CS 4244 Project – Stage 2

Due on the last day of classes; i.e. April 17, 2020

1 Objective

The objective of this project is to demonstrate learning of the material in CS 4244. To this end, you have an option to choose among the following three projects. I have marked every project with my ratings of difficulty, but remember, your mileage may vary.

Programming Language

You are free to choose your preferred programming language.

Honor Code

You are allowed to consult internet but the analysis (theoretical or empirical) as well as the code must be your own. In other words, you are not allowed to copy a code or paraphrase text on the internet in your report.

2 Behavior of Random CNF Formulas: (Difficulty: Easy)

Let $X = \{X_1, \cdots, X_n\}$ be a set of propositional variables and let F be a formula defined over X. A *satisfying assignment* or *witness* of F is an assignment of truth values to the variables in X such that F evaluates to true. Let $\#F$ denote the number of satisfying assignments of F. We say that F is *satisfiable* (or *sat.*) if $\#F > 0$ and that F is *unsatisfiable* (or *unsat.*) if $\#F = 0$.

A k-clause is the disjunction of k literals out of $\{X_1, \cdots, X_n\}$, with each variable possibly negated. For fixed positive integers k and n and a nonnegative real number r, let the random variable $F_k(n, rn)$ denote the formula consisting of the conjunction of $\lceil rn \rceil$ k-clauses, with each clause chosen uniformly and independently from all $\binom{n}{k}2^k$ possible k-clauses over n variables.

The goal of this project is to conduct an empirical study followed by an analysis of the behavior of the satisfiability of random variable $F_k(n, rn)$. You can fix value of $n = 150$. You should experiment with $k \in [3, 5]$ and $r \in (0, 10)$ with different values of r at the intervals of 0.2. You can use any state of the art SAT solvers available. (For starters, you can use Crypto-MiniSAT). Once you fix k and r, you should sample $F_k(n, rn)$, i.e., generate several formulas and check if the generated formulas are satisfiable or not. For every fixed k and r, generate 50 formulas and calculate the probability of satisfiability of $F_k(n, rn)$ for fixed k,n,r as the number of satisfiable formulas divided by the total number of formulas. For every k, plot a curve of satisfiability of $F_k(n, rn)$ vs. r. Explain the behavior that you observe. Try to be as rigorous as possible.

3 Encoding of Bayesian Networks (Difficulty: Between Easy and Moderate)

The goal of this project is to encode the probabilistic inference query over a Bayesian network into a weighted counting problem. Your program should take the Bayesian network (*model*) and a query (*evidence*) as input and return a weighed counting instance. Below are the formats for inputs and outputs:

1. Model Format:
   We use the simple text file (with a .uai suffix) format specified below to describe problem instances (Bayesian networks).

   Structure
   A file in the UAI format consists of the following two parts, in that order:
   <Preamble>
   <Function tables>

   The contents of each section (denoted < ... > above) are described in the following:

   Preamble

   Our description of the format will follow a simple Bayes network with three variables and two functions. A sample preamble for such a network is:

```
BAYES
3
2 2 3
2
2 0 1
2 1 2
```

The preamble starts with BAYES (since the network is a Bayesian network).

The second line contains the number of variables. The next line specifies the cardinalities of each variable, one at a time, separated by a whitespace (note that this implies an order on the variables which will be used throughout the file). The fourth line contains only one integer, denoting the number of CPTs in the problem. Then, one CPT per line, the scope of each CPT is given as follows: The first integer in each line specifies the number of variables in the CPT, followed by the actual indexes of the variables. The order of this list is not restricted. Note that the ordering of variables within a CPT will follow the order provided here.

Referring to the example above, the first line denotes the Bayesian network, the second line tells us the problem consists of three variables, let's refer to them as X, Y, and Z. Their cardinalities are 2, 2, and 3 respectively (from the third line). Line four specifies that there are 2 CPTs. The first CPT is X,Y, while the second CPT is Y,Z. Note that variables are indexed starting with 0.

CPTs

In this section each CPT is specified by giving its full table (i.e, specifying value for each assignment). The order of the CPT is identical to the one in which they were introduced in the preamble, the first variable has the role of the 'most significant' digit. For each CPT, first the number of entries is given (this should be equal to the product of the domain sizes of the variables in the scope). Then, one by one, separated by whitespace, the values for each assignment to the variables in the function's scope are enumerated. Tuples are implicitly assumed in ascending order, with the last variable in the scope as the 'least significant'. To illustrate, we continue with our Bayes network example from above, let's assume the following conditional probability tables:

X	P(X)
0	0.436
1	0.564

Y	Z	P(Z,Y)
0	0	0.210
0	1	0.333
0	2	0.457
1	0	0.811
1	1	0.000
1	2	0.189

Then the file will have the content:

− −

```
2
0.436 0.564

6
0.210 0.333 0.457
0.811 0.000 0.189
```

− −

(Note that line breaks and empty lines are effectively just a whitespace, exactly like plain spaces “ ”. They are used here to improve readability.)

2. Weighted Formulas: You should output a CNF formula in DIMACS format and another “weight file” that maps weights of all the literals. The header of “weight file” should be p < *# of variables*> followed by every row that starts with w, followed by literal and its weight(any positive real number) and then a 0 to end the line.
   Here is a toy example:

   ```
   p 2
   w 1 0.1 0
   w -1 0.2 0
   w 2 0.8 0
   w -2 1.0 0
   ```

   This file specifies that there are two variables, and each of the lines specifies the weights of literals.

4 Proof of Unsatisfiability(Difficulty: Moderate)

Modify the SAT solver you have implemented so that if the formula is unsatisfiable, the solver should return a proof of unsatisfiability using resolution. The solver should print out the proof in a file, where the first line should begin with “v <# of clauses>” where <# of clauses> specifies the number of clauses in your resolution proof. Then the next <# of clauses> lines should list down all the clauses that are used in resolution proof. An empty clause is also a clause; you can use the symbol -1 to specify the empty clause. Note that this ordering also assigns a unique id to every clause, which is the line number of the clause. The first clause gets id 1.

Following the list of all the clauses, you should list down the actual proof in the following format. Every line should first list the id two clauses used in resolution and then the clause that was generated. (E.g., "1 2 3" would represent clause 1 and clause 2 were used in resolution, and clause 3 was produced). The last line should end with the empty clause being generated.

Here is an example proof of unsatisfiability for $(1 \lor 2) \land (1 \lor -2) \land (-1)$:

```
v 4
1 2
1 −2
−1
1
1 2 4
3 4 -1
```

5 Final Submission

Your final submission should include:

1. The source code, along with a verbose explanation of your design choices. Remember, when you are working as a software engineer, your code will never make it to production if your peers can not review it.

2. In addition, you should submit a written report of at most ten pages (excluding references), including:

   (a) You should include a written report of a verbose summary and an analysis of your findings. Your analysis can be intuitive, but you need to come up with an explanation of your findings.

   (b) I believe that you will be a leader in your field in a few years. As a leader, you are expected to evaluate yourself. You should include what letter grade would you have assigned to yourself if you were the instructor. Finally, your report should end with a paragraph explaining what you learned from this project.