

AML PROJECT 2 REPORT

Probabilistic Graphical Models

Markov Networks

DataSet

Hepatitis Dataset

Source:

G.Gong

(Carnegie-Mellon University)

Bojan Cestnik

(Jozef Stefan Institute)

By

Manisha Tadikonda

(50207628)

Puneeth Pepalla

(50206906)

Aim: The goal of this project is to implement and evaluate algorithms pertaining to inference with Probabilistic Graphical Models (Markov Networks) and use that model/algorithm to make meaningful deductions on a dataset.

Approach: The project is done in python by using a library called PGMPY. For this project we use Undirected probabilistic graphical models (Markov Networks). Markov Models are best used when the variables are connected to each other. For example, image segmentation, Natural Language Processing etc. A Markov network or MRF is similar to a Bayesian network in its representation of dependencies, the differences being that Bayesian networks are directed and acyclic, whereas Markov networks are undirected and may be cyclic. Thus, a Markov network can represent certain dependencies that a Bayesian network cannot (such as cyclic dependencies); on the other hand, it can't represent certain dependencies that a Bayesian network can (such as induced dependencies). The underlying graph of a Markov random field may be finite or infinite.

Domain: The dataset selected for this project is Hepatitis dataset. The dataset is from a medical domain. So, a lot of domain knowledge is required in the medical field to estimate the factors and dependencies to form a Markov network. There are 16 different variables in the dataset. So we spent first couple of days to study the different variables in the dataset and learn the dependencies among them. After obtaining good domain knowledge about the dataset and its variables, we proceeded further with the model.

Dataset Description: The hepatitis dataset is obtained from UCI repository. The dataset is provided by G.Gong from Carnegie-Mellon University Bojan Cestnik from Jozef Stefan Institute. The dataset contains 16 variables and there is one final variable class that says if the patient is dead or alive. And some of the variables in the dataset exhibit some influences on the others. The default dataset contains some null values and noise. We first cleaned the dataset and performed some set of transformations on the dataset. This includes categorizing some continuous data into discrete data by including some intervals.

Attribute information:

1. Class: DIE, LIVE (When the patient is alive or dead)
2. AGE: 10, 20, 30, 40, 50, 60, 70, 80 (Age of the patient)
3. SEX: male, female
4. STEROID: no, yes (If the patient has been using steroids or not)
5. ANTIVIRALS: no, yes (Antiviral drugs are a class of medication used specifically for treating viral infections rather than bacterial ones. This says if the patient uses any antivirals or not)
6. FATIGUE: no, yes (If the patient shows any symptoms of fatigue)
7. MALAISE: no, yes (a general feeling of discomfort, illness, or uneasiness whose exact cause is difficult to identify. This tells if the patient has any symptoms of malaise or not)
8. ANOREXIA: no, yes (Anorexia is nothing but losing appetite for food or not feeling hungry. This tells if the patient lost his appetite or not)
9. LIVER BIG: no, yes (This attribute tells us if the patient's liver is expanded or not).
10. LIVER FIRM: no, yes (This attribute gives the information about the liver being too firm or not)
11. SPLEEN PALPABLE: no, yes (an abdominal organ involved in the production and removal of blood cells in most vertebrates and forming part of the immune system. Palpable is the feeling of being too intense as to seem almost tangible. This attribute tells us if the spleen is palpable or not)
12. SPIDERS: no, yes (A group of blood vessels accumulating as the shape of spider and settling close to skin)
13. ASCITES: no, yes (Ascites is a gastroenterological term for an accumulation of fluid in the peritoneal cavity that exceeds 25 ml)
14. VARICES: no, yes (Esophageal varices (sometimes spelled oesophageal varices) are extremely dilated sub-mucosal veins in the lower third of the esophagus. They

are most often a consequence of portal hypertension, commonly due to cirrhosis; patients with esophageal varices have a strong tendency to develop bleeding.)

