PES UNIVERSITY

UE21CS343BB2
Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Anirudh Chandrasekar, Teaching Assistant

Topics in Deep Learning

RNN Architecture

Data for RNN

The data fed to RNN is of the form:

(timesteps, input_features)

Let's look at an example: Movie Reviews

Review	Sentiment
movie was good	1
movie was bad	0
movie was not good	0

Vocabulary is made up of 5 words.

Data for RNN

Input needs to be in vector format.

- One Hot Encoding

Review	Sentiment
movie was good	1
movie was bad	0
movie was not good	0

movie	[1,0,0,0,0]
was	[0,1,0,0,0]
good	[0,0,1,0,0]
bad	[0,0,0,1,0]
not	[0,0,0,0,1]

So review 1 "movie was good" becomes:
[[1,0,0,0,0],[0,1,0,0,0],[0,0,1,0,0]]

Review 1 has the shape of (3,5), where
3 is the number of timesteps
5 is the number of features

How RNN works?

Much like ANNs, RNNs also has an input, hidden and output layer.
But there are 2 major differences:

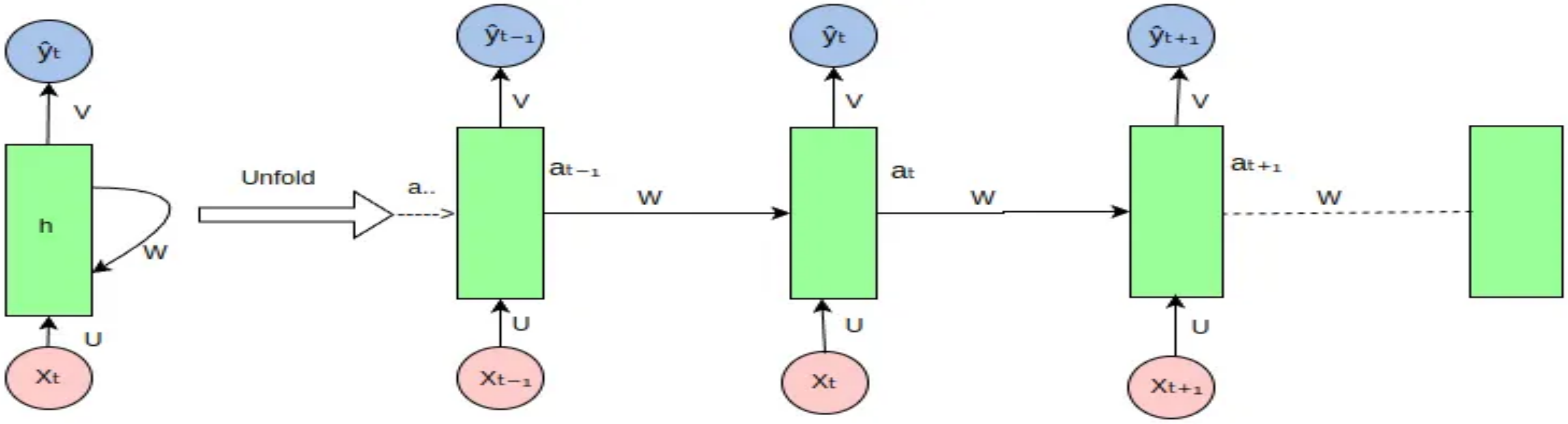
1. Input is fed to the network on a timestep basis i.e. at t=1 we send X_{11} to the network, at t=2 we send X_{22} and so on.
2. ANN is feed-forward whereas RNNs have a concept of state.

Review (X)	Sentiment
X_1 movie was good	1
X_2 movie was bad	0
X_3 movie was not good	0

movie	[1,0,0,0,0]
was	[0,1,0,0,0]
good	[0,0,1,0,0]
bad	[0,0,0,1,0]
not	[0,0,0,0,1]

