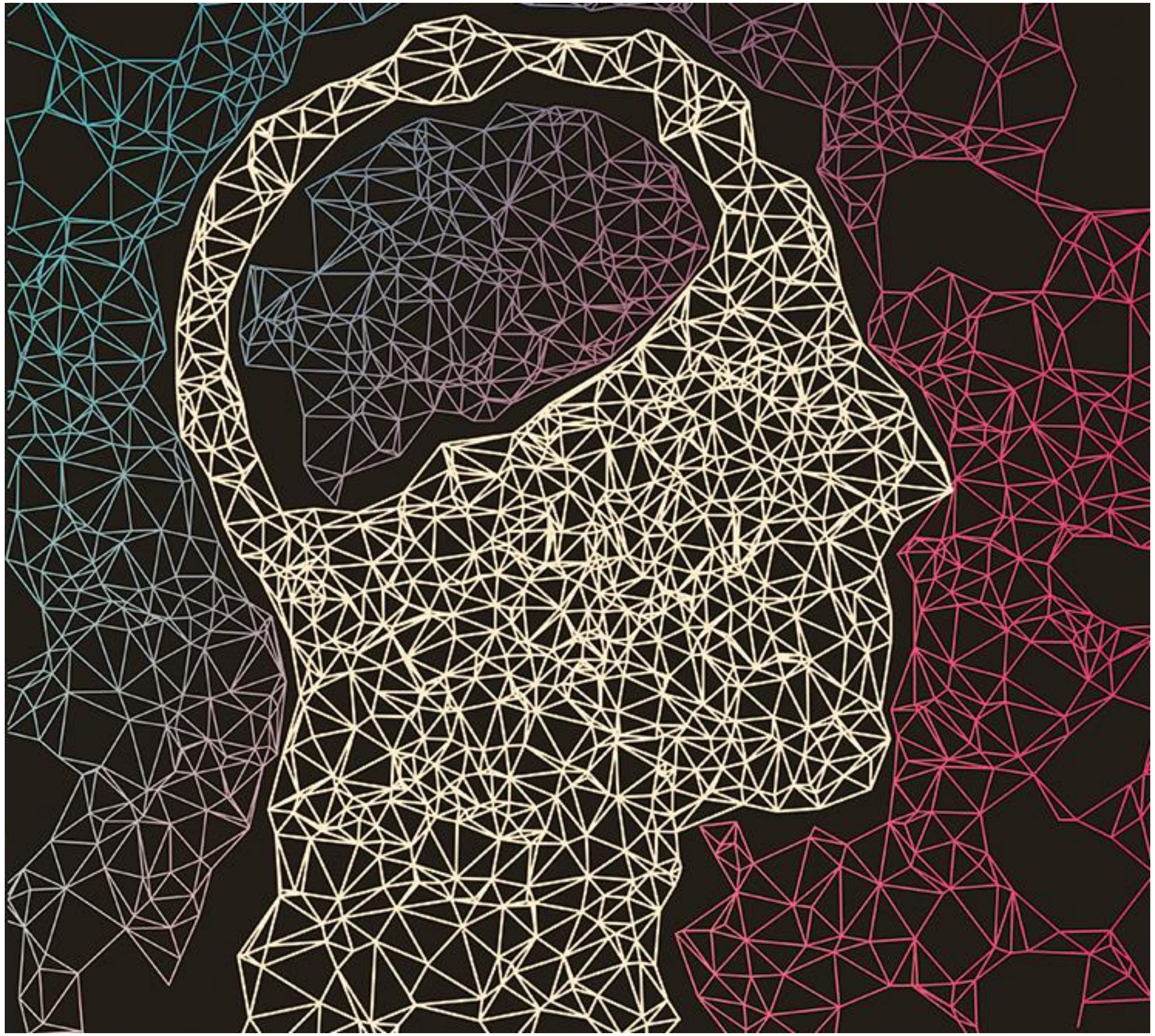


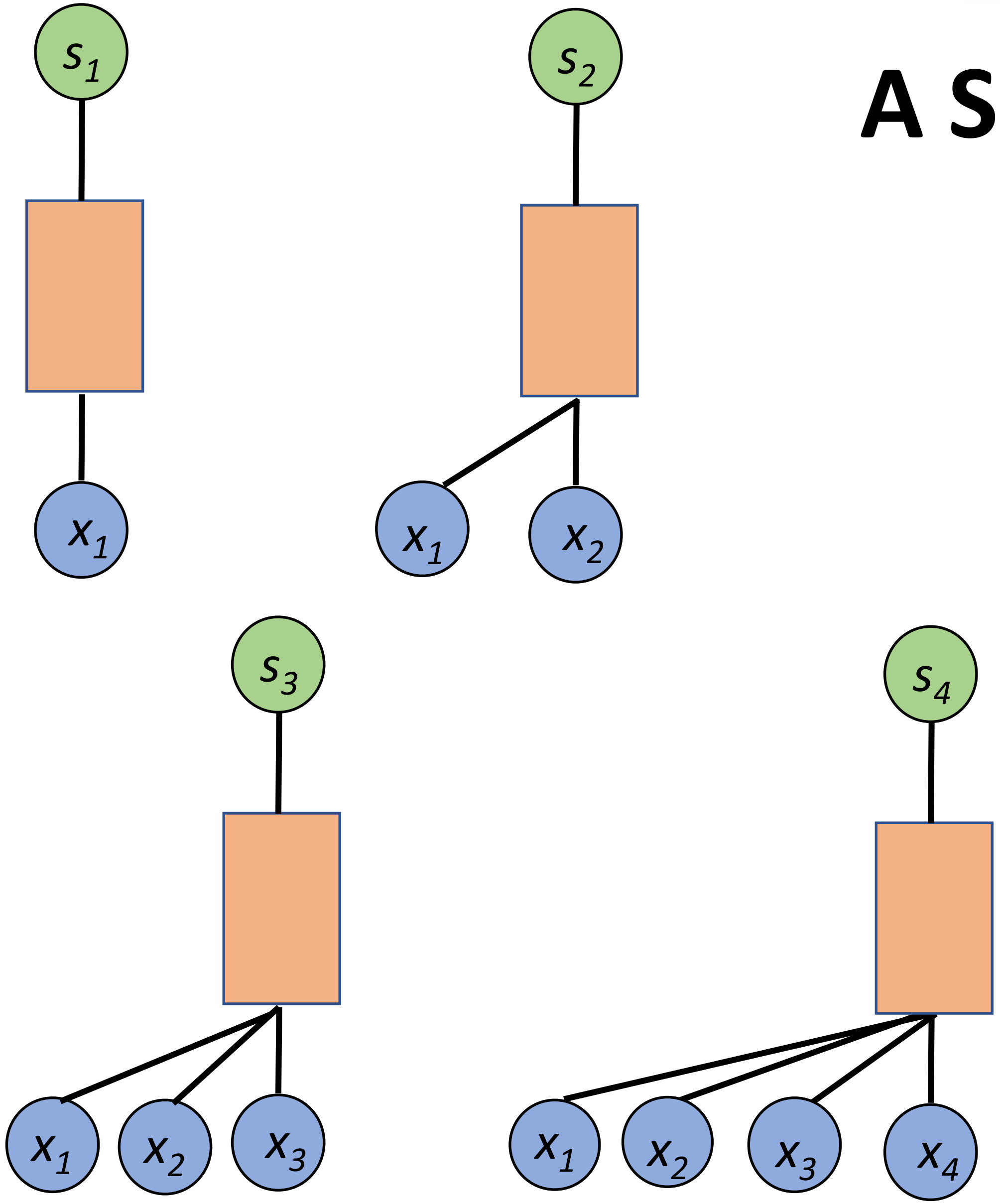
Deep Learning

Course Code:

Sequential models & Recurrent Neural Networks (RNNs)

Introduction to RNNs and their applications

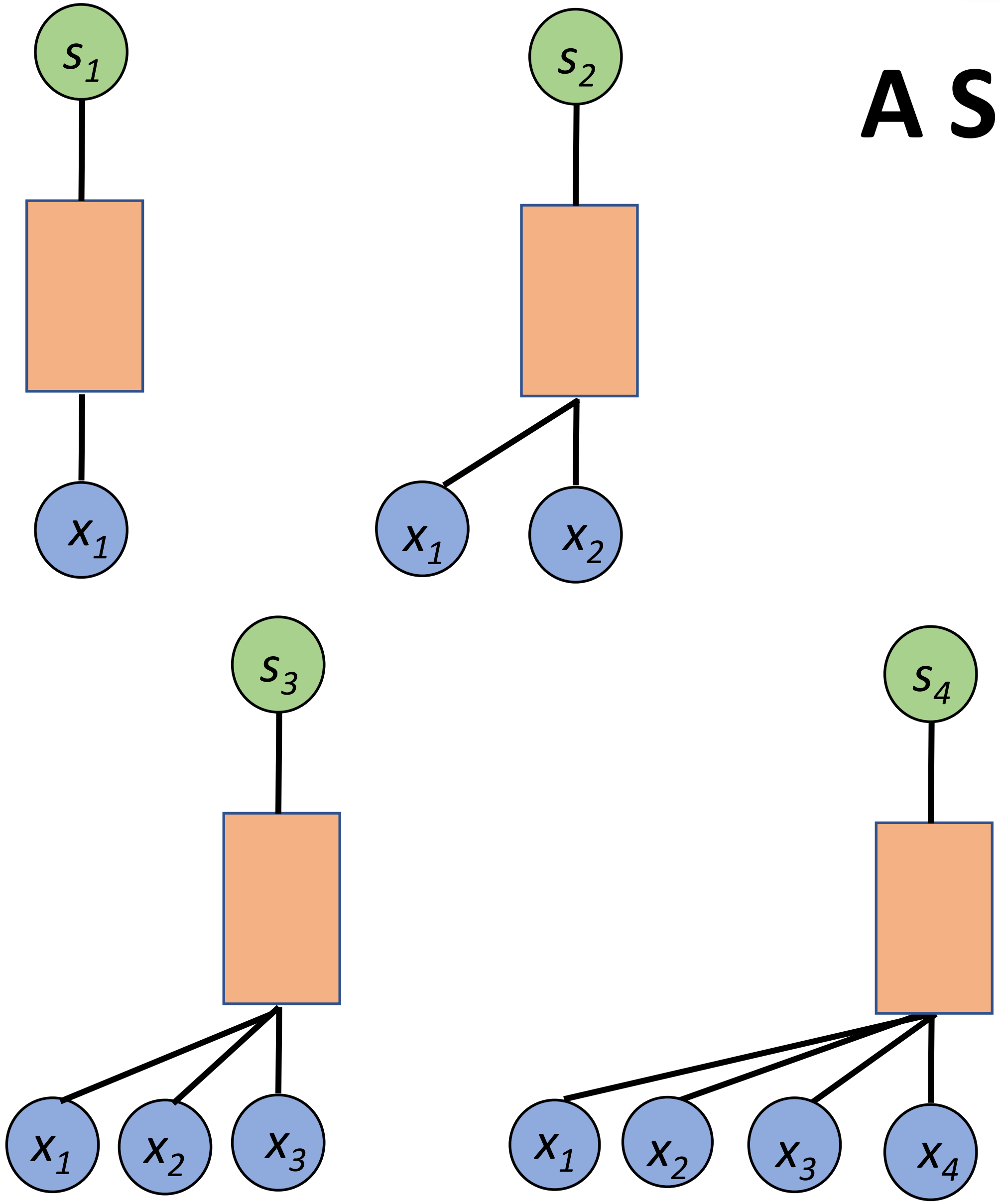
A Simple Approach



Problem

- Different function for different time-step

$$s_1 = f_1(x_1)$$
$$s_2 = f_2(x_1,x_2)$$
$$s_3 = f_3(x_1,x_2,x_3) \;\ldots\ldots$$



A Simple Approach

Problem

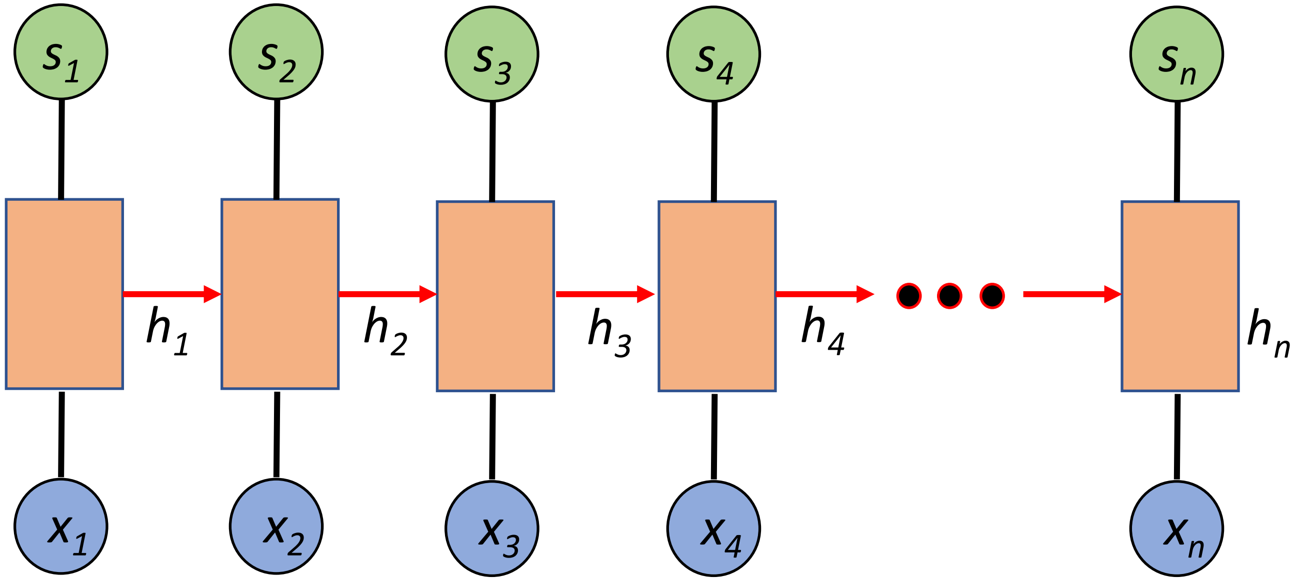
- Different function for different time-step

