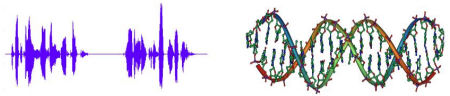
Idea

Use temporal covariations/context to refine prediction at some time t

Recurrent Neural networks

Sequence learning is the study of machine learning algorithms designed for sequential data



Sequential data: data points come in order and successive points may be **dependent**, e.g.,

- Letters in a word
- Words in a sentence/document
- Phonemes in a spoken word utterance
- Time series
- Frames in a video, etc.

So far, we assume that data points in a dataset are i.i.d (independent and identically distributed)

Does not hold in many applications

Recurrent Neural networks

- Plain CNNs are not typically good at length-varying input and output.
- Difficult to define input and output
  - Remember that
    - Input image is a 3D tensor (width, length, color channels)
    - Output is a fixed number of classes.
  - Sequence could be:
    - "I know that you know that I know "
    - Or "I don't know"
- Input and output are strongly correlated within the sequence.
- (Still, people figured out ways to use CNN on sequence learning)

Time series analysis

1. **Autoregressive models**

- Predict the next term in a linear sequence from a **fixed number of previous terms** using delay taps.

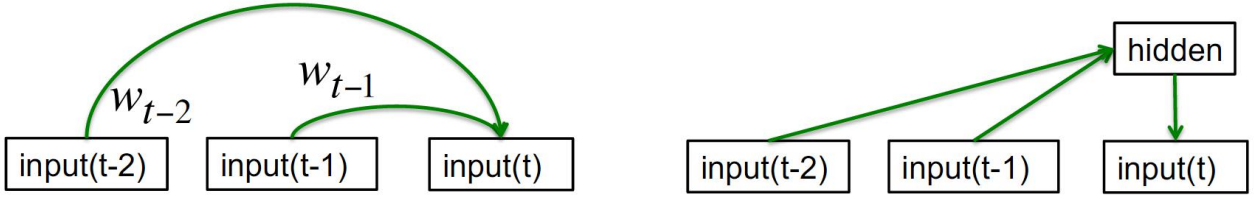
The autoregressive model specifies that the output variable depends linearly on its own previous values

AR(p) $Y_t = \beta_0 + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + \ldots + \beta_p Y_{t-p} + e_t$

Sequence labeling

2. **Feed-forward neural nets**

- These **generalize autoregressive models** by using one or more layers of non-linear hidden units



Memoryless models: limited word-memory window; hidden state cannot be used efficiently.

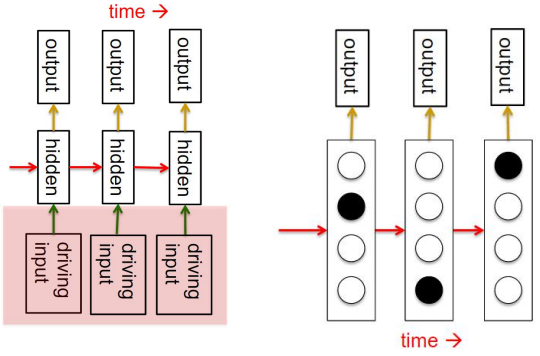
Time series analysis

3**. Linear Dynamical Systems**

- Generative models. They have a hidden state that cannot be observed directly.

4. **Hidden Markov Models**

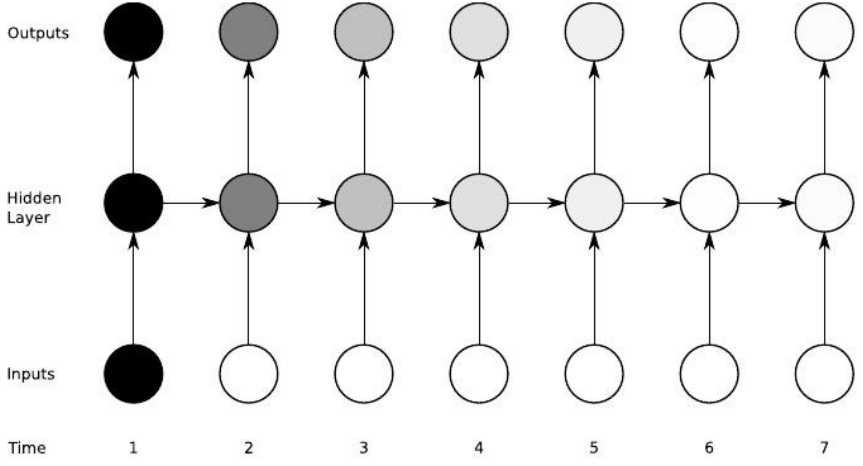
- Have a **discrete one-of-N hidden state**. Transitions between states are stochastic and controlled by a transition matrix. The outputs produced by a state are stochastic.



