ANN2

Irene Volpe

r0740784

Exercise 1: Hopfield Network

Number of real attractors and number of attractor used to create the network

More data points arise the likelihood of having additional attractor.
Such spurious state have been seen as proof of the ability of the network to create new representations of data to handle the information contained in the stored patterns (Figure 1)

Time needed to reach the attractors

It typically takes 18-25 epochs for the datapoints to reach target attractors. If initially closer to these, or if we have a fewer number of datapoints even less number of epoch is sufficient.
Figure 2 shows 5 steps are not sufficient to reach the attractors, nor if the point a is initially closer nor it it's farer to the target, while instead with 50 steps in both scenarios a correctly reaches the closer target.

Final vs stored at network creation attractors

If datapoints created are far from the target attractors stored, these datapoint may converge to another nearer attractor created in addition, so these points becomes spurious states. Figure 3 and 4 show such spurious states in 2d and 3d accordingly (black circles).

Ability of Hopfield network to recontruct noisy digits

Hopfields networks also get to reconstruct noisy images, nevertheless the number of correct reconstructions depends on the amount of noise and the number of iterations (Figure 5).

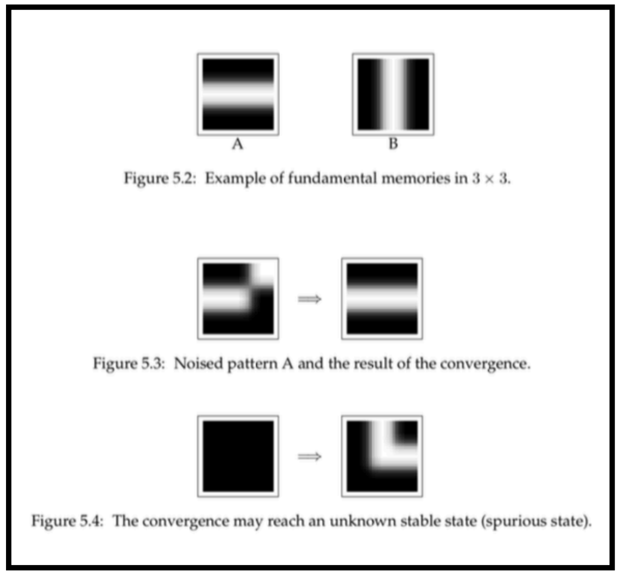


Figure 5.2: Example of fundamental memories in 3 × 3.

Figure 5.3: Noised pattern A and the result of the convergence.

Figure 5.4: The convergence may reach an unknown stable state (spurious state).

