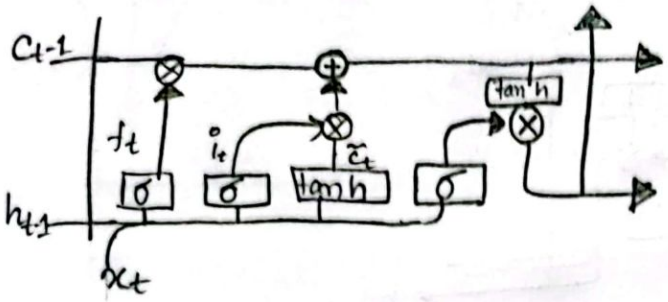
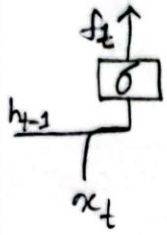
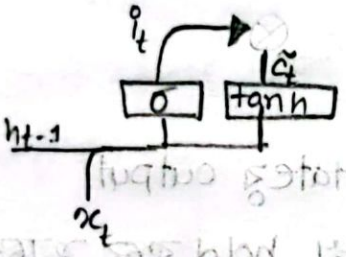
LSTM

* ৩টা sigmoid Gate হিসেবে কাজ করছে। info কতটুকু pass হবে তাটা নির্ধারণ



① 

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

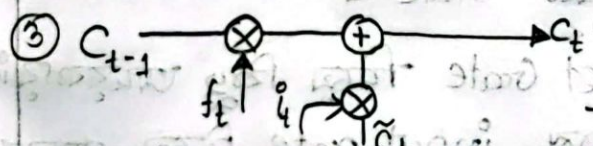
② 

→ কতটুকু input info add হবে

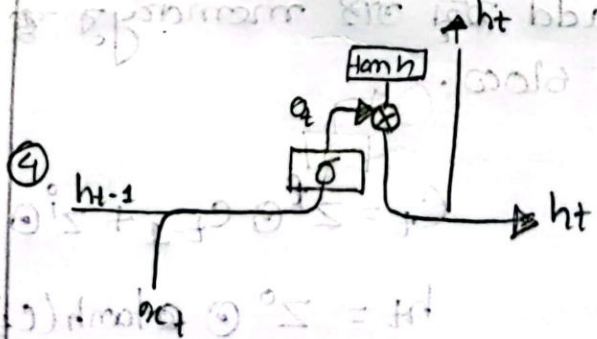
$$i_t = \sigma\left(W_i [h_{t-1}, x_t] + b_i\right)$$ [i_t decide what component is to be updated]

$$\tilde{C}_t = \tanh\left(W_c [h_{t-1}, x_t] + b_c\right)$$ [$\tilde{C}_t$ provides change contents]

↓ info normalize করতে use করি। (-1, 1)

③ 

$$\rightarrow C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$ → Updating the cell state

