AN ANALYSIS OF AI ALGORITHMS ON BAYESIAN NETWORKS

Ayokunle Jephthah Abisola
Robotics and Autonomous Systems
School of Computer Science
University of Lincoln
Lincoln, UK
26736158@students.lincoln.ac.uk

***Abstract*—This report aims to analyze certain AI algorithms using python implementations of them on bayesian networks given the available health data. It spans across simple probabilistic queries on discrete and continuous data, Approximate Inference methods and structure learning with Hill Climbing method.**

***Index Terms*—AI, Machines, Algorithms, probabilities, data**

I. Introduction

The intent to build machines which mimics human intelligence is well grounded in our society today. This subset of machine learning has found its place in various fields of application, such as healthcare, financial analysis, and robotics among many others. Complex systems are a function of smaller irreducible systems. An AI system fits such description as it operates on an algorithm or a combination of algorithms; A step by step process which dictates how a machine should complete a simple calculation or a much bigger task. In simple terms, it guides the machine in its learning process to operate on its own.

Due to its wide range of applications, AI systems operate on several different algorithms depending on what is required or considered as priority. A good example would be classification systems, where the type of classification could be binary or multi-class. In multi-class classification, values could be discrete or continuous. Some types of AI algorithms include Naïve Bayes Classifier, Logistic Regression, Linear Regression, random forests. This report focuses on Bayesian Networks, its simplest form using Naive Bayes, how to learn them from a data, and infer from them.

A Bayesian network is a probabilistic graphical model that measures the conditional dependence structure of a set of random variables based on the Bayes theorem (1) [2].

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

Naïve Bayes (3) can be defined as a simple learning algorithm that utilizes Bayes' rule (2) together with a strong assumption that the attributes are conditionally independent given the class [1].

$$P(Y = y_k|X = x_i) = \frac{P(X = x_i|Y = y_k)P(Y = y_k)}{\sum_j P(X = x_i|Y = y_j)P(Y = y_j)} \tag{2}$$

$$Y = argmax_{y_k} P(Y = y_k) \prod P(X_i|Y = y_k) \tag{3}$$

Realistically, the strong assumption that Naïve Bayes makes is not true, considering that in most cases, features in a given data are not completely independent of each other. However, it requires less computational power and is effective to that regard. Its accuracy can also be reliable to a certain extent, with little or no noise in data.

A Naïve Bayes classifier only generates conditional probability given a single parent. It is the simplest form of a Bayesian network.

II. Algorithms

A. Rejection Sampling

Rejection Sampling is an approximate inference algorithm, a family of the Monte Carlo technique, used on bayesian networks. It can be used to generate samples from a target probability density function (pdf) by drawing from a proposed density. The sample is either accepted or rejected by an adequate test of the ratio of the two pdf's and it can be proved that accepted samples are distributed according to the target density [3]. In simpler terms, samples are generated based on a conditional probability table, in a certain order given a query. All generated samples would have a corresponding value for a feature in the data which aligns with the query, any sample which does not, is rejected.

B. Likelihood Weighting

An Inference algorithm like Rejection sampling. However, it accounts for the query given by assigning the feature to the value instead of a random pick which could lead to rejection of the sample. In this case, a weight value is stated, which changes as the sampling process is done. It multiplies the weight with the probability of the evidence variable (specific feature stated in the query) during sampling.

C. Scoring Functions

Bayesian Networks provides the structural information, and the relationship between variables represented by the probability distributions [4]. An approach to learning these networks is by searching and scoring them based on a scoring function algorithm.

$$Score(B|D) = \sum_{i=1}^{n} score(X_i|parents(X_i), D) \quad (4)$$

1) Bayesian Information Criterion: A common scoring function is the Bayesian Information Criterion (BIC) (6). The BIC was introduced by mathematician Gideon E. Schwarz (1978) [5]. It takes into account the log likelihood score and the penalty value. The log likelihood score (5) takes the logarithmic value of the conditional probability of every variable given the corresponding parent.

$$LL(D|B) = \sum_{i}^{n} \sum_{j}^{N} logP(D_{ij}|PA_{ij}) \quad (5)$$

$$BIC(D|B) = LL(D|B) - \sum_{i=1}^{N} penalty(X_i, B, D) \quad (6)$$

$$localpenalty = \frac{\log n * P_i}{2} \quad (7)$$

D. Gaussian Bayesian Networks

Working with discrete data is smooth, and concise, but there might be loss of data. To solve this problem, a normal distribution is done over continuous data. A Gaussian BN uses a directed acyclic graph where all edges follow a gaussian distribution (8).

