CS362: Artificial Intelligence Laboratory Report - 2

by

Group Members
Anirudh Mitra[1]
Chirag Jain[2]
Yash Sakle[3]
&
Student ID
201951024[1]
201951049[2]
201951177[3]

Code Repository: Github.

Submitted To: Prof. Pratik Shah

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY VADODARA

May 2022

Bayesian Network in R & Structure Learning

Anirudh Mitra, *201951024, B.tech (CSE)*

***Abstract*—Understanding the graphical models for inference under uncertainty, building Bayesian Network in R, Learning the structure and CPTs from Data, Naive Bayes classification with dependency between features.**

I. Introduction

A. Problem

- Consider grades earned in each of the courses as random variables and learn the dependencies between courses.
- Using the data, learn the CPTs for each course node.
- What grade will a student get in PH100 if he earns DD in EC100, CC in IT101 and CD in MA101.
- The last column in the data file indicates whether a student qualifies for an internship program or not. From the given data, take 70 percent data for training and build a naive Bayes classifier (considering that the grades earned in different courses are independent of each other) which takes in the student's performance and returns the qualification status with a probability. Test your classifier on the remaining 30 percent data. Repeat this experiment for 20 random selection of training and testing data. Report results about the accuracy of your classifier.
- Repeat 4, considering that the grades earned in different courses may be dependent.

B. Theory

*1) **Bayesian Network**:* A Bayesian network is a probabilistic graphical model(joint probability distributions that can be factored according to a graph) that measures the conditional dependence structure of a set of random variables based on the Bayes theorem:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

The causal probability relationships between variables can be used to help us with decision making. The inter-variable dependence structure is represented by nodes (which depict the variables) and directed arcs (which depict the conditional relationships) in the form of a directed acyclic graph (DAG) [3]. Independent features have no connection between them while dependent variables are connected using the directed arrow where, the intuitive meaning of an arrow from X to Y is typically that X has a direct influence on Y, which suggests that causes should be parents of effects [1].

*2) **R and bnlearn Package**:* R language along with the bnlearn package is used for modelling and analysis of the given problem. bnlearn is an R package for learning the graphical structure of Bayesian networks, estimate their parameters and perform some useful inference. We use it for operations like learning dependencies between the features, learning CPTs and building Naive Bayes classifier for classifying students based on their grades.

*3) **CPT**:* The local probability information attached to each node in a Bayes Net takes the form of a conditional probability table (CPT). (CPTs can be used only for discrete variables; other representations, including those suitable for continuous variables). Each row in a CPT corresponds to the conditional probability of each node value for a conditioning case. A conditioning case is just a possible combination of values for the parent nodes[1].

*4) **Naive Bayes with Dependent Features**:* In Naive Bayes, the assumption is that features are independent, which in real life applications, is not true. Features are interrelated. In this experiment, we've used Chow-Liu Algorithm for structure learning and maximizing the likelihood.

II. Approach

A. Dependency Learning

For learning the dependencies between the features, the hill-climbing score based structure learning algorithm provided by the bnlean package is used. Since its a score-based algorithm, we need two scores so that the structure could be learned using those: 1. Bayesian information criterion (BIC) score:

$$BIC = kln(n) - 2ln(\hat{L})$$

where, $\hat{L}$ is the maximized value of likelihood function; n is the number of data points; k is the number of parameters estimated by the model.

2. K2 score.

B. CPT Learning

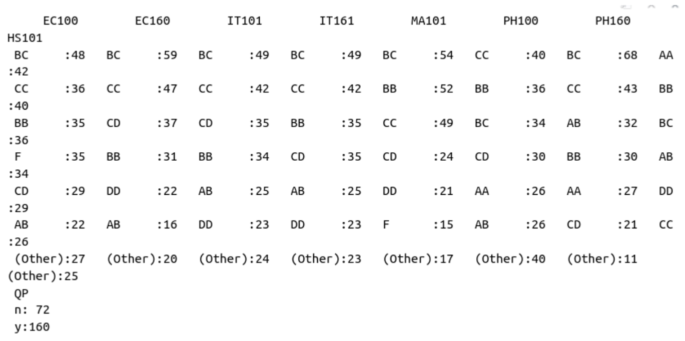
After learning the underlying structure of the network, we can use the conditional probabilities to construct the CPTs.

