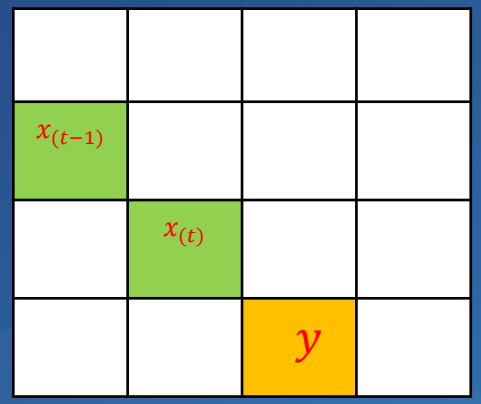
RNN: simple RNN

RNN is usually used to do the prediction for sequential data

Why RNN ?

There are two radar points obtained at $(t-1)$ and (t), and we want to predict the radar point at $(t+1)$

Training



Therefore, in order to make the train the model, we have two points in the training dataset:

$x_{(t-1)}$: the radar point obtained at $(t-1)$

$x_{(t)}$: the radar point obtained at (t)

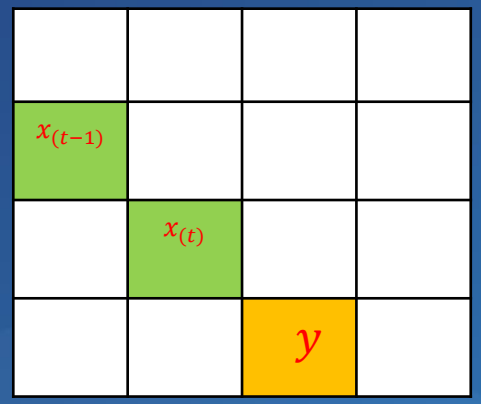
y: the truth in the training data

RNN is usually used to do the prediction for sequential data

Why RNN ?

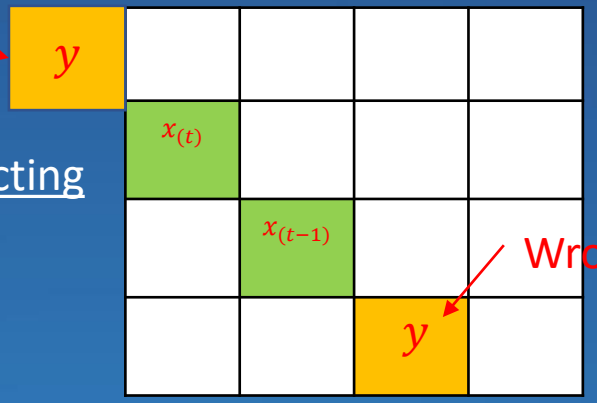
There are two radar points obtained at $(t-1)$ and (t), and we want to predict the radar point at $(t+1)$

Training



Expected right prediction

Predicting



Wrong prediction

Therefore, in order to make the train the model, we have two points in the training dataset:

$x_{(t-1)}$: the radar point obtained at $(t-1)$

$x_{(t)}$: the radar point obtained at (t)

y: the truth in the training data

In the dataset used for predicting, $x_{(t-1)}$ and $x_{(t)}$ located in opposite locations, however from the ANN training process, we are not able to recognize this sequential difference here, therefore the prediction will be in a wrong place (followed the training dataset)

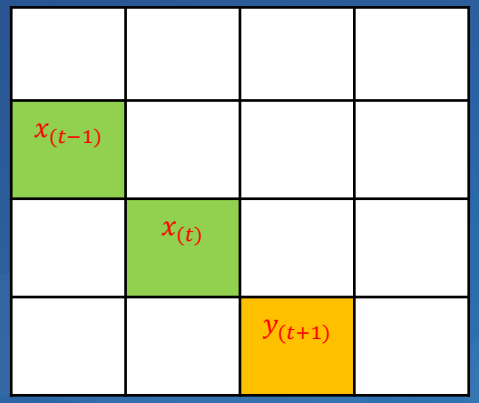
RNN is usually used to do the prediction for sequential data

Why RNN ?

