Learning Bayesian Networks from Inaccurate Data

A Study on the Effect of Inaccurate Data on Parameter Estimation and Structure Learning of Bayesian Networks

Valerie Sessions **Marco Valtorta**

Department of Computer Science and Engineering,

University of South Carolina

Abstract

The wealth of data collected by automated systems, or pen and paper processes, and available on the World Wide Web, is staggering. With these vast amounts of data come amazing possibilities for data mining and learning of data patterns through automated and semi-automated tools. However, these tools are inherently reliant on the quality of the data with which they are supplied. Without a thorough understanding of our data's quality and the effects of poor quality on our learning algorithms, we are consequently unable to accurately judge the quality of our models. We have distilled the ideas of prominent data quality researchers into four features of data sources: data accuracy, completeness, timeliness, and consistency. After understanding various dimensions of data quality we were able to test the effects of inaccurate data on learning of Bayesian Networks, specifically in Parameter Estimation and Structure Learning. By testing the learning algorithms with differing percentages and amounts of inaccurate data, we have begun to develop guidelines for adjusting the learning techniques and judging the quality of the learned models.

1. Data Quality Metrics

Assessing the quality of our data is a difficult process and is ripe with subjectivity. Researchers have been attempting for some time to create quantitative metrics to accurately judge the quality of our data. There are many different measurements that these researchers use to assess data quality. These range from the subjective –value-added and understandability, to those that are more easily quantifiable, such as completeness and timeliness. While some researchers have up to sixteen different metrics [Pipino et al 2002], we will concentrate on a core set of four for this research – accuracy/precision, completeness, consistency/believability, and timeliness, distilled from [Wand and Wang 1996], [Strong et al 1997], and [Tayi and Ballou 1998]. We will explain these terms in the context of our research.

Accuracy and precision are taken from their common scientific meanings. Accuracy represents how close a measurement, or data record, is to the real-world situation that it measures. Precision will refer to two similar ideas in this research. Firstly, it will refer to the standard deviation or variation in a numerical data record with multiple readings. As an example, if a weather sensor is calibrated before its use to have a variation of $\pm$ 0.01°C, we would use this calibrated range as the precision of the instrument. The second use for precision is to quantify the degree to which a sensor, or other data input, gives a consistent data output for a given input, regardless of the accuracy of these results. Of course, we would wish our data to be both accurate and precise. While accuracy is a metric that must be calculated over time and with much diligence to discover discrepancies between what is contained in the real world and what is represented in the data record, precision can more easily be calculated from the data at hand.

Completeness in this research will refer to the amount of missing data records. This can be computed as the total number of missing data, or the total number of rows containing missing data. Each can be useful in our research. This metric is easily calculated from the data at hand, and requires no subjective insertions from the user or data manager. Completeness is especially important in our research, as we can gauge the effects of using approximation algorithms, such as Gibbs Sampling or the Expectation-Maximization (EM) algorithm, in our learning techniques. Large amounts of missing data can also point to problems in the data collection method and the data collection tools.

Timeliness of the data is also important in the context of our data set. Data that is outdated is often useless in many fields of study, such as weather data records, or stock market data that must be used to make buying and selling decisions in seconds. However, in other fields the timeliness of data is not as crucial. If historical data is being used to track consumer spending over the last decade, having data within minutes of the real-world event is not as important. This metric is therefore at once both subjective and objective. If the data is time-stamped, it is not difficult to determine the time difference between the entry and the present time. However, we must have allowed a Subject Matter Expert to then guide the program to determine how current the data should be for that particular purpose.

Lastly, our research will employ the idea of consistency or believability. These metrics are similar to precision, except will be used here to compare data from different data sets. Where precision applies to one particular data source – a particular sensor or manually inputted data source, consistency/believability will apply to multiple data sets reporting on the same real-world situation. If we have three sensors reporting weather information from one location (or small radius) we can measure the consistency of the data for that region as the deviation between the data records from each sensor. Similarly, if we have eyewitness accounts at the scene of a crime, we can judge the consistency of the data set by the similarity of the accounts. This metric is computed separately from the accuracy and precision calculations and gives no weight to one data source over the other. If there are three data sources, we will calculate one consistency metric for that data record which takes into account differences from all the data sets, without declaring which data source is most accurate. This allows us to account for differences in data without knowing the accuracy of the data set. This may seem faulty, and indeed if we have the resources to judge the accuracy of the data sets then we can assign data quality based on those findings. However, consistency allows us a metric to determine that there are indeed differences in data sources, without tracing the data sets back to the sources and having to exhaustively determine which data set is most accurate. Consistency is an easier metric to calculate and more likely to be available for use in our algorithms.

2. Testing the Effects of Inaccuracy

While all four data quality dimensions will eventually be thoroughly studied in our research, the focus of this paper will remain on inaccuracy and its effect on learned Bayesian Networks (BNs). In order to understand these effects, a set of tests was constructed. Two test BNs were modeled – one that will be considered the "true" state of the BN and another, "bad" data set in which all of the potentials were set at 0.5 (equal likelihood of either result). Data sets were then generated with 0%, 10%, 25%, 50%, 75%, and 90% "dirty" data by creating one data set with the "true" data records and a second with the "bad" data and melding them together. For example, to generate a 10% dirty set for 100 records, one would generate 90 cases from the "true" data set and 10 cases from the "bad" data set and combine them for a 100 record set. This was done for 100, 500, 1000, and 10,000 record sets.

