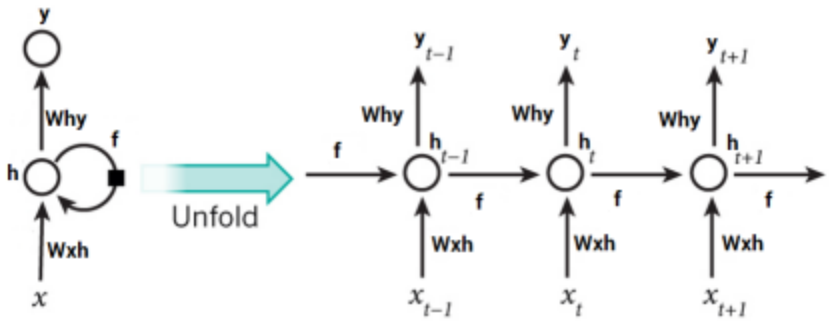
Recurrent Neural Network

What does RNN stand for?

Artificial neural networks classified as recurrent neural networks (RNN) have connections between nodes that build a directed graph in sequence along the nodes. As a result, it's able to display time-dependent dynamic behaviour.

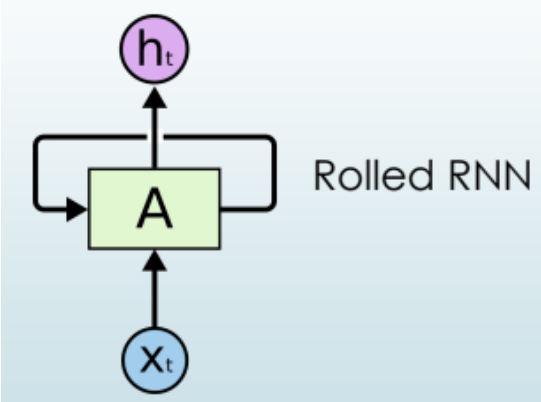
RNNs, as opposed to feedforward neural networks, may process input sequences using their internal state (memory). Because of this, they can be used for tasks like handwriting recognition or speech recognition that aren't segmented.

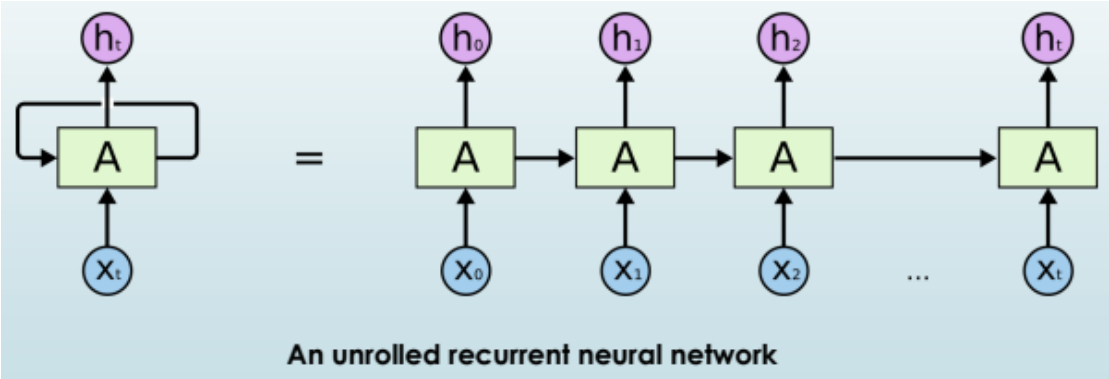


- RNNs leverage sequential information to their advantage.
- All inputs (and outputs) are assumed to be independent in a typical neural network.
- However, this is a terrible concept for the majority of work. What comes next in a sentence can be predicted by knowing what has gone before. For this reason, RNNs are known as recurrent neural networks (RNNs). The output of an RNN is dependent on the results of earlier computations.
- RNNs can also be thought of as having a "memory" that stores information about previous calculations.
- Since they can understand a sequence and context far better than other algorithms, they're the algorithm of choice for sequential data such as time series, speech and text. They're also great for financial data as well as audio and video.
- When dealing with sequential data, other algorithms are unable to provide accurate predictions.

However, when is it necessary to use a Recurrent Neural Network?