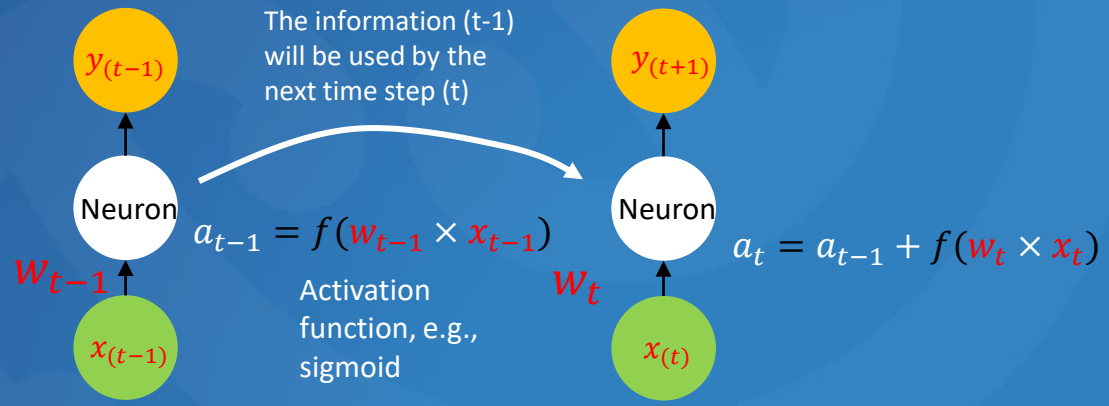
There are two radar points obtained at $(t-1)$ and (t), and we want to predict the radar point at $(t+1)$

In order to make such a prediction right, we need a sequential of training dataset, e.g.,

Training



And during the training, the model needs to be able to ”connect” these two training datasets, e.g.,

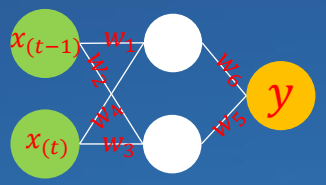


By doing this, the neuro information from $(t-1)$ to (t) are recorded

RNN is usually used to do the prediction for sequential data

Why RNN ?

There are two radar points obtained at $(t-1)$ and (t), and we want to predict the radar point at $(t+1)$



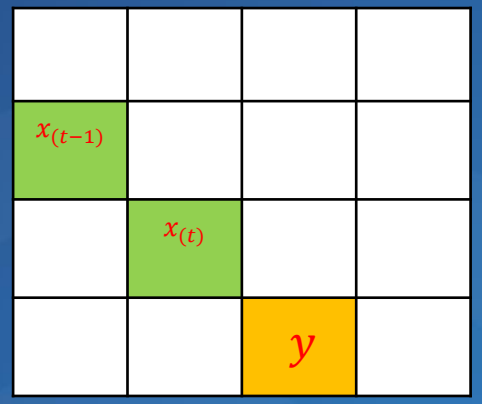
ANN

If it is an ANN, there are 6 weights to be estimated

w_1 -> w_6

From the back propagation method

Training

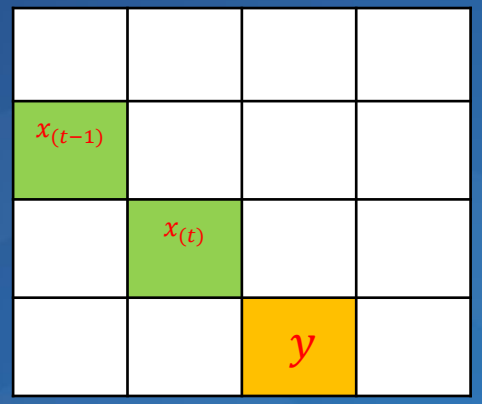


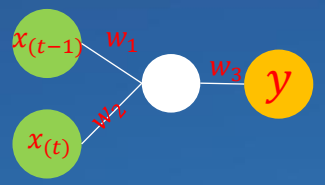
RNN is usually used to do the prediction for sequential data

Why RNN ?