For purposes of this research, two traditional BNs were studied – Trip to Asia and the Stud Farm network. The structures of these networks are shown in Figures 1 and 2 respectively.

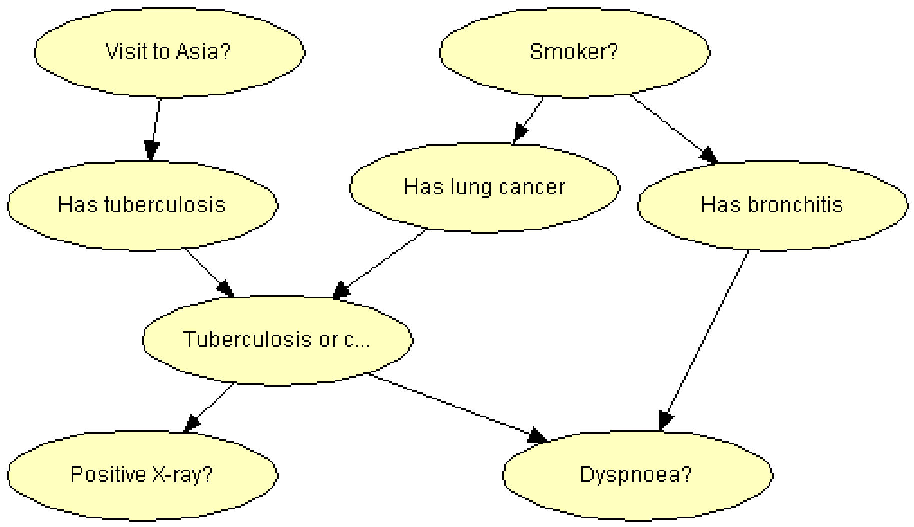


Figure 1 Trip to Asia

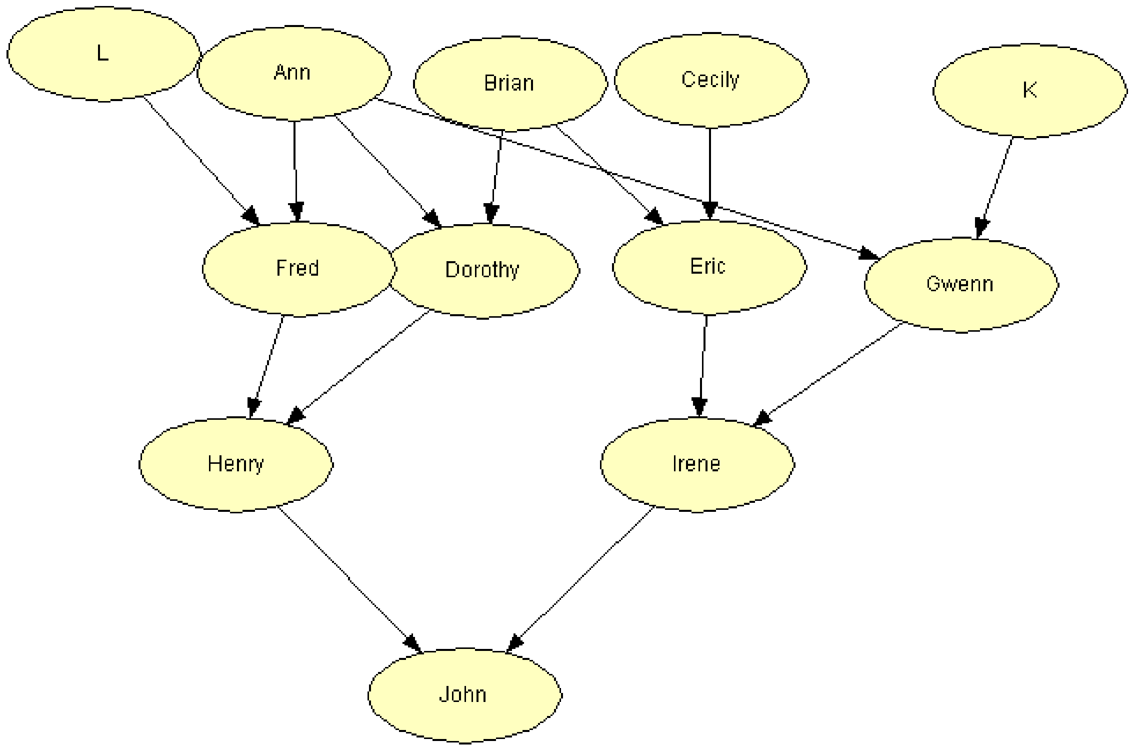


Figure 2 Stud Farm Network

After creating the data sets, the generated data were given to three BN learning programs – a Parameter Estimation algorithm, the Hugin Researcher™ Structure Learning tool that implements the NPC and PC algorithms, and Visual CB, an implementation of the CB algorithm developed by [Singh and Valtorta 1995]. See also [Neapolitan 2004]. The pseudocode for the Parameter Learning algorithm is displayed below in Figure 3. The input to the code is a Hugin™ .net file as well as the data sources either in a flat file or XML schema. The algorithm uses either the linear calculation, or the mean of the beta distribution for the potential calculation. This code can be easily manipulated for testing our new algorithms by changing the potential calculations of each data set.

