Improving RNN by hyper parameter tuning using Genetic Algorithms

...

Siddharth Dwivedi
Yash Rathore
Jai Singh

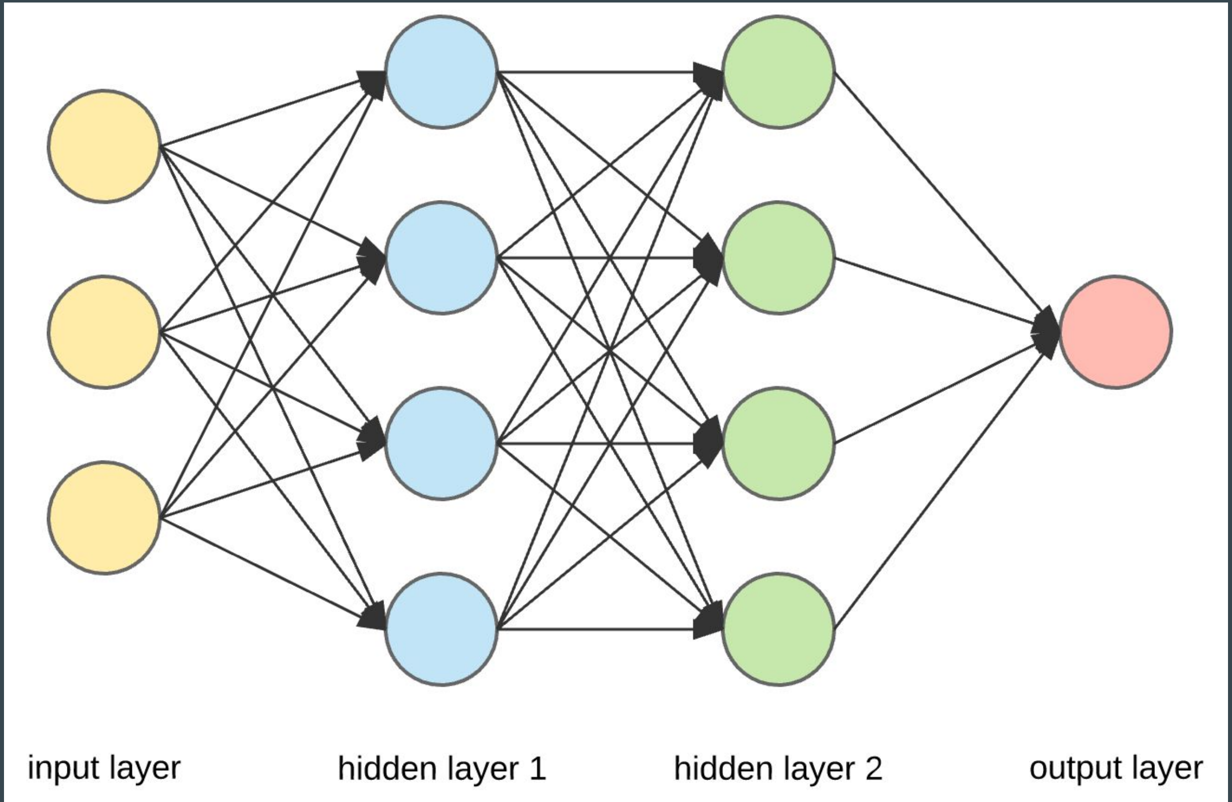
Objective

- Improve Recurrent Neural Network accuracy by optimally setting the values of hyper-parameters used.
- Finding the set of optimal hyper-parameters is an NP-Hard problem. NP-Hard problems are those which cannot be solved in polynomial time.
- One of the best ways of solving such a problem is the use of Genetic Algorithm, which are algorithms based on the theory of natural selection