There are two radar points obtained at $(t-1)$ and (t), and we want to predict the radar point at $(t+1)$

Training





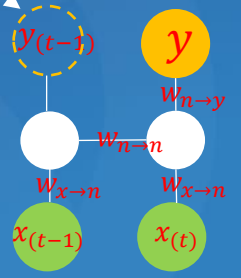
ANN

If it is an ANN, there are 3 weights to be estimated

w_1 -> w_3

From the back propagation method

Pseudo output, usually not used in training



CNN

If it is an CNN, there are also 3 weights to be estimated, but one more neuro to be added

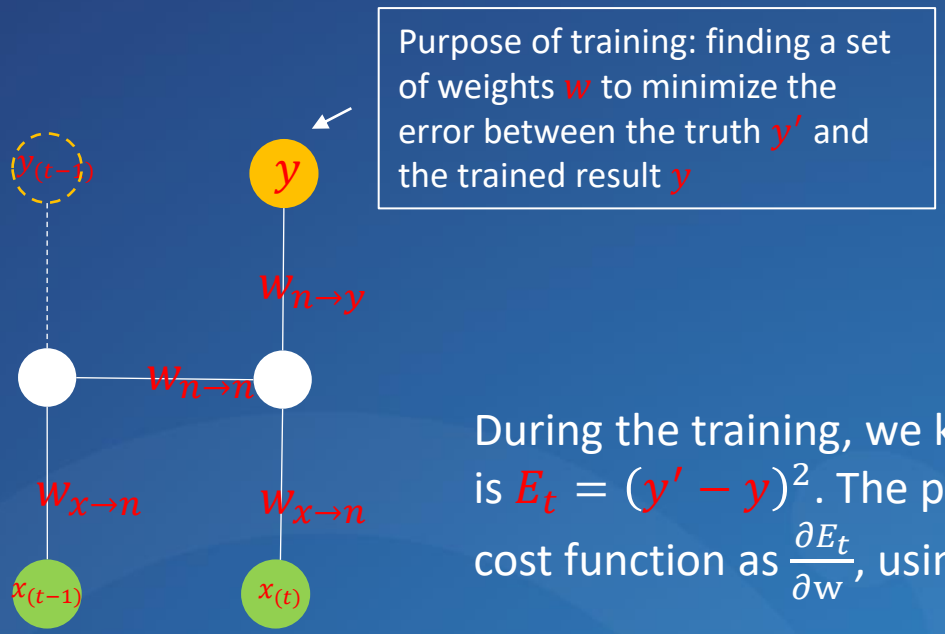
$w_{n\to y}$: the weights for connecting neuro (activation function) to output

$w_{x\to n}$: the weights for connecting input and neuro

$w_{n\to n}$: the weights for connecting neuro from previous timestep to the next

RNN is usually used to do the prediction for sequential data

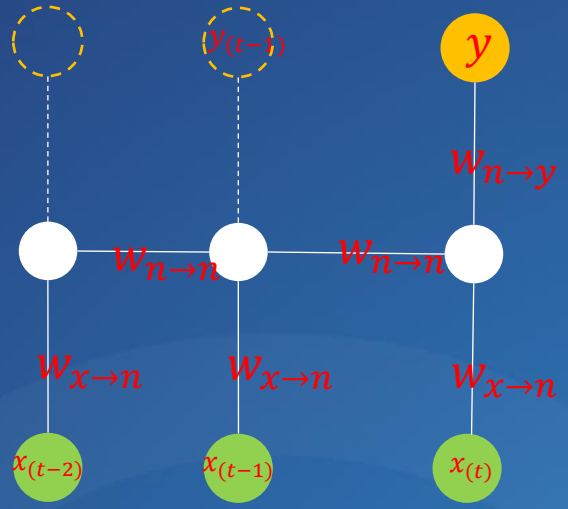
How RNN can be trained ?



During the training, we know the ground truth y', assuming the trained result is y, so the error is $E_t = (y' - y)^2$. The purpose of training is to obtain the minimum E_t, so by setting up the cost function as $\frac{\partial E_t}{\partial \mathrm{w}}$, using a backpropagation method we are able to obtain all the weights.