Memoryful models,
time-cost to infer the hidden state distribution.

Recurrent Neural Networks

- Update the hidden state in a deterministic nonlinear way.
- In a simple speaking case, we send the chosen word back to the network as input.

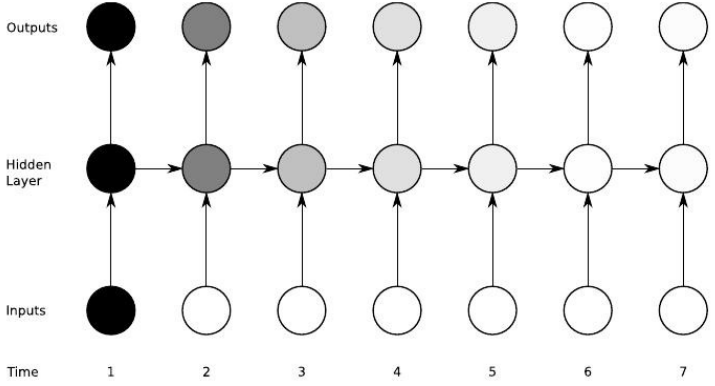


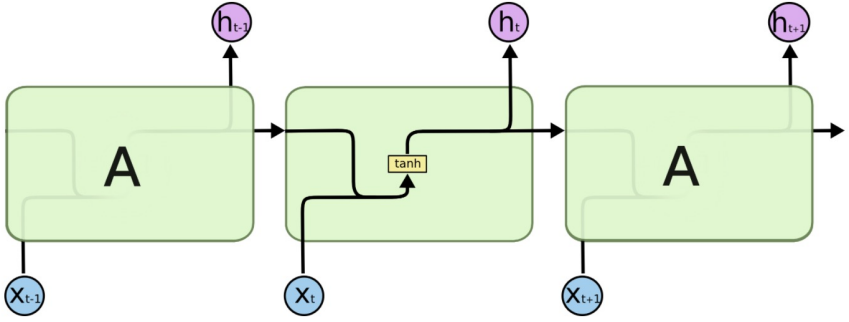
Vanilla forward path RNN

Recurrent Neural Networks

The forward pass of a vanilla RNN

1. The same as that of a perceptron with a single hidden layer
2. Except that activations arrive at the (single) hidden layer from both the current external input and the hidden layer activations one step back in time.



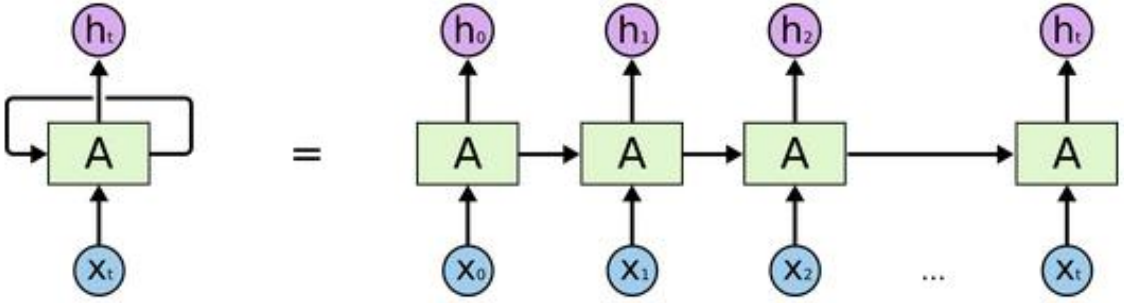


The repeating module in a standard RNN contains a single layer.

Vanilla backward path RNN

Recurrent Neural Networks

- Back-propagation through time
- It's just the standard back-propagation. Many times.