C. Finding Grade

The cpdist function which provides Non-parametric test for change-point detection based on the (multivariate) empirical distribution function is used for finding corresponding distribution probability of PH100 grades given grades of EC100, IT101 and MA101. The probability distribution will give the most likely grade.

III. Results

Using the results that we acquired through plots, probability distributions and tables, we can form our conclusions and connect them to validate each other.



```
     EC100        EC160        IT101        IT161        MA101        PH100        PH160
HS101
 BC     :48   BC     :59   BC     :49   BC     :49   BC     :54   CC     :40   BC     :68   AA
:42
 CC     :36   CC     :47   CC     :42   CC     :42   BB     :52   BB     :36   CC     :43   BB
:40
 BB     :35   CD     :37   CD     :35   BB     :35   CC     :49   BC     :34   AB     :32   BC
:36
 F      :35   BB     :31   BB     :34   CD     :35   CD     :24   CD     :30   BB     :30   AB
:34
 CD     :29   DD     :22   AB     :25   AB     :25   DD     :21   AA     :26   AA     :27   DD
:29
 AB     :22   AB     :16   DD     :23   DD     :23   F      :15   AB     :26   CD     :21   CC
:26
 (Other):27   (Other):20   (Other):24   (Other):23   (Other):17   (Other):40   (Other):11
(Other):25
 QP
 n: 72
 y:160
```

Fig. 1: Summary of the Data

A. Dependency Learning

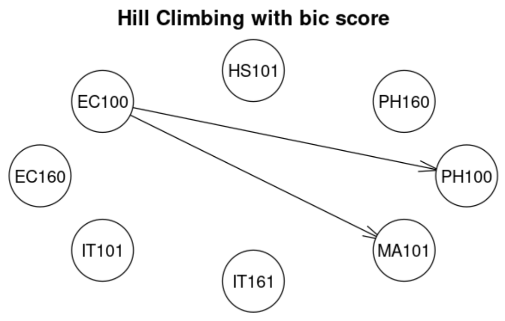


Fig. 2: Structure Plot by Hill Climbing(BIC Score)

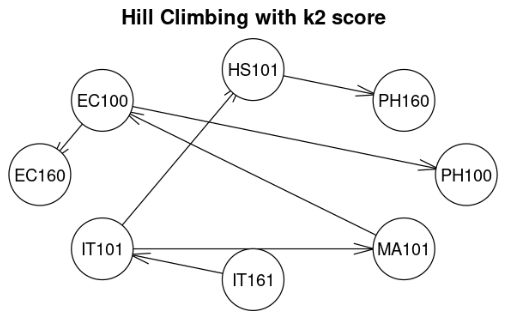


Fig. 3: Structure Plot by Hill Climbing(K2 Score)