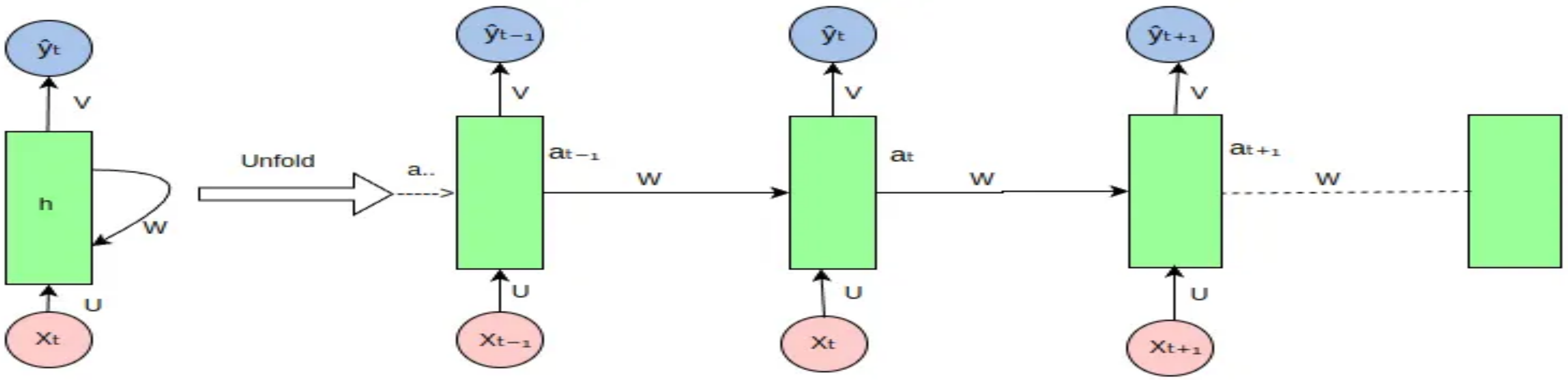
RNN Architecture

- The RNN takes an input vector X and the network generates an output vector y by scanning the data sequentially from left to right, with each time step updating the hidden state and producing an output.



RNN Architecture

- U represents the weight parameter between input layer(X) and hidden layer(h).
- W represents the weight associated with the connection between the hidden layers.
- V represents the weight parameter between the hidden layer(h) and the output layer(y).



RNN Architecture

- The dimensions of the RNN components are as follows:

  $x_i \in R^n$ (n-dimensional input)

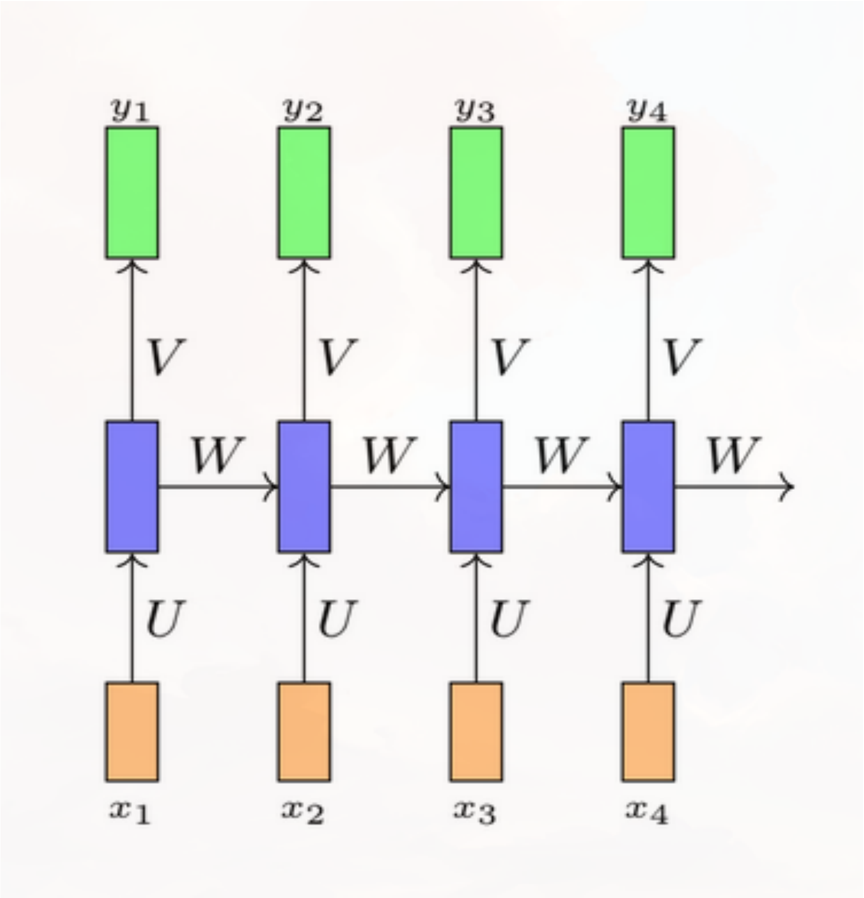
  $h_i \in R^d$ (d-dimensional state)

