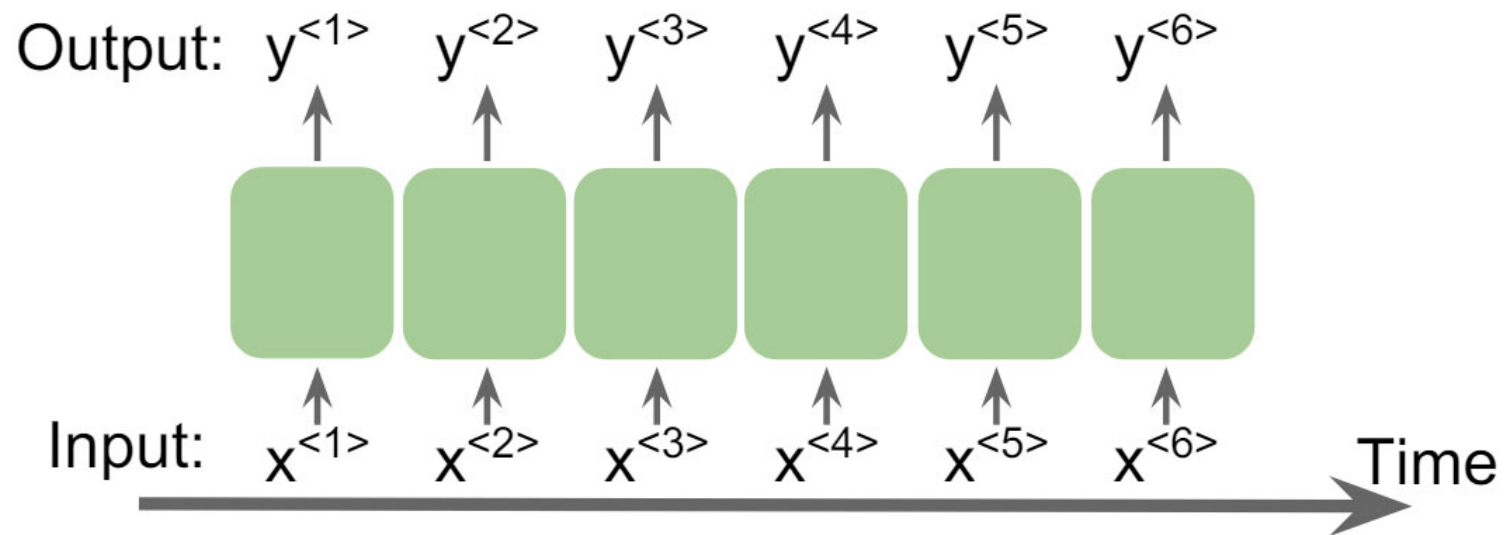


RNN

Sequence data: order matters

The movie my friend has **not** seen is good

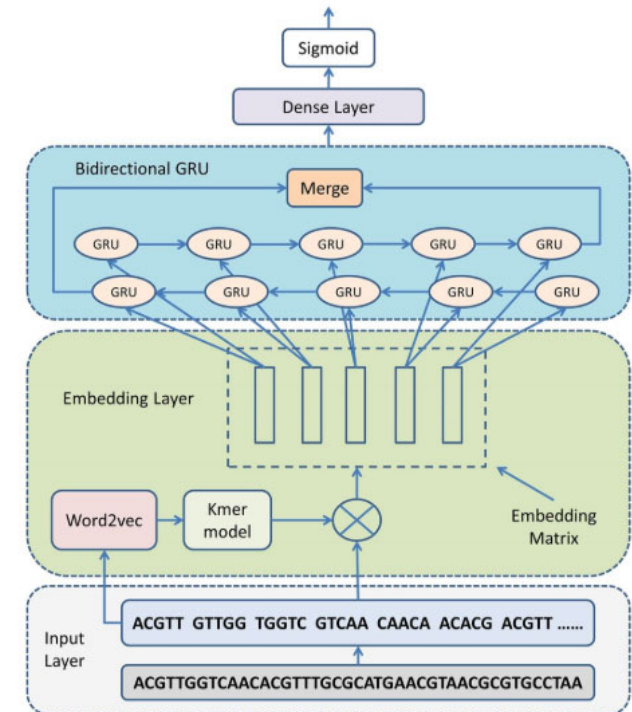
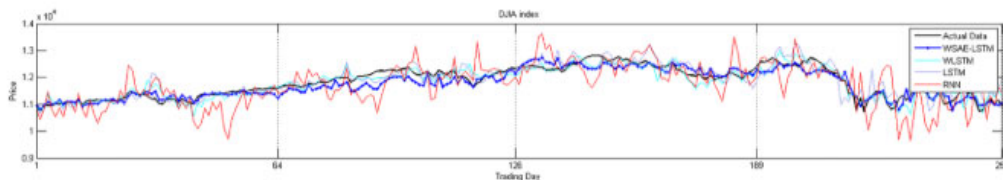
The movie my friend has seen is **not** good



Applications: working with Sequential Data

- Text classification
- Speech recognition (acoustic modeling)
- language translation
- ...

