CSE 471 Project Write-up
Team 2

David Ganey, Kerry Martin, Evan Stoll, Michael Theut, and Ben Roos
School of Computing, Informatics and Decision Systems Engineering
Arizona State University
CSE 471

Executive Summary—**The CEO of CactusCard has commissioned a team to solve several problems facing the up-and-coming financial company. The company is in the process of releasing a new card, called the CactusCardPlus. In order for this new product to be a success, the CEO recognizes that CactusCard Credit must accomplish three distinct goals. First, the company needs to successfully market the new card, through both traditional strategies and in particular to users of social media. In addition to a powerful marketing campaign, the company must ensure it makes good decisions about which applicants are deserving of a line of credit. Failing to do this would put the company at a disadvantage, relative to the numerous credit agencies who do use data to make credit decisions. Finally, the success of the product hinges on its ability to protect users (and the company) from credit card fraud. As such, the team must be prepared to counter fraudulent users through a variety of strategies.**

These tasks will be undertaken by a team of computer science students with knowledge of artificial intelligence. Using search algorithms, the team will maximize the marketing potential of the card. Using machine learning techniques, the team will utilize data to accurately predict which applicants will best utilize their line of credit. Using game theory, the team will develop successful counter-fraud strategies which can be used to protect the company.

First, the team needed to market the card effectively. Using 10 cards with no sign up fee and 0% interest for one year, CactusCard Credit strategically generated interest in the product. Using search techniques described below in Section I, the team devised both simple and complex algorithms which find the optimum individuals to receive these free cards in order to maximize the number of their friends who sign up for the product. The team found that TODO FILL THIS IN was the best algorithm for this, as it BLAH BLAH DETAILS.

After devising a successful marketing strategy, the team turned its focus to the determination of credit worthiness. As seen below in Section II, the team used a well-known machine learning library called Weka to run various machine learning algorithms on the data set provided. This data set, which contained 15 anonymized credit attributes plus whether or not that individual was given a card, was used to train learning machines which were then evaluated on their ability to accurately predict the issuance decision. The team found that a machine running the Multilayer Perceptron algorithm was the most successful at predicting credit decisions and would serve the company well in the future.

Paragraph(s) about the game theory part

I. Part 1

The first part of the project requires the development of search programs to optimize marketing within a social network. Using a dataset given to the study team by the marketing department, the team must propose multiple search-based solutions and document their success. This section will formulate this problem and describe the steps taken to solve it.

A. Problem Importance

For a new product such as the CactusCard Plus, the initial marketing wave may determine whether the product success or fails. This importance of this period, as the product begins to build a user base, cannot be overstated. Even more critical is the fact that the marketing department wishes to give out some of the cards for free. It is critical that the team choose the right users to receive these free cards such that the impact of those cards in their friend network is the largest.

B. Problem Formulation

The goal of the problem is to find the optimal distribution of cards such that the most individuals adopt the card. This means that in traversing the relationships established in the input file, the program should find points to deliver free cards where the cards have the desired effect (increasing exposure to the card) without having an undesired side effect (namely, giving too many free cards to a small group and failing to expose other clusters entirely). By using a naïve solution combined with a heuristic-based solution, the team can guarantee it will provide an optimal answer.

C. Solution Proposals

The social network information given by the marketing department takes the form of a graph of friends. Each row of the dataset indicates a bidirectional relationship between two individuals. The goal is to give out one free card per day for 10 days. After giving a card, we can track the friends of the individual who received the card ("exposed" individuals) and determine the probability that they sign up for CactusCard Plus. Strategically distributing these cards in the friend network will maximize the number of sign ups, while never putting the company in a position where it has to give a free card to a friend of an individual who already received a free card (as that is unlikely to net new sign ups).

Two solutions are proposed. The first solution to this search problem uses a simple depth-first search.

