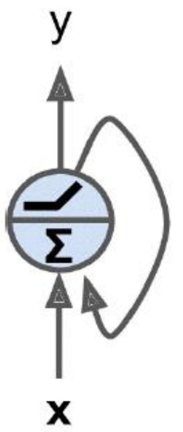
Recurrent Neural Networks

CMPUT328

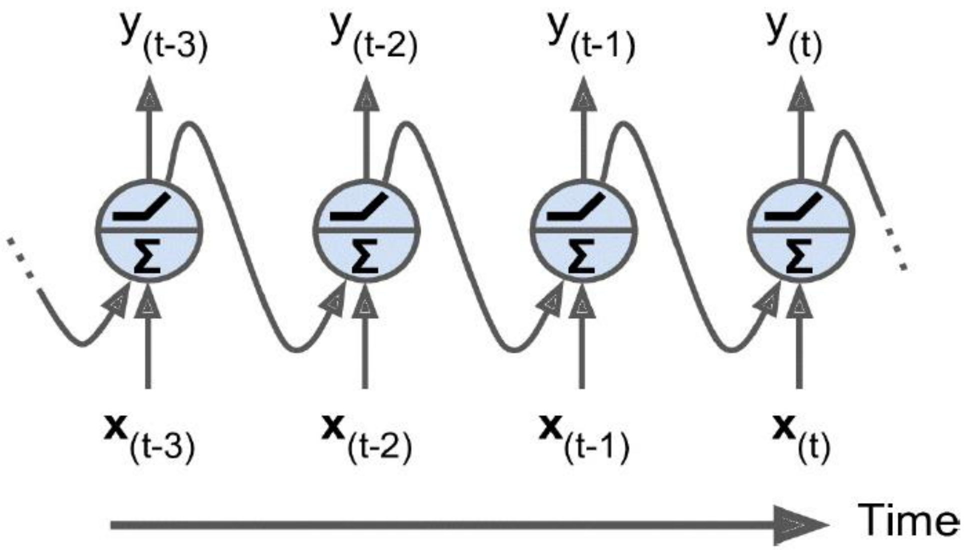
Nilanjan Ray

Source: Hands on Machine Learning Book

RNN: Unrolling through time



A RNN neuron

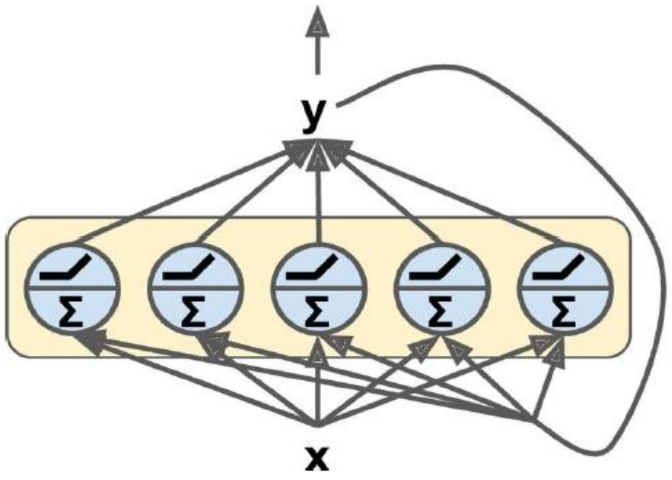


Unrolling through time

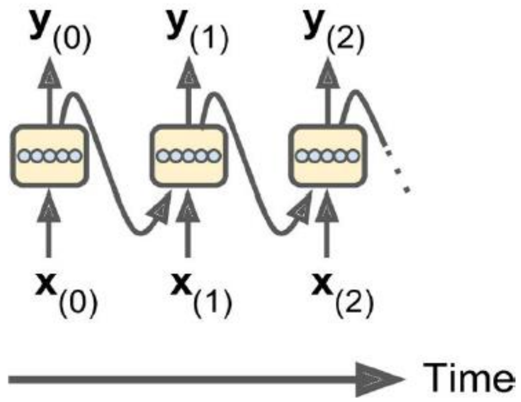
Note: The unrolled RNN is a DAG

Why is the DAG structure important?

