THE EFFECTS OF DATA QUALITY ON MACHINE LEARNING ALGORITHMS

(Research-in-Progress – IQ Concepts, Tools, Metrics, Measures, Models, and Methodologies)

Valerie Sessions
University of South Carolina, USA
sessionv@cse.sc.edu

Marco Valtorta
University of South Carolina, USA
mvg@cse.sc.edu

Abstract: Our research focuses on two goals. First, we seek to demonstrate that data quality is an important component of machine learning tools and that the quality of our data should be carefully considered when developing and using these tools. We believe that while the importance of data quality is now understood in the business community, where researchers have equated quality decision making to earnings, in the machine learning community this realization has not yet occurred. Therefore, we embark upon research into the effects of data quality upon these algorithms in an effort to demonstrate that data quality is a large factor in the outcomes of these algorithms and should be given more respect in their design. Second, after providing evidence that data quality is a large factor in these algorithms, we develop and test some preliminary methods which incorporate data quality assessments, thus creating more robust and useful algorithms. We believe these modifications and methods can have profound effects upon the usage of these machine learning algorithms in actual practice, particularly in the Law Enforcement community.

Key Words: Data Quality, Information Quality, Machine Learning, Bayesian Networks

INTRODUCTION

Our research merges two communities within computer science – data quality and machine learning, specifically the field of Bayesian Networks. While data quality has made great strides in gaining the respect of the business community in the past ten years, within the machine learning/ artificial intelligence realm it has largely been neglected in order to focus more specifically on the learning algorithms and methods themselves. Most research in these fields begins with the assumption that the data feeding the algorithms is of high quality – accurate, complete and timely. Researchers that do take data quality into account normally focus on the aspect of missing data. One example is the development of the Expectation Maximization Algorithm for BNs [5], which deals well with incomplete data sets (when the existing data is of high quality). Also, [7] presented research in the area of Data Mining with Dirty Data which included research on Neural Networks. Our research seeks to expand upon these studies and further demonstrate how important data quality is to the outcomes of these algorithms and how severely they are affected by low quality data. We believe this is an important proof, and if these machine learning algorithms are to mature farther in the practicing communities of medicine or law enforcement, they must be robust in their handling of data sets of differing data quality, or at a minimum be equipped with methods to limit the effects of corrupt or poor data sets on the overall outcome of the learning algorithm.

Because both the fields of data quality and machine learning are vast, several restrictions have been made in order to make this initial research feasible. First, in terms of data quality, we will only be experimenting with the component of data accuracy and its effect on the algorithms. While we understand

that this is only a small piece of the overall data quality of our datasets, we chose to limit our research to this area for several reasons. First, accuracy is a relatively easy component to alter in test data sets. It is less contextually dependent than other metrics such as timeliness, and unlike completeness it has not been well researched in the field of Bayesian Networks. We believe that many of the methods can be easily adapted to other components of data quality in the future. The field of Machine Learning is also too large to exhaustively research the effects of data quality on all algorithms, therefore we are limiting our research to the field of Bayesian Networks (BNs) and within that field specifically the PC algorithm. The fundamentals of this field and algorithm will be given in the Background section of this paper.

With these restrictions, the specific nature of our experimentation deals with the effects of data inaccuracy on the PC learning algorithm for BNs. While this is a specific test case and example, we believe that this research still significantly highlights the effects of data quality on learning algorithms as a whole, and provides a confirmation to researchers in this area that data quality is an important element in the outcomes of these tools and should be considered when designing, enhancing, and especially using these algorithms.

After demonstrating how severely the PC algorithm is affected by inaccurate data, we propose four enhancements to the algorithm which incorporate data quality assessments. We then explain the results of these algorithm modifications, limitations to this research, and our conclusions and proposed future work in this area.

BACKGROUND

Because our research draws on both the machine learning/BN community as well as the data quality community, two background sections are needed here. We will first focus on the fundamentals of BNs and BN learning algorithms, and then briefly discuss some data quality fundamentals.

Bayesian Network Basics

We will attempt to cover the fundamentals of BNs without delving into any complex underlying formulas. For a more detailed mathematical description, we recommend [2], [3], and [5].

A BN is used to model a domain of knowledge using a set of nodes (representing variables) and a set of directed edges between the nodes. The directed edges represent a set of dependencies between the nodes. For example, we can represent the state of a wet lawn using the BN in Figure 1.

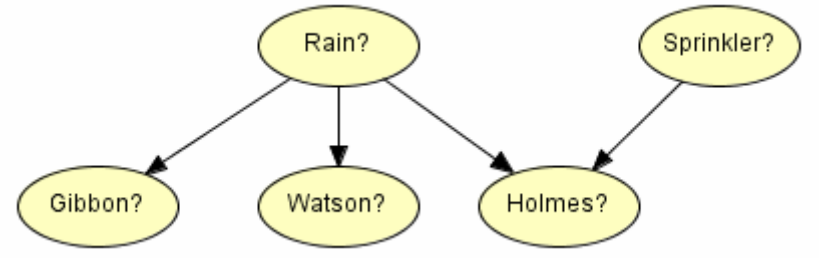
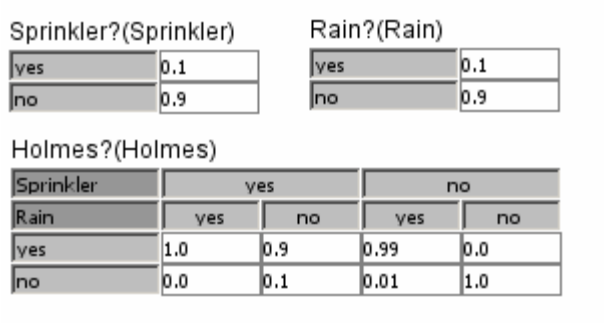


