RECURRENT NEURAL NETWORK (RNN)

Asst.Prof.Dr.Supakit Nootyaskool
IT-KMITL

Learning Outcome

- เข้าใจปัญหาของ Neural Network ในการแก้ปัญหาที่มีลำดับเวลา
- Understand problem of neural network solving time series problem

- ได้ประสบการณ์ในการเขียนโปรแกรมคำนวณเมทริกซ์
- Experiment calculating matrix in R

Topics

- Multiply Matrix in R
- From Feed Forward to RNN
- RNN concept

MULTIPLY MATRIX IN R

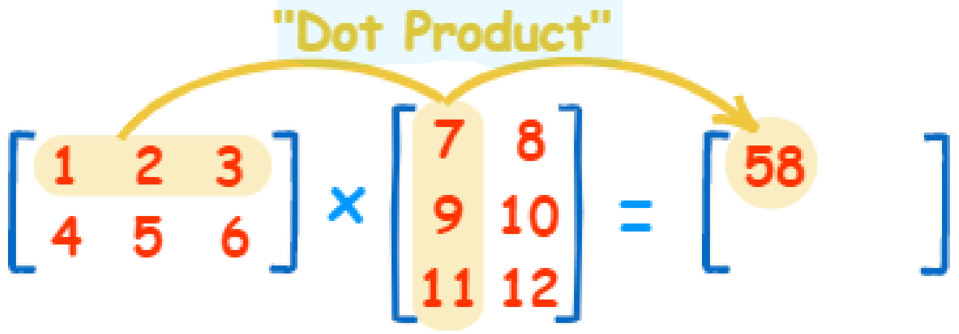
การคูณแมทริกซ์ในR

To understand matrix multiplication in R

Multiplying a Matrix by Another Matrix

But to multiply a matrix **by another matrix** we need to do the "dot product" of rows and columns ... what does that mean? Let us see with an example:

To work out the answer for the **1st row** and **1st column**:



The "Dot Product" is where we **multiply matching members**, then sum up:

$$(1, 2, 3) \bullet (7, 9, 11) = 1\times7 + 2\times9 + 3\times11$$
$$= 58$$

We match the 1st members (1 and 7), multiply them, likewise for the 2nd members (2 and 9) and the 3rd members (3 and 11), and finally sum them up.

Want to see another example? Here it is for the 1st row and **2nd column**:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ & \end{bmatrix}$$

$$(1, 2, 3) \bullet (8, 10, 12) = 1\times8 + 2\times10 + 3\times12$$
$$= 64$$

We can do the same thing for the **2nd row** and **1st column**:

$$(4, 5, 6) \bullet (7, 9, 11) = 4\times7 + 5\times9 + 6\times11$$
$$= 139$$

And for the **2nd row** and **2nd column**:

$$(4, 5, 6) \bullet (8, 10, 12) = 4\times8 + 5\times10 + 6\times12$$
$$= 154$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix} \checkmark$$

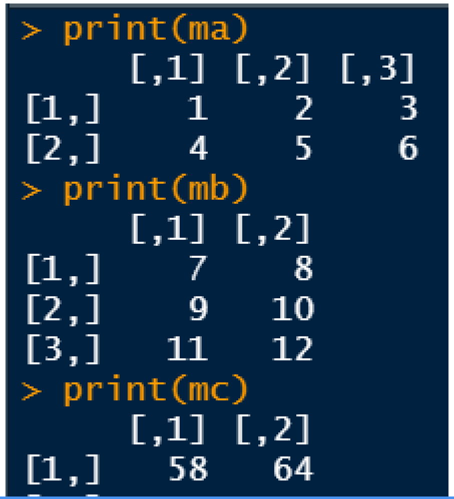
Matrix multiplication

```
a = c(1,2,3,4,5,6) Vector
ma = matrix(a,ncol=3)
print(ma)
ma = matrix(a,ncol=3,byrow=TRUE)
print(ma)

b = c(7,8,9,10,11,12)
mb = matrix(b,ncol=2,byrow=TRUE)
print(mb)

mc = ma*mb   #Error non-conformable arrays
mc = ma%*%mb
print(mc)
mymul(ma,mb)
```

bycolumn

%*%

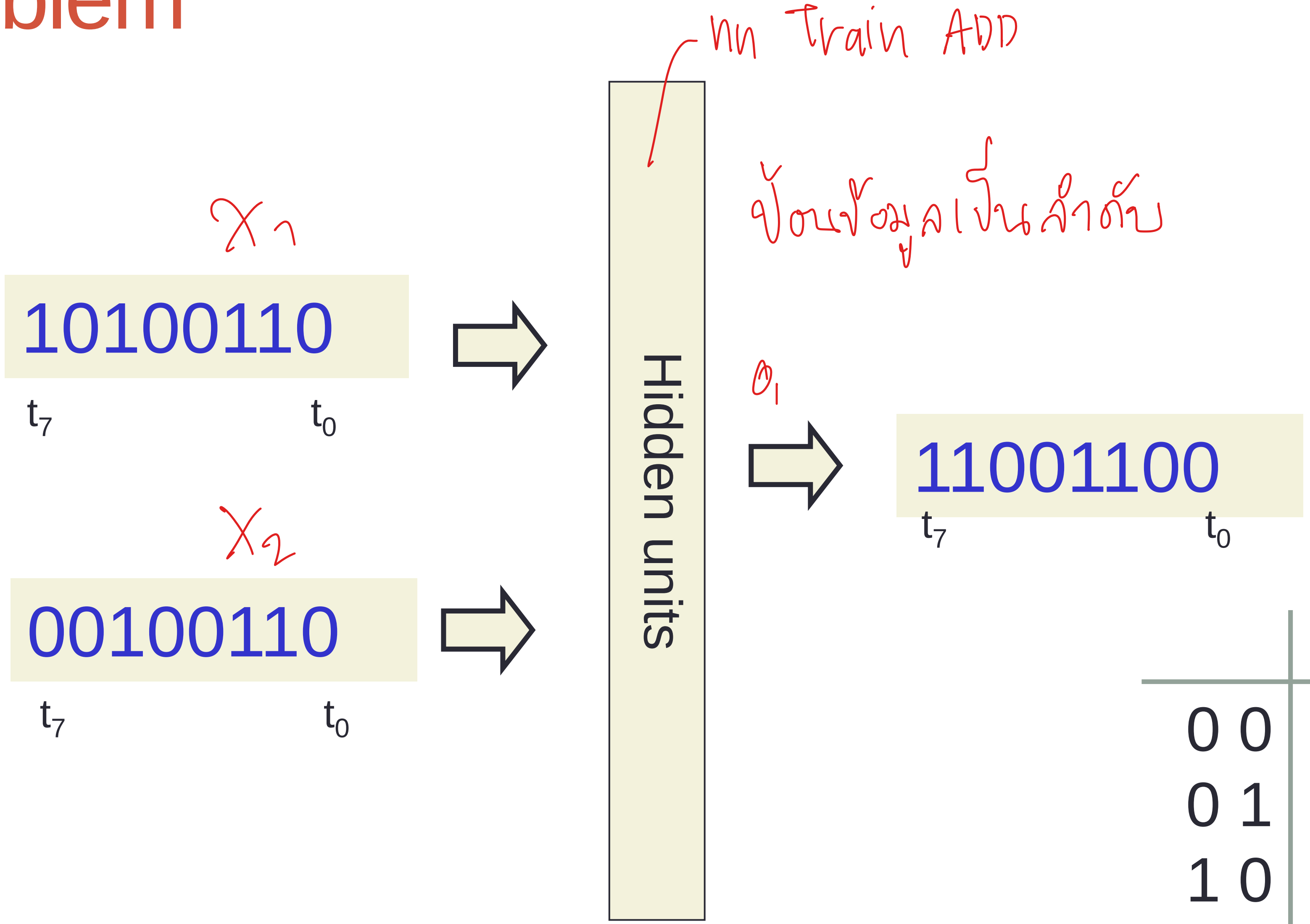
```
#Make calculation without the R operator
print(ma[1,])
print(mb[,2])
mcbar=matrix(rep(0,4),2,2)
mcbar[1,1] = sum(ma[1,] * mb[,1])
mcbar[1,2] = sum(ma[1,] * mb[,2])
mcbar[2,1] = sum(ma[2,] * mb[,1])
mcbar[2,2] = sum(ma[2,] * mb[,2])
print(mcbar)
```



```
> print(ma)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
> print(mb)
     [,1] [,2]
[1,]    7    8
[2,]    9   10
[3,]   11   12
> print(mc)
     [,1] [,2]
[1,]   58   64
```

FROM
FEED FORWARD
TO
RNN

Toy problem



	Add
0 0	0
0 1	1
1 0	1
1 1	0 carry 1

Toy problem

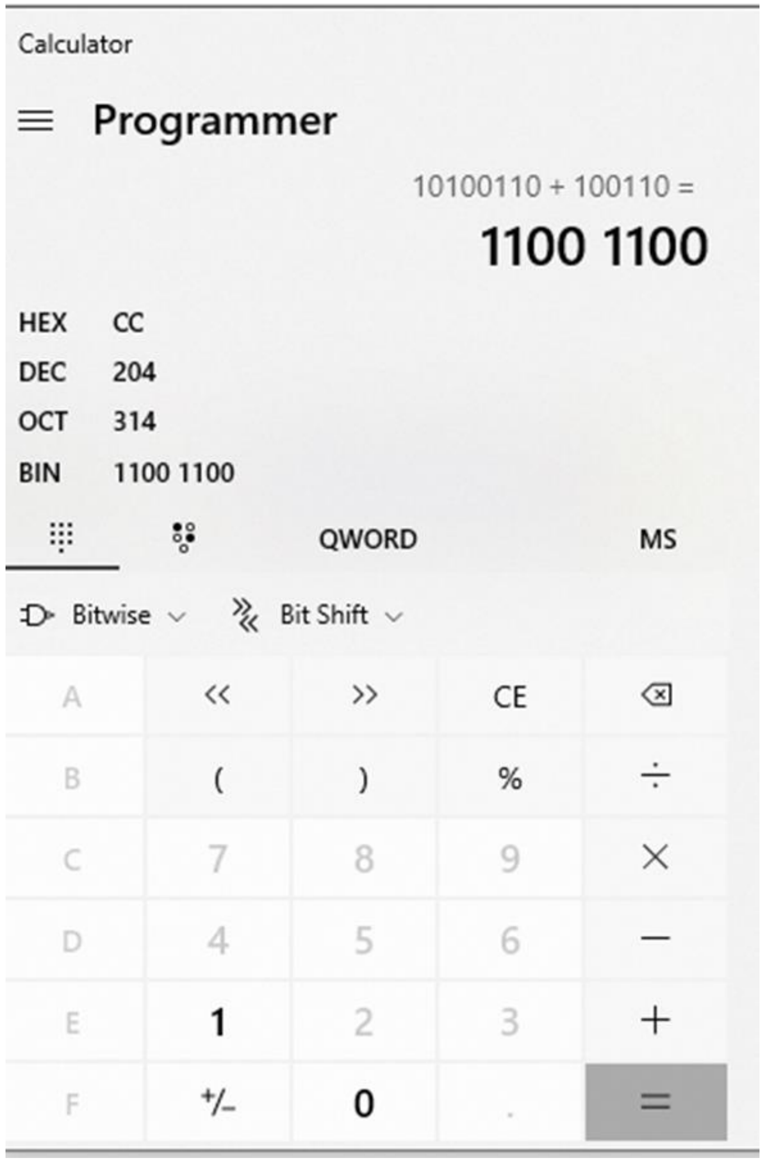
	Add
0 0	0
0 1	1
1 0	1
1 1	0 carry 1

X1 10100110

X2 00100110

+

Y 11001100



10100110 + 100110 =

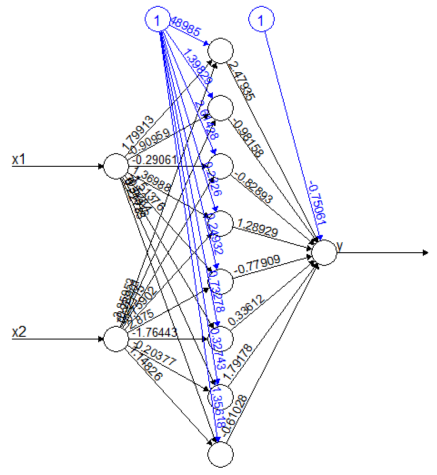
1100 1100

HEX CC
DEC 204
OCT 314
BIN 1100 1100

Activity 7.1 Toy problem on FF

```
##########################################
#ACTIVITY 7-1 TOY PROBLEM WITH FEEDFORWARD
##########################################
library("neuralnet")
x1 = c(1,0,1,0,0,1,1,0)
x2 = c(0,0,1,0,0,1,1,0)
y  = c(1,1,0,0,1,1,0,0)
traindata = data.frame(x1,x2,y)
traindata

####TRAINING
model <- neuralnet( y~x1+x2,
          traindata,
         hidden=8, ##<--Change here
         rep = 1,
         linear.output = FALSE)
print(model)
plot(model)
print(model$net.result)
```

```
####TESTING
x1 = c(0,1,0,1)
x2 = c(0,0,1,1)
input = data.frame(x1,x2)
pred = predict(model,input)
pred
```