RNN: A Layer of recurrent neurons



RNN

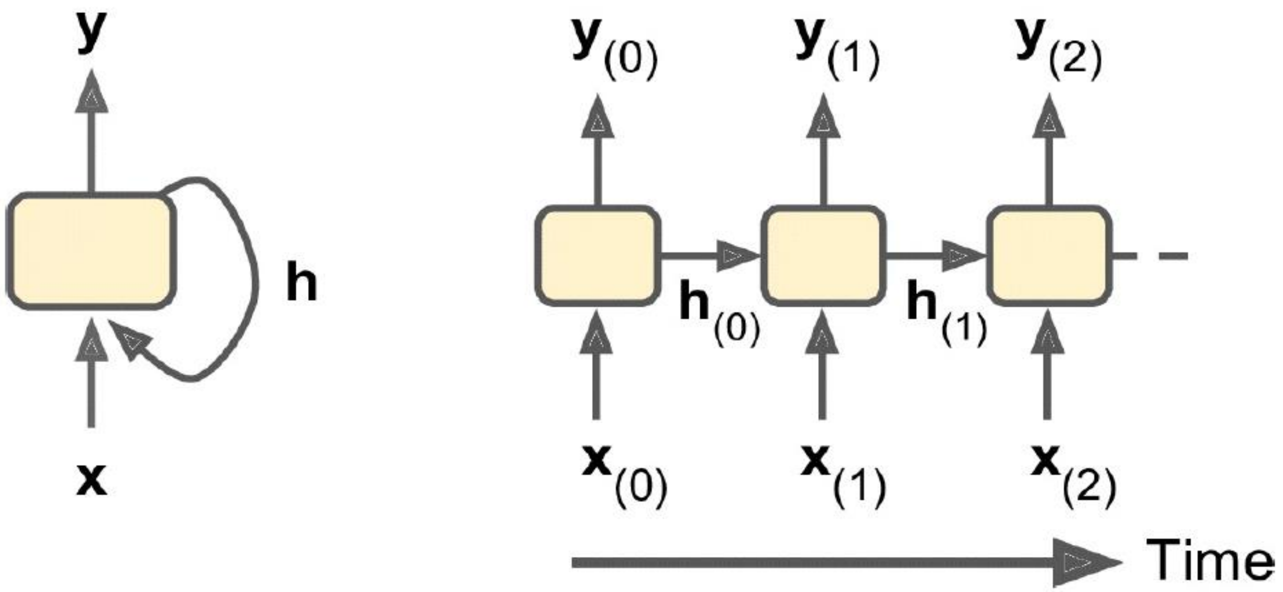


Unrolled through time

$Y(t) = \varphi(X(t)W_x + Y(t-1)W_y + b)$ Note that parameters W_x and W_y are shared.