fig 1

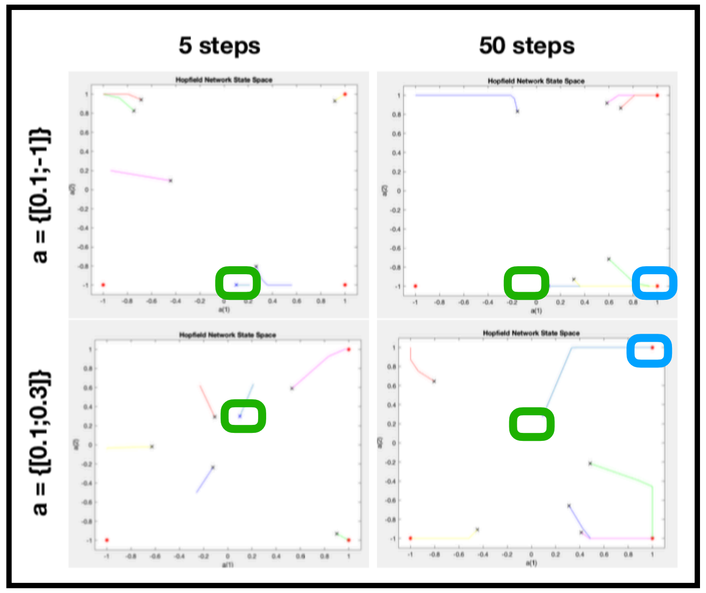


fig 2

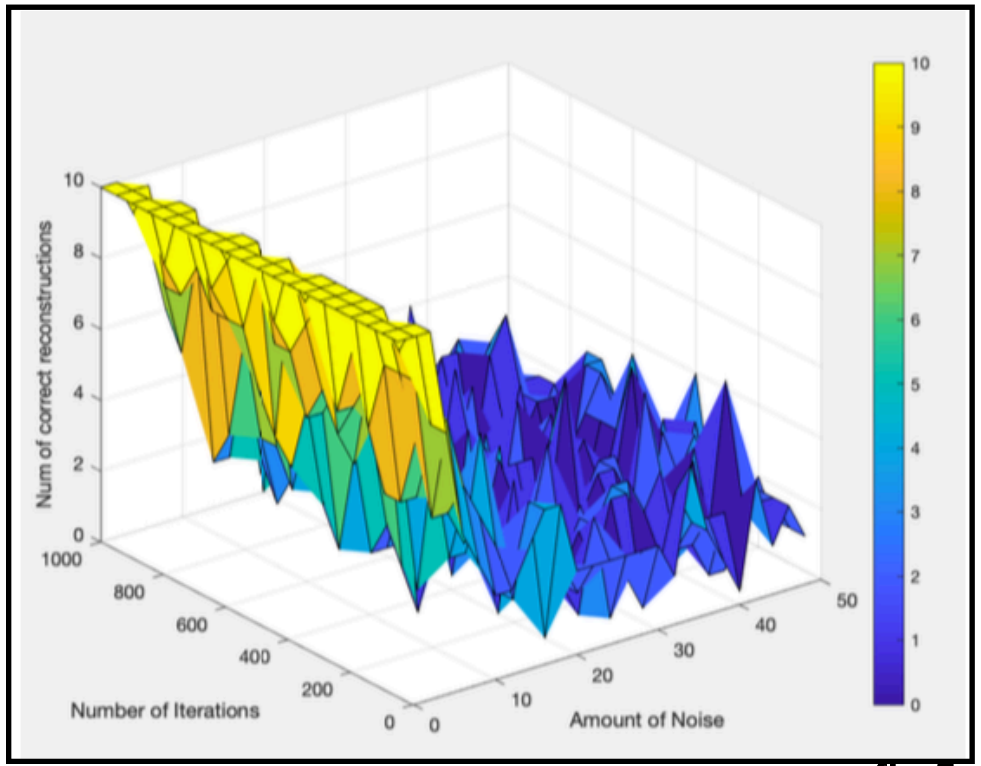


fig 5

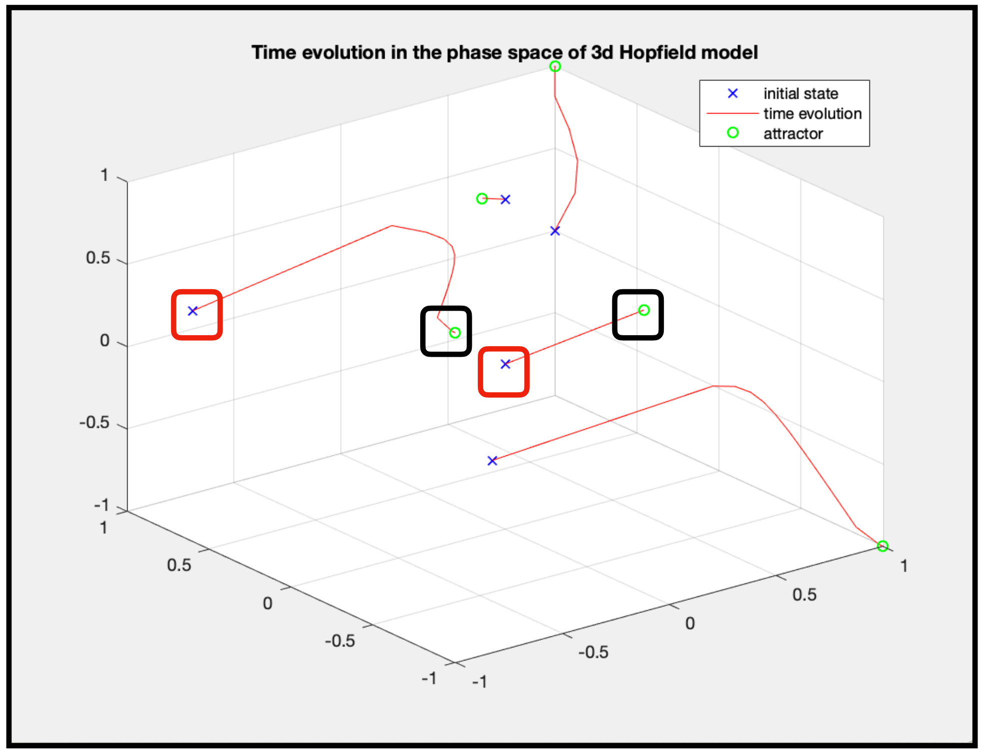
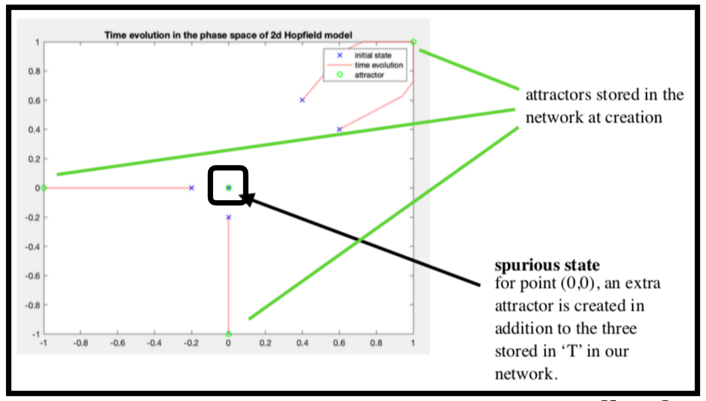


fig 4