Figure 1: Wet Lawn

Mr. Holmes is deducing whether his lawn is wet because it has rained or because his sprinkler is working. He gathers evidence by looking to see if Mr. Watson's and Mrs. Gibbon's lawns are also wet. The strength of these dependencies is modeled as a probability – normally represented by a conditional probability table. The conditional probability tables for three nodes in our Wet Lawn example are shown in Figure 2.



Sprinkler?(Sprinkler)

yes	0.1
no	0.9

Rain?(Rain)

yes	0.1
no	0.9

Holmes?(Holmes)

Sprinkler	yes		no	
Rain	yes	no	yes	no
yes	1.0	0.9	0.99	0.0
no	0.0	0.1	0.01	1.0

Figure 2: Wet Lawn Probability Tables

As we see from this example, Holmes' wet lawn is dependent upon either the sprinkler or the rain. If it has not rained or if the sprinkler has not been working, the lawn will not be wet. If it has either rained or the sprinkler has been working it is a 90-99% chance that it is wet, and if it has both rained and the sprinkler is working, it is 100% chance that the lawn is wet.

Using these tables and some basic formulas, we can also update our beliefs about a situation based on new evidence. From these tables we see that the prior probability that it has rained is about 10%. But if we have evidence that Mr. Holmes', Mr. Watson's and Mrs. Gibbon's lawns are wet, this probability is now 99%. If only Mr. Holmes' lawn is wet the chance that it has rained is only 1%. This is the power of the BN – the ability to update our beliefs for the entire system based on a new piece of evidence.

This is a small sample BN, but we can see both the power of the BN and how quickly its complexity grows as more nodes and dependencies are added. These networks are developed in a variety of ways. First, we can develop these networks based on the experience of Subject Matter Experts (SMEs). The SMEs draw on years of experience in their field to develop a BN consisting of variables and dependencies, and also populate the probability tables which represent the strength of these dependencies. This method works well, but is reliant upon the experts used. We would prefer to learn directly from domain data itself, and thus a set of algorithms has been developed to do just this. One such algorithm is the PC algorithm explained in detail in [5]. This algorithm is used by the Hugin™ Decision Engine [6] which was an integral part of our research. This algorithm draws directly on the data itself to create the nodes and directed edges which make up the structure of the BN. It then uses a second algorithm, the EM algorithm, to create the probability tables. We include a description of the PC algorithm here and refer the reader again to [5] for more detailed information.

The PC algorithm begins by gleaning the node names and states from the data sets. In a relational database, this would be done by field name and then a process of scanning the fields to determine the various states this variable can be in. In our Wet Lawn example, we would find a node for Holmes that can be in state wet or dry, which would also be related to Watson's lawn which can also be wet or dry. The Algorithm then creates a complete graph from the node - that is all nodes are then connected to each other by undirected edges. Then the algorithm discovers which nodes are actually independent so that it can eliminate the edge that connects them. This is done by using a score based on I-divergence (also called KL-divergence or cross-entropy), a non-symmetric measure of the distance between two distributions, which we will call G^2 [9]. If X and Y are random variables with joint probability distribution P, and sample size m, then the G^2 score is defined as:

$$G^2 = 2m * I(X,Y) = 2m * \sum_{x,y} P(X,Y) \log \frac{P(X,Y)}{P(X)P(Y)}$$

.

For simplicity, we denote by *P(X)* and *P(Y)* the marginals of *P(X,Y)* on variables *X* and *Y*, respectively. This score measures the degree of dependence between two variables and is 0.0 only for independent variables.

Once all of the independent edges are removed, the algorithm orients the edges in a series of steps:

A. Head to Head links: If there exists three nodes *X*, *Y*, *Z*, such that $X - Z - Y$ are connected, and *X* and *Y* are independent given a set not containing *Z*, then orient the nodes as $X \rightarrow Z \leftarrow Y$.
B. Remaining Links: Three more rules govern the remaining links, each use the assumption that all Head-to-Head links have already been discovered.
    i. If there exists three nodes X, Y, Z, that are connected as $X \rightarrow Z - Y$, and *X* and *Y* are not connected, then orient $Z - Y$ as $Z \rightarrow Y$.
    ii. If there exists two nodes X, Y, that are connected $X - Y$, and there exist a path from *X* to *Y*, then orient $X - Y$ as $X \rightarrow Y$.
    iii. If there exists four nodes X, Y, Z, W that are connected $X - Z - Y, X \rightarrow W, Y \rightarrow W,$ and *X* and *Y* are not connected, then orient $Z - W$ as $Z \rightarrow W$.
C. If there are any links left, these are oriented arbitrarily, avoiding cycles and head-to-head links.