φ is a non-linear activation function, such as ReLU

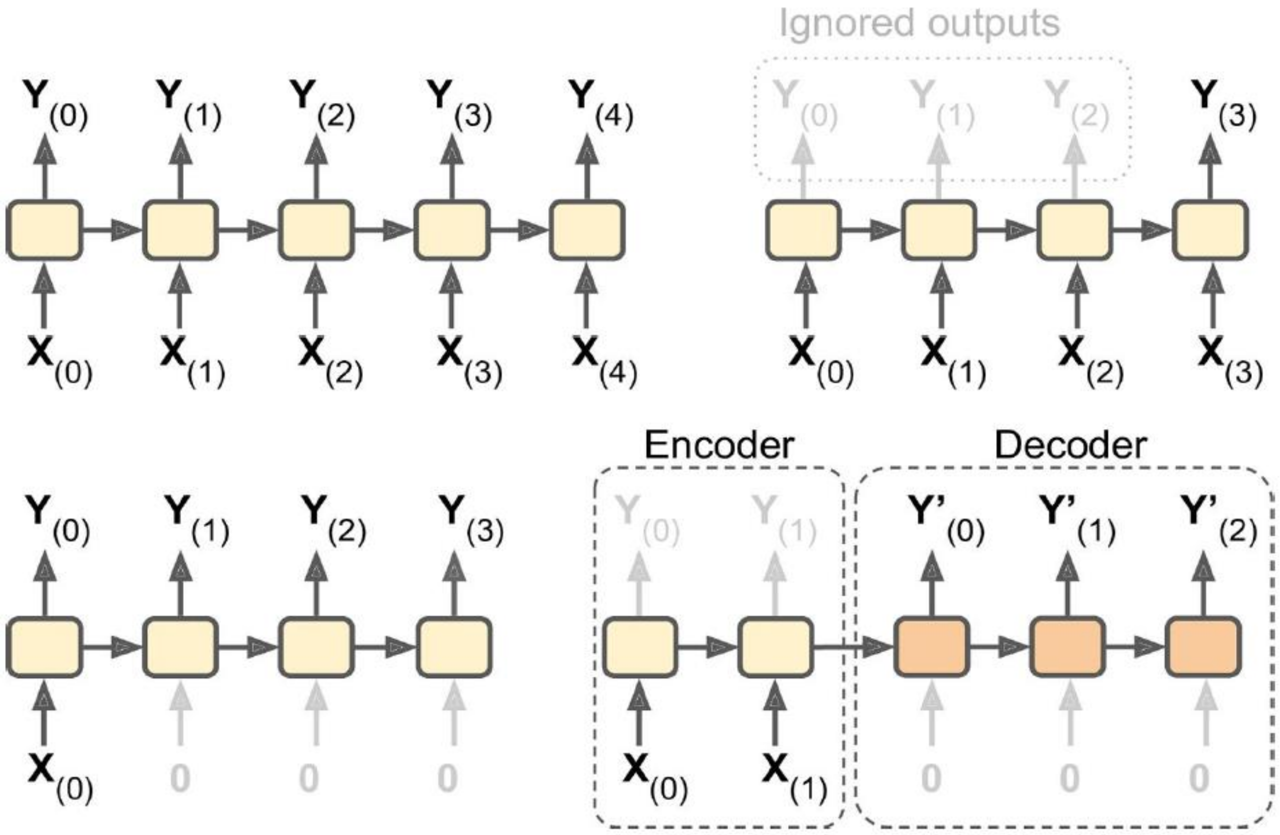
RNN: Memory cell



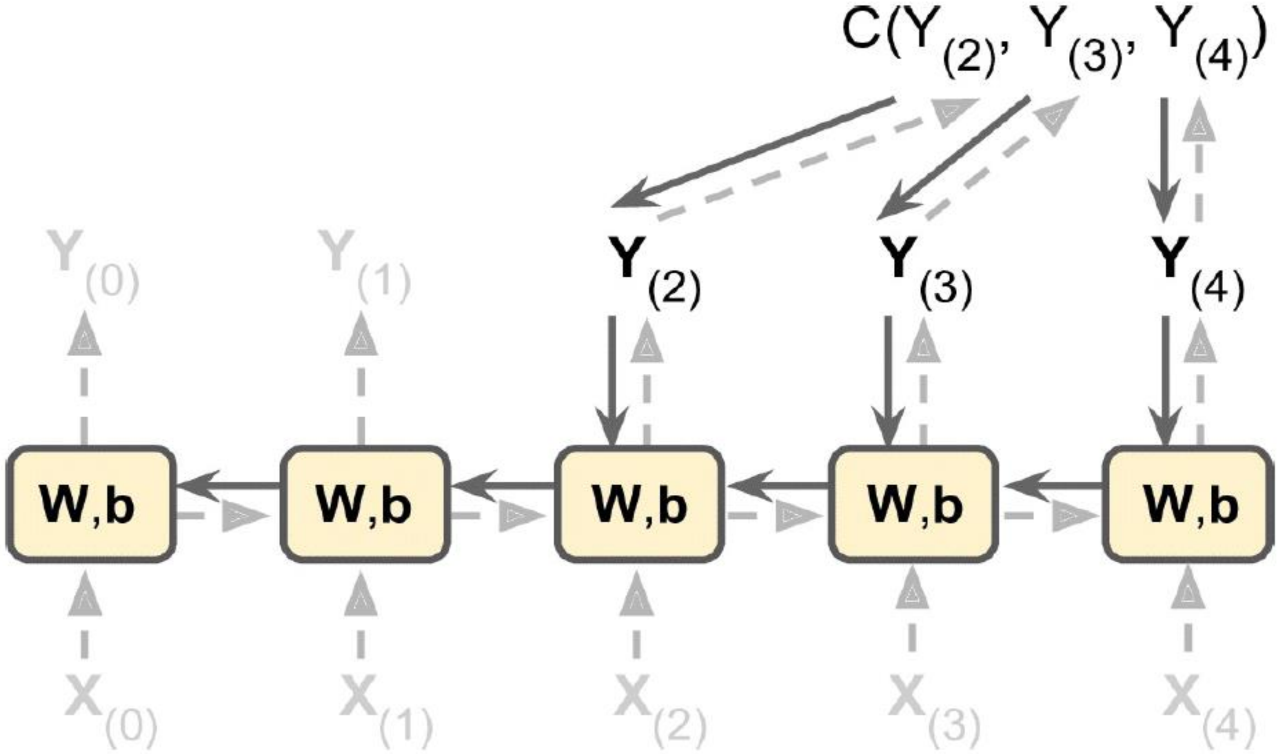
$$\mathrm{h}(t) = f(h(t-1), x(t))$$

h(*t*) acts as a memory cell holding “memories” from past until time point *t*.

