[ICITAIC-2019]

ISSN 2348 – 8034
Impact Factor- 5.070

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

TIC-TAC-TOE

V. Swetha[*1], M. Bhavana[2], Mr. Rajasekhar Sastry[3], Dr. B V Ramana Murthy[4] &Mr. C Kishor Kumar Reddy[5]

[*1,2,3,4&5]Stanley College of Engineering and Technology for Women, Hyderabad

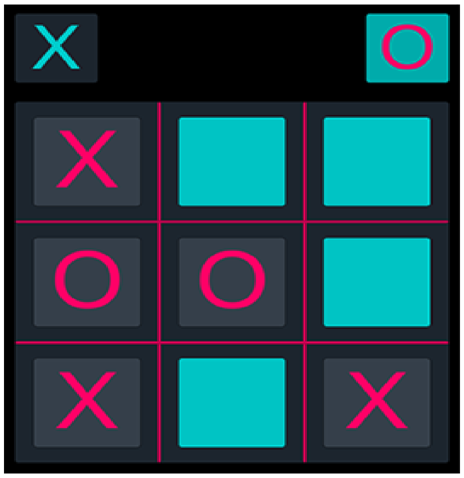
ABSTRACT

This study exhibits the application of the concept of matrices, probas found ability and optimization in making an number of moves in every single game of Tic-Tac-Toe, the moves are recorded in a 3x3 matrix and the subsequent solution, or a winning combination, is presented from the data obtained by playing the electronic game. The solution is also displayed electronically using an LED. The circuit has been designed in a way to apply Boolean logic to analyze player's moves and thus, give a corresponding output from the electronic game and use it in matrices. The electronic Tic-Tac-Toe game is played randomly between 20 pairs of players. The impact of different opening moves is observed. Also, effect of different strategies, aggressive or defensive, on the outcome of the game is explored. The concept of Boolean algebra, logic gates, matrices and probability is applied in this game to make the foundation of the logic for this game. The most productive position for placing an 'X' or 'O' is found out using probabilities.

Keywords: *matrices, Boolean logic, Boolean algebra, logic gates.*

I. INTRODUCTION

In this assignment we will be making a Tic-Tac-Toe game (also known as Noughts and Crosses) using nothing more than the built in Unity UI and two basic scripts.



The approach to creating the game in this lesson should show two important things.
The first is the versatility of the Unity UI toolset. Unity's UI toolset is very powerful and can be used to accomplish many tasks in any given project. Many of these tasks may not seem to be UI related at first but the UI toolset might turn our to be the good and unexpected solution. In this project, we would like to show how, without the use of custom graphics, sprite sheets or complex interactions, we can make this game completely "out of the box" using Unity's UI and two simple scripts.

In addition to this, we want to walk through a project by breaking it down into manageable pieces; approaching and solving one problem at a time. We will be following the steps of designing and prototyping that one could take during any game project rather than jumping straight to the finished product and explaining only how to assemble a completed game from pre-made parts. These design and prototyping steps will include *refactoring code* and adjusting the *scope* of the project as we work through this lesson.

Rather than simply showing the final, complete project, we hope that this approach will help us understand not only the underlying reason for the choices made in this project, but help understand the many small steps taken in making a game.

It is easy to be intimidated by the scope of even a small project. "Where do I start?" is a common question, followed quickly by "What now?". After doing this project, we hope that everyone will be able to start tackling larger projects by breaking them down into smaller, more manageable pieces.

Scope of the game
The game play will be simple.
There will be a simple square game board divided into nine tiles or grid spaces. When the player clicks on one of the grid spaces, it will be assigned either an "X" or an "O". The game is over when one player claims 3 grid spaces in a row or there are no moves left. The game will have a small amount of polish to make it complete. At the start of the game, the board will not be active until the first player has chosen whether they are to play "X" or "O". A panel will indicate whose turn it is. When the game is over, a banner will display the winner or announce a draw if no one wins. A restart button will be displayed when the game is over, returning the game to the starting state when clicked. The game will need a few basic elements.

A background providing a backdrop for the the entire game. An element that will be our game board. An element, or set of elements, that breaks the game board up into nine areas in an even grid. Nine tiles that can be assigned either an "X" or and "O", but once assigned these values will persist and not be changeable by the players - either the current player or the opponent. Logic to change sides when a player takes their turn. Logic to check for a "Win" condition, allowing for draws where no one wins. A panel that displays who is the winner when the game is over. For polish, towards the end of the project, we could have some other, helpful elements.
A way to choose the starting player's side, "X" or "O".
An indicator of whose turn it is.
A restart button.
Very basic instructions.
Lastly, as part of this exercize, we are going to be using nothing more than the elements provided by Unity's built in UI toolset.

