Aubin La

CS427 Paper – AI game Akinator

Introduction

Akinator is a web-based game similar to Twenty Questions. The player has to think of a person or character (fictional or not) and answers yes/no questions. Akinator handles uncertainty, as “probably”, “probably not” and “I don't know” are other possible answers. Using each answers, the program determines the best question to ask next and eventually gives a guess as to who the player is thinking of. It is a very interesting Artificial Intelligence Program that combines elements from decision trees and binary trees, as well as probabilistic methods and machine learning. If the first guess is not correct, Akinator continues to ask questions, and so on up to three guesses, the first one being generally after 15-20 questions. If the third guess is still not correct, the player is asked to add the character into the database. Although the exact algorithm Akinator uses is kept a secret, we can guess that a decision tree is built based on the character entries in the database, and several searches are performed in this tree.

In a way, Akinator emulates human behavior, similarly to a human being playing “Guess Who?”. The characters in “Guess Who?” can be compared to the database of Akinator which is of course a lot bigger. While playing “Guess Who?”, the player will first analyze its data, that is to say look at the specific characteristics of the characters. It is just like Akinator which has a filled character database, with many attributes for each entries. Then, the player has to decide which questions to ask to find the character to guess in the lowest number of questions. From Akinator's point of view, it's looking for the best question that will reduce the search space efficiently. This corresponds to building a decision tree and searching it.

Paper 1: A Probabilistic Approach for Reducing the Search Cost in Binary Decision Trees

This paper presents a probabilistic method that can reduce the complexity of the search for binary decision trees. The Akinator program could possibly be using this kind of method to reduce the number of questions required to decide which character fits the responses the most. The root of the decision tree can be seen as the first question Akinator asks. In reality, the first question is not always the same, but it is always a question that would divide the search space very efficiently, such as “Is this person real?” or “Is this person a man?”. The interior nodes of the decision tree are the following questions the player answers which allow to advance to a next node of the tree. The leaves of the decision tree can be seen as the characters that fit the answers on the path to that leaf. The entire character database should be used to create a tree that is as most binary balanced as possible, allowing to divide the search space by two for every

question. Although having a binary tree reduces the cost of the search, it would still be a very high cost to ask every question that gives an element corresponding to a character.

This paper's introduction highlights the fact that in some problems, decisions are made under uncertainty. This is exactly the case of the Akinator program. As the player answers questions, the program has to eliminate the characters that do not fit the answer given so far. If the data is represented by a decision tree, giving an answer should mean advancing to a next node. The cost of gathering all the information about the character the player is thinking of is high and should require asking too many questions. Therefore the search cost has to be reduced in some way. In this paper, the authors propose a probabilistic method to reduce the search cost, based on past experience. As games are played, Akinator collects a lot of data that could be used. The idea is to skip some decisions for certain nodes, that is to say not ask the question of that particular node if based on experience the probability of taking this path is high enough.

A gain function is introduced. It can be seen as how much the search cost can be reduced. It is reduced the number of decisions that are taken automatically via statistical data. By doing this, there are risks that it was a wrong decision therefore the cost can be increased by the backtracking cost if the statistical decision was wrong. For the gain to be maximum, the decision tree nodes have to be optimally partitioned, that is to say the statistical nodes have to be chosen carefully. Also the backtracking cost has to be reduced.

The conclusion is that using probabilities, it is possible to reduce the search cost in a binary decision tree. The method works best when the tree is perfectly binary balanced, which is hard to achieve for Akinator.

Paper 2: Dynamically Scripted Artificial Intelligence

This paper explains how it is possible to modify persistent data structure to allow an AI program to store or modify data. Akinator has for sure a database that stores every character it knows and their specific characteristics. The authors express the idea that in order for the AI program to have a more human like behavior, it has to be able to expand its knowledge dynamically, as opposed to having everything coded manually by a human.

They take the example of a strategy game and the development of new tactics. A limitation is the fact that for gaining new knowledge, the program has to be in some kind of observation or learning mode. For Akinator it simply means that it has to fail one or several times before it can guess a character that was not in the database before. However, Akinator overcomes this limitation in a way. When the probability of having found the character is high enough, the program asks some questions that seem irrelevant to the player, only to gain more knowledge about the guessed character for future games. Changes in the database might also change the decision tree and therefore the behavior of the program in terms of the order of the questions.

Although Akinator is depicted as a genie that has more knowledge than a human being, a certain emphasis is put on the human like interaction the player has with the program. For example, through animations: the genie appears to be thinking while the program is searching for the next, most appropriate question.

It is also noticeable that some scripting has been implemented in Akinator to respond to some special case scenario, such as answering “i don't know” to all questions.

By working on dynamic learning, it is possible to improve both gaming behavior and also work on human like robotics. Even though Akinator has a quite efficient learning capability, it lacks some human like behavior, for example when it backtracks, some questions are repeated or not relevant.