Types of inputs and outputs in RNN



Backpropagation through time

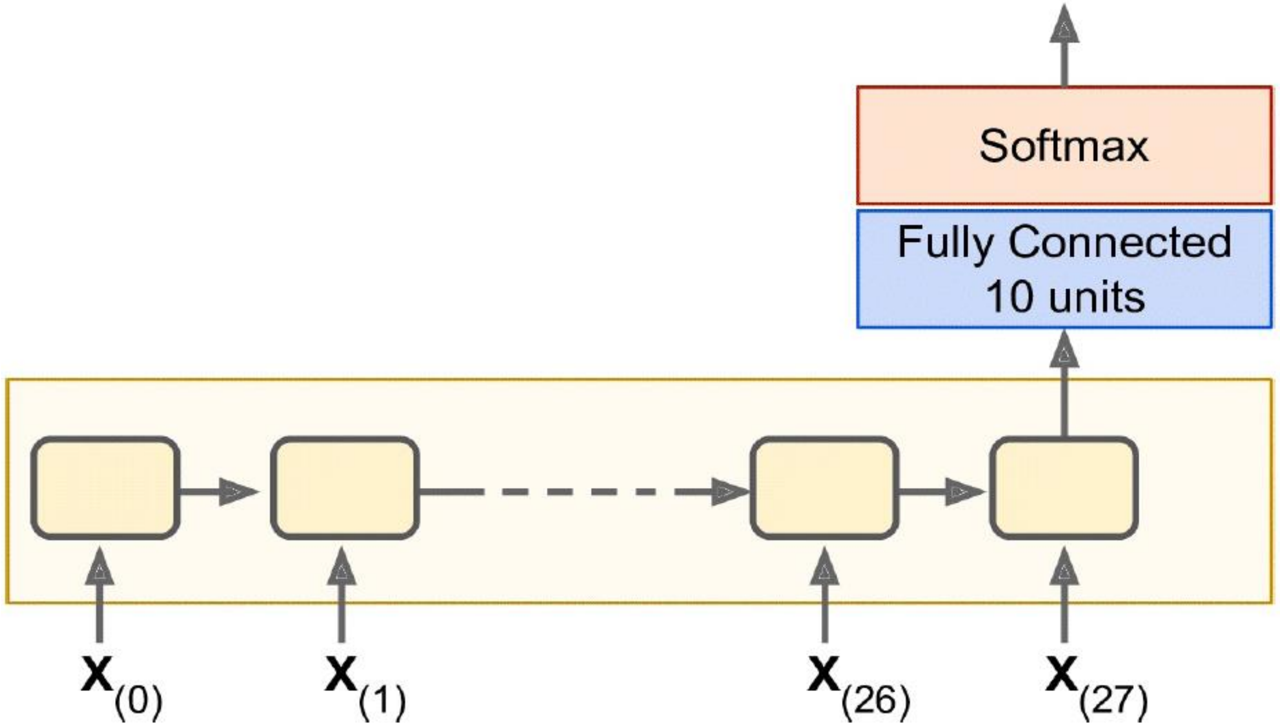


C: cost function
Dotted lines: forward pass
Solid lines: backward pass

Note: parameters are shared!

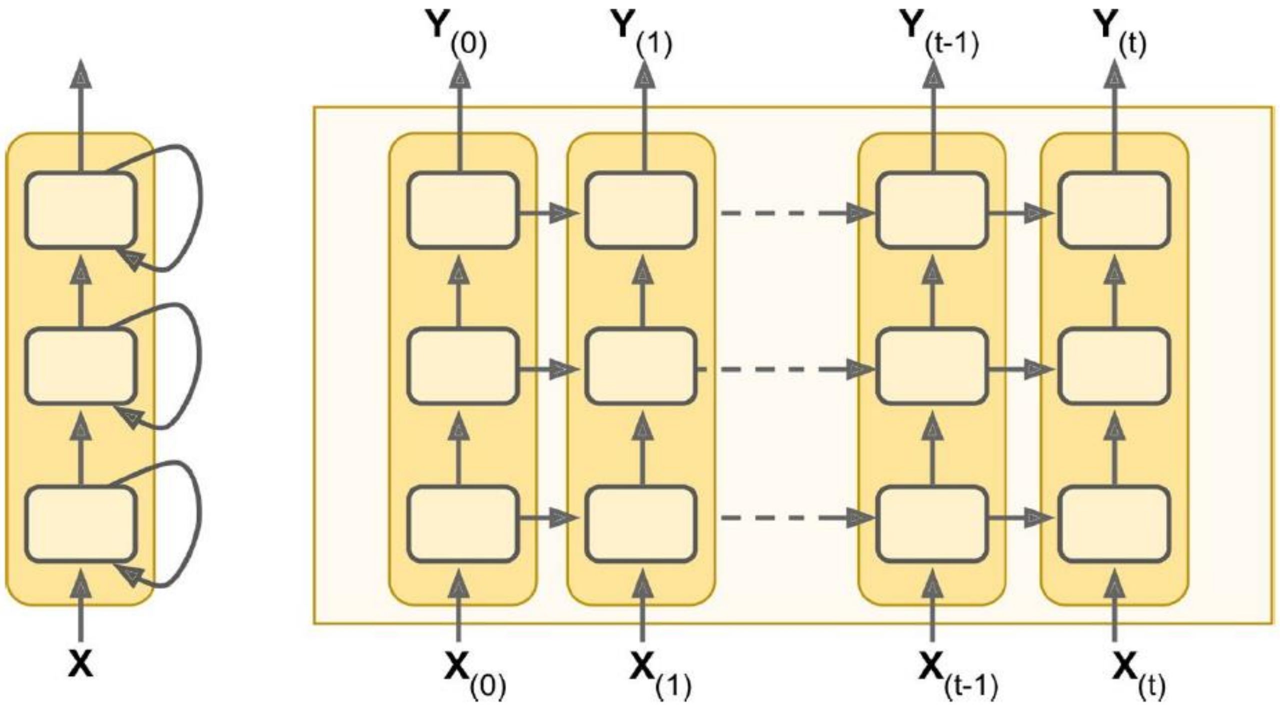
Backpropagation through time: normal backpropagation through unrolled RNN

MNIST classification: Using sequence!



