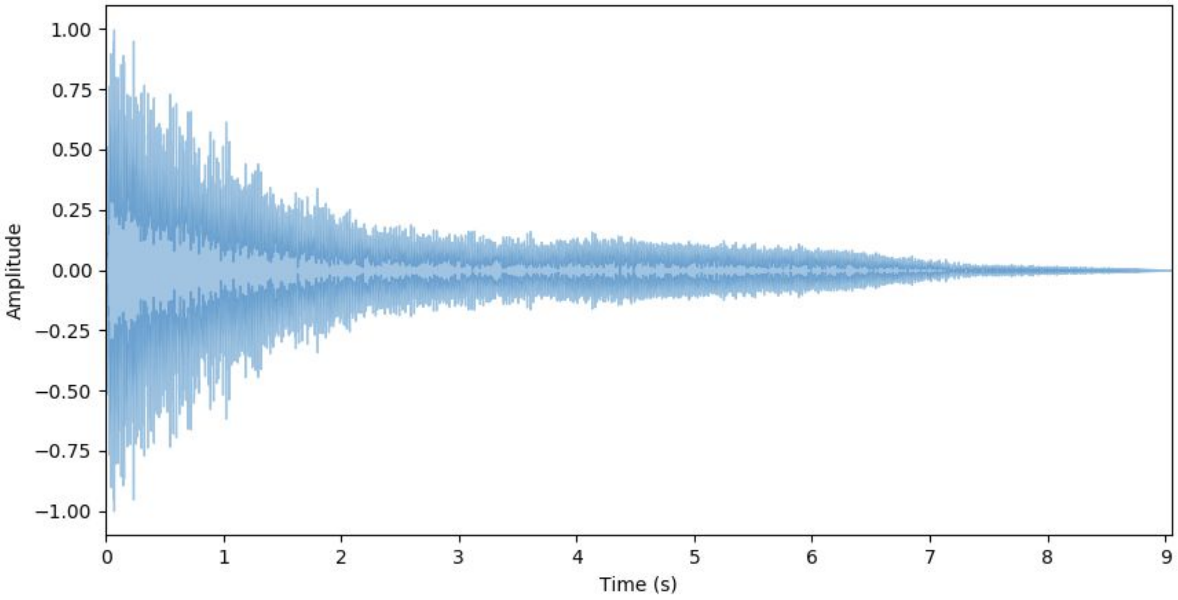
Recurrent Neural Networks explained easily

Valerio Velardo

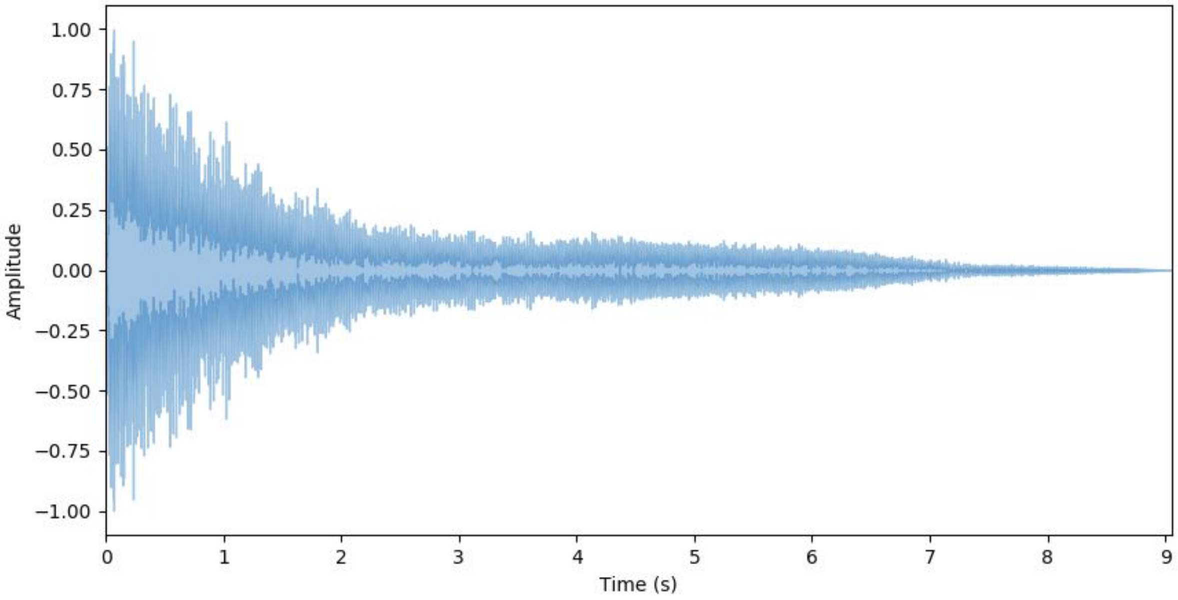
RNNs

- Order is important
- Variable length
- Used for sequential data
- Each item is processed in context
- Ideal for audio/music

