Machine Learning on Combinatorial Games: Predicting if a configuration is winning or losing in the game of Nim

A B. Tech Project Report Submitted in Partial Fulfillment of the Requirements for the Degree of

Bachelor of Technology

by

Jigyasa
(160122016)

under the guidance of

Benny George

to the

DEPARTMENT OF CHEMISTRY

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
GUWAHATI - 781039, ASSAM

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Machine Learning on Combinatorial Games: Predicting if a configuration is winning or losing in the game of Nim**" is a bonafide work of **Jigyasa** (**Roll No. 160122016**), carried out in the Department of Computer Science and Engineering in collaboration with the Department of Chemistry, Indian Institute of Technology Guwahati under my supervision and that it has not been submitted elsewhere for a degree.*

May 20, 2020

Supervisor: **Benny George**

Assistant Professor,

Department of Computer Science & Engineering

Indian Institute of Technology Guwahati, Assam.

DECLARATION

I hereby declare the work embodied in this project report entitled ***Machine Learning on Combinatorial Games: Predicting if a configuration is winning or losing in the game of Nim*** *is the result of investigations carried out by me in the Department of Computer Science and Engineering in collaboration with the Department of Chemistry, Indian Institute of Technology Guwahati under the guidance of* ***Prof. Benny George****. Due acknowledgements have been given as per scientific tradition, where ever the work described is based on the findings of other investigator.*

Jigyasa
Department of Chemistry
Indian Institute of Technology Guwahati, Assam

May 20, 2020

Statement Verified.

Prof. Benny George

Contents

Abstract

Machine Learning is making researchers believe in the ability of training machines to think and make a decision. Since chemistry has a huge database of molecules and compounds and their different kinds of properties which is, as of now majorly handled manually, there is a huge scope of introducing machine learning algorithms to ease out various classifications in chemistry. Even though my thesis doesn't involve Chemistry directly, the studies in the field of machine learning can be extended to find tremendous applications in the field of Chemistry.

Nim is a combinatorial game which can be played by humans easily. My bachelors' thesis project is based on determining if a machine learning model can predict if a given configuration of the game is a winning or a losing configuration. This analysis will also help us conclude whether it is possible for a machine to learn the XOR behaviour. Since XOR is a logic gate, we can also draw conclusions for other gates as well. Any algorithm can be thought of being composed of a circuit of logic gates and logic gates can be simulated by neural networks. Hence by verifying the claim that a logic gate, XOR in this case, can be modeled by a machine, we would be able to conclude that any algorithm can be learnt by a machine. We aim to verify the claim of Machine Leaning that a machine can learn any algorithm.

If a machine could deduce the strategy to win a game, it could probably in future even help deduce whether a certain molecule could be used as a drug which would be a phenomenal change towards making chemical classifications automated.

Chapter 1

Introduction

1.1 Machine Learning and Reinforcement Learning

It is the ability of a machine to learn automatically based on the previous experiences without explicitly programming. The machines are made to learn using the examples provided to them initially which helps them deduce a pattern between them and make decisions on any future input. This is done without there being any human interference and hence the name ‘Machine Learning’. Reinforcement Learning is a kind of Machine Learning in which there is a ‘Reward’ and a ‘Punishment’ associated with every action a machine takes. So that the machine is penalised for any wrong actions, which in turn helps to maximise the cumulative reward associated.

Neural Networks

In this thesis, the focus is on studying prediction of Neural Networks. Neural networks is a series of algorithms, modeled loosely like the human brain, which aims to find patterns in data. They recognize numerical patterns, denoted by vectors, all real-world data, images, sound, text or time series, must be translated into a vector representation. Neural Networks are used to classify and cluster. They allow us to label and group unlabelled data based on similarities after having recognised the patterns in labelled data on which they are trained on. Neural Network is composed of a circuit of artificial neurons called as nodes which are used to solve problems in the field of Artificial Intelligence. Hence in this thesis we aim to analyse if it is possible to model a player's brain in a neural network, and draw conclusions if a machine will be able to play a game as optimally as a human being.