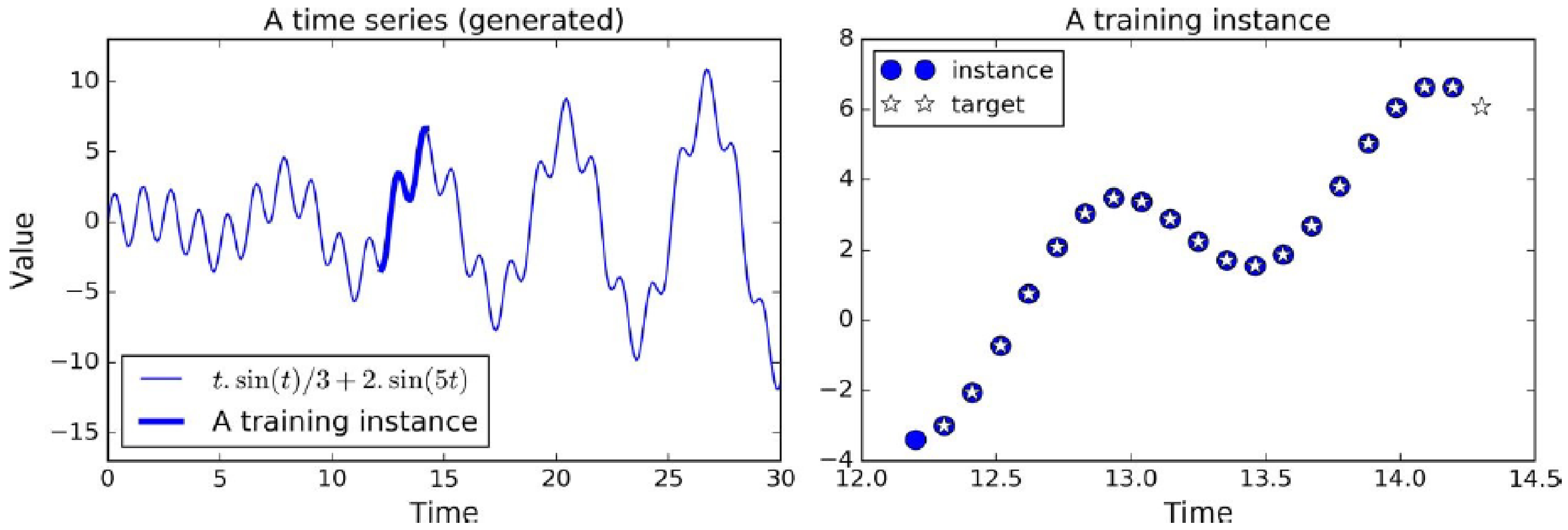
Treat each as a sequence of 28 rows

Deep (Multi-layer) RNN

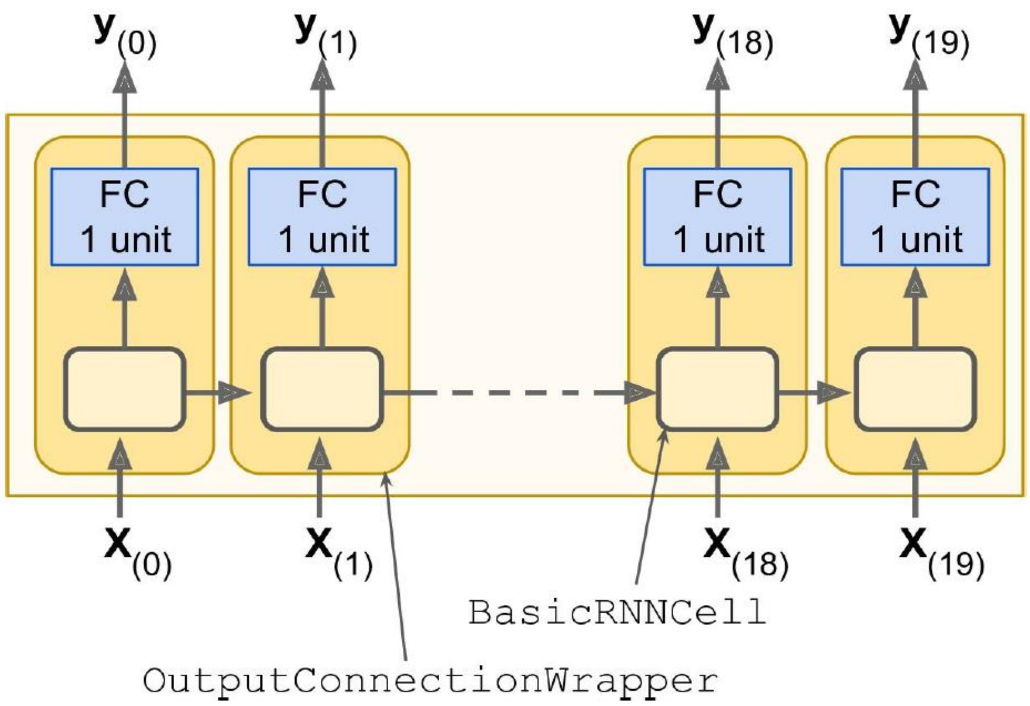


Example implementation for MNIST classification

Predicting a time series

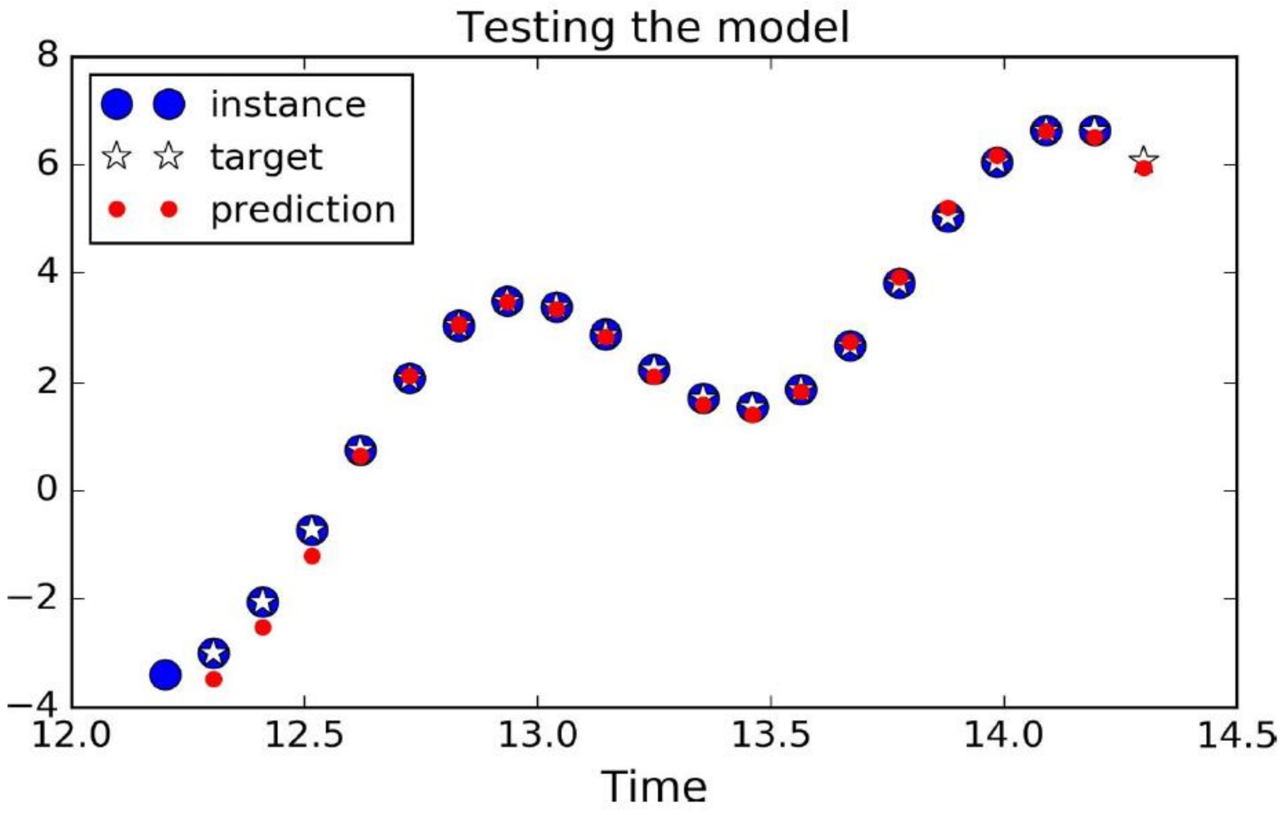


Predicting a time series…

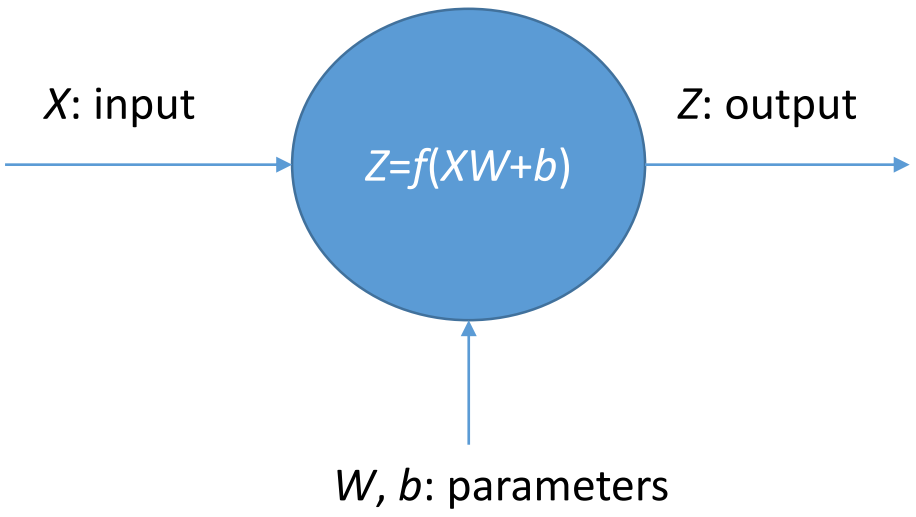