Univariate time series

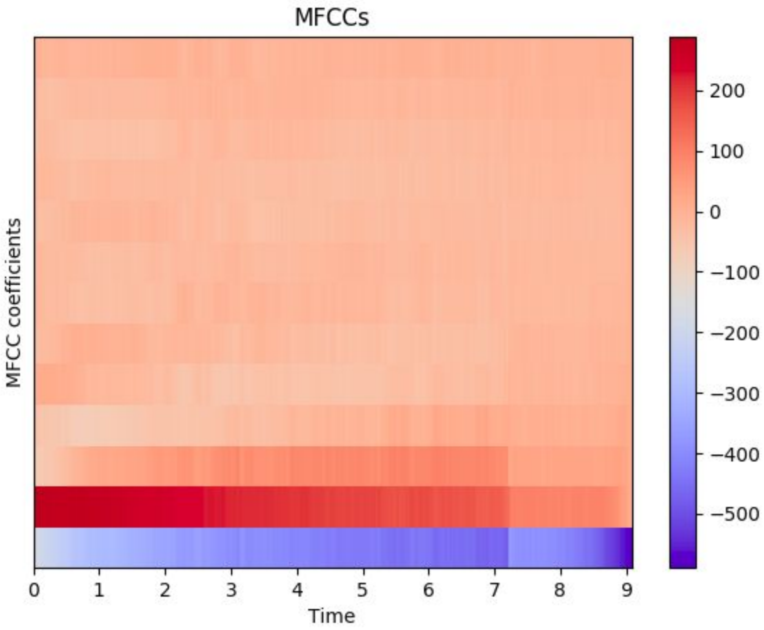


Univariate time series

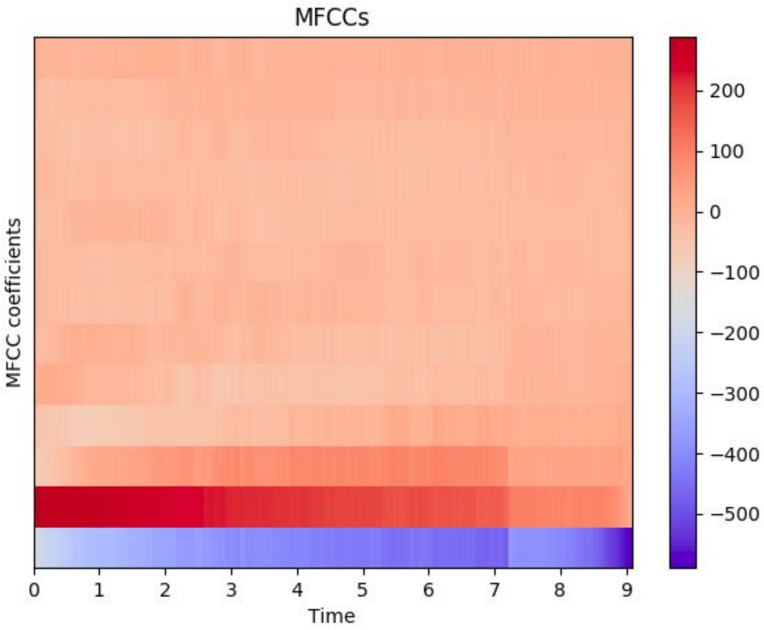


[22050x9, 1]

Multivariate time series



Multivariate time series

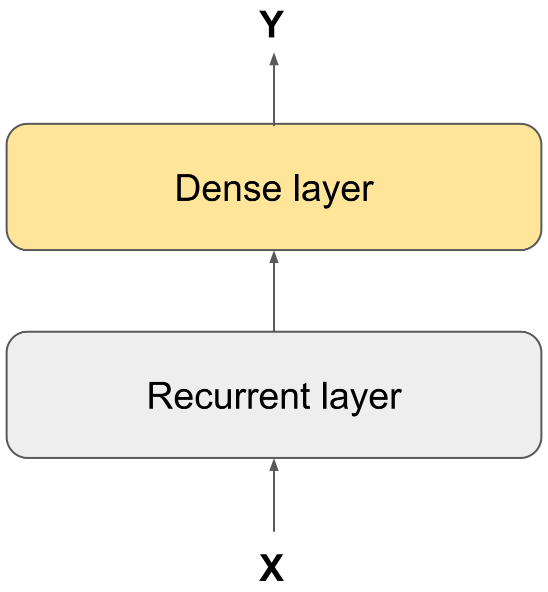


[sr/hop_length x 9, #MFCCs] = [387, 13]

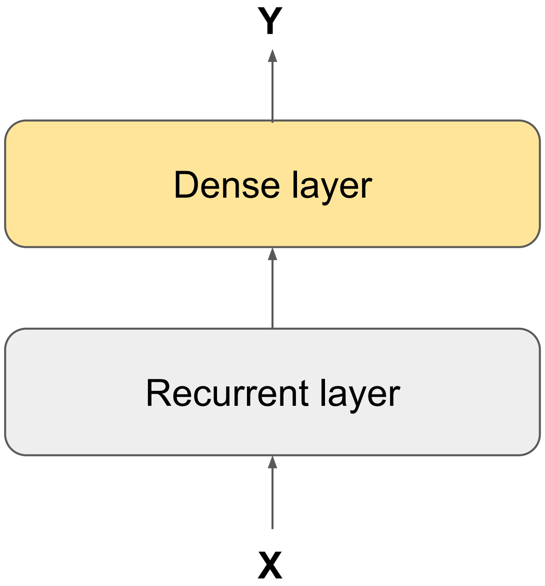
Intuition

- Input data points one at a time
- Predict next step
- Prediction depends on previous data points