  $y_i \in R^k$ (say k classes)

  $U \in R^{nxd}$

  $V \in R^{dxk}$

  $W \in R^{dxd}$



Forward Propagation Example

We pass X_1 as input:

At t=1 X_{11} is the input to the network.

The hidden state h_1 is computed as

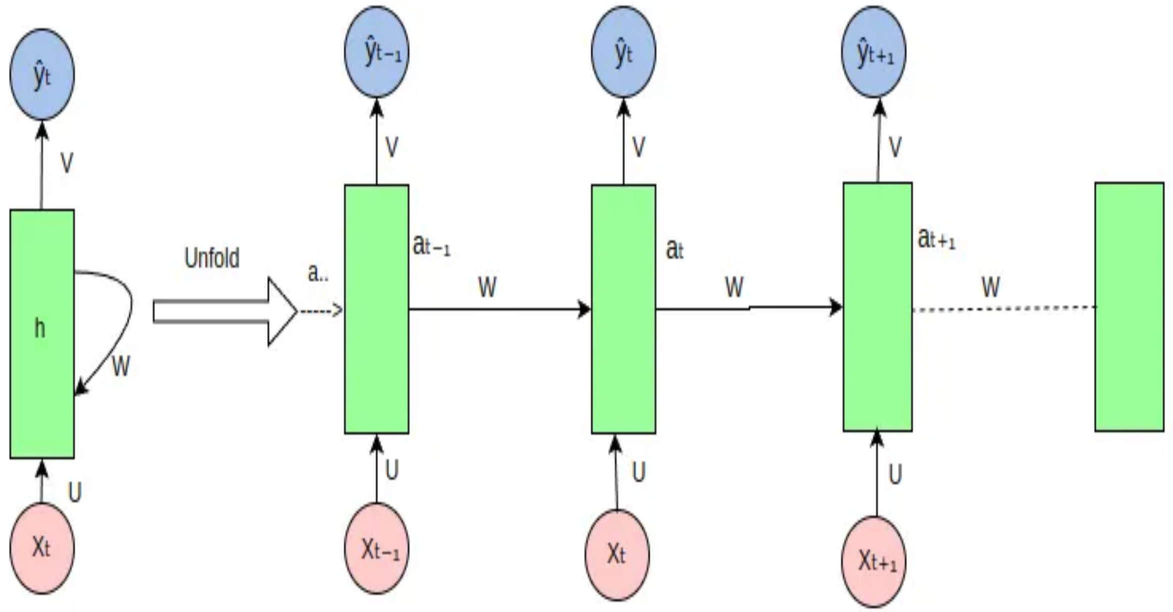
$$h_1 = f(U * X_{11} + W * h_0 + b)$$

h_0 can be initialized to 0 or a predefined value.

The output $\hat{y}_1$ is computed as

$$\hat{y}_1 = g(V * h_1 + c)$$

Review (X)	Sentiment
$X_{11}\ X_{12}\ X_{13}$	1
$X_{21}\ X_{22}\ X_{23}$	0
$X_{31}\ X_{32}\ X_{33}\ X_{34}$	0



Forward Propagation

PES UNIVERSITY

We pass X_1 as input:

At t=2 X_{12} is the input to the network.