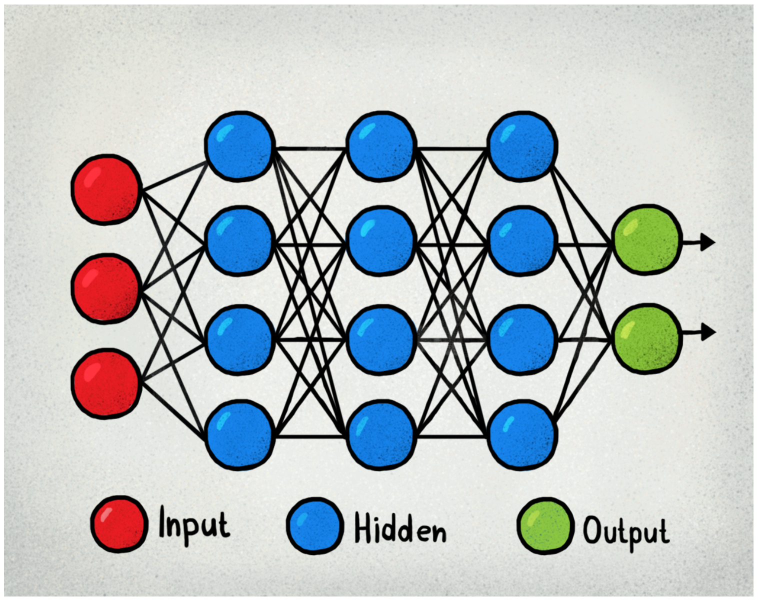


Fig. 1.1 3-3-3 Neural Network

Machine Learning in Chemistry

Machine Learning finds a wide range of application in the various fields of Chemistry owing to the need to process an extremely large amount of data in almost all Chemical processes. Chemists are always very interested in the machines' predictive ability. Drug Discovery has a tremendous amount of data of biological targets and small molecules which are potential drugs, and this data can more than suffice to train algorithms. Artificial Neural Networks is widely used now to predict properties of a molecule based on the known properties of other similar molecules which is then helped to conclude whether that molecule could be a potential drug after analysing all the relevant properties required for a drug. A lot of the other chemical fields listed down have vast usage of Machine Learning.

- *Chemical Graph Theory*

  Molecules are represented as a graph and then deductions are made about their properties using Graph theory and its applications.

- *Cheminformatics*

  This is concerned with storing, indexing, searching, retrieving and applying the information about chemical compounds using pattern recognition and data visualisation.

- *Combinatorial Chemistry*

  This is the field which comprises of the chemical synthetic methods which makes it possible to prepare a huge number of compounds in one single process. These compound libraries could be made as mixtures, structures made through computer softwares and also sets of compounds taken individually.

1.2 Organization of The Report

The following chapter provides a background for high level topics such as Combinatorial Games and the basic concepts involved in combinatorial games. In the Chapter 3, we provide the introduction of the game, Nim and introduce its winning strategy. In Chapter 4, we will discuss our experiment of training different architecture of Neural Networks. We will then analyse the results obtained and will come to the conclusion of the analysis which we aim to do in this thesis. Our aim is to investigate if a machine can identify the winning configurations and the losing configurations of the game of Nim. And finally in Conclusion, we shall conclude with the findings of this thesis and possible scope of the study done in different fields, in Chemistry as well if any.

Chapter 2

Theory

2.1 Combinatorial Games

Combinatorial games are games which have the following features:

1. These games have exactly two players playing the game.
2. They are deterministic, meaning there is no mechanism involving probability or any element of chance such as rolling a die.
3. There is only a finite set of possible positions or game states.
4. The game has perfect information meaning that both of the players know all the information about the game and no information is concealed from any player.
5. Both the players take alternate turns to make their own moves in the game.
6. These are zero sum games, meaning gain of one player is the loss of another. In the end one player wins the game and the other loses.

Some well-known examples of combinatorial games are checkers, chess, Go, tic-tac-toe and Nim.

Games have states in which players perform actions which can lead to new states. A finite game is one which always comes to an end, i.e. no sequence of moves will lead the game to go on infinitely.

Finite Combinatorial Games have the following two game playing conventions:

1. Normal Play Convention: The player who is unable to make a move first is the one who loses.

2. Misère Play Convention: This game has reverse outcome of the normal game, i.e. the last player to move loses.

The combinatorial games are of two kinds listed down:

1. Impartial Games: The games in which there is no difference between the two game players, i.e. it is impartial for the two players implying that the moves available to a player depend only on the position of the game and not on whose turn is it to play and the value of any move is same for both the players. For example, Nim is an impartial game.

2. Partisan Games: The games which are not impartial meaning that there are some moves only available to one player and not to the other player. For instance, in chess only one player can move the white pieces and hence the set of moves is different for him/her than that available to the other player.

2.2 Combinatorial Game Theory

Combinatorial Game Theory is studying of combinatorial games and their sequential moves to deduce a winning strategy for the game.

1. Winning configuration: It is a configuration from which atleast one configuration can be obtained in one single move which is a losing configuration.