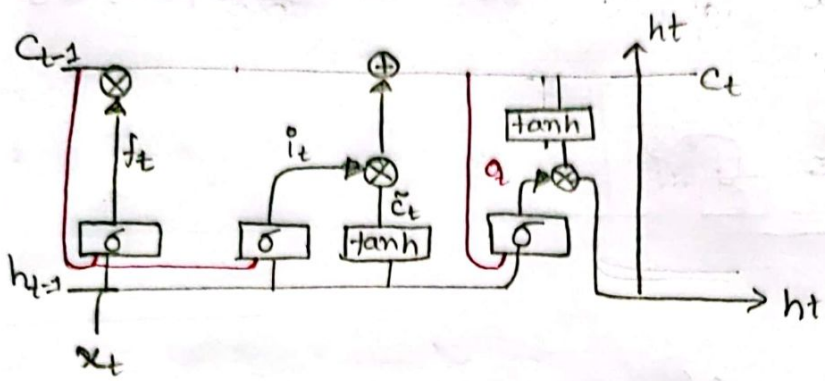
④ 

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

$$h_t = o_t * \tanh(c_t)$$

Decide what part of the cell state to output

Allows 'peeping' into the memory



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

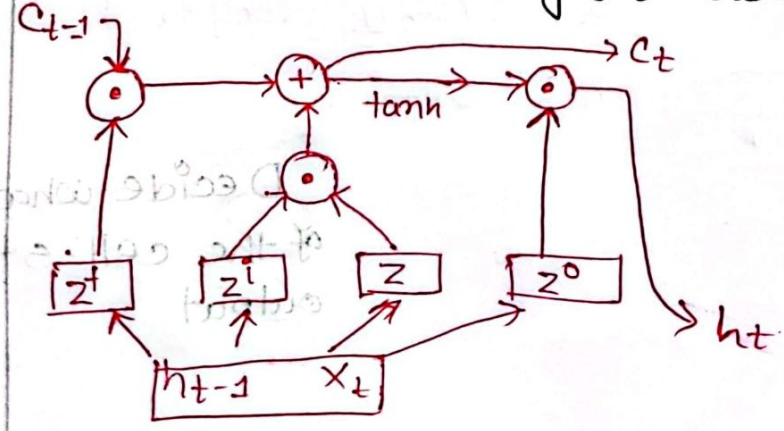
$$i_t = \sigma\left(W_i[C_{t-1}, h_{t-1}, x_t] + b_i\right)$$

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

এখানে h_{t-1} হচ্ছে immediate আগের state এর output
& C_{t-1} হচ্ছে প্রথম থেকে memory hold করে আছে।

h changes faster → কারণ new input এর আগে just merge হয়ে যাচ্ছে এক এক state-এ

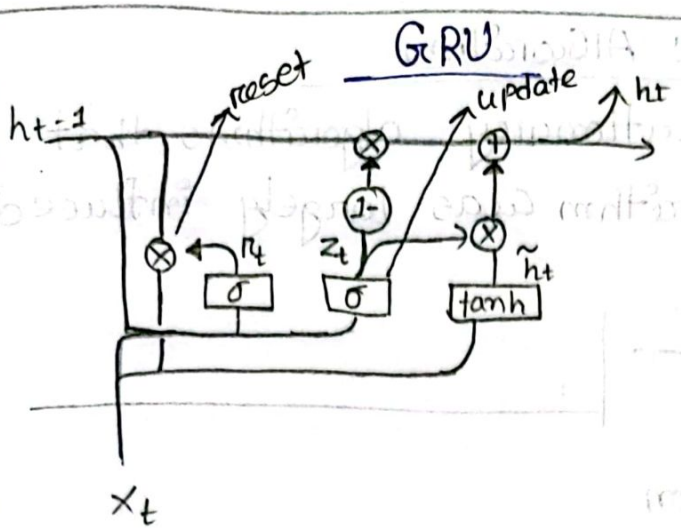
C changes slow → কারণ forget Gate দিয়ে কিছু আগেরকার info ভুলে যায় কিন্তু input gate দিয়ে আবার new info add হয় তাই memory র খুব changes টা slow.



$$C_t = z^f \odot C_{t-1} + z^i \odot z$$

$$h_t = z^o \odot \tanh(C_t)$$

$$y^t = \sigma(W' h_t)$$



$$z_t = \sigma(W_z [h_{t-1}, x_t])$$
$$r_t = \sigma(W_r [h_{t-1}, x_t])$$
$$\tilde{h}_t = \tanh(W[r_t * h_{t-1}, x_t])$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

* Update Gate = Forget + input Gate.
* merge the cell state and hidden state.
* cell state নেই (c_{t-1}) x

যেহেতু এখানে cell state বা memory state টা নেই তাই hidden state দিয়ে cell state এর calculation করা হয় যখন complexity কিছুটা কমে যায়

→ এখানে forget gate একই কাজ এ memory একই কি forget করবে। তাই new state-ও আসলে reset করতে হয়

→ এখানে যেহেতু cell state নেই তাই h_t কখনো যদি দেখে আগের info unnecessary তখন সে সেগুলো বাদ দিয়ে নতুন new info capture করবে অথবা আগের গুরুত্বপূর্ণ important হলে new state থেকে অনেক info নিয়ে আর দরকারটা old info ধরে রাখে।

This calculations within each iteration ensure that the h_t values being passed along either retain a high amount of old information or are jump-started with a high amount of new information.

& It can hold a memory with hidden layer

GENETIC Algorithm

The class of evolutionary algorithms that includes genetic algorithm was largely influced by biological evaluation.