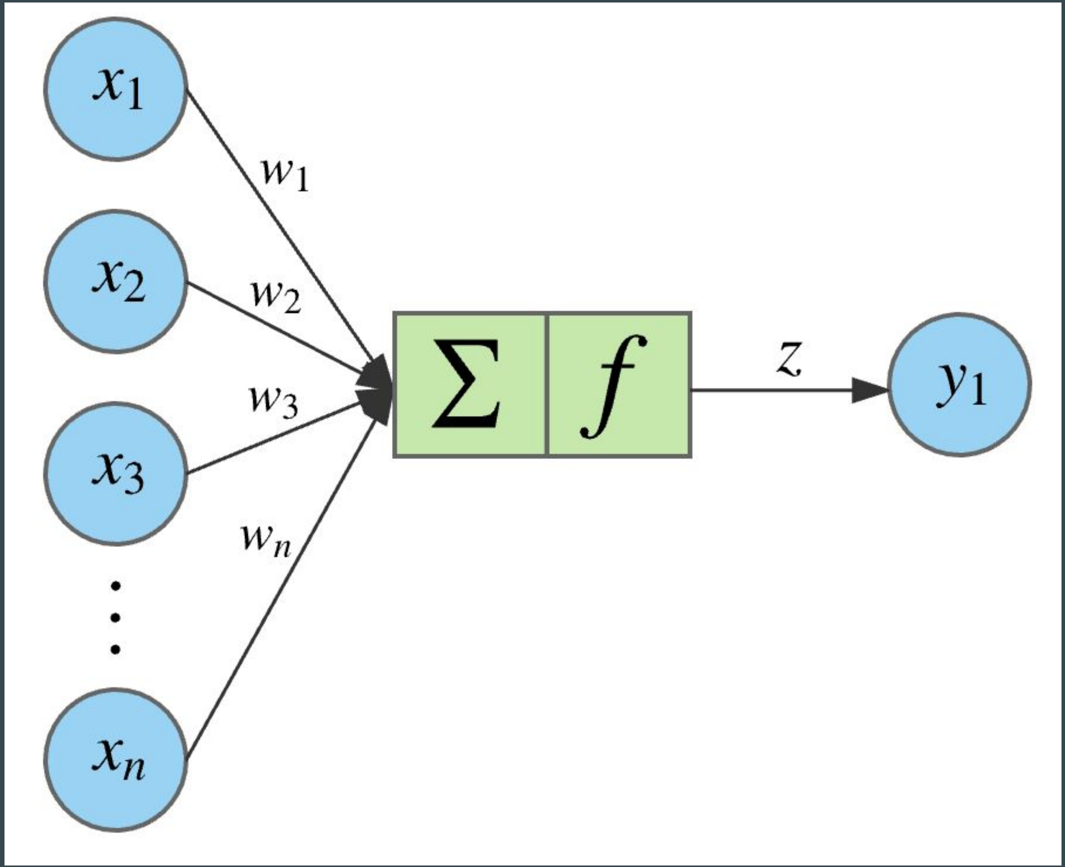
Artificial Neural Network (ANN) and Recurrent Neural Network (RNN)

- ANN are algorithms inspired by biological neural networks of animal brains.
- ANN are composed of artificial neurons, which recieve an input, combine the input with an internal state (using activation function) and produce an output using output function
- ANN are extensively used for prediction, and classification purposes
- RNNs are a class of artificial neural networks which exhibit temporal dynamic behaviour which makes them applicable for tasks such as handwriting recognition, or speech recognition

ANN Model



This is a typical Artificial Neural Network architecture with 2 hidden layers.



A zoomed in look at what the weights and nodes of a layer look like. All weights and inputs are multiplied and summed, after which an activation function is used to get a desired value.

$$z = f(b + x \cdot w) = f\left(b + \sum_{i=1}^{n} x_i w_i\right)$$

$$x \in d_{1\times n},\ w \in d_{n\times 1},\ b \in d_{1\times 1},\ z \in d_{1\times 1}$$

Why RNN?

- ANNs can’t deal with temporal data
- ANNs lack memory
- ANNs have a fixed architecture

Lenny Khazan is a criminal, and should be sent to ______.

