An Investigation into Bayesian Networks and Gaussian Processes

Eden Attenborough
University of Lincoln
Lincoln, United Kingdom
28238148@students.lincoln.ac.uk

***Abstract*—In this assessment we looked at methods and algorithms pertaining to bayesian networks and gaussian processes, including structure learning of bayesean networks, parameter estimation and inference, comparing and contrasting different methods for each. We also looked at different algorithms and methods related to gaussian processes.**

***Index Terms*—Bayesean Networks, Gaussian Processes, Inference**

I. Introduction

The amount of code written by ourselves differs by the part of the task. Each section in turn describes the amount of code written by ourselves, or in the case where an external module or library was used, it is cited appropriately.

II. Part One - Bayesian Networks

For part one, none of the original code written by the lecturer remains. It was all written by ourselves, although in some cases where an external library was used this is referenced.

A. Structure Learning

Structure learning is done by using the file `structure_learning.py`. It takes a path to a training CSV file, a path to an output bayesian configuration file, a method for structure learning, a scoring algorithm, and optionally, a pruning algorithm. [1] was used for the structure learning, which in turn sometimes uses [2]. Structures were evaluated using the K2 score [6], the bayesian information criterion (BIC) [3], Bayesian Dirichlet equivalent uniform (BDeu) [4], and the Bayesian Dirichlet Sparse (BDs) [5]. The structure learning was done by [1] or [2], the options being PC Stable, Hill Climbing, or Naive-Bayes. There is also an exhaustive search implementation but this wasn't used as it was infeasible for both datasets. We can also optionally prune the network, using our implementation of the Chi Square and G Square significance test.

Figure 1 shows resulting metrics on different structure learning algorithms for the *Cardiovascular* dataset, and Figure 2 for the *Diabetes* dataset. The *exhaustivesearch* algorithm was not tested as it is not feasible to use it in datasets with this number of variables. For all datasets, and by all metrics, *Naivebayes* was the most useful structure, followed by *PCStable* and *Hillclimb*.

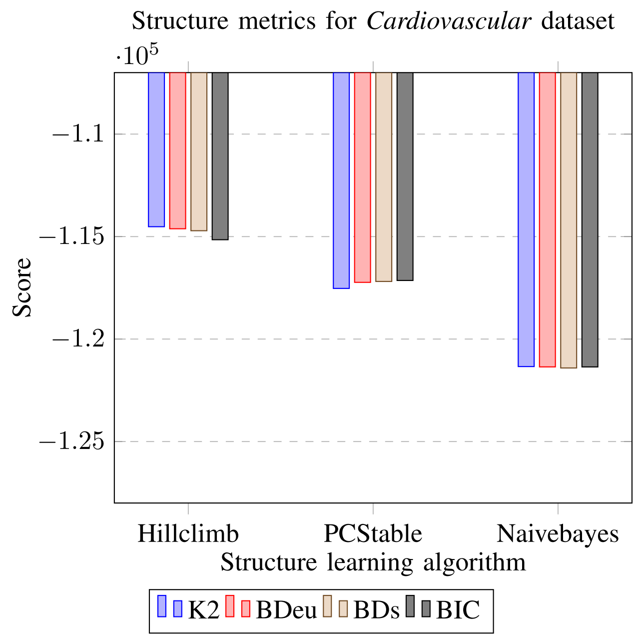

Fig. 1. Graph showing structure learning evaluation metrics for the *Cardiovascular* dataset, discretized using the provided dataset. The *exhaustive search* algorithm was not used as it was infeasible with this size of dataset. Evaluation metrics [6] [4] [5] [3].

B. Parameter Estimation

Our system has two algorithms for parameter estimation, which generate CPT (Conditional Probability) tables: *Maximum Likelihood Estimation* (MLE) and *Bayesian Parameter Estimation*. The implementation of MLE was written by ourselves, and the bayesian parameter estimation comes from [1]. The CPTs for MLE were tested against the implementation made by the lecturer, and were found to be identical.

Table I shows an example conditional probability table for the *Diabetes* dataset, generated using our MLE implementation on the variables $P(Smoke|Gender, Alco)$.

C. Inference

For inference we have to possible options, exact or approximate. We use [2]'s implementation of variable elimination and belief propagation.