Tic Tac Toe game development using c:
While making a Tic Tac Toe game using C language, it is important to make use of arrays. The Xs and Os are kept in different arrays, and they are passed between several functions in the code to keep track of how the game goes. With the code here you can play the game choosing either X or O against the computer.
This Tic Tac Toe C game is such that you will have to input a numerical character, from 1 to 9, to select a position for X or O into the space you want. For example: if you are playing with O and you input 2, the O will go to first row – second column. If you want to place O in third row – first column, you have to enter 7. And, it is similar for the other positions.

This has been done this way because it is just a console application without graphics designed in C language. The gotoxy function has been used to print text in any part of the screen.

II. LITERATURE SURVEY

1. Exploring the Possibilities of Using Tic-Tac-Toe to Think and Communicate about Mathematics (Australian Mathematics Teacher,2008)
Doing mathematics, and thinking about how you are doing it at the same time, are not the easiest things to do. It is even more difficult if students are not aware that they should be attempting both processes at the same time. They are likely to concentrate on the immediate task of "doing" the mathematics, rather than trying to access the deeper process. Yet it is this deeper process that is really at the heart of mathematics. In turn, accessing this deeper process requires in part some command of the appropriate rational/logical language so communication with yourself and others can proceed effectively and efficiently. This article discusses the possibilities of using students' explorations of the traditional strategy game "tic-tac-toe," and some extensions, to set up situations for students to discuss and examine this process.

2.Tic-Tac-Toe Played as a Word Game (Word Ways,2010)
In Problem 48 in Your Move (McGraw-Hill, 1971), David Silverman described a linguistic version of tic-tac-toe consisting of a stockpile of the words ARMY, CHAT, FISH, GIRL, HORN, KNIT, SOUP, SWAN and VOTE. Players alternately select words, and the first to collect three words sharing a common letter is the winner.

In the single-letter analogue of Silverman's game, players draw alternately from a stockpile of nine different letters; the first to select the letters forming one of a specified list of nine words is the winner. To make it easy to remember these words, they can be written in the form of a 3-by-3 word square in which both diagonals are also words.

Squares are easy to find if you allow abbreviations, acronyms, proper names or foreign words. However, I'm half-convinced that there is no solution with common everyday words. It's almost spooky how you can find seven words but not the eighth. My near-misses:

Tic-Tac-Toe Game The Tic-Tac-Toe, also called noughts and crosses, is originally designed as a paper and pencil game. Two players take turns marking down their own Xs or Os in a three by three grid. A player who successfully place three marks in a row, a column, or a diagonal wins the game. Thus, the rule to win the game is “three-in-a-row.” Fig. 1 shows a game won by X due to three X in the diagonal row. There are 255,168 possible games in total. Among the available games, 958 terminal configurations are reached when a winner is found [8].

Classification is one of the typical data mining techniques. It is a supervised learning process to build a model for future predictions (classifications). The model, generated from training data, is to associate characteristics of data observations with a desired category. The observations in the training data consist of a set of characteristics (input variables) and a given category (output variable). Tic-Tac-Toe game is a typical binary classification problem. In classifying Tic-Tac-Toe games, there are nine positional input variables and a given winner output variable. Each input variable can carry an “X”, an “O”, or nothing (blank). The binary winner output variable can be either “X” or “O”. The training data is a set of games {g1, g2, …, gj} with their output category {c1, c2}. A machine learning algorithm, a classifier, is used to find the model ci = f(gj) from correct pairs of <gj, ci> consistently, where game gj is represented by nine positional variables. Once the classifier is built, a correct prediction is made when a game gj can be assigned to the correct class ci, either “X” or “O”. Various classifiers have been successfully applied in several data mining problems such as credit evaluation [1], forecasting financial performance [2], assessing risks of prostate cancer patients [3], generating document taxonomies [4], profiling Web usage in the workplace [5], classifying open source software development projects [6], and web query classification [7]. C. Evaluation Methods 1) Training set method The training set evaluation method is to utilize 100% of observations to build a classifier and apply the same set of observations as test set for evaluation. The performance of such evaluation method is normally optimal or overestimated while the built classifier may not perform well when using different test sets. 2) Holdout method The holdout method, so called percentage split, is to randomly split the observations into two sets, a normally larger training set and a normally smaller test set. In this study, 70% of observations are used for training and 30% of observations are used for evaluation. Good evaluation results may be obtained in some lucky cases randomly allocating easy observations in the test set. 3) Cross validation method In order to address the issues using training set and holdout method, cross validation provides a more rigorous evaluation procedure by randomly splitting observations into a number of subsets, e.g. n folds. The first n-1 folds are used for training and the left fold is used for testing. Next, it rotates the test set forward while still using the other n-1 folds for training. After repeating the procedure n times, the last procedure uses the last n-1 folds for training and the first fold for testing. Finally, the average of the n evaluation results is the overall performance using the cross validation method. "Extensive tests on numerous datasets, with different learning techniques, have shown that 10 is about the right number of folds to get the best estimate of error, and there is also some theoretical evidence that backs this up" [9]. Therefore, in this study, 10-fold cross validation is applied.