RNN architecture

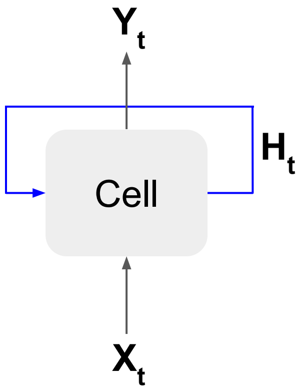


RNN architecture

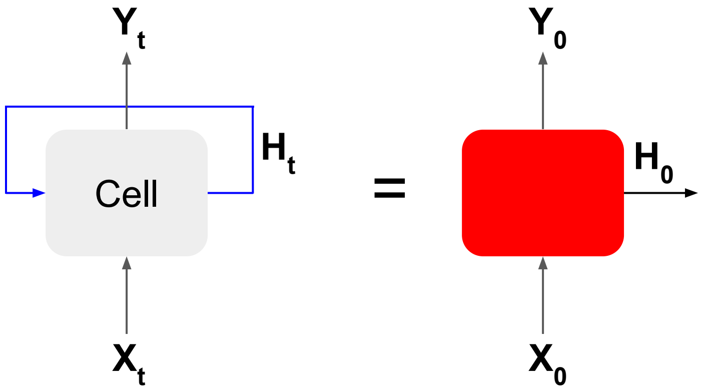


shape of **X** = [batch size, # steps, # dimensions]

