Second Lab Report CS-362

April 30, 2022
Submitted to: Prof Pratik Shah

Shivansh Kumar, Deepanshu Singh, Raj
201951144@iiitvadodara.ac.in 201951054@iiitvadodara.ac.in 201951123@iiitvadodara.ac.in
Code Repository: https://github.com/raj-gupta1/AI-lab-assignments

I. Introduction

In this Report, we have summarized our experiments, observations and results while performing 8 experiments given to us in the labs. We chose the following experiments for this report:

1) Lab Assignment 6: Hidden Markov Model and Expectation Maximization algorithm
2) Lab Assignment 7: Markov Random Field and Hopfield Network
3) Lab Assignment 8: Markov Decision Process and RL
4) Lab Assignment 9: Understanding Exploitation and n-armed bandit reinforcement learning

Visualizing our results through tables and graphs helped us understand the results in a better and more convenient way, as well as formulate our results better. We hope the following discussion in the sections below help the reader understand these topics in a better light that what the they started with.

We performed the experiments primarily in Google Colab for Python Codes and RStudio for R. This report mainly discusses how we approached the problem, and the observations and results that we got from it. We have added links to our codes in this doc under the specific session, as well as at the end of this section.

II. Week 6 Lab Assignment 6

Learning Objective: To implement Expectation Maximization routine for learning parameters of a Hidden Markov Model, to be able to use EM framework for deriving algorithms for problems with hidden or partial information.

Problem Statement: The Google Drive folder with thecodes and datasets can found **here**.

A. PART A

Read through the reference carefully. Implement routines for learning the parameters of HMM given in section 7. In section 8, "A not-so-simple example", an interesting exercise is carried out. Perform a similar experiment on "War and Peace" by Leo Tolstoy.

From the reference "A Revealing Introduction to Hid- den Markov Models" [16], we calculated α-pass, β-pass, di-gammas for re-estimating the state transition probabilitymatrix (A), observation probability matrix (B), initial state distribution (π) for Leo Tolstoy's book War and Peace. We re-estimated A,B and π based on the observed sequence , taking initial values for A, B and π and calculating α, β, the di-gammas and log probability for the data i.e. War and Peace. We took 50000 letters from the book after removing all the punctuation and converting the letters to lower case just as given in the example problem in section 8 [16]. We initialized each element of π and A randomly to approximately 1/2. The initial values are:

$$\pi = \begin{matrix} 0.51316 & 0.48684 \end{matrix}$$

$$A = \begin{matrix} 0.47468 & 0.52532 \\ 0.51656 & 0.48344 \end{matrix}$$

Each element of B was initialized to approximately 1/27. The precise values in the initial B are given in the Tab IX.

After initial iteration,

$$log([P(O|\lambda)]) = -142533.41283009356$$

After 100 iterations,

$$log([P(O|\lambda)]) = -138404.4971796029$$

This shows that the model $\lambda = (A, B, \pi)$ has improved significantly. After 100 iterations the model converged to

$$\pi = \begin{matrix} 0.00000 & 1.00000 \end{matrix}$$

$$A = \begin{matrix} 0.28438805 & 0.71561195 \\ 0.81183208 & 0.18816792 \end{matrix}$$

with final values of transpose of B in the Tab IX.

By looking at the B matrix we can see that the hidden state contains vowels and consonants and space is counted as a vowel. The first column of initial and final values in Tab IX are the vowels and the second column are the consonants.

The Google Colab code and the dataset for War and Peace can be found **here**.