Stock market predictions



Shen, Zhen, Wenzheng Bao, and De-Shuang Huang. "[Recurrent Neural Network for Predicting Transcription Factor Binding Sites](#)." *Scientific reports* 8, no. 1 (2018): 15270.

DNA or (amino acid/protein)
sequence modeling

Applications: Speech Recognition

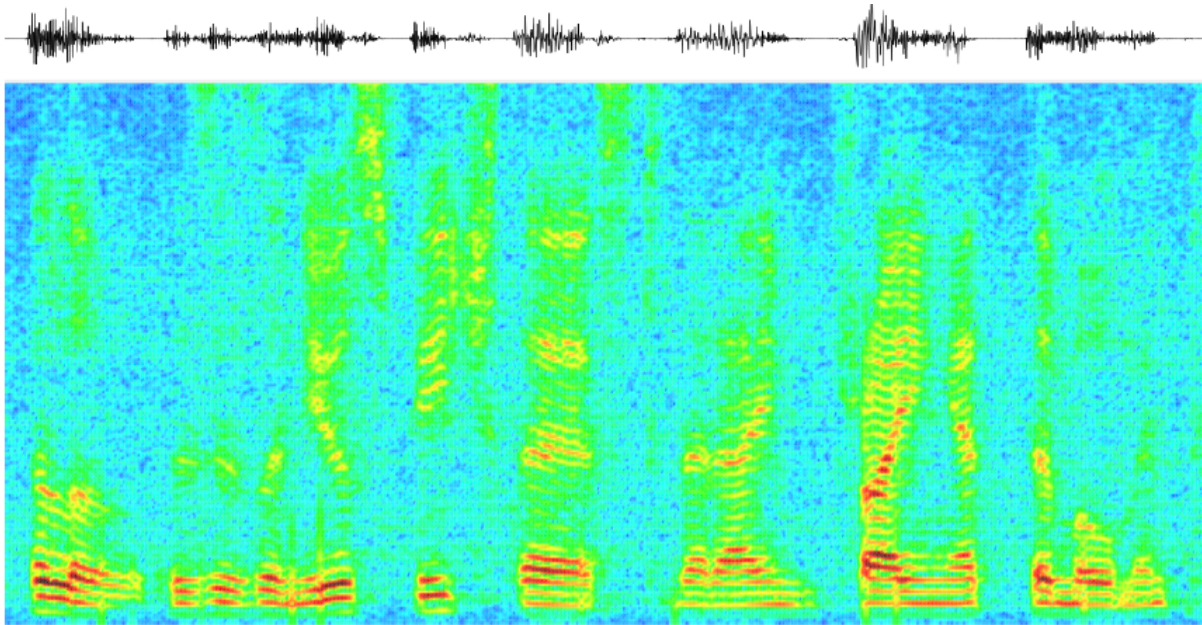


Figure: source

- Speech Recognition
 - ▶ Analyze a series of spectral vectors, determine what was said.
- Note: Inputs are sequences of vectors. Output is a classification result.

Application : Text analysis

Stephen Curry scored 34 points and was named the NBA Finals MVP as the Warriors claimed the franchise's seventh championship overall. And this one completed a journey like none other; after a run of five consecutive finals, then a plummet to the bottom of the NBA, and now a return to greatness just two seasons after having the league's worst record.

- Football or Basketball?

- Text Analysis

- ▶ E.g. analyze document, identify topic
 - Input series of words, output classification output
- ▶ E.g. read English, output Persian
 - Input series of words, output series of words

Application: Stock Market prediction

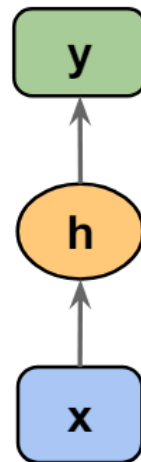


- Stock Market Prediction
 - ▶ Should I invest, vs. should I not invest in X?
 - ▶ Decision must be taken considering how things have fared over time.
- Note: Inputs are sequences of vectors. Output may be scalar or vector.