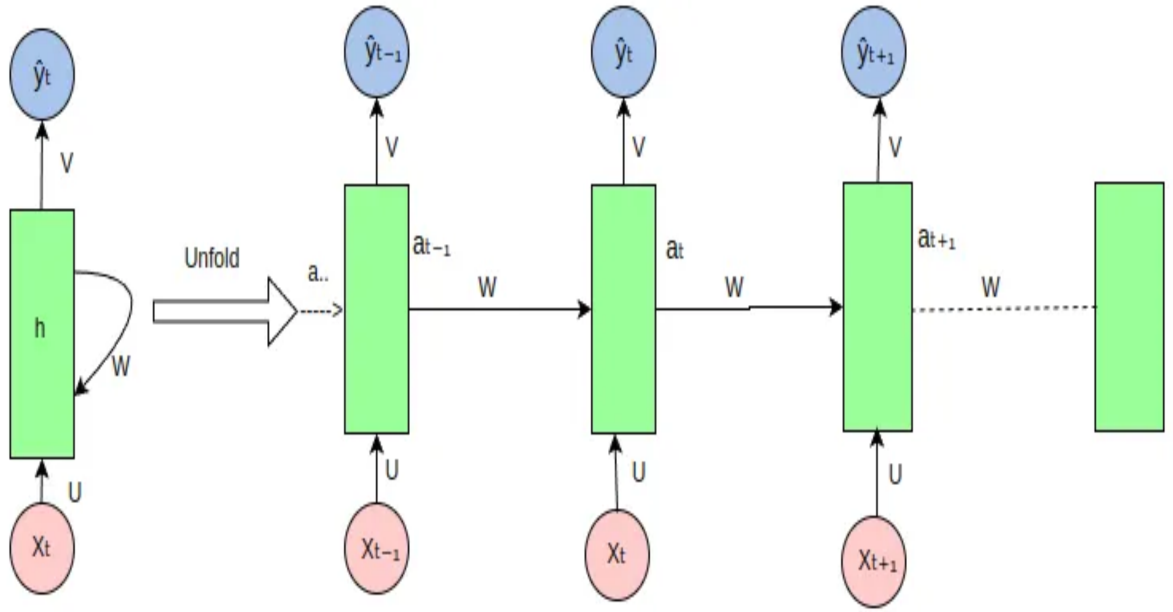
The hidden state h_1 is computed as

$$h_2 = f(U * X_{12} + W * h_1 + b)$$

The output $\hat{y}_2$ is computed as

$$\hat{y}_2 = g(V * h_2 + c)$$

Review (X)	Sentiment
X_{11} X_{12} X_{13}	1
X_{21} X_{22} X_{23}	0
X_{31} X_{32} X_{33} X_{34}	0



Forward Propagation

We pass X_1 as input:

At t=3 X_{13} is the input to the network.