Error: 0.841534   Steps: 20

```
> print(model$net.result)
[[1]]
          [,1]
[1,] 0.8755806
[2,] 0.5141286
[3,] 0.3397498
[4,] 0.5141286
[5,] 0.5141286
[6,] 0.3397498
[7,] 0.3397498
[8,] 0.5141286
```

```
> pred
          [,1]
[1,] 0.5141286
[2,] 0.8755806
[3,] 0.1625577
[4,] 0.3397498
```

Activity 7.1 Toy problem on FF

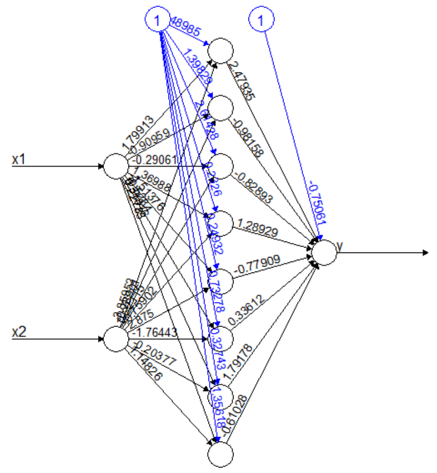
```
##########################################
#ACTIVITY 7-1 TOY PROBLEM WITH FEEDFORWARD
##########################################
library("neuralnet")
x1 = c(1,0,1,0,0,1,1,0)
x2 = c(0,0,1,0,0,1,1,0)
y  = c(1,1,0,0,1,1,0,0)
traindata = data.frame(x1,x2,y)
traindata

####TRAINING
model <- neuralnet( y~x1+x2,
          traindata,
         hidden=8, ##<--Change here
         rep = 1,
         linear.output = FALSE)
print(model)
plot(model)
print(model$net.result)
```

Time

NNX

RNN

```
####TESTING
x1 = c(0,1,0,1)
x2 = c(0,0,1,1)
input = data.frame(x1,x2)
pred = predict(model,input)
pred
```



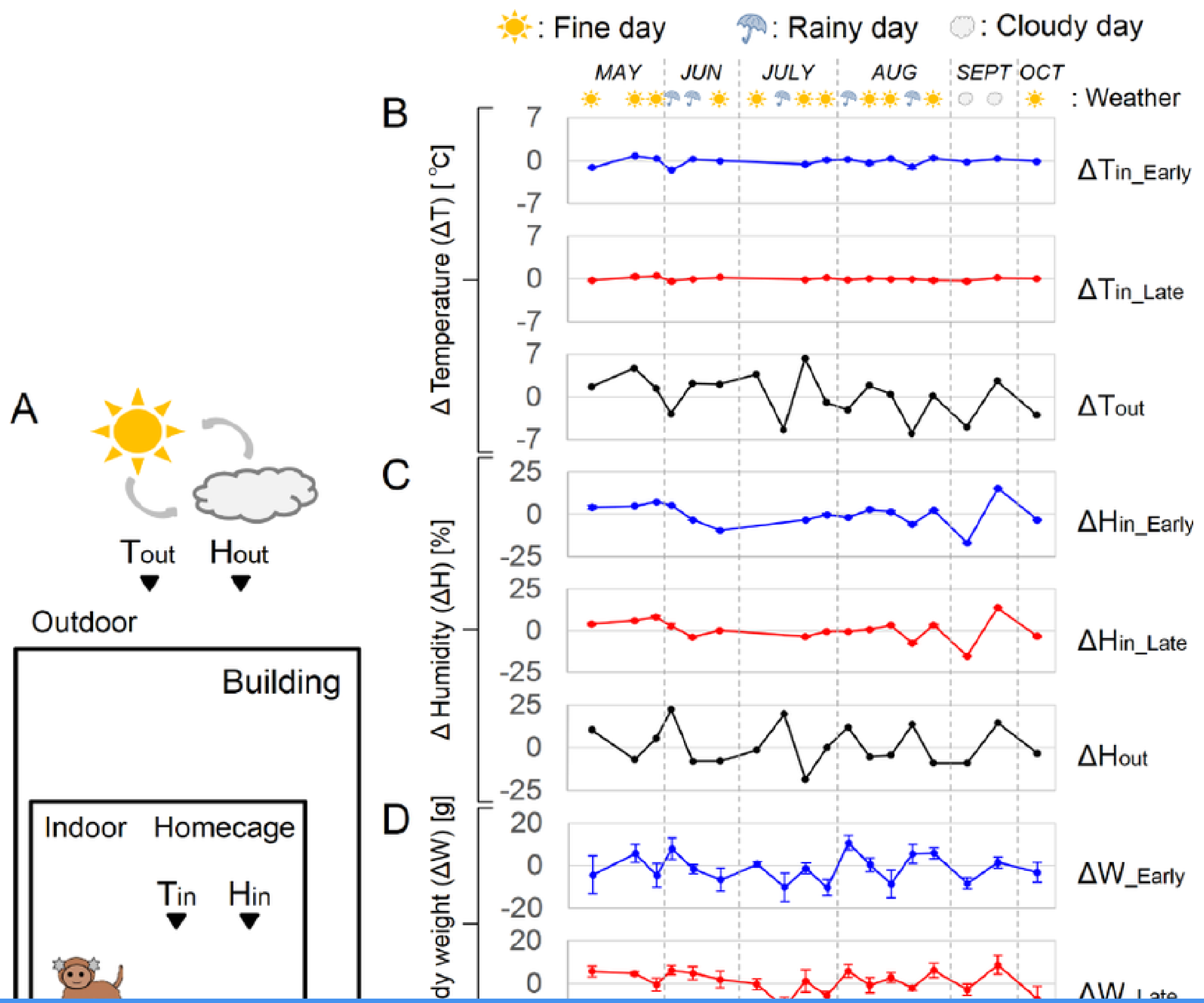
Error: 0.841534   Steps: 20

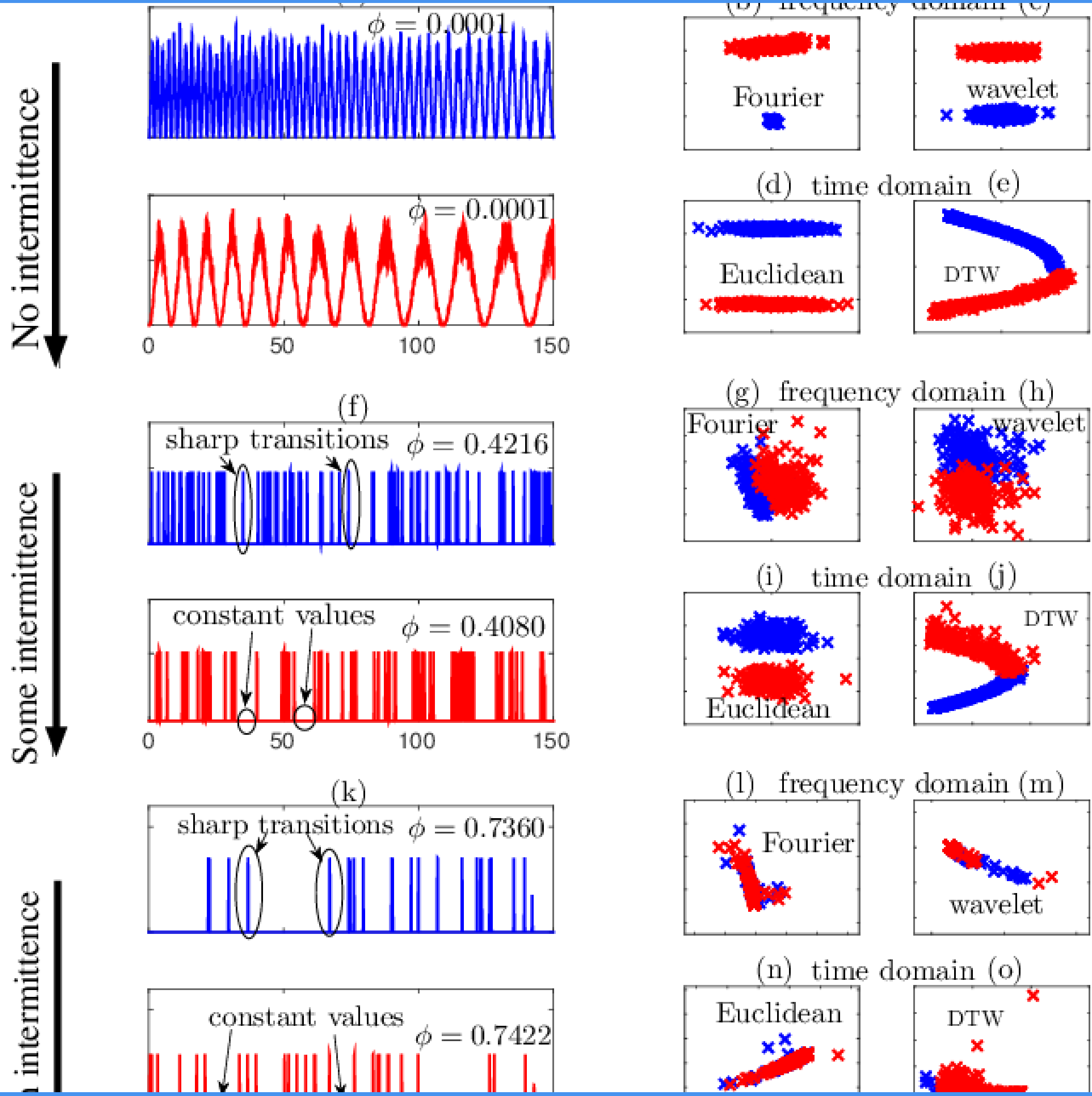
y = c(1,1,0,0,1,1,0,0)

```
> print(model$net.result)
[[1]]
          [,1]
[1,] 0.8755806
[2,] 0.5141286
[3,] 0.3397498
[4,] 0.5141286
[5,] 0.5141286
[6,] 0.3397498
[7,] 0.3397498
[8,] 0.5141286
```

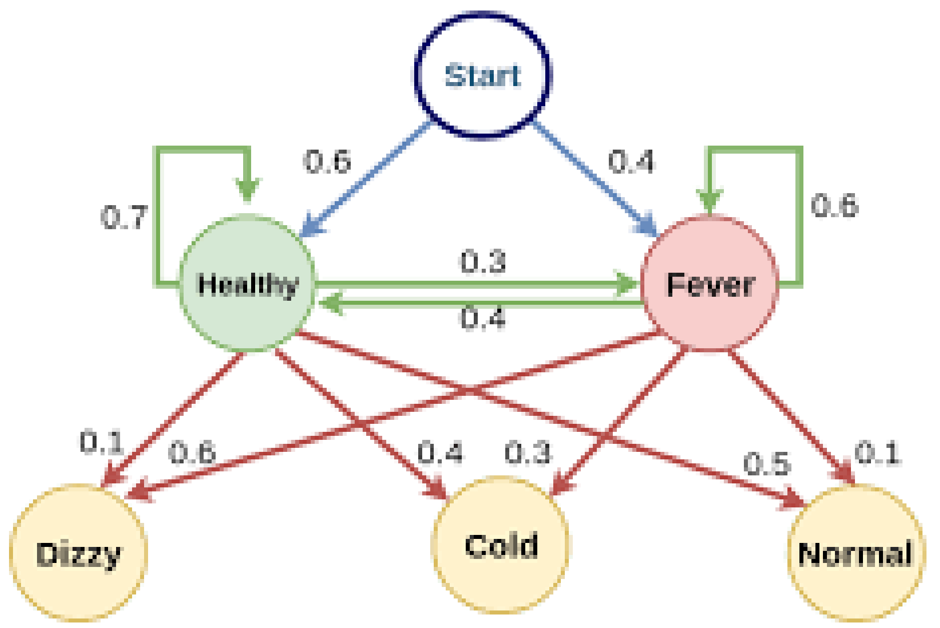
```
> pred
          [,1]
[1,] 0.5141286
[2,] 0.8755806
[3,] 0.1625577
[4,] 0.3397498
```