Where a sequence of data exists, "the geographical content of each frame is less relevant than the temporal dynamics that connect them." -Fridman, Lex (MIT)

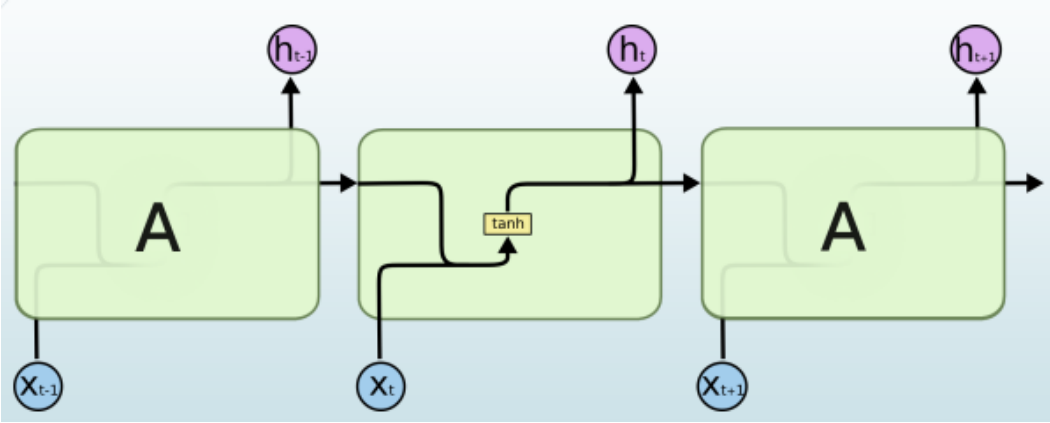




LSTM (long short-term memory)

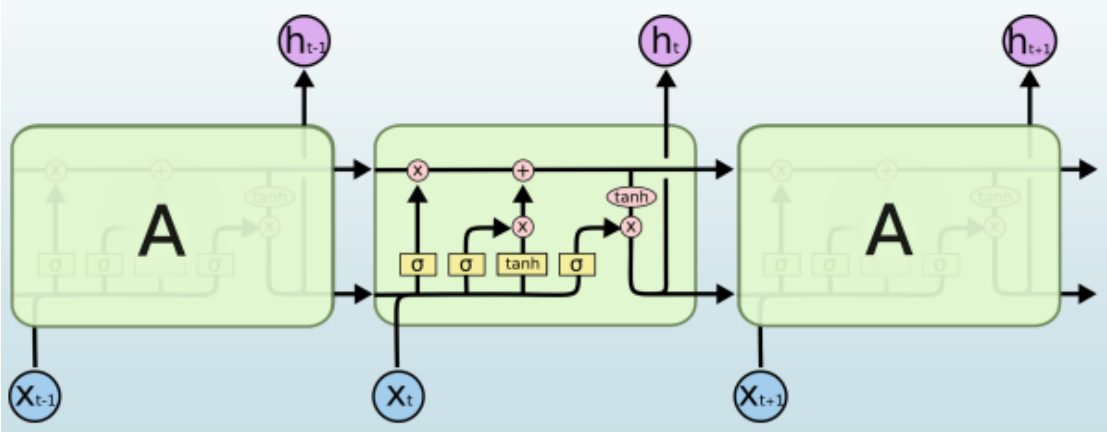
- A unique type of RNN known as "LSTMs" can learn long-term dependencies, which is why they're named "LSTMs." Hochreiter & Schmidhuber were the ones to introduce them (1997).
- The long-term reliance problem is deliberately avoided with LSTMs. It's not like they have to work hard to learn how to retain knowledge for long periods of time.

Standard RNN



Structure of the LSTM

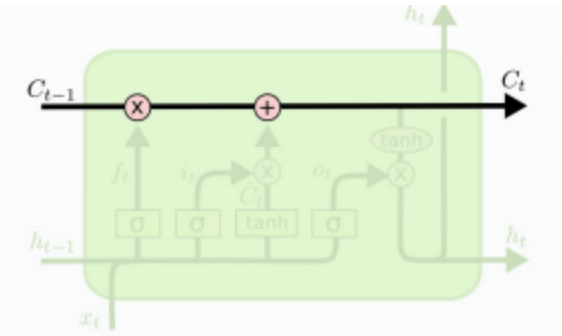
This chain-like pattern can be found in LSTMs as well, but the repeating module has a different structure. There are four layers of neural networks instead of just one, and these layers interact in a unique way.