Paper 3: A Fast Decision Tree Learning Algorithm

Decision tree learning algorithms produce a decision tree using data. Because data sets can be very large in AI programs, it is of high interest to use decision tree learning algorithms that scale well. In this paper, a new, faster decision tree algorithm is described and compared to C4.5 algorithm.

The proposed algorithm searches the entire model space using a heuristic. Experiments were conducted to compare their new method called Naive Tree Algorithm to C4.5 and Naive Bayes algorithms. It performs as well as C4.5 in terms of accuracy and outperforms it in time complexity.

This paper is an interesting example of how decision trees can be built and how that process can be speed up. It highlights the importance of scalability as data sets can be very large nowadays.

Paper 4: General Criteria on Building Decision Trees for Data Classification

This paper talks about the criteria used to build decision trees, in particular for ID3 algorithm. The introduction reminds us that to classifying data means looking for common characteristics in the data set. There are many applications that use data classification, decision trees are one of them. In a decision tree, each leaf node has a class label, which for Akinator would be the name of the character and therefore contain only one element. The other nodes, called decision nodes, split the data based on a specific feature. The number of generated children is equal to the number of values the splitting feature can take. Since each decision is a yes/no questions, the decision tree will be binary. Since it is possible to answer “probably”, “probably not” or “I don't know”, Akinator surely is more complex than this.

ID3 algorithm is built in a top-down recursive manner. It tries to minimize the number of test need to classify an object. To do so, the splitting criteria has to have the highest information gain.

Four factors that influence the building of a decision tree are examined: the number of sub-attributes in each attribute, the number of samples in each sub-attribute, the number of samples for each class in one sub-attribute and the number of classes. The sub-attributes are the values that a splitting attribute can take. Having a higher number of sub-attributes can increase performance as the information gain increases. However, Akinator uses yes/no questions therefore we can assume that the number of sub-attributes is always two. The number of samples in each sub-attribute is interesting for Akinator. The more unbalanced it is, the less effective that attribute will be to split the data. Examples are given to show that when one of these factors change, the decision tree may also change. This is relevant in Akinator. As the database grows, the attributes have to be reevaluated to ensure that the decision tree is still the best and otherwise modify it.

Paper 5: Data Mining Criteria for Tree-Based Regression and Classification

In this paper, the claim is that it could sometimes be better if a splitting criteria did not equally split the data. When playing Akinator, we notice that the first questions surely divide the data equally, with very general questions such as “Is your character real?” or “Is your character related to music?”. Once it has classified it generally, more specific questions are asked and we can guess that these specific questions don't split equally. It is even more noticeable when, for example, it has come to the point where it knows that the character is in a certain music band. The next questions would be very specific questions that are true only for one member of that band. This means that the part of the decision tree it reached is a highly unbalanced one.

Experiments were conducted to compare their one-sided criteria and CART which is an a decision tree learning algorithm that produces binary trees. The trees that are produced are more interpretable.

Conclusion

Decision tree learning algorithms are widely used and data sets can be very large. The problematic of classifying data is very important for Artificial Intelligence applications. Various methods to classify data exist, such as ID3, C4.5 or CART, the latter producing a binary decision tree. Usually when using these algorithms, the elements are classified and several elements belong to the same class. Akinator takes the classification further by going deeper and having only one element per class, which would be a character or person. This forces the search algorithm to be even more efficient, by using probabilities for example.

With the use of probabilities, Akinator is more robust to errors and is able to make a good guess even when a given answer was wrong. Of course, it will perform badly if several wrong answers are given or if too many uncertain answers are given.

Bibliography

Paper 1: A Probabilistic Approach for Reducing the Search Cost in Binary Decision Trees, Athanasios Rontogiannis and Nikitas J. Dimopoulos, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. 25. NO. 2, FEBRUARY 1995.

Paper2: Dynamically scripted artificial intelligence, Nathan Dawson and Hunter Hale, ACM-SE 43 Proceedings of the 43rd annual Southeast regional conference - Volume 1, Pages 59-60, 2005.

Paper 3: A fast decision tree learning algorithm, Jiang Su and Harry Zhang, Faculty of Computer Science, University of New Brunswick, NB, Canada, AAAI'06 Proceedings of the 21st national conference on Artificial intelligence - Volume 1, Pages 500-505, 2006

Paper 4: General Criteria on Building Decision Trees for Data Classification, Yo-Ping Huang and Vu Thi Thanh Hoa, National Taipei University of Technology, ICIS '09 Proceedings of the 2nd International Conference on Interaction Sciences: Information Technology, Culture and Human, Pages 649-654, 2009

Paper 5: Data Mining Criteria for Tree-Based Regression and Classification, Andreas Buja (AT&T Labs, Florham Park, NJ) and Yung-Seop Lee (Dongguk University, Korea), KDD '01 Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, Pages 27-36, 2001.