B. CPT Learning

C. Finding Grade

D. Naive Bayes

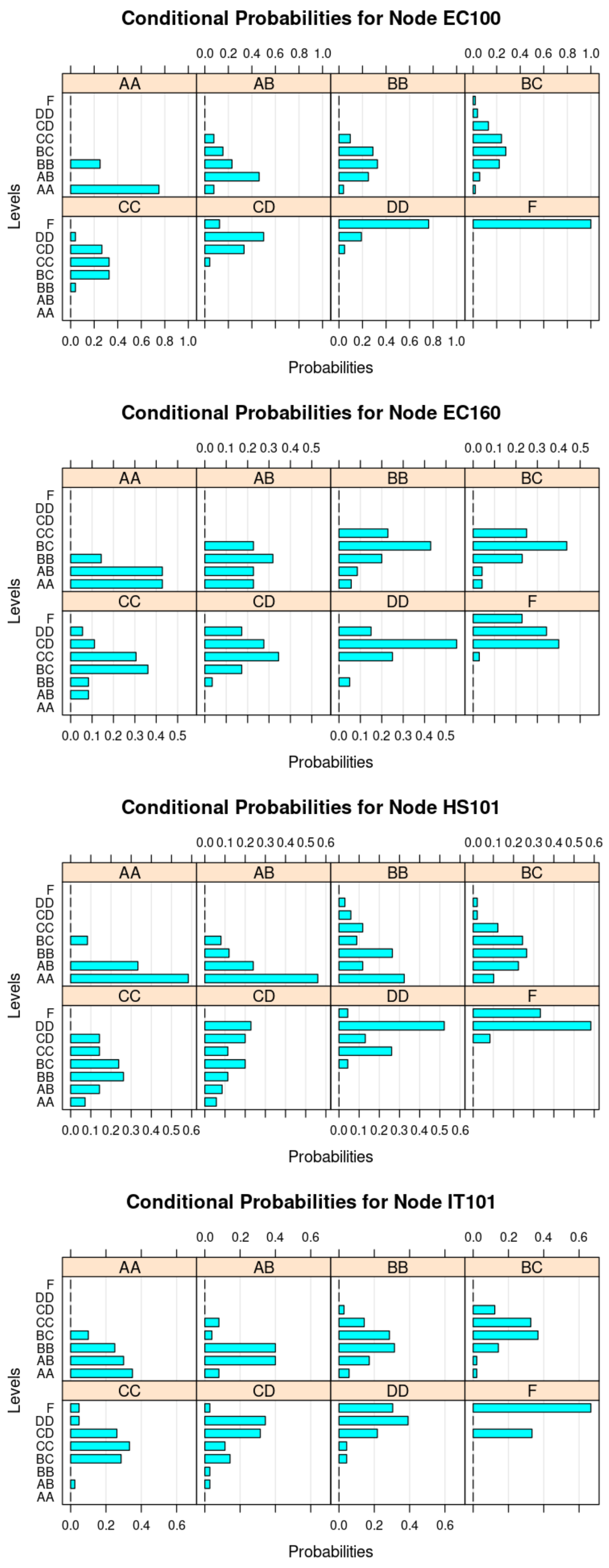


Fig. 4: Conditional Probability Tables for EC100, EC160, HS101 & IT101

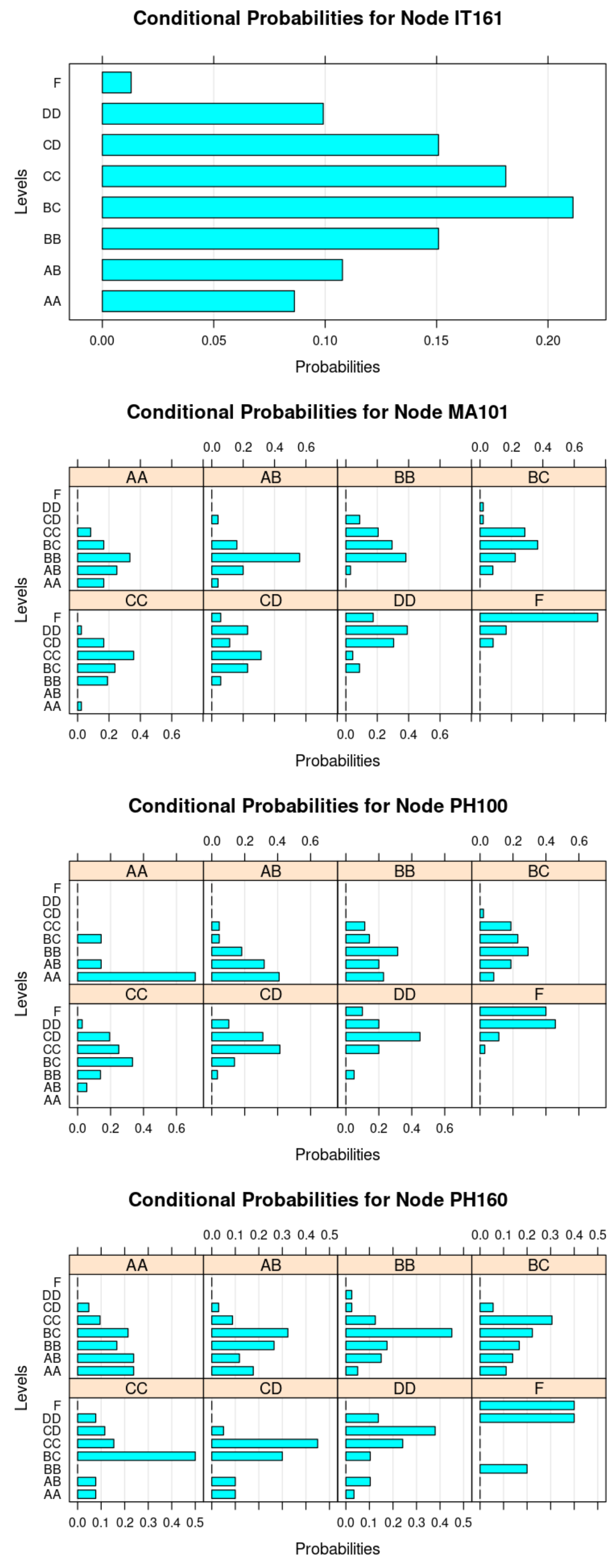


Fig. 5: Conditional Probability Tables for IT161, MA101, PH100 & PH160

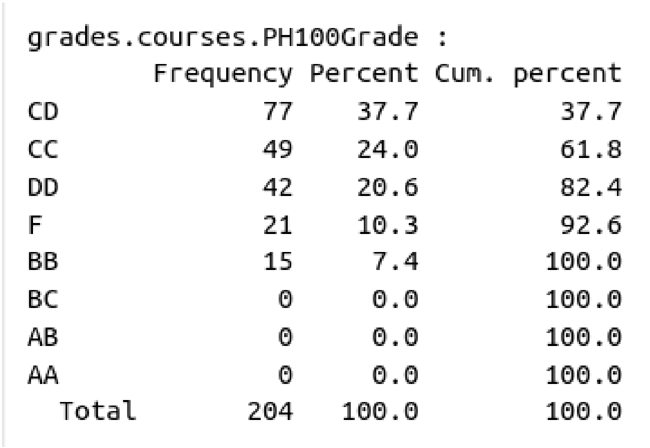

```
grades.courses.PH100Grade :
         Frequency Percent Cum. percent
CD              77    37.7         37.7
CC              49    24.0         61.8
DD              42    20.6         82.4
F               21    10.3         92.6
BB              15     7.4        100.0
BC               0     0.0        100.0
AB               0     0.0        100.0
AA               0     0.0        100.0
  Total        204   100.0        100.0
```

