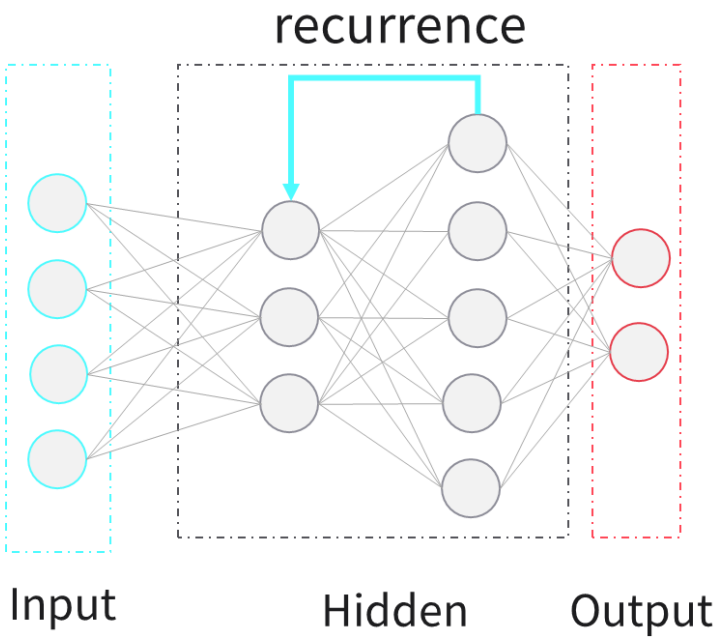
Recurrent
Neural Networks

Holberton

What are Recurrent Neural Networks?

Motivation

- What if patterns of data *change over time?*
- Internal *memory* as a distinctive feature
- A passage *through time*