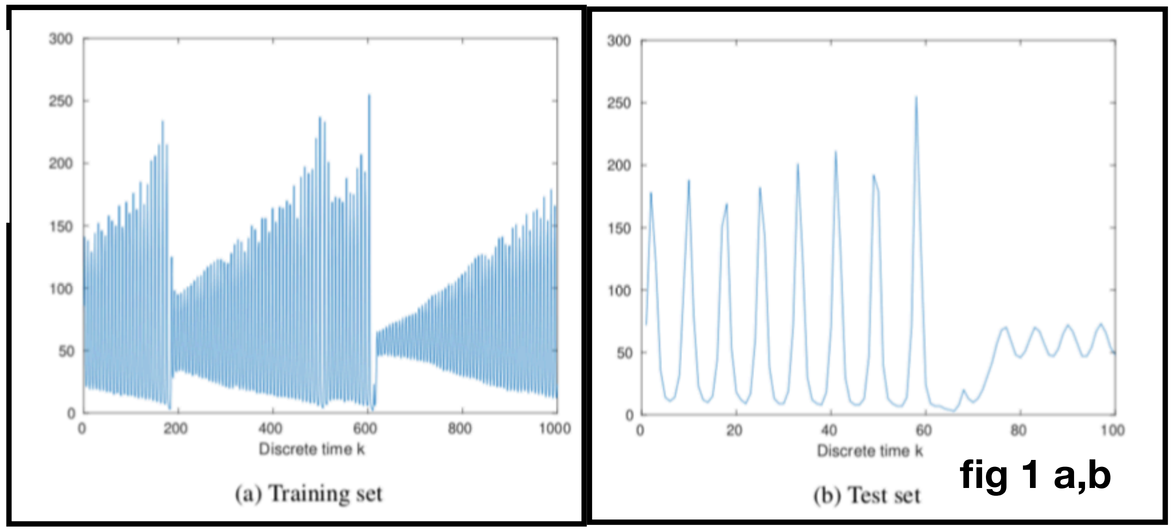
attractors stored in the network at creation

spurious state for point (0,0), an extra attractor is created in addition to the three stored in 'T' in our network.