2. Losing configuration: It is a configuration from which every configuration obtained is a winning configuration.

The strategy of winning a game mostly boils down to getting to the correct configurations after having identified them correctly.

Sum of Games

If we take the sum of two games which are both impartial games, their sum is also an impartial game. The sum of two games means that on every turn, the player chooses one game and makes a move on that game. The winner is the last player who makes a move. Hence, we can sum games up to get new impartial games.

Sprague-Grundy Theorem

The Sprague Grundy theorem states that every impartial game played with the normal play game convention can be expressed as an equivalent of the game of Nim.

2.3 Choice of game

We study the game Nim in this thesis because according to the Sprague Grundy theorem, we can draw conclusions about other games by converting them into their Nim equivalent version. Another reason is that Nim is a mathematical game which can be completely solved, and hence the optimal strategy is known. When two people are playing the game, the outcome of the game can be determined given only the initial game configuration. This holds true only when both the players are playing optimally when one person doesn't play optimally, we can be sure that the other person will win the game no matter the starting position. Since the optimal strategy is known we can also test the performance of our machine learning model and it becomes a Supervised Learning problem.

Chapter 3

Theory of Nim

3.1 Game of Nim

The game of Nim is a combinatorial two player game in which there are n number of heaps which have any number of elements. The number of elements in these heaps, and hence the initial game configuration can be expressed as an n-tuple. The game proceeds with the two players taking turns at selecting a heap and then removing all or some of the elements of that heap. Any game state in Nim can be expressed by the number of elements present in all the heaps the game began with. The mathematical theory of Nim describes an optimal way to win the game which would determine the outcome of the game given only the initial configuration. This assumes that both the players will play optimally, if that is not the case, the optimal player shall always win.

Nim Sum

We define the Nim Sum operator as the operator $\oplus$ which is the XOR (exclusive OR) operator. The Nim Sum of a state in a game of Nim is the XOR of all the number of elements in all the heaps. Classification of **P** and **N** positions in Nim according to the winning strategy of the game is listed as follows:

1. P Position: A position belongs to P position if its nim-sum is zero which is basically all the winning positions.

2. N Position: A position belongs to N position if its nim-sum is not zero which is basically all the losing positions.

3.2 Optimal Strategy of Nim

For instance, let's consider a game of 3 heaps with heap sizes as 3, 4 and 5 as the initial configuration. The binary conversions of 3, 4 and 5 is done and XOR is computed as illustrated.

$$\begin{array}{rr|ccc} & 3 & 0 & 1 & 1 \\ & 4 & 1 & 0 & 0 \\ \oplus & 5 & 1 & 0 & 1 \\ \hline & 2 & 0 & 1 & 0 \end{array}$$

The nim-sum of 2 indicates that 2 should be the optimal move of this player. The player decides to take away 2 from the first heap and hence the new configuration becomes 1, 4 and 5. So, this was a winning configuration for the player.

$$\begin{array}{rr|ccc} & 1 & 0 & 0 & 1 \\ & 4 & 1 & 0 & 0 \\ \oplus & 5 & 1 & 0 & 1 \\ \hline & 0 & 0 & 0 & 0 \end{array}$$

Here, Nim sum is 0, hence this next state becomes a losing configuration for the opponent as now any move will make the nim sum non zero.

The optimal strategy of the game of Nim is based on a simple mathematical theory. Optimal strategy of the game is to keep making the Nim sum 0 i.e the player must always try to make such a move so that the XOR of all the heap sizes shall become 0. When the Nim sum is non zero, then it is a winning position for the current player as s/he can play a move to make the Nim sum 0, which becomes a losing position for the opponent as then any move played by this opponent will now make the nim sum again non zero allowing the other player to use the next move to make the nim sum 0 again and this cycle continues. Hence given only the initial configuration, there is only one way to play the game with one winner always, provided that both the players keep playing optimally. If the initial configuration has nim sum 0 then the first player loses the game whereas if the initial configuration has nim sum non-zero, the first player wins the game.