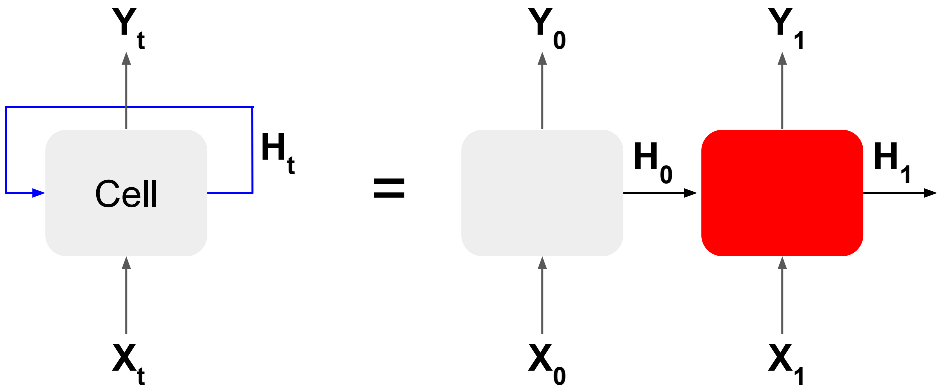
Recurrent layer



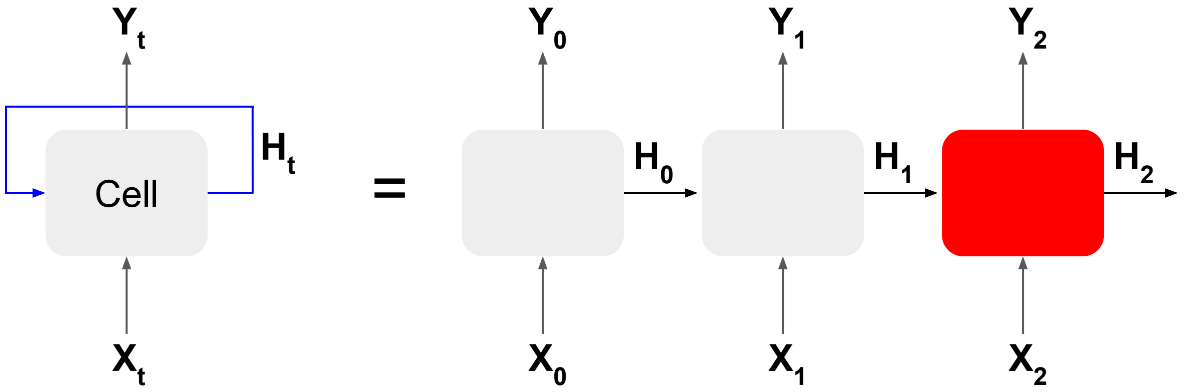
Unrolling a recurrent layer



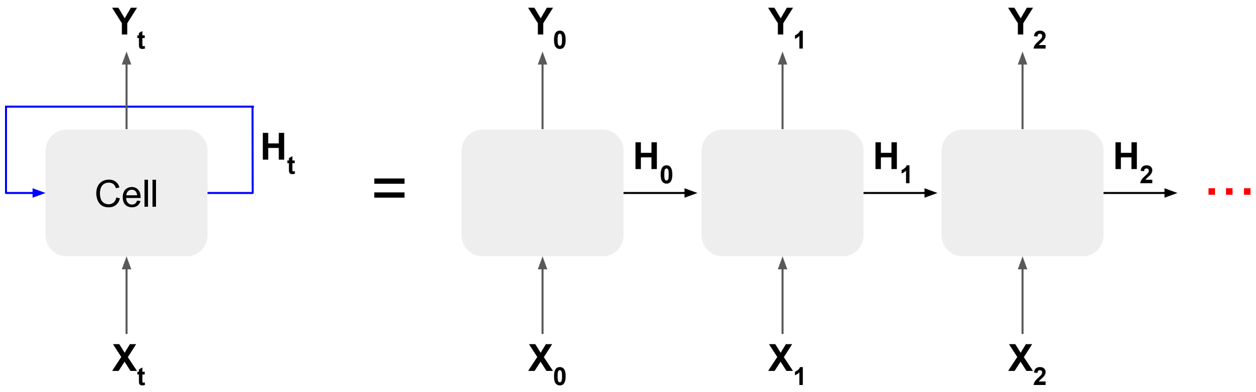
Unrolling a recurrent layer



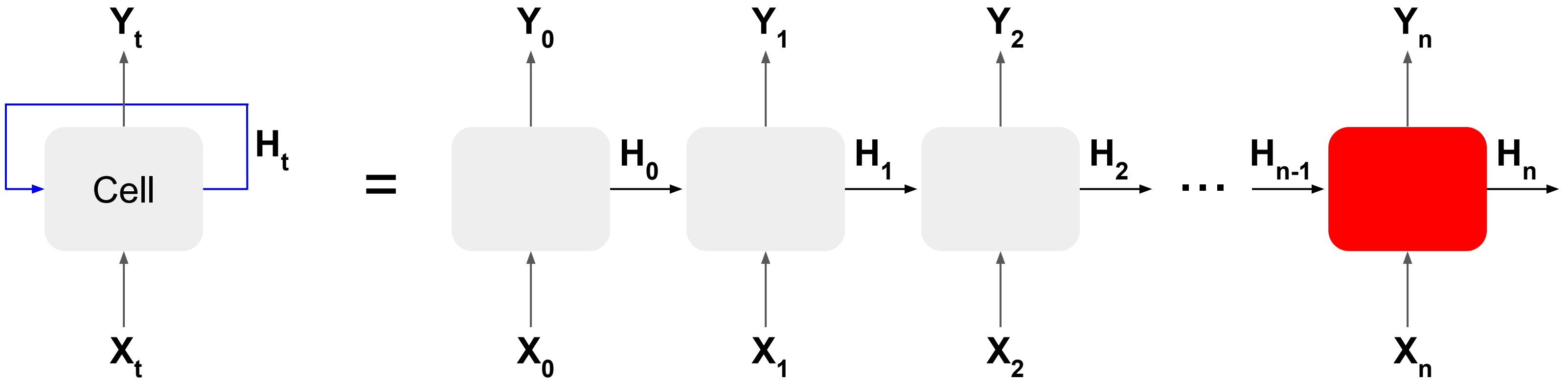
Unrolling a recurrent layer



Unrolling a recurrent layer



Unrolling a recurrent layer

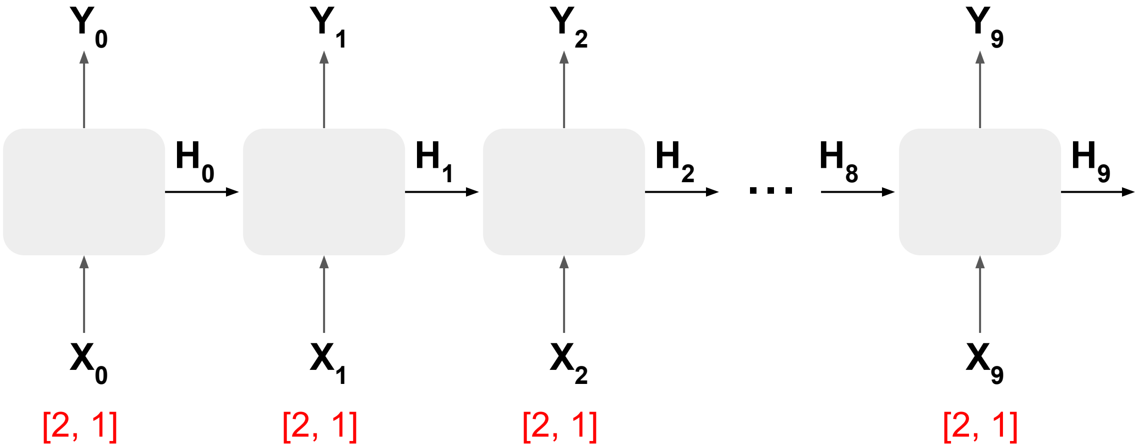


Data shape

[batch size, # steps, # dimensions] = [2, 9, 1]

Data shape

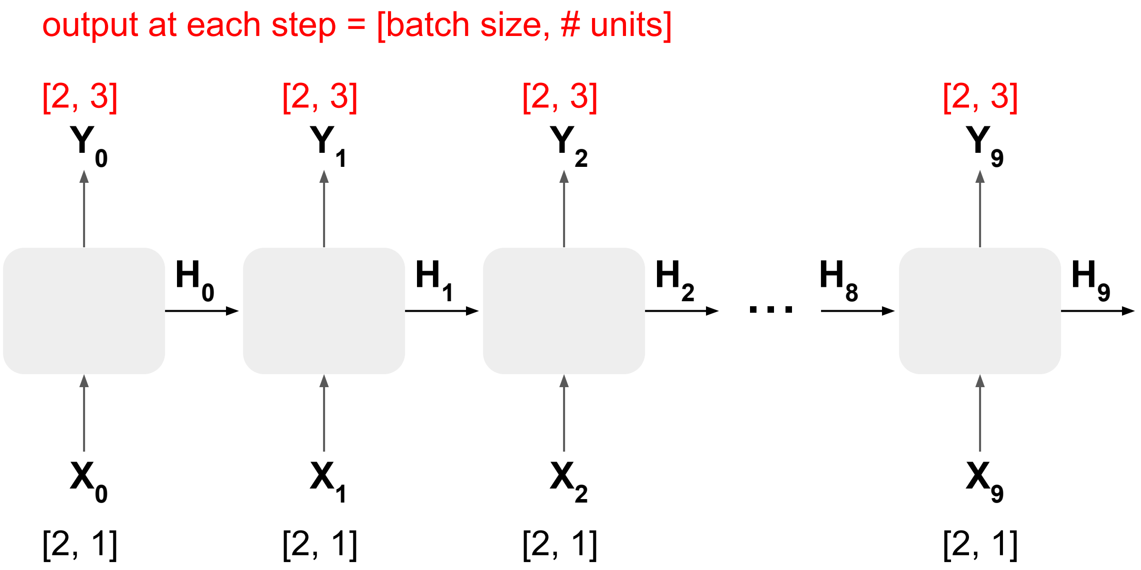
[batch size, # steps, # dimensions] = [2, 9, 1]



