Exploring Viral Parameters and Demographics using Markov Networks

Muhammad Fahad Baig {muhammad.fahad@curemd.com}

***Abstract*—Most viral diseases are modelled and interlinked with other diseases through complex patters. However, most existing work in this area uses pairwise relationships to make predictions. In this work, we use tree-structured conditional probability distributions to form a dependency network which then acts as basis to form state-of-the-art Markov Networks. We then employ some major learning methods and inference routines and make comparison between the model results and standard set of queries. We then confirm that sophisticated structures are denser than simple pairwise and lead to more accurate models that can help in revealing interesting patterns.**

***Index Terms*—Dependency Network, Markov Networks, Learning, Medical, Viral**

I. Introduction

THE human body is a highly complex organism, and a certain disease is rarely a consequence of a defect in one single part of the system. Most diseases are correlation of several molecular and genetic pathways; thus, they appear as cluster of other same types of disease which impact a common part of the system.

Several works done in this domain try to figure out networks that try to cluster diseases. These networks capture disease progression, i.e. can show that a person will develop a similar series of diseases within than cluster. However, viral infections are different to common diseases. Having a cluster of some diseases increases the likelihood of getting a viral infection but this isn't true in reverse. While viral diseases are independent of other diseases, their strength and effect on human body is directly proportional to them, due to the state of a person's immune system.

Commonly used measures to identify relations between diseases and infections include Relative Risk (RR) and pair-wise correlation. Recently there has been a shift to measuring multiple diseases and conditions within a person and trying to establish links between them. However, such work is still based on building long feature sets through pairwise comparisons, which makes them highly unsuitable to discover higher-order complex patterns.

In this article, we employ Dependency Trees to create a probabilistic model of the given data, then use DT and L1 regression to extract features, further fine tune the features using PRUNE and NONZERO, and finally construct a Markov Network and train the weights using learning algorithms. Our model representation of MN is as a log-linear model, which allows structured learning to be as cost optimization or feature learning problem.

Decision Trees (DTs) help in learning more complex patters in the given dataset. L1 regression computes smaller pairwise interactions, combining them both offers a superior representation of the given dataset as compared to using each of them alone. These predictive models can then be easily converted to features. Further fine tuning, such as dropping complex sets or adding more simple interactions is done through PRUNE and DEFAULT. Finally, we use pseudo-log-likelihood PLL to learn weights of the final network, sample them using inferring techniques and verify its performance against validation and query data set.

This article is submitted as solution to Part 2 of CureMD Research Grit Aptitude test and should not be used as proper research article. It builds on models, theories and algorithms that are already well established and tested.

II. Preliminaries

A. Markov Networks

A Markov Network represents a joint distribution between set of variables X. It consists of an undirected graph G and a set of potential functions ϕ. The graph has node for each variable and model has potential function for each clique in the graph. A normalization constant Z is also added. The joint distribution is represented as:

$$P(X = x) = \frac{1}{Z}\prod_{k} \phi_k(x\{_k\})$$

These models are conveniently represented as log-linear models where the product is replaced by exponential sum:

$$P(X = x) = \frac{1}{Z}\exp\left(\sum_{j} w_j f_j(x)\right)$$

A feature can be any real-valued function. For discrete data, which is true for our dataset in experiment, it is a series of conjunction of tests on the variables. A feature matches an example if it's true for that example.

B. Dependency Networks

The core motive behind learning Dependency Networks is to

learn a strict probabilistic decision tree for each variable and then combine it to form a single model. For every variable in the tree, we compute probability for a variable X_i given all other variables. These probabilities are represented in Decision Trees (DTs) and then converted to features which can represent the complex probability distribution as the tree.

Libra Library allows us to create Dependency Networks thus the algorithm for the routine itself isn't provided in this article.

C. Feature Pruning

DTs can include plethora of complex features and may avoid certain basic features. Pruning allows us to add basic stripped feature sets without any intermediary notes and remove longer feature sets to prevent overfitting over the training data. This done when converting the DNs to MNs using Libra to build traceable networks.

D. Weight Learning

In weight learning task, given a set of features and data, we try to learn the weight associated with each feature. Weights help to maximize an objective function which can be cost, or distance based. In this approach, we learn weights that maximize log-likelihood of the training data. As weights are convex functions in Markov Network, they can be learned iteratively using an optimization technique which computes gradient.

Several existing techniques optimize PLL, which is efficient to compute and can be learned via convex optimization. It is defined as:

$$\log P_w(X = x) = \sum_{j=1}^{V} \sum_{i=1}^{|D|} log P_w(X_{i,j} = x_{i,j} | MB_x(X_{i,j}))$$

V is the number of variables; D is the number of examples and MB is the Markov Blanket of $X_{i,j}$.

E. Structure Learning

The structure learning task tries to build a local model by discovering Markov Blanket given set of features and their weights. The model is built such as that a variable can be predicted given other variables. Any standard weight learning algorithm can be used to learn the final model structure when features are already extracted.