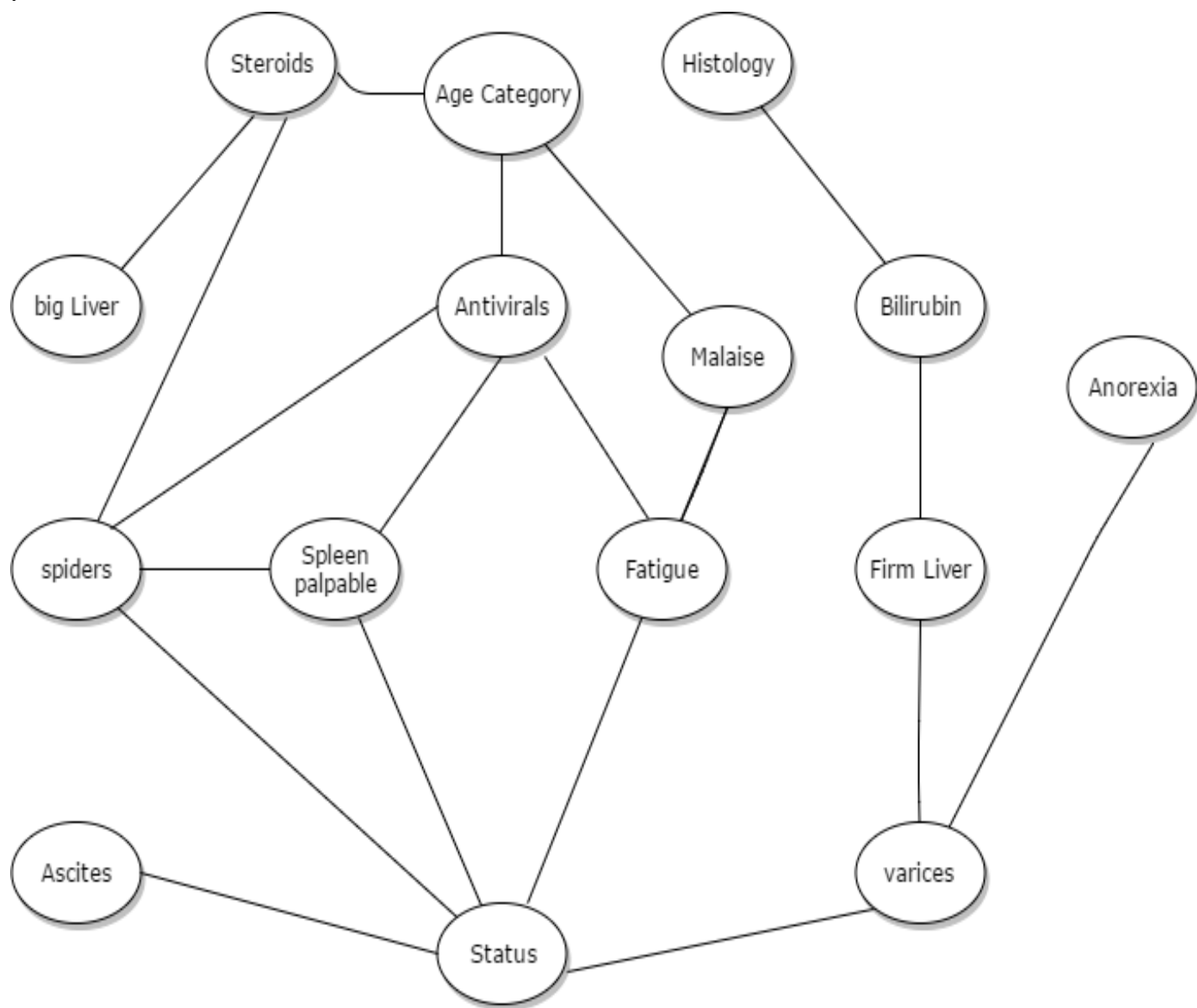
15. BILIRUBIN: 0.39, 0.80, 1.20, 2.00, 3.00, 4.00 (Bilirubin testing checks for levels of bilirubin in your blood. Bilirubin (bil-ih-ROO-bin) is an orange-yellow substance made during the normal breakdown of red blood cells. Bilirubin passes through the liver and is eventually excreted out of the body.)

16. HISTOLOGY: no, yes (The study of the form of structures seen under the microscope (light, electron, infrared). Also called microscopic anatomy, as opposed to gross anatomy which involves structures that can be observed with the naked eye.)

Markov Network:

The dataset contains 16 different variables. So the Markov Network for the variables is as below. This markov network is drawn after obtaining thorough knowledge on the domain of the dataset and even observing the data dependencies among every pair of variables. From the results obtained the below markov network is constructed.