Table II shows an example query conducted on the *Diabetes* dataset, where CPTs were generated using the *MLE* method.

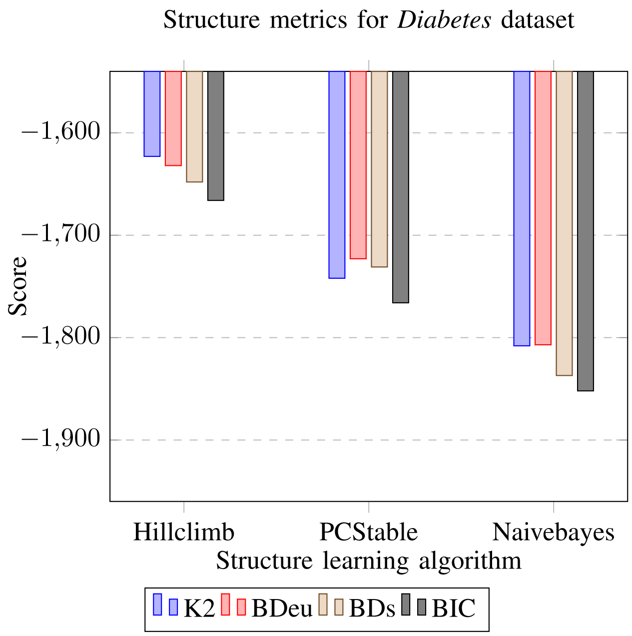

Fig. 2. Graph showing structure learning evaluation metrics for the *Diabetes* dataset, discretized using the provided dataset. The *exhaustive search* algorithm was not used as it was infeasible with this size of dataset. Evaluation metrics [6] [4] [5] [3].

TABLE I
CONDITIONAL PROBABILITY TABLE FOR VARIABLES $P(Smoke|Gender, Alco)$ GENERATED WITH MAXIMUM LIKELIHOOD ESTIMATION ON *Diabetes* DATASET

Child	**Parents**		
S(Smoke)	***G(Gender)***	***A(Alco)***	p
0	2	0	0.832885
0	2	1	0.354572
0	1	0	0.985105
0	1	1	0.861690
1	2	0	0.167115
1	2	1	0.645428
1	1	0	0.014895
1	1	1	0.138310

Table III shows an example query on the *Cardiovascular* dataset, where this time CPTs were generated using the alternate method.

We found that for all queries *Variable Elimination* was significantly faster, with this effect increasing exponentially with the complexity of the query (the number of variables considered).

Our implementation make is possible to query multiple variables at the same time, for example both *Outcome* and *Gluc*, which returns results in n dimensions.

D. Inference Evaluation

So far we have only considered individual queries which are not especially useful for evaluating the overall usefulness of models. To do this, we created the script `inference_evaluator.py` which evaluates models, and inference implementations on a test dataset by a number of metrics, specifically *accuracy*, *balanced accuracy*, *brier score loss*, *F1 score*, *Precision*, *Recall*, and *Area under the ROC*. Table IV shows accuracy metrics on the *Cardiovascular* dataset. We also tested on the other dataset but could not include the results due to page count issues. We only tested using variable elimination since the alternative was far too slow with the number of queries needed. The exact evaluation metric calculations were done by [7] when provided with a results vector.

TABLE II
RESULTS OF INFERENCE QUERY $P(Outcome|Insulin = 2, Glucose = 1, Age = 4)$ ON *Diabetes* DATASET, WITH *Naive-Bayes* STRUCTURE AND *MLE* CPTS

Method	**Options**		**Time (Seconds)**
	$P(0)$	$P(1)$	
Belief Propagation	0.8699	0.1301	6.41
Variable Elimination	0.8742	0.1258	0.05

TABLE III
RESULTS OF INFERENCE QUERY $P(Target|Age = 2, Weight = 3, Ap\_Hi = 3, Gluc = 3)$ ON *Cardiovascular* DATASET, WITH *Hillclimb* STRUCTURE AND *Bayesian Parameter* CPTS

Method	**Options**		**Time (Seconds)**
	$P(0)$	$P(1)$	
Belief Propagation	0.758	0.242	23.08
Variable Elimination	0.7581	0.2419	0.12