	Initial		Final	
a	0.03735	0.03909	7.74626608e-02	6.20036245e-02
b	0.03408	0.03537	9.00050395e-10	2.34361673e-02
c	0.03455	0.03537	2.85482586e-08	5.54954482e-02
d	0.03828	0.03909	1.90968101e-10	6.91132175e-02
e	0.03782	0.03583	1.70719479e-01	2.11816156e-02
f	0.03922	0.03630	2.33305198e-12	2.99248707e-02
g	0.03688	0.04048	3.57358519e-07	3.08209307e-02
h	0.03408	0.03537	6.11276932e-02	4.10475218e-02
i	0.03875	0.03816	1.04406415e-01	1.70940624e-02
j	0.04062	0.03909	6.60956491e-26	1.92099741e-03
k	0.03735	0.03490	2.53743574e-04	1.21345926e-02
l	0.03968	0.03723	1.94001259e-02	4.17688047e-02
m	0.03548	0.03537	4.65877545e-12	3.85907034e-02
n	0.03735	0.03909	4.83856571e-02	6.14790535e-02
o	0.04062	0.03397	1.05740124e-01	4.23129392e-05
p	0.03595	0.03397	2.82866053e-02	1.84540755e-02
q	0.03641	0.03816	9.92576058e-19	1.32335377e-03
r	0.03408	0.03676	8.29107989e-06	1.07993337e-01
s	0.04062	0.04048	2.54927739e-03	9.75975025e-02
t	0.03548	0.03443	3.96236489e-05	1.50347802e-01
u	0.03922	0.03537	2.94555063e-02	8.54757059e-03
v	0.04062	0.03955	2.83949667e-19	3.05652032e-02
w	0.03455	0.03816	4.70315572e-25	3.37241767e-02
x	0.03595	0.03723	2.20419809e-03	1.38065996e-02
y	0.03408	0.03769	6.42635319e-04	3.04765034e-02
z	0.03408	0.03955	4.88081324e-27	1.10990961e-03
space	0.03688	0.03397	3.49317577e-01	4.28089789e-08

TABLE IX: Initial and final B transpose

B. PART B

Ten bent (biased) coins are placed in a box with unknown bias values. A coin is randomly picked from the box and tossed 100 times. A file containing results of five hundred such instances is presented in tabular form with 1 indicating head and 0 indicating tail. Find out the unknown bias values. (2020 ten bent coins.csv) To help you, a sample code for two bent coin problem along with data is made available in the work folder: two bent _coins.csv and embentcoinsol.m

We used Expectation Maximization algorithm from the reference material [19] already shared. An expectation–maximization (EM) algorithm is an iterative method to find (local) maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models,where the model depends on unobserved latent variables[20]. In our case, the latent variables are z, the coin types. We have no knowledge of the coins but the final outcome of 500 trials. We used EM to estimate the probabilities for each possible completion of the missing data, using the current parametersϑ.

We tried to implement EM for 10 bent coins to estimate their unknown bias on Google Colab, it's code can be found **here**. We took reference from Karl Rosaen's solution for 2 bent coins problem[21].

C. PART C

A point set with real values is given in _2020 em clustering.csv. Considering that there are two clusters, use EM to group together points belonging to the same cluster. Try and argue that k-means is an EM algorithm.

We used Expectation Maximization algorithm for grouping the points belonging to the same cluster. We also used *k-means* and compared the results of EM algorithm clustering and *k-means* clustering [19].

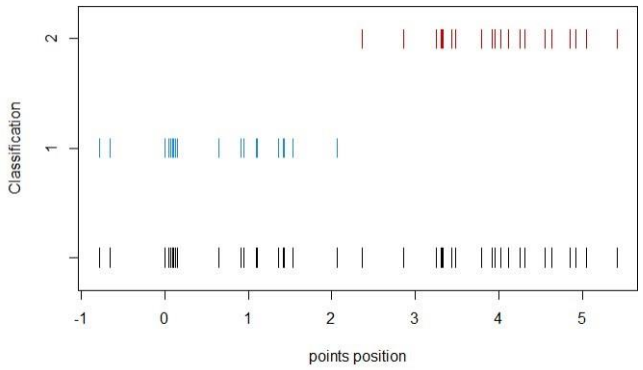


Fig. 24: EM clustering

In Fig 24, the lines depicted with black color are the points from the given dataset and the lines depicted with blue and red respectively are the two clusters formed by the EM algorithm from the dataset. The model has grouped the points belonging to the same cluster.