Time series data





Algorithms for Time series data



Initial Probability

Transition Probability

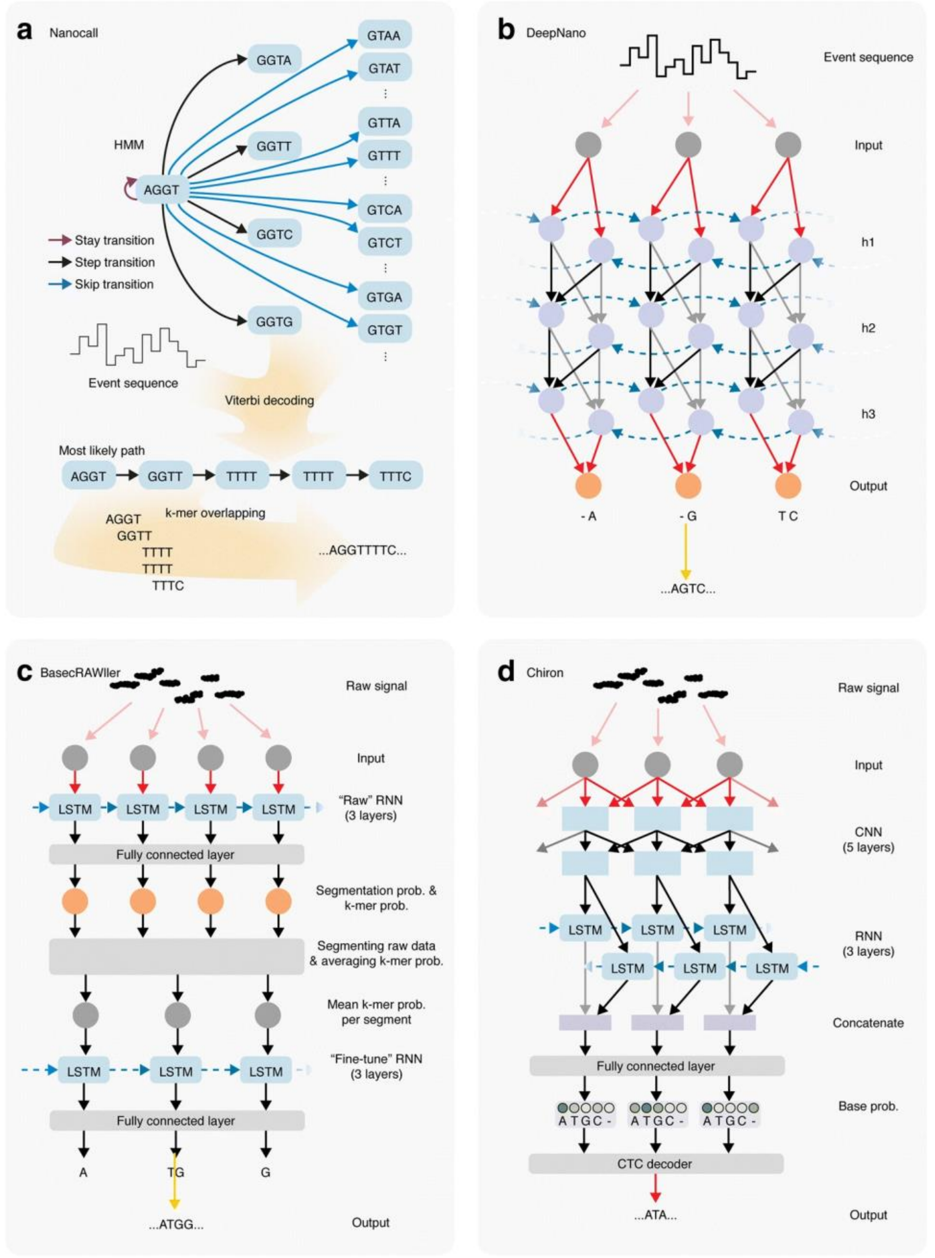
Emission Probability

Hidden Markov Models ⟷ RNN

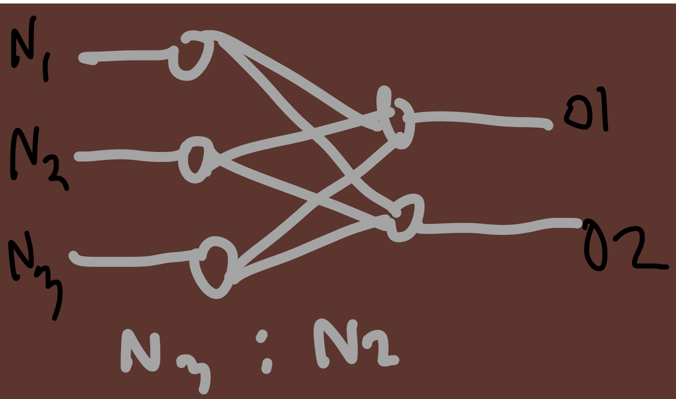
ไม่ขึ้นอยู่กับ จน. state

ก |↑ ↗ ↘ |↓

ขึ้นอยู่กับจำนวน state

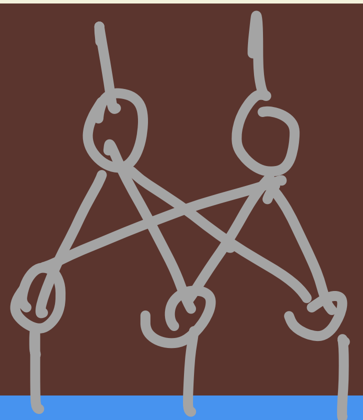


Neural network



RNN CONCEPT

ต่างไปแบบเขียนกลับด้านแทน



แนวตั้ง

Define vectors

0 0 1

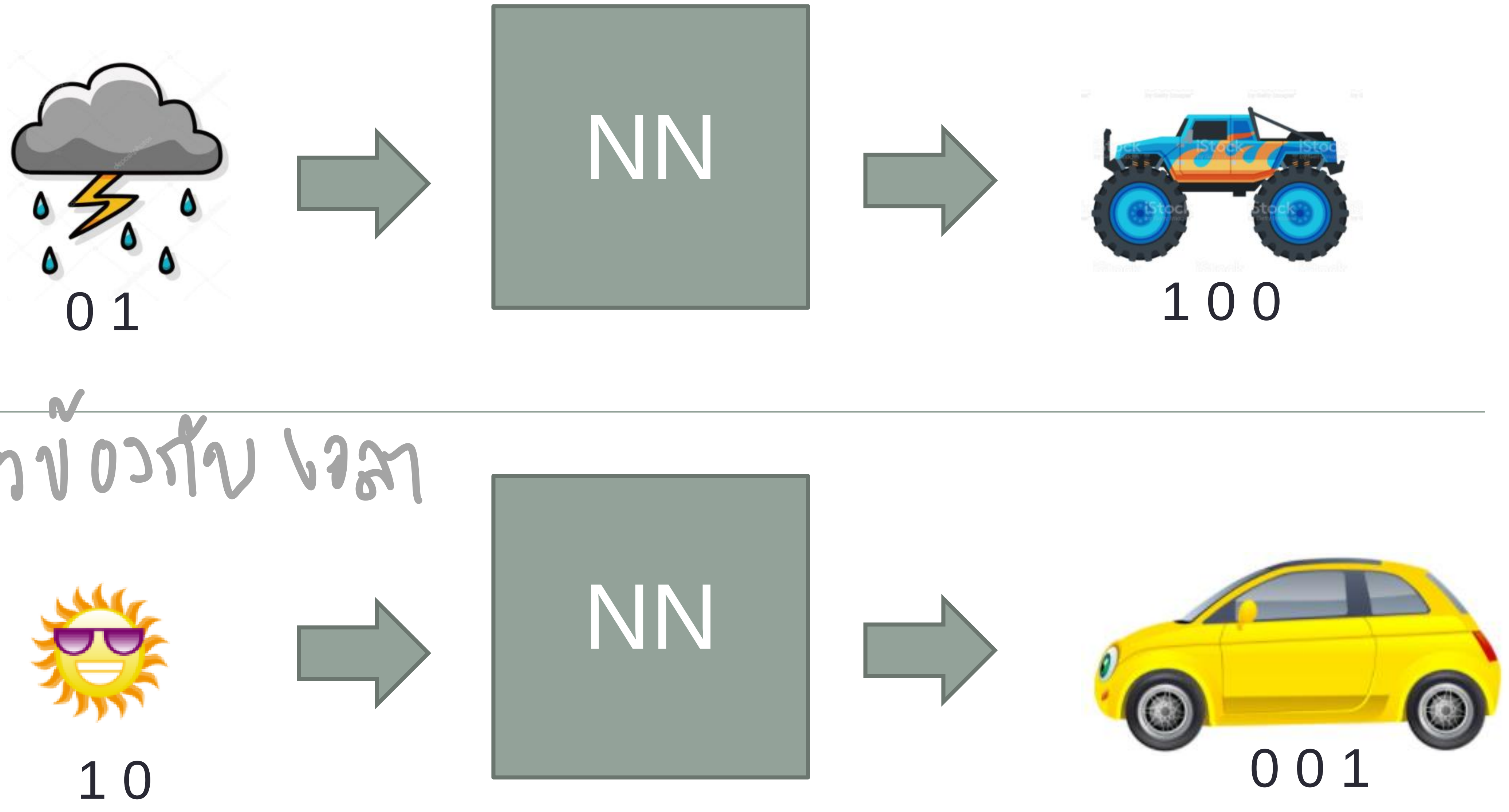
0 1 0

1 0 0

0 1

1 0

NN



ไม่เกี่ยวข้องกับเวลา

NN

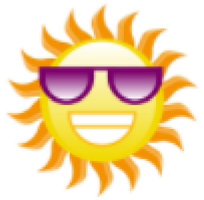
0 1

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

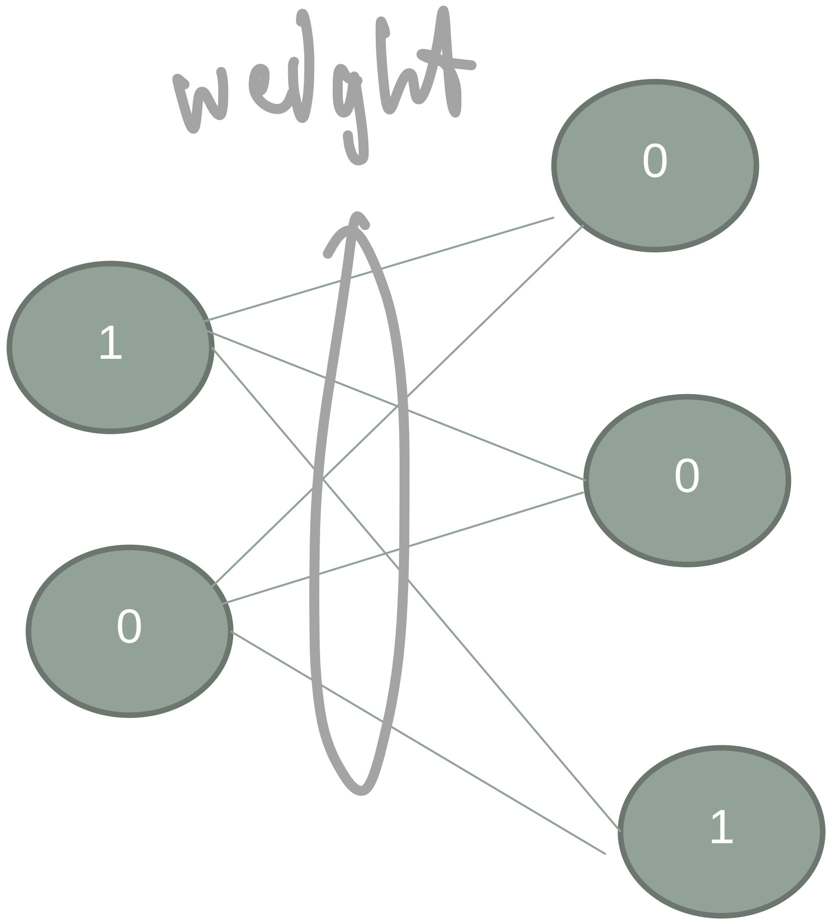
1 0 0

```
a = matrix(c(1,0,0,0,1,0),3,2)
b = matrix(c(1,0),2,1)
c = a %*% b
a;b;c
```