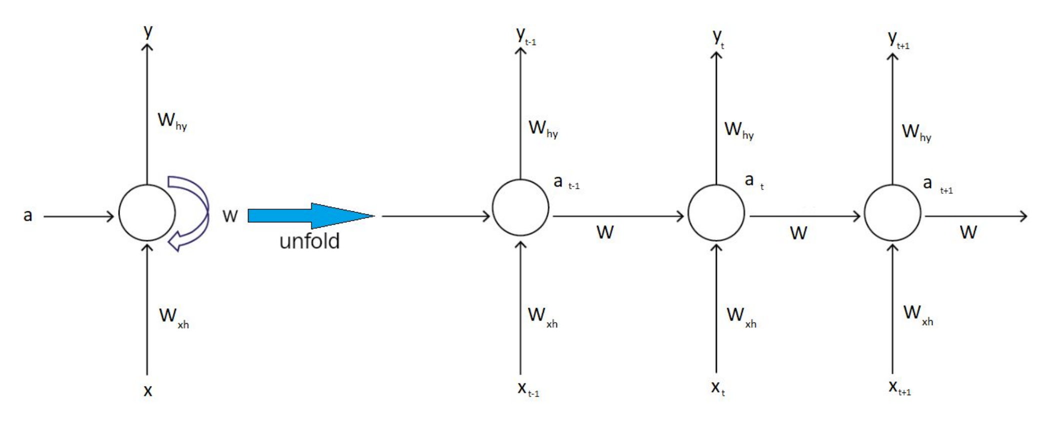
Lenny Khazan is a criminal, and should be sent to school.

Lenny Khazan is a criminal, and should be sent to the military.

Lenny Khazan is a criminal, and should be sent to heaven.

Lenny Khazan is a criminal, and should be sent to jail.

RNN Architecture



Where:

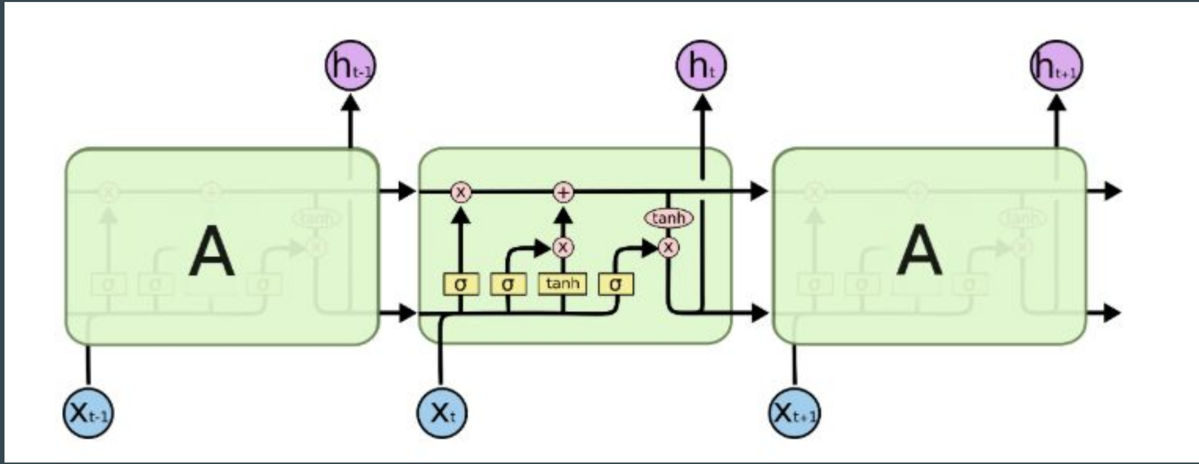
- Wxh: input-layer to hidden-layer weights
- W: hidden layer to hidden layer weights
- Why: hidden-layer to output layer weights
- a: Activation Function

Formula for calculating current state: $a_t = f(a_{t-1}, x_t)$
Formula for calculating output: $y_t = W_{hy} * a_t$

Long Short Term Memory (LSTM)

- Generic RNN with artificial neurons have short memory. This is because of Vanishing Gradient problem
- The Vanishing Gradient problem refers to a minimal update in the gradient descent step for the initial layers of the network during back propagation
- To solve this problem, the LSTM unit was proposed
- LSTM cells handle memory in a very intelligent way, enabling them to learn long-term dependencies and perform better than other units