Fig. 6: PD of PH100

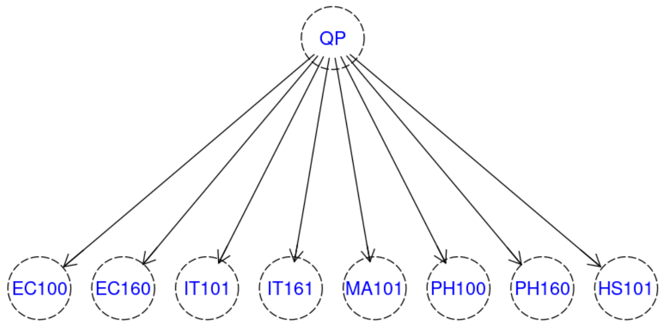


Fig. 7: Naive Bayes with Independent Features

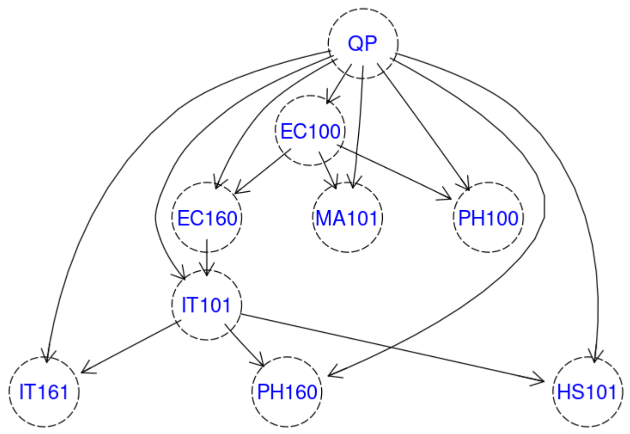


Fig. 8: Naive Bayes with Dependent Features

References

[1] "Artificial Intelligence: A Modern Approach, 4th US ed.," Berkeley.edu, 2021. http://aima.cs.berkeley.edu/
[2] Khemani, Deepak. A First Course in Artificial Intelligence. India: McGraw Hill Education (India), 2013.
[3] X.-S. Yang, "Mathematical foundations," Introduction to Algorithms for Data Mining and Machine Learning, pp. 19–43, 2019, doi: 10.1016/b978-0-12-817216-2.00009-0.
[4] [1] B. Mihaljević, "Bayesian networks with R," 2018. Accessed: May 01, 2022. [Online]. Available: https://gauss.inf.um.es/umur/xjurponencias/talleres/J3.pdf