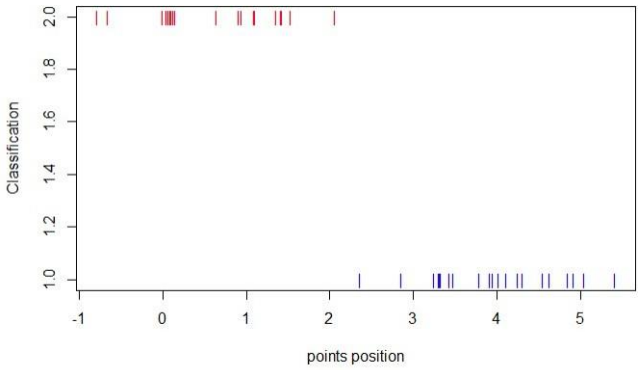


Fig. 25: k-means clustering

From the Fig 25, we can infer that *k-means* and EM algorithm results are same i.e. 19 points lie in the first cluster and 21 points lie in the second cluster out 40 given points.

k ~~-~~means uses hard-clustering which means that a point either belongs to the cluster or it does not belong the cluster. EM algorithm uses soft-clustering technique to assign points to a cluster, it gives the probability that a particular point belongsto a cluster. In this assignment, from the given dataset both the methods EM algorithm and *k–means* yield the same result because the dataset was in such a way that the probability of a point belonging to a cluster was sufficient to assign the point to that cluster therefore the soft-clustering and hard-clustering yield the same result.

The R-markdown and the high resolution images can be found **here**.

III. Week 7 Lab Assignment 7

Learning Objective: To model the low level image processing tasks in the framework of Markov Random Field and Conditional Random Field. To understand the workingof Hopfield network and use it for solving some interesting combinatorial problems

A. Part A

Many low level vision and image processing problems are posed as minimization of energy function defined over a rectangular grid of pixels. We have seen one such problem, image segmentation, in class. The objective of image denoising is to recover an original image from a given noisy image, sometimes with missing pixels also. MRF models denoising as a probabilistic inference task. Since we are conditioning the original pixel intensities with respect to the observed noisy pixel intensities, it usually is referred to as a conditional Markov random field. Refer to (3) above. It describes the energy function based on data and prior (smoothness). Use quadratic potentials for both singleton and pairwise potentials. Assume that there are no missing pixels. Cameraman is a standard test image for benchmarking denoising algorithms. Add varying amounts of Gaussian noise to the image fortesting the MRF based denoising approach. Since the energy function is quadratic, it is possible to find the minima bysimple gradient descent. If the image size is small (100x100) you may use any iterative method for solving the system of linear equations that you arrive at by equating the gradient to zero.

We started with first importing the image of the 'cameraman', which is a standard image for benchmarking denoising algorithms, as it is very dynamic in the grayscale pixel range. [22]

This is a 512x512 grayscale image. We normalize the pixel values to be between 0 and 1, by dividing all values by 255, and then 'binarizing' it for the Markov Random Field byconverting all the normalized pixel values below 0.5 to 0 and the rest to 1, as shown in Fig 26.

We, then introduce noise to this 'binarized' image. In order to test the capability, we add varying levels of noise, from 5% to 25% of the pixel values.

The varying levels of noises are shown in Fig 28.

Markov random fields sue a quadratic potential function to measure the energy potential of the image when changing a particular pixel, with respect to the neighbouring pixels.

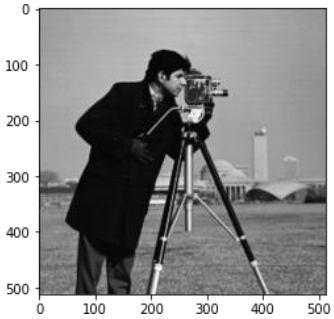
The quadratic potential function is given by:

$$E(u) = \sum_{n=1}^{N} (u_n - v_n)^2 + \lambda \sum_{n=1}^{N-1} (u_{n+1} - u_n)^2 \qquad (18)$$

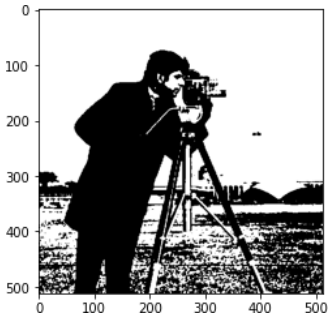
where v is the smooth ID signal, u is the IID and E is the energy function.

This is because for most cases, the values around a pixel are close to the pixel value. [23], [24]. We use the value ofthe constant lambdas as -100, while computing the quadratic potential function.

