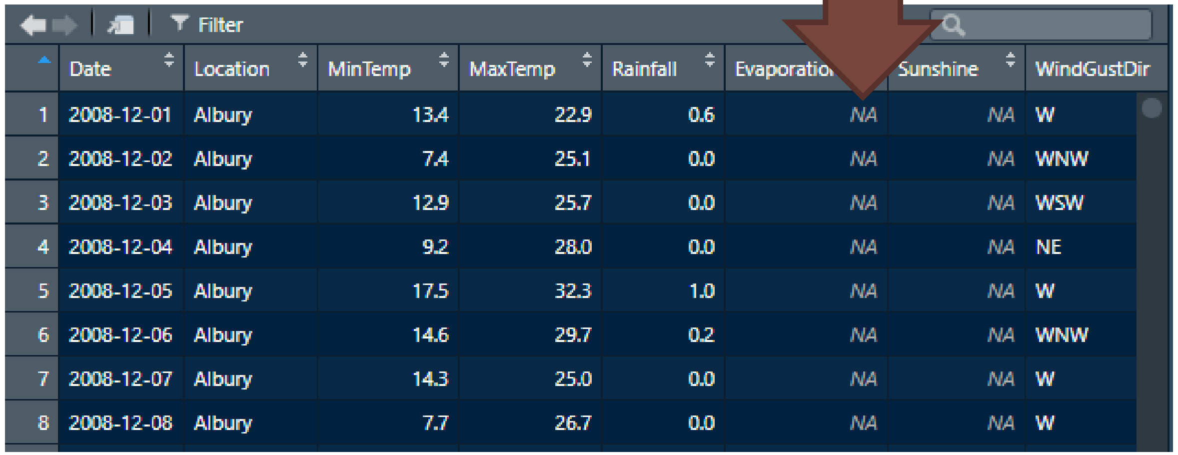
- Date: The date of observation (a Date object).
- Location: The common name of the location of the weather station.
- MinTemp: The minimum temperature in degrees Celsius.
- MaxTemp: The maximum temperature in degrees Celsius.
- Rainfall: The amount of rainfall recorded for the day in mm.
- Evaporation: The so-called class a pan evaporation (mm) in the 24 hours to 9 am.

- Sunshine: The number of hours of bright sunshine in the day.
- WindGustDir: The direction of the strongest wind gust in the 24 hours to midnight.
- WindGustSpeed: The speed (km/h) of the strongest wind gust in the 24 hours to midnight.
- Temp9am: Temperature (degrees C) at 9 a.m.
- RelHumid9am: Relative humidity (percent) at 9 a.m.
- Cloud9am: Fraction of the sky obscured by clouds at 9 a.m. This is measured in oktas, which are a unit of eighths. It records how many eighths of the sky are obscured by cloud. A zero measure indicates completely clear sky whilst an 8 indicates that it is completely overcast.
- WindSpeed9am: Wind speed (km/hr) averaged over 10 minutes prior to 9 a.m. 6 weatherAUS.
- Pressure9am: Atmospheric pressure (hpa) reduced to mean sea level at 9 a.m.
- Temp3pm: Temperature (degrees C) at 3 p.m.
- RelHumid3pm: Relative humidity (percent) at 3 p.m.
- Cloud3pm: Fraction of sky obscured by cloud (in oktas: eighths) at 3 p.m.
- WindSpeed3pm: Wind speed (km/hr) averaged over 10 minutes prior to 3 p.m.
- Pressure3pm: Atmospheric pressure (hpa) reduced to mean sea level at 3 p.m.

- ChangeTemp: Change in temperature.
- ChangeTempDir: Direction of change in temperature.
- ChangeTempMag: Magnitude of change in temperature.
- ChangeWindDirect: Direction of wind change.
- MaxWindPeriod: Period of maximum wind.
- RainToday: Integer 1 if precipitation (mm) in the 24 hours to 9 a.m. exceeds 1 mm, and 0 otherwise.
- TempRange: Difference between minimum and maximum temperatures (degrees C) in the 24 hours to 9 a.m.
- PressureChange: Change in pressure.
- RISK_MM: The amount of rain. A kind of measure of the risk.
- RainTomorrow: The target variable. Will it rain tomorrow?

Activity 8.2 Humidity forecasting with RNN

```
###################################################
## Activity 8.2 Humidity forecasting with RNNs
###################################################

library("rattle.data")
library("rnn")

data(weatherAUS)
View(weatherAUS)

#extract only 1 and 14 clumn and first 3040 rows
(Albury location)
data=weatherAUS[1:3040,c(1,14)]
summary(data)

data_cleaned = na.omit(data)
data_used=data_cleaned[1:3000,]

x = data_cleaned[,1]
y = data_cleaned[,2]
```



	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporatio	Sunshine	WindGustDir
1	2008-12-01	Albury	13.4	22.9	0.6	NA	NA	W
2	2008-12-02	Albury	7.4	25.1	0.0	NA	NA	WNW
3	2008-12-03	Albury	12.9	25.7	0.0	NA	NA	WSW
4	2008-12-04	Albury	9.2	28.0	0.0	NA	NA	NE
5	2008-12-05	Albury	17.5	32.3	1.0	NA	NA	W
6	2008-12-06	Albury	14.6	29.7	0.2	NA	NA	WNW
7	2008-12-07	Albury	14.3	25.0	0.0	NA	NA	W
8	2008-12-08	Albury	7.7	26.7	0.0	NA	NA	W

- A not available data (NA) or (N/A) is an empty data.
- 5 / 0 = Infinity
- 5 / NA = NA
- 5 + NA = NA

na.omit(obj)

stats

- Removing data at a row with NA.

```
DF = data.frame(x = c(1, 2, 3), y = c(0, 10, NA))

print(DF)    #having a NA value

na.omit(DF)

print(DF)    #The DF had removed NA value.
```

```
> DF
  x  y
1 1  0
2 2 10
3 3 NA
> na.omit(DF)
  x  y
1 1  0
2 2 10
```

is.na(obj)

stats

- Check NA in an object or data

```
data = c(4, 8, 12, NA, 99, - 20, NA)

is.na(data) #found NA returns true.

which(is.na(data)) #show NA locations

### [1] 4 7
```

Activity 8.2 Humidity forecasting with RNN

```
#################################################
## Activity 8.2 Humidity forecasting with RNNs
#################################################

library("rattle.data")
library("rnn")

data(weatherAUS)
View(weatherAUS)

#extract only 1 and 14 clumn and first 3040 rows
(Albury location)
data=weatherAUS[1:3040,c(1,14)]
summary(data)

data_cleaned = na.omit(data)
data_selected = data_cleaned[1:3000,]

x = data_selected[,1]
y = data_selected[,2]
```