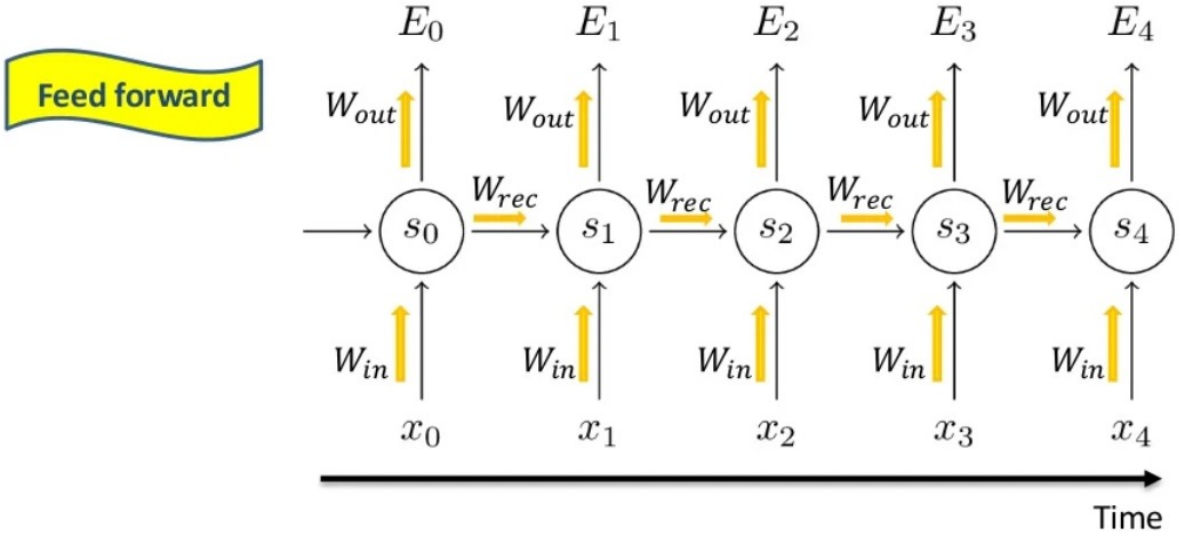


An unrolled recurrent neural network.

Vanilla forward path RNN

Recurrent Neural Networks

- Like standard back-propagation, RNN consists of a repeated application of the chain rule.