input at each step = [batch size, # dimensions]

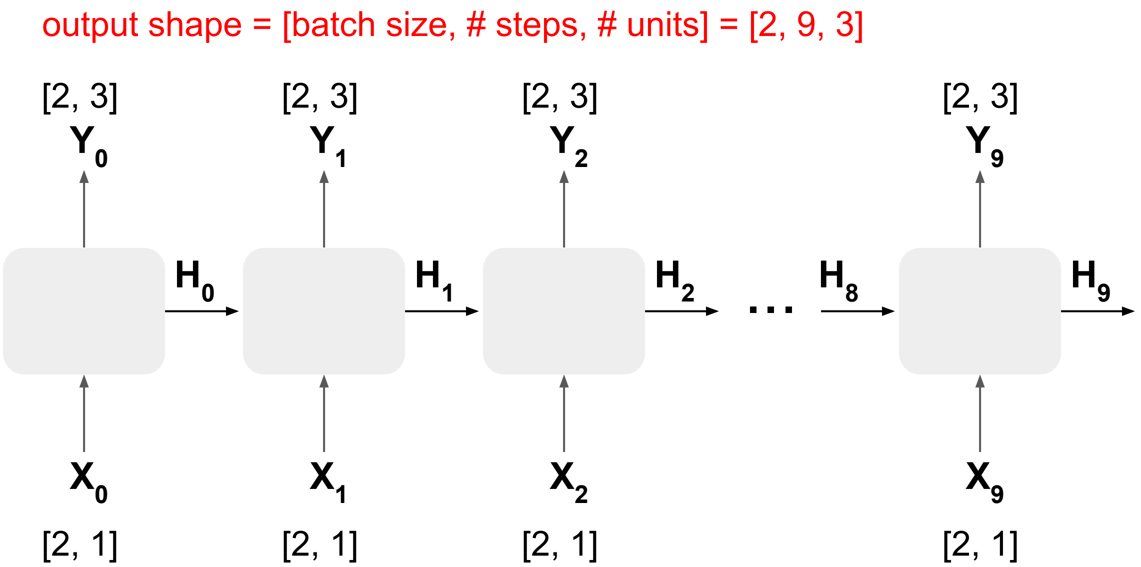
Data shape

[batch size, # steps, # dimensions] = [2, 9, 1]



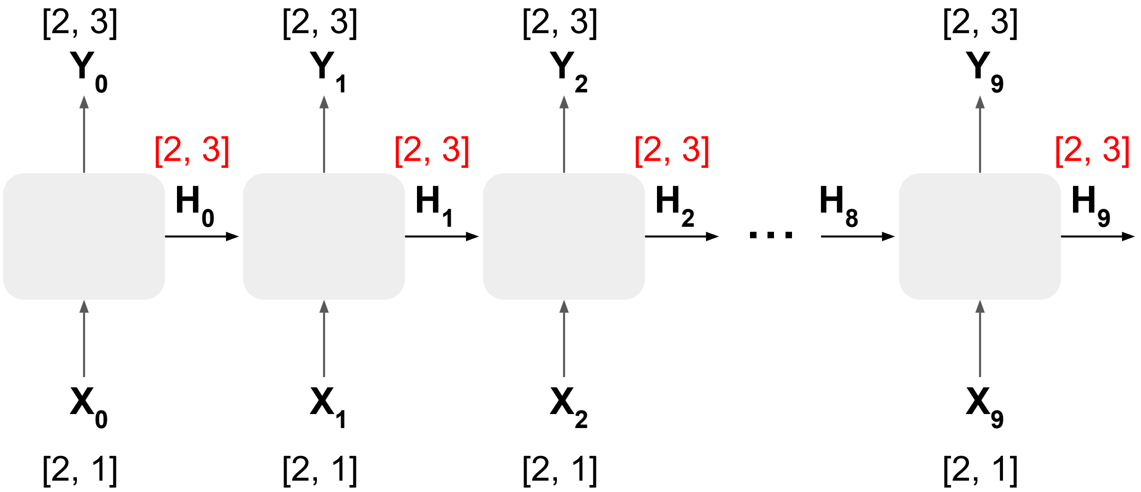
Data shape

[batch size, # steps, # dimensions] = [2, 9, 1]