Exploitation - Exploration in Simple Multi-Armed Bandit, Epsilon-Greedy Algorithm

Chirag Jain, *201951049, B.tech(CSE)*

***Abstract*—In this paper we will understanding the Exploitation - Exploration in simple n-arm bandit reinforcement learning task. The n-armed bandit problem is used in reinforcement learning to formalize the notion of decision-making under uncertainty. Further, we'll understand the Epsilon-Greedy Algorithm. The goal is to identify which action to choose to get the maximum reward after a given set of trials.**

I. Introduction

Reinforcement learning problems involve learning what to do—how to map situations to actions—so as to maximize a numerical reward signal. The learner is not told which actions to take, as in many forms of machine learning, but instead must discover which actions yield the most reward by trying them out. For that the agent has to exploit what it already knows in order to obtain reward, but it also has to explore in order to make better action selections in the future. So the user have to choose between exploration and exploitation.

In the first problem statement we are given with a binary bandit in which we have two rewards and two stationary actions. In the second problem statement we are given with 10-armed bandit normally distributed increment with mean zero. We have used epsilon-greedy algorithm since the rewards are non-stationary.

II. Theory

The **n-armed bandit** problem is used in reinforcement learning to formalize the notion of decision-making under uncertainty. In a multi-armed bandit problem, an agent chooses between n different actions and receives a reward based on the chosen action. The multi-armed bandits are also used to describe fundamental concepts in reinforcement learning, such as rewards, timesteps, and values. For selecting an action by an agent, we assume that each action has a separate distribution of rewards and there is at least one action that generates maximum numerical reward. Thus, the probability distribution of the rewards corresponding to each action is different and is unknown to the agent. Agent will always try to get the maximum reward.

Epsilon-Greedy Algorithm is a simple algorithm to balance exploration and exploitation by choosing between exploration and exploitation randomly.The epsilon-greedy, where epsilon refers to the probability of choosing to explore, exploits most of the time with a small chance of exploring. Initially it will start with the random action , as time goes on it will get a sense of which choices are returning it with the highest reward.

III. Problem Statements

A. Consider a binary bandit with two rewards 1-success, 0-failure. The bandit returns 1 or 0 for the action that you select, i.e. 1 or 2. The rewards are stochastic (but stationary). Use an epsilon-greedy algorithm discussed in class and decide upon the action to take for maximizing the expected reward. There are two binary bandits named binaryBanditA.m and binaryBanditB.m are waiting for you.

Here we cam observe that initially reward were low fro the bandit-A and is close to 0. As the number of trial increase we can see that the reward is getting higher with increase of trial. As in case of bandit-B the probability of the reward is high and is close to one so the expected reward is also close to one. When we keep increasing the trial the reward become constant due to averaging factor and the probability will also approach to one .

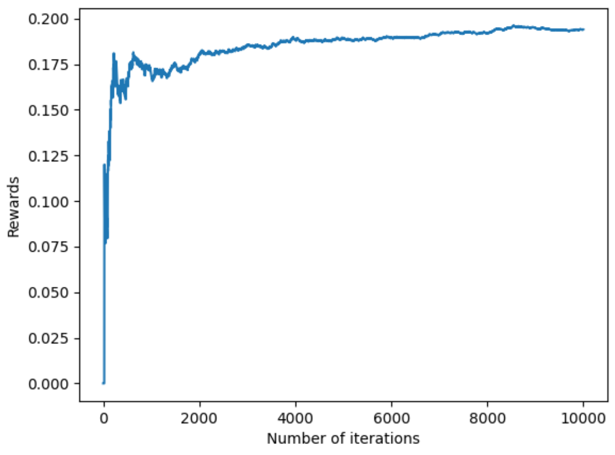


Fig. 1

B. Develop a 10-armed bandit in which all ten mean-rewards start out equal and then take independent random walks (by adding a normally distributed increment with mean zero and standard deviation 0.01 to all mean-rewards on each time step). function [value] = bandit$_nonstat(action)$

In this case there are 10 arm-bandits given that the initial mean-reward are constant and initialised by one. Exploration and exploitation were performed based on the Epsilon Greedy algorithm.A random number between 0 and 1 is generated, if it is greater than epsilon it will perform an exploit based on based on prior knowledge, otherwise exploration. For every iteration an array of ten value is generated with mean 0 and standard deviation 0.01. This array is added to the mean array and this updated array is used every time. For every action a non stationary reward is given from this updated mean-reward array.