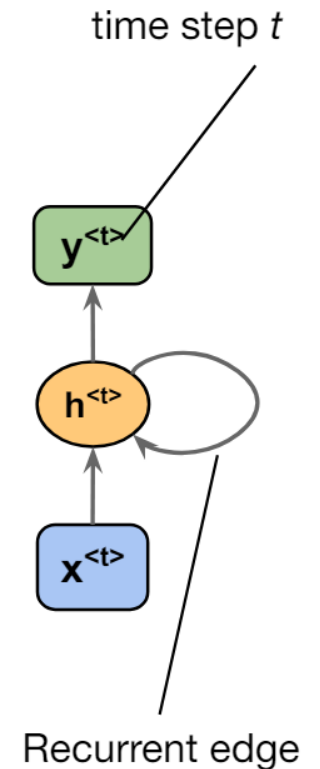
Recurrent Neural Network

- A variant of the conventional feed-forward artificial neural networks to deal with **sequential** data
- Hold the knowledge about the past (Have **memory**!)

Networks we used previously: also called feedforward neural networks



Recurrent Neural Network (RNN)



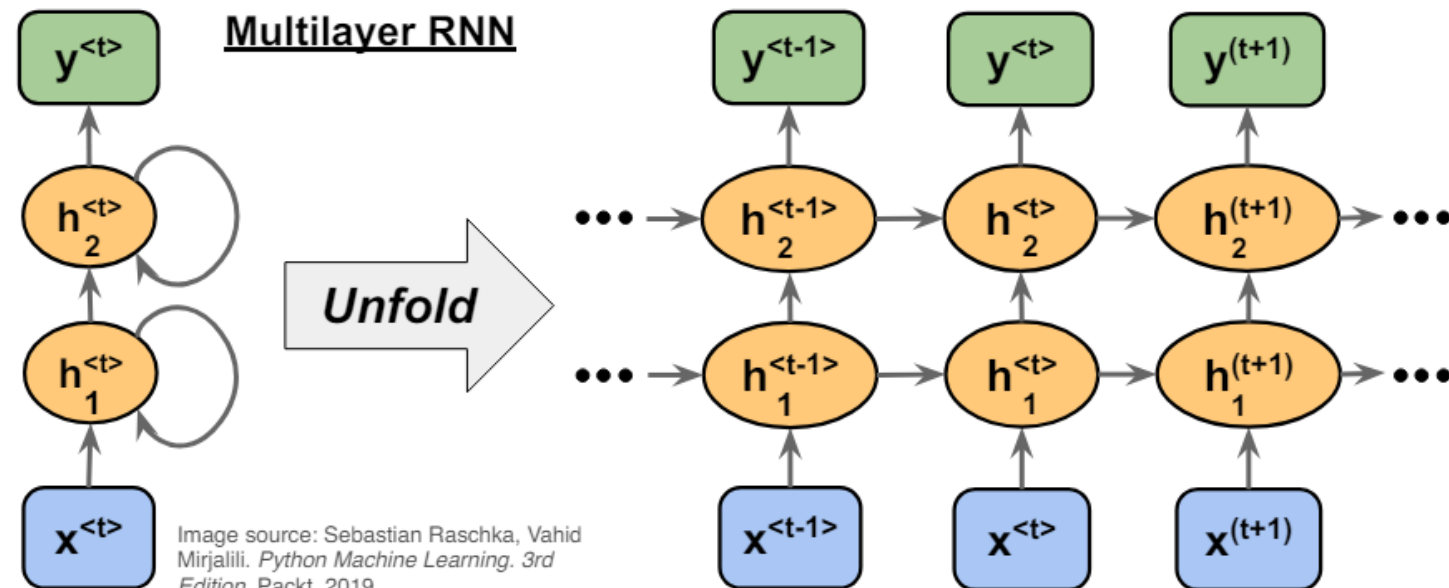
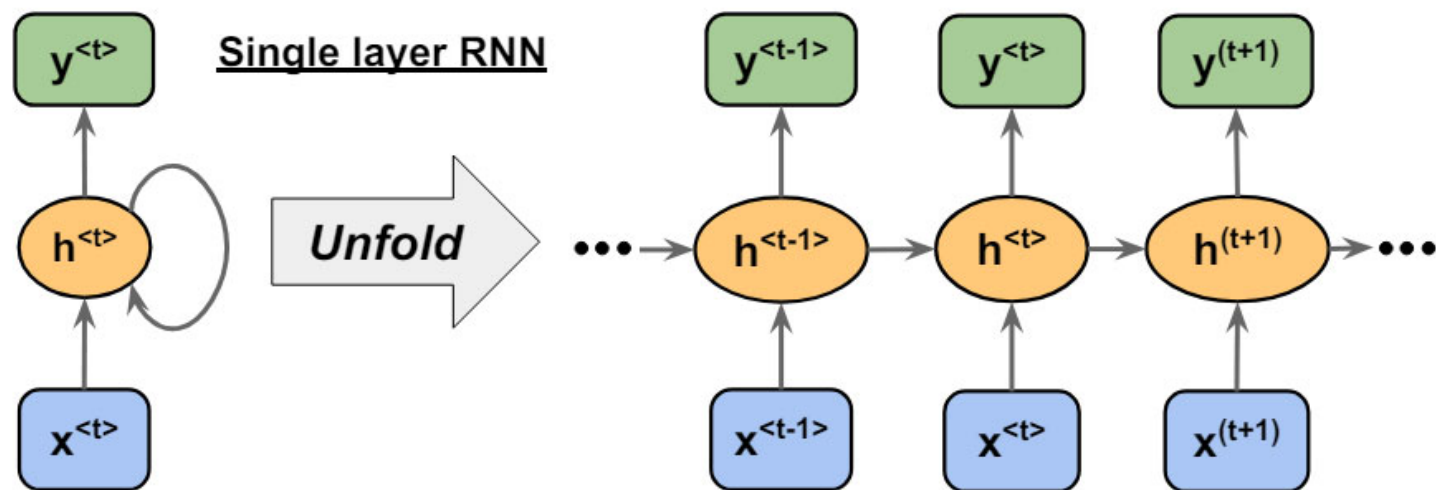
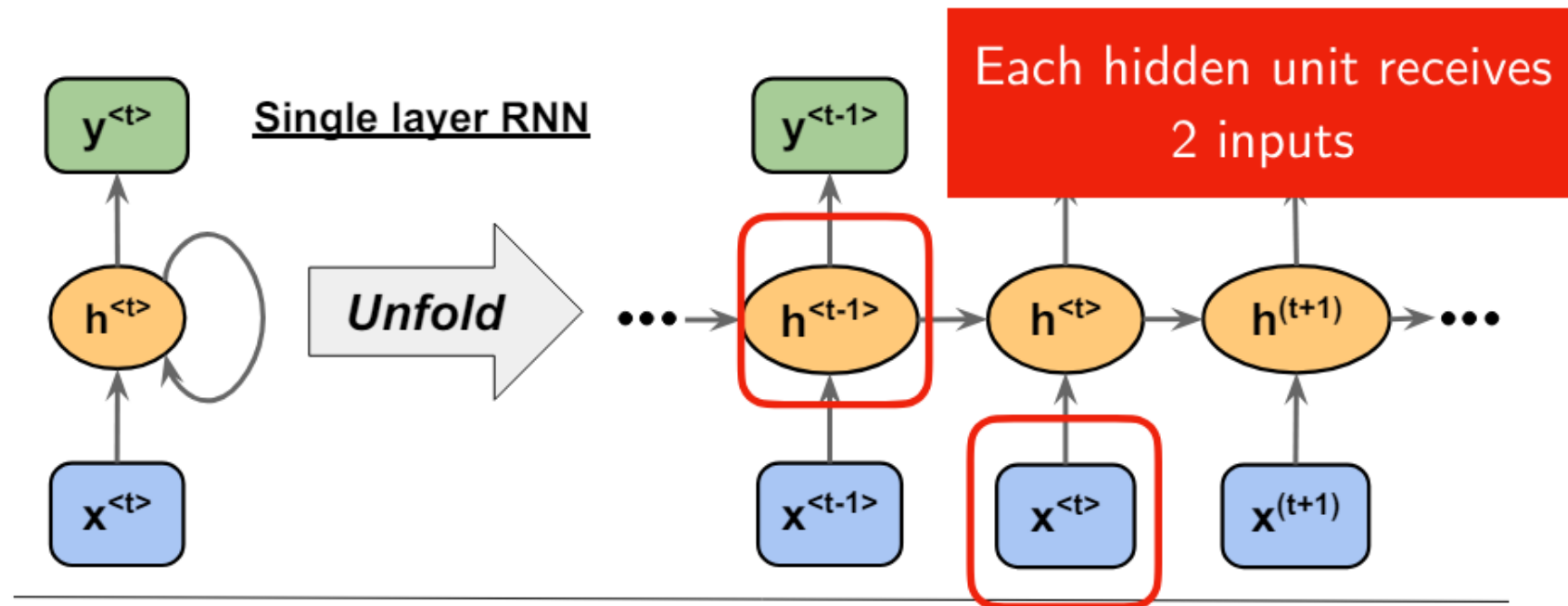


