Assignment 2

COMP9418 – Advanced Topics in Statistical Machine Learning

Lecturer: Gustavo Batista

Last revision: Monday 28th October, 2019 at 14:19

Instructions

Submission deadline: Monday, November 18th, 2019, at 14:00:00.

Late Submission Policy: The penalty is set at 20% per late day. This is ceiling penalty, so if a group is marked 60/100 and they submitted two days late, they still get 60/100.

Form of Submission: This is a group assignment. Each group can have up to three students. Write the names and zIDs of each student in both the report and Jupyter notebook. **Only one member of the group should submit the assignment**.

The group should submit your solution in one single file in zip format with the name solution.zip. There is a maximum file size cap of 5MB, so make sure your submission does not exceed this size. The zip file should contain one Jupyter notebook file and one pdf file. The Jupyter notebook should contain all your source code. Use markdown text to organise and explain your implementation. The pdf file is a 5-page report summarising your findings. The report can include text and plots to illustrate your results.

Submit your files using give. On a CSE Linux machine, type the following on the command-line:

```
$ give cs9418 ass2 solution.zip
```

Alternatively, you can submit your solution via the course website.

Recall the guidance regarding plagiarism in the course introduction: this applies to this homework, and if evidence of plagiarism is detected, it may result in penalties ranging from loss of marks to suspension.

Introduction

The objective of this assignment is to develop a collection of Python functions that together constitute a library for representation and inference of Graphical Models.

This project has three main parts:

1. Notebook. The source code of the functions should be organised in a Python notebook. The Python notebook will present the main functionalities of your library. It should also provide a cell, for each library main functionality, that contains a simple demonstration/test code. The demonstration/test code shows how to call the function(s) responsible for the task correctly and the output provided by the function(s) as well as a comparison to an expected output.
2. Benchmark. You will benchmark the implementation using Bayesian networks of different dimensions. The bnlearn repository provides a collection of discrete Bayesian networks with a different number of nodes. You are free to select one Bayesian network from each category (small, medium, large, very large and massive) for your tests. Your Python notebook should organise the execution and reporting of results of every benchmark test you performed. Therefore, your notebook will be the primary tool we will use to replicate your results.

3. Report. Write a brief (5-pages or 5,000 words) describing the results you obtained in the benchmark. Compare the methods you implemented in terms of accuracy of the probability estimates (when these methods provide approximate results) and runtime. Show how the results scale with networks of different sizes.

Libraries and source code

You can use any source code provided by us or developed by you in the tutorials. You are also free to use any of the libraries we used in the practical parts of the tutorials. For reference, these libraries are:

1. graphviz
2. collections
3. itertools
4. tabulate
5. numpy
6. pandas
7. scipy
8. matplotlib
9. math

Lightly document your source code indicating for each function: their expected inputs, outputs and possible side effects.

[10 Marks] Representation

Your code needs to represent a Bayesian network specified by a user or loaded from a file. You also need to save the Bayes net to a file. There are some popular file formats for networks. Implementing, at least a subset, of one of these file formats is useful to work with the network files in the benchmark part of this assignment.

For instance, the hugin file format (.net) is a text file with two main sections. The first one specifies the nodes. For instance:

```
node HISTORY
{
  states = ( "TRUE" "FALSE" );
}
node LVFAILURE
{
  states = ( "TRUE" "FALSE" );
}
```

The second part specifies the probability tables, and has the following format:

```
potential ( HISTORY | LVFAILURE )
{
  data = ((0.9 0.1)(0.01 0.99)) ;
}
```

Although hugin and other Bayesian network file formats are poorly documented, they are text file formats not challenging to reverse-engineer.

The representation part of your assignment should have at least these functionalities:

1. Insert and remove nodes.

2. Connect and disconnect nodes with edges.
3. Specify probabilities for a node.
4. Save Bayesian network to a file.
5. Load Bayesian network from a file.

[20 Marks] Pruning and pre-processing techniques for inference

As inference in Bayesian networks is an NP-Hard problem, we frequently rely on pruning, heuristics and pre-processing techniques to speed-up inference. In this section, we list sets of functionalities in this category and their respective marks. You are free to choose any categories you want. However, consider groups that can be evaluated together so your benchmark report will be more interesting.

1. [10 Marks] Network pruning techniques based on query structure. These techniques are composed of edge and node pruning.
2. [10 Marks] Min-fill heuristic for variable order elimination. Deterministic and stochastic composition of min-fill heuristic with min-degree heuristic.
3. [10 Marks] The four optimal elimination rules for prefixes.
4. [10 Marks] Depth-first search to find an optimal elimination order with lower-bound pruning. Use degeneracy to provide a lower-bound.

[25 Marks] Exact inference

For exact inference, implement the Jointree Algorithm using the Shenoy-Shaffer architecture. You need this minimal set of functionalities:

1. A representation for the jointree that can be specified by the user
2. A function that converts an elimination order into a jointree
3. A function to set evidence
4. A function to answer a query based on the jointree clusters
5. Four functions that provide the following jointree transformations: add a variable, merge clusters, add a cluster and remove a cluster.

[25 Marks] Approximate inference

For approximate inference, you have two options:

[25 Marks] Iterative Joingraph Propagation (IJGP)

These are the functionalities for the IJGP algorithm:

1. A representation for the joingraph that can be specified by the user
2. A function that creates a Bethe cluster graph and uses it as a joingraph
3. A function to set evidence
4. A function to answer a query based on the joingraph clusters

[25 Marks] Gibbs sampling

These are the functionalities for Gibbs sampling. Your implementation should run on multiple chains.

1. A function that let the user specify the number of chains.

2. A function to mix up chains until convergence.
3. A function that samples the chains for a specified number of samples.
4. A function to answer queries based on the samples.

[20 Marks] Report

Write a five-page report (around 5,000 words) summarising your findings in this assignment. Use the benchmark Bayesian networks to compare the exact and approximate inference methods as well as the pruning, heuristic and pre-processing techniques. Use plots to illustrate your results.

Discuss the difficulties and challenges of this assignment. For instance, were all inference techniques able to provide answers for all networks in feasible time or you had to set a time out (for instance, maximum number of interations) for certains methods?