Filter

	Date	Location	MinTemp	MaxTemp	Rainfall	Evaporatio	Sunshine	WindGustDir
1	2008-12-01	Albury	13.4	22.9	0.6	NA	NA	W
2	2008-12-02	Albury	7.4	25.1	0.0	NA	NA	WNW
3	2008-12-03	Albury	12.9	25.7	0.0	NA	NA	WSW
4	2008-12-04	Albury	9.2	28.0	0.0	NA	NA	NE
5	2008-12-05	Albury	17.5	32.3	1.0	NA	NA	W
6	2008-12-06	Albury	14.6	29.7	0.2	NA	NA	WNW
7	2008-12-07	Albury	14.3	25.0	0.0	NA	NA	W
8	2008-12-08	Albury	7.7	26.7	0.0	NA	NA	W

- A not available data (NA) or (N/A) is an empty data.
- 5 / 0 = Infinity
- 5 / NA = NA
- 5 + NA = NA

```
X = matrix(x, nrow = 30)
Y = matrix(y, nrow = 30)

print(X)

# Standardize in the interval 0 to 1
Yscaled = (Y - min(Y)) / (max(Y) -
min(Y))
Y=t(Yscaled)

train=1:70
test=71:100
```

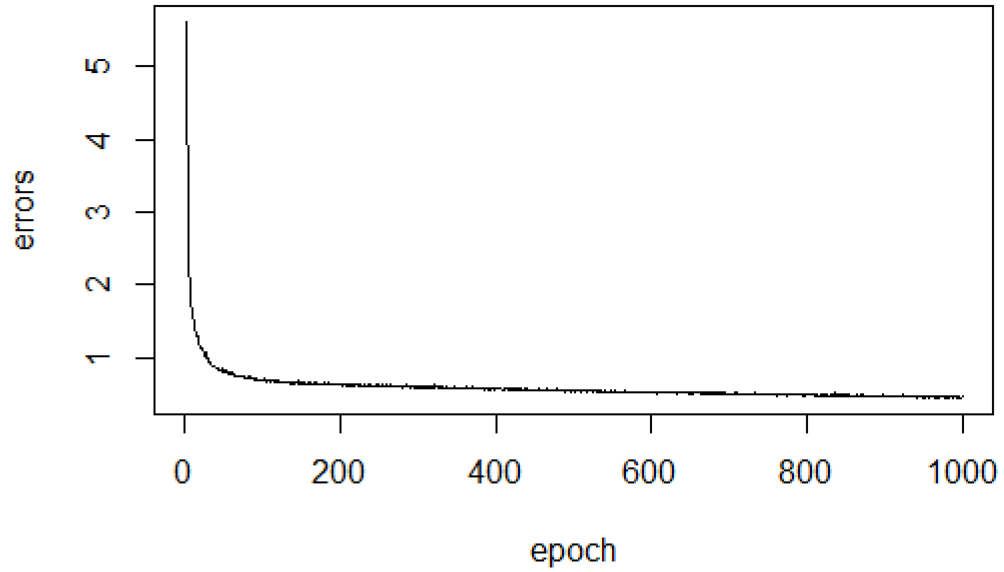
70%

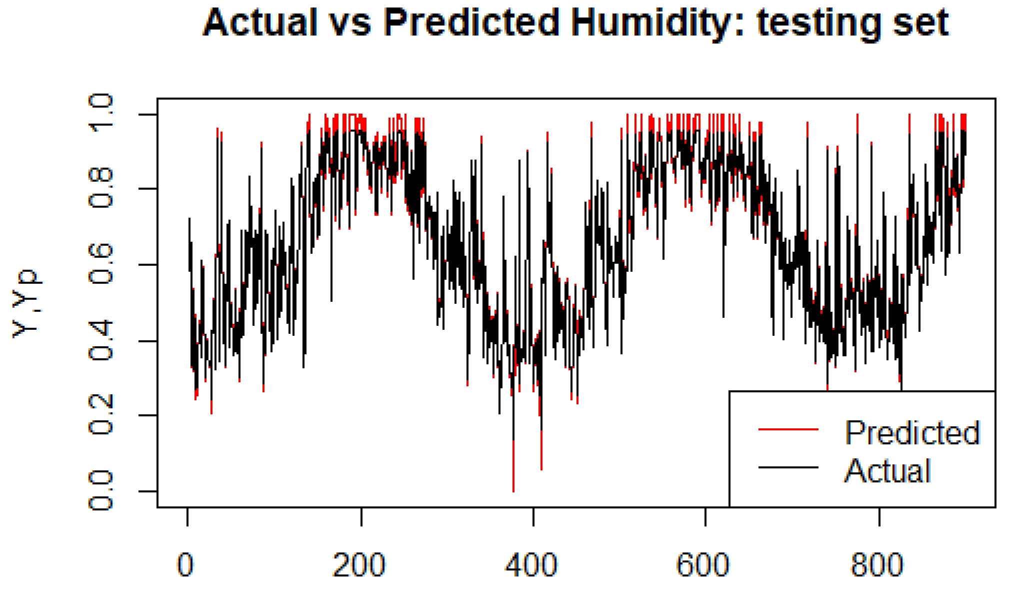
30%

```
model <- trainr(Y = Y[train,],
                X = Y[train,],
                learningrate = 0.05,
                hidden_dim = 16,
                numepochs = 1000)
```

```
Trained epoch: 994 - Learning rate: 0.05
Epoch error: 0.455020451068391
Trained epoch: 995 - Learning rate: 0.05
Epoch error: 0.457130585028692
Trained epoch: 996 - Learning rate: 0.05
Epoch error: 0.460447184841569
Trained epoch: 997 - Learning rate: 0.05
Epoch error: 0.459342413686247
Trained epoch: 998 - Learning rate: 0.05
Epoch error: 0.448618045460613
Trained epoch: 999 - Learning rate: 0.05
Epoch error: 0.453310520494415
Trained epoch: 1000 - Learning rate: 0.05
Epoch error: 0.463290460033698
>
```

```
plot(colMeans(model$error),type='l',xlab='epo
ch',ylab='errors')

Yp <- predictr(model, Y[test,])

plot(as.vector(t(Y[test,])), col = 'red',
type='l',
     main = "Actual vs Predicted Humidity:
testing set",
     ylab = "Y,Yp")
lines(as.vector(t(Yp)), type = 'l', col =
'black')
legend("bottomright", c("Predicted",
"Actual"),
       col = c("red","black"),
       lty = c(1,1), lwd = c(1,1))
```