These are the dependencies obtained from the data set:



Construction of model:

Step 1: Markov model can be imported from PGMPY and model can be designed by giving all possible markov factors to the model.

```

4 import pgmpy
5 from pgmpy.factors.discrete import DiscreteFactor
6 df = panda.read_csv("Desktop/AML PROJECT 2/amlmod.csv")
7
8 model = MarkovModel([("Status", "Sex"), ("Status", "Ascites"), ("Status", "Varices"), ("Status", "Spleen.palpable"), ("S
9
10 model.add_factors(DiscreteFactor(["Status", "Sex"], cardinality=[2,2],
11                                 values=[103,32,16,1]))
12
13
14 model.add_factors(DiscreteFactor(["Status", "Ascites"], cardinality=[2,2],
15                                 values=[112,17,7,15]))
16
17
18 model.add_factors(DiscreteFactor(["Status", "Varices"], cardinality=[2,2],
19                                 values=[112,20,7,12]))
20
21
22 model.add_factors(DiscreteFactor(["Status", "Spleen.palpable"], cardinality=[2,2],
23                                 values=[101,20,18,12]))
24
25
26 model.add_factors(DiscreteFactor(["Status", "Liver.big"], cardinality=[2,2],
27                                 values=[97,27,22,5]))
28
29
30 model.add_factors(DiscreteFactor(["Status", "Liver.Firm"], cardinality=[2,2],
31                                 values=[69,16,50,16]))
32
33 model.add_factors(DiscreteFactor(["Status", "Bilrub"], cardinality=[2,3],
34                                 values=[109,18,9,7,1,7]))

```

on file | length: 3,530 lines: 84 | Ln: 44 Col: 53 Sel: 0 | 0 | Unix (LF) | UTF-8

Inference Algorithms:

Inference Algorithms Used: The inference algorithms used in the project are:

Approximate Inference algorithm:

Belief Propagation: Belief propagation is also known as sum-product message passing, which is a technique(algorithm) for performing inference on graphical models, such as Bayesian networks and Markov random fields. It calculates the marginal distribution for each unobserved node, conditional on any observed nodes. Belief propagation is commonly used in artificial intelligence and information theory and has demonstrated empirical success in numerous applications including low-density parity-check codes, turbo codes, free energy approximation, and satisfiability.

This is the code for Beliefpropagation using pgmpy library

belpro = BeliefPropagation(MarkovModel)

Since our data set doesn't have any continuous variables, we had to resort to approximate inference from variable elimination and exact inference and we use belief propagation to obtain the maximum value of variables given their evidences.

Step 3: Now we got the model, so we have to apply inference algorithm by the following model/syntax.

```
from pgmpy.inference import VariableElimination
```

```
vemodel = VariableElimination(model)
```

```
from pgmpy.inference import BeliefPropagation
```

```
bpmodel = BeliefPropagation(model)
```

Queries:

These are the sample queries that can be answered:

```
bp_status = bpmodel.query(variables = ["Varices"],evidence={"Anorexia" : 1})  
print(bp_status["Varices"])
```

Varices	phi(Varices)
Varices_0	0.8485
Varices_1	0.1515

```
bp_status = bpmodel.query(variables = ["Antivirals","Steroid"],evidence={"agecat": 2})
```

```
print(bp_status["Antivirals"])
```

Antivirals	phi(Antivirals)
Antivirals_0	0.7638
Antivirals_1	0.2362

```
print(bp_status["Steroid"])
```

Steroid	phi(Steroid)
Steroid_0	0.8421
Steroid_1	0.1579

```
bp_status = bpmodel.query(variables = ["Histology"],evidence={"Status":1})
```

Histology	phi(Histology)
Histology_0	0.9136
Histology_1	0.0864


```
bp_status = bpmodel.query(variables = ["agecat"],evidence={"Fatigue" : 1})
```

agecat	phi (agecat)
agecat_0	0.3298
agecat_1	0.0106
agecat_2	0.6596

```
bp_status = bpmodel.query(variables = ["Spiders"],evidence={"Antivirals":1})
```

```
print(bp_status["Spiders"])
```

Spiders	phi (Spiders)
Spiders_0	0.8990
Spiders_1	0.1010

```
bp_status = bpmodel.query(variables = ["Fatigue","Malaise"],evidence={"Histology":1})
```

```
print(bp_status["Malaise"])
```

Malaise	phi (Malaise)
Malaise_0	0.9011
Malaise_1	0.0989

Fatigue	phi(Fatigue)
Fatigue_0	0.9615
Fatigue_1	0.0385

```
bp_status = bpmmodel.query(variables = ["Ascites"],evidence={"agecat":1})
print(bp_status["Ascites"])
```

Ascites	phi(Ascites)
Ascites_0	0.8682
Ascites_1	0.1318

Gibbs Sampling:

Gibbs sampling is a Markov chain Monte Carlo (MCMC) algorithm for obtaining a sequence of observations which are approximated from a specified multivariate probability distribution, when direct sampling is difficult. This sequence can be used to approximate the joint distribution (generating a histogram of the distribution), to approximate the marginal distribution of one of the variables, or some subset of the variables (for example, the unknown parameters or latent variables); or to compute an integral. We tried gibbs sampling and ancestral sampling to sample our data set. So, we used Gibbs Sampling model.