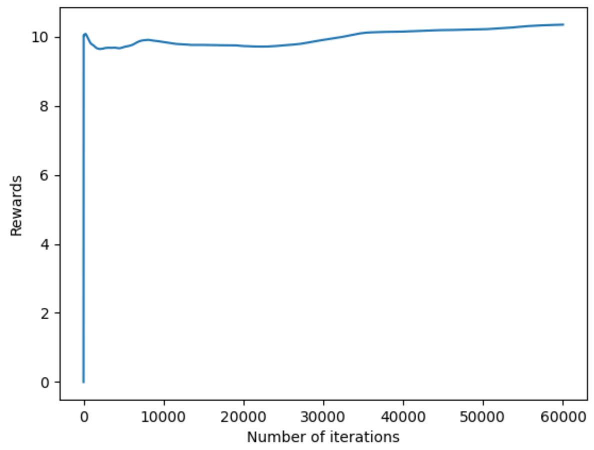


Fig. 2

C. *The 10-armed bandit that you developed (banditnonstat) is difficult to crack with standard epsilon-greedy algorithm since the rewards are non-stationary. We did discuss about how to track non-stationary rewards in class. Write modified epsilon-greedy agent and show whether it is able to latch onto correct actions or not.*

This problem is same as the above problem the only difference is that in above problem we used averaging method to update estimation of action reward was used whereas here more weight is given to current earned reward by using alpha parameter.

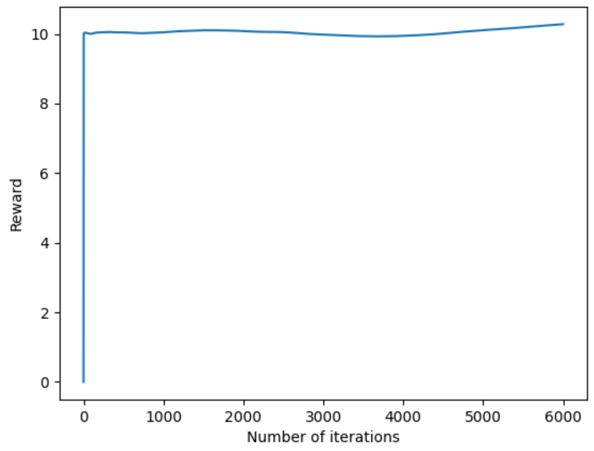


Fig. 3

IV. CONCLUSION

Exploitation is the right thing to do to maximize the expected reward on the one step, but exploration may produce the greater total reward in the long run. Reward is lower in the short run, during exploration, but higher in the long run because after you have discovered the better actions, you can exploit them many times. The Epsilon Greedy on the other hand, explores too much because even when one action seem to be the optimal one, the methods keeps allocating a fixed percentage of the time for exploration, thus missing opportunities and increasing total regret.

REFERENCES

[1] Parkinson, Avery. 2019. “The Epsilon-Greedy Algorithm for Reinforcement Learning.” Medium. Analytics Vidhya. December 2, 2019. https://medium.com/analytics-vidhya/the-epsilon-greedy-algorithm-for-reinforcement-learning-5fe6f96dc870.

[2] Sutton, H. 2014. “Peter Morgan Sutton.” BMJ 348 (mar31 11): g2466–66. https://doi.org/10.1136/bmj.g2466.

[3] “Epsilon-Greedy Algorithm in Reinforcement Learning - GeeksforGeeks.” 2020. GeeksforGeeks. May 2020. https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/.