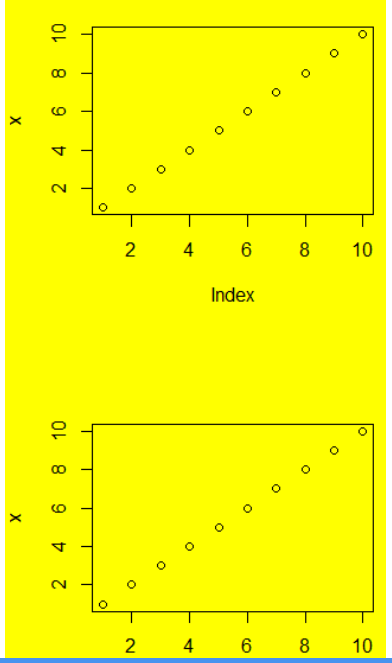


par(mfrow=c(2,3))

graphics

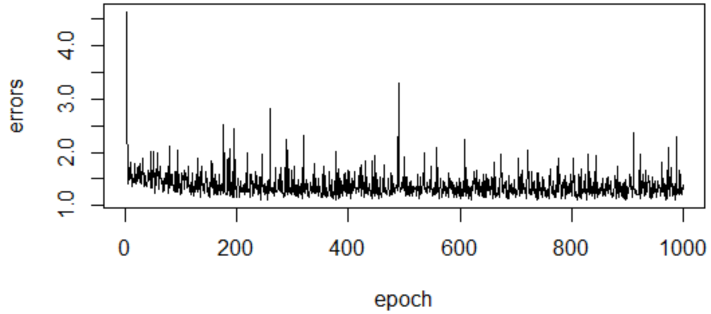
- Setting the query of graphical parameters
- Many parameters for setting pls look at the par manual by (?par).

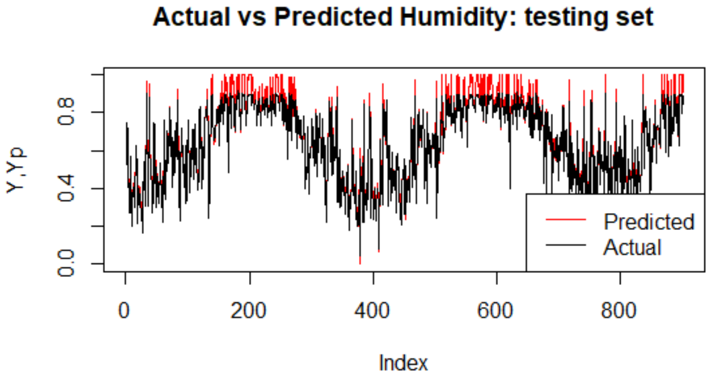
```
par(mfrow = c(2,3)) #place plot 2rows 3columns

par(bg = "yellow") #change background color

# make labels and margins smaller
par(cex=0.7, mai=c(0.1,0.1,0.2,0.1))
```



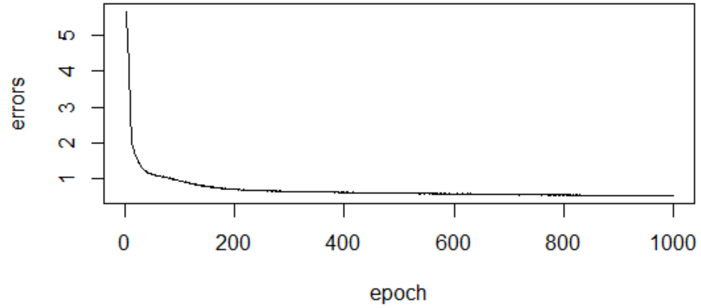
Activity 8.3 Adjustment parameter from 8.2

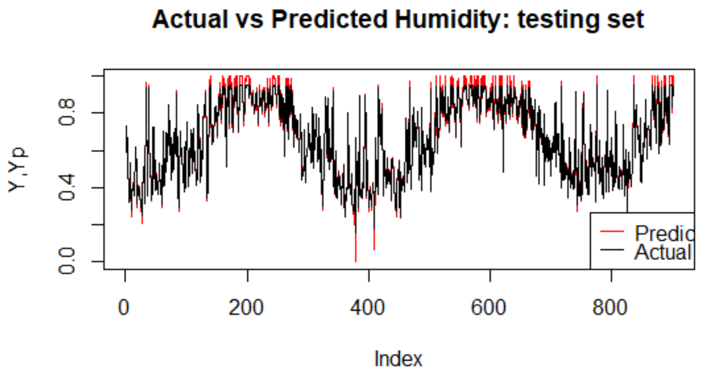
```
model <- trainr(Y = Y[train,],
                X = Y[train,],
                learningrate = 0.5,
                hidden_dim = 2,
                numepochs = 1000)
par(mfrow=c(2,1))
```



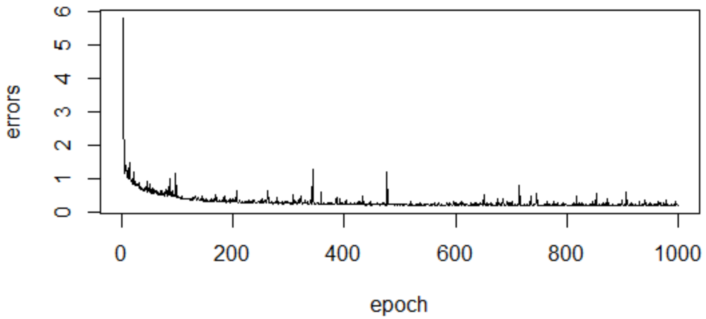


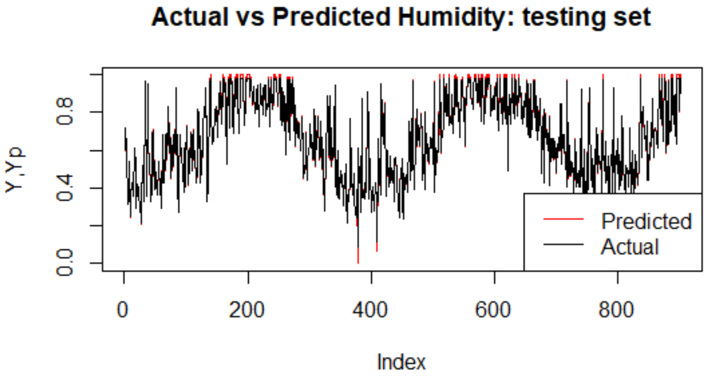

```
model <- trainr(Y = Y[train,],
                X = Y[train,],
                learningrate = 0.05,
                hidden_dim = 5,
                numepochs = 1000)
par(mfrow=c(2,1))
```






```
model <- trainr(Y = Y[train,],
                X = Y[train,],
                learningrate = 0.5,
                hidden_dim = 10,
                numepochs = 1000)
par(mfrow=c(2,1))
```





setDTthreads — Single core ทำให้รันได้หลายๆ core

data.table

- Set and get number of threads

i9 ≈ 10 core
20 threads

```
library("data.table")

getDTthreads()    #get number of threads

setDTthreads(4)   #set using 4 threads
```