LSTM Architecture



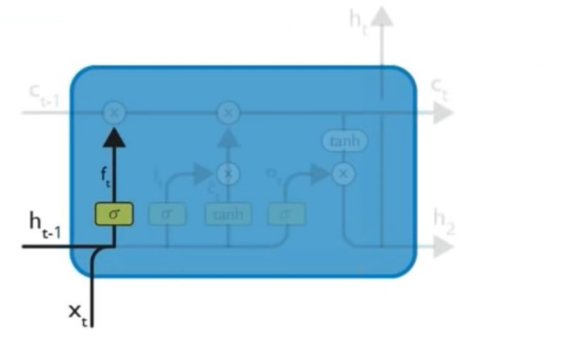
$$i_t = \sigma\big(x_t U^i + h_{t-1} W^i\big)$$
$$f_t = \sigma\big(x_t U^f + h_{t-1} W^f\big)$$
$$o_t = \sigma\big(x_t U^o + h_{t-1} W^o\big)$$
$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$
$$C_t = \sigma\big(f_t * C_{t-1} + i_t * \tilde{C}_t\big)$$
$$h_t = \tanh(C_t) * o_t$$

Long Short Term Memory

STEP-1

The first step in the **LSTM** is to identify those information that are not required and will be thrown away from the cell state. This decision is made by a sigmoid layer called as forget gate layer

$$w_f = Weight$$
$$h_{t-1} = Output\ from\ the\ previous\ time\ stamp$$
$$x_t = New\ input$$
$$b_f = Bias$$



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

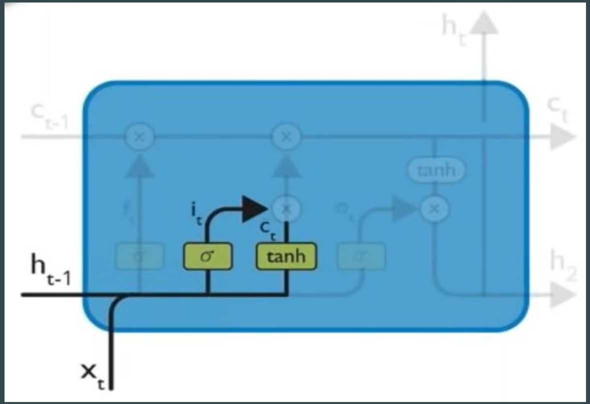
Long Short Term Memory

STEP-2

The next step is to decide, what new information we're going to store in the cell state. This whole process comprises of following steps. A **sigmoid layer** called the "input gate layer" decides which values will be updated. Next, a **tanh layer** creates a vector of new candidate values , that could be added to the state.

$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i)$$

$$\tilde{c}_t = tanh(w_c[h_{t-1}, x_t] + b_c)$$

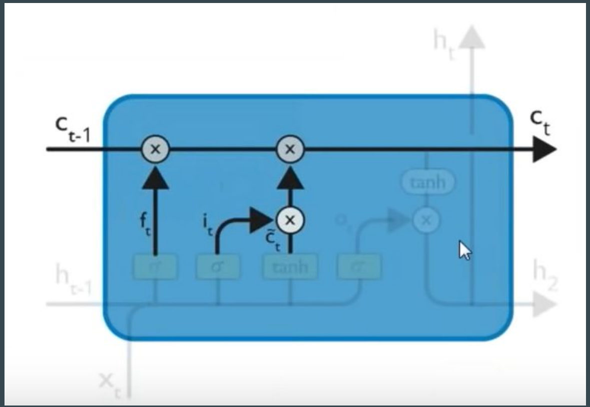


Long Short Term Memory

STEP-3

Now, we will update the old cell state, C_{t-1}, ino the new cell state C_t. First, we multiply the old state (C_{t-1}) by f_t, forgetting the things we decided to forget earlier. Then, we add $i_t * c\sim_t$. This is the new candidate values, scaled by how much we decided to update each state value.