  $s_1 = f_1(x_1)$

  $s_2 = f_2(x_1,x_2)$

  $s_3 = f_3(x_1,x_2,x_3)$ ……
- Depends on input length

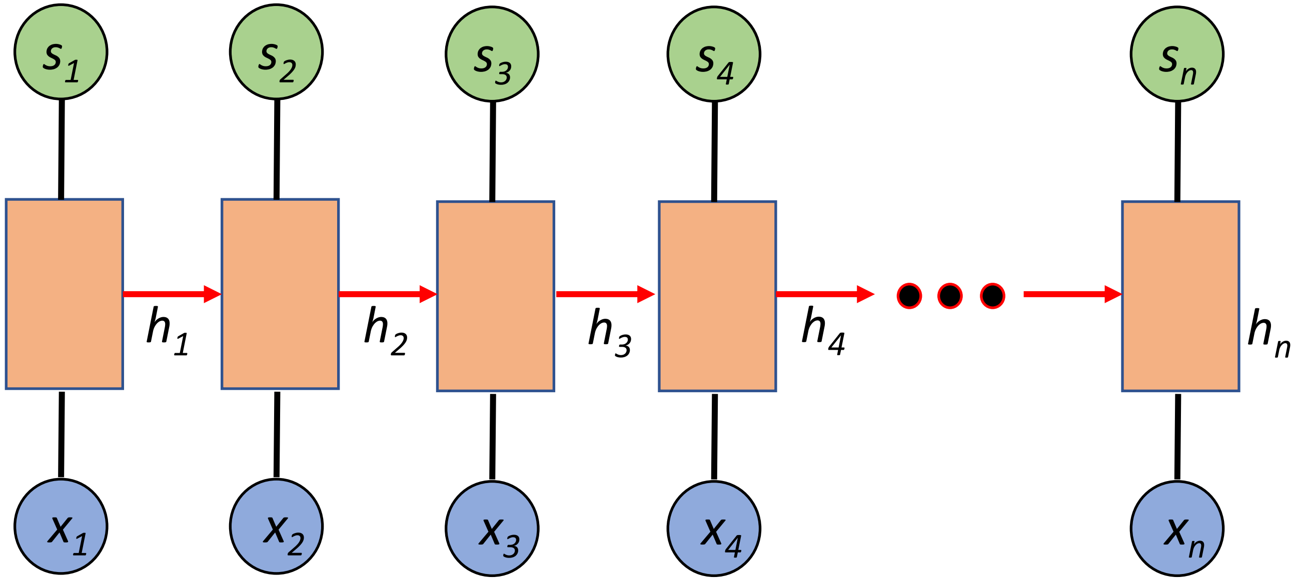
Recurrent Neural Network



Solution

Add recurrent connection

Recurrent Neural Network

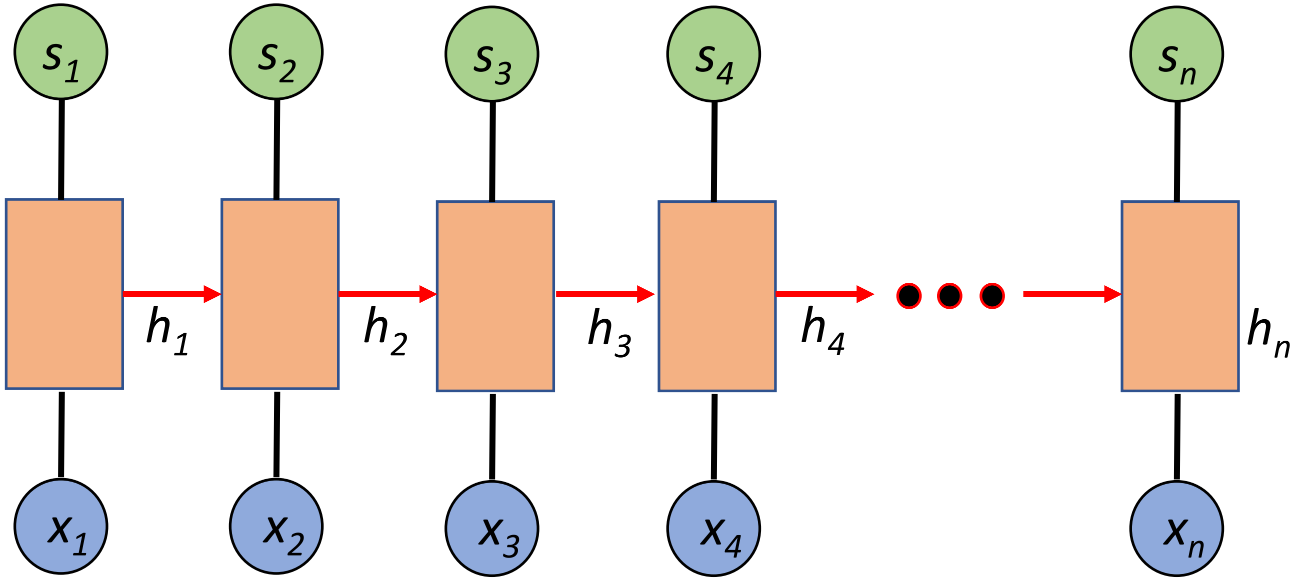


Solution

Add recurrent connection

$$s_n = f(x_n, h_{n-1})$$

Recurrent Neural Network



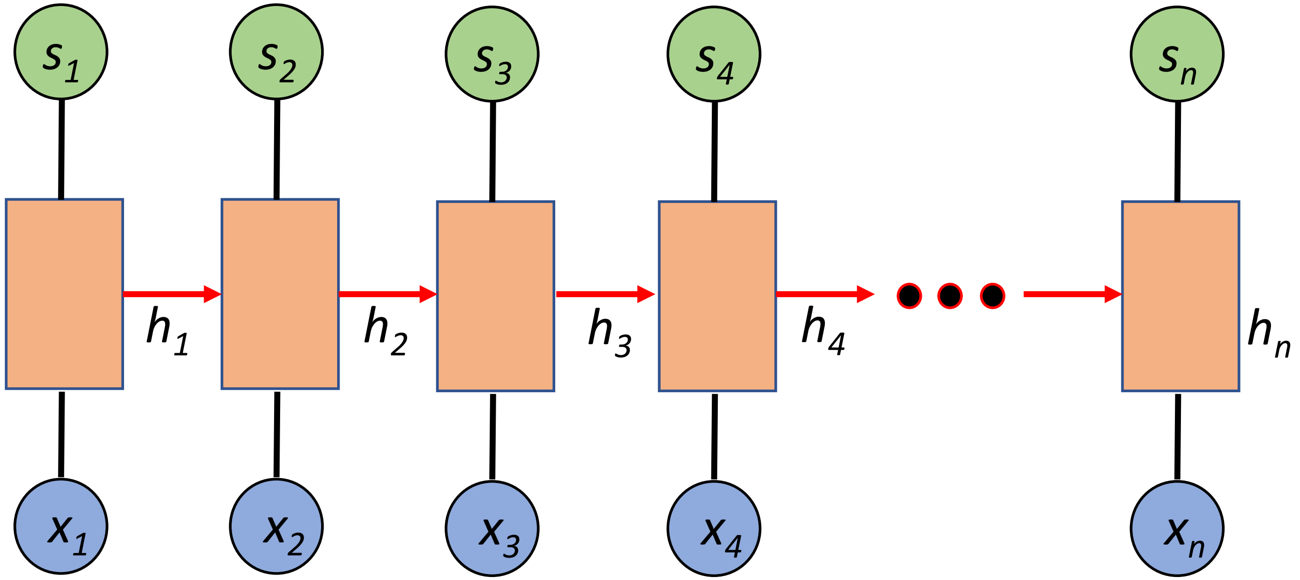
Solution

Add recurrent connection

$$s_n = f(x_n, h_{n-1})$$

input

Recurrent Neural Network



Solution

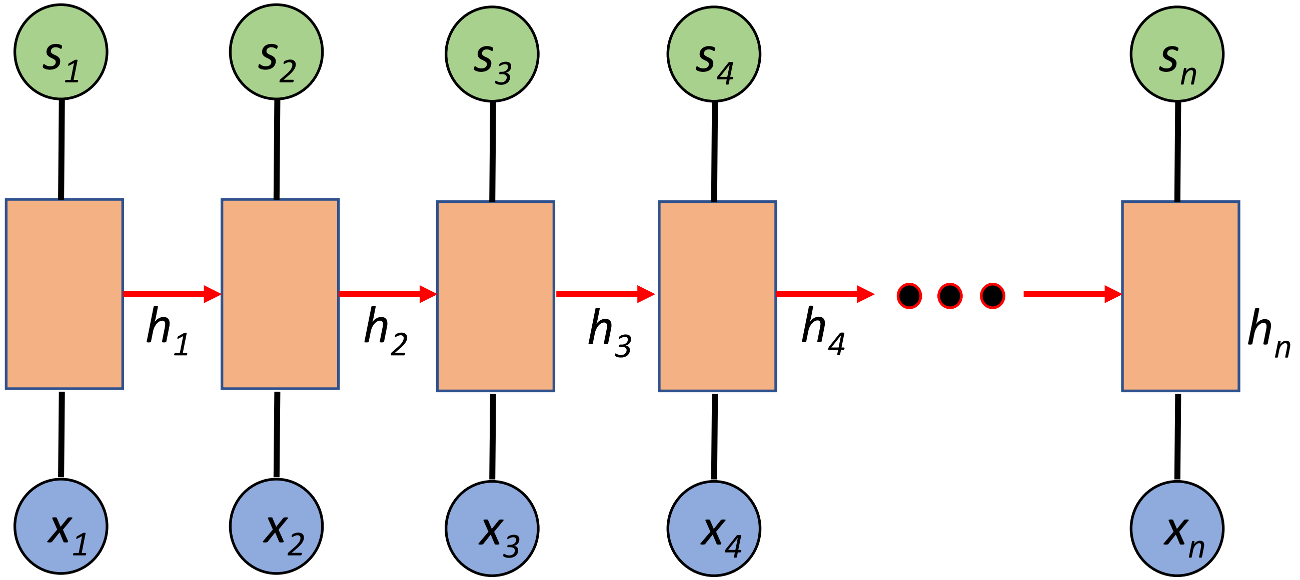
Add recurrent connection

$$s_n = f(x_n, h_{n-1})$$

output

input

Recurrent Neural Network



Solution

Add recurrent connection

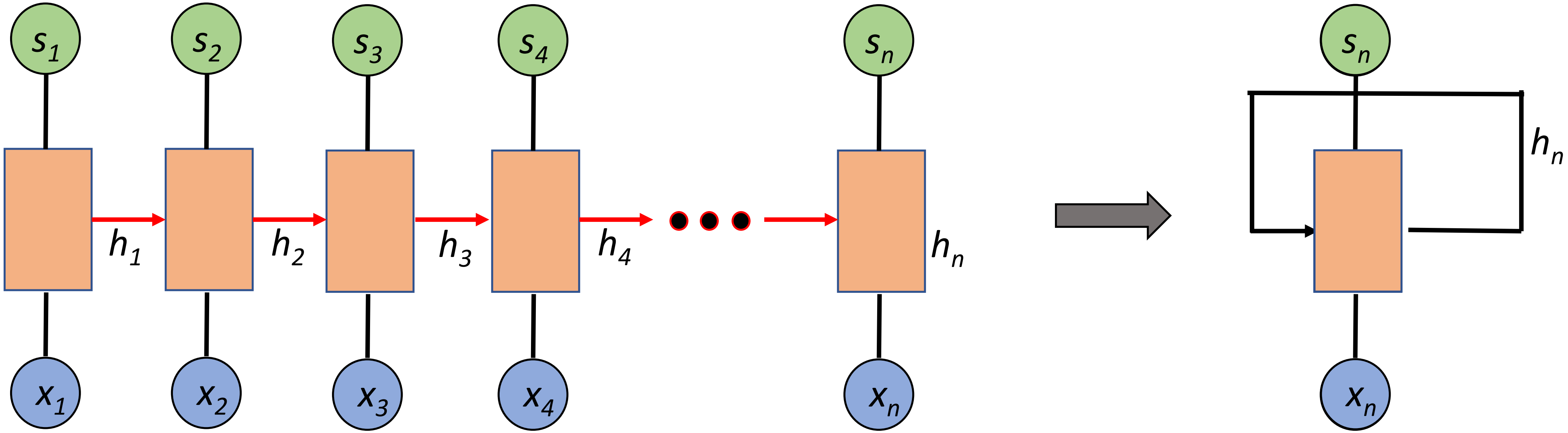
$$s_n = f(x_n, h_{n-1})$$

output

input

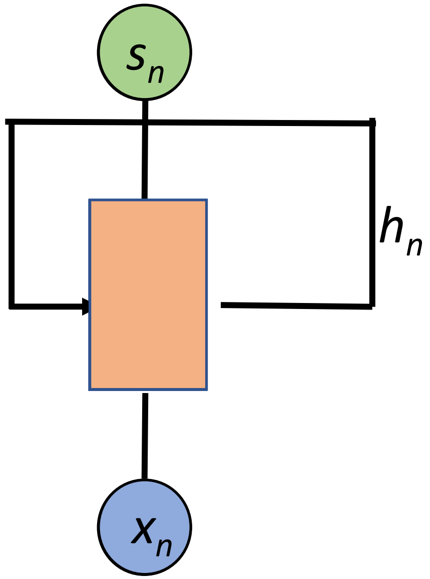
past memory

Recurrent Neural Network



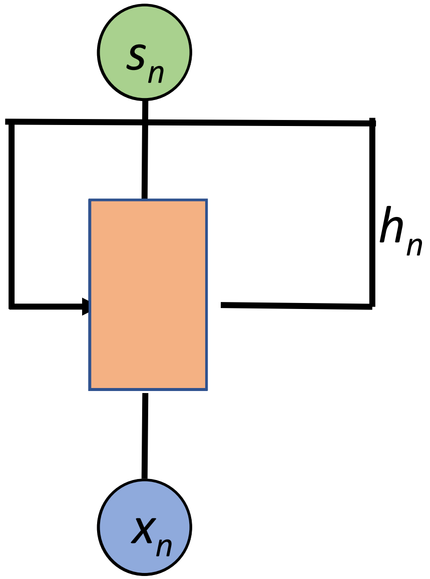
Can be represented more compactly

Recurrent Neural Network



A sequence of vectors is processed by applying a recurrence formula at each time step.

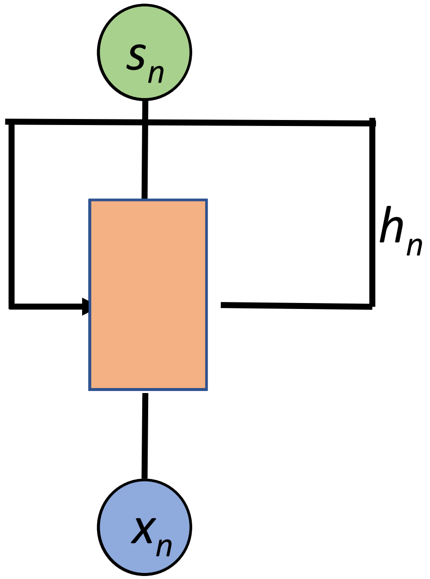
Recurrent Neural Network



A sequence of vectors is processed by applying a recurrence formula at each time step.

$$h_n = f_W(x_n, h_{n-1})$$

Recurrent Neural Network

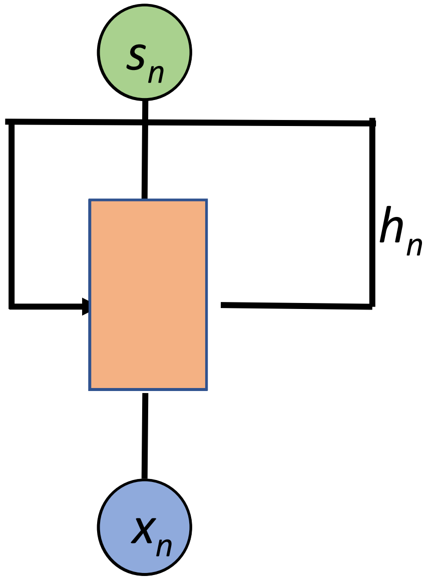


A sequence of vectors is processed by applying a recurrence formula at each time step.

$$h_n = f_W(x_n, h_{n-1})$$

Hidden state

Recurrent Neural Network



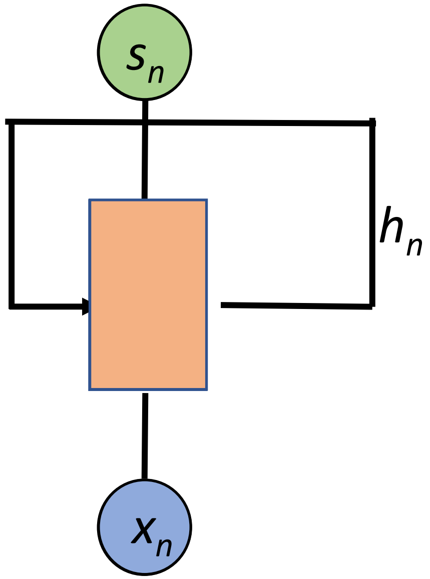
A sequence of vectors is processed by applying a recurrence formula at each time step.

$$h_n = f_W(x_n, h_{n-1})$$

Hidden state

Function with weight W

Recurrent Neural Network



A sequence of vectors is processed by applying a recurrence formula at each time step.

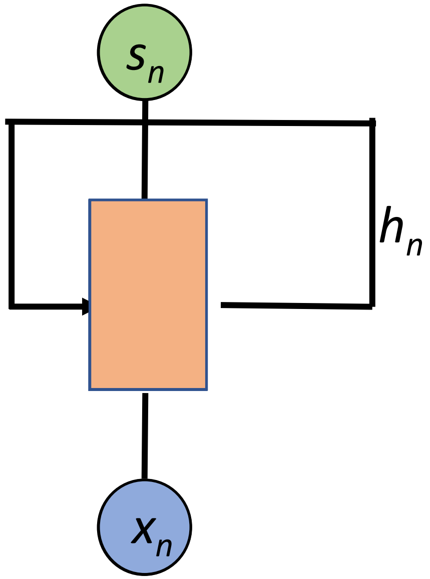
$$h_n = f_W(x_n, h_{n-1})$$

Hidden state

Function with weight W

Input

Recurrent Neural Network



A sequence of vectors is processed by applying a recurrence formula at each time step.

$$h_n = f_W(x_n, h_{n-1})$$

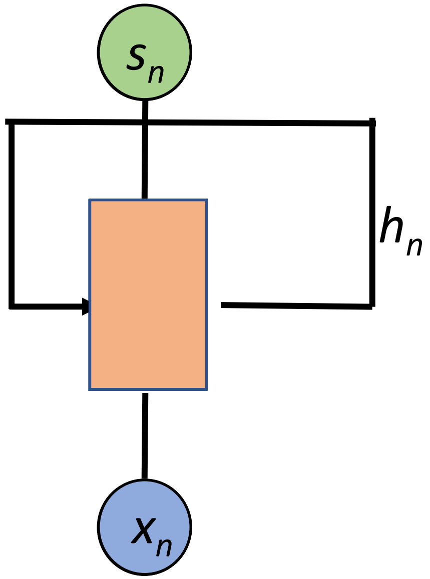
Hidden state

Function with weight W

Input

Old hidden state

Recurrent Neural Network



A sequence of vectors is processed by applying a recurrence formula at each time step.

$$h_n = f_W(x_n, h_{n-1})$$

Hidden state

Function with weight W

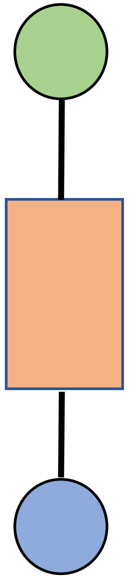
Input

Old hidden state

- Same function is used
- Ensure parameter sharing
- Handles the temporal dependency between sequence