NN



1 0



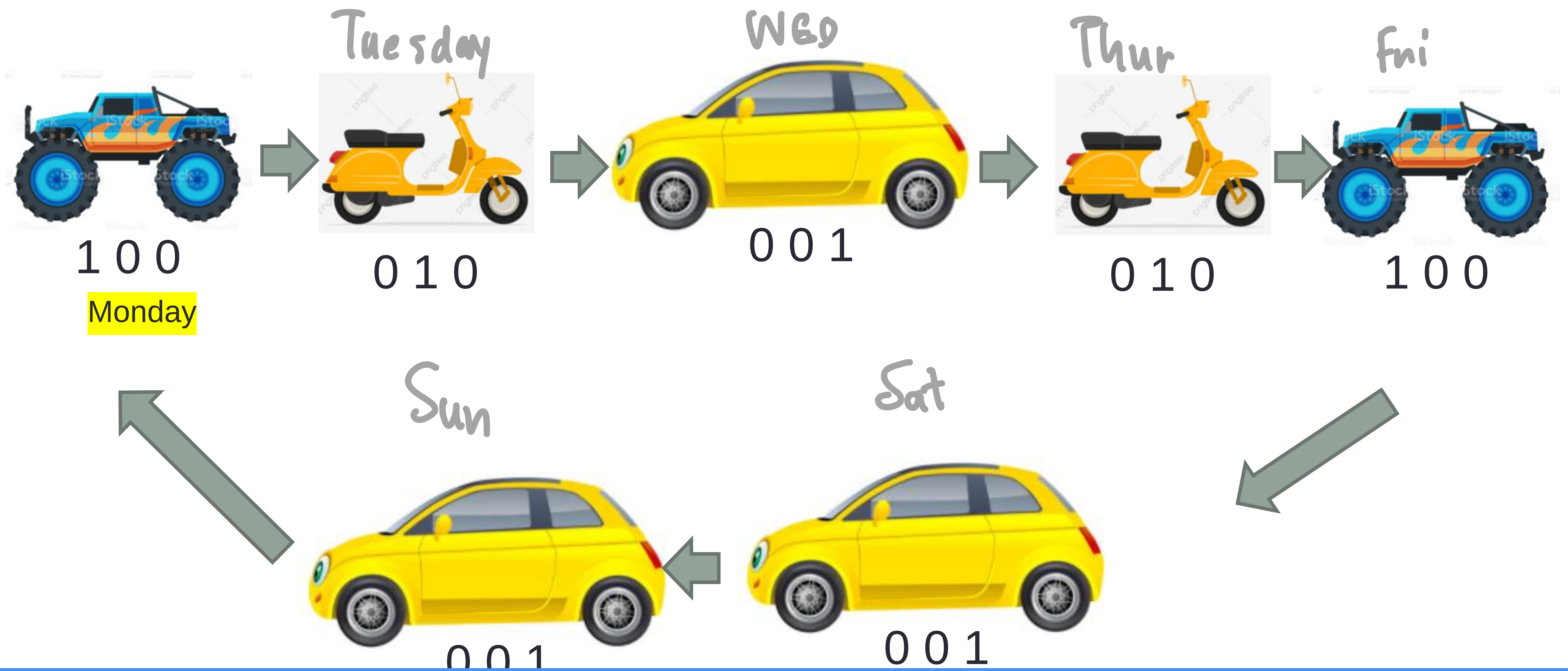
1 0 0

0 1 0

0 0 1

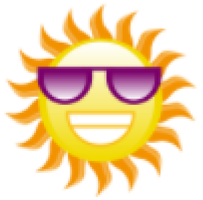
Driving Schedule

NN เดียวทดวัน



ค.เป็นจริง

รถ ตามสภาพอากาศ



1 0

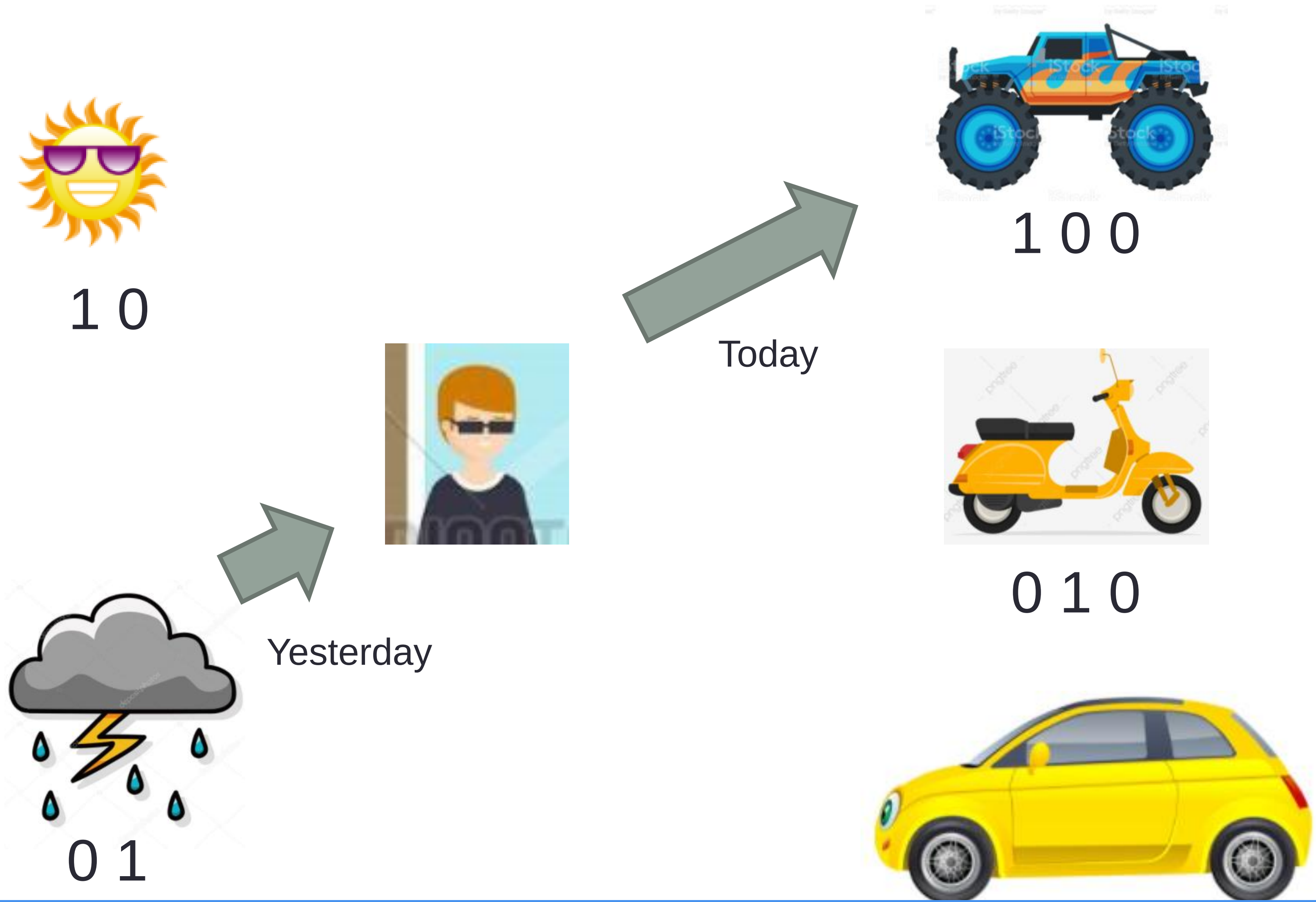
Yesterday

1 0 0

0 1 0

0 1

Today

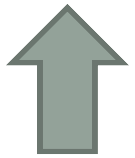

1 0
0 1
Yesterday
Today
1 0 0
0 1 0

Driving on Monday

Output รถ

0 0 1

Monday



Input

สภาพอากาศ

1 0

Driving on Tuesday

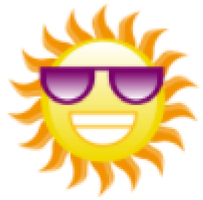