Data shape

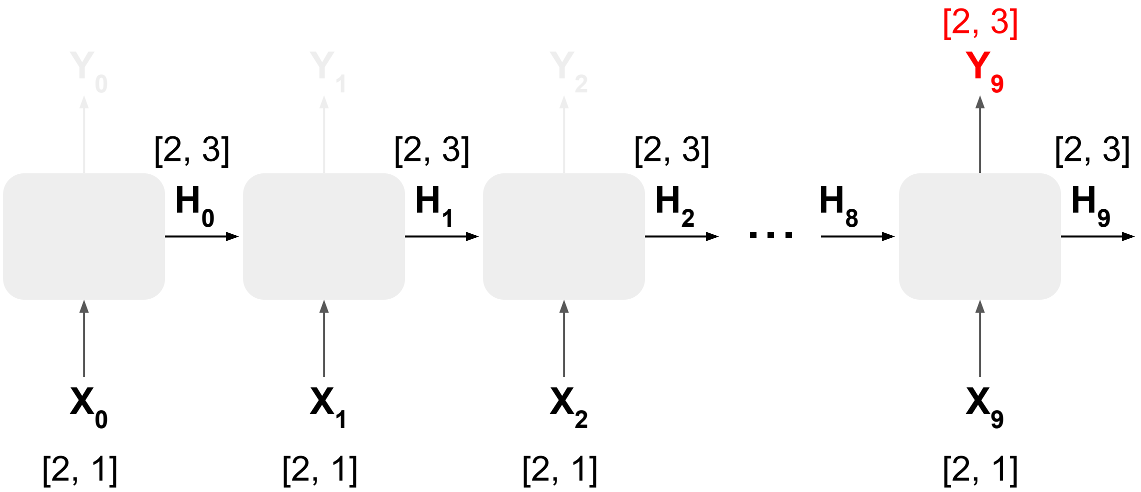
[batch size, # steps, # dimensions] = [2, 9, 1]



$$H_t = Y_t$$

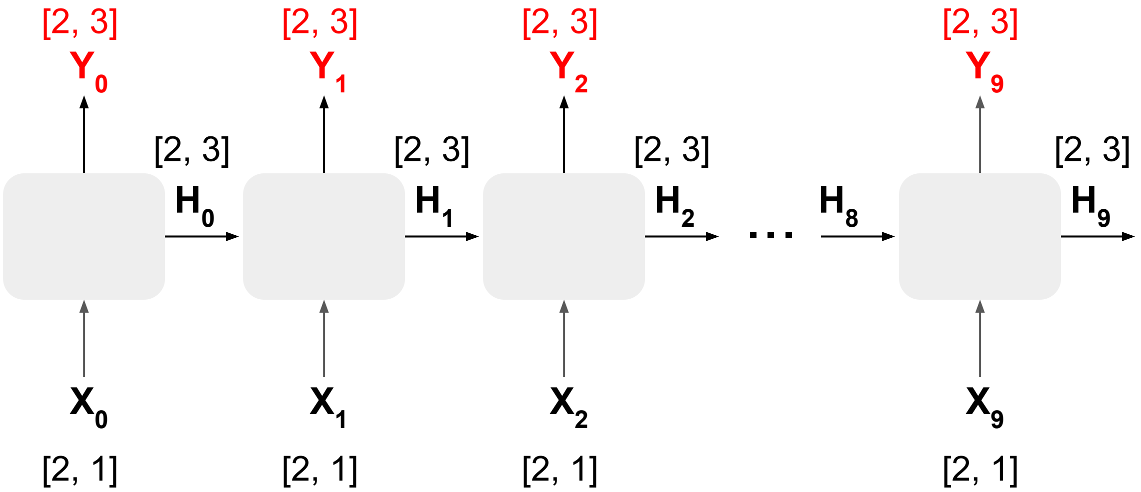
Sequence to vector RNN

[batch size, # steps, # dimensions] = [2, 9, 1]



Sequence to sequence RNN

[batch size, # steps, # dimensions] = [2, 9, 1]