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Packt, 2019



Different type of sequence modeling tasks

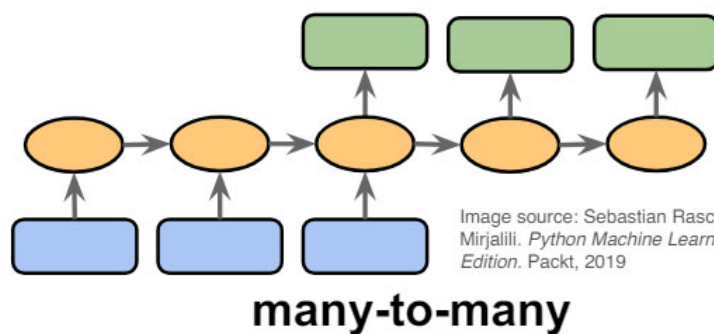
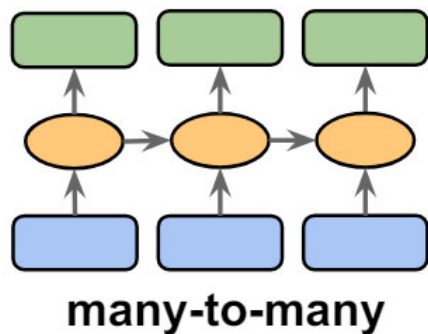
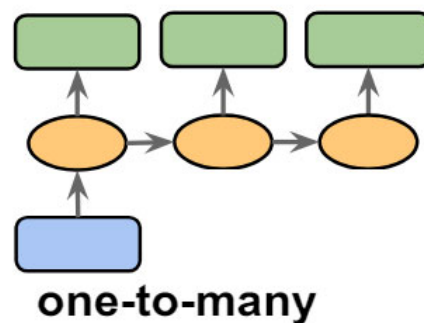
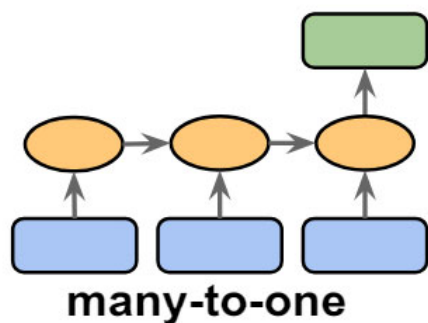
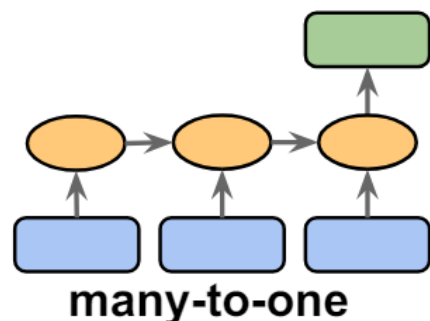


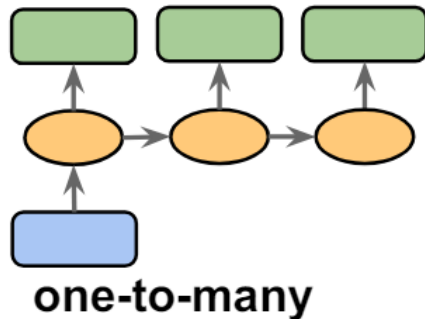
Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*, 3rd Edition. Packt, 2019

Different type of sequence modeling tasks



- Many-to-one: The input data is a sequence, but the output is a fixedsize vector, not a sequence.
- Ex.: sentiment analysis, the input is some text, and the output is a class label.

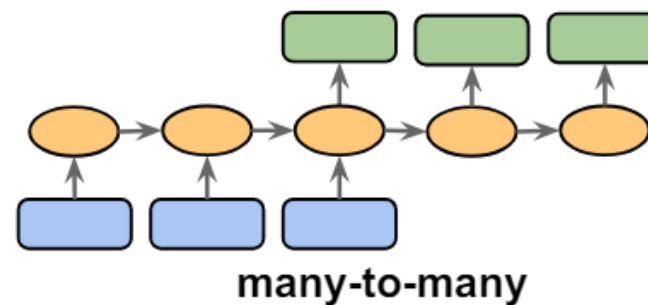
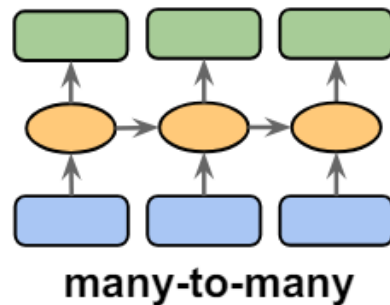
Different type of sequence modeling tasks