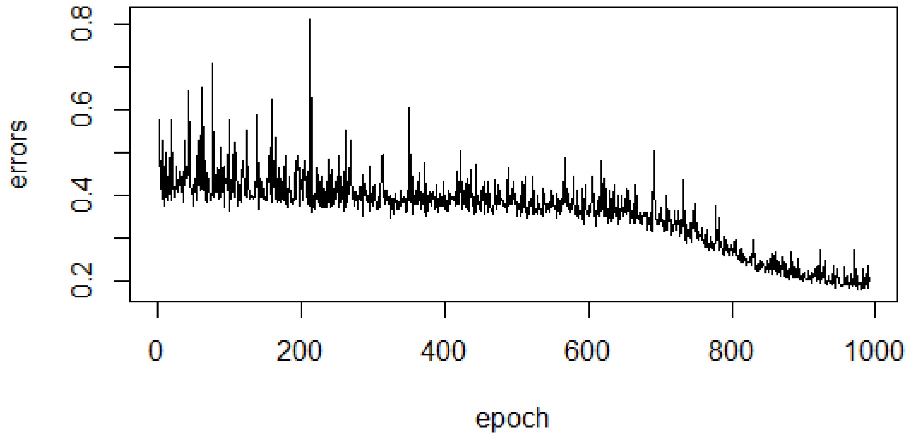
เปลี่ยน Data set

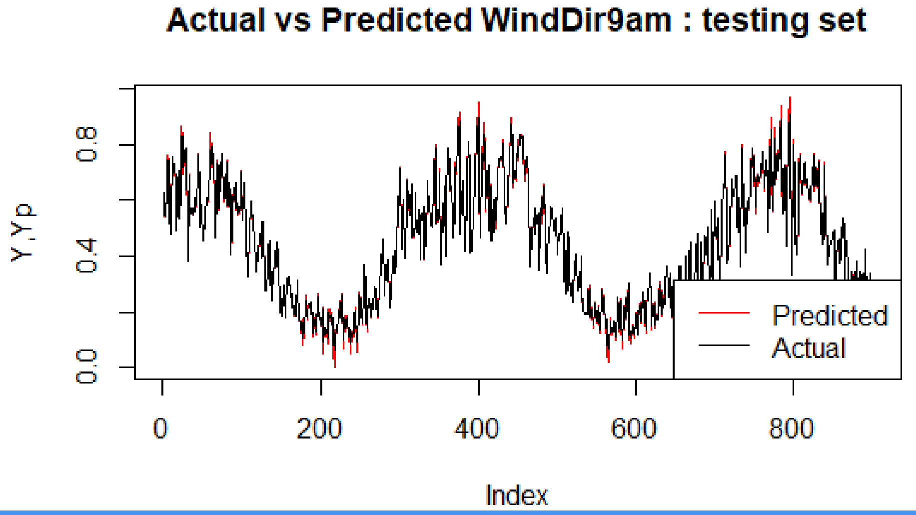
CHANGING DATA FOR TRAINING WEATHERAUS DATASET

Activity 8.4 Change data train in weatherAUS

```r
setDTthreads(4, percent = 90)
names(weatherAUS)#show list of data name
data=weatherAUS[1:3040,c(1,10)]
summary(data)
```

```r
par(mfrow=c(2,1))
plot(colMeans(model$error[,10:1000]),type='l',xlab='epoch'
     ,ylab='errors')
Yp <- predictr(model, Y[test,])
plot(as.vector(t(Y[test,])), col = 'red', type='l',
     main = "Actual vs Predicted WindDir9am : testing set",
     ylab = "Y,Yp")
lines(as.vector(t(Yp)), type = 'l', col = 'black')
legend("bottomright", c("Predicted", "Actual"),
       col = c("red","black"),
       lty = c(1,1), lwd = c(1,1))
```





Activity 8.5 Two inputs RNN

```
> names(weatherAUS)
 [1] "Date"          "Location"      "MinTemp"       "MaxTemp"
 [5] "Rainfall"      "Evaporation"   "Sunshine"      "WindGustDir"
 [9] "WindGustSpeed" "WindDir9am"    "WindDir3pm"    "WindSpeed9am"
[13] "WindSpeed3pm"  "Humidity9am"   "Humidity3pm"   "Pressure9am"
[17] "Pressure3pm"   "Cloud9am"      "Cloud3pm"      "Temp9am"
[21] "Temp3pm"       "RainToday"     "RISK_MM"       "RainTomorrow"
```

```
####################################################
## Activity 8.5 Changing data train in weatherAUS
## Prediction rainfall with data from  WindGuestSpeed
## and Humidity9am
####################################################
rm(list=ls())#clear all old data

library("rattle.data")
library("rnn")
library("data.table")

# Standardize in the interval 0 - 1
std0to1 = function(v)
{
        r= (v - min(v)) / (max(v) - min(v))
        return(r)
}

data(weatherAUS)
#View(weatherAUS)
```

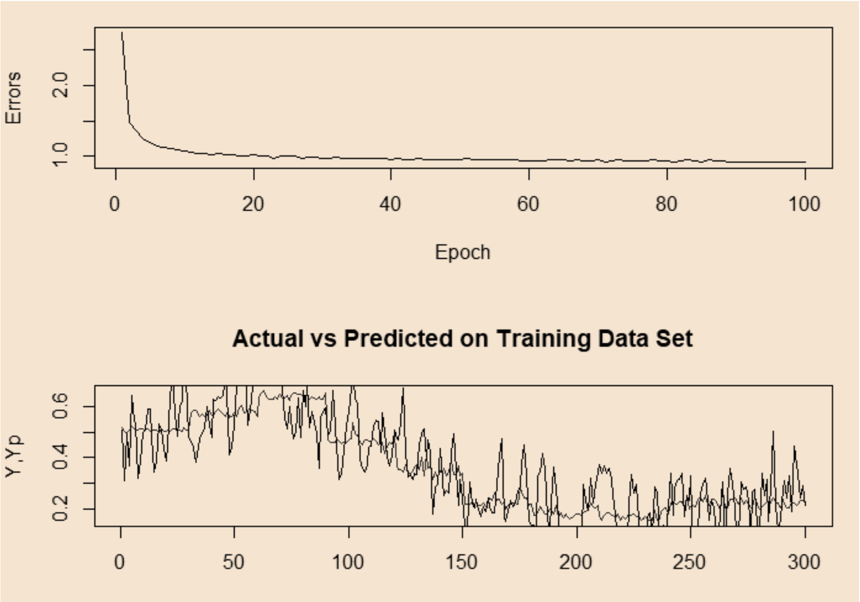
```
setDTthreads(3)
names(weatherAUS)#show list of data name

head(weatherAUS)
data=weatherAUS[1:400,c(3,14,15)]
summary(data)

data_cleaned = na.omit(data)

y = data_cleaned[1:300,1]
x1 = data_cleaned[1:300,2]
x2 = data_cleaned[1:300,3]


#convert data from the large range to 0..1
y = std0to1(y)
x1 = std0to1(x1)
x2 = std0to1(x2)
```

```r
tim = 10
sam = length(x1)/tim
X1 = array(x1,c(sam,tim))
X2 = array(x2,c(sam,tim))
Y =  array(y,c(sam,tim))

# create 3d array: dim 1: samples; dim 2: time; dim 3: variables
Xt <- array( c(X1,X2), dim=c(dim(X1),2) )
Yt <- array( c(Y,Y), dim=c(dim(Y),1) )
dim(Xt);dim(Yt)

maxiter = 100

model <- trainr(Y = Yt,
                X = Xt,
                learningrate = 0.01,
                hidden_dim = 100,
                numepochs = maxiter)
```