Some standard algorithms used are L1 (logistic regression), that creates pairwise interactions which guarantee that Markov Blanket for each variable can be easily formed.

We'll be using combination of DT, DN and L1 to create model and then try reducing non-unique features by either pruning or setting their weights to zero.

III. Experiment

A. Data

The dataset used for analysis is a list of COVID-19 patients, their testing results, symptoms observed, life status, type of care received, smoking patterns and age class. The total dataset contained records for over 500k patients which have been trimmed down to remove missing, incomplete and duplicate data to form a modest set of approximately 8000 patients. This high reduction occurred because we segregated most of the data into classes to ensure that certain variables such as age do not impact the accuracy of prediction due to its high range. For each patient, 19 distinct variables were recorded, as binary and tuple values

The accompanying code contains python code to clean up dataset, in case a new version is downloaded from Mexican Government website.

B. Methodology

To fairly evaluate performance of the learned models, a ten-fold cross-validation was performed. Thus, in every iteration, we trained over six folds of data, evaluated over three folds and tested over one-fold. These parameters can be easily adjusted in the accompanying python notebooks.

Publicly available implementations of all methods were used. Objective function was to optimize PLL. Same weight learning procedure was applied to all models. To select the best Dependency Network, we used varying split penalty constants and selected the one that had the best score out of all. Computation times are also reported for each approach where applicable. Feature lengths and parent distribution for each approach are also reported where applicable.

Our learning approach is listed below:

1. Build Dependency Networks using several split constants, 0.0001, 0.001, 0.01, 0.1 and 1.0. Select the model that scores the best. Report both the scores and the computation times for each network. Also select the network with highest features as comparison.
2. Construct a BN using Chow Liu's algorithm with several prior values. Select the one with best PLL and another one with highest feature count. Report both scores and computation times. This is for comparison.
3. Convert DNs to MNs and apply weight learning to them.
4. Build traceable ACs to compute exact inference
5. Report marginal probabilities per variable for each model and report PLLs for the queries.

We'll also try to answer some basic queries:

1. Predict marginal probability of death given person positive result and suffers from all mentioned disease traits (obesity, smoking, heart, etc.).
2. Predict marginal probability of a young male (age class 0) to have a positive test result.
3. Predict marginal probability of having positive result for class 1 aged female who is obese, smokes, asthma and hypertensive.

IV. RESULTS

Table 1 shows the PLL scores and computation times (CT) for all DN split values. The constant with best PLL score was selected for further testing.

Table 1 PLL and CT scores for DN Split values

SPLIT	PLL	CT
0.0001	-10.755946	0.101397
0.001	-10.663731	0.102164
0.01	**-10.570995**	**0.118552**
0.1	-10.630340	0.169404
1.0	-11.613556	0.324388

As evident from the table above, 0.01 has the best PLL score of all and thus was selected. The number of features and average parents for each split is also shown below:

Table 2 Features and Parents info for each Split

SPLIT	FEATURES	AVG. PARENT
0.0001	385	5.789474
0.001	591	8.052632
0.01	991	10.263158
0.1	2726	15.210526
1.0	**13604**	**17.05263**

The network with highest features is primarily the one with largest split, which is bolded in Table 2. We use it as comparison and analysis of further results.

For our second methodology, we constructed BNs using Chow Liu's algorithm with different smoothening constants i.e. prior values. This is to serve as a comparison benchmark to evaluate performance of MNs and DNs against other graph structures. BNs are directed graphs as compared to MNs undirected graphs. PLL and CT are reported in Table 3, whereas features and parents average in Table 4.

Table 3 PLL and CT scores for BN Prior Values

PRIOR	PLL	CT
1	**-10.917231**	**0.032035**
10	-10.919780	0.032669
100	-10.947165	0.032419
1000	-11.284163	0.034605

Table 4 Features and Parents info for each Prior

PRIOR	FEATURES	AVG. PARENT
1	122	0.947368
10	122	0.947368
100	122	0.947368
1000	122	0.947368

BNs have same features, thus showing their first demerit as being unable to be tuned to learn more complex features. However, the PRIOR value only smoothens the resulting network. The algorithm itself works by greedily building up the whole tree with the parent presenting most of the information from the given dataset.

The third stage of methodology involves converting our DNs to MNs. DNs represent highly complex feature information which will now be converted to Markov Networks so that weight training and inferring algorithms can be applied to them. The first is to learn traditional MNs. We'll be reporting results from two MN structures, one with many duplicate results and one which is pruned to contain only exact and unique features; the later might lead to slower performance which will be evaluated later. Results are show in Table 5 below:

Table 5 DN to MN Conversion Results

MODEL	PLL	PARAMS
0.01-all	-10.469238	27379
0.01-unique	-10.469238	9768
1-all	*stack overflow*	555902
1-unique	-10.642613	198544

Using pruned unique features resulted in a reduction of 64% approximately for both models. Unfortunately, the model for 1-all increased dramatically in size that Libra threw a stack overflow exception. Since Libra can't be compiled on newer build systems and we're using pre-compiled binaries, this error cannot be remedied. Regardless, we can see that converting models to Markov Networks yields a better PLL score as compared to the DN networks, with obviously a great increase in network parameters as the whole DN is unfolded into features. Going forward, we'll be only using the 0.01-all and 1-unique model and drop the other two.