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} \quad (8)$$

III. ANALYSIS

All implementations are written in python programming language.

A. Task 1 - Naive Bayes Implementation

1) Process: This implementation uses a simple Bayesian network based on a Naïve Bayes classifier (3). The data parsing process involves stripping the train data file of all features and all instances. It then calculates the probabilities of each feature, which includes their single probabilities and the conditional probability given its parent, the actual target. In the case of zero probabilities, the implementation accounts for this using a function called the maximum likelihood estimation (9). It also tests the Bayesian network learnt on a test data file to check for the performance.

$$P(x|y) = \frac{count(x|y) + 1}{count(y) + |X|} \quad (9)$$

2) Result:

- Queries
  - $P(target = 1|gender = 1, cp = 3)$ - [0.027757178298175235]
  - $P(target|gender = 0, cp = 3)$ - [0.003985239852398524, 0.021443313706744814]
  - $P(stroke = 1|gender = Female, age = 2, smoking_status = smokes)$ - [0.00038815390248185256]
  - $P(stroke|gender = Male, age = 2, smoking_status = formerlysmoked)$ - [0.000502349905675789, 0.01572783972079521]

Metrics	Stroke	Heart
Classification Accuracy	0.9301355578727841	0.8443396226415094
Balanced Accuracy	0.6048694232861807	0.843344953964423
Area Under Curve	0.8580794341675735	0.9359524448020023
Brier Score	0.05084526489772181	0.10944705395247319
KL Divergence	61.61377288847838	31.962070369401538
Training time	0.1010732650756836	0.11070799827575684
Inference time	0.0725393295288086	0.015628814697265625
log likelihood score	-36022.70887174522	-10028.714775619497
BIC	-36176.8343058842	-10202.933784467741

TABLE I
SIMPLE BAYESIAN NETWORK PERFORMANCE

3) Discussion: As seen in the performance on both data, the simple Bayesian network performs well, depending on the data and the noise in it. The stroke data has a very high classification accuracy, and a very low brier score which is indicative of almost perfect accuracy.However, taking a look at the data, it is a biased data, as such, the classification accuracy is not a good metric to look at, but the balanced accuracy, which takes into account the confusion matrix of true and false positives and negatives gives us a value of 0.60. On the heart data, a more balanced data set, the simple Bayesian network, performed much better, which lower BIC score, which shows the structure, although simple, is a good one. The overall performance on both data is well above average, considering the minimal computational power required for the network, as seen in the training and inference times.

B. Task 2 - Approximate Inference Comparison

1) Process: In this task, two approximate inference algorithms would be compared, namely ; Rejection sampling and Likelihood weighting. In both implementations, a generated conditional probability table, an ordered structure, a query and sample size are arguments given to the algorithm. It samples step by step according to the order in the structure file and keeps into account the value of the evidence variable in the query. Using the random library in python, a random pick is made in each step of the sample process. In rejection sampling, if the variable value chosen for an evidence variable does not match the one stated in the query, the entire sample is rejected while in likelihood weighting, it assigns them to be that value.

2) Result: Using a sample size of 10000, the result of the queries are below

- Query 1 = "stroke—age=2,gender=Female,smoking$_s tatus = smokes$"
  - Rejection sampling - '0': 0.49700598802395207, '1': 0.5029940119760479
  - Rejection Inference time - 7.2023701667785645
  - Likelihood weighting - '0': 0.75201977932091, '1': 0.2479802206790872

  - Likelihood Inference time - 21.596174240112305
- Query 2 = ”target—sex=0,cp=3”
  - Rejection sampling - ’0’: 0.5137540453074434, ’1’: 0.48624595469255666
  - Rejection Inference time - 8.960100889205933
  - Likelihood weighting - ’0’: 0.25403119656710804, ’1’: 0.7459688034328869
  - Likelihood Inference time - 13.459975004196167

Query 1 rejected samples: 9833 good samples: 167
Query 2 rejected samples: 8764 good samples: 1236

3) Discussion: From the Inference time in each case, that is the time to generate the samples, it is clear that likelihood weighting takes longer. However, due to having more samples, the predicted probability is going to be more accurate than Rejection sampling. Both algorithms are an approximate method, so the more samples taken, the closer it gets to the true probability value of the query. The edge these methods have over exact inference method, would be how fast they work as compared. But when trying to be more accurate, and taking millions of samples, this edge might no longer be valid, as it would be better to use an exact method instead.