fig 3

Exercise 2: RNN vs LSTM for time-series prediction

The Santa Fe data set is obtained from a chaotic laser which can be described as a nonlinear dynamical system. Given are 1000 training data points. The aim is to predict the next 100 points (it is forbidden to include these points in the training set!). The training data are stored in lasertrain.dat and are shown in Figure **1a.** The test data are contained in laserpred.dat and shown in Figure **1b.**



(a) Training set (b) Test set **fig 1 a,b**

RNN for time-series prediction

Investigate RNN performance using different lag and number of neurons

Figure 2a shows the comparison of the actual, in blue, and the predicted, in red, values on the training set.

#train
100 epochs;
Hidden: 100 neurons.
Time lag: 1.
task: learn to correctly predict the next 100 **training** set points

Figure 2b shows the comparison of the actual, in blue, and the predicted, in red, values on the test set.

#test
Predict the next 100 **test** set points—> with time MSE decreases.

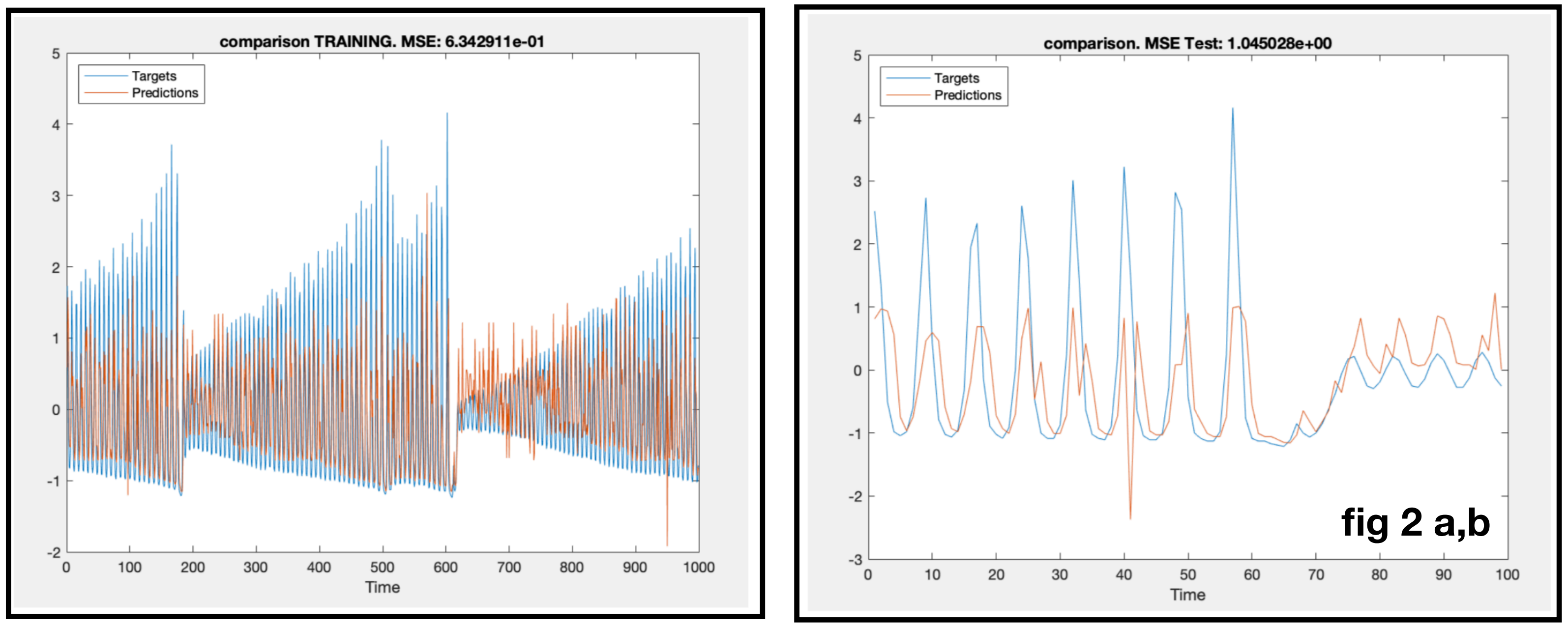


fig 2 a,b

Figure 3 shows the performance of different combination of lag values (1,2,3 and 10) and number of neurons (10,50 and 100)

#training performance of an MLP
100 epochs;
Hidden: 10, 50, 100 neurons;
time lag: 1,2,5,10.
#results
Increasing # of neurons does not affect MSE much.
Increased lag time (keep more digits in memory) reduces MSE.

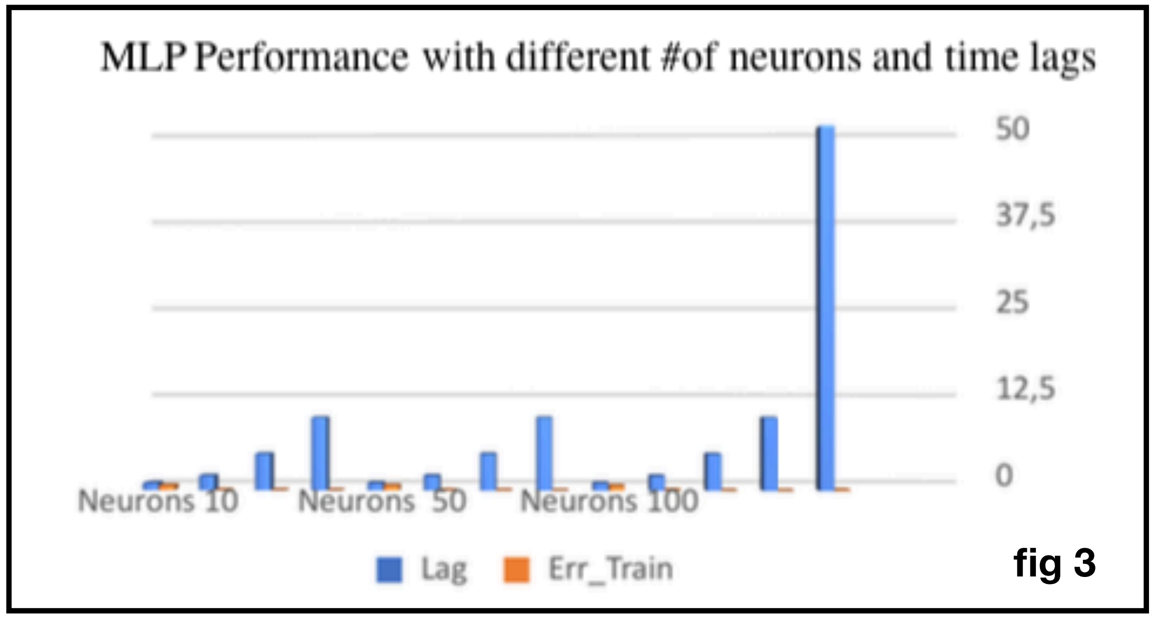


fig 3

RNN for time-series prediction

"Whenever there is a sequence of data and that temporal dynamics that connects the data is more important than the spatial content of each individual frame."
– Lex Fridman (MIT)