We ran the algorithm for 5*512*512 = 1310720 iterations.



(a) Original Cameraman Image



(b) 'Binarized' Cameraman Image

Fig. 26: Cameraman Images

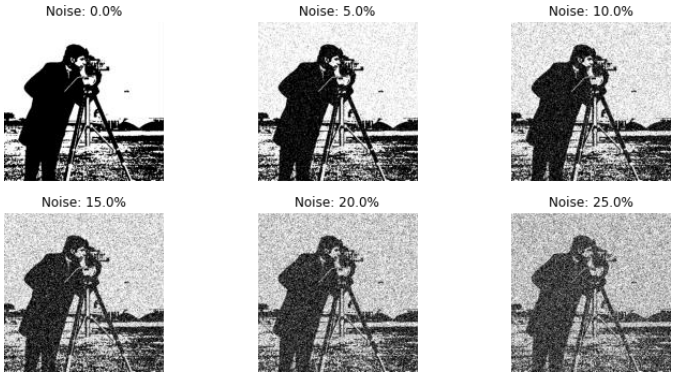


Fig. 27: Varying levels of noise added in the cameraman image

The max pixel denoised for all the noise levels are given in table

The images after denoising with the MRF are: as shown in fig .

% noise Level	% pixels denoised
5	4.70
10	6.99
15	9.30
20	11.67
25	14.02

TABLE X: Noise level and percent of pixels denoised

B. Part B

For the sample code hopfield.m supplied in the lab-work folder, find out the amount of error (in bits) tolerable for each of the stored patterns.

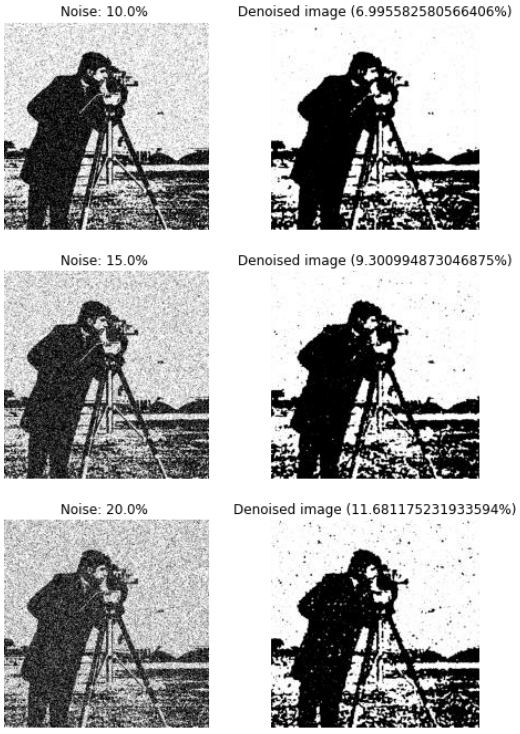


Fig. 28: Denoised images using MRF

For this task, we converted the hopfield.m MATLAB codes to Python so that we could do it in the same Notebook as the other parts.

The original images looked as shown in fig. 29.

The network was trained using Hebb's rule, as in the file and then they were noised by changing some random pixel values. At max, 15 pixel values were noised.

We can see that suprisingly, the network can correct upto max 8 errors. The results are show in fig 30.

C. *Part C*

Solve a TSP (travelling salesman problem) of 10 cities with a Hopfield network. How many weights do you need for the network?

This is the usual famous NP-hard problem of Travelling Salesman, that is done using the a Hopfield Networks.

Since in a Hopfield Ntwrosk, each node is connected to each other node, we needed a total of 10x10 = 100 weights.

We first generate 10 cities randomly, as shown in fig 31.

And then let the hopfield network predict an optimal least path cost.