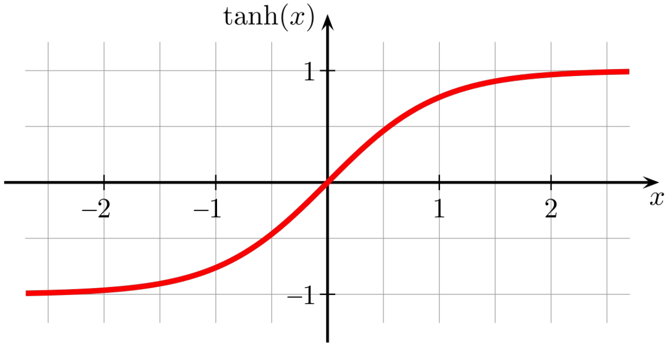


Memory cell for simple RNN

- Dense layer
- Input = state vector + input data
- Activation function = *tanh*

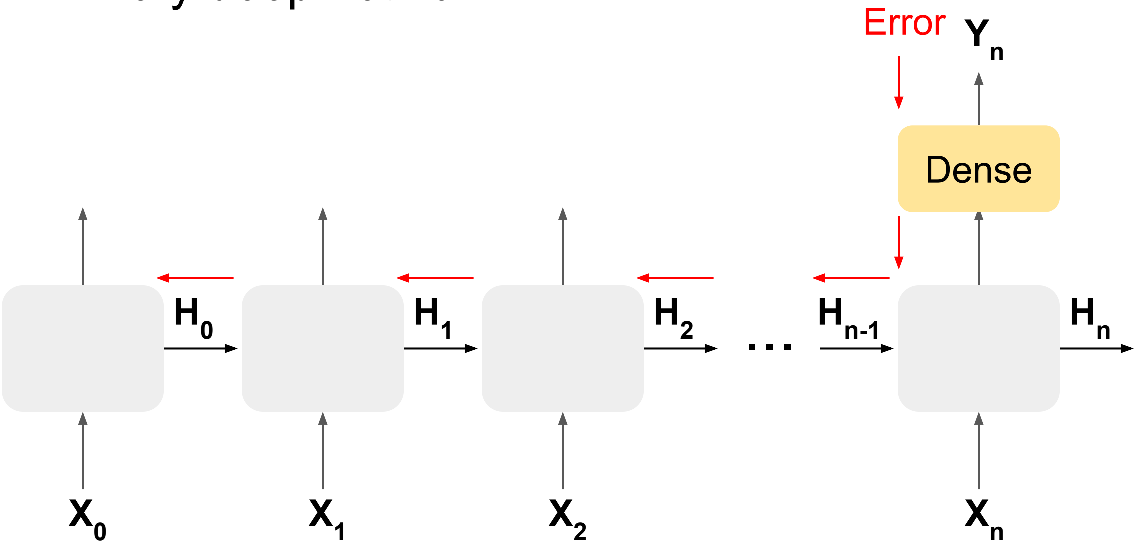
Why do we use *tanh*?

- Training RNNs is difficult
- Vanishing gradients + exploding gradients
- RELU can explode!
- *tanh* maintains values in [-1, 1]

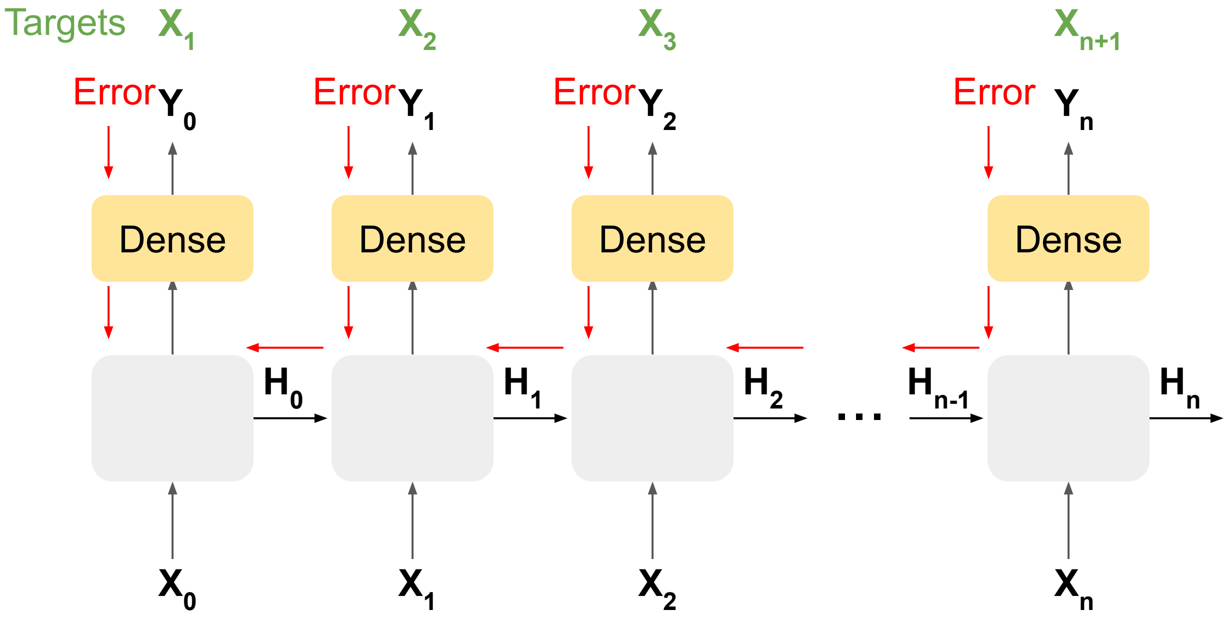


Backpropagation through time (BPTT)

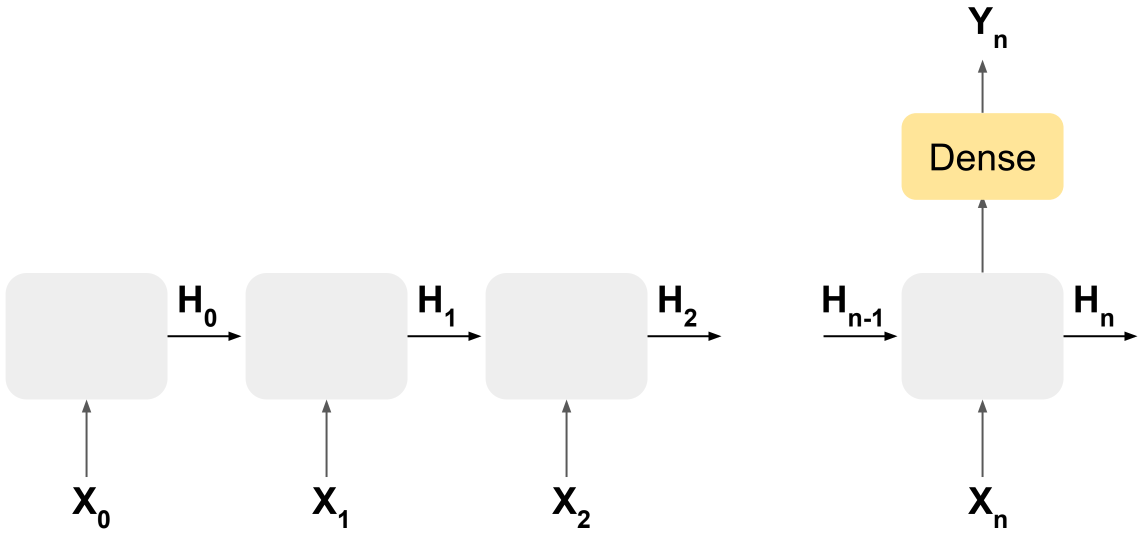
- Error is back propagated through time
- RNN is unrolled and treated as a feedforward network
- Very deep network!