RNN is usually used to do the prediction for sequential data

How RNN can be trained ? (how about we have more than one unfolded layers)

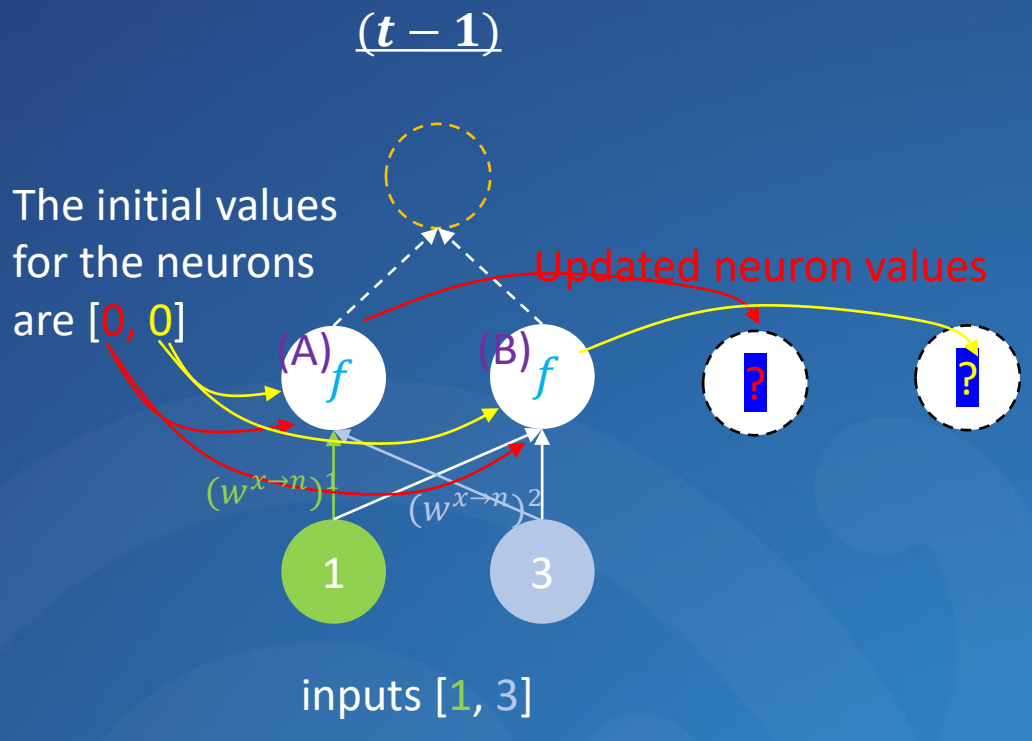


Note that in RNN, the weights are the same from different time steps (as the left figure shown)

** The left figure just shows an extremely simplified situation, in reality, we will have multiple neuros in multiple hidden layers*

RNN is usually used to do the prediction for sequential data

How RNN do the prediction ?



The neuron value in a RNN is updated based on

$$x_t' = c + f(w_t \cdot x_t)$$

Therefore, for the neuro "A" the updated value of neuron is

$$0 + 0 + f[(w^{x\to n})^1 \cdot 1] + f[(w^{x\to n})^2 \cdot 3]$$

Contributio n from 1st initial neuro

Contributio n from 2nd initial neuro

Contribution from 1st input with the activation function

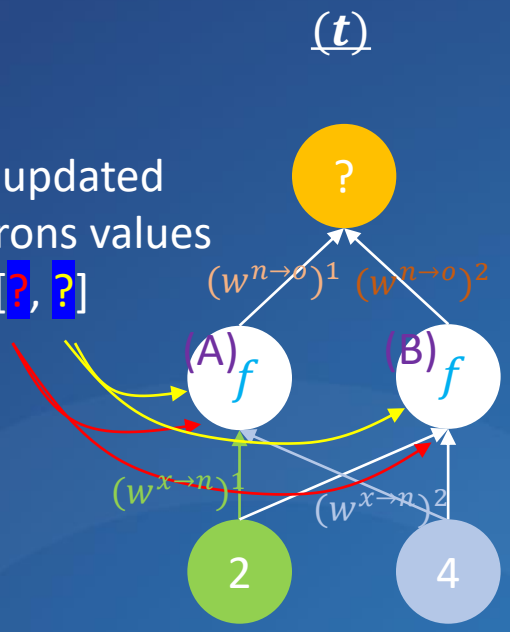
Contribution from the 2nd input with the activation function

We can update the neuro "B" the same way

If it is not the last time step, we don't need to produce the output

RNN is usually used to do the prediction for sequential data

How RNN do the prediction ?