Algorithm

1. Initialize Population
2. Select parents by evaluating their fitness.
3. Crossover parents to reproduce
4. Mutate the offsprings
5. Evaluate the offsprings (survivor selection)
6. Merge offsprings with the main population and sort.

2. **Parent selection:**

→ Random selection

→ Tournament selection → কয়েকটা grup থেকে best one বের হয়ে গেলে তাদের মধ্যে আবার selection হবে। Fitness এর ভিত্তিতে selection হবে।

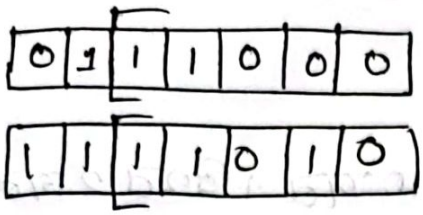
→ Roulatte wheel selection → manupulated Roulatte wheel বানাবো। যেখানে প্রতিটা ঘর same size এর থাকবে না। যাতে probability vary করে।

$P_1 = 10\%$ $P_2 = 20\%$ $P_3 = 30\%$ $P_4 = 40\%$

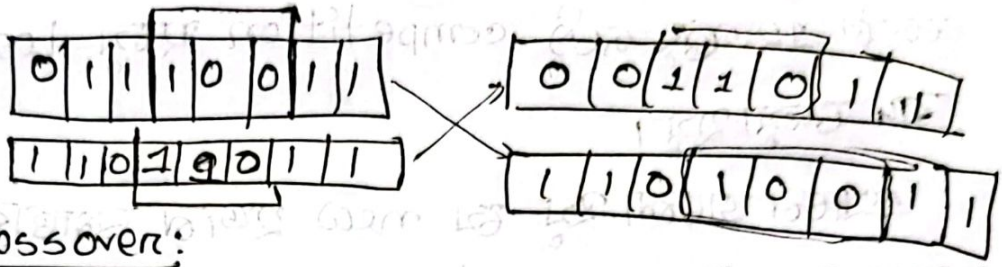
→ অর্থাৎ parent এর যাদের fitness value বেশি তাদের probability value বেশি হয়ে যাবে।

3. Cross-Over

Single Point crossover:

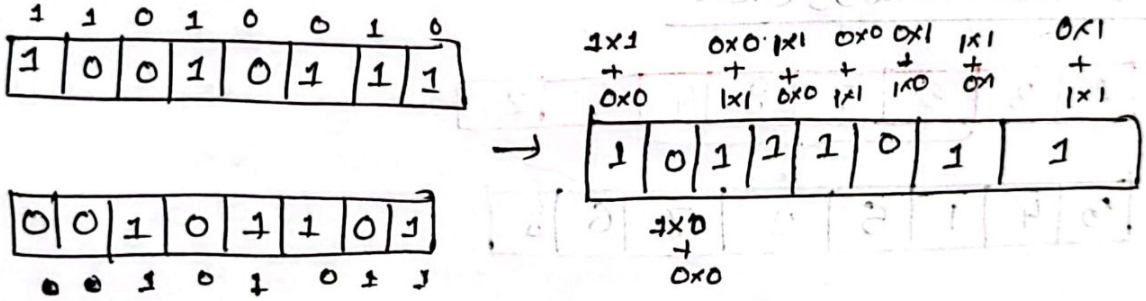


multiple



Uniform crossover:

Randomly choose which genes are supposed to be inherited from both the parent chromosomes. Then model them as 0s and 1s. The gene to be inherited is encoded as 1 and the gene that should not be inherited is encoded as 0.



④ mutation → is a natural process that occurs due to an error in replication or copying of genes.