Train the LSTM model and explain the design process

#preparation
load the data, normalize it, and convert it to a time sequence.
split data to train the model on the first 1000 (90%) data of the sequence, and test the model on the last 100 (10%) data of the sequence.
For each epoch t, predictions(sequences without the final time step) are fed back to the model and are used to predict the value of t+1 (ie values shifted by one-time step).
#architecture and parameters
Input layer : 1 unit;
H layer: 200 neurons;
threshold: 1 (avoid vanishing/exploding gradient problems);
lr = 0.005, (decreased by 0.2 for 125 epochs).
Adam optimization algorithm to update the weights.
Gates determine how much the safe state information to use through weights and biases, which are adapted at each iteration step.
# of epochs: 250;
#training
RMSE first increases 1 to peak value of 1.4, but that drops steadily to less than our threshold value of 1 after 250 epochs.
Dropping the lr RMSE fluctuate less after about 125 epochs, and it decreases at steadier rate.
#Comparison of the forecasted values with the test data
We then compare the forecasted values with the test data.
The comparison shows that the forecasted values are closer to actual test data points till first 40 values.
RMSE at the 100th point is almost 180.

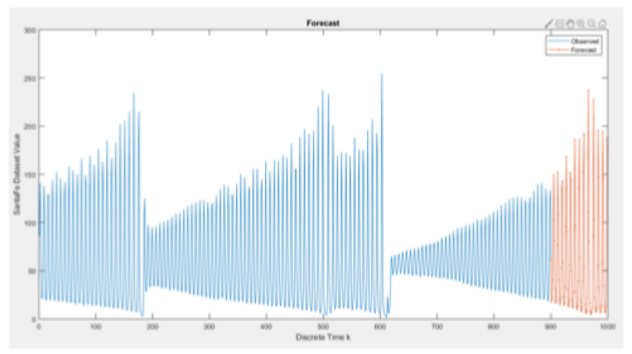
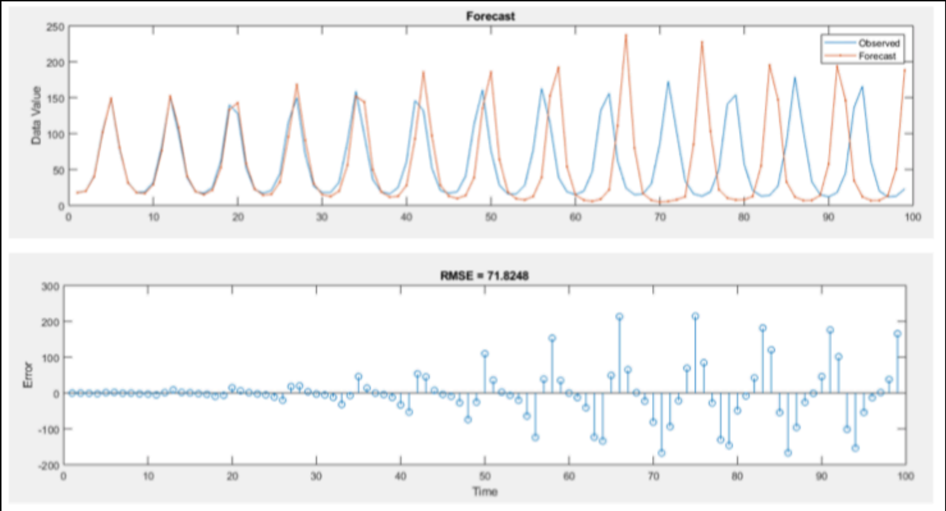
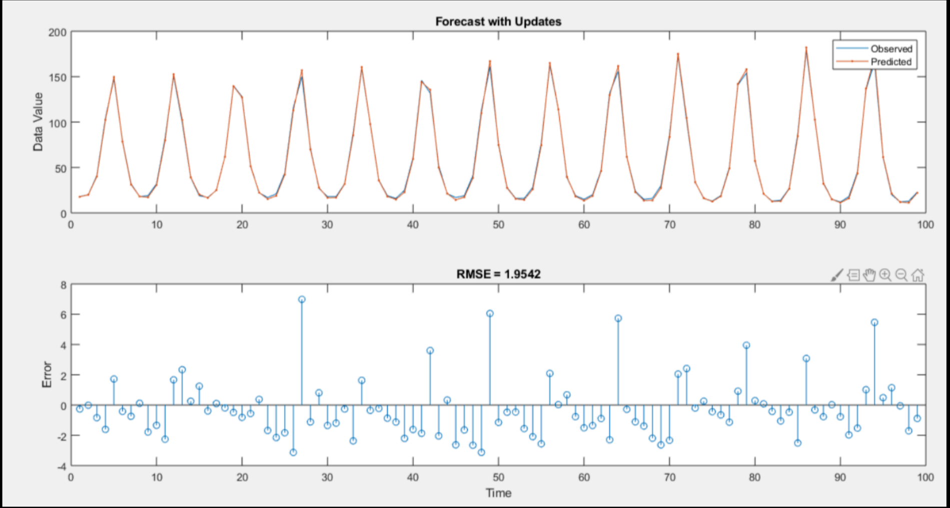