Core Idea

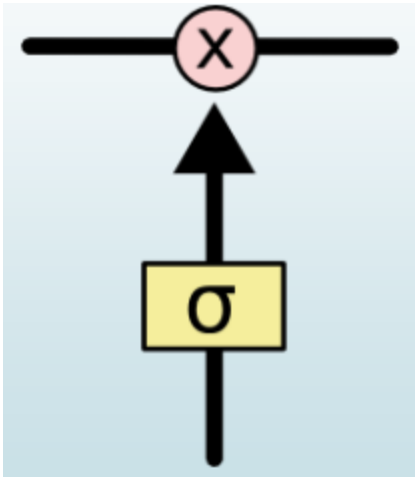
The horizontal line that runs through the top of the diagram is the key to understanding LSTMs.

The status of a cell can be compared to a conveyor belt.It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

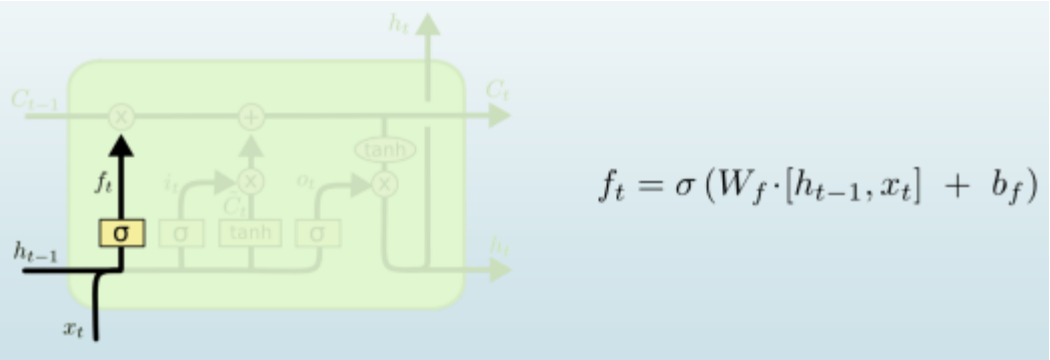


Gates

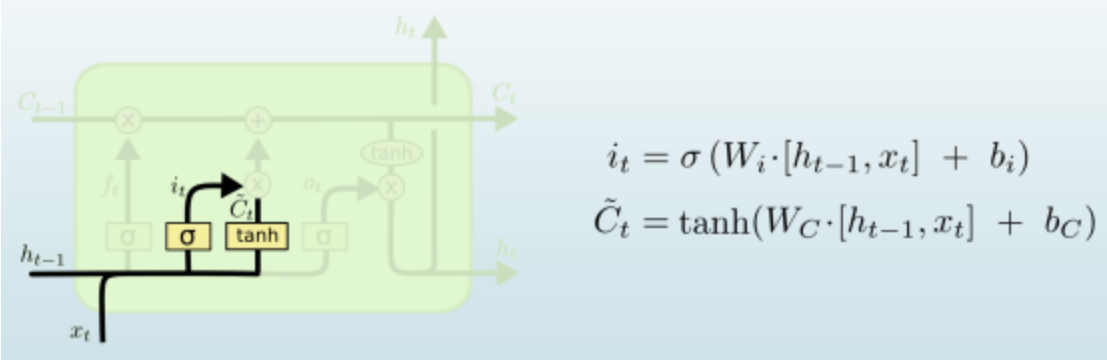
- Gates in the LSTM control how much information can be added or subtracted from a cell's state.
- Gates allow you to decide whether or not information is allowed to pass through. A sigmoid neural net layer plus a pointwise multiplication operation make up the algorithm.
- For each component, the sigmoid layer generates an output number between zero and one. Let nothing through with a value of zero, and everything through with a value of one!



The first stage in building our LSTM is deciding what information from the cell state should be discarded. The "forget gate layer" is a sigmoid layer that makes this judgement. It examines ht-1 and xt and returns a value between 0 and 1 for each number in the cell state. Ct−1. There is a 1 that means "totally keep this," and 0 that means "completely remove this."