D. Test Plan

TODO

E. Solution 1

TODO

F. Solution 2

TODO
TODO PROVE HEURISTIC ADMISSIBILITY

II. Part 2

Part 2 of the project describes another task given by the CEO of the CactusCard Credit Company, wherein a program should be developed to provide "a *machine learning based approach* to identify individuals who should/should not be given the CactusCardPlus." This section will formulate this problem and describe the steps taken to solve it.

A. Problem Importance

The decision whether or not to grant a line of credit (and, as a corollary, how large that line of credit should be) is one of the most important decisions made by a credit agency. In this age of big data, agencies which fail to accurately utilize existing data to make these decisions are likely to fall behind agencies who are doing so. In this problem, CactusCard Credit is fortunate to have a dataset which tracks many credit attributes for a large number of users.

B. Problem Formulation

The problem itself is a machine-learning problem, which at its core simply means a program designed to recognize patterns. Machine learning can encompass a wide variety of problems, which can be divided into "regression" problems and "classification" problems. In this case, the consumer research division has given the team a dataset which contains information about customers *for whom the decision to award or not to award the CactusCard has already been made*. This is important information – because customers can either be given the card or not (a binary *classification*), this falls under the realm of *classification* machine learning problems. Additionally, the fact that the classification program will be trained with data means this is a *supervised* learning problem.

As a supervised classification machine, the program must be able to read the data set and, with that information, draw conclusions about which credit variables

have the largest impact on whether or not the customer was given the card. To be an effective tool for the CactusCard corporation, the dataset supplied by the consumer research division should only include proper decisions – cases where the card was awarded to the correct individual and denied to individuals who would have abused the line of credit. Otherwise, the machine will learn "bad habits" and will draw conclusions from the data which are not useful to the company.

The dataset itself is comprised of 15 credit attributes for the individuals, as well as an indication of whether the individual received a card. The credit attributes are not specified, though the problem specification does include the data types for each attribute. (For example, attribute 1 can be either "b" or "a", and therefore represents some binary information about the customer such as whether they are male or female). These attributes, therefore, are all weighted equally, which may present a disadvantage. Further discussion of this issue can be found in Section IV. Additionally, it is important to note that the dataset is approximately balanced — approximately half of the individuals were granted cards, while half were not. This is helpful in correctly training the classifier.

The goal of this component of the project is to determine an algorithm which will effectively use the dataset to learn how the credit attributes affect the decision to award the card, and from there use that knowledge to predict card decisions within a certain degree of accuracy.

C. Solution Proposals

Weka, a software tool developed at the University of Waikato, is a software package with implementations of a large number of machine learning algorithms. Weka can be used in multiple ways. One option is to simply use the Weka API to access the algorithms from Java code. This is advantageous when one wishes to write a full machine learning system, or integrate machine learning algorithms with an existing Java system. Another option is to use the Weka Explorer, which is a wrapper around the algorithms. This GUI tool allows the user to load a data set and run the algorithms on it, then view the results in various formats.

It is the latter method which is proposed to solve the task given by the CEO. Using the Weka Explorer, we have a straightforward method for gaining insight into the effect of the 15 credit attributes. We can use the Explorer to test various machine learning algorithms, and based on their success rates, determine which should be used in the future to make the actual issue decision. An additional advantage of using this method is that Weka is tested, proven, open-source code. The GitHub repository has nearly 8,000 commits, and therefore represents much more stable code than the team could have produced in the time allotted to this study [**?**]. This provides greater value for the company.