$$c_t = f_t * c_{t-1} + i_t{}^* \, c\tilde{}_t$$



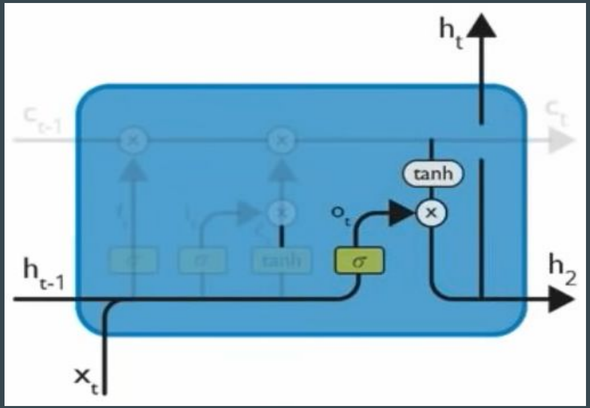
Long Short Term Memory

STEP-4

We will run a sigmoid layer which decides what parts of the cell state we're going to output. Then, we put the cell state through tanh (push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

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

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



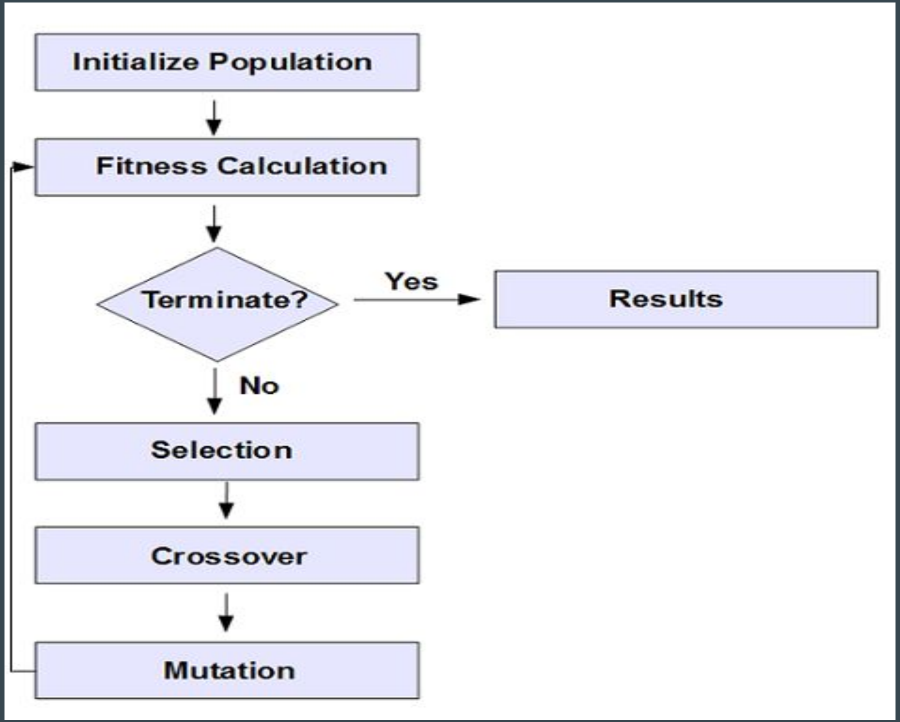
Hyper-Parameters in RNN

- Hyper-parameters are those parameters whose value is set before the learning process begins. By contrast, the values of other parameters are derived via training.
- Different model training algorithms require different hyperparameters, some simple algorithms require none.
- Different Hyper-parameters in RNN include: number of hidden layers, number of nodes in each layer, learning rate for gradient descent, number of iterations, unit of RNN used, etc.