วิธีจัด

Array

```r
par(mfrow=c(2,1))

plot(colMeans(model$error[,1:maxiter]),type='l',xlab='Epoch',ylab='Errors')

Yp = predictr(model, Xt)

plot(as.vector(Yp), col = 'red', type='l',
     main = "Actual vs Predicted on Training Data Set",
     ylab = "Y,Yp")

lines(as.vector(Yt), type = 'l', col = 'black')
```



UNDERSTAND 3D DIMENSIONS AND SETTING RNN()

Train ข้อมูลเล็กๆ

array(data, dim_length)

- Create an array variable

ข้อมูล 1-6

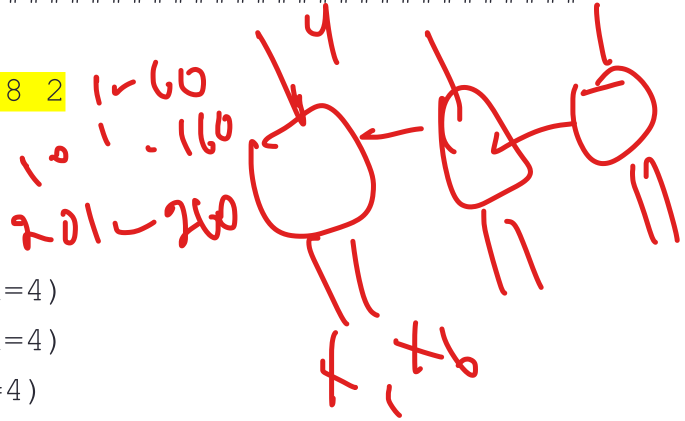
```
d = 1:(5*4*3)

arr = array(d, c(5,4,3))

dim(arr)

arr
```

```
> dim(arr)
[1] 5 4 3
> arr
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20

, , 2

     [,1] [,2] [,3] [,4]
[1,]   21   26   31   36
[2,]   22   27   32   37
[3,]   23   28   33   38
[4,]   24   29   34   39
[5,]   25   30   35   40

, , 3

     [,1] [,2] [,3] [,4]
[1,]   41   46   51   56
[2,]   42   47   52   57
[3,]   43   48   53   58
[4,]   44   49   54   59
```

Activity 8.6 Understand 3D data to trainr

รู้ของข้อมูล

```
##################################################
# Activity 8.6 To understand 3D dimension on rnn
# Modified by supakit@it.kmitl.ac.th
##################################################

x1 = 1:(5*4*3) #1000 8 2
x2 = 101:(100+5*4*3)
y  = 201:(200+5*4*3)

mx1 =  matrix(x1,ncol=4)
mx2 =  matrix(x2,ncol=4)
my =   matrix(y,ncol=4)

X <- array( c(mx1,mx2), dim=c(dim(mx1),2) )
Y <- array( c(my), dim=c(dim(my),1) )    #Is it
should be 1 or 2
dim(X)
X
dim(Y)
Y
```

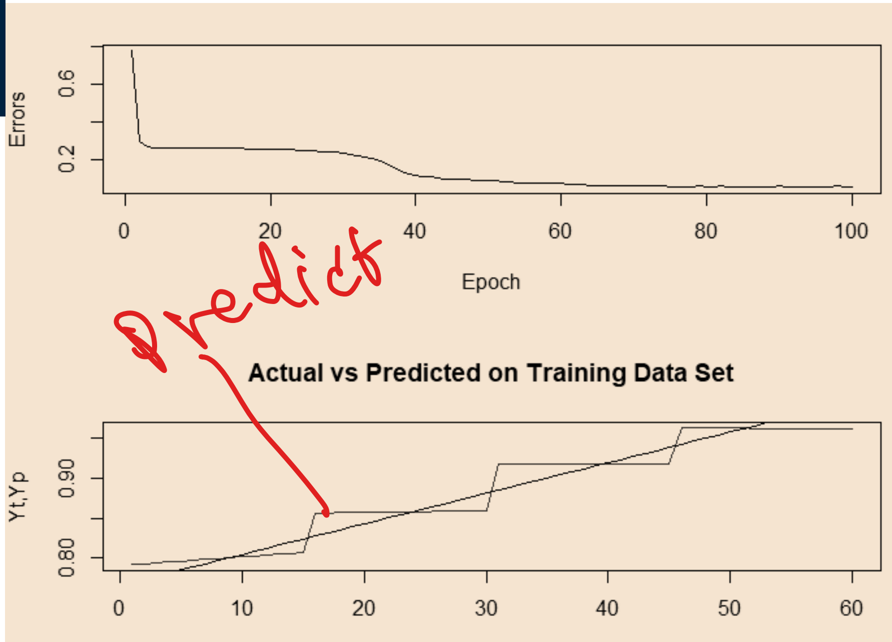



```
Xt = X/261
Yt = Y/261

model = trainr(Y=Yt,
               X=Xt,
               learningrate   =  0.1,
               hidden_dim     = 100,
               numepochs = 100)


Yp = predictr(model, Xt)

par(mfrow=c(2,1))
plot(colMeans(model$error[,1:maxiter]),type='l'
,xlab='Epoch',ylab='Errors')
plot(as.vector(Yp), col = 'red', type='l',
     main = "Actual vs Predicted on Training
Data Set",
     ylab = "Yt,Yp")
lines(as.vector(Yt), type = 'l', col = 'black')
```

```r
##################################################
# Activity 8.6 To understand 3D dimension on rnn
# Modify by supakit@it.kmitl.ac.th
##################################################

x1 = 1:(5*4*3) #1000 8 2
x2 = 101:(100+5*4*3)
y  = 201:(200+5*4*3)

mx1 =  matrix(x1,ncol=4)
mx2 =  matrix(x2,ncol=4)
my =   matrix(y,ncol=4)

X <- array( c(mx1,mx2), dim=c(dim(mx1),2) )
Y <- array( c(my), dim=c(dim(my),1) )    #Is it should be 1 or 2
dim(X)
X
dim(Y)
Y

Xt = X/261
Yt = Y/261

model = trainr(Y=Yt,
               X=Xt,
               learningrate   =  0.1,
               hidden_dim     = 100,
               numepochs = 100)



Yp = predictr(model, Xt)

par(mfrow=c(2,1))
plot(colMeans(model$error[,1:maxiter]),type='l',xlab='Epoch',ylab='Errors')
plot(as.vector(Yp), col = 'red', type='l',
     main = "Actual vs Predicted on Training Data Set",
     ylab = "Yt,Yp")
lines(as.vector(Yt), type = 'l', col = 'black')
```

```
> dim(X)
[1] 15  4  2
> X
, , 1

      [,1] [,2] [,3] [,4]
 [1,]    1   16   31   46
 [2,]    2   17   32   47
 [3,]    3   18   33   48
 [4,]    4   19   34   49
 [5,]    5   20   35   50
 [6,]    6   21   36   51
 [7,]    7   22   37   52
 [8,]    8   23   38   53
 [9,]    9   24   39   54
[10,]   10   25   40   55
[11,]   11   26   41   56
[12,]   12   27   42   57
[13,]   13   28   43   58
[14,]   14   29   44   59
[15,]   15   30   45   60

, , 2

      [,1] [,2] [,3] [,4]
 [1,]  101  116  131  146
 [2,]  102  117  132  147
 [3,]  103  118  133  148
 [4,]  104  119  134  149
 [5,]  105  120  135  150
 [6,]  106  121  136  151
 [7,]  107  122  137  152
 [8,]  108  123  138  153
 [9,]  109  124  139  154
[10,]  110  125  140  155
[11,]  111  126  141  156
[12,]  112  127  142  157
[13,]  113  128  143  158
[14,]  114  129  144  159
[15,]  115  130  145  160
```

```
> dim(Y)
[1] 15  4  1
> Y
, , 1

      [,1] [,2] [,3] [,4]
 [1,]  201  216  231  246
 [2,]  202  217  232  247
 [3,]  203  218  233  248
 [4,]  204  219  234  249
 [5,]  205  220  235  250
 [6,]  206  221  236  251
 [7,]  207  222  237  252
 [8,]  208  223  238  253
 [9,]  209  224  239  254
[10,]  210  225  240  255
[11,]  211  226  241  256
[12,]  212  227  242  257
[13,]  213  228  243  258
[14,]  214  229  244  259
[15,]  215  230  245  260
```