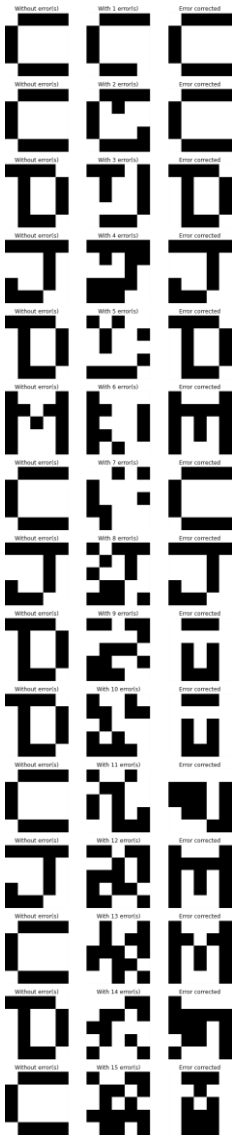


Fig. 30: The original, noised and corrected letters

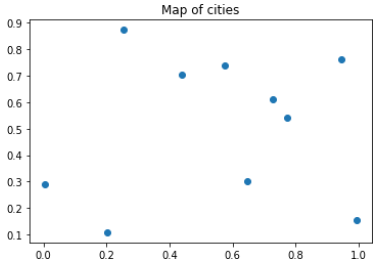


Fig. 31: Cities

IV. Week 8 Lab Assignment 8

Learning Objective: Basics of data structure needed for state-space search tasks and use of random numbers required for MDP and RL.

Problem Statement: Read the reference on MENACE by Michie and check for its implementations. Pick the one that you like the most and go through the code carefully. Highlight the parts that you feel are crucial. If possible, try to code the MENACE in any programming language of your liking.

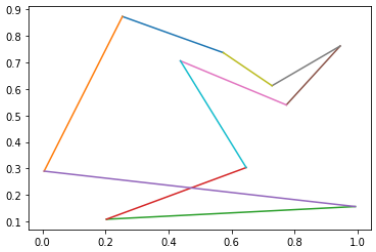


Fig. 32: Shortest Path

A. What is MENACE?

MENACE stands for Machine Educable Noughts and Crosses Engine [17]. It was originally described by Donald Michie, who used matchboxes to record each game he played against this algorithm. This provides an adequate conceptual basis for a trial-and-error learning device, provided that the total number of choice-points which can be encountered is small enough for them to be individually listed. Michie's aim was to prove that a computer could "learn" from failure and success to become good at a task.

B. Why Matchbox machine?

Matchboxes were used because each box contained an assortment of variously coloured beads. The different colours correspond to the different unoccupied squares to which moves could be made.

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

Fig. 33: The colour code used in the matchbox machine. The system of numbering the squares is that adopted for the subsequent computer simulation program

C. MENACE Strategies and how it learns.

A loss is punished by a removing the beads that were chosen from the boxes. This means that MENACE will be

less likely to pick the same colors again and has learned. A win is rewarded with three beads of the chosen color which is added to each box, reinforcing the MENACE to make the same move again. If a game is a draw, one bead is added to each box.

From the figure [34] , we can see the number of beads present inside a box on any given stage. Removing one bead from each box after losing means that later these moves are less likely to be picked and this helps MENACE learn more quickly, as the later moves are more likely to have led to the loss.

It is possible that after few games some boxes may end up empty. If one of these boxes is to be used, then MENACE resigns. When playing against skilled players, it is possible that the first move box runs out of beads. In this case, MENACE should be reset with more beads in the earlier boxes to give it more time to learn before it starts resigning.[18]

STAGE OF PLAY	NUMBER OF TIMES EACH COLOUR IS REPLICATED
1	4
3	3
5	2
7	1

Fig. 34: Variation of the number of colour-replicates of a move according to the stage of play.

D. Results

We tried to implement MENACE on Google Colab, it's code can be found **here**.

IV. Week 9 Lab Assignment 9

Learning Objective: Understanding Exploitation - Exploration in simple n-arm bandit reinforcement learning task, epsilon-greedy algorithm

A. Part 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 decideupon the action to take for maximizing the expected reward. There are two binary bandits named binaryBanditA.m and binaryBanditB.m are waiting for you.