Parameter Estimation Pseudo code

```
Input: Bayesian Network structure in form of Hugin .net file, flat file or
        XML schema of data sets

Parse structure – get nodes and states of nodes from Hugin structure
For each Node
    If Node with no parents
        i = count total number of entries for that node
        For each state of node (yes, no, other, etc)
            j = Count entries of that state
            Calculate beta distribution for that state
            Potential for the state = effective value, $E(F) = j/i$,
                or mean of beta distribution
    If Node with parents
        For each state of parent
            i = count total number of entries where parent = that state
        For each state of node
            j = count number of entries where parent = current state and node = current state
            Calculate beta distribution for that state
            Potential for the state = effective value, $E(F) = j/i$,
                or mean of beta distribution

Output: Bayesian Network with learned potentials in form of Hugin .net file
```

Figure 3 Parameter Estimation Pseudocode

The potentials generated by the algorithm were compared to the baseline of learning from 0% dirty data. The variance between the baseline and the dirty data sets was then calculated (standard variance calculation - $\sigma^2 = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$, where α and β are standard parameters of the beta distribution). Then the average variance was calculated for each size of the data set - 100, 500, 1000, and 10,000 data records. Also, the data sets' potential calculations were graphed to show differences in original and dirty data sets.

In order to test the effects of inaccurate data on the Hugin™ Structure Learning algorithms, the dirty data sets were inputted to this program and compared to a structure learned from a baseline of 0% dirty data. These structures were compared for missing links, reversed links, and extra links to determine the effects of this inaccurate data on the learned structures. Further, the data sets were run through a second Structure Learning algorithm developed by [Singh and Valtorta 1995] – the CB algorithm. This algorithm is a combination of a conditional independence testing algorithm with the traditional K2 algorithm for determining BN structure. These structures were also compared for missing links, reversed links, and extra links against the baseline data.

3. Parameter Estimation Results

The resultant potential calculations for the Trip to Asia experiments are shown in tabular form in Figure 4 and graphical form in Figure 5.

Average Variance						
	Baseline	**10 percent**	**25 percent**	**50 percent**	**75 percent**	**90 percent**
100	0.018178	0.032588	0.033372	0.040341	0.063067	0.072432
500	0.008783	0.019858	0.02729	0.042288	0.053119	0.070345
1000	0.008304	0.025121	0.026132	0.039092	0.053945	0.067608
10000	0.001273	0.019202	0.028597	0.039567	0.055699	0.068735

Figure 4 – Average Variance From Baseline

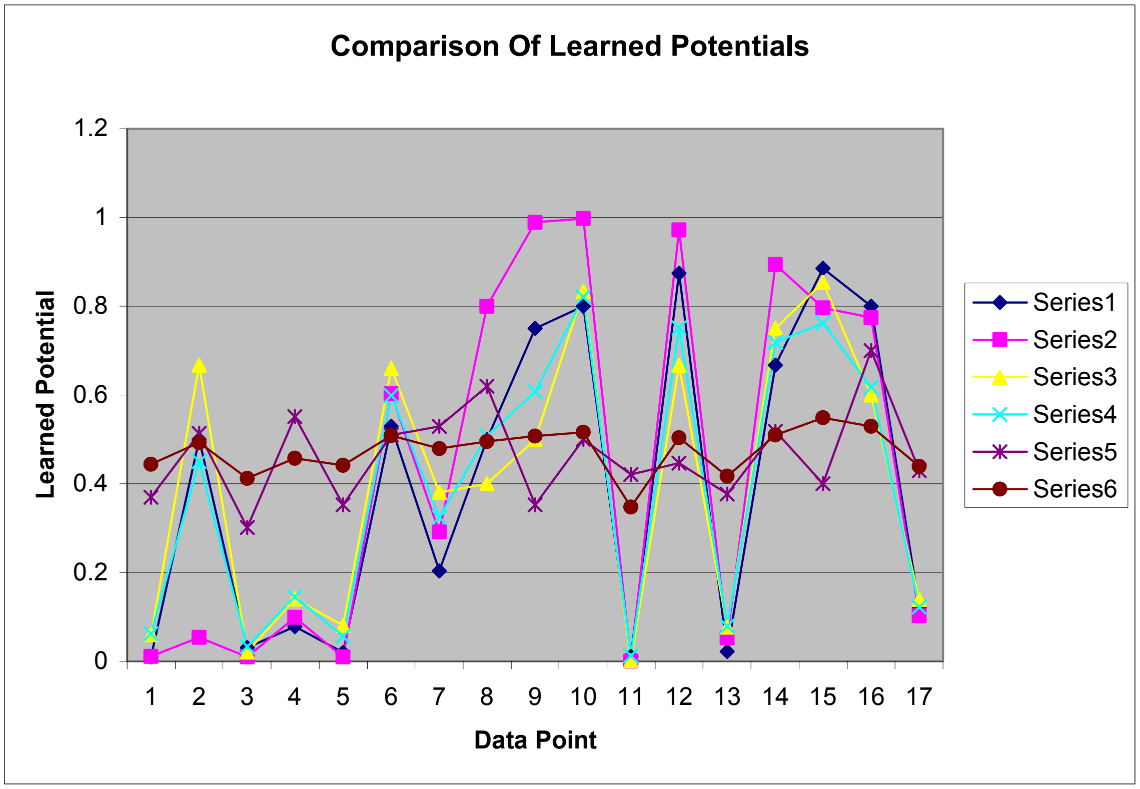


Figure 5 – Comparison of learned potentials from dirty data sets.
Series 1 – Baseline Data Potentials learned from 0% dirty data on 100 records
Series 2 – Baseline Data Potentials learned from 0% dirty data on 10,000 records
Series 3 – Potentials learned from 10% dirty data on 100 records
Series 4 – Potentials learned from 10% dirty data on 10,000 records
Series 5 – Potentials learned from 90% dirty data on 100 records
Series 6 – Potentials learned from 90% dirty data on 10,000 records