The neuron value in a RNN is updated based on

$$x_t' = c + f(w_t \cdot x_t)$$

Therefore, for the neuro "A" the updated value of neuron is

$$X_1 = ? + ? + f[(w^{x\to n})^1 \cdot 2] + f[(w^{x\to n})^2 \cdot 4]$$

Contribution from the previous 1st neuro

Contribution from the previous 2nd initial neuro

Contribution from 1st input with the activation function

Contribution from the 2nd input with the activation function

We can update the neuro "B" the same way as X_2

So the final prediction would be

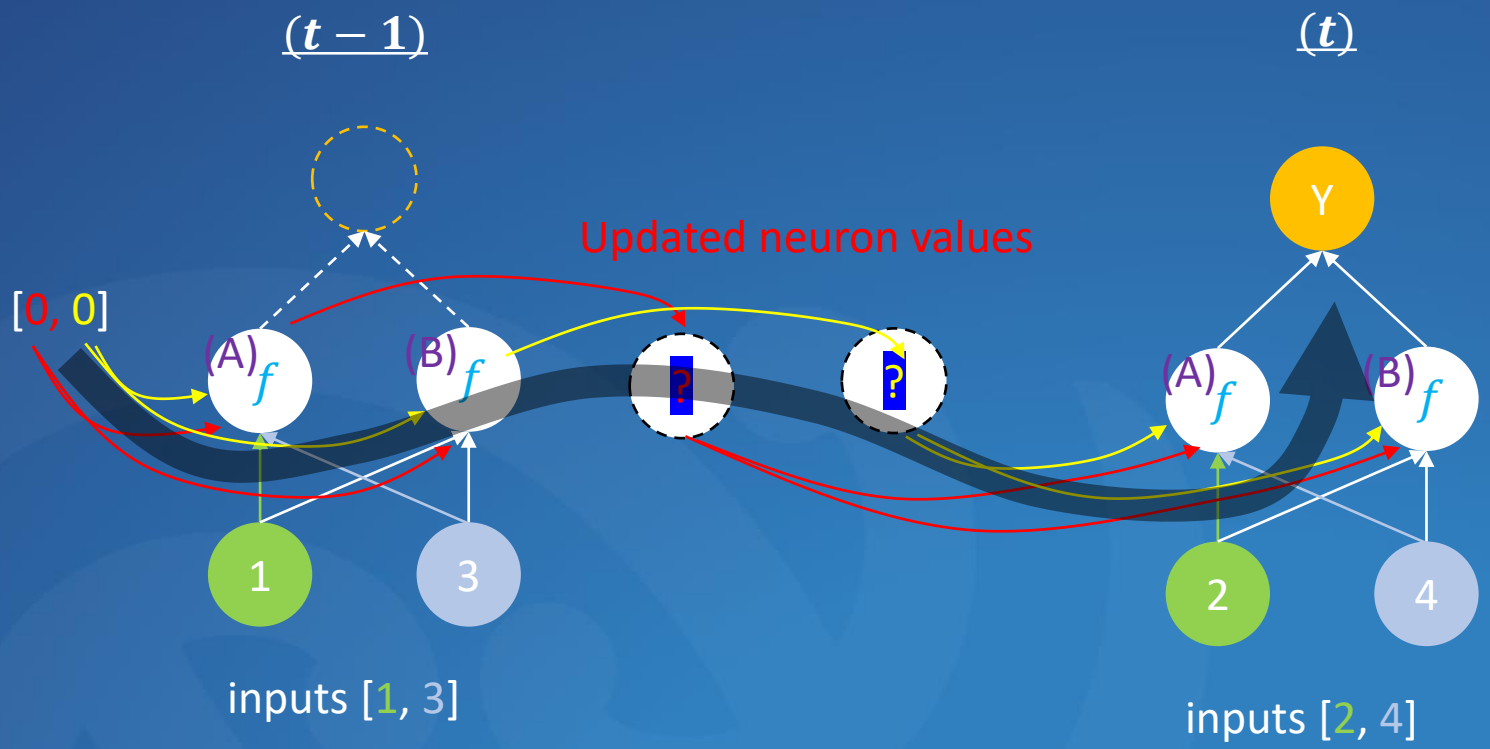
$$Y = (w^{n\to o})^1 X_1 + (w^{n\to o})^2 X_2$$