III. EVALUATION METHODS

A. Data Collection and Data Preparation In a Tic-Tac-Toe game, nine input cells are available for nine positions. They are upper-left (UL), upper-middle (UM), upper-right (UR), middle-left (ML), middle-middle (MM), middle-right (MR), lower-left (LL), lower-middle (LM), and lower-right (LR). According to Schaeffer [8], 958 terminal configurations are reached when a winner is found assuming X is the first player. Therefore, the 958 games are used for this study. Table I lists 10 games (observations) out of the 958 games. The first nine variables are the positional input variables and last variable—Winner is the output variable. Notation "x" is used for input variables if the player X marks on the cells. Similarly, notation "o" is used if the player O marks on the cells. When a cell is blank without any mark, "b" is used. For the Winner variable, it could be "x" if the player X gets three marks in a row. "o" is stored if the play O is the winner.

Table I: sample observations of tic-tac toe games

UL	UM	UR	ML	MM	MR	LL	LM	LR	Winner
X	O	X	O	X	O	O	X	X	X
X	O	X	O	X	O	B	B	X	X
X	O	X	O	X	B	X	O	B	X
X	O	X	O	X	B	X	B	O	X
X	O	X	O	X	B	O	B	X	X
X	O	X	X	O	X	O	O	B	O
X	O	X	X	O	X	B	O	O	O
X	O	X	X	O	O	B	O	X	O
X	O	X	X	O	B	O	O	X	O
X	O	X	X	O	B	B	O	B	O
X	O	X	X	O	X	O	O	B	O

B. Experimental Design Seven machine learning methods (Navïe Bayes [10], Neural Network, Support Vector Machine [11], [12], Logistic [13], k-Nearest Neighbor [14], and Decision Treess: ID3 [15] and C4.5 [16]) were used for performance evaluations using Weka [17] tool. Two experiments were conducted in order to answer the research questions.

1) Design of experiment 1 In first experiment, for each learning method, 10-fold cross validation was performed 10 times. Accuracy rate of a fold was treated as a performance outcome. Thus, 100 performance results were obtained for each learning method. ANOVA was then applied to determine whether the seven learners have similar performance. When difference was found, Bonferroni post hoc test was applied to differentiate the performance of seven learners.

2) Design of experiment 2 In the second experiment, novice participants were asked to conduct the experiments using Weka. Two pedagogies were used to instruct the use of Weka tool. Through both pedagogies, three evaluation methods (training set, holdout, and cross validation) were used for evaluating the seven learning methods. Finally, the 21 evaluation results from each participant were aggregated and t tests were conducted to determine whether the different pedagogies matter in instructing Weka. In total, 136 students taking upper-level management information systems course in two different semesters participated in the experiments. After covering the concepts of machine classifiers and evaluation methods, the experiments took place in a computer lab. Different pedagogies were used in

the two semesters. In the first semester, textual instructions using Weka tool and settings for the experiments were given (see one example in Fig. 2). Participants followed the printed instructions and conducted the experiments their own. In the second semester, same printed instructions were given with additional screenshots leading the processes.

IV. EXPERIMENTAL RESULTS

A. Experiment 1 Based on ANOVA analysis, there was a significant mean performance difference among the seven machine classifiers (p=.000), Bonferroni post hoc test was performed to conduct pairwise comparisons with a control of overall error rate. The results of pairwise t tests were listed in Table II. Based on the mean performance of 10 runs of 10-fold cross validation results (second column of Table II), 3-Nearest Neighbor outperformed other methods (see second column of Table II). The performance of two decision tree classifiers J4.8 and ID3 were significantly different at .05 level, but not at the .01 level. Pairwisely, no difference was found in a group of four classifiers: Neural Network (NN), Support Vector Machine (SVM), Logistic (LOG), 3-Nearest Neighbor (3NN). Since there was no significant difference found in the four classifiers, they were in the top performer's group. Navïe Bayes (NB) had the poorest performance among the seven classifiers.