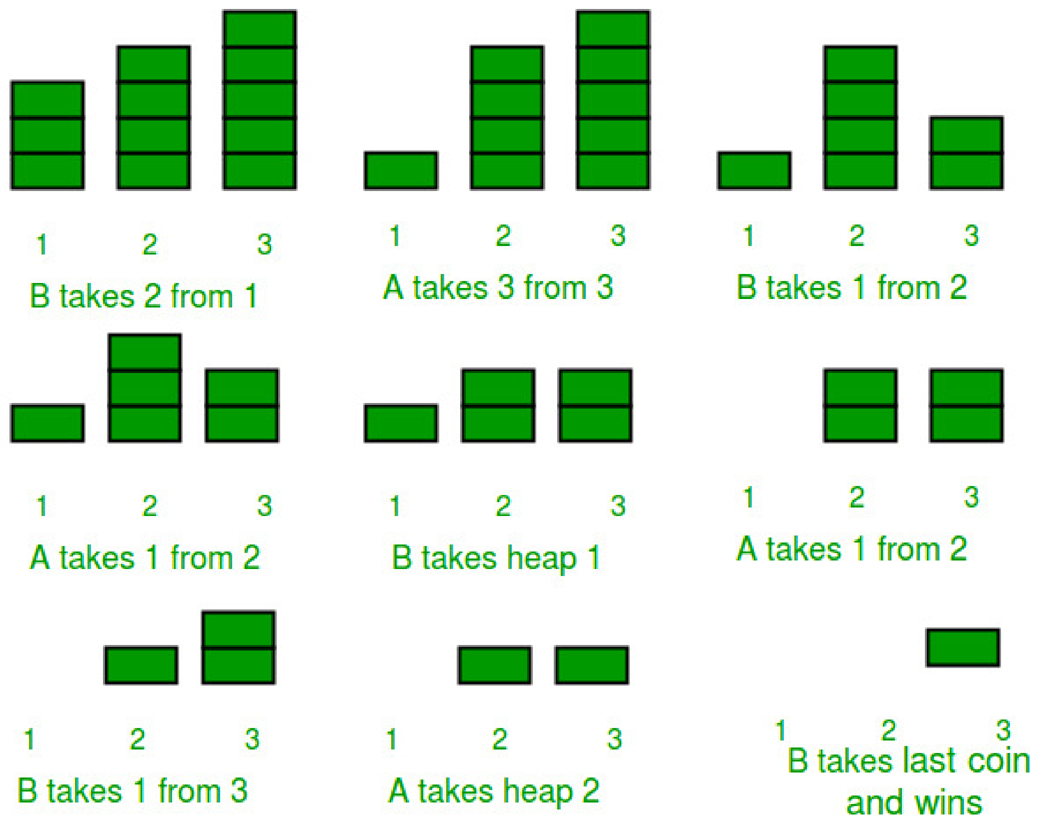
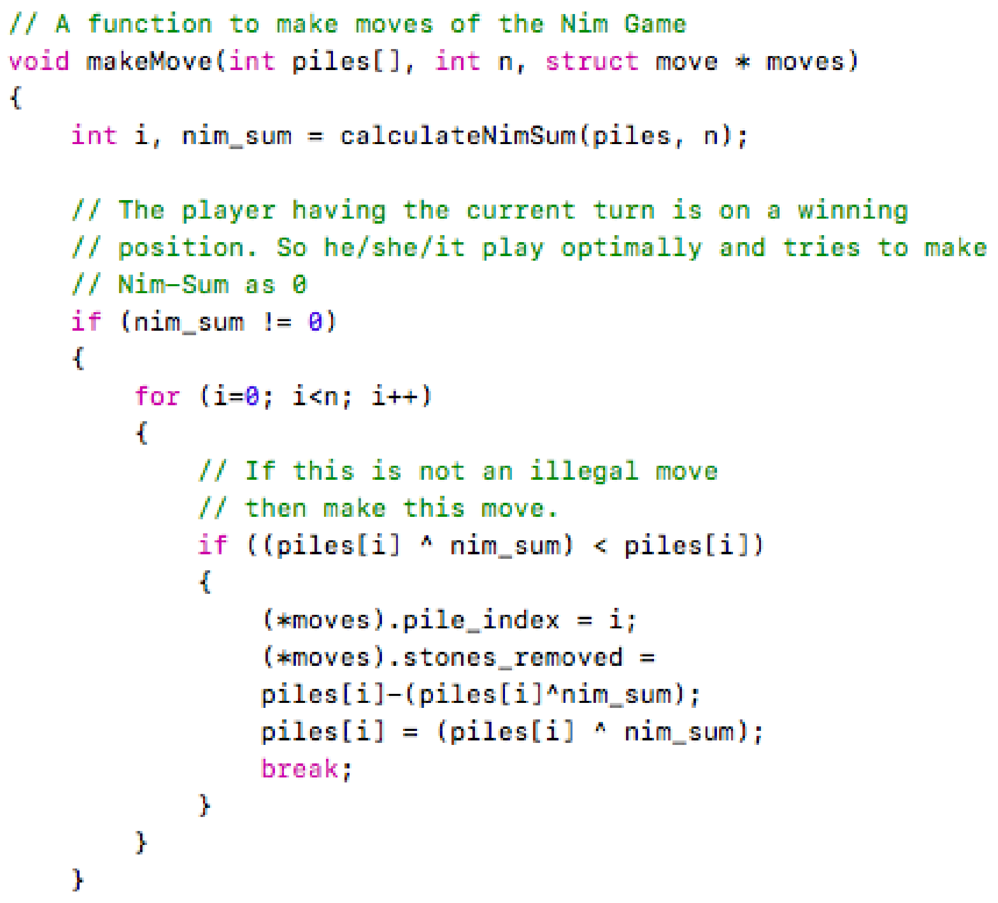