The results from the graph are surprising at first, because it seems as though the smaller data sets yield better results than the larger ones. This is not the case however, and the apparent rise in variance as the data set gets larger is due to normal variances of the beta distribution. While the distribution of "good" and "bad" data sets is important, the results do not change on average with a rise in the total number of data records used. We will prove this in two parts. Firstly, we will prove that the potential of a state is due to the distributions of the data, and is not affected by the amount of data used to learn the data sets. Secondly, we will show that the variance from the original learned data sets is directly related to the inherent properties of the beta distribution.

Theorem 1 – The learned potential of a state is invariant with respect to the number of "good" and "bad" data sets, n_g and n_b, respectively, when the ratio $\frac{n_g}{n_b}$ is fixed.

Proof: Consider a binary variable, say X, in state 'yes'. The potential of X = yes is ϕ_{good}. If a sample of size n_g is obtained by simulation, the number of cases in which X = yes has mean $n_g * \phi_{good}$.

Consider a second distribution in which the probability of X = yes is ϕ_{bad}. If a sample of n_b size is obtained from this distribution, the number of cases for which X = yes has mean $n_b * \phi_{bad}$.

Considering a sample that is an aggregate of the above two, we obtain the fraction of the cases for which X = yes is

$$\phi_{overall} = \frac{(n_g * \phi_{good}) + (n_b * \phi_{bad})}{(n_g + n_b)}. \text{ (1)}$$

To prove that this ratio is invariant with respect to $n_g + n_b$ when $\frac{n_g}{n_b}$ is fixed, we will replace n_g with $k * n_g$ and n_b with $k * n_b$.

$$\phi_{overall} = \frac{((k * n_g) * \phi_{good}) + ((k * n_b) * \phi_{bad})}{((k * n_g) + (k * n_b))}.$$

This reduces to

$$\phi_{overall} = \frac{k * ((n_g * \phi_{good}) + (n_b * \phi_{bad}))}{k * (n_g + n_b)}.$$

The factor k cancels out, proving that the $\phi_{overall}$ is invariant with respect to the size of the overall data set, when $\frac{n_g}{n_b}$ is fixed.

Lemma 1 - The variances in the data sets that are apparent in the experiment are due to the inherent properties of the beta distribution.

Explanation
The formula for the beta distribution is

$$B(\alpha,\beta)=\frac{(\alpha-1)!(\beta-1)!}{(\alpha+\beta-1)!}\ .$$

The mean is given by $\mu=\dfrac{\alpha}{\alpha+\beta}$ and the variance by

$$\sigma^2=\frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}.$$

We proved in Theorem 1 that the mean is invariant with respect to additional overall data when the ratio of "good" data to "bad" data is constant. Now we will see if the variance is also invariant with respect to $n_g + n_b$ when the ratio $\dfrac{n_g}{n_b}$ is fixed.

We have the same n_g, n_b, ϕ_{good}, and ϕ_{bad} from the previous proof. Substituting for α and β we have

$$\sigma^2=\frac{[(n_g*\phi_{good})+(n_b*\phi_{bad})]*[(n_g-(n_g*\phi_{good}))+(n_b-(n_b*\phi_{bad}))]}{(n_g+n_b)^2(n_g+n_b+1)}.$$

Substituting n_g with $k*n_g$ and n_b with $k * n_b$ we have

$$\sigma^2=\frac{[(kn_g*\phi_{good})+(kn_b*\phi_{bad})]*[(kn_g-(kn_g*\phi_{good}))+(kn_b-(kn_b*\phi_{bad}))]}{(kn_g+kn_b)^2(kn_g+kn_b+1)}.$$

Factoring k we have

$$\sigma^2=\frac{k^2[(n_g*\phi_{good})+(n_b*\phi_{bad})]*[(n_g-(n_g*\phi_{good}))+(n_b-(n_b*\phi_{bad}))]}{k^2(n_g+n_b)^2(kn_g+kn_b+1)}$$

$$\sigma^2=\frac{[(n_g*\phi_{good})+(n_b*\phi_{bad})]*[(n_g-(n_g*\phi_{good}))+(n_b-(n_b*\phi_{bad}))]}{(n_g+n_b)^2(kn_g+kn_b+1)}.$$

We cannot factor out the k from the denominator.

The $\lim\limits_{k\to\infty}\sigma^2=0$. Therefore, as k gets larger the variance becomes smaller.

Results from the Stud Farm Network are similar, but were run on only 100 and 10,000 data records, for the baseline, 10%, 50%, and 90% dirty data. The Parameter Estimation results are shown in tabular form in Figure 6, and graphically in Figure 7.

