BMI 826. Computational network biology. Homework 1
Due Oct 16th, 2018

Instructions

In this assignment we will review concepts on probabilistic graphical models and how they can be used to learn molecular networks. There are 7 problems in all, the first 6 are required and Problem 7 is extra credit/optional. Problem 5 is an implementation assignment and needs to be submitted by Oct 16th 5:00pm via the biostat servers
`mi1.biostat.wisc.edu` or `mi2.biostat.wisc.edu`.
You should have accounts created on these servers. Please see detailed submission instructions for Problem 5. Answers to all other questions are due in class on Oct 16th.

Problem 1 (10 points). Defining concepts.

For each of the terms below provide a brief description of what they mean.

- Probabilistic graphical model
- Regulatory network
- Conditional independence
- Regression tree
- Precision-recall curve
- Lasso
- Precision matrix
- Conditional probability table
- Markov blanket
- Structure learning

Problem 2 (10 points). Probabilistic Graphical Model types.

Listed below are pairs of models that could be used to represent a network. Compare and contrast each model in a pair describing one strength and weakness of each approach. Briefly describe a structure learning algorithm for each model type.

(a) Correlational networks versus Gaussian Graphical Models.

(b) Dependency network versus Bayesian network.

Problem 3 (20 points). Probabilistic graphical models for gene expression data.

For each algorithm below describe what each algorithm takes as input, produces as output, and what it does.

(a) Sparse candidate

(b) Module networks

(c) GENIE3

(d) Given gene expression data for 5,000 genes, 200 experiments which algorithm would you use to reconstruct a gene regulatory network? Explain your choice.

(e) Now assume a biologist told you that there are some genes that are regulators. How will you incorporate this information in the network inference problem?

Problem 4 (10 points). Bayesian networks.

You are given the following set of observed measurements in **Table 1** for each of the five binary random variables, A, B, C, D and E and the following partially known Bayesian network structure. This Bayesian network is partially known because we do not know the parents of A yet.

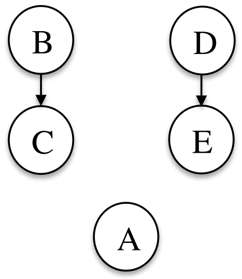

Figure 1: A partial Bayesian network for five boolean random variables

(a) Using the Bayesian network structure of **Figure 1**, estimate the relevant conditional probability tables for B and C.

A	B	C	D	E
0	1	1	1	1
1	1	0	1	0
1	0	0	1	0
1	0	0	1	0
0	0	0	1	1
1	0	0	0	1
1	1	1	0	1
0	0	1	0	1
0	0	1	1	1
0	1	1	0	0

Table 1: Samples of observations for the random variables

(b) Describe a score-based procedure to determine the parents of A. Assume that A can have no more than 2 parents and the parents can be any of the other variables. Assume that the rest of the network is the same as in **Figure 1**.

(c) Briefly describe a greedy structure learning algorithm for Bayesian networks. Describe one advantage and weakness of a greedy structure search learning algorithm.

(d) What would you change in your Bayesian network representation and learning if A's values could be one of five different colors, RED, BLUE, GREEN, YELLOW, CYAN?

(e) What would you change in your Bayesian network learning if the variables were continuous variables?

Problem 5 (25 points). Comparing the structure of two graphs.

Problem description. The goal of this problem is to write a program, `compareGraphs`, that will compare the structure of two graphs, $G_1 = (E_1, V)$ and $G_2 = (E_2, V)$, using precision, recall and F-score. Assume G_1 is the true graph and G_2 is the predicted one. E_1 and E_2 are the edge sets and V is the set of vertices. Precision is defined as $p = \frac{|E_1 \cap E_2|}{|E_2|}$, Recall is defined as $r = \frac{|E_1 \cap E_2|}{|E_1|}$, and F-score is defined $\frac{2*p*r}{p+r}$. F-score is the harmonic mean of precision and recall and provides a concise way to report how similar a set of predictions are to the true set of predictions.

`compareGraphs` should take in three arguments, two are the file names of the two graphs and the third binary input to denote whether the graph is directed or undirected. Hence, it should be able to handle both directed and undirected graphs. The exact usage of your program will depend on the language but it should be command line runnable. For example, if you compiled your code into an executable `compareGraphs`, the usage for two undirected graphs should be as follows:

```
./compareGraphs graph1.txt graph2.txt 0
```

and if the graphs are directed should be as follows:

```
./compareGraphs graph1.txt graph2.txt 1
```

The format of `graph1.txt` and `graph2.txt` is a tab-limited file, with one line per edge. Each is tab-delimited with two columns, one for each of the vertices of the edge. See `http://pages.discovery.wisc.edu/~sroy/teaching/network_biology/fall2018/homeworks/hw1/factors_net.txt` for an example of the input graphs.