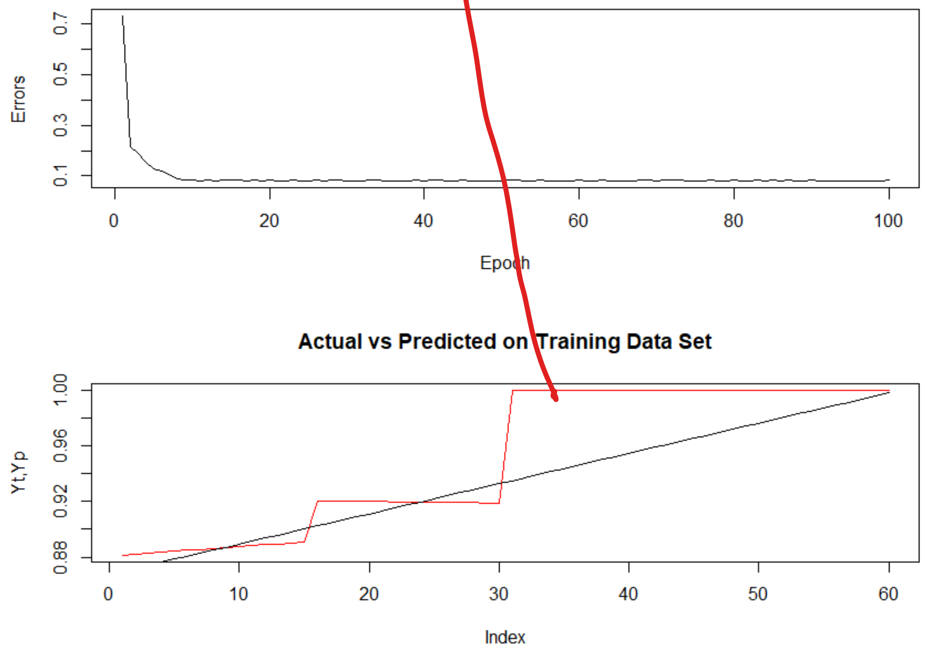


4-INPUT RNN

```r
# Modify by supakit@it.kmitl.ac.th
##################################################
rm(list=ls())
x1 = 1:(5*4*3) #1000 8 2
x2 = 101:(100+5*4*3)
x3 = 201:(200+5*4*3)
x4 = 301:(300+5*4*3)
y  = 401:(400+5*4*3)

mx1 =  matrix(x1,ncol=4)
mx2 =  matrix(x2,ncol=4)
mx3 =  matrix(x3,ncol=4)
mx4 =  matrix(x4,ncol=4)
my =   matrix(y,ncol=4)

X <- array( c(mx1,mx2,mx3,mx4), dim=c(dim(mx1),4) ) #4inputs
Y <- array( c(my), dim=c(dim(my),1) )    #Is it should be 1 or 2
dim(X)
X
dim(Y)
Y

Xt = X/461
Yt = Y/461
maxiter = 100
model = trainr(Y=Yt,
               X=Xt,
               learningrate   =  0.1,
               hidden_dim     = 300,
               numepochs = maxiter)



Yp = predictr(model, Xt)

par(mfrow=c(2,1))
plot(colMeans(model$error[,1:maxiter]),type='l',xlab='Epoch',ylab='Errors')
plot(as.vector(Yp), col = 'red', type='l',
     main = "Actual vs Predicted on Training Data Set",
```