Fig b: Forecasted values 901 - 1000



Predict the test set

#test
Use the previous time step observed value to predict next time step
RMSE is calculated once more and the forecasted values are compared with the test data.
RMSE has decreased from almost 180 to 1.



Compare results of RNN and LSTM models

RNNs maintain information in 'memory' over time but can be difficult to train for solving problems that require learning long-term temporal dependencies. This is due to the vanishing gradient problem. LSTMs overcomes this with input, forget and output gate to control when information enters the memory, when it's output, and when it's forgotten. This lets them learn longer-term dependencies.

RNNs do not have a cell state and only have hidden states as the memory. LSTM has both cell states and a hidden states. The cell state can remove or add information to the cell, regulated by "gates".

LSTM nodes are more complicated than RNN nodes which makes them better at learning complex interdependencies in sequences of data.

LSTMs have different sets of weights to filter the input for input, output and forgetting in order to control block or pass of information based on its strength and import. These weights are adjusted via the recurrent networks learning process.

As LSTMs capture diverse time scales and remote dependencies and are thus like RNN operating on different scales of time at once.

Forget Gate:
Decides how much of the past you should remember.
This gate Decides which information to be omitted in from the cell in that particular time stamp. It is decided by the sigmoid function. it looks at the previous state(ht-1) and the content input(Xt) and outputs a number between 0(omit this)and 1(keep this)for each number in the cell state Ct−1.

EX: lets say ht-1 →Roufa and Manoj plays well in basket ball.
Xt →Manoj is really good at webdesigning.
- Forget gate realizes that there might be change in the context after encounter its first fullstop.
- Compare with Current Input Xt.
- Its important to know that next sentence, talks about Manoj. so information about Roufa is omited.

Update Gate/input gate:
Decides how much of this unit is added to the current state.

Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1.

EX: Manoj good webdesigining, yesterday he told me that he is a university topper.
- input gate analysis the important information.
- Manoj good webdesigining, he is university topper is important.
- yesterday he told me that is not important, hence forgotten.

Output Gate:
Decides which part of the current cell makes it to the output.

Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1 and multiplied with output of Sigmoid.

EX: Manoj good webdesigining, he is university topper so the Merit student _______________ was awarded University Gold medalist.
- there could be lot of choices for the empty dash. this final gate replaces it with Manoj.

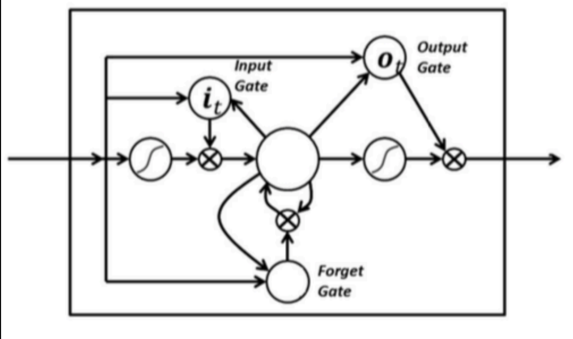


Fig a: LSTM model