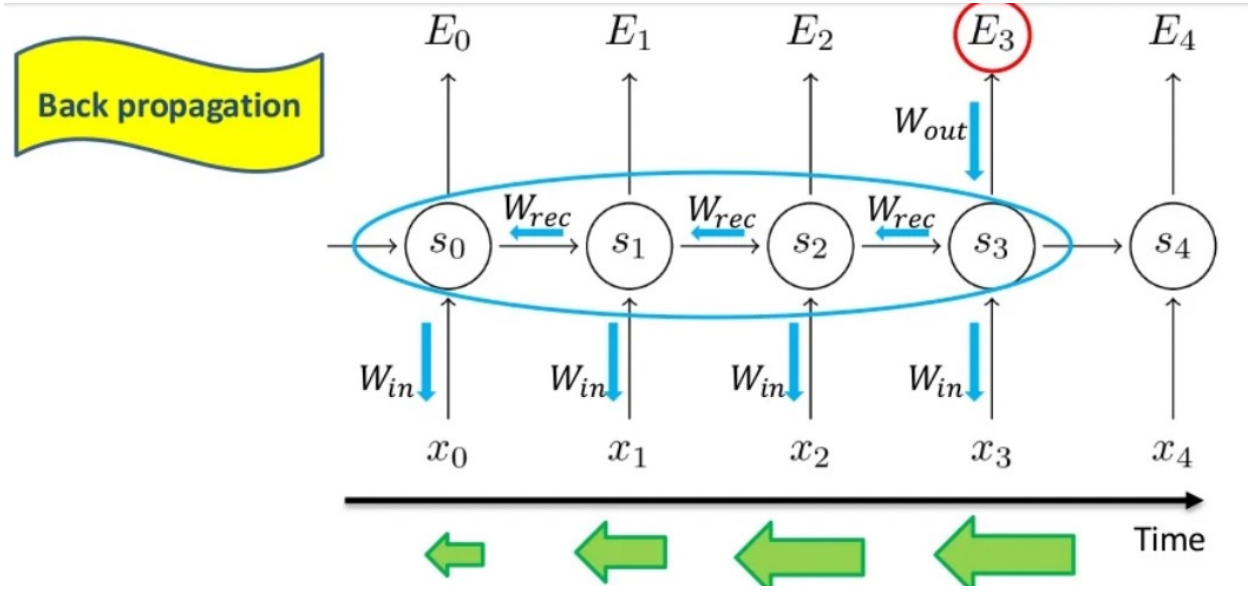


Vanilla forward path RNN

Recurrent Neural Networks

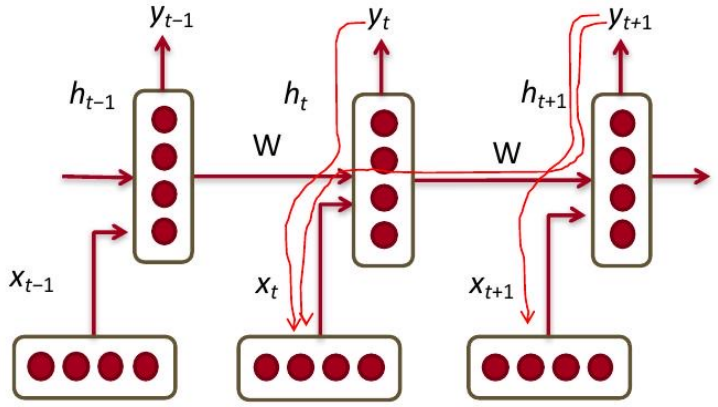
•Like standard back-propagation, RNN consists of a repeated application of the chain rule.

→ **Vanishing gradients**



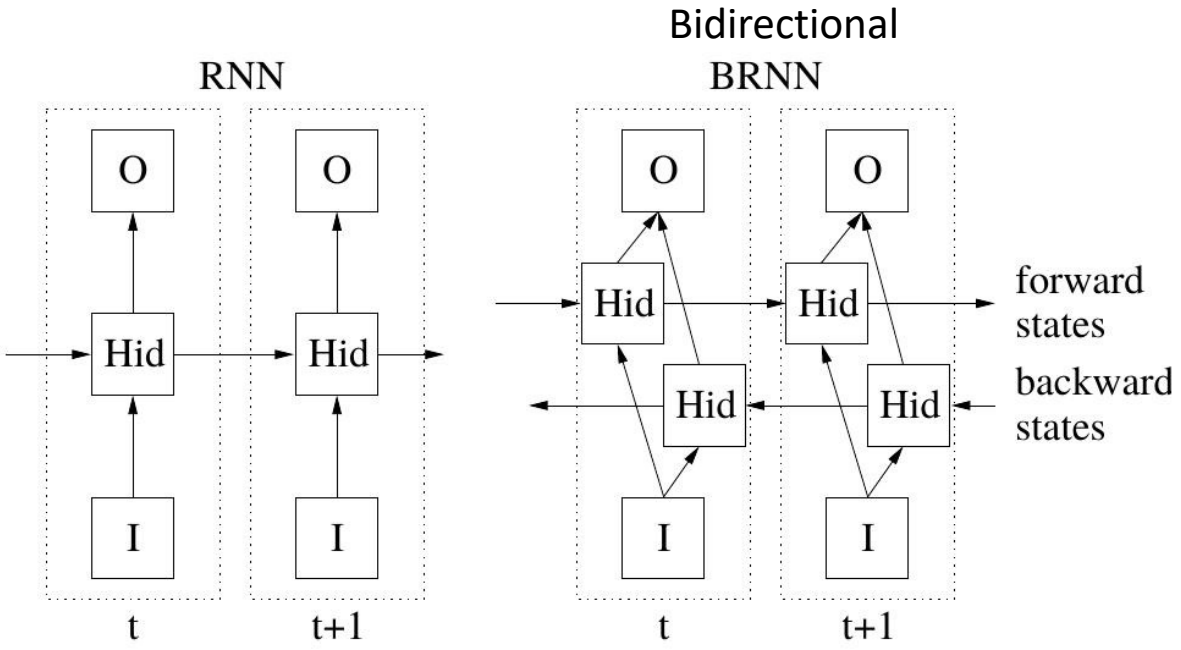
Problem: Vanishing and exploding gradients

•Multiply the same matrix at each time step during back- prop



Vanilla bidirectional path RNN

•For many time series analysis, we might have access to future.