* Bit Flip mutation
* Swap mutation
* Scramble mutation → দুটা point ধরে ভিতরে value গুলোর pos change করা।
* Inversion mutation → range ধরে just উলটা করে serially বসানো।

5. Evaluate the offsprings (Survivor selection)

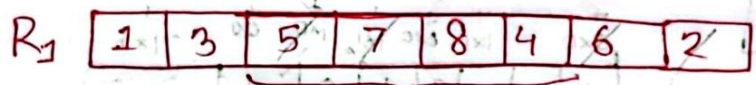
→ Age Based selection: সবচেয়ে পুরোনো parent কে replace করব।

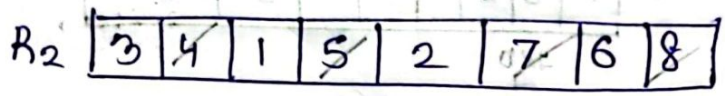
→ Fitness based selection: আগের 10 জন + এখন 2 জন মোট 12 জনের মধ্যে competition হয়ে Loser 2 জনকে বাদ দেয়া হয়।

এখানে ভালো হয় যে new 2 জন আগের 10 জন থেকে বাজে বা fitness value কম so competition 12 জনের মধ্যে হচ্ছে better parent-ই survive করবে।

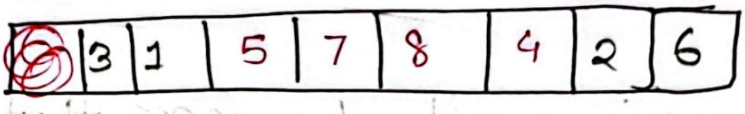
~~Davis O~~

Davis Order Cross-over

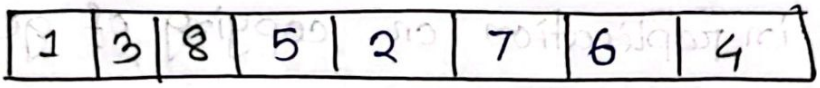




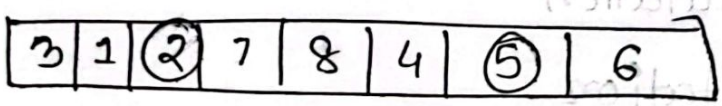
Crossover: C_1



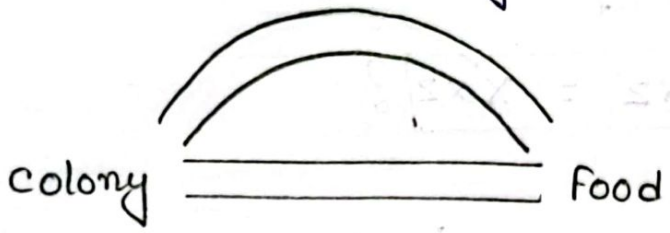
Again C_2



mutation



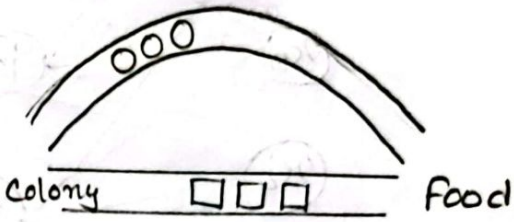
Aant Colony Optimization (ACO)



stage 1

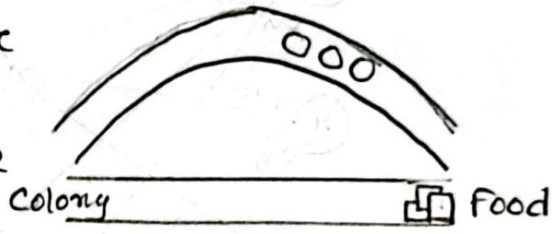
Stage 1: All ants are in their nest. There is no pheromenone content in the environment.

Stage 2: Ants begin their search with equal ($\frac{1}{2}$=0.5) probability along each path. Clearly the curved path is the longer and hence the time taken by the ants to reach food is greater than other.



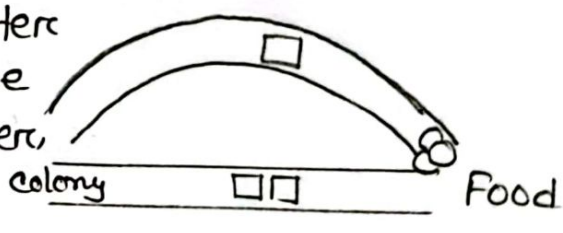
stage 2

Stage 3: The ants through the shorter path reaches food source earlier. Now, evidently they faces with the similar selection dilemma, but this time due to phenomenon trail along the shorter path already available probability of selection is higher.