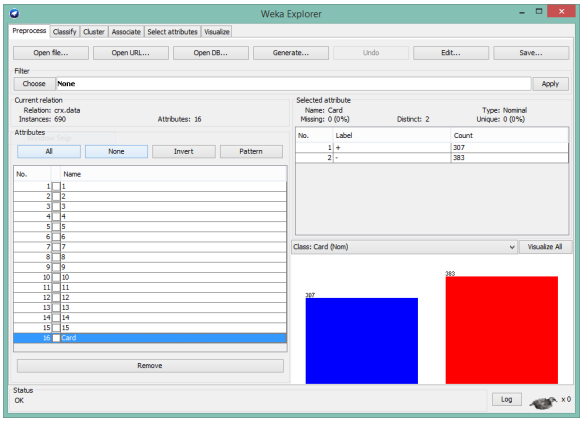

Fig. 1. The Weka Explorer interface

D. Test Plan

The team will evaluate several popular machine learning algorithms in the Weka Explorer. The success of each algorithm will be determined by a number of factors. The team must consider runtime, as the algorithm cannot take an extreme amount of time to complete or it will be useless. The most important factor, however, is the accuracy of the classification. Using random chance to classify card issuance would theoretically have an accuracy rate of approximately 50%, so the team must find an algorithm with a higher percentage than that. To ensure CactusCard's continued financial success, the algorithm must predict card issuance correctly at least 80% of the time. The team proposes this number as it represents a realistic target for algorithms which make mistakes, yet still shows a tremendous potential advantage for CactusCard Credit.

E. Solution 1

Support Vector Machines (SVM) are, according to Russell and Norvig, "the most popular approach for 'off-she-shelf' supervised learning"' [**?**]. SVMs do a good job of classification because they separate groups as much as possible (called a "maximum margin separator"). Additionally, SVMs can work in multiple dimensions (creating a "hyperplane") which will be important with the high number of credit attributes in the data set. Finally, SVMs are considered a non-parametric method, because they store some of the training examples, and yet like parametric models, they often avoid over-fitting the data [**?**].

To use an SVM on the data set provided by the CEO, Weka supports the addition of a library called libsvm. This Java library contains an implementation of the SVM algorithm (called SMO, for sequential minimal optimization) which hooks into the Weka Explorer interface. The team will set the Explorer to use varying numbers of crossfold validation passes, in order to evaluate the classifier. This will split the dataset, train the SVM with a subset of the dataset, and then evaluate the SVM's effectiveness at predicting the card issuance decision by comparing the SVM's predictions to the actual data. The results from running the libsvm algorithm on the dataset are shown in the table below:

Crossfold validations	5	10	15
Time taken to build model (s)	.15	.21	.16
Correctly classified (%)	587 (85.07%)	586 (84.93%)	585 (84.78%)
Incorrectly classified (%)	103 (14.93%)	104 (15.07%)	105 (15.22%)

TABLE I. A TABLE SHOWING THE RESULTS OF THE LIBSVM APPLIED TO THE DATA SET

As we can see from the table, no significant difference appears to exist when adjusting the number of crossfold validation passes. All of the attempts fall well within the expected runtime, and predict with approximately 84% accuracy whether someone was or was not given a CactusCard. This is well within the criteria established above in Section II-D, and thus is a valid contender for the chosen algorithm.

F. Solution 2

Another type of machine learning algorithm is called a Multilayer Perceptron (MLP). This is a type of neural network, which maps from inputs (in this case, the credit attributes) to outputs (whether someone was given a card). The MLP is suitable for this task because it uses *backpropagation*, which is a supervised learning technique. The network is initialized with random initial weights, then backpropagation algorithm minimizes the error function using the method of gradient descent, which in turn recursively updates the initial weights. By using backpropagation, MLPs are able to handle hidden values which are present in this dataset. Given that MLPs handle such hidden values whereas SVMs do not, we intuitively expect that MLPs will better classify the data.