After orienting the links, the algorithm is complete.

In reality, where to determine independence is not set at exactly $G^2 = 0.0$. This is too tight of a constraint for "real" data. In Hugin's™ implementation of the PC algorithm, where to determine independence is partially handled by a value called the significance level. Hugin™ exploits the fact that G^2 is roughly χ^2 and if its tail probability, or α, is less than the significance level, dependence is assumed. This level is by default set at 0.05. Therefore, the smaller the significance level, the larger a Cross Entropy score that is considered independence.

We will examine the algorithm more thoroughly in the Results section when we go over why the algorithm is affected so severely by inaccurate data.

Data Quality Fundamentals

Due to an increased reliance upon data to support business decisions, research in the area of data quality has made great strides in recent years. Researchers have been seeking to define what is meant by such terms as accuracy, completeness, and believability, and to determine what quality factors are most important when assessing the quality of our data. [4] shows the various differences in both practitioner and the academic view of the components of data quality. While some researchers have over 20 metrics for determining data quality, we will focus solely on data accuracy for this research. Most researchers include accuracy in their list of data quality components, and we will follow the definition and ideas of [10].

Accuracy is considered how close a measurement, or data record, is to the real world situation it represents. It is normally considered an intrinsic data quality component, independent of its context within the system. While much more complex and powerful metrics exist to detail our data quality, we chose to focus on accuracy because it is the easiest to manipulate in our test data sets and it is of universal concern to researchers and decision makers.

Assessing the quality of our data has normally been an ad hoc process and ripe with subjectivity. The

focus in the last 5-10 years in data quality studies has been to determine a method for assessing our data in a more uniform and consistent way. Most of these assessment methods can be incorporated into our research, but our methods meld most closely with the Total Data Quality Management (TDQM) assessments. Following [4], the most common functional form of accuracy assessments involves a ratio of the desired outcomes divided by total outcomes subtracted from 1. This gives us a ratio between 1 and 0 that describes our data quality. We interpret a number closer to 1 as more accurate and closer to 0 as inaccurate. In our research, it is not necessary to assess the quality of our data, because we are using test data sets from BNs of high quality and corrupting them with inaccurate data. Therefore, we already know the quality of our data because we have created that ratio and quality level. This is necessary for our tests, and without knowing the data quality with certainty, the effect on the machine learning algorithm would not be the true dependent variable in our experiments. In future research, however, it is our goal to use more practical data sets that we will assess using the TDQM method. We will explain this more fully in the Limitations section.

RATIONALE AND PURPOSE

One focus of our research is to demonstrate how important data accuracy is to the PC algorithm, and in the future to other machine learning algorithms. In order to understand how data accuracy affects the PC algorithm, a set of tests was developed. For the purposes of our research we tested using three traditional BNs – Visit to Asia, Studfarm, and the ALARM network. Due to space constraints we cannot include a complete description of each network in our paper, however Visit to Asia is described in [2], Studfarm in [3], and ALARM in [1]. In order for the reader to understand the size of these networks; Visit to Asia consists of 8 nodes with 8 directed edges, Studfarm is slightly larger with 12 nodes and 14 directed edges, and finally the ALARM network is much larger with 37 nodes and 45 directed edges.

To create test data sets for these networks, two test BNs per network were modeled – one that will be considered the "true" state of the BN (with default Hugin™ example network potentials), and another, "bad" network in which all of the potentials were set at 0.5 (equal likelihood of either result). It is not possible to include the potential tables for each of these networks in our paper, but these can be obtained through emailing the authors. Inaccurate data sets were then generated in the following way. Using the Hugin™ data generation tool, data sets were generated from both the "true" network and the "bad" network. These were then combined in different ratios of true/bad data – 100/0, 90/10, 25/75, 50/50, 75/25, and 10/90, in order to obtain inaccurate data sets for our tests. As an example, in order to generate a 10% inaccurate set for 100 records, one would generate 90 cases from the "true" network and 10 cases from the "bad" network and combine them for a 100 record set. We used only data sets of 10,000 data records in order to limit the effects of variance due to the size of the data sets. Table 1 shows the overall categories developed and a breakdown of true/bad data within each category.

	Number of	Data Records
	True BN	Bad BN
Gold Standard	10000	0
Cat A1	9000	1000
Cat A2	7000	3000
Cat A3	5000	5000
Cat A4	3000	7000
Cat A5	1000	9000

Table 1: Test Data Sets

Using these data sets we ran a series of initial tests in which the inaccurate data sets were inputted to the program and compared to a structure learned from a baseline of 0% inaccurate data. We are interested in

determining what effect the inaccurate data has on the structures that the PC algorithm learns. Therefore, the resultant structures were compared and scored based on correct edges, misoriented edges, false positive (extra) edges, and false negative (missing) edges. Using a causal interpretation of BNs, we then categorized the number of incorrect edges as the addition of the number of misoriented, false positive, and false negative edges. The results for Visit to Asia are show in tabular form in Table 2, and graphical form in Figure 3.