Average Variance				
	Baseline	**10 Percent**	**50 Percent**	**90 Percent**
100	0.067763969	0.080317446	0.083564781	0.115821561
10000	0.015885502	0.0203215	0.065876298	0.047815416

Figure 6 – Stud Farm Average Variance from Baseline

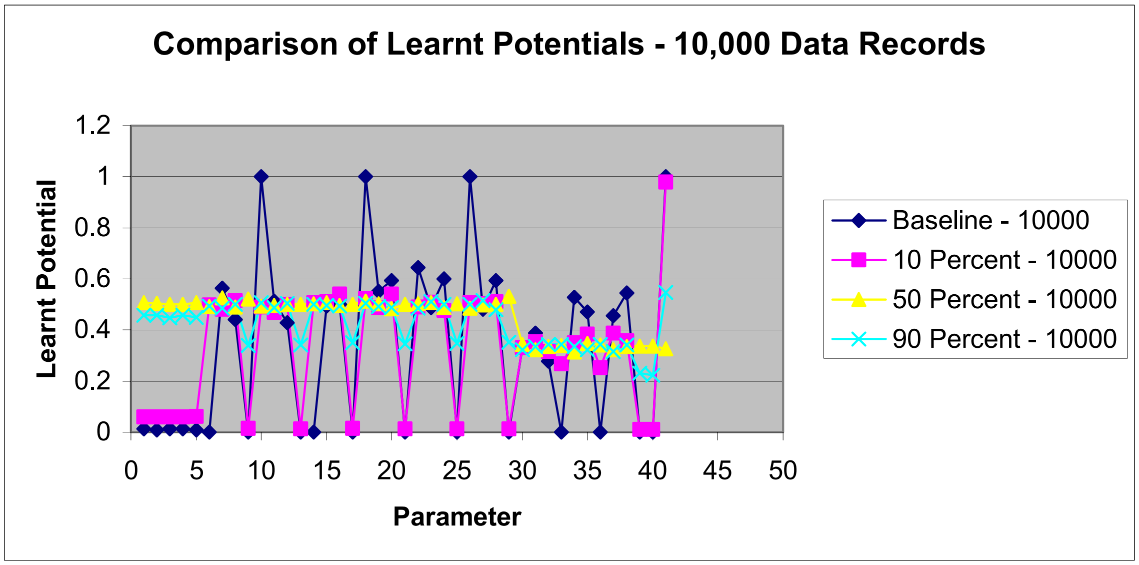


Figure 7 – Stud Farm Learnt Potentials Graph

4. Structure Learning Results

The results of the structure learning tests for the NPC/PC algorithm using the Hugin Researcher™ tool, and the CB algorithm, are shown in tabular form in Figure 8. Because the results from the NPC and PC tests were very similar, we will display only the NPC results for ease.

	NPC-100	NPC-500	NPC-1000	NPC-10000	CB-100	CB-500	CB-1000	CB-10000
Total Wrong								
Baseline	6	5	4	4	4	5	7	2
10 Percent	6	10	9	13	11	14	13	16
25 Percent	7	5	10	14	15	14	12	14
50 Percent	7	11	10	15	8	15	13	14
75 Percent	12	9	9	10	15	9	12	14
90 Percent	6	8	10	10	7	9	9	11

Figure 8 – NPC and CB Structure Learning Results

The total incorrect links are calculated by adding the extra links, reversed links, and missing links. This is displayed in graphical form for 100 and 10,000 data records in Figure 9.