Fig. 3.1 Illutration of playing Nim optimally

Following is a code written in C++ in which the moves have been explicitly programmed according to the winning strategy of the game and it tells whether a given initial configuration is winning configuration or not. Basically the computer is made to play with itself and both the moves are optimal moves and eventually we get a result saying whether the player who started the game has won or the player who played second wins, hence helping us to draw a conclusion about the initial configuration being a winning one or not.



```c
// A function to make moves of the Nim Game
void makeMove(int piles[], int n, struct move * moves)
{
    int i, nim_sum = calculateNimSum(piles, n);

    // The player having the current turn is on a winning
    // position. So he/she/it play optimally and tries to make
    // Nim-Sum as 0
    if (nim_sum != 0)
    {
        for (i=0; i<n; i++)
        {
            // If this is not an illegal move
            // then make this move.
            if ((piles[i] ^ nim_sum) < piles[i])
            {
                (*moves).pile_index = i;
                (*moves).stones_removed =
                piles[i]-(piles[i]^nim_sum);
                piles[i] = (piles[i] ^ nim_sum);
                break;
            }
        }
    }
```

Fig. 3.2 Implementation of game strategy

Chapter 4

Training Model and Results

4.1 Experiment Description

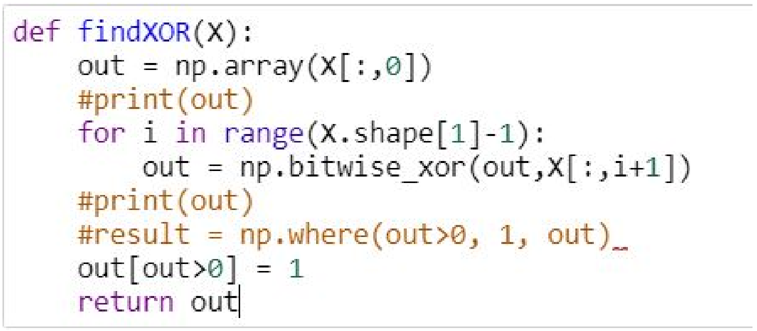
Through various studies carried out by researchers, we know that it is possible to train an agent using reinforcement learning who learns through a feedback mechanism of getting rewards on making a move which makes it win the game but is punished on making a move which causes it to lose a game. The approach of training the RL agent to play the game by learning through its own moves after playing multiple instances of the game, works through a lookup table which consists of a value attributed to every action taken from every step. The problem here is its computational expense because the number of possibilities of moves available to the player at every single state of the game is extremely huge.

If the game of Nim has n heaps with the n^{th} heap having h_n objects:
Total possible actions : $h_0 * h_1 * h_2 * ....h_{n-1}$

Hence the lookup table will become extremely huge having a value associated with every possible action at every state. Hence, our problem statement is to analyse if we can classify the winning and losing configurations of Nim. The aim is to understand if a neural network can be trained to predict the winning and losing configurations of the game of Nim. Since we already know the optimal strategy of the game of Nim, we can use the game solution to our advantage. We can test this as it becomes a simple instance of Supervised Learning. We already know from the previous chapter that the Nim sum helps us classify a configuration as winning or losing. Since Nim sum is calculated using XOR, our final aim becomes to decipher if a neural network can be made to learn the XOR function. If we are able to make a neural network learn the XOR behaviour we can conclude that the classification of winning and losing configurations can be done. We aim to find the neural network architecture which gives the best performance in prediction of winning and losing configurations.