	Baseline	Cat A1	Cat A2	Cat A3	Cat A4	Cat A5
Total Incorrect Edges	4	13	14	15	10	10
Zeroed from Baseline	0	9	10	11	6	6

Table 2: Initial Results Visit to Asia

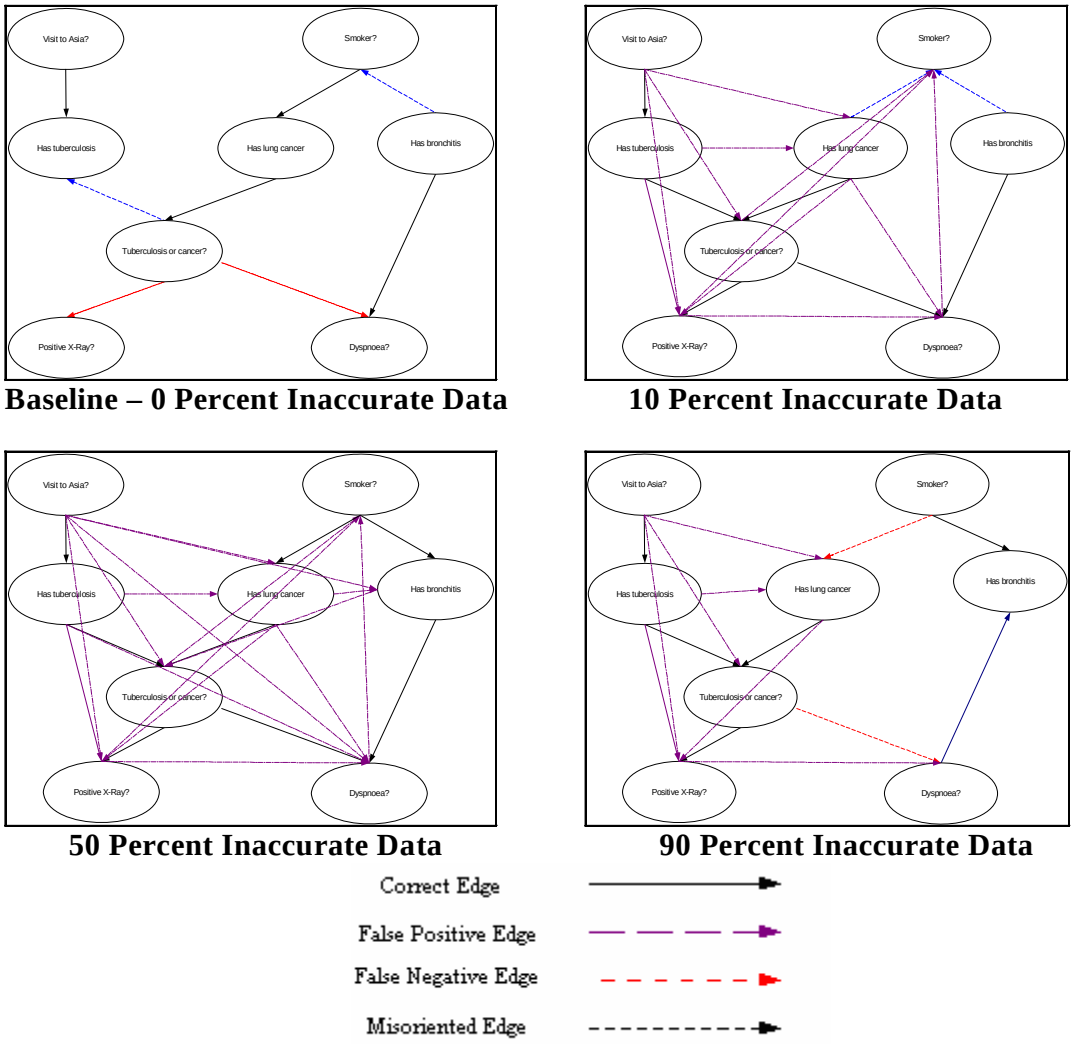


Figure 3: Initial Results Visit to Asia

These results are surprising at first, because it seems that the structures steadily become worse until about Category A4 data and then improve. This is not the case, however, as revealed by a careful examination of the total correct, false negative, misoriented, and false positive edges. Around Category A2/A3 data,

the learning algorithm begins to discover twice as many total edges than the Baseline/Category A1 or Category A4/A5 data sets. With the larger amounts of conflicting data, the algorithm finds edges from both the "bad" data set model and the true model – giving us twice as many edges – both correct and incorrect. As the percentage of inaccurate data increases, or as the inaccurate data percentage is low, the algorithms learn only one set of edges, either a correct set or an incorrect set.

A second set of tests was designed with more complexity. These test paired multiple data sets together – for instance pairing Category A1 data with Category A5 data. These tests were originally designed to test the methods we have developed which incorporate data quality assessments into the PC algorithm. We received surprising results from these tests, however, which we will explain here. We conducted tests for Visit to Asia, Studfarm, and ALARM data sets and were surprisingly not able to complete the tests for the ALARM data. At only 10% inaccurate data, the Hugin™ Decision Engine ran out of memory when trying to learn from the data. After some trial and error testing, we discovered that this was not a result of the size of the data set, but rather was based on the percentage of inaccurate data within the data set. We adjusted our percentage of inaccurate data, and finally at 0.5% inaccurate data the learning algorithm would complete.