Recurrent Neural Network Architectures

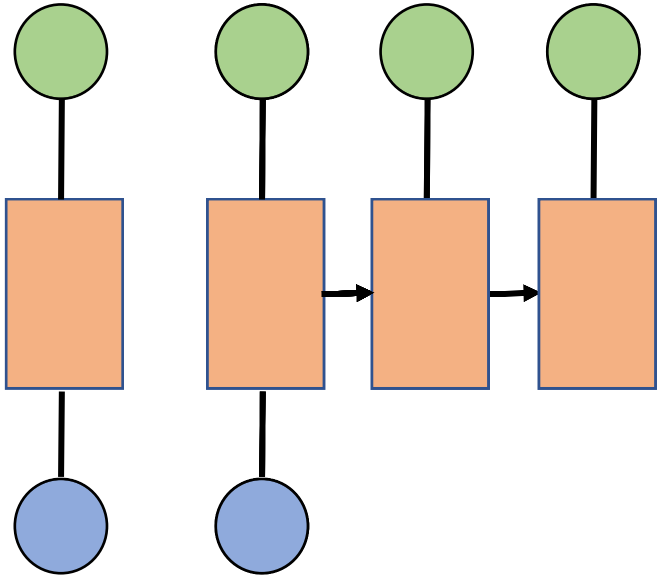
one to one



Vanilla NN

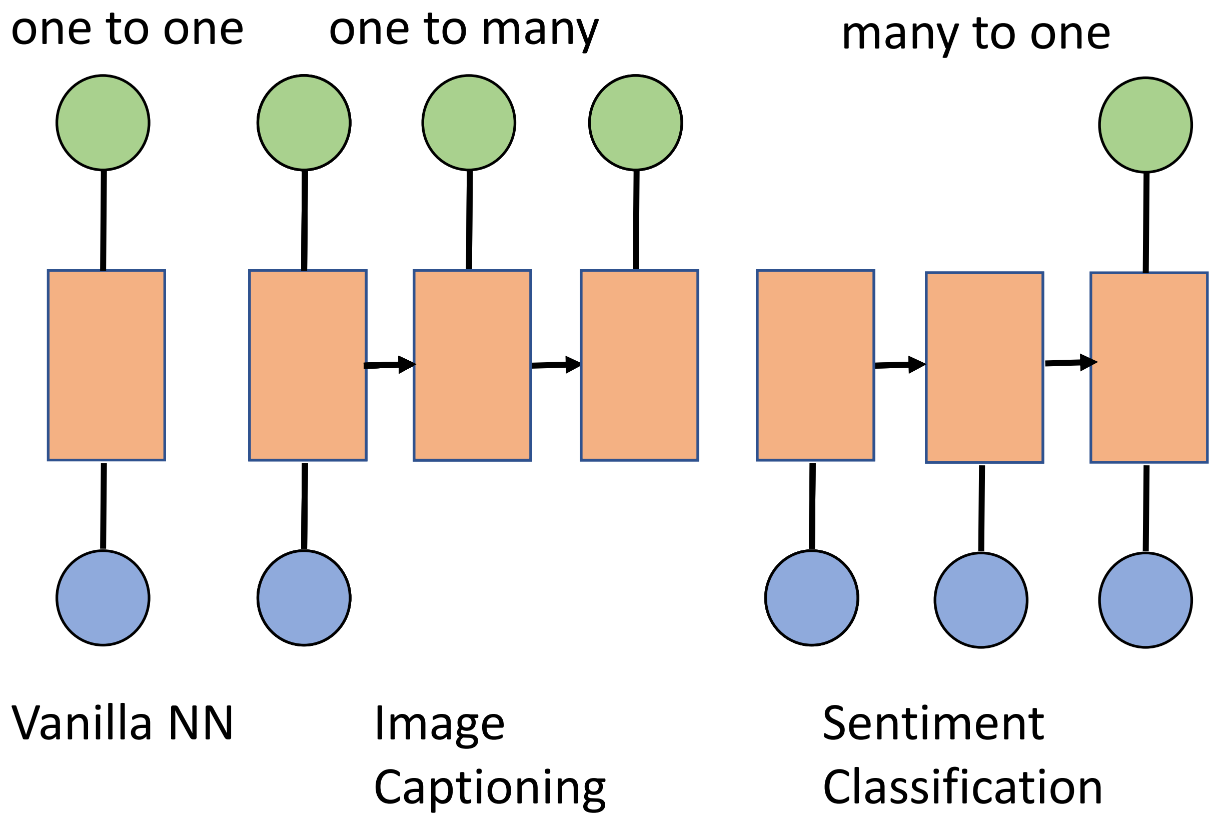
Recurrent Neural Network Architectures

one to one one to many



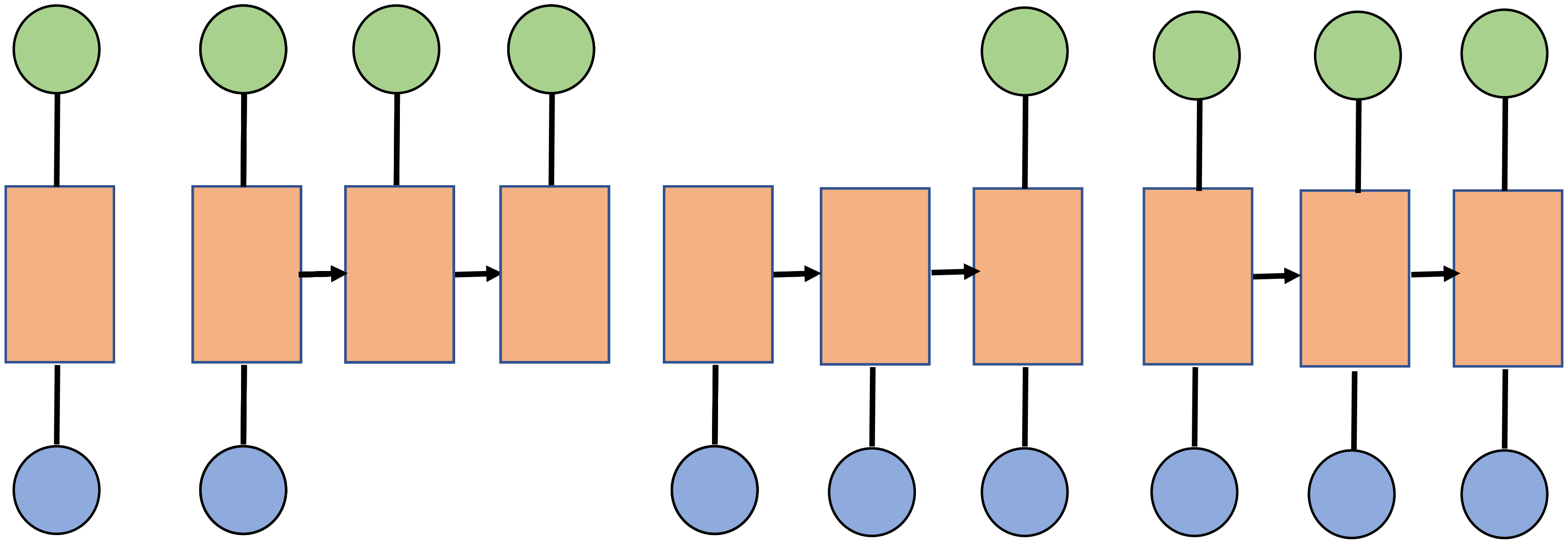
Vanilla NN Image Captioning

Recurrent Neural Network Architectures



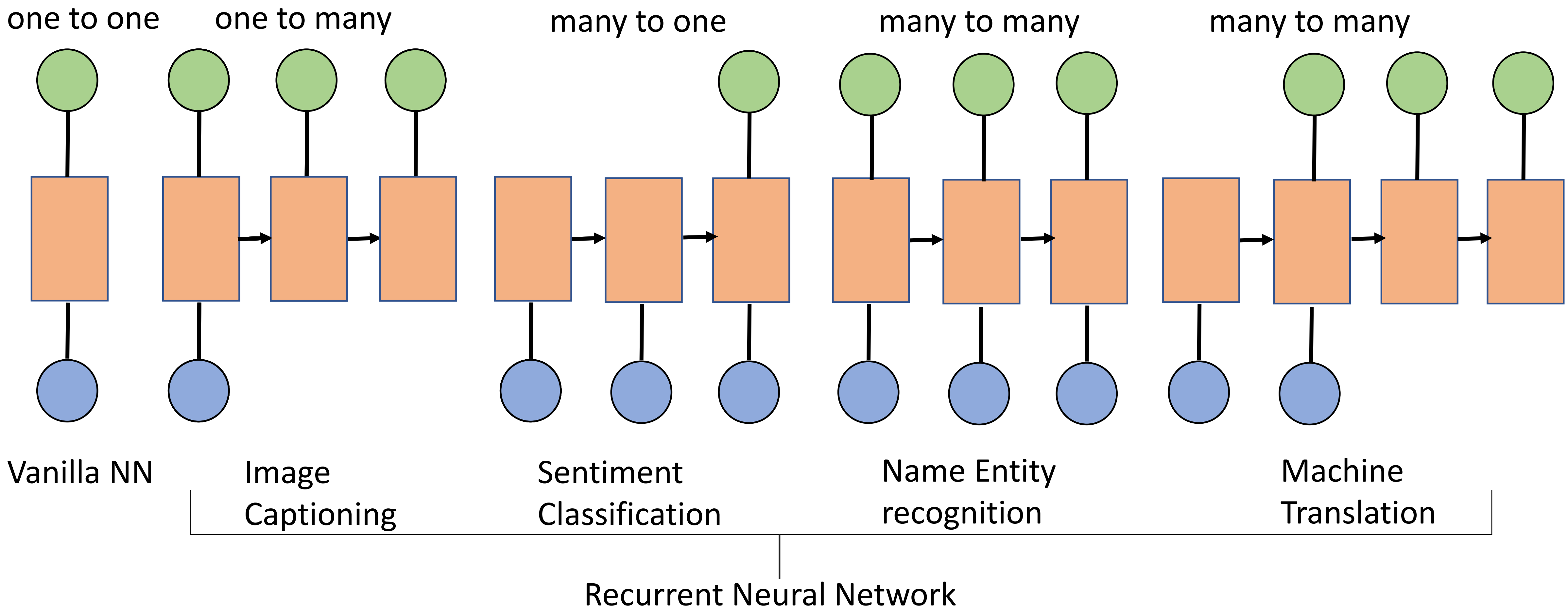
Recurrent Neural Network Architectures

one to one | one to many | many to one | many to many

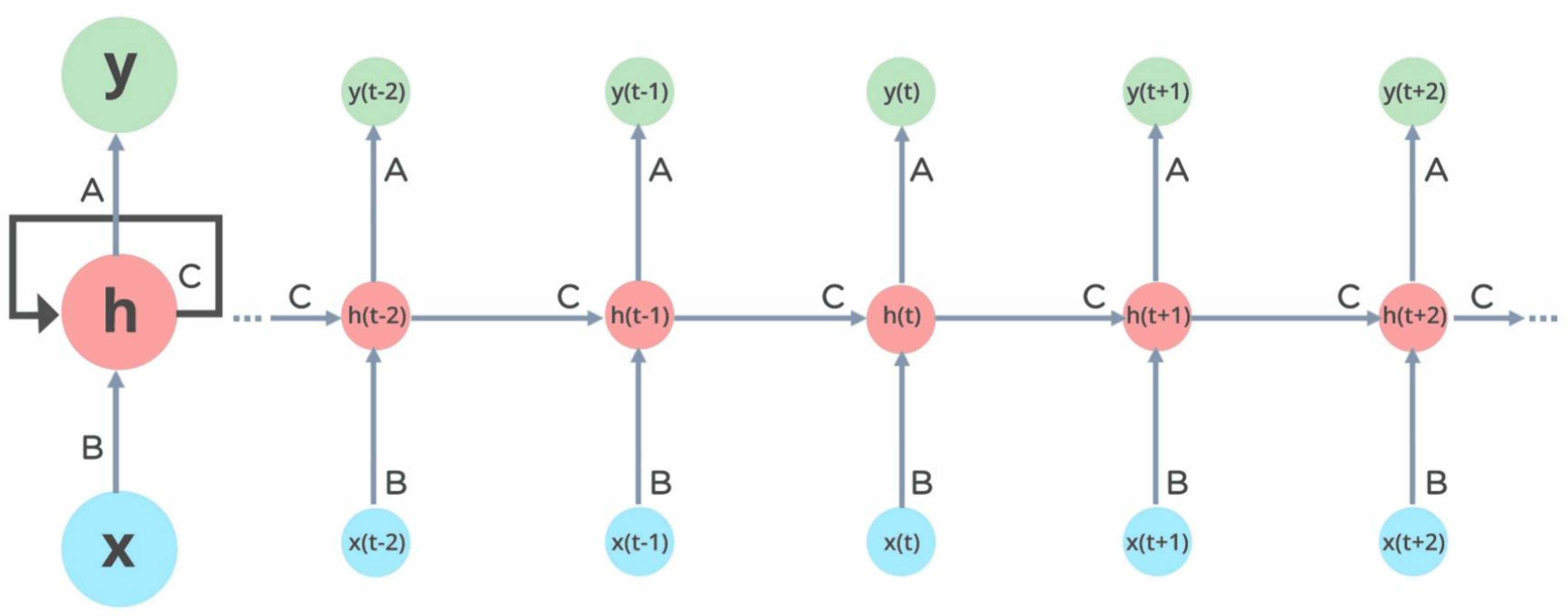


Vanilla NN | Image Captioning | Sentiment Classification | Name Entity recognition

Recurrent Neural Network Architectures

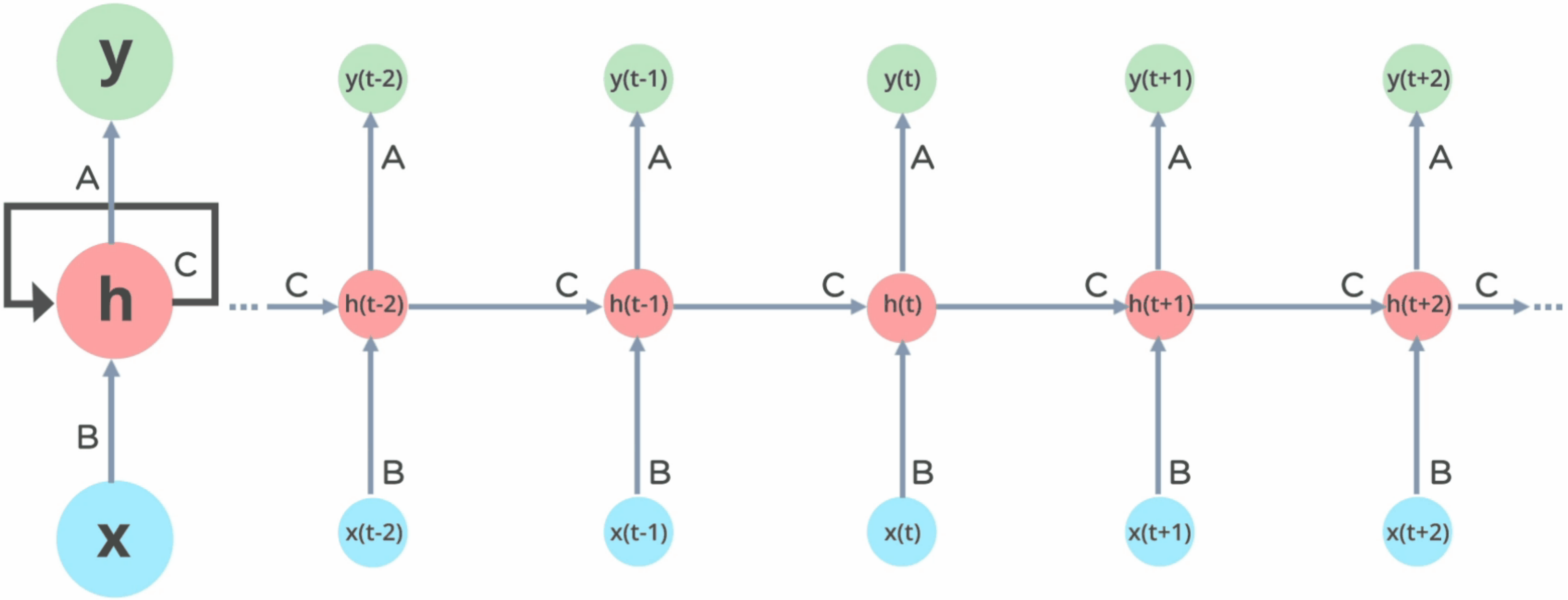


RNN: Forward Propagation



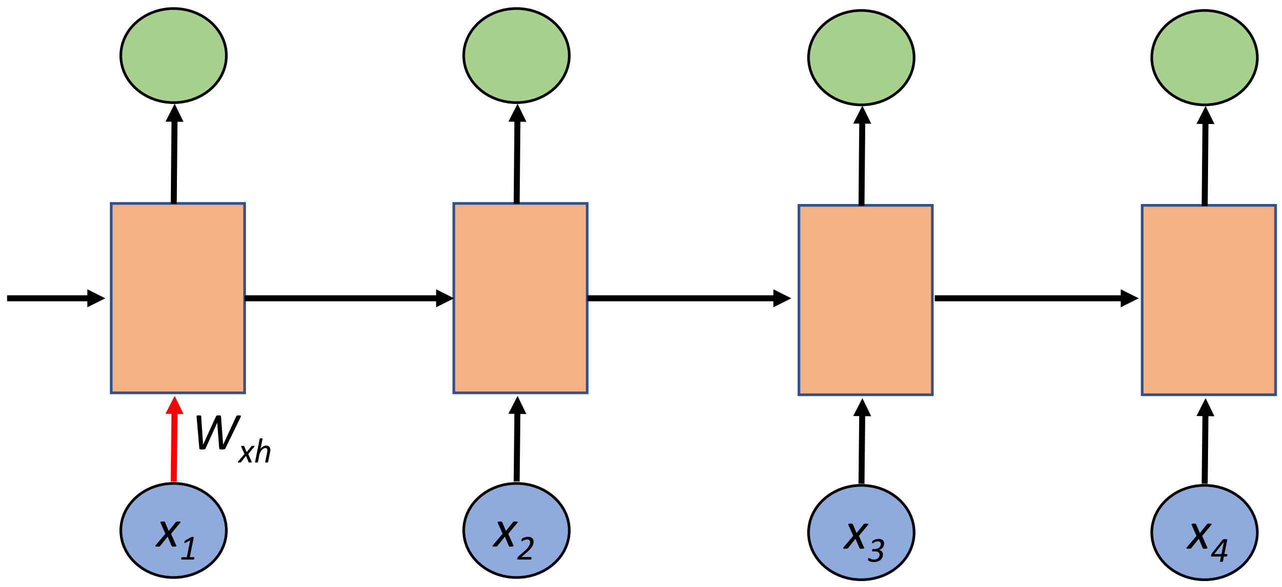
Source: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

RNN: Forward Propagation



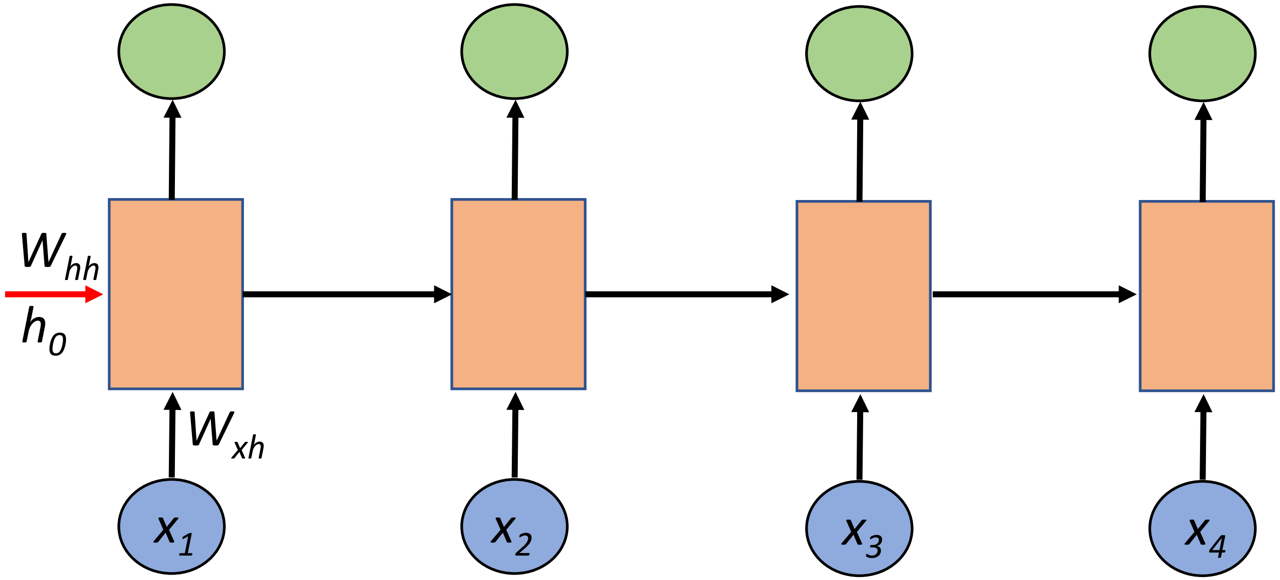
Source: https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn

RNN: Forward Propagation



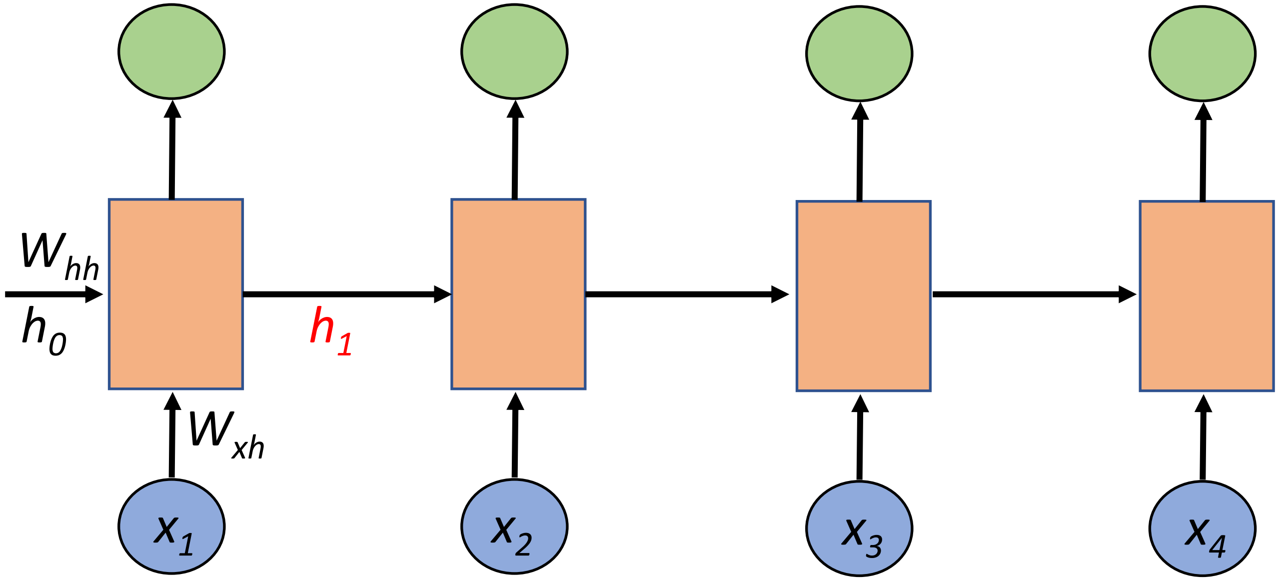
$$W_{xh} \cdot x_1$$

RNN: Forward Propagation



$$W_{xh}.\,x_1 + W_{hh}.\,h_0$$

RNN: Forward Propagation

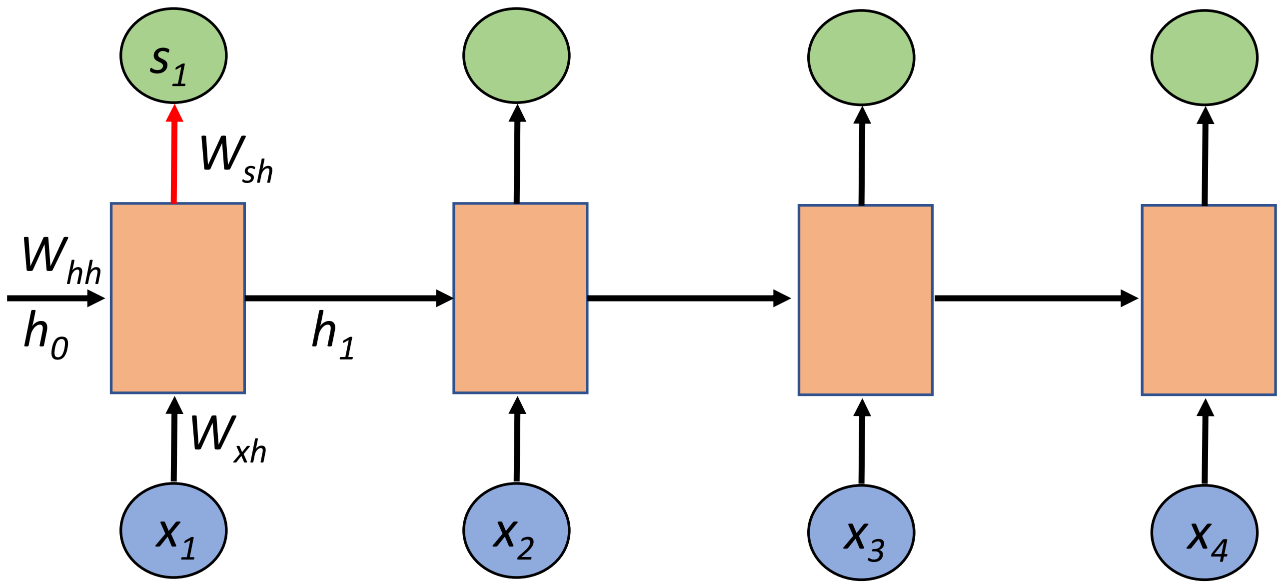


$$h_1 = g(W_{xh}.x_1 + W_{hh}.h_0 + b)$$

Activation function

Bias

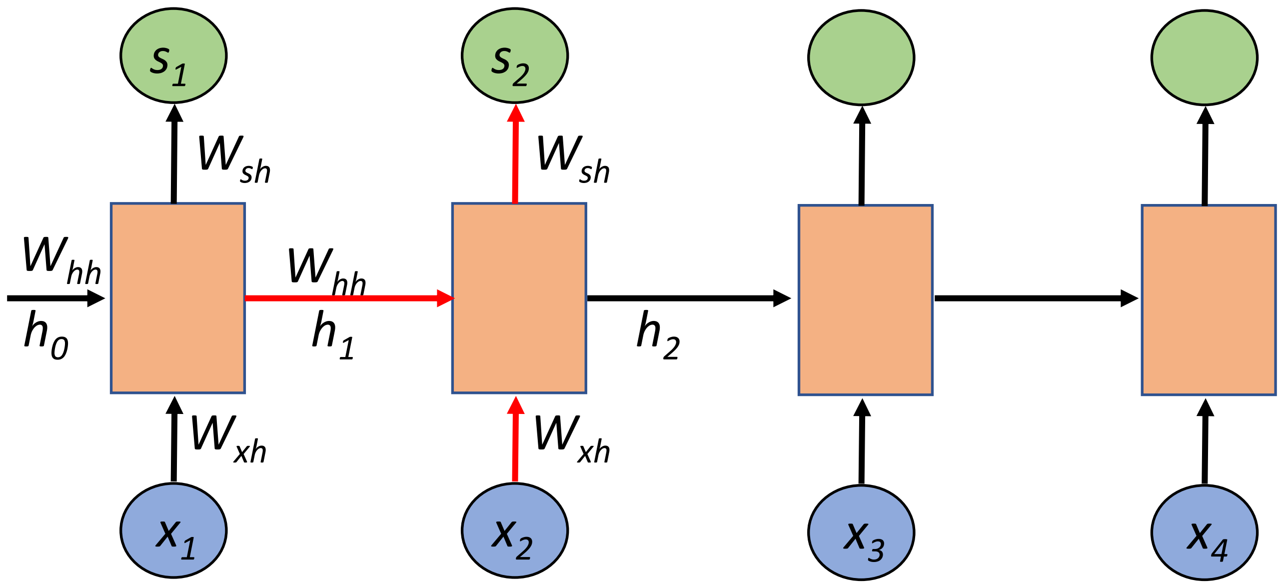
RNN: Forward Propagation



$$h_1 = g(W_{xh}.x_1 + W_{hh}.h_0 + b)$$
$$s_1 = \emptyset(W_{sh}.h_1 + c)$$

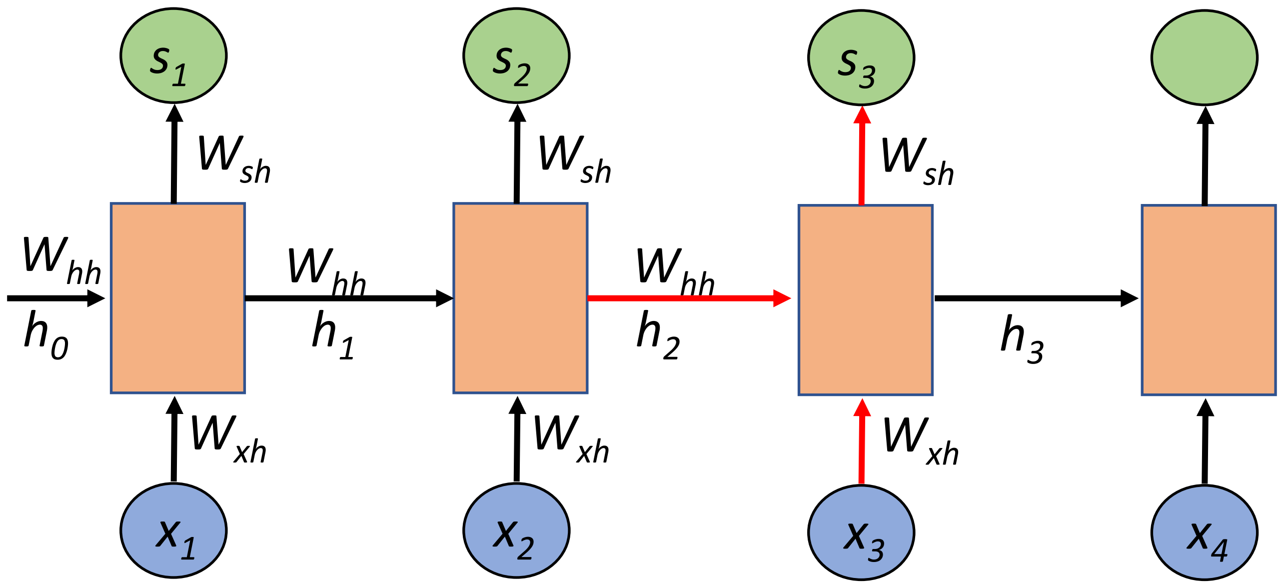
x_1	Input
h_0	Previous hidden state
h_1	Current hidden state
W_{xh}, W_{hh}, W_{sh}	Shared parameters
$g, \emptyset$	Activation functions
b,c	Biases
s_1	Output state

RNN: Forward Propagation



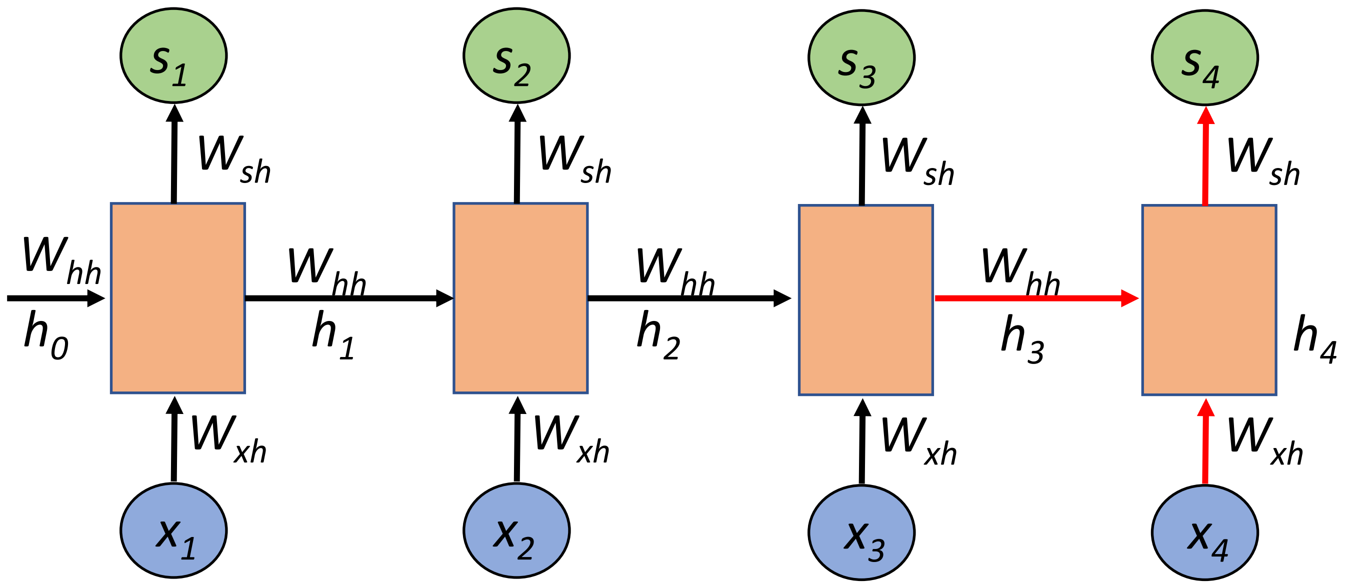
$$h_1 = g(W_{xh}.x_1 + W_{hh}.h_0 + b)$$
$$s_1 = \emptyset(W_{sh}.h_1 + c)$$
$$h_2 = g(W_{xh}.x_2 + W_{hh}.h_1 + b)$$
$$s_2 = \emptyset(W_{sh}.h_2 + c)$$

RNN: Forward Propagation



$$h_1 = g(W_{xh}.x_1 + W_{hh}.h_0 + b)$$
$$s_1 = \emptyset(W_{sh}.h_1 + c)$$
$$h_2 = g(W_{xh}.x_2 + W_{hh}.h_1 + b)$$
$$s_2 = \emptyset(W_{sh}.h_2 + c)$$
$$h_3 = g(W_{xh}.x_3 + W_{hh}.h_2 + b)$$
$$s_3 = \emptyset(W_{sh}.h_3 + c)$$

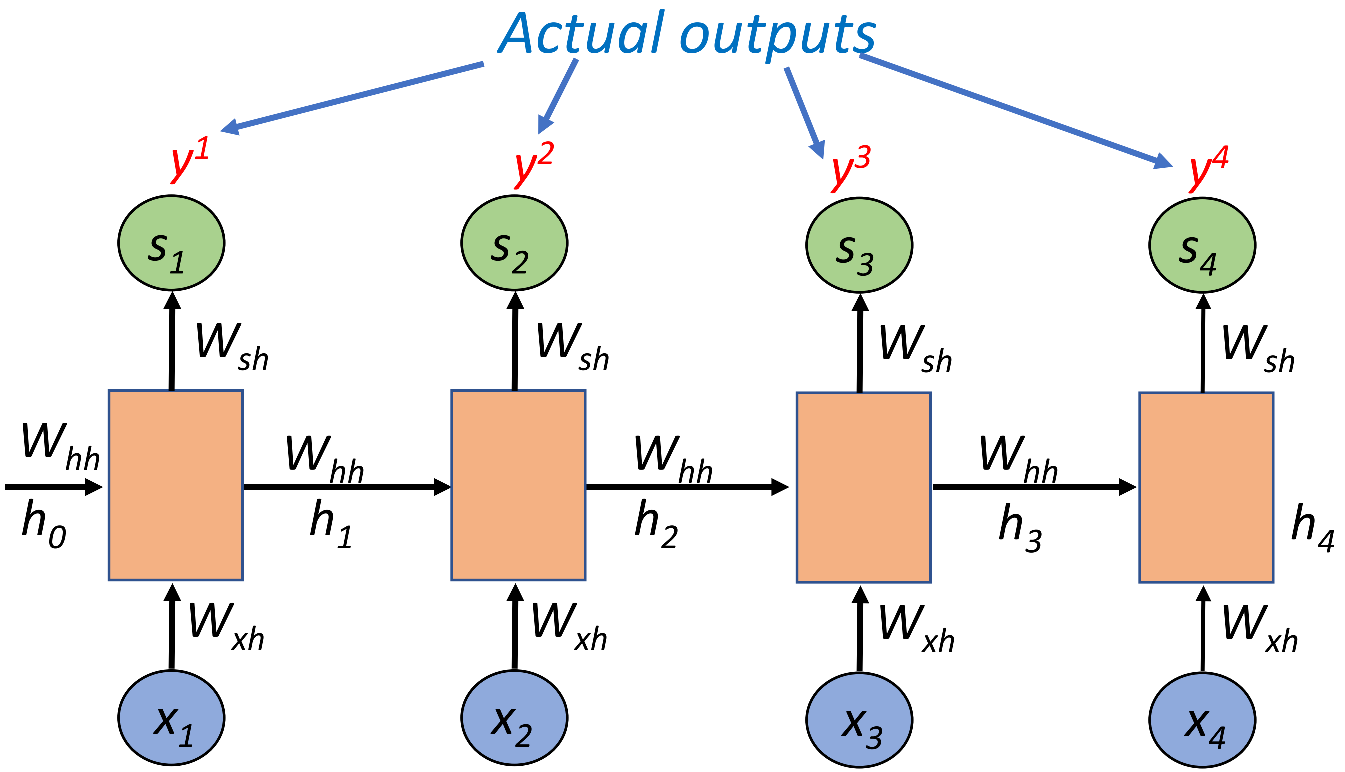
RNN: Forward Propagation



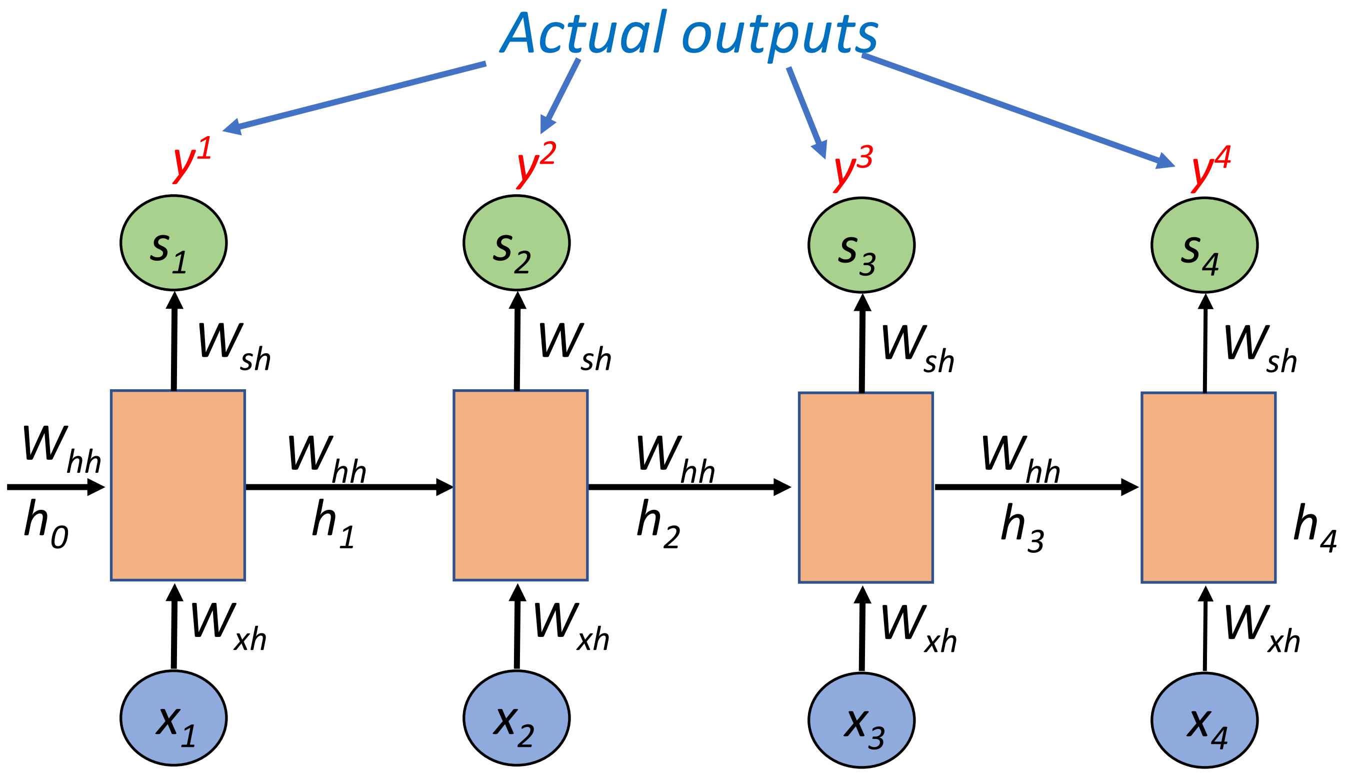
$$h_1 = g(W_{xh}.x_1 + W_{hh}.h_0 + b)$$
$$s_1 = \emptyset(W_{sh}.h_1 + c)$$
$$h_2 = g(W_{xh}.x_2 + W_{hh}.h_1 + b)$$
$$s_2 = \emptyset(W_{sh}.h_2 + c)$$
$$h_3 = g(W_{xh}.x_3 + W_{hh}.h_2 + b)$$
$$s_3 = \emptyset(W_{sh}.h_3 + c)$$
$$h_4 = g(W_{xh}.x_4 + W_{hh}.h_3 + b)$$
$$s_4 = \emptyset(W_{sh}.h_4 + c)$$

Weight matrix W_{xh} , W_{hh} and W_{sh} remains same throughout the forward propagation, thus ensuring parameter sharing

RNN: Back Propagation Through Time (BPTT)



RNN: Back Propagation Through Time (BPTT)



Loss calculation:

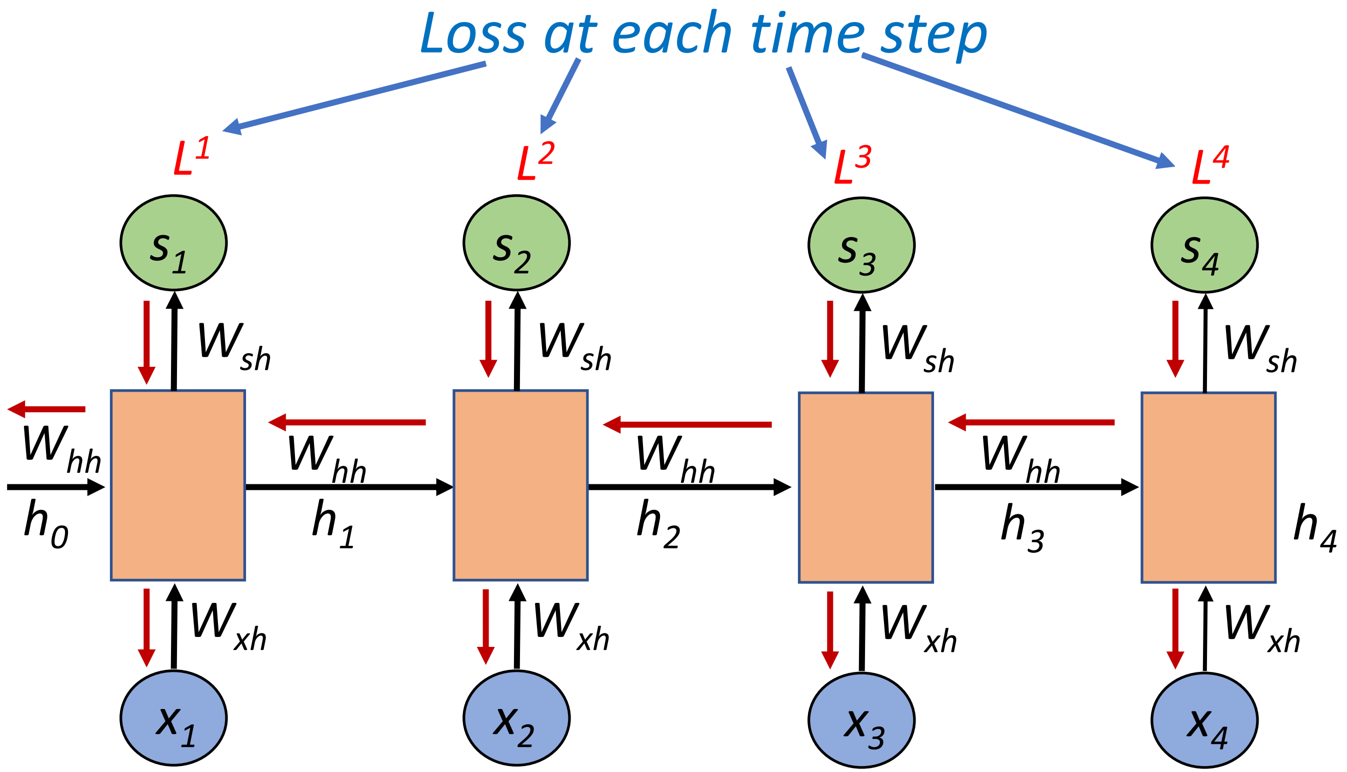
Loss $L = \mathcal{L}(y_1, s_1) + \mathcal{L}(y_2, s_2) + \mathcal{L}(y_3, s_3) + \mathcal{L}(y_4, s_4)$

$\mathcal{L}$ = *Loss function*

In general:

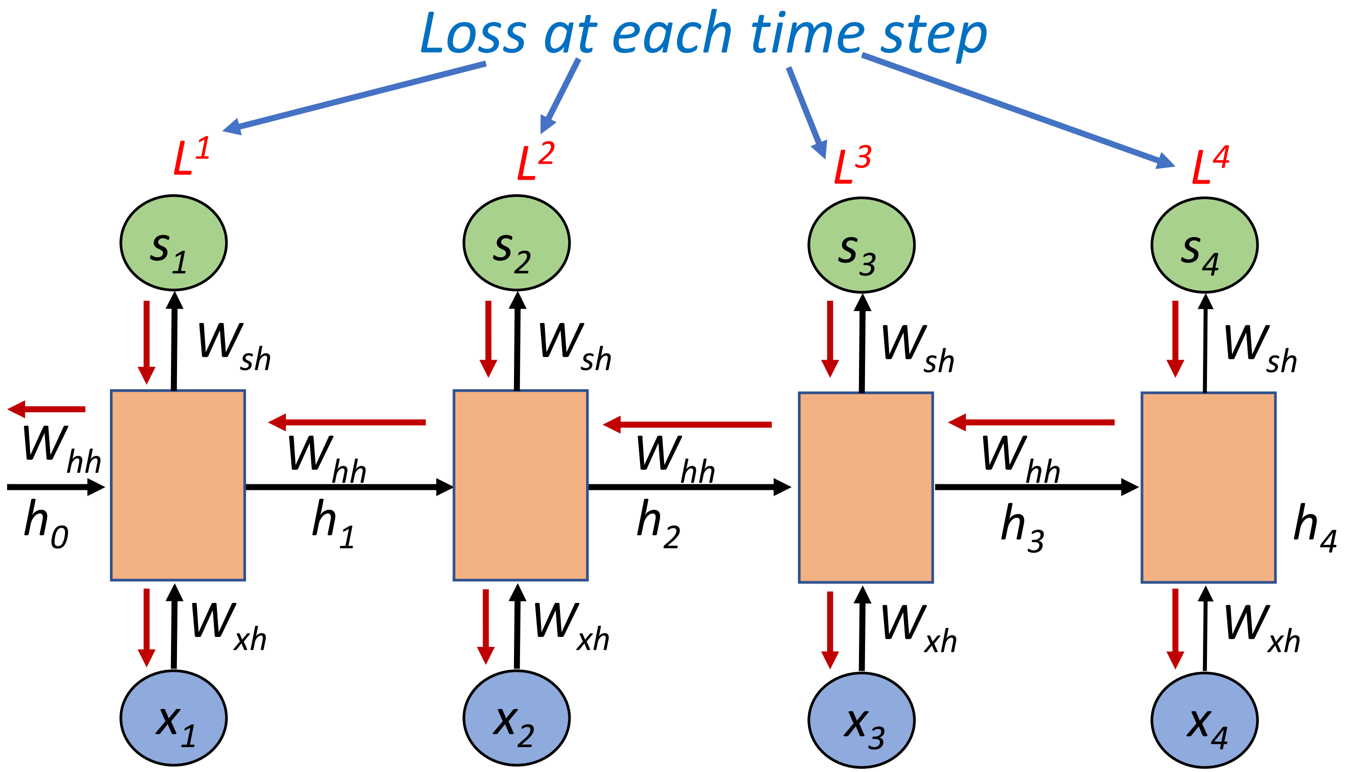
$L = \sum_{i=1}^{n} \mathcal{L}(y_i, s_i)$

RNN: Back Propagation Through Time (BPTT)



Gradient calculation wrt W_{sh} :

RNN: Back Propagation Through Time (BPTT)

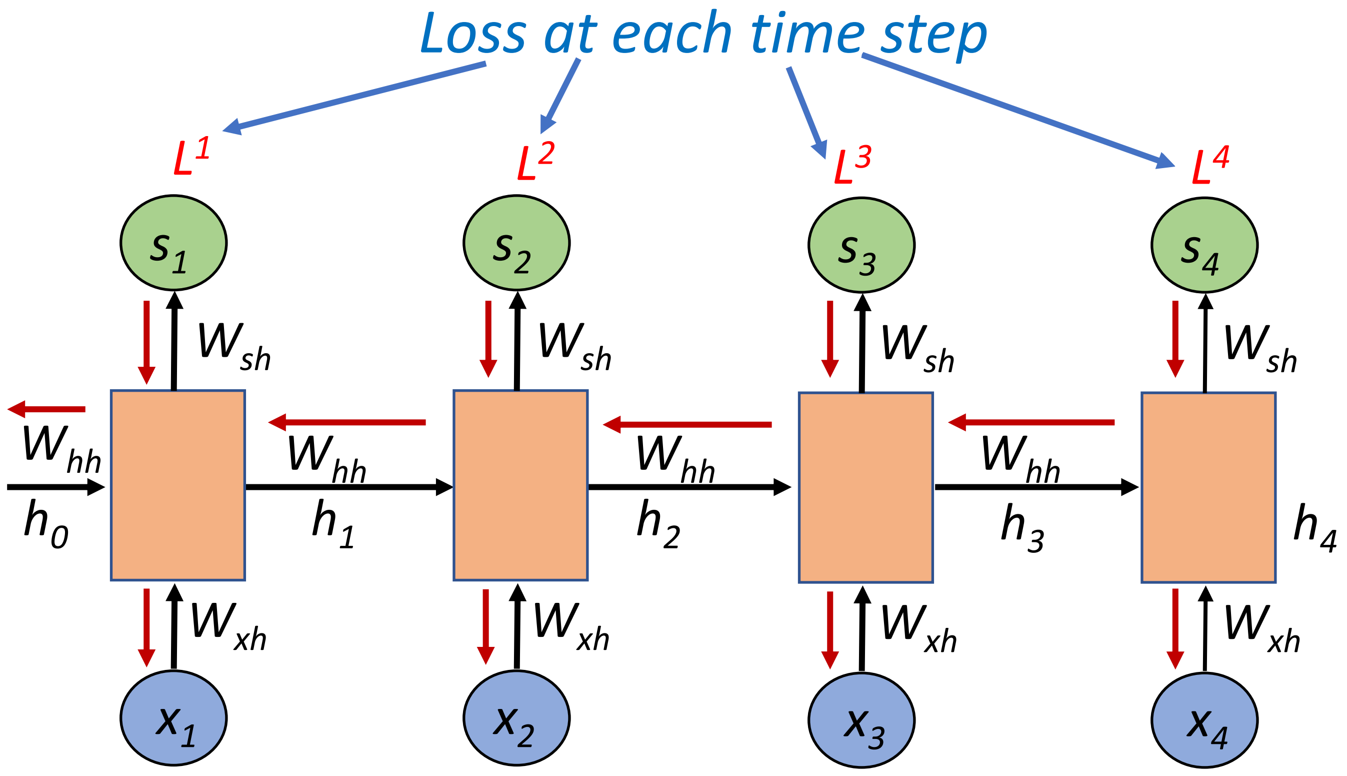


Gradient calculation wrt W_{sh} :

Assumptions:

$s_4 = W_{sh} . h_4$ *and* $\mathcal{L}$ = *least square function* $= \frac{1}{2} (y_4 - s_4)^2$

RNN: Back Propagation Through Time (BPTT)



Gradient calculation wrt W_{sh} :

$$\frac{\partial L^4}{\partial W_{sh}} = \frac{\partial L^4}{\partial s_4} \cdot \frac{\partial s_4}{\partial W_{sh}}$$

$$= -(y_4 - s_4).h_4$$

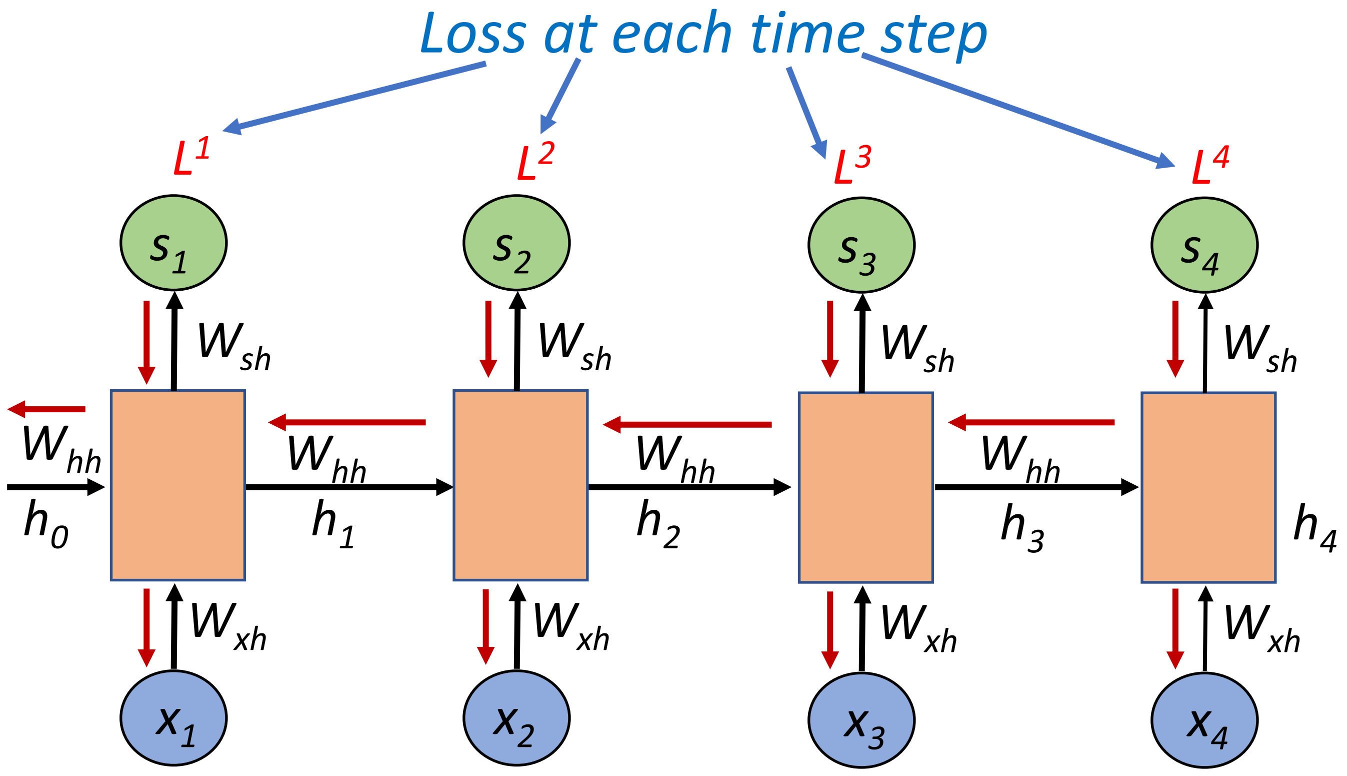
Weight updation wrt W_{sh} :

$$W_{sh} = W_{sh} - \sum_{i=1}^{n}(y_i - s_i).h_i$$

Assumptions:

$s_4 = W_{sh}.h_4$ *and* $\mathcal{L}$ = *least square function* $=\frac{1}{2}(y_4 - s_4)^2$

RNN: Back Propagation Through Time (BPTT)



Gradient calculation wrt W_{hh} :

$$\frac{\partial L^4}{\partial W_{hh}} = \frac{\partial L^4}{\partial s_4} \cdot \frac{\partial s_4}{\partial h_4} \cdot \frac{\partial h_4}{\partial W_{hh}}$$

$$= -(y_4 - s_4).W_{sh}.\frac{\partial h_4}{\partial W_{hh}}$$

Now, $h_4 = g(W_{xh}.x_4 + W_{hh}.h_3)$

Simply, $h_4 = g(z_4)$

Then, $\frac{\partial h_4}{\partial W_{hh}} = \frac{\partial g}{\partial z_4} \cdot \frac{\partial z_4}{\partial W_{hh}}$

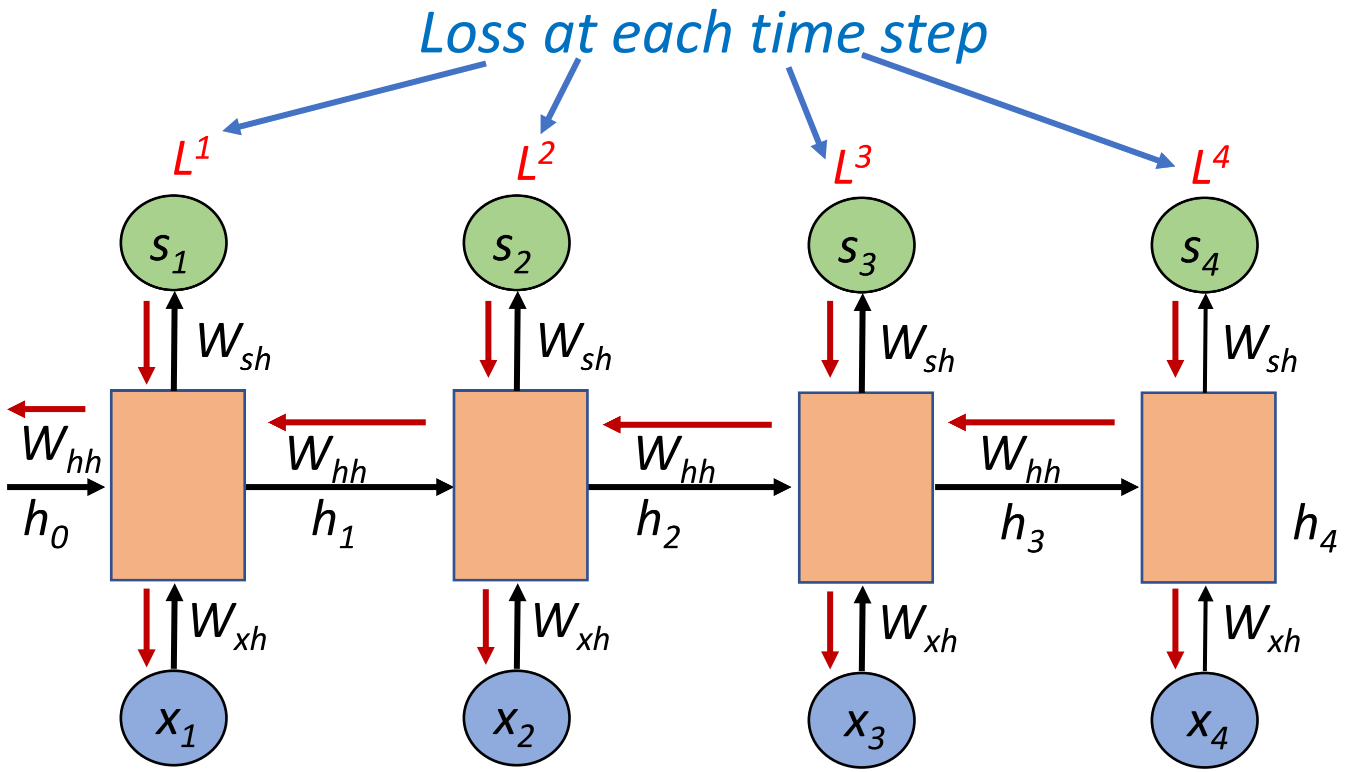
$$= g'.[h_3 + \frac{\partial h_3}{\partial W_{hh}}]$$

Assumptions:

$s_4 = W_{sh}.h_4$ *and* $\mathcal{L}$ = *least square function* $= \frac{1}{2}(y_4 - s_4)^2$

$$= g'.[h_3 + h_2 + \frac{\partial h_2}{\partial W_{hh}}]$$

RNN: Back Propagation Through Time (BPTT)



Gradient calculation wrt W_{hh} :

$$\frac{\partial L^4}{\partial W_{hh}} = -(y_4 - s_4).W_{sh}.g'.[h_3 + h_2 + \cdots + \frac{\partial h_0}{\partial W_{hh}}]$$