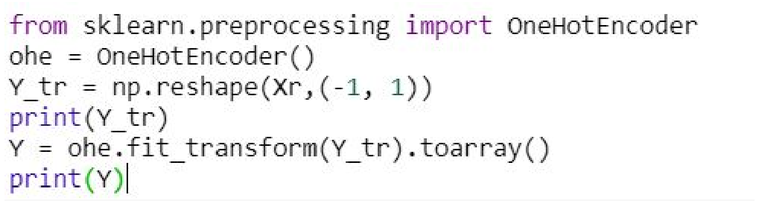
4.2 Training of Neural Network

To implement the experiment, Python 3.0 is used in Jupyter notebook. We use keras with tensorflow as backend to specify the network. We aim to find which neural network gives the best out-of-sample performance i.e. best performance on configurations it hasn't seen before. For training we need to make sure that the training data set is not biased. When random triplets are generated, most of them will be winning configurations because their XOR would be non zero, and very low percentage of these triplets will have XOR value as zero indicating losing configurations. The Nim sum which is the XOR of all three heap sizes is computed using the function whose code snippet is shown below. If the XOR value is greater than 0, 1 is returned, else if the XOR is 0, 0 is returned.



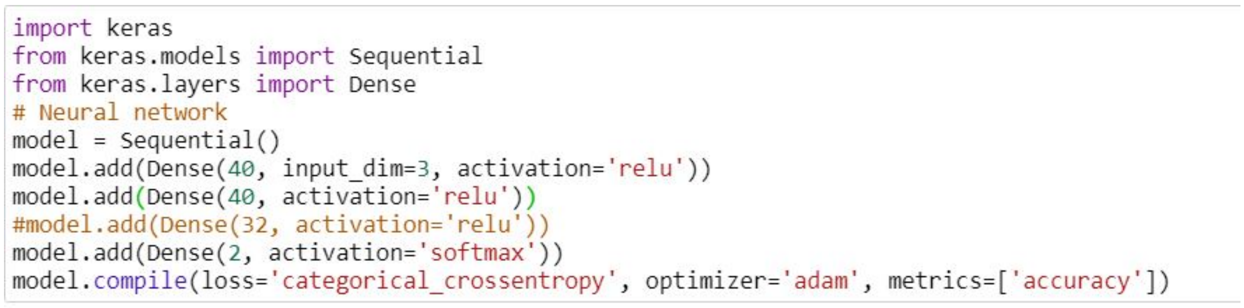
```
def findXOR(X):
    out = np.array(X[:,0])
    #print(out)
    for i in range(X.shape[1]-1):
        out = np.bitwise_xor(out,X[:,i+1])
    #print(out)
    #result = np.where(out>0, 1, out)
    out[out>0] = 1
    return out
```

We need to make sure the training data set is not biased. After generating 10,000 random triplets, all the triplets having XOR value 0 are picked up and then equal number of triplets having XOR value 1 are picked up which constitutes our training set. In one instance of the above method followed, this training set formed is of size 2506 with 1253 triplets having XOR 0 and 1253 triplets having XOR greater than 0. Expected output is 0 when XOR of the triplets is 0, i.e. a losing configuration and is 1 when XOR of the triplets is not zero i.e. a winning configuration. One hot encoding is done on the expected output of the training data set. The code snippet is attached.

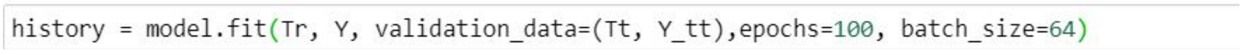


```
from sklearn.preprocessing import OneHotEncoder
ohe = OneHotEncoder()
Y_tr = np.reshape(Xr,(-1, 1))
print(Y_tr)
Y = ohe.fit_transform(Y_tr).toarray()
print(Y)
```

The model is trained on different architecture of neural networks. The number of layers to be used and the number of nodes in each layer is specified by the user. A layer is added using *model.add* and specifying the number of nodes in that particular layer. The activation function used here is $ReLU$, rectified linear unit. Mathematically it is defined as $y = max(0, x)$. Finally, *softmax* is used as the activation function. XOR 0 indicated by [1 0] and XOR 1 indicated by [0 1] because of one hot encoding. The code snippet defining the model with 2 layer having 40 nodes each is shown below.



```
import keras
from keras.models import Sequential
from keras.layers import Dense
# Neural network
model = Sequential()
model.add(Dense(40, input_dim=3, activation='relu'))
model.add(Dense(40, activation='relu'))
#model.add(Dense(32, activation='relu'))
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```