TABLE IV
INFERENCE ACCURACY METRICS WITH VARYING STRUCTURES AND CPT METHODS ON THE *Cardiovascular* DATASET

Metric	**Naive-Bayes**		**Hillclimb**		**PC-Stable**	
	MLE[a]	***BP***[b]	***MLE***	***BP***	***MLE***	***BP***
Accuracy	0.42	0.43	0.42	0.36	0.42	0.43
Balanced Accuracy	0.5	0.5	0.47	0.4	0.45	0.41
Brier Score Loss	0.52	0.57	0.58	0.64	0.55	0.6
F1 Score	0.6	0.6	0.55	0.49	0.47	0.6
Precision	0.42	0.43	0.41	0.37	0.4	0.35
Recall	1	1.0	0.82	0.73	0.7	0.7
Area under ROC	0.5	0.5	0.47	0.4	0.45	0.47
Time (Seconds)	252	252	250	251	252	251

[a] Our implementation of Maximum Likelihood Estimation.
[b] Bayesian Parameter Estimation from [1].

III. PART TWO - GAUSSIAN PROCESSES

A. Gaussian Processes Parameter Tuning

In order to investigate the effects of the parameters on inference accuracy, we made minor modifications to `GaussianProcess.py` such that parameters can be optionally tuned by command line arguments, and created a script `run_model_range.py` that runs evaluations using a range of different parameters on a logarithmic scale.

1) Noise: Figure 3 shows the effect of the noise parameter on the accuracy, where multiple possible metrics of accuracy are provided. It is important to consider that the values for the other parameters are not static, they are optimised each time.

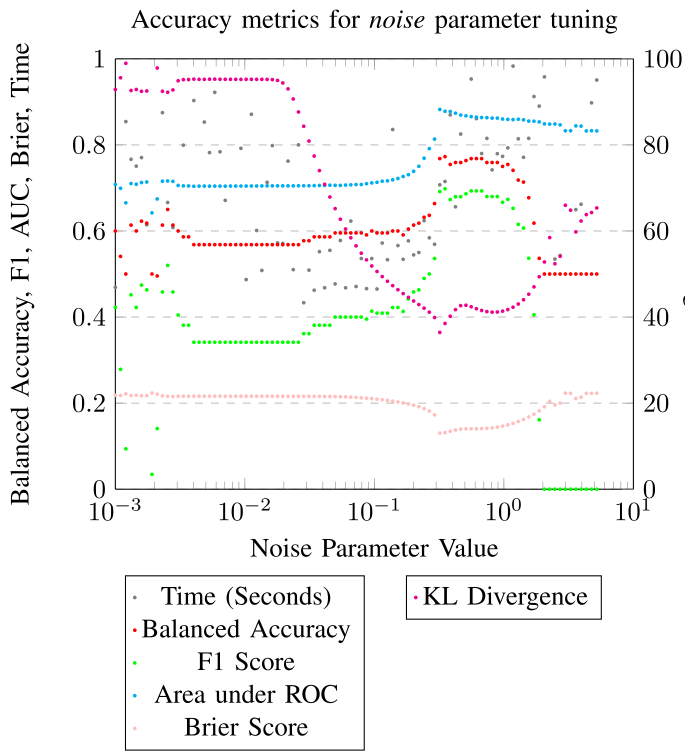

Fig. 3. Graph showing accuracy by a number of metrics, as well as time taken, for a range of values with the *noise* parameter, using the *diabetes* dataset. For each point, the values for the *l* and *sigma_f* parameters was calculated using the Limited-memory BGGS-B algorithm.