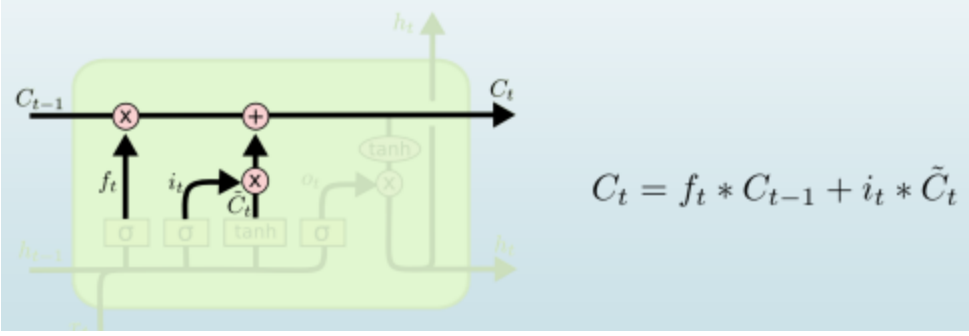


Identifying what additional information will be stored in the cell state comes next. There are two sections to this. The "input gate layer," a sigmoid layer, determines which values will be updated first. In order to add new values to the state, the tanh layer generates a vector of candidate values, Ct. We'll combine these two in the next step to make a state update.



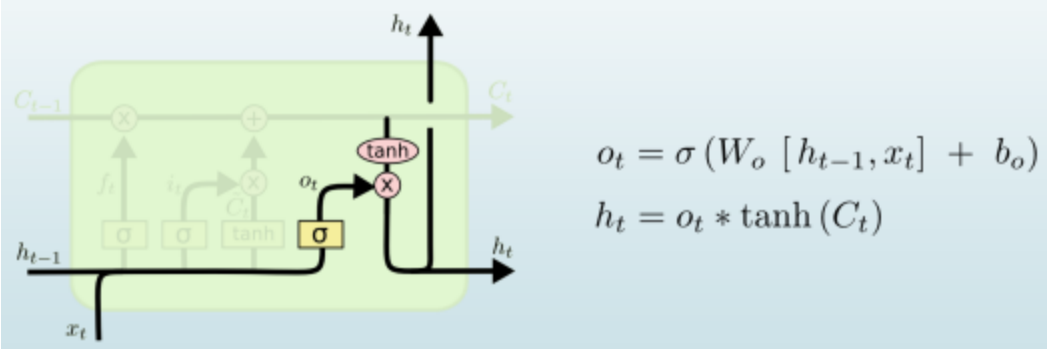
The old cell state, Ct1, must now be updated to the new cell state, Ct. We know what to do because of the previous steps; now it's time to put those plans into action.

We take the old condition and double it by foot while forgetting what we are already determined to forget. Then we add it∗Ct. A scaled-down version of the previous candidate values is shown below.



Finally, we have to make a decision about what we want to produce. This output will be derived from our current cell state, but it will be filtered.

To begin, we use a sigmoid layer to select which aspects of the cell state to output. In order to only output sections of the cell state, we pass it through tanh (to force the values to be between -1 and 1) and multiply it by the sigmoid gate's output.



RNN-based instruction

1. The network receives a single time step of input.
2. Then, using the current input and the previous state, determine the system's current state.
3. For the following time step, ht becomes ht-1.
4. As many time steps as necessary can be taken to solve the problem and the data from all prior stages can be combined.
5. To calculate an output, the final current state is used once all of the time steps have been completed.
6. In order to generate an error, you must compare your results with the expected results (i.e. the target results).
7. As a result, the weights are updated and an error is propagated backwards to the RNN.

The Benefits of the Recurrent Neural Network

- An RNN keeps track of everything that has happened in the past. Only the feature of remembering past inputs makes it suitable for time series prediction. In psychology, we refer to this as Long Short Term Memory (LSTM).

- Even with convolutional layers, recurrent neural networks are utilised to increase the effective pixel neighbourhood.

Drawbacks of Recurrent Neural Network

- Gradual disappearance and explosion issues.
- It's quite difficult to train an RNN.
- If tanh or relu is used as an activation function, it will be unable to handle very long sequences.