For the last timestep, we don't need to update the neurons anymore

RNN is usually used to do the prediction for sequential data

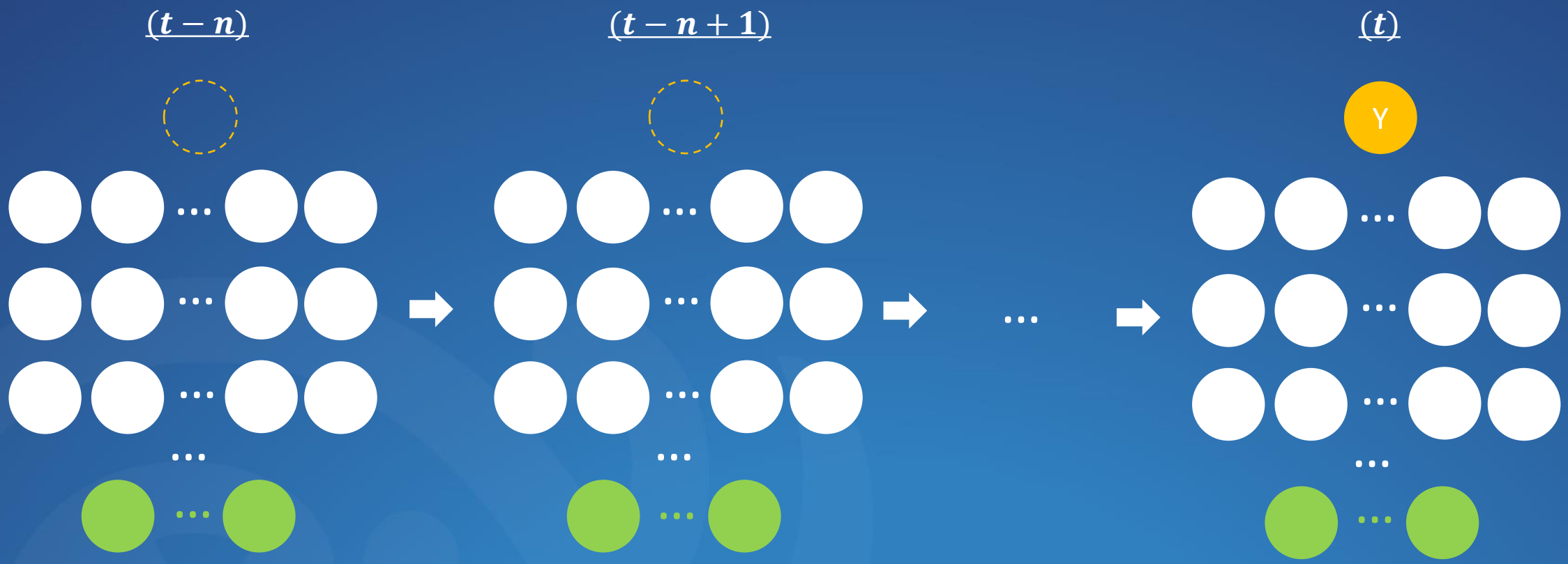
How RNN do the prediction ?

In summary, after we obtained all the weights from the training, the prediction can be done as below:



RNN is usually used to do the prediction for sequential data

Of course, RNN can become deep, and can use many timesteps, e.g.,



RNN is usually used to do the prediction for sequential data

Of course, RNN can be bidirectional, e.g.,