ข้อมูลจริง

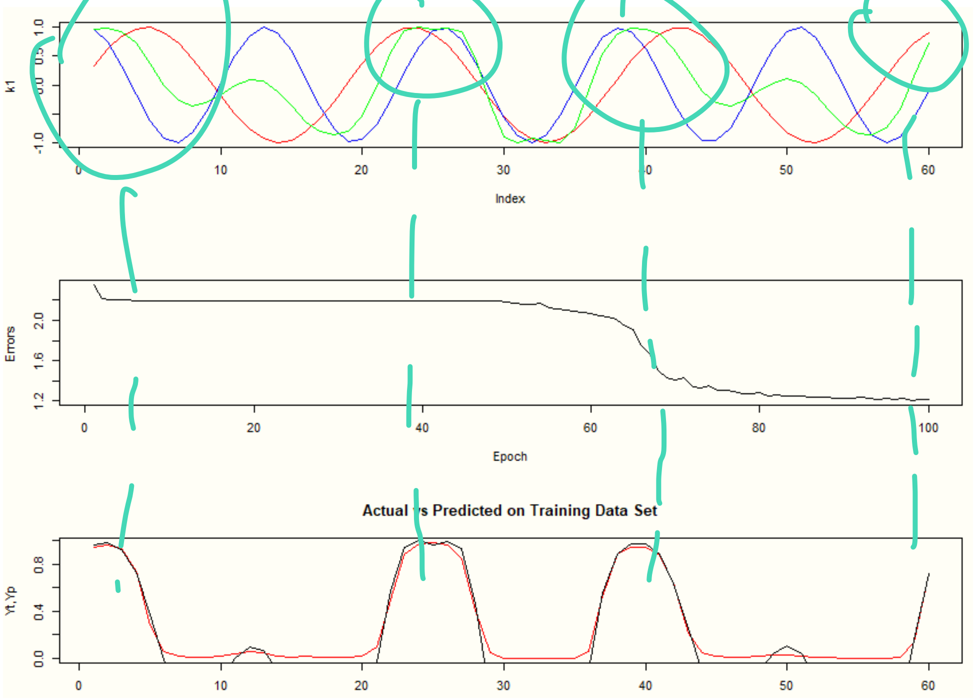
predict



SUM OF SINE WAVES

```r
# Example for learning trainr
# by supakit@it.kmitl.ac.th
###############################################
rm(list=ls())
library("rnn")
x1 = 1:(5*4*3) #1000 8 2
x2 = 101:(100+5*4*3)
y  = 401:(400+5*4*3)

##SIGNAL GENERATION
par(mfrow=c(3,1))
k1 = sin(x1/3)
plot(k1,type='l',col='red')
k2 = cos(x2/2)
lines(k2,type='l',col='blue')
k3 = sin(k1+k2)
lines(k3,type='l',col='green')
x1 = k1; x2 = k2; y = k3


mx1 =  matrix(x1,ncol=4)
mx2 =  matrix(x2,ncol=4)
my =   matrix(y,ncol=4)

X <- array( c(mx1,mx2), dim=c(dim(mx1),2) ) #4inputs
Y <- array( c(my), dim=c(dim(my),1) )    #Is it should be 1 or 2
dim(X)
X
dim(Y)
Y

#Xt = X/461
#Yt = Y/461
Xt = X
Yt = Y
maxiter = 100
model = trainr(Y=Yt,
               X=Xt,
               learningrate   =  0.1,
```

```r
plot(colMeans(model$error[,1:maxiter]),type='l',xlab='Epoch',ylab='Errors')
plot(as.vector(Yp), col = 'red', type='l',
     main = "Actual vs Predicted on Training Data Set",
     ylab = "Yt,Yp")
lines(as.vector(Yt), type = 'l', col = 'black')
```



แปลงเป็น sin wave
ให้เริ่มพร้อมกัน

Addition

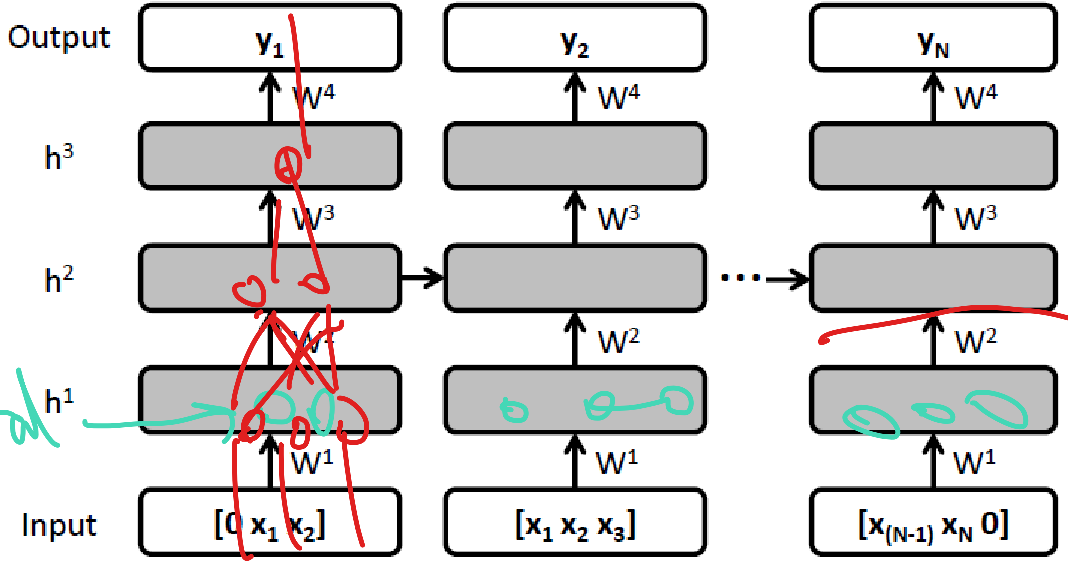


Figure 1: *Deep Recurrent Denoising Autoencoder. A model with 3 hidden layers that takes 3 frames of noisy input features and predicts a clean version of the center frame*

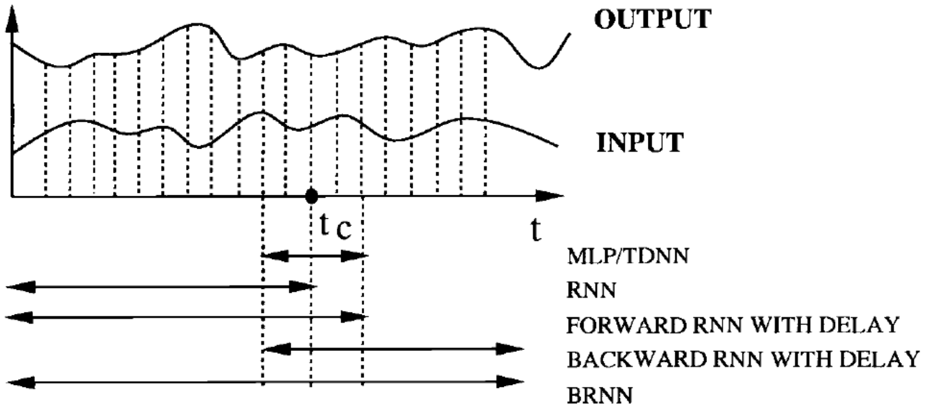


Fig. 2. Visualization of the amount of input information used for prediction by different network structures.