Applications

- Natural language processing

- Autonomous driving

- Time series

"Predict" the near **future** based on **past** observations

Categories and applications

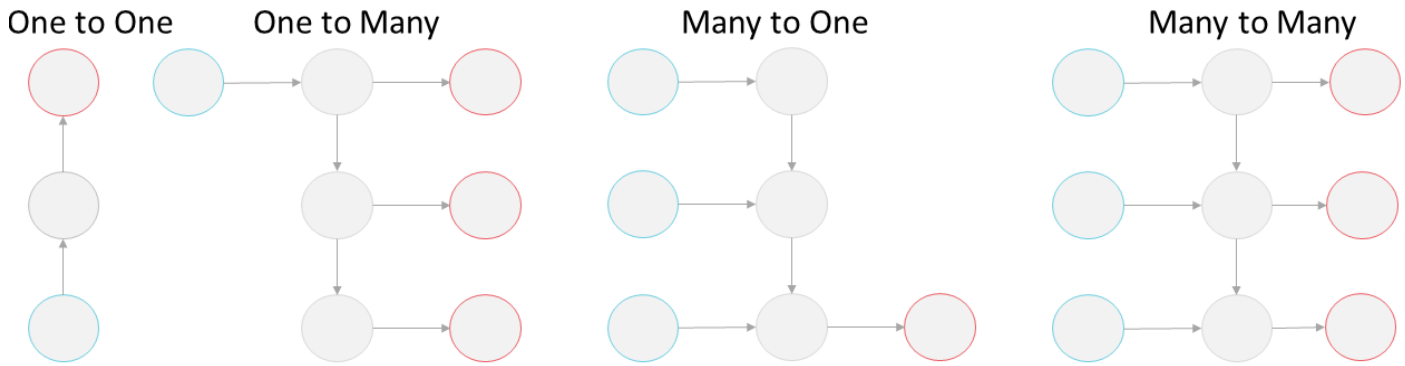


image captioning

music generation

text classification

automatic translation

Applications

- Natural language processing

- Autonomous driving

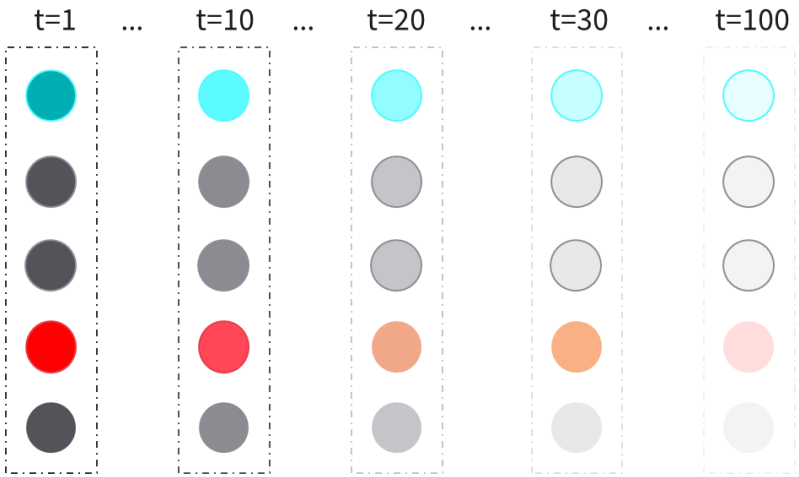
- Time series

"Predict" the near **future** based on **past** observations

Challenges of standard RNN

An old problem: backpropagation is used to train RNNs

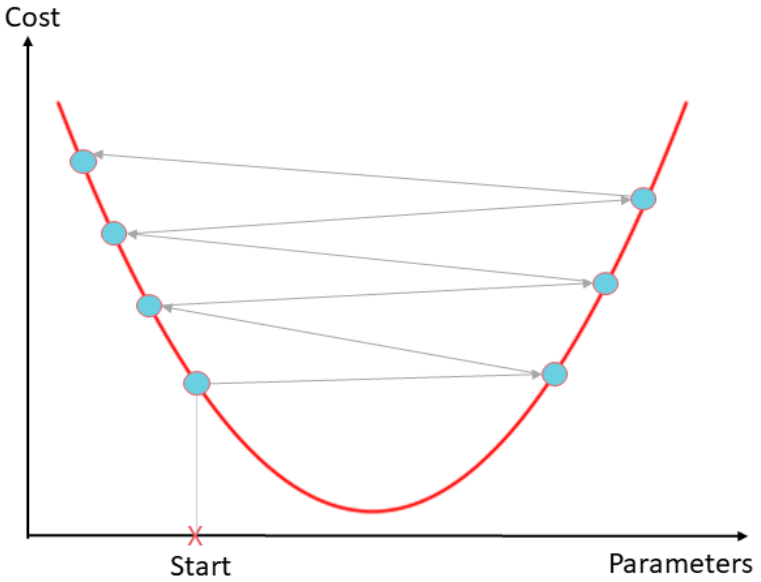
- It *suffers* from gradient issues
- *Decay of information* through time
- *Signal gets lost* as it travels in time
- Difficult for network to learn



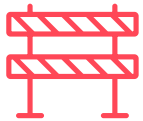
Challenges of standard RNN

Another old problem: exploding gradients

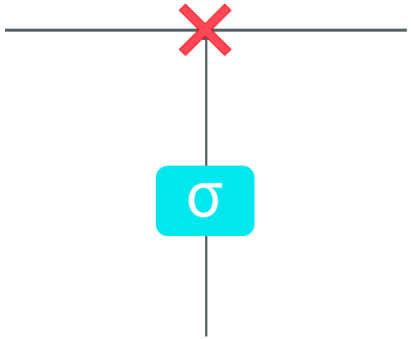
- Unreasonably high weights
- Large gradient *errors accumulate*
- Network gets *unstable*



Addressing the challenges

Solution: **gated** units 

- Long Short-Term Memory Unit (LSTM)
- Gated Recurrent Unit (GRU)
- Capable of handling long-term dependencies
- Carry information across time-steps