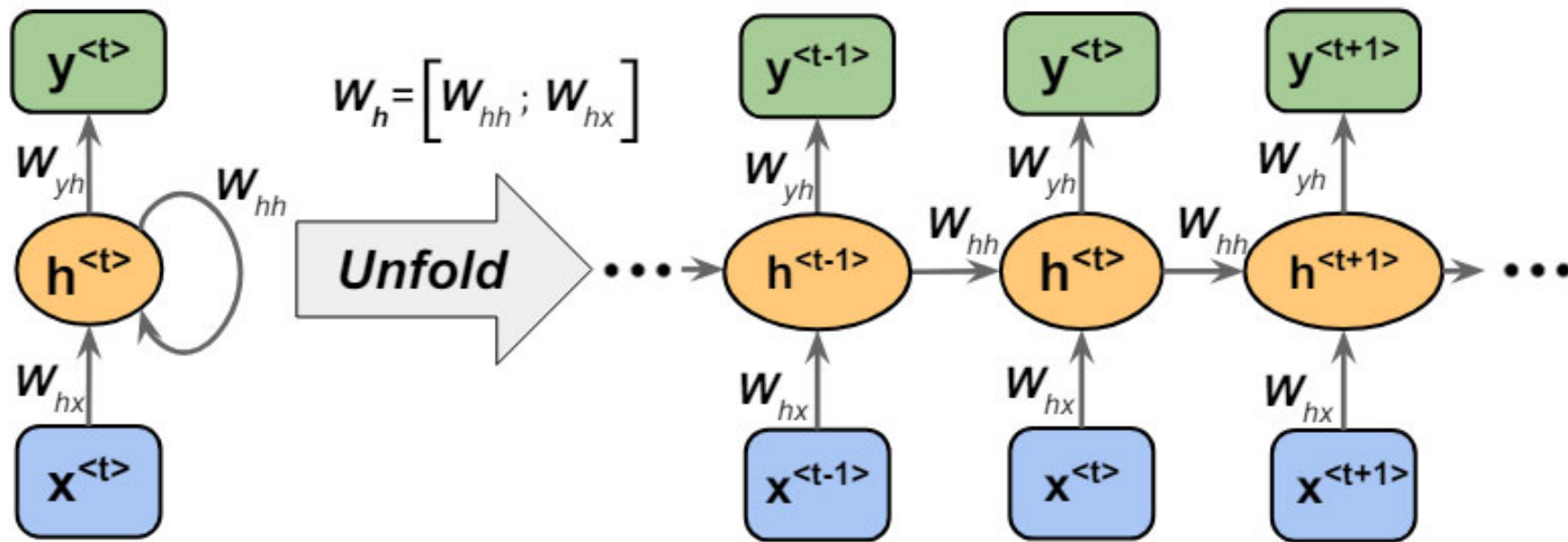
- One-to-many: Input data is in a standard format (not a sequence), the output is a sequence.
- Ex.: Image captioning, where the input is an image, the output is a text description of that image

Different type of sequence modeling tasks

- Many-to-many: Both inputs and outputs are sequences. Can be direct or delayed.
- Ex.: Video-captioning, i.e., describing a sequence of images via text (direct).
- Translating one language into another (delayed)



Weight matrices in a single-hidden layer RNN



Weight matrices in a single-hidden layer RNN

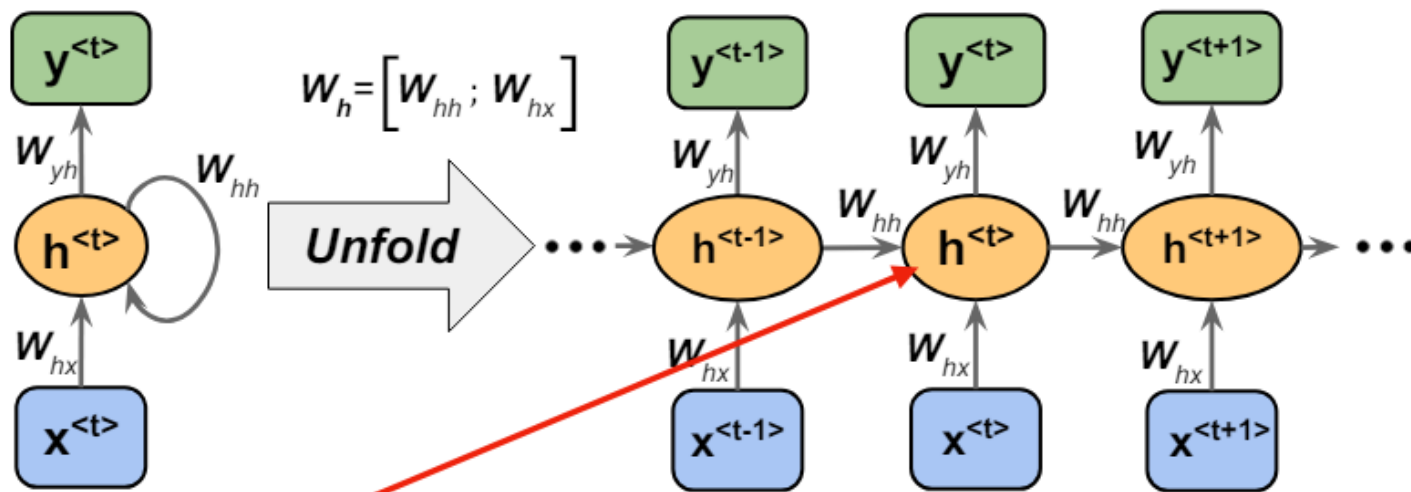


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Packt, 2019

Net input:

$$\mathbf{z}_h^{\langle t \rangle} = \mathbf{W}_{hx} \mathbf{x}^{\langle t \rangle} + \mathbf{W}_{hh} \mathbf{h}^{\langle t-1 \rangle} + \mathbf{b}_h$$