LSTM: long short term memory

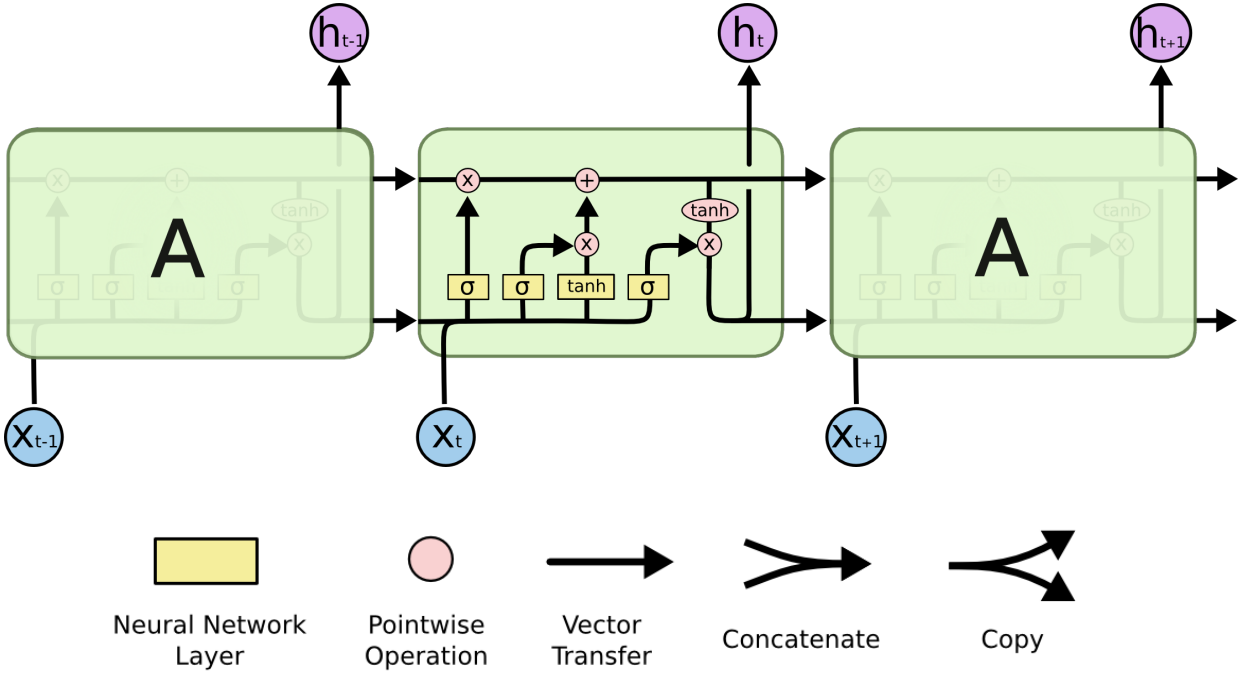
- The problem is that the influence of a given input on the hidden layer, and therefore on the network output, either decays or blows up exponentially as it cycles around the network's recurrent connections.
- The **most effective solution so far is the Long Short Term Memory** (LSTM) architecture (Hochreiter and Schmidhuber, 1997).
- The LSTM architecture consists of a set of recurrently connected subnets, known as **memory blocks**. These blocks can be thought of as a differentiable version of the memory chips in a digital computer.

  Write, read and reset operations for the cells

- The **input, output and forget gates**.