อดีต

ปัจุบัน

0 0 1

0 1 0

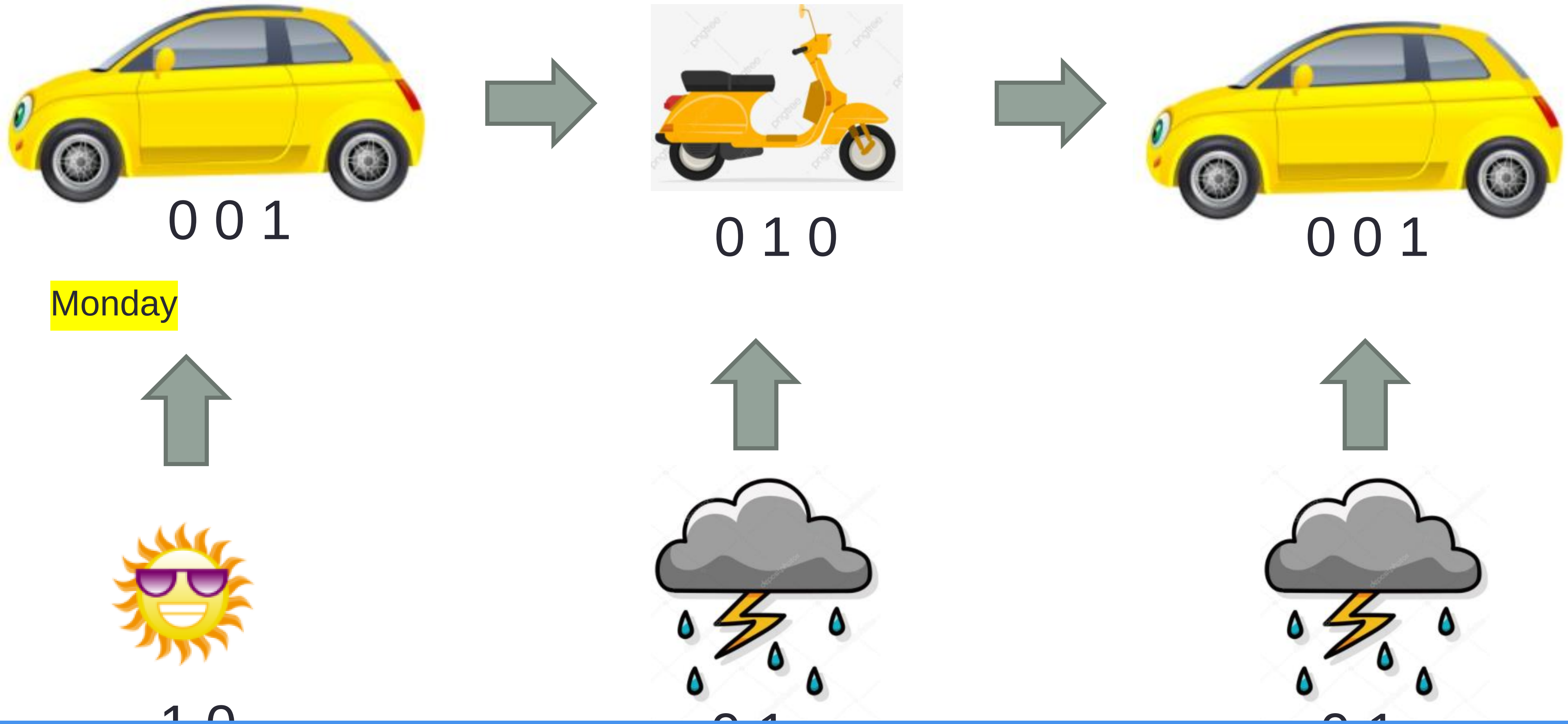
Monday



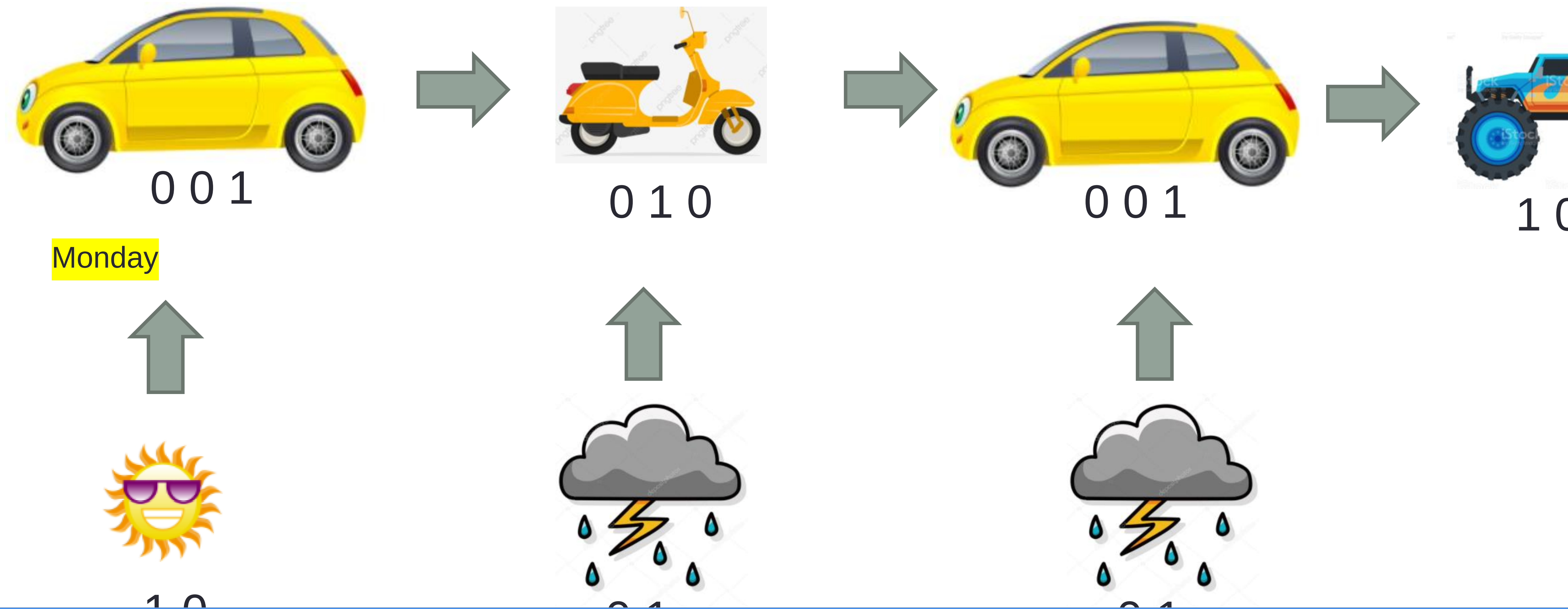
1 0

0 1

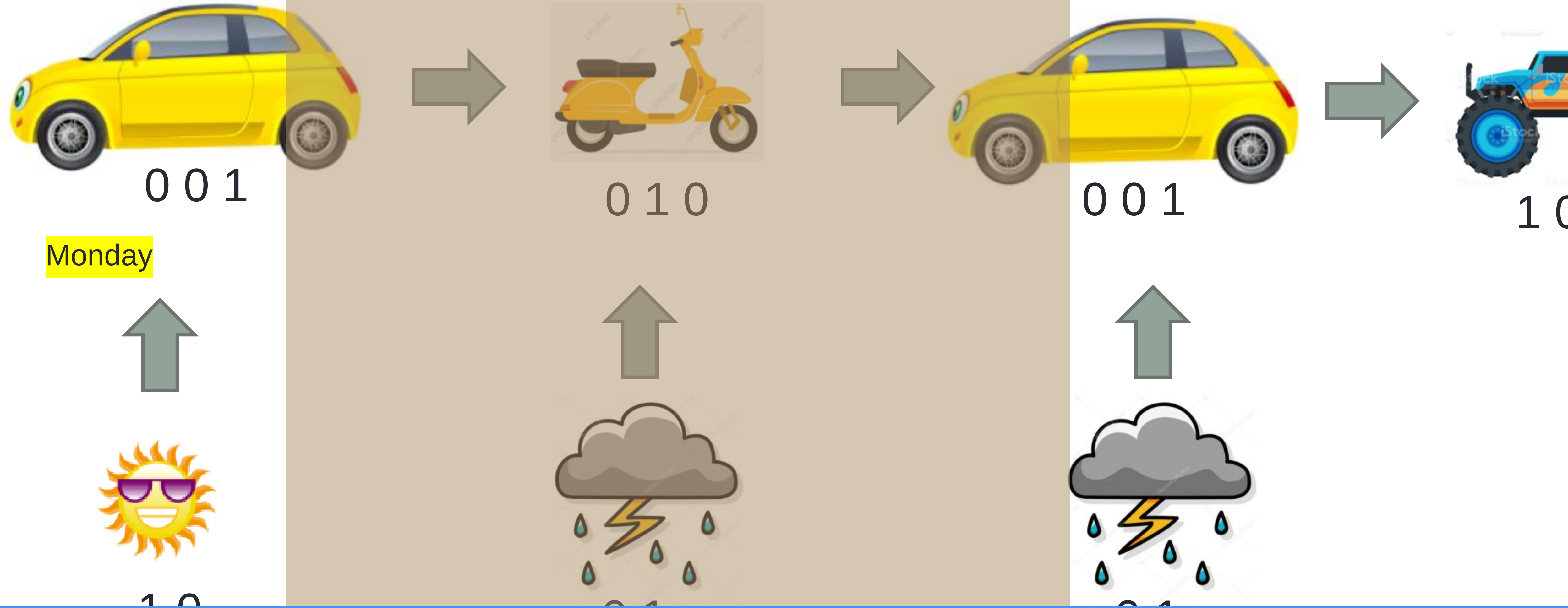
Driving on Wednesday

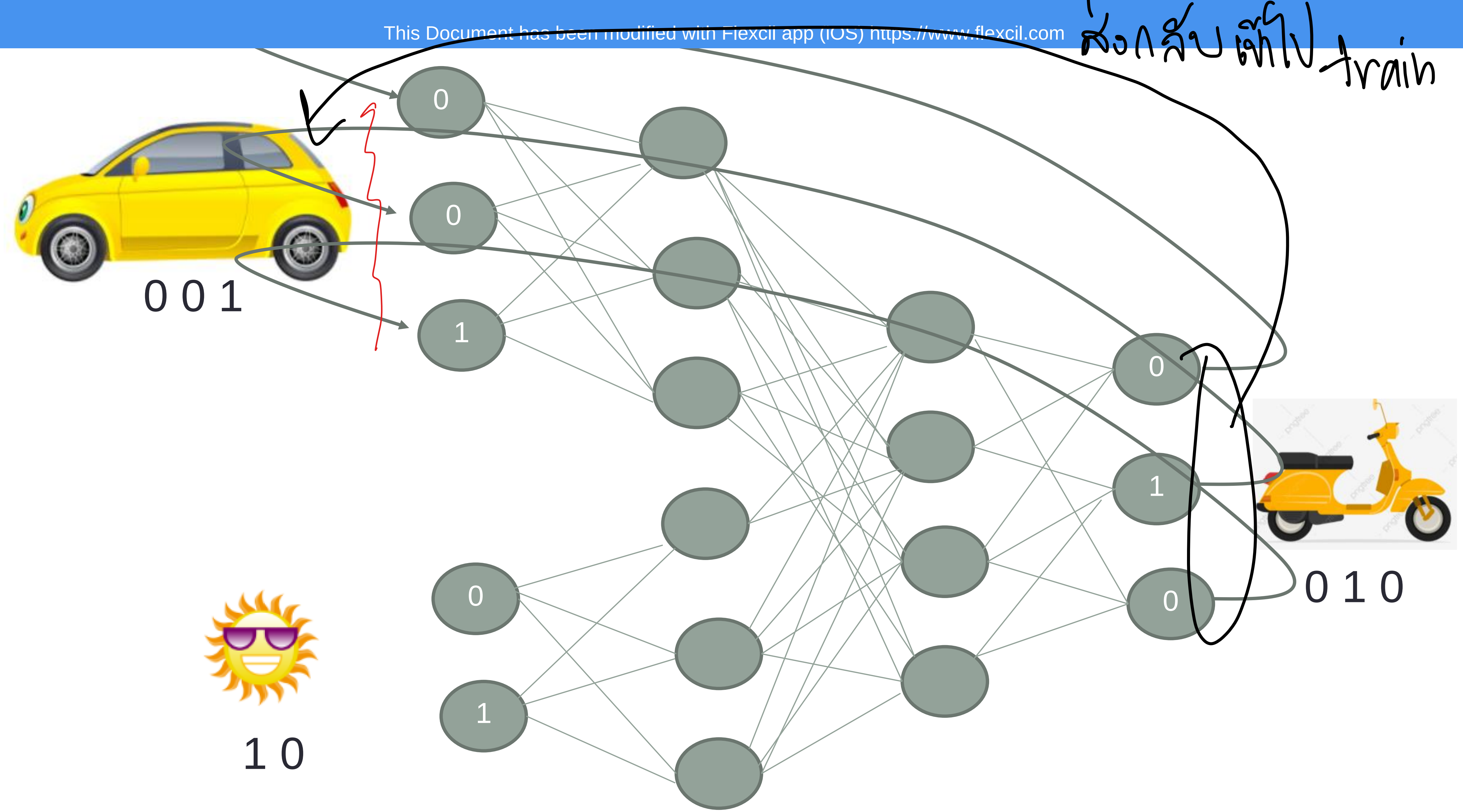


Driving on Thursday

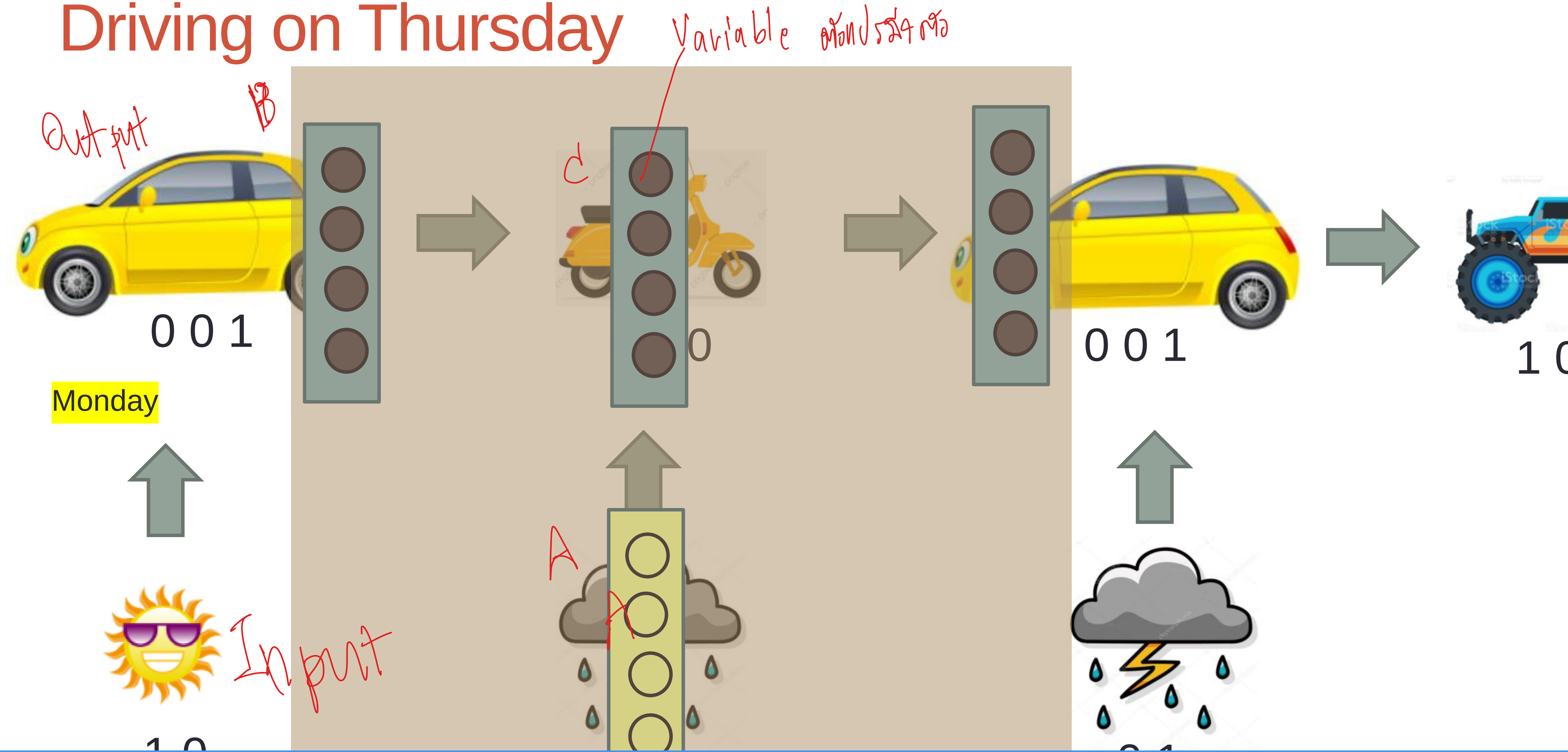


Driving on Thursday

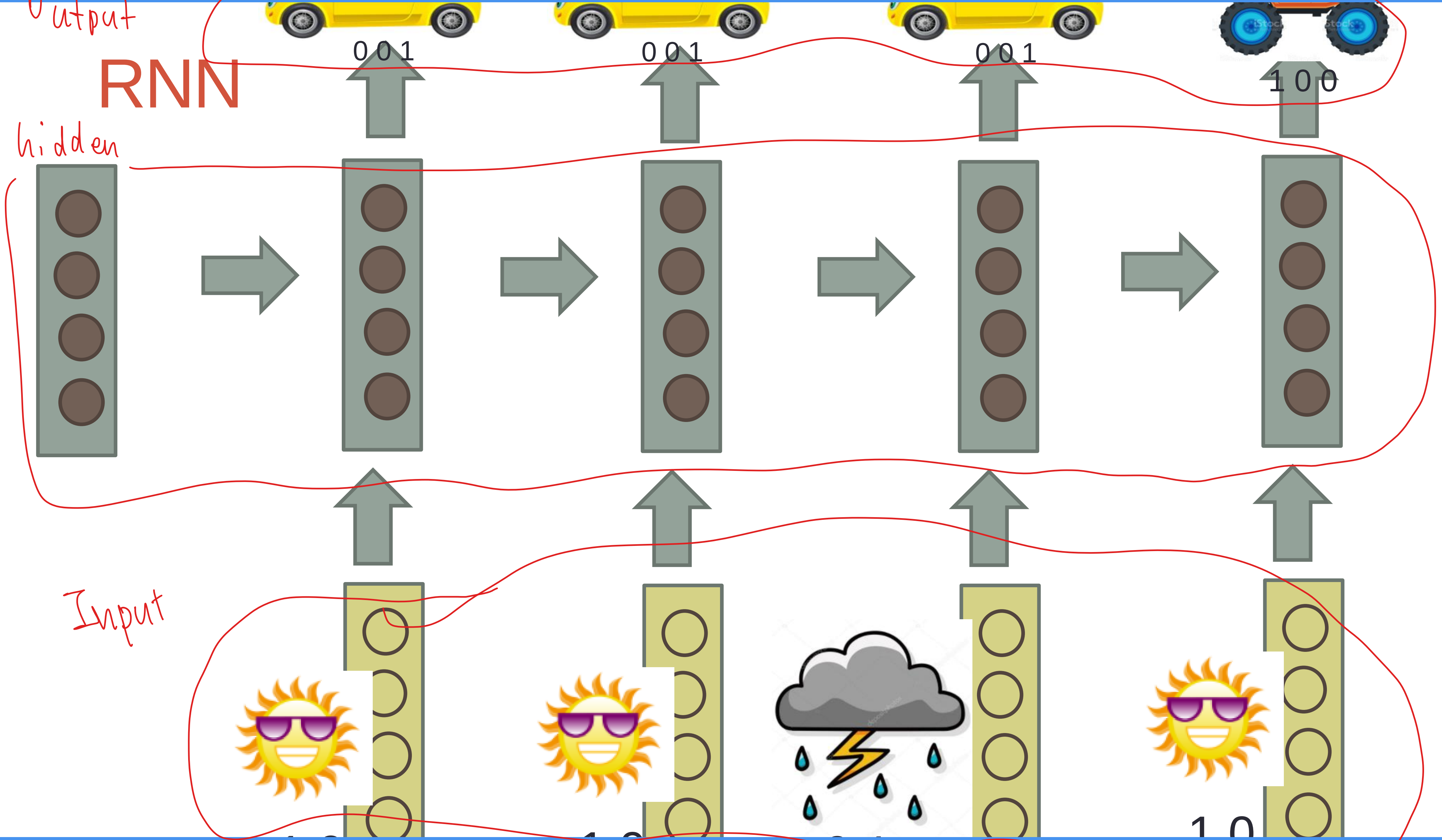


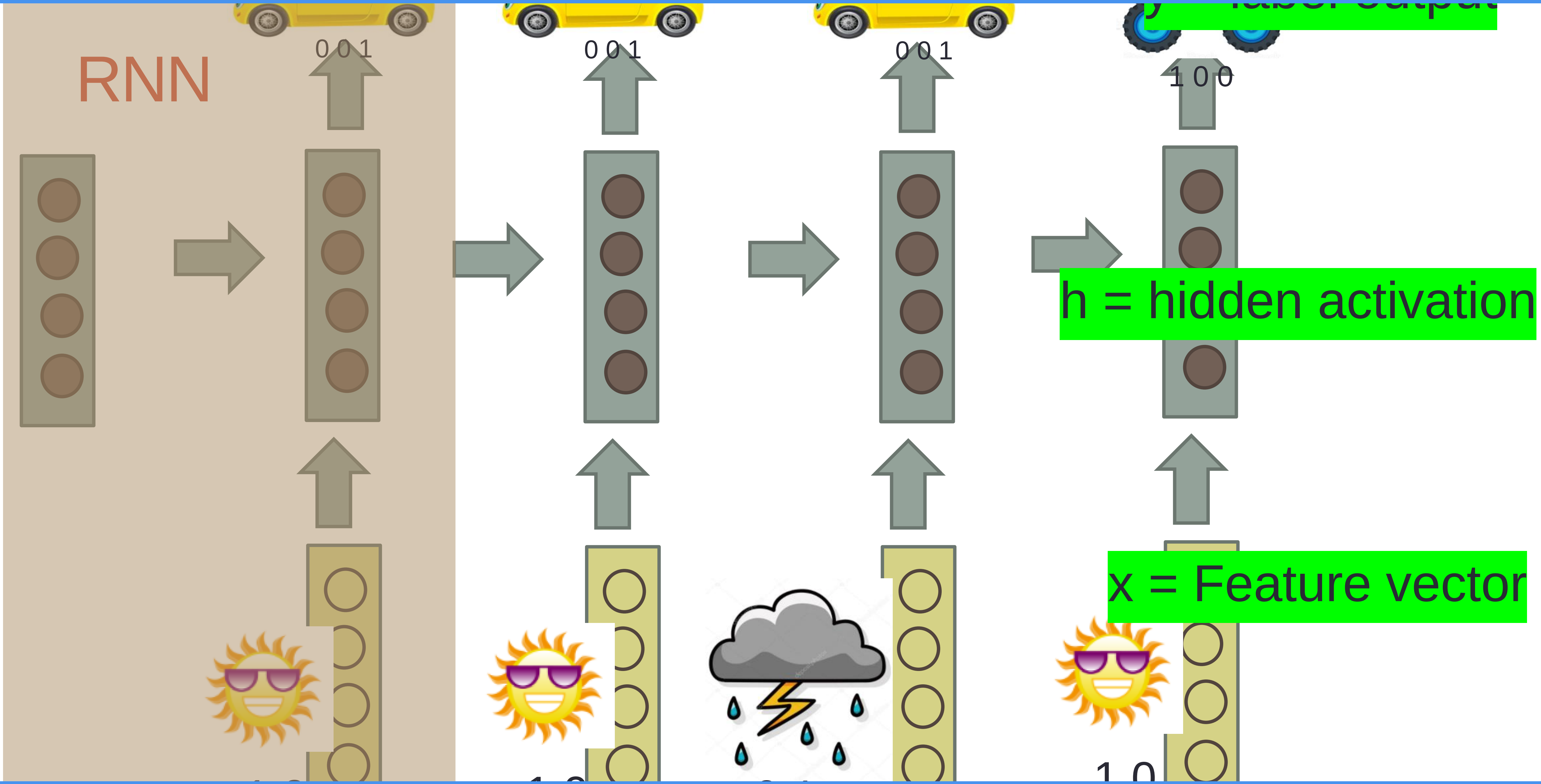

ส่งกลับเข้าไป train
0
0
1
0 0 1
0
1
1 0
0
1
0
0 1 0

Driving on Thursday