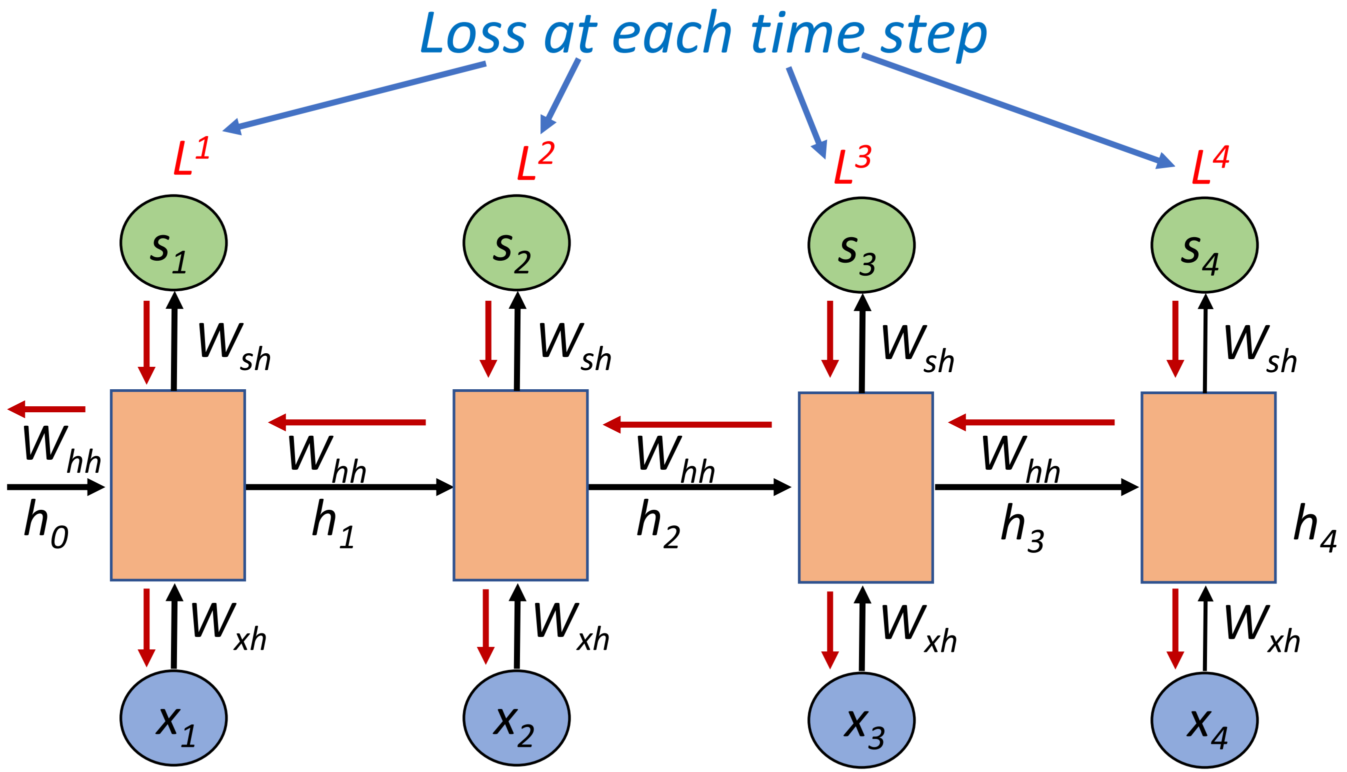
Weight updation wrt W_{hh} :

$$W_{hh} = W_{hh} - \sum_{i=1}^{n} \frac{\partial L^i}{\partial W_{hh}}$$

Assumptions:

$s_4 = W_{sh}.h_4$ *and* $\mathcal{L}$ = *least square function* $= \frac{1}{2}(y_4 - s_4)^2$

RNN: Back Propagation Through Time (BPTT)



Gradient calculation wrt W_{xh} :

$$\frac{\partial L^4}{\partial W_{xh}} = \frac{\partial L^4}{\partial s_4} \cdot \frac{\partial s_4}{\partial h_4} \cdot \frac{\partial h_4}{\partial W_{xh}}$$

$$= -(y_4 - s_4).W_{sh} \cdot \frac{\partial h_4}{\partial W_{xh}}$$

Now, $h_4 = g(W_{xh}.x_4 + W_{hh}.h_3)$

Simply, $h_4 = g(z_4)$

Then, $\frac{\partial h_4}{\partial W_{xh}} = \frac{\partial g}{\partial z_4} \cdot \frac{\partial z_4}{\partial W_{xh}}$

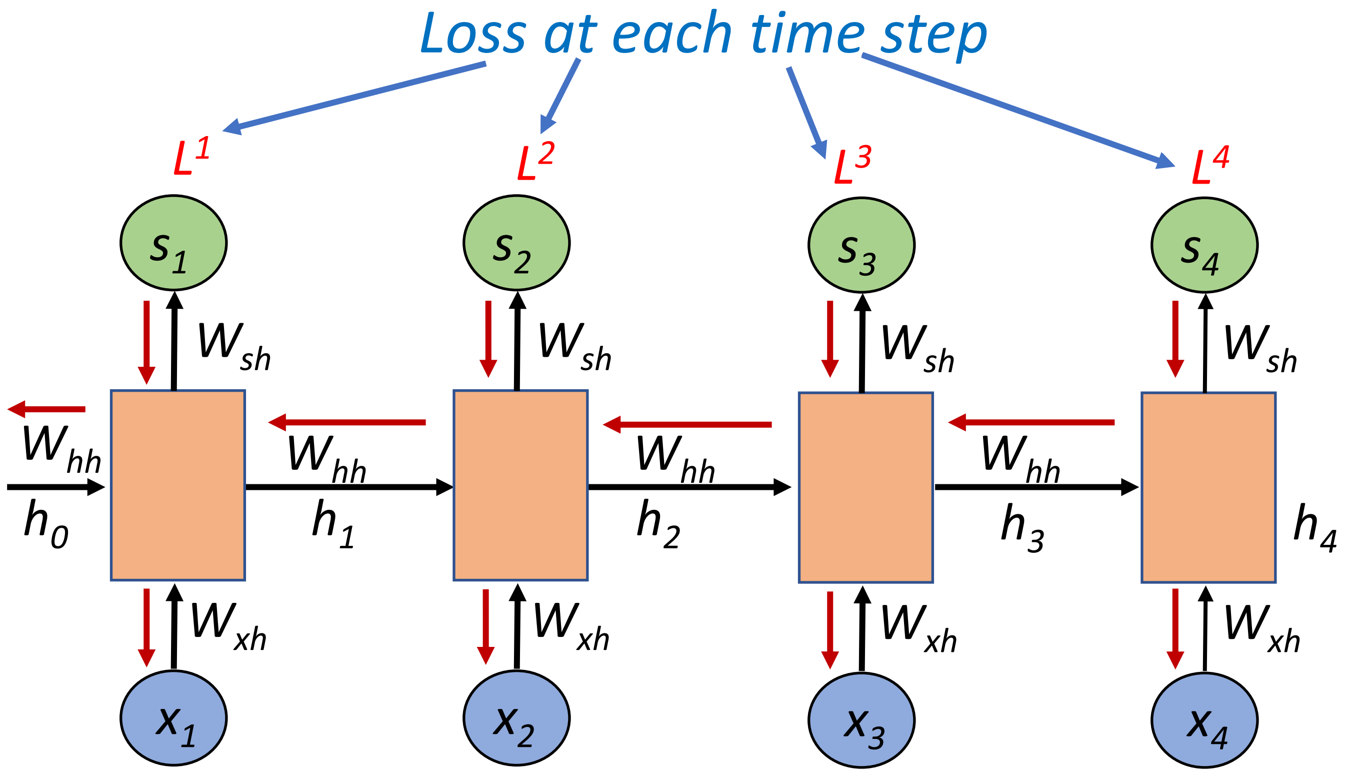
$$= g'.[x_4 + \frac{\partial W_{hh}.h_3}{\partial W_{xh}}]$$

$$= g'.[x_4 + W_{hh}.g'(z_2).\frac{\partial z_2}{\partial W_{xh}}]$$

Assumptions:

$s_4 = W_{sh}.h_4$ *and* $\mathcal{L}$ = *least square function* $= \frac{1}{2}(y_4 - s_4)^2$

RNN: Back Propagation Through Time (BPTT)



Gradient calculation wrt W_{xh} :

$$\frac{\partial L^4}{\partial W_{xh}}$$

$$= -(y_4 - s_4)W_{sh}.\, g'.\,[x_4 + W_{hh}.\, g'(z_2).\frac{\partial z_2}{\partial W_{xh}} \ldots .]$$

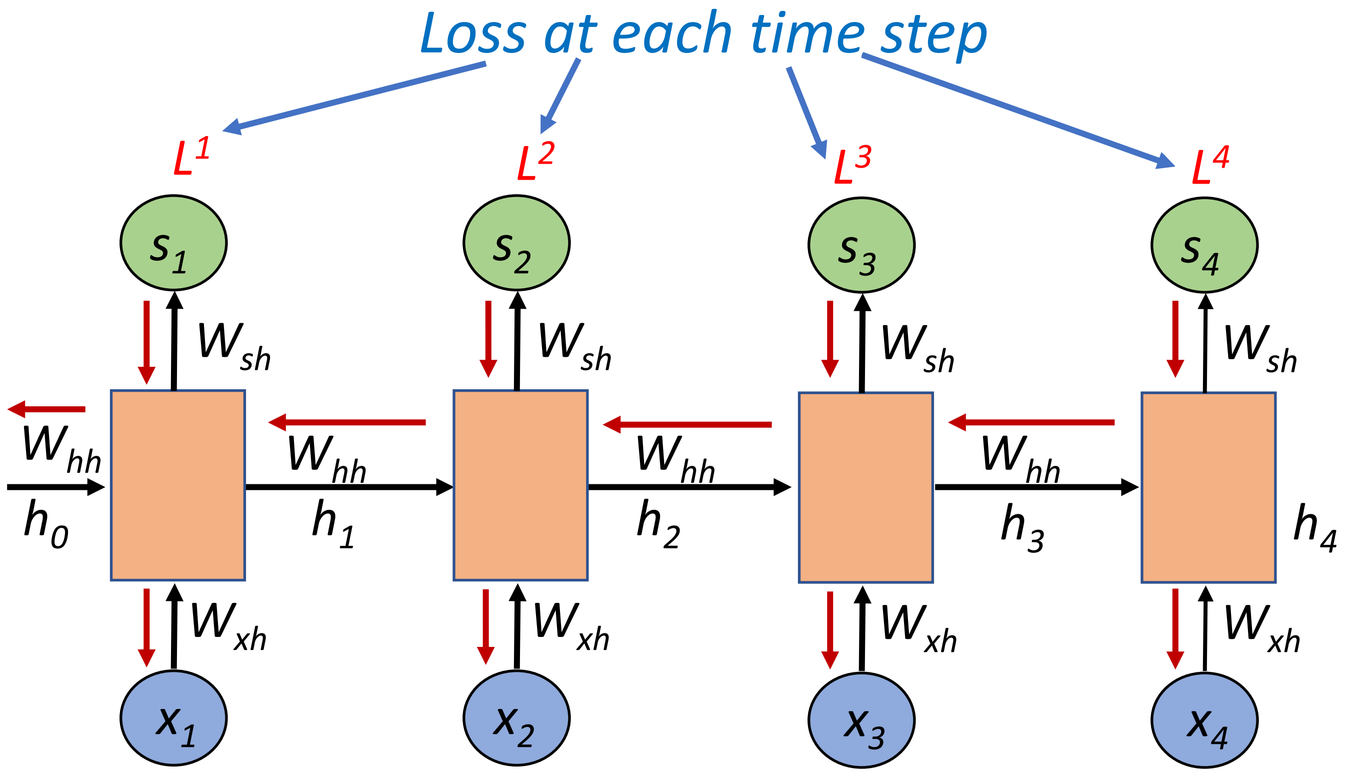
Weight updation wrt W_{xh} :

$$W_{xh} = W_{xh} - \sum_{i=1}^{n} \frac{\partial L^i}{\partial W_{xh}}$$

Assumptions:

$s_4 = W_{sh}.h_4$ *and* $\mathcal{L}$ *= least square function* $= \frac{1}{2}(y_4 - s_4)^2$

RNN: Back Propagation Through Time (BPTT)



Gradient calculation wrt W_{xh} *:*

$$\frac{\partial L^4}{\partial W_{xh}}$$

$$= -(y_4 - s_4)W_{sh}.\, g'.\, [x_4 + W_{hh}.\, g'(z_2).\frac{\partial z_2}{\partial W_{xh}} \dots .]$$

Weight updation wrt W_{xh} *:*

$$W_{xh} = W_{xh} \ - \sum_{i=1}^{n} \frac{\partial L^i}{\partial W_{xh}}$$

tf.keras.layers.SimpleRNN(rnn_units)

Assumptions:

$s_4 = W_{sh}.h_4$ *and* $\mathcal{L}$ *= least square function* $= \frac{1}{2}(y_4 - s_4)^2$

Limitations of RNN

- Gradient calculation involves many factors of weights and contribution of activation function.

Limitations of RNN

- Gradient calculation involves many factors of weights and contribution of activation function.
- This may lead to:

Exploding Gradient

Limitations of RNN

- Gradient calculation involves many factors of weights and contribution of activation function.
- This may lead to:

Exploding Gradient

Vanishing Gradient

Limitations of RNN

- Gradient calculation involves many factors of weights and contribution of activation function.
- This may lead to:

Exploding Gradient

$$\frac{\partial L}{\partial W} \gg 1 \rightarrow resulting\ in\ NAN\ values$$

Vanishing Gradient

Limitations of RNN

- Gradient calculation involves many factors of weights and contribution of activation function.
- This may lead to:

Exploding Gradient

$$\frac{\partial L}{\partial W} \gg 1 \rightarrow resulting\ in\ NAN\ values$$

Vanishing Gradient

$$\frac{\partial L}{\partial W} << 1 \rightarrow resulting\ in\ 0\ values$$

Limitations of RNN

- Gradient calculation involves many factors of weights and contribution of activation function.
- This may lead to:

Exploding Gradient

Vanishing Gradient

  - make learning unstable

Limitations of RNN

- Gradient calculation involves many factors of weights and contribution of activation function.
- This may lead to:

Exploding Gradient

- make learning unstable

Vanishing Gradient

- Short term dependencies

"the stars shine in the ?" → sky ***(RNN works good here)***

- Long term dependencies

"I grew up in Spain………………………………… I speak fluent Spanish". ***(Difficult for RNN to remember as gap increases)***

Possible Solutions

Exploding Gradient

Vanishing Gradient

- Gradient clipping

```
# inside the optimizer we are doing clipping
optimizer=tf.keras.optimizers.SGD(clipvalue=0.5)
```

Possible Solutions

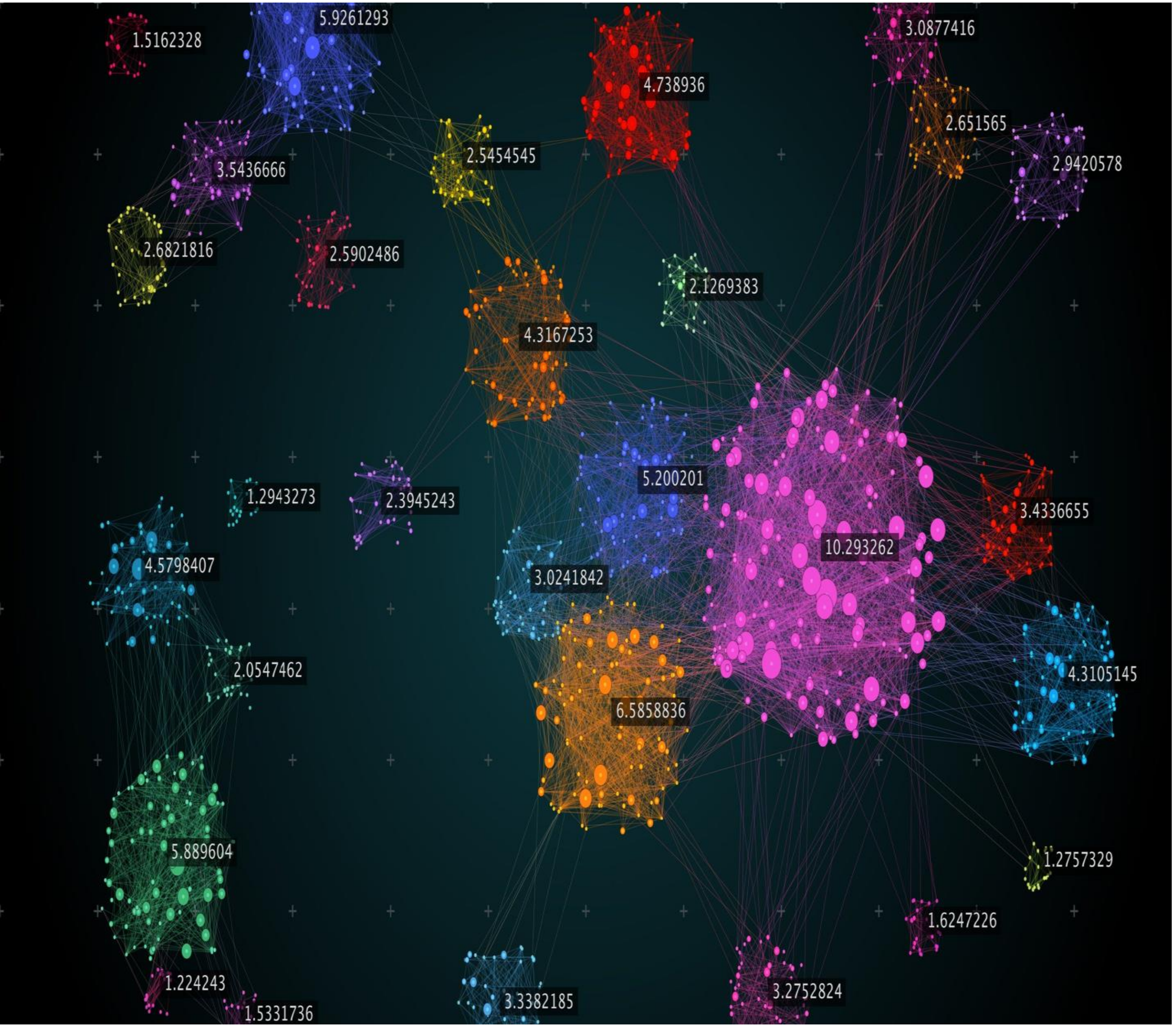
Exploding Gradient

- Gradient clipping

```
# inside the optimizer we are doing clipping
optimizer=tf.keras.optimizers.SGD(clipvalue=0.5)
```