Hyper-Parameter Tuning using Genetic Algorithm

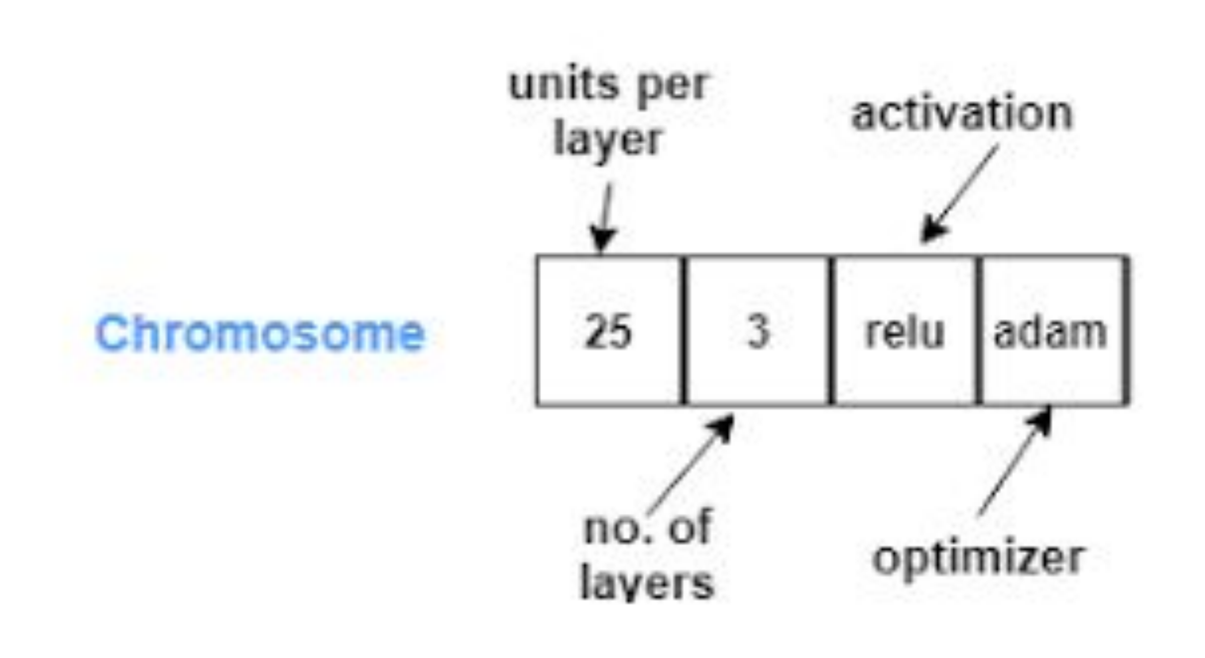
- The choice of hyper-parameters affect the accuracy of the output of the Recurrent Neural Network, given some input
- We use Genetic Algorithms to find optimal hyper parameters for a given model
- Genetic Algorithms uses 3 operations viz. Crossover, Mutation, and Selection
- Crossover: Used to multiply population and to explore search space
- Mutation: Used to exploit a particular solution space
- Selection: Using a fitness criteria, we select the best solution for our problem

Genetic Algorithm architecture



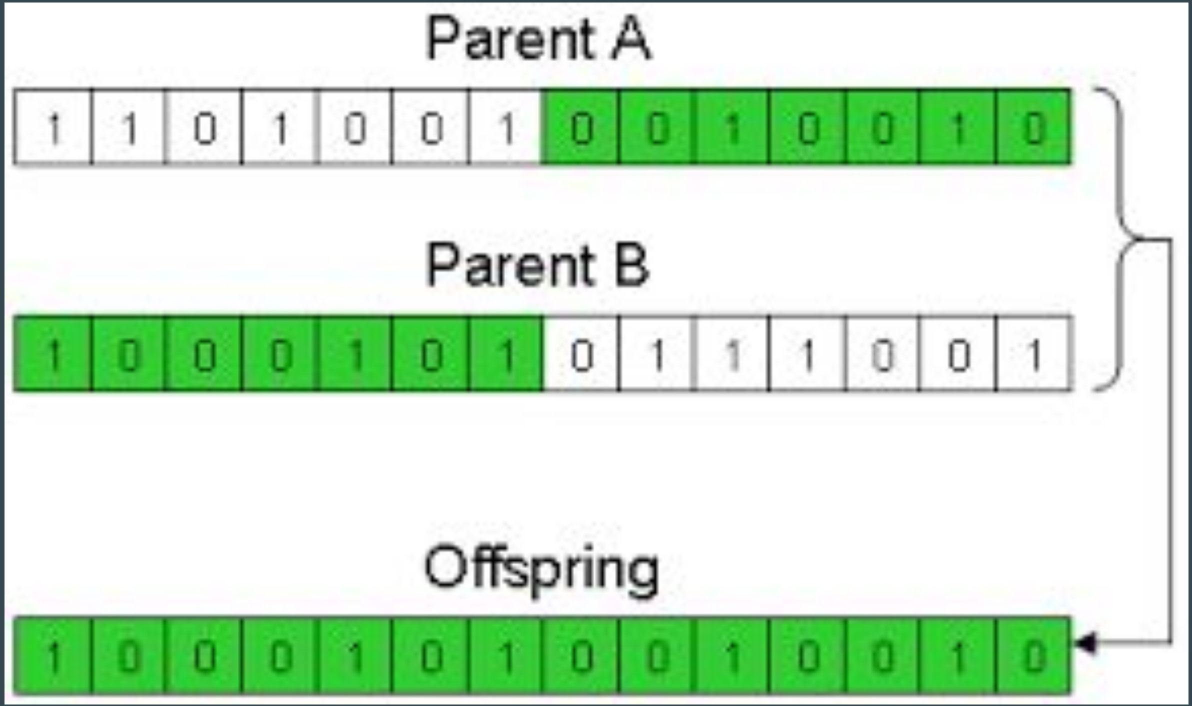
- On the left, a generic GA architecture is shown with all the operations which are performed in the optimisation process.