This led us to more thoroughly examine the specifics behind the PC algorithm and why we believe it is affected so severely by dirty data. Recalling our earlier explanation, the PC algorithm operates in a series of steps. The first step is to create a complete graph from the nodes – to connect each node to every other node. Then the algorithm determines, for each edge, if the nodes connecting it are independent using the Cross Entropy score. If they are indeed independent, the edge is removed. The number of independence tests required of the PC algorithm is dependent upon the number of nodes and the degree of each node. We will call the number of nodes *n*, and the maximum degree of the nodes, *k*. The upper bound is then given by

$$\frac{n^2(k+1)(n-2)^k}{k!}.$$

As is noted in [9], and which we will confirm here, normally this worse case is not achieved. From earlier research which we used in the development of our methods, we determined that the average degree of each node in our example networks is 2.22, and the maximum degree is 5. These numbers are empirically derived from examining sample canonical BNs and while not scientifically verified numbers, they serve us well in this research. Using this degree and maximum degree number for good BNs created with clean data, this gives us a running time for ALARM of

$$\frac{37^2 * 6 * 35^5}{5!} = 3.60 * 10^9.$$

However, when we give the algorithm inaccurate data we hit the worst case scenario – 37 nodes and maximum degree of 36 (complete graph) -

$$\frac{37^2 * 37 * 35^{36}}{36!} = 5.25 * 10^{18}.$$

This makes the computation infeasible.

Why do we hit the worst case because of our inaccurate data? In order to determine if each of the nodes is independent we use a score based on I-divergence (also called KL-divergence or cross-entropy):

$$CE(X,Y) = \sum_{x,y} P(X,Y) \log \frac{P(X,Y)}{P(X)P(Y)}.$$

This score is 0.0 when the nodes are independent and is positive otherwise. If we have even one data record which points to a dependency, the score is non-zero, and therefore many of the edges from the complete graph remain in the learned structure. The reason we are able to run the algorithm at 0.5% inaccurate data for ALARM, is because of the number of significant digits we use for the score – in other words where we decide to round to zero. In Hugin's™ implementation of the PC algorithm, where to determine independence is partially handled by the significance level. This is by default set at 0.05. If the Cross Entropy score is less than this significance level, dependence is assumed. Otherwise the nodes are considered independent. Therefore, the smaller the significance level, the larger a Cross Entropy score that is considered independence. Inaccurate data affects this algorithm severely because it only takes a few (inaccurate) data records which point to a dependency to keep the edge in the learned graph structure and therefore lead to the worst case runtime scenario.

Our test results confirm this observation. Even from a cursory overview of the results in Table 2 and Figure 3, it is evident that the PC learning algorithm is very intolerant of inaccuracies in the data. At only 10% inaccurate data, the algorithm had an increase in incorrect edges of 325% for 10,000 records. The results after 10% inaccurate data, while not increasing at a significant rate, also do not become closer to the original structure, as would be expected. These tests demonstrate that data quality, specifically data accuracy, is a very important component in the outcomes of the algorithm. It is not surprising that introducing inaccurate data causes the algorithm to create structures that are not as useful as the original BN. What is distressing is that the algorithm degrades so quickly under inaccurate data and not only creates less useful BNs, but also cannot complete its calculations when confronted with a large network and small percentage of inaccurate data.

Because of these results, we believe it is important to design our learning algorithms with the ability to incorporate data quality assessments. We have created and tested four methods for incorporating data quality assessments into the PC algorithm as implemented by the Hugin™ Decision Engine. These methods are entitled "Do Nothing", "Threshold", and the "DQ Method Part One, Two, and Three".

The Do Nothing method ignores all of the data quality categories and simply combines the data sets into one. This data set is then used by the Hugin™ Decision Engine to create a BN. The purpose of this method is to further our inaccuracy research and create a baseline for determining the effectiveness of enhancements to the algorithms.

The Threshold and DQ Method require that a Data Quality Assessment be conducted. We have designed the methods in such a way as to allow for a variety of such assessments. The code for the current design of the program, however, requires the user to enter a DQ weight between 1 and 5, corresponding to the five categories of data sets we have developed. These weights can easily be deduced from a ratio between 1 and 0 following [8] and the program can be easily changed to allow for this ratio instead of our more arbitrary 1 through 5 rating. If an assessment is less exact, and relies on more subjective terms, the user must equate these to a weight between 1 and 5.

The Threshold method takes into account the data quality associated with each data set. This method establishes a threshold measure that the data must meet in order to be used in creating the BN. This method can be likened to a bouncer in a bar – if your data is not up to the standard, it is not incorporated into the learning algorithms. This serves as a method for filtering out the most inaccurate data sets and aids those modelers with a good understanding of their data.