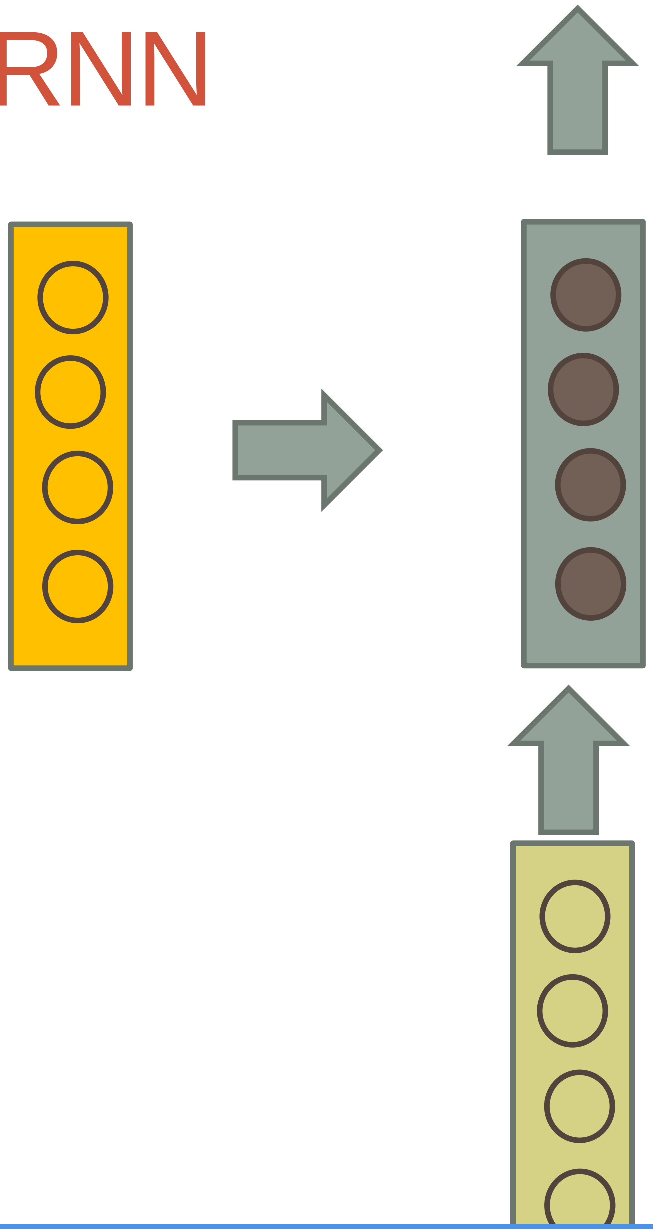


RNN

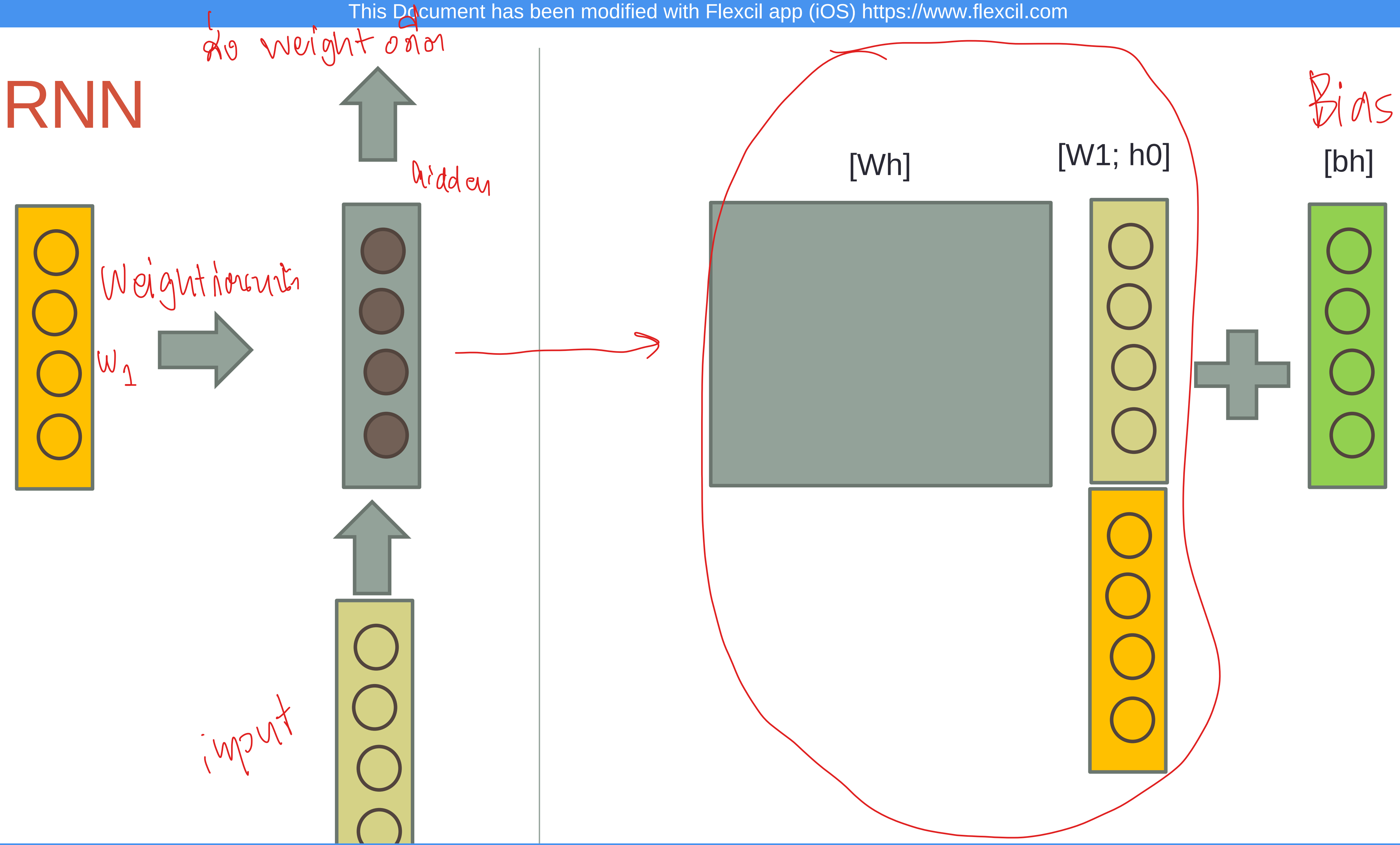



RNN
0 0 1
0 0 1
0 0 1
1 0 0
h = hidden activation
x = Feature vector
1 0

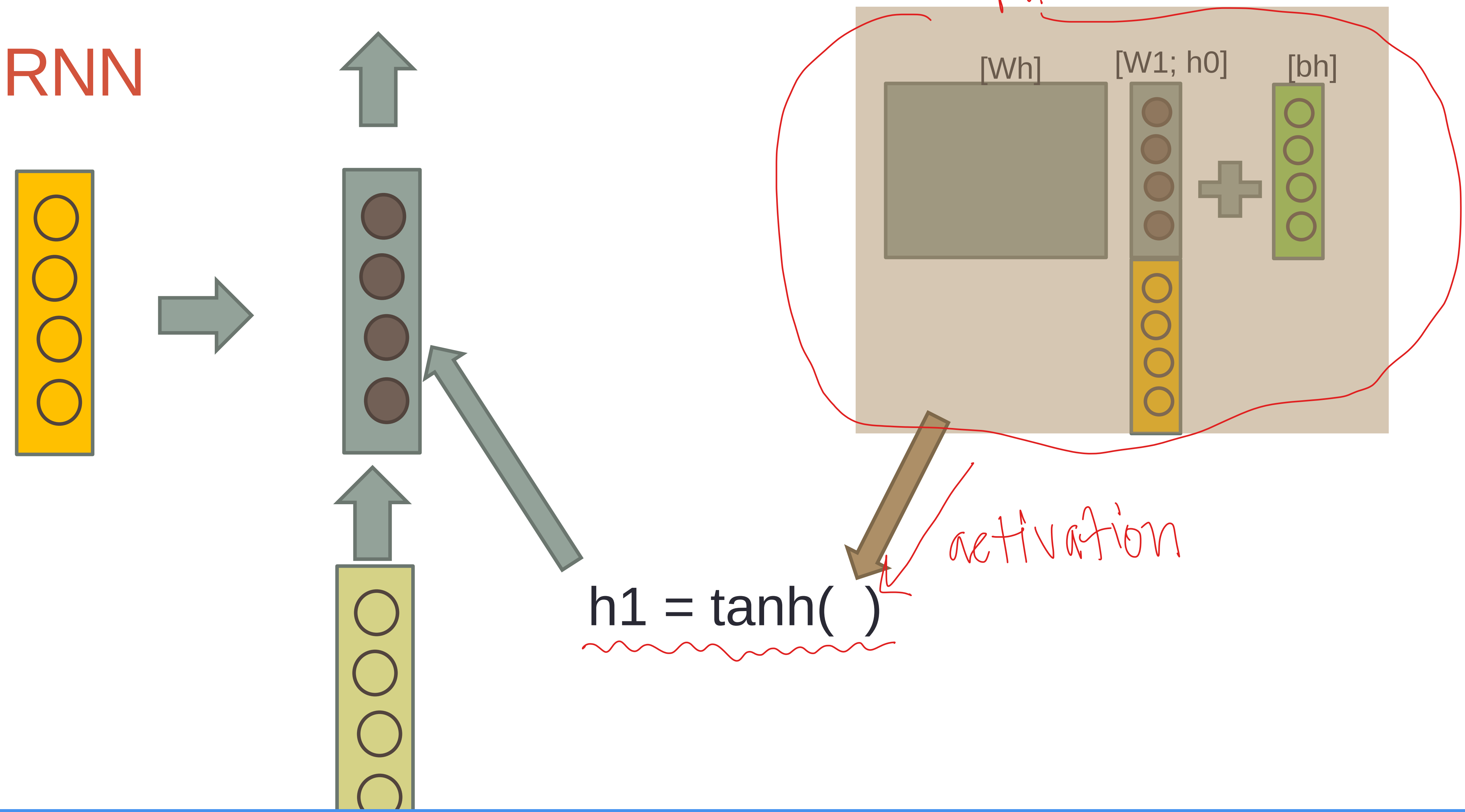
RNN



RNN

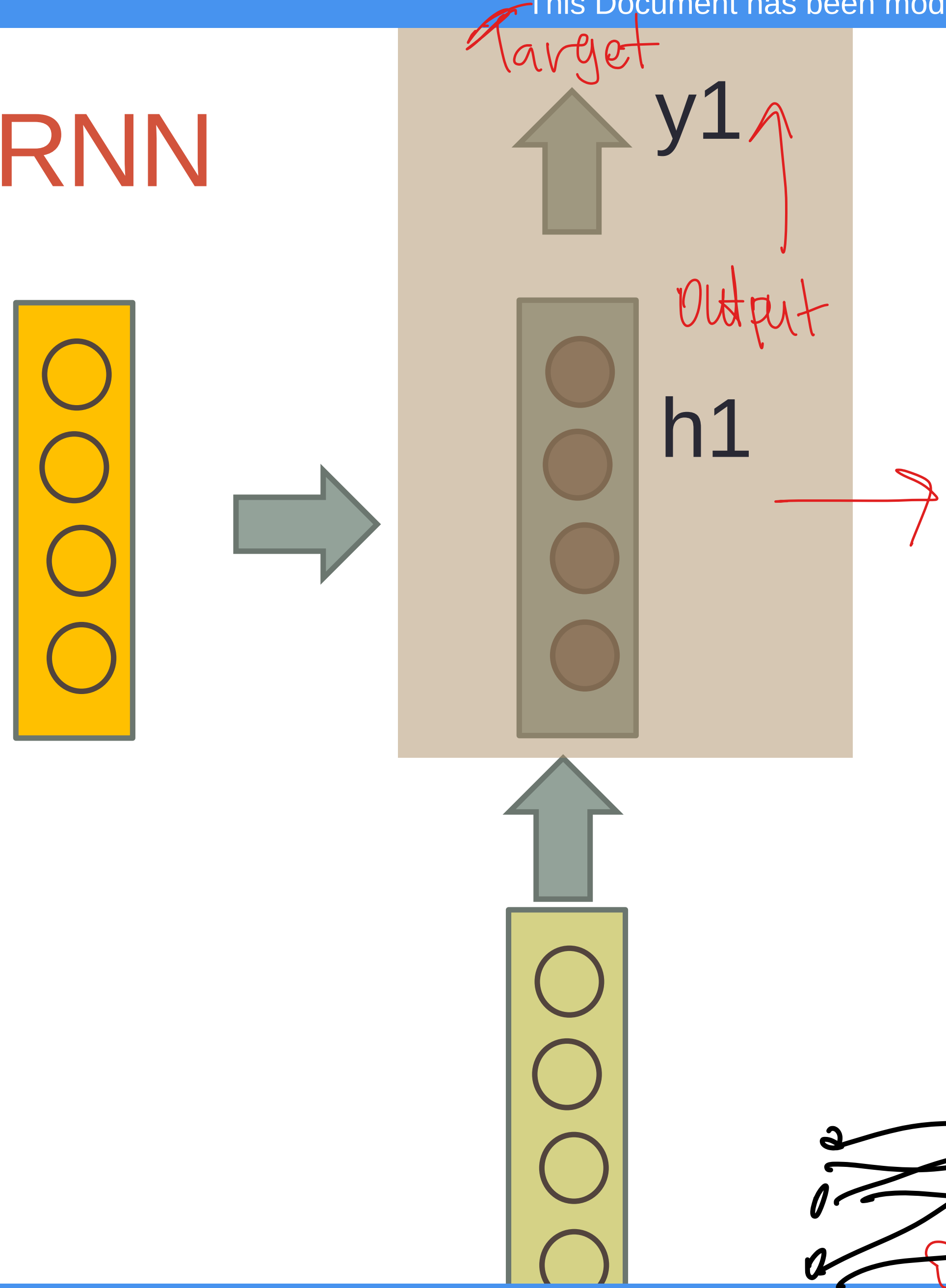


RNN



h1 = tanh( )

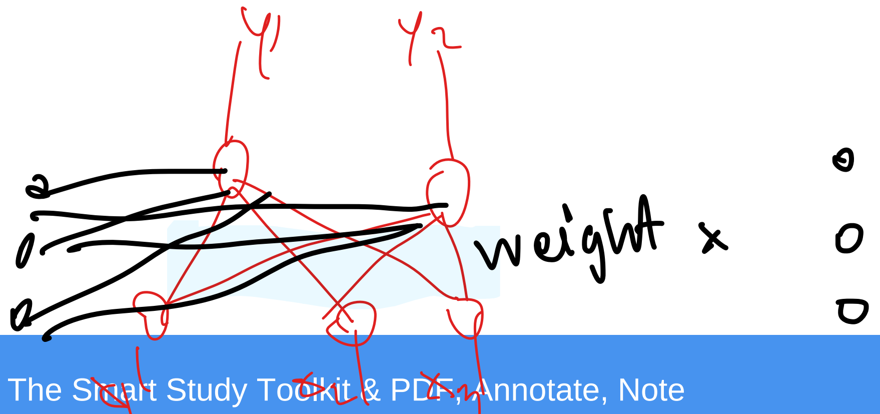
RNN



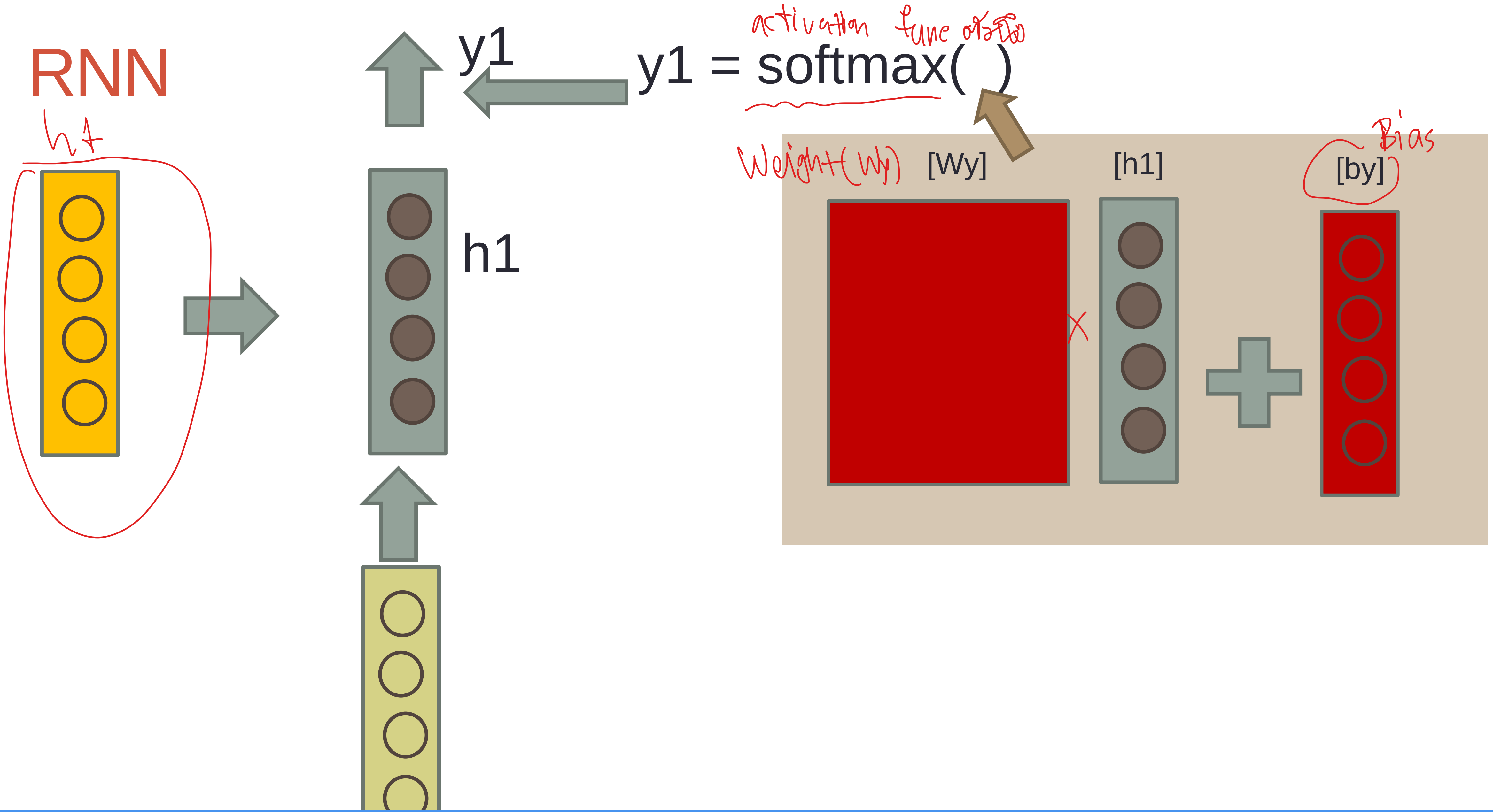
รูปของมัน RNN

สิ่งสำคัญ คือ weight ข้อมูลก่อนหน้า

ไปหา ปัจจุบัน แล้วส่งไปให้ตัวถัดไป



RNN


ht
y1
h1