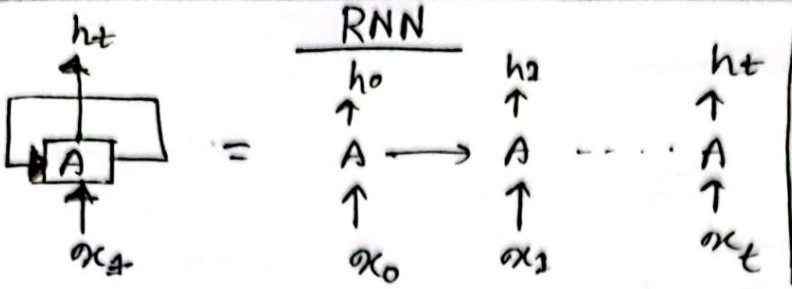


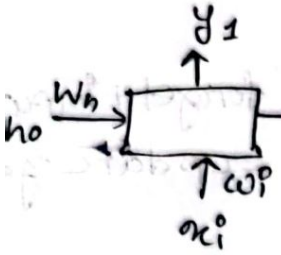
stage 3

Stage 4: More ants return via the shorter path and subsequently the pheromenoe concentration also increase. Moreover, due to evaporation, the pheromone concentration of longer path reduces, decrease the probability of selection of this path in further stages. Therefore, the whole colony gradually uses the shorter path in higher probabilities. So path optimization is attained.

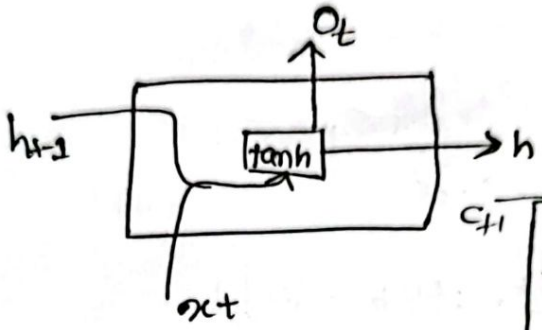


stage 4

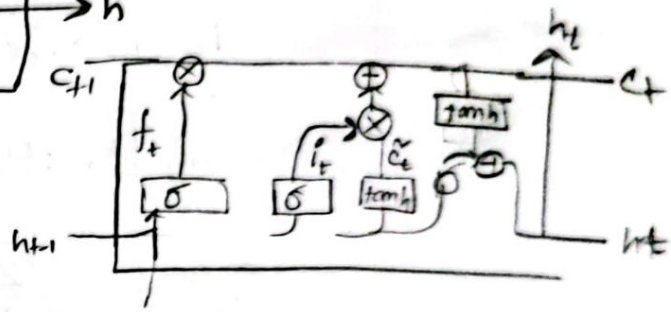




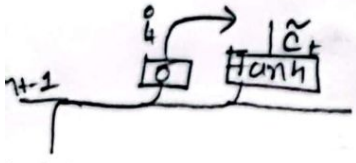
$W_h h_0 + W_i x_1 + b$ $\quad y_1 = h_0 h'$



LSTM

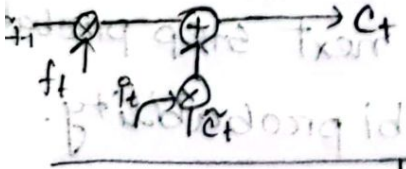


$f_t = \sigma(W_f(h_{t-1}, x_t) + b_f)$

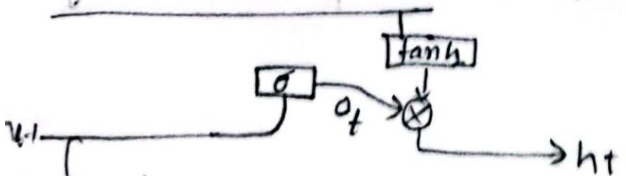


$i_t = \sigma(W_i(h_{t-1}, x_t) + b_i)$ → i_t decide what component to be updated

$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$ → $\tilde{C}_t$ provides change contents



$C_t = C_{t-1} * f_t + i_t * \tilde{C}_t$ → updating the cell state



$h_t = \tanh(C_t) * O_t$

$O_t = \sigma(W_O[h_{t-1}, x_t] + b_O)$

Decide what part of the cell state to output.