Chromosome



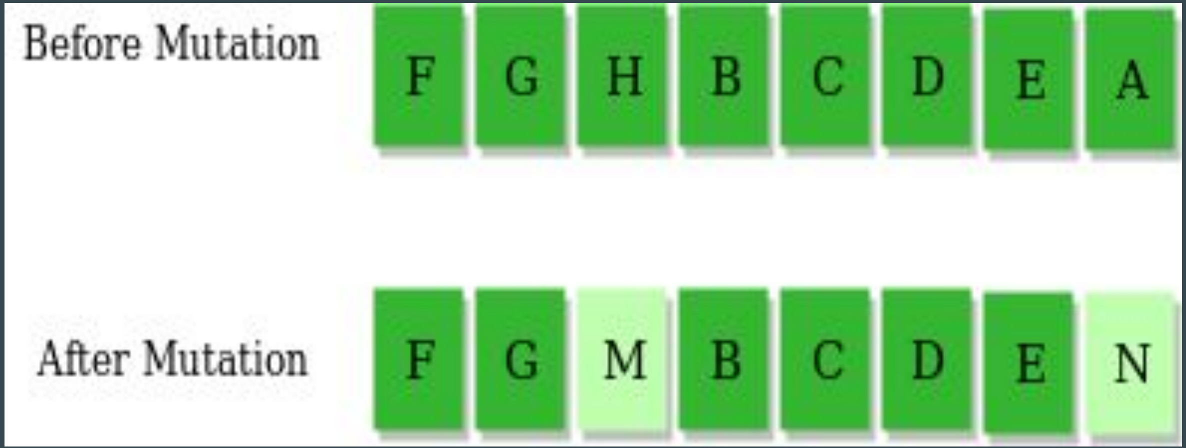
Here is an example chromosome (array) upon which the algorithm may work. Each block represents a hyperparameter which may be changed and tuned.

Crossover Operation



This is an example of the crossover operation. In which the two halves of both parents are combined to form an offspring.

Mutation Operation



This is an example of the Mutation operation on a chromosome. Few values are randomly changed to explore the different values possible.

Working

- The initial population consists of multiple arrays of hyper parameters in which a single array is the chromosome upon which the genetic algorithm works
- All the 3 operations will be performed on an initial random population
- From each generation, the best chromosomes will be selected using the fitness criteria, which in this case is the accuracy of the RNN model using those particular hyper-parameters
- After a pre-defined number of iterations of the above model, we will finally use the hyper-parameters which will give us a higher accuracy without changing anything in the RNN model. This can easily be applied to all RNN problems such as Machine Translation, Music generation etc to improve accuracy.

THANK YOU!!