Activation:

$$\mathbf{h}^{\langle t \rangle} = \sigma_h(\mathbf{z}_h^{\langle t \rangle})$$

Weight matrices in a single-hidden layer RNN

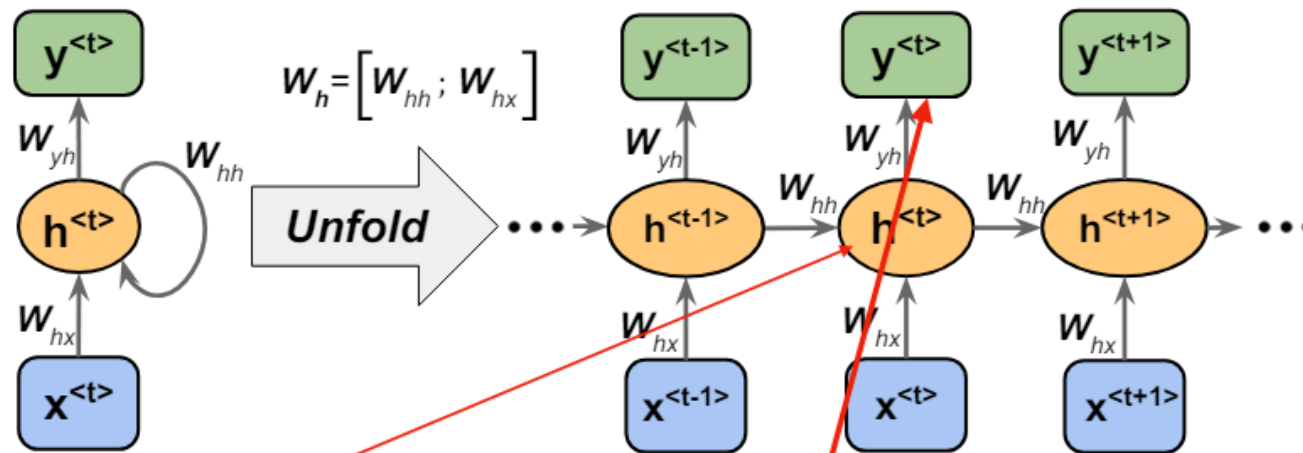


Image source: Sebastian Raschka, Vahid Mirjalili. *Python Machine Learning*. 3rd Edition. Packt, 2019

Net input:

$$\mathbf{z}_h^{(t)} = \mathbf{W}_{hx} \mathbf{x}^{(t)} + \mathbf{W}_{hh} \mathbf{h}^{(t-1)} + \mathbf{b}_h$$

Activation:

$$\mathbf{h}^{(t)} = \sigma_h(\mathbf{z}_h^{(t)})$$

Net input:

$$\mathbf{z}_y^{(t)} = \mathbf{W}_{yh} \mathbf{h}^{(t)} + \mathbf{b}_y$$

Output:

$$\mathbf{y}^{(t)} = \sigma_y(\mathbf{z}_y^{(t)})$$

RNN

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

input vector at
some time step

some function
with parameters W

The diagram shows the RNN recurrence formula $h_t = f_W(h_{t-1}, x_t)$. Each term is enclosed in a colored box: h_t is blue, f_W is purple, h_{t-1} is green, and x_t is red. Below the boxes are labels: 'new state' under h_t , 'old state' under h_{t-1} , and 'input vector at some time step' under x_t . A purple line connects the f_W box to the text 'some function with parameters W'.

Figure: RNN formula, [source](#)

- We can process a sequence of vectors x by applying a recurrence formula at every time step
- The same function and the same set of parameters are used at every time step.