Forword

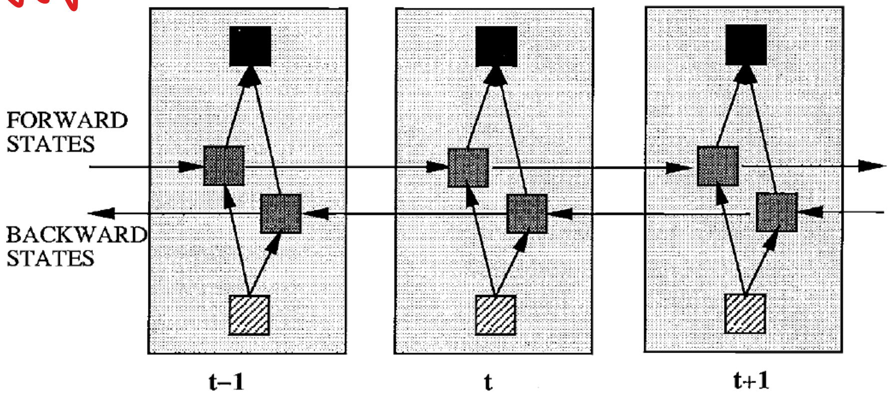
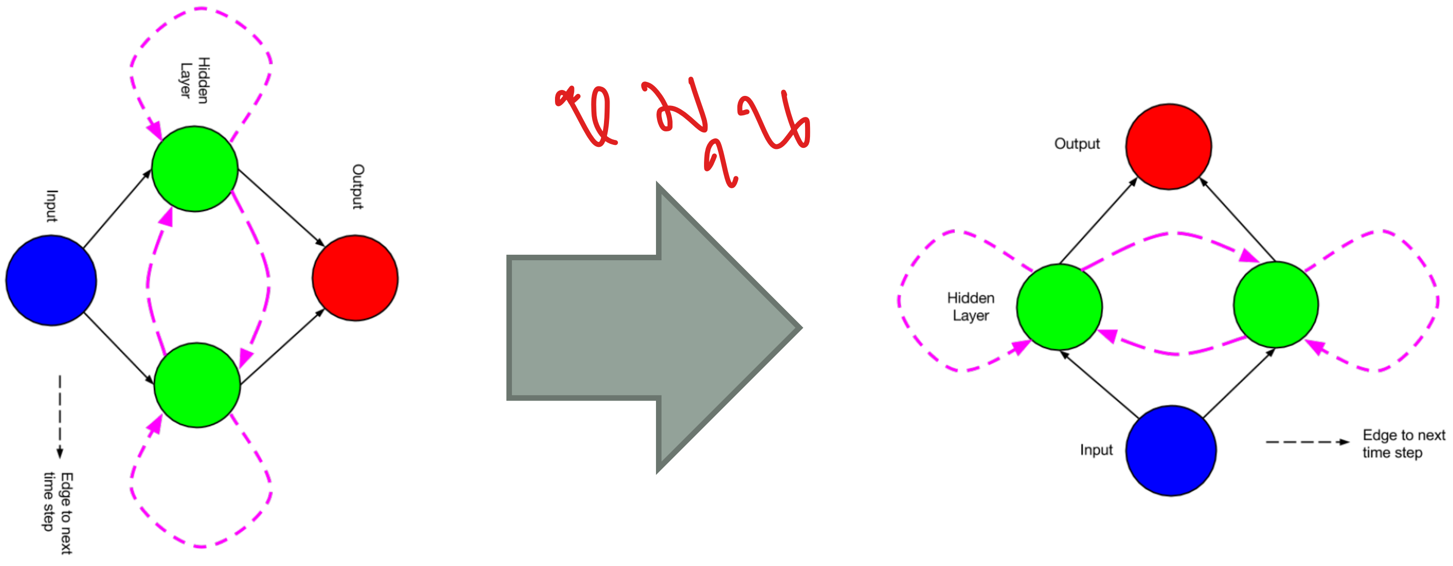


Fig. 3. General structure of the bidirectional recurrent neural network (BRNN) shown unfolded in time for three time steps.

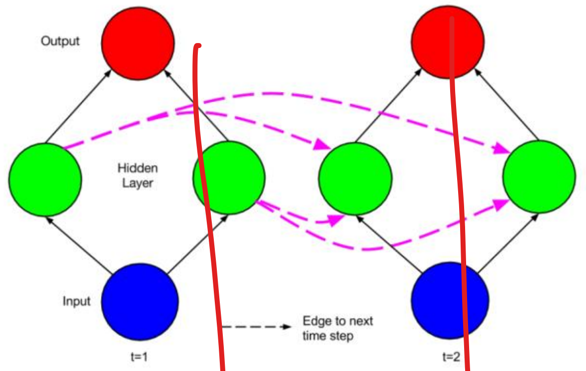
ลองนำไปใส่กลับ

backword

หมุน

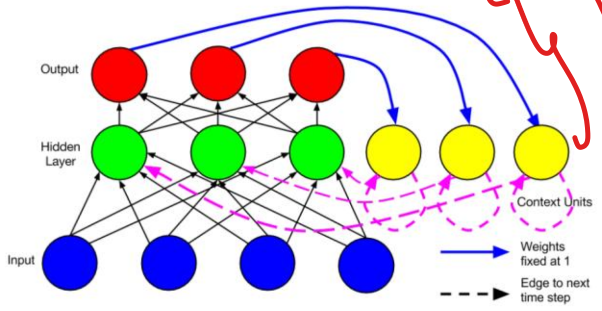


Output



ใช้ในเรื่องเวลาต่อไป

RNN



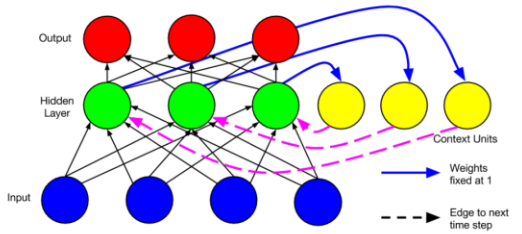


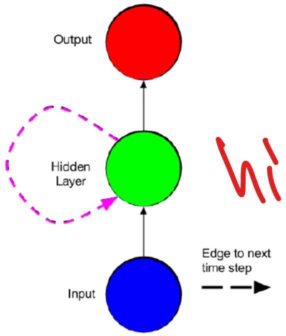
Figure 4: Visualizing the network unfolded across time steps.

t1- Input t2

Figure 5: A recurrent neural network as proposed by Jordan (1986).

Figure 6: An Elman network as described in *Finding Structure in Time* (1990) [17]. Hidden units are connected 1-to-1 to context units across time steps, which in turn feed back into the corresponding hidden units.

[illegible] cell



hidden

Figure 7: A simple recurrent net with one input unit, one output unit, and one recurrent hidden unit.

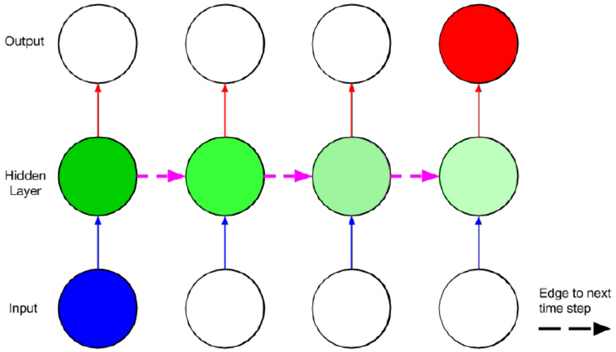


Figure 8: A visualization of the vanishing gradient problem, using the architecture depicted in Figure 7. If the weight along the purple edge is less than one, the effect of the input at the first time step on the output at the final time step will rapidly diminish as a function of the size of the interval in between. An illustration like this appears in [23]

Summary