After using both the rewards(function) binaryBanditA and binaryBanditB here is what we observe :

Here we can observe that when probability of rewards were low (in the case of banditA) the expected rewards were initially

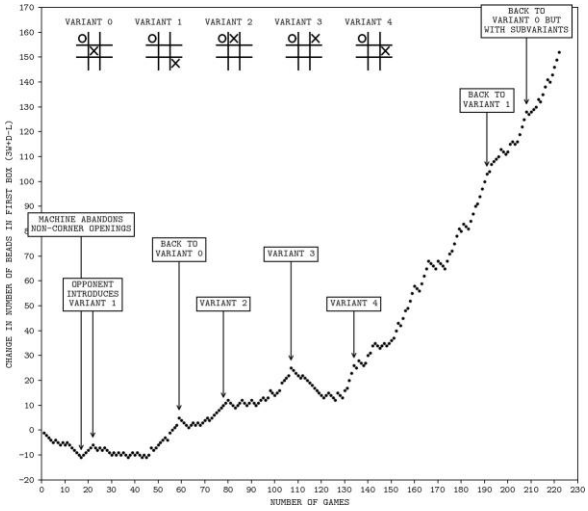


Fig. 35: The progress of MENACE'S maiden tournament against a human opponent. The line of dots drops one level for a defeat, rises one level for a draw and rises three levels for a victory.

0 as the n o. of trials increased we can see that expected rewards too become higher but this is not in the case of banditB as the probability of the reward were high(0.8 and 0.9) so the expected rewards for m the start are a close to 1and the steps go by the expected rewards became a kind of constant because of the averaging factor

B. Part 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).

This is the case of 10 arm-bandit where the mean-rewards are initially taken as a constant valued array initialised by 1. We performed exploration and exploitation based on the Epsilon Greedy Algorithm. A random number between 0 and 1 is generated and if it comes out to be greater than epsilon, we perform exploitation, which is based on the prior knowledge otherwise exploration. For every iteration an array of ten values is generated such that they are normally distributedwith mean value zero and standard deviation of 0.01. This array is added to the mean array and this updated array isused every time. For every action a reward is given from this updated mean-rewards array. The rewards given in this case are non-stationary.

Here we observe that as initially all the rewards were equal spo that we get a constant reward of say like 1 but as the steps increases , it affects the rewarding policy of every action and

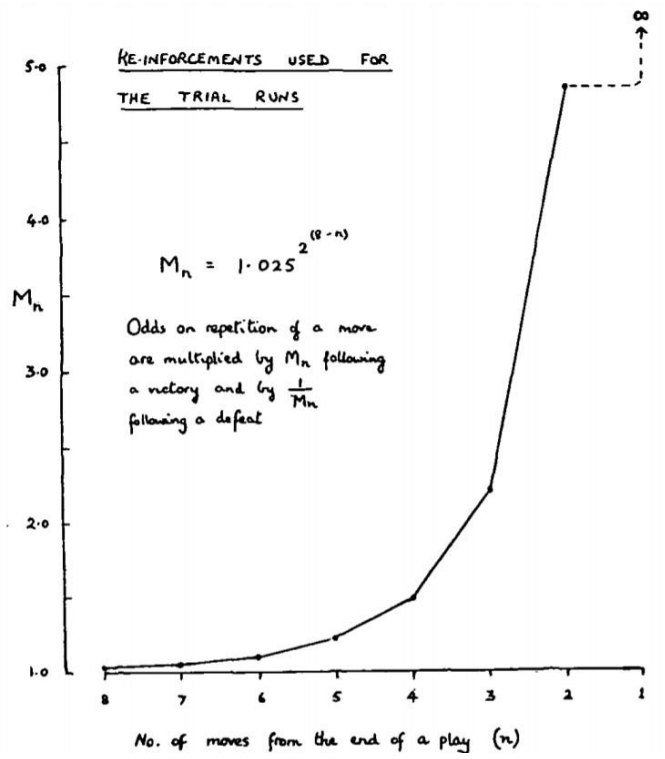


Fig. 36: The multipliers used for reinforcement in the trial runs.