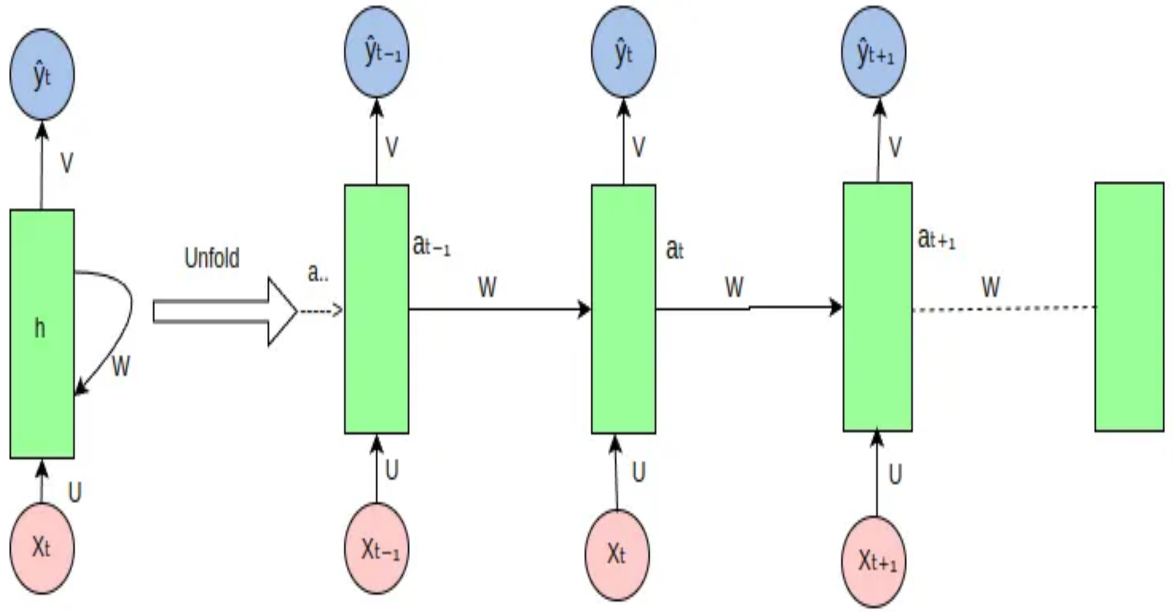
The hidden state h_1 is computed as

$$h_3 = f(U * X_{13} + W^* h_2 + b)$$

The output $\hat{y}_3$ is computed as

$$\hat{y}_3 = g(V * h_3 + c)$$

Review (X)	Sentiment
X_{11} X_{12} X_{13}	1
X_{21} X_{22} X_{23}	0
X_{31} X_{32} X_{33} X_{34}	0



RNN Architecture

- At each time step t, the hidden state h_t is given by:

$$h_t = f(U * X_t + W * h_{t-1} + b)$$

where

X_t is the input at time step t.

h_{t-1} is the output from hidden layer at time t-1.

b is the bias vector for the hidden layer.

f is the activation function.

U is the weight matrix between input and hidden layer.

W is the weight matrix between hidden layers.

- h_0 can be set to 0 or some predefined value.

RNN Architecture

- The output at each time step t, denoted as y_t is computed based on the hidden state output h_t using the following formula:

$$\hat{y}_t = g(V * h_t + c)$$

where

$\hat{y}_t$ is the output predicted at time step t.

V is the weight matrix governing the connections from the hidden layer to the output layer.

b is the bias vector for the output layer.

g is the activation function.

RNN Architecture Code

RNN_Architecture.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
[1] from keras import Sequential
    from keras.layers import Dense,SimpleRNN
```

```
[2] model = Sequential()

    model.add(SimpleRNN(3,input_shape = (4,5)))
    model.add(Dense(1,activation='sigmoid'))

    model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 simple_rnn (SimpleRNN)      (None, 3)                 27

 dense (Dense)               (None, 1)                 4

=================================================================
Total params: 31 (124.00 Byte)
Trainable params: 31 (124.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

RNN Architecture Code

```
print(model.get_weights()[0].shape) #Matrix U
model.get_weights()[0]
```

```
(5, 3)
array([[ 0.24314839,  0.65584403,  0.12467551],
       [ 0.07774234, -0.6701355 , -0.5714668 ],
       [-0.15539342, -0.37312293, -0.5985743 ],
       [ 0.7748706 ,  0.15962046, -0.01933241],
       [-0.4337626 , -0.1516878 ,  0.6292947 ]], dtype=float32)
```

[4]
```
print(model.get_weights()[1].shape) #Matrix W
model.get_weights()[1]
```

```
(3, 3)
array([[-0.95396507,  0.01400834,  0.29959065],
       [ 0.23049797,  0.67335117,  0.70247334],
       [-0.19188926,  0.73919   , -0.6455824 ]], dtype=float32)
```

[5]
```
print(model.get_weights()[2].shape) #Bias b
model.get_weights()[2]
```

```
(3,)
array([0., 0., 0.], dtype=float32)
```

RNN Architecture Code

```
print(model.get_weights()[3].shape) #Matrix V
model.get_weights()[3]
```

```
(3, 1)
array([[ 0.45543277],
       [-1.0071795 ],
       [-0.80624163]], dtype=float32)
```

```
print(model.get_weights()[4].shape) #Bias c
model.get_weights()[4]
```

```
(1,)
array([0.], dtype=float32)
```

RNN Architecture Code

Link to notebook:
https://colab.research.google.com/drive/1Eq42O0PiQWZsJDdSwLIR0iZqTzQ9oW3u?usp=sharing

Acknowledgements & References

- https://deeplearning.ai
- https://www.youtube.com/watch?v=BjWqCcbusMM

UE21CS343BB2
Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Anirudh Chandrasekar, Teaching Assistant