activation func 에해당
y1 = softmax( )
Weight(Wy)
[Wy]
[h1]
Bias
[by]
x

Equations

hA ค่า นนัก

Bias h

$$h_t = \tanh(W_h \cdot [x_t; h_{t-1}]) + b_h$$

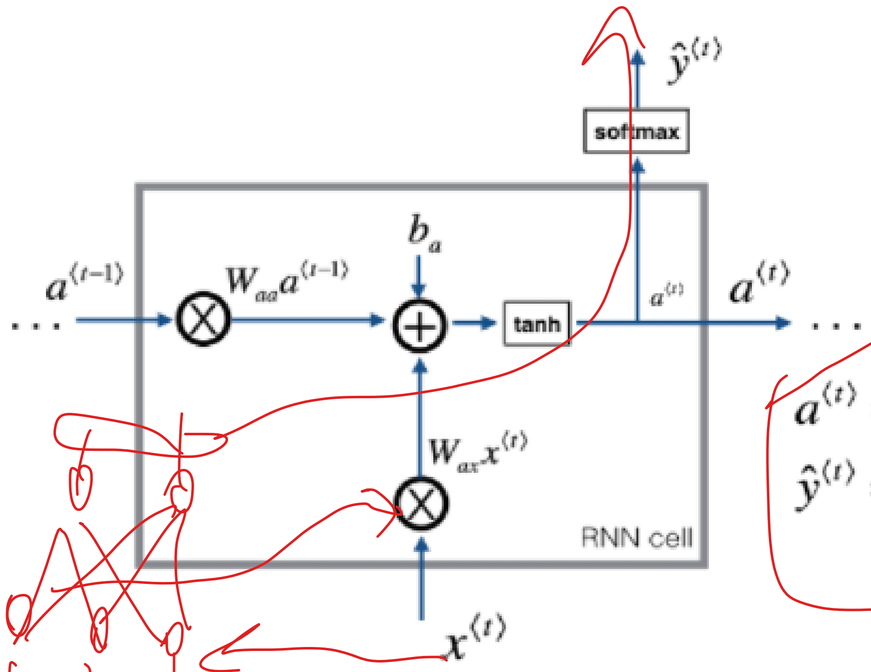
$$y_t = softmax(W_y \cdot h_t + b_y)$$

RNN CELL

A Recurrent neural network can be seen as the repetition of a single cell. You are first going to implement the computations for a single time-step. The following figure describes the operations for a single time-step of an RNN cell.