Last, we have the DQ Method. This method was created specifically for dealing with inaccurate data. As input into our original DQ Method, we provide the method with an assessment/confidence level between 1 and 5 corresponding to our data set categories. The higher the assessment, the higher that data set was weighted over the other data sets imported to the PC algorithm. We believed that by downplaying the effects of seemingly inaccurate data over more accurate sets, we would create more robust BNs. This did

not have the overall results we were seeking, because of our earlier finding that the PC algorithm is grossly affected by inaccurate data, even in small increments of inaccuracy. Also, as the research in Data Quality confirms, knowing the quality of our data is a luxury and not the rule. Often we have multiple data sets from multiple data providers, all of which may have different data quality levels of which we are unaware. We wanted an algorithm that could progressively assign different data quality weights/measures to each data set and evaluate the results from the algorithm to determine if perhaps this is an accurate set or an inaccurate one. Determining if the learned BN is of high quality is based on heuristics, or rules of thumb, for what a high quality BN normally looks like. In our experiments the heuristic was the degree of each node – how many dependencies it has to other nodes. It was found that in poorly learned networks, the degree of each node is either excessively small or large. Through studying a variety of networks, it was determined that the average degree of each node in a "good" network is 2.22. In the original DQ Algorithm, we scored the networks based on how close the average degree of the nodes in the created BN was to 2.22. While this worked well in our experiments, in future forms of the algorithm this heuristic could be anything – orientation of edges, common domain knowledge, etc.

After analyzing the PC algorithm and the effect the significance level has on the quality of the created BNs, we changed the DQ Method into a third form which incorporates a "roving" significance level for determining independence of nodes in the PC algorithm. Instead of progressively assigning different weights to the data sets, we assigned different significance levels – between 0.05 – 0.00005 – and then scored these BNs based on average degree of the node.

The psuedocode follows:

DQ Method

Part One:
Input from User: Data files, confidence level of each data file.
Create single data file from the imported files.
Total Confidence Level = Sum(all data file confidence levels)
Intermediate Number = Total Length of Data File/ Total Confidence Level
For each Data File
*Number of cases to Use = Intermediate Number * dataFile.confidenceLevel*
Add Number of Cases to Use
Using Hugin API
Create empty domain;
Learn structure(data file); //Hugin uses PC algorithm
Learn probabilities(data file); //Hugin uses EM algorithm
Export to .net file;
Output: Hugin .net file

Part Two:
Input from User: Data Files
For each data file
For each weight combination 1-5
Using Hugin API
Create empty domain;
Learn structure(data file); //Hugin uses PC algorithm
Learn probabilities(data file); //Hugin uses EM algorithm
Determine average degree per node of network;
Score = Absolute Value(2.22 – degree of network);
File to Output = Network with lowest score
Output: Path to Hugin .net file

Part Three:
Input from User: Data Files.
Create single data file from the imported files.
Set five different significance levels from ranging 0.05 – 0.00005
For each significance level
Using Hugin API
Create empty domain;
Learn structure(data file); //Hugin uses PC algorithm
Learn probabilities(data file); //Hugin uses EM algorithm
Export to .net file;
Determine average degree per node of network;
Score = Absolute Value(2.22 – degree of network);
File to Output = Network with lowest score

Output: Hugin .net file

As we discovered, the DQ Method Part Three was most effective, although in future studies we would like to try combining data set weights, significance levels, and new heuristics into one algorithm.

METHODS

After determining the effects of data inaccuracy on the PC algorithm and creating methods to deal with this problem, we tested our methods in several ways. First, we conducted initial tests as was explained in the Rationale and Purpose Section. These initial results were helpful in demonstrating the effects of inaccuracy on the PC algorithm, but did not serve to elevate one method over the others. Therefore we conducted a second set of tests in which we paired Gold Standard data with Category A3 and A5 data for Visit to Asia and Studfarm networks, and Gold Standard data with 0.5% inaccurate data for the ALARM network. We used these data sets to test the Do Nothing, Threshold, and DQ Part One and Two methods. In order to test the DQ Method Part Three we presented this method with one data set for Category A1, A3, and A5 data and compared the results to the Do Nothing method. For our tests with the Threshold method for Visit to Asia and Studfarm, our threshold was set at Category A3 data. This effectively means a data set with quality less than Category A3 was not admitted into the algorithm via the Threshold method. For the ALARM network the Threshold was set at 0.49% dirty data.

We show the results of the tests in three forms. First we show the raw results – number of correct edges, misoriented edges, false negative edges, and false positive edges. Then, using a causal interpretation of BNs, we combine the misoriented, false negative, and false positive edges together as incorrect edges. Lastly, we zero out the results using the Do Nothing method as a baseline. When interpreting the zeroed results, a positive value for correct edges (learned more correct edges than Do Nothing) and negative value for incorrect edges (learned fewer incorrect edges than Do Nothing) is optimal.

RESULTS

We will present the raw results and combined and zeroed results from Visit to Asia in Figures 4 and 5 respectively.