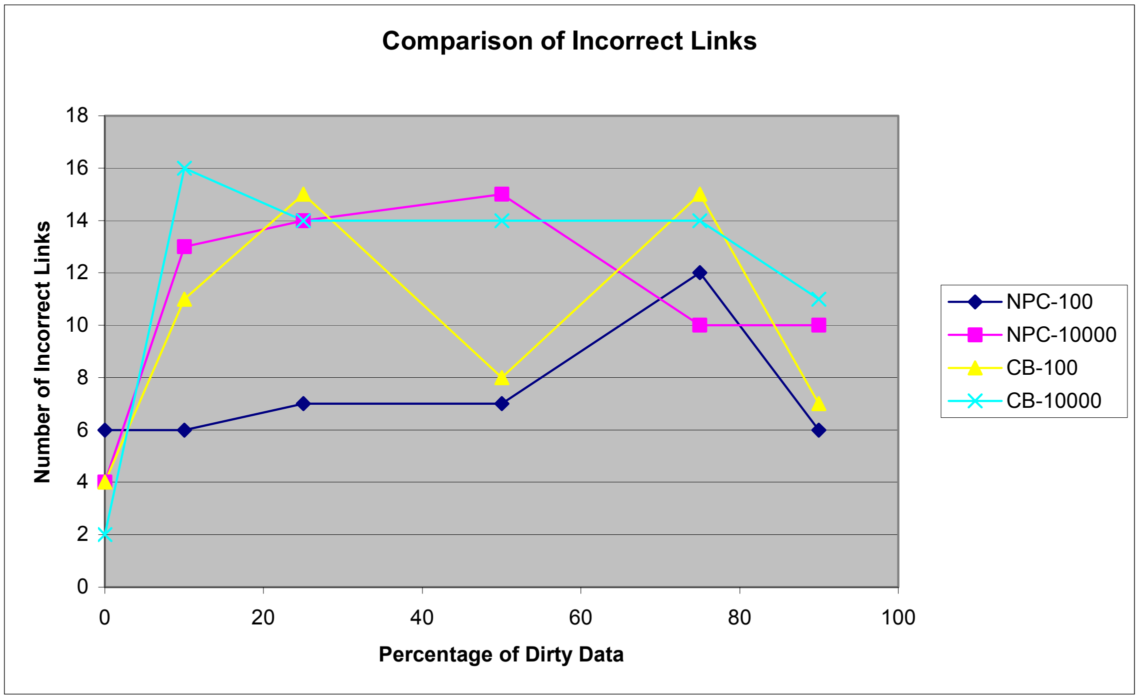


Figure 9 – Comparison of Incorrect Links – NPC and CB Algorithms

These results are also deceiving at first, as it appears that the total amount of wrong data increases until approximately 75% of dirty data, and then declines at 75-90% dirty data. This is not the case, however, as revealed by a careful examination of the total correct, missing, reversed, and extra links. At around 25-75% missing data, the learning algorithms begin to discover about twice as many total links as in the 0-10% and 75-90% dirty data sets. With the larger amounts of conflicting data, the algorithm finds links in both the 50-50 “bad” data set and the true data set – giving us twice as many links – both correct and incorrect. As the percentage of 50-50, or “bad” data increases to 75-90%, or as the “bad” data percentage is low, 0-10%, the algorithms finds only one set of links. This can be seen as we graph the total number of incorrect links divided by the total number of links found. This is shown in tabular form in Figure 10 and graphically for 100 and 10,000 data records in Figure 11.

A similar result is noted in [Spirtes et al 2000]. The authors discovered that if the PC algorithm fails to find an edge from the true graph in the second stage of the algorithm, then the algorithm fails in subsequent steps to remove nonexistent edges. They also discovered that the failure to find an edge in the true graph could lead to orientation mistakes in subsequent stages.

Total Wrong/Total Links	NPC-100	NPC-500	NPC-1000	NPC-10000	CB-100	CB-500	CB-1000	CB-10000
Baseline	2	1.25	0.8	0.666666667	0.5	0.555556	0.636364	0.25
10 Percent	1.5	1	0.75	0.684210526	0.7857143	0.823529	0.8125	0.761904762
25 Percent	1.75	0.625	0.714285714	0.666666667	1	0.823529	0.666667	0.7
50 Percent	2.333333333	0.785714286	0.769230769	0.652173913	0.6666667	0.882353	0.866667	0.666666667
75 Percent	3	0.818181818	0.75	0.714285714	1.875	0.692308	0.923077	0.736842105
90 Percent	3	4	2	0.769230769	1.75	9	1.5	0.6875

Figure 10 – NPC and CB Structure Learning Results Total Incorrect/Total Links

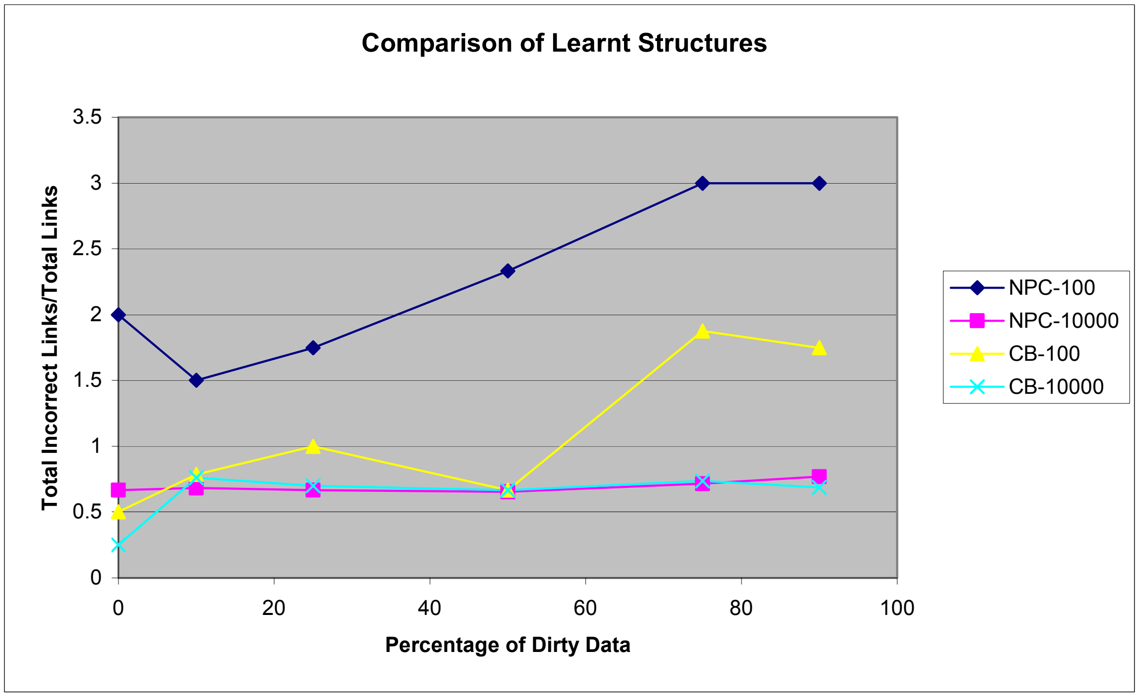


Figure 11 –Comparison of Incorrect Links – Total Incorrect/Total Links

Results from the Stud Farm network were similar and are shown in tabular form in Figure 12 and graphically in Figure 13.