ถ้าไม่รู้ว่า RNN = NNแต่ง

ก็งงอยู่ดี



softmax
$\hat{y}^{\langle t \rangle}$
b_a
$a^{\langle t-1 \rangle}$
$W_{aa}a^{\langle t-1 \rangle}$
tanh
$a^{\langle t \rangle}$
$a^{\langle t \rangle}$
$W_{ax}x^{\langle t \rangle}$
RNN cell
$x^{\langle t \rangle}$

$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$
$$\hat{y}^{\langle t \rangle} = softmax(W_{ya}a^{\langle t \rangle} + b_y)$$

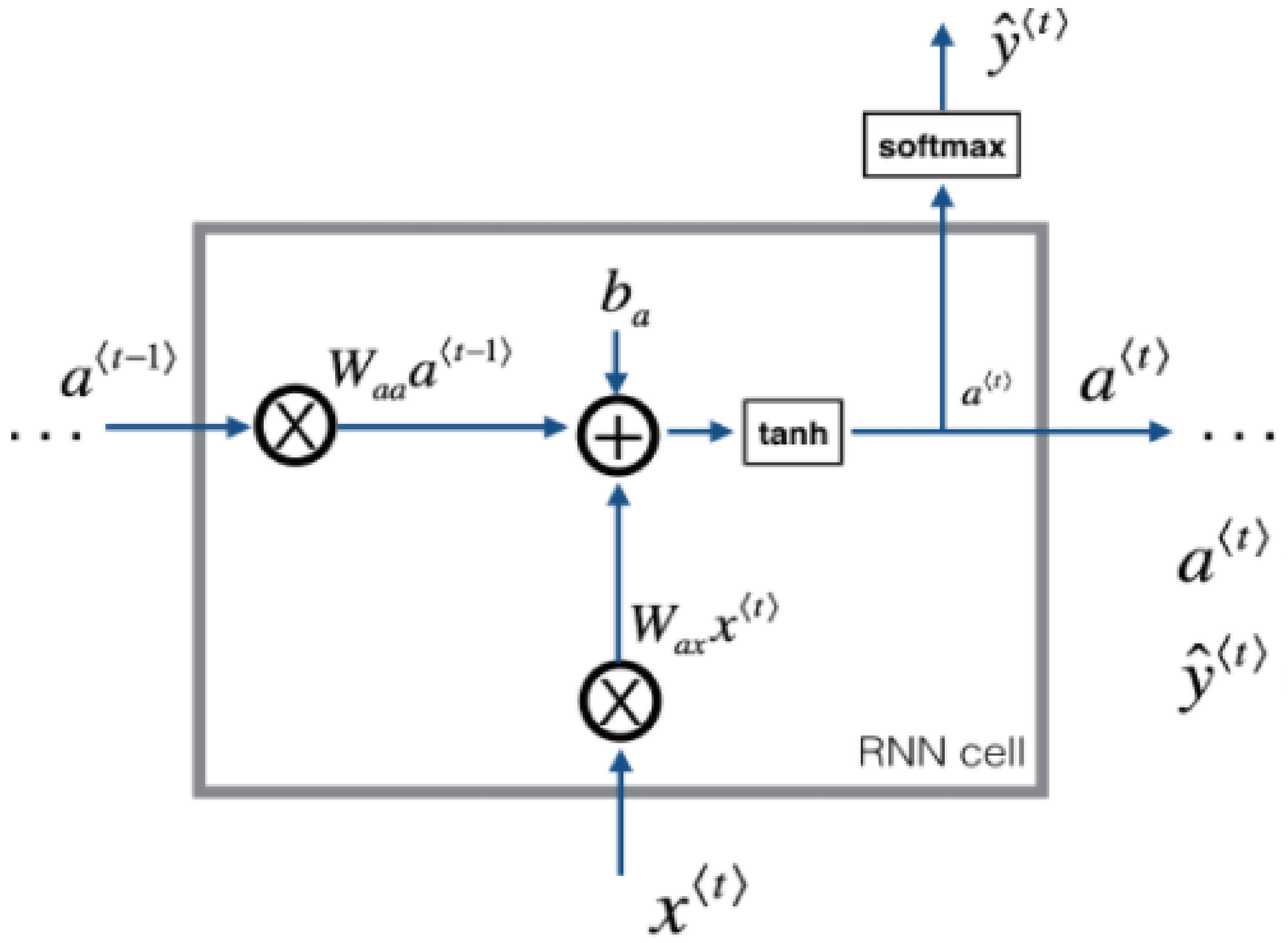
สมการ

Figure 2: Basic RNN cell. Takes as input $x^{\langle t \rangle}$ (current input) and $a^{\langle t-1 \rangle}$ (previous hidden state containing information from the past), and outputs $a^{\langle t \rangle}$ which is given to the next RNN cell and also used to predict $y^{\langle t \rangle}$

Exercise: Implement the RNN-cell described in Figure (2).

Instructions:

1. Compute the hidden state with tanh activation: $a^{\langle t \rangle} = \tanh(W_{aa}a^{\langle t-1 \rangle} + W_{ax}x^{\langle t \rangle} + b_a)$.
2. Using your new hidden state $a^{\langle t \rangle}$, compute the prediction $\hat{y}^{\langle t \rangle} = softmax(W_{ya}a^{\langle t \rangle} + b_y)$. We provided you a function: `softmax`.
3. Store $(a^{\langle t \rangle}, a^{\langle t-1 \rangle}, x^{\langle t \rangle}, parameters)$ in cache
4. Return $a^{\langle t \rangle}$, $y^{\langle t \rangle}$ and cache


$\hat{y}^{\langle t \rangle}$
softmax
b_a
$a^{\langle t-1 \rangle}$
$W_{aa}a^{\langle t-1 \rangle}$
tanh
$a^{\langle t \rangle}$
$a^{\langle t \rangle}$
$W_{ax}x^{\langle t \rangle}$
RNN cell
$x^{\langle t \rangle}$

ทดสอบการคำนวณ
ตาม code

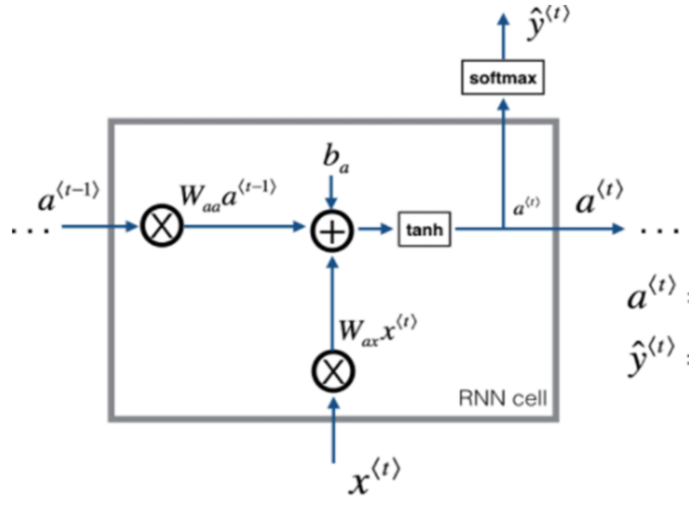
$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$

$$\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$$

RNN Cell

```
#RNN Cell
create_matrix_rand_val = function(nr,nc)
{
    #vr = runif(nr*nc)
    vr = rnorm(nr*nc)
    mr = matrix(vr, ncol=nc)
    return(mr)
}

set.seed(1)
xt = create_matrix_rand_val(3,10) #input data at timestep t

a_prev = create_matrix_rand_val(5,10) #Hidden state at timestep
t-1

Wax = create_matrix_rand_val(5,3)  #Weight matrix the input
Waa = create_matrix_rand_val(5,5)  #Weight matrix the input
Wya = create_matrix_rand_val(2,5)  #Weight matrix the hidden-
state to the output

ba = create_matrix_rand_val(5,1)   #Bias
by = create_matrix_rand_val(2,1)   #Bias relating the hidden-
state to the output

W_a = Waa %*% a_prev
W_x = Wax %*% xt
W_aW_xby = rowSums(W_a + W_x) + ba
a_next = tanh(W_aW_xby)
```

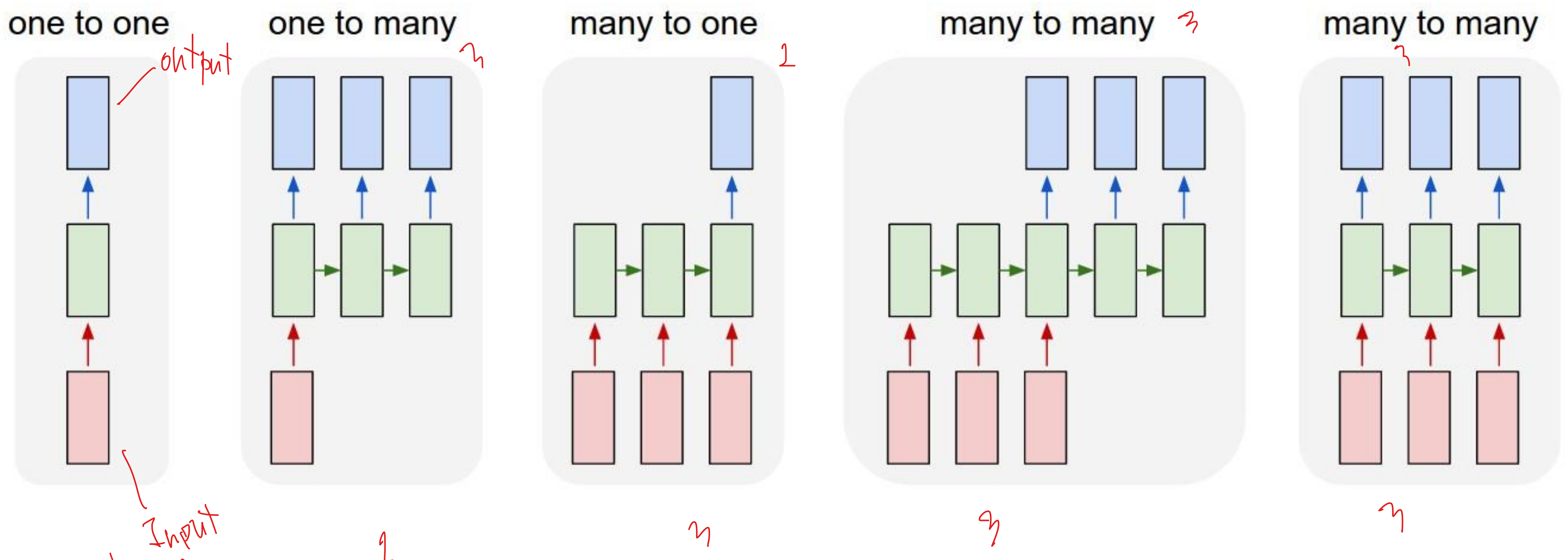
```
Wya_anext = Wya %*% a_next
library(sigmoid)
yt_pred = SoftMax(Wya_anext+by)

print(a_next)
print(yt_pred)
```


$\hat{y}^{\langle t \rangle}$
softmax
b_a
$a^{\langle t-1 \rangle}$
$W_{aa}a^{\langle t-1 \rangle}$
tanh
$a^{\langle t \rangle}$
$a^{\langle t \rangle}$
$W_{ax}x^{\langle t \rangle}$
RNN cell
$x^{\langle t \rangle}$

$$a^{\langle t \rangle} = \tanh(W_{ax}x^{\langle t \rangle} + W_{aa}a^{\langle t-1 \rangle} + b_a)$$
$$\hat{y}^{\langle t \rangle} = soft\max(W_{ya}a^{\langle t \rangle} + b_y)$$

RNNs

1 cell



ไม่มีอดีต มาเกี่ยวข้อง

Summary

ไม่มีการบ้าน

ไม่มีการบ้าน