Entropy :

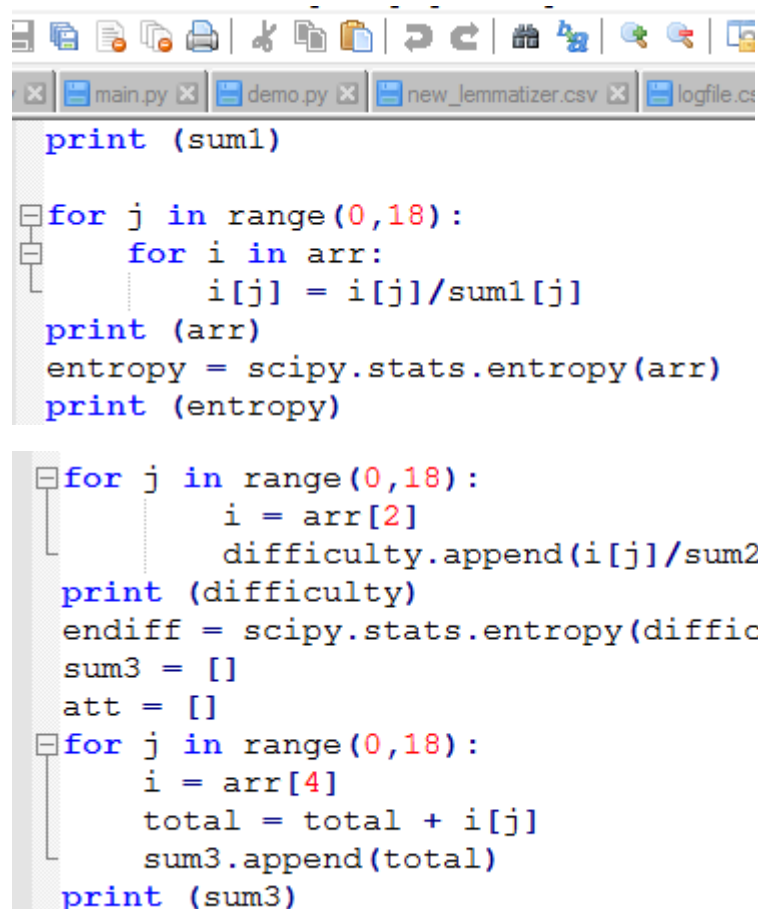
Entropy, in general is defined as the randomness or noise in a distribution.

In order to fully specify the Markov network and thus fully represent the joint probability distribution, it is necessary to specify for each node X the probability distribution for X conditional upon X 's parents. The distribution of X conditional upon its parents may have any form. It is common to work with discrete or Gaussian

distributions since that simplifies calculations. Sometimes only constraints on a distribution are known; one can then use the principle of maximum entropy to determine a single distribution, the one with the greatest entropy given the constraints. Relative entropy can be found out by

- It can be estimated using N samples as:

$$\hat{K}L(p||q) = -\frac{1}{N} \sum_i [\ln q(\mathbf{x}_k) - \ln p(\mathbf{x}_k)] .$$



```
print (sum1)

for j in range(0,18):
    for i in arr:
        i[j] = i[j]/sum1[j]
    print (arr)
    entropy = scipy.stats.entropy(arr)
    print (entropy)

for j in range(0,18):
    i = arr[2]
    difficulty.append(i[j]/sum2[j])
print (difficulty)
enddiff = scipy.stats.entropy(difficulty)
sum3 = []
att = []
for j in range(0,18):
    i = arr[4]
    total = total + i[j]
    sum3.append(total)
print (sum3)
```

Conclusion:

From this project we implemented various sophisticated algorithms like belief propagation and gibbs sampling and made some sense of raw data and the data is given as an input to the Markov model and this helps in transformation of the data to some useful results.

On the whole we tried to model the dataset using Markov models using the dependencies among the variables. We tried to apply approximate inference using belief propagation and sample the data and tried to answer the queries limited to our dataset that would comprehend the dataset in a better way.