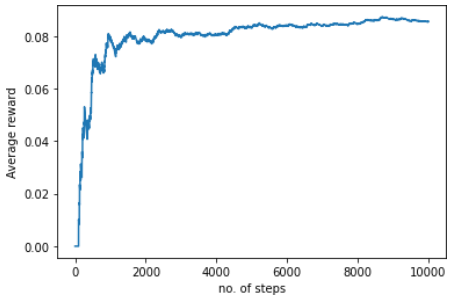


Fig. 37: Expected rewards we get after using banditA

thus we see that the expected rewards also started increasing at a very high non zero rate (Fig.31)

C. Part D

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 is same as problem 2 except the calculation of action rewards. In this case, instead of using averaging method to update estimation of action reward, we are assigning more weights to the current reward earned by using alpha parameter having value 0.7

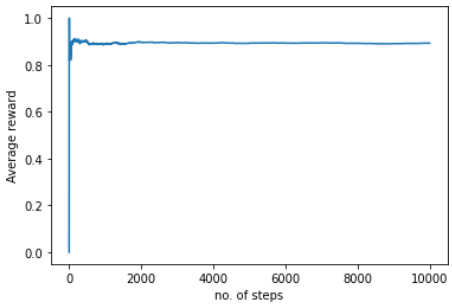


Fig. 38: Expected rewards we get after using banditB

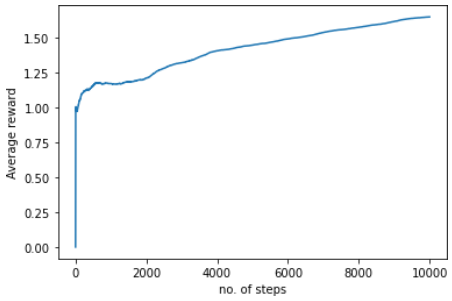


Fig. 39: Expected rewards we get when rewards are non stationary (averaging method)

Here we can see that as compared to problem 2, in which the expected rewards were low , we saw that when instead of averaging the profit percentage of state when we gave higher weightage to the to the current value at every step (a normally distributed array is added to the rewards array), the expected rewards we get were much higher as in the case 2 and the slope of the graph was too steeper when compared with case 2.

The Google Colab Notebook code for this assignment can be found **here**.

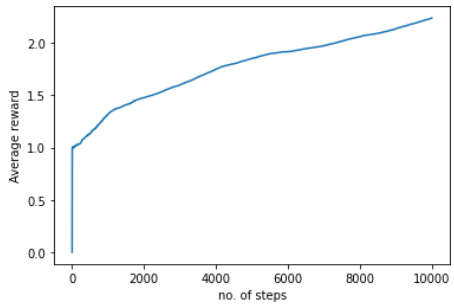


Fig. 40: Expected rewards we get when rewards are non stationary (non averaging method)

V. Conclusion

As demonstrated above we have solved 8 puzzle problem, Travelling Salesman Problem, Noughts and Crosses game,Nim Game and understood the Bayesian graphical models for to build network graphs.

In 8-puzzle problem we were able observe performance of different heuristic functions, where heuristic with Manhattan distance gave best results whereas when solved without heuristic performance was the worst.

For the Travelling Salesman Problem, we observed that with simulated annealing it provides an optimal or sub optimal solution which reduces the cost for given set of points/coordinates. We also observed that how the decrease in temperature and number of iterations affects the solution.

From our observation for mini-max and alpha-beta pruning less nodes were evaluated in alpha-beta pruning than in mini-max algorithm (See table V), which was the essence of the assignment. We can clearly say that alpha-beta pruning is not a different algorithm than mini-max it is just a speed up process which does not evaluate approximate values instead evaluates to perfectly same values.

We explored the Bayesian network and were able to model it using grades data-set, we then calculated Conditional Probability Tables (CPTs) for them and saw how they were dependent (See Fig. 20). Naive Bayes Classifier gave very accurate results on the test data-set which can be observed here at Fig 23 and Fig 22.

From the Markov Random Field and Hopfield networks, we learned how to construct learning machines using Markov Decision Processes. Hopfield Network showed how we can use a simple collection of neurons to construct a very robust network, that can resist ignore large noises and still predict a correct output and lastly, we saw how they can be used in Travelling Salesman Problem to get an optimal path.

We learnt about MENACE [17] and how it learns through reinforcing its decision. The Aim was to prove that machines can learn from failure and make better decisions after trial and error. From the fig. 35, we can see how it performs against humans as more and more games are played. Initially, it struggled a lot and lost but after sometime there was consecutive draws but it faltered again and lost few matches and finally it kept wining.