Now weight learning algorithms is applied to both MNs. Optimization iterations are set to 500 to both and L1 normalization is set to 0.01 and L2 (SD) to 0.1. This algorithm will serve to further reduce the PLL scores of the model as reported above for the training dataset. The results are shown in Table 6 below:

Table 6 PLL and CT scores for Weight Training Models

MODEL	PLL (eval)	PLL (train)	CT
0.01-all	-9.981439	-8.92652	58.979
1-unique	-10.813462	-10.051385	74.852

While the results can be further improved by adjusting the L1 and L2 through multiple combinations and evaluating results, it is omitted due to time constraints.

Now that we have all our three models, we'll convert them to AC (circuits) so that exact inference and queries can be calculated on them. Apparently, Libra isn't well adapted to large MN models and thus didn't outputted any result on our test system. We thus resorted to using Gibbs Sampling for MNW models and traditional AC for BN models. The marginal probabilities for each model and variables are shown in Table 7 i.e. how much they impact each other variable. This is useful to classify which variables affect unknown variables in a query given them as evidence. The conditional log likelihood for all the queries mentioned in Experiment section is shown in Table 8 and marginal probabilities in Table 9.

Table 7 Marginal Probabilities for each Variable (only for binary 0 and class 0)

Model	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
MN-0.01A	0.535	0.687	0.701	0.471	0.001	0.074	0.566	0.780	0.833	0.776	0.507	0.764	0.723	0.587	0.767	0.724	0.604	0.477	0.466
MN-1-UNQ	0.527	0.645	0.739	0.464	0.034	0.071	0.548	0.781	0.798	0.789	0.502	0.802	0.733	0.567	0.753	0.722	0.604	0.473	0.463
BN-S	0.532	0.666	0.698	0.457	0.000	0.070	0.555	0.771	0.819	0.771	0.469	0.762	0.711	0.582	0.751	0.720	0.605	0.476	0.458

Table 8 PLLs for each Model three scenarios

Model	1	2	3
MN-0.01A	-16.063395	-10.764666	-19.76647
MN-1-UNQ	-16.484121	-10.581478	-15.769430
BN-S	-16.286811	-10.508315	-22.283735

Table 9 Marginal Probability for the target variable

Model	1	2	3
MN-0.01A	0.2901046	0.527149	0.515835
MN-1-UNQ	0.2804308	0.527700	0.536772
BN-S	0.3010723	0.523879	0.523879

V. Interpretation

From a medical perspective, the question is how to interpret complex features. To gain some insight we'll use our queries and some disease parameters and demographics to explain them.

A. Diseases in Features

We can extract exciting features from the data. For example, there's a 30% probabilistic chance of a Mexican to die of the virus given the evidence/state of all other variables. Also, 64% have diabetes, 50% have hypertension, 33% have other disease, 38% have cardiovascular problems, 38% have smoking issues. These percentages are not demographics but as mentioned, relative probabilities of the variables computed against marginal probabilities of all other variables in data. Similarly, hypertension and asthma are related to factors such as obesity and smoking. While some of these features can be expressed by increasing/decreasing analysis trend, the medical factors require a more in-depth investigation which is beyond the scope of this work.

B. Graphical Comparison

Opening the learned models in text editors and scanning for a subset of variables can help us in extracting further trends in the data. For example, Ln 108472 of final-1-unique (MN-1-UNIQUE) model correctly shows a high chance of death (v2) associated with old people (v5_3), diabetes (v6), asthma (v8) and tabaquism (v15), as real world analysis also shows that COVID virus badly impacts patients lungs and thus people with already poor lungs and immune system (diabetes and aging) are more prone to the virus and are more probable to die (shown by real world statistics). Lastly, we can see that how our *unique* model breaks down 19 variables into all possible combinations where the *all* model (MN-0.01-ALL) lists only the detected features from the given dataset. Our test results show that it is extremely fast (40 seconds average compared to 400 seconds average) while producing the same amount of results. This shows how reducing unneeded features can heavily reduce computation times while maintaining the same features.

While comparing all the feature interactions is impossible, the trained models are provided with this article so that the end reader can make required classification and feature extraction as required.

VI. Conclusion

In this article, we analyzed COVID patient's data from Mexican government with multiple diseases and conditions to identify co-occurrence of several diseases and establish links between them. The primary motive was to investigate possibility of using Markov networks to explore relationships involving a person's basic demographics and diseases. Using a real-world dataset, we found out that going beyond just the disease characteristics and considering other potential diseases a patient has suffered in the past or not can result in more accurate learned models. We also inspected some of the manually learned features and found out that they make sense from a medical perspective. In conclusion, we believe using MNs to visualize, model and interpret geographical and medical disease data is a promising approach to identify disease patterns and severity to better utilize resources and care in these tough circumstances.