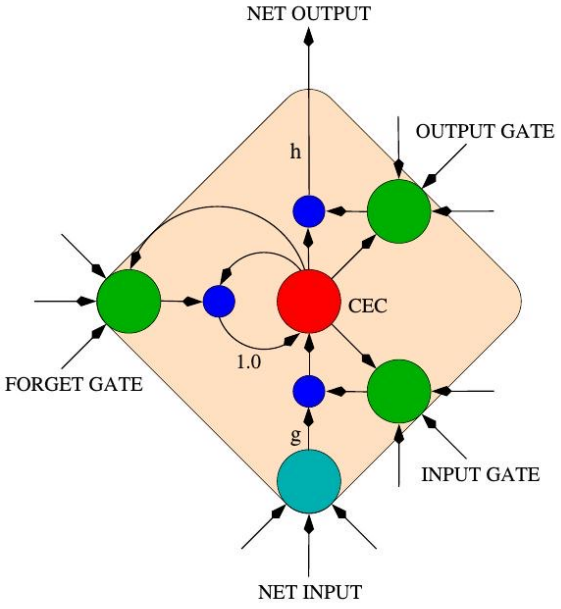
LSTM: long short term memory

- Structure of an LSTM, contains four interacting layers



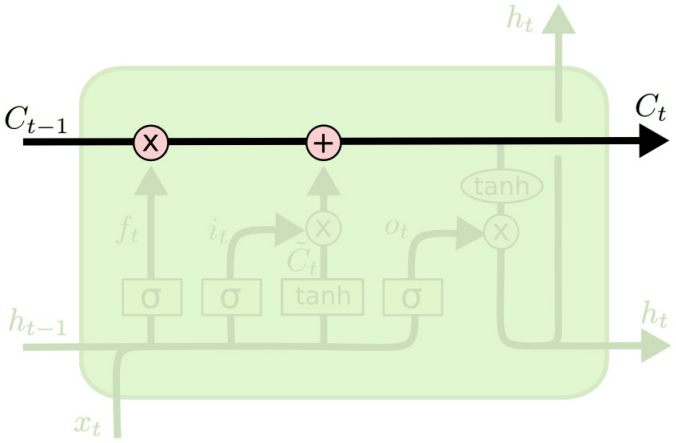
LSTM: long short term memory

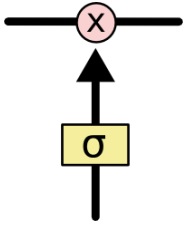
- Sometimes schematized this way



LSTM: long short term memory

- Two key ideas of LSTM:
  - A backbone to carry state forward and gradients backward.
  - Gating (pointwise multiplication) to modulate information flow. Sigmoid makes 0 < gate < 1.

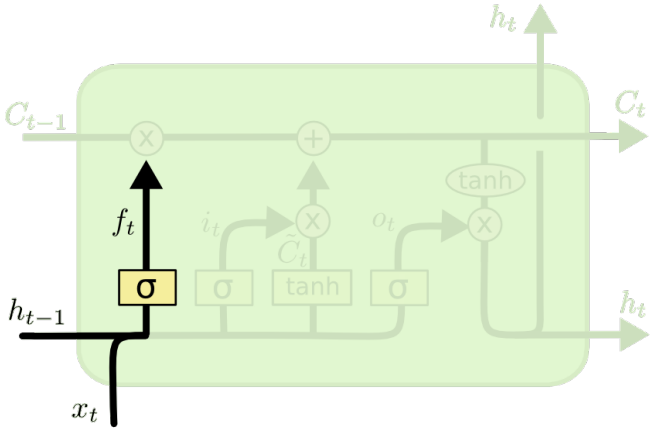




LSTM: long short term memory

• The *f* gate is 'forgetting.' Use previous state, *C,* previous output, *h*, and current input, *x*, to determine how much to suppress previous state.

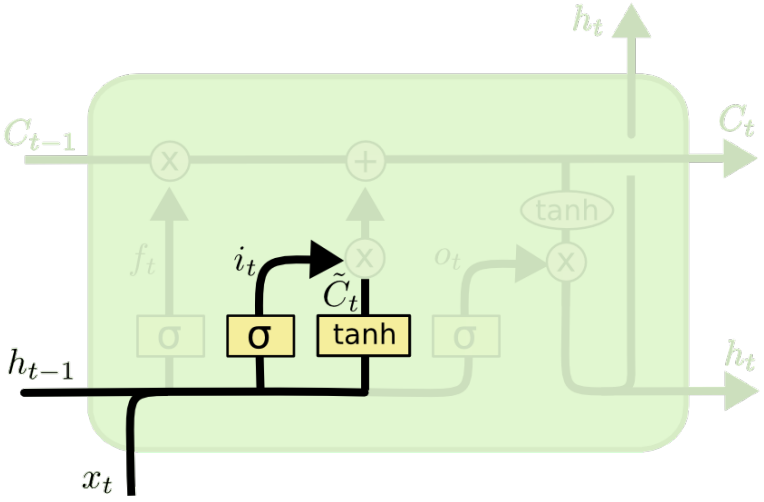
• E.g., *C* might encode the fact that we have a subject and need a verb. Forget that when verb found.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

LSTM: long short term memory

- Input gate *i* determines which values of *C* to update.
- Separate tanh layer produces new state to add to *C.*