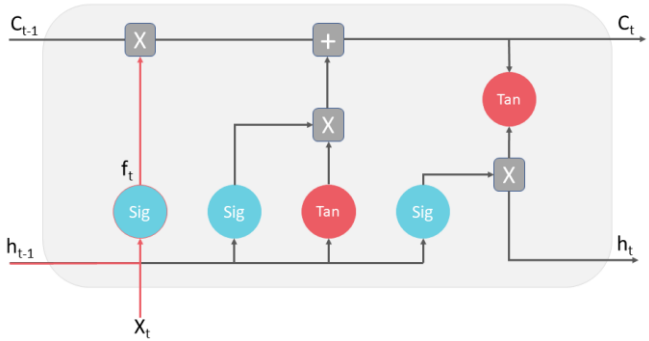


Information is **added** or **removed** through structures called gates

The role of gates in LSTM

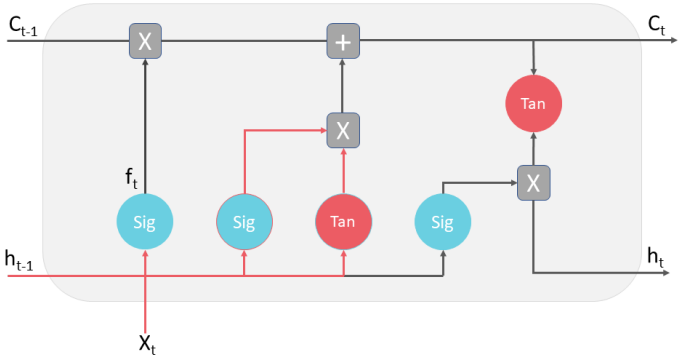
The **forget gate**

- decide what to keep to keep from prior cell state
- consider current input and previous hidden state



The **input gate**

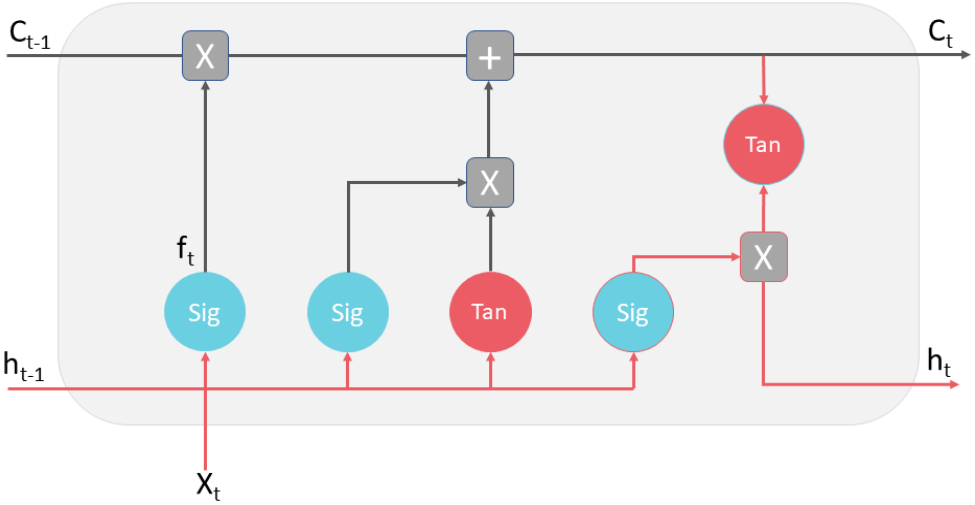
- decide what is relevant to update in current cell
- network calculates cell state



