CpSc810 – Goddard – Notes Chapter 4

Adversarial Search & Information Theory

4.1 Adversarial Search

We examine the problem of trying to find a hidden object using the least resources. Consider the game 20 Questions: you have 20 questions to determine an object. What is the best strategy? Well, you could start by asking whether it is a mammal with black and white stripes: this is great if the answer is yes, but useless if the answer is no. It is better to ask a question where either answer narrows the candidates down. In other words, ask a question where either answer provides a lot of information.

The mathematical definition of information is called ***entropy***. The basic observation is that if there are n candidates, and each question has two answers, then there is always the possibility of taking at least

$$\log_2 n \text{ questions,}$$

since, for each question there is always an answer that leaves at least half the possibilities. (It can further be shown that one needs this many questions on the average.)

On the other hand, if we can enumerate the universe, there is always a question that splits the universe into equal (or almost equal if n is odd) pieces. So we can always solve the problem with $\lceil \log_2 n \rceil$ questions (provided there are no restrictions on the questions used).

We assumed two answers to each question. If each question has a possible answers, then we need

$$\log_a n \text{ questions.}$$

EXAMPLE 4.1. *Suppose you have 9 coins, eight of which are identical and one is lighter. The task is to use a balance to determine the light coin using two weighings.*

The solution is simple enough, but information theory can guide us. There are 9 coins, each question has 3 answers, so at least $\log_3 9 = 2$ *weighings are needed. In fact, to do it in two weighings, this shows that each weighing must split the candidates perfectly. There is only one first weighing that does this: 3 on each side. This leaves 3 possibilities. And then weighing 1 coin on each side finds the light coin.*

4.2 Classification

A typical AI problem is classification of some set (for example, animal classification). The resultant knowledge can be stored as a decision tree (which is largely propositional). This is sometimes called symbolic learning.

Example 4.2. *A simple way to create decision trees was exploited by the early ANIMAL program. It tries to identify an animal by asking a sequence of questions, and finally providing an animal. If successful, then nothing happens. Otherwise, it prompts the user for a question to distinguish the actual solution from what it thought the answer was. The program then replaces that leaf of the decision tree by a node with two children whose content is the supplied question.*

A generic approach is given by the tree induction algorithm ***ID3***. The situation is that there are many attributes, and the algorithm tries to find the best attribute split in the training set. Here is the algorithm in summary:

```
Tree Induction
    Find best property
    Partition input as per property
    Recurse
```

If there are only two values of each attribute, then the best split is clear. But, in general, to find the best attribute, the algorithm uses information theory. If the split creates sets with proportions/probabilities $\{p_i\}$, then the Shannon entropy or information is given by

$$I = \sum_i -p_i \log_2 p_i$$

with the bigger the better.

Example 4.3. *Classification of animals.*

Animal	*Fur*	*Color*	*Small*	*Herbivore*
Kodiak	*yes*	*brown*	*no*	*no*
Lion	*yes*	*yellow*	*no*	*no*
Aardvark	*yes*	*brown*	*yes*	*no*
Canary	*no*	*yellow*	*yes*	*yes*
Salmon	*no*	*pink*	*yes*	*no*

Attribute: Herbivore $I = -\frac{4}{5}\log\frac{4}{5} - \frac{1}{5}\log\frac{1}{5} \approx 0.72$

Attribute: Color: $I = -\frac{2}{5}\log\frac{2}{5} - \frac{2}{5}\log\frac{2}{5} - \frac{1}{5}\log\frac{1}{5} \approx 1.52$

Problems include missing, noisy, multivalued or continuous data. Of course, there is also the bias towards attributes with several values.

4.3 Application: CodeBreaker

One game involving adversarial search is Mastermind/CodeBreaker. One approach to this is to always pick a guess which splits up the universe as evenly as possible. In particular, one might minimize the maximum piece remaining (or equivalently maximize the minimum information obtained). Knuth showed that this solves the standard 6-color 4-hole game in 5 moves.

Exercises

4.1. A bot must determine a hidden value by asking a series of questions. The value is taken from the universe $\{A, B, C, D, E, F, G, H\}$ and there are four possible questions $\{Q, R, S, T\}$. The goal is to minimize the average number of questions. The questions have the following results:

	Q	R	S	T
A	*1*	*1*	*1*	*3*
B	*1*	*1*	*2*	*3*
C	*2*	*1*	*3*	*3*
D	*2*	*1*	*1*	*3*
E	*3*	*2*	*2*	*3*
F	*3*	*2*	*3*	*2*
G	*3*	*2*	*1*	*3*
H	*3*	*2*	*1*	*2*

(a) Calculate the entropy of question Q.

(b) Draw an optimal decision tree.

4.2. Recall that Hangman entails guessing letters to determine a secret word. Each time you guess a letter in the word, where the letter appears is revealed. Each time you guess a letter not in the word, you lose one life. You win if you guess the whole word before running out of lives. Using information theory, write a program to play Hangman .

4.3. *Produce a good decision tree for the game show Lingo.* The task in the game show is to determine a 5-letter word in at most 5 guesses. Each guess is a word. The response indicates each letter that is correct and in the right position; as well as each letter that is correct but in the wrong position. The first letter of the word is given at the start. Each guess must be a valid word starting with that letter.