MENACE: Matchbox Educable Naughts and Crosses Engine

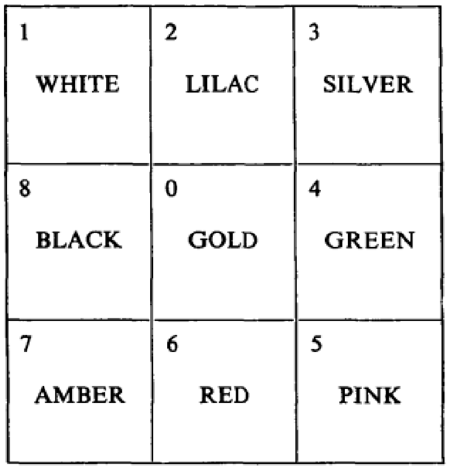
Yash Sakle, *201951177, B.tech(CSE)*

***Abstract*—The goal of this lab is to learn more about MENACE and how it learns by reinforcing its choices. The goal was to show that machines can learn from their mistakes and make better decisions as a result.**

Code Link,

I. Introduction

MENACE stands for Machine Educable Noughts and Crosses Engine. Donald Michie first described it, using matchboxes to record each game he played against this program. This provides a good conceptual foundation for a trial-and-error learning mechanism, as long as the total number of possible choice-points is small enough to be individually listed. Michie wanted to show that a computer could "learn" from both failure and success in order to become proficient at a task.
MENACE works in a similar way to a neural network. It starts off with a random optimization, but after a few games, it changes to favour movements that are reportedly more successful in each case. The success of the model will determine whether it is punished or rewarded.
When training begins, all boxes contain colour-coded beads, where each colour represents a move (or position) on a board.



1 WHITE	2 LILAC	3 SILVER
8 BLACK	0 GOLD	4 GREEN
7 AMBER	6 RED	5 PINK

Fig. 1: Colour Code Used in Matchbox Machine.

II. Approach

When the human player selects a bead from the box that reflects the game's current state at random, MENACE makes a move. MENACE's movement is determined by the colour of the bead. There were beads in some versions of MENACE that only signified more explicit motions, such as the side, centre, or corner.

The beads are chosen at random by the human player, just as the weights of a neural network are chosen at random at the start. When there is failure or success, the beads, like weights, are adjusted. If MENACE loses a game, each bead MENACE utilised is removed from each box at the end of the game. If MENACE wins, three beads of the same colour as the one used during each turn are added to the appropriate box. If the game ends in a tie, one more bead is added.

III. Experiment Result

When MENACE plays against human, the results look like this:

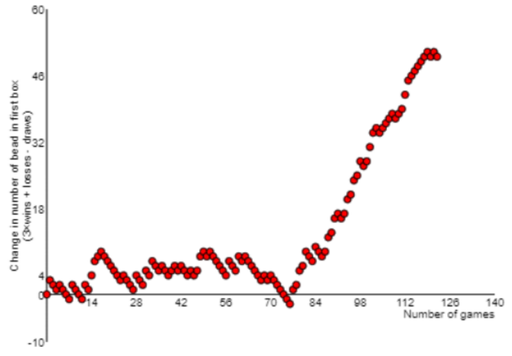


Fig. 2: Playing with Human.

When MENACE played a with a random picking opponent, the result is a near-perfect positive correlation:

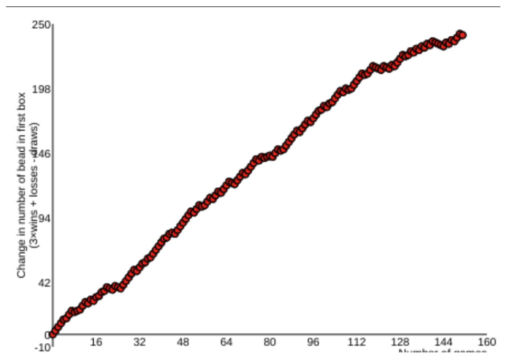


Fig. 3: Playing with Random Picking Opponent[2].

IV. Conclusion

The goal was to show that machines can learn from their mistakes and make better decisions after going through the trial and error process.
As more games are played, we can see how it does against humans. It struggled at first and lost a lot, but after a while, it had a string of draws, before faltering again and losing a few matches until eventually winning.

References

[1] Michie, D. (1963). Experiments on the Mechanization of Game-Learning Part I. Characterization of the Model and its parameters. The Computer Journal, 6(3), pp.232–236.

[2] Science, O.-O.D. (2018). How 300 Matchboxes Learned to Play Tic-Tac-Toe Using MENACE. [online] Medium. Available at: https://odsc.medium.com/how-300-matchboxes-learned-to-play-tic-tac-toe-using-menace-35e0e4c29fc [Accessed 1 May 2022].