B. Experiment 2 The objective of the second experiment was to examine whether the novice participants were able to conduct data mining classifications correctly using three different evaluation methods. The same seven machine learners in experiment 1 were used in this experiment. The three evaluation methods used were training set, holdout, and 10-fold cross validation. Each participant was asked to conduct the experiments and report the performance results, accuracy rates, of the classifiers along with the original result generated from the Weka tool (see Appendix). 21 (7 classifiers by 3 evaluation methods) experimental results were collected from each participant. A total of 136 participants joined this experiment and similar size of participants was allocated in the two semesters. The participants came with similar background taking the same course. The mistakes found from participants conducting the classifications were summarized in the Table III. While getting correct results from Weka tool, three participants reported the results incorrectly in the first semester and one was found in the second semester. With such mistake, the results of cross validation method, for example, were reported as results of holdout method. Default settings were used for the classifiers except for NN and 3NN. Four participants in the first semester and one in the second
When evaluating the performance of 3-Nearest Neighbor classifier using holdout method, you need to change the classifier setting and holdout setting. 1. After choosing the IBk classifier, click on the bar in the Classifier section in the top portion of screen and change the value to 3 for the KNN in the pop-up window. 2. Next, in the test options, choose Percentage split for holdout method and specify the percentage to 70. 3. Finally, you can click on Start button to execute the evaluation. The result will be displayed in the Classifier output area.

Similarly, three participants in the first semester and one participant in the second semester used the 3NN classifier without changing the settings. In addition, the default numerical split for the holdout is 66%. Three from the first semester forgot to change it to the required 70%. Based on the number of participants making mistakes evaluating machine learners, pedagogy in the second semester improved the evaluation processes.

Furthermore, to determine whether there was a significant difference between the correct accuracy rate of a classifier and the corresponding mean accuracy rates of the classifier prepared by participants using a particular evaluation method, one t test was applied. A total of 42 t tests were applied for the two semesters. Table IV, Table V, and Table VI summarizes the performance results of classifiers using training set, holdout, and cross validation evaluation methods, respectively.typical binary classification problem. In classifying Tic-Tac-Toe games, there are nine positional input variables and a given winner output variable. Each input variable can carry an "X", an "O", or nothing (blank). The binary winner output variable can be either "X" or "O". The training data is a set of games {g1, g2, …, gj} with their output category {c1, c2}. A machine learning algorithm, a classifier, is used to find the model ci = f(gj) from correct pairs of <gj, ci> consistently, where game gj is represented by nine positional variables. Once the classifier is built, a correct prediction is made when a game gj can be assigned to the correct class ci, either "X" or "O". Various classifiers have been successfully applied in several data mining problems such as credit evaluation [1], forecasting financial performance [2], assessing risks of prostate cancer patients [3], generating document taxonomies [4], profiling Web usage in the workplace [5], classifying open source software development projects

ISSN 2348 – 8034
Impact Factor- 5.070

[6], and web query classification [7]. C. Evaluation Methods 1) Training set method The training set evaluation method is to utilize 100% of observations to build a classifier and apply the same set of observations as test set for evaluation. The performance of such evaluation method is normally optimal or overestimated while the built classifier may not perform well when using different test sets. 2) Holdout method The holdout method, so called percentage split, is to randomly split the observations into two sets, a normally larger training set and a normally smaller test set. In this study, 70% of observations are used for training and 30% of observations are used for evaluation. Good evaluation results may be obtained in some lucky.

In the single-letter analogue of Silverman's game, players draw alternately from a stockpile of nine different letters; the first to select the letters forming one of a specified list of nine words is the winner. To make it easy to remember these words, they can be written in the form of a 3-by-3 word square in which both diagonals are also words.

Squares are easy to find if you allow abbreviations, acronyms, proper names or foreign words. However, I'm half-convinced that there is no solution with common everyday words. It's almost spooky how you can find seven words but not the eighth. My near-misses:
Tic-Tac-Toe Game The Tic-Tac-Toe, also called noughts and crosses, is originally designed as a paper and pencil game. Two players take turns marking down their own Xs or Os in a three by three grid. A player who successfully place three marks in a row, a column, or a diagonal wins the game. Thus, the rule to win the game is "three-in-a-row." Fig. 1 shows a game won by X due to three X in the diagonal row. There are 255,168 possible games in total. First column of the tables indicates the machine classifiers. Second column reports the correct evaluation accuracy rates in percentage. Mean accuracy rates of classifiers prepared by participants in the first and second semesters are listed in the third and fourth columns. P-values of the t tests are listed below the accuracy rates. Statistically, no significant differences were found in most participants' evaluations, when compared with correct performance results. When training set evaluation method was used, mean accuracy rates of J4.8 (at .1 level), ID3 (at .1 level), and NN (at .05 level) from the participants in the first semester were significantly different from the corresponding correct results. Using the same evaluation method, all participants from the second semester obtained correct results for SVM and Logistic classifiers. When holdout method was used, the results of J4.8 and NN from the participants in the first semester were significantly different from the correct results at .05 level. No significant differences were found in the second semester.