The role of gates in LSTM

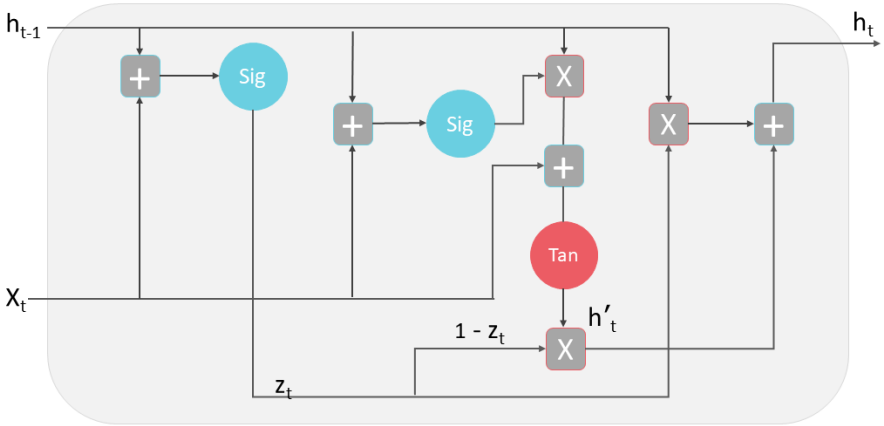
The **output** gate

- decide what to output from memory cell
- Current state becomes input for next unit



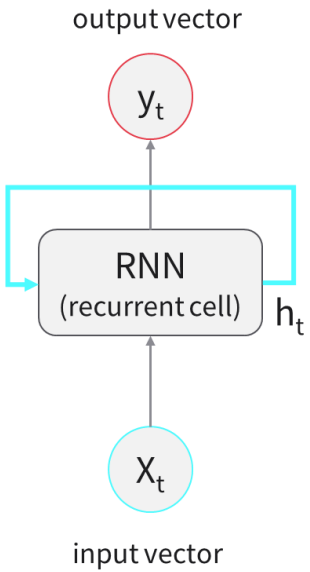
Gated recurrent units

- decide what to output from memory cell
- Current state becomes input for next unit
- **Update** gate decides what information to pass
- **Reset** gate decides how much to forget



Training RNNs

- Apply **recurrence** to each timestep

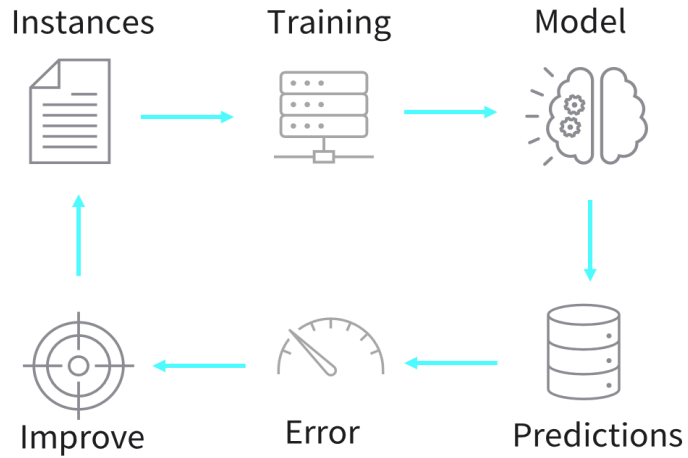


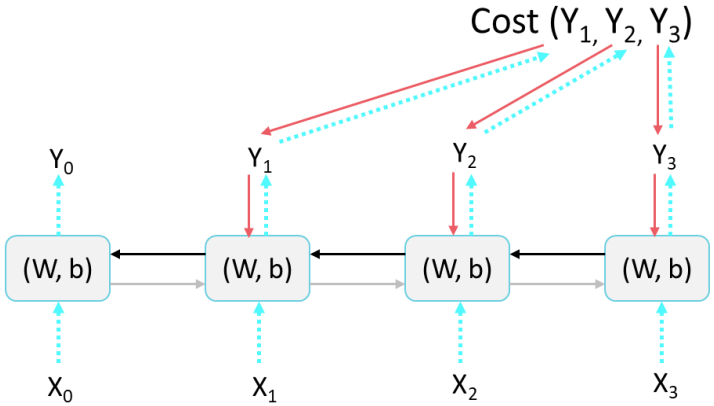
$$h_t = f_W(h_{t-1}, x_t)$$

cell state | function parametrized by W | previous state | input vector

Training RNNs

- Backpropagation **through time**
- Forward pass: process information & generate input for next timestep
- Backward pass: compute gradient of classification loss & adjust





Any questions?