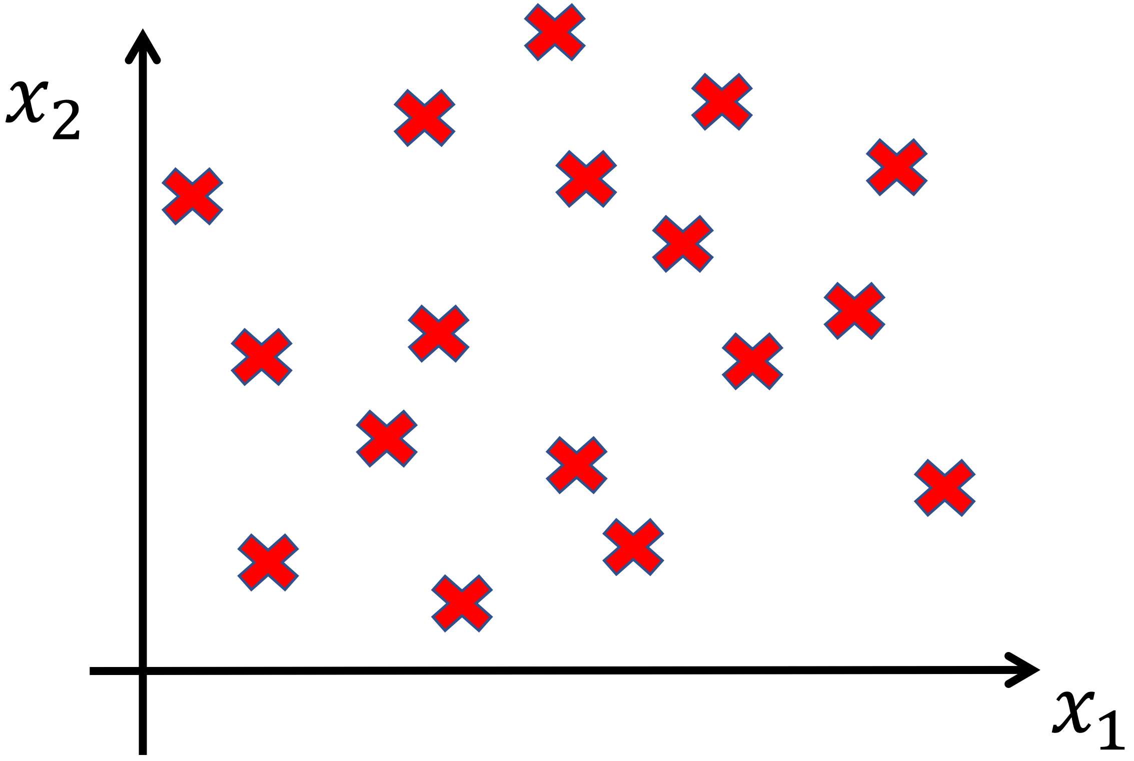
Vanishing Gradient

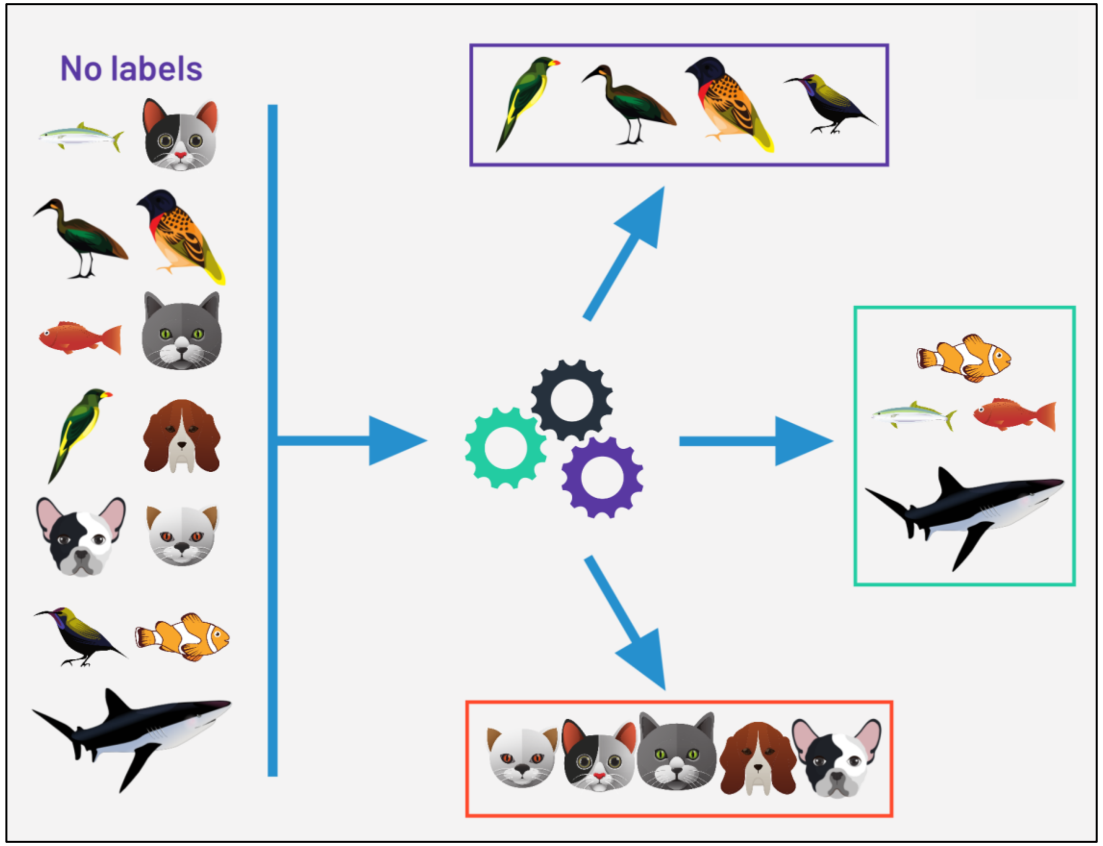
- Activation function (Relu)
- Weight initialization (identity matrix)
- Gated cells (LSTM,GRU,etc)



Introduction to Clustering

Unsupervised Learning

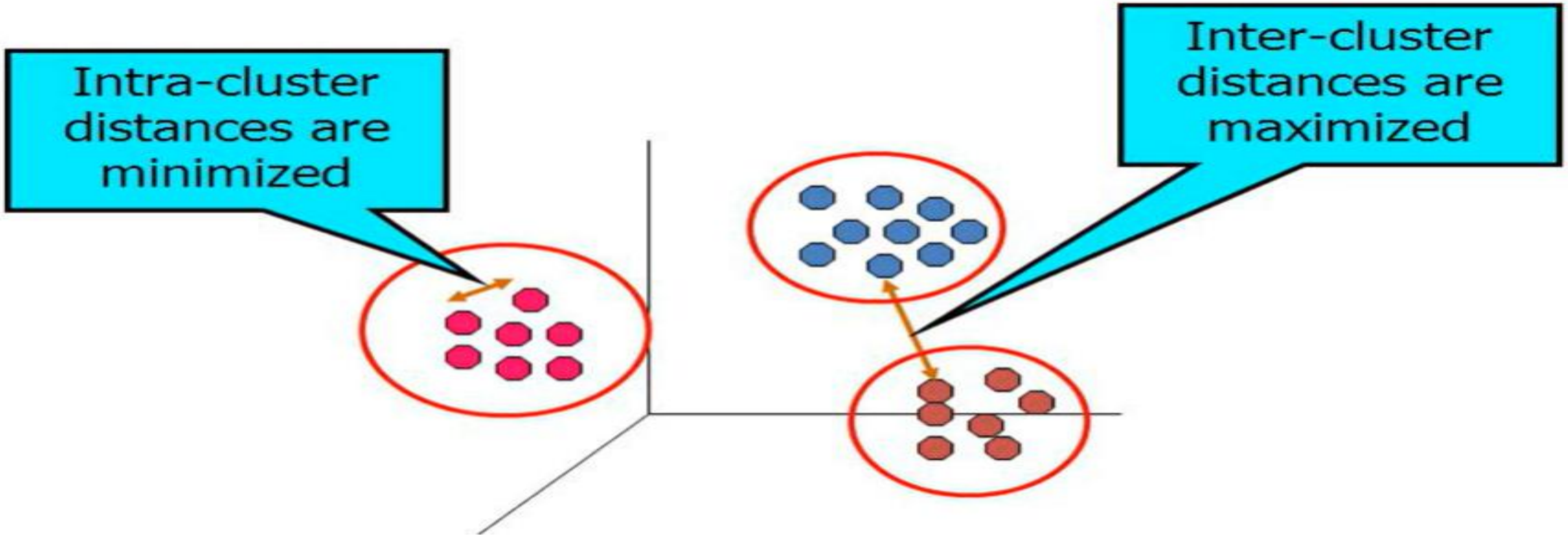




- Training set: $\{x^{(1)}, x^{(2)}, x^{(3)}, \cdots, x^{(m)}\}$

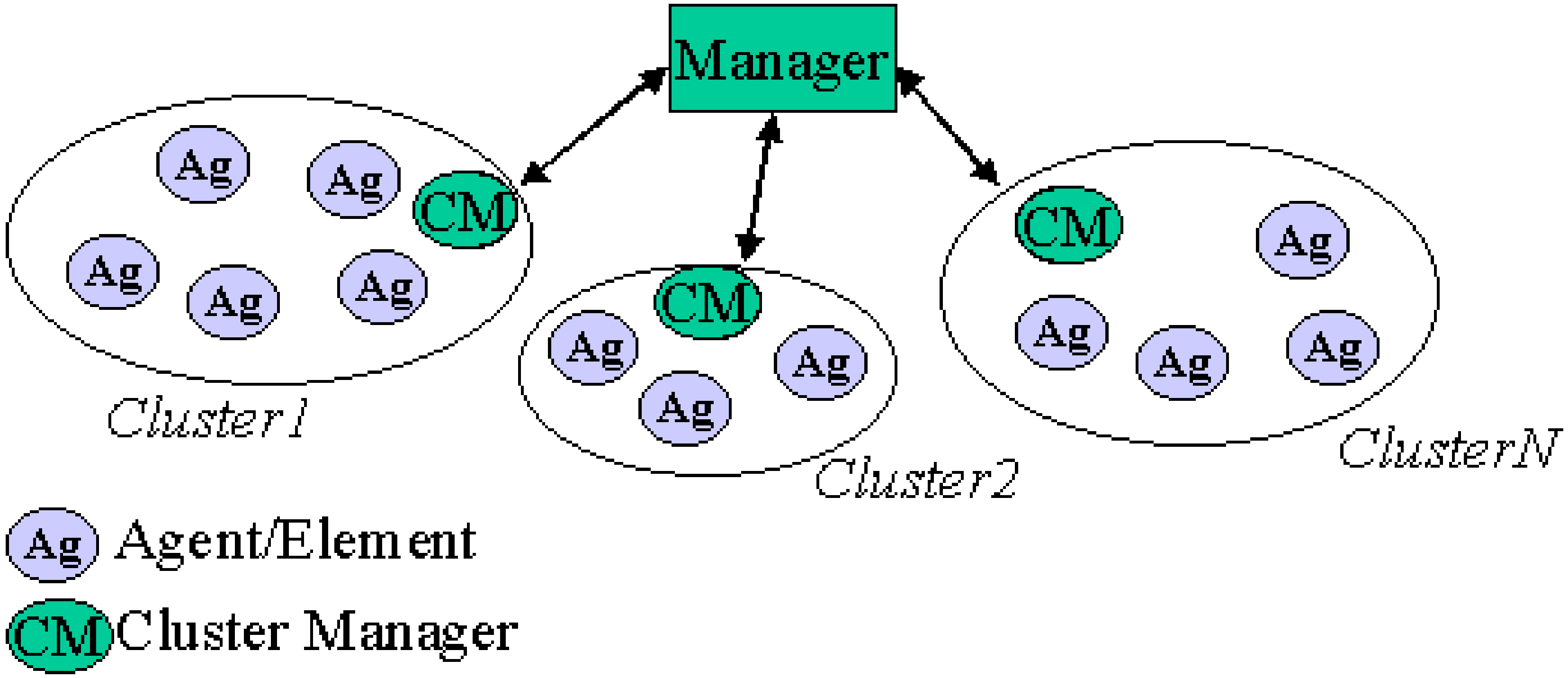
What is a Clustering?

- In general a grouping of objects such that the objects in a group (cluster) are similar (or related) to one another and different from (or unrelated to) the objects in other groups



Why Clustering ?

- ***logical/efficient/convenient groupings of elements***



Applications of clustering

Banking/ Finance/ insurance

Recommendation System

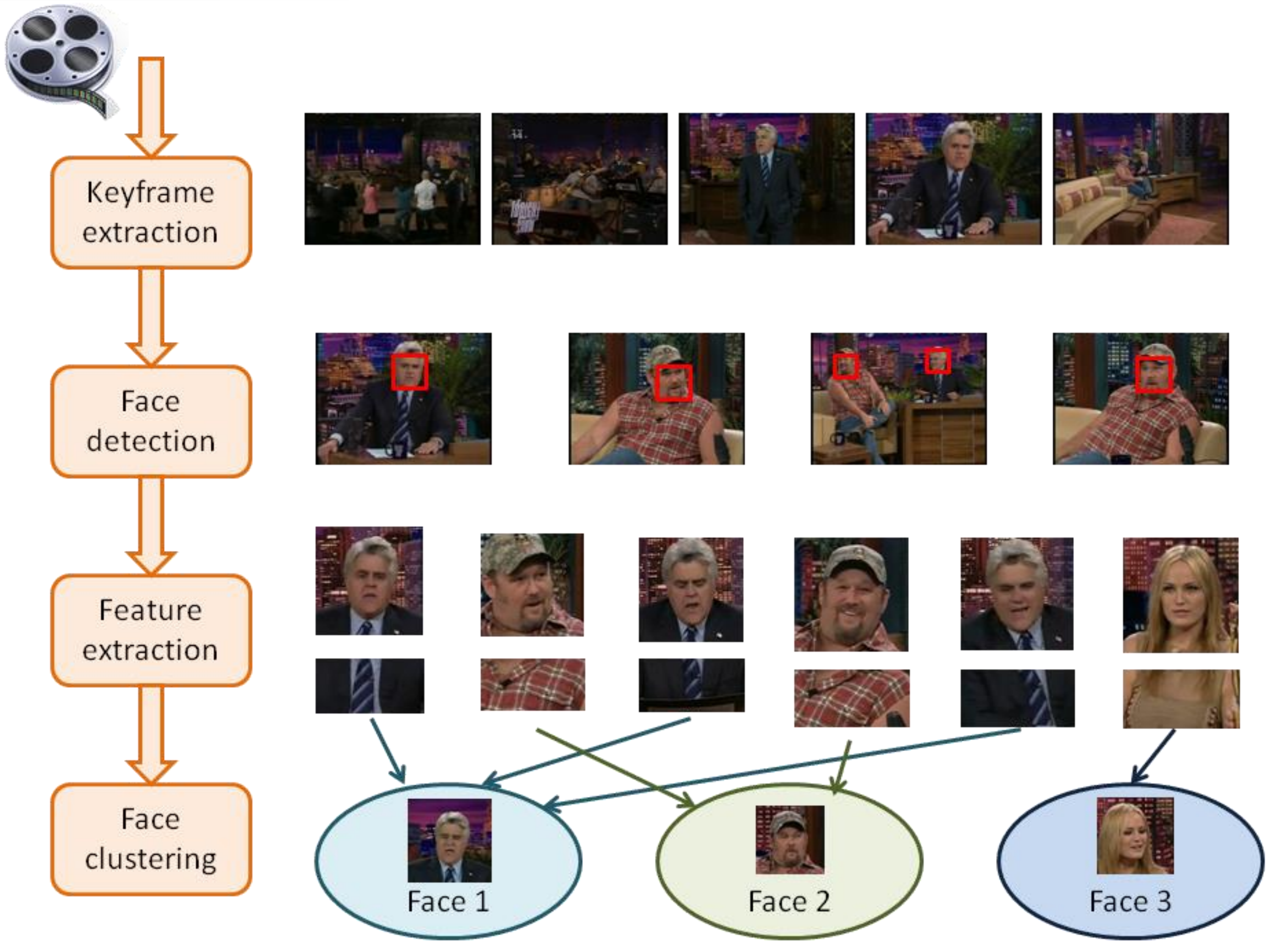
Search results

Retail Stores

Grouping of photos

Movie Recommendation

Applications of clustering (Face Clustering)



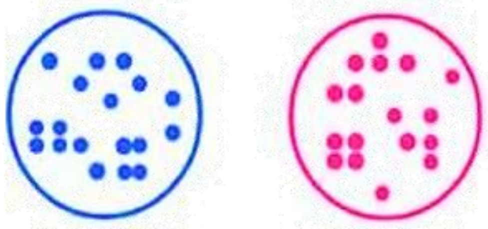
Applications of clustering (Face Clustering)

Types of CLUSTERING

- ❑ Exclusive Clustering
- ❑ Overlapping Clustering
- ❑ Hierarchical Clustering

Exclusive Clustering

- Hard Clustering
- Data Point/Item belongs exclusively to one cluster
- For Example- K-Means Clustering

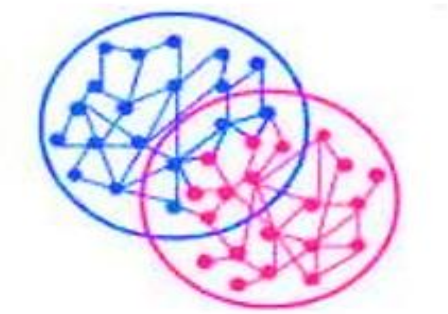


Types of CLUSTERING

- ❑ Exclusive Clustering
- ❑ Overlapping Clustering
- ❑ Hierarchical Clustering

Overlapping Clustering

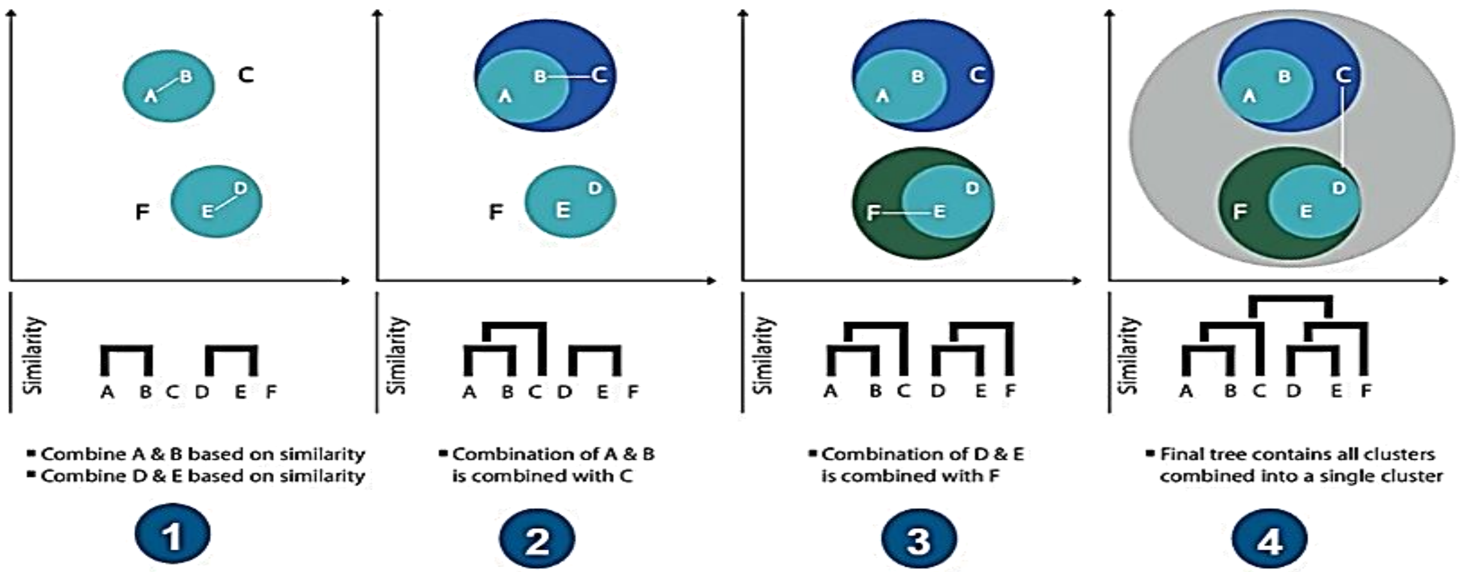
- Soft Cluster
- Data Points/Item Belong to Multiple Cluster
- For Example- Fuzzy/ C-Means Clustering



Types of CLUSTERING

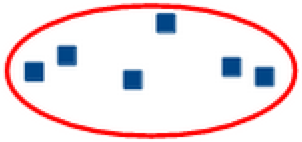
- ❑ Exclusive Clustering
- ❑ Overlapping Clustering
- ❑ Hierarchical Clustering

Hierarchal Clustering

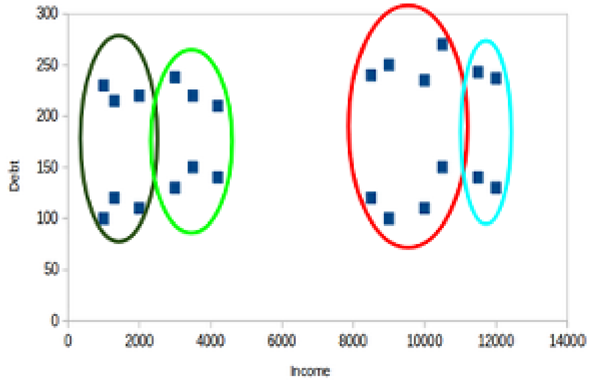


Properties of Clustering Algorithm

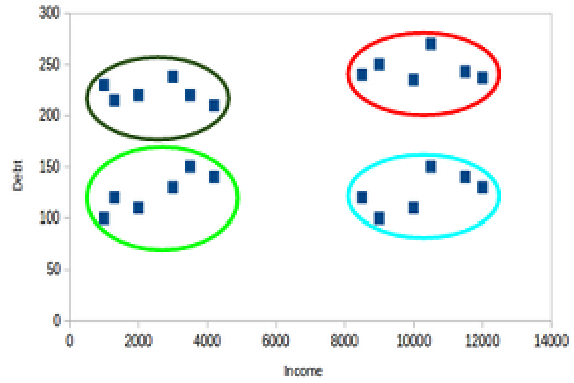
All the data points in a cluster should be similar to each other.