Total Incorrect Links/Total Links	NPC-100	NPC-10000	CB-100	CB-10000
Baseline	2.6	0.071428571	0.416666667	0
10 Percent	2.333333333	0.138461538	0.269230769	0.078125
50 Percent	1.4	1.714285714	0.083333333	6.5
90 Percent	2.166666667	0.134328358	0.282608696	0.093023

Figure 12 Stud Farm NPC and CB Structure Learning Results

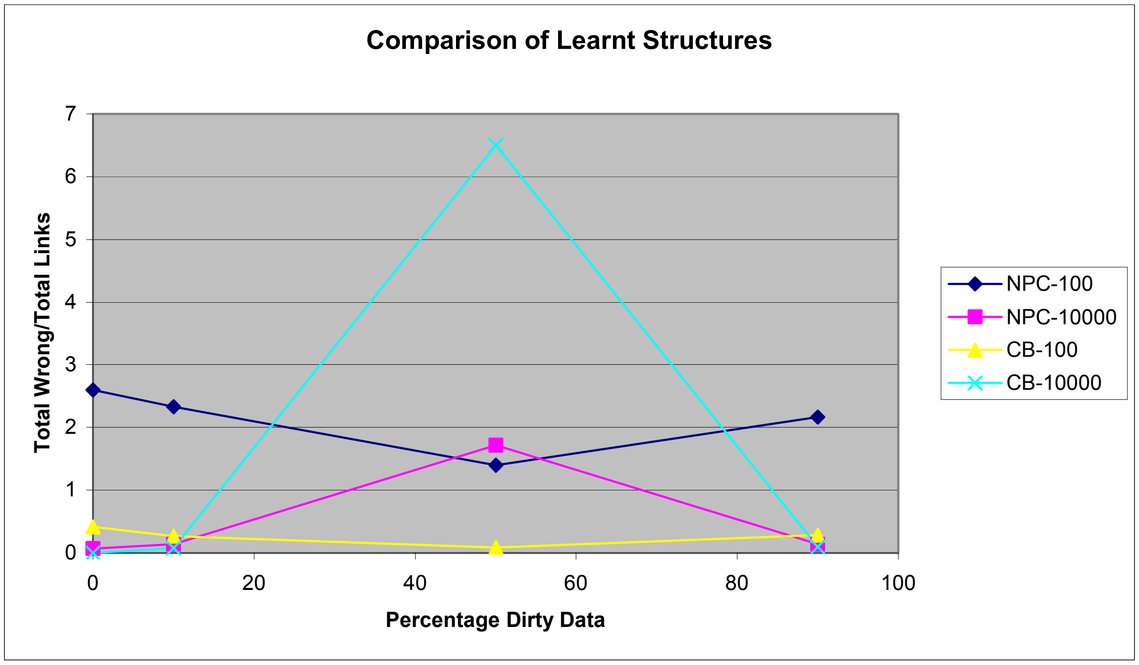


Figure 13 Stud Farm Comparison of Incorrect Links – Total Incorrect/Total Links

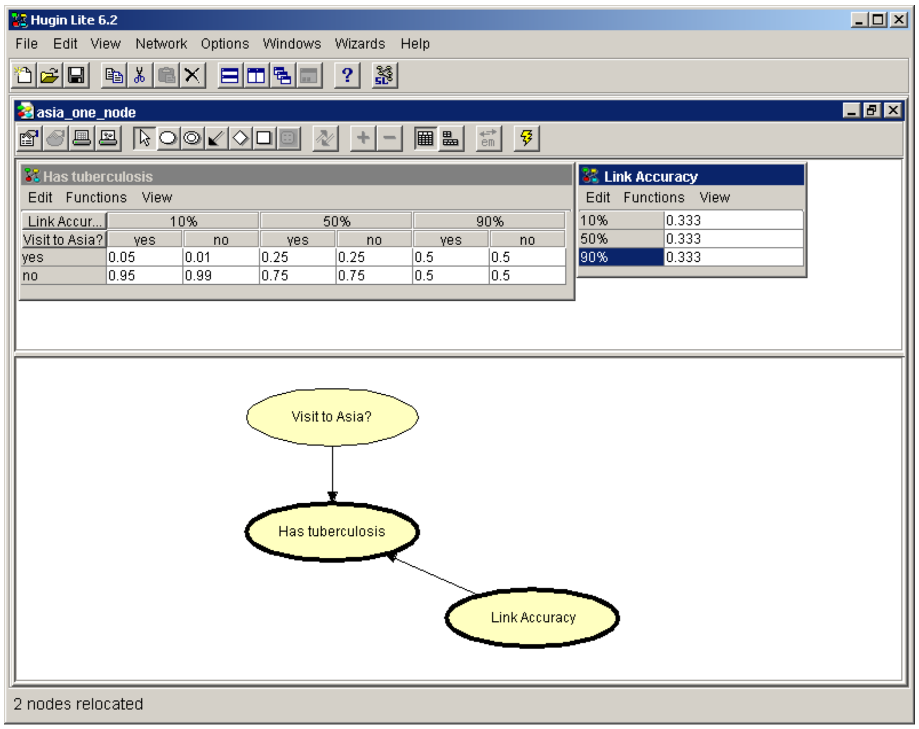
5. Conclusions

Even from a cursory overview of the results of these experiments, it is evident that Parameter Estimation algorithms are more tolerant of inaccuracies in data than the Structure Learning algorithms we have studied – NPC and CB. At only 10% dirty data, the CB algorithm with 10,000 data records increased the number of incorrect links from 2 (baseline) to 16 - an 800% increase. NPC also had an increase in incorrect links of 325% for 10,000 records with only 10% dirty data. The results after 10% dirty data, while not increasing at a significant rate, also do not become closer to the original structure, as would be expected. There does not seem to be a significant advantage to either of these Structure Learning algorithms in terms of handling inaccurate data, although the CB algorithm appears, in these two examples, to create better structures for smaller data records, and NPC seems to create better structures with larger amounts of data. Therefore when dealing with small data sets of inaccurate data, the CB algorithm may be a better choice for learning structures. The most significant finding of this research, though, is that even at small amounts of inaccurate data, these Structure Learning algorithms are unable to generate correct structures. Therefore a thorough understanding of the quality of the data is very important when using Structure Learning algorithms, as even small inaccuracies can create completely incorrect models.

Our parameter estimation algorithm, however, does follow a gradual increase in inaccurate modeling when learned from inaccurate data sets. As the inaccuracy of the data increases, the learned potentials also deviate from the original and increase gradually according to percentage of missing data. Therefore, unlike the Structure Learning algorithms we have studied, BNs learned from inaccurate data can be useful when the potentials are given a range of accuracy based on the believed accuracy of the data. Therefore if a model is believed to be based on data that is 90% accurate with 100 data records, the potentials can be given with a standard deviation of approximately 0.18 ($\sqrt{\text{variance}} = \sqrt{0.032588}$). This variance was taken from the Trip to Asia data results, however we realize that the Stud Farm standard deviation for 10% missing data would be higher – 0.28. Our future research will strive to verify an average potential deviation for the various percentages of inaccurate data that can be applied to a wide range of new models.

6. Future Research