The Multilayer Perceptron is significantly more complex than the SVM, but conveniently Weka does include an implementation in the Explorer. Using the same technique as above, the team will evaluate this algorithm with various levels of crossfold validation to determine the optimal setting. The results are seen below:

Crossfold validations	5	10	15
Time taken to build model (s)	8.46	8.47	8.51
Correctly classified (%)	584 (84.64%)	572 (82.90%)	572 (82.90%)
Incorrectly classified (%)	106 (15.36%)	118 (17.10%)	118 (17.10%)

TABLE II. A TABLE SHOWING THE RESULTS OF THE MULTILAYER PERCEPTRON APPLIED TO THE DATA SET

Once again, it is clear that adjusting the number of crossfold validations has little to no effect on the accuracy of the classifier. As a result, the recommendation of the team is to use 5 validations on data sets of this size, so the algorithm does not take too long to run. While the time taken to build the data model was always nearly the same, the total runtime for all passes of the validation increased dramatically with larger numbers of cross fold validations. Since they appear to have no impact on the effectiveness of the algorithm, there is no need for the program to take that additional time. It is likely that on future datasets with larger numbers of individuals, the runtime would increase further still, reinforcing the decision that 5 crossfold validations is the correct choice.

Although no difference between the columns in Table II is visible, we can see immediately that the Multilayer Perceptron is a good choice for the CactusCard company. In all cases, the success of the classifier was above the criteria set above in Section II-D. The MLP algorithm averaged approximately 83% correctness, which is significantly higher than random chance. This algorithm would be a good choice for future data sets, where it could use the information it learned from this training set to choose who receives a card.

G. Conclusion

In summary, both the vector-based classifier and the neural network succeeded at predicting the credit issuance decision with at least 80% accuracy. Both algorithms are therefore suitable for the company and can be used to make decisions regarding new clients. However, the SVM performs significantly better, as the time taken to build the model is a fraction of the time taken by the multilayer perceptron. As a result, the team recommends the sequential minimal optimization implementation of the support vector machine for CactusCard Credit.

III. PART 3

The third part of the CactusCard project asks us to implement and test two models used to simulate

malicious attacks on the card company so that they can better understand how to defend against credit card attacks (fraud, phishing, theft, etc.).

A. Problem Importance

For a credit card company, security is a top priority. When handling such sensitive data a security breach can have massive consequences. It makes sense for CactusCard to want to try it's best to understand how these attacks work and what methods are best to defend against them. Card companies that have a reputation of not being secure can expect to not be in the business for long.

B. Problem Formulation(DAM model)

The problem here is a game theory problem where we're asked to analyze the performance of two proposed models for defending against attacks. The game theory models here simulate an attacker and a defender trying to maximize their score by increasing the payoff they receive for either successfully defending or attacking. The attacker and defender both utilize a list of attack vectors that determine the outcome of the game. Both players have to pay a $100 cost for each attack vector they utilize, but winning the game results in a $10,000 payoff. For both models their will be 100 possible attack vectors available to the attacker and defender.

One of the proposed models is the Deterministic Adversary with Memory(DAM) model. In this model both attacker and defender will pick a single attack vector per turn. In a given turn the defender is considered the winner if it picks the same vector that the attacker picks in that turn, otherwise the attacker wins. Both players will have memory that is used to store the last five attack vectors used by the attacker. We will use M to represent the size of the memory for the attacker and defender, note both attacker and defender memory is independent from each other. For the first M turns the attacker and defender will pick an attack vector from the list of vectors uniformly at random, completely independent from each other. After the initial moves each player will choose a vector either from memory or a vector that is not in memory. The probability that a player will choose its vector from memory is determined by a parameter S. S ranges from 0.0 to 1.0, a player will have a 100% chance to pick from memory with an S = 1.0.

C. Solution Proposals(DAM model)

For the DAM model CactusCard asks us to find a strategy for the defender that performs better than the defender in the DAM model. Given that the number of attack vectors is 100 and the number of vectors stored in memory at any point is 5, it makes sense for the defender to try to maximize the number of times both defender and attacker pick their vector from memory. Even if the attacker is unlikely to pick from memory (low S value), it should be more beneficial to always pick from memory since the chance that both the attacker and defender will pick the same not-in-memory vector is very small (1/95). When both attacker and defender pick from memory, the probability for the defender to successfully defend jumps to (1/5). It seems that the best strategy the defender can use is to always pick a vector from memory.