The program should output the precision, recall and F-scores when comparing the two input graphs in the following format

```
Precision:  0.5 Recall:  0.5 F-score:  0.5
```

Submission instructions. Please put your code into `/u/medinfo/bmi826/2018-hw//hw1/username` where username is your user name you use to login to this machine. You can use either C++, Java, Perl, or Python to implement this program. If you want to use R, please make sure to write an R command line script for me to run your program. Note, I am most familiar with C++. Your program should be executable on `mi1.biostat.wisc.edu` or `mi2.biostat.wisc.edu`. I will run this using one of the following versions depending upon the language.

```
C++:  ./compareGraphs graph1.txt graph2.txt 0
Java:  java ./compareGraphs graph1.txt graph2.txt 0
Perl:  perl ./compareGraphs graph1.txt graph2.txt 0
```

Problem 6 (25 points). PGMs in practice.

In this assignment, we will explore some practical applications of Bayesian network and Graphical Gaussian Model learning algorithms and learning PGMs from data. The assignment requires a little bit of MATLAB familiarity.

Instructions.

1. Download required data and MATLAB scripts from `http://pages.discovery.wisc.edu/~sroy/teaching/network_biology/fall2018/homeworks/hw1/prob6.tgz`. You would need to run these programs on a computer that has MATLAB installed. The class servers `mi1.biostat.wisc.edu` and `mi2.biostat.wisc.edu` should have MATLAB installed.
2. Unzip this file as follows `tar -xvzf prob6.tgz`. This will result in the directory `prob6`.
3. List the contents of this directory, `ls prob6` and you should see two directories `code` and `data`. `code` has the list of MATLAB programs that can be executed within the MATLAB prompt. `data` has two datasets, `dataset1` and `dataset2`. In each dataset-specific directory, the `_data.txt` file has the data to be used to learn a network and `_net.txt` has the Bayesian network that was used to generate these data.
4. Start MATLAB by typing `matlab`
5. Set the current directory to `code` as all the scripts are in that directory.
6. Execute the scripts listed below as described in the subsequent question parts. Note, for each script, you can type `help <script_name>` and you should be able to see a basic explanation of the inputs of the script.

Problem description

(a) We will first apply the Sparse Candidate algorithm to infer a Bayesian network on `dataset1` available in the files you have downloaded in `prob6.tgz`. Apply the `bnetLearn.m` script on `data/dataset1`. This can be run within MATLAB as follows.
`bnetLearn('dataset.txt',k,'outputgraph.txt')`
This script calls the sparse candidate structure learning algorithm and takes in as argument the file path for the input dataset (`'dataset.txt'`), the number of possible candidates (`k`), and the file path for the output learned network (`'outputgraph.txt'`). Experiment with different sizes of parents, $k \in \{2, 3, 4\}$. Use the program, `compareGraphs` you wrote in Problem 5 to compare the true and the predicted graph and report the corresponding precision, recall and F-score for each candidate size. What can you conclude from these results?

(b) We will use Sparse Candidate again but we will first learn a Dependency Network and use that as input for constraining the network structure. Apply the code `bnetLearn_MB.m` to learn a Dependency network constrained Bayesian network on `dataset1`. This can be run in MATLAB as
`bnetLearn('dataset.txt',mbtype,'outputgraph.txt')`
Here `mbtype` is the type of Markov blanket correction that needs to be used. If `mbtype=0`, the `OR` correction is used and `mbtype=1`, the `AND` correction is used. Compare the output networks learned using these two settings using your `compareGraphs` program and report the precision, recall and F-score. Which one gives you a better network? Discuss these results against the precision, recall and F-score when using a fixed k candidate parents?

(c) We will compare the two algorithms for learning a GGM using the data we used. Apply the code `GGMLearn.m` to learn an undirected graphical model. This can be run in MATLAB as
`ggmLearn('dataset.txt',lambda,algo,'outputgraph.txt')`
`lambda` is the regularization parameter for the sparsity of LASSO. `algo` is 0 or 1 to determine the algorithm that will be used for learning the GGM model. `algo=0` will select the Graphical LASSO algorithm, while `algo=1` will use the neighborhood selection algorithm. Experiment with `lambda` $\in$ {`0.1, 0.3, 0.5,0.7,0.9`} for the two algorithms and report the precision, recall and F-score numbers for each value of lambda for both algorithms. Note that the GGM will output an undirected graph and hence you should compare the true and inferred networks as undirected graphs. What do you observe in terms of the effect of `lambda` on the structure of the network inferred? Which algorithm seems to work best here? Please discuss your observations.

Problem 7 (25 points). Extra credit PGMs in practice.

Using the instructions and scripts described in Problem 6, compare Sparse Candidate, Markov blanket constrained Sparse Candidate and the GGM models on `dataset2`.