RNN: forward pass

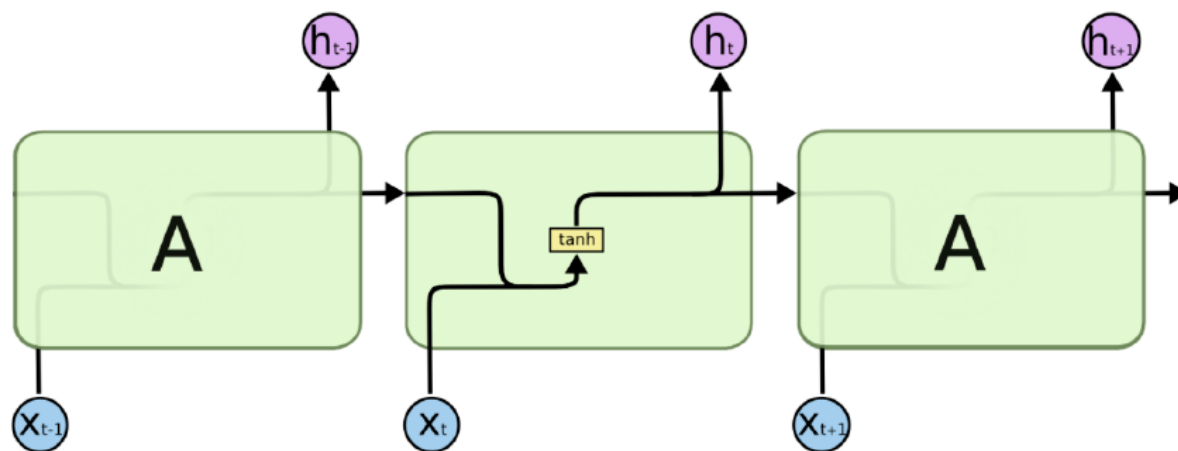
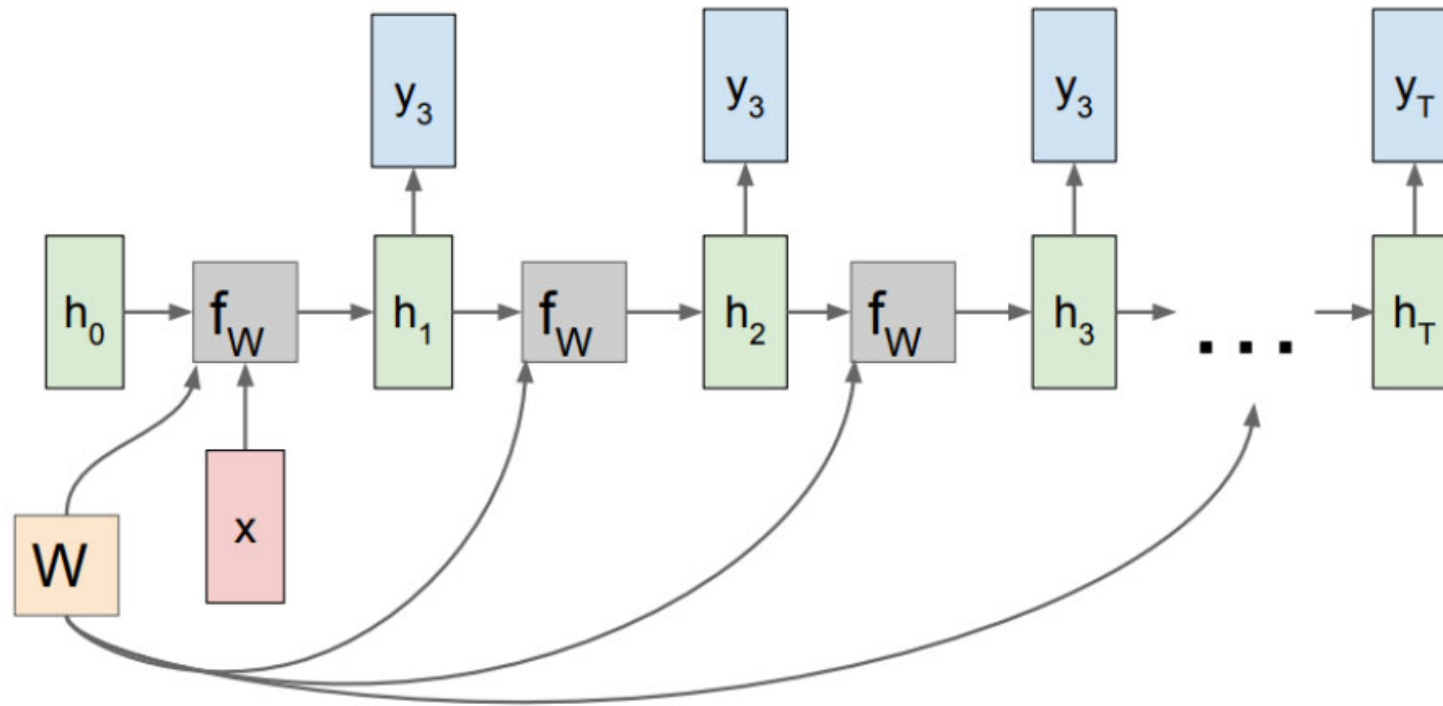


Figure: The repeating module in a standard RNN contains a single layer, [source](#)

$$\begin{aligned} h_t &= \tanh(W_{hh}h_{t-1} + W_{hx}x_t + b_h) \\ &= \tanh((W_{hh}W_{hx}) \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h) \\ &= \tanh(W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} + b_h) \end{aligned}$$

RNN: computational Graph



RNN: Backpropagation Through Time

The overall loss can be computed as the sum over all time steps