ASIA			Do Nothing Method			
Data Set	Quality #1	Quality #2	Correct Edges	Misoriented	False Negative	False Positive
1	Gold Standard	A3	4	4	0	15
2	Gold Standard	A5	4	4	0	14
			Threshold			
			Correct Edges	Misoriented	False Negative	False Positive
1	Gold Standard	A3	4	4	0	15
2	Gold Standard	A5	1	5	2	0
			DQ Method Part I			
			Correct Edges	Misoriented	False Negative	False Positive
1	Gold Standard	A3	6	2	0	17
2	Gold Standard	A5	7	1	0	16
			DQ Method Part II			
			Correct Edges	Misoriented	False Negative	False Positive
1	Gold Standard	A3	6	2	0	17
2	Gold Standard	A5	7	1	0	16

Figure 4: Visit to Asia Raw Results

ASIA			Do Nothing Method		Zeroed	
Data Set	Quality #1	Quality #2	Correct Edges	Incorrect Links	Correct Links	Incorrect Links
1	Gold Standard	A3	4	19	0	0
2	Gold Standard	A5	4	18	**0**	**0**
			Threshold		Zeroed	
			Correct Edges	Incorrect Links	Correct Links	Incorrect Links
1	Gold Standard	A3	4	19	0	0
2	Gold Standard	A5	1	7	-3	-11
			DQ Method Part I		Zeroed	
			Correct Edges	Incorrect Links	Correct Links	Incorrect Links
1	Gold Standard	A3	6	19	2	0
2	Gold Standard	A5	7	17	3	-1
			DQ Method Part II		Zeroed	
			Correct Edges	Incorrect Links	Correct Links	Incorrect Links
1	Gold Standard	A3	6	19	2	0
2	Gold Standard	A5	7	17	3	-1

Figure 5: Visit to Asia Combined and Zeroed Results

Studfarm results are similar, and in order to save space we will not present those here.
We will present the raw results and combined and zeroed results from ALARM in Figures 6 and 7 respectively.

ALARM			Do Nothing Method			
Data Set	Quality #1	Quality #2	Correct Edges	Misoriented	False Negative	False Positive
1	Gold Standard	5% dirty	30	15	0	25
			Threshold			
			Correct Edges	Misoriented	False Negative	False Positive
1	Gold Standard	5% dirty	37	6	2	0
			DQ Method Part I			
			Correct Edges	Misoriented	False Negative	False Positive
1	Gold Standard	5% dirty	34	11	0	53
			DQ Method Part II			
			Correct Edges	Misoriented	False Negative	False Positive
1	Gold Standard	5% dirty	30	25	15	0

Figure 6: ALARM Raw Results

ALARM			Do Nothing Method		Zeroed	
Data Set	Quality #1	Quality #2	Correct Edges	Incorrect Links	Correct Links	Incorrect Links
1	Gold Standard	5% dirty	30	40	**0**	**0**
			Threshold		Zeroed	
			Correct Edges	Incorrect Links	Correct Links	Incorrect Links
1	Gold Standard	5% dirty	37	8	7	-32
			DQ Method Part I		Zeroed	
			Correct Edges	Incorrect Links	Correct Links	Incorrect Links
1	Gold Standard	5% dirty	34	64	4	24
			DQ Method Part II		Zeroed	
			Correct Edges	Incorrect Links	Correct Links	Incorrect Links
1	Gold Standard	5% dirty	30	40	**0**	**0**

Figure 7: ALARM Combined and Zeroed Results

Finally, we present results for DQ algorithm Part Three for Visit to Asia in combined and zeroed form in Figures 8 and 9 respectively.

ASIA		Do Nothing	
Sign Level	Quality	Correct Edges	Incorrect Edges
0.05	A1	4	12
0.05	A3	4	13
0.05	A5	3	11
		DQ Method Part III	
Sign Level	Quality	Correct Edges	Incorrect Edges
0.0125	A1	3	7
0.003125	A3	2	7
0.000049	A5	2	11

Figure 8: DQ Method Part Three Combined Results, Visit to Asia

ASIA		Do Nothing	
Sign Level	Quality	Correct Edges	Incorrect Edges
0.05	A1	0	0
0.05	A3	0	0
0.05	A5	0	0
		DQ Method Part III	
Sign Level	Quality	Correct Edges	Incorrect Edges
0.0125	A1	-1	-5
0.003125	A3	-2	-6
0.000049	A5	-1	0

Figure 9: DQ Method Part Three Zeroed Results, Visit to Asia

DISCUSSIONS

We organize our discussion of our results by method.

The results of our testing of the Threshold method were not surprising. Because the method does not admit Category A5 data, the resulting BN has a structure more similar to the true BN. For Visit to Asia, this results in 11 less incorrect edges. It also results in learning 3 fewer correct edges. This is not surprising though when we consider how the PC algorithm works. It first creates a complete graph from

the nodes – it connects all nodes to each other. Then it seeks to eliminate those edges which are incorrect. The reason the Do Nothing method has more correct edges, is because it leaves so many edges in place. The Threshold method is operating with accurate data and is therefore able to remove edges that are incorrect, while also removing some edges that are correct. Also, looking at Figure 4 raw results, we actually see that only 2 of the edges were not existent, and 1 was misoriented. The results from the ALARM network tests are also promising, with the Threshold method finding 7 more correct edges and 32 fewer incorrect edges. The results from this method confirm our earlier finding that data quality is a very important factor in the outcomes of the algorithm, and show that incorporating data quality assessment information is useful and results in more robust BN structures.