Wrapper function in TensorFlow for dimensionality reduction

Predicting a time series…



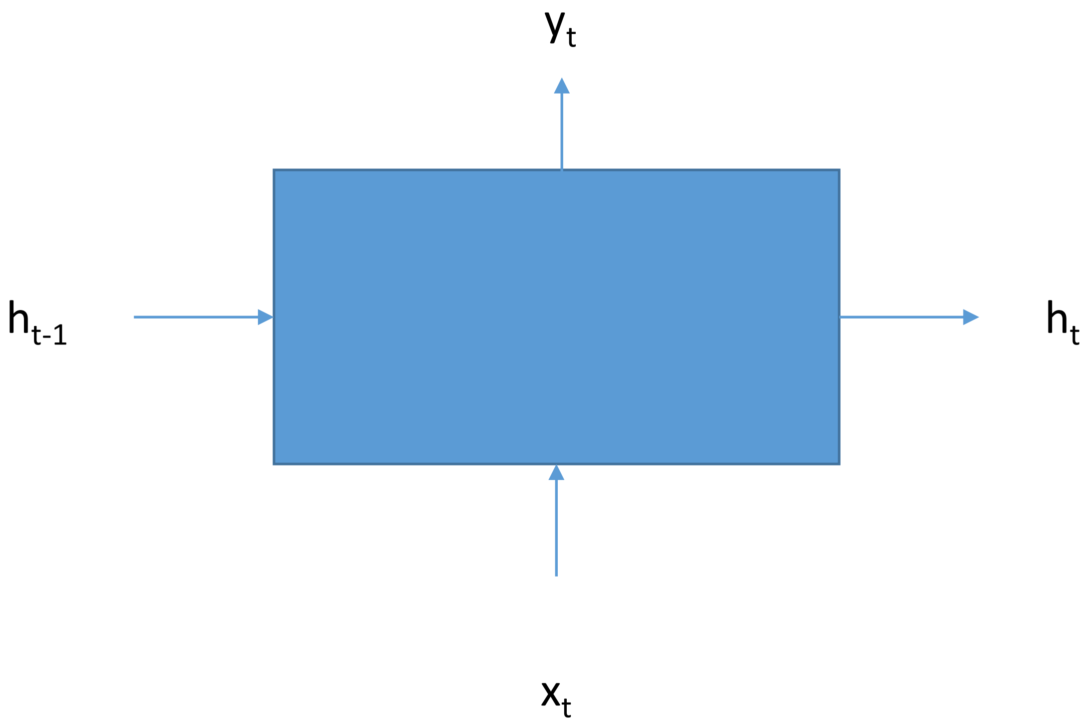
Backpropagation in RNN



Let's recall rule of BP
for a computation node

$$\delta X = [f'(XW + b) \cdot \delta Z] * W^T \qquad \delta W = X^T * [f'(XW + b) \cdot \delta Z] \qquad \delta b = \sum_k [f'(XW + b) \cdot \delta Z]_{k,:}$$