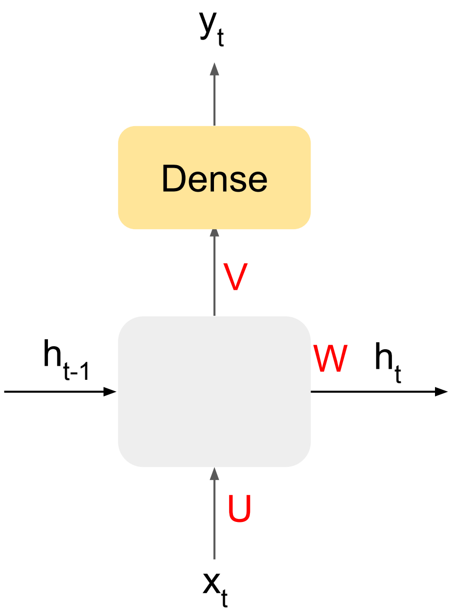
Backpropagation through time (BPTT)



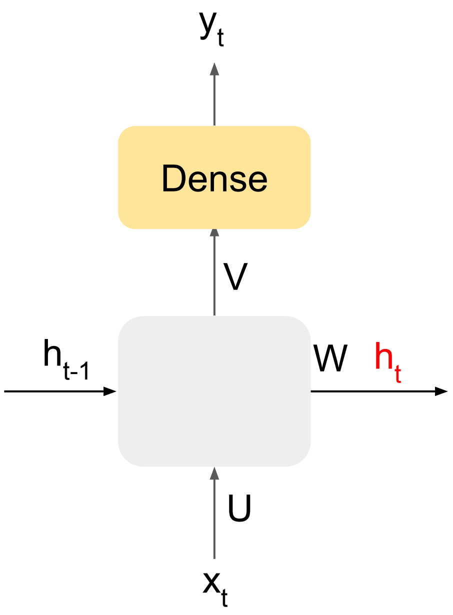
Backpropagation through time (BPTT)



The math behind

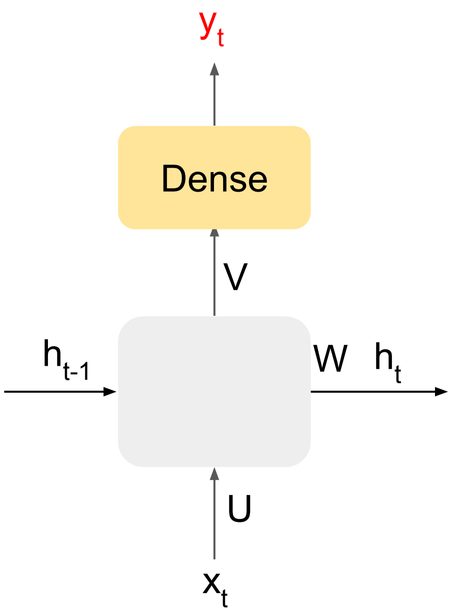


The math behind



$$\boxed{h_t = f(Ux_t + Wh_{t-1})}$$

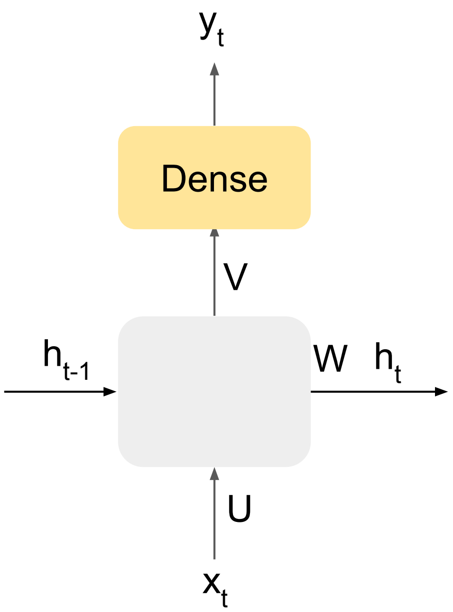
The math behind



$$h_t = f(Ux_t + Wh_{t-1})$$

$$\boxed{y_t = softmax(Vh_t)}$$

The math behind



$$h_t = f(\boxed{U}x_t + \boxed{W}h_{t-1})$$

$$y_t = softmax(\boxed{V}h_t)$$

Issues with simple RNNs

- No long-term memory
- Network can’t use info from the distant past
- Can’t learn patterns with long dependencies

What’s up next?

- Long Short Term Memory networks