V. CONCLUSIONS AND FUTURE DIRECTIONS

In the first experiment, 3-Nearest Neighbor classifier outperformed others with accuracy rate close to 99% classifying Tic-Tac-Toe games. Four machine learners, Neural Network, Support Vector Machine, Logistic, and 3-Nearest Neighbor, were the top performers with over 98% accuracy rates. There was no statistically significant difference among these four learners. In the second experiment, two different instructional pedagogies were conducted for novices to learn machine classifications. Both pedagogies showed that most novices can correctly conduct experiments to evaluate machine classifiers using data mining tool Weka. The graphical in-depth instructional pedagogy from the second semester did improve the correctness of evaluations statistically. In this study, a familiar game was used for learning machine classifications. Future studies may adopt another familiar scenario—academic admission decision for learning classifications. The decision could be acceptance or rejection based on a set of applicant's characteristics such as admission exam score, years of working experience, grade point average, etc. Another direction of future studies is to explore the Tic-Tac-Toe rules generated from data mining techniques such as decision trees and association rules.

REFERENCES

1. *W., Weisstein, Eric. "Tic-Tac-Toe". mathworld.wolfram.com. Retrieved 2017-05-12.*
2. *Zaslavsky, Claudia (1982). Tic Tac Toe: And Other Three-In-A Row Games from Ancient Egypt to the Modern Computer. Crowell. ISBN 0-690-04316-3.*
3. *https://books.google.co.uk/books?id=N4nmGjq5dWsC&pg=PA153*
4. *"Tic tac toe Ancient Roman 1st century BCE". Sweetooth Design Company. Retrieved 2016-12-04.*
5. *Canisius College – Morris Games*

6. Notes and Queries, 2nd Series Volume VI 152, Nov. 27 1858.
7. Oxford English Dictionary entries for "Noughts and Crosses", "Tick-Tack" and "Tick-Tack-Toe", dictionary.oed.com
8. Wolf, Mark J. P. (2012-08-16). Encyclopedia of Video Games: The Culture, Technology, and Art of Gaming. Greenwood Publishing Group. pp. 3–7. ISBN 978-0-313-37936-9.
9. Cohen, D. S. (2014-09-20). "OXO aka Noughts and Crosses - The First Video Game". About.com. IAC. Archived from the original on 2015-12-22. Retrieved 2015-12-18.
10. "Tinkertoys and tic-tac-toe". Archived from the original on August 24, 2007. Retrieved 2007-09-27.
11. Bolon, Thomas (2013-05-21). How to never lose at Tic-Tac-Toe. BookCountry. ISBN 9781463001926.
12. "Searching for the cat in tic tac toe". TimesDaily. Retrieved 2016-12-19.
13. Kevin Crowley, Robert S. Siegler (1993). "Flexible Strategy Use in Young Children's Tic-Tac-Toe". Cognitive Science. **17** (4): 531–561. doi:10.1016/0364-0213(93)90003-Q.
14. Martin Gardner (1988). Hexaflexagons and Other Mathematical Diversions. University of Chicago Press.
15. Kutschera, Ant. "The best opening move in a game of tic-tac-toe - The Kitchen in the Zoo". blog.maxant.co.uk. Retrieved 2018-04-07.
16. Oren Patashnik, Qubic: 4 x 4 x 4 Tic-Tac-Toe, Mathematical Magazine 53 (1980) 202–216.
17. Golomb, Solomon W.; Hales, Alfred W. (2002), "Hypercube tic-tac-toe", More games of no chance (Berkeley, CA, 2000), Math. Sci. Res. Inst. Publ., **42**, Cambridge: Cambridge Univ. Press, pp. 167–182, MR 1973012.
18. Averbach, Bonnie; Chein, Orin (1980), Problem Solving Through Recreational Mathematics, Dover, p. 252, ISBN 9780486131740.
19. Mendelson, Elliott (2016-02-03). Introducing Game Theory and its Applications. CRC Press. ISBN 9781482285871.
20. "Puzzles in Education - Wild Tic-Tac-Toe". puzzles.com. Retrieved 2016-11-29.
21. Epstein, Richard A. (2012-12-28). The Theory of Gambling and Statistical Logic. Academic Press. ISBN 9780123978707.