Backpropagation in RNN…



How do we apply BP here?

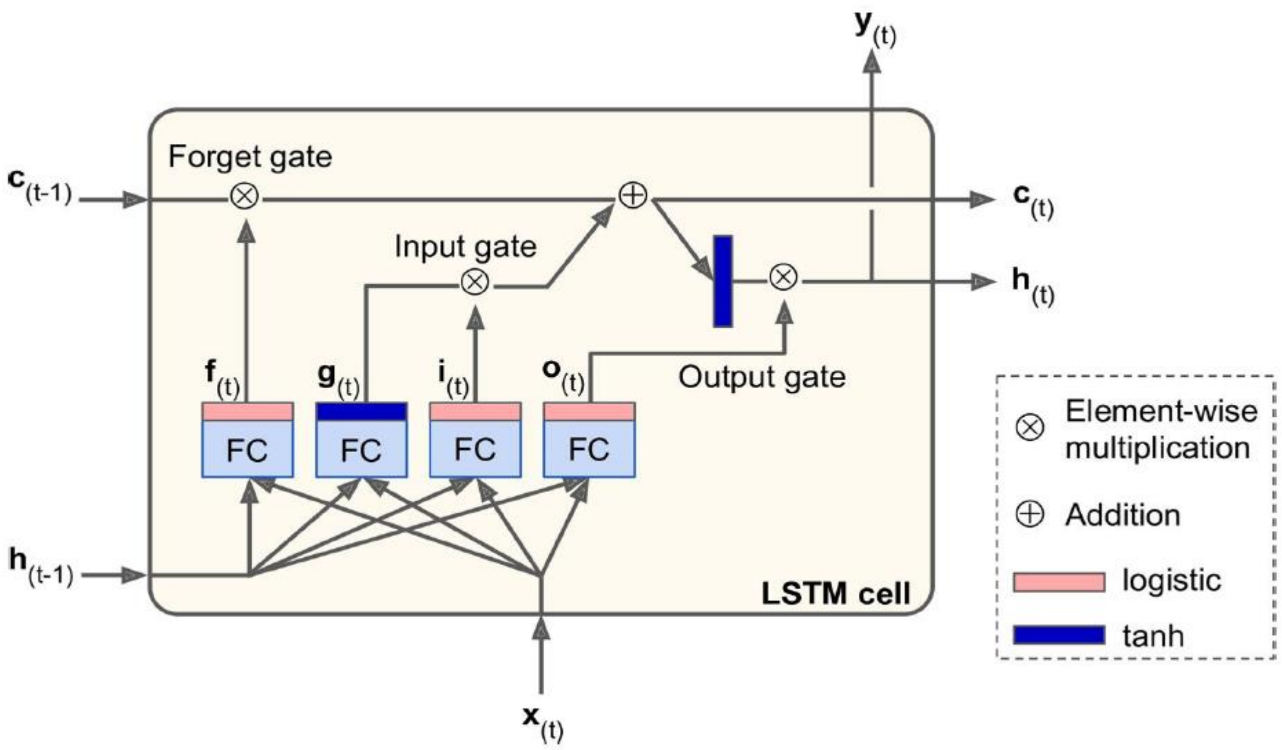
$$h_t = f(x_t W_x^h + h_{t-1} W_h^h + b^h)$$

$$y_t = g(x_t W_x^y + h_{t-1} W_h^y + b^y)$$

Vanishing / Exploding gradient problem

- Use BP formula to understand this issue

Long Short-Term Memory (LSTM)