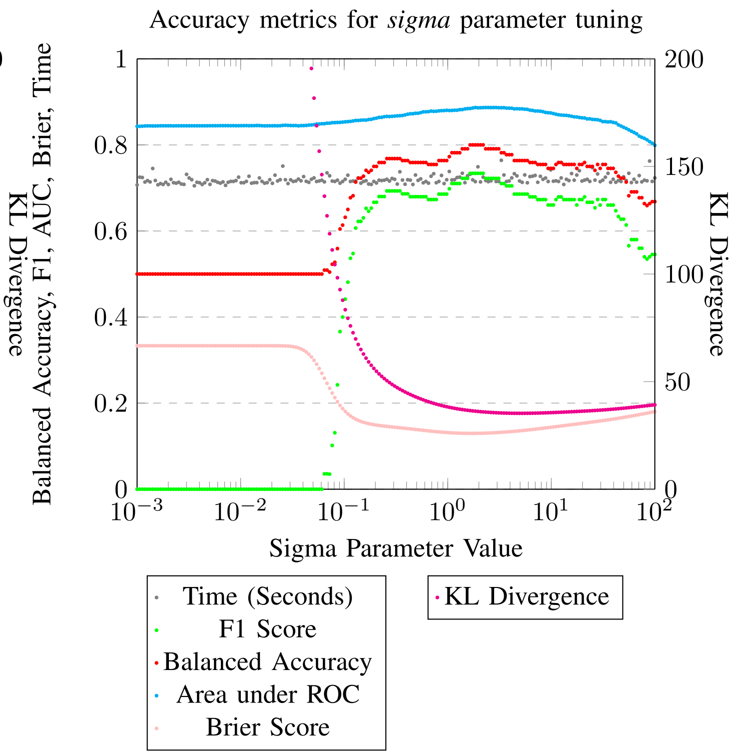

Fig. 4. Graph showing accuracy by a number of metrics, as well as time taken, for a range of values with the *sigma* parameter, using the *diabetes* dataset. Unlike Figure 3, the other parameters were set to constant values: 0.4 for *noise* and 6.955 for *l*.

2) Sigma Kernel Parameter: By default, the system uses an Isotropic Squared Exponential (ISE) Kernel. We investigated the effect of changing its parameters. Figure 4 shows the effect of varying the *sigma* parameter. Unlike Figure 3, the values of the other parameters were kept constant in this test.

3) L Kernel Parameter: We also tested the effect of changing the *l* parameter, but we couldn't fit the plot on this document. The raw CSV is enclosed within the code submission. The shape was similar to Figure 4.

B. Kernel Analysis

We changed the code to use an arbitrary kernel from [7]. Using the existing infrastructure for hyperparameter optimization with a limited memory Broyden–Fletcher–Goldfarb–Shanno algorithm, we tested a number of different kernels. The results for this are shown in Table V.

In conclusion, there was not a huge difference between the different kernels. This is perhaps because the Matérn and Rational quadratic kernels are derived from the Radial Basis Function kernel. We wanted to test on more kernels but we were disappointed by the number of kernels implemented in [7].

TABLE V
TABLE SHOWING ACCURACY METRICS WITH DIFFERENT KERNELS

Metric	**Kernel**		
	ISE[a]	***Matérn***[b]	***Rational Quadratic***[c]
Balanced Accuracy	0.75	0.77	0.76
F1 Score	0.67	0.7	0.69
Area under ROC	0.88	0.88	0.88
Brier Score	0.14	0.14	0.13
KL Divergence	40.8	35.8	39
Time (Seconds)	0.63	26	1

[a]Default Isotropic Squared Exponential (Radial Basis Function)
[b]Matérn kernel from [7]
[c]Rational Quadratic kernel from [7]

References

[1] E. Taskesen, 'Learning Bayesian Networks with the bnlearn Python Package.'. Zenodo, Nov. 10, 2023. doi: 10.5281/zenodo.10108817.

[2] Ankan, A. and Textor, J, 'pgmpy: A Python Toolkit for Bayesian Networks.'. Proceedings of the 14th python in science conference (scipy 2015) (Vol. 10), 2015

[3] Chickering, D.M., 'A Transformational Characterization of Equivalent Bayesian Network Structures.'. Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence, 1995, pp. 87–98.

[4] Heckerman, D. and Geiger, D. and Chickering D.M., 'Learning Bayesian Networks: The Combination of Knowledge and Statistical Data.'. Machine Learning 20(3), 1995, pp. 197–243.

[5] Scutari, M. 'An Empirical-Bayes Score for Discrete Bayesian Networks.'. Journal of Machine Learning Research, 2016, pp. 438–48.

[6] Korb, K. and Nicholson, A.E., 'Bayesian Artificial Intelligence.'. Chapman & Hall/CRC, 2nd edition, 2010

[7] Varoquaux, G. and Buitinck, L. and Louppe, G. and Grisel, O. and Pedregosa, F. and Mueller, A., 'Scikit-Learn: Machine Learning Without Learning the Machinery', Association for Computing Machinery, 19:1, January 2015, pp. 29–33