D. Test Plan(DAM model)

To test the DAM model alternate strategy, the base DAM model defender strategy will be compared to the alternative defender strategy that we proposed. The base DAM model will be implemented using JAVA. Testing the alternate strategy for the defender will be simple, we can just leave the S parameter for the defender as 1.0 so that the defender always picks its vector from memory. Using the implementation a series of test will be ran analyzing the average payoff for each player for the different S parameters. The same will be done for the alternative strategy. The two strategies will be compared after the average payoff for the players is gathered.

E. Solution 1(DAM model)

Overall the defender performs very poorly in the base DAM model. In the best case scenario, where both defender and attacker have a 100% chance to pick their vector from memory the defender still only blocks roughly 20% of the attacks. We took the average number of attacks successfully defended of 20 simulations and compiled them into the table below:

Defense S \ Attack S	0.0	0.25	0.5	0.75	1.0
0.0	5.4	4.3	2.6	1.4	0.2
0.25	4.3	8.8	12.8	17.9	25.2
0.5	2.3	14.5	24.4	37.3	50.4
0.75	1.8	17.2	35.1	57.7	74.3
1.0	0.2	23.5	46.0	71.2	100.5

TABLE III. AVERAGE NUMBER OF ATTACKS SUCCESSFULLY DEFENDED (OUT OF 500) ATTACKS BY VARYING S PARAMETERS

Since our proposed alternative strategy was for the defender to simply always pick from memory no further testing was needed. This table not only shows the performance of our base model, but also the performance of our proposed alternative strategy for the defender.

Although our alternative model no longer requires an S parameter, its performance can be accurately represented by the table when the defenders S parameter equals 1.0.

Based on these results, I would not recommend using a defender strategy that follows the DAM model.

F. Dominant Strategy

Whenever this game is played, CactusCard cannot make money. Since CactusCard can only lose money, the goal then is to make sure that the game is played for the shortest amount of time possible. The most cost-effective way to make sure the game is not played very long is to make sure the attacker is not making any money, and then minimizing our cost while still ensuring that the attacker is not making money.

So the goal is to defend the number of attacks that will maximize his losses and minimize the money we spend.

The first priority is to ensure the attacker is losing money. This is computed through determining the expected value. Once we get him losing money, we need to find the optimal amount of vectors to defend that will minimize out losses. The following is our findings when the attacker attacks 1, 2, and 10 vectors.

Assuming: attacker = 1, catk = 1000

Cost Results:

Defender ⇒ Assuming they only attack one vector, then we're going to average out to about -$10000 anyway we play, no matter what the other variables are. The best results I found were defending 97 attacks. That way, the attacker will average out at -$797.98 and the defender will average out at -$9902.02.

Assuming: attacker = 2, catk = 500

Cost Results:

Defender ⇒ The attacker's next best move is to attack 2 vectors. This time, the most cost effective way to ensure that he doesn't average out to make any money is to defend 68 vectors. This averages out to the defender losing $41.43 and the attacker losing $41.43 and the defender losing $7758.77

Assuming: attacker = 10, catk = 300

Cost Results:

Defender ⇒ This time the attacker attacks 10 vectors. The best response when the defender is paying $300 per attack is to defend 22 vectors. This still gives a likelihood of defense at 92.96% and averages out the defender to lose $2192.31 per round the defender to lose $2907.69

Overall, the reasonable investment for the company entirely depends on the amount of attack vectors that the attacker decides to play. The fewer he plays, the more expensive it is for the company.

IV. Conclusion

The conclusion goes here. Discuss issues with part 2 dataset – all attributes weighted equally

Appendices

Source code and stuff goes here

Work

We're supposed to document the work done by each team member here.

References

[1] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.