```
history = model.fit(Tr, Y, validation_data=(Tt, Y_tt),epochs=100, batch_size=64)
```

The model defined above is fitted with training data set denoted as Tr having the expected output as Y. The trained model is checked on validation data set denoted here by Tt having the expected output Y_tt. Epochs and batch size is also specified.

4.3 Results

We must define the game of our experiment. In our experiment, there are 3 heaps with maximum allowed heap size as 20. The experiment is repeated with changing the architecture of the neural network specified. On changing the number of layers and the number of nodes in each layer, the training data set accuracy and test data set accuracy of the model is noted. The accuracy of training and test set is depicted in the table below.

NN Architecture	**Training data set accuracy**	**Test data set accuracy**
5-5	0.7282	0.6810
12-12	0.8018	0.7540
12-12-12	0.7477	0.7610
20-20	0.8483	0.7850
20-20-20	0.8529	0.7690
26-26	0.8514	0.8150
26-26-26	0.8498	0.8020
31-31	0.8438	0.7550
32-32	0.8483	0.7890
32-32-32	0.8529	**0.8390**
33-33	0.8168	0.7690
40-40	0.8634	0.7970
40-40-40	0.8589	0.7790
50-50	0.8529	0.8020

Looking at the observations from the table we can conclude that 32-32-32 neural network shall be the best in predicting winning configurations of the game of 3 heaps having maximum value 20 of each heap. To conclude a model's performance we need to look at its performance on out-of-sample data, the configurations which it hasn't seen before. The in-sample accuracy just tells us how good the model is trained on the subset of all possible configurations which it sees. To verify that 32-32-32 architecture of the neural network which gives the best prediction to us, the experiment was repeated by increasing the number of nodes but the accuracy is seen to be decreasing. In general we expect the network's accuracy to increase with increasing nodes and layers. From the table we can observe 32-32-32 is the best fit architecture as the accuracy decreases on both increasing or decreasing the number of layers or nodes in each layer. The out-of-sample accuracy for this network is highest at 83.9% whereas in-sample accuracy is 85.29%. We can also notice that the in-sample accuracy and out-of-sample accuracy don't have a large difference. Hence it seems fair to conclude that the network generalises sufficiently well and it also predicts the configurations which it has not seen before i.e. which haven't been part of training the model also almost equally well as the configurations which have been explicitly used in training the network.

Further analysis is done on accuracy of the model by analysing the accuracy of correct classifications in winning configurations and the accuracy of correct classifications in losing configurations separately. We look at the number of correct and incorrect classifications from within the winning and losing configurations. 1000 random triplets are generated to check the out-of-sample performance separately in winning and losing configurations. Out of these, 953 were winning configurations i.e. they had their XOR non-zero, and the rest 47 were losing configurations having XOR 0. We have restricted this study to only two architectures, 32-32 and 32-32-32 because of our above conclusion that these networks perform best. In the performance of 32-32-32 architecture, we can observe that the model's performance in correctly classifying both winning and losing configurations is similar, being 83.95% and 82.98% respectively. Hence we can say that 32-32-32 architecture is better than 32-32 because in the 32-32 architecture the network seems to be performing better in the losing configurations at the cost of its performance within the winning configurations. One more important point to keep in mind here is that since the percentage of losing configurations is so small, the number of losing configurations on which we are testing the model is too small to draw any conclusion about the accuracy of the network's performance within losing configurations. It should be just taken as an estimate and not be taken as an accurate depiction of model's performance on losing configurations. The table with observed data is given below.

NN Arc.	Correctly classified	Incorrectly classified	Total	Accuracy
32-32	743	210	Winning : 953	0.7796
	43	4	Losing : 47	0.9149
32-32-32	800	153	Winning : 953	0.8395
	39	8	Losing : 47	0.8298

Another study is done on varying heap sizes. If the maximum heap size is reduced to 7, we get a better performance of the neural network, but if the maximum heap size is increased to 50, the accuracy decreases. As we know that when the maximum allowed heap size of heaps is increased, the possible number of actions which are available to the player increases by a lot. Hence it is fair to conclude that as the heap sizes are increased, the accuracy of predication of a particular neural network decreases. This observation can be verified in the table below.

NN Architecture	Max heap size	Test data set accuracy
32-32	7	0.8970
	20	0.7890
	50	0.7670
32-32-32	7	0.9000
	20	0.8390
	50	0.7050

Chapter 5

Conclusion

Through the experiment and analysis conducted, the following are important conclusions and facts which we have analysed.

1. The neural network architecture 32-32-32 seems to work best for predicting if a configuration is winning or losing in the game of Nim when the number of heaps are fixed at 3, it converges to an accuracy of 83% when the maximum size of each heap is 20 and to 90% when the maximum heap size is 7.