C. Task 3 - Structure Learning

1) Process: This focuses on Hill Climbing algorithm, which uses the log likelihood (LL) and Bayesian Information Criterion (BIC) values to find the best structure. The implementation takes in a structure file, a conditional probability file and a training data. It strips the train file, taking every row, then calculating the LL for each variable value in the row, to sum up to the LL of the structure. The penalty calculation follows a similar pattern, only taking into account the features as a whole instead of each row. The grid consists of 7 different structures, 6 of which was generated by reversing the parent to child edge in the first structure. The python test file searches through this grid, calculating the LL and BIC for each structure.

2) Result: The table below (II) shows the scores for several structures.

Structure	Log Likelihood	BIC
STRUCTURE 0	-31948.370705813555	-74478.65942362431
STRUCTURE 1	-31955.697218656816	-74552.634772852
STRUCTURE 2	-31947.391019539613	-73744.54253712171
STRUCTURE 3	-31972.518268874377	-74419.49594120462
STRUCTURE 4	-31929.216894581466	-54464.85469706493
STRUCTURE 5	-31976.42174929824	-74548.36598984926
STRUCTURE 6	-32022.498946158372	-42561.34619944547

TABLE II
STRUCTURE SCORES

3) Discussion: In the grid of structures, the search has shown that structure 6 has the highest BIC score. Hence, it is the best structure. This is followed by structure 4, making it the second best structure. The hill climbing algorithm is better than a simple PC- stable algorithm. A PC-stable infers a skeleton based on conditional independence tests, and following a set of rules, it directs the graph. Eventually, it arrives at one structure, which is accepted as the best. On the other hand, Hill Climbing searches a grid of structures to find the best. A combination of both would be the best.

D. Task 4 - Gaussian Naive Bayes

1) Process: An extension of task 1. The implementation of this takes in train and test files, it checks while parsing the data if any of the features in the data contains continuous values, if it does, it implements the Gaussian method. This involves calculating the means and standard deviation of each feature and passing it into the Gaussian distribution formula. Also, it is based on a simple Bayesian network.

2) Result: The table below (III) shows the different metrics measured for the discrete and continuous heart Bayesian network

Metrics	Original Heart	Discrete Heart
Classification Accuracy	0.8207547169811321	0.8443396226415094
Balanced Accuracy	0.8168409761330115	0.843344953964423
Area Under Curve	0.912308930008045	0.9359524448020023
Brier Score	0.13254694224394684	0.10944705395247319
KL Divergence	36.04711315359399	31.962070369401538
Training time	0.01399850845336914	0.11070799827575684
Inference time	0.03899192810058594	0.015628814697265625
log likelihood score	-19832.79161994718	-10028.714775619497
BIC	-21169.587476301967	-10202.933784467741

TABLE III
GAUSSIAN BAYESIAN NETWORK PERFORMANCE

3) Discussion: It appears the discrete data performed better than the continuous data with higher BIC score. The various values to be accounted for in the continuous data might be one reason for this. However, the original data set has a faster training time. It is also to be noted that the performance of the original data can be said to be the true performance of the heart data.

IV. CONCLUSION

A Naïve Bayes classifier is a good algorithm to implement on a data that is not noisy, and when looking to use less computational power. Although, it would be better to have a more defined structure other than a simple Bayesian network, which would improve the overall performance. To do this, a learning algorithm such as hill climbing or a hybrid algorithm can be used to find a better structure. The probability distribution of such a structure can be inferred by the inference methods. While using approximate inference algorithms might be faster, if accuracy is of utmost importance, an exact inference method is more appropriate. For continuous data, they can be good representation of true values in a data, but discretizing them might give a better performance. Bayesian Networks are a nice way of understanding the relationship in a data, which can then allow for an assessment of a new instance, to give a predicted conclusion. It no doubts plays a main role in AI systems.

REFERENCES

[1] Webb, G.I., Keogh, E. and Miikkulainen, R., 2010. Naïve Bayes. Encyclopedia of machine learning, 15, pp.713-714.

[2] Liu, S., McGree, J., Ge, Z. and Xie, Y., 2015. Computational and statistical methods for analysing big data with applications. Academic Press

[3] Martino, L. and Míguez, J., 2011. A generalization of the adaptive rejection sampling algorithm. Statistics and Computing, 21(4), pp.633-647.

[4] De Campos, L.M. and Friedman, N., 2006. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. Journal of Machine Learning Research, 7(10).

[5] 5.Schwarz, G., 1978. Estimating the dimension of a model Ann Stat 6: 461–464.