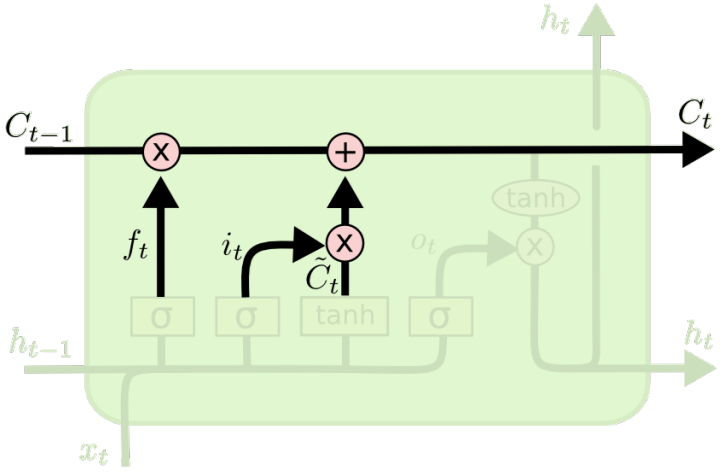


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTM: long short term memory

- Forget gate does pointwise modulation of *C*.
- Input gate modulates the tanh layer – this is added to *C*.



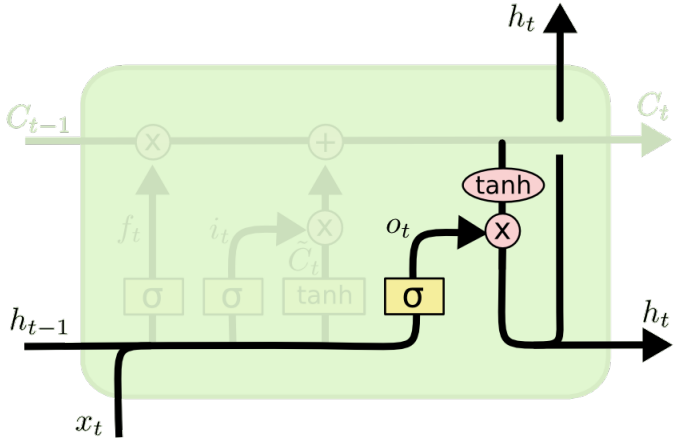
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

LSTM: long short term memory

• *o* is the output gate: modulates what part of the state *C* gets passed (via tanh) to current output *h*.

•E.g., could encode whether a noun is singular or plural to prepare for a verb.

•But the real features are learned, not engineered.



$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

LSTM: long short term memory

•Comparison

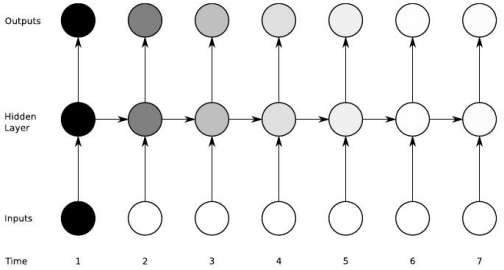


Figure 4.1: **Vanishing gradient problem for RNNs.** The shading of the nodes indicates the sensitivity over time of the network nodes to the input at time one (the darker the shade, the greater the sensitivity). The sensitivity decays exponentially over time as new inputs overwrite the activation of hidden unit and the network 'forgets' the first input.

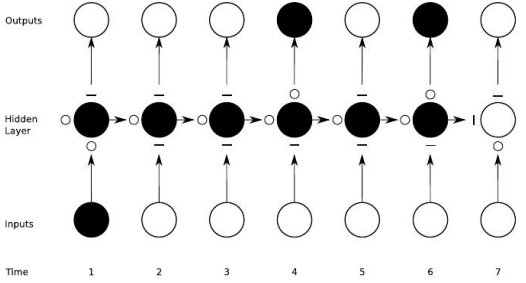


Figure 4.3: **Preservation of gradient information by LSTM.** As in Figure 4.1 the shading of the nodes indicates their sensitivity to the input unit at time one. The state of the input, forget, and output gate states are displayed below, to the left and above the hidden layer node, which corresponds to a single memory cell. For simplicity, the gates are either entirely open ('O') or closed ('—'). The memory cell 'remembers' the first input as long as the forget gate is open and the input gate is closed, and the sensitivity of the output layer can be switched on and off by the output gate without affecting the cell.