Future research shall include methods for incorporating the results of these experiments into learning algorithms for BNs. For Parameter Estimation this opens two areas of future research – quantifying average standard deviation figures for expressing the range of accuracy of a model learned from partially inaccurate data, and investigation into methods for discounting those sets of data that are deemed to be partially inaccurate. We believe this will add to the adaptation work of [Olesen et al 1992] which was incorporated into the Hugin™ engine. The results of these experiments are discouraging for improvement of Structure Learning algorithms for learning from inaccurate data sets, as even small amounts of inaccurate data appear to greatly impact the usefulness and accuracy of the learned models. Our research will continue to test further Structure Learning algorithms with the intent of finding an algorithm that handles inaccuracies more effectively. Along with this research, it is also our belief that the inaccuracies of the data itself can be modeled effectively to show the dependence of the model upon data quality. For example, each node or link in the learned Bayesian Network can also have a quality node that quantifies the overall accuracy, completeness, timeliness, and consistency of the data set that was used for learning. There exists a causal link from the accuracy of the data to the learned potentials and structure. An example of this idea is shown in Figure 14.



Link Accur...	10%		50%		90%	
Visit to Asia?	yes	no	yes	no	yes	no
yes	0.05	0.01	0.25	0.25	0.5	0.5
no	0.95	0.99	0.75	0.75	0.5	0.5

10%	0.333
50%	0.333
90%	0.333

Figure 14 Modeling Inaccuracies as a Causal Entity

The potential table for Tuberculosis changes to add the parameter of Link Dependence on Data Accuracy. As the accuracy of the data decreases, the causal relationship between Trip to Asia and Has Tuberculosis begins to fade and the potentials become evenly split – no dependence on one parameter over another. This allows us to change our model dynamically as our beliefs about the quality of the data changes. This is a challenging problem, as more experiments will be needed to confirm the relationship between accuracy of the data and existence of a learned causal link. Using this initial research as a guide, however, it is our belief that we can begin to understand and account for data quality in the automated learning of BNs.

7. Bibliography

Neapolitan, R.E. (2004). *Learning Bayesian Networks*. Upper Saddle River, NJ: Pearson Education, Inc.

Olesen, K., Lauritsen, S., Jensen, F. (1992) aHUGIN: A System Creating Adaptive Causal Probabilistic Networks. *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, 223-229.

Pipino, L., Lee, Y., and Wang, R. (2002). Data Quality Assessment. *Communications of the ACM*, Vol 45. 211-218.

Singh, M., and Valtorta, M. (1995). Construction of Bayesian Network Structures from Data: a Brief Survey and an Efficient Algorithm. *International Journal of Approximate Reasoning.*

Spirtes, P., Glymour, C., and Scheines, R. *Causation, Prediction, and Search.* MIT Press, Cambridge, Massachusetts, 2000.

Strong, D., Lee, Y., and Wang, R. (1997). Data Quality in Context. *Communications of the ACM.* Vol. 40. 103-110.

Tayi, G. and Ballou, D. (1998). Examining Data Quality. Communications of the ACM. Vol. 41. 54-57.

Wand, Y. and Wang, R. (1996). Anchoring Data Quality Dimensions in Ontological Foundations. *Communications of the ACM* Vol 39. 86-95.