The data points from different clusters should be as different as possible.



Case - I



Case - II

Case I or Case II which is better?
Obviously, Case II

What is Similarity

- In the field of cluster analysis, this **similarity** plays an important part.
- Now, we shall learn how similarity (this is also alternatively judged as "dissimilarity") between any two data can be measured.

The quality or state of being similar; likeness; resemblance; as, a similarity of features.
Webster's Dictionary

Similarity is hard to define, but… "*We know it when we see it*"

The real meaning of similarity is a philosophical question. We will take a more pragmatic approach.

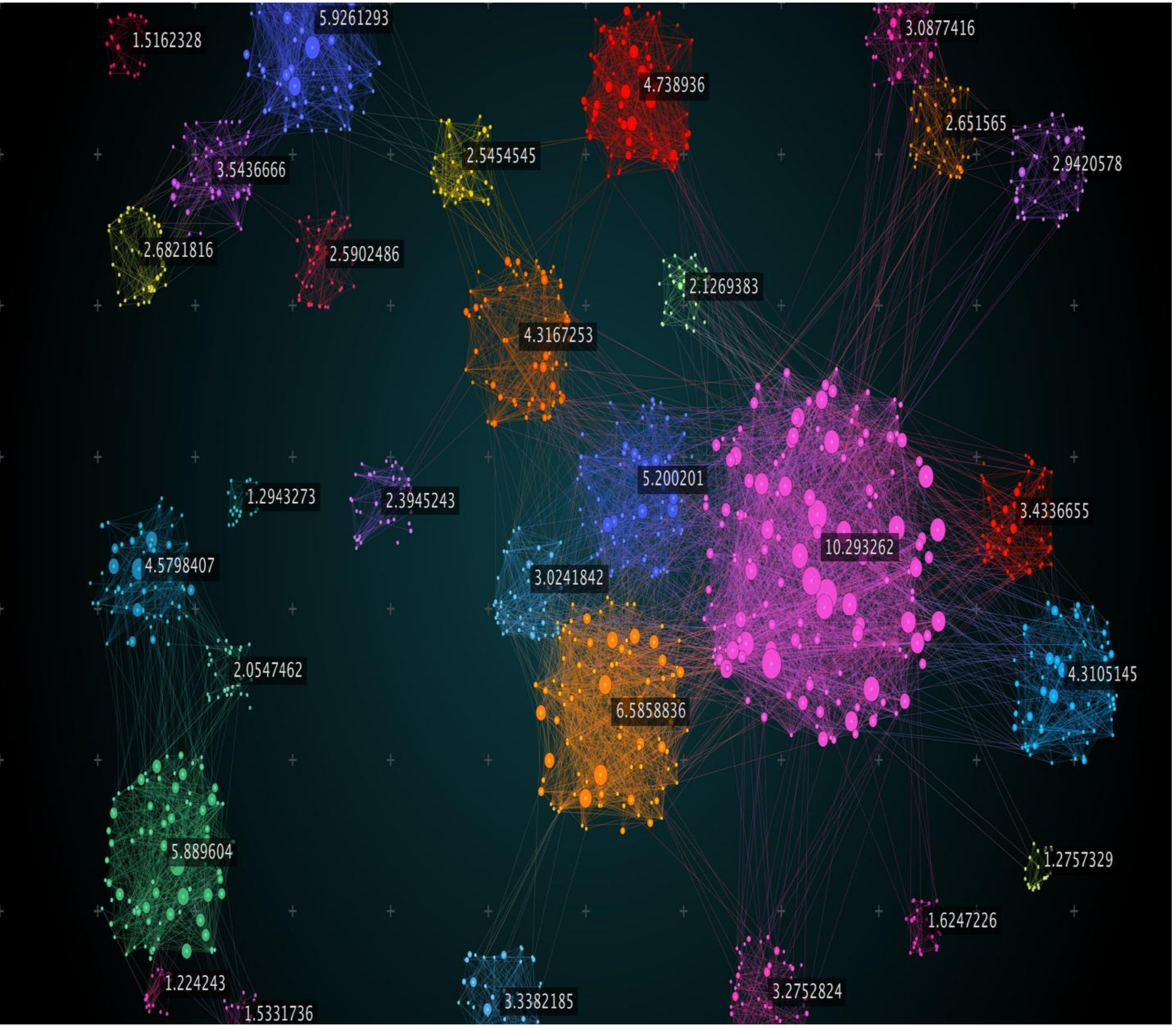
Measuring similarity between two data points in a cluster

The similarity between data points can be established using different distance metrics. The lesser the distance the more the similarity between data points.

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$



Non-Hierarchial Clustering (k-Means Clustering)

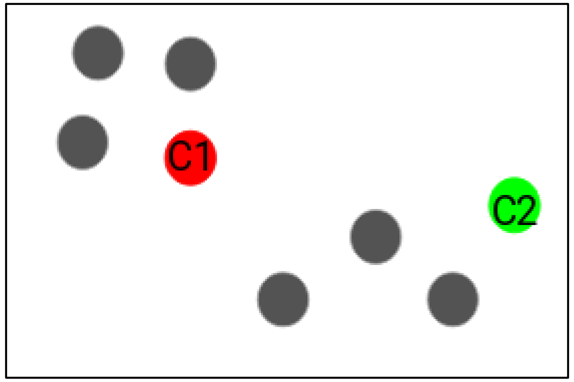
How to Apply K-Means Clustering Algorithm?

- We want to apply k-means to create clusters for these points.

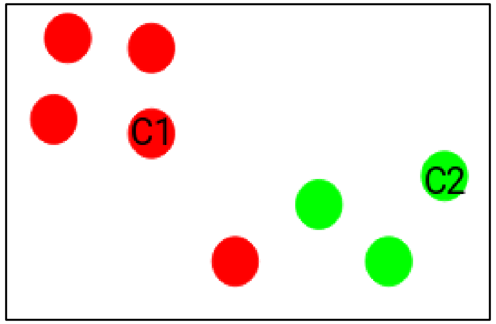
K-Means Clustering Algorith(Cont.)

Step 1: Choose the number of clusters k

Step 2: Select k random points from the data as centroids

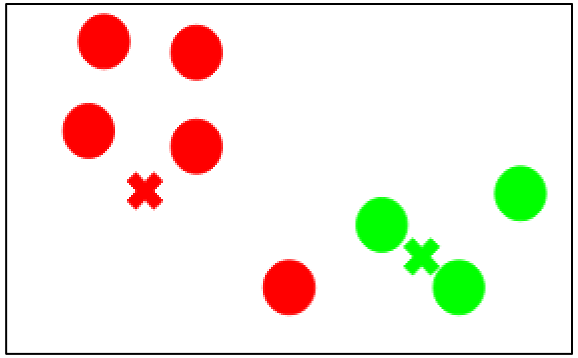


Step 3: Assign all the points to the closest cluster centroid

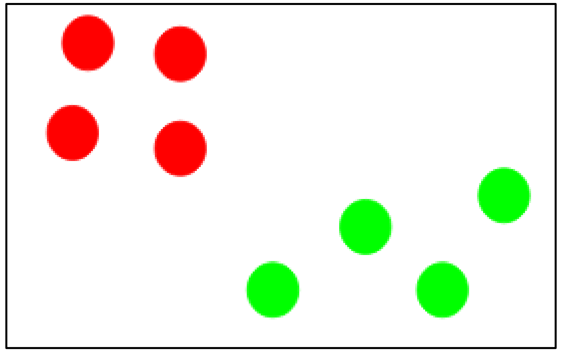


K-Means Clustering Algorithm(Cont.)

- **Step 4: Recompute the centroids of newly formed clusters**



Step 5: Repeat steps 3 and 4

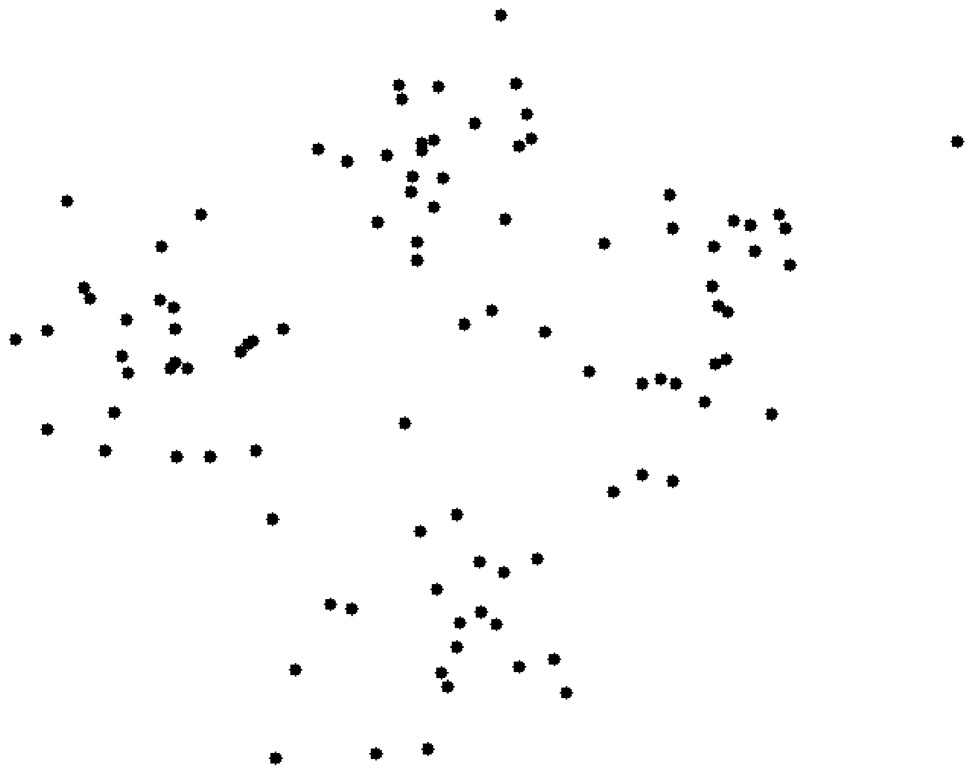


K-Means Clustering Algorithm

Algorithm 1 k-means algorithm

1: Specify the number k of clusters to assign.
2: Randomly initialize k centroids.
3: **repeat**
4:     **expectation:** Assign each point to its closest centroid.
5:     **maximization:** Compute the new centroid (mean) of each cluster.
6: **until** The centroid positions do not change.

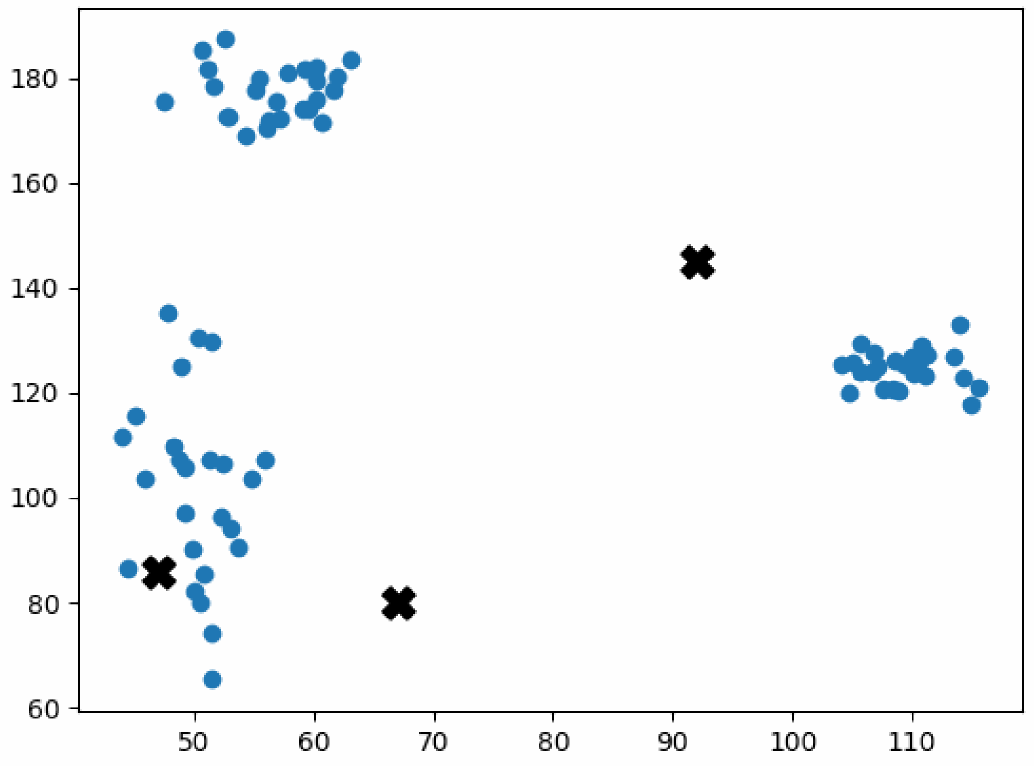
Visualisation of cluster formation using k-means clustering?



Assume K=4,

Drawbacks of k-means Clustering

- Due to poorly-selected initial positions for the three centroids, unrealistic clusters are created.



Stopping Criteria for K-Means Clustering

- Centroids of newly formed clusters do not change
- Points remain in the same cluster
- Maximum number of iterations is reached

Finding optimal value of K (Elbow Method)

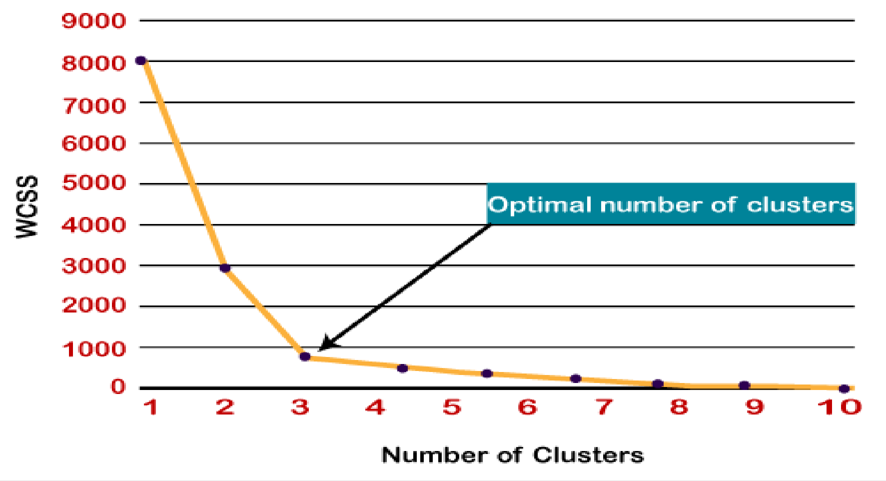
To find the optimal value of clusters, the elbow method follows the below steps:

1. It executes the K-means clustering on a given dataset for different K values (ranges from 1-10).
2. For each value of K, calculates the WCSS value.

a. Plots a curve between calculated WCSS values and the number of clusters K.

b. The sharp point of bend or a point of the plot looks like an arm, then that point is considered as the best value of K.

c. Since the graph shows the sharp bend, which looks like an elbow, hence it is known as the elbow method.

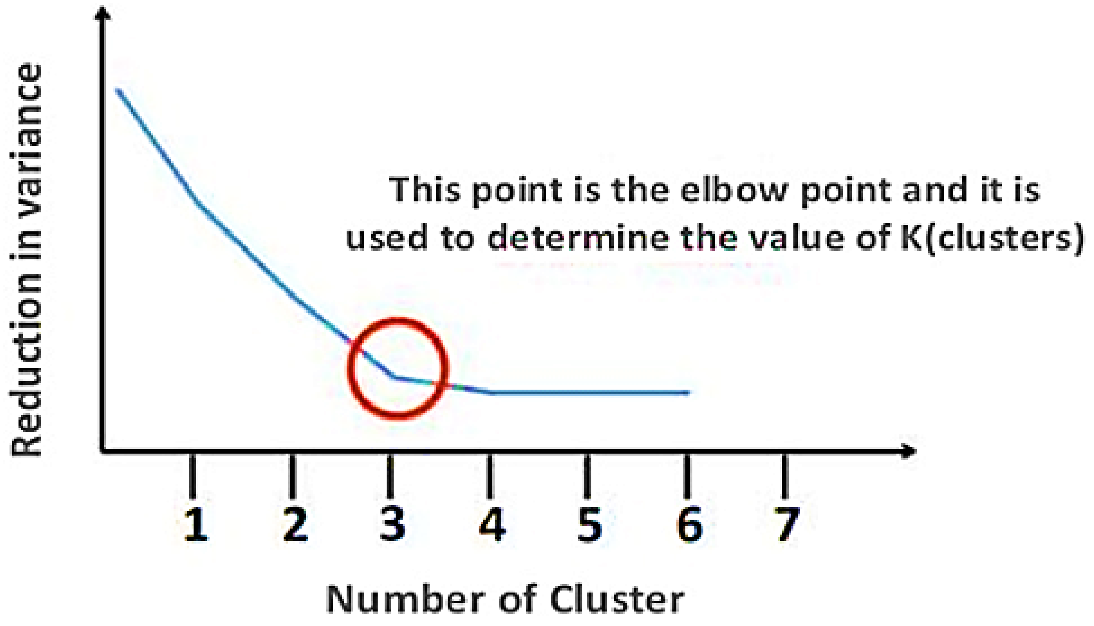


- for 3 clusters WCSS (Within Cluster Sum of Squares) can be calculated as follows:

$$WCSS = \sum P_{i\ in\ cluster1} distance(P_i, C_1)$$
$$+ \sum P_{i\ in\ cluster2} distance(P_i, C_2) + \sum P_{i\ in\ cluster3} distance(P_i, C_3)$$

$\sum P_{i\ in\ cluster1} distance(P_i, C_1)$: It is the sum of the square of the distances between each data point and its centroid within a cluster1 and the same for the other two terms.

How will you find value of k



No. of iterations Vs. WCSS Plot