From the HMM and EM algorithm problem, we applied a HMM model to the 50,000 letters observations including space and all lowercase characters. After about 100 iterations we observed that the B matrix (see Table IX) tells us about two distinct categories of characters. Upon closely observing, we find that one set contains consonants while the other contains vowels. For the second part we apply EM algorithm to the 10 bent coin problem (similar to well known 2 bent coin example) to find the latent parameters and estimate the hidden biases for the coins. In the third part we learnt about soft and hard clustering and EM algorithm clustering for given dataset, we also learnt about the k-means and how k-means compares to EM algorithm.

For the n arm bandit we learned how the problem works and the epsillon greedy algorithm ,we saw that how that algorithm can be modelled for both the stationary environment case and the non stationary case just be simply giving more weightage to the current step.

Acknowledgment

We would like to thank Prof. Pratik Shah for guiding us through this course and the contents, both during the lecture and laboratory hours. The lectures have helped us immensely in understanding the contents of this course and to apply this knowledge in performing these experiments.
We'd also like to thank our fellow colleagues, that help in healthy discussions of the course content.
Lastly, we would like to thank any source that we might have referred to, for performing these experiments but may have missed to give credits to, in the reference section.

References

[1] Josh Richard, *An Application Using Artificial Intelligence*, January 2016
[2] dpthegrey, *8 puzzle problem* June 2020.
[3] Ajinkya Sonawane, *Solving 8-Puzzle using A* Algorithm* September 2018
[4] javatpoint *Uninformed Search Algorithms*
[5] Zhou, Ai-Hua, *Traveling-salesman-problem algorithm based on simulated annealing and gene-expression programming*. Information 10.1 2019.
[6] Frank Liang, *Optimization Techniques for Sim- ulated Annealing*. https://towardsdatascience.com/ optimization-techniques-simulated-annealing-d6a4785a1de7
[7] Traveling Salesperson Problem. https://www.youtube.com/watch?v=35fzyblVdmA&t=656s.
[8] Marco Scutar, *Learning Bayesian Networks with the bnlearn R Package*, Issue 3. Journal of Statistical Software, July 2010.
[9] Bojan Mihaljević, *Bayesian networks with R* November 2018.
[10] Alexandra M. Carvalho, *Scoring functions for learning Bayesian networks*
[11] Shah Pratik, *An Elementary Introduction to Graphical Models* February 2021.
[12] The R Foundation, *R: Documentation* February 2021.
[13] Marianne Freiberger, *Play to win with Nim*.July 21, 2014. https://plus.maths.org/content/play-win-nim#:~:text=The%20strategy%20is%20to%20always,B%20has%20a%20winning%20strategy.

[14] Best-Case Analysis of Alpha-Beta Pruning. http://www.cs.utsa.edu/~bylander/cs5233/a-b-analysis.pdf
[15] 8puzzle problem. https://medium.com/@dpthegrey/8-puzzle-problem-2ec7d832b6db
[16] Stamp, Mark, *A revealing introduction to hidden Markov models*, 2004.
[17] MENACE: Machine Educable Noughts And Crosses Engine https://people.csail.mit.edu/brooks/idocs/matchbox.pdf
[18] MENACE: Machine Educable Noughts And Crosses Engine Blog https://www.mscroggs.co.uk/blog/19
[19] What is the expectation maximization algorithm? ChuongB Do and Serafim Batzoglou, Nature Biotechnology, Vol 26, Num 8, August 2008 https://datajobs.com/data-science-repo/ Expectation-Maximization-Primer-[Do-and-Batzoglou].pdf
[20] Expectation–maximization algorithm https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm
[21] Expectation Maximization with Coin Flips http://karlrosaen.com/ml/notebooks/em-coin-flips/
[22] What is the reason the test image ”Cameraman” is used widelyto test algorithms in image processing and image encryption? https://www.researchgate.net/post/What_is_the_reason_the_test_image_Cameraman_is_used_widely_to_test_algorithms_in_image_processing_and_image_encryption
[23] Image Denoising Benchmark https://www.cs.utoronto.ca/~strider/Denoise/Benchmark/
[24] Markov Random Fields https://web.cs.hacettepe.edu.tr/~erkut/bil717.s12/w11a-mrf.pdf