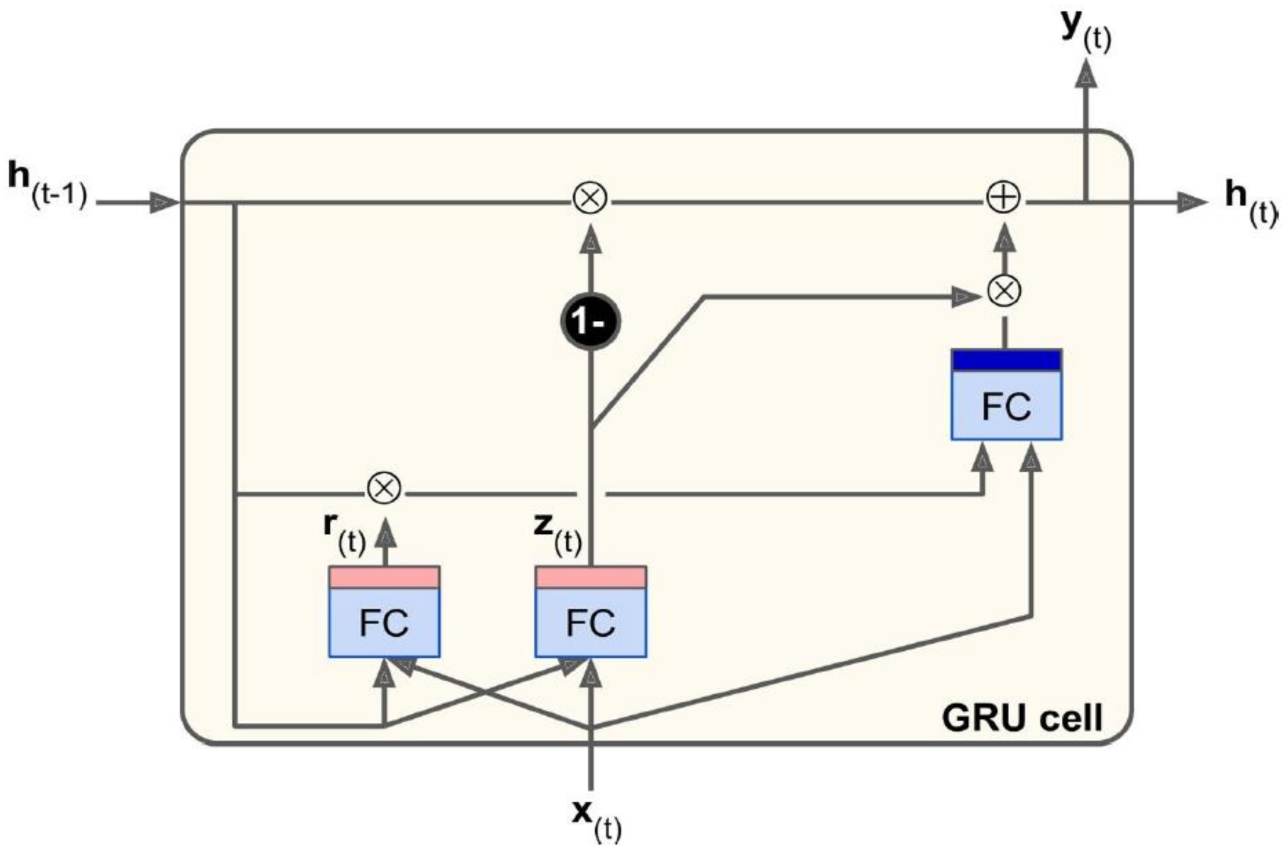


LSTM computations:

$$
\begin{aligned}
\boldsymbol{i}_{(t)} &= \sigma(\boldsymbol{W}_{xi}{}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{hi}{}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{i}) \\
\boldsymbol{f}_{(t)} &= \sigma(\boldsymbol{W}_{xf}{}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{hf}{}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{f}) \\
\boldsymbol{o}_{(t)} &= \sigma(\boldsymbol{W}_{xo}{}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{ho}{}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{o}) \\
\boldsymbol{g}_{(t)} &= \tanh(\boldsymbol{W}_{xg}{}^{T} \cdot \boldsymbol{x}_{(t)} + \boldsymbol{W}_{hg}{}^{T} \cdot \boldsymbol{h}_{(t-1)} + \boldsymbol{b}_{g}) \\
\boldsymbol{c}_{(t)} &= \boldsymbol{f}_{(t)} \otimes \boldsymbol{c}_{(t-1)} + \boldsymbol{i}_{(t)} \otimes \boldsymbol{g}_{(t)} \\
\boldsymbol{y}_{(t)} &= \boldsymbol{h}_{(t)} = \boldsymbol{o}_{(t)} \otimes \tanh(\boldsymbol{c}_{(t)})
\end{aligned}
$$

This architecture helps to mitigate vanishing/exploding gradient problem. Why?
Let’s use LSTM in our time series prediction.

Gated Recurrent Unit (GRU)



GRU computations;

$$
\begin{aligned}
\mathbf{z}_{(t)} &= \sigma\left(\mathbf{W}_{xz}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hz}{}^{T} \cdot \mathbf{h}_{(t-1)}\right) \\
\mathbf{r}_{(t)} &= \sigma\left(\mathbf{W}_{xr}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hr}{}^{T} \cdot \mathbf{h}_{(t-1)}\right) \\
\mathbf{g}_{(t)} &= \tanh\left(\mathbf{W}_{xg}{}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^{T} \cdot (\mathbf{r}_{(t)} \otimes \mathbf{h}_{(t-1)})\right) \\
\mathbf{h}_{(t)} &= (1 - \mathbf{z}_{(t)}) \otimes \tanh\left(\mathbf{W}_{xg}{}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{z}_{(t)} \otimes \mathbf{g}_{(t)}\right)
\end{aligned}
$$

GRU is a much more simplified recurrent unit; but it is almost as good as LSTM.