2. As the maximum heap size is increased we notice the accuracy of prediction of the neural network decreases. This can be attributed to the huge increase in the possible number of moves now available with the player.

3. We have also verified that a good model has similar accuracy of correct classification in both the classes, in this case the 32-32-32 architecture has comparable accuracy of correctly classified configurations in winning and losing configurations when analysed separately.

4. We have seen that the accuracy is comparable for the training data set and the test data set, hence we can conclude that the performance of our neural network will be almost similar whether it encounters a configuration it has seen before or it hasn't.

5. The fact that our model does not perform badly on positions which it hasn't seen before is proof that there hasn't been any over-fitting and the model has generalised to a good extent.

6. Nim is a solved game with an optimal strategy and hence the study of winning and losing configurations reduces to an instance of supervised learning.

7. One important precaution in this experiment which must be taken is having a good mix of winning and losing positions in the training data set used to train the network because if a randomly generated sample is used then almost 90% of the samples have XOR value as non zero, so if that is used to train the model then the model will perform extremely good in classifying winning positions but it won't be able to identify any losing configuration but the accuracy given would still be around 90%. This is not a desired model.

8. Hence, we can conclude that there exists a neural network which can classify the winning and losing configurations of Nim up to a satisfactory extent. It is also fair to conclude that the 32-32-32 neural network comes very close to learning the XOR behaviour.

On an ending note, if we have succeeded in predicting winning and losing configurations of Nim and in the past researchers have been able to make machines play Go and defeat the world's best player, there sure is a possibility of doing that for other games as well. If machines could be made to learn to play games, similar path could be taken up for training machines for classification problems in chemistry and eventually, we can have machine learning algorithms classify whether a molecule is suitable for a specific purpose maybe also identifying if it can be used as a cancer drug. This topic is ever evolving and would save chemists and pharmaceutical firms big bucks and time, because all the tremendous amount of molecules and chemicals won't be needed to be handled and experimented with, manually.

ACKNOWLEDGEMENTS

I am extremely thankful to Prof. Lal Mohan Kundu, Department of Chemistry, for giving me the opportunity to carry out my Bachelors' thesis Project in collaboration with the Department of Computer Science and Engineering. I would like to acknowledge the Department of Computer Science and Engineering for providing me the facilities that were necessary to pursue my project. I am also thankful to all the faculty members for their constant assistance and support. I am grateful to the God for the good health and well-being that were necessary to complete this report. I would like to express my sincere gratitude to my supervisor Prof. Benny George for his guidance, inspiration, priceless suggestions and support, which helped me to carry out my project. He has been extremely supportive and encouraging even during this ongoing pandemic. I also thank my family for the unceasing encouragement, affection, mental, financial support and attention without which I could not have achieved this during the ongoing uncertain scenario. I am also grateful to my friends who supported me throughout this venture. I also place on record, my sense of gratitude to everyone, who directly or indirectly, have lent his or her hand in this venture.

Jigyasa

160122016

Bachelor of Technology, Final year Chemical Science and Technology

Indian Institute of Technology Guwahati.

References

[1] William Lord and Paul Graham. Reinforcement learning and the game of nim, 2015.

[2] Erik Järleberg. Reinforcement learning on the combinatorial game of nim, 2011.

[3] Erik D Demaine. Playing games with algorithms: Algorithmic combinatorial game theory. In *International Symposium on Mathematical Foundations of Computer Science*, pages 18–33. Springer, 2001.

[4] Richard K Guy. Unsolved problems in combinatorial games. In *Combinatorics Advances*, pages 161–179. Springer, 1995.

[5] https://arthought.com/game-of-nim-supervised-learning/.

[6] https://web.stanford.edu/class/cs97si/05-combinatorial-games.pdf.

[7] https://cen.acs.org/physical-chemistry/computational-chemistry/machine-learning-overhyped/96/i34.

[8] https://brilliant.org/wiki/combinatorial-games-winning-positions/.

[9] https://brilliant.org/wiki/sprague-grundy-theorem/.

[10] http://www.numpy.org/.

[11] https://cocalc.com/.

BTP_Report

ORIGINALITY REPORT

6%	1%	2%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to The University of Manchester Student Paper	1%
2	Submitted to University of Strathclyde Student Paper	1%
3	Submitted to School of the Arts, Singapore Student Paper	<1%
4	Submitted to Stuyvesant High School Student Paper	<1%
5	"Sources, properties, recovery", Fuel and Energy Abstracts, 200209 Publication	<1%
6	brilliant.org Internet Source	<1%
7	Submitted to Republic Polytechnic Student Paper	<1%
8	www.itproportal.com Internet Source	<1%
9	www.onemachinelearning.com	