The results for the DQ Method Part One and Two are also promising, although not as exciting as we had hoped. Recall that the method works by weighting high quality data sets over low quality sets. Therefore all data sets are used, but those of higher quality are weighted above those of lower quality. The results for both Visit to Asia and ALARM show that the DQ Method learns around 3 more correct edges than the Do Nothing method, but is not successful in removing more incorrect edges. As we show in our analysis of the PC algorithm, this is because of the algorithm's low tolerance of inaccurate data. Because we are leaving some low quality data in the data set, the algorithm is unable to remove false positive or extra edges and therefore creates structures which are not as robust as when we completely remove the inaccurate data. We therefore created a DQ Method Part Three which uses a second strategy. Instead of limiting or weighting the data sets that are used by the algorithm, we instead stretch the limits of what the PC algorithm believes is independence. In a sense, we stretch the parameters for what is a meaningful edge and what is not. A good illustration of how this works is to think of how we might target our marketing advertisements under high and low quality data. If we have high quality data we may be able to target a specific region of town for our grand opening – maybe even a specific subdivision in which we know the makeup of the residents. If our data is of low quality, we may only be able to limit our advertising region to a town, thus sending advertisements to residents who are uninterested in our product, and therefore wasting money. In the case of our DQ Method Part Three, we are widening the region we will target because we know our data is of low quality. Even though we may not be able to precisely locate a specific subdivision, at least we are not in the completely wrong town. By increasing the significance level, we are basically lowering the bar on what is considered a dependency and therefore removing more incorrect edges.

The results for DQ Method Part Three are shown in a slightly different format. In Figure 8 and 9 we see in the leftmost cell the significance level which generated the BN with the best score – in other words the significance level which was picked by the DQ Method (using the degree of node heuristic) to create the best BN structure. We then compare that structure to the Gold Standard and record the correct and incorrect edges. The results for DQ Method Part Three are very promising. First, we see that the lower significance level did indeed create more useful BNs than the Do Nothing method. Also, our average node degree heuristic was useful in helping to find the optimal significance level for each quality category and the combination of the two had very good results. Again we see that the algorithm created BNs with many fewer incorrect edges (5-6), but also a few less correct edges (1-2 fewer). As we stated above, this is a result of how the PC algorithm works, and while a few correct edges were missed when using this method, the reduction in incorrect edges leaves us with a BN that is more useful than that created by the Do Nothing method.

LIMITATIONS

As we stated in the introductory section of our paper, the fields of both Data Quality and Machine Learning are too extensive to exhaustively conduct our research. Therefore, we have limited our research to a specific component of Data Quality – accuracy, and a specific Machine Learning algorithm – the PC algorithm. In the future, we would like to expand our research in both fields, by testing the effects of

other data components on the PC algorithm, and also by demonstrating the effects of low quality data on other Machine Learning algorithms.

Because we were embarking on a new area of research, we also limited our test data sets to well known BNs. This allowed us to have a controlled test environment in which the data quality assessments were fixed (because we created the inaccuracies to begin with). Our upcoming research, however, will take our research in a more practical direction and allow for more robust testing. Through a grant from the National Institute of Justice we will be conducting a Data Quality Assessment of South Carolina's Sex Offender data using the TDQM methodology. After conducting these assessments we will then use our Threshold and DQ methods to examine the data sets using the PC algorithm. This will allow us to test our results on a subset of the Law Enforcement domain. It is our belief that having these assessments and using them as inputs to Machine Learning algorithms will create more useful tools for the Law Enforcement community.

CONCLUSIONS

In this work we have made several contributions. First, we have proven that data quality, specifically accuracy, has an enormous effect on the results and efficiency of BN learning algorithms – specifically the PC algorithm. While it is not surprising that inaccurate data limits the effectiveness of the learning algorithms, this is the first attempt we have found to prove that assumption. Second, based on our initial results, we have further analyzed the PC algorithm and have determined why it is so severely affected by inaccurate data. Again, while this may not be a surprising result, this is the first attempt to show an analysis of the problem and its cause within the PC algorithm. Third, we have developed several methods with which to lessen the effects of inaccurate data on the learning algorithms. These methods have been tested and show much promise for dealing with this problem. We are encouraged by these results and are looking forward to expanding our research to assess data sets from the Law Enforcement community.

REFERENCES

[1] Beinlich, I., H. Suermont, R. Chovez, and G. Cooper. The alarm monitoring system: A case study with two probabalistic inference techniques for belief networks. *Proceedings of the Second European Conference on Artificial Intelligence in Medical Care*, Springer-Verlag, 1989. 247-256.

[2] Cowell, R.G., Dawid, A.P., Lauritzen, S.L., Spiegelhalter, D.J. (1999). *Probabilistic Networks and Expert Systems*. New York, NY: Springer-Verlag.

[3] Jensen, F.V. (2001). *Bayesian Networks and Decision Graphs*. New York, NY: Springer-Verlag.

[4] Lee Y., D. Strong, B. Kahn, R. Wang. AIMQ: A Methodology for Information Quality Assessment. *Information and Management.* 2002. 133-146.

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

[6] 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.

[7] Pipino, L, D. Kopsco. Data Mining, Dirty Data, and Costs. *Proceedings of the 9th International OCnference on Information Quality*, Cambridge, MA, 2004. 164-169.

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

[9] Spirtes, P., Glymour, C., and Scheines, R. (2000). *Causation, Prediction, and Search.* MIT Press, Cambridge, Massachusetts.

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